ELECTRIC FAULT DETECTION AND CLASSIFICATION

TEAM MEMBERS:

I.PRUDVI SATYA SRI RAM(521141)

M.THARAK PRABHU(521157)

P.NAGENDRA(521216)

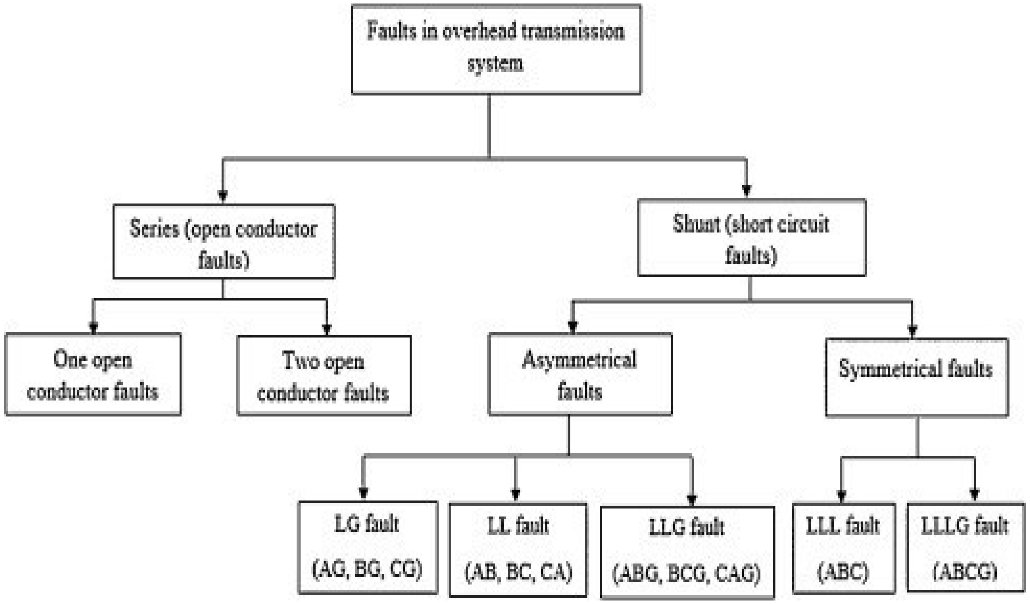
SUPERVISOR: DR. SRI PHANI KRISHNA KARRI

TABLE OF CONTENTS

PROJECT OVERVIEW

INTRODUCTION

This project aims to develop a sophisticated machine learning model to accurately detect and classify various types of electrical faults in transmission lines. By leveraging advanced algorithms and comprehensive data analysis, the goal is to enhance the efficiency of power distribution and significantly reduce risks associated with electrical faults, such as power outages and wildfires.



DATA UNDERSTANDING AND PREPARATION

In this section, we delve into the dataset at hand, understanding its characteristics and preparing it for subsequent analysis and model development

Current in Line A (Ia)

Current in Line B (Ib)

Current in Line C (Ic)

Voltage in Line A (Va)

Voltage in Line B (Vb)

Voltage in Line C (Vc)

The dataset also includes labels indicating different types of faults in a binary format, corresponding to various fault conditions in the transmission line.

