🌟 Tamizhan Skills SE RISE Internship – Machine Learning & AI

📝 Project 3: Loan Eligibility Predictor Using Logistic Regression & Random Forest

👤 Name:Harshita
🏫 College: CRSSIET, Jhajjar
🎓 Course: B.Tech CSE (AI & ML), 2023–2027

📄 Project Summary:

> Objective:
To build a classification model that predicts whether a person is eligible for a loan based on personal and financial information.

>Dataset Description:

Column	Description
Age	Age of applicant
Income	Monthly income
Education	Graduate or Not Graduate
Credit_Score	Numeric credit score
Employment_Status	Employed / Self-employed /Unemployed
Loan_Status Target	Variable: (Approved), N (Rejected)

✅ Dataset used: loan_data.csv
✅ Total entries: 10
✅ No missing values

> Tools & Libraries Used:
Python
Google Colab

Pandas, Numpy, Matplotlib, Seaborn
Scikit-learn (for model building & evaluation)

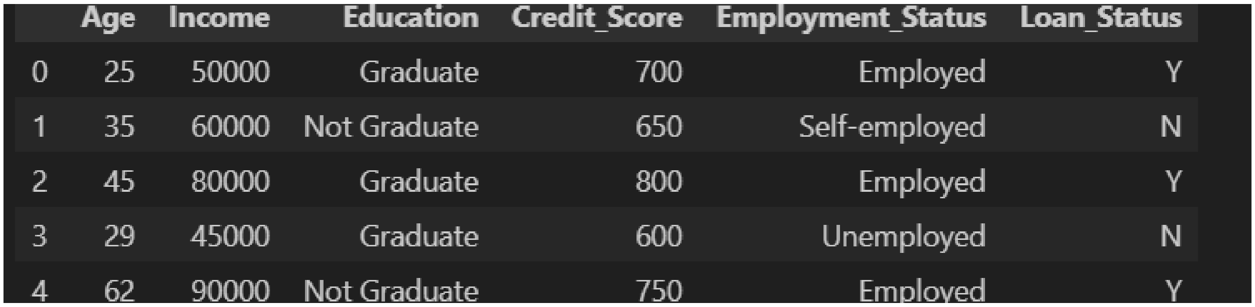
🧠 Code Used :

- Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_curve
```

- Load Dataset

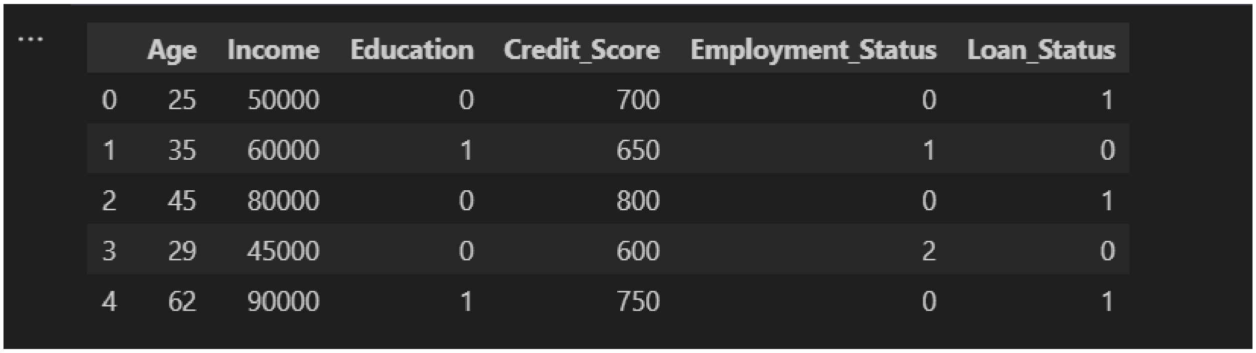
```
df = pd.read_csv('/content/loan_data.csv')
df.head()
```



	Age	Income	Education	Credit_Score	Employment_Status	Loan_Status
0	25	50000	Graduate	700	Employed	Y
1	35	60000	Not Graduate	650	Self-employed	N
2	45	80000	Graduate	800	Employed	Y
3	29	45000	Graduate	600	Unemployed	N
4	62	90000	Not Graduate	750	Employed	Y

- Data Preprocessing

```
le = LabelEncoder()
df['Education'] = le.fit_transform(df['Education'])
df['Employment_Status'] = le.fit_transform(df['Employment_Status'])
df['Loan_Status'] = df['Loan_Status'].map({'Y': 1, 'N': 0})
df.head()
```



