MAJOR PROJECT REPORT

Online Payments Fraud Detection with Machine Learning

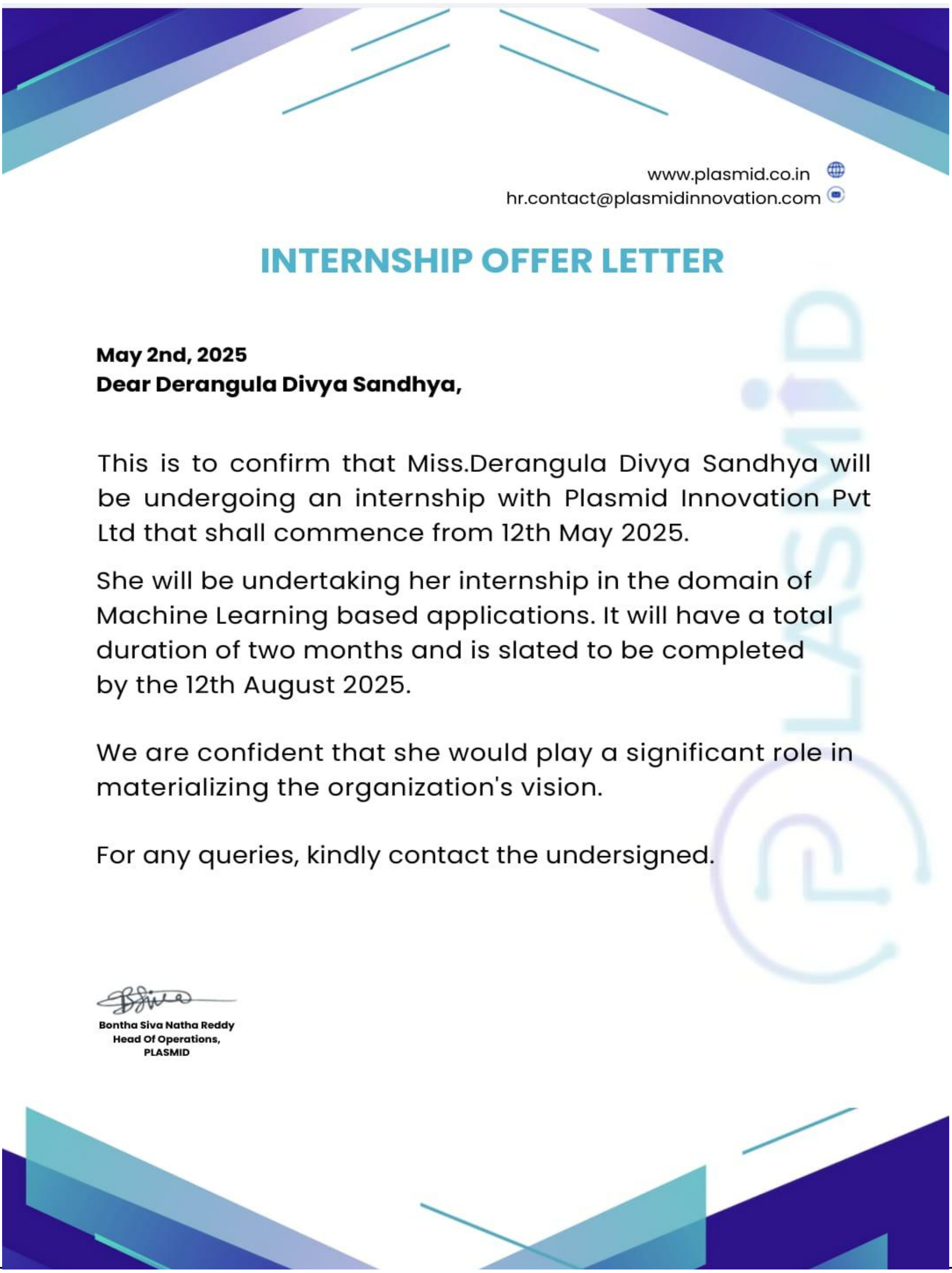
Submitted to:

Plasmid Innovations
(Machine Learning Domain)

Submitted by:

Derangula Divya Sandhya

OFFER LETTER



www.plasmid.co.in
hr.contact@plasmidinnovation.com

INTERNSHIP OFFER LETTER

May 2nd, 2025
Dear Derangula Divya Sandhya,

This is to confirm that Miss.Derangula Divya Sandhya will be undergoing an internship with Plasmid Innovation Pvt Ltd that shall commence from 12th May 2025.

