Sources with the BigML Dashboard

The BigML Team

Version 2.2

MACHINE LEARNING MADE BEAUTIFULLY SIMPLE

About this Document

This document provides a comprehensive description of how BigML sources work. A BigML source is the basic building block to bring your data to BigML and configure how BigML will parse it. BigML sources are used to create datasets that can later be transformed into predictive models or used as input to batch processes.

To learn how to use the BigML Dashboard to create datasets read:

- Datasets with the BigML Dashboard. The BigML Team. June 2016. [5]

To learn how to use the BigML Dashboard to build supervised predictive models read:

- Classification and Regression with the BigML Dashboard. The BigML Team. June 2016. [3]
- Time Series with the BigML Dashboard. The BigML Team. July 2017. [6]

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

- Cluster Analysis with the BigML Dashboard. The BigML Team. June 2016. [4]
- Anomaly Detection with the BigML Dashboard. The BigML Team. June 2016. [1]
- Association Discovery with the BigML Dashboard. The BigML Team. June 2016. [2]
- Topic Modeling with the BigML Dashboard. The BigML Team. November 2016. [7]

Contents

CHAPTER 1

Introduction

BigML is consumable, programmable, and scalable Machine Learning software that helps solving **Classification**, **Regression**, **Cluster Analysis**, **Anomaly Detection**, and **Association Discovery** problems, using a number of patent-pending technologies.

BigML helps you address these problems *end-to-end*. That is, you can seamlessly transform data into actionable predictive models, and later use these models (either as remote services or locally embedded into your applications) to make predictions.

To be processed by BigML, your data need to be first in *Machine Learning-Ready Format* (see Section 1.1) and stored in a data source (a source for short). Basically, a source is a collection of instances of the entity that you want to model stored in tabular format in a computer file. Typically, in a source, each row represents one of the instances and each column represents a field of the entity (see Figure 1.6). Section 1.1 describes the structure BigML expects a source to have. There are different types and data formats of BigML sources. There are also different file formats that BigML can process. They are all covered in Chapter 2.

Every time a new source is brought to BigML, a corresponding BigML source is created. Section 1.2 gives you a first example of how to create a BigML source. BigML uses the icon in Figure 1.1 to represent a BigML source.

Figure 1.1: Source icon

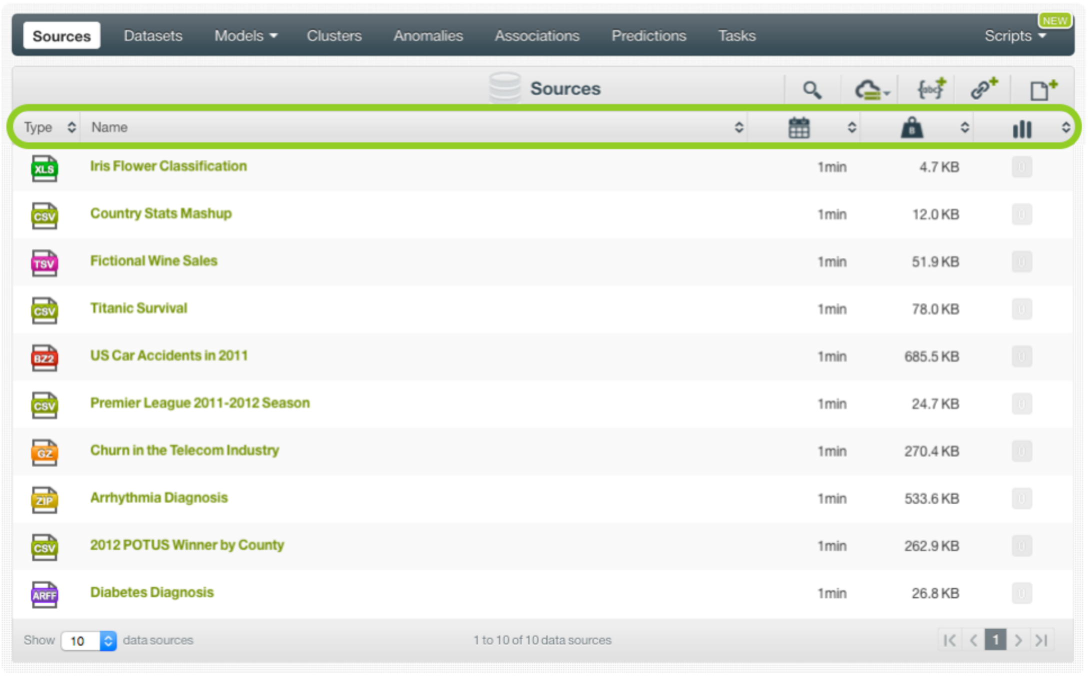
The main purpose of BigML sources is to make sure that BigML parses and interprets each instance in your source correctly. This can save you some time before proceeding with any modeling on your data that involves heavier computation. BigML analyzes the initial part of each source to automatically infer the type of each field. BigML accepts fields of type: *numeric*, *categorical*, *date-time*, *text*, and *items*. These types are explained in detail in Chapter 5. The BigML Dashboard lets you update each field type individually to fix those cases in which BigML does not recognize the type of a field correctly (see Section 6.11). The BigML Dashboard also allows you to configure many other settings to ensure that your sources are correctly parsed. Chapter 6 describes all the available settings.

BigML is able to ingest sources from three different origins:

- **Local Sources** that are accessible in your local computer. (See Chapter 7.)
- **Remote Sources** that can be accessed using different transfer protocols or configuring different cloud storage providers. (See Chapter 8.)

- **Inline Sources** that can be created using a simple editor provided by the BigML Dashboard. (See Chapter 9.)

The first tab of the BigML Dashboard's main menu allows you to list all your available sources. When you first create an account at BigML, you will find a list of promotional BigML sources. (See Figure 1.2.) In this **source list view** (Figure 1.2), you can see, for each source, the **Type**, **Name**, **Age** (time since the BigML source was created), **Size**, and **Number of Datasets** that have been created using that BigML source.



Type	Name			
XLS	Iris Flower Classification	1min	4.7 KB	0
CSV	Country Stats Mashup	1min	12.0 KB	0
TSV	Fictional Wine Sales	1min	51.9 KB	0
CSV	Titanic Survival	1min	78.0 KB	0
BZ2	US Car Accidents in 2011	1min	685.5 KB	0
CSV	Premier League 2011-2012 Season	1min	24.7 KB	0
GZ	Churn in the Telecom Industry	1min	270.4 KB	0
ZIP	Arrhythmia Diagnosis	1min	533.6 KB	0
CSV	2012 POTUS Winner by County	1min	262.9 KB	0
ARFF	Diabetes Diagnosis	1min	26.8 KB	0

Show 10 data sources 1 to 10 of 10 data sources 1

Figure 1.2: Source list view

On the top right corner of the **source list view**, you can see the menu options shown on Figure 1.3.

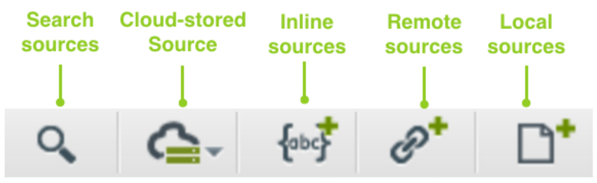


Figure 1.3: Menu options of the source list view

These menu options perform the following operations (from right to left):

1. **Create a source from a local source** opens a file dialog that helps you browse files in your local drives. (See Chapter 7.)
2. **Create a source from a URL** opens a modal window that helps you input the URL of that BigML will use to automatically download a remote source. (See Chapter 8.)
3. **Create a inline source** opens an editor where you can directly input or paste data into it. (See Chapter 9.)
4. **Cloud Storage Drop Down** helps you browse through previously configured cloud storage providers. (See Subsection 8.7.1.)
5. **Search** searches your sources by name.

By default, every time you start a new project, your list of sources will be empty. (See Figure 1.4.)

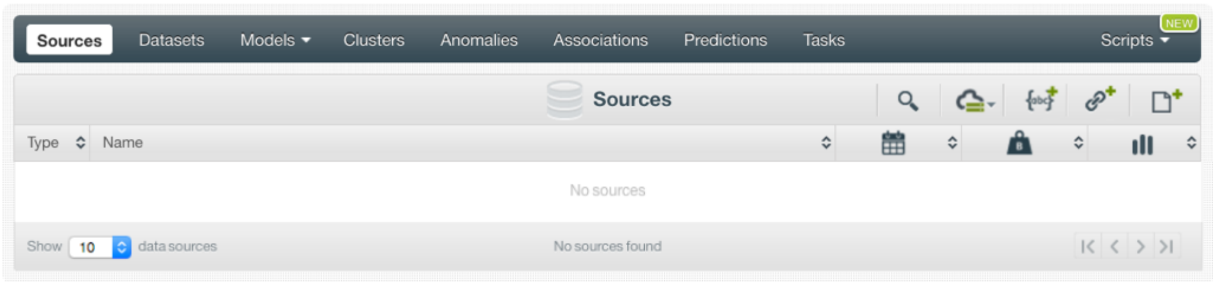


Figure 1.4: Empty Dashboard sources view

BigML does not impose any limit on the number of sources you can have under an individual BigML account or project. In addition, there are no limits on either the number of instances or the number of fields per source, though there are some limits on the total size a source can have, as explained in Chapter 10.

Each BigML source has a **Name**, a **Description**, a **Category**, and **Tags**. These allow you to provide documentation, and can also be helpful when searching through your sources. More details are in Chapter 11.

A BigML source can be associated with a specific project. You can move a source between projects. To perform this operation, see Chapter 13. A source can also be deleted permanently from your account. (See Chapter 14.)

A BigML source is the first resource that you need to create to apply Machine Learning to your own data using BigML. The only direct operation you can perform on a BigML source is creating a BigML dataset. BigML makes a clear distinction between sources and datasets: BigML sources allow you to ensure that BigML correctly transfers, parses, and interprets the content in your data, while a BigML dataset is a structured version of your data with basic statistics computed for each field. The main purpose of BigML sources is, therefore, to give you configuration options to ensure that your data is being parsed correctly. For a detailed explanation of BigML datasets, read the Datasets with the BigML Dashboard document [5].

1.1 Machine Learning-Ready Format

A data source is in Machine Learning-ready (ML-ready) format when a collection of instances of the entity you want to model has been transformed into tabular format (see Figure 1.5), in order to solve a specific Machine Learning task (i.e., **classification**, **regression**, **cluster analysis**, **anomaly detection**, or **association discovery**).

To get your data in ML-ready format requires:

1. Selecting a modeling task appropriate to your needs.
2. Denormalizing, aggregating, pivoting, and other data wrangling tasks to generate a suitable "feature space" for your selected modeling task.
3. Using domain knowledge and Machine Learning expertise to generate additional features that help better represent the instances.
4. Choosing the right file format to store each type of feature into a field and each instance into a record using a tabular structure. Each row is used to represent one of the instances, and each column is used to represent a field that describes all the instances. Each field can be: *numeric*, *categorical*, *text*, *items*, or *date-time*. (See Chapter 5.)

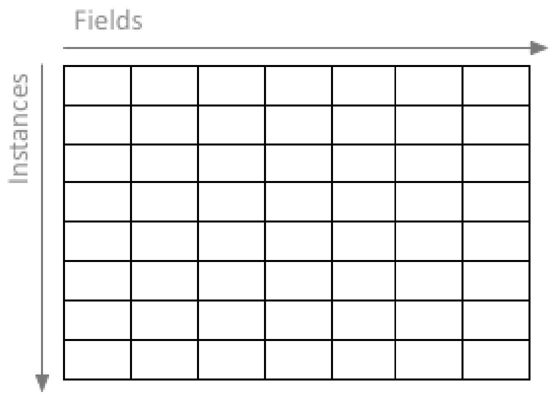


Figure 1.5: Instances and fields in tabular format

By structuring your data into ML-ready format before uploading it to BigML, you will better prepared to maximize the BigML capabilities and discover more insightful patterns and build better predictive models.

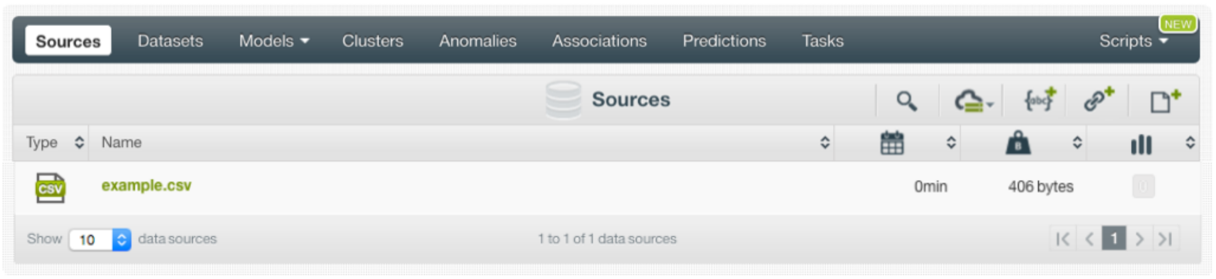
1.2 Creating a First Source

Figure 1.6 shows an example of a source in ML-ready format. Each row represents a user of a cell phone service and each column is an attribute of each user. The data is structured to predict whether a user will be canceling her account (Churn?) given her current plan (Plan), the number of minutes used last month (Talk), the number of text messages sent last month (Text), the number of applications purchased last month (Purchases), the number of megabytes of data consumed last month (Data), and the current age of the user (Age). The source is a CSV (Comma Separated Values) file and, therefore, in the right format to be processed by BigML.

```
Plan, Talk, Text, Purchases, Data, Age, Churn?
family, 148, 72, 0, 33.6, 50, TRUE
business, 85, 66, 0, 26.6, 31, FALSE
business, 83, 64, 0, 23.3, 32,TRUE
individual, 9,  66, 94, 28.1, 21, FALSE
family, 15, 0, 0, 35.3, 29, FALSE
individual, 66, 72, 175, 25.8, 51,TRUE
business, 0, 0, 0, 30, 32, TRUE
family, 18, 84, 230, 45.8, 31,TRUE
individual, 71, 110, 240, 45.4, 54, TRUE
family, 59, 64, 0, 27.4, 40, FALSE
```

Figure 1.6: An example of a CSV file

To bring the source in Figure 1.6 to BigML, you can just drag and drop the file containing it on top of the BigML Dashboard. You can also paste its content into the BigML inline editor (see Chapter 9). A new source in the **source list view** will be shown. (See Figure 1.7.)



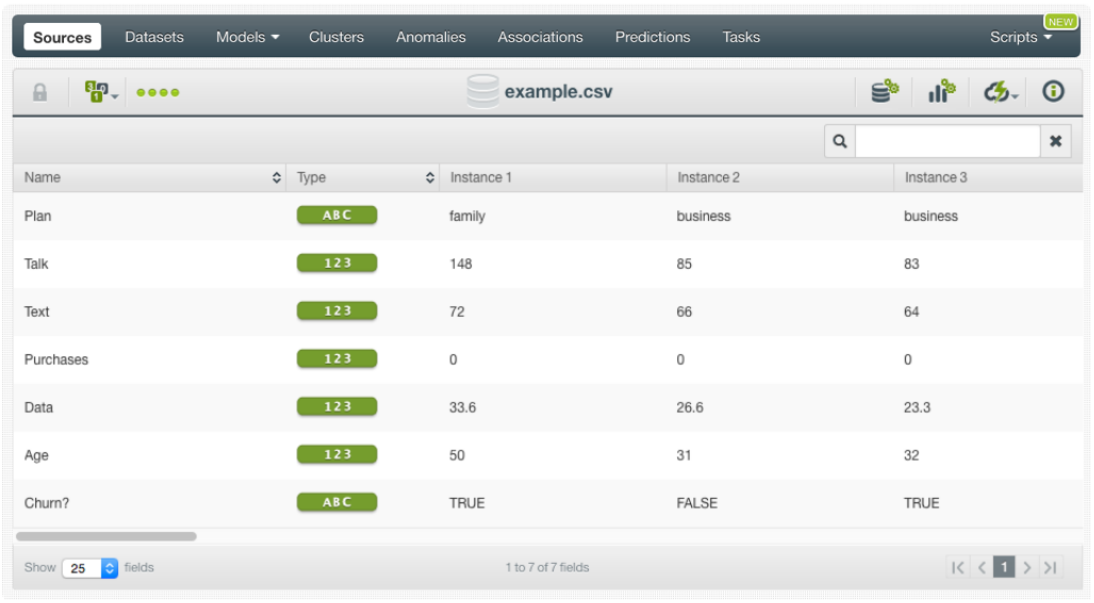
Type	Name			
CSV	example.csv	0min	406 bytes	

Show 10 data sources — 1 to 1 of 1 data sources

Figure 1.7: Source list view with a first source on it

BigML automatically assigns to each source a unique identifier, “**source/id**”, where **id** is a string of 24 alpha-numeric characters, e.g., “**source/570c9ae884622c5ecb008cb6**”. This special ID can be used to retrieve and refer to the source both via the BigML Dashboard and the BigML API.

Once you click on the newly created source, you will arrive at a new page whose URL matches with the assigned ID. You will see that BigML has parsed the source and automatically identified the type of each of its seven fields as shown in Figure 1.8.



Name	Type	Instance 1	Instance 2	Instance 3
Plan	ABC	family	business	business
Talk	123	148	85	83
Text	123	72	66	64
Purchases	123	0	0	0
Data	123	33.6	26.6	23.3
Age	123	50	31	32
Churn?	ABC	TRUE	FALSE	TRUE

Show 25 fields — 1 to 7 of 7 fields

Figure 1.8: A source view

Note: In a source view, BigML transposes rows and columns compared to your original data (compare Figure 1.6 and Figure 1.8). That is, each row is associated with one of the fields of your original data, and each column shows the corresponding values of an instance. It becomes much easier to navigate them using a web browser if they are arranged this way when sources contain hundreds or thousands of fields. A source view only shows the first 25 intances of your data. The main goal of this view is to help you quickly identify if BigML is parsing your data correctly.

CHAPTER 2

Source Types and Formats

Every BigML source has a type and a format. The type specifies its structure, for instance, whether it's a single or composite source. The format describes the type of data parsed to extract rows from, for example, it's CSV data or image.

2.1 Source Types

A BigML source is one of the following types:

- **Single**: A source created from a single non-zip file, such as a CSV, ARFF, JSON or image file. If a zip file contains only one file, the source created from it is also a **Single** source.
- **Composite**: A collection of other sources. The sources inside a composite source are called component sources. Component sources could be any type. A composite source can be created from an archive file containing multiple files, such as a zip or tar file. The detailed description about composite sources are in Chapter 4.
- **Inline**: A source created from inline data sent in the JSON of the creation request.

2.2 Source Formats

Again, a source's format tells what kind of data was parsed for extracting rows. To some extent, it's related to the format of the file Chapter 3 that was used to create the source. A BigML source can have following formats:

- **Table**: A source extracting rows parsing CSV or JSON data.
- **ARFF**: A source extracting rows parsing ARFF data.
- **Excel**: A source extracting data from **Microsoft Excel** files and similar files in binary formats.
- **Image**: A source containing images. See more details in Section 2.4.
- **Table+Image**: A composite source that extracts its rows from a CSV or JSON component, which contains cells referencing images which were also included as components of the table+image composite. See more details in (Section 4.4).
- **Mixed**: A composite source that contains heterogeneous components, which have different sets of fields. For a mixed source, there is no predefined method to extract data homogeneously from them. Therefore a mixed source has no fields and can not create datasets.
- **Empty**: A composite source that is empty or contains only empty sources.

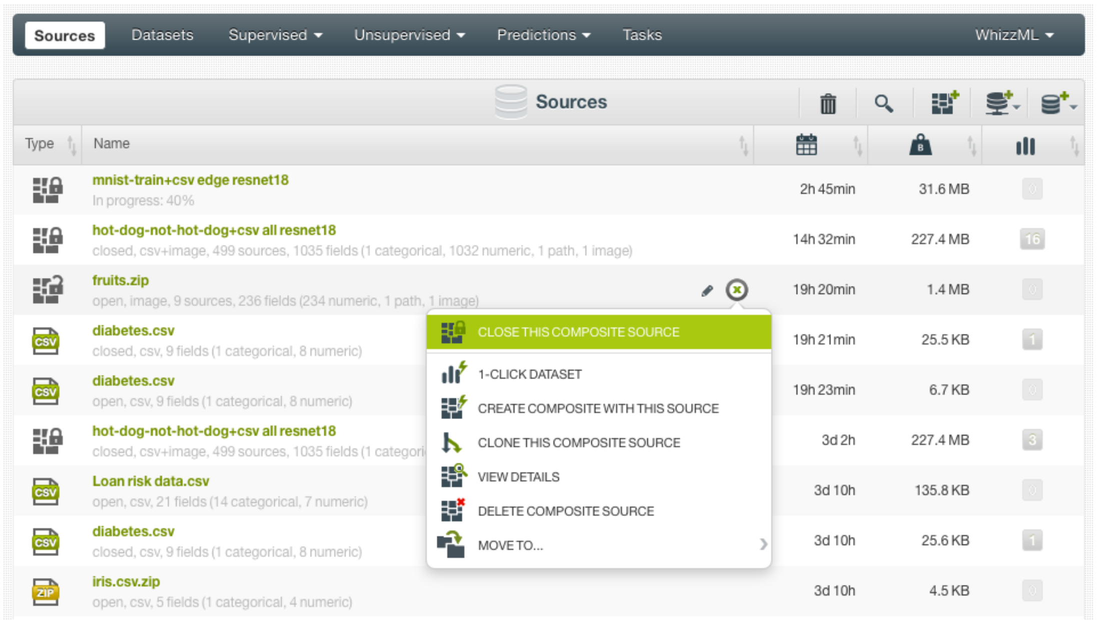
2.3 Open and Closed Sources

When a source is created, BigML tries automatically to figure out how to extract rows from the raw data. For instance, if the file is a CSV file, most of time BigML will discover what columns and column types it has, and they will become the source's fields.

Occasionally however, if BigML doesn't get everything right, users can edit that metadata and adjust the definitions of the source's fields until they are correct. Source is the only BigML resource that is initially mutable.

A source is **open** when it can be modified. All sources are initially created open.

A source can be changed from **open** to **closed**. For composite sources, this is done by performing the CLOSE THIS COMPOSITE SOURCE operation.



Type	Name			
	mnist-train+csv edge resnet18 In progress: 40%	2h 45min	31.6 MB	0
	hot-dog-not-hot-dog+csv all resnet18 closed, csv+image, 499 sources, 1035 fields (1 categorical, 1032 numeric, 1 path, 1 image)	14h 32min	227.4 MB	16
	fruits.zip open, image, 9 sources, 236 fields (234 numeric, 1 path, 1 image)	19h 20min	1.4 MB	0
CSV	diabetes.csv closed, csv, 9 fields (1 categorical, 8 numeric)	19h 21min	25.5 KB	1
CSV	diabetes.csv open, csv, 9 fields (1 categorical, 8 numeric)	19h 23min	6.7 KB	0
	hot-dog-not-hot-dog+csv all resnet18 closed, csv+image, 499 sources, 1035 fields (1 categori	3d 2h	227.4 MB	3
CSV	Loan risk data.csv open, csv, 21 fields (14 categorical, 7 numeric)	3d 10h	135.8 KB	0
CSV	diabetes.csv closed, csv, 9 fields (1 categorical, 8 numeric)	3d 10h	25.6 KB	1
ZIP	iris.csv.zip open, csv, 5 fields (1 categorical, 4 numeric)	3d 10h	4.5 KB	0

Figure 2.1: Close a composite source in the source list view

For a non-composite source (either single or inline source), it will be automatically closed when its dataset is created. When creating a dataset from a composite source, the composite source is automatically closed too.

For a composite source, the lock in its icon under "Type" column represents whether it is open or closed, as seen in the figure above. The open or closed status is also in the source's description string.

Once a source has been closed, it cannot be modified anymore.

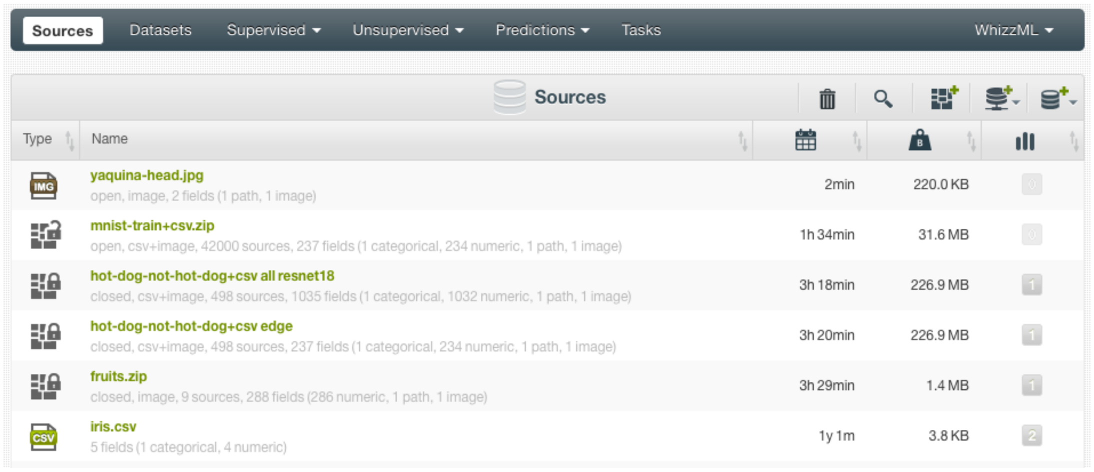
What if you want to change something in a closed source? You can **clone** it. Cloning a source creates a new and open source that shares the original underlying data. Cloning is quick and efficient. Because a cloned source is created as an open source, it can be modified again.

An open source can be cloned too.

2.4 Image Sources

Users can create sources using image files. Please refer to Chapter 3 to see the image file formats BigML supports.

When a source is created by a single image file, it's a **Single** source with an "Image" format, or is simply called an image source. You can see this in the **source list view**:



Type	Name			
IMG	yaquina-head.jpg open, image, 2 fields (1 path, 1 image)	2min	220.0 KB	0
	mnist-train+csv.zip open, csv+image, 42000 sources, 237 fields (1 categorical, 234 numeric, 1 path, 1 image)	1h 34min	31.6 MB	0
	hot-dog-not-hot-dog+csv all resnet18 closed, csv+image, 498 sources, 1035 fields (1 categorical, 1032 numeric, 1 path, 1 image)	3h 18min	226.9 MB	1
	hot-dog-not-hot-dog+csv edge closed, csv+image, 498 sources, 237 fields (1 categorical, 234 numeric, 1 path, 1 image)	3h 20min	226.9 MB	1
	fruits.zip closed, image, 9 sources, 288 fields (286 numeric, 1 path, 1 image)	3h 29min	1.4 MB	1
CSV	iris.csv 5 fields (1 categorical, 4 numeric)	1y 1m	3.8 KB	2

Figure 2.2: An image source in the source list view

Please note the “image” format in the source’s description string above, and the “image” icon in the “Type” column:

Figure 2.3: The image icon to represent an image source

An image source has two fields, the image, with type *image*, and its filename, with type *path*.

You can see the image information in the source: filename, pixel dimension, and file size.

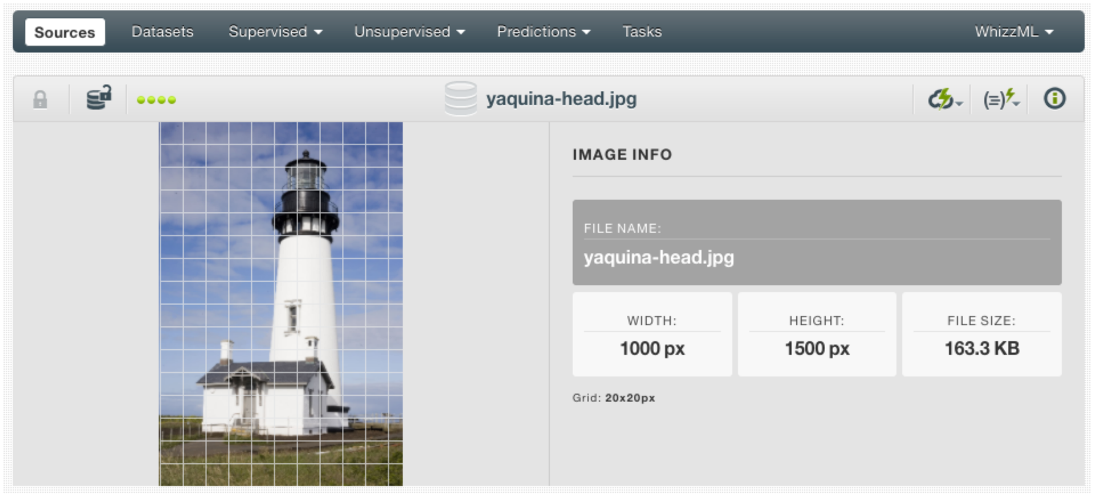


IMAGE INFO

FILE NAME:
yaquina-head.jpg

WIDTH:	HEIGHT:	FILE SIZE:
1000 px	1500 px	163.3 KB

Grid: **20x20px**

Figure 2.4: An image source

You can create composite sources using an image source. You can do this by clicking on the cloud action icon.

You cannot create a dataset using an image source. But you can clone and close image sources.

When a source is created by an archive file containing multiple images, such as a zip or tar file, it’s a **Composite** source, which is covered in Chapter 4.

CHAPTER 3

File Formats

BigML can process many different files to accommodate users. The following subsections review the file formats accepted by BigML.

3.1 Comma-Separated Values

The CSV[1] (Comma Separated Values) file format is a well-known format that has long been used for exchanging data between applications.

Your CSV files must conform to the following rules before creating a source in BigML:

- A CSV file uses plain text to store tabular data.
- In a CSV file, each line of the file is a record.
- Each record is usually separated by a comma (",") but other **separators** like the semi-colon (";"), the colon (":"), or the pipe "|", can also be used.
- Each record must contain exactly the same number of fields.
- Fields can be quoted using double quotes ("").
- Fields that contain commas (or the corresponding separator), double quotes, or line separators must be quoted.
- The character encoding must be UTF-8[2].
- Optionally, a CSV file can use the first line as a header to provide the names of each field.

BigML automatically parses your CSV files and is capable of dealing with most variants of the above options. It also provides you with different configuration options. (See Chapter 6.)

3.2 ARFF

BigML also accepts ARFF[3] (Attribute-Relation File Format) files. This type of file was first introduced by WEKA[4]. ARFF files basically come with a richer version of the header than a CSV file does which can define extra information about the type of the fields. An ARFF file separates its content into two sections: **Header** and **Data**. The **header** is used to define the name of the relation being modeled, the name of attributes, and their types. The **data** section contains the actual data using comma-separated values. (See Figure 3.1.)

[1] https://tools.ietf.org/html/rfc4180
[2] https://en.wikipedia.org/wiki/UTF-8
[3] http://www.cs.waikato.ac.nz/ml/weka/arff.html
[4] http://www.cs.waikato.ac.nz/ml/weka/

```
% Customer Churn Dataset
@RELATION Customers
@ATTRIBUTE Plan {'family', 'business', 'individual'}
@ATTRIBUTE Talk NUMERIC
@ATTRIBUTE Text NUMERIC
@ATTRIBUTE Purchases NUMERIC
@ATTRIBUTE Data NUMERIC
@ATTRIBUTE Age NUMERIC
@ATTRIBUTE Churn? {TRUE, FALSE}
@DATA
family, 148, 72, 0, 33.6, 50, TRUE
business, 85, 66, 0, 26.6, 31, FALSE
business, 83, 64, 0, 23.3, 32,TRUE
individual, 9,  66, 94, 28.1, 21, FALSE
family, 15, 0, 0, 35.3, 29, FALSE
individual, 66, 72, 175, 25.8, 51,TRUE
business, 0, 0, 0, 30, 32, TRUE
family, 18, 84, 230, 45.8, 31,TRUE
individual, 71, 110, 240, 45.4, 54, TRUE
family, 59, 64, 0, 27.4, 40, FALSE
```

Figure 3.1: An example of an ARFF file

3.3 JSON

BigML sources can also be created using JSON data in one of the following two formats:

3.3.1 List of Lists

A top-level list of lists of atomic values, each one defining a row. (See Figure 3.2.)

3.3.2 List of Dictionaries

A top-level list of dictionaries, where each dictionary's values represent the row values and the corresponding keys represent the column names as shown in Figure 3.3. The first dictionary defines the keys that will be selected.

```
[
    ["Plan","Talk","Text","Purchases","Data","Age","Churn?"],
    ["family", 148, 72, 0, 33.6, 50, "TRUE"],
    ["business", 85, 66, 0, 26.6, 31, "FALSE"],
    ["business", 83, 64, 0, 23.3, 32, "TRUE"],
    ["individual", 9,  66, 94, 28.1, 21, "FALSE"],
    ["family", 15, 0, 0, 35.3, 29, "FALSE"],
    ["individual", 66, 72, 175, 25.8, 51,"TRUE"],
    ["business", 0, 0, 0, 30, 32, "TRUE"],
    ["family", 18, 84, 230, 45.8, 31, "TRUE"],
    ["individual", 71, 110, 240, 45.4, 54, "TRUE"],
    ["family", 59, 64, 0, 27.4, 40, "FALSE"]
]
```

Figure 3.2: An example of a JSON source using a list of lists

```
[
  {
    "Plan": "family", "Talk": 148, "Text": 72, "Purchases": 0, "Data": 33.6,
    "Age": 50, "Churn?": "TRUE"
  },
  {
    "Plan": "business", "Talk": 85, "Text": 66, "Purchases": 0, "Data": 26.6,
    "Age": 31, "Churn?": "FALSE"
  },
  {
    "Plan": "business", "Talk": 83, "Text": 64, "Purchases": 0, "Data": 23.3,
    "Age": 32, "Churn?": "TRUE"
  },
  {
    "Plan": "individual", "Talk": 9, "Text": 66, "Purchases": 94, "Data": 28.1,
    "Age": 21, "Churn?": "FALSE"
  },
  {
    "Plan": "family", "Talk": 15, "Text": 0, "Purchases": 0, "Data": 35.3,
    "Age": 29, "Churn?": "FALSE"
  },
  {
    "Plan": "individual", "Talk": 66, "Text": 72, "Purchases": 175, "Data":
    25.8,
    "Age": 51, "Churn?": "TRUE"
  },
  {
    "Plan": "business", "Talk": 0, "Text": 0, "Purchases": 0, "Data": 30,
    "Age": 32, "Churn?": "TRUE"
  },
  {
    "Plan": "family", "Talk": 18, "Text": 84, "Purchases": 230, "Data": 45.8,
    "Age": 31, "Churn?": "TRUE"
  },
  {
    "Plan": "individual", "Talk": 71, "Text": 110, "Purchases": 240, "Data":
    45.4,
    "Age": 54, "Churn?": "TRUE"
  },
  {
    "Plan": "family", "Talk": 59, "Text": 64, "Purchases": 0, "Data": 27.4,
    "Age": 40, "Churn?": "FALSE"
  }
]
```

Figure 3.3: An example of a JSON source using a list of dictionaries

3.4 Other File Formats

BigML can also process **Microsoft Excel** and **Numbers for Mac** files. These files are usually readable in their native formats, but occasionally experience parsing issues. We recommend exporting them to CSV format before importing them to BigML to better guarantee proper parsing.

3.5 Image

BigML accepts many formats of image files: **jpg**, **png**, **gif**, **tiff**, **bmp**, **webp**, **cur**, **ico**, **pcx**, **psd**, **psb**. More formats will be supported in the future.

Note: Only the first frame of an animated (multi-framed) **gif** file will be used for model training.

3.6 Compressed Formats

Users can save bandwidth and time by creating sources from compressed files. The files can be **gzipped** (**.gz**), **compressed** (**.bz2**), as well as **zipped** (**.zip**). BigML also supports compressed TAR archive files, such as **.tar.gz**, **.tgz**, **.tar.bz2** and **.tbz2**.

When creating sources from compressed formats, if the archive contains only one file, a **single** source will be created. If the archive contains two or more files, a **composite** source will be created.

CHAPTER 4

Composite Sources

A composite source is a collection of other sources, which are called component sources. Component sources can be any type (Section 2.1), they can be single sources, they can be composite sources as well. In other words, composite sources can be nested. A composite source can only be a component source if it's closed (See Section 2.3).

BigML uses the icon in Figure 4.1 to represent a BigML compoiste source.

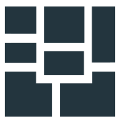
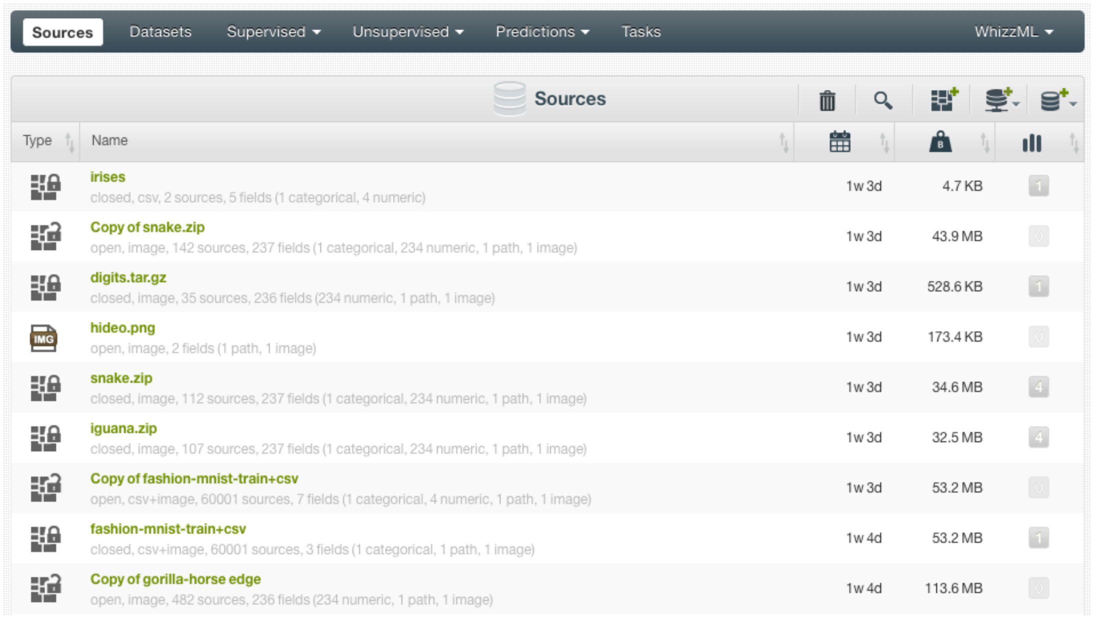


Figure 4.1: Composite source icon

You can see the composite source icons in the "Type" column under the **source list view**:



Type	Name			
	irises closed, csv, 2 sources, 5 fields (1 categorical, 4 numeric)	1w 3d	4.7 KB	1
	Copy of snake.zip open, image, 142 sources, 237 fields (1 categorical, 234 numeric, 1 path, 1 image)	1w 3d	43.9 MB	0
	digits.tar.gz closed, image, 35 sources, 236 fields (234 numeric, 1 path, 1 image)	1w 3d	528.6 KB	1
IMG	**hideo.png** open, image, 2 fields (1 path, 1 image)	1w 3d	173.4 KB	0
	snake.zip closed, image, 112 sources, 237 fields (1 categorical, 234 numeric, 1 path, 1 image)	1w 3d	34.6 MB	4
	iguana.zip closed, image, 107 sources, 237 fields (1 categorical, 234 numeric, 1 path, 1 image)	1w 3d	32.5 MB	4
	Copy of fashion-mnist-train+csv open, csv+image, 60001 sources, 7 fields (1 categorical, 4 numeric, 1 path, 1 image)	1w 3d	53.2 MB	0
	fashion-mnist-train+csv closed, csv+image, 60001 sources, 3 fields (1 categorical, 1 path, 1 image)	1w 4d	53.2 MB	1
	Copy of gorilla-horse edge open, image, 482 sources, 236 fields (234 numeric, 1 path, 1 image)	1w 4d	113.6 MB	0

Figure 4.2: Composite sources under the source list view

The open/closed lock in the icons of composite sources signifies their open/closed status.

Figure 4.3: Icons for open and closed composite sources

When all the component sources of a composite source have the same fields, the composite source will inherit those fields, and a dataset can be created from it. The result will just be the concatenation of all the rows extracted from each component source inside the composite source. For instance, a composite source may have several CSV component sources, which were created from several CSV files with exactly the same fields, and the composite source will inherit those fields and behave like a single source.

A composite source is created open (Section 2.3), so is any other source. Being open means it's modifiable. The following operations can be performed to an open composite source:

- Add component sources.
- Remove component sources.
- Replace the full list of component sources with a new list.

A source can belong to as many composite sources as desired. However, when a source belongs to one or more composite sources, it cannot be modified, regardless of whether it is open or closed. This way all composite sources see the same version of the source all the time.

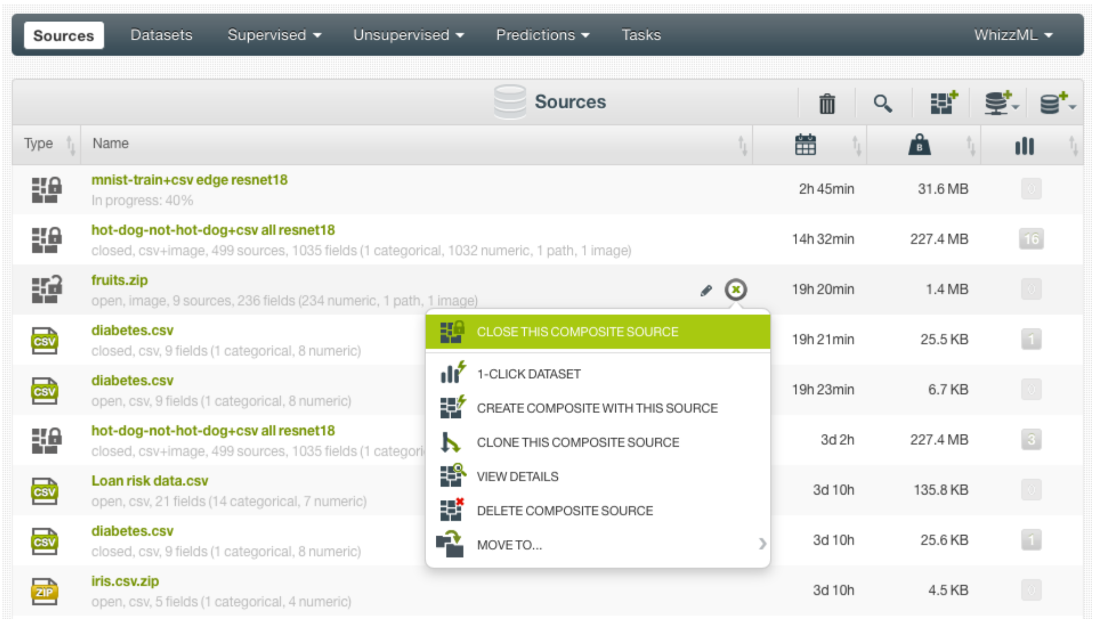
When adding or removing component sources to a composite source, it will check the compatibility of the fields of all its component sources, and update its own set of fields.

Once a composite source's components are finalized and it must be closed to create datasets. When closing a composite source, all its component surces will be automatically closed.

You may create a dataset from an open composite source, but the composite source will be closed at the same time.

Unlike all other types of sources, composite sources must be **explicitly** closed by an API call or UI action. This is mainly to avoid accidentally closing a composite source by mistake. For instance, since composite sources can have a huge number of component sources, they may be shared and worked on by several collaborators. Then the mistake of accidentally closing the component sources would be costly.

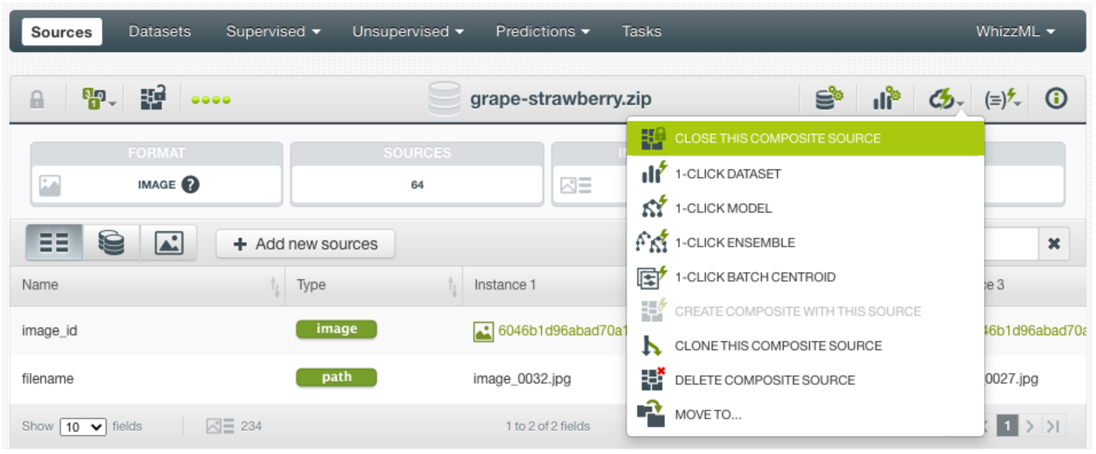
To close a composite source, click on the Close This Composite Source menu item, either from the source's context menu in a **source list view**,



Type	Name			
	mnist-train+csv edge resnet18 In progress: 40%	2h 45min	31.6 MB	0
	hot-dog-not-hot-dog+csv all resnet18 closed, csv+image, 499 sources, 1035 fields (1 categorical, 1032 numeric, 1 path, 1 image)	14h 32min	227.4 MB	16
	fruits.zip open, image, 9 sources, 236 fields (234 numeric, 1 path, 1 image)	19h 20min	1.4 MB	0
CSV	diabetes.csv closed, csv, 9 fields (1 categorical, 8 numeric)	19h 21min	25.5 KB	1
CSV	diabetes.csv open, csv, 9 fields (1 categorical, 8 numeric)	19h 23min	6.7 KB	0
	hot-dog-not-hot-dog+csv all resnet18 closed, csv+image, 499 sources, 1035 fields (1 categori	3d 2h	227.4 MB	3
CSV	Loan risk data.csv open, csv, 21 fields (14 categorical, 7 numeric)	3d 10h	135.8 KB	0
CSV	diabetes.csv closed, csv, 9 fields (1 categorical, 8 numeric)	3d 10h	25.6 KB	1
ZIP	iris.csv.zip open, csv, 5 fields (1 categorical, 4 numeric)	3d 10h	4.5 KB	0

Figure 4.4: Close a composite source in the source list view

or from the cloud action menu in the **source view**.



Name	Type	Instance 1	e 3
image_id	image	6046b1d96abad70a1	46b1d96abad70a
filename	path	image_0032.jpg	0027.jpg

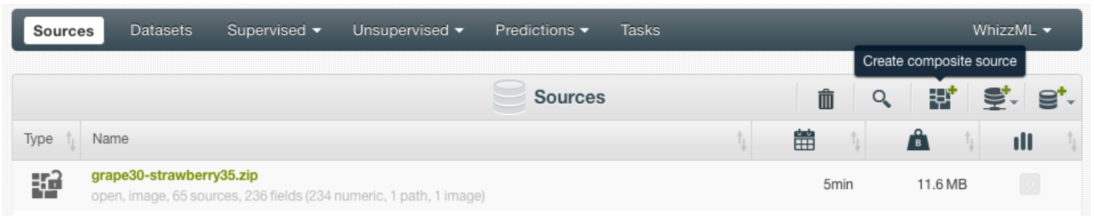
Figure 4.5: Close a composite source in the source view

Note: BigML currently limits the number of component sources in a composite source to 445,000. In other words, a composite source can have at most 445,000 components.

4.1 Creating Composite Sources From Existing Sources

A composite source can be created by combining existing sources.

