Jamboree Admission Probability Factor Mini Project

Jamboree has helped thousands of students to make it to top colleges abroad. Using GMAT, GRE or SAT scores, their unique problem-solving methods ensure maximum scores with minimum effort. They want to analyse the probability of getting into the IVY league college based on the following factors.

Following are the attributes of the dataset.

1. S.No.
2. GRE Score: GRE score out of 340.
3. TOEFL Score: TOEFL score out of 120.
4. University Rating
5. SOP: Statement of Purpose out of 5
6. LOR: Letter of Recommendation out of 5
7. CGPA: GPA out of 10
8. Research: If students have taken part in research.
9. Chance of Admission: How eligible is a student for admission.

Problem Statement:

Chances of getting admission into IVY league college based on muliple factors such as entrance test scores, GPA, etc.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
```

```
df=pd.read_csv("jamboree_admission.csv")
df=df.drop('Serial No.',axis=1)
df.head(10)
```

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

```
df.shape, df.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
((500, 8), None)
```

```
print(df["GRE Score"].value_counts(),df["TOEFL Score"].value_counts(),df["University Rating"].value_counts(),df["SOP"].value_counts(),df["LO
```

```
GRE Score
312    24
324    23
316    18
321    17
322    17
327    17
311    16
320    16
314    16
317    15
325    15
315    13
308    13
323    13
326    12
319    12
313    12
304    12
300    12
318    12
305    11
301    11
310    11
307    10
329    10
299    10
298    10
331     9
340     9
328     9
309     9
334     8
332     8
330     8
306     7
302     7
297     6
296     5
295     5
336     5
303     5
338     4
335     4
333     4
339     3
337     2
290     2
294     2
293     1
Name: count, dtype: int64 TOEFL Score
110    44
105    37
104    29
107    28
106    28
112    28
103    25
```

```
print(df["GRE Score"].nunique(),df["TOEFL Score"].nunique(),df["University Rating"].nunique(),df["SOP"].nunique(),df["LOR "].nunique(),df["C
```

```
49 29 5 9 9 184 2 61
```

```
df.describe()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

COMMENT ON THE RANGE OF ATTRIBUTES

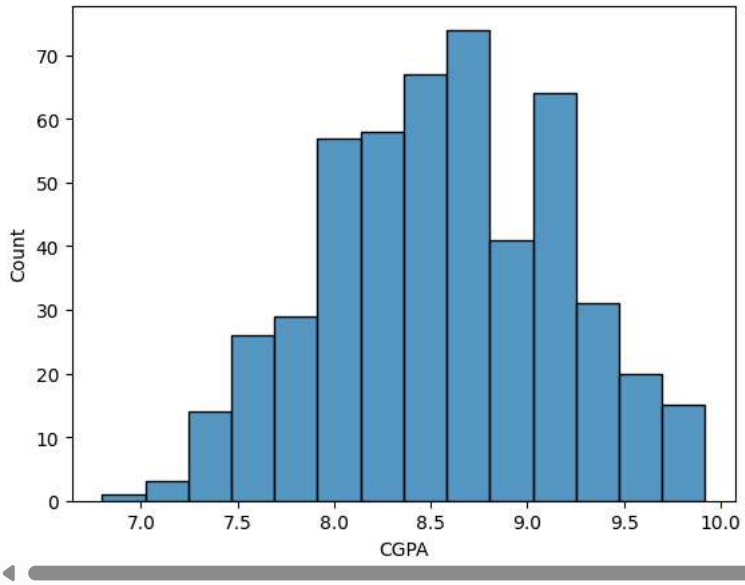
1. There are 500 rows and 9 columns.
2. Column names are 'Serial No.','GRE Score','TOEFL Score','University Rating','SOP','LOR','CGPA','Research','Chance of Admit'.
3. 'Serial No.','GRE Score','TOEFL Score','University Rating','Research' are all integers.
4. 'SOP','LOR','CGPA','Chance of Admit' are all floating numbers.
5. Statistics of the Chance to Admit columns:
    - Mean : 0.72174
    - Standard deviation : 0.14114
    - Minimum value : 0.34
    - Maximum value : 0.97
6. There are 500 unique entries for checking for admission stats.
7. There are only two values for research: 0 or 1, where 0 means research not done and 1 means research completed.
8. Maximum chances of getting into IVY League college is 0.97 while minimum is 0.34.

UNIVARIATE AND BIVARIATE ANALYSIS

For continous variables : DISTPLOT,COUNTPLOT,HISTOGRAM for univariate analysis

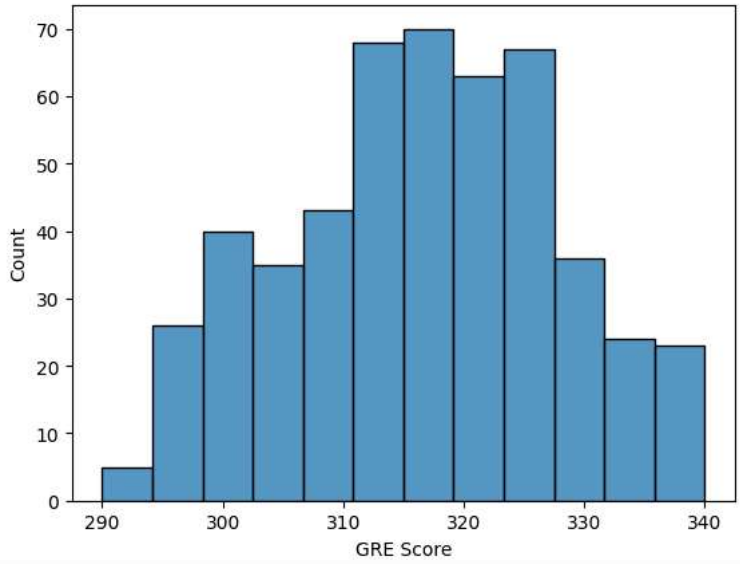
```
sns.histplot(df["CGPA"])
```

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



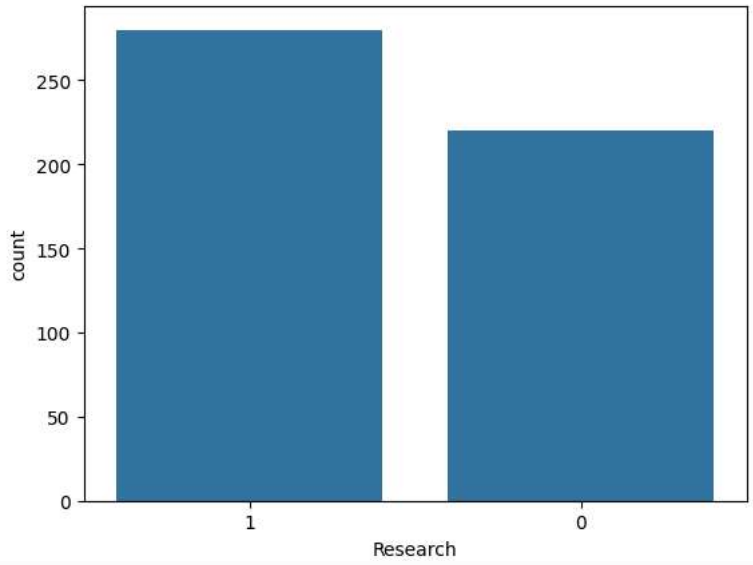

```
sns.histplot(df["GRE Score"])
```

```
<Axes: xlabel='GRE Score', ylabel='Count'>
```



