EE258: Neural Networks

PROJECT #2

Under the guidance of

Prof. Birsen Sirkeci

Fall – 2020

IGNV Volcanic Eruption Prediction

By

Pradeepa Rathi Thiagarasu (015015104)

Pooja Narasimmarao Padmanaban (014637805)

ABSTRACT

The objective of the project is to determine when the next volcanic eruption will occur by evaluating a broad geophysical data collection of sensors installed on active volcanoes. Our learning algorithm will be used to classify the seismic waveform signatures that describe the evolution of the eruption. Currently scientists collect the sensor readings, manually analyze and apply some sort of mathematical formulation to predict the approximate eruption time. This is a regression problem, here we need to analyze a sequence of 10 minutes of observations of 10 sensors deployed around an active volcano. Initially we developed an end to end pipeline model with MLP and produced the results. The baseline model suffered from overfitting which was reduced by doing feature extraction using Short Term Fourier Transform (STFT), tuning the hyperparameters and ensemble the best fit models. Mean Absolute Error (MAE) is the error metrics we used to evaluate the model performance.

DATASET DESCRIPTION

The dataset is downloaded from Kaggle [1]. It has both training and test dataset split into two files train.csv and submission.csv respectively. The csv files have two columns segment_id and time_to_erupt, corresponding to each segment_id there is a csv file in train folder which has the sensor readings. The training dataset has 4431 volcanic instances and each instance has 60001 observations for 10 different sensors. Basically, we need to analyze 60001 datapoints to make one prediction. The same goes for test data as well.

Dataset	Datapoints
Training	4431 x60001x10
Testing	4520 x60001x10

Train.csv is the metadata for the train files.

segment_id: Primary key (ID) for the data segment. Matches the name of the relevant data file.

time_to_eruption: The target value, the time until the next eruption.

Fig 1 is the visualization of sensor readings of the segment_id 1000015382. Each color in the graph represents one sensor and their readings for 10 mins. The graph clearly shows that

there are outliers in each of the readings and in this case, we are interested on the outliers as they represent the intense seismic waves compared to the normal readings.

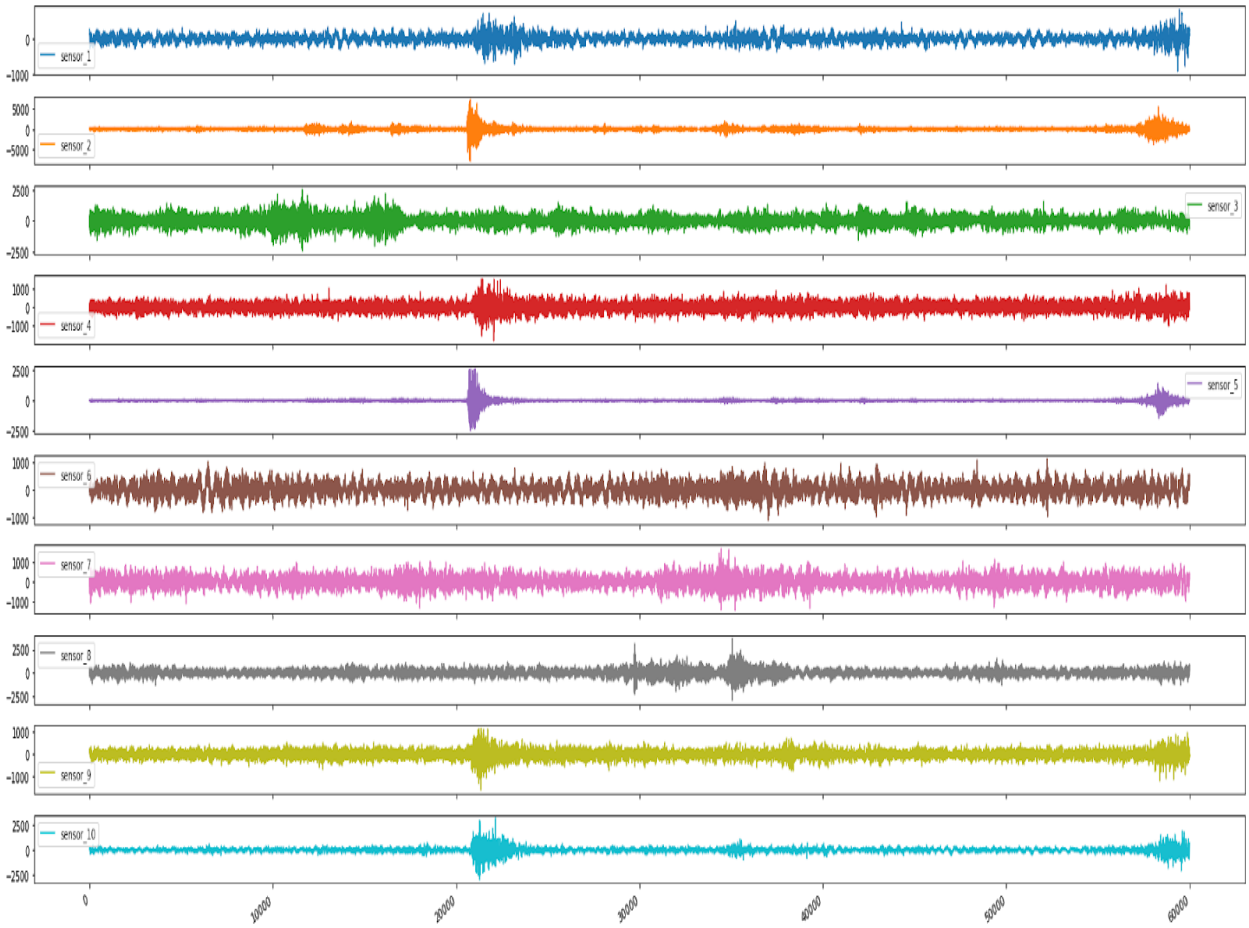


Fig 1. Visualization of sensor readings from a sample training file(1000015382.csv

The time domain and frequency domain wave form representations are shown in Fig 2 and Fig 3 for sensor 4 readings. The graphs are helpful to interpret the behavior of the waveforms. As seen from the graphs, sensor 4 readings did not exhibit a uniform pattern. Sensor 4 readings from different segment_ids have different pattern. This in a way helps to confirm that these sensors are deployed at different volcanic instances and training data has readings for multiple volcanic instances. For frequency domain representation, the data is first transformed from time to frequency domain with the following specifications.

