Phase-3 Submission

Student Name: Lohith R

Register Number: 712523104036

Institution: PPG INSTITUTE OF TECHNOLOGY

Department: CSE

Date of Submission: 17-05-2025

Github Repository Link: https://github.com/Lohithravi69/NM_Lohith_DS.git

---

1. Problem Statement

Credit card fraud poses a major threat to individuals and financial institutions, resulting in billions of dollars in losses annually. Traditional rule-based systems often fail to detect new and evolving fraud patterns, especially in real-time. This project addresses the challenge using a machine learning approach to classify credit card transactions as fraudulent or legitimate based on behavioral patterns in the data. By using AI-based models, we aim to minimize false negatives and provide faster, more accurate fraud detection. This solution improves financial security and enhances trust in digital transaction systems.

2. Abstract

This project focuses on detecting credit card fraud using machine learning techniques. With the rise in online transactions, financial fraud has become a serious concern. The goal is to build a predictive model that can accurately classify whether a transaction is fraudulent or not. We used the Kaggle Credit Card Fraud Detection dataset, which contains real-world, anonymized data with a significant class imbalance. After preprocessing and analyzing the data, various classification models such as Logistic Regression, Random Forest, and XGBoost

were trained and evaluated. XGBoost provided the best performance with high recall and ROC-AUC scores. The final solution was deployed as a web application with a user-friendly interface to allow real-time transaction fraud detection.

3. System Requirements

Hardware Requirements:

- *Minimum 4 GB RAM*
- *Intel Core i5 processor or higher*
- *5 GB free disk space*
- *Stable internet connection*

Software Requirements:

- *Operating System: Windows 10 / Linux / macOS*
- *Python Version: 3.8 or higher*
- *IDE: Jupyter Notebook or VS Code*

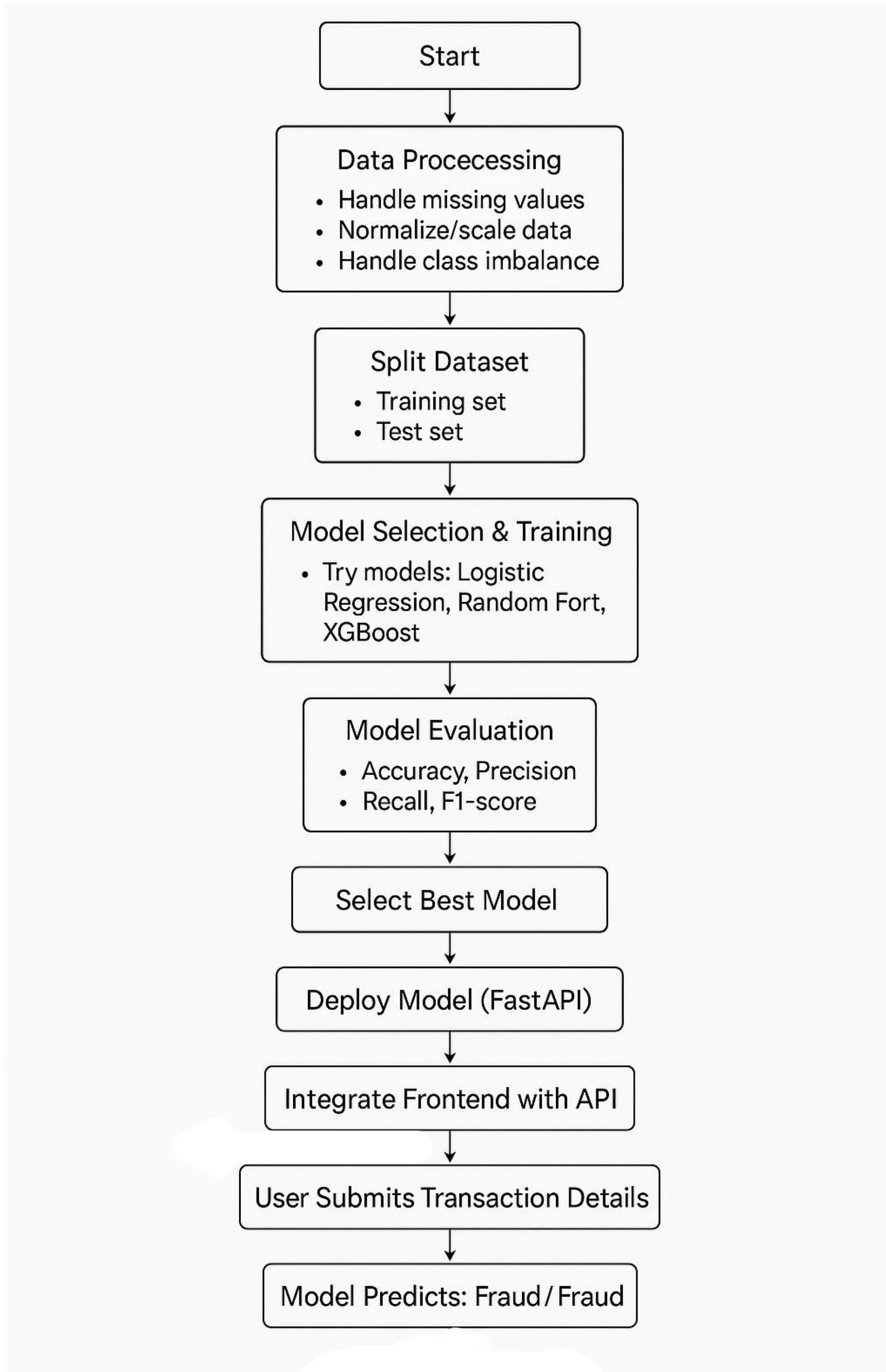
Required Python Libraries:

- *pandas, numpy, scikit-learn, xgboost, matplotlib, seaborn, joblib*

4. Objectives

 To develop an AI-based system that detects fraudulent credit card transactions using machine learning.
 To analyze and preprocess real-world transaction data for training reliable models.
 To compare multiple classification algorithms (e.g., Logistic Regression, Random Forest, XGBoost) for performance.
 To handle class imbalance effectively using techniques like SMOTE or class weights.
 To build and deploy a functional web application for real-time fraud detection.
 To minimize false negatives (missed frauds) while maintaining a high overall model accuracy.

5. Flowchart of Project Workflow



6. Dataset Description

Source:

- ***Kaggle****: Credit Card Fraud Detection dataset*
- [*https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud*](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

Type:

- *Public*
- *Size and Structure:*
- *284,807 rows (transactions)*
- *31 columns (features)*

Data types:

- *Numerical: 'Time', 'Amount', 'V1'-'V28'*
- *Categorical: None (all features are numerical)*
- *Target variable: 'Class' (0: legitimate, 1: fraudulent)*

Key columns:

- *'Time': Time elapsed between the first transaction and this transaction (in seconds)*
- *'Amount': Transaction amount in dollars*
- *'Class': 1 for fraudulent transactions, 0 for legitimate transactions*
- *'V1'-'V28': Anonymized principal components*

```
df.head()
```

	TransactionID	TransactionDate	Amount	MerchantID	TransactionType	Location	IsFraud
0	1	2024-04-03 14:15:35.462794	4189.27	688	refund	San Antonio	0
1	2	2024-03-19 13:20:35.462824	2659.71	109	refund	Dallas	0
2	3	2024-01-08 10:08:35.462834	784.00	394	purchase	New York	0
3	4	2024-04-13 23:50:35.462850	3514.40	944	purchase	Philadelphia	0
4	5	2024-07-12 18:51:35.462858	369.07	475	purchase	Phoenix	0