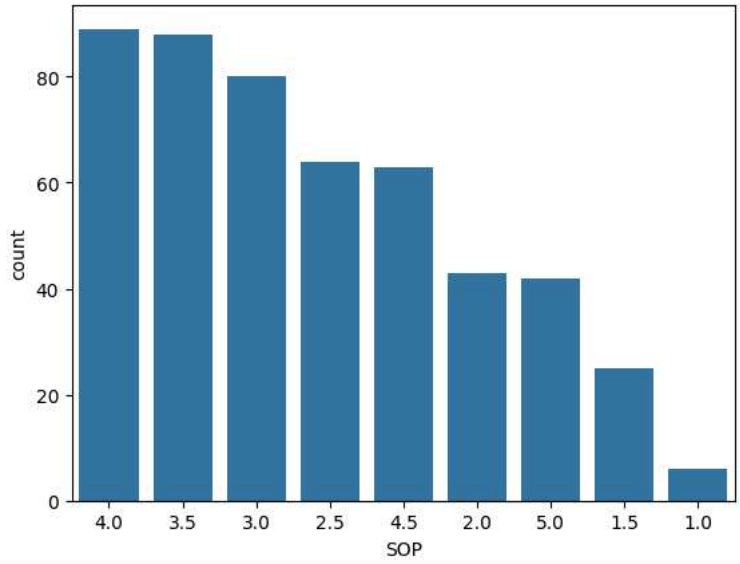

```
sns.countplot(data = df,x = "Research",order = df["Research"].value_counts().index)
```

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



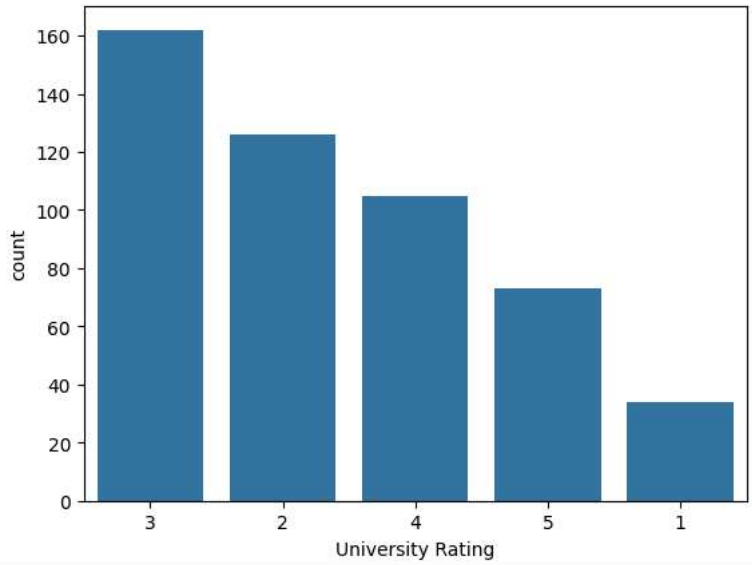

```
sns.countplot(data = df,x = "SOP",order = df["SOP"].value_counts().index)
```

```
<Axes: xlabel='SOP', ylabel='count'>
```



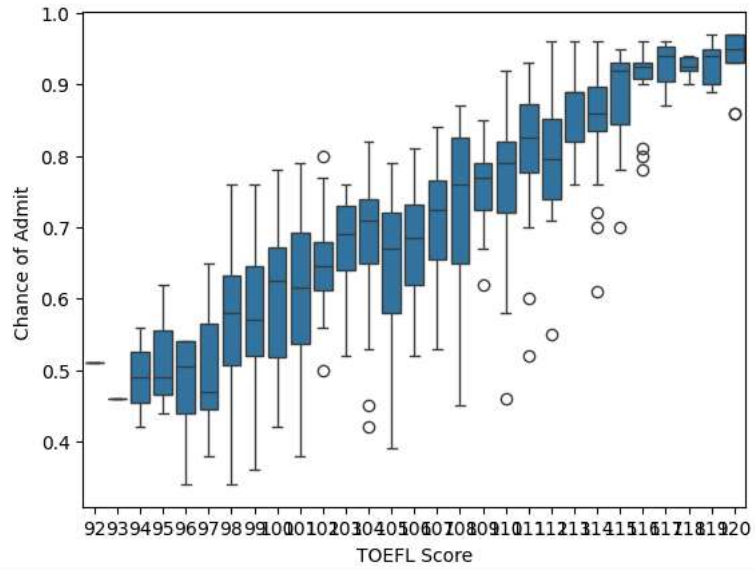

```
sns.countplot(data = df,x = "University Rating",order = df["University Rating"].value_counts().index)
```

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



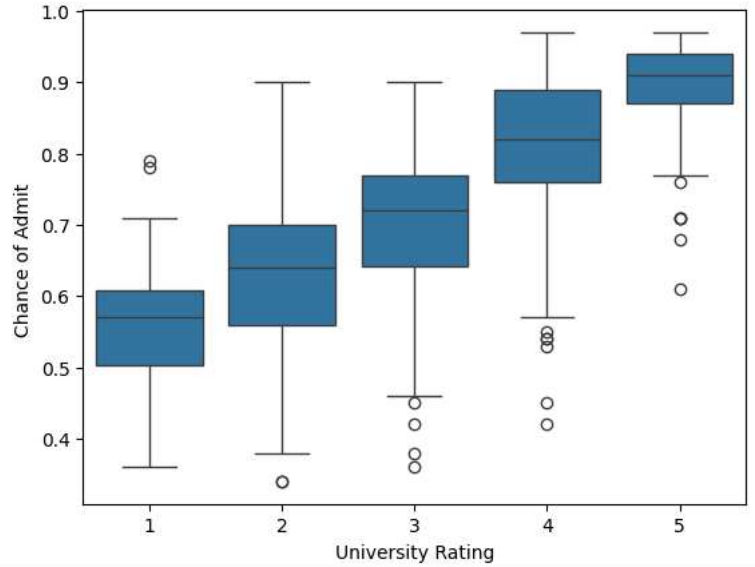

```
sns.boxplot(data = df,x = "TOEFL Score", y = "Chance of Admit ")
```

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



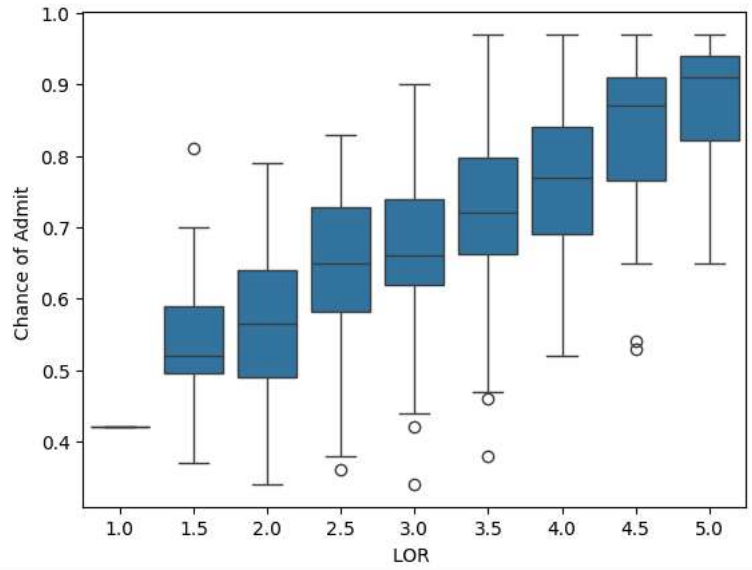

```
sns.boxplot(data = df,x = "University Rating", y = "Chance of Admit ")
```

```
<Axes: xlabel='University Rating', ylabel='Chance of Admit '>
```