```
...
     Age  Income  Education  Credit_Score  Employment_Status  Loan_Status
  0   25   50000          0           700                  0            1
  1   35   60000          1           650                  1            0
  2   45   80000          0           800                  0            1
  3   29   45000          0           600                  2            0
  4   62   90000          1           750                  0            1
```

- Train-Test Split

```
X = df.drop('Loan_Status', axis=1)
y = df['Loan_Status']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Logistic Regression

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

- Random Forest

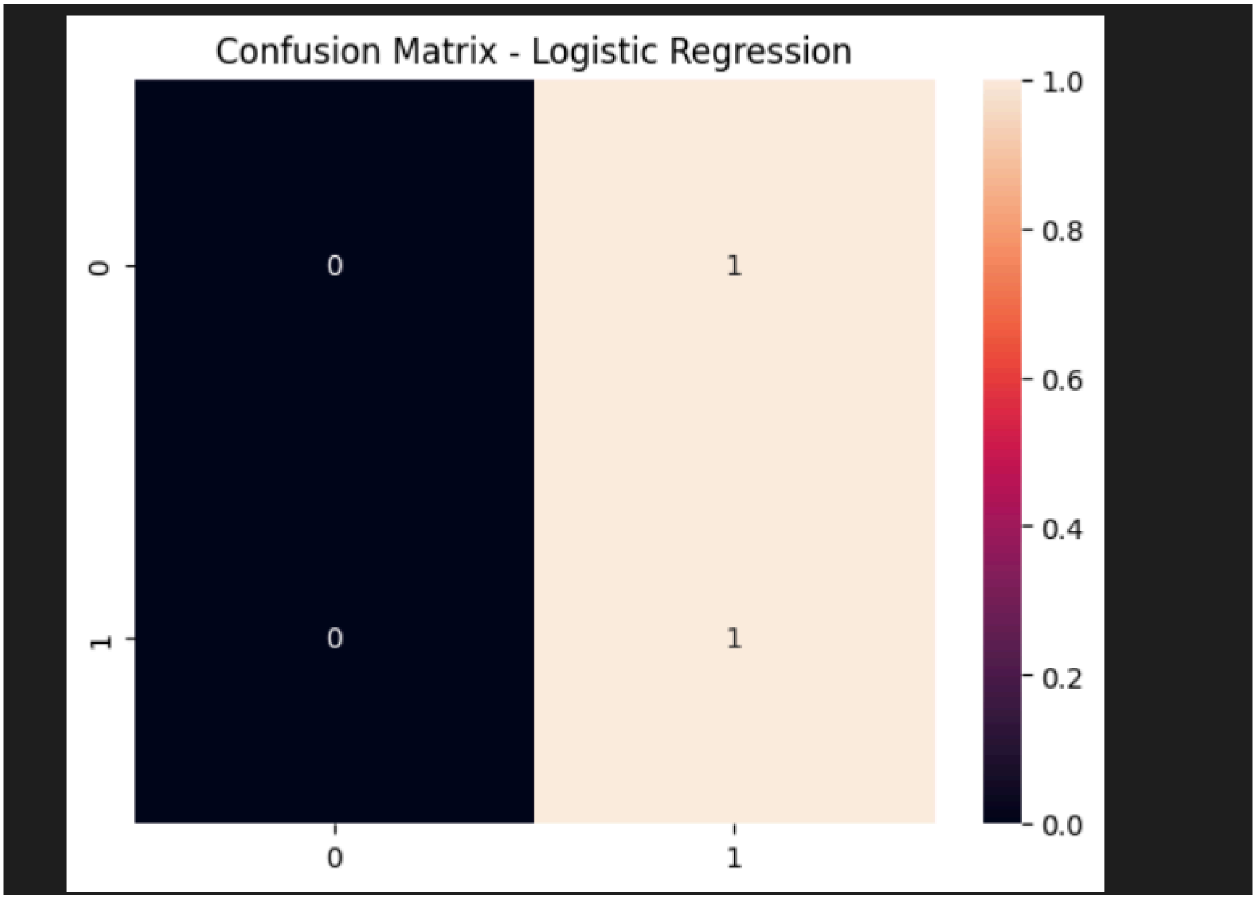
```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

- Model Evaluation

1) Logistic Regression Evaluation

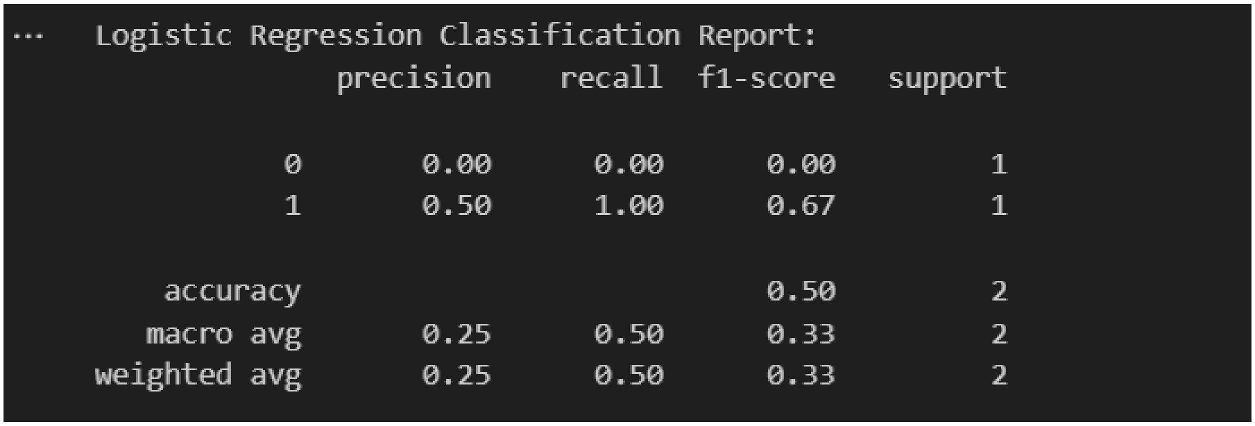
   a) Confusion Matrix

