Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Foundations Of Data Analytics

CSE3505

Loan Approval Prediction

Project Report

Faculty

Dr. Priyadarshini R

<u>Team Member</u>:

Niranjan J 19MIA1003

Academic Year

Fall Semester 2022 - 2023

Bonafide Certificate

This is to certify that this project titled “Loan Approval Prediction” has been carried out by **Niranjan J** (19MIA1003) under the guidance of Dr. Priyadarshini R as a project component in foundations of data analytics course, and this is my original work.

Index

Abstract

Account firms and banks need to automatize the credit qualification activity (continuously) essentially dependent on data given by customers when rounding out an online structure. The attributes include Sex, Marital Status, Education, Number of Dependents, Salary, Loan Amount, Credit History, and different subtleties are incorporated.To digitize this interaction, they made an issue to group the client sections that can apply for a credit sum, permitting them to focus on these clients explicitly. They have introduced a fractional informational collection for this situation. Approval of Loan is a very common real-life problem that every company faces in their lending operations. If the loan approval process is automated, it can save a lot of man hours and improve the speed of service to the customers. The increase in customer satisfaction and savings in operational costs are significant. However, the rewards can only be realised if the bank has a sturdy model in place to accurately forecast which client's loans it should accept and which it should reject, in order to reduce potential risk. In this project 3 algorithms of classification based machine learning will be applied, that is Decision tree, Random Forest Classifier and Naive Bayes to predict the loan approval with large accuracy. By implementation of this project we will be able to predict and ensure that applicant for the loan is safe or not by this loan approval prediction system.

Introduction

Banking system have large number of products to earn profit, but their vital source of income is from its credit system. Because Credit system can earn from interests of that loans which they credit. Banking system always need accurate modelling system for large number of issues. The prediction of credit defaulters is one of the difficult task for anybank.But by forecasting the loan defaulters, the banks definitely may reduce its loss by reducing its non-profit assets,so that recovery of approved loans can take place without any loss and it can play as the contributing parameter of the bank statement. This makes the study of this loan approval prediction important. Machine Learning techniques are very crucial and useful in prediction of these type of data. Machine Literacy is a subset of artificial intelligence that allows computer programs to automatically learn from former tasks. It works by analysing the data, relating patterns, and incorporating minimum mortal intervention. Nearly any work that can be done using a data description pattern or set of rules can be done using a machine learning machine. This allows companies to modify processes that preliminarily only humans could make hypotheticals for client service calls, accounts, and reviews. Loan distribution is the middle enterprise of virtually every bank. Loan distribution is the middle enterprise of nearly every bank. Utmost of a bank's means come directly from the gains it makes from the loans it makes. The main thing of the banking terrain is to put wealth in a safe place. Currently, many banks / financial companies approve loans after a verification and validation redemption process, but it is not yet clear if the selected applicant is correct among all applicants. Through this system, it is possible to predict whether a particular applicant is safe and the entire process of verifying the characteristics will be automated by machine learning technology. Credit forecasting is very useful for both bank employees and applicants

Literature Review

Vaidya had:

Suggested a method for approving loan forecasts using logistic regression. Logistic Regression is one of the most popular and very useful classification based algorithm. The purpose or the importance of using Logistic Regression was that it uses the concept of predictive analysis which was suitable enough for describing the data.

M. Bayraktar et al. Proposed :

A method for credit risk analysis using machine learning. Boltzman machine was used to make the analysis for risk calculation of loan.

