Project report

INTELLIGENT ADMISSIONS: THE FUTURE OF UNIVERSITY DECISION MAKING WITH MACHINE LEARNING

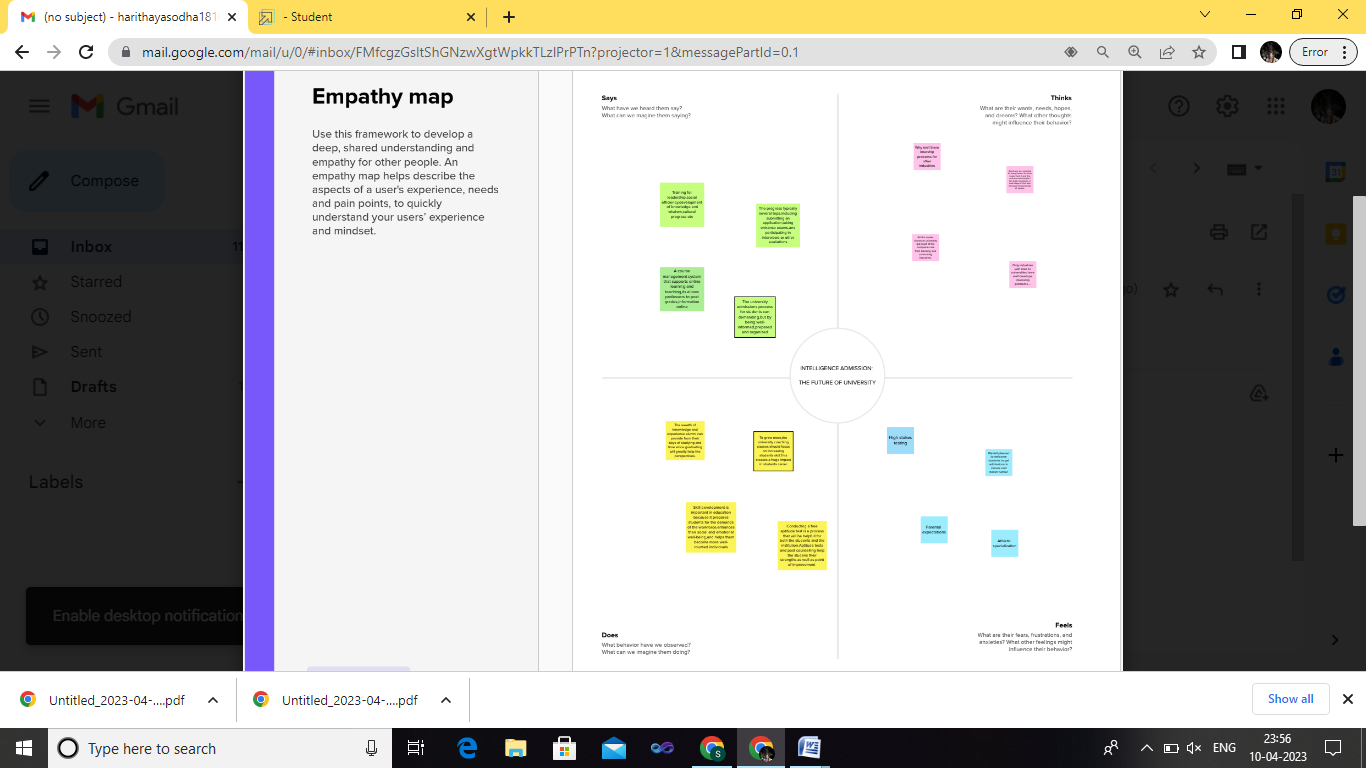
PROJECT DESCRIPTION :

* University admission is the process by which students are selected to attend a college or university. The process typically involves several steps, including submitting an application, taking entrance exams, and participating in interviews or other evaluations.
* Students are often worried about their chances of admission in University. The university admission process for students can be demanding, but by being well-informed, prepared, and organized, students can increase their chances of being admitted to the university of their choice.
* The project aim to predict the chances of a student getting adimitted to a particular university based on certain factors the business value of this project is that it will helps students make more informed decision about which universities to apply to , and help university counsellors to better advise students on the universities they are most likely to be admitted to the university .
* The ability to accurately predict the chances of university admission can help students make more informed decision about which university to apply to , increasing the chances of begin admitted and ultimately gaining access to higher education .