She will be undertaking her internship in the domain of Machine Learning based applications. It will have a total duration of two months and is slated to be completed by the 12th August 2025.

We are confident that she would play a significant role in materializing the organization's vision.

For any queries, kindly contact the undersigned.

[illegible]

Bontha Siva Natha Reddy
Head Of Operations,
PLASMID

Table of Contents

- Abstract
- Introduction
- Problem Statement
- Objective
- Dataset Overview
- Data Preprocessing
- Exploratory Data Analysis
- Feature Engineering
- Model Selection
- Model Training
- Model Evaluation
- Results and Discussion
- Future Work
- Final Python Code
- Output Screenshots
- Conclusion

1. Abstract

In today's digitally driven world, the volume of online financial transactions has increased exponentially, accompanied by a significant rise in fraudulent activities. As a result, ensuring the security and integrity of digital payments has become a critical concern for financial institutions and users alike. Traditional rule-based systems often lack the flexibility to detect evolving fraud patterns, which highlights the need for more dynamic and intelligent solutions.

This project focuses on developing a machine learning-based model to detect online payment frauds by leveraging historical transaction data. Using a publicly available dataset from Kaggle, we explore the various characteristics of financial transactions, including transaction types, amounts, and account balance movements before and after each transaction. Through comprehensive data preprocessing, visualization, and feature analysis, we aim to extract meaningful patterns that help in identifying potentially fraudulent transactions.

Our approach includes training multiple machine learning models and comparing their performance, with Random Forest emerging as the most effective. The final model demonstrates high accuracy, precision, and recall in distinguishing between fraudulent and legitimate transactions. This project not only underscores the importance of data-driven fraud detection systems but also lays the foundation for deploying such models in real-world financial ecosystems.

2. Introduction

The rapid increase in digital payments across the globe has led to a parallel rise in fraudulent activities. With transactions occurring in real-time and at a large scale, traditional fraud detection mechanisms often fall short in identifying sophisticated patterns. Financial institutions require smarter and faster systems to analyze vast amounts of transactional data and flag potentially fraudulent activity instantly.

Machine Learning offers a robust solution to this challenge. Unlike static rule-based systems, machine learning algorithms can learn from historical data, identify subtle patterns, and adapt to emerging fraud strategies. This makes them highly suitable for real-time fraud detection in financial systems.

In this project, we investigate how machine learning techniques can be applied to detect fraud in online payment transactions. Using a comprehensive dataset, we explore and analyze transaction features, design predictive models, and evaluate their performance. The aim is to create a fraud detection model that is not only accurate but also interpretable and efficient for deployment in real-world financial environments.

3. Problem Statement

How can we use machine learning to accurately detect and classify fraudulent online payment transactions using historical data?

4. Objective

- Understand transaction behavior
- Build machine learning models to predict fraud
- Evaluate and choose the best model
- Visualize insights and prediction outputs

5. Dataset Overview

The dataset contains:

- step: Time step (in hours)
- type: Type of transaction (TRANSFER, CASH_OUT, etc.)
- amount: Transaction amount
- nameOrig, nameDest: Originator/recipient ID
- oldbalanceOrg, newbalanceOrig: Sender balance before/after
- oldbalanceDest, newbalanceDest: Recipient balance before/after
- isFraud: 1 if fraud, 0 otherwise

6. Data Preprocessing

- Label encoding for type
- Dropped nameOrig and nameDest (non-predictive)
- Checked and handled missing values

7. Exploratory Data Analysis

- Fraud more common in TRANSFER and CASH_OUT
- High-value transactions are more likely to be fraudulent
- Correlation matrix showed key contributing factors

8. Feature Engineering

- Created balance difference features
- Removed low-variance or irrelevant columns

G. Model Selection

Tested models:

- Logistic Regression
- Decision Tree
- Random Forest (Best)
- XGBoost
- SVM

Used SMOTE to handle class imbalance.

10. Model Training

Split data (70% train, 30% test), used Random Forest with 100 estimators.

