BigML Benchmarks

The BigML Team

Version 1.1

MACHINE LEARNING MADE BEAUTIFULLY SIMPLE

info@bigml.com

Last updated June 27, 2022

About this Document

This document provides detailed timing benchmarks for a variety of BigML resources, including sources, datasets, anomaly detectors, ensembles, clusters, logistic regressions, and association rules.

Contents

CHAPTER 1

Benchmarking Data and Setup

1.1 Hardware

The benchmarks were run on commodity hardware (a standard Intel Core i7-4930K CPU, see Chapter 2) and on three memory-optimized AWS EC2 server instances[1] with Intel Xeon processors and enhanced networking and SSD storage (described in Table 1.1, see Chapter 3.

Name	Type	CPUs	Mem (GiB)	Storage (GB)
i7	Interl i7 4930K 3.40GHz	4	60	225
r3.2xlarge	Intel Xeon E5-2670 v2	8	61	160
r3.4xlarge	Intel Xeon E5-2670 v2	16	122	320
r3.8xlarge	Intel Xeon E5-2670 v2	32	244	640

Table 1.1: Hardware used for benchmarking

1.2 Synthetic Data

Synthetic data for benchmarking is generated using the gen-data program[2].

The data contains 20 numeric fields and one categorical field (the objective for the supervised models). Each category has a gaussian cluster associated with it. The clusters have random projections and translations applied to them and may overlap. For these benchmarks we used 10 classes.

The synthetic data may be produced with `gen-data` using the following command:

```
lein run --output synth100M.csv \
         --classes 10 \
         --fields 20 \
         --rows 100000000 \
         --seed 0
```

For example, Figure 1.1 shows two-dimensional data clustered around five centroids generated with this method.

[1] https://aws.amazon.com/ec2/instance-types/
[2] https://github.com/bigmlcom/gen-data

The benchmarks below were computed for synthetic datasets of 10K, 100K, 1M, 10M, and 100M instances (rows). The uncompressed size of the data ranged from approximately 4MB to 40GB.

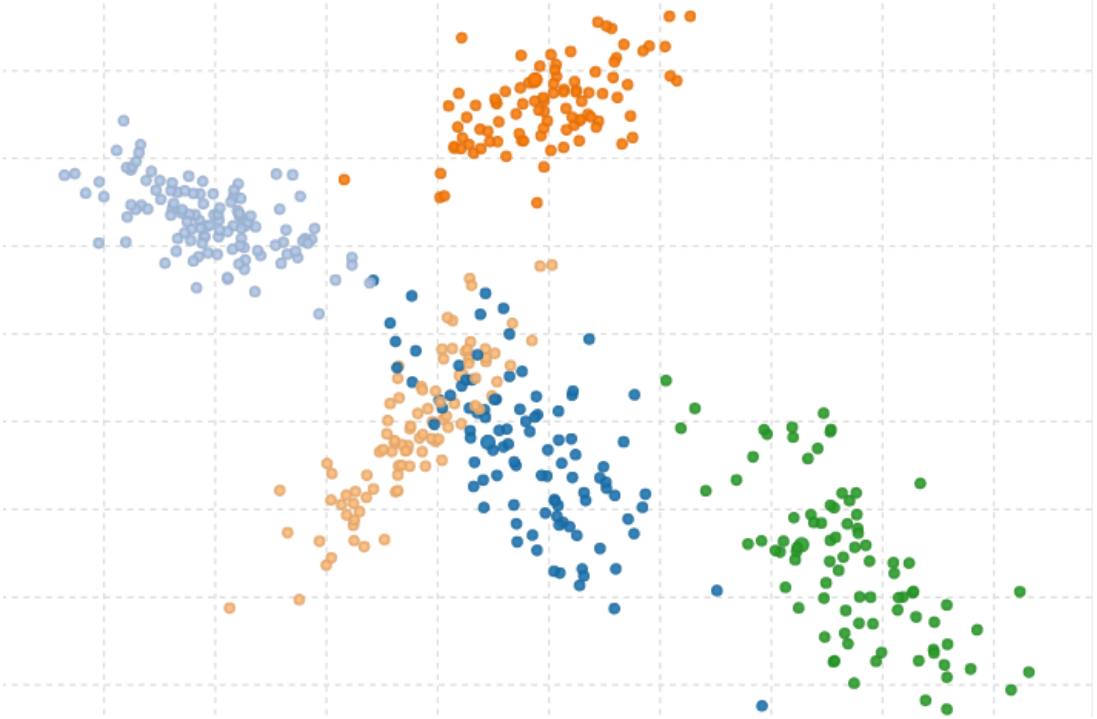

Figure 1.1: An example of two-dimensional synthetic data with five clusters

We have used a synthetic sources with varying row number (10K, 100K, 1M, 10M) and varying number of numeric input fields (16, 64, 256).

1.3 Algorithms

Benchmarks include creation times for the following BigML resoures: sources, datasets, anomaly detectors, ensembles, clusters and logistic regressions, parameterized as described in Table 1.2.

Algorithm	Parameters
Anomaly detector	isolation forest with 128 trees
K-means cluster	10 centroids ($k = 10$)
G-means cluster	critical value 5
Ensemble (random decision forest)	100 trees and node limit 10k
Logistic regression	EPS (stopping criterion) 0.02

Table 1.2: Tested algorithms and their parameters

CHAPTER 2

Local Server

The following table shows timings, in seconds, for creation of sources, datasets, anomaly detectors, ensembles, clusters and logistic regressions on an Intel Core i7-4930K CPU @ 3.40Gh for varying number of source instances and using the parameters in Table 1.2.

Instances	Size	Source	Dataset	K-means	G-means	Anomaly	Ensemble	Logistic
10K	3.9 MB	5	5	5	20	10	5	10
100K	38.6 MB	5	25	5	25	20	10	55
1M	386 MB	31	326	65	100	167	141	441
10M	3.86 GB	304	2185	137	190	1017	1090	3767
100M	38.6 GB	3472	21762	1498	2933	9711	13258	-

Table 2.1: Timings for resource creation in local server

The timings in Table 2.1 are depicted graphically in Figure 2.1 for ML algorithms and Figure 2.2 for sources and datasets, as functions of the number of rows of the input source.

In the case of logistic regression, the 100M row case could not be computed because the local server has not enough RAM for that computation.