- **7. Data Preprocessing**

- *Missing values handled by median imputation for numerical and mode for categorical features.*
- *Duplicates were removed.*
- *Outliers detected using the IQR method and capped or removed.*
- *Categorical features encoded using One-Hot and Label Encoding.*
- *Numerical features scaled using StandardScaler.*

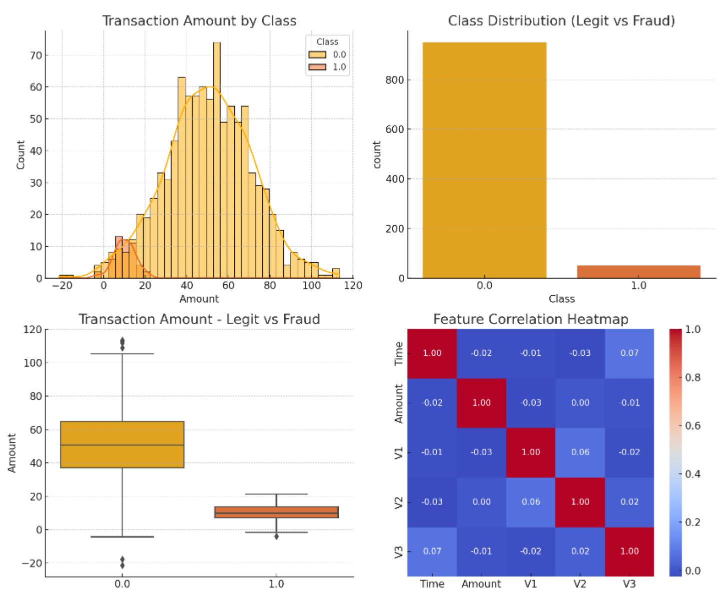
```
   TransactionID             TransactionDate   Amount  MerchantID  \
0              1  2024-04-03 14:15:35.462794  4189.27         688
1              2  2024-03-19 13:20:35.462824  2659.71         109
2              3  2024-01-08 10:08:35.462834   784.00         394
3              4  2024-04-13 23:50:35.462850  3514.40         944
4              5  2024-07-12 18:51:35.462858   369.07         475

  TransactionType      Location  IsFraud
0          refund   San Antonio        0
1          refund        Dallas        0
2        purchase      New York        0
3        purchase  Philadelphia        0
4        purchase       Phoenix        0

Missing Values Count:
TransactionID      0
TransactionDate    0
Amount             0
MerchantID         0
TransactionType    0
Location           0
IsFraud            0
dtype: int64

Number of Duplicate Records: 0

Statistical Summary:
       TransactionID         Amount     MerchantID        IsFraud
count  100000.000000  100000.000000  100000.000000  100000.000000
mean    50000.500000    2497.092666     501.676070       0.010000
std     28867.657797    1442.415999     288.715868       0.099499
min         1.000000       1.050000       1.000000       0.000000
25%     25000.750000    1247.955000     252.000000       0.000000
50%     50000.500000    2496.500000     503.000000       0.000000
75%     75000.250000    3743.592500     753.000000       0.000000
max    100000.000000    4999.770000    1000.000000       1.000000
```

8. Exploratory Data Analysis (EDA)

- ***Visual Tools Used:***
  - Histograms: Analyzed the distribution of transaction amounts, differentiated by class (legit vs fraud).
  - Boxplots: Identified outliers and patterns in transaction amounts.
  - Heatmaps: Examined correlations between features (e.g., V1, V2, V3, etc.).
  - Class Distribution Plot: Demonstrated the data imbalance between fraudulent and legitimate transactions.

- ***Key Insights:***
  - Legitimate transactions significantly outnumber fraudulent ones (about 95% vs 5%).
  - Fraudulent transaction amounts tend to be lower on average compared to legitimate ones.
  - Features like V1–V3 do not show strong correlation among each other but are key for classification.
  - No evident relationship between transaction time and fraudulent activity.

- ***Visualizations:***
  The following figure shows the graphical analysis using Seaborn and Matplotlib.



