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 admissions in university. The university admissions 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 admitted to a particular university based on certain factors the business values of this project is that it will helped 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 admissions 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

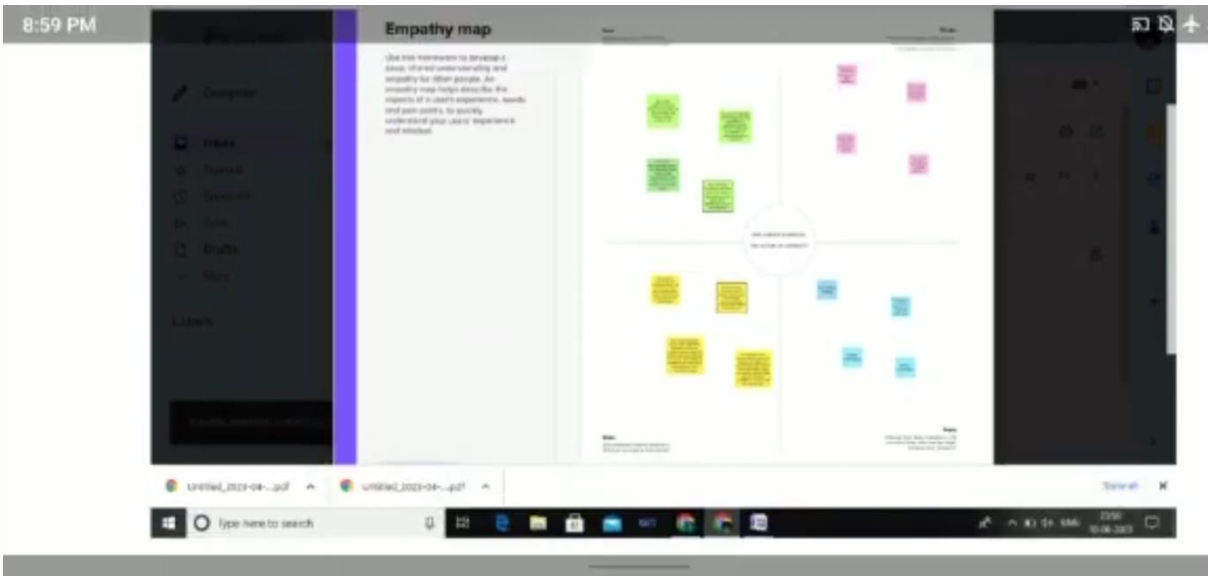
THE PURPOSE OF THE PROJECT:

- ✓ 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 modal 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 applicant.

- ✓ **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**

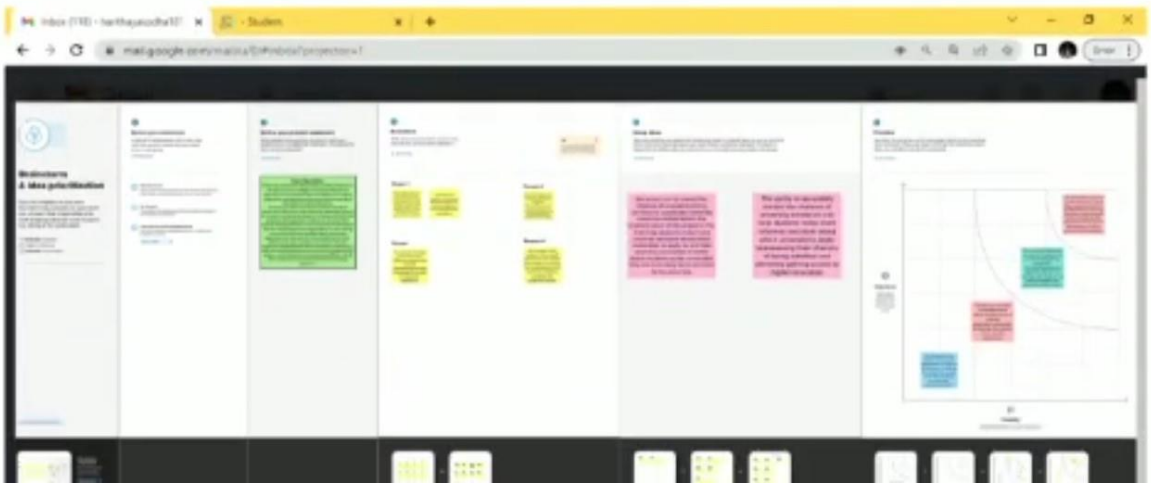
PROBLEM DEFINITION AND DESIGN THINKING:

EMPATHY MAP

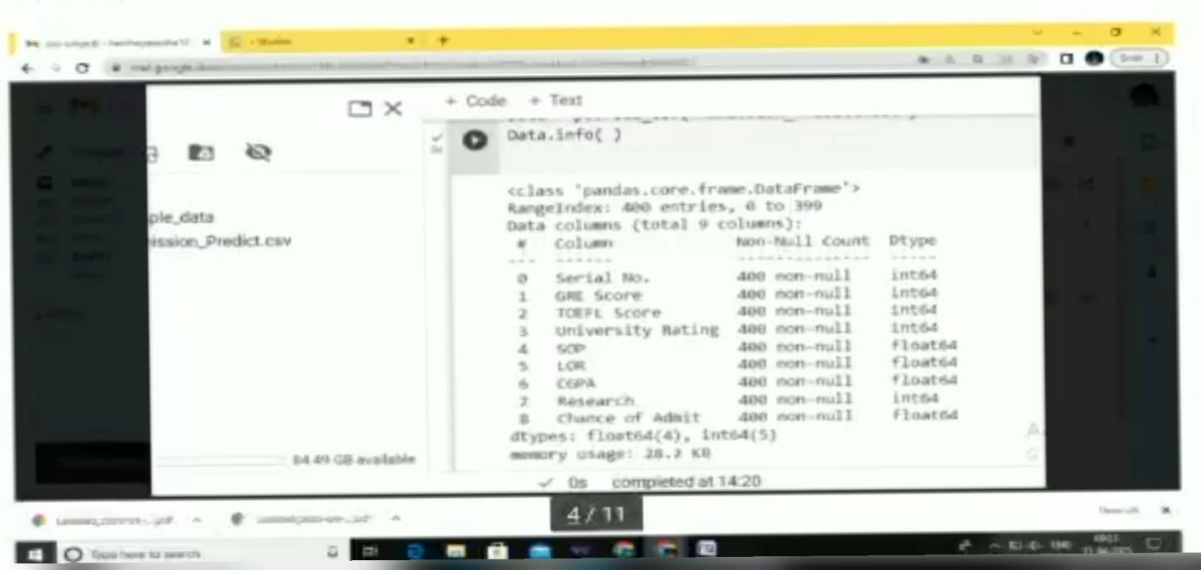


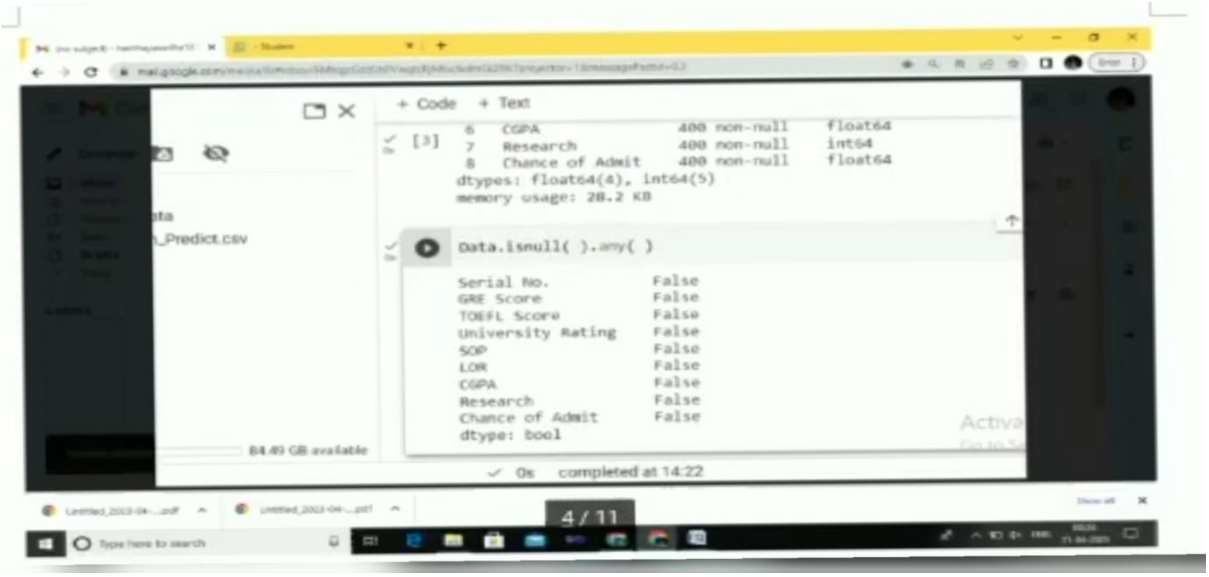
