CONTENTS

1.INTRODUTION

1.1 Overview

Loan Prediction is very helpful for employee of banks as well as for the applicant also. The aim this paper is to provide quick, immediate and easy way to choose the deserving applicants. Dream housing Finance Company deals in all loans. They have presence across all urban and rural areas. Customer first apply for loan after that company or bank validates the customer eligibility for loan.

Company or bank wants to automate the loan eligibility process(real time) based on customer details provide while filing application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and other. This project has taken the data of previous customer of various banks to whom on a set of parameters loan were appoved.

So the machine learning model is trained on that record to get accurate results. Our main objective of this project is to predict the safety of loan. To predict loan safety, th SVM and Navie Bayes algorithm are used. First the data is cleaned so as to avoid the missing values in the data set.

1.2 Purpose

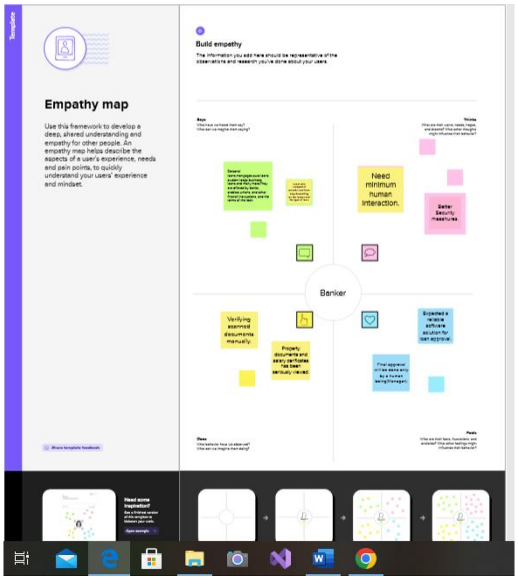
It is done by predicting if the loan can be given to that person on the basis of various parameters like credit score, income, age, marital status, gender, etc.

The prediction model not only helps the applicant but alos help the bank by minimizing the risk reducing the number od defaulters.

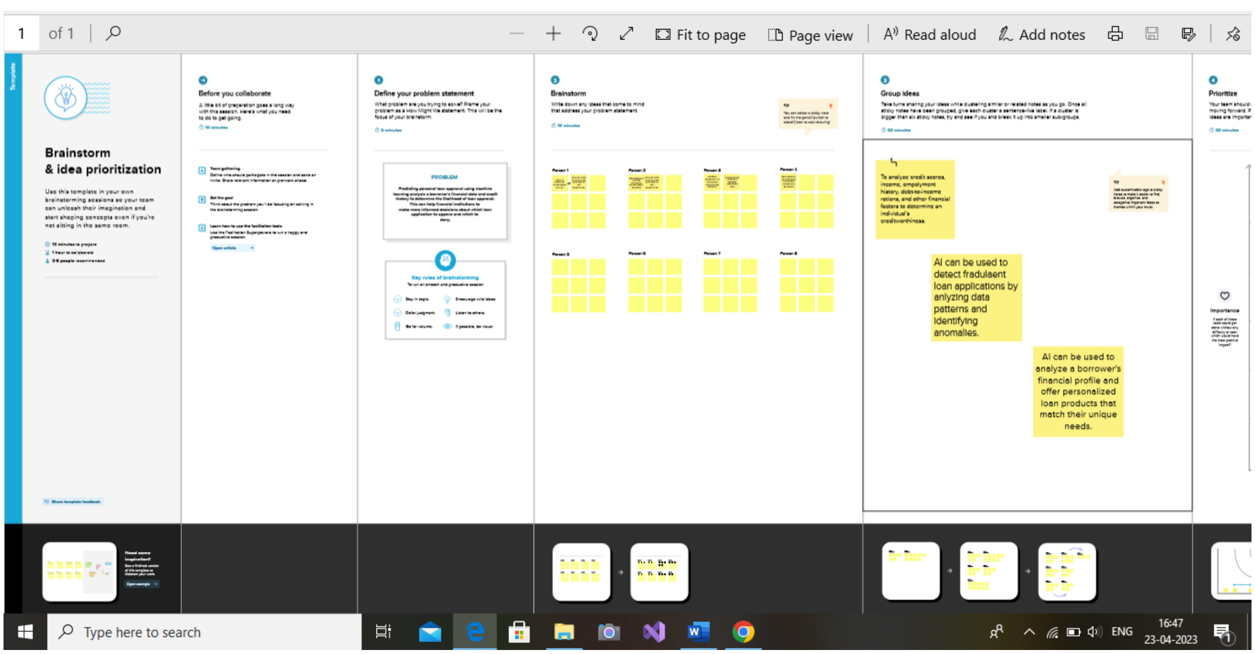
Loan Prediction System alloes jumping to specify application so that it can be check on priority basis. This Paperis execlusively for the managing authority of Bank/Finance company. Whole process of prediction is done private so stakeholders would be able to after the processing.