Y. Shi and P. Song :

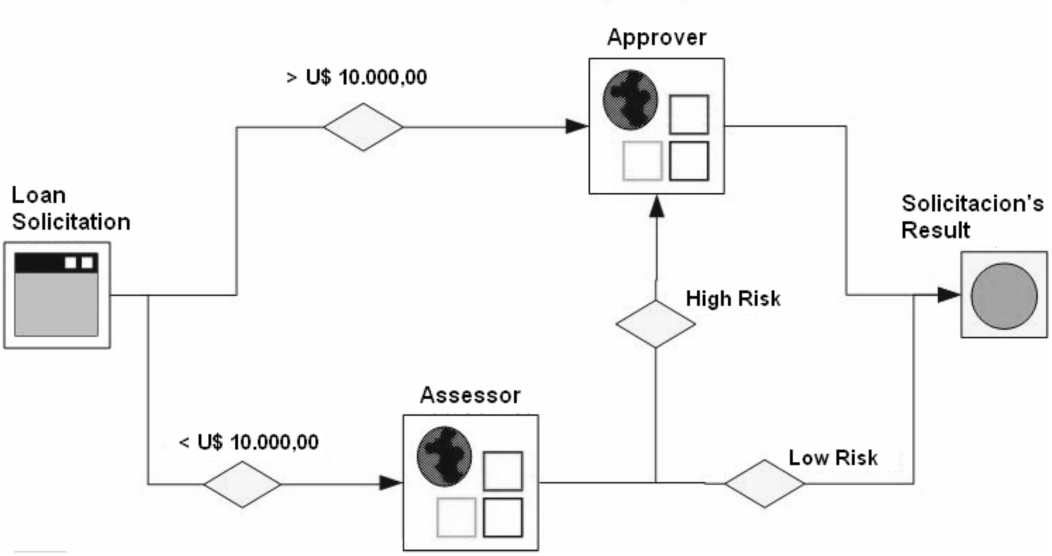
Proposed a method for evaluating project loans using risk analysis. The method evaluate the risk involved in loans of commercial banks.

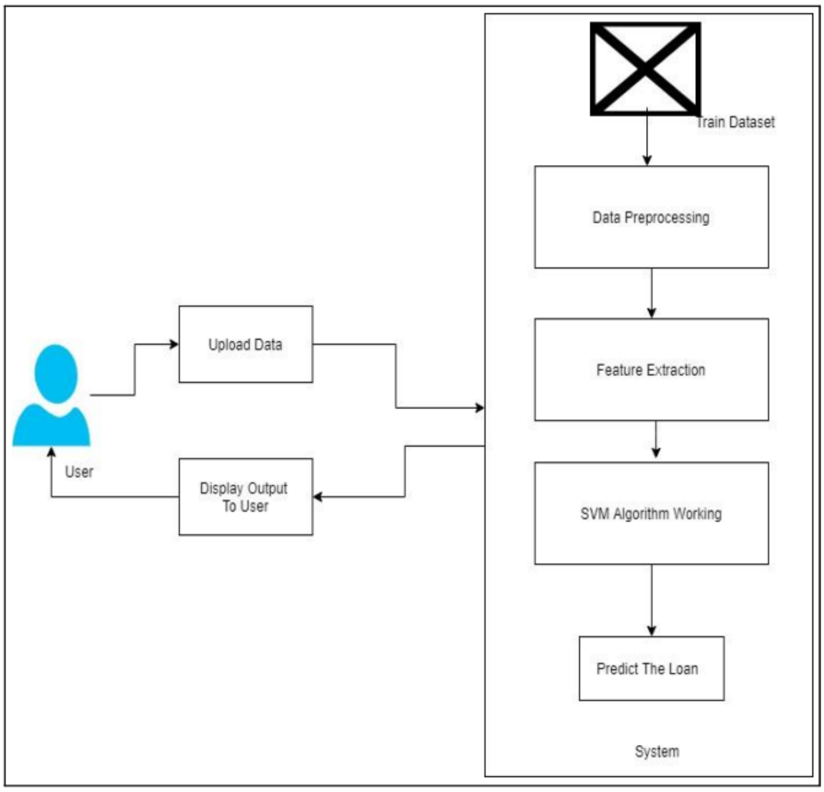
V. C. T. Chan et al:

Proposed a credit approval system using web-services. For clients loan as approved by the system. The consumer provides extra relevant information with the credit application. This information's are processed by Credit Approval System which finally give credit score to the applicant. After going through this, Machine learning algorithm are very helpful in predicting outcomes even when the data is huge in size.