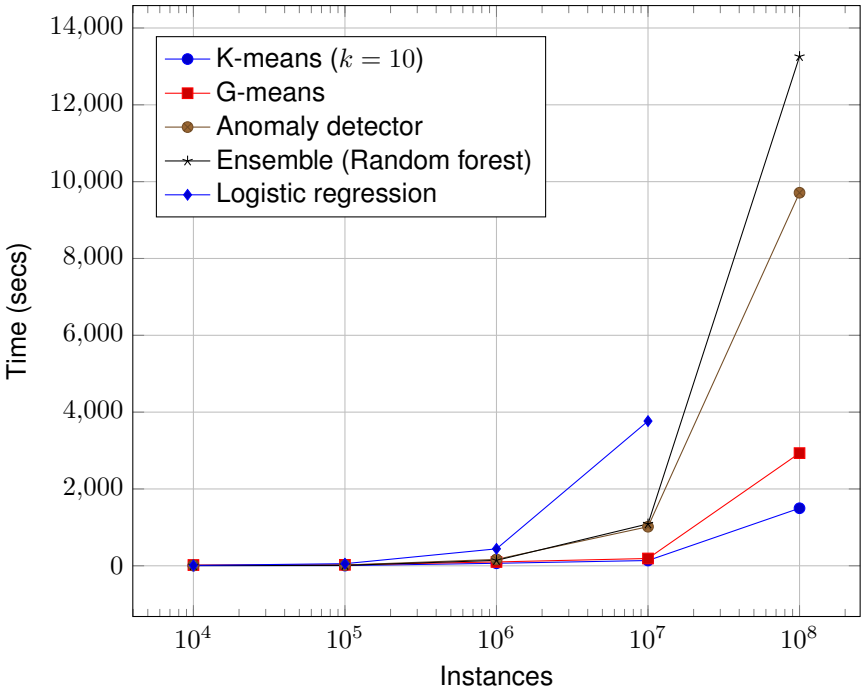

Figure 2.1: Timings for ML algoritms in i7 local server

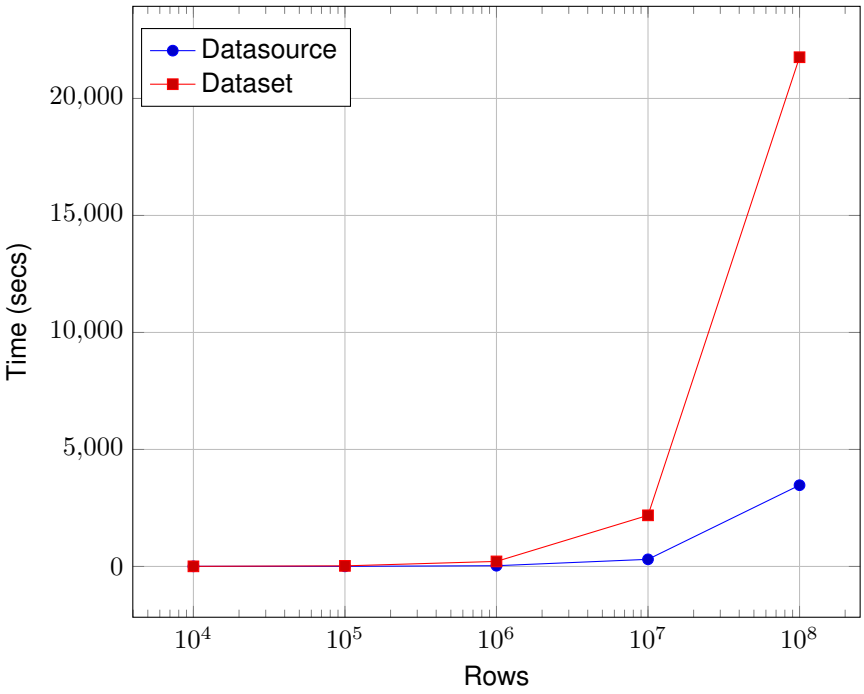

Figure 2.2: Timings for sources and datasets in i7 local server

CHAPTER 3

Cloud Servers (Amazon EC2)

Timings listed in this chapter were taken using three different Amazon EC2 server instances, memory-optimized, as described in Section 1.1 and Table 1.1. As with local server benchmarks (cf. Chapter 2), we use a varying row number and field number for the algorithm inputs. Thus, in the tables below, we have timings for every combination of instance type, row size, and field count (16, 64, and 256).

3.1 Datasource

Instances	r3.2xlarge			r3.4xlarge			r3.8xlarge		
	16	*64*	*256*	*16*	*64*	*256*	*16*	*64*	*256*
10000	5	9	30	4.5	9	31	4.5	9	32
100000	5	18	71	6	19	74	6	19	74
1000000	32	166	481	34	121	512	31	115	459
10000000	225	923		215	836	3596	223	866	3385

