In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
data=pd.read_csv(r"C://Users//Jamboree_Admission.csv")
```

In [3]:

```
data.head(10)
```

Out[3]:

	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
5	6	330	115	5	4.5	3.0	9.34	1	0.90
6	7	321	109	3	3.0	4.0	8.20	1	0.75
7	8	308	101	2	3.0	4.0	7.90	0	0.68
8	9	302	102	1	2.0	1.5	8.00	0	0.50
9	10	323	108	3	3.5	3.0	8.60	0	0.45

In [4]:

```
data.describe()
```

Out[4]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	L(
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.484
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.925
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.500
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.000

In [5]:

```
data.isnull().sum()
```

Out[5]:

```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

In [7]:

```
data["University Rating"].astype("category")
```

Out[7]:

```
0      4
1      4
2      3
3      3
4      2
      ..
495    5
496    5
497    5
498    4
499    4
Name: University Rating, Length: 500, dtype: category
Categories (5, int64): [1, 2, 3, 4, 5]
```

In [8]:

```
data["University Rating"].value_counts()
```

Out[8]:

```
3    162
2    126
4    105
5     73
1     34
Name: University Rating, dtype: int64
```

In [9]:

```
data["Research"].astype("category")
```

Out[9]:

```
0      1
1      1
2      1
3      1
4      0
      ..
495    1
496    1
497    1
498    0
499    0
Name: Research, Length: 500, dtype: category
Categories (2, int64): [0, 1]
```

In [10]:

```
data=data.drop("Serial No.",axis=1)
```

In [11]:

```
data.describe()
```

Out[11]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CG
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.0000
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.5764
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.6048
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.8000
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.1275
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.5600
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.0400
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.9200

In [12]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   GRE Score          500 non-null    int64  
 1   TOEFL Score        500 non-null    int64  
 2   University Rating  500 non-null    int64  
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64  
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 31.4 KB
```

In [13]:

```
data
```

Out[13]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...
495	332	108	5	4.5	4.0	9.02	1	0.87
496	337	117	5	5.0	5.0	9.87	1	0.96
497	330	120	5	4.5	5.0	9.56	1	0.93
498	312	103	4	4.0	5.0	8.43	0	0.73
499	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 8 columns

In [14]:

```
data["University Rating"].value_counts().sort_values()
```

Out[14]:

```
1     34
5     73
4    105
2    126
3    162
Name: University Rating, dtype: int64
```

In [15]:

```
data["Research"].value_counts()
```

Out[15]:

```
1    280
0    220
Name: Research, dtype: int64
```

In [16]:

```
pd.qcut(data["TOEFL Score"],q=5).value_counts()
```

Out[16]:

```
(91.999, 102.0]    122
(109.0, 113.0]     111
(105.0, 109.0]      94
(102.0, 105.0]      91
(113.0, 120.0]      82
Name: TOEFL Score, dtype: int64
```

In [17]:

```
data["University Rating"].value_counts().sort_values()
```

Out[17]:

```
1     34
5     73
4    105
2    126
3    162
Name: University Rating, dtype: int64
```

In [18]:

```
pd.crosstab(index=data["University Rating"],columns=data["SOP"])
```

Out[18]:

SOP	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
University Rating									
1	5	11	9	3	4	2	0	0	0
2	1	12	19	42	28	11	10	3	0
3	0	0	14	16	34	61	30	5	2
4	0	2	1	3	12	11	31	32	13
5	0	0	0	0	2	3	18	23	27

1. we see that univerty rating low , then sop of 4.5, 5 are 0
2. we see that univeristy rating high and sop of 1-3 are low

In [19]:

```
data["Chances"]=pd.qcut(data["Chance of Admit "],[0, .25, .5, .75, 1.],lab
```

In [20]:

```
data["Chances"].value_counts()
```

Out[20]:

```
Low          127
Medium       125
High         124
Very high    124
Name: Chances, dtype: int64
```

In [21]:

```
data
```

Out[21]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit	Cha
0	337	118	4	4.5	4.5	9.65	1	0.92	Very
1	324	107	4	4.0	4.5	8.87	1	0.76	
2	316	104	3	3.0	3.5	8.00	1	0.72	Me
3	322	110	3	3.5	2.5	8.67	1	0.80	
4	314	103	2	2.0	3.0	8.21	0	0.65	Me
...	...	...	...	...	...	...	...	...	
495	332	108	5	4.5	4.0	9.02	1	0.87	Very
496	337	117	5	5.0	5.0	9.87	1	0.96	Very
497	330	120	5	4.5	5.0	9.56	1	0.93	Very
498	312	103	4	4.0	5.0	8.43	0	0.73	
499	327	113	4	4.5	4.5	9.04	0	0.84	Very

500 rows × 9 columns

In [ ]:

In [ ]:

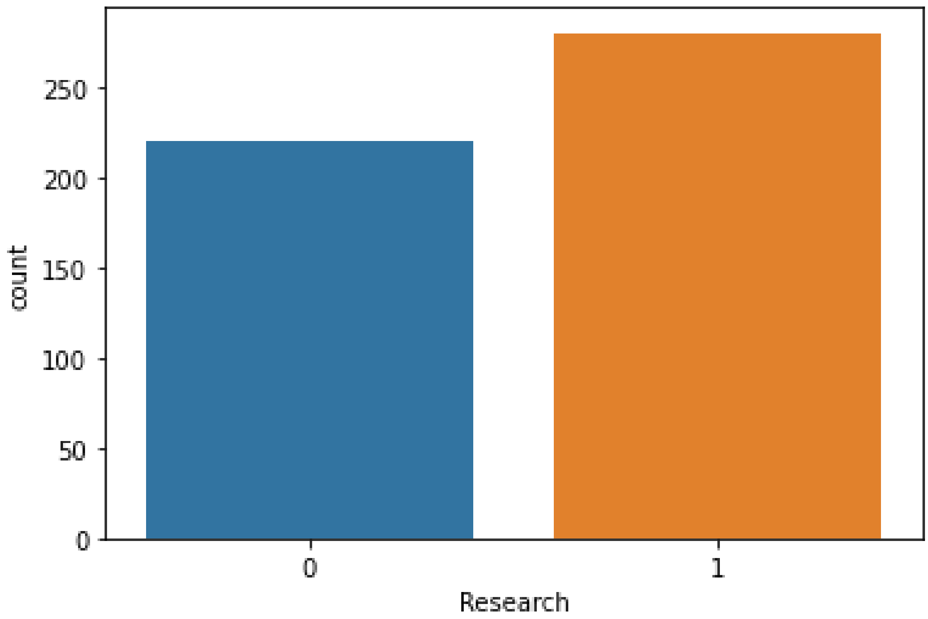
#UNIVARIATE ANALSYIS

In [22]:

```
sns.countplot(x=data["Research"])
```

Out[22]:

```
<AxesSubplot:xlabel='Research', ylabel='count'>
```

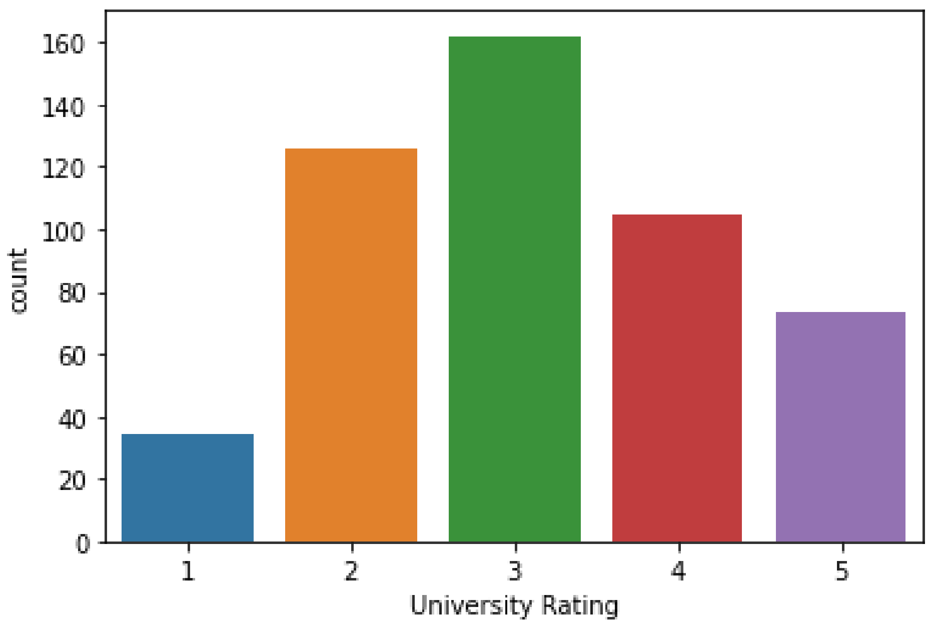


In [23]:

```
sns.countplot(x=data["University Rating"])
```

Out[23]:

```
<AxesSubplot:xlabel='University Rating', ylabel='count'>
```

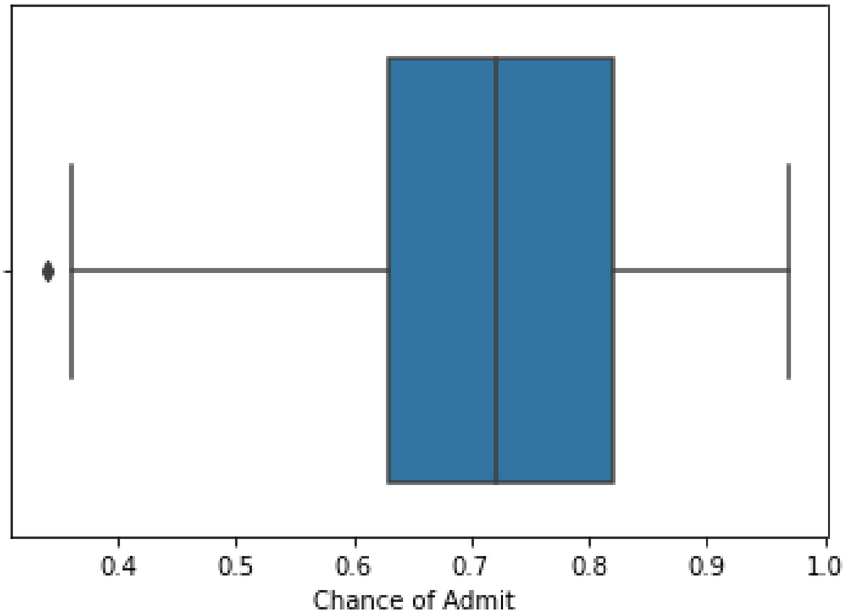


In [24]:

```
sns.boxplot(x=data["Chance of Admit "])
```

Out[24]:

```
<AxesSubplot:xlabel='Chance of Admit '>
```



#OUTLIER TREATMENT

In [25]:

```
a=data["Chance of Admit "].quantile(0.25)
b=data["Chance of Admit "].quantile(0.75)
a,b
```

Out[25]:

```
(0.63, 0.82)
```

In [26]:

```
1.5*(b-a)
```

Out[26]:

```
0.2849999999999999
```

In [27]:

```
iqr_lower_limit=0.63-1.5*(b-a)
iqr_lower_limit
```

Out[27]:

```
0.3450000000000001
```

In [28]:

```
len(data[data["Chance of Admit "]<0.3450000000000001])
```

Out[28]:

```
2
```

In [29]:

```
data["Chance of Admit "]=np.where(data["Chance of Admit "]<iqr_lower_limit
```

In [30]:

```
len(data[data["Chance of Admit "]<iqr_lower_limit])
```

Out[30]:

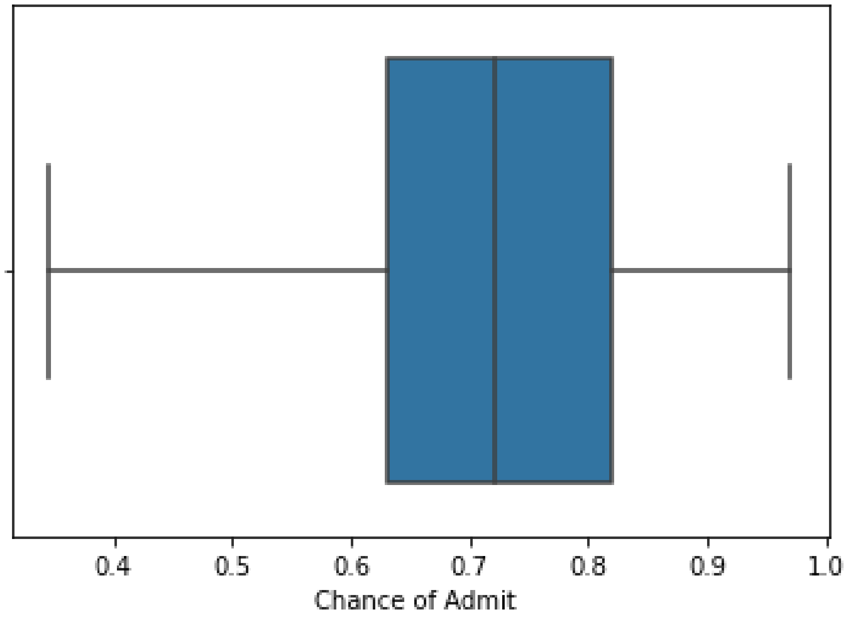
```
0
```

In [31]:

```
sns.boxplot(x=data["Chance of Admit "])
```

Out[31]:

<AxesSubplot:xlabel='Chance of Admit '>

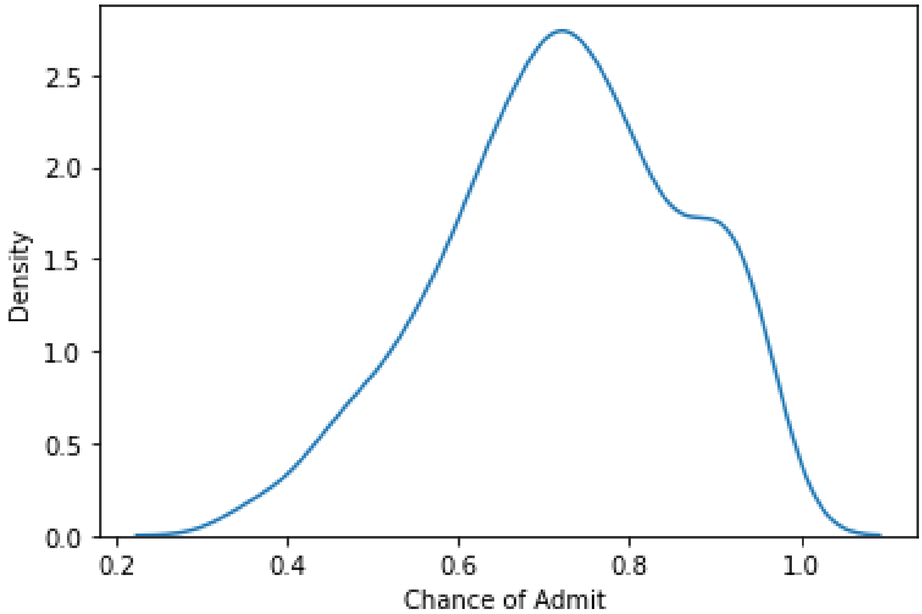


In [32]:

```
sns.kdeplot(data["Chance of Admit "])
```

Out[32]:

<AxesSubplot:xlabel='Chance of Admit ', ylabel='Density'>

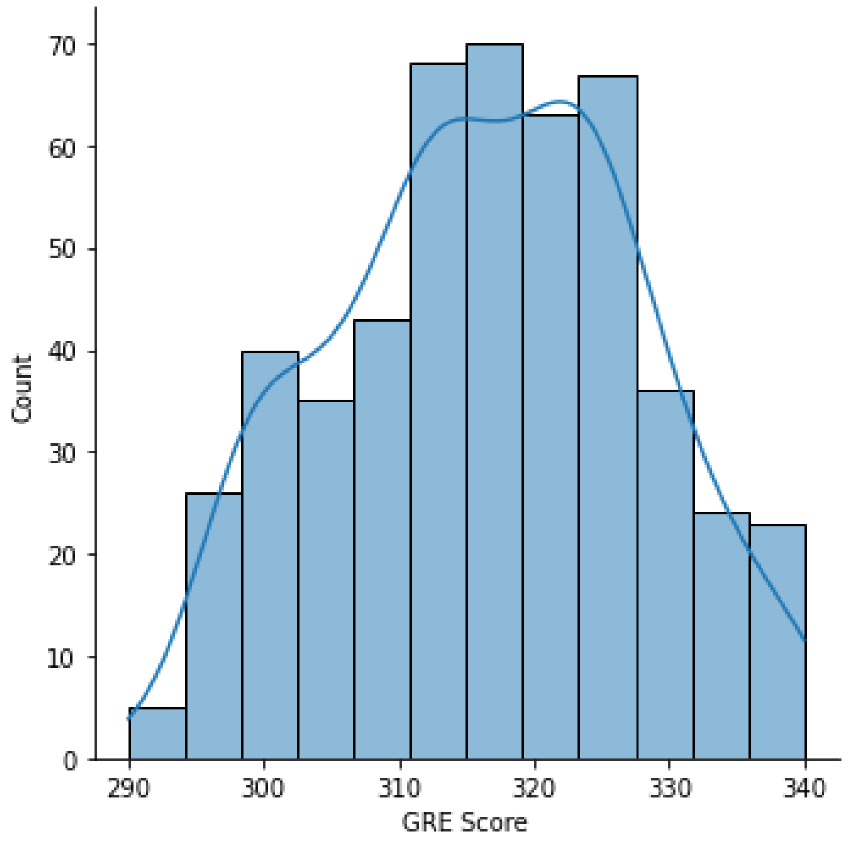


In [33]:

```
sns.displot(x=data["GRE Score"],kde=True)
```

Out[33]:

```
<seaborn.axisgrid.FacetGrid at 0x28fd1cd7880>
```

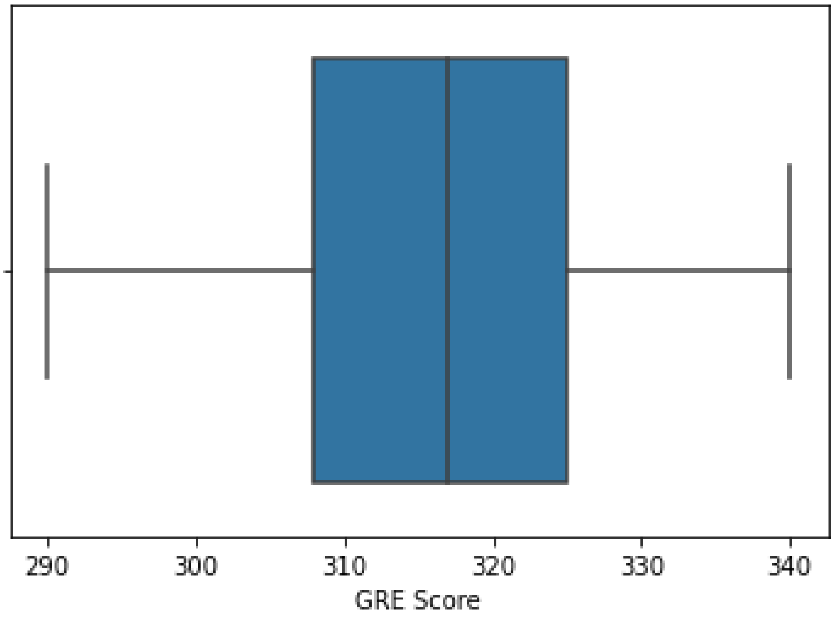


In [34]:

```
sns.boxplot(x=data["GRE Score"])
```

Out[34]:

<AxesSubplot:xlabel='GRE Score'>

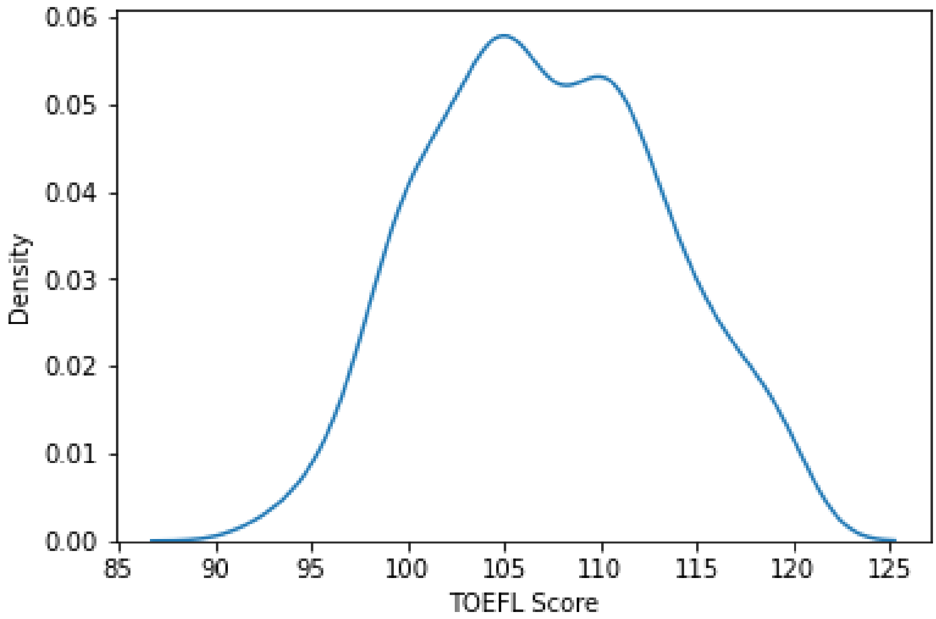


In [35]:

```
sns.kdeplot(data["TOEFL Score"])
```

Out[35]:

<AxesSubplot:xlabel='TOEFL Score', ylabel='Density'>

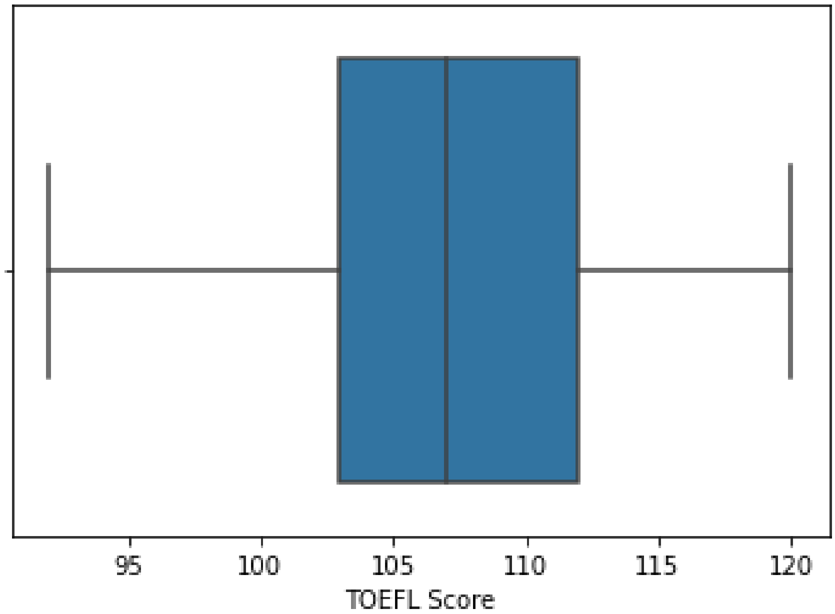


In [36]:

```
sns.boxplot(x=data["TOEFL Score"])
```

Out[36]:

```
<AxesSubplot:xlabel='TOEFL Score'>
```

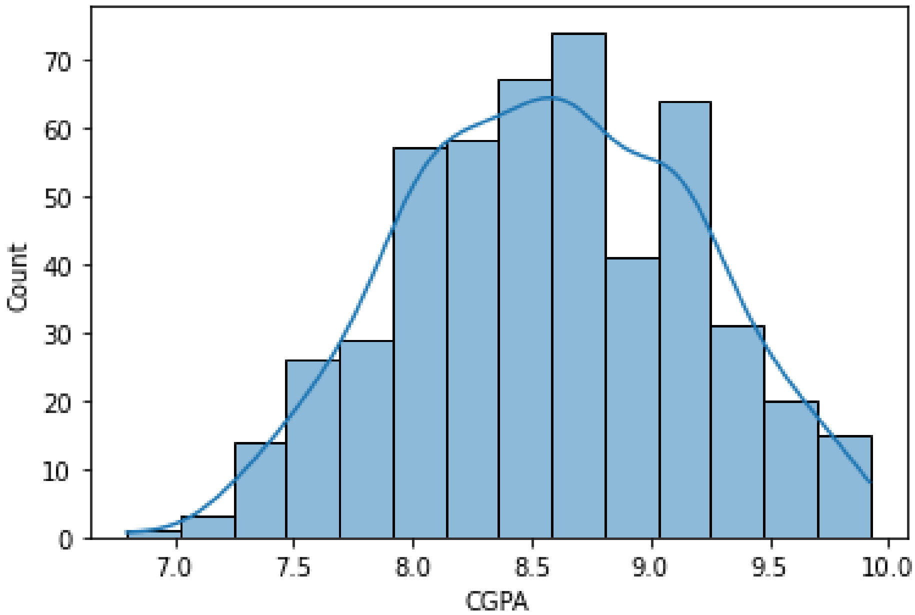


In [37]:

```
sns.histplot(data["CGPA"],kde=True)
```

Out[37]:

```
<AxesSubplot:xlabel='CGPA', ylabel='Count'>
```

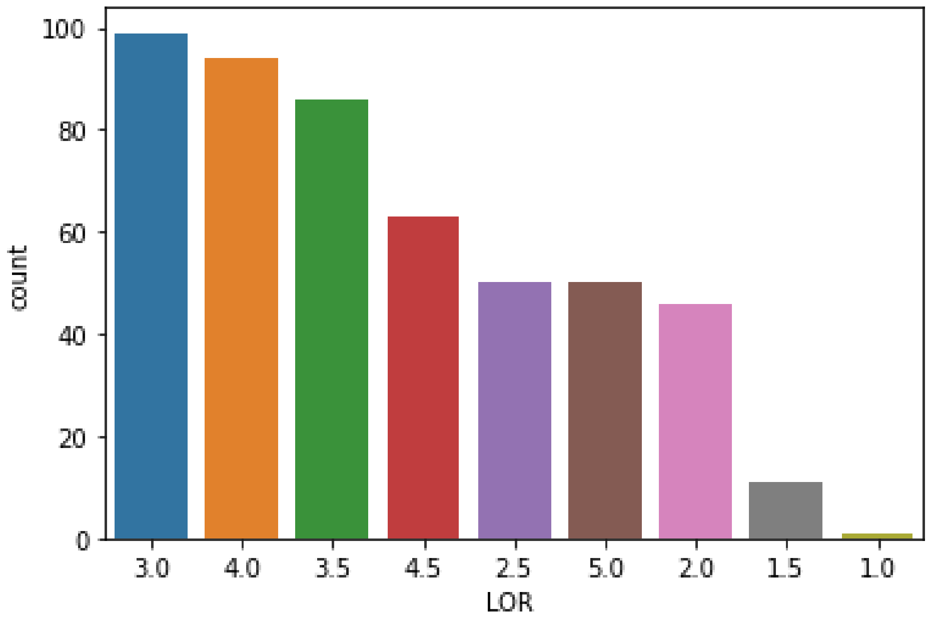


In [38]:

```
sns.countplot(x=data["LOR "],order = data["LOR "].value_counts().index)
```

Out[38]:

<AxesSubplot:xlabel='LOR ', ylabel='count'>

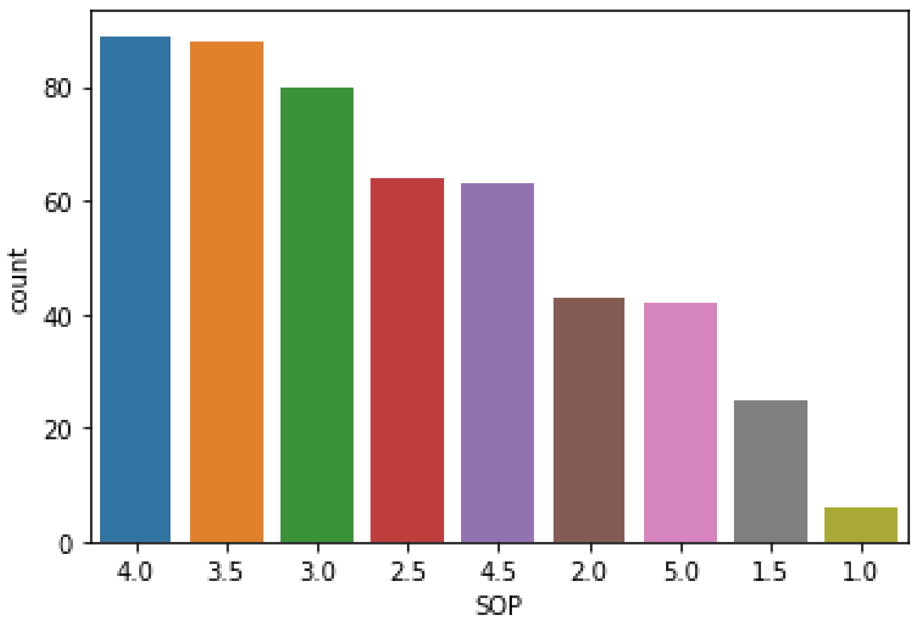


In [39]:

```
sns.countplot(x=data["SOP"],order = data["SOP"].value_counts().index)
```

Out[39]:

<AxesSubplot:xlabel='SOP', ylabel='count'>



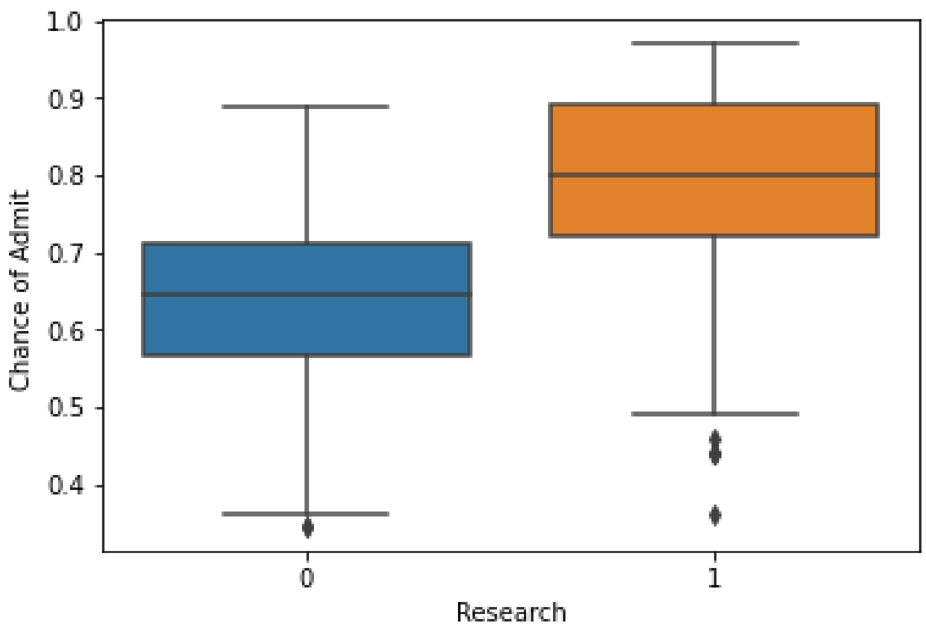
BIVARIATE ANALYSIS

In [40]:

```
sns.boxplot(x=data["Research"],y=data["Chance of Admit "])
```

Out[40]:

```
<AxesSubplot:xlabel='Research', ylabel='Chance of Admit '>
```



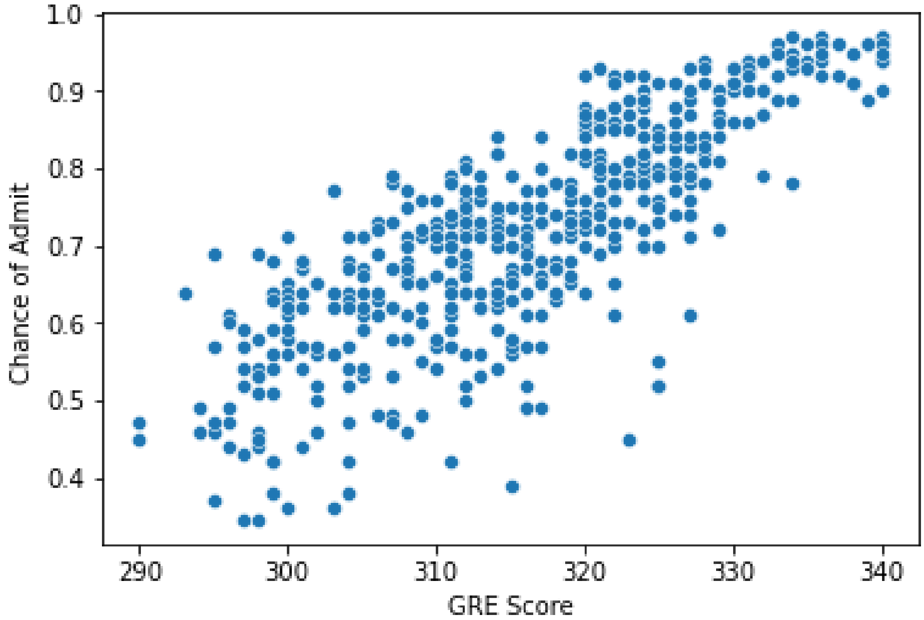
We can see that the average chances of admission are considerably high in case the students have some research experience

In [41]:

```
sns.scatterplot(x=data["GRE Score"],y=data["Chance of Admit "],data=data)
```

Out[41]:

```
<AxesSubplot:xlabel='GRE Score', ylabel='Chance of Admit '>
```



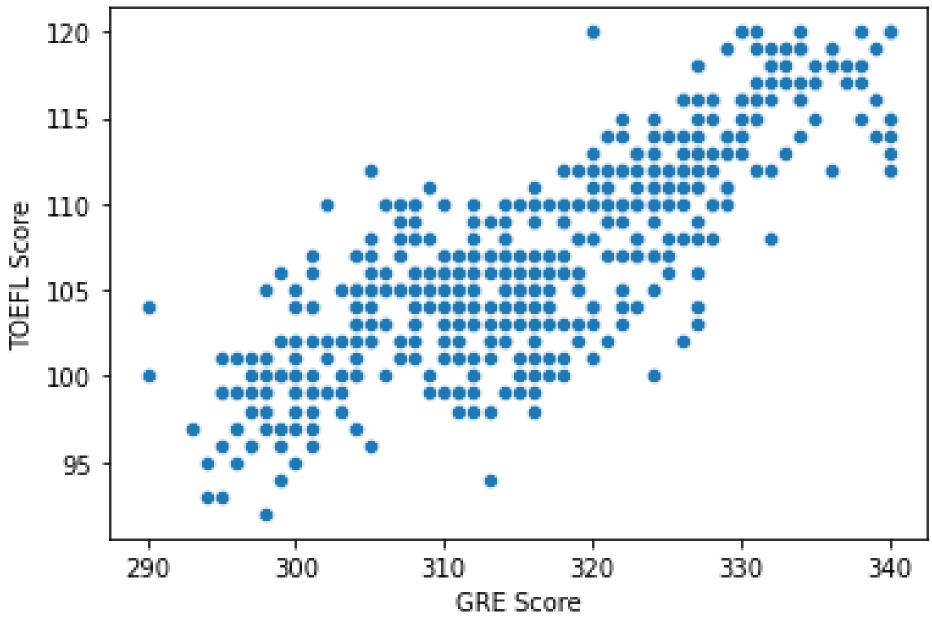
We can see that as GRE score increases so does the Chances of Admit

In [42]:

```
sns.scatterplot(x=data["GRE Score"],y=data["TOEFL Score"],data=data)
```

Out[42]:

```
<AxesSubplot:xlabel='GRE Score', ylabel='TOEFL Score'>
```

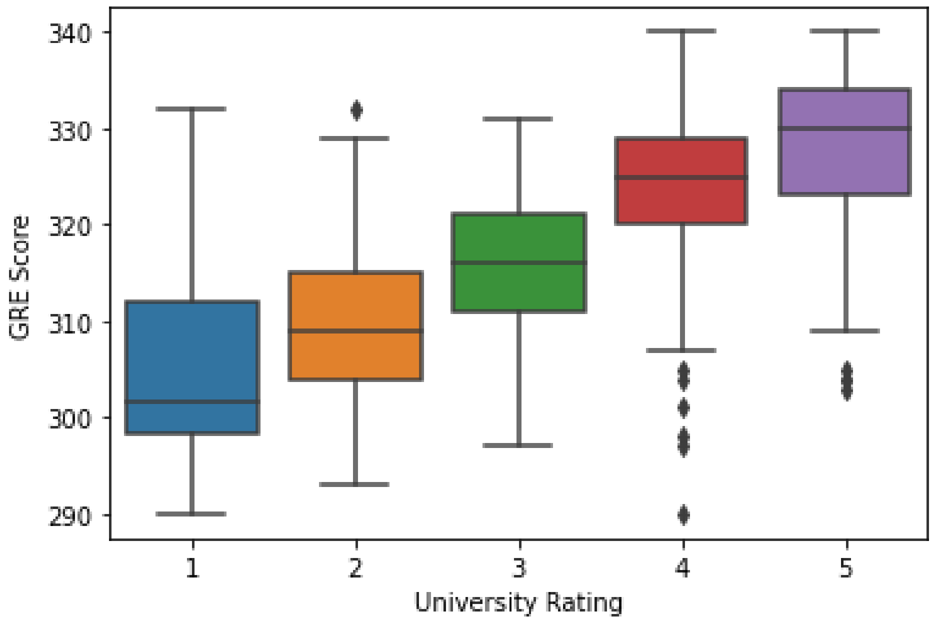


In [43]:

```
sns.boxplot(x=data["University Rating"],y=data["GRE Score"])
```

Out[43]:

```
<AxesSubplot:xlabel='University Rating', ylabel='GRE Score'>
```



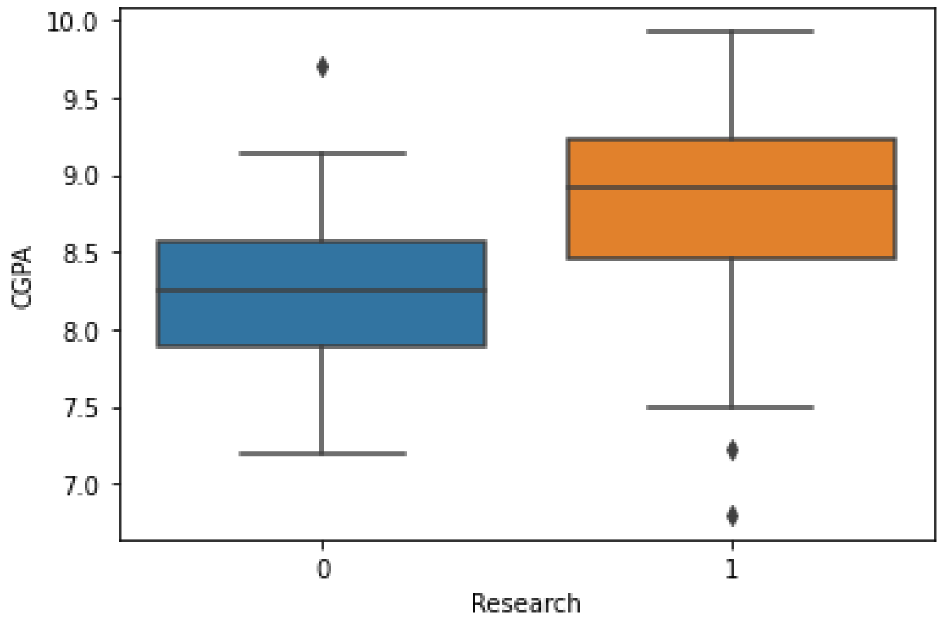
```
Interesting Observation
1. In univesity ranking with 4 and 5. there are some outliers wher
e in GRE Score is low as opposite to the trend
```

In [44]:

```
sns.boxplot(x=data["Research"],y=data["CGPA"])
```

Out[44]:

<AxesSubplot:xlabel='Research', ylabel='CGPA'>

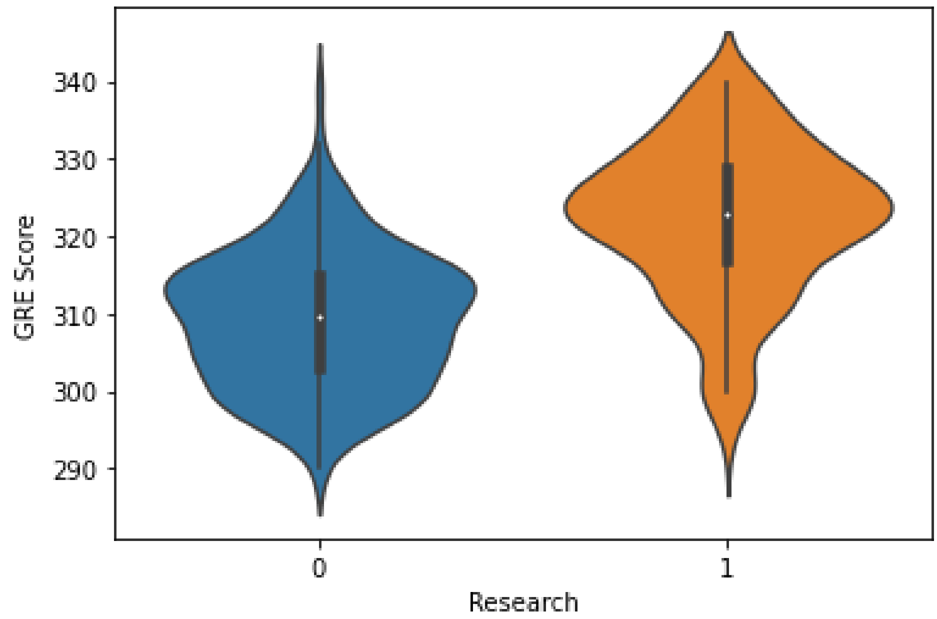


In [45]:

```
sns.violinplot(x=data["Research"],y=data["GRE Score"])
```

Out[45]:

```
<AxesSubplot:xlabel='Research', ylabel='GRE Score'>
```

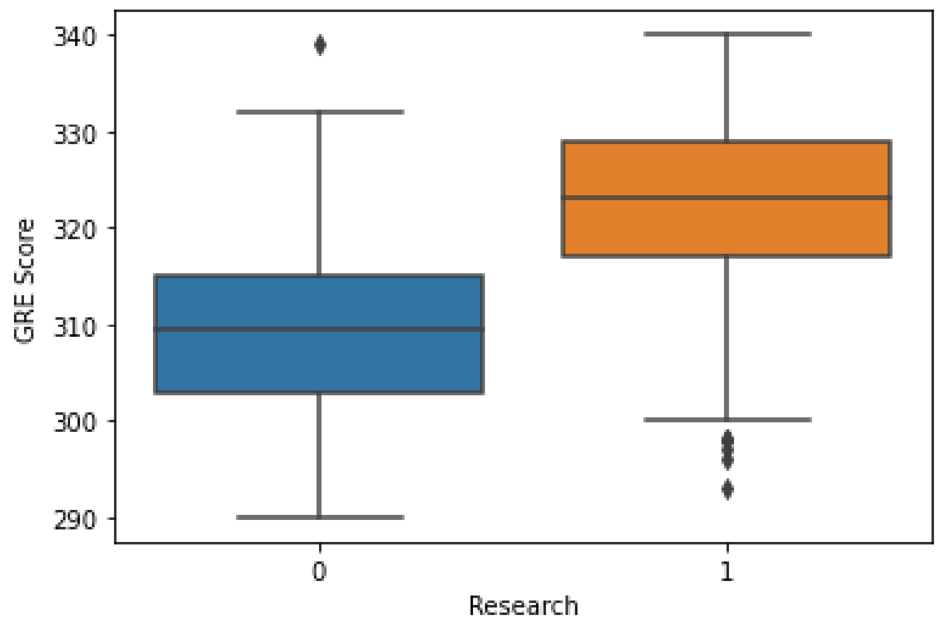


In [46]:

```
sns.boxplot(x=data["Research"],y=data["GRE Score"])
```

Out[46]:

```
<AxesSubplot:xlabel='Research', ylabel='GRE Score'>
```



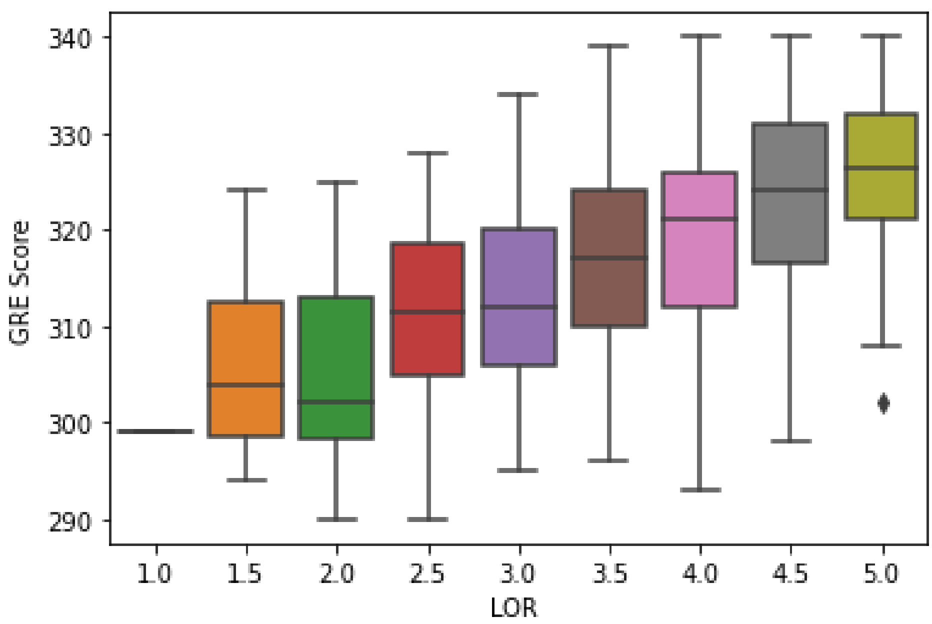
```
<font size='6'>
1.Students with Research Experience have more Median GRE Scores
2.Also as vsisible from the violinplot :-
    * For Students  with **Research Experience** :-  GRE marks dis
tirbution is more centered at 325
    * For Students with No **Research Experience** :- GRE marks di
stribution revolve around at 315<font>
```

In [47]:

```
sns.boxplot(x=data["LOR "],y=data["GRE Score"])
```

Out[47]:

```
<AxesSubplot:xlabel='LOR ', ylabel='GRE Score'>
```

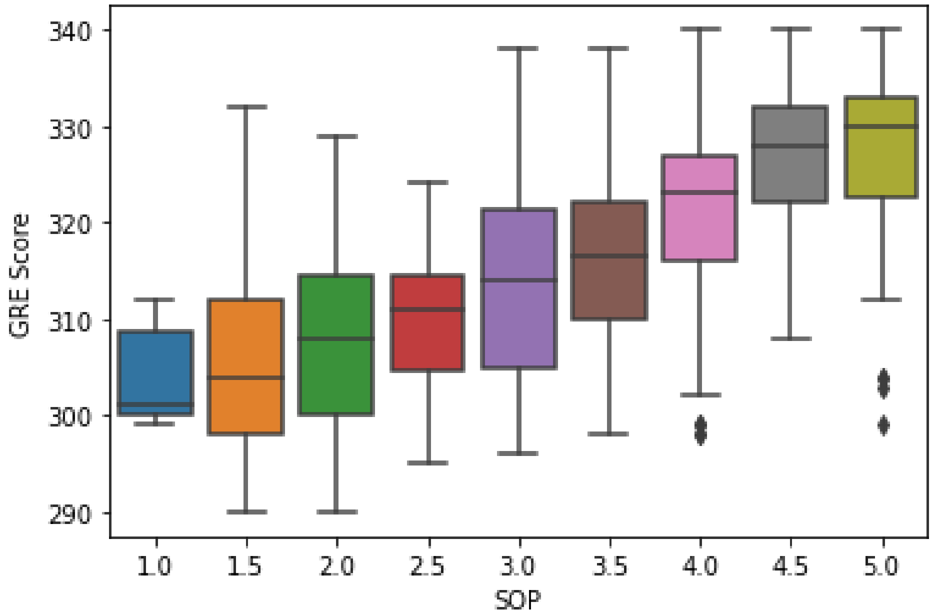


In [48]:

```
sns.boxplot(x=data["SOP"],y=data["GRE Score"])
```

Out[48]:

<AxesSubplot:xlabel='SOP', ylabel='GRE Score'>

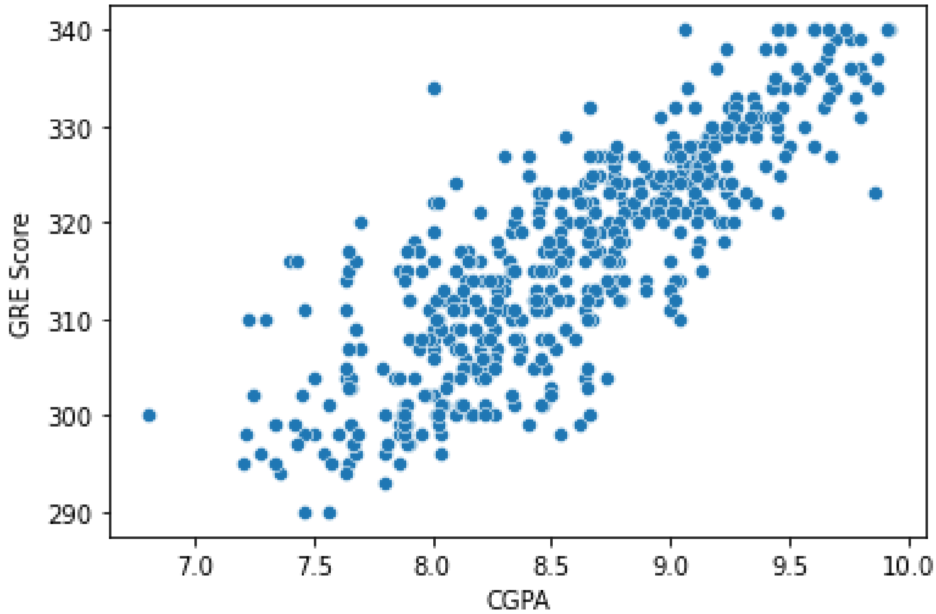


In [49]:

```
sns.scatterplot(x=data["CGPA"],y=data["GRE Score"])
```

Out[49]:

<AxesSubplot:xlabel='CGPA', ylabel='GRE Score'>

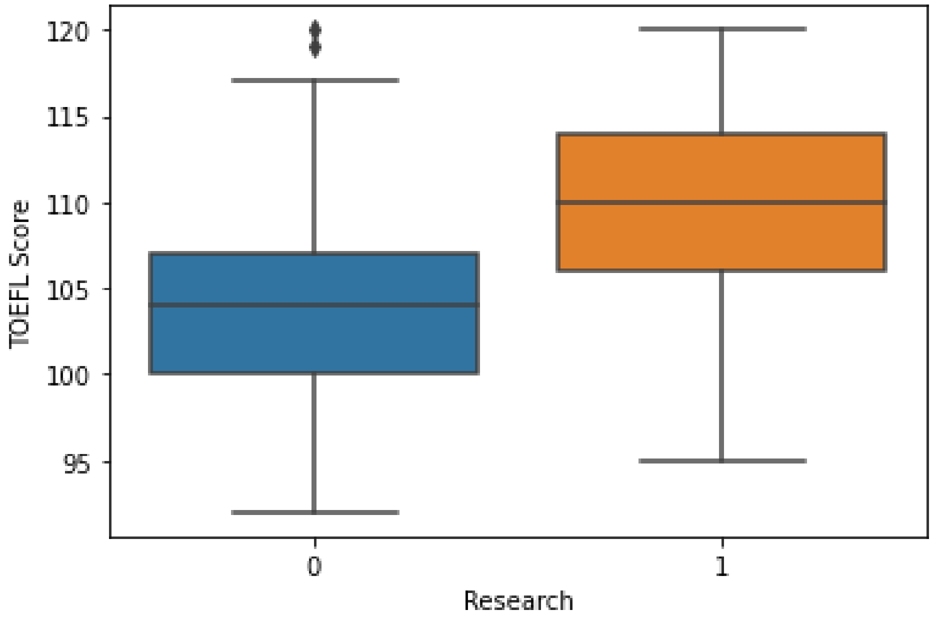


In [50]:

```
sns.boxplot(x=data["Research"],y=data["TOEFL Score"])
```

Out[50]:

```
<AxesSubplot:xlabel='Research', ylabel='TOEFL Score'>
```

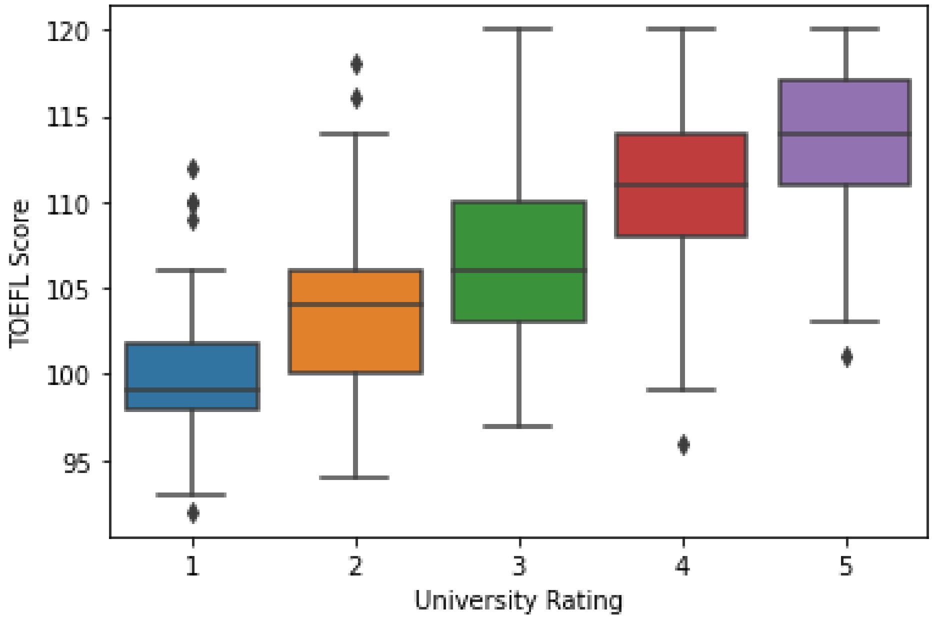


In [51]:

```
sns.boxplot(x=data["University Rating"],y=data["TOEFL Score"])
```

Out[51]:

```
<AxesSubplot:xlabel='University Rating', ylabel='TOEFL Scor
e'>
```

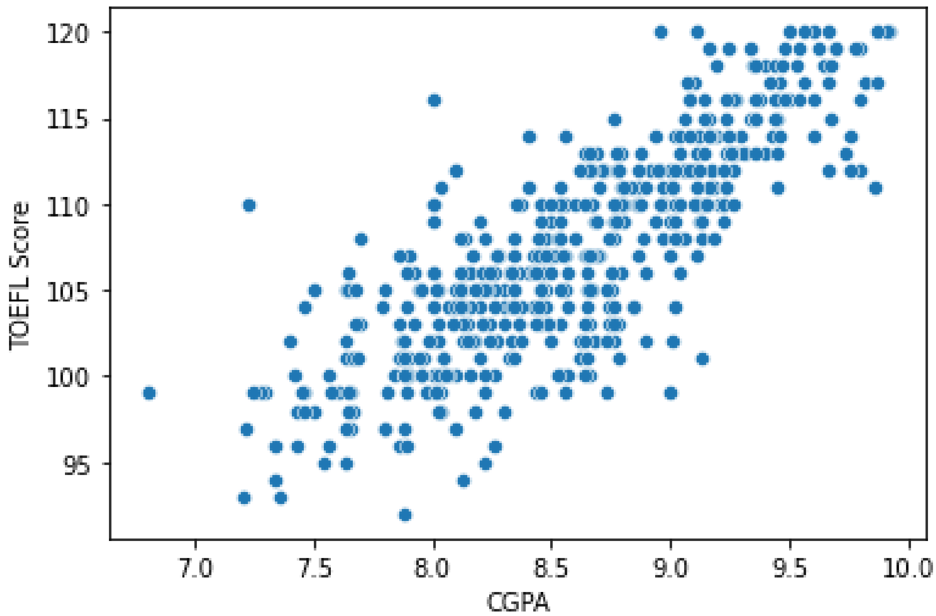


In [52]:

```
sns.scatterplot(x=data["CGPA"],y=data["TOEFL Score"])
```

Out[52]:

<AxesSubplot:xlabel='CGPA', ylabel='TOEFL Score'>

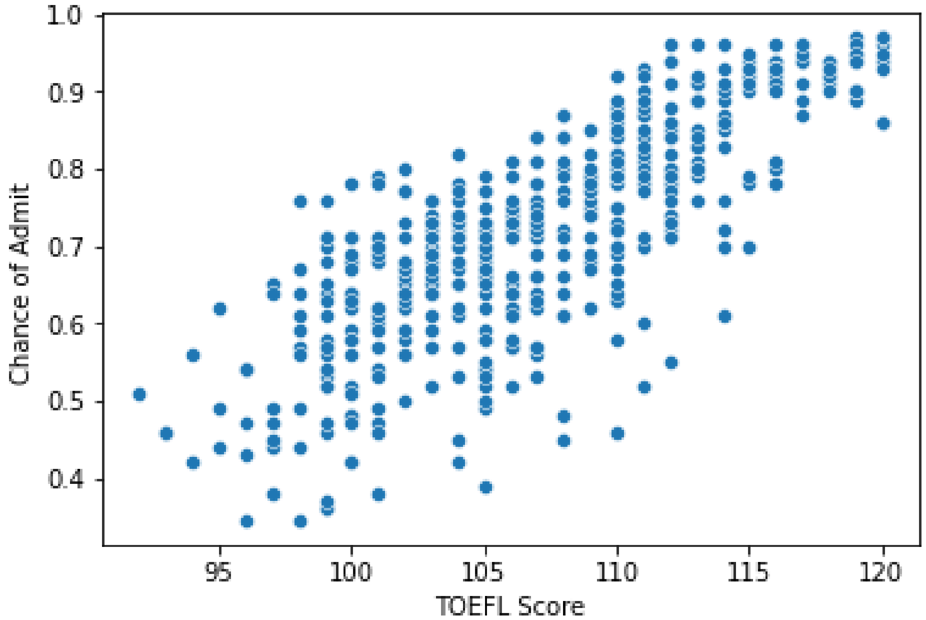


In [53]:

```
sns.scatterplot(y=data["Chance of Admit "],x=data["TOEFL Score"])
```

Out[53]:

<AxesSubplot:xlabel='TOEFL Score', ylabel='Chance of Admit
'>

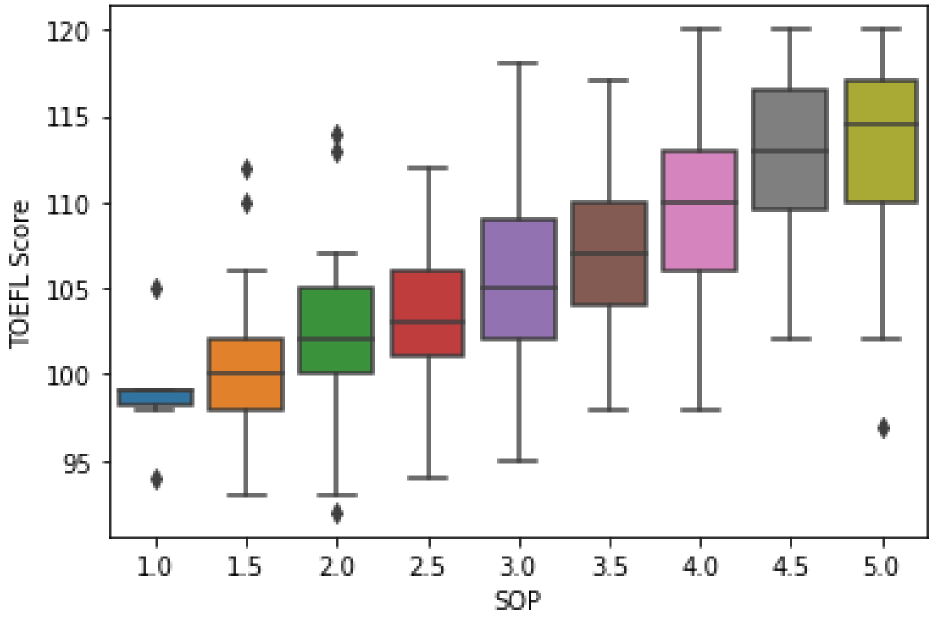


In [54]:

```
sns.boxplot(x=data["SOP"],y=data["TOEFL Score"])
```

Out[54]:

<AxesSubplot:xlabel='SOP', ylabel='TOEFL Score'>

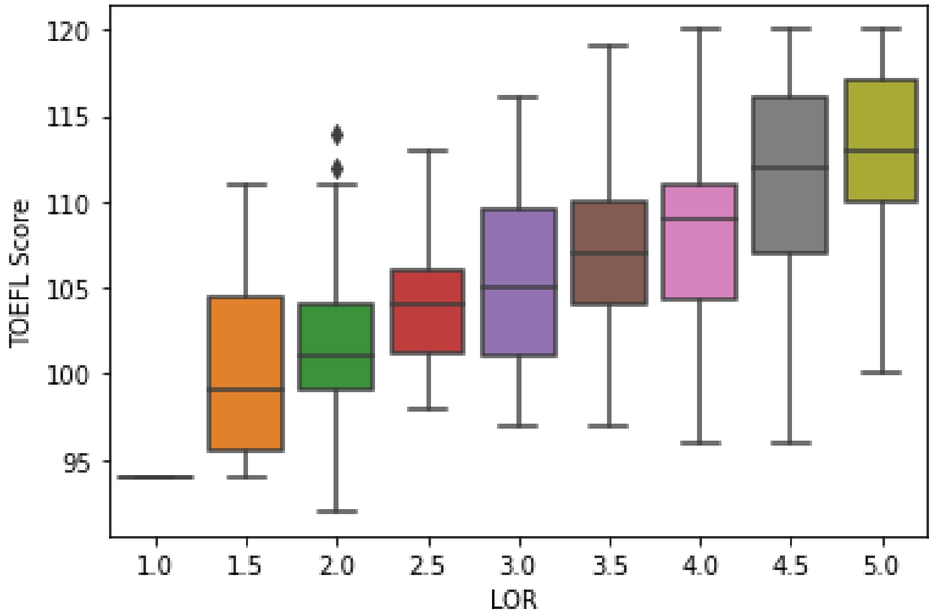


In [55]:

```
sns.boxplot(x=data["LOR "],y=data["TOEFL Score"])
```

Out[55]:

<AxesSubplot:xlabel='LOR ', ylabel='TOEFL Score'>

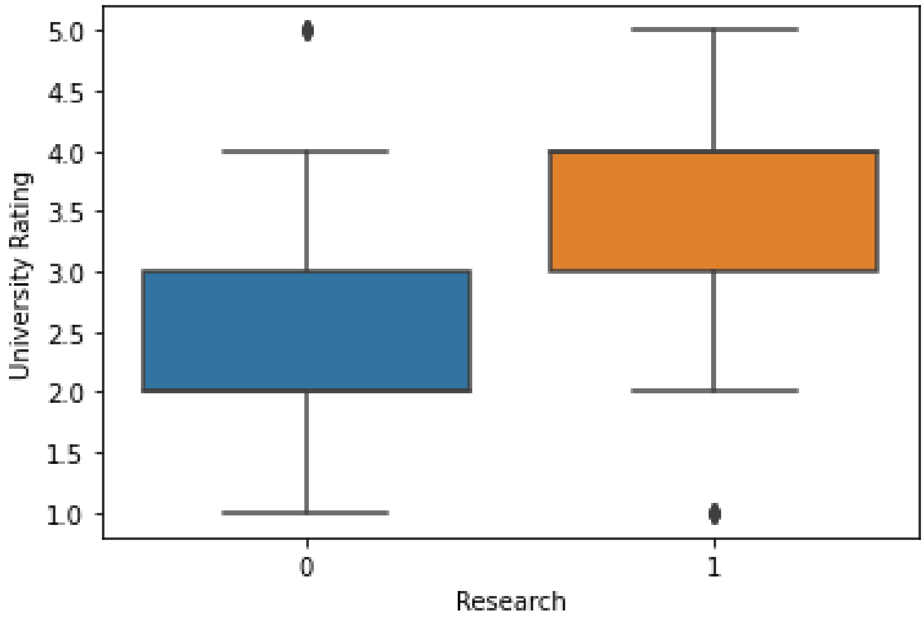


In [56]:

```
sns.boxplot(x=data["Research"],y=data["University Rating"])
```

Out[56]:

```
<AxesSubplot:xlabel='Research', ylabel='University Rating'>
```



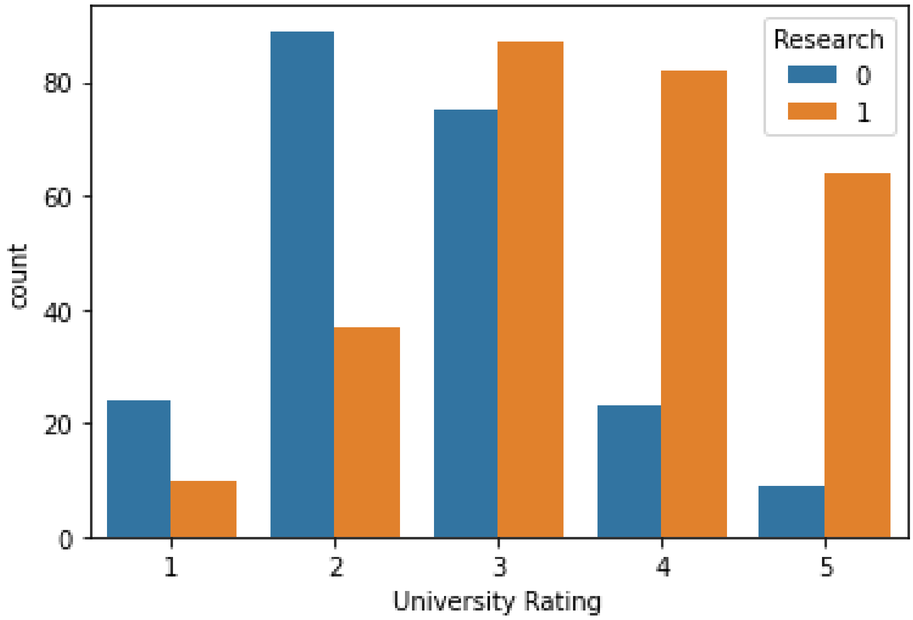
We can see that student with research experience have overall better University Ratings

In [57]:

```
sns.countplot(hue=data["Research"],x=data["University Rating"])
```

Out[57]:

<AxesSubplot:xlabel='University Rating', ylabel='count'>



STUDENTS WITH RESEARCH BACKGROUND **V/S** STUDENTS WITHOUT RESEARCH BACKGROUND

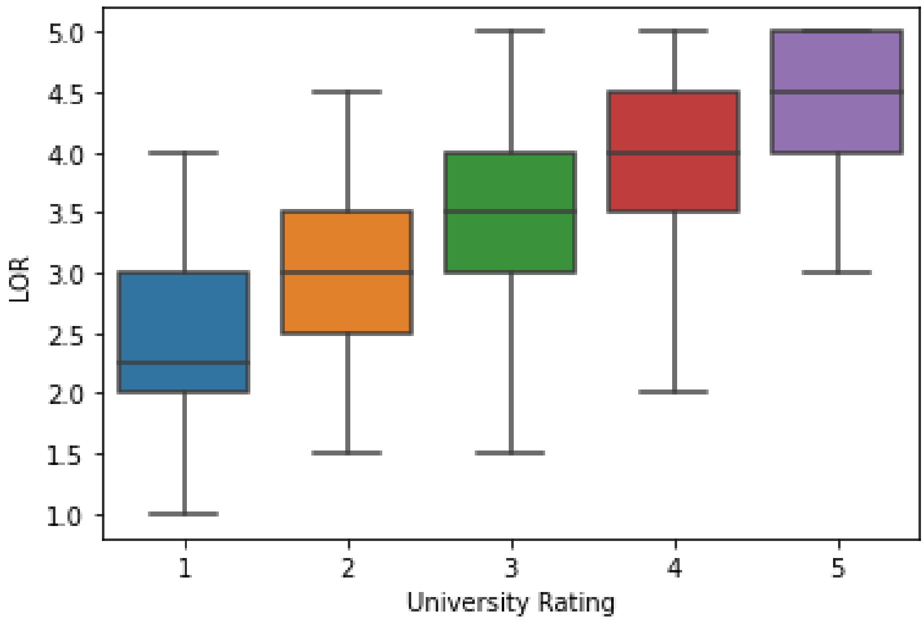
1. No. of Students **With** Research Experience are more when university Ratings Improve in comparison to students without
   research experience
2. No. of Students **Without** Research Experience are more when University Ratings are B/W 1-2 as compared to students with research experience.

In [58]:

```
sns.boxplot(x=data["University Rating"],y=data["LOR "])
```

Out[58]:

<AxesSubplot:xlabel='University Rating', ylabel='LOR '>

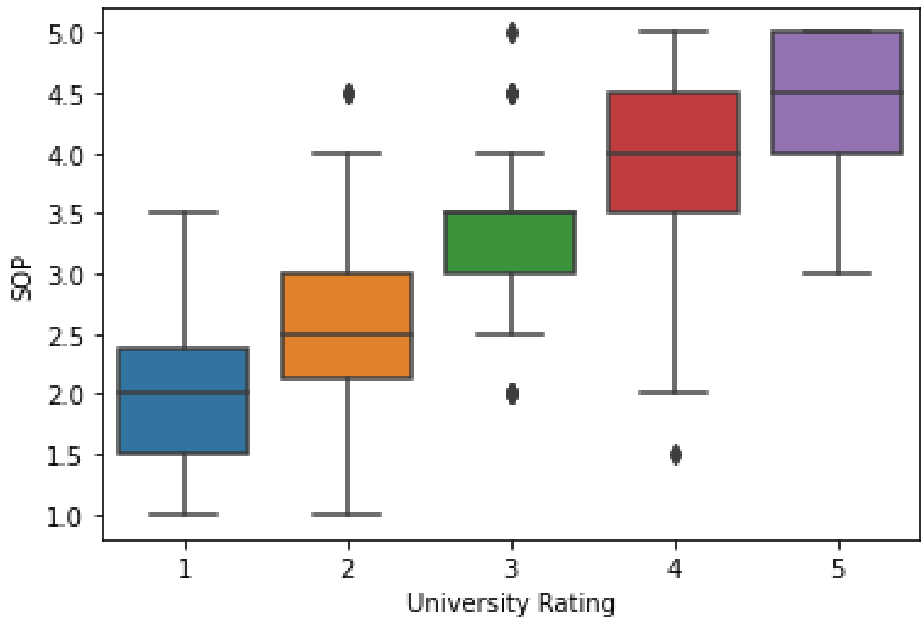


In [59]:

```
sns.boxplot(x=data["University Rating"],y=data["SOP"])
```

Out[59]:

<AxesSubplot:xlabel='University Rating', ylabel='SOP'>

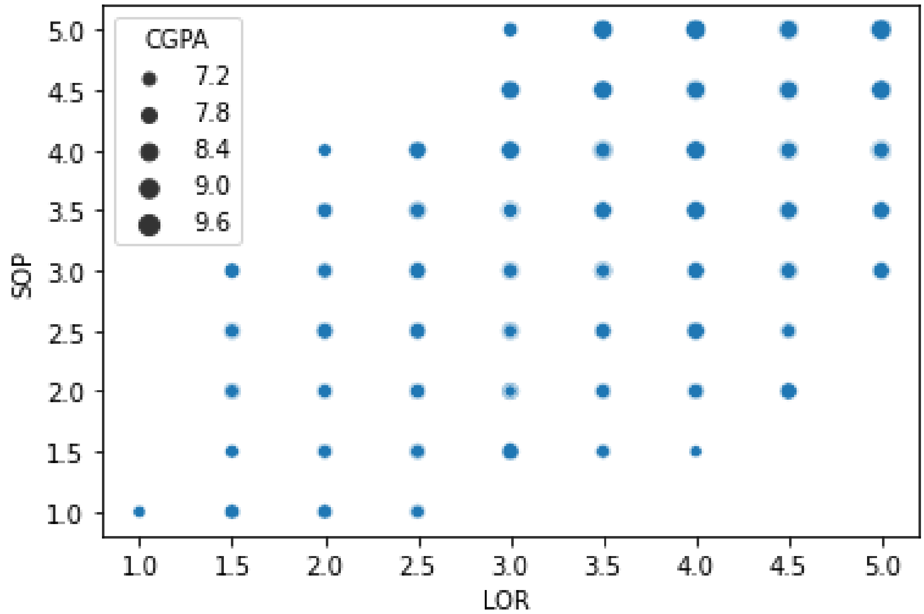


In [60]:

```
sns.scatterplot(x=data["LOR "],y=data["SOP"],size=data["CGPA"])
```

Out[60]:

```
<AxesSubplot:xlabel='LOR ', ylabel='SOP'>
```



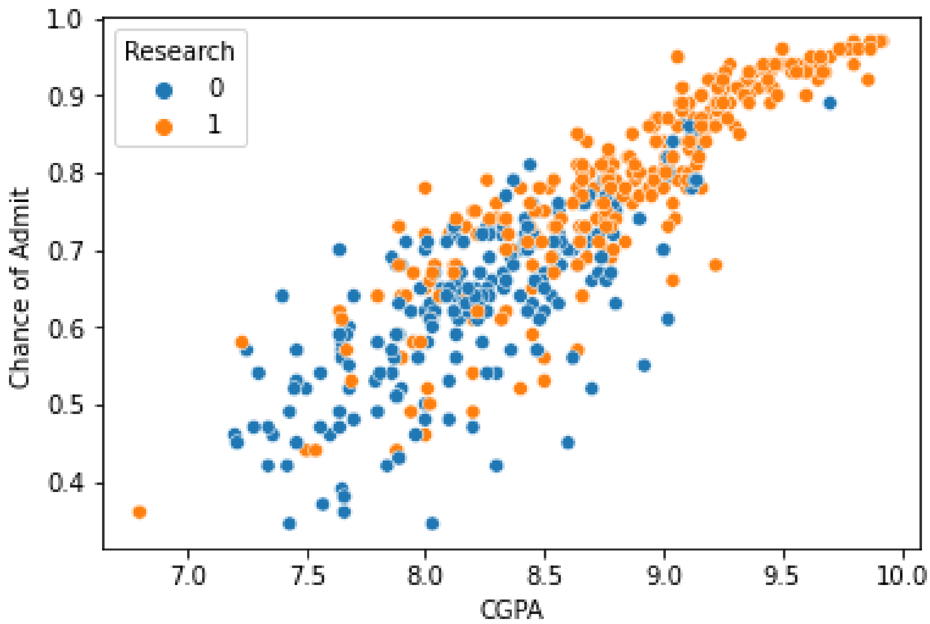
MULTIVARIATE ANALYSIS

In [61]:

```
sns.scatterplot(x=data["CGPA"],y=data["Chance of Admit "],hue=data["Researc
```

Out[61]:

```
<AxesSubplot:xlabel='CGPA', ylabel='Chance of Admit '>
```

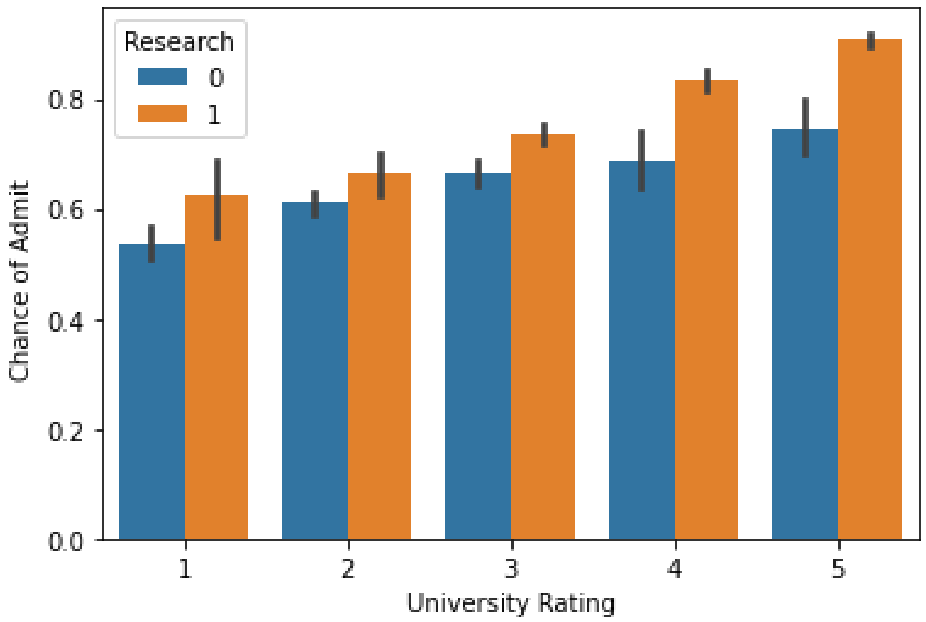


In [62]:

```
sns.barplot(hue=data["Research"],y=data["Chance of Admit "],x=data["Univers
```

Out[62]:

```
<AxesSubplot:xlabel='University Rating', ylabel='Chance of A
dmit '>
```

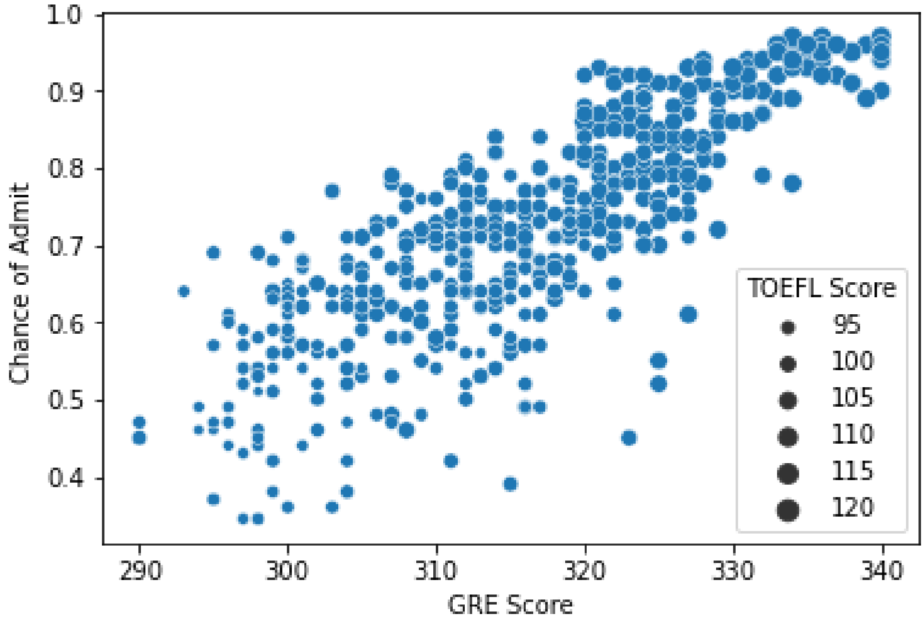


In [63]:

```
sns.scatterplot(size=data["TOEFL Score"],x=data["GRE Score"],y=data["Chanc
```

Out[63]:

```
<AxesSubplot:xlabel='GRE Score', ylabel='Chance of Admit '>
```

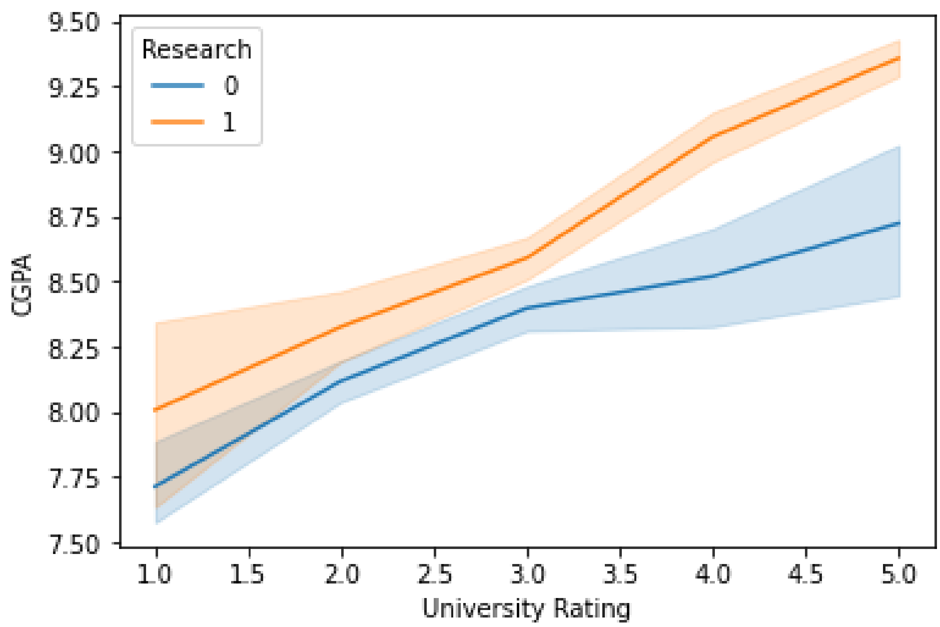


In [64]:

```
sns.lineplot(hue=data["Research"],y=data["CGPA"],x=data["University Rating
```

Out[64]:

```
<AxesSubplot:xlabel='University Rating', ylabel='CGPA'>
```

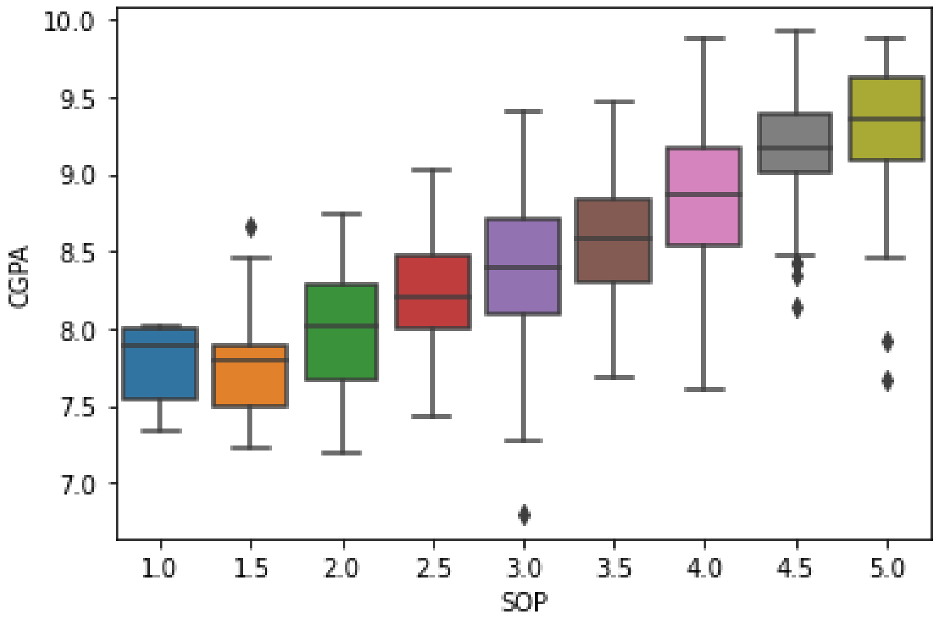


In [65]:

```
sns.boxplot(x=data["SOP"],y=data["CGPA"])
```

Out[65]:

<AxesSubplot:xlabel='SOP', ylabel='CGPA'>

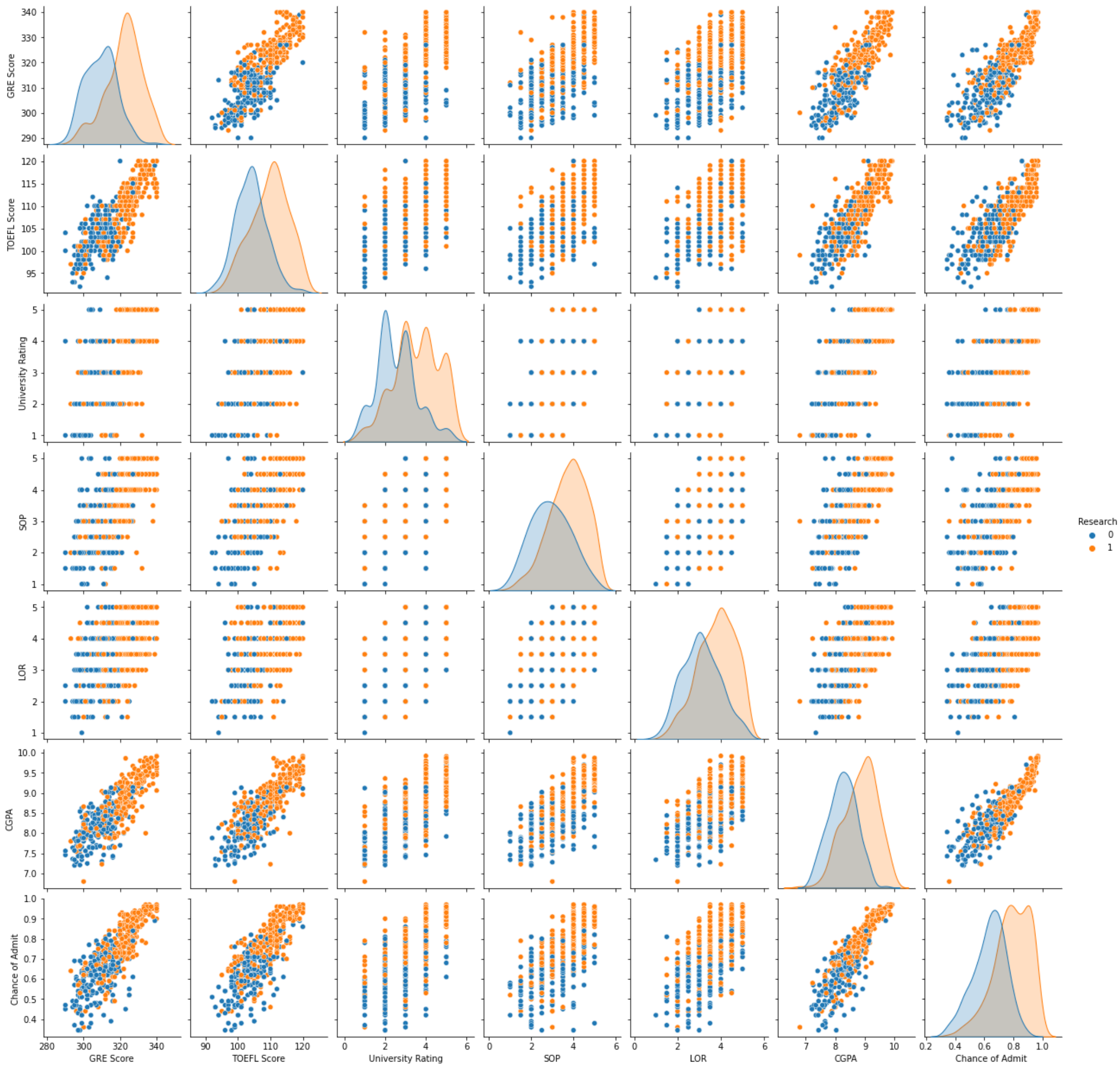


In [66]:

```
sns.pairplot(data=data,hue="Research")
```

Out[66]:

```
<seaborn.axisgrid.PairGrid at 0x28fd546abb0>
```

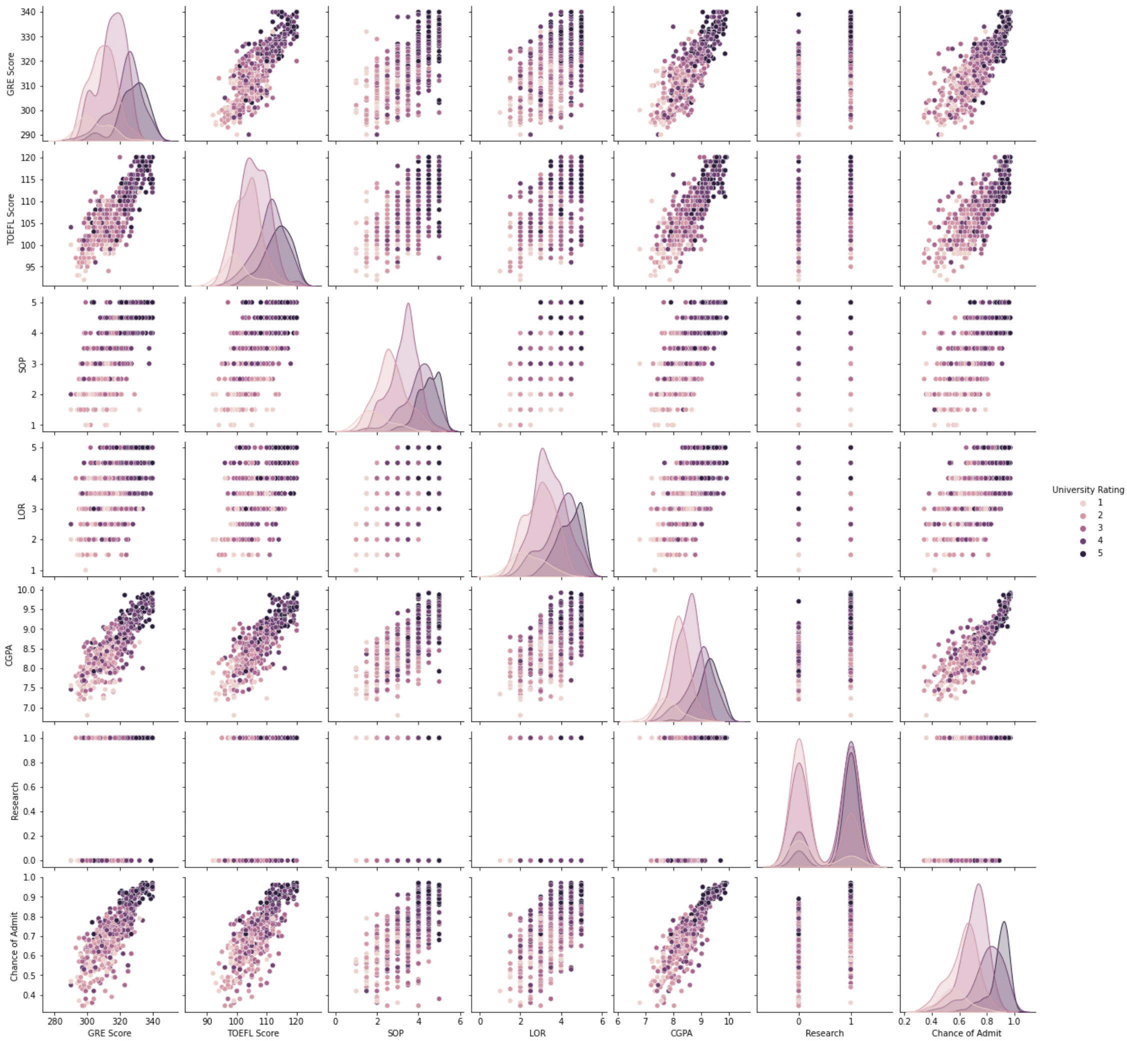


In [67]:

```
sns.pairplot(data=data,hue="University Rating")
```

Out[67]:

<seaborn.axisgrid.PairGrid at 0x28fd7d228e0>



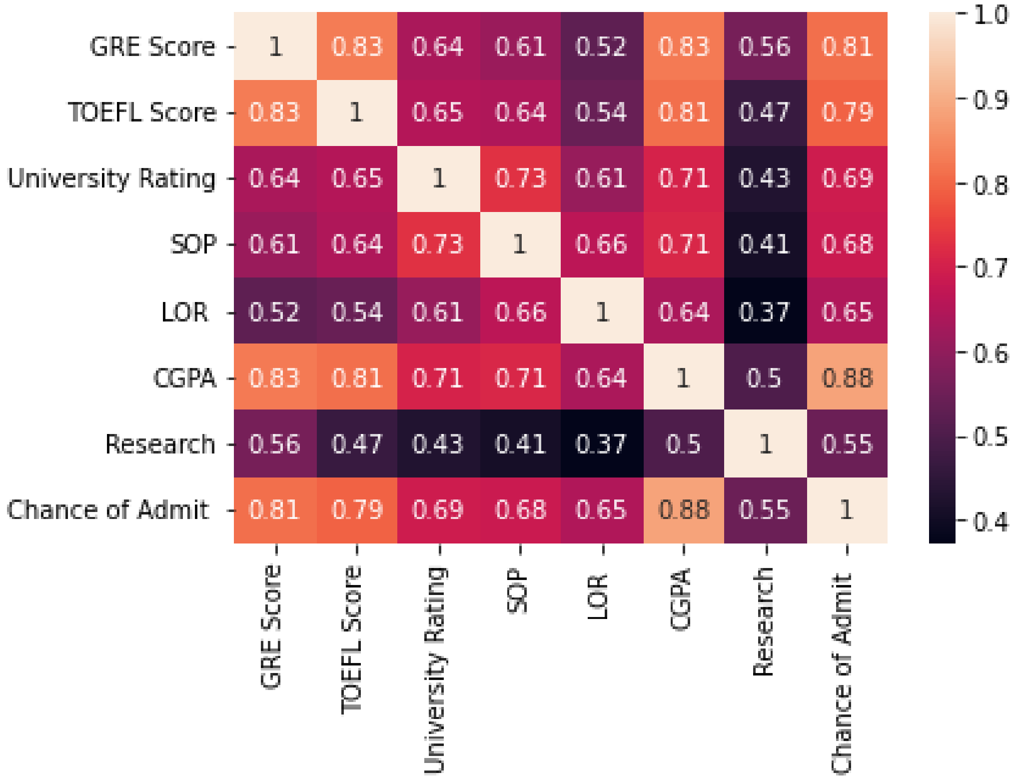
CHECKING CORRELATION

In [68]:

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

Out[68]:

<AxesSubplot:>



We can see that **Chances of Admit** have high correlation with **CGPA** followed by **GRE Score, Toefl Score,Univrsity Rating ,SOP,LOR & Research**

So we need to train a Multivariate Linear Regression Model which can predict the Chances of Admit taking taking into account all of the factors .

TRAINING MULTIVARIATE LINEAR REGRESSION MODEL

In [69]:

```
X=data.drop(columns=["Chance of Admit ","Chances"])
```

In [70]:

```
X
```

Out[70]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	337	118	4	4.5	4.5	9.65	1
1	324	107	4	4.0	4.5	8.87	1
2	316	104	3	3.0	3.5	8.00	1
3	322	110	3	3.5	2.5	8.67	1
4	314	103	2	2.0	3.0	8.21	0
...	...	...	...	...	...	...	...
495	332	108	5	4.5	4.0	9.02	1
496	337	117	5	5.0	5.0	9.87	1
497	330	120	5	4.5	5.0	9.56	1
498	312	103	4	4.0	5.0	8.43	0
499	327	113	4	4.5	4.5	9.04	0

500 rows × 7 columns

In [71]:

```
Y=data["Chance of Admit "]
```

In [72]:

```
Y
```

Out[72]:

```
0      0.92
1      0.76
2      0.72
3      0.80
4      0.65
       ...
495    0.87
496    0.96
497    0.93
498    0.73
499    0.84
Name: Chance of Admit , Length: 500, dtype: float64
```

MODEL PREPROCESSING

In [73]:

```
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge,Lasso
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [74]:

```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_st
```

In [75]:

```
from sklearn.preprocessing import StandardScaler
```

In [76]:

```
x_train_columns=x_train.columns
std_scaler=StandardScaler()
```

USING LINEAR REGRESSION

In [77]:

```
def adj_r2(X, y, r2_score):
    return 1 - ((1-r2_score)*(len(y)-1))/(len(y)-X.shape[1]-1)
```

In [78]:

```
std_scaler_model=make_pipeline(std_scaler,LinearRegression())
std_scaler_model.fit(x_train,y_train)
a=pd.DataFrame([std_scaler_model.score(x_train,y_train),std_scaler_model.s
a.rename(columns = {0:"R2_SCORE"},inplace=True)
a["RMSE"]= [np.sqrt(mean_squared_error(y_train,std_scaler_model.predict(x_
a["Adj_R2"]=[adj_r2(x_train,y_train,std_scaler_model.score(x_train,y_train
a["MAE"]=[mean_absolute_error(y_train,std_scaler_model.predict(x_train)),m
a
```

Out[78]:

	R2_SCORE	RMSE	Adj_R2	MAE
TRAIN_SET	0.821592	0.059750	0.818406	0.042929
TEST_SET	0.821689	0.058622	0.808122	0.040149

In [79]:

```
np.sqrt(mean_squared_error(y_test,std_scaler_model.predict(x_test)))
```

Out[79]:

0.058621972796798844

In [80]:

```
np.sqrt(mean_squared_error(y_train,std_scaler_model.predict(x_train)))
```

Out[80]:

0.05974961869476739

In [81]:

```
model=std_scaler_model.steps[-1][1]
```

In [82]:

```
model.intercept_
```

Out[82]:

0.7209375000000001

In [83]:

```
pd.DataFrame(model.coef_,index=X.columns)
```

Out[83]:

	0
GRE Score	0.020912
TOEFL Score	0.019643
University Rating	0.007022
SOP	0.003066
LOR	0.013512
CGPA	0.070673
Research	0.009887

In [84]:

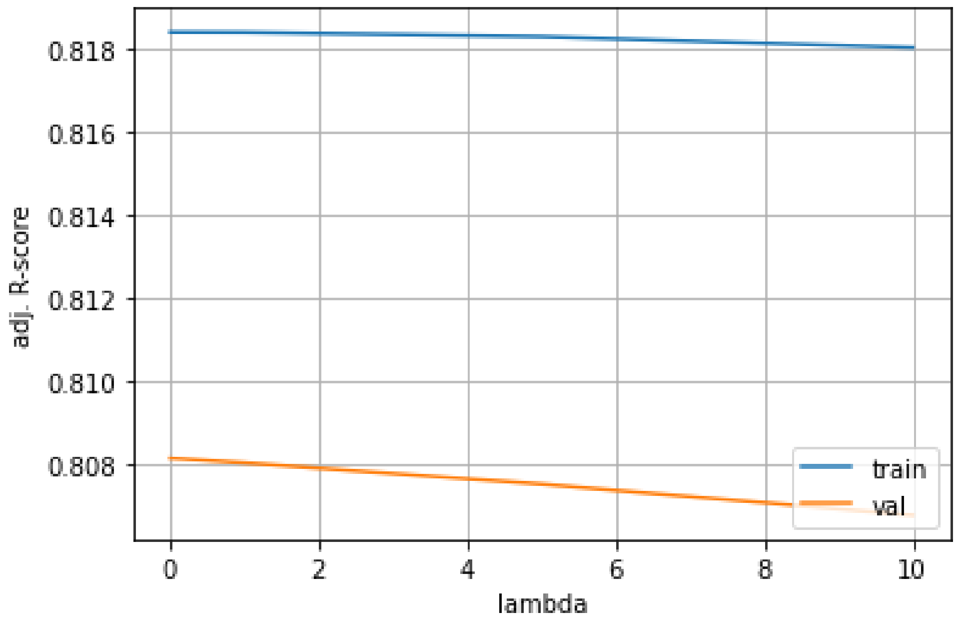
```
train_scores = []
test_scores = []
rate_list = [0.01, 0.1, 1,5, 10]
for rate in rate_list:
    std_scaler_model = make_pipeline(std_scaler, Ridge(alpha=rate))
    std_scaler_model.fit(x_train, y_train)
    train_score = adj_r2(x_train, y_train, std_scaler_model.score(x_train,
    test_score= adj_r2(x_test, y_test, std_scaler_model.score(x_test,y_tes
    train_scores.append(train_score)
    test_scores.append(test_score)
```

In [ ]:

```
```

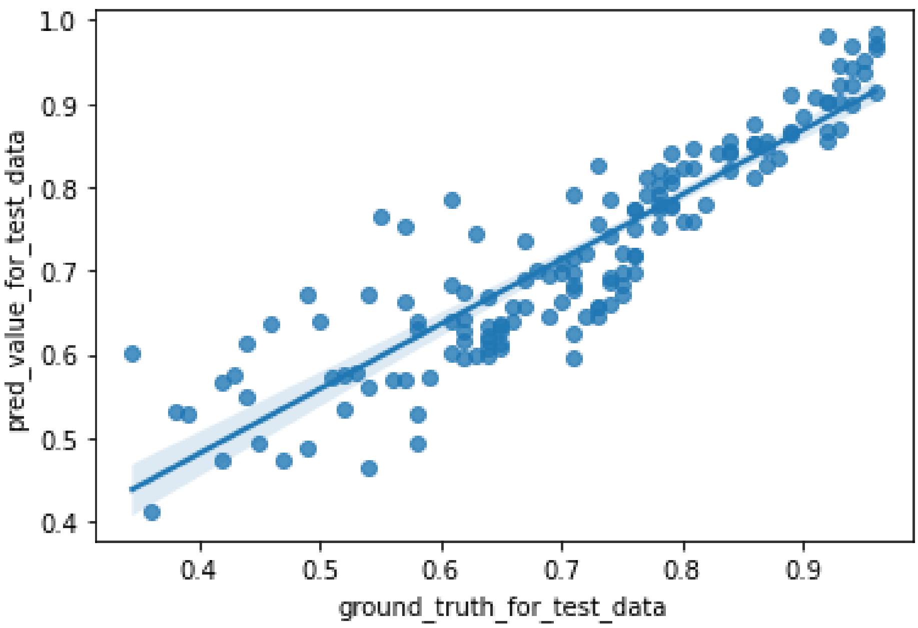
In [85]:

```
plt.figure()
plt.plot(rate_list, train_scores, label="train")
plt.plot(rate_list, test_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("lambda")
plt.ylabel("adj. R-score")
plt.grid()
plt.show()
```



In [162]:

```
sns.regplot(x=y_test,y=std_scaler_model.predict(x_test))
plt.xlabel("ground_truth_for_test_data")
plt.ylabel("pred_value_for_test_data")
plt.show()
```



1. WE CAN SEE THat the regplot for PREDICTED and GROUND TRUTH FRO TEST DATA IS MAJORLY EQUAL AND IS A CLOSE TO 45 DEGREE LINE

USING POLYNOMIAL REGRESSION

In [87]:

```
from sklearn.preprocessing import PolynomialFeatures
```

In [88]:

```
std_scaler_model=make_pipeline(PolynomialFeatures(2),std_scaler,LinearRegr
std_scaler_model.fit(x_train,y_train)
d=pd.DataFrame([std_scaler_model.score(x_train,y_train),std_scaler_model.s
d.rename(columns = {0:"R2_SCORE"},inplace=True)
d["RMSE"]= [np.sqrt(mean_squared_error(y_train,std_scaler_model.predict(x_
d["Adj_R2"]=[adj_r2(x_train,y_train,std_scaler_model.score(x_train,y_train
d["MAE"]=[mean_absolute_error(y_train,std_scaler_model.predict(x_train)),m
d
```

Out[88]:

	R2_SCORE	RMSE	Adj_R2	MAE
TRAIN_SET	0.837354	0.057049	0.834450	0.040574
TEST_SET	0.824500	0.058158	0.811147	0.039546

1. This is a interesting feature , we can see that there might be some non-linearity present in the data and increasing the complexity , or adding some feature will definitely increase the model performance .

