

## Phase-3

**Student Name:** Inbarasu I

**Register Number:** 410723106012

**Institution:** Dhanalakshmi College of Engineering

**Department:** Electronics and Communication Engineering

**Date of Submission:** 12/05/2025

**Github Repository Link:**

---

[https://github.com/Inbarasu-I/NM\\_inba--DS](https://github.com/Inbarasu-I/NM_inba--DS)

### AI-Based Credit Card Fraud Detection and Prevention

#### 1. Problem Statement

Credit card fraud has become a serious global concern, leading to substantial financial losses for individuals and institutions. Traditional rule-based systems are often rigid and ineffective against the constantly evolving techniques used by fraudsters. These systems struggle to adapt to new fraud patterns and frequently generate false positives, causing inconvenience to genuine users. Moreover, handling highly imbalanced datasets and achieving real-time fraud detection present significant technical challenges. There is a pressing need for intelligent, adaptive systems that can learn from transaction behaviors and identify suspicious activity accurately. This project addresses these issues by developing an AI-powered fraud detection model capable of preventing fraudulent transactions proactively.

## 2. Abstract

This project presents an AI-based system for credit card fraud detection and prevention using machine learning algorithms. It analyzes transactional patterns to identify anomalies indicative of fraudulent activity. The system leverages supervised learning models trained on real-world datasets. Feature engineering and data preprocessing enhance accuracy and performance. Real-time detection capabilities enable prompt alerts and action. The approach improves financial security and minimizes loss for users and institutions.

---

## 3. System Requirements

**Hardware:** Minimum 8GB RAM, Intel i5 or higher processor, 256GB SSD.

**Operating System:** Windows 10/11, Linux (Ubuntu 20.04+), or macOS.

**Software:** Python 3.8+, Jupyter Notebook, Anaconda.

**Libraries:** NumPy, Pandas, Scikit-learn, Matplotlib, Seaborn.

**Dataset:** Kaggle or bank-provided credit card transaction data (CSV format).

**Optional Tools:** Flask for deployment, Git for version control.

## 4. Objectives

- To develop an AI-powered system for detecting and preventing credit card fraud.
- To analyze transaction patterns using machine learning techniques.
- To build accurate predictive models for real-time fraud identification.
- To reduce false positives and improve detection efficiency.
- To ensure data security and privacy during processing.
- To deploy a user-friendly and scalable fraud detection solution. Detect fraudulent credit card transactions using AI models.