BRAINSTORMING MAP

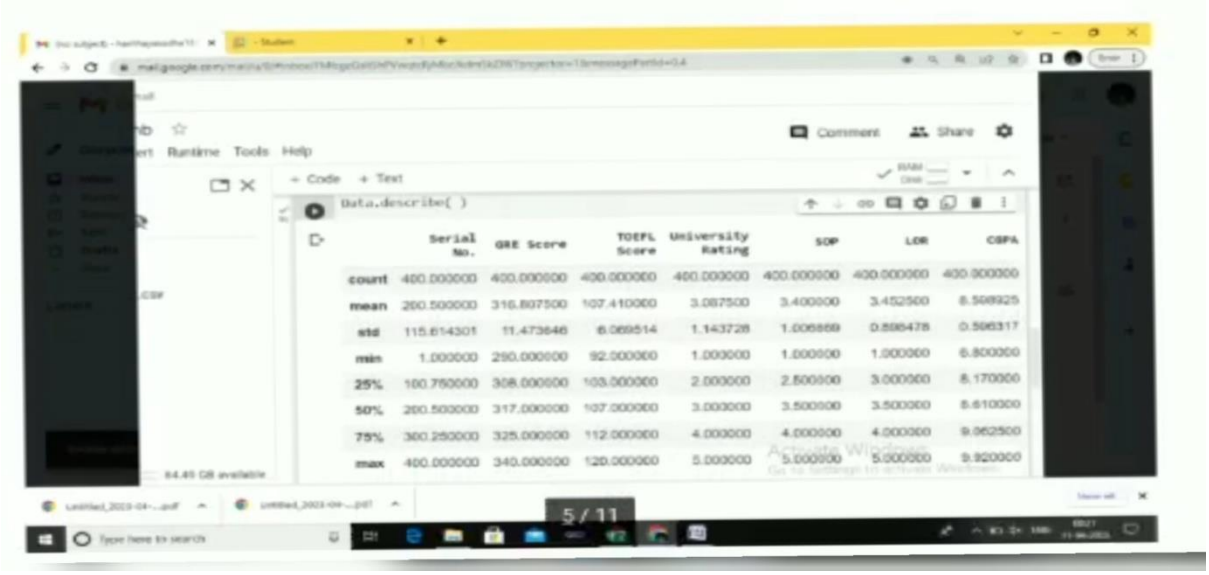
Brainstroming map:

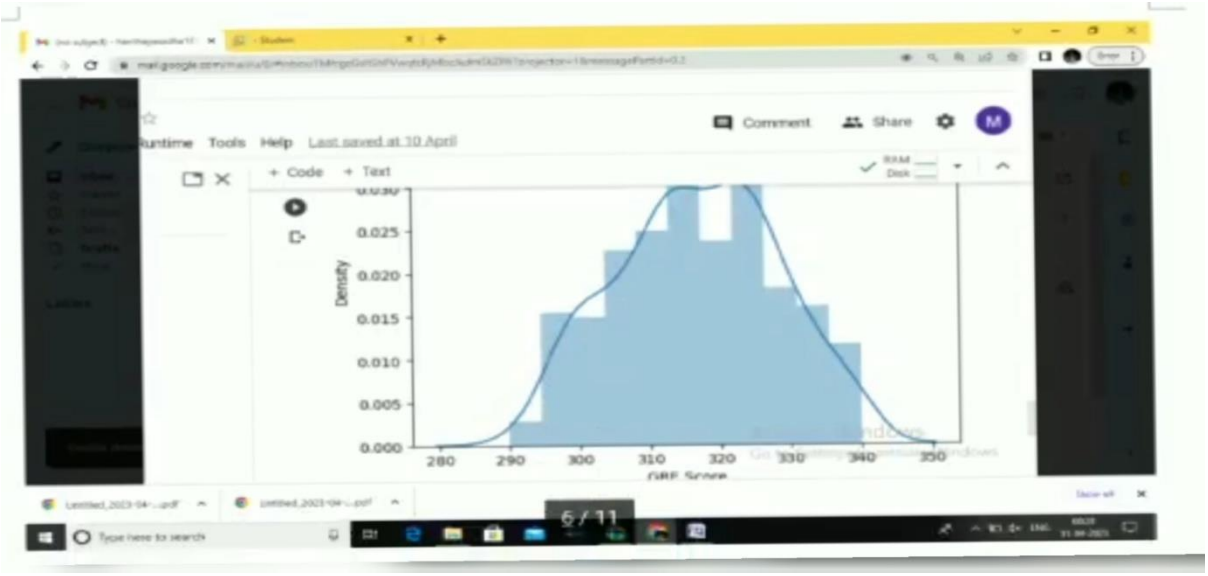


RESULT:




+ Code + Text
[3]
6 CGPA 400 non-null float64
7 Research 400 non-null int64
8 Chance of Admit 400 non-null float64
dtypes: float64(4), int64(5)
memory usage: 28.2 KB
Data.isnull( ).any( )
Serial No. False
GRE Score False
TOEFL Score False
University Rating False
SOP False
LOR False
CGPA False
Research False
Chance of Admit False
dtype: bool
84.49 GB available
0s completed at 14:22
4 / 11


Runtime Tools Help
Comment
Share
+ Code + Text
Data.describe( )
Serial No. GRE Score TOEFL Score University Rating SOP LOR CGPA
count 400.000000 400.000000 400.000000 400.000000 400.000000 400.000000 400.000000
mean 200.500000 316.807500 107.410000 3.087500 3.400000 3.452500 8.598925
std 115.614301 11.473646 6.069514 1.143728 1.006869 0.898478 0.596317
min 1.000000 290.000000 92.000000 1.000000 1.000000 1.000000 6.800000
25% 100.750000 308.000000 103.000000 2.000000 2.500000 3.000000 8.170000
50% 200.500000 317.000000 107.000000 3.000000 3.500000 3.500000 8.610000
75% 300.250000 325.000000 112.000000 4.000000 4.000000 4.000000 9.062500
max 400.000000 340.000000 120.000000 5.000000 5.000000 5.000000 9.920000
84.49 GB available
5 / 11




```
model.summary()
```

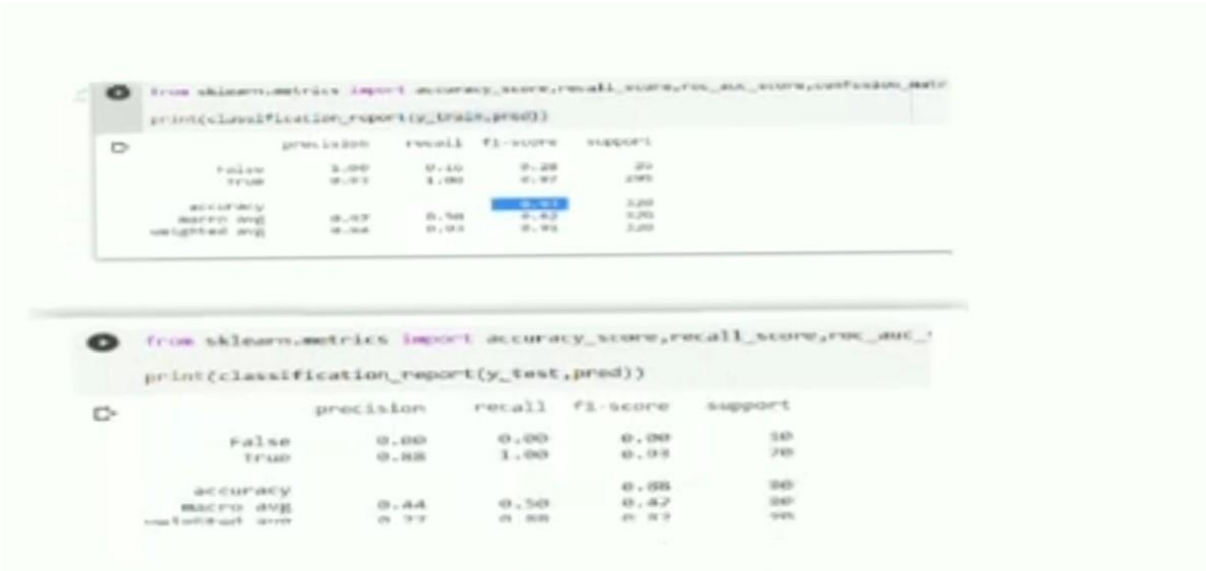
```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 7)	56
dense_1 (Dense)	(None, 7)	56
dense_2 (Dense)	(None, 1)	8

