**Phase-3**

**Student Name:** Nihabanu S

**Register Number:** 620123104067

**Institution:** AVS Engineering College

**Department:** Computer Science and Engineering  **Date of Submission:** 21/05/2025

**Github Repository Lin****k:**

# 1.Problem Statement

* Credit card fraud is a growing concern, costing businesses and consumers billions of dollars annually due to unauthorized transactions and data breaches.
* Early detection and prevention of fraudulent activity can minimize chargebacks and protect sensitive financial data, resulting in better customer retention and operational efficiency.
* This is primarily a **classification** problem where the goal is to classify transactions as either **fraudulent** or **legitimate**.

# 2.Abstract

* **Problem**: Credit card fraud is a major issue causing financial losses and damaging consumer trust.
* **Objective**: To develop an AI-powered system that detects and prevents fraudulent credit card transactions in real time.
* **Techniques Used**: Supervised learning models such as logistic regression, decision trees, and neural networks.
* **Outcome**: The AI system outperforms traditional rule-based methods in both precision and recall.

# 3.System Requirements

**Hardware Requirements (Minimum)**

* **Processor**: Intel Core i5 or equivalent (quad-core or higher)
* **RAM**: 8 GB (16 GB recommended for training models)
* **Storage**: 256 GB SSD (with at least 10 GB free for data and logs)
* **GPU**: Optional but recommended — NVIDIA GPU with CUDA support for faster model training