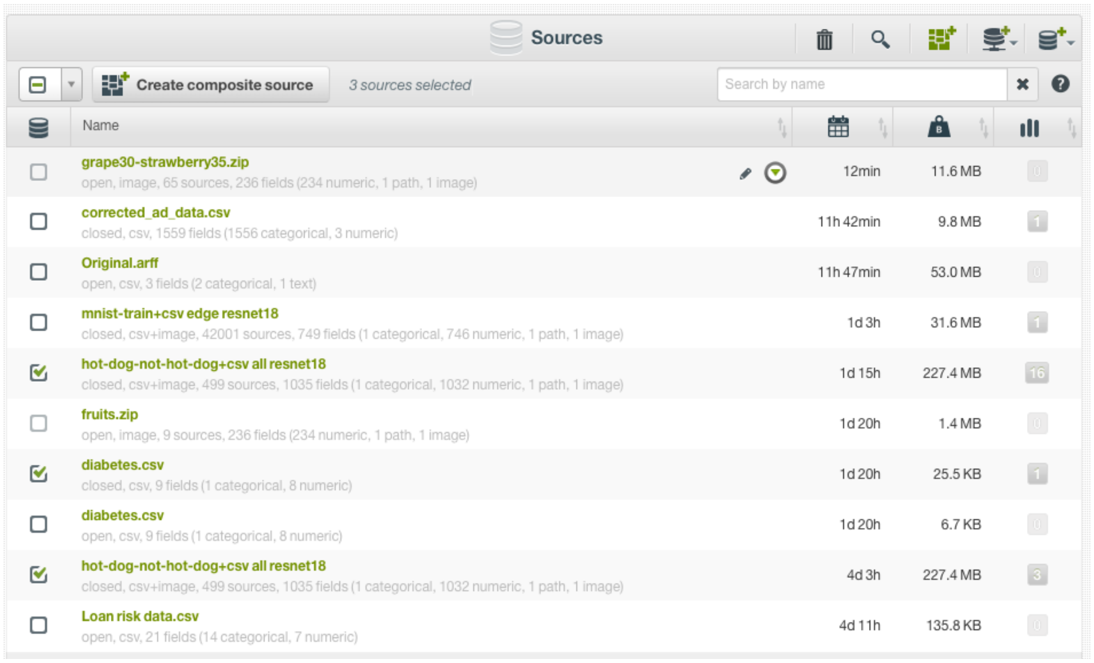
Under the **source list view**, there is a “Create composite source” icon on the source action bar:



Type	Name				
	grape30-strawberry35.zip open, image, 65 sources, 236 fields (234 numeric, 1 path, 1 image)	5min	11.6 MB	0

Figure 4.6: Create composite source under the source list view

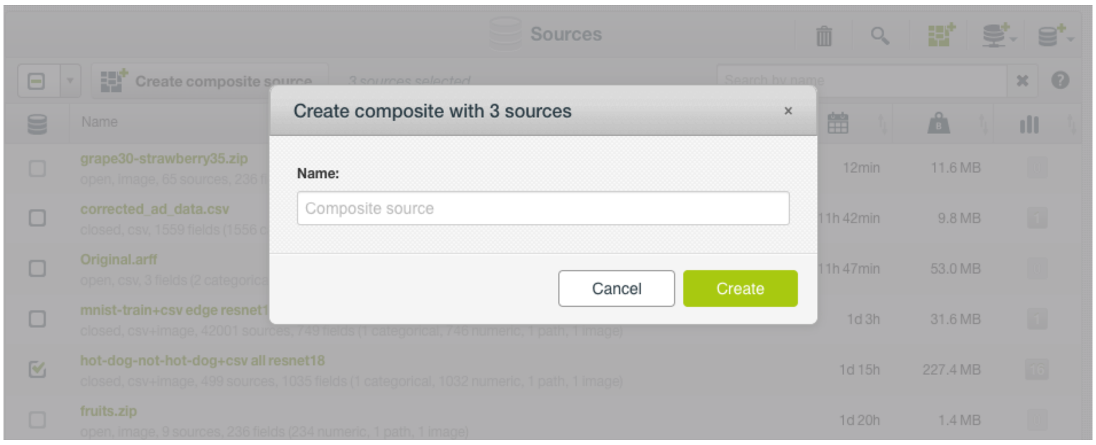
Clicking on it will bring the **Composite source creation view**:



	Name			
☐	grape30-strawberry35.zip open, image, 65 sources, 236 fields (234 numeric, 1 path, 1 image)	12min	11.6 MB	0
☐	corrected_ad_data.csv closed, csv, 1559 fields (1556 categorical, 3 numeric)	11h 42min	9.8 MB	1
☐	Original.arff open, csv, 3 fields (2 categorical, 1 text)	11h 47min	53.0 MB	0
☐	mnist-train+csv edge resnet18 closed, csv+image, 42001 sources, 749 fields (1 categorical, 746 numeric, 1 path, 1 image)	1d 3h	31.6 MB	1
✓	hot-dog-not-hot-dog+csv all resnet18 closed, csv+image, 499 sources, 1035 fields (1 categorical, 1032 numeric, 1 path, 1 image)	1d 15h	227.4 MB	16
☐	fruits.zip open, image, 9 sources, 236 fields (234 numeric, 1 path, 1 image)	1d 20h	1.4 MB	0
✓	diabetes.csv closed, csv, 9 fields (1 categorical, 8 numeric)	1d 20h	25.5 KB	1
☐	diabetes.csv open, csv, 9 fields (1 categorical, 8 numeric)	1d 20h	6.7 KB	0
✓	hot-dog-not-hot-dog+csv all resnet18 closed, csv+image, 499 sources, 1035 fields (1 categorical, 1032 numeric, 1 path, 1 image)	4d 3h	227.4 MB	3
☐	Loan risk data.csv open, csv, 21 fields (14 categorical, 7 numeric)	4d 11h	135.8 KB	0

Figure 4.7: Composite source creation view

Users can use the checkboxes to select the component sources, then clicking on the Create composite source button on top will bring the composite source creation confirmation box:


Create composite with 3 sources

Name:

Composite source

Figure 4.8: Create composite source confirmation

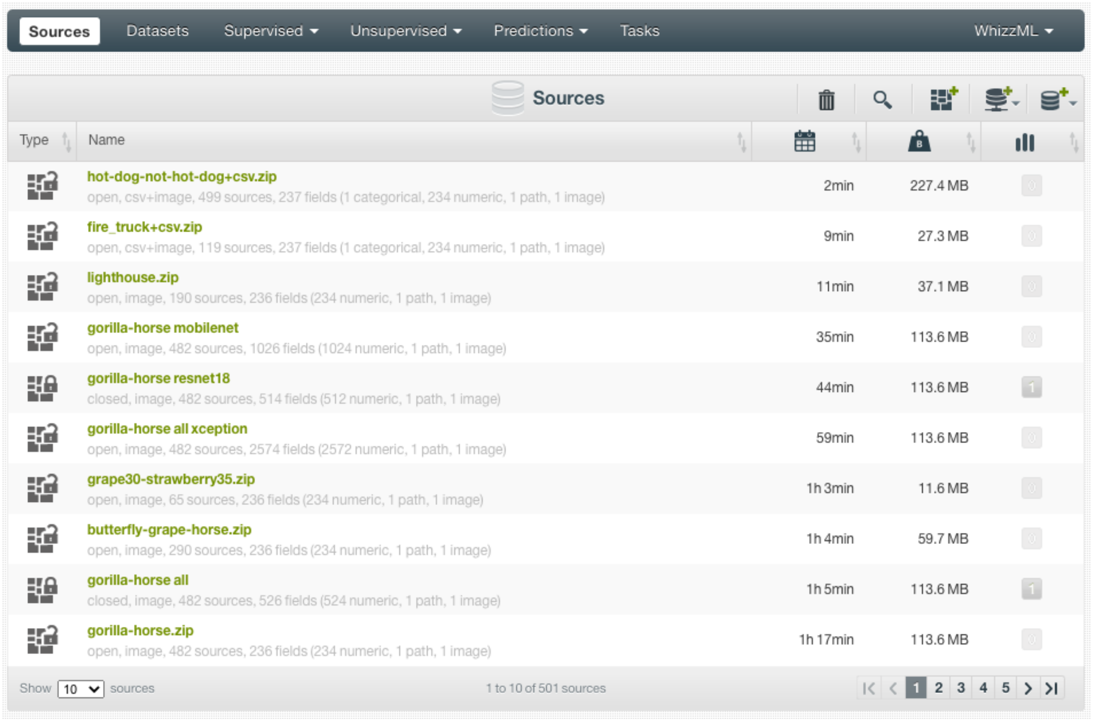
It lists the number of component sources and prompts users for the name of the composite source. Finish the creation by clicking on the Create button.

Remember: you can create a composite source with any number and any type of sources. However, only composite sources with homogeneous component sources, which mean they have the same format and exactly the same fields, can be used to create datasets.

4.2 Creating Composite Sources From Archive Files

Another way to create a composite source is by uploading an archive file, such as zip or tar, that contains more than one files. BigML supports a range of archive formats (See Chapter 3).

When an archive file of multiple files is uploaded, BigML will automatically create one source for each file inside the archive, and put them all together in a composite source. See the composite sources created by uploading archive files below:



Type	Name			
	hot-dog-not-hot-dog+csv.zip open, csv+image, 499 sources, 237 fields (1 categorical, 234 numeric, 1 path, 1 image)	2min	227.4 MB	0
	fire_truck+csv.zip open, csv+image, 119 sources, 237 fields (1 categorical, 234 numeric, 1 path, 1 image)	9min	27.3 MB	0
	lighthouse.zip open, image, 190 sources, 236 fields (234 numeric, 1 path, 1 image)	11min	37.1 MB	0
	gorilla-horse mobilenet open, image, 482 sources, 1026 fields (1024 numeric, 1 path, 1 image)	35min	113.6 MB	0
	gorilla-horse resnet18 closed, image, 482 sources, 514 fields (512 numeric, 1 path, 1 image)	44min	113.6 MB	1
	gorilla-horse all xception open, image, 482 sources, 2574 fields (2572 numeric, 1 path, 1 image)	59min	113.6 MB	0
	grape30-strawberry35.zip open, image, 65 sources, 236 fields (234 numeric, 1 path, 1 image)	1h 3min	11.6 MB	0
	butterfly-grape-horse.zip open, image, 290 sources, 236 fields (234 numeric, 1 path, 1 image)	1h 4min	59.7 MB	0
	gorilla-horse all closed, image, 482 sources, 526 fields (524 numeric, 1 path, 1 image)	1h 5min	113.6 MB	1
	gorilla-horse.zip open, image, 482 sources, 236 fields (234 numeric, 1 path, 1 image)	1h 17min	113.6 MB	0

Show 10 sources 1 to 10 of 501 sources 1 2 3 4 5

Figure 4.9: Composite sources created by archive files

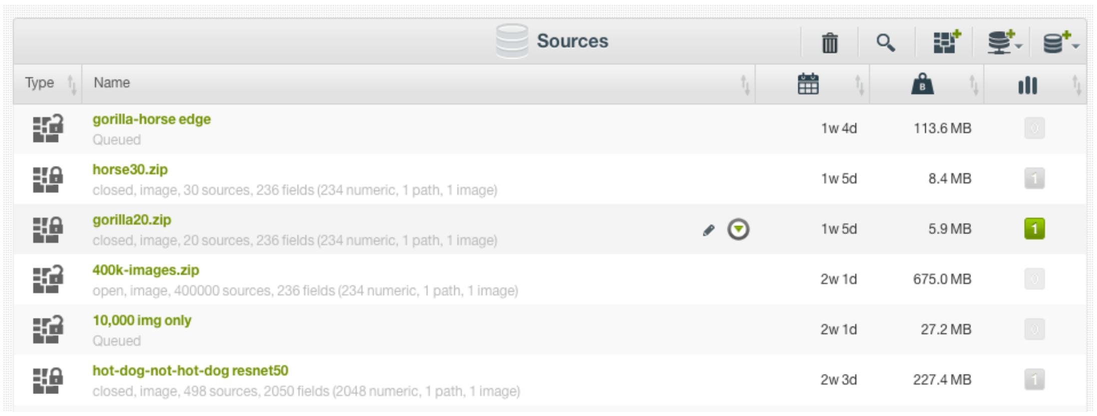
4.3 Image Composite Sources

When the component sources of a composite source are all image sources (See Section 2.4), that is, they all have the format "Image" (See Section 2.2), the composite source will also have the format "Image". This is an image composite source.

An image composite source represents a collection of images and is more useful in marchine learning than a single image. While a single image can be used for prediction, a machine learning model is trained on a collection of images, not a single image. A dataset intended for image applications contains many images, with each image as an instance in the dataset.

While single image sources cannot be used to create dataset, image composite sources can. In the dataset created from an image composite source, every row will correspond to one of the images in the composite source, and have a column representing the image data (type *image*), a second column as its filename (type *path*), and a possible third column as its label.

Note: As stated before, BigML allows up to 445,000 components in a composite source. Hence an image composite source can have at most 445,000 images.



Sources

Type	Name			
	gorilla-horse edge Queued	1w 4d	113.6 MB	
	horse30.zip closed, image, 30 sources, 236 fields (234 numeric, 1 path, 1 image)	1w 5d	8.4 MB	1
	gorilla20.zip closed, image, 20 sources, 236 fields (234 numeric, 1 path, 1 image)	1w 5d	5.9 MB	1
	400k-images.zip open, image, 400000 sources, 236 fields (234 numeric, 1 path, 1 image)	2w 1d	675.0 MB	
	10,000 img only Queued	2w 1d	27.2 MB	
	hot-dog-not-hot-dog resnet50 closed, image, 498 sources, 2050 fields (2048 numeric, 1 path, 1 image)	2w 3d	227.4 MB	1

Figure 4.10: Image composite sources

4.3.1 Automatic Image Labels

Image labels are important in machine learning, and especially they are indispensable in image classification. A common practice in the industry is to group images by folders, with the folder names being their labels. BigML accommodates such practice by providing automatic image labeling with folders.

When an image composite source is created by uploading an archive file, most of times, it will have three fixed fields plus a set of autogenerated image fields (See next section Subsection 4.3.2). The three fixed fields are inherited from the single image sources as follows:

- **image_id** is a field with optype *image* that contains the identifier of the corresponding single image source.
- **filename** is a field with optype *path* that contains the path inside of the archive file for each image file.
- **label** is a categorical field whose values are extracted from the filenames. More specifically, they correspond to the innermost directory segment of the filename.

  For instance, the label is “foo” for “bar/baz/foo/image.jpg” or is missing for “another_image.png”.

The **label** field is omitted in the following cases:

- If BigML doesn’t detect any directory names in the filenames.
- If the source creation request is made via API and it includes the property *disable_autolabel* which is set to the Boolean value *true*.

Zip file grape-strawberry-dir.zip has two folders named *grape* and *strawberry*:

```
Archive:  grape-strawberry-dir.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  11-19-2020 22:05   grape/
    44691  01-04-2006 21:33   grape/092_0020.jpg
    13527  01-04-2006 21:33   grape/092_0008.jpg
    37413  02-18-2006 16:46   grape/092_0009.jpg
    ......
        0  11-19-2020 22:05   strawberry/
    10378  05-17-2020 20:57   strawberry/image_0032.jpg
     8912  05-17-2020 20:57   strawberry/image_0026.jpg
     8717  05-17-2020 20:57   strawberry/image_0027.jpg
     ......
```

Each folder contains a number of images. Once this zip is uploaded to the BigML Dashboard, an image composite source will be created:

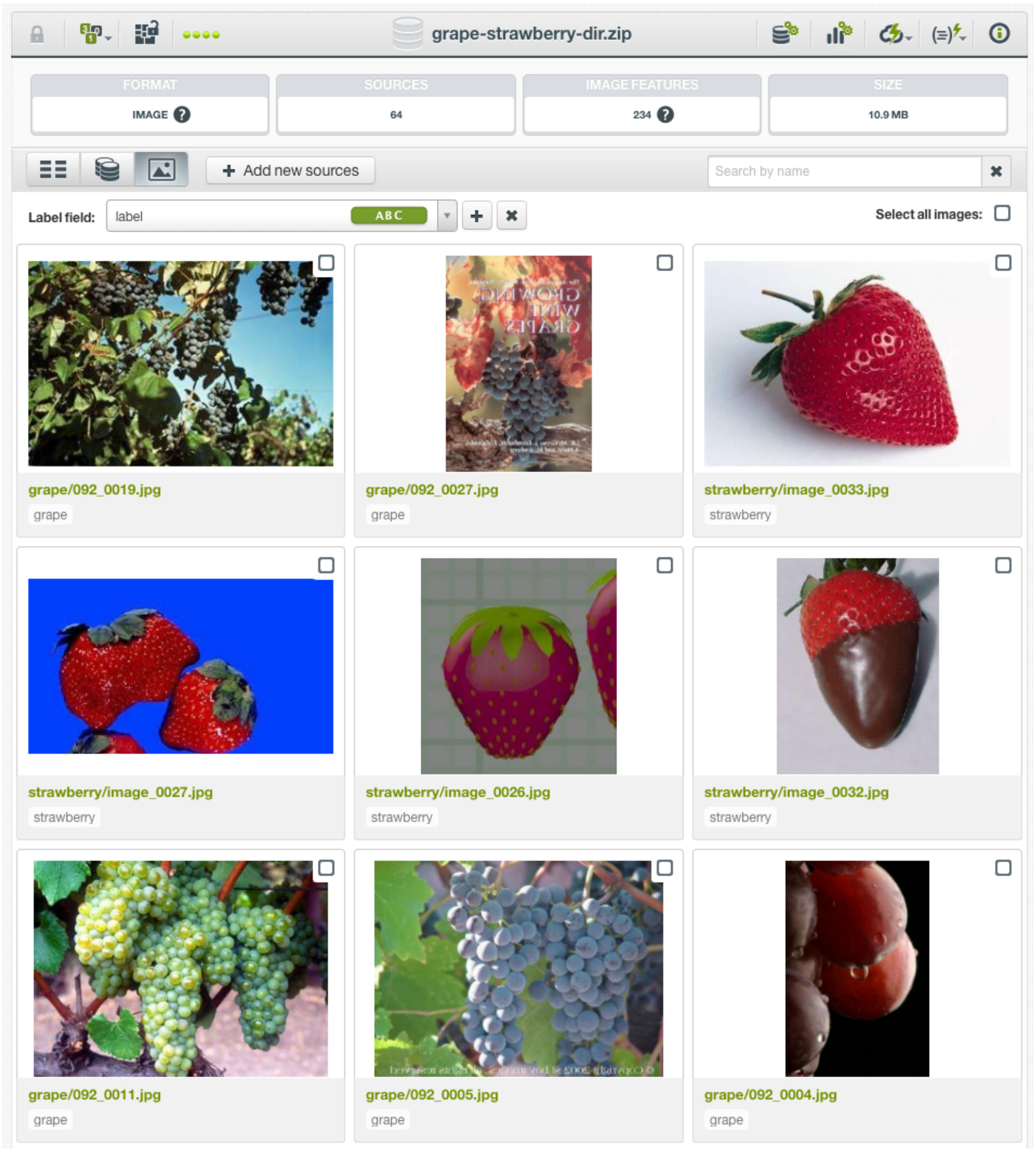
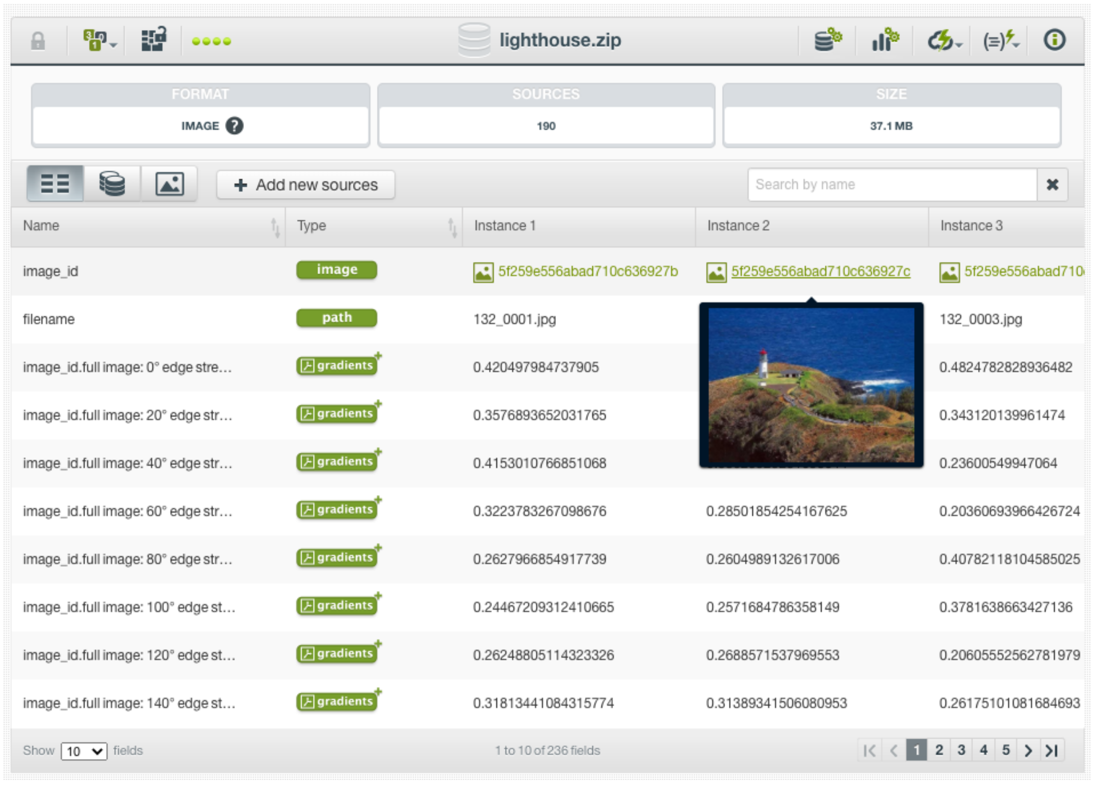


Figure 4.11: Image labels automatically generated from folder names

As shown, image labels will be automatically generated from their folder names.

4.3.2 Extracted Image Features

When an image composite source is created, BigML analyzes the images and automatically generate a set of numeric features for each image. Those features appear as new, added fields in the composite source. While those fields can be viewed in the source, their values cannot be changed because they are generated fields.



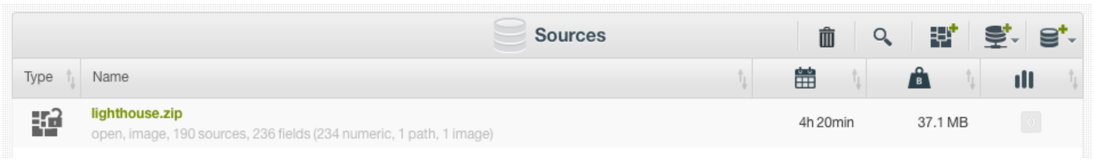
Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	5f259e556abad710c636927b	5f259e556abad710c636927c	5f259e556abad710
filename	path	132_0001.jpg		132_0003.jpg
image_id.full image: 0° edge stre…	gradients	0.420497984737905		0.4824782828936482
image_id.full image: 20° edge str…	gradients	0.3576893652031765		0.343120139961474
image_id.full image: 40° edge str…	gradients	0.4153010766851068		0.23600549947064
image_id.full image: 60° edge str…	gradients	0.3223783267098676	0.28501854254167625	0.20360693966426724
image_id.full image: 80° edge str…	gradients	0.2627966854917739	0.2604989132617006	0.40782118104585025
image_id.full image: 100° edge st…	gradients	0.24467209312410665	0.2571684786358149	0.3781638663427136
image_id.full image: 120° edge st…	gradients	0.26248805114323326	0.2688571537969553	0.20605552562781979
image_id.full image: 140° edge st…	gradients	0.31813441084315774	0.31389341506080953	0.26175101081684693

Show 10 fields   1 to 10 of 236 fields   1 2 3 4 5

Figure 4.12: Extracted image features in a composite source

Image features are only generated for image composite sources, they are not generated for single image sources.

By default, BigML extracts 234 features per image, representing its histogram of gradients. You can see the number of fields in the description string under the source name:

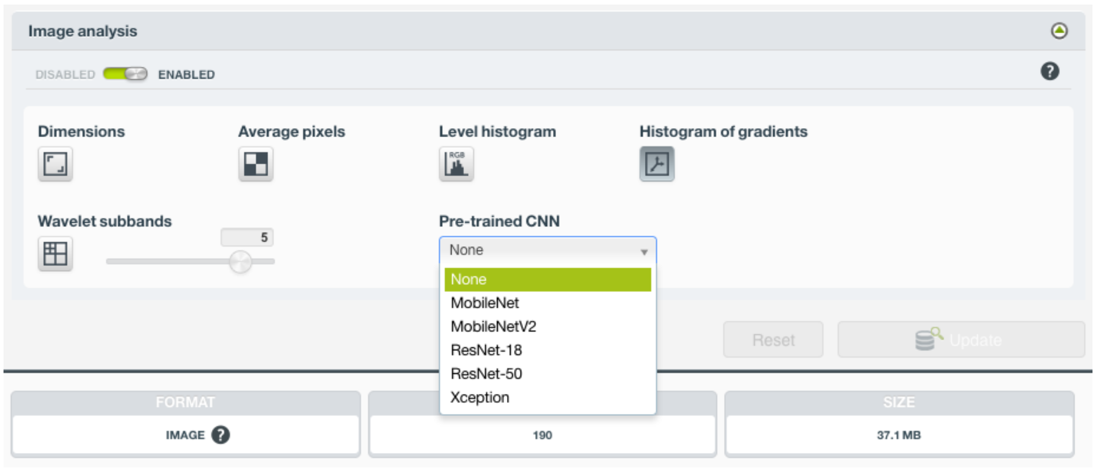


Type	Name			
	lighthouse.zip open, image, 190 sources, 236 fields (234 numeric, 1 path, 1 image)	4h 20min	37.1 MB	

Figure 4.13: Image feature fields in the description string

In total, BigML offers five sets of extracted image features:

- Dimensions
- Average pixels
- Level histogram
- Histogram of gradients
- Wavelet subbands



Wavelet subbands 5

Pre-trained CNN None

FORMAT		SIZE
IMAGE	190	37.1 MB

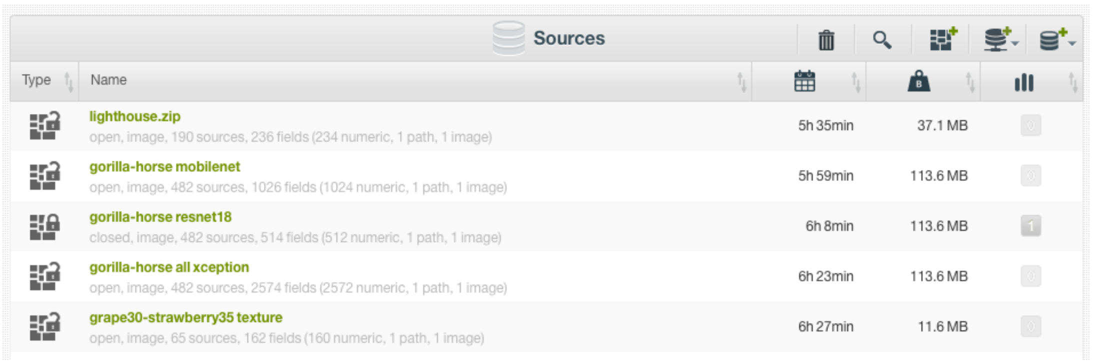
Figure 4.14: Options of image features in the source configuration

These features are all numeric values describing the structure and contents of the images in the composite source.

BigML also offers five pre-trained convolutional neural networks (CNNs):

- MobileNet
- MoibleNetV2
- ResNet-18
- ResNet-50 (Available to customers of virtual private clouds and private deployments)
- Xception (Available to customers of virtual private clouds and private deployments)

Users can configure and select different combinations of image features in the Source Configuration Options (Chapter 6). Different configurations are reflected by the numbers of fields in the description strings under the source names:



Type	Name			
	lighthouse.zip open, image, 190 sources, 236 fields (234 numeric, 1 path, 1 image)	5h 35min	37.1 MB	
	gorilla-horse mobilenet open, image, 482 sources, 1026 fields (1024 numeric, 1 path, 1 image)	5h 59min	113.6 MB	
	gorilla-horse resnet18 closed, image, 482 sources, 514 fields (512 numeric, 1 path, 1 image)	6h 8min	113.6 MB	1
	gorilla-horse all xception open, image, 482 sources, 2574 fields (2572 numeric, 1 path, 1 image)	6h 23min	113.6 MB	
	grape30-strawberry35 texture open, image, 65 sources, 162 fields (160 numeric, 1 path, 1 image)	6h 27min	11.6 MB	

Figure 4.15: Different image feature fields in the description string

Please see "Image Analysis" (Section 6.9) for the detailed explanation of image features and how to configure them.

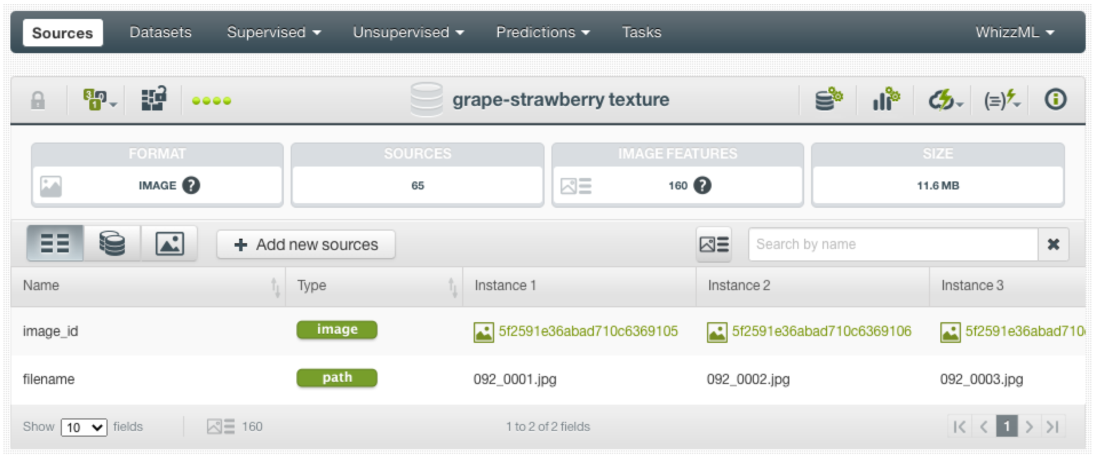
By exposing those image features as explicit fields in your datasets, they can be used by all machine learning algorithms and models on the BigML platform, just as any other feature.

4.3.3 Image Composite Views

On the Dashboard, there are three views for an image composite source. When users click on an image composite source in the **source list view**, it enters **fields view**.

4.3.3.1 Fields View

A **fields view** lists all the fields in the composite source:



FORMAT	SOURCES	IMAGE FEATURES	SIZE
IMAGE	65	160	11.6 MB

Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	5f2591e36abad710c6369105	5f2591e36abad710c6369106	5f2591e36abad710
filename	path	092_0001.jpg	092_0002.jpg	092_0003.jpg

Show 10 fields 160 1 to 2 of 2 fields 1

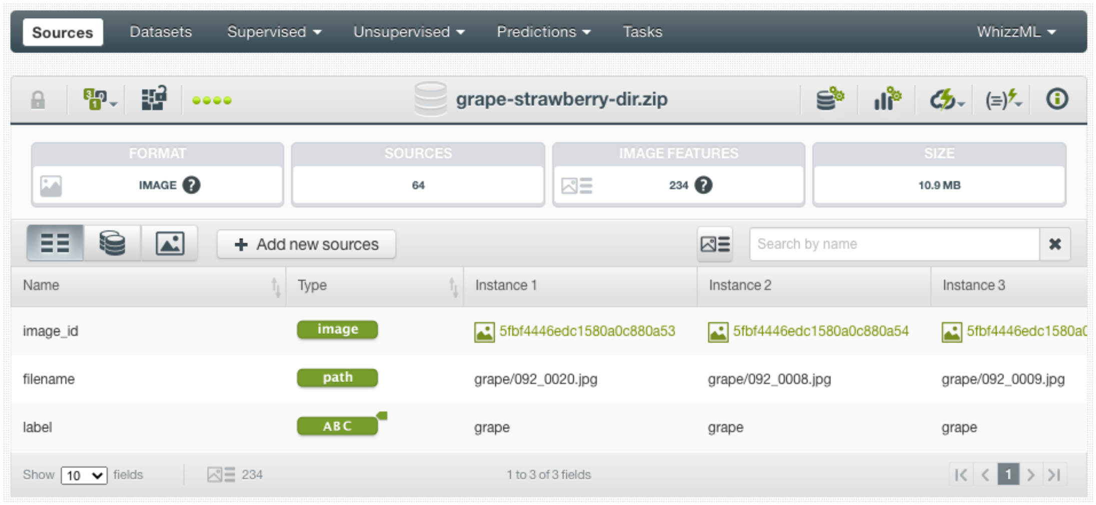
Figure 4.16: Fields view of an image composite source

As seen in the figure above, an image comes with two fields, with the following icons representing their types:

Figure 4.17: Image field type

Figure 4.18: Path field type

When the images are organized by folders with the folder names intended as their labels (see Subsection 4.3.1), an image comes with three fields. In addition to the two fields "image_id" and "filename", as shown above, whose respective types are *image* and *path*, it has a third field named "label", which type is *categorical*.



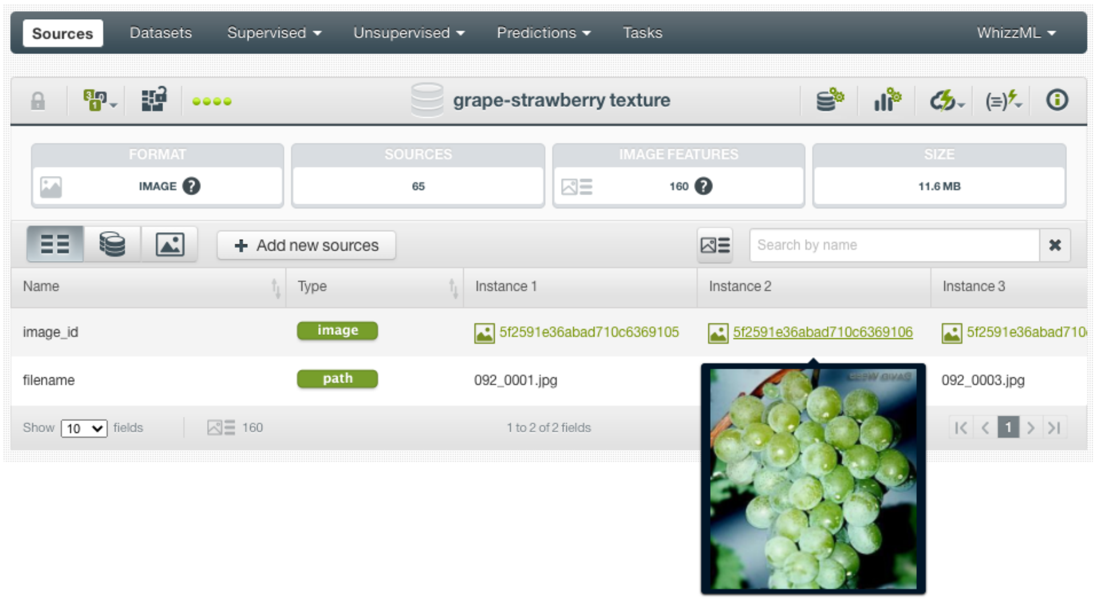
FORMAT	SOURCES	IMAGE FEATURES	SIZE
IMAGE	64	234	10.9 MB

Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	5fbf4446edc1580a0c880a53	5fbf4446edc1580a0c880a54	5fbf4446edc1580a0
filename	path	grape/092_0020.jpg	grape/092_0008.jpg	grape/092_0009.jpg
label	ABC	grape	grape	grape

Show 10 fields 234 1 to 3 of 3 fields 1

Figure 4.19: Fields view of an image composite source with a label field

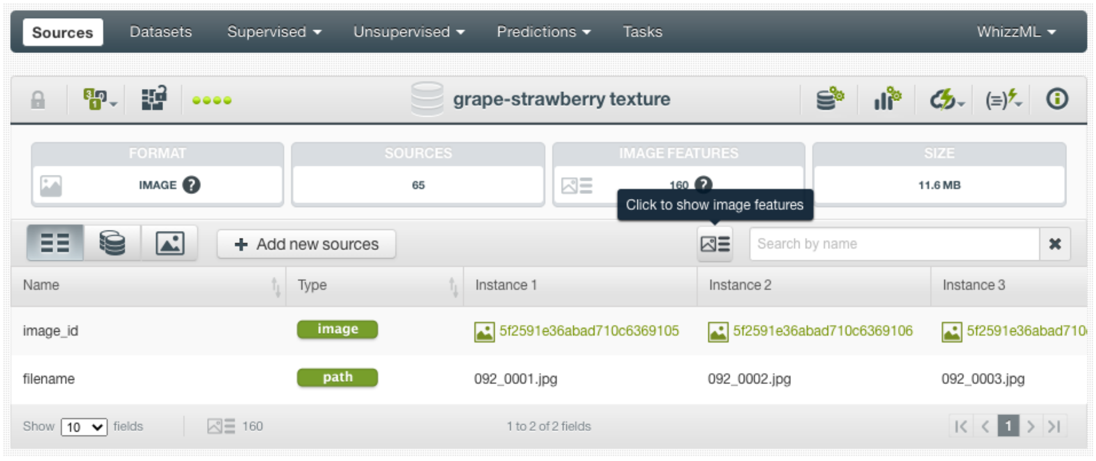
Users can preview all the fields, including the images by hovering the mouse over the image IDs:



Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	5f2591e36abad710c6369105	5f2591e36abad710c6369106	5f2591e36abad710
filename	path	092_0001.jpg		092_0003.jpg

Figure 4.20: Preview of an image in an image composite source

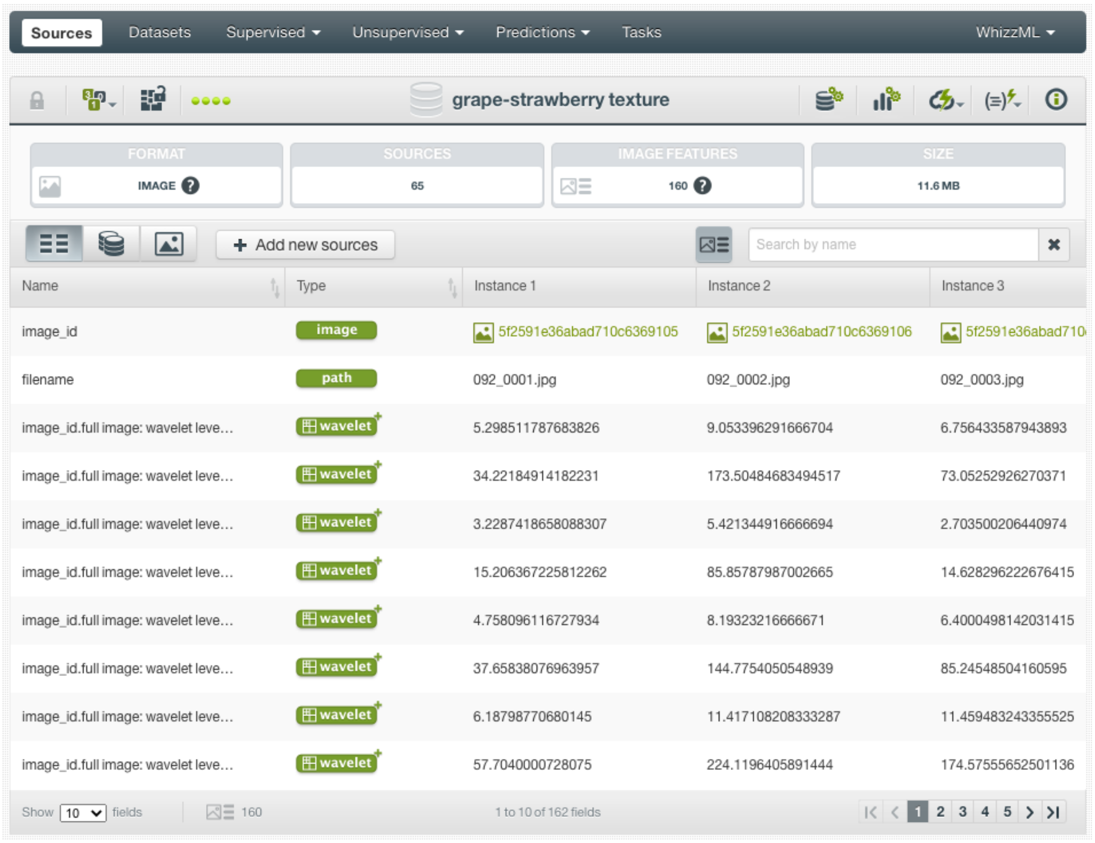
By default, all fields of the extracted image features are hidden. That's why only two image fields are shown in Figure 4.20. However, you can click on the "show image features" icon next to the search box as shown below:



Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	5f2591e36abad710c6369105	5f2591e36abad710c6369106	5f2591e36abad710
filename	path	092_0001.jpg	092_0002.jpg	092_0003.jpg

Figure 4.21: Icon to click to show image feature fields

Then, you can preview all fields, including the image features. At this point, when all fields are shown, you can click on the same icon to hide the fields of image features.



Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	5f2591e36abad710c6369105	5f2591e36abad710c6369106	5f2591e36abad710
filename	path	092_0001.jpg	092_0002.jpg	092_0003.jpg
image_id.full image: wavelet leve...	wavelet	5.298511787683826	9.053396291666704	6.756433587943893
image_id.full image: wavelet leve...	wavelet	34.22184914182231	173.50484683494517	73.05252926270371
image_id.full image: wavelet leve...	wavelet	3.2287418658088307	5.421344916666694	2.703500206440974
image_id.full image: wavelet leve...	wavelet	15.206367225812262	85.85787987002665	14.628296222676415
image_id.full image: wavelet leve...	wavelet	4.758096116727934	8.19323216666671	6.4000498142031415
image_id.full image: wavelet leve...	wavelet	37.65838076963957	144.7754050548939	85.24548504160595
image_id.full image: wavelet leve...	wavelet	6.18798770680145	11.417108208333287	11.459483243355525
image_id.full image: wavelet leve...	wavelet	57.7040000728075	224.1196405891444	174.57555652501136

Show 10 fields 160 1 to 10 of 162 fields 1 2 3 4 5

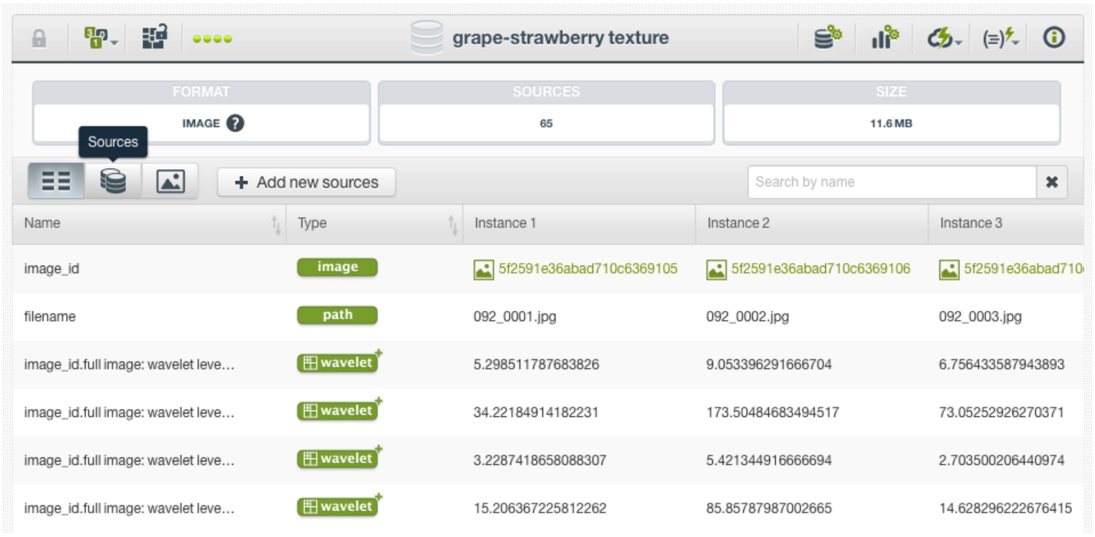
Figure 4.22: Preview of all fields of a composite source

This **fields view** of a composite source is equivalent to the source view of a non-composite source (Figure 1.8).

Again, in a **fields view**, BigML transposes rows and columns compared to the original data. That is, each row is associated with one of the fields of the original data, and each column shows the corresponding values of an instance. It becomes much easier to navigate them using a web browser if they are arranged this way when sources contain hundreds or thousands of fields.

A **fields view** only shows the first 25 instances of the data. The main goal of this view is to help quickly identify if BigML is parsing the data correctly.

Using the tabs on top of the field list, users can switch to other two views.