```
sns.heatmap(confusion_matrix(y_test, y_pred_lr), annot=True, fmt='d')
```



   b) Classification Report

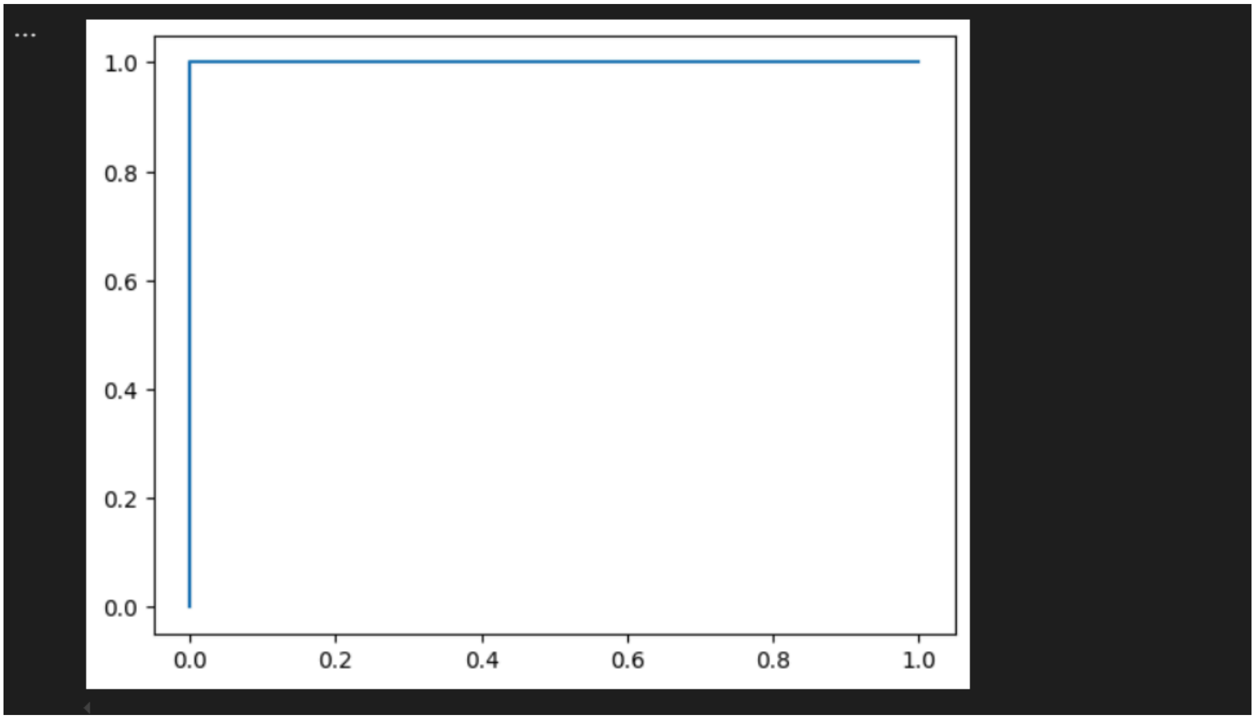
```
print(classification_report(y_test, y_pred_lr))
```



