PREDICTION LOAN APPROVAL

INDRODUCTION:

\*This is a classification problem in which we need to classify whether the loan will be approved or not.

\*The company wants to automate the loan eligibility process based on customer detail provided while filing out outline application forms.

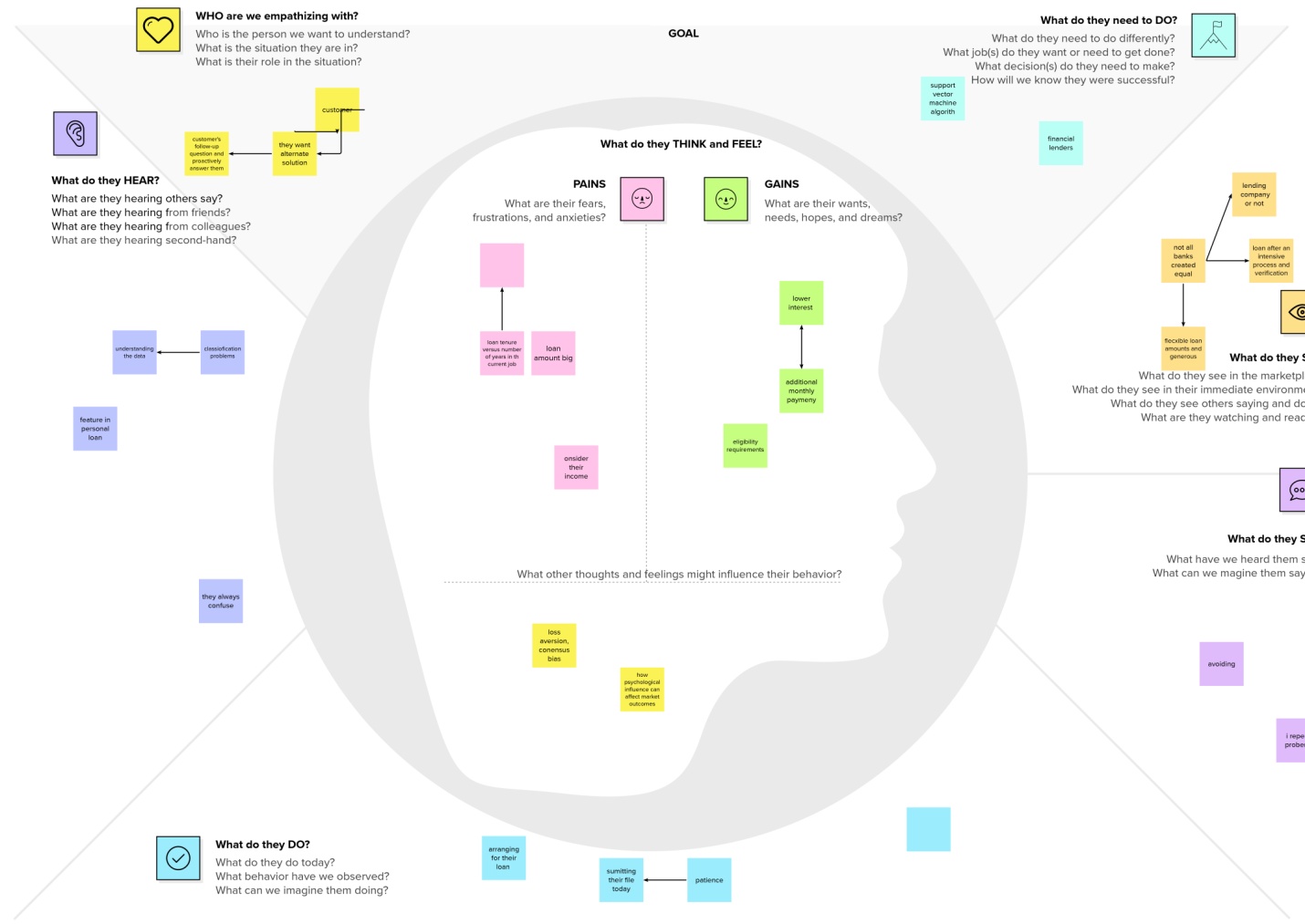
\*To automate this process, they have providd a dataset to identify the customer segments that are eligible for loan amounts so that they can specifically target these customer.

