LANL Earthquake Prediction

1. Introduction and problem description

Forecasting earthquakes is a crucial problem in Earth science because of their devastating and large scale consequences.
The objective of this project is to when the earthquake will take place. Specifically predict the time remaining before laboratory earthquakes occur from real-time seismic data.
Impact of this project: potential to improve earthquake hazard assessments that could save lives and billions of dollars in infrastructure.
Given seismic signals we are asked to predict the time until the onset of laboratory earthquakes. https://www.kaggle.com/c/LANL-Earthquake-Prediction

2. Related work

Shaking Earth, a user in Kaggle has published a nice kernel to get the feel of the dataset. https://www.kaggle.com/allunia/shaking-earth . The visualization of the data explains that earthquake occurs (time to earthquake = 0) fraction of seconds after the peak in the seismic signal. Another take away from the kernel is, the more features you add the better the prediction, as the signal is the only input column in the dataset. The cycles depend on each other. There is temporal correlation of future signals with past ones. Another user mentions to extract the features using rolling window approach

3. Dataset description

The data comes from a well-known experimental setup used to study earthquake physics. The acoustic_data input signal is used to predict the time remaining before the next laboratory earthquake (time_to_failure).
There are 18 earthquakes in total, which is found by visualizing the data.
Training Data:

- acoustic_data - the seismic signal [int16]
- time_to_failure - the time (in milliseconds) until the next laboratory earthquake [float64]

Training Data instances: 629 million points
Data Distribution:

```
signal quaketime
count  1.000000e+07
mean   4.502072e+00
std    1.780707e+01
min    -4.621000e+03
25%    2.000000e+00
50%    4.000000e+00
75%    7.000000e+00
max    3.252000e+03
```

Test Data:

- seg_id- the test segment ids for which predictions should be made (one prediction per segment)
- acoustic_data - the seismic signal [int16] for which the prediction is made.

Test Data instances: 2624 files, with 150,000 instances for each file => 393,600,000 instances

Both the training and the testing set come from the same experiment. There is no overlap between the training and testing sets, that are contiguous in time. there is indeed a gap every 4096 samples, an artifact of the recording device

4. Pre-processing techniques

Feature Engineering:
Feature generation: Create several groups of features:

- Usual aggregations: mean, std, min and max: Basic idea of about the distribution.
- Average difference between the consecutive values in absolute and percent values: Gives an idea about how fast the time to earthquake is varying.
- Absolute min and max values: Necessary for the model to understand the status of the earthquake.

- Quantile features (1, 5. 95 & 99 percentile): By observing different kernels and finding out the importance of parameters we found that the distribution is gaussian. The values when the time to earthquake is decreasing to almost zero, and the values where time to earthquake jumps from zero to a high value are really important. Therefore taking the quantiles into consideration will greatly enhance the score. Gives fine grain details than aggregations.
- Rolling Window features: we can calculate the mean of the previous two values and use that to predict the next value.
  Standard Scaling

5. Proposed solution, and methods

Techniques used:

- Light Gradient Boosting LGBM
- Extreme Gradient Boosting

Light GBM and Extreme GBM are good models to feed in data for Time Series Data. We have also used CatBoostRegressor , but almost everytime we use it the kernel inside Kaggle dies. Therefore, we have withheld providing the information in the code.

Experimental methodology:

- Divide the training data into chunks of 150,000 data points as the test data consists of 150,000 points
- We are not creating validation dataset as the input dataset is a contiguous data from a sensor. Creating validation dataset by choosing the data randomly will not give any good results, as the earthquake cycles are dependent on each other. Dividing it into validation data will only make the learning worse.
- Scale the data
- KFolds Cross Validation: Cross Validating different folds helps to learn the data better. Shuffling the data inside the Fold increased the score.

Coding Language:

- Python : Abundance of Libraries

Computing Resources
- Kaggle: Provides high computation which makes it easier to train the model.

6. Experimental results and analysis

Experiment Number	Parameters Chosen	Results
1.	LIGHT GRADIENT BOOSTING Number of leaves: 64 Minimum data in leaf: 50 Objective: Mean Absolute Error Maximum Depth: -1 Learning rate: 0.001 Boosting technique: gbdt Fractions of features: 0.5 Bagging frequency: 2 Bagging fraction: 0.5 Number of bagging seed: 0 Result metric: Mean Absolute Error Verbosity: -1 Regularization alpha: 1.0 Regularization lambda: 1.0	Train Data Instances: 629145480 Number of attributes in Original Dataset: 1 Total features after feature engineering: 60 Validation: K-Fold with 5 folds cross validation Test Dataset: 2624 files corresponding to segments each having 150,000 data instances MAE = 2.040027143542845
2.	LIGHT GRADIENT BOOSTING Number of leaves: 128 Minimum data in leaf: 50 Objective: Mean Absolute Error Maximum Depth: -1 Learning rate: 0.001 Boosting technique: gbdt Fractions of features: 0.5 Bagging frequency: 2 Bagging fraction: 0.7 Number of bagging seed: 0	Train Data Instances: 629145480 Number of attributes in Original Dataset: 1 Total features after feature engineering: 60 Validation: K-Fold with 5 folds cross validation Test Dataset: 2624 files corresponding to segments each having 150,000 data instances MAE = 2.040986582375291