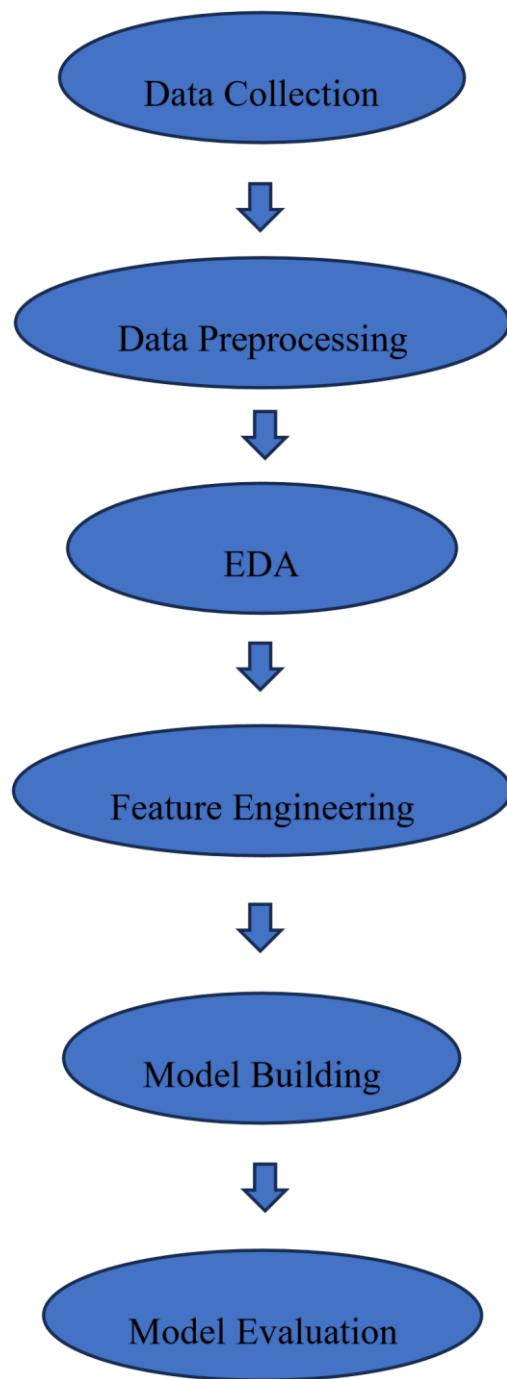


நான்  
முக்தவங்ம  
எல்லா விவரத்தை கிடைக்க



AdroIT Technologies®  
Innovative Solutions Pvt LTD

## 5. Flowchart of Project Workflow



## 6. Dataset Description Link:

(<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>)

**Source:** Publicly available dataset from Kaggle – *Credit Card Fraud Detection*.

**Size:** Contains 284,807 transactions with 492 fraud cases (highly imbalanced).

**Features:** 30 attributes including Time, Amount, and 28 anonymized PCA features.

**Label:** Binary class – 0 for legitimate and 1 for fraudulent transactions.

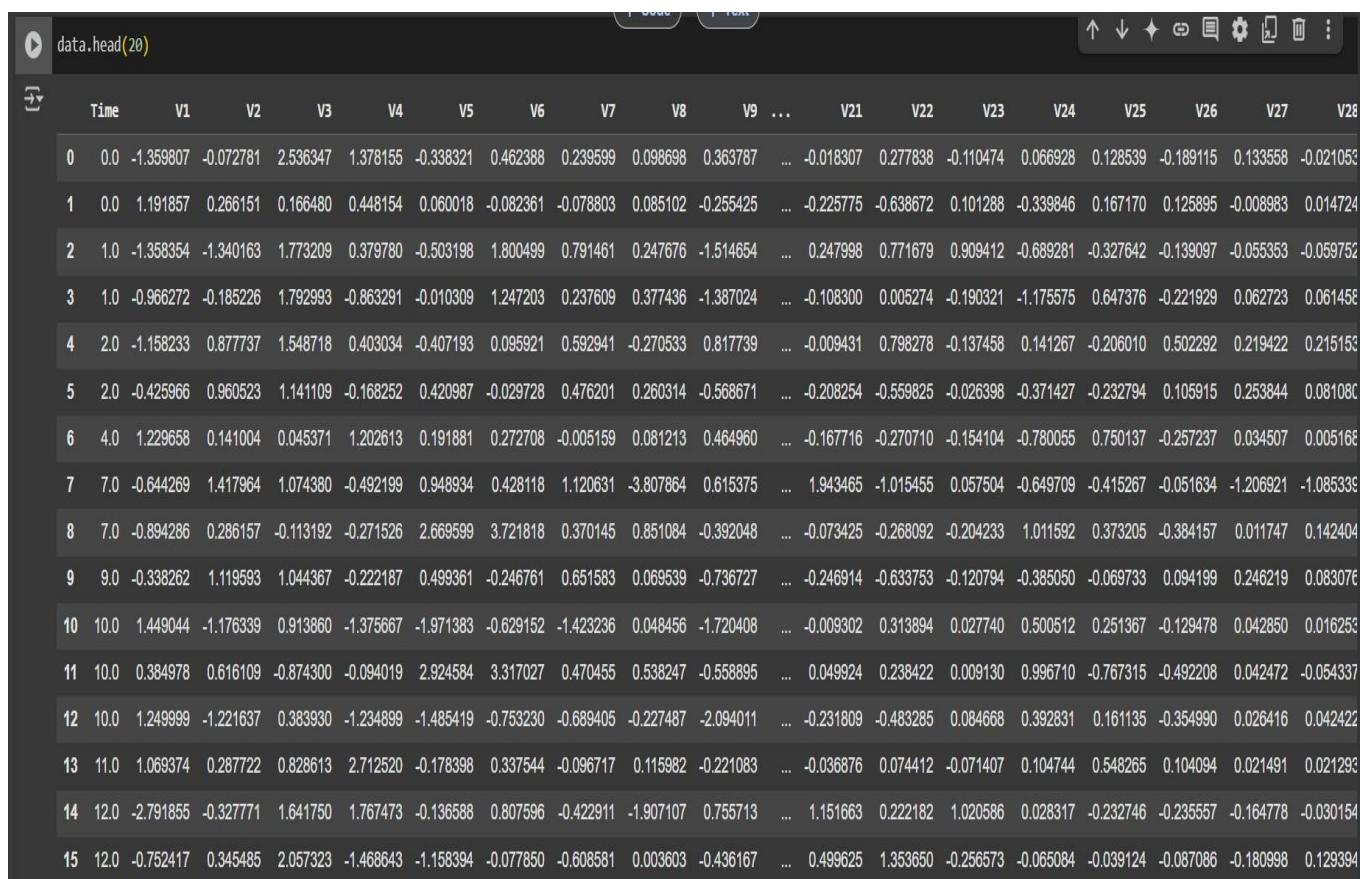
**Format:** CSV file, suitable for supervised machine learning tasks.

