INGV - Volcanic Eruption Prediction

Chezka Sino
Machine Learning Engineer Nanodegree
Capstone Project

December 15, 2020

1 Project Definition

1.1 Project Overview

This Kaggle competition was launched by Italy's Istituto Nazionale di Geofisica e Vulcanologia (INGV) which focuses on geophysics and volcanology. The institute monitors seismic and volcanic activity all over the country.

The data provided by the competition organizers consist of 30GB of data collected from sensors around the volcano. Data is split into training and test folders where each file contains 10 minutes of logs from 10 different sensors around the volcano with 100Hz sampling rate. These have been normalized within each segment. The file train.csv contains ID code of each segment, which corresponds to the file names from the train folder, and the "time to eruption". The file sample_submission.csv is similar to train.csv but it contains the IDs from the test folder and the "time to eruption" are all set to 0 since this will be where the predictions will be entered.

1.2 Problem Statement

Scientists can predict the "time to eruption" based on the volcano and seismic activity but these patterns are hard to interpret. Their current way can predict for minutes in advance but not for long-term predictions. The competition aims to have a better model for long-term predictions of time to eruption.

This project consists of data exploration and pre-processing to ensure that the data would be fit for the model we would be using for training and predicting. As mentioned previously, our goal is to fit a model that would predict the volcano's "time to eruption" based on the sensor data logs from the different sensors on the volcano.

1.3 Metrics

The submissions for the competition would be evaluated based on the Mean Absolute Error of the predictions using the test data.

2 Analysis

2.1 Data Exploration

The train folder contains approx. 4,400 files pertaining to different segments, and these files have approx. 60,000 rows of data from 10 different sensors around the volcano. The train.csv file contains the time to eruption for each segment and this is what we'll be predicting.

After sorting the data based on the time to eruption, these are the segment IDs with the maximum and minimum time to eruption:

segment_id	**time_to_eruption**
1923243961	49046087
601524801	6250

Table 1. Maximum and Minimum time_to_eruption in Training Data

Checking these two segment IDs, we can see that a few of the sensor data were completely missed.

```
sensor_1      60001
sensor_2          0
sensor_3          0
sensor_4          0
sensor_5      60001
sensor_6          0
sensor_7          0
sensor_8          0
sensor_9          0
sensor_10         0
dtype: int64
```

Table 2. Number of NaN values in Segment ID 1923243961

```
sensor_1          0
sensor_2      60001
sensor_3      60001
sensor_4          0
sensor_5          0
sensor_6          0
sensor_7          0
sensor_8      60001
sensor_9          0
sensor_10       722
dtype: int64
```

Table 3. Number of NaN values in Segment ID 601524801

2.2 Data Visualization

The following figure shows the distribution of the time to eruption in the train data set. We could see that the distribution is somewhat uniform.

