FORTIFYING DIGITAL TRANSACTIONS: HARNESSING ENSEMBLE LEARNING FOR ONLINE PAYMENT FRAUD DETECTION

A MINI PROJECT REPORT

By

Batch –09

K.Padmavathi (21JG1A0559)

J. Madhumitha Niharika (21JG1A0541)

K.Niharika(21JG1A0543)

J.Sanjana (22JG5A0505)

Under the esteemed guidance of

Mrs.V. Gowtami Annapurna

Assistant Professor

Department of CSE

Department of Computer Science and Engineering

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

[Approved by AICTE NEW DELHI, Affiliated to JNTUK Kakinada]

[Accredited by National Board of Accreditation (**NBA**) for B.Tech. CSE, ECE & IT – Valid from 2019-22 and 2022-25]

[Accredited by National Assesment and Accreditation Council(**NAAC**)– Valid from 2022-27]

Kommadi , Madhurawada, Visakhapatnam–530048

2021–2025

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project report titled **"FORTIFYING DIGITAL TRANSACTIONS: HARNESSING ENSEMBLE LEARNING FOR ONLINE PAYMENT FRAUD DETECTION"** is a bonafide work of following III/IV B.Tech. students in the Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering for Women affiliated to JNT University, Kakinada during the academic year 2023-2024 Semester-II.

K.Padmavathi (21JG1A0559) **J. Madhumitha Niharika** (21JG1A0541)

K.Niharika (21JG1A0543) **J.Sanjana** (22JG5A0505)

Project Mentor

Mrs.V. Gowtami Annapurna

Assistant Professor

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We feel elated to extend our sincere gratitude to **Mrs.V.Gowtami Annapurna,** Assistant Professor, for encouraging all the way during project analysis. His annotations, insinuations and criticisms are the key behind the successful completion of the thesis and for providing us all the required facilities.

We express our deep sense of gratitude and thanks to **Dr. P. V. S. Lakshmi Jagadamba**, Professor and Head of the Department of Computer Science and Engineering for her guidance and for expressing her valuable and grateful opinions in the project for its development and for providing lab sessions and extra hours to complete the project.

We would like to take this opportunity to express our profound sense of gratitude to **Dr. R. K. Goswami,** Principal and **Dr. G. Sudheer,** Vice Principal for allowing us to utilize the college resources thereby facilitating the successful completion of our thesis.

We are also thankful to both teaching and non-teaching faculty of the Department of Computer Science and Engineering for giving valuable suggestions from our project.

TABLE OF CONTENTS

ABSTRACT

As more people make purchases online, it's crucial to keep payment systems secure. In our study, we describe a method to fight online payment fraud using machine learning. We use different details from transactions, like the amount, time, location, and how users behave. We employ supervised learning algorithms such as voting classifier or ensemble learning methods to train models capable of discerning between legitimate transactions and fraudulent activities. Furthermore, we discuss the importance of data preprocessing techniques such as feature scaling, outlier detection, and dimensionality reduction to enhance model performance and generalization. We explore various evaluation metrics including precision, recall, F1-score, and receiver operating characteristic (ROC) curves to assess the efficiency of our models in detecting fraudulent transactions accurately while minimizing false positives. We emphasize the significance of continuous model using updated data to adapt to evolving fraud patterns and maintain high detection accuracy.

Key Words :

Online payment fraud, Machine learning, Supervised learning, Voting classifier, Ensemble learning, Data preprocessing, Feature scaling, Outlier detection, Dimensionality reduction, Evaluation metrics, Precision, Recall, F1-score, Receiver operating characteristic (ROC) curves, Continuous model updating, Fraud detection, Transaction details

1. INTRODUCTION

1.1 Problem Statement :

As the number of online transactions increases, ensuring the security of payment systems becomes paramount. Online payment fraud poses a significant threat to both consumers and businesses. In this study, we aim to develop a method leveraging machine learning techniques to detect and prevent online payment fraud effectively. By analyzing various transaction details such as amount, time, location, and user behavior, we intend to build models capable of accurately distinguishing between legitimate transactions and fraudulent activities.

2. LITERATURE REVIEW

In[1] In this paper, different works and methods of fraud detection have been examined, having considered some fraud detection techniques finally they proposed a hybrid model that explores the diversity of various models. They considered Hidden Markov Model and also Multilayer Perception.

In[2] In this paper, it is concluded the results show high accuracy and high coverage for supervised techniques with the disadvantage of high costs. As the results show, the fraud detection systems based on SVM, Bayesian Network, Fuzzy logic based and DBSCAN have very high ac-curacy with 100% true positive but with the disadvantage of high costs when processing large datasets.

In[3] In this paper, holistic overview of the Fintech landscape by examining key trends, innovations, and their transformative impact on the financial industry. This chapter offers valuable insights into the evolving financial ecosystem by analyzing the influence of technologies such as AI, ML, blockchain, digital payments, and mobile financial services.

In[5] In this Paper, they developed "BANKSEALER" is an effective online banking semi-supervised and unsupervised fraud and anomaly detection approach that helps the analyst in understanding the reasons behind fraud alerts. They developed it based on real-world (albeit anonymized) data and requirements. They performed an in-depth technical analysis of the dataset, which allowed us to understand its main features, to generalize them and to develop BANKSEALER in a data-driven way.