2.Problem Definition&Design Thinking

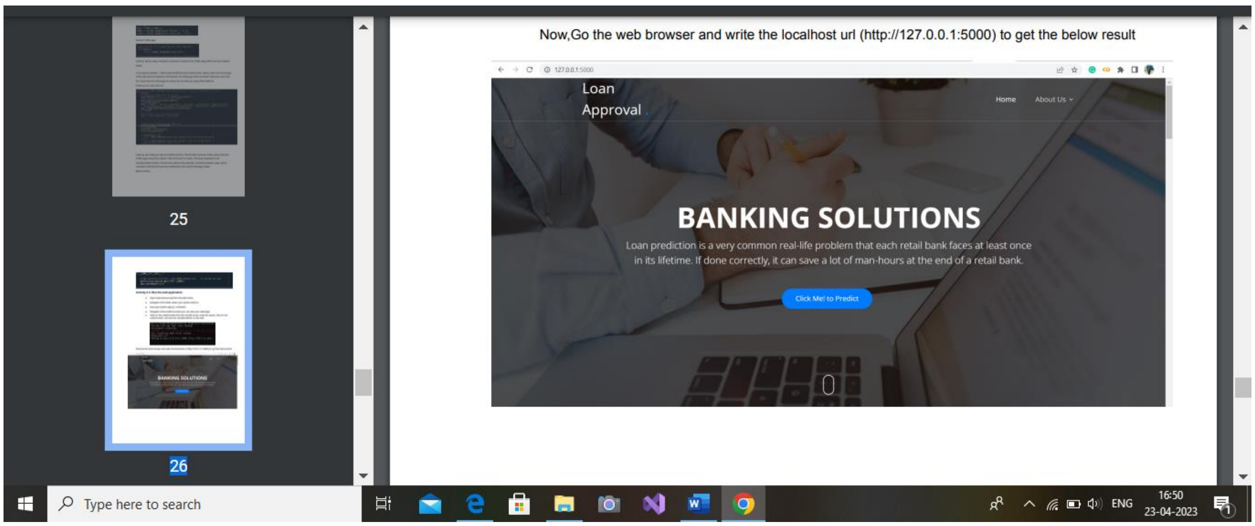
2.1 Empathy map

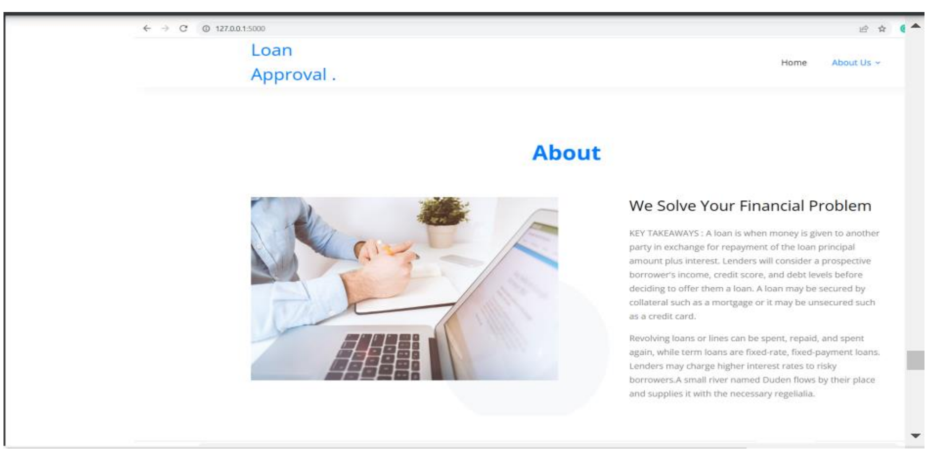


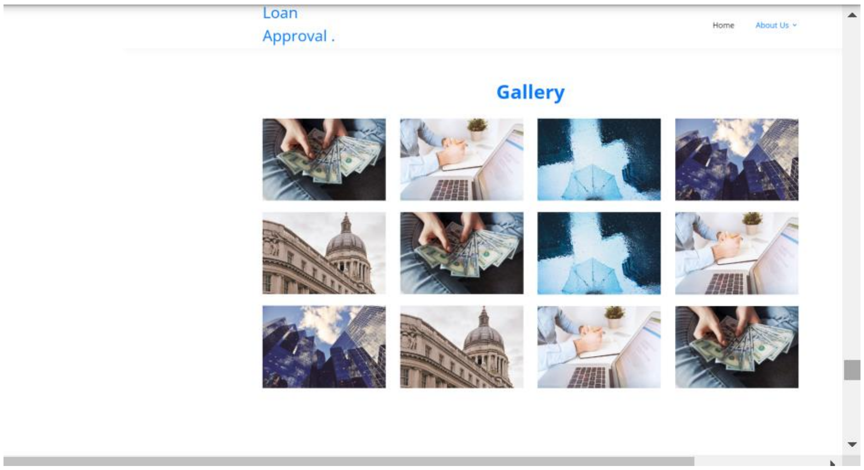
2.2 Ideation&Brainstorming Map

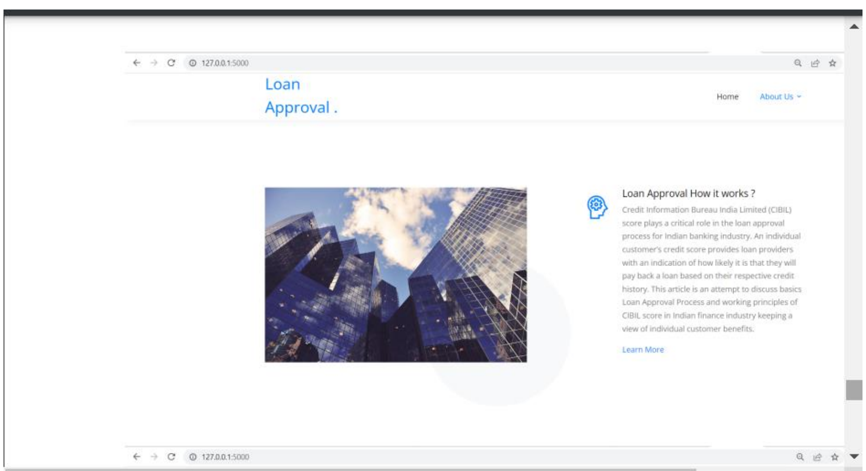


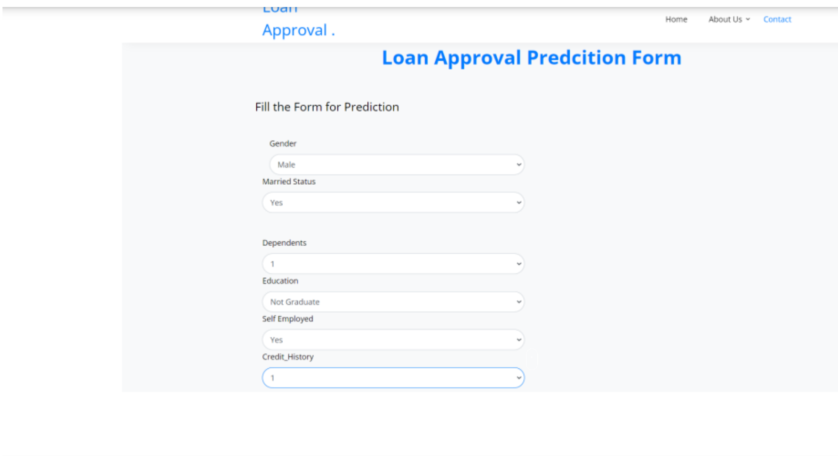
3. Result


Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result
Loan
Approval
Home
About Us
BANKING SOLUTIONS
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.
Click Me! to Predict
25
26
Type here to search
ENG
16:50
23-04-2023

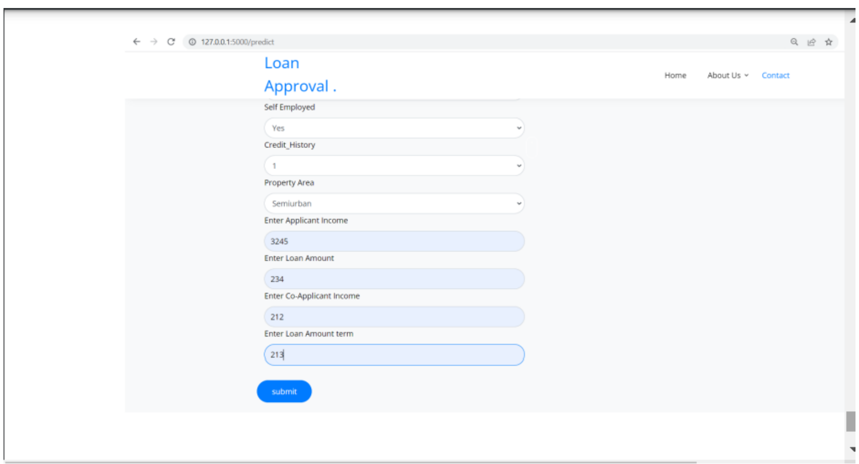

Loan
Approval .
Home
About Us
About
We Solve Your Financial Problem
KEY TAKEAWAYS : A loan is when money is given to another party in exchange for repayment of the loan principal amount plus interest. Lenders will consider a prospective borrower's income, credit score, and debt levels before deciding to offer them a loan. A loan may be secured by collateral such as a mortgage or it may be unsecured such as a credit card.
Revolving loans or lines can be spent, repaid, and spent again, while term loans are fixed-rate, fixed-payment loans. Lenders may charge higher interest rates to risky borrowers.A small river named Duden flows by their place and supplies it with the necessary regelialia.


Loan
Approval .
Home
