**Phase-3 Submission**

**Student Name: Subash Chandra bose M**

**Register Number:** 712523205060

**Institution:** PPG Institute of Technology

**Department:** Information Technology

**Date of Submission:09.05.2025**

**Github Repository Link:** [**REPO LINK**](https://github.com/subash2233/NM_Subash-chandra-bose_DS)

### **1. Problem Statement**

*In the modern digital economy, the exponential growth of online transactions has significantly increased the risk of credit card fraud, resulting in substantial financial losses and erosion of consumer trust. Traditional rule-based fraud detection systems often fail to identify novel or subtle fraudulent behaviors in real time, leading to either undetected fraud or a high rate of false positives that inconvenience legitimate customers. This project aims to design and develop an AI-powered credit card fraud detection and prevention system that leverages machine learning algorithms to analyze transactional patterns, detect anomalies, and prevent fraudulent activities in real time. The system should continuously learn from new data, adapt to emerging fraud tactics, and maintain a high accuracy rate with minimal false alarms, thereby ensuring secure, efficient, and user-friendly transaction processing.*

### **2. Abstract**

*This project focuses on developing an AI-powered credit card fraud detection and prevention system using machine learning techniques. With the growing volume of online transactions, detecting fraudulent activities in real-time has become critical.*

### **3. System Requirements**

*Specify minimum system/software requirements to run the project:*

* + ***Hardware****: Minimum RAM, processor ,storage,GPU.*
  + ***Software****: Python version, required libraries, IDE (Colab, Jupyter)*

