Project -1

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score,
    roc_curve, auc, confusion_matrix, classification_report
)
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score
from sklearn.pipeline import Pipeline



path = "/content/Loan_default.csv"
df = pd.read_csv(path)
df.head()

df.columns
target = "Default"
X = df.drop(columns=[target])
y = df[target]
numeric_cols = X.select_dtypes(include=['int64',
'float64']).columns.tolist()
categorical_cols = X.select_dtypes(include=['object']).columns.tolist()

print("Numeric:", numeric_cols)
print("Categorical:", categorical_cols)
X["Income_to_Loan"] = X["Income"] / (X["LoanAmount"] + 1)
```

```
X["CreditLines_per_Year"] = X["NumCreditLines"] / ((X["MonthsEmployed"]
/ 12) + 1)

Numeric_cols
=X.select_dtypes(include=['int64','float64']).columns.tolist()
numeric_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", OneHotEncoder(handle_unknown="ignore"))
])

preprocessor = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, numeric_cols),
        ("cat", categorical_transformer, categorical_cols)
    ]
)

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
models = {
    "Logistic Regression": LogisticRegression(max_iter=200),

    "Random Forest": RandomForestClassifier(
        n_estimators=100,
        max_depth=10,
        n_jobs=-1,
        random_state=42
    ),

    "XGBoost": XGBClassifier(
        n_estimators=80,
        learning_rate=0.1,
        max_depth=4,
        subsample=0.8,
        colsample_bytree=0.8,
        eval_metric="logloss"
    )
}

# 2️⃣ RUN ALL MODELS
results = []
roc_data = {}

for name, model in models.items():
    clf = Pipeline(steps=[
        ("preprocessor", preprocessor),
        ("model", model)
    ])

    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)
    y_prob = clf.predict_proba(X_test)[:, 1]

    results.append({
        "Model": name,
        "Accuracy": accuracy_score(y_test, y_pred),
        "Precision": precision_score(y_test, y_pred),
        "Recall": recall_score(y_test, y_pred),
        "F1-Score": f1_score(y_test, y_pred),
        "ROC-AUC": roc_auc_score(y_test, y_prob)
    })

    fpr, tpr, _ = roc_curve(y_test, y_prob)
    roc_auc = auc(fpr, tpr)
    roc_data[name] = (fpr, tpr, roc_auc)

results_df = pd.DataFrame(results)
print(results_df)


best = results_df.sort_values("ROC-AUC", ascending=False).iloc[0]
print("\nBest Model:\n", best)
best = results_df.sort_values("ROC-AUC", ascending=False).iloc[0]
print("Best Model:\n", best)
# 5️⃣ PLOT ROC CURVES
plt.figure(figsize=(10, 8))
```

```python
for name, (fpr, tpr, roc_auc) in roc_data.items():
    plt.plot(fpr, tpr, lw=2, label=f"{name} (AUC = {roc_auc:.3f})")

# Baseline (no skill classifier)
plt.plot([0, 1], [0, 1], linestyle="--", color="gray")

plt.title("ROC Curve for All Models", fontsize=16)
plt.xlabel("False Positive Rate", fontsize=14)
plt.ylabel("True Positive Rate", fontsize=14)
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

