Earthquake Prediction based on Seismic Signals using Machine Learning and Distributed Computing

Anush Kocharyan, Jyoti Prakash Maheswari,
Viviana Márquez, Wenkun Xiao
MSDS 697 - University of San Francisco

January 18th, 2019

1 Introduction

Forecasting earthquakes is a subject of importance to both scientists and legislators. Current knowledge estimates that millions of earthquakes occur globally per year, and even though most of them are imperceptible, some can kill thousands of people and cause massive destruction. This underscores the importance of investing in research time and resources in predicting them.

At present time, scientists can only estimate the probability that an earthquake will occur based its recurrence interval, but since the recurrence interval of earthquakes is not constant, prediction using this approach results in large error bounds. The global community ambitions that one day scientists will learn to predict them with exactitude in order to minimize their destruction and save many lives.

A successful forecast of an earthquake consists of three elements: when an earthquake will happen, where it will occur, and how powerful it will be. In this report, we will focus on the when. Specifically, we will estimate when an earthquake will happen given a seismic signal using machine learning and distributed computing.

2 Overview

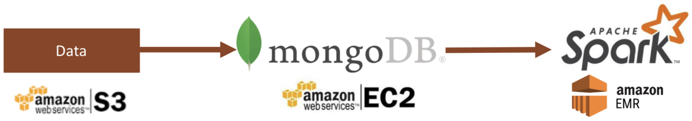
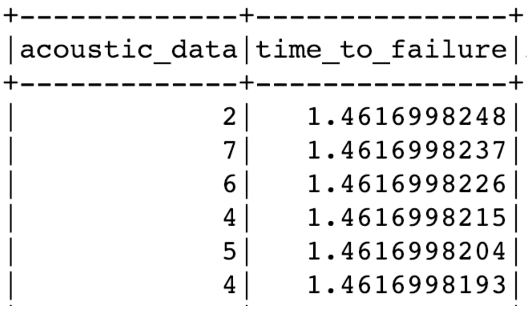


Figure 1: Work flow pipeline

3 Data on S3

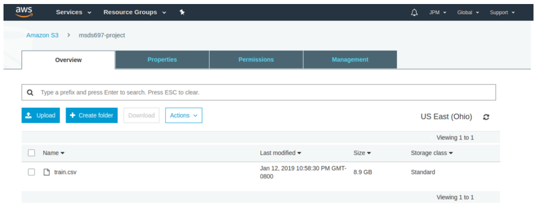
In this report we will be working with data obtained from *kaggle* produced by the Los Alamos National Laboratory. The data set is a 8.9GB *.csv* file containing almost 700,000,000 rows of experimental data. It is composed of two columns: `acoustic_data` and `time_to_failure`.



```
+-------------+---------------+
|acoustic_data|time_to_failure|
+-------------+---------------+
|            2|   1.4616998248|
|            7|   1.4616998237|
|            6|   1.4616998226|
|            4|   1.4616998215|
|            5|   1.4616998204|
|            4|   1.4616998193|
```

Figure 2: Data appearance

Our goal is to predict how much time there is until the next earthquake is going to hit based on seismic signals. We decided to work with this data because San Francisco is near the San Andreas Fault and six other significant fault zones. Due to the dense and vertical layout of the city, earthquake detection is important for our community.

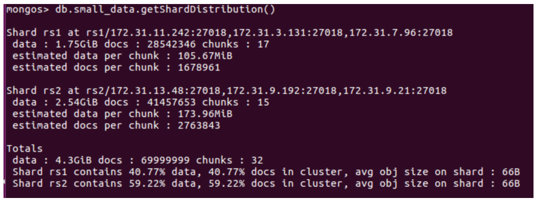


	Name	Last modified	Size	Storage class
□	train.csv	Jan 12, 2019 10:58:30 PM GMT-0800	8.9 GB	Standard

Figure 3: Data on S3 bucket

4 MongoDB on AWS EC2s

For the purpose of this project, we will be working with 1GB of the data. We created a MongoDB cluster with two shards and each one with three replicas on Amazon Web Services (AWS). Subsequently, we loaded the data from our S3 bucket to MongoDB.



```
mongos> db.small_data.getShardDistribution()

Shard rs1 at rs1/172.31.11.242:27018,172.31.3.131:27018,172.31.7.96:27018
 data : 1.75GiB docs : 28542346 chunks : 17
 estimated data per chunk : 105.67MiB
 estimated docs per chunk : 1678961

Shard rs2 at rs2/172.31.13.48:27018,172.31.9.192:27018,172.31.9.21:27018
 data : 2.54GiB docs : 41457653 chunks : 15
 estimated data per chunk : 173.96MiB
 estimated docs per chunk : 2763843

Totals
 data : 4.3GiB docs : 69999999 chunks : 32
 Shard rs1 contains 40.77% data, 40.77% docs in cluster, avg obj size on shard : 66B
 Shard rs2 contains 59.22% data, 59.22% docs in cluster, avg obj size on shard : 66B
```

Figure 4: MongoDB on AWS EC2s

The first shard contains 40% of the data and the second one the remaining 60%. While doing this process, our original 1GB data increased its size four times. As to the instance specifics, we obtained six *t2.medium* 8GB servers for the shards and their replicas, and four *t2.small* 8GB servers for the configuration and mongos.