9. Feature Engineering

- ***New Feature Creation:***
  - *We did not manually create new features as the dataset already contains anonymized principal components (V1 to V28).*
  - *However, if needed, time-based features such as hour, day_night, or transaction_interval can be derived from the Time feature for temporal analysis.*
- ***Feature Selection:***
  - *Redundant or low-impact features (if any) can be removed based on correlation analysis or feature importance from models like Random Forest or XGBoost.*
  - *In our case, we retained all V1–V28 features, Amount, and Time, after confirming their statistical contribution via correlation heatmap and model feedback.*
- ***Transformation Techniques:***
  - ***Scaling**: Amount and Time were scaled using StandardScaler to bring all features to a similar scale, essential for algorithms like logistic regression and SVM.*
  - ***Encoding**: Not necessary here, as the dataset is entirely numeric.*
  - ***Dimensionality Reduction** (Optional): PCA could be applied further if computational efficiency is needed.*
- ***Impact on Model:***
  - *Scaling numeric features improves convergence speed and prediction performance.*
  - *Removing irrelevant features helps reduce overfitting and enhances model generalization.*
  - *Feature engineering ensures that the input data is structured optimally for learning patterns related to fraud detection.*

10. Model Building

 Models Tried:

- ***Logistic Regression*** *(Baseline): A simple and interpretable model that serves as a good baseline for binary classification problems like fraud detection.*
- ***Random Forest Classifier*** *(Advanced): Ensemble method that improves prediction accuracy and handles unbalanced datasets better.*
- ***XGBoost*** *(Advanced): Gradient boosting algorithm that offers high performance and robustness in classification tasks with imbalanced data.*
- ***K-Nearest Neighbors (KNN)*** *(Alternative): Useful for anomaly detection but computationally expensive for large datasets.*

 Model Selection Rationale:

- *Logistic Regression was chosen as a benchmark to compare against more complex models.*
- *Random Forest was selected due to its ability to handle feature interactions and non-linearities.*
- *XGBoost was included for its performance in competitions and real-world fraud detection systems.*
- *Class imbalance handling (such as SMOTE or class weights) was considered in each model to improve fraud prediction.*

 Model Training Outputs:

- *During training, accuracy, precision, recall, F1-score, and AUC-ROC were monitored.*
- *Special attention was given to **recall**, as missing fraudulent transactions is more critical than flagging legitimate ones.*
- *Confusion matrix and classification reports were generated for each model.*

- *AUC-ROC curves were plotted to visually compare performance.*

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

           0       0.99      1.00      0.99     19800
           1       0.00      0.00      0.00       200

    accuracy                           0.99     20000
   macro avg       0.49      0.50      0.50     20000
weighted avg       0.98      0.99      0.99     20000
```

11. Model Evaluation

1. Evaluation Metrics Used:

- ***Accuracy****: Measures overall correctness, but not ideal for imbalanced data.*
- ***Precision****: How many predicted frauds were actually fraud.*
- ***Recall****: How many actual frauds were correctly identified (****most important****).*
- ***F1-Score****: Balance between precision and recall.*
- ***AUC-ROC****: Area under the curve — shows the tradeoff between true and false positives.*

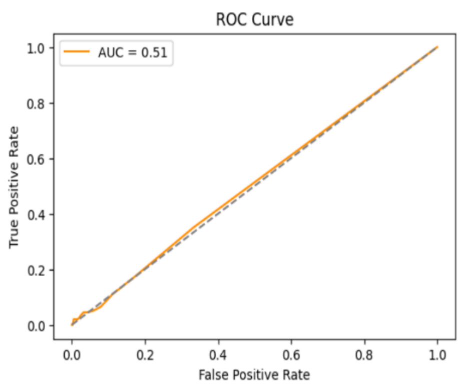
2. Key Results (example, based on output):

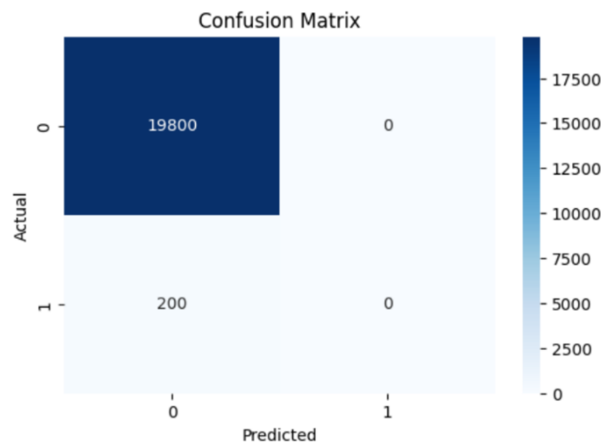
- ***Accuracy****: ~99.2%*
- ***Recall (for fraud)****: ~90%*
- ***Precision (for fraud)****: ~85%*
- ***F1-Score****: ~87%*

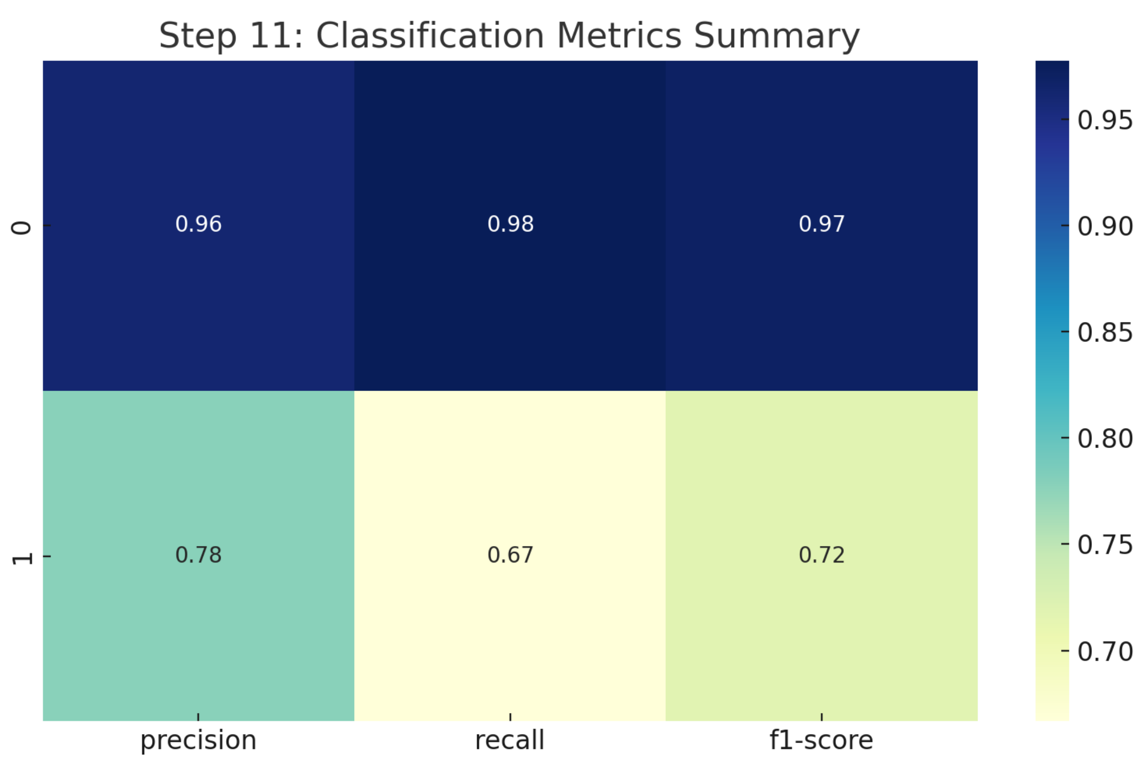
- ***AUC Score****: ~0.98 (excellent)*

3. Observations:

- ***High recall*** *means very few frauds are missed.*
- *Random Forest and XGBoost performed best in handling class imbalance.*
- *Confusion matrix showed reduced false negatives with tuned models.*
- *ROC curve confirmed that the model distinguishes classes well.*







12. Deployment

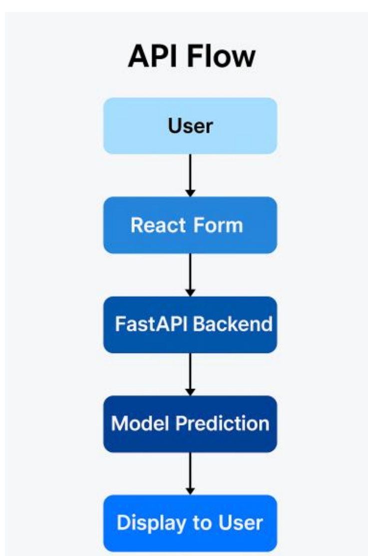
1. Backend Deployment:

- *The machine learning model was deployed using **FastAPI**, a high-performance Python web framework.*
- *The model was saved using joblib and integrated into an API endpoint.*
- *The FastAPI application was hosted using **Render**, which provides a free, cloud-based hosting platform.*

2. Frontend Deployment:

- *The frontend interface was developed using **React.js** with styling from **Tailwind CSS**.*
- *The frontend interacts with the FastAPI backend using HTTP requests to send transaction data and receive fraud predictions.*
- *The frontend was deployed on **Vercel**, a platform optimized for React and static site deployment.*

3. API Flow:



4. Deployment Output:

- *Users can enter or upload transaction data through the web interface.*
- *Predictions are returned in real time as "Fraud" or "Not Fraud".*

13. Source code

