Matthew Miller

Final Report:
LANL Earthquake Prediction

The Problem

Los Alamos National Laboratory (LANL), an institution dedicated to developing technologies addressing threats to national security, has teamed up with Purdue University to tackle the calamitous power of seismic shifts. Earthquakes are estimated to cost the United States $6.1 billion annually in infrastructure and about 20,000 deaths annually worldwide.

While we cannot prevent earthquakes from occurring, we can mitigate the damages by forewarning the people within the suspected region. By predicting when the next earthquake will occur, the people can take the necessary precautions to ensure their own safety, and informed early enough, can prevent damages within buildings and infrastructure as well.

Our target is to provide reliable predictions for the time until the next earthquake event. An important aspect of this project is the ability of the results to scale and generalize to real world data. Assuming the results scale appropriately, then aid services can prepare for expected injuries and provide the necessary precautions to prevent the loss of life by 5x. Additionally, the advanced and precise notice would allow preparations to be made, which can save costs in the range of $200,000-$5,000,000 depending on the location of the earthquake.

The Data

LANL has simulated many earthquakes in their laboratory setting collecting data on the events and released the datasets as part of a Kaggle competition. The experiments are performed by applying a shear force to a sample equivalent of earth and rock, which contains a fault line. Acoustic signals from the experiment are collected in a single sequence during which several simulated earthquakes are observed.

The data was collected in the form of an 8.89GB .csv file. The data contains the acoustic sample (independent variable) and the time remaining before the next simulated earthquake (dependent variable), without any null values. Considering there is only one sequence of data provided for the experiment it was important to create many more features, which would in turn create an even larger data set. As the necessary features were generated it became evident that we would be pushing the processing and memory capacity of my hardware to its limit. However, it was also clear that taking a subset of this data, one sample in every one hundred, would provide a suitable representation of the original dataset. Figure 1 displays the vibration amplitude received by the sensor in the form of an acoustic signal overlayed by the time to failure, or time until the next earthquake.

