CREDIT CARD FRAUD DETECTION REPORT

1. DESCRIPTION OF DESIGN CHOICES

1.1. Data Preprocessing Steps

Data Cleaning: Data preparation, data imputation, data cleaning, error checks, duplication removal, handling the missing variables through simply mean/mode.

Outlier Detection: We used the Interquartile Range (IQR) technique to seek and clear away outliers from numerical characteristics mainly in the 'Amount' column to fortify model robustness.

Normalization: Normalization is done through Standard Scaling to standardize the variables, as it ensures that all the attributes contribute equally to model training.

Data Balancing: The balancing of the data has been done using SMOTE (Synthetic Minority Over-sampling Technique) to resolve the class imbalance existing between fraudulent and non-fraudulent transactions.

1.2. Model Selection

For model selection, we used **pyCaret** module.

PyCaret is an open-source application of simplified machine learning models for use in the Python programming language. PyCaret simplifies the process of create – build – evaluate - deploy for machine learning models, allowing small snippets of code such that beginners and experienced users alike may invoke effortless machine learning tasks.

It incorporates a high-level API which rests upon automated machine learning for a multitude of machine learning pipeline processes such as data preprocessing, model selection, hyperparameter tuning, and model evaluation.

1.2.1. Model Comparison

PyCaret is a computer-aided, straightforward comparison of all machine learning models to identify that which best performs in respect to a given dataset.

1.2.2. Visualization

The built-in library functions allow the user to visualize performance parameters as far as features are concerned. Interrupting the flow of results interpretation has therefore been made quite easy.

1.2.3. Model Deployment

Once a finalized model is available, PyCaret allows the effortless saving and loading of such a model for future use in production.

PyCaret communicates with other libraries, such as pandas, scikit-learn, and Matplotlib, allowing users to intermingle their advantages with simplicity in workflow.

1.3. Model Used

We have used pycaret module to identify the best model for this project. The following are the 5 promising models based on accuracy metric:

A) Extreme Gradient Boosting (XGBoost)

XGBoost (Extreme Gradient Boosting) is a popular and powerful machine learning algorithm designed for supervised learning tasks, particularly for regression and classification problems.

As we have imbalanced dataset for this project, Xgboost is a good choice as Xgboost works well with imbalanced data when combined with additional data augmentation techniques like, SMOTE, Standard scaling, feature scaling, etc.

OverSampling:

```
from imblearn.over_sampling import SMOTE

# Handle class imbalance with SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
print("X_resampled shape:", X_resampled.shape)
print("y_resampled shape:", y_resampled.shape)
```

Standard Scaling:

```
from sklearn.preprocessing import StandardScaler

# Scale the features
scaler = StandardScaler()

X_resampled_scaled = scaler.fit_transform(X_resampled)
X_test_scaled = scaler.transform(X_test)
```

Model Building:

```
import xgboost as xgb

xgb_model = xgb.XGBClassifier()
xgb_model.fit(X_resampled_scaled, y_resampled)
```

```
# Make predictions on the resampled training set
y_train_pred = xgb_model.predict(X_resampled_scaled)

# Make predictions on the test set
y_test_pred = xgb_model.predict(X_test_scaled)

# Evaluation metrics for the resampled training set
print("Training Classification Report:\n", classification_report(y_resampled, y_train_pred))
print("Training Confusion Matrix:\n", confusion_matrix(y_resampled, y_train_pred))

# Evaluation metrics for the test set
print("Testing Classification Report:\n", classification_report(y_test, y_test_pred))
print("Testing Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
```

B) Random Forest Classifier

Random Forest Classifier is an ensemble algorithm mainly used for classification and regression. It generates a multitude of decision trees during training and outputs the mode (for classification) or mean prediction (for regression) of individual trees.

It could be a suitable option for other methodologies by controlling for class weights or applying methods like oversampling and under sampling in performing variation analysis from imbalanced data sets.

OverSampling:

```
from imblearn.over_sampling import SMOTE

# Handle class imbalance with SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
print("X_resampled shape:", X_resampled.shape)
print("y_resampled shape:", y_resampled.shape)
```

Standard Scaling:

```
from sklearn.preprocessing import StandardScaler

# Scale the features
scaler = StandardScaler()

X_resampled_scaled = scaler.fit_transform(X_resampled)
X_test_scaled = scaler.transform(X_test)
```

Model Building:

```
# Train the Random Forest Classifier
rf_model = RandomForestClassifier(max_depth=10, n_estimators=100, random_state=42)
rf_model.fit(X_resampled_scaled, y_resampled)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=10, random_state=42)
```

```
# Make predictions on the resampled training set
y_train_pred = rf_model.predict(X_resampled_scaled)

# Make predictions on the test set
y_test_pred = rf_model.predict(X_test_scaled)

# Evaluation metrics for the resampled training set
print("Training Classification Report:\n", classification_report(y_resampled, y_train_pred))
print("Training Confusion Matrix:\n", confusion_matrix(y_resampled, y_train_pred))

# Evaluation metrics for the test set
print("Testing Classification Report:\n", classification_report(y_test, y_test_pred))
print("Testing Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
```

C) Extra Trees Classifier

The Extra Trees classifier is an ensemble learning method, which combines the predictions of multiple decision trees. Like Random Forest, it works on the same principle of bagging, although it introduces an extra bit of randomness into the tree-building process. This often results in more accurate predictions and reduces overfitting.

OverSampling:

```
from imblearn.over_sampling import SMOTE

# Handle class imbalance with SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
print("X_resampled shape:", X_resampled.shape)
print("y_resampled shape:", y_resampled.shape)
```

Standard Scaling:

```
from sklearn.preprocessing import StandardScaler

# Scale the features
scaler = StandardScaler()

X_resampled_scaled = scaler.fit_transform(X_resampled)
X_test_scaled = scaler.transform(X_test)
```

Model Building:

```
from sklearn.ensemble import ExtraTreesClassifier

# Initialize the Extra Trees Classifier
etc_model= ExtraTreesClassifier(n_estimators=100, max_depth=10, min_samples_split=3, random_state=42)
etc_model.fit(X_resampled_scaled, y_resampled)

# Make predictions on the resampled training set
y_train_pred = etc_model.predict(X_resampled_scaled)

# Make predictions on the test set
y_test_pred = etc_model.predict(X_test_scaled)

# Evaluation metrics for the resampled training set
print("Training Classification Report:\n", classification_report(y_resampled, y_train_pred))
print("Training Confusion Matrix:\n", confusion_matrix(y_resampled, y_train_pred))

# Evaluation metrics for the test set
print("Testing Classification Report:\n", classification_report(y_test, y_test_pred))
print("Testing Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
```

D) Logistic Regression

Logistic Regression is a standard binary classification algorithm in machine learning. Its simplicity, interpretability, and effectiveness make it a valuable tool for various applications. Thus, the simple model can provide you fairly good results if you understand the assumptions of Logistic Regression and tune its parameters while using it for predictive modelling across domains.

OverSampling:

```
from imblearn.over_sampling import SMOTE

# Handle class imbalance with SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
print("X_resampled shape:", X_resampled.shape)
print("y_resampled shape:", y_resampled.shape)
```

Standard Scaling:

```
from sklearn.preprocessing import StandardScaler

# Scale the features
scaler = StandardScaler()

X_resampled_scaled = scaler.fit_transform(X_resampled)
X_test_scaled = scaler.transform(X_test)
```

Model Building:

```
from sklearn.linear_model import LogisticRegression

# Train the Random Forest Classifier
lr_model = LogisticRegression(class_weight='balanced')
lr_model.fit(X_resampled_scaled, y_resampled)

# Make predictions on the resampled training set
y_train_pred = lr_model.predict(X_resampled_scaled)

# Make predictions on the test set
y_test_pred = lr_model.predict(X_test_scaled)

# Evaluation metrics for the resampled training set
print("Training Classification Report:\n", classification_report(y_resampled, y_train_pred))
print("Training Confusion Matrix:\n", confusion_matrix(y_resampled, y_train_pred))

# Evaluation metrics for the test set
print("Testing Classification Report:\n", classification_report(y_test, y_test_pred))
print("Testing Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
```

E) Light Gradient Boosting Machine

LightGBM is a state-of-the-art gradient boosting framework. It runs faster and uses less memory than other gradient boosting implementations, which makes it particularly helpful with large datasets. This is achieved through numerous innovations and optimization features.

Apart from these plain explanations, LightGBM is greatly qualified to work with unbalanced datasets via the various built-in features provided like SMOTE, scaling, feature selection etc. Usage of such tools brings out the better model performance and, thus improved predictive accuracy for the minority class.

OverSampling:

```
from imblearn.over_sampling import SMOTE

# Handle class imbalance with SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
print("X_resampled shape:", X_resampled.shape)
print("y_resampled shape:", y_resampled.shape)
```

Standard Scaling:

```
from sklearn.preprocessing import StandardScaler

# Scale the features
scaler = StandardScaler()

X_resampled_scaled = scaler.fit_transform(X_resampled)
X_test_scaled = scaler.transform(X_test)
```

Model Building:

```
import lightgbm as lgb

lgb_model = lgb.LGBMClassifier( n_estimators=100, max_depth=5, random_state=42)
lgb_model.fit(X_resampled_scaled, y_resampled)

# Make predictions on the resampled training set
y_train_pred = lgb_model.predict(X_resampled_scaled)

# Make predictions on the test set
y_test_pred = lgb_model.predict(X_test_scaled)

y_probs = lgb_model.predict_proba(X_test_scaled)[:, 1]

# Evaluation metrics for the resampled training set
print("Training Classification Report:\n", classification_report(y_resampled, y_train_pred))
print("Training Confusion Matrix:\n", confusion_matrix(y_resampled, y_train_pred))

# Evaluation metrics for the test set
print("Testing Classification Report:\n", classification_report(y_test, y_test_pred))
print("Testing Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
```

2. PERFORMANCE EVALUATION

After building the model, we evaluated the model to see how well they are doing on unseen data. With the help of Accuracy, Precision, Recall Score and F1 Score, we predicted the best model among 5 selected models.

Accuracy: It represents the proportion of correctly classified instances out of the total instances in the dataset.

Precision: Precision is one of the evaluation metrics for the performance of a classifying model that measures the correctness of prediction of members of the positive class. It is equal to the number of correct predictions of positive class by the model divided by all positive predictions made by the model.

Recall: Recall, also called the sensitivity or true positive rate, is a measure of classification model performance that measures how well the model classifies positive instances. It describes the proportion of actual positive cases correctly predicted by the model.

F1 Score: F1 Score is one of the performance metrics that one uses in classification problems particularly when classes are imbalanced. It's actually the harmonic mean of precision and recall such that the score perfectly balances both metrics. When you need an optimal balance between precision and recall, the F1 Score comes out to be the most useful.

NOTE: When dealing with imbalanced data, we should NOT be solely depending on Accuracy, as high accuracy can be misleading in such cases. It's often advisable to consider additional metrics such as: Precision, Recall, F1 – Score and ROC value.