Table 3.1: Sources timings for EC2 instances

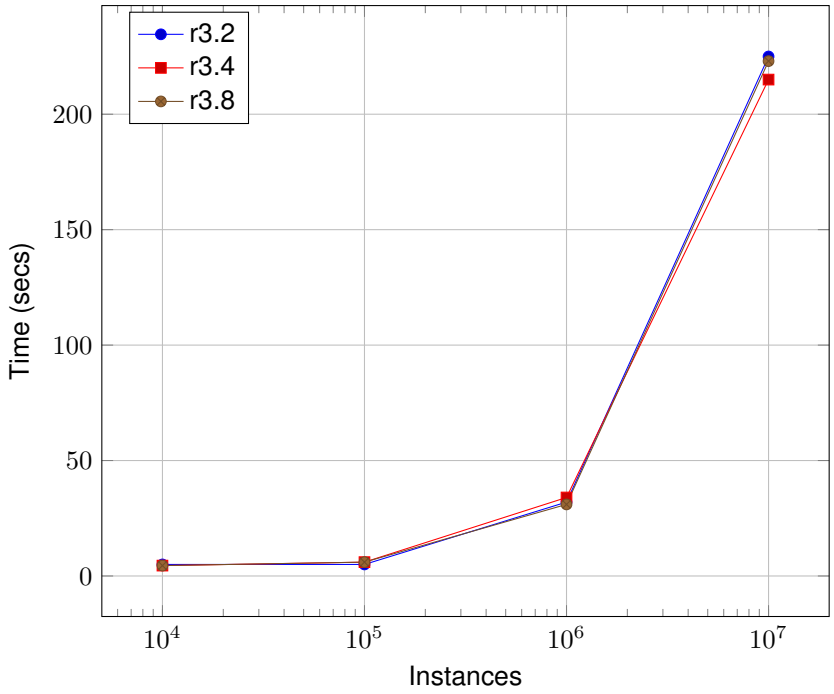

Figure 3.1: Timings for Sources (16 fields) in EC2 instances

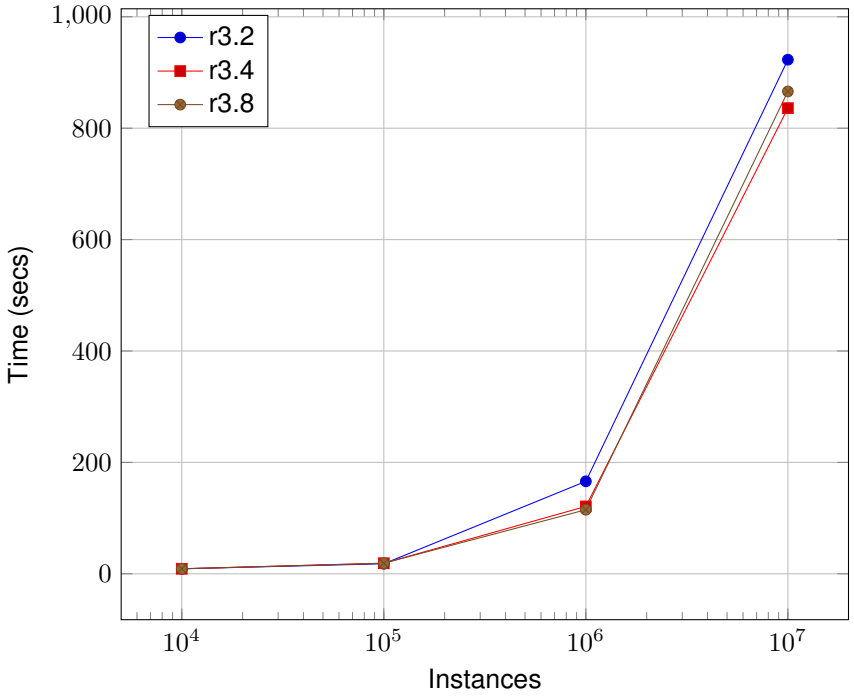

Figure 3.2: Timings for Sources (64 fields) in EC2 instances

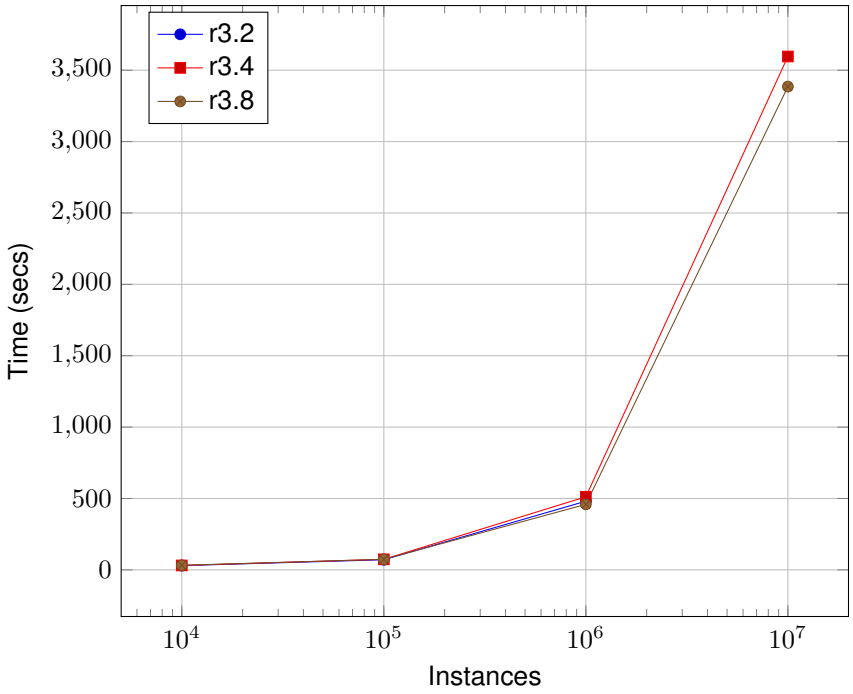

Figure 3.3: Timings for Sources (256 fields) in EC2 instances

3.2 Dataset

Instances	r3.2xlarge			r3.4xlarge			r3.8xlarge		
	16	*64*	*256*	*16*	*64*	*256*	*16*	*64*	*256*
10000	5	15	86	5	15	76	10	20	101
100000	25	105	831	25	95	723	30	100	677
1000000	205	1004	8020	190	898	7126	195	909	6777
10000000	1552	5749		1393	4785	20595	1675	4604	20358

Table 3.2: Datasets timings for EC2 instances

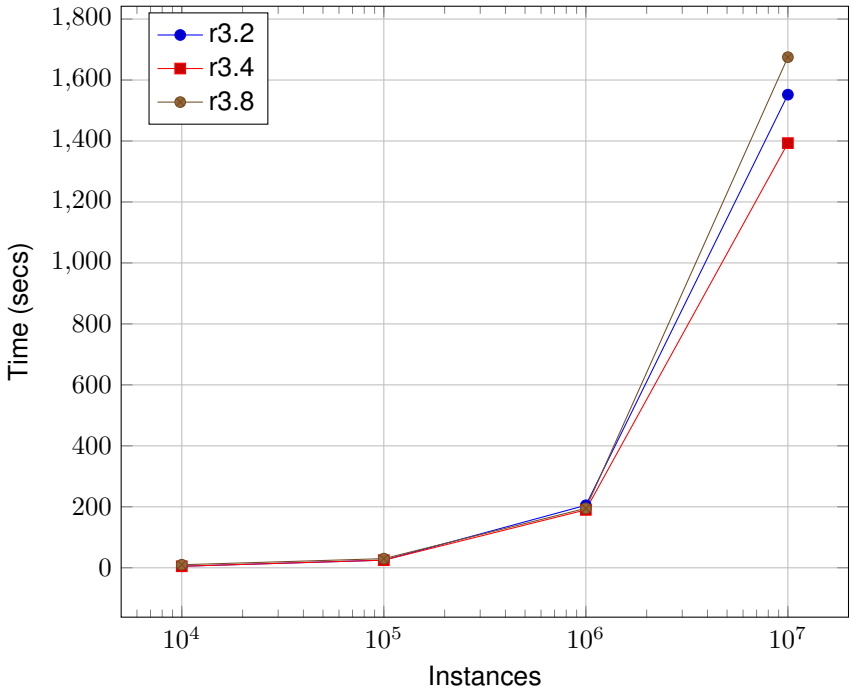

Figure 3.4: Timings for Datasets (16 fields) in EC2 instances

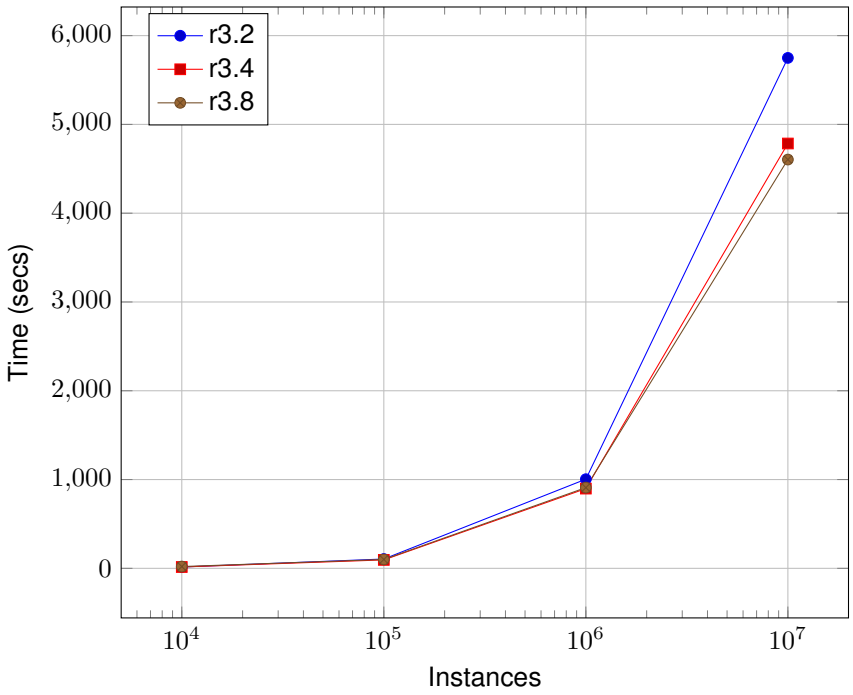

Figure 3.5: Timings for Datasets (64 fields) in EC2 instances

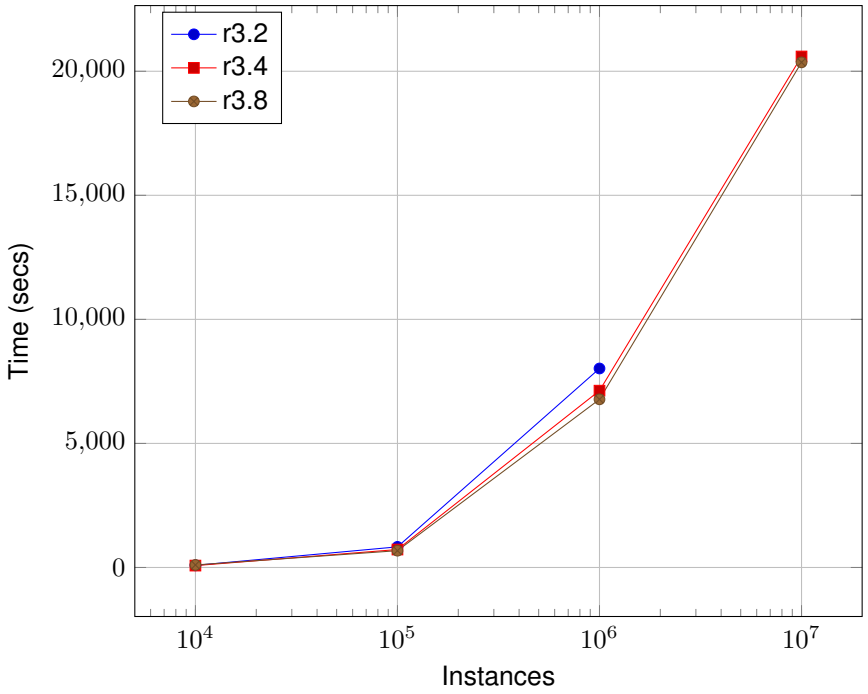

Figure 3.6: Timings for Datasets (256 fields) in EC2 instances

3.3 K-Means

This section provides benchmarks for creating BigML k-means clusterings using a k of 10. For k-means, the more cores are available, the more alternate clusterings are tested (as k-means is stocastic). So more cores does not necessarily lead to a faster runtime, but will often mean a better quality set of clusters.

