PERSONAL LOAN APPROVAL

INTRODUCTION:

LOANS are the major requirement of the modern world. By this only, Banks get a major part of the total profit. It is beneficial for students to manage their education and living expenses, and for people to buy any kind of luxury like houses, cars, etc.

But when it comes to deciding whether the applicant's profile is relevant to be granted with loan or not. Banks have to look after many aspects.

So, here we will be using Machine Learning with Python to ease their work and predict whether the candidate's profile is relevant or not using key features like Marital Status, Education, Applicant Income, Credit History, etc.
Dream Housing Finance company deals in all kinds of home loans. They have a presence across all urban, semi-urban and rural areas. The customer first applies for a home loan and after that, the company validates the customer eligibility for the loan.
The company wants to automate the loan eligibility process (real-time) based on customer detail provided while filling out online application forms. These details are Gender, Marital Status, Education, and number of Dependents, Income, Loan Amount, Credit History, and others.

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

PURPOSE:

Using historical data, machine learning algorithms can learn to identify trends and patterns to evaluate possible outcomes. Using this as its reference framework, the algorithm can repeat the process with current data to make predictions about the future

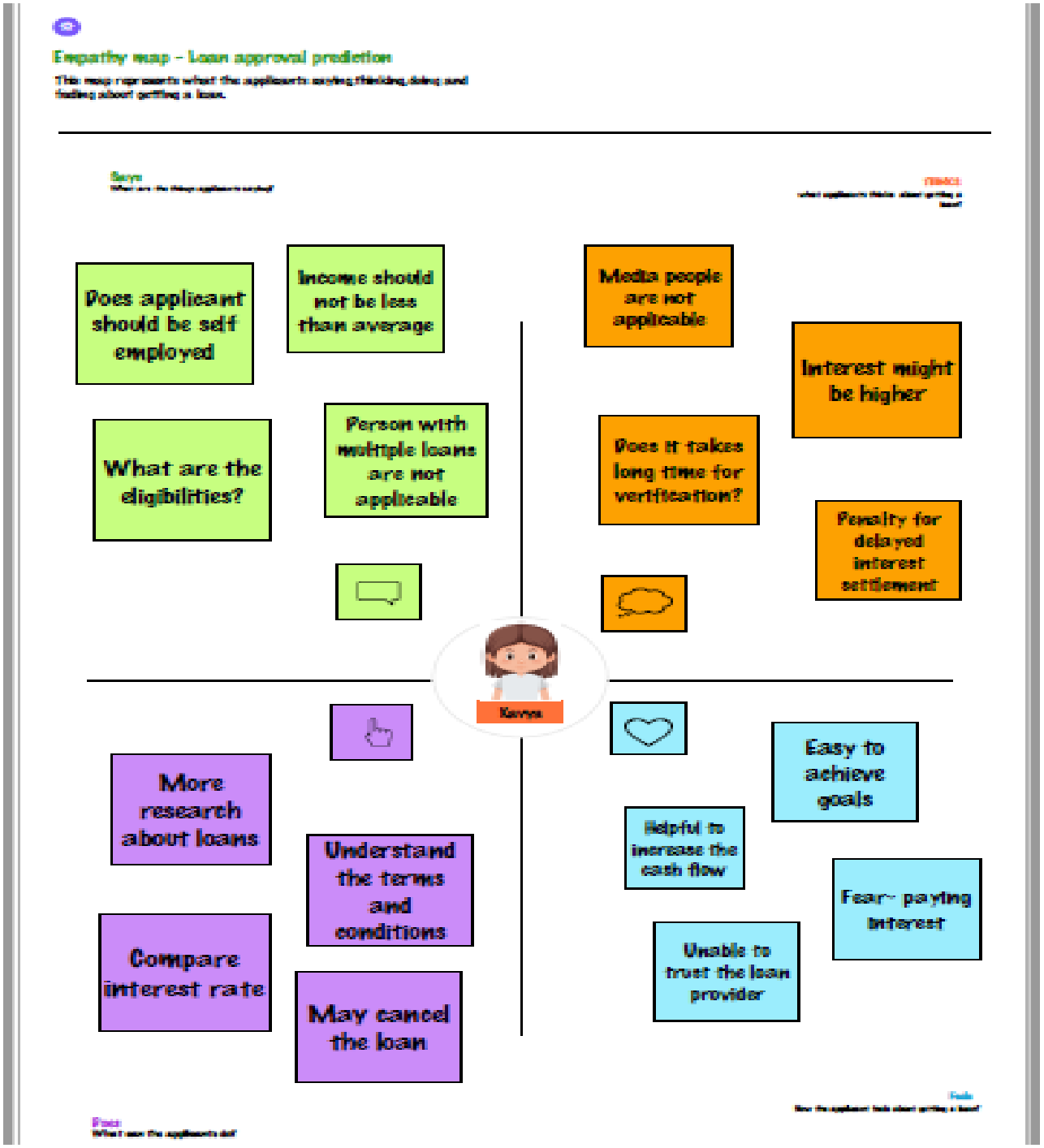
Personal loans are borrowed money that can be used for **large purchases, debt consolidation, emergency expenses** and much more. These loans are paid back in monthly installments over the course of a few months or upwards of a few years.

WHAT CAN BE ACHIEVED?

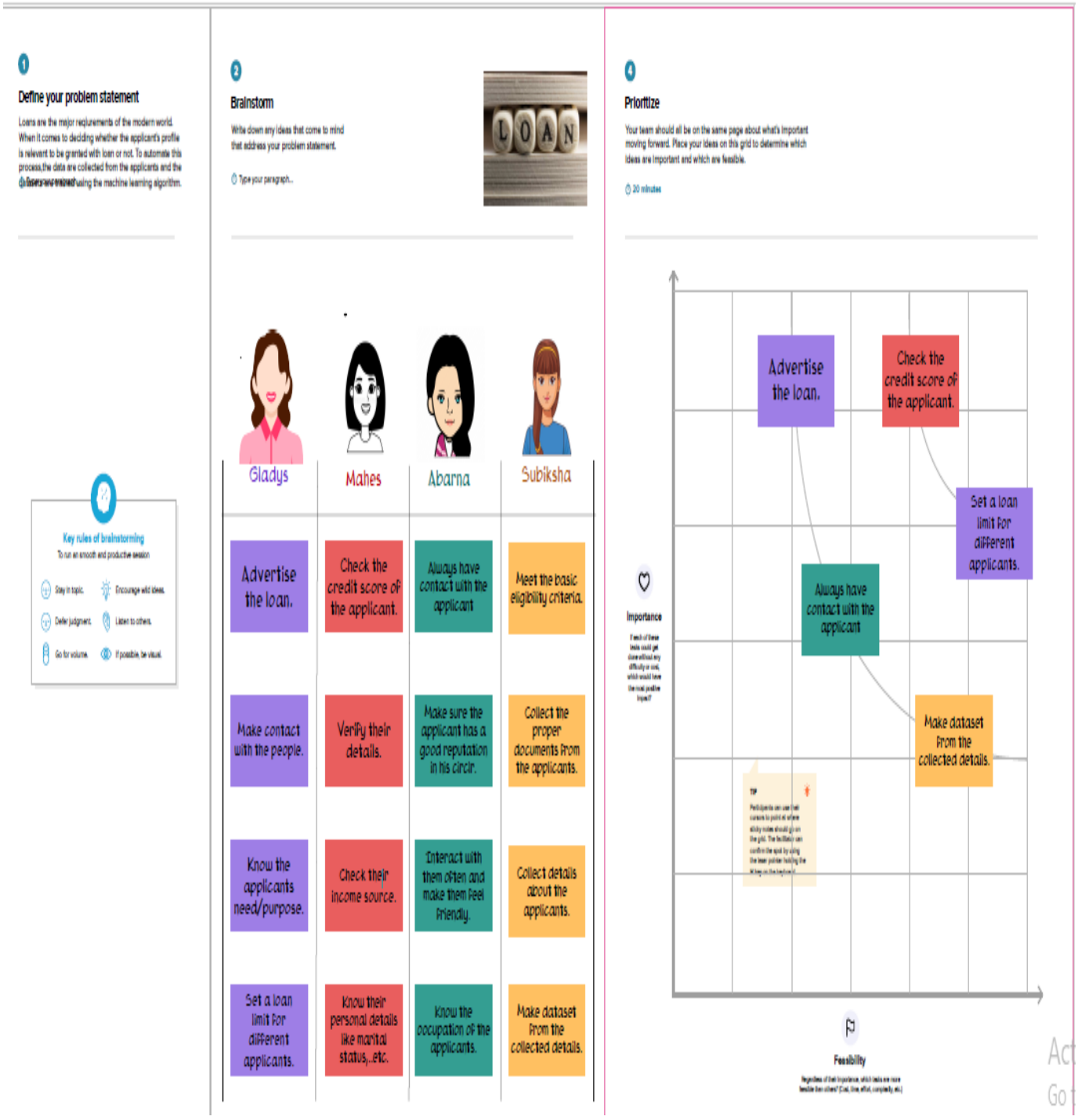
Accuracy—one of the primary benefits of using machine learning for credit scoring is **its accuracy**. Unlike human manual processing, ML-based models are automated and less likely to make mistakes. This means that loan processing becomes not only faster but more accurate, too, cutting costs on the whole.

Machine learning can be used to automate the income verification process. By using data from previous loan applications, tax returns, bank statements, machine learning models can learn to identify patterns that are predictive of loan default.

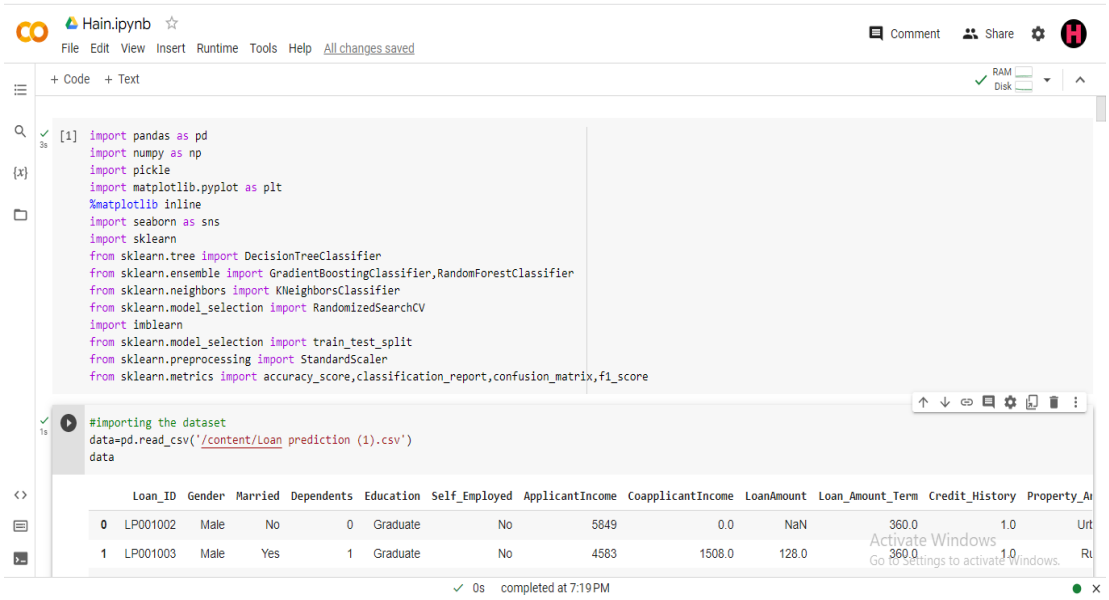
EMPATHY MAP:


Empathy map - Loan approval prediction
This map represents what the applicant is saying, thinking, doing and feeling about getting a loan.
Says
What are the things applicant is saying?
Does applicant should be self employed
Income should not be less than average
What are the eligibilities?
Person with multiple loans are not applicable
Media people are not applicable
Interest might be higher
Does it takes long time for verification?
Penalty for delayed interest settlement
Karyn
More research about loans
Understand the terms and conditions
Compare interest rate
May cancel the loan
Easy to achieve goals
Helpful to increase the cash flow
Fear- paying interest
Unable to trust the loan provider
Does
What does the applicant do?
Feels
How the applicant feels about getting a loan?