USING LASSO L1 REGRESSION

In [89]:

```
std_scaler_model=make_pipeline(std_scaler,Lasso(alpha=0.1))
std_scaler_model.fit(x_train,y_train)
b=pd.DataFrame([std_scaler_model.score(x_train,y_train),std_scaler_model.s
b.rename(columns = {0:"R2_SCORE"},inplace=True)
b["RMSE"]= [np.sqrt(mean_squared_error(y_train,std_scaler_model.predict(x_
b["Adj_R2"]=[adj_r2(x_train,y_train,std_scaler_model.score(x_train,y_train
b["MAE"]=[mean_absolute_error(y_train,std_scaler_model.predict(x_train)),m
b
```

Out[89]:

	R2_SCORE	RMSE	Adj_R2	MAE
TRAIN_SET	0.279254	0.120093	0.266383	0.096795
TEST_SET	0.279989	0.117799	0.225206	0.095813

In [90]:

```
model=std_scaler_model.steps[-1][1]
```

In [91]:

```
model.intercept_
```

Out[91]:

```
0.7209375
```

In [92]:

```
pd.DataFrame(model.coef_,index=X.columns)
```

Out[92]:

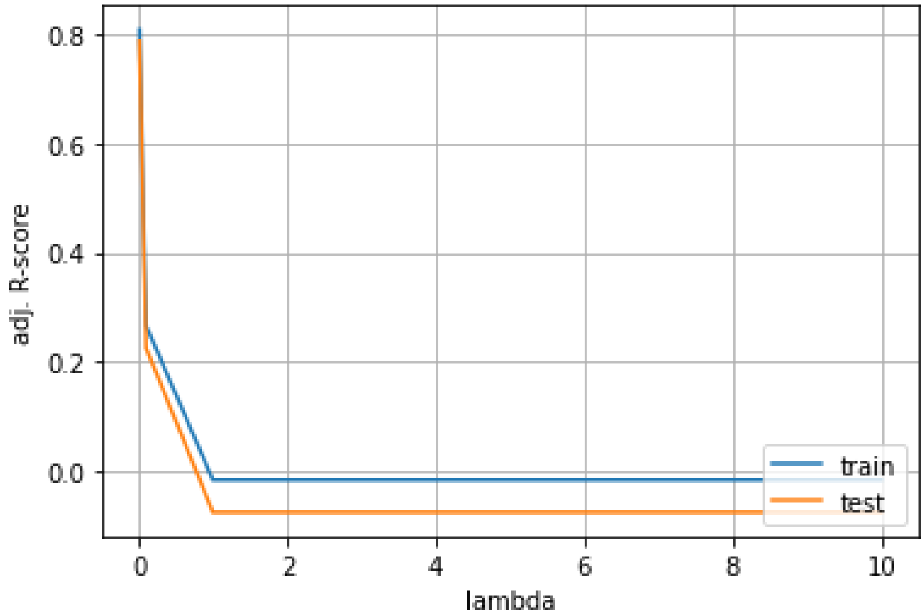
	0
GRE Score	0.000000
TOEFL Score	0.000000
University Rating	0.000000
SOP	0.000000
LOR	0.000000
CGPA	0.024852
Research	0.000000

In [93]:

```
train_scores = []
test_scores = []
rate_list = [0.01, 0.1, 1,5, 10]
for rate in rate_list:
    std_scaler_model = make_pipeline(std_scaler, Lasso(alpha=rate))
    std_scaler_model.fit(x_train, y_train)
    train_score = adj_r2(x_train, y_train, std_scaler_model.score(x_train,
    test_score= adj_r2(x_test, y_test, std_scaler_model.score(x_test,y_tes
    train_scores.append(train_score)
    test_scores.append(test_score)
```

In [94]:

```
plt.figure()
plt.plot(rate_list, train_scores, label="train")
plt.plot(rate_list, test_scores, label="test")
plt.legend(loc='lower right')
plt.xlabel("lambda")
plt.ylabel("adj. R-score")
plt.grid()
plt.show()
```



USING RIDGE-L2 REGRESSION

In [95]:

```
std_scaler_model=make_pipeline(std_scaler, Ridge(alpha=1.0))
std_scaler_model.fit(x_train,y_train)
b=pd.DataFrame([std_scaler_model.score(x_train,y_train),std_scaler_model.s
b.rename(columns = {0:"R2_SCORE"},inplace=True)
b["RMSE"]= [np.sqrt(mean_squared_error(y_train,std_scaler_model.predict(x_
b["Adj_R2"]=[adj_r2(x_train,y_train,std_scaler_model.score(x_train,y_train
b["MAE"]=[mean_absolute_error(y_train,std_scaler_model.predict(x_train)),m
b
```

Out[95]:

	R2_SCORE	RMSE	Adj_R2	MAE
TRAIN_SET	0.821588	0.059750	0.818402	0.042917
TEST_SET	0.821584	0.058639	0.808009	0.040178

In [96]:

```
model=std_scaler_model.steps[-1][1]
```

In [97]:

```
model.intercept_
```

Out[97]:

```
0.7209375000000001
```

In [98]:

```
model.coef_
```

Out[98]:

```
array([0.02111161, 0.01976244, 0.00710444, 0.003217  , 0.013
55454,
       0.07003263, 0.00990384])
```

In [99]:

```
pd.DataFrame(model.coef_,index=X.columns)
```

Out[99]:

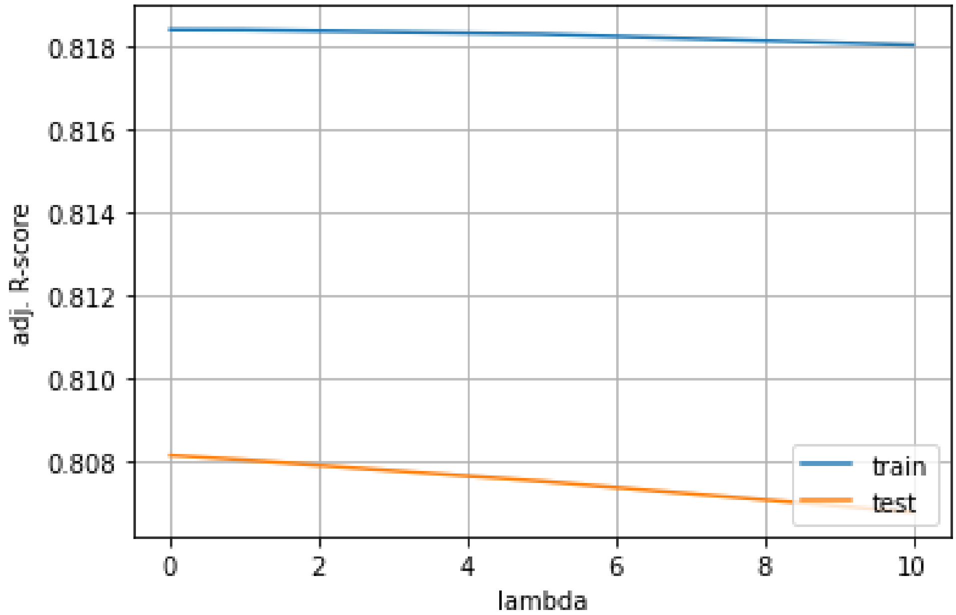
	0
GRE Score	0.021112
TOEFL Score	0.019762
University Rating	0.007104
SOP	0.003217
LOR	0.013555
CGPA	0.070033
Research	0.009904

In [100]:

```
train_scores = []
test_scores = []
rate_list = [0.01, 0.1, 1,5, 10]
for rate in rate_list:
    std_scaler_model = make_pipeline(std_scaler, Ridge(alpha=rate))
    std_scaler_model.fit(x_train, y_train)
    train_score = adj_r2(x_train, y_train, std_scaler_model.score(x_train,
    test_score= adj_r2(x_test, y_test, std_scaler_model.score(x_test,y_tes
    train_scores.append(train_score)
    test_scores.append(test_score)
```

In [101]:

```
plt.figure()
plt.plot(rate_list, train_scores, label="train")
plt.plot(rate_list, test_scores, label="test")
plt.legend(loc='lower right')
plt.xlabel("lambda")
plt.ylabel("adj. R-score")
plt.grid()
plt.show()
```



OBSERVATIONS

1. ALTHOUGH OUR LINEAR REGRESSION MODEL PERFORMED GOOD , BUT ADDDING SOME COMPLEXITY WILL INCREASE THE PERFORMANCE FURTHER.
2. THE MODEL WAS NOT OVERFITTING ,BUT WE CAN SEE IN LASSO ,I IT PENALIZED THE MODEL AND IT STARTED PERFORMING WORSE, THUS UNDERFITTING.
3. RIDGE WITH REGULARISATION RATE=O.1 HAS ALMOST SAME PERFORMANCE AS LINEAR REGRESSION MODEL .
4. AS WE CAN THE VARIOUS STATISTICS ALSO REVEAL HOW THE MODELS PERFORMED.

Testing the assumptions of the linear regression model

1. CHECKING LINEARITY

In [102]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
X = X
Y = Y
X_1 = np.array(X)
Y_1 = np.array(Y)
model.fit(X_1, Y_1)
Y_hat = model.predict(X_1)
```

In [103]:

```
model.score(X_1,Y_1)
```

Out[103]:

```
0.8221241806410019
```

2.. CHECKING IF ERRORS ARE NORMALLY DISTRIBUTED

In [104]:

```
errors = Y - Y_hat
```

In [105]:

```
pd.DataFrame(errors).describe()
```

Out[105]:

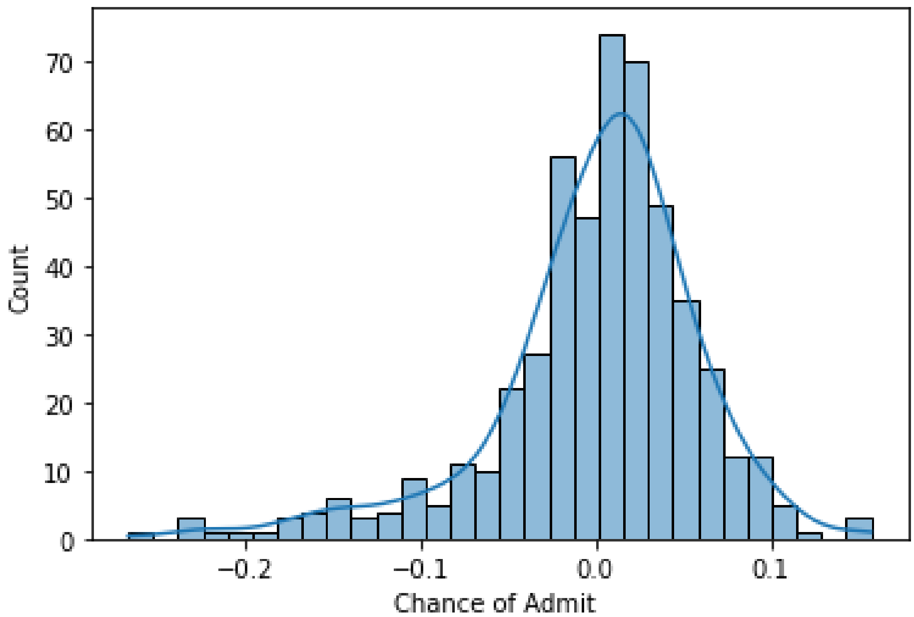
	Chance of Admit
count	5.000000e+02
mean	-2.687850e-16
std	5.950371e-02
min	-2.666883e-01
25%	-2.340110e-02
50%	9.201410e-03
75%	3.368162e-02
max	1.567263e-01

In [106]:

```
sns.histplot(x=errors,kde=True)
```

Out[106]:

```
<AxesSubplot:xlabel='Chance of Admit ', ylabel='Count'>
```

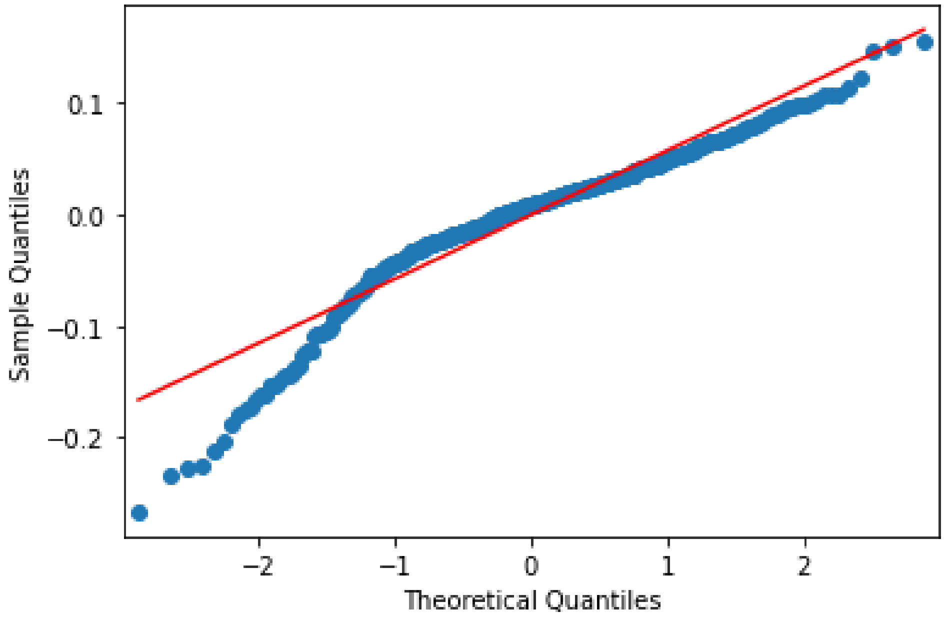


In [107]:

```
from statsmodels.graphics.gofplots import qqplot
```

In [108]:

```
qqplot(errors, line="r")
plt.show()
```

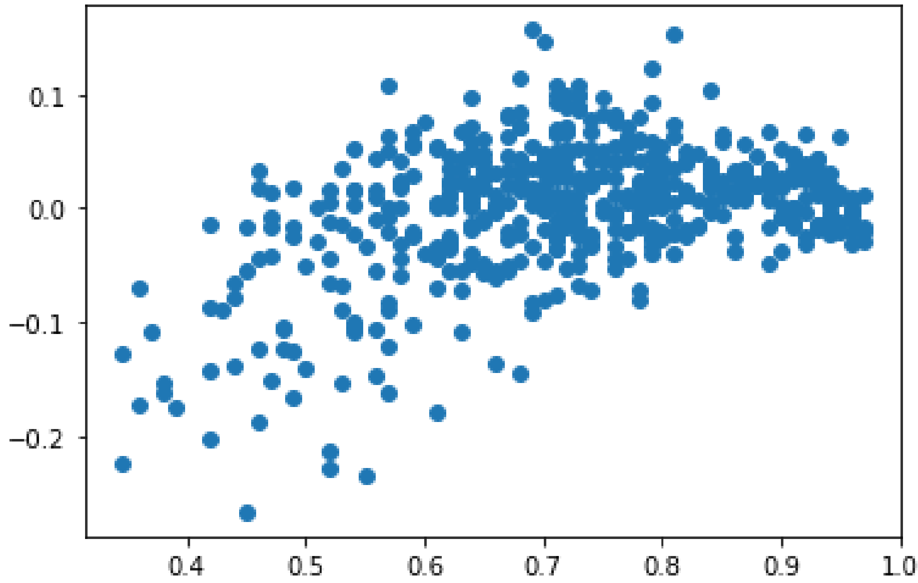


1. Errors are not normally distributed , it means we need to some seperate analysis on the outliers .

3. CHECKING HETEROSKADASTICITY

In [110]:

```
import matplotlib.pyplot as plt
plt.scatter(Y, errors)
plt.show()
```



4. CHECKING AUTOCORRELATION

In [112]:

```
from statsmodels.stats.stattools import durbin_watson
durbin_watson(errors)
```

Out[112]:

```
0.7957978111842413
```

NOTE:- As the value is close to 0 it indicates there is positive autocorrelation in errors

5. CHECKING MUTLICOLLINEARITY

In [113]:

```
import statsmodels.api as sm
X_sm = sm.add_constant(X)
```

In [114]:

```
sm_model = sm.OLS(Y, X_sm).fit()
```

In [115]:

```
print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
==================
Dep. Variable:      Chance of Admit    R-squared:
0.822
Model:                          OLS    Adj. R-squared:
0.820
Method:               Least Squares    F-statistic:
324.9
Date:              Tue, 04 Jul 2023    Prob (F-statistic):
6.03e-180
Time:                      22:12:15    Log-Likelihood:
701.89
No. Observations:               500    AIC:
-1388.
Df Residuals:                   492    BIC:
-1354.
Df Model:                         7
Covariance Type:          nonrobust
==============================================================================
=========================
                       coef    std err          t      P>|t
|      [0.025      0.975]
------------------------------------------------------------------------------
-------------------------
const               -1.2747      0.104    -12.234      0.00
0      -1.479      -1.070
GRE Score            0.0019      0.001      3.700      0.00
0       0.001       0.003
TOEFL Score          0.0028      0.001      3.182      0.00
2       0.001       0.004
University Rating    0.0059      0.004      1.562      0.11
9      -0.002       0.013
SOP                  0.0016      0.005      0.360      0.71
9      -0.007       0.011
LOR                  0.0168      0.004      4.073      0.00
0       0.009       0.025
CGPA                 0.1184      0.010     12.210      0.00
0       0.099       0.137
Research             0.0243      0.007      3.681      0.00
0       0.011       0.037
==============================================================================
==================
Omnibus:                    112.106    Durbin-Watson:
0.796
Prob(Omnibus):                0.000    Jarque-Bera (JB):
259.157
Skew:                        -1.155    Prob(JB):
5.31e-57
Kurtosis:                     5.665    Cond. No.
```

```
1.30e+04
==============================================================
==================

Notes:
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.
[2] The condition number is large, 1.3e+04. This might indic
ate that there are
strong multicollinearity or other numerical problems.
```

In [116]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [117]:

```
vif = pd.DataFrame()
X_t = X
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[117]:

	Features	VIF
0	GRE Score	1308.06
1	TOEFL Score	1215.95
5	CGPA	950.82
3	SOP	35.27
4	LOR	30.91
2	University Rating	20.93
6	Research	2.87

In [118]:

```
X_new = X.drop(columns=['GRE Score'])
```

In [119]:

```
X2_sm = sm.add_constant(X_new)

sm_model = sm.OLS(Y, X2_sm).fit()
```

