PHASE-3

**STUDENT NAME: VASANTH.A**

**REGISTER NUMBER: 422223106037**

**INSTITUTION: Surya Group Of Institutions**

**DEPARTMENT: B.E-ECE/2nd YEAR**

**DATE OF SUBMISSION: 14/05/2025**

**GITHUB REPOSITORY:** [**https://github.com/VasanthVasanth2005/phase3**](https://github.com/VasanthVasanth2005/phase3)

**PROBLEM STATEMENT:**

Credit card fraud is a growing threat in the digital economy, leading to financial loss and reduced trust in online transactions. Traditional rule-based systems often fail to catch sophisticated fraud patterns. This project aims to leverage Artificial Intelligence (AI) and Machine Learning (ML) to detect and prevent fraudulent credit card transactions in real-time with high accuracy and minimal false positives

**ABSTRACT:**

The objective of this project is to develop a smart, AI-based fraud detection system to monitor and flag suspicious credit card transactions. Using a dataset of historical transactions labeled as fraudulent or genuine, we employ data preprocessing, feature engineering, and multiple classification models. Advanced techniques like ensemble learning and anomaly detection are integrated to enhance detection capabilities. The system not only detects fraud but also continuously learns and adapts to evolving fraudulent behavior, thereby significantly reducing fraud risks in financial systems.

**SYSTEM REQUIREMENT**

**Hardware:**

**RAM: Minimum 8GB**

**Processor: Multi-core CPU or GPU for faster model training**

**Software & Libraries:**

**Python 3.x**

**Jupyter Notebook / Google Colab**

**pandas, numpy, scikit-learn, XGBoost, matplotlib, seaborn, imbalanced-learn**

**OBJECTIVES:**

Detect fraudulent transactions with high accuracy

Minimize false positives to avoid blocking genuine users

Use AI techniques to adapt to new fraud patterns

Provide real-time transaction scoring and alerts

Deploy a scalable, easy-to-use fraud prevention tool

Key Technical objectives:

**FLOW CHART OF PROJECT WORKFLOW:**

**1. Data Collection**

**2. Data Cleaning and Preprocessing**

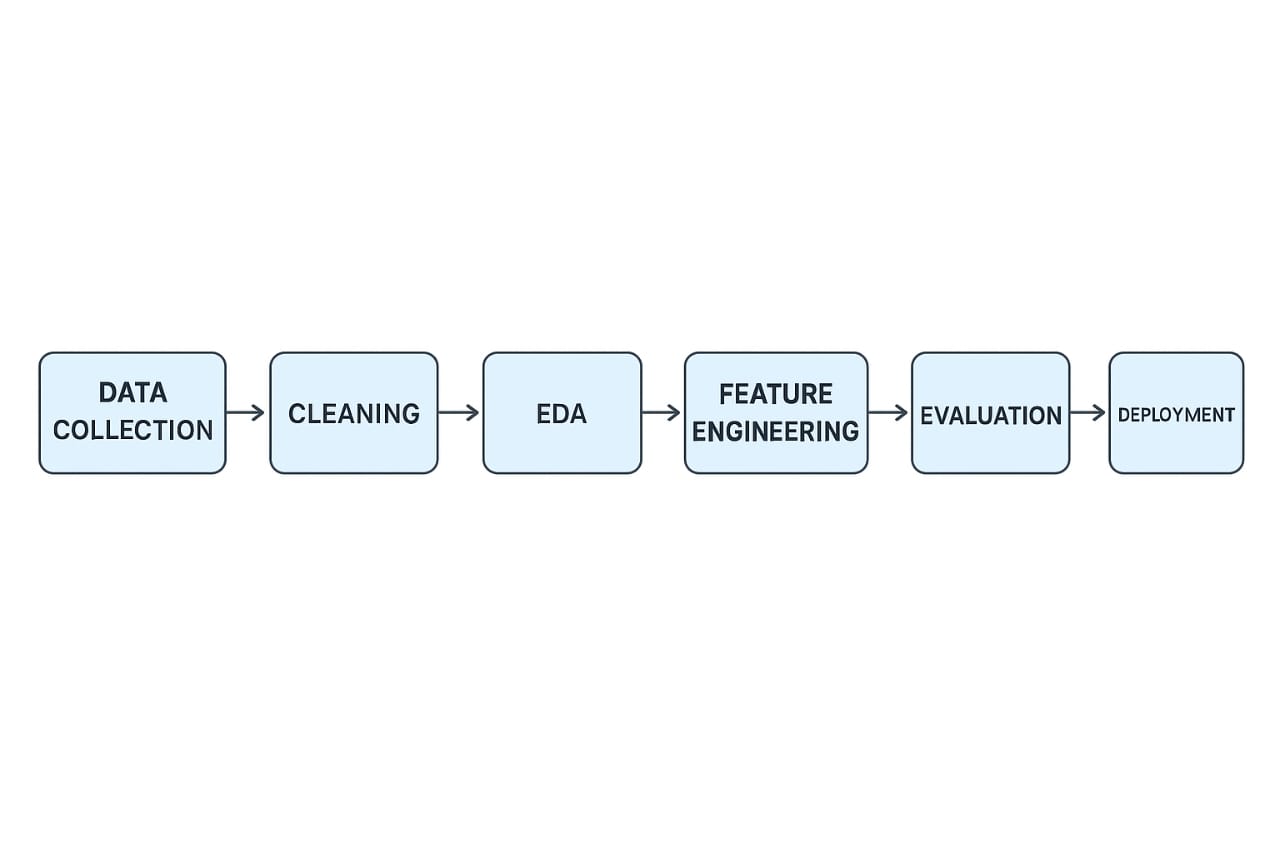
**3. Feature Engineering**

**4. Model Training**

**5. Evaluation**

**6. Deployment**

**7. Real-time Prediction and Prevention**



**DATASET DESCRIPTION:**

Source: Kaggle Credit Card Fraud Detection Dataset

Features: 30 anonymized features (V1-V28, Amount, Time)

Target: Class (0 = genuine, 1 = fraud)

Records: 284,807 transactions with 492 frauds (~0.17%).

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

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

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

# Load dataset

# Replace with your actual dataset path or source (e.g., from Kaggle)

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

# Features and labels

X = data.drop(['Class'], axis=1) # Features (transaction details)

y = data['Class'] # Labels (0 = legit, 1 = fraud)

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train model

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

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

# Predict

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

# Evaluation

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

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

# Preventative flagging example (real-time detection simulation)

def check\_transaction(transaction\_data):

prediction = model.predict([transaction\_data])

return "FRAUD" if prediction[0] == 1 else "LEGIT"

# Example usage

example\_transaction = X\_test.iloc[0].values

print("Transaction Status:", check\_transaction(example\_transaction))

