About this Document

This document provides a comprehensive description of how to solve classification and regression problems using the BigML Dashboard. BigML models, ensembles, linear regressions, logistic regressions, deepnets, and fusions are covered in detail. Learn how to use the BigML Dashboard to configure, visualize, and interpret these supervised predictive models and use them to create evaluations of their performance and real-time predictions or batch predictions.

This document assumes that you are familiar with:

- Sources with the BigML Dashboard. The BigML Team. June 2016. [11]
- Datasets with the BigML Dashboard. The BigML Team. June 2016. [9]

To learn how to use the BigML Dashboard to build time series and unsupervised models read:

- Time Series with the BigML Dashboard. The BigML Team. July 2017. [12]
- Cluster Analysis with the BigML Dashboard. The BigML Team. June 2016. [8]
- Anomaly Detection with the BigML Dashboard. The BigML Team. June 2016. [6]
- Association Discovery with the BigML Dashboard. The BigML Team. June 2016. [7]
- Topic Modeling with the BigML Dashboard. The BigML Team. November 2016. [13]
## Contents

1 Models ................................................................. 1
   1.1 Introduction ...................................................... 1
   1.2 Understanding BigML Models .................................... 3
       1.2.1 Streaming Model ........................................... 3
       1.2.2 In-Memory Model ......................................... 4
       1.2.3 Scoring and Splitting .................................... 4
           1.2.3.1 Early Splitting ..................................... 4
       1.2.4 Pruning .................................................... 4
       1.2.5 Field Importance ......................................... 5
       1.2.6 Confidence and Probability .............................. 5
           1.2.6.1 Confidence ........................................... 5
               1.2.6.1.1 Formulation .................................. 5
               1.2.6.1.2 Interpretation ............................... 6
           1.2.6.2 Probability ......................................... 6
       1.2.7 Expected Error ............................................. 7
           1.2.7.1 Formulation ........................................... 7
           1.2.7.2 Interpretation ..................................... 8
       1.2.8 Models with Images ....................................... 8
   1.3 Creating Models with 1-Click .................................. 10
   1.4 Model Configuration Options ................................... 10
       1.4.1 Objective Field .......................................... 12
       1.4.2 Automatic Optimization ................................ 12
           1.4.2.1 Training duration .................................. 14
           1.4.2.2 Model candidates ................................ 14
       1.4.3 Pruning .................................................... 14
       1.4.4 Missing Splits ............................................ 15
       1.4.5 Node Threshold ........................................... 15
       1.4.6 Weight Options ........................................... 15
           1.4.6.1 Balance Objective .................................. 16
           1.4.6.2 Objective Weights ................................ 16
           1.4.6.3 Weight Field ....................................... 16
       1.4.7 Sampling Options ......................................... 17
           1.4.7.1 Rate .................................................. 17
           1.4.7.2 Range ................................................ 18
           1.4.7.3 Sampling ............................................ 18
           1.4.7.4 Replacement ....................................... 18
           1.4.7.5 Out of Bag .......................................... 18
       1.4.8 Advanced Ordering ........................................ 19
           1.4.8.1 Deterministic Shuffling ............................. 19
           1.4.8.2 Linear Shuffling .................................... 19
           1.4.8.3 Random Shuffling .................................. 19
   1.4.9 Creating Models with Configured Options .................. 19
   1.4.10 API Request Preview ....................................... 20
## 3 Linear Regressions

### 3.1 Introduction

### 3.2 Understanding Linear Regressions

#### 3.2.1 Input Field Transformations

#### 3.2.2 Missing Values

#### 3.2.3 Number of Predictors

#### 3.2.4 Ill-Conditioned Problems

### 3.3 Creating Linear Regressions with 1-Click

### 2.1 Introduction

### 2.2 Creating Ensemble Predictions

#### 2.2.1 Predict

##### 2.2.1.1 Local Predictions

##### 2.2.1.2 Predictions with Images

#### 2.2.2 Batch Predictions

##### 2.2.2.1 Batch Prediction with Images

### 2.3 Configuring Ensemble Predictions

#### 2.3.1 Missing Strategies

#### 2.3.2 Combine single tree predictions: probability, confidence or votes

#### 2.3.3 Probability, confidence, and votes thresholds

#### 2.3.4 Default Numeric Value

### 2.4 Visualizing Ensemble Predictions

#### 2.4.1 Single Predictions

##### 2.4.1.1 Prediction explanation

#### 2.4.2 Batch Predictions

##### 2.4.2.1 Output File

##### 2.4.2.2 Output Dataset

##### 2.4.2.3 Batch Prediction 1-Click Actions

### 2.5 Consuming Ensemble Predictions

#### 2.5.1 Using Ensemble Predictions via the BigML API

#### 2.5.2 Using Ensemble Predictions via the BigML Bindings

### 2.6 Descriptive Information

#### 2.6.1 Name

#### 2.6.2 Description

#### 2.6.3 Category

#### 2.6.4 Tags

### 2.7 Consuming Ensembles

#### 2.7.1 Exporting and Downloading Ensembles

#### 2.7.2 Using Ensembles Via the BigML API

#### 2.7.3 Using Ensembles Via the BigML Bindings

### 2.8 Ensemble Limits

### 2.9 Descriptive Information

#### 2.9.1 Name

#### 2.9.2 Description

#### 2.9.3 Category

#### 2.9.4 Tags

#### 2.9.5 Counters

### 2.10 Ensemble Privacy

### 2.11 Moving Ensembles to Another Project

### 2.12 Stopping Ensemble Creation

### 2.13 Deleting Ensembles

### 2.14 Takeaways

---

Copyright © 2023, BigML, Inc.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>Linear Regression Configuration Options</td>
<td>174</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Objective Field</td>
<td>174</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Automatic Optimization</td>
<td></td>
</tr>
<tr>
<td>3.4.2.1</td>
<td>Training duration</td>
<td>177</td>
</tr>
<tr>
<td>3.4.2.2</td>
<td>Linear regression candidates</td>
<td>177</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Default Numeric Value</td>
<td>178</td>
</tr>
<tr>
<td>3.4.4</td>
<td>Weights</td>
<td>179</td>
</tr>
<tr>
<td>3.4.4.1</td>
<td>Weight Field</td>
<td>179</td>
</tr>
<tr>
<td>3.4.5</td>
<td>Bias</td>
<td></td>
</tr>
<tr>
<td>3.4.6</td>
<td>Field Codings</td>
<td>180</td>
</tr>
<tr>
<td>3.4.6.1</td>
<td>Dummy Coding</td>
<td>180</td>
</tr>
<tr>
<td>3.4.6.2</td>
<td>Contrast Coding</td>
<td>185</td>
</tr>
<tr>
<td>3.4.6.3</td>
<td>Other Coding</td>
<td>192</td>
</tr>
<tr>
<td>3.4.7</td>
<td>Sampling Options</td>
<td>198</td>
</tr>
<tr>
<td>3.4.7.1</td>
<td>Rate</td>
<td>198</td>
</tr>
<tr>
<td>3.4.7.2</td>
<td>Range</td>
<td>198</td>
</tr>
<tr>
<td>3.4.7.3</td>
<td>Sampling</td>
<td>198</td>
</tr>
<tr>
<td>3.4.7.4</td>
<td>Replacement</td>
<td>198</td>
</tr>
<tr>
<td>3.4.7.5</td>
<td>Out of bag</td>
<td></td>
</tr>
<tr>
<td>3.4.8</td>
<td>Advanced Ordering</td>
<td>199</td>
</tr>
<tr>
<td>3.4.8.1</td>
<td>Deterministic Shuffling</td>
<td>199</td>
</tr>
<tr>
<td>3.4.8.2</td>
<td>Linear Shuffling</td>
<td>199</td>
</tr>
<tr>
<td>3.4.8.3</td>
<td>Random Shuffling</td>
<td>199</td>
</tr>
<tr>
<td>3.4.9</td>
<td>Creating Linear Regressions with Configured Options</td>
<td>200</td>
</tr>
<tr>
<td>3.4.10</td>
<td>API Request Preview</td>
<td>201</td>
</tr>
<tr>
<td>3.5</td>
<td>Visualizing Linear Regressions</td>
<td></td>
</tr>
<tr>
<td>3.5.1</td>
<td>Linear Regression Chart</td>
<td>202</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Coefficient Table</td>
<td>207</td>
</tr>
<tr>
<td>3.6</td>
<td>Linear Regression Predictions</td>
<td>214</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Introduction</td>
<td>214</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Creating Linear Regression Predictions</td>
<td>215</td>
</tr>
<tr>
<td>3.6.2.1</td>
<td>Predict</td>
<td>215</td>
</tr>
<tr>
<td>3.6.2.2</td>
<td>Batch Predictions</td>
<td>219</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Configuring Linear Regression Predictions</td>
<td>224</td>
</tr>
<tr>
<td>3.6.3.1</td>
<td>Default Numeric Value</td>
<td>224</td>
</tr>
<tr>
<td>3.6.3.2</td>
<td>Excluded Fields</td>
<td>225</td>
</tr>
<tr>
<td>3.6.3.3</td>
<td>Fields Mapping</td>
<td>226</td>
</tr>
<tr>
<td>3.6.3.4</td>
<td>Output Settings</td>
<td>227</td>
</tr>
<tr>
<td>3.6.3.5</td>
<td>Prediction explanation</td>
<td>228</td>
</tr>
<tr>
<td>3.6.4</td>
<td>Consuming Linear Regression Predictions</td>
<td>230</td>
</tr>
<tr>
<td>3.6.4.1</td>
<td>Local Predictions</td>
<td>230</td>
</tr>
<tr>
<td>3.6.4.2</td>
<td>Using Linear Regression Predictions via the BigML API</td>
<td>230</td>
</tr>
<tr>
<td>3.6.4.3</td>
<td>Using Linear Regression Predictions via BigML Bindings</td>
<td>230</td>
</tr>
<tr>
<td>3.6.5</td>
<td>Descriptive Information</td>
<td>230</td>
</tr>
<tr>
<td>3.6.5.1</td>
<td>Name</td>
<td>231</td>
</tr>
<tr>
<td>3.6.5.2</td>
<td>Description</td>
<td>231</td>
</tr>
<tr>
<td>3.6.5.3</td>
<td>Category</td>
<td>232</td>
</tr>
<tr>
<td>3.6.5.4</td>
<td>Tags</td>
<td>232</td>
</tr>
<tr>
<td>3.6.6</td>
<td>Linear Regression Predictions Privacy</td>
<td>232</td>
</tr>
<tr>
<td>3.6.7</td>
<td>Moving Linear Regression Predictions to Another Project</td>
<td>233</td>
</tr>
<tr>
<td>3.6.8</td>
<td>Stopping Linear Regression Predictions</td>
<td>233</td>
</tr>
<tr>
<td>3.6.9</td>
<td>Deleting Linear Regression Predictions</td>
<td>233</td>
</tr>
<tr>
<td>3.7</td>
<td>Consuming Linear Regressions</td>
<td>235</td>
</tr>
<tr>
<td>3.7.1</td>
<td>Downloading Linear Regressions</td>
<td>235</td>
</tr>
<tr>
<td>3.7.2</td>
<td>Using Linear Regressions Via the BigML API</td>
<td>235</td>
</tr>
<tr>
<td>3.7.3</td>
<td>Using Linear Regressions Via the BigML Bindings</td>
<td>236</td>
</tr>
<tr>
<td>3.8</td>
<td>Linear Regression Limits</td>
<td>236</td>
</tr>
<tr>
<td>3.8.0.1</td>
<td>Field Limits</td>
<td>236</td>
</tr>
</tbody>
</table>
4 Logistic Regressions

4.1 Introduction

4.2 Understanding Logistic Regressions
  4.2.1 Input Field Transformations
  4.2.2 Missing Values
  4.2.3 Logistic Regressions with Images

4.3 Creating Logistic Regressions with 1-Click

4.4 Logistic Regression Configuration Options
  4.4.1 Objective Field
  4.4.2 Automatic Optimization
    4.4.2.1 Training duration
    4.4.2.2 Logistic regression candidates
  4.4.3 Weights
    4.4.3.1 Balance Objective
    4.4.3.2 Objective Weights
    4.4.3.3 Weight Field
  4.4.4 Default Numeric Value
  4.4.5 Missing Numerics
  4.4.6 Eps
  4.4.7 Stats
  4.4.8 Bias and Auto-Scaling
  4.4.9 Regularization
  4.4.10 Field Codings
    4.4.10.1 One-hot Coding
    4.4.10.2 Dummy Coding
    4.4.10.3 Contrast Coding
    4.4.10.4 Other Coding
  4.4.11 Sampling Options
    4.4.11.1 Rate
    4.4.11.2 Range
    4.4.11.3 Sampling
    4.4.11.4 Replacement
    4.4.11.5 Out of bag
  4.4.12 Creating Logistic Regressions with Configured Options
  4.4.13 API Request Preview

4.5 Visualizing Logistic Regressions
  4.5.1 Logistic Regression Chart
  4.5.2 Coefficient Table
    4.5.2.1 Coefficient Table with Stats Computation

4.6 Logistic Regression Predictions
  4.6.1 Introduction
  4.6.2 Creating Logistic Regression Predictions
    4.6.2.1 Predict
      4.6.2.1.1 Logistic Regression Prediction with Images
5 Deepnets

5.1 Introduction ................................................... 346
5.2 Understanding Deepnets ........................................ 347
  5.2.1 Convolutional Neural Network ............................. 348
  5.2.2 Automatic Parameter Optimization ....................... 349
  5.2.3 Deepnets Use Cases ........................................ 349
  5.2.4 Missing Values ........................................... 350
5.3 Creating Deepnets with 1-Click ................................ 350
5.4 Deepnet Configuration Options ................................ 351
  5.4.1 Objective Field .......................................... 351
  5.4.2 Automatic Parameter Optimization ........................ 352
  5.4.3 Default Numeric Value .................................... 354

4.8 Logistic Regression Limits .................................... 335
  4.8.0.1 Field Limits ........................................... 335
  4.8.1 Chart Limits ............................................... 335
  4.8.2 Coefficient Table Limits .................................. 336
4.9 Descriptive Information ......................................... 337
  4.9.1 Logistic Regression Name .................................. 337
  4.9.2 Description ................................................ 338
  4.9.3 Category .................................................. 338
  4.9.4 Tags ...................................................... 339
  4.9.5 Counters .................................................. 339
4.10 Logistic Regression Privacy .................................... 340
4.11 Moving Logistic Regressions to Another Project ............. 340
4.12 Stopping Logistic Regressions ................................ 341
4.13 Deleting Logistic Regressions ................................ 342
4.14 Takeaways ....................................................... 344

4.6.2.2 Batch Predictions ........................................ 310
  4.6.2.2.1 Batch Prediction with Images ......................... 315
4.6.3 Configuring Logistic Regression Predictions ................ 316
  4.6.3.1 Probability threshold .................................... 316
  4.6.3.2 Default Numeric Value .................................. 318
  4.6.3.3 Fields Mapping .......................................... 318
  4.6.3.4 Output Settings ......................................... 319
4.6.4 Visualizing Logistic Regression Predictions ................ 320
  4.6.4.1 Single Predictions ....................................... 320
  4.6.4.1.1 Local Predictions .................................... 324
  4.6.4.1.2 Prediction explanation .............................. 324
  4.6.4.2 Batch Prediction ......................................... 326
  4.6.4.2.1 Output CSV file ...................................... 326
  4.6.4.2.2 Output Dataset ........................................ 327
4.6.5 Consuming Logistic Regression Predictions ................ 328
  4.6.5.1 Using Logistic Regression Predictions via the BigML API 328
  4.6.5.2 Using Logistic Regression Predictions via BigML Bindings 328
4.6.6 Descriptive Information ...................................... 329
  4.6.6.1 Name .................................................... 329
  4.6.6.2 Description .............................................. 329
  4.6.6.3 Category ................................................ 330
  4.6.6.4 Tags .................................................... 330
4.6.7 Logistic Regression Predictions Privacy ..................... 330
4.6.8 Moving Logistic Regression Predictions to Another Project 331
4.6.9 Stopping Logistic Regression Predictions ................... 331
4.6.10 Deleting Logistic Regression Predictions .................. 332
4.7 Consuming Logistic Regressions ................................ 333
  4.7.1 Downloading Logistic Regressions ......................... 333
  4.7.2 Using Logistic Regressions Via the BigML API ............. 334
  4.7.3 Using Logistic Regressions Via the BigML Bindings ........ 334
4.8 Logistic Regression Limits ..................................... 335
  4.8.0.1 Field Limits ........................................... 335
  4.8.1 Chart Limits ............................................... 335
  4.8.2 Coefficient Table Limits .................................. 336
5 Deepnets .................................................................. 346
5.4.4 Missing Numerics .......................................................... 354
5.4.5 Training Duration .......................................................... 355
5.4.6 Maximum Iterations ......................................................... 355
5.4.7 Network Architecture ...................................................... 356
  5.4.7.1 Hidden Layers ......................................................... 356
  5.4.7.2 Learn Residuals ....................................................... 358
  5.4.7.3 Batch Normalization .............................................. 358
  5.4.7.4 Tree Embedding ...................................................... 359
5.4.8 Algorithm ................................................................. 360
  5.4.8.1 Gradient Descent Algorithm .................................... 360
  5.4.8.2 Learning Rate ....................................................... 365
  5.4.8.3 Dropout Rate ........................................................ 366
  5.4.8.4 Seed ................................................................. 367
5.4.9 Weights ................................................................. 368
  5.4.9.1 Balance Objective ................................................. 369
  5.4.9.2 Objective Weights ............................................... 369
  5.4.9.3 Weight field ....................................................... 369
5.4.10 Sampling Options ....................................................... 370
  5.4.10.1 Rate ............................................................... 370
  5.4.10.2 Range ............................................................. 370
  5.4.10.3 Sampling .......................................................... 370
  5.4.10.4 Replacement ...................................................... 370
  5.4.10.5 Out of bag ......................................................... 370
5.4.11 Creating Deepnets with Configured Options ....................... 371
5.4.12 API Request Preview .................................................. 372
5.5 Visualizing Deepnets ...................................................... 373
  5.5.1 Partial Dependence Plot ............................................ 373
  5.5.2 Image Deepnet Page .................................................. 378
    5.5.2.1 Image Deepnet Page - Classification ....................... 378
    5.5.2.2 Image Deepnet Page - Regression ......................... 384
  5.5.3 Summary Report ...................................................... 390
    5.5.3.1 Field Importances ............................................. 390
    5.5.3.2 Summary ....................................................... 390
5.6 Deepnet Predictions ...................................................... 391
  5.6.1 Introduction .......................................................... 391
  5.6.2 Creating Deepnet Predictions ..................................... 393
    5.6.2.1 Predict .......................................................... 393
    5.6.2.2 Deepnet Prediction with Images ............................. 397
    5.6.2.3 Batch Predictions .............................................. 402
      5.6.2.3.1 Batch Predictions with Images ......................... 406
  5.6.3 Configuring Deepnet Predictions ................................ 407
    5.6.3.1 Probability threshold ........................................ 407
    5.6.3.2 Default Numeric Value ...................................... 410
    5.6.3.3 Fields Mapping ............................................... 410
    5.6.3.4 Output Settings ............................................... 411
  5.6.4 Visualizing Deepnet Predictions ................................ 412
    5.6.4.1 Single Predictions ............................................ 412
      5.6.4.1.1 Prediction explanation .................................. 413
    5.6.4.2 Batch Prediction ............................................... 415
      5.6.4.2.1 Output CSV file .......................................... 415
      5.6.4.2.2 Output Dataset ........................................... 416
  5.6.5 Consuming Deepnet Predictions ................................ 417
    5.6.5.1 Using Deepnet Predictions via the BigML API ............ 417
    5.6.5.2 Using Deepnet Predictions via BigML Bindings .......... 417
  5.6.6 Descriptive Information .......................................... 418
    5.6.6.1 Name ........................................................... 418
    5.6.6.2 Description .................................................... 419
    5.6.6.3 Category ...................................................... 419
6 Fusions

6.1 Introduction .................................................. 434
6.2 Understanding Fusions ........................................ 435
  6.2.1 Fusion Objective Field ................................... 436
  6.2.2 Fusion Field Importances ................................. 436
6.3 Creating Fusions ................................................ 436
6.4 Fusion Configuration Options .................................. 440
  6.4.1 Models .................................................. 440
  6.4.2 Objective Field .......................................... 441
  6.4.3 Weights ................................................ 443
  6.4.4 Creating Fusions with Configured Options .............. 444
  6.4.5 API Request Preview ..................................... 445
6.5 Visualizing Fusions ............................................. 445
  6.5.1 Fusion Partial Dependence Plot ........................ 446
  6.5.2 Fusion Model List ........................................ 451
6.6 Fusion Predictions .............................................. 452
  6.6.1 Introduction .............................................. 452
  6.6.2 Creating Fusion Predictions .............................. 454
    6.6.2.1 Predict ............................................. 454
    6.6.2.2 Batch Predictions ................................... 457
  6.6.3 Configuring Fusion Predictions ........................... 460
    6.6.3.1 Missing Strategies .................................. 460
    6.6.3.2 Probability threshold ................................ 461
    6.6.3.3 Default Numeric Value ............................... 465
    6.6.3.4 Excluded Fields ..................................... 466
    6.6.3.5 Fields Mapping ...................................... 467
    6.6.3.6 Output Settings ..................................... 467
  6.6.4 Visualizing Fusion Predictions ............................ 469
    6.6.4.1 Single Predictions .................................. 469
    6.6.4.1.1 Prediction explanation ............................ 470
    6.6.4.2 Batch Prediction ..................................... 472
    6.6.4.2.1 Output CSV file .................................. 472
    6.6.4.2.2 Output Dataset .................................. 473
  6.6.5 Consuming Fusion Predictions ............................. 475

5.8 Deepnet Limits ................................................. 424
  5.8.0.1 Field Limits ........................................... 424
  5.8.1 PDP Limits ................................................ 424
5.9 Descriptive Information ......................................... 424
  5.9.1 Deepnet Name .............................................. 425
  5.9.2 Deepnet Description ....................................... 425
  5.9.3 Deepnet Category .......................................... 426
  5.9.4 Deepnet Tags .............................................. 427
  5.9.5 Deepnet Counters ......................................... 427
5.10 Deepnet Privacy ................................................ 428
5.11 Moving Deepnets to Another Project .......................... 428
5.12 Stopping Deepnets ............................................. 429
5.13 Deleting Deepnets ............................................. 430
5.14 Takeaways .................................................... 432

5.7 Consuming Deepnet ............................................. 422
  5.7.1 Downloading Deepnet ..................................... 422
  5.7.2 Using Deepnets Via the BigML API ....................... 423
  5.7.3 Using Deepnets Via the BigML Bindings .................. 423
5.8 Deepnet Limits ................................................. 424
  5.8.0.1 Field Limits ........................................... 424
  5.8.1 PDP Limits ................................................ 424
5.9 Descriptive Information ......................................... 424
  5.9.1 Deepnet Name .............................................. 425
  5.9.2 Deepnet Description ....................................... 425
  5.9.3 Deepnet Category .......................................... 426
  5.9.4 Deepnet Tags .............................................. 427
  5.9.5 Deepnet Counters ......................................... 427
5.10 Deepnet Privacy ................................................ 428
5.11 Moving Deepnets to Another Project .......................... 428
5.12 Stopping Deepnets ............................................. 429
5.13 Deleting Deepnets ............................................. 430
5.14 Takeaways .................................................... 432
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3.4 Deepnet Evaluations</td>
<td>523</td>
</tr>
<tr>
<td>7.3.5 Fusion Evaluations</td>
<td>525</td>
</tr>
<tr>
<td>7.3.6 Cross-Validation Evaluations</td>
<td>528</td>
</tr>
<tr>
<td>7.4 Evaluation Configuration Options</td>
<td>536</td>
</tr>
<tr>
<td>7.4.1 Missing Strategies</td>
<td>537</td>
</tr>
<tr>
<td>7.4.2 Select the Voting Strategy for Decision Forests</td>
<td>538</td>
</tr>
<tr>
<td>7.4.3 Fields Mapping</td>
<td>539</td>
</tr>
<tr>
<td>7.4.4 Sampling Your Dataset</td>
<td>540</td>
</tr>
<tr>
<td>7.4.4.1 Rate</td>
<td>540</td>
</tr>
<tr>
<td>7.4.4.2 Range</td>
<td>540</td>
</tr>
<tr>
<td>7.4.4.3 Sampling</td>
<td>540</td>
</tr>
<tr>
<td>7.4.4.4 Replacement</td>
<td>540</td>
</tr>
<tr>
<td>7.4.4.5 Out of bag</td>
<td>541</td>
</tr>
<tr>
<td>7.4.5 Cross-Validation Configuration Options</td>
<td>541</td>
</tr>
<tr>
<td>7.4.5.1 Basic 5-fold cross-validation</td>
<td>541</td>
</tr>
<tr>
<td>7.4.5.2 Model's k-fold cross-validation</td>
<td>542</td>
</tr>
<tr>
<td>7.4.5.3 Ensemble's k-fold cross-validation</td>
<td>543</td>
</tr>
<tr>
<td>7.5 Visualizing Evaluations</td>
<td>544</td>
</tr>
<tr>
<td>7.5.1 Original Resources Information</td>
<td>545</td>
</tr>
<tr>
<td>7.5.2 Classification Evaluations</td>
<td>546</td>
</tr>
<tr>
<td>7.5.2.1 General Evaluation</td>
<td>546</td>
</tr>
<tr>
<td>7.5.2.2 Evaluation curves</td>
<td>550</td>
</tr>
<tr>
<td>7.5.3 Regression Evaluations</td>
<td>551</td>
</tr>
<tr>
<td>7.5.4 Cross-Validation Visualization</td>
<td>552</td>
</tr>
<tr>
<td>7.6 Evaluation Comparison</td>
<td>554</td>
</tr>
<tr>
<td>7.6.1 Compare Evaluations Side By Side</td>
<td>554</td>
</tr>
<tr>
<td>7.6.2 Compare Multiple Evaluations</td>
<td>556</td>
</tr>
<tr>
<td>7.6.2.1 Exporting Evaluation Comparison</td>
<td>563</td>
</tr>
<tr>
<td>7.7 Consuming Evaluations</td>
<td>564</td>
</tr>
<tr>
<td>7.7.1 Using Evaluations Via the BigML API</td>
<td>564</td>
</tr>
<tr>
<td>7.7.2 Using Evaluations Via the BigML Bindings</td>
<td>565</td>
</tr>
<tr>
<td>7.8 Evaluation Limits</td>
<td>565</td>
</tr>
<tr>
<td>7.9 Descriptive Information</td>
<td>565</td>
</tr>
<tr>
<td>7.9.1 Evaluation Name</td>
<td>566</td>
</tr>
<tr>
<td>7.9.2 Evaluation Description</td>
<td>566</td>
</tr>
<tr>
<td>7.9.3 Evaluation Category</td>
<td>566</td>
</tr>
<tr>
<td>7.9.4 Evaluation Tags</td>
<td>567</td>
</tr>
<tr>
<td>7.10 Evaluation Privacy</td>
<td>567</td>
</tr>
<tr>
<td>7.11 Moving Evaluations to Another Project</td>
<td>568</td>
</tr>
<tr>
<td>7.12 Stopping Evaluation Creation</td>
<td>569</td>
</tr>
<tr>
<td>7.13 Deleting Evaluation</td>
<td>570</td>
</tr>
<tr>
<td>7.14 Takeaways</td>
<td>571</td>
</tr>
</tbody>
</table>

List of Figures 573

List of Tables 588

Glossary 589

References 593
Models

1.1 Introduction

There are multiple Machine Learning problems that require a model to predict an output variable (objective field in BigML parlance) given a number of input variables (input fields in BigML). These problems can be divided into classification and regression problems, depending on the data type of the objective field:

• **Classification**: when the objective field is categorical. For these problems, a Machine Learning algorithm is used to build a model that predicts a category (label or class) for a new example (instance). That is, it "classifies" new instances into a given set of categories (or discrete values). For example, "true or false", "fraud or not fraud", "high risk, low risk or medium risk", etc. There can be hundreds of different categories.

• **Regression**: when the objective field is numeric. For these problems, a Machine Learning algorithm is used to build a model that predicts a continuous value. That is, given the fields that define a new instance the model predicts a real number. For example, "the price of a house", "the number of units sold for a product", "the potential revenue of a lead", "the number of hours until next system failure", etc.

Both classification and regression problems can be solved using supervised Machine Learning techniques. They are called supervised in the sense that the values of the output variable has either been provided by a human expert (e.g., the patient had been diagnosed with diabetes or not) or by a deterministic automated process (e.g., customers who did not pay their fees in the last three months are labeled as "delinquent"). The objective field values along with the input fields need to be collected for each instance in a structured dataset that is used to train the model. The algorithms learn a predictive model that maps your input data to a predicted objective field value.

A BigML Model uses a proprietary decision tree algorithm based on the Classification and Regression Trees (CART) algorithm proposed by Leo Breiman. Section 1.2 explains in detail BigML models implementation and interpretation.

This chapter contains comprehensive description of BigML models including how they can be created with 1-click (Section 1.3), all configuration options available (Section 1.4), and the different visualizations provided by BigML (Section 1.5). Once you create a model, you can get a report for each field importance (Subsection 1.6.1). See Section 1.7 for an explanation of how models can be used to make predictions. You can also export your models in different formats to make local predictions faster at no cost (Section 1.8). The process to evaluate your model’s predictive performance in BigML is explained in a different chapter (Chapter 7).

In BigML, the third tab of the main menu of your Dashboard allows you to list all your available models. In the model list view (Figure 1.2), you can see, for each model, the dataset it was created from, the model’s Name, Type (either classification or regression), Objective, Age (time elapsed since it was created), Size, and number of predictions, batch predictions, or evaluations that have been created.
using that model. The SEARCH menu option in the top right corner of the ensemble list view allows you to search your models by name.

When you first create an account at BigML, or every time that you start a new project, your list view for models will be empty. (See Figure 1.2.)

Finally, in Figure 1.3 you can see the icon used to represent a model in BigML.
1.2 Understanding BigML Models

This section describes internal details about BigML models. Besides being useful to better understand how BigML implements them and the optimizations it applies to improve performance, it will also provide the foundations to understand BigML models’ configuration options. You can safely skip this section on first read.

Models are built by splitting the data into partitions so each of them maximizes the information gain for classification models or minimizes the mean squared error for regression models. Each partition, i.e., split, has an associated predicate that defines it, e.g., balance < 1,000. The process of partitioning the instances in a dataset is recursive, i.e., partitions are themselves split into smaller ones, thus forming a hierarchy of partitions with their associated predicates. At each step of this process, a number of summary statistics is also computed, such as how many instances belong partition, its confidence, etc.

BigML uses a proprietary algorithm for models that produces Classification And Regression Tree (CART) style decision trees. BigML algorithm adapts itself to the dataset you are learning from to provide optimal performance. Indeed, for datasets that have a large number of instances, BigML will use “streaming”, i.e., it will process the data instances in chunks to reduce the memory footprint it requires. Alternatively, for datasets that have a large number of fields, BigML tends to load the whole dataset into memory.

The key points of BigML streaming algorithm (stree) are listed below:

- Support for large datasets
- Multi-core parallelism or multi-machine distribution
- Anytime algorithm with frequent updates

The key points of BigML in-memory algorithm (mtree) are listed below:

- Support for datasets with a large number of fields
- Faster than stree, albeit at the cost of needing more memory
- Anytime algorithm with frequent updates

1.2.1 Streaming Model

Inspired by algorithms by Ben-Haim and Tyree, BigML models are grown breadth-first. To grow the next generation of an stree, training instances are run through the tree and summary statistics are collected in each leaf. The summary statistics are then used to choose new splits for each leaf.

The main advantage of growing in this fashion is that an stree does not need to keep the training data in memory. Instead, the memory footprint for an stree is dependent on the size of the tree and the number of input features in the training data.

The summaries collected by an stree over a set of training instances may be merged with summaries from a separate block of training instances. This feature allows the stree summary collection phase to be easily parallelized or distributed.

Traditionally, CART trees consider every training instance that reaches a leaf when determining how to split. While an stree may be built similarly, it can also generate a split early using only a sample of the training data. stree attempts to detect when an early split is safe by calculating when the summary statistics have “settled”. Early splitting requires that the training instances are shuffled beforehand. (See Subsection 1.4.8.)

---

1 https://en.wikipedia.org/wiki/Information_gain_in_decision_trees
2 https://en.wikipedia.org/wiki/Mean_squared_error
3 https://en.wikipedia.org/wiki/Decision_tree_learning#Types
4 http://en.wikipedia.org/wiki/Anytime_algorithm
5 http://en.wikipedia.org/wiki/Anytime_algorithm
7 https://en.wikipedia.org/wiki/Breadth-first_search
1.2.2 In-Memory Model

Unlike \texttt{stree}, which grows an entire new tree layer on every iteration (a breadth first\footnote{https://en.wikipedia.org/wiki/Breadth-first_search} expansion), \texttt{mtree} grows the tree one split at a time. At each iteration, \texttt{mtree} chooses the split that looks to improve the tree the most. So the tree grows more like an A* search\footnote{https://en.wikipedia.org/wiki/A*_search_algorithm} than a breadth-first search.

\texttt{mtree} makes it possible to cap the overall size of BigML trees, so this approach helps prevent defining splits on relatively unimportant parts of the tree.

When choosing the best candidate split, \texttt{mtree} uses the reduction in squared error\footnote{https://en.wikipedia.org/wiki/Mean_squared_error} for regression trees. For classification trees, we use a split's information gain\footnote{https://en.wikipedia.org/wiki/Decision_tree_learning#Information_gain} multiplied by its population. These scores proved better than reusing pruning error estimates. (See also Subsection 1.4.3.)

As expected, this model growth technique outperforms the breadth-first approach for similarly sized trees.

With \texttt{stree}, the node threshold option (see Subsection 1.4.5) sets the minimum number of nodes in the tree, but that threshold could be exceeded. A threshold of 255 nodes could result in a tree sized anywhere between 255 and 509 nodes. However, asking for an \texttt{mtree} with 255 nodes will produce a tree with 255 nodes.

1.2.3 Scoring and Splitting

To choose a split for a leaf node, \texttt{stree} picks the best split for each input field and then selects the best of those candidates.

For categorical fields a candidate split is scored for every category, with a predicate in the form fieldX == "some_category". Along with the binary splits, \texttt{stree} scores an exploded split where every category is given its own child node.

Numeric splits are always binary with a predicate of the form fieldX >= 42. In traditional CART implementations, a split is considered between every two adjacent data points. Since \texttt{stree} does not keep the training data in memory, we use the summary histogram\footnote{https://en.wikipedia.org/wiki/Summary_histogram} instead. The median between every pair of adjacent histogram bins becomes a candidate split.

1.2.3.1 Early Splitting

A node in an \texttt{stree} can grow for one of two reasons. First, when the node collects summary information over the entire dataset it grows, as there is no more information to help inform the split. Secondly, a node can grow after collecting summary information over just a portion of the training data. This is the early split.

The core assumption for early splitting is that a sample of the dataset can often provide enough information to pick a good split. This is commonly the case for splits near the top of the tree. \texttt{stree} detects when enough information has been collected for an early split.

To find the similarity, \texttt{stree} calculates split scores for each field. The scores are normalized across all fields in the field summary set. At this point \texttt{stree} has two normalized score distributions, one for each field summary set. The L1 distance\footnote{https://en.wikipedia.org/wiki/Taxicab_geometry} between the distributions produces the similarity metric. A score of 0 means an exact match while a score of 1 means complete disagreement.

1.2.4 Pruning

As mentioned, models are trained by recursively partitioning the data, i.e., splitting it in two parts and then splitting each of them again in two and so on. This process goes on and on unless a stopping criteria is used. If a model grows too much, it ends up generating rules on minutiae and losing the ability...
to generalize. This phenomenon is called \textit{overfitting}. Pruning strategies play an essential role in avoiding it.

If pruning is enabled, then at each split BigML tries to determine whether that split increases the confidence (for classification models) or decreases the expected error (for regression models). If it does not, then it is pruned away. The pruning uses pessimistic error estimates as given by either the Wilson score interval or a standard confidence interval for regression trees.

\textbf{Note:} a node needs at least two instances to have a split, so models built on datasets with less than 200 instances will not be pruned.

### 1.2.5 Field Importance

The field importance is the relative contribution of each field to the objective field predictions. So, a field with higher importance will have a greater impact on predictions. You can find a visual histogram containing the importance for all fields in the Model Summary Report. (See Subsection 1.6.1.)

BigML calculates the field importances by estimating the prediction error each field helps to reduce for each split in a tree (these same estimates are used when pruning). Then, BigML sums the error reduction for every split grouped by field. Then, those sums are normalized, so that they total to 1. Conceptually, BigML measure of importance is based on the popular Random Forest measure of Breiman’s Gini importance \cite{1}, except that BigML does not use Gini as the error metric.

Fields with an importance of zero may still be correlated with the objective field. It just means the model preferred other fields when choosing splits.

It is worth noting that the field importance values from Random Decision Forests will often be more meaningful than they are from a single tree.

\textbf{Note:} The concept of field importance is also used in prediction explanation for single predictions (See Subsection 1.7.4.1.1). But they are calculated differently. A field can be very important for the model but insignificant for a given prediction.

### 1.2.6 Confidence and Probability

\textbf{Classification models} in BigML include two different measures of the model’s certainty when predicting a class at a node: \textit{confidences} and \textit{probabilities}. Both metrics can take values between 0\%, which means the prediction is no better than randomly selecting a class, and 100\% which indicates absolute certainty. The main difference between both metrics is that the confidence penalizes more heavily a low number of instances of the predicted class at the node. The confidence is considered a pessimistic measure of the model certainty because it is usually lower than the probability unless the number of instances at the node is very high in which case the confidence and probability take similar values. A detailed explanation of each one can be found in the following subsections.

#### 1.2.6.1 Confidence

The confidence is calculated in BigML using the Wilson score formula. You can find a detailed explanation of the \textit{confidence formulation} and its interpretation below.

#### 1.2.6.1.1 Formulation

BigML computes confidence based on the lower bound of the Wilson score interval\textsuperscript{13}, which means it provides a pessimistic confidence:

\[
\hat{p} + \frac{1}{2n} z^2 - z \sqrt{\hat{p}(1-\hat{p}) + \frac{z^2}{4n^2}}
\]

\textsuperscript{13}\url{https://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval#Wilson_score_interval}
Where \( \hat{p} \) is the proportion of instances for the predicted class, \( n \) is the number of instances at that node and \( z = 1.96 \), calculated as the \( \frac{1-\alpha}{2} \) quantile of the standard normal distribution for a level of error of \( \alpha = 5\% \).

### 1.2.6.1.2 Interpretation

The main goal is to balance the proportion of the predicted class with the uncertainty of a small number of instances. This formula takes into account two factors:

- The **class distribution** of the node: assuming equal number of instances, the more homogeneous the node, the higher the confidence. In other words, the node containing more instances for the predicted class will have higher confidence.

- The **number of instances** in the node: a node containing 1,000 instances with the same class distribution vs. another one containing five instances deserves higher confidence as uncertainty about the predicted class is higher whenever fewer instances are involved.

For example, if we are trying to predict if a person will have diabetes, **Figure 1.4** shows both example nodes have the same predicted class distribution (93.75%), but the node with 64 instances has more confidence than the node with 16 instances because of the adjustment the Wilson interval formula makes to penalize the lower number of instances.

The idea is to provide a confidence measure underestimating the performance of your model, so you can be sure that predictions will not do it worse than the confidence.

BigML shows the confidence in several places: in a model’s top panel, at each node, in the prediction form (see **Subsection 1.7.2.2**), and in the BigML API.

![Figure 1.4: Example of confidences in two different nodes of the same tree](image)

### 1.2.6.2 Probability

The **probability** of a class at a certain node is the **percentage of instances for that class at the node**. BigML also adds Laplace smoothing also known as **additive smoothing**\(^1\) to this probability. This means the class probabilities will tend to be pulled towards their overall proportion in the dataset. If there are only a few instances in the leaf leading to the prediction, this smoothing can have a significant impact. If there are a high number of instances, the smoothing will have little effect.

For the probability calculation BigML takes into account the **total number of instances for the class** of interest at a given node ("node_class_count"), the **total number of instances in the node** ("node_total_count"), and the **percentage of instances for that class** over the total instances in the dataset ("class_prior"):

\[
\text{Probability}_{\text{class}} = \frac{\text{node_class_count} + \text{class_prior}}{\text{node_total_count} + 1}
\]

To **illustrate** the probability calculation with an example, imagine a dataset with a total of 100 instances and three classes: class_a (30 instances), class_b (50 instances), and class_c (20 instances). The "class_prior" for each of them is:

\(^1\)See [Additive Smoothing](https://en.wikipedia.org/wiki/Additive_smoothing)
If we were calculating the probabilities at a given node containing a total of 20 instances where 10 instances belong to class_a and other 10 instances to class_c, the probabilities for each class will be calculated as:

\[
\text{probability\_class\_a} = \frac{10 + 0.3}{20 + 1} = 0.49
\]

\[
\text{probability\_class\_b} = \frac{0 + 0.5}{20 + 1} = 0.024
\]

\[
\text{probability\_class\_c} = \frac{10 + 0.2}{20 + 1} = 0.486
\]

### 1.2.7 Expected Error

The measure of the predictions certainty at a node for regression trees is the **expected error**.

#### 1.2.7.1 Formulation

The formula below is the standard formula to estimate the average squared error for the prediction at a given node:

\[
\hat{e} = \frac{\sum_i (v_i - x)^2}{n}
\]

Here, \(x\) is the predicted output at the node, \(n\) is the total number of instances, and \(v_1...v_n\) are the rest of the objective values at that node. Note that if we use \(n - 1\) instead of \(n\), this formula becomes the sample variance (\(S^2\)) of the node. To calculate the expected error for the predictions, we need to include the error that could arise from our sample variance not representing the true population variance. We solve this problem by constructing a confidence derived from the true variance, which we will call \(\sigma^2\).

\[
\left[ S^2(n-1), S^2(n-1) \right] \chi^2_{\alpha/2, n-1}, \chi^2_{1-\alpha/2, n-1}
\]

The notation \(\chi^2_{p,f}\) indicates the critical value from the \(\chi^2\), distribution at probability \(p\) with \(f\) degrees of freedom. Using the upper bound, we can calculate the standard error of our estimate of the mean. The estimated error using the bounds is then our upper bound on the standard deviation \(\hat{\sigma}\), plus the possible error in the mean above, squared:

\[
\hat{e} = (z \hat{\sigma} \sqrt{\frac{n}{n}} + \hat{\sigma})^2
\]

\[
= \frac{z^2 \hat{\sigma}^2}{n} + \frac{2z\hat{\sigma}^2 \sqrt{\frac{n}{n}}}{n} + \frac{\hat{\sigma}^2 n}{n}
\]

\[
= \hat{\sigma}^2 (\frac{z^2 + 2z\sqrt{n} + n}{n})
\]

\[
= \hat{\sigma}^2 (\frac{z + \sqrt{n}}{n})^2
\]

If the variance (and thus error \(\hat{e}_c\)) is zero, we use the parent error \(\hat{e}_p\) to construct the following estimate of the child error given the parent prior \(\hat{e}_{c|p}\):

\[
\hat{e}_{c|p} = \frac{\hat{e}_p + n\hat{e}_c}{n + 1} = \frac{\hat{e}_p}{n + 1}
\]
1.2.7.2 Interpretation

The expected error is an average of the instance errors at a given node. In Figure 1.5, the node predicts a value of 70.60 with an error of 28.59, meaning the prediction is, on average, likely to be within 28.59 of the target (between 42.01 and 99.19 in this example). This is an average, so while a single prediction may be more than 28.59 away from the true target, on average it is expected to do better than that. As in the confidence calculations for classification models, it is a pessimistic measure of your model performance, so that you can be sure your predictions will not do worse than the predicted value +/- the expected error.

![Figure 1.5: Example of the expected error at a node](image)

Along with the prediction and the expected error at a node, BigML also provides the histogram with the instances and their values. This allows you to find out the predicted value distributions and gives you information such as the complete possible range for the objective variable or whether the error is more likely to be skewed to one side of the prediction. If you need to further explore the data for a specific branch, you can also create a dataset from that branch (see Figure 1.26).

1.2.8 Models with Images

BigML models do not take images as input directly, however, they can use image features as those fields are numeric.

BigML extracts image features at the source level. Image features are sets of numeric fields for each image. They can capture parts or patterns of an image, such as edges, colors and textures. For information about the image features, please refer to section Image Analysis of the Sources with the BigML Dashboard\(^\text{15}\) [11].

\(^{15}\text{https://static.bigml.com/pdf/BigML_Sources.pdf}\)
As shown in Figure 1.6, the example dataset has an image field `image_id`. It also has image features extracted from the images referenced by `image_id`. Image feature fields are hidden by default to reduce clutter. To show them, click on the icon “Click to show image features”, which is next to the “Search by name” box. In Figure 1.7, the example dataset has 234 image feature fields, called Histogram of gradients.

From image datasets like this, models can be created and configured using the steps described in the following sections. All other operations including prediction, evaluation applies too.
1.3 Creating Models with 1-Click

To create a model in BigML you have two options: either the 1-click option which uses the default values for all available configuration options, or you can tune the parameters in advanced by using the configuration options explained in Section 1.4. You can also create a BigML model either from a dataset or from a cluster. Creating BigML models from clusters will be explained with more detail in the Cluster Analysis with the BigML Dashboard document [8]. This section will guide you through the process of creating BigML models from a dataset with just 1-click.

The BigML unique 1-click feature provides a convenient way to quickly train a model from a dataset. The 1-CLICK MODEL menu option is available in the 1-click action menu. (See Figure 1.8.)

Alternatively, you can select the 1-CLICK MODEL option in the 1-click action menu from the dataset list view. (See Figure 1.9.)

Either option builds a new model using default values for all available configuration options (see Section 1.4). This can give you a quick starting point to understand how your model behaves and how it can be improved.

Creating a model may take a variable time, depending on how big your dataset is, your subscription plan, etc. Once your BigML model is ready, it will be automatically displayed on your BigML Dashboard and you will be able to explore it through BigML sophisticated visualizations (see Section 1.5), and using it for evaluations (see Chapter 7) or predictions (see Section 1.7.)

1.4 Model Configuration Options

While the 1-click creation menu option (see Section 1.3) provides a convenient and easy way to create a BigML model from a dataset, there are cases when you want more control. This section will focus on
the options that BigML offers to configure its internal algorithms for BigML models.

You can set a number of parameters that affect the way BigML creates models from a dataset. Such parameters can be grouped in two categories:

- Parameters that are permanently associated to the dataset, such as its objective field and preferred fields. Once you provide a value for a dataset's permanent parameters, they will be used as a default value for the creation of models from that dataset.

- Parameters that only affect the model that is currently being created and that you are expected to set each time. Those include the objective field, included/excluded fields, and a number of configuration options that are described below.

Set a dataset's permanent parameters by clicking on the **edit** button that is displayed when you hover on the dataset's fields. This opens a modal dialog where you can set some of the field properties (See Figure 1.10).

![Figure 1.10: Configure permanent parameter modal](image)

Click on the **preferred field** button to make that field **non-preferred**.

Click on the **objective field** button to make that field the new objective field.

To access the configuration panel, select the **CONFIGURE MODEL** menu option located in the **configuration menu** of your dataset's detail view. (See Figure 1.11.)
When the configuration panel is displayed, you can:

- Select or deselect individual fields for them to be included in or excluded from the model computation.
- Change the objective field used for the model to be created.
- Manually configure a number of configuration options or automatically optimize these options.

Note: when the configuration panel is displayed, the edit is not visible, so you cannot set the dataset’s permanent properties.

See below for a detailed explanation of the configuration options that are available as well as the corresponding default values.

1.4.1 Objective Field

Also known as “target field”, the objective field is the output variable you want to predict.

Select your objective field in BigML in either of two ways. Specify the objective field each time you create a model from the configuration panel or set a field as the default objective for all models by clicking the edit button and then the objective field button.

By default, BigML will use the last valid field in your dataset as objective, with the exemption fields of type text and items that cannot be used as objective.

1.4.2 Automatic Optimization

You can turn on the Automatic optimization option so BigML will automatically tune the parameters of your model (see Figure 1.12).
Figure 1.12: Automatic optimization

The high number of possible combinations for parameter values makes it difficult to find the optimum configuration since the combinations that lead to a poor result outnumber the ones that result in a satisfying performance. Hand-tuning different configurations is a time-consuming process that requires a high level of expertise and intuition. To combat this problem, BigML offers first-class support for automatic model parameter optimization.

Behind the scenes, BigML uses the same technology for model parameter optimization as the one used for OptiML. If you want to know more about the technical details, please read the Chapter 2 of the document OptiML with the BigML Dashboard [10].

When you turn on the Automatic optimization option, all the model parameters will be disabled (because they will be automatically optimized), except the Missing splits and the Weights parameters which you can manually configure (see Subsection 1.4.4 and Subsection 1.4.6).

Figure 1.13: Configure the missing splits and the weights for your model
Since the optimization process can take some time, BigML offers two configurable parameters to limit the time to create the optimized model: a training duration (see Subsection 1.4.2.1) and the model candidates (see Subsection 1.4.2.2).

### 1.4.2.1 Training duration

The scale parameter to regulate the model runtime. It's set as an integer from 1 to 10. It indicates the user preference for the amount of time they wish the optimization to take. The higher the number, the more time that users are willing to wait for possibly better model performance. The lower the number, the faster that users wish the model training to finish. The default value is set to 5.

The training duration is set in a scale. The actual training time depends on the dataset size, among other factors.

### 1.4.2.2 Model candidates

The maximum number of different models (i.e., models using a unique configuration) to be trained and evaluated during the optimization process. The default is 128 models which is usually enough to find the best model, but you can set it from 4 up to 200. The top-performing model model will be returned. If the training duration is very low (see Subsection 1.4.2.1) given the dataset size, it is possible that not all the model candidates will be tried out.

### 1.4.3 Pruning

As explained in Subsection 1.2.4, pruning strategies are essential to avoid overfitting, a phenomenon that reduces a model's ability to generalize.

In BigML you can choose three different strategies for pruning (Figure 1.16):
• **Smart Pruning**: considers pruning the nodes with less than 1% of the instances.
• **Statistical Pruning**: considers every node for pruning.
• **No Statistical Pruning**: deactivates pruning altogether.

By default, BigML uses Smart Pruning to create your models.

### 1.4.4 Missing Splits

When training a model, BigML may encounter missing values, which can be either considered or ignored for the definition of splitting rules.

To include missing splits in your model, enable the **missing splits** option. (See Figure 1.16.) If missing values are included in your model, you may find rules with predicates of the following kind: `field x = "is missing" or field x = "y or is missing"`. BigML includes missing values following the MIA approach\(^\text{16}\) [14].

By default, BigML does not include missing splits.

### 1.4.5 Node Threshold

Set the **node threshold** to set a limit to a BigML model's growth. (See Figure 1.16.) A lower threshold simplifies the model while helping to avoid overfitting. However, it may also have reduce the model's predictive power compared to deeper models. The ideal number of nodes may depend on the dataset size and the number of features. Larger datasets with many important features may require more complex models. Reducing the number of nodes can also be useful to get an initial understanding of the basic data patterns. Then you can start growing the model from there.

By default, BigML sets a **512 node threshold**. Since nodes are computed in batches, on occasion the final number of nodes can be greater than the node threshold. (See Subsection 1.2.2.)

![Figure 1.16: Model configuration options](image)

### 1.4.6 Weight Options

It is not unusual for a dataset to have an unbalanced **objective field**, where some categories are common and others very rare. For example, in datasets used to predict fraud, usually fraudulent transactions are very scarce compared to regular ones. When this happens, models tend to predict the most frequent values simply because the overall model's performance metric improves with that approach. However,

\(^{16}\) [http://oro.open.ac.uk/22531/1/decision_trees.pdf](http://oro.open.ac.uk/22531/1/decision_trees.pdf)
in cases such as fraud prediction, you may be more interested in predicting rare values rather than successfully predicting frequent ones. In that case, you may want to assign more weight to the scarce instances so they are equivalent to the abundant ones.

BigML provides three different options to assign specific weight to your instances.

### 1.4.6.1 Balance Objective

When you set the balance objective weight, BigML automatically balances the classes of the objective field by assigning a higher weight to the less frequent classes, with the most frequent class always having a weight of 1. This option is only available for classification models. For example, take the following frequencies for each class:

[False, 2000; True, 50]

By enabling the balance objective option, BigML will automatically apply the following weights:

[False, 1; True, 40]

In this example, the class “True” is getting forty times more weight as it is forty times less frequent than the most abundant class.

### 1.4.6.2 Objective Weights

The objective weights option allows you to manually set a specific weight for each class of the objective field. BigML oversamples your weighted instances replicating them as many times as the weight establishes. If you do not list a class, it is assumed to have a weight of 1. Weights of 0 are also valid. This option is only available for classification models.

This option can be combined with the Weight field (see Subsection 1.4.6.3). When combining it with the Weight field, both weights are multiplied. For example if you assign a weight of 3 for the “True” class and the weight field assigns a weight of 2 for a given instance labeled as “True”, that instance will have a total weight of 6.

### 1.4.6.3 Weight Field

The Weight Field option allows you to assign individual weights to each instance by choosing a special weight field. It can be used for both regression and classification models. The selected field must be numeric and it must not contain any missing values. The weight field will be excluded from the input fields when building the model. You can select an existing field in your dataset or you may create a new one in order to assign customized weights.

For example, below is a dataset for which we included a field called “Weight” that assign a ten time higher weight to fraudulent transactions in comparison to non-fraudulent ones. BigML provides a powerful tool, the BigML Flatline editor, to add new fields to your dataset, such as a weight field. As an additional example, we could also take into account the transaction “Amount” to calculate the weights, so transactions with higher amounts will have higher weights.
### 1.4.7 Sampling Options

Sometimes you do not need all the instances contained in your testing dataset to build your model. If you have a very large dataset, sampling may be a good way of getting faster results. (See Figure 1.18.) The same sampling options described in the Datasets with the BigML Dashboard document to sample datasets, are also available when building BigML models. They are divided in two groups: sampling and advanced sampling options.

#### 1.4.7.1 Rate

The sampling rate is the frequency of instances being extracted from the dataset and included in your sample. A sampling rate of 100% means that all instances are included; a rate of 10% means 10% of the instances are included. This option may take any value between 0% and 100%. You can easily configure the rate by moving the slider in the configuration panel for sampling, or by typing the percentage in the tiny input box, both highlighted in Figure 1.18.

---

Table 1.1: Weight Field example for transactional dataset

<table>
<thead>
<tr>
<th>Trans. ID</th>
<th>Products</th>
<th>Online</th>
<th>Amount $</th>
<th>Fraud</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxxxxxx098</td>
<td>XYZGH</td>
<td>yes</td>
<td>3,218</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxxx345</td>
<td>VBHGF</td>
<td>no</td>
<td>1,200</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxxx123</td>
<td>UYFHJ</td>
<td>yes</td>
<td>5,000</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxxx567</td>
<td>HSNKI</td>
<td>no</td>
<td>390</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxxx789</td>
<td>SHSYA</td>
<td>yes</td>
<td>500</td>
<td>TRUE</td>
<td>10</td>
</tr>
<tr>
<td>xxxxxxx093</td>
<td>DFSTU</td>
<td>yes</td>
<td>423</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxxx012</td>
<td>TYISJ</td>
<td>yes</td>
<td>60,000</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxxx342</td>
<td>SJSOP</td>
<td>no</td>
<td>789</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxxx908</td>
<td>IOPKJ</td>
<td>no</td>
<td>9,450</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxxx334</td>
<td>HIOPN</td>
<td>yes</td>
<td>50,678</td>
<td>TRUE</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 1.17: Weighting arguments for models
By default, BigML uses a 100% rate.

1.4.7.2 Range
The sampling range is the subset of the dataset instances from which to sample, e.g., from instance 5 to instance 1,000. The rate will be applied over the range configured.

By default, all instances are included, i.e., the range is \((1, \text{num. rows in dataset})\).

1.4.7.3 Sampling
The sampling option represents the type of the sampling process, which can be either random or deterministic.

When using deterministic sampling the random-number generator will always use the same seed, producing repeatable results.

By default, BigML uses random sampling.

1.4.7.4 Replacement
The replacement option controls whether a single instance can be selected multiple times or not. Sampling without replacement ensures that each instance cannot be selected more than once.

By default, BigML generates samples without replacement.

1.4.7.5 Out of Bag
The out of bag option allows you to include in your sample only those instances that were not selected in the first place, thus effectively inverting the sampling outcome. It is only selectable when a sample is deterministic and the sample rate is less than 100%. The total percentage of instances included in your sample will be one minus the rate (when replacement is not allowed). This can be useful for splitting a dataset into training and testing subsets.

By default, BigML will not use out of bag instances.
### 1.4.8 Advanced Ordering

Ordering options are relevant to ensure that BigML can correctly determine whether it can take an early split of your dataset to accelerate the training process. In particular, early splitting can only be safely used if the training instances have been previously shuffled. (See Subsection 1.2.3.1.)

If your instances are already shuffled, BigML allows you to choose the linear option. This will make the process of building the model much faster, since it will not required to reshuffle the dataset. If you need to shuffle your instances, BigML provides two options to that aim, deterministic shuffling and random shuffling, which are described below.

Ordering options have no influence on datasets of less than 34GB, since the whole dataset is used to build the model.

By default, BigML uses deterministic shuffling to ensure the same (deterministic) sample of the instances is used and the built model is thus repeatable.

#### 1.4.8.1 Deterministic Shuffling

The deterministic shuffling option ensures that the row shuffling of a dataset is always the same, so that retraining a BigML model from the same dataset yields the same results.

By default, this option is true.

#### 1.4.8.2 Linear Shuffling

The linear shuffling option is useful when you know that your instances are already in random order. Using linear shuffling, the BigML model will be constructed faster.

By default, this option is false.

#### 1.4.8.3 Random Shuffling

The random shuffling option will ensure that a different shuffling will be tried each time you train your model.

By default, this option is false.

![Figure 1.19: Ordering argument for models](image)

### 1.4.9 Creating Models with Configured Options

After finishing the configuration of your options, you can change the default model name in the editable text box. Then you can click on the Create model button to create the new model, or reset the
configuration by clicking on the Reset button.

Figure 1.20: Create model after configuration

1.4.10 API Request Preview

The API Request Preview button is in the middle on the bottom of the configuration panel, next to the Reset button (See Figure 1.20). This is to show how to create the model programmatically: the endpoint of the REST API call and the JSON that specifies the arguments configured in the panel. Please see (Figure 1.21) below:

Figure 1.21: Model API request preview

There are options on the upper right to either export the JSON or copy it to clipboard. On the bottom there is a link to the API documentation for models, in case you need to check any of the possible values or want to extend your knowledge in the use of the API to automate your workflows.
Please note: when a default value for an argument is used in the chosen configuration, the argument won’t appear in the generated JSON. Because during API calls, default values are used when arguments are missing, there is no need to send them in the creation request.

1.5 Visualizing BigML Models

Being able to effectively visualize a model is paramount to exploring it, interpreting it, and explaining why it produces certain outcomes. To this aim, BigML provides two powerful interactive visualizations for BigML models, the compact tree and the sunburst views, which will be introduced below.

1.5.1 BigML Compact Tree Visualization

BigML compact tree is a powerful, patented, visual tree representation of your BigML model. Thanks to their being interactive, BigML compact trees bring to a new level your ability to explore your BigML models, by allowing you to see at a glance the most relevant information encoded in the model, by allowing you to collapse less important parts, and by enabling advanced filtering options.

1.5.1.1 BigML Compact Tree Visual Representation

BigML compact trees rely on the almost natural representation of BigML models as trees to convey a wealth of information in a compact format:

- The color of each node in the compact tree identifies the input field associated to it, i.e. the field that was identified as providing the best split. See Subsection 1.2.3 for more information.

- The first node at the top, called the tree root, corresponds to the split that best divides your data into two segments according to your objective variable. However, one must not confuse the best split with the most significant field since a root field may not always be as important as other predictors after growing the full tree. See Subsection 1.2.5 for more information.

- The branches connect the root to its child nodes at the next level down. Each child node may recursively have a number of children, to which it is connected through the afferent branches. This goes forth until we get to the terminal nodes, called leaves, that have no further children and are used to generate predictions.

- The width of a node represents the number of instances that the branch rooting at that node includes.

![Figure 1.22: Tree visualization](image-url)
1.5.1.2 Interacting with BigML Compact Trees

You can explore even more information that is encoded in your BigML model interacting with it. If you hover over the tree nodes, additional information about those nodes will be displayed. Additionally, you can set interactive filters to display only a subset of nodes based on a given criteria. Finally, you can create a dataset from a node to further investigate the data instances that support it. See below a description for all these interaction modes.

If you mouse over a node in the tree, BigML compact tree will display:

- The prediction at that node along with a measure of the prediction’s certainty on the top navigation bar above the tree visual:
  - For classification models you can display the confidence or the probability by clicking in the switcher shown in Figure 1.23. See a detailed explanation of both measures in Subsection 1.2.6
  - For regression models you will get the expected error. See a detailed explanation in Subsection 1.2.7).

- The decision prediction path for the node, i.e. the series of rules that lead to it, on the right side of the tree.
- The number of instances that follow this decision path as well as their distribution in terms of the objective field values in the tooltip appearing on top of the hovered node. In the same tooltip you will see again the prediction at that node along with the confidence or probability (for classification models) or the expected error (for regression models).

Figure 1.23: Node prediction
Because models often grow too wide to comfortably fit the display area, BigML makes it possible to collapse some of the less important parts:

- To show all the nodes for a particular path, just click on that node and the whole branch will be displayed.
- To return to the default view, click on the root of the tree.

### 1.5.1.2.1 Interactive Filters

BigML compact trees provide a set of filters that let you interact with your model and control what part of it is displayed or hidden. Filters are interactive, meaning that you will see in real-time how applying a given filter changes the visualization. Filters aim to make it easier for you to analyze the information the model conveys in a number of ways, as explained below.

All available filters are shown in the top bar just above the tree, as depicted in Figure 1.25.

- **Filter by support**: this filter displays or hides branches based on how well they are supported by all data instances. To apply it, you specify the minimum and maximum percentage of instances that you require to provide support to visible branches. This is useful to identify branches with a minimum threshold of supporting instances.
- **Filter by prediction**: this filter displays or hides branches based on the predictions they provide. To apply it, you choose what predictions you are interesting in: for regression models, a slider allows you to set a range of values for your predictions; for classification models, select the predicted class.
- **Filter by confidence, probability or expected error**: this filter displays or hides nodes based on the certainty of the predictions they provide. For classification models you will be able to filter the model by the confidence or the probability depending on the option you select (see Figure 1.23). For regression models you will be able to filter your model by the expected error. This is useful to identify the most likely predictions your model provides.
- **Rare interesting patterns**: this filter displays only those branches that are supported by few instances, i.e. represent not so common cases, while still providing high confidence. Low support is defined as being supported by less than 1% of instances. High confidence is defined at least 50% confidence for classification models, or being in the lowest expected error decile for regression models.
- **Frequent interesting patterns**: this filter displays only those branches that have high support, i.e. represent frequent cases, and also provide high confidence. High support is defined as being
supported by more than 1% of instances. High confidence is defined at least 50% confidence for classification models, or being in the lowest expected error decile for regression models.

1.5.1.2.2 Creating a Dataset from a Node

As mentioned, you can create a dataset collecting all the instances that support a given node. This can be useful if you want to apply other Machine Learning algorithms to it to gain further insight into the set of rules that belong to that branch.

Create a dataset from any node of your model by following these steps:

- Click on a node to select a specific view of the model.
- Press the `SHIFT` key on your keyboard to freeze the current view.
- In the frozen view, click the `Create dataset` button. (See Figure 1.26.)
- Release the frozen view by pressing the `ESC` key on your keyboard.

By executing the above listed steps, a new dataset will be created containing only the instances that support the selected node.
1.5.2 Partial Dependence Plot

Apart from the main compact tree view, BigML also offers an alternative view for models: the Partial Dependence Plot (PDP). The main goal is to represent the marginal effect of a set of variables (input fields) on the model predictions disregarding the rest of the variables. It is a common method for visualizing and interpreting the impact of the variables on the predictions, and it can be used for classification and regression models.

Figure 1.26: Create dataset from a branch
Figure 1.27: Model PDP view

Note: the model PDP is not a representation of the dataset values; it is a representation of the model results and their dependence from a set of variables used as inputs.

The PDP allows you to visualize the model predictions in the heatmap chart. In the case of classification models, the different classes of the objective field are represented by different colors. The different color shadings for each class represent the different confidences or probabilities depending on the option you select from the switcher highlighted in Figure 1.28.
Figure 1.28: Classification model PDP

For *regression* models, the different prediction values for the objective field are represented by differences in the *color scale*.
You can select any categorical or numeric field for each axis. You can also switch the axis by clicking on the option on top of the chart area. In the grey area next to the axis selectors you can see the axis values. You can freeze the view by pressing \texttt{Shift} and release it again by pressing \texttt{Escape} from your keyboard. When the view is frozen, an edition icon will appear and you can edit the axis values to obtain a prediction for that value. (See Figure 1.30.)
The prediction legend next to the chart allows you to visualize the **objective field classes** (classification models) or the predicted **value** (regression models). In the case of classification models you will also obtain the **confidence** or the **probability** depending on your selected option (see Subsection 1.2.6). By default, the color tones and shadings are set according to the range of values shown in the chart area. This is the default because for some configurations of the chart the predictions may vary a small amount relative to the global range. For example, imagine the chart is showing temperature predictions based on location, time-of-year, and time-of-day. San Diego’s daily range (13°C to 18°C) could be tiny compared to the Earth’s global range (-62°C to 48°C). You can change this behavior and see the color scales and shading according to the total range of possible predicted values by clicking on the icon next to the probability bar |Total|. For classification models, this option allows you to see the **color shading** for the total range of potential confidence or probability values (from 0% to 100%). For regression models, the |Total| colors option allows you to see the **color scale** for the total range of predictions. For classification models you can also select to see only one of the classes using the class selector at the bottom of the legend. (See Figure 1.31.)
Figure 1.31: Prediction legend options

Below the chart legend, you can find the input fields form. (See Figure 1.32.) You can configure the values for any numeric, categorical, text or items field. By changing their values, you can see the predictions changing in real-time. You can sort the fields by their importance, select or disable them. If you disable an input field, it will be ignored to calculate the final prediction. The strategy used to calculate predictions when some fields are disabled is the proportional missing strategy (see Subsection 1.7.3.1).
If you want to know more about how to interpret the PDP view, please refer to Subsection 2.5.1.2.

1.5.3 Sunburst Visualization

In addition to compact trees and the PDP, BigML provides an additional, BigML-unique visualization for BigML models: the Sunburst, which allows you to visualize the entire model in a single view, without collapsing or expanding branches.

The Sunburst diagram can be seen as a representation of a BigML tree as if you were looking at a real pine tree directly down from its tip. So, the center circle in Figure 1.33 represents the root of the tree and its children wrap around it. Similarly, moving from the center further out, you will see the first split of the tree followed by the others.

For the uninitiated, BigML Sunburst diagram may not appear as immediately intuitive as our regular tree view. Nonetheless, it can help advanced users gain more insight into their models. In particular:

• More layers indicate deeper (i.e., more complex) prediction paths.
• The arc length of each child ring corresponds to the percentage of the training set covered by that child.
• Smaller arcs indicate lower support for that child.
• When you mouse over the chart, the prediction rules will be shown to the right of the diagram as in the tree visualization view. However, the Sunburst diagram summarizes the prediction rules whenever possible. For example, two rules applicable to a leaf node may read: \( x \leq 2 \) and \( 2 < x < 7 \). The sunburst view would combine these conditions and restate it as \( x < 7 \), which makes it easier to read and understand.
• When you click on a child ring, you zoom in the diagram, and all layers above the clicked one are collapsed into the diagram center. In the pine tree analogy, clicking on a ring would be equivalent to moving the viewpoint down along the tree, so you can see more detail from the bottom layers. Click the center of the Sunburst to move up one level up to the tip.

As a final note, you can color the chart by field, by prediction, or by confidence (or expected error in case of regression trees).
1.6 Model Summary Report

From a model's detail view, you can access the model summary report by clicking on the icon highlighted in Figure 1.36.

The model summary report has two tabs: field importance and summary, explained in the following subsections.

1.6.1 Field Importance

The field importance provides you a measure of how important a field is relative to the other fields. See Subsection 1.2.5 for more details.
1.6.2 Summary

This summarized view of your model, includes:

- **Data distribution**: for classification problems, percentage of the dataset instances that belong to each of the objective field classes; for regressions, this is the objective field distribution.

- **Predicted distribution**: for classification problems, percentage of predicted instances for each of the objective field classes; for regressions.

- **Field importance**: a measure of how important a field is relative to the other fields (see also Subsection 1.2.5).

- **Rules summary**: a summary of the rules that the model learned from the dataset. This means that for each possible prediction outcome, you can find here a summary of the paths that would produce that prediction.
1.7 BigML Model Predictions

1.7.1 Introduction

The ultimate goal in building a BigML model is being able to make predictions for previously unseen instances with an unknown label. In BigML, you can make predictions for single instances or for many instances in a batch. Each prediction comes with a measure indicating its reliability, expressed either as a confidence, probability or as an expected error.

The predictions tab in the main menu of your BigML Dashboard is where all your saved predictions are listed. (See Figure 1.39.)