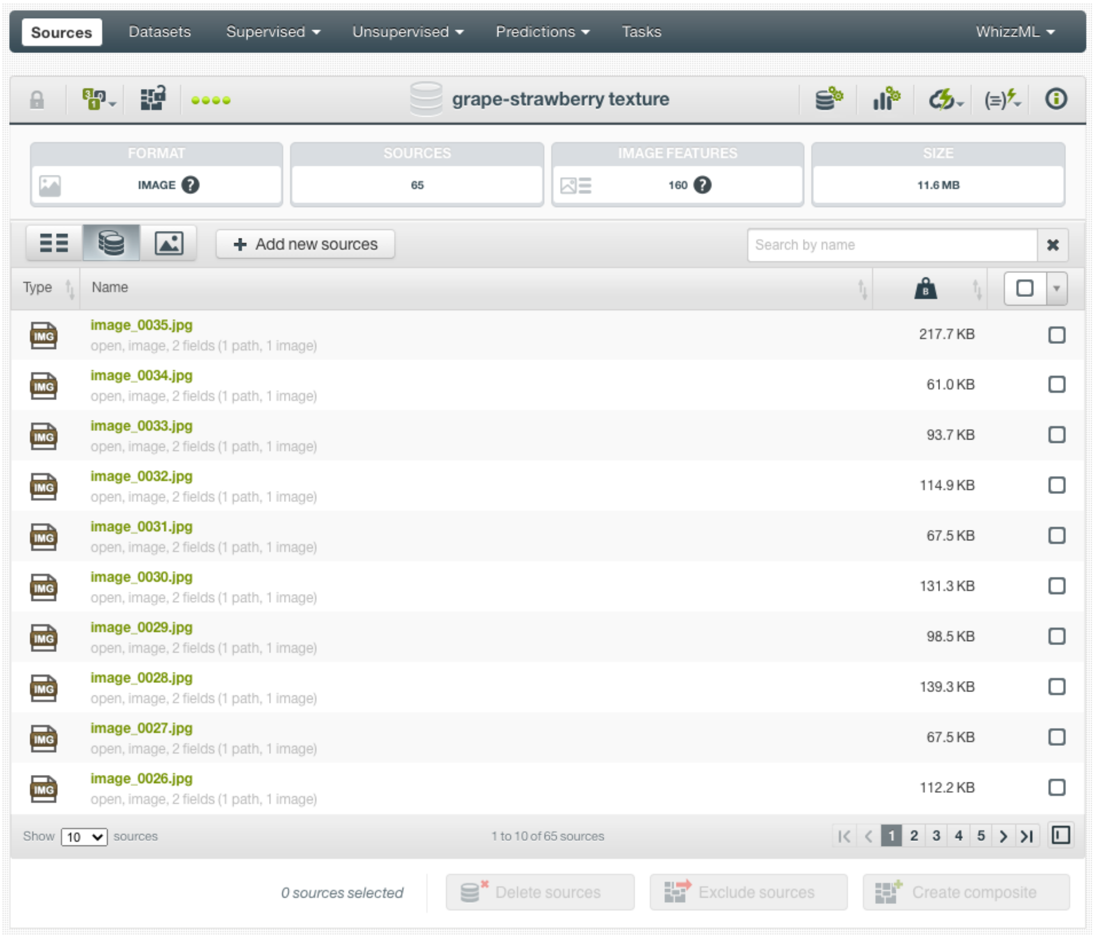


Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	5f2591e36abad710c6369105	5f2591e36abad710c6369106	5f2591e36abad710
filename	path	092_0001.jpg	092_0002.jpg	092_0003.jpg
image_id.full image: wavelet leve...	wavelet	5.298511787683826	9.053396291666704	6.756433587943893
image_id.full image: wavelet leve...	wavelet	34.22184914182231	173.50484683494517	73.05252926270371
image_id.full image: wavelet leve...	wavelet	3.2287418658088307	5.421344916666694	2.703500206440974
image_id.full image: wavelet leve...	wavelet	15.206367225812262	85.85787987002665	14.628296222676415

Figure 4.23: Tabs for switching views

4.3.3.2 Sources View

A **sources view** lists all the component sources in a composite source:



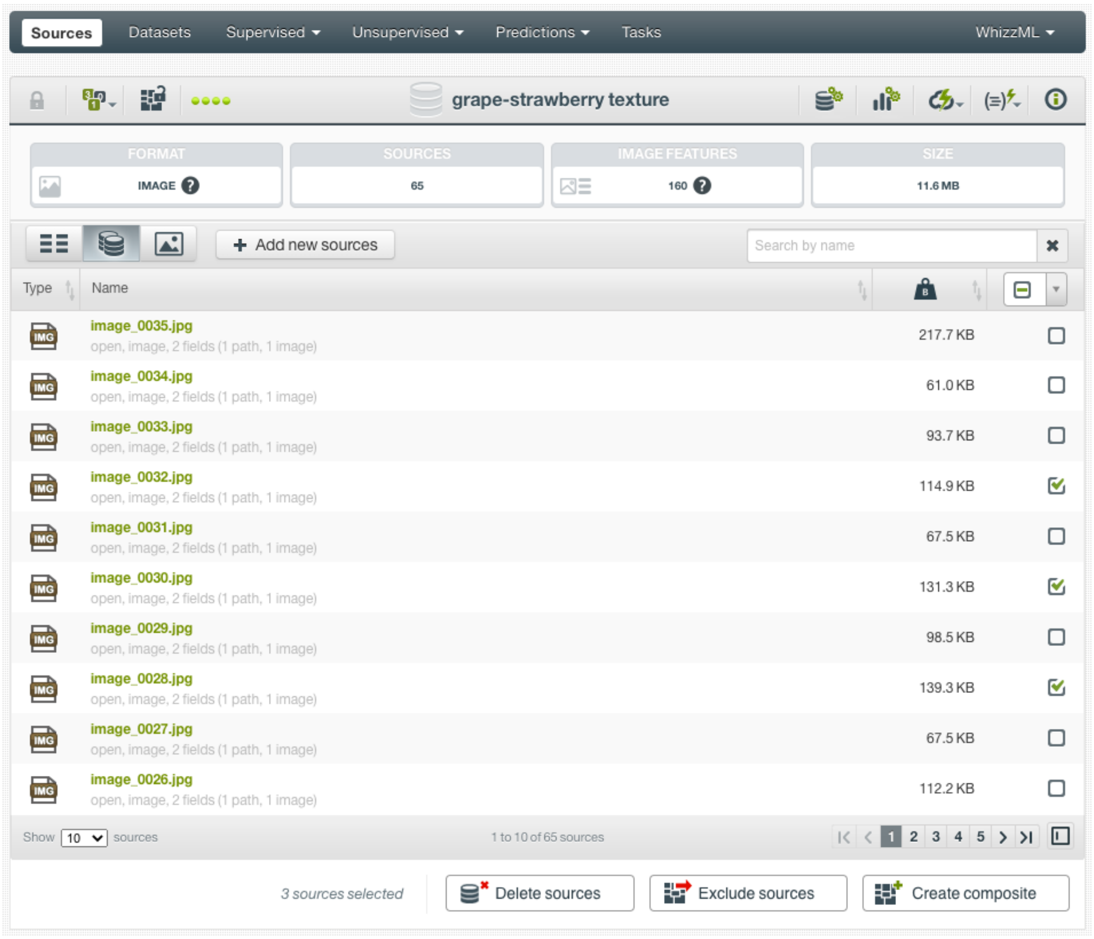
Type	Name		
IMG	image_0035.jpg open, image, 2 fields (1 path, 1 image)	217.7 KB	☐
IMG	image_0034.jpg open, image, 2 fields (1 path, 1 image)	61.0 KB	☐
IMG	image_0033.jpg open, image, 2 fields (1 path, 1 image)	93.7 KB	☐
IMG	image_0032.jpg open, image, 2 fields (1 path, 1 image)	114.9 KB	☐
IMG	image_0031.jpg open, image, 2 fields (1 path, 1 image)	67.5 KB	☐
IMG	image_0030.jpg open, image, 2 fields (1 path, 1 image)	131.3 KB	☐
IMG	image_0029.jpg open, image, 2 fields (1 path, 1 image)	98.5 KB	☐
IMG	image_0028.jpg open, image, 2 fields (1 path, 1 image)	139.3 KB	☐
IMG	image_0027.jpg open, image, 2 fields (1 path, 1 image)	67.5 KB	☐
IMG	image_0026.jpg open, image, 2 fields (1 path, 1 image)	112.2 KB	☐

Show 10 sources 1 to 10 of 65 sources 1 2 3 4 5

0 sources selected Delete sources Exclude sources Create composite

Figure 4.24: Sources view of an image composite source

It's essentially a source list view inside a composite source. Users can click on each component source to view its details. Besides viewing the information of component sources, users can select them to perform the follow operations in an **open** composite source:



Type	Name	Size	
IMG	image_0035.jpg open, image, 2 fields (1 path, 1 image)	217.7 KB	☐
IMG	image_0034.jpg open, image, 2 fields (1 path, 1 image)	61.0 KB	☐
IMG	image_0033.jpg open, image, 2 fields (1 path, 1 image)	93.7 KB	☐
IMG	image_0032.jpg open, image, 2 fields (1 path, 1 image)	114.9 KB	✓
IMG	image_0031.jpg open, image, 2 fields (1 path, 1 image)	67.5 KB	☐
IMG	image_0030.jpg open, image, 2 fields (1 path, 1 image)	131.3 KB	✓
IMG	image_0029.jpg open, image, 2 fields (1 path, 1 image)	98.5 KB	☐
IMG	image_0028.jpg open, image, 2 fields (1 path, 1 image)	139.3 KB	✓
IMG	image_0027.jpg open, image, 2 fields (1 path, 1 image)	67.5 KB	☐
IMG	image_0026.jpg open, image, 2 fields (1 path, 1 image)	112.2 KB	☐

Show 10 sources   1 to 10 of 65 sources   1 2 3 4 5

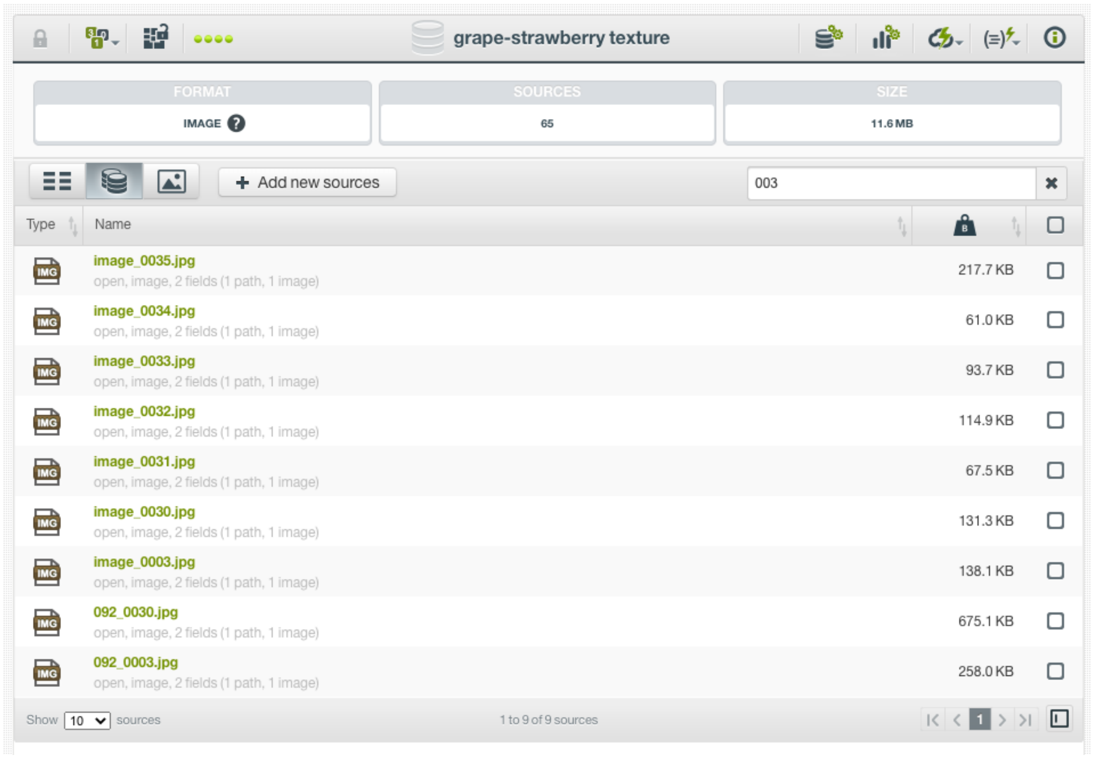
3 sources selected

Figure 4.25: Selecting component sources of a composite source

- **Delete sources**: Delete the selected component sources. This will not only remove the component sources from the composite source, but also delete them from the platform permanently.
- **Exclude sources**: Exclude the selected component sources. Thie will exclude them from the composite source, but they will stay as invididual sources on the platform and won't be deleted.
- **Create composite**: Create a new composite source using the selected sources as its component sources.

For a **closed** composite source, users can only select component sources to perform Create composite.

When making selections, users can use the "Select all sources" checkbox on the top right to sell all component sources. They can also use the "Search by name" box which acts as a name filter. That is, when a text string is typed into the box, all component sources whose name contain the string are shown and are for selection.



Name	Size
image_0035.jpg open, image, 2 fields (1 path, 1 image)	217.7 KB
image_0034.jpg open, image, 2 fields (1 path, 1 image)	61.0 KB
image_0033.jpg open, image, 2 fields (1 path, 1 image)	93.7 KB
image_0032.jpg open, image, 2 fields (1 path, 1 image)	114.9 KB
image_0031.jpg open, image, 2 fields (1 path, 1 image)	67.5 KB
image_0030.jpg open, image, 2 fields (1 path, 1 image)	131.3 KB
image_0003.jpg open, image, 2 fields (1 path, 1 image)	138.1 KB
092_0030.jpg open, image, 2 fields (1 path, 1 image)	675.1 KB
092_0003.jpg open, image, 2 fields (1 path, 1 image)	258.0 KB

Show 10 sources
1 to 9 of 9 sources

Figure 4.26: Filtering component sources by using name search

4.3.3.3 Images View

An **images view** shows all the images in the composite source.

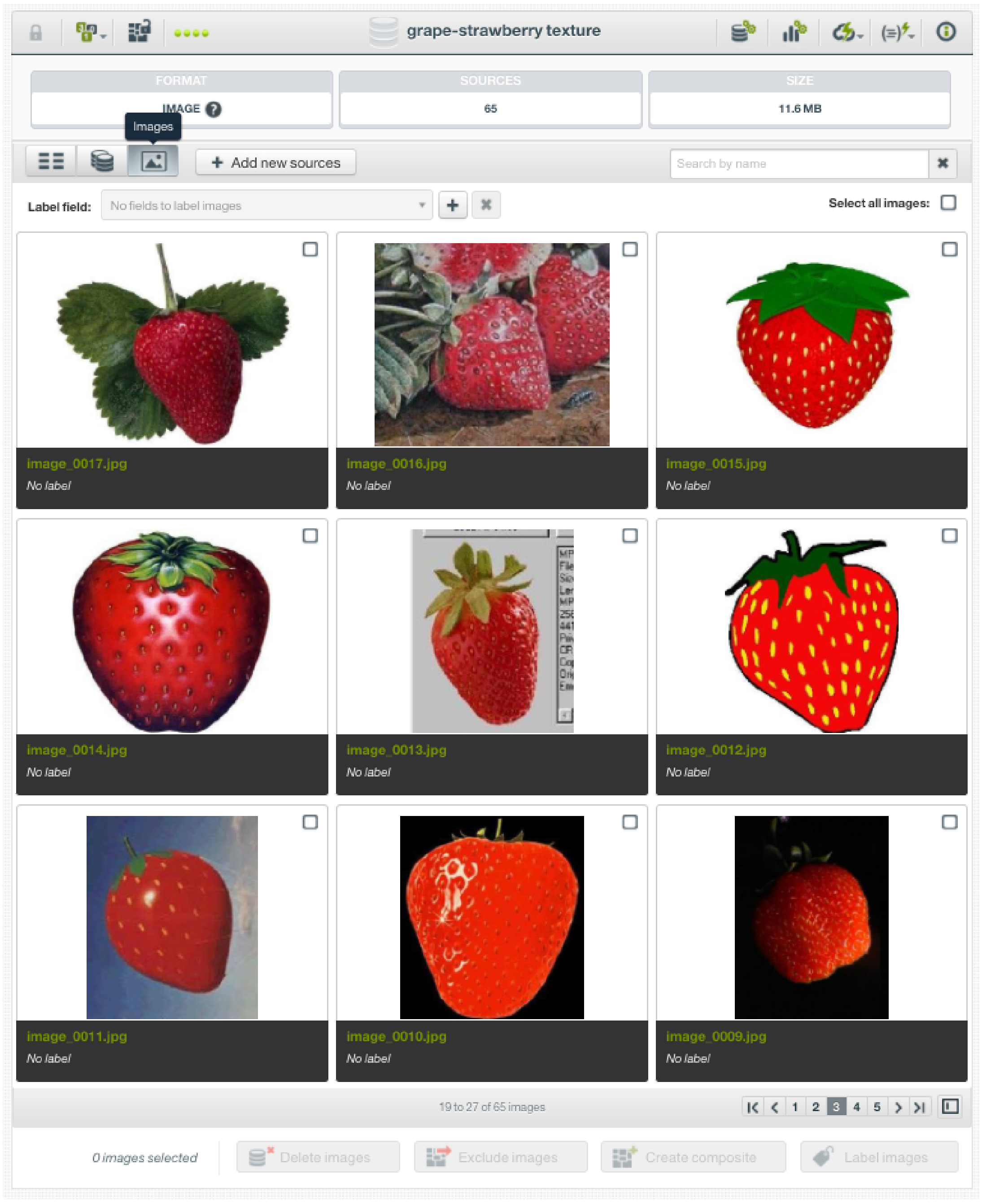


Figure 4.27: Images view of an image composite source

For a **closed** image composite source, users can only view images, and they can load new images by clicking on the Get new preview button on the bottom right.

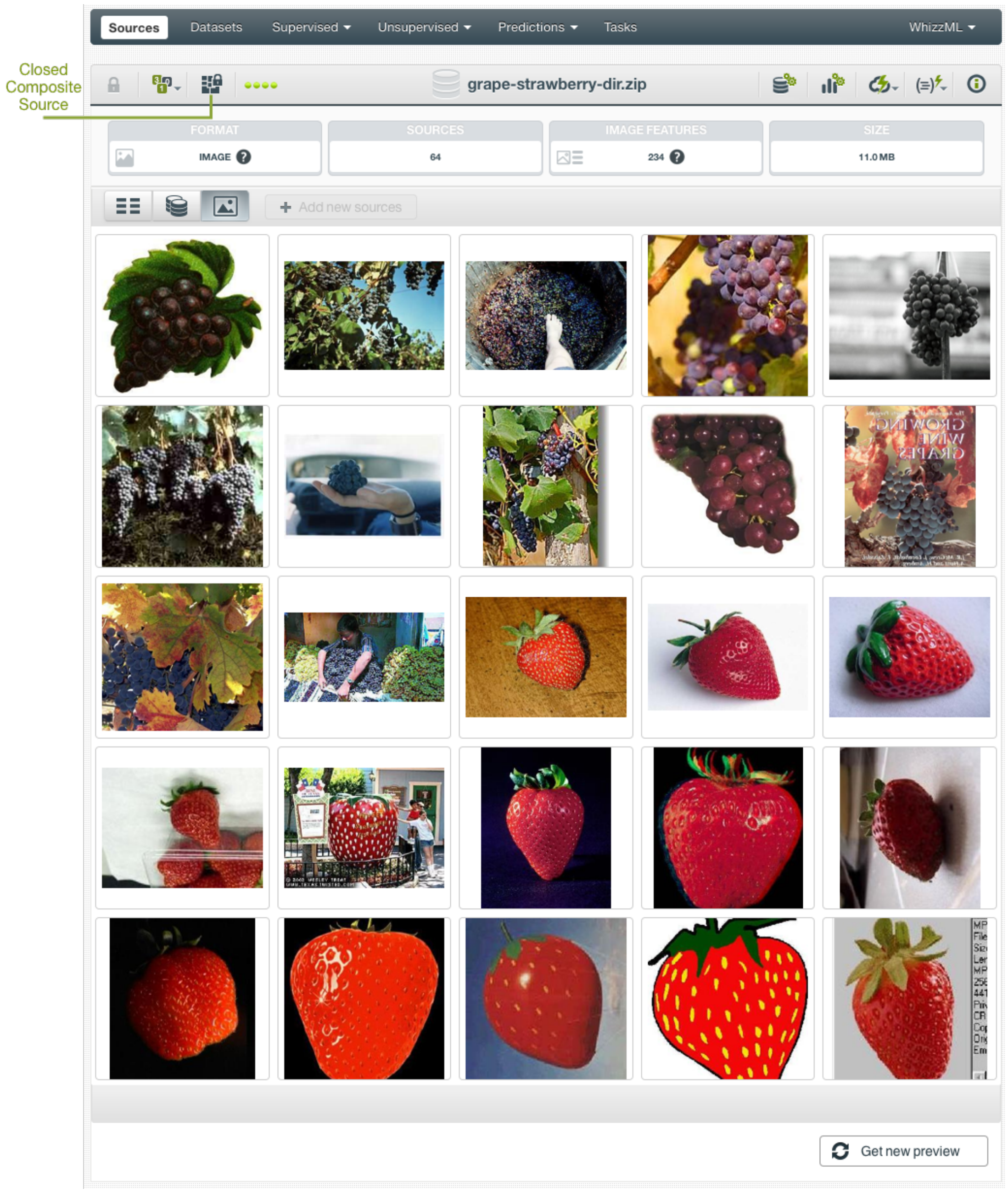


Figure 4.28: Images view of a closed image composite source

In the **images view** of an **open** image composite source, besides viewing all images, users can select them to perform certain operations. When making selections, users can use the “Select all images” checkbox on the top right to sell all images. They can also use the “Search by name” box which acts as a name filter. That is, when a text string is typed into the box, all images whose name contain the string are shown and are for selection.

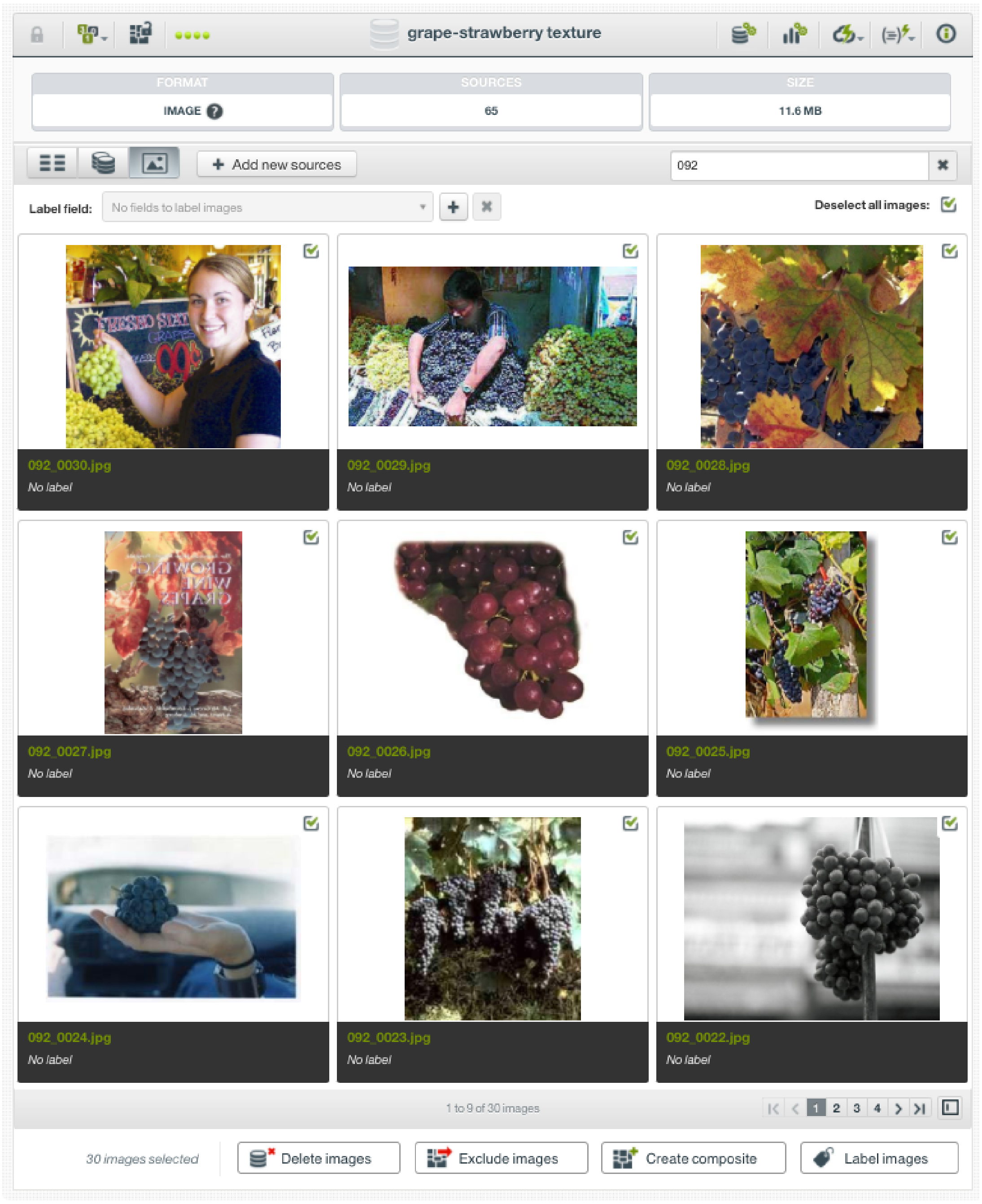
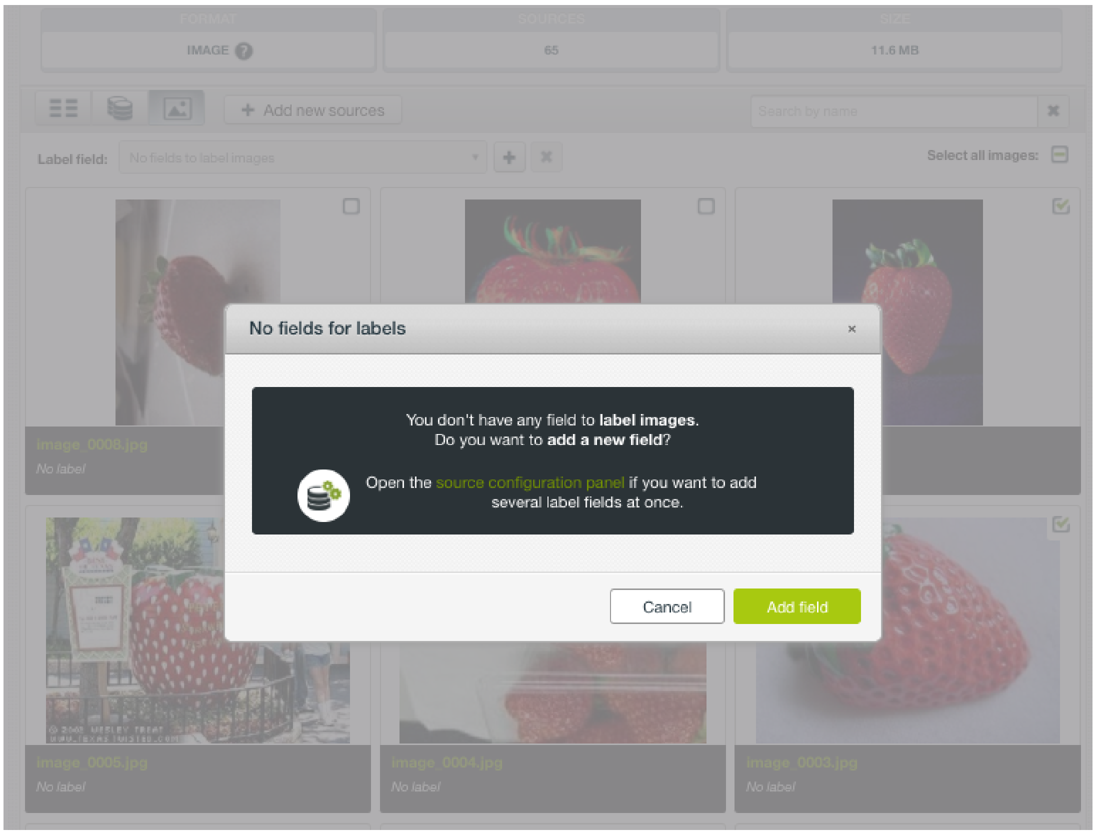


Figure 4.29: Filtering images by using name search

After the selection of images, the following operations can be performed to them:

- **Delete images**: Delete the selected images. This will not only remove the images from the composite source, but also delete them from the platform permanently.
- **Exclude images**: Exclude the selected imagess. Thie will exclude them from the composite source, but they will stay as invididual image sources on the platform and won't be deleted.
- **Create composite**: Create a new composite source using the selected images as its component sources.
- **Label images**: Give a label to the selected images.

When labeling images, there has to be a label field. When such field doesn't exist, there will be a prompt to create one:



No fields for labels

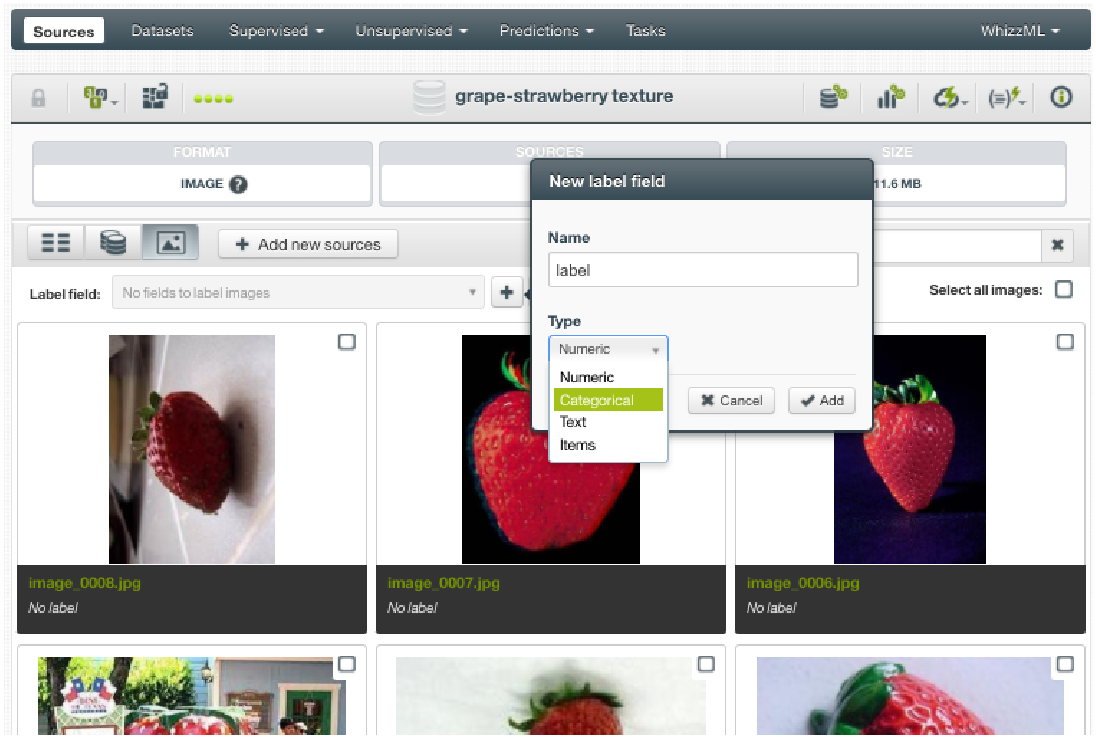
You don't have any field to **label images**.
Do you want to **add a new field**?

Open the source configuration panel if you want to add several label fields at once.

Figure 4.30: Prompt if no field for labels

There are two ways to create a label field.

- First, on top of the images, on the left is a "Label field" textbox. Clicking on the "+" next to it, users will be prompted with a dialog box asking for the field name and the field type:



New label field

Name: label

Type: Numeric

Figure 4.31: Create a new label

After inputing the field name and selecting its type from the dropdown, click the Add button to create the label field.

When there are one or more labels in the composite source, clicking on the “x” next to the “Label field” textbox will remove them.

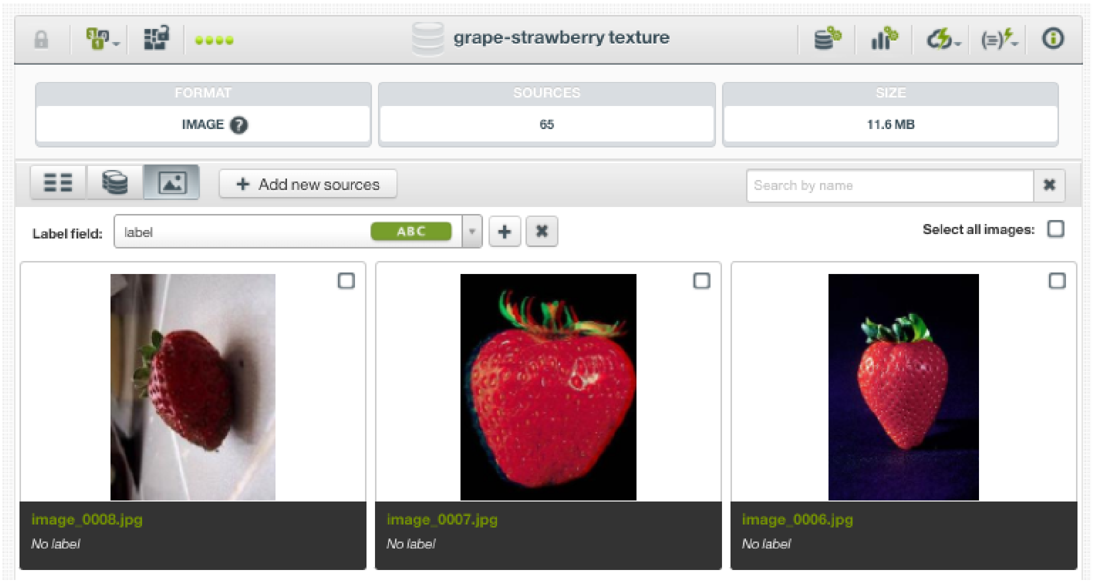
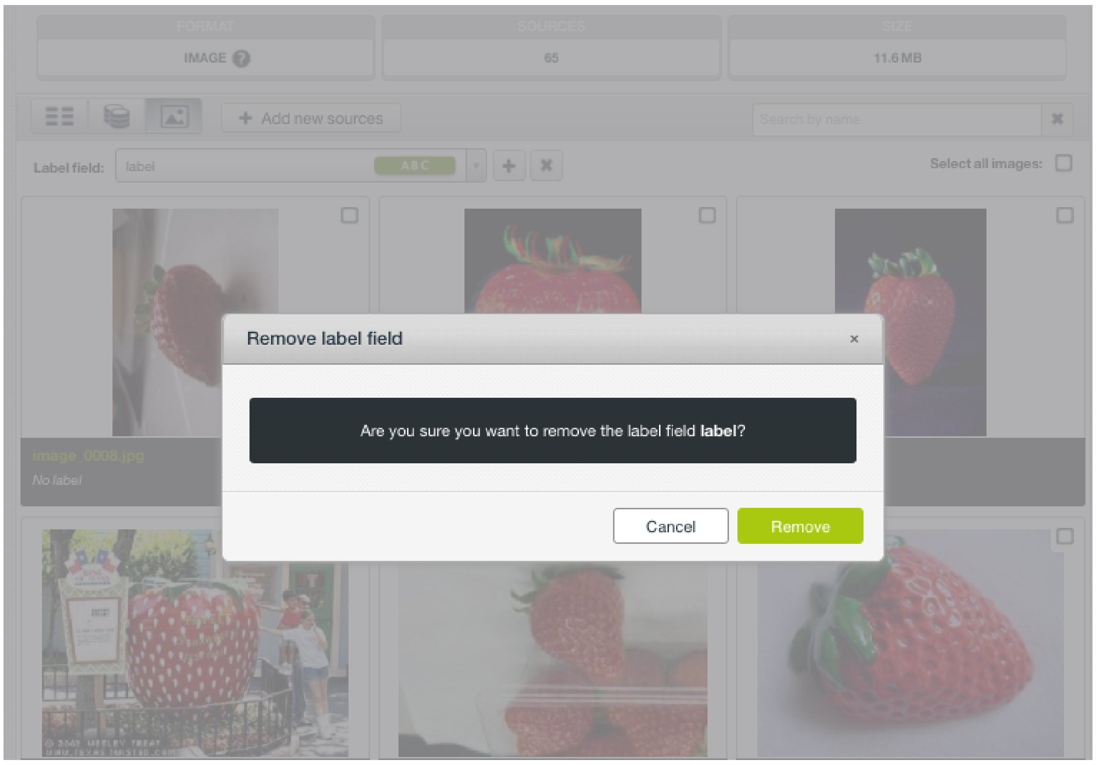


Figure 4.32: Remove a label

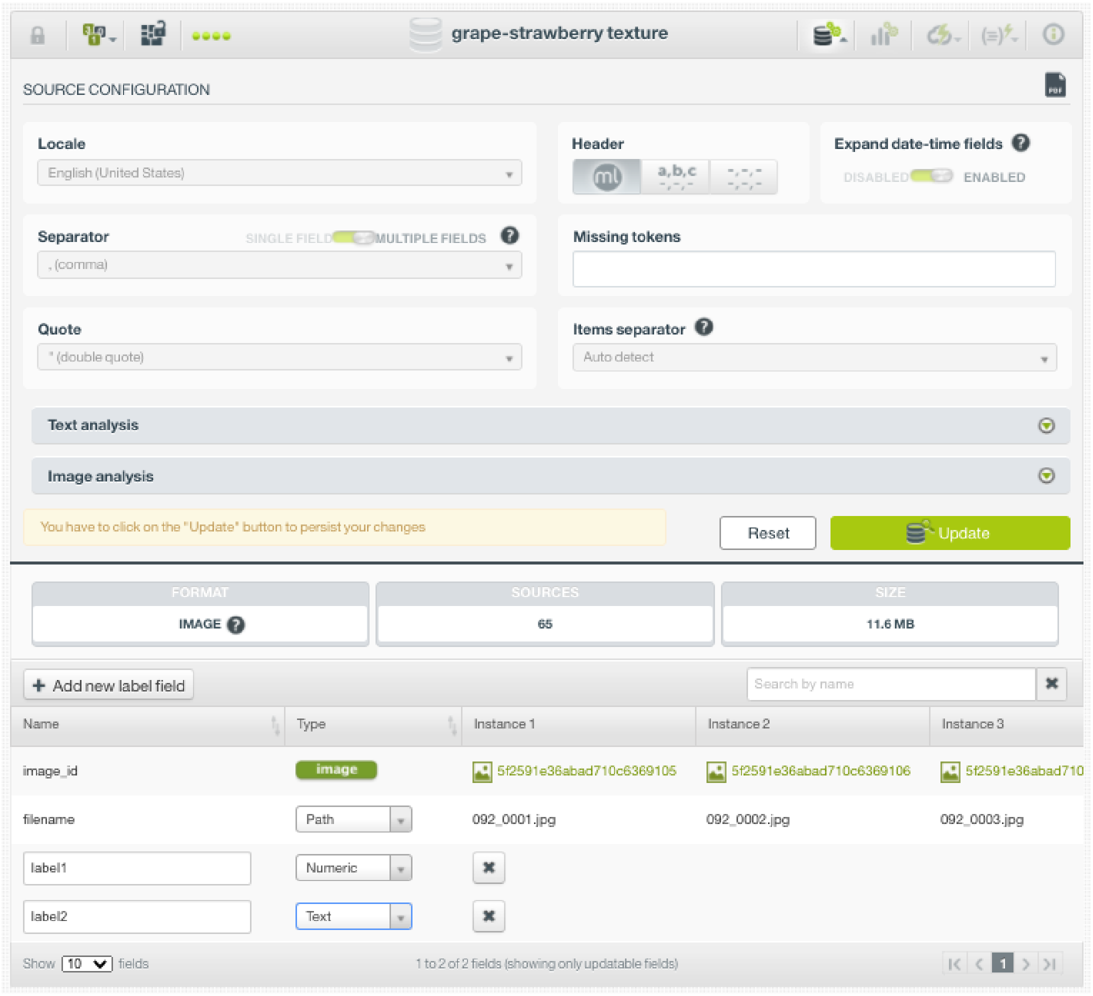
Users will get a modal box to confirm the removal:



Are you sure you want to remove the label field **label**?

Figure 4.33: Confirm to remove a label

- The second way to create a label field is, by following the link in the prompt Figure 4.30, using the Source Configuration Panel of the composite source.


grape-strawberry texture

SOURCE CONFIGURATION

Locale: English (United States)

Header: ml | a,b,c

Expand date-time fields: DISABLED / ENABLED

Separator (SINGLE FIELD / MULTIPLE FIELDS): , (comma)

Missing tokens:

Quote: " (double quote)

Items separator: Auto detect

Text analysis

Image analysis

You have to click on the "Update" button to persist your changes

Reset | Update

FORMAT	SOURCES	SIZE
IMAGE	65	11.6 MB

Add new label field | Search by name

Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	5f2591e36abad710c6369105	5f2591e36abad710c6369106	5f2591e36abad710
filename	Path	092_0001.jpg	092_0002.jpg	092_0003.jpg
label1	Numeric	✖		
label2	Text	✖		

Show 10 fields 1 to 2 of 2 fields (showing only updatable fields)

Figure 4.34: Adding a field in source configuration

As shown above, there is an Add new label field button above the field list. Clicking on it will add a new field at the bottom of the field list. This is especially useful in creating multiple labels. After entering filed names and selection field types, users have to click on the Update button to save the changes.

After a label field is created, users can go to the **images view**, select images and label them by giving the label field a value, such as “grape” shown below:

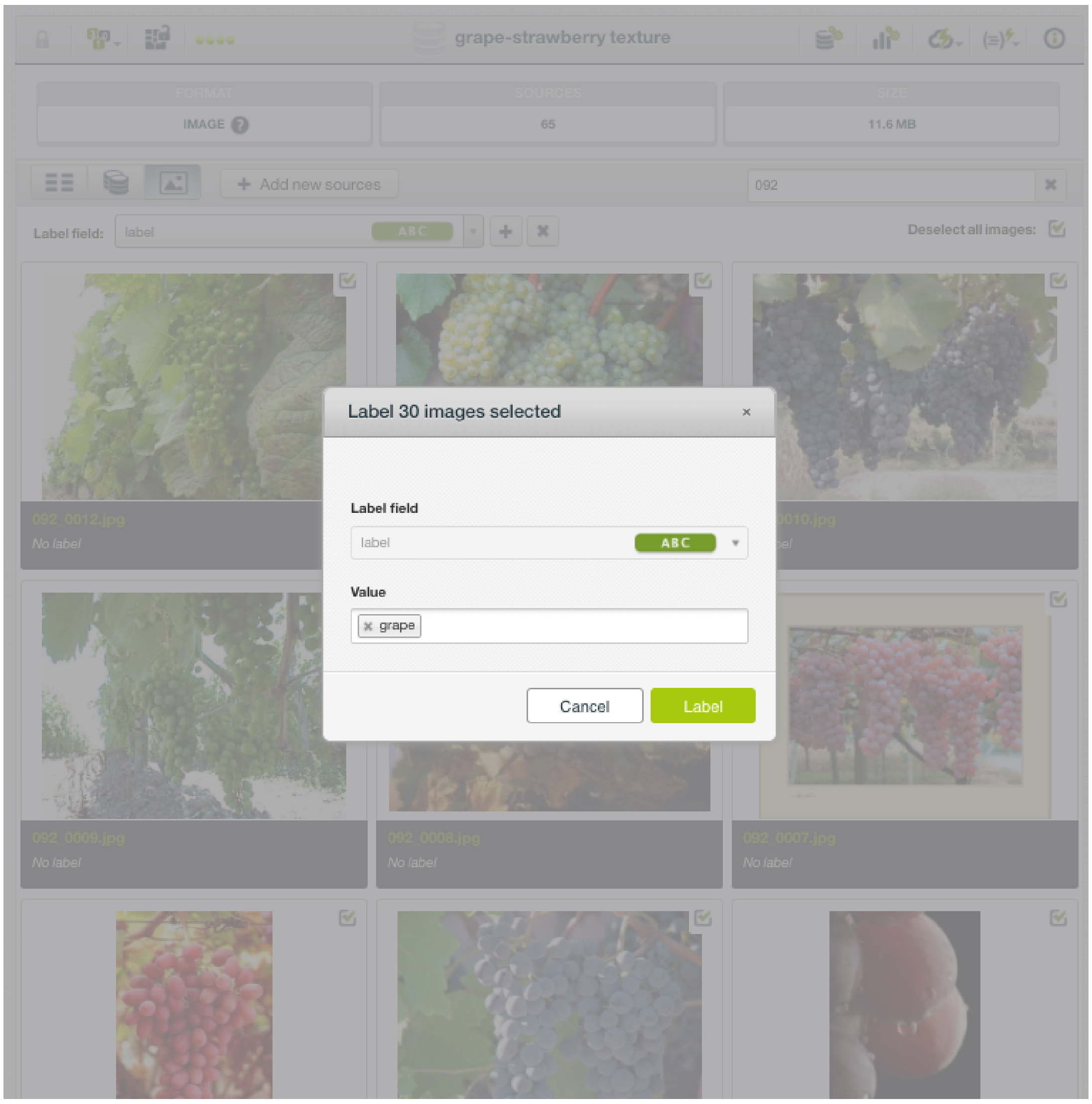


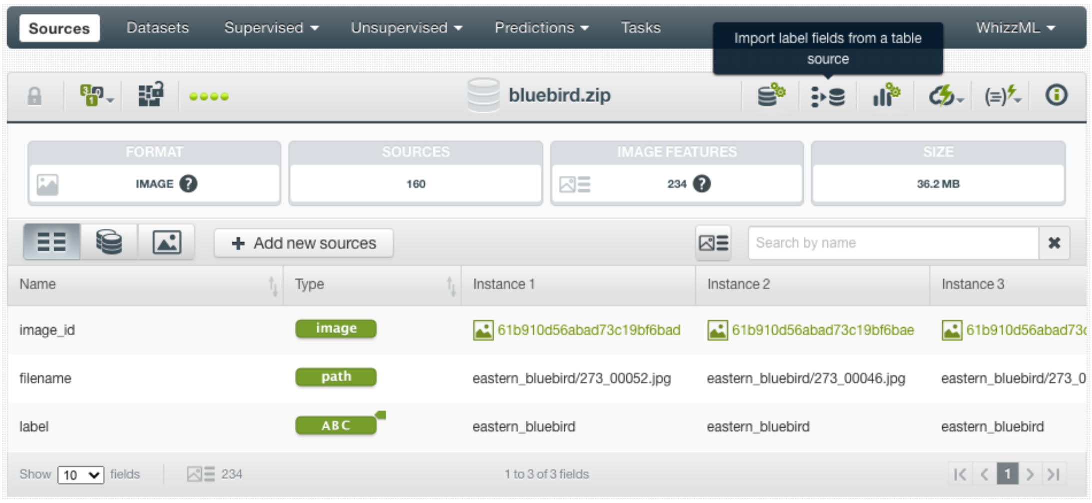
Figure 4.35: Labeling images

4.3.4 Importing Label Fields to Image Composites

Label fields can be added to image composites by importing them from CSV or JSON files. This is very useful as sometimes it's easier to enter labels in a different file or the labels are prepared separately from images in the business workflow. Moreover, additional label fields can provide more information about the images, such as captions, comments, geo-coordinates, which can be in the CSV or JSON files.

What CSV or JSON files do in such context is to provide information about the images in a table format, so here we call these files table files.

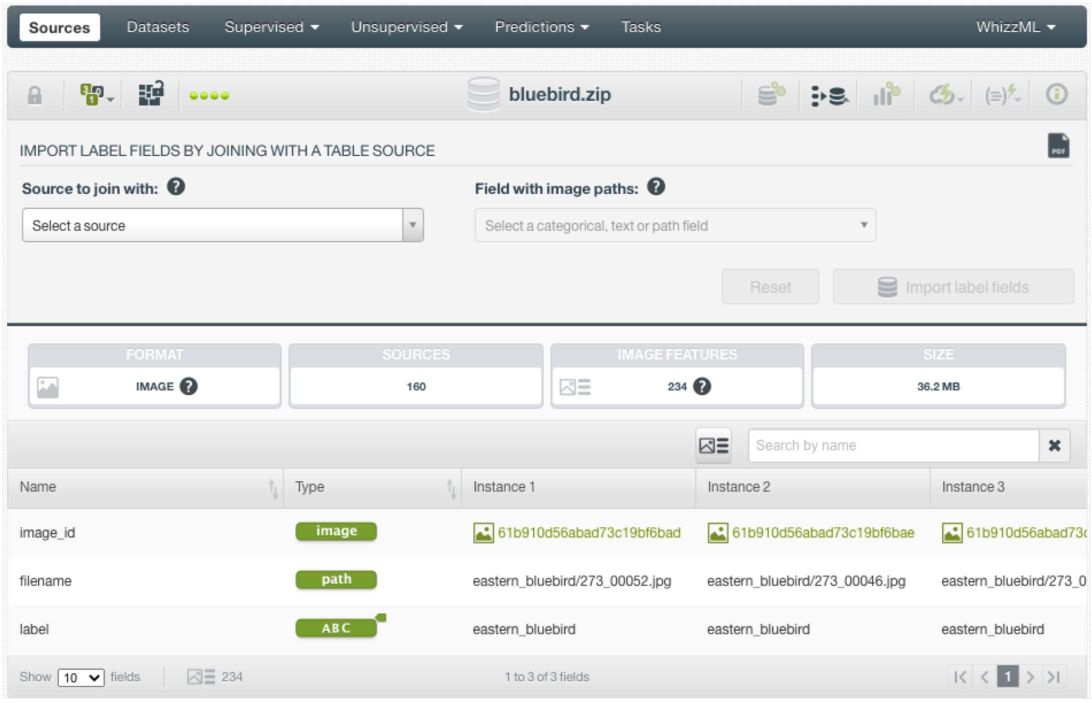
Importing label fields is basically a join operation, which combines rows from the image source and the table source. The join is based on a related field between the two sources, and the related field is the image filename or path.



Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	61b910d56abad73c19bf6bad	61b910d56abad73c19bf6bae	61b910d56abad73c
filename	path	eastern_bluebird/273_00052.jpg	eastern_bluebird/273_00046.jpg	eastern_bluebird/273_0
label	ABC	eastern_bluebird	eastern_bluebird	eastern_bluebird

Figure 4.36: The menu icon for importing label fields

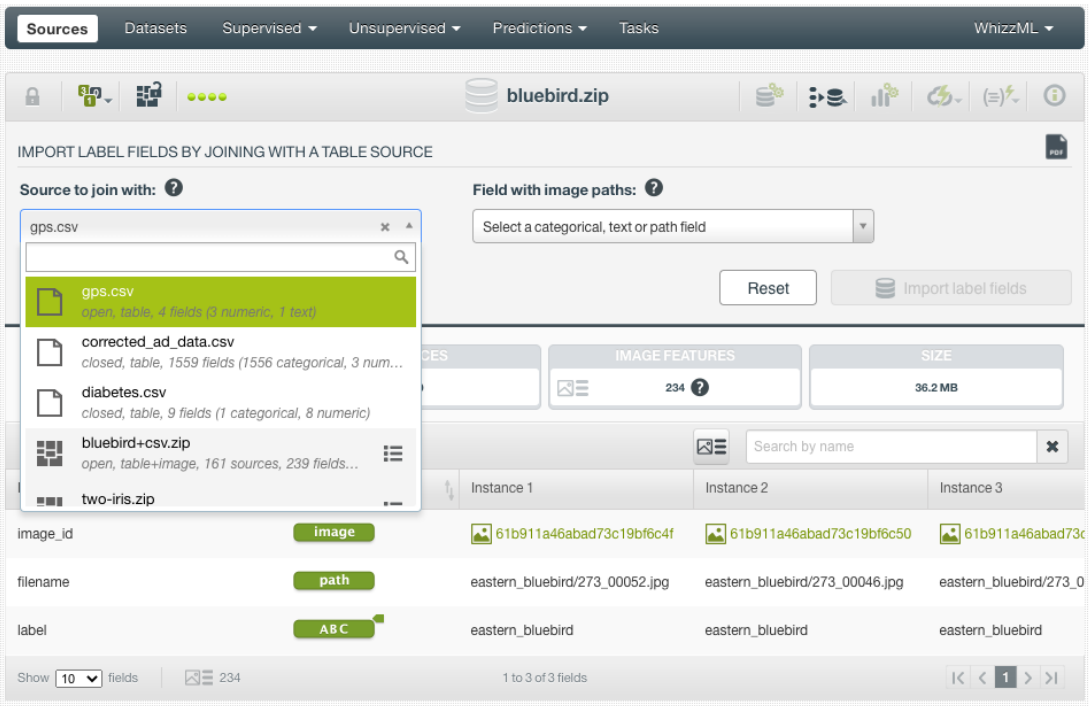
From an image composite source, click on the IMPORT LABEL FIELDS FROM A TABLE SOURCE menu icon, as shown in Figure 4.36. The icon is composed of three rows, an arrow and a source, signifying a table adding to a source.



Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	61b910d56abad73c19bf6bad	61b910d56abad73c19bf6bae	61b910d56abad73c
filename	path	eastern_bluebird/273_00052.jpg	eastern_bluebird/273_00046.jpg	eastern_bluebird/273_0
label	ABC	eastern_bluebird	eastern_bluebird	eastern_bluebird

Figure 4.37: Import label fields from a table source

Then users are presented with the input panel, as show in Figure 4.37. This panel is for selecting the table source and the image path field.



IMPORT LABEL FIELDS BY JOINING WITH A TABLE SOURCE

Source to join with: gps.csv

- gps.csv — open, table, 4 fields (3 numeric, 1 text)
- corrected_ad_data.csv — closed, table, 1559 fields (1556 categorical, 3 num…
- diabetes.csv — closed, table, 9 fields (1 categorical, 8 numeric)
- bluebird+csv.zip — open, table+image, 161 sources, 239 fields…
- two-iris.zip

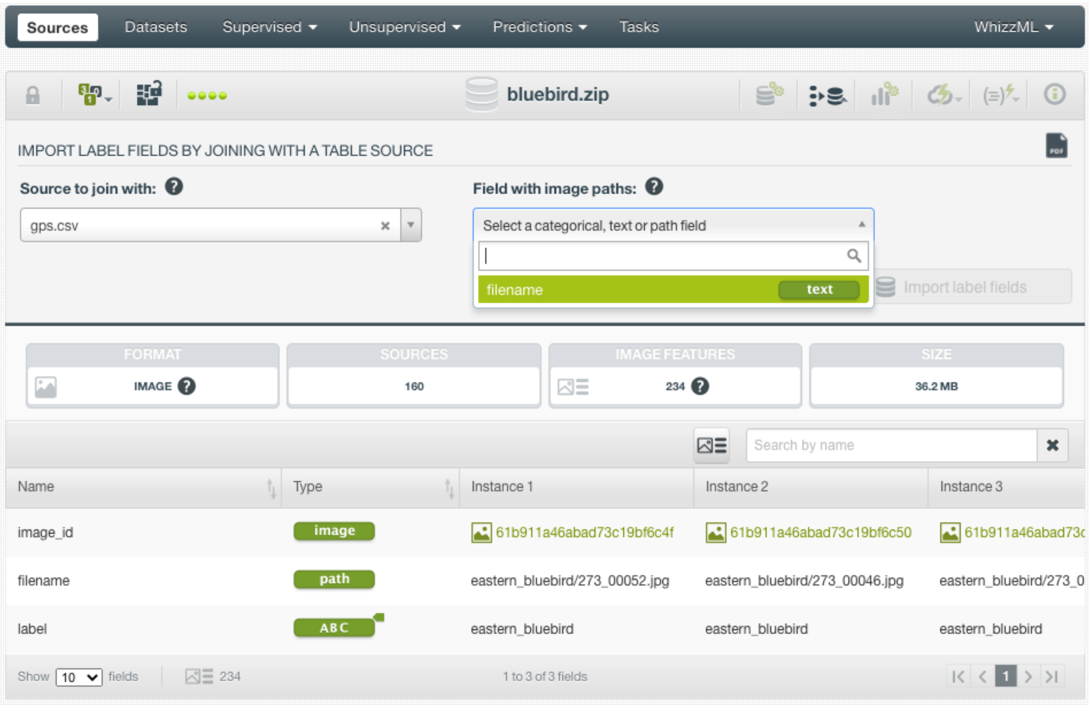
Field with image paths: Select a categorical, text or path field

IMAGE FEATURES: 234 | SIZE: 36.2 MB

Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	61b911a46abad73c19bf6c4f	61b911a46abad73c19bf6c50	61b911a46abad73c
filename	path	eastern_bluebird/273_00052.jpg	eastern_bluebird/273_00046.jpg	eastern_bluebird/273_0
label	ABC	eastern_bluebird	eastern_bluebird	eastern_bluebird

Figure 4.38: Select the table source

As shown in Figure 4.38, clicking on the “Select a source” input box will bring up a dropdown list, which are all table sources available. Select the desired one from the list.



IMPORT LABEL FIELDS BY JOINING WITH A TABLE SOURCE

Source to join with: gps.csv

Field with image paths: Select a categorical, text or path field

- filename — text

FORMAT: IMAGE | SOURCES: 160 | IMAGE FEATURES: 234 | SIZE: 36.2 MB

Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	61b911a46abad73c19bf6c4f	61b911a46abad73c19bf6c50	61b911a46abad73c
filename	path	eastern_bluebird/273_00052.jpg	eastern_bluebird/273_00046.jpg	eastern_bluebird/273_0
label	ABC	eastern_bluebird	eastern_bluebird	eastern_bluebird

Figure 4.39: Select the path field

Then select the path field in the table source that corresponds to the image path field in the current image composite source. Clicking on the input box will show the candiate field, after the selected table source was parsed.

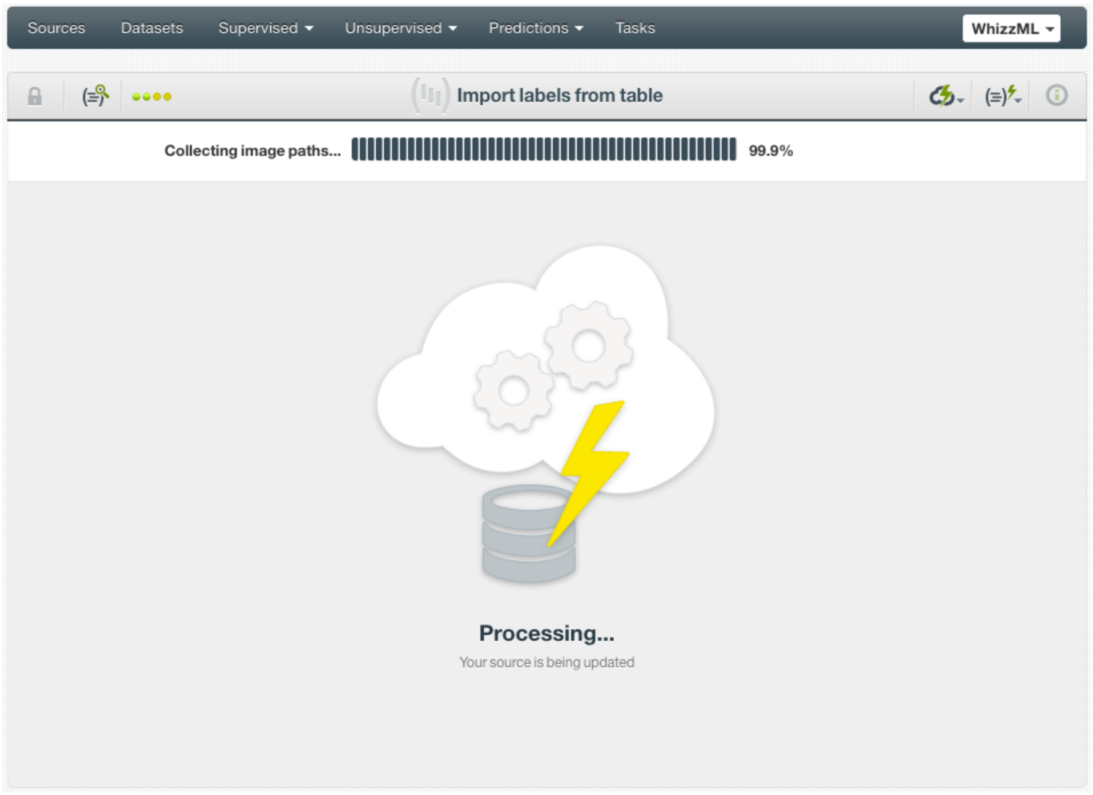
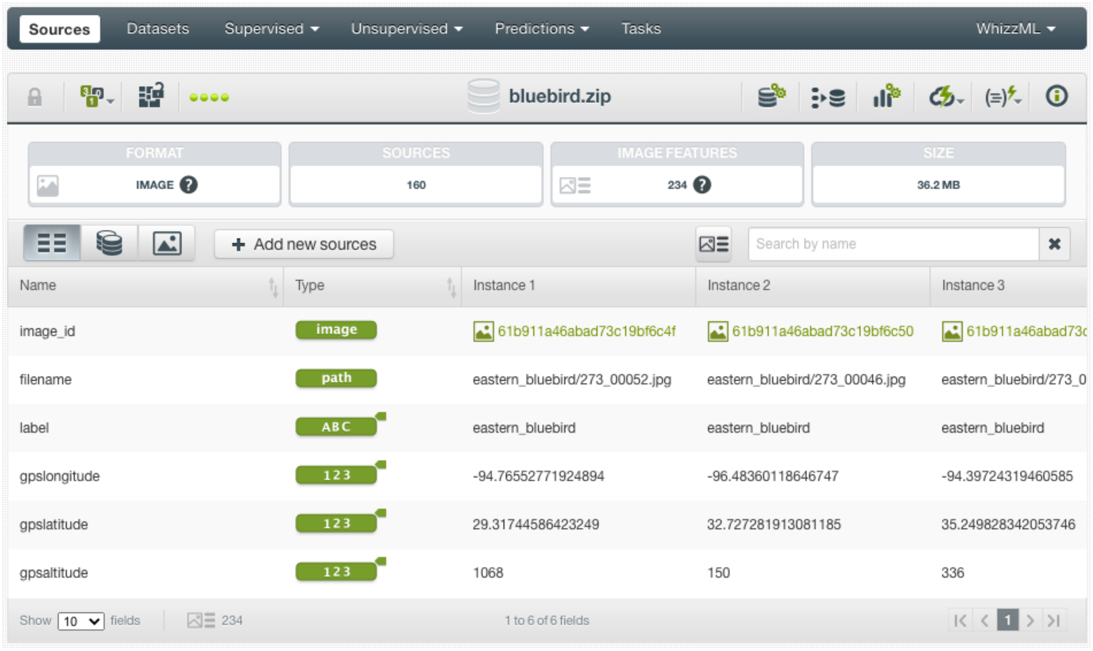


Figure 4.40: Progress of the importing

After both selections were made, click on the Import label fields button to start the import. During the operation, there will be a progress bar along with relevant information (Figure 4.40).



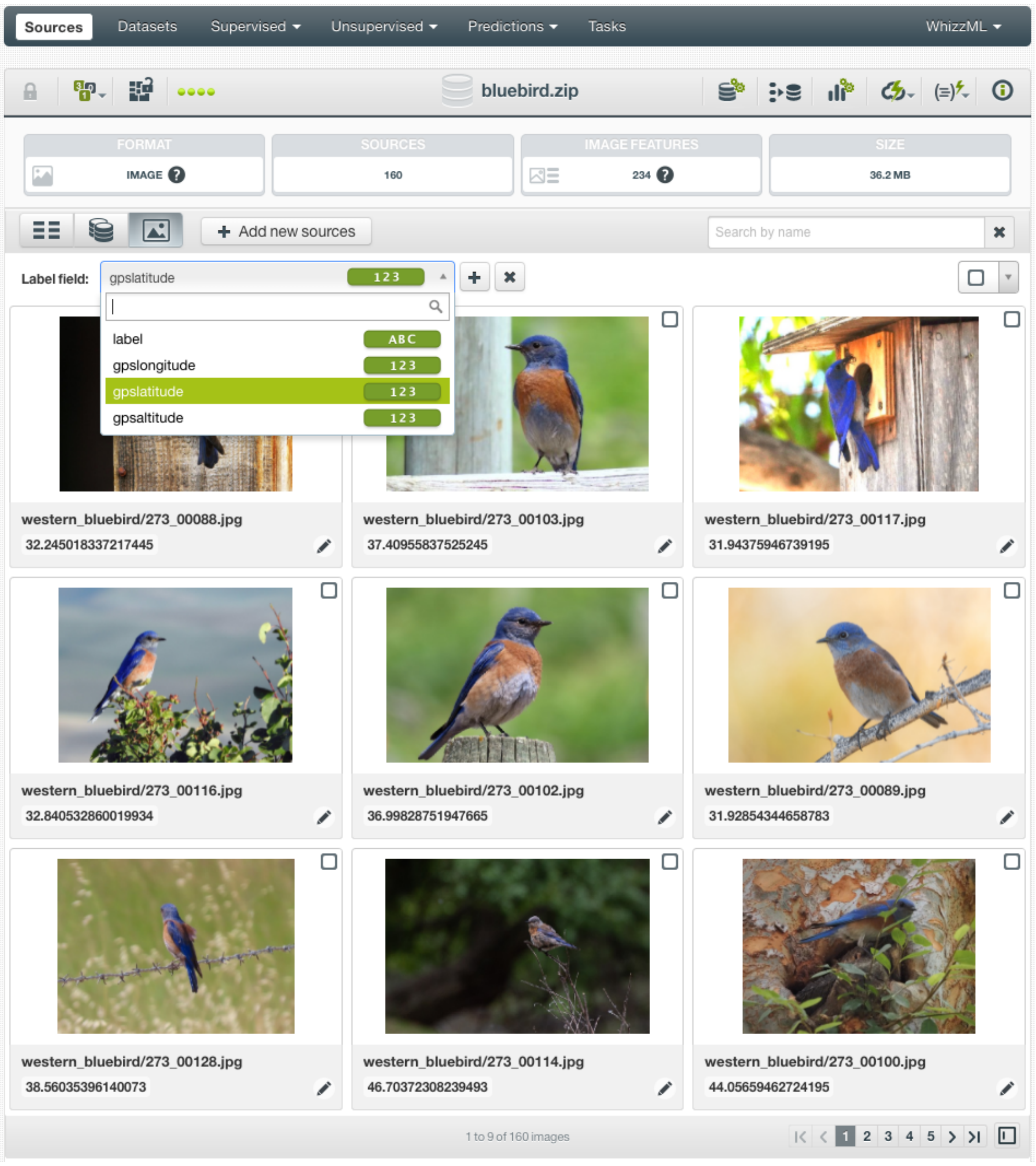
Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	61b911a46abad73c19bf6c4f	61b911a46abad73c19bf6c50	61b911a46abad73c
filename	path	eastern_bluebird/273_00052.jpg	eastern_bluebird/273_00046.jpg	eastern_bluebird/273_0
label	ABC	eastern_bluebird	eastern_bluebird	eastern_bluebird
gpslongitude	123	-94.76552771924894	-96.48360118646747	-94.39724319460585
gpslatitude	123	29.31744586423249	32.727281913081185	35.249828342053746
gpsaltitude	123	1068	150	336

Figure 4.41: Newly imported label fields

After the operation is finished, the new label fields are shown in the **Fields view** of the composite source (Figure 4.41).

Note: if the current image composite source is closed, importing label fields to it from a table source will create a new open image composite source, which will have the newly imported label fields. If the current image composite source is open, the importing operation will add the newly imported label fields to the image composite source and keep it open.

After the new label fields are imported, users can inspect them in the **Images view** of the composite source. As shown in Figure 4.42, different label fields can be selected by using the dropdown list of the “Label field” on top. They will show up in the image captions.



western_bluebird/273_00088.jpg
32.245018337217445

western_bluebird/273_00103.jpg
37.40955837525245

western_bluebird/273_00117.jpg
31.94375946739195

western_bluebird/273_00116.jpg
32.840532860019934

western_bluebird/273_00102.jpg
36.99828751947665

western_bluebird/273_00089.jpg
31.92854344658783

western_bluebird/273_00128.jpg
38.56035396140073

western_bluebird/273_00114.jpg
46.70372308239493

western_bluebird/273_00100.jpg
44.05659462724195

Figure 4.42: Selecting different label fields to view

4.4 Table+Image Composite Sources

Machine learning with images oftentimes requires labels of the images, especially in the applications of image classification. In many scenarios, images and their labels were prepared separately, so they appear in separate files. For instance, there is a collection of images while their labels is in a CSV or JSON file.

Besides labels, CSV or JSON files can provide other information of the images, such as captions, comments, geo-coordinates, etc.

What CSV or JSON files do in such context is to provide information about the images in a table format, so here we call these files table files.

To accommodate this common practice of using a separate table file for image labels and other information, BigML provides two solutions. The first one is that users can upload the images and the table file separately. Then users can import all label fields in the table file to the image composite source. This is covered in Subsection 4.3.4.

Another solution is by using composite sources of format “Table+Image”. As implied by the name of the format, there are two parts in the data. One is a collection of images, another a table file that is a CSV or JSON file. The ultimate goal of such format is to create datasets that include the images and the fields of the CSV. In the case when a JSON is used, it is to create datasets that include the images and the lists or dictionaries of the JSON.

By using “Table+Image” composite sources, users don’t have to upload the images and the table file separately. They don’t need to preform the importing operation, and they can create datasets from “Table+Image” composite sources.

As described in Section 4.2, a composite source can be created by uploading an archived collection of files. When the archive contains a list of images and a table file, the resulting composite will have the format “Table+Image”.

The components of the composite source created from images and a table file will have different fields, with the image sources having only image fields while the CSV or JSON source having other fields including labels.

Strictly speaking, such “Table+Image” composite source is heterogeneous, which means not all component sources have uniformly the same fields, hence it could be of the “mixed” format (Section 2.2). Instead, BigML recognizes that this is a table plus images, with the CSV or JSON providing tabular data, and essentially setting the fields of the composite source to those of the CSV or JSON. Additionally, attached to each row are the auto-generated fields of the extracted features from the images (Subsection 4.3.2).

The component source from the CSV or JSON in the “Table+Image” composite source is also called the table component. It is expected that one or more columns of the table component refer to an image. Those columns will become fields in the composite source and have the optype *path*, which contain the (relative) file name of the corresponding image, as extracted from the zip file index. BigML tries to discover which fields in the table component refer to images using the following heuristics:

- The field is named “file”, “filename”, “file name”, “path” or “image”, possibly by punctuation (/, -, _, or blank) and a number (e.g. “path 3”, “image/2”).
- The preview of the field contains values also found in the preview of the filenames extracted from the images.

In the rare cases of BigML not recognizing the path field properly, users can go to the source configuration, and update the optype of the intended field to *path*.

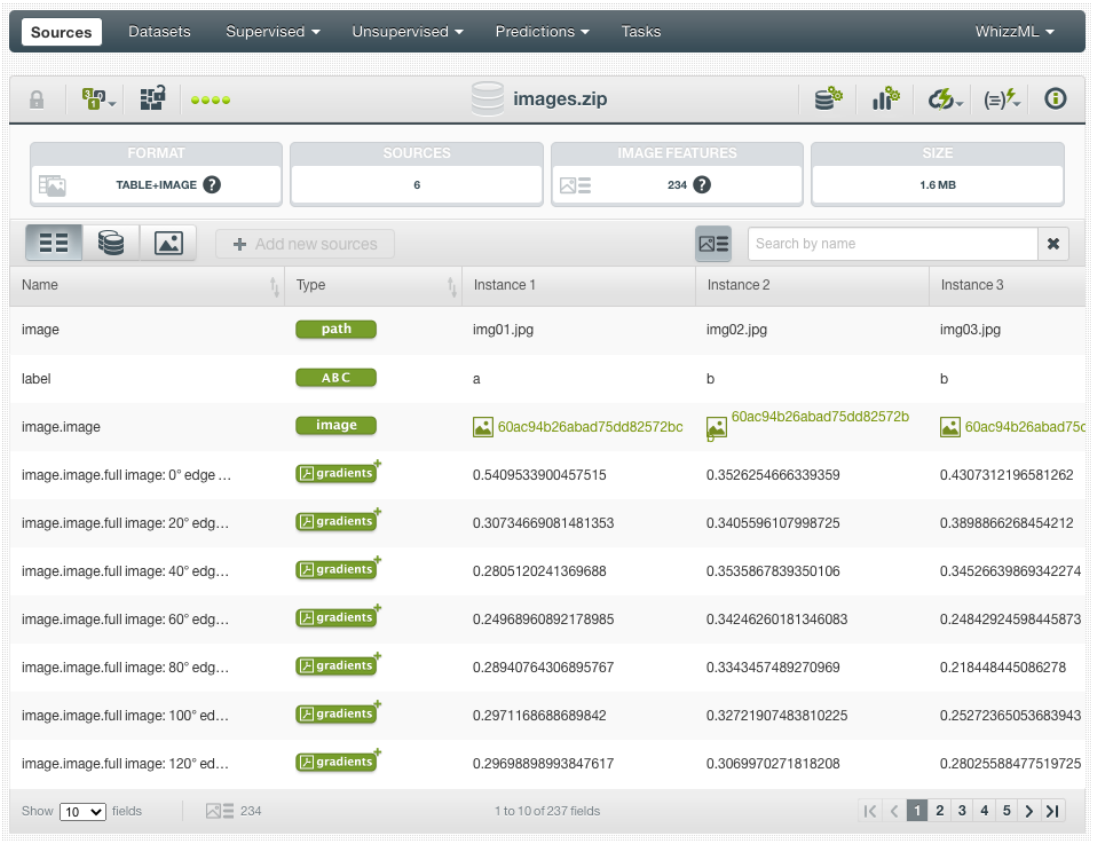
Here is a simple example, which is a zip file containing 6 files:

```
Archive:  images.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
       78  08-18-2020 17:07   label.csv
    15980  08-18-2020 17:03   img03.jpg
    51886  08-18-2020 17:03   img02.jpg
    38361  08-18-2020 17:03   img01.jpg
    17700  08-18-2020 17:03   img05.jpg
    55856  08-18-2020 17:03   img04.jpg
---------                     -------
   179861                     6 files
```

There are 5 images file and a CSV which looks like:

```
image, label
img01.jpg, a
img02.jpg, b
img03.jpg, b
img04.jpg, c
img05.jpg, a
```

Once the zip file is uploaded to the BigML Dashboard, a “Table+Image” composite source is created:



FORMAT	SOURCES	IMAGE FEATURES	SIZE
TABLE+IMAGE	6	234	1.6 MB

Name	Type	Instance 1	Instance 2	Instance 3
image	path	img01.jpg	img02.jpg	img03.jpg
label	ABC	a	b	b
image.image	image	60ac94b26abad75dd82572bc	60ac94b26abad75dd82572b	60ac94b26abad75c
image.image.full image: 0° edge …	gradients	0.5409533900457515	0.3526254666339359	0.4307312196581262
image.image.full image: 20° edg…	gradients	0.30734669081481353	0.3405596107998725	0.3898866268454212
image.image.full image: 40° edg…	gradients	0.2805120241369688	0.3535867839350106	0.34526639869342274
image.image.full image: 60° edg…	gradients	0.24968960892178985	0.34246260181346083	0.24842924598445873
image.image.full image: 80° edg…	gradients	0.28940764306895767	0.3343457489270969	0.218448445086278
image.image.full image: 100° ed…	gradients	0.2971168688689842	0.32721907483810225	0.25272365053683943
image.image.full image: 120° ed…	gradients	0.29698898993847617	0.3069970271818208	0.28025588477519725

Show 10 fields 234 1 to 10 of 237 fields 1 2 3 4 5

Figure 4.43: A composite source in Table+Image format

As seen in Figure 4.43 above, there are not only image path, image id and extracted image features as fields in the composite source, but also the “label” field which was from the CSV.

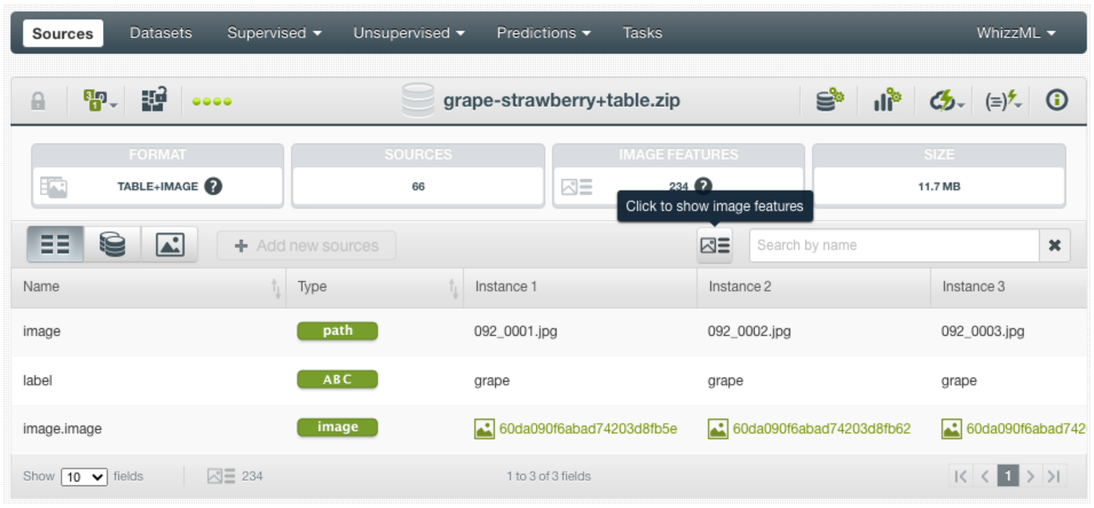
The uploading of the zip file above and the subsequent creation of its dataset is essentially equivalent to the following operations combined:

1. Upload the CSV and create a source;
2. Upload the images and create an image composite source;
3. Create datasets from the two sources, respectively;
4. Perform a dataset join by the column “image” in the CSV resulted dataset and the field “image” in the image dataset.

4.4.1 Views of Table+Image Composite Sources

Just like “Image” composite sources, “Table+Image” composite sources also have three views.

The **fields view** lists all the fields in the sources, that are the fields from the images, which includes at least the image field and the path field, as well as the fields from the table source, such as from a CSV.



FORMAT	SOURCES	IMAGE FEATURES	SIZE
TABLE+IMAGE	66	234	11.7 MB

Name	Type	Instance 1	Instance 2	Instance 3
image	path	092_0001.jpg	092_0002.jpg	092_0003.jpg
label	ABC	grape	grape	grape
image.image	image	60da090f6abad74203d8fb5e	60da090f6abad74203d8fb62	60da090f6abad742

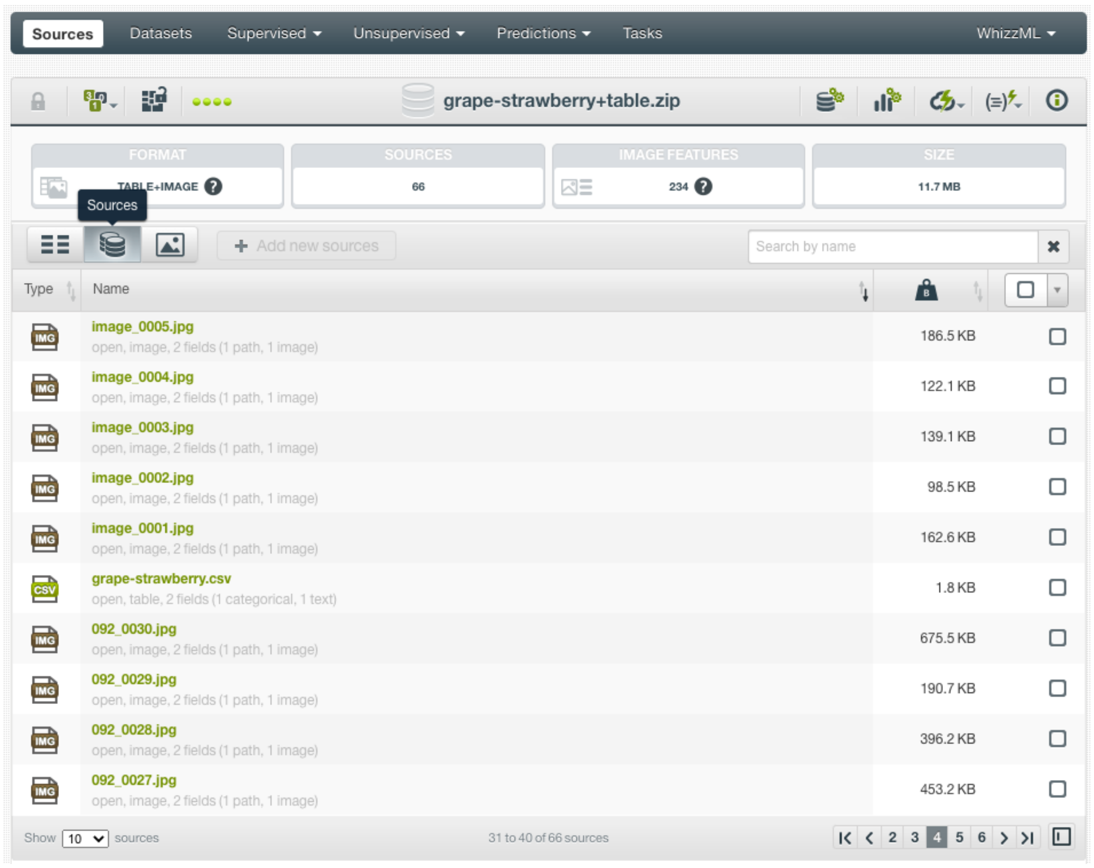
Show 10 fields 234 1 to 3 of 3 fields 1

Figure 4.44: The fields view of a Table+Image composite source

As seen above, the **fields view** shows the image field and the path field from the images, and the categorical field called “label” from the table source.

Optionally, users can click on the “show image features” icon next to the search box to show all image features in the view.

The **sources view** list all component sources, including the table component, which is a CSV file in the example below.



FORMAT	SOURCES	IMAGE FEATURES	SIZE
TABLE+IMAGE	66	234	11.7 MB

Type	Name		
IMG	image_0005.jpg open, image, 2 fields (1 path, 1 image)	186.5 KB	☐
IMG	image_0004.jpg open, image, 2 fields (1 path, 1 image)	122.1 KB	☐
IMG	image_0003.jpg open, image, 2 fields (1 path, 1 image)	139.1 KB	☐
IMG	image_0002.jpg open, image, 2 fields (1 path, 1 image)	98.5 KB	☐
IMG	image_0001.jpg open, image, 2 fields (1 path, 1 image)	162.6 KB	☐
CSV	grape-strawberry.csv open, table, 2 fields (1 categorical, 1 text)	1.8 KB	☐
IMG	092_0030.jpg open, image, 2 fields (1 path, 1 image)	675.5 KB	☐
IMG	092_0029.jpg open, image, 2 fields (1 path, 1 image)	190.7 KB	☐
IMG	092_0028.jpg open, image, 2 fields (1 path, 1 image)	396.2 KB	☐
IMG	092_0027.jpg open, image, 2 fields (1 path, 1 image)	453.2 KB	☐

Show 10 sources 31 to 40 of 66 sources 2 3 4 5 6

Figure 4.45: The sources view of a Table+Image composite source

The **images view** of a “Table+Image” composite is different from that of a “Image” composite. For an

"Image" composite source, users can preview images, add and edit labels in its **images view**. But in the **images view** of a "Table+Image" composite source, users can only preview images.

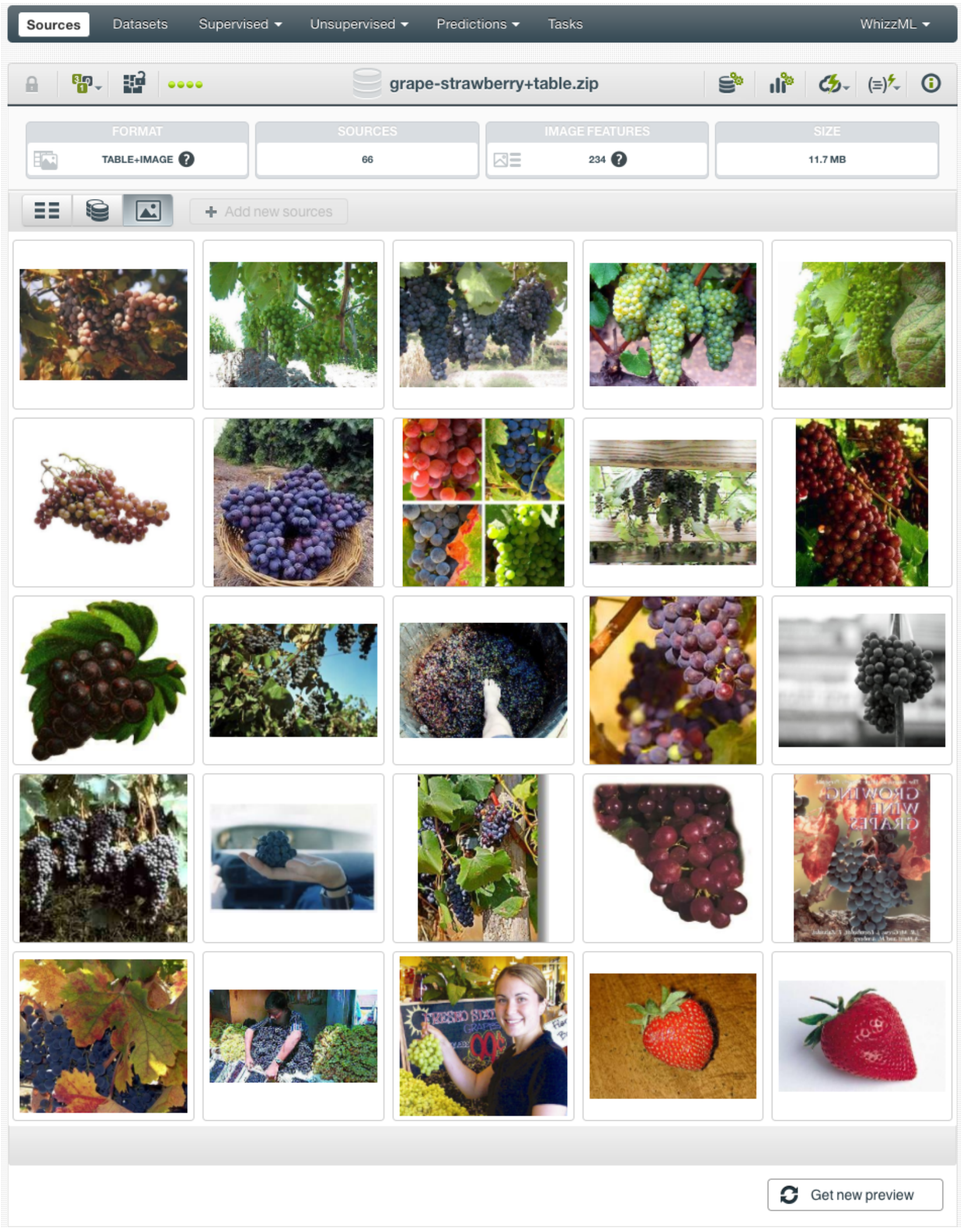
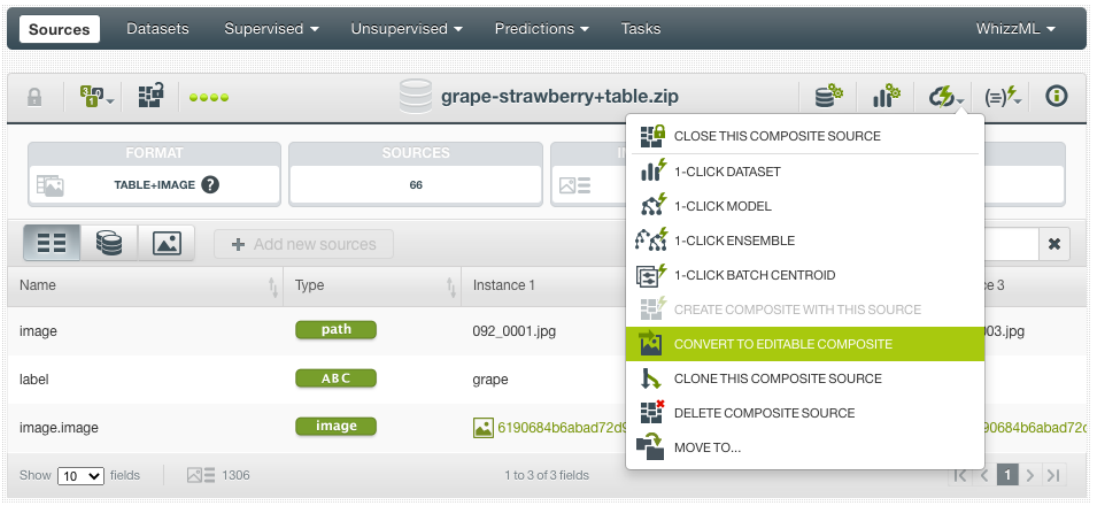


Figure 4.46: The images view of a Table+Image composite source

4.4.2 Convert Table+Image Composites to Editable Image Composites

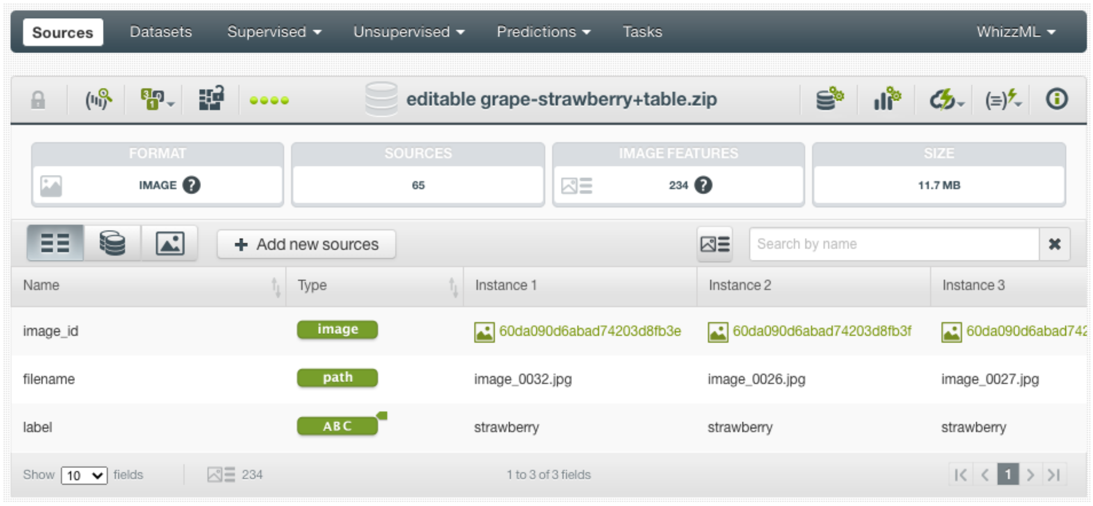
Users can convert a "Table+Image" composite source to an "Image" composite source, which becomes editable. Under any view of a "Table+Image" composite source, mover over the cloud action icon on the right of the source title, then click on the menu item **CONVERT TO EDITABLE COMPOSITE**



Name	Type	Instance 1	
image	path	092_0001.jpg	03.jpg
label	ABC	grape	
image.image	image	6190684b6abad72d	90684b6abad72c

Figure 4.47: Convert a Table+Image composite source to an editable Image composite source

A new “Image” composite source is then created, adding “editable” to its original title as the default new title.



Name	Type	Instance 1	Instance 2	Instance 3
image_id	image	60da090d6abad74203d8fb3e	60da090d6abad74203d8fb3f	60da090d6abad742
filename	path	image_0032.jpg	image_0026.jpg	image_0027.jpg
label	ABC	strawberry	strawberry	strawberry

Figure 4.48: A new Image composite source after conversion

In the new “Image” composite source, all fields from the table source become label fields. It contains all image sources as its component sources, and the total number of component sources is reduced by 1 comparing to the orginal “Table+Image” composite source – the table source is gone.

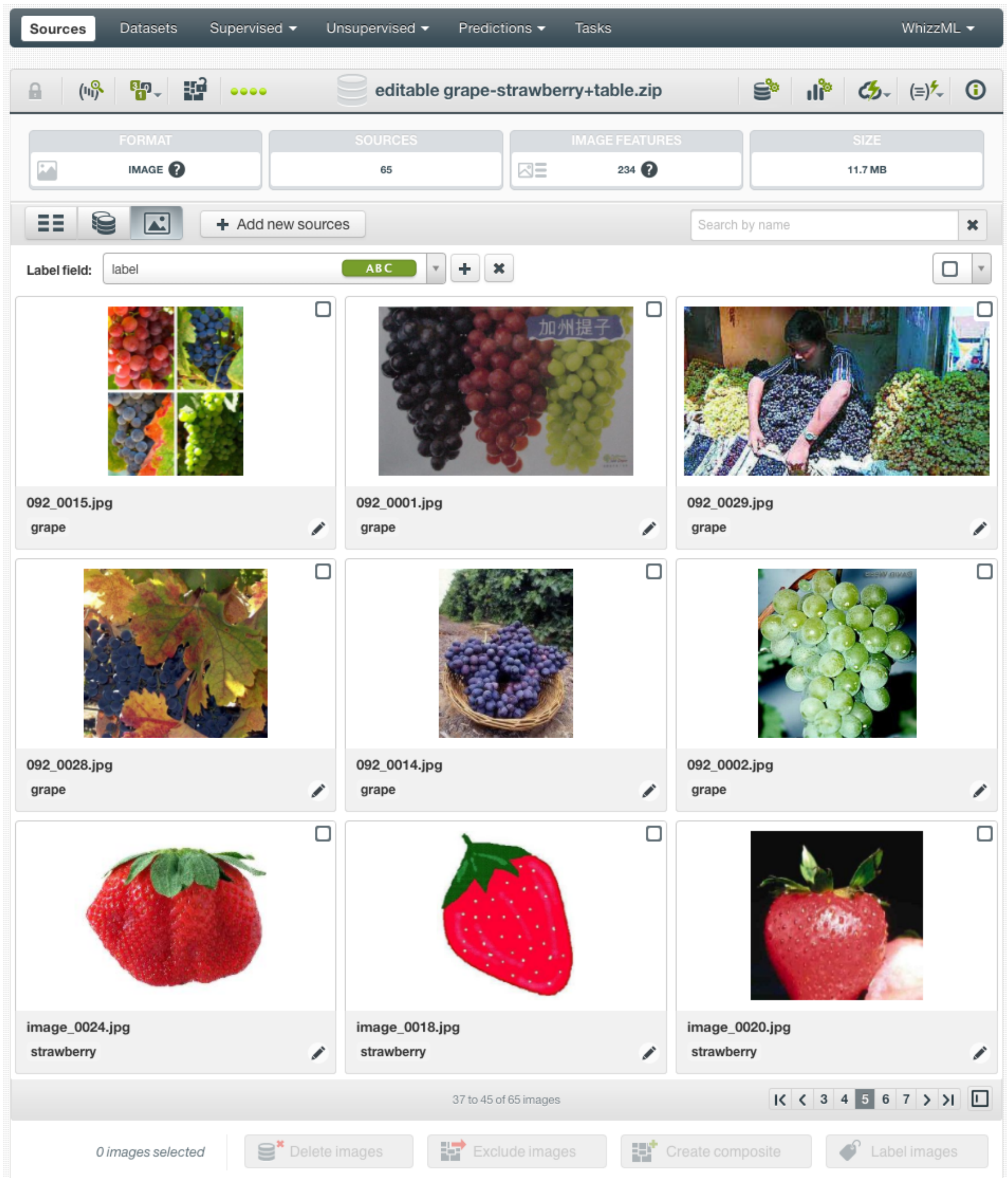


Figure 4.49: The images view after conversion

In the **images view** of the converted composite, not only the images can be previewed with pagination, but also the label fields can be edited.

4.5 Merging Image Composite Sources

As stated in the beginning of the chapter, a composite source can be created from closed composite sources. For creating a composite from any existing sources, please see Section 4.1. A good application of this is to merge image composite sources. Here, "merge" refers to the concatenation of all the rows from each component source inside the created composite source.

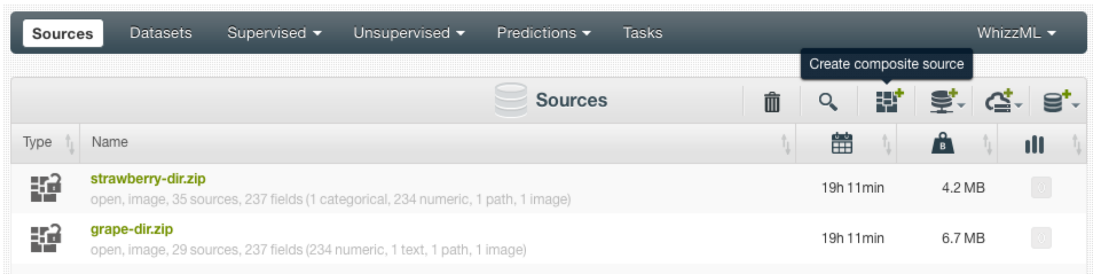
For instance, images are labeled separately sometimes. One class of images were processed and uploaded to the BigML platform. Then another class of images were uploaded. They need to be merged to create datasets for machine learning.

When creating a composite source, the component sources should have the same fields, in other words they are homogeneous, so that the composite source can inherit all fields, the same as each component source. This maintains the "format" of the composite source (Section 2.2).

If the component sources don't have the same fields, or they are heterogeneous, the resulting composite source will have "Mixed" format, which cannot be used to create datasets.

Therefore, when merging image composite sources, it is very important to make sure that they have identical fields.

To create a composite source from existing composite source, use the Create composite source button on the action bar in the **source list view** as shown below.

