import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_curve, auc

import warnings

# Suppress specific warnings

warnings.filterwarnings('ignore', category=UserWarning) # Suppress UserWarnings (from StandardScaler)

warnings.filterwarnings('ignore', message=".\*precision is ill-defined.\*") # Suppress UndefinedMetricWarning related to precision

# Step 1: Load Data from the CSV file

data = pd.read\_csv('D:/Mini Project/transactions.csv')

# Convert the 'Fraud' column to a binary numeric column ('Yes' = 1, 'No' = 0)

data['isFraud'] = data['Fraud'].map({'Yes': 1, 'No': 0})

# Drop the 'TransactionID' and 'Fraud' columns as they are not needed for the model

data = data.drop(['TransactionID', 'Fraud'], axis=1)

# Step 2: Data Preprocessing

# Features and labels

X = data.drop('isFraud', axis=1)

y = data['isFraud']

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize features (for better performance of ML models)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Step 3: Model Training (Logistic Regression)

model = LogisticRegression()

model.fit(X\_train\_scaled, y\_train)

# Step 4: Model Evaluation

# Predictions

y\_pred = model.predict(X\_test\_scaled)

# Accuracy Score

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

# Confusion Matrix and Classification Report

print("Confusion Matrix:")

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

print("\nClassification Report:")

# Handle UndefinedMetricWarning by setting zero\_division=0 to avoid warnings in metrics like precision

print(classification\_report(y\_test, y\_pred, zero\_division=0))

# Step 5: Plotting Graphs

# Turn on interactive mode (non-blocking)

plt.ion()

# 1. Confusion Matrix Heatmap

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['Normal', 'Fraud'], yticklabels=['Normal', 'Fraud'])

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('Actual')

# 2. ROC Curve

fpr, tpr, thresholds = roc\_curve(y\_test, model.predict\_proba(X\_test\_scaled)[:, 1])

roc\_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC Curve (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--') # Diagonal line

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend(loc='lower right')

# Display the plots without blocking the execution

plt.show()

# Step 6: Predicting New Transaction Fraud

# Example: New transaction details

new\_transaction = np.array([[1500, 5000, 3500]]) # Amount, OldbalanceOrg, NewbalanceOrig

new\_transaction\_scaled = scaler.transform(new\_transaction)

# Predict fraud

fraud\_prediction = model.predict(new\_transaction\_scaled)

if fraud\_prediction == 1:

print("Fraudulent transaction detected!")

else:

print("Transaction seems normal.")