

**KAMARAJAR GOVERNMENT ARTS AND
SCIENCE COLLEGE
SURANDAI
TENKASI – 627859**

MACHINE LEARNING WITH PYTHON

April-2023

The Future of University Decision Making

Using with Machine Learning

Team Size: 5

Team Leader : **Menaka.V**

REG No : **(20201061506121)**

Team Member :**Divya. M**

REG No : **(20201061506107)**

Team Member : **Malathi.K**

REG No : **(20201061506115)**

Team Member : **Muthu Lakshmi.P**

REG No : **(20201061506128)**

Team Member : **Selva Jothi.G**

REG No : **(20201061506135)**

1.INTRODUCTION:

The Future of University decision making with machine learning is an exciting and rapidly developing field. With the help of machine learning algorithms, Universities can make data-driven decisions that can improve everything from student retention to campus safety.

One area where machine learning can be particularly useful in predicting student outcomes. By analyzing large datasets of student information, including demographic data, academic performance, and extracurricular activities, machine learning algorithms can identify patterns and make predictions about which students are at risk of dropping out, which students are likely to

excel, and which interventions will be most effective for different groups of students.

Machine learning can also be used to improve campus safety. By analyzing data on crime patterns and campus security measures, machine learning algorithms can identify areas where security can be improved and predict where incidents are likely to occur.

Overall, The Future of University decision making with machine learning is bright. As universities continue to gather more data and refine their algorithms, they will be able to make more accurate predictions and more informed decisions that benefit students, faculty, and staff alike.

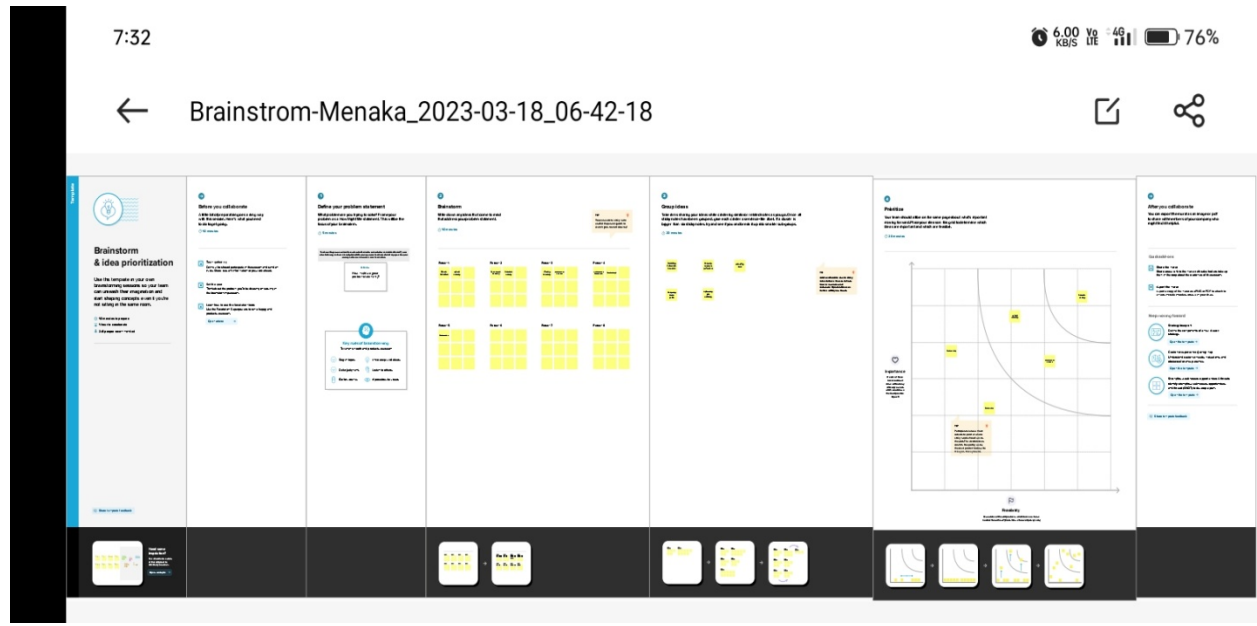
OVERVIEW:

University decision-making with machine learning involves the use of algorithms and statistical models to analyze data and provide insights that can inform decisions made by universities. Machine learning can be used to identify patterns in large datasets, predict outcomes, and automate tasks, which can lead to more informed and efficient decisions making.