Let us look at model performances.

a) Extreme Gradient Boost (XGBoost)

After building the model, we tested the model with unseen data (Test data) and the following results were obtained (Fraud Class):

Accuracy: 99.94%

Recall: 83.67%

Precision: 82.82%

F1 – Score: 83%

```
Testing Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.83      0.84      0.83        98

    accuracy                           1.00     56962
   macro avg       0.91      0.92      0.92     56962
weighted avg       1.00      1.00      1.00     56962

Testing Confusion Matrix:
 [[56847    17]
 [   16    82]]
```

Here, we can see, the model is performing well on unseen data, however there is scope for improvement. In order to enhance our model performance, we performed **Threshold Tuning.**

thresholds = np.arange(0.1, 1.0, 0.1)

PREDICTION AFTER THRESHOLD TUNING

Class 1: Fraud Transaction metrics

Accuracy: 99.96%

Recall: 83.67%

Precision: 93.18%

F1 – Score: 88%

```
Best threshold based on F1-score: 0.9
Final Testing Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.93      0.84      0.88        98

    accuracy                           1.00     56962
   macro avg       0.97      0.92      0.94     56962
weighted avg       1.00      1.00      1.00     56962
```

Best threshold value based on F1 – score: 0.9

RESULT INTEPRETATION:

Class 0: Normal Transaction

Class 1: Fraud Transaction

Precision:

Class 0: 1.00 (100%) – Perfect accuracy in predicting class 0.
Class 1: 0.93 (93%) – 93% of predicted class 1 instances were correct.

Recall:

Class 0: 1.00 (100%) – The model correctly identified all actual class 0 instances.
Class 1: 0.84 (84%) – The model identified 84% of actual class 1 instances.

F1-Score:

Class 0: 1.00 (100%) – Perfect balance for class 0.
Class 1: 0.88 (88%) – Good balance for class 1.

Performance on Class 1: While precision and F1 score for class 1 are good, recall indicates that some instances were missed.

PREDICTION AFTER HYPERPARAMETER TUNING

Class 1: Fraud Transaction metrics

Accuracy: 99.94%

Recall: 83%

Precision: 92%

F1 – Score: 88%

Classification Report for the Best Model:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.92	0.83	0.87	98
accuracy			1.00	56962
macro avg	0.96	0.91	0.94	56962
weighted avg	1.00	1.00	1.00	56962

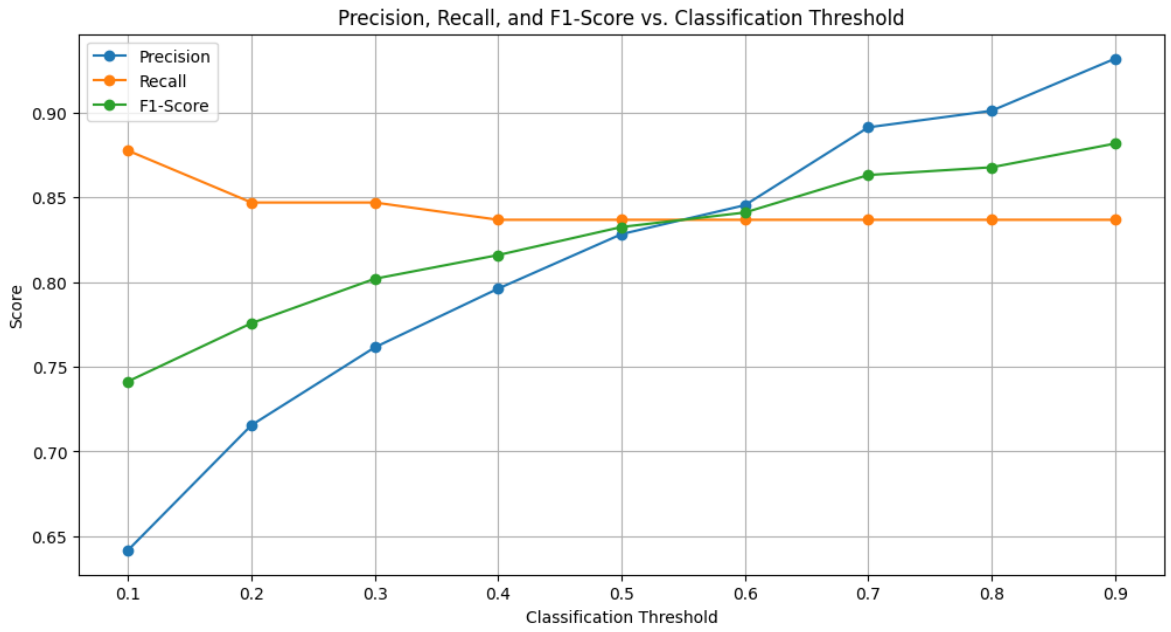
Best Parameters: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 200, 'subsample': 0.8}

As we can predict, the model performance is almost the same after hyperparameter tuning, we will move ahead with “Threshold Adjusted” model and plot graphs.

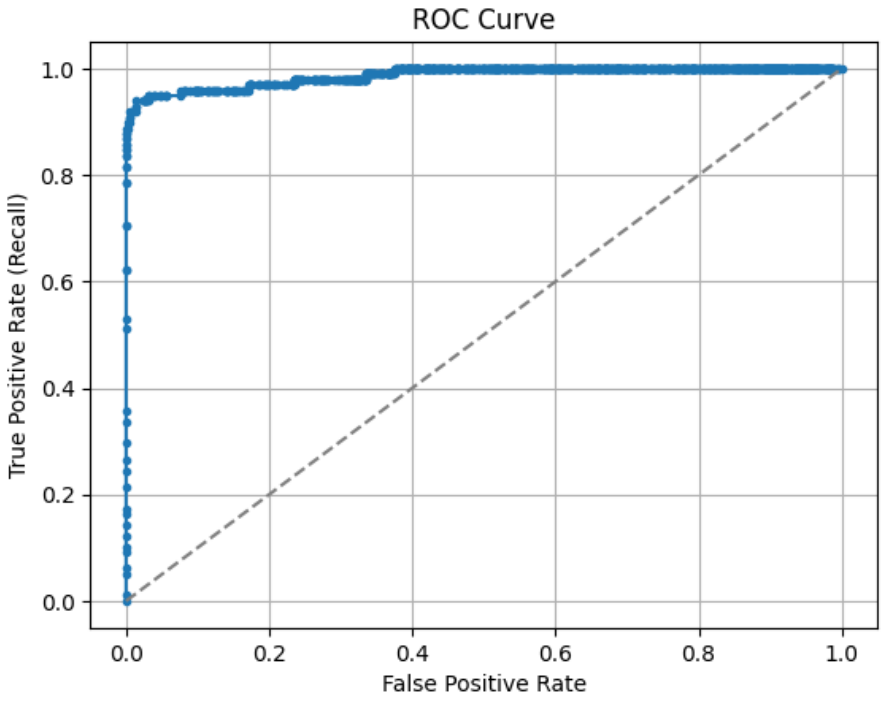
VISUALISATIONS:

1. Precision, Recall, and F1 – Score vs Classification Threshold

Best threshold value based on F1 – score: 0.9

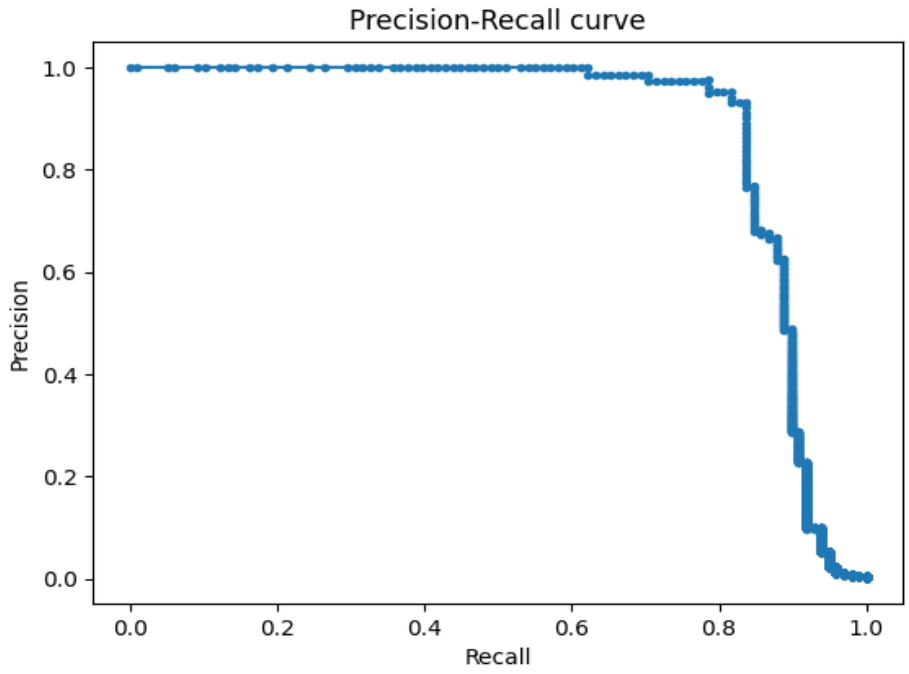


2. ROC Curve

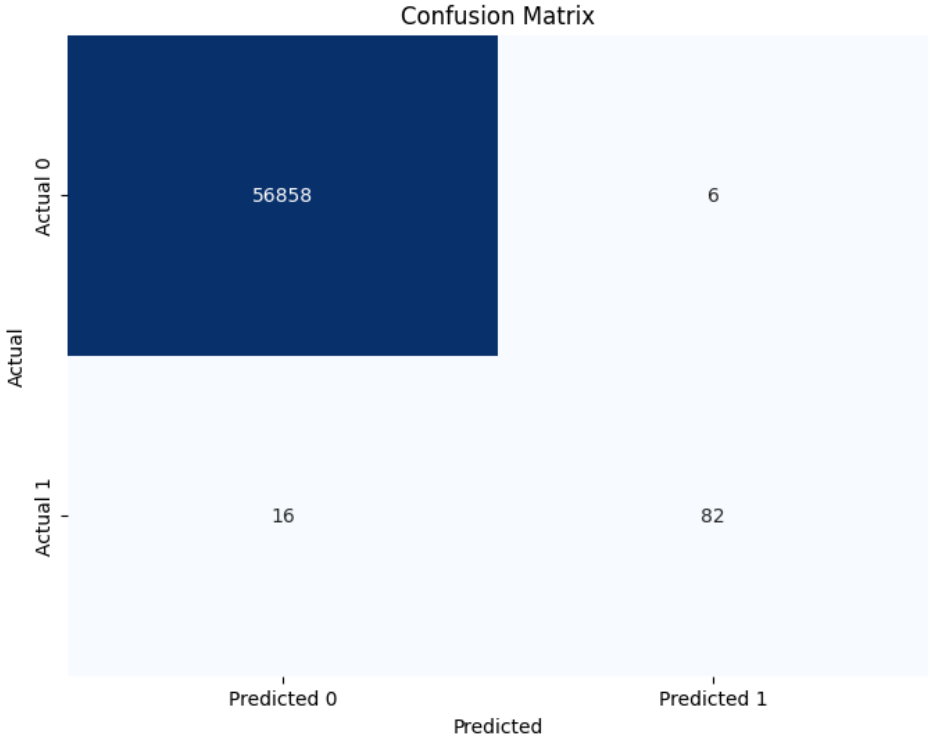


3. Precision - Recall Curve

ROC AUC Score: 0.9870670658527901 (model is excellent at distinguishing between the positive and negative classes)