Sampling frequency , fs = 100Hz

Data length, N = length of the segment_id representing the datapoints

FFT segment size , n = 256

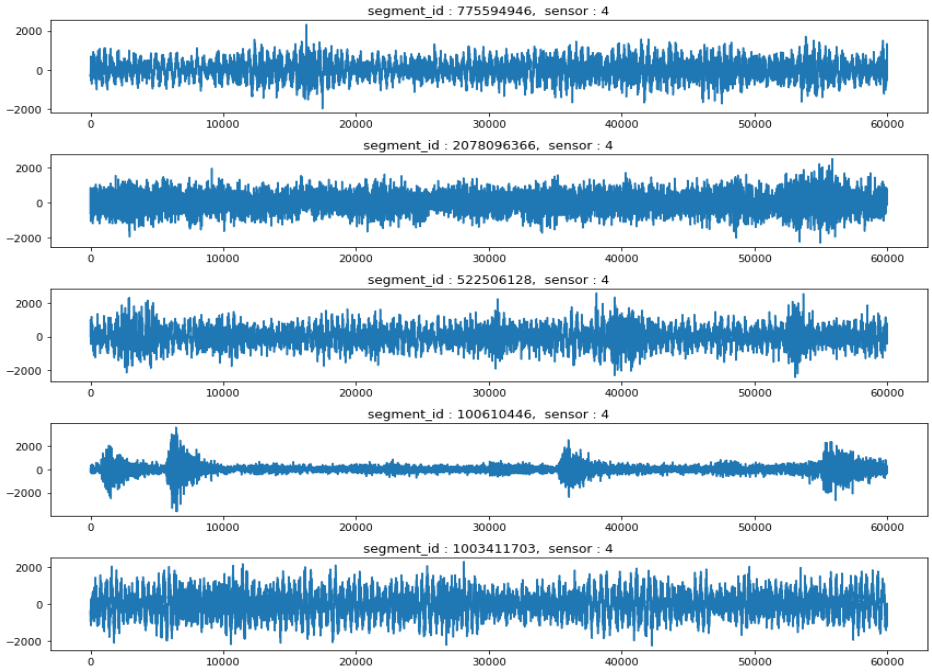


Fig 2. Time domain visualization of Sensor 4 readings for different volcanic instances

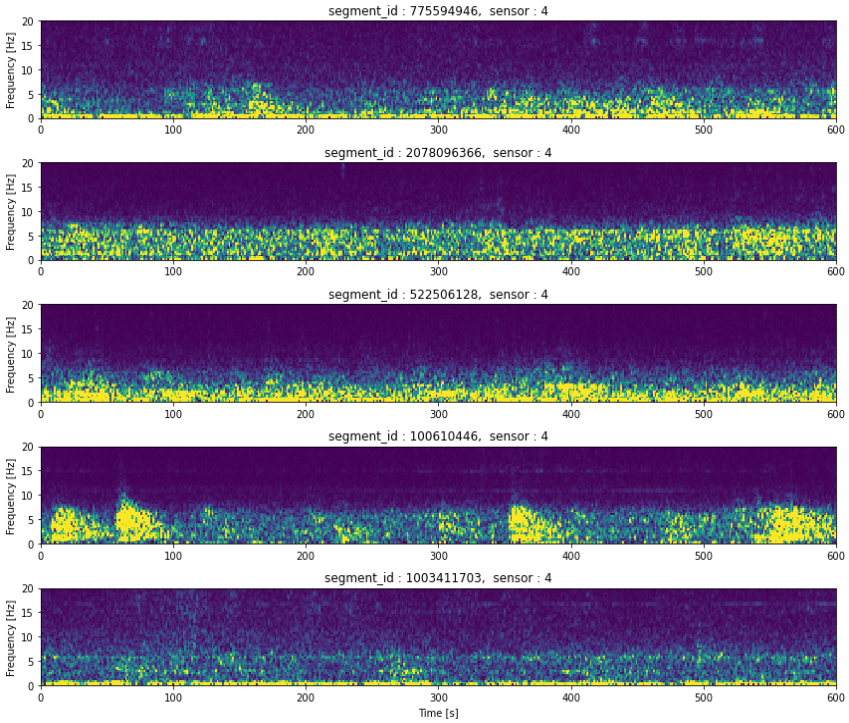


Fig 3. Frequency domain visualization of Sensor 4 readings for different volcanic instances (sampling frequency, fs=100)

Fig 4 gives the histogram of each sensor for a sample segment id. The mean is calculated for the sensor readings and then the graph is plotted. The time_to_eruption graph shows that there is a considerable variation in the eruption time for the recorded sensor readings.