	Result metric: Mean Absolute Error Verbosity: -1 Regularization alpha: 2.0 Regularization lambda: 1.5	
3.	LIGHT GRADIENT BOOSTING Number of leaves: 32 Minimum data in leaf: 10 Objective: Mean Absolute Error Maximum Depth: -1 Learning rate: 0.001 Boosting technique: gbdt Fractions of features: 0.8 Bagging frequency: 2 Bagging fraction: 0.8 Number of bagging seed: 0 Result metric: Mean Absolute Error Verbosity: -1 Regularization alpha: 1.0 Regularization lambda: 0.5	Train Data Instances: 629145480 Number of attributes in Original Dataset: 1 Total features after feature engineering: 60 Validation: K-Fold with 5 folds cross validation Test Dataset: 2624 files corresponding to segments each having 150,000 data instances MAE = 2.040416773591041
4.	EXTREME GRADIENT BOOSTING Learning rate: 0.01 Maximum depth: 6 Subsample: 0.8 Column samples by tree: 0.8 Column samples by level: 0.8 Column samples by node: 0.8 Lambda: 0.1 Alpha: 0.1 Objective: reg:linear Result metric: Mean Absolute Error Silent: True Number of threads: 4	Train Data Instances: 629145480 Number of attributes in Original Dataset: 1 Total features after feature engineering: 60 Validation: K-Fold with 5 folds cross validation Test Dataset: 2624 files corresponding to segments each having 150,000 data instances MAE = 2.060464872800424
5.	EXTREME GRADIENT BOOSTING Learning rate: 0.01	Train Data Instances: 629145480 Number of attributes in Original Dataset: 1

	Maximum depth: 12 Subsample: 0.5 Column samples by tree: 0.5 Column samples by level: 0.5 Column samples by node: 0.5 Lambda: 0.5 Alpha: 0.5 Objective: reg:linear Result metric: Mean Absolute Error Silent: True Number of threads: 4	Total features after feature engineering: 60 Validation: K-Fold with 5 folds cross validation Test Dataset: 2624 files corresponding to segments each having 150,000 data instances MAE = 2.069876007816448
6.	A combination of all the above and the result measured as an average of those.	Train Data Instances: 629145480 Number of attributes in Original Dataset: 1 Total features after feature engineering: 60 Validation: K-Fold with 5 folds cross validation Test Dataset: 2624 files corresponding to segments each having 150,000 data instances MAE = 2.041287541602259

Best parameter set selected: from experiment number 1

- LIGHT GRADIENT BOOSTING
- Number of leaves: 64
- Minimum data in leaf: 50
- Objective: Mean Absolute Error
- Maximum Depth: -1
- Learning rate: 0.001
- Boosting technique: gbdt
- Fractions of features: 0.5
- Bagging frequency: 2
- Bagging fraction: 0.5
- Number of bagging seed: 0
- Result metric: Mean Absolute Error
- Verbosity: -1
- Regularization alpha: 1.0

- Regularization lambda: 1.0

7. Conclusion:

There is more scope in reducing the Mean absolute error. Average MAE for 900 rank in the leaderboard is around 2.045. I think taking into consideration that gap between 4096 samples in the training data set might be the key to reducing MAE even further. But in conclusion, Light gradient boosting gave us better results than Extreme gradient boosting for the time series data.

https://www.kaggle.com/maneshreyashs/rolling-quantiles

8. Contribution of team members:

1. Chandra Kiran Saladi (cxs172130): Competition selection. Finding out relevant libraries and techniques to use. Feature Engineering for Testing data. Idea about Cross Validation, Report
2. Shreyash Mane (ssm170730): Code for lgb and finding best parameters.Insights about Shaking Earth kernel. Optimizations in the code. Report
3. Tanya Tukade (txt171230): Code for XGB. Optimizations in the code. Report.
4. Supraja Ponnur (sxp179130): Feature Engineering for training data. Report. Idea about dividing the dataset into different parts as the test data

9. References

1. https://www.kaggle.com/allunia/shaking-earth
2. https://www.kaggle.com/mjbahmani/probability-of-earthquake-eda-fe-5-models
3. https://github.com/tqdm/tqdm
4. https://machinelearningmastery.com/basic-feature-engineering-time-series-data-python/