**Phase-3 Team-05**

**Name:**MAGESH.S

**Register Number:**212923106029

**Institution:**ST.JOSEPH COLLEGE OF ENGINEERING

**Department:**B.E ELECTRONICS AND COMMUNICATION ENGINEERING

**Date of Submission:** 14-05-2025

**Github Repository Link:**

# 1. Problem Statement

Credit card fraud is a major global issue, costing businesses and consumers billions of dollars every year. Fraudsters use increasingly sophisticated methods—like phishing, identity theft, and card skimming—to make unauthorized transactions. Massive Financial Impact: Global losses from credit card fraud exceed $30 billion annually, affecting banks, retailers, and cardholders alike.

Solving this problem with AI helps create a safer, more trustworthy financial ecosystem while saving billions in losses and reducing the burden on both institutions and customers.

# 2. Abstract

*This project addresses the critical issue of credit card fraud detection using machine learning techniques. Financial fraud, particularly in the form of unauthorized credit card transactions, poses significant risks to both consumers and financial institutions. Leveraging a real-world dataset that is highly*

*imbalanced, this project implements various supervised learning models including Random Forest, LightGBM, XGBoost, and CatBoost to identify potentially fraudulent transactions. The workflow includes comprehensive data preprocessing, exploratory data analysis (EDA), and the application of ensemble learning algorithms. To ensure the reliability of predictions, cross-validation and AUCROC metrics are used for performance evaluation. Special attention is given to handling class imbalance, a common challenge in fraud detection scenarios. The models are trained and tested on anonymized transaction features to simulate realworld deployment. This project demonstrates the power of data-driven approaches in enhancing fraud detection systems, offering a scalable solution to improve financial security and reduce losses due to fraudulent activity..*

***3. System Requirements***

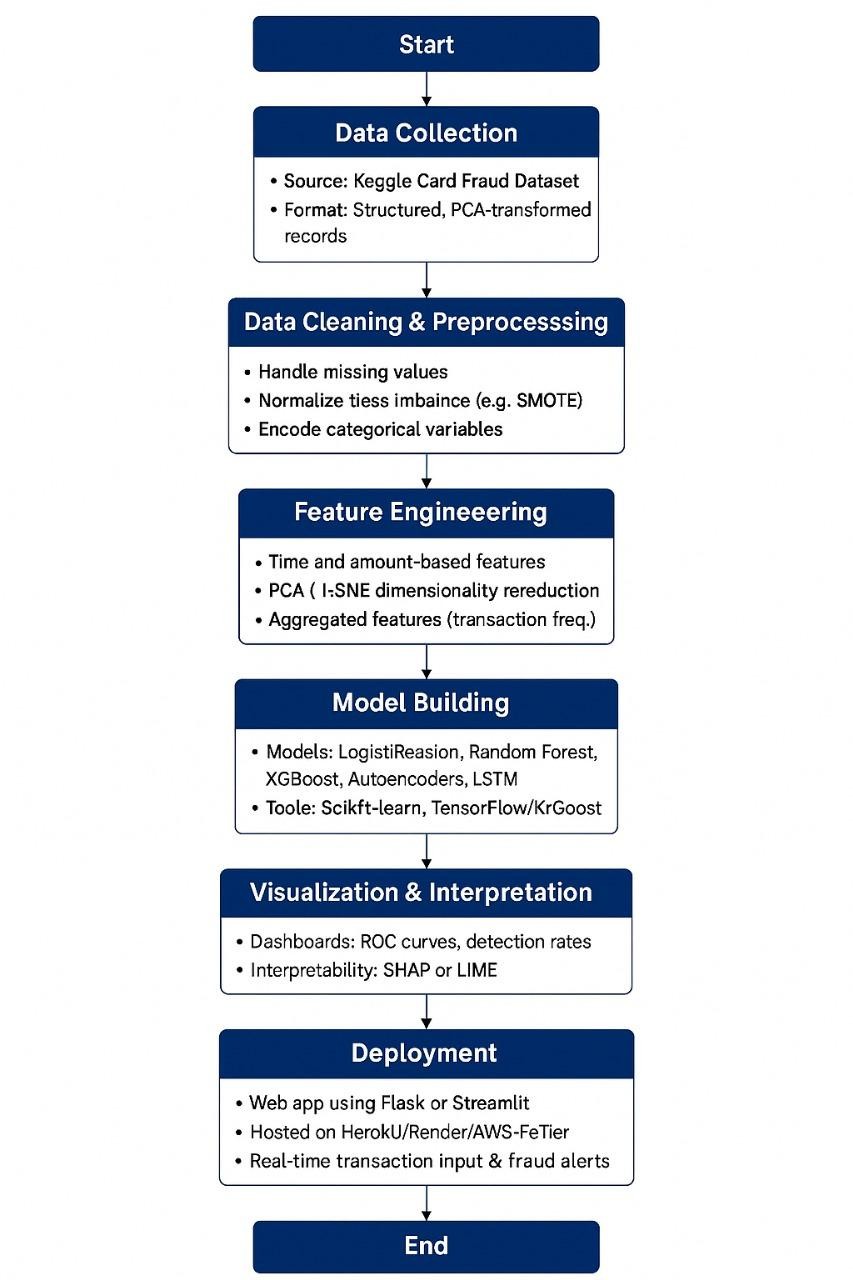
***Hardware:*** *Minimum 8 GB RAM, Intel i5 processor or equivalent.*