BRAIN STROM:


1
Define your problem statement
Loans are the major requirements of the modern world. When it comes to deciding whether the applicant's profile is relevant to be granted with loan or not. To automate this process,the data are collected from the applicants and the observations trained using the machine learning algorithm.
2
Brainstorm
Write down any ideas that come to mind that address your problem statement.
Type your paragraph...
LOAN
4
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes
Key rules of brainstorming
To run an smooth and productive session
Stay in topic.
Encourage wild ideas.
Defer judgment.
Listen to others.
Go for volume.
If possible, be visual.
Gladys
Mahes
Abarna
Subiksha
Advertise the loan.
Check the credit score of the applicant.
Always have contact with the applicant
Meet the basic eligibility criteria.
Make contact with the people.
Verify their details.
Make sure the applicant has a good reputation in his circle.
Collect the proper documents from the applicants.
Know the applicants need/purpose.
Check their income source.
Interact with them often and make them feel friendly.
Collect details about the applicants.
Set a loan limit for different applicants.
Know their personal details like marital status,..etc.
Know the occupation of the applicants.
Make dataset from the collected details.
Advertise the loan.
Check the credit score of the applicant.
Set a loan limit for different applicants.
Always have contact with the applicant
Make dataset from the collected details.
Importance
Feasibility

RESULT:



Hain.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

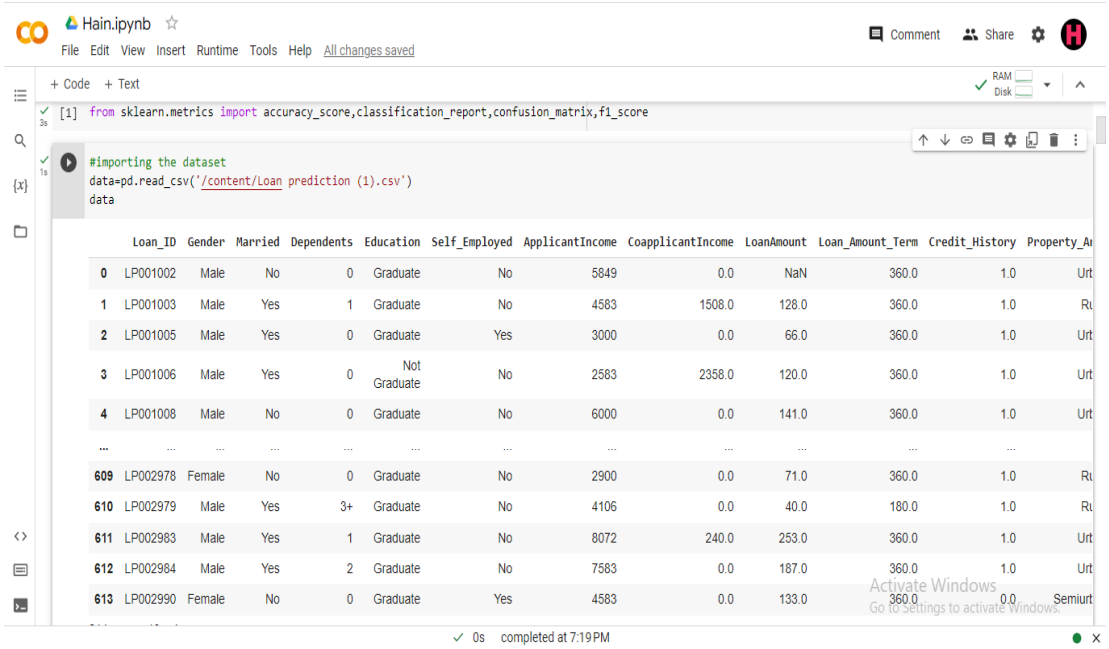
```
[1] import pandas as pd
    import numpy as np
    import pickle
    import matplotlib.pyplot as plt
    %matplotlib inline
    import seaborn as sns
    import sklearn
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import GradientBoostingClassifier,RandomForestClassifier
    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
```

```
#importing the dataset
data=pd.read_csv('/content/Loan prediction (1).csv')
data
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Ar
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urt
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Ru

Activate Windows
Go to settings to activate Windows.

0s completed at 7:19 PM



Hain.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

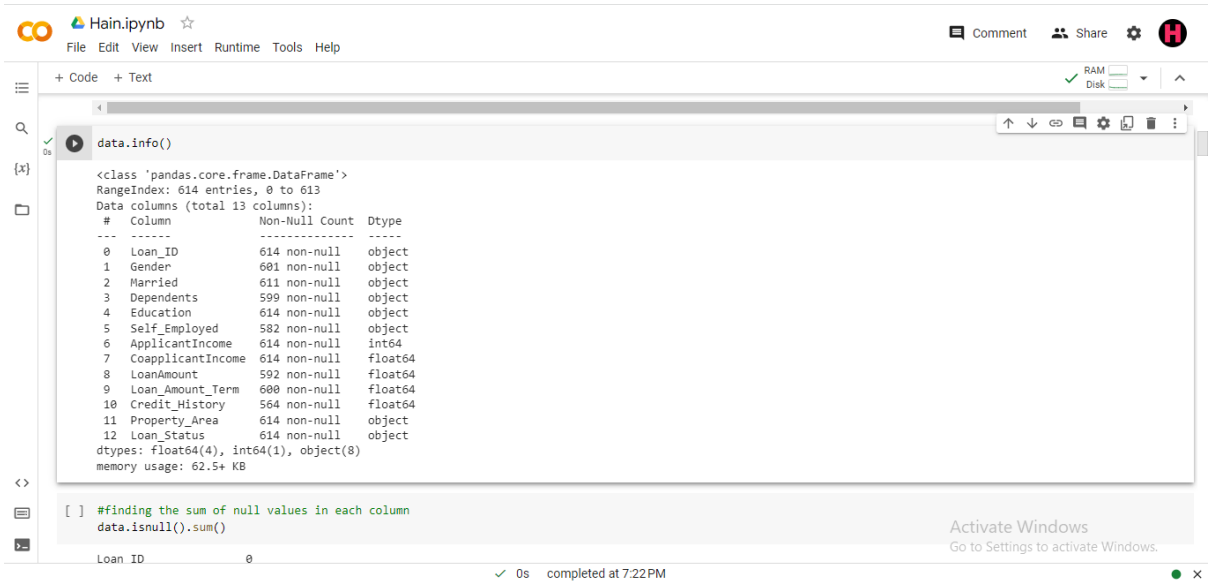
```
[1] from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,f1_score
```

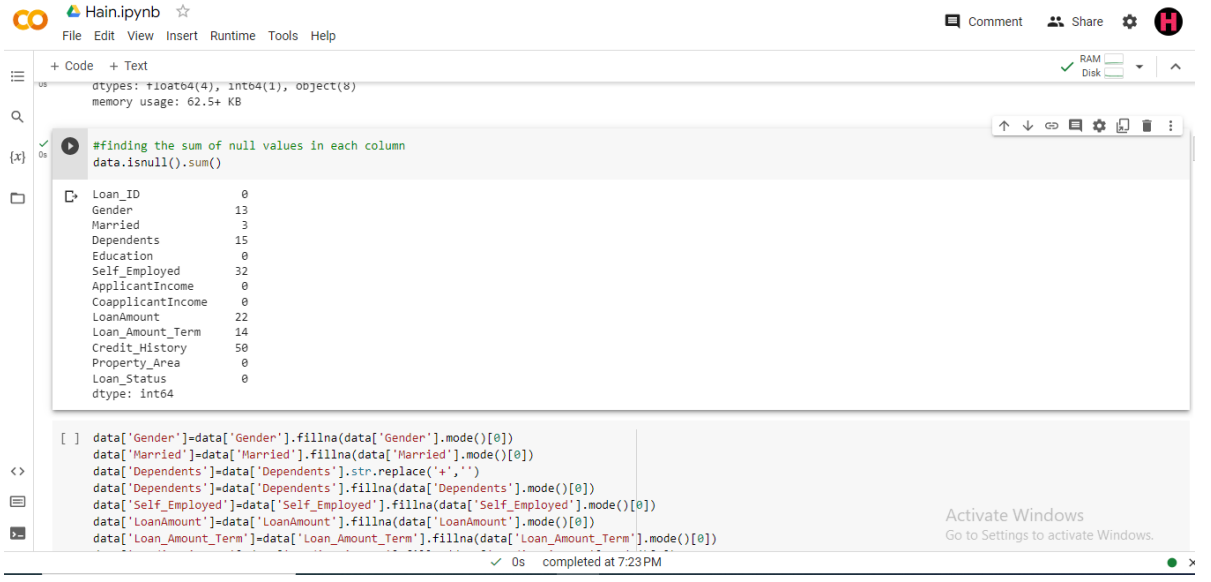
```
#importing the dataset
data=pd.read_csv('/content/Loan prediction (1).csv')
data
```

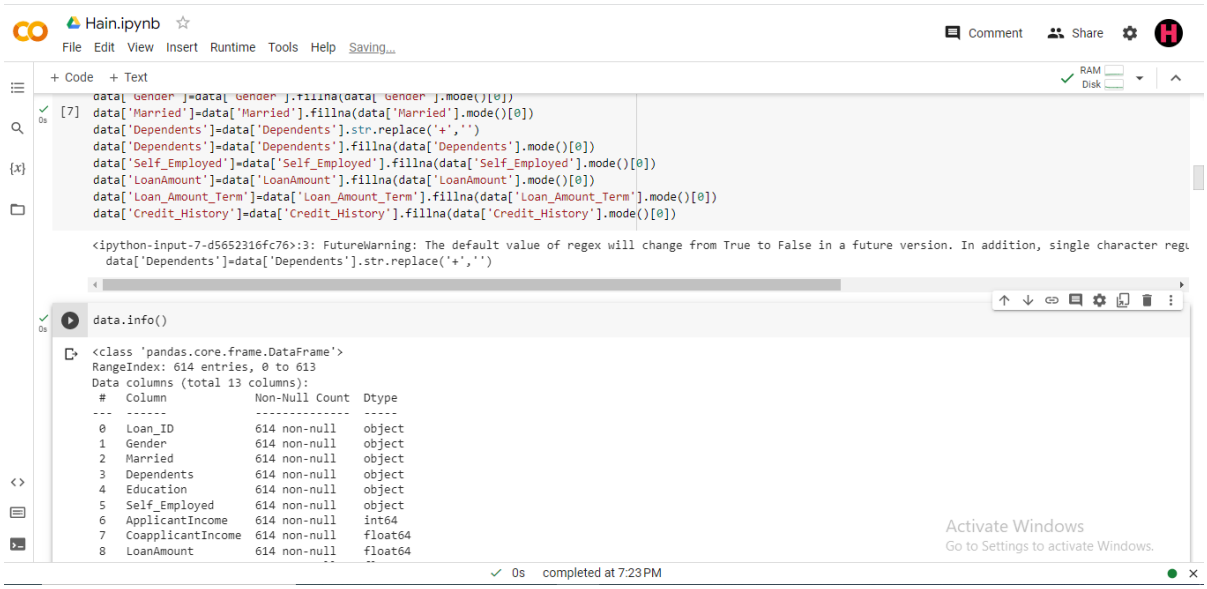
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Ar
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urt
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Ru
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urt
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urt
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urt
...	...	...	...	...	...	...	...	...	...	...	...	
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	Urt
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urt
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurt

Activate Windows
Go to settings to activate Windows.

0s completed at 7:19 PM


Hain.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
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
5 Self_Employed 582 non-null object
6 ApplicantIncome 614 non-null int64
7 CoapplicantIncome 614 non-null float64
8 LoanAmount 592 non-null float64
9 Loan_Amount_Term 600 non-null float64
10 Credit_History 564 non-null float64
11 Property_Area 614 non-null object
12 Loan_Status 614 non-null object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
#finding the sum of null values in each column
data.isnull().sum()
Loan_ID 0
Activate Windows
Go to Settings to activate Windows.
0s completed at 7:22 PM


Hain.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
memory usage: 62.5+ KB
#finding the sum of null values in each column
data.isnull().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
data['Gender']=data['Gender'].fillna(data['Gender'].mode()[0])
data['Married']=data['Married'].fillna(data['Married'].mode()[0])
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])
Activate Windows
Go to Settings to activate Windows.
0s completed at 7:23 PM


Hain.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment
Share
+ Code + Text
RAM
Disk
[7] data['Gender']=data['Gender'].fillna(data['Gender'].mode()[0])
data['Married']=data['Married'].fillna(data['Married'].mode()[0])
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])
<ipython-input-7-d5652316fc76>:3: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regu
data['Dependents']=data['Dependents'].str.replace('+','')
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 614 non-null object
2 Married 614 non-null object
3 Dependents 614 non-null object
4 Education 614 non-null object
5 Self_Employed 614 non-null object
6 ApplicantIncome 614 non-null int64
7 CoapplicantIncome 614 non-null float64
8 LoanAmount 614 non-null float64
Activate Windows
Go to Settings to activate Windows.
0s completed at 7:23 PM


Hain.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
data=data.replace(['Female'],[1])
data=data.replace(['No'],[0])
data=data.replace(['Yes'],[1])
data=data.replace(['Graduate'],[1])
data=data.replace(['Not Graduate'],[0])
data=data.replace(['Urban'],[0])
data=data.replace(['Semiurban'],[1])
data=data.replace(['Rural'],[2])
data=data.replace(['Y'],[1])
data=data.replace(['N'],[0])
print(data['Gender'].value_counts(),'\n',
data['Married'].value_counts(),'\n',
data['Self_Employed'].value_counts(),'\n',
data['Education'].value_counts(),'\n',
data['Property_Area'].value_counts(),'\n',
data['Loan_Status'].value_counts())
0 502
1 112
Name: Gender, dtype: int64
1 401
0 213
Name: Married, dtype: int64
0 532
1 82
Name: Self_Employed, dtype: int64
1 480
0 134
Name: Education, dtype: int64
Activate Windows
Go to Settings to activate Windows.
1s completed at 7:24 PM

Hain.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk

```
[9]  1    422
     0    192
     Name: Loan_Status, dtype: int64
```

```
[10] data.drop('Loan_ID',axis=1,inplace=True)
```

```
[11] 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['Property_Area']=data['Property_Area'].astype('int64')
```

```
[12] data['ApplicantIncome']=data['ApplicantIncome'].astype('int64')
     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')
```

