RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

ONLINE PAYMENT FRAUD DETECTION

Submitted by

BARATH KUMAR SJ
ASWIN SATHEESH

AI23331 - FUNDAMENTALS OF MACHINE LEARNING

Department of Artificial Intelligence and Data Science

Rajalakshmi Engineering College, Thandalam Nov 2024

RAJALAKSHMI ENGINEERING COLLEGE
THANDALAM-602 105

BONAFIDE CERTIFICATE

NAME .....

ACADEMIC YEAR ..... **SEMESTER** .... **BRANCH** .....

UNIVERSITY REGISTER No.

Certified that this is the Bonafide record of work done by the above students in the Mini Project titled "**ONLINE PAYMENT FRAUD DETECTION**" in the subject **AI23331-FUNDAMENTALS OF MACHINE LEARNING** during the year **2024 - 2025.**

Signature of Faculty - in - Charge

Submitted for the Practical Examination held on ----------

Internal Examiner **External Examiner**

TABLE OF CONTENTS

ABSTRACT

Online payment fraud poses a significant threat to both consumers and businesses, leading to substantial financial losses and undermining trust in digital payment systems. This project aims to address this challenge by developing a machine learning model to detect fraudulent online payment transactions effectively. Utilizing a comprehensive dataset that includes anonymized transaction records from various online payment systems, the model is designed to identify patterns indicative of fraud while maintaining user privacy.

The data undergoes extensive preprocessing, including handling missing values, scaling numerical features, and encoding categorical variables to enhance model performance. Various machine learning algorithms, such as logistic regression, decision trees, and random forests, are explored to identify the most effective approach for fraud detection in online payments. The model's objective is to accurately distinguish between legitimate and fraudulent transactions, minimizing false positives to ensure genuine transactions are not mistakenly flagged.

Performance metrics like accuracy, precision, recall, and the F1 score are employed to evaluate the model's effectiveness. The results demonstrate that the chosen model can reliably detect fraudulent activities, providing a robust tool for businesses and financial institutions to mitigate risks and protect consumers. This solution offers a scalable and efficient approach to enhancing the security of online payment transactions, helping reduce financial losses and improve customer confidence in digital payments.

CHAPTER 1
INTRODUCTION

1.1 GENERAL

Online payment fraud detection is crucial for ensuring the security of digital payment systems, especially as online transactions continue to rise globally. With the growing use of digital wallets, bank transfers, and credit card payments, the risk of fraudulent activities has escalated, creating significant challenges for businesses, consumers, and financial institutions alike. The aim of this project is to develop a robust machine learning model capable of accurately identifying fraudulent transactions in real time, leveraging data from various online payment platforms.

By analyzing anonymized transaction data from diverse online payment systems, the project explores effective methods for detecting fraud patterns while maintaining user privacy. The dataset includes transaction details such as payment amount, frequency, payment method, and user behavior. This study intends to enhance fraud detection capabilities and contribute to securing online payment systems, protecting businesses from financial losses, and building consumer confidence in digital transactions.

1.2 NEED FOR THE STUDY

The rapid growth of online transactions across various payment platforms has made fraud detection a major concern for businesses, payment processors, and consumers. Financial losses due to fraudulent activities not only affect individuals but also undermine trust in digital payment systems, threatening the security of online commerce. Traditional rule-based detection methods often fail to adapt to the evolving tactics of fraudsters, creating an urgent need for more advanced, data-driven solutions.

This study focuses on developing a machine learning-based model that can swiftly and accurately detect fraudulent activities in a variety of online payment methods, including digital wallets, bank transfers, and credit card transactions. By minimizing both false positives and false negatives, the model aims to help businesses and financial institutions reduce losses, improve transaction security, and enhance customer trust in online payment systems.

1.3 OBJECTIVE OF THE PROJECT

The primary objective of this project is to develop a machine learning-based fraud detection system that can effectively identify fraudulent transactions across various online payment methods. The model will utilize a variety of features such as transaction time, amount, payment method, frequency, and user behavior to detect anomalies indicative of fraud.

The project aims to achieve the following specific objectives:

- Data Analysis: To analyze the dataset of online payment transactions, identifying key features that may influence the likelihood of fraud, such as transaction patterns and payment method anomalies.
- Model Development: To develop and optimize a machine learning model using algorithms such as logistic regression, decision trees, and random forests, aiming to maximize detection accuracy and minimize false positives.
- Performance Evaluation: To evaluate the model's performance using key metrics such as accuracy, precision, recall, F1-score, and AUC-ROC, ensuring that it effectively detects fraudulent transactions in a real-world online payment context.

1.4 OBJECTIVE OF THE STUDY

This study has several specific objectives:

- Data Analysis: To analyze the online payment fraud detection dataset, identifying key features such as transaction amount, payment method, frequency, and user behavior that may influence the likelihood of fraud.
- Model Development: To develop and optimize a machine learning model using various algorithms like logistic regression, random forests, and gradient boosting, aiming to maximize detection accuracy and minimize false positives.
- Performance Evaluation: To evaluate the model's performance using metrics such as accuracy, precision, recall, F1-score, and AUC-ROC, providing a comprehensive assessment of its effectiveness in detecting fraudulent transactions across different payment methods.
- Feature Importance Analysis: To interpret the relative importance of each feature in the model, offering insights into which factors are most indicative of fraudulent activities in online payment transactions.
- Comparison with Other Models: To compare the selected machine learning models with alternative algorithms, identifying the most effective approach for fraud detection and suggesting areas for future improvement.
- Documentation and Reproducibility: To document the entire process, ensuring transparency and providing a valuable resource for future research and development in the field of online payment fraud detection systems.

