UNIVERSITY OF CAPE COAST
SCHOOL OF PHYSICAL SCIENCE
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

COURSE NAME: **INTRODUCTION TO INTELLIGENT SYSTEMS**
COURSE CODE: **INF402**
LECTURER'S NAME: **Dr. PATRICK**
STUDENT'S NAME: **ANYIZAH KORBLA WISDOM**
INDEX NO.: **PS/ITC/20/0057**

ASSIGNMENT 1, Project 2.

Fraud Detection Using AI

Developing a Machine Learning Model for Detecting Fraudulent Transactions, and its Deployment

Introduction

- Importance of fraud detection in the financial sector.
- Objective: Develop a machine learning model for fraud detection.

Objective

- Develop a machine learning model to detect fraudulent transactions.
- Use the "Credit Card Fraud Detection" dataset from Kaggle.

Dataset Description

- Overview of the dataset: transactions made by European cardholders in September 2013.
- Total transactions: 284,807
- Fraudulent transactions: 492
- Features: 30
- Imbalance data

Tools and Libraries

- Python
- pandas, numpy, scikit-learn, matplotlib, seaborn, imblearn
- Jupyter Notebook
- Flask
- joblib

Methodology

- Data Collection
- Data Preprocessing
- Exploratory Data Analysis (EDA)
- Feature Engineering
- Model Selection
- Model Training and Evaluation
- Model Tuning
- Model Deployment

Data Collection and Preprocessing

- Loaded dataset into pandas DataFrame.
- Checked for null values and basic statistics.
- Scaled 'Amount' and 'Time' features.

Exploratory Data Analysis (EDA)

- Visualized data distribution using histograms, line, bar and box plots.
- Correlation analysis using heatmap.

Feature Engineering

- No additional features created.
- Used existing features in the dataset.

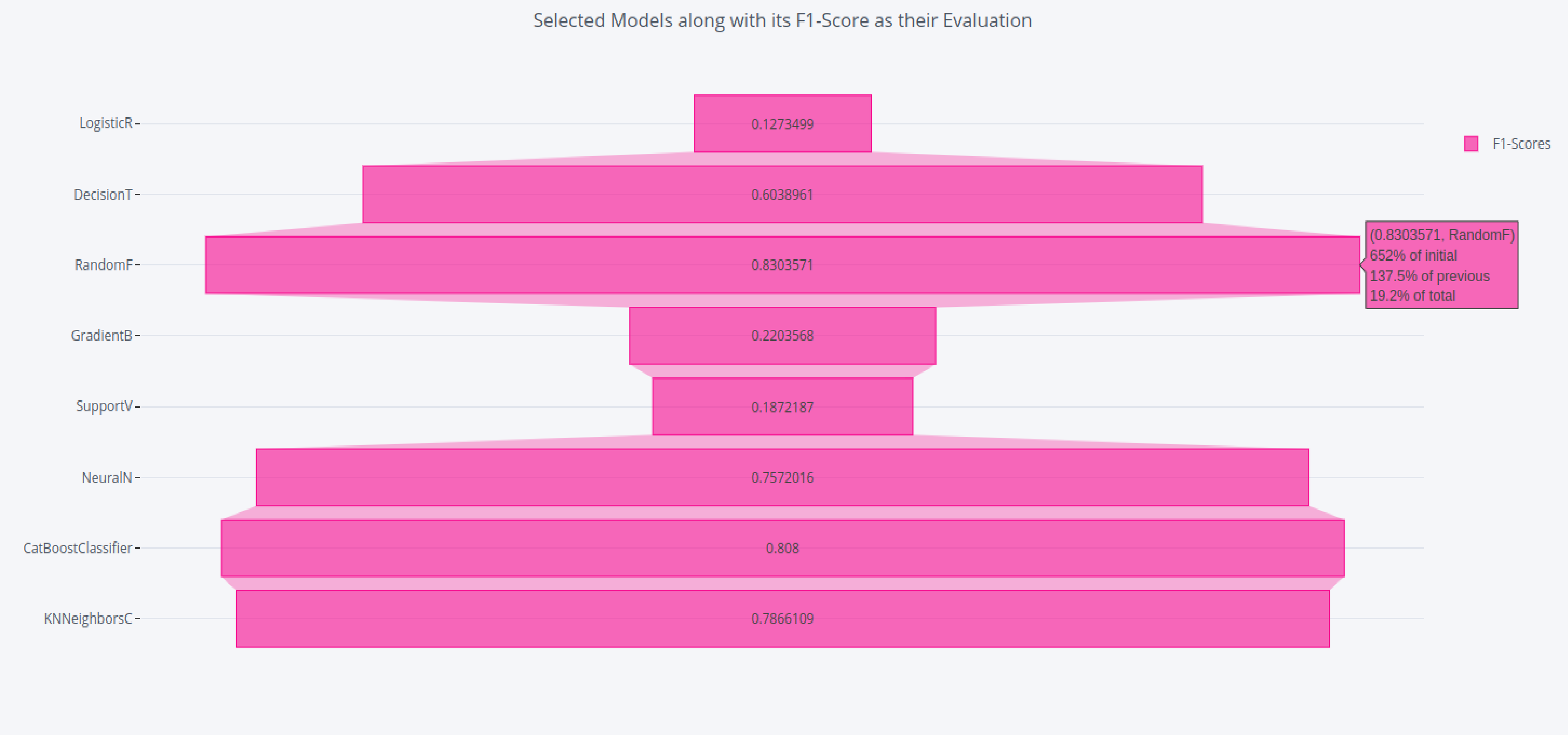
Model Selection

- Models: Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, Neural Network, Support Vector Classifier (SVC), CatBoostClassifier, and K-Nearest Neighbors Classifier.
- Split data into training and testing sets.