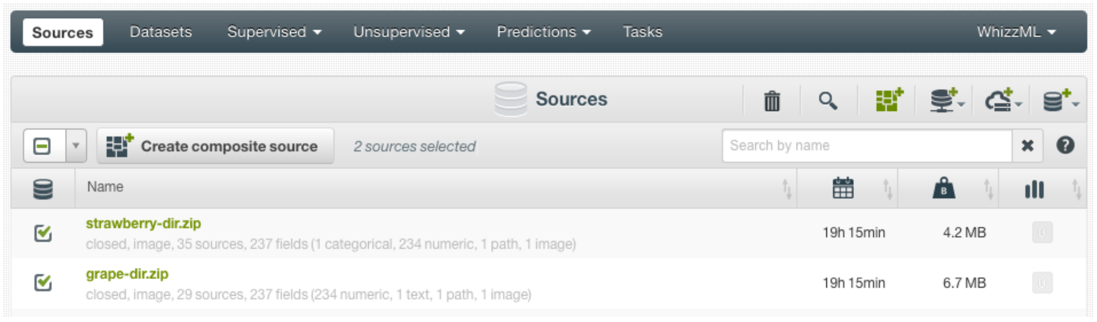


strawberry-dir.zip
open, image, 35 sources, 237 fields (1 categorical, 234 numeric, 1 path, 1 image) 19h 11min 4.2 MB

grape-dir.zip
open, image, 29 sources, 237 fields (234 numeric, 1 text, 1 path, 1 image) 19h 11min 6.7 MB

Figure 4.50: Creating a new composite source

Next, select the component sources by using the checkboxes next to them. In this case, select the existing image composite sources for the purpose of merging them.



strawberry-dir.zip
closed, image, 35 sources, 237 fields (1 categorical, 234 numeric, 1 path, 1 image) 19h 15min 4.2 MB

grape-dir.zip
closed, image, 29 sources, 237 fields (234 numeric, 1 text, 1 path, 1 image) 19h 15min 6.7 MB

Figure 4.51: Selecting component sources

Then click on the Create composite source button to create the new composite source.

CHAPTER 5

Source Fields

BigML will automatically classify the fields in your source into one of the types defined in the following subsections.

5.1 Numeric

Numeric fields are used to represent both integer and real numbers. Figure 5.1 shows the icon that BigML uses to refer to them.

Figure 5.1: Numeric Field Icon

5.2 Categorical

Categorical[1] fields, also known as nominal fields, take a small number of pre-defined values or categories. The icon BigML uses to represent categorical fields is shown in Figure 5.2.

Figure 5.2: Categorical Field Icon

When BigML processes a field that only takes two values (like 0 or 1), it automatically assigns the type categorical to the field.

BigML has a limit of **1,000 categories** for each categorical field. When BigML detects a field with more than 1,000 categories, it automatically changes the type to **text**. If you are interested in modeling more categories in only one field, consider a **BigML Private Deployment** that allows the number of categories to be upgraded to tens of thousands.

5.3 Date-Time

Date-time fields are used to represent machine-readable date/time information. The icon BigML uses to represent date-time fields is shown in Figure 5.3.

[1]https://en.wikipedia.org/wiki/Categorical_variable

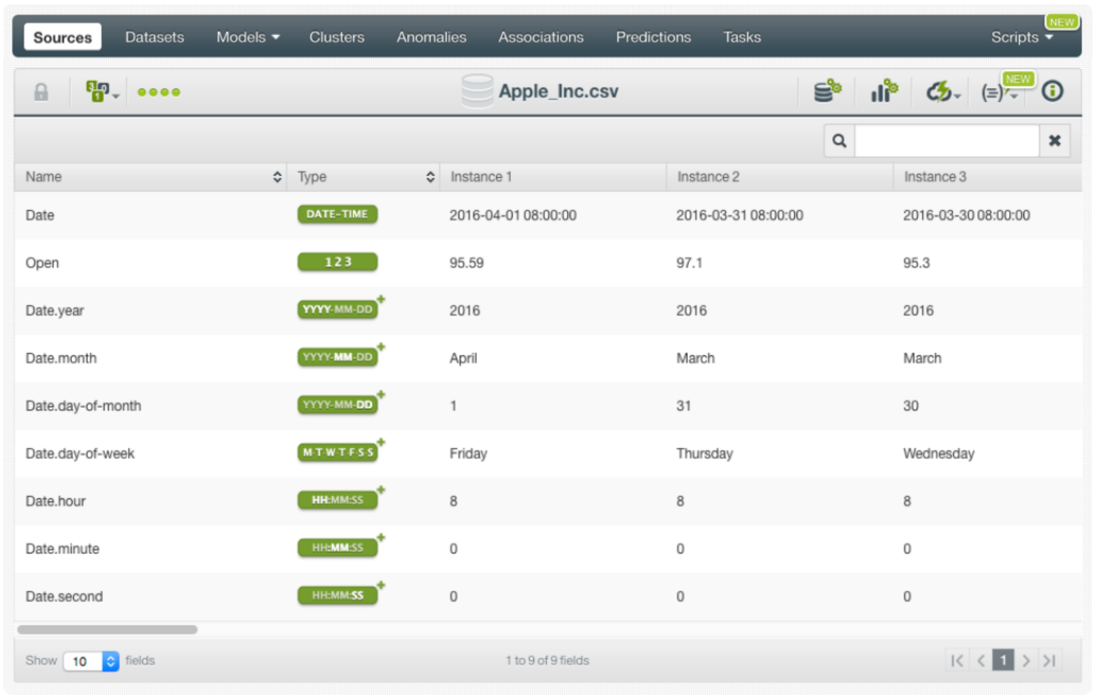
DATE-TIME

Figure 5.3: Date-time field icon

When BigML detects a date-time field, it expands it into additional fields with their numeric components. For date fields, **year**, **month**, **day**, and **day of the week** are generated. For time fields, **hour**, **minute**, and **second** are generated (see Figure 5.4). For fields that include both a date and time component, the seven fields above are generated. For example, the following CSV file has a date-time field named **Date** that will get expanded into the seven additional fields shown on Figure 5.5.

```
Date, Open
2016-04-01 08:00:00, 95.59
2016-03-31 08:00:00, 97.1
2016-03-30 08:00:00, 95.3
```

Figure 5.4: A CSV file with a date-time field



Name	Type	Instance 1	Instance 2	Instance 3
Date	DATE-TIME	2016-04-01 08:00:00	2016-03-31 08:00:00	2016-03-30 08:00:00
Open	123	95.59	97.1	95.3
Date.year	YYYY-MM-DD	2016	2016	2016
Date.month	YYYY-MM-DD	April	March	March
Date.day-of-month	YYYY-MM-DD	1	31	30
Date.day-of-week	M T W T F S S	Friday	Thursday	Wednesday
Date.hour	HH:MM:SS	8	8	8
Date.minute	HH:MM:SS	0	0	0
Date.second	HH:MM:SS	0	0	0

Figure 5.5: A source with a date-time field expanded

You can enable or disable automatic generation by switching the Expand date-time fields setting in the CONFIGURE SOURCE menu option. (See Chapter 6.) When disabled, potential date-time fields will be treated as either categorical or text fields.

By default, BigML, accepts date and times that follow the ISO 8601[2] standard. BigML also recognizes the formats listed on Table 5.1.

Table 5.1: Extra date-time formats recognized by BigML

Name	Example	Format
basic-date-time	19690714T173639.592Z	`YYYYMMdd'T'HHmmss.SSSXX`

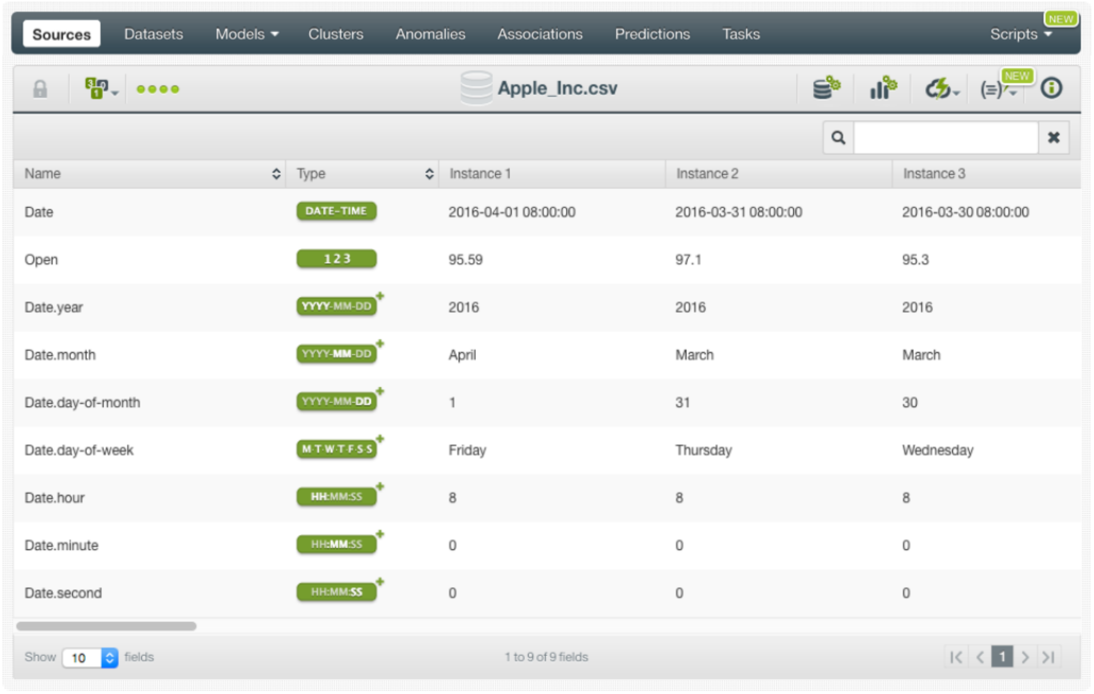
[2]https://en.wikipedia.org/wiki/ISO_8601

basic-date-time-no-ms	19690714T173639Z	`YYYYMMdd'T'HHmmssXX`
basic-iso-date	19690714Z	`YYYYMMddXX`
basic-ordinal-date-time	1969195T173639.592Z	`YYYYDDD'T'HHmmss.SSSXX`
basic-ordinal-date-time-no-ms	1969195T173639Z	`YYYYDDD'T'HHmmssXX`
basic-t-time	T173639.592Z	`'T'HHmmss.SSSXX`
basic-t-time-no-ms	T173639Z	`'T'HHmmssXX`
basic-time	173639.592Z	`HHmmss.SSSXX`
basic-time-no-ms	173639Z	`HHmmssXX`
basic-week-date	1969W291	`xxxx'W'wwe`
basic-week-date-time	1969W291T173639.592Z	`xxxx'W'wwe'T'Hmmss.SSSXX`
basic-week-date-time-no-ms	1969W291T173639Z	`xxxx'W'wwe'T'HmmssXXX`
bigquery	1969-07-14 17:36:39Z	`Y-M-d H:m:sXXX`
bigquery-alt	1969-07-14 17:36:39 UTC	`Y-M-d H:m:s z`
bigquery-alt-millisecond	1969-7-14 17:36:39.592000 UTC	`Y-M-d H:m:s.SSSSSS z`
bigquery-millisecond	1969-7-14 17:36:39.592000Z	`Y-M-d H:m:s.SSSSSSXXX`
clock-minute	5:36 PM	`h:m a`
clock-minute-nospace	5:36PM	`h:ma`
clock-second	5:36:39 PM	`h:m:s a`
clock-second-nospace	5:36:39PM	`h:m:sa`
date	1969-07-14	`Y-M-d`
date-hour	1969-07-14T17	`YYYY-MM-dd'T'H`
date-hour-minute	1969-07-14T17:36	`YYYY-MM-dd'T'H:mm`
date-hour-minute-second	1969-07-14T17:36:39	`YYYY-MM-dd'T'H:mm:ss`
date-hour-minute-second-fraction	1969-07-14T17:36:39.592	`YYYY-MM-dd'T'H:mm:ss.SSS`
date-hour-minute-second-fraction-with-solidus	1969/07/14T17:36:39.592	`YYYY/MM/dd'T'H:mm:ss.SSS`
date-hour-minute-second-ms	1969-07-14T17:36:39.592	`YYYY-MM-dd'T'HH:mm:ss.SSS`
date-hour-minute-second-ms-with-solidus	1969/07/14T17:36:39.592	`YYYY/MM/dd'T'HH:mm:ss.SSS`
date-hour-minute-second-with-solidus	1969/07/14T17:36:39	`YYYY/MM/dd'T'H:mm:ss`
date-hour-minute-with-solidus	1969/07/14T17:36	`YYYY/MM/dd'T'H:mm`
date-hour-with-solidus	1969/07/14T17	`YYYY/MM/dd'T'H`
date-time	1969-07-14T17:36:39.592Z	`YYYY-MM-dd'T'H:mm:ss.SSSXXX`
date-time-no-ms	1969-07-14T17:36:39Z	`YYYY-MM-dd'T'H:mm:ssXXX`
date-time-no-ms-with-solidus	1969/07/14T17:36:39Z	`YYYY/MM/dd'T'H:mm:ssXXX`
date-time-with-solidus	1969/07/14T17:36:39.592Z	`YYYY/MM/dd'T'H:mm:ss.SSSXXX`
date-with-solidus	1969/07/14	`YYYY/MM/dd`
elasticsearch-nanos	1969-07-14T17:36:39.592000Z	`YYYY-MM-dd'T'HH:mm:ss.SSSSSSZ`
eu-date	14/7/1969	`d/M/Y`

eu-date-clock-minute	14/7/1969 5:36 PM	`d/M/Y h:m a`
eu-date-clock-minute-nospace	14/7/1969 5:36PM	`d/M/Y h:ma`
eu-date-clock-second	14/7/1969 5:36:39 PM	`d/M/Y h:m:s a`
eu-date-clock-second-nospace	14/7/1969 5:36:39PM	`d/M/Y h:m:sa`
eu-date-millisecond	14/7/1969 17:36:39.592	`d/M/Y H:m:s.SSS`
eu-date-minute	14/7/1969 17:36	`d/M/Y H:m`
eu-date-second	14/7/1969 17:36:39	`d/M/Y H:m:s`
eu-ddate	14.7.1969	`d.M.Y`
eu-ddate-clock-minute	14.7.1969 5:36 PM	`d.M.Y h:m a`
eu-ddate-clock-minute-nospace	14.7.1969 5:36PM	`d.M.Y h:ma`
eu-ddate-clock-second	14.7.1969 5:36:39 PM	`d.M.Y h:m:s a`
eu-ddate-clock-second-nospace	14.7.1969 5:36:39PM	`d.M.Y h:m:sa`
eu-ddate-millisecond	14.7.1969 17:36:39.592	`d.M.Y H:m:s.SSS`
eu-ddate-minute	14.7.1969 17:36	`d.M.Y H:m`
eu-ddate-second	14.7.1969 17:36:39	`d.M.Y H:m:s`
eu-sdate	14-7-1969	`d-M-Y`
eu-sdate-clock-minute	14-7-1969 5:36 PM	`d-M-Y h:m a`
eu-sdate-clock-minute-nospace	14-7-1969 5:36PM	`d-M-Y h:ma`
eu-sdate-clock-second	14-7-1969 5:36:39 PM	`d-M-Y h:m:s a`
eu-sdate-clock-second-nospace	14-7-1969 5:36:39PM	`d-M-Y h:m:sa`
eu-sdate-millisecond	14-7-1969 17:36:39.592	`d-M-Y H:m:s.SSS`
eu-sdate-minute	14-7-1969 17:36	`d-M-Y H:m`
eu-sdate-second	14-7-1969 17:36:39	`d-M-Y H:m:s`
hour-minute	17:36	`H:mm`
hour-minute-second	17:36:39	`H:mm:ss`
hour-minute-second-fraction	17:36:39.592	`H:mm:ss.SSS`
hour-minute-second-ms	17:36:39.592	`H:mm:ss.SSS`
iso-date	1969-07-14Z	`Y-M-dXXX`
iso-date-time	1969-07-14T17:36:39.592Z	`Y-M-d'T'HH:mm:ss.SSSXXX`
iso-instant	1969-07-14T17:36:39.592Z	`Y-M-d'T'HH:mm:ss.SSSXXX`
iso-local-date	1969-07-14	`Y-M-d`
iso-local-date-time	1969-07-14T17:36:39.592	`Y-M-d'T'HH:mm:ss`
iso-local-time	17:36:39.592	`HH:mm:ss.SSS`
iso-offset-date	1969-07-14Z	`Y-M-dXXX`
iso-offset-date-time	1969-07-14T17:36:39.592Z	`Y-M-d'T'HH:mm:ss.SSSXXX`
iso-offset-time	17:36:39.592Z	`HH:mm:ss.SSSXXX`
iso-ordinal-date	1969-195Z	`YYYY-DDDXX`

iso-time	17:36:39.592Z	`HH:mm:ss.SSSXXX`
iso-week-date	1969-W29-1Z	`xxxx-'W'ww-ez`
iso-zoned-date-time	1969-07-14T17:36:39.592Z	`Y-M-d'T'HH:mm:ss.SSSXXX`
mysql	1969-07-14 17:36:39	`YYYY-MM-dd H:mm:ss`
no-t-date-hour-minute	1969-7-14 17:36	`YYYY-MM-dd H:m`
odata-format	/Date(-14711000408)/	
ordinal-date-time	1969-195T17:36:39.592Z	`YYYY-DDD'T'H:mm:ss.SSSXXX`
ordinal-date-time-no-ms	1969-195T17:36:39Z	`YYYY-DDD'T'H:mm:ssXXX`
rfc-1123-date-time	Mon, 14 Jul 1969 17:36:39 GMT	`EEE, dd MMM YYYY HH:mm:ss z`
rfc822	Mon, 14 Jul 1969 17:36:39 +0000	`EEE, dd MMM YYYY HH:mm:ss ZZZZ`
t-time	T17:36:39.592Z	`'T'HH:mm:ss.SSSXXX`
t-time-no-ms	T17:36:39Z	`'T'HH:mm:ssXXX`
time	17:36:39.592Z	`HH:mm:ss.SSSXXX`
time-no-ms	17:36:39Z	`HH:mm:ssXXX`
timestamp	-14711000	
timestamp-msecs	-14711000408	
twitter-time	Mon Jul 14 17:36:39 +0000 1969	`E MMM d H:m:s Z Y`
twitter-time-alt	1969-7-14 17:36:39 +0000	`Y-M-d H:m:s Z`
twitter-time-alt-2	1969-7-14 17:36 +0000	`Y-M-d H:m Z`
twitter-time-alt-3	Mon Jul 14 17:36 +0000 1969	`E MMM d H:m Z Y`
us-date	7/14/1969	`M/d/Y`
us-date-clock-minute	7/14/1969 5:36 PM	`M/d/Y h:m a`
us-date-clock-minute-nospace	7/14/1969 5:36PM	`M/d/Y h:ma`
us-date-clock-second	7/14/1969 5:36:39 PM	`M/d/Y h:m:s a`
us-date-clock-second-nospace	7/14/1969 5:36:39PM	`M/d/Y h:m:sa`
us-date-millisecond	7/14/1969 17:36:39.592	`M/d/Y H:m:s.SSS`
us-date-minute	7/14/1969 17:36	`M/d/Y H:m`
us-date-second	7/14/1969 17:36:39	`M/d/Y H:m:s`
us-sdate	7-14-1969	`M-d-Y`
us-sdate-clock-minute	7-14-1969 5:36 PM	`M-d-Y h:m a`
us-sdate-clock-minute-nospace	7-14-1969 5:36PM	`M-d-Y h:ma`
us-sdate-clock-second	7-14-1969 5:36:39 PM	`M-d-Y h:m:s a`
us-sdate-clock-second-nospace	7-14-1969 5:36:39PM	`M-d-Y h:m:sa`
us-sdate-millisecond	7-14-1969 17:36:39.592	`M-d-Y H:m:s.SSS`
us-sdate-minute	7-14-1969 17:36	`M-d-Y H:m`
us-sdate-second	7-14-1969 17:36:39	`M-d-Y H:m:s`
week-date	1969-W29-1	`xxxx-'W'ww-e`
week-date-time	1969-W29-1T17:36:39.592Z	`xxxx-'W'ww-e'T'H:mm:ss.SSSXXX`

week-date-time-no-ms	1969-W29-1T17:36:39Z	`xxxx-'W'ww-e'T'H:mm:ssXXX`
weekyear-week	1969-W29	`xxxx-'W'ww`
weekyear-week-day	1969-W29-1	`xxxx-'W'ww-e`
year-month	1969-07	`YYYY-MM`
year-month-day	1969-07-14	`YYYY-MM-dd`
other	Define your own format	



Name	Type	Instance 1	Instance 2	Instance 3
Date	DATE-TIME	2016-04-01 08:00:00	2016-03-31 08:00:00	2016-03-30 08:00:00
Open	123	95.59	97.1	95.3
Date.year	YYYY-MM-DD	2016	2016	2016
Date.month	YYYY-MM-DD	April	March	March
Date.day-of-month	YYYY-MM-DD	1	31	30
Date.day-of-week	M T W T F S S	Friday	Thursday	Wednesday
Date.hour	HH:MM:SS	8	8	8
Date.minute	HH:MM:SS	0	0	0
Date.second	HH:MM:SS	0	0	0

Figure 5.6: A source with a date-time field expanded

If your date-time field is not automatically recognized, you can configure your field and select the right format or input a custom format. See a detailed explanation in Subsection 6.11.1.

5.4 Text

Text fields (or string fields) are used to represent an arbitrary number of characters. Many Machine Learning algorithms are designed to work only with numeric and categorical fields and cannot easily handle text fields. BigML takes a basic and reliable approach, leveraging some basic Natural Language Processing[3] (NLP) techniques along with a simple (bag-of-words[4]) style method of feature generation to include text fields within its modeling framework.

Text fields are specially processed by BigML using the configuration options explained in Chapter 6.

First, BigML performs some basic language detection. BigML recognizes texts in Arabic, Catalan, Chinese, Czech, Danish, Dutch, English, Farsi/Persian, Finish, French, German, Hungarian, Italian, Japanese, Korean, Polish, Portuguese, Turkish, Romanian, Russian, Spanish, and Swedish. Please let the Support Team at BigML[5] know if you want BigML to add your language.

[3] https://en.wikipedia.org/wiki/Natural_language_processing
[4] https://en.wikipedia.org/wiki/Bag-of-words_model
[5] support@bigml.com

BigML can also perform case sensitive or insensitive analyses, remove stop words[6] before processing the text, search for n-grams[7] in the text, use some basic stemming[8], and apply different filters to your text fields. Finally, it can use different tokenization[9] strategies. All these options are described in Chapter 6.

The icon that BigML uses to refer to text fields is shown on Figure 5.7.

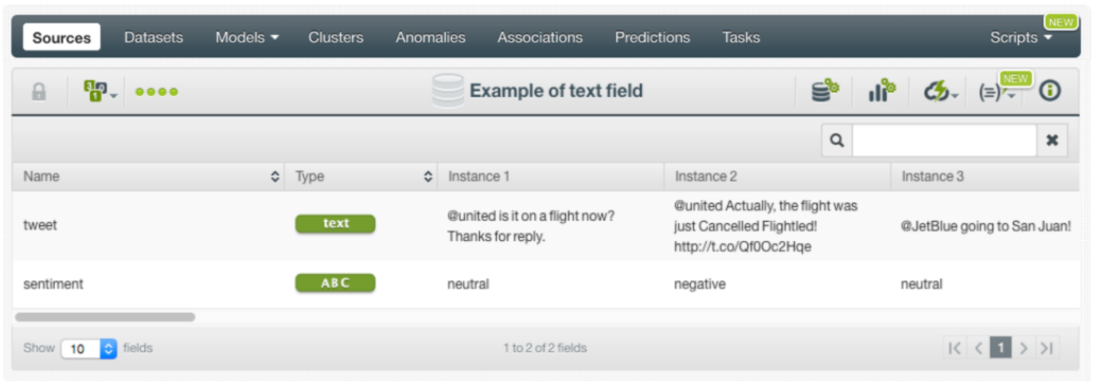
text

Figure 5.7: Text field icon

Figure 5.8 is an example of a CSV[10] file with a text field. It has two fields: the first one is the text of a tweet directed to an airline, and the second one is a label that represents a sentiment (i.e., positive, negative, or neutral). If you create a source with that file, BigML will automatically assign the types **text** and **categorical** as shown on Figure 5.9.

```
tweet, sentiment
@united is it on a flight now? Thanks for reply.,neutral
"@united Actually, the flight was just Cancelled Flightled!
http://t.co/Qf0Oc2HqeZ",negative
@JetBlue going to San Juan!,neutral
@united flights taking off from IAD this afternoon?,neutral
@JetBlue I LOVE JET BLUE!,positive
@JetBlue thanks. I appreciate your prompt response.,positive
"@united diverged to Burlington, Vermont. This sucks.",negative
@SouthwestAir and thx for not responding,negative
@AmericanAir  @SouthwestAir  - Y'all will like this one.
http://t.co/hF8aJZ4ffl,neutral
@USAirways you guys lost my luggage,negative
```

Figure 5.8: An excerpt of an example of a CSV file with a text field



Name	Type	Instance 1	Instance 2	Instance 3
tweet	text	@united is it on a flight now? Thanks for reply.	@united Actually, the flight was just Cancelled Flightled! http://t.co/Qf0Oc2Hqe	@JetBlue going to San Juan!
sentiment	ABC	neutral	negative	neutral

Figure 5.9: An example of a source with a text field

[6]https://en.wikipedia.org/wiki/Stop_words
[7]https://en.wikipedia.org/wiki/N-gram
[8]https://en.wikipedia.org/wiki/Stemming
[9]https://en.wikipedia.org/wiki/Tokenization_(lexical_analysis)
[10]https://github.com/monkeylearn/sentiment-analysis-benchmark

5.5 Items

When a field contains an arbitrary number of items (categories or labels), BigML assigns the type **items** to it. Items are separated using a special separator that is configured independently of the CSV separator used to separate the rest of fields of the source. These types of fields are used mainly for association discovery.

The icon used by BigML to denote items fields is shown in Figure 5.10.

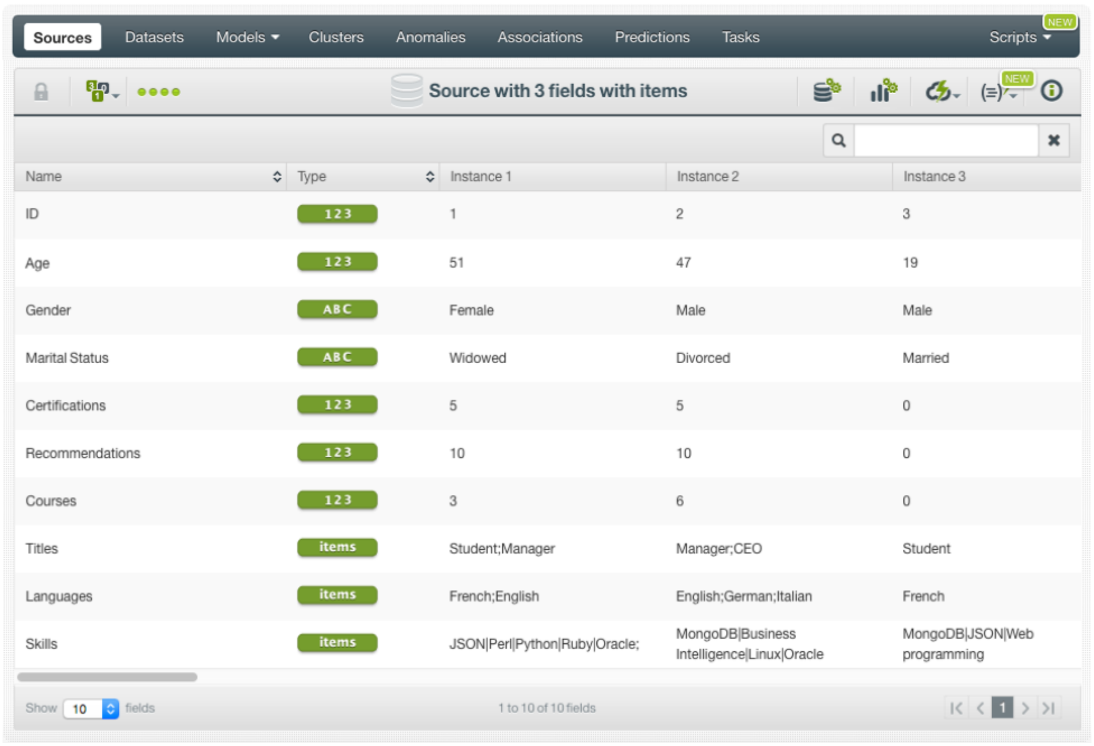
items

Figure 5.10: Items field icon

A source can have multiple fields with items each one using a different **items separator**. Figure 5.11 shows an example of sources with three items fields. The first two use the ";" (semicolon) as items separator, and the third one uses the "|" (pipe) as items separator. Figure 5.12 shows how BigML recognizes them after being configured, using the panel described in Chapter 6 to set up a different separator for each field.

```
ID,Age,Gender,Marital
Status,Certifications,Recommendations,Courses,Titles,Languages,Skills
1,51,Female,Widowed,5,10,3,Student;Manager,French;English,JSON|Perl|Python|Ruby|Oracle;
2,47,Male,Divorced,5,10,6,Manager;CEO,English;German;Italian,MongoDB|Business
Intelligence|Linux|Oracle
3,19,Male,Married,0,0,0,Student,French,MongoDB|JSON|Web
programming
4,45,Male,Divorced,1,5,3,Engineer,German;English,Windows|MongoDB|Algorithm
Design|MySQL|Linux
```

Figure 5.11: An excerpt of an example of a CSV file with three items fields



Name	Type	Instance 1	Instance 2	Instance 3
ID	123	1	2	3
Age	123	51	47	19
Gender	ABC	Female	Male	Male
Marital Status	ABC	Widowed	Divorced	Married
Certifications	123	5	5	0
Recommendations	123	10	10	0
Courses	123	3	6	0
Titles	items	Student;Manager	Manager;CEO	Student
Languages	items	French;English	English;German;Italian	French
Skills	items	JSON|Perl|Python|Ruby|Oracle;	MongoDB|Business Intelligence|Linux|Oracle	MongoDB|JSON|Web programming

Show 10 fields 1 to 10 of 10 fields

Figure 5.12: An example of a source with 3 fields with items

5.6 Image

When images are recognized in the raw data, BigML will generate two fields for each image. One field is assigned as “image”, which is pointing to the normalized version of the original image. This is the icon that BigML uses to refer to them:

image

Figure 5.13: Image field icon

Another field is its filename and is assigned as “path”. BigML uses this icon to refer to them:

path

Figure 5.14: Path field icon

Images are normalized to a “maximum resolution” of 512x512, while holding the aspect ratio constant.

5.7 Regions

Regions fields are used to represent areas of interest in an image. They are typically used as the objective field in object detection models. Figure 5.15 shows the icon BigML uses to refer to them.

regions

Figure 5.15: Regions field icon

The values of a regions field can be specified either by a list of 5-tuples, or a list of JSON maps, also know as JSON objects.

The canonical form of a region is a 5-tuple, or a list of 5 elements. The first element is the class label, the rest are the coordinates of the bounding box. Specifically, this is a region:

```
[label xmin ymin xmax ymax]
```

so is this (commas are optional):

```
[label, xmin, ymin, xmax, ymax]
```

where `label` is the class to which the object belongs, `(xmin ymin)` are the coordinates of the top left vertex of the bounding box, `(xmax ymax)` the coordinates of the bottom right vertex.

For example, the following two represent the same region:

```
["abc" 10 15 36 29]
["abc", 10, 15, 36, 29]
```

Equivalently, a JSON map can be used to represent a region:

```
{"label": label, "xmin": xmin, "ymin": ymin, "xmax": xmax, "ymax": ymax}
```

A JSON map representing a region may use any of these key collections:

1. `(label, xmin, ymin, xmax, ymax)` where `(xmin, ymin)` is the top left vertex of the region, while `(xmax, ymax)` the bottom right vertex. This is the canonical tuple representation. If no keys are used in a JSON map, the coordinates are assumed to be in this format.
2. `(label, xmin, ymin, width, height)` where `(xmin, ymin)` is the top left vertex of region, while `(width, height)` the width and height of the region, respectively.
3. `(label, xmax, ymax, width, height)` where `(xmax, ymax)` is the bottom right vertex of region, while `(width, height)` the width and height of the region, respectively.
4. `(label, x, y, width, height)` where `(x, y)` is the center of the region, while `(width, height)` the width and height of the region, respectively.
5. `(label, xcenter, ycenter, width, height)` which is a synonym of 4, with `xcenter` for `x` and `ycenter` for `y`.

As one image may contain more than one object, the value of its regions field is a list of 5-tuples, or a list of JSON maps. For instance, the following two JSON maps are equivalent for the annotation of one image (the backslash \ is the line continuation mark, meaning the two lines should be read as a single line):

```
{"file":"a.png","box":[["cat",20,150,250,300]]}
```

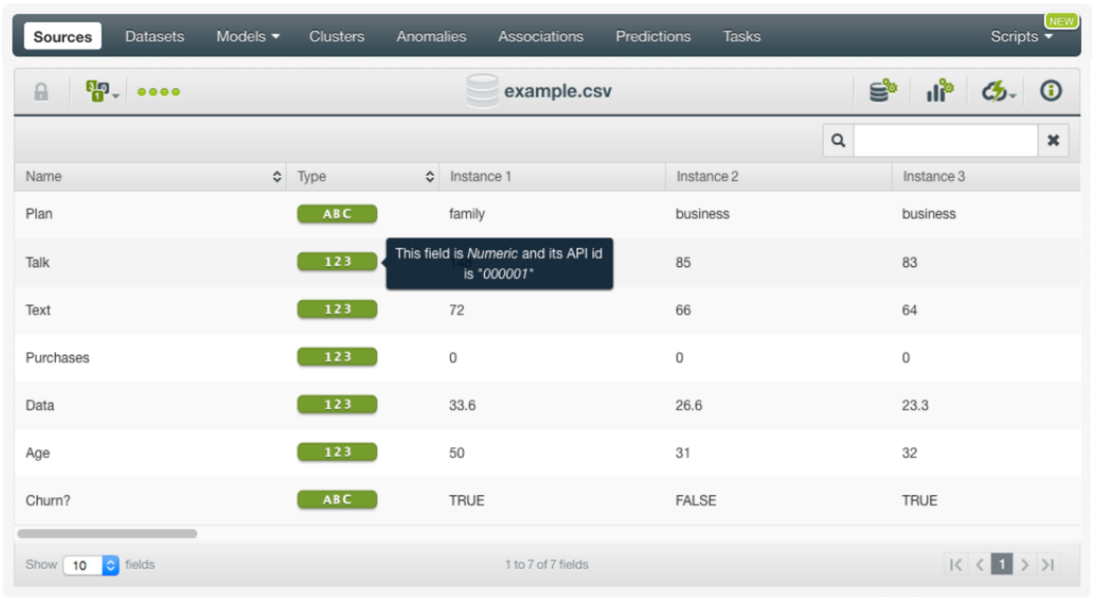
```
{"file":"a.png","box":[{"label":"cat", "xmin": 20, "ymin": 150, \
                       "xmax": 250, "ymax": 300}]}
```

The same region can also be specified as:

```
{"file": "a.png", "box": [{"label": "cat", "x": 135, "y": 225, \
                           "width": 230, "height": 150}]}
```

5.8 Field IDs

Each field is automatically assigned an ID in the form of a six-character hexadecimal number (e.g., "**000001**"). This ID can be used via the BigML API to retrieve and update the fields of a source. If you mouse over a field on the source view, you will see a tooltip with the corresponding ID of the field. (See Figure 5.16.)



This field is *Numeric* and its API id is "*000001*"

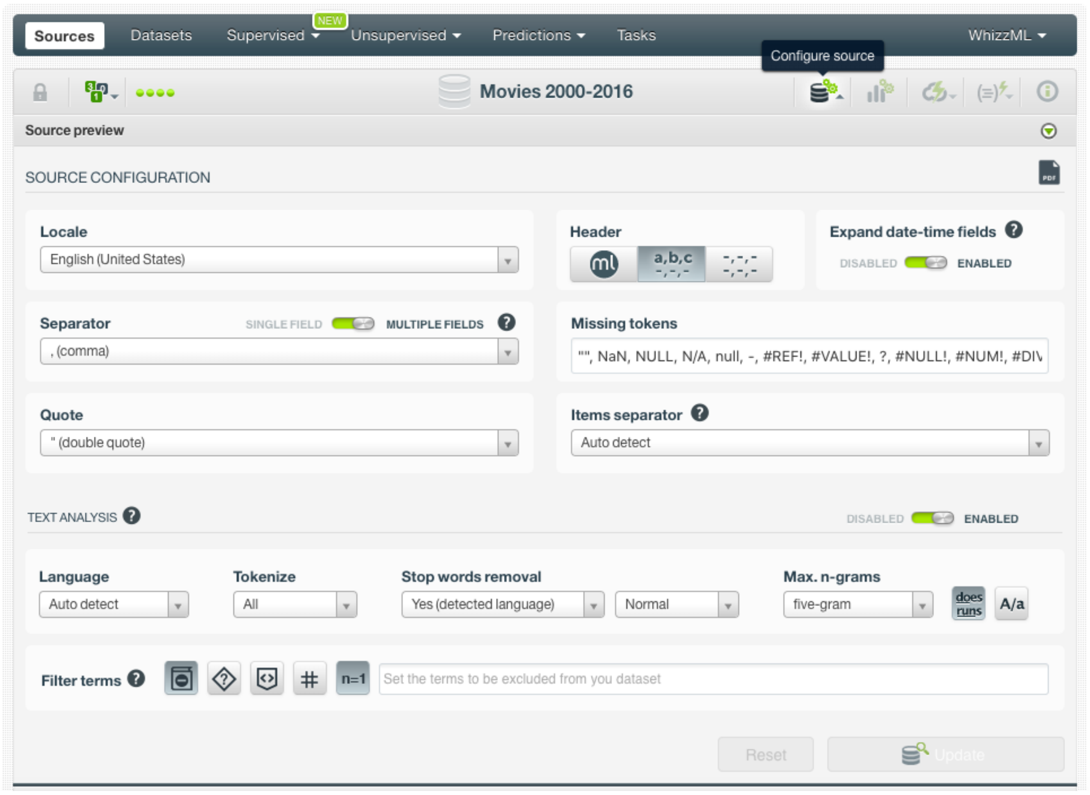
Name	Type	Instance 1	Instance 2	Instance 3
Plan	ABC	family	business	business
Talk	123		85	83
Text	123	72	66	64
Purchases	123	0	0	0
Data	123	33.6	26.6	23.3
Age	123	50	31	32
Churn?	ABC	TRUE	FALSE	TRUE

Figure 5.16: Field ID for API usage

CHAPTER 6

Source Configuration Options

Click on the CONFIGURE SOURCE menu option of a source view to get access to a panel (see Figure 6.1) where you can alter the way BigML processes your sources. The following subsections cover the available options. **Note: most of these options are only available for CSV files, not for other formats.**



SOURCE CONFIGURATION

Locale: English (United States)

Header

Expand date-time fields: DISABLED / ENABLED

Separator: , (comma) — SINGLE FIELD / MULTIPLE FIELDS

Missing tokens: "", NaN, NULL, N/A, null, -, #REF!, #VALUE!, ?, #NULL!, #NUM!, #DIV

Quote: " (double quote)

Items separator: Auto detect

TEXT ANALYSIS: DISABLED / ENABLED

Language: Auto detect

Tokenize: All

Stop words removal: Yes (detected language), Normal

Max. n-grams: five-gram

Filter terms: Set the terms to be excluded from you dataset

Reset

Update

Figure 6.1: Source configuration panel

6.1 Locale

The locale[1] allows you to define the specific language preferences you want BigML to use to process your source. This helps to ensure that some characters in your data are interpreted in the correct way. For example, different countries use different symbols for decimal marks.

[1] https://en.wikipedia.org/wiki/Locale

BigML tries to infer the locale from your browser. BigML also makes the locales listed in Table 6.1 available.

Language	Country
Arabic	United Arab Emirates
Chinese	China
Dutch	Netherlands
English	United Kingdom
English	United States
French	France
German	Germany
Greek	Greece
Hindi	India
Italian	Italy
Japanese	Japan
Korean	South Korea
Portuguese	Brazil
Russian	Russia
Spanish	Spain

Table 6.1: Default locales accepted by BigML

If your locale does not show on the **Locale** selector, and BigML does not process your data correctly, please let the Support Team at BigML[2] know.

6.2 Single Field or Multiple Fields

The **Single Field or Multiple Fields** switch allows you to tell BigML if your source is composed of only one field of type items.

6.2.1 Auto-Detection of Single, Item-Type Fields

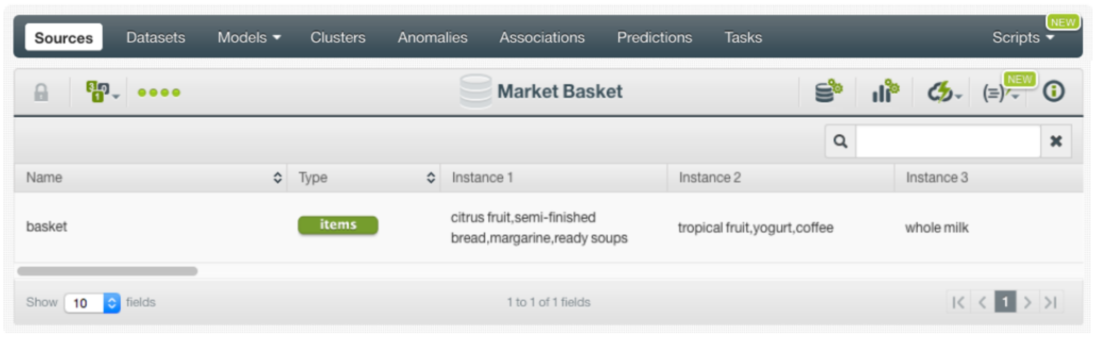
Sources containing a field of type items may be submitted without surrounding quotes, in which case the input will appear to have a varying number of columns in each row. Figure 6.2 shows an excerpt of a single-field source[3]. BigML will attempt to detect this case, rather than assume a "square" CSV format with a large number of bad rows. (See Figure 6.3). The criteria are as follows:

- The proportion of rows, whose column counts differ from the most frequent count, is greater than 0.25.
- There are no missing values as items.
- There are no items greater in length than 64 characters.

[2] support@bigml.com
[3] http://www.salemmarafi.com/code/market-basket-analysis-with-r/

```
basket
citrus fruit,semi-finished bread,margarine,ready soups
tropical fruit,yogurt,coffee
whole milk
pip fruit,yogurt,cream cheese ,meat spreads
other vegetables,whole milk,condensed milk,long life bakery
product
whole milk,butter,yogurt,rice,abrasive cleaner
rolls/buns
other vegetables,UHT-milk,rolls/buns,bottled beer,liquor
(appetizer)
pot plants
whole milk,cereals
tropical fruit,other vegetables,white bread,bottled
water,chocolate
citrus fruit,tropical fruit,whole
milk,butter,curd,yogurt,flour,bottled
water,dishes
beef
frankfurter,rolls/buns,soda
chicken,tropical fruit
```

Figure 6.2: An example of single field file with an item-type field



Name	Type	Instance 1	Instance 2	Instance 3
basket	items	citrus fruit,semi-finished bread,margarine,ready soups	tropical fruit,yogurt,coffee	whole milk

Figure 6.3: Source with a single field of type items

When a single-column source is detected, its **separator** is set to the **empty** string (""). There is no separator when there are not at least two columns to separate. You can also indicate that a source consists of a single column by setting the **separator** to the **empty** string ("").

Conversely, erroneous single-column auto-detections can be overridden via an update of the source by setting an items separator that is not the empty string.

6.3 Separator

The **separator** is the symbol that is used to separate each field within a CSV file. The default symbol is a **comma** (',') but you can choose one of the following ones or even input your own separator.

- semicolon (';')
- tab ('\t')

- space (' ')
- pipe ('|')

6.4 Quotes

You can select the symbol that will be used to quote complete fields. This is mandatory when the field includes the character used as separator or break lines. The two options are single quote (') or double quote (").

6.5 Missing Tokens

You can specify a list of tokens that will be considered equivalent to a missing value. By default, BigML recognizes the following ones:

- `""`
- `-`
- `?`
- `NA`
- `NaN`
- `NIL`
- `NULL`
- `N/A`
- `na`
- `null`
- `nil`
- `n/a`
- `#REF!`
- `#VALUE!`
- `#NULL!`
- `#NUM!`
- `#DIV/0`
- `#NAME?`
- `#N/A`

You can alter the list at your own convenience using the corresponding input.

6.6 Header

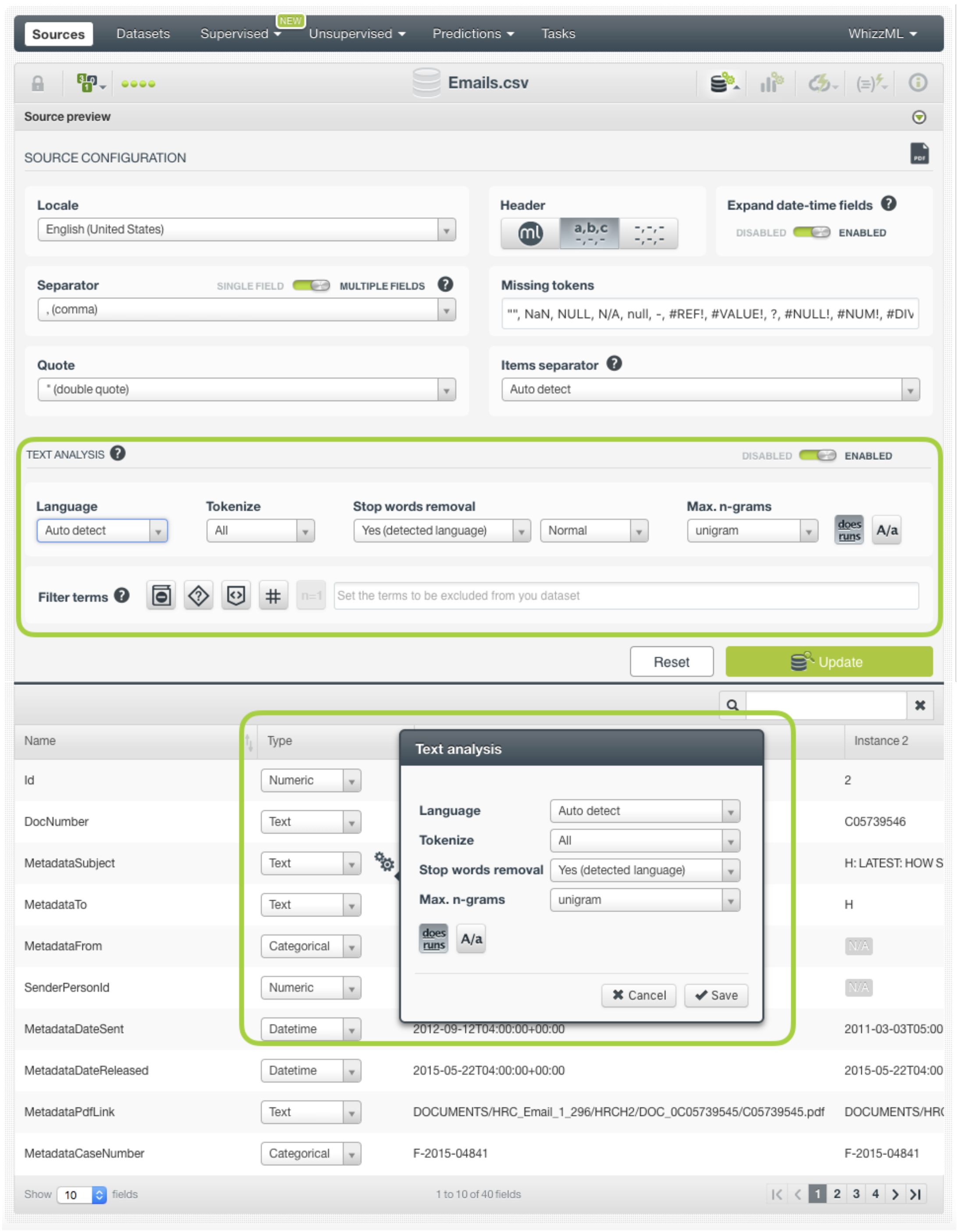
You can instruct BigML to parse the first line of your CSV file as a header (i.e., First row is header information) or not (i.e., Don't use the first row as header), or rely on BigML to auto-detect the presence of a header row (i.e., Smart header selection).

6.7 Expand Date-Time Fields

The **Expand date-time fields** toggles expansion of date-time fields into their numeric components. (See Section 5.3.)

