EARTHQUAKE PREDICTION

Dharma Teja Bandaru
Department of Computer Science
New Mexico State University
Las Cruses, New Mexico
dharma@nmsu.edu

Mahlet Zemedie
Department of Computer Science
New Mexico State University
Las Cruses, New Mexico
mayiizem@nmsu.edu

Shah Muhammad Hamdi
Department of Computer Science
New Mexico State University
Las Cruses, New Mexico
shamdi1@nmsu.edu

***Abstract*—Forecasting earthquakes is one of the most important problems in Earth science because of their devastating consequences. This project is from Kaggle competition to predict the time remaining before laboratory earthquakes occur from real-time seismic data, and It is hosted by Los Alamos National Labratory.**

I. Introduction

Man has always looked to computer systems for help in every area of human endeavours and even in understanding very complex systems as the human ability is limited in handling the amount of data generated every day. As the size of this data has increased considerably, the need to explore the use of computer system to learn from the data became imperative. Machine learning [1] , provides computer systems the ability to learn autonomously from experience and improve as well without being explicitly programmed. Natural disasters cause massive casualties, damages and leave many injure. Human beings cannot stop them, but timely prediction and due safety measures can prevent human life losses and many precious objects can be saved. Earthquake is one of the major catastrophes. Unlike other disasters, there is no specific mechanism for earthquake prediction, which makes it even more destructive. Though many of them say that it is impossible to make earthquake predictions, few Scientists declare that it is a predictable phenomenon.

II. Motivation

Believing that earthquakes could be predicted, from the past experiences we can say that we can avoid life loss and property damage if we have an estimation on severity of earthquake that will occur in near future.

On a yearly basis, there are approximately 14,000—16,000 fatalities due to earthquakes [2]. The count of fatalities per year due to earthquakes when compared to other disasters and accidents might be less. Whereas, an abrupt and a huge earthquake in a city could take away thousands of lives at once with huge loss of property as well. The solution mentioned in the project doesn't directly help in predicting the earthquake but it forms a good resource for the people who are actually working on it since the dataset used in the project is artificially created in the lab environment.

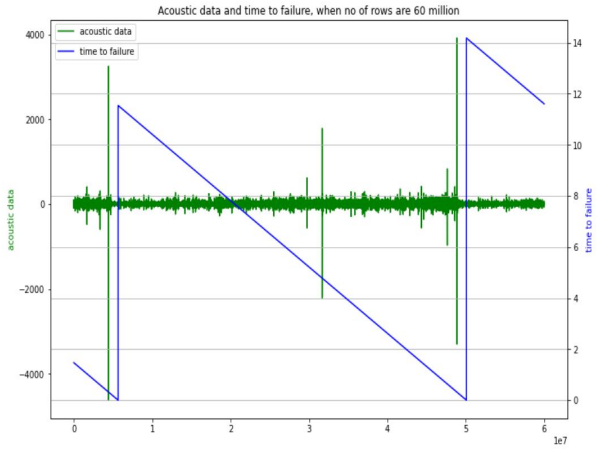
III. Problem Definition

What could be the solution? : Predicting the time left for the quake to occur when the acoustic signal is passed as input from the artificial created lab environment.

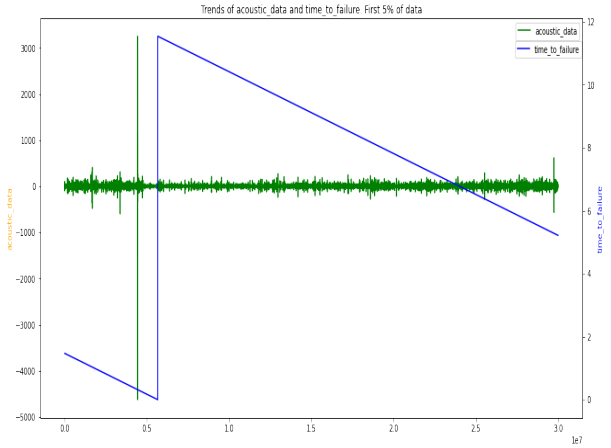
What are Solution Benefits? : As mentioned in the motivation the results from this project form a very good resource for the actual earthquake prediction. **How Would we solve the problem?** Predicting the time remaining before the next laboratory earthquake i.e.; "time to failure", for which we are using the "acoustic data" from the training data and can accomplish the task and Comparison of regression techniques performances.