According to the authors, the forecasting process begins with data clean-upandprocessing, missing value substitution, data set experimental analysis, and modelling, and continues to model evaluation and test data testing. A logistic regression model has been executed. The highest accuracy obtained with the original dataset is 0.811. Models are compared based on performance measurements such as sensitivity and specificity. As a result of analysing, the following conclusions were drawn. However, other characteristics of customers that play a very important role in lending decisions and forecasting defaulters should also be evaluated. Some other traits, such as gender and marriage history, do not seem to be considered by the company. A credit credibility soothsaying system that helps companies make the right opinions to authorize or reject the credit claims of guests. This helps the banking assiduity to open effective distribution channels. This means that if the customer has a minimum repayment capacity, their system can avoid future risks. Including other techniques (using the Weka tool) that are better than the general data mining model has been implemented and tested for domains

System Architecture and Design





Model Implementation

- Collect data
- Importing libraries
- Reading dataset
- Preprocessing dataset (imputation and encoders)
- Splitting dataset into training and testing
- Building model for classification
- Model evaluation and accuracy
- Comparing built machine learning models with neural network models.

<u>Performance Analysis</u>

The evaluation metrics includes precision, recall, f1-score and support. Using these metrics we can find the performance analysis like confusion matrix (True positives, False positives, True negatives and False negatives) of the machine learning models.

```
### NAIVE BAYES CLASSIFIER

model = GaussianNB()

model.fit(X_train, y_train)
pred = model.predict(X_valid)
print(classification_report(y_valid, pred))
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.47      0.60        43
         1.0       0.82      0.96      0.89       111

    accuracy                           0.82       154
   macro avg       0.83      0.71      0.74       154
weighted avg       0.83      0.82      0.81       154
```

```
/home/niranjan/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column
-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

```
matrix = confusion_matrix(y_valid, pred)
matrix_df = pd.DataFrame(matrix)
print(matrix_df)
```

```
    0    1
0  20   23
1   4  107
```

0 - No loan | 1 - loan accepted

```
### RFC

model2 = RandomForestClassifier(n_estimators=100, criterion='entropy')

model2.fit(X_train, y_train)
pred2 = model2.predict(X_valid)
print(classification_report(y_valid, pred2))
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.49      0.61        43
         1.0       0.83      0.95      0.89       111

    accuracy                           0.82       154
   macro avg       0.82      0.72      0.75       154
weighted avg       0.82      0.82      0.81       154
```

```
/tmp/ipykernel_32460/669114507.py:1: DataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples,), for example using ravel().
  model2.fit(X_train, y_train)
```

```
matrix = confusion_matrix(y_valid, pred2)
matrix_df = pd.DataFrame(matrix)
print(matrix_df)
```

```
    0    1
0  21   22
1   5  106
```

0 - No loan | 1 - loan accepted

Results

Loan companies grant loans after a thorough verification and validation process. However, they do not know with absolute certainty whether the applicant will be able to repay the loan without difficulty. The loan Prediction System will allow them to choose the most deserving applicants quickly, easily, and efficiently. It may provide the bank with unique benefits. In this paper, we have reviewed the process of building a Loan Approval Prediction System. Data collection, Exploratory Data Analysis, Data Preprocessing, Model Building, and Model Testing are the analytical processes involved in building this system. We conducted a thorough review of the previous research papers in this field in this paper. The most widely used algorithms, such as Decision Tree, and Random Forest Technique, have been examined and implemented.

Conclusion and Future Work

In this project, machine learning was used to predict loan acceptance. The prediction method begins with data pre-processing, filling the missing values, experimental data analysis and evaluating the machine learning models. After evaluating model on test dataset, each of these algorithms obtained a precision rate between 75% and 85%.

The future work is I will build a Neural network model for this data and find the performance analysis of that Neural network model comparing with the classification models that are already built.

References

https://www.semanticscholar.org/paper/Loan-Prediction-by-using-Machine-Learning-Models-Supriya-Pavani/54646ad5279f94bb717dd57263da4cf360a9a8c8#paper-header

https://ieeexplore.ieee.org/document/9155614

https://www.ijert.org/predict-loan-approval-in-banking-system-machine-learning-approach-for-cooperative-banks-loan-approval

https://ijirt.org/master/publishedpaper/IJIRT151769_PAPER.pdf

https://www.projectpro.io/article/loan-prediction-using-machine-learning-project-source-code/632

https://www.itm-conferences.org/articles/itmconf/pdf/2022/04/itmconf_icacc2022_03019.pdf

[1] M. Sheikh, A. Goel, T. Kumar, “An Approach for Prediction of Loan Approval using Machine Learning Algorithm,” International Conference on Electronics and Sustainable Communication Systems (ICESC), (2020).

[2] S. M S, R. Sunny T, “Loan Credibility Prediction System Based on Decision Tree Algorithm,” International Journal of Engineering Research & Technology (IJERT) Vol. 4 Issue 09, (2015).

[3] A. Kumar, I. Garg and S. Kaur, “Loan Approval Prediction based on Machine Learning Approach,” IOSR Journal of Computer Engineering, (2016).

[4] Dr K. Kavitha, “Clustering Loan Applicants based on Risk Percentage using K-Means Clustering Techniques,” IJARCSSE - Volume 6, Issue 2, (2016).

[5] P. Dutta, “A STUDY ON MACHINE LEARNING ALGORITHM FOR ENHANCEMENT OF LOAN PREDICTION”, International Research 3 ITM Web of Conferences 44, 03019 (2022) https://doi.org/10.1051/itmconf/20224403019 ICACC-2022 Journal of Modernization in Engineering Technology and Science, (2021). [

6] G. Arutjothi, Dr C. Senthamarai, “Prediction of Loan Status in Commercial Bank using Machine Learning Classifier,” Proceedings of the International Conference on Intelligent Sustainable Systems, (2017).