5 SparkSQL on AWS EMR

5.1 Machine learning algorithms

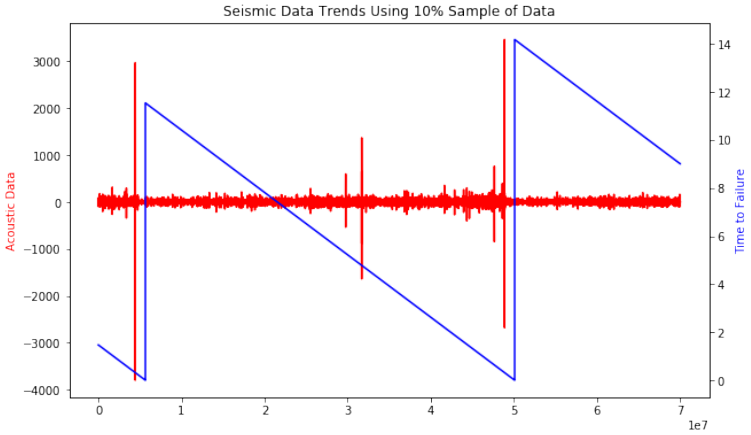


Figure 5: Exploratory data analysis

Since we are dealing with continuous output, we decided to apply linear regression, decision tree, and random forest algorithms.

According to the domain experts, a good way to tackle this problem is to aggregate data every 150,000 rows and use descriptive statistics (such as minimum, maximum, average, etc.) as features to predict when the next earthquake will strike.

For that reason, we started by exporting our data from MongoDB to feature engineer the respective characteristics. Next, we imported those features back to MongoDB on a new collection in the same database. We created a new collection with 340,000 data points as our feature training data set.

Afterwards, we exported the new collection to the Spark ML AWS EMR where we conducted the machine learning algorithms. In addition, we split the data in a 8:2 training to test ratio data; then we performed grid search and cross validation to find the best hyper-parameters.

5.2 Results

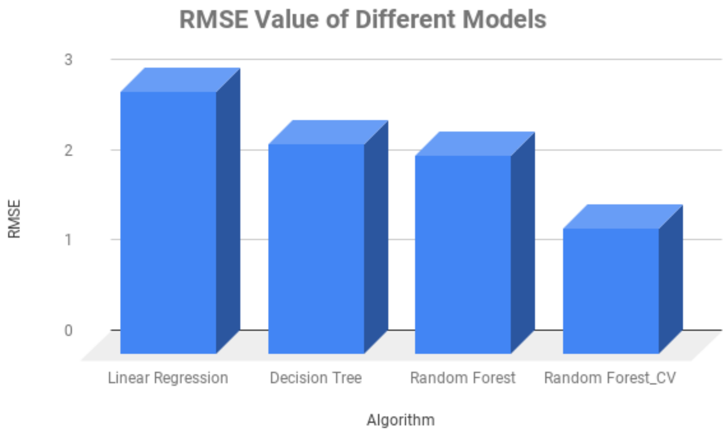


Figure 6: Root mean squared errors

In our experiment we found that Random Forest was the best performing algorithm. For that reason, we decided to run cross validation to further improve our results. Our lowest RMSE score is 1.40, which is a significant improvement from our previous score of 2.20 for Random Forest with default parameters.

5.3 Instance specs

We ran our experiments in four clusters and obtained the following speed results:

Instance/Execution time	Decision Tree	Linear Regression	Random Forest
m4.2xlarge	1.68s	1.10s	5.13s
m4.xlarge	21.9s	2.39s	5.81s
m4.large	57.9s	4.15s	10.5s
m3.xlarge	2.93s	1.73s	7.47s

6 Lessons learned

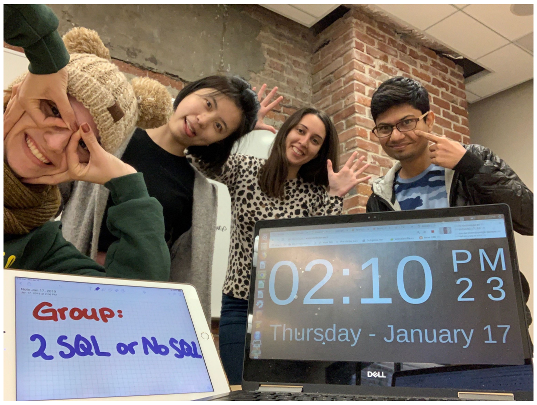


Figure 7: 2SQL or NoSQL, that's the query

In this project, we learned how to store data using MongoDB on Amazon Web Services. We did this by creating two shards and three replicas, in that way, we distributed the data and mirrored the data to obtain fault-tolerance in our project.

We spent a significant amount of time setting up EMR cluster to run Spark in Jupyter Notebook. In the process, we learned how to install various dependencies on EMR cluster and configuring various profile values so that we could start our machine learning experiments.

In addition, we gained experience with SparkSQL and Spark ML on a real life project. More importantly, we also learned how to be patient and perseverant when working on real life data science projects.