# CHAPTER FOUR

# RESULT AND DISCUSSION

## 4.0 INTRODUCTION

This section presents the result and discussion of findings generated from the study. Section 4.1 shows the data analysis and model training, while 4.2 presented the performance evaluation of the ML algorithms.

## 4.1 DATA ANALYSIS AND MODEL IMPLEMENTATION

A transformer dataset containing 19,352 instances and 11 attributes was extracted from a CSV file and loaded into a Jupyter Notebook environment (Figures 4.1), After data collection, an exploratory data analysis (EDA) is carried out on the dataset to  
evaluate and classify the data's key features by means of visualizations, then data cleaning and preparation is carried out before the models are implemented.

### 4.1.1 DATASET DESCRIPTION

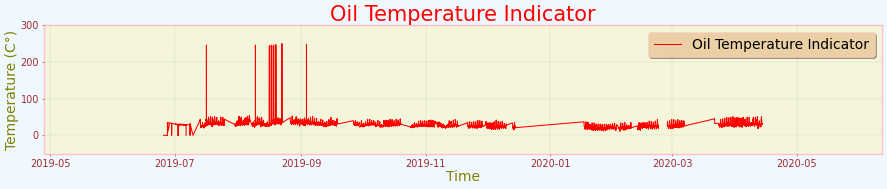
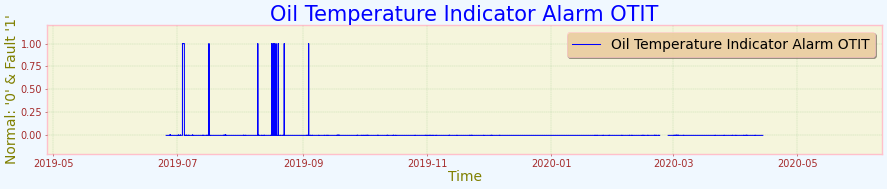
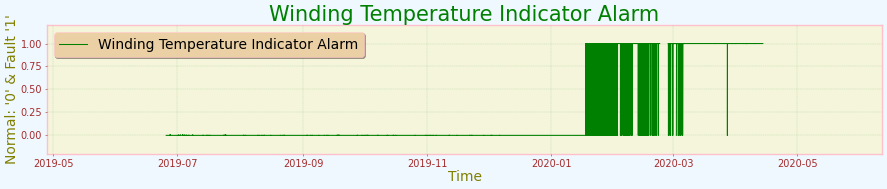
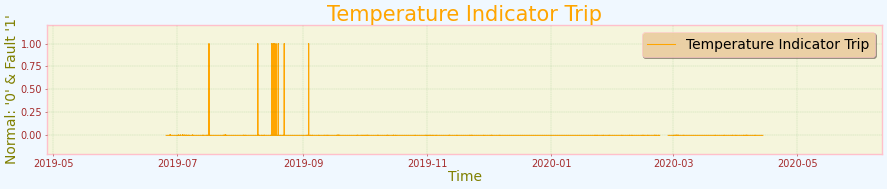
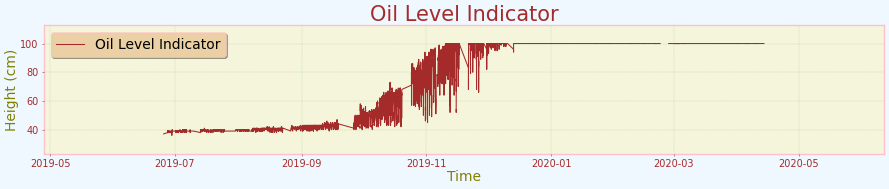
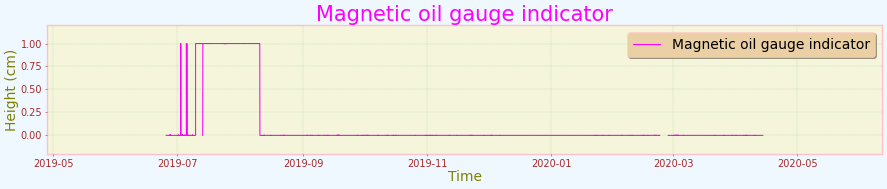
This data was collected via IoT devices from June 25th, 2019 to April 14th, 2020 which was updated every 15 minutes. The table below shows the first 5 rows of the dataset.

