Phase-3

Student Name: LOKESH J

Register Number: 410723104040

Institution: Dhanalakshmi College of Engineering

Department: Computer Science and Department

Date of Submission:14-05-2025

GitHub Repository Link: https://github.com/Lokesh151120

Guarding transactions with AI-powered credit card fraud detection and prevention

1.Problem Statement

Credit card fraud has become increasingly sophisticated and prevalent, causing significant financial losses to consumers, businesses, and financial institutions. Traditional rule-based fraud detection systems often fail to adapt to evolving fraud tactics, leading to high false positive rates and delayed detection. There is a critical need for a more intelligent, adaptive, and real-time solution to accurately identify and prevent fraudulent transactions without compromising customer experience. This project aims to develop and implement an AI-powered credit card fraud detection and prevention system that leverages machine learning and behavioural analysis to enhance transaction security, minimize financial losses, and ensure seamless user experiences.

2. Abstract

Credit card fraud poses a serious threat to the global financial ecosystem, leading to billions of dollars in losses each year. As fraudulent activities become more complex and adaptive, traditional rule-based fraud detection systems struggle to keep up, often resulting in delayed responses and high false positive rates. This paper presents an AI-powered approach to credit card fraud detection and prevention that leverages machine learning algorithms to analyse transactional data in real time and detect anomalies indicative of fraudulent behaviour. By training models on historical transaction patterns and integrating behavioural analytics, the system aims to accurately identify fraudulent transactions with minimal impact on legitimate users. The proposed solution focuses on improving detection accuracy, reducing response time, and lowering operational costs for financial institutions. Experimental results on benchmark datasets demonstrate the effectiveness of AI in enhancing fraud detection performance, making it a vital tool for securing digital financial transactions in an increasingly interconnected world.

3.System Requirements

1. Hardware Requirements
   - Development Environment:
   - Processor: Intel i5/i7 or AMD Ryzen 5/7 (quad-core or better)
   - RAM: Minimum 16 GB (32 GB recommended for training large datasets)
   - Storage: 512 GB SSD (1 TB recommended)
   - GPU: NVIDIA GPU with CUDA support (e.g., GTX 1660 or higher for deep learning)

- Network: Stable internet connection for cloud access and dataset retrieval
- Production Environment (for real-time deployment):
- Server: Cloud-based (e.g., AWS EC2, Azure, GCP) or on-premises with scalable architecture
- RAM: Minimum 32 GB
- Storage: SSD with redundancy (RAID), scalable based on data volume
- GPU/TPU (Optional): If using deep learning models for real-time scoring
- High Availability: Load balancer, failover support, and auto-scaling for traffic spikes

---

2. Software Requirements

- Operating System:
- Windows 10/11, Linux (Ubuntu 20.04+ preferred), or macOS (for development)
- Linux (Ubuntu Server, CentOS, or Amazon Linux) for deployment
- Programming Languages & Libraries:
- Python 3.8+ (preferred for AI/ML development)
- Key libraries:
- NumPy, Pandas (data manipulation)
- Scikit-learn (ML algorithms)
- XG Boost, Light GBM (advanced models)
- TensorFlow/Kera's or Py Torch (if using deep learning)
- Matplotlib, Seaborn (visualization)
- imbalanced-learn (for handling class imbalance)

Database:

- SQL (PostgreSQL, MySQL) or NoSQL (MongoDB)
- Data warehouse (e.g., Snowflake, Big Query) for large-scale historical data

Development Tools:

- Jupyter Notebook / VS Code / PyCharm
- Git (for version control)
- Docker (for containerization)
- Virtual environments (e.g., vend, Conda)

Deployment/Monitoring Tools:

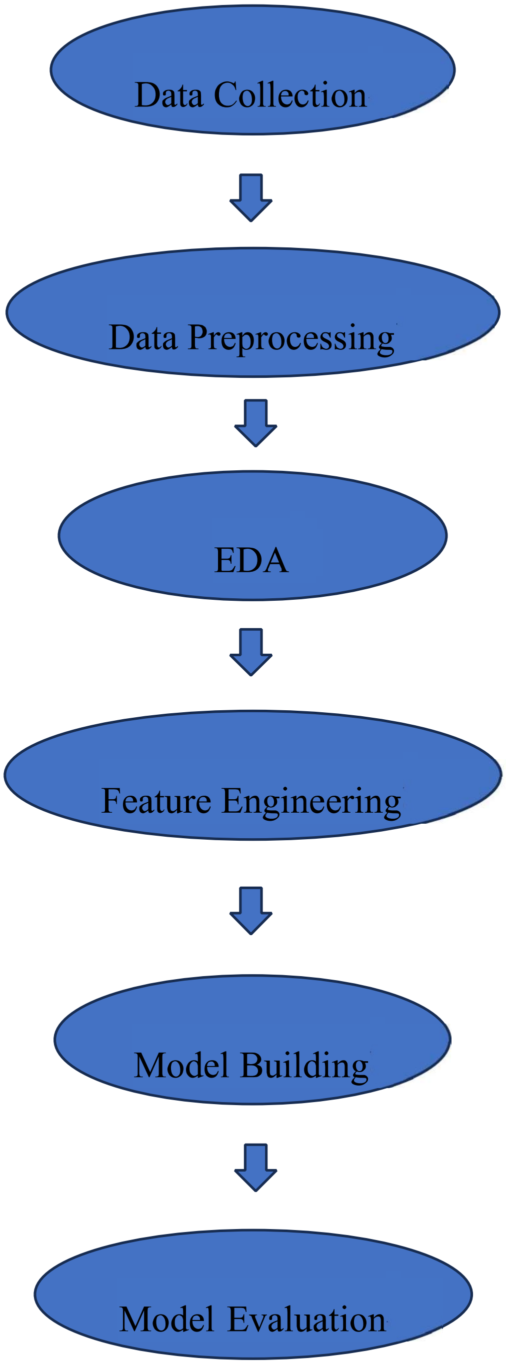
- REST API framework: Flask, Fast API, or Django
- Cloud platform: AWS (S3, Lambda, SageMaker), GCP, or Azure
- Monitoring: Prometheus + Grafana or ELK Stack (Elasticsearch, Logstash, Kibana)
- CI/CD: GitHub Actions, Jenkins, or GitLab CI