```
[13] from imblearn.combine import SMOTETomek
     smote=SMOTETomek(sampling_strategy=0.90)
```

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


0s completed at 7:25 PM

Hain.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
[14] y=data['Loan_Status']
```

```
[15] x_bal,y_bal=smote.fit_resample(x,y)
```

```
[16] print(y.value_counts())
     print(y_bal.value_counts())
```

```
1    422
0    192
Name: Loan_Status, dtype: int64
1    349
0    306
Name: Loan_Status, dtype: int64
```

Descriptive statistical

```
data.describe()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Are
count	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.00000	614.000000	614.000000	614.000000	614.00000
mean	0.182410	0.653094	0.744300	0.781759	0.133550	5403.459283	1621.24430	145.465798	342.410423	0.855049	0.96254
std	0.386497	0.476373	1.009623	0.413389	0.340446	6109.041673	2926.24876	84.180967	64.428629	0.352339	0.78748
min	0.000000	0.000000	0.000000	0.000000	0.000000	150.000000	0.00000	9.000000	12.000000	0.000000	0.00000

0s completed at 7:25 PM

Hain.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

[17]

min	0.000000	0.000000	0.000000	0.000000	0.000000	150.000000	0.00000	9.000000	12.000000	0.000000	0.00000
25%	0.000000	0.000000	0.000000	1.000000	0.000000	2877.500000	0.00000	100.250000	360.000000	1.000000	0.00000
50%	0.000000	1.000000	0.000000	1.000000	0.000000	3812.500000	1188.50000	125.000000	360.000000	1.000000	1.00000
75%	0.000000	1.000000	1.000000	1.000000	0.000000	5795.000000	2297.25000	164.750000	360.000000	1.000000	2.00000
max	1.000000	1.000000	3.000000	1.000000	1.000000	81000.000000	41667.00000	700.000000	480.000000	1.000000	2.00000

Univariate Analysis

```
plt.figure(figsize=(5,3))
plt.subplot(121)
sns.distplot(data['ApplicantIncome'],color='r')
plt.subplot(122)
sns.displot(data['Credit_History'],color='b',kde=True)
plt.show()
```

```
<ipython-input-18-b806fb277601>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```


1s completed at 7:26 PM

Hain.ipynb

File Edit View Insert Runtime Tools Help All changes saved

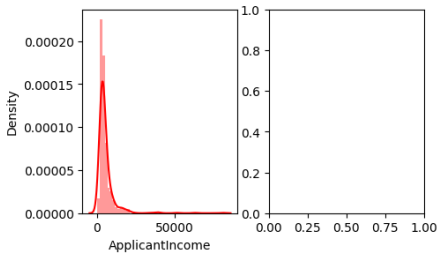
Comment Share

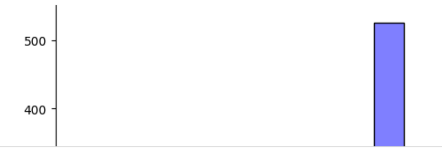
+ Code + Text

RAM Disk

```
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

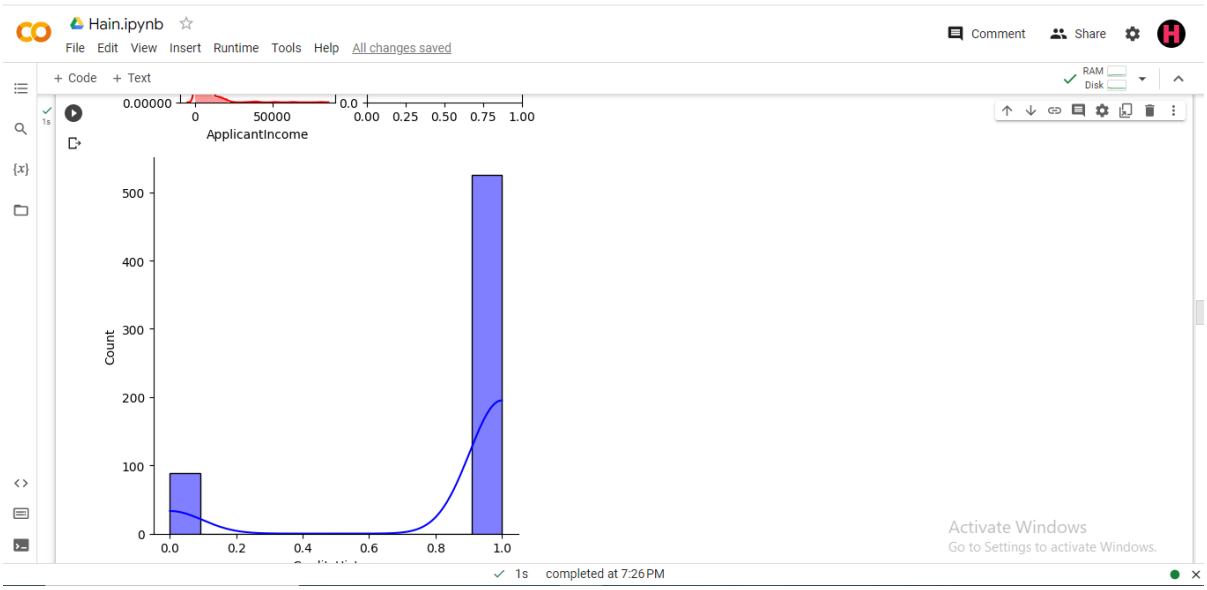
```
sns.distplot(data['ApplicantIncome'],color='r')
```

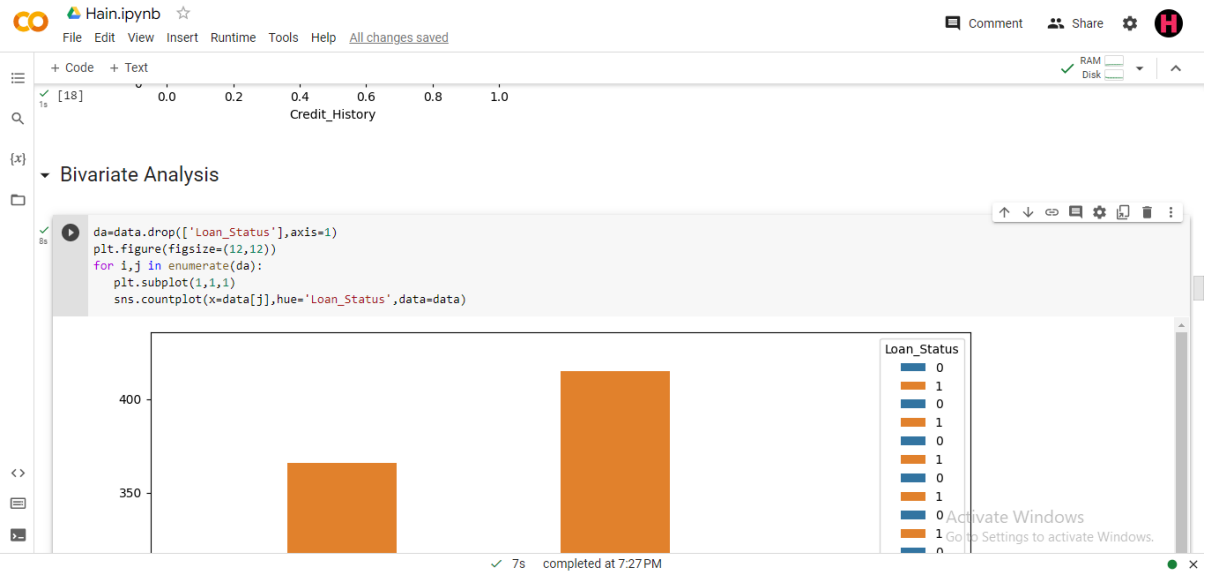




Activate Windows
Go to Settings to activate Windows.

1s completed at 7:26 PM


Hain.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
0.00000
0
50000
0.0
0.00 0.25 0.50 0.75 1.00
ApplicantIncome
500
400
300
200
100
0
Count
0.0 0.2 0.4 0.6 0.8 1.0
Activate Windows
Go to Settings to activate Windows.
1s completed at 7:26 PM


Hain.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[18]
0.0 0.2 0.4 0.6 0.8 1.0
Credit_History
Bivariate Analysis
da=data.drop(['Loan_Status'],axis=1)
plt.figure(figsize=(12,12))
for i,j in enumerate(da):
plt.subplot(1,1,1)
sns.countplot(x=data[j],hue='Loan_Status',data=data)
400
350
Loan_Status
0
1
0
1
0
1
0
1
0
1
Activate Windows
Go to Settings to activate Windows.
7s completed at 7:27 PM

Hain.ipynb

File Edit View Insert Runtime Tools Help

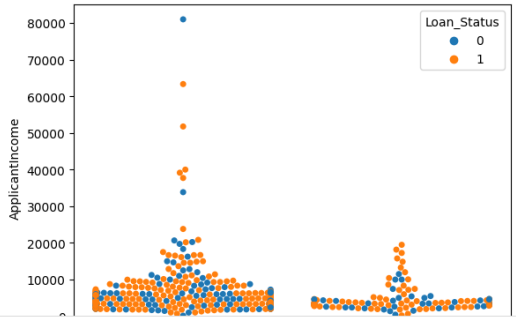
Comment Share

+ Code + Text

RAM Disk

```
sns.swarmplot(x=data['Gender'],y=data['ApplicantIncome'],hue=data['Loan_Status'])
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py:3544: UserWarning: 45.8% of the points cannot be placed; you may want to decrease the size of the ma
  warnings.warn(msg, UserWarning)
<Axes: xlabel='Gender',
ylabel='ApplicantIncome'>/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py:3544: UserWarning: 61.6% of the points cannot be placed; you may want to de
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py:3544: UserWarning: 25.0% of the points cannot be placed; you may want to decrease the size of the ma
  warnings.warn(msg, UserWarning)
```



11s completed at 7:28 PM

Hain.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

Spliting data into train and test

```
[21] x=data.drop(columns=['Loan_Status'],axis=1)
     y=data['Loan_Status']
     x_train,x_test,y_train,y_test=train_test_split(x_bal,y_bal,test_size=0.33,random_state=42)
```

Model Building

Decision Tree Model

```
def decisionTree(x_train,x_test,y_train,y_test):
 dt=DecisionTreeClassifier()
 x=data.drop(columns=['Loan_Status'],axis=1)
 y=data['Loan_Status']
 x_train,x_test,y_train,y_test=train_test_split(x_bal,y_bal,test_size=0.33,random_state=42)
 dt.fit(x_train,y_train)
 yPred=data.predict(x_test)
 print('***DecisionTreeClassifier***')
 print('confusion matrix')
 print(confusion_matrix(y_test,yPred))
```


0s completed at 7:28 PM

Hain.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk

Random Forest Model

```
[23] 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(classification_report(y_test,yPred))
```

KNN

```
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))
```


Xgboost model

0s completed at 7:29 PM

Hain.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

Xgboost model

