Sahil Punchhi, z5204256
Rishabh Khurana, z5220427
Aarushi Gera, z5228145
Amritesh Singh, z5211987

COMP 9417, 2019T2

Laboratory Earthquake Prediction from real-time Seismic data

COMP 9417 Machine Learning Project

Introduction

Forecasting earthquakes is one of the most significant problems in Earth science because of their devastating consequences. Current scientific studies related to earthquake forecasting focus on three key points: when the event will occur, where it will occur, and how large it will be. In this project, an attempt has been made to address when the earthquake will take place. Specifically, the time remaining before laboratory earthquakes occur from real-time seismic data is predicted. The seismic signal originates from continuous grain motions of the fault gouge as the fault blocks displace. The laboratory test applies shear forces to a sample of earth and rock containing a fault line. If the physics are ultimately shown to scale from the laboratory to the field, researchers will have the potential to improve earthquake hazard assessments that could save lives and billions of dollars in infrastructure. The topic selected for the project is chosen from Kaggle competitions where Los Alamos National Laboratory (LANL) team has provided an interesting and a challenging dataset with notably more aperiodic earthquake failures and the objective is to predict the failures of each test data set.

Related Work

A lot of work has been going on at Los Alamos National Laboratory to study the physics which drives the geological faults which could help in predicting earthquakes more accurately. The collapse of stress chains inside the earthquake gouge gives rise to seismic signals in the lab which directs the study to the similar phenomenon taking place inside the Earth. This work was published in Physical Review Letters by Ke Gao, a computational geophysicist in the Geophysics group at LANL under the heading “From Stress Chains to Acoustic Emission”. Stresses are transmitted from one side of fault block to the other in bridges composed of grains called stress chains. According to recent developments, machine learning algorithms using simple decision trees have given a lot of insightful information to the researchers at LANL about

the principles of physics in the mechanics of earthquakes which even deep neural network failed to achieve. Statistics around the seismic data from the laboratory experiments answers various questions established by decision trees and the algorithm branches to a new decision based on the previous decision, thus forming a tree with branches.

Bertrand Rouet-Leduc investigates that machine learning discerns the frictional state when applied to laboratory seismic data recorded during a shear experiment in his geophysical research letter "Machine Learning Predicts Laboratory Earthquakes" *(https://doi.org/10.1002/2017GL074677)*. He further mentions that working through continuous seismic data would help in advancing through the identification of currently unknown signals, in deeper understanding of earthquake fault physics and in the accuracy of fault failure times. Paul Johnson from LANL has applied machine learning approach to study the acoustic signal data from slow displacement of formerly adjacent points on opposite sides of a fault, measured on the fault surface, in the real-world scenario from earthquake prone regions in America and New Zealand.

Experimental Data

The goal of this project activity is to use seismic signals to predict the timing of laboratory earthquakes. The data comes from a well-known experimental setup used to study earthquake physics at Los Alamos National Laboratory. The seismic input signal (acoustic_data) is used to predict the time remaining before the next laboratory earthquake (time_to_failure). The training data is a single, continuous segment of experimental data consisting of 629,145,480 rows each containing the seismic signal and the corresponding time to failure. The test data consists of a folder containing many small segments. The data within each test file is continuous, but the test files do not represent a continuous segment of the experiment; thus, the predictions cannot be assumed to follow the same regular pattern seen in the training file.

Kaggle category: Research Prediction
Kaggle competition Link: https://www.kaggle.com/c/LANL-Earthquake-Prediction

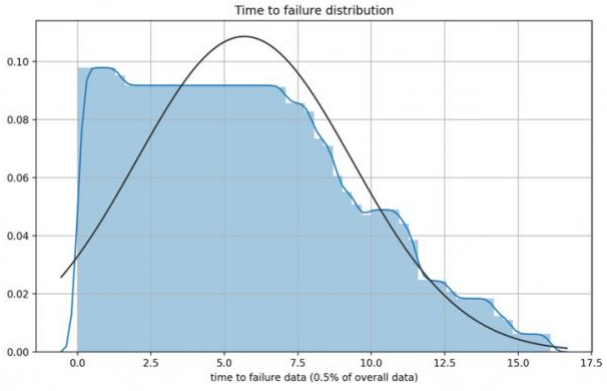
Data Exploration

The data set given has a very large dimension, more than 600 million rows of data. Descriptive summaries are obtained for both the columns, acoustic_data and time_to_failure which are presented below.