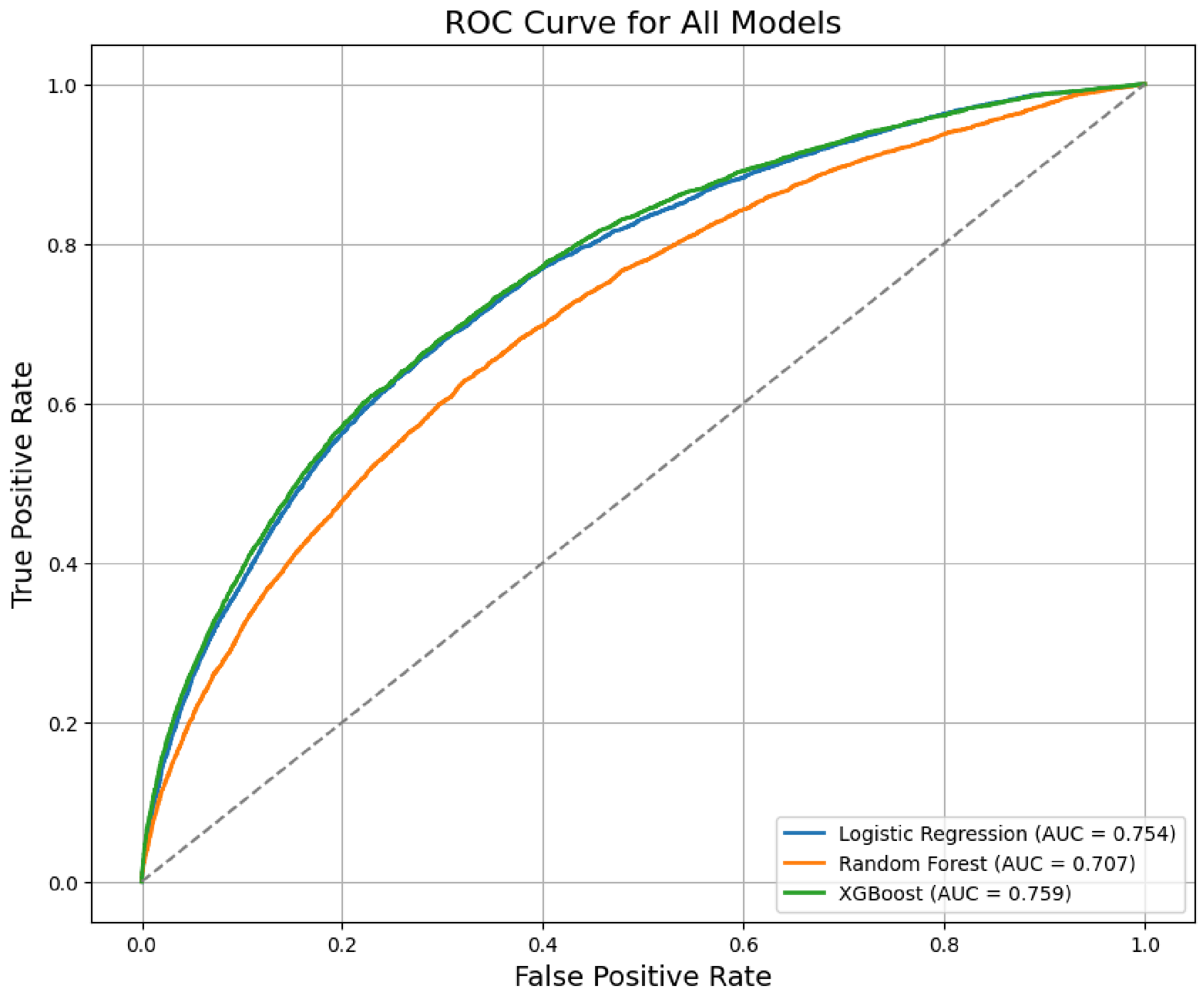
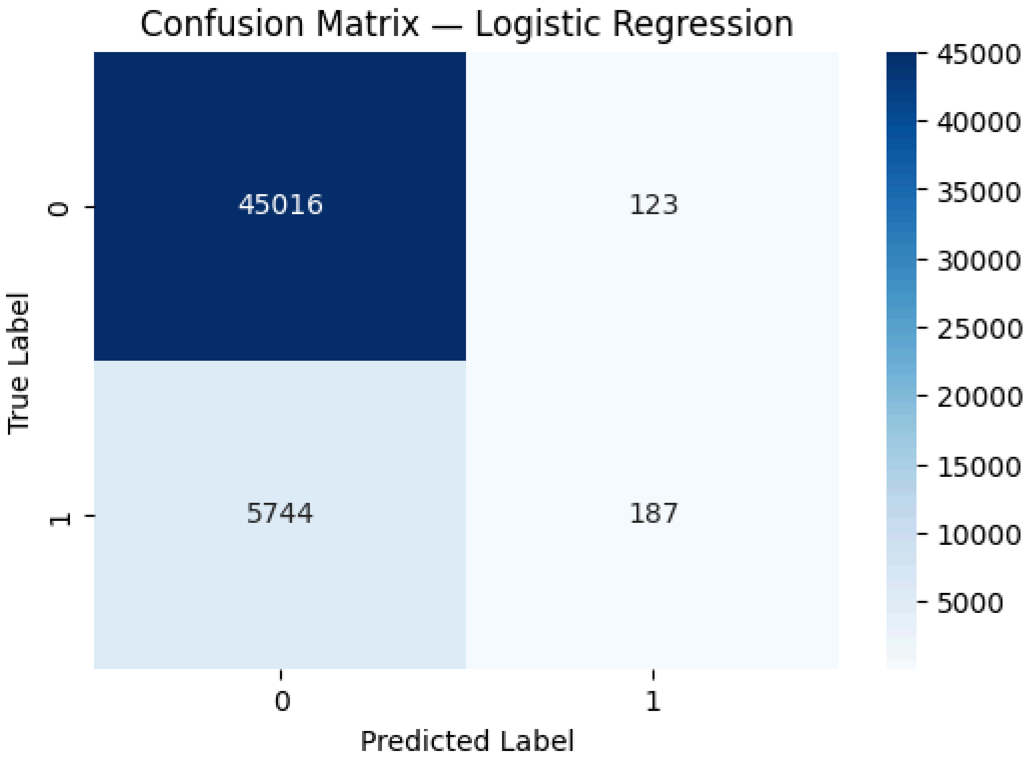


