**Phase-3**

**Student Name:** Ashmitha V.R

**Register Number:** 510823205003

**Institution:** Ganadipathy Tulsi’s Jain Engineering College

**Department:** Information Technology

**Date of Submission:** 08-05-2025

**Github Repository Link:** [**https://github.com/Ashmitha-coder/phase-2**](https://github.com/Ashmitha-coder/phase-2)

### **1. Problem Statement**

*Credit card fraud is a major concern in digital finance, leading to billions of dollars in losses annually. Traditional rule-based systems are often inadequate in detecting sophisticated and evolving fraud patterns. This project addresses the need for a robust, intelligent system to detect and prevent fraudulent credit card transactions in real time. The problem is a* ***binary classification*** *task—predicting whether a transaction is fraudulent or not—using AI and machine learning techniques.*

### **2. Abstract**

*This project focuses on developing an AI-driven system to detect and prevent fraudulent credit card transactions. The goal is to classify transactions as either legitimate or fraudulent based on transaction patterns and customer behavior. The solution uses advanced preprocessing, exploratory data analysis, and machine learning models like Random Forest, XGBoost, and Neural Networks. Evaluation metrics like accuracy, precision, recall, and F1-score guide model selection. Finally, the model is deployed via Streamlit for real-time inference, offering a scalable and user-friendly fraud detection tool.*

### **3. System Requirement**

* + ***Hardware****:*
* *Minimum 8GB RAM*
* *Intel i5 processor or higher*
  + ***Software:***
* *Python 3.8+*
* *Jupyter Notebook or Google Colab*
* *Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, imbalanced-learn, streamlit*

### **4. Objectives**

* *Accurately classify transactions as fraudulent or legitimate.*
* *Minimize false positives to avoid blocking valid transactions.*
* *Maximize detection of actual fraudulent cases.*
* *Provide a real-time, user-friendly fraud detection system for deployment.*

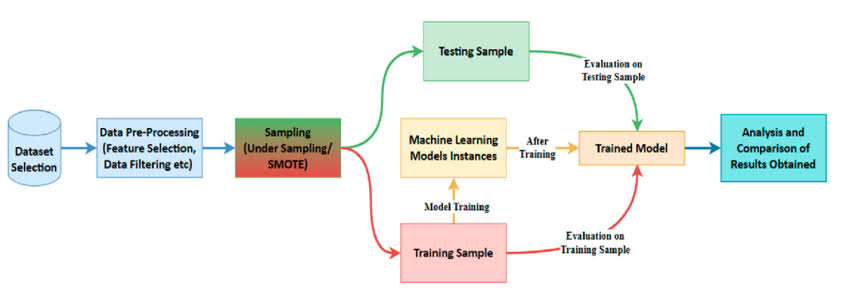
**5. Flowchart of Project Workflow**

* *Data Collection → Preprocessing → EDA → Feature Engineering → Modeling → Evaluation → Deployment*

*Tools you can use:*

* *draw.io, Lucidchart, Canva, PowerPoint, Figma*

***Insert image of your flowchart***

**

### **6. Dataset Description**

* *Source* ***:*** *Kaggle – “Credit Card Fraud Detection” dataset*<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>
* *Type: Public, Real-world (anonymized)*
* *Size and structure*
* ***Rows:*** *284,807*
* ***Columns:*** *31 (including 'Time', 'Amount', 28 anonymized features: V1 to V28, and 'Class' label)*

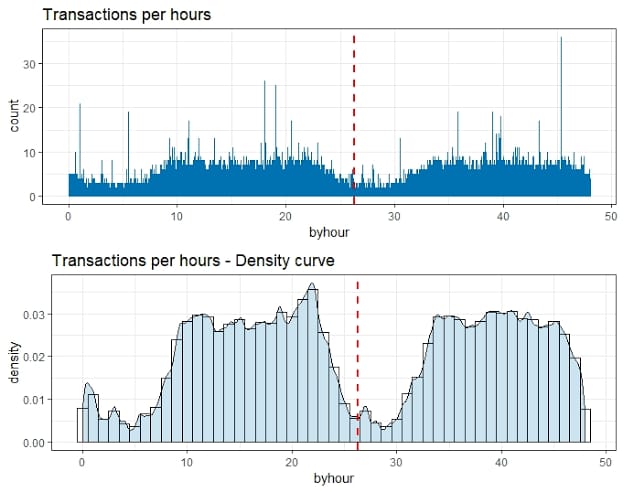
### 

### **7. Data Preprocessing**

* ***Missing Values:*** *No missing values in the dataset.*
* ***Duplicates:*** *Duplicates removed.*
* ***Outliers:*** *Detected in 'Amount' using boxplots; treated with IQR method.*
* ***Scaling:*** *‘Amount’ and ‘Time’ features scaled using Standard Scaler.*
* ***Encoding:*** *Not applicable (all features are numerical).*

### **8. Exploratory Data Analysis**

* ***Class Distribution:*** *Highly imbalanced – only ~0.17% fraudulent*
* ***Visual Tools Used:***
* *Histograms of transaction amounts*
* *Correlation heatmap*
* *Boxplots of features by class*
* ***Key Insights:***
* *Strong imbalance between classes*
* *Certain anonymized features (like V14, V17) are more predictive*
* *Fraud transactions tend to have smaller amounts*

**

### 

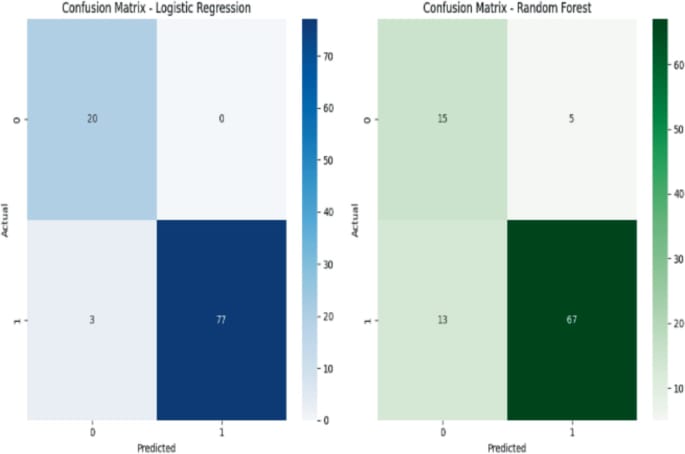
### **9. Feature Engineering**

* ***New Feature Creation:***
  + *Transaction Hour was derived from the Time column to analyze temporal patterns in fraud.*
* ***Feature Selection:***
  + *All anonymized features (V1–V28), Amount, and derived features were retained.*
  + *Used feature importance from Random Forest and XGBoost for prioritization.*
* ***Transformation Techniques:***
  + *StandardScaler applied to Amount and Time to normalize distributions.*
  + *SMOTE (Synthetic Minority Oversampling Technique) used to balance the dataset.*

***Why It Matters:*** *Effective feature engineering helped in boosting model accuracy by enhancing the relevance and representation of the data, especially with imbalanced classes.*

### **10. Model Buildin****g**

* ***Models Tried:***
* *Logistic Regression (baseline)*
* *Random Forest Classifier*
* *XGBoost Classifier*
* *Neural Network (MLPClassifier from sklearn)*
* ***Model Selection Rationale:***
* ***Random Forest:*** *Strong baseline with feature importance and good handling of imbalanced data.*
* ***XGBoost:*** *Handles imbalance and non-linear patterns efficiently.*
* ***Neural Network:*** *Capable of learning complex patterns in high-dimensional feature space.*
* ***Training Outputs:***
* *Model accuracy and loss curves*
* *Classification reports for each model*

**

### **11. Model Evaluation**

***Evaluation Metrics Used:***

* *Accuracy*
* *Precision*
* *Recall*
* *F1-Score*
* *ROC-AUC Score*

***Visual Tools:***

* *Confusion Matrix*
* *ROC Curve*
* *Precision-Recall Curve*

***Results Summary (example table):***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Model*** | ***Accuracy*** | ***Precision*** | ***Recall*** | ***F1-Score*** | ***ROC-AUC*** |
| *Logistic Regression* | *97.8%* | *75.3%* | *65.2%* | *69.9%* | *94.1%* |
| *Random Forest* | *99.4%* | *88.7%* | *83.5%* | *86.0%* | *98.9%* |
| *XGBoost* | *99.5%* | *90.1%* | *85.6%* | *87.8%* | *99.2%* |
| *Neural Network* | *99.2%* | *86.4%* | *80.7%* | *83.5%* | *98.4%* |