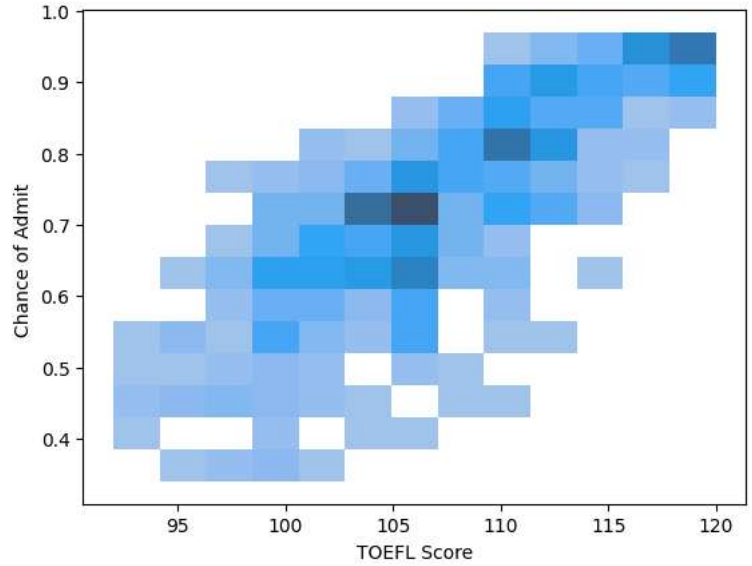

```
sns.boxplot(data = df,x = "LOR ", y = "Chance of Admit ")
```

```
<Axes: xlabel='LOR ', ylabel='Chance of Admit '>
```



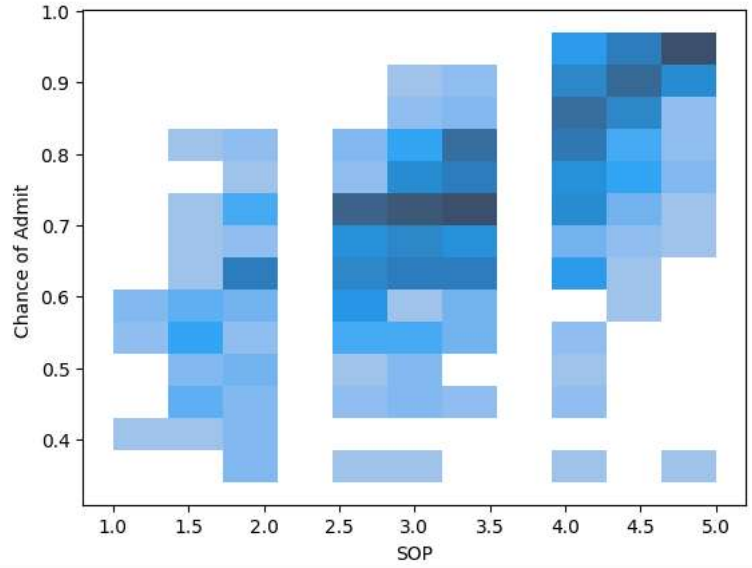

```
sns.histplot(x = df["TOEFL Score"], y = df["Chance of Admit "])
```

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



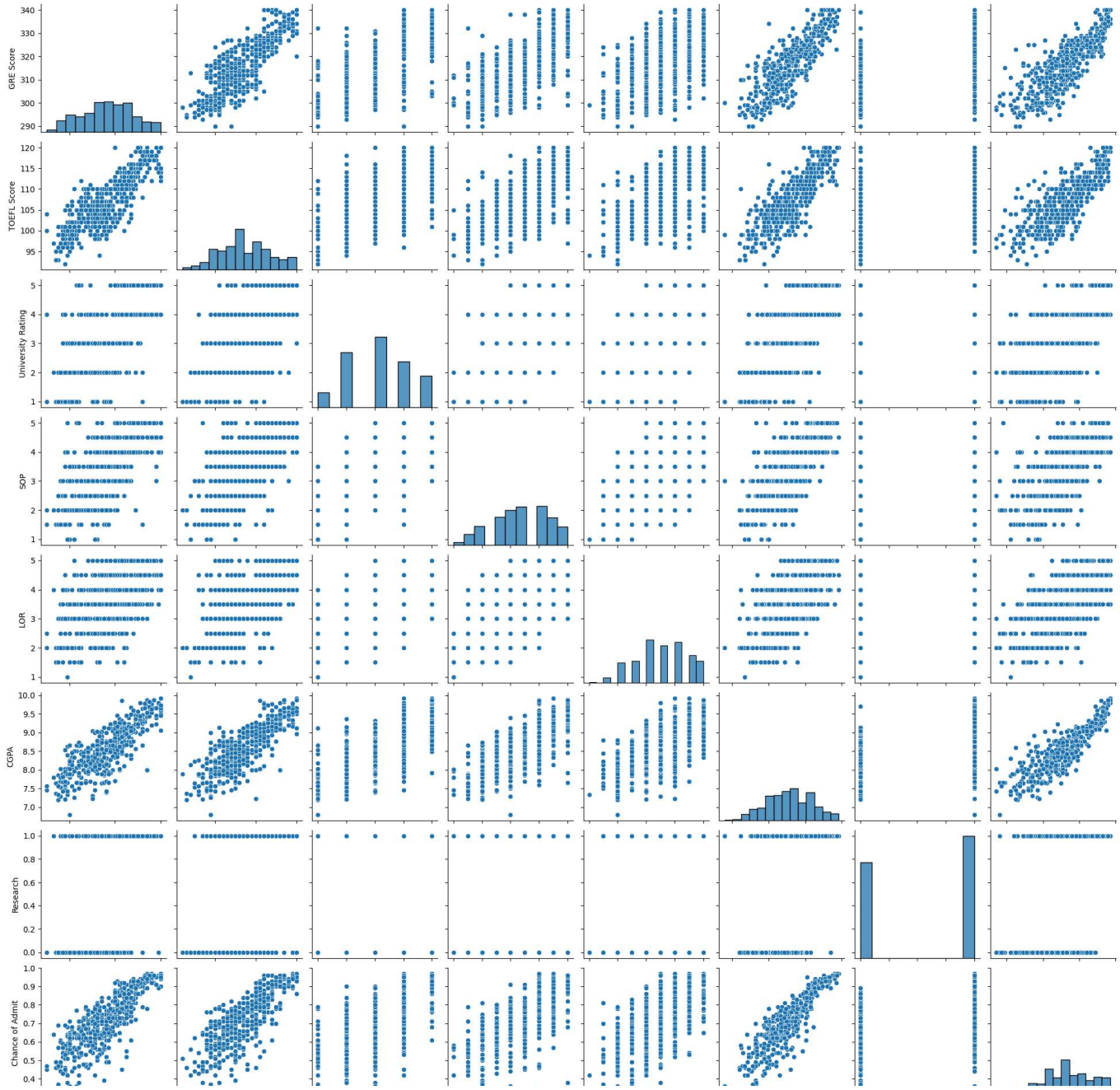

```
sns.histplot(x = df["SOP"], y = df["Chance of Admit "])
```

```
<Axes: xlabel='SOP', ylabel='Chance of Admit '>
```




```
sns.pairplot(data = df)
```

```
<seaborn.axisgrid.PairGrid at 0x7bc7cf1bb8b0>
```