4. Confusion Matrix

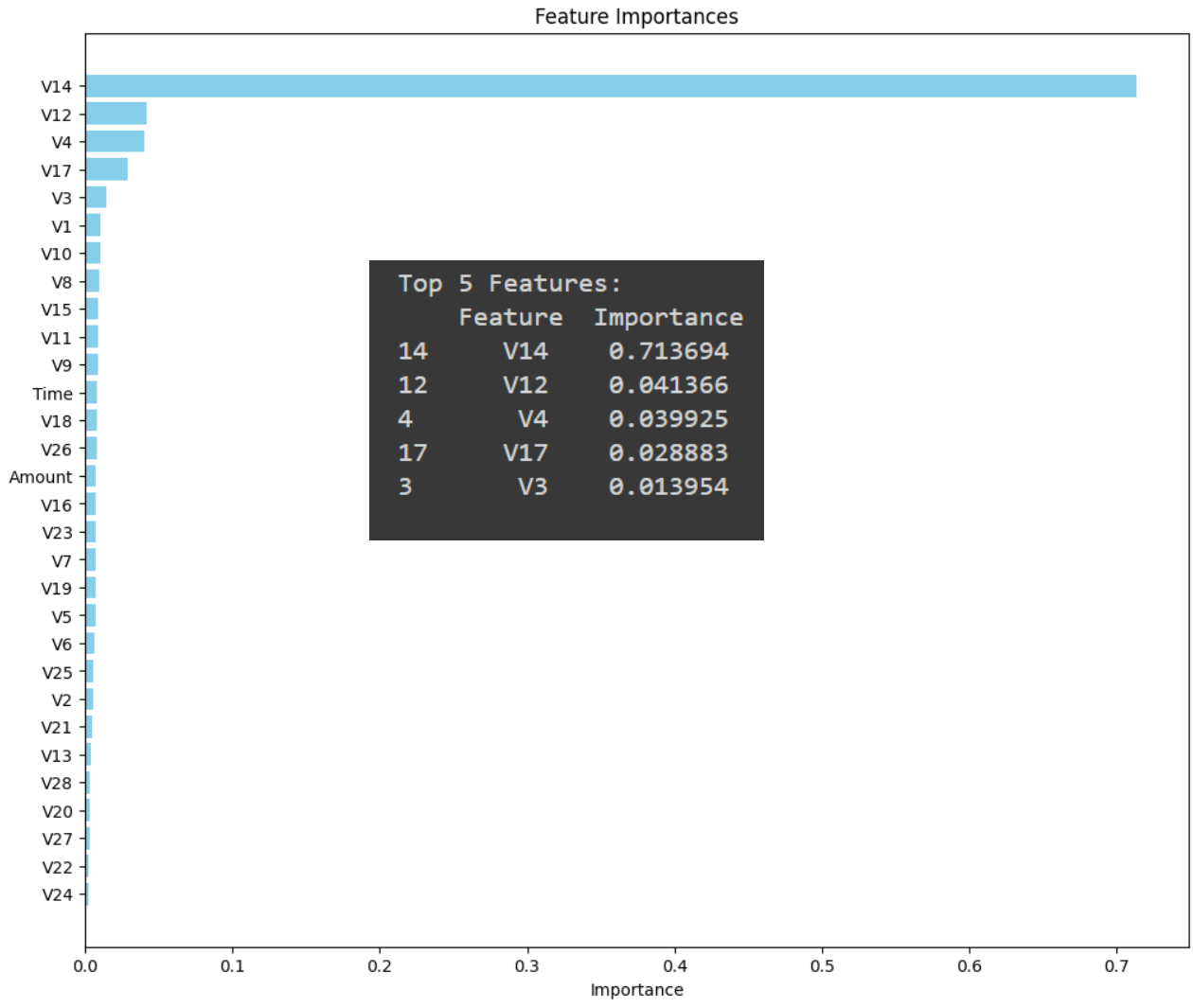


True Negatives (56858): The model correctly predicted the negative class.
False Positives (6): The model incorrectly predicted the positive class when it was actually negative.
False Negatives (16): The model incorrectly predicted the negative class when it was actually positive.
True Positives (82): The model correctly predicted the positive class.

In summary, this confusion matrix would reflect very high numbers of true negatives and true positives but relatively much smaller numbers for the false positives and false negatives, which might indicate that it is doing pretty well in telling the difference between the two classes.

FEATURE SELECTION

Feature selection is one of the major processes in machine learning. In this process, a subset of relevant features or variables, also known as predictors for model construction, needs to be found.



b) Random Forest Classifier

After building the model, we tested the model with unseen data (Test data) and the following results were obtained (Fraud Class):

Accuracy: 99.88%

Recall: 88.77%

Precision: 61.70%

F1 – Score: 73%

```
Testing Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.62      0.89      0.73        98

    accuracy                           1.00     56962
   macro avg       0.81      0.94      0.86     56962
weighted avg       1.00      1.00      1.00     56962

Testing Confusion Matrix:
 [[56810    54]
 [   11    87]]
```

Here, we can see, the model is performing moderately on unseen data, however there is scope for improvement. In order to enhance our model performance, we performed **Threshold Tuning.**

thresholds = np.arange(0.1, 1.0, 0.1)

PREDICTION AFTER THRESHOLD TUNING

Class 1: Fraud Transaction metrics

Accuracy: 99.94%

Recall: 83.67%

Precision: 84.53%

F1 – Score: 84%

```
Best threshold based on F1-score: 0.8
Final Testing Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.85      0.84      0.84        98

    accuracy                           1.00     56962
   macro avg       0.92      0.92      0.92     56962
weighted avg       1.00      1.00      1.00     56962
```

Best threshold value based on F1 – score: 0.8

The performance of Random Classifier Model has increased after adjusting the threshold.

RESULT INTEPRETATION:

Class 0: Normal Transaction

Class 1: Fraud Transaction

Precision:

Class 0: 1.00 (100%) – Perfect accuracy in predicting class 0.
Class 1: 0.85 (85%) – 85% of predicted class 1 instances were correct.

Recall:

Class 0: 1.00 (100%) – The model correctly identified all actual class 0 instances.
Class 1: 0.84 (84%) – The model identified 84% of actual class 1 instances.

F1-Score:

Class 0: 1.00 (100%) – Perfect balance for class 0.
Class 1: 0.84 (84%) – Good balance for class 1.

Performance on Class 1: The precision (85%) and recall (84%) for class 1 indicate that while the model is reasonably good at identifying the positive class, there's still room for improvement.

PREDICTION AFTER HYPERPARAMETER TUNING

Class 1: Fraud Transaction metrics

Accuracy: 99.94%

Recall: 80%

Precision: 97%

F1 – Score: 88%

Classification Report for the Best Model:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.97	0.80	0.88	98
accuracy			1.00	56962
macro avg	0.99	0.90	0.94	56962
weighted avg	1.00	1.00	1.00	56962

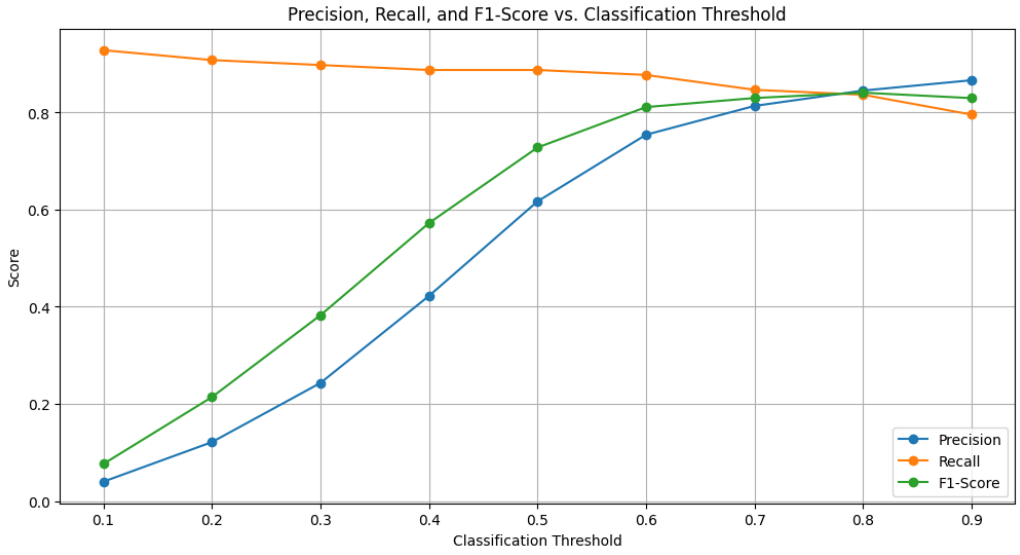
Best Parameters: {'max_depth': None, 'n_estimators': 100}

As we can predict, the model performance slightly increased after hyperparameter tuning, we will move ahead with the model and plot graphs.

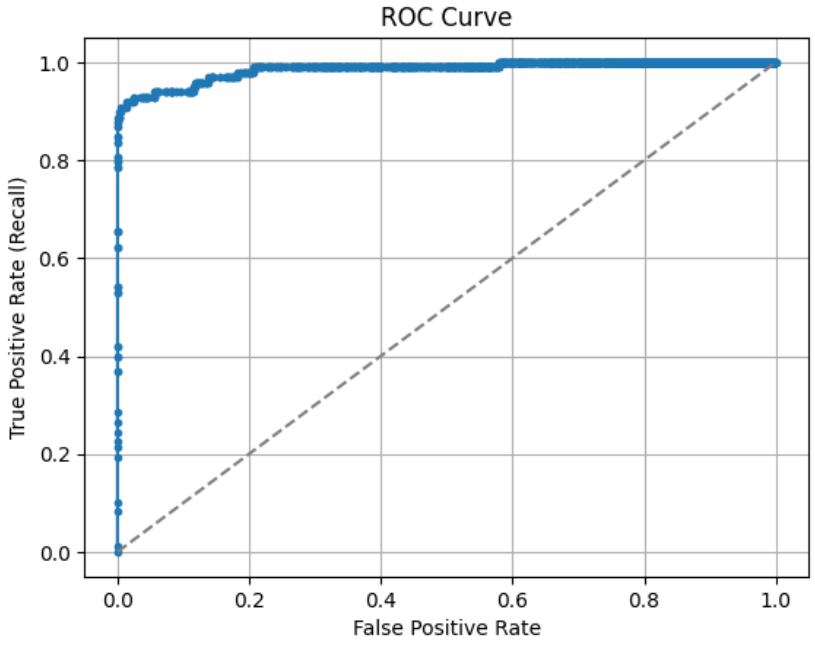
VISUALISATIONS:

1. **Precision, Recall, and F1 – Score vs Classification Threshold**

Best threshold value based on F1 – score: 0.8

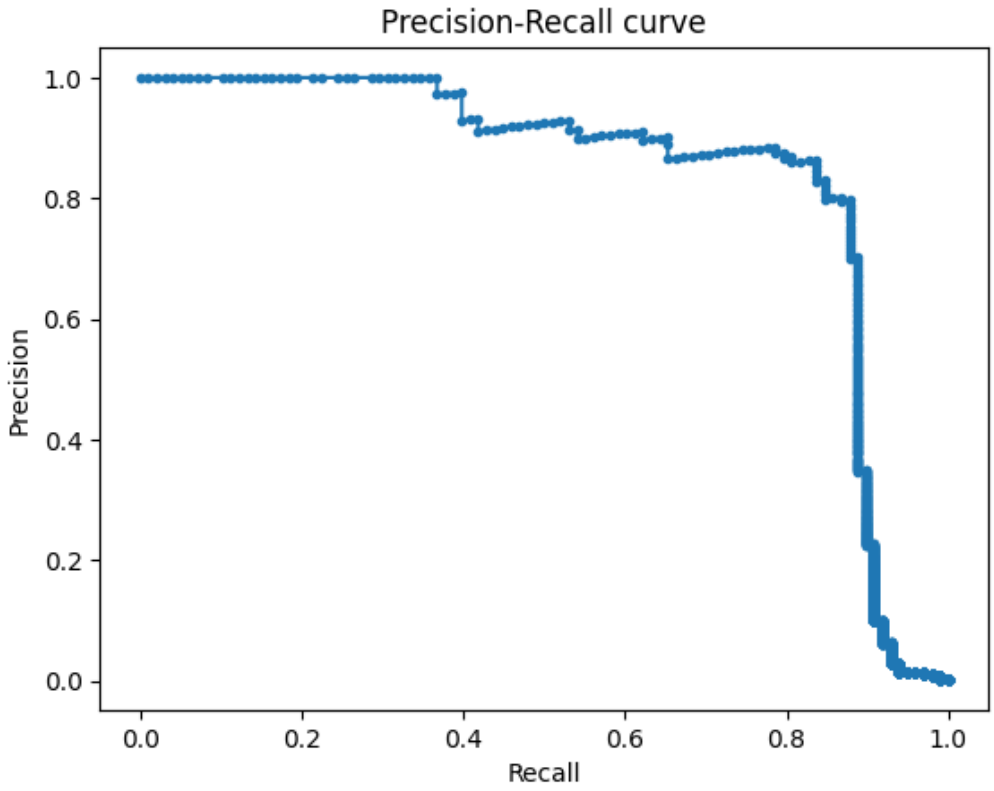


2. **ROC Curve**

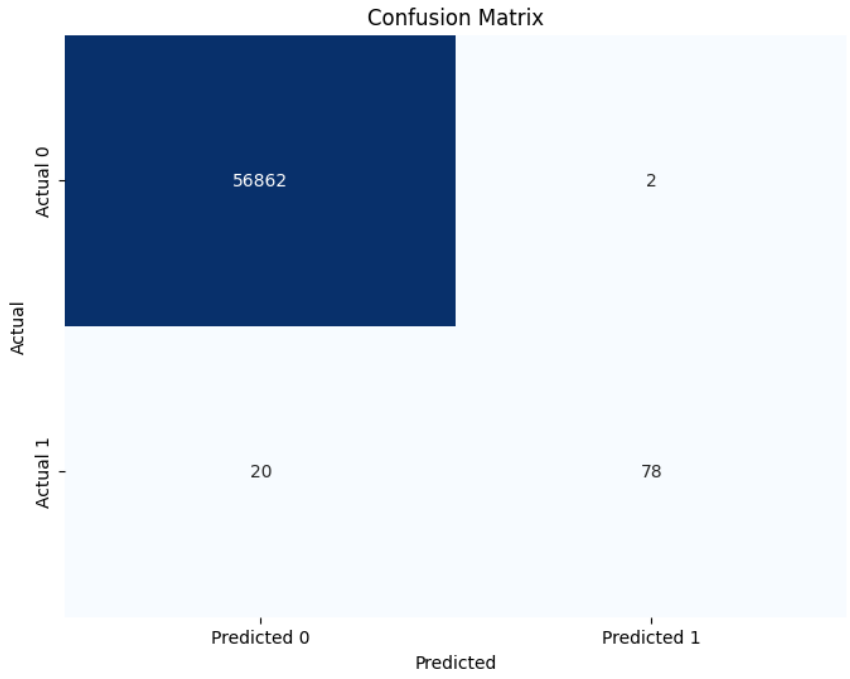


3. Precision - Recall Curve

ROC AUC Score: 0.9852153509124527 (model is excellent at distinguishing between the positive and negative classes)



4. Confusion Matrix (After hyperparameter tuning)

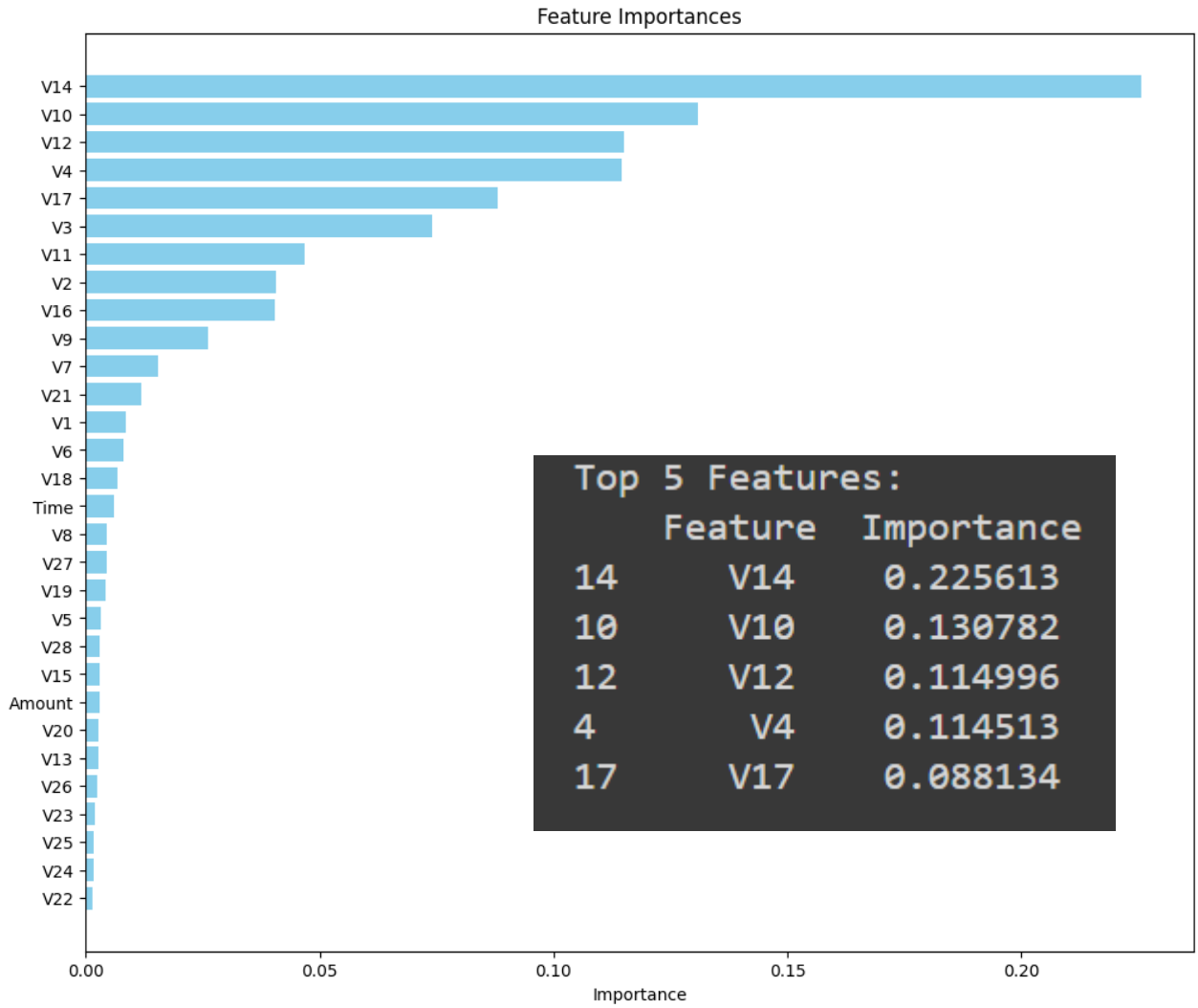


True Negatives (56862): The model correctly predicted the negative class.
False Positives (2): The model incorrectly predicted the positive class when it was actually negative.
False Negatives (20): The model incorrectly predicted the negative class when it was actually positive.
True Positives (78): The model correctly predicted the positive class.

In summary, this confusion matrix would reflect very high numbers of true negatives and true positives but relatively much smaller numbers for the false positives and false negatives, which might indicate that it is doing pretty well in telling the difference between the two classes.

FEATURE SELECTION

Feature selection is one of the major processes in machine learning. In this process, a subset of relevant features or variables, also known as predictors for model construction, needs to be found.



c) Extra Trees Classifier

After building the model, we tested the model with unseen data (Test data) and the following results were obtained (Fraud Class):

Accuracy: 99.87%

Recall: 86.73%

Precision: 58.21%

F1 – Score: 70%

```
Testing Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.58      0.87      0.70        98

    accuracy                           1.00     56962
   macro avg       0.79      0.93      0.85     56962
weighted avg       1.00      1.00      1.00     56962

Testing Confusion Matrix:
 [[56803    61]
 [   13    85]]
```

Here, we can see, the model is performing not so well on unseen data (for fraud cases), however there is scope for improvement. In order to enhance our model performance, we performed **Threshold Tuning.**

PREDICTION AFTER THRESHOLD TUNING

Class 1: Fraud Transaction metrics

Accuracy: 99.95%

Recall: 78.6%

Precision: 93.90%

F1 – Score: 86%

```
Best threshold based on F1-score: 0.8
Final Testing Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.94      0.79      0.86        98

    accuracy                           1.00     56962
   macro avg       0.97      0.89      0.93     56962
weighted avg       1.00      1.00      1.00     56962
```

Best threshold value based on F1 – score: 0.8

RESULT INTEPRETATION:

Class 0: Normal Transaction

Class 1: Fraud Transaction

Precision:

Class 0: 1.00 (100%) – Perfect accuracy in predicting class 0.
Class 1: 0.94 (94%) – 94% of predicted class 1 instances were correct.

Recall:

Class 0: 1.00 (100%) – The model correctly identified all actual class 0 instances.
Class 1: 0.79 (79%) – The model identified 79% of actual class 1 instances.

F1-Score:

Class 0: 1.00 (100%) – Perfect balance for class 0.
Class 1: 0.86 (86%) – Good balance for class 1.

Area for improvement: For Class 1, the recall is 0.79, which means that the model misses 21% of the actual positive instances. This will lead to significant challenges, especially in those applications where detection of positive cases becomes crucial (fraud detection, disease diagnosis, etc.).

