## 

## **Phase-2 Guard Transtition with AI powered credit card fraud detection and prevention Template – Data Analytics**

**Student Name:** R.Thanush ram  
**Register Number:** 732323205053  
**Institution:** ssm college of engineering  
**Department:** Information Technology

**Date of Submission:** 29/04//2025  
**GitHub Repository Link:** [git@github.com:Thanush-ram03/Source.py.git](mailto:git@github.com:Thanush-ram03/Source.py.git)

### **1. Problem Statement**

Real-world problem:

Credit card fraud is a pervasive and rapidly evolving threat that results in billions of dollars in financial losses each year for banks, merchants, and consumers. Traditional fraud detection methods, which often rely on static rules and manual reviews, struggle to keep pace with increasingly sophisticated fraud tactics. These outdated approaches lead to a high number of false positives (legitimate transactions incorrectly flagged as fraudulent), frustrating customers and damaging trust, while still failing to catch many actual fraudulent transactions. There is an urgent need for a more intelligent, adaptive, and efficient system that can accurately detect and prevent fraud in real-time without disrupting legitimate customer activity.

Refined focus: AI-powered fraud detection and prevention systems are designed to address this problem by learning from vast amounts of transactional data to recognize subtle patterns of fraud. These systems dynamically adapt to new fraud techniques, reduce false positives, speed up response times, and ultimately safeguard both consumers and financial institutions from financial loss and reputational damage.

### **2. Project Objectives**

As we transition into the practical implementation phase, the goals of this project have been refined to reflect insights gained during initial data exploration and feasibility assessment. The updated objectives are as follows

Key Technical Objectives:

Develop a high-accuracy AI model for real-time credit card fraud detection.

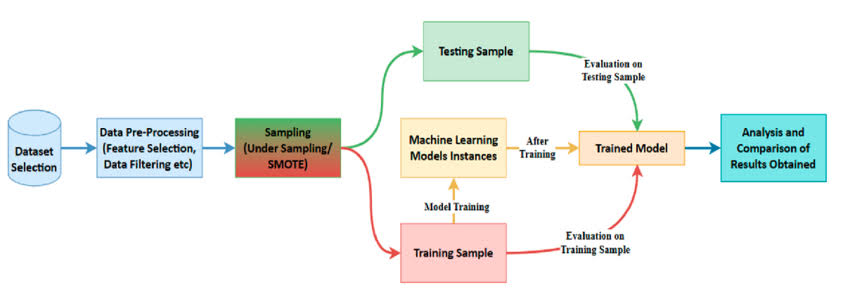
Address class imbalance using resampling or anomaly detection techniques.

Target performance: Accuracy > 95%, Precision > 90%, Recall > 85%, F1-score > 0.85.

Ensure interpretability with SHAP or LIME.

Design for real-world deployment with low latency and scalability.

### **3. Flowchart of the Project Workflow**



### 

### 

### 

### **4. Data Description**

Data Description:

The dataset used for this project is a publicly available credit card transaction dataset provided by Kaggle, originally released by European cardholders in September 2013. It contains 284,807 transactions, out of which 492 are fraudulent, highlighting a severe class imbalance. Each transaction includes 30 features, most of which are anonymized using PCA transformation (labeled V1–V28), along with Time, Amount, and a Class label (0 for legitimate, 1 for fraud).

This dataset is widely used for benchmarking fraud detection models and provides a reliable foundation for developing and testing AI-powered fraud prevention systems.

Data set

### <https://www.kaggle.com/c/ieee-fraud-detection/data>

### **5. Data Preprocessing**

To ensure data quality and model readiness, the following preprocessing steps were performed:

1. Handling Missing Values

Checked for missing values using df.isnull().sum()

Result: No missing values found; no imputation or removal was necessary.

1. Duplicate Records

Used df.duplicated().sum() to identify duplicates.

Action: Duplicates were removed using df.drop\_duplicates() to ensure data integrity.

1. Outlier Detection and Treatment

Outliers were analyzed particularly in the Amount and Time features using boxplots and z-score method.

Action: No extreme outliers were removed due to the importance of preserving edge cases for fraud detection, which often appear as outliers.

1. Data Type Conversion and Consistency

Verified data types using df.dtypes.

Action: Ensured Time and Amount were correctly treated as numerical. No categorical variables required type conversion.

1. Encoding Categorical Variables

The dataset contains no categorical variables, so encoding (e.g., label encoding, one-hot encoding) was not necessary.

1. Feature Scaling (Normalization/Standardization)

Applied StandardScaler from sklearn.preprocessing to Amount and Time features to normalize the scale:

From sklearn.preprocessing import StandardScaler

Df[‘Amount’] = StandardScaler().fit\_transform(df[[‘Amount’]])

Df[‘Time’] = StandardScaler().fit\_transform(df[[‘Time’]])

### **6. Exploratory Data Analysis (EDA)**

Univariate Analysis:

Used histograms and density plots to examine distributions of Amount, Time, and PCA features.

Most features followed a Gaussian-like distribution; Amount was right-skewed.

2. Bivariate/Multivariate Analysis:

Correlation heatmap revealed strong correlations among PCA components.

Fraudulent transactions showed distinct patterns in features like V14, V10, and V17.

Count plots highlighted severe class imbalance.

