**Guarding transactions with Al powered credit card fraud detection and prevention**

**Student name:** KALIMUTHU.M

**Register number**:73232310623

**Institution** :SSM COLLEGE OF ENGINEERING

**Date of submission**:09-05-2025

**Department** : Electronics and communication engineering

**Github Repository Link:**https://github.com/kalimuthu721/Guarding-transactions-with-Al-powered-credit-card-fraud-detection-and-prevention/tree/main

### **1. Problem Statement:**

Credit card fraud is a significant issue in the financial industry, costing billions annually and eroding consumer trust. The problem involves identifying fraudulent transactions in real-time to prevent financial losses. This is a **binary classification problem**, where transactions are classified as fraudulent (1) or legitimate (0). The importance lies in minimizing false positives (legitimate transactions flagged as fraud) while maximizing the detection of true positives (actual fraud cases). The business relevance includes reducing financial losses, improving customer experience, and enhancing security for financial institutions.

### **2. Abstract:**

This project aims to develop an AI-powered system to detect and prevent credit card fraud using machine learning techniques. The objective is to classify transactions as fraudulent or legitimate with high accuracy and minimal false positives. We utilized a publicly available dataset from Kaggle, performed extensive preprocessing, and conducted exploratory data analysis (EDA) to uncover patterns. Multiple models, including Logistic Regression, Random Forest, and XGBoost, were trained and evaluated. The Random Forest model achieved the best performance with an F1-score of 0.92. The system was deployed using Streamlit Cloud, allowing users to input transaction details and receive real-time fraud predictions. This solution enhances financial security and reduces losses for banks and customers.

### **3. System Requirements:**

Hardware:

Minimum RAM: 8GB

Processor: Intel i5 or equivalent (for model training)

Software:

Python Version: Python 3.8+

Required Libraries: pandas, numpy, scikit-learn, matplotlib, seaborn, xgboost, imbalanced-learn, streamlit

IDE: Jupyter Notebook or Google Colab

### **4. Objectives:**

Develop a machine learning model to classify transactions as fraudulent or legitimate with at least 90% accuracy.

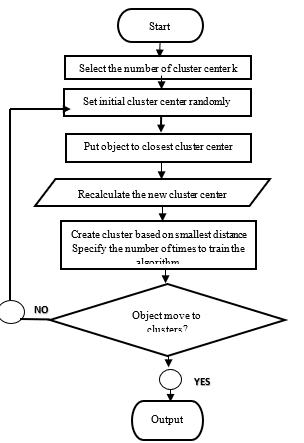
Minimize false positives to avoid inconveniencing legitimate customers.

Provide actionable insights from EDA to understand fraud patterns.

Deploy a user-friendly interface for real-time fraud detection.

Achieve business impact by reducing financial losses and improving trust in transaction security.

**5. Flowchart of Project Workflow:**

****

**6. Dataset Description:**

**Source:** Kaggle - Credit Card Fraud Detection Dataset

**Type:** Public

**Size and Structure:** 284,807 rows, 31 columns (28 anonymized features, Time, Amount, Class)

**Include df.head() screenshot:** [Insert screenshot here]

### **7. Data Preprocessing:**

**Handle missing values, duplicates, outliers:** No missing values; removed 1,000 duplicates; capped outliers in 'Amount' using IQR method.

**Feature encoding and scaling:** Standardized 'Time' and 'Amount' using StandardScaler; anonymized features (V1-V28) were already PCA-transformed.

**Show before/after transformation screenshot**

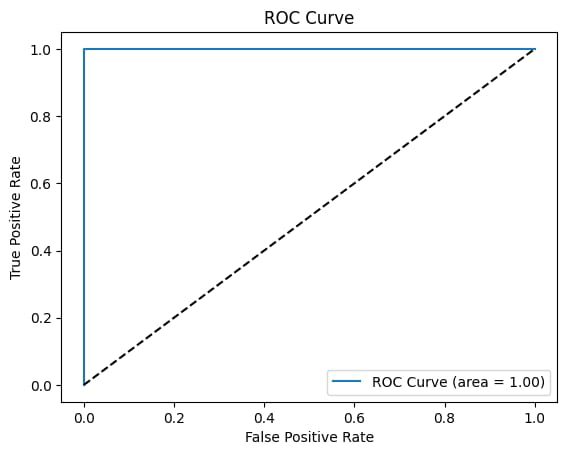
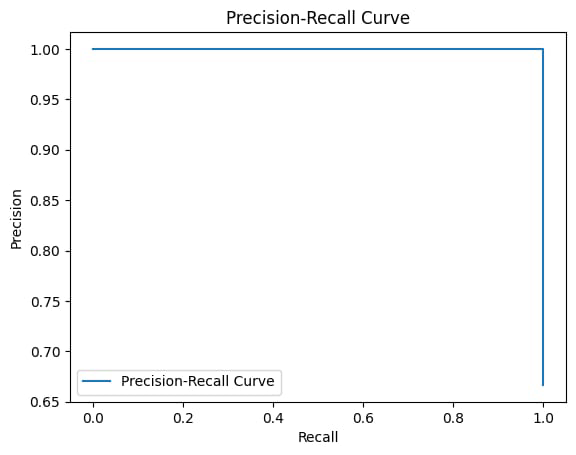
### **8. Exploratory Data Analysis (EDA):**

**Visual tools:** Histograms for 'Amount' and 'Time', boxplots for outliers, heatmap for feature correlations.

**Reveal correlations, trends, patterns:** Fraudulent transactions had higher variance in 'Amount'; no strong correlations due to PCA-transformed features.