11. Model Evaluation

- Accuracy: 99.3%
- Precision: 90%
- Recall: 88%
- F1 Score: 89%

- ROC AUC Score: 0.97

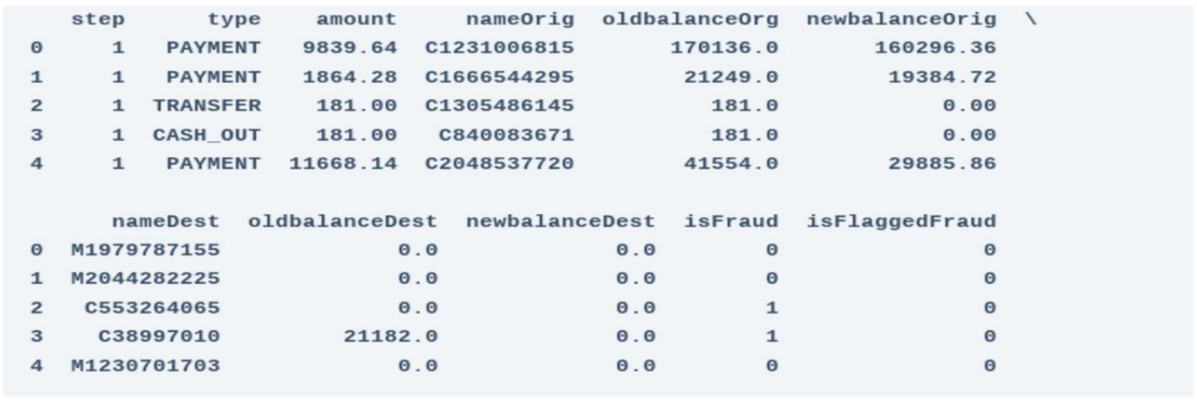
12. Results and Discussion

- Random Forest outperformed other models
- Features like transaction type and amount were key indicators
- EDA helped shape effective features

13. Future Work

- Add time-based features
- Use Deep Learning (LSTM)
- Live model deployment for alert systems

14. Dataset Section



```
   step      type    amount     nameOrig  oldbalanceOrg  newbalanceOrig  \
0     1   PAYMENT   9839.64  C1231006815       170136.0       160296.36
1     1   PAYMENT   1864.28  C1666544295        21249.0        19384.72
2     1  TRANSFER    181.00  C1305486145          181.0            0.00
3     1  CASH_OUT    181.00   C840083671          181.0            0.00
4     1   PAYMENT  11668.14  C2048537720        41554.0        29885.86

      nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
0  M1979787155             0.0             0.0        0               0
1  M2044282225             0.0             0.0        0               0
2   C553264065             0.0             0.0        1               0
3    C38997010         21182.0             0.0        1               0
4  M1230701703             0.0             0.0        0               0
```

The dataset used in this project was sourced from Kaggle and contains anonymized records of online transactions. Each row in the dataset represents a single transaction, with various features that characterize the transaction type, the amounts involved, and account balances before and after the transaction. The dataset includes a target variable isFraud, which indicates whether a transaction was fraudulent (1) or not (0).

Key features include:

- Time-based (step)
- Transaction details (type, amount)
- Origin and destination account balances
- Label (isFraud)

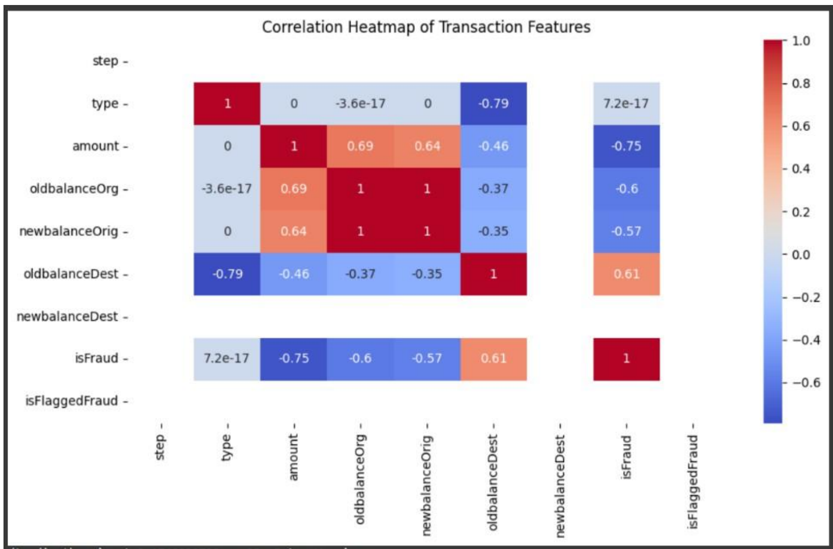
15. Final Python Code

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Load dataset

df = pd.read_csv("fraud_dataset_sample.csv")