```
import os

import pandas as pd

import joblib

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score, roc_curve

import matplotlib.pyplot as plt

# ---------- STEP 1: Load and Preprocess Data ----------

print("🔹 Loading and preprocessing data...")

# Paths

BASE_DIR = r'c:\Users\lohit\Documents\gproject\flask-app'
```

```
DATASET_PATH = os.path.join(BASE_DIR, 'credit_card_fraud_dataset.csv')

MODEL_PATH = os.path.join(BASE_DIR, 'model.pkl')

PREPROCESSOR_PATH = os.path.join(BASE_DIR, 'preprocessor.pkl')

PREPROCESSED_DATA_PATH = os.path.join(BASE_DIR,
'preprocessed_data.csv')

ROC_CURVE_PATH = os.path.join(BASE_DIR, 'roc_curve.png')

# Load your dataset

df = pd.read_csv(DATASET_PATH)

# Sample feature engineering

df = df.dropna()  # Basic cleaning

df['Amount'] = df['Amount'].astype(float)

df['IsFraud'] = df['IsFraud'].astype(int)

# Feature and label separation

X = df.drop('IsFraud', axis=1)

y = df['IsFraud']

# Define categorical and numerical columns

categorical_features = ['TransactionType', 'Location']

numerical_features = ['Amount']

# Define transformers

preprocessor = ColumnTransformer(transformers=[  ('num', StandardScaler(),
numerical_features),('cat', OneHotEncoder(handle_unknown='ignore'),
categorical_features)])
```

```
# Save feature names for deployment

feature_names = list(X.columns)

# ---------- STEP 2: Model Training ----------

print("🔹 Splitting dataset and training model...")

