****

**Assessment Report**

on

**“Predict Credit Card Fraud”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**CSE-AIML**

Isha Tyagi

Roll no: 202401100400100

CSE-AIML, B

**Under the supervision of**

“Sandeep Raj”

**Problem Statement: Predicting Credit Card Fraud**

* Credit card fraud is one of the most significant challenges in the financial industry today. Fraudulent transactions result in billions of dollars in losses each year. Detecting fraudulent transactions in real time is crucial for preventing such losses and ensuring the security of users' financial information. This project aims to develop a classification model that can detect fraudulent credit card transactions based on various patterns in the data, such as:
  + Transaction Amount: Larger or unusual amounts could be indicative of fraud.
  + Location: Geographical discrepancies may signal fraudulent activity, such as transactions made from a country or city not typically associated with the user.
  + Device Usage: Transactions made from unfamiliar devices may suggest that an unauthorized user is attempting to make a purchase.
  + User Behaviour: Patterns such as frequency of transactions, time of day, and purchase history can help identify unusual activities that might indicate fraud.

**GOAL:**

Develop a machine learning model to detect fraudulent credit card transactions using patterns in:

* Transaction amount
* Location or time
* Device usage
* User behaviour (like frequency, unusual activity)

This is a binary classification problem, where each transaction is classified as:

* 0 — Not Fraudulent
* 1 — Fraudulent

**Why This Problem Is Important?**

Credit card fraud results in billions of dollars in losses globally every year.  
Detecting fraudulent activity quickly is essential to protect customers and banks.

Traditional methods are manual or rule-based.  
Machine Learning can automatically identify subtle patterns that suggest fraud**.**

**Dataset Characteristics:**

| **Feature Type** | **Description** |
| --- | --- |
| **Time** | **Time since first transaction** |
| **Amount** | **Transaction amount** |
| **V1-V28** | **Anonymized features (via PCA)** |
| **Class** | **Target (0 = Not Fraud, 1 = Fraud)** |

**Example of how data looks:**

| Time | V1 | V2 | ... | V28 | Amount | Class |
| --- | --- | --- | --- | --- | --- | --- |
| 1000 | … | … | ... | … | 200.00 | 0 |
| 1250 | … | … | ... | … | 500.00 | 1 |

**Real-World Challenge: Imbalanced Data**

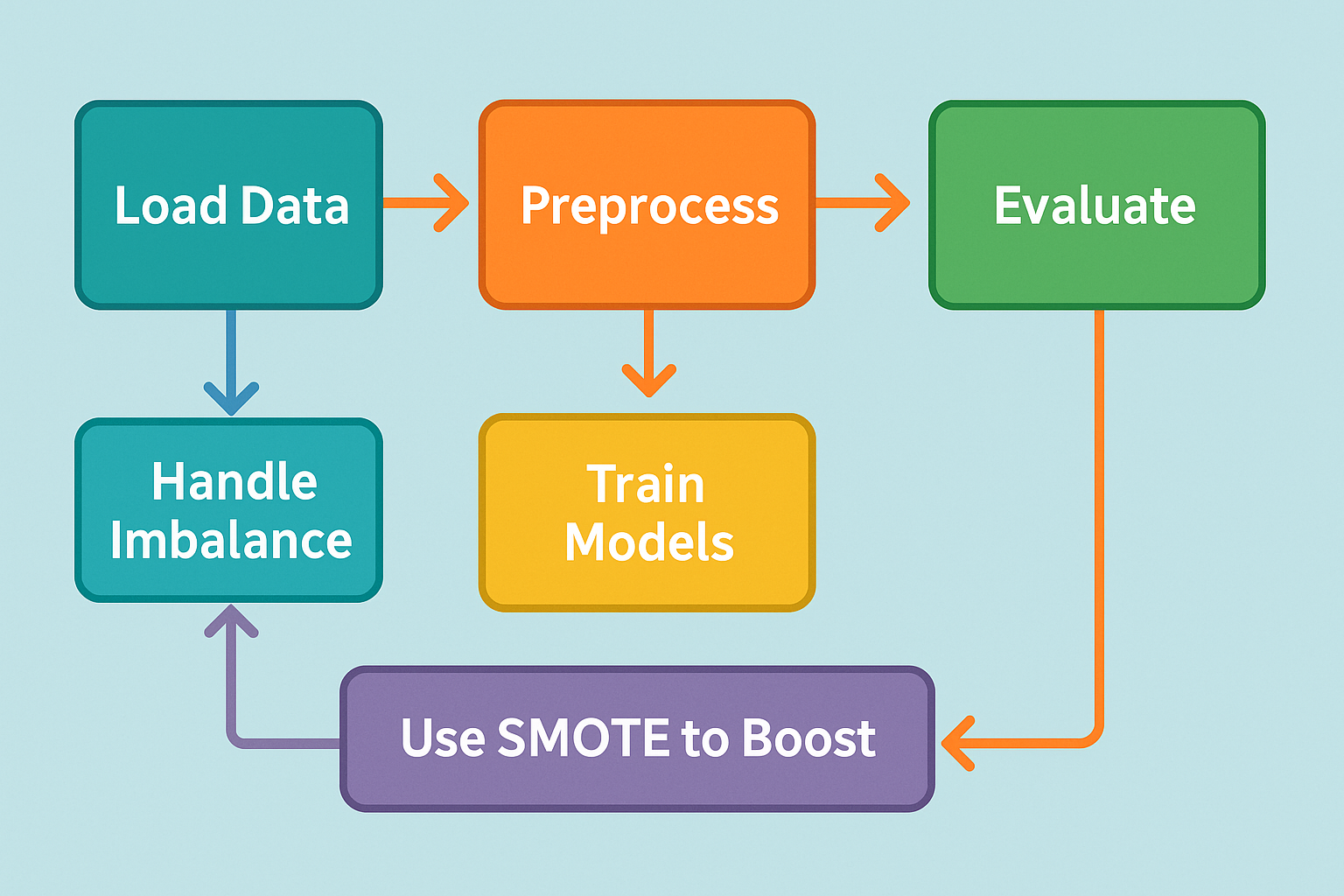
In real data, fraud is rare. Like this:

Non-Fraud: ████████████████████████████████████ 99.8%

Fraud: █ 0.2%

This imbalance makes it hard for models to learn from very few fraud examples.  
We use SMOTE (Synthetic Minority Over-sampling Technique) to balance the data.

**Processing Steps:**

****

**Models Used:**

* **Logistic Regression**: Good baseline for binary classification
* **Random Forest**: Robust, handles nonlinear relationships well

***Note****: XG Boost was also considered but not used due to environment limitations.*

**Methodology**

This section outlines the step-by-step approach followed to detect credit card fraud using machine learning models with a focus on handling class imbalance using SMOTE.

**1. Data Acquisition**

* The dataset was obtained in CSV format named "7. Predict Credit Card Fraud.csv".
* It consists of anonymized transaction data, with 30 features including Time, Amount, and 28 PCA-transformed variables (V1–V28).
* The target variable Class indicates whether a transaction is fraudulent (1) or not (0).

**2. Data Preprocessing**

* Feature and Target Separation: The dataset was split into input features X and target variable y.
* Feature Scaling:
  + The Time and Amount features were scaled using Standard Scaler to normalize their range.
* **Train-Test Split:**
  + The dataset was split into training and testing sets using an 80-20 ratio.
  + Stratified sampling was used to maintain class proportions in both sets.

**3. Handling Class Imbalance**

* The dataset is highly imbalanced with a small proportion of fraudulent transactions.
* To address this, SMOTE (Synthetic Minority Over-sampling Technique) was applied only to the training data.
  + SMOTE generates synthetic examples for the minority class by interpolating between existing minority samples.
  + This helps to improve the model’s ability to detect fraud cases.

**4. Model Development**

Two classification models were used:

* **Logistic Regression:**
  + A linear model suitable for binary classification.
  + Configured with max\_iter =1000 to ensure convergence.
* **Random Forest Classifier:**
  + An ensemble method using 100 decision trees.
  + Known for robustness and handling non-linearity.

Each model was trained on the SMOTE-balanced training set.

**5. Model Evaluation**

The trained models were evaluated on the original (imbalanced) test set using the following metrics:

* **Confusion Matrix:** To understand the true/false positives and negatives.
* **Classification Report:**
  + Includes precision, recall, and F1-score for each class.
* **ROC AUC Score:** Measures the area under the Receiver Operating Characteristic curve.
* **ROC Curve:**
  + Plotted for each model to visualize the trade-off between true positive rate (TPR) and false positive rate (FPR).
  + Helps compare the discriminative ability of models.

**6. Visualization**

* ROC Curves for both models were plotted on a single graph.
* Each curve was labelled with its corresponding model name and AUC score for easy comparison**.**

**Typed code**

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.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

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

from imblearn.over\_sampling import SMOTE

# Load data from a CSV file into a pandas DataFrame

df = pd.read\_csv("7. Predict Credit Card Fraud.csv")

# Separate features (X) and the target label (y)

X = df.drop(columns=['Class']) # All columns except 'Class' are features

y = df['Class'] # 'Class' column is the target

# Scale 'Time' and 'Amount' features to normalize them

scaler = StandardScaler()

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

# Split data into training and testing sets (80% train, 20% test), while preserving class distribution

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

# Apply SMOTE to oversample the minority class in the training data to balance class distribution

smote = SMOTE(random\_state=42)

X\_train\_res, y\_train\_res = smote.fit\_resample(X\_train, y\_train)

# Define the machine learning models to evaluate

models = {

"Logistic Regression": LogisticRegression(max\_iter=1000),

"Random Forest": RandomForestClassifier(n\_estimators=100)