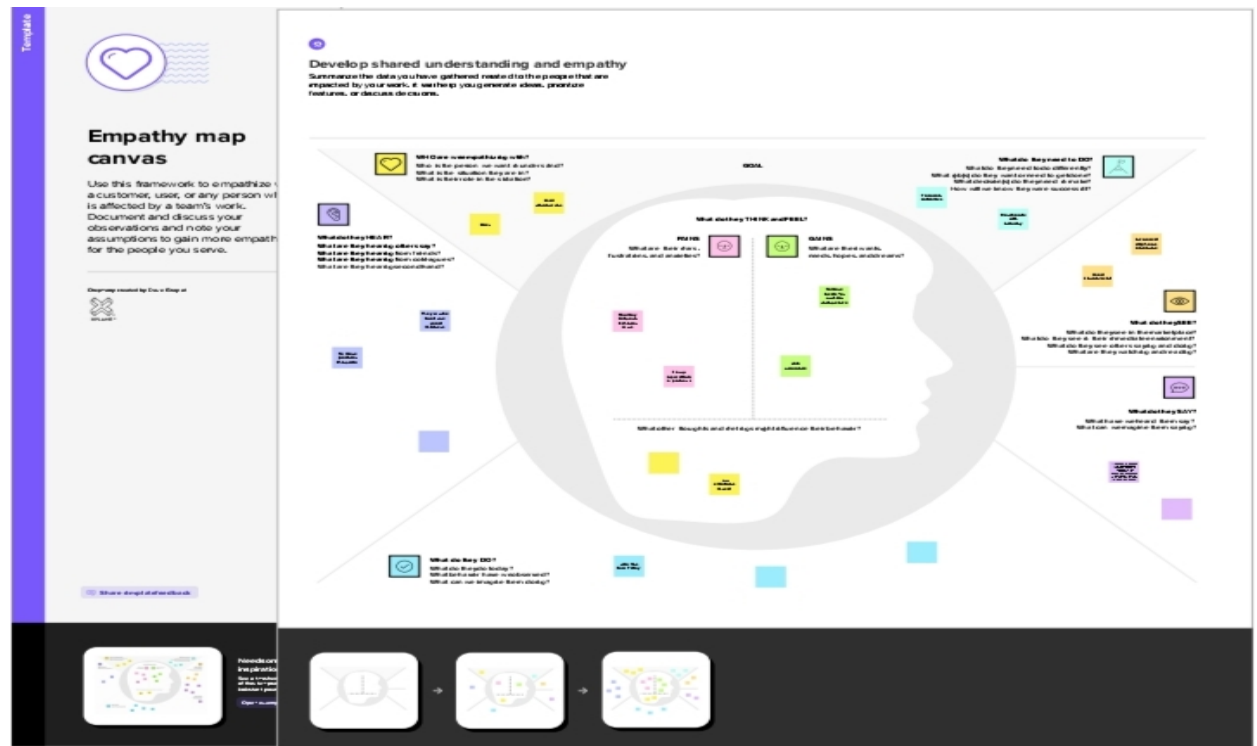
One example of how machine learning can be applied in universities is through student admissions. Machine learning algorithms can be used to analyze past admission data, including factors such as grades, test scores, and extracurricular activities, to identify patterns and predict

2.PROBLEM DEFINITION:

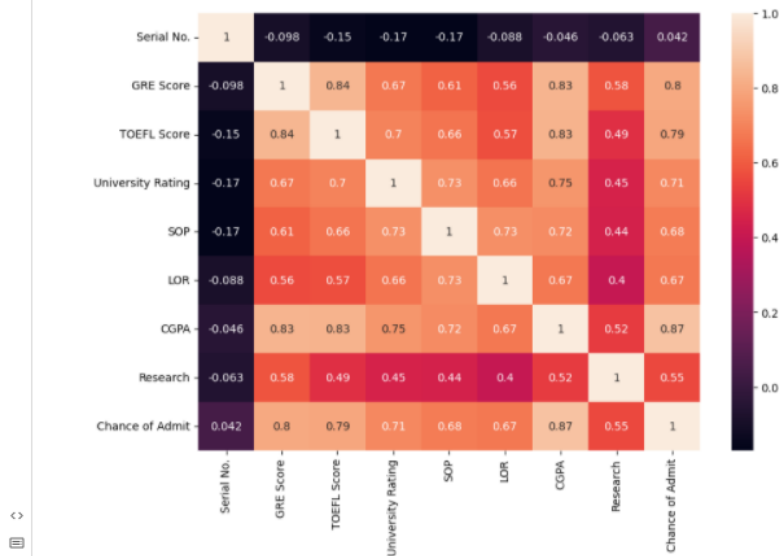
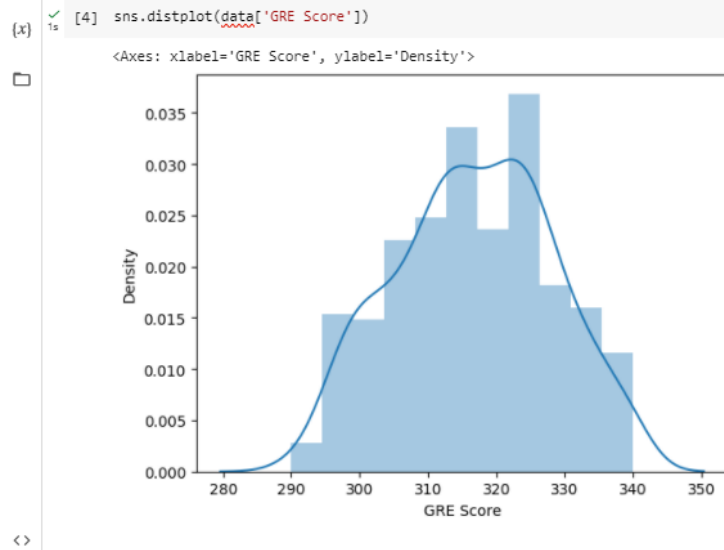
Brainstrom Map:

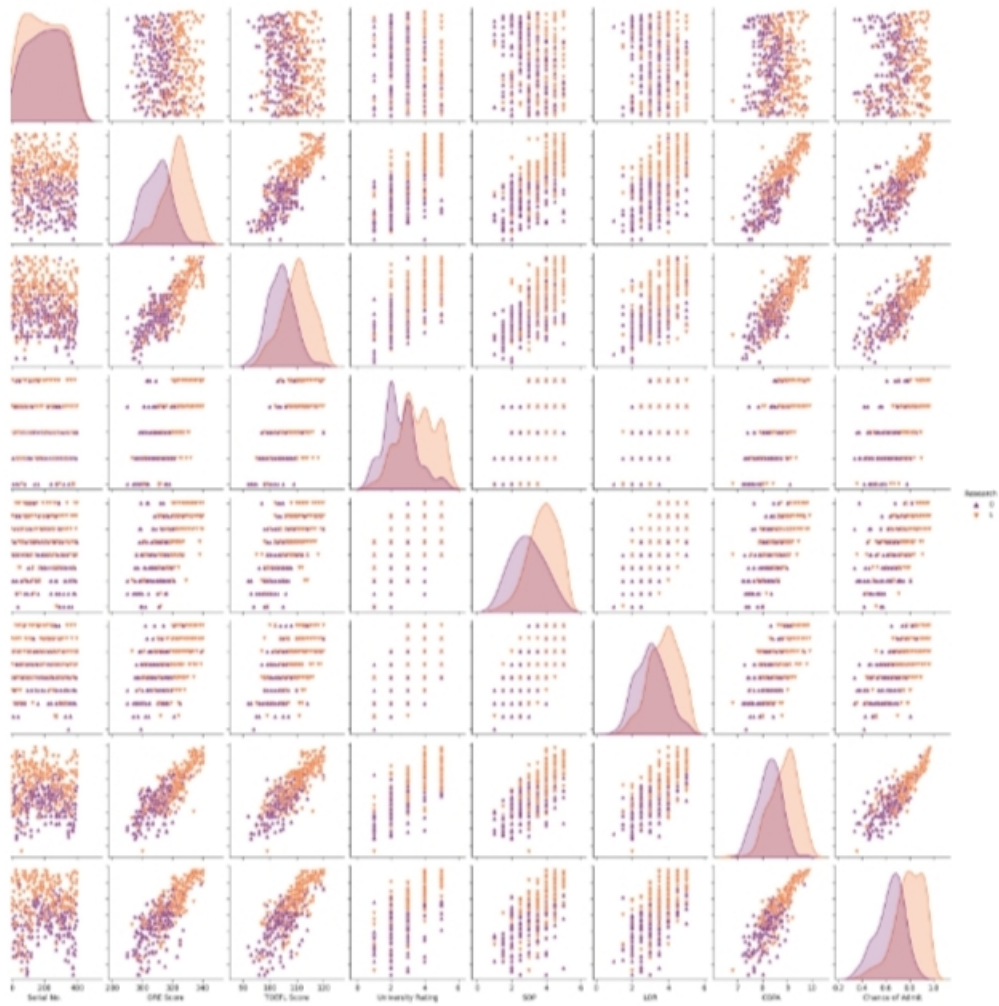


Empathy Map:



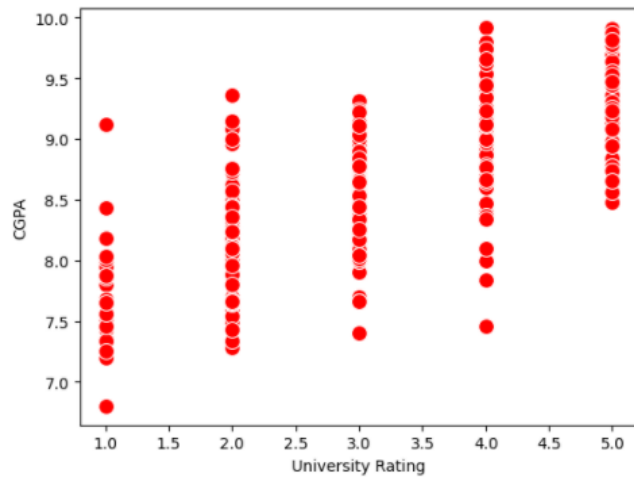
3.RESULTS:

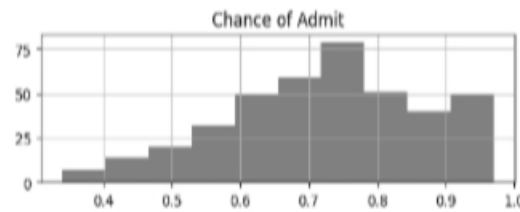
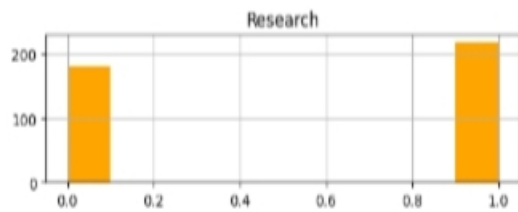
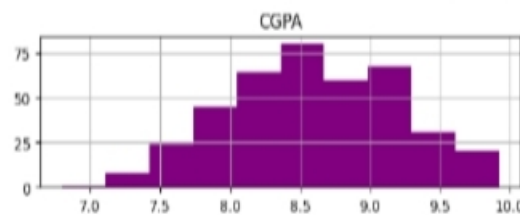
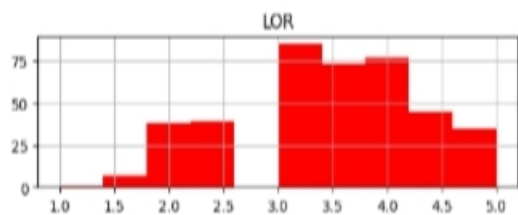
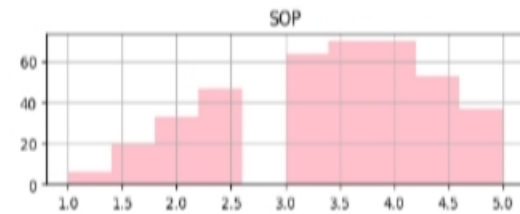
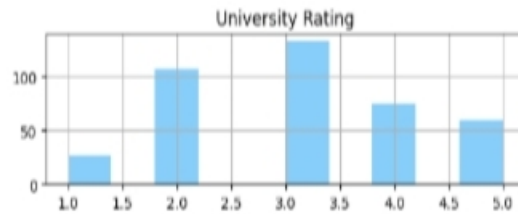
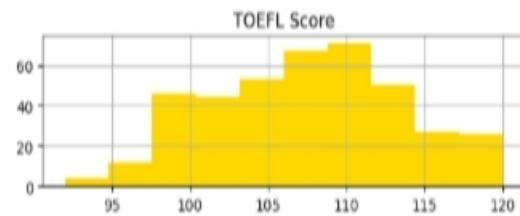




```
[4] ✓ 2s sns.scatterplot(x='University Rating',y='CGPA',data=data,color='r',s=100)
```

<Axes: xlabel='University Rating', ylabel='CGPA'>





4.ADVANTAGES & DISADVANTAGES:

Advantages:

1.Better data analysis:

Machine learning can analyze vast amounts of data and provide insights that may be difficult or impossible for human to

identify. This can lead to more informed decision making.

2.Increased efficiency:

Machine learning algorithms can process large amounts of data quickly, leading to faster decision making and reduced administrative overhead.

3.Improved student outcomes:

By analyzing data on student performance and engagement, machine learning can help universities identify at-risk students and provide targeted interventions to improve their outcomes.

4.Personalized learning: Machine learning can analyze individual student data to provide customized learning experiences,

such as adaptive learning paths and personalized feedback.

Disadvantages:

1.Bias in data:

Machine learning algorithms are only as good as the data they are trained on. If the data is biased, the algorithm will produce biased results. This can perpetuate existing inequalities in the university system.

2.Lack of transparency:

Machine learning algorithms can be complex and difficult to understand. This can make it difficult for stakeholders to understand why certain decisions are being

made, learning to a lack of trust in the system.

3.Limited flexibility:

Machine learning algorithms are designed to learn from data and make decisions based on that data. However, they may not be able to take into account important factors that are not included in the data, such as a student's personal circumstances or a faculty member's teaching style.

4.Cost:

Implementing machine learning systems can be expensive, particularly if universities need to hire data scientists and invest in new infrastructure to support the

technology. This cost may be prohibitive for some institutions, particularly smaller ones.

5.APPLICATIONS:

The primary application of university admissions to allow students to apply for admission to a higher education institution. The admission process typically involves submitting an application, including academic records, standardized test score, essays, and letters of recommendation. The university then evaluates the application and decides whether to offer the student admission.

There are several reasons why university admissions are important:

- Education: University Online admissions allow students to pursue higher education and gain knowledge and skills that can help them in their careers.
- Career opportunities: Many employers require a college degree for certain positions, so university admissions can open up new job opportunities for students.
- Personal development: University admission provides students with opportunities to develop new interests, hobbies, and skills which can enrich their lives and help them become more well-rounded individuals.
- Research: University admissions can also contribute to scientific research

and knowledge generation. Universities are often at the forefront of cutting-edge research in a wide range of fields, and students can contribute to this research through their academic work.