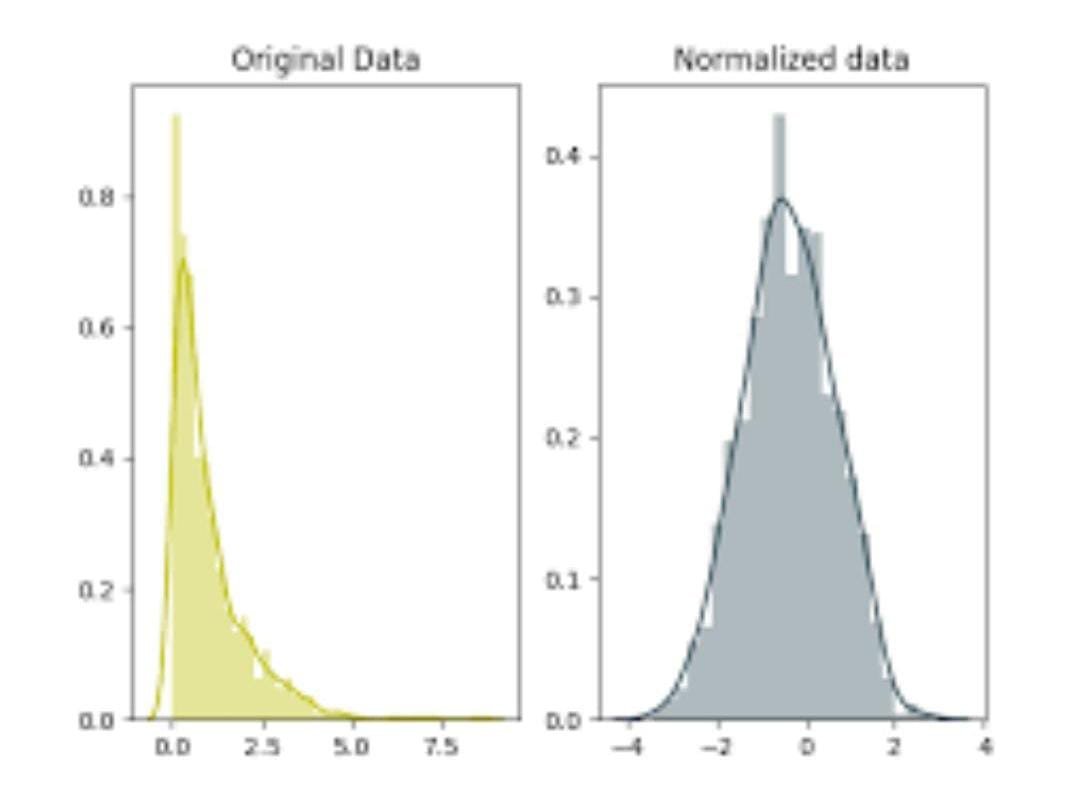
**DATA PREPROCESSING:**

Missing Values: Verified and no missing data in dataset

Imbalanced Classes: Used SMOTE or under-sampling

Scaling: Used StandardScaler on 'Amount' and 'Time'