4. Objectives

- Detect fraudulent credit card transactions using AI and machine learning.
- Reduce false positives and false negatives for improved accuracy.
- Analyse historical data to identify fraud patterns.
- Enable real-time monitoring of transactions.
- Adapt to evolving fraud techniques through model updates.
- Handle data imbalance effectively.
- Develop a scalable and deployable system.
- Enhance customer trust by securing financial transactions.
- Provide model explainability for transparency and trust.

5.

5. Flowchart of Project Workflow



6. Dataset Description

- Source: Kaggle - Telco
- Customer Churn Type: Public
- Size: 7,043 rows × 21 columns

```
data.head()
```

1 to 5 of 5 entries Filter

index	Amount	MerchantID	TransactionType	Location	IsFraud	TransactionHour	TransactionDay	TransactionMonth
0	4189.27	688	1	7	0	14	3	4
1	2659.71	109	1	1	0	13	19	3
2	784.0	394	0	4	0	10	8	1
3	3514.4	944	0	5	0	23	13	4
4	369.07	475	0	6	0	18	12	7

7. Data Preprocessing

- **Load Data** – Import dataset and check for missing or duplicate values.
- **Clean Data** – Handle anomalies or irrelevant features.
- **Feature Scaling** – Normalize Amount and optionally Time using scalers.
- **Handle Imbalance** – Apply SMOTE, over/under-sampling, or use class weights.
- **Split Data** – Use train_test_split() to create training and testing sets.
- **Model Input** – Prepare the processed data for machine learning models.
- **Evaluation Prep** – Set up metrics like Precision, Recall, F1-Score, and AUC-ROC

data.info ()

7.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   TransactionID    100000 non-null  int64
 1   TransactionDate  100000 non-null  object
 2   Amount           100000 non-null  float64
 3   MerchantID       100000 non-null  int64
 4   TransactionType  100000 non-null  object
 5   Location         100000 non-null  object
 6   IsFraud          100000 non-null  int64
dtypes: float64(1), int64(3), object(3)
memory usage: 5.3+ MB
```

data.describe()

```
data.describe()
```

	TransactionID	Amount	MerchantID	IsFraud
count	100000.000000	100000.000000	100000.000000	100000.000000
mean	50000.500000	2497.092666	501.676070	0.010000
std	28867.657797	1442.415999	288.715868	0.099499
min	1.000000	1.050000	1.000000	0.000000
25%	25000.750000	1247.955000	252.000000	0.000000
50%	50000.500000	2496.500000	503.000000	0.000000
75%	75000.250000	3743.592500	753.000000	0.000000
max	100000.000000	4999.770000	1000.000000	1.000000

df.isnull( ).sum( )

```
data.isnull().sum()
```

	0
TransactionID	0
TransactionDate	0
Amount	0
MerchantID	0
TransactionType	0
Location	0
IsFraud	0

df.drop_duplicates( )

```
data.drop_duplicates()
```

	TransactionID	TransactionDate	Amount	MerchantID	TransactionType	Location	IsFraud
0	1	2024-04-03 14:15:35.462794	4189.27	688	refund	San Antonio	0
1	2	2024-03-19 13:20:35.462824	2659.71	109	refund	Dallas	0
2	3	2024-01-08 10:08:35.462834	784.00	394	purchase	New York	0
3	4	2024-04-13 23:50:35.462850	3514.40	944	purchase	Philadelphia	0
4	5	2024-07-12 18:51:35.462858	369.07	475	purchase	Phoenix	0
...	...	...	...	...	...	...	...
99995	99996	2024-06-07 00:57:36.027591	1057.29	289	refund	San Antonio	0
99996	99997	2023-10-22 23:12:36.027594	297.25	745	refund	San Antonio	0
99997	99998	2024-05-31 19:27:36.027597	3448.56	690	purchase	San Antonio	0
99998	99999	2024-10-18 09:43:36.027601	3750.79	644	purchase	Philadelphia	0
99999	100000	2024-03-05 19:41:36.027606	1596.79	675	refund	Houston	0

100000 rows × 7 columns

df.duplicated( ).sum( )