**Purpose:**

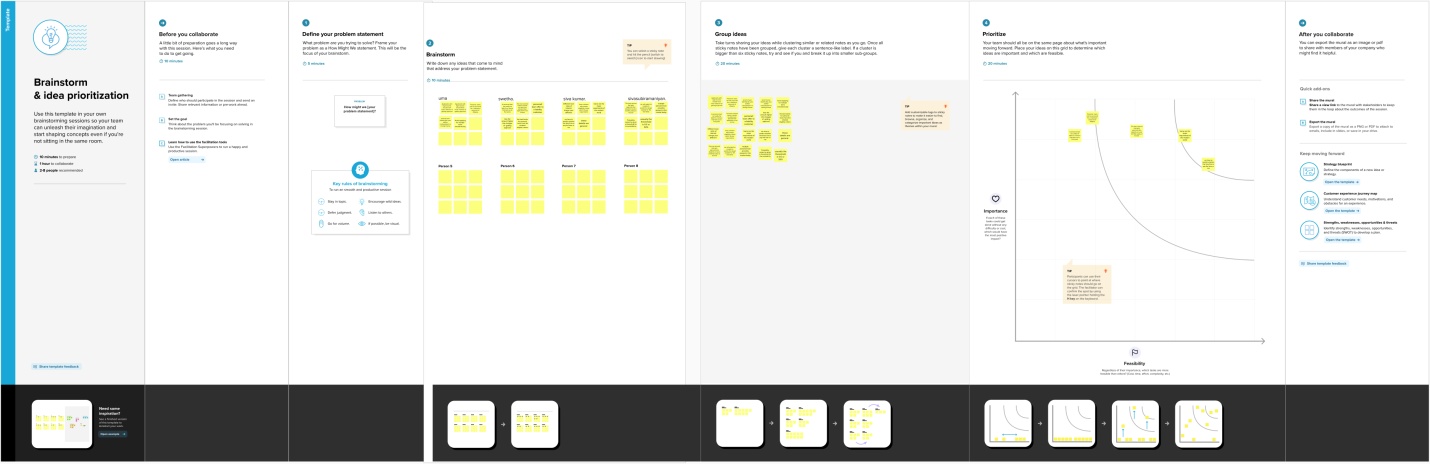
**\*The prediction model not only helps the applicant but also helps the bank by minimizing the risk and reducing the number of defaulters.**

**\*It is done by predicting if the loan can be given to the person and basis of veries parameters like credit score ,income ,age ,marital status , gender , etc.**

**Empathy Map**



**Brain Strom:**



**Advantages :**

* you to consolidate high-interest debt. ...
* You can use them to finance your wedding or dream vacation. ...
* They have predictable payment schedules. ...
* Personal loans are flexible in their uses
* They help you pay for emergency expenses without draining your savings. ...

They enable

**Low Interest Rates**: Generally, bank loans have the cheapest interest rates. The rates you pay will be cheaper than other types of high interest loans, such as venture capital. As Bizfluent says, bank loans offer significantly lower interest rates than you will find with credit cards or overdraft.

Advantages of Loan Stock  
  
**The money raised from the market does not have to be repaid**, unlike debt financing which has a definite repayment schedule. read more. In the stock, the finance business keeps shares of its own as security to secure the finance

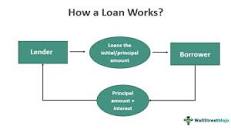
What is the advantage of loan portfolio

Portfolio lenders **focus more on cash flow and the individual's business history rather than the borrower's income and other personal metrics**. In some instances, investors may not have to provide personal tax returns if the cash flow being considered by the portfolio lender is based on rent rather than personal income.

**Disadvantages:**

**Loans are not very flexible** - you could be paying interest on funds you're not using. You could have trouble making monthly repayments if your customers don't pay you promptly, causing cashflow problems

What is the problem of loans?



