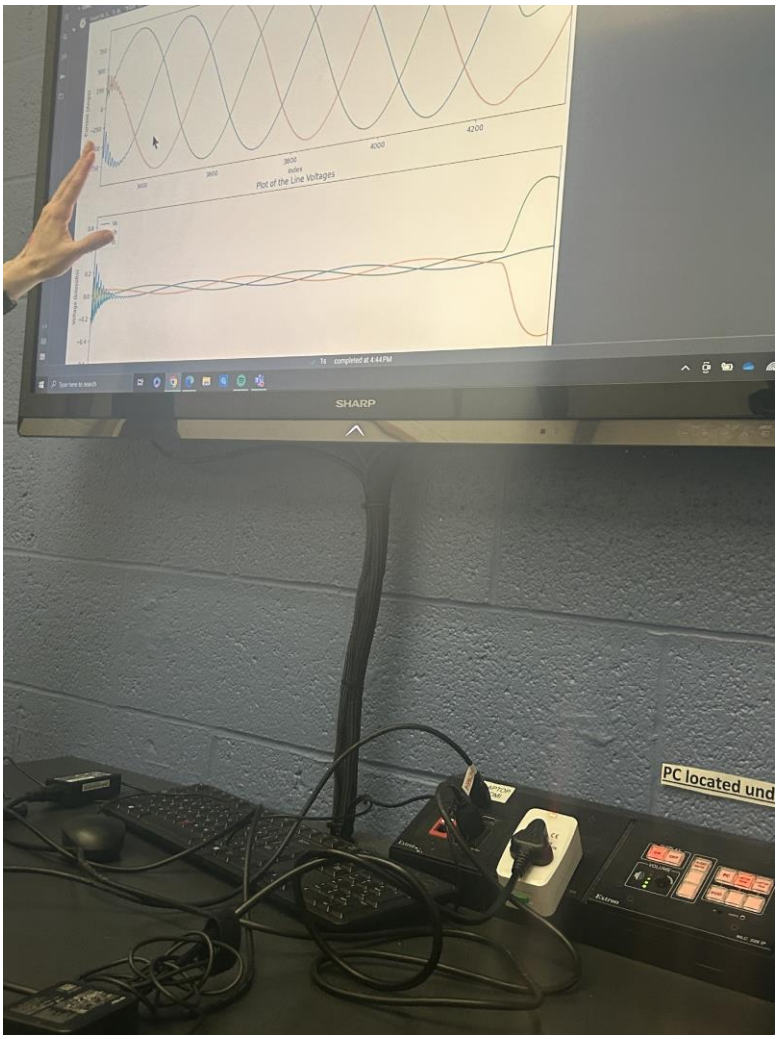


Classification of Electrical Faults

SUBMITTED BY:

CHAD HUCEY
DEEP SUCHAK
ISHA JAIN
TANISH KANDIVLIKAR

Statistical Methods for Data Science | 11-28-2023

Contents

List of Figures:

List of Tables:

Problem Description

The requirement of readily available power has grown exponentially over the years, and the prominent role of a transmission line is to transmit electric power from a source to the distribution network. The electrical power system consists of many complex, dynamic and interacting elements that are always prone to disturbances or electrical faults.

High-capacity electrical generation power plants require fault detection and the operation of protective equipment to remain stable. Faults on electrical transmission lines must be quickly and accurately detected and classified, and then resolved as quickly as possible. An exemplary fault detection system provides a practical, reliable, fast, and secure way of relaying information from the grid to the operators. Furthermore, the application of pattern recognition could help to differentiate between faulty and healthy electrical power systems.

In this project, this team will attempt to address the classification problem detecting faults in a 3-phase electrical transmission system. The dataset used for this project leverages the most rudimentary attributes of an electrical system, voltage and current of the three phases, as a means of detecting faults. The problem can be approached as a binary classification problem as well as a multiclass (6 classes) classification model. However, this team will attempt the multiclass classification problem, which is to accurately detect 6 classes of faults.

Dataset

In order to predict/detect electric faults, the [electric fault detection dataset from Kaggle] [1] was selected. This dataset was obtained by researchers who simulated the performance of a distribution system under various operating conditions using MATLAB, recorded their results and labeled their data accordingly. The configuration of the simulated three phase system can be seen in Figure 1 below.

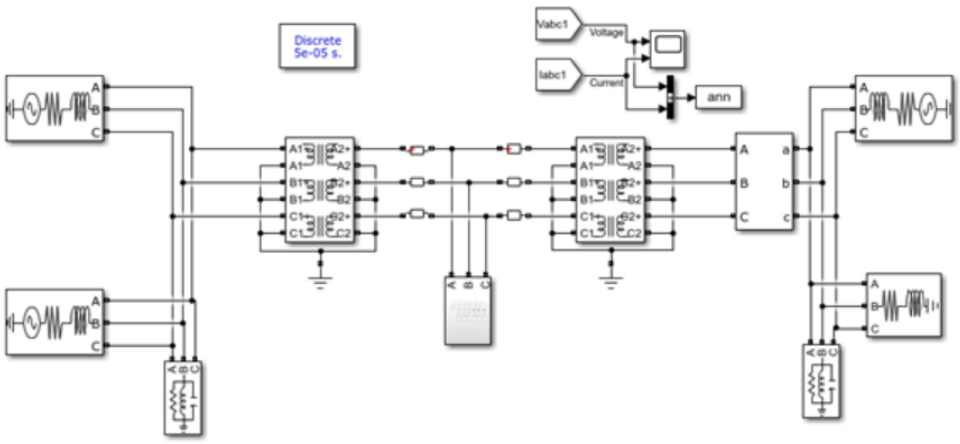


Figure 1: Electrical Distribution System Schematic

The simulated power system consisted of 4 generators that each produced 11 kV. A pair of generators were located at each end of the transmission line. Transformers were also modeled to simulate and study the various faults at the midpoint of the transmission line.

The researchers recorded 7861 samples representing 6 classes of faults. Please find a summary of the dataset below.

Inputs/Features - Ia, Ib, Ic, Va, Vb, Vc
Ia, Ib and Ic represent line currents for line A, B and C respectively and Va, Vb and Vc are line voltages for line A, B and C respectively.

