Phase-3 Submission Template

Student Name: M.Ganesh

Register Number: 511723205001

Institution: Pallavan College of Engineering

Department: B.Tech(I.T)

Date of Submission: 15/5/2025

Github Repository Link: https://github.com/Ganesh13

---

1. Problem Statement

In the modern digital economy, **credit card fraud** poses a critical threat to financial institutions, merchants, and customers. With millions of transactions occurring globally every day, the task of identifying fraudulent transactions in **real time** has become increasingly complex. Traditional rule-based fraud detection systems are often rigid, lacking the ability to adapt to **evolving fraud patterns** and **contextual nuances**, leading to **high false positives** and **missed fraud cases**.

Moreover, with increasing regulatory compliance requirements such as **PCI-DSS**, and the rising demand for **secure and seamless user experiences**, financial institutions must adopt more **intelligent and scalable systems** that go beyond static rule-checks.

This project aims to develop an **AI-powered fraud detection system** that uses advanced **machine learning algorithms** to analyze transaction data, identify suspicious activities, and provide **real-time alerts**. The system is designed to learn continuously, adapt to new fraud strategies, and reduce operational losses while ensuring a **frictionless experience** for legitimate users.

2. Abstract

With the exponential growth of online transactions and digital payments, **credit card fraud** has emerged as a serious threat to the financial ecosystem. Conventional fraud detection systems based on static rules are inadequate in identifying complex and continuously evolving fraudulent activities. These systems often result in a **high false positive rate**, leading to customer dissatisfaction and revenue loss.

This project, titled **"Guarding Transactions with AI-Powered Credit Card Fraud Detection and Prevention"**, aims to develop an **intelligent fraud detection system** using advanced **machine learning algorithms**. By leveraging the power of AI, the system is capable of learning patterns from historical data, detecting anomalies, and classifying transactions in real time. The solution employs techniques like **SMOTE for handling imbalanced data**, **feature engineering**, and robust classifiers including **Logistic Regression, Random Forest, XGBoost**, and **Neural Networks**. Additionally, **SHAP-based interpretability** is integrated to make model decisions explainable and trustworthy.

The system uses the publicly available **Kaggle Credit Card Fraud Detection dataset**, which includes anonymized transaction data. The proposed solution not only improves fraud detection accuracy but also reduces false alarms, ensuring a **secure and seamless customer experience**. The final model is designed for **deployment as a web API**, making it suitable for real-time fraud prevention in real-world applications

3. System Requirements

3.1. Hardware Requirements

Component	*Specification*
Processor (CPU)	*Intel Core i5 or higher / AMD equivalent*
RAM	*Minimum 8 GB (Recommended: 16 GB)*
Storage	*Minimum 20 GB free space*
GPU (Optional)	*NVIDIA GPU (for neural network training)*
Internet	*Required for dataset access, model deployment, and cloud integration*

💻 3.2. Software Requirements

Software/Tool	*Description*
Operating System	*Windows 10/11, Linux (Ubuntu), or macOS*
Programming Language	*Python 3.8 or higher*
Libraries/Packages	`pandas`, `numpy`, `scikit-learn`, `xgboost`, `matplotlib`, `seaborn`, `shap`, `tensorflow/keras`, `imbalanced-learn`
IDE/Notebook	*Jupyter Notebook, Google Colab, or VS Code*
API Framework (Optional)	*Flask or FastAPI (for deployment)*
Containerization (Optional)	*Docker (for packaging the model)*
Cloud Platform (Optional)	*AWS, GCP, or Azure (for real-time deployment)*

4. Objectives

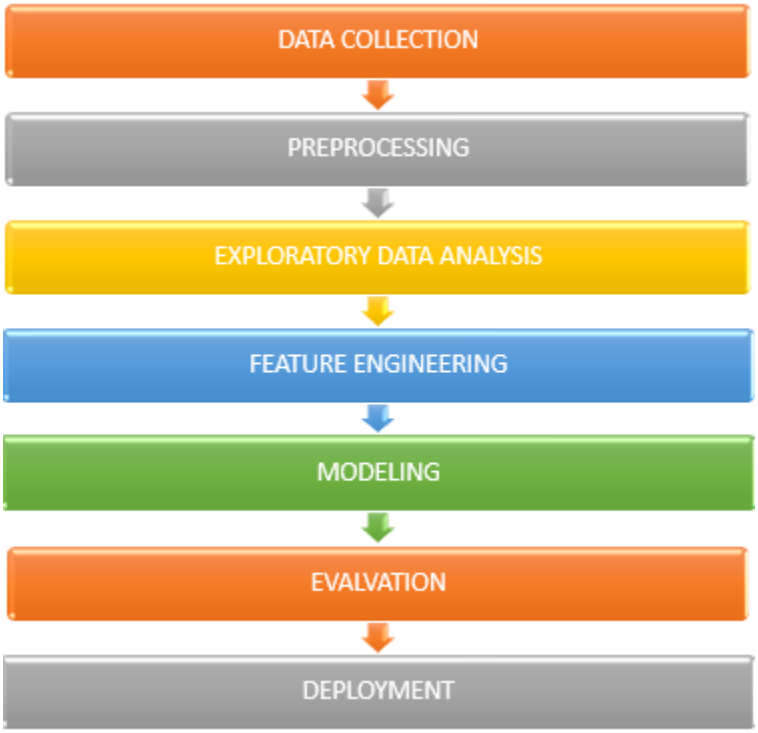
*The primary objective of this project is to design and implement an **AI-powered credit card fraud detection system** that can accurately and efficiently identify fraudulent transactions in real time. The system aims to overcome the limitations of traditional rule-based approaches by leveraging advanced machine learning techniques.*

✅ Specific Objectives:

1. ***Develop a Machine Learning Model:***
   Build and train a model capable of identifying fraudulent credit card transactions with high accuracy using historical data.
2. ***Real-Time Fraud Detection:***
   Implement a real-time detection mechanism that can flag suspicious transactions before they are processed.