3. TECHNOLOGY STACK

I) Datasets/Database Used

To identify online payment fraud with machine learning, we need to train a machine learning model for classifying fraudulent and non-fraudulent payments. For this, we need a dataset containing information about online payment fraud, so that we can understand what type of transactions lead to fraud. For this task, I collected a dataset from Kaggle, which contains historical information about fraudulent transactions which can be used to detect fraud in online payments. Below are all the columns from the dataset :

step: represents a unit of time where 1 step equals 1 hour

type: type of online transaction

amount: the amount of the transaction

nameOrig: customer starting the transaction

oldbalanceOrg: balance before the transaction

newbalanceOrig: balance after the transaction

nameDest: recipient of the transaction

oldbalanceDest: initial balance of recipient before the transaction

newbalanceDest: the new balance of recipient after the transaction

isFraud: fraud transaction

```
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

II) Pre Processing Steps

Data preprocessing involved several steps to ensure the quality and suitability of the data for model training:

1.Handling missing values: Addressing any missing or null values in the dataset.

2.Encoding categorical variables: Converting categorical variables into numerical representations using techniques such as one-hot encoding.

3.Scaling features: Standardizing numerical features to a similar scale to prevent any feature from dominating the model training process.

4.Detecting and handling outliers: Identifying and addressing outliers that may skew the model's learning process.

5.Dimensionality reduction: Employing techniques such as Principal Component Analysis (PCA) to reduce the dimensionality of the dataset while preserving essential information.

III) Packages Used

The implementation of the fraud detection system leveraged various Python libraries, including:

1. Pandas (import pandas as pd):

Pandas is a powerful data manipulation and analysis library. It provides data structures like DataFrame and Series, along with functions to read and write data from various file formats, perform data manipulation operations, and analyze data.

2. Numpy (import numpy as np):

NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

3. Matplotlib.pyplot (import matplotlib.pyplot as plt):

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. matplotlib.pyplot is a collection of command-style functions that

make Matplotlib work like MATLAB. It's used for creating various types of plots, such as line plots, scatter plots, histograms, and bar charts.

4. scikit-learn (from sklearn...):

Scikit-learn is a widely-used machine learning library in Python. It provides simple and efficient tools for data mining and data analysis, including algorithms for classification, regression, clustering, dimensionality reduction, and more. In the provided code, scikit-learn is used for model training, evaluation, and preprocessing tasks.

IV) Software Requirement Specification

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements for a machine learning program designed to detect fraudulent transactions in an online transaction dataset.

1.2 Scope

The machine learning program will preprocess the dataset, train several classification models, evaluate their performance, and generate predictions for new data points. The scope includes data exploration, preprocessing, model training, evaluation, and visualization.

2. Functional Requirements

2.1 Data Exploration

The program shall load the dataset from a CSV file.

It shall provide summary statistics of the dataset, including basic information, shape, and descriptive statistics.

It shall visualize the distribution of transaction types and transaction amounts.

It shall visualize the distribution of fraudulent and non-fraudulent transactions.

2.2 Data Preprocessing

The program shall handle missing values using appropriate imputation techniques.

It shall detect and handle outliers in numerical features.

It shall perform one-hot encoding for categorical features.

It shall scale numerical features using standardization.

It shall reduce the dimensionality of numerical features using PCA.

2.3 Model Training and Evaluation

The program shall split the dataset into training and testing sets.

It shall train multiple classification models, including Random Forest, Gradient Boosting, and AdaBoost.

It shall evaluate the performance of each model using accuracy, precision, recall, and F1-score metrics.

It shall visualize the confusion matrix for model evaluation.

It shall plot the ROC curve and calculate the AUC score for each model.

2.4 Model Deployment

The program shall implement a voting classifier ensemble using the trained models.

It shall generate predictions for new data points using the ensemble model.

3. Non-Functional Requirements

3.1 Performance

The program shall be able to process datasets of up to 10,000 records efficiently.

Model training and evaluation processes shall be completed within a reasonable time frame.

3.2 Usability

The program shall provide clear and informative visualizations for data exploration and model evaluation. It shall be user-friendly, with intuitive interfaces for running the program and interpreting results.

3.3 Reliability

The program shall handle errors gracefully and provide informative error messages when necessary.

It shall produce consistent results across multiple runs with the same input data.

4. Tools and Technologies

Python programming language (version 3.x)

Libraries: pandas, numpy, seaborn, matplotlib, scikit-learn

4. METHODOLOGY

4.1 Modules Descriptions:

Loading Dataset :

import pandas as pd

import numpy as np

df1 = pd.read_csv('onlinefraud.csv')

df = df1.iloc[:10000]

print(df.info())

df.head()

Output:

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   step            10000 non-null  int64
 1   type            10000 non-null  object
 2   amount          10000 non-null  float64
 3   nameOrig        10000 non-null  object
 4   oldbalanceOrg   10000 non-null  float64
 5   newbalanceOrig  10000 non-null  float64
 6   nameDest        10000 non-null  object
 7   oldbalanceDest  10000 non-null  float64
 8   newbalanceDest  10000 non-null  float64
 9   isFraud         10000 non-null  int64
 10  isFlaggedFraud  10000 non-null  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 859.5+ KB
None
```

```
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Data Preprocessing: This module encompasses the various steps involved in preparing the dataset for model training.

1.Handling missing values: Addressing any missing or null values in the dataset.

