VSB Power Line Fault Detection

Machine Learning Final Project

Olivia Keirn
University of Pittsburgh
odk1@pitt.edu

Shanim Manzoor
University of Pittsburgh
shm150@pitt.edu

Christopher Tomei
University of Pittsburgh
cbt7@pitt.edu

ABSTRACT

Partial discharge (PD), a large change in a signal measurement on power lines, damages a power line – leaving it out of service which can have detrimental impacts for the regions this power line provides power to. This makes it extremely important to be able to determine if a power line has been damaged by partial discharge very quickly, without physically inspecting each line seeing as that would be time consuming. Using signal measurements on each power line can help with this task. Since faults in power lines are not common, the data set is extremely imbalanced with over 90% of the instances classified as 0 (no fault). By visualizing the data we were able to come to two major conclusions. First, the change in values for a power line with PD is much larger than the change in values for a power line without PD. Second, it is extremely difficult to differentiate between instances with PD and instances without PD. They are not clearly grouped separately, rather scattered among each other.

Our first classification approach was to classify each instance as the majority, but did not perform well on the test data because it must be more evenly distributed than the test data. Second, we tried random forest, which scored high, but performed very poorly. Our next model was with K-means, which scored okay, but performed significantly better. Lastly, we tried a long short-term memory neural network and performed well, but with a low precision. With more computing power and time, we will be able to scale our test data given by Kaggle and apply both the models we have running and also apply a support vector machine model, recurrent neural network, and try feature extraction.

KEYWORDS

Classification, Imbalanced Data, K-Means, Neural Networks, Partial Discharge, Power Lines, Prediction, Random Forest, Time Series

ACM Reference format:
Olivia Keirn, Shanim Manzoor, and Christopher Tomei. 2019. VSB Power Line Fault Detection. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 8 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Overhead power lines cover huge areas in order to deliver power to cities, villages, and any customer in between. Since these power lines carry a large responsibility to its customers, it's important to make sure they are running correctly. With the large distances, manual inspection of damage is out of the question. VSB (Technical University of Ostrava) has added meters to power lines in order to read signals along them, so our challenge is to find which signals represent partial discharge (PD). PD damages the power line over time which will end up in power outages if no action is taken, which is what needs to be avoided by finding a reliable and accurate way of detecting PD so there can be maintenance before it's too late.

Partial Discharge is an incomplete electrical breakdown between two conductors and is the result of a high amount of electrical stress on the conductor surface. The occurrence happens in a very short time period or 2-5 nanoseconds (ns). It is hard to detect on normal lines because of the high amount of insulation which won't allow for easy inspection for an outsider and will damage the conductor before long. It can be considered as a symptom for poor manufacturing. PD can be evaluated in 2 ways: with measurement of voltage signal of electrical stray field across the covered conductor (CC).

2 DATA SET

In this ML problem we are trying to detect a very dangerous phenomenon in electrical equipment in the high voltage bracket. Here, there is a 3 phase signal for each line being considered and each signal has 800,000 measurements taken over 20 milliseconds. This entire measurement would encompass a cycle (50Hz). In both the train and test set, each instance or signal is a column of 800,000 measurements, which is oriented opposite to the usually row wise instance we usually observe. These files are both in Apache Parquet (Hadoop) formats, so the pyarrow library was used to decipher this in Python. Also, each signal is unique across the training and test datasets.

The metadata .csv files have an id_measurement (the ID code for a trio of signals recorded at the same time), a signal_id (the foreign key for the signal data), and phase feature (the phase ID code within the signal trio, which may or may not all be impacted by a fault on the line). The train metadata file also has a target feature which is 0 if the power line is undamaged and 1 if there is a fault [13].

3 RELATED WORK

Misak, et. al. describes a PD detection experiment and steps taken to process data. They recommend various methods of removing noise and identifying PD patterns. Evaluates different denoising and detection algorithms and their performance. Ultimately, AI algorithm SOMA (Self-Organized Migrating Optimization) achieves best performance [9].

Prilepok and Vantuch proposed a method for classifying partial discharge patterns with 4 steps: signal denoising, signal normalization, signal encoding (with codebook), and creating fuzzy signatures. The fuzzy signatures are strings, so they use TF-IDF to weight each code from a signal. They then used kNN to classify signals as fault or no fault. Within the fault classification, there are different types that were also used as classifications. Since the signals they used were real-world data, there are significantly more no-fault signals than fault signals, which caused classification issues. To try and fix this, they used an under-sampling method, but it still had problems. Their method ended up producing 0.7671 accuracy [10].

Vantuch, the creater of this Kaggle competition, wrote his thesis on this exact problem. For him, to evaluate PD, two methods were used. The first consisted of measuring the current signal in CC by a sensor and the second consisted of measuring the voltage signal of electrical stray along the CC. The data produced from these methods is imbalanced however because there are significantly less cases of PD activity than failure free signals [12].

