**PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING**

**1.Introduction**

**1.1 Overview**

**1.2 purpose**

**2.Problem Definition & Design Thinking**

**2.1 Empathy Map**

**2.2 Ideation & brainstorming map screenshot**

**3.Result**

**4.Advantages & Disadvantages**

**5.Applications**

**6.Conclusion**

**7.Future scope**

**8.Appendix**

**A. Source Code**

**1.INTRODUCTION**

* 1. **Overview**

Finance companies deal with all kinds of loans such as house loans , vehicle loans, educational loans ,personal loans…And has a presence across areas such as cities ,towns and village areas.

A customer first requests for a loan and after that finance company validates the customer eligibility for the loan and of approve .

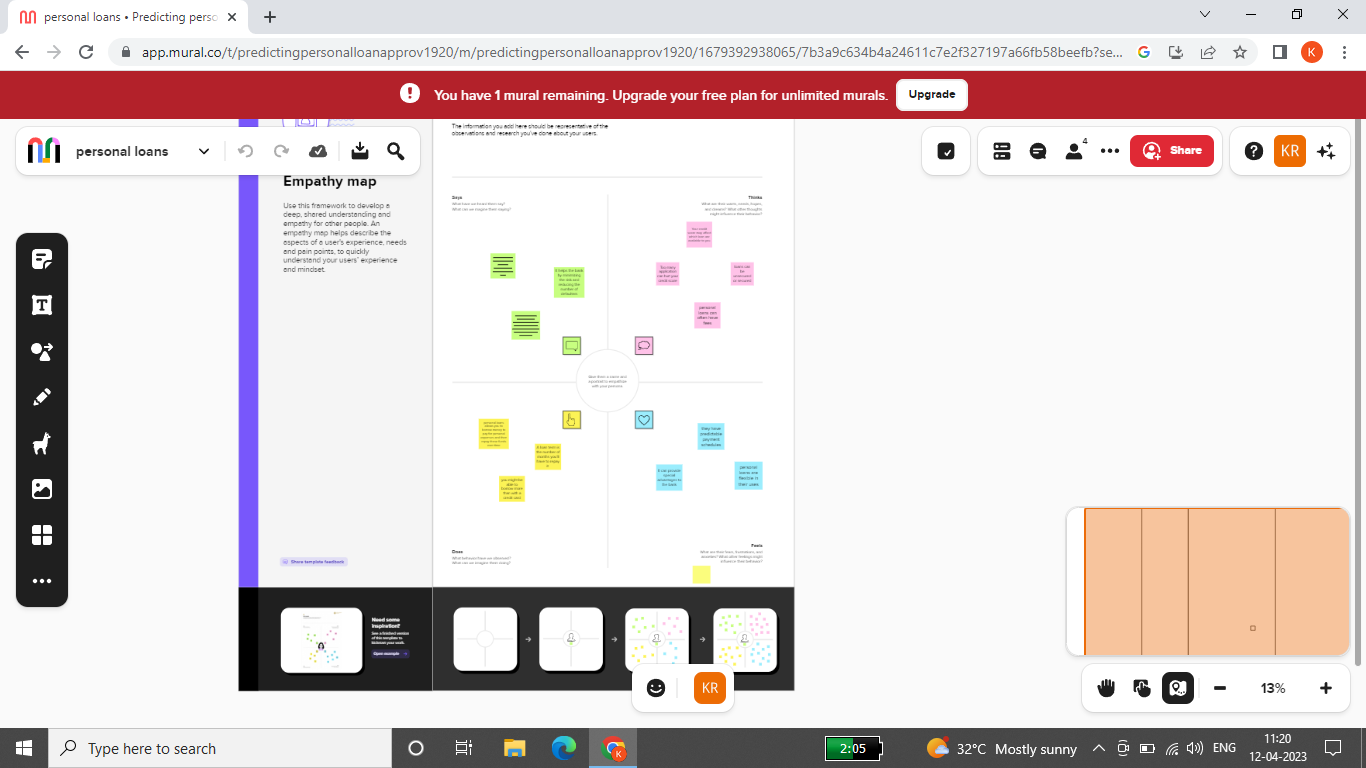
Details like marital status , gender ,education ,and number of dependents ,income ,loan amount credit history ,and others are given in the form to fill up by the applicants .