```
# Niranjan J 19MIA1003

import pandas as pd
import csv
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB, GaussianNB,
BernoulliNB, CategoricalNB
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

TRAINING DATA

```
print(train.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

print(train.isnull().sum())   # finding missing values

Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
```

```
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

```
print(train['Gender'].unique())
```

```
['Male' 'Female' nan]
```

```
print(train['Dependents'].unique())
```

```
['0' '1' '2' '3+' nan]
```

```
print(train['Education'].unique())
```

```
['Graduate' 'Not Graduate']
```

```
print(train['Self_Employed'].unique())
```

```
['No' 'Yes' nan]
```

```
print(train['Property_Area'].unique())
```

```
['Urban' 'Rural' 'Semiurban']
```

```
encoder = OrdinalEncoder()
df = train.copy()
cate = ['Dependents']
df[cate] = encoder.fit_transform(train[cate])
Dependents = df[cate]

imputer = SimpleImputer()
depn = Dependents.copy()
depn = pd.DataFrame(imputer.fit_transform(Dependents))

train['Dependents'] = depn
#print(train)

X = train.copy()
X = X.drop(['Loan_ID', 'Loan_Status'], axis=1)       # dropping
columns

cate_col = (X.dtypes == 'object')
cate_col = list(cate_col[cate_col].index)
print("Categorical Variables : ", cate_col)
```

```
Categorical Variables :  ['Gender', 'Married', 'Education',
'Self_Employed', 'Property_Area']
```

```
X[cate_col] = encoder.fit_transform(df[cate_col])
#print(X)
```

```
X_prep = X.copy()
X_prep = pd.DataFrame(imputer.fit_transform(X))
X_prep.columns = X.columns
#print(X_prep)                   # preprocessed X - features

df2 = train.copy()
cate2 = ['Loan_Status']
df2[cate2] = encoder.fit_transform(train[cate2])
y_prep = df2[cate2].copy()

#print(y_prep)
```

TEST DATA

```
print(test.head())
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001015   Male     Yes          0      Graduate            No
1  LP001022   Male     Yes          1      Graduate            No
2  LP001031   Male     Yes          2      Graduate            No
3  LP001035   Male     Yes          2      Graduate            No
4  LP001051   Male      No          0  Not Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5720                  0       110.0             360.0
1             3076               1500       126.0             360.0
2             5000               1800       208.0             360.0
3             2340               2546       100.0             360.0
4             3276                  0        78.0             360.0

   Credit_History Property_Area
0             1.0         Urban
1             1.0         Urban
2             1.0         Urban
3             NaN         Urban
4             1.0         Urban
```