A few techniques were suggested to denoise the data including threshold and wavelet-based decompositions before applying the machine learning algorithms. To reduce the dimension of the data, PCA, non-negative matrix factorization, and autoencoder were implemented. Vantuch tried multiple algorithms such as ANNs, SVM, adaboost, random forest, extreme gradient boosting, grid search, swarm intelligence based optimization, and self organizing migrating algorithms [12].

Godahewa realizes that time series data deals with classifying data using behavior of dataset over the temporal dimension. For time series features, generally feature scaling, entropy, stationarity, correlation structure, and distribution are used. Since anomaly detection is the major use, PCA (or principal component analysis) has been promoted a lot by researchers. Ben Fulcher used a greedy approach to select optimal features after generation. Another method involves the use of neural networks or specifically LSTM (long short-term memory) [6].

Himmetoglu goes into the usage of deep learning with the usage of Convolutional neural networks and LSTMs. It showed similar results when a dataset with deep learning was compared with a dataset with pre-engineered features without deep learning. This is useful as the analyst won't need to be a domain expert to create features. It also suggests a mix of LSTMs and CNNs for large datasets which is extremely relevant for this project [7].

Soni talks about different metrics such as precision, recall and AUCROC (area under curve and receiver operating characteristics). It further suggests the use of cost sensitive learning where minority mis-classification is penalized more after converting the entire set into a cost matrix. Sampling techniques help and an advanced one that can be used is SMOTE, which forms new instances of minority classes using convex combinations. This helps in balancing the data. Anomaly detection can be used and Soni suggests the usage of clustering methods such as one-class SVMs and Isolation Forests [11].

Maalouf and Trafalis understand that while rare events poses several problems to classification algorithms, they tend to have a greater value when they are correctly classified. They realize that sampling is a very important - if not the most important - technique when dealing with rare events. There are two types of sampling: basic or intelligent (which can be broken up into under-sampling and over-sampling). They have found that undersampling is the superior sampling method for most classification methods. Maalouf and Trafalis implemented Rare Event Weighted Kernel Logistic Regression (RE-WKLR) with positive results. Using 8 different datasets, they used RE-WKLR, SVM, and TR-KLR (Trauncated Kernel Logistic Regression) by creating both balanced and imbalanced training sets. For the balanced training data, the majority had highest accuracy with RE-WKLR and with the imbalanced dataset all had the highest accuracy using RE-WKLR [8].

Aminghafari, et. al. talks about combining PCA and normal wavelet denoising done in signal processing. Originally, a simple regression model is considered and universal threshold is considered for simplicity after deciding the correlation between the noise, which is structured in two components. For noise covariance, MCD (or minimum covariance determinant) is used. PCA is then used, not to build features, but to eliminate those which are adding to the noise. MSPCA (or multi-scale PCA) is useful for dimensionality reduction. Adding to MSPCA, this research uses the previously mentioned threshold and the noise covariance matrix (MCD) to effectively perform the denoising on the dataset [5].

4 DATA EXPLORATION

Our data set is very imbalanced with only about 6% of the training data targets equal to 1, which means that there is a fault in the line. This means that approximately 94% of the training data targets are equal to 0–no fault in the line. We can see the proportion in the bar graph in Figure 1.

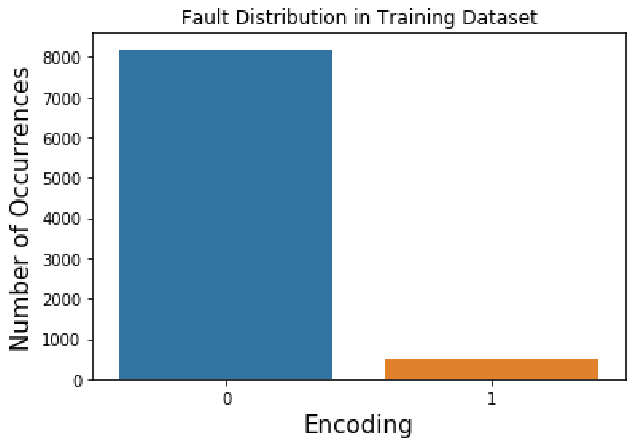

Figure 1: Training Data Target Distribution

Since the signal data is so large (800,000 instances) we took a look at the values in the first 15 signal ID's from the train data–in Figure 2–and the first 20 signal ID's from the test data–in Figure 3. We can see that for both train and test, the median hovers right around 0, but with values mainly ranging from -50 to 50 and outliers ranging from -150 to -50 and 50 to 150.

Figure 4 shows the first 3 columns in the training dataset, which are the 3 phases of the first signal, and there is no partial discharge (PD).