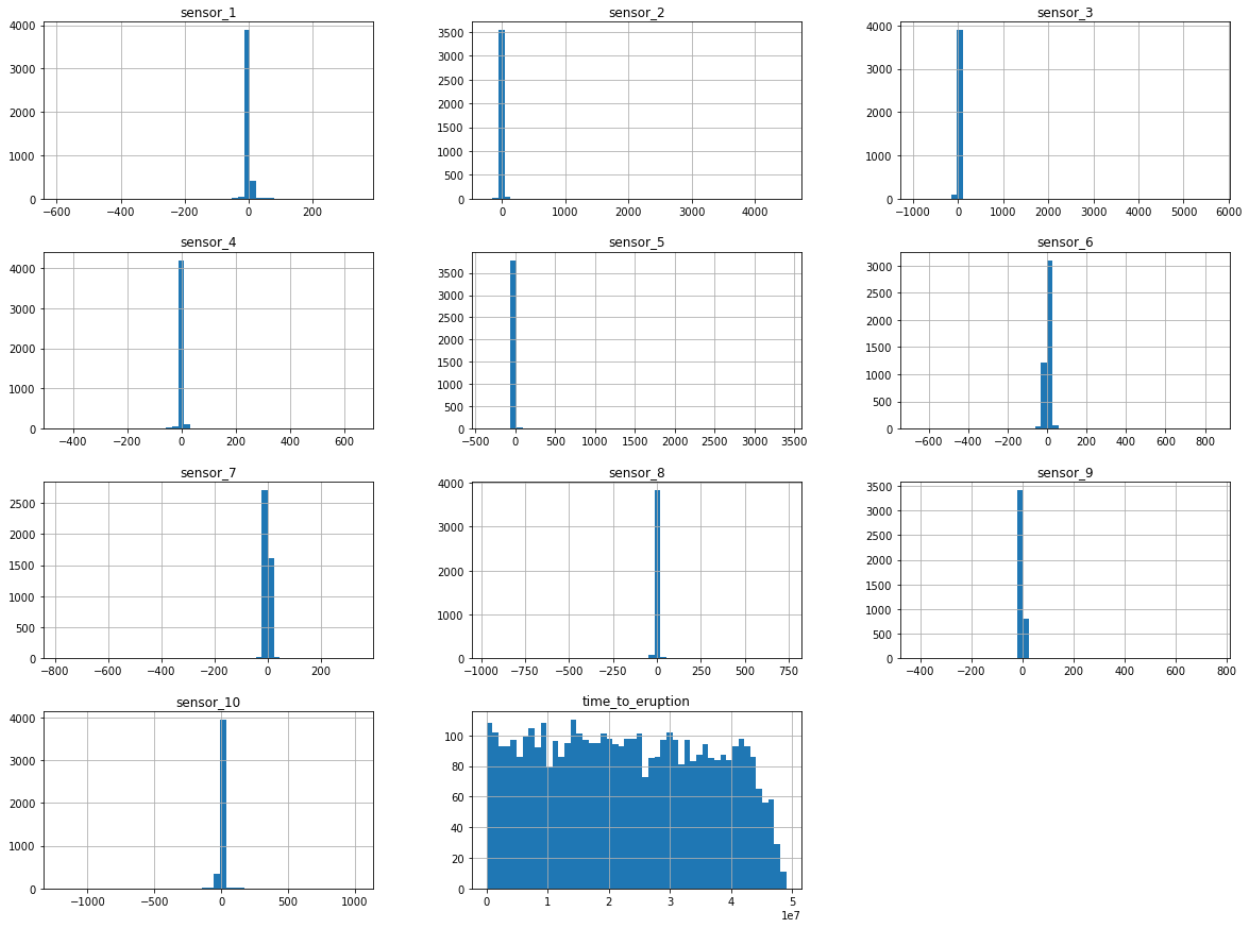


Fig 4. Histograms of mean of all the training data and time to eruption

PREPROCESSING

a) DATACLEANING

The segment_id files had many Nan values which were replaced by 0 and some of the sensors have 0 readings. While handling the 0 readings of the sensors one significant point to note is that whether the sensor has consecutive 0 values or intermediate 0's. then it could be because of two reasons.

1. Sensor is not working

2. Recent eruption is possible, that would have caused the sensor damage. (This is identified by analyzing the previous sensor value before going to '0')

The '0' values of this kind were dropped, however sensors with intermediate zeros were handled with mean evaluated for the rest of the sensor values.

b) BASIC STATISTICS ON DATASETS

The original data consists of nearly 60001 observations for a single time_to_errupt prediction. It was difficult to handle such a huge datapoints which in total accounts to 4431*60001 = 265864431 datapoints with 10 features. So, we decided to perform basic statistics on this data like mean, sum, standard deviation, variance, maximum, minimum, median, skew, mad etc., to reduce the data dimensions. This preprocessing helped to train the models easily and see the initial results for the baseline model. The training data dimensions were reduced from 4431 * 60001 *10 to 4431 *1 *441.

Quantiles are used to segment the spectrum of the probability distribution into constant intervals with equal probabilities, or to divide the results into samples of the same manner. Quantiles and percentiles were also calculated. All the preprocessing techniques were executed on the both training and test datasets.

c) FEATURE EXTRACTION

The raw sensor data were distributed in wide range and hence feature extraction had to be done before feeding the data in neural network. We have also analyzed the model's performances in raw data, since there were not up to the mark, we decided to do feature extraction.

1) Fast Fourier Transform (FFT)

Before doing FFT, the above-mentioned basic statics were performed on the raw data and the dimensions are reduced. After that using the numpy.fft.fft (one-dimensional discrete Fourier Transform) transformations were done. This function calculates a single-dimensional n-point discreet Fourier Transform (DFT) with an efficient Fast Fourier Transform (FFT) [CT] algorithm where only the sensor readings were given as the input to the function. After performing the fft the basic statistics were again executed on the data. There were 441 features and one timestep observation for one time_to_errupt prediction.

2) Short Time Fourier Transform (STFT)

The seismic sensor readings were fluctuating even though the dataset given by kaggle were already normalized. This is also visible in the visualization graphs and one of the reasons to adapt STFT to extract the relevant features by doing time-frequency domain transformation. The scipy.signal.stft function is the module in scipy that was used to perform the feature extraction.

STFT Specifications:

Sampling Frequency, fs = 100Hz

Data length, N = length of the observations in the segment_id.csv

FFT segment size, n = 256

Maximum frequency = 20 Hz

Delta frequency = fs/n = 100/256 = 0.39 Hz

Delta time = n / fs / 2 = 256 /2 / 100 = 1.28s

The total no. of features extracted after stft execution is 100 and it gave the best performance compared to fft feature extraction on basic statistics.

d) NORMALISATION

Data normalization was done using the min max scaler to reduce the range of sensor readings. Standard scaler didn't perform well on this data. CNN and RNN models were not able to calculate the MAE after the standard scaling. First, X_train data was normalized using fit_transform function and then X_test was just transformed based on the fit values obtained using X_train transformation.

BASELINE MODEL

The baseline model was created by training the MLP algorithm using the preprocessed data where basic statics like mean, sum, quantile and linear regression and fft were done. We decided to use MLP as the baseline model as its simple, easy to implement and less complicated compared to CNN and RNN.