```
[30] data.duplicated().sum()

np.int64(0)
```

8. Exploratory Data Analysis (EDA)

Dataset Overview

- 284,807 transactions, 31 features
- Highly imbalanced: ~0.17% fraud cases

Class Distribution

- Visualize fraud vs. non-fraud counts (e.g., bar plot)

Feature Correlation

- Check correlation heatmap
- PCA features (V1–V28) mostly uncorrelated

Feature Distributions

- Analyze Amount and Time
- Use histograms, boxplots, and density plots

Outlier Detection

- Use boxplots to identify outliers, especially in fraud cases

Data Visualization

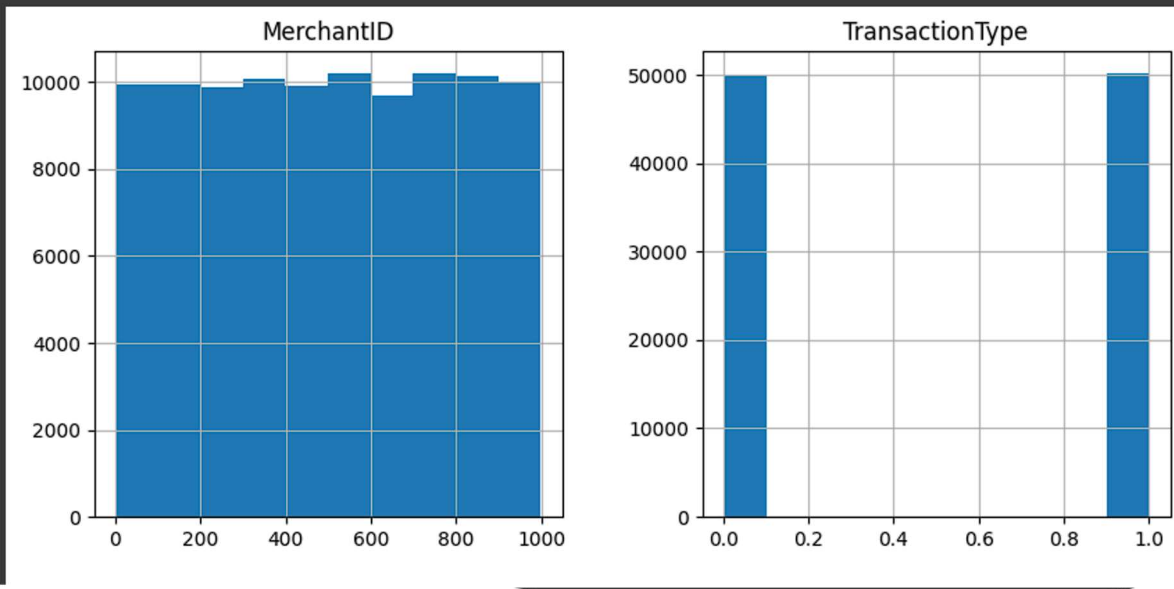
- Optional t-SNE or PCA for 2D class separation

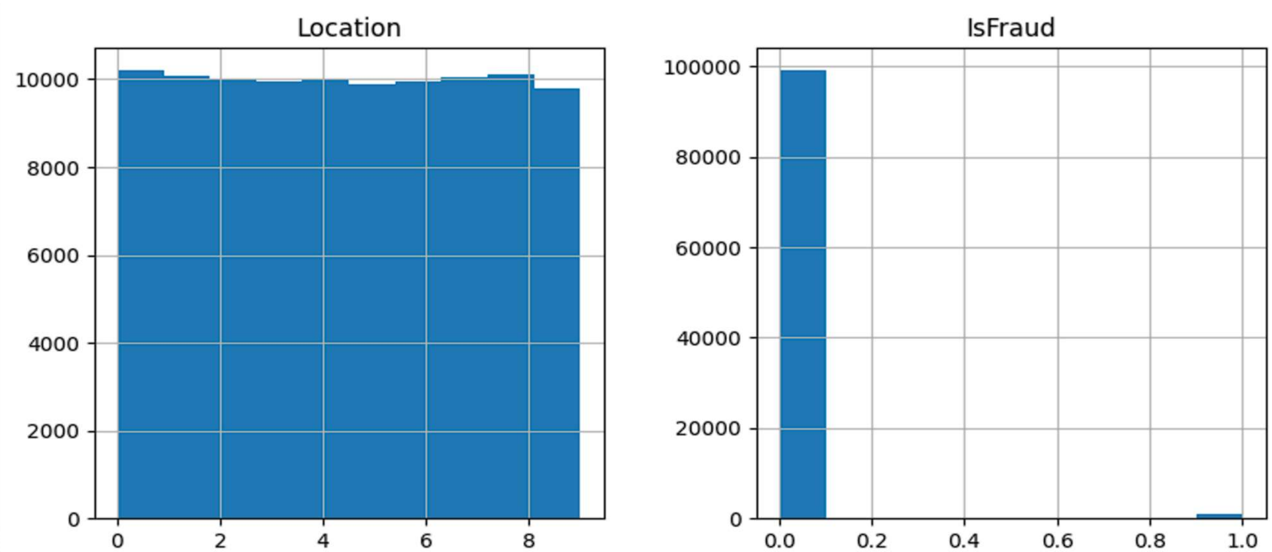
Feature Importance

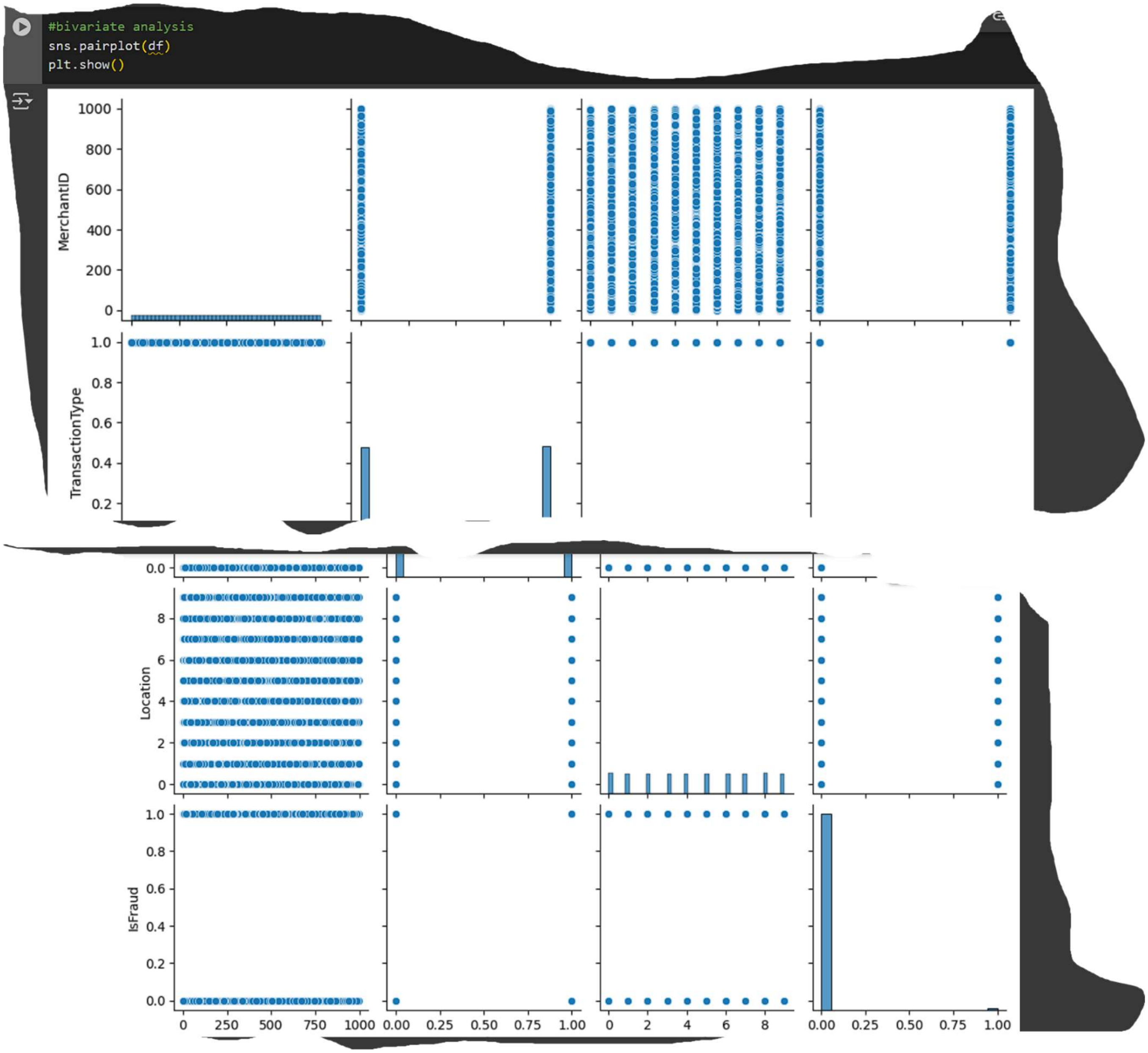
- Use a basic model (e.g., Random Forest) to check important features

10.