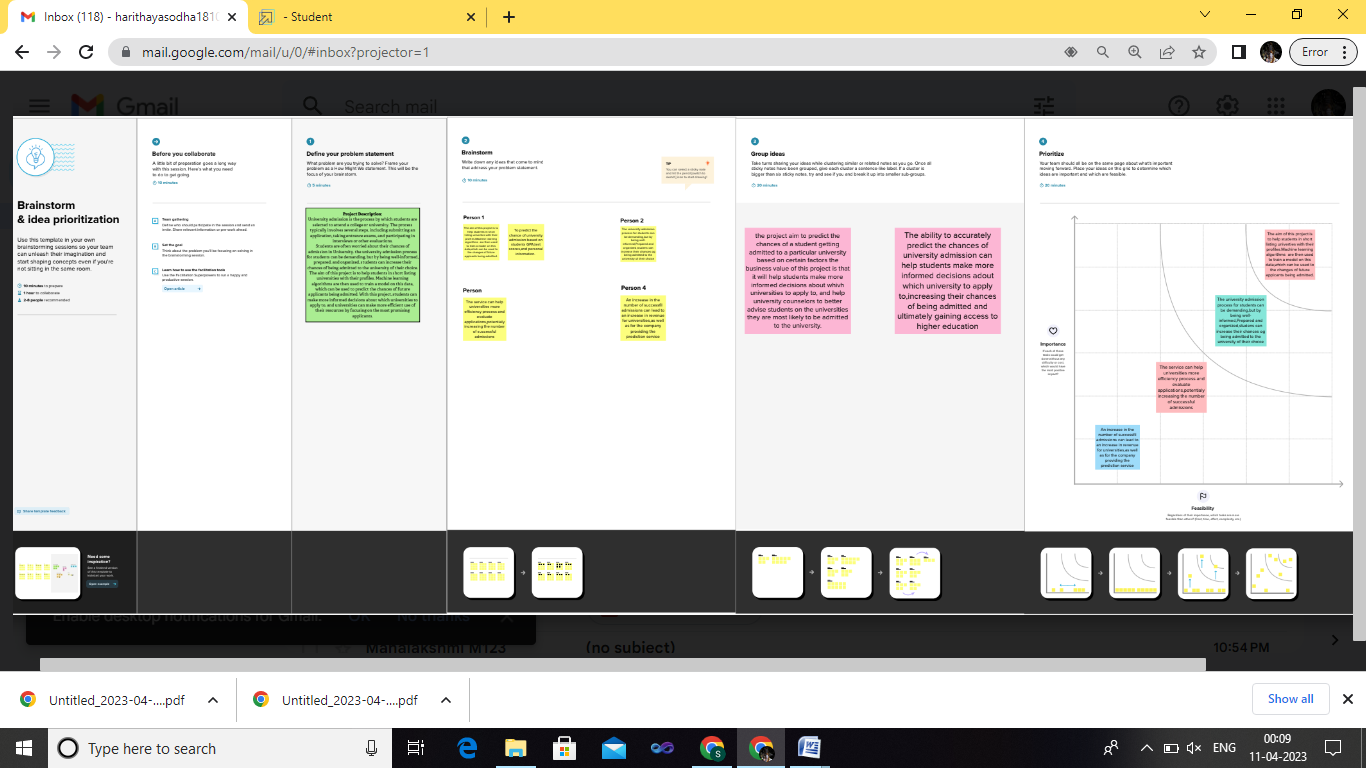
THE PURPOSE OF THIS PROJECT:

* The aim of this project is to help students in short listing universities with their profiles. Machine learning algorithms are then used to train a model on this data, which can be used to predict the chances of future applicants being admitted. With this project, students can make more informed decisions about which universities to apply to, and universities can make more efficient use of their resources by focusing on the most promising applicants .
* The predicted output gives them a fair idea about their admission chances in a particular university. This analysis should also help students who are currently preparing or will be preparing to get a better idea.

