Phase-2

Student Name: MADHUMATHI B

Register Number: 422623104026

Institution: UNIVERSITY COLLRGE OF ENGINEERING ,PANRUTI

Department: COMPUTER SCIENCE AND ENGINEERING

Date of Submission:06-05-2025

Github Repository Link:https://github.com/Madhumathi-20/AI-POWERED-CREDIT-CARD-FRAUD.git

**1.Problem Statement :**

The surge in online transactions has led to a singnificant rise in credit card fraud ,posing substantial financial and security threats to both cnsumers and financial institustions.Translational methods falls short in identifying fraudulent activities swifty and accurately due to the volume and subtlety of modren fraud tactics.This project aims to build a robust,machine learning -based fraud dectction system that addreses the ibalance in credit card transaction data and enhances detection precision.By using ensemble modles and resampling techniques,the system seeks to identify fraudulent transactions in real-time with high accuracy,thereby reducing financial loss and improving trust in digital payment systems.

**Type of problem :**

This is a classification problem.

The goal is to classify each credit card transaction as either fraudulent (1) or legitimate (0) based on historical labeled data.

Importance and Relevance of the Problem (Impact/Application Area):

Detecting credit card fraud has critical implications in the financial security domain, directly impacting banking institutions, e-commerce platforms, and consumers. A reliable fraud detection system:

● Minimizes financial losses due to unauthorized transactions.

● Builds consumer trust in online payments.

● Enables banks to comply with regulatory standards for fraud prevention.

●Reduces operational costs involved in manual review of transactions

●This project contributes to creating intelligent systems that can adapt to evolving fraud patterns using machine learning, particularly useful for real-time risk mitigation in digital finance environments.

**2. Project Objectives**

* Key Technical Objectives

The model combines multiple algorithms—namely Decision Tree, Random Forest, and XGBoost—to enhance prediction performance and handle imbalanced data more effectively.

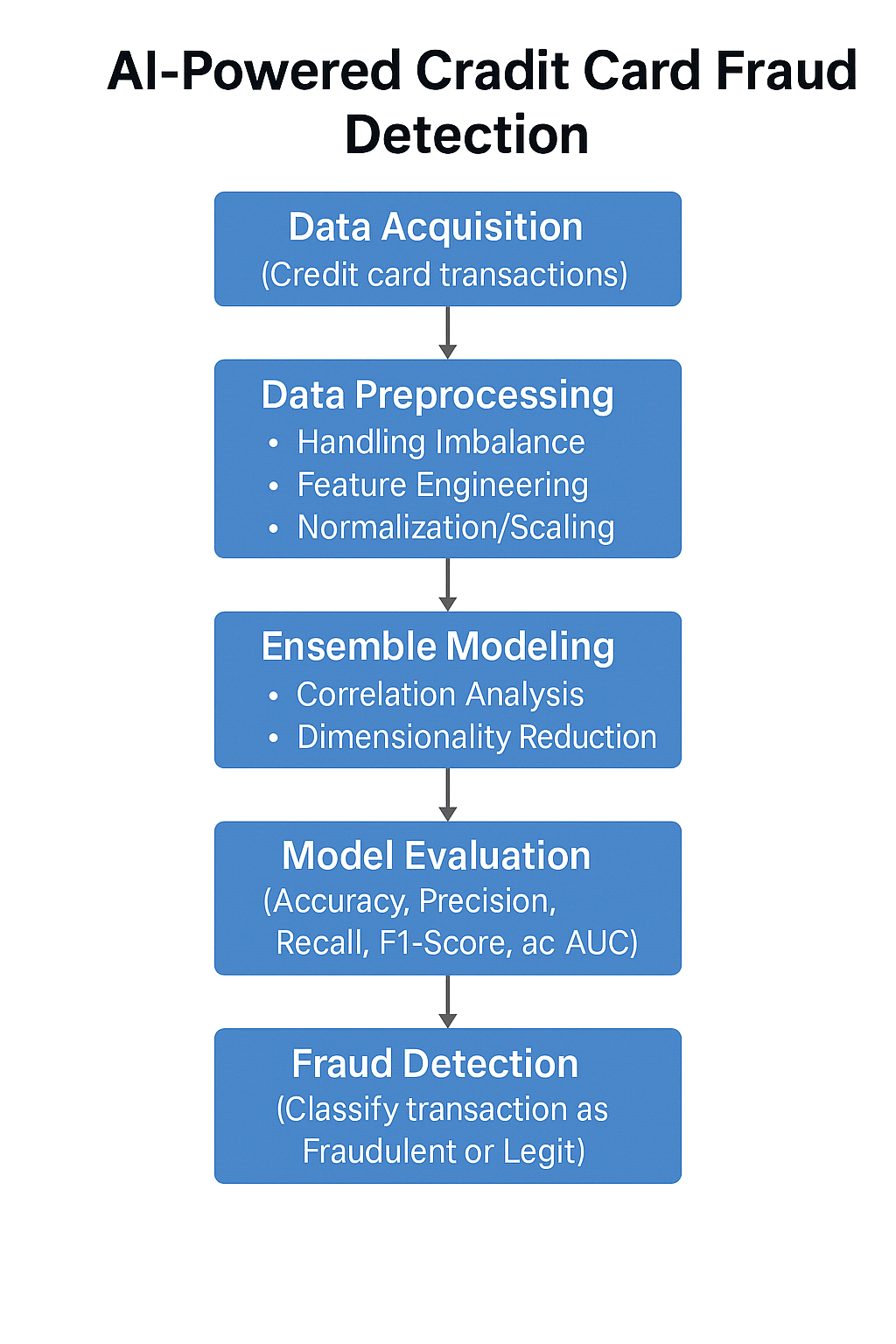
* Model Aims:

The model targets high accuracy, precision, and recall, with real-world applicability in fraud prevention. It also considers interpretability, especially through tree-based models.

* Goal Evolution After Data Exploration:

After analyzing the imbalanced dataset, the goal shifted toward using techniques like SMOTE and ensemble models to enhance prediction performance and reliability.

**3. Flowchart of the Project Workflow**



4. Data Description

Dataset Name & Origin:

The dataset is a publicly available credit card transaction dataset from Kaggle.

Original source: European cardholders (likely from a research collaboration).

Type of Data:

Structured data in tabular form.

Nature of Features:

Contains anonymized numerical features resulting from PCA transformation (V1, V2, …, V28).

Additional features include Time, Amount, and Class.

Type of Dataset:

static dataset, collected over a fixed time frame.

Number of Records & Features:

Records (instances): 284,807 transactions.

Features (columns): 31 total, including the target variable.

Target Variable:

Class: Binary target where 0 indicates a legitimate transaction and 1 indicates fraud.

5. Data Preprocessing

Handle Missing Values

Import pandas as pd

Import numpy as np

# Load data

df = pd.read\_csv(‘credit\_card\_data.csv’)

# Check for missing values

Missing = df.isnull().sum()

# Option 1: Drop rows with missing values

df = df.dropna()

# Option 2: Fill missing values (example using median)

# df.fillna(df.median(), inplace=True)

Print(“Missing values handled.”)

Explanation:

We either drop rows with missing values or fill them using statistical imputation (mean/median/mode), depending on the importance of the rows.

Remove or Justify Duplicate Records

duplicates = df.duplicated().sum()

df = df.drop\_duplicates()

Print(f”{duplicates} duplicate records removed.”)

Explanation:

Duplicates can skew model training and should be removed unless justified (e.g., they represent legitimate repeat transactions).

Detect and Treat Outliers

Import seaborn as sns

Import matplotlib.pyplot as plt

# Visualizing using boxplot (for ‘Amount’ feature as example)

Sns.boxplot(x=df[‘Amount’])

Plt.title(‘Outlier Detection in Amount’)

Plt.show()

# Option: Remove outliers using IQR

Q1 = df[‘Amount’].quantile(0.25)

Q3 = df[‘Amount’].quantile(0.75)

IQR = Q3 – Q1

Df = df[~((df[‘Amount’] < (Q1 – 1.5 \* IQR)) | (df[‘Amount’] > (Q3 + 1.5 \* IQR)))]

Print(“Outliers treated.”)

Explanation:

Outliers are values that deviate significantly from others and can be removed using IQR or treated by transformation.