A problem loan is a scenario where **borrowers fail to repay monthly loan installments**. The bank labels these loans as nonperforming assets (NPA). It can occur with either a commercial loan or a consumer loan. The loan is considered a default when borrowers miss consecutive repayments beyond the delinquency periods.

What are the disadvantages of loan prediction system?

The disadvantage of this model is that **it emphasize different weights to each factor** but in real life sometime loan can be approved on the basis of single strong factor only, which is not possible through this system. Loan Prediction is very helpful for employee of banks as well as for the applicant also

**Application:**

**Loan Prediction Project using Machine Learning in Python**

1. Understanding the various features (columns) of the dataset: ...
2. Understanding Distribution of Categorical Variables: ...
3. Outliers of LoanAmount and Applicant Income: ...
4. Data Preparation for Model Building: ...
5. Generic Classification Function: ...
6. Model Building:

\*We have data of some predicted loans from history. So when there is name of some **‘Data’** there is a lot interesting for **‘Data Scientists’.**

Introduction Loan Prediction Problem

Welcome to this article on Loan Prediction Problem. Below is a brief introduction to this topic to get you acquainted with what you will be learning.

The Objective of the Article

This article is designed for people who want to solve binary classification problems using [Python](https://www.analyticsvidhya.com/blog/2022/05/working-with-dynamodb-in-python-using-boto3/). By the end of this article, you will have the necessary skills and techniques required to solve such problems. This article provides you with sufficient theory and practice knowledge to hone you

[Download Brochure](https://blackbelt.analyticsvidhya.com/plus?utm_source=blog_india&utm_medium=banner_between_articles&utm_campaign=24-Mar-2023&utm_content=brochure)

Problem Statement

Understanding the problem statement is the first and foremost step. This would help you give an intuition of what you will face ahead of time. Let us see the problem statement.

Dream Housing Finance company deals in all home loans. They have a presence across all urban, semi-urban and rural areas. Customers first apply for a home loan after that company validates the customer’s eligibility for a loan. The company wants to automate the loan eligibility process (real-time) based on customer detail provided while filling out the 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 customer segments, that are eligible for loan amounts so that they can specifically target these customers.

It is a classification problem where we have to predict whether a loan would be approved or not. In these kinds of problems, 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, we have to predict either of the two given classes. For example: classifying the “gender” as male or female, predicting the “result” as to win or loss, 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, classifying “fruits” like oranges, apples, pears, etc.

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.

Although this course is specifically built to give you a walkthrough of the Loan Prediction problem, you can always refer to the content to get a comprehensive overview to solve a classification problem.

**Conclusion:**

… The **conclusion** derived from such assessments helps banks and other financial institutions  
… **CONCLUSION** In this paper, various algorithms were implemented to **predict** **loan** defaulters. …

 This **conclusion** follows from the first and third columns of the table, which show that … credit  
standards helped **predict** **loan** growth in both periods and that the total effect of **loan** growth on …

**Source code:**

**Loan\_prediction.ipynb**

**# -\*- coding: utf-8 -\*-**

**"""Loan Prediction.ipynb**

**Automatically generated by Colaboratory.**

**Original file is located at**

**https://colab.research.google.com/drive/14vOj9-kHfGlwsIWLtWkC1lCJhlQCSlD3**

**"""**

**# Commented out IPython magic to ensure Python compatibility.**

**import pandas as pd**

**import numpy as np**

**import pickle**

**import matplotlib.pyplot as plt**

**# %matplotlib inline**

**import seaborn as sns**

**import sklearn**

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.ensemble import GradientBoostingClassifier,RandomForestClassifier**

**from sklearn.neighbors import KNeighborsClassifier**

**from sklearn.model\_selection import RandomizedSearchCV**

**import imblearn**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.metrics import accuracy\_score,classification\_report,confusion\_matrix, f1\_score**