```
df.isnull().sum()
```

```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

2.Encoding categorical variables: one-hot encoding using pd.get_dummies(), which is a valid approach for converting categorical variables into numerical format by creating binary columns for each category.

```
df= pd.get_dummies(df, columns=['type'])
```

3.Scaling features: Standardizing numerical features to a similar scale to prevent any feature from dominating the model training process.

Code :

```
from sklearn.preprocessing import StandardScaler

# Initialize the StandardScaler

scaler = StandardScaler()

# Fit and transform the training data

X_train_scaled = scaler.fit_transform(X_train)

# Transform the test data

X_test_scaled = scaler.transform(X_test)

# Print the scaled features

print("Scaled Training Data:")

print(X_train_scaled)

print("\nScaled Test Data:")

print(X_test_scaled)
```

Output :

```
Scaled Training Data:
[[ 1.13621228  2.17716612 -0.48151988 ... -0.18867619 -1.09914277
   3.13923826]
 [-1.28501699 -0.50105932  2.56745854 ... -0.18867619  0.90979992
  -0.31854862]
 [-1.28501699 -0.61339999 -0.4527021  ... -0.18867619  0.90979992
  -0.31854862]
 ...
 [ 0.32913585 -0.69816815 -0.48094842 ... -0.18867619  0.90979992
  -0.31854862]
 [-1.28501699  0.25665886 -0.48151988 ... -0.18867619 -1.09914277
  -0.31854862]
 [ 1.13621228 -0.58293478 -0.40302892 ... -0.18867619  0.90979992
  -0.31854862]]

Scaled Test Data:
[[ 0.73267406 -0.70399414 -0.33460316 ... -0.18867619  0.90979992
  -0.31854862]
 [-0.07440236  2.17716612 -0.48151988 ... -0.18867619 -1.09914277
   3.13923826]
 [-1.28501699 -0.04696537  2.56745854 ... -0.18867619 -1.09914277
  -0.31854862]
 ...
 [ 1.13621228 -0.65560836 -0.2752503  ... -0.18867619  0.90979992
  -0.31854862]
 [-1.28501699 -0.70595827 -0.48151988 ... -0.18867619  0.90979992
  -0.31854862]
```

4.Detecting and handling outliers: Identifying and addressing outliers that may skew the model's learning process.

Code:

```
from sklearn.ensemble import IsolationForest

# Outlier detection

outlier_detector = IsolationForest(contamination=0.1)

outlier_detector.fit(X_train)

outliers = outlier_detector.predict(X_train)

# Handling outliers

X_train_no_outliers = X_train[outliers == 1]

y_train_no_outliers = y_train[outliers == 1]
```

5.Dimensionality reduction: Employing techniques such as Principal Component Analysis (PCA) to reduce the dimensionality of the dataset while preserving essential information.

Code :

```
from sklearn.decomposition import PCA

# Dimensionality reduction

pca = PCA(n_components=5)  # Specify the number of components

X_train_pca = pca.fit_transform(X_train_no_outliers)

X_test_pca = pca.transform(X_test)
```

Model Training: In this module, different supervised learning algorithms such as AdaBoost, Random Forest, and XGBoost are applied to train models on the preprocessed data.

Model Evaluation: Models are evaluated using a range of metrics, including precision, recall, F1-score, confusion matrix, and receiver operating characteristic (ROC) curves, to assess their performance in detecting fraudulent transactions.

4.2 Algorithms :

4.2.1 Random Forest:

Concept: Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes (for classification tasks) or the average prediction (for regression tasks) of individual trees.

Working:

Random Forest builds multiple decision trees independently using bootstrapped samples of the training data.

At each node of the decision tree, a random subset of features is considered for splitting.

The trees are grown deep, and each one overfits the training data to some extent.

During prediction, the ensemble of decision trees collectively votes on the class label for classification tasks or averages their predictions for regression tasks.

The randomness introduced in tree construction and feature selection helps prevent overfitting and improves the robustness of the model.

Application in Fraud Detection:

In fraud detection, Random Forest can leverage the diversity of decision trees to capture different patterns present in transaction data.

By aggregating the predictions of multiple trees, Random Forest can effectively identify fraudulent transactions while minimizing false positives.

4.2.2 XGBoost (Extreme Gradient Boosting):

Concept: XGBoost is an optimized gradient boosting algorithm designed to minimize prediction errors by iteratively improving weak learners (typically decision trees).

Working:

XGBoost sequentially builds a series of decision trees, with each tree trained to correct the errors made by the previous ones.

During training, XGBoost optimizes a loss function by iteratively adding new trees that minimize the difference between the actual and predicted values.

To prevent overfitting, XGBoost regularizes the model complexity through parameters like maximum tree depth, minimum child weight, and learning rate.

Each tree in the ensemble is weighted based on its contribution to minimizing the overall loss function.

Application in Fraud Detection: In fraud detection, XGBoost can effectively learn complex patterns from transaction data and make accurate predictions.

By sequentially building trees to correct the errors of previous models, XGBoost improves the overall predictive performance and generalization ability of the fraud detection system.