Model predictions are saved under the CLASSIFICATION & REGRESSION option in the menu. (See Figure 1.40.)
From this view you can select to view the list for your single instances predictions or your batch predictions by clicking in the corresponding icons. (See Figure 1.41 and Figure 1.42.)

In the predictions list view, you can see, for each prediction, the Model or Ensemble icon used for the prediction, the Name of the prediction, the Objective (objective field name), the Prediction (the prediction result), and the Age (time since the prediction was created). (See Figure 1.43.)

You can also search your predictions by name by clicking the search button on the top right menu.

1.7.2 Creating Model Predictions

As shown in Figure 1.44, BigML provides three options to make predictions from your models:

- **PREDICT BY QUESTION**: to predict a single instance answering just the relevant questions required by the model.
- **PREDICT**: to predict a single instance using the prediction form.
- **BATCH PREDICTION**: to predict multiple instances simultaneously.
1.7.2.1 Predict by Question

This option will ask for a series of input field values, one question at a time, required to make a single instance prediction. In the general case, only the values of a subset of all input fields are needed to reach a prediction. So although your model may have seven data fields as potential predictors, if it is able to make a prediction with only two answered questions, then you will not have to specify the remaining seven fields by way of answering more questions. Follow these steps:

1. Start the questions by clicking **Predict**.

2. Set the values for each question and click **Next question**.

3. After the model gives you a final answer, you will be able to optionally **Save** the prediction.
BigML allows you to quickly make predictions for single instances by providing a form containing the fields used by the model, so you can easily set the input data and get an immediate response. This option is only available from the BigML Dashboard for models with less than 100 fields. If you want to perform single instance predictions for models with a higher number of fields, you can use the BigML API.\(^{17}\)

Follow the steps detailed below to create a single prediction:

1. Choose the PREDICT option under the model 1-click menu. (See Figure 1.48.)

Alternatively, you can choose the PREDICT option in the pop up menu in the list view as shown in Figure 1.49.

\(^{17}\)https://bigml.com/api/predictions
2. You will be redirected to the **prediction form** where you will find all the fields used by the model as predictors ordered by **field importance**. The importance percentage is found next to the field name as shown in **Figure 1.50**. You may not find all the fields from your original dataset because the model may find them irrelevant or redundant in terms of their predictive impact.

3. **Select the fields** you want to be taken into account for your prediction as shown in **Figure 1.51**. Non-selected fields will be considered as missing values during the prediction. If your model was trained with Missing splits (see **Subsection 1.4.4**), then missing values are considered by the model as any other valid value. If your model was built without missing values then any of the Missing strategies may apply during your prediction (see **Subsection 1.7.3.1**.)
4. **Set input values** for your selected fields. Depending on the field type, you will need to input the values differently:

- Numeric fields: move the slider or input a specific value in the edition box.
- Categorical fields: select one class from the selector.
- Text fields: write one or several terms in the free text box.
- Date-time fields: select the appropriate values from the selector.
- Items fields: when you write the first three characters of an item name, several items matching those characters will appear, so you can select the right one. You can input more than one item for a field.

5. **Get the prediction** along with the **confidence**, the **probability** or **expected error** on the top of the form. BigML predictions are synchronous, i.e., when you send the input data you get an immediate response. Read more about local predictions in Subsection 1.7.2.2.1.
6. Optionally save the prediction so you can access them afterwards from the model predictions list view.

1.7.2.2.1 Local Predictions

BigML provides local predictions from the BigML Dashboard for single instance predictions. Local predictions allow you to get a real-time prediction without consuming any credits or requiring an internet connection. This is possible because your model is saved in the browser's memory so when the input values change, BigML immediately navigates your model to obtain their predictions in a matter of microseconds.
1.7.2.2 Predictions with Images

BigML models can be trained from images using extracted image features (Subsection 1.2.8). Because image features are automatically generated numeric fields, creating model predictions with images is the same as creating other models. The only thing different is input fields of images.

Note: When the input fields contain images, in order to create the single prediction, BigML will extract image features automatically to match what were used in the dataset to train the model.

![Figure 1.54: Select a single image source in the image input field](image)

The model in Figure 1.54, “grape-strawberry”, was created from a dataset containing image features Histogram of gradients. This set of image features are extracted by default for all image composite sources. Creating a prediction using the model will be directed to the prediction form which presents all input fields used by the model. One of them is the image field. Because this is a single prediction, an image is input by using a single image source. Clicking on the input field box, single image sources available will be in the dropdown list. There is also a search box which can be used to locate specific ones.
Oftentimes single image sources were used for creating a composite source, they become component sources of the composite source. Or an image was uploaded as a part of an archive file (zip/tar) which created a composite source. In those cases, the composite source will be shown in the dropdown list, along with an icon “List components”. In the example in Figure 1.55, predict-images.zip is a composite source, click on the icon to show its component sources.
After the component sources of the composite are listed, scroll the dropdown list to find the desired one, then click to select it, as shown in Figure 1.56. There is also a search box to locate specific component sources.

Figure 1.56: Select a component of a composite source

Figure 1.57: Model prediction form, image field and more
In addition to images, models may use other fields, which will be in the prediction form too. As shown in Figure 1.57, all the fields can be selected, and their input values be set by dragging the knobs on the sliders or by entering precise values in their input boxes.

Once all fields are selected, click on the green button Predict to create a prediction.

After a new prediction is created, as shown in Figure 1.58, the predicted class is at the top of the form along with its probability. The prediction interface is the same as ones created by non-image models. Everything described earlier in this section (Subsection 1.7.2.2) applies.

### 1.7.2.3 Batch Predictions

BigML batch predictions allow you to make simultaneous predictions for multiple instances. All you need is the model you want to use to make predictions and a dataset containing the instances for which you want to obtain prediction. BigML will create a prediction for each instance in the dataset. Follow the steps detailed below to create a batch prediction:

1. Select the BATCH PREDICTION option under the model 1-click menu (see Figure 1.59) or the CREATE BATCH PREDICTION option in the pop up menu of the list view (see Figure 1.60.)
2. **Select the dataset** containing all the instances you want to create a prediction for. The instances should contain the input values for the fields used by the model as predictors. You can also select a subset of the model fields to be taken into account by configuring your prediction (see Subsection 1.7.3.4.) BigML batch predictions can handle missing data in your prediction dataset (see Subsection 1.7.3.1.)

3. **Optionally, select the model** you want to use for the prediction. BigML pre-selects the model you created the batch prediction from at step 1, but you can change it at any time in the batch prediction view by selecting another model from the model selector displayed in the right pane. You can even switch to an ensemble or logistic regression by selecting the corresponding icon in the top left menu.
4. After you have selected the model and the dataset, the batch prediction configuration options (see Subsection 1.7.3) will appear along with a preview of the prediction output, which is formatted as a comma-separated list of values (CSV format). (See Figure 1.62.) The default output includes all the fields in your prediction’s dataset plus a last column containing the calculated predictions.

Note: BigML does not include the predictions’ confidence, probability or expected error by default so you will have to configure your output file to include that information as explained in Subsection 1.7.3.5.

Figure 1.62: Configuration options displayed and output preview
5. By default, BigML generates an output dataset containing the batch prediction results. You can find in BigML Dashboard’s dataset list view and can use it as any other dataset to analyze the batch prediction output afterwards. If you do not want a dataset with all the prediction results to be created, you can deselect the button highlighted in Figure 1.63.

![New Batch Prediction](image)

**Figure 1.63: Create dataset from batch predictions**

6. Once you are done configuring your batch prediction, click the **predict** green button to generate it. This process may take some time depending on the size of the input dataset. (See Figure 1.64.)
7. After the batch prediction has been created, you will be able to download a CSV file with all the instances found in your input dataset along with the prediction corresponding to each one of them. (See Figure 1.65.)

8. If you did not disable the option to create a dataset, as explained above (see step 4), an Output dataset button will also be available to allow you to directly jump to the output dataset. (See Figure 1.66.)
1.7.2.3.1 Batch Prediction with Images

BigML models can be trained from images using extracted image features (Subsection 1.2.8). The input of a batch prediction is a dataset. So when creating a batch prediction with images, the dataset has to have the same image features used to train the model. The image features are in the dataset used to create the model.

As shown in Figure 1.67, the input for the model batch prediction is selected as predict-images, which is a dataset consisting of six images and contains the default set of extracted image features, Histogram of gradients.

Image features are configured at the source level. For more information about the image features and how to configure them, please refer to section Image Analysis of the Sources with the BigML Dashboard\[18\].

\[18\]https://static.bigml.com/pdf/BigML_Sources.pdf
1.7.3 Configuring Model Predictions

BigML provides several options to change its default behavior when calculating predictions. For single predictions as well as for batch predictions you can configure the strategy used for handling missing values (see Subsection 1.7.3.1.) and set a probability or confidence threshold for a given class only for classification models (see Subsection 1.7.3.2.). For batch predictions, you can also set a default numeric value for missing values (Subsection 1.7.3.3), the automatic fields mapping performed by BigML (Subsection 1.7.3.4), and define the output file settings (Subsection 1.7.3.5.)

1.7.3.1 Missing Strategies

When you create a new prediction, BigML will automatically navigate through the corresponding model to find the leaf node that best classifies the new instance.

However, it may just so happen that your new data (the instances you want to predict) does not have populated values for all the fields used in building the original model. For example, imagine that you are trying to predict diabetes and you have the patient’s glucose level and BMI (Body Mass Index) but not his blood pressure. If the model arrives at a node where the blood pressure level is required, BigML can handle this missing value by using one of these two strategies:

- **Last prediction**: it returns the prediction value and confidence or probability of the parent node.
- **Proportional**: it combines all subtrees predictions beneath the current node based on the data distribution of their child nodes in order to compute the prediction value and confidence or probability.

For single predictions you can select any of both Missing strategies by clicking in the icons shown in Figure 1.68.

![Figure 1.68: Missing strategies for single predictions](image)

For batch predictions you can find both options under the configuration panel as shown in Figure 1.69.

For the rest of batch predictions with images, including batch prediction configuration options and output datasets, everything stated earlier in current section (Subsection 1.7.2.3) applies.
Figure 1.69: Missing strategies for batch predictions

1.7.3.2 Confidence and Probability Threshold

Confidence and probability thresholds are only available for classification models, and they usually make sense when you want to minimize false positives at the cost of false negatives. The positive class will be predicted if its confidence or probability is greater than the given threshold, otherwise the following class with greater confidence or probability will be predicted instead.

To configure a threshold for your single predictions follow these steps:

1. Select the probability or the confidence measure using the buttons shown in Figure 1.70. To learn more about the differences between model confidences and probabilities refer to Subsection 1.2.6.
Chapter 1. Models

2. Select the **positive class**, i.e. the class for which you want to apply the confidence or probability threshold:

3. Set a value for the **threshold** using the slider. The positive class will only be predicted when the confidence or probability of the prediction is above the established threshold, otherwise the
following class with higher probability or confidence will be predicted instead.

Figure 1.72: Set a threshold

For batch predictions, you will find the same options under the Configure panel. (See Figure 1.73.)

Figure 1.73: Configure a threshold for batch predictions
1.7.3.3 Default Numeric Value

If the dataset used to make the batch prediction contains instances with missing values for the numeric fields you can easily replace them by the field’s Mean, Median, Maximum, Minimum or by Zero using the Default numeric value before creating your batch prediction. (See Figure 1.74.)

![Figure 1.74: Default numeric value for batch predictions](image)

1.7.3.4 Field Mapping

By default, BigML maps fields based on their names. If there is a mismatch between the field names in your model and those in the input dataset you selected for the batch prediction, you can specify the right correspondence between the two sets of fields by explicitly assigning to each field appearing in the “Model fields” column its associated input field in the “Dataset fields” column. (See Figure 1.75.)

If the dataset’s and model’s field names do not match but their IDs do, which happens when corresponding fields appear in the same order, you can tell BigML to use the field ID instead of the field name to map the fields. To this aim, click the green switcher shown in Figure 1.75.

If you do not want some of the fields to be considered during the evaluation, you can also manually search for those fields and remove them from the “Dataset fields” column.
Chapter 1. Models

Figure 1.75: Fields Mapping for batch predictions

The fields mapping from the BigML Dashboard has a limit of 200 fields. For batch predictions with a higher number of fields, use the argument \texttt{field\_map} from BigML API\footnote{https://bigml.com/api/batchpredictions#bp_batch_prediction_arguments} if you need to map your fields.

1.7.3.5 Output Settings

As mentioned, batch predictions can create a CSV file containing all input instances along with the predictions BigML calculated for each of them. Define the following settings to customize your prediction file:

- **Separator**: this option allows you to choose a separator for your output file values. The default separator is the comma. You can also select the semicolon, the tab, or the space.

- **New line**: this option allows you to set the new line character to use as the line break in the generated csv file: “LF”, “CRLF”.

- **Output fields**: this option allows you to include or exclude any of your dataset fields from the output file from the preview shown in Figure 1.76.

Note: a maximum of 100 fields are displayed in the preview, but all your dataset fields are included in the output file by default unless you exclude them.

- **Headers**: this option includes or excludes a first row in the output file (and in the output dataset) with the names of each column (input field names, prediction column name, probability and/or confidence column name, field importances column names, etc.). By default, BigML activates the headers.

- **Prediction column name**: this option allows you to customize the name for your predictions column. By default BigML uses the name of the model’s objective field.
• **Confidence or expected error**: this option allows you to include an additional column in the output file with the confidence or expected error per instance. By default, neither the confidence nor the expected error are included.

• **Confidence column name**: this option allows you to customize the name for the confidence (or expected error) column in case you include it in the output file. By default BigML uses “confidence”.

• **Probability**: this option allows you to include an additional column in the output file with the predicted class probability for each instance. By default, it is not included.

• **Probability column name**: this option allows you to customize the name for the probabilities column in case you include it in the output file. By default BigML uses “probability”.

• **All class confidences**: this option allows you to include the confidences for each class in the objective field. There is a column per class, named “<class_name> confidence”.

• **All class probabilities**: this option allows you to include the probabilities for each class in the objective field. There is a column per class, named “<class_name> probability”.

• **Importances**: this option allows you to include a column for each of the field relative importances for the model predictions. There is a column per field, named “<field_name> importance”.

![Figure 1.76: Output settings for batch predictions](image)

1.7.4 Visualizing Model Predictions

Model predictions visualization changes depending on whether you are predicting one single instance or you are predicting multiple instances using the **batch predictions** option. (See Subsection 1.7.4.1.)

1.7.4.1 Single Predictions

For single predictions you can find the prediction for your objective field at the top of the form along with the performance measure.
• For classification models you will find the objective field class predicted along with the probability or the confidence depending on which measure you select. You will also get all the class distribution histogram according to the measure selected, i.e., all class probabilities or all class confidences. (See Figure 1.77.)

![Figure 1.77: Single predictions view for classification](image)

• For regression models you will get a numeric prediction and the expected error for that prediction as shown in Figure 1.78.

![Figure 1.78: Single predictions view for regression models](image)

In either case, you can change any time the value of the displayed input fields to have your prediction recalculated in real-time.

If you have saved your prediction, you can go back to it and visualize it.

Read a detailed explanation of confidence, the probability, and expected error calculations in Subsection 1.2.6 and Subsection 1.2.7 respectively.
1.7.4.1.1 Prediction explanation

Prediction explanation helps understand why a model makes a certain prediction. This is very useful in many applications, and the reasons behind a model’s prediction are often as important as the prediction itself.

BigML prediction explanation is based on Shapley values. For more information, please refer to this research paper: *A Unified Approach to Interpreting Model Predictions* [3].

For any classification or regression model, you can request the explanation for the prediction by clicking the prediction explanation icon and then click Save (see Figure 1.79).

The prediction explanation represents the most important factors considered by the model in a prediction given the input values. Each input value will yield an associated importance, as you can see Figure 1.80. The importances across all input fields should sum 100%.
For some input fields you will see a “+” icon next to the importance. This is because the importance may not be directly associated with the input value, i.e., it can be explained by other reasons. In the Figure 1.81 below, the importance of 6.12% for the field “Age” is not explained by this field being equal to 44.93. Rather, it is because this field value is higher than 30.5 and lower than 45.47.

The prediction explanation for models is calculated using the prediction path of the decision tree.

Note: the input field importances in the prediction explanation are different from the overall field importances of the model. A field can be very important for the model but insignificant for a given prediction.

1.7.4.2 Batch Predictions

For batch predictions, you always get a CSV file and an optional output dataset.

1.7.4.2.1 Output CSV File

From the batch prediction view, you can access the CSV file containing your predictions for each of your dataset instances in the last column. (See Figure 1.82.) You can configure several options to customize your CSV file including the separator for the columns, the name of your prediction column, the dataset
fields you want to include, whether you want to include a first row with the names of your columns. You can find a detailed explanation of those options in Subsection 1.7.3.5.

Note: by default BigML does not include the predictions confidence, probability or expected error in your output file. Again you will need to click that option from the output settings panel if you want to include it.

Figure 1.82: Download batch prediction output CSV file

See an output CSV file example in Figure 1.83 where the two last columns contain the prediction and the confidence for each instance.

| Pregnancies, Glucose, Blood pressure, Skinfold, Insulin, BMI, Diabetes, Confidence |
| 8, 183, 64, 0, 23, 3, True, 0.6574 |
| 5, 116, 74, 0, 25, 6, False, 0.845 |
| 10, 115, 0, 0, 35, 3, True, 0.6459 |
| 8, 125, 96, 0, 0, 0, False, 0.9366 |
| 1, 189, 60, 23, 846, 30, 1, True, 0.7574 |
| 1, 103, 30, 36, 83, 43, 3, False, 0.675 |
| 7, 103, 66, 32, 0, 39, 1, False, 0.7622 |
| 1, 101, 50, 15, 36, 24, 2, False, 0.948 |
| 0, 100, 88, 60, 110, 46, 8, False, 0.5413 |

Figure 1.83: An example of a batch prediction CSV file

1.7.4.2.2 Output Dataset

By default, BigML creates a dataset out of your batch prediction. (See Subsection 1.7.3.5.) You can access your output dataset from the batch prediction view by clicking the Output dataset button shown in Figure 1.85.
In the output dataset you can find an additional field (named by default as per your model's objective field) containing the predictions for each one of your instances. If you configured your batch prediction to include the confidence, the probability or expected error you will be able to find it in the last field of your output dataset as shown in Figure 1.85.

1.7.4.2.3 Batch Prediction 1-Click Actions

From the batch prediction view you can perform the following actions (see Figure 1.86):
• **BATCH PREDICTION AGAIN**: this option will redirect you to the batch prediction creation view, with the same model and prediction dataset already selected. This option allows you to rapidly recreate the batch prediction using a different configuration.

• **BATCH PREDICTION WITH ANOTHER DATASET**: this option allow you to easily create a batch prediction using the same model and a different dataset.

• **BATCH PREDICTION USING ANOTHER MODEL**: this option allows you easily create a batch prediction using the same dataset and a different model.

• **NEW BATCH PREDICTION**: this option redirects you to the batch prediction creation view where you can select a prediction dataset and a model to create your prediction.

![Batch prediction 1-click actions](image)

**Figure 1.86: Batch prediction 1-click actions**

### 1.7.5 Consuming Model Predictions

BigML provides plenty of means for developers to integrate BigML model predictions within their apps. In the following sections, we will describe how you can use the BigML REST API and the BigML Python bindings to work with model predictions.

#### 1.7.5.1 Using Model Predictions via the BigML API

Model predictions have full citizenship in the BigML API. This means you can programmatically create, update, list, delete, and use them for predictions. For example, this is how you can create a single prediction using the command line from a given model and defining the input data. This will require properly setting the `BIGML_AUTH` environment variable to contain your authentication credentials:

```bash
curl "https://bigml.io/prediction?$BIGML_AUTH" 
-X POST 
-H 'content-type: application/json' 
-d '{"model": "model/50650bdf3c19201b64000020", 
"input_data": {"000001": 3, "000002":4.5, "000003"}}'
```

For more information on using model predictions through the BigML API, please refer to [prediction REST API documentation](https://bigml.com/api/predictions).

#### 1.7.5.2 Using Model Predictions via the BigML Bindings

BigML bindings provide a convenient way to access BigML REST API from your language of choice. They offer a higher-level view of BigML Machine Learning resources and algorithms in a number of resources and algorithms in a number of
languages, including Python, Node.js, Java, Swift, and Objective-C. For example, this is how you can
create a model prediction in Python using BigML bindings:

```python
from bigml.api import BigML
api = BigML()
prediction = api.create_prediction("model/573d997058a27e0f620038df",
    {"sepal length": 5,
     "sepal width": 2.8},
    {"name": "my prediction"})
```

BigML bindings also provide the means to carry through predictions locally, without ever hitting the
network, which can greatly improve the latency of predicting from your apps. This is made possible
by BigML models being white-box, meaning you can download them and use them independently from
BigML. For example, the following code snippet shows how you can download a model and use it for
making a local prediction using the BigML bindings for Python:

```python
from bigml.model import Model
from bigml.api import BigML
api = BigML()
model = api.get_model("model/502fdbff15526876610002615",
    query_string="only_model=true;limit=-1")

local_model = Model(model)
prediction = local_model.predict({"petal length": 3, "petal width": 1})
```

For more information on using models through the BigML bindings, please refer to BigML bindings
documentation.

### 1.7.6 Descriptive Information

Descriptive information is what allows you to describe a prediction so you can find it later and easily
recognize it among other predictions.

Each prediction has an associated name, description, category, and tags. You can find a brief de-
scription for each concept in the following sections. In Figure 1.87, you can see the options that the
More info panel gives to edit them.

![Figure 1.87: Edit predictions]

Copyright © 2023, BigML, Inc.
1.7.6.1 Name

If you do not specify a name for your predictions, BigML assigns a default name depending on the type of predictions:

- **Single predictions**: the name always follows the structure “Prediction for \(<objective\) field name>”
- **Batch predictions**: BigML combines your prediction dataset name and the model name: “Batch prediction of \(<model\) name> with \(<dataset\) name>”.

Predictions names are displayed on the list view and also on the top bar of a prediction view. Predictions names are indexed to be used in searches. You can rename your predictions at any time from the More info panel.

The name of a prediction cannot be longer than 256 characters. More than one prediction can have the same name even within the same project, since they are automatically assigned unique internal identifiers.

1.7.6.2 Description

Each model prediction also has a description that it is very useful for documenting your Machine Learning projects. Predictions take the description from the models used to create them.

Descriptions can be written using plain text and also markdown\(^{21}\). BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 1.88.)

![Markdown editor for evaluations descriptions](https://en.wikipedia.org/wiki/Markdown)

Figure 1.88: Markdown editor for evaluations descriptions

Descriptions cannot be longer than 8192 characters and can use almost any character.

1.7.6.3 Category

Each prediction has associated a category taken from model used to create it. Categories are useful to classify predictions according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

A prediction category must be one of the categories listed on table Table 1.2.

---

\(^{21}\)https://en.wikipedia.org/wiki/Markdown
Table 1.2: Categories used to classify predictions by BigML

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace and Defense</td>
</tr>
<tr>
<td>Automotive, Engineering and Manufacturing</td>
</tr>
<tr>
<td>Banking and Finance</td>
</tr>
<tr>
<td>Chemical and Pharmaceutical</td>
</tr>
<tr>
<td>Consumer and Retail</td>
</tr>
<tr>
<td>Demographics and Surveys</td>
</tr>
<tr>
<td>Energy, Oil and Gas</td>
</tr>
<tr>
<td>Fraud and Crime</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Higher Education and Scientific Research</td>
</tr>
<tr>
<td>Human Resources and Psychology</td>
</tr>
<tr>
<td>Insurance</td>
</tr>
<tr>
<td>Law and Order</td>
</tr>
<tr>
<td>Media, Marketing and Advertising</td>
</tr>
<tr>
<td>Miscellaneous</td>
</tr>
<tr>
<td>Physical, Earth and Life Sciences</td>
</tr>
<tr>
<td>Professional Services</td>
</tr>
<tr>
<td>Public Sector and Nonprofit</td>
</tr>
<tr>
<td>Sports and Games</td>
</tr>
<tr>
<td>Technology and Communications</td>
</tr>
<tr>
<td>Transportation and Logistics</td>
</tr>
<tr>
<td>Travel and Leisure</td>
</tr>
<tr>
<td>Uncategorized</td>
</tr>
<tr>
<td>Utilities</td>
</tr>
</tbody>
</table>

1.7.6.4 Tags

A prediction can also have a number of **tags** associated with it that can help to retrieve it via BigML API or to provide predictions with some extra information. Your prediction inherits the tags from the model use to create it. Each tag is limited to a maximum of 128 characters. Each prediction can have up to 32 different tags.

1.7.7 Model Predictions Privacy

The link displayed in the **privacy panel** is the private URL of your prediction, so only a user logged into your account is able to see it. Neither single predictions nor batch predictions can be shared from your BigML Dashboard by sharing a link, as you can do with other resources.

![Figure 1.89: Private link of a prediction](https://example.com/private-url.png)
1.7.8 Moving Model Predictions to Another Project

When you create a prediction it will be assigned to the same project where the original model is located. You cannot move predictions between projects as you do with other resources.

1.7.9 Stopping Models Predictions

**Single predictions** are synchronous resources, so you cannot cancel them during the creation since you get the result immediately.

**Batch predictions** are asynchronous resources, so you can stop the creation before the task is finished. You can use the DELETE option from the 1-click action menu (Figure 1.90) or from the pop up menu on the prediction list view. (See Figure 1.91.) You can see in Figure 1.91 that the objective field column has the label PROCESSING to indicate the batch prediction is still in progress. If you stop the prediction during its creation, you will not be able to resume the same task again, so if you want to create the same prediction, you will have to re-start a new task.

![Figure 1.90: Stop prediction from the 1-click menu](image)

![Figure 1.91: Stop prediction from the predictions list view](image)

1.7.10 Deleting Model Predictions

You can **DELETE** your **single or batch predictions** from the predictions view, using the 1-click action menu (see Figure 1.92) or using the pop up menu on the predictions list view (see Figure 1.93.)
A modal window will be displayed asking you for confirmation. Once a prediction is deleted, it is permanently deleted and there is no way you (or even the IT folks at BigML) can retrieve it.

1.8 Consuming Models

In the previous sections, we have described how you can create BigML models, configure them, use them to make predictions, and more. This section will introduce a number of BigML features that enable interesting ways of taking advantages of BigML models: exporting them locally, and using them programmatically via the BigML REST API and Bindings.

1.8.1 Exporting and Downloading Models

You can export your model in a variety of programming languages, in PMML or in Excel format. Just click on the download icon in the top menu and select your preferred option.
The main goal of downloading your model in a programming language is to make local predictions faster and at no cost. (See Subsection 1.7.5.2.)

If you are interested in exporting only the rules of a particular branch, you click on the branch leaf and press **SHIFT** from your keyboard. An icon to export the branch rules will appear below the prediction path. (See Figure 1.96.) Release the frozen view by pressing **ESCAPE**.

### 1.8.2 Using Models Via the BigML API

Models have full citizenship in the BigML API. This means you can programmatically create, update, list, delete, and use them for predictions. For example, this is how you can create a model from the...
command line with custom values for a few available arguments. This will require you have properly set the `BIGML_AUTH` environment variable to contain your authentication credentials:

```
curl "https://bigml.io/model?$$BIGML_AUTH" \\
-X POST \\
-H 'content-type: application/json' \\
-d '{"dataset": "dataset/4f66a8063ce840c5000006", 
    "name": "my model", 
    "range": [25, 125]}'
```

For more information on using models through the BigML API, please refer to the model REST API documentation.

### 1.8.3 Using Models Via the BigML Bindings

BigML bindings provide a convenient way to access BigML REST API from your language of choice. They offer a higher-level view of BigML Machine Learning resources and algorithms in a number of languages, including Python, Node.js, Java, Swift, and Objective-C. For example, this is how you can create a model in Python using BigML bindings:

```
from bigml.api import BigML
api = BigML()
prediction = api.create_model("dataset/573d997058a27e0f620038df", 
                             "range": [1, 10], 
                             {"name": "my Model"})
```

For more information on using models through the BigML API, please refer to the BigML bindings documentation.

### 1.9 Model Limits

BigML imposes a few limits on the characteristics of a model that it can handle:

- **Fields**: there is no enforced limit to the number of fields that can be present in a model.
- **Instances**: there is no enforced limit to the number of instances that can be handled.
- **Classes**: a maximum number of 1,000 distinct classes per field is allowed.
- **Term-tokens**: BigML can handle up to 1,000 tokens total. In case multiple text fields are defined, then the token limit per field is divided by the number of text fields.
- **Term-full terms**: BigML can handle up to 256 characters total.
- **Items**: a maximum of 10,000 items per field is allowed.
- **Node threshold**: BigML supports a value between 3 and 2,000.

### 1.10 Descriptive Information

Each model has an associated `name`, `description`, `category`, and `tags`. A brief description follows for each concept. In Figure 1.97, you can see the possibilities that the More Info menu option gives to edit them.
1.10.1 Model Name

Each model has an associated name that is displayed on the list view and also on the top bar of a model view. Model names are indexed to be used in searches. When you create a model, by default, it gets the name of the dataset used to create it. The name of a model cannot be longer than 256 characters. There is no restriction on the characters that can be used in a model name. More than one model can have the same name even within the same project, since they are automatically assigned unique internal identifiers.

1.10.2 Description

Each model also takes the description from the dataset used to create it. Having a description can be very useful for documenting your Machine Learning projects. Descriptions can be written using plain text and also markdown. BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 1.98.)

---

22https://en.wikipedia.org/wiki/Markdown
Descriptions cannot be longer than 8192 characters and can use almost any character.

1.10.3 Category

Each model has an associated category taken from the dataset used to create it. Categories are useful to classify models according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

A model category must be one of the 24 categories listed on table Table 1.3.
Table 1.3: Categories used to classify models by BigML

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace and Defense</td>
</tr>
<tr>
<td>Automotive, Engineering and Manufacturing</td>
</tr>
<tr>
<td>Banking and Finance</td>
</tr>
<tr>
<td>Chemical and Pharmaceutical</td>
</tr>
<tr>
<td>Consumer and Retail</td>
</tr>
<tr>
<td>Demographics and Surveys</td>
</tr>
<tr>
<td>Energy, Oil and Gas</td>
</tr>
<tr>
<td>Fraud and Crime</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Higher Education and Scientific Research</td>
</tr>
<tr>
<td>Human Resources and Psychology</td>
</tr>
<tr>
<td>Insurance</td>
</tr>
<tr>
<td>Law and Order</td>
</tr>
<tr>
<td>Media, Marketing and Advertising</td>
</tr>
<tr>
<td>Miscellaneous</td>
</tr>
<tr>
<td>Physical, Earth and Life Sciences</td>
</tr>
<tr>
<td>Professional Services</td>
</tr>
<tr>
<td>Public Sector and Nonprofit</td>
</tr>
<tr>
<td>Sports and Games</td>
</tr>
<tr>
<td>Technology and Communications</td>
</tr>
<tr>
<td>Transportation and Logistics</td>
</tr>
<tr>
<td>Travel and Leisure</td>
</tr>
<tr>
<td>Uncategorized</td>
</tr>
<tr>
<td>Utilities</td>
</tr>
</tbody>
</table>

1.10.4 Tags

A model can also have a number of tags associated with it that can help to retrieve it via BigML API or to provide models with some extra information. Models inherit the tags from the dataset used to create them.

Each tag is limited to a maximum of 128 characters. Each model can have up to 32 different tags.

1.10.5 Counters

For each model, BigML also stores a number of counters to track the number of other resources that have been created using the model. In the model view, you can see a menu option that displays these counters. It also allows you to quickly jump to all the resources of one type that have been created with this model as shown in Figure 1.99.
1.11 Models Privacy

In this section, you will learn about the privacy options that BigML provides for models, including how to share a resource privately, or make it public through the BigML BigML Gallery.

Privacy options for a BigML model can be defined in the More Info menu option, displayed in Figure 1.100. There are three levels of privacy for BigML models:

- **Private**: only accessible by authorized users (the owner and those who have been granted access).
- **Shared**: by enabling the secret link you will get two different links to share your model. The first one is a sharing link that you can copy and send to others so they can visualize and interact with your model. The second one is a link to embed your model directly on your web page.
- **Public**: accessible and clonable as private resources by any user. Public resources are listed in BigML Gallery. (See Section 1.12).

1.12 Models in the BigML Gallery

This section will introduce the BigML BigML Gallery, which provides a marketplace for Machine Learning models and workflows running on the BigML platform. Models and workflows published in the BigML Gallery may have a price or be free.

Using the BigML Gallery, you can publish your models so they are available to other users, or you can clone models that other users made public.
1.12.1 Publishing Models in the Gallery

You can make your model public for other BigML users. To accomplish this, publish your model in the BigML Gallery following these steps:

1. Provide a description for your model, which is mandatory for models in the BigML Gallery. We recommend you also assign a proper name, category, and tags to the model. (See Section 1.10 to learn how to update your model’s descriptive information.)

2. Choose how to publish your model, either as a **Black Box** or a **White Box** model:

   - Black Box models allow BigML users to make predictions with it, but they are not available to purchase or clone. You can publish a Black Box model by clicking the black lock icon as shown in Figure 1.101.

A modal window will automatically appear asking for confirmation. Decide whether to share your model for predictions for free, or sell it and obtain benefits per prediction. Set the price you consider appropriate by just moving the price slider and click **Update**. (See Figure 1.102.)

---

Figure 1.101: Black Box models let other BigML users make predictions

Figure 1.102: Set a price for other users to make predictions with your model
• White Box models allow users to clone or purchase the full model. Click the white lock icon to share your model as White Box. (See Figure 1.103.)

Figure 1.103: White Box models let other users to clone or purchase the full model

A modal window will automatically appear asking for confirmation. Decide whether to share your model for free, or charge a price either by cloning the full model or by making predictions with your model. Set the prices you consider appropriate by just moving the price slider and click Update. (See Figure 1.104.)

Figure 1.104: Set a price for other users to clone your model and make predictions with it

3. After publishing your model, the gallery link automatically appears in the privacy panel and the
status changes from “Private” to “Black Box” or “White Box”, depending on your choice. You can change the set price anytime by clicking on the edit icon. (See Figure 1.105.)

You can only publish a model when the model is yours. If you are using a model previously cloned from another user, BigML will display a warning stating you cannot share that model or sell it. (See Figure 1.106.)

1.12.2 Cloning Models From Gallery

BigML lets you use models that are public in BigML Gallery. These models are publicly available because other users have shared them. Some of the models available are free of charge and others have a specific cost. The owner of the model decides its cost. (See Subsection 1.12.1 for more details on how to share or sell your models.)

1. To import a model from the Gallery into your Dashboard, first you need to clone it. The link that gives you access to BigML public Gallery is on the very top menu on the left. (See Figure 1.107.)

2. Select “Models” on the top menu. Then click the model you are interested in. (See Figure 1.108.) Clone it by clicking the Buy label. If the model is free of charge, click the Free label, which changes to Buy when you mouse over it, but actually BigML will not charge you anything.
3. If the dataset that was used to create the model is also available in the gallery, a modal window (see Figure 1.109) will be displayed asking you if you want to add it to your current purchase. Click the Yes button to confirm you want to clone it along with the model; No to just buy the model.

4. A modal window (see Figure 1.110) will be displayed asking you for confirmation.
5. Your new model goes directly to your BigML Dashboard. Notice that any task performed from a cloned model from BigML Gallery is free of charge, no matter the size of the task to be performed.

1.13 Moving Models to Another Project

When you create a model it will be assigned to the same project where the original dataset used to create it belongs to.

Models can only be assigned to a single project. However, you can move models between projects. The menu option to do this can be found in two places:

1. In the model detail view, within the 1-click actions menu (see Figure 1.111).

![Figure 1.110: Modal window to confirm you want to clone this model]

![Figure 1.111: 1-click menu option to move models]

2. In the model list view, within the pop up menu (see on Figure 1.112).
1.14 Stopping Model Creation

You can also stop a model’s creation process before the task is finished from the 1-click menu (Figure 1.113) or from the pop up menu in the models list view (Figure 1.114).

Figure 1.112: Pop up menu option to move models

Figure 1.113: 1-click menu option to stop a model’s creation
Chapter 1. Models

Figure 1.114: Pop up menu option to stop a model’s creation

A modal window (Figure 1.115) will be displayed asking you for confirmation.

![Modal window](image)

Figure 1.115: Menu option to stop a model’s creation

Note: if you stop the model during its creation, you will not be able to resume the same task. If you want to create the same model, you will have to start a new task.

1.15 Deleting Models

You can delete your models using two different options:

- From the model view, using the 1-click action menu (Figure 1.116).
Figure 1.116: Menu option to delete a model

• Using the pop up menu on the model list view (Figure 1.117).

Figure 1.117: Model deletion pop up menu option

A modal window (see Figure 1.118) will be displayed asking you for confirmation. Once you delete a model, it is deleted permanently and there is no way you (or even the IT folks at BigML) can retrieve it.

Figure 1.118: Model deletion modal window
1.16 Takeaways

This chapter explained models in detail. We conclude it with a list of key points:

- BigML models are human friendly as opposed to many other Machine Learning models. They provide a set of rules organized in a tree structure that are easy to understand for non-experts.
- You can use BigML models to solve classification and regression problems.
- BigML models support any type of fields as input fields (categorical, numeric, date and time, text, and items fields).
- BigML Models are not affected by uninformative or redundant fields.
- Normalization is not needed.
- To build a BigML model you just need a dataset. (See Figure 1.119).
- A BigML model can be an input to an evaluation, to a prediction, or to a batch prediction. (See Figure 1.119).
- A BigML model can be the output of a cluster [8]. (See Figure 1.119).
- You can create a BigML model with just 1-click or configure it as you wish. BigML models are easy to tune without having to configure difficult parameters.
- You can also create models using BigML REST API or the BigML bindings for your language of choice.
- If you do not specify any objective field, BigML will use the last valid field in your dataset.
- You can choose three different pruning strategies when building your BigML model to avoid overfitting: smart pruning, statistical pruning, or no statistical pruning.
- By default, BigML models do not consider missing values when choosing splitting rules, but you can explicitly include them.
- You can set the maximum number of nodes you want for your BigML model. Greater number of nodes will grow more complex trees that will perform better with the training data at the expense of generalization.
- To deal with imbalanced datasets, BigML provides three different options to assign specific weight your instances: balance objective, objective weights, weight field.
- For classification problems, the confidence is a measure of the model’s certainty when predicting a class at a certain node.
- For regression problems, the expected error is a measure of the expected error at a node.
- You can visualize BigML models in an interactive decision tree’s structure or with the Sunburst view.
- BigML provides a summarized view of your model, including: the data distribution, prediction distribution, field importance, and the rules summary.
- You can easily see which fields in your dataset have more impact on predictions by clicking in the model’s summary report.
- You need to evaluate your model’s performance using data that the model has not seen before. Evaluating a model is a key step to understand if that model is satisfactory or needs training adjustments. In the latter case, you can use different creation options until you achieve the desired evaluation results.
- You can download your model to your preferred programming language to use it in your local environment, and make predictions faster at no cost.
- Once you get a satisfactory model, you can make single or batch predictions using your model.
• BigML provides local predictions from the BigML Dashboard for single instance predictions. Local predictions allow you to get a real-time prediction without consuming any credits or requiring an internet connection.

• BigML batch predictions allow you to make simultaneous predictions for multiple instances. For batch predictions, you always get a CSV file and an optional output dataset.

• You can furnish your model with descriptive information (name, description, tags, and category).

• You can clone an existing model from BigML Gallery.

• You can share your model in the BigML Gallery as Black Box or White Box, so other BigML users can clone your model and make predictions with it.

• You can stop the model's creation before the task is finished.

• You can permanently delete a model.

Figure 1.119: Model Workflows
Ensembles

2.1 Introduction

There are multiple Machine Learning problems that can be solved using supervised Machine Learning techniques. Some of these problems require to predict an output variable (objective field) given a number of input variables (input fields). These problems can be divided into classification and regression depending on whether you need to predict a category (label or class) or a continuous value (a real number), respectively. To learn more about concrete use cases for both problems refer to Section 1.1.

However, most of these problems cannot be solved with a single model. One of the pitfalls of Machine Learning is that the algorithm has the potential to overfit your data, so its performance on your training data is very good, but it does not generalize well to new data, which makes single tree models poorer predictive models. Ensembles avoid this disadvantage.

An ensemble is a collection of multiple decision trees which are combined to create a stronger model with better predictive performance. An ensemble of models built on samples of the data can become a powerful predictor by averaging away the errors of each individual model. Generally, ensembles perform better than a single decision tree because they are less sensitive to outliers in your training data, which helps them mitigate the risk of overfitting and generalize better when applied to new data.

Depending on the nature of your data and the specific values for the ensemble parameters, you can significantly boost the performance over using a single model. For a technical explanation of why ensembles perform better, please see this tutorial paper from our Chief Scientist, Tom Dietterich [2].

BigML currently provides three types of ensembles:

- **Bagging** (also known as Bootstrap Aggregating): this algorithm builds each single model composing the ensemble from a random subset of the dataset instances. By default the samples are taken using a rate of 100% with replacement (this is explained in Subsection 2.4.9). While this is a simple strategy, it often outperforms more complex strategies. Read more about Bagging.

- **Random Decision Forests**: similar to Bagging but it adds an additional element of randomness by choosing a random subset of features at each tree split. Read more about Random Decision Forests.

- **Boosted Trees**: (or gradient boosted trees) this algorithm sequentially builds a set of weak learners and then combines their outputs in an additive manner to get a final prediction. In every boosting iteration, each single model tries to correct the errors made in the previous iteration by optimizing a loss function. Read more about Gradient Boosting.

---

See Subsection 2.4.3 for an explanation of each algorithm.

This chapter contains comprehensive description of BigML's ensembles including how they can be created with 1-click (Section 2.3), all configuration options available (Section 2.4), and the visualization provided by BigML (Section 2.5). Once you create an ensemble, you can get a report for each field importance (see Subsection 2.2.2), and a heatmap chart, known as Partial Dependence Plot (Section 2.5), to visualize the impact of your input fields on predictions. See Section 2.6 for an explanation of how ensembles can be used to make predictions. Moreover, you can also export your ensembles in different formats to make local predictions faster at no cost (Subsection 2.7.1), move your ensembles to another project (Section 2.11), or delete them permanently (Section 2.13). The process to evaluate your ensemble's predictive performance in BigML is explained in a different chapter (Chapter 7).

In BigML, the third tab of the main menu of your Dashboard allows you to list all your available ensembles. In the ensemble list view (Figure 2.1), you can see, for each ensemble, the Dataset it was created from, the ensemble's Name, Type (either classification or regression), Objective (objective field name), Age (time elapsed since it was created), Size, and number of predictions, batch predictions, or evaluations that have been created using that ensemble. The SEARCH menu option in the top right corner of the ensemble list view allows you to search your models by name.

By default, when you first create an account at BigML, or every time that you start a new project, your list view for ensembles will be empty. (See Figure 2.2.)

Finally, in Figure 2.3 you can see the icon used to represent a model in BigML.
2.2 Understanding Ensembles

In this section, we are going to describe a few internal details about ensembles and how BigML implements them. Specifically, since ensembles are based on BigML models, all the information provided in Section 1.2 also applies to ensembles, unless overridden here.

BigML grows ensembles in a very similar way to how it grows simple models (see Section 1.2). In particular, BigML does not use dataset streaming (see Subsection 1.2.1) for ensembles, thus requiring the entire dataset to be loaded into memory. This choice is motivated by the intrinsic behavior of growing ensembles, which aggressively uses sampling to be able to create multiple significant models from the same dataset.

2.2.1 Decision Forests Versus Boosted Trees

BigML offers three types of algorithms, Bagging, Random Decision Forests (which are included under the same ensemble type in BigML called Decision Forests), and Boosted Trees. The following subsections contain some technical details of the main commonalities and differences for these methods. To read about the situations under which is better to use one method or another please refer to Subsection 2.4.3.

All ensemble methods have in common that they are composed by several single trees and their output is combined to yield a final prediction. The ensemble always has a better performance than each of the individual learners it is composed of.

The main difference between Decision Forests and Boosted Trees is the way single trees are grown and the way their predictions are combined to get the final ensemble prediction. See the following subsections for a detailed explanation.

2.2.1.1 Single trees

Each single tree in Decision Forests tries to predict the objective field using a certain level of randomness, either by selecting a random percentage of the dataset instances (Bagging) and/or by selecting a random subset of the input fields at each split (Random Decision Forests). However, in Boosted Trees, each single learner, instead of predicting the objective field, tries to learn from the mistakes made by the previous model by fitting a gradient step towards minimizing the error of the previous classifier. For regression problems, the error is the usual squared error (the squared difference between the true objective and the current prediction). For classification problems, a tree is trained for each class in each iteration. The scores from the trees are normalized using the softmax function\(^6\) to obtain a probability distribution over classes given a datapoint. The error is the difference between this distribution and the “true distribution” over classes for the datapoints, which is one for the correct class and zero for all others.

Another characteristic for Boosted Trees is that they have a weight associated with each tree which measures their importance to calculate the final ensemble prediction. The weights are chosen via a line search\(^7\), where possible weights are evaluated on a group of test points to find a weight that is near-optimal. Test points for the line search are randomly selected from the training data. Lastly, the weights are multiplied by the learning rate (see Subsection 2.4.7.2) to calculate the final weight. Read more about choosing weights using line search here\(^8\).

---

\(^6\)https://en.wikipedia.org/wiki/Softmax_function

\(^7\)https://en.wikipedia.org/wiki/Line_search

\(^8\)https://en.wikipedia.org/wiki/Gradient_boosting#Gradient_tree_boosting
2.2.1.2 Ensemble predictions: confidence, probability and expected error

For Decision Forests, single tree predictions are averaged to get a final prediction. The same quality measures obtained when building a single model are returned for Decision Forest predictions: confidence and probabilities, for classification problems, and expected error, for regression problems. Find the calculations details for each measure in Subsection 1.2.6. For regression ensembles, all the single trees predictions are averaged to get a single prediction. For classification ensembles, all the per-class confidences and probabilities are averaged taking into account all the trees in the ensemble. The class with the highest confidence or probability is returned. There is an additional technique to calculate predictions for classification problems called “votes”. It is based on the percentage of trees voting for each class in the ensemble to select the winner class. See Subsection 2.6.3.2 for a detailed explanation about probabilities, confidences and votes to calculate predictions.

For Boosted Trees, the single model predictions are additive rather than averaged. For boosted trees you only get the class probabilities in the case of classification ensembles, neither the confidence or the expected error can be calculated. For regression ensembles the final prediction is generated by calculating the sum of each tree prediction multiplied by its boosting weight (see the explanation for boosting weight in Subsection 2.2.1.1). The expected error cannot be calculated for Boosted Trees so there is not quality measure returned for regression problems. Predictions for classification ensembles are similar, but separate weighted sums are found for each objective class. The resulting vector of weighted sums is then transformed into class probabilities using the softmax function<sup>9</sup>. Hence, the probability for each of the classes in the objective field is returned to measure the prediction quality for boosting.

2.2.2 Field Importance

As with individual decision trees, the field importance for ensembles provides a measure of how important a data field is relative to the others. It is computed by taking a weighted average of how much each field reduces the predicted error of the tree at each split (more details in Subsection 1.2.5). For individual trees this measure can be misleading as it assumes that the tree structure is correct, but for ensembles it is a more meaningful measure. (See Figure 2.4).

Figure 2.4: Field importance for ensembles

To visualize the marginal contribution of a field in the ensemble predictions, BigML offers Partial Depen-

<sup>9</sup>https://en.wikipedia.org/wiki/Softmax_function
2.2.3 Ensembles with Images

BigML ensembles do not take images as input directly, however, they can use image features as those fields are numeric.

BigML extracts image features at the source level. Image features are sets of numeric fields for each image. They can capture parts or patterns of an image, such as edges, colors and textures. For information about the image features, please refer to section Image Analysis of the Sources with the BigML Dashboard\(^\text{10}\).[11]

![Figure 2.5: A dataset with images and image features](https://static.bigml.com/pdf/BigML_Sources.pdf)

As shown in Figure 2.5, the example dataset has an image field `image_id`. It also has image features extracted from the images referenced by `image_id`. Image feature fields are hidden by default to reduce clutter. To show them, click on the icon “Click to show image features”, which is next to the “Search by name” box. In Figure 2.6, the example dataset has 160 image feature fields, called *Wavelet subbands*.

\(^\text{10}\)https://static.bigml.com/pdf/BigML_Sources.pdf
From image datasets like this, ensembles can be created and configured using the steps described in the following sections. All other operations including prediction, evaluation applies too.

### 2.3 Creating Ensembles with 1-Click

To create an ensemble in BigML you have two options: either the BigML unique 1-click feature which provides a convenient way to quickly train an ensemble from a dataset, or configuring a number of options that give you fine control on how the ensemble is created. This section will guide you through the process of creating an ensemble with just 1-click.

By far, the easiest and quickest way to create an ensemble is using the 1-CLICK MODEL option that is available in the 1-click action menu. (See Figure 2.7.)

Figure 2.7: 1-click ensemble from dataset detail view

Alternatively, you can select the 1-CLICK MODEL option in the 1-click action menu from the dataset list view. (See Figure 2.8.)
Chapter 2. Ensembles

Figure 2.8: 1-click ensemble from dataset list view

Both options build a new ensemble using default values for all available configuration options (see Section 2.4). This can give you a quick starting point to understand how your ensembles behaves and how it can be improved.

Creating an ensemble may take a variable time, depending on how big your dataset is, your subscription plan, etc. Once your ensemble is ready, it will be automatically displayed on your BigML Dashboard and you will be able to explore it through BigML visualizations (see Section 2.5), and using it for evaluations (see Chapter 7) or predictions (see Section 2.6.)

2.4 Ensemble Configuration Options

While 1-click creation (see Section 2.3) provides a convenient and easy way to create an ensemble from a dataset, there are cases when you want more control. This section will focus on the options that BigML offers to configure its internal algorithms for BigML ensembles.

You can set a number of parameters that affect the way BigML creates ensembles from a dataset. Such parameters can be grouped in two categories:

- Parameters that are permanently associated to the dataset, such as its objective field and preferred fields. Once you provide a value for a dataset's permanent parameters, they will be used as a default value for the creation of ensembles from that dataset.

- Parameters that only affect the ensemble that is currently being created and that you are expected to set each time, such as included/excluded fields, and a number of configuration options that are described below. The objective field can also be specified on a per-ensemble basis, if you do not want to tie it to the dataset as described above.

Set a dataset's permanent parameters by clicking on the edit button that is displayed when you hover on the dataset's fields. This opens a modal dialog where you can set some of the field properties (See Figure 2.9).
Click on the preferred field button to make that field non-preferred.

Click on the objective field button to make that field the new objective field.

To access the configuration panel, select the CONFIGURE ENSEMBLE menu option located in the configuration menu of your dataset's detail view. (See Figure 2.10.)

When the configuration panel is displayed, you can:

- Select or deselect individual fields for them to be included in or excluded from the ensemble computation.
- Change the objective field used for the ensemble to be created.
- Manually configure a number of configuration options or automatically optimize these options.

Note: when the configuration panel is displayed, the edit is not visible, so you cannot set the dataset's permanent properties.
Configuration options are the same for ensembles as for models (Section 1.4) plus a few more: type of algorithm (Decision Forests—which includes bagging and random decision forests—and Boosted Trees), number of models, random candidates, and the Boosting parameters (number of iterations, early stopping, learning rate and setp out of bag). Sampling options are also important for the configuration of ensembles. (See Subsection 1.4.7).

You can find a detailed explanation of the configuration options below.

### 2.4.1 Objective Field

Also known as “target field”, the **objective field** is the output variable you want to predict.

Select your objective field in BigML in either of two ways. Specify the objective field each time you create an ensemble from the configuration panel or set a field as the default objective for all ensembles by clicking the **edit** button and then the **objective field** button.

By default, BigML will use the last valid field in your dataset as objective, with the exemption fields of type text and items that cannot be used as objective.

![Figure 2.11: Ensemble objective field](image)

### 2.4.2 Automatic Optimization

You can turn on the **Automatic optimization** option so BigML will automatically tune the parameters of your ensemble (see Figure 2.12).
The high number of possible combinations for parameter values makes it difficult to find the optimum configuration since the combinations that lead to a poor result outnumber the ones that result in a satisfying performance. Hand-tuning different configurations is a time-consuming process that requires a high level of expertise and intuition. To combat this problem, BigML offers first-class support for automatic ensemble parameter optimization.

Behind the scenes, BigML uses the same technology for ensemble parameter optimization as the one used for OptiML. If you want to know more about the technical details, please read the Chapter 2 of the document OptiML with the BigML Dashboard [10].

When you turn on the **Automatic optimization** option, all the ensemble parameters will be disabled (because they will be automatically optimized), except the **Missing splits** and the **Weights** parameters which you can manually configure (see Subsection 2.4.6.2 and Subsection 2.4.8).
Chapter 2. Ensembles

Figure 2.13: Configure the missing splits and weights for your ensemble

**Note:** there is a maximum of 256 trees per ensemble that will be tried out during the optimization process. If you think that your ensemble needs a higher number of models, you can manually configure it.

Since the optimization process can take some time, BigML offers two configurable parameters to limit the time to create the optimized ensemble: a training duration (see Subsection 2.4.2.1) and the ensemble candidates (see Subsection 2.4.2.2).

### 2.4.2.1 Training duration

The scale parameter to regulate the ensemble runtime. It’s set as an integer from 1 to 10. It indicates the user preference for the amount of time they wish the optimization to take. The higher the number, the more time that users are willing to wait for possibly better ensemble performance. The lower the number, the faster that users wish the ensemble training to finish. The default value is set to 5.

The training duration is set in a scale. The actual training time depends on the dataset size, among other factors.
2.4.2.2 Ensemble candidates

The maximum number of different ensembles (i.e., ensembles using a unique configuration) to be trained and evaluated during the optimization process. The default number is 128 ensembles which is usually enough to find the best ensemble, but you can set it from 4 up to 200. Only the top-performing ensemble will be returned. If the training duration is very low (see Subsection 2.4.2.1) given the dataset size, it is possible that not all the ensemble candidates will be tried out.

2.4.3 Type

This option allows you to choose between two methods to build your ensemble: Decision Forests and Boosting. (See Figure 2.16.) By selecting Decision Forests you can either build a Bagging or a Random Decision Forests ensemble. By default, Decision Forests take a random subset of instances from the dataset, which creates a Bagging ensemble. You can also add an additional element of randomness by choosing random features at each split (see Subsection 2.4.6.4) so you get a Random Decision Forest ensemble. By choosing Boosted Trees BigML builds gradient boosting trees. Read a technical description about Decision Forests and Boosted Trees in Subsection 2.2.1.

There is not an easy answer for the question of which method yields the best results. Usually you will have to try and test all options. However, depending on your dataset’s characteristics, you can sometimes have an initial idea of which may perform better.
In Boosted Trees, the effect of additional trees is essentially an expansion of the hypothesis space in a way that it is not for Decision Forests. So if you expect the decision function to be very complex and you have a lot of data, boosting may work better than Decision Forests.

On the other hand, if you have a noisy domain, where overfitting is a concern, Decision Forests may be a better option than Boosted Trees, since boosting is more vulnerable to label noise.

Finally, if you have what you suspect is an "easily learnable" function, the additional power offered by Boosted Trees, or even Random Decision Forests, may not help; a more simple method, like Bagging or even models may perform better than the other two options.

---

**2.4.4 Number of Models**

This option allows you to configure the number of decision trees to create your ensembles when you select Decision Forest as your ensemble type (see Subsection 2.4.3). By default, the number of models is set to 10 and the maximum allowed is 1,000 trees. (See Figure 2.17.) For Boosted Trees the number of models will be determined by the number of iterations (see Subsection 2.4.5) with a maximum of 2,000 trees per ensemble.

Generally, increasing the number of trees will yield better results. Furthermore, there is no downside except higher computational time. The situations where more models are likely to provide the most improvements are those where the dataset is not very large (e.g., in the thousands of instances or less), the data is very noisy, and (in the case of Random Decision Forests) when there are many correlated features that are all somewhat useful.

Take into account that each additional model tends to deliver less marginal improvement, so if the difference between nine and ten models is very small, it is very unlikely that an eleventh model will make a big difference.
2.4.5 Number of Iterations

This parameter sets the maximum number of iterations to be performed when **Boosted Trees** is selected. For regression ensembles, one boosted tree will be generated for every iteration. For classification ensembles, however, “N” trees will be generated for every iteration where “N” is the number of classes in the objective field.

By default, the number of iterations is 10 and the maximum allowed is 1,000 with a limit of 2,000 maximum single models built. If you set 1,000 iterations using a dataset with more than two classes for the objective field, in the case the ensemble reaches the maximum of 2,000 trees, it will stop.

**Note:** when one of the early stopping options is enabled (see Subsection 2.4.7.1), the final number of iterations may be lower than the number of iterations configured.
2.4.6 Trees

Since ensembles are composed of several decision trees, some of the parameters that can be configured for BigML models, can also be applied for ensembles like Missing splits and the Node threshold. You can also use the Randomize parameter to select a random subset of your fields at each split and create a Random Decision Forest. You can find a detailed explanation of the three parameters in the subsections below.

2.4.6.1 Pruning

If pruning is enabled, BigML determines whether each tree split increases the confidence (for classification ensembles) or decreases the expected error (for regression ensembles). If it does not, then the split is pruned away. As explained in Subsection 1.2.4, pruning strategies are key to avoid overfitting, a phenomenon that reduces an ensemble's ability to generalize. The statistical pruning is only available for decision forests (see Subsection 2.4.3).

In BigML you can choose three different strategies for pruning (Figure 2.19):

- **Smart Pruning**: considers pruning the nodes with less than 1% of the instances.
- **Statistical Pruning**: considers every node for pruning.
- **No Statistical Pruning**: deactivates pruning altogether.

By default, BigML uses Smart Pruning to create your decision forests.

2.4.6.2 Missing Splits

When training an ensemble, BigML may encounter missing values, which can be either considered or ignored for the definition of splitting rules.

To include missing splits in your ensemble, enable the missing splits option. (See Figure 2.19.) If missing values are included in your ensemble, you may find rules with predicates of the following kind:

field x = "is missing" or field x = "y or is missing".

BigML includes missing values following the MIA approach\(^\text{11}\) [14].

By default, BigML does not include missing splits.

2.4.6.3 Node Threshold

Set the node threshold to set a limit to each single tree growth within the ensemble. (See Figure 2.19.) A lower threshold simplifies the ensemble while helping to avoid overfitting. However, it may also have reduce the ensemble’s predictive power compared to deeper ensembles. The ideal number of nodes may depend on the dataset size and the number of features. Larger datasets with many important features may require more complex ensembles. Reducing the number of nodes can also be useful to get an initial understanding of the basic data patterns. Then you can start growing the ensemble from there.

By default, BigML sets a 512 node threshold. Since nodes are computed in batches, on occasion the final number of nodes can be greater than the node threshold. (See Subsection 1.2.2.)

2.4.6.4 Randomize and Random Candidates

The randomize option allows another layer of randomization and it works for Decision Forest and for Boosted Trees. If randomize is enabled, a random subset of the input fields will be selected at each tree split. If you selected Decision Forest as the ensemble type (see Subsection 2.4.3), this will create a Random Decision Forest.

When you click the randomize option, the random candidates option will be enabled so you can configure the number of input fields in the random subset to be consider at each split. BigML provides three options (Figure 2.19):

\(^{11}\)http://oro.open.ac.uk/22531/1/decision_trees.pdf
• **Default**: the number of fields is the square root of the total number of input fields. This is a basic rule which works pretty well in most cases.

• **Number of fields**: this sets a fixed number of the fields to be considered at each split.

• **Ratio of fields**: sets the number of fields to be considered at each split as a percentage of the total number of input fields.

**Note**: for text fields each term (e.g., each word if space is the chosen separator) counts as an individual field.

![Figure 2.19: Trees parameters configuration](image)

### 2.4.7 Boosting Parameters

When you select **Boosted Trees** as the ensemble type (see Subsection 2.4.3), the parameters in the Boosting tab will be enabled: **early stopping** and **learning rate**. See the following subsections for a detailed explanation.

#### 2.4.7.1 Early stopping

The **early stopping** options try to find the optimal number of iterations by testing the single models after every iteration and resulting in an early stop if not significant improvement is made. Consequently, the total iterations for the Boosted Trees, may be lower than the one set in the **Number of iterations** parameter (see Subsection 2.4.5).

You can select one of these three options:

- **Early out of bag**: this option tries to find out the optimal number of iterations by recursively building single trees and testing the out-of-bag samples after every iteration. This option will use the
parameters set in the ensemble sample to build the single trees (see Subsection 2.4.9). If no significant improvement is made, it may result in an early stop. By default this option is enabled. (See Figure 2.20.)

- **Early holdout**: this option tries to find out the optimal number of iterations by recursively building single trees and holding out a portion of the dataset for testing at the end of every iteration. If no significant improvement is made on the holdout, it may result in an early stop. The percentage of the dataset holdout is set to 30% by default, but you can configure it. By default this option is disabled.

- **None**: this option deactivates the early stopping parameter so the total iterations will be the same as the ones specified in the Number of iterations parameter (see Subsection 2.4.5). By default, this option is disabled.

### 2.4.7.2 Learning rate

The **learning rate**, also known as the gradient step, controls how aggressively the boosting algorithm fits the data. You can set values greater than 0 and smaller than 1. Larger values will prevent overfitting, but smaller values generally work better (usually 0.1 or lower). By default it is set to 0.1. (See Figure 2.20.)

---

**Figure 2.20: Boosting parameters**

---

### 2.4.8 Weight Field Options

It is not unusual for a dataset to have an unbalanced **objective field**, where some categories are common and others very rare. For example, in datasets used to predict fraud, usually fraudulent transactions are very scarce compared to regular ones. When this happens, ensembles tend to predict the most frequent values simply because the overall ensemble’s performance metric improves with that approach. However, in cases such as fraud prediction, you may be more interested in predicting rare values rather than successfully predicting frequent ones. In that case, you may want to assign more **weight** to the scarce instances so they are equivalent to the abundant ones.

BigML provides three different options to assign specific weight to your instances.
2.4.8.1 Balance Objective

When you set the balance objective weight (see Figure 2.21), BigML automatically balances the classes of the objective field by assigning a higher weight to the less frequent classes, with the most frequent class always having a weight of 1. This option is only available for classification ensembles. For example, take the following frequencies for each class:

\[
[\text{False, 2000}; \text{True, 50}]
\]

By enabling the Balance objective option, BigML will automatically apply the following weights:

\[
[\text{False, 1}; \text{True, 40}]
\]

In this example, the class “True” is getting forty times more weight as it is forty times less frequent than the most abundant class.

2.4.8.2 Objective Weights

The objective weights option allows you to manually set a specific weight for each class of the objective field. BigML oversamples your weighted instances replicating them as many times as the weight establishes. If you do not list a class, it is assumed to have a weight of 1. Weights of 0 are also valid. This option is only available for classification ensembles. (See Figure 2.21.)

This option can be combined with the Weight field (see Subsection 2.4.8.3). When combining it with the Weight field, both weights are multiplied. For example if you assign a weight of 3 for the “True” class and the weight field assigns a weight of 2 for a given instance labeled as “True”, that instance will have a total weight of 6.

2.4.8.3 Weight Field

The Weight Field option allows you to assign individual weights to each instance by choosing a special weight field. (See Figure 2.21.) It can be used for both regression and classification ensembles. The selected field must be numeric and it must not contain any missing values. The weight field will be excluded from the input fields when building the ensemble. You can select an existing field in your dataset or you may create a new one in order to assign customized weights.

For example, below is a dataset for which we included a field called “Weight” that assign a ten time higher weight to fraudulent transactions in comparison to non-fraudulent ones. BigML provides a powerful tool, the BigML Flatline editor, to add new fields to your dataset, such as a weight field. As an additional example, we could also take into account the transaction “Amount” to calculate the weights, so transactions with higher amounts will have higher weights.

<table>
<thead>
<tr>
<th>Trans. ID</th>
<th>Products</th>
<th>Online</th>
<th>Amount $</th>
<th>Fraud</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxxxxx098</td>
<td>XYZGH</td>
<td>yes</td>
<td>3,218</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxx345</td>
<td>VBHGF</td>
<td>no</td>
<td>1,200</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxx123</td>
<td>UYFHJ</td>
<td>yes</td>
<td>5,000</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxx567</td>
<td>HSNKI</td>
<td>no</td>
<td>390</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxx789</td>
<td>SHSYA</td>
<td>yes</td>
<td>500</td>
<td>TRUE</td>
<td>10</td>
</tr>
<tr>
<td>xxxxxx093</td>
<td>DFSTU</td>
<td>yes</td>
<td>423</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxx012</td>
<td>TYISJ</td>
<td>yes</td>
<td>60,000</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxx342</td>
<td>SJ SOP</td>
<td>no</td>
<td>789</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxx908</td>
<td>IOPKJ</td>
<td>no</td>
<td>9,450</td>
<td>FALSE</td>
<td>1</td>
</tr>
<tr>
<td>xxxxxx334</td>
<td>HIOPN</td>
<td>yes</td>
<td>50,678</td>
<td>TRUE</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2.1: Weight Field example for transactional dataset
Chapter 2. Ensembles

2.4.9 Trees Sampling

The trees sampling is a different concept than the dataset sampling (see Subsection 2.4.10). The dataset sampling is the one present in other BigML resources and it applies sampling just once to the input dataset before building the resource. In the case of the trees sampling, sampling is applied to the dataset as many times as the number of models that compose the ensemble. This way, separate samplings are created for each tree composing the final ensemble.

The default is to sample with a rate of 100% and with replacement, meaning that the same instance can be selected more than once. This ensures a different sampling for each tree. The out of bag option is not available for trees sampling.

How to Get Repeatable Ensembles

If you create two ensembles using the same dataset and same configuration you will see that the individual trees composing each ensemble may be different. This is due to sampling “randomization” that BigML algorithm uses. Although this does not usually affect the predictive performance of your ensembles, there may be situations in which it is desirable to produce exactly the same ensembles. You can achieve this by setting the Sampling option to deterministic (see Subsection 2.4.9.2).

See below an explanation of each sampling option to build the single trees.

2.4.9.1 Rate

The sampling rate is the percentage of instances being extracted from the dataset and included in your per-tree sample. A sampling rate of 100% means that all instances are included; a rate of 10% means that only every tenth instance is included in each single model. This option may take any value between 0% and 100%. You can easily configure the rate by moving the slider in the configuration panel for sampling, or by typing the percentage in the tiny input box, both highlighted in Figure 2.22.

By default, BigML uses a 100% rate combined with replacement (see Subsection 2.4.9.3).
2.4.9.2 Sampling

The **sampling** option represents the type of the sampling process, which can be either random or deterministic. (See Figure 2.22.)

When using deterministic sampling the random-number generator will always use the same seed, producing repeatable results. (See Subsection 2.4.9.)

By default, BigML uses random sampling.

2.4.9.3 Replacement

The **replacement** option controls whether a single instance can be selected multiple times or not. Sampling without replacement ensures that each instance cannot be selected more than once. (See Figure 2.22.)

By default, BigML generates samples with replacement.

![Figure 2.22: Trees sampling for ensembles](image)

2.4.10 Dataset Sampling

Sometimes you do not need all the instances contained in your testing dataset to build your ensemble. If you have a very large dataset, **sampling** may be a good way of getting faster results. (See Figure 2.23.)

The same sampling options described in the Datasets with the BigML Dashboard document [9] to sample datasets, are also available when building BigML ensembles. They are divided in two groups: sampling and advanced sampling options. (See Figure 2.23.)

2.4.10.1 Rate

The sampling **rate** is the frequency of instances being extracted from the dataset and included in your sample. A sampling rate of 100% means that all instances are included; a rate of 10% means 10% of the instances are included. This option may take any value between 0% and 100%. You can easily configure the rate by moving the slider in the configuration panel for sampling, or by typing the percentage in the tiny input box, both highlighted in Figure 2.23.
By default, BigML uses a 100% rate.

2.4.10.2 Range

The sampling range is the subset of the dataset instances from which to sample, e.g., from instance 5 to instance 1,000. The rate will be applied over the range configured.

By default, all instances are included, i.e., the range is (1, num. rows in dataset).

2.4.10.3 Sampling

The sampling option represents the type of the sampling process, which can be either random or deterministic.

When using deterministic sampling the random-number generator will always use the same seed, producing repeatable results.

By default, BigML uses random sampling.

2.4.10.4 Replacement

The replacement option controls whether a single instance can be selected multiple times or not. Sampling without replacement ensures that each instance cannot be selected more than once.

By default, BigML generates samples without replacement.

2.4.10.5 Out of Bag

The out of bag option allows you to include in your sample only those instances that were not selected in the first place, thus effectively inverting the sampling outcome. It is only selectable when a sample is deterministic and the sample rate is less than 100%. The total percentage of instances included in your sample will be one minus the rate (when replacement is not allowed). This can be useful for splitting a dataset into training and testing subsets.

By default, BigML will not use out of bag instances.
2.4.11 Advanced Ordering

Ordering options are relevant to ensure that BigML can correctly determine whether it can take an early split of your dataset to accelerate the training process. In particular, early splitting can only be safely used if the training instances have been previously shuffled. (See Subsection 1.2.3.1.)

If your instances are already shuffled, BigML allows you to choose the linear option. This will make the process of building the ensemble much faster, since it will not required to reshuffle the dataset. If you need to shuffle your instances, BigML provides two options to that aim, deterministic shuffling and random shuffling, which are described below.

Ordering options have no influence on datasets of less than 34GB, since the whole dataset is used to build the ensemble.