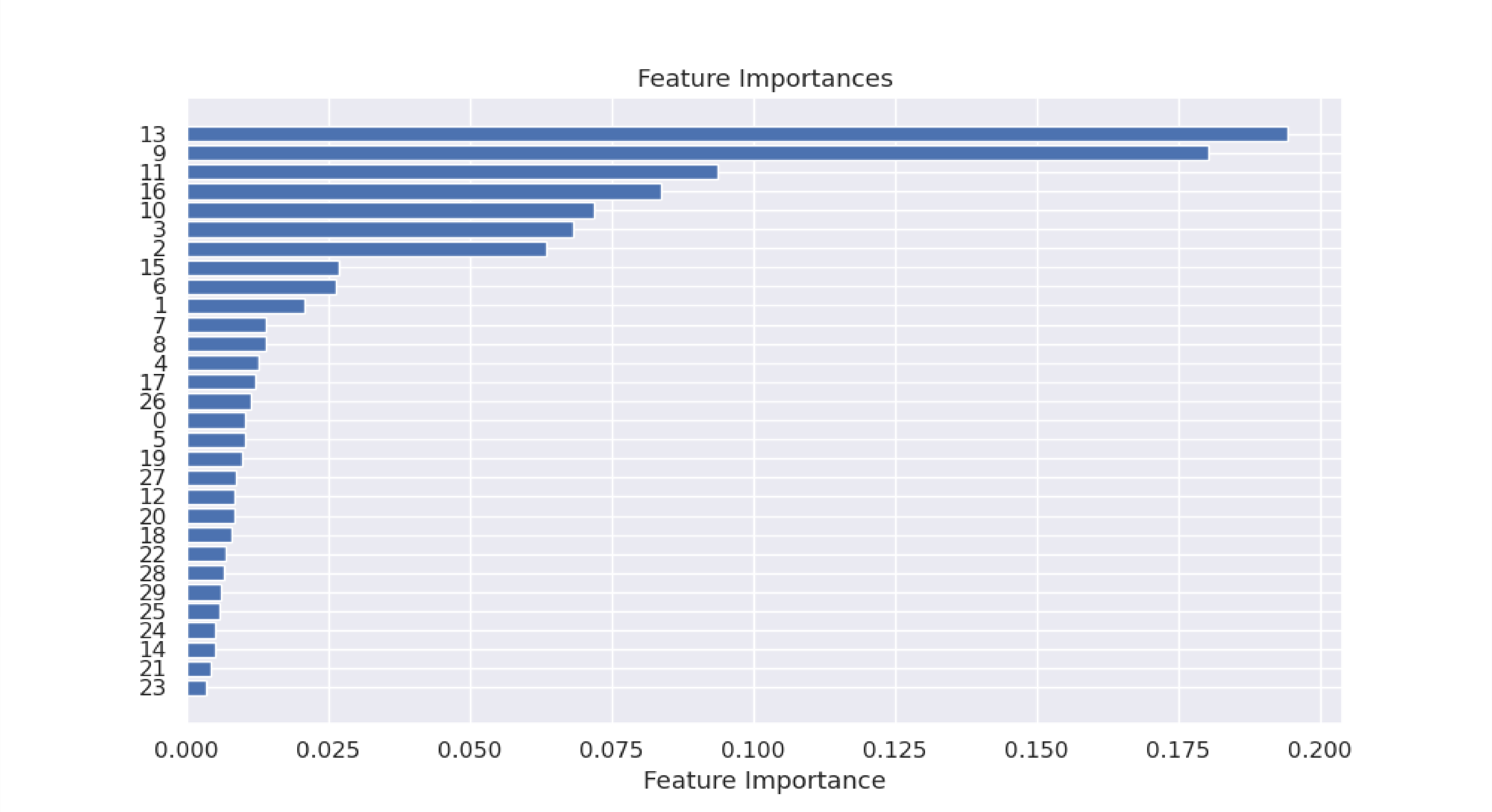
Model Training and Evaluation

- Trained, predict, and evaluated models using metrics: accuracy, precision, recall, F1-score, AUC-ROC, and Confusion Matrix.
- Calculate F1-Scores using precision and recall.