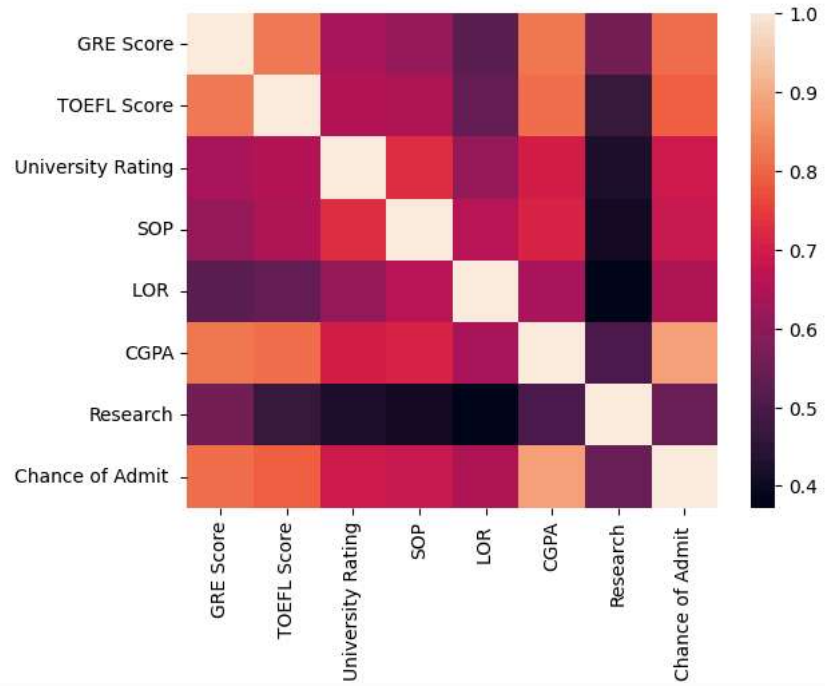



```
df1 = df.corr()
df1
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1.000000	0.827200	0.635376	0.613498	0.524679	0.825878	0.563398	0.810351
TOEFL Score	0.827200	1.000000	0.649799	0.644410	0.541563	0.810574	0.467012	0.792228
University Rating	0.635376	0.649799	1.000000	0.728024	0.608651	0.705254	0.427047	0.690132
SOP	0.613498	0.644410	0.728024	1.000000	0.663707	0.712154	0.408116	0.684137
LOR	0.524679	0.541563	0.608651	0.663707	1.000000	0.637469	0.372526	0.645365
CGPA	0.825878	0.810574	0.705254	0.712154	0.637469	1.000000	0.501311	0.882413
Research	0.563398	0.467012	0.427047	0.408116	0.372526	0.501311	1.000000	0.545871
Chance of Admit	0.810351	0.792228	0.690132	0.684137	0.645365	0.882413	0.545871	1.000000

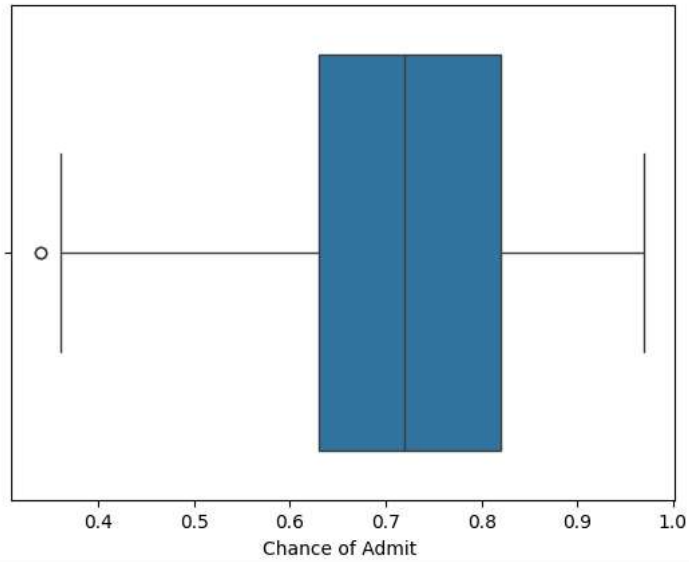
```
sns.heatmap(data = df1)
```

```
<Axes: >
```




```
# Plotting the boxplot for checking outliers
sns.boxplot(data = df,x = "Chance of Admit ")
```

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




```
df.groupby("Research")["Chance of Admit "].agg(["min","max","count","sum"])
```

Research	min	max	count	sum
0	0.34	0.89	220	139.68
1	0.36	0.97	280	221.19

1. As we can see, the minimum is not too different from those who have done research to those who haven't.
2. The rate of admission has a difference of 0.08, i.e. 8% difference in acceptance rate.
3. There is a noticable difference in admission count between those who have/haven't done research.
4. There is a huge difference in the sum total of admission chance between the two.

```
lower_limit_95 = np.percentile(sample_survey,5)
upper_limit_95 = np.percentile(sample_survey, 95)

lower_limit_97= np.percentile(sample_survey,2.5)
upper_limit_97 = np.percentile(sample_survey, 97.5)

lower_limit_99 = np.percentile(sample_survey,0.5)
upper_limit_99 = np.percentile(sample_survey, 99.5)

print("The 95% confidence interval is : (",lower_limit_95,",",upper_limit_95,")")
print("The 97.5% confidence interval is : (",lower_limit_97,",",upper_limit_97,")")
print("The 99.5% confidence interval is : (",lower_limit_99,",",upper_limit_99,")")
```

```
The 95% confidence interval is : ( 0.7111820000000001 , 0.7318880000000001 )
The 97.5% confidence interval is : ( 0.7099214999999999 , 0.7346109999999999 )
The 99.5% confidence interval is : ( 0.708376 , 0.7409146999999999 )
```

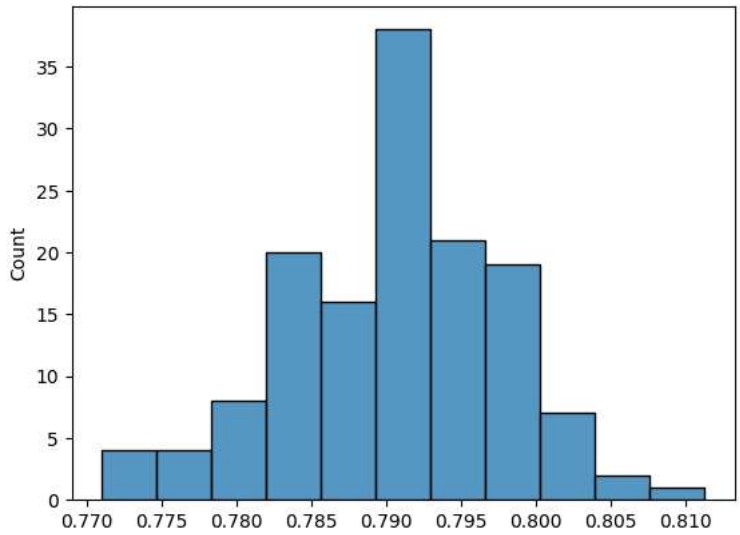
```
#differentiating on the basis of research
re_true= df.loc[df["Research"]== 1]
re_false= df.loc[df["Research"]== 0]
re_true.shape,re_false.shape
```

```
((280, 8), (220, 8))
```

```
sample_size = 280
number_of_samples = 140
survey = [] # list which will store the mean of every sample
for sample_number in range(number_of_samples):
  samples = np.random.choice(re_true["Chance of Admit "],size = sample_size)# selecting random values
  samples_mean = np.mean(samples) # calculating the mean of sample
  survey.append(samples_mean) # Appending the mean into the list

print("The mean of samples are : ",survey)
sns.histplot(survey)
print("The mean of averages of samples are : ", np.mean(survey))
```