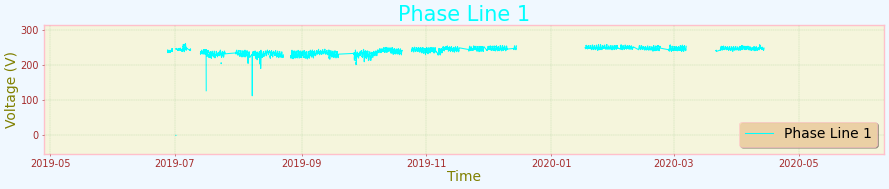
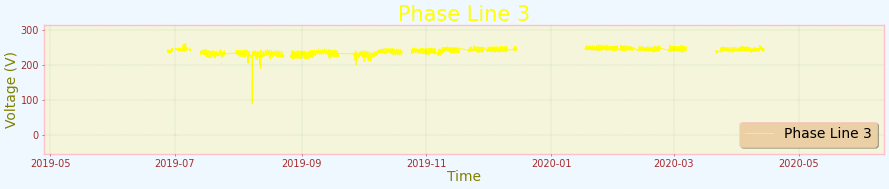
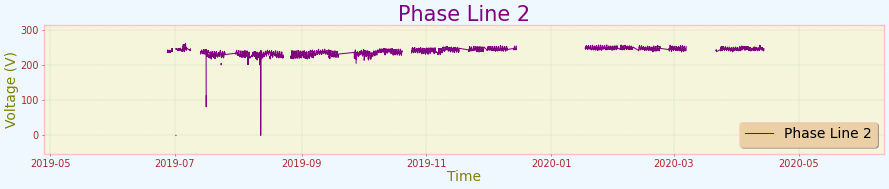
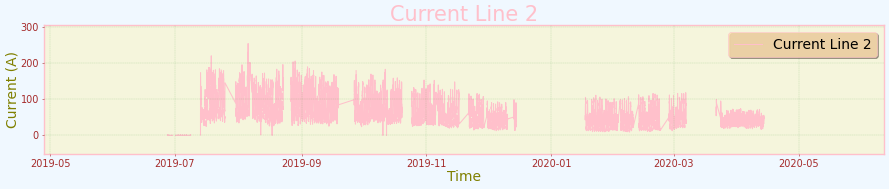
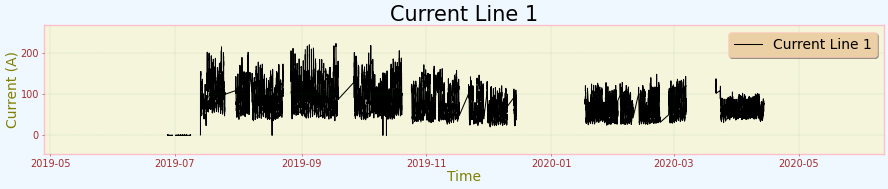
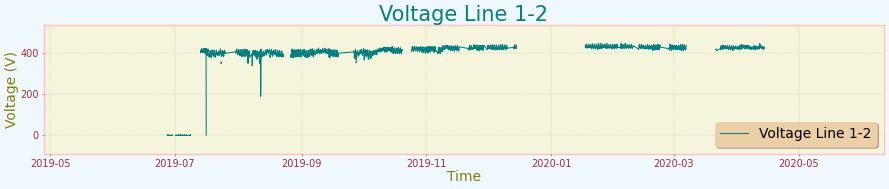
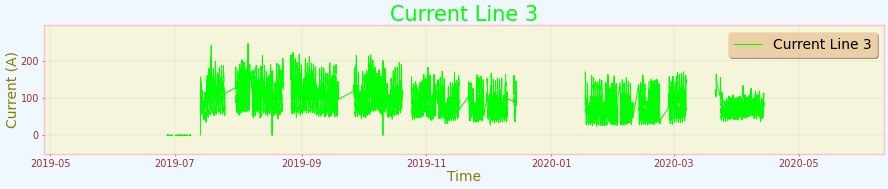
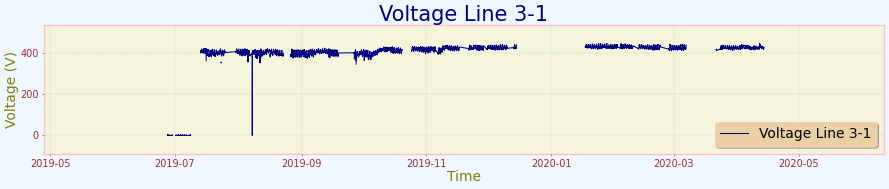
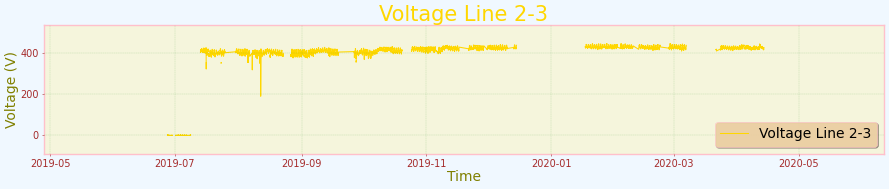
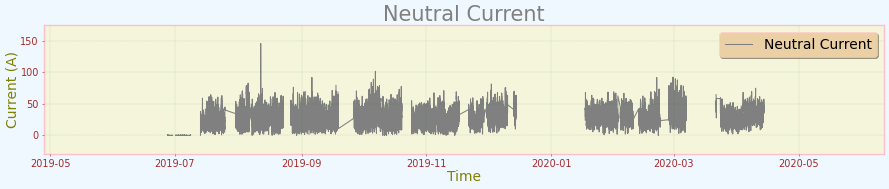
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **DeviceTimeStamp** | **OTI** | **WTI** | **ATI** | **OLI** | **OTI\_A** | **OTI\_T** | **MOG\_A** |
| 6/25/2019 13:06 | 0 | 0 | 0 | 37 | 0 | 0 | 0 |
| 6/25/2019 13:09 | 0 | 0 | 0 | 37 | 0 | 0 | 0 |
| 6/27/2019 10:49 | 0 | 0 | 0 | 38 | 0 | 0 | 0 |
| 6/27/2019 10:51 | 0 | 0 | 0 | 38 | 0 | 0 | 0 |
| 6/27/2019 10:52 | 0 | 0 | 0 | 39 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DeviceTimeStamp** | **VL1** | **VL2** | **VL3** | **IL1** | **IL2** | **IL3** | **VL12** | **VL23** | **VL31** | **INUT** |
| 6/25/2019 13:06 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6/27/2019 10:49 | 238.7 | 238.7 | 238.8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6/27/2019 10:51 | 238.4 | 238.5 | 238.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6/27/2019 10:52 | 239.9 | 240 | 240 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6/27/2019 10:52 | 239.9 | 240 | 240 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 4.1.2 EXPLORATORY DATA ANALYSIS