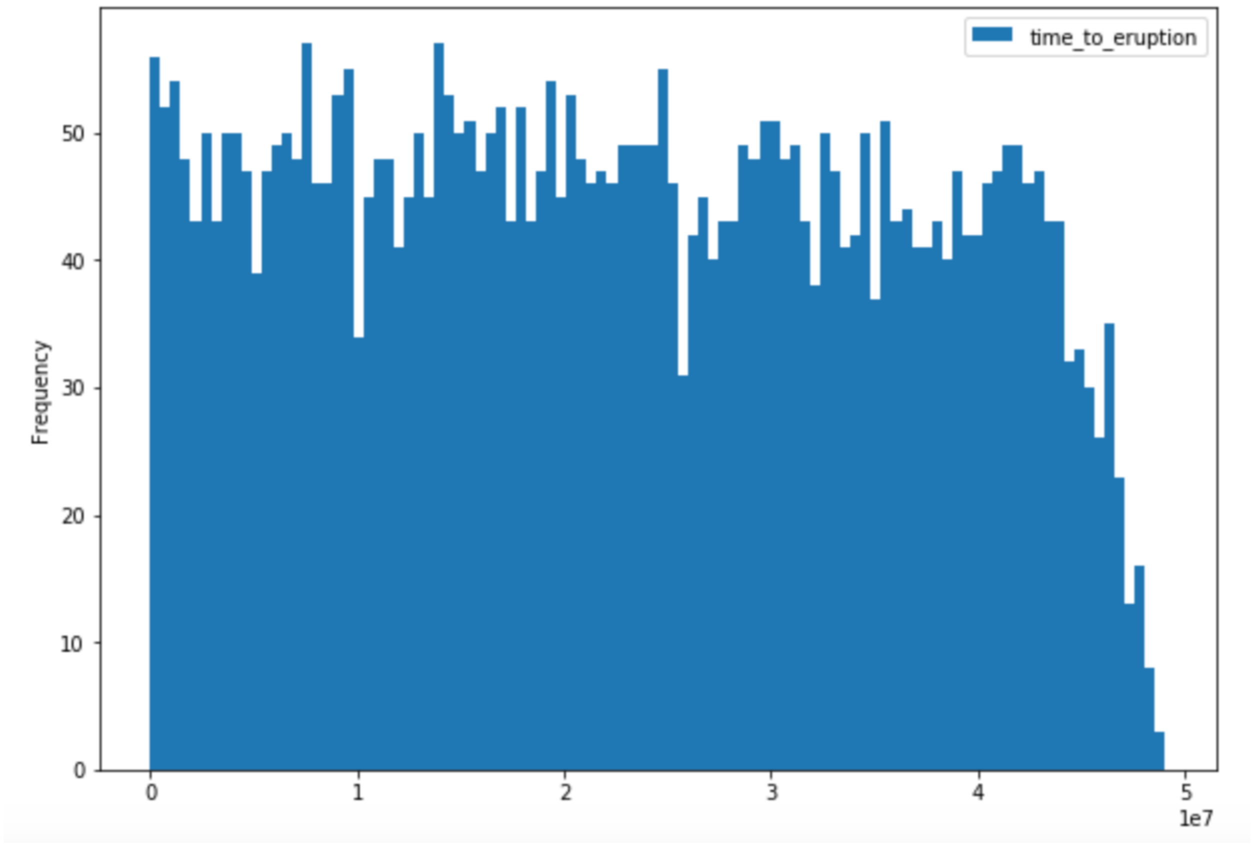

Figure 1. Distribution of time_to_eruption on Training Data

Let’s look at some like plots of the two segment IDs that we have previously looked at.

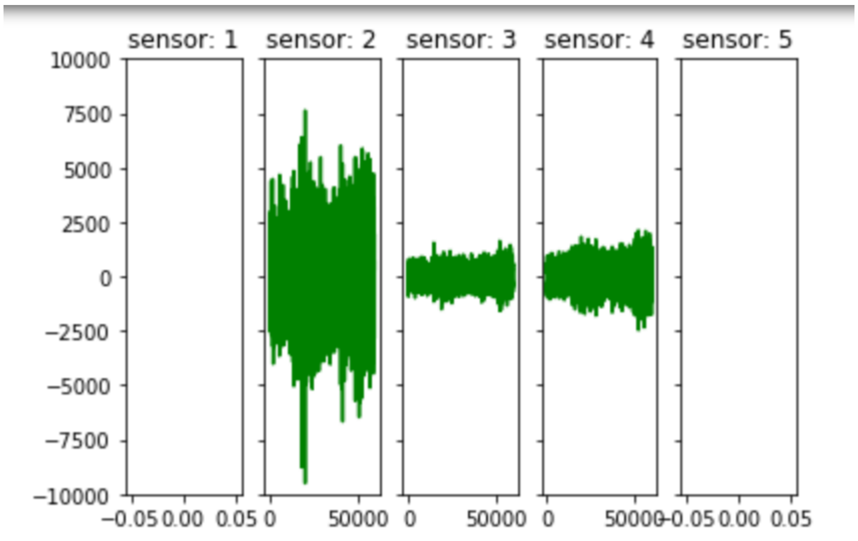

Figure 2. Line plot of sensors 1-5 of segment ID 1923243961

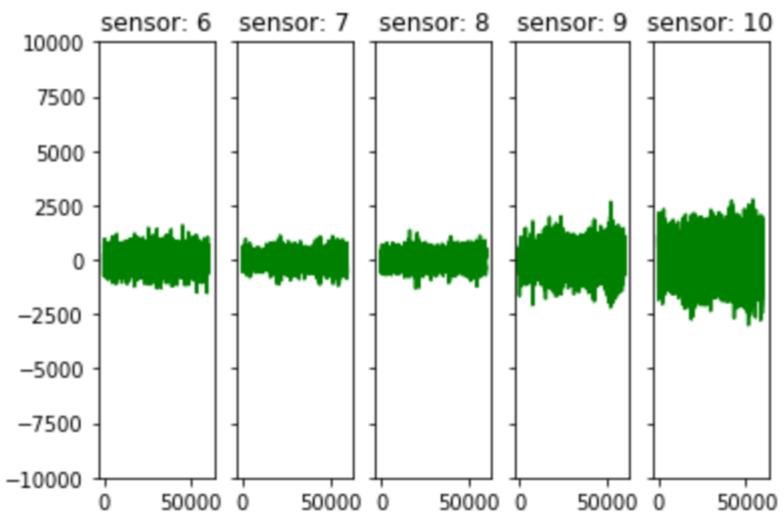

Figure 3. Line plot of sensors 6-10 of segment ID 1923243961

Looks like sensors 1 and 5 we completely missed in this segment. Sensor 2 seems to be the one with the most fluctuations compared to the remaining sensors.