```
The mean of samples are :  [0.7933571428571428, 0.7980357142857143, 0.7968928571428572, 0.7864642857142857, 0.8042142857142858, 0.79025,
The mean of averages of samples are :  0.790538775510204
```



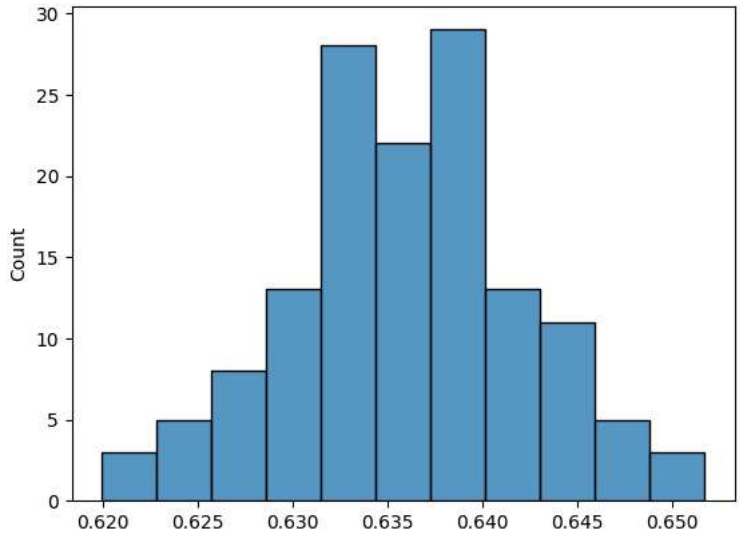
We can see that the sample size of 280 gives a normal/gaussian distribution. We can see the mean of admission chances is approximately 0.792.

```
sample_size = 280
number_of_samples = 140
survey = [] # list which will store the mean of every sample
for sample_number in range(number_of_samples):
  samples = np.random.choice(re_false["Chance of Admit "],size = sample_size)# selecting random values
  samples_mean = np.mean(samples) # calculating the mean of sample
  survey.append(samples_mean) # Appending the mean into the list

print("The mean of samples are : ",survey)
```

```
sns.histplot(survey)
print("The mean of averages of samples are : ", np.mean(survey))
```

```
The mean of samples are :  [0.63075, 0.6407142857142857, 0.6319999999999999, 0.631, 0.6305357142857143, 0.6355000000000001, 0.6351428571
The mean of averages of samples are :  0.6359961734693877
```



We can see that the sample size of 280 gives a normal/gaussian distribution. We can see the mean of admission chances is approximately 0.6375.

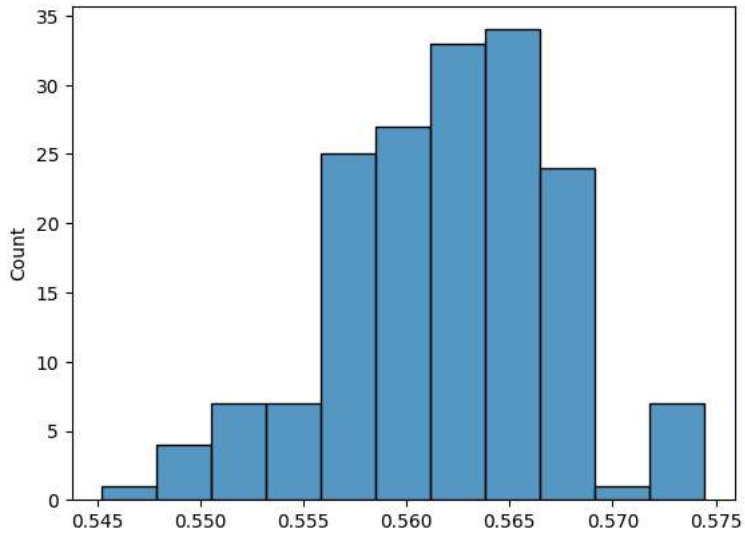
```
#differentiating on the basis of research
ur_1= df.loc[df["University Rating"]== 1]
ur_2= df.loc[df["University Rating"]== 2]
ur_3= df.loc[df["University Rating"]== 3]
ur_4= df.loc[df["University Rating"]== 4]
ur_5= df.loc[df["University Rating"]== 5]
ur_1.shape,ur_2.shape,ur_3.shape,ur_4.shape,ur_5.shape
```

```
((34, 8), (126, 8), (162, 8), (105, 8), (73, 8))
```

```
sample_size = 340
number_of_samples = 170
ur1_survey = [] # list which will store the mean of every sample
for sample_number in range(number_of_samples):
  samples = np.random.choice(ur_1["Chance of Admit "],size = sample_size)# selecting random values
  samples_mean = np.mean(samples) # calculating the mean of sample
  ur1_survey.append(samples_mean) # Appending the mean into the list

print("The mean of samples are : ",ur1_survey)
sns.histplot(ur1_survey)
print("The mean of averages of samples are : ", np.mean(ur1_survey))
```

```
The mean of samples are :  [0.5611470588235294, 0.559, 0.5669117647058823, 0.5620882352941176, 0.568529411764706, 0.5518823529411765, 0.
The mean of averages of samples are :  0.5619069204152248
```

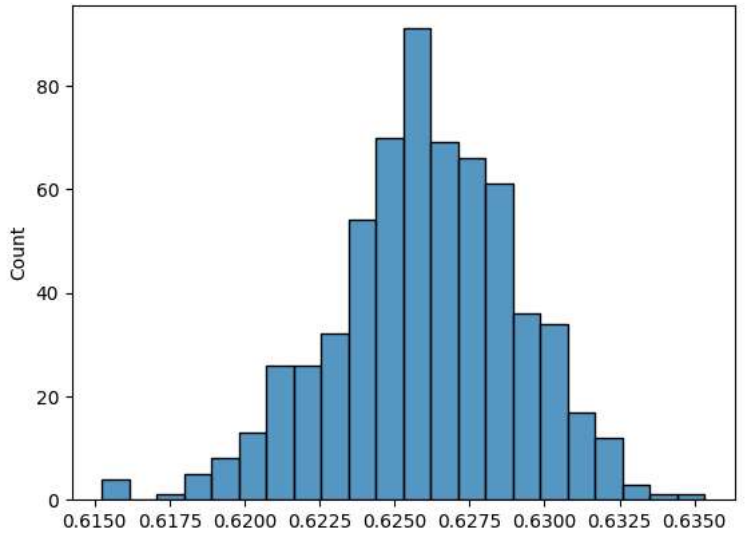


We can see that the sample size of 340 gives a normal/gaussian distribution. We can see the mean of admission chances is approximately 0.564.