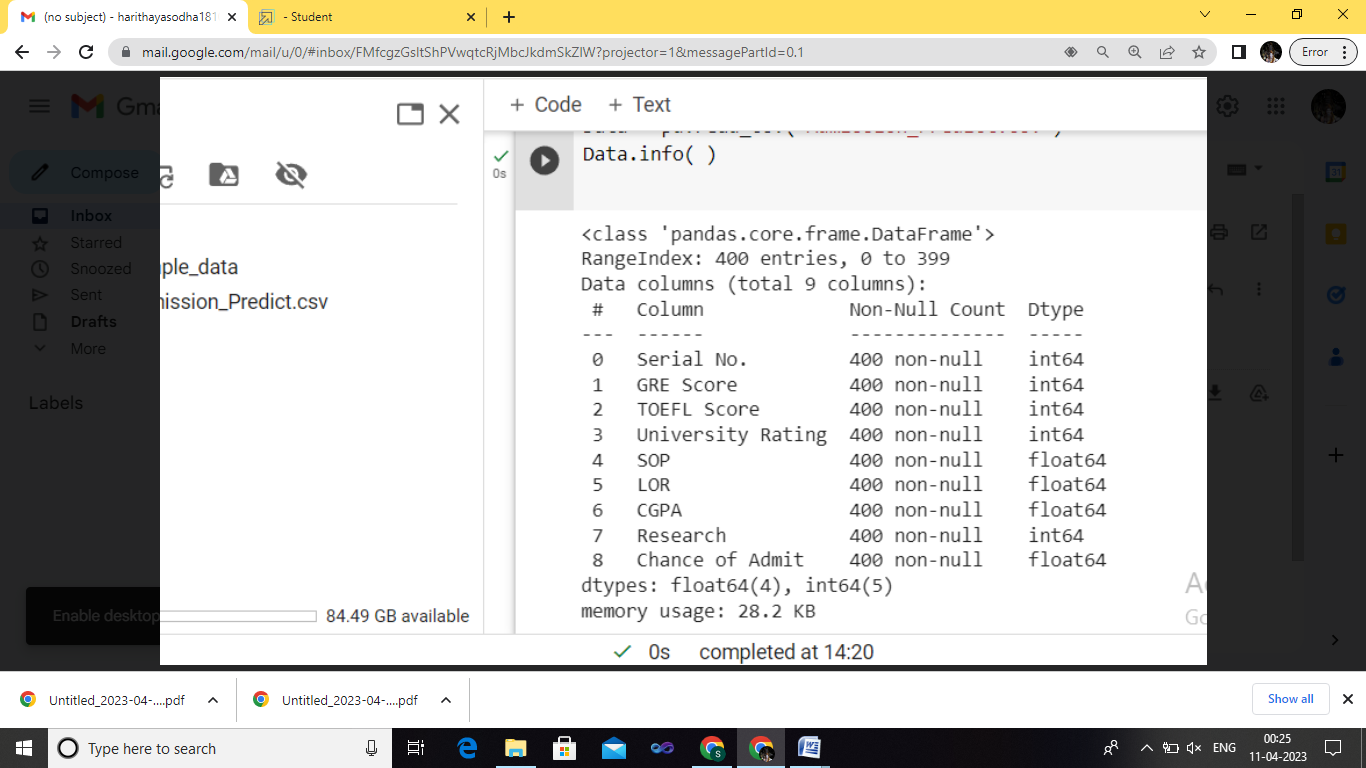
Problem definition and design thinking :