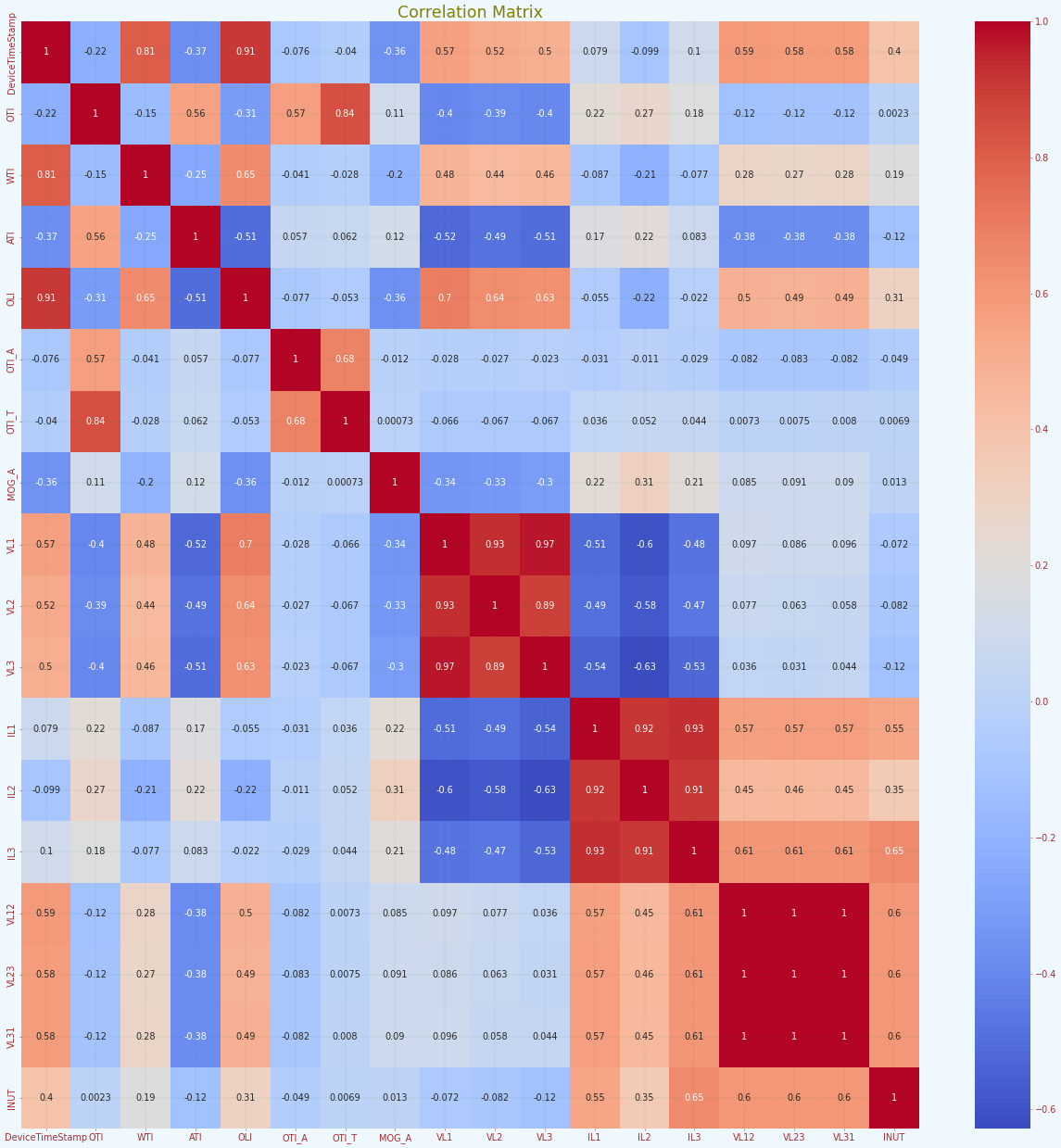
Visualizations relating to the distribution of the data to be used for the modeling are provided below.

