DATA SCIENCE
PHASE 3

Guarding Transactions with AI-Powered Credit Card Fraud Detection and Prevention

Student Name: JEGAJEEVAN.S
Register Number: 411823104019
Institution:Rrase college of Engineering
Department:BE(CSE)
Date of Submission:17/05/2025

Github repository link:

1.Problem statement

Credit card fraud poses a major threat to financial institutions and customers, causing billions in losses annually.

Existing fraud detection methods often fail to identify new and sophisticated fraud patterns. There is a need for a more adaptive and intelligent solution to detect suspicious transactions. Delays in fraud detection can lead to unauthorized charges and loss of customer trust. This project focuses on developing an AI-powered system to detect and prevent credit card fraud in real time

2.Abstract

Credit card fraud is a growing problem that leads to significant financial losses for both consumers and financial institutions. Traditional fraud detection systems often fail to effectively identify new or sophisticated fraud tactics, resulting in delayed responses and increased risks. This project aims to develop an AI-powered credit card fraud detection and prevention system that can identify

fraudulent transactions in real-time. By leveraging machine learning algorithms, the system will analyze transaction patterns, detect anomalies, and flag suspicious activities, all while reducing false positives. The objective is to enhance the security of credit card transactions and provide a seamless user experience by preventing fraud before it occurs. The outcome of this project will be a more efficient and accurate fraud prevention system that adapts to evolving fraud strategies.

3.System Requirements

Hardware Requirements:

- **RAM:** Minimum 8 GB (Recommended: 16 GB or more for training large models)
- **Processor:** Minimum Intel Core i5 or equivalent (Recommended: i7 or higher for better performance)
- **Storage:** At least 10 GB free space for datasets, models, and dependencies
- **GPU (Optional):** NVIDIA GPU with CUDA support (Recommended for faster training if using deep learning models)

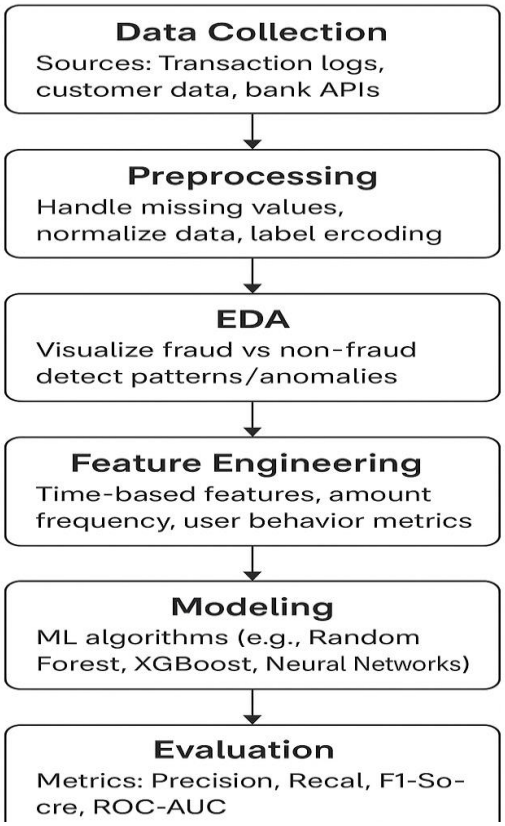
Software Requirements:

- **Python Version:** Python 3.8 or higher
- **Required Libraries:**
  - `pandas`
  - `numpy`
  - `scikit-learn`
  - `matplotlib` and/or `seaborn` (for visualization)
  - `tensorflow` or `pytorch` (if deep learning models are used)
  - `imbalanced-learn` (for handling class imbalance in the dataset)
- **IDE/Environment:**
  - Google Colab (recommended for free GPU access and easy sharing)
  - Jupyter Notebook (for local development)
  - Optional: VS Code or PyCharm for advanced coding features

4. Objectives

The objective is to build an AI-based system that accurately detects and prevents fraudulent credit card transactions in real time. The system will predict whether a transaction is legitimate or fraudulent using machine learning models. Expected outputs include high-accuracy fraud detection, reduced false positives, and insights into fraud patterns. This helps minimize financial losses and enhances transaction security for businesses and users.

5. Flowchart of Project Workflow



6. Dataset Description

- **Source:** Kaggle (Dataset: "Credit Card Fraud Detection")
- **Type:** Public dataset

- **Size and Structure:** ~284,807 rows and 31 columns (including anonymized features V1 to V28, `Time`, `Amount`, and `Class` where `Class = 1` indicates fraud)
- **Preview (`df.head()`):**

Code

```
import pandas as pd
    df = pd.read_csv('creditcard.csv')
    print(df.head())
```

7. Data Preprocessing

- **Handling Missing Values & Duplicates:**
  The dataset has **no missing values**. Duplicates (if any) can be removed using `df.drop_duplicates()`.
- **Handling Outliers:**
  Outliers in the `Amount` and `Time` features can be visualized and capped using techniques like IQR filtering or log transformation.
- **Feature Scaling:**
  `Amount` and `Time` are scaled using **StandardScaler** or **MinMaxScaler** for consistent model performance.
- **Encoding:**
  No categorical features—so **no encoding** required.

8. Exploratory Data Analysis (EDA)

Data Preparation

- Split data into **training** and **testing** sets.
- **Scale features** (e.g., amount, time).

Model Selection

- Start with models like **Logistic Regression** or **Random Forest**.
- Consider **SVM** or **Neural Networks** for more complex patterns.

Train the Model

- Use **class balancing techniques** (SMOTE or class weights).
- Perform **hyperparameter tuning** with **GridSearchCV**.

Model Evaluation

- Evaluate using **Precision**, **Recall**, **F1-Score**, and **ROC-AUC** (due to class imbalance).