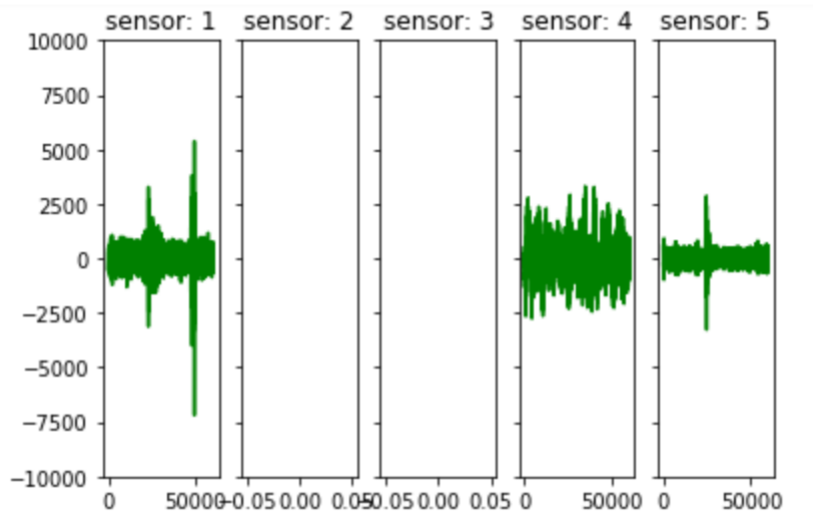

Figure 4. Line plot of sensors 1-5 of segment ID 601524801

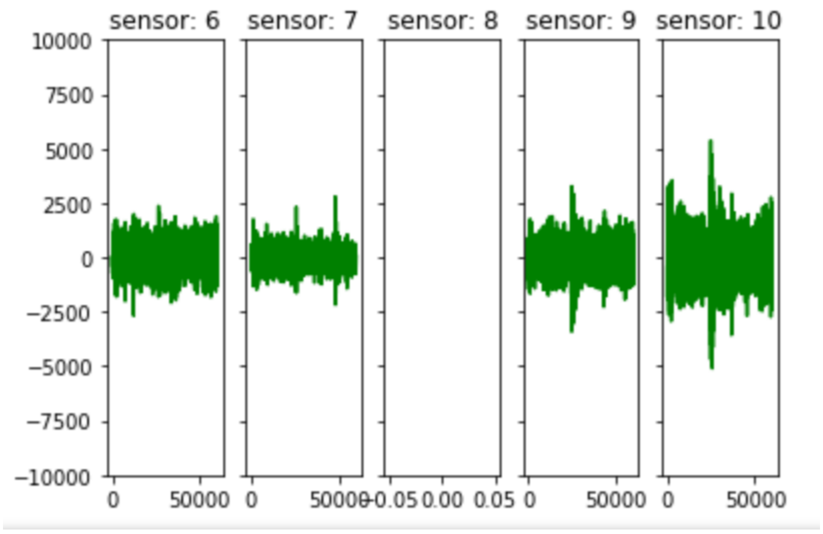

Figure 5. Line plot of sensors 6-10 of segment ID 601524801

Looks like sensors 2, 3 and 8 we completely missed in this segment. Compared to the segment with the longest time to eruption, there wasn't just a single sensor that had a wide range. We can see that sensors 1, 5 and 10 had a big fluctuation at roughly the same time. This may be an indication that these may indicate a closer time to eruption.