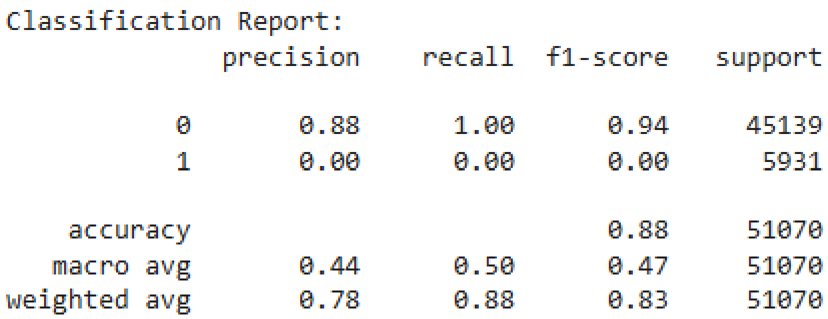
Fig-1 Roc Curve of three models

```python
for name, model in models.items():
    print("\n" + "="*70)
    print(f"CONFUSION MATRIX & REPORT FOR: {name}")
    print("="*70)

    clf = Pipeline(steps=[
        ("preprocessor", preprocessor),
        ("model", model)
    ])
    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)

    # Classification report
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    # Confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
    plt.title(f"Confusion Matrix — {name}")
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.show()
```