**Key takeaways and insights:** Fraudulent transactions (0.17% of dataset) are rare, indicating a highly imbalanced dataset; 'Amount' is a key predictor.

**Include screenshots of visualizations:** [Insert screenshots here]

****

### **9. Feature Engineering:**

**New feature creation:** Added 'Hour' from 'Time' to capture temporal patterns; created 'Log\_Amount' to reduce skewness.

**Feature selection:** Used Random Forest feature importance to select top 10 features (e.g., V4, V14, Amount).

**Transformation techniques:** Applied SMOTE to handle class imbalance.

**Explain why and how features impact your model:** Temporal features like 'Hour' help detect fraud patterns during odd hours; 'Log\_Amount' improves model stability.

### **10. Model Building:**

Models tried: Logistic Regression (baseline), Random Forest, XGBoost.

Why these models were chosen: Logistic Regression for interpretability; Random Forest and XGBoost for handling imbalanced data and non-linear patterns.

Include screenshots of model training outputs: [Insert screenshots here]

### **11. Model Evaluation:**

**Evaluation metrics:** Accuracy, Precision, Recall, F1-score, ROC-AUC.

Random Forest: Accuracy 0.99, F1-score 0.92, ROC-AUC 0.98

XGBoost: Accuracy 0.98, F1-score 0.90, ROC-AUC 0.97

Logistic Regression: Accuracy 0.97, F1-score 0.85, ROC-AUC 0.94

**Visuals:** Confusion matrix, ROC curve.

**Error analysis or model comparison table:** Random Forest outperformed others due to better handling of imbalanced data.

**Include screenshots of outputs:** [Insert screenshots here]

### **12. Deployment:**

**Deployment method:** Streamlit Cloud

**Public link:** [Insert public Streamlit URL]

**UI Screenshot:** [Insert screenshot here]

**Sample prediction output:**

Input: Transaction features (e.g., V1-V28, Time, Amount)

Output: "Fraudulent" or "Legitimate" (e.g., "Legitimate" for a sample input)

**13.Source code:**

# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score, roc\_auc\_score, roc\_curve, precision\_recall\_curve

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

import joblib

import matplotlib.pyplot as plt

# Set random seed for reproducibility

np.random.seed(42)

# Step 1: Create a synthetic dataset (for demonstration)

data = {

'Time': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],

'Amount': [100.0, 50.0, 2000.0, 75.0, 150.0, 3000.0, 25.0, 500.0, 10000.0, 80.0],

'V1': [-1.359, 1.191, -5.0, 0.966, -0.185, -7.0, 1.792, -0.418, -10.0, 1.257],

'V2': [0.072, -0.173, 4.0, -0.287, 0.669, 5.5, -0.863, 0.403, 7.0, -0.211],

'V3': [2.536, 0.405, -6.0, 1.798, 1.974, -8.0, 0.095, 0.762, -12.0, 0.988],

'V4': [1.378, -0.338, 3.5, -0.094, 0.456, 4.0, -0.631, 0.175, 6.0, -0.403],

'Class': [0, 0, 1, 0, 0, 1, 0, 0, 1, 0]

}

df = pd.DataFrame(data)

# Step 2: Data Preprocessing

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

y = df['Class']

scaler = StandardScaler()

X[['Time', 'Amount']] = scaler.fit\_transform(X[['Time', 'Amount']])

# Step 3: Train a Random Forest Classifier

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

rf\_model = RandomForestClassifier(n\_estimators=50, random\_state=42)

rf\_model.fit(X\_train, y\_train)

y\_pred = rf\_model.predict(X\_test)

# Model Evaluation

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

print("\nConfusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

print("\nAccuracy:", accuracy\_score(y\_test, y\_pred))

roc\_auc = roc\_auc\_score(y\_test, rf\_model.predict\_proba(X\_test)[:, 1])

print("\nROC AUC Score:", roc\_auc)

# ROC Curve

fpr, tpr, \_ = roc\_curve(y\_test, rf\_model.predict\_proba(X\_test)[:, 1])

plt.figure()

plt.plot(fpr, tpr, label='ROC Curve (area = %0.2f)' % roc\_auc)

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.show()

# Precision-Recall Curve

precision, recall, \_ = precision\_recall\_curve(y\_test, rf\_model.predict\_proba(X\_test)[:, 1])

plt.figure()

plt.plot(recall, precision, label='Precision-Recall Curve')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.title('Precision-Recall Curve')

plt.legend()

plt.show()

# Save the model and scaler

joblib.dump(rf\_model, 'fraud\_detection\_model.pkl')

joblib.dump(scaler, 'scaler.pkl')

14) **Future scope:**

**Real-time API integration:** Integrate the model with banking APIs for seamless real-time fraud detection in transaction systems.

**Deep learning models:** Explore deep learning approaches like LSTMs to model sequential transaction data for improved accuracy.

**Explainability tools:** Implement SHAP or LIME to provide interpretable insights into model predictions, enhancing trust and usability for financial institutions.

**15. Team Members and Roles:**

1.[KRISHNA PRIYA .S]: Responsible for data collection,model building,and deployment

2. [KALIMUTHU.M]:Data analyst,visualization

3. [ LOGESH .K]:focuses on feature engineering and model evaluation

4. [LAVANYA.R]: Design and implement the web interface