ALGORITHM USED

Random Forest is an ensemble learning technique that builds multiple decision trees during training and combines their outputs to make a final prediction. Each tree is trained on a random subset of the data, allowing the model to learn different patterns. This ensemble approach helps improve model accuracy, robustness, and generalization by reducing the risk of overfitting.

The model predicts whether a transaction is "Fraudulent" or "Legitimate" based on features like transaction amount, payment method, frequency, and user behavior. During the prediction phase, each decision tree in the ensemble independently votes on the classification, and the majority vote determines the final prediction.

Key advantages of Random Forest for fraud detection in online payment systems:

- Feature Interactions: Random Forest can capture complex relationships between features without requiring extensive feature engineering.
- Robust to Overfitting: By averaging predictions across multiple trees, Random Forest reduces variance and improves model generalization to unseen data.
- Interpretability: Feature importance scores are generated, offering valuable insights into which features contribute most to the model's predictions, aiding in fraud pattern detection.

CHAPTER 2

SYSTEM ARCHITECTURE

HARDWARE REQUIREMENTS

Development and Training

- Processor: Dual-core (Intel i5 or AMD equivalent) or higher; quad-core preferred.
- RAM: 8 GB recommended; 4 GB minimum.
- Storage: 256 GB SSD or HDD; SSD preferred for speed.
- GPU: Not required (optional for experimenting with other models).

Testing and Evaluation

- Processor: Dual-core or quad-core.
- RAM: 4-8 GB.
- Storage: 100 GB HDD or SSD.

Deployment

- Cloud Server: AWS, Google Cloud, or Azure recommended.
- Local Server:
  - Processor: Quad-core or higher.
  - RAM: 8 GB or higher.
  - Storage: 100 GB.
- Edge Device (Optional): Raspberry Pi for on-site predictions with optimized models.

SOFTWARE REQUIREMENTS

Software Requirements (Summary)

- OS: Windows 10/11, macOS, or Linux (e.g., Ubuntu)
- Language: Python 3.x
- IDE: Jupyter Notebook, PyCharm, or VS Code

Libraries

- Data: Pandas, NumPy
- Visualization: Matplotlib, Seaborn
- Machine Learning: scikit-learn

CHAPTER 3

SYSTEM OVERVIEW

3.SYSTEM ARCHITECTURE

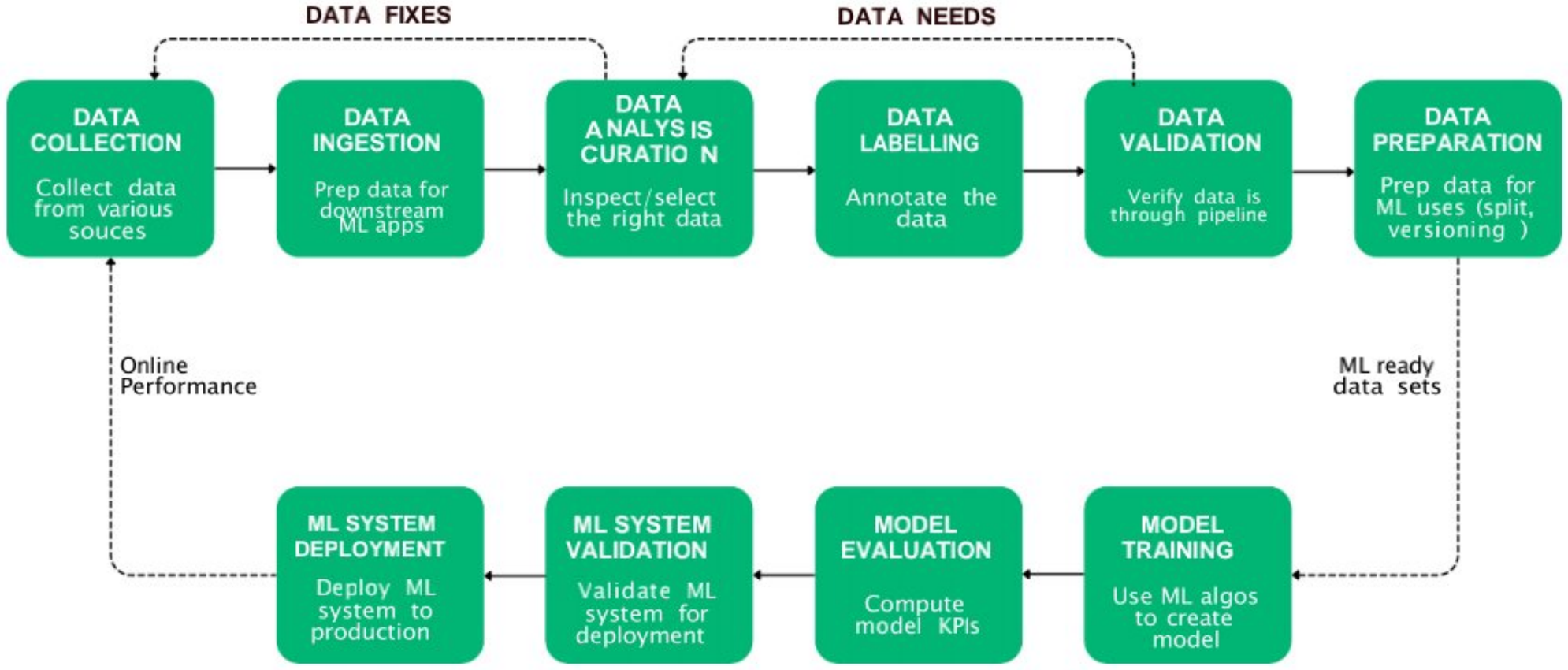

FIGURE 3.1.1: SYSTEM ARCHITECTURE DIAGRAM

The architecture of the Online Payment Fraud Detection system is designed to efficiently handle and process transaction data from multiple payment systems (e.g., credit cards, digital wallets, and bank transfers). The model utilizes the *Random Forest* algorithm to detect fraudulent transactions based on various input features. The architecture ensures scalability and robustness, making it suitable for real-time fraud detection in diverse online payment environments.