```
[25] 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))
```

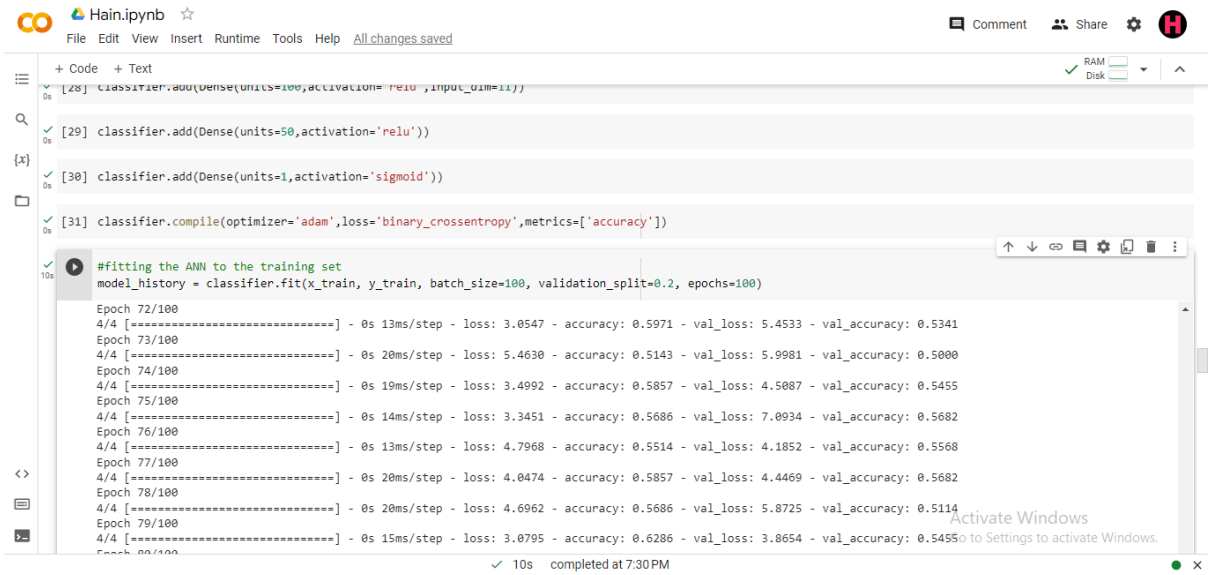
ANN model

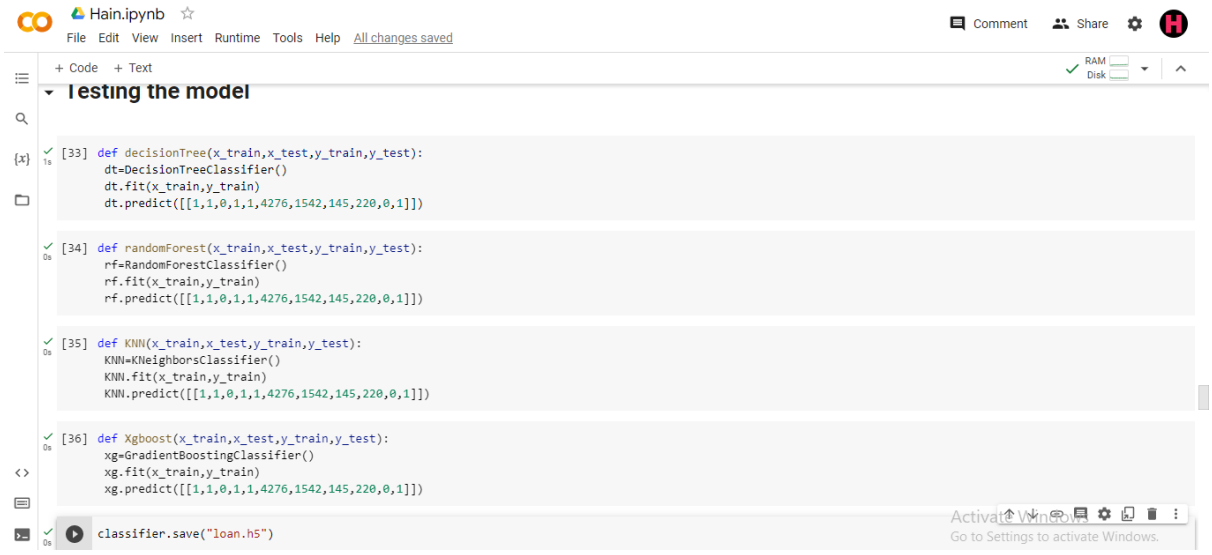
```
[26] import tensorflow
     from tensorflow import keras
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense
```

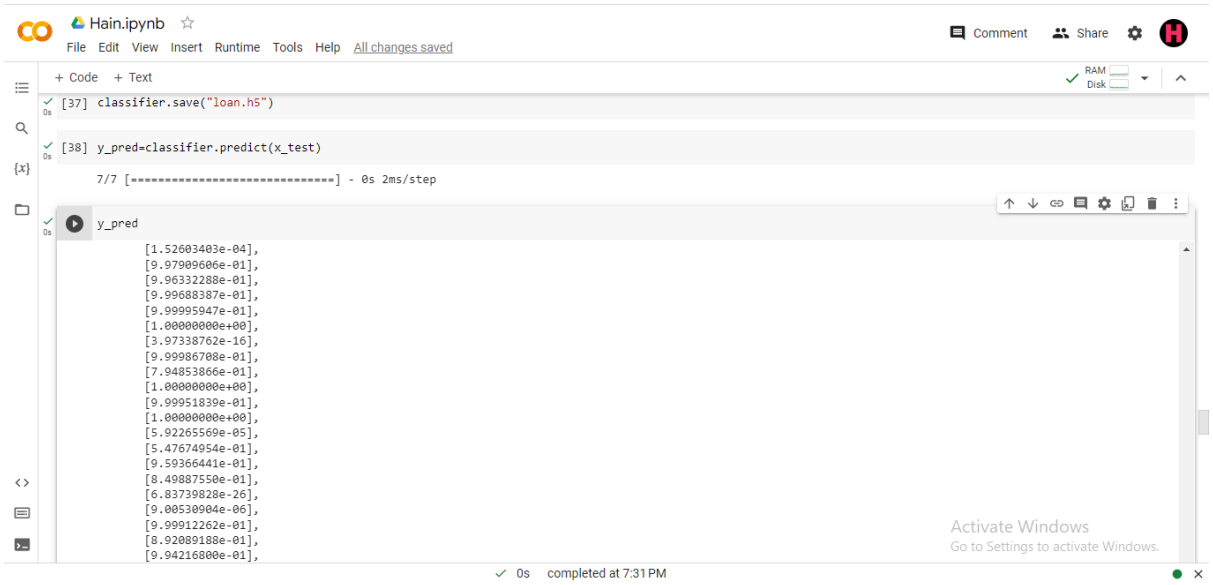
```
[27] classifier=Sequential()
```

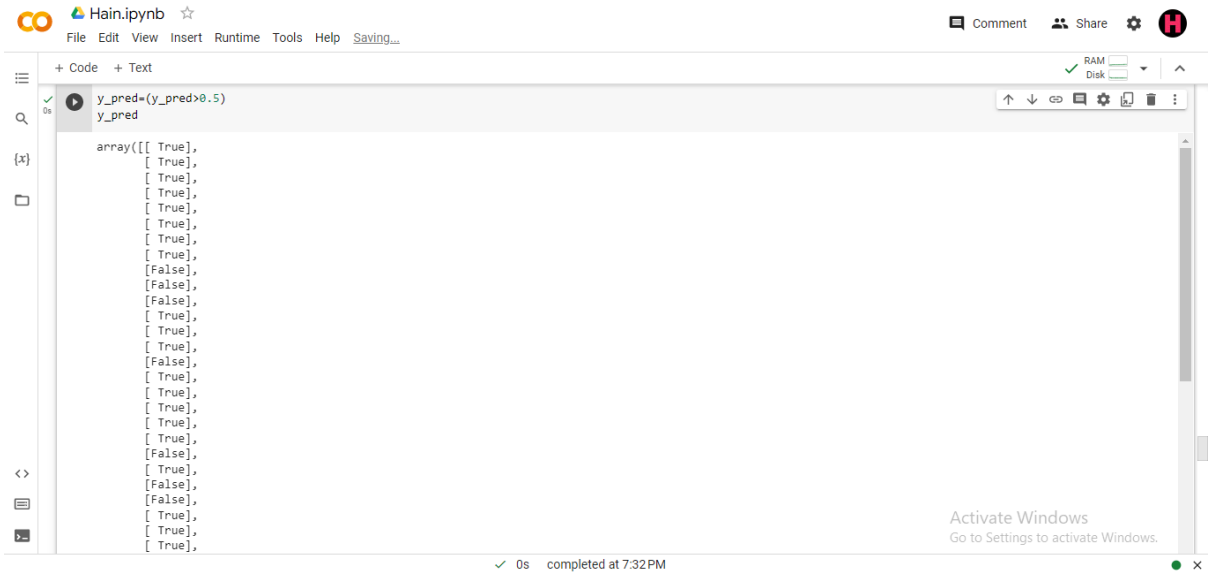
```
classifier.add(Dense(units=100,activation='relu',input_dim=11))
```


0s completed at 7:29 PM

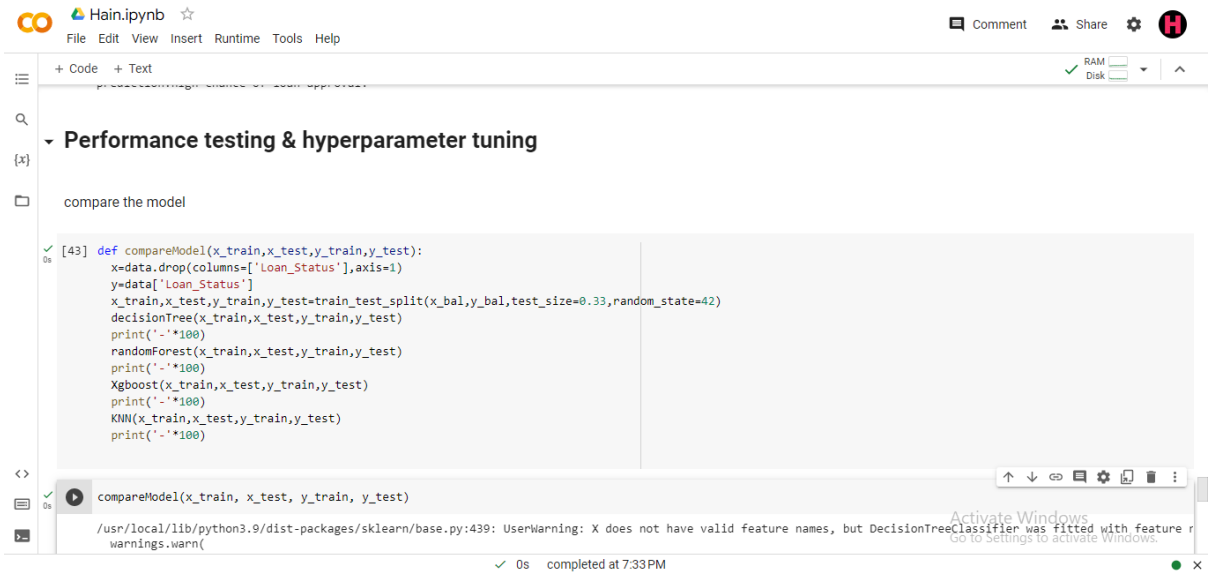

Hain.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[29] classifier.add(Dense(units=50,activation='relu'))
[30] classifier.add(Dense(units=1,activation='sigmoid'))
[31] classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
#fitting the ANN to the training set
model_history = classifier.fit(x_train, y_train, batch_size=100, validation_split=0.2, epochs=100)
Epoch 72/100
4/4 [==============================] - 0s 13ms/step - loss: 3.0547 - accuracy: 0.5971 - val_loss: 5.4533 - val_accuracy: 0.5341
Epoch 73/100
4/4 [==============================] - 0s 20ms/step - loss: 5.4630 - accuracy: 0.5143 - val_loss: 5.9981 - val_accuracy: 0.5000
Epoch 74/100
4/4 [==============================] - 0s 19ms/step - loss: 3.4992 - accuracy: 0.5857 - val_loss: 4.5087 - val_accuracy: 0.5455
Epoch 75/100
4/4 [==============================] - 0s 14ms/step - loss: 3.3451 - accuracy: 0.5686 - val_loss: 7.0934 - val_accuracy: 0.5682
Epoch 76/100
4/4 [==============================] - 0s 13ms/step - loss: 4.7968 - accuracy: 0.5514 - val_loss: 4.1852 - val_accuracy: 0.5568
Epoch 77/100
4/4 [==============================] - 0s 20ms/step - loss: 4.0474 - accuracy: 0.5857 - val_loss: 4.4469 - val_accuracy: 0.5682
Epoch 78/100
4/4 [==============================] - 0s 20ms/step - loss: 4.6962 - accuracy: 0.5686 - val_loss: 5.8725 - val_accuracy: 0.5114
Epoch 79/100
4/4 [==============================] - 0s 15ms/step - loss: 3.0795 - accuracy: 0.6286 - val_loss: 3.8654 - val_accuracy: 0.5455
Activate Windows
Go to Settings to activate Windows.
10s completed at 7:30 PM

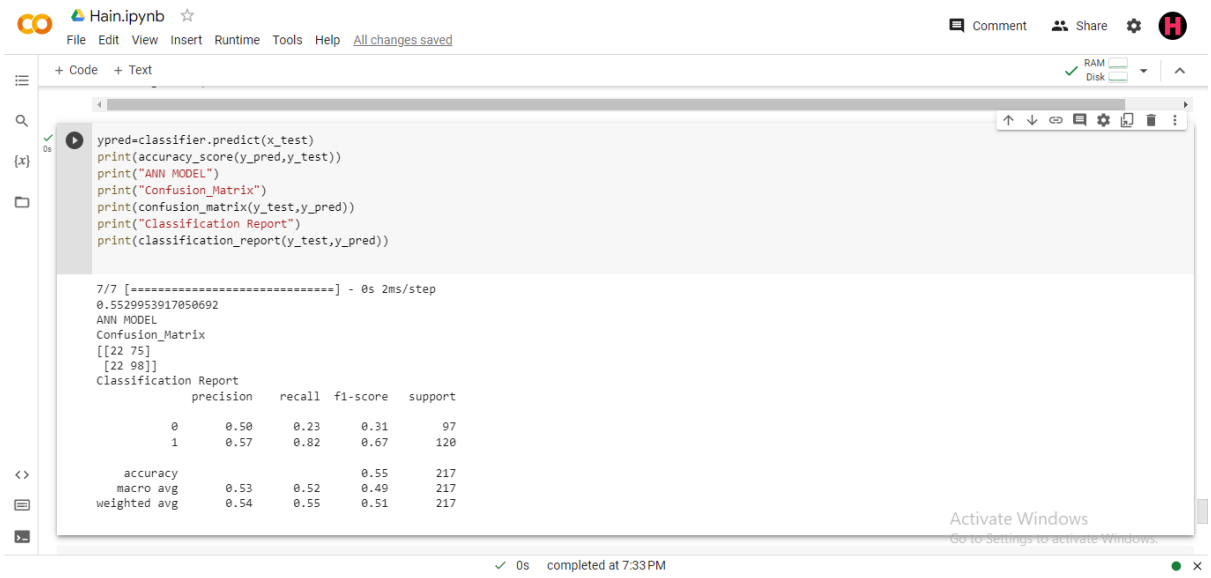

Hain.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Testing the model
[33] def decisionTree(x_train,x_test,y_train,y_test):
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
dt.predict([[1,1,0,1,1,4276,1542,145,220,0,1]])
[34] def randomForest(x_train,x_test,y_train,y_test):
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
rf.predict([[1,1,0,1,1,4276,1542,145,220,0,1]])
[35] def KNN(x_train,x_test,y_train,y_test):
KNN=KNeighborsClassifier()
KNN.fit(x_train,y_train)
KNN.predict([[1,1,0,1,1,4276,1542,145,220,0,1]])
[36] def Xgboost(x_train,x_test,y_train,y_test):
xg=GradientBoostingClassifier()
xg.fit(x_train,y_train)
xg.predict([[1,1,0,1,1,4276,1542,145,220,0,1]])
classifier.save("loan.h5")
Activate Windows
Go to Settings to activate Windows.


Hain.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[37] classifier.save("loan.h5")
[38] y_pred=classifier.predict(x_test)
7/7 [==============================] - 0s 2ms/step
y_pred
[1.52603403e-04],
[9.97909606e-01],
[9.96332288e-01],
[9.99688387e-01],
[9.99995947e-01],
[1.00000000e+00],
[3.97338762e-16],
[9.99986708e-01],
[7.94853866e-01],
[1.00000000e+00],
[9.99951839e-01],
[1.00000000e+00],
[5.92265569e-05],
[5.47674954e-01],
[9.59366441e-01],
[8.49887550e-01],
[6.83739828e-26],
[9.00530904e-06],
[9.99912262e-01],
[8.92089188e-01],
[9.94216800e-01],
Activate Windows
Go to Settings to activate Windows.
0s completed at 7:31 PM


Hain.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment
Share
+ Code + Text
RAM
Disk
y_pred=(y_pred>0.5)
y_pred
array([[True],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[False],
[False],
[False],
[True],
[True],
[True],
[False],
[True],
[True],
[True],
[True],
[True],
[False],
[True],
[False],
[False],
[True],
[True],
[True],
Activate Windows
Go to Settings to activate Windows.
0s completed at 7:32 PM

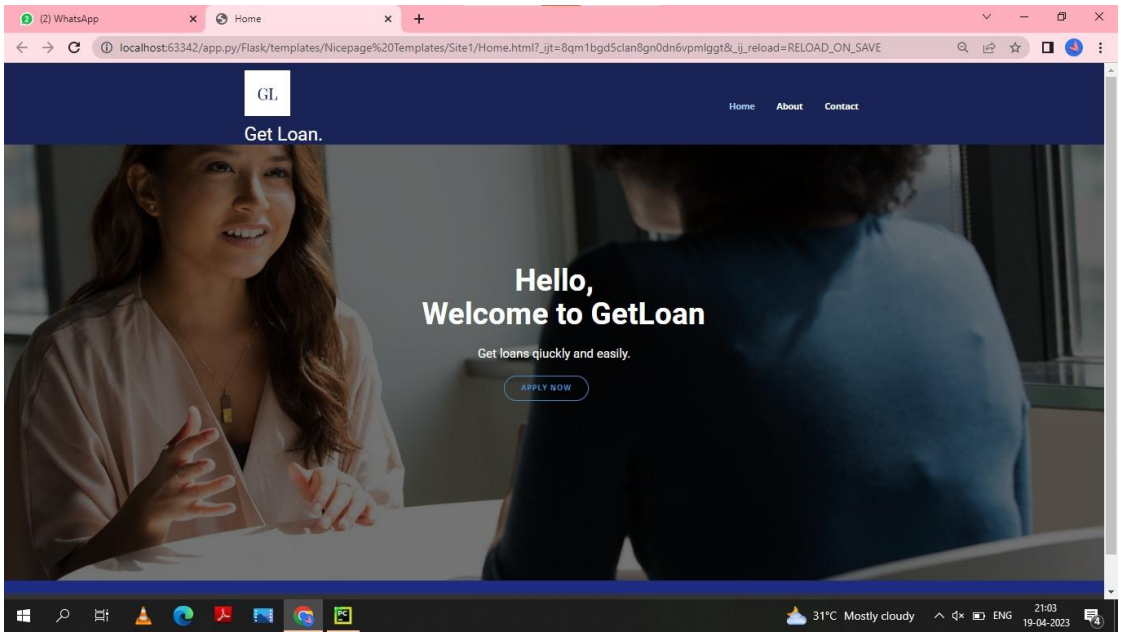

Hain.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[40] [True],
[True],
[True],
[True],
[True],
[True],
[True],
[True],
[False]])
[41] def predict_exit(sample_value):
sample_value=np.array(sample_value)
sample_value=sample_value.reshape(1,-1)
return classifier.predict(sample_value)
sample_value=[[1,1,0,1,1,4276,1542,145,240,0,1]]
if predict_exit(sample_value).all()>0.5:
print('prediction:high chance of loan approval!')
else:
print('prediction:low chance loan approval.')
1/1 [==============================] - 0s 24ms/step
prediction:high chance of loan approval!
Activate Windows
Go to Settings to activate Windows.
0s completed at 7:32 PM

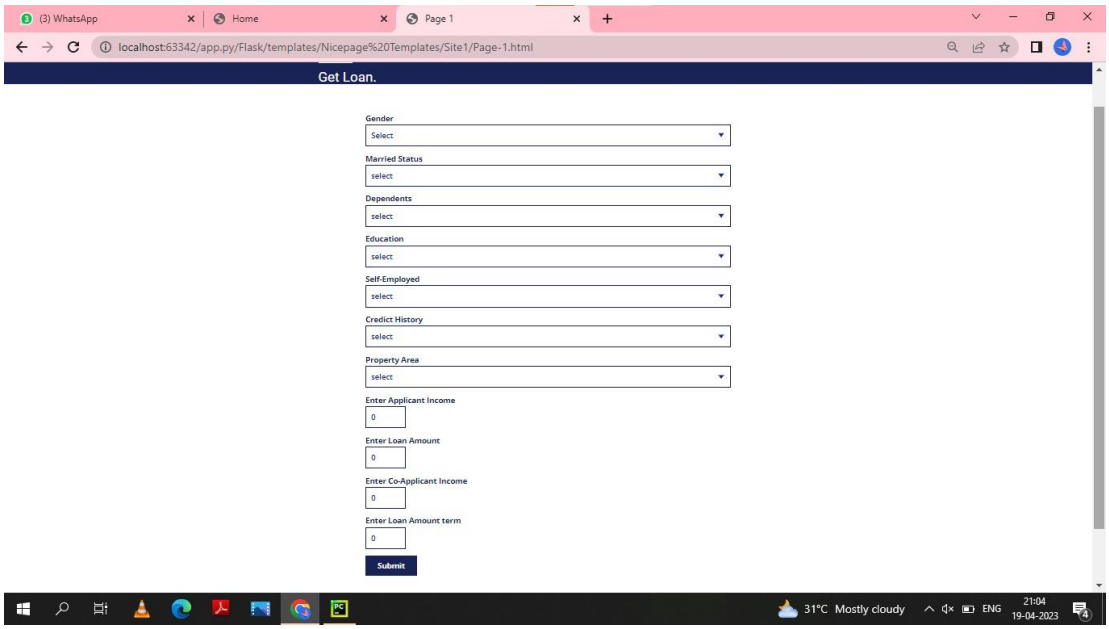

Hain.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Performance testing & hyperparameter tuning