By default, BigML uses deterministic shuffling to ensure the same (deterministic) sample of the instances is used and the built ensemble is thus repeatable.

2.4.11.1 Deterministic Shuffling

The deterministic shuffling option ensures that the row shuffling of a dataset is always the same, so that retraining a BigML ensemble from the same dataset yields the same results. (See Figure 2.23.) By default, this option is true.

2.4.11.2 Linear Shuffling

The linear shuffling option is useful when you know that your instances are already in random order. Using linear shuffling, the BigML ensemble will be constructed faster. (See Figure 2.23.) By default, this option is false.
2.4.11.3 Random Shuffling

The random shuffling option will ensure that a different shuffling will be tried each time you train your ensemble. (See Figure 2.23.)

By default, this option is false.

2.4.12 Creating Ensembles with Configured Options

After finishing the configuration of your options, you can change the default ensemble name in the editable text box. Then you can click on the Create ensemble button to create the new ensemble, or reset the configuration by clicking on the Reset button.
2.4.13 API Request Preview

The **API Request Preview** button is in the middle on the bottom of the configuration panel, next to the **Reset** button (See (Figure 2.25)). This is to show how to create the ensemble programmatically: the endpoint of the REST API call and the JSON that specifies the arguments configured in the panel. Please see (Figure 2.26) below:
There are options on the upper right to either export the JSON or copy it to clipboard. On the bottom there is a link to the API documentation for ensembles, in case you need to check any of the possible values or want to extend your knowledge in the use of the API to automate your workflows.

Please note: when a default value for an argument is used in the chosen configuration, the argument won’t appear in the generated JSON. Because during API calls, default values are used when arguments are missing, there is no need to send them in the creation request.

## 2.5 Visualizing Ensembles

Being able to effectively visualize an ensemble is paramount to exploring it, interpreting it, and explaining why it produces certain outcomes. BigML provides two different visualizations, a Partial Dependence Plot (PDP) and a list of the single models:

- **Partial Dependence Plot**: a graphic representation of the marginal effect that the combination of two fields (predictors) have on the objective field (ensemble predictions) keeping the rest of the field values constant.

- **Model list**: provides a list of the single models that form the ensemble. This visualization is only available for **Decision Forest** ensembles (see Subsection 2.4.3).

Note: BigML does not provide the model list for Boosted Trees because they cannot be interpreted the same way as the models for other ensemble types (see Subsection 2.5.2).

In the top menu you will find a summary of the ensemble results: the number of models in the ensemble, the sample rate, the objective field used, if the ensemble has been randomized, the type of ensemble used (Decision Forest or Boosted Trees), and the number of instances in the dataset.

![Figure 2.27: Ensemble top menu](image)
For Decision Forests, below the top menu, you will find the icons corresponding to each one of the views (the PDP and the model list) to switch from one view to another. (See Figure 2.28).

![Figure 2.28: Switch from the chart view to the model list view](image)

You can find a detailed explanation of each view in the following subsections.

### 2.5.1 Partial Dependence Plot (PDP)

The PDP is the main default view you will find when creating an ensemble. The main goal is to represent the marginal effect of a set of variables (input fields) on the ensemble predictions disregarding the rest of the variables. It is a common method for visualizing and interpreting the impact of the variables on ensemble predictions, and it can be used for classification and regression ensembles.

Note: the ensemble PDP is not a representation of the dataset values; it is a representation of the ensemble results and their dependence from a set of variables used as inputs.

In order to ensure responsiveness, the PDP is built using 10 models by default. For ensembles with a higher number of models, a random sample of 10 models will be selected to calculate the predictions. A warning message will appear at the top of the ensemble view to indicate that the chart has been built with a lower number of models because this may cause slight differences between the chart predictions and the ensemble actual predictions. Although in most cases these differences should be imperceptible, you can use the slider to increase the number of trees (up to 100 trees) and the re-sampling option to take another random sample of trees. Click on the corresponding options as shown in Figure 2.29.
You can visualize classification and regression ensembles in the heatmap chart. In the case of classification ensembles, the different classes of the objective field are represented by different colors. The different color shadings for each class represent the different votes in the case of Decision Forests, i.e., the percentage of trees voting for a given class in the ensemble (see Subsection 2.6.3.2) and the class probabilities in the case of Boosted Trees (see Subsection 2.2.1). For regression ensembles, the different prediction values are represented by differences in the color scale.

The chart view is always composed of three main parts: the CHART itself, the PREDICTION legend and
the INPUT FIELDS form. (See Figure 2.31.)

Figure 2.31: Ensemble chart

- The CHART allows you to view the impact of the two input fields on the objective classes predictions. You can select any categorical or numeric field for each axis. You can also switch the axis by clicking on the option on top of the chart area. (See Figure 2.32.) In the grey area next to the axis selectors you can see the axis values. You can freeze the view by pressing \textit{Shift} and release it again by pressing \textit{Escape} from your keyboard. When the view is frozen, an edition icon will appear and you can edit the axis values to obtain a prediction for that value. (See Figure 2.32.)
• The PREDICTION legend allows you to visualize the objective field classes (classification ensembles) or the predicted value (regression ensembles). In the case of classification ensembles you will also obtain the votes, i.e., the percentage of trees voting for a given class in the ensemble (see Subsection 2.6.3.2), for Decision Forests or the class probabilities for Boosted Trees (see Subsection 2.2.1). By default, color tones and shadings are set according to the range of values shown in the chart area. This is the default because for some configurations of the chart the predictions may vary a small amount relative to the global range. For example, imagine the chart is showing temperature predictions based on location, time-of-year, and time-of-day. San Diego’s daily range (13°C to 18°C) could be tiny compared to the Earth’s global range (-62°C to 48°C). You can change this behavior and see the color scales and shading according to the total range of possible predicted values by clicking on the icon 'Total'. (See Figure 2.33.) For classification ensembles, this option allows you to see the color shading for the total range of potential values (from 0% to 100%). For regression ensembles, the 'Total' colors option allows you to see the color scale for the total range of predictions. For classification ensembles you can also select to see only one of the classes using the class selector at the bottom of the legend. (See Figure 2.33.)
• Below the chart legend, you can find the INPUT FIELDS form. (See Figure 2.34.) You can configure the values for any numeric or categorical field. Text and items fields are not yet supported. By changing their values, you can see the predictions changing in real-time. You can sort the fields by their importance, select or disable them. If you disable an input field, it will be ignored to calculate the final prediction. The strategy used to calculate predictions when some fields are disabled is the proportional missing strategy (see Subsection 2.6.3.1).

Note: it is important to notice that disabled fields will be ignored when calculating the chart predictions. This is because the original intent of the PDP is to understand the impact of the axis fields by ignoring the influence of all the other fields. So if you trained the ensemble with missing values (see Subsection 2.4.6.2) and they have some impact on predictions, you will not see it in the chart predictions. In this case it will be a mismatch between the chart predictions and your final predictions.
2.5.1.1 Export chart as an image

Download the ensemble chart as an image in PNG format with or without legends. To download it with legends, press `Shift` from your keyboard to freeze the chart view. (See Figure 2.35.)

2.5.1.2 Interpreting Partial Dependence Plots

You can easily see field impact on predictions using the ensembles chart. See below three different situations using an ensemble which aims to predict if a person has diabetes based on several input fields:
• **Both fields impact predictions:** in the image below, the combination of the selected fields, “BMI” (Body Mass Index) and “Glucose”, have a high impact on predicting diabetes since variations in both fields cause variations in predictions.

![Diabetes diagnosis dataset ensemble](image)

Figure 2.36: Both selected fields impact predictions

• **Only one of the fields impacts predictions:** looking at the image below we can conclude that “Skinfold” is not a good predictor for diabetes since variations in this field don’t affect predictions. However, the level of “Glucose” has great impact on predictions. (See Figure 2.37.)
Both fields have low or no impact on predictions: if you select variables with little or no influence on predictions, you can see that variations in the selected fields don’t lead to differences in predictions. In this case, any combination of “Blood pressure” and “Insulin” always returns the same value for diabetes, “False”. 

Figure 2.37: One of the selected fields impact predictions
2.5.2 Model List

The model list is only available for Decision Forests. The model list is not provided for Boosted Trees because they cannot be analyzed the same way as the models for other ensemble types. Instead of predicting the objective field, each boosted tree tries to fit a gradient to correct the mistakes made by the previous single tree. Therefore the single models are hardly interpretable individually.

The same BigML interactive visualization for models is used to show each single tree composing the ensembles. The ensemble model list provides a general overview on the ensemble and a means to get down to the single model level. The list of models comprise the following information for each of them:

- **Model preview**: this is a snapshot of the tree representation for each model.
- **Model link**: you can access each underlying tree model by clicking on this link.
- **Data distribution histogram**: the distribution of the target field values in the training set that was used to build the model.
- **Predicted distribution histogram**: predictions distribution for the model. The predicted distribution should roughly match the data distribution.
Chapter 2. Ensembles

If you click on the model preview or link, you will be taken to that model view (see Figure 2.40), where you can use all of the model visualization features described in Section 1.5, such as BigML proprietary tree and sunburst dynamic visualization.

2.6 BigML Ensemble Predictions

2.6.1 Introduction

The ultimate goal in building a BigML ensemble is being able to make predictions for previously unseen instances with an unknown label. In BigML, you can make predictions for single instances or for many instances in a batch. Each prediction comes with a measure indicating its reliability, expressed either as a probability (for classification ensembles), as a confidence or votes (only for classification Decision Forests), or as an expected error (only for regression Decision Forests).
The predictions tab in the main menu of your BigML Dashboard is where all your saved predictions are listed. (See Figure 2.41.)

![Figure 2.41: Predictions list empty view](image)

Ensemble predictions are saved under the CLASSIFICATION & REGRESSION option in the menu. (See Figure 2.42.)

![Figure 2.42: Menu options of the predictions list view](image)

From this view you can select to view the list for your **single instances predictions** or your **batch predictions** by clicking in the corresponding icons. (See Figure 2.43 and Figure 2.44.)

![Figure 2.43: Single predictions icon](image)

![Figure 2.44: Batch predictions icon](image)

In the **predictions list view**, you can see, for each prediction, the **Model**, **Ensemble** or **Logistic Regression** icon used for the prediction, the **Name** of the prediction, the **Objective** (objective field name), the **Prediction** (the prediction result), and the **Age** (time since the prediction was created). (See Figure 2.45.)

![Figure 2.45: Predictions list view](image)
2.6.2 Creating Ensemble Predictions

As shown in Figure 2.46, BigML provides three options to make predictions from your ensembles:

1. **PREDICT**: to predict a single instance using the prediction form.
2. **BATCH PREDICTION**: to predict multiple instances simultaneously.

---

2.6.2.1 Predict

BigML allows you to quickly make predictions for single instances by providing a form containing the fields used by the ensemble, so you can easily set the input data and get an immediate response. This option is only available from the BigML Dashboard for ensembles with less than 100 fields. If you want to perform single instance predictions for ensembles with a higher number of fields, you can use the BigML API\[^{12}\].

Follow the steps detailed below to create a single prediction:

\[^{12}\]https://bigml.com/api/predictions
1. Choose the PREDICT option under the ensemble 1-click menu. (See Figure 2.47.)

![Figure 2.47: Predict option from ensemble 1-click menu](image)

Alternatively, you can choose the PREDICT option in the pop up menu in the list view as shown in Figure 2.48.

![Figure 2.48: Predict option from ensemble pop-up menu](image)

2. You will be redirected to the prediction form where you will find all the fields used by the ensemble as predictors ordered by field importance. The importance percentage is found next to the field name as shown in Figure 2.49. You may not find all the fields from your original dataset because the ensemble may find them irrelevant or redundant in terms of their predictive impact.
3. **Select the fields** you want to be taken into account for your prediction as shown in Figure 2.50. Non-selected fields will be considered as missing values during the prediction. If your ensemble was trained with Missing splits (see Subsection 2.4.6.2), then missing values are considered by the ensemble as any other valid value. If your ensemble was built without missing values then any of the Missing strategies may apply during your prediction (see Subsection 2.6.3.1.)
4. **Set input values** for your selected fields. Depending on the field type, you will need to input the values differently:
   - Numeric fields: move the slider or input a specific value in the text box.
   - Categorical fields: select one class from the selector.
   - Text fields: write one or several terms in the free text box.
   - Date-time fields: select the appropriate values from the selector.
   - Items fields: when you write the first three characters of an item name, several items matching those characters will appear, so you can select the right one. You can input more than one item for a field.

5. **Get the prediction** on the top of the form. For classification ensembles you will get all classes distribution and for regression ensembles you will get the predicted value for the objective field. Both types of ensembles will also show a certainty measure along with the prediction:
   - For classification Decision Forests, you can get the **probability**, the **confidence** or the **votes** depending on the option you choose. For regression Decision Forests you get the **expected error**.
   - For classification Boosted Trees, you get the **probability**. For regression Boosted Trees the expected error cannot be calculated. Read more about ensemble predictions in **Subsection 2.2.1.2.**
BigML predictions are synchronous, i.e., when you send the input data you get an immediate response. Read more about local predictions in Subsection 2.6.2.1.1.

6. Optionally Save the prediction so you can access them afterwards from the ensemble predictions list view.
2.6.2.1 Local Predictions

BigML provides local predictions from the BigML Dashboard for single instance predictions. Local predictions allow you to get a real-time prediction without consuming any credits or requiring an internet connection. This is possible because your ensemble is saved in the browser’s memory so when the input values change, BigML immediately evaluates all models, obtaining their predictions and then combining them in a matter of microseconds.

Local predictions are only available for ensembles built with 15 models or less for Decision Forests and 30 models or less in the case of Boosted trees. For ensembles with higher number of models, you can still perform remote single predictions.

2.6.2.1.2 Predictions with Images

BigML ensembles can be trained from images using extracted image features (Subsection 2.2.3). Because image features are automatically generated numeric fields, creating ensemble predictions with images is the same as creating other ensembles. The only thing different is input fields of images.

Note: When the input fields contain images, in order to create the single prediction, BigML will extract image features automatically to match what were used in the dataset to train the ensemble.
The ensemble in Figure 2.53, “grape-strawberry texture”, was created from a dataset containing image features Wavelet subbands. Creating a prediction using the ensemble will be directed to the prediction form which presents all input fields used by the ensemble. One of them is the image field. Because this is a single prediction, an image is input by using a single image source. Clicking on the input field box, single image sources available will be in the dropdown list. There is also a search box which can be used to locate specific ones.

Oftentimes single image sources were used for creating a composite source, they become component sources of the composite source. Or an image was uploaded as a part of an archive file (zip/tar) which created a composite source. In those cases, the composite source will be shown in the dropdown list, along with an icon “List components”. In the example in Figure 2.54, predict-images.zip is a composite source, click on the icon to show its component sources.
After the component sources of the composite are listed, scroll the dropdown list to find the desired one, then click to select it, as shown in Figure 2.55. There is also a search box to locate specific component sources.

In addition to images, ensembles may use other fields, which will be in the prediction form too. As shown in Figure 2.56, all the fields can be selected, and their input values be set by dragging the knobs on the sliders or by entering precise values in their input boxes.

Once all fields are selected, click on the green button \textbf{Predict} to create a prediction.
Chapter 2. Ensembles

After a new prediction is created, as shown in Figure 2.57, the predicted class is at the top of the form along with its probability. The prediction interface is the same as ones created by non-image ensembles. Everything described earlier in this section (Subsection 2.6.2.1) applies.

2.6.2.2 Batch Predictions

BigML batch predictions allow you to make simultaneous predictions for multiple instances. All you need is the ensemble you want to use to make predictions and a dataset containing the instances which you want to use as prediction inputs. BigML will create a prediction for each instance in the dataset. Follow the steps detailed below to create a batch prediction:

1. Select the BATCH PREDICTION option under the ensemble 1-click menu (see Figure 2.58) or the CREATE BATCH PREDICTION option in the pop up menu of the list view (see Figure 2.59).
2. **Select the dataset** containing all the instances you want to create a prediction for. The instances should contain the input values for the fields used by the ensemble as predictors. You can also select a subset of the ensemble fields to be taken into account by configuring your prediction (see Subsection 2.6.3.5.) BigML batch predictions can handle missing data in your prediction dataset (see Subsection 2.6.3.1.)

3. **Optionally, select the ensemble** you want to use for the prediction. BigML pre-selects the ensemble you created the batch prediction from at step 1, but you can change it at any time in the batch prediction view by selecting another ensemble from the ensemble selector displayed in the right pane. You can even switch to a model or logistic regression by selecting the corresponding icon in the top left menu.
4. After you have selected the ensemble and the dataset, the batch prediction **configuration options** (see Subsection 2.6.3) will appear along with a **preview of the prediction output**, which is formatted as a comma-separated list of values (CSV format). (See Figure 2.61.) The default output includes all the fields in your prediction’s dataset plus a last column containing the calculated predictions.

**Note:** BigML does not include the predictions’ probability, confidence or expected error by default so you will have to configure your output file to include that information as explained in Subsection 2.6.3.6.
5. By default, BigML generates an output dataset containing the batch prediction results. You find it in the BigML Dashboard’s dataset list view and can use it as any other dataset to analyze the batch prediction output afterwards. If you do not want a dataset with all the prediction results to be created, you can deselect the button highlighted in Figure 2.62.

![Figure 2.62: Create dataset from batch predictions](image)

6. Once you are done configuring your batch prediction, click the **predict** green button to generate it. This process may take some time depending on the size of the input dataset. (See Figure 2.63.)
Chapter 2. Ensembles

7. After the batch prediction has been created, you will be able to download a file with all the instances found in your input dataset along with the prediction corresponding to each one of them. (See Figure 2.64.)

8. If you did not disable the option to create a dataset, as explained above (see step 4), an Output dataset button will also be available to allow you to directly jump to the output dataset. (See Figure 2.65.)
2.6.2.2.1 Batch Prediction with Images

BigML ensembles can be trained from images using extracted image features (Subsection 2.2.3). The input of a batch prediction is a dataset. So when creating a batch prediction with images, the dataset has to have the same image features used to train the ensemble. The image features are in the dataset used to create the ensemble.

As shown in Figure 2.66, the input for the ensemble batch prediction is selected as predict-images\_texture, which is a dataset consisting of six images and contains a set of extracted image features, Wavelet subbands.

Image features are configured at the source level. For more information about the image features and how to configure them, please refer to section Image Analysis of the Sources with the BigML Dashboard\[^{15}\].

\[^{15}\]https://static.bigml.com/pdf/BigML_Sources.pdf
For the rest of batch predictions with images, including batch prediction configuration options and output datasets, everything stated earlier in current section (Subsection 2.6.2.2) applies.

2.6.3 Configuring Ensemble Predictions

BigML provides several options to change its default behavior when calculating predictions. For single predictions as well as for batch predictions you can configure the strategy used for handling missing values (see Subsection 2.6.3.1.) In the case of Decision Forests you can also configure the prediction operating kind, i.e. the criteria used to combine the single tree predictions, confidence, probability or votes (see Subsection 2.6.3.2.) For classification ensembles you can also use a threshold for your single an batch predictions. For batch predictions you can configure the automatic fields mapping performed by BigML (Subsection 2.6.3.5), and define the output file settings (Subsection 2.6.3.6.)

2.6.3.1 Missing Strategies

When you create a new prediction, BigML will automatically navigate through the corresponding ensemble to find the leaf node that best classifies the new instance.

However, it may just so happen that your new data (the instances you want to predict) does not have populated values for all the fields used in building the original ensemble. For example, imagine that you are trying to predict diabetes and you have the patient’s glucose level and BMI (Body Mass Index) but not his blood pressure. If the ensemble arrives at a node where the blood pressure level is required, BigML can handle this missing value by using one of these two strategies:

- **Last prediction**: it returns the prediction value and confidence of the parent node.
- **Proportional**: it combines all subtrees’ predictions beneath the current node based on the data distribution of their child nodes in order to compute the prediction value and confidence.

For single predictions you can select either Missing strategies by clicking in the icons shown in Figure 2.67.

![Figure 2.67: Missing strategies for single predictions](image-url)
For batch predictions you can find both options under the configuration panel as shown in Figure 2.68.

Figure 2.68: Missing strategies for batch predictions

### 2.6.3.2 Combine single tree predictions: probability, confidence or votes

Ensembles are composed of several trees. Each tree returns a different prediction given an input data. These single predictions need to be combined to get a final prediction for the ensemble. For **Boosted Trees** there is only one way to combine the single tree predictions explained in Subsection 2.2.1.2, so you will not see any options in the prediction view. For **Decision Forests** there are three different options (called operating kinds in BigML API) to combine the single tree predictions that compose the ensemble and get a final prediction:

- **Probabilities**:
  - For **classification** ensembles per-class probabilities are averaged taking into account all trees composing the ensemble. The class with the highest probability is the winner class. For example find below a classification ensemble built using three decision trees trying to predict two classes, “True” and “False”:

<table>
<thead>
<tr>
<th>Trees</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree 1</td>
<td>80%</td>
<td>20%</td>
</tr>
<tr>
<td>Tree 2</td>
<td>40%</td>
<td>60%</td>
</tr>
<tr>
<td>Tree 3</td>
<td>60%</td>
<td>40%</td>
</tr>
<tr>
<td><strong>Ensemble</strong></td>
<td><strong>60%</strong></td>
<td><strong>40%</strong></td>
</tr>
</tbody>
</table>

Table 2.2: Example of classification ensemble using probability

---

[14](https://bigml.com/api)
The predicted class is “True” because it has a higher probability \((\frac{80\% + 40\% + 60\%}{3} = 60\%)\) than the “False” class \((\frac{20\% + 60\% + 40\%}{3} = 40\%)\).

- For **regression** ensembles the probability option averages the predictions of the trees composing the ensemble. For example, considering again three different trees in an ensemble but predicting a numeric output this time:

<table>
<thead>
<tr>
<th>Trees</th>
<th>Total Sales</th>
<th>Expected Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree 1</td>
<td>$200</td>
<td>$2.40</td>
</tr>
<tr>
<td>Tree 2</td>
<td>$250</td>
<td>$2.10</td>
</tr>
<tr>
<td>Tree 3</td>
<td>$180</td>
<td>$1.45</td>
</tr>
<tr>
<td><strong>Ensemble</strong></td>
<td><strong>$210</strong></td>
<td><strong>$1.98</strong></td>
</tr>
</tbody>
</table>

Table 2.3: Example of regression ensemble using probability

The final result will be a total sale of $210 \((\frac{$200 + $250 + $180}{3} = $210)\) with an error of $1.98 \((\frac{$2.4 + $2.1 + $1.45}{3} = $1.98)\).

- **Confidences:**
  - For **classification** ensembles per-class confidences are averaged taking into account all trees composing the ensemble. The class with the highest confidence is the winner class. It is calculated in the same way as the prediction in Table 2.2 but using the per-class confidences instead of the probabilities.
  - For **regression** ensembles the confidence option averages the predictions of the trees composing the ensemble in the same way that explained for probabilities in Table 2.3 but weighted by the expected error.

- **Votes:**
  - For **classification** ensembles each tree prediction is considered as one vote. The “votes” of a given class is the percentage of trees in the ensemble that vote for that class. You can find below an example that shows a classification ensemble built using three decision trees to predict two classes, “True” and “False”:

<table>
<thead>
<tr>
<th>Trees</th>
<th>Predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree 1</td>
<td>True</td>
</tr>
<tr>
<td>Tree 2</td>
<td>False</td>
</tr>
<tr>
<td>Tree 3</td>
<td>True</td>
</tr>
<tr>
<td><strong>Ensemble</strong></td>
<td><strong>True</strong></td>
</tr>
</tbody>
</table>

Table 2.4: Example of classification ensemble using votes

Since there are more trees that predicted the class “True” (two versus only one tree that predicted the class “False”), the final prediction is “True” with the 66.67% of votes \((\frac{2}{3} = 0.6667)\).

**Note:** if two or more classes have the same number of votes, the first class in alphabetical order will take precedence.

- For **regression** ensembles the votes option averages the predictions of the trees composing the ensemble. It gives the same results as the probability (see Table 2.3).
You can choose any option (probabilities, confidences or votes) from the BigML Dashboard to calculate the Decision Forests **single predictions** (see Figure 2.69) or **batch predictions** (see Figure 2.70).

Figure 2.69: Probabilities, confidences or votes for Decision Forest single predictions
2.6.3.3 Probability, confidence, and votes thresholds

The thresholds are only available for classification ensembles, and it usually makes sense for unbalanced binary classifications, when you want to minimize false positives at the cost of false negatives. The positive class will be predicted if the probability, confidence or the votes are greater than the given threshold, otherwise the following class with greater probability, confidence or votes will be predicted instead.

To configure a threshold for your single predictions follow these steps:

1. Select the probability, the confidence, or votes measure depending on the criteria you want to use for Decision Forests (see Figure 2.71). To learn more about these three ways to combine single tree predictions in the ensemble refer to Subsection 2.6.3.2. Boosted trees will only have the option to set a probability threshold.
2. Select the **positive class**, i.e. the class for which you want to apply the threshold:
3. Set a value for the **threshold** using the slider. The positive class will only be predicted when the probability, confidence or votes of the prediction is above the established threshold, otherwise the following class with higher probability, confidence or votes will be predicted instead.

![Figure 2.73: Set a threshold](image)

For batch predictions, you will find the same options under the *CONFIGURE* panel. (See *Figure 2.74.*)
2.6.3.4 Default Numeric Value

If the dataset used to make the batch prediction contains instances with missing values for the numeric fields you can easily replace them by the field's Mean, Median, Maximum, Minimum or by Zero using the Default numeric value before creating your batch prediction, (See Figure 2.75.)
2.6.3.5 Field Mapping

By default, BigML maps fields based on their names. If there is a mismatch between the field names in your ensemble and those in the input dataset you selected for the batch prediction, you can specify the right correspondence between the two sets of fields by explicitly assigning to each field appearing in the “Ensemble fields” column its associated input field in the “Dataset fields” column. (See Figure 2.76.)

If the dataset’s and ensemble’s field names do not match but their IDs do, which happens when corresponding fields appear in the same order, you can tell BigML to use the field ID instead of the field name to map the fields. To do this, click the green switcher shown in Figure 2.76.

If you do not want some of the fields to be considered during the evaluation, you can also manually search for those fields and remove them from the “Dataset fields” column.

The fields mapping from the BigML Dashboard has a limit of 200 fields. For batch predictions with a higher number of fields, use the argument field_map from BigML API if you need to map your fields.

2.6.3.6 Output Settings

As mentioned, batch predictions can create a file containing all input instances along with the predictions BigML calculated for each of them. Define the following settings to customize your prediction file:

- **Separator**: this option allows you to choose a separator for your output file values. The default separator is the comma. You can also select the semicolon, the tab, or the space.
- **New line**: this option allows you to set the new line character to use as the line break in the generated csv file: “LF”, “CRLF”.

[15]https://bigml.com/api/batchpredictions#bp_batch_prediction_arguments
• **Output fields**: this option allows you to include or exclude any of your dataset fields from the output file from the preview shown in Figure 2.77.

  **Note**: a maximum of 100 fields are displayed in the preview, but all your dataset fields are included in the output file by default unless you exclude them.

• **Headers**: this option includes or excludes a first row in the output file (and in the output dataset) with the names of each column (input field names, prediction column name, probability and/or confidence column name, field importances column names, single tree predictions column names, etc.). By default, BigML includes the headers.

• **Prediction column name**: this option allows you to customize the name for your predictions column. By default BigML uses the name of the ensemble’s objective field.

• **Probability, confidence (or expected error) or votes**: this option allows you to include an additional column in the output file with the probability, the confidence (or expected error in case of regression trees) or the votes per instance. In the case of classification Boosted Trees you will always see the probability option and in the case of classification Decision Forests it will depend on the option selected to combine the single tree predictions (see Figure 2.70). These measures are not included by default in your batch predictions.

  **Note**: remember that the expected error cannot be calculated for Boosted Trees, therefore this option will be disabled for regression Boosted trees.

• **Probability, confidence (or expected error) or votes column name**: this option allows you to customize the name for the probability, confidence (or expected error) or votes column in case you include it in the output file. By default BigML uses “probability”, “confidence” (for both confidence and expected error) or “votes”.

• **Single tree predictions**: this option allows you to include a column for each of the individual model predictions only in the case of Decision Forests. That will add a column per model, named <prediction_name>_<n> where n is the position of the model in the model list in the ensemble, starting at 1.

• **All class confidences**: this option allows you to include the probabilities for each class in the objective field for classification Decision Forests. There is a column per class, named "<class_name> confidence".

• **All class votes**: this option allows you to include the votes for each class in the objective field for classification Decision Forests and Boosted trees. There is a column per class, named "<class_name> votes".

• **All class probabilities**: this option allows you to include the probabilities for each class in the objective field for classification Decision Forests and Boosted trees. There is a column per class, named "<class_name> probability".

• **Importances**: this option allows you to include a column for each of the field relative importances for the ensemble predictions (for Boosted trees and Decision Forests). There is a column per field, named "<field_name> importance".
2.6.4 Visualizing Ensemble Predictions

Visualization of ensemble predictions changes depending on whether you are predicting one single instance or you are predicting multiple instances using the batch predictions option. (See Subsection 2.6.4.1.)

2.6.4.1 Single Predictions

There are some essential differences between Decision Forests and Boosted Trees predictions, hence the visualizations for both of them differ. These differences are highlighted in the following paragraphs.

For single predictions you can find the prediction for your objective field at the top of the form along with the performance measure.

For classification ensembles, you will get the predicted class at the top and all the objective field class probabilities in the histogram below as shown in Figure 2.78.
For regression Decision forest ensembles you will get a numeric prediction and the expected error for that prediction as shown in Figure 2.79.
For **regression Boosted trees ensembles** you will get a numeric prediction (see Figure 2.80) but the expected error for that prediction cannot be calculated as in the case of Decision Forests (see Subsection 2.2.1).

In any of the above-mentioned cases, you can change any time the value of the displayed input fields to have your prediction recalculated in real-time.

If you have saved your prediction, you can go back to it and visualize it.

Read a detailed explanation of confidences, probabilities and expected error calculations in Subsection 2.2.1.2.

**2.6.4.1.1 Prediction explanation**

Prediction explanation helps understand why an ensemble makes a certain prediction. This is very useful in many applications, and the reasons behind an ensemble's prediction are often as important as the prediction itself.

BigML prediction explanation is based on Shapley values. For more information, please refer to this research paper: *A Unified Approach to Interpreting Model Predictions* [3].

For any classification or regression ensemble, you can request the explanation for the prediction by clicking the prediction explanation icon and then click Predict (see Figure 2.81).
The prediction explanation represents the most important factors considered by the ensemble in a prediction given the input values. Each input value will yield an associated importance, as you can see Figure 2.82. The importances across all input fields should sum 100%.

For some input fields you will see a “+” icon next to the importance. This is because the importance may not be directly associated with the input value, i.e., it can be explained by other reasons. In the Figure 2.83 below, the importance of 13.62% for the field “Fare today” is not explained by this field being equal to 24,719. Rather, it is because this field value is not missing (which accounts for an importance of 8.80%) and because it is higher than 17,000 (4.82% of importance).
The prediction explanation for ensembles is calculated using the results of over a thousand distinct predictions using random perturbations of the input data. For this reason, the calculation of the explanation may take some time to be computed.

**Note:** the input field importances in the prediction explanation are different from the overall field importances of the ensemble. A field can be very important for the ensemble but insignificant for a given prediction.

### 2.6.4.2 Batch Predictions

For batch predictions, you always get a **file** and an optional **output dataset**.

#### 2.6.4.2.1 Output File

From the batch prediction view, you can access the output file containing your **predictions** for each of your dataset instances in the last column. (See Figure 2.84.) You can configure several options to **customize your output file** including the separator for the columns, the name of your prediction column, the dataset fields you want to include, whether you want to include the a first row with the headers for your column names. You can find a detailed explanation of those options in **Subsection 2.6.3.6**.
See an output file example in Figure 2.85 where the two last columns contain the prediction and the confidence for each instance.

```
Pregnancies,Glucose,Blood pressure,Skinfold,Insulin,BMI,Diabetes,Confidence
8,183,64,0,0,23.3,True,0.6574
5,116,74,0,0,25.6,False,0.845
10,115,0,0,0,35.3,True,0.6469
8,125,64,0,0,0.0,False,0.9356
1,189,60,23,846,30.1,True,0.7574
1,103,30,38,83,43.3,True,0.7662
7,103,66,32,0,39.1,True,0.7662
1,101,50,15,36,24.2,False,0.948
0,100,88,60,110,46.8,False,0.5413
```

Figure 2.85: An example of a batch prediction output file

### 2.6.4.2.2 Output Dataset

By default, BigML creates a dataset out of your batch prediction. (See Subsection 2.6.3.6.) You can access your output dataset from the batch prediction view by clicking the **Output dataset** button shown in Figure 2.87.
In the output dataset you can find an additional field (named by default as per your ensemble's objective field) containing the predictions for each one of your instances. If you configured your batch prediction to include the confidence or expected error you will be able to find it in the last field of your output dataset as shown in Figure 2.87.

**2.6.4.2.3 Batch Prediction 1-Click Actions**

From the batch prediction view you can perform the following actions (see Figure 2.88):

- **BATCH PREDICTION AGAIN**: this option will redirect you to the batch prediction creation view, with the same ensemble and prediction dataset already selected. This option allows you to rapidly recreate the batch prediction using a different configuration.
• **BATCH PREDICTION WITH ANOTHER DATASET**: this option allows you to easily create a batch prediction using the same ensemble and a different dataset.

• **BATCH PREDICTION USING ANOTHER ENSEMBLE**: this option allows you easily create a batch prediction using the same dataset and a different ensemble.

• **NEW BATCH PREDICTION**: this option redirects you to the batch prediction creation view where you can select a prediction dataset and an ensemble to create your prediction.

2.6.5 Consuming Ensemble Predictions

BigML provides plenty of means for developers to integrate BigML ensemble predictions within their apps. In the following sections, we will describe how you can use BigML REST API and BigML Python bindings to work with ensemble predictions.

2.6.5.1 Using Ensemble Predictions via the BigML API

Ensemble predictions have full citizenship in the BigML API. This means you can programmatically create, update, list, delete, and use them for predictions. For example, this is how you can create a single prediction using the command line from a given ensemble and defining the input data. This will require properly setting the `BIGML_AUTH` environment variable to contain your authentication credentials:

```
curl "https://bigml.io/prediction?$BIGML_AUTH" 
-X POST 
-H 'Content-Type: application/json' 
-d '{"ensemble": "ensemble/50650bdf3c19201b64000020", "input_data": {"000001": 3, "000002": 4.5, "000003": 3}}'
```

For more information on using ensemble predictions through the BigML API, please refer to [prediction REST API documentation](https://bigml.com/api/predictions).

2.6.5.2 Using Ensemble Predictions via the BigML Bindings

BigML bindings provide a convenient way to access the BigML REST API from your language of choice. They offer a higher-level view of BigML Machine Learning resources and algorithms in a number of languages, including Python, Node.js, Java, Swift, and Objective-C. For example, this is how you can create an ensemble prediction in Python using BigML bindings:

```python
# Create a prediction using the BigML Python bindings
from bigml.api import Auth, connect
auth = Auth(BIGML_USERNAME, BIGML_API_KEY)
connection = connect(auth, timeout=30)
ensemble = connection.get_ensemble(ensemble_id)
input_data = {'000001': 3, '000002': 4.5, '000003': 3}
prediction_id = connection.create_prediction(ensemble['resource'], input_data)
```

Copyright © 2023, BigML, Inc.
from bigml.api import BigML
api = BigML()
prediction = api.create_prediction("ensemble/573d997068a27e0f620038df",
   {"sepal length": 5,
    "sepal width": 2.5},
   {"name": "my prediction"})

BigML bindings also provide the means to carry through predictions locally, without ever hitting the network, which can greatly improve the latency of predicting from your apps. This is made possible by BigML ensembles being white-box, meaning you can download them and use them independently from BigML. For example, the following code snippet shows how you can download an ensemble and use it for making a local prediction using BigML bindings for Python:

```python
from bigml.ensemble import Ensemble
from bigml.api import BigML
api = BigML()
ensemble = api.get_ensemble("ensemble/502fdbff15526876610002615",
   query_string="only_ensemble=true;limit=-1")
local_ensemble = Ensemble(ensemble)
prediction = local_ensemble.predict({"petal length": 3, "petal width": 1})
```

For more information on using ensembles through the BigML API, please refer to BigML bindings documentation.

### 2.6.6 Descriptive Information

Descriptive information is what allows you to describe a prediction so you can find it later and easily recognize it among other predictions.

Each prediction is associated with **name**, **description**, **category**, and **tags**. See the following sections for a brief description of each concept. In Figure 2.89, you can see the possibilities that the MORE INFO menu option gives to edit them.

![Figure 2.89: Edit predictions](Copyright © 2023, BigML, Inc.)
2.6.6.1 Name

If you do not specify a name for your predictions, BigML assigns a default name depending on the type of predictions:

- **Single predictions**: the name always follows the structure “Prediction for <objective field name>”
- **Batch predictions**: BigML combines your prediction dataset name and the ensemble name: “Batch prediction of <ensemble name> with <dataset name>”.

Predictions names are displayed on the list view and also on the top bar of a prediction view. Predictions names are indexed to be used in searches. You can rename your predictions at any time from the More info panel. The name of a prediction cannot be longer than 256 characters. More than one prediction can have the same name even within the same project, since they are automatically assigned unique internal identifiers.

2.6.6.2 Description

Each ensemble prediction also has a description that is very useful for documenting your Machine Learning projects. Predictions take the description from the ensembles used to create them.

Descriptions can be written using plain text and also markdown\(^\text{17}\). BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 2.90.)

![Markdown editor for evaluations descriptions](image)

Figure 2.90: Markdown editor for evaluations descriptions

Descriptions cannot be longer than 8192 characters and can use almost any character.

2.6.6.3 Category

Each prediction is associated with a category taken from ensemble used to create it. Categories are useful to classify predictions according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

A prediction category must be one of the categories listed on table Table 2.5.

\(^\text{17}\)https://en.wikipedia.org/wiki/Markdown
### Table 2.5: Categories used to classify predictions by BigML

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace and Defense</td>
</tr>
<tr>
<td>Automotive, Engineering and Manufacturing</td>
</tr>
<tr>
<td>Banking and Finance</td>
</tr>
<tr>
<td>Chemical and Pharmaceutical</td>
</tr>
<tr>
<td>Consumer and Retail</td>
</tr>
<tr>
<td>Demographics and Surveys</td>
</tr>
<tr>
<td>Energy, Oil and Gas</td>
</tr>
<tr>
<td>Fraud and Crime</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Higher Education and Scientific Research</td>
</tr>
<tr>
<td>Human Resources and Psychology</td>
</tr>
<tr>
<td>Insurance</td>
</tr>
<tr>
<td>Law and Order</td>
</tr>
<tr>
<td>Media, Marketing and Advertising</td>
</tr>
<tr>
<td>Miscellaneous</td>
</tr>
<tr>
<td>Physical, Earth and Life Sciences</td>
</tr>
<tr>
<td>Professional Services</td>
</tr>
<tr>
<td>Public Sector and Nonprofit</td>
</tr>
<tr>
<td>Sports and Games</td>
</tr>
<tr>
<td>Technology and Communications</td>
</tr>
<tr>
<td>Transportation and Logistics</td>
</tr>
<tr>
<td>Travel and Leisure</td>
</tr>
<tr>
<td>Uncategorized</td>
</tr>
<tr>
<td>Utilities</td>
</tr>
</tbody>
</table>

#### 2.6.6.4 Tags

A prediction can also have a number of **tags** associated with it that can help to retrieve it via the BigML API or to provide predictions with some extra information. Your prediction inherits the tags from the ensemble used to create it. Each tag is limited to a maximum of 128 characters. Each prediction can have up to 32 different tags.

#### 2.6.7 Ensemble Predictions Privacy

The link displayed in the **More Info** panel is the private URL of your prediction, so only a user logged into your account is able to see it. Neither single predictions nor batch predictions can be shared from your BigML Dashboard by sharing a link, as you can do with other resources.

![Private link of a prediction](https://mpec.dev.bigmli.com/dashboard/prediction/573d593b95b394440000086)

Figure 2.91: Private link of a prediction
2.6.8 Moving Ensemble Predictions to Another Project

When you create a prediction it will be assigned to the same project where the original ensemble is located. You cannot move predictions between projects as you do with other resources.

2.6.9 Stopping Ensembles Predictions

**Single predictions** are synchronous resources, so you cannot cancel them during the creation since you get the result immediately.

**Batch predictions** are asynchronous resources, so you can stop the creation before the task is finished. You can use the DELETE option from the 1-click action menu (Figure 2.92) or from the pop up menu on the prediction list view. (See Figure 2.93.) You can see in Figure 2.93 that the objective field column has the label **PROCESSING** to indicate the batch prediction is still in progress. If you stop the prediction during its creation, you will not be able to resume the same task again, so if you want to create the same prediction, you will have to restart a new task.

![Figure 2.92: Stop prediction from the 1-click menu](image)

![Figure 2.93: Stop prediction from the predictions list view](image)

2.6.10 Deleting Ensemble Predictions

You can DELETE your single or batch predictions from the predictions view, using the 1-click action menu (see Figure 2.94) or using the pop up menu on the predictions list view (see Figure 2.95.)
A modal window will be displayed asking you for confirmation. Once a prediction is deleted, it is permanently deleted and there is no way you (or even the IT folks at BigML) can retrieve it.

2.7 Consuming Ensembles

In the previous sections, we have described how you can create ensembles, configure them, use them to make predictions, and more. This section will introduce a number of BigML features that enable interesting ways of taking advantages of ensembles: exporting them locally, and using them programmatically via the BigML REST API and Bindings.

2.7.1 Exporting and Downloading Ensembles

You can export your ensemble in a number of programming languages, including Python, Java, and Node.js. Just click on the download icon in the top menu and select your preferred option.
The main goal of downloading your ensemble in a programming language is to make local predictions faster and at no cost. (See Figure 2.97.)

2.7.2 Using Ensembles Via the BigML API

Ensembles have full citizenship in the BigML API. This means you can programmatically create, update, list, delete, and use them for predictions. For example, this is how you can create an ensemble from the command line with custom values for a few available arguments. This will require you have properly set the `BIGML_AUTH` environment variable to contain your authentication credentials:

```
curl "https://bigml.io/ensemble?$BIGML_AUTH" \
-X POST \
-H 'content-type: application/json' \
-d '{"dataset": "dataset/4f66a80803ce8940c5000006",  
    "name": "my ensemble",  
    "number_of_models": 32}'
```

For more information on using ensembles through the BigML API, please refer to ensemble REST API documentation.

2.7.3 Using Ensembles Via the BigML Bindings

BigML bindings provide a convenient way to access BigML REST API from your language of choice. They offer a higher-level view of BigML Machine Learning resources and algorithms in a number of languages, including Python, Node.js, Java, Swift, and Objective-C. For example, this is how you can create an ensemble in Python using BigML bindings:

```
from bigml.api import BigML
api = BigML()
prediction = api.create_ensemble("dataset/573d997058a27e0f620038df",  
    "number_of_models": 32,  
    {"name": "my Ensemble"})
```

For more information on using ensembles through the BigML API, please refer to BigML bindings documentation.
2.8 Ensemble Limits

BigML imposes a few limits on the characteristics of an ensemble that it can handle. Some of them are the same that applies to models.

- **Number of trees**: A maximum of 1,000 trees are allowed for Decision Forests and a maximum of 2,000 trees for Boosted Trees.
- **Number of iterations**: A maximum of 1,000 iterations are allowed for Boosted Trees.
- **Fields**: There is no enforced limit to the number of fields that can be present in a model.
- **Instances**: There is no enforced limit to the number of instances that can be handled.
- **Classes**: A maximum number of 1,000 distinct classes per field is allowed.
- **Terms-tokens**: BigML can handle up to 1,000 tokens total. In case multiple text fields are defined, then the token limit per field is divided by the number of text fields.
- **Terms-full terms**: BigML can handle up to 256 characters total.
- **Items**: A maximum of 10,000 items per field is allowed.
- **Node threshold**: BigML supports a value between 3 and 2,000.

2.9 Descriptive Information

Each ensemble has an associated **name**, **description**, **category**, and **tags**. In Figure 2.98, you can see the options that the More info panel gives to edit them.

![Figure 2.98: Panel to edit an ensemble’s name, category, description and tags](image)

2.9.1 Name

Each ensemble has an associated **name** that is displayed on the list view and also on the top bar of an ensemble view. Ensemble names are indexed to be used in searches. When you create an ensemble, by default, it gets the name of the file that you used to create it. You can edit it using the More info menu option on the right corner of the ensemble view (see Figure 2.98).

The name of an ensemble cannot be longer than 256 characters. There is no restriction on the characters that can be used in an ensemble name. More than one ensemble can have the same name even...
within the same project, since they are automatically assigned unique internal identifiers.

2.9.2 Description

Each ensemble also has a description, taken from the dataset used to create them, that it is very useful for documenting your Machine Learning projects. Descriptions can be written using plain text and also markdown\(^{18}\). BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 2.99.)

![Figure 2.99: Markdown editor for ensemble descriptions](image)

Descriptions cannot be longer than 8192 characters and can use almost any character.

2.9.3 Category

Each ensemble has an associated category inherited from the dataset used to create it. Categories are useful to classify ensembles according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

An ensemble category must be one of the categories listed on table Subsection 2.9.3.

\(^{18}\)https://en.wikipedia.org/wiki/Markdown
Chapter 2. Ensembles

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace and Defense</td>
</tr>
<tr>
<td>Automotive, Engineering and Manufacturing</td>
</tr>
<tr>
<td>Banking and Finance</td>
</tr>
<tr>
<td>Chemical and Pharmaceutical</td>
</tr>
<tr>
<td>Consumer and Retail</td>
</tr>
<tr>
<td>Demographics and Surveys</td>
</tr>
<tr>
<td>Energy, Oil and Gas</td>
</tr>
<tr>
<td>Fraud and Crime</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Higher Education and Scientific Research</td>
</tr>
<tr>
<td>Human Resources and Psychology</td>
</tr>
<tr>
<td>Insurance</td>
</tr>
<tr>
<td>Law and Order</td>
</tr>
<tr>
<td>Media, Marketing and Advertising</td>
</tr>
<tr>
<td>Miscellaneous</td>
</tr>
<tr>
<td>Physical, Earth and Life Sciences</td>
</tr>
<tr>
<td>Professional Services</td>
</tr>
<tr>
<td>Public Sector and Nonprofit</td>
</tr>
<tr>
<td>Sports and Games</td>
</tr>
<tr>
<td>Technology and Communications</td>
</tr>
<tr>
<td>Transportation and Logistics</td>
</tr>
<tr>
<td>Travel and Leisure</td>
</tr>
<tr>
<td>Uncategorized</td>
</tr>
<tr>
<td>Utilities</td>
</tr>
</tbody>
</table>

Table 2.6: Categories used to classify ensembles by BigML

2.9.4 Tags

An ensemble can also have a number of tags associated with it that can help to retrieve it via BigML’s API or to provide ensembles with some extra information. Each tag is limited to a maximum of 128 characters. Each ensemble can have up to 32 different tags.

2.9.5 Counters

For each ensemble, BigML also stores a number of counters to track the number of other resources that have been created using it as a starting point. In the ensemble view, you can see a menu option that displays these counters. It also allows you to quickly jump to all the resources of one type that have been created with this ensemble as shown in Figure 2.100.

Figure 2.100: Menu option to quickly access to resources created with an ensemble
2.10 Ensemble Privacy

Privacy options for an ensemble can be defined in the More Info panel, displayed in Figure 2.101. There are two levels of privacy for BigML ensembles:

- **Private**: only accessible by authorized users (the owner and those who have been granted access).
- **Shared**: by enabling the secret link you will get two different links to share your ensemble. The first one is a sharing link that you can copy and send to others so they can visualize and interact with your ensemble chart. The second one is a link to embed your model directly on your web page.

You can also share the individual component models (see Section 1.11).

![Figure 2.101: Ensemble privacy options](image)

2.11 Moving Ensembles to Another Project

When you create an ensemble it will be assigned to the project where the original dataset used to create it belongs to.

Ensembles can only be assigned to a single project. However, you can move ensembles between projects. The Move to menu option to do this can be found in two places:

1. In the ensemble list view, within the 1-click actions for each ensemble (Figure 2.102).

![Figure 2.102: Menu option to move ensembles](image)
2. Within the one-click actions of an ensemble in the ensemble list view as you can see on Figure 2.103.

![Figure 2.103: Menu option to move ensembles from the ensemble list view](image)

In both cases, by selecting the MOVE TO menu option, you will be displayed a list of existing projects that you can assign your resource to. Alternatively, create a new project for your ensemble using the NEW PROJECT menu option.

### 2.12 Stopping Ensemble Creation

You can also stop an ensemble creation process while BigML is not yet done with it from the one-click actions menu (Figure 2.104). This can be useful, e.g., when you are creating an ensemble from a large dataset, or an ensemble with many nodes, and later realize that you should have done something differently, be it when configuring the resource, preparing the dataset, etc.

![Figure 2.104: Menu option to stop an ensemble creation](image)

A modal window (Figure 2.105) will be displayed asking you for confirmation.
Chapter 2. Ensembles

Figure 2.105: Menu option to stop an ensemble's creation

**Note:** If you stop the model during its creation, you will not be able to resume the same task. If you want to create the same model, you will have to start a new task.

### 2.13 Deleting Ensembles

You can delete your ensembles in two ways:

- From the ensemble view, using the **DELETE ENSEMBLE** option from 1-click action menu. (See **Figure 2.106**.)

  ![Figure 2.106: Menu option to delete a ensemble](image)

- Using the **DELETE ENSEMBLE** pop up menu option on the ensemble list view. (See **Figure 2.107**.)

  ![Figure 2.107: Ensemble deletion pop up menu option](image)

A modal window (see **Figure 2.108**) will be displayed asking you for confirmation. Once you delete an ensemble, it is deleted permanently and there is no way you (or even the IT folks at BigML) can retrieve
2.14 Takeaways

This chapter explains ensembles in detail. See below a list of key points:

- An ensemble is a collection of decision trees which are combined together to create a stronger model with better predictive performance.
- Ensembles are one of the best performing Machine Learning algorithms, often winning Machine Learning competitions across a multitude of domains and use cases.
- Ensembles are very fast to train and test, which significantly streamlines real-life Machine Learning projects.
- You can use ensembles to solve classification and regression problems.
- BigML provides three types of ensembles: Bagging (a.k.a. Bootstrap Aggregating), Random Decision Forests and Boosted Trees. Bagging builds each model from a random subset of dataset. Random Decision Forests adds an additional element of randomness by choosing random features at each split. Boosted Trees iteratively builds each single model trying to learn from the previous model mistakes.
- You can build ensembles from datasets that have been created in BigML. (See Figure 2.109)
- An ensemble can be an input to an evaluation, to a prediction, or to a batch prediction. (See Figure 2.109).
- BigML ensembles support any type of fields as input fields (categorical, numeric, text and items fields).
- You can create an ensemble with just 1-click or configure it as you wish. Ensembles are virtually parameter free, giving excellent results with no tuning.
- If you don’t specify any objective field, BigML will take the last valid field in your dataset.
- The default number of models for your ensemble is set to 10, and the maximum allowed from the Dashboard is 1,000 for Decision Forests and 2,000 for Boosted Trees.
- In BigML, you can choose three different pruning strategies when building your ensemble: smart pruning, statistical pruning, or no statistical pruning.
- By default, BigML ensembles don’t consider missing values when choosing splitting rules, but you can explicitly include them.
- BigML provides three different options to assign specific weight your instances: balance objective, objective weights, weight field.
• They inherit all the good qualities of individual trees including handling missing data and speed of prediction. However, they are not as easy to interpret as a single decision tree.

• You can visualize your ensemble using the ensemble chart. The chart is a graphic representation of the marginal effect a subset of input fields have on the objective field (ensemble predictions) disregarding the rest of the fields.

• As with individual decision trees, the field importance for ensembles provides a measure of a field importance on predictions relative to the others.

• You need to evaluate your ensemble’s performance with data that the ensemble has not seen before.

• For Decision Forests, the final prediction and, probability, confidence (or expected error), or votes is not known until all the component tree predictions are combined with the selected voting strategy (a.k.a. operating kinds in BigML).

• BigML provides three different strategies to combine the final Decision Forests prediction: probability, confidence and votes.

• Predictions for Boosted ensembles do not use combiners since the final prediction is an additive process rather than averaged.

• For classification Boosted Trees the probability of each class is returned at the prediction time while regression Boosted Trees do not have any accuracy measure for predictions.

• You can predict single instances or multiple instances in batch using your ensemble.

• BigML provides local predictions from the Dashboard for single instances, which allow you to get a real-time prediction without consuming any credits or requiring any internet connection.

• BigML batch predictions allow you to make simultaneous predictions for multiple instances. For batch predictions, you always get a CSV file and an optional output dataset.

• You can download an ensemble in a number of programming languages including Python, Java, Node.js, and Objective-C, among others, to use it in your local environment, and make predictions faster at no cost.

• You can furnish your ensemble with descriptive information (name, description, tags, and category).

• You can move an ensemble between different projects.

• Ensembles cannot be shared, but you can share the individual component models.

• You can stop the ensemble creation before the task has finished.

• You can permanently delete an ensemble.
Figure 2.109: Model Workflows
Linear Regressions

3.1 Introduction

There are multiple Machine Learning problems that can be solved using supervised Machine Learning techniques. These problems require predicting an output variable (objective field) given a number of input variables (input fields). They can be divided into classification and regression depending on whether you need to predict a category (label or class) or a continuous value (a real number), respectively. To learn more about concrete use cases for both problems refer to Section 1.1.

Linear Regression is a supervised Machine Learning technique that can be used to solve regression problems. These problems can also be solved with other Machine Learning methods, such as models, ensembles, or deepnets. These methods are explained in Chapter 1, Chapter 2, and Chapter 5 respectively. Depending on the problem you are trying to solve and the data available, some techniques may perform significantly better than others. The main difference between linear regression and others is that linear regression assumes your objective field has a linear relationship with your input fields. For this reason, linear regressions work best in those problems where this assumption is accurate.

This chapter contains a comprehensive description of BigML’s linear regression models including how they can be created with 1-click (Section 3.3), all configuration options available (Section 3.4), and the different visualizations provided by BigML (Section 3.5). See Section 3.6 for an explanation of how linear regressions can be used to make predictions. You can also export your linear regressions in different formats to make local predictions faster at no cost (Subsection 3.6.4.1). The process to evaluate your linear regressions’ predictive performance in BigML is explained in a different chapter (Chapter 7).

In BigML, the third tab (Supervised) of the main menu of the Dashboard allows you to list all of your available linear regressions. The linear regression list view (Figure 3.1), details the Dataset used to create it, the Name, Objective (objective field field name), Age (time elapsed since it was created), Size, and number of evaluations, predictions, and batch predictions that have been created using that linear regression. The SEARCH menu option in the top right corner of the linear regression list view allows you to search your linear regressions by name.
By default, when you first create an account at BigML, or every time that you start a new project, your list of linear regressions will be empty. (Figure 3.2)

Finally, in Figure 3.3 you can see the icon used to represent a linear regression.

Figure 3.1: Linear regression list view

Figure 3.2: Empty Dashboard linear regression view

Finally, in Figure 3.3 you can see the icon used to represent a linear regression.

Figure 3.3: Linear Regression icon

3.2 Understanding Linear Regressions

As mentioned in the introduction of this chapter, linear regression is a supervised learning algorithm used to solve regression problems. It’s simple to understand and has very good interpretability.

Linear Regression assumes a linear relationship between the input fields, also called predictors, and the single objective field, or the output variable. More specifically, the objective field can be modeled from a linear combination of the input fields:
\[ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_n x_n \]

This is the linear regression formula, also called linear equation, where \( y \) is the objective field, \( (x_1, x_2, \ldots, x_n) \) represent the \( n \) variables, also called predictors, for the input fields in the input data, and \( (\beta_1, \beta_2, \ldots, \beta_n) \) are the coefficients which are the scale factors assigned to the respective variables. The one additional coefficient \( \beta_0 \) is often called the intercept or the bias coefficient.

Learning a linear regression model means estimating the values of the coefficients with the data available. A positive coefficient \( (\beta_i > 0) \) for a input field, indicates a positive correlation between the input field and the objective field, while a negative coefficient \( (\beta_i < 0) \) indicate a negative correlation. Higher absolute coefficient values for a field results in a greater influence of that field on predictions. When a coefficient becomes zero, it effectively removes the influence of the input field on the model and hence on the predictions.

BigML Linear Regression produces an estimate for the coefficient values \( \beta_0, \beta_1, ..., \beta_n \) using a least-squares fit on the training data.

BigML provides a table containing all your linear regression coefficients. (See Subsection 3.5.2)

When the linear regression has learned the coefficients, you can use the model to make predictions for new instances. In a prediction, the linear regression returns a predicted value of the objective field.

By definition, the input fields \( (x_1, x_2, \ldots, x_n) \) in the linear regression formula need to be numeric values. However, BigML linear regressions can support any type of fields by applying a set of transformations to categorical, text, and items fields. Moreover, BigML can also handle missing values for any type of field. The following subsections detail both behaviors.

### 3.2.1 Input Field Transformations

Apart from numeric fields, BigML linear regressions are optimized to support categorical, text and items fields by applying a set of transformations in order to convert them to numeric values:

- **Categorical fields** are Dummy encoded by default. Dummy encoding converts one \( n \)-class categorical field to \( n \) separate variables. One of the classes is designated as the reference or dummy class, which is assigned a value of 0 for each variable. The dummy class, if not specified by the user, is the first class value in lexicographic order. So there are \( n-1 \) variables (\( n \) classes minus the dummy class); one additional variable for missing values is created if the training dataset contains missing values for this field. For a given instance, the variable corresponding to the instance’s categorical value has its value set to 1 (except if the categorical value is the dummy class), while the other variables are set to 0.

  BigML also provides Contrast coding and Other coding, that you can configure for each of your categorical fields. See Subsection 3.4.6 for a complete explanation of categorical fields encoding.

- **For text fields**, each term is mapped to a corresponding numeric variable, whose value is the number of occurrences of that term in the instance. Text fields without term analysis enabled are excluded from the model (read the Sources with the BigML Dashboard document to learn more about text analysis [11]).

- **For Items fields**, each different item is mapped to a corresponding numeric field, whose value is the number of occurrences of that item in the instance.

### 3.2.2 Missing Values

BigML linear regressions can handle missing values for any type of field. For categorical, text, and items fields, missing values are always included as another category, term or item by default.

For numeric fields, missing values are always included. If a field in the training data contains missing data, then a corresponding binary-valued predictor will be created which takes a value of 1 when that field is missing in a particular row, and 0 otherwise. The other predictors pertaining to that field will have a value of 0 when the value is missing. Once the linear regression is created, you can find an additional coefficient for each field at the end of the coefficient table. (See Figure 3.4) Learn more about the coefficient table in Subsection 3.5.2.
Alternatively, you can replace your missing numeric values by another valid value like the field’s mean, median, maximum, minimum or zero (see Subsection 3.4.3).

If the input data does not contain missing values for a field, the coefficient for missing values will be zero, except in the case of text fields which can be different from zero. This is due to the fact that BigML has a limit of 1,000 terms for text fields, so there may be instances not containing any of the terms considered to build the model and appear as missing values instead. (See Subsection 3.8.0.1 to know more about term limits for text fields.)

![Figure 3.4: Missing numeric coefficients at the end of linear regression table](image)

### 3.2.3 Number of Predictors

Because of input field transformations and missing values, one input field may become more than one predictors in the linear equation to fit the data. This table summarizes the number of predictors which will be generated for each input field type.
<table>
<thead>
<tr>
<th>Input field type</th>
<th>No Missing Value</th>
<th>Missing Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Categorical, dummy-encoded</td>
<td>(number of classes) - 1</td>
<td>number of classes</td>
</tr>
<tr>
<td>Categorical, contrast or other-encoded</td>
<td>number of classes</td>
<td>(number of classes) + 1</td>
</tr>
<tr>
<td>Text</td>
<td>number of terms</td>
<td>(number of terms) + 1</td>
</tr>
<tr>
<td>Items</td>
<td>number of items</td>
<td>(number of items) + 1</td>
</tr>
</tbody>
</table>

Table 3.1: Number of predictors per input field

Bias term, which is also called intercept term and enabled by default, is a predictor.

For example, if the data has 10 numeric input fields, 2 of them have missing values, it will generate $8 + 2 \times 2 = 12$ predictors. It also has 2 categorical fields, one has 6 classes and is dummy-encoded without missing values, another has 3 classes and is contrast-encoded with missing values, which will generate $6 - 1 + 3 + 1 = 9$ predictors. It's also got a text field which has 15 terms and missing values, and this will generate $15 + 1 = 16$ predictors. In addition, it has 1 items field which has 8 items and no missing values and this will become 8 predictors. Bias term is enabled. Altogether, this linear regression will have 46 predictors from its 14 input fields.

### 3.2.4 Ill-Conditioned Problems

A linear regression is ill-conditioned when there is insufficient data to estimate the value of the coefficients. Typically, this is when the number of rows is fewer than the number of predictors. In this case, the coefficients which are unable to be estimated will be set to 0, and in the JSON response of the model, the stats output will not contain `standard_errors`, `z_values`, `confidence_intervals`, and `p_values`. A warning will also be added to the model status.

Predictions with an ill-conditioned linear regression will have confidence and prediction intervals equal to 0.

### 3.3 Creating Linear Regressions with 1-Click

To create a linear regression in BigML you have two options: either the 1-click option which uses the default values for all available configuration options, or you can tune the parameters in advanced by using the configuration options explained in Section 3.4. This section explains how to create a linear regression with 1-click.

From the dataset view, select the 1-CLICK LINEAR REGRESSION option in the 1-click action menu menu. (See Figure 3.5)
Figure 3.5: Create 1-click linear regression from dataset 1-click action menu

Alternatively, you can use the 1-CLICK LINEAR REGRESSION option in the pop up menu from the dataset list view. (See Figure 3.6.)

Figure 3.6: Create 1-click linear regression from dataset popup menu

Either option builds a linear regression using the default values for all available configuration options. (See Section 3.4.)

Note: for some datasets, the 1-click option may be disabled. This can be due to the fact that your dataset does not contain any numeric field or the field taken as the default objective field field is not numeric. If you do not specify any objective field BigML takes the last numeric field in your dataset as the objective field by default. To change the objective field, either configure your linear regression or select the objective field from your dataset (both options are explained in Subsection 3.4.1)
3.4 Linear Regression Configuration Options

While the 1-click creation menu option (see Section 3.3) provides a convenient and easy way to create a BigML linear regression, you can also have more control over the linear regression creation and configure a number of parameters that affect the way BigML creates linear regressions. Click the **CONFIGURE LINEAR REGRESSION** menu option in the **configuration menu** of your dataset view. (See Figure 3.7.)

![Figure 3.7: Configure linear regression](image)

3.4.1 Objective Field

The objective field, or “target field”, is the field you want to predict. Linear regressions only support **numeric** fields as the objective field.

BigML takes the **last numeric field** in your dataset as the objective field by default. If you want to change the objective field, you have two options: you can select another field from the configuration panel to build the linear regression, or you can change it permanently from your dataset view.

- Select the **Objective field** from the linear regression **configuration panel**. This option will only affect the linear regression you are building that time. (See Figure 3.8.)
Figure 3.8: Configure the objective field to create the linear regression

- Change the **default objective field** for the dataset. This option will save your objective field preference for any model you build. Click on the edition icon next to the field name when you mouse over it, a pop up window will be displayed. Then click on the **Objective field** icon and **Save** it. (See Figure 3.9.)

Figure 3.9: Change the default objective field
3.4.2 Automatic Optimization

You can turn on the **Automatic optimization** option so BigML will automatically tune the parameters of your linear regression (see Figure 3.10).

![Figure 3.10: Automatic optimization](image)

The main focus of optimization in linear regression is the bias term, also known as the intercept term. Hand-tuning it is a time-consuming process and BigML offers first-class support for automatic linear regression parameter optimization.

Behind the scenes, BigML uses the same technology for linear regression parameter optimization as the one used for OptiML. If you want to know more about the technical details, please read the Chapter 2 of the document OptiML with the BigML Dashboard [10].

When you turn on the **Automatic optimization** option, all the linear regression parameters will be disabled (because they will be automatically optimized), except the **Default numeric value** and the **Weights** parameters which you can manually configure (see Figure 3.11).

![Figure 3.11: Configure the default numeric value](image)

Since the optimization process can take some time, BigML offers two configurable parameters to limit the time to create the optimized linear regression: a training duration (see Subsection 3.4.2.1) and the linear regression candidates (see Subsection 3.4.2.2).
3.4.2.1 Training duration

The scale parameter to regulate the linear regression runtime. It's set as an integer from 1 to 10. It indicates the user preference for the amount of time they wish the optimization to take. The higher the number, the more time that users are willing to wait for possibly better linear regression performance. The lower the number, the faster that users wish the linear regression training to finish. The default value is set to 5.

The training duration is set in a scale. The actual training time depends on the dataset size, among other factors.

![Figure 3.12: Training duration](image)

3.4.2.2 Linear regression candidates

The maximum number of different linear regressions (i.e., linear regressions using a unique configuration) to be trained and evaluated during the optimization process. The default number is 128 candidates which is usually enough to find the best linear regression, but you can set it from 4 up to 200. Only the top-performing linear regression will be returned. If the training duration is very low (see Subsection 3.4.2.1) given the dataset size, it is possible that not all the linear regression candidates will be tried out.

![Figure 3.13: Linear regression candidates](image)
3.4.3 Default Numeric Value

Linear regressions can include missing values as valid values for any type of fields as explained in Subsection 3.2.2. However, there can be situations for which you don’t want to include them in your model. For those cases, the Default numeric value parameter is an easy way to replace missing numeric values by another valid value. You can select to replace them by the field’s Mean, Median, Maximum, Minimum or by Zero. (See Figure 3.14.)

Figure 3.14: Select a default numeric value to replace missing numeric values

Note: if your dataset does not contain missing values for your numeric fields, this parameter will not have impact on your linear regression.

3.4.4 Weights

It is not unusual for a dataset to have unbalanced fields, which means there are many instances in certain ranges, and few in others. For example, in datasets used to model company financials, there are many more companies with employees numbered from 50-500, while there are only a few with more than 100,000 employees. So as the company size increases, there are fewer cases to fit. In that case, you may want to assign more weights to the scarce instances so they are equivalent to the abundant ones.

BigML provides an option to assign specific weights to your instances (see Figure 3.15).
3.4.4.1 Weight Field

The **Weight field** option allows you to assign individual weights to each instance by choosing a special weight field. The selected field should be integer, with a minimum value of 1, and it must not contain any negative or missing values. However, any non-negative weight field will be accepted. If the minimum value is different from 1, each value in the weight field will be divided by the minimum value and rounded to the nearest integer.

If an instance has a weight of 3 it will be replicated three times in the dataset to train the model.

The weight field will be excluded from the input fields when building the linear regression. You can select an existing field in your dataset or you may create a new one in order to assign customized weights.

3.4.5 Bias

You can include or exclude the **Bias** from the model, a.k.a. the intercept term of the linear regression formula. (See formula in Section 3.2.) For most cases, including the bias results in a better model. By default it is included. (See Figure 3.16.)
3.4.6 Field Codings

Categorical fields must be converted to numeric values in order to train a linear regression model. By default, they are Dummy encoded, with the default dummy class as the first class in lexicographic order. BigML also allows you to configure two other types of coding for each one of your categorical fields: Contrast coding, and Other coding. See the following subsections for a detail explanation of each option. (Learn more about input fields transformations in Subsection 3.2.1.)

3.4.6.1 Dummy Coding

The main goal of using dummy coding\(^1\) is to compare a class selected as the reference or control class with the rest of classes. The control class is assigned a value of 0 for each variable. The control class is called dummy class in BigML and it is usually a class with a representative number of instances compared to the other classes in the dataset. See an example of dummy coding schema for three different classes, with the “Class 1” being the dummy class, in Table 3.2:

<table>
<thead>
<tr>
<th>Classes</th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Class 2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Class 3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MISSING</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.2: Dummy coding example for 3 classes

\(^1\)https://en.wikipedia.org/wiki/Categorical_variable#Dummy_coding
To set **Dummy coding** for a field:

1. Click on the configuration icon next to the field name. (See Figure 3.17.)

![Figure 3.17: Field coding configuration](image)

2. A modal window will be displayed so you can configure the field codings for that field. If the field does not have a previous configuration for field codings, it will be disabled. **Enable** field coding configuration by clicking on the green switcher shown in Figure 3.18.
Figure 3.18: Enable field coding configuration

3. Select the class you want to set as the dummy class. (See Figure 3.19.)

Figure 3.19: Select the dummy class

4. Click [Save]. Make sure you saved your configuration by looking at the bottom message “Configured Coding: DUMMY”. (See Figure 3.20.)
5. Close the modal window by clicking outside or by clicking **Cancel**.

6. After configuring the field codings for a field, the configuration icon will become green. (See Figure 3.22.)
7. To remove the field coding configuration for that field, click **Disable** from the switcher and click **Save** again. (See Figure 3.23.)
After creating your linear regression, your dummy class will be identified with the dummy icon in the coefficients table view (see Subsection 3.5.2). (See Figure 3.24.)