Instances	r3.2xlarge			r3.4xlarge			r3.8xlarge		
	16	*64*	*256*	*16*	*64*	*256*	*16*	*64*	*256*
10000	5	5	10	5	5	10	5	5	5
100000	5	5	20	5	5	15	5	10	20
1000000	10	36	137	70	30	112	76	90	124
10000000	105	324		80	277	1153	80	270	1144

Table 3.3: K-Means timings for EC2 instances

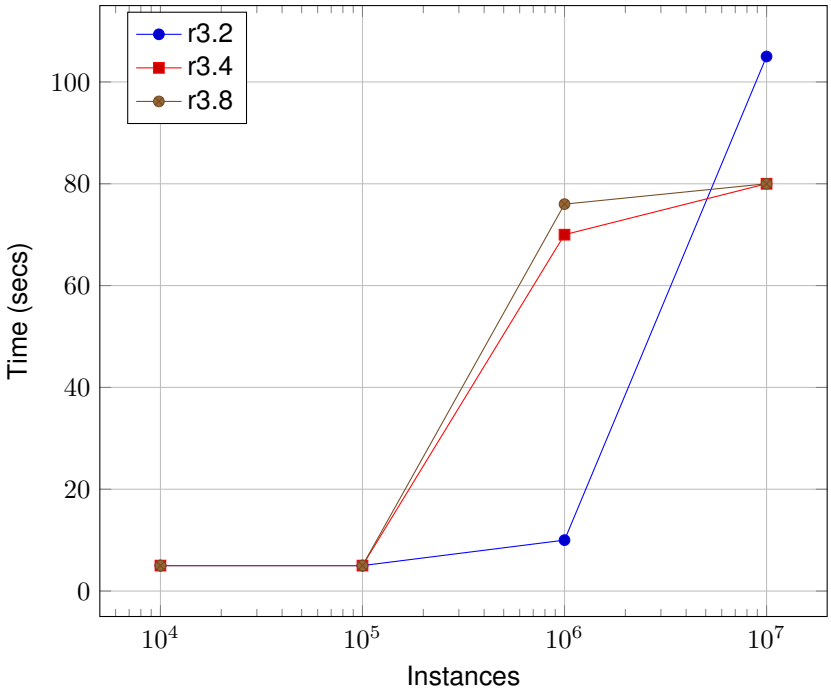

Figure 3.7: Timings for K-Means (16 fields) in EC2 instances

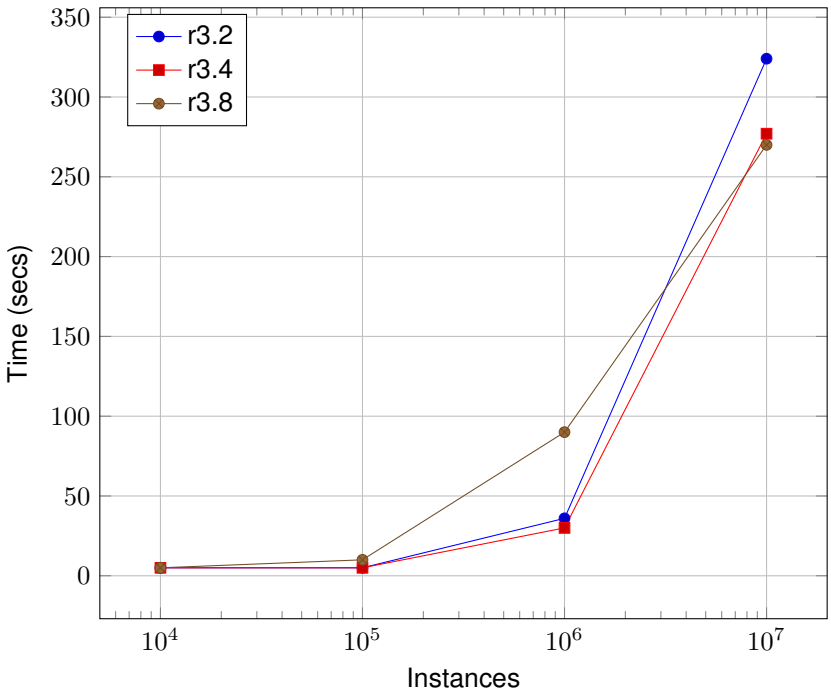

Figure 3.8: Timings for K-Means (64 fields) in EC2 instances

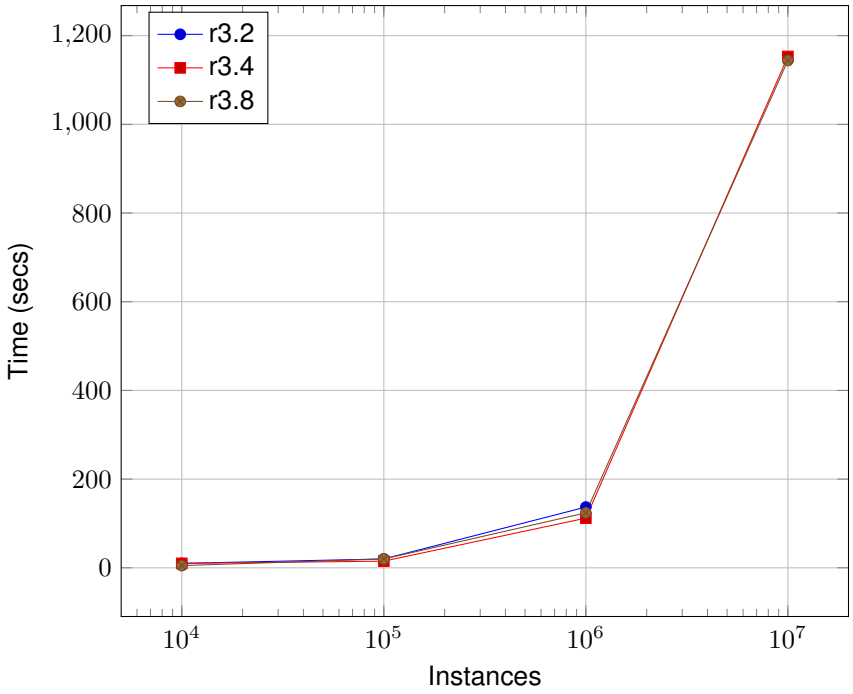

Figure 3.9: Timings for K-Means (256 fields) in EC2 instances

3.4 G-Means

This section provides benchmarks for creating BigML k-means clusterings. For g-means, the more cores are available, the more alternate clusterings are tested (as our implementation of g-means is stocastic). So more cores does not necessarily lead to a faster runtime, but will often mean a better quality set of clusters.

Instances	r3.2xlarge			r3.4xlarge			r3.8xlarge		
	16	*64*	*256*	*16*	*64*	*256*	*16*	*64*	*256*
10000	20	35	202	20	35	121	20	35	212
100000	30	60	268	30	85	212	30	85	258
1000000	50	150	666	120	126	556	120	186	561
10000000	242	946		214	277	4133	255	727	4766