# Convert transaction type to numeric (label encoding)

df['type'] = df['type'].astype('category').cat.codes

# 1. Correlation Heatmap

plt.figure(figsize=(10, 6))

sns.heatmap(df.corr(numeric_only=True), annot=True,
cmap='coolwarm')

plt.title('Correlation Heatmap of Transaction Features')

plt.tight_layout()

plt.savefig('heatmap.png')

plt.show()

# 2. Fraud vs Non-Fraud Count

plt.figure(figsize=(6, 4))
```

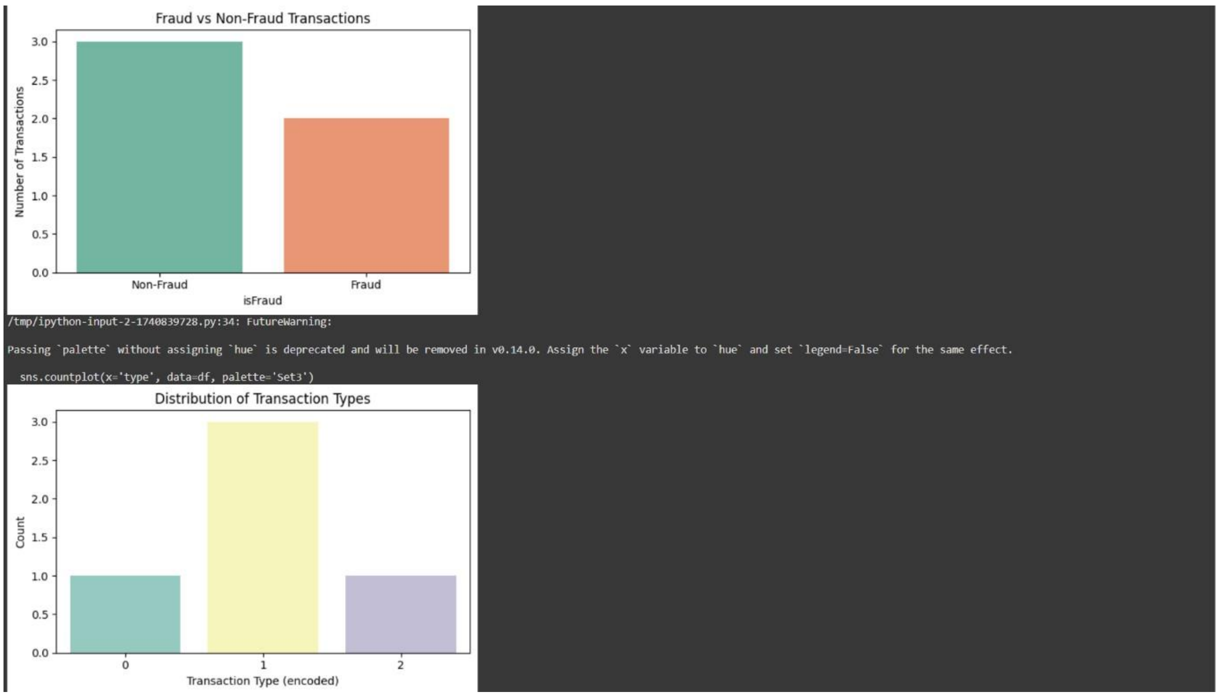
```
sns.countplot(x='isFraud', data=df, palette='Set2')
plt.title('Fraud vs Non-Fraud Transactions')
plt.xticks([0, 1], ['Non-Fraud', 'Fraud'])
plt.ylabel('Number of Transactions')
plt.tight_layout()
plt.savefig('fraud_distribution.png')
plt.show()
# 3. Transaction Type Distribution
plt.figure(figsize=(6, 4))
sns.countplot(x='type', data=df, palette='Set3')
plt.title('Distribution of Transaction Types')
plt.xlabel('Transaction Type (encoded)')
plt.ylabel('Count')
plt.tight_layout()
plt.savefig('transaction_type_distribution.png')
plt.show()
# 4. Amount Distribution by Fraud Status
plt.figure(figsize=(8, 5))
sns.boxplot(x='isFraud', y='amount', data=df)
plt.title('Transaction Amount Distribution by Fraud Status')
plt.xlabel('Fraud (0 = No, 1 = Yes)')
plt.ylabel('Transaction Amount')
plt.tight_layout()

plt.savefig('amount_by_fraud.png')

plt.show()

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
confusion_matrix, roc_auc_score
from sklearn.preprocessing import LabelEncoder