```
print(test.isnull().sum())
```

```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
```

```
Property_Area         0
dtype: int64
```

```
print(test['Gender'].unique())
```

```
['Male' 'Female' nan]
```

```
print(test['Dependents'].unique())
```

```
['0' '1' '2' '3+' nan]
```

```
print(test['Education'].unique())
```

```
['Graduate' 'Not Graduate']
```

```
print(test['Self_Employed'].unique())
```

```
['No' 'Yes' nan]
```

```
print(test['Property_Area'].unique())
```

```
['Urban' 'Semiurban' 'Rural']
```

```
encoder = OrdinalEncoder()
df3 = test.copy()
cate = ['Dependents']
df3[cate] = encoder.fit_transform(test[cate])
Dependents2 = df3[cate]

imputer = SimpleImputer()
depn2 = Dependents2.copy()
depn2 = pd.DataFrame(imputer.fit_transform(Dependents2))

test['Dependents'] = depn2
#print(test)

X2 = test.copy()
X2 = X2.drop(['Loan_ID'], axis=1)       # dropping columns
#print(X2)

X2[cate_col] = encoder.fit_transform(df3[cate_col])
#print(X2)

X_test_prep = X2.copy()
X_test_prep = pd.DataFrame(imputer.fit_transform(X2))
X_test_prep.columns = X2.columns
#print(X_test_prep)                  # preprocessed X - features
```

PREPROCESSED DATA

```
print(X_prep)
```

```
     Gender  Married  Dependents  Education  Self_Employed
ApplicantIncome  \
```

```
0       1.0      0.0          0.0        0.0             0.0
5849.0
1       1.0      1.0          1.0        0.0             0.0
4583.0
2       1.0      1.0          0.0        0.0             1.0
3000.0
3       1.0      1.0          0.0        1.0             0.0
2583.0
4       1.0      0.0          0.0        0.0             0.0
6000.0
..      ...      ...          ...        ...             ...
...
609     0.0      0.0          0.0        0.0             0.0
2900.0
610     1.0      1.0          3.0        0.0             0.0
4106.0
611     1.0      1.0          1.0        0.0             0.0
8072.0
612     1.0      1.0          2.0        0.0             0.0
7583.0
613     0.0      0.0          0.0        0.0             1.0
4583.0

     CoapplicantIncome  LoanAmount  Loan_Amount_Term
Credit_History  \
0                  0.0  146.412162             360.0             1.0

1               1508.0  128.000000             360.0             1.0

2                  0.0   66.000000             360.0             1.0

3               2358.0  120.000000             360.0             1.0

4                  0.0  141.000000             360.0             1.0

..                 ...         ...               ...             ...

609                0.0   71.000000             360.0             1.0

610                0.0   40.000000             180.0             1.0

611              240.0  253.000000             360.0             1.0

612                0.0  187.000000             360.0             1.0

613                0.0  133.000000             360.0             0.0


     Property_Area
```

```
0              2.0
1              0.0
2              2.0
3              2.0
4              2.0
..             ...
609            0.0
610            0.0
611            2.0
612            2.0
613            1.0

[614 rows x 11 columns]
```

```
print(y_prep)
```

```
     Loan_Status
0            1.0
1            0.0
2            1.0
3            1.0
4            1.0
..           ...
609          1.0
610          1.0
611          1.0
612          1.0
613          0.0

[614 rows x 1 columns]
```

```
print(X_test_prep)
```

```
     Gender  Married  Dependents  Education  Self_Employed
ApplicantIncome  \
0       1.0      1.0         0.0        0.0            0.0
5720.0
1       1.0      1.0         1.0        0.0            0.0
3076.0
2       1.0      1.0         2.0        0.0            0.0
5000.0
3       1.0      1.0         2.0        0.0            0.0
2340.0
4       1.0      0.0         0.0        1.0            0.0
3276.0
..      ...      ...         ...        ...            ...
...
362     1.0      1.0         3.0        1.0            1.0
4009.0
363     1.0      1.0         0.0        0.0            0.0
4158.0
```

```
364     1.0      0.0         0.0        0.0           0.0
3250.0
365     1.0      1.0         0.0        0.0           0.0
5000.0
366     1.0      0.0         0.0        0.0           1.0
9200.0

     CoapplicantIncome  LoanAmount  Loan_Amount_Term
Credit_History  \
0                  0.0       110.0             360.0        1.000000

1               1500.0       126.0             360.0        1.000000

2               1800.0       208.0             360.0        1.000000

3               2546.0       100.0             360.0        0.825444

4                  0.0        78.0             360.0        1.000000

..                 ...         ...               ...             ...