Exploring the Data

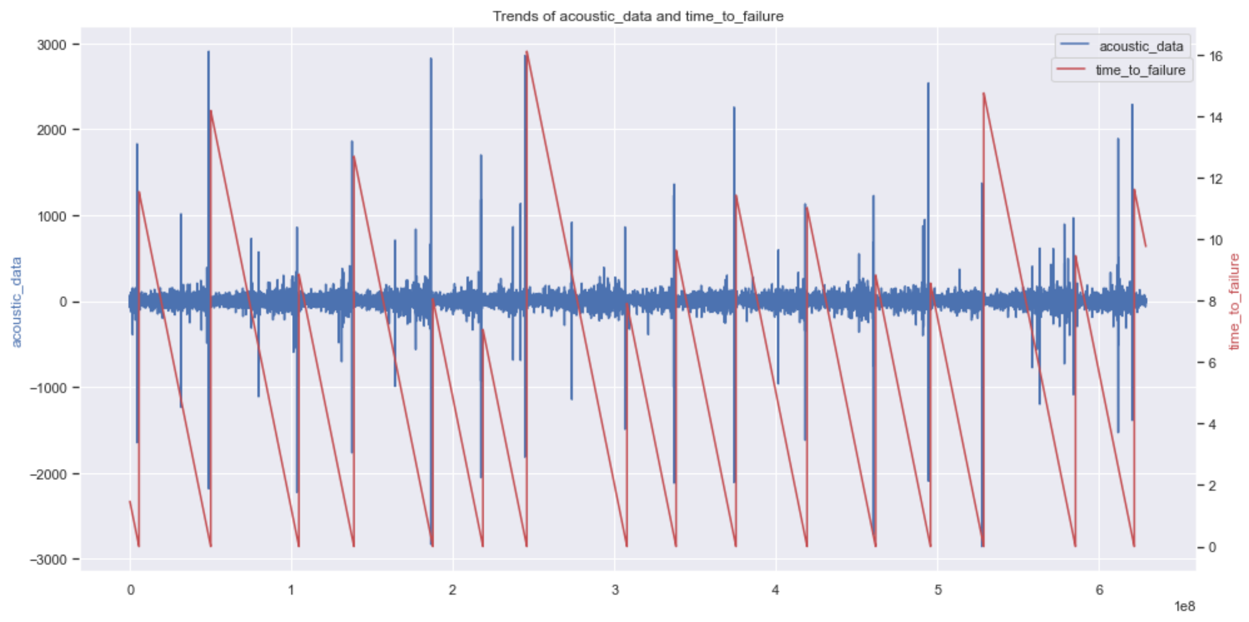


Figure 1. Acoustic data and time to failure

Predicting earthquakes is not a simple task, but the nature of the data lends itself to machine learning techniques related to time series analyses. The acoustic signal represents the amplitude of vibration collected by the sensor in the experiment, but we needed to extract information for the models to identify relevant patterns. Therefore, I used my experience with signal processing to explore and create several features from the acoustic signal. Figure 1 above displays the vibration amplitude received by the sensor in the form of an acoustic signal overlayed by the time to failure, or time until the next earthquake.

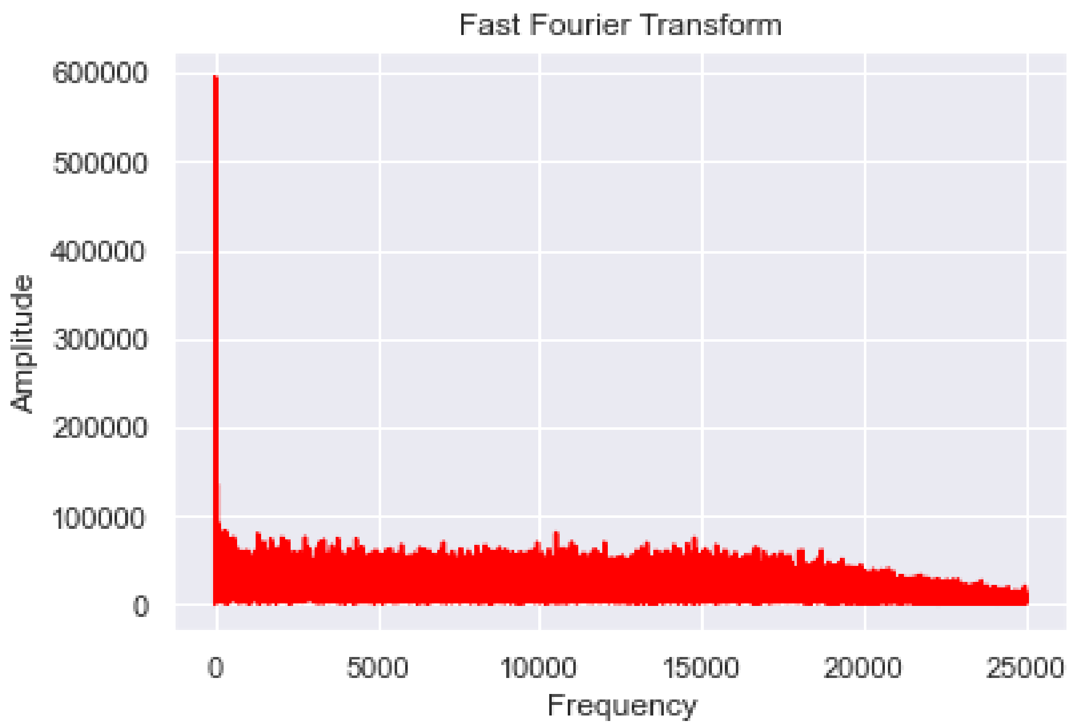


Figure 2. FFT of the acoustic signal

When analyzing a signal, it is often useful to look at the fast Fourier transform (FFT), which transforms the signal from the time domain to the frequency domain. Observing the signal in the frequency domain would allow us to determine if any frequency is dominating, and if so, then we could isolate that frequency to reduce the noise. However, as you can see in Figure 2, no frequency was found to dominate the signal, which is made up mostly of noise, as expected.

Next, we want to explore the time to failure, or time before the next simulated earthquake. In Figure 3 we examine the attributes of the time to failure as it steps closer to zero. We see in Figure 1 that the time continuously decreases until it approaches zero seconds. As the time falls the rolling difference between samples is negative up until the moment of the event, at which point the rolling difference is positive and we know the simulated earthquake occurs. Figure 3 shows to us that the time to failure is constantly changing between samples, but not at a constant rate. There are many step change decreases of varying sizes. This is an important characteristic of the data for time series analysis, as a constant rate of change in the time to failure data could bleed into our model predictions and produce poor results.