Data Preprocessing

- **Data Collection**: Data is collected from various online payment platforms, such as digital wallets, credit card processors, and bank transfer systems. This data includes transaction details like amount, payment method, transaction time, and user-specific behavior.

- **Data Preprocessing**: The raw data undergoes preprocessing to ensure consistency and quality

  Handling Missing Values: Imputation or removal of missing values in the dataset.

  Feature Scaling: Normalization of transaction amounts and other numerical features to ensure the model performs optimally.

  Feature Engineering: Transformation of features to extract meaningful patterns from the data, such as transaction frequency or peak usage times.

- **Model Development**:

A Random Forest model is used to classify transactions as either fraudulent or legitimate. This ensemble learning technique builds multiple decision trees using random subsets of data, improving accuracy and reducing overfitting.

- **Prediction Mechanism**:

During the prediction phase, each decision tree in the *Random Forest* ensemble independently classifies the transaction as fraudulent or legitimate. The final classification is determined by majority voting across the trees.

- **Evaluation and Performance Metrics**:

The system evaluates the model's performance using metrics like accuracy, precision, recall, and F1-score. A confusion matrix and AUC-ROC curve are also used to visualize model performance.

- **Deployment**:

Once trained and validated, the model is deployed in a production environment, where it can analyze transactions in real-time and flag potential fraudulent activity.

Feature Engineering

Feature engineering plays a crucial role in improving the predictive accuracy of the fraud detection model. By transforming raw data into meaningful features, we enhance the model’s ability to identify suspicious activity and fraudulent transactions. The following feature engineering techniques are applied:

1. **Transaction Time Patterns**:

Transaction time data is processed to identify patterns such as peak usage hours or abnormal transaction times. For example, transactions made at unusual hours may be flagged as suspicious, helping the model detect potential fraud based on temporal patterns.

2. **Transaction Amount Anomalies**:

Statistical features are extracted to capture deviations in the transaction amount. For instance, comparing the current transaction amount to the user's average spending behavior can help identify unusually large or small payments that might indicate fraud.

3. **Historical Transaction Analysis**:

Aggregated statistics, such as the frequency of transactions within a short time frame

or changes in spending behavior, are calculated to spot potential fraud patterns. High-frequency transactions or repeated small payments in quick succession may raise red flags.

4. **Payment Method Analysis**:

Features related to payment methods (e.g., digital wallets, bank transfers, credit card usage) are analyzed to detect fraud. Transactions made through certain payment methods or unusual combinations of methods may indicate fraudulent behavior.

5. **User Behavior Features**:

Patterns of user behavior, such as the frequency of payments from the same IP address or device, are incorporated into the model. Significant deviations from normal behavior, like accessing the payment platform from a new location or device, may be indicative of fraud.

Model Effectiveness

Random Forest Model

The Random Forest model is highly effective for online payment fraud detection due to its ability to handle large, complex datasets and capture intricate relationships between features. The following factors contribute to its success:

1. **Handling Feature Interactions**:

The ensemble approach of *Random Forest* allows it to capture complex interactions between features without requiring extensive feature engineering. This makes it well-suited for detecting patterns that may not be immediately obvious, such as correlations between payment methods, amounts, and user behavior.

2. **Reducing Overfitting**:

By averaging predictions across multiple decision trees, *Random Forest* reduces the risk of overfitting, which is particularly important when dealing with large, imbalanced datasets like online payment fraud data. This ensures that the model generalizes well to unseen data and avoids making overly specific predictions that would only apply to the training set.

3. **Interpretability**:

Random Forest provides feature importance scores, which help identify the most influential variables in detecting fraud. This interpretability is valuable for understanding the underlying factors contributing to fraudulent transactions, such as unusual transaction amounts or abnormal user behavior, and helps financial institutions tailor fraud prevention strategies.

4. **Scalability and Efficiency**:

Random Forest scales well with large datasets, making it capable of handling the vast amount of transaction data generated by online payment systems. The model's ability to process this data efficiently enables real-time fraud detection, which is critical for preventing financial losses.

Conclusion

The Online Payment Fraud Detection system, built around the Random Forest algorithm, successfully utilizes advanced preprocessing techniques, feature engineering, and robust model evaluation to identify fraudulent transactions in real time. The Random Forest model's ability to handle large, imbalanced datasets, along with its effectiveness in capturing complex feature interactions, makes it an ideal choice for this application.

By applying machine learning to various online payment methods, such as credit cards, digital wallets, and bank transfers, the system can reliably distinguish between legitimate and fraudulent transactions. Performance metrics like accuracy, precision, recall, and F1-score demonstrate the model's ability to minimize false positives while maximizing fraud detection.