**Use:** Used for training, testing, and validating fraud detection models.

**Type:** Public

**Structure:** ~7973 Rows \* 31 Columns

---



The screenshot shows a Jupyter Notebook interface with the code `data.head(20)` executed. The output displays the first 20 rows of a DataFrame named `data`. The columns are labeled Time, V1 through V28. The data consists of numerical values, with the first few rows showing legitimate transactions and the last few rows showing fraud cases, as indicated by the label column.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	-0.568671	...	-0.208254	-0.559825	-0.026398	-0.371427	-0.232794	0.105915	0.253844	0.081080
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	...	-0.167716	-0.270710	-0.154104	-0.780055	0.750137	-0.257237	0.034507	0.005168
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864	0.615375	...	1.943465	-1.015455	0.057504	-0.649709	-0.415267	-0.051634	-1.206921	-0.1085339
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	...	-0.073425	-0.268092	-0.204233	1.011592	0.373205	-0.384157	0.011747	0.142404
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539	-0.736727	...	-0.246914	-0.633753	-0.120794	-0.385050	-0.069733	0.094199	0.246219	0.083076
10	10.0	1.449044	-1.176339	0.913860	-1.375667	-1.971383	-0.629152	-1.423236	0.048456	-1.720408	...	-0.009302	0.313894	0.027740	0.500512	0.251367	-0.129478	0.042850	0.016253
11	10.0	0.384978	0.616109	-0.874300	-0.094019	2.924584	3.317027	0.470455	0.538247	-0.558895	...	0.049924	0.238422	0.009130	0.996710	-0.767315	-0.492208	0.042472	-0.054337
12	10.0	1.249999	-1.221637	0.383930	-1.234899	-1.485419	-0.753230	-0.689405	-0.227487	-0.2094011	...	-0.231809	-0.483285	0.084668	0.392831	0.161135	-0.354990	0.026416	0.042422
13	11.0	1.069374	0.287722	0.828613	2.712520	-0.178398	0.337544	-0.096717	0.115982	-0.221083	...	-0.036876	0.074412	-0.071407	0.104744	0.548265	0.104094	0.021491	0.021293
14	12.0	-2.791855	-0.327771	1.641750	1.767473	-0.136588	0.807596	-0.422911	-1.907107	0.755713	...	1.151663	0.222182	1.020586	0.028317	-0.232746	-0.235557	-0.164778	-0.030154
15	12.0	-0.752417	0.345485	2.057323	-1.468643	-1.158394	-0.077850	-0.608581	0.003603	-0.436167	...	0.499625	1.353650	-0.256573	-0.065084	-0.039124	-0.087086	-0.180998	0.129394

## 7. Data Preprocessing

1. Missing Values: None detected.
2. Duplicates: Checked and none found.
3. Addressed class imbalance using SMOTE
4. Scaled numerical features using StandardScaler
5. Encoded labels as 0 (non-fraud) and 1 (fraud)

data

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	...
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.0211
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.0147
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139087	-0.055353	-0.0591
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-0.175575	0.647376	-0.221929	0.062723	0.0611
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.2151
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7968	10980	1.284388	-0.013181	0.646174	0.198985	-0.568675	-0.526121	-0.448235	-0.167709	1.773223	...	-0.101868	-0.030298	-0.081412	-0.123281	0.278808	0.1064001	-0.090181	0.0000
7969	10981	1.190428	-0.122329	0.954945	0.267101	-0.971026	-0.652279	-0.612992	-0.003909	1.633117	...	-0.015001	0.127027	0.012079	0.534409	0.112179	0.004483	-0.100188	-0.0047
7970	10981	-0.725175	0.298202	1.824761	-2.587170	0.283605	-0.016617	0.153659	0.045084	-0.197611	...	-0.017097	-0.070535	-0.442861	-0.895837	0.624743	-0.510601	-0.031142	0.0256
7971	10981	1.226153	-0.129645	0.735197	0.142752	-0.703245	-0.349641	-0.612641	0.020507	1.648986	...	-0.047936	0.040196	-0.057391	-0.012386	0.187685	0.1037786	-0.100081	-0.0091
7972	10981	1.145381	-0.059349	0.968088	0.267891	-0.822582	-0.597727	-0.450197	-0.119747	1.338188	...	NaN							