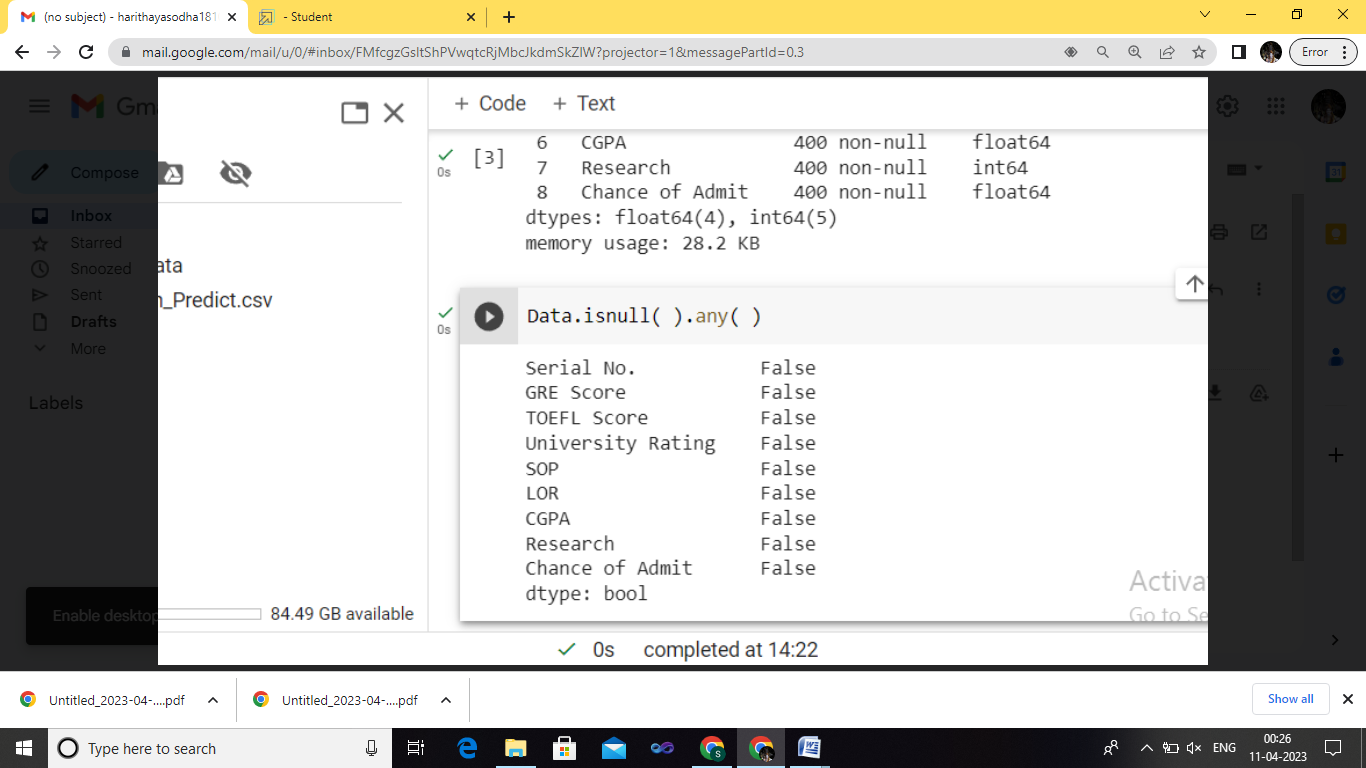