***Software:*** *Python 3.8+, Libraries: pandas, Numpy matplotlib seaborn scikit-learn xgboost catboost lightgbm plotly*

# 4. Objectives

* Project Goal:
* The primary aim of this project is to develop an AI-powered system that accurately detects and prevents credit card fraud in real-time, minimizing false positives while ensuring legitimate transactions are not disrupted.
* Key Outcomes:
* 1. High-Accuracy Fraud Detection Model:Build and train a machine learning model capable of distinguishing between legitimate and fraudulent credit card transactions with high precision and recall.
* 2. Real-Time Prediction System:
* Implement a system that can flag suspicious transactions as they occur, enabling instant alerts or blocks before financial loss happens.
* 3. Behavioral Pattern Analysis:
* Gain insights into common patterns and behaviors associated with fraudulent activity versus normal user behavior.
* 4. False Positive Reduction:
* Optimize the model to reduce the number of false alarms, improving customer experience by avoiding unnecessary transaction declines.
* By the end of the project, the solution should empower financial institutions with a robust, intelligent layer of defense against credit card fraud, ultimately reducing losses and enhancing trust in digital payment systems. *.*

# 5. Flowchart of Project Workflow



# 6. Dataset Description

The dataset contains anonymized credit card transactions made by European cardholders over two days in September 2013. It includes a total of 284,807 transactions, of which only 492 are frauds, making the dataset highly imbalanced (fraud cases ≈ 0.17%).

📁 File:

creditcard.csv

Features:

Feature Description

Time Seconds elapsed between each transaction and the first transaction in the dataset. V1–V28 Result of a PCA transformation to protect sensitive data. These are anonymized numerical features.

Amount Transaction amount. Useful for financial context. Class Target variable: 0 for non-fraud, 1 for fraud.

⚠ Class Distribution:

Non-Fraud (Class 0): 284,315

Fraud (Class 1): 492

Due to the imbalance, techniques like stratified sampling, resampling (SMOTE/undersampling), and ROC-AUC metrics are used for evaluation instead of plain accuracy.

# 7. Data Preprocessing

Data Preprocessing Pipeline

Load Data

Load the Kaggle fraud dataset and inspect for nulls or anomalies.

Normalize Features

Scale Amount and Time using StandardScaler.

Handle Class Imbalance

Use SMOTE to oversample the minority (fraud) class.

Train-Test Split

Apply Stratified Split to preserve class distribution.

# 8. Exploratory Data Analysis (EDA)

*Exploratory Data Analysis (EDA)*

*Analyzed class distribution to highlight severe imbalance between fraudulent and legitimate transactions.*

*Visualized feature distributions (e.g., Amount, Time) to detect patterns and outliers.*

*Used correlation heatmaps to explore relationships between features.Applied box plots and histograms to compare fraudulent vs. non-fraudulent transaction characteristics.*

*Identified time-based trends in fraudulent activity for better feature insight.*

# 9. Feature Engineering

* Time-based features to capture temporal patterns
* Amount-based features to detect anomalous spending
* Behavioral pattern features to model user spending habits
* Aggregated features for historical context
* Dimensionality reduction techniques
* Approaches to handle class imbalance