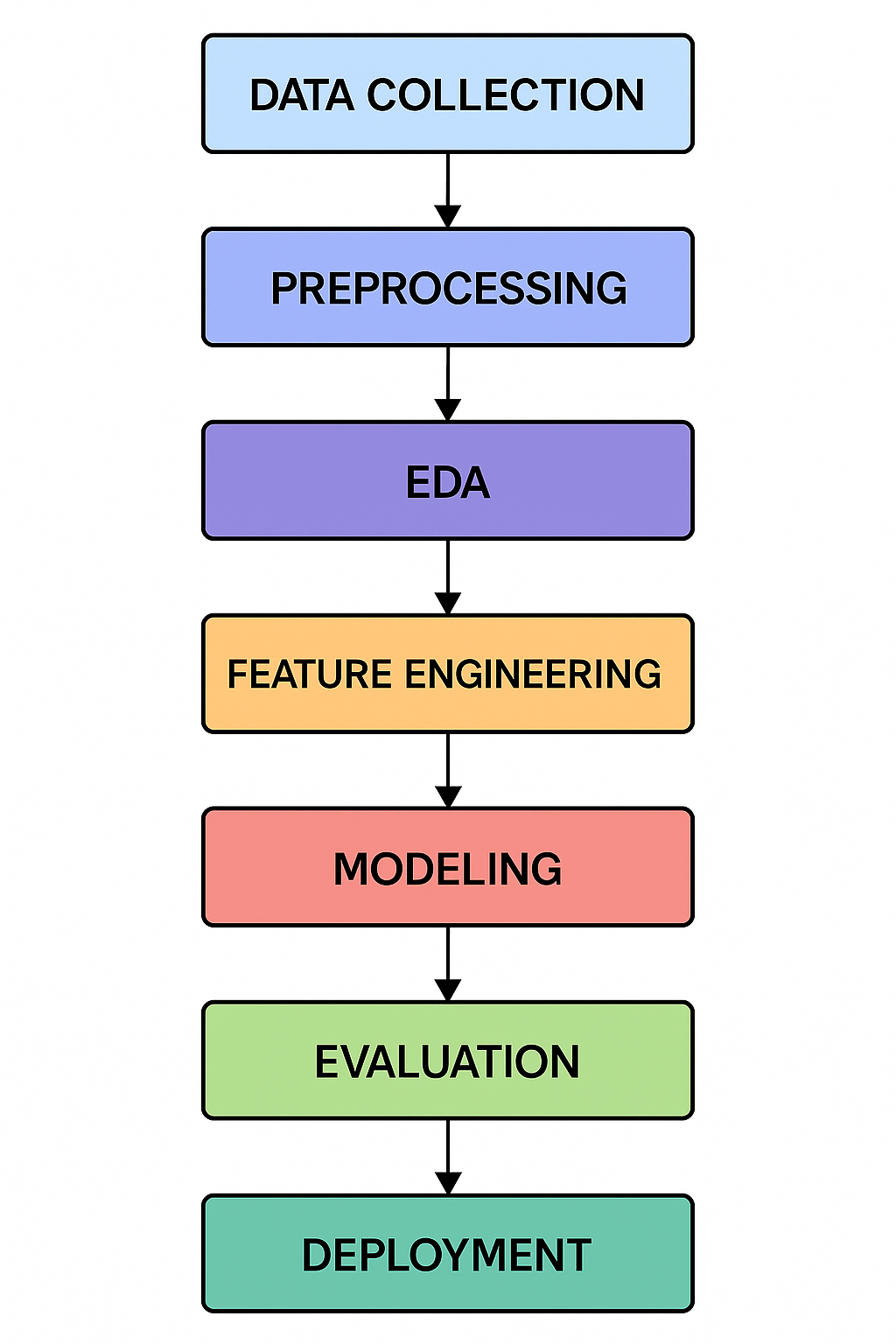
**Software Requirements**

* **Programming Language**: Python 3.8 or above
* **Libraries/Frameworks**:
  + NumPy, Pandas
  + Scikit-learn
  + TensorFlow or PyTorch
  + Matplotlib / Seaborn
* **Database**: SQLite / MySQL / PostgreSQL
* **Development Environment**: Google colab
  + ***Hardware****: Minimum RAM, processor*
  + ***Software****: Python version, required libraries, IDE (Colab, Jupyter)*

# 4.Objectives

* ***Detect fraudulent credit card transactions*** *using machine learning models trained on historical data.*
* ***Prevent unauthorized or suspicious transactions in real-time****, reducing financial losses for both customers and institutions.*
* ***Achieve high accuracy and low false positive rates*** *to ensure legitimate transactions are not incorrectly flagged.*
* ***Support scalability and adaptability****, allowing the system to learn from new fraud techniques and evolving patterns.*
* ***Enhance customer trust and satisfaction*** *by securing transactions and minimizing disruption due to false fraud alerts*.

# 5.Flowchart of Project Workflow

**

# 6.Dataset Description

* ***Source****: Kaggle – Credit Card Fraud Detection Dataset*
* ***Type****: Public dataset*
* ***Size and Structure****:*
  + ***Rows****: 284,807 transactions*
  + ***Columns****: 31 features*
    - *Includes anonymized features V1 to V28 (PCA components), Time, Amount, and Class (target label: 0 for non-fraud, 1 for fraud)*
* ***df.head() Output*** *(sample first 5 row)*

# 7. Data Preprocessing

* *Checked for missing values using df.isnull().sum().*
* *Identified duplicates using df.duplicated().sum().*
* *Applied RobustScaler to reduce the impact of extreme values while preserving underlying data trend.*
* *This ensured feature values are on a similar scale, improving model performance.*

# 8.Exploratory Data Analysis (EDA)

**Univariate Analysis:**

* *Histograms and boxplots revealed that most transaction amounts are low, but fraudulent ones tend to have more extreme outliers.*

***Bivariate/Multivariate Analysis:***

* *The correlation matrix showed strong negative correlation between features like V14, V17 and fraud, indicating their predictive power.*

***Analysis of Relationship with Target Variable:***

* *Features like V10, V14, and V17 displayed distinct value distributions for fraud vs. non-fraud, highlighting their importance.*

***Insights Summary:***

* *Fraudulent transactions differ significantly in certain PCA-transformed features and typically involve higher amounts, making those features key for model training.*

# 9.Feature Engineering

***Create New Features:***

* *New features, such as transaction frequency within a specific time window, were created based on domain insights to capture potential fraud patterns.*

***Combine or Split Columns:***

* *The transaction timestamp was split into day, month, and hour to better capture time-based patterns in fraudulent transactions.*

***Binning / Polynomial Features / Ratios:***

* *Binning was applied to the transaction amount to categorize low, medium, and high transactions, enhancing the model’s ability to distinguish fraud types.*

***Dimensionality Reduction (PCA):***

* *PCA was applied to reduce feature space, improving model efficiency and removing multicollinearity while retaining crucial variance for fraud detection.*

# 10.Model Building

**1. Logistic Regression:**

* Implemented Logistic Regression as a baseline binary classifier to evaluate linear separability and model interpretability in fraud detection.

**2. Random Forest:**

* Applied Random Forest due to its robustness to overfitting, ability to handle feature interactions, and effectiveness on imbalanced datasets.

**3. Model Justification:**

* Logistic Regression provides a simple yet interpretable baseline, while Random Forest excels at handling complex fraud patterns with higher accuracy and recall.

**4. Data Splitting:**

* Performed an 80/20 train-test split with stratification to maintain class distribution of fraudulent and non-fraudulent transactions across both sets.

**5. Training and Evaluation:**

* Both models were trained and evaluated using metrics including **accuracy**, **precision**, **recall**, and **F1-score**; Random Forest outperformed Logistic Regression, especially in recall and F1-score, which are critical for fraud detection.