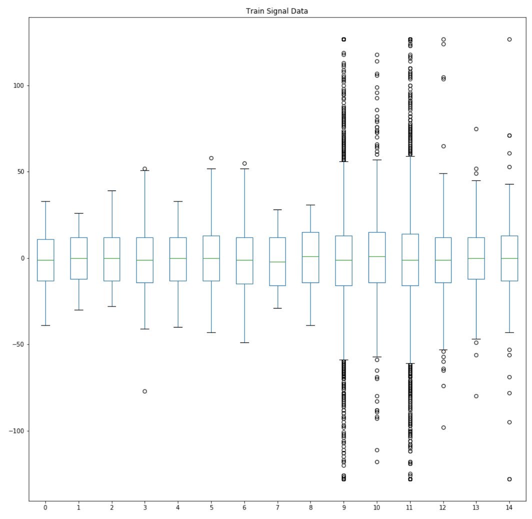

Figure 2: Training Data Signal Data

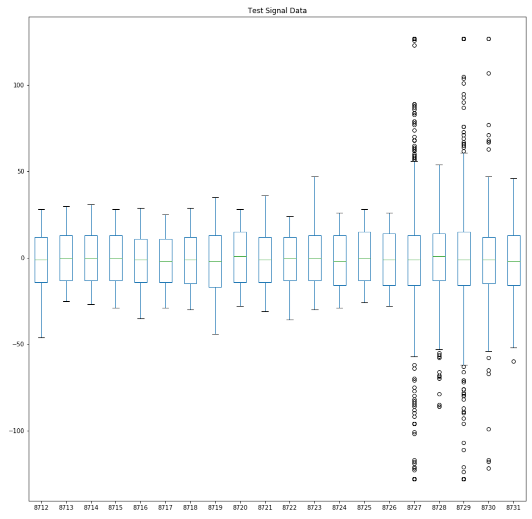

Figure 3: Test Data Signal Data

Figure 5 shows the first three phases of the first signal that has partial discharge. Both Figure 4 and Figure 5 have no feature scaling.

Next, we tried scaling using minmax scaler in Python's preprocessing library. In terms of using feature scaling, if tried on the entire dataset, any contemporary GPU/processor would fail due to memory error. So, for the purpose of the visualization, only the

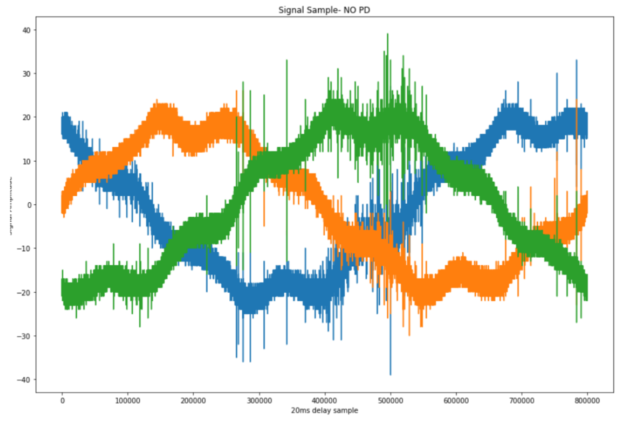

Figure 4: Waveforms Without Scaling and No Partial Discharge

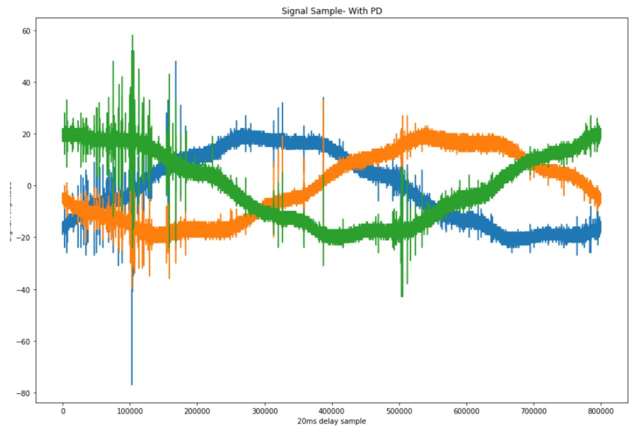

Figure 5: Waveforms Without Scaling and With Partial Discharge

first 10 columns were considered. We can see in Figure 6 the same data as Figure 4, but with feature scaling.

And Figure 7 shows the same data as Figure 5, but with feature scaling.

As stated previously, partial discharge is a significant change in signal between measurements. Since the data we have is in sequence – time series – we can see the change in signal between measurements by calculating the difference between them.