Network Architecture : 441-4431-256-256-256-1

Activation functions : ReLU for the hidden layers and Linear for the output layer

Optimizer : Adam with learning rate 0.001

Metric : Mean Absolute Error(MAE)

Cross-Validation : Validation set approach(0.05 which is nearly 221 datapoints)

Table 1 Initial results of the Baseline Model

Baseline Model	Training MAE	Validation MAE	Kaggle Score
MLP	435254	4960415	9341609

Table 1 shows the initial results of the baseline model. Only the training and validation MAE were compared because the y_target value for the test data were not given. We can only evaluate the test performance by submitting the test predictions in kaggle. Our target MAE value was 4,000,000 validation MAE which was done considering the bestscores of published notebooks in kaggle. Fig 5 represents the scatter of y_predicted and y_target and the best fit line. The model suffered from overfitting and the actions taken to resolve it were explained in model improvements.

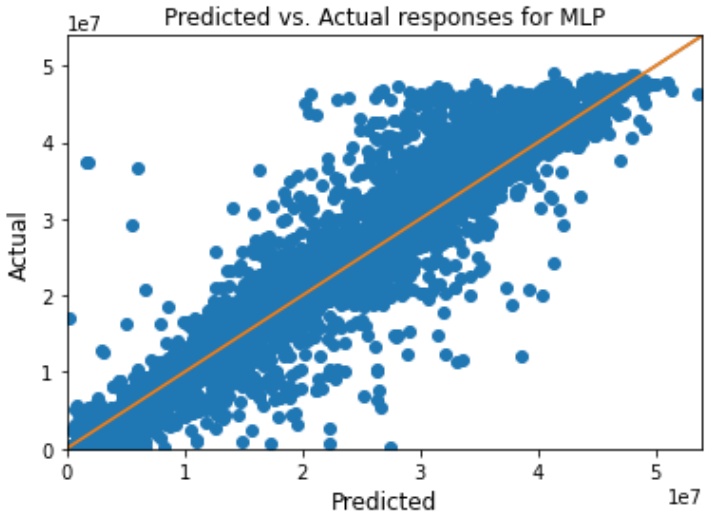


Fig 5. Linear Regression best fit line for predictions training data of MLP Baseline model trained on Dataset 1(*explained in model improvements)

MODEL IMPROVEMENTS

The model improvements were carried out on three different training datasets that were obtained as a result of different feature extraction techniques.

Dataset 1 - by performing the FFT on the raw sensor data with 441 features.

Dataset 2 - by performing the STFT on the raw sensor data with 100 features.

Dataset 3 - Entire raw dataset with 60001 rows and 10 columns for each instance.

DATASET 1

This dataset was obtained by performing the basic statistics and FFT feature extraction. Below table gives the dimensions of the Dataset 1 after the preprocessing and feature extraction techniques. Multi-layer perceptron, CNN and RNN were trained on Dataset 1 and the performances were analyzed.

Dataset	Dimension
Training	4431 x441
Testing	4520 x441

MULTI-LAYER PERCEPTRON (MLP)

MLP models were our first choice for building the baseline model and understanding the dataset behaviors. Table 2 represents the Model 1 and Model 2, their architecture, training and validation MAE along with the Kaggle score. We were conscious that the initial hidden layer should consider all the features of the data points, so Model 1(Baseline) was designed with a large no. of neurons in the first hidden layer. However, the model overfitted. So, in Model 2, we reduced the number of neurons in the first hidden layer to 100 but this time not only there was a large generalization gap, but the training error was also high. We used an "Adam" optimizer with fixed learning rate 0.01 for both the models. When SGD was used as the optimizer it was not able to calculate the MAE and gave NaN values. So, we stick to Adam optimizer for the rest of our models. The activation function of the output layer was "Linear" and the hidden layer was "ReLU". We also tried using "ReLU" in the output layer, but the performance decreased.

Table 2 Performance Analysis of MLP Models trained on Dataset 1

Model Name	Model Architecture	Hyperparams	Training MAE	Validation MAE	Kaggle Score
Model 1	441-4431-256-256-256-1	Optimizer = Adam (0.001) Batch Size = 10 Hidden Activation Function = ReLU Output Activation Function = Linear	4018313	5188396	9341609
Model 2	441-100-256-256-100-1	Optimizer = Adam (0.001) Batch Size = 1 Hidden Activation Function = ReLU Output Activation Function = Linear	4579050	5350268	9710810

Fig 6 gives the scatter plot for y_predicted and y_target of the training data along with the best fit line. It shows that the model couldn't predict the y_target correctly even for the dataset on which it was trained on.

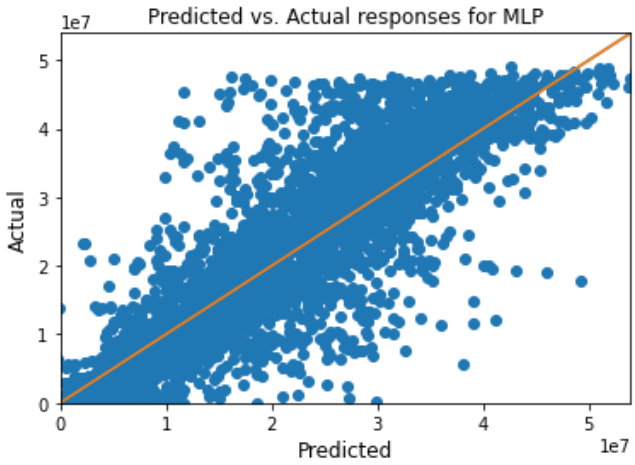


Fig 6. Linear Regression best fit line for Training Predictions of MLP Model 2 trained on Dataset 1

CONVOLUTIONAL NEURAL NETWORK (CNN)

After analyzing the test performance of the MLP models we decided to try CNN as it is good in extracting the features compared to other algorithms. Two 1D convolutional layers along with 1D Max pooling layers were used. The convolution layer here was one dimensional due to the regression dataset dimensions. The training dataset was reshaped from 4431x441 to 4431x441x1 where the one represents the channel. But the performance was not as expected and in fact it went way below the MLP baseline model. So, we didn't try more CNN models on this dataset. Fig 7 represents the CNN performance using the best fit line for training y_target.