Outputs - [G C B A]
The researchers provided four output columns:

Output Column	Meaning	Output Entry
G	Ground	0 = not fault, 1 = fault
C	Line C	0 = not fault, 1 = fault
B	Line B	0 = not fault, 1 = fault
A	Line A	0 = not fault, 1 = fault

Table 1: Output Columns Description

Combining these outputs into one column gives the different classes of faults represented in the dataset. This concatenation was done as part of the feature engineering process.

The classes are summarized below in the following format [G C B A]:

1. [0 0 0 0] - No Fault
2. [1 0 0 1] - LG fault (Between Phase A and ground)
3. [0 1 1 0] - LL fault (Between Phase B and Phase C)
4. [1 0 1 1] - LLG Fault (Between Phases A, B and ground)
5. [0 1 1 1] - LLL Fault (Between all three phases)
6. [1 1 1 1] - LLLG fault (Three phase symmetrical fault)

Data Exploration

On exploring the data, the team found that there were 7861 rows and 10 columns. Additionally, there were no missing values in this dataset. The team then produced some basic statistics including the mean, standard deviation, and percentiles for each numeric column. Figure 2 shows the basic statistics for this dataset.

	count	mean	std	min	25%	50%	75%	max
G	7861.0	0.432006	0.495387	0.000000	0.000000	0.000000	1.000000	1.000000
C	7861.0	0.411271	0.492095	0.000000	0.000000	0.000000	1.000000	1.000000
B	7861.0	0.555527	0.496939	0.000000	0.000000	1.000000	1.000000	1.000000
A	7861.0	0.571429	0.494903	0.000000	0.000000	1.000000	1.000000	1.000000
Ia	7861.0	13.721194	464.741671	-883.542316	-119.802518	2.042805	227.246377	885.738571
Ib	7861.0	-44.845268	439.269195	-900.526951	-271.845947	5.513317	91.194282	889.868884
Ic	7861.0	34.392394	371.107412	-883.357762	-61.034219	-4.326711	49.115141	901.274261
Va	7861.0	-0.007667	0.289150	-0.620748	-0.130287	-0.005290	0.111627	0.595342
Vb	7861.0	0.001152	0.313437	-0.608016	-0.159507	0.001620	0.153507	0.627875
Vc	7861.0	0.006515	0.307897	-0.612709	-0.215977	0.009281	0.239973	0.600179

Figure 2: Statistical Summary

To better visualize the data, the team plotted the three line currents on a single plot and produced a similar plot for the line voltages as shown in Figure 3 below.

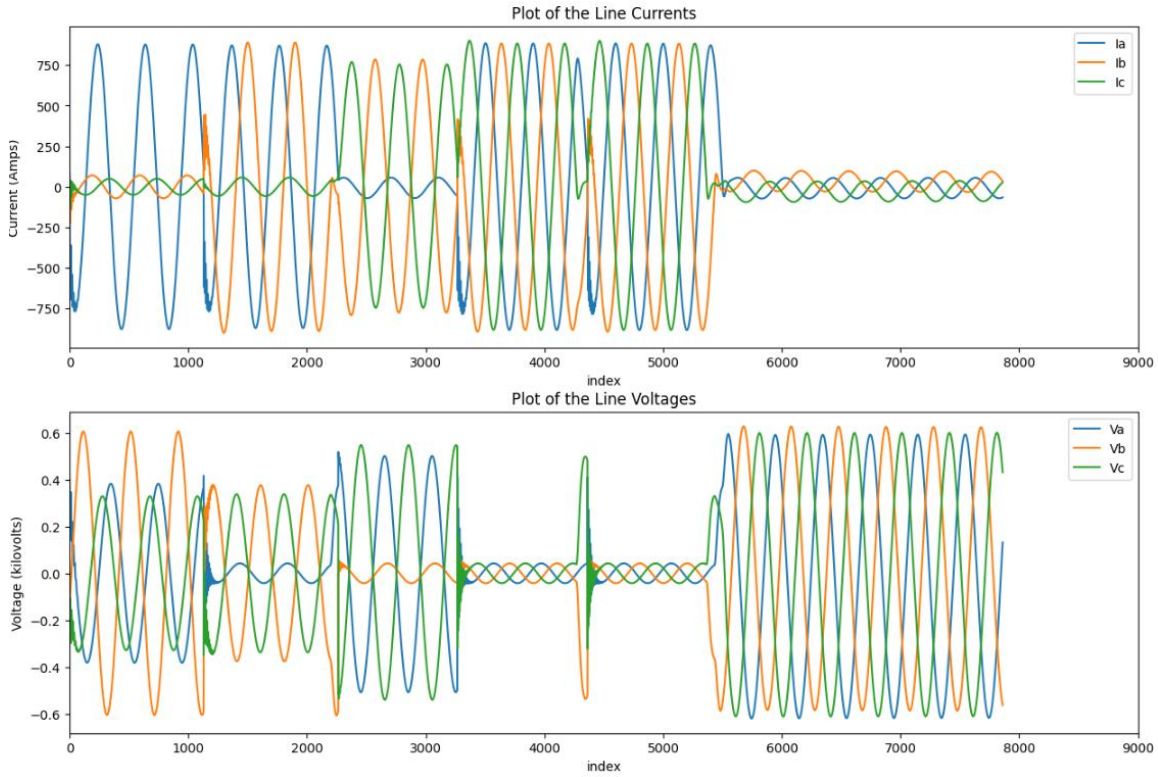


Figure 3: Line Currents and Voltages

Figure 3 shows that the data has a periodic nature, which is typical of an electrical distribution system. With that said, the amplitude of the currents and voltages change over time. Given Figure 3 and the fact that the data was gathered from a MATLAB simulation, it appears that the researchers simulated each fault sequentially. Before training any machine learning models, this data will have to be shuffled to mitigate the risk of a model becoming biased to a certain class of faults.

To better show the relationship between voltage and current, the team also plotted the voltage and current of each line. Figure 4 shows this relationship.