compare the model
[43] def compareModel(x_train,x_test,y_train,y_test):
x=data.drop(columns=['Loan_Status'],axis=1)
y=data['Loan_Status']
x_train,x_test,y_train,y_test=train_test_split(x_bal,y_bal,test_size=0.33,random_state=42)
decisionTree(x_train,x_test,y_train,y_test)
print('-'*100)
randomForest(x_train,x_test,y_train,y_test)
print('-'*100)
Xgboost(x_train,x_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)
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature r
warnings.warn(
Activate Windows
Go to Settings to activate Windows.
0s completed at 7:33 PM

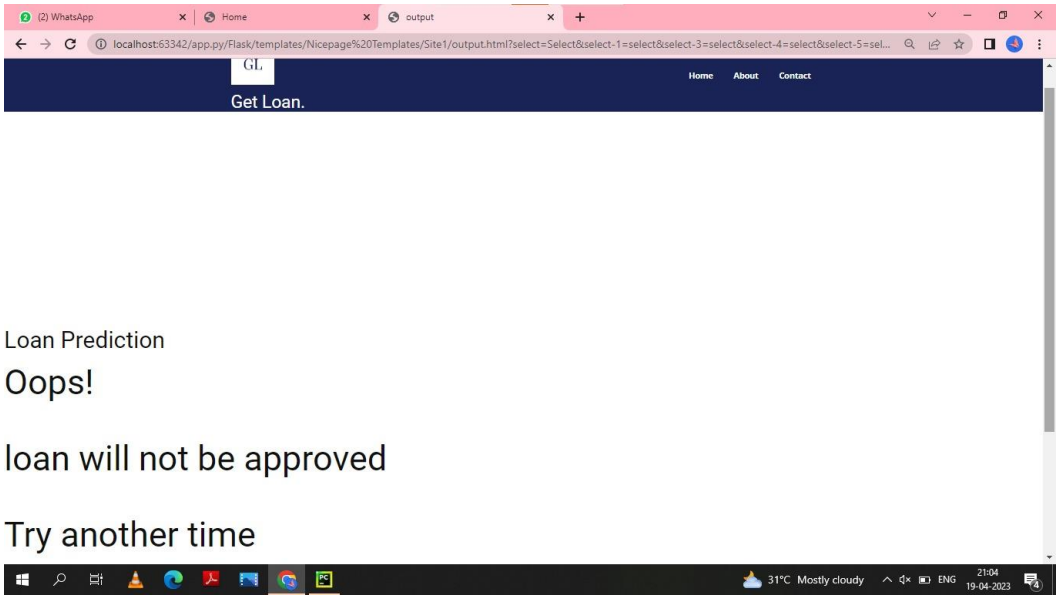

Hain.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
ypred=classifier.predict(x_test)
print(accuracy_score(y_pred,y_test))
print("ANN MODEL")
print("Confusion_Matrix")
print(confusion_matrix(y_test,y_pred))
print("Classification Report")
print(classification_report(y_test,y_pred))
7/7 [==============================] - 0s 2ms/step
0.5529953917050692
ANN MODEL
Confusion_Matrix
[[22 75]
[22 98]]
Classification Report
precision recall f1-score support
0 0.50 0.23 0.31 97
1 0.57 0.82 0.67 120
accuracy 0.55 217
macro avg 0.53 0.52 0.49 217
weighted avg 0.54 0.55 0.51 217
Activate Windows
Go to Settings to activate Windows.
0s completed at 7:33 PM


Hain.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment
Share
+ Code + Text
RAM
Disk
[45] macro avg 0.53 0.52 0.49 217
weighted avg 0.54 0.55 0.51 217
[46] from sklearn.model_selection import cross_val_score
[47] #random forest model is selected
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
yPred=rf.predict(x_test)
print('-'*100)
[48] f1_score(yPred,y_test,average='weighted')
0.8356206005975592
[49] cv=cross_val_score(rf,x,y,cv=5)
np.mean(cv)
0.7817539650806344
Activate Windows
Go to Settings to activate Windows.
0s completed at 7:35 PM


Hain.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[49] cv=cross_val_score(rf,x,y,cv=5)
[50] np.mean(cv)
0.7817539650806344
[51] model=decisionTree,randomForest,KNN,Xgboost,(x_train,x_test,y_train,y_test)
[52] pickle.dump(model,open('rdf.pkl','wb'))
pickle.dump(model,open('scale1.pkl','wb'))
Activate Windows
Go to Settings to activate Windows.
0s completed at 7:35PM


(2) WhatsApp
Home
localhost:63342/app.py/Flask/templates/Nicepage%20Templates/Site1/Home.html?_ijt=8qm1bgd5clan8gn0dn6vpmlggt&_ij_reload=RELOAD_ON_SAVE
GL
Get Loan.
Home
About
Contact
Hello,
Welcome to GetLoan
Get loans qiuckly and easily.
APPLY NOW
31°C Mostly cloudy
ENG
21:03
19-04-2023


(3) WhatsApp
Home
Page 1
localhost:63342/app.py/Flask/templates/Nicepage%20Templates/Site1/Page-1.html
Get Loan.
Gender
Select
Married Status
select
Dependents
select
Education
select
Self-Employed
select
Credict History
select
Property Area
select
Enter Applicant Income
0
Enter Loan Amount
0
Enter Co-Applicant Income
0
Enter Loan Amount term
0
Submit
31°C Mostly cloudy
ENG
21:04
19-04-2023



ADVANTAGES:

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.

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.

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.

APPLICATIONS:

Predictive analytics using machine learning helps detect fraudulent activities in the financial sector. Fraudulent transactions are identified by training machine learning algorithms with past datasets. The models find risky patterns in these datasets and learn to predict and deter fraud.

Various financial tasks utilize machine learning technology, including credit scoring, investment monitoring and recommendations, fraud detection, and algorithmic trading. Machine learning can help financial companies make better pricing, risk, and customer behaviour decisions.

CONCLUSION:

We did Exploratory data Analysis on the features of this dataset and saw how each feature is distributed.

We did bivariate and multivariate analysis to see impact of one another on their features using charts.

We analysed each variable to check if data is cleaned and normally distributed.

We calculated correlation between independent variables and found that applicant income and loan amount have significant relation.

We constructed models taking different variables into account and found through odds ratio that credit credit history is creating the most impact on loan giving decision.

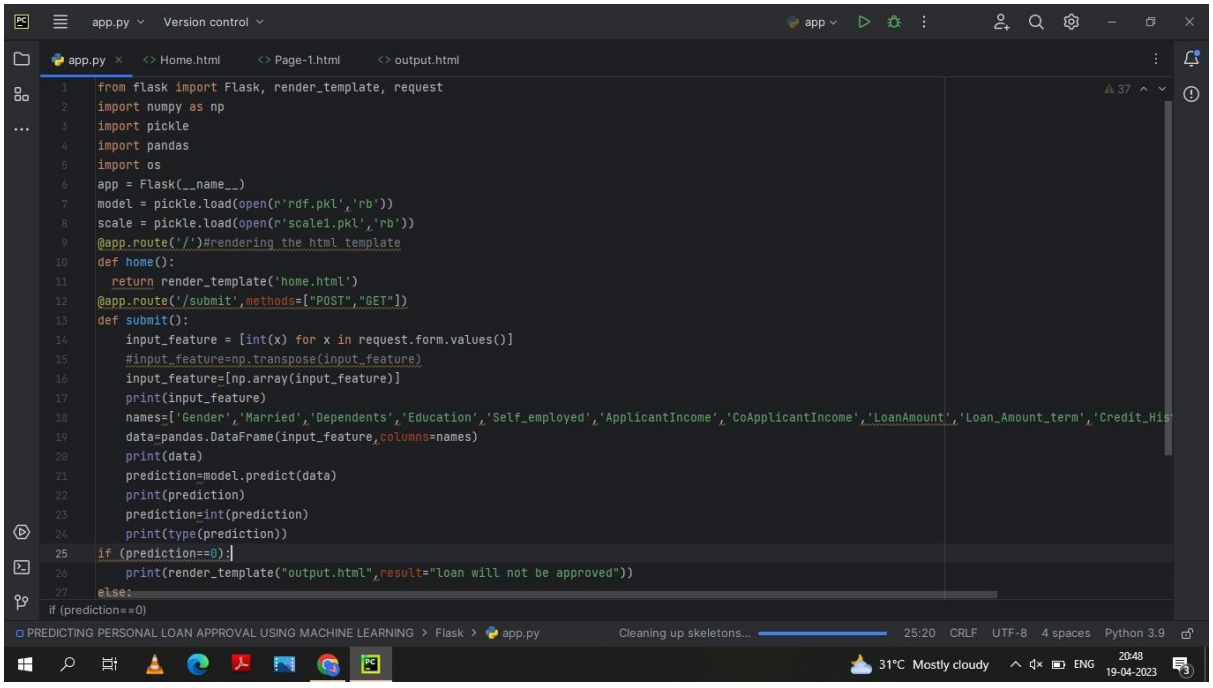
FUTURE SCOPE:

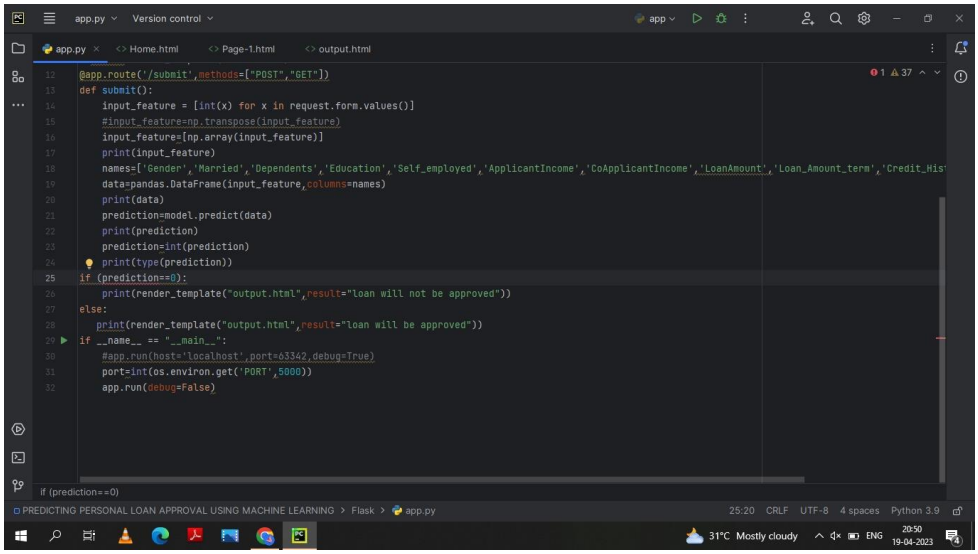
Generally, loan prediction involves the lender looking at various background information about the applicant and deciding whether the bank should grant the loan. Parameters like credit score, loan amount, lifestyle, career, and assets are the deciding factors in getting the loan approved.

APPENDIX:

SOURCE CODE:

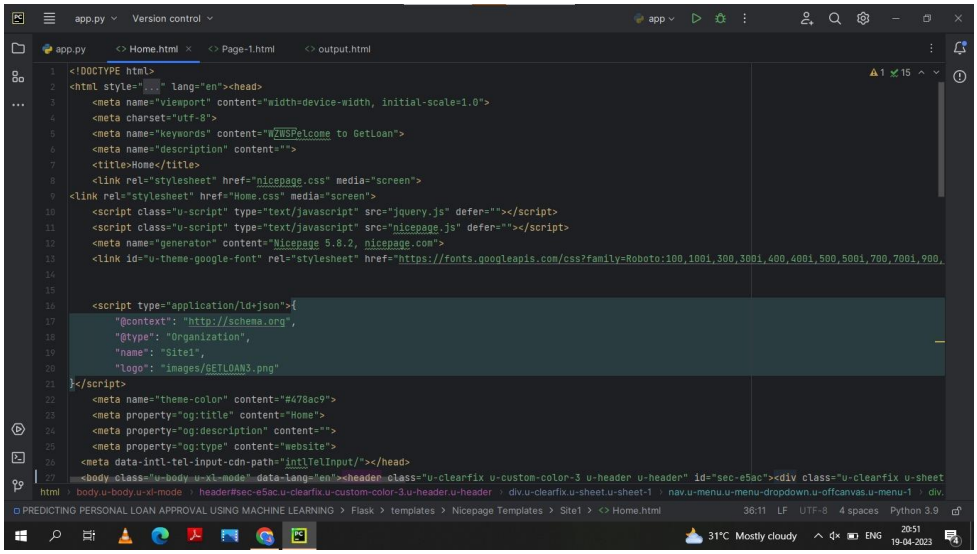
APP.PY



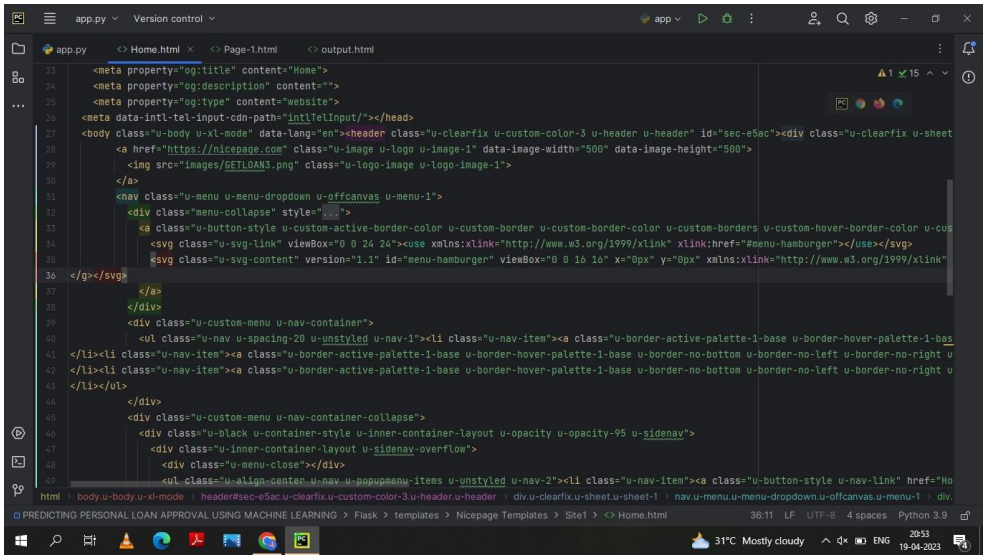