**1.2.purpose**

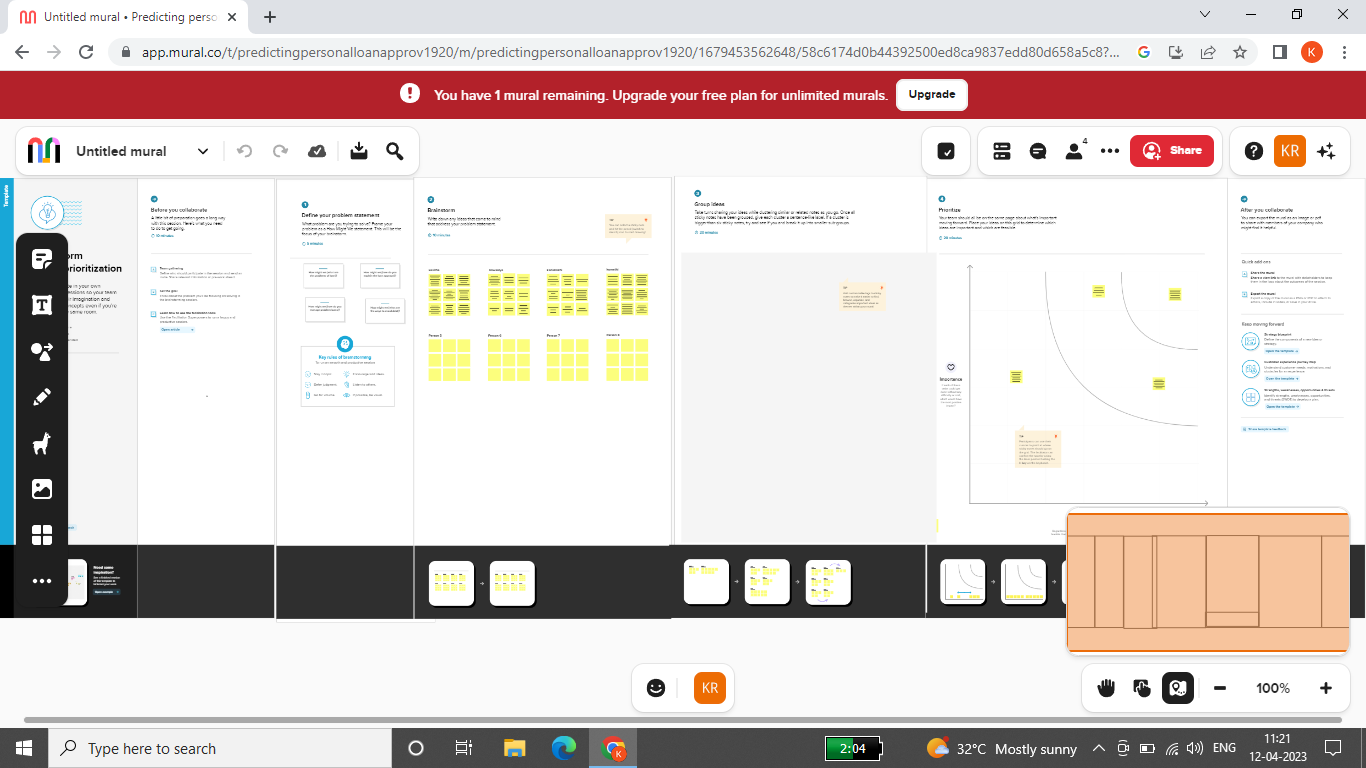
Therefore ,a robust model is built taking those details as input to verify whether an applicant is eligible to apply for loan or not .The target variable here is applicants ”Loan Status” and the other variables are predictors .After building the machine learning model a web application is to be developed for a user interface that allows the user to see instantly if he/she is eligible to get a loan by entering the given details.

**2.PROBLEM DEFINITION & DESIGN THINKING**

**2.1.Empathy map:**

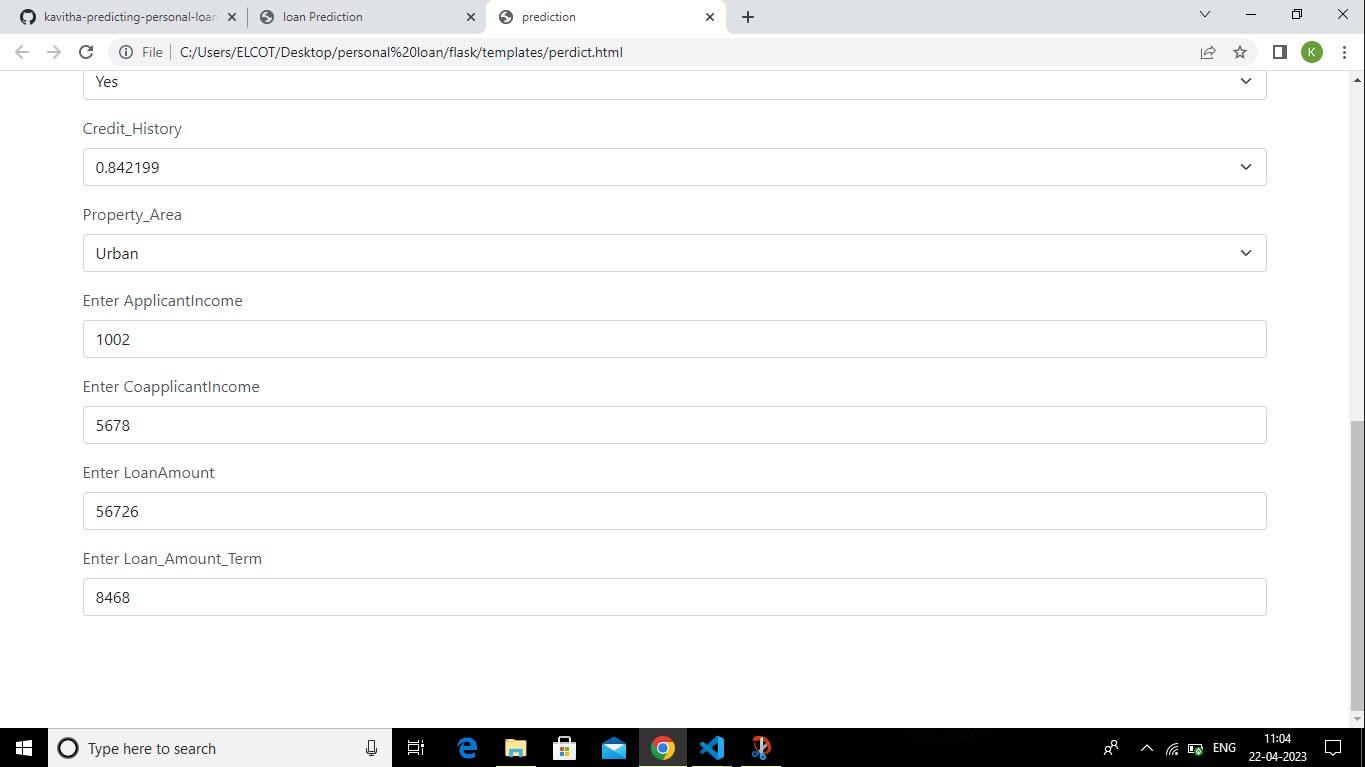
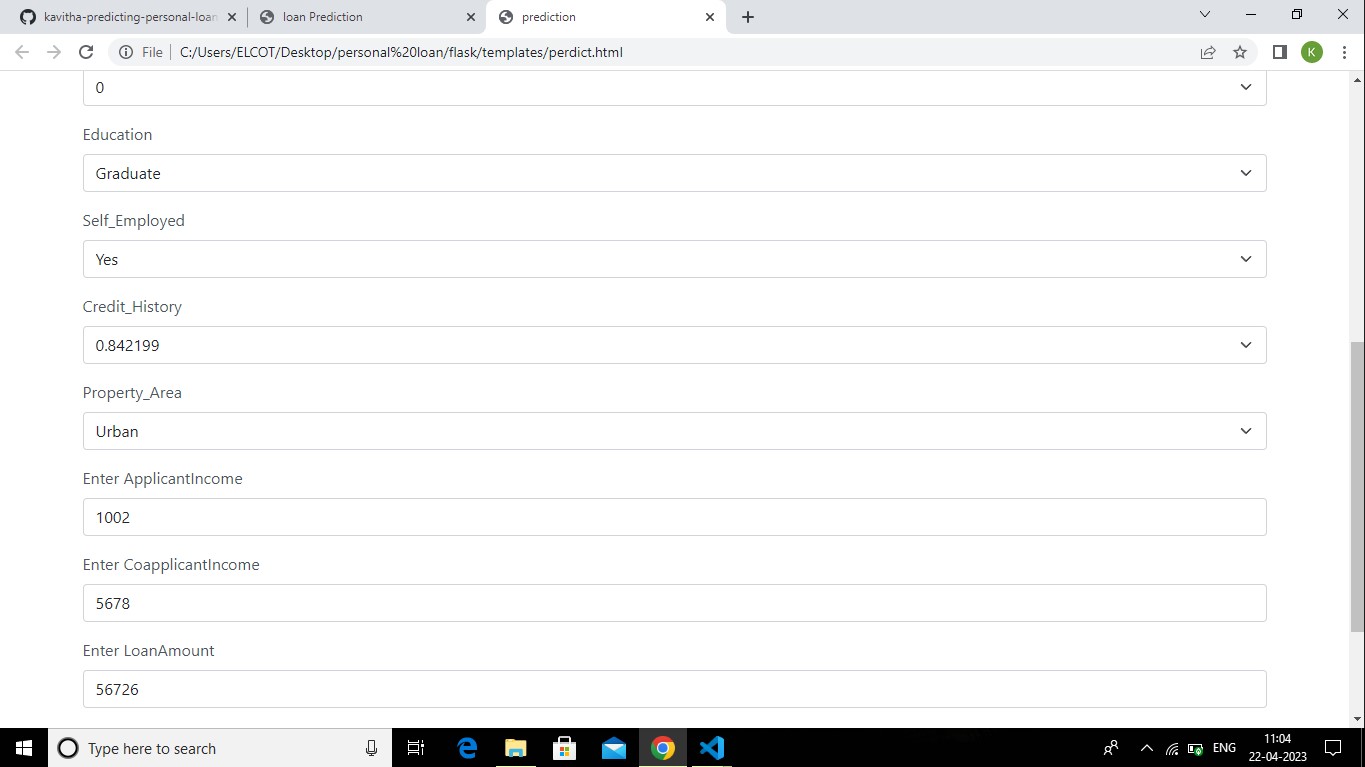
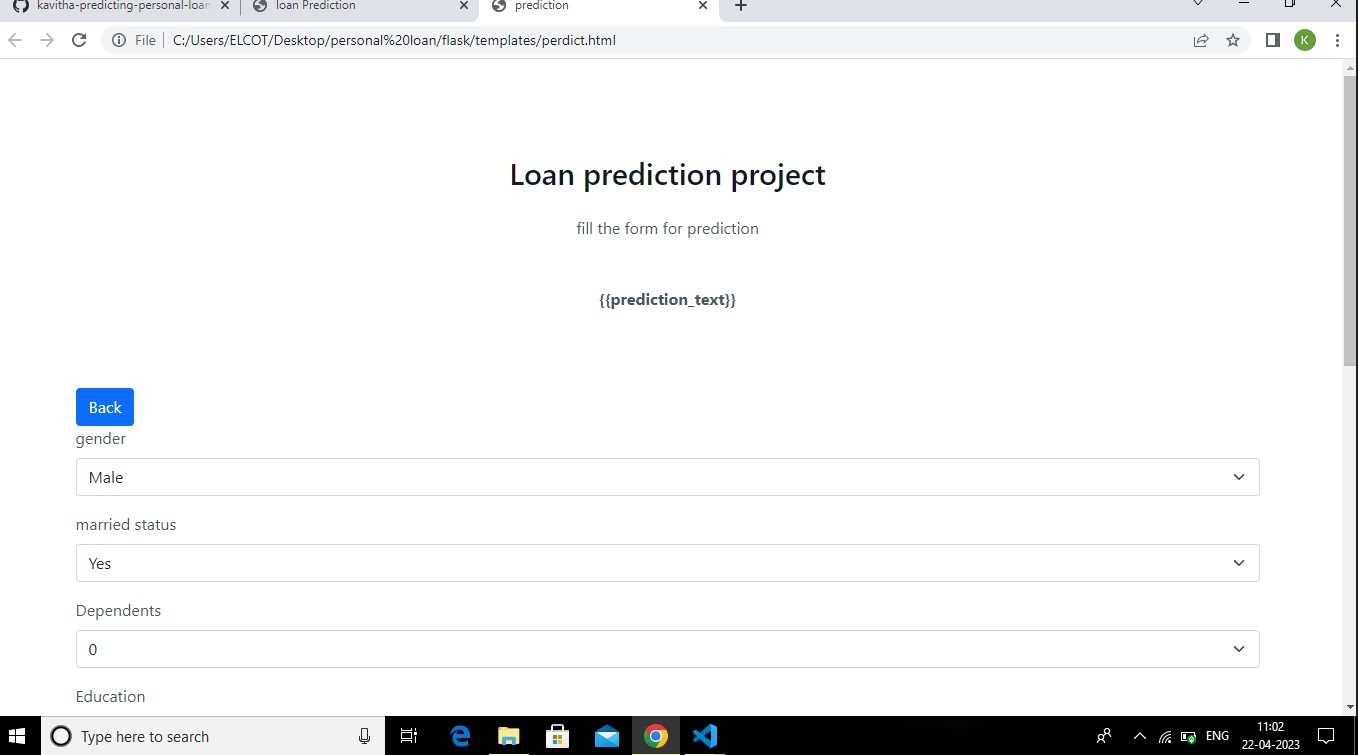


**2.2.Ideation and brainstorming map:**



**3.RESULT:**





**4.ADVANTAGES & Disadvantage**

* Spread the cost of a significant purchase safely
* Can help you manage your personal finances
* Ideal if you have struggled to save in the past
* Unsecured loans are not tied to assets
* Loan term commitment
* Good product requires a good credit score
* Certain loan types are riskier than others
* Will never get 0% interest unlike a credit card or finance deal

**5.APPLICATION**

* CASHe Personal Loan App
* Instant Personal Loan App
* PaySense
* NIRA Instant Personal Loan App
* Finnable
* Personal Loan App-RapidRupee
* Stashfin- Credit line & Loans
* Fullerton India InstaLoan
* mPokket
* Hero FinCorp Personal Loan App

**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.

**7.FUTURE SCOPE:**

* Today fast growing IT sector requires the development of new technology and the updating of existing technology that allows us to eliminate human interference and boost job productivity.
* This model is used for the banking system or anyone who wants to apply for a loan. based on the examination of the data, it is apparent that it reduces all frauds committed during the loan approval process.
* Time is valuable to everyone, and by doing so, not only the bank, but also the applicants waiting time will be reduced.
* cleaning and processing of data, imputation of missing values, experimental analysis of data set ,model construction ,and testing on test data are all steps in the prediction process, The best case accuracy attained on the original data set is 0.9189 on data set.

**APPENDIX**

1. **Source code:**

importing necessary libraries

import pandas as pd

import numpy as py

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

\*\*Importing the dataset\*\*

data = pd.read\_csv('/content/train\_u6lujuX\_CVtuZ9i.csv')

data

data.drop(['Loan\_ID'],axis=1,inplace=True)

data.head()

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

data.head()

data['Property\_Area']=data['Property\_Area'].map({'Urban':2,'Semiurban':1,'Rural':0})

data.head()

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

data.head()

data['Education']=data['Education'].map({'Graduate':1,'Not Graduate':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['Married'].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['Credit\_History'].mode()[0])

data.isnull().sum()

data.info()

data['ApplicantIncome']=data['ApplicantIncome'].astype('float64')

data['Gender']=data['Gender'].astype('object')

data['Married']=data['Married'].astype('object')

data['CoapllicantIncome']=data['CoapplicantIncome'].astype('object')

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

plt.subplot(121)

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

plt.subplot(122)

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

plt.show()

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

plt.subplot(1,4,1)

sns.countplot(x='Gender',data=data)

plt.subplot(1,4,2)

sns.countplot(x='Education',data=data)

plt.show()

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

plt.subplot(131)

sns.countplot(x='Married',hue='Gender',data=data)

plt.subplot(132)

sns.countplot(x='Self\_Employed',hue='Education',data=data)

plt.subplot(133)

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

pd.crosstab(data['Gender'],[data['Self\_Employed']])

sns.swarmplot(x='Gender',data=data, hue='Loan\_Status')

from imblearn.combine import SMOTETomek

smote=SMOTETomek()

y=data['Loan\_Status']

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

x.shape

y.shape

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

print(y.value\_counts())

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

names = x\_bal.columns

sc=StandardScaler()

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

x\_bal=pd.DataFrame(x\_bal,columns=names)

x\_bal.head()

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

X\_train.shape,y\_test.shape

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))

RandomForest(X\_train,X\_test,y\_train,y\_test)

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))

DecisionTree(X\_train,X\_test,y\_train,y\_test)

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))

KNN(X\_train,X\_test,y\_train,y\_test)

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))

XGB(X\_train,X\_test,y\_train,y\_test)

import tensorflow

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.linear\_model import LinearRegression

linear\_regressor=LinearRegression()

linear\_regressor.fit(X\_train,y\_train)

linear\_regressor.predict(X\_test)

classifier = Sequential()

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

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

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

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

y\_Pred = linear\_regressor.predict(X\_test)

y\_Pred

y\_Pred = y\_Pred.astype(int)

y\_Pred

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)

import pandas as pd

import numpy as np

np.mean(cv)

import pickle

from sklearn.model\_selection import RandomizedSearchCV

import imblearn

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

import pickle

pickle.dump(XGB,open("rdf.pkl",'wb'))

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

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