Loan Approval Prediction Machine Learning

BANKING BEGINNER MACHINE LEARNING

Introduction

In this article, we are going to solve the Loan Approval Prediction Hackathon hosted by Analytics Vidhya. This is a classification problem in which we need to classify whether the loan will be approved or not. classification refers to a predictive modeling problem where a class label is predicted for a given example of input data. A few examples of classification problems are Spam Email detection, Cancer detection, Sentiment Analysis, etc.

To understand more about classification problems you can go through this link.

Table of Content

Understanding the Problem Statement

Dream Housing Finance company deals in all kinds of home loans. They have a presence across all urban, semi-urban and rural areas. The customer first applies for a home loan and after that, the company validates the customer eligibility for the loan.

The company wants to automate the loan eligibility process (real-time) based on customer detail provided while filling out online application forms. These details are Gender, Marital Status, Education, number of Dependents, Income, Loan Amount, Credit History, and others.

To automate this process, they have provided a dataset to identify the customer segments that are eligible for loan amounts so that they can specifically target these customers.

You can find the complete details about the problem statement here and also download the training and test data.

As mentioned above this is a Binary Classification problem in which we need to predict our Target label which is “Loan Status”.

Loan status can have two values: Yes or NO.

Yes: if the loan is approved

NO: if the loan is not approved

So using the training dataset we will train our model and try to predict our target column that is “Loan Status” on the test dataset.

About the dataset

So train and test dataset would have the same columns except for the target column that is “Loan Status”.

Train dataset:

Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self_Employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_History	credit history meets guidelines
Property_Area	Urban/ Semi Urban/ Rural
Loan_Status	(Target) Loan approved (Y/N)

Load Essential Python Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.model_selection import train_test_split
```

Load Training/ Test Dataset

```
train=pd.read_csv("/content/drive/MyDrive/train_ctrUa4K.csv")
test = pd.read_csv("/content/drive/MyDrive/test_lAUu6dG.csv")
ss = pd.read_csv("/content/drive/MyDrive/sample_submission_49d68Cx.csv")
```

Size of Train/Test Data

So we have 614 rows and 13 columns in our training dataset.

In test data, we have 367 rows and 12 columns because the target column is not included in the test data.

First look at the Dataset

Categorical Columns: Gender (Male/Female), Married (Yes/No), Number of dependents (Possible values:0,1,2,3+), Education (Graduate / Not Graduate), Self-Employed (No/Yes), credit history(Yes/No), Property Area (Rural/Semi-Urban/Urban) and Loan Status (Y/N)(i. e. Target variable)

Numerical Columns: Loan ID, Applicant Income, Co-applicant Income, Loan Amount, and Loan amount term

Data Preprocessing

Concatenating the train and test data for data preprocessing:

dropping the unwanted column :

Identify missing values :

Imputing the missing values:

- Fill null values with mode

Next, we will be using Iterative imputer for filling missing values of LoanAmount and Loan_Amount_Term

So now as we have imputed all the missing values we go on to mapping the categorical variables with the integers.

We map the values so that we can input the train data into the model as the model does not accept any string values.

Exploratory Data Analysis (EDA)

Splitting the data to new_train and new_test so that we can perform EDA.

Mapping ‘N’ to 0 and ‘Y’ to 1

Univariate Analysis:

Output:

Univariate Analysis Observations

1. More Loans are approved Vs Rejected
2. Count of Male applicants is more than Female
3. Count of Married applicant is more than Non-married
4. Count of graduate is more than non-Graduate
5. Count of self-employed is less than that of Non-Self-employed
6. Maximum properties are located in Semiurban areas
7. Credit History is present for many applicants
8. The count of applicants with several dependents=0 is maximum.

Bivariate Analysis

Mean ApplicantIncome of 0 and 1 are almost the same (o: no,1: Yes)

Mean Co- ApplicantIncome of 1 is slightly more than 0 (o: no,1 Yes)

The mean value of Loan Amount applied by males (0) is slightly higher than Females(1).

If you are married then the loan amount requested is slightly higher than non-married

Male have higher Co-applicant income than females in all three property areas

Correlation matrix

Output:

Feature Engineering

Total Income :

EMI :

Lets assume that interest rate=10.0 # hence r = ((10/12)/100) = 0.00833

Additional Features :

Bin Information :

Drop Unwanted Column :

Size after feature engineering :

We have added 8 new features

Building Machine Learning Model:

Creating X (input variables) and Y (Target Variable) from the new_train data.

Using train test split on the training data for validation

We have a (70:30) split on the training data.

Using ML algorithm for training

We have used multiple algorithms for training purposes like Decision Tree, Random Forest, SVC, Logistic Regression, XGB Regressor, etc.

Among all the algorithms logistic regression performs best on the validation data with an accuracy score of **82.7%**.

After getting an accuracy of 82.7% I tried fine-tuning it to improve my accuracy score using GridSearchCV.

The best parameters I got after fine-tuning were:

After fine-tuning the logistic regression model the accuracy score improved from 82.7% to 83.24%.

Predicting on test data

Prepare Sumbisson file:

Conclusion

After the Final Submission of test data, my accuracy score was 78%.

Feature engineering helped me increase my accuracy.

Amazingly Logistic Regression worked better than all other Ensemble models.

Article Url - https://www.analyticsvidhya.com/blog/2022/02/loan-approval-prediction-machine-learning/

Vedansh Shrivastava