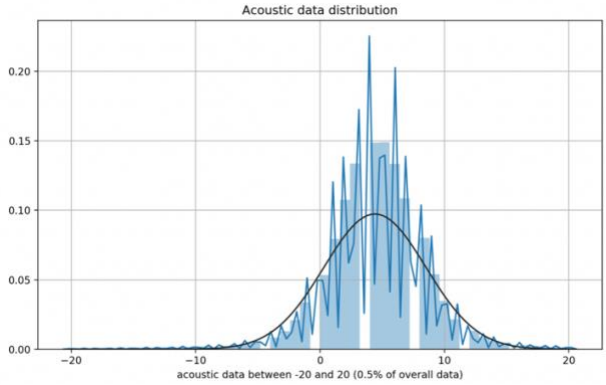
acoustic_data	
count	629,145,500
mean	4.52
std	10.74
min	-5,515
max	5,444

time_to_failure	
count	629,145,500
mean	5.68
std	3.67
min	0.00
max	16.11

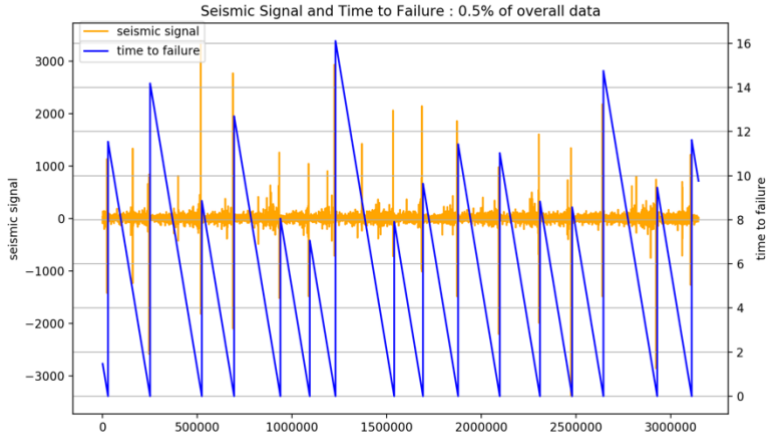
Time to failure (given in seconds) is the target variable and below is the univariate distribution obtained for it.



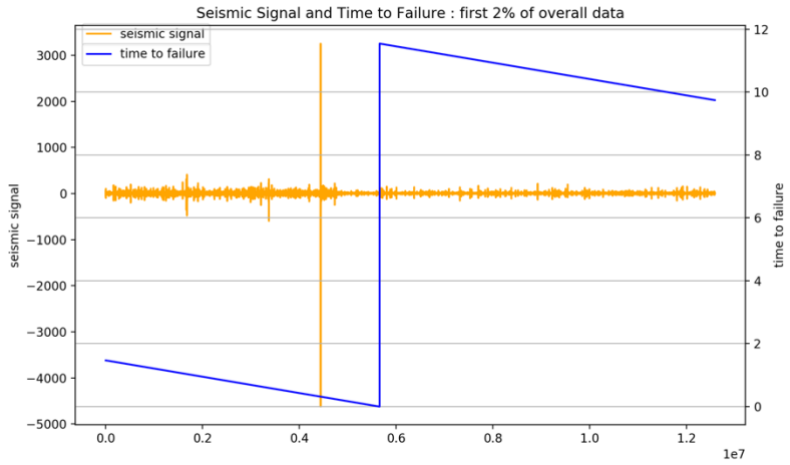
Acoustic data distribution is plotted for 0.5% of the overall data and outliers are observed in both the directions. Then, only the values which are between -20 and 20 are considered and the distribution is again plotted using distplot() function from seaborn library. This function draws a histogram and fits a kernel density estimate (KDE) for a univariate distribution. The black line is the closest normal distribution (gaussian) possible.