Figure 8 represents an instance that does not observe partial discharge, while Figure 9 represents an instance that does observe partial discharge. In Figure 8, we can see that the largest signal change for an instance that does not have partial discharge is around 25, but the largest change for an instance that does have partial discharge is a little under 50, which can be seen in Figure 9.

5 FEATURE ENGINEERING

During our trials, we needed to generate features as this is a univariate signal measured over 160,000 seconds (44.44 hours). Using one of the popular libraries, called tsfresh, which has the capability

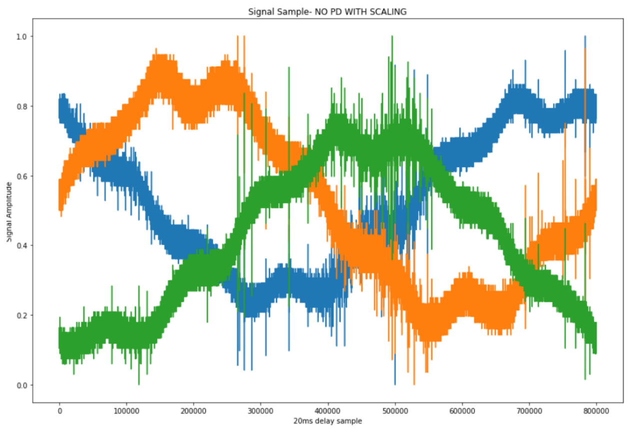

Figure 6: Waveforms With Feature Scaling and No Partial Discharge

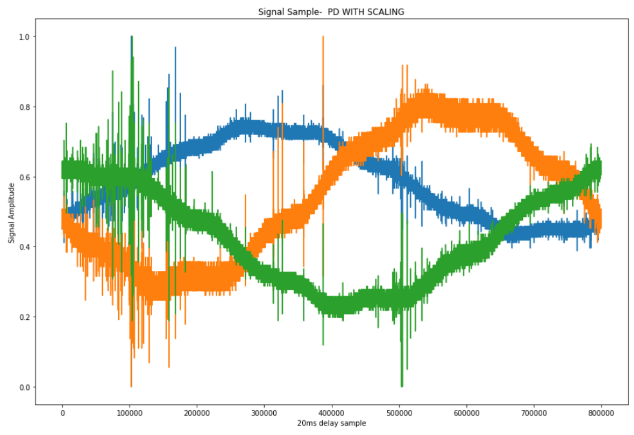

Figure 7: Waveforms With Feature Scaling and Partial Discharge

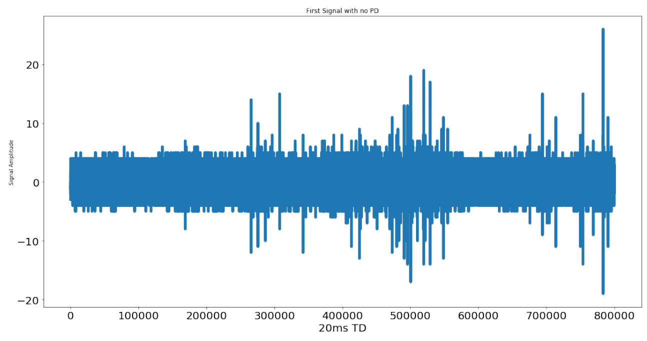

Figure 8: Difference Between Signals – No PD

to extract some very useful features from a time signal including mean, number of peaks, auto-correlation, kurtosis, etc. The advantage here would have been the ability to use multiple features for the training and test set and create supervised learning models

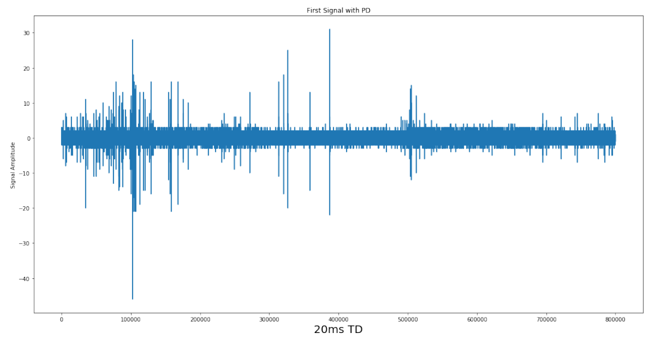

Figure 9: Difference Between Signals – With PD

using them. During the training, our cloud platform failed multiple times, even at 624 GB RAM as it kept getting out of memory. The process is in itself extremely intensive and due to this, results were unable to be obtained. Even after using only an eighth of the data set, these issues kept arising. Based on some discussions on various platforms, since tsfresh uses pandas and numpy in the back-end, the question of their scalability for such large data sets have come into play and might have factored into the issue we faced.