- **Innovation:** University admissions can also lead to new innovations in technology, medicine and other fields. Many groundbreaking innovations have emerged from university research and collaboration between students, faculty and industry partners.
- Overall, university admissions play a critical role in society by providing students with opportunities to gain knowledge, develop new skills, and contribute to research and innovation.

6.CONCLUSION:

In conclusion, machine learning has the potential to transform the way universities make decisions, By analyzing large amounts of data, machine learning algorithms can identify patterns and provide insights that can help university administrators make more informed decisions about everything from admissions about everything from admissions to resource allocation.

However, it is important to recognize that machine learning is not a magic solution that can replace human decision making entirely. Instead, it should be seen as a powerful tool that can support and enhance the decision making process.

7.FUTURE SCOPE:

1. Website and online portal for admission should be prepared well in advance and Ready portal should be demonstrated to the Admission Committee of the college Before final selection of the service provider. After the section of the service provider selected party must present a trial run to the Admission Committee before the commencement of receiving online applications.

2. An applicant should be able to fill up the form directly without any login. The information technology industry is growing at a very fast pace. Every sector is adopting new and superior technologies to carry out tiresome tasks to optimise efficiency. The

high demand for IT professionals in today's world has made courses offered in this field really popular among masses. Bachelor of Science in Information Technology is one such course that equips students with knowledge of storage processing, securing, and maintaining databases and exposure to information processing. But before you sign up for the course, it is essential to know about the different career pathways you can explore. Here is an insightful blog on focusing on the its scope

8.APPENDIX:

A.Source code:

```
import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import pickle

warnings.filterwarnings('ignore')

data=pd.read_csv('/content/drive/MyDrive/Project.csv')
```

```
data.info()
```

```
data.isnull().any()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 400 entries, 0 to 399
```

```
Data columns (total 9 columns):
```

#	Column	Non-Null Count	Dtype
0	Serial No.	400 non-null	int64
1	GRE Score	400 non-null	int64
2	TOEFL Score	400 non-null	int64
3	University Rating	400 non-null	int64
4	SOP	400 non-null	float64
5	LOR	400 non-null	float64
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
```

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

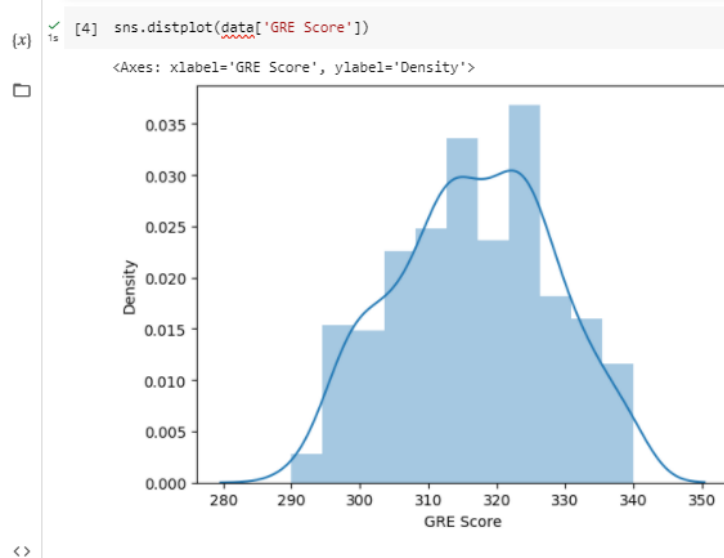
data.head()

data.describe()

The screenshot shows a Jupyter Notebook interface with two code cells. The first cell contains `data.head()` and displays a table with 10 columns: Serial No., GRE Score, TOEFL Score, University Rating, SOP, LOR, CGPA, Research, Chance of Admit, and a final column with values 0.92, 0.76, 0.72, 0.80, and 0.65. The second cell contains `data.describe()` and displays a summary table with 10 columns: Serial No., GRE Score, TOEFL Score, University Rating, SOP, LOR, CGPA, Research, Chance of Admit, and a final column. The summary table includes rows for count, mean, std, min, 25%, 50%, 75%, and max. The bottom status bar indicates the notebook is completed at 9:09 PM.

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	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	0.547500	0.724350
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

```
sns.distplot(data['GRE Score'])
```



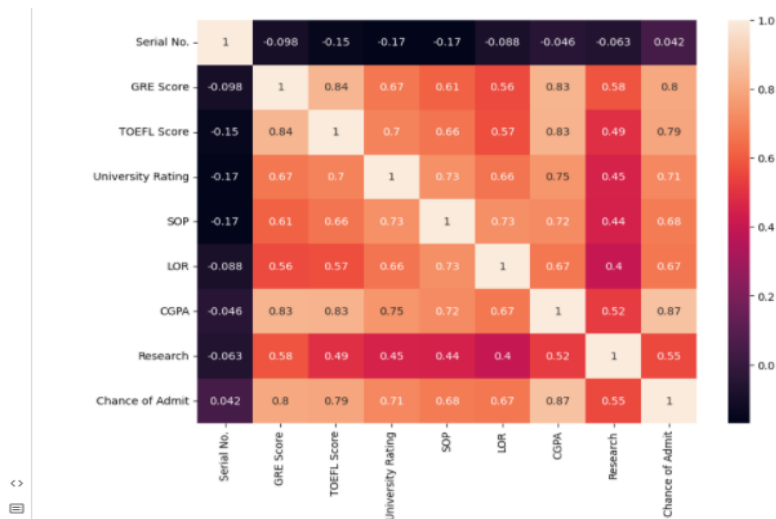
```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