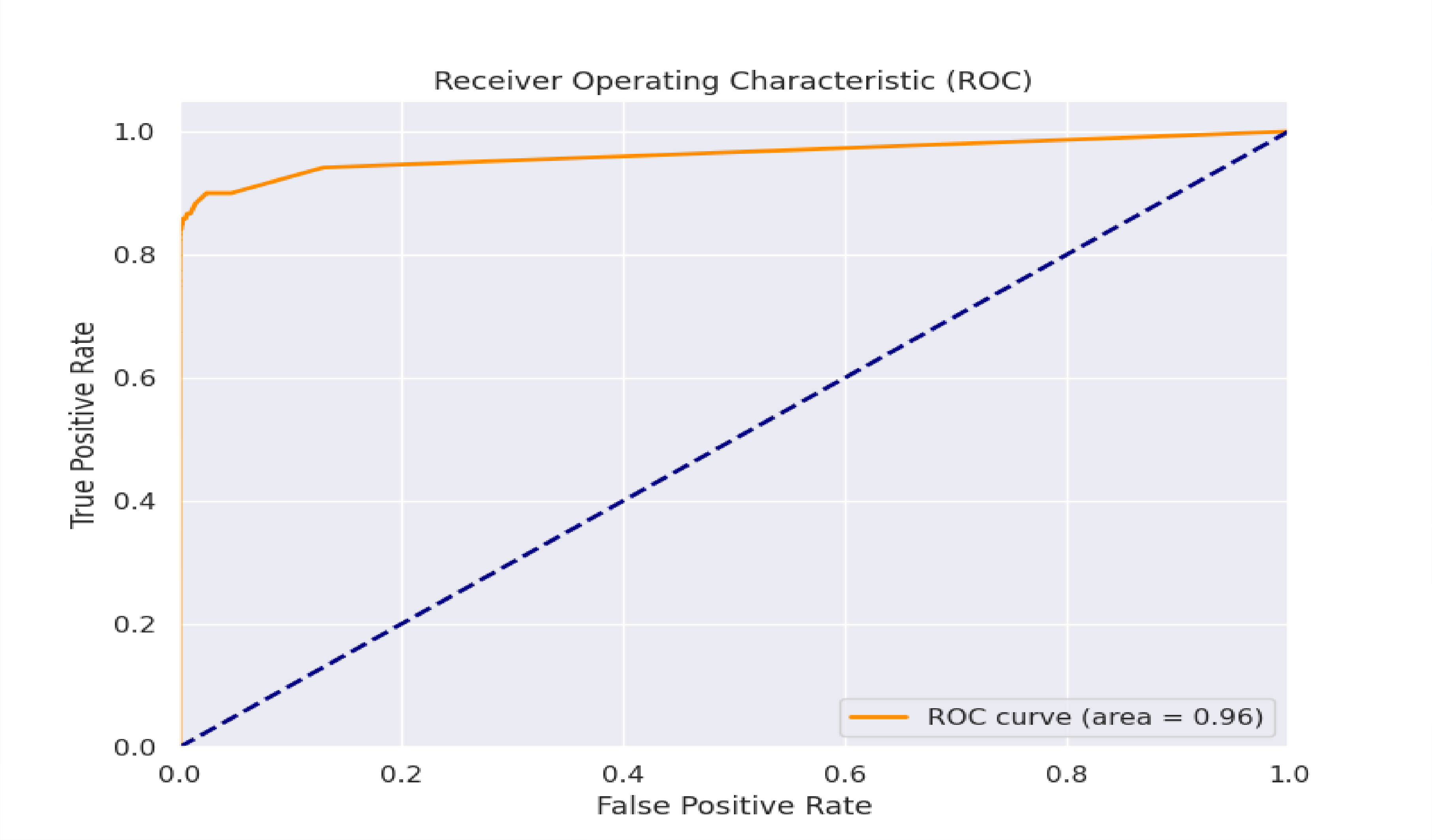
Model Training and Evaluation



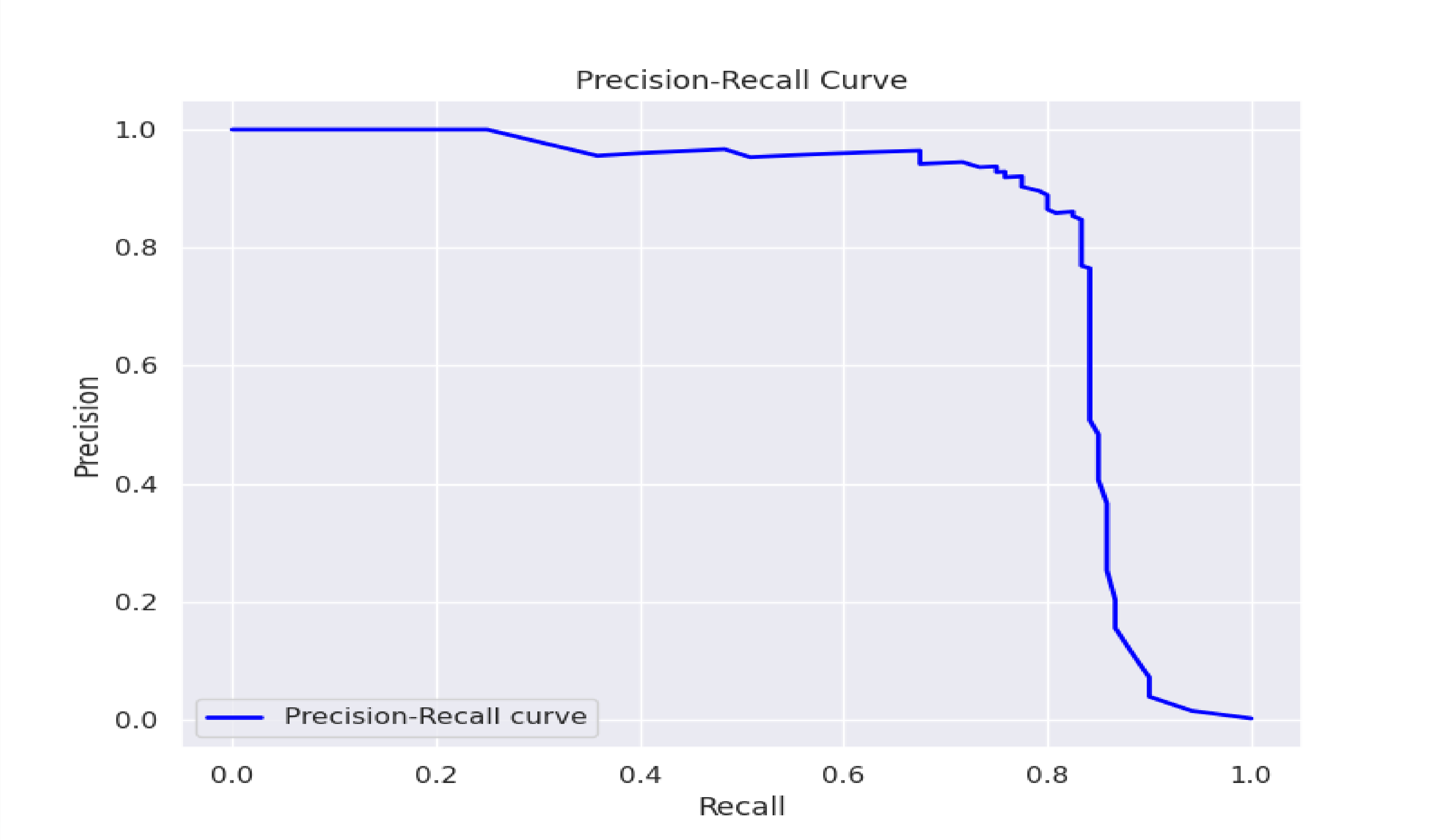
Feature Importances of the Untuned Random Forest Classifier



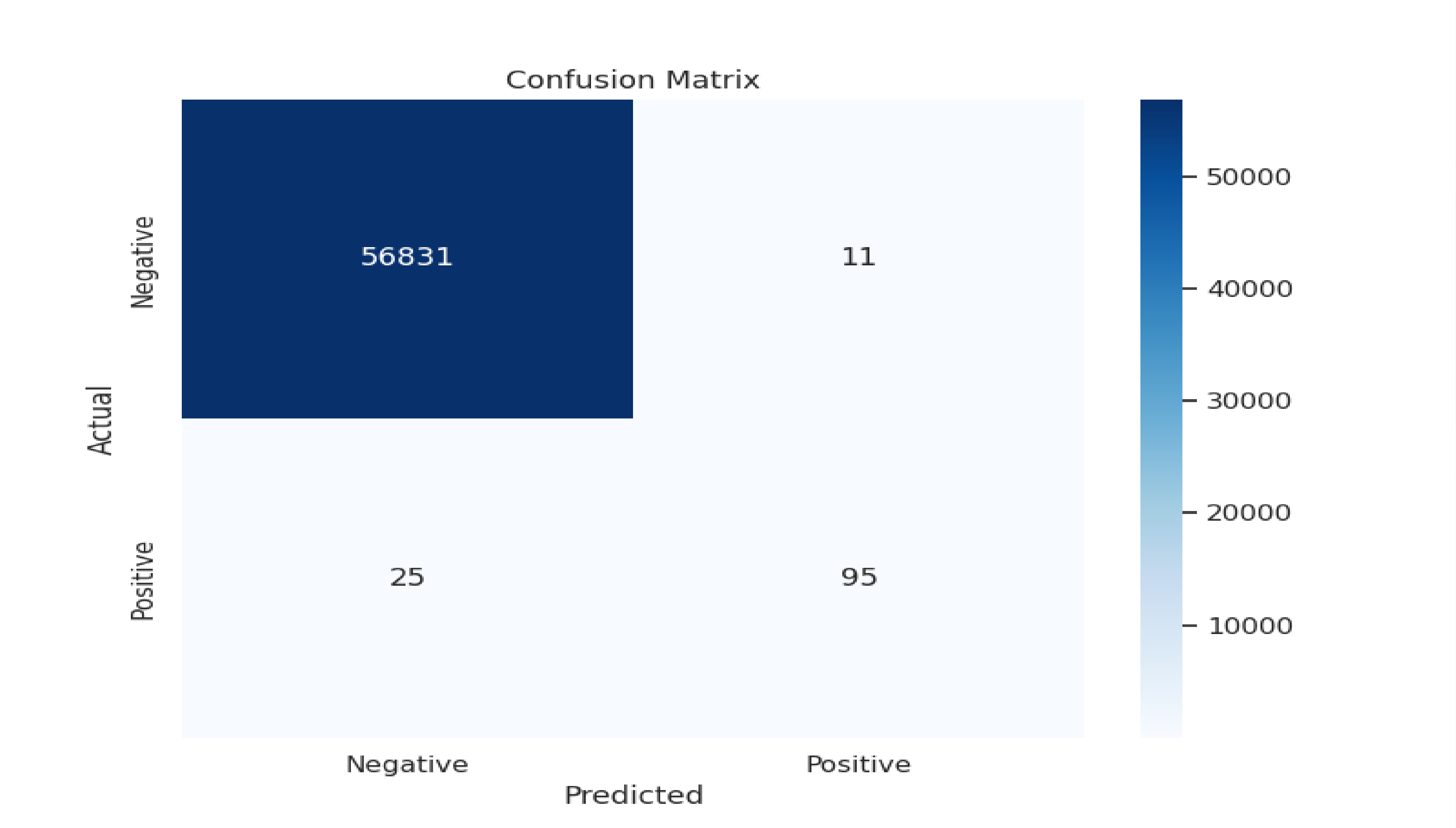
AUC-ROC curve of Best Model (Untuned RFC)
Random Forest Classifier



Precision, and Recall of Random Forest Classifer (Best Model)