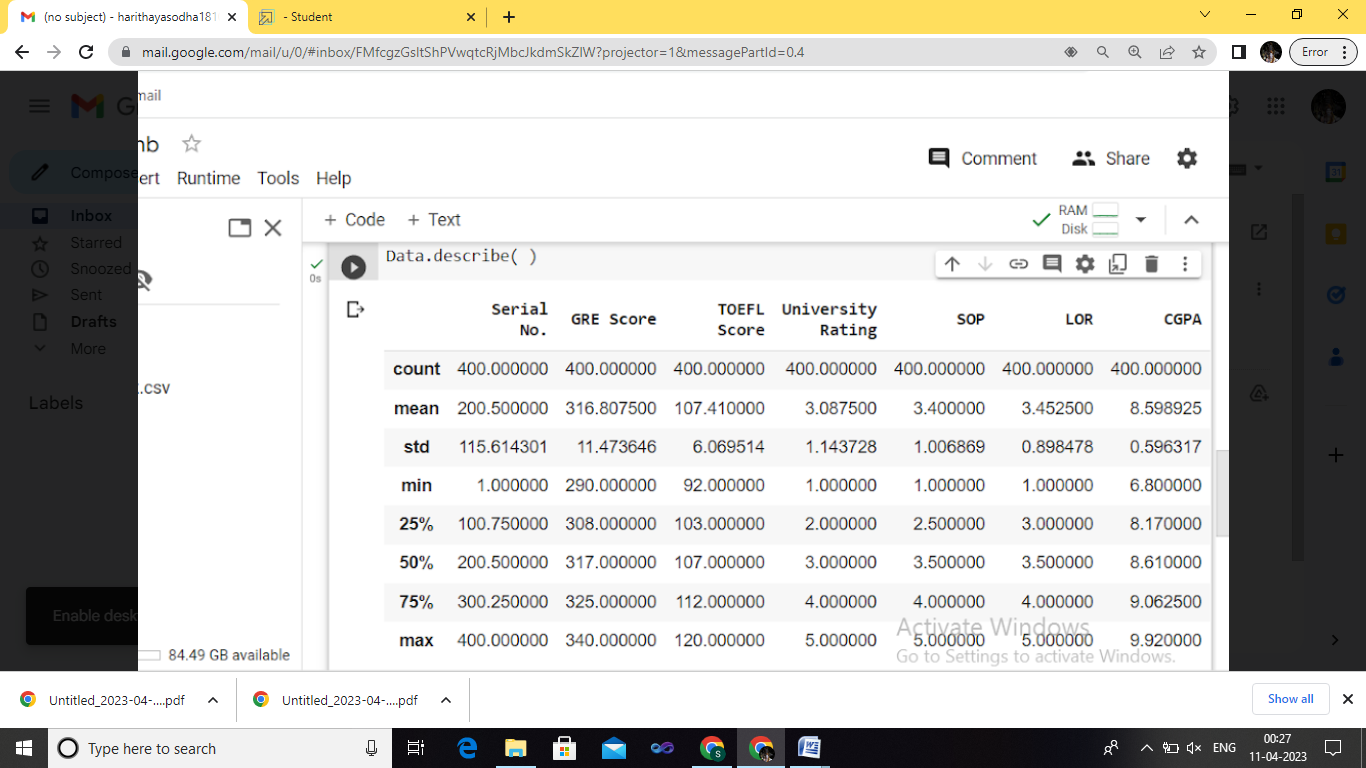
Brainstroming map:

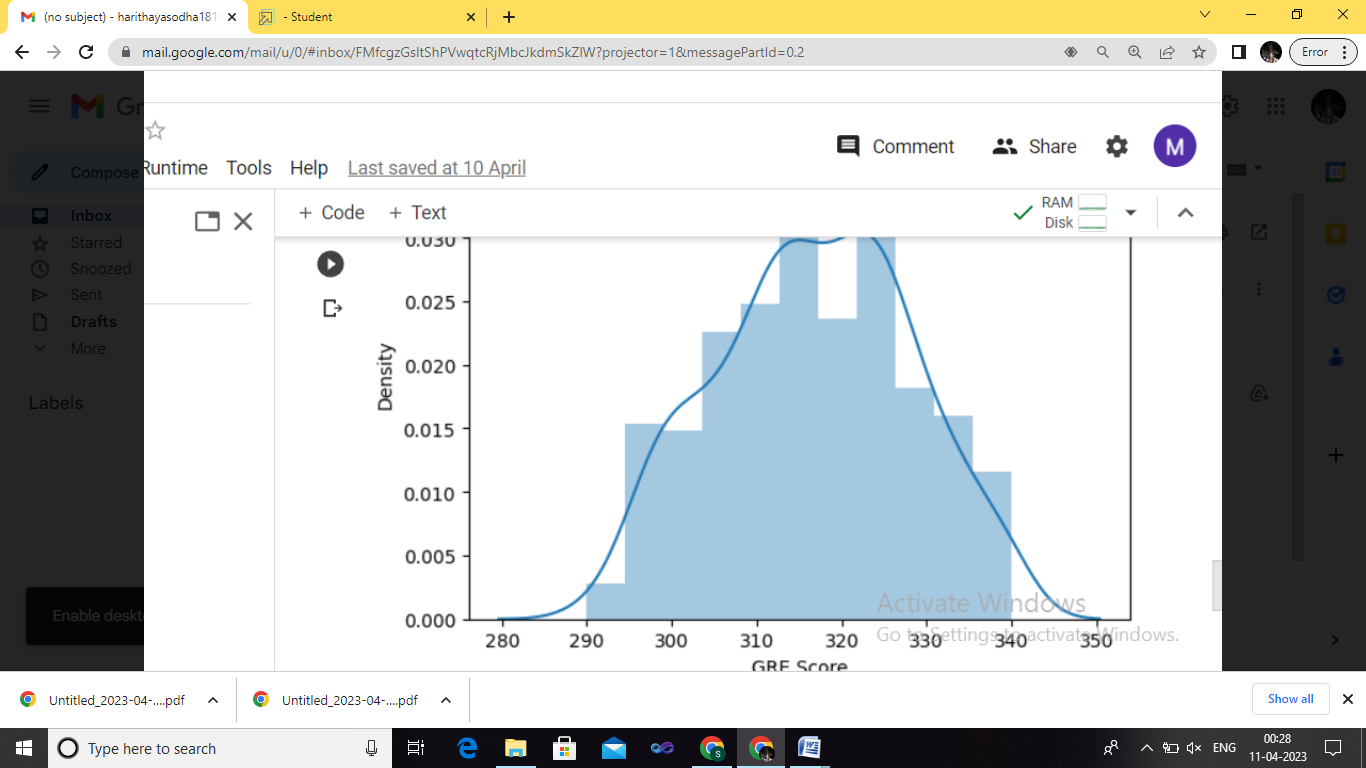


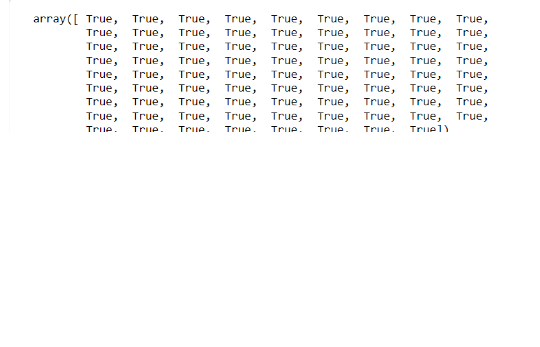
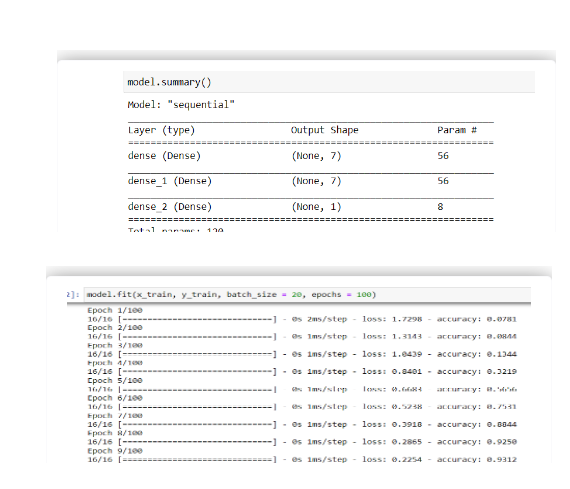
Result:











Advantages of intelligent admissions:

* It helps student for making decision for choosing a right college.
* Here the chance of occurrence of error is less when compared with the existing system.
* It is fast, efficient and reliable.
* Avoids data redundancy and inconsistency.
* Very user-friendly.
* Easy accessibility of data.
* Students and Placement Officer post are first sent to admin for approval. It keeps the system stable by not posting any spam or miscellaneous content on the web.

Disadvatages of intelligent admissions :

* Required active internet connection.
* System will provide inaccurate results if data entered incorrectly.
* May produce inaccurate results if the data is not feed properly.
* Requires an active internet connection

Applications:

* Schools and colleges
* Medical industry
* Business
* campus

Conclusion:

Future work will focus on implementing the proposed architecture for the education system.

Future scope:

High level information technology Skills will continue to be in demand as technoloy advantages and become more intergrated into every area of daily life.

