PROJECT REPORT FOR LOAN APPROVAL PREDICTION USING MACHINE LEARNING MODELS

Submitted in partial fulfillment for the requirement of the award of

TRAINING

IN

Data Analytics, Machine Learning and AI using Python

Submitted by:

Pragya Moondra

(moondrapragya@gmail.com)

SRM Institute of Science and Technology,Chennai

ABSTRACT

With the enhancement in the banking sector lots of people are applying for bank loans but the bank has its limited assets which it has to grant to limited people only, so finding out to whom the loan can be granted which will be a safer option for the bank is a typical process. So in this paper we try to reduce this risk factor behind selecting the safe person so as to save lots of bank efforts and assets. This is done by mining the Big Data of the previous records of the people to whom the loan was granted before and on the basis of these records/experiences the machine was trained using the machine learning model which gives the most accurate result. The main objective of this paper is to predict whether assigning the loan to a particular person will be safe or not. This paper is divided into four sections (i)Data Collection (ii) Comparison of machine learning models on collected data (iii) Training of system on most promising model (iv) Testing.

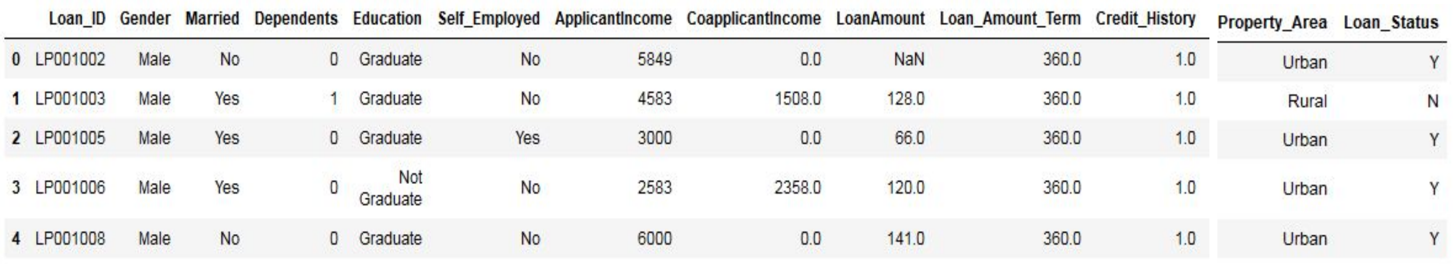
INTRODUCTION

Distribution of the loans is the core business part of almost every bank. The main portion of the bank's assets directly came from the profit earned from the loans distributed by the banks. The prime objective in the banking environment is to invest their assets in safe hands where it is. Today many banks/financial companies approve loans after a regress process of verification and validation but still there is no surety whether the chosen applicant is the deserving right applicant out of all applicants. Through this system we can predict whether that particular applicant is safe or not and the whole process of validation of features is automated by machine learning technique. The disadvantage of this model is that it emphasizes different weights to each factor but in real life sometimes loans can be approved on the basis of a single strong factor only, which is not possible through this system.Loan Prediction is very helpful for employees of banks as well as for the applicant also. The aim of this Paper is to provide a quick, immediate and easy way to choose the deserving applicants. It can provide special advantages to the bank. The Loan Prediction System can automatically calculate the weight of each feature taking part in loan processing and on new test data the same features are processed with respect to their associated weight .A time limit can be set for the applicant to check whether his/her loan can be sanctioned or not. Loan Prediction System allows jumping to specific applications so that it can be checked on priority basis. This Paper is exclusively for the managing authority of the Bank/finance company, the whole process of prediction is done privately no stakeholders would be able to alter the processing. Results against particular Loan Id can be sent to various departments of banks so that they can take appropriate action on application. This helps all others departments to carry out other formalities