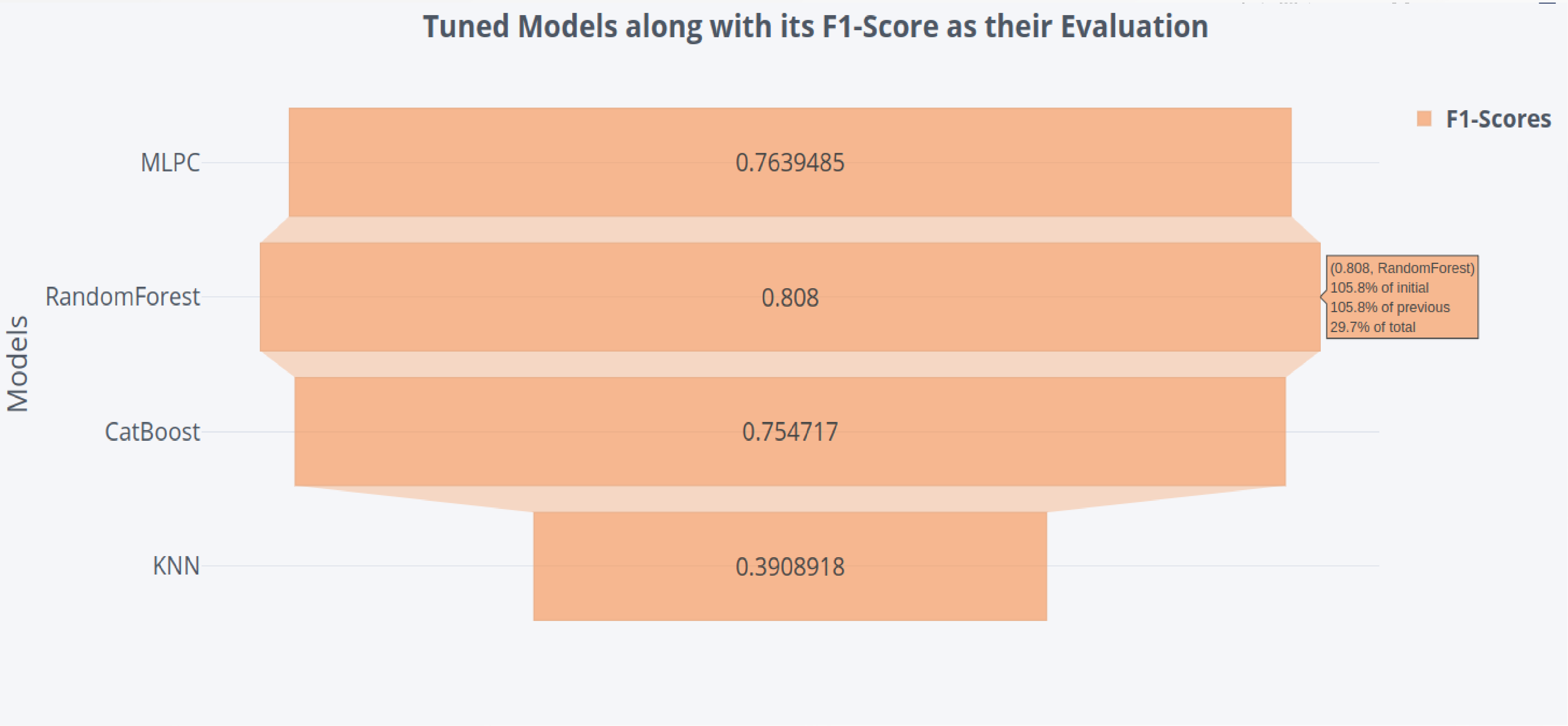
Confusion Matrix Evaluation best untuned model (RandomForestClassifier)



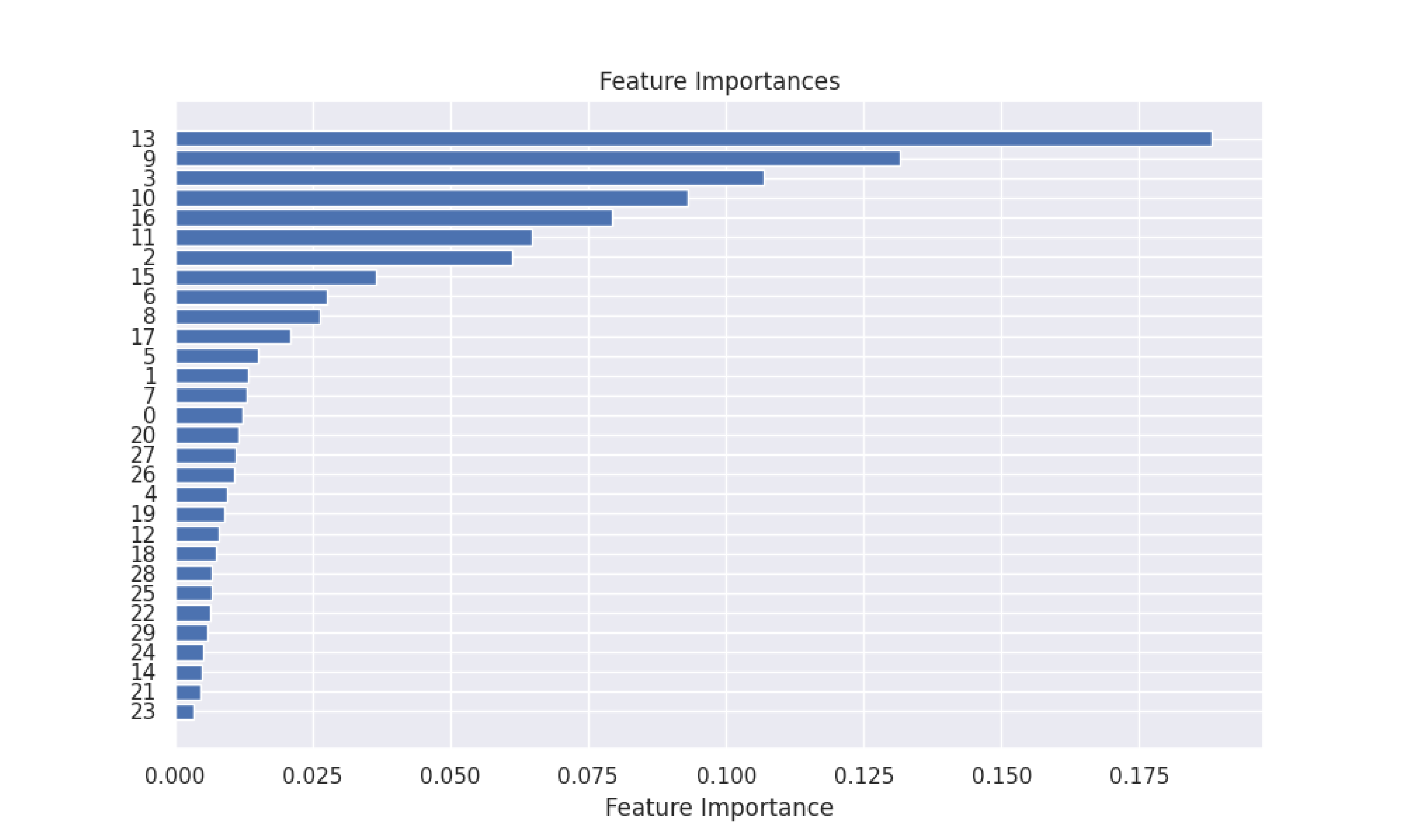
Model Tuning

- Hyperparameter optimization using Random SearchCV.
- Handled class imbalance with SMOTE.