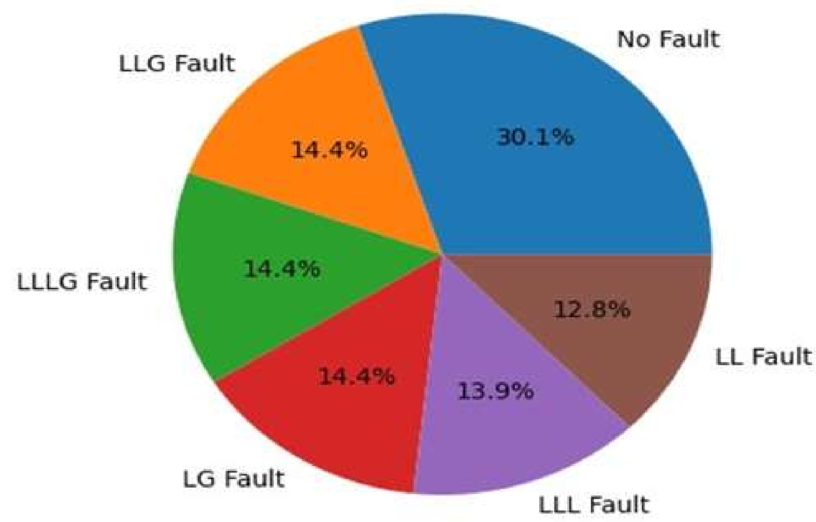
	G	C	B	A	Ia	Ib	Ic	Va	Vb	Vc
0	1	0	0	1	-151.291812	-9.677452	85.800162	0.400750	-0.132935	-0.267815
1	1	0	0	1	-336.186183	-76.283262	18.328897	0.312732	-0.123633	-0.189099
2	1	0	0	1	-502.891583	-174.648023	-80.924663	0.265728	-0.114301	-0.151428
3	1	0	0	1	-593.941905	-217.703359	-124.891924	0.235511	-0.104940	-0.130570
4	1	0	0	1	-643.663617	-224.159427	-132.282815	0.209537	-0.095554	-0.113983
...	...	...	...	...	...	...	...	...	...	...
7856	0	0	0	0	-66.237921	38.457041	24.912239	0.094421	-0.552019	0.457598
7857	0	0	0	0	-65.849493	37.465454	25.515675	0.103778	-0.555186	0.451407
7858	0	0	0	0	-65.446698	36.472055	26.106554	0.113107	-0.558211	0.445104
7859	0	0	0	0	-65.029633	35.477088	26.684731	0.122404	-0.561094	0.438690
7860	0	0	0	0	-64.598401	34.480799	27.250065	0.131669	-0.563835	0.432166