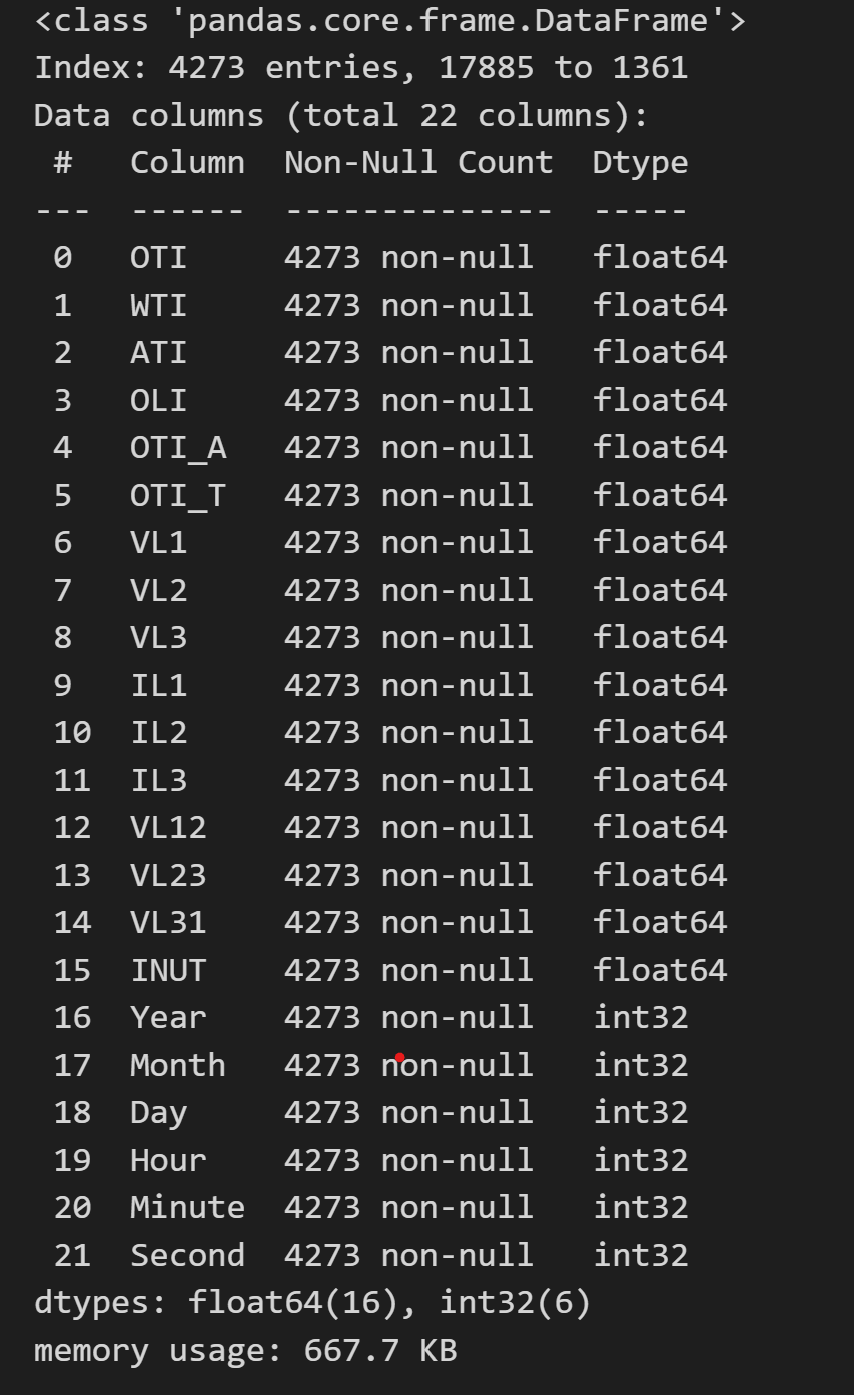
### CORRELATION MARTRIX

To further understand the data, we take a closer look to see the correlations between the features. This is done to ensure that the correlation between them is not too high and that they are suitable for machine learning algorithms and avoid overfitting or underfitting the models. Figure below shows the heatmap of the features in the dataset.



### DATA PREPARATION

This stage entails the steps taken to ensure that data is suitable for machine learning.  
It begins with viewing the data types and checking if there are missing values, then  
investigating the data type. Figure below shows the information of individual feature.



### DATA SPLITING

To ensure robust and unbiased model evaluation, we divided our dataset into two subsets: training (80%) and test (20%). The training set is used to fit the models and tune hyperparameters, while the test set provides an independent assessment of model performance.

### DATA NORMALIZATION

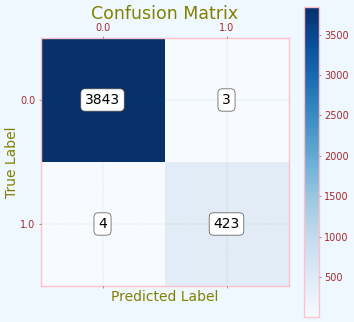
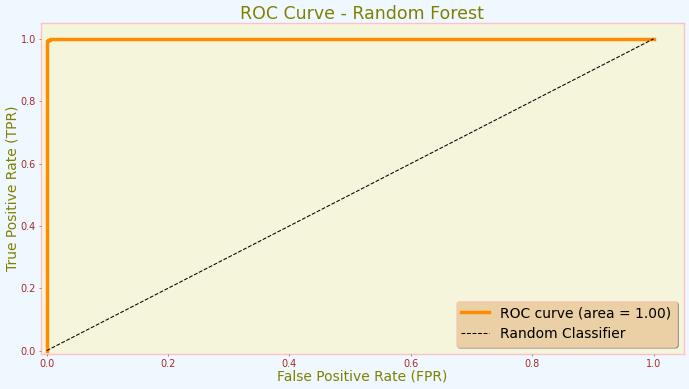
Data can include attributes with a mixture of scales for different quantities, and this variance affects the results of machine learning algorithms. Normalization is a scaling method in which values are moved and rescaled such that they end up between 0 and 1. It is also called Min-Max scaling. Scaling was done using Scikit-Learn's RobustScaler because of it less sensitivity to extreme values.