IV. Data Analysis and Data Visualisation:

The data set is huge with 600 million rows of data with two columns namely "acoustic data", and "time to failure".
Acoustic data: It is the acoustic signal measured in the laboratory experiment.
Time to failure: The time left for a failure/next failure to occur. In the below graph, we have visualized the pattern for the 60 million data between the acoustic data and the time to failure. If we observe, whenever there is a spike in the activity of seismic data there is also a rise in the time to failure.



Based on the above graph, whenever there is a huge spike in time to failure, there is a earth quake, huge sesmic wave.i.e when ever there is a high in green, there is also a spike in blue.



The above graph shows us the 5% of the 60 million dataset, how the "time to failure" varies when the "acoustic data" has got a spike. When we try to plot the whole data set (600 million) the kernel were crashing. So we were only able to plot 10% and below of the data.

V. Data Pre-Processing

As we need to predict "time to failure" only with "acoustic data" present in test datasets, the data in the training dataset i.e.; acoustic data, time to failure alone won't train the model accurate enough. So, we use **feature engineering** and add few common statistical features like Mean, Min, Max, Standard Deviation, Kurtosis, Skew, Quantile to the data in dataset (**train.CSV**). Doing feature engineering will allow us to accurately represent the underlying structure of the data and therefore create the best model.

VI. Features Added

- **Mean:** Mean or Average is a central tendency of the data i.e. a number around which a whole data is spread out.
- **Median:** Median is the value which divides the data in 2 equal parts i.e. number of terms on right side of it is same as number of terms on left side of it when data is arranged in either ascending or descending order
- **Standard Deviation:** Standard deviation is the measurement of average distance between each quantity and mean.
- **Kurtosis:** It is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution. It also helps us in finding outliers.
- **Skew:** Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive or negative, or undefined.
- **Quantile:** Cut points dividing the range of a probability distribution into continuous intervals with equal probabilities or dividing the observations in a sample in the same way.

VII. Data Splitting

We are split the data set into 70% for training and 30% for testing. We are also performing some preprocessing and modifying the dataset considering the temporality of "time to failure" data.

The process of modifying the dataset: We are considering 600 million rows of data(complete data set) to train and test the model. Then dividing the data into 4000 separate chunks of data leaving 150,000 rows of data in each chunk. We find all the above-mentioned special features for all the 4000 chunks(600million/150000); Now each chunk has 150,000 data rows, for each chunk we calculate all the feature engineered statistics like (mean, median—etc) of the acoustic data and the new row is formed with all the feature engineered statistics and the time to failure for the new row is considered to be the time to failure of the last row of each chunk. Hence, we now have 4000 new rows of data with columns mean, min, max, median, standard deviation, Kurtosis, Skew, quantile and time to failure. Now as we have a better data to predict, we train the model, predict and obtain the MSE, R2 score values.

Similarly, we have also increased the sample size for training and testing by encapsulating 100000 rows into a chunk instead of 150000 which resulted in 6000 chunks which eventually resulted in around 6000 rows of data. We discussed the results observed in the result section.

Why not the mean of "time to failure" values instead of last row data?? We really can't consider the mean of "time to failure" values because the value of a mean of two different set of values will never be equal, hence all the 4000 values will be unique, and no two values are same. As we are considering the "time to failure" values to be as a label to predict we can't use mean value and hence we go with the last row data.

VIII. Implementation

We applied five different regression analysis to bring out the 'time to failure' values. We implemented Linear regression, Cataboost regression, Decision Tree regression, Support vector regression, and Random forest regression. We also applied three different Neural network models with different

structures, and Long short term memory (LSTM) [8] with three different LSTM blocks. In this section of the paper we explained the detail structure and architecture of the machine learning and neural network models we have implemented.