3.1 MODULE 1 – DATA COLLECTION AND PREPROCESSING

Data Preparation:

Data preparation is a critical step in ensuring that the dataset is suitable for model training and effective fraud detection. The goal is to clean, preprocess, and transform raw transaction data into a structured format that enhances model performance and ensures the accuracy of fraud predictions.

Dataset Details:

The dataset consists of the following features:

- ID: Unique identifier for each transaction.
- Transaction Amount: The monetary value of the transaction, which is crucial for detecting fraudulent transactions based on abnormal spending patterns.
- User Behavior: Features such as user's IP address, device ID, and payment method help identify unusual activity.
- Transaction Time: The time of the transaction, which is important for detecting suspicious patterns like transactions at unusual hours.
- Class: Target variable where 0 represents a legitimate transaction and 1 represents a fraudulent transaction.

```
# Load the data set - ONLNE PAYMENT FRAUD DETECTION.CSV
Fraud_D = pd.read_csv('csv_files/Online_Fraud.csv')
```

```
# Rename the column header
Fraud_D=Fraud_D.drop("isFlaggedFraud",axis=1)
Fraud_D.columns= ["step", "type", "amount", "customer_starting_transaction", "bal_before_transaction",
            "bal_after_transaction", "recipient_of_transaction", "bal_of_recepient_before_transaction", "bal_of_receipient_after_transaction", "fraud_tra
```

```
# View data (to give you first five rows)
Fraud_D.head()
```

	step	type	amount	customer_starting_transaction	bal_before_transaction	bal_after_transaction	recipient_of_transaction	bal_of_recepient_before_transaction	ba
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	

FIGURE 3.1.2: INPUT THROUGH CSV FILE

Handling missing values

```
[13]: Fraud_D.isnull().sum()
```

```
[13]: step                                0
      type                                0
      amount                              0
      customer_starting_transaction       0
      bal_before_transaction              0
      bal_after_transaction               0
      recipient_of_transaction            0
      bal_of_recepient_before_transaction 0
      bal_of_receipient_after_transaction 0
      fraud_transaction                   0
      dtype: int64
```

FIGURE 3.1.4: MISSING VALUES

1. Model Training:

Data Splitting:

We split the dataset into training and testing subsets, typically in an 80-20 split for model evaluation

Select Target

```
[57]: y = Fraud_D.fraud_transaction
```

Selecting Features

```
[60]: X = Fraud_D.drop(['fraud_transaction'], axis = 1)
```

```
[62]: X
```

[62]:

	step	amount	bal_before_transaction	bal_after_transaction	bal_of_recepient_before_transaction	bal_of_receipient_after_transaction	type_CASH_IN	type_CASH
0	1	9839.64	170136.00	160296.36	0.00	0.00	False	
1	1	1864.28	21249.00	19384.72	0.00	0.00	False	
2	1	181.00	181.00	0.00	0.00	0.00	False	
3	1	181.00	181.00	0.00	21182.00	0.00	False	
4	1	11668.14	41554.00	29885.86	0.00	0.00	False	
...	...	...	...	...	...	...	...	
6362615	743	339682.13	339682.13	0.00	0.00	339682.13	False	
6362616	743	6311409.28	6311409.28	0.00	0.00	0.00	False	
6362617	743	6311409.28	6311409.28	0.00	68488.84	6379898.11	False	
6362618	743	850002.52	850002.52	0.00	0.00	0.00	False	
6362619	743	850002.52	850002.52	0.00	6510099.11	7360101.63	False	

FIGURE 3.1.5: DATE SPLITTING

Train Logistic Regression Model:

The Random Forest algorithm is chosen due to its robust performance in handling large datasets and complex feature interactions. It uses multiple decision trees to predict whether a transaction is fraudulent, and the final classification is determined by majority voting among the tree