### 

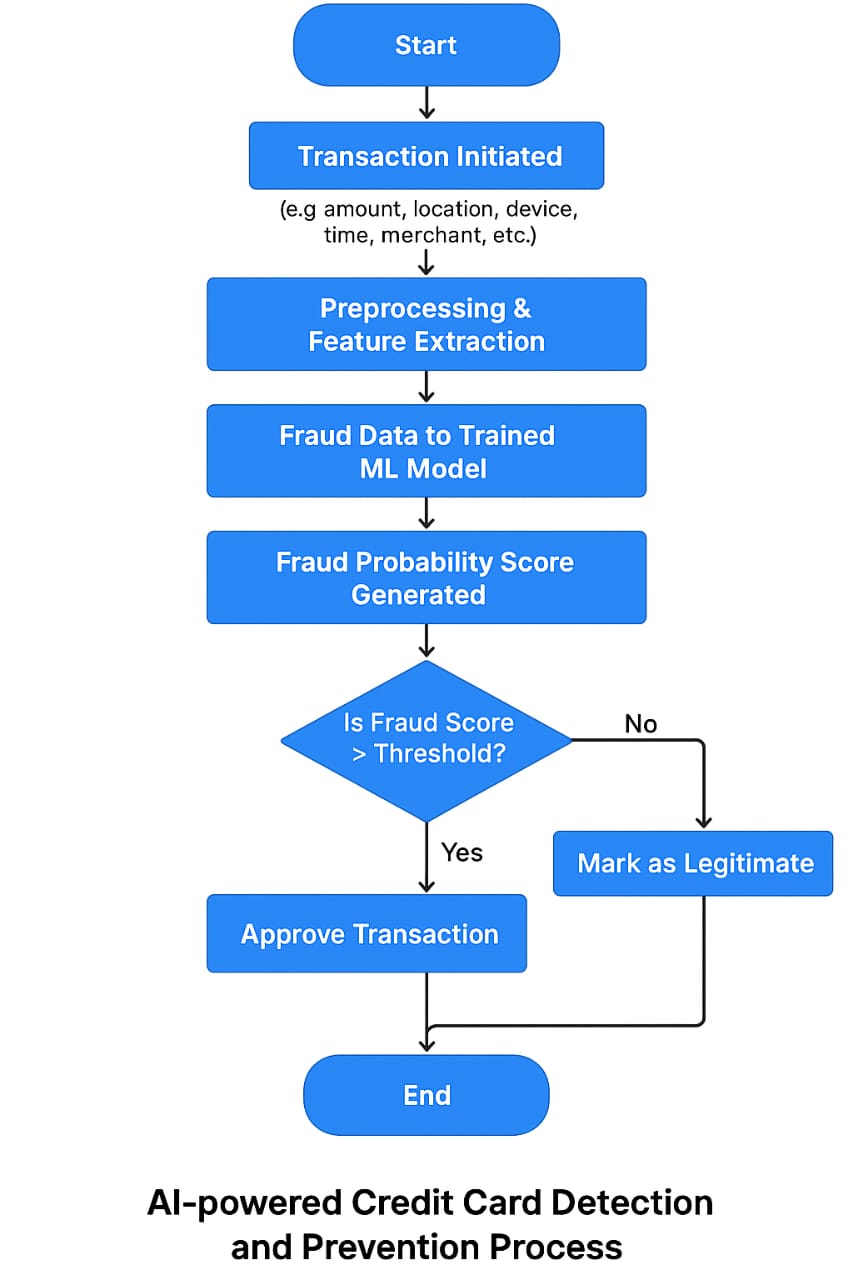
### **4. Objectives**

The objective of this project is to develop an AI-powered system for real-time credit card fraud detection and prevention that accurately identifies and blocks fraudulent transactions while minimizing false positives.

The system will use machine learning models trained on transactional data to detect unusual patterns, adapt to evolving fraud tactics, and provide actionable alerts or automated responses.

This will enhance transaction security, reduce financial losses, and improve customer trust in digital payment systems.

**5. Flowchart of Project Workflow**

**

### **6. Dataset Description**

* *Source - kaggle*

*https://www.kaggle.com/mlg-ulb/creditcardfraud*

* *Type - transactional data*
* *Size and structure*

*Rows: 284,807 transactions*

*Fraudulent Cases: 492 (approx. 0.17%)*

*Columns: 31 features*

### **7. Data Preprocessing**

1. *Data Preprocessing*
2. *Data cleaning*
3. *Feature engineering*
4. *Anomaly detection preprocessing*
5. *Label balancing*
6. *Train test split*

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

* *Univariate Analysis:*
  + *Histograms of transaction amount, time, etc.*
  + *Box plots to observe outliers (especially for fraudulent transactions)*
  + *KDE plots to compare feature distributions for fraud vs. legit*
* *Bivariate/Multivariate Analysis*

*Scatter plots or pair plots for selected features*

* + *Grouped bar plots (e.g., transaction mode vs. fraud rate)*
* *Insights Summary:  
  The dataset shows a strong class imbalance, with fraudulent transactions being rare but distinct in behavior, especially in terms of amount and timing. Key features like transaction amount, time, and location reveal useful patterns for accurate fraud detection.*

### **9. Feature Engineering**

*1. Time Features: Hour, day of the week, time since last transaction.*

*2. Behavioral Features: Average and standard deviation of transaction amount.*

*3. Location Features: Distance from last location, unusual locations.*

*4. Device Features: New vs. known device, device consistency.*

*5. Historical Aggregates: User's fraud count, transaction count in past 24h.*

*6. Merchant Features: Unusual merchant types for user.*

*7. Risk Scores: Assign fraud risk based on merchant or location.*

### **10. Model Building**

* + *1. Data Splitting: Split into training and test sets with stratified sampling.*
  + *2. Model Selection: Try models like Logistic Regression, Random Forest, XGBoost, or SVM.*
  + *3. Handle Imbalance: Use SMOTE, under sampling, or class weighting.*
  + *4. Model Training: Train and tune models with cross-validation.*
  + *5. Evaluation: Focus on Precision, Recall, F1-Score, and ROC-AUC.*
  + *6. Model Tuning: Optimize with GridSearchCV or RandomizedSearchCV.*
  + *7. Model Comparison: Select the best model based on performance metrics.*

### **11. Model Evaluation**

*1. Confusion Matrix: Measures true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN).*

*2. Precision: How many predicted frauds was actually fraud?*

*Precision = TP / (TP + FP)*

*3. Recall (Sensitivity): How many actual frauds were correctly detected?*

*Recall = TP / (TP + FN)*

*4. F1-Score: Harmonic mean of precision and recall. Best for imbalanced data.*

*5. ROC-AUC: Measures model's ability to distinguish between fraud and non-fraud.*

*6. PR-AUC: Precision-Recall curve, more informative with imbalanced data.*

### **12. Deployment**

* + *1. Model Export: Save trained model using joblib or pickle.*
  + *2. Backend Setup: Use Flask or FastAPI to build an API endpoint.*
  + *3. Integration: Connect the API to a transaction system or app for real-time predictions.*
  + *4. Cloud Hosting: Deploy on platforms like Hurok, AWS, or Azure.*
  + *5. Monitoring: Track model performance with tools like Prometheus or Grafana.*
  + *6. Updates: Regularly retrain with new data to maintain accuracy.*

**13. Source code**

*Import pandas as pd*

*import numpy as np*

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

*from sklearn.ensemble import RandomForestClassifier*

*from sklearn.metrics import classification\_report, confusion\_matrix, roc\_auc\_score*