About Us
Gallery

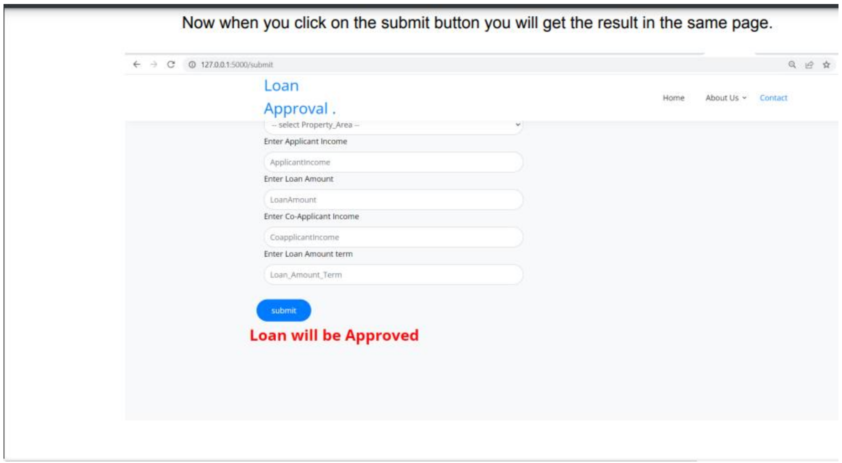

127.0.0.1:5000
Loan
Approval .
Home
About Us
Loan Approval How it works ?
Credit Information Bureau India Limited (CIBIL) score plays a critical role in the loan approval process for Indian banking industry. An individual customer's credit score provides loan providers with an indication of how likely it is that they will pay back a loan based on their respective credit history. This article is an attempt to discuss basics Loan Approval Process and working principles of CIBIL score in Indian finance industry keeping a view of individual customer benefits.
Learn More
127.0.0.1:5000



Approval .
Home
About Us
Contact
Loan Approval Predcition Form
Fill the Form for Prediction
Gender
Male
Married Status
Yes
Dependents
1
Education
Not Graduate
Self Employed
Yes
Credit_History
1



127.0.0.1:5000/predict
Loan
Approval .
Home
About Us
Contact
Self Employed
Yes
Credit_History
1
Property Area
Semiurban
Enter Applicant Income
3245
Enter Loan Amount
234
Enter Co-Applicant Income
212
Enter Loan Amount term
213
submit



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 model are automated and less likely to make mistakes.

This mean that loan processing becomes not only faster but more acurate too cutting costs on the whole.