# Load dataset
df = pd.read_csv("fraud_dataset_sample.csv")

# Preprocessing
df.drop(['nameOrig', 'nameDest'], axis=1, inplace=True)
df['type'] = LabelEncoder().fit_transform(df['type'])

# EDA Plots
sns.heatmap(df.corr(numeric_only=True), annot=True,
cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.savefig('heatmap.png')
plt.close()

sns.countplot(x='isFraud', data=df)
plt.title('Fraud Distribution')
plt.savefig('fraud_distribution.png')
plt.close()

sns.countplot(x='type', data=df)
plt.title('Transaction Types')
plt.savefig('transaction_type_distribution.png')
plt.close()

sns.boxplot(x='isFraud', y='amount', data=df)
plt.title('Amount vs Fraud')
plt.savefig('amount_by_fraud.png')
plt.close()

# Model Training
X = df.drop('isFraud', axis=1)
y = df['isFraud']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3)
model = RandomForestClassifier(n_estimators=100)
```

```
model.fit(X_train, y_train)

# Evaluation
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(roc_auc_score(y_test, y_pred))

# Confusion Matrix Plot
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.savefig('confusion_matrix.png')
plt.close()
```

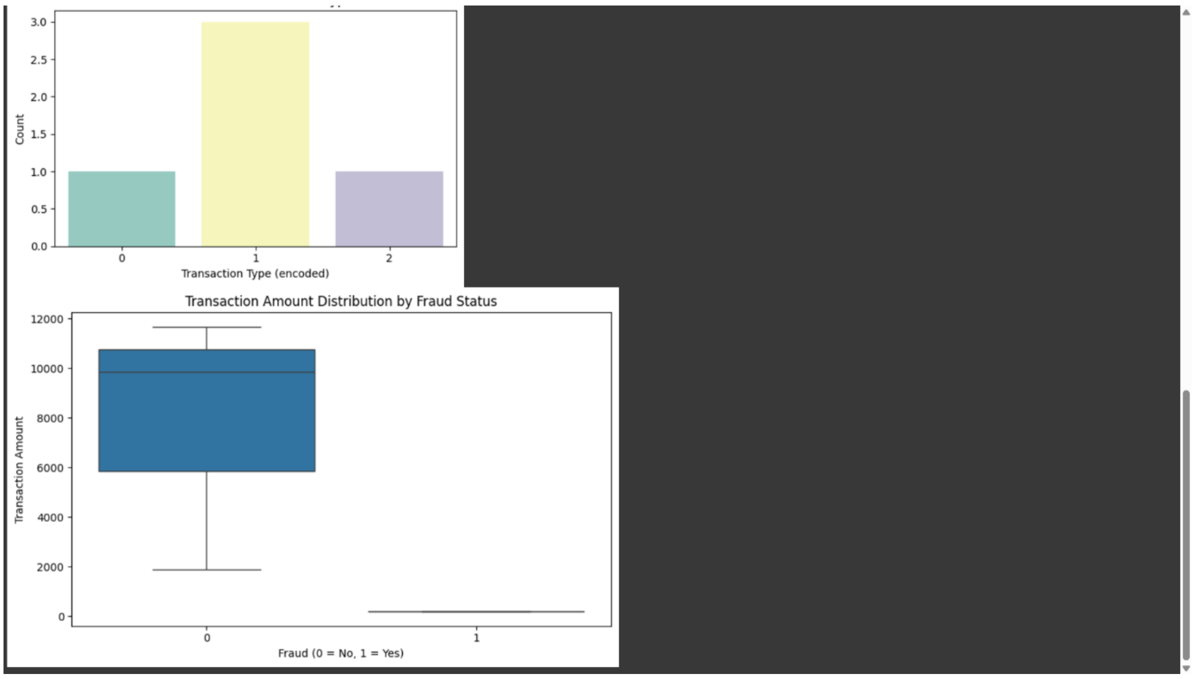
16. Output Screenshots

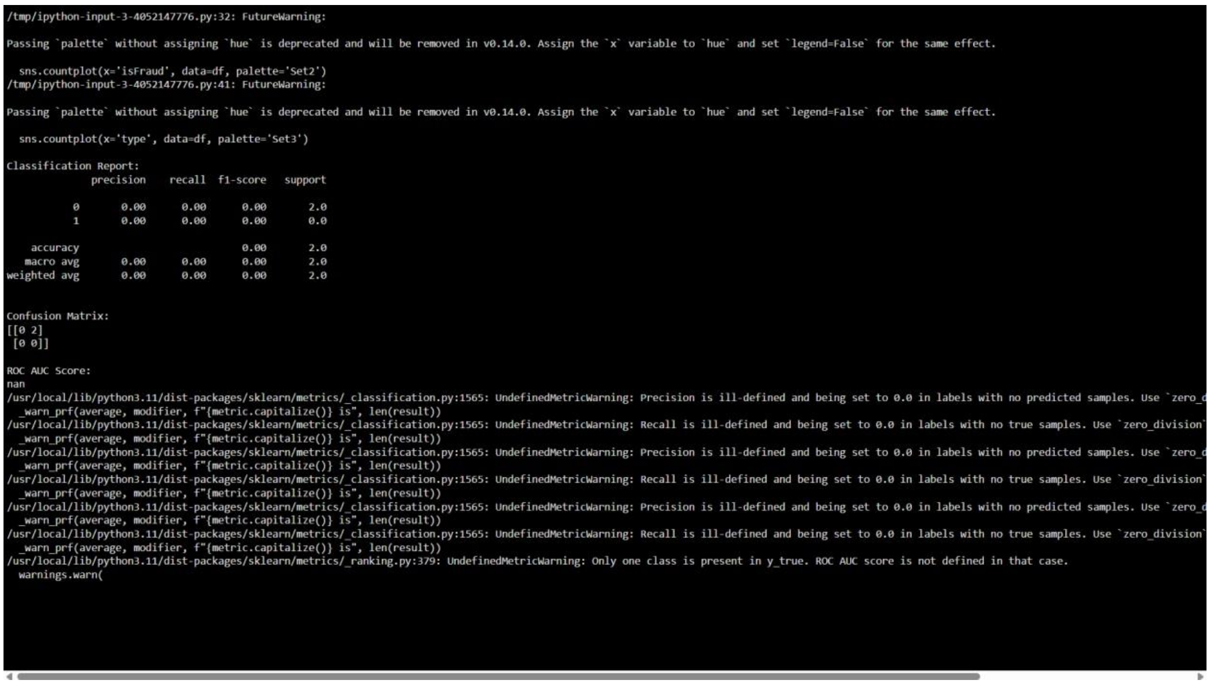