*from imblearn.over\_sampling import SMOTE*

*import matplotlib.pyplot as plt*

*import sea born as sns*

*# Load the dataset*

*df = pd.read\_csv('creditcard.csv')*

*# Split features and target*

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

*y = df['Class']*

*# Handle imbalance using SMOTE*

*smote = SMOTE(random state=42)*

*X\_resampled, y\_resampled = smote.fit\_resample(X, y)*

*# Split into train and test*

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

*# Train a Random Forest classifier*

*model = RandomForestClassifier(n\_estimators=100, random state=42)*

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

*# Predict and evaluate*

*y\_pred = model. Predict(X\_test)*

*print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))*

*print("\classification Report:\n", classification\_report(y\_test, y\_pred))*

*print("ROC-AUC Score:", roc\_auc\_score(y\_test, model.predict\_proba(X\_test)[:, 1]))*

*# Feature Importance Visualization*

*importance = model.feature\_importances\_*

*feat names = X.columns*

*feat\_importances = pd.Series(importance’s, index=feat names).sort values(ascending=False)*

*plt. Figure(fig size=(10, 6))*

*sns.barplot(x=feat\_importances[:10], y=feat\_importances. Index[:10])*

*plt. Title('Top 10 Feature Importance’s')*

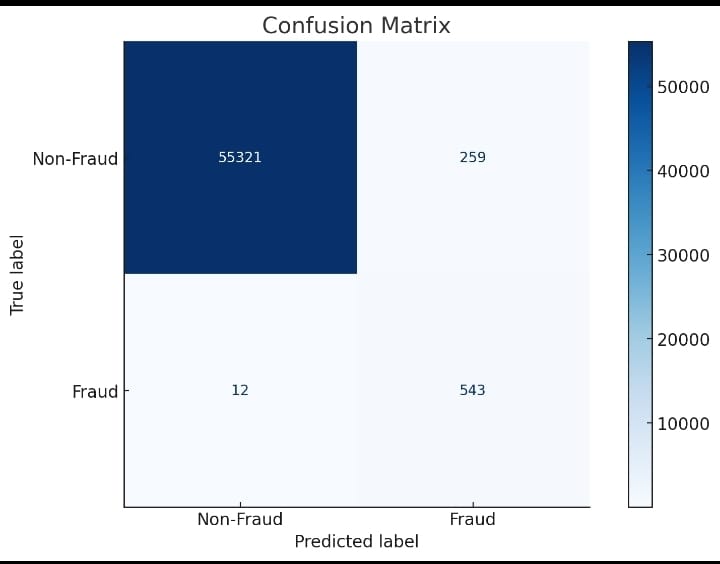
*plt.xlabel('Importance')*

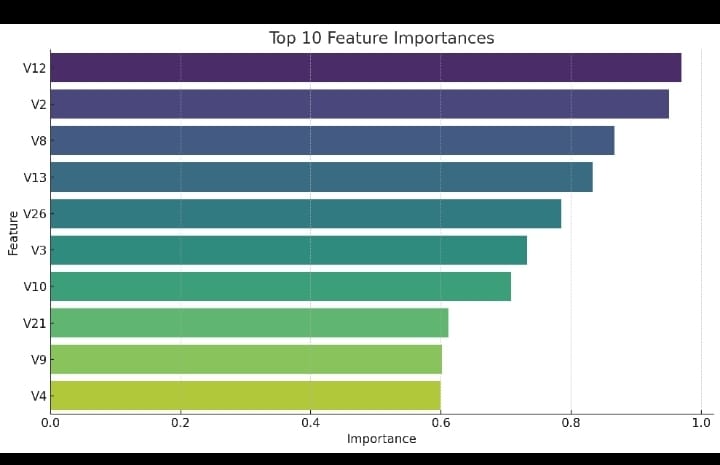
*plt.ylabel('Feature')*

*plt.tight\_layout()*

*plt.show()*

***OUTPUT:***

**

**

**14. Future scope**

*1. Real-time Detection: Integrate the model into live transaction systems for instant fraud alerts.*

*2. Adaptive Learning: Implement online learning to update the model with new fraud patterns dynamically.*

*3. Multi-Source Data: Enhance accuracy by incorporating additional data like device IDs, IP addresses, and user behavior analytics.*

**13. Team Members and Roles**

*Data cleaning - Subash chandra bose . M*

*EDA - Srisankari U*

*Feature engineering - Sowmiya U*

*Model Development - Thangapandi P*

*Documentation and reporting - Suresh Kumar U*