Figure 3.24: Dummy class in table view

### 3.4.6.2 Contrast Coding

Contrast coding\(^2\) allows you to set different values for different classes. Instead of the 0-1 values of Dummy coding, you will be able to set any integer or float value for each of the classes, plus an additional one for missing values. The sum of all values must equal 0. The values of the classes need to be set based on certain hypothesis, e.g., higher values for a class assume this class has more influence on the objective field than the others. A positive value indicates a positive relationship between the class and the objective field while a negative value indicates a negative relationship. A coefficient of 0 will exclude the class from the model. In the Table 3.3 you can see an example of contrast coding schema for three different classes.

<table>
<thead>
<tr>
<th>Classes</th>
<th>C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>0.5</td>
</tr>
<tr>
<td>Class 2</td>
<td>-0.25</td>
</tr>
<tr>
<td>Class 3</td>
<td>-0.25</td>
</tr>
<tr>
<td>MISSING</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.3: Contrast coding example for 3 classes

To set Contrast coding for a field, follow these steps:

1. Click on the configuration icon next to the field name. (See Figure 3.25.)

\(^2\)https://en.wikipedia.org/wiki/Categorical_variable#Contrast_coding
2. A modal window will be displayed so you can configure the field codings for that field. If the field does not have a previous configuration for field codings, it will be disabled. **Enable** field coding configuration by clicking on the green switcher shown in Figure 3.26.
3. Select the **Contrast coding** option. (See Figure 3.27.)

4. Set the values you want for your classes based on your hypothesis. All classes values must sum 0. (See Figure 3.28.) By using the BigML API, multiple contrast codings can be given for a field as long as all the codings are **orthogonal** to ensure there are no co-dependent coefficients. Check the corresponding
Figure 3.28: Set the contrast coding values for each class

Note: you cannot select several field codings for the same field simultaneously.

5. Click Save. Make sure you saved your configuration by looking at the bottom message “Configured Coding: CONTRAST”. (See Figure 3.29.)

3https://bigml.com/api/linearregressions#lr_coding_categorical_fields
Chapter 3. Linear Regressions

6. Close the modal window by clicking outside or by clicking \textbf{Cancel}.

7. After configuring the field codings for a field, the configuration icon will become green. (See Figure 3.31.)

Figure 3.29: Contrast coding saved

Figure 3.30: Close modal window

\textbf{Note:} if the \textbf{Cancel} button is red, it indicates there are changes you have not saved yet so you will lose them by closing the modal window.
8. To remove the field coding configuration for that field, click Disable from the switcher and click Save again. (See Figure 3.32.)
After creating your linear regression, you will be able to see your **Contrast coding** values in the **coefficients table** view (see Subsection 3.5.2) by clicking on the icon. (See Figure 3.33.)

A modal window will be displayed with your codings values and you can download them in CSV or JSON.
format by clicking on the corresponding icons. (See Figure 3.34.)

![Figure 3.34: Contrast modal window in table view](image)

### 3.4.6.3 Other Coding

Other coding allows you to set different values for different classes. It works the same way as contrast coding (see Subsection 3.4.6.2), but in this case the values do not need to sum 0. In the Table 3.4 you can see an example of other coding schema for three different classes.

<table>
<thead>
<tr>
<th>Classes</th>
<th>C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>2</td>
</tr>
<tr>
<td>Class 2</td>
<td>-0.4</td>
</tr>
<tr>
<td>Class 3</td>
<td>3</td>
</tr>
<tr>
<td>MISSING</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.4: Other coding

To set Other coding for a field, follow these steps:

1. Click on the configuration icon next to the field name. (See Figure 3.35.)

---

4[https://en.wikipedia.org/wiki/Categorical_variable#Contrast_coding](https://en.wikipedia.org/wiki/Categorical_variable#Contrast_coding)
Chapter 3. Linear Regressions

2. A modal window will be displayed so you can configure the field codings for that field. If the field does not have a previous configuration for field codings, it will be disabled. **Enable** field coding configuration by clicking on the green switcher shown in Figure 3.36

![Figure 3.36: Enable field coding configuration](image_url)
3. Select the **Other coding** option. (See Figure 3.37.)

4. Set the values you want for your classes based on your hypothesis. You can set any float or integer value. (See Figure 3.38.) By using the BigML API, multiple other codings can be given for a field. Check the corresponding [documentation](https://bigml.com/api/linearregressions#lr_coding_categorical_fields).
Note: you cannot select several field codings for the same field simultaneously.

5. Click Save. Make sure you saved your configuration by looking at the bottom message “Configured Coding: OTHER”. (See Figure 3.39.)

![Figure 3.39: Other coding saved](image1)

6. Close the modal window by clicking outside or by clicking Cancel.

![Figure 3.40: Close modal window](image2)

Note: if the Cancel button is red, it indicates there are changes you have not saved yet so you will lose them by closing the modal window.
7. After configuring the field codings for a field, the configuration icon will become green. (See Figure 3.41.)

![Field codings configured](image)

**Figure 3.41: Field codings configured**

8. To remove the field coding configuration for that field, click **Disable** from the switcher and click **Save** again. (See Figure 3.42.)
After creating your linear regression, you will be able to see your Other coding values in the coefficients table view (see Subsection 3.5.2) by clicking on the icon. (See Figure 3.43.)

A modal window will be displayed with your coding values and you can download them in CSV or JSON format by clicking on the corresponding icons. (See Figure 3.44.)
3.4.7 Sampling Options

Sometimes you do not need all the data contained in your dataset to build your linear regression. If you have a very large dataset, sampling may be a good way of getting faster results. BigML allows you to sample your dataset before creating the linear regression, so you do not need to create a separate dataset first. You can find a detailed explanation of the sampling parameters available in the following subsections. (See Figure 3.45.)

3.4.7.1 Rate

The Rate is the proportion of instances to include in your sample. Set any value between 0% and 100%. Defaults to 100%.

3.4.7.2 Range

Specifies a subset of instances from which to sample, e.g., choose from instance 1 until 200. The Rate you set will be computed over the Range configured. This option may be useful when you have temporal data, and you want to train your linear regression with historical data, and test it with the most recent one to check if it can predict based on time.

3.4.7.3 Sampling

By default, BigML selects your instances for the sample by using a random number generator, which means two samples from the same dataset will likely be different even when using the same rates and row ranges. If you choose deterministic sampling, the random-number generator will always use the same seed, thus producing repeatable results. This lets you work with identical samples from the same dataset.

3.4.7.4 Replacement

Sampling with replacement allows a single instance to be selected multiple times. Sampling without replacement ensures that each instance cannot be selected more than once. By default, BigML generates samples without replacement.

3.4.7.5 Out of bag

This argument will create a sample containing only out-of-bag instances for the currently defined rate, so the final total number of instances for your sample will be one minus the rate configured for your sample (when replacement is false). This can be useful for splitting a dataset into training and testing subsets. It is only selectable when a sample rate is less than 100%.
3.4.8 Advanced Ordering

Ordering options are relevant to ensure that BigML can correctly determine whether it can take an early split of your dataset to accelerate the training process. In particular, early splitting can only be safely used if the training instances have been previously shuffled.

If your instances are already shuffled, BigML allows you to choose the linear option. This will make the process of building the model much faster, since it will not required to reshuffle the dataset. If you need to shuffle your instances, BigML provides two options to that aim, deterministic shuffling and random shuffling, which are described below.

Ordering options have no influence on datasets of less than 34GB, since the whole dataset is used to build the model.

By default, BigML uses deterministic shuffling to ensure the same (deterministic) sample of the instances is used and the built model is thus repeatable.

3.4.8.1 Deterministic Shuffling

The deterministic shuffling option ensures that the row shuffling of a dataset is always the same, so that retraining a BigML model from the same dataset yields the same results.

By default, this option is true.

3.4.8.2 Linear Shuffling

The linear shuffling option is useful when you know that your instances are already in random order. Using linear shuffling, the BigML model will be constructed faster.

By default, this option is false.

3.4.8.3 Random Shuffling

The random shuffling option will ensure that a different shuffling will be tried each time you train your model.

By default, this option is false.
3.4.9 Creating Linear Regressions with Configured Options

After finishing the configuration of your options, you can change the default linear regression name in the editable text box. Then you can click on the Create linear regression button to create the new linear regression, or reset the configuration by clicking on the Reset button.

Copyright © 2023, BigML, Inc.
3.4.10 API Request Preview

The API Request Preview button is in the middle on the bottom of the configuration panel, next to the Reset button (See Figure 3.47). This is to show how to create the linear regression programmatically: the endpoint of the REST API call and the JSON that specifies the arguments configured in the panel. Please see Figure 3.48 below:

![Linear regression API request preview](image)

There are options on the upper right to either export the JSON or copy it to clipboard. On the bottom there is a link to the API documentation for linear regressions, in case you need to check any of the possible values or want to extend your knowledge in the use of the API to automate your workflows.

Please note: when a default value for an argument is used in the configuration, the argument won’t appear in the generated JSON. Because during API calls, default values are used when arguments are missing, there is no need to send them in the creation request.

3.5 Visualizing Linear Regressions

After creating your linear regression, you will be able to analyze your results with BigML unique visualization: a 1D chart and a Partial Dependence Plot (PDP), to examine the impact of the input fields in the objective field, and a coefficients table, for more advanced users, to interpret the resulting coefficients for each input field. The following subsections explain both visualizations in detail.

Switch among the three views of 1D chart, PDP and table by clicking on icons in the top bar menu of the linear regression view. (See Figure 3.49.)
3.5.1 Linear Regression Chart

Both views of 1D chart and PDP are composed of three main parts: the CHART or PLOT, the PREDICTION legend and the INPUT FIELDS form. (See Figure 3.50). You can find a detailed explanation of each one below.

- Both the 1D Chart and the Partial Dependence Plot (PDP) allow you to view the impact of the input fields on the objective classes predictions, and their relationship. You can select the 1D chart or PDP by clicking on the selection buttons in the top bar menu.

The **1D chart**, allows you to select one input field for the x-axis. The y-axis represents the objective field. Only numeric fields can be selected for the x-axis. (See chart limitations in Subsection 3.8.1). You can extend the upper limit of the x-axis by clicking on the plus icon.

The blue band is formed by the upper bounds and lower bounds of the 95% prediction intervals. This means, for any given point at x-axis, its y value will be within this blue range with 95% probability. You can hide or show the band by clicking on the icon next to the field name of the y-axis.
The **PDP** allows you to select two different input fields for both axis and the values of the objective field are represented by differences in a color scale in the heatmap chart. You can select numeric or categorical fields for the axis. You can switch the axis by clicking on the option on top of the chart area.

Figure 3.51: 1D chart
In both charts you can inspect the axis values in the gray area boxes next to the selector. You can freeze the view by pressing `Shift` and release it again by pressing `Esc` from your keyboard. When the view is frozen, an edit icon will appear you can edit the axis values and obtain a prediction for another preferred value. The resulting predicted probabilities are in the prediction legend to the right.

- The PREDICTION legend allows you to visualize the predicted values represented in the chart along with their corresponding colors. By default, in PDP, colors are shaded according to the prediction range shown in the chart area. This way, smaller differences in predictions are easier to perceive. However, you can choose to see the color shading according to the total range of values for the objective field by clicking on the icon next to the prediction bar `Total`. This `Total` option allows you to see the color scale for the total range of predictions. (See Figure 3.53)
Again, freeze this view by pressing `Shift`, and release it again by pressing `Esc` from your keyboard.

- Below the chart legend, you can find the INPUT FIELDS form. (See Figure 3.54). You can configure the values for any numeric, categorical, text or items field. By changing their values, you can see the predictions changing in real-time.
Moreover, the chart includes a reset option for your input fields values, and an export option to download your chart in PNG format explained below:

- After selecting the fields for the axis or configuring the input fields values, you can set them again to the default view by clicking the **reset** icon highlighted in Figure 3.55.

  ![Figure 3.55: Reset the values for the input fields](image)

- You can also **export** your chart in PNG format with or without legends. Freeze the view by pressing **Shift** from your keyboard and export the chart to get the classes percentages in the legend. Release the view by pressing **Esc**.

Copyright © 2023, BigML, Inc.
Chapter 3. Linear Regressions 207

3.5.2 Coefficient Table

The main goal of the linear regression algorithm is to learn the coefficients of the linear function for each of the dependent variables, i.e., for each of the input fields. See Section 3.2 for a detailed explanation of linear regression coefficients interpretation.

BigML allows you to inspect the learned coefficients for each one of the input fields in the coefficient table. The table columns represent coefficients and their statistics while the table rows represent the input field variables and the Bias (a.k.a. intercept term) of the linear regression. In the first row you will always find the Bias coefficients. You can sort the table rows by clicking on any of the columns labels.

Note: there are some limitations in the number of input fields to visualize your linear regression in the chart (explained in Subsection 3.8.1).
For numeric fields, there is always one coefficient by field, however categorical, text and items fields have one coefficient by value (category, term or item). This is due to the required transformations, explained in Section 3.2, to convert categorical, text, and items fields to numeric fields (each single value is mapped to a separate variable in the formula). Missing values also get their own coefficients.

- For numeric fields, you always get one coefficient per field. If a field contains missing values, you will find an additional coefficient per field for the missing values. (See Subsection 3.2.2.)

- For categorical fields, you have one coefficient per class and an additional one for missing values per field. (See Subsection 3.2.3 and Subsection 3.4.6.)

- For text fields, there is one coefficient per term and an additional one for missing values per field.

- For items fields, you get one coefficient per item and an additional one for missing values per field.

See an example of coefficients for a categorical field in Figure 3.58 where one single field, “Atmospheric condition”, yields eleven different variables associated with different coefficients. There are twelve classes in the categorical field, with one set as the dummy class.
Chapter 3. Linear Regressions

Figure 3.58: Multiple field variables for categorical dummy encoded fields

Coefficients for **missing values** are always found at the end of the table. (See **Figure 3.59**)
Figure 3.59: Missing numeric coefficients at the end of linear regression table

Next to each coefficient you will find one icon indicating if it is significant (see Figure 3.60) or non-significant (see Figure 3.61).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias</td>
<td>12.3</td>
<td>3.90015</td>
</tr>
<tr>
<td>Sex = M</td>
<td>ABC</td>
<td>0.03022</td>
</tr>
<tr>
<td>Sex = F</td>
<td>ABC</td>
<td>-0.82452</td>
</tr>
<tr>
<td>Length</td>
<td>12.3</td>
<td>0</td>
</tr>
<tr>
<td>Diameter</td>
<td>12.3</td>
<td>11.07560</td>
</tr>
<tr>
<td>Height</td>
<td>12.3</td>
<td>10.73700</td>
</tr>
<tr>
<td>Whole_weight</td>
<td>12.3</td>
<td>8.93378</td>
</tr>
<tr>
<td>Shucked_weight</td>
<td>12.3</td>
<td>-19.76880</td>
</tr>
<tr>
<td>Viscera_weight</td>
<td>12.3</td>
<td>-10.57760</td>
</tr>
<tr>
<td>Shell_weight</td>
<td>12.3</td>
<td>8.73112</td>
</tr>
<tr>
<td>Sex</td>
<td>missing</td>
<td>1.88662</td>
</tr>
<tr>
<td>Diameter</td>
<td>missing</td>
<td>2.90368</td>
</tr>
</tbody>
</table>

The significance of a coefficient is determined by comparing the $p$-value against the significance level selected in the top menu (see Figure 3.62). If the $p$-value is higher than the significance level, the coefficient will be non-significant. If the $p$-value is lower than the significance level, the coefficient will be significant. A good practice is to retrain the linear regression removing the non-significant coefficients. However, in most cases, the model performance should not be affected.
Next to each icon indicating the significance of a coefficient, you will find a $\sigma$ symbol. If you mouse over it, a tooltip will display a summary of the stats for that coefficient. (See Figure 3.64.) First, you will find the $p$-value from the Wald test\(^6\). As mentioned in the previous point, this $p$-value is compared against the selected significance level to determine the coefficient’s significance. Then, associated with the Z score chart, you will find the Z score value, the confidence interval for a 95% confidence and the standard error, or variance, of the coefficient estimate.

\(^6\)https://en.wikipedia.org/wiki/Wald_test
You can download all the stats information by clicking in the download CSV icon in the top menu to the right. (See Subsection 3.7.1.)

Additional options for the table include a filtering option and an export option:

- You can filter the table first column by field name, class, term or item using the search box at the top of the table (see Figure 3.65.)
Chapter 3. Linear Regressions

Figure 3.65: Search and filter linear regression table

- You can also export the table in a CSV file by clicking on the icon highlighted in Figure 3.66.

Figure 3.66: Export table in CSV file
3.6 Linear Regression Predictions

3.6.1 Introduction

The ultimate goal in building a linear regression is being able to make predictions with it. In BigML, you can make predictions for single instances or for many instances in batch. Each prediction comes with a measure, prediction interval, indicating the 95% confidence range for the predicted value.

The predictions tab in the main menu of the BigML Dashboard is where all your saved predictions are listed. (See Figure 3.67.) You can search your predictions by name clicking on the search option on the top menu. In the predictions list view, you can see, for each prediction, the linear regression icon used for the prediction, the Name of the prediction, the Objective (objective field name), the Prediction (the prediction result), and the Age (time since the prediction was created).

Figure 3.67: Predictions list view

When you first create an account at BigML, or every time that you start a new project, your list of predictions will be empty. (See Figure 3.68).

Figure 3.68: Empty predictions list view

Linear regression predictions are saved under the CLASSIFICATION & REGRESSION option in the menu (see Figure 3.69).

Copyright © 2023, BigML, Inc.
3.6.2 Creating Linear Regression Predictions

BigML provides two options to predict with your linear regressions explained in the following subsections:

- **PREDICT**: to predict one single instance
- **BATCH PREDICTION** to predict multiple instances in batch.

### 3.6.2.1 Predict

BigML allows you to quickly make predictions for single instances by providing a form containing the input fields used by the linear regression, so you can easily set the values and get an immediate response.

Follow these steps to create a single prediction:

1. Click **PREDICT** in the linear regression 1-click action menu. (See Figure 3.72)
2. You will be redirected to the **prediction form** where you will find all the fields used by the linear regression as input fields. (See Figure 3.74.)
3. **Set input values** for your selected fields. BigML supports numeric, categorical, text and items fields as inputs.

4. **Get the prediction** at the top of the view along with the prediction interval. (See Figure 3.75.) BigML predictions are synchronous, i.e., when you send the input data, you get an immediate response. Moreover, single predictions from the BigML Dashboard are performed locally, so unless you save your prediction, it will not consume any credits and it will be updated instantly when you change your input values. Learn more about local predictions in Subsection 3.6.4.1.
Figure 3.75: Get the linear regression prediction

5. Optionally, you can Save the linear regression prediction, so you will find it afterwards in the predictions list view. (See Figure 3.76.)
Chapter 3. Linear Regressions

3.6.2.2 Batch Predictions

BigML batch predictions allow you to make simultaneous predictions for multiple instances. All you need is the linear regression you want to use to make predictions and a dataset containing the instances you want to predict. BigML will create a prediction for each instance in the dataset.

Follow these steps to create a batch prediction:

1. Click on BATCH PREDICTION option under the linear regression 1-click action menu (Figure 3.77).

Alternatively, click on CREATE BATCH PREDICTION in the pop up menu of the list view (Figure 3.78).

---

Note: this option is only available from the BigML Dashboard for linear regressions with less than 100 fields. If you want to perform single instance predictions for a higher number of fields, use the BigML API.

---

https://bigml.com/api/predictions
2. **Select the dataset** containing all the instances you want to predict. The instances should contain the input values for the fields used by the linear regression as input fields. From this view you can also select another linear regression from the selector or even a model or ensemble by clicking on the icons on the top left menu. (See Figure 3.79.)

3. After the linear regression and the dataset are selected, the batch prediction **configuration options** will appear along with a **preview** of the prediction output (a CSV file). (See Figure 3.80.) The default output format includes all your prediction dataset fields and adds an extra column with the class predicted. See Subsection 3.6.3 for a detailed explanation of all configuration options.
4. By default, BigML generates an output dataset with your batch predictions that you can later find in your datasets section in the BigML Dashboard. This option is active by default but you can deactivate it by clicking on the icon shown in Figure 3.81.

Figure 3.80: Configuration options for linear regression batch prediction
Figure 3.81: Create a dataset from batch prediction

5. After you configure your batch prediction, click on the green button Predict to generate your batch prediction. (See Figure 3.82.)
6. When the batch prediction is created, you will be able to download the CSV file containing all your dataset instances along with a prediction for each one of them. (See Figure 3.83.)

7. If you didn’t disable the option to create a dataset explained in step 4, you will also be able to access the output dataset from the batch prediction view. (See Figure 3.84.)
3.6.3 Configuring Linear Regression Predictions

BigML provides several options to configure your predictions such as setting default values for your missing numeric values (see Subsection 3.6.3.1), fields mapping (see Subsection 3.6.3.3), and output file settings (see Subsection 3.6.3.4.)

3.6.3.1 Default Numeric Value

By using the Default numeric value before creating your batch prediction, you can easily replace all the missing numeric values by the field’s Mean, Median, Maximum, Minimum or by Zero. (See Figure 3.85.)
3.6.3.2 Excluded Fields

You can specify which field or fields to exclude from the input data when creating your batch prediction. You search the field by typing the name and click on the field found to add to the list of exclusion. (See Figure 3.86)
3.6.3.3 Fields Mapping

You can specify which input fields of the linear regression match with the fields in the dataset containing the instances you want to predict. BigML automatically matches fields by name, but you can also set an automatic match by field ID by clicking on the green switcher. Additionally, you can manually search for fields or remove them from the Dataset fields column if you do not want them to be considered during the batch prediction. (See Figure 3.87.)
Chapter 3. Linear Regressions

3.6.3.4 Output Settings

Batch predictions return a CSV file containing all your instances and the final predictions. Tune the following settings to customize your prediction file (see Figure 3.88):

- **Separator**: this option allows you to choose the best separator for your output file columns. The default separator is the comma. You can also select the semicolon, the tab, or the space.

- **New line**: this option allows you to set the new line character to use as the line break in the generated csv file: “LF”, “CRLF”.

- **Output fields**: by clicking on the list icon next to the separator selector, you can include or exclude all your dataset fields from your output file. You can also individually select the fields you want to include or exclude using the multiple output fields selector. Note: a maximum of 100 fields can be displayed in this selector, but all your dataset fields will be included in the output file by default unless you exclude them.

- **Headers**: this option includes or excludes a first row in the output file (and in the output dataset) with the names of each column (input field names, prediction column name, probability column name, etc.). By default, BigML includes the headers.

- **Prediction column name**: customize the name for your predictions column. By default, BigML takes the name of the linear regression's objective field.

- **Confidence bounds**: this option allows you to include two additional columns with the confidence interval and prediction interval. By default they are not included in your output file.

- **Confidence interval column name**: customize the name for the confidence interval column if you include it in the output file. BigML sets “confidence interval” as the default name.

Note: Fields mapping from the BigML Dashboard is limited to 200 fields. For batch predictions with a higher number of fields, map your fields using the BigML API.

---

8https://bigml.com/api/batchpredictions#bp_batch_prediction_arguments
- **Prediction interval column name**: customize the name for the prediction interval column if you include it in the output file. BigML sets “prediction interval” as the default name.

![Figure 3.88: linear regression output settings for batch predictions](image)

### 3.6.3.5 Prediction explanation

Prediction explanation helps understand why a linear regression makes a certain prediction. This is very useful in many applications, and the reasons behind a prediction are often as important as the prediction itself.

BigML prediction explanation is based on Shapley values. For more information, please refer to this research paper: *A Unified Approach to Interpreting Model Predictions* [3].

When creating single linear regression prediction, you can request the explanation for the prediction by clicking the **prediction explanation** icon and then click **Save** (see Figure 3.89).
Figure 3.89: Prediction explanation

The prediction explanation represents the most important factors considered by the linear regression in a prediction given the input values. Each input value will yield an associated importance, as you can see Figure 3.90. The importances across all input fields should sum 100%.

Figure 3.90: Input field importances

You can export the prediction explanation to a PNG image file, a CSV file or a JSON file by clicking the top right icons respectively.
3.6.4 Consuming Linear Regression Predictions

3.6.4.1 Local Predictions

Local predictions are provided for single instances from the BigML Dashboard which are performed faster at no cost. Local predictions allow you to get a real-time prediction without consuming any credits or requiring any internet connection. This is possible because the linear regression is saved in-memory, so when the input values change, BigML is able to compute predictions in microseconds.

In addition to BigML Dashboard, you can fully use single and batch predictions via the BigML API and bindings. The following subsections explain both tools.

3.6.4.2 Using Linear Regression Predictions via the BigML API

Linear regression predictions have full citizenship in the BigML API which allows you to programmatically create, configure, retrieve, list, update, and delete single and batch predictions.

In the example below, see how to create a single prediction using a linear regression and defining the input data once you have properly set the `BIGML_AUTH` environment variable to contain your authentication credentials:

```
curl "https://bigml.io/prediction?$BIGML_AUTH" \
  -X POST \
  -H 'content-type: application/json' \
  -d '{"linearregression": "linearregression/5c79513a983efc522f000009", "input_data": {"000003":0.61, "000004":1.58, "000005":1.15, "000007":0.55}}'
```

For more information on using linear regressions through the BigML API, please refer to the documentation.

3.6.4.3 Using Linear Regression Predictions via BigML Bindings

You can also create, configure, retrieve, list, update, and delete single and batch predictions via BigML bindings which are libraries aimed to make it easier to use the BigML API from your language of choice. BigML offers bindings in multiple languages including Python, Node.js, Java, Swift and Objective-C. See below an example to create a linear regression with the Python bindings.

```
from bigml.api import BigML
api = BigML()
prediction = api.create_prediction("linearregression/5c702c91983efc4cc6000016", 
  {"age": 230, "cement": 326.81, "blast_furnace_slag": 205.33, "fly_ash":106.17})
```

For more information on BigML bindings, please refer to the bindings page.

3.6.5 Descriptive Information

Each linear regression prediction has an associated name, description, category, and tags. You can find a brief description of each concept in the following subsections. The MORE INFO menu option displays a panel that provides editing options. (See Figure 3.91)
3.6.5.1 Name

If you do not specify a name for your predictions, BigML assigns a default name depending on the type of predictions:

- **Single predictions**: BigML uses the linear regression name “<linear regression name>”.
- **Batch predictions**: BigML combines your prediction dataset name and the linear regression name: “<linear regression name> with <dataset name>”.

Predictions names are displayed on the list and also on the top bar of a prediction view. Predictions names are indexed to be used in searches. Rename your predictions any time from the MORE INFO menu.

The name of a prediction cannot be longer than 256 characters. More than one prediction can have the same name even within the same project, but they will always have different identifiers.

3.6.5.2 Description

Each prediction also has a description that it is very useful for documenting your Machine Learning projects. Predictions take their description from the linear regression used to create them.

Descriptions can be written using plain text and also markdown\(^\text{11}\). BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 3.92.)

\(^{11}\text{https://en.wikipedia.org/wiki/Markdown} \)
3.6.5.3 Category

A category taken from the linear regression used to create it is associated with each prediction. Categories are useful to classify predictions according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

A prediction category must be one of the categories listed on table Table 3.5.

3.6.5.4 Tags

A prediction can also have a number of tags associated with it. These tags help to retrieve the prediction via the BigML API or to provide predictions with some extra information. Your prediction inherits the tags from the linear regression used to create it. Each tag is limited to a maximum of 128 characters. Each prediction can have up to 32 different tags.

3.6.6 Linear Regression Predictions Privacy

The link displayed in the Privacy panel is the private URL of your prediction, so only a user logged into your account is able to see it. Neither single predictions nor batch predictions can be shared by using a secret link. (See Figure 3.93.)
3.6.7 Moving Linear Regression Predictions to Another Project

When you create a prediction, it will be assigned to the same project where the original linear regression is located. You cannot move predictions between projects as you do with other resources.

3.6.8 Stopping Linear Regression Predictions

Single predictions are synchronous resources, so you cannot cancel them during the creation since you get the result immediately.

By contrast, batch predictions are asynchronous resources, so you can stop their creation before the task is finished. Use the DELETE BATCH PREDICTION option from the 1-click action menu (Figure 3.94) or from the pop up menu on the list view.

![Figure 3.94: Stop linear regression batch prediction from 1-click action menu](image)

A modal window will be displayed asking you for confirmation. If you stop the prediction during its creation you won’t be able to resume the same task again, so if you want to create the same prediction you will have to start a new task.

![Figure 3.95: linear regression delete prediction confirmation](image)

3.6.9 Deleting Linear Regression Predictions

You can DELETE your single or batch predictions from the predictions view, using the 1-click action menu (see Figure 3.96) or using the pop up menu on the predictions list view (see Figure 3.97).
A modal window will be displayed asking you for confirmation. Once a prediction is deleted, it is permanently deleted, and there is no way you (or even the IT folks at BigML) can retrieve it.
3.7 Consuming Linear Regressions

Similarly to other models in BigML, linear regressions are white-boxed models, so you can **download** them and use them locally to make predictions. You can also create and consume your linear regressions programmatically via the **BigML API and bindings**. The following subsections explain those three options.

3.7.1 Downloading Linear Regressions

You can download your linear regression in several programming languages including JSON PML, Python or Node.js. By downloading your linear regression you will be able to compute **predictions locally**, free of latency and at no cost. Click on the download icon in the top menu (see Figure 3.99), and select your preferred option (see Figure 3.100).

![Figure 3.99: Click download icon](image)

![Figure 3.100: Select language to download linear regression](image)

3.7.2 Using Linear Regressions Via the BigML API

Linear regression have full citizenship in the BigML API which allows you to programmatically create, configure, retrieve, list, update, delete, and use them for predictions.

In the below example, see how to create a linear regression using an existing dataset once you have properly set the `BIGML_AUTH` environment variable to contain your authentication credentials:

```bash
curl "https://bigml.io/linearregression?$BIGML_AUTH" \
-x POST \
-H "content-type: application/json" \
-d '{"dataset": "dataset/5c7057a9983efc4cc2000005"}''
```
For more information on using linear regressions through the BigML API, please refer to the documentation\textsuperscript{12}.

### 3.7.3 Using Linear Regressions Via the BigML Bindings

You can also create and use linear regressions via BigML bindings which are libraries aimed to make it easier to use the BigML API from your language of choice. BigML offers bindings in multiple languages including Python, Node.js, Java, Swift and Objective-C. See below an example to create a linear regression with the Python bindings.

```python
from bigml.api import BigML
api = BigML()
linearregression = api.create_linear_regression(
    'dataset/5c87d50d983efc4a02000085', {
        "objective_field": "plasma glucose"})
```

For more information on BigML bindings, please refer to the bindings page\textsuperscript{13}.

### 3.8 Linear Regression Limits

There are some limits that apply for the creation of any BigML resource. These are limits based on the number of classes, terms and items that can be considered to create your models. This is explained in Subsection 3.8.0.1.

Additionally, some specific limits apply for your linear regressions visualization, i.e. to the linear regression chart and the coefficient table views, depending on the number of input fields in your dataset. See Subsection 3.8.2 and Subsection 3.8.1 for a detailed explanation.

**Note:** chart limits and coefficient table limits just affect to the visualization of the model, i.e., despite your dataset reach those limits, you can still creating the linear regression, evaluating it and using it to make predictions.

#### 3.8.0.1 Field Limits

Linear regression, similarly to other BigML models, has the following limitations according to the type of field:

- **Classes:** for categorical, a maximum number of 1,000 distinct classes per field is allowed.

- **Terms:** BigML can handle up to 1,000 terms in total. If multiple text fields are defined, then the token limit per field is evenly divided by the number of text fields evenly, e.g., a dataset with two text fields would result in 500 terms per text field. BigML selects those terms with most significant frequency, discarding both those that appear either too often or too infrequently. A maximum of 256 characters per term is allowed.

- **Items:** a maximum number of 10,000 distinct items per field is allowed.

#### 3.8.1 Chart Limits

There are some circumstances under which your chart cannot be displayed:

- As the 1D chart only supports numeric fields for the x-axis, if your linear regression only contains categorical, text, or items fields, the 1D chart cannot be displayed. When you try to click on the 1D chart icon you will see a warning message. (See Figure 3.101)
• If your linear regression contains more than 800 predictors, the prediction interval bounds cannot be displayed in the chart. You will see the warning message shown in Figure 3.102. You can still see your coefficients by downloading the CSV file. For how to calculate the number of predictors in a linear regression, see Subsection 3.2.3.

Figure 3.102: Warning message when the linear regression has more than 800 predictors
3.8.2 Coefficient Table Limits

If your linear regression contains more than 1,000 input fields, the coefficient table cannot be displayed. You will need to download the CSV if you want to see your linear regression coefficients. You will get the message shown in Figure 3.103:

![Figure 3.103: Warning message when the coefficient table limits are reached](image)

3.9 Descriptive Information

Each linear regression has an associated name, description, category, and tags. The following subsections provide a brief description for each concept. In Figure 3.104, you can see the options the MORE INFO menu provides to edit them.

![Figure 3.104: Edit linear regression descriptive information](image)

3.9.1 Linear Regression Name

Each linear regression has a name that is displayed in the linear regression list view and also on the top bar of the linear regression view. Linear regression's names are indexed to be used in searches. When you create a linear regression, it gets a default name. Change it using the MORE INFO menu option on the right corner of the linear regression view. The name of a linear regression cannot be longer than 256 characters.
characters. More than one linear regression can have the same name even within the same project, but they will always have different identifiers.

### 3.9.2 Description

Each linear regression also has a description that it is very useful for documenting your Machine Learning projects. Linear regressions take the description of the datasets used to create them by default. Descriptions can be written using plain text and also markdown\(^\text{14}\). BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 3.105.)

![Markdown editor for linear regression descriptions](image)

Figure 3.105: Markdown editor for linear regression descriptions

Descriptions cannot be longer than 8192 characters and can use almost any character.

### 3.9.3 Category

A category, taken from the dataset used to create it, is associated with each linear regression. Categories are useful to classify linear regressions according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

A linear regression category must be one of the 24 categories listed on Table 3.5.

---

\(^\text{14}\)https://en.wikipedia.org/wiki/Markdown
Table 3.5: Categories used to classify linear regression by BigML

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace and Defense</td>
</tr>
<tr>
<td>Automotive, Engineering and Manufacturing</td>
</tr>
<tr>
<td>Banking and Finance</td>
</tr>
<tr>
<td>Chemical and Pharmaceutical</td>
</tr>
<tr>
<td>Consumer and Retail</td>
</tr>
<tr>
<td>Demographics and Surveys</td>
</tr>
<tr>
<td>Energy, Oil and Gas</td>
</tr>
<tr>
<td>Fraud and Crime</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Higher Education and Scientific Research</td>
</tr>
<tr>
<td>Human Resources and Psychology</td>
</tr>
<tr>
<td>Insurance</td>
</tr>
<tr>
<td>Law and Order</td>
</tr>
<tr>
<td>Media, Marketing and Advertising</td>
</tr>
<tr>
<td>Miscellaneous</td>
</tr>
<tr>
<td>Physical, Earth and Life Sciences</td>
</tr>
<tr>
<td>Professional Services</td>
</tr>
<tr>
<td>Public Sector and Nonprofit</td>
</tr>
<tr>
<td>Sports and Games</td>
</tr>
<tr>
<td>Technology and Communications</td>
</tr>
<tr>
<td>Transportation and Logistics</td>
</tr>
<tr>
<td>Travel and Leisure</td>
</tr>
<tr>
<td>Uncategorized</td>
</tr>
<tr>
<td>Utilities</td>
</tr>
</tbody>
</table>

3.9.4 Tags

A linear regression can also have a number of tags associated with it that can help to retrieve it via the BigML API or to provide linear regressions with some extra information. A linear regression inherits the tags from the dataset used to create it. Each tag is limited to a maximum of 128 characters. Each linear regression can have up to 32 different tags.

3.9.5 Counters

For each linear regression, BigML also stores a number of counters to track the number of other resources that have been created using the linear regression as a starting point. In the linear regression view, you can see a menu option that displays counters for evaluations, single and batch predictions, and the fusions created. It also allows you to quickly jump to all the resources of one type. (See Figure 3.106)

Figure 3.106: Counters for linear regressions
3.10 Linear Regression Privacy

Privacy options for a linear regression can be defined in the More Info panel, displayed in Figure 3.107. There are two levels of privacy for BigML linear regressions:

- **Private**: only accessible by authorized users (the owner and those who have been granted access by him or her).
- **Shared**: accessible by any user with whom the owner shares the secret link.

3.11 Moving Linear Regressions to Another Project

When you create a linear regression, it will be assigned to the same project where the original dataset is located.

Linear regressions can only be assigned to a single project. However, you can move linear regressions between projects. The menu option to do this can be found in two places:

1. From the linear regression list, in either the chart view or table view, click the MOVE TO... option within the 1-click action menu and select another project or create a new one. (See Figure 3.108.)
2. In the linear regression list view, click the **MOVE TO...** option within the **pop up menu** and select another project or create a new one. (See Figure 3.109.)

![Figure 3.109: Change project from pop up menu](image)

### 3.12 Stopping Linear Regressions

You can stop the creation of a linear regression before the task is finished by clicking the **DELETE LINEAR REGRESSION** option from the **1-click action menu** (see Figure 3.110), or from the **pop up menu** in the linear regression list view (see Figure 3.111).

![Figure 3.110: Stop linear regression creation from 1-click action menu](image)
A modal window will be displayed asking you for confirmation. If you stop the linear regression during its creation you won't be able to resume the same task. If you want to create the same linear regression, you will have to start a new task.

### 3.13 Deleting Linear Regressions

You can delete your linear regressions by clickin in the **DELETE LINEAR REGRESSION** option from the **1-click action menu** (see Figure 3.113) or using the **pop up menu** on the linear regression list (see **Figure 3.114**).
A modal window will be displayed asking you for confirmation. Once a linear regression is deleted, it is permanently deleted, and there is no way you (or even the IT folks at BigML) can retrieve it.
3.14 Takeaways

This chapter explains linear regressions in detail. Here is a list of key points:

- A linear regression is a supervised Machine Learning algorithm used to solve regression problems.
- A linear regression is built from a dataset available in BigML and used to make an evaluation, a single prediction, or a batch prediction.
- You can create a linear regression with just one click or configure it as you wish. BigML provides several configuration options before creating your linear regression.
- To create a linear regression you need a dataset containing at least one numeric field.
- Categorical fields must be converted to numeric values in order to train a linear regression model.
- If you do not specify any objective field, BigML will take the last numeric field in your dataset.
- BigML allows you to include your numeric fields' missing values as valid values to train your linear regression model.
- The chart view provides a visual way to analyze a field impact on predictions given certain values for the rest of the fields.
- You get a prediction interval along with the predicted value.
- BigML displays all your linear regression coefficients in a table view which you can also download as a CSV file.
- You need to evaluate your linear regression model's performance using data that the model has not seen before.
- The ultimate goal in building a linear regression is being able to make predictions with it.
- BigML allows you to quickly make predictions for single instances by providing a form containing the fields used by the linear regression, so you can easily set the input data and get an immediate response.
- BigML batch predictions allow you to make simultaneous predictions for multiple instances. All you need is the linear regression you want to use to make predictions and a dataset containing the instances for which you want to obtain predictions.
- You can configure your batch predictions output file settings.
- You can download your linear regression to perform local predictions.
- You can add descriptive information to your linear regressions (name, description, tags, and category).
- You can move your linear regressions between projects.
- You can share your linear regressions with other people using the secret link.
- You can stop your linear regression creation by deleting them.
- You can permanently delete an existing linear regression.
Logistic Regressions

4.1 Introduction

There are multiple Machine Learning problems that require predicting a categorical value, such as “true or false”, “churn or not churn”, “fraud or not fraud”, “high risk, low risk or medium risk”, etc. These are called classification problems, and there can be multiple categories (or classes) to predict.

Logistic regression is a supervised Machine Learning technique that can be used to solve classification problems. These problems can also be solved with other Machine Learning methods, such as models, ensembles or deepnets. We explain these methods in Chapter 1, Chapter 2 and Chapter 5 respectively. The main difference is that logistic regression assumes you input fields can be mapped to predict your objective field following linear patterns. For this reason, logistic regressions work better in those cases for which the problem can be linearly solved.

For each class of the objective field, logistic regression computes a probability modeled as a logistic function value, whose argument is a linear combination of the field values. See Section 4.2 for more details on the logistic regression formula.

This chapter contains comprehensive description of BigML’s logistic regressions including how they can be created with 1-click (Section 4.3), all configuration options available (Section 4.4), and the different visualizations provided by BigML (Section 4.5). See Section 4.6 for an explanation of how logistic regressions can be used to make predictions. You can also export your logistic regressions in different formats to make local predictions faster at no cost (Section 4.6). The process to evaluate your logistic regressions’ predictive performance in BigML is explained in a different chapter (Chapter 7).

In BigML, the third tab of the main menu on the Dashboard allows you to access all of your available supervised models. Select Logistic Regressions from the drop-down menu (Figure 4.1), you will reach the logistic regression list view.

Figure 4.1: Logistic regressions under Supervised tab
Chapter 4. Logistic Regressions

The logistic regression list view (Figure 4.2) lists all your available logistic regressions. For each logistic regression, the view shows the link to the Dataset used to create it, its Name, Objective (objective field name), Age (time elapsed since it was created), Size, and number of evaluations, predictions, and batch predictions that have been created using that logistic regression. The SEARCH menu option in the top right corner of the logistic regression list view allows you to search your logistic regressions by name.

By default, when you first create an account at BigML, or every time that you start a new project, your list of logistic regressions will be empty. (See Figure 4.3.)

Finally, in Figure 4.4 you can see the icon used to represent a logistic regression.

4.2 Understanding Logistic Regressions

As mentioned in the introduction of this chapter, logistic regression is a supervised learning algorithm to solve classification problems. Logistic regression works better in those cases for which the features are roughly linear and the problem can be linearly solved. This is mainly due to the fact that the logistic regression generates linear decision boundaries to separate the objective field classes. You can find a
detailed explanation of this behavior in the following blog post\(^1\).

The reason behind this linear behavior can be found in the logistic regression formula which consist of a logistic function whose argument is a linear combination of the input field values. You can see the logistic regression formula below (Figure 4.5), where dependent variable, \( p_\text{i} \), is the probability for each of the \( i \) classes of the objective field, and the independent variables, \( (X_1, X_2, ... X_k) \) represent the \( k \) variables for the input fields in your dataset, which are multiplied by the logistic regression coefficients \( (b_{0,i}, b_{1,i}, b_{2,i}, ..., b_{k,i}) \).

\[
p_\text{i} = \frac{1}{1 + e^{-f_\text{i}(X)}}
\]

where

\[
f_\text{i}(X) = b_{0,i} + b_{1,i}X_1 + b_{2,i}X_2 + ... + b_{k,i}X_k
\]

Figure 4.5: Logistic regression formula

The logistic regression tries to learn the \( k \) coefficients \( (b_{0,i}, b_{1,i}, b_{2,i}, ..., b_{k,i}) \) of the linear function, \( f_\text{i}(X) \), using maximum likelihood estimation techniques. BigML logistic regression is an optimized implementation of the liblinear library\(^2\) which uses the Trust-Region Newton Optimization method to estimate the coefficients.

Each class of the objective field will have a different set of coefficients associated, e.g., if the objective field has two classes, two different functions, \( p_1 \) and \( p_2 \), will be learned from the training data, one by class (see Figure 4.6).

\[
p_1 = \frac{1}{1 + e^{-(b_{0,1} + b_{1,1}X_1 + b_{2,1}X_2 + ... + b_{k,1}X_k)}}
\]

\[
p_2 = \frac{1}{1 + e^{-(b_{0,2} + b_{1,2}X_1 + b_{2,2}X_2 + ... + b_{k,2}X_k)}}
\]

Figure 4.6: Logistic regression formulas for two classes

A positive coefficient (\( b_k > 0 \)) for a field, indicates a positive correlation with the predicted class, while negative coefficients (\( b_k < 0 \)) indicate a negative relationship. Higher absolute coefficient values for a field results in a greater impact on predictions of that field. This should not be misinterpreted as field importance due to several reasons:

- **Field importance** in Logistic Regression can be defined as the contribution of a field to the final class probability which depends not only on the field coefficient but also on the interactions with the rest of the input fields. Since the model assumes independence between the different inputs, coefficients can be considered as absolute measures of field importance only when all inputs are independent, but this is often not the case. In many real datasets, the impact of a particular field is also dependent on the values of other fields.

- Different field magnitudes make coefficients incomparable. Coefficients for fields with different magnitudes, e.g., salary and age, are not comparable since they will tend to be higher for fields with smaller scales. Changing the scale of the field would change the coefficient value. BigML automatically scales your numeric fields. (See Subsection 4.4.8.)

- Fields can be multi-collinear. If two fields are highly correlated, they can be effectively substituted for each other during model training, so the field importance could be split between the two. The greater number of fields in the dataset, the more likely the fields are multi-collinear.

BigML provides a table containing all your logistic regression coefficients. (See Subsection 4.5.2.)

---

\(^1\)https://blog.bigml.com/2016/09/28/logistic-regression-versus-decision-trees

\(^2\)https://www.csie.ntu.edu.tw/~cjlin/liblinear/
When the logistic regression has learned the coefficients, you can use the model to make predictions for new instances. The logistic regression always returns a probability per class of the objective field. The class with the highest probability will be the predicted class. Taking into account the previous formulas in Figure 4.6, for a given set of input values, \((X_1, X_2, \ldots, X_k)\), you will get two probabilities, one per class, e.g., \(p_1 = 85\%\) and \(p_2 = 15\%\). In BigML, when there are more than two classes, the probabilities are normalized so the sum of all probabilities for each instance prediction is equal to 100%.

By definition, the input fields \((X_1, X_2, \ldots, X_k)\) in the logistic regression formula need to be numeric values. However, BigML logistic regressions can support any type of fields by applying a set of transformations to categorical, text, and items fields. Moreover, BigML can also handle missing values for any type of field. The following subsections detail both behaviors.

### 4.2.1 Input Field Transformations

Apart from numeric fields, BigML logistic regressions are optimized to support categorical, text and items fields by applying a set of transformations in order to convert them in numeric values:

- **Categorical fields** are One-hot encoded by default, i.e., each class is mapped to a separate 0-1 numeric variable. For a given instance, the variable corresponding to the instance class, has its value set to 1, while the other variables are set to 0.

  For example, imagine you are trying to predict the probability of customer \(\text{churn} = [\text{True}, \text{False}]\) given two input fields: number of calls (numeric), \(\text{numCalls}\), and the tariff plan (categorical), \(\text{tariffPlan} = [B, N, P]\), which includes three different classes, \(B\) (basic), \(N\) (normal), \(P\) (professional). The logistic regression will create one variable for the numeric field and another three variables for the categorical field, one by class. Letting \(i\) be the objective field classes (\(\text{True}, \text{False}\)), the logistic regression formula will be:

  \[
  p_i = \frac{1}{1 + e^{-f_i(X)}}
  \]

  where

  \[
  f_i(X) = b_{0,i} + b_{1,i} \text{numCalls} + b_{2,i}B + b_{3,i}N + b_{4,i}P
  \]

  For a new customer with values \(\text{numCalls} = 240\) and \(\text{tariffPlan} = N\) then:

  \[
  f_i(X) = b_{0,i} + b_{1,i}240 + b_{2,i}0 + b_{3,i}1 + b_{4,i}0
  \]

  BigML also provides three other types of coding, **Dummy**, **Contrast** and **Other** coding, that you can configure for each of your categorical fields. See Subsection 4.4.10 for a complete explanation of categorical fields encoding.

- For **text fields**, each term is mapped to a corresponding numeric variable, whose value is the number of occurrences of that term in the instance. Text fields without term analysis enabled are excluded from the model (read the Sources with the BigML Dashboard document to learn more about text analysis [11]).

- For **Items fields**, each different item is mapped to a corresponding numeric field, whose value is the number of occurrences of that item in the instance.

### 4.2.2 Missing Values

BigML logistic regressions can handle missing values for any type of field. For categorical, text, and items fields, missing values are always included as another category, term or item by default.

For numeric fields, missing values are also included by default, but you can deactivate this option by configuring your logistic regression (see Subsection 4.4.5). Alternatively, you can replace your missing numeric values by another valid value like the field’s mean, median, maximum, minimum or zero (see...
If none of the mentioned options has been enabled for building your logistic regression, the instances containing missing values for numeric fields in your dataset will be ignored by the model.

For missing values, a separate variable is created to build the logistic regression. Once the logistic regression is created, you can find an additional coefficient for each field at the end of the coefficient table. (See Figure 4.7.) Learn more about the coefficient table in Subsection 4.5.2.

If the dataset does not contain missing values for a field, the coefficient for missing values will be zero, except in the case of text fields which can be different from zero. This is due to the fact that BigML has a limit of 1,000 terms for text fields, so there may be instances not containing any of the terms considered to build the model and appear as missing values instead. (See Subsection 4.8.0.1 to know more about term limits for text fields.)

![Figure 4.7: Missing numeric coefficients at the end of logistic regression table](image)

### 4.2.3 Logistic Regressions with Images

BigML logistic regressions do not take images as input directly, however, they can use image features as those fields are numeric.

BigML extracts image features at the source level. Image features are sets of numeric fields for each image. They can capture parts or patterns of an image, such as edges, colors and textures. For information about the image features, please refer to section Image Analysis of the Sources with the BigML Dashboard[^11].

[^11]: [https://static.bigml.com/pdf/BigML_Sources.pdf](https://static.bigml.com/pdf/BigML_Sources.pdf)
As shown in Figure 4.8, the example dataset has an image field `image_id`. It also has image features extracted from the images referenced by `image_id`. Image feature fields are hidden by default to reduce clutter. To show them, click on the icon “Click to show image features”, which is next to the “Search by name” box. In Figure 4.9, the example dataset has 512 image feature fields, extracted by a pre-trained CNN, ResNet-18.

From image datasets like this, logistic regressions can be created and configured using the steps described in the following sections. All other operations including prediction, evaluation applies too.
4.3 Creating Logistic Regressions with 1-Click

To create a logistic regression in BigML you have two options: either the 1-click option which uses the default values for all available configuration options, or you can tune the parameters in advanced by using the configuration options explained in Section 4.4. This section explains how to create a logistic regression with 1-click.

From the dataset view, select the 1-CCLICK LOGISTIC REGRESSION option in the 1-click action menu. (See Figure 4.10.)

Alternatively, you can use the 1-CCLICK LOGISTIC REGRESSION option in the pop up menu from the dataset list view. (See Figure 4.11.)

Figure 4.10: Create 1-click logistic regression from dataset 1-click action menu

Figure 4.11: Create 1-click logistic regression from dataset popup menu

Either option builds a logistic regression using the default values for all available configuration options. (See Section 4.4.)

Note: for some datasets, the 1-click option may be disabled. This can be due to the fact that your dataset does not contain any categorical field or the field taken as the default objective field is not categorical. If you do not specify any objective field BigML takes the last valid field in your dataset as the objective field by default. To change the objective field, either configure your logistic regression or select the objective field from your dataset (both options are explained in Subsection 4.4.1.)
4.4 Logistic Regression Configuration Options

While the 1-click creation menu option (see Section 4.3) provides a convenient and easy way to create a BigML logistic regression, you can also have more control over the logistic regression creation and configure a number of parameters that affect the way BigML creates logistic regressions. Click the CONFIGURE LOGISTIC REGRESSION menu option in the configuration menu of your dataset view. (See Figure 4.12.)

![Figure 4.12: Configure logistic regression](image)

Note: for some datasets, the configuration option may be disabled. This can only be due to the fact that the dataset does not contain any categorical field to be selected as the objective field. In case the field to be predicted in your dataset is a numeric field, you can discretize it so it becomes categorical. Read the Datasets documentation [9] to learn how to discretize a numeric field.

4.4.1 Objective Field

The objective field, or “target field”, is the field you want to predict. Logistic regressions only support categorical fields as the objective field.

BigML takes the last categorical field in your dataset as the objective field by default. If you want to change the objective field, you have two options: you can select another field from the configuration panel to build the logistic regression, or you can change it permanently from your dataset view.

- Select the Objective field from the logistic regression configuration panel. This option will only affect the logistic regression you are building that time. (See Figure 4.13.)
• Change the **default objective field** for the dataset. This option will save your objective field preference for any model you build. Click on the edition icon next to the field name when you mouse over it, a pop up window will be displayed. Then click on the **Objective field** icon and **Save** it. (See **Figure 4.14**.)

**Figure 4.13: Configure the objective field to create the logistic regression**

**Figure 4.14: Change the default objective field**

### 4.4.2 Automatic Optimization

You can turn on the **Automatic optimization** option so BigML will automatically tune the parameters of your logistic regression (see **Figure 4.15**).
The high number of possible combinations for parameter values makes it difficult to find the optimum configuration since the combinations that lead to a poor result outnumber the ones that result in a satisfying performance. Hand-tuning different configurations is a time-consuming process that requires a high level of expertise and intuition. To combat this problem, BigML offers first-class support for automatic logistic regression parameter optimization.

Behind the scenes, BigML uses the same technology for logistic regression parameter optimization as the one used for OptiML. If you want to know more about the technical details, please read the Chapter 2 of the document OptiML with the BigML Dashboard [10].

When you turn on the Automatic optimization option, all the logistic regression parameters will be disabled (because they will be automatically optimized), except the Default numeric value, the Missing numerics, and the Weights parameters which you can manually configure (see Figure 4.16).

Since the optimization process can take some time, BigML offers two configurable parameters to limit the time to create the optimized logistic regression: a training duration (see Subsection 4.4.2.1) and the logistic regression candidates (see Subsection 4.4.2.2).
4.4.2.1 Training duration

The scale parameter to regulate the logistic regression runtime. It's set as an integer from 1 to 10. It indicates the user preference for the amount of time they wish the optimization to take. The higher the number, the more time that users are willing to wait for possibly better logistic regression performance. The lower the number, the faster that users wish the logistic regression training to finish. The default value is set to 5.

The training duration is set in a scale. The actual training time depends on the dataset size, among other factors.

4.4.2.2 Logistic regression candidates

The maximum number of different logistic regressions (i.e., logistic regressions using a unique configuration) to be trained and evaluated during the optimization process. The default number is 128 candidates which is usually enough to find the best logistic regression, but you can set it from 4 up to 200. Only the top-performing logistic regression will be returned. If training duration is very low (see Subsection 4.4.2.1) given the dataset size, it is possible that not all the logistic regression candidates will be tried out.
4.4.3 Weights

It is not unusual for a dataset to have some categories that are common and others very rare. For example, in datasets used to predict fraud, usually fraudulent transactions are very scarce compared to regular ones. When this happens, logistic regression tends to predict the most frequent values simply because the overall model's performance metrics improve with that approach. However, in cases such as fraud prediction, you may be more interested in predicting rare values rather than successfully predicting frequent ones. In that case, you may want to assign more weights to the scarce instances so they are equivalent to the abundant ones.

BigML provides three different options to assign specific weights to your instances, balance objective, objective weights, and weight field explained in the following sections (see Figure 4.19).

4.4.3.1 Balance Objective

When you set the balance objective weight (see Figure 2.21), BigML automatically balances the classes of the objective field by assigning a higher weight to the less frequent classes, with the most frequent class always having a weight of 1. For example, take the following frequencies for each class:

- [False, 2000; True, 50]

By enabling the Balance objective option, BigML will automatically apply the following weights:

- [False, 1; True, 40]

In this example, the class “True” is getting forty times more weight as it is forty times less frequent than the most abundant class.

4.4.3.2 Objective Weights

The Objective weights option allows you to manually set a specific weight for each class of the objective field. BigML oversamples your weighted instances replicating them as many times as the weight establishes. If you do not list a class, it is assumed to have a weight of 1. Weights of 0 are also valid, but if all classes have a weight of 0, the logistic regression creation will produce an error.

This option can be combined with the Weight field (see Subsection 4.4.3.3). When combining it with the Weight field, both weights are multiplied. For example if you assign a weight of 3 for the “True” class and the weight field assigns a weight of 2 for a given instance labeled as “True”, that instance will have a total weight of 6.
4.4.3.3 Weight Field

The **Weight field** option allows you to assign individual weights to each instance by choosing a special weight field. The selected field must be numeric and it must not contain any negative or missing values. The weight field will be excluded from the input fields when building the logistic regression. You can select an existing field in your dataset or you may create a new one in order to assign customized weights.

The weights of your weight field will impact in the same way as the **Objective weights**. If an instance has a weight of 3 it will be replicated three times in the dataset to train the model.

4.4.4 Default Numeric Value

Logistic regressions can include missing values as valid values for any type of fields as explained in Subsection 4.2.2. However, there can be situations for which you don’t want to include them in your model. For those cases, the **Default numeric value** parameter is an easy way to replace missing numeric values by another valid value. You can select to replace them by the field’s **Mean**, **Median**, **Maximum**, **Minimum** or by **Zero**. (See Figure 4.20.)

![Select a default numeric value to replace missing numeric values](image)

**Figure 4.20:** Select a default numeric value to replace missing numeric values

*Note: if your dataset does not contain missing values for your numeric fields, this parameter will not have impact on your logistic regression. By contrast, if your dataset contains missing numeric values and you neither select a Default numeric value or enable the Missing numerics configuration option, instances with missing numeric values will be ignored to build the model. (See next Subsection 4.4.5.)*

4.4.5 Missing Numerics

By default, missing values for your numeric fields are included as valid values to build your logistic regression. However, as explained in the previous subsection, there can be cases for which you don’t want them to be included in your model. The **Missing numerics** option allows you to select if you want to include or exclude the missing numeric values to build your logistic regression. (See Figure 4.21.)
Chapter 4. Logistic Regressions

Figure 4.21: Include missing numeric values in your logistic regression

Note: missing values are always included for categorical, text, and items fields. (See Section 4.2.) As explained in Subsection 4.2.2, by including missing numeric values, you will get an additional coefficient for your numeric fields denoting the missing values. You will find them at the end of the coefficients table as you can see in Figure 4.22. If a numeric field has no missing values, those coefficients will be zero.

Figure 4.22: Missing numeric coefficients at the end of logistic regression table
**Missing numerics** and **Default numeric value** parameters are substitutes. (See Subsection 4.4.4.) Therefore, if you enable the default numeric value parameter, the **Missing numerics** icon will automatically be disabled. (See Figure 4.23.)

![Figure 4.23: Missing numerics is disabled when there is a Default numeric value selected](image1)

**Note:** If your dataset contains missing numeric values and you do not either enable **Missing numerics** or select a Default numeric value, instances containing missing values will be ignored when building the logistic regression.

### 4.4.6 Eps

This parameter sets the **stopping criteria** for the solver. If the difference between the current results and the last iteration results is smaller than **Eps**, then the solver is finished. You can set positive float values greater than 0 and smaller than 1. The default value is 0.0001. Higher values make the model to be faster built but they may result in a worse predictive performance.

![Figure 4.24: Eps parameter for logistic regression](image2)
4.4.7 Stats

This parameter allows you to include or exclude the **statistical tests** to assess the quality of the model's fit to the data. (See **Figure 4.25**.) BigML does not include these statistics by default to speed up the model creation because the **time required** to compute them can be considerably **high**.

When **stats** are enabled, BigML automatically configures your categorical fields with **dummy coding**. BigML selects the dummy class in alphabetical order, therefore for fields with classes such as “0”/“1”, “yes”/“no” or “true”/“false”, the dummy class will always be “0”, “no” or “false”. You can see the field codings configuration by clicking in the wheel next to your categorical fields. (See **Figure 4.26**.) This automatic configuration is to avoid multicollinearity in your model, otherwise all the statistics computed will be null. You can configure the categorical fields with any of the other field codings, **contrast** or **other**, but you cannot configure one-hot coding again if **stats** are enabled. See **Subsection 4.4.10.1** to read more about field codings.
Figure 4.26: Check your categorical fields encoding by clicking in the green wheel icon

BigML computes two kinds of statistical tests, a test that measures the quality of the coefficients as a whole, and a set of tests that measure the significance of the individual coefficient estimates:

- **Statistical test to measure the quality of the whole model:**
  - **Likelihood ratio**: tests whether the coefficients as a whole have any predictive power to predict a certain class of the objective field over an intercept-only model.

- **Statistical tests to measure the significance of the individual coefficient estimates:**
  - **Standard error**: the variance of the coefficient estimates.
  - **Z scores**: how many standard deviations are the coefficient estimates from the mean. Letting $\beta_i$ be the coefficient estimate and $SE$ the standard error, then the formula for each $i$ coefficient is:
    $$Z \text{ score}_i = \frac{\beta_i}{SE(\beta_i)}$$
  - **p-value**: determines the statistical significance of each coefficient in the logistic regression formula. The p-value is associated with the $\chi^2$ of the Wald test with one degree of freedom at the $Z^2$ value. The Wald test indicates how far the estimated parameters are from zero to test their significance.

The p-value needs to be compared against a significance level, usually 0.1, 0.05 or 0.01. A p-value higher than the significance level, indicates that the null hypothesis can be accepted, hence the coefficient is non-significant. A p-value lower than the significance level indicates...
that the null hypothesis can be rejected, so the coefficient estimate is significant. A good practice is to retrain the logistic regression removing the non-significant coefficients.

**Note:** note that *p*-values are not extracted directly from Z score values, they are associated with the $\chi^2$ of the Wald test.

- **Confidence intervals:** give the size of the 95% confidence interval for each coefficient estimate. That is, for a coefficient estimate $\beta$, and an interval value $n$, the value of the coefficient is $\beta \pm n$ with a confidence of 95%.

Once your logistic regression has been created, you will be able to visualize the stats in the coefficient table. (See Subsection 4.5.2.1.)

To avoid lengthy computation times, stats from large input datasets will be computed from a subsample of the dataset such that the number of coefficients $X$ rows is less than or equal to $1E+8$.

It is possible that some statistic values contain null values. Wald test statistics cannot be computed for zero-value coefficients, and so their corresponding entries are null. Moreover, if the coefficients’ information matrix is ill-conditioned, e.g. if there are fewer instances of the positive class than the number of coefficients, or if there is a perfect correlation between the input fields (multicollinearity), the standard error, Z score, *p*-value, and confidence intervals will also have a null value.

### 4.4.8 Bias and Auto-Scaling

You can include or exclude the **Bias** from the model, a.k.a. the intercept term of the logistic regression formula. (See formula in Section 4.2.) For most cases, including the bias results in a better model. By default it is included. (See Figure 4.27.)

You can also scale the **numeric fields** of your dataset, to ensure each field will have equivalent influence despite differences in magnitudes, e.g., salary and age. By enabling the **Auto-scale** parameter, BigML automatically transforms your numeric fields so their standard deviations equal 1. Auto-scaling fields will allow you to compare the different coefficients learned by the model as explained in Section 4.2. Fields are auto-scaled by default. (See Figure 4.27.)

![Figure 4.27: Bias parameter and auto-scaling parameter for numeric fields](image-url)
4.4.9 Regularization

The main goal for having a regularization factor is to avoid overfitting, i.e., tailoring the model to the training data at the expense of generalization.

In BigML you can choose between **L1 or L2 Regularization**. L1 norm causes more coefficients to be zero, while using the L2 norm forces the values of all coefficients towards zero. Usually L2 yields better results, so it is the default option to create your logistic regression.

You can also tune the **Strength (c)** parameter, which is the inverse of the regularization strength, so higher values indicate less regularization. It must be a positive integer greater than 0. Too high values for strength will make the algorithm perfectly fit the training data boundaries, so the logistic regression will perform poorly when trying to predict new instances not seen before by the model. Too low values for strength will result in vague decision boundaries not following the data patterns, hence resulting also in a bad predictive performance. The default value is 1 which usually works well for most cases. (See Figure 4.28.)

![Figure 4.28: Regularization parameters](image)

4.4.10 Field Codings

**Categorical fields** must be converted to numeric values in order to train a logistic regression model. By default, they are **One-hot** encoded, but BigML allows you to configure three other types of coding for each one of your categorical fields: **Dummy coding**, **Contrast coding**, and **Other coding**. See the following subsections for a detail explanation of each option. (Learn more about input fields transformations in Subsection 4.2.1.)

**Note:** if stats are enabled, then the default coding is Dummy for all your categorical fields, selecting the dummy class by alphabetical order. See Subsection 4.4.7 for more details.

4.4.10.1 One-hot Coding

Categorical fields are **One-hot** encoded by default. That is, a separate 0-1 numeric field is created for each category and an additional one for missing values. For a given instance, the variable corresponding to the instance’s categorical value has its value set to 1, while the other variables are set to 0. See an example of One-hot coding scheme for a field containing three classes in Table 4.1:
### 4.4.10.2 Dummy Coding

The main goal of using dummy coding is to compare a class selected as the reference or control class with the rest of classes. The control class is assigned a value of 0 for each variable. The control class is called dummy class in BigML and it is usually a class with a representative number of instances compared to the other classes in the dataset. See an example of dummy coding schema for three different classes, being the “Class 1” the dummy class, in Table 4.2:

<table>
<thead>
<tr>
<th>Classes</th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Class 2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Class 3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MISSING</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.2: Dummy coding example for 3 classes

To set **dummy coding** for a field:
1. Click on the configuration icon next to the field name. (See Figure 4.29.)

---

5[https://en.wikipedia.org/wiki/Categorical_variable#Dummy_coding](https://en.wikipedia.org/wiki/Categorical_variable#Dummy_coding)
2. A modal window will be displayed so you can configure the field codings for that field. If the field has not a previous configuration for field codings, it will be disable. **Enable** field coding configuration by clicking on the green switcher shown in Figure 4.30.
3. Select the class you want to set as the dummy class. (See Figure 4.31.)

4. Click ‘Save’. Make sure you saved your configuration by looking at the bottom message “Configured Coding: DUMMY”. (See Figure 4.32.)
Figure 4.32: Field codings: dummy

**Note:** you cannot select several field codings for the same field simultaneously.

5. Close the modal window by clicking outside or by clicking **Cancel**.

Figure 4.33: Close modal window

**Note:** if the **Cancel** button is red, it indicates there are changes you have not saved yet so you will lose them by closing the modal window.

6. After configuring the field codings for a field, the configuration icon will become green. (See Figure 4.34.)
7. To remove the field coding configuration for that field, click **Disable** from the switcher and click **Save** again. (See Figure 4.35.)

Figure 4.34: Field codings configured

Figure 4.35: Disable field coding configuration
After creating your logistic regression, your dummy class will be identified with the dummy icon in the **coefficients table** view (see Subsection 4.5.2). (See Figure 4.36.)

![Figure 4.36: Dummy class in table view](image)

**4.4.10.3 Contrast Coding**

Contrast coding allows you to set different values for different classes. Instead of the 0-1 values of One-hot coding, you will be able to set any integer or float value for each of the classes, plus an additional one for missing values. The sum of all values must equal 0. The values of the classes need to be set based on certain hypothesis, e.g., higher values for a class assume this class has more influence on the objective field than the others. A positive value indicates a positive relationship between the class and the objective field while a negative value indicates a negative relationship. A coefficient of 0 will exclude the class from the model. In the Table 4.3 you can see an example of contrast coding schema for three different classes.

<table>
<thead>
<tr>
<th>Classes</th>
<th>C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>0.5</td>
</tr>
<tr>
<td>Class 2</td>
<td>-0.25</td>
</tr>
<tr>
<td>Class 3</td>
<td>-0.25</td>
</tr>
<tr>
<td>MISSING</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.3: Contrast coding example for 3 classes

To set **Contrast coding** for a field, follow these steps:

6[https://en.wikipedia.org/wiki/Categorical_variable#Contrast_coding](https://en.wikipedia.org/wiki/Categorical_variable#Contrast_coding)
1. Click on the configuration icon next to the field name. (See Figure 4.37.)

![Field coding configuration](image)

**Figure 4.37: Field coding configuration**

2. A modal window will be displayed so you can configure the field codings for that field. If the field has not a previous configuration for field codings, it will be disable. **Enable field coding configuration** by clicking on the green switcher shown in Figure 4.38.
3. Select the **Contrast coding** option. (See Figure 4.39.)

4. Set the values you want for your classes based on your hypothesis. All classes values must sum to 0. (See Figure 4.40.) By using the BigML API, multiple contrast codings can be given for a field as long as all the codings are orthogonal to ensure there are no co-dependent coefficients. Check the corresponding documentation\(^7\).

\(^7\)https://bigml.com/api/logisticregressions#lr_coding_categorical_fields
Note: you cannot select several field codings for the same field simultaneously.

5. Click **Save**. Make sure you saved your configuration by looking at the bottom message “Configured Coding: CONTRAST”. (See Figure 4.41.)
Note: if the "Cancel" button is red, it indicates there are changes you have not saved yet so you will lose them by closing the modal window.

7. After configuring the field codings for a field, the configuration icon will become green. (See Figure 4.43.)
8. To remove the field coding configuration for that field, click \textit{Disable} from the switcher and click \textit{Save} again. (See Figure 4.44.)

![Figure 4.44: Disable field coding configuration](image)

After creating your logistic regression, you will be able to see your \textit{Contrast coding} values in the \textit{coefficients table} view (see Subsection 4.5.2) by clicking on the icon. (See Figure 4.45.)

![Figure 4.45: Contrast icon in table view](image)

A modal window will be displayed with your codings values and you can download them in CSV or JSON format by clicking on the corresponding icons. (See Figure 4.45.)
4.4.10.4 Other Coding

Other coding\(^8\) allows you to set different values for different classes. It works the same way as contrast coding (see Subsection 4.4.10.3), but in this case the values do not need to sum 0. In the Table 4.4 you can see an example of other coding schema for three different classes.