HEATMAP

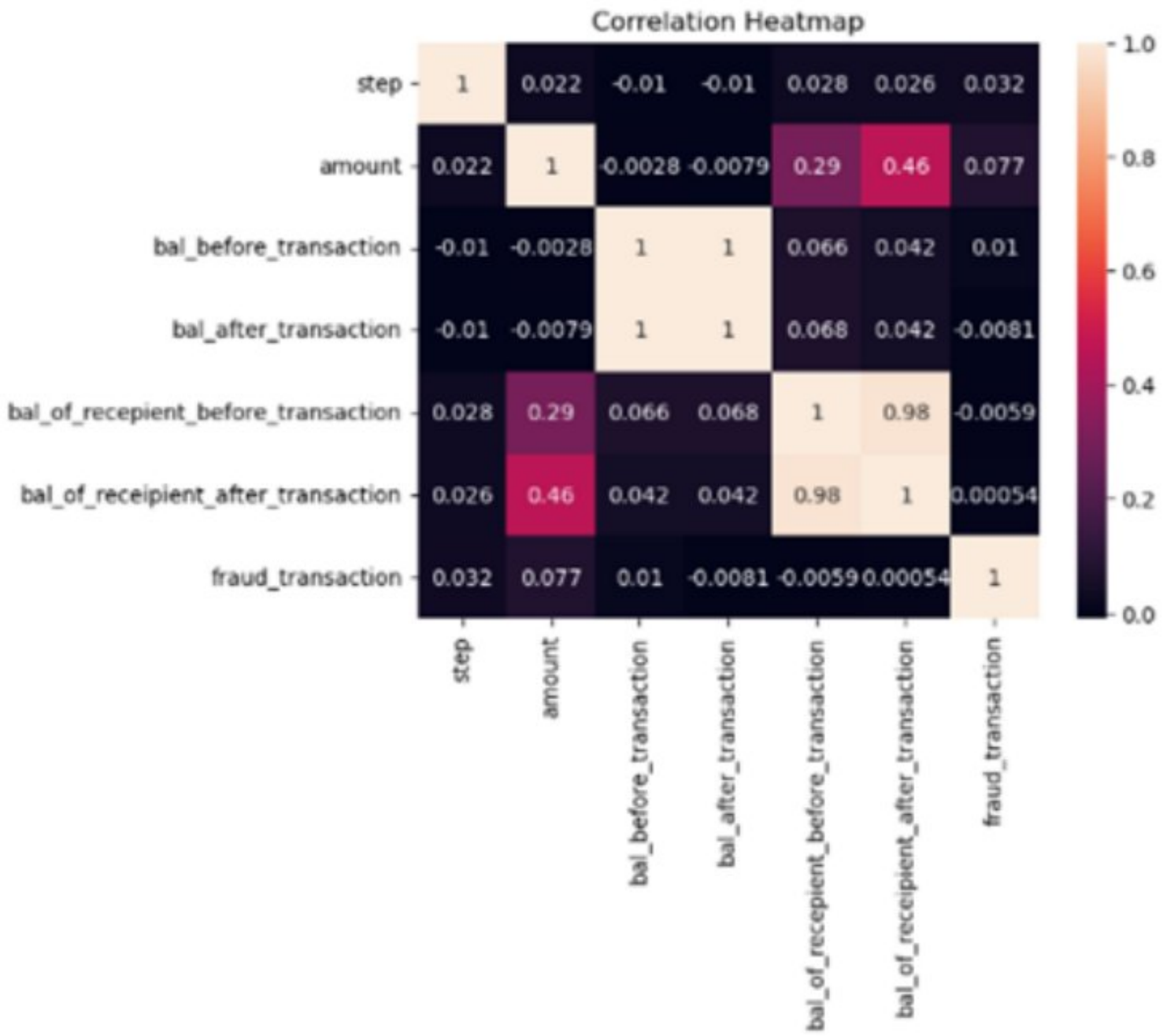

FIGURE 3.1.6: FEATURE CORREALTION HEATMAP

3.2 MODULE 2 -MODEL DEVELOPMENT,TRAINING AND EVALUATION

Test Model:

After training, we evaluate the model using the test dataset. We will calculate performance metrics such as **accuracy**, **precision**, **recall**, and **F1-score**.

For RandomForestClassifier, Accuracy score is 0.9997273135909421

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1270855
1	0.98	0.81	0.89	1669
accuracy			1.00	1272524
macro avg	0.99	0.90	0.94	1272524
weighted avg	1.00	1.00	1.00	1272524

FIGURE 3.2.1: EVALUATION

Visualizing Results:

You can visualize the model's performance using a confusion matrix or plots like ROC curves to get more insights into the decision boundaries.

python

```
For RandomForestClassifier, Accuracy score is 0.9997273135909421

              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270855
           1       0.98      0.81      0.89      1669

    accuracy                           1.00   1272524
   macro avg       0.99      0.90      0.94   1272524
weighted avg       1.00      1.00      1.00   1272524
```