Appendix:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

Data = pd.read\_csv('Admission\_Predict.csv')

Data.info( )

Data.isnull( ).any( )

Data=data.rename(columns = {'Change of Admit':'Change of Admit'})

Data.describe( )

Sns.distplot(data['GRE Score'])

Sns.pairplot(data=data,hue='Research',markers=["^"," v"],palette= 'inferno')

Sns.scatterplot(x='University Rating',y='CGPA',data=data,color='Red',s=100)

Category = ['GRE Score','TOEFL Score','University Rating','SOP','LOR','CGPA','Research','Change of Admit']

Color = ['yellowgreen','gold','lightskyblue','pink','red','purple','orange','gray']

start = True

for i in np.arange(4):

fig = plt.figure(figsize=(14,8))

plt.subplot2grid((4,2),(i,0))

data[category[2\*i]].hist(color=color[2\*i],bins=10)

plt.title(category[2\*i])

plt.subplot2grid((4,2),(i,1))

data[category[2\*i+1]].hist(color=color[2\*i+1],bins=10)

plt.title(category[2\*i+1])

plt.subplots\_adjust(hspace = 0.7,wspace = 0.2)

plt.show( )

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler( )

x=sc.fit\_transform(x)

x

x=data.iloc[:,0:7].values

x

y=data.iloc[:,7:].values

y

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y test\_size=0.30,random\_state=101)

y\_train=(y\_train>0.5)

y\_train

y\_test=(y\_test>0.5)

from sklearn.liner\_model.logistic import LogisticRegression

cls =LogisticRegression(random\_state =0)

lr=cls.fit(x\_train, y\_train)

y\_pred =lr.predict(x\_test)

y\_pred

import tensorflow as tf

from tensorflow import keras

from tensorflow .kears.layers import Dense,Activation,Dropout

from tensorflow .kears.optimizers import Adam

model=keras.Sequential( )

model.add(Dense(7,activation ='relu',input\_dim=7))

model.add(Dense(7,activation='relu'))

model.add(Dense(1,activation='linear'))

model.summary( )

model.fit(x\_train, y\_train,batch\_size = 20, epochs = 100)

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

model.fit(x\_train, y\_train, batch\_size = 20, epochs = 100)

from sklearn.metrics import accuracy\_score

train\_predictions = model.predict(x\_train)

print(train\_predictions)

train\_acc = model.evaluate(x\_train, y\_train, verbose=0)[1]

print(train\_acc)

test\_acc = model.evaluate(x\_test, y\_test, verbose=0)[1]

print(test\_acc)

pred=model.predict(x\_test)

pred = (pred>0.5)

pred

from sklearn.metrics import accuracy\_score,recall\_score,roc\_auc\_score,confusion\_matrix

print("\nAccuracy score: %f" %(accuracy\_score(y\_test,y\_pred) \* 100))

print("Recall score : %f" %(recall\_score(y\_test,y\_pred) \*100))

print("ROC score : %f\n" %(roc\_auc\_score(y\_test,y\_pred) \* 100))

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

from sklearn.metrics import accuracy\_score,recall\_score,roc\_auc\_score,confusion\_matrix

print(classification\_report(y\_train,pred))

from sklearn.metrics import accuracy\_score,recall\_score,roc\_auc\_score,confusion\_matrix

print(classification\_report(y\_test,pred))

model.save('model.h5')

import numpy as np

from flash import flask,request,jsonify,rendre\_rempalate

import pickle

app = Flask(\_name\_)

from tensorflow.kears.models import load\_model

model = load\_model('model.h5')

@app.route('/')

def home( ):

return render\_template('Demo2.html')

def y\_predict( ):

min1=[290.0,92.0,1.0,1.0,1.0,6.8,0.0]

max1=[340.0,120.0,5.0,5.0,5.0,9.92,1.0]

k= [float(x) for x in request.form.values( )]

p=[ ]

for i in range(7):

l=(k[i]-min1[i])/(max1[i]-min1[i])

p.append(1)

prediction = model.predict([p])

print(prediction)

output=prediction[0]

if(output==False):

return render\_template('noChange.html',prediction\_text='You Dont have a chance of getting admission'

else:

return render\_template('change.html',prediction\_text='You have a chance of grtting admission'

if\_name\_=="\_main\_":

app.run(debug=False)