In the next graph, both the variables are simultaneously plotted to see their change over time. Initially 0.5% of the overall data is taken by sampling every 200 points. The acoustic data reveals complex oscillations with varying amplitude, but just before each failure there is a significant increase in the amplitude of the seismic signal. The graph also shows that amplitude also increases somewhere midway between two consecutive failure points. The time to failure decreases slowly linearly before each failure.



Additionally, after looking at the above graph, more analysis is done by closely observing first 2% of the data. Carefully looking at the graph now, it is observed that large amplitude of the seismic signal is not just before the failure time. Also, there are repetitive oscillations with comparatively higher amplitude after this large one.



When 100% of the data is plotted, it is observed that there are 16 earthquakes in the training data. Acoustic values greater than 450 are considered high and more than 80% of high acoustic values are around 0.3 seconds before an earthquake, which is an important finding.

Feature Engineering

Why generate new features?

In order to correctly predict time to failure for each test segment, one (given) attribute is not sufficient for applying any specific machine learning model on the training data. Sufficient number of features are need to be generated which are 'relevant' to the domain of the given data i.e. acoustic data, and can potentially contribute in increasing the accuracy of prediction of time to failure values. Features are structured column data which are given as input to algorithms to generate outputs. Feature Engineering is the process of extracting features from the raw dataset. This is done by filtering the data based on the cut-off time to make valid features. These features are then passed to the Machine Learning algorithm and the model is constructed [2].

Feature generation can improve prediction on many different instances, for example generated features may not individually influence the target feature, however a correlation of two features might influence the prediction of target feature significantly [1]. We have used pandas method corrwith() to calculate Pearson's standard correlation coefficient for features. Pearson's correlation is calculated by the following formula:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y}$$

It can be Positive (value of target variable increases by increasing the value of one feature) or Negative (value of the target variable decreases by increasing the value of one feature).

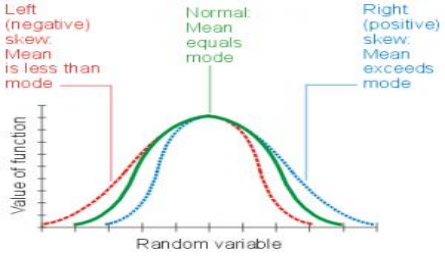
Various features have been generated from the initial data provided for acoustic signals. Some of which are mentioned below:

Features Implemented

1. **Autocorrelation** is a great tool to find repetitive patterns in acoustic data by comparing the true signal itself with a delayed version of the signal. It can effectively recognize patterns in acoustic data which is shadowed by noise or any disruption which might affect the obvious pattern in the signal from being recognized. The autocorrelation value is generated for different lag values of the acoustic signal using the formula listed below, where l is the lag value and X is the time series for which the feature is being calculated:

$$\frac{1}{(n-l)\sigma^2}\sum_{t=1}^{n-l}(X_t-\mu)(X_{t+l}-\mu)$$

2. **Skewness** measures the shift of a time series data from normally distributed plot. A standard normally distributed, non-skewed version of the plot would have equal values of mean and mode. The skew method is imported from the stats library provided by scipy.



3. The **mean absolute deviation** is calculated by finding the mod of difference between the mean value of the data and each data point in the data set.

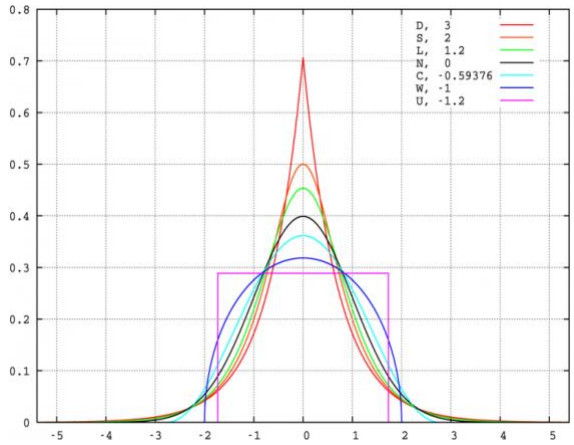
$$MAD = \frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}|$$

4. The **k-statistic** property is an estimator of the nth cumulant (here nth cumulant value is specified by the interval values in the 'feature_interval' dictionary) Kappa-n. The cumulants Kappa-n of a probability distribution are a set of quantities that provide an alternative to the **moments** of the distribution. The method of calculation of moment values differ for each data domain. For our implementation we have only considered the k-th central moment for calculation which is returned by the moment function in the stats module. The formula for calculating k-th central moment is given as follows:

$$m_k = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^k$$

here k values are in the range from 1 to 4 for our implementation.