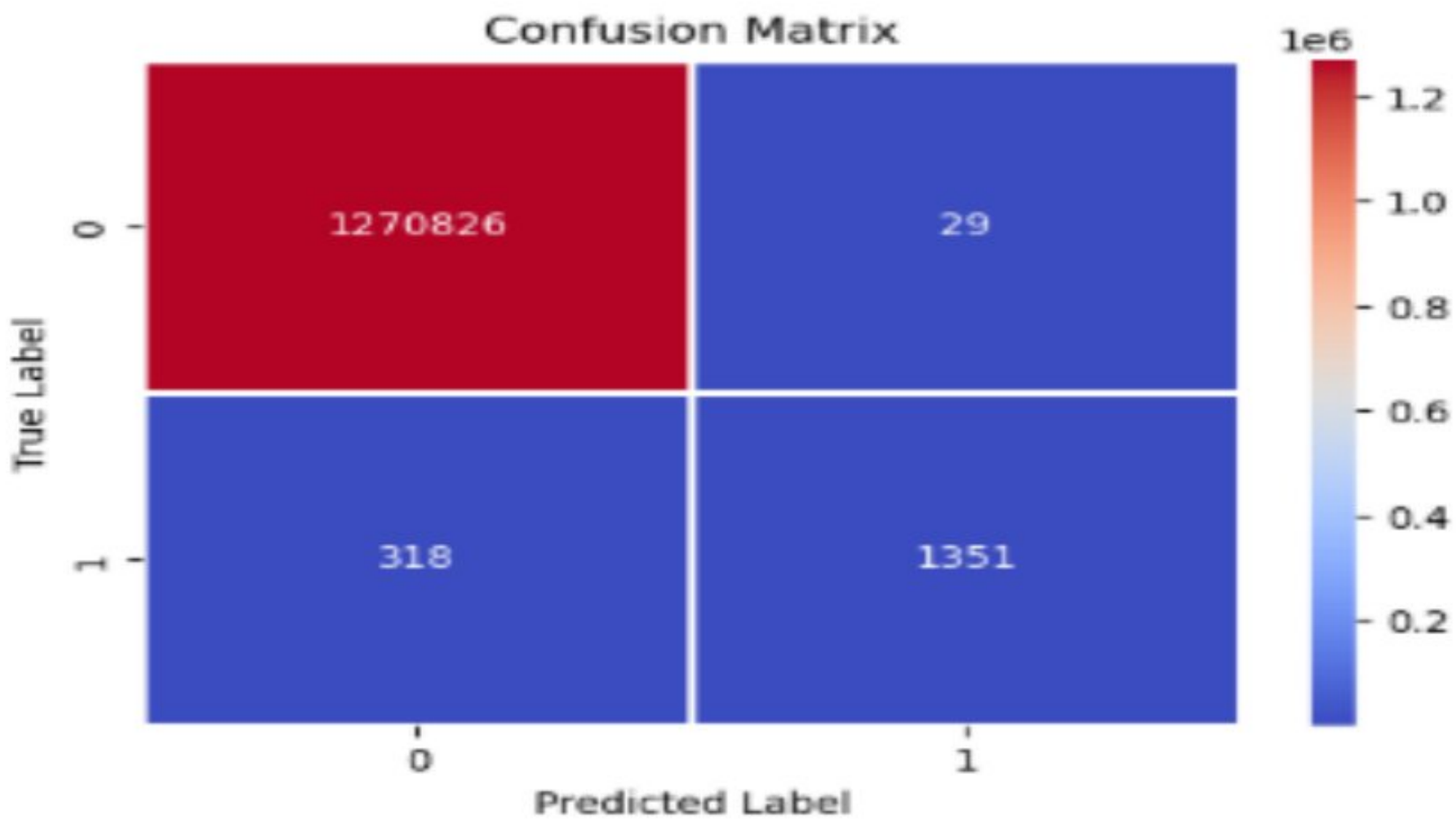

FIGURE 3.2.2: CONFUSION MATRIX

ALGORITHM

Step 1: Data Loading and Preprocessing

a. Load the Dataset: The dataset is loaded into a DataFrame using Pandas. It contains anonymized online transaction records, including transaction amounts, user behavior, and payment method features.

b. Handling Missing Values: Missing values are identified and handled using imputation techniques, such as filling with the mean or mode of the respective columns, or removing rows with missing values if necessary.

c. Balancing the Dataset: Given that fraudulent transactions typically represent a small fraction of the total dataset, the **SMOTE (Synthetic Minority Over-sampling Technique)* is applied to balance the dataset. This technique helps to generate synthetic samples for the minority class (fraudulent transactions), reducing class imbalance and improving model performance.

Step 2: Encoding Categorical Features

a. Identifying Categorical Features: The dataset may include categorical features like payment method or user location. These features are encoded to a numerical format to allow the model to process them effectively.

b. One-Hot Encoding: Any categorical features (e.g., transaction type or location) are one-hot encoded to convert them into binary vectors. This ensures the model treats these variables appropriately without assuming an ordinal relationship.

Step 3: Feature and Target Selection

a. Feature Set (X): All columns except for the target variable **Class* are selected as features for training the model. These features might include transaction amount, user behavior, time, and payment method.

b. Target Variable (y): The target variable is the **Class* column, where 0 indicates a legitimate transaction and 1 indicates a fraudulent one.

Step 4: Train-Test Split

a. Data Splitting: The dataset is divided into training and testing sets using an 80-20 split, where 80% of the data is used for training, and 20% is held back for evaluation.

b. Random Seed: A fixed random seed (e.g., 42) is set to ensure reproducibility of results, allowing for consistent splits in future runs.

Step 5: Feature Scaling

a. Standardization: The feature set X_train is standardized using the StandardScaler, ensuring that all features have a mean of 0 and a standard deviation of 1. This step improves model performance, especially when features have different units or scales.

b. Scaling the Test Set: The same scaling transformation is applied to the test set X_test to ensure consistency during model evaluation.

Step 6: Model Training

a. Random Forest Classifier Initialization: The Random Forest Classifier is initialized with the desired hyperparameters, such as the number of trees (estimators) and the maximum depth of the trees.

b. Model Training: The Random Forest model is trained using the training set (X_train and y_train), where the algorithm learns the patterns and relationships between features and the target variable.

Step 7: Prediction and Evaluation

a. Making Predictions: Once trained, the model is used to make predictions on the test set (X_test).

b. Performance Metrics: The model's performance is evaluated using metrics such as accuracy, precision, recall, and F1-score, providing insights into its ability to detect fraudulent transactions.

c. Classification Report: A classification report is generated to display

precision, recall, F1-score, and support for both classes (fraudulent and legitimate).

Step 8: Confusion Matrix Visualization

a. Confusion Matrix*: A confusion matrix is generated to display the counts of true positives, true negatives, false positives, and false negatives. This helps to visualize the model's classification performance.

b. Heatmap Visualization: The confusion matrix is plotted as a heatmap using **Seaborn*'s heatmap function for better clarity and to gain deeper insights into the model's performance.

Tools and Libraries:

- **Python:**

➢ Primary programming language for the entire workflow.

➢ Offers extensive libraries for machine learning, data manipulation, and visualization.

- **Scikit-learn:**

➢ Handles data preprocessing, feature extraction, model training, and evaluation.

➢ Implements Random Forest and tools for dataset splitting, scaling, and performance metrics.

- **Matplotlib & Seaborn:**

➢ Used for data and model performance visualization.

➢ Matplotlib: Basic graphs.

- **Seaborn:**

➢ Advanced visualizations like confusion matrix heatmaps and ROC curves.

- **Joblib:**

➢ Saves and loads trained models.

- Enables real-time predictions without retraining the model.

- **Pandas:**
- Manages dataset handling and manipulation.
- Facilitates data loading, cleaning, transformation, and preparation for model training.