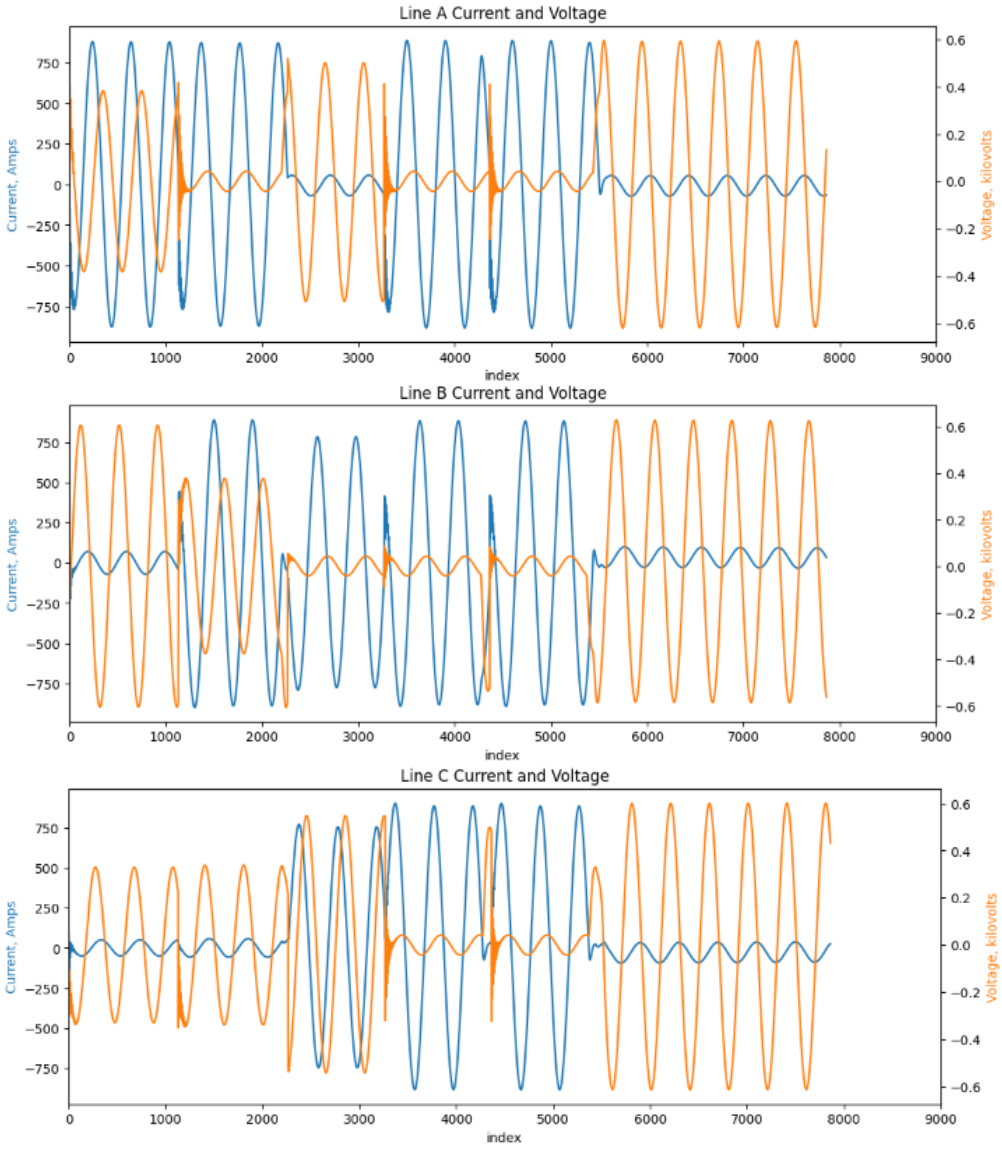


Figure 4: Current/Voltage Relationship

Figure 4 shows the current and voltage on each line. It can be observed that the current and voltage on each line reach their maximum value at different times. This is a typical and expected result as real distribution systems also show this trend.

Note that in Figure 3 and Figure 4, the voltages are measured in kilovolts (kV). Furthermore, the team observed from Figure 2, Figure 3 and Figure 4 that the values of the line currents, Ia, Ib and Ic, were much larger than those of their respective line voltages. To avoid training models that may be biased towards the higher values of the line currents, the team standardized the data prior to training.

The team plotted histograms to get an idea of the distribution of the current and voltage values in the dataset. Figure 5 shows that most of the values were concentrated around the mean while the frequency of other values remained uniform.

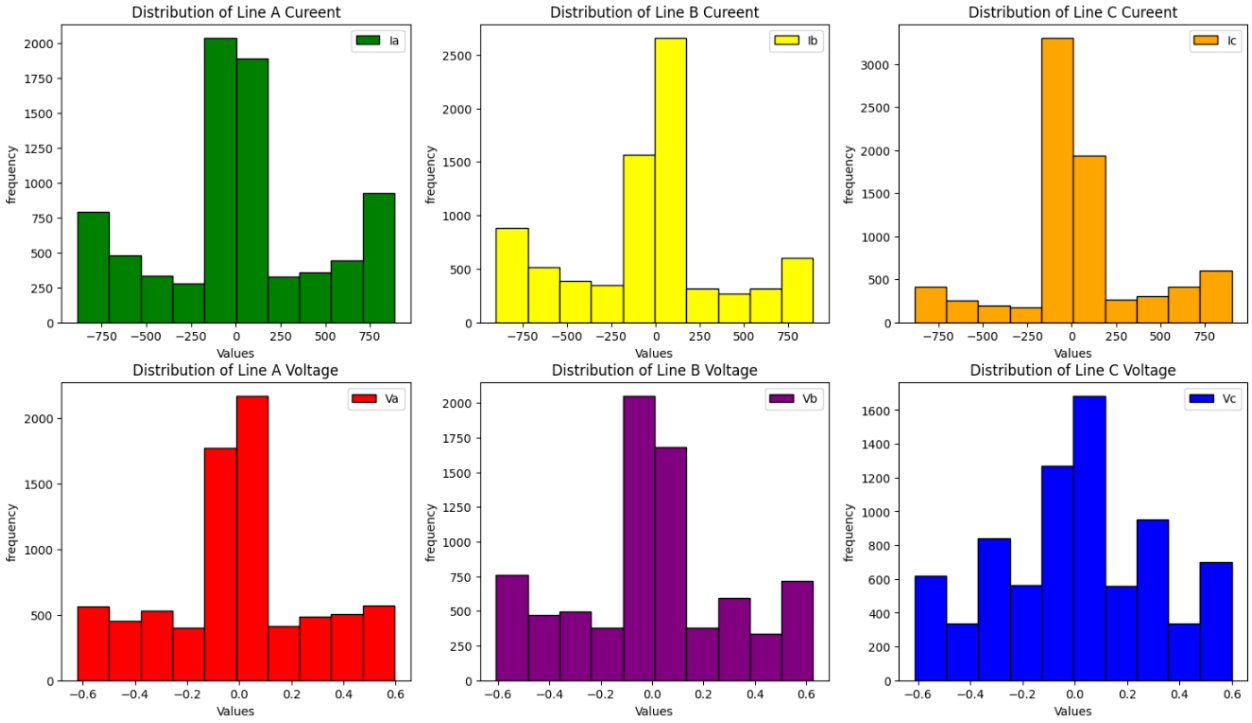


Figure 5: Current and Voltage Values Distribution

One concern that arose was that the output classes could be imbalanced. It is possible to have more samples representing faults than those representing no fault. It is also possible that some faults occur more frequently than others. These imbalances may lead our classifiers to be biased towards a subset of faults. To determine if we were dealing with an imbalanced dataset, we counted the amount of data points that belonged to each class. Table 2 summarizes the counts of each class. From the table below, it can be seen that the classes are relatively balanced.