}

# Train and evaluate models

results = {}

for name, model in models.items():

# Train the model on the resampled training set

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

# Predict class labels on the test set

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

# Predict probabilities for the positive class (fraud) on the test set

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

# Generate classification report to evaluate model performance

report = classification\_report(y\_test, y\_pred, output\_dict=True)

# Compute the ROC AUC score

auc = roc\_auc\_score(y\_test, y\_proba)

# Generate confusion matrix to evaluate classification performance

cm = confusion\_matrix(y\_test, y\_pred)

# Store the results for each model

results[name] = {

"report": report,

"auc": auc,

"confusion\_matrix": cm,

"fpr\_tpr": roc\_curve(y\_test, y\_proba)

}

# Plot ROC Curves and Confusion Matrices for comparison

plt.figure(figsize=(14, 6))

# ROC Curve comparison

plt.subplot(1, 2, 1)

for name, result in results.items():

fpr, tpr, \_ = result["fpr\_tpr"]

plt.plot(fpr, tpr, label=f"{name} (AUC = {result['auc']:.4f})")

# Plot a diagonal line representing random classifier

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve Comparison")

plt.legend()

plt.grid(True)

# Confusion Matrix heatmap

plt.subplot(1, 2, 2)

for name, result in results.items():

cm = result["confusion\_matrix"]

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["Not Fraud", "Fraud"], yticklabels=["Not Fraud", "Fraud"])

plt.title(f"{name} - Confusion Matrix")

plt.xlabel('Predicted')

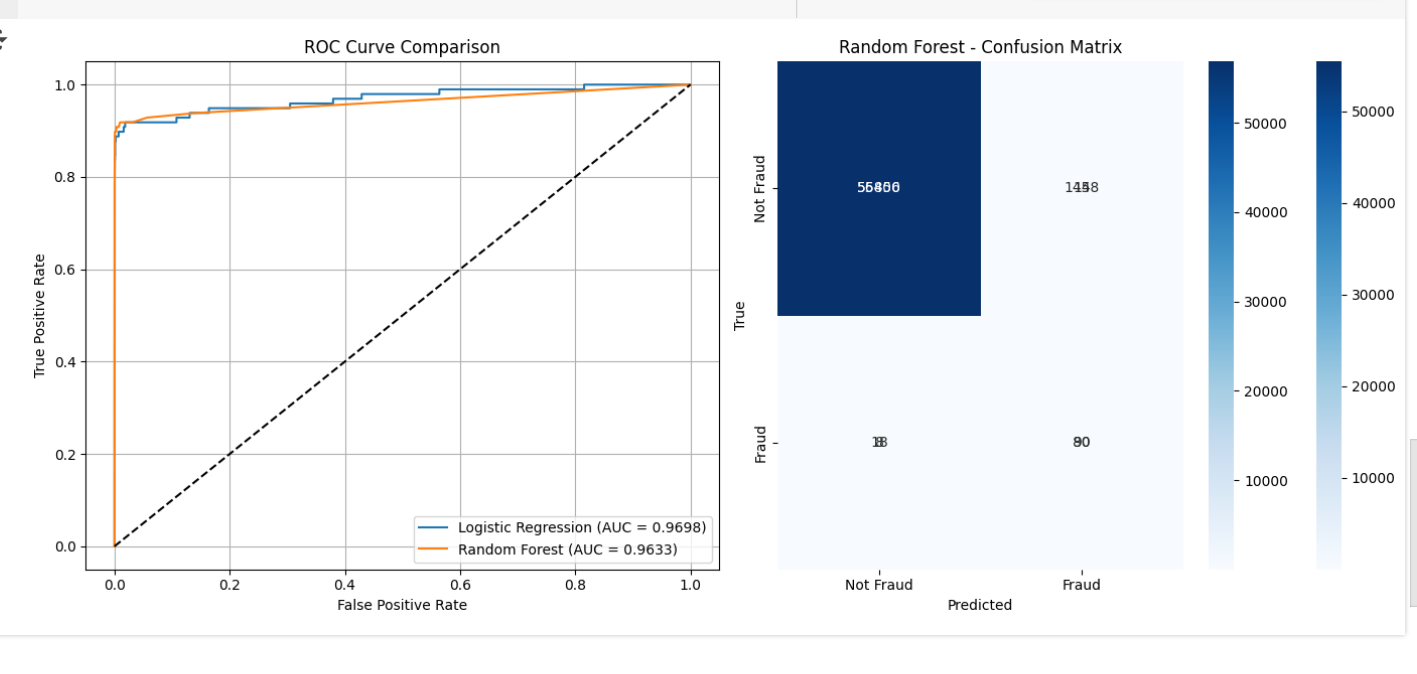
plt.ylabel('True')

# Adjust layout and show plots

plt.tight\_layout()

plt.show()

**OUTPUT**

****