Table 3 Performance Analysis of CNN trained on Dataset 1

Model Name	Model Architecture	Hyperparams	Training MAE	Validation MAE	Kaggle Score
CNN	**Conv→ BN→ Max-Pooling→ Conv→ BN→ Max-Pooling→ Flatten→ Dense Output**	Optimizer = Adam(0.001) Batch Size = 1 Hidden Activation Function = ReLU Output Activation Function = Linear	**5545721**	**8267549**	**11086804**

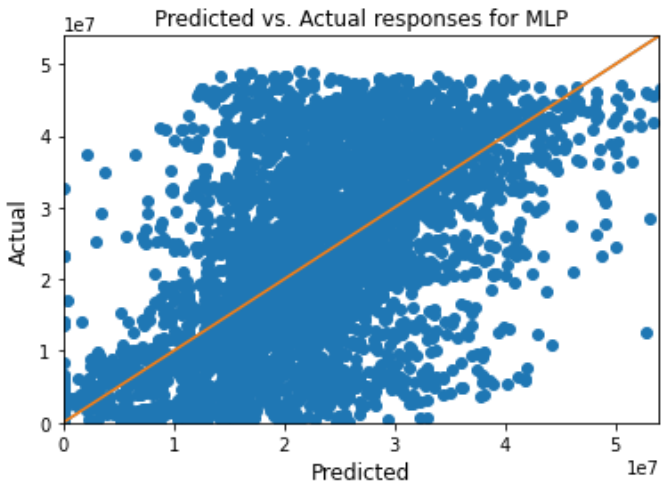


Fig 7. Linear Regression best fit line for training predictions of CNN model trained on Dataset 1

RECURRENT NEURAL NETWORK(RNN)

This is a time series data of sensor logs which made us adopt RNN for training. Long Short-Term Memory (LSTM) is a special kind of RNN which resolves the vanishing gradient problem of backpropagation through time training algorithm by creating a self-loop within the hidden layer. The self-loop in the hidden layer creates a path where the gradients flow for a long duration. This is a Multivariate Unit step RNN which means there are more than one variable change for each time instance (representing the features) and one observation to be considered for making the prediction. The training and test dataset were normalized and reshaped before giving to RNN. The training dataset was reshaped from 4431x441 to 4431x1x441 where 4431 is the batchsize, 1 is the timestep and 441 is the no. of features. Like CNN even RNN didn't give good performance for Dataset 1. Fig 8 represents the RNN performance using the best fit line for training y_target.

Table 4 Performance Analysis of RNN trained on Dataset 1

Model Name	Model Architecture	Hyperparams	Training MAE	Validation MAE	Kaggle Score
LSTM	**Hidden: 400** **Output Layers: 400-400-1**	Optimizer = Adam (0.001) Batch Size = 1 Hidden Activation Function = ReLU Output Activation Function = Linear	**5664951**	**8379087**	**11181432**

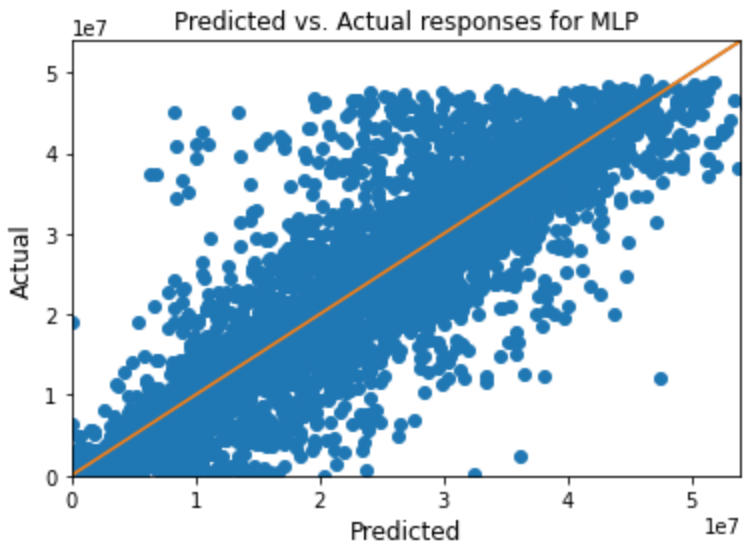
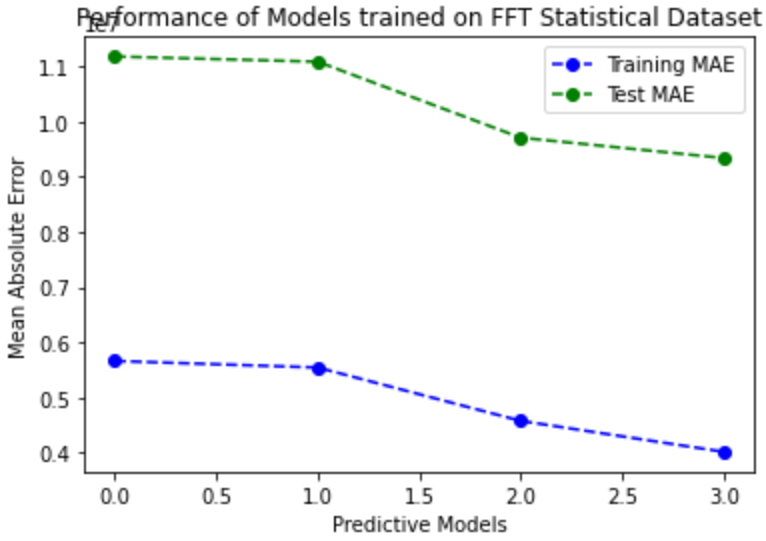


Fig 8. Linear Regression best fit line for Training Predictions of RNN model trained on Dataset 1

Fig 9 represents the performance of all the models trained on Dataset 1. As the graph shows, the training and test MAE for both RNN and CNN were high compared to the MLP models. This gave us an idea about the MLP performance on this sensor data.



Predictive Model 0: RNN

Predictive Model 1: CNN

Predictive Model 2: 100-256-256-100-1

Predictive Model 3: Baseline Model (4431-256-256-256-1)

Fig 9. Performance Graph of Models trained on Dataset 1

DATASET 2

This dataset was obtained by performing the basic statistics and STFT feature extraction. Below table gives the dimensions of the Dataset 2 after the preprocessing and feature extraction techniques. Multi-layer perceptron, CNN and RNN were trained on Dataset 2 and the performances were analyzed.

Dataset	Dimension
Training	4431 x100
Testing	4520 x100

MULTI-LAYER PERCEPTRON

MLP with different architecture was trained on this dataset. Ensemble model of the MLP models gave the best Kaggle score. Below table gives the performance of the models trained. Fig 10 , 11 and 12 represents the scatter plot for training predictions of each model.