```
@app.route('/submit',methods=["POST","GET"])
def submit():
    input_feature = [int(x) for x in request.form.values()]
    #input_feature=np.transpose(input_feature)
    input_feature=[np.array(input_feature)]
    print(input_feature)
    names=['Gender','Married','Dependents','Education','Self_employed','ApplicantIncome','CoApplicantIncome','LoanAmount','Loan_Amount_term','Credit_His
    data=pandas.DataFrame(input_feature,columns=names)
    print(data)
    prediction=model.predict(data)
    print(prediction)
    prediction=int(prediction)
    print(type(prediction))
if (prediction==0):
    print(render_template("output.html",result="loan will not be approved"))
else:
   print(render_template("output.html",result="loan will be approved"))
if __name__ == "__main__":
    #app.run(host='localhost',port=63342,debug=True)
    port=int(os.environ.get('PORT',5000))
    app.run(debug=False)
```

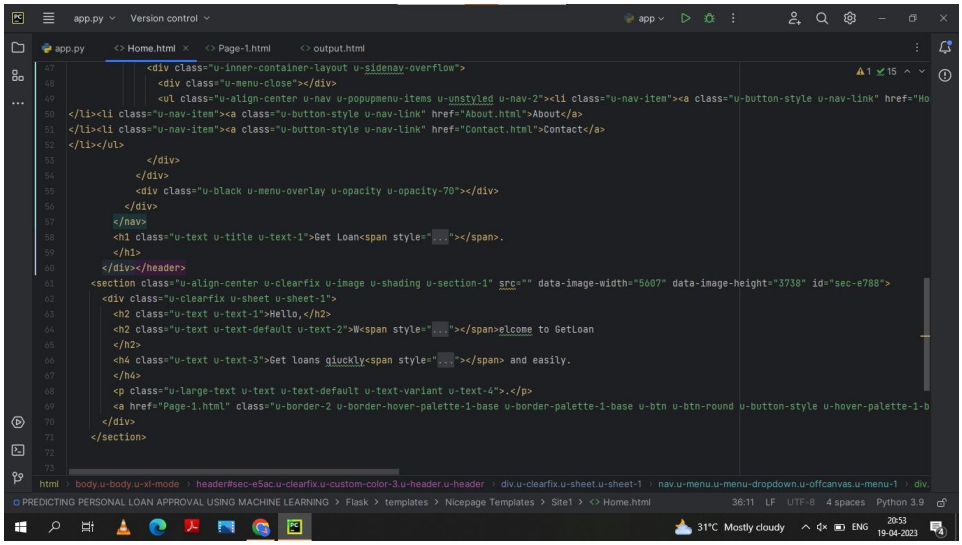
HOME.HTML



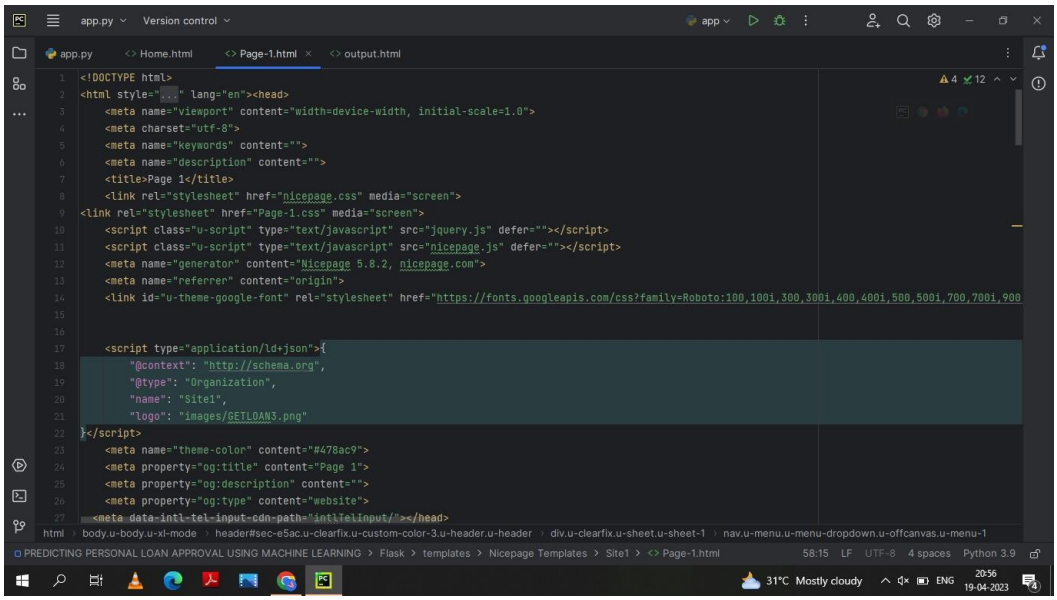
```
<!DOCTYPE html>
<html style="..." lang="en"><head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta charset="utf-8">
    <meta name="keywords" content="WZWSPelcome to GetLoan">
    <meta name="description" content="">
    <title>Home</title>
    <link rel="stylesheet" href="nicepage.css" media="screen">
<link rel="stylesheet" href="Home.css" media="screen">
    <script class="u-script" type="text/javascript" src="jquery.js" defer=""></script>
    <script class="u-script" type="text/javascript" src="nicepage.js" defer=""></script>
    <meta name="generator" content="Nicepage 5.8.2, nicepage.com">
    <link id="u-theme-google-font" rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:100,100i,300,300i,400,400i,500,500i,700,700i,900,