ABOUT THE DATASET

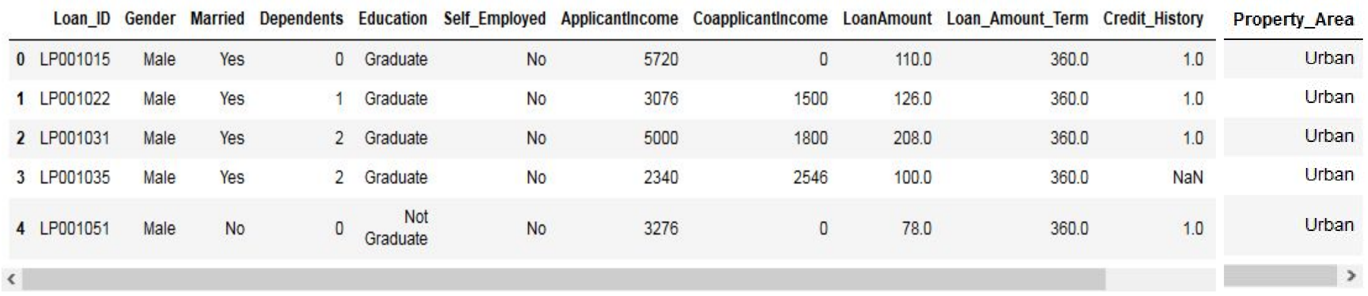
There are two csv files present in the dataset namely,the training dataset and the test dataset.Our machine learning model will be trained on the training dataset and will be tested on the testing dataset.

Training Dataset-



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

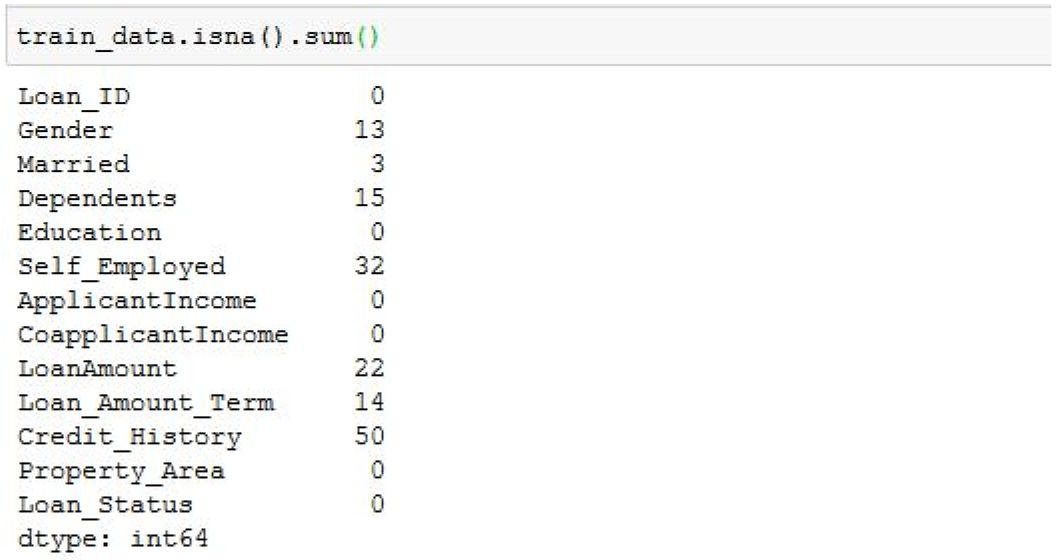
Testing Data



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

PRE-PROCESSING OF DATASET

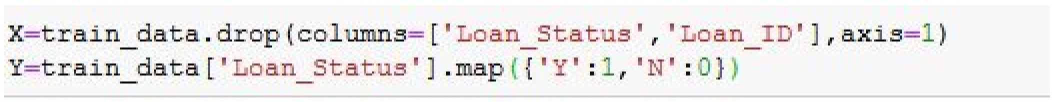
1.Check for null values:



```
train_data.isna().sum()
```