```
model.fit(x_train, y_train, batch_size = 32, epochs = 100)
```

```
Epoch 1/100
25/25 [==============================] - 0s 1ms/step - loss: 1.7298 - accuracy: 0.4701
Epoch 2/100
25/25 [==============================] - 0s 1ms/step - loss: 1.5183 - accuracy: 0.5854
Epoch 3/100
25/25 [==============================] - 0s 1ms/step - loss: 1.0439 - accuracy: 0.6346
Epoch 4/100
25/25 [==============================] - 0s 1ms/step - loss: 0.8484 - accuracy: 0.7349
Epoch 5/100
25/25 [==============================] - 0s 1ms/step - loss: 0.6884 - accuracy: 0.7981
Epoch 6/100
25/25 [==============================] - 0s 1ms/step - loss: 0.5778 - accuracy: 0.7949
Epoch 7/100
25/25 [==============================] - 0s 1ms/step - loss: 0.5458 - accuracy: 0.8045
Epoch 8/100
25/25 [==============================] - 0s 1ms/step - loss: 0.5069 - accuracy: 0.8269
Epoch 9/100
25/25 [==============================] - 0s 1ms/step - loss: 0.4754 - accuracy: 0.8333
```

```
True, True, True, True, True, True, True, True, True,
True, True, True, True, True, True, True, True, True,
True, True, True, True, True, True, True, True, True,
True, True, True, True, True, True, True, True,
```



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.
- Student 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.

DISADVANTAGES OF INTELLIGENT ADMISSIONS:

- Required active internet connection.
- System will provide inaccurate results if data entered in correctly.
- 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 technology advantages and become more intergrated into every area of daily life .

APPENDIX:

```
Import numpy as np

Import pandas as pd

Import matplotilb.pyplot as plt

Impor seaborn as sns

%matpltlib inline

Data=pd.read_csv('Admission_predict.csv)

Data.info()

Data.isnull().any()

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

Data.describe()

Sns.displot(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 Scoree','University
Rating','SOP','LOR','GGPA','Research','change of admit']

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

Start=True

For i in np.arange(4):

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

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

    Data(category[2*i].his(color=color[2*i]P.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 – o.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_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=Logistic Regression(random_state=0)

Ir=cls fit(x_train,y_train)

Y_pred=ir predict (x_test)

Y_pred

Import tensorflow as tf

From tensorflow import keras

From tensorflow.kears.layers import Dence,Activation,Dropout

From tensorflow.kears.optimizers import  Adam

Model=kears.sequential()

Model.add(Dence(7,activation='relu',input_dim=7))

Model.add(Dence(7,activation='relu'))

Model.add(Dence(7,activation='relu'))

Model.add(Dence(1,activation='linear'))

Model.summery()

Model.fit(x_train,y_train,batch_size=20,epochs=100)

Model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['aqiurac
y'])
```

```
Model.fit(x_train,y_train,batch_size=20,epochs=100)

From sklearm.matrice import accuracy_score

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)
```