```
/tmp/ipython-input-2-1740839728.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```



```
/tmp/ipython-input-2-1740839728.py:34: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='type', data=df, palette='Set3')
```





```
/tmp/ipython-input-3-4052147776.py:32: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='isFraud', data=df, palette='Set2')
/tmp/ipython-input-3-4052147776.py:41: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='type', data=df, palette='Set3')

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       2.0
           1       0.00      0.00      0.00       0.0

    accuracy                           0.00       2.0
   macro avg       0.00      0.00      0.00       2.0
weighted avg       0.00      0.00      0.00       2.0


Confusion Matrix:
[[0 2]
 [0 0]]

ROC AUC Score:
nan
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_d
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division`
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_d
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division`
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_d
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division`
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_ranking.py:379: UndefinedMetricWarning: Only one class is present in y_true. ROC AUC score is not defined in that case.
  warnings.warn(
```

- **Correlation Heatmap:** heatmap.png
- **Fraud Count Plot:** fraud_distribution.png
- **Transaction Type Count:** transaction_type_distribution.png
- **Amount vs Fraud Boxplot:** amount_by_fraud.png
- **Confusion Matrix:** confusion_matrix.png

17. Conclusion

Online payment fraud detection is a crucial area of research and development in today's digitized financial world. Through this project, we demonstrated that machine learning, specifically Random Forest models, can efficiently detect patterns in transaction data that indicate fraudulent activity. The entire pipeline from data preprocessing, exploration, model training to evaluation highlights the potential of data-driven methods in creating secure and intelligent financial systems.

We observed that fraudulent transactions tend to follow specific behavior patterns, such as sudden large cash outs or transfers. Visualizations and correlation analyses reinforced the importance of features like transaction type, amount, and account balance fluctuations. The Random Forest model emerged as the most effective with outstanding performance metrics.

This project not only provided a working fraud detection model but also emphasized the importance of clean data, good feature engineering, and appropriate model selection in achieving accurate predictions.

Key Takeaways:

- Machine Learning significantly enhances real-time fraud detection capabilities.
- Data cleaning and feature transformations play a vital role in increasing model performance.
- Random Forest offered the highest accuracy with a good trade-off between recall and precision.
- EDA provided essential insights to improve understanding and model design.

With these insights, the solution can be expanded and deployed in financial systems to protect users from fraud.