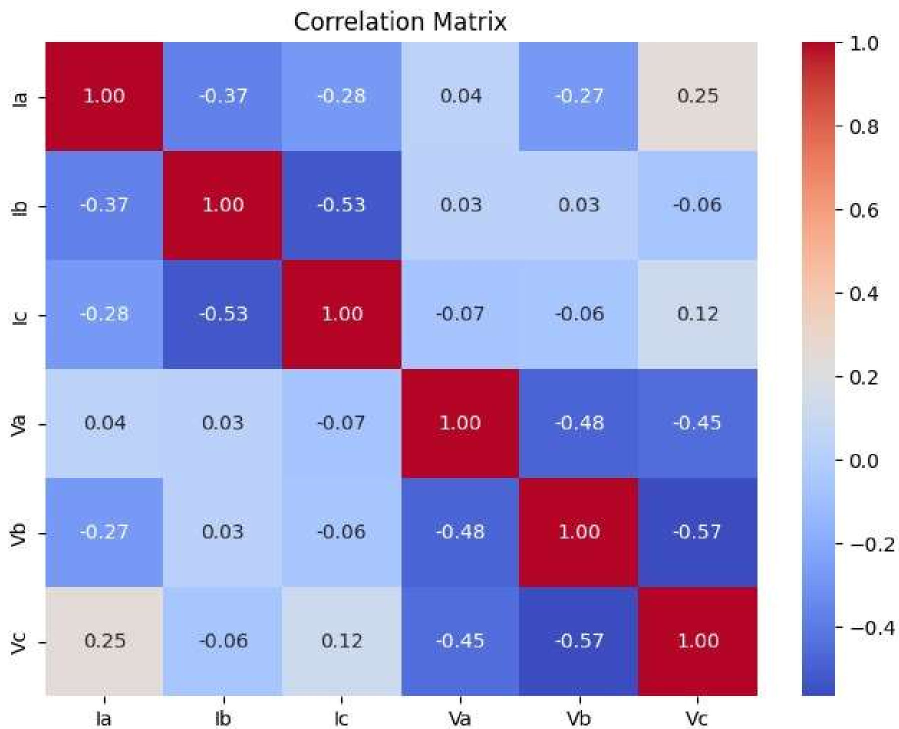
7861 rows × 10 columns

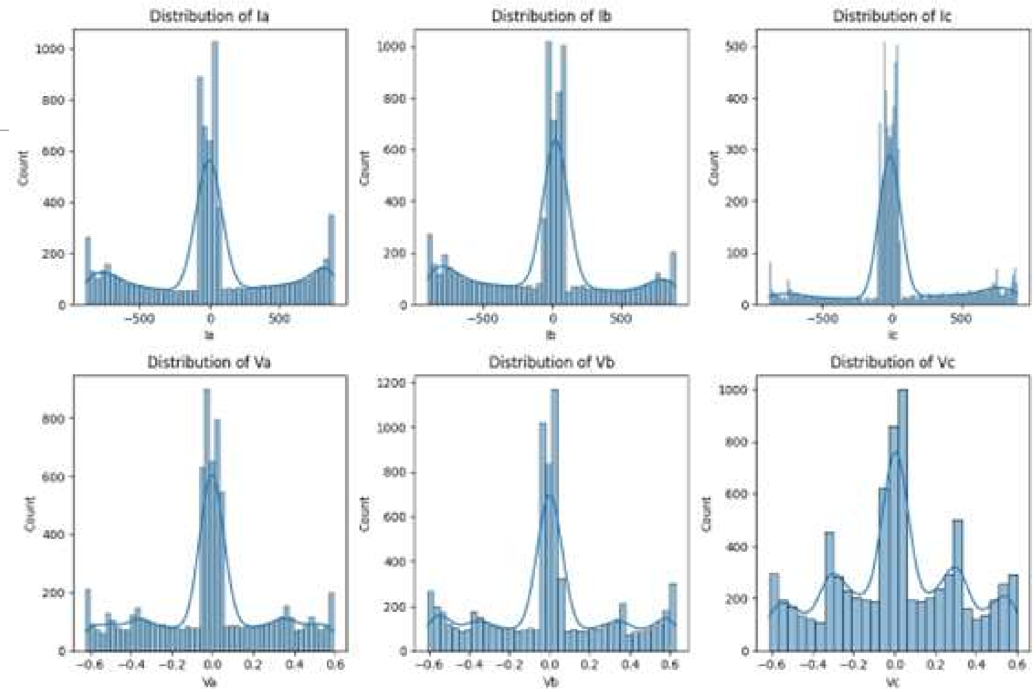
'0111': Three-Phase
'0110': Line-to-Line with Ground BC
'1001': Line-to-Line AB
'1111': Three-Phase with Ground
'1010': Line-to-Line with Ground AB
'0000': No Fault