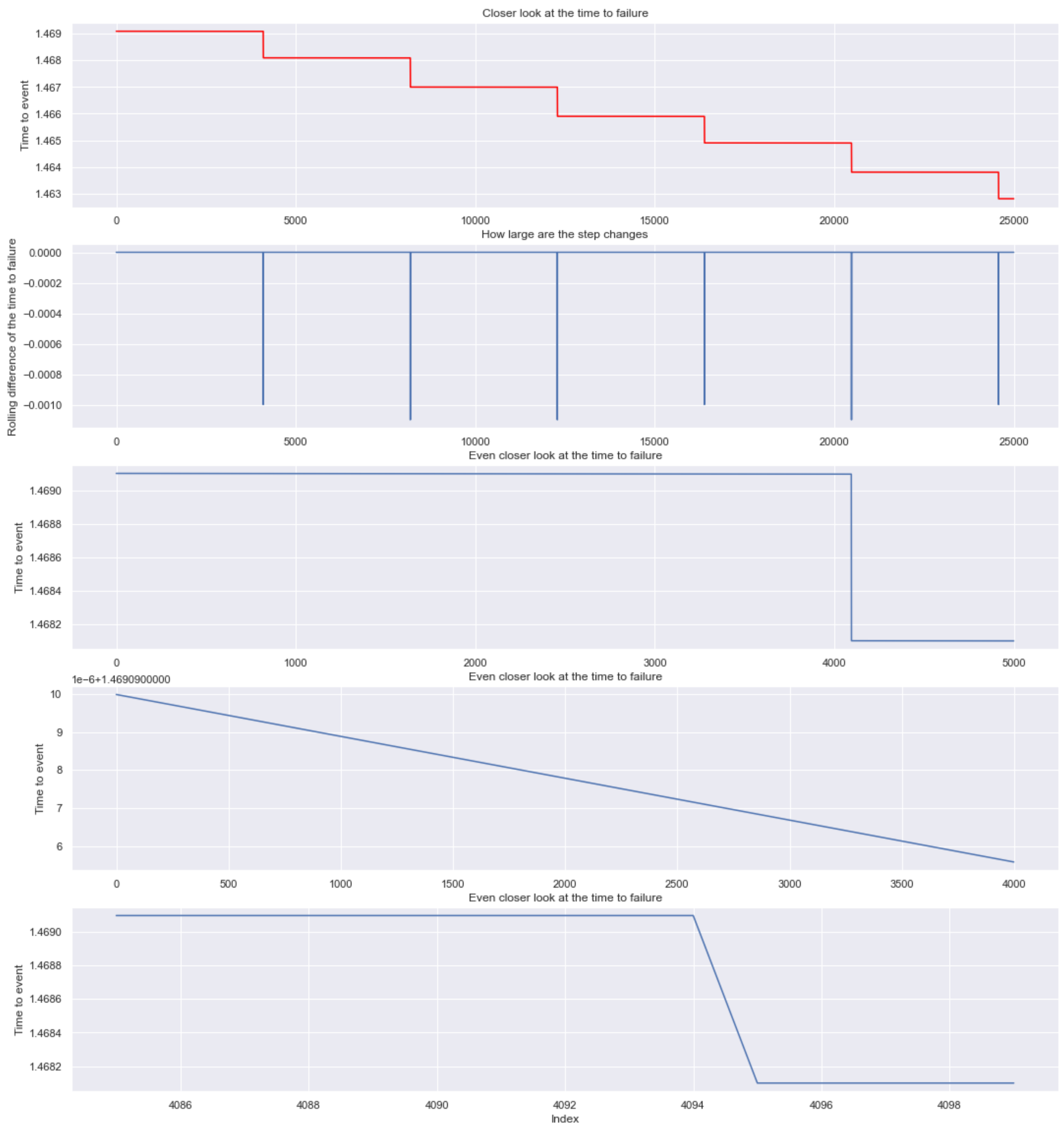


Figure 3. Time until next simulated earthquake

When generating features from the acoustic signal my intention was to isolate or highlight the impulsive signals that were observe before each earthquake. Experts in the field of seismology often attempt to identify an impending earthquake using the short-term-average and long-term-average ratios. These ratios rely on rolling averages of a small window size and a large window size to denoise the signal and better recognize any impulsive behaviors. In addition to the STA/LTA ratios we used skew and kurtosis rolling averages to further isolate the impulsive behaviors. Considering the time series data, it was not required to create dummy features for the regression models, nor was it necessary to normalize the data. I created rolling averages and rolling standard deviations of varying lengths to define statistical characteristics of the acoustic signal. In the end, I would develop twenty-two statistical features, all derived from the original acoustic signal, for the models to extrapolate patterns from.