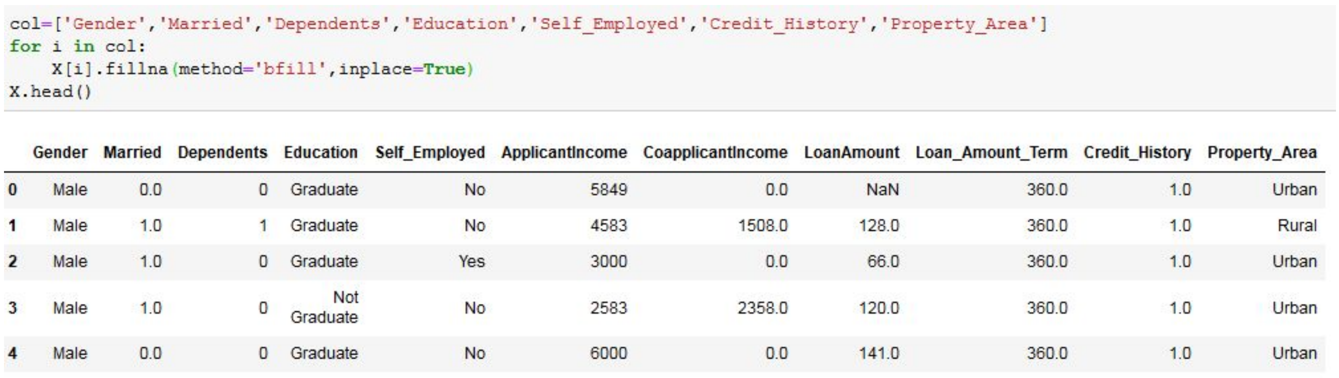
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

2.Classify into Labels and Features:



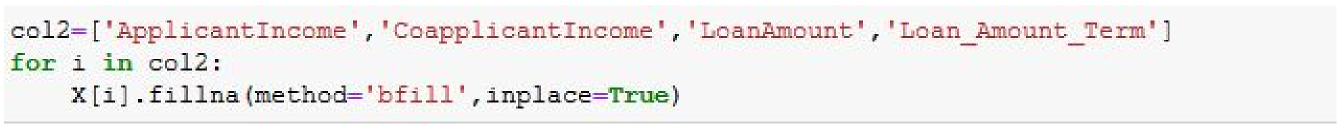
```
X=train_data.drop(columns=['Loan_Status','Loan_ID'],axis=1)
Y=train_data['Loan_Status'].map({'Y':1,'N':0})
```

3.Fill null values



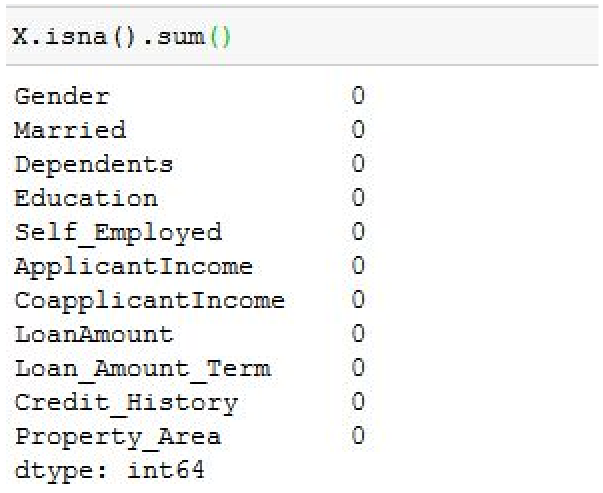
```
col=['Gender','Married','Dependents','Education','Self_Employed','Credit_History','Property_Area']
for i in col:
    X[i].fillna(method='bfill',inplace=True)
X.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	Male	0.0	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban
1	Male	1.0	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural
2	Male	1.0	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban
3	Male	1.0	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban
4	Male	0.0	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban



```
col2=['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term']
for i in col2:
    X[i].fillna(method='bfill',inplace=True)
```

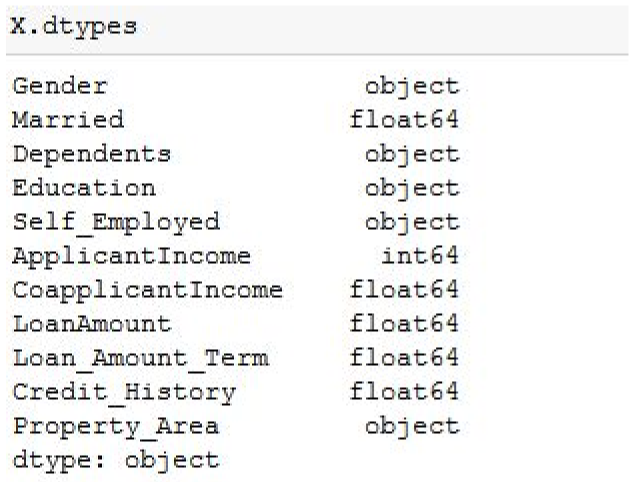
4.Re-check for null values:



```
X.isna().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
dtype: int64
```

5.Check the datatypes of features:



```
X.dtypes
```

```
Gender                object
Married              float64
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
dtype: object
```

6. Map the values:

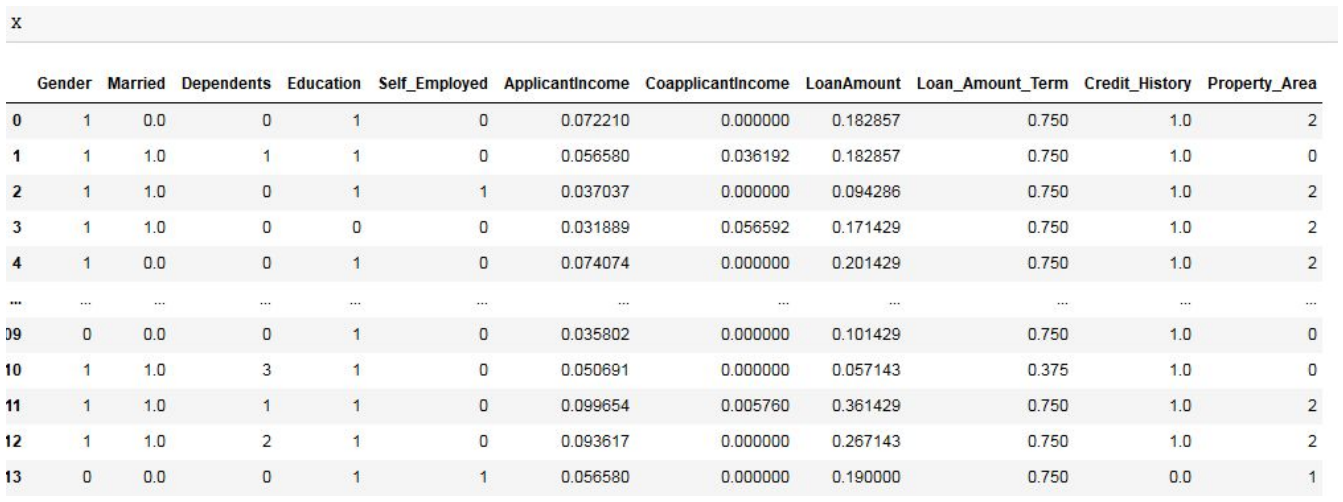
```
X['Gender']=X['Gender'].astype(str)
X['Gender']=X['Gender'].replace({'Male':1,'Female':0})
X['Education']=X['Education'].astype(str)
X['Education']=X['Education'].replace({'Graduate':1,'Not Graduate':0})
X['Self_Employed']=X['Self_Employed'].astype(str)
X['Self_Employed']=X['Self_Employed'].replace({'Yes':1,'No':0})
X['Property_Area']=X['Property_Area'].astype(str)
X['Property_Area']=X['Property_Area'].replace({'Rural':0,'Semiurban':1,'Urban':2})
```