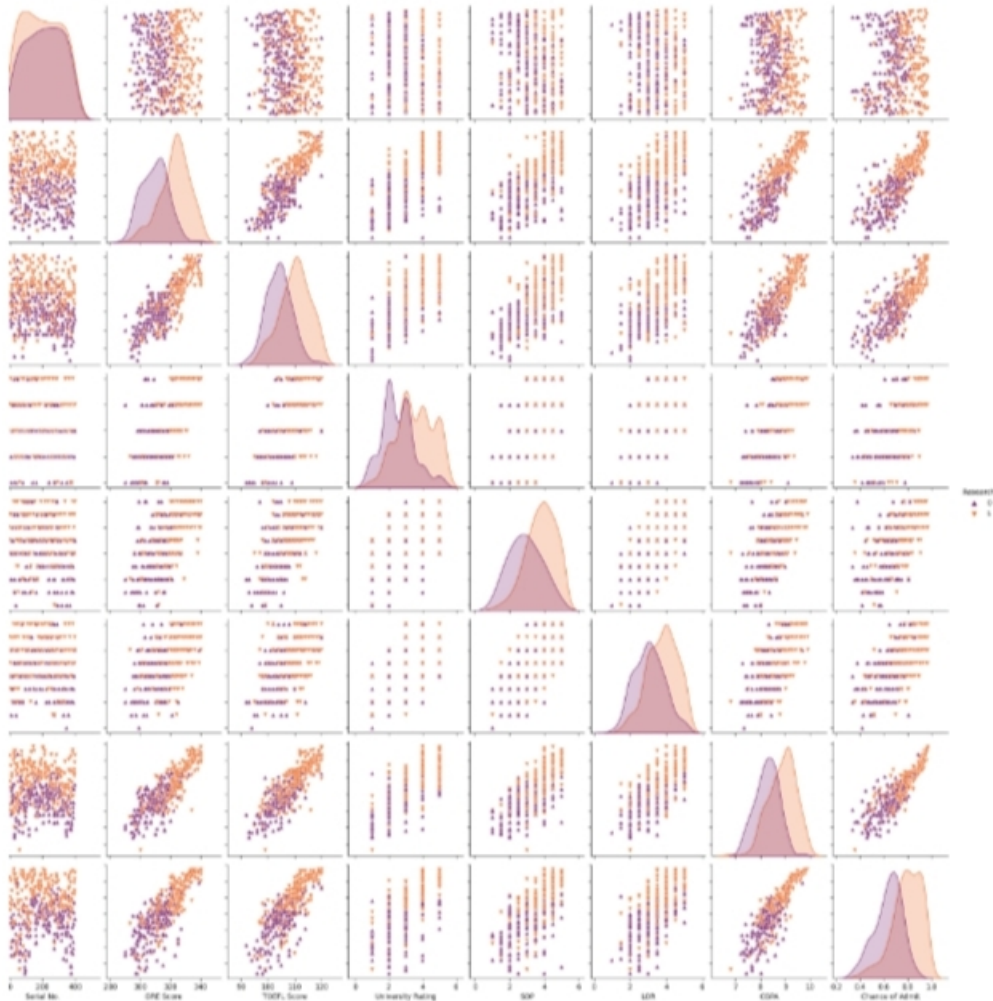
Drive already mounted at /content/drive

```
plt.figure(figsize=(10,7))
```

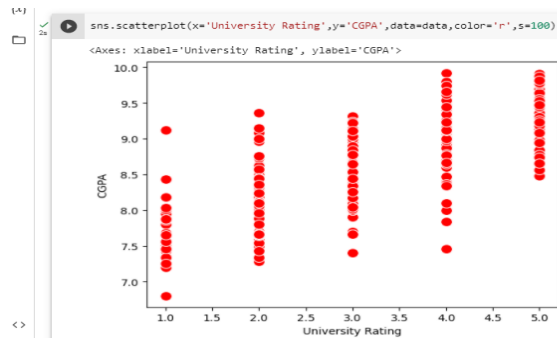
```
sns.heatmap(data.corr(),annot=True)
```



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



```
sns.scatterplot(x='University Rating',y='CGPA',data=data,color='r',s=100)
```



```
category=['GRE Score','TOEFL Score','University Rating','SOP','LOR','CGPA','Research','Chance 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))
```