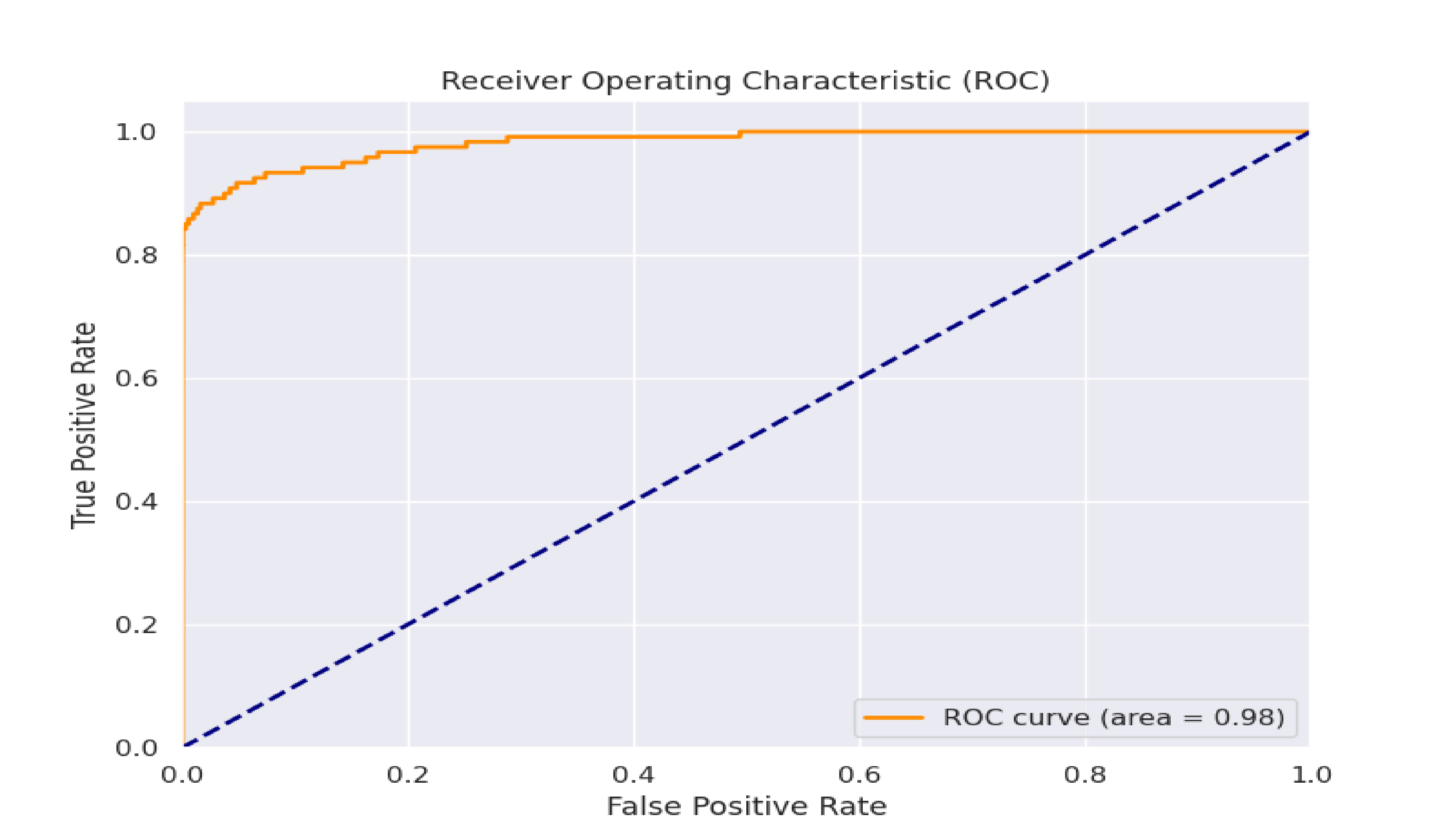
Tuned Model Hyperparameters Training and Evaluation



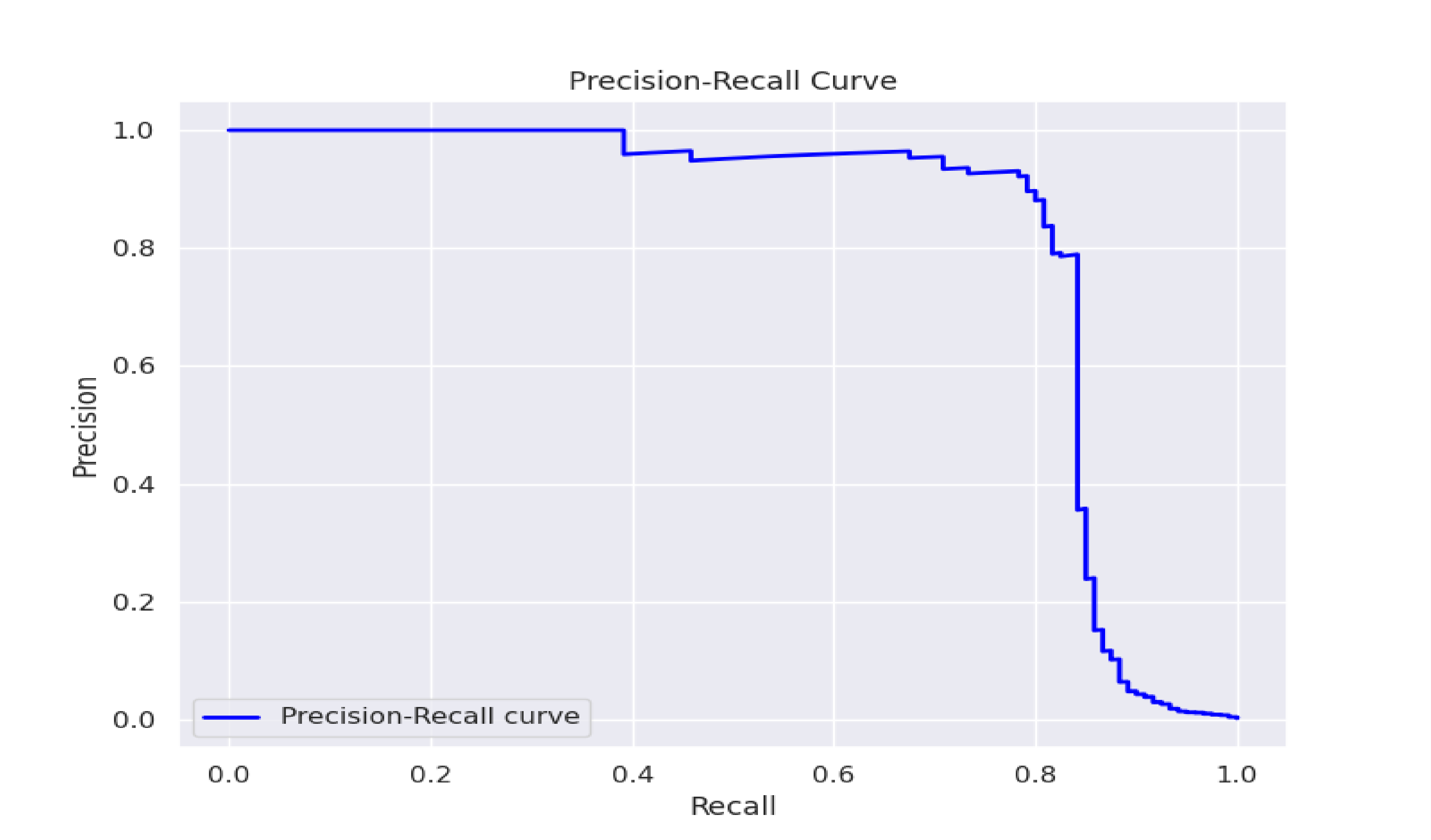
Features Importance of Tuned Random Forest Classifier