Table 5 Performance Analysis of the models trained on Dataset 2(STFT)

Model Name	Model Architecture	Hyperparams	Training MAE	Validation MAE	Kaggle Score
Model 1	100-100-256-256-100-1	Optimizer = Adam (0.001) with l1_l2 regularize Batch Size = 1 Hidden Activation Function = ReLU Output Activation Function = Linear	1860381	2701694	6926457
Model 2	100-100-256-256-100-1	Optimizer = Adam (0.001) without l1_l2 regularize Batch Size = 1 Hidden Activation Function = ReLU Output Activation Function = Linear	1614293	2547394	-
Model 3	100-100-256-100-1	Optimizer = Adam (0.001) Batch Size = 1 Hidden Activation Function = ReLU Output Activation Function = Linear	1495936	3049761	-
Model 4	100-100-256-1	Optimizer = Adam (0.001) Batch Size = 1 Hidden Activation Function = ReLU Output Activation Function = Linear	1678359	2906106	-
Model 5	100-100-256-1	Optimizer = Adam (0.001) with ALR Batch Size = 10 Hidden Activation Function = ReLU	1887303	3322255	-

		Output Activation Function = Linear			
Model 6	100-400-256-100-1	Optimizer = Adam (0.001) Batch Size = 1 Hidden Activation Function = ReLU Output Activation Function = Linear	1731122	3277207	-
Ensemble	Model1 Model2 Model3	Averaging the predictions of Model 1, Model 2, Model 3	-	-	6638920
Ensemble	Model1 Model2 Model3 Model4 Model5 Model6	Averaging the predictions of Model 1, Model 2, Model 3, Model 4, Model 5, Model 6	-	-	5766010

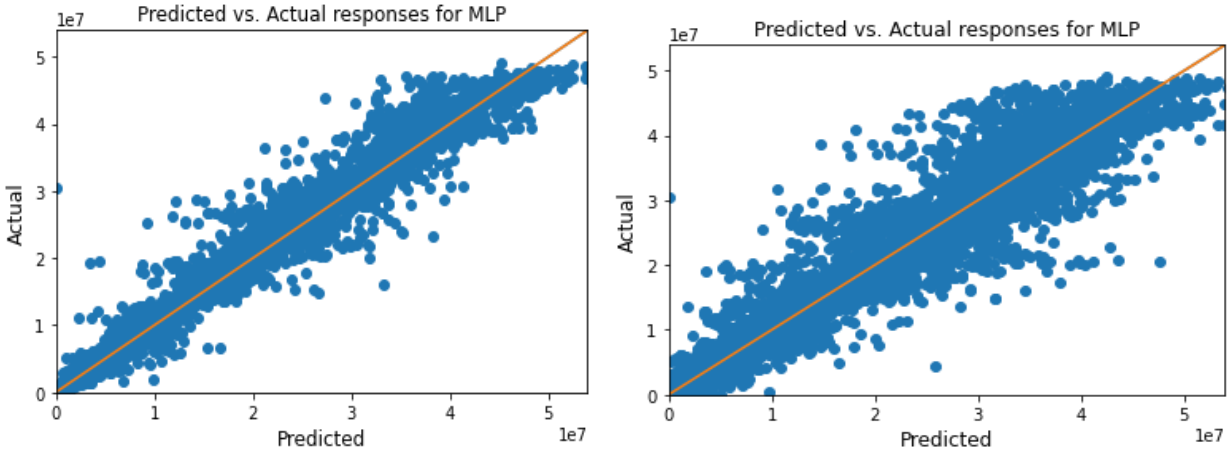


Fig 10. Linear Regression best fit line of Model 1(left) and Model2(right) on training prediction

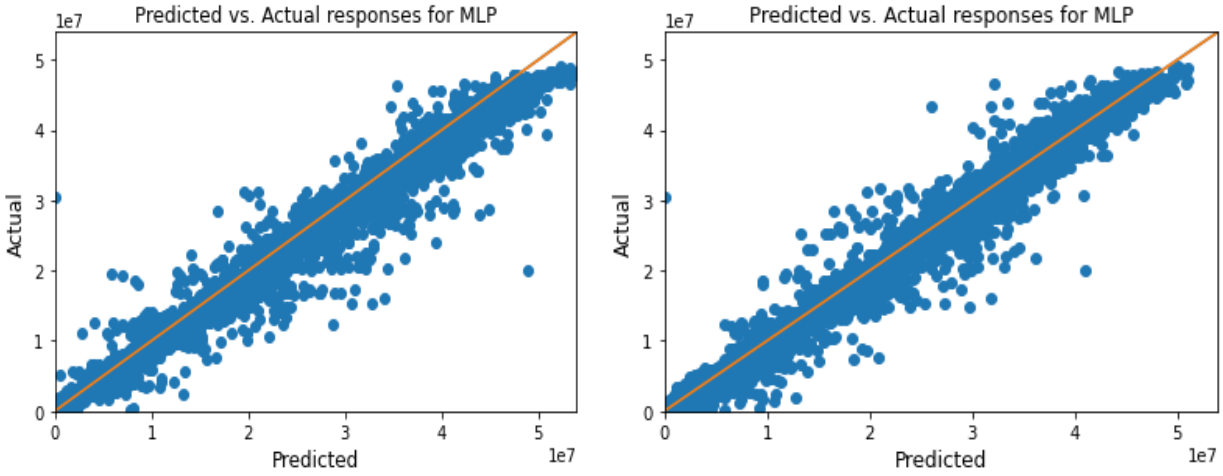


Fig 11. Linear Regression best fit line of Model 3(left) and Model4(right) on training prediction