5. **Kurtosis** is the measure of outliers present in the distribution. For acoustic data it is measured as the extremity or sharpness in values of highs from the normal distribution. Similar to skewness it focuses on the shape of the curve. The graph below represents curves of varying Kurtosis values [3].

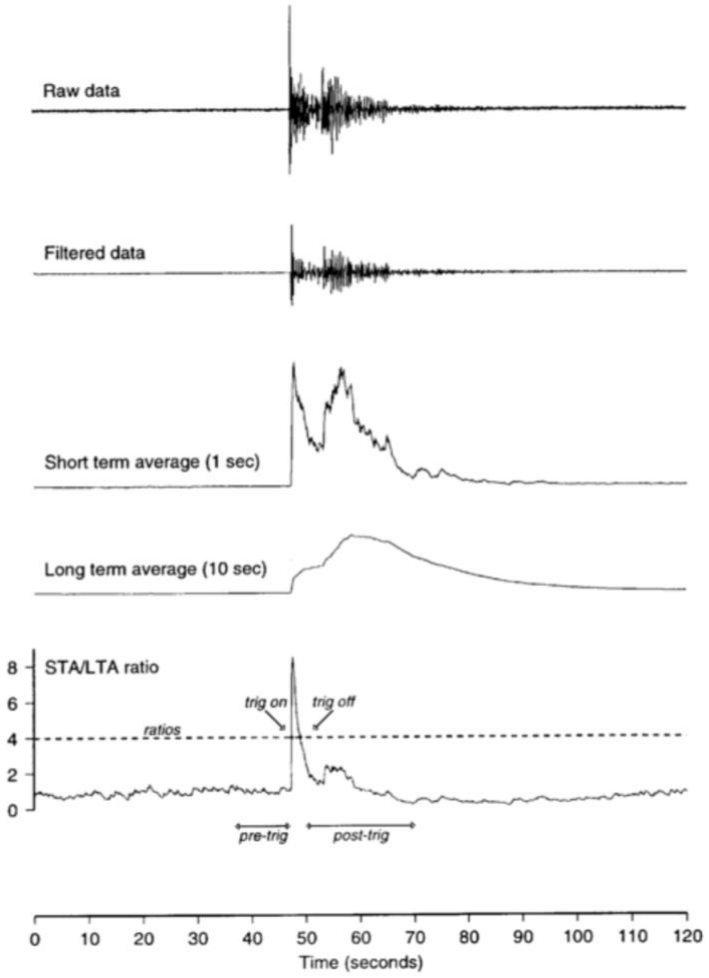


6. **Slice_less_than_threshold** feature is generated by considering the absolute value of a slice of the input and comparing it with set threshold limit. The values in the slice that are less than the set threshold limit are summed together. The features are generated on various combinations of slices and threshold limits and their values are compared to get the appropriate results.

7. **Slice_greater_than_threshold** feature is similar to the feature above. It is generated by considering the absolute value of a slice of the input and comparing it with set threshold limit. The values in the slice that are greater than the set threshold limit are summed together. The features are generated on various combinations of slices and threshold limits and their values are compared to get the appropriate results.

8. **Sta_Lta_Mean** is the Anti- Trigger Algorithm for Short Term Averaging (STA) - Long Term Averaging (LTA) Mean which is used for weak-motion seismology. Seismic signals are received continuously but not data is not stored continuously or permanently. Trigger Algorithms are required to trigger recording of data when a typical seismic signal in the constant noise signal is achieved. The STA_LTA_MEAN algorithm uses two consecutively moving window frames and

calculates average values for the absolute amplitude of seismic signal. The information about temporal amplitude of seismic noise at the site is calculated by Long-Term window and sensitivity to seismic events is provided by the Short-Term window. The data starts being recorded in a file when the ratio of the two exceeds a threshold value. Operational parameters need to be adjusted in a sophisticated manner because if the threshold is set too low, it will detect more false triggers and if the threshold is set too high, it will miss weak events [4].