Table 3.4: G-Means timings for EC2 instances

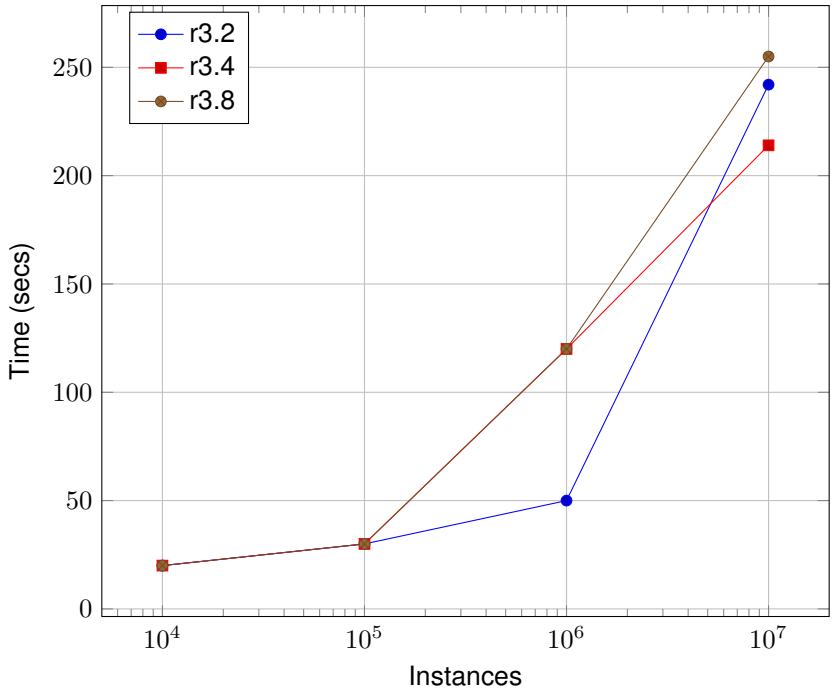

Figure 3.10: Timings for G-Means (16 fields) in EC2 instances

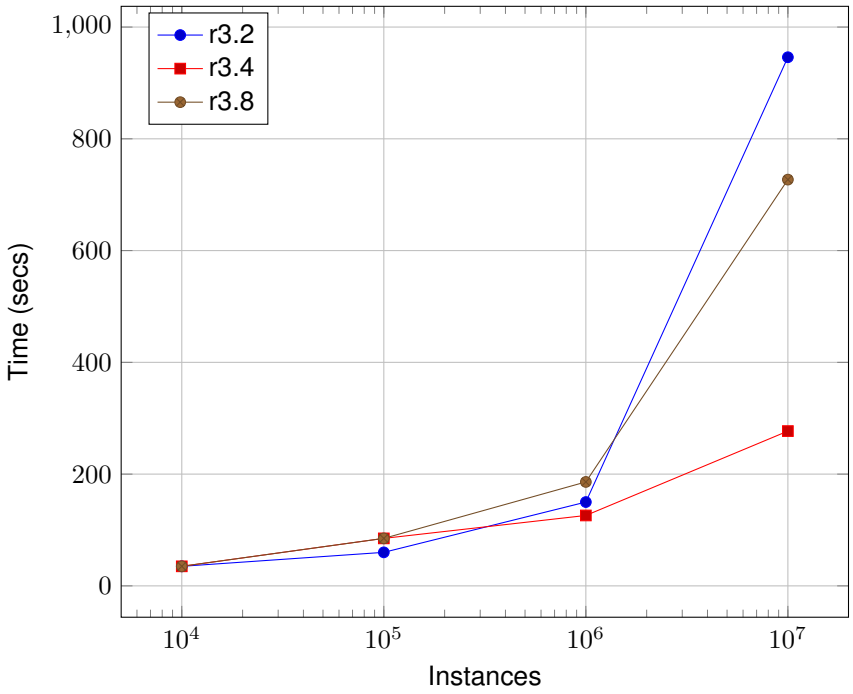

Figure 3.11: Timings for G-Means (64 fields) in EC2 instances

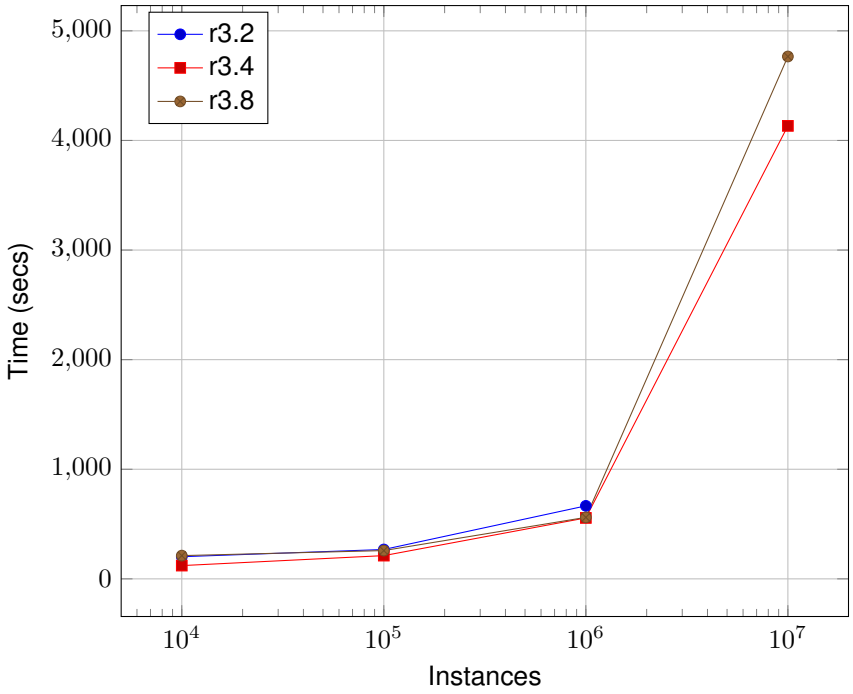

Figure 3.12: Timings for G-Means (256 fields) in EC2 instances

3.5 Anomaly Detector

This section provides benchmarks for creating BigML anomaly detectors, based on isolation forests containing 128 trees. The creation of the forests themselves only takes a small portion of the runtime, but we automatically apply the forest to the training data to find the top anomalies. Scoring to find the top anomalies takes the majority of the runtime. Note that generally the more fields in the data, the more trees needed to detect quality anomalies. We did not vary the number of trees in this benchmark, however.

Instances	r3.2xlarge			r3.4xlarge			r3.8xlarge		
	16	*64*	*256*	*16*	*64*	*256*	*16*	*64*	*256*
10000	10	10	10	10	10	10	10	10	10
100000	20	20	30	10	15	20	15	15	20
1000000	115	131	196	140	101	160	120	149	132
10000000	1005	1258		697	891	1556	501	649	1203