Disadvantages:

The disadvantages of this model is that it esphasize different weight to each factors but in real life sometimes loan can be approved on the basis of single strong factor only, which is not possible through this system you could be paying interest on funds you’re not using. You could have trouble making monthly repayments if yours customers don’t pay you promptly, causing cashflow problems.

5.Applications

Banking and finance: In the banking and finance sector, loan approval prediction can help lender asses the creditworthiness of borrowers and make informed decisions about whether or not to approve a loan.

E-commerce: These companies can use loan approval prediction to offer financing options to their customers.

Insurance: These companies can use loan approval prediction to access the financial stability of potential policy holders.

Real Estate: In this industry, loan approval prediction can help lender assess the risk of default on montage loans.

6.Conclusion

So here, it can be concluded with confidence that the Naïve Bayes model is extremely efficient and gives a better result when compared to other models. It works correctly and fulfils all requirements of bankers. This system properly and accurately calculate the result. It predicts the loan is approve or reject to loan applicant or customer very accurately.

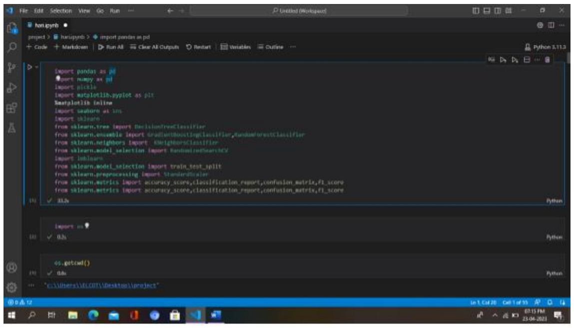
7.Future Scope

With the help of loan prediction, business could provide more targeted recommandations based on users prediction location.

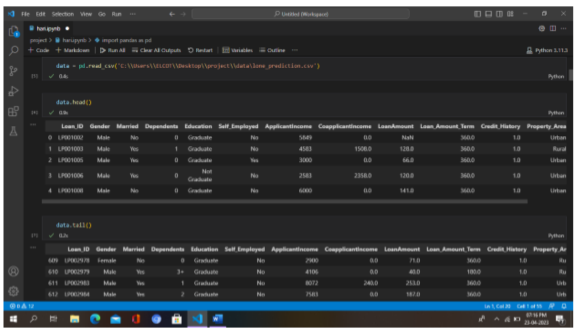
Loan prediction can be used to improve transportation services as prediction traffic congestion and optimizing routes for public transportation ride sharing services.

These models can be used to segment customer based on the creditworthiness and other factors.

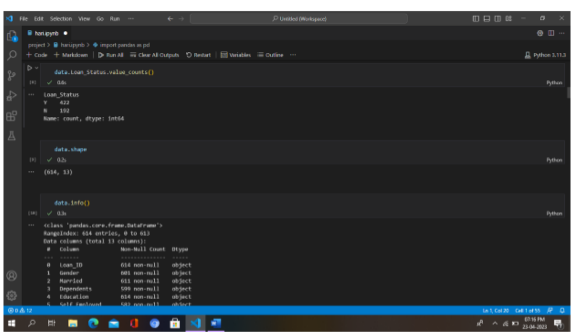
8.Appendix



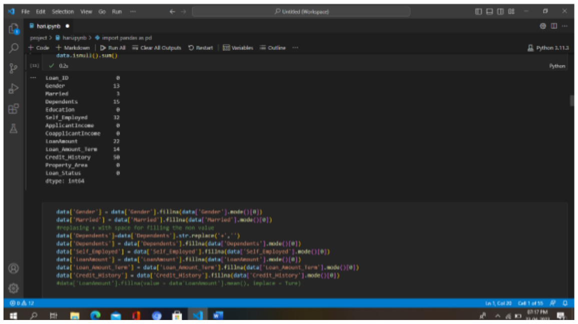
File Edit Selection View Go Run
Untitled (Workspace)
hari.ipynb
project > hari.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.3
import pandas as pd
import numpy as pd
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier,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
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,f1_score
import os
os.getcwd()
'c:\\Users\\ELCOT\\Desktop\\project'