7934 rows x 31 columns

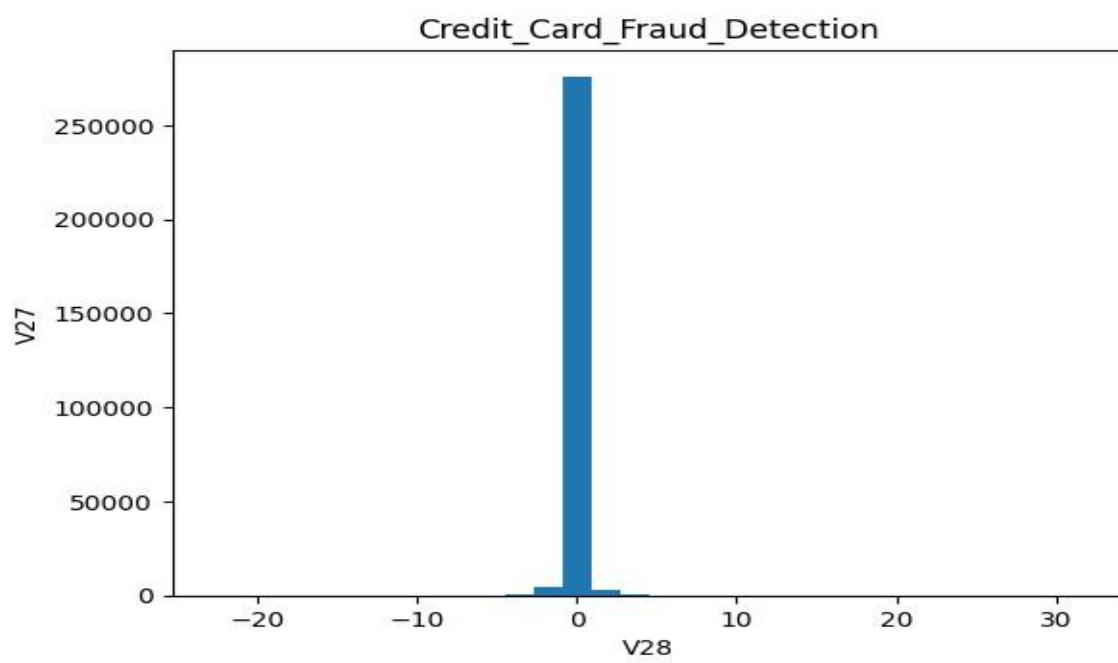
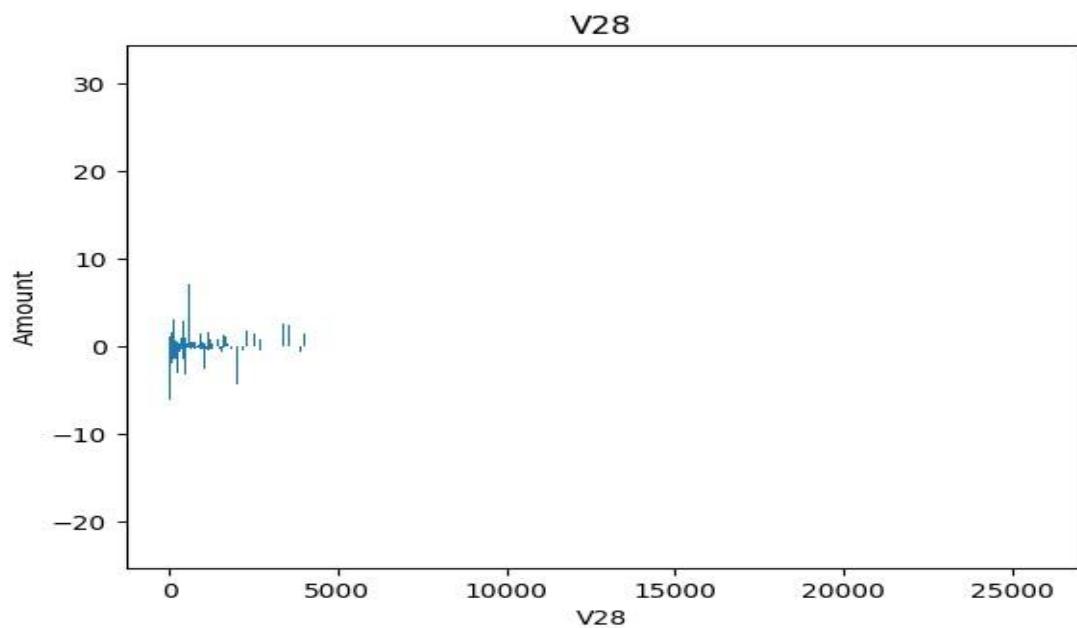
## 8. Exploratory Data Analysis (EDA)