```
X['Dependents']=X['Dependents'].replace('3+',3)
X['Dependents']=X['Dependents'].astype(int)
```

7.Normalize the values:

```
X['ApplicantIncome']=X['ApplicantIncome']/X['ApplicantIncome'].max()
X['CoapplicantIncome']=X['CoapplicantIncome']/X['CoapplicantIncome'].max()
X['Loan_Amount_Term']=X['Loan_Amount_Term']/X['Loan_Amount_Term'].max()
X['LoanAmount']=X['LoanAmount']/X['LoanAmount'].max()
```

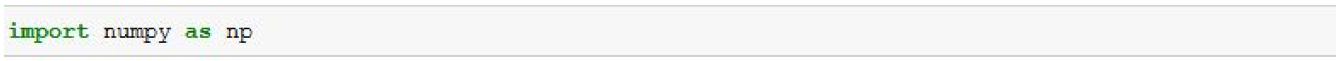
After all the processing our dataset looks like this-



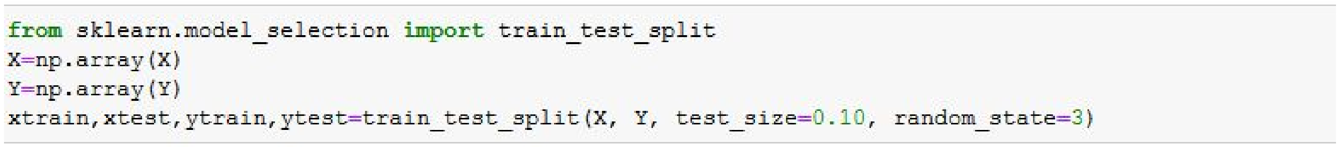
```
X
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	1	0.0	0	1	0	0.072210	0.000000	0.182857	0.750	1.0	2
1	1	1.0	1	1	0	0.056580	0.036192	0.182857	0.750	1.0	0
2	1	1.0	0	1	1	0.037037	0.000000	0.094286	0.750	1.0	2
3	1	1.0	0	0	0	0.031889	0.056592	0.171429	0.750	1.0	2
4	1	0.0	0	1	0	0.074074	0.000000	0.201429	0.750	1.0	2
...	...	...	...	...	...	...	...	...	...	...	...
09	0	0.0	0	1	0	0.035802	0.000000	0.101429	0.750	1.0	0
10	1	1.0	3	1	0	0.050691	0.000000	0.057143	0.375	1.0	0
11	1	1.0	1	1	0	0.099654	0.005760	0.361429	0.750	1.0	2
12	1	1.0	2	1	0	0.093617	0.000000	0.267143	0.750	1.0	2
13	0	0.0	0	1	1	0.056580	0.000000	0.190000	0.750	0.0	1

TRAIN THE DATA ON TRAINING DATASET



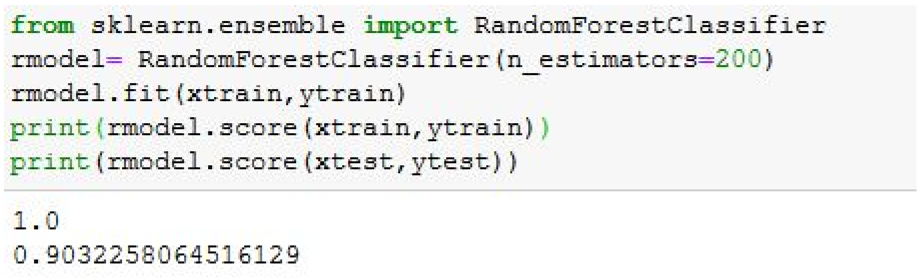
```
import numpy as np
```



```
from sklearn.model_selection import train_test_split
X=np.array(X)
Y=np.array(Y)
xtrain,xtest,ytrain,ytest=train_test_split(X, Y, test_size=0.10, random_state=3)
```