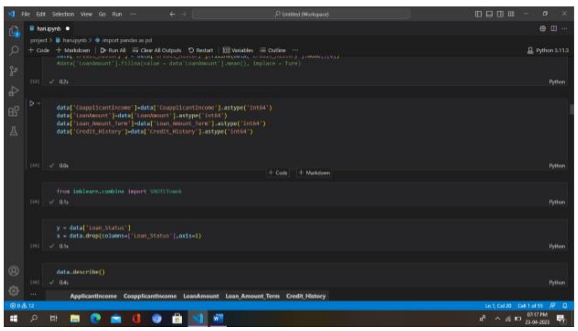


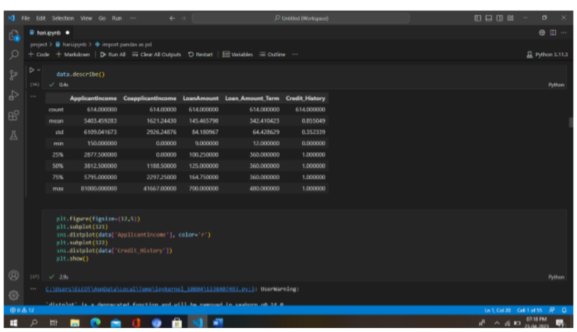
File Edit Selection View Go Run
Untitled (Workspace)
hari.ipynb
project > hari.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.3
data = pd.read_csv('C:\\Users\\ELCOT\\Desktop\\project\\data\lone_prediction.csv')
data.head()
Loan_ID Gender Married Dependents Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History Property_Area
0 LP001002 Male No 0 Graduate No 5849 0.0 NaN 360.0 1.0 Urban
1 LP001003 Male Yes 1 Graduate No 4583 1508.0 128.0 360.0 1.0 Rural
2 LP001005 Male Yes 0 Graduate Yes 3000 0.0 66.0 360.0 1.0 Urban
3 LP001006 Male Yes 0 Not Graduate No 2583 2358.0 120.0 360.0 1.0 Urban
4 LP001008 Male No 0 Graduate No 6000 0.0 141.0 360.0 1.0 Urban
data.tail()
Loan_ID Gender Married Dependents Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History Property_Ar
609 LP002978 Female No 0 Graduate No 2900 0.0 71.0 360.0 1.0 Ru
610 LP002979 Male Yes 3+ Graduate No 4106 0.0 40.0 180.0 1.0 Ru
611 LP002983 Male Yes 1 Graduate No 8072 240.0 253.0 360.0 1.0 Urb
612 LP002984 Male Yes 2 Graduate No 7583 0.0 187.0 360.0 1.0 Urb
Ln 1, Col 20 Cell 1 of 55
07:16 PM
23-04-2023

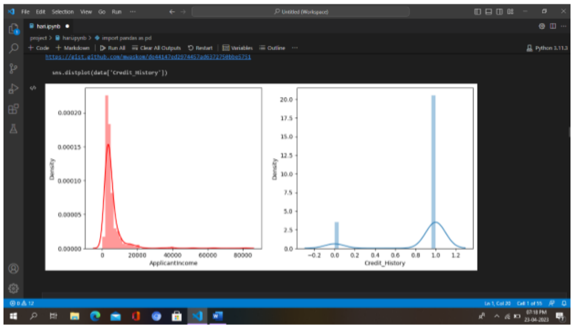


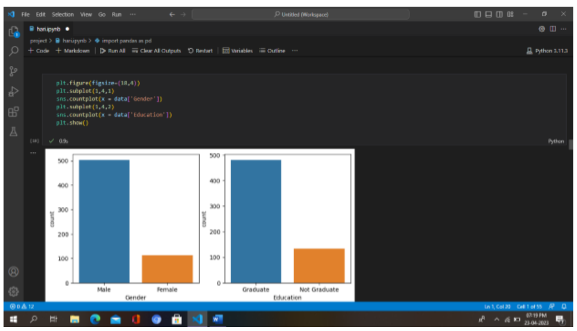
File Edit Selection View Go Run
hari.ipynb
project > hari.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Restart Variables Outline
data.Loan_Status.value_counts()
Loan_Status
Y 422
N 192
Name: count, dtype: int64
data.shape
(614, 13)
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
Column Non-Null Count Dtype
0 Loan_ID 614 non-null object
1 Gender 601 non-null object
2 Married 611 non-null object
3 Dependents 599 non-null object
4 Education 614 non-null object

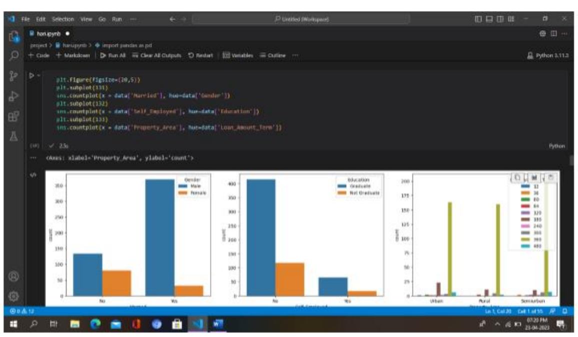

File Edit Selection View Go Run
Untitled (Workspace)
hari.ipynb
project > hari.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.3
data.isnull().sum()
0.2s
Python
Loan_ID 0
Gender 13
Married 3
Dependents 15
Education 0
Self_Employed 32
ApplicantIncome 0
CoapplicantIncome 0
LoanAmount 22
Loan_Amount_Term 14
Credit_History 50
Property_Area 0
Loan_Status 0
dtype: int64
data['Gender'] = data['Gender'].fillna(data['Gender'].mode()[0])
data['Married'] = data['Married'].fillna(data['Married'].mode()[0])
#replacing + with space for filling the non value
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])
#data['LoanAmount'].fillna(value = data'LoanAmount'].mean(), implace = Ture)
Ln 1, Col 20 Cell 1 of 55

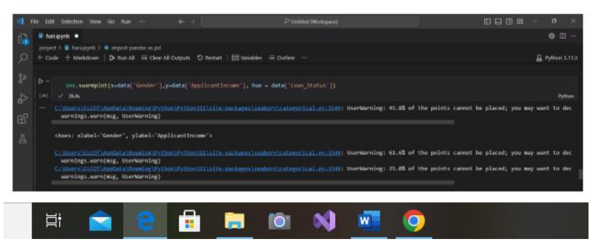

File Edit Selection View Go Run
Untitled (Workspace)
hari.ipynb
project > hari.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.3
#data['LoanAmount'].fillna(value = data'LoanAmount'].mean(), implace = Ture)
0.2s
Python
data['CoapplicantIncome']=data['CoapplicantIncome'].astype('int64')
data['LoanAmount']=data['LoanAmount'].astype('int64')
data['Loan_Amount_Term']=data['Loan_Amount_Term'].astype('int64')
data['Credit_History']=data['Credit_History'].astype('int64')
0.0s
Python
Code Markdown
from imblearn.combine import SMOTETomek
0.1s
Python
y = data['Loan_Status']
x = data.drop(columns=['Loan_Status'],axis=1)
0.1s
Python
data.describe()
0.4s
Python
ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History
Ln 1, Col 20 Cell 1 of 55
07:17 PM
23-04-2023