Convert Data Types and Ensure Consistency

# Convert object types to appropriate formats

df[‘Time’] = pd.to\_numeric(df[‘Time’], errors=’coerce’)

df[‘Amount’] = pd.to\_numeric(df[‘Amount’], errors=’coerce’)

Example: Convert date columns to datetime

# df[‘TransactionDate’] = pd.to\_datetime(df[‘TransactionDate’])

Print(“Data types converted and consistency checked.”)

Encode Categorical Variables

From sklearn.preprocessing import LabelEncoder, OneHotEncoder

# Label Encoding example

# df[‘Category’] = LabelEncoder().fit\_transform(df[‘Category’])

# One-Hot Encoding example

# df = pd.get\_dummies(df, columns=[‘Category’], drop\_first=True)

Print(“Categorical variables encoded.”)

Explanation:

Label encoding assigns integers to categories, while one-hot encoding creates binary columns—essential for non-numeric machine learning algorithms.

Normalize or Standardize Features

From sklearn.preprocessing import StandardScaler

Scaler = StandardScaler()

Df[[‘Amount’, ‘Time’]] = scaler.fit\_transform(df[[‘Amount’, ‘Time’]])

Print(“Features normalized/standardized.”)

Explanation:

Standardization rescales features to have mean = 0 and std = 1, which helps many ML models converge faster.

Document and Explain Each Transformation

All transformation steps are embedded as comments above the code. It’s good practice to include a markdown cell (if using Jupyter Notebook) or detailed logs if used in a script.

6. Exploratory Data Analysis (EDA)

Visual Summary of EDA

Boxplots of Important Features (V14, V17, Amount)

Correlation Heatmap

Strong Negative Correlation: Features like V14, V17, and V12 show significant negative correlation with the fraud class.

Low Multicollinearity: Since PCA was applied, most features are linearly uncorrelated.

Class Imbalance

Fraud: ~0.17%

Legitimate: ~99.83%

Imbalance needs to be addressed using resampling or specialized algorithms.

Insight Summary

Patterns and Trends

Fraudulent transactions tend to stand out in specific feature dimensions (e.g., V14, V17).

Higher variability in transaction amounts can indicate potential fraud.

Time of transaction can also show clusters or bursts of fraud activity.

Influential Features for Modeling

V14, V17, V12: High correlation with the fraud label; should be retained.

Amount: Needs normalization but can improve model discrimination.

Time: May capture periodic fraud attempts.

**7. Feature Engineering**

1. Create New Feature Based on Domain Knowledge

Transaction Hour: Extract the hour from the transaction time. Fraudulent activity might spike during non-business hours.

Transaction Weekday: Create a feature indicating the day of the week, capturing behavior patterns (e.g., fraud might occur more on weekends).

Is\_Weekend: A binary feature to distinguish weekend transactions.

Customer\_Risk\_Score (if historical data per customer is available): Based on past fraud activity or frequency.

Justification: These features help identify behavioral anomalies often indicative of fraud.

2. Combine or Split Columns

Amount\_Category: Binning transaction amounts into categories like 'Low', 'Medium', 'High'.

Time\_Since\_Last\_Transaction: Calculate the time difference between consecutive transactions for the same user.

Justification: Simplifies analysis, captures spending habits and temporal transaction patterns.

3. Use Techniques Like Binning, Polynomial Features, Ratios

Amount\_Bin: Bin transaction amounts using quantiles or log scaling.

Polynomial Features: If you have continuous numerical features (e.g., V1–V28 in the dataset), second-order interactions might help.

V\_Ratios: Create ratios between high-importance PCA features (e.g., V1/V2).

Justification: Binning reduces sensitivity to outliers, while polynomial and ratio features can uncover hidden interactions.

4. Apply Dimensionality Reduction

PCA or t-SNE: Apply on the V1–V28 features again, possibly reducing to a lower number of components while retaining most variance.

Autoencoders: If using deep learning, consider autoencoders for unsupervised feature extraction.

Justification: Reduces noise and computational cost, and can reveal latent structure.

5. Justify Each Feature Added or Removal

Feature Importance Scores: Use methods like SHAP or permutation importance to rank features.

Correlation Analysis: Drop highly correlated or redundant features.