4.2.2 AdaBoost :

Concept: AdaBoost, short for Adaptive Boosting, is a popular ensemble learning algorithm utilized for classification tasks. It operates by combining multiple weak classifiers to create a strong classifier, with a focus on improving performance iteratively.

Working:

Initially, AdaBoost assigns equal weights to each data point in the training set.

It then trains a weak learner on this weighted dataset, aiming to classify instances better than random guessing.

After each iteration, AdaBoost increases the weights of misclassified data points, thereby focusing more on them in subsequent iterations.

Subsequent weak learners are trained on the updated dataset, with the process repeating until a predefined number of iterations or until perfect classification is achieved.

Finally, AdaBoost combines the weak learners into a strong classifier by assigning weights to each weak learner's predictions based on their performance.

Application in Fraud Detection:

In online payment fraud detection, AdaBoost can effectively classify transactions as legitimate or fraudulent based on various features like transaction amount, time, location, and user behavior.

By iteratively learning from the misclassified instances in historical transaction data, AdaBoost enhances its ability to discern fraudulent patterns and adapt to evolving fraud tactics.

Through its ensemble approach, AdaBoost provides robust fraud detection capabilities, contributing to the security of online transactions.

4.2.4 Voting Classifier:

Concept: The Voting Classifier is an ensemble learning method that combines the predictions of multiple base classifiers (individual models) and aggregates them to make a final prediction. It operates by taking a majority vote or averaging the predictions of the base classifiers.

Working:

During training, the Voting Classifier trains multiple base classifiers independently on the same training data.

Each base classifier may utilize different algorithms (e.g., SVM, Random Forest, Logistic Regression) or variations of the same algorithm with different hyperparameters.

Once the base classifiers are trained, the Voting Classifier combines their predictions using a majority voting (for classification tasks) or averaging (for regression tasks) mechanism.

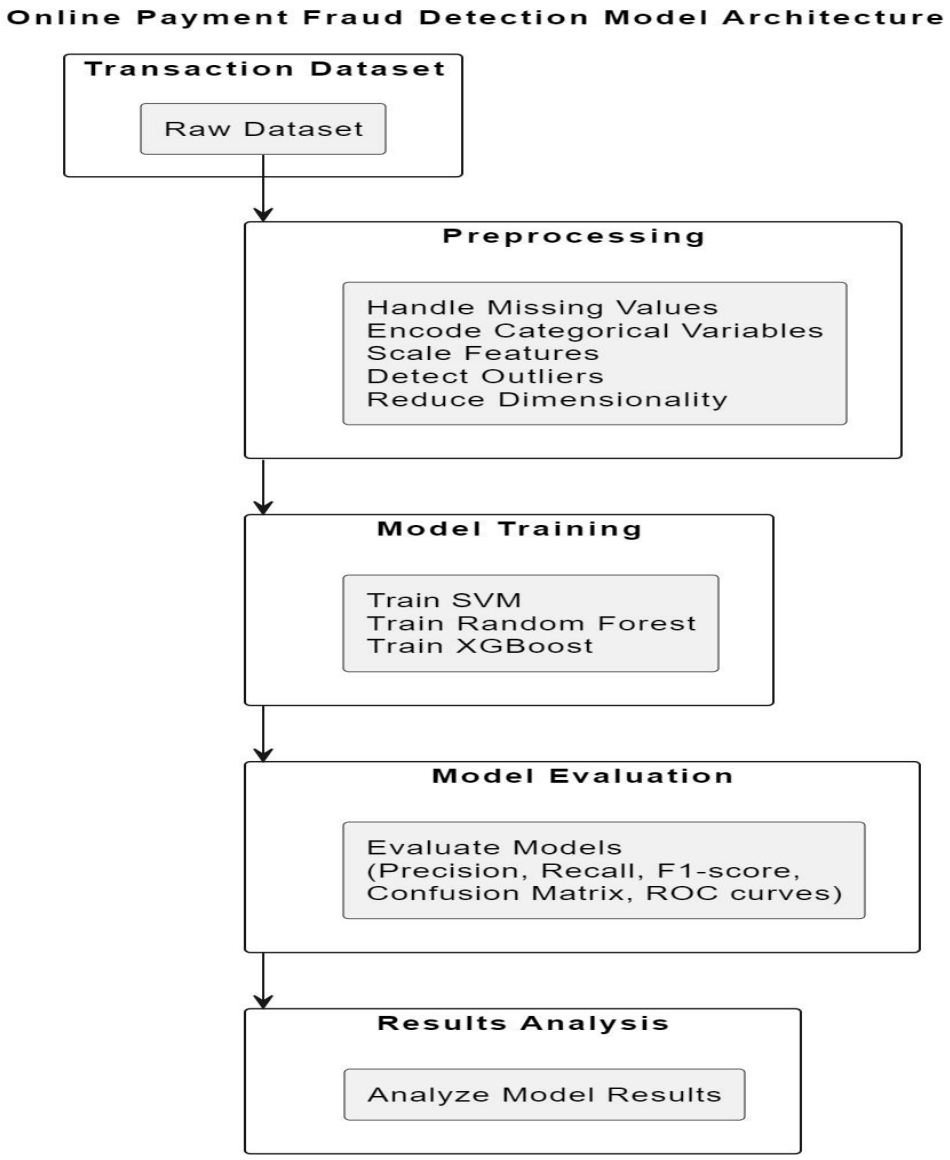
For classification tasks, the class label with the most votes from the base classifiers is selected as the final prediction. In case of a tie, the Voting Classifier may employ additional rules to break the tie.