<table>
<thead>
<tr>
<th>Classes</th>
<th>C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>2</td>
</tr>
<tr>
<td>Class 2</td>
<td>-0.4</td>
</tr>
<tr>
<td>Class 3</td>
<td>3</td>
</tr>
<tr>
<td>MISSING</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.4: Other coding

To set Other coding for a field, follow these steps:

1. Click on the configuration icon next to the field name. (See Figure 4.47.)

\(^8\)https://en.wikipedia.org/wiki/Categorical_variable#Contrast_coding
2. A modal window will be displayed so you can configure the field codings for that field. If the field has not a previous configuration for field codings, it will be disable. **Enable** field coding configuration by clicking on the green switcher shown in Figure 4.48.
3. Select the **Other coding** option. (See Figure 4.39.)

4. Set the values you want for your classes based on your hypothesis. You can set any float or integer value. (See Figure 4.50.) By using the BigML API, multiple other codings can be given for a field. Check the corresponding documentation\(^9\).

\(^9\)https://bigml.com/api/logisticregressions#lr_coding_categorical_fields
Figure 4.50: Set the other coding values for each class

Note: you cannot select several field codings for the same field simultaneously.

5. Click **Save**. Make sure you saved your configuration by looking at the bottom message “Configured Coding: OTHER”. (See Figure 4.51.)

Figure 4.51: Other coding saved

6. Close the modal window by clicking outside or by clicking **Cancel**.
Figure 4.52: Close modal window

Note: if the Cancel button is red, it indicates there are changes you have not saved yet so you will lose them by closing the modal window.

7. After configuring the field codings for a field, the configuration icon will become green. (See Figure 4.53.)

Figure 4.53: Field codings configured
8. To remove the field coding configuration for that field, click Disable from the switcher and click Save again. (See Figure 4.54.)

Figure 4.54: Disable field coding configuration

After creating your logistic regression, you will be able to see your Other coding values in the coefficients table view (see Subsection 4.5.2) by clicking on the icon. (See Figure 4.55.)

Figure 4.55: Other coding in coefficients table

A modal window will be displayed with your coding values and you can download them in CSV or JSON format by clicking on the corresponding icons. (See Figure 4.55.)
4.4.11 Sampling Options

Sometimes you do not need all the data contained in your dataset to build your logistic regression. If you have a very large dataset, sampling may be a good way of getting faster results. BigML allows you to sample your dataset before creating the logistic regression, so you do not need to create a separate dataset first. You can find a detailed explanation of the sampling parameters available in the following subsections. (See Figure 4.57.)

4.4.11.1 Rate

The Rate is the proportion of instances to include in your sample. Set any value between 0% and 100%. Defaults to 100%.

4.4.11.2 Range

Specifies a subset of instances from which to sample, e.g., choose from instance 1 until 200. The Rate you set will be computed over the Range configured. This option may be useful when you have temporal data, and you want to train your logistic regression with historical data, and test it with the most recent one to check if it can predict based on time.

4.4.11.3 Sampling

By default, BigML selects your instances for the sample by using a random number generator, which means two samples from the same dataset will likely be different even when using the same rates and row ranges. If you choose deterministic sampling, the random-number generator will always use the same seed, thus producing repeatable results. This lets you work with identical samples from the same dataset.

4.4.11.4 Replacement

Sampling with replacement allows a single instance to be selected multiple times. Sampling without replacement ensures that each instance cannot be selected more than once. By default, BigML generates samples without replacement.

4.4.11.5 Out of bag

This argument will create a sample containing only out-of-bag instances for the currently defined rate, so the final total number of instances for your sample will be one minus the rate configured for your sample (when replacement is false). This can be useful for splitting a dataset into training and testing subsets. It is only selectable when a sample rate is less than 100%.
4.4.12 Creating Logistic Regressions with Configured Options

After finishing the configuration of your options, you can change the default logistic regression name in the editable text box. Then you can click on the Create logistic regression button to create the new logistic regression, or reset the configuration by clicking on the Reset button.
4.4.13 API Request Preview

The API Request Preview button is in the middle on the bottom of the configuration panel, next to the Reset button (See Figure 4.58). This is to show how to create the logistic regression programmatically: the endpoint of the REST API call and the JSON that specifies the arguments configured in the panel. Please see Figure 4.59 below:

![API Request Preview](image)

Figure 4.59: Logistic regression API request preview

There are options on the upper right to either export the JSON or copy it to clipboard. On the bottom there is a link to the API documentation for logistic regressions, in case you need to check any of the possible values or want to extend your knowledge in the use of the API to automate your workflows.

Please note: when a default value for an argument is used in the configuration, the argument won’t appear in the generated JSON. Because during API calls, default values are used when arguments are missing, there is no need to send them in the creation request.

4.5 Visualizing Logistic Regressions

After creating your logistic regression, you will be able to analyze your results with BigML unique visualization: a 1D and 2D chart, to examine the impact of the input fields in the objective field, and a coefficients table, for more advanced users, to interpret the resulting coefficients for each input field. The following subsections explain both visualizations in detail.

Switch from the chart view to the table view by clicking on icons in the top bar menu of the logistic regression view. (See Figure 4.60.)
4.5.1 Logistic Regression Chart

The chart view is composed of three main parts: the CHART itself, which you can view in one dimension (1D) or two dimensions (2D), the PREDICTION legend and the INPUT FIELDS form. (See Figure 4.61.) You can find a detailed explanation of each one below.

- The CHART allows you to view the impact of the input fields on the objective classes predictions. You can select the 1D chart or 2D chart by clicking on the green switcher in the top bar menu.

The **1D chart**, allows you to select one input field for the x-axis. The y-axis represents the probabilities for each of the predicted classes. Only numeric fields can be selected for the x-axis.(See chart limitations in Subsection 4.8.1.) You can extend the upper limit of the x-axis by clicking on the plus icon.
The **2D chart** allows you to select two different input fields for both axis and the class probabilities are represented in different colors in a heat map. You can select numeric or categorical fields for the axis. You can switch the axis by clicking on the option on top of the chart area.

In both charts you can inspect the axis values in the grey area next to the selector. You can freeze the view by pressing `Shift` and release it again by pressing `Escape` from your keyboard. When
the view is frozen, an edition icon will appear you can edit the axis values and obtain a prediction for another preferred value. The resulting predicted probabilities are in the prediction legend to the right.

To know more about how to interpret the influence of the input fields on predictions, read this post¹⁰.

• The PREDICTION legend allows you to visualize the classes represented in the chart along with their corresponding colors. The main probability color bar at the top is the probability for the predicted class. By default, in the 2D chart, colors are shaded according to the range of probabilities shown in the chart area. That way, smaller differences in predictions are easier to perceive. However, you can choose to see the color shading according to the total range of class probabilities (from 0% to 100%) by clicking on the icon next to the probability bar  Total . (See Figure 4.64.) You can also select to see only one of the classes using the class selector int he bottom of the legend.

![Figure 4.64: Prediction legend](https://blog.bigml.com/2016/09/26/predicting-airbnb-prices-with-logistic-regression/)

Freeze this view by pressing  Shift , and release it again by pressing  Escape  from your keyboard.

• Below the chart legend, you can find the INPUT FIELDS form. (See Figure 4.65.) You can configure the values for any numeric, categorical, text or items field. By changing their values, you can see the class probabilities changing in real-time. You can also select or disable your input fields, so they will be treated as missing values. If you configured your model to deactivate the Missing numerics option, you will not be able to disable your numeric fields. (See Subsection 4.4.5.)

Moreover, the chart includes a reset option for your input fields values, and an export option to download your chart in PNG format explained below:

- After selecting the fields for the axis or configuring the input fields values, you can set them again to the default view by clicking the reset icon highlighted in Figure 4.66.

- You can also export your chart in PNG format with or without legends. Freeze the view by pressing
Shift from your keyboard and export the chart to get the classes percentages in the legend. Release the view by pressing Escape.

![Figure 4.67: Export chart as image with or without legends](image)

Note: there are some limitations in the number of classes of the objective field and the number of input fields to visualize your logistic regression in the chart (explained in Subsection 4.8.1).

### 4.5.2 Coefficient Table

The main goal of the logistic regression algorithm is to learn the coefficients of the logistic function for each of the dependent variables, i.e., for each of the input fields. A different set of coefficients is associated with each class of the objective field. See Section 4.2 for a detailed explanation of logistic regression coefficients interpretation.

BigML allows you to inspect the learned coefficients for each one of the input fields in the coefficient table. The table columns represent the objective field classes while the table rows represent the input field variables and the bias (a.k.a. intercept term) of the logistic regression. In the first row you will always find the Bias coefficients. If your objective field has a high number of classes, you may need to scroll horizontally to see all of them. You can order the table rows by clicking on any of the columns labels.
For numeric fields, there is always one coefficient by field, however categorical, text and items fields have one coefficient by value (category, term or item). This is due to the required transformations, explained in Section 4.2, to convert categorical, text, and items fields in numeric fields (each single value is mapped to a separate variable in the formula). Missing values also get their own coefficients.

- For **numeric** fields, you always get one coefficient per field. If you train the logistic regression with **Missing numerics**, you will find an additional coefficient per field for the missing values. (See Subsection 4.4.5.)
- For **categorical** fields, you have one coefficient per class per field and an additional one for missing values per field. (See Subsection 4.4.10.) **Note:** if you configure the field with **Contrast** or **Other coding** there will be just one coefficient for that field (see Subsection 4.4.10). 
- For **text** fields, there is one coefficient per term and an additional one for missing values per field.
- For **items** fields, you get one coefficient per item and an additional one for missing values per field.

See an example of coefficients for a categorical field in Figure 4.69 where one single field, “Atmospheric condition”, yields twelve different variables associated with different coefficients, one per each one of the field’s classes.
Figure 4.69: Multiple field variables for categorical one-hot encoded fields

Coefficients for **missing values** are always found at the end of the table. (See Figure 4.70.) For fields without missing values in the original dataset, those coefficients should be zero (see Subsection 4.2.2.)
Figure 4.70: Missing numeric coefficients at the end of logistic regression table

Additional options for the table include a filtering option and an export option:

- You can filter the table first column by field name, class, term or item using the search box at the top of the table (see Figure 4.71.)
Figure 4.71: Search and filter logistic regression table

- You can also **export** the table in a CSV file by clicking on the icon highlighted in Figure 4.72.
4.5.2.1 Coefficient Table with Stats Computation

When stats are enabled, BigML displays them in the coefficient table. See a detailed explanation of how to include the stats in your model in Subsection 4.4.7.

In the resulting coefficient table you will find three new elements: a new row at the top of the table containing the likelihood ratio, an icon per coefficient indicating its significance and a summary of the stats per coefficient. You can find a detailed explanation below:

- The likelihood ratio tests if the coefficients as a whole have any predictive power. It is the difference in the log likelihood between the fitted model and an intercept-only model. You will find it in the first row of the table. (See Figure 4.73.)
If this difference is significant, i.e., the $p$-value is lower than the significance level, then the coefficients as a whole have more predictive power than an intercept-only model. The icon shown in Figure 4.74 indicates if the likelihood ratio is significant or not given the selected significance level.

You can select your preferred significance level by using the selector shown in Figure 4.75.
You can see the \textit{p-value} associated to the likelihood ratio by mousing over the sigma icon. (See Figure 4.76.)

• Next to each coefficient you will find one icon indicating if it is significant (see Figure 4.77) or non-significant (see Figure 4.78).
The significance of a coefficient is determined by comparing the \( p \)-value against the significance level selected in the top menu (see Figure 4.75). If the \( p \)-value is higher than the significance level, the coefficient will be non-significant. If the \( p \)-value is lower than the significance level, the coefficient will be significant. A good practice is to retrain the logistic regression removing the non-significant coefficients. However, in most cases, the model performance should not be affected.

\[\text{\( p \)-value from the Wald test}^{11}\text{. As mentioned in the previous point, this \( p \)-value is compared against the selected significance level to determine the coefficient's significance. Then, associated with the Z score chart, you will find the Z score value, the confidence interval for a 95% confidence and the standard error, or variance, of the coefficient estimate.}

\[\text{\url{https://en.wikipedia.org/wiki/Wald_test}}\]
Chapter 4. Logistic Regressions

4.6 Logistic Regression Predictions

4.6.1 Introduction

The ultimate goal in building a logistic regression is being able to make predictions with it. In BigML, you can make predictions for single instances or for many instances in batch. Each prediction comes with a measure indicating the probability of the predicted class, a percentage ranging from 0% up to 100%. For each prediction, BigML also provides the probabilities for the rest of classes in the objective field.

The predictions tab in the main menu of the BigML Dashboard is where all your saved predictions are listed. (See Figure 4.81.) You can search your predictions by name clicking on the search option on the top menu. In the predictions list view, you can see, for each prediction, the logistic regression icon used for the prediction, the Name of the prediction, the Objective (objective field name), the Prediction (the prediction result), and the Age (time since the prediction was created).
When you first create an account at BigML, or every time that you start a new project, your list of predictions will be empty. (See Figure 4.82.)

![Figure 4.82: Empty predictions list view](image)

Logistic regression predictions are saved under the CLASSIFICATION & REGRESSION option in the menu (see Figure 4.83.)

![Figure 4.83: Menu options of the predictions list view](image)

Select the list for your single instances predictions or your batch predictions by clicking on the corresponding icons. (See Figure 4.84 and Figure 4.85.)

![Figure 4.84: Single predictions icon](image)

![Figure 4.85: Batch predictions icon](image)

### 4.6.2 Creating Logistic Regression Predictions

BigML provides two options to predict with your logistic regressions explained in the following subsections:

- **PREDICT**: to predict one single instance
- **BATCH PREDICTION** to predict multiple instances in batch.
4.6.2.1 Predict

BigML allows you to quickly make predictions for single instances by providing a form containing the input fields used by the logistic regression, so you can easily set the values and get an immediate response.

Follow these steps to create a single prediction:

1. Click **PREDICT** in the logistic regression **1-click action menu**. (See Figure 4.86.)

![Figure 4.86: Predict using the 1-click action menu](image)

Alternatively, click **PREDICT** in the **pop up menu** in the list view. (See Figure 4.87.)

![Figure 4.87: Predict using the pop up menu](image)

2. You will be redirected to the **prediction form** where you will find all the fields used by the logistic regression as input fields. (See Figure 4.88.)

![Figure 4.88](image)
3. **Select the fields** to be used for your prediction (Figure 4.89.) Non-selected fields will be considered as missing values during the prediction.
If your logistic regression was not trained with **Missing numerics** (see Subsection 4.4.5) you won’t be able to disable your numeric fields for the prediction and a warning message will appear instead: “This field cannot be disabled because your model has been trained without missing numerics”.

4. **Set input values** for your selected fields. BigML supports numeric, categorical, text and items fields as inputs.

5. **Get the prediction** at the top of the view along with the predicted class probability. (See Figure 4.90.) BigML predictions are synchronous, i.e., when you send the input data, you get an immediate response. Moreover, single predictions from the BigML Dashboard are performed locally, so unless you save your prediction, it will not consume any credits and it will be updated instantly when you change your input values. Learn more about **local predictions** in Subsection 4.6.4.1.1.
6. Below the prediction you can see a histogram view containing the rest of your **class probabilities** (Figure 4.91). You can download all the probabilities in PNG format, in CSV or JSON file by clicking on the corresponding icons. (See Subsection 4.6.4.1.)
7. Optionally, you can save the logistic regression prediction, so you will find it afterwards in the predictions list view. (See Figure 4.92.)
Figure 4.92: Save your logistic regression predictions

Note: this option is only available from the BigML Dashboard for logistic regressions with less than 100 fields. If you want to perform single instance predictions for a higher number of fields, use the BigML API.\(^{12}\)

4.6.2.1.1 Logistic Regression Prediction with Images

BigML logistic regressions can be trained from images using extracted image features (Subsection 4.2.3). Because image features are automatically generated numeric fields, creating logistic regression predictions with images is the same as creating other logistic regressions. The only thing different is input fields of images.

Note: When the input fields contain images, in order to create the single prediction, BigML will extract image features automatically to match what were used in the dataset to train the logistic regression.

\(^{12}\)https://bigml.com/api/predictions
The logistic regression in Figure 4.93, “grape-strawberry resnet18”, was created from a dataset containing image features extracted from a pre-trained CNN, ResNet-18. Creating a prediction using the logistic regression will be directed to the prediction form which presents all input fields used by the logistic regression. One of them is the image field. Because this is a single prediction, an image is input by using a single image source. Clicking on the input field box, single image sources available will be in the dropdown list. There is also a search box which can be used to locate specific ones.

Figure 4.93: Select a single image source in the image input field
Oftentimes single image sources were used for creating a composite source, they become component sources of the composite source. Or an image was uploaded as a part of an archive file (zip/tar) which created a composite source. In those cases, the composite source will be shown in the dropdown list, along with an icon “List components”. In the example in Figure 4.94, predict-images.zip is a composite source, click on the icon to show its component sources.
Figure 4.95: Select a component of a composite source

After the component sources of the composite are listed, scroll the dropdown list to find the desired one, then click to select it, as shown in Figure 4.95. There is also a search box to locate specific component sources.
In addition to images, logistic regressions may use other fields, which will be in the prediction form too. As shown in Figure 4.96, all the fields can be selected, and their input values be set by dragging the knobs on the sliders or by entering precise values in their input boxes.

Once all fields are selected, click on the green button \textit{Predict} to create a prediction.
After a new prediction is created, as shown in Figure 4.97, the predicted class is at the top of the form along with its probability. The prediction interface is the same as ones created by non-image logistic regression. Everything described earlier in this section (Subsection 4.6.2.1) applies.

### 4.6.2.2 Batch Predictions

BigML batch predictions allow you to make simultaneous predictions for multiple instances. All you need is the logistic regression you want to use to make predictions and a dataset containing the instances you want to predict. BigML will create a prediction for each instance in the dataset.

Follow these steps to create a batch prediction:

1. **Click on** BATCH PREDICTION option under the logistic regression 1-click action menu (Figure 4.98)

   ![Figure 4.98: Create batch prediction using 1-click action menu](image)

   Alternatively, click on CREATE BATCH PREDICTION in the **pop up menu** of the list view (Figure 4.99).
2. **Select the dataset** containing all the instances you want to predict. The instances should contain the input values for the fields used by the logistic regression as input fields. From this view you can also select another logistic regression from the selector or even a model or ensemble by clicking on the icons on the top left menu. (See Figure 4.100.)

3. After the logistic regression and the dataset are selected, the batch prediction **configuration options** will appear along with a **preview** of the prediction output (a CSV file). (See Figure 4.101.) The default output format includes all your prediction dataset fields and adds an extra column with the class predicted. See Subsection 4.6.3 for a detailed explanation of all configuration options.
Figure 4.101: Configuration options for logistic regression batch prediction

4. By default, BigML generates an output dataset with your batch predictions that you can later find in your datasets section in the BigML Dashboard. This option is active by default but you can deactivate it by clicking on the icon shown in Figure 4.102.
5. After you configure your batch prediction, click on the green button [Predict] to generate your batch prediction. (See Figure 4.103.)
6. When the batch prediction is created, you will be able to **download the CSV file** containing all your dataset instances along with a prediction for each one of them. (See Figure 4.104.)

7. If you didn’t disable the option to create a dataset explained in step 4, you will also be able to access the **output dataset** from the batch prediction view. (See Figure 4.105.)
Chapter 4. Logistic Regressions 315

4.6.2.2.1 Batch Prediction with Images

BigML logistic regression can be trained from images using extracted image features (Subsection 4.2.3). The input of a batch prediction is a dataset. So when creating a batch prediction with images, the dataset has to have the same image features used to train the logistic regression. The image features are in the dataset used to create the logistic regression.

As shown in Figure 4.106, the input for the logistic regression batch prediction is selected as predict-images resnet18, which is a dataset consisting of six images and contains image features extracted from a pre-trained CNN, ResNet-18.

Image features are configured at the source level. For more information about the image features and how to configure them, please refer to section Image Analysis of the Sources with the BigML Dashboard[13][11].

For the rest of batch predictions with images, including batch prediction configuration options and output datasets, everything stated earlier in current section (Subsection 4.6.2.2) applies.

### 4.6.3 Configuring Logistic Regression Predictions

BigML provides several options to configure your predictions such as setting a probability threshold (Subsection 4.6.3.1), default values for your missing numeric values (see Subsection 4.6.3.2), fields mapping (see Subsection 4.6.3.3), and output file settings (see Subsection 4.6.3.4.)

#### 4.6.3.1 Probability threshold

Probability thresholds usually makes sense when you want to minimize false positives at the cost of false negatives. The positive class will be predicted if its probability is greater than the given threshold, otherwise the following class with greater probability will be predicted instead.

To configure a threshold for your predictions follow these steps:

1. Use the selector shown in Figure 4.107 to select the **positive class**, i.e., the class for which you want to apply the probability threshold.

![Figure 4.107: Select the positive class for single predictions](image)

2. Then set a threshold value between 0% and 100% using the slider (see Figure 4.108). The positive class will be predicted if its probability is greater than the given threshold, otherwise the following class with greater probability will be predicted instead.
Figure 4.108: Set the probability threshold for single predictions

For batch predictions you can find the same options under the CONFIGURE panel:

Figure 4.109: Set the probability threshold for batch predictions
### 4.6.3.2 Default Numeric Value

If the dataset used to make the batch prediction contains instances with missing values for the numeric fields, the prediction will not be computed for them, unless you built the logistic regression enabling the **Missing numerics** parameter (see Subsection 4.4.5).

By using the **Default numeric value** before creating your batch prediction, you can easily replace all the missing numeric values by the field’s **Mean**, **Median**, **Maximum**, **Minimum** or by **Zero**. (See Figure 4.110.)

![Figure 4.110: Configure Default numeric value for batch prediction](image)

### 4.6.3.3 Fields Mapping

You can specify which input fields of the logistic regression match with the fields in the dataset containing the instances you want to predict. BigML automatically matches fields by **name**, but you can also set an automatic match by **field ID** by clicking on the green switcher. Additionally, you can manually search for fields or remove them from the **Dataset fields** column if you do not want them to be considered during the batch prediction. (See Figure 4.111.)
Note: Fields mapping from the BigML Dashboard is limited to 200 fields. For batch predictions with a higher number of fields, map your fields using the BigML API.\[14\]

4.6.3.4 Output Settings

Batch predictions return a CSV file containing all your instances and the final predictions. Tune the following settings to customize your prediction file (see Figure 4.112):

- **Separator**: this option allows you to choose the best separator for your output file columns. The default separator is the comma. You can also select the semicolon, the tab, or the space.
- **New line**: this option allows you to set the new line character to use as the line break in the generated csv file: “LF”, “CRLF”.
- **Output fields**: by clicking on the list icon next to the separator selector, you can include or exclude all your dataset fields from your output file. You can also individually select the fields you want to include or exclude using the multiple output fields selector. **Note**: a maximum of 100 fields can be displayed in this selector, but all your dataset fields will be included in the output file by default unless you exclude them.
- **Headers**: this option includes or excludes a first row in the output file (and in the output dataset) with the names of each column (input field names, prediction column name, probability column name, etc.). By default, BigML includes the headers.
- **Prediction column name**: customize the name for your predictions column. By default, BigML takes the name of the logistic regression's objective field.
- **Probability**: this option allows you to include an additional column with the probability for the predicted class. By default it is not included in your output file.
- **Probability column name**: customize the name for the probability column if you include it in the output file. BigML sets “probability” as the default name.

\[14\]https://bigml.com/api/batchpredictions#bp_batch_prediction_arguments
• **All class probabilities**: this includes all the probabilities of the objective field classes per instance. This option will add \( n \) extra columns, one by class in the objective field.

![Logistic regression output settings for batch predictions](image)

**Figure 4.112: Logistic regression output settings for batch predictions**

### 4.6.4 Visualizing Logistic Regression Predictions

Logistic regression predictions visualization changes depending on if you are predicting one **single** instance (Subsection 4.6.4.1), or you are predicting multiple instances using the **batch predictions** option (Subsection 4.6.4.2).

#### 4.6.4.1 Single Predictions

For single predictions, find the predicted class given the input fields values at the top of the form along with its probability. (See Figure 4.113.)
Figure 4.113: Logistic regression single prediction

Below the prediction, there’s a histogram representing the rest of the objective field class probabilities. All the class probabilities must sum 100%. Show or hide this view by clicking on the icon highlighted in Figure 4.114.
You can see up to seven different classes at the same time; if you have more than seven classes, you will see that some arrow icons appear next to the histogram so you can see the rest of classes.

**Export** this view in PNG format, in a CSV file, or in a JSON file by clicking on the corresponding icons. (See Figure 4.115.)
Figure 4.115: Logistic regression export all class probabilities histogram

Set a **probability threshold** for a selected positive class to minimize false negatives at the cost of false positives. If the probability for the positive class is greater than the established threshold, then the positive class will be predicted. Otherwise, the next class with higher probability will be predicted instead.
4.6.4.1 Local Predictions

Local predictions are provided for single instances from the BigML Dashboard which are performed faster at no cost. Local predictions allow you to get a real-time prediction without consuming any credits or requiring any internet connection. This is possible because the logistic regression is saved in-memory, so when the input values change, BigML is able to compute predictions in microseconds.

4.6.4.1.2 Prediction explanation

Prediction explanation helps understand why a logistic regression makes a certain prediction. This is very useful in many applications, and the reasons behind a prediction are often as important as the prediction itself.

BigML prediction explanation is based on Shapley values. For more information, please refer to this research paper: A Unified Approach to Interpreting Model Predictions [3].

For any logistic regression, you can request the explanation for the prediction by clicking the prediction explanation icon and then click Save (see Figure 4.117).
Chapter 4. Logistic Regressions

The prediction explanation represents the most important factors considered by the logistic regression in a prediction given the input values. Each input value will yield an associated importance, as you can see Figure 4.118. The importances across all input fields should sum 100%.

For some input fields you will see a “+” icon next to the importance. This is because the importance may not be only directly associated with the input value, i.e., it can also be explained by other reasons. In the Figure 4.119 below, the importance of 28.81% for the field “Class/Dept” is not only explained by this field being equal to “1st Class”. Rather, it is because this field value is not “3rd Class” (which accounts for the majority of importance, 23.98%) and, then because it is “1st Class” (4.83% of importance).
The prediction explanation for logistic regressions is calculated using the results of over a thousand distinct predictions using random perturbations of the input data. For this reason, the calculation of the explanation may take some time to be computed.

4.6.4.2 Batch Prediction

After creating your batch prediction, you get a CSV file and, optionally, an output dataset. Both outputs are explained in the following subsections.

4.6.4.2.1 Output CSV file

The batch prediction generates a CSV file containing your predictions for each of your dataset instances in the last column. (See Figure 4.120.)

You can configure several options to customize your CSV file. You can find a detailed explanation of those options in Subsection 4.6.3.4.

See an output CSV file example in Figure 4.121. The column class in this example contains the final prediction (it is named by default as your logistic regression’s objective field). In this case we are predicting whether a person is a good or a bad candidate for holding a credit. This file has been configured to contain also the probability for each prediction.
<table>
<thead>
<tr>
<th>duration</th>
<th>age</th>
<th>amount</th>
<th>purpose</th>
<th>class</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>26</td>
<td>5433</td>
<td>used car</td>
<td>good</td>
<td>0.88785</td>
</tr>
<tr>
<td>36</td>
<td>42</td>
<td>8086</td>
<td>new car</td>
<td>bad</td>
<td>0.58526</td>
</tr>
<tr>
<td>24</td>
<td>28</td>
<td>1376</td>
<td>radio/tv</td>
<td>good</td>
<td>0.8385</td>
</tr>
<tr>
<td>48</td>
<td>31</td>
<td>6758</td>
<td>radio/tv</td>
<td>bad</td>
<td>0.72576</td>
</tr>
<tr>
<td>26</td>
<td>30</td>
<td>7966</td>
<td>used car</td>
<td>good</td>
<td>0.7201</td>
</tr>
<tr>
<td>12</td>
<td>42</td>
<td>2577</td>
<td>furniture/equipment</td>
<td>good</td>
<td>0.67644</td>
</tr>
<tr>
<td>36</td>
<td>30</td>
<td>4455</td>
<td>business</td>
<td>good</td>
<td>0.52227</td>
</tr>
<tr>
<td>18</td>
<td>32</td>
<td>1442</td>
<td>new car</td>
<td>bad</td>
<td>0.75488</td>
</tr>
<tr>
<td>9</td>
<td>22</td>
<td>276</td>
<td>new car</td>
<td>good</td>
<td>0.57819</td>
</tr>
</tbody>
</table>

**Figure 4.121: An example of a logistic regression batch prediction CSV file**

### 4.6.4.2.2 Output Dataset

By default BigML automatically creates a dataset out of your batch prediction. You can disable this option by configuring your batch prediction. (See Subsection 4.6.3.4.) You will find the output dataset in your batch prediction view as shown in Figure 4.122.

![Batch prediction output dataset](image)

In the output dataset, you can find an additional field (named by default as your logistic regression’s objective field) containing the **class predicted** for each one of your instances. If you configured your batch prediction to include the prediction **probabilities** and **all class probabilities** you will be able to find them in the last fields of your output dataset. (See Figure 4.123.)
4.6.5 Consuming Logistic Regression Predictions

You can fully use single and batch predictions via the BigML API and bindings. The following subsections explain both tools.

4.6.5.1 Using Logistic Regression Predictions via the BigML API

Logistic regression predictions have full citizenship in the BigML API which allows you to programmatically create, configure, retrieve, list, update, and delete single and batch predictions.

In the example below, see how to create a single prediction using a logistic regression and defining the input data once you have properly set the `BIGML_AUTH` environment variable to contain your authentication credentials:

```bash
curl "https://bigml.io/prediction?$BIGML_AUTH" \
  -X POST \
  -H "content-type: application/json" \
  -d '{"logisticregression": "logisticregression/50650bdf3c19201b64000020", 
       "input_data": {"000001": 3, "000002": 4.5, "000003": 5}}'
```

For more information on using logistic regressions through the BigML API, please refer to the documentation.

4.6.5.2 Using Logistic Regression Predictions via BigML Bindings

You can also create, configure, retrieve, list, update, and delete single and batch predictions via BigML bindings which are libraries aimed to make it easier to use the BigML API from your language of choice. BigML offers bindings in multiple languages including Python, Node.js, Java, Swift and Objective-C. See below an example to create a logistic regression with the Python bindings.

15https://bigml.com/api/logisticregressions
from bigml.api import BigML
api = BigML()
prediction = api.create_prediction(
  "logisticregression/50650bdf3c19201b64000020",
  {"credit_amount": 5, "duration": 2.5})

For more information on BigML bindings, please refer to the bindings page\textsuperscript{16}.

### 4.6.6 Descriptive Information

Each logistic regression prediction has an associated \textit{name}, \textit{description}, \textit{category}, and \textit{tags}. You can find a brief description of each concept in the following subsections. The \textit{MORE INFO} menu option displays a panel that provides editing options. (See Figure 4.124.)

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{logistic_regression_prediction_descriptive_information.png}
\caption{Logistic regression prediction descriptive information}
\end{figure}

#### 4.6.6.1 Name

If you do not specify a \textit{name} for your predictions, BigML assigns a default name depending on the type of predictions:

- **Single predictions**: the name always follows the structure “Prediction for \textit{<objective field name>}".

- **Batch predictions**: BigML combines your prediction dataset name and the logistic regression name: “Batch prediction of \textit{<logistic regression name>} with \textit{<dataset name>}”.

Predictions names are displayed on the list and also on the top bar of a prediction view. Predictions names are indexed to be used in searches. Rename your predictions any time from the \textit{MORE INFO} menu.

The name of a prediction cannot be longer than 256 characters. More than one prediction can have the same name even within the same project, but they will always have different identifiers.

#### 4.6.6.2 Description

Each prediction also has a \textit{description} that it is very useful for documenting your Machine Learning projects. Predictions take their description from the logistic regression used to create them.

\textsuperscript{16}https://bigml.com/tools/bindings
Descriptions can be written using plain text and also markdown\(^{17}\). BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 4.125.)

![Markdown editor for logistic regression descriptions](image)

Figure 4.125: Markdown editor for logistic regression descriptions

Descriptions cannot be longer than 8192 characters.

4.6.6.3 Category

A category taken from the logistic regression used to create it is associated with each prediction. Categories are useful to classify predictions according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

A prediction category must be one of the categories listed on table Table 4.5.

4.6.6.4 Tags

A prediction can also have a number of tags associated with it. These tags help to retrieve the prediction via the BigML API or to provide predictions with some extra information. Your prediction inherits the tags from the logistic regression used to create it. Each tag is limited to a maximum of 128 characters. Each prediction can have up to 32 different tags.

4.6.7 Logistic Regression Predictions Privacy

The link displayed in the Privacy panel is the private URL of your prediction, so only a user logged into your account is able to see it. Neither single predictions nor batch predictions can be shared by using a secret link. (See Figure 4.126.)

\(^{17}\) [https://en.wikipedia.org/wiki/Markdown](https://en.wikipedia.org/wiki/Markdown)
Chapter 4. Logistic Regressions

4.6.8 Moving Logistic Regression Predictions to Another Project

When you create a prediction, it will be assigned to the same project where the original logistic regression is located. You cannot move predictions between projects as you do with other resources.

4.6.9 Stopping Logistic Regression Predictions

Single predictions are synchronous resources, so you cannot cancel them during the creation since you get the result immediately.

By contrast, batch predictions are asynchronous resources, so you can stop their creation before the task is finished. Use the DELETE BATCH PREDICTION option from the 1-click action menu (Figure 4.127) or from the pop up menu on the list view.

A modal window will be displayed asking you for confirmation. If you stop the prediction during its creation you won’t be able to resume the same task again, so if you want to create the same prediction you will have to start a new task.
4.6.10 Deleting Logistic Regression Predictions

You can delete your single or batch predictions from the predictions view, using the 1-click action menu (see Figure 4.129) or using the pop up menu on the predictions list view (see Figure 4.130).
A modal window will be displayed asking you for confirmation. Once a prediction is deleted, it is permanently deleted, and there is no way you (or even the IT folks at BigML) can retrieve it.

Figure 4.131: Logistic regression delete prediction confirmation

## 4.7 Consuming Logistic Regressions

Similarly to other models in BigML, logistic regressions are white-boxed models, so you can download them and use them locally to make predictions. You can also create and consume your logistic regressions programmatically via the BigML API and bindings. The following subsections explain those three options.

### 4.7.1 Downloading Logistic Regressions

You can download your logistic regression in several programming languages including JSON PML, Python or Node.js. By downloading your logistic regression you will be able to compute predictions locally, free of latency and at no cost. Click on the download icon in the top menu (see Figure 4.132), and select your preferred option (see Figure 4.133.)

Figure 4.132: Click download icon
4.7.2 Using Logistic Regressions Via the BigML API

Logistic regression have full citizenship in the BigML API which allows you to programmatically create, configure, retrieve, list, update, delete, and use them for predictions.

In the below example, see how to create a logistic regression using an existing dataset once you have properly set the `BIGML_AUTH` environment variable to contain your authentication credentials:

```
curl "https://bigml.io/logisticregression?\$BIGML_AUTH" \
  -X POST \
  -H 'content-type: application/json' \
  -d '{"dataset": "dataset/50650bdf3c19201b64000020"}'
```

For more information on using logistic regressions through the BigML API, please refer to the documentation.

4.7.3 Using Logistic Regressions Via the BigML Bindings

You can also create and use logistic regressions via BigML bindings which are libraries aimed to make it easier to use the BigML API from your language of choice. BigML offers bindings in multiple languages including Python, Node.js, Java, Swift and Objective-C. See below an example to create a logistic regression with the Python bindings.

```python
from bigml.api import BigML
api = BigML()
logistic_regression = api.create_logistic_regression(
    'dataset/57506c472275c1666b004b10', {'objective_field':"churn"})
```

18https://bigml.com/api/logisticregressions
4.8 Logistic Regression Limits

There are some limits that apply for the creation of any BigML resource. These are limits based on the number of classes, terms and items that can be considered to create your models. This is explained in Subsection 4.8.0.1.

Additionally, some specific limits apply for your logistic regressions visualization, i.e. to the logistic regression chart and the coefficient table views, depending on the number of classes in the objective field and the number of input fields in your dataset. See Subsection 4.8.2 and Subsection 4.8.1 for a detailed explanation.

Note: chart limits and coefficient table limits just affect to the visualization of the model, i.e., despite your dataset reach those limits, you can still creating the logistic regression, evaluating it and using it to make predictions.

4.8.0.1 Field Limits

Logistic regression, similarly to other BigML models, has the following limitations according to the type of field:

- **Classes**: a maximum number of 1,000 distinct classes per field is allowed.
- **Terms**: BigML can handle up to 1,000 terms in total. If multiple text fields are defined, then the token limit per field is evenly divided by the number of text fields evenly, e.g., a dataset with two text fields would result in 500 terms per text field. BigML selects those terms with most significant frequency, discarding both those that appear either too often or too infrequently. A maximum of 256 characters per term is allowed.
- **Items**: a maximum number of 10,000 distinct items per field is allowed.

4.8.1 Chart Limits

There are some circumstances under which your chart cannot be displayed:

- As the chart only supports numeric fields for the x-axis, if your logistic regression only contains categorical, text, or items fields, the chart cannot be displayed. The view displayed by default will be the coefficient table. When you try to click on the chart icon you will see a warning message. (See Figure 4.134.)

![Figure 4.134: Warning message when the logistic regression does not have any numeric field](https://bigml.com/tools/bindings)

For more information on BigML bindings, please refer to the bindings page\(^{19}\).

\(^{19}\)https://bigml.com/tools/bindings
If your logistic regression contains more than **100 fields** the chart cannot be displayed and again the default view will be the table view. When you try to click on the chart icon, you will see the warning message shown in Figure 4.135.

Figure 4.135: Warning message when the logistic regression has more than 100 fields

If your logistic regression contains more than **200 categories** in the objective field, neither the chart nor the table can be displayed and you will see the warning message shown in Figure 4.136. You can still see your coefficients by downloading the **CSV file**.

Figure 4.136: Warning message when the objective field has more than 200 classes

### 4.8.2 Coefficient Table Limits

If your logistic regression contains more than **1,000 fields** and/or more than **200 categories** in the objective field, the table cannot be displayed. You will need to **download the CSV** if you want to see your logistic regression coefficients. You will get the message shown in Figure 4.137:
4.9 Descriptive Information

Each logistic regression has an associated name, description, category, and tags. The following subsections provide a brief description for each concept. In Figure 4.138, you can see the options the MORE INFO menu provides to edit them.

4.9.1 Logistic Regression Name

Each logistic regression has a name that is displayed in the logistic regression list view and also on the top bar of the logistic regression view. Logistic regression's names are indexed to be used in searches. When you create a logistic regression, it gets a default name. Change it using the MORE INFO menu option on the right corner of the logistic regression view. The name of a logistic regression cannot be longer than 256 characters. More than one logistic regression can have the same name even within the
same project, but they will always have different identifiers.

### 4.9.2 Description

Each logistic regression also has a **description** that it is very useful for documenting your Machine Learning projects. Logistic regressions take the description of the datasets used to create them by default.

Descriptions can be written using plain text and also markdown\(^{20}\). BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 4.139.)

![Figure 4.139: Markdown editor for logistic regression descriptions](https://en.wikipedia.org/wiki/Markdown)

Descriptions cannot be longer than **8192** characters and can use almost any character.

### 4.9.3 Category

A **category**, taken from the dataset used to create it, is associated with each logistic regression. Categories are useful to classify logistic regressions according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

A logistic regression category must be one of the 24 categories listed on Table 4.5.

---

\(^{20}\)https://en.wikipedia.org/wiki/Markdown
Table 4.5: Categories used to classify logistic regression by BigML

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace and Defense</td>
</tr>
<tr>
<td>Automotive, Engineering and Manufacturing</td>
</tr>
<tr>
<td>Banking and Finance</td>
</tr>
<tr>
<td>Chemical and Pharmaceutical</td>
</tr>
<tr>
<td>Consumer and Retail</td>
</tr>
<tr>
<td>Demographics and Surveys</td>
</tr>
<tr>
<td>Energy, Oil and Gas</td>
</tr>
<tr>
<td>Fraud and Crime</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Higher Education and Scientific Research</td>
</tr>
<tr>
<td>Human Resources and Psychology</td>
</tr>
<tr>
<td>Insurance</td>
</tr>
<tr>
<td>Law and Order</td>
</tr>
<tr>
<td>Media, Marketing and Advertising</td>
</tr>
<tr>
<td>Miscellaneous</td>
</tr>
<tr>
<td>Physical, Earth and Life Sciences</td>
</tr>
<tr>
<td>Professional Services</td>
</tr>
<tr>
<td>Public Sector and Nonprofit</td>
</tr>
<tr>
<td>Sports and Games</td>
</tr>
<tr>
<td>Technology and Communications</td>
</tr>
<tr>
<td>Transportation and Logistics</td>
</tr>
<tr>
<td>Travel and Leisure</td>
</tr>
<tr>
<td>Uncategorized</td>
</tr>
<tr>
<td>Utilities</td>
</tr>
</tbody>
</table>

4.9.4 Tags

A logistic regression can also have a number of tags associated with it that can help to retrieve it via the BigML API or to provide logistic regressions with some extra information. A logistic regression inherits the tags from the dataset used to create it. Each tag is limited to a maximum of 128 characters. Each logistic regression can have up to 32 different tags.

4.9.5 Counters

For each logistic regression, BigML also stores a number of counters to track the number of other resources that have been created using the logistic regression as a starting point. In the logistic regression view, you can see a menu option that displays counters for evaluations, single and batch predictions, and the fusions created. It also allows you to quickly jump to all the resources of one type. (See Figure 4.140.)

Figure 4.140: Counters for logistic regressions
4.10 Logistic Regression Privacy

Privacy options for a logistic regression can be defined in the More Info panel, displayed in Figure 4.141. There are two levels of privacy for BigML logistic regressions:

- **Private**: only accessible by authorized users (the owner and those who have been granted access by him or her).
- **Shared**: accessible by any user with whom the owner shares the secret link.

4.11 Moving Logistic Regressions to Another Project

When you create a logistic regression, it will be assigned to the same project where the original dataset is located.

Logistic regressions can only be assigned to a single project. However, you can move logistic regressions between projects. The menu option to do this can be found in two places:

1. In the logistic regression list view, click the **MOVE TO** option within the 1-click action menu and select another project or create a new one. (See Figure 4.142.)

2. In the logistic regression list view, click the **MOVE TO** option within the pop up menu and select another project or create a new one. (See Figure 4.143.)
4.12 Stopping Logistic Regressions

You can stop the creation of a logistic regression before the task is finished by clicking the DELETE LOGISTIC REGRESSION option from the 1-click action menu (see Figure 4.144), or from the pop up menu in the logistic regression list view (see Figure 4.145).
A modal window will be displayed asking you for confirmation. If you stop the logistic regression during its creation you won’t be able to resume the same task. If you want to create the same logistic regression, you will have to start a new task.

4.13 Deleting Logistic Regressions

You can delete your logistic regressions by clicking in the DELETE LOGISTIC REGRESSION option from the 1-click action menu (see Figure 4.147) or using the pop up menu on the logistic regression list (see Figure 4.148).
Chapter 4. Logistic Regressions

A modal window will be displayed asking you for confirmation. Once a logistic regression is deleted, it is permanently deleted, and there is no way you (or even the IT folks at BigML) can retrieve it.

Copyright © 2023, BigML, Inc.
4.14 Takeaways

This chapter explains logistic regressions in detail. Here is a list of key points:

- A logistic regression is a supervised Machine Learning algorithm used to solve classification problems.
- A logistic regression is built from a dataset available in BigML and used to make an evaluation, a single prediction, or a batch prediction. (See Figure 4.150.)
- You can create a logistic regression with just one click or configure it as you wish. BigML provides several configuration options before creating your logistic regression.
- To create a logistic regression you need a dataset containing at least one categorical field.
- Categorical fields must be converted to numeric values in order to train a logistic regression model.
- If you do not specify any objective field, BigML will take the last valid field in your dataset.
- BigML allows you to include your numeric fields' missing values as valid values to train your logistic regression model.
- The chart view provides a visual way to analyze a field impact on predictions given certain values for the rest of the fields.
- You get all the objective field class probabilities along with the predicted class.
- BigML displays all your logistic regression coefficients in a table view which you can also download as a CSV file.
- You need to evaluate your logistic regression model's performance using data that the model has not seen before.
- The ultimate goal in building a logistic regression is being able to make predictions with it.
- BigML allows you to quickly make predictions for single instances by providing a form containing the fields used by the logistic regression, so you can easily set the input data and get an immediate response.
- BigML batch predictions allow you to make simultaneous predictions for multiple instances. All you need is the logistic regression you want to use to make predictions and a dataset containing the instances for which you want to obtain predictions.
- You can configure your batch predictions output file settings.
- You can download your logistic regression to perform local predictions.
- You can add descriptive information to your logistic regressions (name, description, tags, and category).
- You can move your logistic regressions between projects.
- You can share your logistic regressions with other people using the secret link.
- You can stop your logistic regression creation by deleting them.
- You can permanently delete an existing logistic regression.
Figure 4.150: Logistic Regression Workflow
Deepnets

5.1 Introduction

There are multiple Machine Learning problems that can be solved using supervised Machine Learning techniques. Some of these problems require to predict an output variable (objective field) given a number of input variables (input fields). These problems can be divided into classification and regression depending on whether you need to predict a category (label or class) or a continuous value (a real number), respectively. To learn more about concrete use cases for both problems refer to Section 1.1.

Deep neural networks (or deepnets) are a Machine Learning technique that can be used to solve classification and regression problems. These problems can also be solved with other Machine Learning methods, such as models, ensembles, or logistic regressions. These methods are explained in Chapter 1, Chapter 2, and Chapter 4 respectively. Depending on the problem you are trying to solve and the data available, some techniques may perform significantly better than others. See in Subsection 5.2.3 a detailed explanation of how deepnets perform compared to other supervised learning techniques for different use cases.

Deepnets are a class of machine learning models inspired by the neural circuitry of the human brain. See Section 5.2 for more details on the deepnets algorithm.

This chapter contains comprehensive description of BigML’s deepnets including how they can be created with 1-click (Section 5.3), all configuration options available (Section 5.4), and the different visualizations provided by BigML (Section 5.5). See Section 5.6 for an explanation of how deepnets can be used to make predictions. You can also export your deepnets in different formats to make local predictions faster at no cost (Subsection 5.7.1). The process to evaluate your deepnets predictive performance in BigML is explained in a different chapter (Chapter 7).

In BigML, the third tab of the main menu of the Dashboard allows you to list all of your available deepnets. The deepnet list view (Figure 5.1), details the Dataset used to create it, the Name, Objective (objective field name), Age (time elapsed since it was created), Size, and number of evaluations, predictions, and batch predictions that have been created using that deepnet. The search menu option in the top right corner of the deepnet list view allows you to search your deepnets by name.
By default, when you first create an account at BigML, or every time that you start a new project, your list of deepnets will be empty. (See Figure 5.2.)

Finally, in Figure 5.3 you can see the icon used to represent a deepnet.

Figure 5.1: Deepnet list view

Figure 5.2: Empty Dashboard deepnet view

Figure 5.3: Deepnet icon

5.2 Understanding Deepnets

A deepnet in BigML is a supervised learning method to solve classification and regression problems. Deepnets are an optimized version of Deep Neural Networks, a class of machine learning models inspired by the neural circuitry of the human brain. In these classifiers, the input features are fed to one or several groups of “nodes”. Each group of nodes is called a “layer”. Each node is essentially a function on the input that transforms the input features into another value or collection of values (see also Subsection 5.4.7.1). Then the entire layer transforms an input vector into a new “intermediate”
feature vector. This new vector is fed as input to another layer of nodes. This process continues layer by layer, until we reach the final “output”, which is also a layer of nodes. The output is the network’s prediction: an array of per-class probabilities for classification problems or a single, real value for regression problems.

The “deep” in deep neural networks refers to the presence of more than one “hidden layer”; that is, more than one layer of nodes between the input and the output layers. The network architectures supported by BigML can be deep or shallow. The advantage of training deep architectures is that hidden layers have the opportunity to learn “higher-level” representations of the data that can be used to make correct predictions in cases where a direct mapping between input and output is difficult. For example, when classifying images of numeric digits, the input layer is raw pixels, the output layer is the probability for each digit, and the intermediate layers may learn features that represent the presence of, say, a loop or a vertical stroke. Read Subsection 5.2.3 below for an explanation of the use cases where deepnets have a better performance.

5.2.1 Convolutional Neural Network

When the dataset used to create a deepnet contains images, the deepnet created will be a convolutional neural network.

Convolutional neural network, also known as CNN or ConvNet, is a type of deep neural networks. All deepnet operations described in this section, such as 1-click creation and configuration, also apply to CNNs. So does optimization, including automatic network search and structure suggestion.

The main difference between CNN and other types of neural networks is that the hidden layers of a CNN include at least one convolutional layer.

A convolutional layer performs one or more convolution operations. Each convolution operation transforms a set of neighboring inputs to an output, which is passed to the next layer. In the case of an image, the input layer consists of two-dimensional pixels. A convolution operation converts a block of pixels (say 3x3) to a single number. It can be imagined as a filter, sliding across the whole image and converting each (3x3) block of pixels to a number. The output of a convolutional layer can be thought as another image with a possibly decreased size and its pixels holding information from multiple (9) pixels of the image in the previous layer. The size of the pixel block (filter size) can be different (e.g. 3x3, 5x5), and one convolutional layer may have multiple convolution operations (number of filters).

There are other important layers in CNNs, such as ReLU and pooling layers. In the end when a CNN is fully trained, each convolutional layer, coupled with its ReLU and pooling layers, effectively captures image features. The first layers extract low-level features such as edges and colors while the deeper layers capture high-level features unique to the objects in the images. The outputs of convolutional layers are also called feature maps.

BigML can extract image features at the source level. For a composite source with images, different sets of image features can be extracted, which include features on edges, colors and texture. There are also pre-trained CNNs which capture more sophisticated features. Image features can be used to train supervised models as well as unsupervised models. If you know specific image features that will help you achieve your machine learning goals, you don’t have to use CNNs. Instead, you can configure your sources to extract image features. For information about the image features, please refer to section Image Analysis of the Sources with the BigML Dashboard.1

Note: When training a deepnet from a dataset containing images, that is, when training a CNN, all image feature fields extracted from the images will be ignored. In other words, when a dataset contains images, its deepnet is trained from raw image pixels, not from its extracted image features.

Because of convolution operations, which transform neighboring inputs such as 3x3 blocks, CNNs are excellent in machine learning using spatial data, especially images. However, CNNs may not work as well for non-spatial data such as tabular data. Think of this way: if many rows of a tabular data are swapped, the data is still considered the same. But doing the same to an image, it becomes a different image.

1https://static.bigml.com/pdf/BigML_Sources.pdf
For a comprehensive introduction to CNN, please refer to its wikipedia entry\(^2\).

### 5.2.2 Automatic Parameter Optimization

Deep neural networks are notoriously sensitive to the chosen topology (or network structure) and the algorithm used to learn the weights for that topology. This sensitivity means that hand-tuning the topology and optimization algorithm can be difficult and time-consuming as the number of choices that lead to poor networks typically vastly outnumber the choices that lead to good ones.

To combat this problem, BigML offers first-class support for automatic parameter optimization that allows for automated discovery of better networks via two different methods:

- **Automatic network search**: during the deepnet creation, BigML trains and evaluates over many possible network configurations, returning the best networks found for your problem. The final deepnet returned by the search is a “compromise” between the top “n” networks found in the search. The algorithm BigML uses for this optimization technique is a variant on the hyperband algorithm. Instead of selecting parameter value candidates for evaluation at random, however, BigML uses an acquisition technique based on techniques from Bayesian parameter optimization. The main downsides of using this optimization method is that the creation of the deepnet may be significantly slower.

  **Note**: the search process is not totally deterministic, so although you are using the same dataset you might get slightly different results from run to run. This is because BigML trains multiple models concurrently and the order in which they finish is important. After each model finishes, the search modifies its behavior based on the performance of the one that just finished (i.e., the next trained model in the search depends on the previous ones). Although results may not be repeatable, the differences should be almost unperceivable in most cases.

- **Automatic structure suggestion**: BigML offers a faster technique that can also give quality results. The ability to quickly train and test your deepnets is especially useful when working on feature engineering. BigML has trained thousands of networks on dozens of datasets in order to understand the effectiveness of various network topologies. As such, BigML has learned some general rules about what makes one network structure better than another for a given dataset. BigML will automatically suggest a structure and set of parameter values that are likely to perform well for your dataset.

To learn more about these optimization techniques, read this blog post\(^3\).

You can choose either optimizing technique by selecting it in the configuration panel (see Subsection 5.4.2). Alternatively, you can manually set the parameters for your deepnet. By default, BigML uses the automatic structure suggestion strategy to create deepnets.

### 5.2.3 Deepnets Use Cases

A common question when solving classification and regression problems is which algorithm should be used to get the best results: models and ensembles? Logistic regressions? Deepnets? In most cases, there is not an effective way to know in advanced which method will perform better so the best strategy is training and evaluating each one of them and compare their performances. However, a general rule for deepnets is that they usually perform better with complex datasets and difficult problems. That is, high-dimensional datasets where either only a few features are non-noise, or where the decision function is spread across many different features.

On one hand, decision trees and ensembles have spectacular representational power when the dataset has a high number of variables because their hypothesis space grows with the data and the decision tree algorithm is able to efficiently search through the space to get a good solution in reasonable time. However, trees have trouble representing objectives that are smooth functions of lots of variables.

---

\(^2\)https://en.wikipedia.org/wiki/Convolutional_neural_network

\(^3\)https://blog.bigml.com/2017/10/04/deepnets-behind-the-scenes/
On the other hand, logistic regression optimizes a function that takes into account all of the variables at once, not just one or two at a time, and is able to optimize this function efficiently. However, their representational power is considerably lower; they can only represent linear decision boundaries.

Deepnets try to get the best of both methods. Because their structure is very flexible, they have high representational power, and because they’re optimized via gradient descent (like logistic regression), they do fine with smooth functions of potentially all of the input variables.

The main downside of deepnets is the efficiency that both decision trees and logistic regression provide. The structure of deepnets is super-flexible, but there is no way to search through the possible structures and parameters as quickly as can be done for trees or logistic regression. The only way to find an optimal structure is to try a lot of them. BigML tries to make this search as clever as possible (see Subsection 5.2.2), but it is still significantly more time-consuming than trees and logistic regression with no guarantee that it will beat them.

### 5.2.4 Missing Values

BigML deepnets can handle missing values for any type of field. For categorical, text, and items fields, missing values are always included by default.

For numeric fields, missing values are also included by default, but you can deactivate this option by configuring your deepnet (see Subsection 5.4.4). If the missing numeric option is disabled, the instances containing missing values for numeric fields in your dataset will be ignored by the deepnet. Also when using your deepnet to make predictions, you will not be able to have missing values for the numeric fields in the input data.

### 5.3 Creating Deepnets with 1-Click

To create a deepnet in BigML you have two options: either the 1-click option which uses the default values for all available configuration options or you can tune the parameters in advanced by using the configuration options explained in Section 5.4. This section explains how to create a deepnet with 1-click.

From the dataset view, select the 1-CLICK DEEPNET option in the 1-click action menu. (See Figure 5.4.)

Alternatively, you can use the 1-CLICK DEEPNET option in the pop up menu from the dataset list view. (See Figure 5.5.)
Figure 5.5: Create 1-click deepnet from dataset popup menu

Either option builds a deepnet using the default values for all available configuration options. (See Section 5.4.)

Note: your dataset needs to contain at least one categorical or numeric field to be selected as the objective field to create the deepnet.

5.4 Deepnet Configuration Options

While the 1-click creation menu option (see Section 5.3) provides a convenient and easy way to create a BigML deepnet, you can also have more control over the deepnet creation and configure a number of parameters that affect the way BigML creates deepnets. Click the CONFIGURE DEEPNET menu option in the configuration menu of your dataset view. (See Figure 5.6.)

Figure 5.6: Configure deepnet

5.4.1 Objective Field

The objective field, or "target field", is the field you want to predict. Deepnets support categorical or numeric fields as the objective field.
BigML takes the last valid field in your dataset as the objective field by default. If you want to change the objective field, you have two options: you can select another field from the configuration panel to build the deepnet, or you can change it permanently from your dataset view.

- Select the **Objective field** from the deepnet configuration panel. This option will only affect the deepnet you are building that time. (See Figure 5.7.)

  ![Figure 5.7: Configure the objective field to create the deepnet](image)

- Change the **default objective field** for the dataset. This option will save your objective field preference for any model you build. Click on the edition icon next to the field name when you mouse over it, a pop up window will be displayed. Then click on the **Objective field** icon and **Save** it. (See Figure 5.8.)

  ![Figure 5.8: Change the default objective field](image)

### 5.4.2 Automatic Parameter Optimization

The high number of configurable parameters for neural networks makes it difficult to find the optimum configuration to get good results. Hand-tuning different configurations is a time-consuming process in
which the combinations that lead to a poor result outnumber the ones that result in a satisfying performance. To combat this problem, BigML offers first-class support for automatic parameter optimization via two different methods:

- **Automatic network search**: during the deepnet creation, BigML trains and evaluates over many possible network configurations, returning the best networks found for your problem. The final deepnet returned by the search is a “compromise” between the top “n” networks found in the search. The main problem of using this optimization method is that the creation of the deepnet may be significantly slower.

  **Note**: the search process is not totally deterministic, so although you are using the same dataset you might get slightly different results from run to run.

- **Automatic structure suggestion**: BigML offers a faster technique that can also give quality results. BigML has learned some general rules about what makes one network structure better than another for a given dataset. BigML will automatically suggest a structure and a set of parameter values that are likely to perform well for your dataset.

Read more about the automatic parameter optimization in Subsection 5.2.2. You can choose either optimizing technique by selecting it in the configuration panel (see Figure 5.9).

![Figure 5.9: Select an automatic optimization option](image)

When you select an optimization strategy, the **Network architecture** parameters (see Subsection 5.4.7), the **Algorithm** parameters (see Subsection 5.4.8) and the **Weights** (see Subsection 5.4.9) will be automatically set. You cannot manually tune any of them except the **Weights** which you can manually configure it. If you want to configure the rest, you need to deactivate the automatic optimization options using the switcher as shown in Figure 5.10,
5.4.3 Default Numeric Value

Deepnets can include missing values as valid values for any type of fields as explained in Subsection 5.2.4. However, there can be situations for which you do not want to include them in your model. For those cases, the default numeric value parameter is an easy way to replace missing numeric values by another valid value. You can select to replace them by the field's Mean, Median, Maximum, Minimum or by Zero. (See Figure 5.11.)

Note: if your dataset does not contain missing values for your numeric fields, this parameter will not have impact on your deepnet. If your dataset contains missing numeric values and you neither select a default numeric value or enable the missing numerics configuration option, instances with missing numeric values will be ignored to build the model.

5.4.4 Missing Numerics

By default, missing values for your numeric fields are included as valid values to build your deepnets. However, there can be cases for which you do not want them to be included in your model. The Missing numerics option allows you to select if you want to include or exclude the missing numeric values to build your deepnets. (See Figure 5.12.)
5.4.5 Training Duration

The scale parameter to regulate the deepnet runtime. It’s set as an integer from 1 to 10. It indicates the user preference for the amount of time they wish the optimization to take. The higher the number, the more time that users are willing to wait for possibly better deepnet performance. The lower the number, the faster that users wish the deepnet training to finish. The default value is set to 5.

The training duration is set in a scale. The actual training time depends on the dataset size, among other factors.

5.4.6 Maximum Iterations

The number of iterations in a deepnet is the number of gradient steps the algorithm takes during the optimization process. You can set the maximum number of iterations to train your deepnet by activating this option using the switcher (see Figure 5.14). By default, this option will be deactivated, in which case
BigML will stop training the network if a certain number of iterations goes by without substantial progress or if it reaches the limit of the training duration (see Subsection 5.4.5).

5.4.7 Network Architecture

The basic architecture specification for a network consists of specifying the number of hidden layers in the network, the number of nodes in each layer, the activation function, and other parameters related to how the network connections are arranged such as residual learning, batch normalization and tree embedding.

5.4.7.1 Hidden Layers

The hidden layers in a neural network are the intermediate layers between the input layer (the one containing the input field values) and the output layer (the one containing the predictions). (See Section 5.2). In BigML you can configure up to 32 hidden layers. For each layer you can specify the following parameters:

- The **activation function**: by applying an activation function, the deepnet is able to represent a non-linear mapping between the inputs and the outputs which is necessary to solve complex problems. The activation function converts an input into an output that is used as an input again to feed the next layer in the network and so on. If no activation function is applied, the output will be simply a linear function of the inputs. In BigML you can select one of the following functions: “Tanh”\(^4\), “Sigmoid”\(^5\), “Softplus”\(^6\), “Softmax”\(^7\), “ReLU”\(^8\) or “None”. If “None” is selected, no activation function will be set for the layers so the raw output values for each node will be used as inputs for the next layer.

To know more about each type of activation function please refer to this article\(^9\)

- The **number of nodes**: each hidden layer in the network can have a variable number of nodes. Determining the optimal number of hidden nodes (also called hidden units or neurons) per layer is a complex task depending on:

  - The number of nodes of the **input and output layers**. The number of nodes in the input layer is equal to the number of fields of the dataset used to create the deepnet. The output

---

\(^4\)https://en.wikipedia.org/wiki/Hyperbolic_function  
\(^5\)https://en.wikipedia.org/wiki/Sigmoid_function  
\(^7\)https://en.wikipedia.org/wiki/Softmax_function  
\(^8\)https://en.wikipedia.org/wiki/Rectifier_(neural_networks)  
\(^9\)https://medium.com/towards-data-science/activation-functions-and-its-types-which-is-better-a6a65310cc8f
layer always has one node if it is a regression problem or as many nodes as classes has the objective field if it is a classification problem.

- The number of instances in the training dataset.
- The complexity of the problem that is trying to be solved.
- The gradient descent algorithm (see Subsection 5.4.8.1) and the activation function used.

The higher the number of nodes, the higher the risk of overfitting. However, too few nodes may lead to a poor solution if the function to be learned has some complexity. In most cases, you will need to try different size for the layers or use one of the BigML optimization options (see Subsection 5.4.2) to reach a number of nodes per layer that provide satisfying results. In BigML you can set up to 8,192 nodes per layer. A rule of thumb to determine the size of the layers is that the number of hidden nodes should be somewhere in between the sizes of the input and the output layers. Also, the size of the hidden layer should not exceed twice the size of the input layer because at this point it is very likely to overfit. Read this article\(^{10}\) to know more about the variables that should be taken into account when deciding the size of the layers.

The weights for each layer are initialized randomly according to Xavier’s method\(^{11}\).

If one of the automatic optimization options (see Subsection 5.4.2) is enabled, the hidden layers will be automatically set. If you disable the automatic options, you can add and remove layers, select the activation function and the number of nodes for each layer (see Figure 5.15). You can add a minimum of one layer up to 32 layers.

---


\(^{11}\)http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf
5.4.7.2 Learn Residuals

If learning residuals is enabled, it will cause alternate layers to learn a representation of the residuals for a given layer rather than the layer itself, by introducing shortcut connections. In other words, residual networks tweak the mathematical formula of the typical layer’s equation to include the inputs of a lower layer in a node of a higher layer. Residual learning has proved to be very successful for image recognition as described in this paper\(^\text{12}\).

If one of the automatic optimization options (see Subsection 5.4.2) is enabled, the residual learning will be automatically set. If you disable the automatic options, you can choose to include or exclude it (see Figure 5.16).

5.4.7.3 Batch Normalization

If batch normalization is enabled, it will cause the outputs of a network to be normalized before being passed to the activation function, as described in this paper\(^\text{13}\). This will introduce extra parameters in each layer (the mean, variance, and scale of the layer), and will significantly slow down training.

If one of the automatic optimization options (see Subsection 5.4.2) is enabled, the batch normalization will be automatically set. If you disable the automatic options, you can choose to include or exclude it (see Figure 5.17).

---

\(^{12}\)https://arxiv.org/abs/1512.03385  
\(^{13}\)https://arxiv.org/abs/1502.03167
5.4.7.4 Tree Embedding

If **tree embedding** is enabled, the network will learn a tree-based representation of the data as engineered features along with the raw features, essentially by learning trees over slices of the input space and a small amount of the training data. The theory is that these engineered features will linearize obvious non-linear dependencies before training begins accelerating the learning process.

If one of the automatic optimization options (see Subsection 5.4.2) is enabled, the tree embedding will be automatically set. If you **disable the automatic options**, you can choose to include or exclude it (see Figure 5.18).
To learn more about the residual learning, the batch normalization and the tree embedding, read this [blog post](https://blog.bigml.com/2017/10/04/deepnets-behind-the-scenes/).

### 5.4.8 Algorithm

BigML deepnets allow you to select different gradient descent algorithms to optimize the network weights in order to minimize the loss function. These algorithms have some specific parameters explained in Subsection 5.4.8.1 and also some common ones such as the learning rate, the dropout rate, and the seed described in Subsection 5.4.8.1, Subsection 5.4.8.3, and Subsection 5.4.8.4, respectively.

If one of the automatic optimization options (see Subsection 5.4.2) is enabled, the algorithm parameters will be automatically set. If you disable the automatic options, you can configure them.

#### 5.4.8.1 Gradient Descent Algorithm

The most widely used algorithm during the neural networks training is the [gradient descent algorithm](https://en.wikipedia.org/wiki/Gradient_descent). This optimization algorithm is used to minimize the loss function. Although the gradient descent is the most important and popular technique used to train neural networks, it presents some problems associated such as converging to a sub-optimal local minima or setting a proper learning rate (not too small to avoid slow convergence and not too big to avoid divergence). To further optimize the gradient descent, BigML offers several optimization algorithms:

- **Momentum**: this method helps accelerate the gradient descent in the right direction dampening the oscillations until convergence. Therefore, it usually leads to faster and stable converge by reducing the unnecessary parameter updates. The configurable parameters for this algorithm include:

  - **Momentum**: higher values accelerate the gradient descent.

---


15 [https://en.wikipedia.org/wiki/Gradient_descent](https://en.wikipedia.org/wiki/Gradient_descent)
However, this method does not solve the problem that the final performance heavily depends on the selected learning rate.

- **Adagrad**: is an adaptative learning method. Essentially, it adapts the learning rate to each parameter making big updates for infrequent parameters and small updates for frequent parameters. It solves the problem of selecting a unique learning rate since it can take a default rate and then adapt it for each parameter. This method works very well with sparse data. The configurable parameters for this algorithm include:
  
  - **Initial Accumulator Value**: this is the initial value for the gradient accumulator.
The problem of Adagrad is that the learning rate tends to decay to a very small number so the learning stops. RMSProp tries to solve this problem.

- **RMSProp**: is another adaptative learning method that can be considered an extension of Adagrad and tries to solve the problem of decaying rates. The configurable parameters for this algorithm include:
  - **Momentum**: higher values accelerate the gradient descent.
  - **Decay**: the speed to decay the moving average.
  - **Epsilon**: a parameter to avoid numeric precision problems.
• **Adam**: (Adaptive Moment Estimation) is another adaptive method that computes adaptive learning rates for each parameter like Adagrad and it also solves the problem about decaying rates like RMSProp. Moreover, it keeps an exponentially decaying average of past gradients like Momentum. Adam usually works well compared to other algorithms as it converges fast and it solves the problems that other algorithms may have. The configurable parameters for this algorithm include:
  
  – **Beta1**: decay rate for the first moment estimate (the mean).
  – **Beta2**: decay rate for the second moment estimate (the variance).
  – **Epsilon**: a parameter to avoid numeric precision problems.
• **FTRL**: it also adapts the learning rate by slowing the learning rate per parameter. The configurable parameters for this algorithm include:

  – **Regularization**: the regularization factor to avoid overfitting, i.e., tailoring the model to the training data at the expense of generalization. You can choose between **L1** or **L2** regularization.

  – **Strength**: is the inverse of the regularization strength, so higher values indicate less regularization. It must be a positive integer greater than 0. Too high values for strength will make the algorithm perfectly fit the training data boundaries. Too low values for strength will result in vague decision boundaries not following the data patterns.

  – **Learning rate power**: the learning rate power for the FTRL algorithm.

  – **Initial Accumulator Value**: this is the initial value for the gradient accumulator.
In summary, if your data is sparse, some of the adaptative algorithms may perform better. Adagrad, RMSProp and Adam are quite similar and perform well for similar use cases. However, Adam is the one that usually outperforms the rest due to its bias correction.

Regarding the algorithm-specific parameters (momentum, beta1 and beta2, accumulator values, learning rate power, etc.), they all offer similar ways of controlling how much gradient descent remembers previous iterations and uses those to inform the current gradient step. Tuning those parameters have a similar impact: too high values for this sort of correction will send the search zooming off in the wrong direction; too low values will result in the same problems as vanilla gradient descent (overfitting and getting stuck in a local minima). If these parameters are set just right, they improve the speed at which the algorithm converges, and helps it to avoid local minima.

For all these parameters, though, the most important rule is not to hand-tune and iterate them unless you have a specific reason to do it. The best values for them depend on your data, the topology of your network, and the random conditions you start in. Hence, the best option if you are not very experienced with neural networks is to use one of the BigML optimization options (see Subsection 5.4.2) which will find the best configuration for your network automatically.

To learn more about the optimization algorithms please refer to this article

5.4.8.2 Learning Rate

The learning rate, also known as the gradient step, controls how aggressively the gradient descent algorithm fits the training data. You can set values greater than 0% and smaller than 100%. Larger values will prevent overfitting, but smaller values generally work better (usually 1% or lower), although it usually takes longer to train the deepnet. As a general rule, you want to find a learning rate that is low enough so the network converges to a satisfying solution, but high enough to reduce as much as possible the training time.