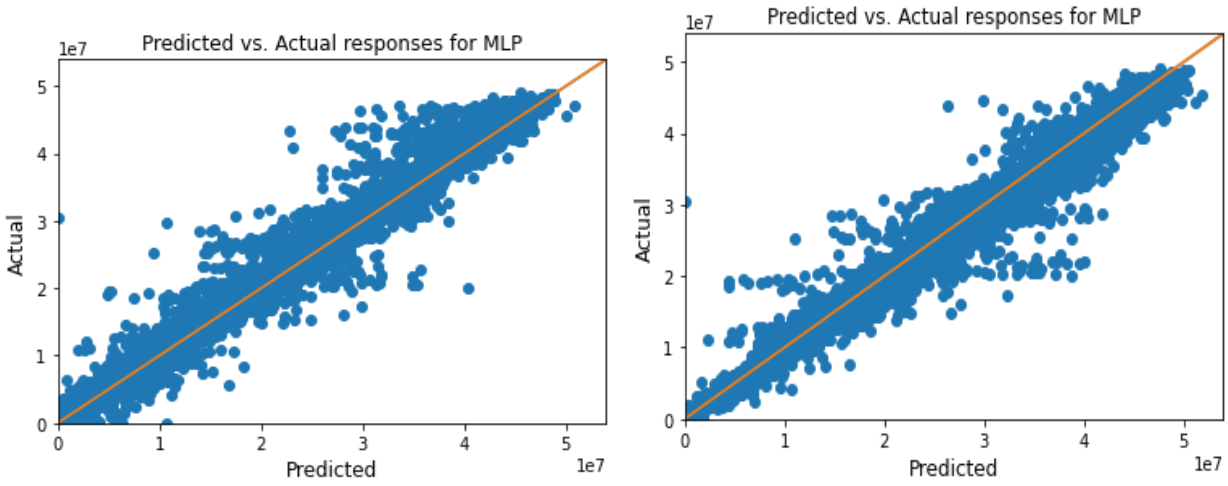


Fig 12. Linear Regression best fit line of Model 5(left) and Model6(right) on training prediction

CONVOLUTIONAL NEURAL NETWORK (CNN)

Table 6 Performance Analysis of CNN trained on Dataset 2

Model Name	Model Architecture	Hyperparams	Training MAE	Validation MAE	Kaggle Score
CNN	**Conv→ BN→ Max-Pooling→ Conv→ BN→ Max-Pooling→ Flatten→ Dense Output**	Optimizer = Adam(0.001) Batch Size = 1 Hidden Activation Function = ReLU Output Activation Function = Linear	**10056800**	**13301182**	-

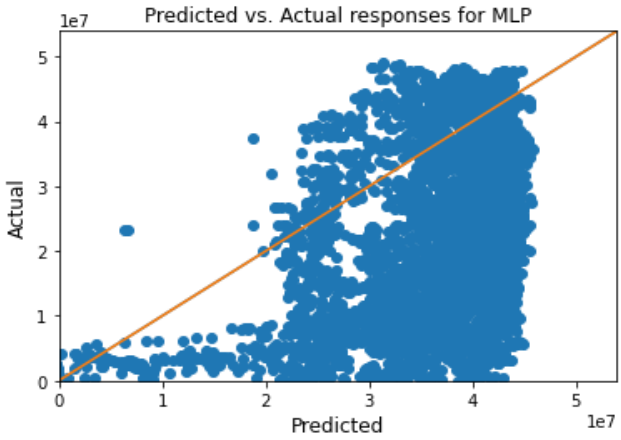
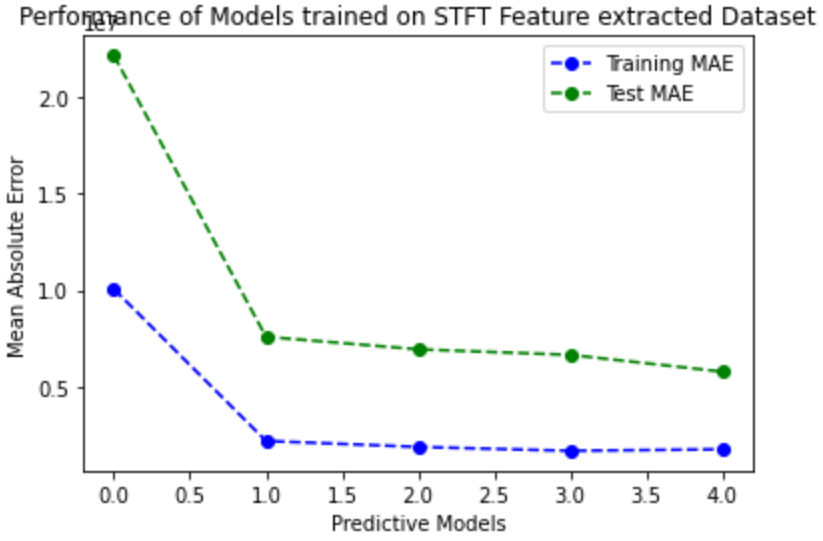


Fig 13. Linear Regression best fit line for Training Predictions of CNN model trained on Dataset 2

RECURRENT NEURAL NETWORK(RNN)

Table 7 Performance Analysis of RNN trained on Dataset 1

Model Name	Model Architecture	Hyperparams	Training MAE	Validation MAE	Kaggle Score
LSTM	**Hidden Output Layers : 400-400-400-1**	Optimizer = Adam(0.001) Batch Size = 1 Hidden Activation Function = ReLU Output Activation Function = Linear	**2170282**	**3024143**	**7577085**



Predictive Model 0: CNN

Predictive Model 1: RNN

Predictive Model 2: 100-256-256-100-1

Predictive Model 3: Ensemble of 3 models

Predictive Model 4: Ensemble of 6 models

Fig 14. Performance Graph of Models trained on Dataset 2(STFT)

Fig 14 gives the overall performance of all the models trained on Dataset 2.

DATASET 3

Entire dataset was used in Dataset 3 with size 29 GB. In order to load the dataset customized data generator and iterator are used.

MLP, CNN and RNN were trained on this dataset.we faced issues while setting the iterator to accept the data. Data cleaning and normalization functions implemented as soon as loading each file.Matching the y_train with each file and combining them into batches.

The data generator yields an array in the form of :- (batch size, rows in each file, column in each file)

The iterator was not providing the right number of files in batches initially. The fit function was not calling the iterator variable properly as the batch size was not matching the size of files the iterator was providing.Due to memory constraints, we were only able to run 5 epochs during training. The performance was not as expected.

Kaggle score - 31539777

Used Google Cloud TPU service to load the entire dataset. Then we linked the Cloud environment with the localhost Jupyter. We chose the plan with 52GB memory to load the VM image on the Google Cloud. But since the dataset was really large, after loading the data, we were not able to train immediately, and we needed to keep erasing values in the variables that were of no use anymore and were present in the previous cells. We saved the trained model in file and loaded that later for predicting test labels. We were able to do this only after deleting all the variables that were used for training like the X_train, cnn_model, etc. The performance was also medium.