```
#histogram chart
df.hist(figsize=(10,10))
plt.show()
```







9. Feature Engineering

Scaling Features

- Normalize Amount and Time using Standard Scaler or MinMaxScaler to improve model performance.

Creating New Features (Optional)

- **Hour of Transaction** from Time to capture time-based fraud patterns.
- **Log-transformed Amount** to reduce skewness.

Handling Class Imbalance

- Use **SMOTE**, **ADASYN**, or **random over/under-sampling** to balance classes.

Dropping Unnecessary Features

- Remove Time if not useful after transformation.
- Keep only informative features based on correlation and importance.

Dimensionality Reduction (Optional)

- Use **PCA** or **feature selection techniques** if needed to reduce noise and improve efficiency.

Feature Importance Analysis

- Use tree-based models (e.g., Random Forest, XG Boost) to identify and keep only the most important features.

```
data
```

	Amount	MerchantID	TransactionType	Location	IsFraud	TransactionHour	TransactionDay	TransactionMonth
0	4189.27	688	1	7	0	14	3	4
1	2659.71	109	1	1	0	13	19	3
2	784.00	394	0	4	0	10	8	1
3	3514.40	944	0	5	0	23	13	4
4	369.07	475	0	6	0	18	12	7
...	...	...	...	...	...	...	...	...
99995	1057.29	289	1	7	0	0	7	6
99996	297.25	745	1	7	0	23	22	10
99997	3448.56	690	0	7	0	19	31	5
99998	3750.79	644	0	5	0	9	18	10
99999	1596.79	675	1	2	0	19	5	3

100000 rows × 8 columns

Scaler standardization

```
#scalar standardization
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)
```

```
df
```

	MerchantID	TransactionType	Location	IsFraud
0	687	1	7	0
1	108	1	1	0
2	393	0	4	0
3	943	0	5	0
4	474	0	6	0
...	...	...	...	...
99995	288	1	7	0
99996	744	1	7	0
99997	689	0	7	0
99998	643	0	5	0
99999	674	1	2	0

100000 rows × 4 columns

10. Model Building

Select Algorithms

- Use classification models suited for imbalanced data:
- **Logistic Regression**
- **Random Forest**
- **XG Boost**
- **Light GBM**

- **Support Vector Machine (SVM)**
- **Neural Networks (optional)**

Train the Model

- Split data into training and testing sets.
- Train the model using the processed and balanced dataset.
- Use class_weight='balanced' for models like Logistic Regression or SVM

Hyperparameter Tuning

- Use **GridSearchCV** or **RandomizedSearchCV** to optimize model parameters

Cross-Validation

- Apply K-Fold cross-validation (e.g., 5-fold) to ensure model stability

Model Evaluation

- Use evaluation metrics:
- **Precision, Recall, F1-Score**
- **Confusion Matrix**
- **ROC-AUC Curve**

Select Best Model

- Choose the model with the best balance of high **recall** (to catch fraud) and low false positives.

```python
X = data.drop('IsFraud', axis=1)
y = data['IsFraud']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)
```

```python
# Print the count before SMOTE
print("Before SMOTE:")
print(f"Train Data - Positive class (Fraud): {y_train.sum()} | Negative class (Non-Fraud): {(y_train == 0).sum()}")
print(f"Test Data - Positive class (Fraud): {y_test.sum()} | Negative class (Non-Fraud): {(y_test == 0).sum()}")
```

```
Before SMOTE:
Train Data - Positive class (Fraud): 787 | Negative class (Non-Fraud): 79213
Test Data - Positive class (Fraud): 213 | Negative class (Non-Fraud): 19787
```

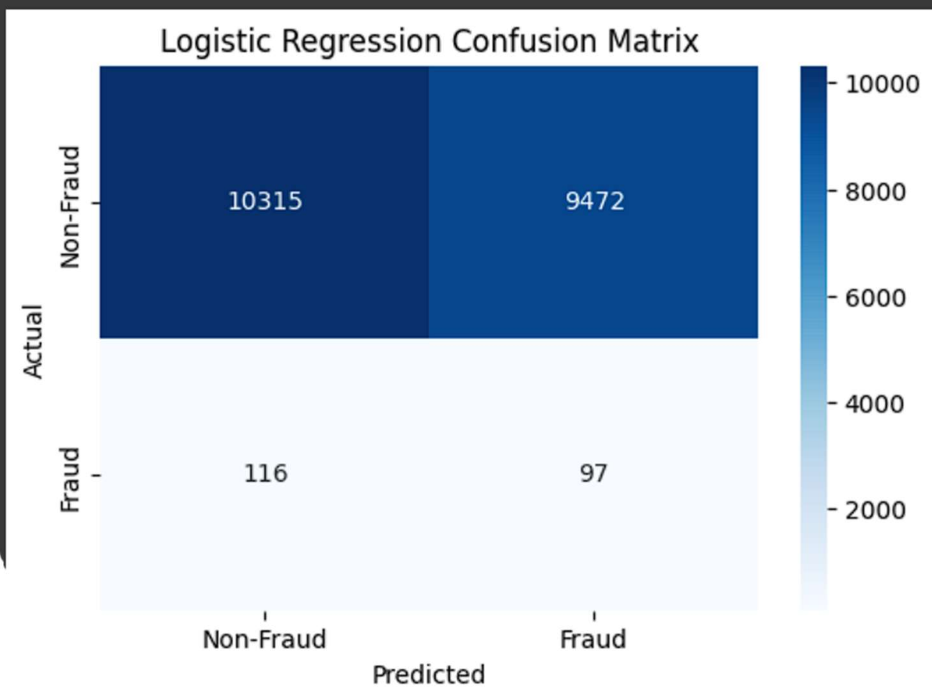
11. Model Evaluation

- **Confusion Matrix** – Shows correct and incorrect predictions
- **Precision** – Accuracy of fraud predictions
- **Recall** – How many frauds were correctly caught
- **F1-Score** – Balance between precision and recall
- **ROC-AUC** – Overall model performance
- **PR-AUC** – Better for imbalanced datasets