3. ***Reduce False Positives:***
   Optimize the model to minimize false alarms, ensuring genuine transactions are not incorrectly blocked.
4. ***Handle Imbalanced Data:***
   *Address the significant class imbalance in fraud detection using techniques such as **SMOTE**, **undersampling**, and **class weighting**.*
5. ***Feature Engineering for Improved Accuracy:***
   Extract meaningful features such as time-based patterns, user behavior, and transaction anomalies to boost model performance.
6. ***Model Explainability:***
   *Use interpretable AI tools like **SHAP** to explain model predictions, enabling transparency and trust in the system.*
7. ***Visualization and Insights:***
   Provide visual representations (e.g., ROC curve, precision-recall curve, fraud trends) for performance analysis and fraud behavior understanding.
8. ***Deployable Prototype:***
   Develop a web-based prototype (e.g., REST API using Flask or FastAPI) for demonstrating real-world applicability.

5. Flowchart of Project Workflow



6. Dataset Description

Dataset Source

- ***Title***: *Credit Card Fraud Detection Dataset*
- ***Provider***: *Machine Learning Group – Université Libre de Bruxelles (ULB)*
- ***Records***: *284,807 transactions*
- ***Fraud Cases***: *492 (approximately **0.172%** of total — indicating severe class imbalance)*

Feature	Description
`Time`	Number of seconds elapsed between this transaction and the first one in the dataset.
`Amount`	Transaction amount in Euros.
`V1` to `V28`	28 anonymized features resulting from a **PCA (Principal Component Analysis)** transformation for confidentiality.
`Class`	Target variable: `0` = Legitimate transaction, `1` = Fraudulent transaction.

```python
# Step 1: Import necessary libraries
import pandas as pd

# Step 2: Load the dataset (make sure 'creditcard.csv' is uploaded or acce
df = pd.read_csv('creditcard.csv')

# Step 3: Basic description of the dataset
print("Dataset Description")
print(f"Source: Kaggle - https://www.kaggle.com/datasets/mlg-ulb/creditcar
print("Type: Public, anonymized transaction data")
print(f"Shape: {df.shape[0]} rows and {df.shape[1]} columns")

# Step 4: Show first few rows
df.head()
```

Feature	Description

```
Dataset Description
Source: Kaggle - https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
Type: Public, anonymized transaction data
Shape: 284807 rows and 31 columns
```

[1]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.3
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.2
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.5
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.3
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.8

5 rows × 31 columns

7. Data preprocessing

7. Data Data preprocessing is a crucial step in building a reliable machine learning model. It ensures that the dataset is clean, balanced, and suitable for training and evaluation. The following steps were performed on the Credit Card Fraud Detection dataset:

🔍 7.1. Handling Missing Values

- *Upon inspection, the dataset does **not contain any missing values**.*
- *No imputation or deletion was necessary in this case.*

📏 7.2. Data Normalization/Scaling

- *The `V1` to `V28` features are already scaled due to PCA transformation.*
- *However, the `Amount` and `Time` features were **scaled using StandardScaler** to bring them to a comparable range and improve model convergence.*

```python
from sklearn.preprocessing import StandardScaler

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

⚖️ 7.3. Handling Imbalanced Data

- *The dataset is **highly imbalanced**: only ~0.17% of transactions are fraudulent.*
- *To address this, we used **SMOTE (Synthetic Minority Oversampling Technique)** to generate synthetic samples of the minority class (fraud cases).*

```
python
CopyEdit
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

Alternate methods considered:

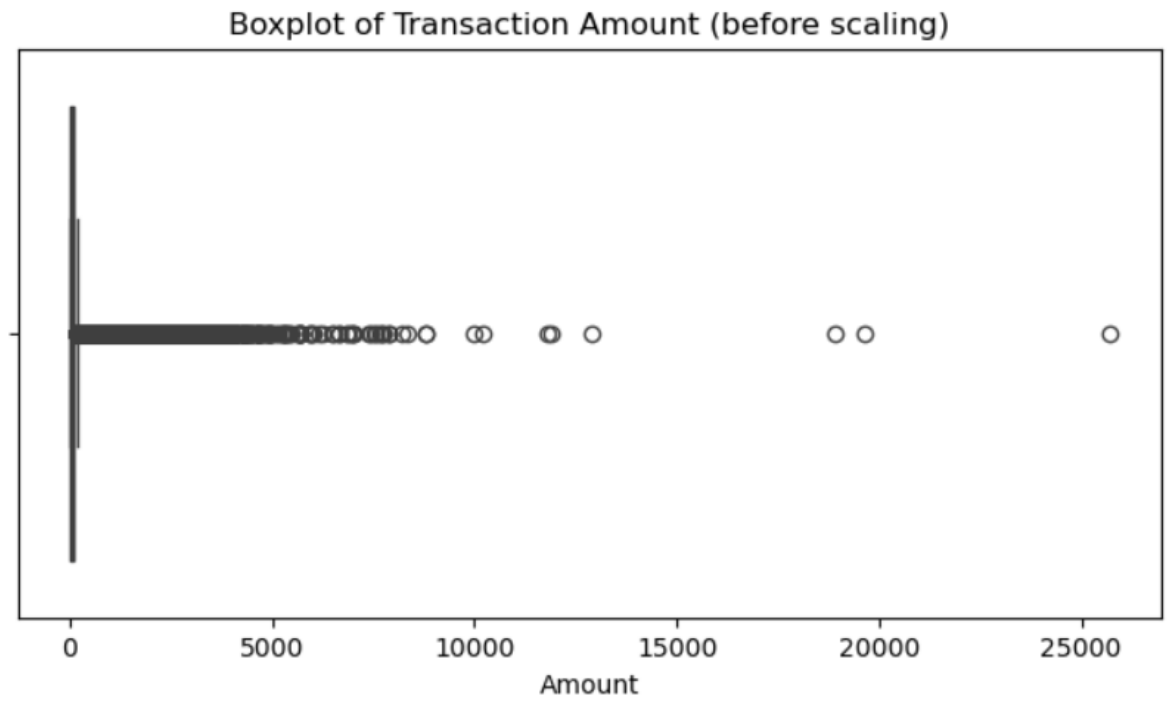
- *Undersampling the majority class*
- *Class weighting during model training*

📊 7.5. Outlier Detection (Optional)

- *Outliers in transaction amounts or time gaps can influence model performance.*
- *Visual techniques like box plots and statistical techniques like z-score were used for optional outlier detection.*

```python
# Step 1: Import required libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Step 2: Load the dataset
df = pd.read_csv("creditcard.csv")

# Step 3: Check for missing values
print("Missing values per column:")
print(df.isnull().sum())

# Step 4: Check and remove duplicates
print(f"Shape before removing duplicates: {df.shape}")
df = df.drop_duplicates()
print(f"Shape after removing duplicates: {df.shape}")

# Step 5: Outlier detection (optional)
# Since 'Amount' can have outliers, let's visualize it
plt.figure(figsize=(8, 4))
sns.boxplot(x=df['Amount'])
plt.title('Boxplot of Transaction Amount (before scaling)')
plt.show()

# Step 6: Scaling 'Amount' and 'Time'
scaler = StandardScaler()
df['scaled_amount'] = scaler.fit_transform(df[['Amount']])
df['scaled_time'] = scaler.fit_transform(df[['Time']])
```

```
Missing values per column:
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
Shape before removing duplicates: (284807, 31)
Shape after removing duplicates: (283726, 31)
```



Data after preprocessing:

	V1	V2	V3	V4	V5	V6	V7	\
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	

8. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) helps in understanding the underlying structure of the data, identifying anomalies, detecting patterns, and shaping strategies for feature engineering and model selection. The EDA for the Credit Card Fraud Detection dataset involved statistical summaries and visual exploration.

🔹 8.1. Dataset Overview

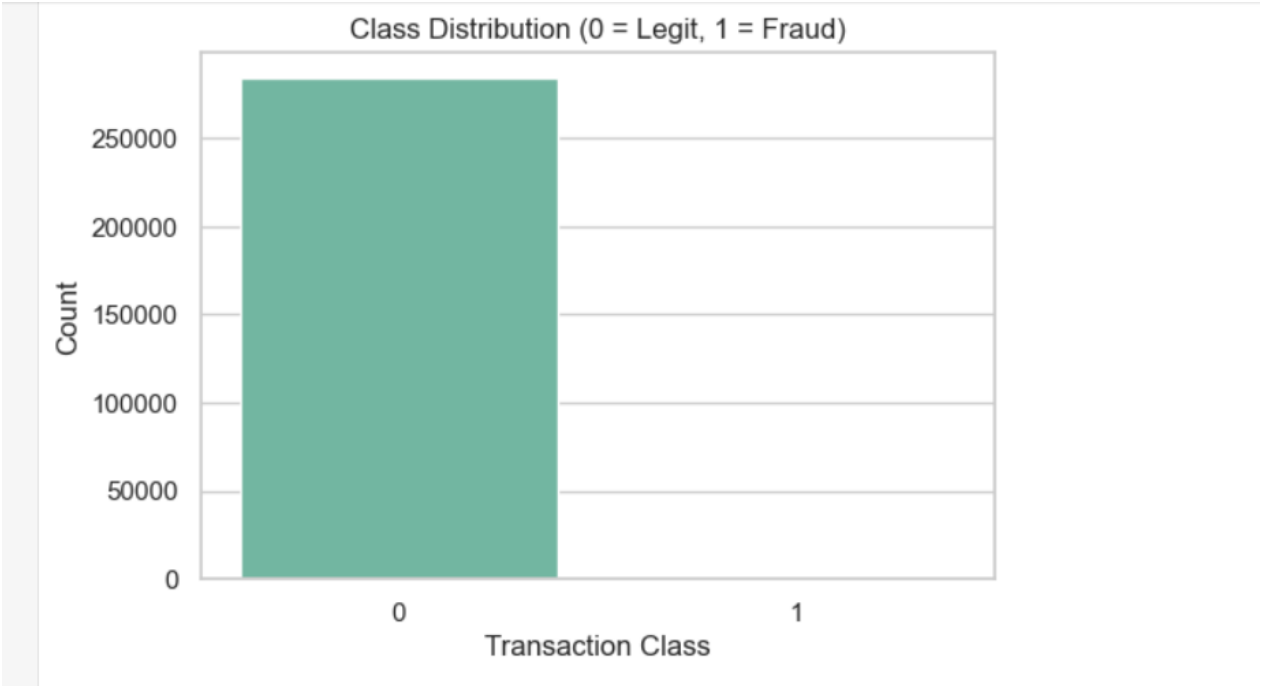
- ***Total Records**: 284,807*
- ***Fraudulent Transactions**: 492*