6 DIMENSIONALITY REDUCTION

Our training dataset that we were working with has 100 times more features than instances – 8,712 instances vs. 800,000 features – which makes it very difficult to work with. It also is hard to visualize the data with this many dimensions since we can really only visualize up to a 3rd dimension. We decided to try two different dimensionality reduction techniques to solve this.

6.1 Principle Component Analysis

Principle component analysis (PCA) is used as preprocessing for supervised learning and visualization. Principle components are a smaller number of representative features that explain most of the variability in the original data. PCA represents a low-dimensional representation of the dataset by finding a sequence of linear combinations of the original features that have maximal variance and are mutually correlated [4]. For our dataset, we applied PCA with two principle components in order to visualize our data better.

In Figure 10, the white points with a blue outline are instances that measured 0 (no fault) and the black points are instances that measured 1 (fault). It reinforces that the fault and no fault instances are very close together – scattered among each other – making distinguishing them a momentous task.

6.2 t-SNE

t-distributed stochastic neighbor embedding (t-SNE) is a non-linear dimensionality reduction technique and is used for visualization – normally used to create two new features. This transformation depends on how points are in their original space by trying to make points that are close in the original space closer in the new space and points that are far apart in the original space are farther apart in the new space. t-SNE uses joint probabilities and Kullback-Leibler divergence for this [4].

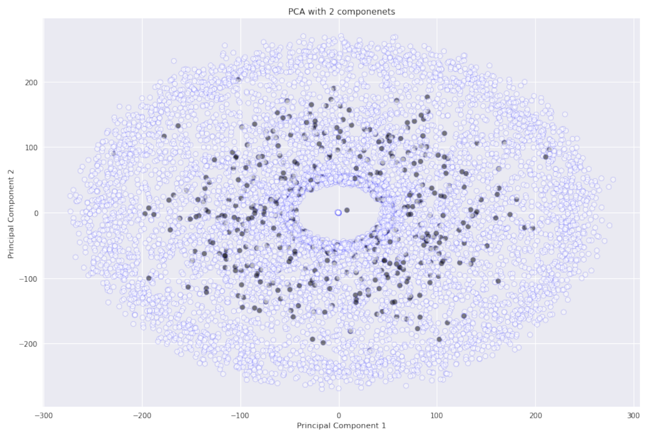

Figure 10: Principle Component Analysis: p=2

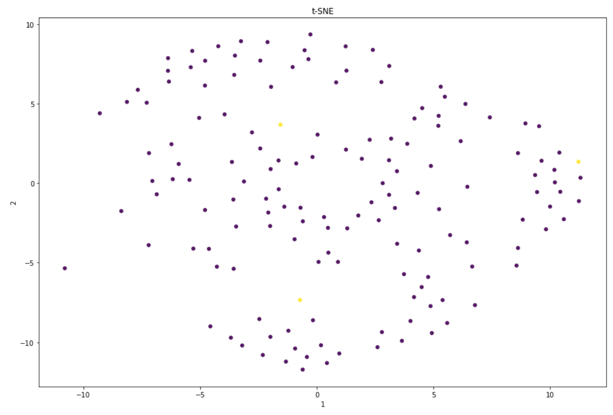

Figure 11: t-Distributed Stochastic Neighbor Embedding

In Figure 11, the purple points represent instances that have a label of 0 (no fault), while the yellow points represent instances that have a label of 1 (fault). As noticed in Figure 10, Figure 11 also shows that fault and no fault instances are scattered among each other and are difficult to distingush.

7 CLASSIFICATION METHODS

7.1 Classify All as Undamaged (0's)

Since the dataset is very unbalanced - many lines with no fault - we decided to see how well a submission with each prediction as 0 would do in the Kaggle competition. To evaluate how well your predictions fare against the actual values, Kaggle uses the Matthews correlation coefficient (MCC) which is defined as [13]:

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

A coefficient of +1 indicates perfect predictions, 0 indicates no better than random, and -1 indicates complete disagreement between the predictions and actual values. This submission with all zeros also gave us a score of 0, basically as good as random.

7.2 Random Forest Classifier

Decision trees segment the feature space into multiple regions using a set of splitting rules which are represented as nodes, continuously splitting until reaching a terminal node. Random forests builds multiple decision trees and in each tree, a randomly selected subset of features is used and then find the best one for splitting. For classification – like the one we used – it then aggregates the results through voting [1].

We tried two different random forest classification methods. The first without cross-validation and the second with cross-validation.

For the random forest classifier without cross-validation, we used 500 estimators and default parameters. This gave us a score of 0.9377. However, we can see in the confusion matrix for this classifier (shown in Figure 12) that this model only predicts 9 instances to have partial discharge damage – and none of these are actually observations with damage. This gives us a precision score of 0% and a recall score of 0% as well, which means that although this model scores well, it does not perform well for the data.