File Edit Selection View Go Run
Untitled (Workspace)
hari.ipynb
project > hari.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.3
data.describe()
0.4s
Python
ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History
count 614.000000 614.00000 614.000000 614.000000 614.000000
mean 5403.459283 1621.24430 145.465798 342.410423 0.855049
std 6109.041673 2926.24876 84.180967 64.428629 0.352339
min 150.000000 0.00000 9.000000 12.000000 0.000000
25% 2877.500000 0.00000 100.250000 360.000000 1.000000
50% 3812.500000 1188.50000 125.000000 360.000000 1.000000
75% 5795.000000 2297.25000 164.750000 360.000000 1.000000
max 81000.000000 41667.00000 700.000000 480.000000 1.000000
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(data['ApplicantIncome'], color='r')
plt.subplot(122)
sns.distplot(data['Credit_History'])
plt.show()
2.9s
Python
UserWarning:
Ln 1, Col 20 Cell 1 of 55

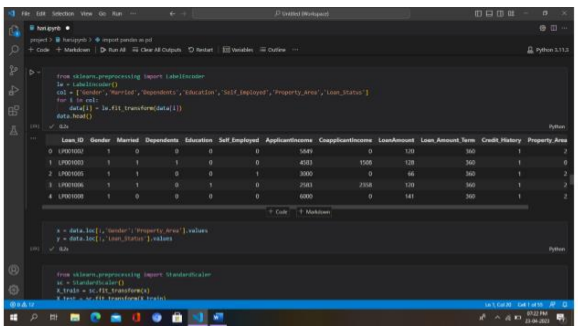

sns.distplot(data['Credit_History'])
Density
ApplicantIncome
Credit_History


plt.figure(figsize=(10,4))
plt.subplot(1,4,1)
sns.countplot(x = data['Gender'])
plt.subplot(1,4,2)
sns.countplot(x = data['Education'])
plt.show()
count
Male
Female
Gender
Graduate
Not Graduate
Education

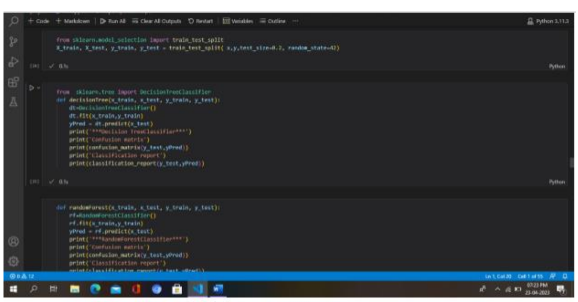

Gender
Male
Female
Education
Graduate
Not Graduate
Urban
Rural
Semiurban


sns.swarmplot(x=data['Gender'],y=data['ApplicantIncome'], hue = data['Loan_Status'])
<Axes: xlabel='Gender', ylabel='ApplicantIncome'>

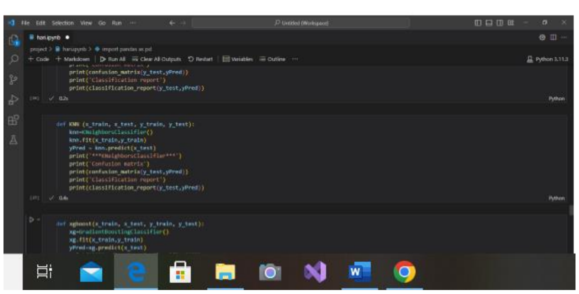
warnings.warn(msg, UserWarning)
UserWarning: 25.0% of the points cannot be placed; you may want to dec
Loan_Status
Y
N
ApplicantIncome
80000
70000
60000
50000
40000
30000
20000
10000
0
Male
Female
Gender
from sklearn.preprocessing import LabelEncoder



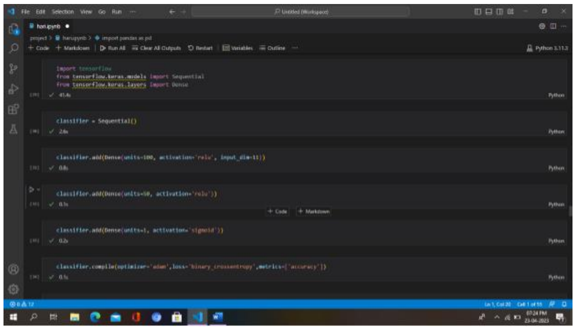
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
col = ['Gender','Married','Dependents','Education','Self_Employed','Property_Area','Loan_Status']
for i in col:
data[i] = le.fit_transform(data[i])
data.head()
Loan_ID Gender Married Dependents Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History Property_Area
0 LP001002 1 0 0 0 0 5849 0 120 360 1 2
1 LP001003 1 1 1 0 0 4583 1508 128 360 1 0
2 LP001005 1 1 0 0 1 3000 0 66 360 1 2
3 LP001006 1 1 0 1 0 2583 2358 120 360 1 2
4 LP001008 1 0 0 0 0 6000 0 141 360 1 2
x = data.loc[:,'Gender':'Property_Area'].values
y = data.loc[:,'Loan_Status'].values
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(x)