```
sample_size = 1260
number_of_samples = 630
ur2_survey = [] # list which will store the mean of every sample
for sample_number in range(number_of_samples):
  samples = np.random.choice(ur_2["Chance of Admit "],size = sample_size)# selecting random values
  samples_mean = np.mean(samples) # calculating the mean of sample
  ur2_survey.append(samples_mean) # Appending the mean into the list

print("The mean of samples are : ",ur2_survey)
sns.histplot(ur2_survey)
print("The mean of averages of samples are : ", np.mean(ur2_survey))
```

```
The mean of samples are :  [0.6239047619047619, 0.6245634920634922, 0.626484126984127, 0.6312222222222222, 0.6301587301587301, 0.625023
The mean of averages of samples are :  0.6260604686318972
```



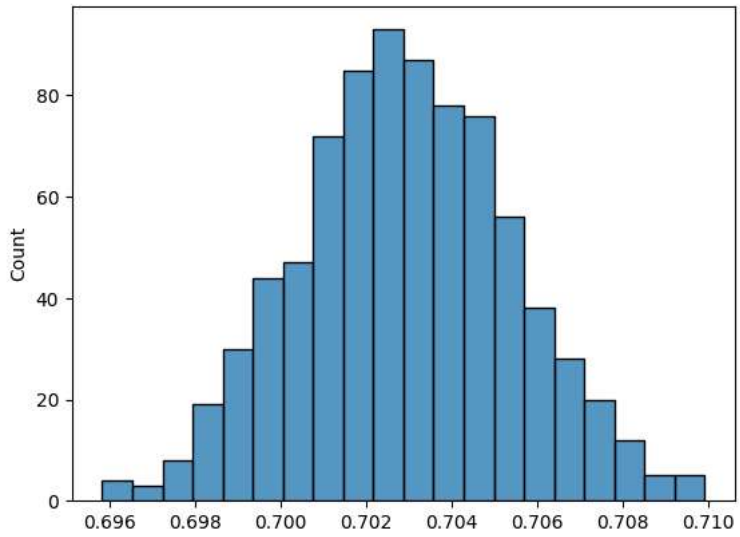
We can see that the sample size of 1260 gives a normal/gaussian distribution. We can see the mean of admission chances is approximately 0.6262.

```
sample_size = 1620
number_of_samples = 810
ur3_survey = [] # list which will store the mean of every sample
```

```
for sample_number in range(number_of_samples):
  samples = np.random.choice(ur_3["Chance of Admit "],size = sample_size)# selecting random values
  samples_mean = np.mean(samples) # calculating the mean of sample
  ur3_survey.append(samples_mean) # Appending the mean into the list

print("The mean of samples are : ",ur3_survey)
sns.histplot(ur3_survey)
print("The mean of averages of samples are : ", np.mean(ur3_survey))
```

```
The mean of samples are :  [0.707716049382716, 0.7011296296296298, 0.7002962962962963, 0.7036604938271603, 0.7018333333333333, 0.7045802
The mean of averages of samples are :  0.7029273662551441
```

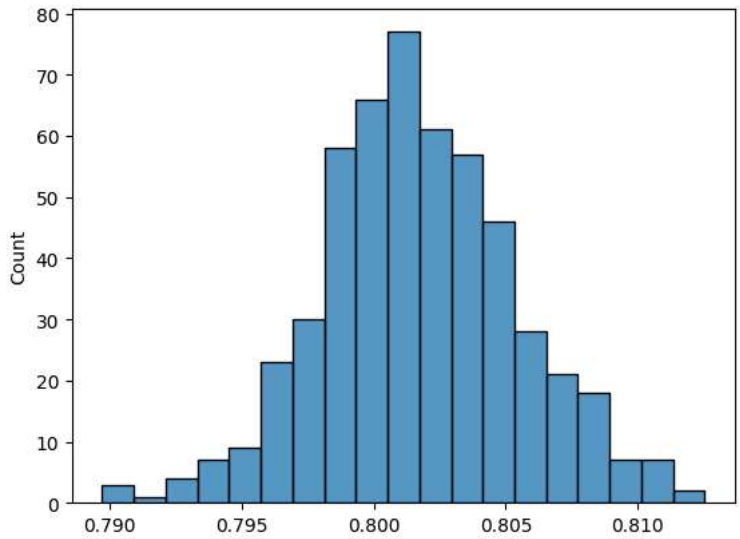


We can see that the sample size of 1620 gives a normal/gaussian distribution. We can see the mean of admission chances is approximately 0.7025.

```
sample_size = 1050
number_of_samples = 525
ur4_survey = [] # list which will store the mean of every sample
for sample_number in range(number_of_samples):
  samples = np.random.choice(ur_4["Chance of Admit "],size = sample_size)# selecting random values
  samples_mean = np.mean(samples) # calculating the mean of sample
  ur4_survey.append(samples_mean) # Appending the mean into the list

print("The mean of samples are : ",ur4_survey)
sns.histplot(ur4_survey)
print("The mean of averages of samples are : ", np.mean(ur4_survey))
```

```
The mean of samples are :  [0.804447619047619, 0.8077428571428571, 0.8098095238095238, 0.8068095238095239, 0.8075238095238096, 0.801561
The mean of averages of samples are :  0.8016822131519274
```

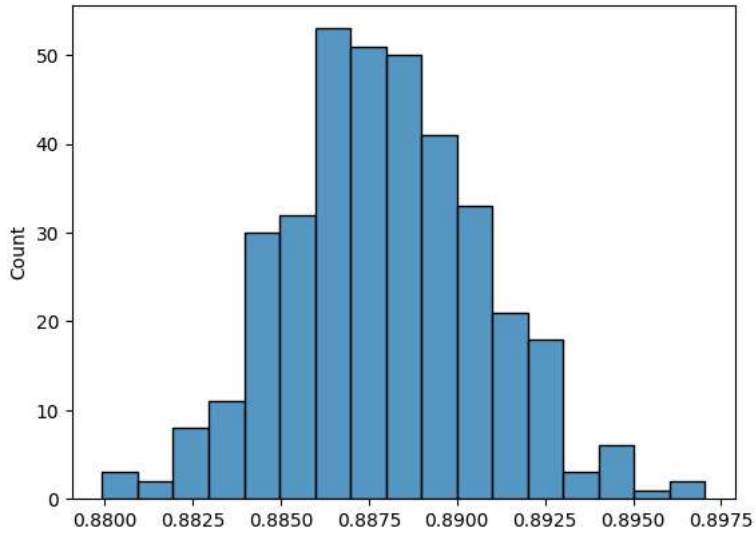


We can see that the sample size of 1050 gives a normal/gaussian distribution. We can see the mean of admission chances is approximately 0.801.

```
sample_size = 730
number_of_samples = 365
ur5_survey = [] # list which will store the mean of every sample
for sample_number in range(number_of_samples):
  samples = np.random.choice(ur_5["Chance of Admit "],size = sample_size)# selecting random values
  samples_mean = np.mean(samples) # calculating the mean of sample
  ur5_survey.append(samples_mean) # Appending the mean into the list

print("The mean of samples are : ",ur5_survey)
sns.histplot(ur5_survey)
print("The mean of averages of samples are : ", np.mean(ur5_survey))
```

```
The mean of samples are :  [0.8864931506849313, 0.8843013698630137, 0.891027397260274, 0.8875753424657533, 0.8901917808219177, 0.889027
The mean of averages of samples are :  0.8879577406642897
```



We can see that the sample size of 730 gives a normal/gaussian distribution. We can see the mean of admission chances is approximately 0.887.