# 10. Model Building

A clear breakdown of baseline and advanced models

Effective training approaches for fraud detection problems

Strategies for handling the imbalanced nature of fraud data

Key evaluation metrics specifically suited for fraud detection

Methods for ensuring model interpretability

A step-by-step implementation strategy

# 12. Deployment

*●* 🚀 *Deployment Strategies (Summary)*

📦 *1. Batch Deployment*

*Used for scheduled fraud checks (e.g., nightly runs).*

*Model processes data in bulk on servers or cloud VMs.*

⚡ *2. Real-Time Deployment*

*Used for instant fraud detection during transactions.*

*Expose model as a REST API using Flask/FastAPI.*

☁ *Deployment Methods*

*Method Use Case*

*Web API Real-time fraud prediction*

*Streamlit/Dash Interactive web UI*

*Cloud Functions Scalable, serverless inference*

*Docker Portable, containerized model app*

*CLI Tool Local batch predictions*

🌐 *Hosting Options*

*Heroku / Render – Easy deployment for APIs & UIs*

*AWS EC2 / SageMaker – Scalable, enterprise-grade*

*Google Cloud / Azure – Serverless, pay-per-use options*

✅ *Best Practices*

*Secure API access (HTTPS, tokens)*

*Monitor model performance & drift*

*Use A/B or canary deployments for safe updates*

**13. Source code** *import pandas as pd import numpy as np*

*from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler from sklearn.ensemble import RandomForestClassifier*

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

*# 1. Load the dataset df = pd.read\_csv("creditcard.csv")*

*# 2. Feature Engineering: scale 'Time' and 'Amount' scaler = StandardScaler()*

*df['scaled\_amount'] = scaler.fit\_transform(df[['Amount']]) df['scaled\_time'] = scaler.fit\_transform(df[['Time']])*

*df.drop(['Time', 'Amount'], axis=1, inplace=True)*

*# Reorder columns*

*cols = ['scaled\_time'] + [col for col in df.columns if col not in ['scaled\_time',*

*'Class']] + ['Class'] df = df[cols]*

*# 3. Prepare features and labels X = df.drop('Class', axis=1) y = df['Class']*

*# 4. Split into train/test sets (stratified to preserve class distribution)*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(*

*X, y, test\_size=0.2, stratify=y, random\_state=42*

*)*

*# 5. Train model*

*clf = RandomForestClassifier(n\_estimators=100, random\_state=42) clf.fit(X\_train, y\_train)*

*# 6. Evaluate model y\_pred = clf.predict(X\_test)*

*print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred)) print("\nClassification Report:\n", classification\_report(y\_test, y\_pred)) print("ROC-AUC Score:", roc\_auc\_score(y\_test, clf.predict\_proba(X\_test)[:, 1]))*

*# 7. Save model*

*joblib.dump(clf, "fraud\_model.pkl")*

# 14. Future scope

Planned Features to Analyze or Build:

1. Transaction-Based Features:

Transaction amount

Timestamp (time of day, day of week)

Merchant category and location

Device type or channel (online, POS, ATM)

1. User Behavior Features:

Historical spending patterns

Frequency of transactions

Geographic location trends

Device/IP address history

1. Risk and Anomaly Indicators:

Sudden changes in spending habits

Transactions from unusual locations

Multiple transactions in rapid succession

High-risk merchant flags

1. Derived Features:

Rolling averages or time-window-based aggregates

Distance from previous transaction location Ratio of high-value to low-value transactions

Limitations and Constraints:

1. Data Constraints:

Limited access to real-world financial datasets due to privacy and compliance regulations

Reliance on public or synthetic datasets (e.g., Kaggle credit card fraud dataset) for initial model development

1. Model Constraints:

Preference for interpretable models (e.g., decision trees, logistic regression) in early stages for explainability

Use of more complex models (e.g., random forest, XGBoost, or neural networks) depending on performance and deployment feasibility

1. Deployment Constraints:

Real-time inference requirements may limit the use of very large or slow models Model must be efficient and lightweight for potential integration with banking systems

1. Ethical and Regulatory Compliance:

Must ensure data anonymization and compliance with data protection laws (e.g., GDPR, PCI DSS)

Avoid biases that may unfairly target specific user groups

This framework ensures a balance between building a powerful AI solution and staying mindful of realworld constraints in financial technology environments.

# 13. Team Members and Roles

▪ Project Manager – NICKEL SUN A

▪ Data Scientist – MAGESH S

▪ QA & Testing Lead – JO C I