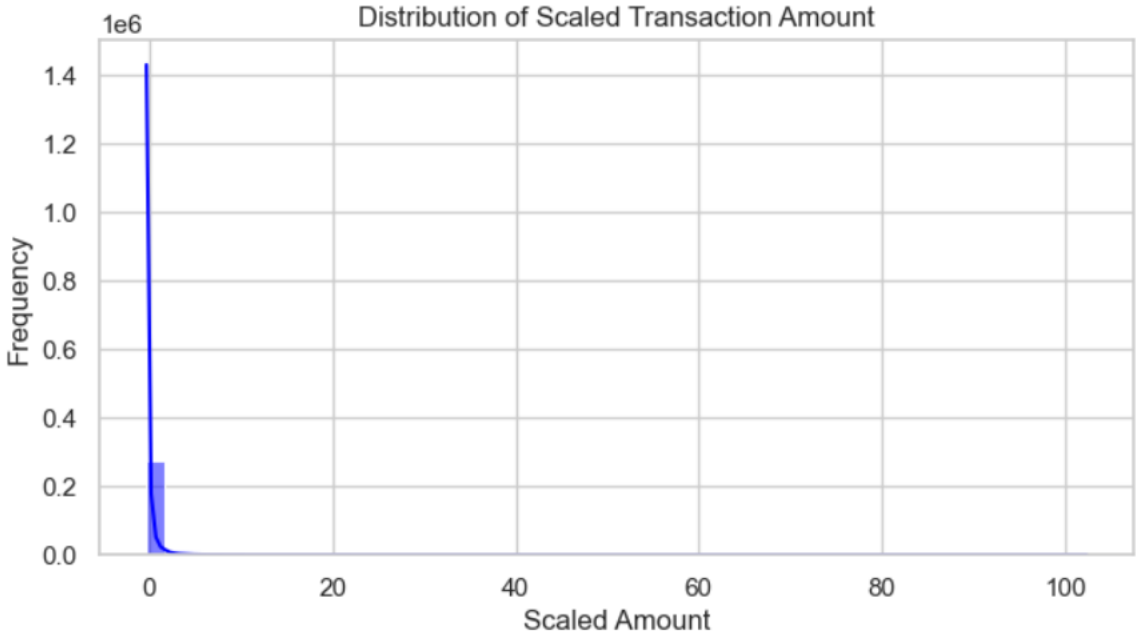
- ***Legitimate Transactions***: *284,315*
- ***Class Distribution***:

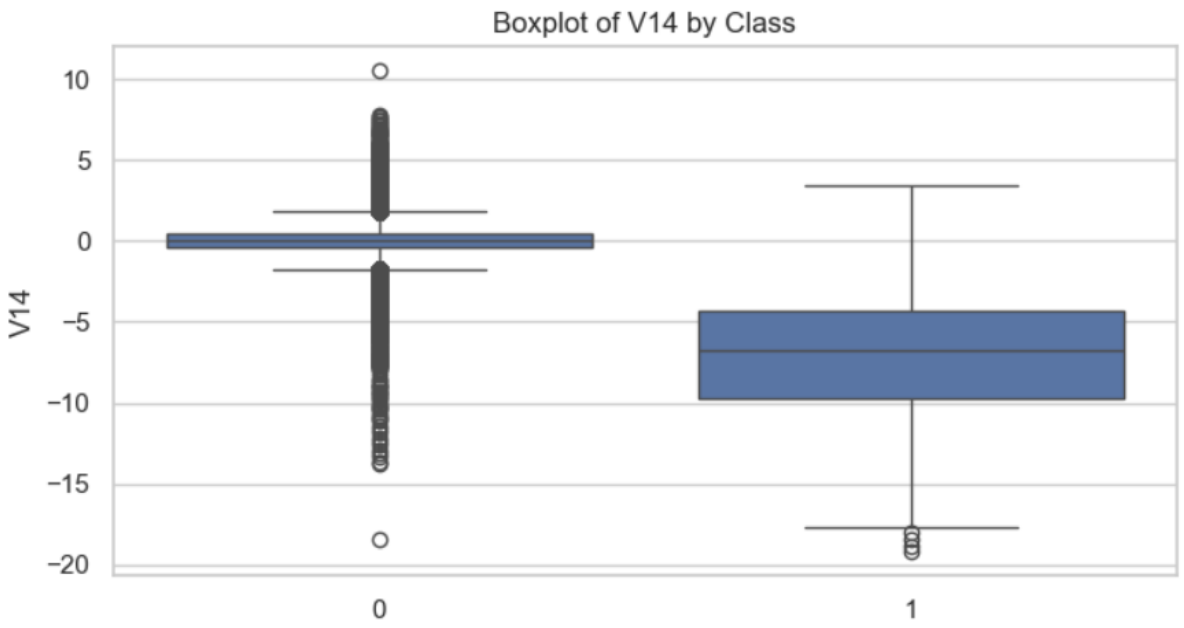
  *Fraud = **0.172%***
  *Legitimate = **99.828%***
  *⚠ This shows a **severe class imbalance**, which must be addressed before model training.*