Fault	Count	% of the Dataset
0000	2365	30.09
1011	1134	14.42
1111	1133	14.41
1001	1129	14.36
0111	1096	13.39
0110	1004	12.77

Table 2: Counts of each Class

The team also plotted a correlation matrix to get an idea of the degree of correlation of all the features. The matrix in Figure 6 is shown below. From Figure 6, it can be seen that the predictors are weakly correlated. This result is also corroborated in Figure 4 in which the voltages and currents on each line reach their peak values at different times.

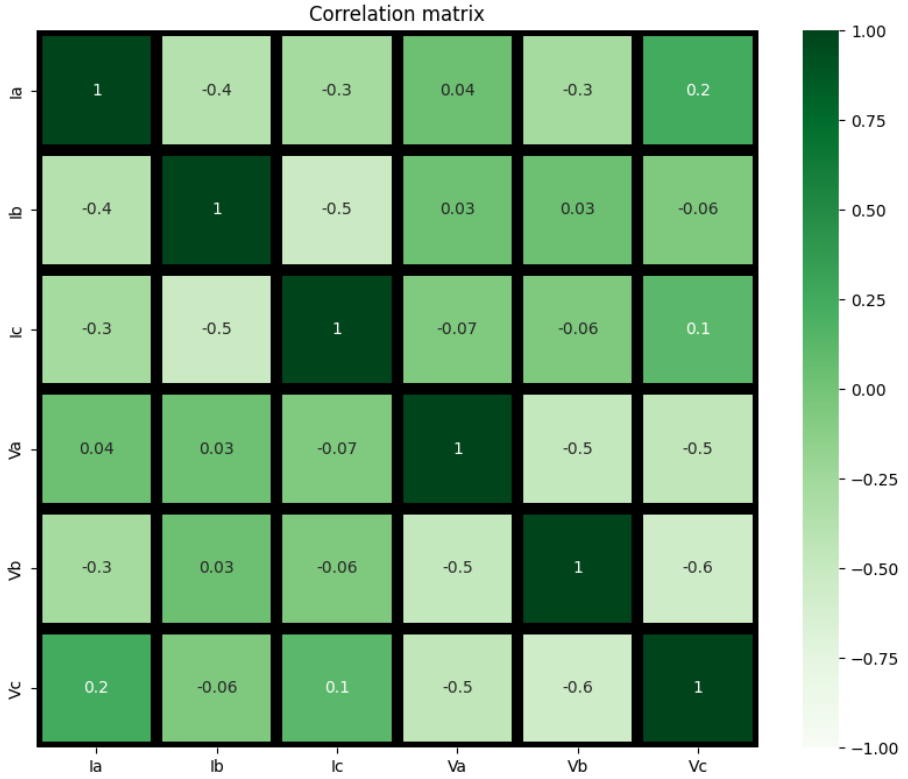


Figure 6: Correlation Matrix

Feature Engineering

Given Figure 6 above, the team will use the line currents and voltages, Ia, Ib, Ic, Va, Vb, Vc, as the predictors and a newly created combined faults column as the target variable. As seen in Figure 3 and Figure 4, the predictor values follow a periodic trend varying between a maximum value (> 0) and minimum value (< 0).

To better prepare the data to be used for machine learning, the team ensured that all predictor data points were greater than or equal to zero. This was done by adding the predictor's absolute value to its original value. This transformation maintained the data's periodic trend, however, the predictor values now ranged from 0 to two times their original maximum value. This transformation was done to potentially reduce the bias of the machine learning models by training these models using only positive values.

From Figure 3 and Figure 4, the values of the line currents were substantially larger than those of the line voltages. To ensure that the trained models do not become biased towards the larger values of current, the team decided to scale the data. This was done using Min/Max scaling in which all values were scaled from 0 to 1. Doing this will improve the performance of distance-based models while retaining the notion of distance. A visual representation of the feature engineering can be seen in Figure 7.

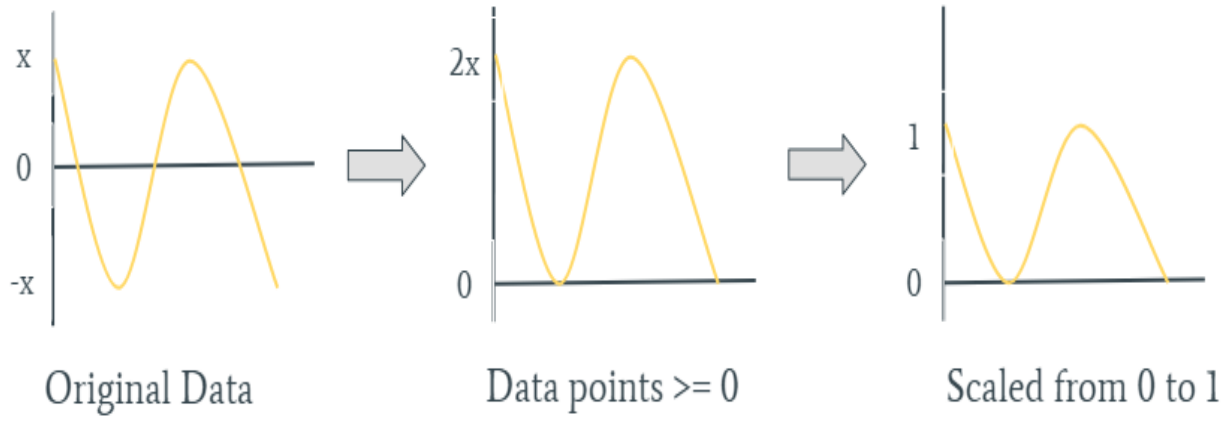


Figure 7: Feature Engineering

Machine Learning Methods

As mentioned in the problem description, this team will build machine learning models that will accurately predict 6 classes of faults in an electrical distribution system. To facilitate the development of these models, the team shuffled the data, created a training set to train the various models and a test set to evaluate the results of each model.