Table 3.5: Anomalies timings for EC2 instances

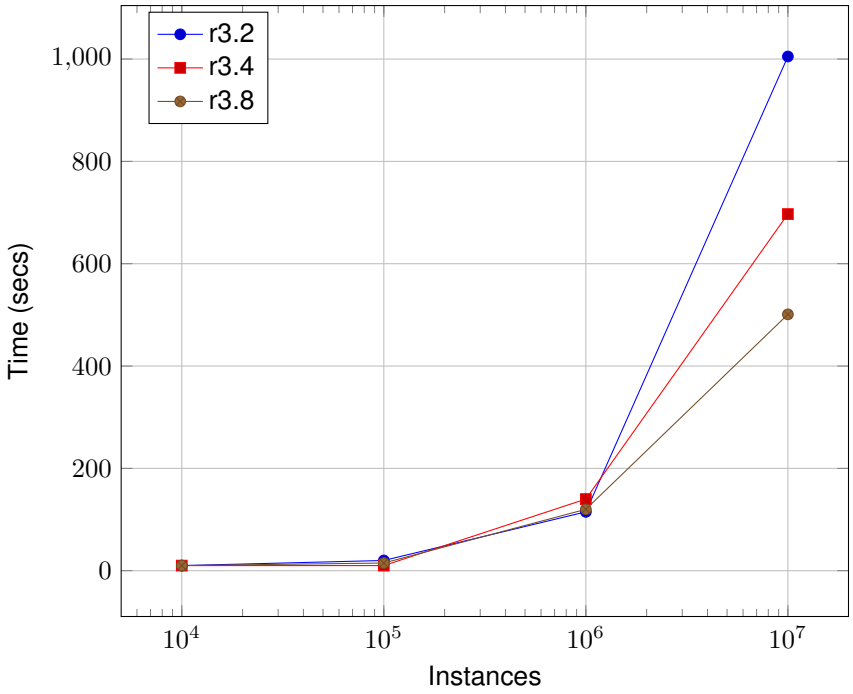

Figure 3.13: Timings for Anomalies (16 fields) in EC2 instances

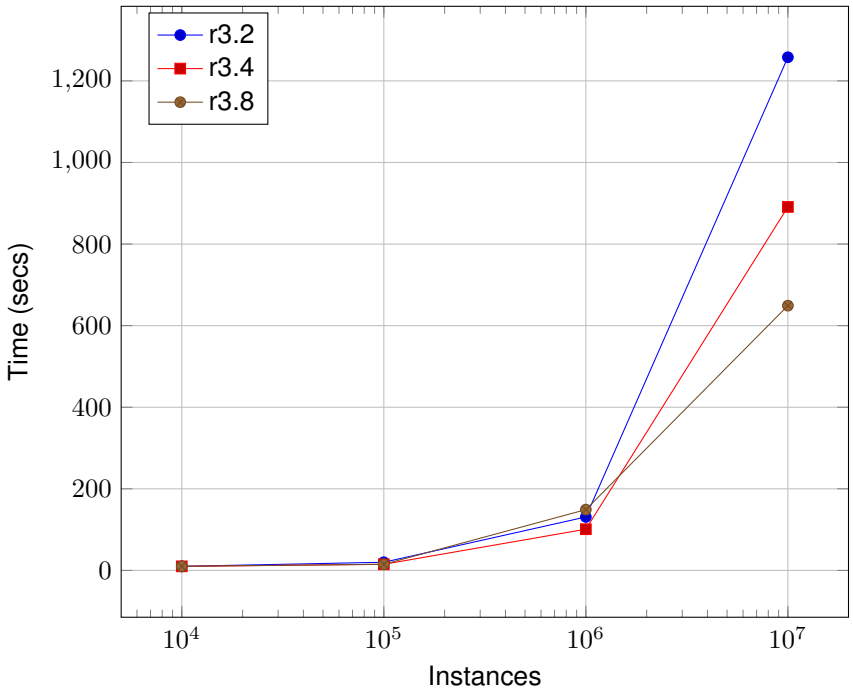

Figure 3.14: Timings for Anomalies (64 fields) in EC2 instances

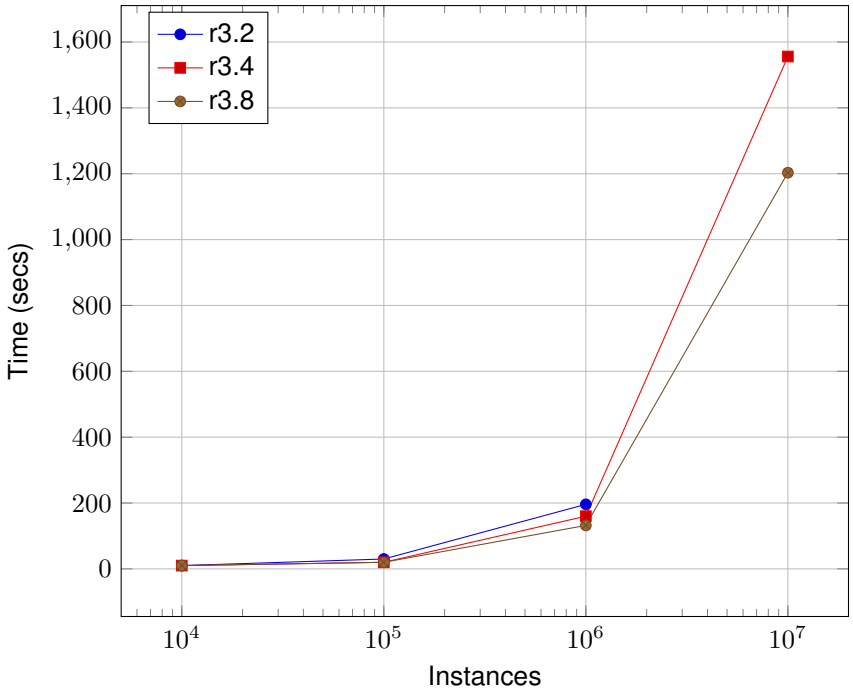

Figure 3.15: Timings for Anomalies (256 fields) in EC2 instances

3.6 Ensembles

This section provides benchmarks for creating BigML random decision forests containing 128 trees each with a limit of 10,000 nodes.

Instances	r3.2xlarge			r3.4xlarge			r3.8xlarge		
	16	*64*	*256*	*16*	*64*	*256*	*16*	*64*	*256*
10000	5	10	41	5	10	26	5	10	31
100000	15	60	253	10	35	149	10	35	176
1000000	117	387	885	125	212	476	106	213	365
10000000	1468	3271		748	752	3494	451	1111	2293