- **Regression:-** We can apply the regression and bring out the "time to failure" values. Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent and independent variable (s). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables. The regression methods [3] we are trying to implement are below:
  - **Linear Regression:** Linear regression models predict a continuous target when there is a linear relationship between the target and one or more predictors. In this method, the variable to be predicted depends on only one other variable. This is calculated by using the formula that is generally used in calculating the slope of a line.
  - **Decision Tree Regressor:**Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The result is a tree with decision nodes and leaf nodes.
  - **Random Forest Regressor:** A Random Forest [4] is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.
  - **Catboost Regressor:** CatBoost is a machine learning algorithm that uses gradient boosting on decision trees. It is available as an open source library. It's particularly useful for predictive models that analyse ordered (continuous) data and categorical data. Catboost model one of the most efficient ways to build ensemble models. The combination of gradient boosting with decision trees provides state-of-the-art results in many applications with structured data.
  - **Support Vector Regression:** Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. The main Idea is to minimize error, individualizing the hyperplane which maximizes the margin. The kernel functions transform the data into a higher dimensional feature space to make it possible to perform the linear separation.

  The reason behind selection of the models Linear regression, Decision Tree regressor and Random forest regressor is, these models are well known this days in the data science environment so we would like to test how these go on the dataset. We choose Catboost Regressor [5] since it is going very well in the current data science trend and reason for choosing Support Vector Regression is since support vectors are well defined and used in many regressions.

- **Neural Network Models:**A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operate. It consists of three layers namely Input layer, hidden layers, output layer. Input layer is used to take the input, hidden layer is the place where computations and decisions happen and output layer for the output.Along with the layers one other important component is activation functions to introduce non linearity in the neural networks. We used three different neural network models with different structure. The input data fed into the input layer of neural networks is z-normalized to maintain common scale among the items of the columns.
  - **Neural network model 1:** Neural Network Model with three hidden layer and each hidden layer consists of 256 neurons. The activation function used is relu. The loss function used is 'mean squared error. The input layer contains 128 neurons and activation function is relu. The output layer contains only one neuron and activation function is linear.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               1408
_________________________________________________________________
dense_2 (Dense)              (None, 256)               33024
_________________________________________________________________
dense_3 (Dense)              (None, 256)               65792
_________________________________________________________________
dense_4 (Dense)              (None, 256)               65792
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 257
=================================================================
Total params: 166,273
Trainable params: 166,273
Non-trainable params: 0
```

  - **Neural network model 1 with batch normalization:** We have replicated the above neural networks with batch normalization in an assumption to increase the stability of the neural network since batch normalization reduces the amount of shift of hidden layers than the usual.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 128)               1408
_________________________________________________________________
batch_normalization_3 (Batch (None, 128)               512
_________________________________________________________________
dense_13 (Dense)             (None, 256)               33024
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
dense_14 (Dense)             (None, 256)               65792
_________________________________________________________________
batch_normalization_5 (Batch (None, 256)               1024
_________________________________________________________________
dense_15 (Dense)             (None, 256)               65792
_________________________________________________________________
batch_normalization_6 (Batch (None, 256)               1024
_________________________________________________________________
dense_16 (Dense)             (None, 1)                 257
_________________________________________________________________
batch_normalization_7 (Batch (None, 1)                 4
=================================================================
Total params: 169,861
Trainable params: 168,067
Non-trainable params: 1,794
```

– **Neural network model2:** Neural Network Model with three hidden layer and each hidden layer consists of 512 neurons. The activation function used is relu. The loss function used is 'mean squared error', and input layer contains 128 neurons. The output layer contains only one neuron and activation function is linear.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 256)               2816
_________________________________________________________________
dense_7 (Dense)              (None, 512)               131584
_________________________________________________________________
dense_8 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_9 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 513
=================================================================
Total params: 660,225
Trainable params: 660,225
Non-trainable params: 0
_________________________________________________________________
```

– **Neural network model3:** Neural Network Model with three hidden layer and each hidden layer consists of 128 neurons. The loss function used is 'mean squared error'. The input layer contains 64 neurons and activation function is relu. The output layer contains only one neuron and activation function is linear.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_11 (Dense)             (None, 64)                704
_________________________________________________________________
dense_12 (Dense)             (None, 128)               8320
_________________________________________________________________
dense_13 (Dense)             (None, 128)               16512
_________________________________________________________________
dense_14 (Dense)             (None, 128)               16512
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 129
=================================================================
Total params: 42,177
Trainable params: 42,177
Non-trainable params: 0
_________________________________________________________________
```