All models will be evaluated based on how accurately they can predict an assigned label. The team chose to use accuracy as the evaluation metric since our goal is to maximize the number of times that our models accurately predict a fault.

Since it was unclear which model would produce the best results, the team decided to train five baseline models. These models were created to assess the potential performance of different algorithms. Furthermore, the team decided to only change one parameter per baseline model, and to leave all other parameters set to their default values. Consequently, the team trained the five baseline models below:

Baseline Model	Parameter
Logistic Regression	"multi_class" parameter set to "multinomial"
K nearest neighbors	"n_neighbors" set to 5
Decision Tree	"max_depth" of 10
Random Forest	"n_estimators" set to 100
Support Vector Machine	"kernel" set to "rbf"

Table 3: Baseline Models

Logistic Regression

Logistic regression is a simple and fast algorithm that performs well when the decision boundary between classes is linear. We wanted to test if the faults can be separated by linear boundaries in the feature space, Logistic regression is a reasonable first model to try as it can serve as a baseline to measure the performance of more complex models.

K-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning algorithm that is intuitive and simple—faults are classified based on how similar they are to observations in the training set. Given that electrical faults may have distinct signatures in terms of current and voltage profiles, KNN could effectively classify faults by looking at 'nearby' instances in the feature space, if similar types of faults have similar electrical characteristics.

Decision Tree

Decision trees are easy to interpret and can handle both numerical and categorical data. They perform feature selection implicitly, which can be beneficial if some features are more indicative of faults than others. If there are specific thresholds of currents and voltages that are indicative of different fault types, a decision tree could use these thresholds to make decisions, making it a good fit for data with clear rule-based patterns.

Random Forest

Random forests improve upon decision trees by creating an ensemble of trees with controlled variance, which generally leads to better performance and less overfitting. Because electrical fault detection can be complex and might not be well-modeled by a single decision tree, random forests can capture a wider range of fault signatures by combining the learning power of multiple trees.

Support Vector Machine (SVM)

SVMs can model complex, non-linear relationships using the kernel trick, and they focus on maximizing the margin, which can lead to better generalization. Fault detection often involves complex boundaries between different fault types. The ability of SVMs to use different kernels allows them to model the complex boundaries that could occur in the feature space of electrical signals.

Once the baseline models have been trained, the team will tune the parameters of the four best performing models using GridSearch and 10-fold cross validation to find the models with the best hyperparameters and noted the test accuracies of each of them.

Results

The Logistic Regression model performed poorly, only achieving a test accuracy of 65%. The remainder of this section highlights the results from the other baseline models as well as the results from the parameter tuning.

K-Nearest Neighbors (KNN)

The initial model trained with k=5 gave test accuracy of around 79.34%. On implementing gridsearchCV with 10 folds and varying k from 2 to 12, we obtained the following plot for cross validation error.

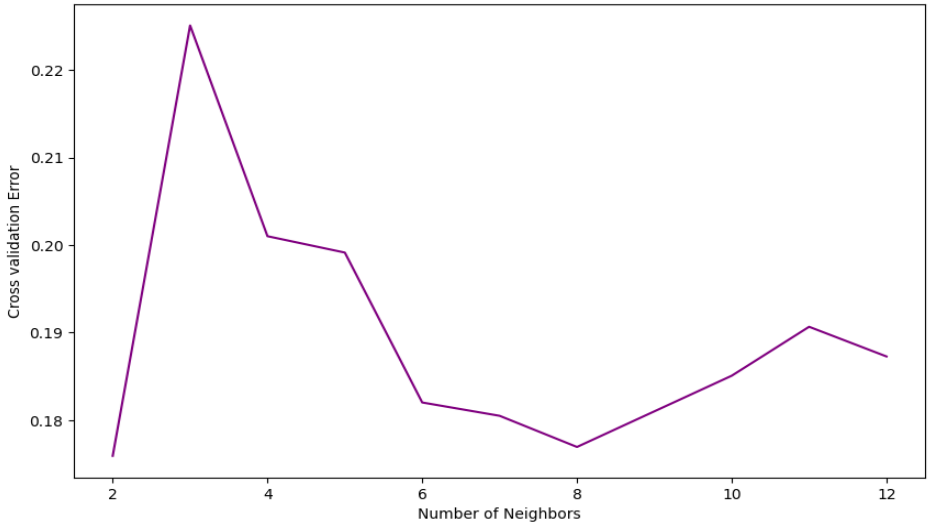


Figure 8: KNN Cross Validation Error

From the above, k=2 seems to give the lowest cross validation error; however, this model may be very sensitive since k is small. k=8 seems to provide comparable accuracy, so the test accuracy and confusion matrix were evaluated for both models (k=2 and k=8).

k=2 gave a test accuracy of around 83.57% and the confusion matrix is shown in Figure 9.

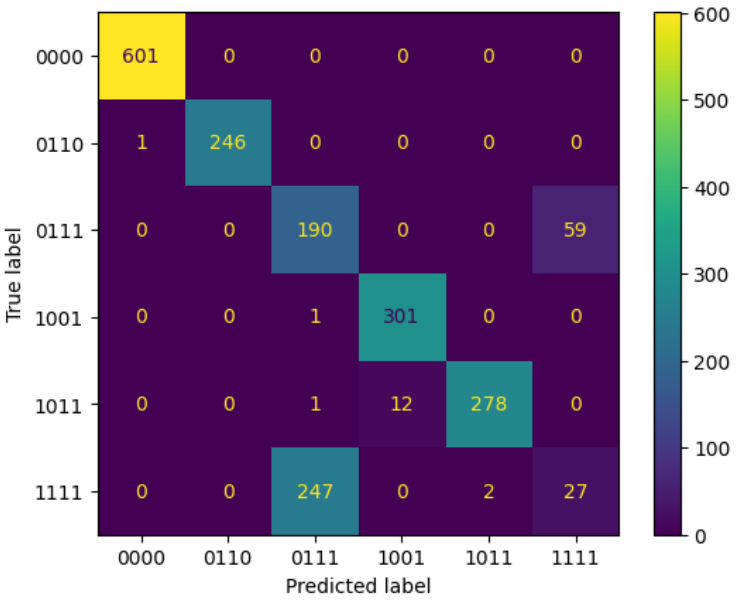


Figure 9: KNN (k=2) Confusion Matrix

k=8 gave a test accuracy of around 82.25% and the confusion matrix is shown in Figure 10.