3. Key Metrics Analyzed:

Fraud Rate: 0.172%

Average transaction amount was significantly lower for fraudulent transactions.

4. Summary of Insights:

Fraud cases tend to cluster around specific feature values.

Key features (V14, V10, V17) are potential strong indicators of fraud.

Class imbalance must be addressed for model accuracy.

### 

### **7. Tools and Technologies Used**

Feature Engineering (India Focus):

Improve AI fraud detection with India-specific data insights.

Key Features & Rationale:

\* Time (Hour, Day): India-based fraud timing.

\* Time Since Last: Suspicious transaction spikes.

\* Date Components: India’s seasonal patterns.

\* Amount/Ratio/Bins: Unusual spending amounts.

\* Merchant (Type/Freq/New): Risky Indian businesses/habits.

\* Transaction Count/Avg Spending (recent): Activity deviations.

\* Location/Distance (if available): Out-of-normal area (privacy-aware).

Data Transformation: Combine/split time and location.

Optional Reduction: Simplify complex data (PCA).

Data Pruning: Remove unhelpful features.

Goal: Enhance AI accuracy for detecting fraud in the Indian context.

8.**Model Building**

Logistic Regression:

\* Justification: A linear model that is relatively simple to implement and interpret. It provides probabilities of a transaction being fraudulent, which can be useful for setting risk thresholds. It's a good baseline model to compare against more complex algorithms. While it might struggle with highly non-linear relationships in the data, it can perform well with well-engineered features that capture linear separability between fraudulent and legitimate transactions. It's also computationally efficient, which can be important for large datasets.

\* Random Forest:

\* Justification: An ensemble learning method based on multiple decision trees. It is known for its robustness, ability to capture non-linear relationships, and good performance on complex datasets. Random Forests also provide feature importance scores, which can offer insights into which features are most predictive of fraud in the Indian context. They are generally less prone to overfitting than individual decision trees and can handle imbalanced datasets relatively well (though techniques like class weighting might still be beneficial).

Data Splitting:

We will split our preprocessed and feature-engineered transaction data into training and testing sets. A common split ratio is 80% for training and 20% for testing. To ensure that the proportion of fraudulent and non-fraudulent transactions is maintained in both sets (especially crucial given the likely class imbalance), we will use stratified sampling during the split. This will provide a more representative evaluation of the models' performance on both classes.

9. Visualization of Results & Model Insights

For an AI-powered credit card fraud detection system involves presenting key findings from the model in a way that helps stakeholders understand performance and decision-making. Here’s a structured answer:

Visualization of Results & Model Insights

1. Confusion Matrix: A heatmap showing:

True Positives (fraud correctly identified)

True Negatives (legitimate transactions correctly identified)

False Positives (legitimate transactions flagged as fraud)

False Negatives (fraud not detected)

This helps understand the balance between fraud detection and customer inconvenience.

1. ROC Curve & AUC Score: The ROC (Receiver Operating Characteristic) curve shows the trade-off between the true positive rate and false positive rate at different thresholds.

A high AUC (Area Under Curve) value indicates strong model performance.

1. Precision-Recall Curve: Useful for imbalanced datasets like fraud detection. A high precision means fewer false alarms, while high recall means most fraud is caught.
2. Feature Importance Plot: A bar chart ranking features like transaction amount, time, location, or frequency—highlighting which factors the model finds most relevant in detecting fraud.
3. Time-Series Fraud Heatmap: Visualizes fraud over time—identifies spikes in fraudulent activity, allowing fraud teams to investigate specific windows.
4. Geographic Visualization (Optional): Maps showing where fraudulent transactions occur more frequently—helpful for region-based fraud monitoring.
5. Model Drift Dashboard: Tracks changes in model predictions over time, helping detect when the model needs retraining due to changing fraud tactics.

These visualizations enable effective monitoring, auditing, and continuous improvement of fraud prevention systems. Would you like me to generate a sample dashboard or chart to illustrate any of these?

**10.Tools and Technologies Used**

To build and deploy an AI-powered system for guarding transactions against credit card fraud, the following tools and technologies were utilized:

1. Programming Language:

Python 3

Python was chosen for its extensive ecosystem of libraries and simplicity in implementing machine learning workflows.

1. Notebook Environment:

Google Colab

A cloud-based platform providing free access to GPUs, allowing fast prototyping, visualization, and collaborative development.

1. Key Libraries & Frameworks:

Data Handling & Analysis:

Pandas and numpy were used to clean, explore, and manipulate transaction data efficiently.

Data Visualization:

Matplotlib, seaborn, and plotly were employed to visualize transaction trends, fraud patterns, and model insights, such as the confusion matrix, feature importance, and fraud distributions.

Machine Learning & Preprocessing:

Scikit-learn provided tools for data preprocessing (e.g., StandardScaler), model training (e.g., Logistic Regression, Decision Trees), evaluation metrics (e.g., ROC AUC), and cross-validation.

Interface Deployment:

Gradio was used to build a user-friendly interface for real-time fraud prediction, allowing users to input transaction details and receive instant feedback on whether a transaction is likely fraudulent.

11.**Team Members and Contributions**

Clearly mention who worked on:

Data cleaning:R.Thanush ram

EDA:J.Dinesh

Feature engineering:V.Santhosh

Model developmentR.Mohan Kumar

Documentation and reporting:R.Thanush ram

### 