LOAN PREDICTION CLASSIFICATION

(Machine Learning Project)

By

**Name of the Student Roll Number**

Jatin 10103263

**Submitted to:**

Mr. Prateek Bhatia



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**(DEEMED TO BE A UNIVERSITY), PATIALA, PUNJAB**

**INDIA**

**Aug - Dec 2019**

**Problem Statement**

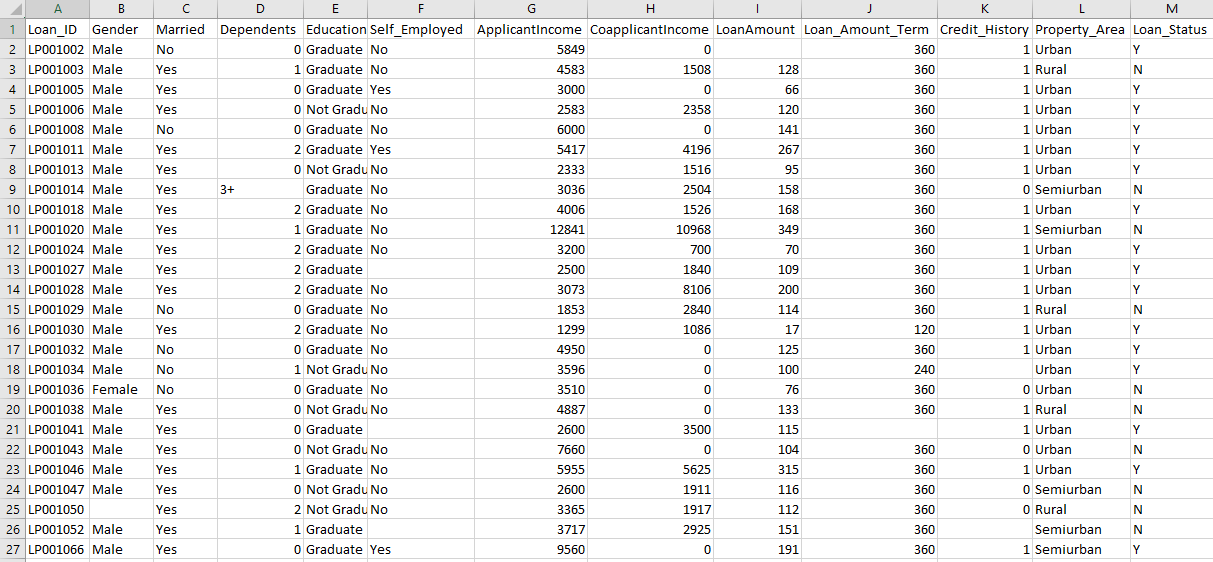
Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan. Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers. It is a live competition problem on Analytics Vidhya where solution is unknown to us.

It is a classification problem where we have to predict whether a loan would be approved or not. In a classification problem, we have to predict discrete values based on a given set of independent variables(s). Classification can be of two types:

Binary Classification: In this classification we have to predict either of the two given classes. For example: classifying the gender as male or female, predicting the result as win or lose, etc.

Multiclass Classification: Here we have to classify the data into three or more classes. For example: classifying a movie's genre as comedy, action or romantic, classify fruits as oranges, apples, or pears, etc.

**Dataset**

****

***Fig: Screenshot of Dataset***

**Dataset Description**

Dataset contains 13 columns and is split into two parts:

1. train data (614 rows)
2. test data (367 rows)

Description of the fields are as follow:

***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 - Loan approved (Y/N)

**Software Used:**

.



[This Photo](https://en.wikipedia.org/wiki/PyCharm) by Unknown Author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)

PyCharm is used for this project. PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains

**Importing data:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

*TRAIN DATASET*

dataset\_train = pd.read\_csv("loan\_train.csv")

X\_train = dataset\_train.iloc[:, 1:-1].values

y\_train = dataset\_train.iloc[:, -1].values

*TEST DATASET*

dataset\_test = pd.read\_csv("loan\_test.csv")

X\_test = dataset\_test.iloc[:, 1:12].values

**Data Preprocessing**

**Missing Values:**

from sklearn.impute import SimpleImputer

*TRAIN DATASET*

imputer1 = SimpleImputer(missing\_values=np.nan, strategy='mean')

imputer1 = imputer1.fit(X\_train[:, [7, 8]])

X\_train[:, [7, 8]] = imputer1.transform(X\_train[:, [7, 8]])

imputer2 = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent')

imputer2 = imputer2.fit(X\_train[:, [0, 1, 2, 4, 9]])

X\_train[:, [0, 1, 2, 4, 9]] = imputer2.transform(X\_train[:, [0, 1, 2, 4, 9]])

*TEST DATASET*

imputer3 = SimpleImputer(missing\_values=np.nan, strategy='mean')

imputer3 = imputer3.fit(X\_test[:, [7, 8]])

X\_test[:, [7, 8]] = imputer3.transform(X\_test[:, [7, 8]])

imputer4 = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent')

imputer4 = imputer4.fit(X\_test[:, [0, 2, 4, 9]])

X\_test[:, [0, 2, 4, 9]] = imputer4.transform(X\_test[:, [0, 2, 4, 9]])

**Encoding categorical data:**

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

from sklearn.compose import ColumnTransformer

*TRAIN DATASET*

labelencoder\_X = LabelEncoder()

X\_train[:, 0] = labelencoder\_X.fit\_transform(X\_train[:, 0])

X\_train[:, 1] = labelencoder\_X.fit\_transform(X\_train[:, 1])

X\_train[:, 2] = labelencoder\_X.fit\_transform(X\_train[:, 2])

X\_train[:, 3] = labelencoder\_X.fit\_transform(X\_train[:, 3])

X\_train[:, 4] = labelencoder\_X.fit\_transform(X\_train[:, 4])

transformer = ColumnTransformer(

transformers=[

("OneHot", # Just a name

OneHotEncoder(), # The transformer class

[10] # The column(s) to be applied on.