TRY DIFFERENT MODELS

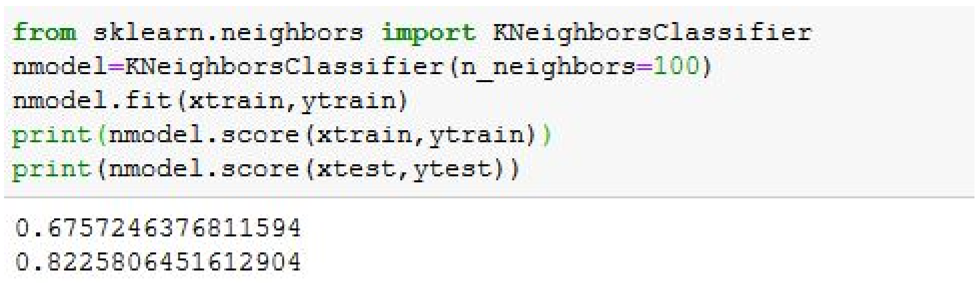
Random Forest Classifier:



```
from sklearn.ensemble import RandomForestClassifier
rmodel= RandomForestClassifier(n_estimators=200)
rmodel.fit(xtrain,ytrain)
print(rmodel.score(xtrain,ytrain))
print(rmodel.score(xtest,ytest))
```

```
1.0
0.9032258064516129
```

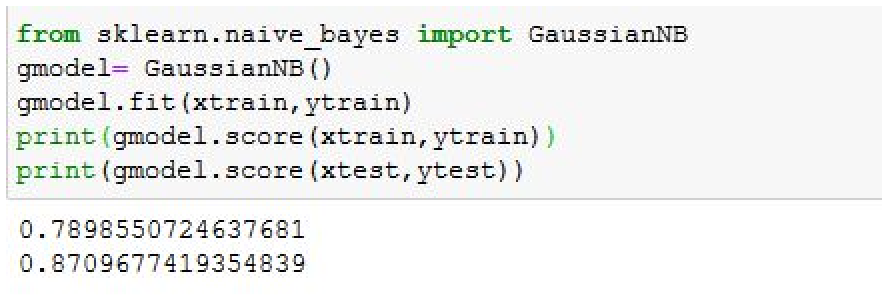
KNeighbors Classifier:



```
from sklearn.neighbors import KNeighborsClassifier
nmodel=KNeighborsClassifier(n_neighbors=100)
nmodel.fit(xtrain,ytrain)
print(nmodel.score(xtrain,ytrain))
print(nmodel.score(xtest,ytest))
```

```
0.6757246376811594
0.8225806451612904
```

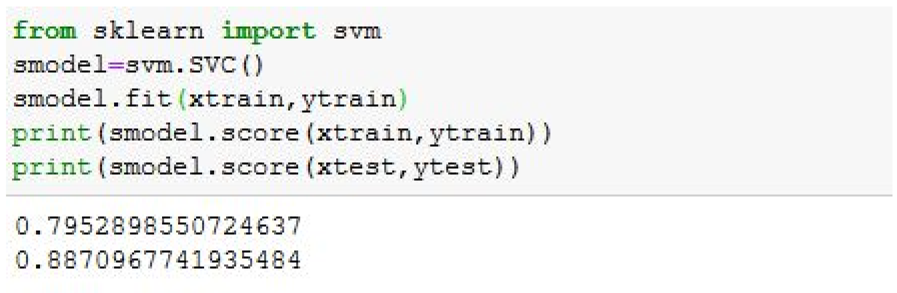
Naive Bayes:



```
from sklearn.naive_bayes import GaussianNB
gmodel= GaussianNB()
gmodel.fit(xtrain,ytrain)
print(gmodel.score(xtrain,ytrain))
print(gmodel.score(xtest,ytest))
```

```
0.7898550724637681
0.8709677419354839
```