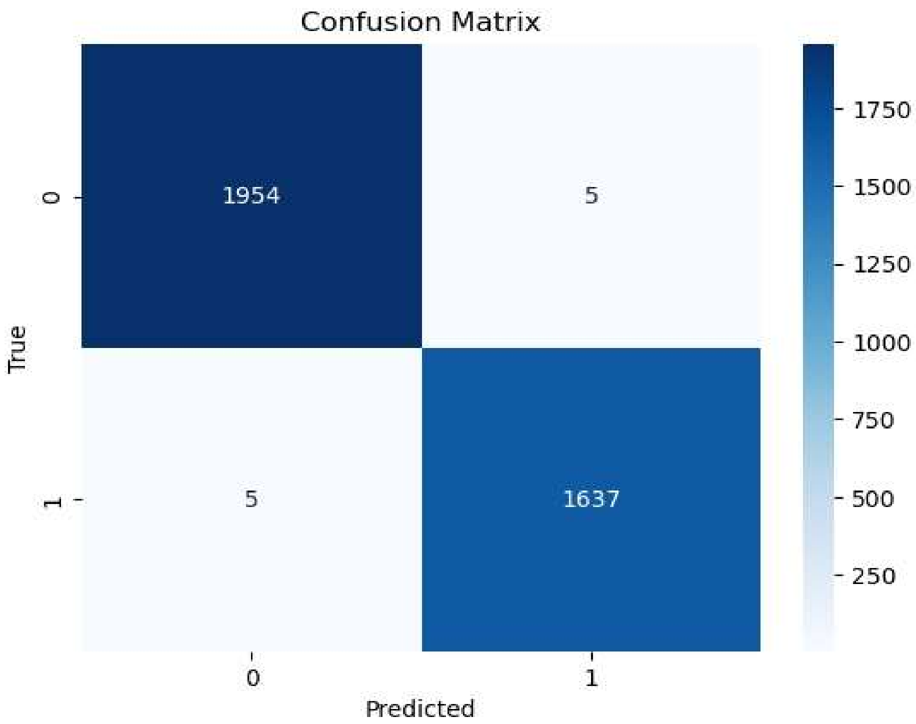
EXPLORATORY DATA ANALYSIS

- No Fault: 2365 occurrences
- Line A Line B to Ground Fault: 1134 occurrences
- Three-Phase with Ground: 1133 occurrences
- Line-to-Line AB: 1129 occurrences
- Three-Phase: 1096 occurrences
- Line-to-Line with Ground BC: 1004 occurrences

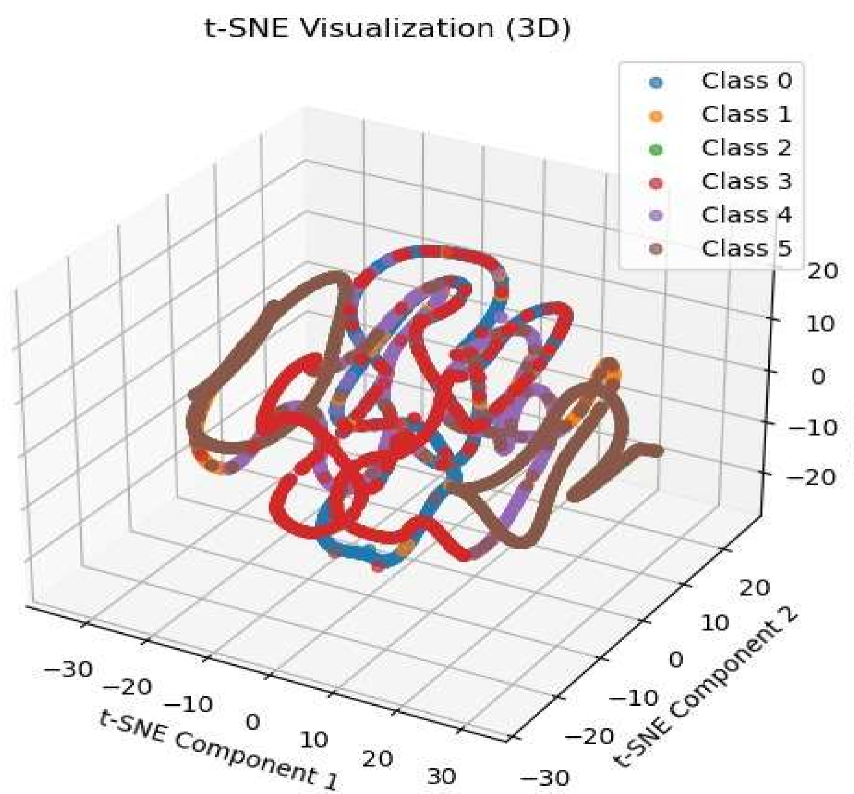



Correlation Matrix
1.00
-0.37
-0.28
0.04
-0.27
0.25
-0.37
1.00
-0.53
0.03
0.03
-0.06
-0.28
-0.53
1.00
-0.07
-0.06
0.12
0.04
0.03
-0.07
1.00
-0.48
-0.45
-0.27
0.03
-0.06
-0.48
1.00
-0.57
0.25
-0.06
0.12
-0.45
-0.57
1.00
Ia
Ib
Ic
Va
Vb
Vc
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Distribution of Ia
Distribution of Ib
Distribution of Ic
Distribution of Va
Distribution of Vb
Distribution of Vc
Count