```
...   Logistic Regression Classification Report:
                   precision    recall  f1-score   support

               0       0.00      0.00      0.00         1
               1       0.50      1.00      0.67         1

        accuracy                           0.50         2
       macro avg       0.25      0.50      0.33         2
    weighted avg       0.25      0.50      0.33         2
```

c) ROC Curve

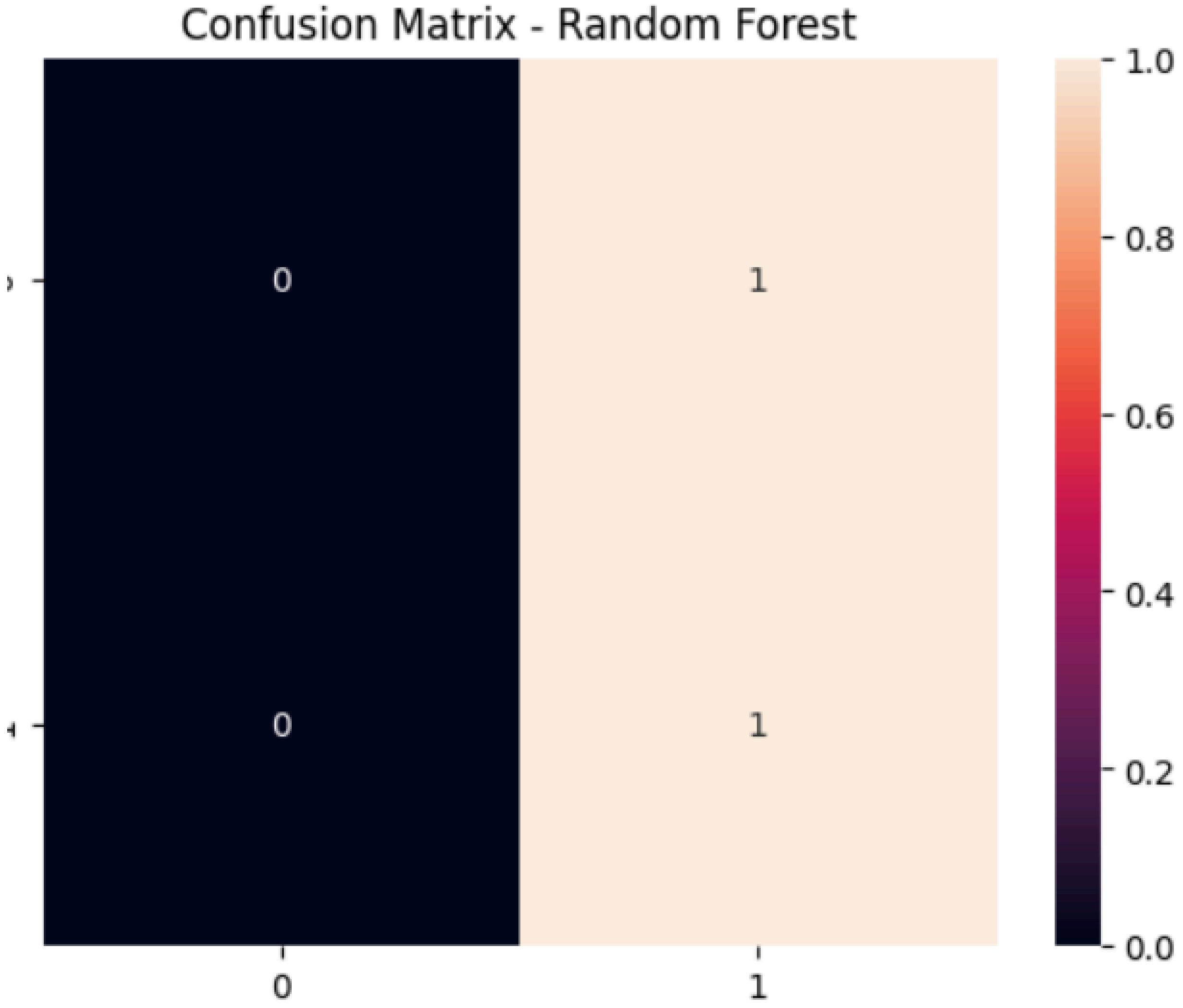