    <script type="application/ld+json">{
        "@context": "http://schema.org",
        "@type": "Organization",
        "name": "Site1",
        "logo": "images/GETLOAN3.png"
}</script>
    <meta name="theme-color" content="#478ac9">
    <meta property="og:title" content="Home">
    <meta property="og:description" content="">
    <meta property="og:type" content="website">
  <meta data-intl-tel-input-cdn-path="intlTelInput/"></head>
  <body class="u-body u-xl-mode" data-lang="en"><header class="u-clearfix u-custom-color-3 u-header u-header" id="sec-e5ac"><div class="u-clearfix u-sheet
```

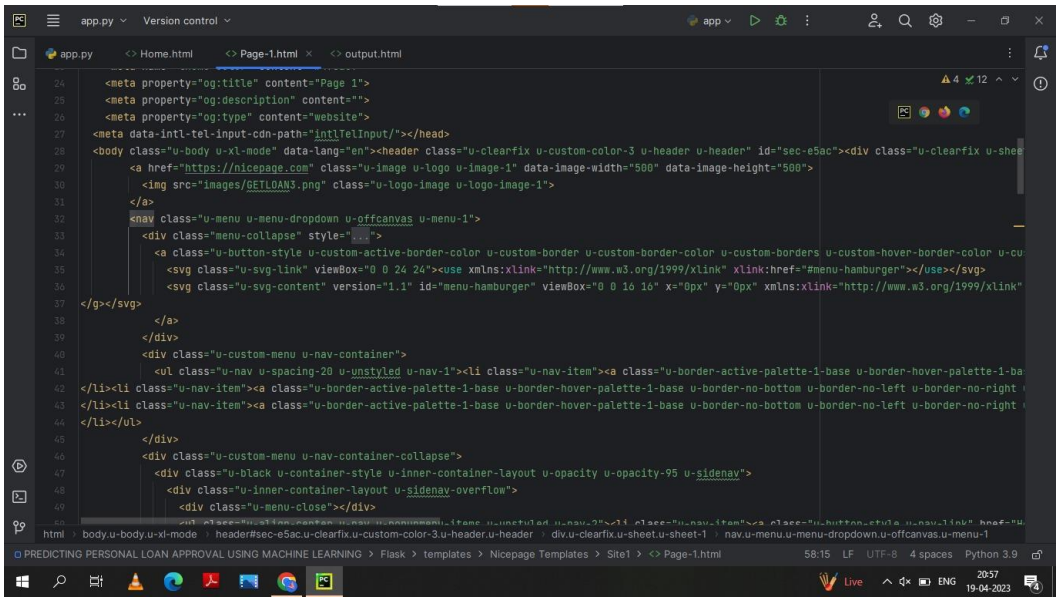


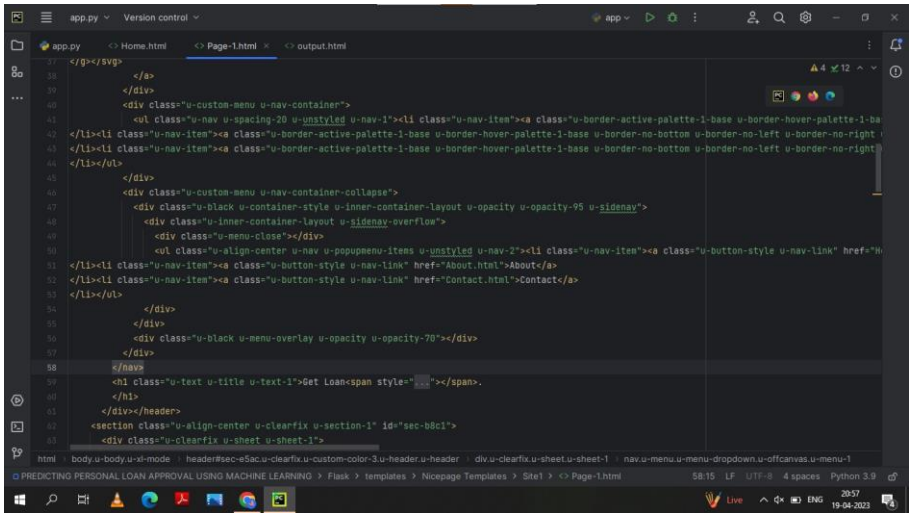
```
    <meta property="og:title" content="Home">
    <meta property="og:description" content="">
    <meta property="og:type" content="website">
  <meta data-intl-tel-input-cdn-path="intlTelInput/"></head>
  <body class="u-body u-xl-mode" data-lang="en"><header class="u-clearfix u-custom-color-3 u-header u-header" id="sec-e5ac"><div class="u-clearfix u-sheet
        <a href="https://nicepage.com" class="u-image u-logo u-image-1" data-image-width="500" data-image-height="500">
          <img src="images/GETLOAN3.png" class="u-logo-image u-logo-image-1">
        </a>
        <nav class="u-menu u-menu-dropdown u-offcanvas u-menu-1">
          <div class="menu-collapse" style="...">
            <a class="u-button-style u-custom-active-border-color u-custom-border u-custom-border-color u-custom-borders u-custom-hover-border-color u-cus
              <svg class="u-svg-link" viewBox="0 0 24 24"><use xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#menu-hamburger"></use></svg>
              <svg class="u-svg-content" version="1.1" id="menu-hamburger" viewBox="0 0 16 16" x="0px" y="0px" xmlns:xlink="http://www.w3.org/1999/xlink"
</g></svg>
            </a>
          </div>
          <div class="u-custom-menu u-nav-container">
            <ul class="u-nav u-spacing-20 u-unstyled u-nav-1"><li class="u-nav-item"><a class="u-border-active-palette-1-base u-border-hover-palette-1-bas
</li><li class="u-nav-item"><a class="u-border-active-palette-1-base u-border-hover-palette-1-base u-border-no-bottom u-border-no-left u-border-no-right u
</li><li class="u-nav-item"><a class="u-border-active-palette-1-base u-border-hover-palette-1-base u-border-no-bottom u-border-no-left u-border-no-right u
</li></ul>
          </div>
          <div class="u-custom-menu u-nav-container-collapse">
            <div class="u-black u-container-style u-inner-container-layout u-opacity u-opacity-95 u-sidenav">
              <div class="u-inner-container-layout u-sidenav-overflow">
                <div class="u-menu-close"></div>
                <ul class="u-align-center u-nav u-popupmenu-items u-unstyled u-nav-2"><li class="u-nav-item"><a class="u-button-style u-nav-link" href="Ho
```

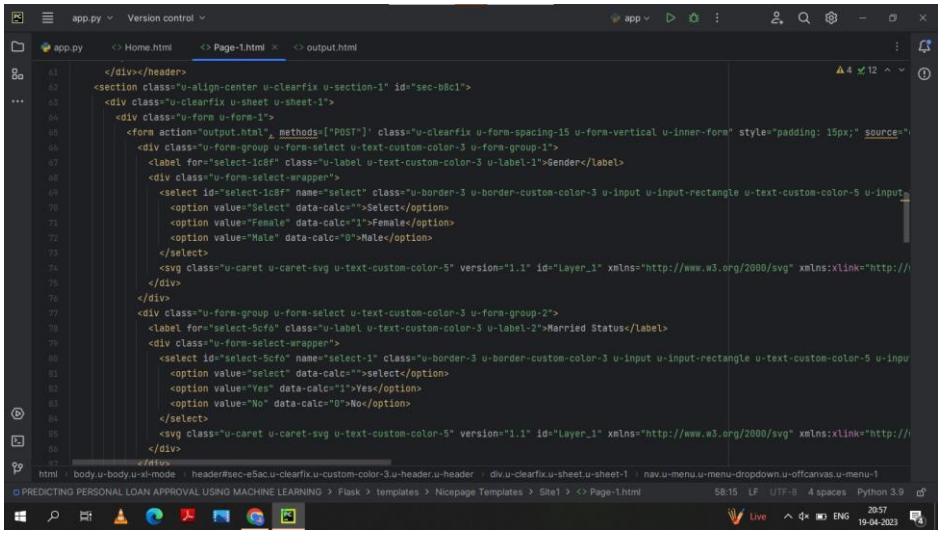


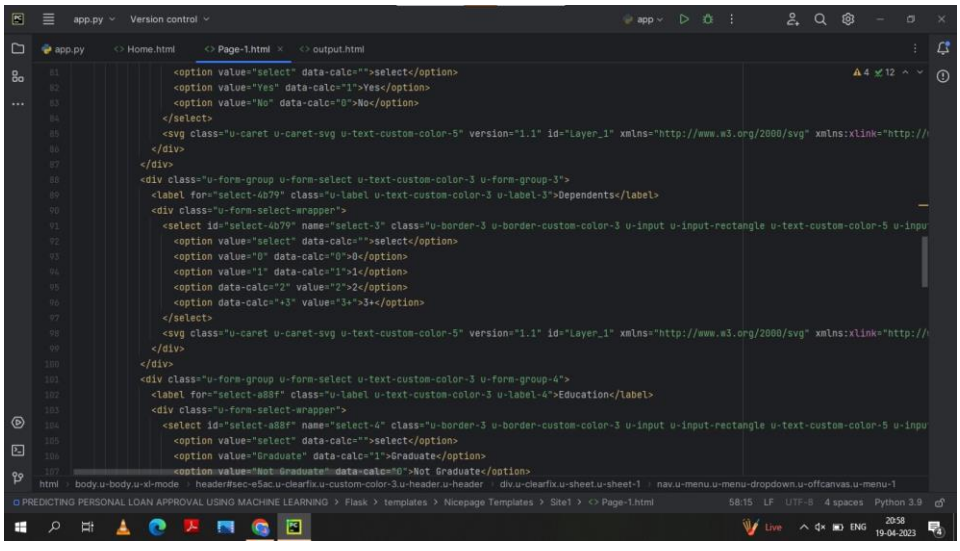
PAGE1.HTML:

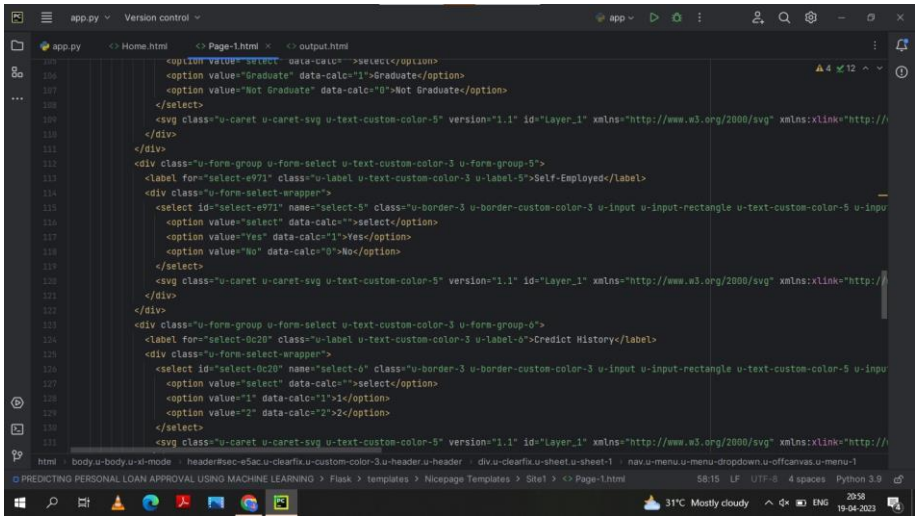


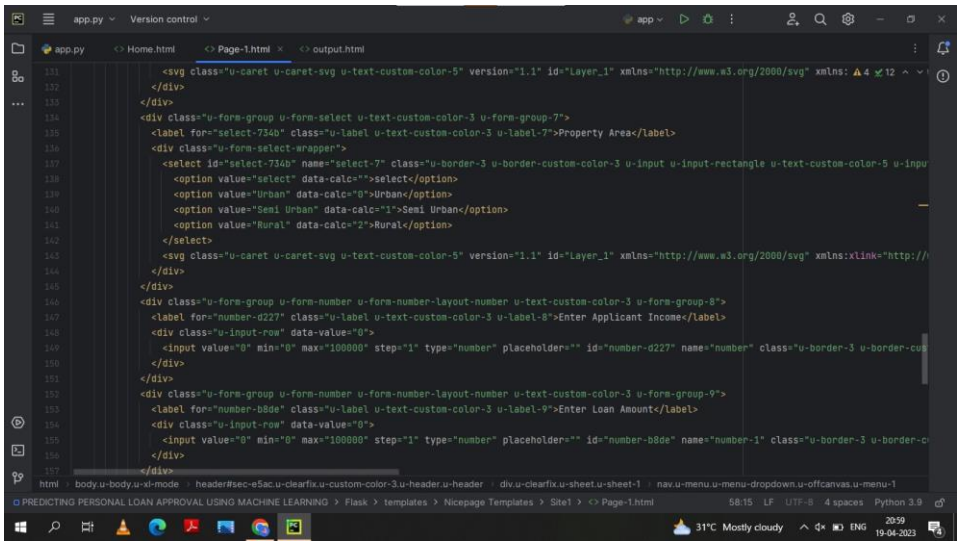


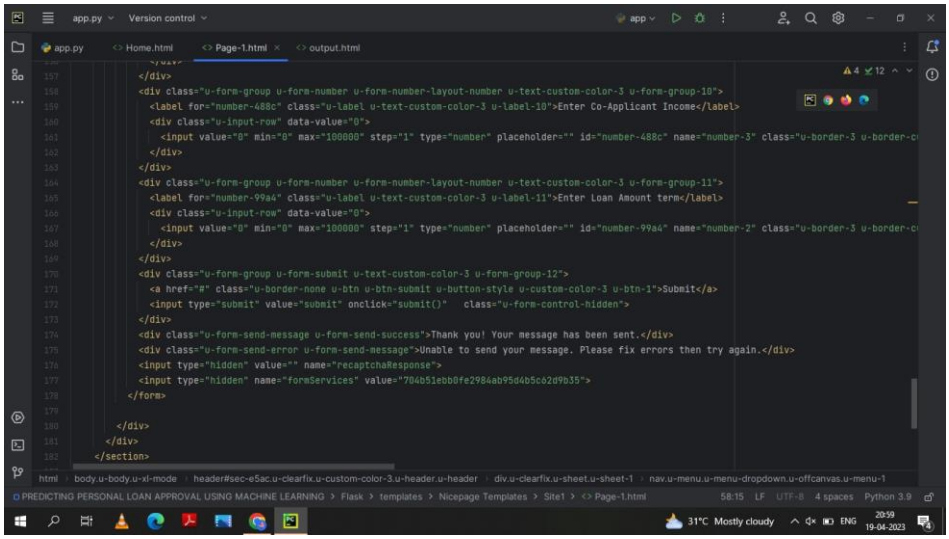


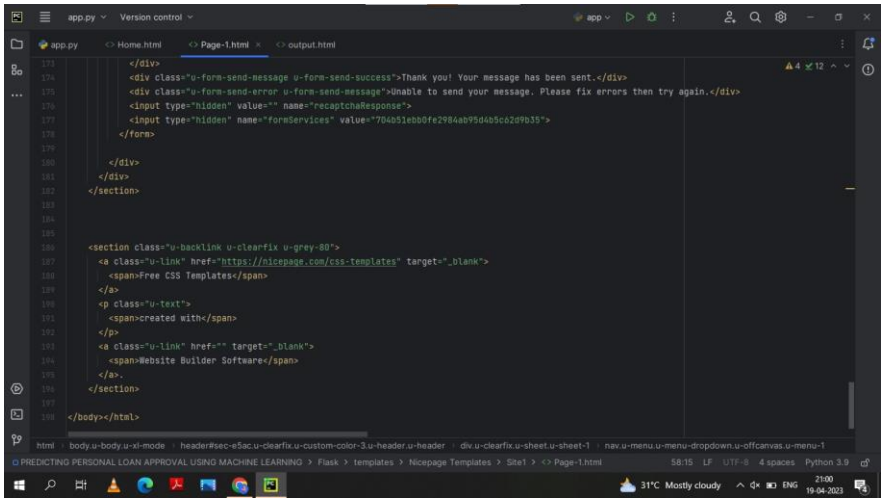




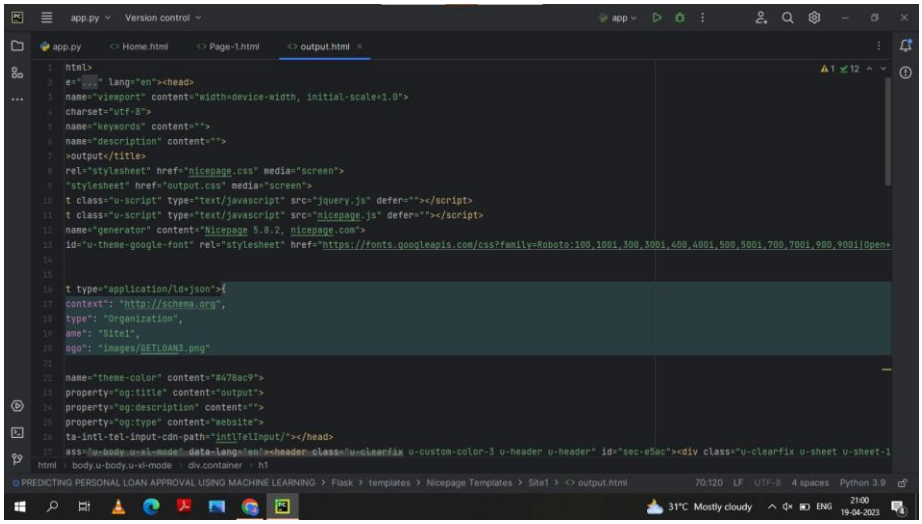

app.py
Version control
Home.html
Page-1.html
output.html
<option value="Graduate" data-calc="1">Graduate</option>
<option value="Not Graduate" data-calc="0">Not Graduate</option>
<label for="select-e971" class="u-label u-text-custom-color-3 u-label-5">Self-Employed</label>
<option value="Yes" data-calc="1">Yes</option>
<option value="No" data-calc="0">No</option>