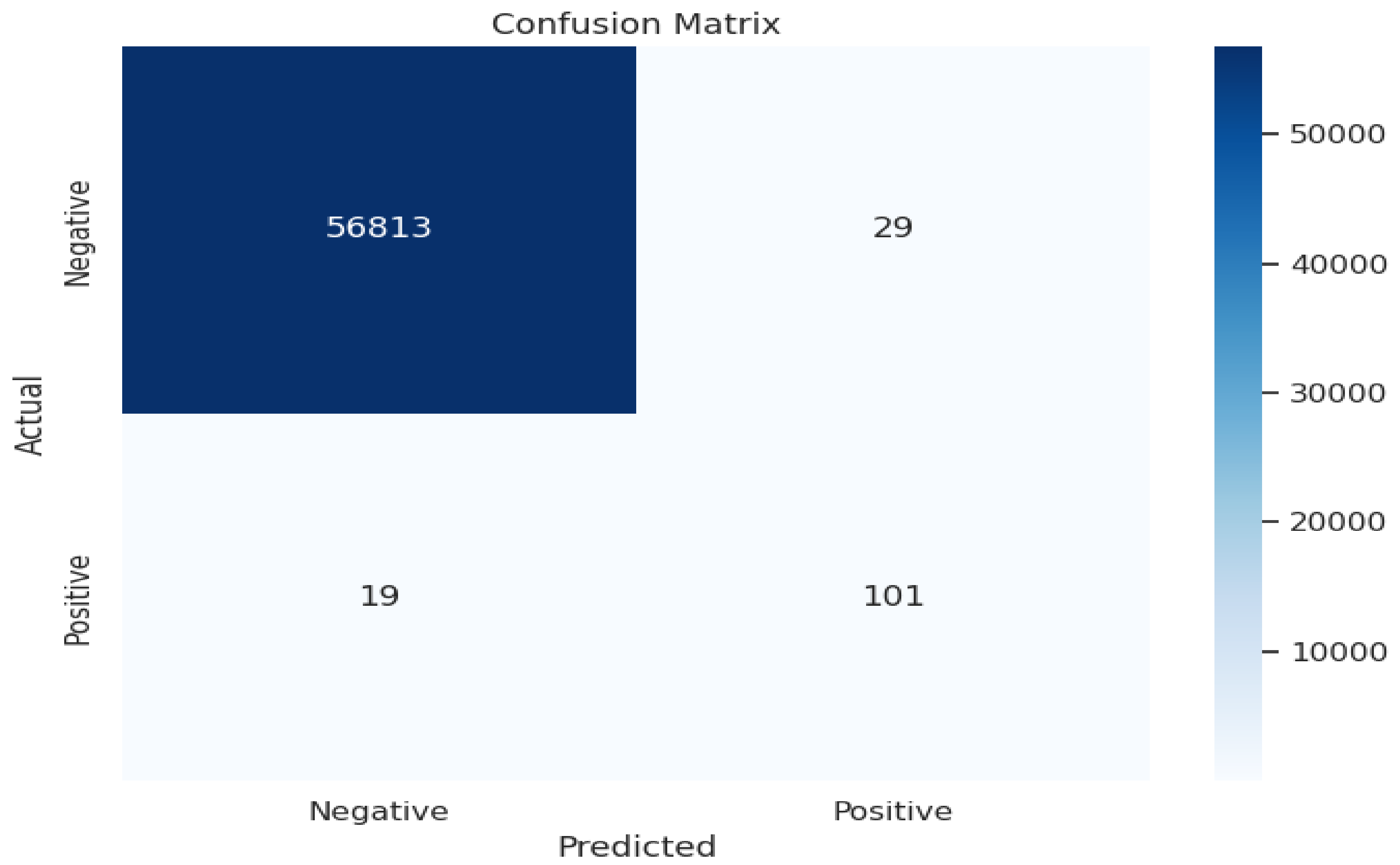
AUC-ROC curve of Best Model (Tuned RFC)



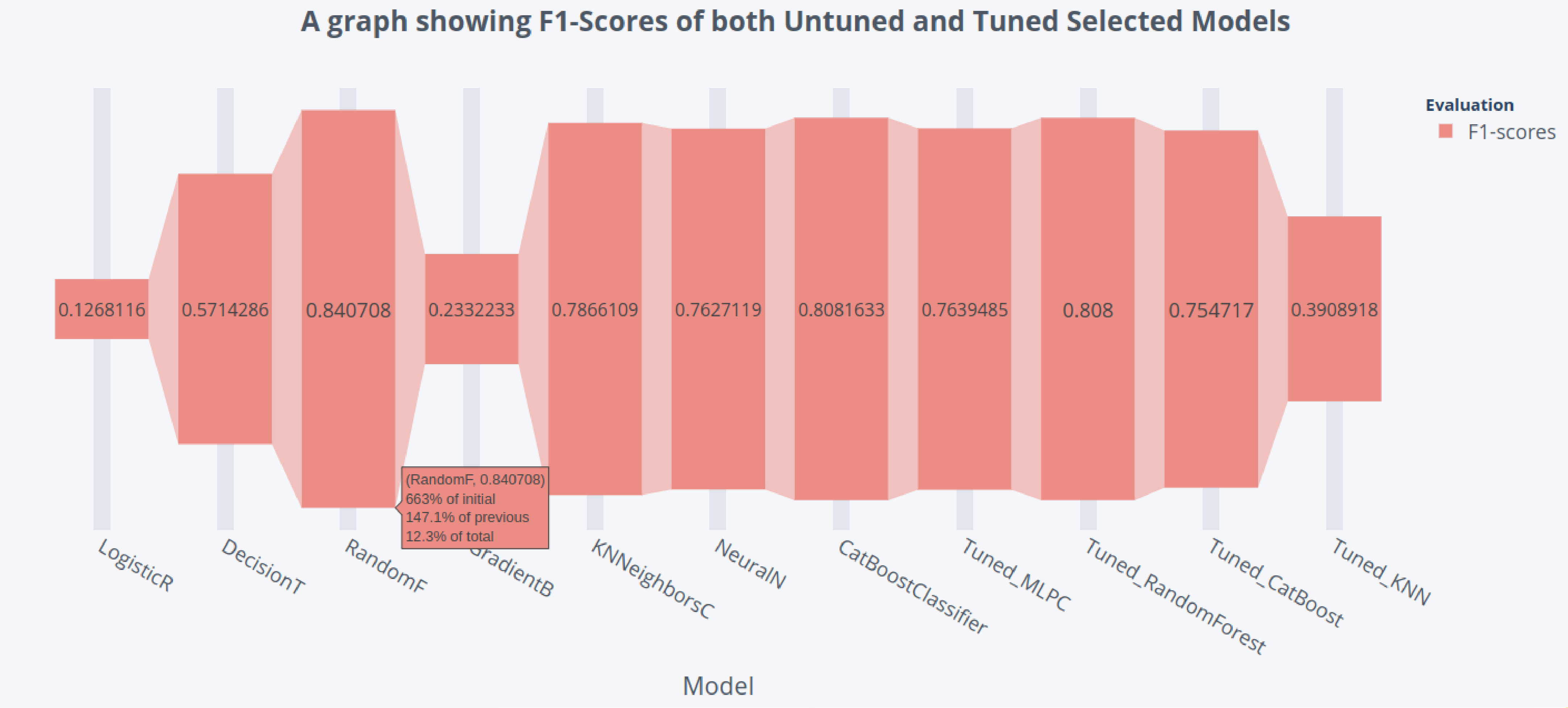
Precision, and Recall of the Best Model (RFC)