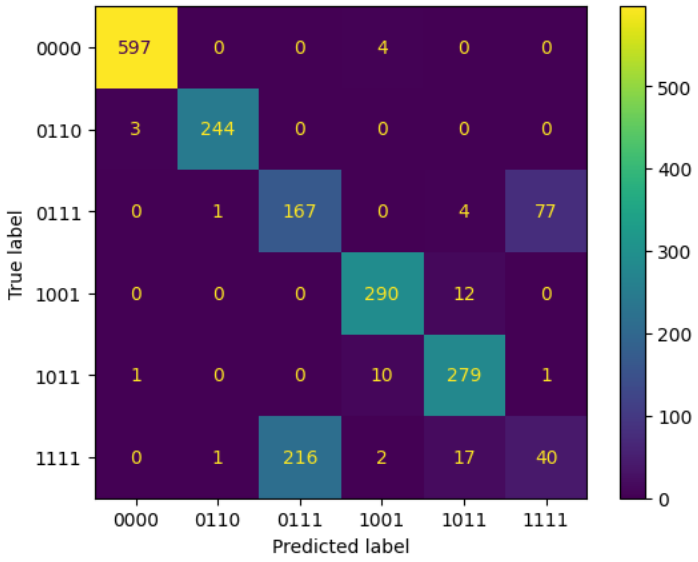


Figure 10: KNN (k=8) Confusion Matrix

Decision Tree

The baseline decision tree model was trained with the "max_depth" parameter set to 10 and holding all other parameters at their default values. The test accuracy achieved under these conditions was 83.33%.

On tuning the "max_depth" parameter from 2 to 20, the following cross validation error was observed.

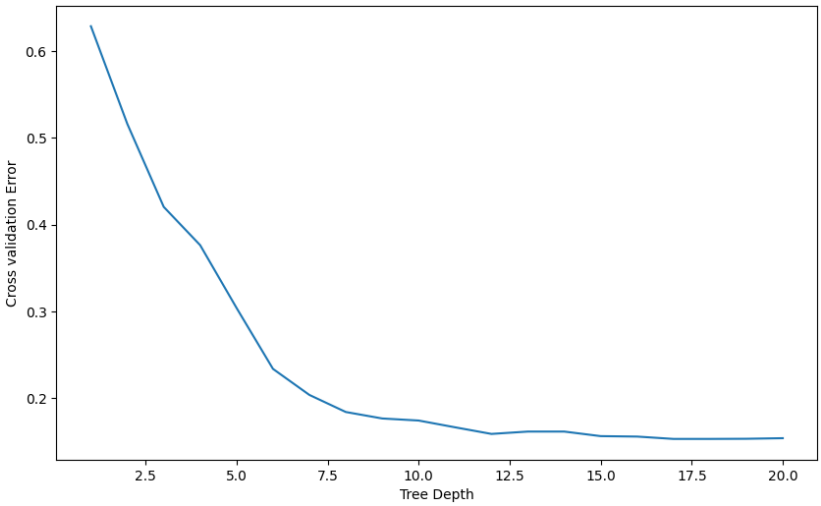


Figure 11: Decision Tree Cross Validation Error

The lowest error occurred at a depth of 17, and after tuning, the test accuracy achieved was 85.69%. The accompanying confusion matrix is shown below.

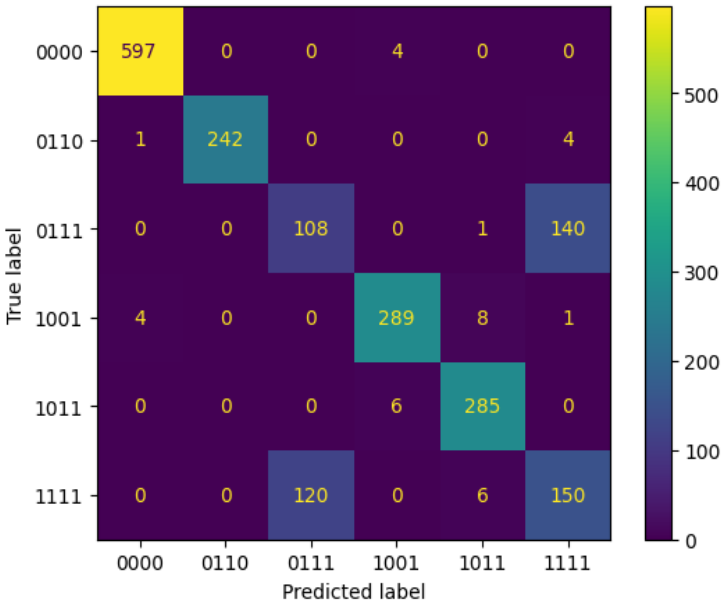


Figure 12: Decision Tree Confusion Matrix

Random Forest Classifier

The baseline random forest classifier model was trained with its pre-set hyperparameters, where n_estimators equal to 100. The test accuracy achieved under these conditions was 87.23%.

On tuning in the hyperparameters using GridSearchCV and by checking the cross validation error score, it was observed that with n_estimators equal to 400 gives us not only a boost in accuracy but also yields a lower cross-validation error score.

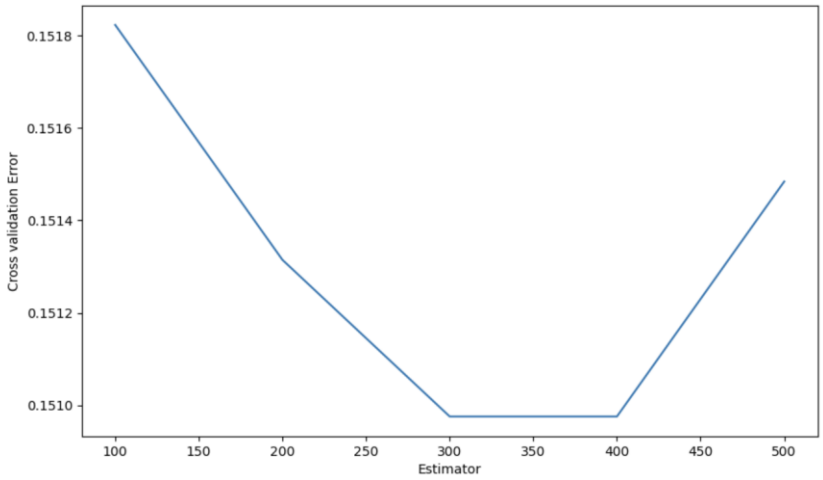


Figure 13: Random Forest Cross Validation Error

The lowest error occurred at n_estimators = 400, and after tuning, the test accuracy achieved was 87.33%. The accompanying confusion matrix is shown below.

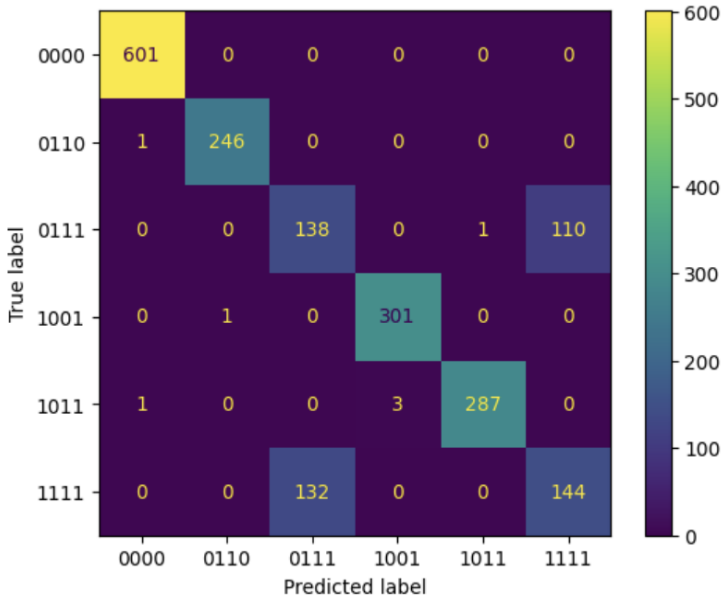


Figure 14: Random Forest Confusion Matrix

Support Vector Machine

The baseline support vector classifier model was trained with the "kernel" set to "rbf" and its other hyperparameters set to default. Under those conditions, the test accuracy came out to be 76.14%.

Hyperparameter tuning was done using GridSearchCV with hyperparameter combinations of C: [0.1, 1, 10, 100], gamma: [1, 0.1, 0.01, 0.001] and kernel = rbf. The results of these experiments can be seen below.

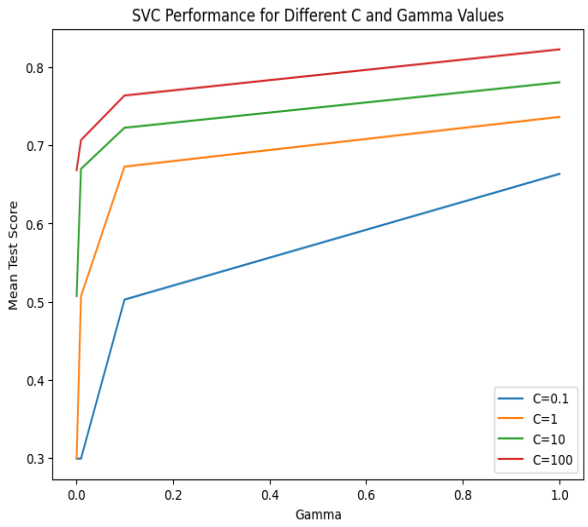


Figure 15: SVM Parameter Tuning Results

The test accuracy improved to 83.26% with best hyperparameters of C = 100, gamma = 1, kernel = rbf. The accompanying confusion matrix is shown below:

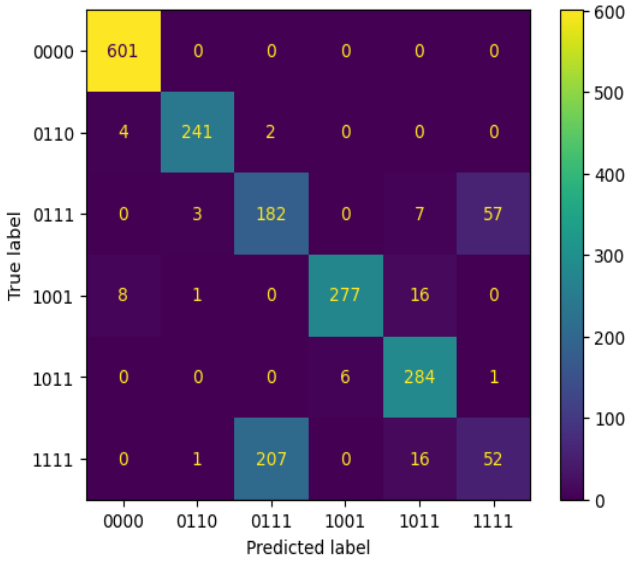


Figure 16: SVM Confusion Matrix

A summary of the 4 models tuned with GridsearchCV is provided in Table 1 below.

Model	Test Accuracy	Best Parameters (
Logistic Regression	65%	multi_class = “multinominal”
k-Nearest Neighbors (KNN)	82.25%	n_neighbors = 8*
Decision Tree	85.69%	max_depth = 17
Random Forest Classifier	87.33%	n_estimators = 400
Support Vector Machine (SVM)	83.26%	c = 100, gamma = 1, kernel = rbf

Table 4: Hyperparameter Tuning Summary

*Not considering k=2 as the best model as we believe it would lead to overfitting