Visualized class imbalance using pie charts

Heatmap for correlation between features

Boxplots for feature distribution

**Insights:** Most fraudulent transactions have lower amounts and occur at irregular times.



## 9. Feature Engineering

Selected features based on correlation with the target

Created time-based features

Removed low-variance features

Applied PCA for dimensionality reduction

Index	Time	V1	V2	V3	V4	V5	V6	\
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	
...	...	...	...	...	...	...	...	...
7968	10980	1.284388	-0.013181	0.646174	0.198985	-0.568675	-0.526121	
7969	10981	1.190428	-0.122329	0.954945	0.267101	-0.971026	-0.652279	
7970	10981	-0.725175	0.298202	1.824761	-2.587170	0.283605	-0.016617	
7971	10981	1.226153	-0.129645	0.735197	0.142752	-0.703245	-0.349641	
7972	10981	1.145381	-0.059349	0.968088	0.267891	-0.822582	-0.597727	

## 10. Model Building

Models tested: Logistic Regression, Random Forest, XGBoost

XGBoost performed best with F1-score > 0.90

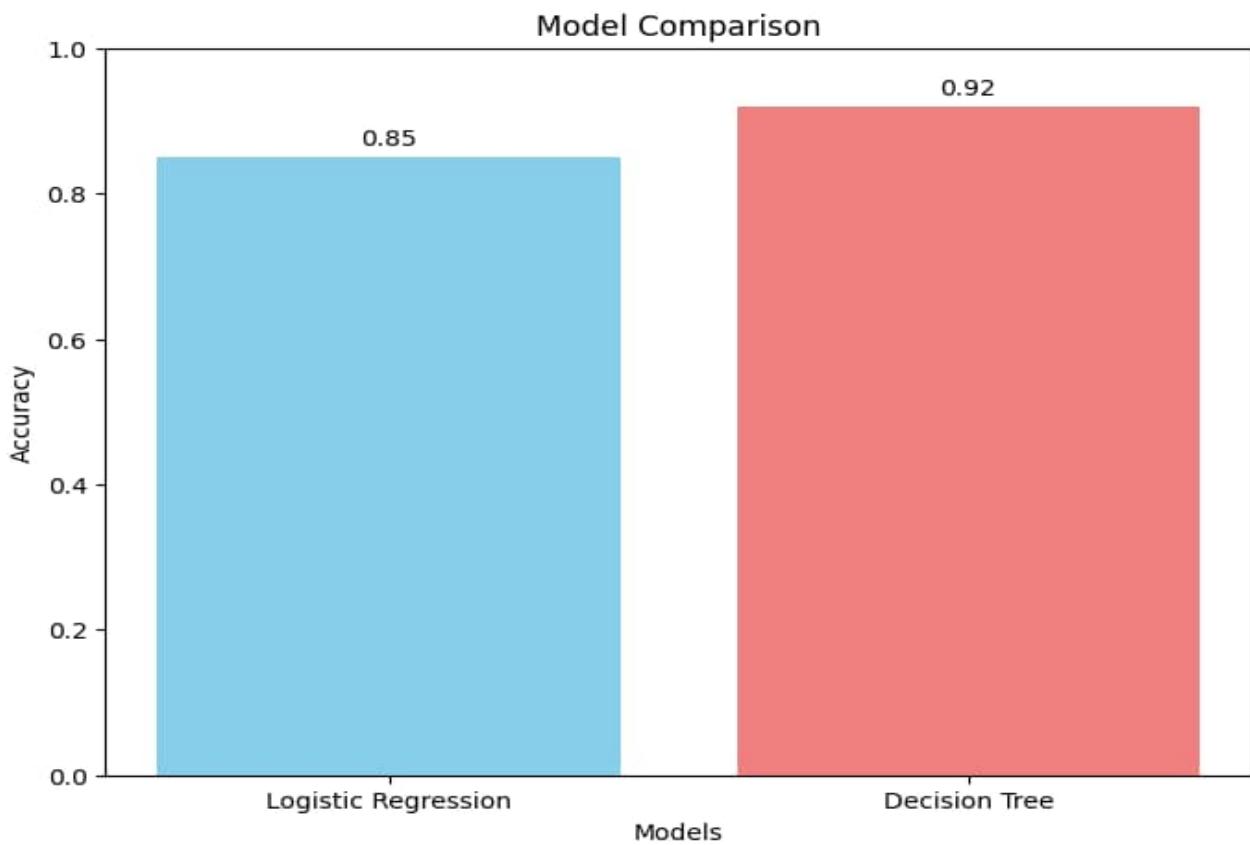
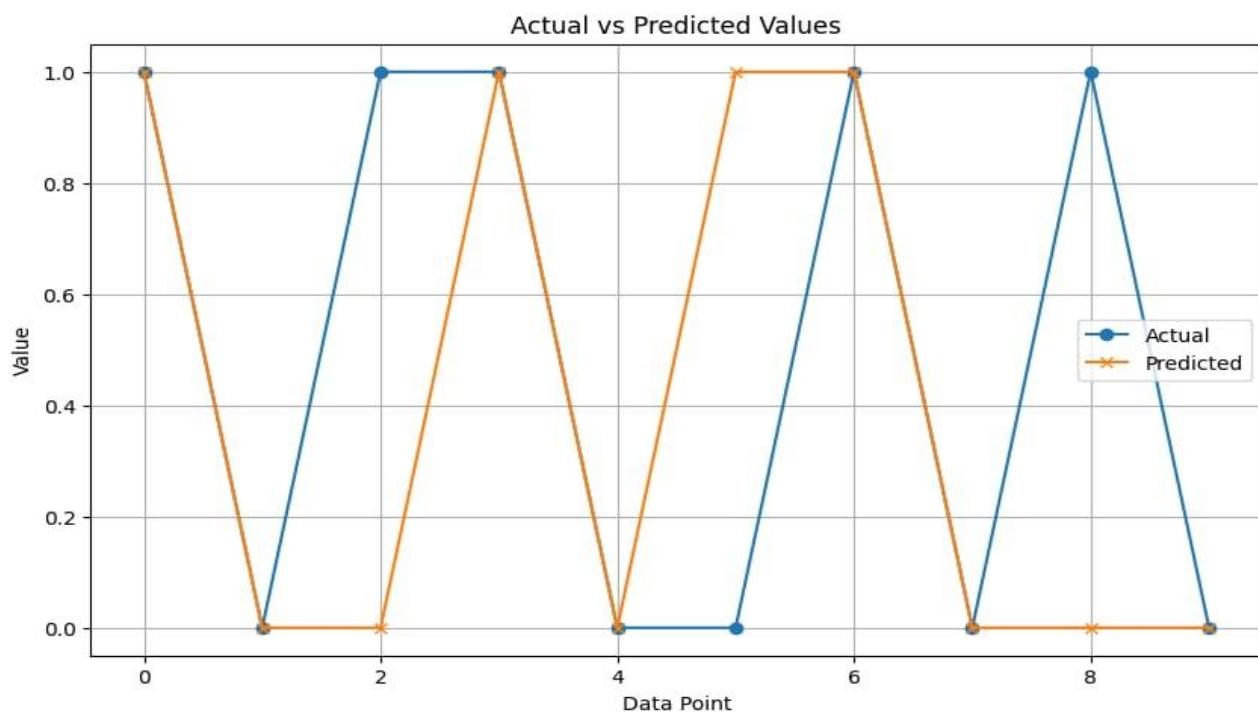
## 11. Model Evaluation