ACCURACY OF BINARY CLASSIFICATION-99.7%




```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.manifold import TSNE

def plot_tsne_3d(X, y):
    # Apply t-SNE
    tsne = TSNE(n_components=3, random_state=42)
    X_tsne = tsne.fit_transform(X)

    # Plot t-SNE embeddings
    fig = plt.figure(figsize=(8, 6))
    ax = fig.add_subplot(111, projection='3d')
    for label in np.unique(y):
        ax.scatter(X_tsne[y == label, 0], X_tsne[y == label, 1], X_tsne[y == label, 2], label=f"Class {label}", alpha=0.7)
    ax.set_title('t-SNE Visualization (3D)')
    ax.set_xlabel('t-SNE Component 1')
    ax.set_ylabel('t-SNE Component 2')
    ax.set_zlabel('t-SNE Component 3')
    plt.legend()
    plt.show()
plot_tsne_3d(X, y)
```

MODEL TRAINING & EVALUATION

```python
def train_and_evaluate_model(model, model_name, X_train, y_train):
    # Define the scoring metrics for multi-class classification
    scoring = {
        'accuracy': make_scorer(accuracy_score),
    }

    # Perform cross-validation using StratifiedKFold
    skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    scores = cross_validate(model, X_train, y_train, cv=skf, scoring=scoring)

    # Store the cross-validation metrics
    cv_metrics['Model'].append(model_name)
    cv_metrics['Accuracy'].append(scores['test_accuracy'].mean())
    print(f"{model_name}: Cross-validation metrics calculated")

    # Fit the model on the entire training set
    model.fit(X_train, y_train)
    return model

# Define a function to evaluate the model on the test set and store the metrics
def evaluate_on_test_set(model, model_name, X_test, y_test):
    y_pred = model.predict(X_test)
    test_metrics['Model'].append(model_name)
    test_metrics['Accuracy'].append(accuracy_score(y_test, y_pred))
    print(f"{model_name}: Test metrics calculated")
```

```python
# Train and evaluate each algorithm
models = [
    (LogisticRegression(random_state=42, max_iter=1000), "Logistic Regression"),
    (SVC(random_state=42), "Support Vector Machines"),
    (KNeighborsClassifier(), "K-Nearest Neighbors"),
    (DecisionTreeClassifier(random_state=42), "Decision Trees"),
    (RandomForestClassifier(random_state=42), "Random Forest"),
    (GradientBoostingClassifier(random_state=42), "Gradient Boosting"),
    (MLPClassifier(random_state=42, max_iter=1000), "Neural Networks"),
    (GaussianNB(), "Naive Bayes"),
    (AdaBoostClassifier(random_state=42), "AdaBoost"),
    (XGBClassifier(random_state=42), "XGBoost"),
    (LGBMClassifier(random_state=42), "LightGBM"),
    (CatBoostClassifier(random_state=42, verbose=0), "CatBoost")
]

# Train and evaluate each model
for model, model_name in models:
    fitted_model = train_and_evaluate_model(model, model_name, X_train, y_train)
    evaluate_on_test_set(fitted_model, model_name, X_test, y_test)
```

ANALYSING BEST PERFORMANCE MODELS SO FAR

Cross-validation metrics:

MODEL ACCURACY

- K-Nearest Neighbors: 0.8296753
- Decision Trees : 0.8633924
- Random Forest : 0.859733
- XGBoost : 0.830632

Test Metrics:

MODEL ACCURACY

- K-Nearest Neighbors: 0.804196
- Decision Trees : 0.886205
- Random Forest : 0.879212
- XGBoost : 0.813096