PREDICTION AFTER HYPERPARAMETER TUNING

Class 1: Fraud Transaction metrics

Accuracy: 99.94%

Recall: 77%

Precision: 95%

F1 – Score: 85%

Classification Report for the Best Model:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.95	0.77	0.85	98
accuracy			1.00	56962
macro avg	0.97	0.88	0.92	56962
weighted avg	1.00	1.00	1.00	56962

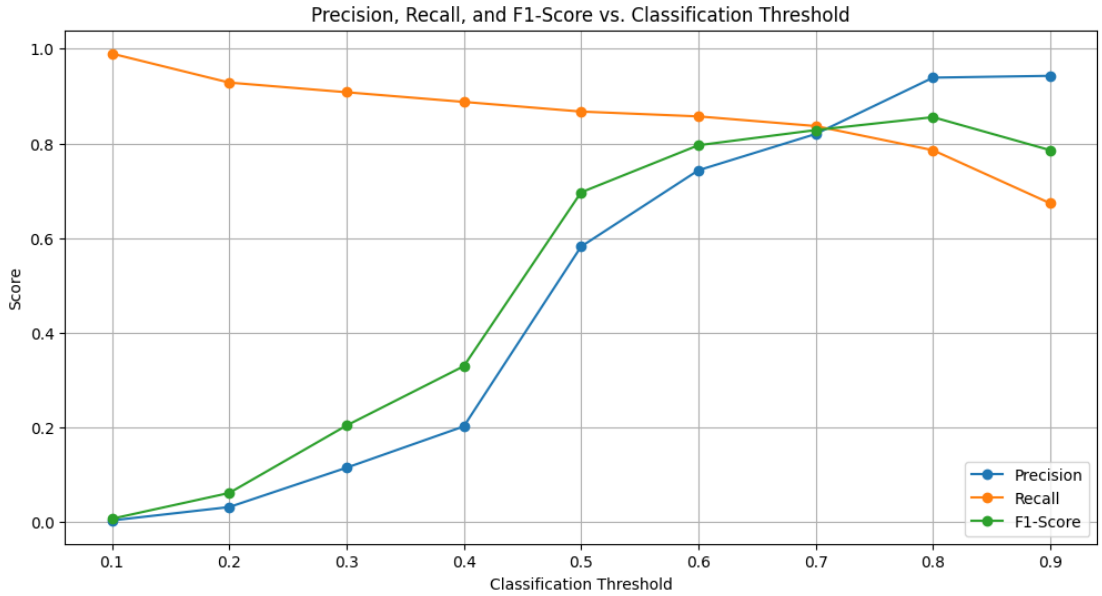
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}

As we can predict, the model performance is almost the same after hyperparameter tuning, we will move ahead with the model and plot graphs.

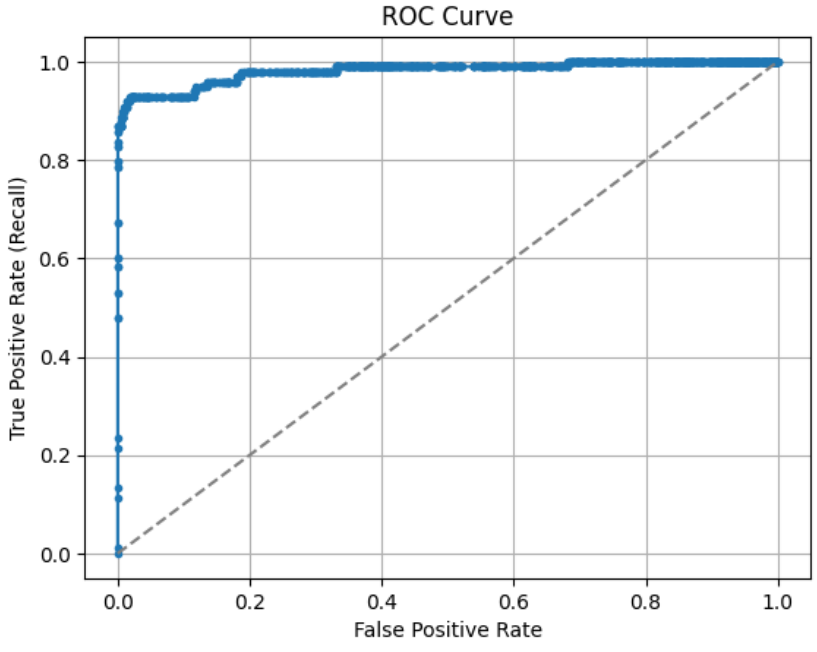
VISUALISATIONS:

1. Precision, Recall, and F1 – Score vs Classification Threshold

Best threshold value based on F1 – score: 0.8

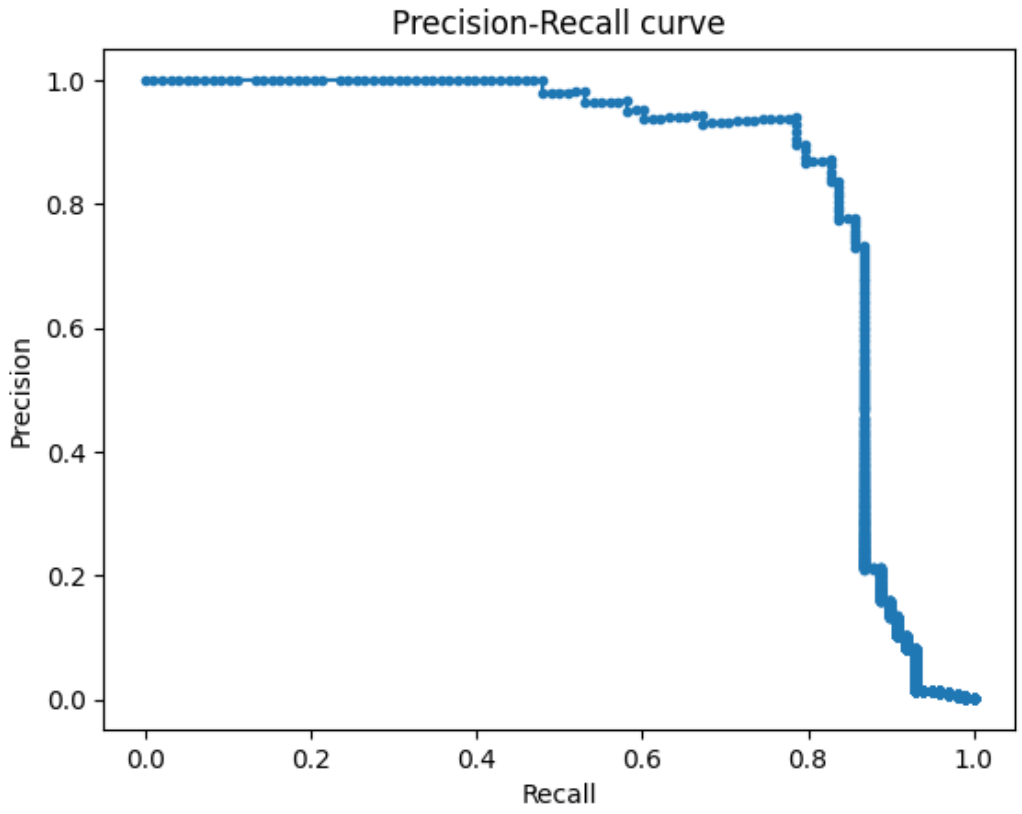


2. ROC Curve

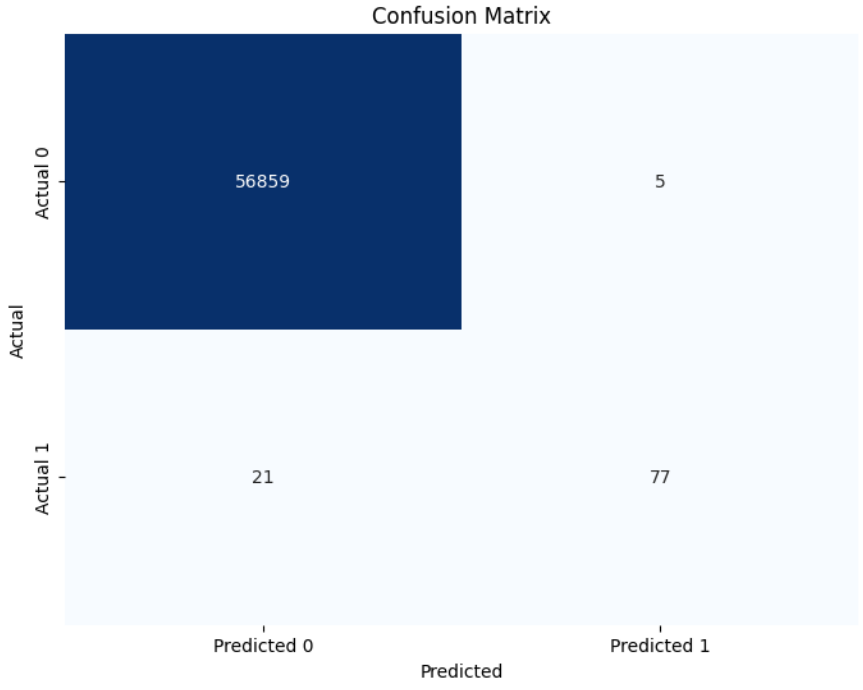


3. Precision - Recall Curve

ROC AUC Score: 0.9814792616540144 (model is excellent at distinguishing between the positive and negative classes)



4. Confusion Matrix

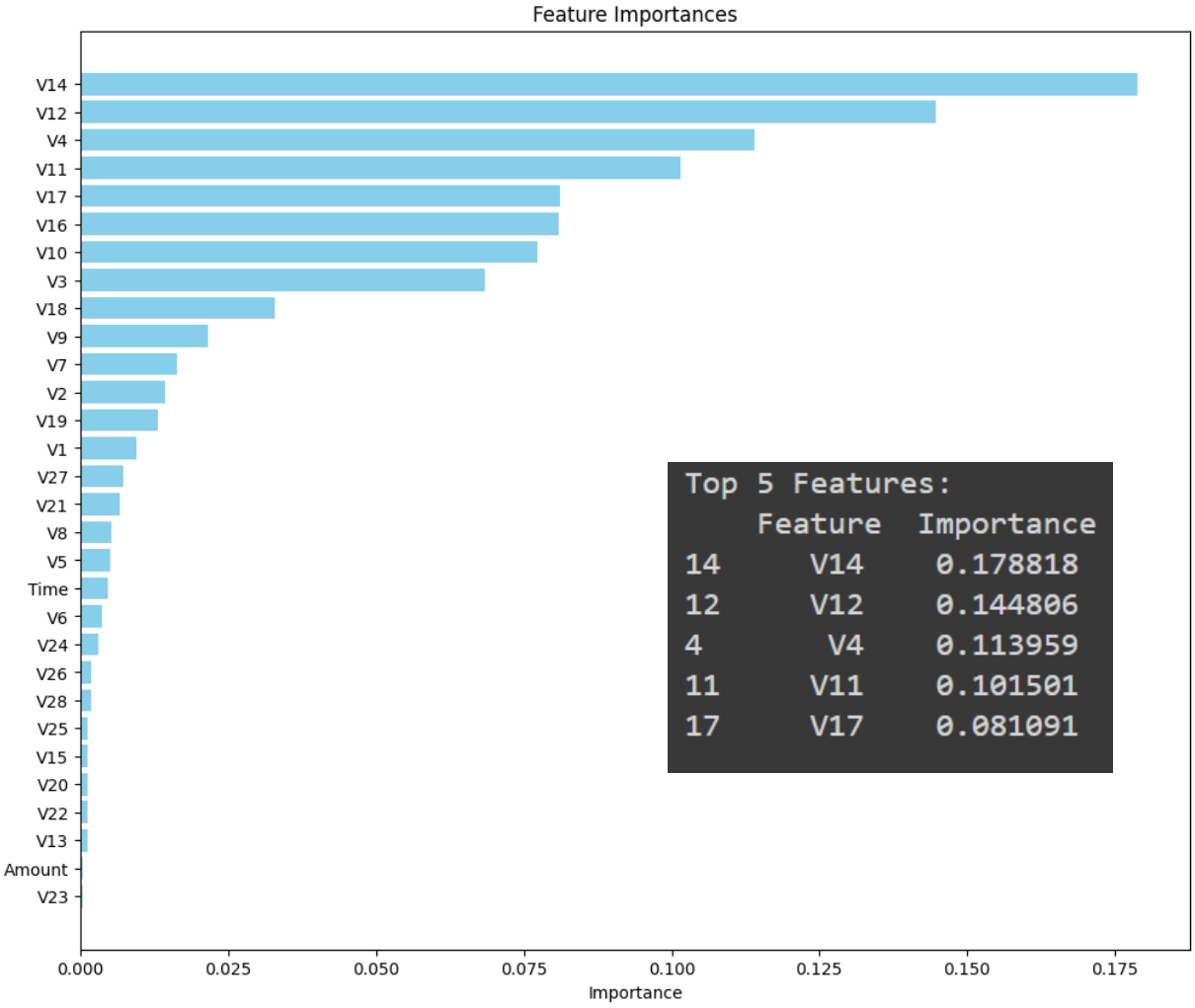


True Negatives (56859): The model correctly predicted the negative class.
False Positives (5): The model incorrectly predicted the positive class when it was actually negative.
False Negatives (21): The model incorrectly predicted the negative class when it was actually positive.
True Positives (77): The model correctly predicted the positive class.