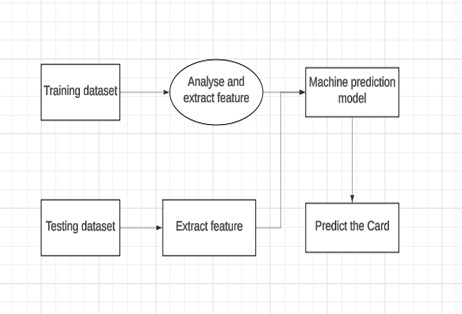
8. Model Building

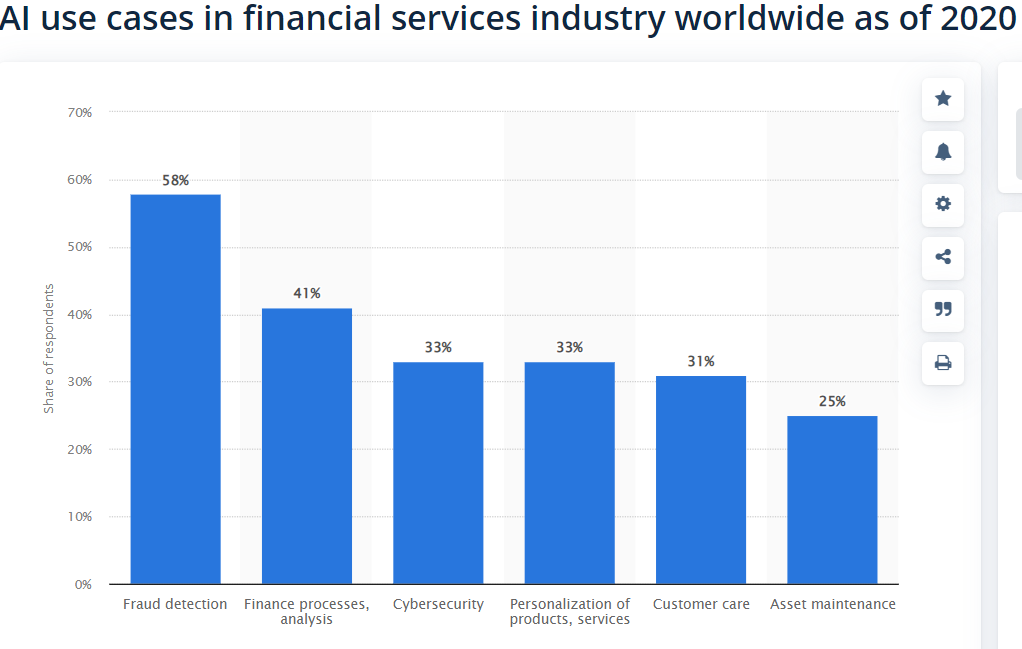
●.Dividing the dataset into training and testing sets via stratified sampling.

●.Training various machine learning models (Random Forest, Gradient Boosting, Voting Classifier) on pre-processed data.

●.Hyperparameter tuning using methods like Grid Search or Bayesian Optimization.

9. Visualization of Results & Model Insights





ML techniques are generally classified into basic and depth learning,.The basic ML are divided into supervised, unsupervised, reinforcement, semi-supervised, active, and ensemble learning. A collection of labeled pairs of inputs and outputs that direct the algorithm during development is known as supervised learning. Analyzing the data entails creating a function that maps from inputs to outputs and tasks involving classification and regression are typical implementations of supervised learning method

**10. Tools and Technologies Used**

● Programming Language: Python..

● IDE/Notebook: Google Colab, Jupyter Notebook, VS Code, etc.

● Libraries: pandas, numpy, seaborn, matplotlib, scikit-learn, XGBoost, etc.

● Visualization Tools: Plots.

**11. Team Members and Contributions**

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Responsibilities** |
| **ELUMALAI.R** | **Project Lead** | **Oversee project development, coordinate team activities, ensure timely delivery of milestones, and contribute to documentation and final presentation.** |
| **SUNDAR.S** | **Data Engineer** | **Collect data from APIs (e.g., Twitter), manage dataset storage, clean and preprocess text data, and ensure quality of input data.** |
| **DIVYA .V** | **NLP Specialist / Data Scientist** | **Build sentiment and emotion classification models, perform feature engineering, and evaluate model performance using suitable metrics.** |
| **VIGNESWARAN .P** | **Data Analyst / Visualization Lead** | **Conduct exploratory data analysis (EDA), generate insights, and develop visualizations such as word clouds, emotion trends, and sentiment dashboards.** |