USING DOMAIN KNOWLEDGE TO FURTHER IMPROVE THE MODEL

ADVANCED FEATURE ENGINEERING

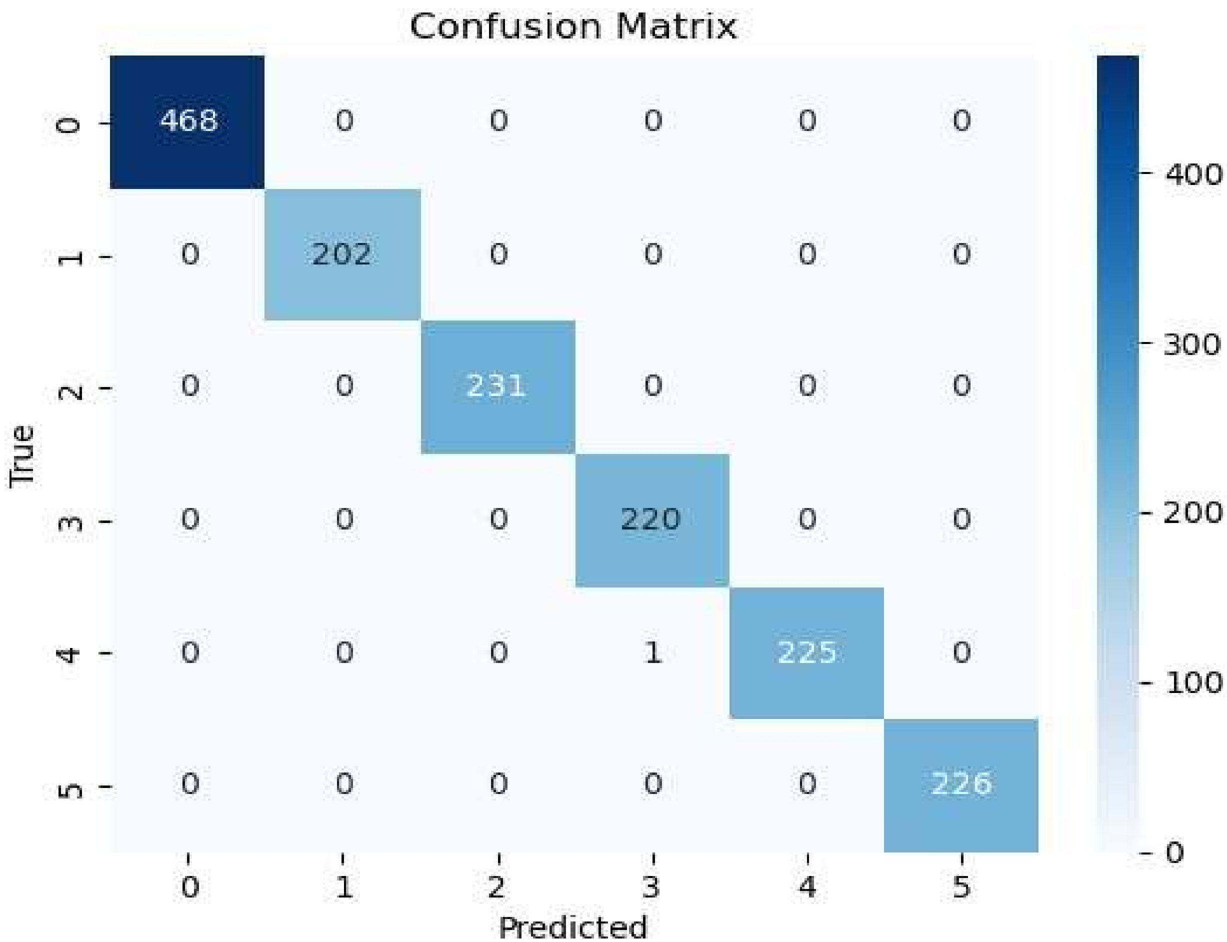
```python
poly_features_df['ZeroSeqCurrent'] = (poly_features_df['Ia'] + poly_features_df['Ib'] + poly_features_df['Ic']) / 3
poly_features_df['ZeroSeqVoltage'] = (poly_features_df['Va'] + poly_features_df['Vb'] + poly_features_df['Vc']) / 3

# Phase Angle Differences (approximated by product of current and voltage)
poly_features_df['PhaseAngleDiffI'] = poly_features_df['Ia'] * poly_features_df['Ib'] * poly_features_df['Ic']
poly_features_df['PhaseAngleDiffV'] = poly_features_df['Va'] * poly_features_df['Vb'] * poly_features_df['Vc']

# Voltage and Current Ratios
poly_features_df['V_I_Ratio_A'] = poly_features_df['Va'] / poly_features_df['Ia']
poly_features_df['V_I_Ratio_B'] = poly_features_df['Vb'] / poly_features_df['Ib']
poly_features_df['V_I_Ratio_C'] = poly_features_df['Vc'] / poly_features_df['Ic']
```