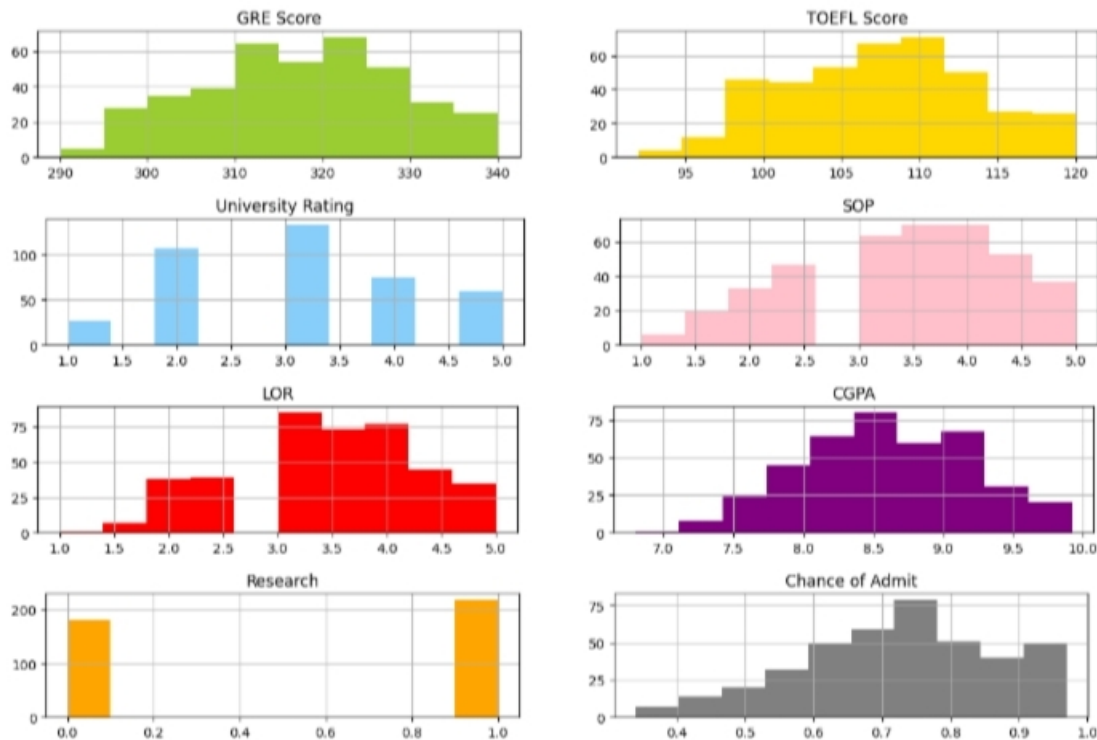
```
    data[category[2*i]].hist(color=color[2*i],bins=10)
```

```
    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])
```



```
from sklearn.preprocessing import MinMaxScaler
```

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

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

```
sc=MinMaxScaler()
```

```
x=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=0.30,  
random_state=101)
```

```
y_train=(y_train>0.5)
```



```
y_train
```

```
y_test=(y_test>0.5)
```

```
y_test
```

```
from sklearn.linear_model import LogisticRegression
```

```
cls=LogisticRegression()
```

```
lr=cls.fit(x_train,y_train)
```

```
y_pred=lr.predict(x_test)
```

```
y_pred
```

```
array([False, False, False, False, False, True, True, False, True,  
True, True, False, True, True, False, True, True, False, False,  
False, False, False, True, False, False, True, True, True, False,  
True, True, True, False, True, False, False, False, False, True,  
True, False, True, True, True, True, True, False, True, False,  
True, False, True, True, False, True, True, True, False, False,  
False, False, True, False, False, True, False, False, False, False,  
False, False, True, True, True, True, True, False, True, False,  
False, True, True, False, True, True, True, True, False, True,  
False, False, False, True, True, False, False, False, True, False,  
True, False, False, True, True, False, True, False, True, True,  
True, True, True, False, True, True, True, False, False, True,  
True])
```

```
print(x_train.shape)
```

(280,7)

#libraries to train Neural network

```
import tensorflow as tf
```

```
from tensorflow import keras
```

```
from tensorflow.keras.layers import Dense,Activation,Dropout
```

```
from tensorflow.keras.optimizers import Adam
```

#Initialize the model

```
model=keras.Sequential()
```

#Add input layer

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

#Add hidden layers

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

#Add output layer

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

```
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

=====

Total params: 120

Trainable params: 120

Non-trainable params: 0