In summary, this confusion matrix would reflect very high numbers of true negatives and true positives but relatively much smaller numbers for the false positives and false negatives, which might indicate that it is doing pretty well in telling the difference between the two classes.

FEATURE SELECTION

Feature selection is one of the major processes in machine learning. In this process, a subset of relevant features or variables, also known as predictors for model construction, needs to be found.



d) Logistic Regression

After building the model, we tested the model with unseen data (Test data) and the following results were obtained (Fraud Class):

Accuracy: 99.17%

Recall: 92.83%

Precision: 16.24%

F1 – Score: 28%

```
Testing Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     56864
           1       0.16      0.92      0.28        98

    accuracy                           0.99     56962
   macro avg       0.58      0.96      0.64     56962
weighted avg       1.00      0.99      0.99     56962

Testing Confusion Matrix:
 [[56400   464]
 [    8    90]]
```

Here, we can see, the model is NOT performing well on unseen data. Though, we are getting testing accuracy as 99%, but as discussed earlier "high accuracy can be misleading in such cases with imbalanced data. It's often advisable to consider additional metrics such as: Precision, Recall, F1 – Score and ROC value".

In order to enhance our model performance, we performed **Threshold Tuning.**

PREDICTION AFTER THRESHOLD TUNING

Class 1: Fraud Transaction metrics

Accuracy: 99.96%

Recall: 86.73%

Precision: 48.02%

F1 – Score: 62%

```
Final Testing Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.48      0.87      0.62        98

    accuracy                           1.00     56962
   macro avg       0.74      0.93      0.81     56962
weighted avg       1.00      1.00      1.00     56962
```

Best threshold value based on F1 – score: 0.9

RESULT INTEPRETATION:

Class 0: Normal Transaction

Class 1: Fraud Transaction

Precision:

Class 0: 1.00 (100%) – Perfect accuracy in predicting class 0.
Class 1: 0.48 (48%) – 48% of predicted class 1 instances were correct.

Recall:

Class 0: 1.00 (100%) – The model correctly identified all actual class 0 instances.
Class 1: 0.87 (87%) – The model identified 87% of actual class 1 instances.

The model correctly identifies 87% of the actual positive instances that are within Class 1, which is a good recall score. The problem, though is precision. This means some of those predictions cannot be relied upon.

F1-Score:

Class 0: 1.00 (100%) – Perfect balance for class 0.
Class 1: 0.62 (62%) – Not so good balance for class 1(room for improvement).

PREDICTION AFTER HYPERPARAMETER TUNING

Class 1: Fraud Transaction metrics

Accuracy: 99.94%

Recall: 87%

Precision: 49%

F1 – Score: 63%

Classification Report for the Best Model:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.49	0.87	0.63	98
accuracy			1.00	56962
macro avg	0.75	0.93	0.81	56962
weighted avg	1.00	1.00	1.00	56962

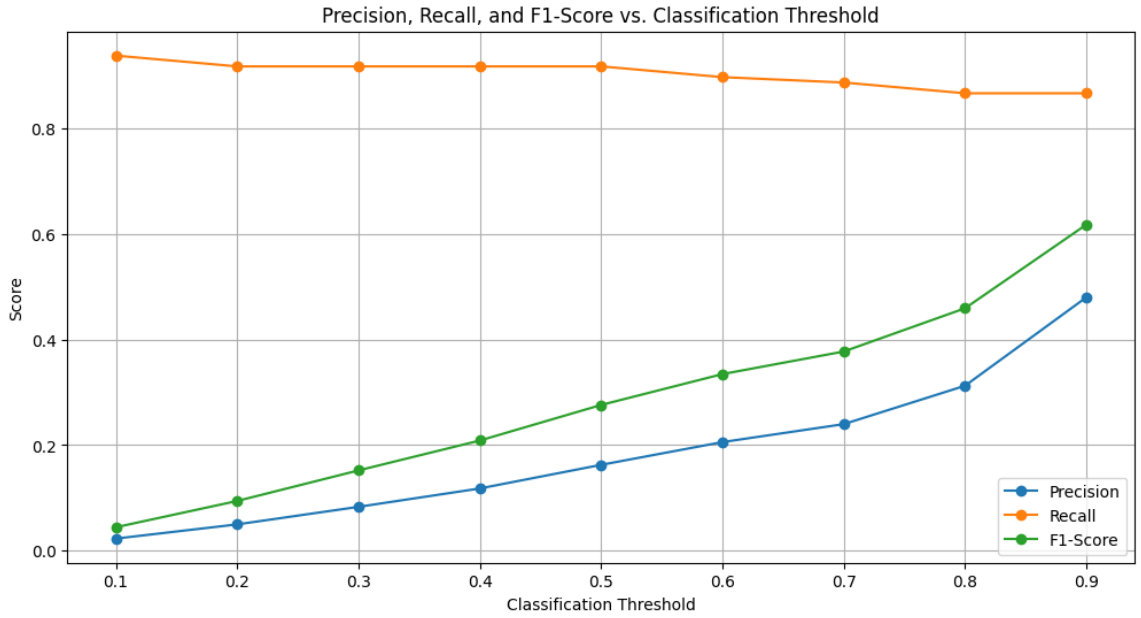
Best Parameters: {'C': 4.281332398719396}

As we can predict, the model performance is almost the same after hyperparameter tuning, we will move ahead and plot graphs.

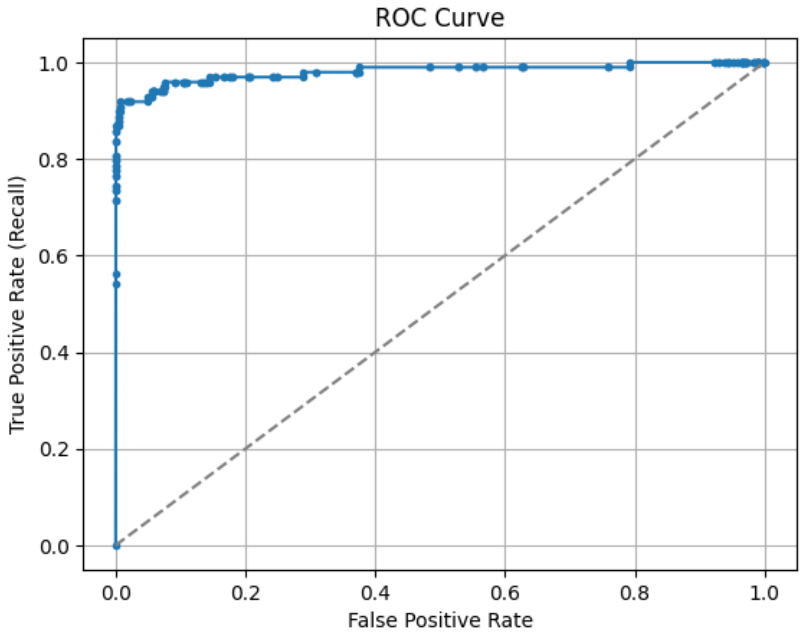
VISUALISATIONS:

1. Precision, Recall, and F1 – Score vs Classification Threshold

Best threshold value based on F1 – score: 0.9

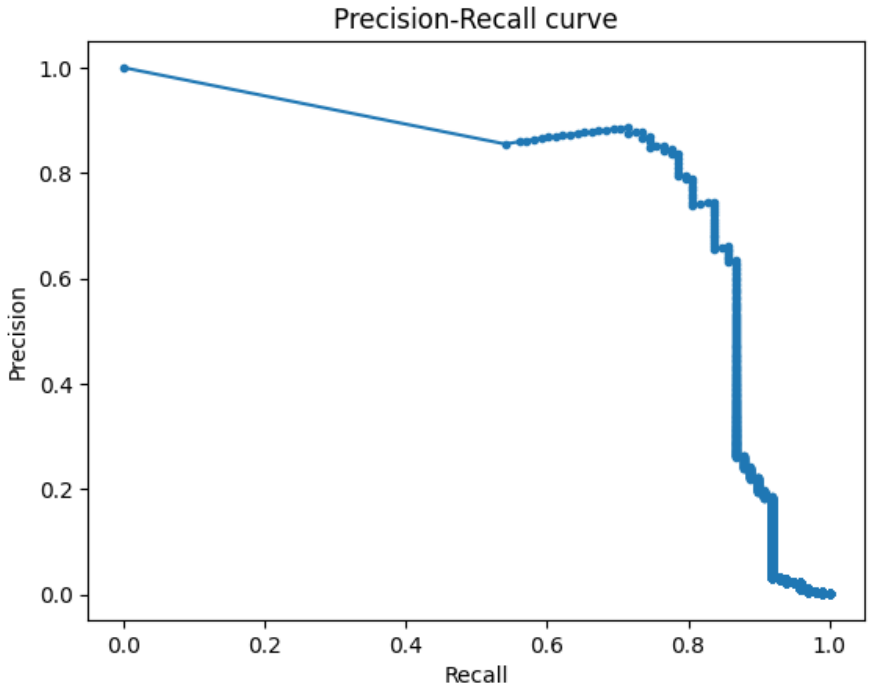


2. ROC Curve

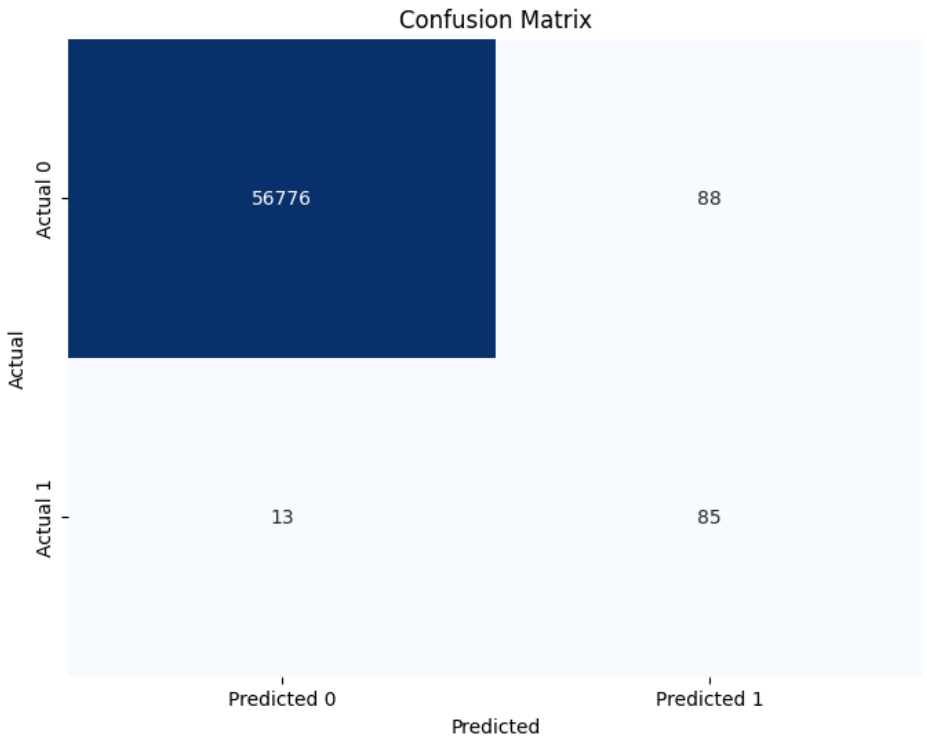


3. Precision - Recall Curve

ROC AUC Score: 0.9806449222204356 (model is excellent at distinguishing between the positive and negative classes)



4. Confusion Matrix (after Hyperparameter Tuning)

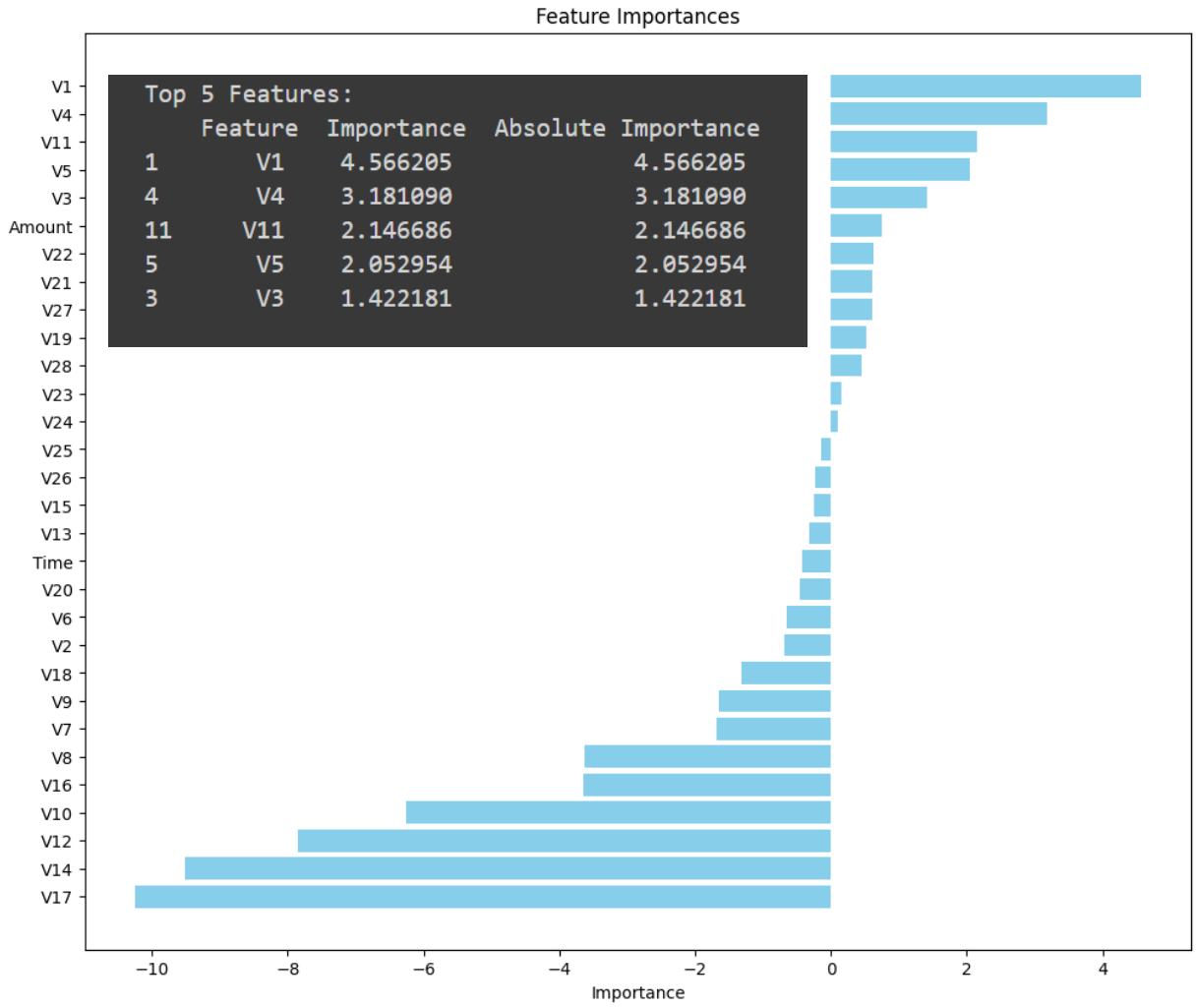


True Negatives (56776): The model correctly predicted the negative class.
False Positives (88): The model incorrectly predicted the positive class when it was actually negative.
False Negatives (13): The model incorrectly predicted the negative class when it was actually positive.
True Positives (85): The model correctly predicted the positive class.

This logistic Regression model is not performing well for Fraud Detection case as the number of False Positives cases (88) are more than True Positives (85). This indicates that the model is predicting more instances as positive when they are actually negative, compared to correctly identifying positive instances.

FEATURE SELECTION

Feature selection is one of the major processes in machine learning. In this process, a subset of relevant features or variables, also known as predictors for model construction, needs to be found.



e) Light Gradient Boosting Machine

After building the model, we tested the model with unseen data (Test data) and the following results were obtained (Fraud Class):

Accuracy: 99.79%

Recall: 86.73%

Precision: 45.21%

F1 – Score: 59%

```
Testing Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.45      0.87      0.59        98

    accuracy                           1.00     56962
   macro avg       0.73      0.93      0.80     56962
weighted avg       1.00      1.00      1.00     56962

Testing Confusion Matrix:
 [[56761   103]
 [   13    85]]
```

Here, we can see, the model is performing moderately on unseen data, however there is scope for improvement. In order to enhance our model performance, we performed **Threshold Tuning.**

thresholds = np.arange(0.1, 1.0, 0.1)

PREDICTION AFTER THRESHOLD TUNING

Class 1: Fraud Transaction metrics

Accuracy: 99.94%

Recall: 81.63%

Precision: 86.02%

F1 – Score: 84%

```
Best threshold based on F1-score: 0.9
Final Testing Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.86      0.82      0.84        98

    accuracy                           1.00     56962
   macro avg       0.93      0.91      0.92     56962
weighted avg       1.00      1.00      1.00     56962
```

The performance of LightGB Model has increased after adjusting the threshold.

RESULT INTEPRETATION:

Class 0: Normal Transaction

Class 1: Fraud Transaction

Precision:

Class 0: 1.00 (100%) – Perfect accuracy in predicting class 0.
Class 1: 0.86 (86%) – 86% of predicted class 1 instances were correct.

Recall:

Class 0: 1.00 (100%) – The model correctly identified all actual class 0 instances.
Class 1: 0.82 (82%) – The model identified 82% of actual class 1 instances.

F1-Score:

Class 0: 1.00 (100%) – Perfect balance for class 0.
Class 1: 0.84 (84%) – Good balance for class 1.

The model performs fairly well for **class 0**, as one would expect given the high support. For class 1, although performing reasonably well (**precision = 0.86** and **recall = 0.82**), one might likely find room for improvement in its prediction capability, especially given that it has relatively low support for class 1.

PREDICTION AFTER HYPERPARAMETER TUNING

Class 1: Fraud Transaction metrics

Accuracy: 99.94%

Recall: 84%

Precision: 87%

F1 – Score: 85%

```
Classification Report for the Best Model:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.87      0.84      0.85        98

    accuracy                           1.00     56962
   macro avg       0.94      0.92      0.93     56962
weighted avg       1.00      1.00      1.00     56962

Best Model ROC AUC Score: 0.9855631194514947
```

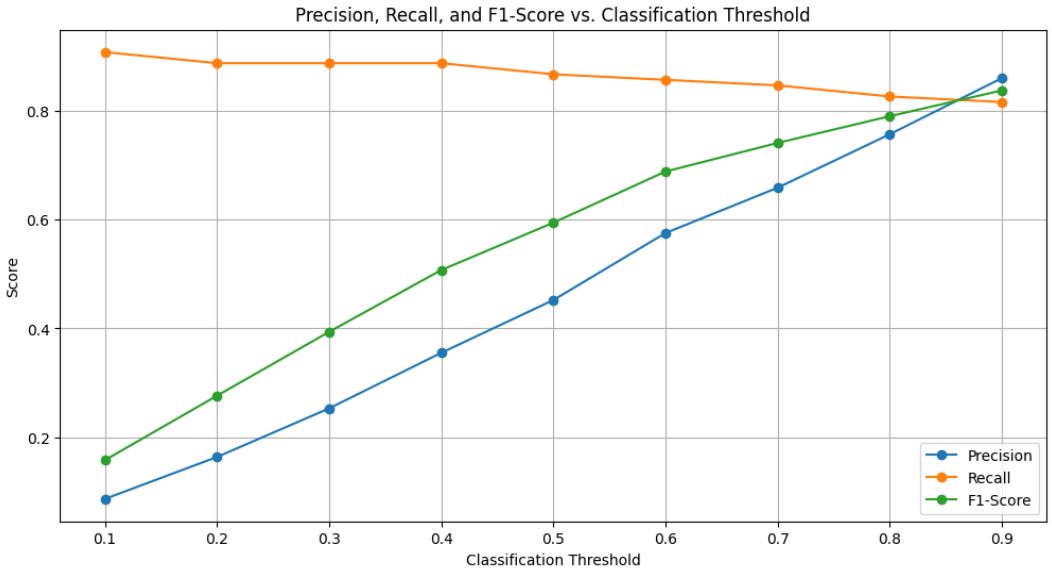
Best Parameters: {'boosting_type': 'gbdt', 'lambda_l1': 0.1, 'learning_rate': 0.1, 'n_estimators': 200, 'num_leaves': 50, 'objective': 'binary'}

As we can predict, the model performed slightly better after hyperparameter tuning. Now we will move ahead with model and plot graphs.