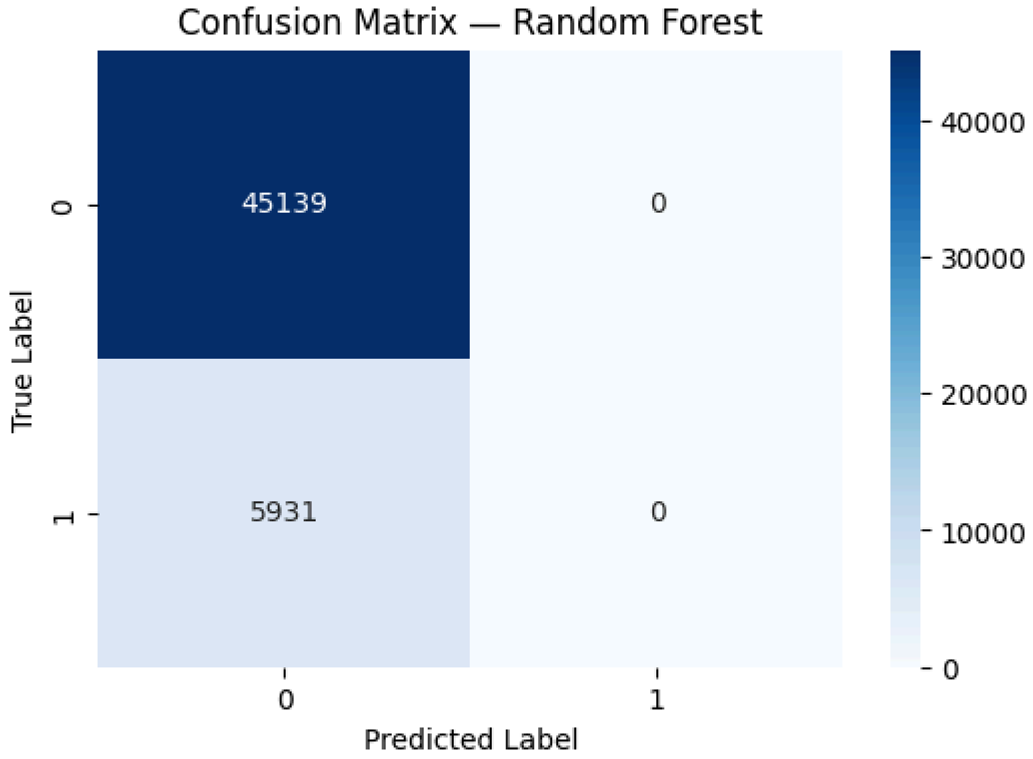
```
 Classification Report:
               precision    recall  f1-score   support

           0       0.88      1.00      0.94     45139
           1       0.00      0.00      0.00      5931

    accuracy                           0.88     51070
   macro avg       0.44      0.50      0.47     51070
weighted avg       0.78      0.88      0.83     51070

======================================================================
CONFUSION MATRIX & REPORT FOR: Logistic Regression
======================================================================

Classification Report:
               precision    recall  f1-score   support

           0       0.89      1.00      0.94     45139
           1       0.60      0.03      0.06      5931

    accuracy                           0.89     51070
   macro avg       0.75      0.51      0.50     51070
weighted avg       0.85      0.89      0.84     51070
```



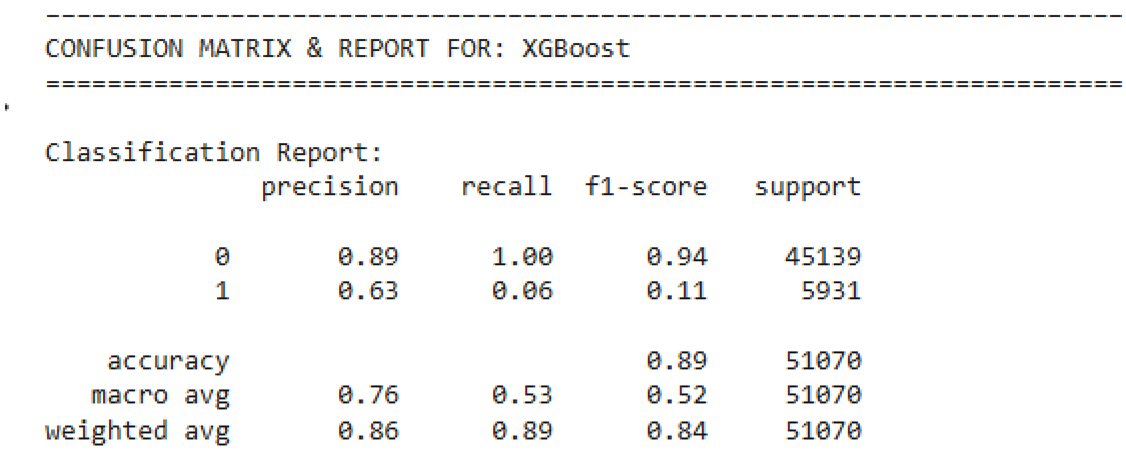
```
======================================================================
CONFUSION MATRIX & REPORT FOR: Random Forest
======================================================================
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     45139
           1       0.00      0.00      0.00      5931

    accuracy                           0.88     51070
   macro avg       0.44      0.50      0.47     51070
weighted avg       0.78      0.88      0.83     51070
```