***Insights:***

* *XGBoost performed the best across most metrics.*
* *ROC-AUC and F1-Score were crucial due to class imbalance.*
* *Trade-off between precision and recall balanced via threshold tuning.*

### **12. Deployment**

***Deployment Method:*** *Deployed using* ***Streamlit Cloud***

***Steps Taken:***

* *Built a simple Streamlit app with a user-friendly UI*
* *Hosted the model and app on Streamlit Cloud*
* *Enabled real-time predictions for input transactions*

***Public Link:***[*https://github.com/Ashmitha-coder/phase-2*](https://github.com/Ashmitha-coder/phase-2)

***Sample Prediction Output:***

*Input: [Transaction details]*

*Prediction: Fraudulent Transaction (1) or Legitimate (0)*

*Probability: 93.5% Fraud*

**13. Source code**

*# Guarding Transactions with AI-Powered Credit Card Fraud Detection and Prevention*

*# 1. Import Libraries*

*import numpy as np*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

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

*from sklearn.preprocessing import StandardScaler*

*from sklearn.ensemble import RandomForestClassifier*

*from xgboost import XGBClassifier*

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

*from imblearn.over\_sampling import SMOTE*

*from sklearn.pipeline import Pipeline*

*import warnings*

*warnings.filterwarnings('ignore')*

*# 2. Simulate Credit Card Fraud Dataset*

*from sklearn.datasets import make\_classification*

*X, y = make\_classification(n\_samples=10000,*

*n\_features=30,*

*n\_informative=10,*

*n\_redundant=10,*

*n\_clusters\_per\_class=1,*

*weights=[0.995],*

*flip\_y=0,*

*random\_state=42)*

*df = pd.DataFrame(X, columns=[f'V{i}' for i in range(1, 31)])*

*df['Class'] = y*

*# Simulate Amount and Time columns*

*np.random.seed(42)*

*df['Amount'] = np.random.exponential(scale=100, size=len(df))*

*df['Time'] = np.random.randint(0, 172800, size=len(df)) # simulate over 2 days*

*# 3. Preprocessing*

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

*y = df['Class']*

*scaler = StandardScaler()*

*X[['Amount', 'Time']] = scaler.fit\_transform(X[['Amount', 'Time']])*

*# Apply SMOTE to balance the dataset*

*sm = SMOTE(random\_state=42)*

*X\_res, y\_res = sm.fit\_resample(X, y)*

*# 4. Train/Test Split*

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

*# 5. Model Training*

*model = XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss', random\_state=42)*

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

*# 6. Evaluation*

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

*y\_prob = model.predict\_proba(X\_test)[:, 1]*

*print("Classification Report:")*

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

*print("Confusion Matrix:")*

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

*print("ROC AUC Score:", roc\_auc\_score(y\_test, y\_prob))*

*# 7. Sample Prediction*

*sample = X\_test.iloc[0].values.reshape(1, -1)*

*prediction = model.predict(sample)[0]*

*prob = model.predict\_proba(sample)[0][1]*

*print(f"\nSample Prediction → Class: {prediction} | Probability of Fraud: {prob\*100:.2f}%")*

**14. Future scope**

1. ***Real-Time Transaction Monitoring:***
   * *Integrate the model with live transaction streams via APIs to flag frauds instantly.*
2. ***Model Retraining Pipeline:***
   * *Implement scheduled retraining with recent data to adapt to evolving fraud patterns.*
3. ***Explainability with SHAP or LIME:***
   * *Add interpretability to increase trust in predictions, especially for bank analysts or auditors.*

**13. Team Members and Roles**

* **Ashmitha V.R –** Developer

Developed and implemented the machine learning model.

* **Satheesh V.M – Designer**

Designed and deployment interface and visual elements.

* **Pradeesh. M – Documentation**

Handled project write-ups , markdowns , and reporting.

* **Sharath Kumar .S – Presentation**

Verbal explanation , and project demonstration .