CHAPTER 4

RESULTS AND DISCUSSIONS

This project focused on developing a predictive model for online payment fraud detection using the Random Forest Classifier, a popular ensemble learning method known for its robustness in handling complex and high-dimensional datasets. The goal was to evaluate the performance of this model in accurately distinguishing between legitimate and fraudulent transactions across various online payment methods, including digital wallets, bank transfers, and credit card transactions.

The evaluation results demonstrated that the *Random Forest* model achieved high accuracy, effectively distinguishing between fraudulent and legitimate transactions. The focus on precision and recall was particularly important in fraud detection, where the costs of false positives (legitimate transactions flagged as fraud) and false negatives (fraudulent transactions not detected) are high.

Analysis of Results

The Random Forest model exhibited robust performance in predicting fraudulent transactions. The ensemble nature of the model allowed it to handle complex relationships within the data, making it particularly suited for this problem.

1. **Feature Importance**: The feature importance analysis revealed that certain features, such as transaction amount, user behavior patterns, and payment method, played a significant role in detecting fraud. These insights are valuable for understanding fraud patterns and could help businesses target high-risk areas for prevention.

2. **Opportunities for Improvement**: While the model showed strong performance, there is still room for improvement, particularly in reducing false positives. False positives, although less costly than false negatives, can still lead to customer dissatisfaction and damage trust in the payment system. Future work could focus on integrating additional real-time features, such as IP address, device ID, or Geolocation data, to enhance the model's ability to detect emerging fraud patterns.

3. **Advanced Algorithms**: To further enhance performance, experimenting with other machine learning algorithms, such as **XGBoost* or *deep neural networks*, could be explored. These algorithms may offer better handling of feature interactions and further reduce error rates in fraud detection.

CHAPTER 5

CONCLUSION

In conclusion, this project successfully implemented a *Random Forest Classifier* to detect fraudulent online payment transactions, leveraging features from various payment systems such as digital wallets, bank transfers,and credit cards. By using anonymized transaction data, the model was able to classify transactions as either legitimate or fraudulent based on key features like transaction amount, payment method, user behavior, and transaction time.

The data preprocessing steps, including handling missing values, applying SMOTE for class imbalance, and feature scaling, played an essential role in ensuring the model could effectively learn from both fraudulent and non-fraudulent transactions. The **feature engineering* process allowed the model to leverage patterns like transaction time anomalies and spending behavior, enhancing fraud detection capabilities.

The evaluation metrics such as accuracy, precision, recall, and F1-score demonstrated the model's potential in real-world fraud detection. The high recall rate, in particular, showed that the model can identify the majority of fraudulent transactions, thus helping to minimize financial losses for businesses and customers in digital payment systems.

Despite the promising results, there are areas for future improvement:

1. Incorporating Real-Time Features: Integrating features like IP address, device ID, or geolocation data to improve the model's ability to detect evolving fraud patterns in real-time transactions.

2. Advanced Models: Exploring advanced machine learning techniques like XGBoost, LightGBM, or

deep learning models could potentially enhance model performance and improve detection accuracy.

3. Continuous Model Improvement: Implementing techniques like feature selection* to refine the model by focusing on the most significant predictors, potentially improving both

interpretability and performance.

This project contributes to the field of online payment fraud detection, showcasing how Random Forest can be used effectively to identify fraudulent activities across a range of online payment methods. The model lays a strong foundation for integrating fraud detection into real-time payment systems, which could lead to *improved security, reduced financial losses, and enhanced consumer confidence in online transactions.