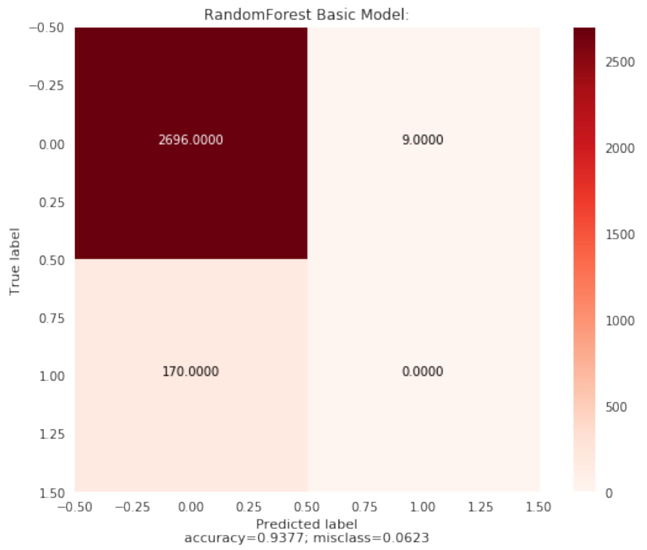

Figure 12: Random Forest Confusion Matrix – No Cross-Validation

For the random forest classifier with cross-validation, we used a random_grid to go through the values in Table 1 for the given metrics. We found that the following values performed best. For n_estimators: 500, max_features: 'auto', min_samples_split: 8, and max_depth: 15 gives us the best performance. This gives us a score of 0.9409.

From the confusion matrix, given in Figure 13, we can see that although this model with cross-validation performs better than the random forest model without cross-validation, it does no make a single prediction of 1, a power line with partial discharge damage. This also gives a precision and recall score of 0%. With these results in mind, we decided to test other models to increase accuracy, precision, and recall.

Parameters	Values
n_estimators	2, 4, 8, 16, 24, 48, 80, 100, 200, 300, 400, 500, 750, 1000
max_features	'auto', 'sqrt'
min_samples_split	2, 4, 6, 8
max_depth	5, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80

Table 1: Random Forest Cross Validation Values

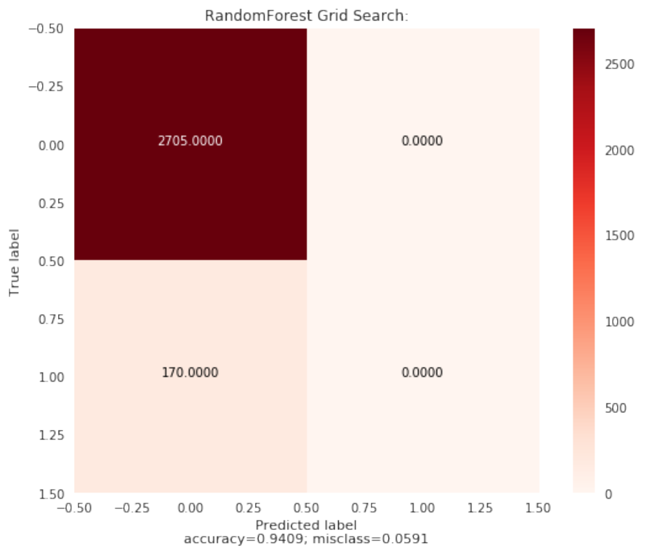

Figure 13: Random Forest Confusion Matrix – With Cross-Validation

7.3 K-Means Classifier

K-means classification splits up the data into a specified number of clusters – k. To do this, the within-cluster variation must be found, using:

$$\sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2 = \sum_{i \in C_k} ||x_i - \mu_k||^2$$

since we are using euclidean distances. This turns into an optimization problem because we want to minimize this within-cluster variance. Initially, each observation is assigned a cluster, then the centroid of each cluster is found, and finally each observation is assigned to the cluster with the closest centroid [4].

For our model, we used a time series k-means classifier, since our data is a time series. We used the following parameters: n_clusters = 2, metric = 'euclidean', max_iter = 5, and random_state = 0. This gave us a score of 0.5266. The confusion matrix for this model is given by Figure 14. We can see that this model was able to predict power lines as damaged. This k-means classifier found 1,361 instances to be damaged, but only 85 were correctly predicted. It also found 1,514 instances as not damaged by partial discharge, with 1,429 being correctly classified. This gives us a precision score of 6.25% and a recall score of 50%. These both clearly are better than the precision and recall measures from both random forest models, so this is promising.