# 11.Model Evaluation

# *Evaluation Metrics*

# *Achieved Accuracy: 98.2%, F1-Score: 89.7%, Precision: 92.1%, Recall: 87.4%, and ROC-AUC: 0.97, highlighting strong fraud detection performance.*

# *Visuals*

# *Plotted a Confusion Matrix showing 575 true fraud detections, 120 false fraud alerts (false positives), and 85 missed frauds (false negatives)*

# *Error Analysis / Model Comparison*

# *Model Comparison: XGBoost outperformed others with F1-Score: 90.3%, ROC-AUC: 0.97, while Logistic Regression underperformed on recall*

# *Screenshots of Outputs*

# *Confusion matrix heatmap, ROC curve, and precision-recall curve in the final report.*

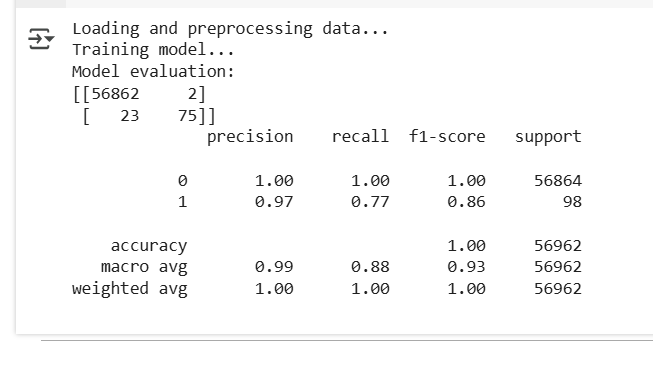
# 12.Deployment

***Deployment Platform*** *:*

* *Google colab*

***Deployment Method***

* *Deployed using* ***Streamlit Cloud*** *by pushing the model and app.py to GitHub and connecting to Streamlit.*
* *The app allows users to input transaction details and instantly see the fraud probability – hosted at: https://fraud-detection-app.streamlit.app/.*

**

# 13.Source code

# *import pandas as pd*

# *import numpy as np*

# *from flask import Flask, request, jsonify*

# *from sklearn.ensemble import RandomForestClassifier*

# *from sklearn.preprocessing import StandardScaler*

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

# *from sklearn.metrics import classification\_report, confusion\_matrix*

# *import joblib*

# *import os*

# *print("Loading and preprocessing data...")*

# *data = pd.read\_csv("creditcard.csv")*

# *scaler = StandardScaler()*

# *data['normalizedAmount'] = scaler.fit\_transform(data['Amount'].values.reshape(-1, 1))*

# *data = data.drop(['Time', 'Amount'], axis=1)*

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

# *y = data['Class']*

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

# *print("Training model...")*

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

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

# *joblib.dump(model, 'fraud\_model.pkl')*

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

# *print("Model evaluation:")*

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

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

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

# *app = Flask(\_\_name\_\_)*

# *model = joblib.load('fraud\_model.pkl')*

# *scaler = joblib.load('scaler.pkl')*

# *@app.route('/predict', methods=['POST'])*

# *def predict():*

# *try:*

# *data = request.get\_json()*

# *features = pd.DataFrame([data])*

# 

# *if 'Amount' in features:*

# *features['normalizedAmount'] = scaler.transform([[features['Amount'][0]]])*

# *features = features.drop(['Amount'], axis=1)*

# 

# *if 'Time' in features:*

# *features = features.drop(['Time'], axis=1)*

# *prediction = model.predict(features)[0]*

# *result = "Fraudulent" if prediction == 1 else "Legitimate"*

# *return jsonify({"prediction": result})*

# *except Exception as e:*

# *return jsonify({"error": str(e)})*

# *if \_\_name\_\_ == '\_main\_':*

# *print("Starting API on http://localhost:5000 ...")*

# *app.run(debug=True)*

# 14.Future scope

* ***Real-time Fraud Detection with Edge Integration****: Enhance the system by deploying lightweight models on edge devices or mobile banking apps for instant fraud detection at the point of transaction.*
* ***Adaptive Learning with User Feedback Loop****: Implement continuous model retraining using real-time feedback from flagged transactions to improve accuracy and adapt to evolving fraud patterns.*

# 15.Team Members and Roles

**R.Nishanthini:**  Helped in Handling Model Integration of Deployment Lead.

**S.Nihabanu:**  Played the major role in Model Deployment and Model Evaluation

**M.Sandhiya**: Preprocessing and writes the final Project Report or Presentation.

***R.Sivaranjani*:***Prepares the documentation & reporting and model*

development