```
model.compile(loss='mse',optimizer='rmsprop',metrics=['accuracy'])
```

```
model.fit(x_train,y_train,batch_size=20,epochs=30)
```

Epoch 1/30

14/14 [=====] - 1s 3ms/step -
loss: 0.6353 - accuracy: 0.4000

Epoch 2/30

14/14 [=====] - 0s 2ms/step -
loss: 0.5872 - accuracy: 0.4000

Epoch 3/30

14/14 [=====] - 0s 2ms/step -
loss: 0.5303 - accuracy: 0.4000

Epoch 4/30

14/14 [=====] - 0s 2ms/step -
loss: 0.1819 - accuracy: 0.7321

Epoch 26/30

14/14 [=====] - 0s 2ms/step -
loss: 0.1803 - accuracy: 0.7357

Epoch 27/30

14/14 [=====] - 0s 2ms/step -
loss: 0.1792 - accuracy: 0.7500

Epoch 28/30

14/14 [=====] - 0s 2ms/step -
loss: 0.1782 - accuracy: 0.7500

Epoch 29/30

14/14 [=====] - 0s 2ms/step -
loss: 0.1770 - accuracy: 0.7536

Epoch 30/30

14/14 [=====] - 0s 2ms/step -
loss: 0.1757 - accuracy: 0.7536

<keras.callbacks.History at 0x7f55c8f9ce20>

`from sklearn.metrics import accuracy_score`

`#Make predications on the training data`

`train_predications=model.predict(x_train)`

`print(train_predications)`

9/9 [=====] - 0s 2ms/step

`[[0.8617917]`

`[0.8685524]`

`[0.6542671]`

`[0.9855276]`

`[0.6590128]`

`[1.0318804]`

```
[0.84019667]
[0.08158495]
[0.8340557 ]
[0.9735674 ]
[0.60013396]]
```

#get the training accuracy

```
train_acc=model.evaluate(x_train,y_train,verbose=0)[1]
```

```
print(train_acc)
0.7571428418159485
```

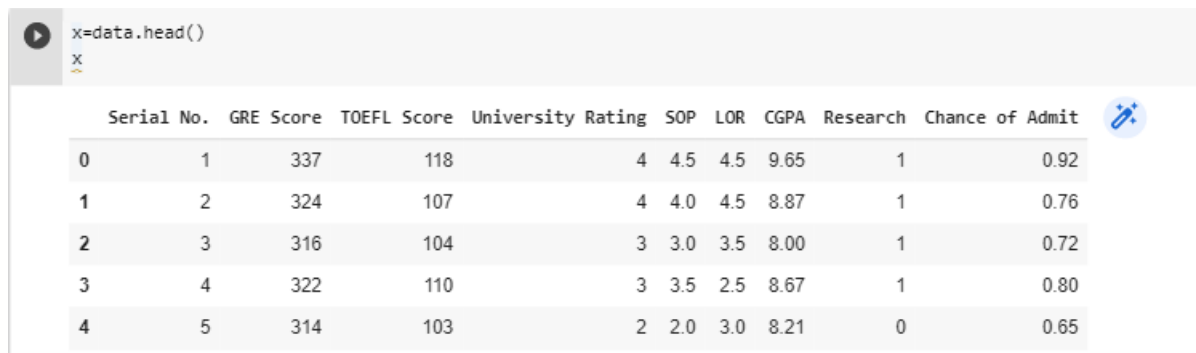
#get the test accuracy

```
test_acc=model.evaluate(x_test,y_test,verbose=0)[1]
```

```
print(test_acc)
0.6416666507720947
```

```
x=data.head()
```

x



The screenshot shows a Jupyter Notebook cell with the code `x=data.head()` and the variable `x` assigned. Below the code, the first five rows of the DataFrame are displayed as a table.

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

```
pred=model.predict(x_test)
```

```
pred=(pred>0.5)
```

```
pred
```

```
4/4 [=====] - 0s 4ms/step
```

```
array([[ True],
```

```
       [False],
```

```
       [False],
```

```
       [False],
```

```
       [False],
```

```
       [ True],
```

```
       [ True],
```

```
       [False],
```

```
       [ True],
```

```
       [ True],
```

```
       [ True],
```

```
       [False],
```

```
       [ True],
```

```
       [ True],
```

```
       [False],
```

```
       [ True],
```

```
       [ True],
```

```
       [False],
```

```
       [False],
```

```
       [False],
```

```
       [ True],
```

```
       [ True],
```

```
       [False],
```

```
       [False],
```

```
       [ True],
```

```
[False],  
[ True],  
[False],  
[ True],  
[ True],  
[ True],  
[ True],  
[ True],  
[False],  
[ True],  
[ True],  
[ True],  
[False],  
[False],  
[ True],  
[ True]])
```

```
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matrix
```

```
print("Accuracy 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))
```

Accuracy score:71.666667

Recall score:78.431373

ROC score 72.549020

```
[[46 23]
 [11 40]]
```

```
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matrix, classification_report
```

```
print(classification_report(y_train, pred))
```

```
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matrix, classification_report
```

```
print(classification_report(y_test, pred))
```

```
precision    recall  f1-score   support
```

False	0.81	0.49	0.61	69
True	0.55	0.84	0.67	51

accuracy			0.64	120
macro avg	0.68	0.67	0.64	120
weighted avg	0.70	0.64	0.64	120

```
pickle.dump(cls, open('university.pkl', 'wb'))
```