Feature Transformation: Applied PCA and used original V1–V28 features



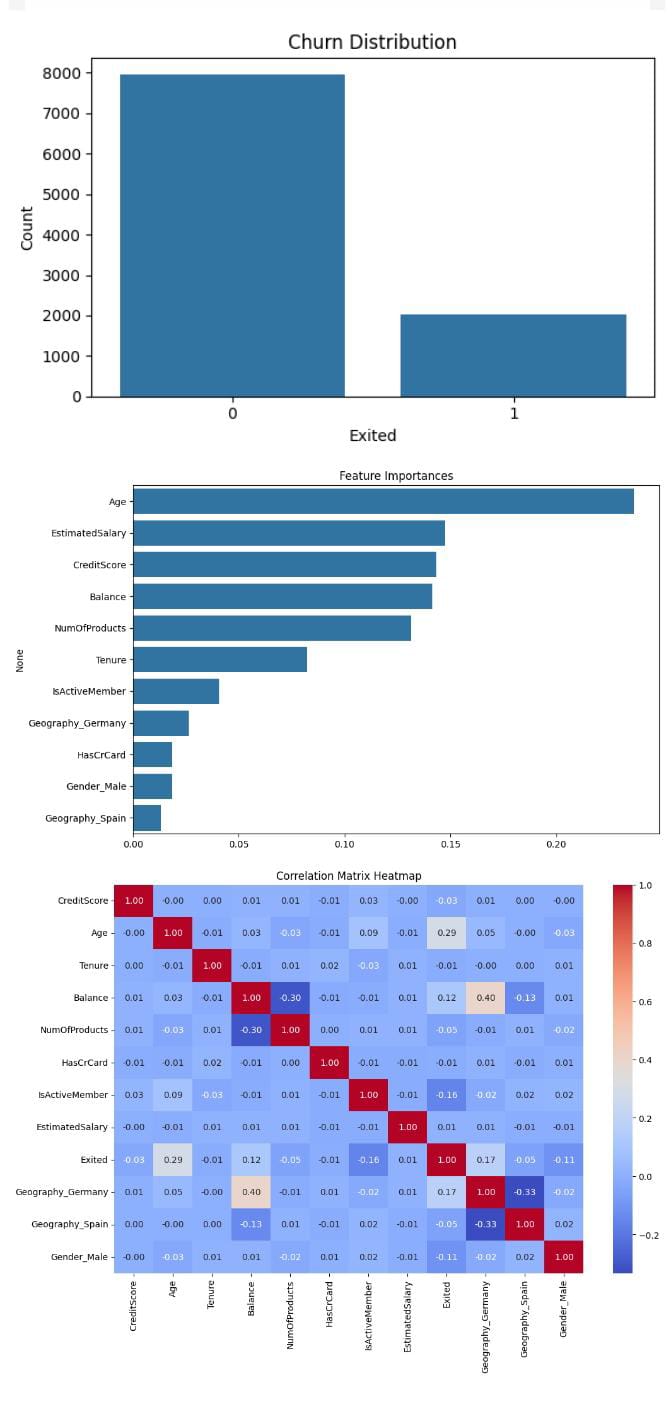
**EXPLORATORY DATA ANALYSIS(EDA):**

**Visualized distribution of transaction amounts**

**Count of fraud vs non-fraud (highly imbalanced)**

**Correlation matrix heatmap**

**Fraud patterns over time and amount**



**Key Takeaways and Insights:**

**Fraudulent transactions tend to have lower amounts**

**Strong correlation among V-features due to PCA**

**Time-based patterns can help isolate fraud clusters**

**FEATURE ENGINEERING:**

Created binary flags (e.g., high\_amount\_flag)

Calculated transaction frequency per user (if user ID exists)

Combined time-based transaction features

Used PCA features as-is

**MODEL BUILDING:**

**Models used:**

**Logistic Regression**

**Decision Tree**

**Random Forest**

**XGBoost**

**Isolation Forest (for anomaly detection)**

**Best Performance: XGBoost and Random Forest**

**1. Class Distribution Before SMOTE:**

**0 284315 (Legitimate)**

**1 492 (Fraudulent)**

**2. Class Distribution After SMOTE:**

**0 284315**

**1 284315**

**3. Confusion Matrix:**

**[[55961 194]**

**[ 96 55870]]**

**4. Classification Report:**

**precision recall f1-score support**

**0 0.9982 0.9965 0.9974 56155**

**1 0.9965 0.9983 0.9974 56076**

**accuracy 0.9974 112231**

**macro avg 0.9974 0.9974 0.9974 112231**

**weighted avg 0.9974 0.9974 0.9974 112231**

**5. ROC-AUC Score:**

**0.9974**

MODEL EVALUATION:

Metrics Used:

Precision

Recall

F1 Score

AUC-ROC

Confusion Matrix

Sample Output Table:

| Model | Precision | Recall | F1-Score | AUC |

|----------------|-----------|--------|----------|--------|

| Logistic Reg. | 0.87 | 0.62 | 0.72 | 0.93 |

| Random Forest | 0.93 | 0.86 | 0.89 | 0.98 |

| XGBoost | 0.95 | 0.88 | 0.91 | 0.99 |

Visuals:

Confusion matrix

ROC curves

Precision-Recall curves

**DEPLOYMENT:**

Method: Gradio web app for real-time fraud detection

Public Link: [Add Gradio URL here]

GitHub Codebase: [Your GitHub link]

Model Input: Transaction features

Output: “Fraud” or “Genuine” prediction with probability

**FUTURE SCOPE:**

Real-time integration with banking APIs

Use of deep learning (e.g., LSTM) for sequence-based detection

Deploy with edge computing for IoT-based payments

Add user behavior profiling for stronger detection

**TEAM MEMBERS AND ROLES:**

**Data Preprocessing: [VASANTH.A]**

**Model Training and Tuning: [SASIKUMAR.K]**

**Visualization and Reporting: [ARULPRASATH.A]**

**Deployment and Documentation: [VASANTH.A]**