```python
# Store results in dictionary
evaluation_results[model_name] = {
    "accuracy": accuracy,
    "classification_report": clf_report,
    "confusion_matrix": conf_matrix
}

# Print evaluation for each model
print(f"\n{model_name} Accuracy: {accuracy}")
print(f"\n{model_name} Classification Report:\n", clf_report)

# Plot confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Non-Fraud', 'Fraud'], yticklabels=['Non-Fraud', 'Fraud'])
plt.title(f'{model_name} Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

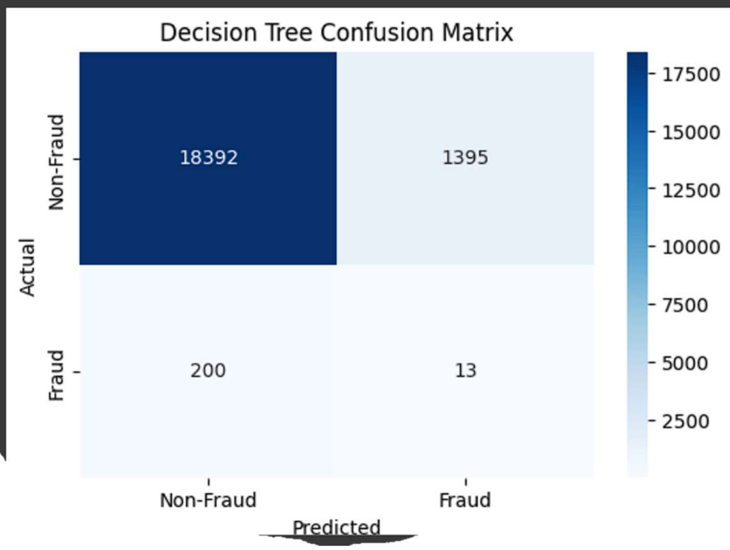
16.

```
           0       0.99      0.52      0.68     19787
           1       0.01      0.46      0.02       213

    accuracy                           0.52     20000
   macro avg       0.50      0.49      0.35     20000
weighted avg       0.98      0.52      0.68     20000
```



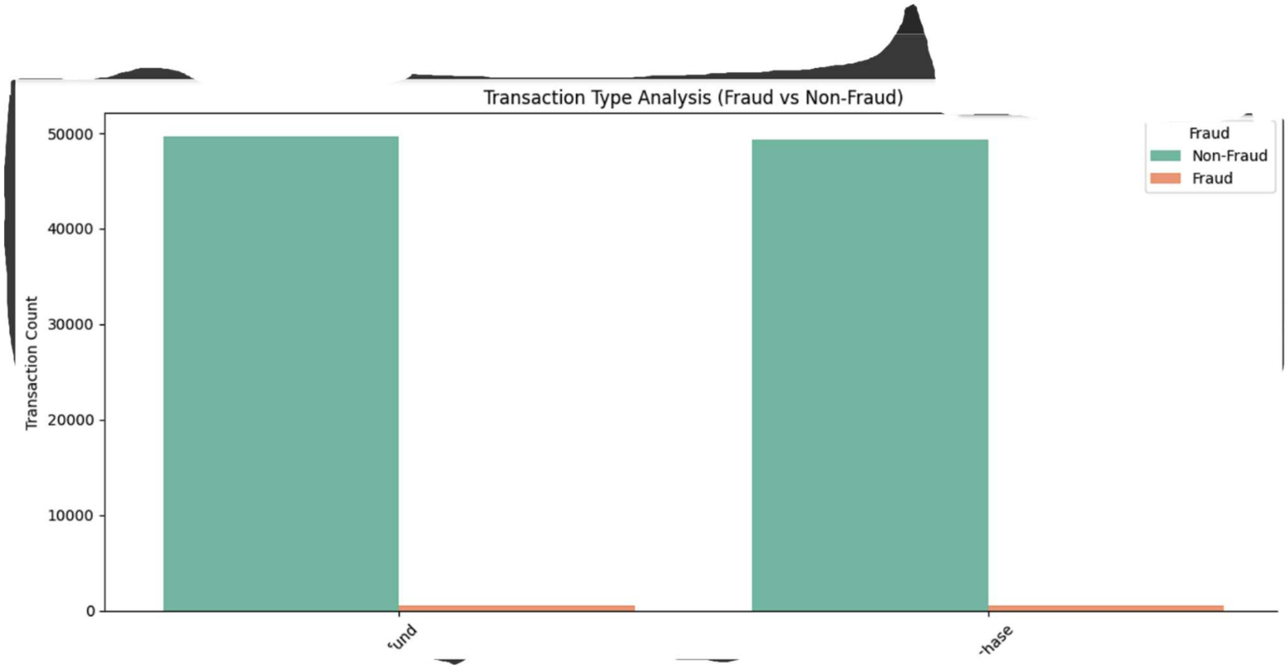
17.