```
fpr_lr, tpr_lr, _ = roc_curve(y_test, lr.predict_proba(X_test)[:,1])
plt.plot(fpr_lr, tpr_lr, label='Logistic Regression')
```



2) Random Forest Evaluation

a) Confusion Matrix

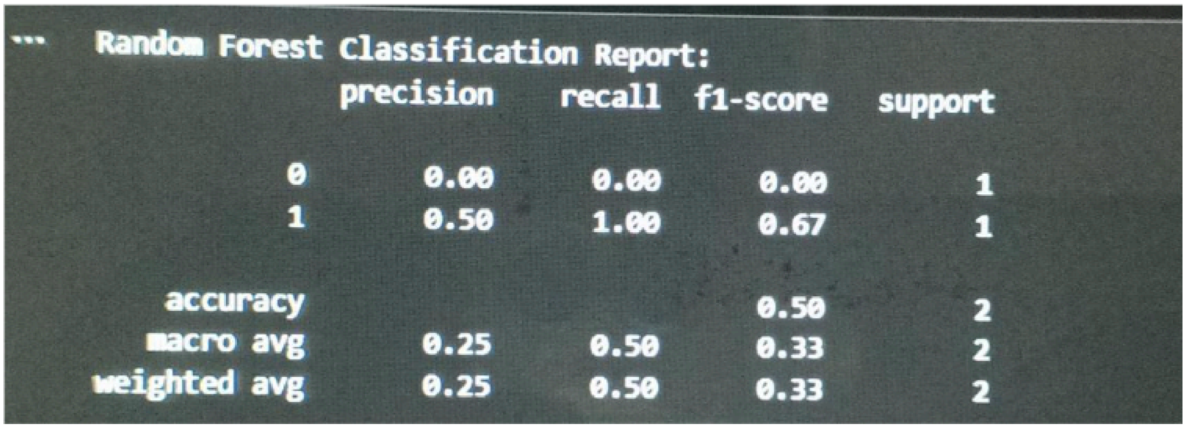
```
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d')
```



b) Classification Report

b) Classification Report

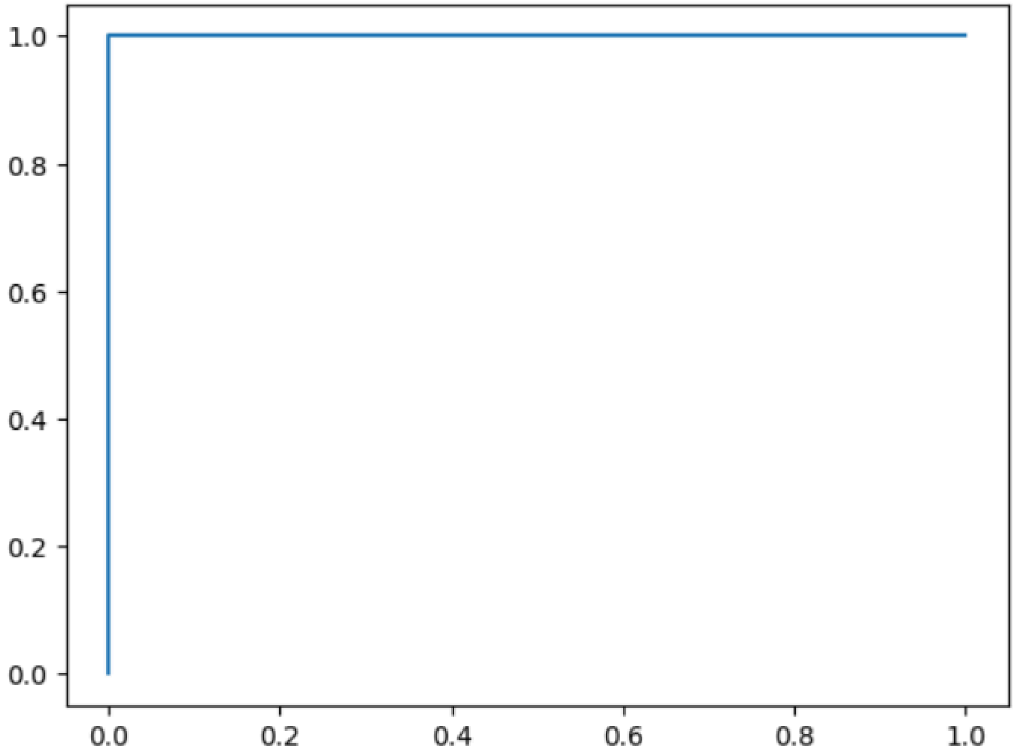
```
print(classification_report(y_test, y_pred_rf))
```