Machine Learning and Results

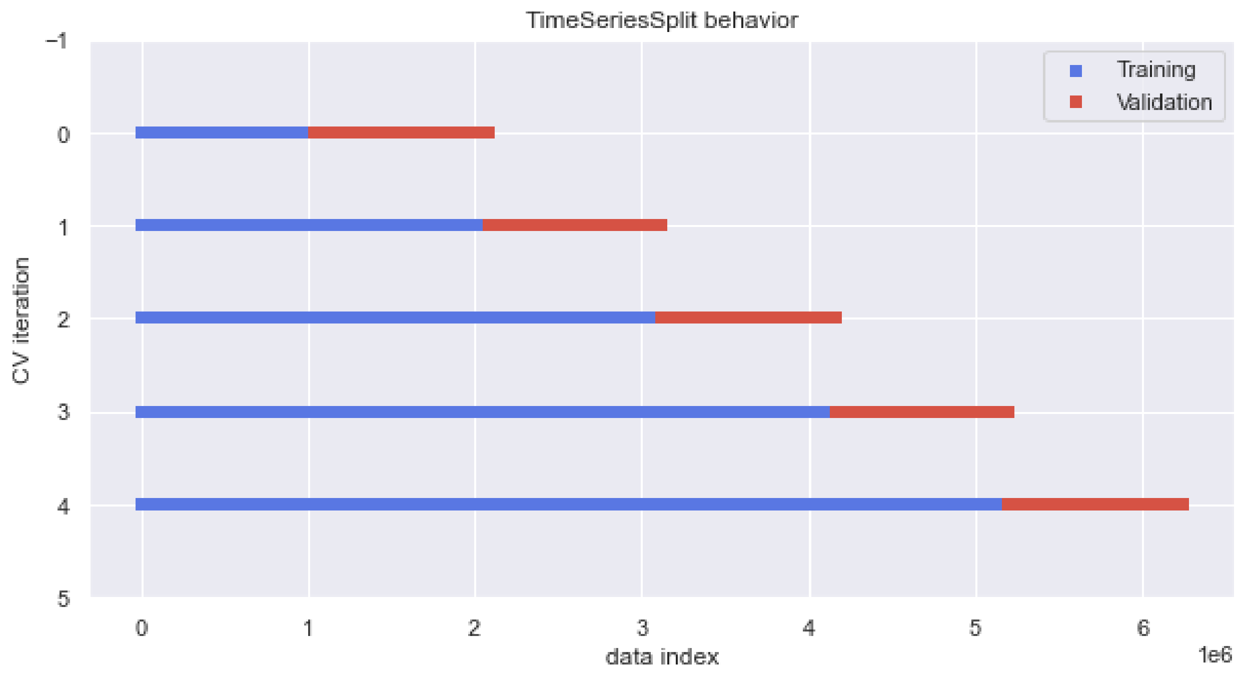


Figure 4. Time series cross-validation train/test splits

When splitting training and test sets of a time series dataset, it is important to maintain the sequential order. Therefore, while performing grid search cross-validation on our models, I used the TimeSeriesSplit from the sklearn.model_selection library. As you can see in Figure 4, using the TimeSeriesSplit maintains the order of the data as it iteratively progresses through a five-fold cross validation of the data. When applying the results of the grid search to our final model we split the train and test sets after the 12th simulated earthquake for approximately a 75%-25% split. In addition, I used an embargo technique to negate our training data from seeping into the test data. I removed data between the train and test sets the same length as our rolling windows.