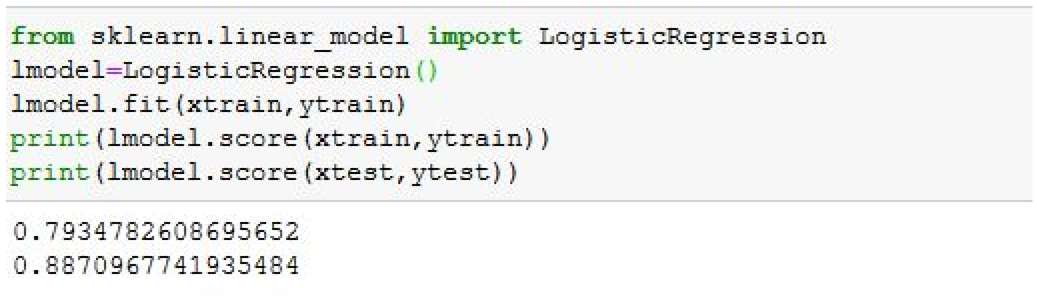
SVM:



```
from sklearn import svm
smodel=svm.SVC()
smodel.fit(xtrain,ytrain)
print(smodel.score(xtrain,ytrain))
print(smodel.score(xtest,ytest))
```

```
0.7952898550724637
0.8870967741935484
```

Logistic Regression:

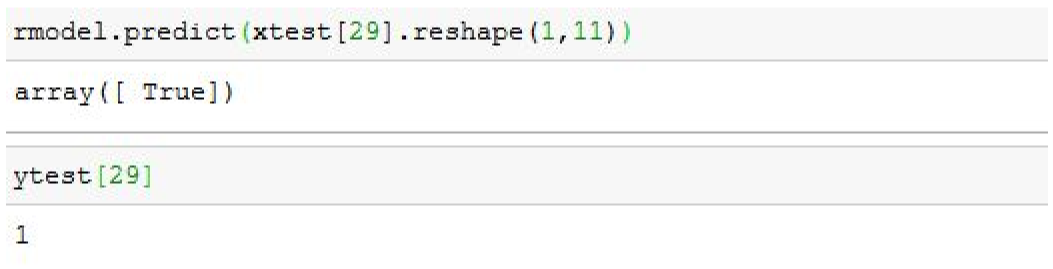


```
from sklearn.linear_model import LogisticRegression
lmodel=LogisticRegression()
lmodel.fit(xtrain,ytrain)
print(lmodel.score(xtrain,ytrain))
print(lmodel.score(xtest,ytest))
```

```
0.7934782608695652
0.8870967741935484
```

As we can observe from the above results, the most optimum model to choose is Random Forest Classifier with training accuracy as 100% and validation accuracy as 90.322%.

PREDICTING VALUES



```
rmodel.predict(xtest[29].reshape(1,11))
```

```
array([ True])
```

```
ytest[29]
```

```
1
```

CONCLUSION

From a proper analysis of positive points and constraints on the component, it can be safely concluded that the product is a highly efficient component. This application is working properly and meeting all Banker requirements. This component can be easily plugged in many other systems.There have been numbers cases of computer glitches, errors in content and most important weight of features is fixed in automated prediction system, So in the near future the so –called software could be made more secure, reliable and dynamic weight adjustment .In near future this module of prediction can be integrate with the module of automated processing system. The system is trained on old training dataset in future software that can be made such that the new testing date should also take part in training data after some fixed time.

BIBLIOGRAPHY

Dataset from-https://www.kaggle.com/altruistdelhite04/loan-prediction-problem-dataset