- **Long short-term memory (LSTM)**: is an artificial recurrent neural network (RNN) architecture used in the field of deep learning.It's a recurrent Neural Network trained using back propagation through time and overcomes the vanishing gradient problem. Instead of neurons LSTM networks have memory blocks that are connected through layers.Three gates within a unit:
  Forget Gate: Decides what information to throw away
  Input Gate : Decides which values from input to update memory state
  Output Gate : Decides what to output based on input and memory of the block.
  We have used a standard scaler from scikit learn to transform the data.
  Standard Scalar: standardizes a feature by subtracting the mean and then scaling to unit variance. It is a part of the vast sci-kit learn library.
  Model 1:

  – 20 LSTM Blocks
  – Activation of LSTM: Tanh
  – 2 Dense layer with 20 & 1 units respectively.
  – Activation function of layer 1: relu
  – Activation function of layer 2: linear
  – Optimizer: Adam

  Model 2:

  – 10 LSTM Blocks
  – Activation of LSTM: Tanh
  – 2 Dense layer with 20 & 1 units respectively.
  – Activation function of layer 1: relu
  – Activation function of layer 2: linear
  – Optimizer: Adam

  Model 3:

  – 6 LSTM Blocks
  – Activation of LSTM: Tanh
  – 2 Dense layer with 20 & 1 units respectively.
  – Activation function of layer 1: relu
  – Activation function of layer 2: linear

– Optimizer: Adam

The only difference between the three models are the number of LSTM blocks used.

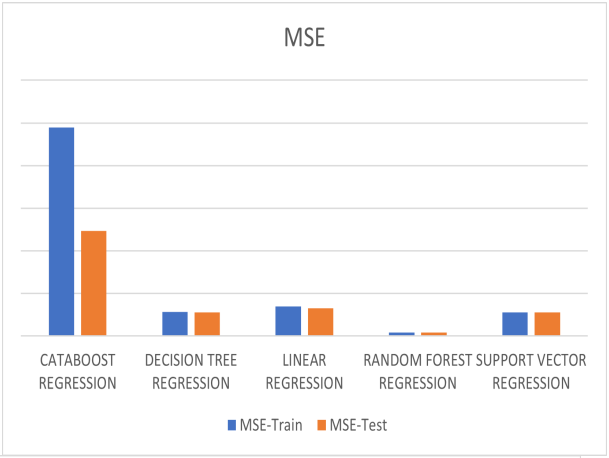
IX. Result

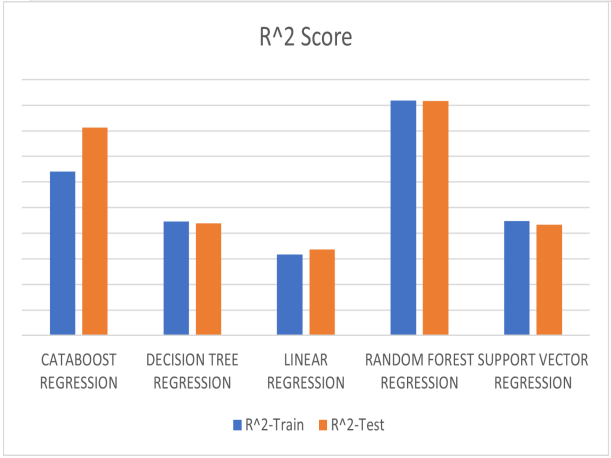
When the data is considered is 600 million rows, we have tuned the data in such a way that 600 million rows are compressed to 4000 rows(Using the Technique mentioned in section VII and we used 2800 rows for training and the remaining 1200 rows for testing. when the chuck size is 100,000, the number of data samples are 6000 rows and we used 4200 rows for training and 1800 rows for training.

After running the different machine learning algorithms, we measure the Mean squared error (MSE) and R-Squared (R^2) to get the average squared difference between the estimated values and the actual value, and to see how close the data are to the fitted regression line respectively. In both cases where the data is compressed to 4000 rows (150000 chunks) and 6000 rows (100000 chucks), the result was showing very similar.

Regression Model	MSE-Train	MSE-Test	R^2-Train	R^2-Test
Linear Regression	0.689	0.648	0.317	0.337
Decision-Tree Regression	0.56	0.549	0.445	0.438
Random Forest Regression	0.082	0.082	0.919	0.916
Catboost Regression	4.896	2.47	0.64	0.813
Support Vector Regression	0.557	0.554	0.448	0.433

TABLE I
COMPARISON BETWEEN THE REGRESSION PERFORMANCES OF THE MODELS





Random forest regressor with the least MSE score proved to be the best machine learning algorithm among the machine algorithms we have used. Cataboost regressor seems to have a high MSE score among the others. Apart from Cataboost all other algorithms have similar run times. Also Cataboost is the outlier among the machine learning algorithms.