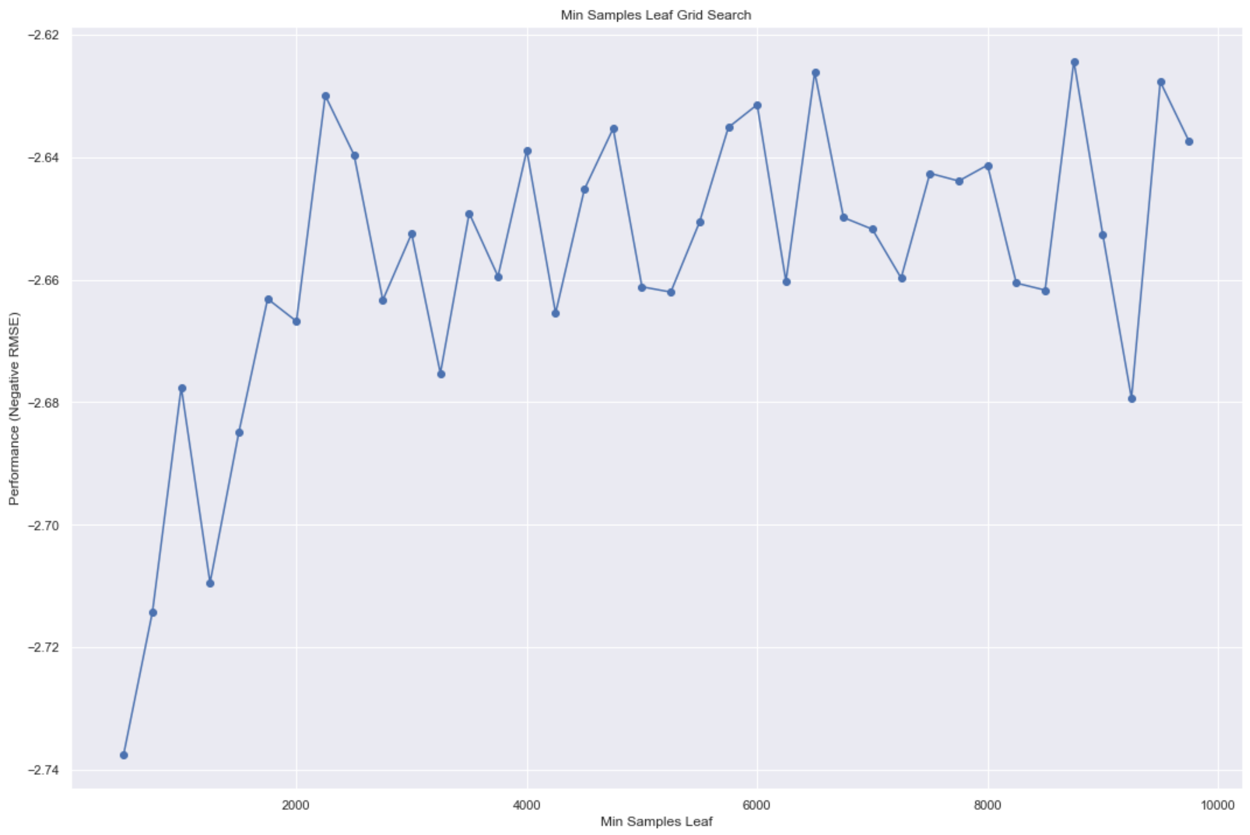
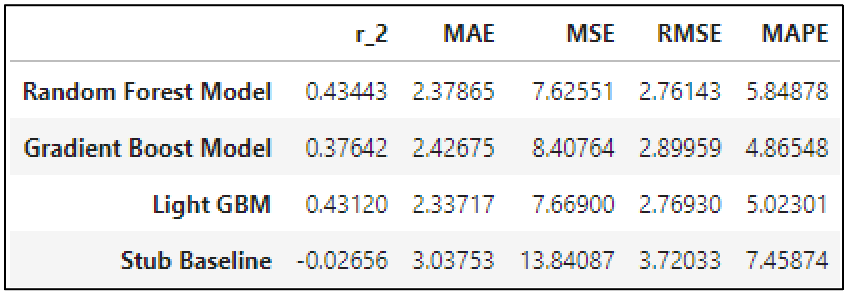