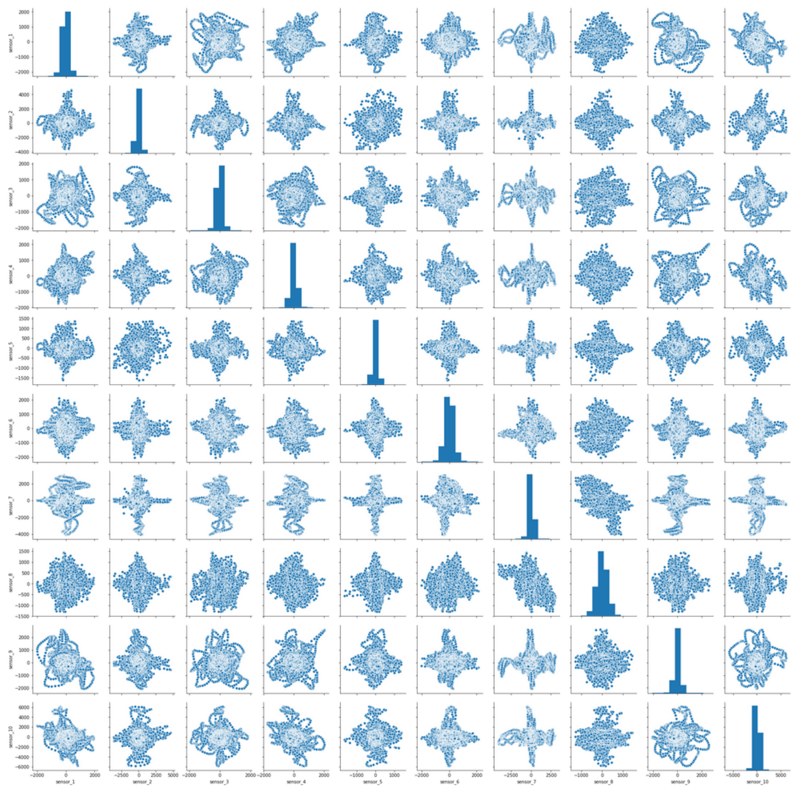

Figure 6. Pairplot of segment ID 513181 sensors

For this pairplot, I've looked at segment ID 513181 since this one has none of the sensors missed. Looks like there isn't any clear correlation between the sensors in the data.

3 Methodology

3.1 Data Preprocessing

Each of the training file corresponds to data for segment_id so these was processed and condensed to a single file where each row corresponds to a segment_id file. The following statistics were computed for each row:

- Mean
- Standard Deviation
- Maximum
- Minimum
- Mean Absolute Deviation
- Skewness
- Kurtosis
- Median
- Mode
- (Unbiased) Standard Error of the Mean
- Number of Unique Values

Additionally, the function fills the NaN values with an interpolation of the data.

The same preprocessing function is also implemented on the test data so we can use it on the model that we will train.

3.2 Implementation

The implementation is broken down into the following steps:

1. Splitting into training, validation and test sets
2. Training an XGBoost model
3. Hyperparameter tuning (after training the base model)
4. Testing on the test subset from the train folder
5. Deploying model on AWS SageMaker
6. Predicting on the test dataset and submitting to Kaggle

The initial XGBoost model parameters was been set as follows:

- max_depth=5
- eta=0.2
- gamma=4
- min_child_weight=6
- subsample=0.8
- objective='reg:linear'
- early_stopping_rounds=10
- num_round=200

3.3 Refinement

As previously mentioned, the model was refined with hyperparameter tuning and I've also tried training the model with a scaled dataset to see if that would improve the performance of the model. The following hyperparameter tuning parameters were used:

- max_depth: IntegerParameter(3, 12)
- eta: ContinuousParameter(0.05, 0.5)
- min_child_weight: IntegerParameter(2, 8)
- subsample: ContinuousParameter(0.5, 0.9)
- gamma: ContinuousParameter(0, 10)

4 Results

4.1 Model Evaluation and Validation

So for this project I've trained a model with unscaled data with and without hyperparameter tuning and the same with the scaled data.

The following table summarizes the performance of the different models on the Kaggle submissions:

	No Hyperparameter Tuning	**With Hyperparameter Tuning**
Unscaled Data	7284365	5672930
Scaled Data	11234233	11313082

Table 4. MAE of Each Trained Model

4.2 Justification

Based on the difference of the MAE on the test data, the best one was the model trained with the unscaled data with hyperparameter tuning. Using the scaled data made the predictions much worse than the ones with unscaled. It's not the best submission on the competition but further tweaking from the current best performing model might improve my rank in the competition.

5 Conclusion

5.1 Reflection

The project was created with the following steps:

1. Reading about the competition details and what its motivation is
2. Data preprocessing and feature engineering
3. Creating a base model
4. Hyperparameter tuning to improve model
5. Comparing the performance of models to "new" data

The beginning was the toughest part of this project. I've never worked with signal data previously so preprocessing and feature engineering basically consisted with condensing the dataset into different statistics. This was interesting to me and would probably be better if I could work on this with someone with knowledge on signal processing.

5.2 Improvement

I've only used XGBoost for this project, but I'm curious if other models would improve the model. I'm currently ranked decently (in my opinion) with what I've trained so maybe a bit more tweaks might improve the performance. Another solution might be feature selection since my preprocessed data ended up with 600 columns.

If I know a bit more about signal processing, then that might also give me more ideas on preprocessing the data for creating models.