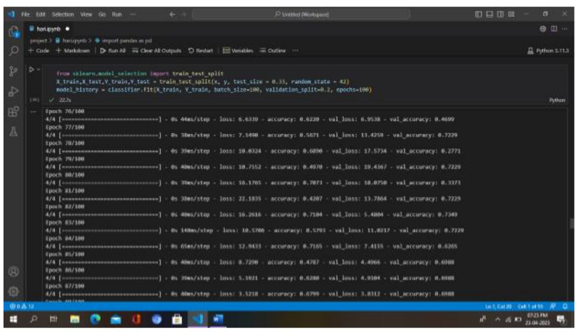


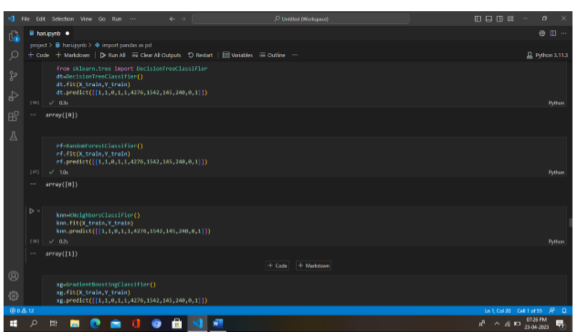
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state=42)
from sklearn.tree import DecisionTreeClassifier
def decisionTree(x_train, x_test, y_train, y_test):
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
yPred = dt.predict(x_test)
print('***Decision TreeClassifier***')
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)
yPred = rf.predict(x_test)
print('***RandomForestClassifier***')
print('Confusion matrix')
print(confusion_matrix(y_test,yPred))
print('Classification report')

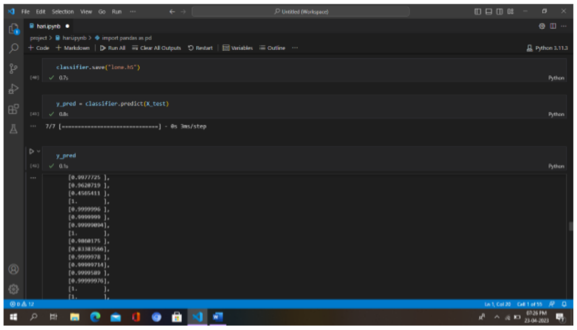


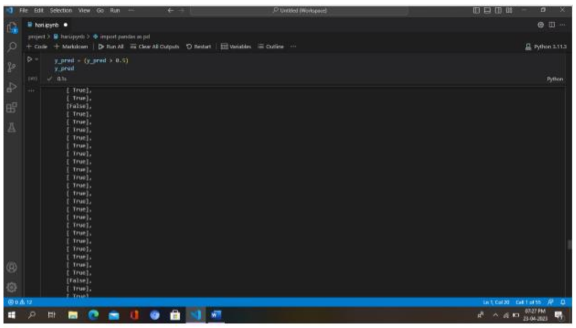
print(confusion_matrix(y_test,yPred))
print('Classification report')
print(classification_report(y_test,yPred))
def KNN (x_train, x_test, y_train, y_test):
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
yPred = knn.predict(x_test)
print('***KNeighborsClassifier***')
print('Confusion matrix')
print(confusion_matrix(y_test,yPred))
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)

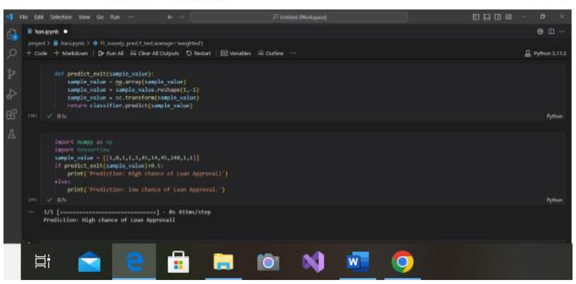

import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
classifier = Sequential()
classifier.add(Dense(units=100, activation='relu', input_dim=11))
classifier.add(Dense(units=50, activation='relu'))
classifier.add(Dense(units=1, activation='sigmoid'))
classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

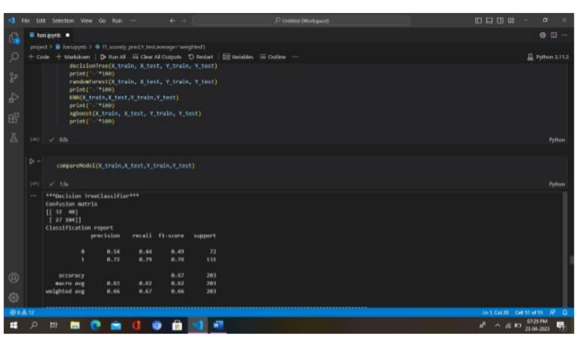

from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(x, y, test_size = 0.33, random_state = 42)
model_history = classifier.fit(X_train, Y_train, batch_size=100, validation_split=0.2, epochs=100)
Epoch 76/100
Epoch 77/100
Epoch 78/100
Epoch 79/100
Epoch 80/100
Epoch 81/100

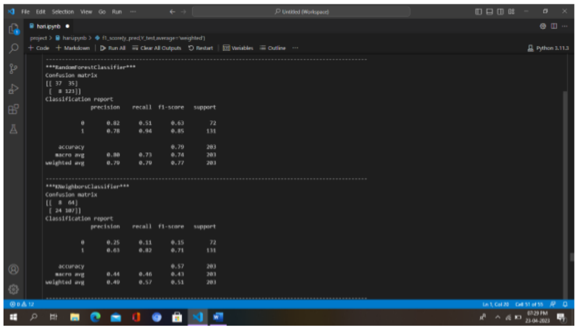

from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(X_train,Y_train)
dt.predict([[1,1,0,1,1,4276,1542,145,240,0,1]])
array([0])
rf=RandomForestClassifier()
rf.fit(X_train,Y_train)
rf.predict([[1,1,0,1,1,4276,1542,145,240,0,1]])
array([0])
knn=KNeighborsClassifier()
knn.fit(X_train,Y_train)
knn.predict([[1,1,0,1,1,4276,1542,145,240,0,1]])
array([1])
xg=GradientBoostingClassifier()
xg.fit(X_train,Y_train)
xg.predict([[1,1,0,1,1,4276,1542,145,240,0,1]])