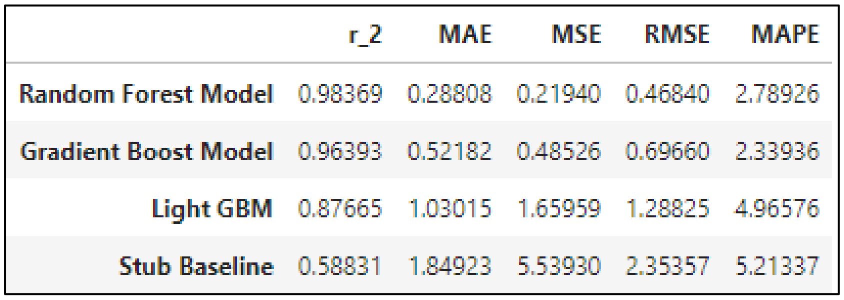
Figure 5. Grid search to tune min_samples_leaf hyperparameter

One of the main hyperparameters I was interested in tuning for the random forest regression model I would use, was the min_samples_leaf. The min_samples_leaf attribute of RandomForestRegressor from sklearn can help tremendously with issues related to overfitting on the training data. Therefore, I ran a grid search on the min_samples_leaf for quantities ranging from 500 to 10,000 at an interval of 500. As shown in Figure 5, we stopped seeing any continued improvement in the data after a quantity of ~2000, when controlling for the number of estimators at 100 and max_depth of 3.



	r_2	MAE	MSE	RMSE	MAPE
Random Forest Model	0.43443	2.37865	7.62551	2.76143	5.84878
Gradient Boost Model	0.37642	2.42675	8.40764	2.89959	4.86548
Light GBM	0.43120	2.33717	7.66900	2.76930	5.02301
Stub Baseline	-0.02656	3.03753	13.84087	3.72033	7.45874

Figure 6. Performance scores for each ML model (Test set)



	r_2	MAE	MSE	RMSE	MAPE
Random Forest Model	0.98369	0.28808	0.21940	0.46840	2.78926
Gradient Boost Model	0.96393	0.52182	0.48526	0.69660	2.33936
Light GBM	0.87665	1.03015	1.65959	1.28825	4.96576
Stub Baseline	0.58831	1.84923	5.53930	2.35357	5.21337

Figure 7. Performance scores for each ML model (Train set)

After running several iterations of each model, the random forest regression model proved to have the best results. Looking at the comparison tables (Figures 6 and 7) of the performance scores, specifically the RMSE, the random forest was the clearly most accurate model. For each model, the random forest, gradient boost, and light GBM, I performed a grid search and applied the best parameters to the associated model. The RandomForestRegressor was found to produce the best results considering the RMSE on the test data was lowest for this model.