### MODEL TRAINING AND IMPLEMENTATION

The model training process involved standardizing the data, tuning hyperparameters for both XGBoost and Random Forest classifiers using grid search, and fitting the best models on the training data. Predictions were then made on the test data, and the accuracy of the models was evaluated to ensure they were well-tuned and capable of making accurate predictions on unseen data.

**RANDOM FOREST ALGORITHM**

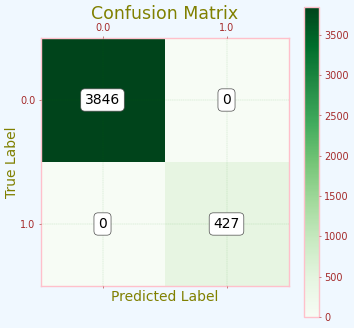
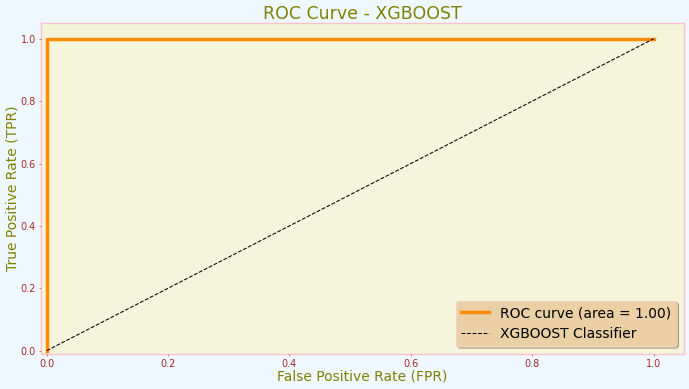
|  |  |
| --- | --- |
| **Measure** | **Value** |
| **Accuracy** | 99.8% |
| **F1 score** | 99.8% |
| **Recall** | 99.8% |
| **precision** | 99.8% |
| **Roc** | 99.99% |

The Random forest classifier had an accuracy score of 99.8% with a precision of  
99.8%. The recall for the model is at 99.8%, with an F1 score of 99.8% and roc of 99.99%.

**XGBOOST ALGORITHM**

|  |  |
| --- | --- |
| **Measure** | **Value** |
| **Accuracy** | 100% |
| **F1 score** | 100% |
| **Recall** | 100% |
| **precision** | 100% |
| **Roc** | 100% |
| **Execution time** | 50 Minutes |

The XGBOOST algorithm had an accuracy of 100% with an F1- score of 100%. The  
recall for the algorithm is at 100% and Roc of 100%.

## 4.2 PERFOMANCE EVALUATION OF THE MODEL

Accuracy, Confusion Matrix, AUROC (Area Under the Receiver Operating Characteristics), Precision, Recall and execution time are metrics used to evaluate the performance of both models.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **MODEL** | **PERFOMANCE SUMMARY OF ALGORITHMS** | | | | | |
| **ACCURACY** | **PRECISION** | **RECALL** | **F1-SCORE** | **ROC** | **EXECUTION**  **TIME** |
| **RANDOM FOREST** | 99.8362 | 99.836 | 99.8362 | 99.8361 | 99.9961 | 8 min |
| **XGBOOST** | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 50 min |

The table in figure above shows the performance of both algorithms. The Random Forest model achieved an accuracy of 99.8%, with precision, recall, and F1-score all hovering around 0.99836. This indicates a high level of reliability in its predictions. Additionally, the model completed its execution in a relatively short time of 8 minutes.

In comparison, the XGBoost model delivered flawless results, with an accuracy, precision, recall, and F1-score all at 100%. The ROC score was also perfect at 100%, showcasing its exceptional ability to distinguish between classes. However, this remarkable performance came with a significantly longer execution time of 50 minutes.

These results highlight the strengths and trade-offs of each algorithm in terms of accuracy and execution time. The figure below shows the ROC comparison of the models.