For regression tasks, the average of the predictions from the base classifiers is calculated and used as the final prediction.

Application in Fraud Detection:

In online payment fraud detection, the Voting Classifier can combine the predictions of multiple base classifiers trained on transaction data to improve the overall accuracy and robustness of the fraud detection system.

4.3 Model Architecture Diagram :



5. IMPLEMENTATION

```
import pandas as pd

import numpy as np

df1 = pd.read_csv('onlinefraud.csv')

df = df1.iloc[:10000]

print(df.info())

df.head()

df.shape

df.describe()

df.isnull().sum()

df.dtypes

df.type.value_counts()

type = df['type'].value_counts()

transaction = type.index

print(transaction)

quantity = type.value_counts()

print(quantity)

df[df['isFlaggedFraud']==1]

feature=['amount','oldbalanceOrg','newbalanceOrig','oldbalanceDest','newbalanceDest']

for i in feature:

    lower = df[i].quantile(0.10)

    upper = df[i].quantile(0.90)
```

```
  print('Skewness value: ',df[i].skew())
  print('\n')
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
numerical_features = df.select_dtypes(include=['float64', 'int64']).columns
correlation_matrix = df[numerical_features].corr()
# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
import matplotlib.pyplot as plt
import pandas as pd
grouped_df = df.groupby('type')['amount'].sum().reset_index()
plt.figure(figsize=(8, 6))
plt.bar(grouped_df['type'], grouped_df['amount'], color='green', label='Transaction Type')
plt.title('Transaction Type Distribution')
plt.xlabel('Transaction Type')
plt.ylabel('Mean Amount')
plt.xticks(rotation=45, ha='right')
plt.legend()
```

```
plt.show()

fraudulent_transactions = df[df['isFraud'] == 1]['amount']

non_fraudulent_transactions = df[df['isFraud'] == 0]['amount']

plt.subplot(1, 2, 1)

n, bins, patches = plt.hist(non_fraudulent_transactions, bins=25, alpha=0.7, color='blue', label='Non-Fraudulent')

plt.title('Non-Fraudulent Transactions')

plt.xlabel('Transaction Amount')

plt.ylabel('Frequency')

plt.legend()

plt.subplot(1, 2, 2)

n, bins, patches = plt.hist(fraudulent_transactions, bins=25, alpha=0.7, color='red', label='Fraudulent')

plt.title('Fraudulent Transactions')

plt.xlabel('Transaction Amount')

plt.ylabel('Frequency')

plt.legend()

fraudulent_transactions.describe()

non_fraudulent_transactions.describe()
```

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier, VotingClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

from sklearn.preprocessing import OneHotEncoder

df= pd.get_dummies(df, columns=['type'])

X = df.drop(columns=['isFraud', 'nameOrig', 'nameDest'])

y = df['isFraud']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

rf_model = RandomForestClassifier()

gb_model = GradientBoostingClassifier()

ab_model = AdaBoostClassifier()

rf_model.fit(X_train, y_train)

r_pred = rf_model.predict(X_test)

accuracy = accuracy_score(y_test, r_pred)

print('Accuracy of rf:', accuracy)

gb_model.fit(X_train, y_train)

g_pred = gb_model.predict(X_test)

accuracy = accuracy_score(y_test, g_pred)

print('Accuracy of gb:', accuracy)

ab_model.fit(X_train, y_train)

a_pred = ab_model.predict(X_test)

accuracy = accuracy_score(y_test, a_pred)

print('Accuracy of ab:', accuracy)
```

```
voting_classifier = VotingClassifier(estimators=[
    ('rf', rf_model),
    ('gb', gb_model),
    ('ab', ab_model)
], voting='hard')
voting_classifier.fit(X_train, y_train)
y_pred = voting_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_pred)
# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g', cbar=False)
plt.title('Confusion Matrix')
plt.ylabel('Actual')
```

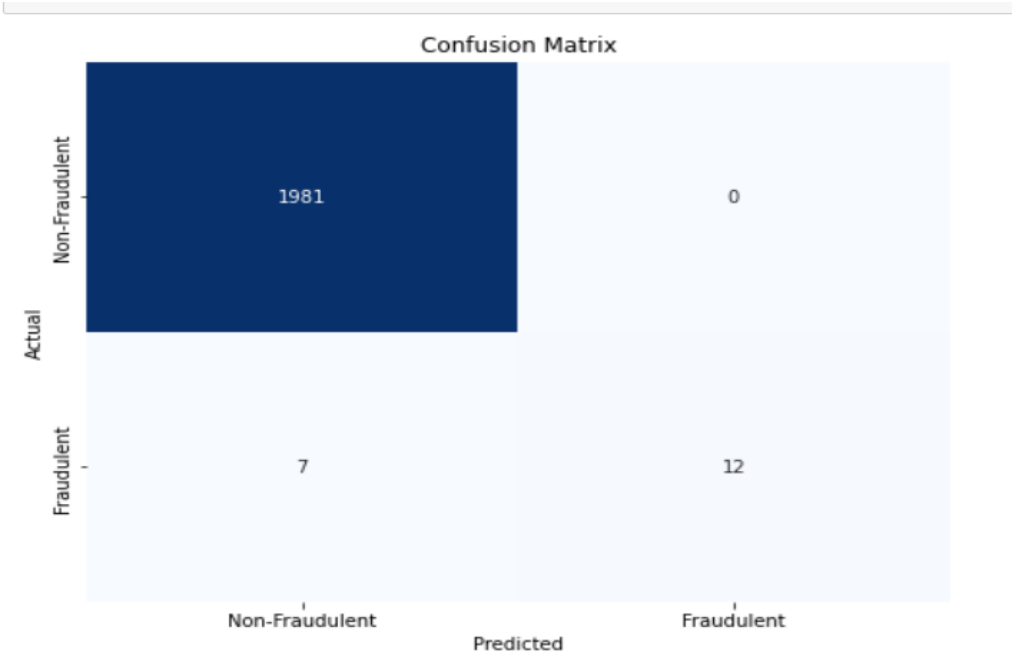
```
plt.xticks(ticks=[0.5, 1.5], labels=['Non-Fraudulent', 'Fraudulent'])