6.8 Text Analysis

The TEXT ANALYSIS switch allows you to enable or disable analysis of text fields. The configuration options in this section are global for all the fields of your source, but you can also configure these options directly on individual text fields by overwriting the global configurations on a field-by-field basis. (See figure Figure 6.4.)



Locale: English (United States)
Header
Expand date-time fields: DISABLED ENABLED
Separator: , (comma) — SINGLE FIELD MULTIPLE FIELDS
Missing tokens: "", NaN, NULL, N/A, null, -, #REF!, #VALUE!, ?, #NULL!, #NUM!, #DIV
Quote: " (double quote)
Items separator: Auto detect

TEXT ANALYSIS — DISABLED ENABLED
Language: Auto detect
Tokenize: All
Stop words removal: Yes (detected language) Normal
Max. n-grams: unigram
Filter terms: Set the terms to be excluded from you dataset

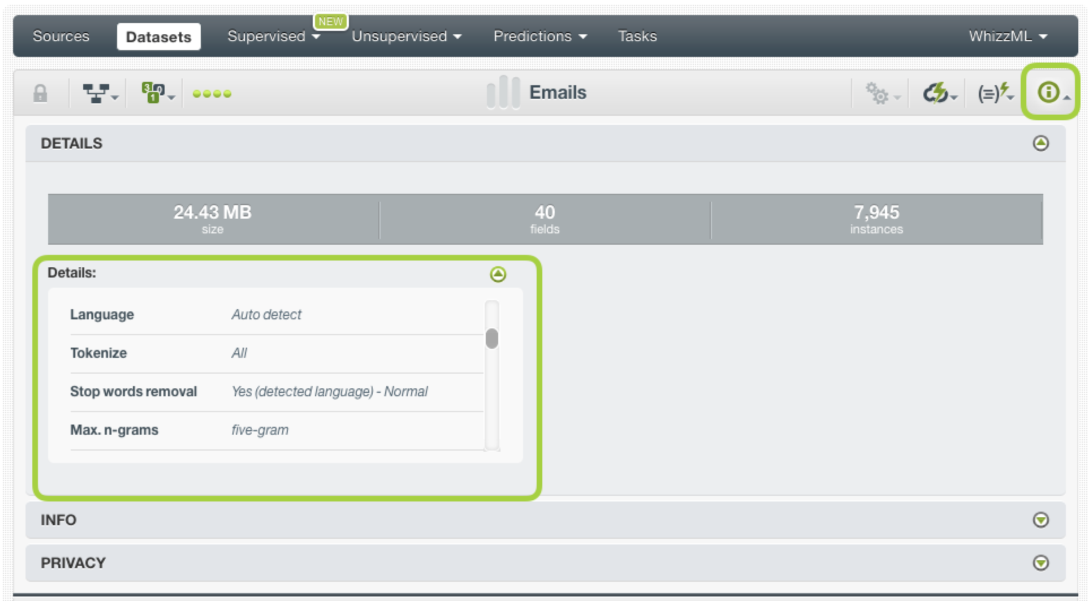
Text analysis
Language: Auto detect
Tokenize: All
Stop words removal: Yes (detected language)
Max. n-grams: unigram

Name	Type		Instance 2
Id	Numeric		2
DocNumber	Text		C05739546
MetadataSubject	Text		H: LATEST: HOW S
MetadataTo	Text		H
MetadataFrom	Categorical		N/A
SenderPersonId	Numeric		N/A
MetadataDateSent	Datetime	2012-09-12T04:00:00+00:00	2011-03-03T05:00
MetadataDateReleased	Datetime	2015-05-22T04:00:00+00:00	2015-05-22T04:00
MetadataPdfLink	Text	DOCUMENTS/HRC_Email_1_296/HRCH2/DOC_0C05739545/C05739545.pdf	DOCUMENTS/HR(
MetadataCaseNumber	Categorical	F-2015-04841	F-2015-04841

Show 10 fields — 1 to 10 of 40 fields — 1 2 3 4

Figure 6.4: Global and text fields configuration

The options configured at the source level will take effect when you create the dataset. You can see the text analysis options configured for a given dataset if you display the DETAILS in the INFO panel from the dataset view (see Figure 6.5). Since a dataset can have many text fields with different languages, you can find the information about which languages have been detected in the tooltip when you mouse hover the text optype green icon or in the tag cloud.



24.43 MB size | 40 fields | 7,945 instances

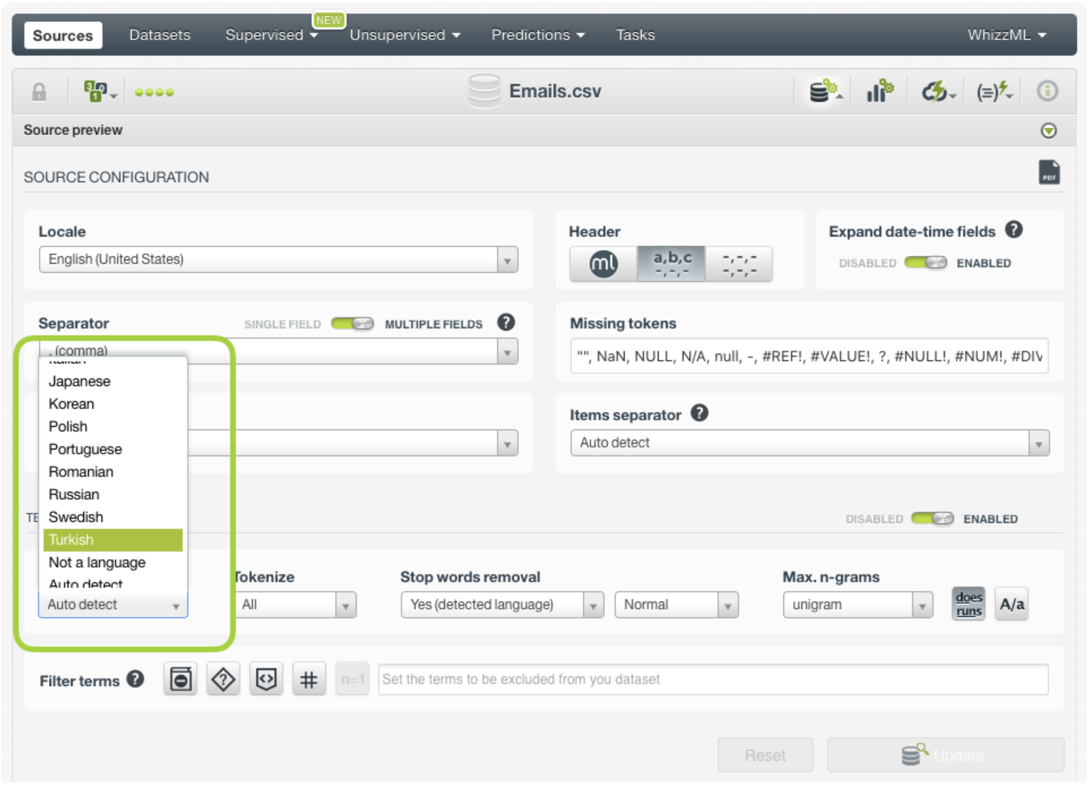
Details:

Language	*Auto detect*
Tokenize	*All*
Stop words removal	*Yes (detected language) - Normal*
Max. n-grams	*five-gram*

Figure 6.5: Text options configured for a given dataset

6.8.1 Language

BigML attempts to do basic language detection of each text field. You can choose any of the following languages at a global level or individual field level: **Arabic, Catalan, Chinese, Czech, Danish, Dutch, English, Farsi/Persian, Finish, French, German, Hungarian, Italian, Japanese, Korean, Polish, Portuguese, Turkish, Romanian, Russian, Spanish, and Swedish**.



Locale: English (United States)

Expand date-time fields: ENABLED

Separator: , (comma)

Missing tokens: "", NaN, NULL, N/A, null, -, #REF!, #VALUE!, ?, #NULL!, #NUM!, #DIV

Items separator: Auto detect

Language: Auto detect

Tokenize: All

Stop words removal: Yes (detected language) | Normal

Max. n-grams: unigram

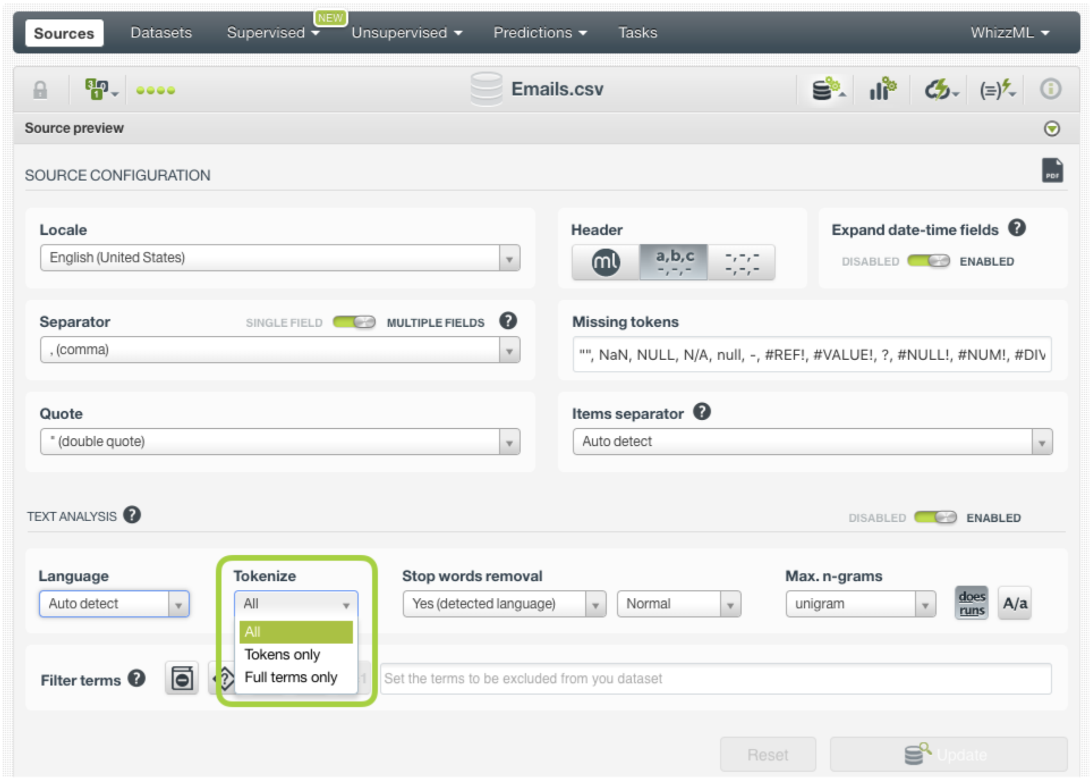
Filter terms: Set the terms to be excluded from you dataset

Figure 6.6: Language configuration options

6.8.2 Tokenize

Tokenization strategy allows splitting the text into several unique values. You can choose one of the following methods (default is “**All**”):

- **Tokens only**: individual words are used as terms. For example, “ML for all” becomes [“ML”, “for”, “all”].
- **Full terms only**: the entire field is treated as a single term as long as it is shorter than 256 characters. In this case “ML for all” stays [“ML for all”]
- **All**: both full terms and tokenized terms are used. In this case [“ML for all”] becomes [“ML”, “for”, “all”, “ML for all”].



Locale: English (United States)
Header
Expand date-time fields: DISABLED ENABLED
Separator: SINGLE FIELD MULTIPLE FIELDS , (comma)
Missing tokens: "", NaN, NULL, N/A, null, -, #REF!, #VALUE!, ?, #NULL!, #NUM!, #DIV
Quote: " (double quote)
Items separator: Auto detect
TEXT ANALYSIS: DISABLED ENABLED
Language: Auto detect
Tokenize: All (All, Tokens only, Full terms only)
Stop words removal: Yes (detected language) Normal
Max. n-grams: unigram
Filter terms: Set the terms to be excluded from you dataset

Figure 6.7: Tokenize configuration options

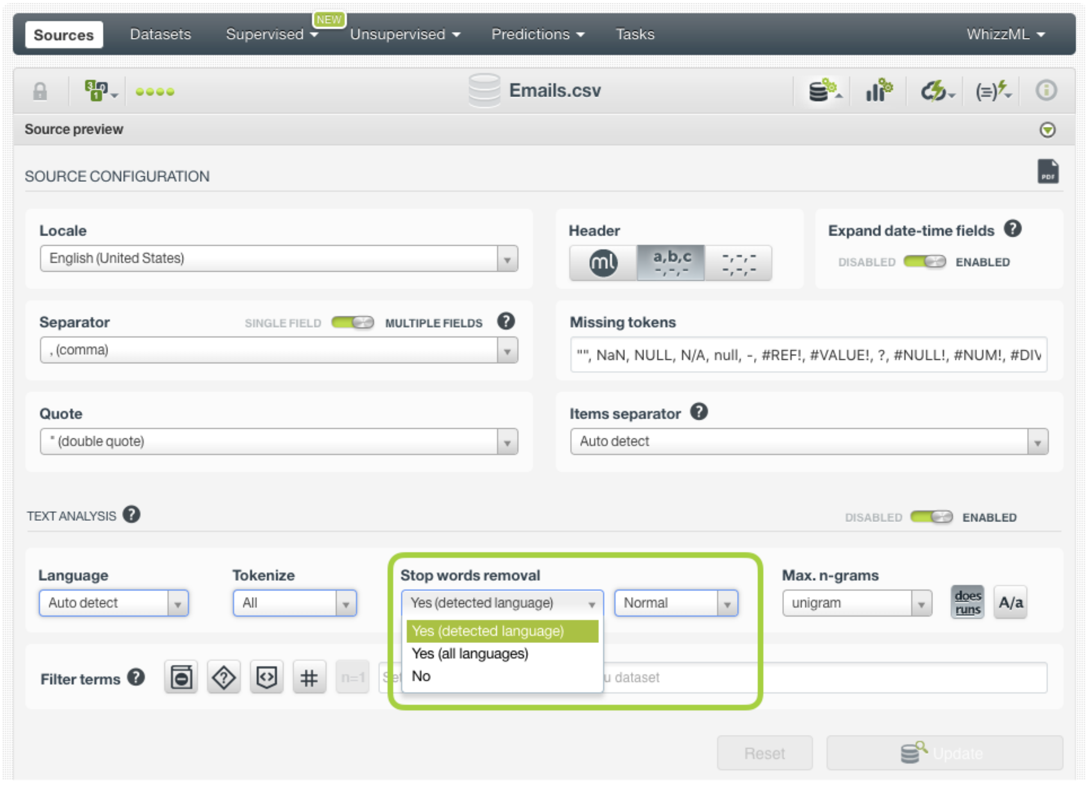
6.8.3 Stop Words Removal

The **Stop words removal** selector allows you to remove the use of usually uninformative stop words[4] as part of the text analysis. Some examples of stop words are: **a**, **the**, **is**, **at**, **on**, **which**, etc. Obviously, these change according to the language chosen to process each text field. This is the reason why BigML offers three options:

- **Yes (detected language)**: this option removes the stop words only for the detected language. If you have several languages mixed within the same field, the stop words of the non-detected languages will appear in your models. This is the option selected by default.
- **Yes (all languages)**: this option removes the stop words for all languages. Although you have several languages mixed within the same field, you will not find any stop words in your models. The downside is that some stop words for some languages may be valid words for other languages.
- **No**: this option will avoid the stop words removal. Therefore, the stop words will be included in your text analysis.

[4] https://en.wikipedia.org/wiki/Stop_words

Next to the **Stop words removal** selector you will find another selector that allows you to choose the aggressiveness of stopword removal where each level is a superset of words in the previous ones: **Light**, **Normal**, and **Aggressive**. By default, BigML performs **Normal** stop words removal.



SOURCE CONFIGURATION

Locale: English (United States)

Header

Expand date-time fields: DISABLED / ENABLED

Separator: , (comma) — SINGLE FIELD / MULTIPLE FIELDS

Missing tokens: "", NaN, NULL, N/A, null, -, #REF!, #VALUE!, ?, #NULL!, #NUM!, #DIV

Quote: " (double quote)

Items separator: Auto detect

TEXT ANALYSIS — DISABLED / ENABLED

Language: Auto detect

Tokenize: All

Stop words removal: Yes (detected language) — Normal

- Yes (detected language)
- Yes (all languages)
- No

Max. n-grams: unigram

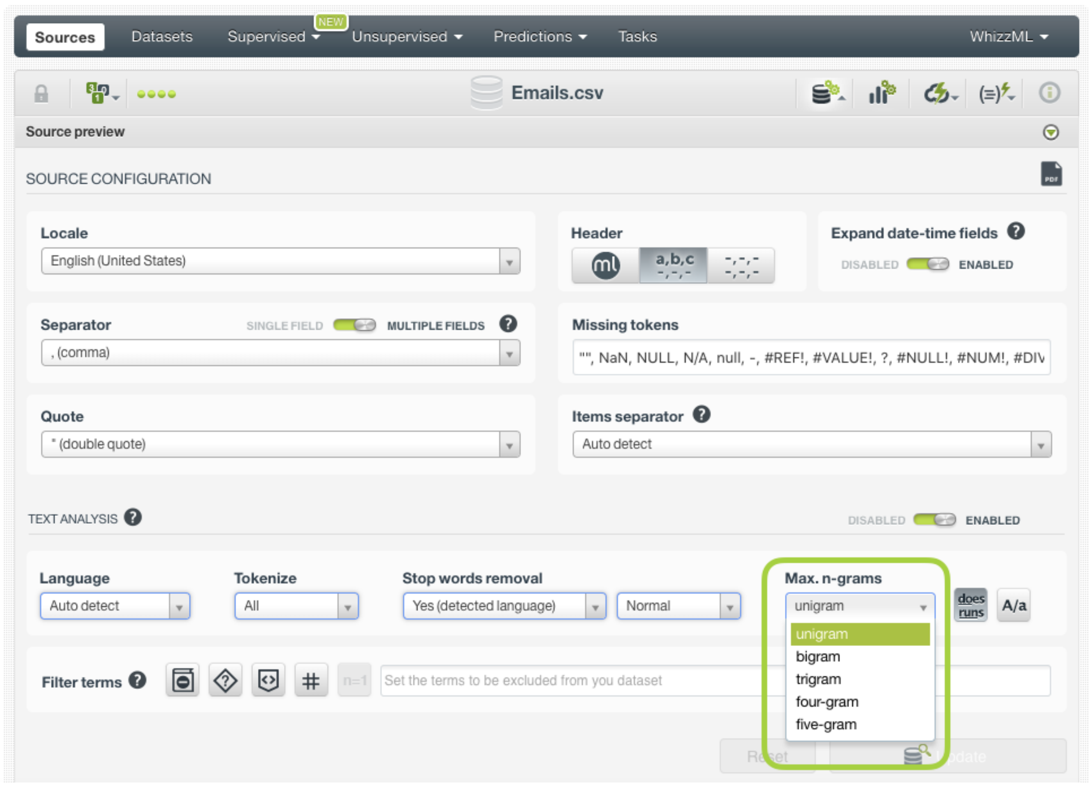
Filter terms

Figure 6.8: Stop words configuration options

6.8.4 Max. N-Grams

The **Max. n-grams** selector allows you to choose the maximum n-gram[5] size to consider for your text analysis. An n-gram is a frequent sequence of *n* terms found in the text. For example, "market" is a unigram (n-gram of size one), "prime minister" is a bigram (n-gram of size two), "Happy New Year" is a trigram (n-gram of size three), and so on. If you choose to keep stop words, they will be considered for the n-grams. You can select from unigrams up to five-grams.

[5] `https://en.wikipedia.org/wiki/N-gram`



Emails.csv

Source preview

SOURCE CONFIGURATION

Locale: English (United States)

Header: a,b,c

Expand date-time fields: DISABLED ENABLED

Separator: , (comma) — SINGLE FIELD MULTIPLE FIELDS

Missing tokens: "", NaN, NULL, N/A, null, -, #REF!, #VALUE!, ?, #NULL!, #NUM!, #DIV

Quote: " (double quote)

Items separator: Auto detect

TEXT ANALYSIS — DISABLED ENABLED

Language: Auto detect

Tokenize: All

Stop words removal: Yes (detected language) — Normal

Max. n-grams: unigram
- unigram
- bigram
- trigram
- four-gram
- five-gram

does runs — A/a

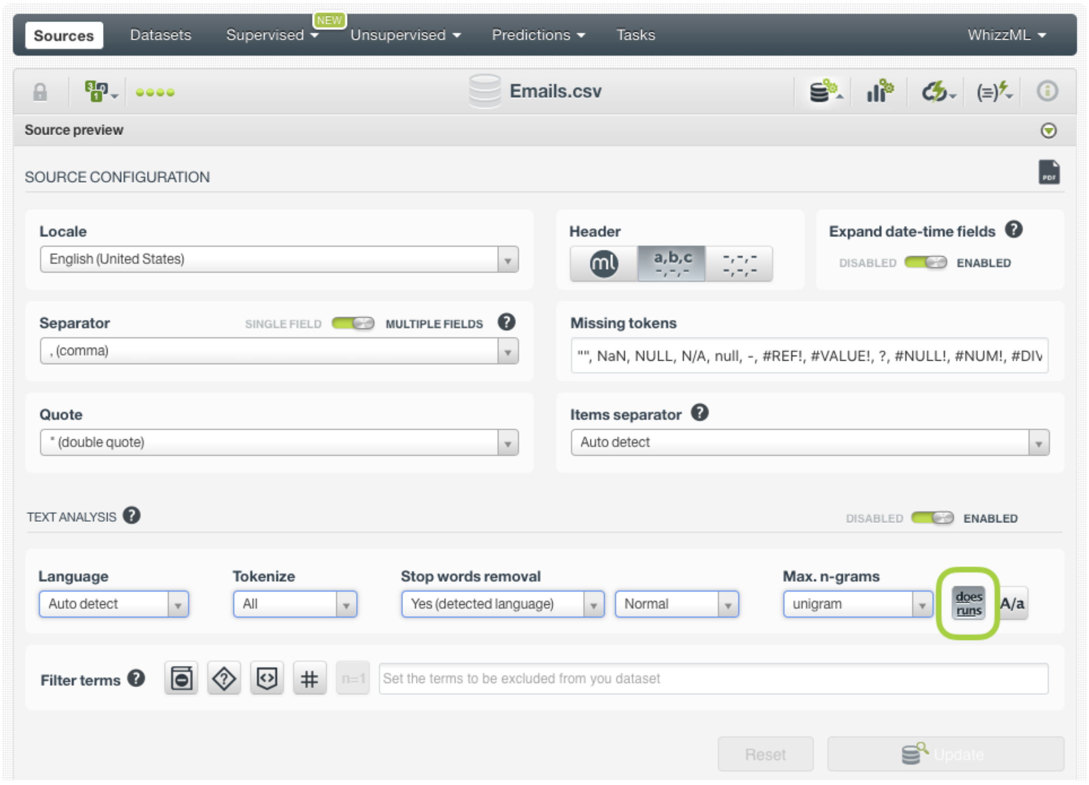
Filter terms: n=1 — Set the terms to be excluded from you dataset

Figure 6.9: n-grams configuration options

6.8.5 Stemming

BigML can differentiate all possible words or apply stemming[6], so words with the same root are considered one single value. For example, if `stemming` is enabled, the words great, greatly and greatness would be considered the same value instead of three different values. This option is enabled by default.

[6]https://en.wikipedia.org/wiki/Stemming



SOURCE CONFIGURATION

Locale: English (United States)

Header

Expand date-time fields: DISABLED ENABLED

Separator: SINGLE FIELD MULTIPLE FIELDS — , (comma)

Missing tokens: "", NaN, NULL, N/A, null, -, #REF!, #VALUE!, ?, #NULL!, #NUM!, #DIV

Quote: " (double quote)

Items separator: Auto detect

TEXT ANALYSIS: DISABLED ENABLED

Language: Auto detect

Tokenize: All

Stop words removal: Yes (detected language) — Normal

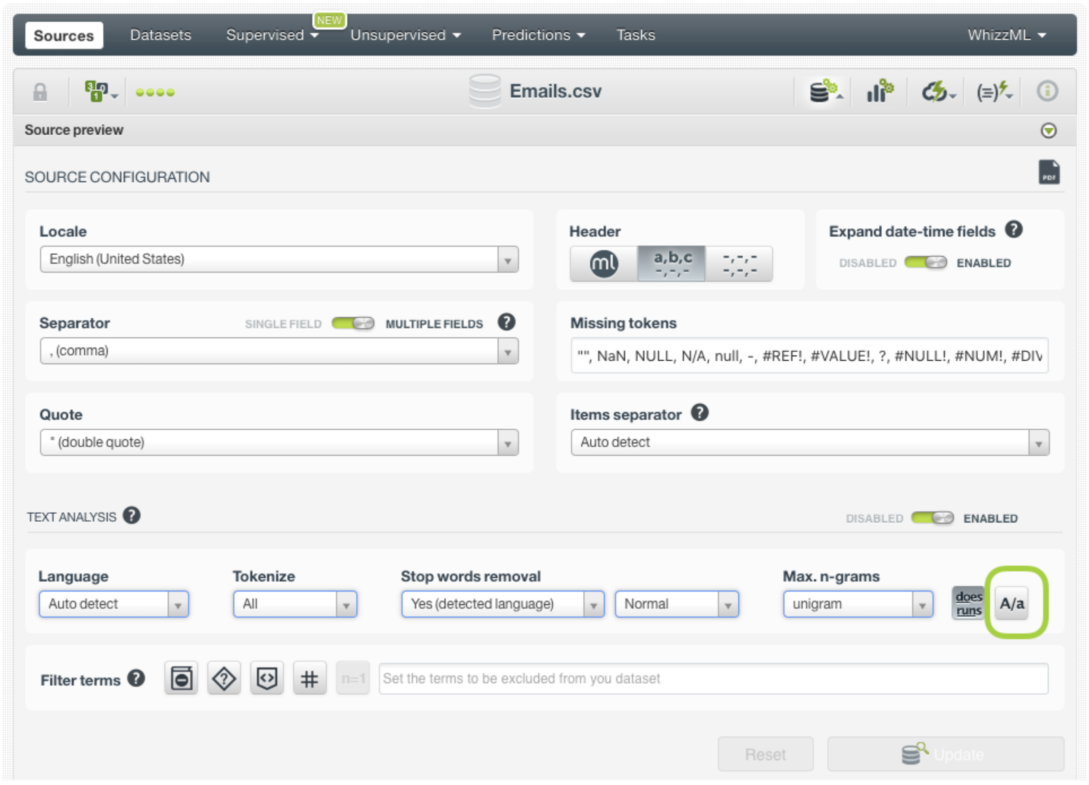
Max. n-grams: unigram

Filter terms: Set the terms to be excluded from you dataset

Figure 6.10: Stemming configuration

6.8.6 Case Sensitivity

Specify whether you want BigML to differentiate words if they contain upper or lower cases. If you click the case sensitivity option, terms with lower and upper cases will be differentiated, e.g., “House” and “house” will be considered two different terms. This option is inactive by default.



SOURCE CONFIGURATION

Locale: English (United States)

Header

Expand date-time fields: DISABLED ENABLED

Separator: SINGLE FIELD MULTIPLE FIELDS — , (comma)

Missing tokens: "", NaN, NULL, N/A, null, -, #REF!, #VALUE!, ?, #NULL!, #NUM!, #DIV

Quote: " (double quote)

Items separator: Auto detect

TEXT ANALYSIS: DISABLED ENABLED

Language: Auto detect

Tokenize: All

Stop words removal: Yes (detected language) — Normal

Max. n-grams: unigram

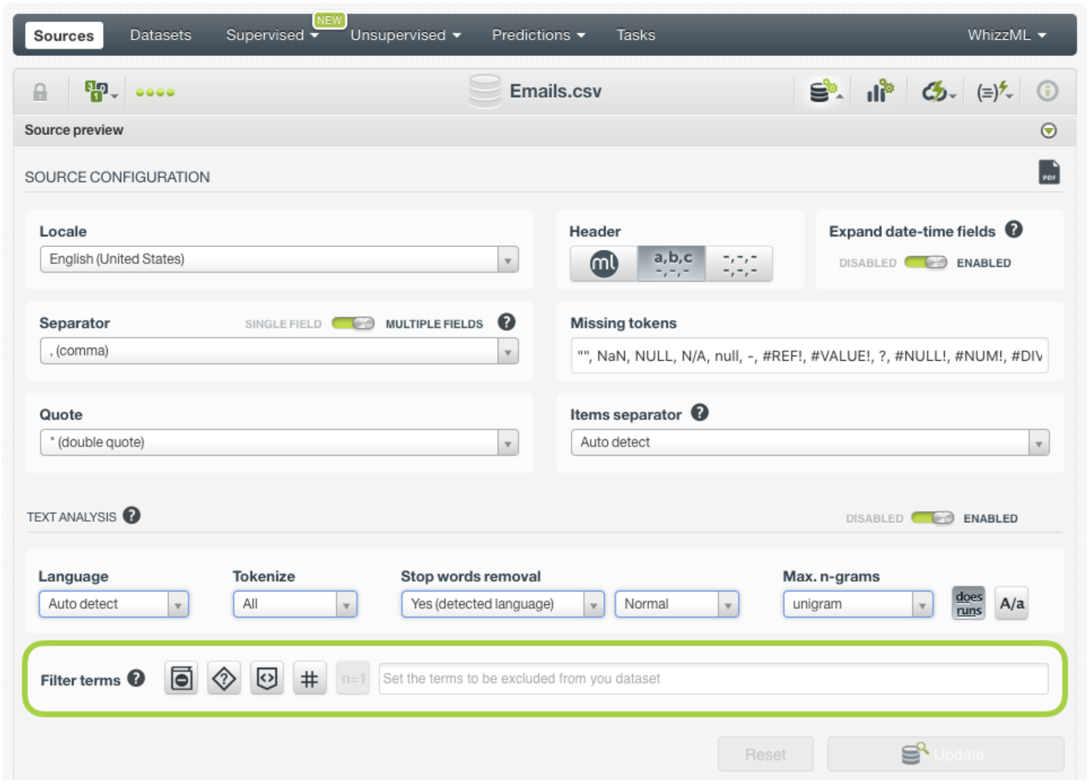
Filter terms: Set the terms to be excluded from you dataset

Figure 6.11: Case sensitivity configuration

6.8.7 Filter Terms

You can select to exclude certain terms from your text analysis. BigML provides the following otpions:

- **Non-dictionary words**: this option excludes terms that are unusual in the provided language. For this filter, BigML uses its own custom dictionaries that are composed of different sources such as online word lists, parses of Wikipedia, movie scripts, etc. These source may change depending on the language. The words in our dictionaries might contain terms like slang, abbreviations, proper names, etc. depending on whether or not these words are common enough to be found in our internet sources.
- **Non-language characters**: this option excludes terms containing uncommon characters for words in the provided language. For example, if the language is Russian, all terms containing non-Cyrillic characters will be filtered out. Numeric digits will be considered non-language characters regardless of language.
- **HTML keywords**: this option excludes JavaScript/HTML keywords commonly seen in HTML documents.
- **Numeric digits**: this option excludes any term that contains a numeric digit in [0-9].
- **Single tokens**: this option excludes terms that contain only a single token, i.e., unigrams. Only bigrams, trigrams, four-grams, five-grams and/or full terms will be considered (at least one of these options needs to be selected, otherwise the single token filter will be disabled).
- **Specific terms**: this is a free text option where you can write any term or group of terms to be excluded from your text analysis.



SOURCE CONFIGURATION

Locale: English (United States)

Header

Expand date-time fields: DISABLED ENABLED

Separator: , (comma) — SINGLE FIELD MULTIPLE FIELDS

Missing tokens: "", NaN, NULL, N/A, null, -, #REF!, #VALUE!, ?, #NULL!, #NUM!, #DIV

Quote: " (double quote)

Items separator: Auto detect

TEXT ANALYSIS: DISABLED ENABLED

Language: Auto detect

Tokenize: All

Stop words removal: Yes (detected language) — Normal

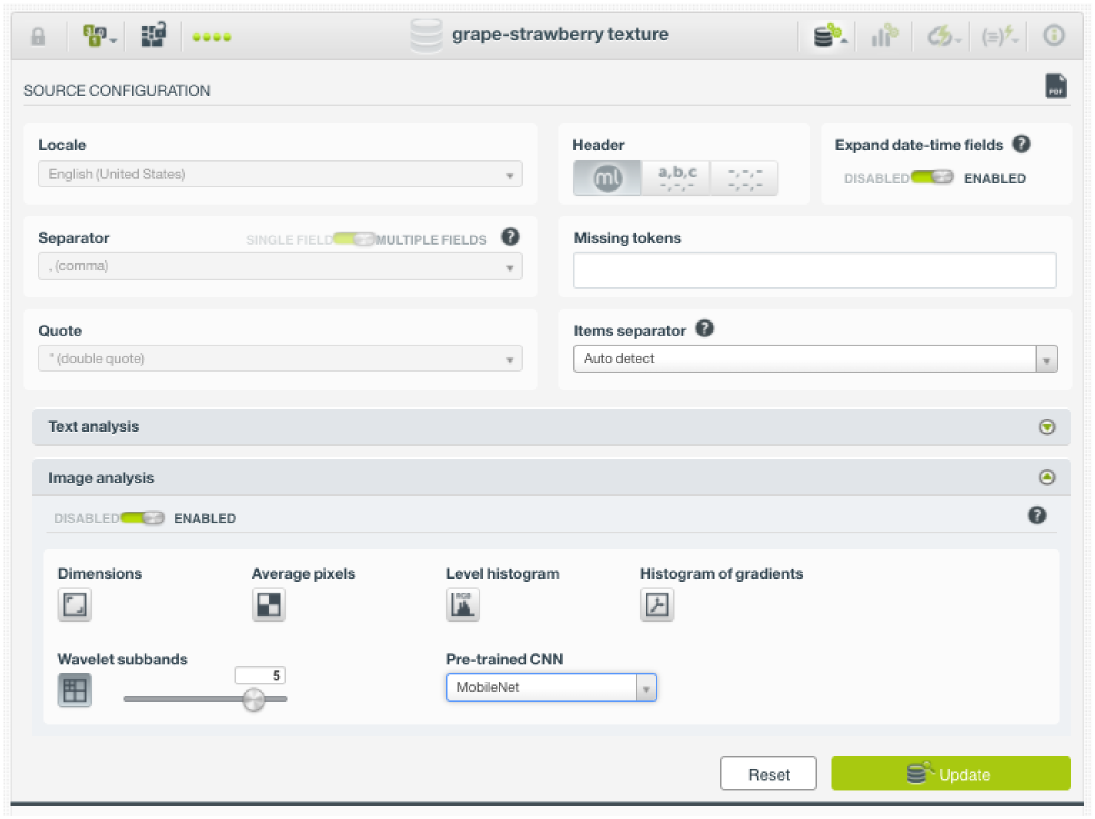
Max. n-grams: unigram

Filter terms: Set the terms to be excluded from you dataset

Figure 6.12: Filter terms

6.9 Image Analysis

The IMAGE ANALYSIS panel allows users to enable or disable extracted features of images, and allows them to configure different sets of the features.



grape-strawberry texture

SOURCE CONFIGURATION

Locale: English (United States)

Header

Expand date-time fields: DISABLED ENABLED

Separator: SINGLE FIELD MULTIPLE FIELDS
, (comma)

Missing tokens

Quote: " (double quote)

Items separator: Auto detect

Text analysis

Image analysis

DISABLED ENABLED

Dimensions

Average pixels

Level histogram

Histogram of gradients

Wavelet subbands: 5

Pre-trained CNN: MobileNet

Figure 6.13: Image analysis panel

The first control is the DISABLED/ENABLED switch. If disabled, there won't be any image features generated.

If enabled, users can configure to have any combination of the following five sets of image features:

- **Dimensions**: This gives four values, corresponding to the raw image file size, pixel width, pixel height and aspect ratio. 4 numeric fields.
- **Average pixels**: This gives the red, blue, and green pixel values for several extremely low resolution versions of the image (1x1, 3x3 and 4x4). This is fast to calculate and captures the high-level spatial and color information, but all detail is lost. 78 numeric fields.
- **Level histogram**: This gives the color information in each channel divided into 16 equally spaced histogram bins. Each color histogram is normalized so all values are between 0 and 1. While this gives very detailed color intensity information, all spatial information is lost. 48 numeric fields.
- **Histogram of gradients**: Computes a histogram of oriented gradients for the entire image, and for all subimages on a 3x3 and 4x4 grid. The histograms are normalized within each subimage, so all values are between 0 and 1. This histogram generally captures useful spatial and detail information, but precise color information is lost. Generally, this extractor is good at classifiying different shapes, or images where the orientation of the edges is a defining characteristic. 234 numeric fields.
- **Wavelet subbands**: Performs an n level Haar wavelet decomposition on the image, where n is a parameter. This parameter determines the number of recursive compositions that the featurizer will undertake, and so determines the number of output features. After decomposition, the pixels in each subband are aggregated using mean and standard deviation for both the full images and

a 2x2 grid. Since each subband contains all image detail at a certain resolution in one of three directions, this feature type contains both spatial and frequency domain information about the nature of the detail in the image, but the directionality of the detail is only coarsely captured (contrast histogram_of_gradients). Typically useful for problems where texture is a defining characteristic of the image, or where there is obvious periodicity. 160 numeric fields.

Users can also configure to have one of the five pre-trained CNNs (Convolutional Neural Networks):

- **MobileNet**: 1024 numeric fields.
- **MoibleNetV2**: 1280 numeric fields.
- **ResNet-18**: 512 numeric fields.
- **ResNet-50**: 2048 numeric fields.
- **Xception**: 2048 numeric fields.

Note: ResNet-50 and Xception are only available to customers of virtual private clouds and private deployments

Each option uses the top layer before the softmax output of an ImageNet pre-trained CNN as the input features. These features generally capture high-level features useful for real-world object classification (the presence of eyes, wheels, or striped fur, for example). While these features are easily the best for natural image classification, poor capture conditions and artificial domains (handwriting, images of documents, low resolution security video, etc.) can make these features unsuitable.

6.10 Items Separator

You can select the specific **separator** that will be used by **items fields**. By default, BigML tries to auto-detect it. If the BigML selection is incorrect, you can select one of the predefined defaults or you can input another one (see Figure 6.14).

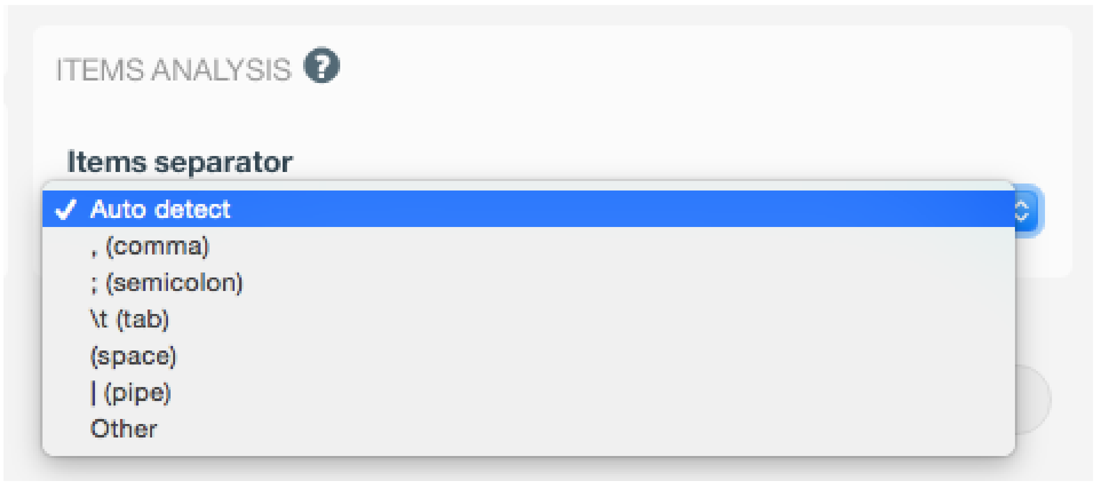
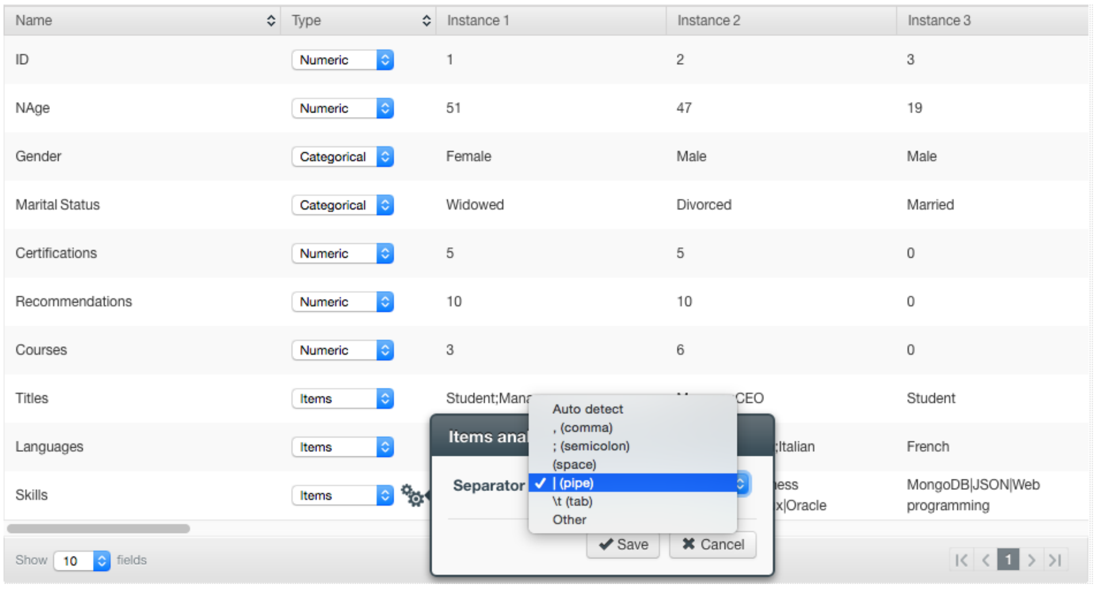


Figure 6.14: Items separator selection

A source can have multiple fields of type items and each one can have a different separator. Once you open a source configuration panel for those fields that are of type items, a configuration icon will allow you to select the specific separator for that field. (See Figure 6.15.)



Name	Type	Instance 1	Instance 2	Instance 3
ID	Numeric	1	2	3
NAge	Numeric	51	47	19
Gender	Categorical	Female	Male	Male
Marital Status	Categorical	Widowed	Divorced	Married
Certifications	Numeric	5	5	0
Recommendations	Numeric	10	10	0
Courses	Numeric	3	6	0
Titles	Items	Student;Mana	CEO	Student
Languages	Items		;Italian	French
Skills	Items		ess x|Oracle	MongoDB|JSON|Web programming

Items ana

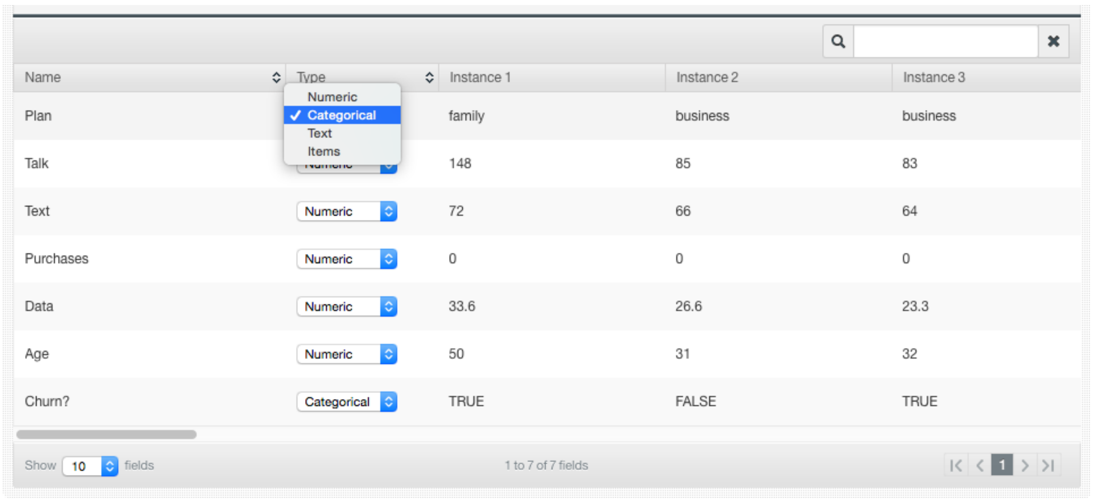
Separator [✓ | (pipe)]

Show 10 fields

Figure 6.15: Separator selector for an items field

6.11 Updating Field Types

The type of each field can be updated individually using the **Configure source** panel and then selecting the new type for each field using the selector provided for each field. (See Figure 6.16.) Text, items and date-time fields also offer additional specific configurations.



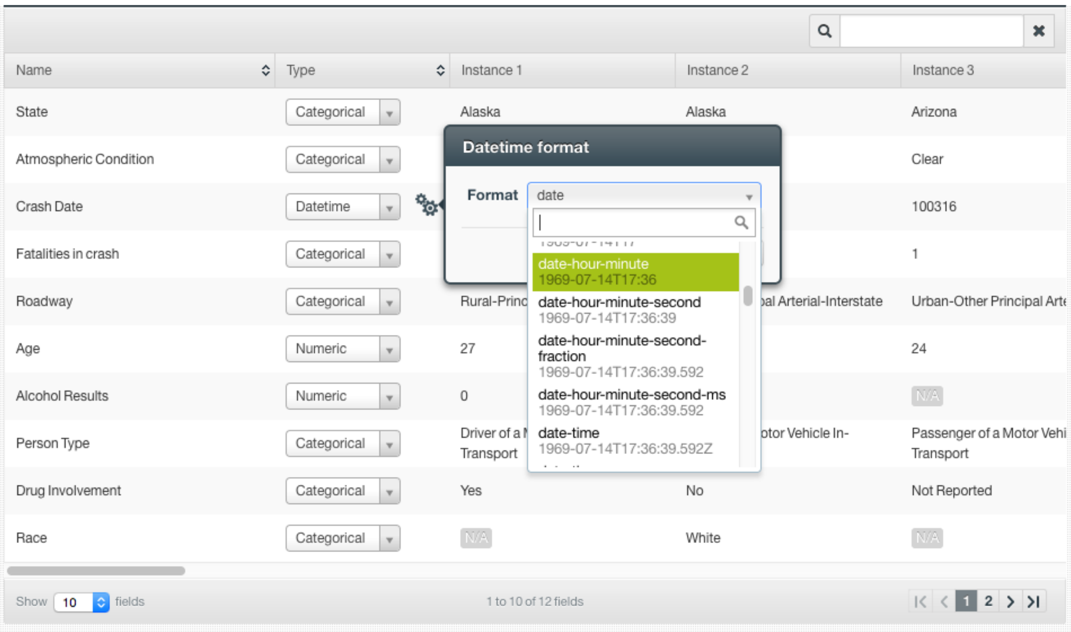
Name	Type	Instance 1	Instance 2	Instance 3
Plan	Categorical	family	business	business
Talk	Numeric	148	85	83
Text	Numeric	72	66	64
Purchases	Numeric	0	0	0
Data	Numeric	33.6	26.6	23.3
Age	Numeric	50	31	32
Churn?	Categorical	TRUE	FALSE	TRUE

Show 10 fields 1 to 7 of 7 fields

Figure 6.16: Individual selector to change the type of each field

6.11.1 Date-Time Formats Configuration

In the case of **date-time** fields, it might happen that BigML is not able to determine the right format. In that case, you can select the specific format of your fields by clicking in the configuration icon shown in Figure 6.17. You can choose any of the pre-defined formats included in Table 5.1 among the selector options.