In [120]:

```
print(sm_model.summary())
```

```
                            OLS Regression Results                            
==============================================================================
==================
Dep. Variable:       Chance of Admit    R-squared:                       
0.817
Model:                           OLS   Adj. R-squared:                  
0.815
Method:                Least Squares   F-statistic:                     
367.3
Date:               Tue, 04 Jul 2023   Prob (F-statistic):          
2.55e-178
Time:                       22:12:17   Log-Likelihood:                 
695.03
No. Observations:                500   AIC:                            
-1376.
Df Residuals:                    493   BIC:                            
-1347.
Df Model:                          6                                         
Covariance Type:           nonrobust                                         
==============================================================================
=========================
                        coef    std err          t      P>|t
|      [0.025      0.975]
------------------------------------------------------------------------------
-------------------------
const                -0.9692      0.064    -15.058      0.00
0      -1.096      -0.843
TOEFL Score           0.0043      0.001      5.429      0.00
0       0.003       0.006
University Rating     0.0066      0.004      1.723      0.08
5      -0.001       0.014
SOP                   0.0011      0.005      0.237      0.81
3      -0.008       0.010
LOR                   0.0160      0.004      3.829      0.00
0       0.008       0.024
CGPA                  0.1326      0.009     14.706      0.00
0       0.115       0.150
Research              0.0313      0.006      4.883      0.00
0       0.019       0.044
==============================================================================
==================
Omnibus:                     100.578   Durbin-Watson:                   
0.809
Prob(Omnibus):                 0.000   Jarque-Bera (JB):              
205.277
Skew:                         -1.094   Prob(JB):                     
2.66e-45
Kurtosis:                      5.251   Cond. No.                     
2.56e+03
==============================================================================
```

```
==================

Notes:
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.
[2] The condition number is large, 2.56e+03. This might indi
cate that there are
strong multicollinearity or other numerical problems.
```

In [121]:

```
vif = pd.DataFrame()
X_t = X_new
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_new.values, i) for i in range(X_
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[121]:

	Features	VIF
4	CGPA	728.78
0	TOEFL Score	639.74
2	SOP	33.73
3	LOR	30.63
1	University Rating	19.88
5	Research	2.86

In [122]:

```
X_new1 = X_new.drop(columns=['CGPA'])
```

In [123]:

```
X_new1
```

Out[123]:

	TOEFL Score	University Rating	SOP	LOR	Research
0	118	4	4.5	4.5	1
1	107	4	4.0	4.5	1
2	104	3	3.0	3.5	1
3	110	3	3.5	2.5	1
4	103	2	2.0	3.0	0
...	...	...	...	...	...
495	108	5	4.5	4.0	1
496	117	5	5.0	5.0	1
497	120	5	4.5	5.0	1
498	103	4	4.0	5.0	0
499	113	4	4.5	4.5	0

500 rows × 5 columns

In [124]:

```
X2_sm = sm.add_constant(X_new1)

sm_model2 = sm.OLS(Y, X2_sm).fit()
```

In [125]:

```
print(sm_model2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Chance of Admit   R-squared:                       0.737
Model:                           OLS   Adj. R-squared:                  0.734
Method:                Least Squares   F-statistic:                     276.8
Date:               Tue, 04 Jul 2023   Prob (F-statistic):          1.03e-140
Time:                       22:12:20   Log-Likelihood:                 604.09
No. Observations:                500   AIC:                            -1196.
Df Residuals:                    494   BIC:                            -1171.
Df Model:                          5
Covariance Type:           nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -0.6603      0.073     -9.058      0.000      -0.804      -0.517
TOEFL Score           0.0108      0.001     14.000      0.000       0.009       0.012
University Rating     0.0166      0.005      3.665      0.000       0.008       0.026
SOP                   0.0136      0.005      2.504      0.013       0.003       0.024
LOR                   0.0286      0.005      5.824      0.000       0.019       0.038
Research              0.0460      0.008      6.067      0.000       0.031       0.061
==============================================================================
Omnibus:                      69.468   Durbin-Watson:                   0.865
Prob(Omnibus):                 0.000   Jarque-Bera (JB):              104.049
Skew:                         -0.916   Prob(JB):                     2.55e-23
Kurtosis:                      4.279   Cond. No.                     2.41e+03
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.
[2] The condition number is large, 2.41e+03. This might indi
cate that there are
strong multicollinearity or other numerical problems.
```

there is a considerbale drop

In [126]:

```
X_new2=X_new
```

In [127]:

```
X_new2
```

Out[127]:

	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	118	4	4.5	4.5	9.65	1
1	107	4	4.0	4.5	8.87	1
2	104	3	3.0	3.5	8.00	1
3	110	3	3.5	2.5	8.67	1
4	103	2	2.0	3.0	8.21	0
...	...	...	...	...	...	...
495	108	5	4.5	4.0	9.02	1
496	117	5	5.0	5.0	9.87	1
497	120	5	4.5	5.0	9.56	1
498	103	4	4.0	5.0	8.43	0
499	113	4	4.5	4.5	9.04	0

500 rows × 6 columns

In [128]:

```
vif = pd.DataFrame()
X_t = X_new2
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_new2.values, i) for i in range(X_
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[128]:

	Features	VIF
4	CGPA	728.78
0	TOEFL Score	639.74
2	SOP	33.73
3	LOR	30.63
1	University Rating	19.88
5	Research	2.86

In [129]:

```
X_new3 = X_new2.drop(columns=['TOEFL Score'])
```

In [130]:

```
X_new3
```

Out[130]:

	University Rating	SOP	LOR	CGPA	Research
0	4	4.5	4.5	9.65	1
1	4	4.0	4.5	8.87	1
2	3	3.0	3.5	8.00	1
3	3	3.5	2.5	8.67	1
4	2	2.0	3.0	8.21	0
...	...	...	...	...	...
495	5	4.5	4.0	9.02	1
496	5	5.0	5.0	9.87	1
497	5	4.5	5.0	9.56	1
498	4	4.0	5.0	8.43	0
499	4	4.5	4.5	9.04	0

500 rows × 5 columns

In [131]:

```
X2_sm = sm.add_constant(X_new3)

sm_model3 = sm.OLS(Y, X2_sm).fit()
```

In [132]:

```
print(sm_model3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Chance of Admit   R-squared:                       0.806
Model:                           OLS   Adj. R-squared:                  0.804
Method:                Least Squares   F-statistic:                     411.1
Date:               Tue, 04 Jul 2023   Prob (F-statistic):          1.91e-173
Time:                       22:12:26   Log-Likelihood:                 680.51
No. Observations:                500   AIC:                            -1349.
Df Residuals:                    494   BIC:                            -1324.
Df Model:                          5
Covariance Type:           nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -0.7670      0.054    -14.206      0.000      -0.873      -0.661
University Rating     0.0092      0.004      2.350      0.019       0.002       0.017
SOP                   0.0034      0.005      0.718      0.473      -0.006       0.013
LOR                   0.0153      0.004      3.557      0.000       0.007       0.024
CGPA                  0.1604      0.008     21.040      0.000       0.145       0.175
Research              0.0347      0.007      5.293      0.000       0.022       0.048
==============================================================================
Omnibus:                      85.826   Durbin-Watson:                   0.855
Prob(Omnibus):                 0.000   Jarque-Bera (JB):              164.191
Skew:                         -0.971   Prob(JB):                     2.22e-36
Kurtosis:                      5.028   Cond. No.                         205.
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.
```

In [133]:

```
vif = pd.DataFrame()
X_t = X_new3
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_new3.values, i) for i in range(X
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[133]:

	Features	VIF
1	SOP	33.63
2	LOR	30.36
3	CGPA	25.10
0	University Rating	19.78
4	Research	2.84

In [134]:

```
X_new4=X_new3.drop(columns=['SOP'])
```

In [135]:

```
X_new4
```

Out[135]:

	University Rating	LOR	CGPA	Research
0	4	4.5	9.65	1
1	4	4.5	8.87	1
2	3	3.5	8.00	1
3	3	2.5	8.67	1
4	2	3.0	8.21	0
...	...	...	...	...
495	5	4.0	9.02	1
496	5	5.0	9.87	1
497	5	5.0	9.56	1
498	4	5.0	8.43	0
499	4	4.5	9.04	0

500 rows × 4 columns

In [136]:

```
X2_sm = sm.add_constant(X_new4)

sm_model4 = sm.OLS(Y, X2_sm).fit()
```

In [137]:

```
print(sm_model4.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Chance of Admit    R-squared:                       0.806
Model:                           OLS   Adj. R-squared:                   0.804
Method:                Least Squares   F-statistic:                      514.3
Date:               Tue, 04 Jul 2023   Prob (F-statistic):          1.02e-174
Time:                       22:12:29   Log-Likelihood:                 680.25
No. Observations:                500   AIC:                            -1350.
Df Residuals:                    495   BIC:                            -1329.
Df Model:                          4
Covariance Type:           nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -0.7758      0.053    -14.755      0.000      -0.879      -0.672
University Rating     0.0103      0.004      2.843      0.005       0.003       0.017
LOR                   0.0162      0.004      3.954      0.000       0.008       0.024
CGPA                  0.1620      0.007     22.204      0.000       0.148       0.176
Research              0.0348      0.007      5.311      0.000       0.022       0.048
==============================================================================
Omnibus:                      83.510   Durbin-Watson:                   0.863
Prob(Omnibus):                 0.000   Jarque-Bera (JB):              156.914
Skew:                         -0.954   Prob(JB):                     8.44e-35
Kurtosis:                      4.972   Cond. No.                         188.
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.
```

In [138]:

```
vif = pd.DataFrame()
X_t = X_new4
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_new4.values, i) for i in range(X_
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[138]:

	Features	VIF
1	LOR	26.92
2	CGPA	22.37
0	University Rating	15.14
3	Research	2.82

In [139]:

```
X_new5=X_new4.drop(columns=['LOR '])
```

In [140]:

```
X_new5
```

Out[140]:

	University Rating	CGPA	Research
0	4	9.65	1
1	4	8.87	1
2	3	8.00	1
3	3	8.67	1
4	2	8.21	0
...	...	...	...
495	5	9.02	1
496	5	9.87	1
497	5	9.56	1
498	4	8.43	0
499	4	9.04	0

500 rows × 3 columns

In [141]:

```
X2_sm = sm.add_constant(X_new5)

sm_model5 = sm.OLS(Y, X2_sm).fit()
```

In [142]:

```
print(sm_model5.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Chance of Admit    R-squared:
0.800
Model:                           OLS   Adj. R-squared:
0.799
Method:                Least Squares   F-statistic:
661.0
Date:               Tue, 04 Jul 2023   Prob (F-statistic):
8.01e-173
Time:                       22:12:32   Log-Likelihood:
672.48
No. Observations:                500   AIC:
-1337.
Df Residuals:                    496   BIC:
-1320.
Df Model:                          3
Covariance Type:           nonrobust
==============================================================================
                        coef    std err          t      P>|t
|      [0.025      0.975]
------------------------------------------------------------------------------
const                -0.8174      0.052    -15.638      0.00
0      -0.920      -0.715
University Rating     0.0144      0.004      4.080      0.00
0       0.007       0.021
CGPA                  0.1719      0.007     24.710      0.00
0       0.158       0.186
Research              0.0360      0.007      5.423      0.00
0       0.023       0.049
==============================================================================
Omnibus:                      85.826   Durbin-Watson:
0.899
Prob(Omnibus):                 0.000   Jarque-Bera (JB):
157.567
Skew:                         -0.989   Prob(JB):
6.09e-35
Kurtosis:                      4.910   Cond. No.
172.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.
```

In [143]:

```
vif = pd.DataFrame()
X_t = X_new5
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_new5.values, i) for i in range(X_
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[143]:

	Features	VIF
0	University Rating	12.50
1	CGPA	11.04
2	Research	2.78

In [144]:

```
X_new6=X_new5.drop(columns=['University Rating'])
```

In [145]:

```
X2_sm = sm.add_constant(X_new6)

sm_model6 = sm.OLS(Y, X2_sm).fit()
```

In [146]:

```
print(sm_model6.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Chance of Admit   R-squared:                       0.793
Model:                           OLS   Adj. R-squared:                  0.792
Method:                Least Squares   F-statistic:                     953.1
Date:               Tue, 04 Jul 2023   Prob (F-statistic):          8.21e-171
Time:                       22:12:36   Log-Likelihood:                 664.22
No. Observations:                500   AIC:                            -1322.
Df Residuals:                    497   BIC:                            -1310.
Df Model:                          2
Covariance Type:           nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.9273      0.045    -20.383      0.000      -1.017      -0.838
CGPA           0.1897      0.005     34.497      0.000       0.179       0.201
Research       0.0392      0.007      5.863      0.000       0.026       0.052
==============================================================================
Omnibus:                      77.758   Durbin-Watson:                   0.896
Prob(Omnibus):                 0.000   Jarque-Bera (JB):              138.462
Skew:                         -0.918   Prob(JB):                     8.58e-31
Kurtosis:                      4.809   Cond. No.                         139.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [147]:

```
vif = pd.DataFrame()
X_t = X_new6
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_new6.values, i) for i in range(X_
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[147]:

	Features	VIF
0	CGPA	2.46
1	Research	2.46

1. SO NOW WE HAVE A SIMPLE MODEL WHICH CAN PERFORM WITH ADJUSTED R_2 OF 0.79 I.E CAN PREDICT
   - 79% TIMES THE RIGHT VALUE
   - AND WHICH HAS NO MULTICOLLINEARITY
2. NOW ASSUMING TO GO FOR A SIMPLE MODEL :-
   - TWO FEATURES **CGPA** AND **RESEARCH** CAN PREDICT **79%** TIMES THE RIGHT VALUES

NOW RE-TRAINING THE MODEL JUST WITH THESE TWO FEATURES:-

In [148]:

```
X_new6=X_t
```

In [149]:

```
X_new6
```

Out[149]:

	CGPA	Research
0	9.65	1
1	8.87	1
2	8.00	1
3	8.67	1
4	8.21	0
...	...	...
495	9.02	1
496	9.87	1
497	9.56	1
498	8.43	0
499	9.04	0

500 rows × 2 columns

now again train the model with these only two features

In [158]:

```
X =X_new6
sc = StandardScaler()

x_train, x_test, y_train, y_test = train_test_split(X_new6,Y, test_size=0.
```

In [159]:

```
std_scaler_model=make_pipeline(std_scaler,LinearRegression())
std_scaler_model.fit(x_train,y_train)
a=pd.DataFrame([std_scaler_model.score(x_train,y_train),std_scaler_model.s
a.rename(columns = {0:"R2_SCORE"},inplace=True)
a["RMSE"]= [np.sqrt(mean_squared_error(y_train,std_scaler_model.predict(x_
a["Adj_R2"]=[adj_r2(x_train,y_train,std_scaler_model.score(x_train,y_train
a["MAE"]=[mean_absolute_error(y_train,std_scaler_model.predict(x_train)),m
a
```

Out[159]:

	R2_SCORE	RMSE	Adj_R2	MAE
TRAIN_SET	0.783648	0.063695	0.782402	0.046782
TEST_SET	0.808849	0.065392	0.806248	0.045254

1. WE CAN SEE TAHT TESTING PERFORMANCE IS ALMOST SAEME AS BEFORE AND IT TELLS US THAT THE MODEL CAN PERFORM EQUALLY WELL WITH JUST TWO FEATURES
2. WELL SINCE TRAINING PERFORMANCE IS LESS THAN TESTING PERFORMANCE , IT PROBABLY MEANS THAT THE DATA IS BIAS , AND WE NEED TO DO K CROSS VALIDATION

INSIGHTS

1. THE BASELINE MODEL WHICH WE TRAINED WAS NOT OVERFITTING AND WAS INDEED HAVING GOOD PERFORMANCE.
2. THERE IS MULTICOLLINEARITY PRESENT IN THE DATA
3. ERRORS ARE NOT NORMALLY DISTRIBUTED
4. THERE IS HOMOSKADASTICITY, I.E NO POSITIVE CORRELATION BW Y AND ERRORS.
5. DATA IS A GOOD LINEAR MODE AS EXPLAINED BY R2_SCORE.
6. ALTHOUGH OUR PERFORMNACE IMPROVED WHEN TRAINING WITH POLYNOMIAL FEATURES, I.E THERE IS SCOPE FOR IMPROVEMENT BY BRINGING BETTER FEATURES.
7. ALTHOUGH THERE IUS TRONG CORRELATION BW SOME FEATURES WITH TARGET VARIABLE , BUT WE CAN THAT ONLY TWO FEATURES ARE ENOUGH TO PREDICT THE VALUE 80% OF TIMES
8. ALSO WE CAN SEE BY L1 REGULARIZATION ,THAT WEIGHTS WERE ASSIGNED ONLY TO CGPA.

9. OUR EDA MADE IT CLEAR THAT CGPA IS THE MOST IMPORTANT FACTOR
10. RESEARCH PLAYS A VITAL ROLE

RECOMMENDATIONS

1. WE CAN IMPROVE THE MODEL BY INCORPORATING SOME COMPLEX FEATURES WHICH CAN BE A MERGER OF SOME FEATURES .
2. ALSO IN REAL WORLD , IF DATA COMES TO US WE CAN PERFORM BETTER WITH MORE DATAPOINTS .
3. SOME ERRORS WHICH WERE OULTIERS SHOULD BE STUDIED IN DEPTH ./ .
4. IN BUSINESS TERMS , IF OUR MODEL IMPROVES, WE CAN ACQUIRE AND RETAIN MANY STUDENTS AND THER WILL BE EXPONENTIAL GROWTH. .
5. ALSO SINCE IT IS A VERY FAST GROWING MARKET IS **35$ BILLION INDUSTRY IN INDIA** .
6. JAMBOREE IS ALREADY A WELL KNOWN COMPANY , BY IMPROVING MODEL PERFORMANCE, BRAND VALUE WILL INCREASE. .
7. WE COULD ADD INTERNSHIPS DONE AS A FETAURE, ALSO NGO EXPERIENCE AND ANY CONTRIBTUION TO SOCIAL CAUSES, ENVIRONMENTAL , AS IT SHOWS HOW RESPONSIBLE A CITIZEN IS AND WHAT IMPACT WILL IT CREATE , WHICH MIGHT BE A BETTER INDICATOR THAN SOP AS ACTION IS BETTER THAN INTENTION.

In [ ]: