CHIKKANNA GOVERNMENT ARTS COLLEGE

DEPARTMENT OF BACHELOR OF COMPUTER APPLICATION

TIRUPUR-641602

(AFFILIATED TO BHARATHIAR UNIVERSITY)

TEAM MEMBERS NAME :

MERCY K (2022J0005)

SABITHA M(2022J0008)

YOGANA L (2022J0009)

ARUN KUMAR M (2022J0016)

PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING

1.INTRODUCTION

1.1 OVERVIEW :

A loan is a sum of money that is borrowed and repaid over a period of time, typically with interest. There are various types of loans available to individuals and businesses, such as personal loans, mortgages, auto loans, student loans, business loans and many more. They are offered by banks, credit unions, and other financial institutions, and the terms of the loan, such as interest rate, repayment period, and fees, vary depending on the lender and the type of loan.

A personal loan is a type of unsecured loan that can be used for a variety of expenses such as home repairs, medical expenses, debt consolidation, and more. The loan amount, interest rate, and repayment period vary depending on the lender and the borrower's creditworthiness.To qualify for a personal loan, borrowers typically need to provide proof of income and have a good credit score.

Predicting personal loan approval using machine learning analyses a borrower's financial data and credit history to determine the likelihood of loan approval. This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny.

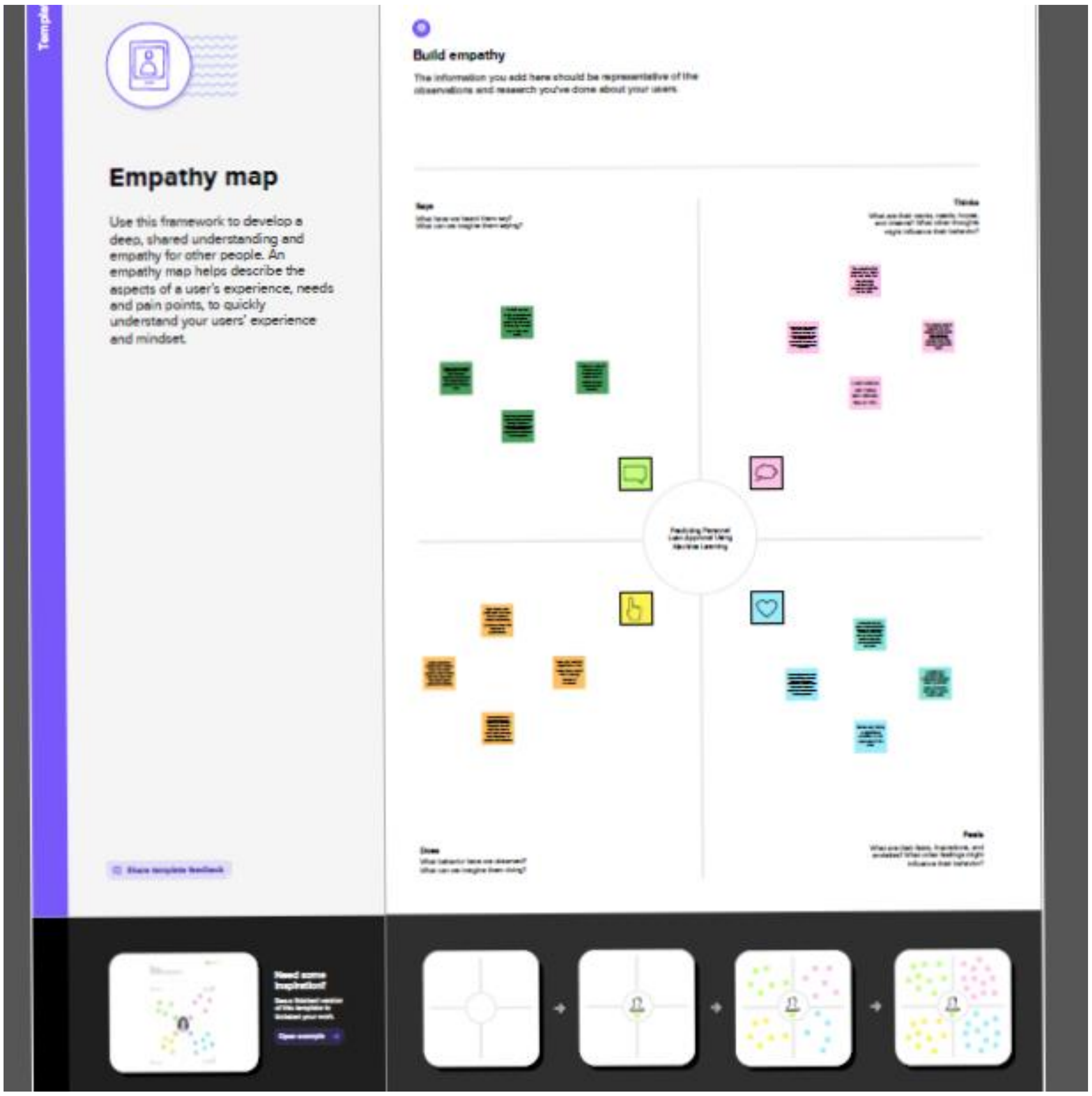
1.2. PURPOSE :

Predictive modeling is used to analyze the data and predict the outcome. Predictive modeling used to predict the unknown event which may occur in the future. In this process, we are going to create, test and validate the model. There are different methods in predictive modeling. They are learning, artificial intelligence and statistics. Once we create a model, we can use many times, to determine the probability of outcomes. So, predict model is reusable. Historical data is used to train an algorithm. The predictive modeling process is an iterative process and often involves training the model, using multiple models on the same dataset.

- **Creating the model:** To create a model to run one or more algorithms on the data set.
- **Testing a model:** The testing is done on past data to see how the best model predicts
- **Validating a model:** Using visualization tools tovalidate the model.
- **Evaluating model:** Evaluating the best fit model from the models used and choosing the model right fitted for the data.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1. EMPATHY MAP



Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Build empathy

The information you add here should be representative of the observations and research you've done about your users.

Says
What have we heard them say?
What can we imagine them saying?

Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

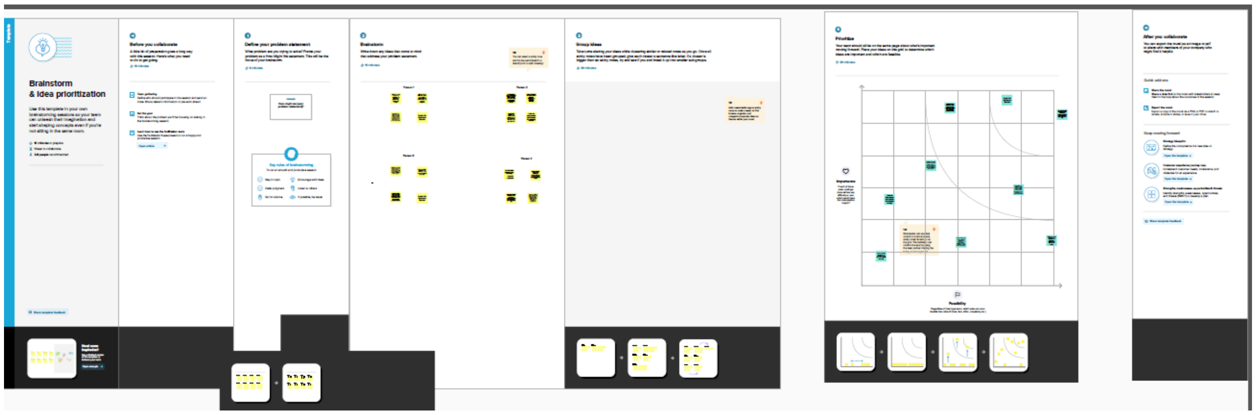
Predicting Personal Loan Approval Using Machine Learning

Does
What behavior have we observed?
What can we imagine them doing?

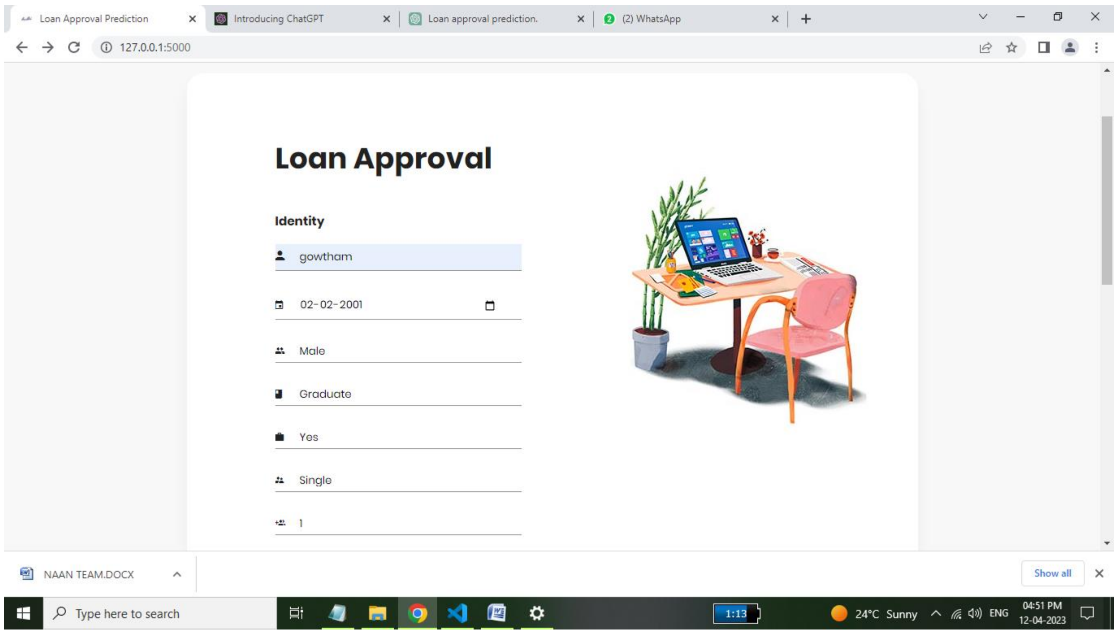
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

Share template feedback
Need some inspiration?
See a finished version of this template to kickstart your work.
Open example

2.2 IDEATION & BRAINSTORMINGS MAP


Brainstorm
& idea prioritization
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.
Before you collaborate
Define your problem statement
Brainstorm
Group ideas
Prioritize
After you collaborate

3. RESULT



Loan Approval

Identity

gowtham

02-02-2001

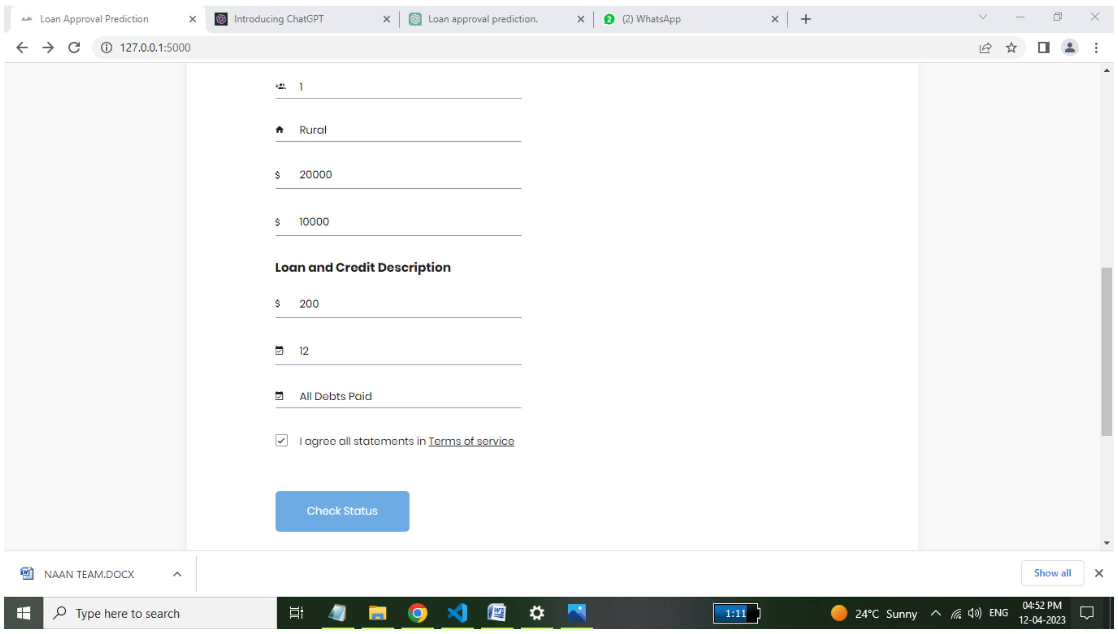
Male

Graduate

Yes

Single

1



1

Rural

20000

10000

Loan and Credit Description

200

12

All Debts Paid

☑ I agree all statements in Terms of service

4.ADVANTAGES & DISADVANTAGES

ADVANTAGES :

Accuracy—one of the primary benefits of using machine learning for credit scoring is its accuracy. Unlike human manual processing, ML-based models are automated and less likely to make mistakes. This means that loan processing becomes not only faster but more accurate, too, cutting costs on the whole.

DISADVANTAGES :

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.

5. APPLICATION

A Prediction Model uses data mining, statistics and probability to forecast an outcome. Every model has some variables known as predictors that are likely to influence future results. The data that was collected from various resources then a statistical model is made. It can use a simple linear equation or a sophisticated neural network mapped using a complex software. As more data becomes available the model becomes more refined and the error decreases meaning then it'll be able to predict with the least risk and consuming as less time as it can. The Prediction Model helps the banks by minimizing the risk associated with the loan approval system and helps the applicant by decreasing the time taken in the process.

The main objective of the Project is to compare the Loan Prediction Models made implemented using various algorithms and choose the best one out of them that can shorten the loan approval time and decrease the risk associated with it. It is done by predicting if the loan can be given to that person on the basis of various parameters like credit score, income, age, marital status, gender, etc. The prediction model not only helps the applicant but also helps the bank by minimizing the risk and reducing the number of defaulters.

In the present scenario, a loan needs to be approved manually by a representative of the bank which means that person will be responsible for whether the person is eligible for the loan or not and also calculating the risk associated with it. As it is done by a human it is a time consuming process and is susceptible to errors. If the loan is not repaid, then it accounts as a loss to the bank and banks earn most of their profits by the interest paid to them. If the banks lose too much money, then it will result in a banking crisis. These banking crisis affects the economy of the country.

So it is very important that the loan should be approved with the least amount of error in risk calculation while taking up as the least time possible. So a loan prediction model is required that can predict quickly whether the loan can be passed or not with the least amount of risk possible.

6. CONCLUSION

From the proper view of analysis this system can be used perfect for detection of clients who are eligible for approval of loan. The software is working perfect and can be used for all banking requirements. This system can be easily uploaded in any operating system. Since the technology is moving towards online, this system has more scope for the upcoming days. This system is more secure and reliable. Since we have used Random Forest Algorithm the system returns very accurate results. There is no issue if there are many no of customers applying for loan. This system accepts data for N no. of customers. In future we can add more algorithms to this system for getting more accurate results.

7. FUTURE SCOPE

The system is trained on old training dataset in future software can be made such that new testing data should also take part in training data after some fix time. In future this model can be used to compare various machine learning algorithm generated prediction models and the model which will give higher accuracy will be chosen as the prediction model.

8. APPENDIX

SOURCE CODE :

LIBRARIES:

```
 import numpy as np
import pandas as pd
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, RandomF
orestClassifier
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_repor
t, confusion_matrix, f1_score



from google.colab import drive
drive.mount('/content/drive')



data = pd.read_csv('/content/drive/MyDrive/loan_dataset_Nm.csv')
data





#dropping the loan id column because there is no use it for the
model building
data.drop(['Loan_ID'],axis=1,inplace=True)
```

Data Preprocessing

Handling Categorical Values

```
data['Gender']=data['Gender'].map({'Female':1,'Male':0})
data.head()

data['Property_Area']=data['Property_Area'].map({'Urban':2,'Semi
urban':1,'Rural':0})
data.head()

data['Married']=data['Married'].map({'Yes':1,'No':0})
data.head()

data['Education']=data['Education'].map({'Graduate':1,'Not Gradu
ate':0})
data.head()

data['Self_Employed']=data['Self_Employed'].map({'Yes':1,'No':0}
)
data.head()

data['Loan_Status']=data['Loan_Status'].map({'Y':1,'N':0})
data.head()

data.isnull().sum()

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

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

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['Cre
dit_History'].mode()[0])

data.isnull().sum()

data.info()


#changing the 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['Coapplicant']=data['CoapplicantIncome'].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')

data.info()
```

Data Visualization

Univariate analysis

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(data['ApplicantIncome'], color='r')
plt.subplot(122)
sns.displot(data['Credit_History'])
plt.show()

#ploting the count plot
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(data['Gender'])
plt.subplot(1,4,2)
sns.countplot(data['Education'])
plt.show()
```

Bivariate analysis

```
#visualising two columns againist each other
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(data['Married'], x=data['Gender'])
plt.subplot(132)
sns.countplot(data['Self_Employed'], x=data['Education'])
plt.subplot(133)
sns.countplot(data['Property_Area'], x=data['Loan_Amount_Term'])


#plotted a column using cross tab function
pd.crosstab(data['Gender'],[data['Self_Employed']])
```

Multi variate analysis

```
#visualised based gender and income what would be the applicanti
on status
sns.swarmplot(x='Gender', y='ApplicantIncome', hue = 'Loan_Statu
s',data=data)
```

Balancing the Dataset

```
#Balancing the dataset by using smote
from imblearn.combine import SMOTETomek

smote = SMOTETomek()

#dividing the dataset into dependent and independent y and x res
pectively
y = data['Loan_Status']
x = data.drop(columns=['Loan_Status'],axis=1)

#shape of x after seperating from the total dataset
x.shape

#shape of y
y.shape

#creating a new x and y variables for the balanced 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())

names = x_bal.columns
```

Scalling the dataset

```
#performing features scaling operation using standard scaller on
 x part of the dataset because
#there different type of values in the columns
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)

x_bal

x_bal = pd.DataFrame(x_bal,columns=names)
x_bal.head()

#spliting the dataset in train and test on balanced dataset
X_train, X_test, y_train, y_test = train_test_split(x_bal, y_bal
, test_size=0.33, random_state=42)

X_train.shape

X_test.shape

y_train.shape, y_test.shape
```

Model building

RANDOM FOREST MODEL

```
#importing & building the random forest model
def RandomForest(X_train,X_test,y_train,y_test):
  model = RandomForestClassifier()
  model.fit(X_train,y_train)
  y_tr = model.predict(X_train)
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(X_test)
  print(accuracy_score(yPred,y_test))


#printing the train accuracy and test accuracy respectively
RandomForest(X_train,X_test,y_train,y_test)
```

```
#importing the building the Decision tree model
def DecisionTree(X_train,X_test,y_train,y_test):
  model = DecisionTreeClassifier()
  model.fit(X_train,y_train)
  y_tr = model.predict(X_train)
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(X_test)
  print(accuracy_score(yPred,y_test))

#printing the train accuracy and test accuracy respectively
DecisionTree(X_train,X_test,y_train,y_test)
```

KNN MODEL

```
#importing and building the KNN model
def KNN(X_train,X_test,y_train,y_test):
  model = KNeighborsClassifier()
  model.fit(X_train,y_train)
  y_tr = model.predict(X_train)
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(X_test)
  print(accuracy_score(yPred,y_test))

#printing the train accuracy and test accuracy respectively
KNN(X_train,X_test,y_train,y_test)

#importing and building the Xg boost model
def XGB(X_train,X_test,y_train,y_test):
  model = GradientBoostingClassifier()
  model.fit(X_train,y_train)
  y_tr = model.predict(X_train)
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(X_test)
  print(accuracy_score(yPred,y_test))

#printing the train accuracy and test accuracy respectively
XGB(X_train,X_test,y_train,y_test)
```

Hyper parameter tuning

```
rf = RandomForestClassifier()
#giving some parameters that can be used in randized search cv
parameters = {
                'n_estimators' : [1,20,30,55,68,74,90,120,115],
                 'criterion':['gini','entropy'],
      'max_depth' : [2,5,8,10], 'verbose' : [1,2,3,4,6,8,9,10]


#performing the randomized cv
RCV = RandomizedSearchCV(estimator=rf,param_distributions=parame
ters,cv=10,n_iter=4)

RCV.fit(X_train,y_train)

#getting the best parameter from the giving list and best score
from them
bt_params = RCV.best_params_
bt_score = RCV.best_score_

bt_params

bt_score

#training and test the xg boost model on the best parameters for
 from the randomized cv
def RandomForest(X_train,X_test,y_train,y_test):
  model = RandomForestClassifier(verbose= 2 , n_estimators= 68,
max_features= 'log2',max_depth=5 ,criterion='entropy')
  model.fit(X_train,y_train)
  y_tr = model.predict(X_train)
  print("Training Accuracy")
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(X_test)
  print('Testing Accuracy')
  print(accuracy_score(yPred,y_test))

model = RandomForestClassifier(verbose= 2 , n_estimators= 68, ma
x_features= 'log2',max_depth=5 ,criterion='entropy')
model.fit(X_train,y_train)


#printing the train and test accuracy after hyper parameter tuni
ng
RandomForest(X_train,X_test,y_train,y_test)
```

Saving the model

```
#saving the model by using pickle function
pickle.dump(model,open('rdf.pkl','wb'))
```