362             1777.0       113.0             360.0        1.000000

363              709.0       115.0             360.0        1.000000

364             1993.0       126.0             360.0        0.825444

365             2393.0       158.0             360.0        1.000000

366                0.0        98.0             180.0        1.000000


     Property_Area
0              2.0
1              2.0
2              2.0
3              2.0
4              2.0
..             ...
362            2.0
363            2.0
364            1.0
365            0.0
366            0.0

[367 rows x 11 columns]
```

TRAINING DATA SPLIT

```
X_train, X_valid, y_train, y_valid = train_test_split(X_prep, y_prep,
test_size=0.25, random_state=0)

X_train.reset_index(inplace=True)            # resetting index
X_train = X_train.drop('index', axis=1)

y_train.reset_index(inplace=True)
y_train = y_train.drop('index', axis=1)

X_valid.reset_index(inplace=True)
X_valid = X_valid.drop('index', axis=1)

y_valid.reset_index(inplace=True)
y_valid = y_valid.drop('index', axis=1)

### NAIVE BAYES CLASSIFIER

model = GaussianNB()

model.fit(X_train, y_train)
pred = model.predict(X_valid)
print(classification_report(y_valid, pred))
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.47      0.60        43
         1.0       0.82      0.96      0.89       111

    accuracy                           0.82       154
   macro avg       0.83      0.71      0.74       154
weighted avg       0.83      0.82      0.81       154


/home/niranjan/.local/lib/python3.8/site-packages/sklearn/utils/
validation.py:1111: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
matrix = confusion_matrix(y_valid, pred)
matrix_df = pd.DataFrame(matrix)
print(matrix_df)
```

```
    0    1
0  20   23
1   4  107
```

0 - No loan | 1 - loan accepted

```
test_pred = model.predict(X_test_prep)
print(test_pred)
```

```
[1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1.
1.
 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0.
1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0.
0.
 1. 1. 1. 0. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0.
1.
 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0.
1.
 0. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 0. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 0. 0. 0. 1.
1.
 1. 0. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 0. 1. 0. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1.
1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
0.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1.]
```

```python
### RFC

model2 = RandomForestClassifier(n_estimators=100, criterion='entropy')

model2.fit(X_train, y_train)
pred2 = model2.predict(X_valid)
print(classification_report(y_valid, pred2))
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.49      0.61        43
         1.0       0.83      0.95      0.89       111

    accuracy                           0.82       154
   macro avg       0.82      0.72      0.75       154
weighted avg       0.82      0.82      0.81       154
```

```
/tmp/ipykernel_32460/669114507.py:1: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  model2.fit(X_train, y_train)
```

```
matrix = confusion_matrix(y_valid, pred2)
matrix_df = pd.DataFrame(matrix)
print(matrix_df)
```

```
    0    1
0  21   22
1   5  106
```

0 - No loan | 1 - loan accepted

```
# Decision Tree
```

```
model3 = DecisionTreeClassifier(criterion='gini', splitter='best',
random_state=0)
```

```
model3.fit(X_train, y_train)
pred3 = model3.predict(X_valid)
print(classification_report(y_valid, pred3))
```

```
              precision    recall  f1-score   support

         0.0       0.47      0.44      0.46        43
         1.0       0.79      0.81      0.80       111

    accuracy                           0.71       154
   macro avg       0.63      0.63      0.63       154
weighted avg       0.70      0.71      0.70       154
```

```
matrix = confusion_matrix(y_valid, pred3)
matrix_df = pd.DataFrame(matrix)
print(matrix_df)
```

```
    0   1
0  19  24
1  21  90
```