Name	Type	Instance 1	Instance 2	Instance 3
State	Categorical	Alaska	Alaska	Arizona
Atmospheric Condition	Categorical			Clear
Crash Date	Datetime			100316
Fatalities in crash	Categorical			1
Roadway	Categorical	Rural-Princ	al Arterial-Interstate	Urban-Other Principal Arte
Age	Numeric	27		24
Alcohol Results	Numeric	0		N/A
Person Type	Categorical	Driver of a I Transport	otor Vehicle In-	Passenger of a Motor Vehi Transport
Drug Involvement	Categorical	Yes	No	Not Reported
Race	Categorical	N/A	White	N/A

Show 10 fields 1 to 10 of 12 fields 1 2

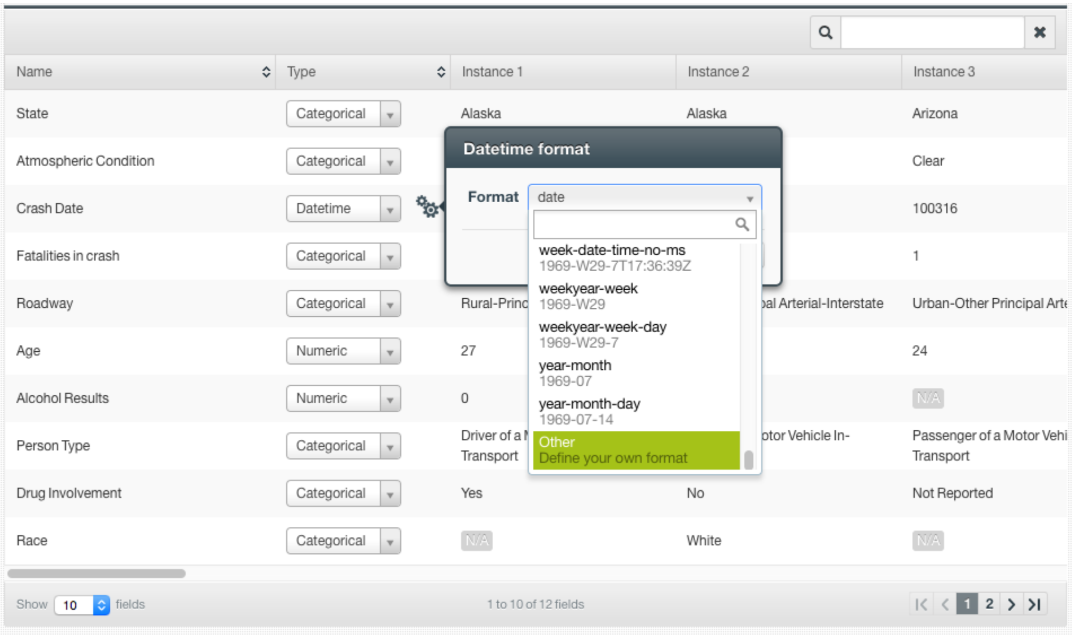
Datetime format

Format: date

- date-hour-minute 1969-07-14T17:36
- date-hour-minute-second 1969-07-14T17:36:39
- date-hour-minute-second-fraction 1969-07-14T17:36:39.592
- date-hour-minute-second-ms 1969-07-14T17:36:39.592
- date-time 1969-07-14T17:36:39.592Z

Figure 6.17: Configure the date-time fields format

If you do not find the format of your date-time field in the pre-defined options you can also configure your own format using the option “Other”. (See Figure 6.18.)



Name	Type	Instance 1	Instance 2	Instance 3
State	Categorical	Alaska	Alaska	Arizona
Atmospheric Condition	Categorical			Clear
Crash Date	Datetime			100316
Fatalities in crash	Categorical			1
Roadway	Categorical	Rural-Princ	al Arterial-Interstate	Urban-Other Principal Arte
Age	Numeric	27		24
Alcohol Results	Numeric	0		N/A
Person Type	Categorical	Driver of a I Transport	otor Vehicle In-	Passenger of a Motor Vehi Transport
Drug Involvement	Categorical	Yes	No	Not Reported
Race	Categorical	N/A	White	N/A

Show 10 fields 1 to 10 of 12 fields 1 2

Datetime format

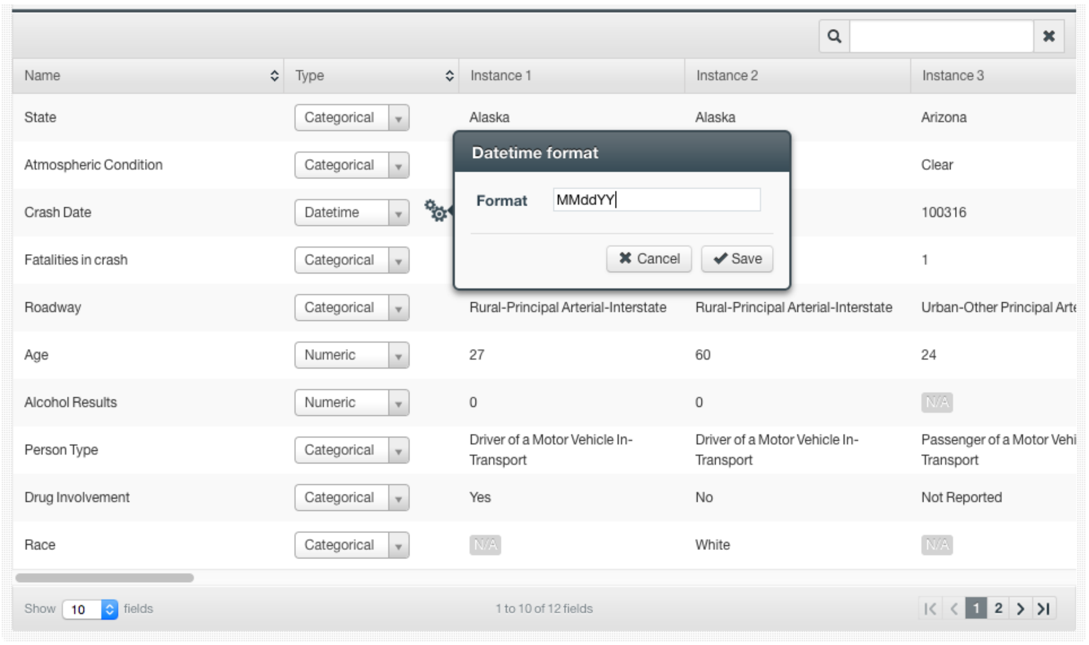
Format: date

- week-date-time-no-ms 1969-W29-7T17:36:39Z
- weekyear-week 1969-W29
- weekyear-week-day 1969-W29-7
- year-month 1969-07
- year-month-day 1969-07-14
- Other Define your own format

Figure 6.18: Configure custom date-time formats

This custom option allows you to input any string using the Java DateTimeFormatter specification[7] for date-time patterns. For example, for month and year you need to use the upper-case letters `MM` and `YY`, while for day you need to use the lower-case letters `dd`. See an example of a custom date format in Figure 6.18 where the date is written as `MMddYY`, i.e., 100314 meaning 3rd of October 2014.

[7] https://docs.oracle.com/javase/10/docs/api/java/time/format/DateTimeFormatter.html



Name	Type	Instance 1	Instance 2	Instance 3
State	Categorical	Alaska	Alaska	Arizona
Atmospheric Condition	Categorical			Clear
Crash Date	Datetime			100316
Fatalities in crash	Categorical			1
Roadway	Categorical	Rural-Principal Arterial-Interstate	Rural-Principal Arterial-Interstate	Urban-Other Principal Arte
Age	Numeric	27	60	24
Alcohol Results	Numeric	0	0	N/A
Person Type	Categorical	Driver of a Motor Vehicle In-Transport	Driver of a Motor Vehicle In-Transport	Passenger of a Motor Vehi Transport
Drug Involvement	Categorical	Yes	No	Not Reported
Race	Categorical	N/A	White	N/A

Datetime format

Format: MMddYY

Show 10 fields — 1 to 10 of 12 fields

Figure 6.19: Custom date format example

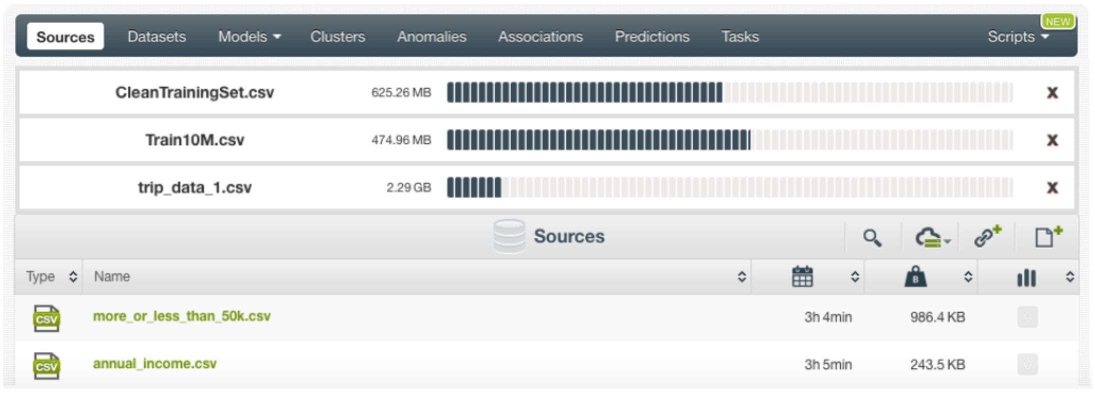
CHAPTER 7

Local Sources

The easiest way to create a new source in BigML is to drag a file that follows one of the formats described in Section 1.2 and drop it on top of the BigML Dashboard.

BigML allows you to upload up to ten files in parallel. For each file, BigML will display a progress bar that indicates how far along the uploading process is. You can navigate to other parts of the BigML Dashboard or initiate other tasks while you upload new sources to BigML. You can also stop every individual upload by clicking on the **X** on the right side of each progress bar. (See Figure 7.1.)

You can also use the upload source button (see Figure 7.2) that is available in the **source list view** to upload a new source. This will open a **Open File Dialog Box** that will allow you to navigate through your local file system.



File	Size
CleanTrainingSet.csv	625.26 MB
Train10M.csv	474.96 MB
trip_data_1.csv	2.29 GB

Type	Name	Age	Size
CSV	more_or_less_than_50k.csv	3h 4min	986.4 KB
CSV	annual_income.csv	3h 5min	243.5 KB

Figure 7.1: Progress bars

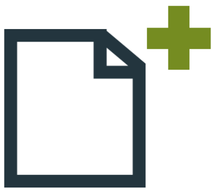


Figure 7.2: Button to create a local sources

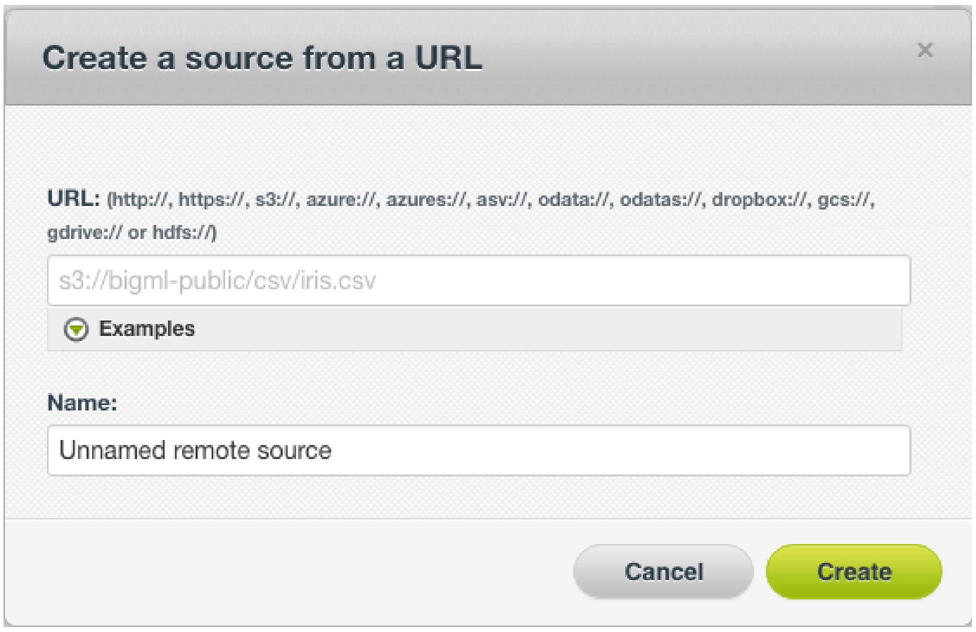
CHAPTER 8

Remote Sources

Sources can also be created using remote data as opposed to the files on local storage. You can connect to your databases and create Sources directly from them (Section 8.8). You can use your cloud storages (Section 8.7). You can also create Sources by using URLs that point to external source files. BigML will use the URL to download the data and create a local copy.

Figure 8.1: Button to upload remote sources

In the **source list view**, you will find the remote source button (see Figure 8.1) that will open a new modal (see Figure 8.2) window, where you can specify the URL and also give a name to the new remote source. URLs must follow one of the accepted protocols described in Section 8.1.



Create a source from a URL

URL: (http://, https://, s3://, azure://, azures://, asv://, odata://, odatas://, dropbox://, gcs://, gdrive:// or hdfs://)

s3://bigml-public/csv/iris.csv

Examples

Name:

Unnamed remote source

Figure 8.2: Modal window to create a remote source using a URL

8.1 Accepted Protocols

The list of accepted protocols to create remote sources is displayed on Table 8.1. The following subsections detail each of the stores BigML can communicate with.

Schema	Description
asv://	Same as azure://
asvs://	Same as azures://
azure://	Microsoft Azure storage
azures://	Same as azure:// but using SSL[1]
drobox://	Drobox-stored files
gcs://	Google Cloud stores
gdrive://	Google Drive files
hdfs://	The distributed storage used by Hadoop applications
http://	Regular HTTP-accesible files
https://	HTTP secure-accessible files
odata://	Open Data Protocol[2] that consumes REST APIs
odatas://	Same as odata:// but using SSL
s3://	Simple Storage Service[3] (**S3**), the file storage provided by Amazon Web Services (**AWS**)

Table 8.1: Protocols recognized by BigML

8.2 HTTP(S) Stores

Regular HTTP and HTTPS links can be used as the URI of remote sources:

```
http://bigml.com/test/data.csv
https://bigml.com/test/data.csv
```

Figure 8.3: Example of HTTP and HTTPS remote sources

By default, BigML does not perform any certificate validation for HTTPS links, but you can ask for it using the query string parameter **validate**, as in this example:

```
https://bigml.com/test/data.csv?validate=true
```

Figure 8.4: Example of an HTTPS remote source requesting validation

8.3 Azure Stores

BigML can retrieve sources directly from Azure as block or page blobs. The URLs take the following forms:

```
azure://<container>/<blob>?AccountKey=<key>&AccountName=<storage account>
azures://<container>/<blob>?AccountKey=<key>&AccountName=<storage account>
```

Figure 8.5: Azure URLs templates to create remote sources

The **azures** variant asks for HTTPS, instead of HTTP, for the end point protocol. You can also use **asv** and **asvs** instead of **azure** and **azures**, respectively.

The **AccountKey** parameter is unnecessary for public blobs; in addition, one can add the following parameters:

- **DefaultEndpointsProtocol** either **http** or **https** overrides the one implied by the URI scheme.
- **BlobEndPoint** for blobs that use their own domain names instead of Azure's default blob.core.windows.net.
- **SharedAccessSignature** for shared containers, in which case the account credentials will be ignored.

Finally, if using the default end points, the URL can be specified as the blob's REST URL:

```
http://<account name>.blob.core.windows.net/<container>/<blob>?AccountKey=...
```

Figure 8.6: Azure Blob REST URL

Having the same parameters as above except that the account name is now part of the URL. HTTPS URLs of the same form are also recognized as Azure blobs.

8.4 HDFS

BigML also allows you to access to files stored using HDFS[4], the primary distributed storage used by Hadoop applications. HDFS remote sources follow this template:

```
hdfs://host:port/path/hdfs/file.csv
```

Figure 8.7: Template of HDFS remote sources

8.5 OData

Remote sources can specify an OData URI as its source, accessible either by HTTP or HTTPS, by using the **odata** or **odatas** scheme. For instance, the URI in Figure 8.8 will request BigML to access the table Customers in the OData root http://services.odata.org/Northwind/Northwind.svc.

```
odata://services.odata.org/Northwind/Northwind.svc/Customers
```

Figure 8.8: Example of an Odata remote source

[4] https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html

You can use any OData URL parameter to construct the result set (BigML will just use the given URL as is, specifying to the OData service that it wants its result in JSON format), as long as the answer to the query contains a list of results (i.e., an entity set, or OData "table" or "view"). To select only the first 100 rows of the above source, and only the City and PostalColumns you could write:

```
odata://services.odata.org/Northwind/Northwind.svc/Customers?$top=100&$select=City,PostalColumns
```

Figure 8.9: Example of an Odata remote source with parameters

BigML also accepts the abbreviations **od:**// and **ods:**// for **odata:**// and **odatas:**//, respectively.

Only columns with atomic (number, string, boolean) values are imported by BigML. For any inner field in a composite value to be part of the source, just construct the appropriate query with the URL parameters.

For more information about OData URIs, see OData URI conventions[5].

As a special case, BigML recognizes Azure Marketplace HTTPS URLs with hostname api.datamarket.azure.com as OData stores. Create a remote source using the URL displayed in Figure 8.10, and it will be treated as if it were the canonical form shown in Figure 8.11.

```
https://api.datamarket.azure.com/www.bcn.cat/BarcelonaFacilities/v1/EquipamentsBCNRefreshed
```

Figure 8.10: Example of Azure Data Market remote source using HTTPS

```
odatas://api.datamarket.azure.com/www.bcn.cat/BarcelonaFacilities/v1/EquipamentsBCNRefreshed
```

Figure 8.11: Example of Azure Data Market remote source using odatas

BigML provides support for Azure Data Market entities protected by an account **id** and account **key**, which must be provided as the query string parameters **AccountId** and **AccountKey**, as shown in Figure 8.12.

```
ods://api.datamarket.azure.com/Data/v1/E?AccountId=adfsf&AccountKey=edj/2+
```

Figure 8.12: Example of a protected Azure Data Market Remote Source

As always, you can also use **odata** or **https** for the schema.

8.6 S3 Stores

Source files stored in Amazon Simple Storage Storage (**S3**) can be specified using URLs of the form shown in Figure 8.13.

[5] http://www.odata.org/documentation/odata-version-2-0/uri-conventions/

```
s3://bucket/path/identifier?access-key=key0&secret-key=key1
```

Figure 8.13: Template of an S3 Remote Source

The two keys **access-key** and **secret-key** are optional. **BigML Private Deployments** will use default values read from its configuration, either in the s3 section of the configuration file or as the CLI parameters –s3–access-key and –s3–secret-key. Keys present in a URL always override those defaults.

8.7 Cloud Storages

You can create BigML Sources by downloading data from your cloud storages. Because of the popularity of cloud storages, BigML gives users the ability to configure their cloud storages on the Dashboard.

8.7.1 Configuring Cloud Storages

BigML allows you to configure the following cloud storage providers at https://bigml.com/account/cloudstorages (see Figure 8.14):

- Google Cloud Storage
- Google Drive
- Dropbox
- Microsoft Azure Marketplace

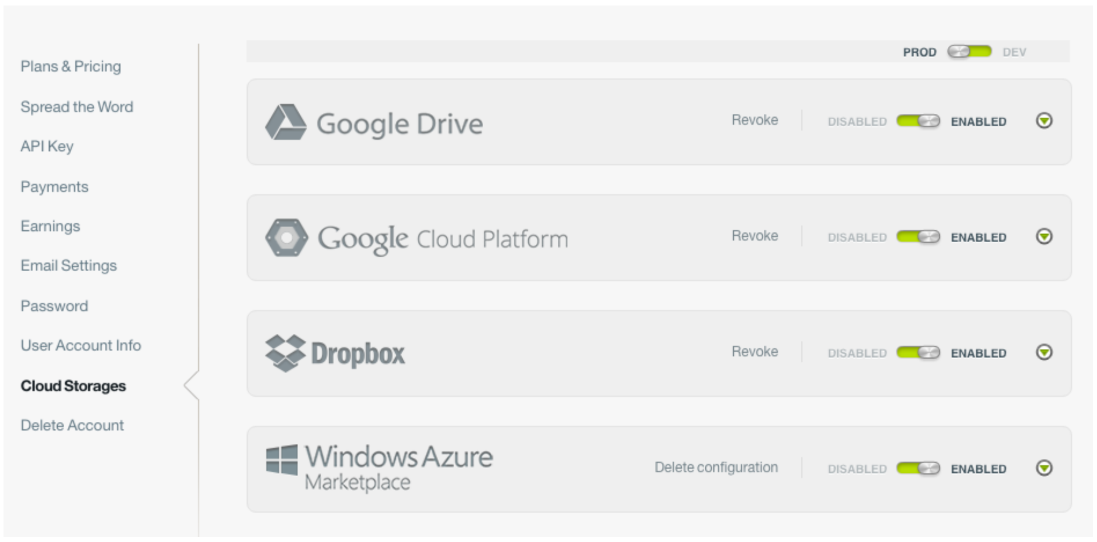


Figure 8.14: Configuration Panel of Cloud Storages

If you enable cloud storage providers, you will have a new menu option in the listing source view where you can use a widget to navigate through those storages and locate your source. (See Figure 8.15.)

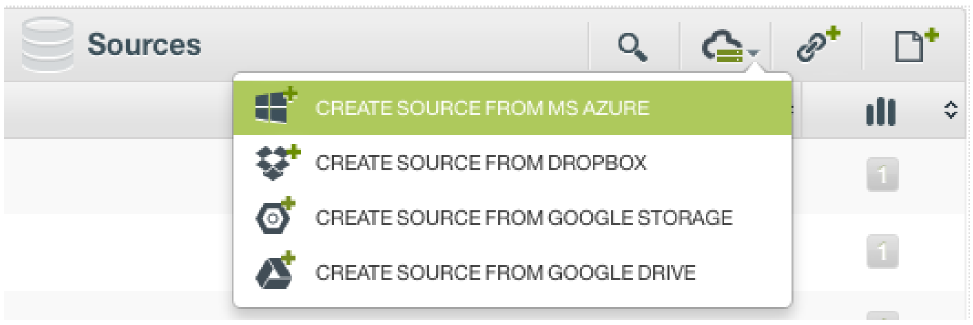

Sources
CREATE SOURCE FROM MS AZURE
CREATE SOURCE FROM DROPBOX
CREATE SOURCE FROM GOOGLE STORAGE
CREATE SOURCE FROM GOOGLE DRIVE

Figure 8.15: Menu options to create a source from cloud storages

To use any of those cloud storage providers, you need to first grant BigML access to it or provide your credentials. You can revoke the access or disable the new menu options at any time.

8.7.2 Dropbox

Given the OAuth token for a Dropbox file, request its download as a source via the Dropbox scheme, providing the token in the query string, without host:

```
dropbox:/path/to/file.csv?token=adfwdfda_weke23423_fheh324sxke33
```

Figure 8.16: Dropbox URL template

For instance, for the file **iris.csv** at the root of your Dropbox you could use:

```
dropbox:/iris.csv?token=adfwdfda_weke23423_fheh324sxke33
```

Figure 8.17: Example of a Dropbox URL

For the same file inside a **csv** folder the correct URI would be:

```
dropbox:/csv/iris.csv?token=adfwdfda_weke23423_fheh324sxke33
```

Figure 8.18: Example of a Dropbox URL using a folder in the path

8.7.3 Google Cloud Storage

Remote sources can use the **gcs** schema to specify any file stored in a Google Cloud Storage bucket. For publicly shared files, no other parameter is needed, e.g., if **iris.csv** is in the folder **customerdata** of the **bigml** bucket use:

```
gcs://bigml/customerdata/iris.csv
```

Figure 8.19: Example of a Google Cloud Storage URL

If the file is protected and you have an OAuth2 access token which has not yet expired, specify it via the token query string parameter:

```
gcs://bigml/test.csv?token=ya29.ygCrfy3xq1Bg5eIPMlIPUUqzEvOnC0kIXPdI
```

Figure 8.20: Example of a Google Cloud Storage URL using OAuth2

In addition, if you also have a refresh token, and your client identifier and application secret, they can be specified together with the token using the additional query string parameters **refresh-token**, **client-id** and **app-secret**, respectively, and BigML will take care of refreshing the possibly expired token as needed.

8.7.4 Google Drive

Remote sources using the **gdrive** protocol refer to files stored in **Google Drive** (GDrive). The full URI does not use a host, so it usually starts with **gdrive:**///, and its only path component refers to the required file's **file-id**, as provided by the Google Drive service.

GDrive files are granted access via OAuth2, so you also need a client ID, app secret, a token, and refresh token to access the file. Generally, a **gdrive** URI looks like:

```
gdrive:///<file-id>?token=<..>&refresh-token=<..>&app-secret=<..>&client-id=<...>
```

Figure 8.21: Template of a Google Drive URL

For example:

```
gdrive:///0BxGbAMhJezOScTFBUVFPMy1xT1E?token=ya29.AQHpyxUssLrU7Gy4oEsUjqyV
mPJSPDuZKSc_ze3_Q8_14miBDJPfOxnqkGC2vPH01savQVGt7oqSg-w&refresh-token=
1/x6zd8Wjy__yk437S7AxZ5Yy7Z
VXjKRME8TUE-Xh06ro&client-id=00723478965317
-07gjg5o912o1v422hhlkf2
rmif7m3no6.apps.googleusercontent.com&app-secret=AvbIGURFindytojt2
342HQWTm4h
```

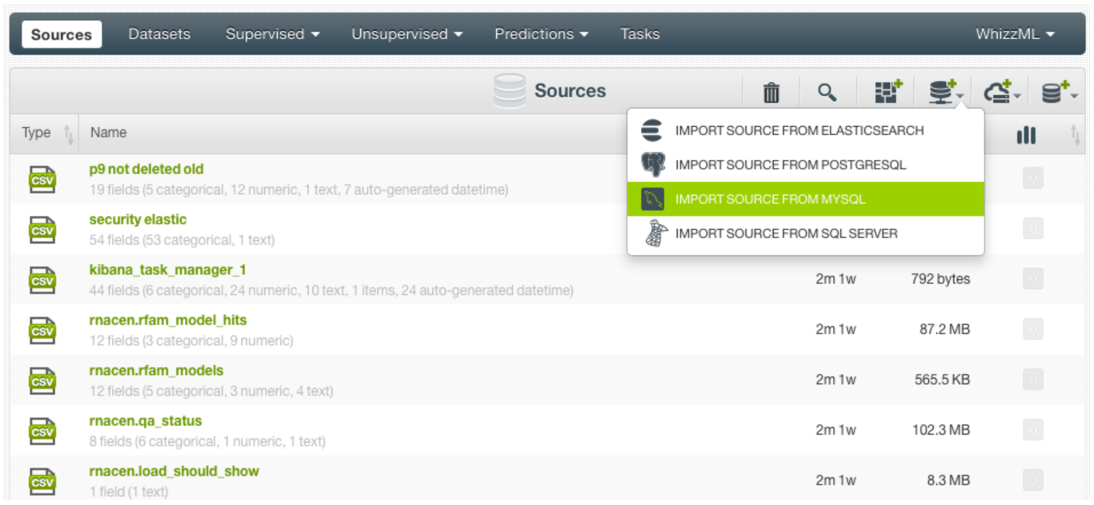
Figure 8.22: Example of a Google Drive URL

8.8 External Data Stores

You can create BigML Sources directly from external data stores, which are databases or search engines. For databases, BigML supports PostgreSQL, MySQL and SQL Sever. The support of Spark SQL will be added in the near future. BigML also integrates with the Elasticsearch engine so that you can directly create Sources from it.

In order to create Sources, you will need establish connections to external data stores by creating external connectors.

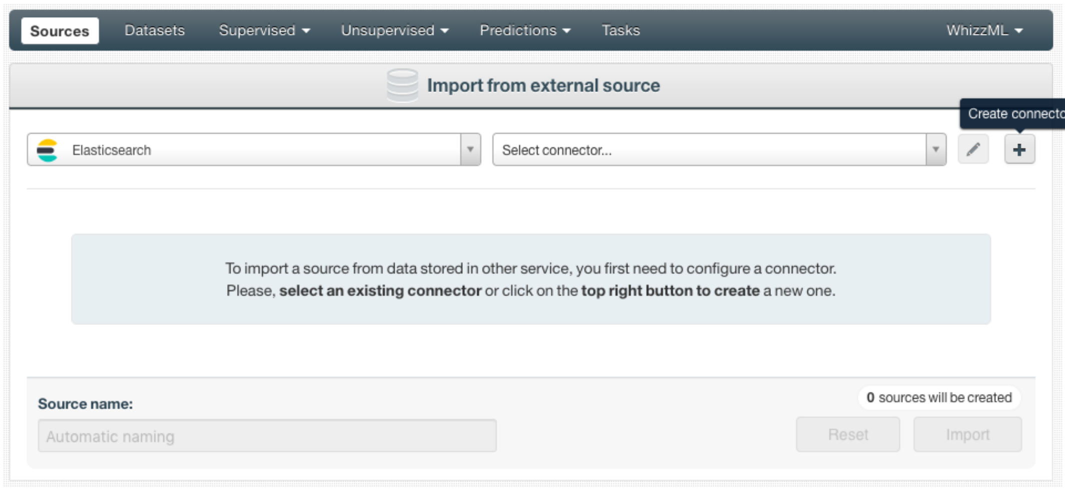
In the BigML Dashboard, under the Sources tab, you can find a database icon with a dropdown for external data stores as shown below:



Type	Name		
CSV	p9 not deleted old 19 fields (5 categorical, 12 numeric, 1 text, 7 auto-generated datetime)		
CSV	security elastic 54 fields (53 categorical, 1 text)		
CSV	kibana_task_manager_1 44 fields (6 categorical, 24 numeric, 10 text, 1 items, 24 auto-generated datetime)	2m 1w	792 bytes
CSV	rnacen.rfam_model_hits 12 fields (3 categorical, 9 numeric)	2m 1w	87.2 MB
CSV	rnacen.rfam_models 12 fields (5 categorical, 3 numeric, 4 text)	2m 1w	565.5 KB
CSV	rnacen.qa_status 8 fields (6 categorical, 1 numeric, 1 text)	2m 1w	102.3 MB
CSV	rnacen.load_should_show 1 field (1 text)	2m 1w	8.3 MB

Figure 8.23: Menu options to create a source from an external data store

After choosing your desired data store from the dropdown list, you can then either select from existing external connectors or create a new external connector. You can also switch to a different data store here.



To import a source from data stored in other service, you first need to configure a connector.
Please, **select an existing connector** or click on the **top right button to create** a new one.

Source name: Automatic naming

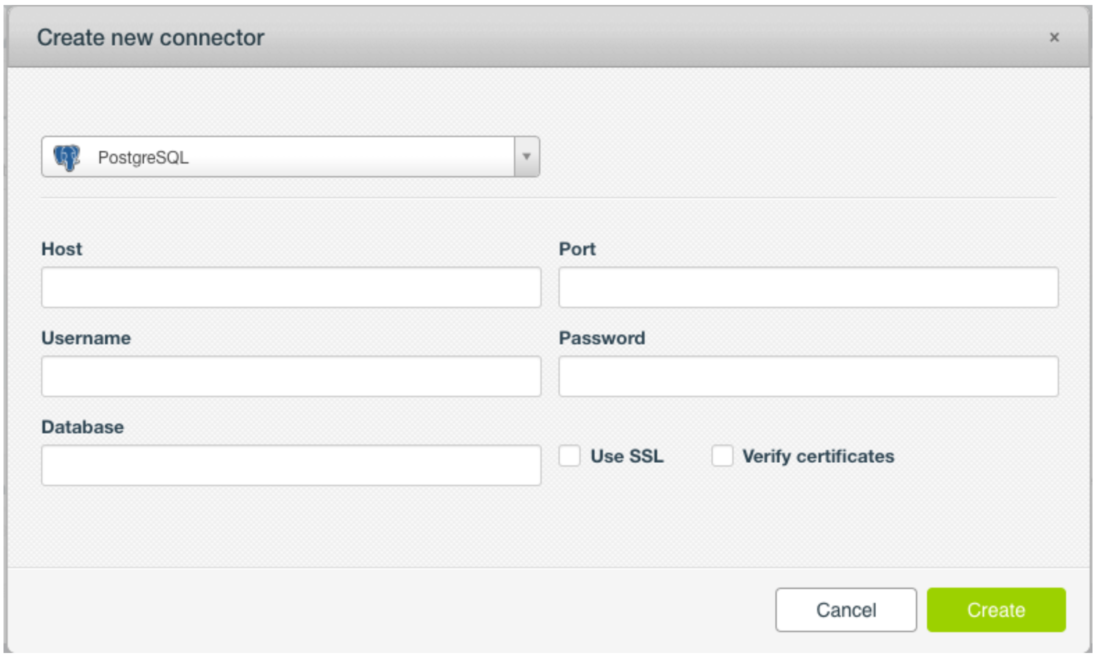
0 sources will be created

Figure 8.24: Options to configure an external connector for a datastore

8.8.1 Creating a New External Connector for Data Stores

To create a new external connector for a data store, you will need to provide all the necessary information. This can vary depending on the different data stores.

For databases such as PostgreSQL, MySQL and SQL Sever, you will need to provide the Host name, the Port number, the authentication information which are the Username and Password, and the name of the Database you want to import. You also can specify to use SSL or Verify certificates.

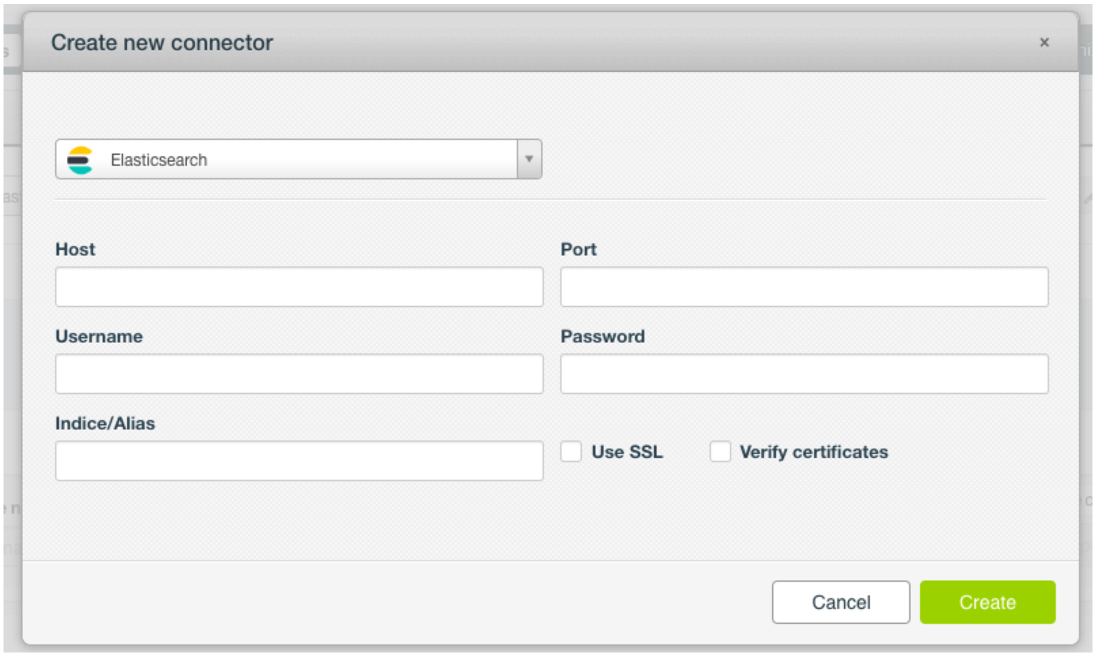

Create new connector ×

PostgreSQL

Host	Port
Username	Password
Database	Use SSL / Verify certificates

Figure 8.25: Form to create a new external connector for a datastore

For Elasticsearch, you will need to prvoide the Host name and the authentication information as well. But instead of the name of the Database, optionally you can provide the Indice or Alias for the search, as shown here:


Create new connector ×

Elasticsearch

Host	Port
Username	Password
Indice/Alias	Use SSL / Verify certificates

Figure 8.26: Form to create a new external connector for Elasticsearch

8.8.2 Editing an External Connector for Data Stores

You can edit an existing external connector by clicking on the pencil icon next to the connector you've selected.

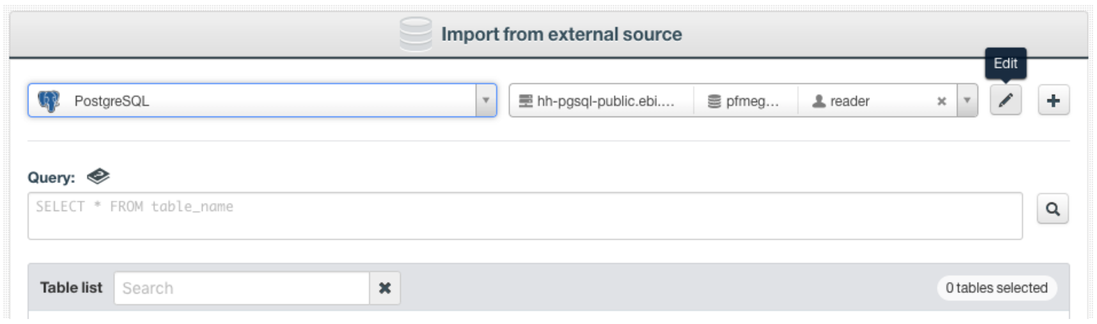
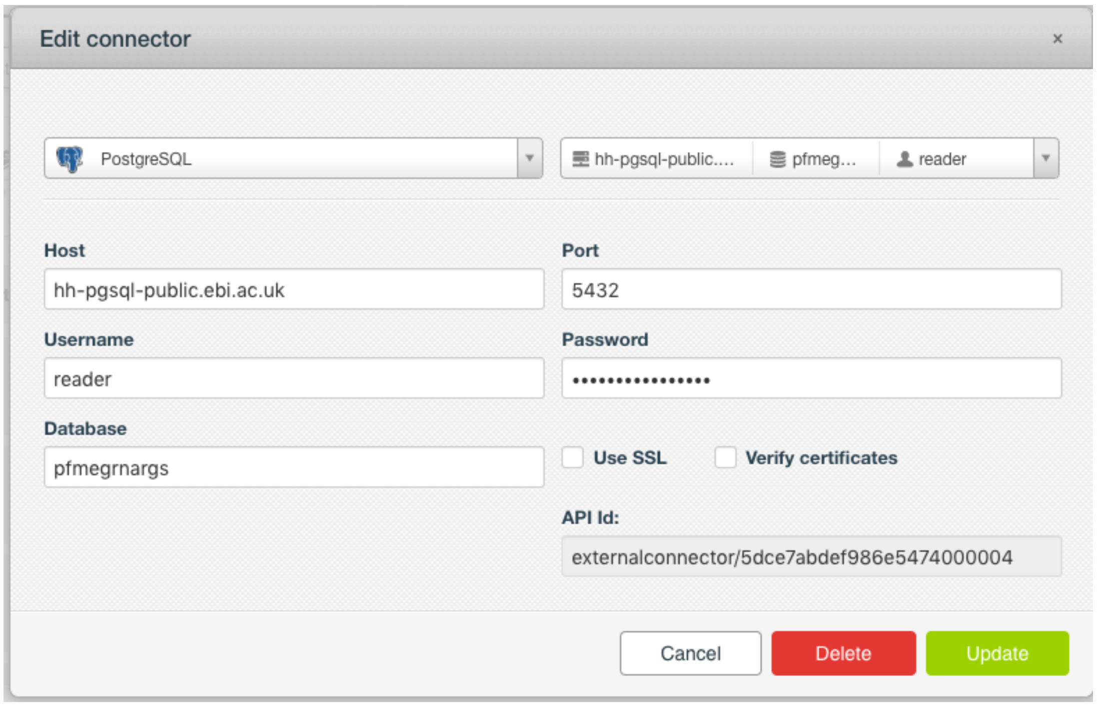


Figure 8.27: Click to edit an external connector for a data store

You then can modify the parameters of the external connector:


Edit connector

PostgreSQL | hh-pgsql-public.... | pfmeg... | reader

Host: hh-pgsql-public.ebi.ac.uk

Port: 5432

Username: reader

Password: ••••••••••••••••

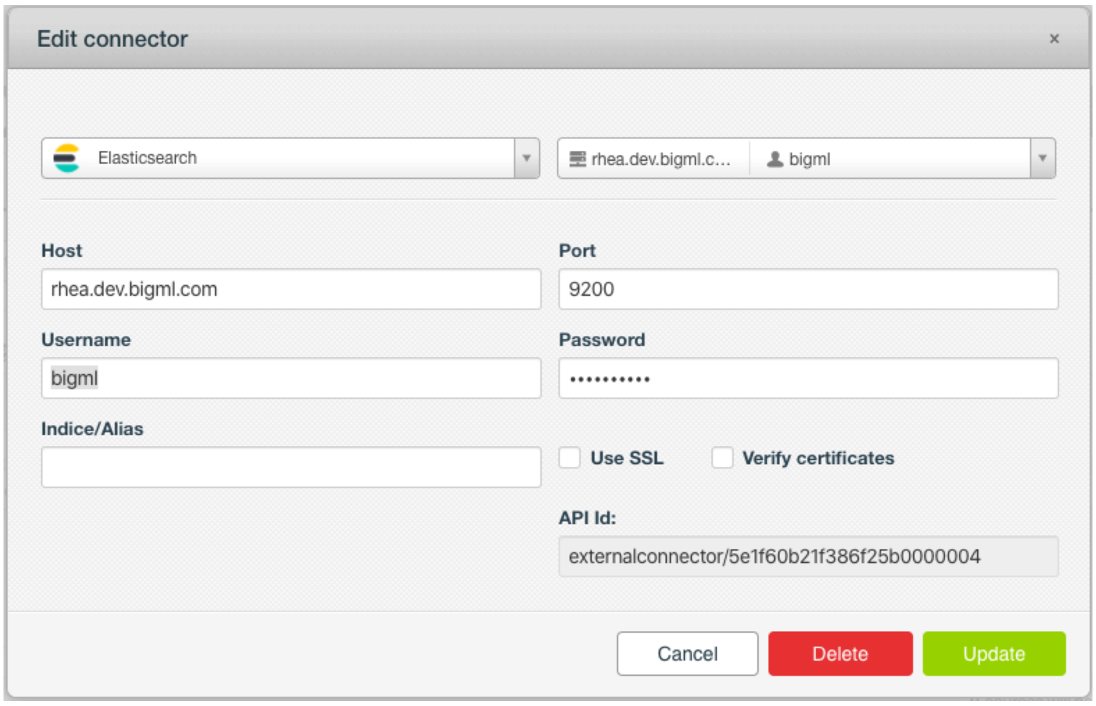
Database: pfmegrnargs

Use SSL ☐ Verify certificates ☐

API Id: externalconnector/5dce7abdef986e5474000004

Figure 8.28: Edit an external connector for a database

Here is an example to edit the parameters of an external connector for Elasticsearch:


Edit connector

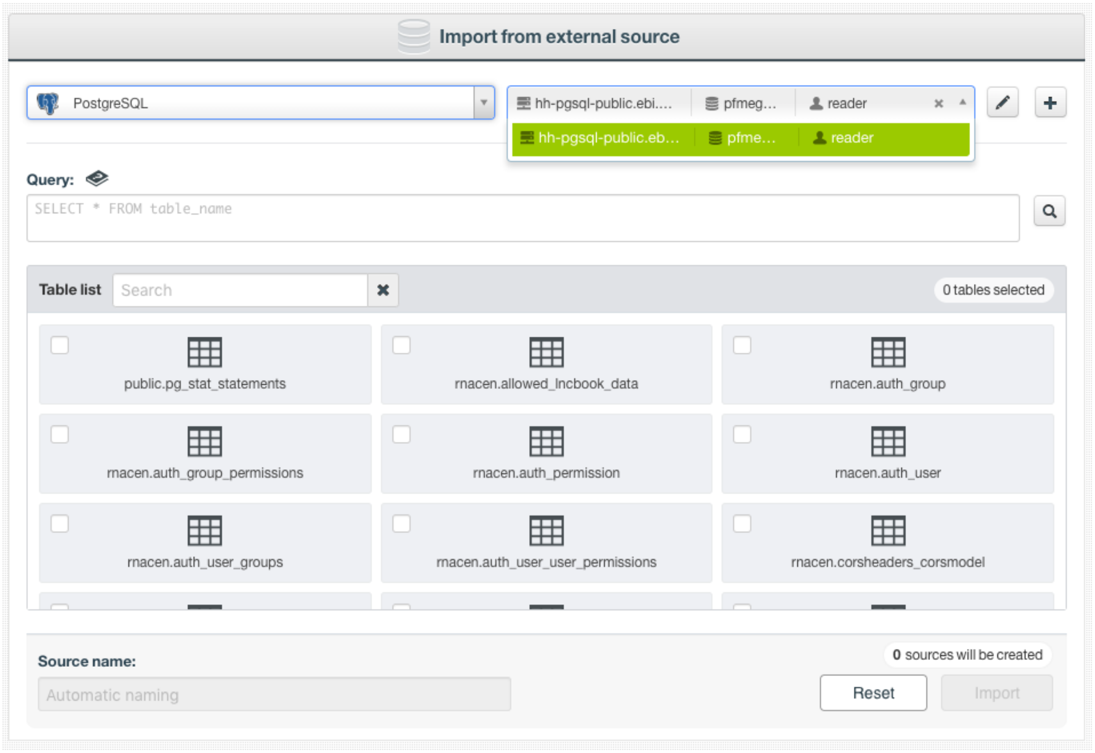
Elasticsearch | rhea.dev.bigml.c... | bigml

Host: rhea.dev.bigml.com
Port: 9200
Username: bigml
Password: ••••••••••
Indice/Alias:
Use SSL
Verify certificates
API Id: externalconnector/5e1f60b21f386f25b0000004

Figure 8.29: Edit an external connector for Elasticsearch

8.8.3 Creating a Source From a Data Store

After you select an existing external connector to the desired data store or the newly created external connector from the dropdown list, you will see a Query box and a Table list:


Import from external source

PostgreSQL | hh-pgsql-public.ebi.... | pfmeg... | reader

Query: SELECT * FROM table_name

Table list | Search | 0 tables selected

public.pg_stat_statements
rnacen.allowed_lncbook_data
rnacen.auth_group
rnacen.auth_group_permissions
rnacen.auth_permission
rnacen.auth_user
rnacen.auth_user_groups
rnacen.auth_user_user_permissions
rnacen.corsheaders_corsmodel

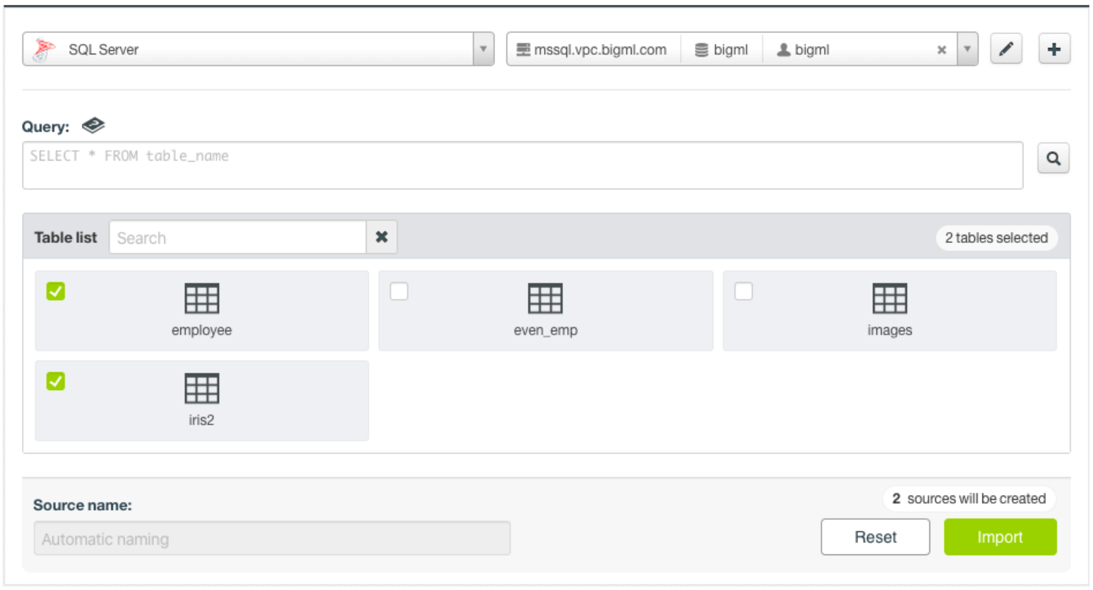
Source name: Automatic naming
0 sources will be created

Figure 8.30: After selecting an external connector for a datastore

Here you have two ways to import your data.