Confusion Matrix Evaluation best tuned model (RandomForestClassifier)



Models Untuned and Tuned Training and Evaluation (F1-Scores)



F-Scores is used to evaluate the models performance

The F1 score is a performance metric used to evaluate the effectiveness of a classification model, especially in situations where you have imbalanced classes. It is the harmonic mean of precision and recall, providing a balance between these two metrics. Here's a detailed explanation:

F1_Scores Evaluation Definition:
Precision: Precision measures the accuracy of the positive predictions. It is the ratio of true positive predictions to the total number of positive predictions (both true positives and false positives). In formula terms: [{Precision} = {TP}/{TP + FP}] where:

(TP) = True Positives (correctly predicted positive cases)
(FP) = False Positives (incorrectly predicted as positive)
Recall (Sensitivity): Recall measures the ability of the model to identify all relevant instances. It is the ratio of true positive predictions to the total number of actual positive cases (both true positives and false negatives). In formula terms: [{Recall} = {TP}/{TP + FN}] where:

(TP) = True Positives
(FN) = False Negatives (actual positive cases missed by the model)
F1 Score Calculation
The F1 score combines precision and recall into a single metric by taking their harmonic mean. It is useful when you need a single measure to evaluate the performance of a model in scenarios where both false positives and false negatives are important.

The formula for the F1 score is: F1-Score = 2 x [(Precision x Recall)/(Precision+Recall)]

Cont’d

Why Use F1 Score?

Balance: F1 score provides a balance between precision and recall. It is particularly valuable when you need to balance the trade-off between false positives, and false negatives.
Imbalanced Datasets: In cases where the classes are imbalanced (e.g., detecting fraud where fraudulent transactions are much less frequent than non-fraudulent ones), accuracy alone can be misleading. The F1 score gives a better visual of the model’s performance on the minority class.
Interpretation:
High F1 Score: A high F1 score indicates that the model has both high precision and high recall. It means that the model is correctly identifying positive cases while minimizing false positives and false negatives.
Low F1 Score: A low F1 score suggests that either precision, recall, or both are low. This means the model is struggling to balance the trade-off between false positives and false negatives.
Example:
Consider a binary classification problem where you have the following results from Tuned Random Forest Classifier:

True Positives (TP): 101
False Positives (FP): 19
False Negatives (FN): 29

Calculate Precision and Recall:

Precision Precision= TP/(TP+FP)
Precision=101/(101+19)
Precision=101/120
Precision ≈ 0.8417

Cont’d

Recall Recall=TP/(TP+FN)
Recall=101/(101+29)
Recall=101/130
Recall≈0.7769

F1 Score The F1 score is the harmonic mean of Precision and Recall.

F1 Score=2 * [(Precision * Recall)/(Precision+Recall)]

F1 Score=2 * [(0.8417 * 0.7769)/(0.8417+0.7769)]
F1 Score=2 * (0.6542/1.6186)

F1 Score≈2 * 0.404
F1 Score≈0.808

So, the F1 Score is approximately 0.808.

Model Deployment

- Best model (MLPC) and (Random Forest Classifier) saved using joblib.
- Developed Flask API for model predictions.

Results and Discussion

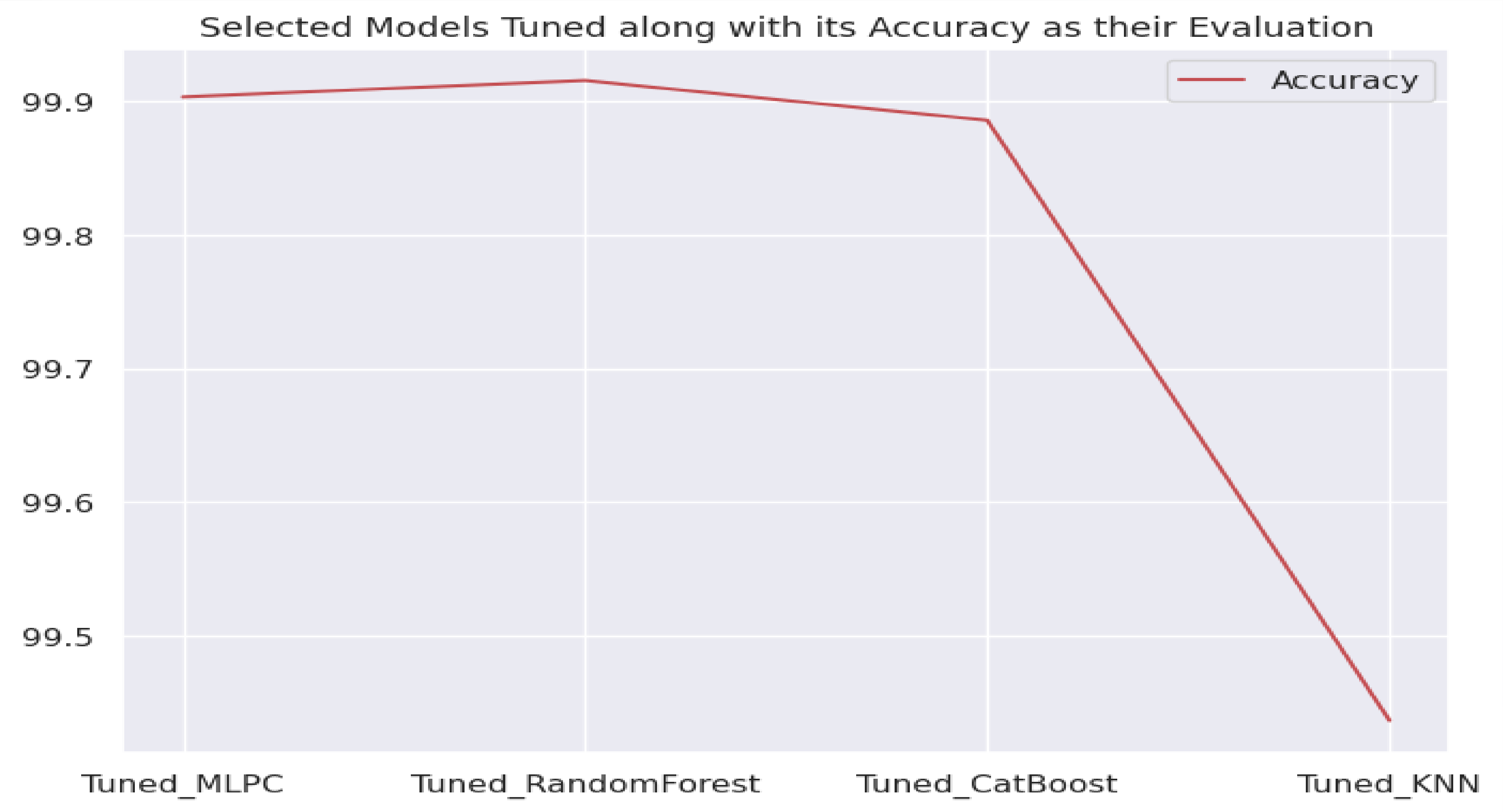
- Random Forest Classifier model performance: accuracy, precision, recall, F1-score, AUC-ROC.

