Loan Approval Prediction

Executive Summary

This project builds supervised machine learning models to predict loan approval based on borrower features.

Key Steps: • Data preprocessing and missing value handling • Categorical encoding and scaling • Class imbalance handling using SMOTE • Model comparison (Logistic Regression & Random Forest)

Results: • Random Forest achieved 83% accuracy. • ROC-AUC score: 0.79 • SMOTE improved class balance and recall performance.

Business Insight: Random Forest model is recommended for deployment with threshold adjustment to reduce risky approvals.

```
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, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, roc_auc_score,
confusion_matrix

from imblearn.over_sampling import SMOTE

df = pd.read_csv("C:\\Users\\DeLL\\OneDrive\\Desktop\\Alfido Tech
Internship\\loan_prediction.csv")
df.head()
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0
```

```
   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
```

```
df.shape
df.info()
df.isnull().sum()
df['Loan_Status'].value_counts()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 67.2+ KB
```

```
Loan_Status
Y    422
N    192
Name: count, dtype: int64
```

Removing Loan_ID as it does not contribute to prediction.

```
df.drop(columns=['Loan_ID'], inplace=True)

categorical_cols =
['Gender','Married','Dependents','Self_Employed','Credit_History']

for col in categorical_cols:
    df[col] = df[col].fillna(df[col].mode()[0])
```

```
numerical_cols = ['LoanAmount','Loan_Amount_Term']

for col in numerical_cols:
    df[col] = df[col].fillna(df[col].median())

df.isnull().sum()
```

```
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

```
df['Loan_Status'] = df['Loan_Status'].map({'Y':1, 'N':0})

df['Loan_Status'].value_counts()
```

```
Loan_Status
1    422
0    192
Name: count, dtype: int64
```

```
df = pd.get_dummies(df, drop_first=True)

df.head()
df.shape
```

```
(614, 15)
```

```
X = df.drop('Loan_Status', axis=1)
y = df['Loan_Status']

from sklearn.model_selection import train_test_split

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

X_train.shape
X_test.shape
```

```
(123, 14)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)

X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

pd.Series(y_train_smote).value_counts()

Loan_Status
1    337
0    337
Name: count, dtype: int64

from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(max_iter=1000)
lr.fit(X_train_smote, y_train_smote)

y_pred_lr = lr.predict(X_test)

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random_state=42)
rf.fit(X_train_smote, y_train_smote)

y_pred_rf = rf.predict(X_test)

from sklearn.metrics import classification_report, roc_auc_score

print("Logistic Regression Report")
print(classification_report(y_test, y_pred_lr))
print("ROC-AUC:", roc_auc_score(y_test, y_pred_lr))

print("\nRandom Forest Report")
print(classification_report(y_test, y_pred_rf))
print("ROC-AUC:", roc_auc_score(y_test, y_pred_rf))

Logistic Regression Report
              precision    recall  f1-score   support

           0       0.70      0.68      0.69        38
           1       0.86      0.87      0.87        85
```

```
    accuracy                           0.81       123
   macro avg       0.78      0.78      0.78       123
weighted avg       0.81      0.81      0.81       123

ROC-AUC: 0.7773993808049535

Random Forest Report
              precision    recall  f1-score   support

           0       0.74      0.68      0.71        38
           1       0.86      0.89      0.88        85

    accuracy                           0.83       123
   macro avg       0.80      0.79      0.80       123
weighted avg       0.83      0.83      0.83       123

ROC-AUC: 0.7891640866873064
```

```
y_prob = rf.predict_proba(X_test)[:,1]

custom_threshold = 0.6
y_custom = (y_prob >= custom_threshold).astype(int)

print(classification_report(y_test, y_custom))
```

```
              precision    recall  f1-score   support

           0       0.63      0.71      0.67        38
           1       0.86      0.81      0.84        85

    accuracy                           0.78       123
   macro avg       0.75      0.76      0.75       123
weighted avg       0.79      0.78      0.78       123
```

Business Interpretation

Random Forest performed slightly better than Logistic Regression with higher accuracy and ROC-AUC.

However, recall for rejected loans is relatively low (0.68), meaning some risky applications may still be approved.

For deployment, a higher decision threshold (e.g., 0.6) can be used to reduce financial risk, even if it slightly lowers recall.

Conclusion

The Random Forest model achieved the best overall performance with 83% accuracy and ROC-AUC of 0.79. While both models performed well in predicting approved loans, further tuning and threshold adjustment is recommended before deployment.