**# Commented out IPython magic to ensure Python compatibility.**

**from google.colab import drive**

**drive.mount('/content/drive')**

**# %cp '/content/drive/MyDrive/Colab Notebooks/loan\_prediction.csv' '/content/'**

**#importing the dataset which is in csv file**

**data = pd.read\_csv('loan\_prediction.csv')**

**data**

**data.info**

**#fining rthe sum of null values un each column**

**data.isnull().sum()**

**data['Gender'] = data['Gender'].fillna(data['Gender'].mode()[0])**

**data['Marrieed'] = data['Married'].fillna(data['Married'].mode()[0])**

**#replacing + with space for filling the non values**

**data['Dependents']=data['Dependents'].str.replace('+','')**

**data['Dependents']=data['Dependents'].fillna(data['Dependents'].mode()[0])**

**data['Self\_Employed'] = data['Self\_Employed'].fillna(data['Self\_Employed'].mode()[0])**

**data['LoanAmount'] = data['LoanAmount'].fillna(data['LoanAmount'].mode()[0])**

**data['Loan\_Amount\_Term'] = data['Loan\_Amount\_Term'].fillna(data['Loan\_Amount\_Term'].mode()[0])**

**data['Credit\_History'] = data['Credit\_History'].fillna(data['Credit\_History'].mode()[0])**

**#changing th datatype of each float column to int**

**data['Gender']=data['Gender'].astype('int64')**

**data['Married']=data['Married'].astype('int64')**

**data['Dependents']=data['Dependents'].astype('int64')**

**data['Self\_Employed']=data['Self\_Employed'].astype('int64')**

**data['CoapplicantIncome']=data['CoapplicationIncome'].astype('int64')**

**data['LoanAmount']=data['LoanAmount'].astype('int64')**

**data['Loan\_Amount\_Term']=data['Loan\_Amount\_Term'].astype('int64')**

**data['Credit\_History']=data['Credit\_History'].astype('int64')**

**#Balancing the dataset by using smote**

**from imblearn.combine import SMOTETomek**

**smote = SMOTETomek(0.90)**

**#dividing the dataset into dependent and independent y and x respectively**

**y = data['Loan\_Status']**

**x = data(columns=['Loan\_Status'],axis=1)**

**#creating a new x and y variables for the balnced set**

**x\_bal,y\_bal, = smote.fit\_resample(x,y)**

**#printing the values of y before balancing the data and after**

**print(y.value\_counts())**

**print(y\_bal.value\_counts())**

**data.describe()**

**#plotting the using displot**

**plt.figure(figsize=(12,5)**

**plt.subplot(121)**

**sns.distplot(data['ApplicantIncome'], color='r')**

**plt.subplot(122)**

**sns.distplot(data[Credit\_History'])**

**plt.show**

**#plotting the count plot**

**plt.figure(figsize=(18,4))**

**plt.subplot(1,4,1)**

**sns.countplot(data['Gender'])**

**plt.sublot(1,4,2)**

**sns.countplot(data['Education'])**

**plt.show()**

**#visualsing two colunms againist each other**

**plt.figure(figsize=(20,5))**

**plt.subplot(131)**

**sns.countplot(data['Married'],hue=data['Gender'])**

**plt.subplot(132)**

**sns.countplot(data['Self\_Employed'],hue=data['Education'])**

**plt.subplot(133)**

**sns.countplot(data['Property\_Area'],hue=data['Loan\_Amount\_Term'])**

**#visulaized based gender and income what would be the application status**

**sns.swarmplot(data['Gender'],data['ApplicantIncome'],hue = data['Loan\_Status'])**

**#perfroming feature scaling operation using standard scaller on x part of the dataset because**

**#there different type of values in the colunms**

**sc=StandardScaler()**

**x\_bal=sc.fit\_transform(x\_bal)**

**x\_bal = pd.DataFrame(x\_bal,colunms=names)**

**#splitting the dataset in train and test on balnmced dataset**

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_bal, y\_bal, test\_size=0.33, random\_state=42)**

**def decisionTree(x\_train,x\_test,y\_train,y\_test)**

**dt=DecisionTreeClassifier()**

**dt.fit(x\_train,y\_train)**

**yPred = dt.predict(x\_test)**

**print('\*\*\*DecisionTreeClassifier\*\*\*')**

**print('Confusion matrix')**

**print(confusion\_matrix(y\_test,yPred))**

**print('Classification report')**

**print(classification\_report(y\_test,ypred))**

**def randomForest(x\_train, x\_test, y\_train, y\_test):**

**rf = RandomForestClassifier()**

**rf.fit(x\_train,y\_train)**

**yperd = rf.predict(x\_test)**

**print('\*\*\*RandomForestClassifier\*\*\*')**

**print('Confusion matrix')**

**print(confusion\_matrix(y\_test,ypred))**

**print('Classification report')**

**print(classification\_report(y\_test,ypred))**

**def KNN(x\_train, x\_test, y\_train, y\_test):**

**knn = KNeighborsClassifier()**

**knn,fit(x\_train,y\_train)**

**ypred = knn.predict(x\_test)**

**print('\*\*\*KNeighborsClassifier\*\*\*')**

**print('confusion matrix')**

**print('Confusion matrix')**

**print(confusion\_matrix(y\_test,ypred))**

**print('Classification report')**

**print(classification\_report(y\_test,ypred))**

**def xgboost(x\_train, x\_test, y\_train, y\_test):**

**xg = GradientBoostingClassifier()**

**xg.fit(x\_train,y\_train)**

**ypred = xg.Predict(x\_test)**

**print('\*\*\*GradientBoostingClassifier\*\*\*')**

**print('Confusion matrix')**

**Print(confusion\_matrix(y\_test,ypred))**

**print('Classification report')'**

**print(classification\_report(y\_test,ypred)**

**#Importing the keras libraries and packages**

**import imblearn.tensorflow**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import Dense**

**#Initialising the ANN**

**classifier = Sequential()**

**#Adding the input layer and the first hidden layer**

**classifier.add(Dense(units=100, activation='relu', input\_dim=11))**

**#Adding the second hidden layer**

**classifier.add(Dense(units=50,activation='relu'))**

**#Adding the output layer**

**classifier.add(Dense(units=1,activation='sigmoid'))**

**#compiling the ANN**

**classifier.compile(optimizer='adam',loss='binary'\_crossentropy',metrics=['accuracy'])**

**#Fitting the ANN to the Training set**

**model\_hostory = classifier.fit(x\_train, y\_train, batch\_size=100,validation\_split=0.2,epochs=100)**

**#Gender Married Dependents Education Self\_Employed Application CoapplicantIncome LoanAmount Loan\_Amount\_Term Credit\_History Property\_Area**

**dtr.predict([[1,1,0,1,1,4276,1542,145,240,0,1]])**

**#Gender Married Dependents Education Self\_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan\_Amount\_Term Credit\_History Property\_Area**

**rfr.Predict([[1,1,0,1,1,4276,1542,145,240,0,1]])**

**#Gender Married Dependents Education Self\_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan\_Amount\_Term Credit\_History property\_Area**

**knn.predict([[1,1,0,1,1,4276,145,240,0,1]])**

**#Gender Married Dependents Education Self\_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan\_Amount\_Term Credit\_History property\_Area**