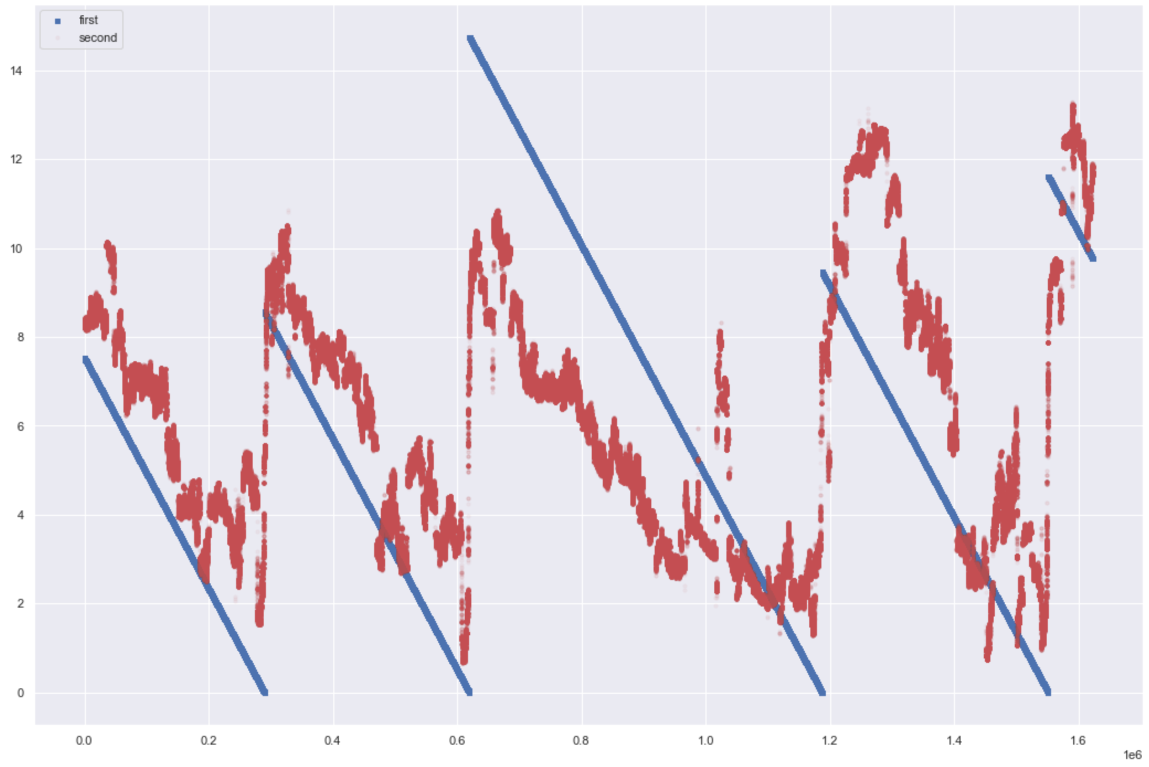


Figure 8. Random Forest time_to_failure predictions vs actual time_to_failure (Train Set)

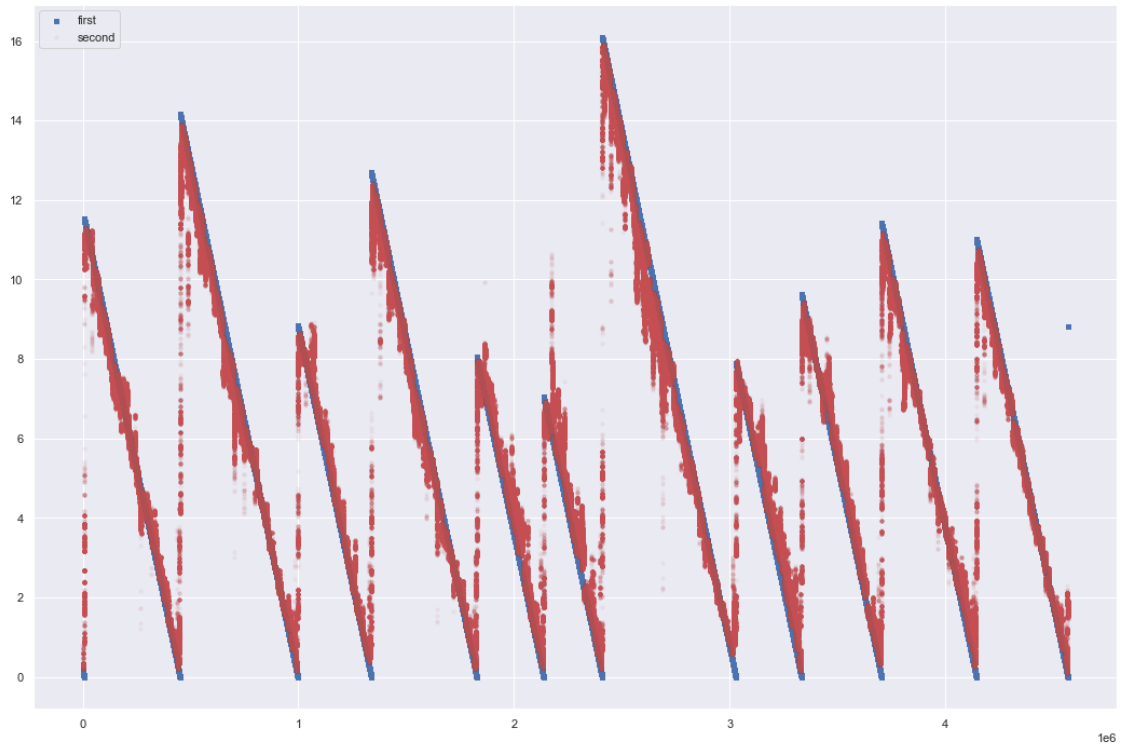


Figure 9. Random Forest time_to_failure predictions vs actual time_to_failure (Test Set)