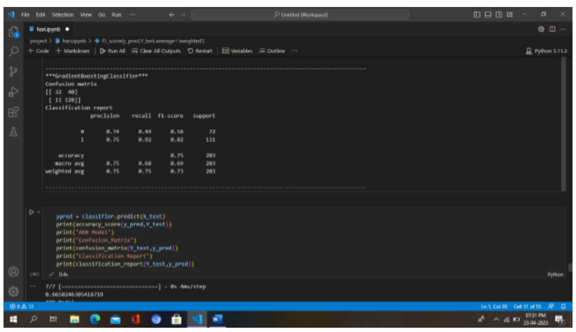

classifier.save("lone.h5")
y_pred = classifier.predict(X_test)
7/7 [==============================] - 0s 3ms/step
y_pred

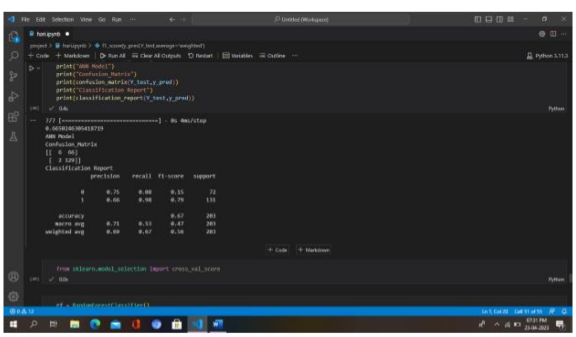

y_pred = (y_pred > 0.5)
y_pred

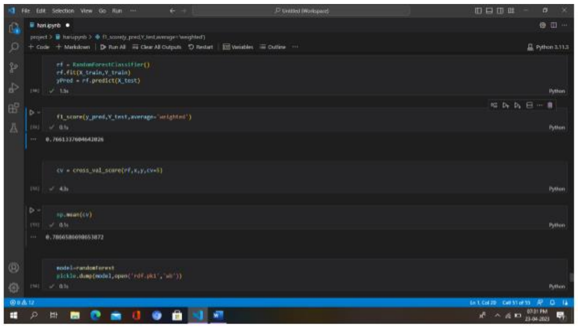

def predict_exit(sample_value):
sample_value = np.array(sample_value)
sample_value = sample_value.reshape(1,-1)
sample_value = sc.transform(sample_value)
return classifier.predict(sample_value)
import numpy as np
import tensorflow
sample_value = [[1,0,1,1,1,45,14,45,240,1,1]]
if predict_exit(sample_value)>0.5:
print('Prediction: High chance of Loan Approval!')
else:
print('Prediction: low chance of Loan Approval.')
Prediction: High chance of Loan Approval!


decisionTree(X_train, X_test, Y_train, Y_test)
randomForest(X_train, X_test, Y_train, Y_test)
KNN(X_train,X_test,Y_train,Y_test)
xgboost(X_train, X_test, Y_train, Y_test)
compareModel(X_train,X_test,Y_train,Y_test)
Decision TreeClassifier
Confusion matrix
Classification report
precision recall f1-score support
accuracy
macro avg
weighted avg


File Edit Selection View Go Run
Untitled (Workspace)
hari.ipynb
project > hari.ipynb > f1_score(y_pred,Y_test,average='weighted')
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.2
RandomForestClassifier
Confusion matrix
[[37 35]
[8 123]]
Classification report
precision recall f1-score support
0 0.82 0.51 0.63 72
1 0.78 0.94 0.85 131
accuracy 0.79 203
macro avg 0.80 0.73 0.74 203
weighted avg 0.79 0.79 0.77 203
KNeighborsClassifier
Confusion matrix
[[8 64]
[24 107]]
Classification report
precision recall f1-score support
0 0.25 0.11 0.15 72
1 0.63 0.82 0.71 131
accuracy 0.57 203
macro avg 0.44 0.46 0.43 203
weighted avg 0.49 0.57 0.51 203


File Edit Selection View Go Run
Untitled (Workspace)
hari.ipynb
project > hari.ipynb > f1_score(y_pred,Y_test,average='weighted')
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.2
GradientBoostingClassifier
Confusion matrix
[[32 40]
[11 120]]
Classification report
precision recall f1-score support
0 0.74 0.44 0.56 72
1 0.75 0.92 0.82 131
accuracy 0.75 203
macro avg 0.75 0.68 0.69 203
weighted avg 0.75 0.75 0.73 203
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))
0.4s
Python
7/7 [==============================] - 0s 4ms/step
0.6650246305418719


File Edit Selection View Go Run
Untitled (Workspace)
hari.ipynb
project > hari.ipynb > f1_score(y_pred,Y_test,average='weighted')
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.2
print("ANN Model")
print("Confusion_Matrix")
print(confusion_matrix(Y_test,y_pred))
print("Classification Report")
print(classification_report(Y_test,y_pred))
0.4s
Python
7/7 [==============================] - 0s 4ms/step
0.6650246305418719
ANN Model
Confusion_Matrix
[[6 66]
[2 129]]
Classification Report
precision recall f1-score support
0 0.75 0.08 0.15 72
1 0.66 0.98 0.79 131
accuracy 0.67 203
macro avg 0.71 0.53 0.47 203
weighted avg 0.69 0.67 0.56 203
+ Code + Markdown
from sklearn.model_selection import cross_val_score
0.0s
Python


File Edit Selection View Go Run
Untitled (Workspace)
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
rf = RandomForestClassifier()
rf.fit(X_train,Y_train)
yPred = rf.predict(X_test)
f1_score(y_pred,Y_test,average='weighted')
cv = cross_val_score(rf,x,y,cv=5)
np.mean(cv)
model=randomforest
pickle.dump(model,open('rdf.pkl','wb'))
Python