Table 3.6: Ensembles timings for EC2 instances

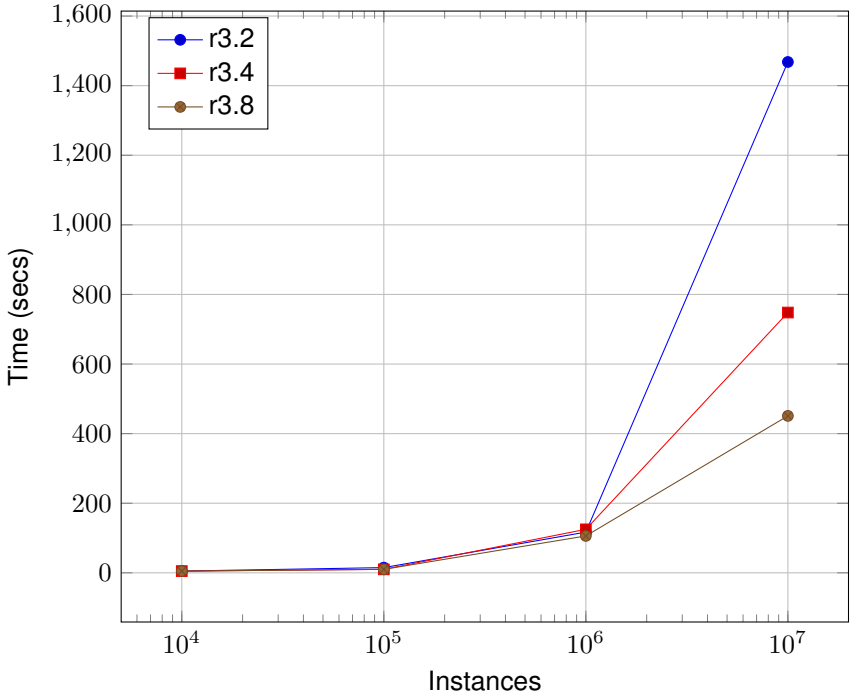

Figure 3.16: Timings for Ensembles (16 fields) in EC2 instances

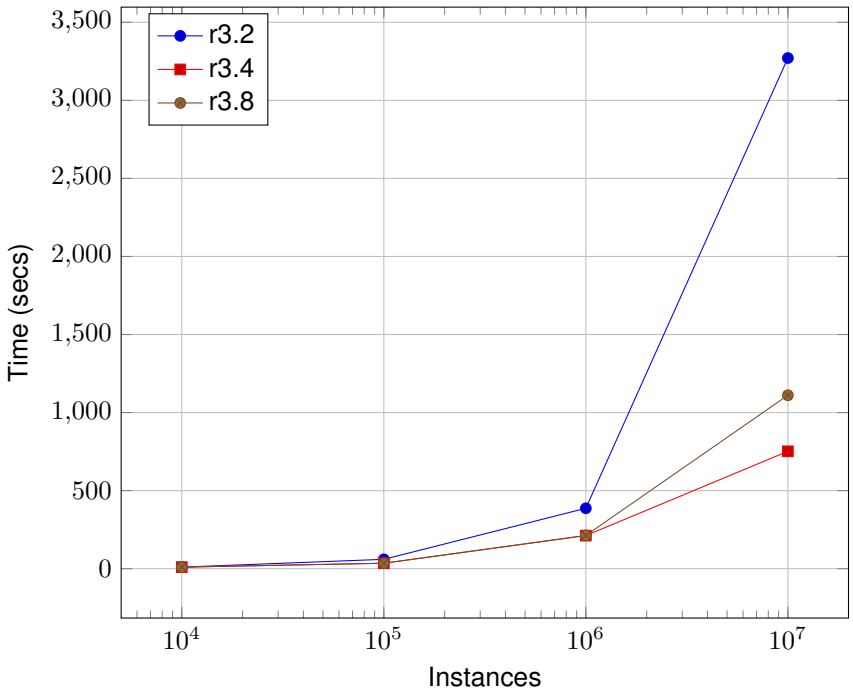

Figure 3.17: Timings for Ensembles (64 fields) in EC2 instances

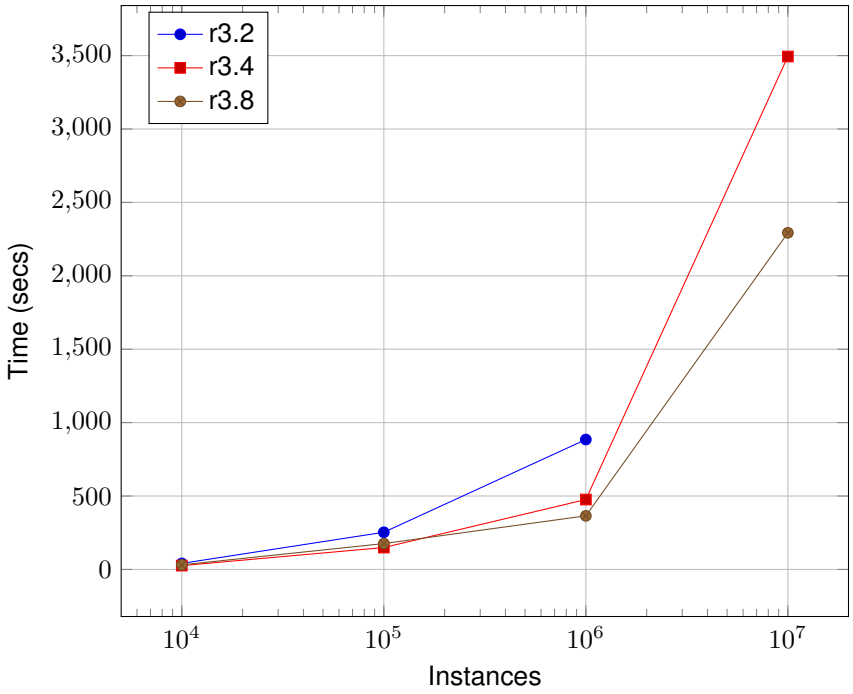

Figure 3.18: Timings for Ensembles (256 fields) in EC2 instances

3.7 Logistic Regression

This section provides benchmarks for creating BigML logistic regression models using a stopping criterion (eps) of 0.02. Logistic regression is single threaded, so the availability of more cores does not improve the runtime.

Instances	r3.2xlarge			r3.4xlarge			r3.8xlarge		
	16	*64*	*256*	*16*	*64*	*256*	*16*	*64*	*256*
10000	10	35	131	5	35	45	5	20	61
100000	100	95	223	100	95	602	50	261	364
1000000	496	502	1162	946	467	1120	1097	1209	1233
10000000	2436	5529		2401	5349		3260	7418	16002