Neural Network Model	MSE-Train	MSE-Validation	MSE Test
Model1	0.5592	0.3554	0.7068
Model2	0.4934	0.4276	0.6206
Model3	0.5578	0.3677	0.5921

TABLE II
COMPARISON BETWEEN THE NEURAL NETWORK PERFORMANCES OF THE THREE MODELS

Almost all the various configurations of the neural network models gave the same result. When we increase the number of epochs, it brought the training error down but it doesn't affect the testing data after 50 epochs in almost all the models. Also we noticed as normalization [7] matters a lot in bringing the optimum result. Even batch normalization doesn't much effected the results in reducing the mean squared loss, It gave analogous results to other models. While comparing the three differen nural network models, model one gave the optimum results [6].

- Neural Network Model with three hidden layer and each hidden layer consists of 256 neurons
- Activation function is relu
- The loss function used is 'mean squared error.
- The input layer contains 128 neurons and activation function is relu.
- The output layer contains only one neuron and activation function is linear.

All the LSTM models gave similar test accuracy, the model with 10 lstm blocks gave the least MSE Score. Input is always is a 3d array, and output can be 2d or 3d array depending on the returnsequences argument. Understanding the Input and output shapes in LSTM is not easy as Neural Networks.

LSTM Model	MSE-Train	MSE-Test
Model1	0.3871	0.614
Model2	0.3724	0.746
Model3	0.48	0.7

TABLE III
COMPARISON BETWEEN THE LSTM PERFORMANCES OF THE THREE DIFFERENT MODELS

X. Conclusion

The goal of this project was to predict the remaining time until an earthquake breaks out based on acoustic data from laboratory experiments. We used three different approaches to solve the problem: Machine learning algorithms, Neural network Models, and LSTM blocks. All the approaches gave almost similar results. Te machine learning algorithms were much easier to implement as they are already predefined. Also we had to deal with large data set. It is very time taking and not easy to deal with such huge dataset but definitely it was a good experience for us.

XI. Future Direction

As all of this is an environment created in a lab and the data is artificially created, they can implement this in the real-time environment at the places vulnerable to earthquakes by taking the 'acoustic data" from the soil layers there and predict the time left for it to occur.

References

[1] "What is Machine Learning? A definition," Expert System, 29-May-2020. [Online]. Available: https://expertsystem.com/machine-learning-definition/. [Accessed: 27-Nov-2020].

[2] Asim, Khawaja & Martínez-Álvarez, Francisco & Basit, Abdul & Iqbal, Talat. (2017). Earthquake magnitude prediction in Hindukush region using machine learning techniques. Natural Hazards. 85. 471-486. 10.1007/s11069-016-2579-3.

[3] "Learn regression algorithms using Python and scikit-learn," IBM Developer. [Online]. Available: https://developer.ibm.com/technologies/data-science/tutorials/learn-regression-algorithms-using-python-and-scikit-learn/. [Accessed: 27-Nov-2020].

[4] AvikDuttaCheck out this Author's contributed articles., AvikDutta, and Check out this Author's contributed articles., "Random Forest Regression in Python," GeeksforGeeks, 28-May-2020. Online. Available: https://www.geeksforgeeks.org/random-forest-regression-in-python/. [Accessed: 27-Nov-2020].

[5] "CatBoost: CatBoost Categorical Features," Analytics Vidhya, 07-Jun-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/08/catboost-automated-categorical-data/. [Accessed: 29-Nov-2020].

[6] @bookgoodfellow2016deep, title=Deep learning, author=Goodfellow, Ian and Bengio, Yoshua and Courville, Aaron and Bengio, Yoshua, volume=1, number=2, year=2016, publisher=MIT press Cambridge

[7] H. Ma, "Tech Services on the Web: Codecademy; http://www.codecademy.com/," Technical Services Quarterly, vol. 30, no. 3, pp. 338–339, 2013.

[8] J. Brownlee, "Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras," Machine Learning Mastery, 27-Aug-2020. [Online]. Available: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/. [Accessed: 13-Dec-2020].