**xgb.predict([[1,1,0,1,1,4276,145,240,0,1]])**

**# Commented out IPython magic to ensure Python compatibility.**

**classifier.save("loan.h5")**

**# %cp '/content/loan.h5'**

**#predicting the Test set rsults**

**y\_pred = classifier.predict(x\_test)**

**y\_pred**

**y\_pred = (y\_pred > 0.5)**

**y\_pred**

**def predit\_exit(sample\_value):**

**sample\_value = np.array(sample\_value)**

**sample\_value = sample\_value.reshape(1,-1)**

**sample\_value = sc.transform(sample\_data)**

**return classifier.predict(sample\_value)**

**sample\_value = [[1,1,0,1,1,4276,1542,145,240,0,1]]**

**if predict\_exit(sample\_values)>0.5:**

**print('prediction:High chance of Loan Approval!')**

**else:**

**print('prediction:Low chance Loan Approval.')**

**sample\_value=[[1,0,1,1,45,14,45,240,1,1]]**

**if predict\_exit(sample\_value)>0.5:**

**print('prediction:High chance of Loan Approval!')**

**else:**

**print('prediction:Low chance of Loan Approval.')**

**def compareModel(x\_train,x\_test,y\_train,y\_test):**

**decisionTree(x\_train,x\_test,y\_train,y\_test)**

**print('\_'\*100)**

**RandomForest(x\_train,x\_test,y\_train,y\_test)**

**print('\_'\*100)**

**xGB(x\_train,\_test,y\_train,y\_test)**

**print('\_'\*100)**

**KNN(x\_train.x\_test,y\_train,y\_test)**

**# print('\_'\*100\_)**

**#compareModel(x\_train,x\_test,y\_train,y\_test\_)**

**#ypred = classifier.predict(x\_test)**

**#print(accuracy\_score(y\_pred,y\_test))**

**#print("ANN Model")**

**#print("Confusion\_Matrix")**

**#print(confusion\_matrix(y\_test,y\_pred))**

**#print("Classification Report")**

**#print(classification\_report(y\_test,y\_pred))**

**from sklearn.model\_selection import cross\_val\_score**

**#rf = RandomForestClassifier()**

**#rf.fit(x\_train,y\_train)**

**#ypred = rf.predict(x\_test)**

**#f1\_score(ypred,y\_test,average='weighted')**

**#cv = cross\_val\_score(rf,x,y,cv=5)**

**#np.mean(cv)**

**#saving the model by using pickle funtion**

**#pickle.dump(model,open('rdf.pkl','wb'))**

**app.py**

**# -\*- coding: utf-8 -\*-**

**"""app.py**

**Automatically generated by Colaboratory.**

**Original file is located at**

**https://colab.research.google.com/drive/1cKrF6VhiLVh1wOgorY67UX4kFHZkTFys**

**"""**

**from flask import Flask, render\_template,request**

**import numpy as np**

**import pickle**

**!pip install pyngrok**

**from pyngrok import ngrok**

**# Commented out IPython magic to ensure Python compatibility.**

**from google.colab import drive**

**drive.mount('/content/drive')**

**# %cp -r '/content/drive/MyDrive/predicting personal loan approval using machine learning/Flask/templates/' '/content/'**

**app = Flask(\_\_name\_\_)**

**ngrok.set\_auth\_token("2OGw99kjZxSOAJbHZyWvRwcBw4U\_6ixdhHh61sGML3gSQT91K")**

**#model = pickle.load(open(r'rdf.pkl', 'rb'))**

**#scale = pickle.load(open(r'scale.pkl', 'rb'))**

**public\_url = ngrok.connect(5000)**

**print(public\_url)**

**@app.route('/') #rendering the html template**

**def home():**

**return render\_template('home.html')**

**@app.route('/submit',methods=["POST","GET"])#route to show the prediction in a web UI**

**def submit():**

**# reading the inputs given by the user**

**input\_faeature=[int(x)for x in request.form.values() ]**

**#input\_feature = np.transpose(input.feature)**

**input\_feature-[np.arry(input\_feature)]**

**print(input\_feature)**

**names = ['Gender', 'Married', 'Dependents', 'Education', 'Self\_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'Loan\_Amount\_Term', 'Credit\_History', 'Property\_Area']**

**data = pandas.DataFrame(input\_feature,columns=names)**

**print(data)**

**#data\_scaled = scale.fit\_transform(data)**

**#data = pandas.DataFrame(,columns=names)**

**#predictions using the loaded model file**

**prediction=mode.predict(data)**

**print(prediction)**

**prediction = int(prediction)**

**print(type(prediction))**

**if(prediction == 0):**

**return render\_template("output.html",result = "Loan will Not be Approved")**

**else:**

**return render\_template("output.html",result = "Loan will be Aproved")**

**app.run(debug=False)**

**Result:**