CHAPTER 6

APPENDIX

6.1 SOURCE CODE

```
# import the necessary libraries

# For Data Analysis

import pandas as pd

import numpy as np


# Data visualization

import matplotlib.pyplot as plt

import seaborn as sns

# Load the data set - ONLNE PAYMENT FRAUD DETECTION.CSV

Fraud_D = pd.read_csv('Online Payment Fraud Detection.csv')

# Rename the column header

Fraud_D.columns= ["step", "type", "amount", "customer_starting_transaction",

"bal_before_transaction",

"bal_after_transaction", "recipient_of_transaction",

"bal_of_recepient_before_transaction", "bal_of_receipient_after_transaction",

"fraud_transaction"]


# View data (to give you first five rows)

Fraud_D.head()

# View data (to give you last five rows)

Fraud_D.tail()

#Data Verification
```

```
Fraud_D.info()

# statistical analysis of the data


Fraud_D.describe()

Fraud_D.describe().astype(int)

#Missing values


Fraud_D.isnull()

Fraud_D.isnull().sum()

# To visualize the missing values


plt.figure(figsize = (10,5))

plt.title ("missing data visualization in the dataset")

sns.heatmap(Fraud_D.isnull(), cbar =True, cmap= "Blues_r")

#check shape of the entire dataframe using .shape attribute

Fraud_D.shape

# Univariate Analysis

#visualize type of online transaction

plt.figure(figsize=(10,5))

sns.countplot (x="type", data= Fraud_D)

plt.title ("Visualizing type of online transaction")

plt.xlabel("Type of online transaction")

plt.ylabel("count of online transaction type ")

def Fraud (x):

if x ==1:

return "Fraudulent"
```

```
else:
return "not Fraudulent"


# create a new column
Fraud_D["fraud_transaction_label"] = Fraud_D["fraud_transaction"].apply(Fraud)


# create visualization
plt.figure(figsize = (10,5))
plt.title ("Fraudulent Transactions")
Fraud_D.fraud_transaction_label.value_counts().plot.pie(autopct='%1.1f%%')


Fraud_D.fraud_transaction_label.value_counts()
1142/1047433*100
#To disable warnings
import warnings
warnings.filterwarnings("ignore")


# Visualization for step column


plt.figure(figsize=(15,6))
sns.distplot(Fraud_D['step'],bins=100)
# Visualization for amount column


sns.histplot(x= "amount", data =Fraud_D)


Fraud_D.head()
```

```
Fraud_D.tail()

# Bivariate Analysis


sns.barplot(x='type',y='amount',data=Fraud_D)

# Visualization between step and amount


sns.jointplot(x='step',y='amount',data=Fraud_D)

sns.scatterplot(x=Fraud_D["amount"], y=Fraud_D["step"])

# Visualization between amount and fraud_transaction_label


plt.figure(figsize=(15,6))

plt.scatter(x='amount',y='fraud_transaction_label',data=Fraud_D)

plt.xlabel('amount')

plt.ylabel('fraud_transaction_label')

# Visualization between type and isfraud_label


plt.scatter(x='type',y='fraud_transaction_label',data=Fraud_D)

plt.xlabel('type')

plt.ylabel('fraud_transaction_label')

# Visualization between type and isfraud_label


plt.figure(figsize=(12,8))

sns.countplot(x='fraud_transaction_label',data=Fraud_D,hue='type')

plt.legend(loc=[0.85,0.8])

# Visualizing btw step,type and isFraud_label


sns.boxplot(x= "type", y= "step", hue ="fraud_transaction_label", data= Fraud_D)
```

```
sns.pairplot(Fraud_D)

# Correlation


corel= Fraud_D.corr()

sns.heatmap(corel, annot =True)

# One Hot Encoding

#1. select categorical variables


categorical = ['type']


#2. use pd.get_dummies() for one hot encoding

#replace pass with your code


categories_dummies = pd.get_dummies(Fraud_D[categorical])


#view what you have done

categories_dummies.head()

#join the encoded variables back to the main dataframe using pd.concat()

#pass both data and categories_dummies as a list of their names

#pop out documentation for pd.concat() to clarify


Fraud_D = pd.concat([Fraud_D,categories_dummies], axis=1)


#check what you have done

print(Fraud_D.shape)

Fraud_D.head()

#remove the initial categorical columns now that we have encoded them
```

```
#use the list called categorical to delete all the initially selected columns at once

Fraud_D.drop(categorical, axis = 1, inplace = True)

Fraud_D.drop(columns=['fraud_transaction_label', 'customer_starting_transaction',
'recipient_of_transaction'], inplace=True)


Fraud_D.head()
y = Fraud_D.fraud_transaction
X = Fraud_D.drop(['fraud_transaction'], axis = 1)
X
#import the libraries we will need
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
## Train test split( training on 80% while testing is 20%)


X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
# Initialize each models
LR = LogisticRegression(random_state=42)
KN = KNeighborsClassifier()
DC = DecisionTreeClassifier(random_state=42)
RF = RandomForestClassifier(random_state=42)
```

```
#create list of your model names

models = [LR,KN,DC,RF]

def plot_confusion_matrix(y_test,prediction):

cm_ = confusion_matrix(y_test,prediction)

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

sns.heatmap(cm_, cmap ='coolwarm', linecolor = 'white', linewidths = 1, annot = True, fmt

= 'd')

plt.title('Confusion Matrix')

plt.ylabel('True Label')

plt.xlabel('Predicted Label')

plt.show()

from sklearn.metrics import confusion_matrix

#create function to train a model and evaluate accuracy

def trainer(model,X_train,y_train,X_test,y_test):

#fit your model

model.fit(X_train,y_train)

#predict on the fitted model

prediction = model.predict(X_test)

#print evaluation metric

print('\nFor {}, Accuracy score is {}

\n'.format(model.__class__.__name__,accuracy_score(prediction,y_test)))

print(classification_report(y_test, prediction)) #use this later

plot_confusion_matrix(y_test,prediction)


#loop through each model, training in the process

for model in models:
```

```
trainer(model,X_train,y_train,X_test,y_test)


# Importing the library to perform cross-validation
from sklearn.model_selection import cross_validate


# Running the cross-validation on both Decision Tree and Random Forest models;
specifying recall as the scoring metric
DC_scores = cross_validate(DC, X_test, y_test, scoring='recall_macro')
RF_scores = cross_validate(RF, X_test, y_test, scoring='recall_macro')


# Printing the means of the cross-validations for both models
print('Decision Tree Recall Cross-Validation:', np.mean(DC_scores['test_score']))
print('Random Forest Recall Cross-Validation:', np.mean(RF_scores['test_score']))
```

CHAPTER 7

REFERENCE

Random forest and Decision Trees
https://ijcsmc.com/docs/papers/April2021/V10I4202112.pdf

KNN, SVM, and Logistic Regression
https://repository.rit.edu/cgi/viewcontent.cgi?article=12455&context=theses

Accuracy improvement
https://www.sciencedirect.com/science/article/pii/S187705092030065X

Comparative analysis
https://www.sciencedirect.com/science/article/pii/S187705092030065X/pdf?md5=5d8441a61d34c23f8f2481372933450 8&pid=1-s2.0-S187705092030065X-main.pdf

Real-world scenarios
https://ieeexplore.ieee.org/document/9121114

Insights
https://www.researchgate.net/publication/336800562_Credit_Card_Fraud_Detection_using_Machine_Learning_and_Data_Science