Challenges

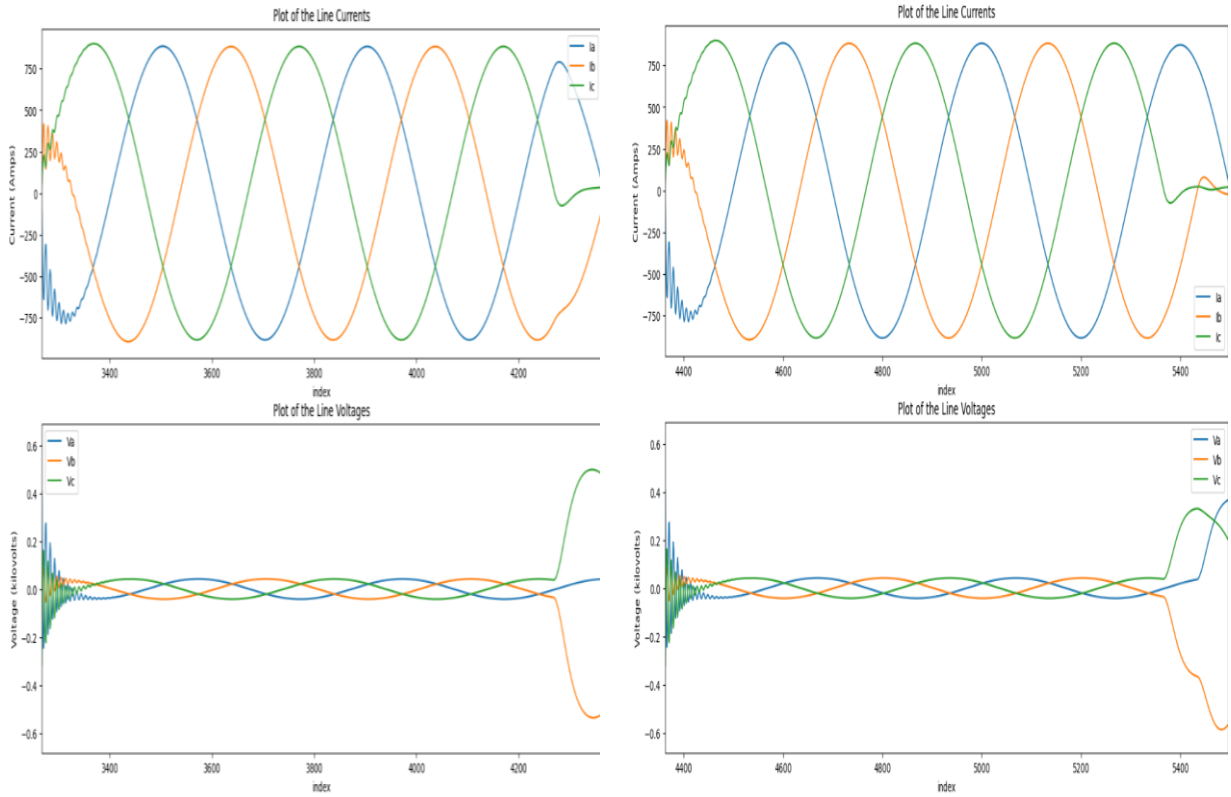
There were several challenges faced while completing this project. The first challenge was understanding the data. Understanding this dataset required knowledge of electrical engineering and distribution systems. Consequently, the first task was to find a way to best represent the data. With knowledge of electrical engineering, the current and voltage should be periodic so Figure 3 and Figure 4 were produced to ensure that our dataset followed this trend.

With the current and voltage profiles plotted, the next task was to understand what a “healthy” and “faulty” profile looked like. As each fault was recorded sequentially, the team could filter all data points related to a specific fault and plot those filtered data points.

The next set of challenges occurred while training the models. As mentioned in the Results section, Logistic Regression performed poorly. This model was originally trained with a standardized version of the original predictors, which led to a test accuracy of roughly 25%. This outcome likely occurred due to a high bias from training with both positive and negative values. The model was retrained with scaled values between 0 and 1, mentioned in the Feature Engineering section, which led to a training accuracy of 65%. The team also applied a cosine transformation to the predictors; however, this transformation reduced the test accuracy to 63% possibly due to the introduction of more bias as the values of cosine range from -1 to 1. It was evident that our implementation of Logistic Regression was too simple for this application, and a more flexible implementation with

more sophisticated feature engineering would be needed to further improve the performance of a Logistic Regression model.

All other models outperformed Logistic Regression due to their higher flexibility, however, a consistent trend occurred. Even though the other models achieved accuracies above 80%, they all had difficulty differentiating between class 0111 (fault between line A, B and C) and 1111 (fault between line A, B, C and Ground). Consider Figure 12 in which the tuned Decision Tree model predicted class 0111 228 times but the true labels for 120 of those predictions were class 1111. It can also be seen that class 1111 was predicted 290 times but the true labels for 140 of those predictions were for class 0111. This outcome is problematic since the models cannot accurately predict those two classes; however, these misclassifications can potentially be explained by the data.



Class 0111 Class 1111

Figure 17: Class 0111 and 1111 visualization

Figure 17 shows the current and voltage profiles for class 0111 on the left and for class 1111 on the right. Both profiles are very similar, which means that the models are learning similar trends for both classes. As a result, if a data point shows trends similar to that of Figure 17, the models are unsure whether to classify that data point as class 0111 or 1111. To address this issue, more training samples will be needed to better identify any hidden trends of these classes. Additionally, more flexible models like neural networks could also be used.

Conclusion

In conclusion, our exploration of various classification algorithms for electrical fault detection has provided valuable insights into their strengths and limitations. Through rigorous experimentation, including hyperparameter tuning and cross-validation, we observed distinct performances across different models. Logistic Regression, while simple and effective for linear relationships, may fall short in capturing the non-linear intricacies of electrical fault data. K-Nearest Neighbors demonstrated adaptability to complex relationships but showed sensitivity to noisy features. Decision Trees, though capable of capturing complex patterns, exhibited a tendency to overfit. Support Vector Machines showcased versatility but may face challenges with larger datasets. Notably, Random Forest emerged as the most robust performer, effectively balancing complexity and generalization. Its ensemble approach proved instrumental in capturing the intricate dependencies inherent in electrical fault detection.

For future works, several avenues could be explored to enhance the performance of the model. Dimensionality reduction techniques, such as Principal Component Analysis (PCA), could be applied to streamline the feature space and potentially improve model efficiency. Additionally, acquiring more data, especially in scenarios where certain fault classes are underrepresented, could further enrich the model's learning capacity. Exploring advanced ensemble techniques or deep learning approaches tailored to the unique characteristics of electrical fault data may also offer avenues for improvement. The combination of these strategies could contribute to a more robust and accurate fault detection system, providing enhanced reliability in real-world applications. Moving forward, these considerations will guide our efforts to continually refine and optimize our classification model for electrical fault detection scenarios.

Reference

1. Prakash, E Sathya. “Electrical Fault Detection and Classification.” *Kaggle*, 22 May 2021, www.kaggle.com/datasets/esathyaprakash/electrical-fault-detection-and-classification/?select=classData.csv.