---

If one of the automatic optimization options (see Subsection 5.4.2) is enabled, the learning rate will be automatically set. If you disable the automatic options, you can select a value for the learning rate (see Figure 5.24).

5.4.8.3 Dropout Rate

The dropout mechanism consists of randomly drop nodes (and their connections) from the network at training time. This prevents nodes from co-adapting so it is an effective method to control overfitting. The dropout rate is the proportion of nodes dropped from the network during training.

If one of the automatic optimization options (see Subsection 5.4.2) is enabled, the dropout rate will be automatically set. If you disable the automatic options, you can select a value for the dropout rate (see Figure 5.25).
5.4.8.4 Seed

The random seed controlling the ordering of training data, the initial network weights, and the behavior of dropout during training. If the automatic network search option is not enabled, by setting the same seed you can get repeatable deepnets using the same dataset.
5.4.9 Weights

It is not unusual for a dataset to have some categories that are common and others very rare. For example, in datasets used to predict fraud, usually fraudulent transactions are very scarce compared to regular ones. When this happens, models tend to predict the most frequent values simply because the overall model's performance metrics improve with that approach. However, in cases such as fraud prediction, you may be more interested in predicting rare values rather than successfully predicting frequent ones. In that case, you may want to assign more weight to the scarce instances so they are equivalent to the abundant ones.

BigML provides two different options to assign specific weight to your instances, balance objective and objective weights explained in the following sections.

If one of the automatic optimization options (see Subsection 5.4.2) is enabled, the weights will be automatically set. If you disable the automatic options, you can configure the weights of your dataset instances (see Figure 5.27).
5.4.9.1 Balance Objective

When you set the balance objective weight, BigML automatically balances the classes of the objective field by assigning a higher weight to the less frequent classes, with the most frequent class always having a weight of 1. For example, take the following frequencies for each class:

\[ \text{False, 2000; True, 50} \]

By enabling the Balance objective option, BigML will automatically apply the following weights:

\[ \text{False, 1; True, 40} \]

In this example, the class “True” is getting forty times more weight as it is forty times less frequent than the most abundant class.

5.4.9.2 Objective Weights

The objective weights option allows you to manually set a specific weight for each class of the objective field. BigML oversamples your weighted instances replicating them as many times as the weight establishes. If you do not list a class, it is assumed to have a weight of 1. Weights of 0 are also valid, but if all classes have a weight of 0, the deepnet creation will produce an error.

This option can be combined with the Weight field (see Subsection 5.4.9.3).

5.4.9.3 Weight field

The Weight Field option allows you to assign individual weights to each instance by choosing a special weight field. It can be used for both regression and classification deepnets. The selected field must be numeric and it must not contain any missing values. The weight field will be excluded from the input fields when building the ensemble. You can select an existing field in your dataset or you may create a new one in order to assign customized weights.

For deepnets, the weight field modifies the loss function to include the instance weight. The outcome is similar to the oversampling technique.
5.4.10 Sampling Options

Sometimes you do not need all the data contained in your dataset to build your deepnet. If you have a very large dataset, sampling may be a good way of getting faster results. BigML allows you to sample your dataset before creating the deepnet, so you do not need to create a separate dataset first. You can find a detailed explanation of the sampling parameters available in the following subsections. (See Figure 5.28.)

5.4.10.1 Rate

The rate is the proportion of instances to include in your sample. Set any value between 0% and 100%. Defaults to 100%.

5.4.10.2 Range

Specifies a subset of instances from which to sample, e.g., choose from instance 1 until 200. The rate you set will be computed over the range configured. This option may be useful when you have temporal data, and you want to train your deepnet with historical data, and test it with the most recent one to check if it can predict based on time.

5.4.10.3 Sampling

By default, BigML selects your instances for the sample by using a random number generator, which means two samples from the same dataset will likely be different even when using the same rates and row ranges. If you choose deterministic sampling, the random-number generator will always use the same seed, thus producing repeatable results. This lets you work with identical samples from the same dataset.

5.4.10.4 Replacement

Sampling with replacement allows a single instance to be selected multiple times. Sampling without replacement ensures that each instance cannot be selected more than once. By default, BigML generates samples without replacement.

5.4.10.5 Out of bag

This argument will create a sample containing only out-of-bag instances for the currently defined rate, so the final total number of instances for your sample will be one minus the rate configured for your sample (when replacement is false). This can be useful for splitting a dataset into training and testing subsets. It is only selectable when a sample rate is less than 100%.
5.4.11 Creating Deepnets with Configured Options

After finishing the configuration of your options, you can change the default deepnet name in the editable text box. Then you can click on the Create deepnet button to create the new deepnet, or reset the configuration by clicking on the Reset button.
5.4.12 API Request Preview

The **API Request Preview** button is in the middle on the bottom of the configuration panel, next to the **Reset** button (See (Figure 5.29)). This is to show how to create the deepnet programmatically: the endpoint of the REST API call and the JSON that specifies the arguments configured in the panel. Please see (Figure 5.30) below:

There are options on the upper right to either export the JSON or copy it to clipboard. On the bottom there is a link to the API documentation for deepnets, in case you need to check any of the possible values or want to extend your knowledge in the use of the API to automate your workflows.
5.5 Visualizing Deepnets

If the dataset for creating a deepnet does not contain images, you can analyze your results with BigML Partial Dependence Plot (PDP) after the deepnet is created. If the dataset for creating a deepnet contains images, you can analyze your results with BigML Image Deepnet Page. In either case, you can also inspect field importances.

5.5.1 Partial Dependence Plot

Partial Dependence Plot (PDP) is a heatmap chart for examining the impact of the input fields on the objective field.

The PDP view is composed of three main parts: the CHART itself, the PREDICTION legend and the INPUT FIELDS form. (See Figure 5.31.) You can find a detailed explanation of each one below.

• The CHART allows you to view the impact of two input fields on the objective class predictions remaining the rest of input field values constant. You can select any categorical or numeric input field for each axis and the class probabilities are represented in different colors in the heatmap. You can switch the axis by clicking on the option on top of the chart area. (See Figure 5.32).

You can find the values for the fields in the axis in the grey area next to the selector. Freeze the view by pressing `Shift` and release it again by pressing `Escape` from your keyboard. When the view is frozen, an edition icon will appear in this grey area so you can edit the axis values and obtain the prediction for another preferred value. (See Figure 5.32).
**Figure 5.32: Deepnet chart**

- The PREDICTION legend allows you to visualize the **objective field classes** (classification deepnets) or the predicted **value** (regression deepnets). For classification, each class is represented by a color, the main probability color bar at the top is the probability for the predicted class. By default, colors are **shaded** according to the prediction range shown in the chart area. That way, smaller differences in predictions are easier to perceive. However, you can choose to see the color shading according to the total range of possible values for the objective field by clicking on the icon next to the prediction bar **Total**. (See **Figure 5.33**.) You can also select to see only one of the classes using the class selector at the bottom of the legend.
Figure 5.33: Prediction legend

Again, freeze this view by pressing \textit{Shift}, and release it again by pressing \textit{Escape} from your keyboard.

- Below the chart legend, you can find the INPUT FIELDS form. (See Figure 5.34.) You can configure the values for any \texttt{numeric}, \texttt{categorical}, \texttt{text} or \texttt{items} field. By changing their values, you can see the predictions changing in real-time. You can also enable or disable the input fields, so they will be treated as missing values. You can sort the fields by their importance to predict the objective field. (See \textit{Subsection 5.4.4}.)
Moreover, the chart includes a reset option for your input fields values, and an export option to download your chart in PNG format explained below:

- After selecting the fields for the axis or configuring the input fields values, you can set them again to the default view by clicking the **reset** icon highlighted in Figure 5.35.
Figure 5.35: Reset the values for the input fields

- You can also export your chart in PNG format with or without legends. Freeze the view by pressing \texttt{Shift} from your keyboard and export the chart to get the class percentages in the legend. Release the view by pressing \texttt{Escape}. 
Note: there are some limitations in the number of classes of the objective field and the number of input fields to visualize your deepnet in the chart (explained in Subsection 5.8.1).

5.5.2 Image Deepnet Page

As stated in the previous section, when the dataset used to create a deepnet contains images, the deepnet created will be a convolutional neural network (CNN). See the explanation in Subsection 5.2.1.

A CNN deepnet uses images (i.e. raw image pixels) as input fields. If there are image feature fields that were extracted from those same images in the dataset, they are ignored during CNN training. After a CNN deepnet is created, you can analyze the results with BigML Image Deepnet Page.

On the top row of the Image Deepnet Page are applicable parameters of the deepnet, which may include its hidden layer number, algorithm, optimization option. It also lists the objective field of the deepnet and its number of instances.

Below the top row, the view of the Image Deepnet Page shows the performance of the deepnet on a set of sampled instances. This set is called holdout set, which is used for validation during deepnet training.

Depending on the objective field, the Image Deepnet Page has two variations.

5.5.2.1 Image Deepnet Page - Classification

If the objective field is categorical, the Image Deepnet Page is the classification variation.

Below the top row, the view is composed of three main sections: the Image Results, the Performance Panel and the Class List. See Figure 5.37.
As stated above, the Image Deepnet Page mainly shows the performance of the deepnet on a set of sampled instances, called the holdout set. The number of the instances in the holdout set is at most 20% of the total instances in the dataset, or 1024, whichever is smaller.

- The Performance Panel shows the overall performance of the deepnet on the holdout set. See Figure 5.38. This is by default. However when a class is selected in the Class List, the Performance Panel shows the performance for that specific class.
The performance metrics, either correctly predicted or incorrectly predicted, are provided as both percentage and count. For instance, when the count is $893/1020$ as shown in Figure 5.38, that means 893 images were predicted correctly in the holdout set of 1020.

The probability slider allows users to filter the results by selecting the range of probabilities for each image classification. This controls how many images are shown in the Image Results section.

- The Image Results section shows two subsections, with each a paginable list of images from the holdout set, Correctly predicted and Incorrectly predicted. In each list, every image has a caption which shows its predicted class, true class and probability. In the subsection of Correctly predicted, because a predicted class is the same as its true class, only one is shown in the caption. In the Incorrectly predicted subsection, both predicted class and true class are shown for each image. The length of the solid color in a probability bar is proportional to its value. See Figure 5.39.
On the top right of each subsection are the performance metrics by percentage and count. The list of images are paginable by using the pagination arrows at the bottom of each subsection. Each page shows up to 6 images, scaled to fit the area.

When users mouse-over an image, a popup box would show prediction result of the image: predicted class, true class and probability. In the example shown in Figure 5.39, the popup box is red in background, signaling an incorrect prediction, which its predicted class is *frog* while its true class is *bird* and the probability is 70.08%.

- The Class List shows all the classes in the holdout set, sorted by their numbers of occurrences. In other words, the list is ranked by class popularity. See Figure 5.40. The recall rate of each class is displayed by percentage and count at the right side of class bar. Recall is defined as the number of correctly predicted images for the class divided by the total number of images for the class. The length of the green color bar in proportion to the full class bar corresponds to the value of the recall of the class.
The Class List is scrollable when there are not enough space to show all of them together. There is also a download icon in the section heading that users can use to download the class list as a CSV.

There are two controls that can change the number of images to show in Image Results section. One is the probability slider in the Performance Panel. Another is a class bar in the Class List.

- By default, the Image Results section shows all the images in the holdout set, both correctly predicted and incorrectly predicted. The probability slider displays the lower and upper ends of the probabilities associated with all the classification results in the set. Either end of the slider can be changed by dragging the respective knob, then the Image Results section will only show images with the probabilities within the range of the slider. The images having the probabilities outside of the range will be filtered out.
As in Figure 5.41, the lower end of the probability slider was changed to 64.0%. Now any image with the classification probability lower than 64.0% won’t show up in the Image Results section, in both correctly predicted and incorrectly predicted. The performance metrics were changed as well, from 893/1020 to 856/928, reflecting the fewer number of images shown.

The probability slider can be reset to the default by dragging its knobs to include all possible probabilities, or by pressing the reset icon in the heading of the Performance Panel.

Both ends of the probability slider are editable text input boxes, so users can enter precise numbers if so desired.

- In the Class List, one class can be selected by clicking on its class bar. When this happens, the Image Results section will only show images of that class. The Correctly Predicted subsection shows all images whose true class is that class, and whose predicted class is that class too. The Incorrectly Predicted subsection shows all images whose true class is that class selected, but whose predicted class is a different class. The performance metrics are changed accordingly.
In Figure 5.42 above, the class truck is selected, as shown by its darkened background. Now the Images Results section shows 89 images in the correctly predicted subsection, all truck, and 13 images in the incorrectly predicted subsection, all truck but all predicted as something else.

Also in Figure 5.42, note that in the Performance Panel, not only the performance metrics are changed accordingly, but also the title becomes “Performance for truck” instead of “Performance overall”.

To reset the Image Results to include all classes, use the reset icon in the Performance Panel heading.

- The class selection and the probability slider can be combined to show only images of one class which has a selected range of probabilities.

5.5.2.2 Image Deepnet Page - Regression

If the objective field is numeric, the Image Deepnet Page is the regression variation.
Below the top row, the view is composed of two main sections: the Image Results and the Performance Panel. See Figure 5.43. The example deepnet in the figures of this section was created from a dataset for estimating the number of penguins in images, with its numeric objective field “count”.

As stated above, the Image Deepnet Page mainly shows the performance of the deepnet on a set of sampled instances, called the holdout set. The number of the instances in the holdout set is at most 20% of the total instances in the dataset, or 1024, whichever is smaller.

- The performance panel shows the performance of the deepnet on the holdout set, in terms of error percentages. All numeric predictions have errors. A value in percentage, called split, is used to divide all predictions in the holdout set into two groups. The default split is 10%, which can be changed by users.
The size of the holdout set is shown right below the panel header. It also shows how many instances are displayed, as this can be changed by a filter.

As seen in Figure 5.44, the split is at 10%, by default, so the images are divided into two groups, one “Predicted within 10.00% error”, another “Predicted outside of 10.00% error”. Both percentages and fractions are employed to show the relative sizes of the two groups in regard to the split. In the example, for instance, there are 2 images predicted within 10% error, so for this group, it’s 6.67% or \(\frac{2}{30}\).

“Average percent error” is a metric showing the average error a prediction has in each group. It is the sum of the absolute values of the percentage errors of all instances, divided by the number of instances.

There are two controls that can change how the Image Results section appears.

1. The DISPLAY ERROR slider allows users to filter the results by selecting the range of errors for each prediction. This controls how many images are shown in the Image Results section.

By default, the Image Results section shows all the images in the holdout set, which are divided into two groups by the split. The DISPLAY ERROR slider shows the greatest negative error at its left end, and the greatest positive error at its right end. **Note: When a prediction is more than the true value, it produces a positive error. Conversely, a negative error.** Either end of the slider can be changed by dragging the respective knob, then the Image Results section will only show images with the predictions within the range of the slider. The images having the prediction errors outside of the range will be filtered out.
As in Figure 5.45, the lower end of the DISPLAY ERROR slider was changed to -15.0 and the upper end to 0.5. Now any image with the prediction error outside of the range, i.e. either less than -15.0 or greater than 0.5, doesn’t show up in the Image Results section. There are 21 images displayed, instead of 30, as indicated inside the parentheses below the panel heading. The relative sizes of the split groups are changed as well, from 2/30 and 28/30 to 2/21 and 19/21, respectively, reflecting the fewer number of images displayed.

The DISPLAY ERROR slider can be reset to the default by dragging its knobs to include all possible errors, or by pressing the reset icon in the heading of the Performance Panel.

Both ends of the DISPLAY ERROR slider are editable text input boxes, so users can enter precise numbers if so desired.

2. The SPLIT ERROR slider allows users to select the split, which is the error percentage used to divide the Image Results section into two groups: one having prediction errors smaller than the split, and another having prediction errors bigger than the split.
As seen in Figure 5.46, moving the knob on the SPLIT ERROR slider changes the split value, which is shown above the slider, anywhere between 0% and the biggest possible error percentage produced by the holdout set. In addition, below the slider shows the total number of instances displayed, on the left of the slider is the number of instances in the group with prediction errors smaller than the split, and on the right the number of instances in the group with prediction errors bigger than the split. In the parentheses are the percentages of the group size in respect to the total instances, respectively. As the split value changes, both group sizes (and percentages) on the left and right sides change accordingly. This gives a direct visualization of how the split affects the partition of the Image Results section.

The SPLIT ERROR slider can be reset to the default value, 10%, by pressing the reset icon in the heading of the Performance Panel.

The split value above the SPLIT ERROR slider is an editable text input box, so users can enter a precise split value if so desired.

The total number of instances below the SPLIT ERROR slider may be affected by the DISPLAY ERROR slider.

Figure 5.46: Image Deepnet Page Regression Split Error Slider
The Image Results section shows two subsections, which represent two groups divided by the split value. The default split value is 10%. One group is called “Predicted within 10% error”, which has the images with prediction errors less than 10%. Another group is called “Predicted outside of 10% error”, which has the images with prediction errors greater than 10%. When the split value is changed by the SPLIT ERROR slider in the Performance Panel, the subsection titles and images change accordingly.

As seen in Figure 5.47, each subsection in the Image Results section is a paginable list of images from the holdout set. In each list, every image has a caption which shows its predicted value, true value, error and error percentage.

On the top right of each subsection is the relative size of the subsection by percentage and count, with respect to the total number of instances displayed. The list of images are paginable by using the pagination arrows at the bottom of each subsection. Each page shows up to 6 images, scaled to fit the area.

When users mouse-over an image, a popup box would show prediction result of the image: pre-
dicted value, true value and the error which is defined as (predicted - true). In the example shown in Figure 5.47, the popup box is red in background, signaling a prediction which error is greater than the split value, with its predicted value as 2.90374, its true value 6.00000 and the error -3.09626.

The total number of images displayed in the Image Results section may be affected by the DISPLAY ERROR slider in the Performance Panel.

5.5.3 Summary Report

5.5.3.1 Field Importances

The field importances for deepnets provide a measure of how important an input field is relative to the others to predict the objective field. Each field importance is normalized to take values between 0% and 100%. All field importances should sum 100%. You can access them by clicking in the Summary Report option shown in Figure 5.48. You can also export the field importances in PNG, CSV and JSON format.

Deepnet field importances are based on Shapley values. For more information, please refer to this research paper: A Unified Approach to Interpreting Model Predictions [3].

5.5.3.2 Summary

If you created your deepnet using the Automatic Network Search option (see Subsection 5.4.2), you will be able to see a tab called “Summary” next to the field importances tab (see Figure 5.49). Find here the configuration of each of the networks composing the deepnet in JSON format. You can find all the parameters explained in Section 5.4 per network.
5.6 Deepnet Predictions

5.6.1 Introduction

The ultimate goal in building a deepnet is being able to make predictions with it. In BigML, you can make predictions for single instances or for many instances in batch. Each prediction comes with a measure indicating the prediction confidence. For regression problems, the expected error is provided along with the predicted value, while for classification problems the vector of probabilities per class is returned (a percentage ranging from 0% up to 100%).

The predictions tab in the main menu of the BigML Dashboard is where all your saved predictions are listed. (See Figure 5.50.) You can search your predictions by name clicking on the search option on the top menu. In the predictions list view, you can see, for each prediction, the deepnet icon used for the prediction, the Name of the prediction, the Objective (objective field name), the Prediction (the prediction result), and the Age (time since the prediction was created).
When you first create an account at BigML, or every time that you start a new project, your list of predictions will be empty. (See Figure 5.51.)

Deepnet predictions are saved under the CLASSIFICATION & REGRESSION option in the menu (see Figure 5.52.)

Select the list for your single instances predictions or your batch predictions by clicking on the corresponding icons. (See Figure 5.53 and Figure 5.54.)
5.6.2 Creating Deepnet Predictions

BigML provides two options to predict with your deepnets explained in the following subsections:

- **PREDICT**: to predict one single instance
- **BATCH PREDICTION**: to predict multiple instances in batch.

5.6.2.1 Predict

BigML allows you to quickly make predictions for single instances by providing a form containing the input fields used by the deepnet, so you can easily set the values and get an immediate response.

Follow these steps to create a single prediction:

1. Click **PREDICT** in the deepnet **1-click action menu**. (See Figure 5.55.)

Alternatively, click **PREDICT** in the **pop up menu** in the list view. (See Figure 5.56.)
2. You will be redirected to the **prediction form** where you will find all the fields used by the deepnet as input fields. (See Figure 5.57.)

3. **Select the fields** to be used for your prediction, set the **input values** for your selected fields and click **Predict** (Figure 5.58.) Non-selected fields will be considered as missing values during the
prediction.

![Deepnet predictions form](image)

**Figure 5.58: Deepnet predictions form**

4. Get the prediction at the top of the view along with the rest of class probabilities. (See Figure 5.59.) BigML predictions are synchronous, i.e., when you send the input data, you get an immediate response.
5. Hide or display the histogram view containing the rest of your **class probabilities** by clicking on the icon highlighted in Figure 5.60. You can download all the probabilities in PNG format, in CSV or JSON file by clicking on the corresponding icons. (See Subsection 5.6.4.1.)

6. Your prediction is automatically saved and you can find it in the predictions list view. For **regression** deepnets, the process is the same, but instead of the predicted classes and the probabilities you get a numeric value for the objective field along with an expected error as the certainty
measure.

**Note:** this option is only available from the BigML Dashboard for deepnets with less than 100 fields. If you want to perform single instance predictions for a higher number of fields, use the [BigML API](https://bigml.com/api/predictions).

### 5.6.2.2 Deepnet Prediction with Images

Deepnet created from a dataset containing images is a convolutional neural network (CNN) (See Subsection 5.2.1). Because BigML treats images as an input type just like any other data types such as numeric and text, creating predictions using CNN is the same as other deepnets. Everything stated in the previous section still applies. The only thing different is input fields of images.

Follow these steps to create a single prediction:

1. Click PREDICT in the deepnet **1-click action menu**. See Figure 5.61.

![Figure 5.61: Predict with images using the 1-click action menu](image)

Alternatively, click PREDICT in the **pop-up menu** in the deepnet list view. See Figure 5.62.

![Figure 5.62: Predict using the pop-up menu](image)

---

17[https://bigml.com/api/predictions](https://bigml.com/api/predictions)
2. You will be directed to the **prediction form** where you will find all the input fields used by the deepnet. One of them is the image field. As seen in Figure 5.63, click on the input field box to select an image. Because this is a single prediction, an image is input by using a single image source. Clicking on the input box, single image sources available will be in the dropdown list. You can also use the search box to locate specific ones.

![Image of prediction form with image input field](image.png)

**Figure 5.63:** Select a single image source in the image input field

3. Oftentimes single image sources were used for creating a composite source, they become component sources of the composite source. Or an image was uploaded as a part of an archive file (zip/tar) which created a composite source. In those cases, the composite source will be shown in the dropdown list, along with an icon “List components”. In the example in Figure 5.64, **predict-images.zip** is a composite source, click on the icon to show its component sources.
Figure 5.64: List the components of a composite source

After the component sources of the composite are listed, scroll the dropdown list to find the desired one, then click to select it, as shown in Figure 5.65.
4. Once an image is selected, click on the green button Predict to create a prediction for the image.
As shown in Figure 5.66, after the new prediction is created, the predicted class is at the top of the form along with its probability.

Below the prediction is a histogram representing all the classes of the objective field with their respective predicted probabilities. Show or hide this histogram by clicking on the icon on the right of the predicted class (Figure 5.66). Up to seven different classes can be shown in the histogram together. When there are more than seven classes, the view can be scrolled by clicking on the arrow icons on the right. See Figure 5.67. If desired, click on the sort icon on the left to sort the classes in the histogram by probability.

The histogram can be exported in PNG format, in a CSV file, or in a JSON file by clicking on the corresponding icons on the top right.

5. In addition to images, deepnets may use other fields, which will be in the prediction form too. As shown in Figure 5.68, all the fields can be selected, and their input values be set by dragging the knobs on the sliders or by entering precise values in their input boxes.
5.6.2.3 Batch Predictions

BigML batch predictions allow you to make simultaneous predictions for multiple instances. All you need is the deepnet you want to use to make predictions and a dataset containing the instances you want to predict. BigML will create a prediction for each instance in the dataset.

Follow these steps to create a batch prediction:

1. Click on **BATCH PREDICTION** option under the deepnet 1-click action menu (Figure 5.69)

Alternatively, click on **CREATE BATCH PREDICTION** in the pop up menu of the list view (Figure 5.70).
2. Select the dataset containing all the instances you want to predict. The instances should contain the input values for the fields used by the deepnet as input fields. From this view you can also select another deepnet from the selector. (See Figure 5.71.)

3. After the deepnet and the dataset are selected, the batch prediction configuration options will appear along with a preview of the prediction output (a CSV file). (See Figure 5.72.) The default output format includes all your prediction dataset fields and adds an extra column with the class predicted. See Subsection 5.6.3 for a detailed explanation of all configuration options.
4. By default, BigML generates an output dataset with your batch predictions that you can later find in your datasets section in the BigML Dashboard. This option is active by default but you can deactivate it by clicking on the icon shown in Figure 5.73.
5. After you configure your batch prediction, click on the green button [Predict] to generate your batch prediction. (See Figure 5.74.)
6. When the batch prediction is created, you will be able to download the CSV file containing all your dataset instances along with a prediction for each one of them. (See Figure 5.75.)

![Figure 5.75: Download batch prediction CSV file](image)

7. If you did not disable the option to create a dataset explained in step 4, you will also be able to access the output dataset from the batch prediction view. (See Figure 5.76.)

![Figure 5.76: Batch prediction output dataset](image)

### 5.6.2.3.1 Batch Predictions with Images

Deepnet created from a dataset containing images is a convolutionary neural network (CNN) (See Subsection 5.2.1). Because BigML treats images as an input type just like any other data types such as numeric and text, creating batch predictions using CNN is the same as other deepnets.

The input of a batch prediction is a dataset. The dataset contains multiple images which was created by a composite source.
As shown in Figure 5.77, the input for the deepnet batch prediction is selected as `predict-images`, which is a dataset consisting of six images.

Everything stated earlier in current section (Subsection 5.6.2.3) applies.

### 5.6.3 Configuring Deepnet Predictions

BigML provides several options to configure your batch predictions such as setting a probability threshold (Subsection 5.6.3.1), default values for your missing numeric values (see Subsection 5.6.3.2), fields mapping (see Subsection 5.6.3.3), and output file settings (see Subsection 5.6.3.4.)

#### 5.6.3.1 Probability threshold

Probability thresholds usually makes sense when you want to minimize false positives at the cost of false negatives. The positive class will be predicted if its probability is greater than the given threshold, otherwise the following class with greater probability will be predicted instead. This option is only available for classification deepnets.

To configure a threshold for your batch prediction follow these steps:

1. Select the positive class, i.e., the class for which you want to apply the threshold (Figure 5.78).
Figure 5.78: Select the positive class

2. Set a probability threshold using the slider shown in Figure 5.79.

Figure 5.79: Set a probability threshold

3. Click Predict. If the positive class probability is greater than the given threshold, it will be predicted, otherwise the following class with greater probability will be predicted instead.
You can also find the same options to set a threshold for batch predictions under the CONFIGURE panel (see Figure 5.81).

Figure 5.80: Save the prediction

Figure 5.81: Configure a probability threshold for batch predictions
5.6.3.2 Default Numeric Value

If the dataset used to make the batch prediction contains instances with missing values for the numeric fields, the prediction will not be computed for them, unless you built the deepnet enabling the Missing numerics parameter (see Subsection 5.4.4).

By using the Default numeric value before creating your batch prediction, you can easily replace all the missing numeric values by the field’s Mean, Median, Maximum, Minimum or by Zero. (See Figure 5.82.)

![Figure 5.82: Configure Default numeric value for batch prediction](image)

5.6.3.3 Fields Mapping

You can specify which input fields of the deepnet match with the fields in the dataset containing the instances you want to predict. BigML automatically matches fields by name, but you can also set an automatic match by field ID by clicking on the green switcher. Additionally, you can manually search for fields or remove them from the Dataset fields column if you do not want them to be considered during the batch prediction. (See Figure 5.83.)
Note: Fields mapping from the BigML Dashboard is limited to 200 fields. For batch predictions with a higher number of fields, map your fields using the BigML API\(^\text{18}\).

### 5.6.3.4 Output Settings

Batch predictions return a CSV file containing all your instances and the final predictions. Tune the following settings to customize your prediction file (see Figure 5.84):

- **Separator**: this option allows you to choose the best separator for your output file columns. The default separator is the comma. You can also select the semicolon, the tab, or the space.

- **New line**: this option allows you to set the new line character to use as the line break in the generated csv file: “LF”, “CRLF”.

- **Output fields**: by clicking on the list icon next to the separator selector, you can include or exclude all your dataset fields from your output file. You can also individually select the fields you want to include or exclude using the multiple output fields selector. **Note**: a maximum of 100 fields can be displayed in this selector, but all your dataset fields will be included in the output file by default unless you exclude them.

- **Headers**: this option includes or excludes a first row in the output file (and in the output dataset) with the names of each column (input field names, prediction column name, probability column name, etc.). By default, BigML includes the headers.

- **Prediction column name**: customize the name for your predictions column. By default, BigML takes the name of the deepnet’s objective field.

- **Probability**: this option allows you to include an additional column with the probability for the predicted class. By default it is not included in your output file. For regression deepnets, you will find the expected error instead of the probability.

- **Probability column name**: customize the name for the probability column if you include it in the output file. BigML sets “probability” as the default name. For regression deepnets, you will find the expected error column name.

- **All class probabilities**: this includes all the probabilities of the objective field classes per instance. This option will add \( n \) extra columns, one by class in the objective field. This option does not exist for regression deepnets.

\(^{18}\)https://bigml.com/api/batchpredictions#bp_batch_prediction_arguments
5.6.4 Visualizing Deepnet Predictions

Deepnet predictions visualization changes depending if you are predicting one single instance (Subsection 5.6.4.1), or you are predicting multiple instances using the batch predictions option (Subsection 5.6.4.2).

5.6.4.1 Single Predictions

For single predictions, find the predicted class given the input fields values at the top of the form along with its probability. (See Figure 5.85.)

Below the prediction, there’s a histogram representing the rest of the objective field class probabilities. All the class probabilities must sum 100%. Show or hide this view by clicking on the icon highlighted in Figure 5.86. You can see up to seven different classes at the same time; if you have more than seven classes, you can see the others by clicking on the arrows icons. Export this view in PNG format, in a CSV file, or in a JSON file by clicking on the corresponding icons. (See Figure 5.86.)
For regression deepnets, instead of the class probabilities you will get the predicted numeric value for the objective field.

5.6.4.1.1 Prediction explanation

Prediction explanation helps understand why a deepnet makes a certain prediction. This is very useful in many applications, and the reasons behind a deepnet's prediction are often as important as the prediction itself.

BigML prediction explanation is based on Shapley values. For more information, please refer to this research paper: A Unified Approach to Interpreting Model Predictions [3].

For any classification or regression deepnet, you can request the explanation for the prediction by clicking the prediction explanation icon and then click Predict (see Figure 5.87).
Figure 5.87: Explain prediction

The prediction explanation represents the most important factors considered by the deepnet in a prediction given the input values. Each input value will yield an associated importance, as you can see Figure 5.88. The importances across all input fields should sum 100%.

Figure 5.88: Input field importances

For some input fields you will see a “+” icon next to the importance. This is because the importance may not be directly associated with the input value, i.e., it can be explained by other reasons. In the Figure 5.89 below, the importance of 36.58% for the field “Rating” is not explained by this field being equal to 90.00. Rather, it is because this field value is not missing.
The prediction explanation for deepnets is calculated using the results of over a thousand distinct predictions using random perturbations of the input data. For this reason, the calculation of the explanation may take some time to be computed.

**Note:** the input field importances in the prediction explanation are different from the overall field importances of the deepnet. A field can be very important for the deepnet but insignificant for a given prediction.

### 5.6.4.2 Batch Prediction

After creating your batch prediction, you get a CSV file and, optionally, an output dataset. Both outputs are explained in the following subsections.

#### 5.6.4.2.1 Output CSV file

The batch prediction generates a CSV file containing your predictions for each of your dataset instances in the last column. (See Figure 5.90.)
You can configure several options to **customize your CSV file**. You can find a detailed explanation of those options in Subsection 5.6.3.4.

See an output CSV file example in **Figure 5.91**. The column **class** in this example contains the final prediction (it is named by default as your deepnet's **objective field**). In this case we are predicting whether a person is a good or a bad candidate for holding a credit. This file has been configured to contain also the **probability** for each prediction.

<table>
<thead>
<tr>
<th>duration</th>
<th>age</th>
<th>amount</th>
<th>purpose</th>
<th>class</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>26</td>
<td>5433</td>
<td>used car</td>
<td>good</td>
<td>0.88785</td>
</tr>
<tr>
<td>36</td>
<td>42</td>
<td>8086</td>
<td>new car</td>
<td>bad</td>
<td>0.55526</td>
</tr>
<tr>
<td>24</td>
<td>26</td>
<td>1376</td>
<td>radio/tv</td>
<td>good</td>
<td>0.8385</td>
</tr>
<tr>
<td>48</td>
<td>31</td>
<td>6758</td>
<td>radio/tv</td>
<td>bad</td>
<td>0.73676</td>
</tr>
<tr>
<td>26</td>
<td>30</td>
<td>7966</td>
<td>used car</td>
<td>good</td>
<td>0.7201</td>
</tr>
<tr>
<td>12</td>
<td>42</td>
<td>2877</td>
<td>furniture/equipment</td>
<td>good</td>
<td>0.67644</td>
</tr>
<tr>
<td>36</td>
<td>30</td>
<td>4455</td>
<td>business</td>
<td>good</td>
<td>0.52227</td>
</tr>
<tr>
<td>18</td>
<td>32</td>
<td>1442</td>
<td>new car</td>
<td>bad</td>
<td>0.75488</td>
</tr>
<tr>
<td>9</td>
<td>22</td>
<td>276</td>
<td>new car</td>
<td>good</td>
<td>0.57819</td>
</tr>
</tbody>
</table>

**Figure 5.91:** An example of a deepnet batch prediction CSV file

### 5.6.4.2.2 Output Dataset

By default BigML automatically creates a dataset out of your batch prediction. You can disable this option by configuring your batch prediction. (See Subsection 5.6.3.4.) You will find the output dataset in your batch prediction view as shown in **Figure 5.92**.

In the output dataset, you can find an additional **field** (named by default as your deepnet's objective field) containing the **class predicted** for each one of your instances. If you configured your batch prediction to include the prediction **probabilities** and all **class probabilities** you will be able to find them in the last fields of your output dataset. (See **Figure 5.93**.)
5.6.5 Consuming Deepnet Predictions

You can fully use single and batch predictions via the BigML API and bindings. The following subsections explain both tools.

5.6.5.1 Using Deepnet Predictions via the BigML API

Deepnet predictions have full citizenship in the BigML API which allows you to programmatically create, configure, retrieve, list, update, and delete single and batch predictions.

In the example below, see how to create a single prediction using a deepnet and defining the input data once you have properly set the `BIGML_AUTH` environment variable to contain your authentication credentials:

```
curl "https://bigml.io/prediction?$BIGML_AUTH" \
  -X POST \n  -H 'content-type: application/json' \n  -d '{"deepnet": "deepnet/50650bdf3c19201b64000020", \n       "input_data": {"000001": 3, "000002":4.5, "000003":5}}'\n```

For more information on using predictions through the BigML API, please refer to the documentation:

5.6.5.2 Using Deepnet Predictions via BigML Bindings

You can also create, configure, retrieve, list, update, and delete single and batch predictions via BigML bindings which are libraries aimed to make it easier to use the BigML API from your language of choice.

---

19 https://bigml.com/api/predictions
BigML offers bindings in multiple languages including Python, Node.js, Java, Swift and Objective-C. See below an example to create a deepnet with the Python bindings.

```python
from bigml.api import BigML
api = BigML()
prediction = api.create_prediction(
    "deepnet/50650bdf3c19201b64000020",
    {"credit_amount": 5, "duration": 2.5})
```

For more information on BigML bindings, please refer to the bindings page[^20].

### 5.6.6 Descriptive Information

Each deepnet prediction has an associated **name**, **description**, **category**, and **tags**. You can find a brief description of each concept in the following subsections. The **MORE INFO** menu option displays a panel that provides editing options. (See Figure 5.94.)

![Figure 5.94: Deepnet prediction descriptive information](https://bigml.com/tools/bindings)

#### 5.6.6.1 Name

If you do not specify a **name** for your predictions, BigML assigns a default name depending on the type of predictions:

- **Single predictions**: the name always follows the structure "*<deepnet name>*".
- **Batch predictions**: BigML combines your prediction dataset name and the deepnet name: "*<deepnet name>* with *<dataset name>*".

Prediction names are displayed on the list and also on the top bar of a prediction view. Prediction names are indexed to be used in searches. Rename your predictions any time from the **MORE INFO** menu.

The name of a prediction cannot be longer than 256 characters. More than one prediction can have the same name even within the same project, but they will always have different identifiers.

[^20]: https://bigml.com/tools/bindings
5.6.6.2 Description

Each prediction also has a description that it is very useful for documenting your Machine Learning projects. Predictions take their description from the deepnet used to create them.

Descriptions can be written using plain text and also markdown\(^{21}\). BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 5.95.)

![Edit description](image)

Figure 5.95: Markdown editor for deepnet descriptions

Descriptions cannot be longer than 8192 characters.

5.6.6.3 Category

A category taken from the deepnet used to create it is associated with each prediction. Categories are useful to classify predictions according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

A prediction category must be one of the categories listed on table Table 5.1.

5.6.6.4 Tags

A prediction can also have a number of tags associated with it. These tags help to retrieve the prediction via the BigML API or to provide predictions with some extra information. Your prediction inherits the tags from the deepnet used to create it. Each tag is limited to a maximum of 128 characters. Each prediction can have up to 32 different tags.

5.6.7 Deepnet Predictions Privacy

The link displayed in the Privacy panel is the private URL of your prediction, so only a user logged into your account is able to see it. Neither single predictions nor batch predictions can be shared by using a secret link. (See Figure 5.96.)

\(^{21}\)https://en.wikipedia.org/wiki/Markdown
5.6.8 Moving Deepnet Predictions to Another Project

When you create a prediction, it will be assigned to the same project where the original deepnet is located. You cannot move predictions between projects as you do with other resources.

5.6.9 Stopping Deepnet Predictions

Single predictions are synchronous resources, so you cannot cancel them during the creation since you get the result immediately.

By contrast, batch predictions are asynchronous resources, so you can stop their creation before the task is finished. Use the DELETE BATCH PREDICTION option from the 1-click action menu (Figure 5.97) or from the pop up menu on the list view.

A modal window will be displayed asking you for confirmation. If you stop the prediction during its creation you won’t be able to resume the same task again, so if you want to create the same prediction you will have to start a new task.
5.6.10 Deleting Deepnet Predictions

You can **DELETE** your single or batch predictions from the predictions view, using the **1-click action menu** (see Figure 5.99) or using the **pop up menu** on the predictions list view (see Figure 5.100).

---

**Figure 5.98:** Deepnet delete prediction confirmation

**Figure 5.99:** Deepnet delete prediction from 1-click menu

**Figure 5.100:** Deepnet delete prediction from pop up menu
A modal window will be displayed asking you for confirmation. Once a prediction is deleted, it is permanently deleted, and there is no way you (or even the IT folks at BigML) can retrieve it.

Figure 5.101: Deepnet delete prediction confirmation

5.7 Consuming Deepnet

Similarly to other models in BigML, deepnets are white-boxed models, so you can download them and use them locally to make predictions. You can also create and consume your deepnet programmatically via the BigML API and bindings. The following subsections explain those three options.

5.7.1 Downloading Deepnet

You can download your deepnet in several programming languages including Python or Node.js. By downloading your deepnet you will be able to compute predictions locally, free of latency and at no cost. Click on the download icon in the top menu (see Figure 5.102), and select your preferred option (see Figure 5.103.)

Figure 5.102: Click download icon
5.7.2 Using Deepnets Via the BigML API

Deepnets have full citizenship in the BigML API which allows you to programmatically create, configure, retrieve, list, update, delete, and use them for predictions.

In the below example, see how to create a deepnet using an existing dataset once you have properly set the `BIGML_AUTH` environment variable to contain your authentication credentials:

```bash
curl "https://bigml.io/deepnet?$BIGML_AUTH" \
    -X POST \
    -H 'content-type: application/json' \
    -d '{"dataset": "dataset/50650bdf3c19201b64000020"}'
```

For more information on using deepnet through the BigML API, please refer to the documentation\(^\text{22}\).  

5.7.3 Using Deepnets Via the BigML Bindings

You can also create and use deepnets via BigML bindings which are libraries aimed to make it easier to use the BigML API from your language of choice. BigML offers bindings in multiple languages including Python, Node.js, Java, Swift and Objective-C. See below an example to create a deepnet with the Python bindings.

```python
from bigml.api import BigML
api = BigML()
deeplnets = api.create_deepnet(
    'dataset/57506c472275c1666b004b10', {"objective_field":"churn"})
```

For more information on BigML bindings, please refer to the bindings page\(^\text{23}\).

---

\(^{22}\)https://bigml.com/api/deepnets  
\(^{23}\)https://bigml.com/tools/bindings
5.8 Deepnet Limits

Some specific limits apply for your deepnets regarding your **dataset field characteristics** (see Subsection 5.8.0.1) and the **visualization**, i.e. to the deepnet Partial Dependence Plot, depending on the number of classes in the objective field and the number of input fields in your dataset (see Subsection 5.8.1).

**Note:** the visualization limits just affect to the visualization of the model, i.e., despite your dataset reach those limits, you can still creating the deepnet, evaluating it and using it to make predictions.

5.8.0.1 Field Limits

Deepnets, similarly to other BigML models, has the following limitations according to the type of field:

- **Classes**: a maximum number of 1,000 distinct classes per field is allowed.
- **Terms**: BigML can handle up to 1,000 terms in total. If multiple text fields are defined, then the token limit per field is evenly divided by the number of text fields evenly, e.g., a dataset with two text fields would result in 500 terms per text field. BigML selects those terms with most significant frequency, discarding both those that appear either too often or too infrequently. A maximum of 256 characters per term is allowed.
- **Items**: a maximum number of 10,000 distinct items per field is allowed.

5.8.1 PDP Limits

There are some circumstances under which your chart cannot be displayed:

- As the PDP only supports **numeric and categorical fields** for the axes, if your deepnet only contains text, or items fields, the PDP cannot be displayed.
- If your deepnet contains more than **100 fields** the top 100 fields will be included as input fields ranked by importance. The rest of fields will be excluded from the view.
- If your deepnet contains more than **200 categories** in the objective field, the PDP cannot be displayed.

5.9 Descriptive Information

Each deepnet has an associated **name, description, category**, and **tags**. The following subsections provide a brief description for each concept. In Figure 5.104, you can see the options the MORE INFO menu provides to edit them.
5.9.1 Deepnet Name

Each deepnet has a name that is displayed in the deepnet list view and also on the top bar of the deepnet view. Deepnet's names are indexed to be used in searches. When you create a deepnet, it gets a default name which is your dataset name. Change it using the MORE INFO menu option on the right corner of the deepnet view. The name of a deepnet cannot be longer than 256 characters. More than one deepnet can have the same name even within the same project, but they will always have different identifiers.

5.9.2 Deepnet Description

Each deepnet also has a description that it is very useful for documenting your Machine Learning projects. Deepnets take the description of the datasets used to create them by default.

Descriptions can be written using plain text and also markdown\(^{24}\). BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 5.105.)

\(^{24}\)https://en.wikipedia.org/wiki/Markdown
Descriptions cannot be longer than 8192 characters and can use almost any character.

5.9.3 Deepnet Category

A category, taken from the dataset used to create it, is associated with each deepnet. Categories are useful to classify deepnet according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

A deepnet category must be one of the 24 categories listed on Table 5.1.
### Table 5.1: Categories used to classify deepnet by BigML

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace and Defense</td>
</tr>
<tr>
<td>Automotive, Engineering and Manufacturing</td>
</tr>
<tr>
<td>Banking and Finance</td>
</tr>
<tr>
<td>Chemical and Pharmaceutical</td>
</tr>
<tr>
<td>Consumer and Retail</td>
</tr>
<tr>
<td>Demographics and Surveys</td>
</tr>
<tr>
<td>Energy, Oil and Gas</td>
</tr>
<tr>
<td>Fraud and Crime</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Higher Education and Scientific Research</td>
</tr>
<tr>
<td>Human Resources and Psychology</td>
</tr>
<tr>
<td>Insurance</td>
</tr>
<tr>
<td>Law and Order</td>
</tr>
<tr>
<td>Media, Marketing and Advertising</td>
</tr>
<tr>
<td>Miscellaneous</td>
</tr>
<tr>
<td>Physical, Earth and Life Sciences</td>
</tr>
<tr>
<td>Professional Services</td>
</tr>
<tr>
<td>Public Sector and Nonprofit</td>
</tr>
<tr>
<td>Sports and Games</td>
</tr>
<tr>
<td>Technology and Communications</td>
</tr>
<tr>
<td>Transportation and Logistics</td>
</tr>
<tr>
<td>Travel and Leisure</td>
</tr>
<tr>
<td>Uncategorized</td>
</tr>
<tr>
<td>Utilities</td>
</tr>
</tbody>
</table>

#### 5.9.4 Deepnet Tags

A deepnet can also have a number of **tags** associated with it that can help to retrieve it via the BigML API or to provide deepnet with some extra information. A deepnet inherits the tags from the dataset used to create it. Each tag is limited to a maximum of 128 characters. Each deepnet can have up to 32 different tags.

#### 5.9.5 Deepnet Counters

For each deepnet, BigML also stores a number of counters to track the number of other resources that have been created using the deepnet as a starting point. In the deepnet view, you can see a menu option that displays counters for evaluations, single and batch predictions. It also allows you to quickly jump to all the resources of one type. (See **Figure 5.106**.)
5.10 Deepnet Privacy

Privacy options for a deepnet can be defined in the More Info panel, displayed in Figure 5.107. There are two levels of privacy for BigML deepnets:

- **Private**: only accessible by authorized users (the owner and those who have been granted access by him or her).
- **Shared**: accessible by any user with whom the owner shares the secret link.

![Figure 5.107: Deepnet privacy](image)

5.11 Moving Deepnets to Another Project

When you create a deepnet, it will be assigned to the same project where the original dataset is located. Deepnets can only be assigned to a single project. However, you can move deepnets between projects. The menu option to do this can be found in two places:

1. In the deepnet list view, click the **MOVE TO** option within the **1-click action menu** and select another project or create a new one. (See Figure 5.108.)

![Figure 5.108: Change project from 1-click action menu](image)

2. In the deepnet list view, click the **MOVE TO** option within the **pop up menu** and select another project or create a new one. (See Figure 5.109.)
5.12 Stopping Deepnets

You can stop the creation of a deepnet before the task is finished by clicking the DELETE DEEPNET option from the 1-click action menu (see Figure 5.110), or from the pop up menu in the deepnet list view (see Figure 5.111).
Figure 5.111: Stop deepnet creation from pop up menu

A modal window will be displayed asking you for confirmation. If you stop the deepnet during its creation you won’t be able to resume the same task. If you want to create the same deepnet, you will have to start a new task.

Figure 5.112: Confirmation message to delete a deepnet

5.13 Deleting Deepnets

You can delete your deepnet by clicking in the DELETE DEEPNET option from the 1-click action menu (see Figure 5.113) or using the pop up menu on the deepnet list (see Figure 5.114).
A modal window will be displayed asking you for confirmation. Once a deepnet is deleted, it is permanently deleted, and there is no way you (or even the IT folks at BigML) can retrieve it.
5.14 Takeaways

This chapter explains deepnets in detail. Here is a list of key points:

- A deepnet is a supervised Machine Learning algorithm used to solve classification and regression problems.
- A deepnet is built from a dataset available in BigML and used to make an evaluation, a single prediction, or a batch prediction. (See Figure 5.116.)
- You can create a deepnet with just one click or configure it as you wish. BigML provides several configuration options before creating your deepnet.
- To create a deepnet you need a dataset containing at least one categorical or numeric field.
- If you do not specify any objective field, BigML will take the last valid field in your dataset.
- BigML allows you to include your numeric fields’ missing values as valid values to train your deepnet.
- The PDP view provides a visual way to analyze a field impact on predictions given certain values for the rest of the fields.
- You get all the objective field class probabilities along with the predicted class.
- You need to evaluate your deepnet performance using data that the model has not seen before.
- The ultimate goal in building a deepnet is being able to make predictions with it.
- BigML allows you to quickly make predictions for single instances by providing a form containing the fields used by the deepnet, so you can easily set the input data and get an immediate response.
- BigML batch predictions allow you to make simultaneous predictions for multiple instances. All you need is the deepnet you want to use to make predictions and a dataset containing the instances for which you want to obtain predictions.
- You can configure your batch predictions output file settings.
- You can download your deepnet to perform local predictions.
- You can add descriptive information to your deepnets (name, description, tags, and category).
- You can move your deepnets between projects.
- You can share your deepnets with other people using the secret link.
- You can stop your deepnet creation by deleting it.
- You can permanently delete an existing deepnet.
Figure 5.116: Deepnet Workflow
Fusions

6.1 Introduction

There are many Machine Learning problems that can be solved using supervised Machine Learning techniques. These techniques solve problems that require the prediction of an output variable (objective field) given a number of input variables (input fields). These problems can be classified into two groups: classification problems if you need to predict a category (label or class) or regression problems if the output is a continuous value (a real number).

Classification and regression problems can be solved using multiple Machine Learning methods in BigML, such as models, ensembles, logistic regressions, and deepnets. These methods are explained in Chapter 1, Chapter 2, Chapter 4, and Chapter 5 respectively. Depending on the problem you are trying to solve and the data available, some techniques may perform significantly better than others. Fusions combine these Machine Learning models and average their predictions to balance out the individual weaknesses of the single models and yield a better performance. Fusions are based on the same “wisdom of the crowds” principle as ensembles under which the combination of multiple models is often more performant than any of its individual models. The component models have to be as accurate and diverse as possible. See Section 6.2 for more details.

This chapter contains a comprehensive description of BigML’s fusions including how they can be created (Section 6.3), all configuration options available (Section 6.4), and the different visualizations provided by BigML (Section 6.5). See Section 6.6 for an explanation of how fusions can be used to make predictions. You can also export your fusions in different formats to make local predictions faster at no cost (Subsection 6.7.1). The process to evaluate your fusions’ predictive performance in BigML is explained in a different chapter (Chapter 7).

On BigML, the third tab of the main menu of the Dashboard allows you to list all your available fusions. The fusion list view (Figure 6.1), details the Name, the objective field Type (classification or regression), the Objective (objective field name), Age (time elapsed since it was created), and number of evaluations, predictions, and batch predictions that have been created using that fusion. The SEARCH menu option in the top right corner of the fusion list view allows you to search your fusions by name or ID. You can also export a fusion by the parameters used to create it by typing in the search box the syntax “config:” followed by the parameters you are looking for.

By default, when you first create an account at BigML, or every time that you start a new project, your list of fusions will be empty. (See Figure 6.2.)

Finally, in Figure 6.3 you can see the icon used to represent a fusion.

6.2 Understanding Fusions

BigML fusions combine multiple Machine Learning models, ensembles, logistic regressions, and deepnets and average their predictions to balance out the individual weaknesses of the single models and yield a better performance. Fusions are based on the assumption that the combination of multiple models often outperforms the component single models. We can see fusions as an heterogeneous ensemble composed by different types of models instead of just decision trees.
For fusions to improve upon the individual model performance, the component models have to be as accurate and diverse as possible. If you use several identical models or models with sub-par performance, the fusion will not be able to improve the results of the models. Although for many cases the gains in performance may not be huge, fusions have other advantages such as the fact that they are usually more stable than single models and the model errors tend to be smoothed out across the entire input space. For problems in which every small gain in model performance measures counts, fusions can be a quick solution because they are so easy to execute on BigML.

Fusions can solve classification and regression problems on BigML. For classification problems, fusions average the per-class probabilities across all the component models. The class with the highest probability is predicted. For regression models, the final prediction is the result of averaging the per-model predicted values. These methods to combine single model predictions and return an output is equivalent to the ensembles “probability” method (see Subsection 2.6.3.2).

You need at least one existing supervised model to create a fusion. The component models of a fusion must have a compatible categorical or numeric objective field (see Subsection 6.2.1). The models can be built using different datasets and different input fields. If two fields from different models have the same name, they will be considered the same field to create the fusion. The per-field importances from the components models will be average to create the fusion field importances (see Subsection 6.2.2).

### 6.2.1 Fusion Objective Field

All the models composing the fusion must have the same objective field. BigML checks that two models have the same objective field by ensuring the objectives have the same field optype (numeric or categorical) and the same name (if you create a fusion from the BigML API the field ID instead of the name will be used to validate the compatibility of two objective fields). You can select models with different objective field names (see Subsection 6.4.2) if they are compatible.

### 6.2.2 Fusion Field Importances

The fusion field importances are calculated by averaging the per-field importances of the following component models: decision trees, ensembles and deepnets. These averages are normalized so the sum of all field importances is 100%. Logistic regressions are excluded from this calculation since the field importances cannot be calculated for them. If the fusion is only composed by logistic regression models, it will not have importances.

### 6.3 Creating Fusions

You need at least one model to create a fusion. You can use several options depending if you want to select multiple models of one type or few models of different types to create a fusion:

- If you want to use a single decision tree, ensemble, logistic regression or deepnet, you can click on the CREATE FUSION option from the model 1-click action menu (see Figure 6.4) or from the pop up menu on the list view (see Figure 6.5):

---

2https://bigml.com/api/fusions
• If you want to select multiple decision trees, ensembles, logistic regressions or deepnets at the same time, you can click on the CREATE FUSION option from the list view (see Figure 6.6).
When you select one model from the list, you will only be able to select other compatible models, i.e., those models with the same objective field type and name. The rest of models will not be selectable (see Figure 6.7).

- You can also select multiple models from the OptiML view and click on the Create fusion button (see Figure 6.8).
If you want to create a new fusion using the same models of another existing fusion, you can click on the CREATE FUSION USING THIS ONE option from the fusion 1-click action menu. This option is very useful to iterate your existing fusions.

All of the above options will redirect you to the fusion creation view (see Figure 6.10). From this view, you can select more models and configure the fusion parameters. See a detailed explanation of these options in Section 6.4.
6.4 Fusion Configuration Options

From the fusion creation view, you can select more models or remove the existing ones (Subsection 6.4.1), remove the objective field filter (Subsection 6.4.2) and configure the model weights (Subsection 6.4.3).

6.4.1 Models

Select up to 1,000 models, ensembles, logistic regressions, and/or deepnets to create your fusion. You can also use existing fusions to include them in a new fusion.

Click on the model icon you want and type the model name in the selector (see Figure 6.11). You can also select models created using OptiML by clicking the option “Show OptiML models”. By enabling this option, only OptiML models will be listed in the selector.

The selected models will be shown below (see Figure 6.12) and you can remove them or assign them different weights (see Subsection 6.4.3). All the selected models must have compatible objective fields;
otherwise, the fusion creation will fail. When you select the first model, the rest of models will be filtered by the same objective type (numeric or categorical) and the same objective name. You can remove the filter in case you want to select models with different objective field names (see Subsection 6.4.2).

6.4.2 Objective Field

The objective field, or “target field”, is the field you want to predict. Fusions support categorical and numeric fields as the objective field. All the models selected to create a fusion must have compatible objective fields.

BigML takes the first model objective field type and name to filter the rest of models available in the selector (see Figure 6.13).
Figure 6.13: The model search will be filtered by the objective field name

You can remove this filter and select models with different objective field names (see Figure 6.14).

Figure 6.14: Remove the objective field filter to select models with different objective field names

When you remove the filter and select a new model with a different objective field name, you can choose which objective field from the selected models you want to use for the filter, if any.
6.4.3 Weights

You can assign different weights to the selected models before creating the fusion. At the prediction time, BigML will perform a weighted average of all model predictions taking into account each model weight. Therefore, if a model has a weight of 2 (while the rest of models have a weight of 1) this model's predictions will count double. You can assign weights of 0 if you do not want a model to have any impact on the final predictions.
6.4.4 Creating Fusions with Configured Options

After finishing the configuration of your options, you can change the default fusion name in the editable text box. Then you can click on the [Create fusion] button to create the new fusion, or reset the configuration by clicking on the [Reset] button.

Figure 6.17: Create fusion after configuration
6.4.5 API Request Preview

The API Request Preview button is in the middle on the bottom of the configuration panel, next to the Reset button (See (Figure 6.17)). This is to show how to create the fusion programmatically: the endpoint of the REST API call and the JSON that specifies the arguments configured in the panel. Please see (Figure 6.18) below:

![Figure 6.18: Fusion API request preview](image)

There are options on the upper right to either export the JSON or copy it to clipboard. On the bottom there is a link to the API documentation for fusions, in case you need to check any of the possible values or want to extend your knowledge in the use of the API to automate your workflows.

Please note: when a default value for an argument is used in the configuration, the argument won’t appear in the generated JSON. Because during API calls, default values are used when arguments are missing, there is no need to send them in the creation request.

6.5 Visualizing Fusions

After creating your fusion, you will be able to analyze the results with BigML’s unique visualization: a Partial Dependence Plot (PDP) to examine how the input fields impact the objective field (see Subsection 6.5.1). You will also be able to find the models composing the fusion under the model list tab (see Subsection 6.5.2).

Switch from the PDP view to the model list view by clicking on the icons in the top bar menu of the fusion view (see Figure 6.19).
6.5.1 Fusion Partial Dependence Plot

The chart view is composed of three main parts: the CHART itself, the PREDICTION legend and the INPUT FIELDS form. (See Figure 6.20.) You can find a detailed explanation of each one below.
The CHART allows you to view the impact of the input fields on the objective field predictions. You can select a different input field for each axis and the predictions are represented in different colors on the heat map. You can select numeric or categorical fields for the axis and inspect the axis values in the grey area next to the selector. You can switch the axis by clicking on the option on top of the chart area (see Figure 6.21).
You can freeze the view by pressing `Shift` and release it again by pressing `Escape` from your keyboard. When the view is frozen, an edition icon will appear so you can edit the axis values and obtain a prediction for another desired value. You can see the predictions in the legend to the right.

- The PREDICTION legend allows you to visualize the objective field classes (classification fusions) or the predicted value (regression fusions). For classification, each class is represented by a color, the main probability color bar at the top is the probability for the predicted class. By default, colors are shaded according to the prediction range shown in the chart area. That way, smaller differences in predictions are easier to perceive. However, you can choose to see the color shading according to the total range of possible values for the objective field by clicking on the icon next to the prediction bar `Total` (see Figure 6.22). You can also select to see only one of the classes using the class selector at the bottom of the legend.

Figure 6.21: Select fields for your fusion PDP axis
Below the chart legend, you can find the INPUT FIELDS form (see Figure 6.23). You can configure the values for any numeric, categorical, text or items field. By changing their values, you can see the predictions changing in real-time. You can also select or disable your input fields, so they will be treated as missing values. If your fusion contains models, ensembles or deepnets, you will be able to see each field importance and order the fields from the highest to lowest importance. If your fusion only contains logistic regressions, the fields will not have importances.

Figure 6.22: Prediction legend

Freeze this view by pressing Shift, and release it again by pressing Escape from your keyboard.
Figure 6.23: Configure the values for other input fields

Note: if any of the models composing the fusion fail to calculate the prediction (e.g., if a logistic regression is trained without missing numerics it will fail for predictions that have missing numeric values). BigML will use the other models to return a result instead of returning an error.

Moreover, the chart includes a reset option for your input fields values, and an export option to download your chart in PNG format explained below:

- After selecting the fields for the axis or configuring the input fields values, you can set them again to the default view by clicking the reset icon highlighted in Figure 6.24.
You can also export your chart in PNG format with or without legends. Freeze the view by pressing `Shift` from your keyboard and export the chart to get the prediction values in the legend. Release the view by pressing `Escape`.

**Figure 6.25: Export chart as image with or without legends**

Note: there are some limitations for the number of classes of the objective field and the number of input fields to visualize your fusion in the chart (explained in Section 6.8).

### 6.5.2 Fusion Model List

The model list contains all the models composing the fusion. You can see the model number (this is the order in which you selected the models to create the fusion), the model type icon, the model name, the model configuration options, and the model weights (see Subsection 6.4.3 for an explanation about the weights). If you click on the model name, you will be taken to that model view. You can filter this list by model type by clicking on the icons shown in **Figure 6.26**.
6.6 Fusion Predictions

6.6.1 Introduction

The ultimate goal in building a fusion is being able to make predictions with it. On BigML, you can make predictions for single instances or for many instances in batch. Each prediction comes with a measure indicating the prediction confidence. For regression problems, the expected error is provided along with the predicted value, for classification problems the vector of probabilities per class is returned (a percentage ranging from 0% to 100%).

The predictions tab in the main menu of the BigML Dashboard is where all your saved predictions are listed. (see Figure 6.27). You can search your predictions by name clicking on the search option on the top menu. In the predictions list view, you can see the fusion icon used for each prediction, the Name of the prediction, the Objective (objective field name), the Prediction (the prediction result), and the Age (time since the prediction was created).
When you first create an account on BigML, or every time that you start a new project, your list of predictions will be empty (see Figure 6.28).

Fusion predictions are saved under the CLASSIFICATION & REGRESSION option in the menu (see Figure 6.29).

Select the list for your single instances predictions or your batch predictions by clicking on the corresponding icons. (See Figure 6.30 and Figure 6.31.)
6.6.2 Creating Fusion Predictions

BigML provides two options to predict with your fusions explained in the following subsections:

- **PREDICT**: to predict a single instance
- **BATCH PREDICTION**: to predict multiple instances in batch.

6.6.2.1 Predict

BigML allows you to quickly make predictions for single instances by providing a form containing the input fields used by the fusion, so you can easily set the values and get an immediate response.

Follow these steps to create a single prediction:

1. Click **PREDICT** in the fusion 1-click action menu. (See Figure 6.32.)

   Alternatively, click **PREDICT** in the pop up menu in the list view. (See Figure 6.33.)
2. You will be redirected to the prediction form where you will find all the fields used by the fusion as input fields. (See Figure 6.34.)

3. Select the fields to be used for your prediction, set the input values for your selected fields and
click [Predict]. Non-selected fields will be considered as missing values during the prediction. See the prediction at the top of the view along with the rest of the class probabilities (Figure 6.35). Hide or display the histogram view containing the rest of your class probabilities. You can download all the probabilities in PNG, CSV or JSON format by clicking on the corresponding icons. (See Subsection 6.6.4.1.)

BigML predictions are synchronous, i.e., when you send the input data, you get an immediate response.

4. Your prediction is automatically saved and you can find it in the predictions list view.

For regression fusions, the process is the same, but instead of the predicted classes and the probabilities you get a numeric value for the objective field along with an expected error as the certainty measure.

Note: this option is only available from the BigML Dashboard for fusions with less than 100 fields. If you want to perform single instance predictions for a higher number of fields, use the BigML API³.

³https://bigml.com/api/predictions
6.6.2.2 Batch Predictions

BigML batch predictions allow you to make simultaneous predictions for multiple instances. All you need is the fusion you want to use to make predictions and a dataset containing the instances you want to predict. BigML will create a prediction for each instance in the dataset.

Follow these steps to create a batch prediction:

1. Click on **BATCH PREDICTION** option under the fusion **1-click action menu** (Figure 6.36)

   ![Figure 6.36: Create batch prediction using 1-click action menu](image)

   Alternatively, click on **CREATE BATCH PREDICTION** in the **pop up menu** of the list view (Figure 6.37).

   ![Figure 6.37: Create batch prediction using pop up menu](image)

2. **Select the dataset** containing all the instances you want to predict. The instances should contain the input values for the fields used by the fusion as input fields. From this view you can also select another fusion from the selector (see Figure 6.38).
3. After the fusion and the dataset are selected, the batch prediction **configuration options** will appear along with a **preview** of the prediction output (a CSV file) (see Figure 6.39). The default output format includes all your prediction dataset fields and adds an extra column with the class predicted. See Subsection 6.6.3 for a detailed explanation of all configuration options.

4. By default, BigML generates an output **dataset** with your batch predictions that you can later find in your datasets section of the BigML Dashboard. This option is active by default but you can deactivate it by clicking on the icon shown in Figure 6.40.
After you configure your batch prediction, click on the green button **Predict** to generate your batch prediction.

5. When the batch prediction is created, you will be able to **download the CSV file** containing all your dataset instances along with a prediction for each one of them. If you did not disable the option to create a dataset previously explained, you will also be able to access the **output dataset** from the batch prediction view (see **Figure 6.41**).

---

**Figure 6.40: Create a dataset from batch prediction**

**Figure 6.41: Download batch prediction CSV file and output dataset**
6.6.3 Configuring Fusion Predictions

BigML provides several options to configure your batch predictions such as the missing strategy (see Subsection 6.6.3.1), setting a probability threshold (see Subsection 6.6.3.2), default values for your missing numeric values (see Subsection 6.6.3.3), fields mapping (see Subsection 6.6.3.5), and output file settings (see Subsection 6.6.3.6).

6.6.3.1 Missing Strategies

This option is available only when the fusion contains models and/or ensembles since the missing strategy has no effect for logistic regressions or deepnets.

When you create a new prediction, BigML will automatically navigate through the corresponding model or ensemble to find the leaf node that best classifies the new instance. However, it may just so happen that your new data (the instances you want to predict) does not have populated values for all the fields used in building the original ensemble. For example, imagine that you are trying to predict diabetes and you have the patient’s glucose level and BMI (Body Mass Index) but not his blood pressure. If the model or ensemble arrives at a node where the blood pressure level is required, BigML can handle this missing value by using one of these two strategies:

- **Last prediction**: it returns the prediction value and probability of the parent node.
- **Proportional**: it combines all predictions beneath the current node based on the data distribution of their child nodes in order to compute the prediction value and probability.

For single predictions you can select either Missing strategies by clicking in the icons shown in Figure 6.42.

![Figure 6.42: Missing strategies for fusion single predictions](image)

For batch predictions you can find both options under the configuration panel as shown in Figure 6.43.

Copyright © 2023, BigML, Inc.
6.6.3.2 Probability threshold

Probability thresholds usually makes sense when you want to minimize false positives at the cost of false negatives. The positive class will be predicted if its probability is greater than the given threshold; otherwise, the following class with greater probability will be predicted. This option is only available for classification fusions.

Follow these steps to configure a threshold for your batch prediction:

1. Select the positive class, i.e., the class for which you want to apply the threshold (Figure 6.44).
Figure 6.44: Select the positive class

2. Set a probability threshold using the slider shown in Figure 6.45 and click **Predict**.
Figure 6.45: Set a probability threshold

3. If the positive class probability is greater than the given threshold, it will be predicted; otherwise, the following class with greater probability will be predicted.
You can also find the same options to set a threshold for batch predictions under the CONFIGURE panel (see Figure 6.47).
6.6.3.3 Default Numeric Value

By using the Default numeric value before creating your batch prediction, you can easily replace all the missing numeric values in the dataset by the field's Mean, Median, Maximum, Minimum or by Zero (see Figure 6.48).
6.6.3.4 Excluded Fields

This option allows you to exclude a set of fields from the prediction calculation but at the same time keep them in the output file and dataset (see Figure 6.49).

---

Figure 6.48: Configure Default numeric value for batch prediction

Figure 6.49: Exclude fields from the fusion prediction
6.6.3.5 Fields Mapping

You can specify which input fields of the fusion match with the fields in the dataset containing the instances you want to predict. BigML automatically matches fields by name, but you can also set an automatic match by field ID by clicking on the green switcher. Additionally, you can manually search for fields or remove them from the Dataset fields column if you do not want them to be considered during the batch prediction (see Figure 6.50).

Figure 6.50: Configure the fields mapping for batch prediction

Note: Fields mapping from the BigML Dashboard is limited to 200 fields. For batch predictions with a higher number of fields, map your fields using the BigML API.

6.6.3.6 Output Settings

Batch predictions return a CSV file containing all your instances and the final predictions. Tune the following settings to customize your prediction file (see Figure 6.51):

- **Separator**: this option allows you to choose the best separator for your output file columns. The default separator is the comma. You can also select the semicolon, tab, or space.

- **New line**: this option allows you to set the new line character to use as the line break in the generated csv file: “LF”, “CRLF”.

See also:

- [BigML API](https://bigml.com/api/batchpredictions#bp_batch_prediction_arguments)
• **Output fields**: by clicking on the list icon next to the separator selector, you can include or exclude all your dataset fields from your output file. You can also individually select the fields you want to include or exclude using the multiple output fields selector. **Note**: a maximum of 100 fields can be displayed in this selector, but all your dataset fields will be included in the output file by default unless you exclude them.

• **Headers**: this option includes or excludes a first row in the output file (and in the output dataset) with the names of each column (input field names, prediction column name, probability column name, etc.). By default, BigML includes the headers.

• **Prediction column name**: customize the name for your predictions column. By default, BigML takes the name of the fusion’s objective field.

• **Probability**: this option allows you to include an additional column with the probability for the predicted class. By default it is not included in your output file. For regression fusions, you will find the expected error instead of the probability.

• **Probability column name**: customize the name for the probability column if you include it in the output file. BigML sets “probability” as the default name. For regression fusions, you will find the expected error column name.

• **Individual model predictions**: this includes all the per-model predictions composing the fusion. That will add a column per model, named `<prediction_name>_<n>` where `n` is the position of the model in the model list in the fusion, starting at 1.

• **All class probabilities**: this includes all the probabilities of the objective field classes per instance. This option will add `n` extra columns, one by class in the objective field. This option does not exist for regression fusions.

• **Field importances**: this option allows you to include a column for each of the field relative importances for the fusion predictions (taking into account the decision trees, ensembles, and deepnets composing the fusion). This option will add a column per field, named “<field_name> importance”. If the fusion only contains logistic regressions, these field importances cannot be calculated.

Figure 6.51: Fusion output settings for batch predictions
6.6.4 Visualizing Fusion Predictions

Fusion predictions visualizations change depending if you are predicting a single instance (Subsection 6.6.4.1), or multiple instances using the batch predictions option (Subsection 6.6.4.2).

6.6.4.1 Single Predictions

For single predictions, find the predicted class given the input fields values at the top of the form along with its probability (see Figure 6.52).

Below the prediction, there’s a histogram representing the rest of the objective field class probabilities. All the class probabilities must sum to 100%. Show or hide this view by clicking on the icon highlighted in Figure 6.53. You can see up to seven different classes at the same time; if you have more than seven classes, you can see the others by clicking on the arrows icons. Export this view in PNG, CSV, or JSON format by clicking on the corresponding icons (see Figure 6.53).
For regression fusions, instead of the class probabilities you will get the predicted numeric value for the objective field.

### 6.6.4.1.1 Prediction explanation

Prediction explanation helps understand why a fusion makes a certain prediction. This is very useful in many applications, and the reasons behind a fusion's prediction are often as important as the prediction itself.

BigML prediction explanation is based on Shapley values. For more information, please refer to this research paper: [A Unified Approach to Interpreting Model Predictions](#) [3].

For any classification or regression fusion, you can request the explanation for the prediction by clicking the **prediction explanation** icon and then click **Predict** (see Figure 6.54).
The prediction explanation represents the most important factors considered by the fusion in a prediction given the input values. Each input value will yield an associated importance, as you can see Figure 6.55. The importances across all input fields should sum to 100%.

Figure 6.55: Input field importances
For some input fields you will see a “+” icon next to the importance. This is because the importance may not be directly associated with the input value, i.e., it can be explained by other reasons. In the Figure 6.56 below, the importance of the field “Pregnancies” is not explained by this field being equal to 10, rather, it is because this field value is greater than 5.

![Figure 6.56: See the detailed explanation](image)

The prediction explanation for fusions is calculated using the results of over a thousand distinct predictions using random perturbations of the input data. For this reason, the calculation of the explanation may take some time to be computed.

**Note:** the input field importances in the prediction explanation are different from the overall field importances of the fusion. A field can be very important for the fusion but insignificant for a given prediction.

### 6.6.4.2 Batch Prediction

After creating your batch prediction, you get a **CSV file** and, optionally, an **output dataset**. Both outputs are explained in the following subsections.

#### 6.6.4.2.1 Output CSV file

The batch prediction generates a CSV file containing your **predictions** for each of your dataset instances in the last column (see Figure 6.57).
You can configure several options to customize your CSV file. You can find a detailed explanation of those options in Subsection 6.6.3.6.

See an output CSV file example in Figure 6.58. The column class in this example contains the final prediction (it is named by default as your fusion’s objective field). In this case, we are predicting whether a person is a good or a bad candidate for holding a credit. This file has been configured to also contain the probability for each prediction.