0	0.99	0.93	0.96	19787
1	0.01	0.06	0.02	213
accuracy			0.92	20000
macro avg	0.50	0.50	0.49	20000
weighted avg	0.98	0.92	0.95	20000

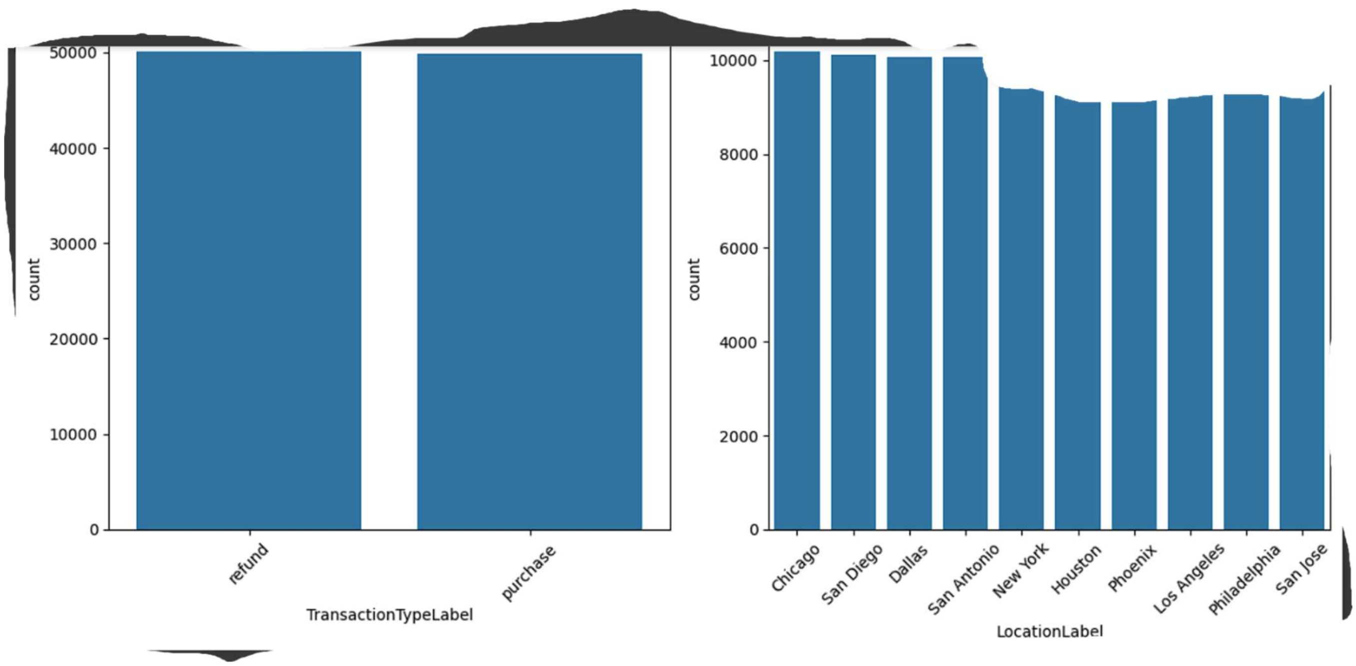



```
plt.figure(figsize=(12, 6))

sns.countplot(x='TransactionTypeLabel', hue='IsFraud', data=data, palette='Set2')
plt.title('Transaction Type Analysis (Fraud vs Non-Fraud)')
plt.xlabel('Transaction Type')
plt.ylabel('Transaction Count')
plt.legend(title='Fraud', loc='upper right', labels=['Non-Fraud', 'Fraud'])
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

18.




```
data['TransactionTypeLabel'] = data['TransactionType'].map(transaction_type_inverse_mapping)
data['LocationLabel'] = data['Location'].map(location_inverse_mapping)

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.countplot(x='TransactionTypeLabel', data=data, order=data['TransactionTypeLabel'].value_counts().index)
plt.title('Transaction Type Distribution')
plt.xticks(rotation=45)

plt.subplot(1, 2, 2)
sns.countplot(x='LocationLabel', data=data, order=data['LocationLabel'].value_counts().index)
plt.title('Location Distribution')
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



12. Deployment

- Use joblib or pickle to save the trained model.
- Use **Flask** or **Fast API** to create a REST API for real-time predictions.
- Deploy on cloud platforms like **AWS**, **Azure**, or **GCP** Or use **Docker** for containerization

13.Source Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('/content/Customer-Churn-Records.csv')
df.head( )
df.info( )
df.describe( )
df.isnull( ).sum( )
df.drop_duplicates( )
df.drop_duplicates( ).sum( )

label_encoder_type = LabelEncoder()
label_encoder_location = LabelEncoder()

data['TransactionType']
label_encoder_type.fit_transform(data['TransactionType'])
data['Location'] = label_encoder_location.fit_transform(data['Location'])

transaction_type_mapping = dict(zip(label_encoder_type.classes_,
range(len(label_encoder_type.classes_))))
location_mapping = dict(zip(label_encoder_location.classes_,
range(len(label_encoder_location.classes_))))

transaction_type_inverse_mapping = {v: k for k, v in
transaction_type_mapping.items()}
location_inverse_mapping = {v: k for k, v in location_mapping.items()}