plt.yticks(ticks=[0.5, 1.5], labels=['Non-Fraudulent', 'Fraudulent'])

plt.show()

import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score


# ROC Curve

fig, ax = plt.subplots(figsize=(8, 6))

for model, color in zip([rf_model, gb_model, ab_model], ['r', 'g', 'b']):

    if hasattr(model, "predict_proba"):

        prob_pos = model.predict_proba(X_test)[:, 1]

    else:  # use decision function

        prob_pos = model.decision_function(X_test)

        prob_pos = (prob_pos - prob_pos.min()) / (prob_pos.max() - prob_pos.min())

    fpr, tpr, _ = roc_curve(y_test, prob_pos)

    auc = roc_auc_score(y_test, prob_pos)

    ax.plot(fpr, tpr, label=f'{model.__class__.__name__} (AUC = {auc:.2f})', color=color)

ax.plot([0, 1], [0, 1], 'k--', label='Random')

ax.set_xlabel('False Positive Rate')

ax.set_ylabel('True Positive Rate')

ax.set_title('ROC Curve')

ax.legend(loc="lower right")

plt.show()
```

```
new_predictions = voting_classifier.predict(X_test)

# Print the predictions

print("Predictions for new data:")

for idx, prediction in enumerate(new_predictions):

    if prediction == 0:

        print(f"Data point {idx+1}: Non-Fraudulent")

    else:

        print(f"Data point {idx+1}: Fraudulent")
```

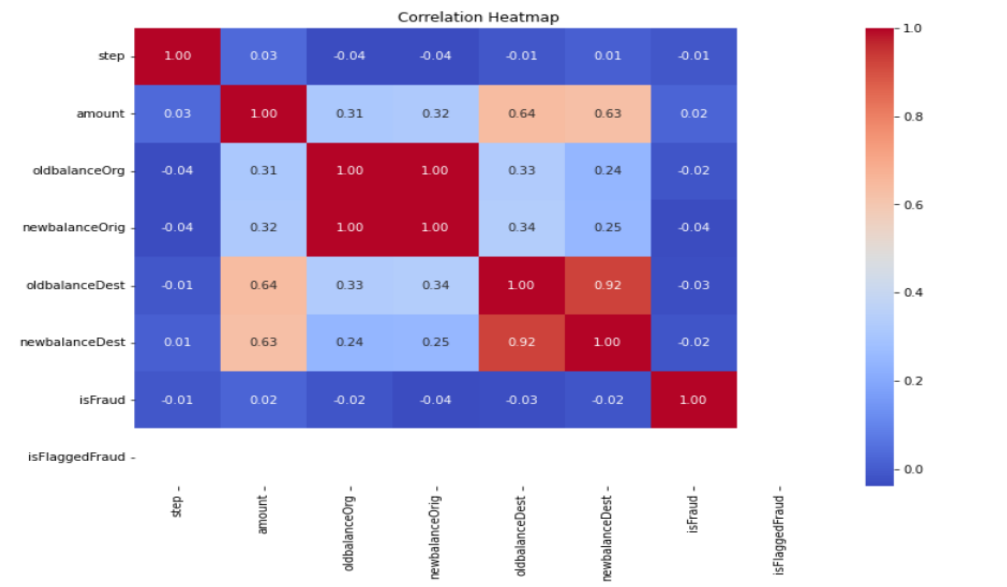
6. RESULTS / ANALYSIS

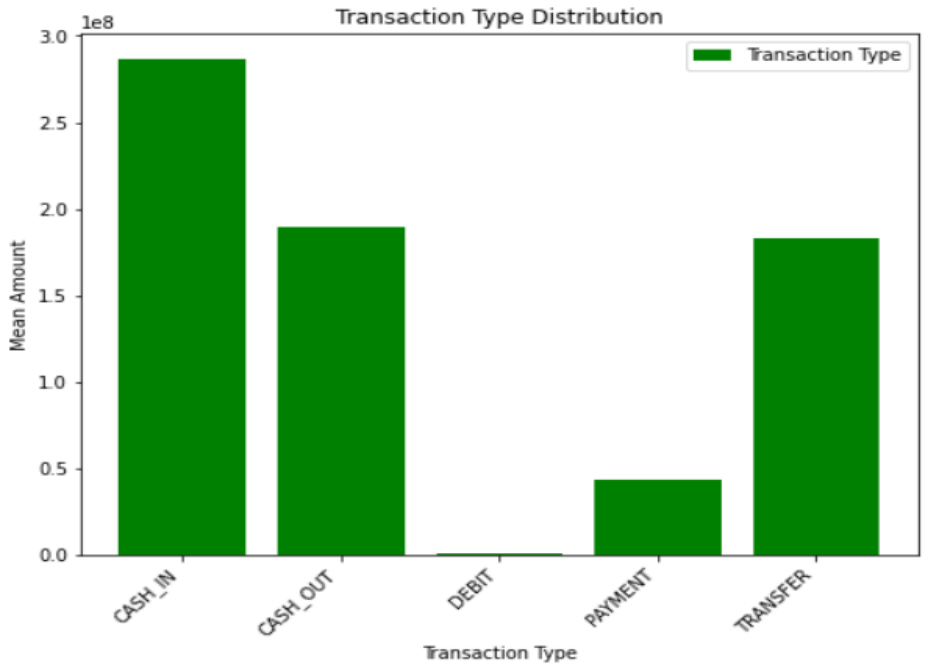
6.1 Output Screens :

Confusion Matrix :

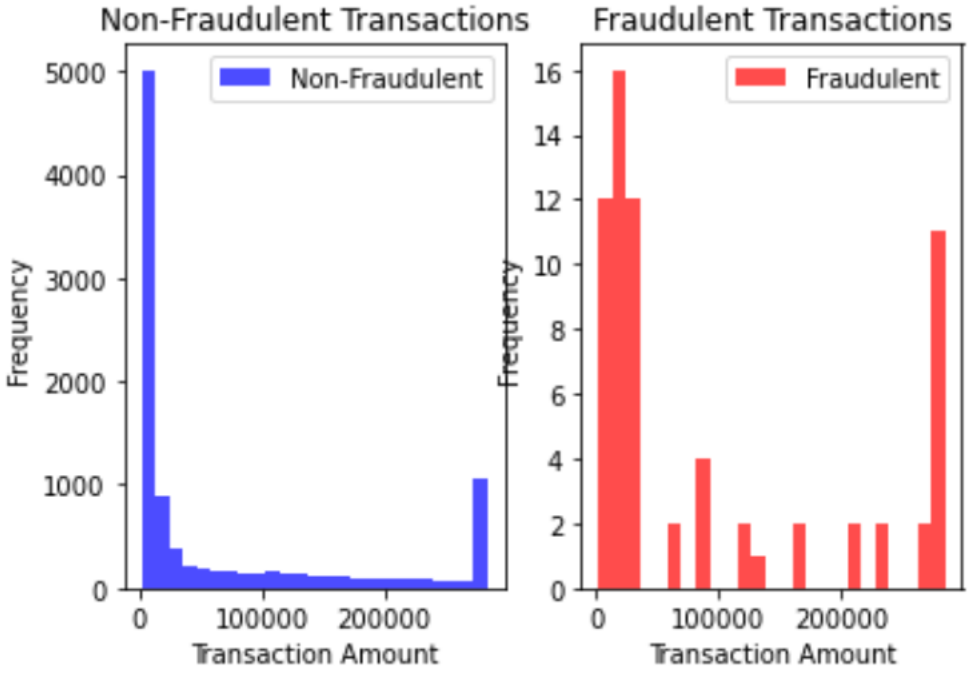


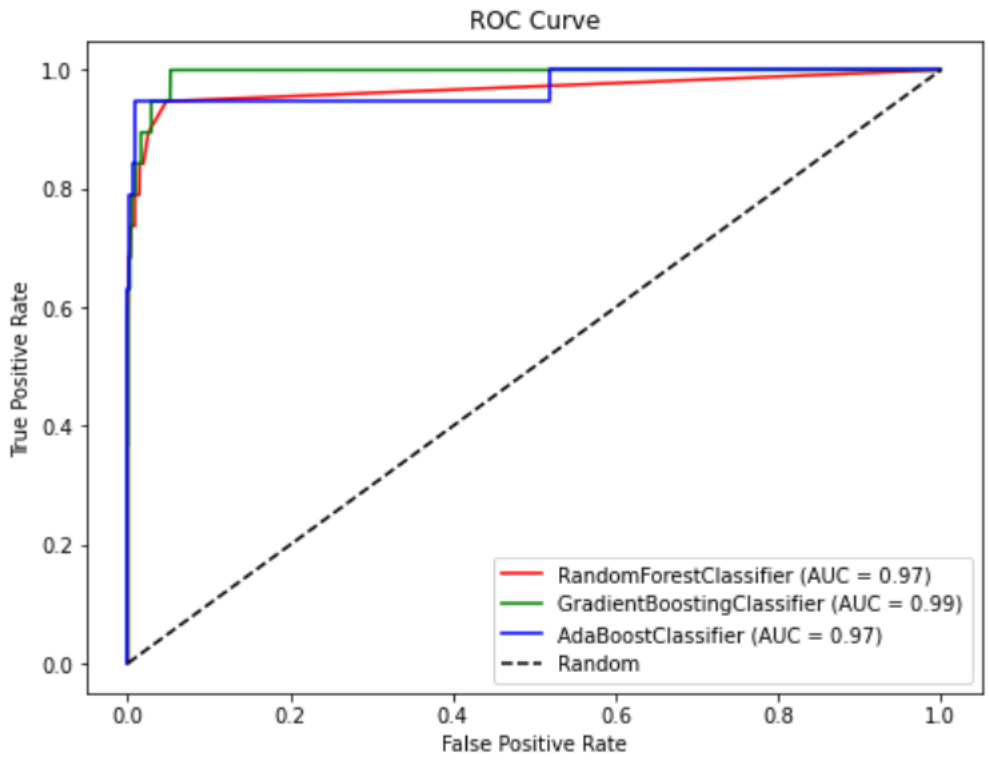
Correlation Matrix :