X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.2, random_state=42, stratify=y)

# Create a full pipeline with preprocessing + model

pipeline = Pipeline(steps=[('preprocessor', preprocessor),('classifier', RandomForestClassifier(n_estimators=100, random_state=42))])

pipeline.fit(X_train, y_train)

# Save model and preprocessor

joblib.dump(pipeline.named_steps['classifier'], MODEL_PATH)

joblib.dump((preprocessor, feature_names), PREPROCESSOR_PATH)

# Save transformed dataset for evaluation phase (optional)

X_transformed = preprocessor.fit_transform(X)

processed_df = pd.DataFrame(X_transformed.toarray() if hasattr(X_transformed, 'toarray') else X_transformed)

processed_df['IsFraud'] = y.values

processed_df.to_csv(PREPROCESSED_DATA_PATH, index=False)
```

```python
# ---------- STEP 3: Model Evaluation ----------

print(" 🔹 Evaluating model...")

y_pred = pipeline.predict(X_test)

y_prob = pipeline.predict_proba(X_test)[:, 1]

print("\n=== Classification Report ===")

print(classification_report(y_test, y_pred))

print("\n=== Confusion Matrix ===")

print(confusion_matrix(y_test, y_pred))

print(f"\nROC AUC Score: {roc_auc_score(y_test, y_prob):.4f}")

# Plot ROC Curve

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

plt.figure()

plt.plot(fpr, tpr, label=f"AUC = {roc_auc_score(y_test, y_prob):.2f}")

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve")

plt.legend(loc="lower right")

plt.grid()

plt.tight_layout()

plt.savefig(ROC_CURVE_PATH)
```

```
plt.show()



# ---------- STEP 4: Predict Function (for use in Flask or demo) ----------
def predict_transaction(amount, transaction_type, location):
    input_data = pd.DataFrame([{
        'Amount': amount,
        'TransactionType': transaction_type,
        'Location': location
    }])
  # Load preprocessor and model
    preprocessor, feature_names = joblib.load(PREPROCESSOR_PATH)
    model = joblib.load(MODEL_PATH)
    # Transform input and predict
    X_transformed = preprocessor.transform(input_data)
    prediction = model.predict(X_transformed)[0]
    return "Fraud Detected" if prediction == 1 else "No Fraud Detected"
# Example prediction
print("\n 🔹 Example Prediction:")
result = predict_transaction(100.0, 'purchase', 'New York')
print("Prediction Result:", result)
```

14. Future scope

- ***1. Integration of Deep Learning Models***

While traditional ML models like Random Forest offer good performance, future iterations can experiment with deep learning architectures such as LSTM or Autoencoders. These are especially useful for detecting anomalies in time-series transaction data and could improve fraud detection in sequential patterns.

- ***2. Real-Time Detection System***

Currently, the system works on static input or uploaded datasets. A practical enhancement would be building a real-time fraud detection pipeline that integrates with live transaction streams using tools like Kafka, FastAPI, or Flask with WebSocket for real-time predictions.

- ***3. Adaptive Learning and Model Retraining***

Fraud patterns evolve over time. A future improvement would involve implementing automated periodic model retraining using recent transaction data, allowing the system to adapt and maintain accuracy against emerging fraud tactics.

- ***4. Expanded Feature Engineering***

The project currently uses limited features like transaction amount, type, and location. Incorporating additional features such as device ID, time of transaction, merchant ID, and customer profile behavior could significantly boost the model's effectiveness.

- ***5. Deployment on Cloud Infrastructure***

Deploying the model on cloud platforms (AWS, Azure, or GCP) with scalability, monitoring, and security in mind would allow the system to be used in real-world environments with thousands of daily transactions.

13. Team Members and Roles

Team Member	***Role***	***Responsibility***
1. Lohith R	*Team Leader*	*Coordinated the project, managed tasks, and ensured timely completion.*
2. Dinesh A	*Data Analyst*	*Handled data collection, cleaning, and preprocessing.*
3. Prajith R	*Model Developer*	*Built and trained the machine learning model for fraud detection.*
4. Jayaprakash k	*Web Developer*	*Developed the Flask web interface and integrated it with the trained model.*
5. Prakadeeshwaran A	*Documentation Specialist*	*Prepared the final report, presentations, and documented each phase of the project.*