```
duration,age,amount,purpose,class,probability
24,26,5433,used car,good,0.88785
36,42,8086,new car,bad,0.55526
24,28,1376,radio/tv,good,0.8385
48,31,6758,radio/tv,bad,0.73576
26,30,7966,used car,good,0.7201
12,42,2577,furniture/equipment,good,0.67644
36,30,4455,business,good,0.62227
18,32,1442,new car,bad,0.75488
9,22,276,new car,good,0.57819
```

Figure 6.58: An example of a fusion batch prediction CSV file

6.6.4.2 Output Dataset

By default, BigML automatically creates a dataset out of your batch prediction. You can disable this option by configuring your batch prediction (see Subsection 6.6.3.6). You will find the output dataset in your batch prediction view as shown in Figure 6.59.
In the output dataset, you can find an additional field (named by default as your fusion’s objective field) containing the class predicted for each one of your instances (see Figure 6.60). If you configured your batch prediction to include the prediction probabilities and all class probabilities, you will be able to find them in the last fields of your output dataset.
6.6.5 Consuming Fusion Predictions

You can fully use single and batch predictions via the BigML API and bindings. The following subsections explain both tools.

6.6.5.1 Using Fusion Predictions via the BigML API

Fusion predictions have full citizenship in the BigML API which allows you to programmatically create, configure, retrieve, list, update, and delete single and batch predictions.

In the example below, see how to create a single prediction using a fusion and define the input data once you have properly set the \$BIGML_AUTH environment variable to contain your authentication credentials:

```bash
curl "https://bigml.io/prediction?$BIGML_AUTH" \
-X POST \
-H "content-type: application/json" \
-d '{"fusion": "fusion/50650bdf3c19201b64000020", "input_data": {"000001": 3, "000002":4.5, "000003":5}}'
```

For more information on using predictions through the BigML API, please refer to the documentation\(^5\).

6.6.5.2 Using Fusion Predictions via BigML Bindings

You can also create, configure, retrieve, list, update, and delete single and batch predictions via BigML bindings, which are libraries aimed to make it easier to use the BigML API from your language of choice including Python, Node.js, Java, Swift and Objective-C. See below an example to create a fusion with the Python bindings.

```python
from bigml.api import BigML
api = BigML()
prediction = api.create_prediction("fusion/50650bdf3c19201b64000020", 
{"credit_amount": 5, "duration": 2.5})
```

For more information on BigML bindings, please refer to the bindings page\(^6\).

6.6.6 Descriptive Information

Each fusion prediction has an associated name, description, category, and tags. You can find a brief description of each concept in the following subsections. The MORE INFO menu option displays a panel that provides editing options (see Figure 6.61).

\(^5\)https://bigml.com/api/predictions
\(^6\)https://bigml.com/tools/bindings
6.6.6.1 Name

If you do not specify a name for your predictions, BigML assigns a default name depending on the type of predictions:

- **Single predictions**: the name always follows the structure "<fusion name>".
- **Batch predictions**: BigML combines your prediction dataset name and the fusion name: "<fusion name> with <dataset name>".

Prediction names are displayed on the list and also on the top bar of a prediction view. Prediction names are indexed to be used in searches. Rename your predictions at any time from the MORE INFO menu.

The name of a prediction cannot be longer than 256 characters. More than one prediction can have the same name even within the same project, but they will always have different identifiers.

6.6.6.2 Description

Each prediction also has a description that is useful for documenting your Machine Learning projects. Predictions have the same description as the fusion used to create them.

Descriptions can be written using plain text and also markdown\(^7\). BigML provides a simple markdown editor that accepts a subset of markdown syntax (see Figure 6.62).

---

\(^7\)https://en.wikipedia.org/wiki/Markdown
Descriptions cannot be longer than \(8192\) characters.

### 6.6.6.3 Category

Each prediction has an associated category taken from the fusion used to create it. Categories are useful to classify predictions according to the domain which your data comes from, which helps when you use BigML to solve problems across industries or multiple customers.

A prediction category must be one of the categories listed on table Table 4.5.

### 6.6.6.4 Tags

A prediction can also have a number of tags associated with it. These tags help to retrieve the prediction via the BigML API or to provide predictions with some extra information. Your prediction inherits the tags from the fusion used to create it. Each tag is limited to a maximum of 128 characters. Each prediction can have up to 32 different tags.

### 6.6.7 Fusion Predictions Privacy

The link displayed in the Privacy panel is the private URL of your prediction, so only a user logged into your account is able to see it. Neither single predictions nor batch predictions can be shared by using a secret link (see Figure 6.63).
6.6.8 Moving Fusion Predictions to Another Project
When you create a prediction, it will be assigned to the same project where the original fusion is located. You cannot move predictions between projects as you do with other resources.

6.6.9 Stopping Fusion Predictions
Single predictions are synchronous resources, so you cannot cancel them during the creation since you get the result immediately.

On the other hand, batch predictions are asynchronous resources, so you can stop their creation before the task is finished. Use the DELETE BATCH PREDICTION option from the 1-click action menu (Figure 6.64) or from the pop up menu on the list view.

![Figure 6.64: Stop fusion batch prediction from 1-click action menu](image)

A modal window will be displayed asking you for confirmation. If you stop the prediction during its creation you won’t be able to resume the same task again. So if you want to create the same prediction, you will have to start a new task.

![Figure 6.65: Fusion delete prediction confirmation](image)
6.6.10 Deleting Fusion Predictions

You can **DELETE** your single or batch predictions from the predictions view, using the **1-click action menu** (see Figure 6.66) or using the **pop up menu** on the predictions list view (see Figure 6.67).

![Figure 6.66: Delete fusion prediction from 1-click menu](image)

![Figure 6.67: Delete fusion prediction from pop up menu](image)

A modal window will be displayed asking you for confirmation. Once a prediction is deleted, it is permanently deleted, and there is no way you (or even the IT folks at BigML) can retrieve it.
6.7 Consuming Fusions

Similarly to other models in BigML, fusions are white-box models, so you can download them and use them locally to make predictions. You can also create and consume your fusions programmatically via the BigML API and bindings. The following subsections explain these three options.

6.7.1 Downloading Fusions

You can download your fusions in several programming languages including JSON PML, Python or Node.js. By downloading your fusions you will be able to compute predictions locally, free of latency and at no cost. Click on the download icon in the top menu (see Figure 6.69), and select your preferred option (see Figure 6.70.)
6.7.2 Using Fusions Via the BigML API

Fusions have full citizenship in the BigML API which allows you to programmatically create, configure, retrieve, list, update, delete, and use them for predictions.

In the example below, see how to create a fusion using two existing models once you have properly set the `BIGML_AUTH` environment variable to contain your authentication credentials:

```
curl "https://bigml.io/fusion?\$BIGML_AUTH" \
-X POST \
-H 'content-type: application/json' \
-d '{"models": [
  "model/5af06df94e17277501000010",
  "model/5af06df84e17277502000019",
  "deepnet/5af06df84e17277502000016",
  "ensemble/5af06df74e1727750100000d"
]}'
```

For more information on using fusions through the BigML API, please refer to the documentation.

6.7.3 Using Fusions Via the BigML Bindings

You can also create and use fusions via BigML bindings which are libraries aimed to make it easier to use the BigML API from your language of choice. BigML offers bindings in multiple languages including Python, Node.js, Java, Swift and Objective-C. See below an example to create a fusion with the Python bindings.

```
from bigml.api import BigML
api = BigML()
```

---

8 https://bigml.com/api/fusions
fusion = api.create_fusion(['model/5af06df94e17277501000010',
                          'model/5af06df84e17277502000019',
                          'deepnet/5af06df84e17277502000016',
                          'ensemble/5af06df74e1727750100000d'])

For more information on BigML bindings, please refer to the bindings page.

### 6.8 Fusion Limits

There are some limits that apply to the creation of any BigML resource. These are limits based on the number of classes, terms and items that can be considered to create your models. In the case of fusions, the single model limits apply along with some specific limits for the fusion creation and visualization:

- **Maximum models**: you can select a maximum of 1,000 models to create your fusion.
- **PDP limits**: the fusion Partial Dependence Plot will not be displayed if:
  - The fusion only has **text** or **items** fields as inputs (because only categorical and numeric fields can be plotted on the PDP axes)
  - The **objective field** has more than 200 categories.
  - The fusion reaches a certain number of models defined by the formula: \((\text{decision trees} + 5 \times \text{ensembles} + \text{logistic regressions} + 10 \times \text{deepnets}) \leq 50\).

In these cases, the model list will be the default view and the icon to access the PDP will be disabled (see Figure 6.71).

![Figure 6.71: The PDP cannot be displayed](https://bigml.com/tools/bindings)

It may happen that the number of component models does not reach the maximum but they are too big to load the fusion predictions in a reasonable time, in this case, BigML will display a warning message (see Figure 6.72).
Figure 6.72: The PDP cannot be displayed because the predictions take too much time to be computed.

**Note:** the PDP limits only affect the fusion visualization, i.e., even if your dataset reaches those limits, you can still create the fusion, evaluate it and use it to make predictions.

**Note 2:** in the case a fusion has more than 100 input fields, only the top 100 fields ordered by importance will be displayed in the PDP view.

### 6.9 Descriptive Information

Each fusion has an associated **name**, **description**, **category**, and **tags**. The following subsections provide a brief description for each concept. In **Figure 6.73**, you can see the editing options provided in the MORE INFO menu.
6.9.1 Fusion Name

Each fusion has a name that is displayed in the fusion list view and also on the top bar of the fusion view. Fusion names are indexed to be used in searches. A fusion inherits the name from the first model used to create it. Change it using the MORE INFO menu option on the right corner of the fusion view. The name of a fusion cannot be longer than 256 characters. More than one fusion can have the same name even within the same project, but they will always have different identifiers.

6.9.2 Description

Each fusion also has a description that is very useful for documenting your Machine Learning projects. Descriptions can be written using plain text and also markdown\(^\text{10}\). BigML provides a simple markdown editor that accepts a subset of markdown syntax (see Figure 6.74).

---

\(^{10}\)https://en.wikipedia.org/wiki/Markdown
Descriptions cannot be longer than 8192 characters and can use almost any character.

### 6.9.3 Category

A category can be associated with each fusion. Categories are useful to classify fusions according to the domain which your data comes from. This is helpful when you use BigML to solve problems across industries or for multiple customers.

A fusion category must be one of the 24 categories listed on Table 6.1.
Table 6.1: Categories used to classify fusions on BigML

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace and Defense</td>
</tr>
<tr>
<td>Automotive, Engineering and Manufacturing</td>
</tr>
<tr>
<td>Banking and Finance</td>
</tr>
<tr>
<td>Chemical and Pharmaceutical</td>
</tr>
<tr>
<td>Consumer and Retail</td>
</tr>
<tr>
<td>Demographics and Surveys</td>
</tr>
<tr>
<td>Energy, Oil and Gas</td>
</tr>
<tr>
<td>Fraud and Crime</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Higher Education and Scientific Research</td>
</tr>
<tr>
<td>Human Resources and Psychology</td>
</tr>
<tr>
<td>Insurance</td>
</tr>
<tr>
<td>Law and Order</td>
</tr>
<tr>
<td>Media, Marketing and Advertising</td>
</tr>
<tr>
<td>Miscellaneous</td>
</tr>
<tr>
<td>Physical, Earth and Life Sciences</td>
</tr>
<tr>
<td>Professional Services</td>
</tr>
<tr>
<td>Public Sector and Nonprofit</td>
</tr>
<tr>
<td>Sports and Games</td>
</tr>
<tr>
<td>Technology and Communications</td>
</tr>
<tr>
<td>Transportation and Logistics</td>
</tr>
<tr>
<td>Travel and Leisure</td>
</tr>
<tr>
<td>Uncategorized</td>
</tr>
<tr>
<td>Utilities</td>
</tr>
</tbody>
</table>

6.9.4 Tags

A fusion can also have a number of tags associated with it that can help to retrieve it via the BigML API or to provide fusions with some extra information. Each tag is limited to a maximum of 128 characters. Each fusion can have up to 32 different tags.

6.9.5 Counters

For each fusion, BigML also stores a number of counters to track the number of other resources that have been created using the fusion as a starting point. In the fusion view, you can see a menu option that displays counters for evaluations, single and batch predictions, and the fusions created using this fusion. It also allows you to quickly jump to all the resources of each type (see Figure 6.75).
6.10 Fusion Privacy

Privacy options for a fusion can be defined in the More Info panel, displayed in Figure 6.76. There are two levels of privacy for BigML fusions:

- **Private**: only accessible by authorized users (the owner and those who have been granted access by him or her).

- **Shared**: by enabling the secret link you will get two different links to share your model. The first one is a sharing link that you can copy and send to others so they can visualize and interact with your model. The second one is a link to embed your model directly on your web page.

6.11 Moving Fusions to Another Project

When you create a fusion within a project, it will be assigned to this project. Fusions can only be assigned to a single project. However, you can move fusions between your Dashboard projects. The menu option to do this can be found in two places:

1. In the fusion view, click the MOVE TO... option within the 1-click action menu and select another project or create a new one (see Figure 6.77).
2. In the fusion list view, click the MOVE TO... option within the pop up menu and select another project or create a new one (see Figure 6.78).

Note: you cannot move fusions to an organization project since the fusion is composed by many models and it cannot be separated from them.

6.12 Stopping Fusions

You can stop the creation of a fusion before the task is finished by clicking the DELETE FUSION option from the 1-click action menu (see Figure 6.79), or from the pop up menu in the fusion list view (see...
A modal window will be displayed asking you for confirmation and if you want to also delete all the models composing the fusion. The models that belong to another fusion or to an OptiML will not be deleted. If you stop the fusion during its creation, you will not be able to resume the same task. If you want to create the same fusion, you will have to start a new task.
6.13 Deleting Fusions

You can delete your fusions by clicking on the DELETE FUSION option from the 1-click action menu (see Figure 6.82) or using the pop up menu on the fusion list (see Figure 6.83).
A modal window will be displayed asking you for confirmation and if you want to also delete all the models composing the fusion. The models that belong to another fusion or to an OptiML will not be deleted. Once a fusion or its models are deleted, they are permanently deleted, and there is no way you (or even the IT folks at BigML) can retrieve it.
6.14 Takeaways

This chapter explains fusions in detail. Here is a list of key points:

- A fusion is a supervised Machine Learning algorithm used to solve classification and regression problems.
- A fusion is built by selecting multiple models in BigML and using to make an evaluation, a single prediction, or a batch prediction (see Figure 6.85).
- You can create a fusion using models, ensembles, logistic regression and/or deepnets.
- You can assign different weights to the models composing a fusion.
- All models need to have the same objective field.
- The PDP view provides a visual way to analyze a field’s impact on predictions given certain values for the rest of the fields.
- For classification problems, you get all the objective field class probabilities along with the predicted class.
- For regression problems, you get the objective field predicted values.
- BigML lists all the models composing the fusion along with their weights.
- You need to evaluate your fusion’s performance using data that the fusion has not seen before.
- The ultimate goal in building a fusion is being able to make predictions with it.
- BigML allows you to quickly make predictions for single instances by providing a form containing the fields used by the fusion, so you can easily set the input data and get an immediate response.
- BigML batch predictions allow you to make simultaneous predictions for multiple instances. All you need is the fusion you want to use to make predictions and a dataset containing the instances for which you want to obtain predictions.
- You can configure your batch predictions output file settings.
- You can download your fusion to perform local predictions.
- You can add descriptive information to your fusion (name, description, tags, and category).
- You can move your fusions between projects.
- You can share your fusions with other people using the secret link.
- You can stop your fusions creation by deleting them.
- You can permanently delete an existing fusion.
Figure 6.85: Fusion Workflow
Evaluations

7.1 Introduction

BigML evaluations provide an easy way to measure and compare the performance of classification and regression models (i.e., models, ensembles, logistic regressions, deepnets, and fusions created using supervised learning algorithms). The main purpose of evaluations is twofold:

- First, obtaining an estimation of the model's performance in production (i.e., making predictions for new instances the model has never seen before).
- Second, providing a framework to compare models built using different configurations or different algorithms to help identify the models with best predictive performance.

The basic idea behind evaluations is to take some test data different from the one used to train a model and create a prediction for every instance. Then compare the actual objective field values of the instances in the test data against the predictions and compute several performance measures based on the correct results as well as the errors made by the model.

The usual way to obtain some test data is to split a dataset into two disjoint subsets: a training set and a test set. You can easily do this from your BigML Dashboard by using the 1-click menu option that automatically splits your dataset into a random 80% subset for training and 20% for testing, or if you prefer, you can also configure those percentages. Chapter 7 in Datasets with the BigML Dashboard document explains how to do this.

Depending on whether you evaluate a classification or a regression model different metrics and visualizations are provided as Subsection 7.2.1 and Subsection 7.2.2 describe. In BigML you can also perform cross-validation, another popular model evaluation technique explained in Subsection 7.3.6. Moreover BigML provides some tools so you can compare several evaluations built with different algorithms and configurations explained in Section 7.6.

BigML evaluations are first-class citizens. This means that they can be created via the BigML API and can also be queried automatically (you can find an example in Subsection 7.7.1). This allows you to automate workflows when you want to iteratively change your model parameters and see how the performance is altered.

The evaluations section in the BigML Dashboard is found in the third tab under the models’ menu (see Figure 7.1.) This section contains all your model’s evaluations ordered by creation date so most recent evaluations are found at the top of the list. Order your evaluations by Name, by Type (classification or regression and cross-validation icons below), by Performance (f-measure or r-squared explained in Subsection 7.2.1.3 and Subsection 7.2.2 respectively), by Age (time since the evaluation was created), Dataset Size (the testing dataset weight) and Instances (number of instances in the testing dataset). (See Figure 7.2.) The icons on the left of the evaluations names in the evaluation list view allow you to go to the original resources used to create the evaluation (the model, ensemble, logistic regression, deepnet, or fusion and the testing dataset).
In the evaluation list view, you can search evaluations by name by clicking the SEARCH menu option in the top right corner. You can also access the 1-click action menu shown in Figure 7.3:

- **Evaluate a model**: create a new evaluation by selecting a model and a testing dataset (see Subsection 7.3.1)
- **Evaluate an ensemble**: create a new evaluation by selecting an ensemble and a testing dataset (see Subsection 7.3.2)
- **Evaluate a logistic regression**: create a new evaluation by selecting a logistic regression and a testing dataset (see Subsection 7.3.3)
- **Evaluate a deepnet**: create a new evaluation by selecting a deepnet and a testing dataset (see Subsection 7.3.4)
- **Evaluate a fusion**: create a new evaluation by selecting a fusion and a testing dataset (see Subsection 7.3.5)
- **COMPARE EVALUATIONS**: compare two existing evaluations side by side (see Subsection 7.6.1)
- **COMPARE MULTIPLE EVALUATIONS**: compare multiple existing evaluations (see Subsection 7.6.2)

Note: time series evaluations are explained in the document Time Series with the BigML Dashboard [12].

A **Name**, **Description**, **Category**, and **Tags** are associated with each evaluation, which can be helpful to retrieve and document your projects. (See Section 7.9.) **Share** your evaluation with others by using the secret link (see Section 7.10.)

An evaluation is associated with the same **project** which the model, ensemble or logistic regression belongs to. **Move** an evaluation between projects (see Section 7.11,) or **delete** it permanently from your account. (See Section 7.13.)

Finally, you can see the corresponding icons used to represent a single **evaluation** and **cross-validation** evaluations in Figure 7.4 and Figure 7.5, respectively.

Figure 7.3: Evaluation list view 1-click action menu

Figure 7.4: Evaluations icon

Figure 7.5: Cross-validation icon

Depending on the evaluations **Type**, you can find the following icons in the **evaluation list view**.
7.2 Understanding Evaluations

In this section, we are going to describe the technicalities behind evaluations. The main goal of evaluating your model is to measure its **predictive performance**. BigML provides two different ways to measure your model performance: by creating single evaluations or cross-validation evaluations.

- **Single evaluations** are available for models, ensembles, logistic regressions, deepnets, and fusions. The basic idea is to make predictions for instances for which the objective field values are known, but the model has not seen before. The dataset containing those instances is called the testing dataset. Based on the comparison between the predicted and the actual values of the testing dataset, a set of **performance measures** are calculated. The usual way to obtain the testing dataset is to split the original dataset into two disjoint subsets: a training set and a test set. You can easily do this by using the 1-click menu option that automatically splits your dataset into a random 80% subset for training and 20% for testing. (Subsection 7.1 in [Datasets with the BigML Dashboard document](https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation) explains how to do this.)

- **Cross-validation** evaluations are available for models, ensembles, logistic regressions, and deepnets. In particular, BigML uses **k-fold cross-validation**. To create a cross-validation evaluation you just need a dataset as input. BigML then automatically splits your dataset in \( k \) complementary subsets. One of the subsets is used for evaluation while the rest of the data is used for training the model (the remaining \( k - 1 \) subsets). This process is performed \( k \) times, each time using different parts of the data for training and testing the models. Finally, cross-validation yields \( k \) different
Chapter 7. Evaluations

models and k evaluations. The results of the k evaluations are averaged to obtain the final performance measures. Cross-validation evaluations usually yield more accurate results than single evaluations since they avoid the potential error derived from randomly selecting a too optimistic testing dataset.

Both types of evaluations yield the same measures except for the evaluation curves, which will not be shown for the final averaged cross-validation. However, different performance measures will be obtained depending on the objective field type: numeric or categorical. In the following subsections you can find a detailed explanation of classification measures and regression measures provided by BigML. You can also find a brief subsection at the end mentioning the specificities of cross-validation evaluation measures.

7.2.1 Classification Measures

The performance of a classification model is usually represented by a confusion matrix, which contains the actual and the predicted values by the model. The evaluation measures are calculated based on the confusion matrix and the concept of positive and negative classes.

In the following subsections you can find a brief explanation of positive and negative concepts, the confusion matrix, the probability threshold, the classification measures and the evaluation curves provided by BigML.

7.2.1.1 Positive and Negative Classes

To understand the confusion matrix and the classification measures explained in Subsection 7.2.1.2 and Subsection 7.2.1.3 respectively, it’s important to have a general idea of the meanings of positive and negative classes.

In Machine Learning, by convention, the objective field classes are often referred to as positive and negative classes. The positive class is the class that is more important to accurately predict; e.g., if you are predicting cancer and you have two classes for the objective field, “true” and “false”, you can afford some mistakes in predicting when cancer is “false”, but it’s essential to identify the “true” cases. In other words, the positive class should be the one that has a greater cost of making a prediction error. The rest of classes will then be considered negative classes. The positive class is usually the minority class and the negative class is the majority class in binary classification problems because in many cases, it is often more interesting to predict the rare cases rather than the most evident and common ones.

Considering a class positive or negative is not related to its labels or meaning, so in many cases the positive class can be the “bad” outcome and the negative class the “good” outcome. For example, again consider disease diagnosis, where “true” indicates the patient has the disease, and it is usually considered the positive class.

7.2.1.2 Confusion Matrix

A common method to analyze the model predictive performance is the confusion matrix. The confusion matrix is a table containing the predictions and the actual values for the objective field classes so you can visualize the correct decisions as well as the errors made by the classifier.

In BigML, the columns represent the actual values and the rows represent the predictions for each of the classes by default. You can transpose rows by columns by clicking the switcher. (See Figure 7.10.)

The intersection between actual values and predicted values yields four possible situations:

- True Positives (TP): positive instances correctly classified
- False Positives (FP): negative instances classified as positive
- True Negatives (TN): negative instances correctly classified as non-positive
- False Negatives (FN): positive instances classified as negative

A model with n classes will yield a confusion matrix with n columns and n rows (see Figure 7.10). However, confusion matrices for the evaluation curves (see Subsection 7.2.1.5) will always yield 2 columns
and 2 rows because negative classes are aggregated as if they were one class (you can find more information here\(^2\)).

![Confusion Matrix example](image)

The cell colors in the confusion matrix indicate the TP, FP, TN and FN values. In the diagonal of the table you can find the correct predictions, the TP and TN.\(^3\) See Subsection 7.2.1.4 to find out how to select the positive class.

![Cell colors indicate the class selected as positive](image)

### 7.2.1.3 Classification Measures

BigML provides different metrics to measure your model's performance:

- Accuracy
- Precision
- Recall
- F-measure
- Phi Coefficient
- Macro-averages
- Kendall’s Tau
- Spearman’s Rho

\(^2\)https://en.wikipedia.org/wiki/Confusion_matrix

\(^3\)Note that all negative classes are aggregated so the TN may include incorrect predictions for the negative classes. E.g., imagine three classes, \(a\), \(b\) and \(c\) where the positive class is \(a\) while \(b\) and \(c\) are the negative classes. The instances of the \(b\) class incorrectly predicted as \(c\) as well as the instances of the \(c\) class predicted as \(b\) will be considered TN since they are “negative instances correctly classified as non-positive”.

Copyright © 2023, BigML, Inc.
You can find an explanation of each measure in the following subsections. All the measures except the Kendall’s Tau and the Spearman’s Rho are derived from the confusion matrices and they change according to the positive class and the threshold selected as explained in Subsection 7.2.1.4. Some of these measures are used to display the evaluation curves explained in Subsection 7.2.1.5.

7.2.1.3.1 Accuracy

Accuracy is calculated as the number of correctly classified instances over the total instances evaluated.

\[
\text{Accuracy} = \frac{TP + TN}{\text{Total instances}}
\]

Figure 7.12: Accuracy example

Accuracy remains a popular measure for model performance since it is very easy to calculate, but for many real life problems it is too simplistic and misleading. One of the most obvious is when the model has to deal with unbalanced classes. For example, suppose that we get 90% of accuracy in a binary classification model for which you have 900 instances for one of the classes and 100 for the other one. A 90% accuracy is reachable just by classifying all the 1,000 instances as the majority class. This is why it is very important to take into account two more measures, Precision and Recall (explained below).

7.2.1.3.2 Precision

Precision is the percentage of correctly predicted instances over the total instances predicted for the positive class. (See Figure 7.13.)

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

Figure 7.13: Precision example

7.2.1.3.3 Recall

Recall is the percentage of correctly classified instances over the total actual instances for the positive class. (See Figure 7.14.)

\[
\text{Recall} = \frac{TP}{TP + FN}
\]
7.2.1.3.4 F-measure

The F-measure, also called the F-score\(^4\), is the balanced harmonic mean between Precision and Recall. The F-measure is often a more useful metric than accuracy since poor performance in either Precision or Recall will result in a low F-measure value. It can range between 0 and 1. Higher values indicate better performance.

\[
F\text{-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

\(\text{Recall} = \frac{TP}{TP + FN}\)

\(\text{Precision} = \frac{TP}{TP + FP}\)

\(\text{F-measure} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}\)

---

\(^4\)https://en.wikipedia.org/wiki/F1_score

\(^5\)https://en.wikipedia.org/wiki/Matthews_correlation_coefficient
BigML also reports the **maximum phi coefficient**, which is the highest phi coefficient given all possible thresholds (see Subsection 7.2.1.4). You can find it as a mark in the threshold slider as shown in Figure 7.17.

To find out more about classification performance measures, refer to this [paper](#) written by BigML’s VP of Machine Learning algorithms, Charles Parker.

### 7.2.1.3.6 Macro-averages

As it is explained in the previous subsections, classification measures are computed per class, except in the case of **Accuracy** which is the only measure that is always computed for the overall model.

BigML computes the average of per class measures to measure the overall model performance. Those global statistics are called the **macro-averages** of the measures since they are computed by giving equal weight to all classes. You can find them in the **evaluation view** under the names of **Average Precision**, **Average Recall**, **Average F-Measure** and **Average Phi** as shown in Figure 7.18.

Read more on **macro-averaging** in this [paper](#).

### 7.2.1.3.7 Kendall’s Tau

The Kendall’s tau and the Spearman’s rho coefficients (see Subsection 7.2.1.3.8) are the only measures that do not have a formula based on confusion matrices. The Kendall’s tau is based on all possible
pair of rankings considering each instance in the testing dataset. It measures the degree of correlation between the ranked instances and it can take values between -1 and 1. You can find a more detailed explanation on how the Kendall’s tau coefficient is computed in this section.

Each instance in the testing dataset has an actual class for the objective field and a probability score result of the model predictions. The positive class is assigned a value of 1 and the negative class a value of 0. If for two different instances \( a \) and \( b \), \( \text{score}(a) > \text{score}(b) \) and \( \text{class}(a) > \text{class}(b) \) or \( \text{score}(a) < \text{score}(b) \) and \( \text{class}(a) < \text{class}(b) \), the pair of instances is “concordant” because the ordering of scores matches the actual classes, if it doesn’t, the pair is “discordant”.

Letting \( C \) be concordant pairs and \( D \) discordant pairs, the tau coefficient is calculated as follows:

\[
\text{Kendall's tau} = \frac{C - D}{C + D}
\]

BigML specifically computes Kendall’s tau-b coefficient\(^6\), which makes an adjustment in the denominator for pairs that are tied. A pair is tied when the \( \text{score}(a) = \text{class}(a) \) or \( \text{score}(b) = \text{class}(b) \). In this case, the pair is neither concordant nor discordant, so the coefficient denominator needs to be modified to keep the range of values \([-1, 1]\).

Figure 7.19: Kendall's Tau-b coefficient

### 7.2.1.3.8 Spearman’s Rho

Similar to the Kendall's tau explained in the previous subsection, the Spearman’s rho\(^7\) coefficient measures the degree of correlation between the model predictions and the testing dataset actual values. It computes the Pearson correlation coefficient\(^8\), between the ranks of the instances and it can take values between -1 and 1. Values closer to 1 indicate a perfect positive correlation which means a better

---

\(^6\)https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient#Tau-b  
\(^7\)https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient  
\(^8\)https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
performing model. On the other hand, closer values to -1 indicate a worse performing model. A value of 0 indicates that the model is no better than another model making random predictions. See the formula below:

$\text{Spearman’s rho} = \rho_{rg_X, rg_Y} = \frac{\text{cov}(rg_X, rg_Y)}{\sigma_{rg_X} \sigma_{rg_Y}}$

The $\text{cov}(rg_X, rg_Y)$ is the covariance of the ranked values and the $\sigma_{rg_X}$ and $\sigma_{rg_Y}$ are the standard deviations of the ranked values.

7.2.1.4 Confidence, Probability and Vote Thresholds

Classification models in BigML always return a confidence and/or a probability for each prediction, i.e. a percentage between 0% and 100% that measures the certainty of the prediction. See Subsection 1.2.6.2, Subsection 2.2.1.2, and Section 4.2 to know how BigML calculates probabilities for models, ensembles and logistic regression. Deepnets and fusions also return a probability measure.

When evaluating or predicting with your model you can set a probability or confidence threshold for a selected class, known as the positive class, so the model only predicts the positive class if the probability or the confidence is greater than the threshold set, otherwise it predicts the negative class. For example, imagine the following diabetes predictions for three different patients:

<table>
<thead>
<tr>
<th>Patients</th>
<th>Diabetes False</th>
<th>Diabetes True</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient 1</td>
<td>80%</td>
<td>20%</td>
</tr>
<tr>
<td>Patient 2</td>
<td>10%</td>
<td>90%</td>
</tr>
<tr>
<td>Patient 3</td>
<td>60%</td>
<td>40%</td>
</tr>
</tbody>
</table>

Table 7.1: Example of diabetes prediction probabilities for three different patients
Without setting any threshold, just by looking at the probabilities for each predicted class, patients 1 and 3 would be predicted as “False” and patient 2 as “True”. However, if we select “True” as the positive class and we set a probability threshold of 30%, patient 2 and 3 will be predicted as “True”, and only patient 1 will have a “False” prediction.

For decision forests (see Subsection 2.2.1), you can also set a vote threshold, i.e., a threshold based on the percentage of models in the ensemble voting for the positive class. The type of threshold (confidence, probability or vote threshold) can be configure before creating the evaluation (see Subsection 7.4.2).

Setting a threshold is specially useful when you want to minimize false positives at the cost of false negatives. By setting a probability or confidence threshold you can easily increase the minority class predictions.

Different thresholds produce different confusion matrices, hence different metrics for the same evaluation. These different metrics according to the threshold set can be seen in the evaluation curve views provided by BigML (see Subsection 7.2.1.5).

In some cases, it is likely that several thresholds yield the same matrix, above all for small test datasets. In BigML you can select any threshold and the positive class by using the options shown in the Figure 7.21. By setting different thresholds you will see the values for the confusion matrix and the metrics changing accordingly. Single points in the evaluation curves correspond to different thresholds.
Figure 7.21: Select the probability threshold and the positive class

The greater the threshold, fewer instances will be predicted for the positive class. By setting a \textbf{threshold of 100\%}, all instances are predicted as the negative class (see Figure 7.22).
On the other hand, by setting a 0% threshold all instances are predicted as the positive class (see Figure 7.23).
7.2.1.5 Evaluation curves

As explained in Subsection 7.2.1.4, setting different thresholds can result in different confusion matrices and different metrics. The best way to evaluate how your model performs for all the possible thresholds is to plot the metrics in different charts:

- Precision-Recall curve
- ROC curve
- Gain curve & K-S statistic
- Lift curve

Single points for each curve represent a probability threshold for the positive class selected.
7.2.1.5.1 Precision-Recall Curve

The Precision-Recall curve visually represents the trade-off\(^9\) between both measures for the positive class. Precision and recall are inversely related, i.e., for the same model you can increase recall using a lower threshold for the positive class, but it will usually result in a decrease in precision, and vice versa. You can find the formulas of both measures in Subsection 7.2.1.3.

A high precision and a high recall are represented by points near the upper right corner of the chart (1,1), thus the greater the area under the precision-recall curve the better. BigML provides two different area calculations:

- **PR AUC**: the Area Under the Curve (AUC) is calculated taking into account the exact curve shape (Figure 7.24).

  ![Figure 7.24: Area Under the Curve for the Precision-Recall curve](image)

- **PR AUCH**: the Area Under the Convex Hull is calculated taking into account the convex shape of the curve where no other points lay above the curve. You can visualize it by clicking the option shown in Figure 7.25,

  ![Figure 7.25: Area Under the Convex Hull for the Precision-Recall curve](image)

\(^9\)https://en.wikipedia.org/wiki/Precision_and_recall
The appropriate balance between precision and recall needs to be decided in a per-case basis according to the costs associated with false positives and false negatives.

### 7.2.1.5.2 ROC Curve

The ROC space graphically represents the existing trade-off between the recall (or sensitivity) and specificity for classification problems. The recall is obtained by calculating the True Positive Rate (TPR), i.e. the ratio of instances that has been correctly classified for the positive class. The False Positive Rate (FPR), equivalent to 1-specificity, is the percentage of negative class instances that have been incorrectly classified.

You can obtain the TPR and FPR by normalizing the confusion matrix results:

\[
\text{TPR} = \text{Recall} = \frac{TP}{TP + FN} \\
\text{FPR} = \frac{FP}{TN + FP}
\]

---

Figure 7.26: The ROC curve

The diagonal of the chart divides the space; all the points found in the upper left part of the chart (where TPR>FPR) can be considered good results, and all those found in the diagonal or below (where TPR=<FPR) are bad results. The diagonal represents a model that has the same performance as choosing a class for each point at random.

Similar to the precision-recall curve, the Area Under the Curve\(^\text{11}\) (AUC) for an evaluation, is the area beneath the evaluation’s ROC curve in the ROC space (see Figure 7.27). Higher AUC values indicate a better classifier performance; however, in extreme cases, such as AUC=1, it may reflect an overfitting problem.

\(^{11}\)https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve
You can also visualize the **Area Under the Convex Hull** by clicking the highlighted option in Figure 7.30.
7.2.1.5.3 Gain Curve & K-S statistic

The Gain curve (or Cumulative Gain curve) represents the relationship between the percentage of correct predictions for the positive class and the effort needed to achieve them measured as the percentage of instances predicted. The y-axis in the Gain curve is equivalent to the recall as well as the True Positive Rate (TPR) and the x-axis is the percentage of positive class instances. The formulas for these metrics are:

$$\text{Gain} = \text{Recall} = \text{TPR} = \frac{TP}{TP + FN}$$

$$\% \text{ of Positive Instances} = \frac{TP + FP}{TP + FP + TN + FN}$$

Similar to the ROC curve, the diagonal of the chart represents the results of a random model. All the points above the diagonal can be considered good results. The closer a point is to the upper left corner (0,1), the better.

Along with the Gain curve, BigML also provides in the same chart the Negative Cumulative Response curve (represented by the black curve shown in Figure 7.29). The Negative Cumulative Response curve represents the percentage of instances incorrectly predicted as positive, so it is equivalent to the False Positive Rate (FPR) as explained in the section on ROC curves (Subsection 7.2.1.5.2):

$$\text{Negative Cumulative Response} = \text{FPR} = \frac{FP}{TN + FP}$$
Along with the Gain and Negative Cumulative response curve, BigML provides the calculation of the **Kolomogorov Smirnov statistic (K-S statistic)**. It measures the maximum difference between the TPR and the FPR over all possible thresholds:

$$K-S\ statistic = \text{max}\ (TPR - FPR)$$

The K-S statistic is an indicator of how well the model separates the positive from the negative classes. A K-S statistic of 100% indicates a perfect separation and a model that classifies everything correctly. Higher values for the K-S statistic indicate a higher quality model.

![Figure 7.29: The Gain curve](image)

### 7.2.1.5.4 Lift Curve

The Lift curve shows the goodness of fit of your model compared to a random class assignment given a sample of positive instances. The Lift is plotted in the y-axis and it is calculated as the ratio between the result predicted by your model and the result using no model. The x-axis represents again the percentage of correct predictions. The formulas for these metrics are:

$$\text{Lift} = \frac{\text{Precision}}{\text{PositiveInstances/TotalInstances}} = \frac{TP}{TP + FP}$$

$$\%\ of\ Positive\ Instances = \frac{TP + FP}{TP + FP + TN + FN}$$
The horizontal line in the chart indicating a 100% lift (see Figure 7.30) represents a model that makes random predictions.

For a more detailed explanation of the Gain and Lift charts, please refer to this article\textsuperscript{12}.

### 7.2.2 Regression Measures

When the objective field of the model, ensemble, deepnet, or fusion is numeric the resulting evaluation includes the regression measures explained below.

#### 7.2.2.1 Mean Absolute Error

The Mean Absolute Error is the mean of the model prediction errors for each instance. It is computed as the average of the absolute values of the differences between the target variable predicted by the model ($y'$) vs. the actual values ($y$). Letting $N$ be the total number of instances evaluated, then:

$$\text{Mean Absolute Error} = \frac{1}{N} \sum_{n} |y'_n - y_n|$$

#### 7.2.2.2 Mean Squared Error

The Mean Squared Error is similar to the Mean Absolute Error, but the differences between predictions and actual values are squared. It is computed as the average of the squares of the differences between

\textsuperscript{12}https://www.analyticsvidhya.com/blog/2016/02/7-important-model-evaluation-error-metrics/
the target variable predicted by the model \( (y') \) vs. the actual values \( (y) \). Letting \( N \) be the total number of instances evaluated, then:

\[
\text{Mean Square Error} = \frac{\sum_n (y'_n - y_n)^2}{N}
\]

### 7.2.2.3 R Squared

The \( R^2 \), also called the coefficient of determination\(^{13}\), measures how much better the model is than always predicting the mean value of the target variable \( (\bar{y}) \) in the test set. It can take values up to 1. Values below 0 indicate the model is worse than predicting the mean; a value of 0 means the model is not any better than predicting the mean; and 1 means the model perfectly fits the data. Although an \( R^2 = 1 \) may not necessarily be desirable (since it can be a symptom of overfitting), higher values for \( R^2 \) usually mean better performance.

\[
R^2 = 1 - \frac{\sum_n (y'_n - y_n)^2}{\sum_n (y_n - \bar{y}_n)^2}
\]

### 7.2.3 Cross-Validation Measures

BigML cross-validation yields \( k \) different models and \( k \) evaluations. To get a single estimation of the model’s performance, the results of the \( k \) evaluations are averaged to obtain the final cross-validation measures (per class and overall measures). Consequently, cross-validation evaluations have the same measures as single classification and regression evaluations. (See Subsection 7.2.1 and Subsection 7.2.2.)

Additionally, apart from the averages, you will also find the standard deviation for each classification and regression measure. (See Subsection 7.5.4.)

Cross-validation evaluations do not include the evaluation curves, but will in a future release.

### 7.3 Creating Evaluations

The process to create an evaluation is different if you want to create a single evaluation or a cross-validation evaluation:

- To create a single evaluation, you need two resources: a testing dataset (different from the one used for training) and a model, an ensemble, a logistic regression, deepnet, or fusion. All three processes follow a similar logic. You can find a separate explanation of each one in the following subsections.

- To create a cross-validation evaluation, you just need a dataset. BigML allows you to create cross-validation for models, ensembles, logistic regressions, deepnets and fusions. This process is explained in Subsection 7.3.6

### 7.3.1 Model Evaluations

To evaluate a model, you can use any of the following options from the BigML Dashboard:

- Click Evaluate A Model in the 1-click action menu from the evaluation list view. (Figure 7.31.)

---

\(^{13}\)https://en.wikipedia.org/wiki/Coefficient_of_determination
This option redirects you to the **New Evaluation** view where you need to select a model and a testing dataset. (See Figure 7.32.) From this view you can also select an ensemble by clicking the ensemble icon above the model selector.

- Click **EVALUATE** in the 1-click action menu from the model view. (See Figure 7.33.)
Alternatively, click **EVALUATE** in the **pop up menu** from the model list view (see Figure 7.34).

By using any of these options, you will be redirected to the **New Evaluation** view where the model will be pre-filled in the selector and you only have to choose the testing dataset. If you previously split your original dataset into two subsets (one for training and another for testing) using the 1-click menu option from your dataset view, BigML will automatically select the corresponding testing dataset. Finally, click the **Evaluate** green button to perform the evaluation. (See Figure 7.35.)
7.3.2 Ensemble Evaluations

To evaluate an ensemble you can use the following options from the BigML Dashboard:

- Click **Evaluate an Ensemble** in the 1-click action menu from the evaluation list view (Figure 7.36).

This option takes you to the **New Evaluation** view where you need to select an ensemble and a testing dataset. (See Figure 7.37.) From this view, you can also select a model by clicking the model icon above the ensemble selector.
Chapter 7. Evaluations

520

• Click **EVALUATE** in the 1-click action menu from the ensemble view- (Figure 7.38.)

![Figure 7.38: Evaluate ensemble from 1-click action menu](image)

Alternatively, click **EVALUATE** in the **pop up menu** from the ensembles list view (Figure 7.39).

![Figure 7.39: Evaluate ensemble from pop up menu](image)

By using any of these two options, you will be redirected to the **New Evaluation** view where the ensemble will be pre-filled and you only have to choose the testing dataset. If you previously split
your original dataset into two subsets (one for training and another for testing) using the 1-click menu option from your dataset view, BigML will automatically select the corresponding testing dataset. Finally, click the **Evaluate** green button to perform the evaluation. (See Figure 7.40.)

![Figure 7.40: Evaluation with pre-filled ensemble and dataset information](image)

### 7.3.3 Logistic Regression Evaluations

To evaluate a logistic regression, you can use these options from the BigML Dashboard:

- **Click Evaluate a Logistic Regression** from the 1-click action menu from the evaluation list view. (See Figure 7.41.)

![Figure 7.41: Evaluate logistic regression from evaluation list view](image)

This option redirects you to the **New Evaluation** view where you need to select a logistic regression and a testing dataset. (See Figure 7.42.)
Chapter 7. Evaluations

Figure 7.42: Select logistic regression and dataset

- Click **Evaluate** from the logistic regression **1-click action menu**. (Figure 7.43.)

Figure 7.43: Evaluate logistic regression from 1-click action menu

Alternatively, click **Evaluate** in the **pop up menu** from the logistic regression list view. (Figure 7.44.)

Figure 7.44: Evaluate logistic regression from pop up menu

By using any of these options, you will be redirected to the **New Evaluation** view where the logistic regression will be pre-filled in the selector and you only need to choose the testing dataset. If you previously split your original dataset into two subsets (one for training and another for testing)
using the 1-click menu option from your dataset view, BigML will automatically select the corresponding testing dataset. Finally, click the Evaluate green button to perform the evaluation. (See Figure 7.45.)

7.3.4 Deepnet Evaluations

To evaluate a deepnet, you can use these options from the BigML Dashboard:

- Click Evaluate a Deepnet from the 1-click action menu from the evaluation list view. (See Figure 7.46.)

This option redirects you to the New Evaluation view where you need to select a deepnet and a testing dataset. (See Figure 7.47.)
Figure 7.47: Select deepnet and dataset

- Click **EVALUATE** from the deepnet 1-click action menu. (See Figure 7.48.)

Figure 7.48: Evaluate deepnet from 1-click action menu

Alternatively, click **EVALUATE** in the **pop up menu** from the deepnet list view. (Figure 7.49.)
By using any of these options, you will be redirected to the New Evaluation view where the deepnet will be pre-filled in the selector and you only need to choose the testing dataset. If you previously split your original dataset into two subsets (one for training and another for testing) using the 1-click menu option from your dataset view, BigML will automatically select the corresponding testing dataset. Finally, click the Evaluate green button to perform the evaluation. (See Figure 7.50.)

7.3.5 Fusion Evaluations

To evaluate a fusion, you can use these options from the BigML Dashboard:

- Click Evaluate a fusion from the 1-click action menu from the evaluation list view. (See Figure 7.51.)
Figure 7.51: Evaluate fusion from evaluation list view

This option redirects you to the **New Evaluation** view where you need to select a fusion and a testing dataset. (See Figure 7.52.)

Figure 7.52: Select fusion and dataset

- Click **Evaluate** from the fusion **1-click action menu**. (See Figure 7.53.)
Alternatively, click Evaluate in the pop up menu from the fusion list view. (Figure 7.54.)

By using any of these options, you will be redirected to the New Evaluation view where the fusion will be pre-filled in the selector and you only need to choose the testing dataset. Finally, click the Evaluate green button to perform the evaluation. (See Figure 7.55.)
7.3.6 Cross-Validation Evaluations

In BigML, you can use k-fold cross-validation\(^{14}\) to evaluate your models, ensembles, logistic regressions, and deepnets. Cross-validation evaluations are implementing in BigML as a WhizzML script and they can be found in BigML Gallery:

1. Go to the scripts Gallery\(^{15}\) where you will find five different scripts to perform cross-validation:
   - Basic 5-fold cross-validation\(^{16}\): performs a 5-fold cross-validation for models with default model configuration options. Learn the default options for models in Subsection 7.4.5.2. (See Figure 7.56.)

---

\(^{14}\)https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation

\(^{15}\)https://bigml.com/gallery/scripts

\(^{16}\)https://bigml.com/dashboard/script/57989125af447f2234000003
• **Model’s k-fold cross-validation**\(^{17}\): performs cross-validation for models. You can configure the \(k\)-fold parameter and the model inputs. Learn about the configurable inputs for models in Subsection 7.4.5.2. (See Figure 7.57.)

\(^{17}\)https://bigml.com/dashboard/script/579e2973af447f2234000602

• **Ensemble’s k-fold cross-validation**\(^{18}\): performs cross-validation for ensembles. You can configure the \(k\)-fold parameter and the ensemble inputs. Learn about the configurable inputs

\(^{18}\)https://bigml.com/dashboard/script/579e9115af447f2234000000
for ensembles in Subsection 7.4.5.3. (See Figure 7.58.)

- **Logistic regression's k-fold cross-validation**\(^{19}\): performs cross-validation for logistic regression. You can configure the \(k\)-fold parameter and the logistic inputs. Learn about the configurable inputs for logistic regressions in Section 4.4. (See Figure 7.59.)

---

\(^{19}\)https://bigml.com/dashboard/script/5aba720eeba31d2795001d6f
• **Deepnet’s k-fold cross-validation**\(^{20}\): performs cross-validation for deepnets. You can configure the \(k\)-fold parameter and the deepnet inputs. Learn about the configurable inputs for deepnets in Section 5.4. (See Figure 7.60.)

![Deepnet’s k-fold cross-validation](https://bigml.com/user/whizzml/gallery/script/5a9ed849eba31d3b480005b1)

**Figure 7.60: Deepnet’s k-fold cross-validation**

2. By clicking the script preview you can inspect script’s details such as the **source-code**, the **input** and the **output**. (See Figure 7.57.) You can find additional documentation about WhizzML scripts [here]\(^{21}\).

---

\(^{20}\)https://bigml.com/user/whizzml/gallery/script/5a9ed849eba31d3b480005b1

\(^{21}\)https://bigml.com/whizzml
3. **Clone** your preferred script for FREE. You can clone it from the script preview by clicking the **FREE** button. (See **Figure 7.62**.)
Alternatively, you can clone it from the script view by clicking the FREE or Buy buttons. (See Figure 7.63.)

A modal window will appear asking you for confirmation. (See Figure 7.64.)
4. Once you clone the script, you will be redirected to the **Execution view** to set your inputs. You need to **select a dataset** and optionally, you can configure the rest of the inputs. If you do not configure them, they will take the default values. You can find an explanation of all your inputs in **Subsection 7.4.5.** (See **Figure 7.65**.)

![Figure 7.65: Configure cross-validation inputs](image)

5. Once you have selected the dataset, click **Execute**. (See **Figure 7.66**.)
6. Once you execute the script, you can check the progress of your script in the execution view where you will find the elapsed time, the total resources generated and the script log messages. (See Figure 7.67.)

7. Finally, cross-validation yields $k$ different models and $k$ different evaluations. The results of the single evaluations are averaged to obtain the final model performance measures. Access the final cross-validation evaluation containing the averaged measures by clicking on the evaluation ID.
link in the Outputs section. Read more about cross-validation measures in Subsection 7.2.3. The $k$ intermediary resources can be found in the same view under the Resources panel. (See Figure 7.68.)

You can perform again any new cross-validation by clicking on your cloned script listed under the scripts tab in the BigML Dashboard. (See Figure 7.69.)

7.4 Evaluation Configuration Options

BigML allows you to configure different options for single evaluations and for cross-validation evaluations. For cross-validation, the configuration options are called inputs.

In the following subsections you will find an explanation of each single evaluation configuration options. The last subsection (Subsection 7.4.5) details cross-validation inputs.

For single evaluations, all the options are found under the configuration panel that appears once you select the model and the testing dataset as shown in Figure 7.70.
Note: some configuration options may change depending on whether you are evaluating a model, an ensemble, a logistic regression, a deepnet, or a fusion.

7.4.1 Missing Strategies

This option is only available for models, ensembles, and fusions that contain models and/or ensembles; for logistic regressions and deepnets the option to include missing values is configured when training the model.

When the testing dataset contains instances with missing values for models and ensembles, BigML can take two different approaches to return a prediction for those instances:

- **Last prediction**: when a missing value is found in the testing data for a decision node, the prediction returned will be the one from the parent of the missing split. (See Figure 7.71.)

  ![Last prediction icon](image)

  Figure 7.71: Last prediction icon

- **Proportional missing strategy**: when a missing value is found in the testing data for a decision node, the prediction is calculated based on all subtrees predictions, taking into account the proportion of data in each subtree. (See Figure 7.72.)

  ![Proportional missing strategy icon](image)

  Figure 7.72: Proportional missing strategy icon

BigML uses **Last prediction** as the default strategy. Select your preferred option by clicking the corresponding icon shown in Figure 7.73.
Chapter 7. Evaluations

7.4.2 Select the Voting Strategy for Decision Forests

For Decision Forests ensembles, you can select the voting strategy you want to calculate the evaluation. See a complete explanation about the three options that BigML provides in Subsection 2.6.3.2. Selecting a given voting strategy will also determine the threshold to calculate the curves (see Subsection 7.2.1.4).

- **Probability**: For classification ensembles per-class probabilities are averaged taking into account all trees composing the ensemble. The class with higher probability is the winner class. For regression ensembles the probability option averages the predictions of the trees composing the ensemble.

- **Confidence**: For classification ensembles per-class confidences are averaged taking into account all trees composing the ensemble. The class with higher confidence is the winner class. For regression ensembles the confidence option averages the predictions of the trees composing the ensemble.

- **Votes**: For classification ensembles each tree prediction is considered as one vote. The “votes” of a given class is the percentage of trees in the ensemble that vote for that class. For regression ensembles the votes option averages the predictions of the trees composing the ensemble. It gives the same results as the probability.

Note: if the model or ensemble has been trained with Missing splits (learn more about this parameter in Ensembles chapter in Subsection 1.4.4), the Missing strategy may not have any impact since missing values would already be considered by the model or the ensemble as valid values.
Figure 7.74: Select the voting strategy to calculate the evaluation

Note: for single decision tree evaluations you can also choose between probabilities and confidences to calculate the thresholds of the evaluation curves.

7.4.3 Fields Mapping
You can specify which fields in the model, ensemble, logistic regression, deepnet, or fusion match with the fields in the testing dataset. BigML automatically matches fields by name, but you can also set an automatic match by field ID by clicking on the green switcher shown in Figure 7.75. You can also manually search for fields or remove them from the Dataset fields column if you do not want them to be considered during the evaluation.
Note: the fields mapping from the BigML Dashboard has a limit of 200 fields; for evaluations with a higher number of fields, you can use from the BigML API\textsuperscript{22} if you need to map your fields.

### 7.4.4 Sampling Your Dataset

Sometimes you do not need all the data contained in your testing dataset to generate your evaluations. If you have a very large dataset, sampling may be a good way of getting faster results. BigML allows you to take a sample before creating an evaluation so you do not need to create a different dataset. You can configure the sampling options detailed in the following subsections. (See Figure 7.76.)

#### 7.4.4.1 Rate

The Rate is the proportion of instances to include in your sample. You can set any value between 0% and 100%. Defaults to 100%.

#### 7.4.4.2 Range

Specifies a subset of instances from which to sample, e.g., choose from instance 1 until 200. The Rate you set will be computed over the Range configured. This option may be useful when you have temporal data, and you want to train your model with historical data, and test it with the most recent one to check if it can predict based on time.

#### 7.4.4.3 Sampling

By default, BigML selects your instances for the sample by using a random number generator, which means two samples from the same dataset will likely be different even when using the same rates and row ranges. If you choose deterministic sampling, the random-number generator will always use the same seed, thus producing repeatable results. This lets you work with identical samples from the same dataset.

#### 7.4.4.4 Replacement

Sampling with replacement allows a single instance to be selected multiple times. Sampling without replacement ensures that each instance cannot be selected more than once. By default, BigML generates

\textsuperscript{22}https://bigml.com/api/evaluations#ev_evaluation_arguments
samples without replacement.

### 7.4.4.5 Out of bag

This argument will create a sample containing only out-of-bag instances for the currently defined rate. If an instance is not selected as part of a sample, it is considered out of bag. Thus, the final total percentage of instances for your sample will be 100% minus the rate configured for your sample (when replacement is false). This can be useful for splitting a dataset into training and testing subsets. It is only electable when a sample rate is less than 100%.

![Sampling options for evaluations](image)

**Figure 7.76: Sampling options for evaluations**

### 7.4.5 Cross-Validation Configuration Options

Depending on the cross-validation script, you will be able to configure different inputs. As it was mentioned in Subsection 7.3.6, you can find three different types of cross-validation scripts in the BigML Gallery. You can find a separate explanation of each one in the following subsections.

#### 7.4.5.1 Basic 5-fold cross-validation

Basic 5-fold cross-validation script only has one configurable input: the **dataset-id**. (See Figure 7.77.) You can click on the selector and search the dataset by name. Once you have selected the dataset you can execute the script and it automatically splits your dataset into $k = 5$ subsets creating five different models and five evaluations using default inputs. Next Subsection 7.4.5.2 explains the default inputs for models.
7.4.5.2 Model’s k-fold cross-validation

You can configure the following inputs for your model’s k-fold cross-validation (see Figure 7.78):

- **dataset-id**: select an existing dataset by clicking on the selector and searching the dataset by name.
- **k-folds**: set the number of partitions for your dataset. Your dataset will be split into \( k \) subsets creating \( k \) different models and \( k \) different evaluations. By default \( k=5 \), a \( k=10 \) is also commonly used.
- **objective-id**: set the ID of the field you want to be the objective field. You can find the field ID in your dataset view by mousing over the field type. It can be a categorical or numeric field. If no ID is given, BigML takes the dataset default objective field. (See Subsection 1.4.1.)
- **missing-splits**: set this input to true to consider missing values as valid values in your model. By default, it is set to false. (See Subsection 1.4.4.)
- **stat-pruning**: apply statistical pruning to all the tree nodes in order to avoid overfitting. By default, this option is disabled and the smart pruning is enabled. (See Subsection 1.4.3.)
- **balance-objective**: enable this option to let BigML automatically balance the classes of the objective field. This is only available for classification models. By default this option is disabled. (See Subsection 1.4.6.1.)
- **weight-field**: weigh the instances considering the values of one field. You need to input the field ID. The selected field must be numerical and it must not contain missing values. This is valid for both regression and classification models. (See Subsection 1.4.6.3.)
- **objective-weights**: set a specific weight for each class of the objective field. You need to list the weights in the same order that classes are found in your dataset histogram, e.g., to weight red, blue and yellow classes you need to input \([2,3,1]\). If a class is not listed, it is assumed to have a weight of 1. Weights of 0 are also valid. This option is only available for classification models. (See Subsection 1.4.6.2.)
- **node-threshold**: set a threshold for the nodes so the model stops growing. You can set a value between 3 and 2,000. By default it is set to -1 which indicates that no threshold applies. This parameter is also useful to avoid overfitting. (See Subsection 1.4.5.)
7.4.5.3 Ensemble’s k-fold cross-validation

You can configure the following inputs for your ensemble’s k-fold cross-validation (see Figure 7.79):

- **dataset-id**: select an existing dataset by clicking on the selector and searching the dataset by name.

- **k-folds**: set the number of partitions for your dataset. Your dataset will be split into $k$ subsets creating $k$ different ensembles and $k$ different evaluations. By default $k=5$, a $k=10$ is also commonly used.

- **objective-id**: set the ID of the field you want to be the objective field. You can find the field ID in your dataset view mousing over the field type. It can be a categorical or numeric field. If no ID is given, BigML takes the dataset default objective field. (See Subsection 2.4.1.)

- **number-of-models**: total number of models in the ensemble. You can choose between 2 and 1,000 models. By default it is set to 10 models. (See Subsection 2.4.4.)

- **missing-splits**: set this input to true to consider missing values as valid values in your ensemble. By default it is set to false. (See Subsection 2.4.6.2.)

- **stat-pruning**: apply statistical pruning to all the tree nodes in order to avoid overfitting. By default this option is disabled and the smart pruning is enabled. (See Subsection 1.4.3.)

- **balance-objective**: enable this option to let BigML automatically balance the classes of the objective field. This is only available for classification ensembles. By default this option is disable. (See Subsection 2.4.8.1.)

- **weight-field**: weigh the instances considering the values of one field. You need to input the field ID. The selected field must be numerical and it must not contain missing values. This is valid for both regression and classification ensembles. (See Subsection 2.4.8.3.)

- **objective-weights**: set a specific weight for each class of the objective field. You need to list the weights in the same order that classes are found in your dataset histogram, e.g., to weight red, blue and yellow classes you need to input [2,3,1]. If a class is not listed, it is assumed to have a
weight of 1. Weights of 0 are also valid. This option is only available for classification ensembles. (See Subsection 2.4.8.2.)

- **node-threshold**: when the number of computed nodes is greater than this threshold, model growth stops. You can set a value between 3 and 2,000. By default it is set to -1 which indicates that no threshold applies. This parameter is useful to avoid overfitting. (See Subsection 2.4.6.3.)

- **sample-rate**: set a percentage of the dataset to build each single tree. By default it is set to 100% with replacement. (See Subsection 2.4.9.1.)

- **replacement**: enable this parameter to allow a single instance to be selected multiple times. Sampling without replacement ensures that each instance cannot be selected more than once. For ensembles, you need to set this parameter to true if your sampling rate is 100% to ensure single trees are built using different subsets of your dataset. (See Subsection 2.4.9.3.)

- **randomize**: set this input to true to use Random Decision Forests algorithm instead of Bagging as the algorithm to build the ensemble. (See Subsection 2.4.3.)

- **seed**: write any character to randomize the ensemble sampling. By leaving this input in blank the random-number generator will always use the same seed, producing repeatable results. By default it is configured to create random samples. (See Subsection 2.4.9.)

7.5 Visualizing Evaluations

After evaluations are created, you can visualize them in the BigML Dashboard. Different visualizations are provided for classification (see Subsection 7.5.2) and regression (see Subsection 7.5.3) evaluations. Both types of evaluations shared the original resources information at the top of the view (see Subsection 7.5.1).

Cross-validation visualizations are very similar to single evaluations, the main differences are explained in Subsection 7.5.4.
7.5.1 Original Resources Information

For single evaluations, the original model and testing dataset used to create the evaluation will appear at the top of the evaluations view along with their corresponding icons so you can come back to see the resources. (See Figure 7.80.)

![Figure 7.80: Example of an ensemble evaluation top view information](image)

The information for each resource includes some of the parameter values used to create them:

- For **models**, **ensembles**, **logistic regressions** and **deepnets**:
  - Model type (regression or classification).
  - Objective field name.
  - Model size.
  - Number of fields: used to build the model.
  - Number of instances: used to build the model.
  - Description (see Section 1.10.)
  - Sampling and ordering options (see Subsection 1.4.7 and Subsection 1.4.8.)
    - Instances: number of sampled instances.
    - Sample rate: percentage of the training dataset used to build the model.
    - Range: instances selected to build the model.
    - Sampling: deterministic or random.
    - Replacement: true or false.
    - Out of bag: true or false.
    - Ordering: linear, random or deterministic.

- **Ensembles** also include:
  - Type: Decision Forests or Boosted Trees (See Subsection 2.4.3)
  - Number of models: total single models in the ensemble

- For testing **datasets**:
– Dataset size
– Number of fields: in the testing dataset
– Number of instances: in the testing dataset
– Description: see Subsection 7.9.2
– Sampling and ordering options (See Subsection 7.4.4.)
  * Instances: number of sampled instances
  * Sample rate: percentage of the testing dataset used to create the evaluation
  * Range: instances selected to sample the testing dataset
  * Sampling: deterministic or random.
  * Replacement: true or false.
  * Out of bag: true or false.
  * Ordering: linear, random or deterministic.

7.5.2 Classification Evaluations

For classification models, BigML provides several views that are accessible by clicking in the corresponding icons at the top menu.

7.5.2.1 General Evaluation

This first visualization provides the general confusion matrix which contains the correctly classified instances as well as the errors made by the model for each of the objective field classes without applying any threshold.

In the table rows you can find the predictions for the positive class and the negative classes, and in the columns the actual instances in the testing dataset. You can transpose rows by columns by clicking on the switcher shown in Figure 7.81. You can also download the confusion matrix in Excel format by clicking the option shown in Figure 7.81.
If you select a positive class in the selector, the confusion matrix cells will be painted according to the True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). By hovering a class in the table you get also these colors Figure 7.82.
In the metrics below the confusion matrix you will see the performance measures for the class selected as the positive class such as **Precision, Recall, F-Measure, Phi Coefficient** and **Accuracy**. If you select “All classes”, you will get the averages for the overall model. You can also compared these metrics to other two types of models: one using the **mode** as its prediction and the other predicting a **random** class of the objective field. At the very least, you would expect your model to outperform these weak benchmarks. (See Figure 7.83).
The confusion matrix will display up to **five different classes**. If your model contains more than five classes in the objective field, you can download the confusion matrix in Excel format by clicking the option shown in Figure 7.84.

**Figure 7.83: Performance metrics and benchmark models view**

**Figure 7.84: Download confusion matrix**
7.5.2.2 Evaluation curves

BigML offers four different evaluation curves: the ROC curve, the Precision-Recall curve, the Gain curve and the Lift curve. All of them are explained in detail in Subsection 7.2.1.5.

You can select the positive class and the threshold for each curve and you will see the confusion matrix values and the metrics changing in real-time (see Figure 7.85).

The confusion matrices for the curves have always two columns and two rows although there are more than two classes in the objective field. This is because the positive class is the one predicted given a certain threshold and if the threshold is not met, the rest of classes are aggregated as a single “Negative class” and predicted instead. Read more about positive and negative classes in Subsection 7.2.1.2.

From each curve visualization you can:

- **Download** the confusion matrix in Excel format
- **Download** the chart in PNG format with or without legends.
Chapter 7. Evaluations

7.5.3 Regression Evaluations

You can visualize the regression measures in the green boxed histograms: **Mean Absolute Error**, **Mean Squared Error** and **R Squared**. Read more about regression measures in Subsection 7.2.2.
Figure 7.87: Performance measures for regression models, ensembles, deepnets, and fusions

By default, BigML provides the measures of two other types of models to compare against your model performance. One of them uses the mean as its prediction and the other predicts a random value in the range of the objective field. At the very least, you would expect your model to outperform these weak benchmarks. You can remove the benchmarks by clicking the icons shown in Figure 7.88.

Figure 7.88: Compare regression models against random and mode values

7.5.4 Cross-Validation Visualization

Since cross-validation measures are calculated by averaging the single \( k \) evaluations, you cannot find the confusion matrix and evaluation curves explained in Subsection 7.5.2. Instead, the main metrics are plot in different histograms. Additionally, you can see the standard deviation per measure under the sigma symbol icon. You can find a cross-validation example for a classification model below.

Copyright © 2023, BigML, Inc.
At the top of the view, you can find the list of the single evaluations used to compute the cross-validation measures by clicking on the Evaluations panel. (See Figure 7.90.)
7.6 Evaluation Comparison

Evaluation comparison allows you to identify the algorithms and configurations that yield the highest performance. In BigML you can compare two evaluations side by side or multiple evaluations simultaneously. Both options are explained in the following subsections.

7.6.1 Compare Evaluations Side By Side

In BigML, you can easily compare two evaluations side by side by clicking COMPARE EVALUATION in the 1-click action menu option from the evaluation view. (Figure 7.91). This option is deprecated for the new evaluations that include the evaluation curves explained in Subsection 7.2.1.5.

![Figure 7.91: Compare evaluations using 1-click action menu](image)

Alternatively, you can use the COMPARE EVALUATIONS menu option, from the pop up menu. (Figure 7.92.)
You can also click on the **COMPARE EVALUATIONS** list view from the **1-click action menu** in the evaluation list view. (Figure 7.93).

Using any of the options you will be redirected to the **Compare Evaluations** view where you will be able to select the evaluations you want to compare. (Figure 7.94.)
Note: you can only compare two evaluations that use the same testing dataset and the same evaluation configuration; otherwise evaluations are not comparable.

The performance measures of your selected evaluations will be displayed side by side. BigML automatically computes the variation of the right-hand-side evaluation compared to the left-hand-side one. You can find those variations next to each performance measure of the right-hand-side evaluation. You can also select another evaluations from the evaluations selectors. (See Figure 7.95.)

7.6.2 Compare Multiple Evaluations

The main goal of comparing multiple evaluations is to analyze different model performances by plotting their evaluation results in a chart. BigML evaluation comparison offers a flexible and visual way to compare multiple classification models built using different algorithms (models, ensembles, logistic regression, deepnets, or fusions) and/or different configurations as long as the testing dataset is the same. Cross-validation evaluations are not eligible for the Evaluation comparison tool since they use different testing datasets to compute the averaged measures.

You can compare multiple classification evaluations using the option from the 1-click action menu as shown in Figure 7.96. Previously built evaluations to the evaluation curves explained in Subsection 7.2.1.5 do not show this option anymore.
You can also click on the COMPARE MULTIPLE EVALUATIONS list view from the 1-click action menu in the evaluation list view. (Figure 7.97).

A chart displaying the curves for the current evaluation will be displayed. You can select any of the four evaluation curves offered by BigML: Precision-Recall curve, ROC curve, Gain curve and Lift curve. (See Figure 7.98.) You can find a detailed explanation of the evaluation curves in Subsection 7.2.1.5.
You can plot different evaluations in the chart by clicking the evaluation selector shown in Figure 7.99. You will only be able to select other comparable evaluations, i.e., other evaluations that were built using the same testing dataset. BigML allows to compare up to 150 evaluations simultaneously selecting them in groups of 20 evaluations from the selector.
Your selected evaluations will appear in the chart with different colors. The evaluation curves and metrics will be calculated according to the positive class selected. You can select to sort your evaluations in the legend by any of the available metrics: the ROC AUC, the PR AUC, the K-S statistic, the Kendall's Tau or the Spearman's Rho. In the legend, for each evaluation, you will have the name, the icon of the model used, the sorting metric, and the configuration parameters used to create the model. By hovering over the evaluations in the legend you will be able to hide or remove them at any time.
Figure 7.100: Manage evaluations from the legend

Apart from the ROC AUC and PR AUC metrics shown in the legend for, you will also be able to see the PR AUCH and ROC AUCH by hovering over the evaluation names. Again, refer to Subsection 7.2.1.5 for a detailed explanation about the AUCH (Are Under the Convex Hull).
Figure 7.101: See the PR AUCH and ROC AUCH by hovering over the evaluation name in the table.

If you select the Gain curve you will also obtain the **K-S statistic** by hovering over the evaluation name in the legend. (See Subsection 7.2.1.5 for a detailed explanation).
Figure 7.102: See the K-S statistic by hovering over the evaluation name in the table.

The panel below the chart, contains a table with all the selected evaluations, their metrics and a set of model parameters to help you identify which algorithm and configuration performs better (see Figure 7.103):

- **ROC AUC**: the Area Under the Curve of the ROC curve.
- **PR AUC**: the Area Under the Curve of the Precision-Recall curve.
- **K-S**: the K-S statistic of the Gain curve.
- **Tau**: the Kendall’s tau coefficient.
- **Rho**: the Spearman’s rho coefficient.
- **Model Type**: if the resource evaluated is a single model, ensemble, logistic regression, deepnet, or fusion.
- **Number of nodes**: maximum number of splits set for training the model or ensemble.
- **Number of models**: number of models set for training the ensemble. For models, it will always be 1.
- **% of Bagging**: percentage of the dataset instances used for training the single trees composing the ensemble.
- **Randomize**: if the single trees composing an ensemble take a random sample of fields per split.
- **Boosted**: if the ensemble is composed of Boosted trees.
- **Balanced**: if the model or ensemble has been built previously balancing the objective field.
- **Missing splits**: if the model or ensemble has been built taking into account the missing values in the dataset.
- **Default Numeric**: if the logistic regression has been built replacing the missing values by the field mean, maximum, minimum or zero.
• **Auto-scaled**: if the fields in the logistic regression have been auto-scaled.

• **Bias**: if the logistic regression includes the bias term.

You can find a detailed explanation of each configuration parameter depending if the model is a single tree (Section 1.4), an ensemble (Section 2.4), a logistic regression (Section 4.4), or a deepnet (Section 5.4) in the corresponding sections.

![Figure 7.103: Table containing all the evaluations and the model parameters](image-url)

### 7.6.2.1 Exporting Evaluation Comparison

You can export the multiple comparison chart in PNG format with or without legends by clicking the icon shown in Figure 7.104. The legend includes the evaluations names, the selected sorting metric and the model configuration options.
7.7 Consuming Evaluations

BigML allows you to create and consume your evaluations programmatically via the **BigML API and bindings**. The following subsections explain both tools.

### 7.7.1 Using Evaluations Via the BigML API

You can perform all the evaluation actions explained in this document via the BigML API such as creating, configuring, retrieving, listing, updating, and deleting evaluations.

See below how to create an evaluation just using a model and a testing dataset once you have properly set the `BIGML_AUTH` environment variable to contain your authentication credentials:

```
curl "https://bigml.io/evaluation?$BIGML_AUTH" \
  -X POST \
  -H "content-type: application/json" \
  -d '{"dataset": "dataset/50650bdf3c19201b64000020", "model": "model/50650bea3c19201b64000024"}';
```

Apart from all the BigML Dashboard actions, you can also create an evaluation using **multiple models** and **multiple datasets** via the BigML API:

- Provide multiple model IDs, and the models will be evaluated as an ensemble and you can select your preferred voting strategy option. (See Subsection 7.4.2.)
- Provide a list of datasets identifiers, and the corresponding datasets will be concatenated and used as input. You can also sample the resulting dataset. (See Subsection 7.4.4.)
Chapter 7. Evaluations

For more information on using evaluations through the BigML API, please refer to documentation.

7.7.2 Using Evaluations Via the BigML Bindings

You can create and use evaluations via BigML bindings which are libraries aimed to make it easier to use the BigML API from your language of choice. BigML offers bindings for several languages including Python, Node.js, Java, Swift or Objective-C. You can find an example to create an evaluation with Python bindings below.