DISCUSSIONS & CONCLUSION

The predictive model for volcanic eruption was built after performing feature extraction on the dataset using Fourier transforms. The Ensemble of MLP models trained on the Dataset 2(STFT feature extracted data) gave the best performance with the below mentioned kaggle score and ranking.

KAGGLE FINAL SCORE : 5766010

KAGGLE FINAL RANKING: 158

Fig 15 shows the performance comparison of the baseline model and one of the best models considered for assembling. There was a significant improvement in the performance of the model after using STFT feature extraction and ensemble technique.

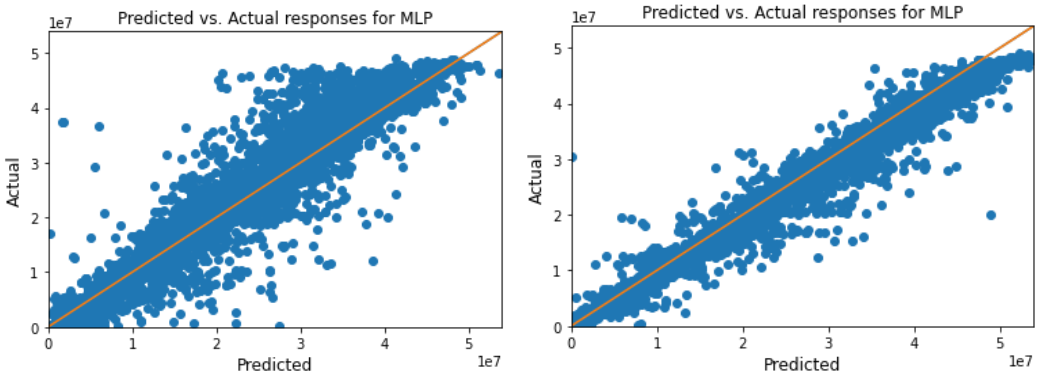


Fig 15. Performance Comparison of Baseline model and Model 3

Below are some of the observations inferred during the project.

The SGD optimizer SGD couldn't give a better performance. It couldn't calculate the mean absolute error and gave NaN values. Hence Adam optimizer was adopted for all the models tried in the project. Adam can be used for complex regression problems.

Feature Extraction of data helped a lot in model performance. In order to produce the initial results for the baseline model we used the dataset where the basic statistics were performed. Our initial assumption was to improve the model performance by training them on raw data (without feature extraction) as it had lots of training data. However, we couldn't handle such a huge amount of data and even RNN or CNN wouldn't have given performance like MLP did. So, for time series data especially when there are fluctuations in the data points like in the case of a speech signal it's always better to do feature extraction using suitable Fourier transforms. In this case Short Time Fourier Transform extracted the most relevant features.

In order to train the RNN and CNN model using all the 60001 observations for one prediction we took a Google cloud instance. But we didn't choose the appropriate memory instance suitable for handling the dataset size. The model gave OOM (Out of Memory) error for certain models and could not make predictions. It is always better to calculate a rough estimation of the memory allocation, no of learnable parameters to optimize the hardware and memory usage.

Data Normalization enabled CNN and RNN to perform better but MLP's performance decreased when trained with the normalized data using Min Max scaler.

Initially we thought RNN is the suitable model for this problem as it's a time series dataset and we need to observe 60001 sequential observations to make one prediction. But MLP performed better compared to the complex models. So, for fixed length time series problems MLP models can also give best results. RNN can be chosen in case if we have variable length inputs.

Ensemble regularization techniques of the MLP models gave the best results. The models are not giving constant results with every training. Even after training on the same dataset at each time we end up with different test and train errors. Ensemble of models is used to consolidate the predictions from these models.

Future Suggestions

We couldn't use the customized data generator in TPU due to time constraints. We will try to run this iterator and generator to process the training dataset.

We wanted to split the data in batches and train the model with data chunks.

Issues

Out of Memory (OOM) is the primary issue we faced while training the models on the complete dataset. We tried to manage by using TPU.

We faced issues in assessing the model performance. The y_target value for the test data was not given. So, the only way to know whether a model was performing good or not was to submit the predictions (y_hat) of test data to kaggle. Upon submission kaggle will evaluate the results and give us the score. Only with the leaderboard position we can consider whether the obtained score is good. Also, there was a limitation of submissions where each group was allowed with only 5 submissions per day. We need to wait for another 21 hrs to make a new set of submissions. Though this is not an issue related to building a predictive model, it gave us an insight for datasets without y_target values.

CODE REFERENCES

[1] Kaggle.com. 2020. *INGV Volcanic : Basic Solution (STFT)*. [online] Available at: <https://www.kaggle.com/amanooo/ingv-volcanic-basic-solution-stft> [Accessed 8 December 2020].

Description: STFT feature extraction technique was referred from the above kaggle notebook. The dataset was formed with 100 features after the processing and it helped us improve the MLP model performance.

[2] Brownlee, J., 2020. *How To Develop An Ensemble Of Deep Learning Models In Keras*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/model-averaging-ensemble-for-deep-learning-neural-networks> [Accessed 12 December 2020].

Description: The above article explains in detail about the ensemble of MLP models for both Regression and Classification. In regression an average of the predictions from different models were taken whereas for classification majority rule is applied. We grabbed the concept from and wrote our own code for averaging the different model predictions.

[3] Kaggle.com. 2020. INGV Volcanic Eruption Prediction - LGBM Baseline. [online] Available at: <https://www.kaggle.com/ajcostarino/ingv-volcanic-eruption-prediction-lgbm-baseline> [Accessed 15 December 2020].

Description: FFT feature extraction technique was referred from the above mentioned kaggle notebook. The dataset was formed with 441 features after the processing. Baseline model was trained with this dataset. It gave better performance than training with RAW sensor data.

[4] Kaggle.com. 2020. INGV - Volcanic Eruption Prediction | Kaggle. [online] Available at: <https://www.kaggle.com/c/predict-volcanic-eruptions-ingv-oe/data> [Accessed 30 November 2020].

Description: Dataset was downloaded from the above kaggle link.