Table 3.7: Logistic-Regression timings for EC2 instances

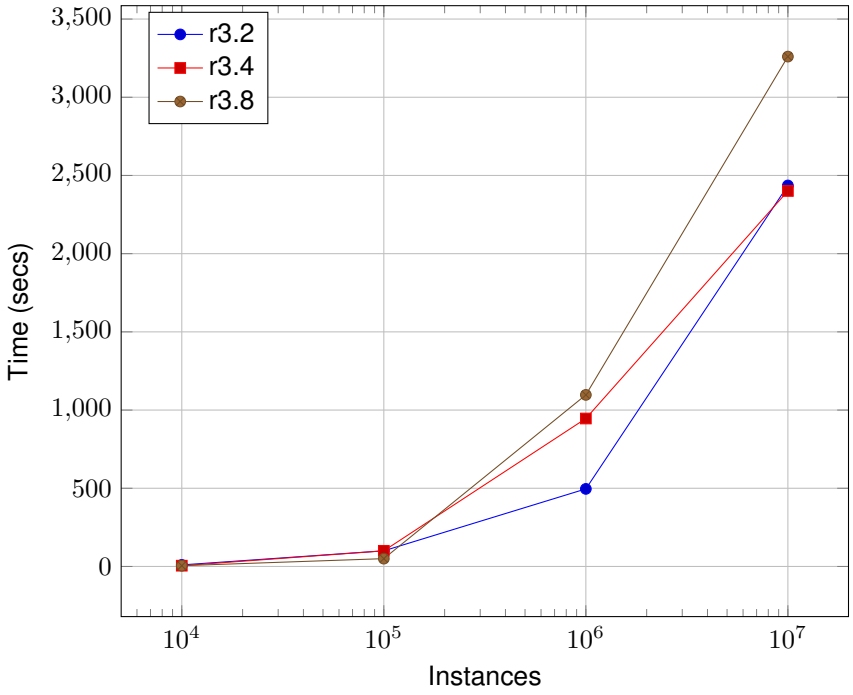

Figure 3.19: Timings for Logistic-Regression (16 fields) in EC2 instances

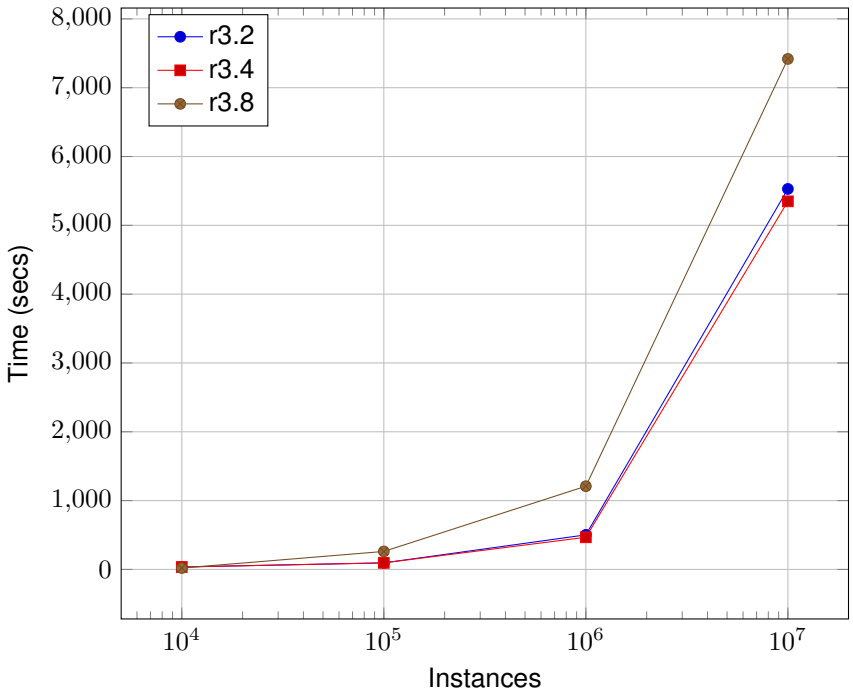

Figure 3.20: Timings for Logistic-Regression (64 fields) in EC2 instances

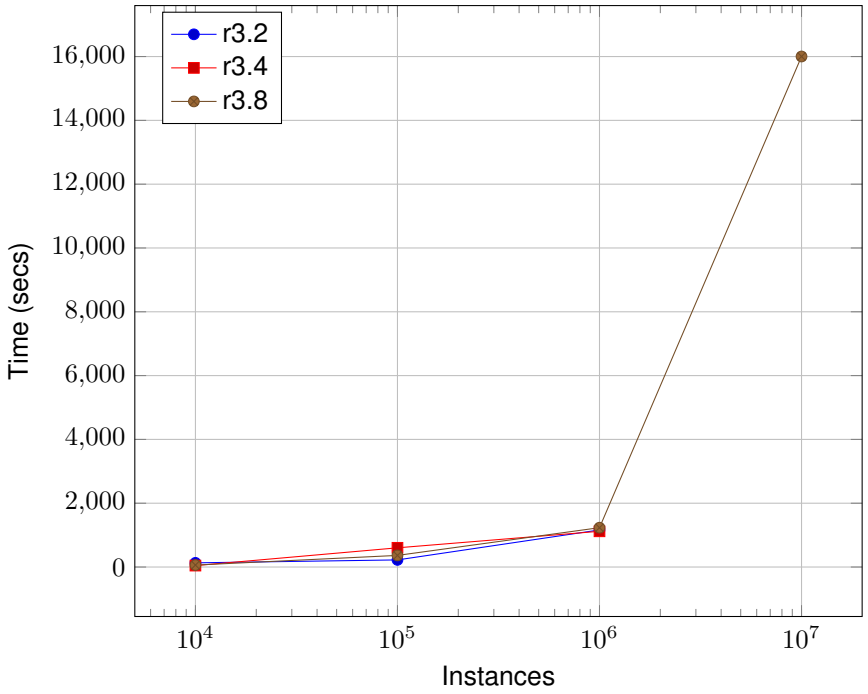

Figure 3.21: Timings for Logistic-Regression (256 fields) in EC2 instances

3.8 Association Discovery

This section provides benchmarks for BigML association discovery, finding the top 100 rules by leverage. The association discovery is single threaded, so the availability of more cores does not improve the runtime.

Instances	r3.2xlarge		r3.4xlarge		r3.8xlarge	
	16	*64*	*16*	*64*	*16*	*64*
10000	5	25	5	25	5	25
100000	15	40	15	40	15	45
1000000	140	327	200	306	215	383
10000000	1853	3535	1825	3261	2087	3547

Table 3.8: Association Discovery timings for EC2 instances

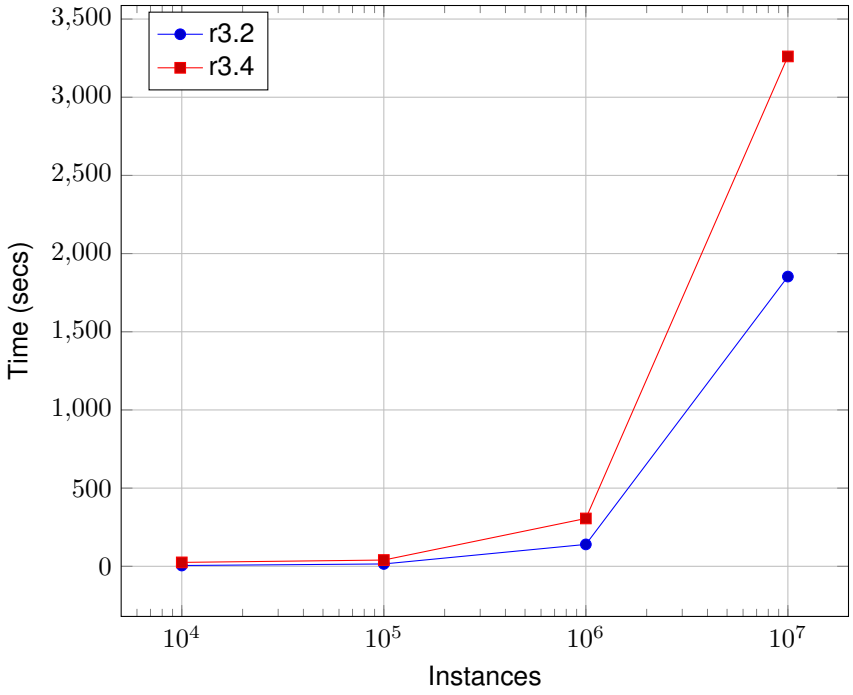

Figure 3.22: Timings for association discovery (16 fields) in EC2 instances

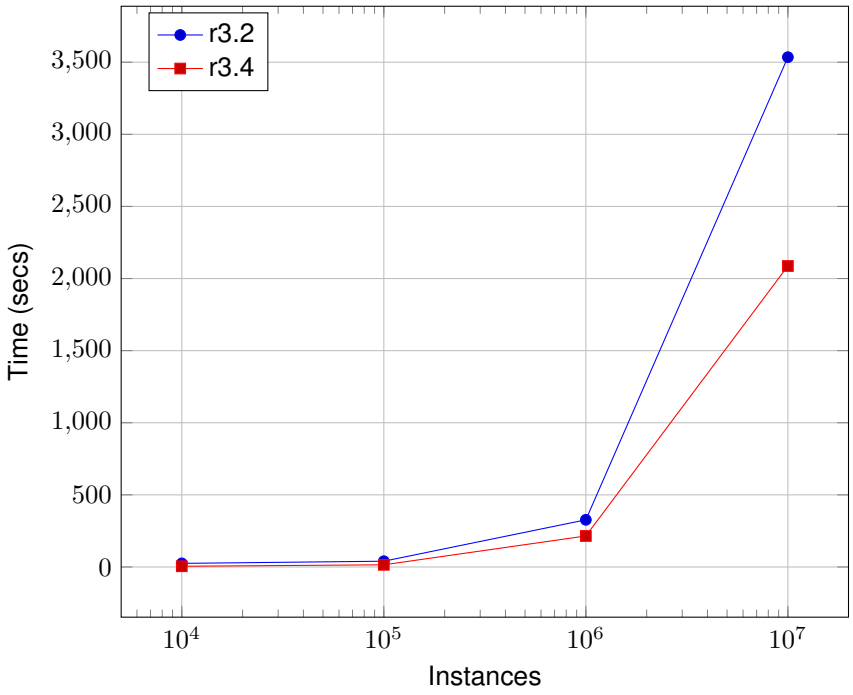

Figure 3.23: Timings for association discovery (64 fields) in EC2 instances

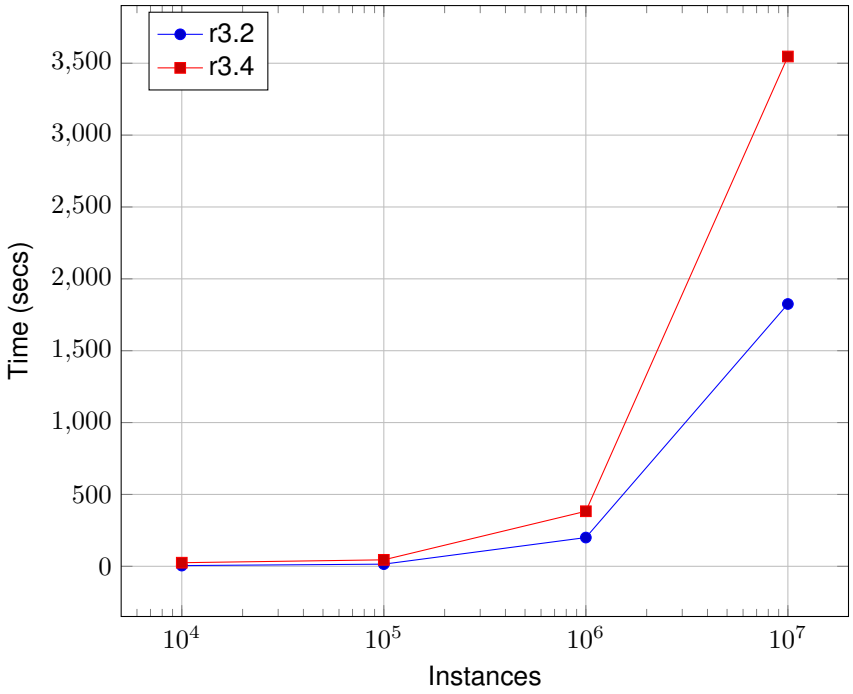

Figure 3.24: Timings for association discovery (64 fields) in EC2 instances

List of Figures

List of Tables


bigml