Which are:

1. Accuracy: **0.9991573329588147**
2. Precision: **0.7769230769230769**
3. Recall: **0.8416666666666667**
4. F1 Score: **0.808**
5. AUC-ROC: **0.9205782402683462**

- Improvements from tuning and class imbalance handling, which also prevented overfitting of the model.

Accuracy of Tuned Models



Running Python server to connect deployed model, and Testing API

1. Open a terminal
2. cd path_to/run.py (Change directory to the work directory)
3. Run “python run.py” in terminal to start the server at port 5000
4. Copy and paste below API test code (one after the order to get the prediction for each, you can change its values) to a new terminal.

API for test 1:
curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d '{ "V1": 0.1, "V2": 0.2, "V3": -0.1, "V4": 1.2, "V5": 0.3, "V6": -1.0, "V7": 0.7, "V8": 0.8, "V9": -0.4, "V10": 0.6, "V11": -0.3, "V12": 0.4, "V13": 0.5, "V14": -0.2, "V15": 1.0, "V16": -0.7, "V17": 0.9, "V18": 0.2, "V19": -1.1, "V20": 0.0, "V21": 0.3, "V22": -0.9, "V23": 0.8, "V24": -0.5, "V25": 0.4, "V26": 0.1, "V27": -0.3, "V28": 0.6, "scaled_amount": 150.0, "scaled_time": 100.0 }'

API for test 2:
curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d '{ "V1": -2.312227, "V2": 1.951992, "V3": -1.609851, "V4": 3.997906, "V5": -0.522188, "V6": -1.426545, "V7": -2.537387, "V8": 1.391657, "V9": -2.770089, "V10": -2.772272, "V11": 3.202033, "V12": -2.899907, "V13": -0.595222, "V14": -4.289254, "V15": 0.389724, "V16": -1.140747, "V17": -2.830056, "V18": -0.016822, "V19": 0.416956, "V20": 0.126911, "V21": 0.517232, "V22": -0.035049, "V23": -0.465211, "V24": 0.320198, "V25": 0.044519, "V26": 0.177840, "V27": 0.261145, "V28": -0.143276, "scaled_amount": 0.00, "scaled_time": 406.0 }'

Running the Python (Flask) Server

```
yesulikplimits@wittymaLab: /media/yesulikplimits/@PERSISTENT_LEARNING/MyF...   yesulikplimits@wittymaLab: /media/yesulikplimits/@PERSISTENT_LEARNING/MyFo...   yesulikplimits@wittymaLab: /media/yesulikplimits/@PERSISTENT_LEARNING/MyFo...
(base) yesulikplimits@wittymaLab:/media/yesulikplimits/@PERSISTENT_LEARNING/MyFolder/AI_Project_L400/DA$ python run.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (inotify)
 * Debugger is active!
 * Debugger PIN: 318-406-254
/home/yesulikplimits/anaconda3/lib/python3.11/site-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
127.0.0.1 - - [28/Jul/2024 19:44:19] "POST /predict HTTP/1.1" 200 -
/home/yesulikplimits/anaconda3/lib/python3.11/site-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
127.0.0.1 - - [28/Jul/2024 19:44:40] "POST /predict HTTP/1.1" 200 -
```

API Testing and its prediction

yesulikplimits@wittymaLab: /media/yesulikplimits/@PERSISTENT_LEARNING/MyF... yesulikplimits@wittymaLab: /media/yesulikplimits/@PERSISTENT_LEARNING/MyFo... yesulikplimits@wittymaLab: /media/yesulikplimits/@PERSISTENT_LEARNING/MyFo...

```
(base) yesulikplimits@wittymaLab:/media/yesulikplimits/@PERSISTENT_LEARNING/MyFolder/AI_Project_L400/DA$ curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d '{
"V1": 0.1, "V2": 0.2, "V3": -0.1, "V4": 1.2, "V5": 0.3, "V6": -1.0, "V7": 0.7, "V8": 0.8, "V9": -0.4, "V10": 0.6, "V11": -0.3, "V12": 0.4, "V13": 0.5, "V14": -0.2, "V15": 1.0, "V16": -0.7, "
V17": 0.9, "V18": 0.2, "V19": -1.1, "V20": 0.0, "V21": 0.3, "V22": -0.9, "V23": 0.8, "V24": -0.5, "V25": 0.4, "V26": 0.1, "V27": -0.3, "V28": 0.6, "scaled_amount": 150.0, "scaled_time": 100.
0 }'

{
  "prediction": "0; not fraud"
}
(base) yesulikplimits@wittymaLab:/media/yesulikplimits/@PERSISTENT_LEARNING/MyFolder/AI_Project_L400/DA$ curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d '{
"V1": -2.312227, "V2": 1.951992, "V3": -1.609851, "V4": 3.997906, "V5": -0.522188, "V6": -1.426545, "V7": -2.537387, "V8": 1.391657, "V9": -2.770089, "V10": -2.772272, "V11": 3.202033, "V12"
: -2.899907, "V13": -0.595222, "V14": -4.289254, "V15": 0.389724, "V16": -1.140747, "V17": -2.830056, "V18": -0.016822, "V19": 0.416956, "V20": 0.126911, "V21": 0.517232, "V22": -0.035049, "
V23": -0.465211, "V24": 0.320198, "V25": 0.044519, "V26": 0.177840, "V27": 0.261145, "V28": -0.143276, "scaled_amount": 0.00, "scaled_time": 406.0 }'
{
  "prediction": "1; fraud"
}
(base) yesulikplimits@wittymaLab:/media/yesulikplimits/@PERSISTENT_LEARNING/MyFolder/AI_Project_L400/DA$
```

Conclusion

- Successful development of a fraud detection model.
- Random Forest Classifer model provided the best performance with highest F1-score and Accuracy,
- Potential for real-world deployment.

Future Works

- Incorporate additional features.
- Explore advanced deep learning models.
- Implement real-time data processing and deployment.

References

- **Kaggle dataset**: *https://www.kaggle.com/mlg-ulb/creditcardfraud*
- **Scikit-learn documentation**: *https://scikit-learn.org/stable/documentation.html*
- **Flask documentation**: *https://flask.palletsprojects.com*
- **Logistic Regression**: *https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html*
- **Gradient Boosting**: *https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html*
- **Decision Tree**: *https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html*
- **Support Vector Classifier**: *https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html*
- **MLP Classifier**: *https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html*
- **CatBoost Classifier**: *https://catboost.ai/docs/concepts/python-reference.html#catboostclassifier*
- **Random Forest Classifier**: *https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html*
- ***RandomizedSearchCV :*** *https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html*
- ***K-Nearest Neighbors Classifier** :* *https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html*