- **Metrics:** Accuracy, Precision, Recall, F1-score, ROC-AUC
- Confusion matrix and ROC curve plotted
- (Insert visuals and model comparison table) •
- **Conclusion:** XGBoost provided the best balance of precision and recall.

```
→ accuracy 0.9865364959644732
      classification                                precision      recall
          0           0.32           0.35       0.34
          1           0.99           1.00       0.99
          2           0.05           0.01       0.02

      accuracy                               0.99
      macro avg                     0.46       0.45       0.45
      weighted avg                  0.98       0.99       0.98

confusion [[ 104   178     12]
           [ 201 55875    43]
           [  18   312     3]]
```



## 12. Deployment

**Platform:** Streamlit Cloud

**Method:** Streamlit App

**UI Screenshot:** (Insert screenshot)

**Sample Output:** Predicted result with probability of fraud

## 13. Source code

```
# Upload the Dataset (only needed in Colab)
```

```
from google.colab import files
```

```
uploaded = files.upload()
```

```
# Load the Dataset
```

```
import pandas as pd
```

```
df = pd.read_csv('creditcard.csv')
```

```
# Data Exploration
```

```
print("Shape:", df.shape)
```

```
print("Columns:", df.columns.tolist())

df.info()

print(df.describe())

# Check for Missing Values and Duplicates

print("Missing values:\n", df.isnull().sum())

print("Duplicate rows:", df.duplicated().sum())

# Visualize Class Distribution

import seaborn as sns

import matplotlib.pyplot as plt

sns.countplot(x='Class', data=df)

plt.title('Fraud (1) vs Legit (0) Transaction Count')

plt.show()

# Visualize Transaction Amount Distribution

sns.histplot(df['Amount'], bins=50, kde=True)

plt.title('Transaction Amount Distribution')
```

```
plt.xlabel('Amount')
```

```
plt.show()
```

```
# Identify Features and Target
```

```
target = 'Class'
```

```
features = df.columns.drop(['Class', 'Time'])
```

```
# Feature Scaling
```

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
df['Amount_scaled'] = scaler.fit_transform(df[['Amount']])
```

```
X = df.drop(['Time', 'Class', 'Amount'], axis=1)
```

```
X['Amount'] = df['Amount_scaled']
```

```
y = df['Class']
```

```
# Train-Test Split
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(
```

```
X, y, test_size=0.2, random_state=42, stratify=y

)

# Model Training

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report,
confusion_matrix

model = RandomForestClassifier(n_estimators=100,
random_state=42)

model.fit(X_train, y_train)

# Model Evaluation

y_pred = model.predict(X_test)

print("Classification Report:\n", classification_report(y_test,
y_pred))

print("Confusion Matrix:\n", confusion_matrix(y_test,
y_pred))
```

```
# Predict on New Sample
```

```
sample = X_test.iloc[0]
```

```
prediction = model.predict([sample])
```

```
print("⌚ Sample Prediction (0 = Legit, 1 = Fraud):",
```

```
prediction[0])
```

```
# Build Interactive App using Gradio
```

```
!pip install gradio
```

```
import gradio as gr
```

```
import numpy as np
```

```
# Define prediction function
```

```
def predict_fraud(*inputs):
```

```
    inputs_array = np.array(inputs).reshape(1, -1)
```

```
    return "Fraud" if model.predict(inputs_array)[0] == 1 else
```

```
"Legit"
```

```
# Gradio Interface
```

```
feature_labels = X.columns.tolist()  
  
gr.Interface(  
    fn=predict_fraud,  
    inputs=[gr.Number(label=col) for col in feature_labels],  
    outputs="text",  
    title="Credit Card Fraud Detection",  
    description="Enter transaction details to predict whether  
    it's Fraud or Legit."  
)
```

## 14. Future scope

- Integrate real-time API to stream live transaction data
- Expand model to include user behavior profiling
- Implement federated learning for cross-bank collaboration

## 15. Team Members and Roles

NAME	ROLE	RESPONSIBLE
Inbarasu I	Member	Data Collection, Data Preprocessing
Deepak kumar S	Member	Feature Engineering
Hemanth D	Member	Exploratory Data Analysis (EDA),
Gokul S	Leader	Model Building, Model Evaluation

## GitHub Screenshot

## Google Colab Link

[https://colab.research.google.com/drive/12bWSnPc0aQhc5KMmjaXJJ\\_\\_8KIZpU8Kf?usp=sharing](https://colab.research.google.com/drive/12bWSnPc0aQhc5KMmjaXJJ__8KIZpU8Kf?usp=sharing)