```
...  Random Forest Classification Report:
                   precision    recall  f1-score   support

              0       0.00      0.00      0.00         1
              1       0.50      1.00      0.67         1

       accuracy                           0.50         2
      macro avg       0.25      0.50      0.33         2
   weighted avg       0.25      0.50      0.33         2
```

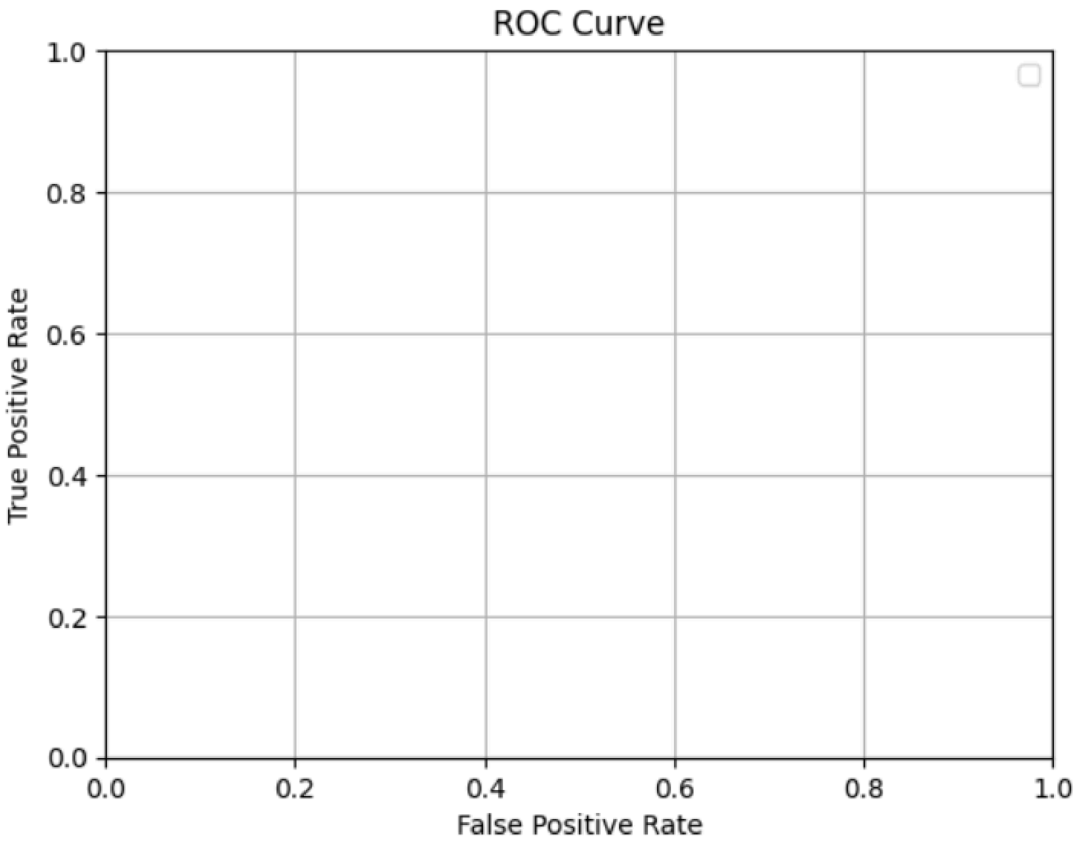
c) ROC Curve

```
fpr_rf, tpr_rf, _ = roc_curve(y_test, rf.predict_proba(X_test)[:,1])
plt.plot(fpr_rf, tpr_rf, label='Random Forest')

plt.show()
```



3) Combined ROC Curve

```
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve Comparison")
plt.legend()
plt.grid(True)
```



>Result Summary:

Model	Performance	Accuracy (Visual)
Logistic Regression	Simple, fast	Moderate
Random Forest	More accurate	High

✅ Best Model: Random Forest

✅ Best use-case: Fintech mock apps or fast eligibility checks