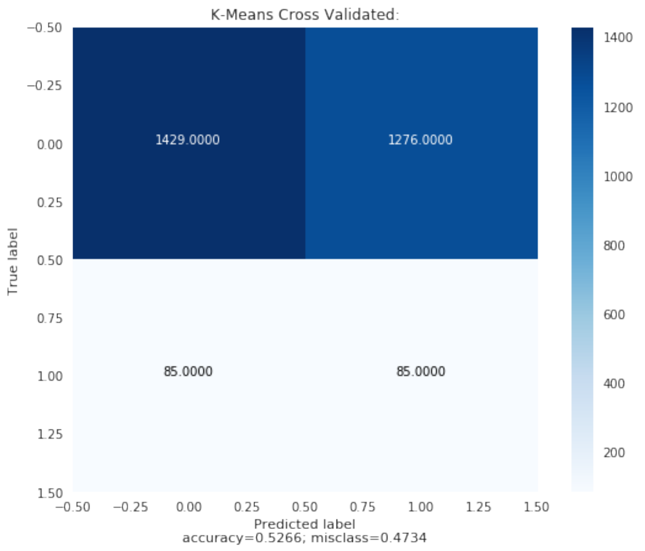

Figure 14: K-Means Times Series Confusion Matrix

7.4 Neural Networks

Neural networks are based off of the human brain and extend linear regression (for regression problems) or logistic regression (for classification problems) by adding hidden layers. Each hidden layer has multiple hidden units which represents intermediate processing steps and a non-linear function is applied at each hidden unit in order to learn complex boundaries. An activation function is then applied at the hidden layer and the output layer for classification has a neuron for each class (k) [3].

Long short-term memory (LSTM) model is a type of neural network. The purpose of this model is to learn when to include important input, when to consider long term memory, and when to shut off the output. It has a short term and a long term state and the network learns what should and should not be stored in the long term state. The output of the state is controlled by the output gate, the input gate controls the part of input that should be added to the long term state, and the forget gate control which part of the long term state to be discarded [2].

On Kaggle, the highest scoring groups or individuals used LSTM to solve this problem [13], so we decided to put it to the test as well. Since a neural network takes a lot of computing power, we were slightly limited (only being able to run it on 500 instances for time constraint), but were able to run it. Our LSTM used mean square error for the loss function, the Adam optimizer, 2 epochs, and a batch size of 100. This gave us an accuracy of 0.8640, precision score of 7.69%, and a recall score of 8.57%.

8 DISCUSSION

The challenges associated with processing this quantity of data were many. We were successful processing data not on the CRC cluster, but on Google Cloud Platform Unix-based virtual machines running Jupyter Notebook. With the help of SSH port forwarding I was able to maintain 4 quad-CPU 600GB+ RAM virtual computers to process these results in isolation from each other. Kernel

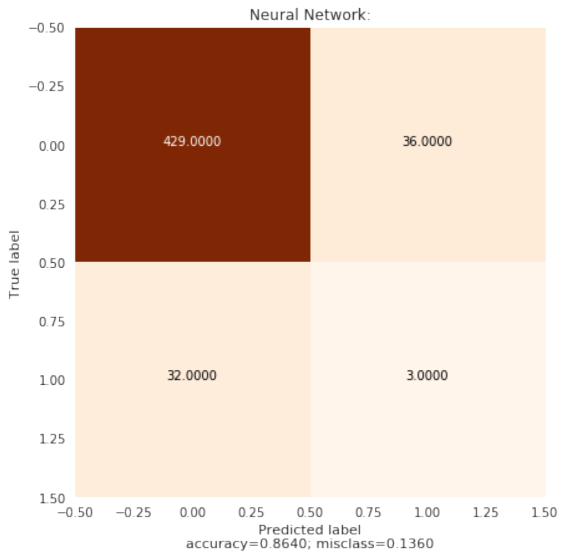

Figure 15: LSTM Confusion Matrix

crashes were too common to use on monolithic machine. Before and during moving to GCP we received all manor of out of memory errors. Some were clear such as "'i' format requires −2147483648 ≤ number ≤ 2147483647" but such an error indicates some 32bit underpinnings of the libraries we used. While this error was clear it was not documented and only after much reading did we learn that the mechanics of the what was really happening with tsfresh: "OverflowError: cannot serialize a string larger than 4Gib", again indicated that not all subroutines were using 64bit components. This does not present a problem to most "Big Data" scientists, yet we experienced those errors with regularity. Our research brought us to the Pandas bug report board, where we saw another user with a 2TB RAM heap reporting this issue. Some of this research led to memory management best practices, such as deleting old dataframes, then calling the garbage collection function.