)

],

remainder='passthrough'

)

X\_train = transformer.fit\_transform(X\_train)

X\_train = np.delete(X\_train, np.s\_[0], axis=1) # to avoid dummy trap

*TEST DATASET*

labelencoder\_X = LabelEncoder()

X\_test[:, 0] = labelencoder\_X.fit\_transform(X\_test[:, 0])

X\_test[:, 1] = labelencoder\_X.fit\_transform(X\_test[:, 1])

X\_test[:, 2] = labelencoder\_X.fit\_transform(X\_test[:, 2])

X\_test[:, 3] = labelencoder\_X.fit\_transform(X\_test[:, 3])

X\_test[:, 4] = labelencoder\_X.fit\_transform(X\_test[:, 4])

transformer2 = ColumnTransformer(

transformers=[

("OneHot", # Just a name

OneHotEncoder(), # The transformer class

[10] # The column(s) to be applied on.

)

],

remainder='passthrough'

)

X\_test = transformer2.fit\_transform(X\_test)

X\_test = np.delete(X\_test, np.s\_[0], axis=1) # to avoid dummy trap

**Feature Scaling:**

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

**Machine Learning Models Applied**

**KNN:**

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=6, metric='minkowski', p=2)

knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

**Naive Bayes:**

from sklearn.naive\_bayes import GaussianNB

naive = GaussianNB()

naive.fit(X\_train, y\_train)

pred\_naive = naive.predict(X\_test)

**Decision Tree:**

from sklearn.tree import DecisionTreeClassifier

decision = DecisionTreeClassifier(criterion='gini', random\_state=0)

decision.fit(X\_train, y\_train)

decision\_tree\_pred = decision.predict(X\_test)

**Random Forest Tree:**

from sklearn.ensemble import RandomForestClassifier

rndom = RandomForestClassifier(n\_estimators=10, criterion="gini", random\_state=0)

rndom.fit(X\_train, y\_train)

rndom\_forest\_pred = rndom.predict(X\_test)

**Logistic Regression:**

logistic = LogisticRegression(random\_state=0)

logistic.fit(X\_train, y\_train)

logistic\_pred = logistic.predict(X\_test)

**SVM:**

from sklearn.svm import SVC

svm = SVC(kernel='rbf', random\_state=0)

svm.fit(X\_train, y\_train)

predict\_svm = svm.predict(X\_test)

**Results**

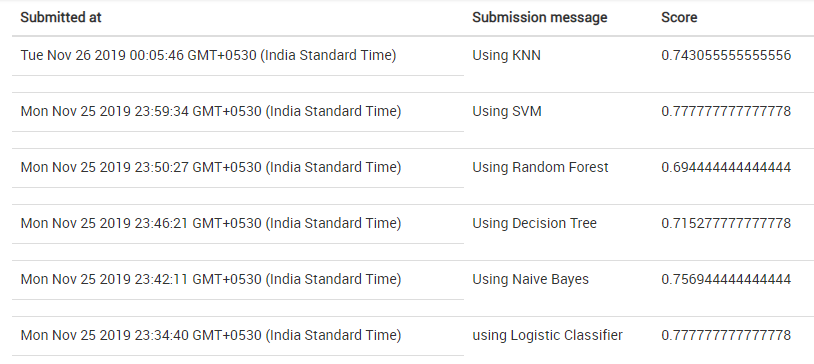
**Converting prediction’s array to DataFrame:**

dataset1 = pd.DataFrame({'Loan\_Status': y\_pred[:, ]})

**Exporting dataframe to csv file:**

export\_csv = dataset1.to\_csv(r'C:\Users\windows 10\Desktop\ML Project\knn.csv', index = None, header=True)

This is a competition problem on Analytic Vidhya. When I submitted my solutions to it my score is as follows:

****

**Conclusion From the Results Analysis**

|  |  |
| --- | --- |
| *Model* | *Accuracy* |
| KNN | 0.7430 |
| Naïve Bayes | 0.7569 |
| Decision Tree | 0.7152 |
| Random Forest | 0.6944 |
| Logistic | 0.7777 |
| SVM | 0.7777 |

As we can see SVM and Logistic Regression Classifier shows the best result. Therefore, both are the best model for this Loan Prediction Problem.

Loan prediction is a very common real-life problem that each retail bank faces at least once in its lifetime. If done correctly, it can save a lot of man hours at the end of a retail bank.

Thus classification is a fundamental machine learning problem with applications across various fields like in Image Recognition, Fraud Detection, Spam Filters in Mails, in Medicals to classify type of disease and so many more.