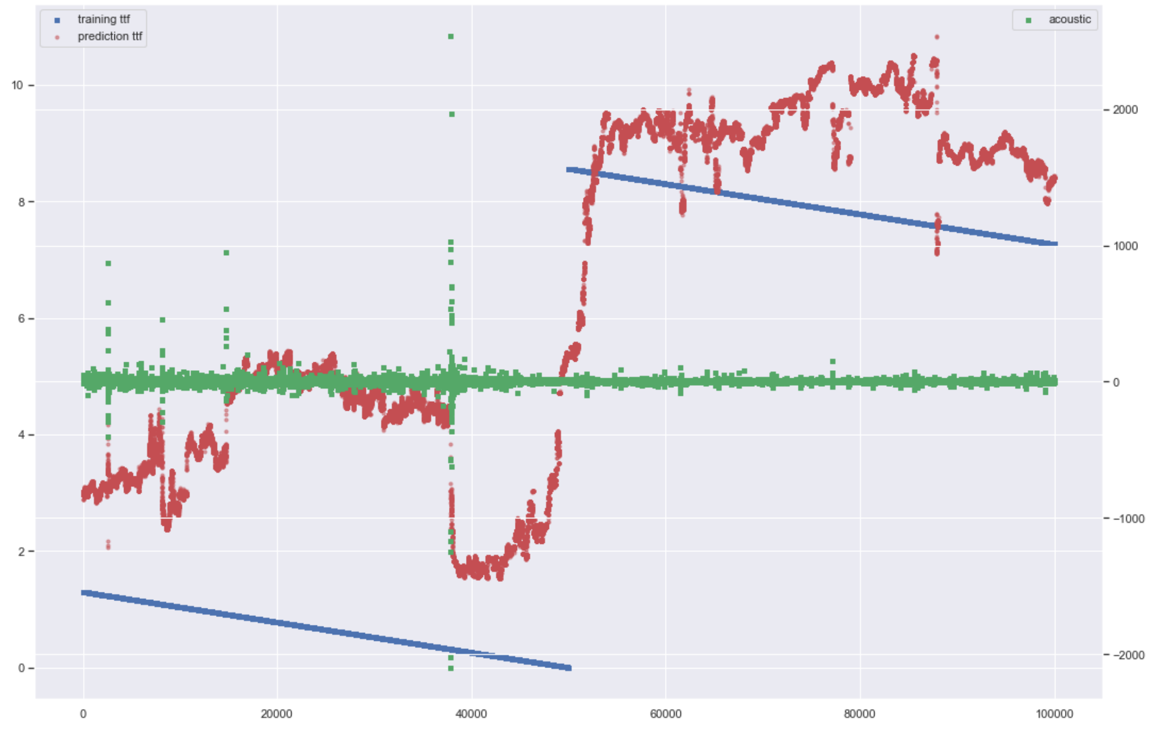


Figure 10. Acoustic data, test data ttf, and random forest prediction ttf

Figures 8 and 9 display the resulting random forest regression predictions on the test dataset and training dataset respectively. Overall, the model overfit to the training dataset even after attempting to curb this behavior through tuning the min_samples_leaf hyperparameter. Figure 10 overlays the acoustic data to the random forest predicted ttf and actual ttf for the 13th earthquake in the data. Here we are interested in the impact of the impulses impact the predictions on the test set made by my model. For each earthquake in the test set, we can observe a step change decrease in the predictions at or near the time of the large impulse in the acoustic data before a sharp rise to indicate the earthquake has occurred.

Further work would include collected more data on these simulated earthquakes. Sixteen earthquakes are a limited number of actual events and constrains our ability to properly model the events. We would like to research the scalability of the data by collecting and analyzing real-world vibrations used for tracking actual earthquakes instead of those simulated in a lab. Additionally, the premise for these experiments allows the observer to know precisely where the earthquake will occur, but we would like to look further at identifying the location of the next earthquake, which would be invaluable for saving lives and cost.