```
#  CONFIDENCE INTERVAL FOR university with rating 1(95%, 97.5%, 99.5%)
lower_limit_95 = np.percentile(ur1_survey,5)
upper_limit_95 = np.percentile(ur1_survey, 95)

lower_limit_97 = np.percentile(ur1_survey,2.5)
upper_limit_97 = np.percentile(ur1_survey, 97.5)

lower_limit_99 = np.percentile(ur1_survey,0.5)
upper_limit_99 = np.percentile(ur1_survey, 99.5)

print("The 95% confidence interval is : (",lower_limit_95,upper_limit_95,")")
print("The 97.5% confidence interval is : (",lower_limit_97,upper_limit_97,")")
print("The 99.5% confidence interval is : (",lower_limit_99,upper_limit_99,")")
```

```
The 95% confidence interval is : ( 0.552039705882353 0.5690411764705882 )
The 97.5% confidence interval is : ( 0.5503492647058823 0.5730124999999999 )
The 99.5% confidence interval is : ( 0.5476708823529413 0.5738741176470589 )
```

```
#  CONFIDENCE INTERVAL FOR university with rating 2(95%, 97.5%, 99.5%)
lower_limit_95 = np.percentile(ur2_survey,5)
upper_limit_95 = np.percentile(ur2_survey, 95)

lower_limit_97 = np.percentile(ur2_survey,2.5)
upper_limit_97 = np.percentile(ur2_survey, 97.5)

lower_limit_99 = np.percentile(ur2_survey,0.5)
upper_limit_99 = np.percentile(ur2_survey, 99.5)

print("The 95% confidence interval is : (",lower_limit_95,upper_limit_95,")")
```

```
print("The 97.5% confidence interval is : (",lower_limit_97,upper_limit_97,")")
print("The 99.5% confidence interval is : (",lower_limit_99,upper_limit_99,")")
```

```
The 95% confidence interval is : ( 0.6208162698412698 0.6309130952380952 )
The 97.5% confidence interval is : ( 0.6196125 0.6317166666666667 )
The 99.5% confidence interval is : ( 0.6162422619047618 0.6328727777777778 )
```

```
#  CONFIDENCE INTERVAL FOR university with rating 3(95%, 97.5%, 99.5%)
lower_limit_95 = np.percentile(ur3_survey,5)
upper_limit_95 = np.percentile(ur3_survey, 95)

lower_limit_97 = np.percentile(ur3_survey,2.5)
upper_limit_97 = np.percentile(ur3_survey, 97.5)

lower_limit_99 = np.percentile(ur3_survey,0.5)
upper_limit_99 = np.percentile(ur3_survey, 99.5)

print("The 95% confidence interval is : (",lower_limit_95,upper_limit_95,")")
print("The 97.5% confidence interval is : (",lower_limit_97,upper_limit_97,")")
print("The 99.5% confidence interval is : (",lower_limit_99,upper_limit_99,")")
```

```
The 95% confidence interval is : ( 0.6988910493827161 0.7071302469135802 )
The 97.5% confidence interval is : ( 0.698105864197531 0.7079217592592594 )
The 99.5% confidence interval is : ( 0.696710987654321 0.7092091666666668 )
```

```
#  CONFIDENCE INTERVAL FOR university with rating 4(95%, 97.5%, 99.5%)
lower_limit_95 = np.percentile(ur4_survey,5)
upper_limit_95 = np.percentile(ur4_survey, 95)

lower_limit_97 = np.percentile(ur4_survey,2.5)
upper_limit_97 = np.percentile(ur4_survey, 97.5)

lower_limit_99 = np.percentile(ur4_survey,0.5)
upper_limit_99 = np.percentile(ur4_survey, 99.5)

print("The 95% confidence interval is : (",lower_limit_95,upper_limit_95,")")
print("The 97.5% confidence interval is : (",lower_limit_97,upper_limit_97,")")
print("The 99.5% confidence interval is : (",lower_limit_99,upper_limit_99,")")
```

```
The 95% confidence interval is : ( 0.7959352380952381 0.8080857142857142 )
The 97.5% confidence interval is : ( 0.7940619047619047 0.8092266666666668 )
The 99.5% confidence interval is : ( 0.7910283809523809 0.8110708571428572 )
```

```
#  CONFIDENCE INTERVAL FOR university with rating 5(95%, 97.5%, 99.5%)
lower_limit_95 = np.percentile(ur5_survey,5)
upper_limit_95 = np.percentile(ur5_survey, 95)

lower_limit_97 = np.percentile(ur5_survey,2.5)
upper_limit_97 = np.percentile(ur5_survey, 97.5)

lower_limit_99 = np.percentile(ur5_survey,0.5)
upper_limit_99 = np.percentile(ur5_survey, 99.5)

print("The 95% confidence interval is : (",lower_limit_95,upper_limit_95,")")
print("The 97.5% confidence interval is : (",lower_limit_97,upper_limit_97,")")
print("The 99.5% confidence interval is : (",lower_limit_99,upper_limit_99,")")
```

```
The 95% confidence interval is : ( 0.8836164383561643 0.8926328767123286 )
The 97.5% confidence interval is : ( 0.8827534246575343 0.8936986301369862 )
The 99.5% confidence interval is : ( 0.8808386301369863 0.895704383561644 )
```

```
#normalization
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
df1=pd.DataFrame(scaler.fit_transform(df),columns=df.columns)
df1
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	0.94	0.928571	0.75	0.875	0.875	0.913462	1.0	0.920635
1	0.68	0.535714	0.75	0.750	0.875	0.663462	1.0	0.666667
2	0.52	0.428571	0.50	0.500	0.625	0.384615	1.0	0.603175
3	0.64	0.642857	0.50	0.625	0.375	0.599359	1.0	0.730159
4	0.48	0.392857	0.25	0.250	0.500	0.451923	0.0	0.492063
...	...	...	...	...	...	...	...	...
495	0.84	0.571429	1.00	0.875	0.750	0.711538	1.0	0.841270
496	0.94	0.892857	1.00	1.000	1.000	0.983974	1.0	0.984127
497	0.80	1.000000	1.00	0.875	1.000	0.884615	1.0	0.936508
498	0.44	0.392857	0.75	0.750	1.000	0.522436	0.0	0.619048
499	0.74	0.750000	0.75	0.875	0.875	0.717949	0.0	0.793651

500 rows × 8 columns

```
y=df1['Chance of Admit ']
x=df1.drop('Chance of Admit ',axis=1)
y.shape, x.shape
```

```
((500,), (500, 7))
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.27,random_state=42)
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
model.score(x_test,y_test)
```