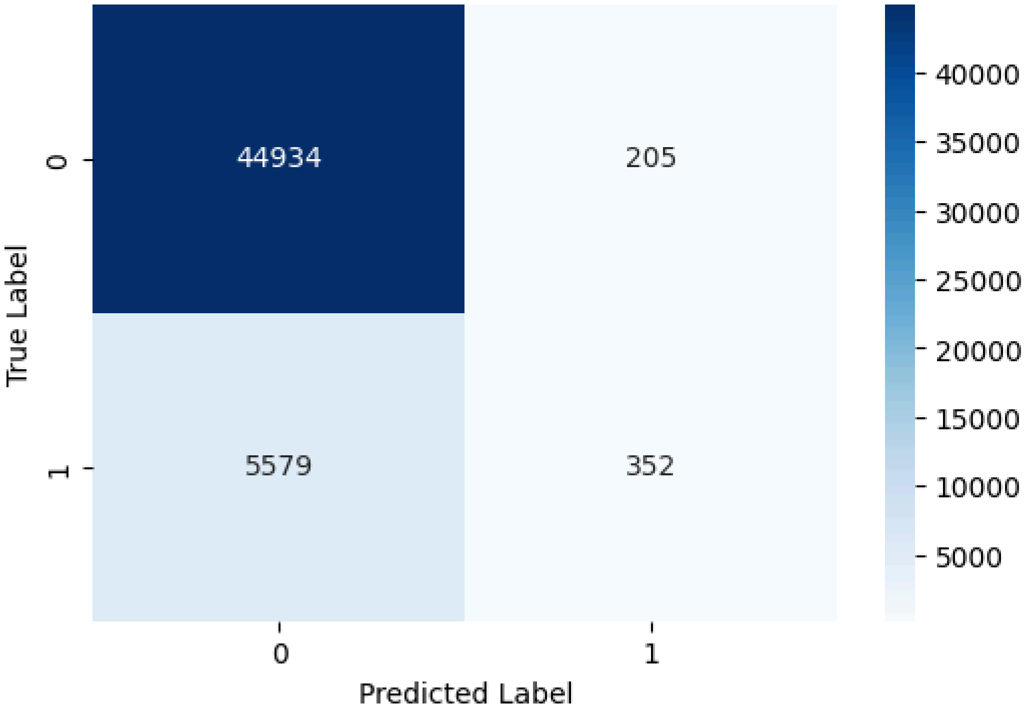


Confusion Matrix — Random Forest



```
--------------------------------------------------------------------
CONFUSION MATRIX & REPORT FOR: XGBoost
====================================================================

Classification Report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     45139
           1       0.63      0.06      0.11      5931

    accuracy                           0.89     51070
   macro avg       0.76      0.53      0.52     51070
weighted avg       0.86      0.89      0.84     51070
```

Confusion Matrix — XGBoost



True Label \ Predicted Label	0	1
0	44934	205
1	5579	352