Our dataset also taxed nonvolatile disk storage, reporting "device out of space" several times, but research led us to similarly to environment/disk management tools: "%env JOBLIB_TEMP_FOLDER=/tmp". These environment variables were more visible in the CRC systems, but when I inquired "where is the scratch volume" I was told "do not mess with scratch" with no other explanation or answers to my additional questions. Ultimately, we were able to create a virtual environment and corrupt it on the CRC, but were not able to use the CRC for any heaving processing. The local storage on the GCP machines was increased to 200GB on SSD media and after increasing the RAM to the maximum allowed without a CPU quota increase, most of our models completed successfully. We were approved for the CPU quota, but already had 4 VMs running separately with good results by then. The GCP machines saved the day, but still struggled to complete the isolated, optimized tasks.

9 CONCLUSIONS

The dimensionality of the dataset as well as volume was a huge challenge that we overcame partly due to computing power of the cloud platform and the time needed for training. A time series classification problem is very different from a standard forecasting one, as it's not necessary that a previous instance would predict something in the future. In the stated problem, an isolated peak will happen in extremely small time interval.

Using visualization we are able to see the difference in wave forms of the different types of signal types (with and without discharge). PCA and t-SNE provides a way for understanding how the dataset lines up.

Random forests are highly accurate overall, but do not predict faults at all and hence are very poor for the problem at hand. K-Means performs better, it can predict faults in the power line, but also has higher false negatives. LSTM, as it was done without cross validation and with different metrics such as cross entropy would have performed better. It has equal recall to K-means but poorer precision. Retraining with these factors would help get better results.

Overall the project was a huge learning curve and experience with a deep dive into big data and all the tribulations it can bring.

10 FUTURE WORK

With more time to work on this problem, the key methods to try in the future include using SVM (using time series), Recurrent neural network with the correct activation function as well as better metrics instead of just accuracy. We also would be able to scale the test data given from Kaggle and make predictions on this data. Once it's scaled, we can try all the methods we currently have running to predict values along with the one's we will implement later. This way, we can submit to Kaggle and compare our results with others, hopefully doing well and scoring above a 50%.

In terms of computing power, the RAM usage kept doubling every 2-3 days towards the end and using a TPU or parallel GPUs to compute seems the way to go forward. The extremely high dimensionality is a big roadblock and can be addressed by compressing some of the information, for example.

11 RESPONSIBILITIES

- Platform deploy and code management: Chris
- Research: all
- Troubleshooting: all

REFERENCES

[1] Mai Abdelhakim. 2019. Decision Trees and Ensemble Methods. (Mar 2019).
[2] Mai Abdelhakim. 2019. Deep Learning – ConvNet & Intro to RNN. (Mar 2019).
[3] Mai Abdelhakim. 2019. Neural Networks. (Mar 2019).
[4] Mai Abdelhakim. 2019. Unsupervised Learning: Dimensionality Reduction & Clustering. (Apr 2019).
[5] Mina Aminghafari, Nathalie Cheze, and Jean-Michel Poggi. 2006. Multivariate denoising using wavelets and principal component analysis. *Computational Statistics & Data Analysis* 50, 9 (May 2006), 2381–2398. https://doi.org/10.1016/j.csda.2004.12.010
[6] Rakshitha Godahewa. 2018. Time Series Classification Using Feature Extraction. (Nov 2018). https://medium.com/datadriveninvestor/time-series-classification-using-feature-extraction-16209570a22e
[7] Burakh Himmetoglu. 2017. Time series classification with Tensorflow. (Sep 2017). https://burakhimmetoglu.com/2017/08/22/time-series-classification-with-tensorflow/
[8] Maher Maalouf and Theodore B. Trafalis. 2011. Robust weighted kernel logistic regression in imbalanced and rare events data. *Computational Statistics & Data Analysis* 55, 1 (Jan 2011), 168–183. https://doi.org/10.1016/j.csda.2010.06.014

[9] S. Misak, J. Fulnecek, T. Vantuch, T. Burianek, and T. Jezowicz. 2017. A complex classification approach of partial discharges from covered conductors in real environment. *IEEE Transactions on Dielectrics and Electrical Insulation* 24, 2 (Apr 2017), 1097–1104. https://doi.org/10.1109/tdei.2017.006135

[10] Michal Prilepok and Tomas Vantuch. 2017. Partial Discharge Pattern Classification Based on Fuzzy Signatures. *Advances in Intelligent Systems and Computing Proceedings of the Second International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'17)* (Sep 2017), 254–264. https://doi.org/10.1007/978-3-319-68321-8_26

[11] Devin Soni. 2018. Dealing with Imbalanced Classes in Machine Learning. (Feb 2018). https://towardsdatascience.com/dealing-with-imbalanced-classes-in-machine-learning-d43d6fa19d2

[12] Tomas Vantuch. 2018. *Analysis of Time Series Data.* Ph.D. Dissertation. Technical University of Ostrava.

[13] Tomas Vantuch. 2018. VSB Power Line Fault Detection. (Dec 2018). https://www.kaggle.com/c/vsb-power-line-fault-detection