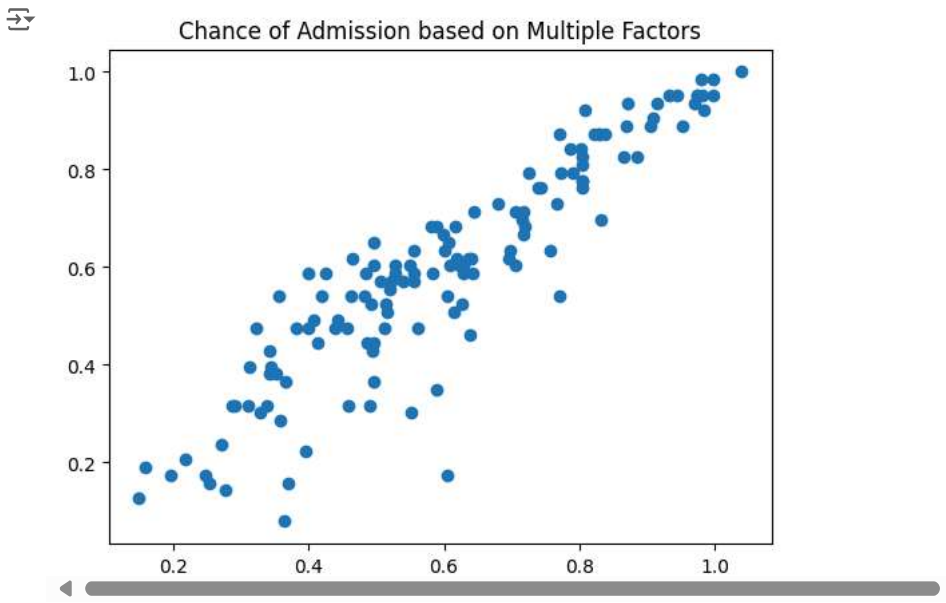
```
0.8300766064752694
```

```
x_pred=model.predict(x_train)
y_pred=model.predict(x_test)
x_pred,y_pred
```

```
(array([0.25910572, 0.76974991, 0.98911123, 0.50784282, 0.53849106,
        0.72727658, 0.44678876, 0.75845709, 0.29795751, 0.96537613,
        0.50322751, 0.99230229, 0.702859  , 0.47764739, 0.60155031,
        0.85074775, 0.50995747, 0.77951609, 0.59958673, 0.41497075,
        0.4234378 , 0.64940326, 0.73978131, 0.26986897, 0.41714091,
        0.52819931, 0.66992677, 0.61110881, 0.23680616, 0.43414324,
        0.46458278, 0.66756311, 0.40760954, 0.65409307, 0.91957004,
        0.32255533, 0.45248834, 0.56618736, 0.2285012 , 0.55747944,
        0.71737982, 0.62792175, 0.48611204, 0.74441781, 0.39642919,
        0.45500902, 0.57519179, 0.36869075, 0.71601324, 0.44333993,
        0.87869486, 0.81008056, 0.62934617, 0.76632724, 0.35075734,
        0.38053035, 0.30823593, 0.42393729, 0.2078023 , 0.55919752,
        0.73169661, 0.64496827, 0.35915568, 0.80500998, 0.27210578,
        0.67982086, 0.86437249, 0.73986702, 0.71725124, 0.56148388,
        0.62262346, 0.63691728, 0.66953913, 0.61776681, 0.34835664,
        0.59059121, 0.69755065, 0.64774576, 0.58633768, 0.56276931,
        0.28464312, 0.95422188, 0.34058313, 0.40279162, 0.86156173,
        0.79032421, 0.39012029, 0.4995276 , 0.4505752 , 0.69535987,
        0.65732294, 0.58095287, 0.51644348, 0.88485405, 0.45386718,
        0.68619102, 0.67275877, 0.36671597, 0.59045834, 0.70932247,
        0.26890613, 0.70253346, 0.43075954, 0.89601635, 0.6968151 ,
        0.58801418, 0.53785101, 0.67752599, 0.6265176 , 0.68731824,
        0.56416775, 0.98985563, 0.27340855, 0.46149703, 0.47378636,
        0.80605516, 0.58170268, 0.44410075, 0.55304474, 0.79365869,
        0.45267485, 0.5227281 , 0.78795175, 0.77729079, 0.75165113,
        0.81674341, 0.49679201, 0.26409018, 0.38844023, 1.04071417,
        0.31061082, 0.91266037, 0.82693176, 0.41675689, 0.49123245,
        0.64962995, 1.05545321, 0.6010012 , 0.69041288, 0.8577035 ,
        0.70176092, 0.38419026, 0.75042832, 0.96400251, 0.52818323,
        0.58945572, 0.55237809, 0.48707562, 0.52606317, 0.9314521 ,
        0.25804936, 0.6652035 , 0.85285534, 0.20381939, 0.32475209,
        0.48987412, 0.49661421, 0.74932732, 0.5719529 , 0.52673451,
        0.45638387, 0.7751136 , 0.81399338, 0.78572963, 0.60385233,
        0.72880622, 0.40602229, 0.79873844, 0.78660542, 0.54343605,
        0.48388094, 0.70906729, 0.50757746, 0.71400124, 0.97183667,
        0.42722122, 0.42881911, 0.71372697, 0.91257981, 0.6416571 ,
        0.86879796, 0.36432061, 0.94249727, 0.44826626, 0.82060108,
```

```
0.70742723, 0.90802797, 0.86328126, 0.66715791, 0.70931741,
0.70394639, 0.59560755, 0.71237957, 0.63112896, 0.43297484,
0.30570064, 0.8166219 , 0.81691843, 0.13312595, 0.46955908,
0.88404372, 0.60031873, 0.47556561, 0.61042237, 0.34241577,
0.43535052, 0.29535257, 0.47922269, 0.2831877 , 0.46592077,
0.50852356, 0.56623365, 0.75843361, 0.39458257, 0.64353651,
0.78344377, 0.42113215, 0.46759206, 0.52943711, 0.67021153,
0.62003456, 0.55240076, 0.87249753, 1.00376161, 0.86184508,
0.89640849, 0.33259957, 0.8996692 , 0.4409497 , 0.9582617 ,
0.93526004, 0.53772357, 0.30353588, 0.58089682, 0.96673727,
0.27748474, 0.90053959, 0.59917886, 0.77850542, 0.69069407,
0.57365295, 0.95196074, 0.78296145, 0.48218551, 0.67776241,
0.76620005, 0.40202718, 0.34048844, 0.34374938, 0.39832319,
0.65792704, 0.41638103, 0.48014166, 0.49675454, 0.49055084,
0.51125459, 0.76101796, 0.84325519, 0.82745658, 0.55340597,
0.44865773, 0.50702463, 0.95033244, 0.81946054, 0.48391244,
1.01277427, 0.51859196, 0.22100992, 0.60100469, 0.68778532,
0.50127447, 0.64011456, 0.56980697, 0.56415838, 0.59578486,
0.24772621, 0.80879838, 0.76967746, 0.50866928, 0.80130734,
0.56895312, 0.91157832, 0.70219212, 0.56633517, 0.62770743,
0.73371422, 0.69150419, 0.70950925, 0.66232878, 0.44879099,
```

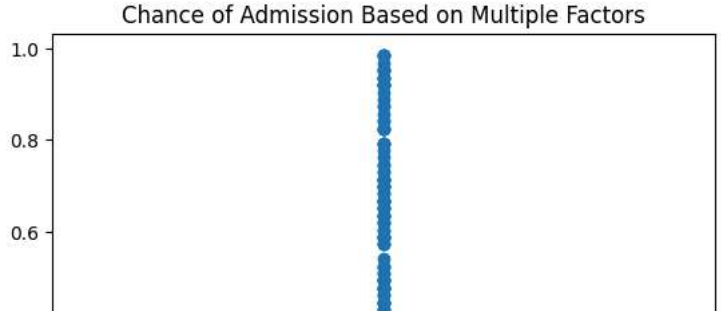
```
fig=plt.figure()
plt.scatter(y_pred,y_test)
plt.title("Chance of Admission based on Multiple Factors")
plt.show()
```




```
from sklearn.model_selection import train_test_split
x1_train,x1_test,y1_train,y1_test=train_test_split(x,y,test_size=0.27,random_state=1)
from sklearn.linear_model import Lasso
model1=Lasso(alpha=0.1)
model1.fit(x1_train,y1_train)
print(model1.score(x1_test,y1_test))
x1_pred=model1.predict(x1_train)
y1_pred=model1.predict(x1_test)
```

```
-0.002801601380672203
```

```
fig=plt.figure()
plt.scatter(y1_pred,y1_test)
plt.title("Chance of Admission Based on Multiple Factors")
plt.show()
```




```
from sklearn.model_selection import train_test_split
x2_train,x2_test,y2_train,y2_test=train_test_split(x,y,test_size=0.27,random_state=1)
from sklearn.linear_model import Ridge
model2=Ridge(alpha=0.1)
model2.fit(x2_train,y2_train)
print(model2.score(x2_test,y2_test))
x2_pred=model2.predict(x2_train)
y2_pred=model2.predict(x2_test)
```

```
0.8223504909854972
```

```
fig=plt.figure()
plt.scatter(y2_pred,y2_test)
plt.title("Chance of Admission Based on Multiple Factors")
plt.show()
```