```python
from bigml.api import BigML
api = BigML()
evaluation = api.create_evaluation('model/57506c472275c1666b004b10',
                                 'dataset/576d213d983efc63e8000038')
```

For more information on BigML bindings, please refer to the bindings page.

7.8 Evaluation Limits

Similarly to the rest of resources in BigML, evaluations have some limits in the number of classes, terms, and items allowed per field:

- **Classes**: a maximum number of 1,000 distinct classes per field is allowed.
- **Terms**: BigML can handle up to 1,000 tokens in total. In case multiple text fields are defined, then the token limit per field is evenly divided by the number of text fields, e.g., a dataset with two text fields would result in 500 terms per text field. BigML selects those terms with most significant frequency, discarding both those that appear either too often or too infrequently. Moreover, a maximum of 256 characters are allowed.
- **Items**: a maximum number of 10,000 distinct items per field is allowed.

7.9 Descriptive Information

Each evaluation has an associated name, description, category, and tags. The following subsections provide a brief description for each concept. See the options that the More info menu option gives to edit them (Figure 7.105).

![Figure 7.105: Panel to edit an evaluation name, description, category, and tags](https://bigml.com/tools/bindings)
7.9.1 Evaluation Name

Each evaluation name is displayed on the list view and also on the top bar of an evaluation view. Evaluation names are indexed to be used in searches. When you create an evaluation, it gets a default name. You can change it using the MORE INFO menu option on the right corner of the evaluation view (see Figure 7.105). The name of an evaluation cannot be longer than 256 characters. More than one evaluation can have the same name even within the same project, but they will always have different identifiers.

7.9.2 Evaluation Description

Each evaluation also has a description that it is very useful for documenting your Machine Learning projects. Evaluations take the description of the models used to create them.

Descriptions can be written using plain text and also markdown. BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 7.106.)

![Figure 7.106: Markdown editor for evaluations descriptions](https://en.wikipedia.org/wiki/Markdown)

Descriptions cannot be longer than 8192 characters and can use almost any character.

7.9.3 Evaluation Category

A category, taken from the model used to create the evaluation, is associated with each evaluation. Categories are useful to classify evaluations according to the domain which your data comes from. This is useful when you use BigML to solve problems across industries or multiple customers.

An evaluation category must be one of the categories listed on Table 7.2.

---

25[https://en.wikipedia.org/wiki/Markdown](https://en.wikipedia.org/wiki/Markdown)
Table 7.2: Categories used to classify evaluations by BigML

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace and Defense</td>
</tr>
<tr>
<td>Automotive, Engineering and Manufacturing</td>
</tr>
<tr>
<td>Banking and Finance</td>
</tr>
<tr>
<td>Chemical and Pharmaceutical</td>
</tr>
<tr>
<td>Consumer and Retail</td>
</tr>
<tr>
<td>Demographics and Surveys</td>
</tr>
<tr>
<td>Energy, Oil and Gas</td>
</tr>
<tr>
<td>Fraud and Crime</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Higher Education and Scientific Research</td>
</tr>
<tr>
<td>Human Resources and Psychology</td>
</tr>
<tr>
<td>Insurance</td>
</tr>
<tr>
<td>Law and Order</td>
</tr>
<tr>
<td>Media, Marketing and Advertising</td>
</tr>
<tr>
<td>Miscellaneous</td>
</tr>
<tr>
<td>Physical, Earth and Life Sciences</td>
</tr>
<tr>
<td>Professional Services</td>
</tr>
<tr>
<td>Public Sector and Nonprofit</td>
</tr>
<tr>
<td>Sports and Games</td>
</tr>
<tr>
<td>Technology and Communications</td>
</tr>
<tr>
<td>Transportation and Logistics</td>
</tr>
<tr>
<td>Travel and Leisure</td>
</tr>
<tr>
<td>Uncategorized</td>
</tr>
<tr>
<td>Utilities</td>
</tr>
</tbody>
</table>

### 7.9.4 Evaluation Tags

An evaluation can also have a number of tags associated with it that can help to retrieve it via the BigML API or to provide evaluations with some extra information. An evaluation inherits the tags from the model used to create it. Each tag is limited to a maximum of 128 characters. Each evaluation can have up to 32 different tags.

### 7.10 Evaluation Privacy

The link displayed in the Privacy panel is the private URL of your evaluation, so only a user logged into your account is able to see it.

BigML allows you to share your evaluations by enabling the secret link. Just click the switcher icon, copy the link and share it with others.
Chapter 7. Evaluations

7.11 Moving Evaluations to Another Project

When you create an evaluation, it will be assigned to the same project where the original model is located. However, you can move evaluations between projects from two different places:

- Clicking the MOVE TO... option in the 1-click action menu from the evaluation view:

![Figure 7.108: Move evaluation from the 1-click action menu](image)

- Clicking the MOVE TO... option in the pop up menu from the evaluation list view:
7.12 Stopping Evaluation Creation

You can stop the creation of an evaluation before the task is finished by clicking the DELETE EVALUATION option from the 1-click action menu from the evaluation view (Figure 7.110), or from the pop up menu on the evaluation list view (see Figure 7.111).
Note: if you stop the evaluation during its creation you will not be able to resume the same task. If you want to create the same evaluation you will have to start a new task.

7.13 Deleting Evaluation

You can delete your evaluations by clicking the DELETE EVALUATION option from the 1-click action menu from the evaluation view, (Figure 7.112).

Alternatively, you can click the DELETE EVALUATION option in the pop up menu on the evaluation list view (Figure 7.113).
A modal window will be displayed asking you for confirmation. Once an evaluation is deleted, it is permanently deleted, and there is no way you (or even the IT folks at BigML) can retrieve it.

7.14 Takeaways

This chapter explains evaluations in detail. Here is a list of key points:

- An evaluation allows you to measure your model, ensemble, logistic regression, deepnet, and fusion performance.

- In BigML you can perform two types of evaluations: single evaluations and cross-validation evaluations.

- You need a model and a testing dataset to create a single evaluation. (See Figure 7.115.)

- You just need an existing dataset to create a cross-validation evaluation. (See Figure 7.116.)

- BigML provides you a range of configuration options before creating your evaluation.

- Performance measures are different for classification and regression models.

- The confusion matrix is a key element to evaluate the performance of classification models.

- You can compare your evaluations measures against models using the mean, the mode, and a random value to predict.

- BigML provides different visualizations for the ROC curve, the Precision-Recall curve, the Gain curve, and the Lift curve along with their AUC, K-S statistic and other metrics.

- You can compare two or more evaluations built with different configurations and algorithms to select the model with the best performance.
• You can **download** your confusion matrix in Excel format.
• You can create and use evaluations via the **BigML API and bindings**.
• You can add **descriptive information** to your evaluations.
• You can **move** your evaluations between projects.
• You can **share** your evaluations with other people using the secret link.
• You can **stop** your evaluations creation by deleting them.
• You can permanently **delete** an existing evaluation.

Figure 7.115: Single evaluations workflow

Figure 7.116: Cross-validation workflow
List of Figures

1.1 Models list view ........................................ 2
1.2 Empty Dashboard models view ........................................ 2
1.3 Models icon ........................................ 2
1.4 Example of confidences in two different nodes of the same tree ........................................ 6
1.5 Example of the expected error at a node ........................................ 8
1.6 A dataset with images and image features ........................................ 9
1.7 A dataset with image feature fields shown ........................................ 9
1.8 1-click model ........................................ 10
1.9 1-click model from the dataset list view ........................................ 10
1.10 Configure permanent parameter modal ........................................ 11
1.11 Configure model ........................................ 12
1.12 Automatic optimization ........................................ 13
1.13 Configure the missing splits and the weights for your model ........................................ 13
1.14 Training duration ........................................ 14
1.15 Model candidates ........................................ 14
1.16 Model configuration options ........................................ 15
1.17 Weighting arguments for models ........................................ 17
1.18 Sampling arguments for models ........................................ 18
1.19 Ordering argument for models ........................................ 19
1.20 Create model after configuration ........................................ 20
1.21 Model API request preview ........................................ 20
1.22 Tree visualization ........................................ 21
1.23 Node prediction ........................................ 22
1.24 Prediction path ........................................ 23
1.25 Interactive filters ........................................ 24
1.26 Create dataset from a branch ........................................ 25
1.27 Model PDP view ........................................ 26
1.28 Classification model PDP ........................................ 27
1.29 Regression model PDP ........................................ 28
1.30 Model PDP axis options ........................................ 29
1.31 Prediction legend options ........................................ 30
1.32 Input fields from in the model PDP ........................................ 31
1.33 Sunburst visualization ........................................ 32
1.34 Change Sunburst colors ........................................ 32
1.35 Sunburst colors ........................................ 33
1.36 Button to display a model’s summary report ........................................ 33
1.37 Field importance ........................................ 34
1.38 Summary Report ........................................ 35
1.39 Predictions list empty view ........................................ 35
1.40 Menu options of the predictions list view ........................................ 36
1.41 Single predictions icon ........................................ 36
1.42 Batch predictions icon ........................................ 36
1.43 Predictions list view ........................................ 36
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.44</td>
<td>Menu options to create predictions</td>
<td>37</td>
</tr>
<tr>
<td>1.45</td>
<td>Predict Question by Question first step</td>
<td>37</td>
</tr>
<tr>
<td>1.46</td>
<td>Predict Question by Question second step</td>
<td>37</td>
</tr>
<tr>
<td>1.47</td>
<td>Predict Question by Question third step</td>
<td>38</td>
</tr>
<tr>
<td>1.48</td>
<td>Predict option from model 1-click menu</td>
<td>38</td>
</tr>
<tr>
<td>1.49</td>
<td>Predict option from model pop-up menu</td>
<td>39</td>
</tr>
<tr>
<td>1.50</td>
<td>Single predictions form</td>
<td>39</td>
</tr>
<tr>
<td>1.51</td>
<td>Select fields in the prediction form</td>
<td>40</td>
</tr>
<tr>
<td>1.52</td>
<td>Single predictions view</td>
<td>41</td>
</tr>
<tr>
<td>1.53</td>
<td>Single predictions list view</td>
<td>41</td>
</tr>
<tr>
<td>1.54</td>
<td>Select a single image source in the image input field</td>
<td>42</td>
</tr>
<tr>
<td>1.55</td>
<td>List the components of a composite source</td>
<td>43</td>
</tr>
<tr>
<td>1.56</td>
<td>Select a component of a composite source</td>
<td>44</td>
</tr>
<tr>
<td>1.57</td>
<td>Model prediction form, image field and more</td>
<td>44</td>
</tr>
<tr>
<td>1.58</td>
<td>Model image single prediction</td>
<td>45</td>
</tr>
<tr>
<td>1.59</td>
<td>Batch predictions option from model 1-click menu</td>
<td>46</td>
</tr>
<tr>
<td>1.60</td>
<td>Batch predictions option from model pop up menu</td>
<td>46</td>
</tr>
<tr>
<td>1.61</td>
<td>Select dataset for batch predictions</td>
<td>47</td>
</tr>
<tr>
<td>1.62</td>
<td>Configuration options displayed and output preview</td>
<td>47</td>
</tr>
<tr>
<td>1.63</td>
<td>Create dataset from batch predictions</td>
<td>48</td>
</tr>
<tr>
<td>1.64</td>
<td>Create batch predictions</td>
<td>49</td>
</tr>
<tr>
<td>1.65</td>
<td>Download batch prediction output CSV file</td>
<td>49</td>
</tr>
<tr>
<td>1.66</td>
<td>View batch predictions output dataset</td>
<td>50</td>
</tr>
<tr>
<td>1.67</td>
<td>Batch prediction using an image dataset</td>
<td>50</td>
</tr>
<tr>
<td>1.68</td>
<td>Missing strategies for single predictions</td>
<td>51</td>
</tr>
<tr>
<td>1.69</td>
<td>Missing strategies for batch predictions</td>
<td>52</td>
</tr>
<tr>
<td>1.70</td>
<td>Select probability or confidence</td>
<td>53</td>
</tr>
<tr>
<td>1.71</td>
<td>Select the positive class</td>
<td>53</td>
</tr>
<tr>
<td>1.72</td>
<td>Set a threshold</td>
<td>54</td>
</tr>
<tr>
<td>1.73</td>
<td>Configure a threshold for batch predictions</td>
<td>54</td>
</tr>
<tr>
<td>1.74</td>
<td>Default numeric value for batch predictions</td>
<td>55</td>
</tr>
<tr>
<td>1.75</td>
<td>Fields Mapping for batch predictions</td>
<td>56</td>
</tr>
<tr>
<td>1.76</td>
<td>Output settings for batch predictions</td>
<td>57</td>
</tr>
<tr>
<td>1.77</td>
<td>Single predictions view for classification</td>
<td>58</td>
</tr>
<tr>
<td>1.78</td>
<td>Single predictions view for regression models</td>
<td>58</td>
</tr>
<tr>
<td>1.79</td>
<td>Explain prediction</td>
<td>59</td>
</tr>
<tr>
<td>1.80</td>
<td>Input field importances</td>
<td>60</td>
</tr>
<tr>
<td>1.81</td>
<td>See the detailed explanation</td>
<td>60</td>
</tr>
<tr>
<td>1.82</td>
<td>Download batch prediction output CSV file</td>
<td>61</td>
</tr>
<tr>
<td>1.83</td>
<td>An example of a batch prediction CSV file</td>
<td>61</td>
</tr>
<tr>
<td>1.84</td>
<td>View batch predictions output dataset</td>
<td>62</td>
</tr>
<tr>
<td>1.85</td>
<td>Batch predictions output dataset</td>
<td>62</td>
</tr>
<tr>
<td>1.86</td>
<td>Batch prediction 1-click actions</td>
<td>63</td>
</tr>
<tr>
<td>1.87</td>
<td>Edit predictions</td>
<td>64</td>
</tr>
<tr>
<td>1.88</td>
<td>Markdown editor for evaluations descriptions</td>
<td>65</td>
</tr>
<tr>
<td>1.89</td>
<td>Private link of a prediction</td>
<td>66</td>
</tr>
<tr>
<td>1.90</td>
<td>Stop prediction from the 1-click menu</td>
<td>67</td>
</tr>
<tr>
<td>1.91</td>
<td>Stop prediction from the predictions list view</td>
<td>67</td>
</tr>
<tr>
<td>1.92</td>
<td>Delete prediction from the 1-click menu</td>
<td>68</td>
</tr>
<tr>
<td>1.93</td>
<td>Delete prediction from pop up menu</td>
<td>68</td>
</tr>
<tr>
<td>1.94</td>
<td>Delete prediction confirmation</td>
<td>68</td>
</tr>
<tr>
<td>1.95</td>
<td>Download your model</td>
<td>69</td>
</tr>
<tr>
<td>1.96</td>
<td>Download branch rules</td>
<td>69</td>
</tr>
<tr>
<td>1.97</td>
<td>Panel to edit a model's name, category, description and tags</td>
<td>71</td>
</tr>
<tr>
<td>1.98</td>
<td>Markdown editor for model descriptions</td>
<td>72</td>
</tr>
<tr>
<td>1.99</td>
<td>Menu option to quickly access to resources created with a model</td>
<td>74</td>
</tr>
<tr>
<td>1.100</td>
<td>Models privacy options within the More Info menu option</td>
<td>74</td>
</tr>
<tr>
<td>1.101</td>
<td>Black Box models let other BigML users make predictions</td>
<td>75</td>
</tr>
</tbody>
</table>
2.1 Ensembles list view .................................................. 86
2.2 Empty Dashboard ensembles view ................................. 86
2.3 Ensemble icon ....................................................... 87
2.4 Field importance for ensembles ................................. 88
2.5 A dataset with images and image features ..................... 89
2.6 A dataset with image feature fields shown .................... 90
2.7 1-click ensemble from dataset detail view ..................... 90
2.8 1-click ensemble from dataset list view ....................... 91
2.9 Configure permanent parameter modal ......................... 92
2.10 Configure ensemble ............................................... 92
2.11 Ensemble objective field ...................................... 93
2.12 Automatic optimization ....................................... 94
2.13 Configure the missing splits and weights for your ensemble ........................................................................ 95
2.14 Training duration .................................................. 96
2.15 Ensemble candidates .............................................. 96
2.16 Ensemble type: Decision Forests or Boosted Trees ........... 97
2.17 Number of models ................................................. 98
2.18 Number of iterations ............................................. 98
2.19 Trees parameters configuration ................................ 100
2.20 Boosting parameters ............................................. 101
2.21 Weighting parameters .......................................... 103
2.22 Trees sampling for ensembles ................................ 104
2.23 Dataset sampling arguments for ensembles ................. 106
2.24 Ordering options for ensembles ............................... 107
2.25 Create ensemble after configuration ......................... 108
2.26 Ensemble API request preview ............................... 108
2.27 Ensemble top menu .............................................. 109
2.28 Switch from the chart view to the model list view .......... 110
2.29 Trees slider and resampling option for ensembles PDP ...... 111
2.30 Classification and regression ensembles ....................... 111
2.31 Ensemble chart ................................................... 112
2.32 Ensemble CHART options .................................... 113
2.33 PREDICTION legend options ............................... 114
2.34 INPUT FIELDS form in ensemble chart ....................... 115
2.35 Download ensemble chart in PNG format .................... 115
2.36 Both selected fields impact predictions ...................... 116
2.37 One of the selected fields impact predictions .............. 117
2.38 None of the selected fields impact predictions ............ 118
2.39 Ensemble model list view ...................................... 119
<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.40</td>
<td>Tree visualization</td>
<td>119</td>
</tr>
<tr>
<td>2.41</td>
<td>Predictions list empty view</td>
<td>120</td>
</tr>
<tr>
<td>2.42</td>
<td>Menu options of the predictions list view</td>
<td>120</td>
</tr>
<tr>
<td>2.43</td>
<td>Single predictions icon</td>
<td>120</td>
</tr>
<tr>
<td>2.44</td>
<td>Batch predictions icon</td>
<td>120</td>
</tr>
<tr>
<td>2.45</td>
<td>Predictions list view</td>
<td>121</td>
</tr>
<tr>
<td>2.46</td>
<td>Menu options to create predictions</td>
<td>121</td>
</tr>
<tr>
<td>2.47</td>
<td>Predict option from ensemble 1-click menu</td>
<td>122</td>
</tr>
<tr>
<td>2.48</td>
<td>Predict option from ensemble pop-up menu</td>
<td>122</td>
</tr>
<tr>
<td>2.49</td>
<td>Single predictions form</td>
<td>123</td>
</tr>
<tr>
<td>2.50</td>
<td>Select fields in the prediction form</td>
<td>124</td>
</tr>
<tr>
<td>2.51</td>
<td>Single prediction view</td>
<td>125</td>
</tr>
<tr>
<td>2.52</td>
<td>Single predictions list view</td>
<td>126</td>
</tr>
<tr>
<td>2.53</td>
<td>Select a single image source in the image input field</td>
<td>127</td>
</tr>
<tr>
<td>2.54</td>
<td>List the components of a composite source</td>
<td>127</td>
</tr>
<tr>
<td>2.55</td>
<td>Select a component of a composite source</td>
<td>128</td>
</tr>
<tr>
<td>2.56</td>
<td>Ensemble prediction form, image field and more</td>
<td>128</td>
</tr>
<tr>
<td>2.57</td>
<td>Ensemble single image prediction</td>
<td>129</td>
</tr>
<tr>
<td>2.58</td>
<td>Batch predictions option from ensemble 1-click menu</td>
<td>130</td>
</tr>
<tr>
<td>2.59</td>
<td>Batch predictions option from ensemble pop up menu</td>
<td>130</td>
</tr>
<tr>
<td>2.60</td>
<td>Select dataset for batch predictions</td>
<td>131</td>
</tr>
<tr>
<td>2.61</td>
<td>Configuration options displayed and output preview</td>
<td>131</td>
</tr>
<tr>
<td>2.62</td>
<td>Create dataset from batch predictions</td>
<td>132</td>
</tr>
<tr>
<td>2.63</td>
<td>Create batch predictions</td>
<td>133</td>
</tr>
<tr>
<td>2.64</td>
<td>Download batch prediction output CSV file</td>
<td>133</td>
</tr>
<tr>
<td>2.65</td>
<td>View batch predictions output dataset</td>
<td>134</td>
</tr>
<tr>
<td>2.66</td>
<td>Batch prediction using an image dataset</td>
<td>134</td>
</tr>
<tr>
<td>2.67</td>
<td>Missing strategies for single predictions</td>
<td>135</td>
</tr>
<tr>
<td>2.68</td>
<td>Missing strategies for batch predictions</td>
<td>136</td>
</tr>
<tr>
<td>2.69</td>
<td>Probabilities, confidences or votes for Decision Forest single predictions</td>
<td>138</td>
</tr>
<tr>
<td>2.70</td>
<td>Probabilities, confidences or votes for Decision Forest batch predictions</td>
<td>139</td>
</tr>
<tr>
<td>2.71</td>
<td>Select probability, confidence or votes</td>
<td>140</td>
</tr>
<tr>
<td>2.72</td>
<td>Select the positive class</td>
<td>140</td>
</tr>
<tr>
<td>2.73</td>
<td>Set a threshold</td>
<td>141</td>
</tr>
<tr>
<td>2.74</td>
<td>Configure a threshold for batch predictions</td>
<td>142</td>
</tr>
<tr>
<td>2.75</td>
<td>Default numeric value for batch predictions</td>
<td>142</td>
</tr>
<tr>
<td>2.76</td>
<td>Fields Mapping for batch predictions</td>
<td>143</td>
</tr>
<tr>
<td>2.77</td>
<td>Output settings for batch predictions</td>
<td>145</td>
</tr>
<tr>
<td>2.78</td>
<td>Single predictions view for classification ensembles</td>
<td>146</td>
</tr>
<tr>
<td>2.79</td>
<td>Single predictions view for regression ensembles</td>
<td>146</td>
</tr>
<tr>
<td>2.80</td>
<td>Single predictions view for regression Boosted Trees</td>
<td>147</td>
</tr>
<tr>
<td>2.81</td>
<td>Explain prediction</td>
<td>148</td>
</tr>
<tr>
<td>2.82</td>
<td>Input field importances</td>
<td>148</td>
</tr>
<tr>
<td>2.83</td>
<td>See the detailed explanation</td>
<td>149</td>
</tr>
<tr>
<td>2.84</td>
<td>Download batch prediction output file</td>
<td>150</td>
</tr>
<tr>
<td>2.85</td>
<td>An example of a batch prediction output file</td>
<td>150</td>
</tr>
<tr>
<td>2.86</td>
<td>View batch predictions output dataset</td>
<td>151</td>
</tr>
<tr>
<td>2.87</td>
<td>Batch predictions output dataset</td>
<td>151</td>
</tr>
<tr>
<td>2.88</td>
<td>Batch prediction 1-click actions</td>
<td>152</td>
</tr>
<tr>
<td>2.89</td>
<td>Edit predictions</td>
<td>153</td>
</tr>
<tr>
<td>2.90</td>
<td>Markdown editor for evaluations descriptions</td>
<td>154</td>
</tr>
<tr>
<td>2.91</td>
<td>Private link of a prediction</td>
<td>155</td>
</tr>
<tr>
<td>2.92</td>
<td>Stop prediction from the 1-click menu</td>
<td>156</td>
</tr>
<tr>
<td>2.93</td>
<td>Stop prediction from the predictions list view</td>
<td>156</td>
</tr>
<tr>
<td>2.94</td>
<td>Delete prediction from the 1-click menu</td>
<td>157</td>
</tr>
<tr>
<td>2.95</td>
<td>Delete prediction from popup menu</td>
<td>157</td>
</tr>
<tr>
<td>2.96</td>
<td>Delete prediction confirmation</td>
<td>157</td>
</tr>
<tr>
<td>2.97</td>
<td>Download your ensemble</td>
<td>158</td>
</tr>
</tbody>
</table>
2.98 Panel to edit an ensemble's name, category, description and tags .......................... 159
2.99 Markdown editor for ensemble descriptions ......................................................... 160
2.100 Menu option to quickly access to resources created with an ensemble .................... 161
2.101 Ensemble privacy options .................................................................................... 162
2.102 Menu option to move ensembles ........................................................................... 162
2.103 Menu option to move ensembles from the ensemble list view ................................. 163
2.104 Menu option to stop an ensemble creation ......................................................... 163
2.105 Menu option to stop an ensemble's creation ....................................................... 164
2.106 Menu option to delete an ensemble ..................................................................... 164
2.107 Ensemble deletion pop up menu option ............................................................... 164
2.108 Ensemble deletion modal window ....................................................................... 165
2.109 Model Workflows ............................................................................................... 167

3.1 Linear regression list view ....................................................................................... 169
3.2 Empty Dashboard linear regression view ............................................................... 169
3.3 Linear Regression icon .......................................................................................... 169
3.4 Missing numeric coefficients at the end of linear regression table ......................... 171
3.5 Create 1-click linear regression from dataset 1-click action menu ......................... 173
3.6 Create 1-click linear regression from dataset popup menu ..................................... 173
3.7 Configure linear regression .................................................................................... 174
3.8 Configure the objective field to create the linear regression .................................... 175
3.9 Change the default objective field ........................................................................ 175
3.10 Automatic optimization ....................................................................................... 176
3.11 Configure the default numeric value ..................................................................... 176
3.12 Training duration ................................................................................................. 177
3.13 Linear regression candidates ............................................................................... 177
3.14 Select a default numeric value to replace missing numeric values ....................... 178
3.15 Weight options for linear regression ..................................................................... 179
3.16 Bias parameter ..................................................................................................... 180
3.17 Field coding configuration .................................................................................... 181
3.18 Enable field coding configuration ......................................................................... 182
3.19 Select the dummy class ....................................................................................... 182
3.20 Field codings: dummy .......................................................................................... 183
3.21 Close modal window ............................................................................................. 183
3.22 Field codings configured ....................................................................................... 184
3.23 Disable field coding configuration ........................................................................ 184
3.24 Dummy class in table view ................................................................................... 185
3.25 Field coding configuration .................................................................................... 186
3.26 Enable field coding configuration ......................................................................... 187
3.27 Field codings: contrast coding ............................................................................. 187
3.28 Set the contrast coding values for each class ....................................................... 188
3.29 Contrast coding saved .......................................................................................... 189
3.30 Close modal window ............................................................................................. 189
3.31 Field codings configured ....................................................................................... 190
3.32 Disable field coding configuration ........................................................................ 191
3.33 Contrast icon in table view ................................................................................... 191
3.34 Contrast modal window in table view ................................................................... 192
3.35 Field coding configuration .................................................................................... 193
3.36 Enable field coding configuration ......................................................................... 193
3.37 Field codings: other coding .................................................................................. 194
3.38 Set the other coding values for each class ............................................................ 194
3.39 Other coding saved ............................................................................................... 195
3.40 Close modal window ............................................................................................. 195
3.41 Field codings configured ....................................................................................... 196
3.42 Disable field coding configuration ........................................................................ 197
3.43 Other coding in coefficients table ........................................................................ 197
3.44 Other coding modal window ................................................................................ 198
3.45 Sampling parameters for linear regression .......................................................... 199
3.46 Ordering argument for linear regression ........................................ 200
3.47 Create linear regression after configuration ................................ 200
3.48 Linear regression API request preview ........................................ 201
3.49 Switch chart, PDP and table views ............................................. 202
3.50 linear regression chart parts .................................................... 202
3.51 1D chart ............................................................................. 203
3.52 Partial Dependence Plot .......................................................... 204
3.53 Prediction legend ................................................................. 205
3.54 Configure the values for other input fields ................................. 205
3.55 Reset the values for the input fields ........................................ 206
3.56 Export chart as image with or without legends ......................... 207
3.57 Table view for linear regression ................................................. 208
3.58 Multiple field variables for categorical dummy encoded fields .... 209
3.59 Missing numeric coefficients at the end of linear regression table 210
3.60 Significant icon ................................................................. 210
3.61 Non-significant icon ............................................................. 210
3.62 Select significance level ........................................................ 211
3.63 Significance icons for coefficient estimates ............................... 211
3.64 Summary of stats per coefficient ............................................. 212
3.65 Search and filter linear regression table ................................... 213
3.66 Export table in CSV file ....................................................... 213
3.67 Predictions list view ............................................................ 214
3.68 Empty predictions list view .................................................... 214
3.69 Menu options of the predictions list view ................................. 215
3.70 Single predictions icon .......................................................... 215
3.71 Batch predictions icon .......................................................... 215
3.72 Predict using the 1-click action menu ...................................... 216
3.73 Predict using the pop up menu .............................................. 216
3.74 Linear regression prediction form ........................................... 217
3.75 Get the linear regression prediction ....................................... 218
3.76 Save your linear regression predictions .................................. 219
3.77 Create batch prediction using 1-click action menu ................. 219
3.78 Create batch prediction using pop up menu ............................ 220
3.79 Select dataset for batch prediction ......................................... 220
3.80 Configuration options for linear regression batch prediction ..... 221
3.81 Create a dataset from batch prediction ................................... 221
3.82 Create batch prediction ........................................................ 222
3.83 Download batch prediction CSV file ...................................... 223
3.84 Batch prediction output dataset ............................................ 223
3.85 Configure Default numeric value for batch prediction ............. 225
3.86 Configure Default numeric value for batch prediction ............. 226
3.87 Configure the fields mapping for batch prediction .................. 227
3.88 linear regression output settings for batch predictions .......... 228
3.89 Prediction explanation ........................................................... 229
3.90 Input field importances .......................................................... 229
3.91 linear regression prediction descriptive information ............... 231
3.92 Markdown editor for linear regression prediction descriptions 232
3.93 linear regression predictions privacy ..................................... 232
3.94 Stop linear regression batch prediction from 1-click action menu 233
3.95 linear regression delete prediction confirmation ..................... 233
3.96 Linear regression delete prediction from 1-click menu ............ 234
3.97 Linear regression delete prediction from pop up menu ............. 234
3.98 Linear regression delete prediction confirmation ..................... 234
3.99 Click download icon ............................................................ 235
3.100 Select language to download linear regression ...................... 235
3.101 Warning message when the linear regression does not have any numeric input field 237
3.102 Warning message when the linear regression has more than 800 predictors 237
3.103 Warning message when the coefficient table limits are reached 238
4.1 Logistic regressions under Supervised tab ................................................. 246
4.2 Logistic regression list view ................................................................. 247
4.3 Empty Dashboard logistic regressions view .............................................. 247
4.4 Logistic regression icon ...................................................................... 247
4.5 Logistic regression formula .................................................................... 248
4.6 Logistic regression formulas for two classes ......................................... 248
4.7 Missing numeric coefficients at the end of logistic regression table .............. 250
4.8 A dataset with images and image features ............................................. 251
4.9 A dataset with image feature fields shown ............................................ 251
4.10 Create 1-click logistic regression from dataset 1-click action menu .. ...... 252
4.11 Create 1-click logistic regression from dataset popup menu ................. 252
4.12 Configure logistic regression ............................................................... 253
4.13 Configure the objective field to create the logistic regression ................. 254
4.14 Change the default objective field ....................................................... 254
4.15 Automatic optimization ....... ____ __ ______ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ _
4.46 Contrast modal window in table view ........................................... 276
4.47 Field coding configuration ............................................................. 277
4.48 Enable field coding configuration ...................................................... 278
4.49 Field codings: other coding ............................................................. 278
4.50 Set the other coding values for each class ........................................ 279
4.51 Other coding saved ................................................................. 279
4.52 Close modal window ................................................................. 280
4.53 Field codings configured ............................................................... 280
4.54 Disable field coding configuration ................................................... 281
4.55 Other coding in coefficients table ................................................... 281
4.56 Other coding modal window ......................................................... 282
4.57 Sampling parameters for logistic regression ...................................... 283
4.58 Create logistic regression after configuration ................................... 283
4.59 Logistic regression API request preview ........................................ 284
4.60 Switch chart and table views .......................................................... 285
4.61 Logistic regression chart parts ......................................................... 285
4.62 1D chart .................................................................................... 286
4.63 2D chart .................................................................................... 286
4.64 Prediction legend ........................................................................ 287
4.65 Configure the values for other input fields ...................................... 288
4.66 Reset the values for the input fields ................................................ 288
4.67 Export chart as image with or without legends ................................. 289
4.68 Table view for logistic regression .................................................... 290
4.69 Multiple field variables for categorical one-hot encoded fields ........... 291
4.70 Missing numeric coefficients at the end of logistic regression table ....... 292
4.71 Search and filter logistic regression table ......................................... 293
4.72 Export table in CSV file ................................................................ 294
4.73 Likelihood ratio ............................................................................ 295
4.74 Significant likelihood ratio .............................................................. 295
4.75 Select significance level ................................................................ 296
4.76 Likelihood ratio p-value ................................................................ 296
4.77 Significant icon ........................................................................... 297
4.78 Non-significant icon ..................................................................... 297
4.79 Significance icons for coefficient estimates ...................................... 297
4.80 Summary of stats per coefficient .................................................... 298
4.81 Predictions list view ....................................................................... 298
4.82 Empty predictions list view ............................................................ 299
4.83 Menu options of the predictions list view ........................................ 299
4.84 Single predictions icon ................................................................. 299
4.85 Batch predictions icon .................................................................. 299
4.86 Predict using the 1-click action menu .............................................. 300
4.87 Predict using the pop up menu ....................................................... 300
4.88 Logistic regression prediction form .................................................. 301
4.89 Logistic regression predictions form ................................................ 302
4.90 Get the logistic regression prediction .............................................. 303
4.91 All classes probabilities distribution ............................................... 304
4.92 Save your logistic regression predictions ........................................ 305
4.93 Select a single image source in the image input field ......................... 306
4.94 List the components of a composite source ..................................... 307
4.95 Select a component of a composite source ....................................... 308
4.96 Logistic regression image prediction form, more fields ....................... 309
4.97 Logistic regression image single prediction ....................................... 310
4.98 Create batch prediction using 1-click action menu ............................ 310
4.99 Create batch prediction using pop up menu ..................................... 311
4.100 Select dataset for batch prediction ............................................... 311
4.101 Configuration options for logistic regression batch prediction .......... 312
4.102 Create a dataset from batch prediction .......................................... 313
4.103 Create batch prediction ............................................................... 314
### LIST OF FIGURES 581

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.10</td>
<td>Download batch prediction CSV file</td>
</tr>
<tr>
<td>4.10</td>
<td>Batch prediction output dataset</td>
</tr>
<tr>
<td>4.10</td>
<td>Batch prediction using an image dataset</td>
</tr>
<tr>
<td>4.10</td>
<td>Set the positive class for single predictions</td>
</tr>
<tr>
<td>4.10</td>
<td>Set the probability threshold for single predictions</td>
</tr>
<tr>
<td>4.10</td>
<td>Set the probability threshold for batch predictions</td>
</tr>
<tr>
<td>4.11</td>
<td>Configure Default numeric value for batch prediction</td>
</tr>
<tr>
<td>4.11</td>
<td>Configure the fields mapping for batch prediction</td>
</tr>
<tr>
<td>4.11</td>
<td>Logistic regression output settings for batch predictions</td>
</tr>
<tr>
<td>4.11</td>
<td>Logistic regression single prediction</td>
</tr>
<tr>
<td>4.11</td>
<td>Logistic regression all class probabilities</td>
</tr>
<tr>
<td>4.11</td>
<td>Logistic regression export all class probabilities histogram</td>
</tr>
<tr>
<td>4.11</td>
<td>Logistic regression probability threshold</td>
</tr>
<tr>
<td>4.11</td>
<td>Explain prediction</td>
</tr>
<tr>
<td>4.11</td>
<td>Input field importances</td>
</tr>
<tr>
<td>4.11</td>
<td>See the detailed explanation</td>
</tr>
<tr>
<td>4.12</td>
<td>Download batch prediction CSV file</td>
</tr>
<tr>
<td>4.12</td>
<td>An example of a logistic regression batch prediction CSV file</td>
</tr>
<tr>
<td>4.12</td>
<td>Batch prediction output dataset</td>
</tr>
<tr>
<td>4.12</td>
<td>Logistic regression batch prediction output dataset</td>
</tr>
<tr>
<td>4.12</td>
<td>Logistic regression prediction descriptive information</td>
</tr>
<tr>
<td>4.12</td>
<td>Markdown editor for logistic regression descriptions</td>
</tr>
<tr>
<td>4.12</td>
<td>Logistic regression predictions privacy</td>
</tr>
<tr>
<td>4.12</td>
<td>Stop logistic regression batch prediction from 1-click action menu</td>
</tr>
<tr>
<td>4.12</td>
<td>Logistic regression delete prediction confirmation</td>
</tr>
<tr>
<td>4.12</td>
<td>Logistic regression delete prediction from 1-click menu</td>
</tr>
<tr>
<td>4.13</td>
<td>Logistic regression delete prediction from pop up menu</td>
</tr>
<tr>
<td>4.13</td>
<td>Logistic regression delete prediction confirmation</td>
</tr>
<tr>
<td>4.13</td>
<td>Click download icon</td>
</tr>
<tr>
<td>4.13</td>
<td>Select language to download logistic regression</td>
</tr>
<tr>
<td>4.13</td>
<td>Warning message when the logistic regression does not have any numeric field</td>
</tr>
<tr>
<td>4.13</td>
<td>Warning message when the logistic regression has more than 100 fields</td>
</tr>
<tr>
<td>4.13</td>
<td>Warning message when the objective field has more than 200 classes</td>
</tr>
<tr>
<td>4.13</td>
<td>Warning message when the table limits are reached</td>
</tr>
<tr>
<td>4.13</td>
<td>Edit logistic regression descriptive information</td>
</tr>
<tr>
<td>4.13</td>
<td>Markdown editor for logistic regression descriptions</td>
</tr>
<tr>
<td>4.14</td>
<td>Counters for logistic regressions</td>
</tr>
<tr>
<td>4.14</td>
<td>Logistic regression privacy</td>
</tr>
<tr>
<td>4.14</td>
<td>Logistic regression probability threshold</td>
</tr>
<tr>
<td>4.14</td>
<td>Change project from 1-click action menu</td>
</tr>
<tr>
<td>4.14</td>
<td>Change project from pop up menu</td>
</tr>
<tr>
<td>4.14</td>
<td>Stop logistic regression creation from 1-click action menu</td>
</tr>
<tr>
<td>4.14</td>
<td>Stop logistic regression creation from pop up menu</td>
</tr>
<tr>
<td>4.14</td>
<td>Confirmation message to delete a logistic regression</td>
</tr>
<tr>
<td>4.14</td>
<td>Delete logistic regression from 1-click action menu</td>
</tr>
<tr>
<td>4.14</td>
<td>Delete logistic regression from pop up menu</td>
</tr>
<tr>
<td>4.14</td>
<td>Confirmation message to delete a logistic regression</td>
</tr>
<tr>
<td>4.15</td>
<td>Logistic Regression Workflow</td>
</tr>
</tbody>
</table>

5.1 Deepnet list view | 347
5.2 Empty Dashboard deepnet view | 347
5.3 Deepnet icon | 347
5.4 Create 1-click deepnet from dataset 1-click action menu | 350
5.5 Create 1-click deepnet from dataset popup menu | 351
5.6 Configure deepnet | 351
5.7 Configure the objective field to create the deepnet | 352
5.8 Change the default objective field | 352
5.9 Select an automatic optimization option | 353
5.10 Disable automatic optimization options to manually configure the rest of the network parameters .................................................. 354
5.11 Select a default numeric value to replace missing numeric values .................................................. 354
5.12 Include missing numeric values in your deepnet .................................................. 355
5.13 Set the training duration for a deepnet .................................................. 355
5.14 Set the maximum number of iterations for a deepnet .................................................. 356
5.15 Configure the hidden layers of the network .................................................. 357
5.16 Enable or disable the residuals learning .................................................. 358
5.17 Enable or disable the batch normalization .................................................. 359
5.18 Enable or disable the tree embedding .................................................. 360
5.19 Momentum algorithm .................................................. 361
5.20 Adagrad algorithm .................................................. 362
5.21 RMSProp algorithm .................................................. 363
5.22 Adam algorithm .................................................. 364
5.23 FTRL algorithm .................................................. 365
5.24 Configure the learning rate .................................................. 366
5.25 Configure the dropout rate .................................................. 367
5.26 Set a seed for the deepnet .................................................. 368
5.27 Weight options for deepnets .................................................. 369
5.28 Sampling parameters for deepnet .................................................. 371
5.29 Create deepnet after configuration .................................................. 372
5.30 Deepnet API request preview .................................................. 372
5.31 Deepnet chart parts .................................................. 373
5.32 Deepnet chart .................................................. 374
5.33 Prediction legend .................................................. 375
5.34 Configure the values for other input fields .................................................. 376
5.35 Reset the values for the input fields .................................................. 377
5.36 Export chart as image with or without legends .................................................. 378
5.37 Image Deepnet Page Classification layout .................................................. 379
5.38 Image Deepnet Page Classification Performance Panel .................................................. 380
5.39 Image Deepnet Page Classification Image Results .................................................. 381
5.40 Image Deepnet Page Classification Class List .................................................. 382
5.41 Image Deepnet Page Classification filtering by probability .................................................. 383
5.42 Image Deepnet Page Classification showing one class .................................................. 384
5.43 Image Deepnet Page Regression layout .................................................. 385
5.44 Image Deepnet Page Regression Performance Panel .................................................. 386
5.45 Image Deepnet Page Regression Display Error Slider .................................................. 387
5.46 Image Deepnet Page Regression Split Error Slider .................................................. 388
5.47 Image Deepnet Page Regression Image Results .................................................. 389
5.48 Field importance for deepnets .................................................. 390
5.49 View networks’ configuration .................................................. 391
5.50 Predictions list view .................................................. 392
5.51 Empty predictions list view .................................................. 392
5.52 Menu options of the predictions list view .................................................. 392
5.53 Single predictions icon .................................................. 393
5.54 Batch predictions icon .................................................. 393
5.55 Predict using the 1-click action menu .................................................. 393
5.56 Predict using the pop up menu .................................................. 394
5.57 Deepnet prediction form .................................................. 394
5.58 Deepnet predictions form .................................................. 395
5.59 Get the deepnet prediction .................................................. 396
5.60 Display all class probabilities .................................................. 396
5.61 Predict with images using the 1-click action menu .................................................. 397
5.62 Predict using the pop-up menu .................................................. 397
5.63 Select a single image source in the image input field .................................................. 398
5.64 List the components of a composite source .................................................. 399
5.65 Select a component of a composite source .................................................. 400
5.66 Deepnet image single prediction .................................................. 400
5.67 Deepnet image prediction with more than seven classes ........................................... 401
5.68 Deepnet image prediction form, more fields ................................................................. 402
5.69 Create batch prediction using 1-click action menu ....................................................... 402
5.70 Create batch prediction using pop up menu ................................................................. 403
5.71 Select dataset for batch prediction .................................................................................. 403
5.72 Configuration options for deepnet batch prediction ...................................................... 404
5.73 Create a dataset from batch prediction ........................................................................... 405
5.74 Create batch prediction .................................................................................................. 405
5.75 Download batch prediction CSV file ............................................................................. 406
5.76 Batch prediction output dataset ...................................................................................... 406
5.77 Batch prediction using an image dataset ........................................................................ 407
5.78 Select the positive class .................................................................................................. 408
5.79 Set a probability threshold ............................................................................................. 408
5.80 Save the prediction ......................................................................................................... 409
5.81 Configure a probability threshold for batch predictions ............................................... 409
5.82 Configure Default numeric value for batch prediction .................................................. 410
5.83 Configure the fields mapping for batch prediction ......................................................... 411
5.84 Deepnet output settings for batch predictions ................................................................ 412
5.85 Deepnet single prediction ............................................................................................... 412
5.86 Deepnet all class probabilities ......................................................................................... 413
5.87 Explain prediction ........................................................................................................... 414
5.88 Input field importances .................................................................................................... 414
5.89 See the detailed explanation ............................................................................................ 415
5.90 Download batch prediction CSV file .............................................................................. 415
5.91 An example of a deepnet batch prediction CSV file ...................................................... 416
5.92 Batch prediction output dataset ...................................................................................... 416
5.93 Deepnet batch prediction output dataset ...................................................................... 417
5.94 Deepnet prediction descriptive information .................................................................... 418
5.95 Markdown editor for deepnet descriptions ..................................................................... 419
5.96 Deepnet predictions privacy ............................................................................................ 420
5.97 Stop deepnet batch prediction from 1-click action menu ............................................... 420
5.98 Deepnet delete prediction confirmation ........................................................................... 421
5.99 Deepnet delete prediction from 1-click menu ................................................................ 421
5.100 Deepnet delete prediction from pop up menu .............................................................. 421
5.101 Deepnet delete prediction confirmation ........................................................................ 422
5.102 Click download icon ....................................................................................................... 422
5.103 Select language to download deepnet ............................................................................ 423
5.104 Edit deepnet descriptive information ........................................................................... 425
5.105 Markdown editor for deepnet descriptions ................................................................... 426
5.106 Counters for deepnet ..................................................................................................... 427
5.107 Deepnet privacy ............................................................................................................. 428
5.108 Change project from 1-click action menu .................................................................... 428
5.109 Change project from pop up menu ................................................................................ 429
5.110 Stop deepnet creation from 1-click action menu .......................................................... 429
5.111 Stop deepnet creation from pop up menu ..................................................................... 430
5.112 Confirmation message to delete a deepnet ................................................................... 430
5.113 Delete deepnet from 1-click action menu ..................................................................... 430
5.114 Delete deepnet from pop up menu ............................................................................... 431
5.115 Confirmation message to delete a deepnet ................................................................... 431
5.116 Deepnet Workflow ......................................................................................................... 433
6.1 Fusion list view ................................................................................................................ 435
6.2 Empty Dashboard fusion view .......................................................................................... 435
6.3 Fusion icon ....................................................................................................................... 435
6.4 Create fusion from 1-click menu ...................................................................................... 437
6.5 Create fusion from pop up menu ...................................................................................... 437
6.6 Create fusion from model list views .................................................................................. 438
6.7 Select models with the same objective field .................................................................... 438
6.8 Select models from the OptiML table ........................................ 439
6.9 Create fusion using another fusion ........................................... 439
6.10 Fusion creation view ............................................................. 440
6.11 Search for models ............................................................... 440
6.12 Selected models ................................................................. 441
6.13 The model search will be filtered by the objective field name .......... 442
6.14 Remove the objective field filter to select models with different objective field names 442
6.15 Select which objective field you want to use to select more models 443
6.16 Assign weights to the models ............................................... 444
6.17 Create fusion after configuration ........................................... 444
6.18 Fusion API request preview .................................................. 445
6.19 Switch from PDP to model list views ....................................... 446
6.20 Fusions chart view parts ...................................................... 447
6.21 Select fields for your fusion PDP axis ..................................... 448
6.22 Prediction legend ............................................................... 449
6.23 Configure the values for other input fields ................................ 450
6.24 Reset the values for the input fields ....................................... 451
6.25 Export chart as image with or without legends ......................... 451
6.26 Fusion model list view .......................................................... 452
6.27 Predictions list view ............................................................ 453
6.28 Empty predictions list view ................................................... 453
6.29 Menu options of the predictions list view ................................. 453
6.30 Single predictions icon .......................................................... 454
6.31 Batch predictions icon .......................................................... 454
6.32 Predict using the 1-click action menu ..................................... 454
6.33 Predict using the pop up menu ................................................ 455
6.34 Fusion prediction form .......................................................... 455
6.35 Fusion prediction ................................................................. 456
6.36 Create batch prediction using 1-click action menu ...................... 457
6.37 Create batch prediction using pop up menu ............................... 457
6.38 Select dataset for batch prediction ......................................... 458
6.39 Configuration options for fusion batch prediction ....................... 458
6.40 Create a dataset from batch prediction .............................. 459
6.41 Download batch prediction CSV file and output dataset ............. 459
6.42 Missing strategies for fusion single predictions .......................... 460
6.43 Missing strategies for fusion batch predictions .......................... 461
6.44 Select the positive class ....................................................... 462
6.45 Set a probability threshold ....................................................... 463
6.46 Get the prediction ................................................................. 464
6.47 Configure a probability threshold for batch predictions .............. 465
6.48 Configure Default numeric value for batch prediction .................. 466
6.49 Exclude fields from the fusion prediction ................................ 466
6.50 Configure the fields mapping for batch prediction ...................... 467
6.51 Fusion output settings for batch predictions ............................... 468
6.52 Fusion single prediction .......................................................... 469
6.53 Fusion all class probabilities .................................................. 470
6.54 Explain prediction ............................................................... 471
6.55 Input field importances ......................................................... 471
6.56 See the detailed explanation .................................................... 472
6.57 Download batch prediction CSV file ....................................... 473
6.58 An example of a fusion batch prediction CSV file ................. 473
6.59 Batch prediction output dataset .............................................. 474
6.60 Fusion batch prediction output dataset ..................................... 474
6.61 Fusion prediction descriptive information ................................ 476
6.62 Markdown editor for fusion descriptions .................................. 477
6.63 Fusion predictions privacy ...................................................... 477
6.64 Stop fusion batch prediction from 1-click action menu ................ 478
6.65 Fusion delete prediction confirmation .................................... 478

Copyright © 2023, BigML, Inc.
7.77 Change project from 1-click action menu
7.78 Change project from pop up menu
7.79 Stop fusion creation from 1-click action menu
7.80 Stop fusion creation from pop up menu
6.81 Confirmation message to delete a fusion
6.82 Delete fusions from 1-click action menu
6.83 Delete fusion from pop up menu
6.84 Confirmation message to delete a fusion
6.85 Fusion Workflow

6.80 Stop fusion creation from pop up menu
6.79 Stop fusion creation from 1-click action menu
6.78 Change project from pop up menu
6.77 Change project from 1-click action menu
6.76 Fusions privacy
6.75 Counters for fusions
6.74 Markdown editor for fusion descriptions
6.73 Edit fusion descriptive information
6.72 The PDP cannot be displayed because the predictions take too much time to be computed
6.71 The PDP cannot be displayed
6.70 Select language to download your fusions
6.69 Click download icon
6.68 Delete fusion prediction confirmation
6.67 Delete fusion prediction from pop up menu
6.66 Delete fusion prediction from 1-click menu
6.65 The PDP cannot be displayed because the predictions take too much time to be computed
6.64 The PDP cannot be displayed
6.63 Cell colors indicate the class selected as positive
6.62 Change project from pop up menu
6.61 Change project from 1-click action menu
6.60 Fusions privacy
6.59 Counters for fusions
6.58 Markdown editor for fusion descriptions
6.57 Edit fusion descriptive information
6.56 Stop fusion creation from pop up menu
6.55 Stop fusion creation from 1-click action menu
6.54 Change project from pop up menu
6.53 Change project from 1-click action menu
6.52 Fusions privacy
6.51 Counters for fusions
6.50 Markdown editor for fusion descriptions
6.49 Edit fusion descriptive information
6.48 Stop fusion creation from pop up menu
6.47 Stop fusion creation from 1-click action menu
6.46 Change project from pop up menu
6.45 Change project from 1-click action menu
6.44 Fusions privacy
6.43 Counters for fusions
6.42 Markdown editor for fusion descriptions
6.41 Edit fusion descriptive information
6.40 Stop fusion creation from pop up menu
6.39 Stop fusion creation from 1-click action menu
6.38 Change project from pop up menu
6.37 Change project from 1-click action menu
6.36 Evaluate ensemble from evaluation list view
6.35 Evaluation with pre-filled model and dataset information
6.34 Evaluate model from pop up menu
6.33 Evaluate model from 1-click action menu
6.32 Select model and dataset
6.31 Evaluate model from evaluation list view
7.96 Compare multiple evaluations using 1-click action menu ............................................. 557
7.97 Compare multiple evaluations using 1-click action menu from the list view ..................... 557
7.98 Select an evaluation curve ......................................................................................... 558
7.99 Select evaluations to compare .................................................................................... 559
7.100 Manage evaluations from the legend ......................................................................... 560
7.101 See the PR AUC and ROC AUC by hovering over the evaluation name in the table .... 561
7.102 See the K-S statistic by hovering over the evaluation name in the table ...................... 562
7.103 Table containing all the evaluations and the model parameters ................................. 563
7.104 Export multiple comparison chart in PNG format ....................................................... 564
7.105 Panel to edit an evaluation name, description, category, and tags ............................ 565
7.106 Markdown editor for evaluations descriptions .......................................................... 566
7.107 Share evaluations .................................................................................................... 568
7.108 Move evaluation from the 1-click action menu .......................................................... 568
7.109 Move evaluation from the pop up menu .................................................................... 569
7.110 Stop evaluation creation .......................................................................................... 569
7.111 Stop evaluation from pop up menu ......................................................................... 570
7.112 Delete evaluation from the 1-click action menu ........................................................ 570
7.113 Delete evaluation from pop up menu ....................................................................... 571
7.114 Delete evaluation confirmation ............................................................................... 571
7.115 Single evaluations workflow ..................................................................................... 572
7.116 Cross-validation workflow ....................................................................................... 572
List of Tables

1.1 Weight Field example for transactional dataset ........................................ 17
1.2 Categories used to classify predictions by BigML .................................... 66
1.3 Categories used to classify models by BigML ............................................ 73

2.1 Weight Field example for transactional dataset ........................................ 102
2.2 Example of classification ensemble using probability ................................. 136
2.3 Example of regression ensemble using probability ..................................... 137
2.4 Example of classification ensemble using votes ........................................ 137
2.5 Categories used to classify predictions by BigML ..................................... 155
2.6 Categories used to classify ensembles by BigML ....................................... 161

3.1 Number of predictors per input field ....................................................... 172
3.2 Dummy coding example for 3 classes ...................................................... 180
3.3 Contrast coding example for 3 classes ..................................................... 185
3.4 Other coding ............................................................................................. 192
3.5 Categories used to classify linear regression by BigML ............................... 240

4.1 One-hot coding ......................................................................................... 265
4.2 Dummy coding example for 3 classes ...................................................... 265
4.3 Contrast coding example for 3 classes ..................................................... 270
4.4 Other coding ............................................................................................. 276
4.5 Categories used to classify logistic regression by BigML ......................... 339

5.1 Categories used to classify deepnet by BigML .......................................... 427

6.1 Categories used to classify fusions on BigML ............................................ 486

7.1 Example of diabetes prediction probabilities for three different patients .... 504
7.2 Categories used to classify evaluations by BigML ....................................... 567
Glossary

**Bagging** an ensemble based algorithm that uses a random subset of instances to generate each single tree. 96

**BigML Gallery** a section of BigML to share, buy or sell datasets, models, and scripts. Go to Gallery. 74, 77, 84, 528

**Classification** a modeling task whose objective field (i.e., the field being predicted) is categorical and predicts classes. ii, 1, 3, 23, 25, 83, 85, 98, 165, 168, 246, 346, 347, 357, 391, 434, 436, 452, 494, 498, 556

**Clustering** an unsupervised Machine Learning task in which dataset instances are grouped into geometrically related subsets. 10

**Confidence** an indicator of the prediction's certainty for classification models and ensembles. It takes into account the class distribution and the number of instances at a certain node. It is a value between 0% and 100%. 35, 83, 119, 166

**Dashboard** The BigML web-based interface that helps you privately navigate, visualize, and interact with your modeling resources. ii, 1, 35, 77, 86, 120, 168, 214, 219, 246, 298, 305, 346, 391, 397, 434, 452, 456, 494

**Dataset** the structured version of a BigML source. It is used as input to build your predictive models. For each field in your dataset a number of basic statistics (min, max, mean, etc.) are parsed and produced as output. 1, 48, 132, 221, 312, 404, 458, 494

**Decision Trees** a class of Machine Learning algorithms used to solve regression and classification problems. Decision trees are composed of nodes and branches that create a model of decisions with a tree graph. Nodes represent the predictors or labels that have an influence in the predictive path, and the branches represent the rules followed by the algorithm to make a given prediction. 1

**Deepnets** an optimized implementation of deep neural networks, a class of supervised learning algorithms, that can be used to solve regression and classification problems. The input features are fed to one or several groups “nodes”, each group of nodes form a “layer”. Each node is essentially a function on the input that transforms the input features into another value or collection of values. This process continues layer by layer, until we reach the final output (prediction), an array of per-class probabilities for classification problems or a single, real value for regression problems. ii, 168, 434, 435, 440, 494

**Early split** when the dataset is bigger than 34 GB, BigML automatically takes a sample of your data and performs an early split so the model creation becomes significantly faster. It detects when an early split is safe by calculating the summary statistics collected at each node. Early splitting requires that the training data is shuffled beforehand to avoid generating inaccurate models caused by ordered fields in the input rows (as it will process the first x instances, then the next x ones and so on). 3, 19, 106, 199
**Ensembles** a class of Machine Learning algorithms in which multiple independent classifiers or regressors are trained, and the combination of these classifiers is used to predict an objective field. An ensemble of models built on samples of the data can become a powerful predictor by averaging away the errors of each individual model. ii, 119, 165, 168, 246, 346, 434, 435, 440, 494

**Expected error** an indicator of the prediction's certainty for regression models and ensembles. It is the average of the predictions errors at a certain node. 7, 35, 83, 119, 166

**Field** an attribute of each instance in your data. Also called “feature”, “covariate”, or “predictor”. Each field is associated with a type (numeric, categorical, text, items, or date-time). 1, 25, 85, 110, 246, 346, 434

**Field importance** a measure of each field importance for predicting the objective field relative to the other fields. It is computed averaging the error each field helps to reduce at every tree split. 39, 122

**Fusions** supervised model that solves classification and regression problems by averaging the predictions of multiple models, ensembles, logistic regressions, and/or deepnets. Fusions are based on the same “wisdom of the crowds” principle than ensembles under which the combination of multiple models is often more performant than any of its individual members.. ii, 434, 440, 494

**Histogram** a bar chart-style visualization of a collection of values, in which the range of the values is broken up into a collection of ranges, and the height of a given bar increases as more points fall into the range associated with that bar. 4

**Instances** the data points that represent the entity you want to model, also known as observations or examples. They are usually the rows in your data with a value (potentially missing) for each field that describes the entity. 1

**Leaf** a terminal node in a tree, i.e., a node that has no children. 3, 21, 592

**Linear regression** a popular technique from the fields of statistics that has been borrowed by Machine Learning to solve regression problems. Linear regression assumes the output variable, or the objective field, is a function of linear combination of the inputs. ii

**Local predictions** the predictions made in your local environment, faster, at no cost, by downloading your model. 41, 69, 126, 158, 217, 230, 302, 324

**Logistic regression** another technique from the fields of statistics that has been borrowed by Machine Learning to solve classification problems. For each class of the objective field, logistic regression fits a logistic function to the training data. Logistic regression is a linear model, in the sense that it assumes the probability of a given class is a function of a weighted combination of the inputs. ii, 346, 434, 435, 440, 494

**Missing value** the data points that represent the entity you want to model may present missing value, i.e., not provide a value for all fields that compose the entity. 15, 99

**Model** a single decision tree-like model when we refer to it in particular, and a predictive model when we refer to it in general. ii, 35, 83, 168, 246, 346, 434, 435, 440, 494

**Node threshold** the maximum number of nodes that a BigML model is allowed to grow. 15, 99

**Non-preferred fields** fields that, for a number of possible reasons, are by default not included in the modeling process. One example of this is fields that contain the same value for every instance; in general, constant fields add no information to the modeling process. 11, 92

**Objective Field** the field that a regression or classification model will predict (also known as target). 1, 12, 15, 83, 85, 86, 93, 101, 109, 165, 168, 173, 174, 201, 245, 246, 247, 252, 253, 284, 326, 344, 346, 351, 373, 416, 432, 434, 436, 441, 445, 473, 492, 494, 498, 542, 543
**OptiML** an automated optimization process for model selection and parametrization (or hyperparametrization) to solve classification and regression problems. 13, 94, 176, 255, 438, 440

**Organization** a collaborative workspace where all the users in the organization can access, work on, and visualize the same projects and resources in the BigML Dashboard. Furthermore, organizations enable you to define different roles and permissions for each user involved on your Machine Learning projects. 488

**Orthogonal** the default “one-hot” coding is orthogonal since a single instance can’t be two categories at the same time, so after recoding we also need to ensure there are not co-dependent coefficients. Orthogonality is met when the dot product for the codings equals 0, e.g., the following codings are orthogonal $[1,1,-1,-1], [-1,1,0,0]$ since $(1) \times (-1) + (1) \times (1) + (-1) \times (0) + (-1) \times (0) = 0$. 187, 272

**Overfitting** the process of tailoring the model to fit the training data at the expense of generalization. 5, 14, 15, 85, 97, 99, 264, 357, 364, 365, 366, 511, 516, 542, 543

**Predicate** a predicate is a statement that can be either true or false depending on the values of its component variables. BigML predicates may use the boolean operators $=$, $<$, $>$, $\leq$, $\geq$. Example of predicates are $\text{balance} < 1,000$ and $\text{field x} = \text{"category"}$. 3

**Predicting** the result of obtaining the objective field value for your new data using an existing model. The model returns the predicted value along with a performance measure (confidence for classification or expected error for regression). 35, 119, 214, 245, 298, 344, 391, 432, 452, 492, 494

**Prediction Path** the series of rules that lead to a certain node in a decision tree. 22

**Predictors** the fields your model uses as inputs to generate the set of rules to make predictions. 109

**Project** an abstract resource that helps you group related BigML resources together. 2, 79, 86, 162, 214, 241, 247, 299, 340, 347, 392, 428, 435, 453, 487, 496

**Random Decision Forests** an ensemble based algorithm which uses a random subset of features to generate anomaly scores. 5, 96

**Regression** a modeling task whose objective field (i.e., the field being predicted) is numeric. ii, 1, 3, 23, 25, 83, 85, 98, 165, 168, 346, 347, 357, 391, 434, 436, 452, 494

**Resource** any of the Machine Learning objects provided by BigML that can be used as a building block in the workflows needed to solve Machine Learning problems. 494

**Root** the node from which a tree originates. 21, 592

**Sampling** the process of partitioning your dataset to consider just a subset of your instances. 17, 104

**Supervised learning** a type of Machine Learning problem in which each instance of the data has a label. The label for each instance is provided in the training data, and a supervised Machine Learning algorithm learns a function or model that will predict the label given all other features in the data. The function can then be applied to data unseen during training to predict the label for unlabeled instances. ii, 1, 85, 168, 169, 246, 247, 346, 347, 391, 434, 436, 452, 494

**Support** the proportion of instances in the dataset which contain an itemset. The support of an association is the portion of instances in the dataset which contain the rule’s antecedent and rule’s consequent together over the total number of instances ($N$) in the dataset. 23

**Task** the process of creating a BigML resource, such as creating a dataset, or training a model. A given task can also create subtasks, as, in the case of a WhizzML script that contains calls to create other resources. 79

**Time series** a sequentially indexed representation of your historical data that can be used to forecasting future values of numerical properties. BigML implements exponential smoothing where the smoothing parameters assign exponentially increasing weights to most recent instances. Exponential smoothing methods allow the modelization of data with trend and seasonal patterns. ii
**Tree**  a data structure that can be described as a collection of nodes, starting at a root node, where each node may recursively have a number of child nodes. Nodes that have no children are called leaves. 3, 21, 590, 591

**Unsupervised learning**  a type of Machine Learning problem in which the objective is not to learn a predictor, and thus does not require each instance to be labeled. Typically, unsupervised learning algorithms infer some summarizing structure over the dataset, such as a clustering or a set of association rules. ii

**WhizzML**  BigML domain-specific language for automating complex Machine Learning workflows and implementing high-level algorithms. 528
References