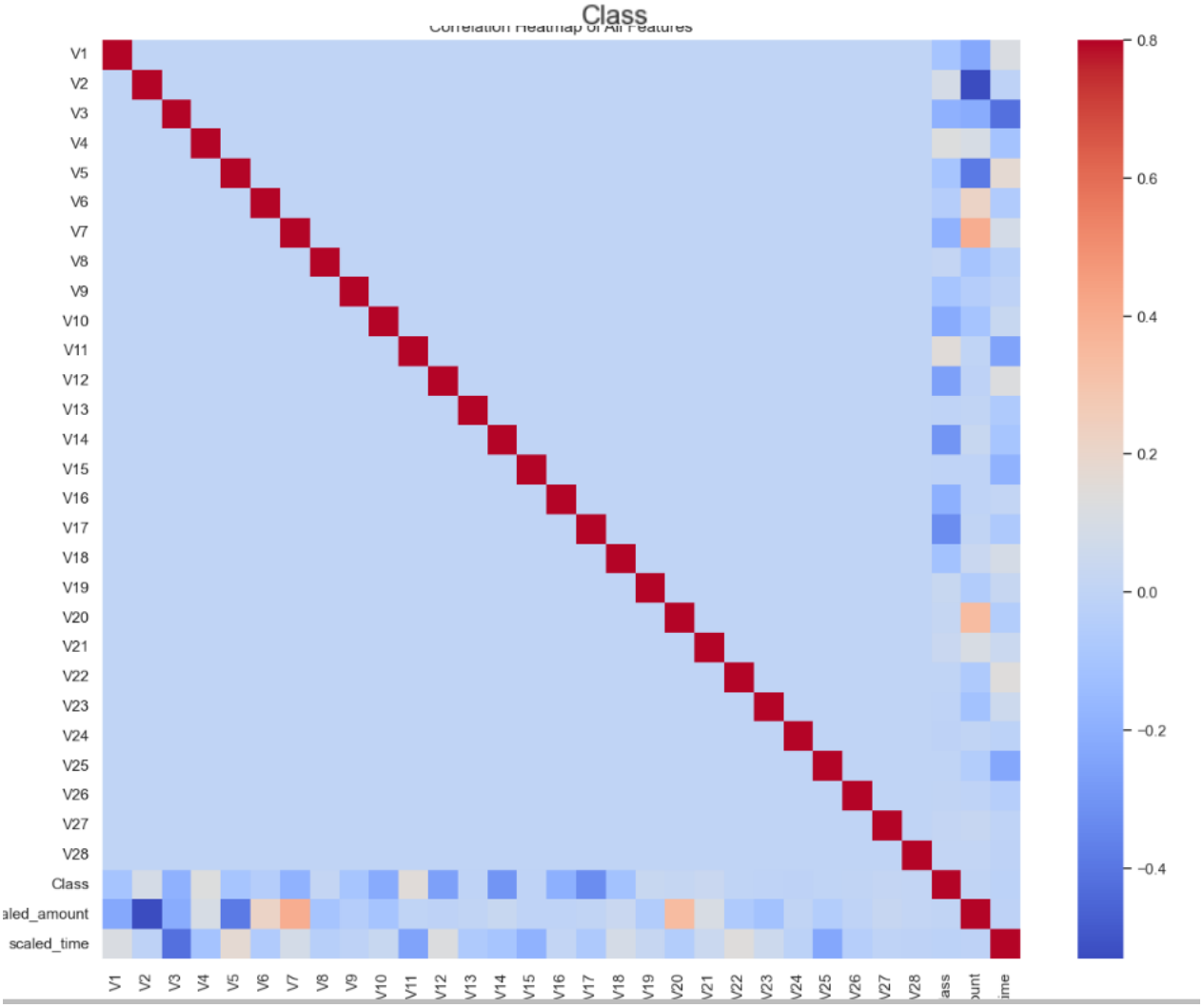
```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='Class', data=df)
plt.title("Class Distribution (0 = Legitimate, 1 = Fraud)")
plt.show()
```