9. **Valid_mean_change_rate** feature is the mean rate of change of input. It is calculated by the ratio of difference in input values to non-zero input values and finally taking the mean of valid non-infinite values.

10. **From_{input_direction}_{slice}_{typeOfAggregation}** is the feature which considers the input direction to slice input data and applies the aggregation method to it. The direction of slice can vary from the beginning or the end. The slice size is chosen to vary between 1000, 5000 and 10000. The type of aggregation to be computed are Standard_deviation, Max, Min and Mean. The mean and standard deviation are calculated by:

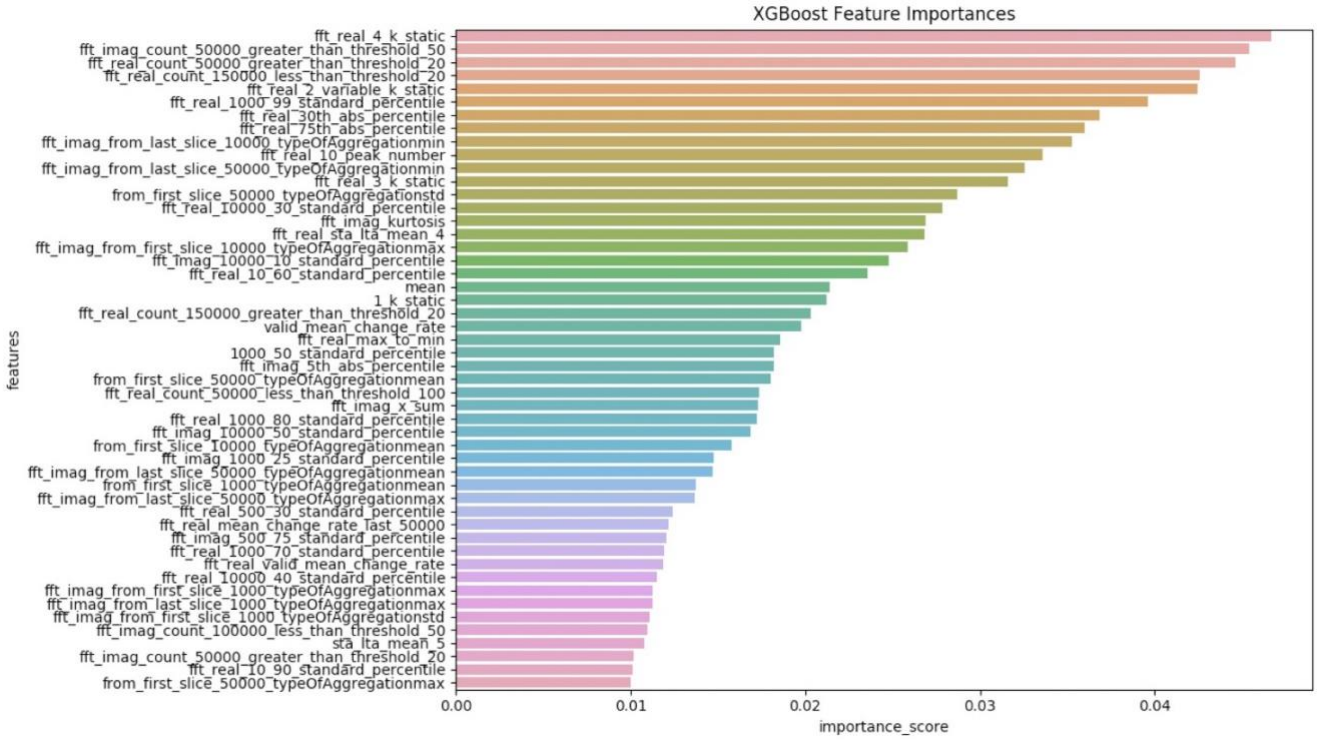
$$\text{Mean } (\bar{x}) = \frac{\sum x}{n} \quad s = \sqrt{\frac{\sum_{i=1}^{n} \left(x_i - \bar{x}\right)^2}{n-1}}$$

Modelling and Implementation

The problem is initially approached by regression modelling. The metric used is Mean Absolute Error (MAE) and thus a lower value is better with zero representing a perfect fit. The acoustic data provided is used to create statistical features which are fed into supervised learning algorithms which then seek to predict the time until an earthquake occurs from test signals.

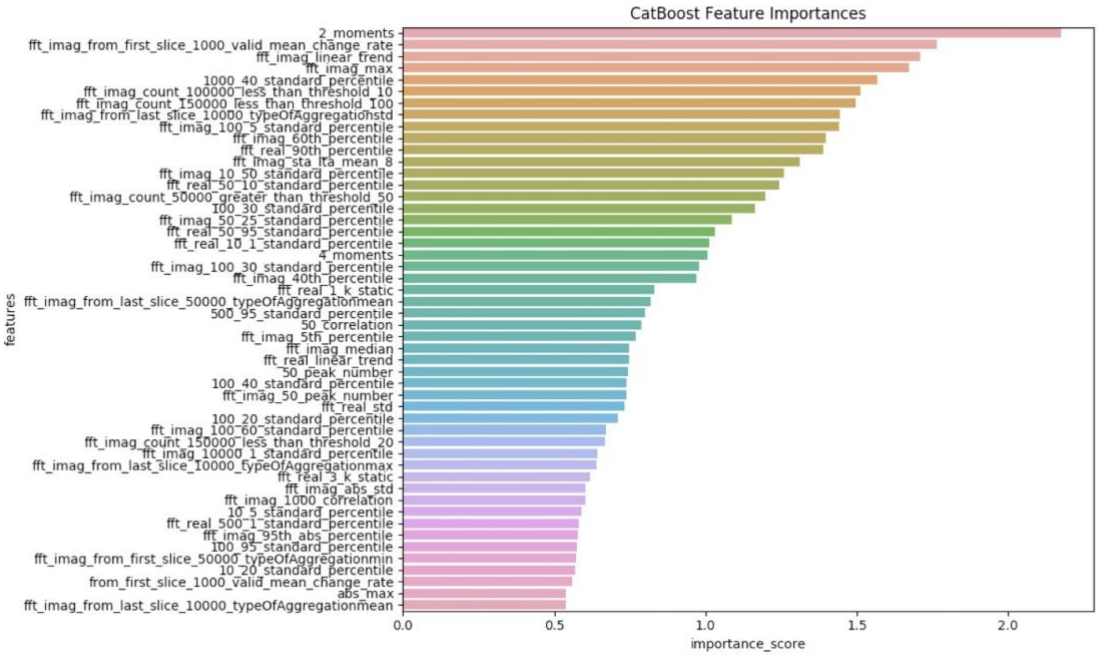
Initially the data is sliced which helped to spread the random generation of data across the signal and with computational time so that multiprocessing can be done. Stratified random sampling was performed on the data which ensured even coverage of the width of the input signal and allowed for multiprocessing so that the code is memory efficient.

XGBoost machine learning technique was then applied for the prediction. It is an implementation of gradient boosted decision trees and focuses on computational speed and model performance. Parameters of this algorithm were tuned to leverage its advantages to other algorithms. This technique allows for tree pruning and built in cross validation. Max depth was set to 9 and the metric used was Mean Absolute Error (MAE) for validation of the data. Learning rate was decreased incrementally while increasing the number of trees and was finally set to 0.01. Another gradient boosting machine algorithm, **Light GBM** was applied to the dataset to compare the results with XGBoost. Light GBM buckets the continuous feature variables into discrete bins which fastens the training process. It thus uses lower memory and produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy. However, since it can sometimes lead to overfitting, XGBoost was performed to compare and analyse its performance. Num_leaf is the main parameter to control the tree complexity in Light GBM and it was fine-tuned to 125 to avoid over-fitting. A leaf-wise tree is typically much deeper than a depth-wise tree for a fixed number of leaves. L1 regularization was set to 0.13 and L2 regularization was set to 0.36. Sub sample ratio of training instance was kept as 0.81. The plot below explains the feature importance from XGBoost.



An interesting algorithm **CatBoost** (coming from words Category and Boosting), recently open-sourced from Yandex was applied on the dataset independently since it has proven to provide powerful out-of-the-box support for descriptive data. CatBoost Regressor was used for running the model. CatBoost is effective strategy based on ordering principle called Target-Based with Prior (TBS). The observed history forms

basis for the next Target Sample. This idea of artificial time is implemented in standard offline setting using random permutations of training samples. In the algorithm, the data is shuffled randomly to calculate mean on the historical data and reshuffled multiple times. Target leakage is avoided in the gradient boosting in CatBoost algorithm since it modifies the standard gradient boosting algorithm. It gives balanced trees which are less prone to overfitting. Of all the three algorithms, this gave fastest testing results since it used the same splitting criteria to split across the entire level of the tree to make it oblivious tree[6]. The number of iterations was set to 5000. However, as the number of iterations were increased to 20,000 this model took significantly much higher time as compared to XGBoost and Light GBM. The feature importance from CatBoost is plotted below.

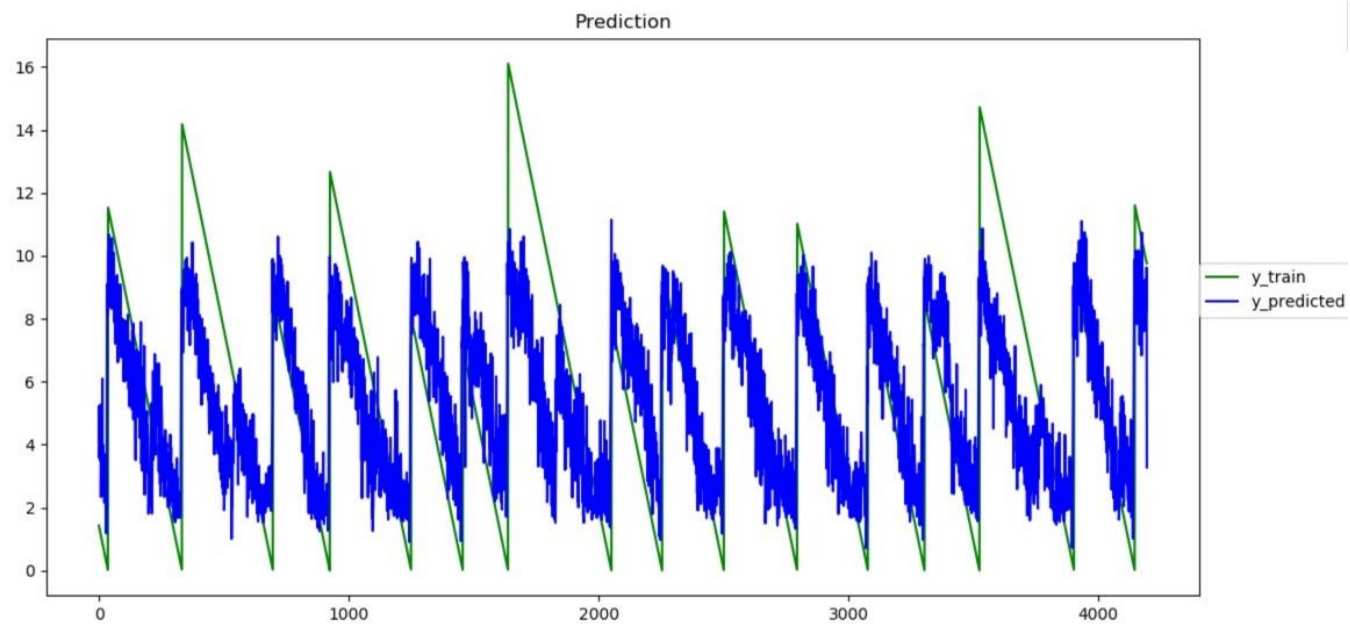


5-fold cross validation was performed on the data. It is a technique where the shuffled data is divided into smaller segments and a part of data is reserved such that it is used to test on rest of the trained data. Following steps were implemented for K-Fold Cross Validation:

1. Randomly shuffle the dataset
2. Split it into 5 groups
3. For each group
    a) Separate that group for testing
    b) Train the model on the rest of the groups
    c) Fit the model on trained data
    d) Assess the fitted model on the test group
    e) Discard the model keeping only the assessment score
4. Calculate the mean and standard deviation of all the assessment scores

Results

Both the models XGBoost and Light GBM performed well and the results were compared to the other algorithms on Kaggle leaderboard. The XGBoost model with 700+ features in this project gave MAE of 1.61. The best public leader board scores was 1.434 for LightGBM and was 1.437 for XGBoost during the active-competition time. Theoritical changes were performed in the directions which might reduce overfitting. Number of leaves was thus decreased to 125 in Light GBM model and the minimum data in the leaf was increased. Tuning parameter 'colsample_bytree' was set to 0.3 in XGBoost. Subsampling occurs once for every tree constructed. This ratio ensured that only 3/10 columns were selected at each level tree for a split, thus helping in over-fitting of the training data.



The graph above shows plot of predicted values for the train data set along with the true values of the train data. It can be observed that the points predicted by XGBoost model closely resemble the trend of the train data points and reinforces the predictivity and accuracy of the machine learning model implemented.

Considerable time was spent to fine-tune the parameters of each individual model. The results of Light GBM and XGBoost models closely mimicked each other. However, XGBoost model was finally preferred because of less memory consumption and faster results for the given data.

Conclusion

This study was completed in order to forecast an earthquake in a laboratory set up. The data set was extremely challenging because it was noisy and training data was only 13% of the potential test data. Hence, 3 different models were performed after carrying out in depth data exploration. Since the training data set had only 1 continuous feature variable, a large number of other features were statistically generated to study correlation and importance, thus increasing the accuracy of prediction of time to failure values. The study shows that XGBoost algorithm shows promise for prediction of the earthquakes on account of seismic signals generated in a timeseries manner. Also, it can be said with confidence that averaging the 3 different models would stabilize the predictions. Boosting algorithms are very useful in the reign of restricted training data and little expertise for parameter tuning. Cat boost is a modern gradient boosting machine and can be worked to a greater extent with the given data set with some more expertise.

Future Work

The results obtained so far are based on 700+ parameters. In the future, more features can be generated including exponential rolling statistics, binned entropy and statistics on different sized rolling windows. The prediction results can be compared to improve the quality of features generated and hence the overall accuracy of the model. Further, the hyperparameters for the models can be tuned to improve the accuracy.

Further, Blending and Stacking can be implemented using already trained classifiers as input to train a second layer of classifiers. The stacking generates good results when the base models are different. This can be done by training on different features or using different parts of the training sets, using different base classifiers or using different parameters.

References

[1] Nijmegen, June 2017. Suzanne van den Bosch: Automatic feature generation and selection in predictive analytics solutions

[2] Will Koehrsen, Nov 13, 2018. Feature Engineering: What Powers Machine Learning. URL:*https://towardsdatascience.com/feature-engineering-what-powers-machine-learning-93ab191bcc2d*

[3] DeCarlo, L. 1997. On the Meaning and Use of Kurtosis Psychological Methods. Vol 2. No.3, 292-307.

[4] Amadej Trnkoczy, September 1999. Understanding and parameter setting of STA/LTA trigger algorithm. URL: *http://gfzpublic.gfzpotsdam.de/pubman/item/escidoc:4097:3/component /escidoc:4098/IS_8.1_rev1.pdf*

[5] Sunil Ray, August 14th, 2017. CatBoost: A machine learning library to handle categorical (CAT) data automatically. URL: *https://www.analyticsvidhya.com/blog/2017/08/catboost-automated-categorical-data/*

[6] Daniel Chepenko,Feb 14. Introduction to gradient boosting on decision trees with Catboost. URL: *https://towardsdatascience.com/introduction-to-gradient-boosting-on-decision-trees-with-catboost-d511a9ccbd14*

[7] Sayak Paul, August 15, 2018. Hyperparameter Optimization in Machine Learning Models. URL: *https://www.datacamp.com/community/tutorials/parameter-optimization-machine-learning-models*