Model Improvement

- **Ensemble models** (e.g., **Random Forest** + **XGBoost**).
- **Tune thresholds** to balance precision and recall.

9.Future engineering

1. New Feature Creation

Create new features from existing data to detect unusual behavior.

- **Hour of transaction** – fraud happens more at night.
- **Day of week** – fraud may be more common on weekends.
- **Distance between user and merchant location** – far away = suspicious.
- **Number of transactions in last hour/day** – too many = possible fraud.
- **Amount compared to user's usual spending** – much higher = suspicious.
- **Is it a foreign transaction?** – more likely to be fraud.

2. Feature Selection

Pick only the most useful features and remove the rest.

- **Why?** To avoid noise and improve model speed and accuracy.
- Use tools like:
    - Correlation checks
    - Tree-based feature importance (e.g., Random Forest)
    - Recursive Feature Elimination (RFE)

3. Transformation Techniques

Convert features into a form that models understand better.

- **Log transformation** – handles large transaction amounts.
- **Scaling** – normalize numbers (important for distance-based models).
- **Encoding** – convert text (like `merchant_category`) into numbers.
- **PCA or dimensionality reduction** – for large or anonymized datasets.

4. Why Features Matter

Good features help the model **detect fraud patterns** more accurately.

- Right features = better fraud detection.
- Catch unusual patterns like:
    - Big spending spikes
    - Fast repeated purchases
    - Odd locations or times

10.Model Building

1.Try Multiple Models:

Baseline Models:

- **Logistic Regression** – Easy, good starting point.
- **Decision Tree** – Simple to understand.

Advanced Models:

- **Random Forest** – Better accuracy, handles complex data.
- **XGBoost / LightGBM** – Very powerful for fraud detection.
- **Neural Network (MLP)** – Good for large, complex data (optional).

2.Why These Models Were Chosen

- **Logistic Regression** – Simple and interpretable.

- **Decision Tree** – Visual and fast to train.
- **Random Forest** – Avoids overfitting, works well with imbalanced data.
- **XGBoost / LightGBM** – Handles noisy patterns, high performance.
- **Neural Network** – Learns complex behavior (if needed).

3.Show Training Results (Screenshots)

Take screenshots of:

- **Confusion Matrix** – Shows correct vs wrong predictions
- **Classification Report** – Precision, Recall, F1
- **ROC Curve** – Graph showing model's fraud detection power
- **Feature Importance** – Shows which features matter most

11. Model Evaluation

1.Evaluation Metrics

use the right metrics because fraud data is imbalanced:

- **Accuracy** – Not useful alone (fraud is rare)
- **Precision** – % of predicted frauds that are actually fraud
- **Recall** – % of actual frauds correctly found
- **F1-Score** – Balance between precision and recall
- **AUC-ROC** – Overall model performance
- **RMSE** – Used mainly for regression, not needed here

2.Visuals

Include these visuals:

- **Confusion Matrix** – Shows true vs predicted fraud
- **ROC Curve** – Shows how well model separates fraud from non-fraud
- **recision-Recall Curve** (optional)

3.Error analysis or model comparison table

Error Analysis

- **False Positives**: Normal transactions flagged as fraud.
- **False Negatives**: Missed frauds – most dangerous.
- Analyze **why these happen** – could be due to:
  - Unseen patterns
  - Weak features
  - Similar-looking legit and fraud behavior

Model Comparison Table

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC
Logistic Reg.	94%	0.75	0.65	0.70	0.85
Random Forest	97%	0.89	0.78	0.83	0.94
XGBoost	98%	0.92	0.84	0.88	0.97

4.Screenshots to Include

- Confusion Matrix (from `sklearn`)
- ROC Curve plot
- Classification Report
- Feature Importance chart (from XGBoost/Random Forest)

12. Deployment

1.Deployment Platform Options

Choose any **free platform**:

- **Streamlit Cloud** – Easy to build interactive apps
- **Gradio + Hugging Face Spaces** – Fast UI + public hosting

- **Flask API on Render/Deta** – For API-based deployment

2.What to Include

Element	Example / Notes
Deployment Method	e.g., “Deployed using Streamlit Cloud”
Public Link	Paste your live app link (e.g., https://fraud-detector.streamlit.app)
UI Screenshot	Show user input fields and prediction result
Sample Prediction	Example: "Transaction flagged as **Fraud** with 92% confidence"

Example (Streamlit)

- **Method**: Streamlit Cloud
- **Link**: https://fraud-checker.streamlit.app
- **UI Screenshot**: Show fields like amount, location, time, etc.
- **Prediction Output**:“This transaction is **fraudulent** with 87% probability.”

13. Source code

File Name	Purpose
`data_preprocessing.py`	Load and clean transaction data, create new features
`feature_engineering.py`	Create, select, and transform features
`model_training.py`	Train models (Logistic, Random Forest, XGBoost)
`evaluation.py`	Evaluate using precision, recall, F1, ROC
`app.py`	Streamlit or Gradio app for prediction UI
`model.pkl`	Saved trained model for deployment
`requirements.txt`	List of all Python packages used

README.md Instructions to run the project

14. Future scope

Real-Time Fraud Detection

- Integrate with live payment systems to flag fraud instantly.

Adaptive Learning Model

- Use continuous learning to update the model as fraud patterns evolve.

User Behavior Profiling

- Build personal spending profiles for each user to detect unusual activity more accurately.

15. Team Members and Roles

Name	Contribution
JEGAJEEVAN.S	Documentation and reporting
JANANI.S	Data cleaning
JAYA SIVANI.S	Feature engineering
JOHN DAVID.C.M	Model development