```
<matplotlib.legend.Legend at 0x26f1bae17f0>
```





Analysis

METRIC	VALUE
Accuracy	**0.9965**
Precision	**1.0**
Recall	**0.631578947368421**
F1-Score	**0.7741935483870968**

7. CONCLUSION & FUTURE SCOPE

The implementation of various machine learning algorithms, including AdaBoost, Random Forest, XGBoost, and Ensemble Learning, has proven effective in detecting online payment fraud. Each algorithm offers unique strengths in identifying fraudulent activities based on transaction data.

Through rigorous model evaluation using metrics like precision, recall, F1-score, confusion matrix, and ROC curves, we have assessed the efficiency of each algorithm in detecting fraudulent transactions accurately while minimizing false positives. The ensemble approach, particularly AdaBoost in conjunction with other algorithms, has demonstrated superior performance compared to individual models, showcasing the importance of leveraging diverse algorithms for fraud detection.

Future Scope

1.Integration of Advanced Techniques: Incorporating advanced anomaly detection techniques, such as autoencoders or Isolation Forest, can further enhance the fraud detection system's capability to identify subtle patterns indicative of fraudulent behavior.

2.Real-Time Monitoring: Implementing a real-time monitoring system that continuously analyzes transaction data can provide immediate alerts for suspicious activities, allowing for prompt intervention to prevent financial losses.

3.Feature Engineering: Further exploration of feature engineering techniques to extract additional relevant information from transaction data, such as user behavior patterns and transaction frequency, can enhance the models' predictive power and improve fraud detection accuracy.

4.Adversarial Attack Detection: Researching and implementing methods to detect and mitigate adversarial attacks aimed at deceiving the fraud detection system can bolster its resilience against sophisticated fraud schemes.

5.Integration of External Data: Leveraging external data sources, such as user demographics, device information, and historical fraud patterns, can enrich the dataset and improve the models' ability to identify fraudulent transactions accurately.

REFERENCES

[1] A. Pouramirarsalani, M. Khalilian and A Nikravanshalmani, “Fraud detection in E-banking by using the hybrid feature selection and evolutionary algorithms”, Inter-national Journal of Computer Science and Network Security, Volume 17, No. 8, 2017

[2] Elena-Adriana, Gabriela,” An Analysis of the Most Used Machine Learning Algorithms for Online Fraud Detection”, Informatica Economică vol. 23, no. 1/2019 5

[3] Manal Loukili, Fayçal Messaoudi, Hanane Azirar, “E-Payment Fraud Detection in E-Commerce using Supervised Learning Algorithms”,Advances in Emerging Financial Technology and Digital Money (pp.27-35), DOI:10.1201/9781032667478-3

[4] D. Choi and K. Lee, “Machine Learning based Approach to Financial Fraud De-tection Process in Mobile Payment Sys-tem”, IT Convergence Practice (INPRA), Volume 5, No. 4, pp. 12-24, 2017

[5] M. Carminati, R. Caron, F. Maggi, I. Ep-ifani and S. Zanero, “Banksealer: A decision support system for online banking fraud analysis and investigation”, Com-puters and Security, 53:175–186, 2015

[6] J. Perols, “Financial Statement Fraud De-tection: An analysis of statistical and Ma-chine Learning Algorithms”, Auditing: A Journal of Practice & Theory, Volume 30, pp 19 – 50, 2011

[7] S. Bhattacharyya, S. Jha, K. Tharakunnel and J.C. Westland, “Data Minining for Credit Card Fraud: A comparative study”, Decision Support Systems, Volume 50, pp 602 – 613, 2011

[8] D. Olszewski, “Fraud detection using Self-Organizing Map Visualizing the User Profiles”, Knowledge-Based Sys-tems, Volume 70, pp 324-334, 2014

[9] P. J. Rana and J. Baria, “A Survey on Fraud Detection Techniques in Ecom-merce”, International Journal of Com-puter Applications, Volume 113, No. 14, 2015

[10] A. Srivastava, A. Kundu, S. Sural and A. K. Majumdar, “Credit Card Fraud Detec-tion using Hidden Markov Model”, IEEE Transactions on dependable and secure computing, Volume 5, pp 37-48, 2008