<label for="select-0c20" class="u-label u-text-custom-color-3 u-label-6">Credict History</label>
<option value="1" data-calc="1">1</option>
<option value="2" data-calc="2">2</option>
PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING
Flask
templates
Nicepage Templates
Site1
Page-1.html
31°C Mostly cloudy
ENG
20:58
19-04-2023

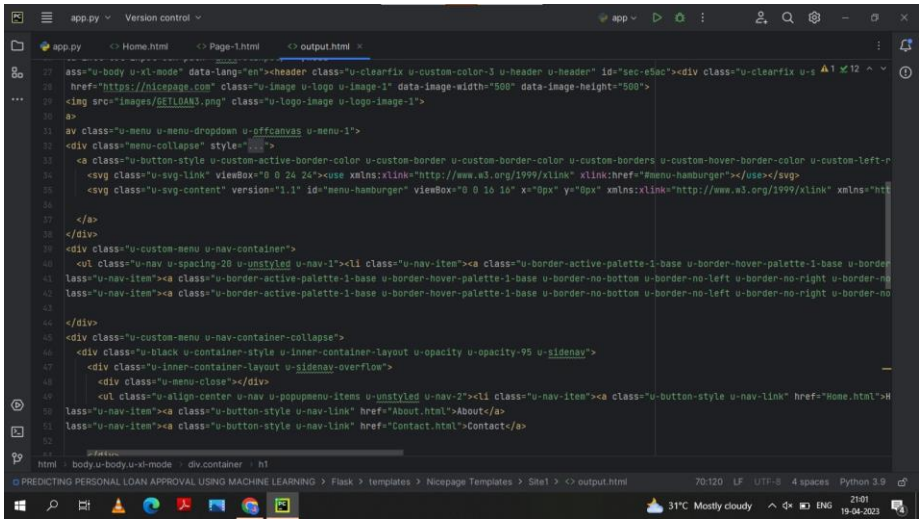

app.py
Version control
Home.html
Page-1.html
output.html
<label for="select-734b" class="u-label u-text-custom-color-3 u-label-7">Property Area</label>
<option value="Urban" data-calc="0">Urban</option>
<option value="Semi Urban" data-calc="1">Semi Urban</option>
<option value="Rural" data-calc="2">Rural</option>
<label for="number-d227" class="u-label u-text-custom-color-3 u-label-8">Enter Applicant Income</label>
<label for="number-b8de" class="u-label u-text-custom-color-3 u-label-9">Enter Loan Amount</label>
PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING
Flask
templates
Nicepage Templates
Site1
Page-1.html
31°C Mostly cloudy
ENG
20:59
19-04-2023

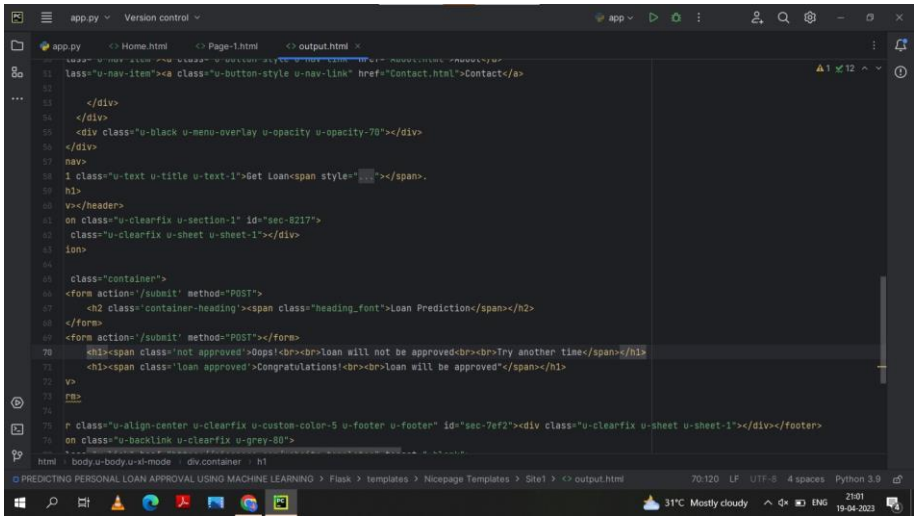

app.py
Version control
Home.html
Page-1.html
output.html
<label for="number-488c" class="u-label u-text-custom-color-3 u-label-10">Enter Co-Applicant Income</label>
<label for="number-99a4" class="u-label u-text-custom-color-3 u-label-11">Enter Loan Amount term</label>
Submit
<input type="submit" value="submit" onclick="submit()" class="u-form-control-hidden">
Thank you! Your message has been sent.
Unable to send your message. Please fix errors then try again.
<input type="hidden" value="" name="recaptchaResponse">
<input type="hidden" name="formServices" value="704b51ebb0fe2984ab95d4b5c62d9b35">
PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING
Flask
templates
Nicepage Templates
Site1
Page-1.html
31°C Mostly cloudy
ENG
20:59
19-04-2023



OUTPUT.HTML:





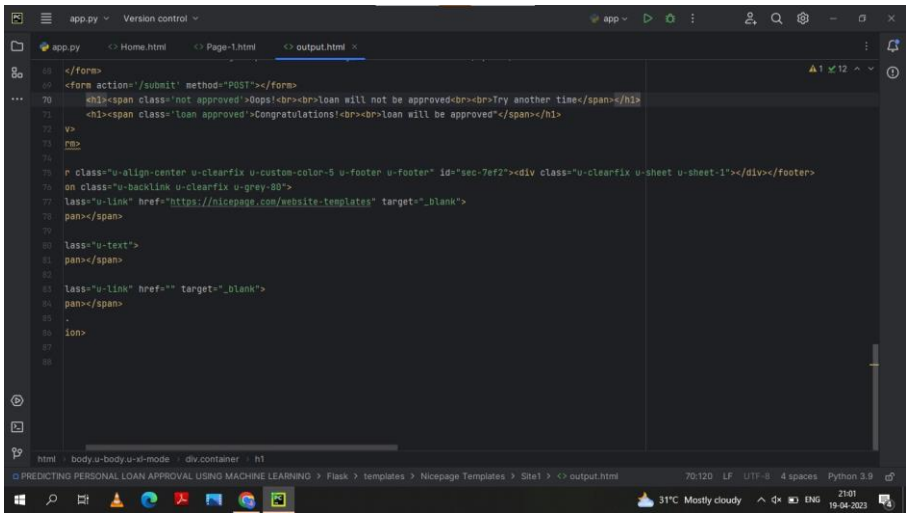

app.py
Version control
app
app.py
Home.html
Page-1.html
output.html
lass="u-nav-item">Contact

nav>
1 class="u-text u-title u-text-1">Get Loan.
h1>
v></header>
on class="u-clearfix u-section-1" id="sec-8217">
class="u-clearfix u-sheet u-sheet-1">
ion>
class="container">
<form action='/submit' method="POST">
<h2 class='container-heading'>Loan Prediction</h2>
</form>
<form action='/submit' method="POST"></form>
<h1>Oops!

loan will not be approved

Try another time</h1>
<h1>Congratulations!

loan will be approved"</h1>
v>
rm>
r class="u-align-center u-clearfix u-custom-color-5 u-footer u-footer" id="sec-7ef2"> </footer>
on class="u-backlink u-clearfix u-grey-80">
html
body.u-body.u-xl-mode
div.container
h1
PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING
Flask
templates
Nicepage Templates
Site1
output.html
70:120
LF
UTF-8
4 spaces
Python 3.9
31°C Mostly cloudy
ENG
21:01
19-04-2023


app.py
Version control
app
app.py
Home.html
Page-1.html
output.html
</form>
<form action='/submit' method="POST"></form>
<h1>Oops!

loan will not be approved

Try another time</h1>
<h1>Congratulations!

loan will be approved"</h1>
v>
rm>
r class="u-align-center u-clearfix u-custom-color-5 u-footer u-footer" id="sec-7ef2"> </footer>
on class="u-backlink u-clearfix u-grey-80">
lass="u-link" href="https://nicepage.com/website-templates" target="_blank">
pan>
lass="u-text">
pan>
lass="u-link" href="" target="_blank">
pan>
.
ion>
html
body.u-body.u-xl-mode
div.container
h1
PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING
Flask
templates
Nicepage Templates
Site1
output.html
70:120
LF
UTF-8
4 spaces
Python 3.9
31°C Mostly cloudy
ENG
21:01
19-04-2023