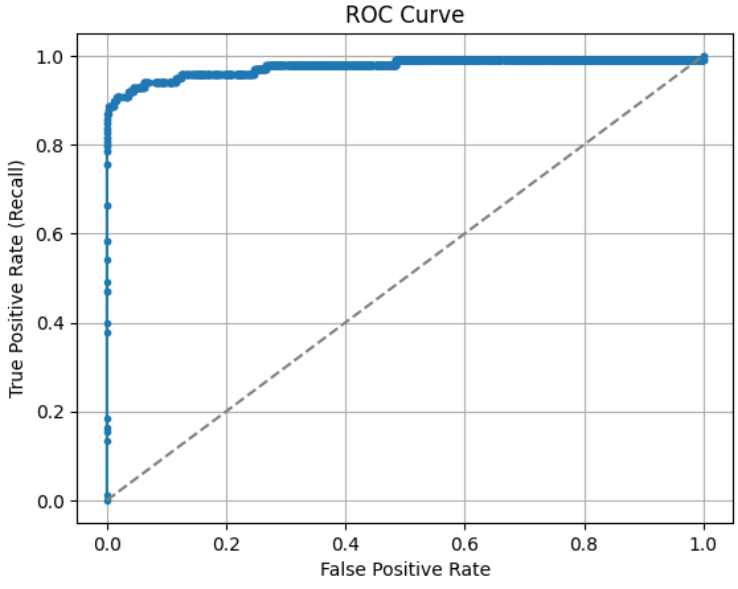
VISUALISATIONS:

1. Precision, Recall, and F1 – Score vs Classification Threshold

Best threshold value based on F1 – score: 0.9

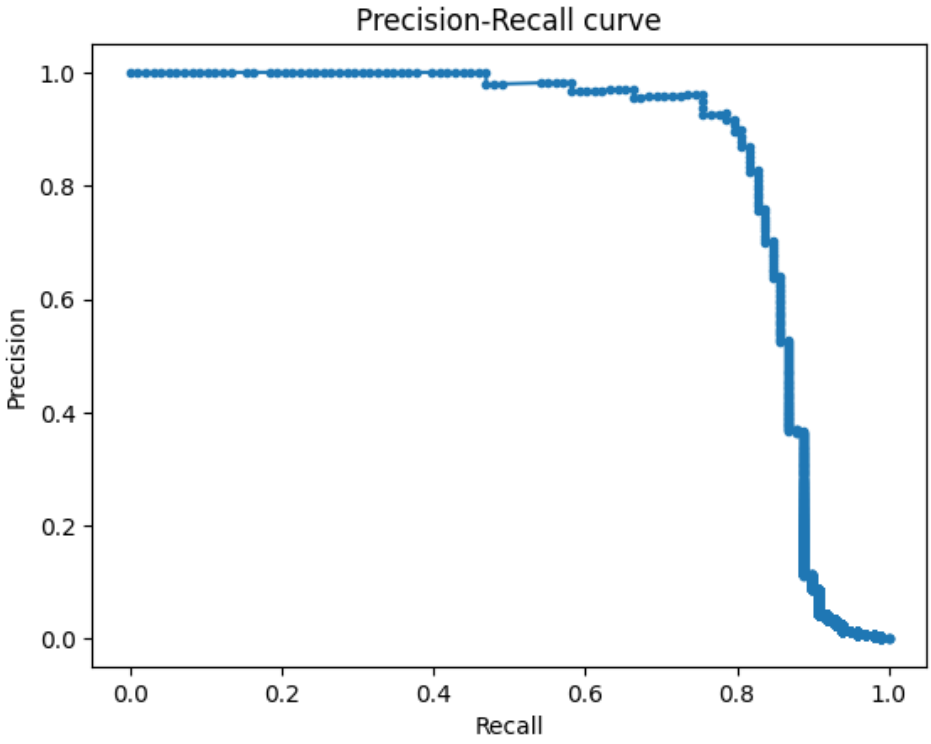


2. ROC Curve

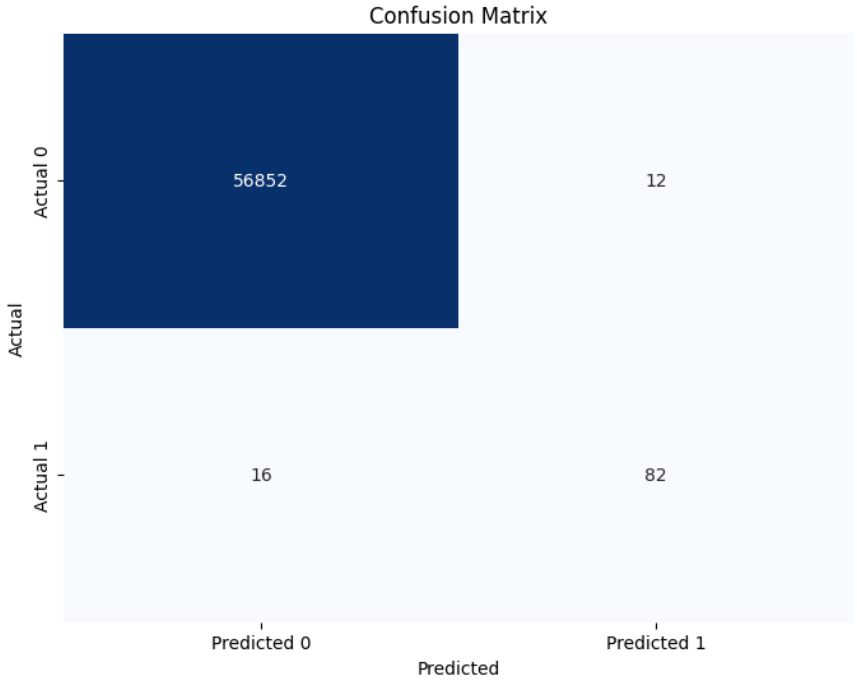


3. Precision - Recall Curve

ROC AUC Score: 0.975303588655496 (model is excellent at distinguishing between the positive and negative classes)



4. Confusion Matrix (after Hyperparameter Tuning)

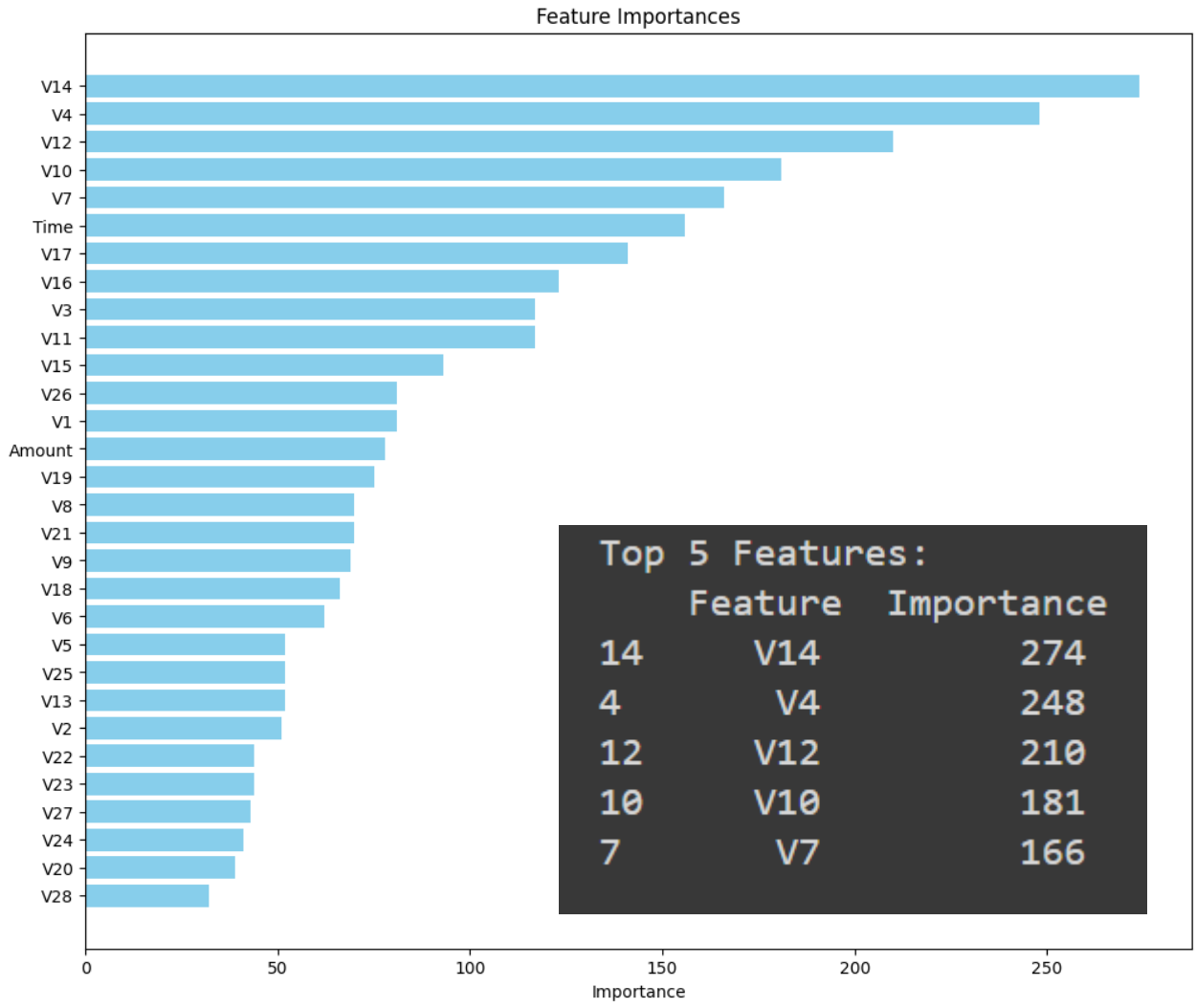


True Negatives (56852): The model correctly predicted the negative class.
False Positives (12): The model incorrectly predicted the positive class when it was actually negative.
False Negatives (16): The model incorrectly predicted the negative class when it was actually positive.
True Positives (82): The model correctly predicted the positive class.

In summary, this confusion matrix would reflect very high numbers of true negatives and true positives but relatively much smaller numbers for the false positives and false negatives, which might indicate that it is doing pretty well in telling the difference between the two classes.

FEATURE SELECTION

Feature selection is one of the major processes in machine learning. In this process, a subset of relevant features or variables, also known as predictors for model construction, needs to be found.



OVERALL SUMMARY

MODEL NAME	ACCURACY	PRECISION	RECALL	F1 – SCORE
Xgboost	99.96%	0.93	0.84	0.88
Random Forest Classifier	99.96%	0.97	0.80	0.88
Extra Trees Classifier	99.95%	0.94	0.79	0.86
Logistic Regression	99.94%	0.49	0.87	0.63
Light GBM	99.94%	0.87	0.84	0.85

When dealing with imbalanced data, we should NOT be solely depending on Accuracy, as high accuracy can be misleading in such cases. It's often advisable to consider additional metrics such as: Precision, Recall, F1 – Score and ROC value.

- The best model among these 5 would be **Randon Forest Classifier Model.** With high test accuracy (99.96%), it also has high precision (97%) and good Recall (80%) and F1 – Score (88%).

- The least fit model among these 5, for this project will be **Logistic Regression**. Though Logistic Regression is having high accuracy (99.94%), but it has precision less than 50%. The model correctly identifies 87% of the actual positive instances that are within Class 1, which is a good recall score. The problem, though is precision. This means some of those predictions cannot be relied upon.

RANKING THE MODELS

1. Random Forest Classifier
2. Xgboost
3. Extra Trees Classifier
4. Light GBM
5. Logistic Regression

3. DISCUSSION OF FUTURE WORK

3.1. Potential Improvements

Advanced Algorithms: Dive deeper into ensemble methods in hopes of improving detection adequacy.

Feature Engineering: Construction of new features such as time-based features or transaction patterns from the same data could yield a greater context for the model.

Anomaly Detection: The key would be in applying unsupervised learning techniques to find outliers or unusual patterns that are perhaps indicative of fraud without being given structured data.

Deep Learning: Similar to the previous point, explore how deep learning methods can suitably discover non-linear relationships in the data.