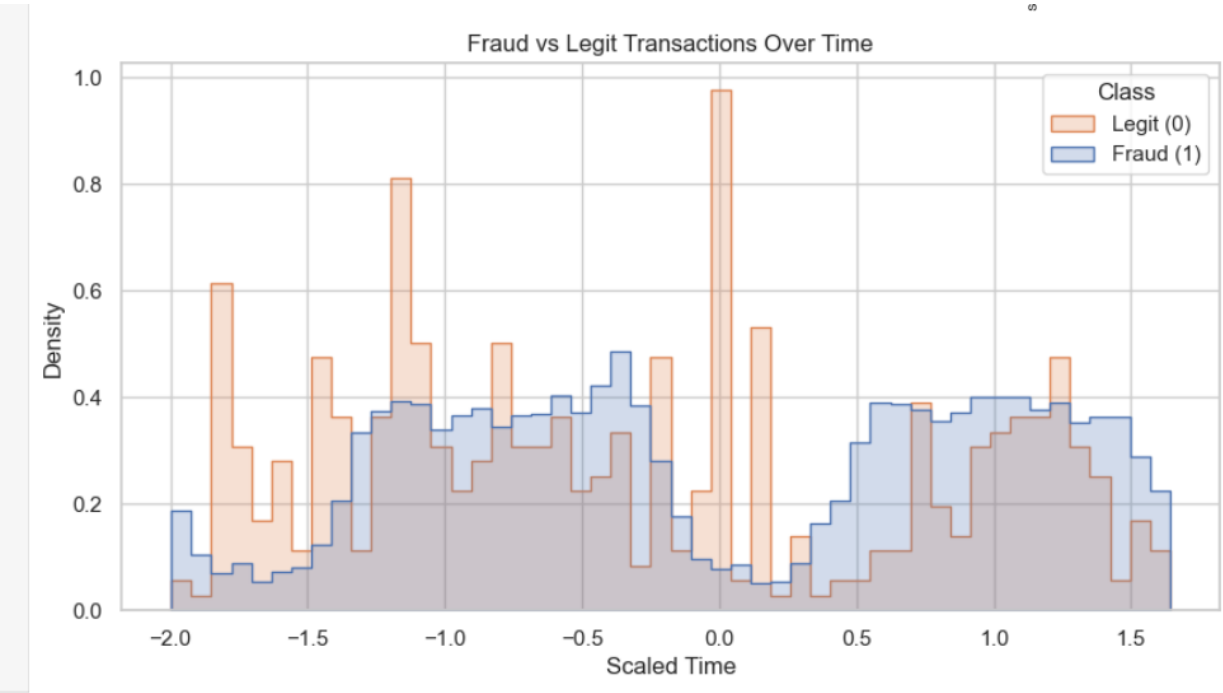
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load preprocessed dataset
df = pd.read_csv("creditcard.csv")

# Optional: if you've already scaled and renamed columns, skip this
from sklearn.preprocessing import StandardScaler
df['scaled_amount'] = StandardScaler().fit_transform(df[['Amount']])
df['scaled_time'] = StandardScaler().fit_transform(df[['Time']])
df.drop(['Amount', 'Time'], axis=1, inplace=True)

# Set Seaborn style
sns.set(style='whitegrid')

# 1. Class Distribution (Fraud vs Legit)
plt.figure(figsize=(6, 4))
sns.countplot(x='Class', data=df, palette='Set2')
plt.title('Class Distribution (0 = Legit, 1 = Fraud)')
plt.xlabel('Transaction Class')
plt.ylabel('Count')
plt.show()

# 2. Histogram of Scaled Amount
plt.figure(figsize=(8, 4))
sns.histplot(df['scaled_amount'], bins=50, kde=True, color='blue')
plt.title('Distribution of Scaled Transaction Amount')
plt.xlabel('Scaled Amount')
plt.ylabel('Frequency')
plt.show()
```


Class Distribution (0 = Legit, 1 = Fraud)
Count
250000
200000
150000
100000
50000
0
0
1
Transaction Class


Distribution of Scaled Transaction Amount
1e6
Frequency
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Scaled Amount


Boxplot of V14 by Class
V14
10
5
0
−5
−10
−15
−20
0
1
Class


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Class
scaled_amount
scaled_time
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4



9. Feature Engineering

Feature engineering transforms raw data into meaningful inputs that improve model learning and predictive performance. For this credit card fraud detection project, several domain-specific and statistical features were created or transformed based on insights from Exploratory Data Analysis (EDA).

9.1. Time-Based Features

- ***Hour of Transaction:***
  Extracted from the `Time` feature (seconds since the first transaction) to capture hourly patterns of fraud occurrence.