8.8.3.1 Creating Sources by Selecting Tables

The Table list contains all tables and views (where applicable) from your data store. By clicking on the checkboxes, you can select one or more tables and views. Then you import them into your BigML account as BigML Sources by clicking on the Import button. Each table will be imported into a separate Source.



SQL Server | mssql.vpc.bigml.com | bigml | bigml

Query:

```
SELECT * FROM table_name
```

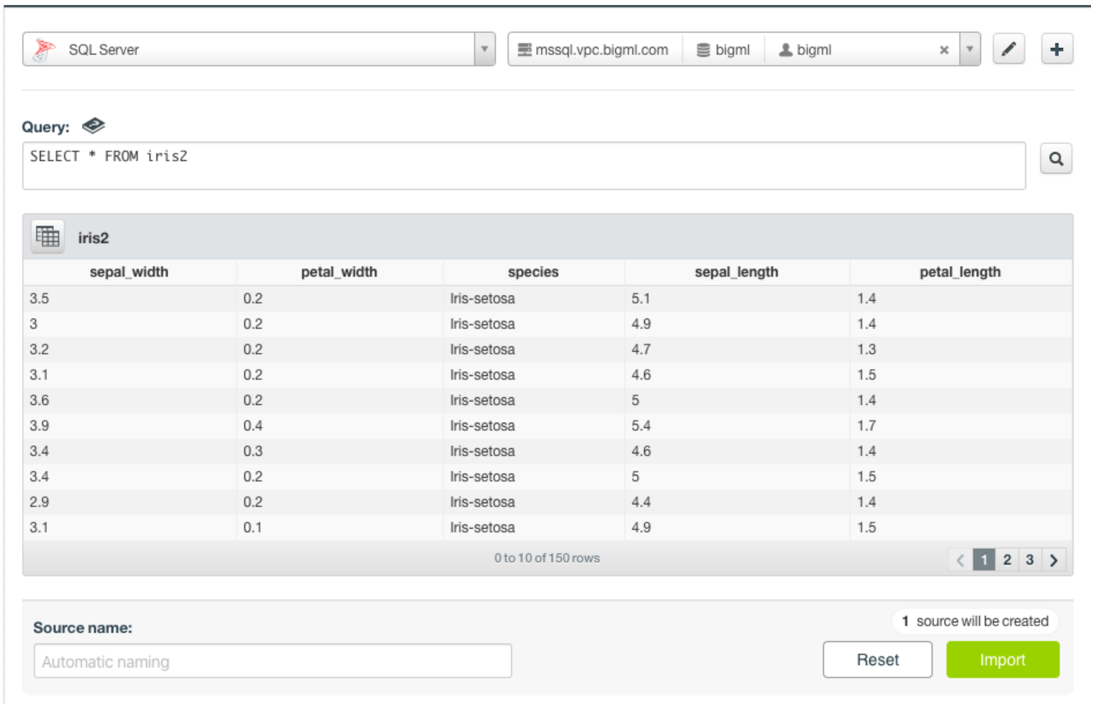
Table list | Search | 2 tables selected

- [✓] employee
- [ ] even_emp
- [ ] images
- [✓] iris2

Source name: Automatic naming | 2 sources will be created | Reset | Import

Figure 8.31: Selecting tables to import

If you would like first to take a look at the data from a given table before importing, you can click on the table int the Table list for a preview. In the preview, you can see the table columns and some sample data. See this for an example of database table preview:



SQL Server | mssql.vpc.bigml.com | bigml | bigml

Query:

```
SELECT * FROM iris2
```

iris2

sepal_width	petal_width	species	sepal_length	petal_length
3.5	0.2	Iris-setosa	5.1	1.4
3	0.2	Iris-setosa	4.9	1.4
3.2	0.2	Iris-setosa	4.7	1.3
3.1	0.2	Iris-setosa	4.6	1.5
3.6	0.2	Iris-setosa	5	1.4
3.9	0.4	Iris-setosa	5.4	1.7
3.4	0.3	Iris-setosa	4.6	1.4
3.4	0.2	Iris-setosa	5	1.5
2.9	0.2	Iris-setosa	4.4	1.4
3.1	0.1	Iris-setosa	4.9	1.5

0 to 10 of 150 rows | 1 2 3

Source name: Automatic naming | 1 source will be created | Reset | Import

Figure 8.32: Previewing a database table

8.8.3.2 Creating Sources by Constructing Queries

You can also create a Source by constructing a SQL query. This is useful because sometimes the database table import is too simplistic. By using a SQL Select statement, you can select the exact data you want, even from multiple tables and views.

If you only wish to import a subset of columns from a table, the query can be as simple as

```
select sepal_width, petal_width, species from iris2
```

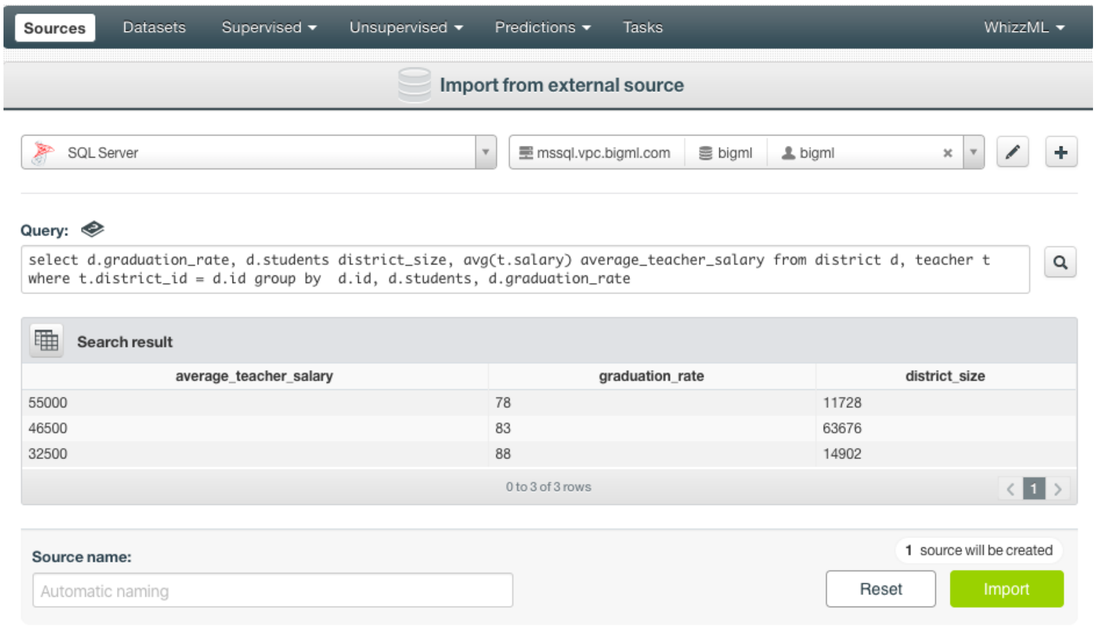
When you preview a whole table, you can see that the select statement in the Query box (See Figure 8.32.):

```
select * from iris2
```

After writing the query in the Query box, you can click on the Search button on the right to execute it. This can verify the query is valid in your data store. It also gives you a preview by showing the initial results, which allows you to confirm your intentions before creating the BigML Source.

You can actually take advantage of your data store's full query language. Below is a more advanced example, which shows a select statement with both a join and group-by clause. In the database, there is one table with school district information, and another containing individual teacher statistics. By using the select statement, we are creating a BigML Source with information about school districts that include the average teach salary in each district:

```
select d.graduation_rate, d.students district_size,
      avg(t.salary) average_teacher_salary
from district d, teacher t
where t.district_id = d.id
group by d.id, d.students, d.graduate_rate
```



SQL Server | mssql.vpc.bigml.com | bigml | bigml

Query:

```
select d.graduation_rate, d.students district_size, avg(t.salary) average_teacher_salary from district d, teacher t
where t.district_id = d.id group by  d.id, d.students, d.graduation_rate
```

Search result

average_teacher_salary	graduation_rate	district_size
55000	78	11728
46500	83	63676
32500	88	14902

0 to 3 of 3 rows

Source name: Automatic naming

1 source will be created

Reset Import

Figure 8.33: Importing by using an advanced query

8.8.4 Creating a Source From a Data Store Via API

Creating Sources from external data sources is fully supported in the BigML API. You can use BigML API to programmatically create, retrieve, update, delete and list external connectors, and then use them to import.

Just like on the Dashboard, you can import data with either a table or a custom query. Here is an example using curl that imports a “Sales” table as a BigML Source.

```
curl "https://bigml.io/source?\$BIGML_AUTH" \
    -X POST \
    -H 'content-type: application/json' \
    -d '{"external_data": {
                "source": "sqlserver",
                "connection": {
                    "host": "db.bigbox.com",
                    "port": 1433,
                    "database": "biztel",
                    "user": "autosource",
                    "password": "********"
                },
                "tables": "Sales"}}'
```

With BigML API, you actually have a few options to control the imported data without using a custom query. You can specify which fields to include or which to exclude, as well as limiting the number of records to import. You can also specify an offset along with an ordering to put that offset in context.

For Elasticsearch, creating a BigML Source using API is similar, as shown by the following example:

```
curl "https://bigml.io/source?\$BIGML_AUTH" \
    -X POST \
    -H 'content-type: application/json' \
    -d '{"external_data": {
                "source": "elasticsearch",
                "connection": {
                    "host": "localhost",
                    "port": 9200,
                    "user": "username",
                    "password": "********"
                },
                "tables": "kibana_sample_data_logs"}}'
```

For more information on importing data directly through the BigML API, please refer to the documentation[6].

[6] https://bigml.com/api/externalconnectors

CHAPTER 9

Inline Sources

The BigML Dashboard also has a simple editor that allows you to create “inline” sources. You can open it using the button shown on Figure 9.1.

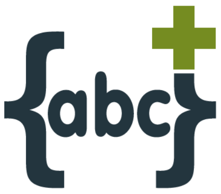
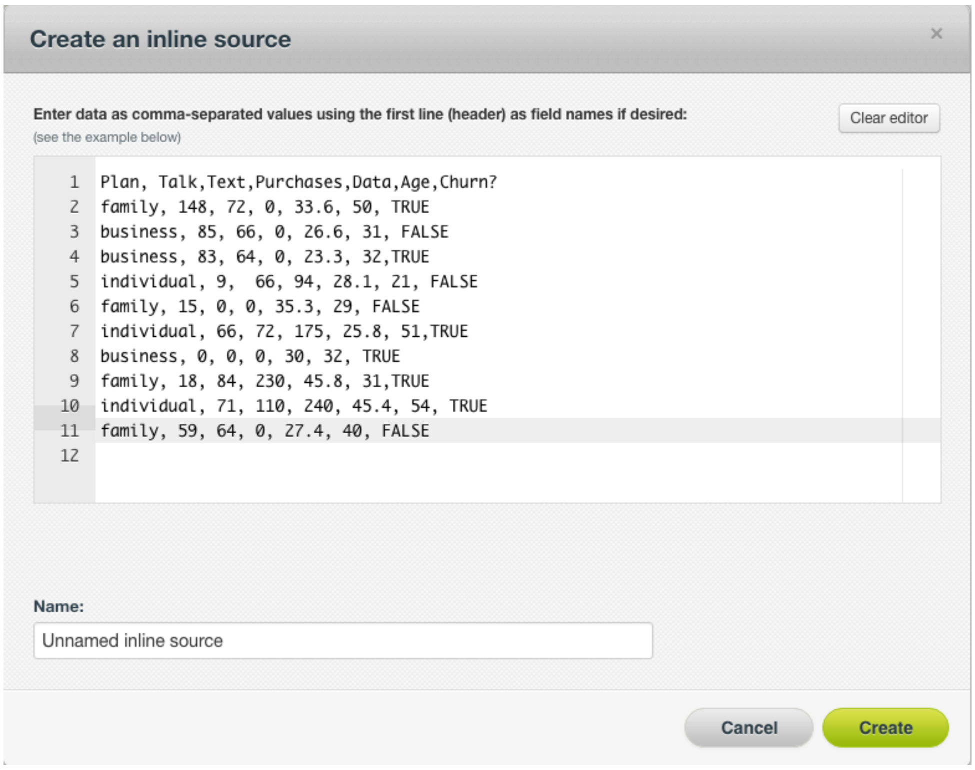


Figure 9.1: Button to open the inline source editor

You can see what the editor looks like on Figure 9.2. You can just type your data or copy and paste it. Inline sources are useful for basic experimentation and to learn and practice Machine Learning with BigML.



Enter data as comma-separated values using the first line (header) as field names if desired:
(see the example below)

```
Plan, Talk,Text,Purchases,Data,Age,Churn?
family, 148, 72, 0, 33.6, 50, TRUE
business, 85, 66, 0, 26.6, 31, FALSE
business, 83, 64, 0, 23.3, 32,TRUE
individual, 9,  66, 94, 28.1, 21, FALSE
family, 15, 0, 0, 35.3, 29, FALSE
individual, 66, 72, 175, 25.8, 51,TRUE
business, 0, 0, 0, 30, 32, TRUE
family, 18, 84, 230, 45.8, 31,TRUE
individual, 71, 110, 240, 45.4, 54, TRUE
family, 59, 64, 0, 27.4, 40, FALSE
```

Name: Unnamed inline source

Figure 9.2: Editor of inline sources

CHAPTER 10

Size Limits

BigML does not impose any limits on the number of sources you can upload to a single account or on the number of sources you can assign to a specific project. Each source can store an arbitrarily-large number of instances and also manage a relatively big number of fields. For example, the BigML multi-tenant version can process datasets with hundreds of millions of rows and dozens of thousands of fields.

The BigML multi-tenant version does impose some limits on the total size of files, depending on the way you bring your data to BigML:

Local sources: files uploaded directly including through the browser, drag and drop, or through the API are limited to **64 GB** in size.

Remote sources: files uploaded using any of the accepted protocols defined in Section 8.1 are also limited up to **64 GB**; however using Amazon Simple Storage Service (**S3**), the limit is **5 TB**.

Inline sources: sources created using the online editor are limited to **16 MB**.

If yours is a case where the machine learning-ready data exceeds these size limits, please consider a **BigML Private Deployment** that can raise those limitations and be tailored to manage bigger datasets.

CHAPTER 11

Descriptive Information

Each source has an associated **name**, **description**, **category**, and **tags**. A brief description follows for each concept. In Figure 11.2, you can see the options that the **More info** panel gives to edit them.

11.1 Source Name

Each source has an associated **name** that is displayed on the list and also on the top bar of a source view. Source names are indexed to be used in searches.

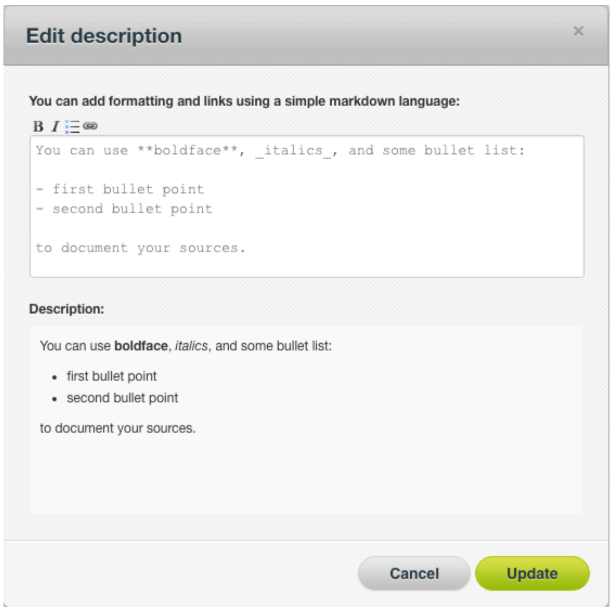
When you create a source, the default name is that of the file used to create it. Edit it using the **More info** panel on the right corner of the source view. (See Figure 11.2.)

The name of a source cannot be longer than **256** characters. There is no restriction on the characters that can be used in a source name. More than one source can have the same name even within the same project. They will always have different identifiers.

11.2 Description

Each source also has a **description** that it is very useful for documenting your Machine Learning projects. Descriptions can be written using plain text and also markdown[1]. BigML provides a simple markdown editor that accepts a subset of markdown syntax. (See Figure 11.1.)

[1] https://en.wikipedia.org/wiki/Markdown



You can add formatting and links using a simple markdown language:

```
You can use **boldface**, _italics_, and some bullet list:

- first bullet point
- second bullet point

to document your sources.
```

Description:

You can use **boldface**, *italics*, and some bullet list:

- first bullet point
- second bullet point

to document your sources.

Figure 11.1: Markdown editor for source descriptions

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

11.3 Category

Each source is associated with a category. Categories are useful to classify sources according to the domain from which your data is taken. (See Figure 11.2.) This is useful when you use BigML to solve problems across industries or multiple customers.

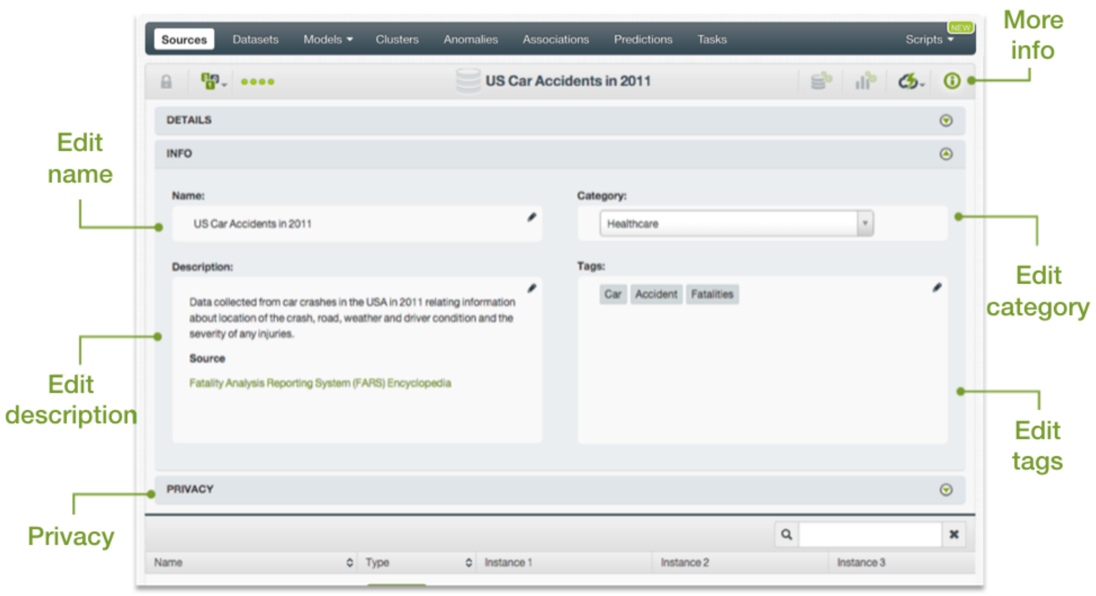
A source category must be one of the categories listed on table Table 11.1.

Table 11.1: Categories used to classify sources by BigML

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

11.4 Tags

A source can also have a number of **tags** associated with it. This helps to retrieve the source via the BigML API and provides sources with some extra information. Each tag is limited to a maximum of 128 characters. Each source can have up to 32 different tags. (See Figure 11.2.)



US Car Accidents in 2011

Name: US Car Accidents in 2011

Category: Healthcare

Description:

Data collected from car crashes in the USA in 2011 relating information about location of the crash, road, weather and driver condition and the severity of any injuries.

Source

Fatality Analysis Reporting System (FARS) Encyclopedia

Tags: Car Accident Fatalities

Figure 11.2: Panel to edit a source name, category, description and tags

11.5 Counters

For each source, BigML also stores a number of counters to track the number of other resources that have been created using the corresponding source as starting point. In the source view, you can see a menu option that displays these counters which also allow you to quickly jump to all the resources of one type that have been created with this source. (See Figure 11.3.)

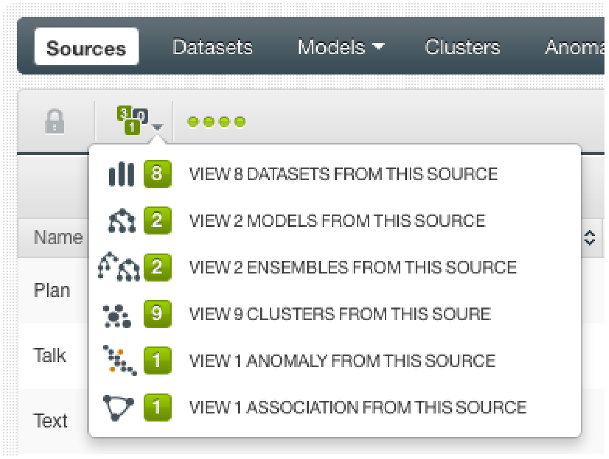
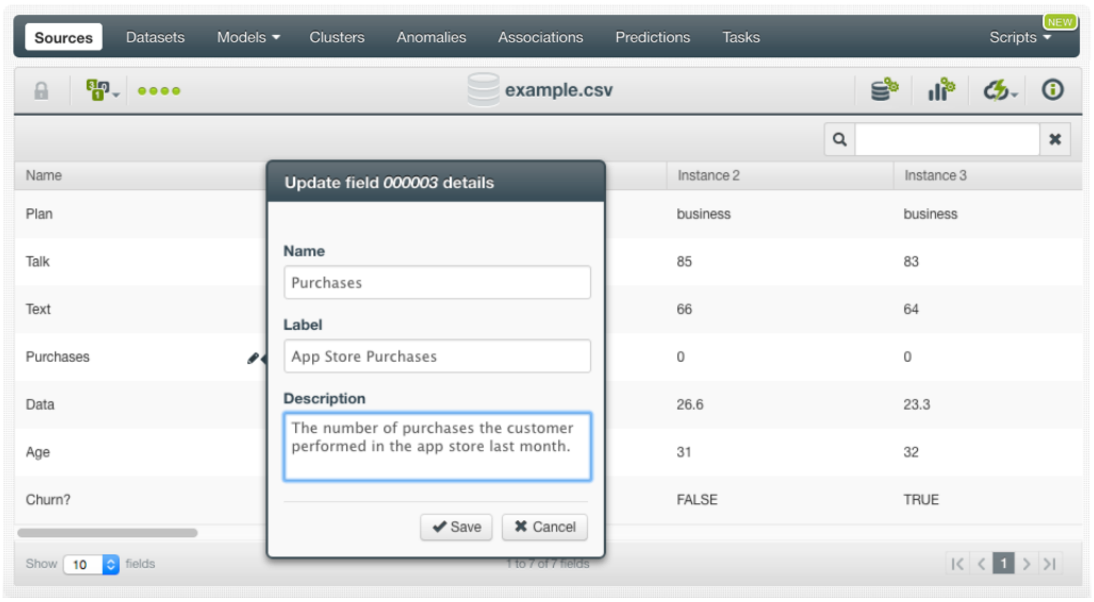

Sources
Datasets
Models
Clusters
Anoma
Name
Plan
Talk
Text
8 VIEW 8 DATASETS FROM THIS SOURCE
2 VIEW 2 MODELS FROM THIS SOURCE
2 VIEW 2 ENSEMBLES FROM THIS SOURCE
9 VIEW 9 CLUSTERS FROM THIS SOURE
1 VIEW 1 ANOMALY FROM THIS SOURCE
1 VIEW 1 ASSOCIATION FROM THIS SOURCE

Figure 11.3: Menu option to quickly access resources created with a source

11.6 Field Names, Labels and Descriptions

In addition to its name, each field of a source can also be furnished with extra information such as a **label** and a **description**. This information is displayed when you mouse over fields. It can be very useful to recognize what each field means on your model since labels and descriptions are inherited when you create other resources.

When you mouse over each field in a source view, you will see a pencil. Clicking on it, opens a dialog box such as the one displayed on Figure 11.4 that will allow you to update the name, label, and description of that field.



Name	Instance 2	Instance 3
Plan	business	business
Talk	85	83
Text	66	64
Purchases	0	0
Data	26.6	23.3
Age	31	32
Churn?	FALSE	TRUE

Update field *000003* details

Name
Purchases

Label
App Store Purchases

Description
The number of purchases the customer performed in the app store last month.

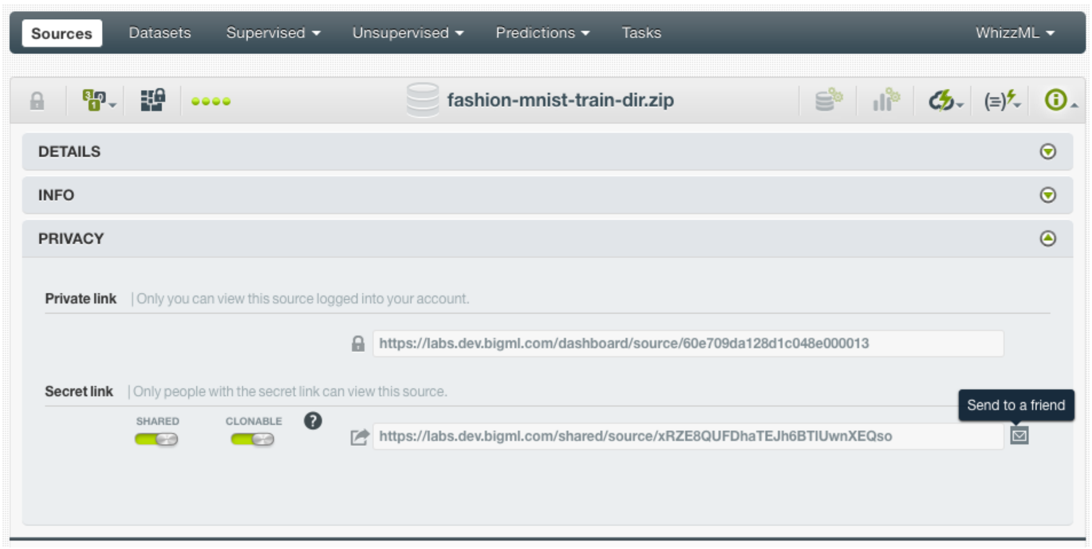
Figure 11.4: Updating a field name, label, and description

CHAPTER 12

Source Privacy

BigML allows you to share sources via secret links as it does with other types of resources; the privacy options for a source can be configured in the MORE INFO menu option, shown in Figure 12.1. There are two **levels of privacy** for BigML sources:

- **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 a secret link. Here users can also decide whether to make the source clonable.



Private link | Only you can view this source logged into your account.

https://labs.dev.bigml.com/dashboard/source/60e709da128d1c048e000013

Secret link | Only people with the secret link can view this source.

https://labs.dev.bigml.com/shared/source/xRZE8QUFDhaTEJh6BTIUwnXEQso

Figure 12.1: Source privacy options

CHAPTER 13

Moving Sources

When you create a source, it will be assigned to the project indicated on the project selector bar. (See Figure 13.1.)

PROJECT: My first project
3

Figure 13.1: Project bar

When the project selector bar shows **All** and you create a new source, it will not be assigned to any project.

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

1. In the source view, within the 1-click actions for each source. (See Figure 13.2.)

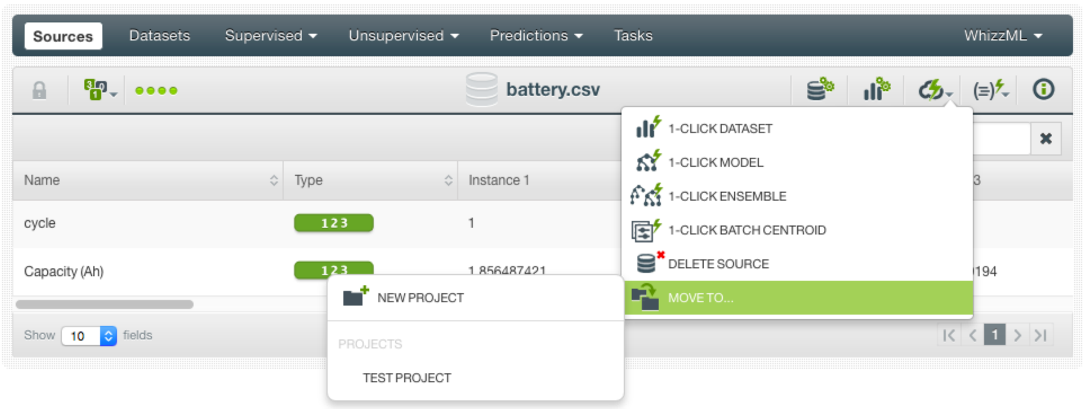
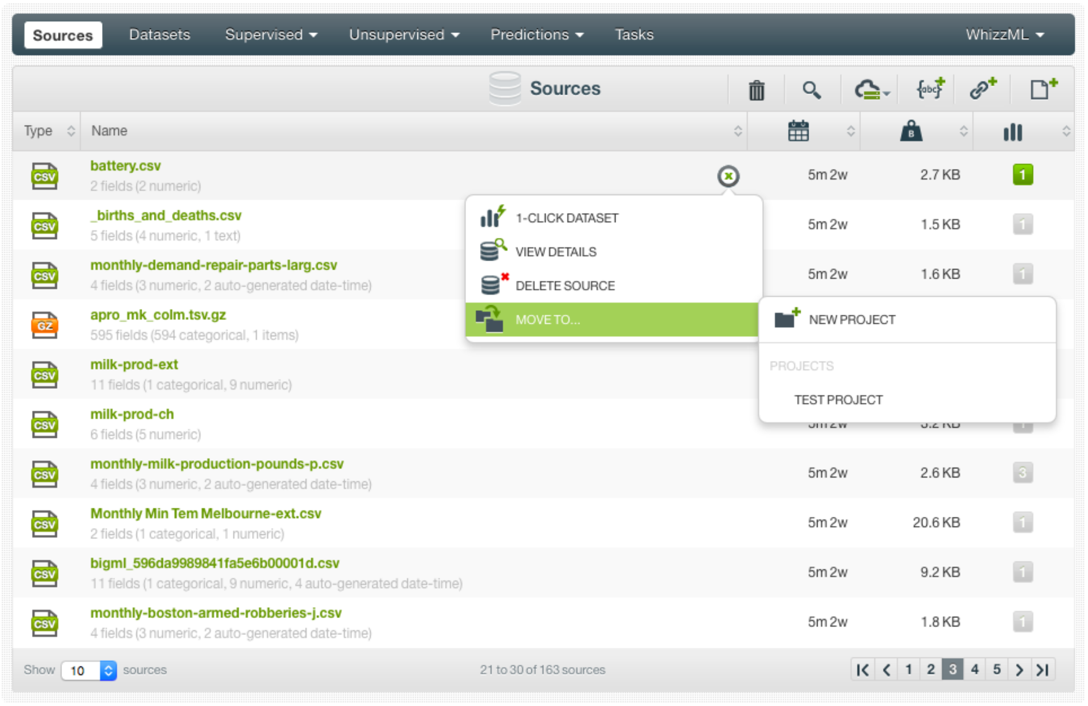


Figure 13.2: Menu option to move sources

2. Within the 1-click actions of a source in the source list view. (See Figure 13.3.)



Type	Name			
CSV	battery.csv 2 fields (2 numeric)	5m 2w	2.7 KB	1
CSV	_births_and_deaths.csv 5 fields (4 numeric, 1 text)	5m 2w	1.5 KB	1
CSV	monthly-demand-repair-parts-larg.csv 4 fields (3 numeric, 2 auto-generated date-time)	5m 2w	1.6 KB	1
GZ	apro_mk_colm.tsv.gz 595 fields (594 categorical, 1 items)			
CSV	milk-prod-ext 11 fields (1 categorical, 9 numeric)			
CSV	milk-prod-ch 6 fields (5 numeric)	5m 2w	3.2 KB	1
CSV	monthly-milk-production-pounds-p.csv 4 fields (3 numeric, 2 auto-generated date-time)	5m 2w	2.6 KB	3
CSV	Monthly Min Tem Melbourne-ext.csv 2 fields (1 categorical, 1 numeric)	5m 2w	20.6 KB	1
CSV	bigml_596da9989841fa5e6b00001d.csv 11 fields (1 categorical, 9 numeric, 4 auto-generated date-time)	5m 2w	9.2 KB	1
CSV	monthly-boston-armed-robberies-j.csv 4 fields (3 numeric, 2 auto-generated date-time)	5m 2w	1.8 KB	1

Show 10 sources 21 to 30 of 163 sources 1 2 3 4 5

Figure 13.3: Menu option to move sources from the source list view

CHAPTER 14

Deleting Sources

You can delete your sources from the source view, using the DELETE SOURCE menu option in 1-click action menu or using the pop up menu in the **source list view**.

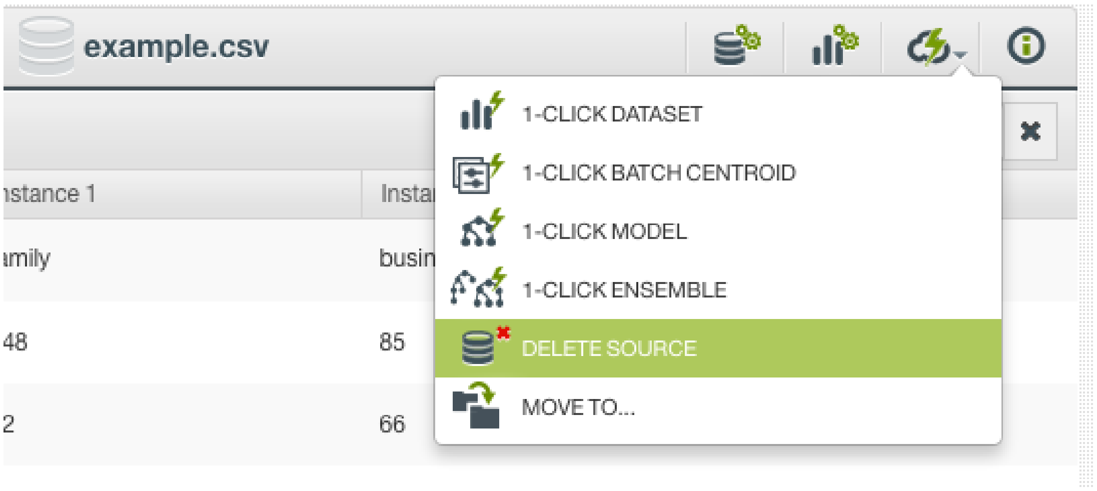
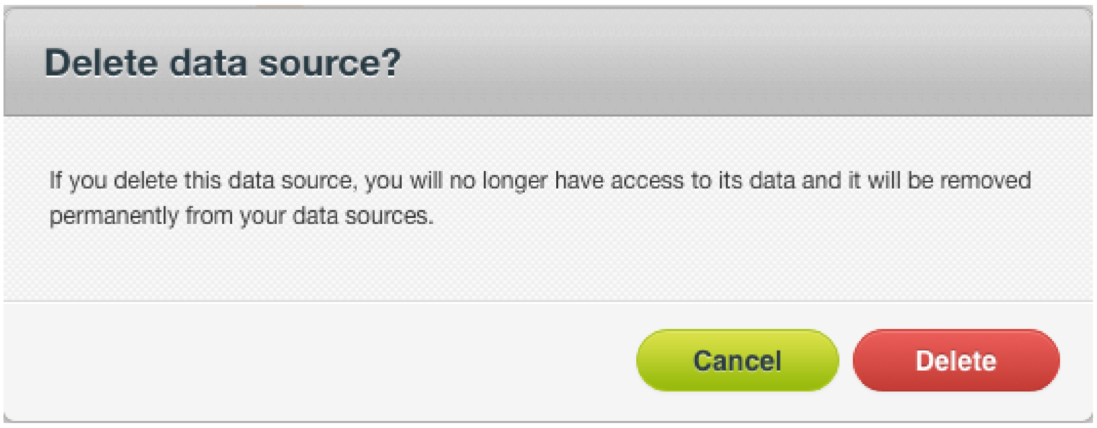


Figure 14.1: Delete a source menu option

A modal window (see Figure 14.2) will be displayed asking you for confirmation. Once a source is deleted, it is permanently deleted, and there is no way you (or even the IT folks at BigML) can retrieve it.


Delete data source?

If you delete this data source, you will no longer have access to its data and it will be removed permanently from your data sources.

Figure 14.2: Delete a source modal window

You can also delete a source from the **source list view**. On the 1-click pop up menu that is displayed for each source, you will find an option for deleting. (See Figure 14.3.)

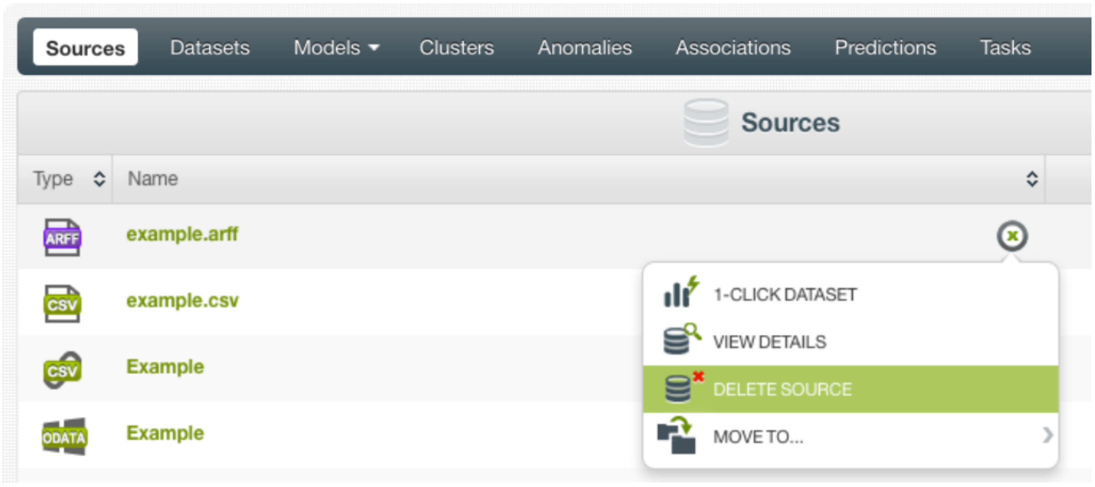
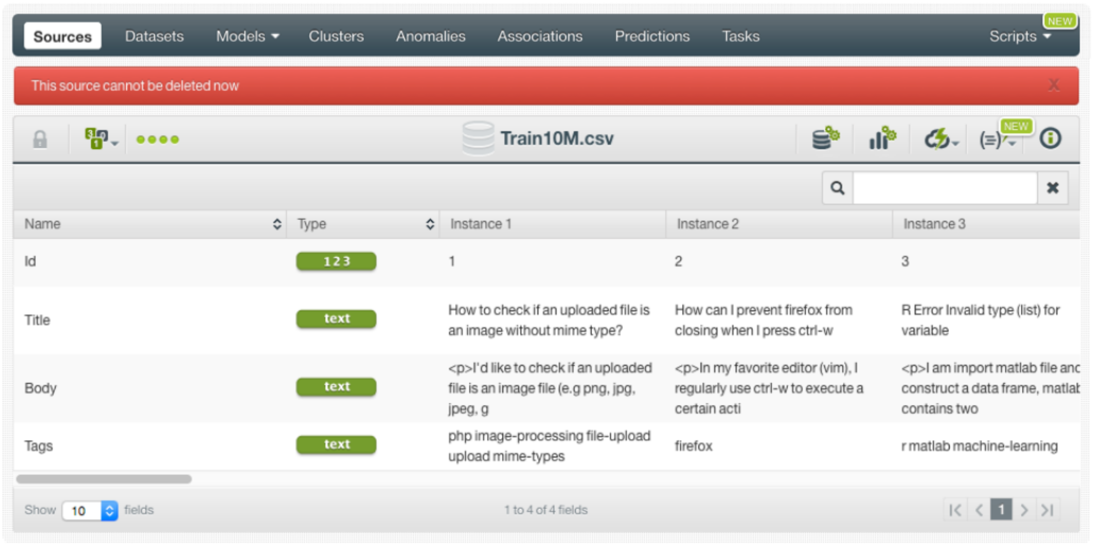


Figure 14.3: Delete a source pop up menu option

Note: if you try to delete a source while it is being used to create a dataset you will see an alert that the source cannot be deleted now. (See Figure 14.4.)



This source cannot be deleted now

Name	Type	Instance 1	Instance 2	Instance 3
Id	123	1	2	3
Title	text	How to check if an uploaded file is an image without mime type?	How can I prevent firefox from closing when I press ctrl-w	R Error Invalid type (list) for variable
Body	text	<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpeg, g	<p>In my favorite editor (vim), I regularly use ctrl-w to execute a certain acti	<p>I am import matlab file and construct a data frame, matlab contains two
Tags	text	php image-processing file-upload upload mime-types	firefox	r matlab machine-learning

Show 10 fields 1 to 4 of 4 fields

Figure 14.4: Alert displayed when trying to delete a source being used to create a dataset

CHAPTER 15

Takeaways

This chapter explains sources in detail. Here's a list of key points:

- A source allows you to bring data to BigML.
- BigML recognizes a variety of formats, protocols, and storages to create new sources.
- A source stores an arbitrarily-large collection of instances describing an entity of interest you want to model.
- BigML works best with data in a tabular format where each row represents an instance of the entity you want to model, and each column represents a field describing all the instances.
- After you create your source in BigML, each field in your source is displayed as a row and each column as an instance. This is because for highly dimensional data the transposed layout provides better navigability (i.e., datasets with thousands of fields can be paginated better).
- A source helps BigML to know how to parse your data so that the instances and field types can be correctly processed.
- You can configure your source in multiple ways to ensure BigML parses every field right.
- You can create sources from local files, remote files, or using an inline editor.
- Uploading one non-archive file, or one archive file (tar or zip) containing only one file, will create a single source. Uploading an archive file (tar or zip) containing multiple files will create a composite source.
- BigML supports sources in different formats, such as Table (CSV or JSON), Image, or Table+Image.
- A source is open when it can be modified. When a source is used to create a dataset, it's automatically closed.
- A source can be cloned. A cloned source is created as an open source.
- You can create sources using image files. BigML supports a wide range of image formats.
- An archive (tar or zip) file containing more than one images will create an image composite source, which can be used to create datasets for machine learning.
- If images are inside folders in the archive file (tar or zip), uploading the archive file will create image composite sources with an added label field, the values of the label being the respective innermost folder names.
- In the **fields view**, **sources view** and **iamges view** of an image composite source, you can view its fields, component sources, and images, respectively. You can also select component sources or images to perform certain operations, including adding labels to images in an open image composite source.
- When an image composite source is created, by default BigML extracts 234 features per image, respnenseting its histogram of gradients. You can configure five sets of extracted image features

in an open image composite source. You can also select one pre-trained convolutional neural network(CNN).

- You can furnish your source with descriptive information (name, description, tags, and category) and also every individual field (name, label, and description).
- You can only assign a source to a specific project.
- You can permanently delete a source.
- Figure 15.1 graphically represents the workflows a BigML source enables. A BigML source can be created using local, remote, cloud-stored, or inline sources and can be used to create datasets.

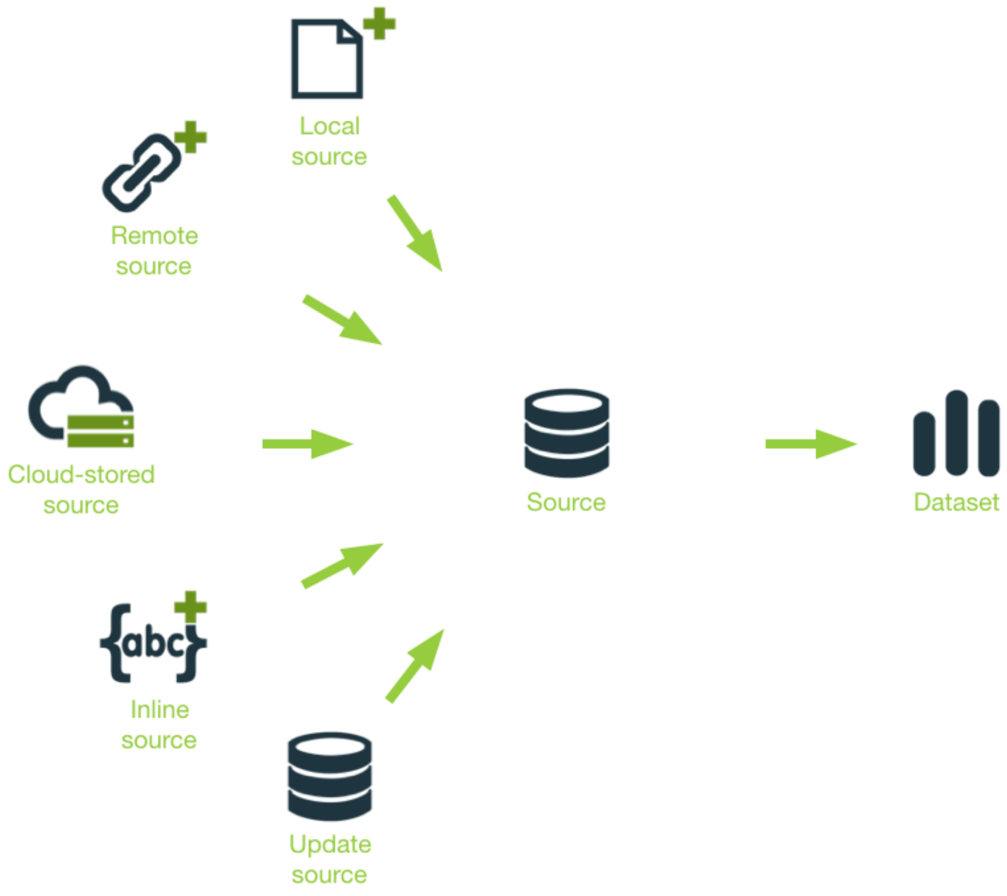


Figure 15.1: Source workflow

```
Please use the noidx option in the documentclass invocation.
```

List of Figures

List of Tables

Glossary

Association Discovery an unsupervised Machine Learning task to find out relationships between values in high-dimensional datasets. It is commonly used for market basket analysis. 54

Dashboard The BigML web-based interface that helps you privately navigate, visualize, and interact with your modeling resources. ii, 1

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. ii, 3, 62

Entity the object or subject of interest in your modeling task. A dataset is a collection of instances of the entity of interest. 1, 3, 103

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, 103

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, 103

Project an abstract resource that helps you group related BigML resources together. 2, 3, 99, 104

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. 3, 98

Source the BigML resource that represents the data source to which you wish to apply Machine Learning. A data source stores an arbitrarily-large collection of instances. A BigML source helps you ensure that your data is parsed correctly. The BigML preferred format for data sources is tabular data in which each row is used to represent one of the instances, and each column is used to represent a field of each instance. ii, 1, 6, 103

References

[1] The BigML Team. *Anomaly Detection with the BigML Dashboard*. Tech. rep. BigML, Inc., Jan. 2016.

[2] The BigML Team. *Association Discovery with the BigML Dashboard*. Tech. rep. BigML, Inc., Dec. 2015.

[3] The BigML Team. *Classification and Regression with the BigML Dashboard*. Tech. rep. BigML, Inc., May 2016.

[4] The BigML Team. *Cluster Analysis with the BigML Dashboard*. Tech. rep. BigML, Inc., May 2016.

[5] The BigML Team. *Datasets with the BigML Dashboard*. Tech. rep. BigML, Inc., Jan. 2016.

[6] The BigML Team. *Time Series with the BigML Dashboard*. Tech. rep. BigML, Inc., July 2017.

[7] The BigML Team. *Topic Models with the BigML Dashboard*. Tech. rep. BigML, Inc., Nov. 2016.

bigml®