Cross-validation metrics:

	MODEL ACCURACY
K-Nearest Neighbors	0.839853
Decision Trees	0.994911
Random Forest	0.998251
XGBoost	0.998251

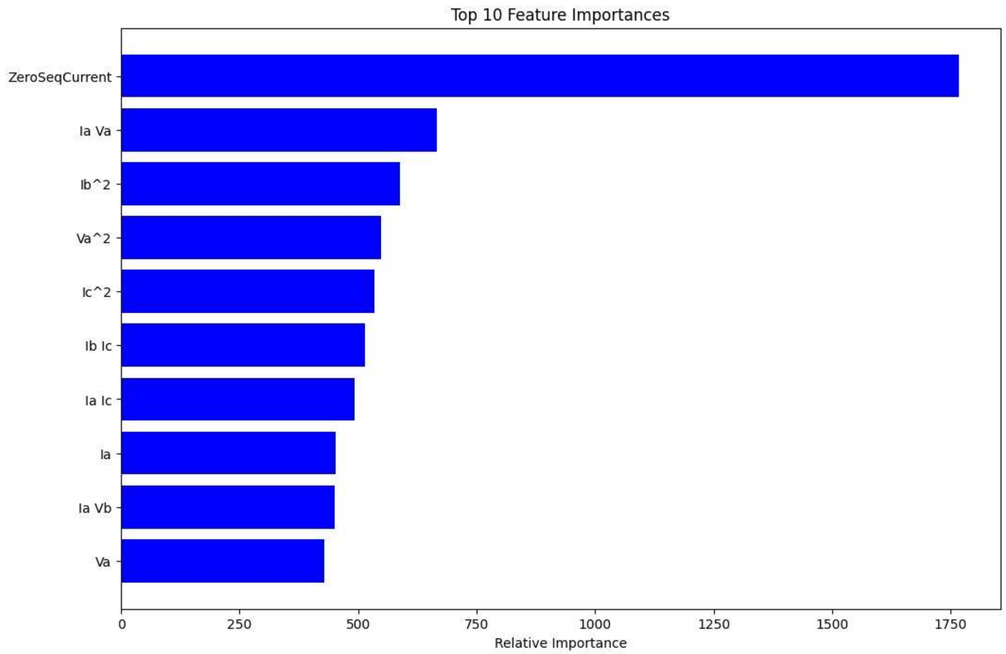
Test Metrics:

	MODEL ACCURACY
K-Nearest Neighbors	0.811825
Decision Trees	0.996186
Random Forest	0.999364
XGBoost	0.999364


Confusion Matrix
468
202
231
220
225
226
1
0
True
Predicted
0
1
2
3
4
5
400
300
200
100
0

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	468
1	1.00	1.00	1.00	202
2	1.00	1.00	1.00	231
3	1.00	1.00	1.00	220
4	1.00	1.00	1.00	226
5	1.00	1.00	1.00	226
accuracy			1.00	1573
macro avg	1.00	1.00	1.00	1573
weighted avg	1.00	1.00	1.00	1573



CONCLUSION

- This project's journey to develop a real-time electrical fault detection and classification system culminates in a robust suite of ensemble models that have far exceeded performance expectations. The Random Forest, XGBoost,K-Nearest Neighbours,Random Forest,Decision Trees, in particular, have shown stellar accuracy, validating the profound impact that machine learning can have on real-time monitoring and predictive maintenance in the electrical domain.
- A key to our success was the strategic feature engineering process, which was significantly enhanced by domain-specific insights. The incorporation of sophisticated features like the zero sequence current and various interactive terms between currents and voltages was instrumental in differentiating between fault types. This is reflected in the top feature importances, which signify the model's dependency on a nuanced combination of engineered and original attributes—a testament to the intricate nature of electrical fault dynamics.