data['TransactionDate'] = pd.to_datetime(data['TransactionDate'])
```

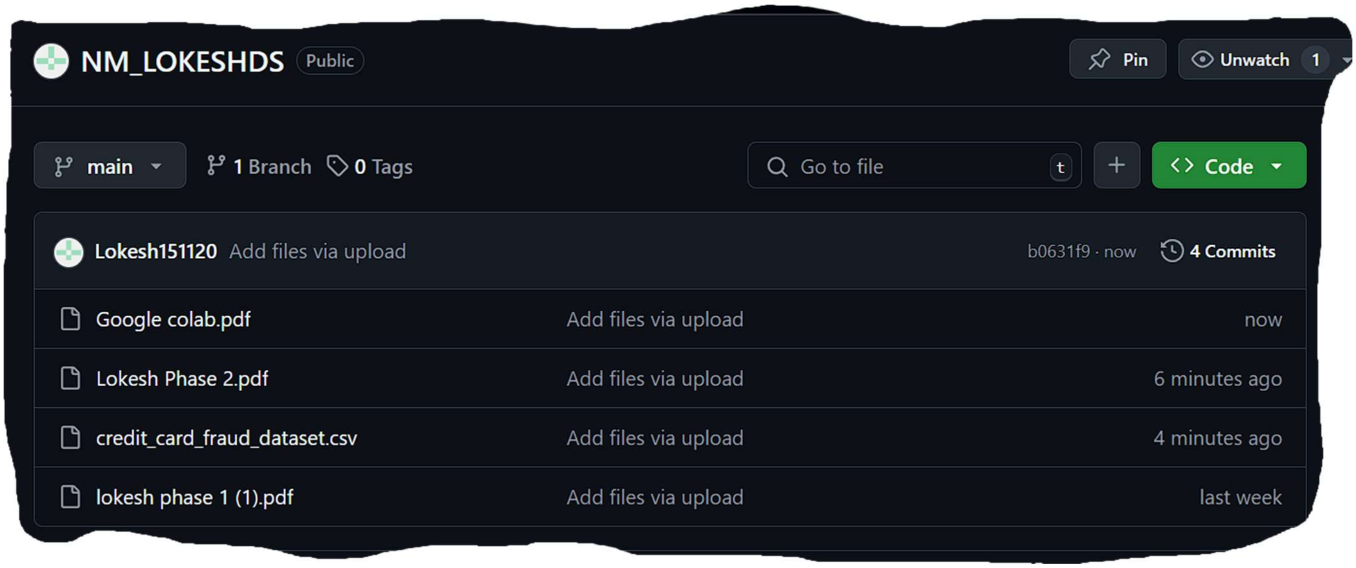
```
data['TransactionHour'] = data['TransactionDate'].dt.hour
data['TransactionDay'] = data['TransactionDate'].dt.day
data['TransactionMonth'] = data['TransactionDate'].dt.month

data = data.drop(columns=['TransactionDate'])

plt.figure(figsize=(6, 4))
sns.countplot(x='IsFraud', data=data)
plt.title('Class Distribution: Fraud vs Legitimate')
plt.xlabel('IsFraud')
plt.ylabel('Count')
plt.show()

plt.figure(figsize=(8, 6))
sns.histplot(data['Amount'], bins=50, kde=True)
plt.title('Distribution of Transaction Amount')
plt.xlabel('Amount')
plt.ylabel('Frequency')
plt.show()

df.hist(figsize=(10,10))
plt.show()

sns.pairplot(df)
plt.show()

for col in ['Geography', 'Gender', 'Card Type']:
le = LabelEncoder()
```

```
df[col] = le.fit_transform(df[col])
df
```

13. Team Members and Roles

S.NO	NAMES	ROLES	RESPONSIBILITY
1	LOKESH J	TEAM LEADER	DATA COLLECTING
2	BINAPALLI MANOJ	MEMBER	DATA CLEANING AND FEATURE ENGINEERING
3	GUNA SEKHER REDDY	MEMBER	MODEL EVALUVATION AND MODEL BUILDING
4	POORNA CHANDRA REDDY	MEMBER	VISUALIZATION AND INTERPRETATION
5	ERUGU PURUSHOTHAM	MEMBER	VISUALIZATION AND INTERPRETATION

GITHUB SCREENSHOT



GOOGLE COLAB LINK

https://colab.research.google.com/drive/1DhuHKjyvE6UbCWOKmfO4f0Ld-1vzLPow#scrollTo=uIRySo2wrH5P

24.

S.NO	NAMES	ROLES	RESPONSIBILITY
1	LOKESH J	TEAM LEADER	DATA COLLECTING
2	BINAPALLI MANOJ	MEMBER	DATA CLEANING AND FEATURE ENGINEERING
3	GUNA SEKHER REDDY	MEMBER	MODEL EVALUVATION AND MODEL BUILDING
4	POORNA CHANDRA REDDY	MEMBER	VISUALIZATION AND INTERPRETATION
5	ERUGU PURUSHOTHAM	MEMBER	VISUALIZATION AND INTERPRETATION