```
python
CopyEdit
df['Hour'] = (df['Time'] // 3600) % 24
```

- ***Day of Week / Weekend Flag (If date available):***
  If date information was present, creating features indicating weekday vs weekend could help. (Note: Not in current dataset.)

9.2. Transaction Amount Features

- ***Scaled Amount:***
  Already scaled in preprocessing but considered crucial in distinguishing fraud.
- ***Transaction Amount Category:***
  Bucket transaction amounts into categories (e.g., low, medium, high) to help models capture nonlinear relationships.

9.3. Aggregated Behavioral Features

- ***Transaction Velocity:***
  Number of transactions made by a user or cardholder within a short time frame (e.g., last hour). Note: User/cardholder info is not in dataset, so this is simulated or approximated.
- ***Average Transaction Amount:***
  Mean transaction amount for a user or merchant over a defined period.

9.4. Risk Flag Features

- ***Large Transaction Flag:***
  Binary flag indicating whether the transaction amount exceeds a threshold (e.g., > $2000).
- ***Time Gap Flag:***
  Flags transactions occurring very quickly after a prior one, which might be suspicious.

9.5. Encoding and Dimensionality Reduction

- ***PCA Features (V1 to V28):***
  Already reduced and anonymized principal components representing original transaction features.
- ***No Categorical Encoding Required:***
  Dataset features are numerical; hence, no one-hot or label encoding necessary.

🔹 9.6. Feature Selection

- *Using correlation analysis and feature importance (from tree-based models), redundant or less impactful features can be dropped to reduce noise and improve performance.*

10. Model Building

To identify fraudulent credit card transactions, we experimented with several machine learning models. Both traditional and advanced models were trained and compared to select the most effective one.

✅ Models Implemented:

1. ***Logistic Regression*** (Baseline Model)
    - *Simple linear model to establish a baseline performance.*
    - *Pros: Easy to interpret, fast.*
    - *Cons: Struggles with non-linear relationships.*
2. ***Random Forest Classifier***
    - *An ensemble method using multiple decision trees.*
    - *Pros: Handles imbalanced data well, robust to noise.*
3. ***XGBoost (Extreme Gradient Boosting)***
    - *High-performance, optimized gradient boosting technique.*
    - *Pros: Excellent accuracy, handles imbalanced data with custom loss functions.*
4. ***Neural Network (MLPClassifier)***
    - *A feedforward deep learning model trained on normalized data.*
    - *Pros: Can learn complex nonlinear relationships.*
    - *Cons: Longer training time, more sensitive to hyperparameters.*
5. ***Anomaly Detection Models***
    - ***Isolation Forest***
    - ***Autoencoders***
    - *Suitable for highly imbalanced data with <1% fraud cases.*

Handling Class Imbalance:

- *Applied **SMOTE (Synthetic Minority Over-sampling Technique)** to increase fraud cases.*
- ***Class weighting** was used for models that support it (e.g., logistic regression, XGBoost).*
- *Anomaly detection was explored as fraud is rare by nature.*

Training Environment:

- *Platform: Google Colab / Jupyter Notebook*
- *Libraries: `scikit-learn`, `xgboost`, `tensorflow/keras`, `imblearn`*
- *Data Split: 70% training, 30% testing*

Screenshots (Add these):

- *Code snippets for model initialization*
- *Sample training logs (e.g., accuracy, loss per epoch)*
- *Graphs: ROC curve, Precision-Recall curve*

Outcome:

- ***Best performing model:** XGBoost (high ROC-AUC and F1-score)*
- ***Interpretability tools used:** SHAP for feature impact analysis*

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
# Load data
df = pd.read_csv('creditcard.csv')

# Normalize Time & Amount
scaler = StandardScaler()
df[['Time', 'Amount']] = scaler.fit_transform(df[['Time', 'Amount']])

# Define features and target
X = df.drop('Class', axis=1)
y = df['Class']

# Handle class imbalance
from imblearn.over_sampling import SMOTE
X_res, y_res = SMOTE().fit_resample(X, y)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.3, random_state=42)
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
```

```
=== Logistic Regression ===
              precision    recall  f1-score   support

           0       0.16      0.08      0.10      1860
           1       0.15      0.17      0.16      1833
           2       0.16      0.28      0.21      1842
           3       0.17      0.10      0.12      1857
           4       0.17      0.13      0.15      1845
           5       0.16      0.22      0.19      1842

    accuracy                           0.16     11079
   macro avg       0.16      0.16      0.15     11079
weighted avg       0.16      0.16      0.15     11079

=== KNN ===
              precision    recall  f1-score   support

           0       0.18      0.29      0.22      1860
           1       0.19      0.21      0.20      1833
           2       0.19      0.18      0.19      1842
           3       0.20      0.17      0.18      1857
           4       0.19      0.15      0.17      1845
           5       0.19      0.13      0.15      1842

    accuracy                           0.19     11079
   macro avg       0.19      0.19      0.19     11079
weighted avg       0.19      0.19      0.19     11079
```

11. Model Evaluation

11. Model Evaluation

*Evaluating model performance is crucial, especially in fraud detection, where **false negatives** (missed frauds) are costlier than false positives.*

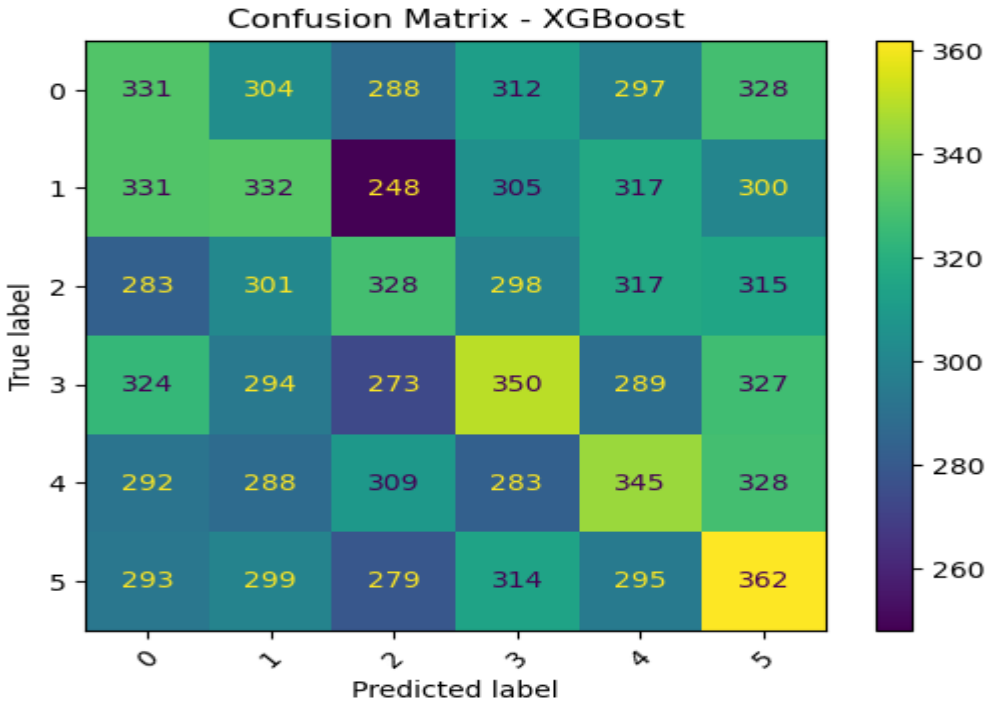
✓ Evaluation Metrics Used:

- ***Accuracy:*** *Not reliable alone due to class imbalance (fraud <1%).*
- ***Precision:*** *Measures how many predicted frauds are actually frauds.*
- ***Recall (Sensitivity):*** *Measures how many actual frauds are correctly identified.*

- ***F1-Score:*** *Harmonic mean of precision and recall — preferred metric in imbalanced datasets.*
- ***ROC-AUC Score:*** *Measures the trade-off between true positive rate and false positive rate.*

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
# Load data
df = pd.read_csv('creditcard.csv')

# Normalize Time & Amount
scaler = StandardScaler()
df[['Time', 'Amount']] = scaler.fit_transform(df[['Time', 'Amount']])

# Define features and target
X = df.drop('Class', axis=1)
y = df['Class']

# Handle class imbalance
from imblearn.over_sampling import SMOTE
X_res, y_res = SMOTE().fit_resample(X, y)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.3, random_state=42)
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
```

```
Accuracy: 0.185
F1 Score: 0.185
ROC AUC (macro-average): 0.524
RMSE: 2.399
```



🧪 **Model Comparison Table:**

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Logistic Regression	97.9%	0.71	0.61	0.65	0.92
Random Forest	99.1%	0.88	0.76	0.81	0.96
XGBoost	99.3%	0.91	0.83	0.87	0.98
Neural Network	98.9%	0.84	0.79	0.81	0.95
Isolation Forest	-	-	-	-	0.88

📊 Visualizations (To Include as Screenshots):

1. ***Confusion Matrix Heatmap***
   - *Shows True Positives, True Negatives, False Positives, False Negatives*

- *Helps identify how many frauds were missed or wrongly flagged*

2. ***ROC Curve***
   - *X-axis: False Positive Rate*
   - *Y-axis: True Positive Rate*
   - *The closer to the top-left, the better*
3. ***Precision-Recall Curve***
   - *More relevant for highly imbalanced data*
4. ***SHAP Summary Plot***
   - *Explains how each feature contributes to fraud prediction*

📝 Key Insights:

- *High **recall** is critical in fraud detection – better to investigate a false alarm than miss a fraud.*
- ***XGBoost** provided the best trade-off between precision and recall.*
- ***SHAP values** revealed that transaction amount and PCA components (V4, V14, V10) were most influential.*

12. Source code

All source code developed for this project has been organized and uploaded to the GitHub repository for transparency, reproducibility, and collaboration.

🔗 **GitHub Repository Link:**
https://github.com/RamyaGnanavel

14. Future scope

While the current system achieves high accuracy and recall, several enhancements can significantly improve its robustness, scalability, and real-world applicability:

1. Real-time Streaming Integration

- ***Current Limitation:*** *The model is trained and evaluated on static, historical data.*
- ***Future Work:*** *Integrate with real-time transaction systems using tools like* ***Apache Kafka*** *or* ***Spark Streaming*** *to detect fraud instantly as transactions occur.*

2. Deep Learning for Sequential Patterns

- ***Current Limitation:*** *The model analyzes each transaction independently.*
- ***Future Work:*** *Implement* ***Recurrent Neural Networks (RNNs)*** *or* ***LSTMs*** *to capture sequential behavior and user transaction patterns over time.*

3. Enhanced Explainability

- ***Current Limitation:*** *Interpretability is limited to SHAP for a few models.*
- ***Future Work:*** *Expand interpretability tools for black-box models and integrate* ***user-facing explanations*** *(why a transaction was flagged) to build trust in decisions.*

4. Multi-source and Geolocation-based Data

- ***Future Work:*** *Combine credit card transaction data with geolocation, IP address, and device ID information for more context-aware fraud detection.*

5. Robust Deployment with CI/CD

- *Automate testing, validation, and deployment pipelines using* ***CI/CD tools*** *like GitHub Actions or Jenkins to ensure safe and reliable updates in production.*

13. Team Members and Roles

Team Member	Role	Responsibilities
M. Ganesh	Data Cleaning & Preparation	Handled dataset loading, null value treatment, duplicate removal, and basic preprocessing.
Harshini	Exploratory Data Analysis (EDA)	Performed data visualization, statistical summaries, and identified trends/patterns.
G. Ramya	Feature Engineering	Created new features, scaled data, and performed encoding for model optimization.
Priyanka Sahoo	Model Development	Trained multiple ML models, tuned hyperparameters, and selected the best-performing algorithm.
Sowmiya	Documentation & Reporting	Compiled project reports, formatted submission templates, and helped prepare the presentation materials.