```python
# Importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import shapiro
from statsmodels.graphics.gofplots import qqplot
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
PolynomialFeatures
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import
variance_inflation_factor

import warnings
warnings.simplefilter('ignore')

df = pd.read_csv('/content/sample_data/Jamboree_Admission.csv')
df.head()
```

```
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR
CGPA  \
0           1        337          118                  4  4.5   4.5
9.65
1           2        324          107                  4  4.0   4.5
8.87
2           3        316          104                  3  3.0   3.5
8.00
3           4        322          110                  3  3.5   2.5
8.67
4           5        314          103                  2  2.0   3.0
8.21

   Research  Chance of Admit
0         1              0.92
1         1              0.76
2         1              0.72
3         1              0.80
4         0              0.65
```

EDA

```python
df.shape
```

```
(500, 9)
```

Dataset has 500 rows and 9 columns

```
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

df.drop('Serial No.', axis = 1, inplace = True)
df.head()

   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

   Chance of Admit
0             0.92
1             0.76
2             0.72
3             0.80
4             0.65
```

Dropping Serial No column as it does not have any major role in the dataset, so keeping it might affect our model.

```
df.isna().sum()

GRE Score            0
TOEFL Score          0
```

```
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

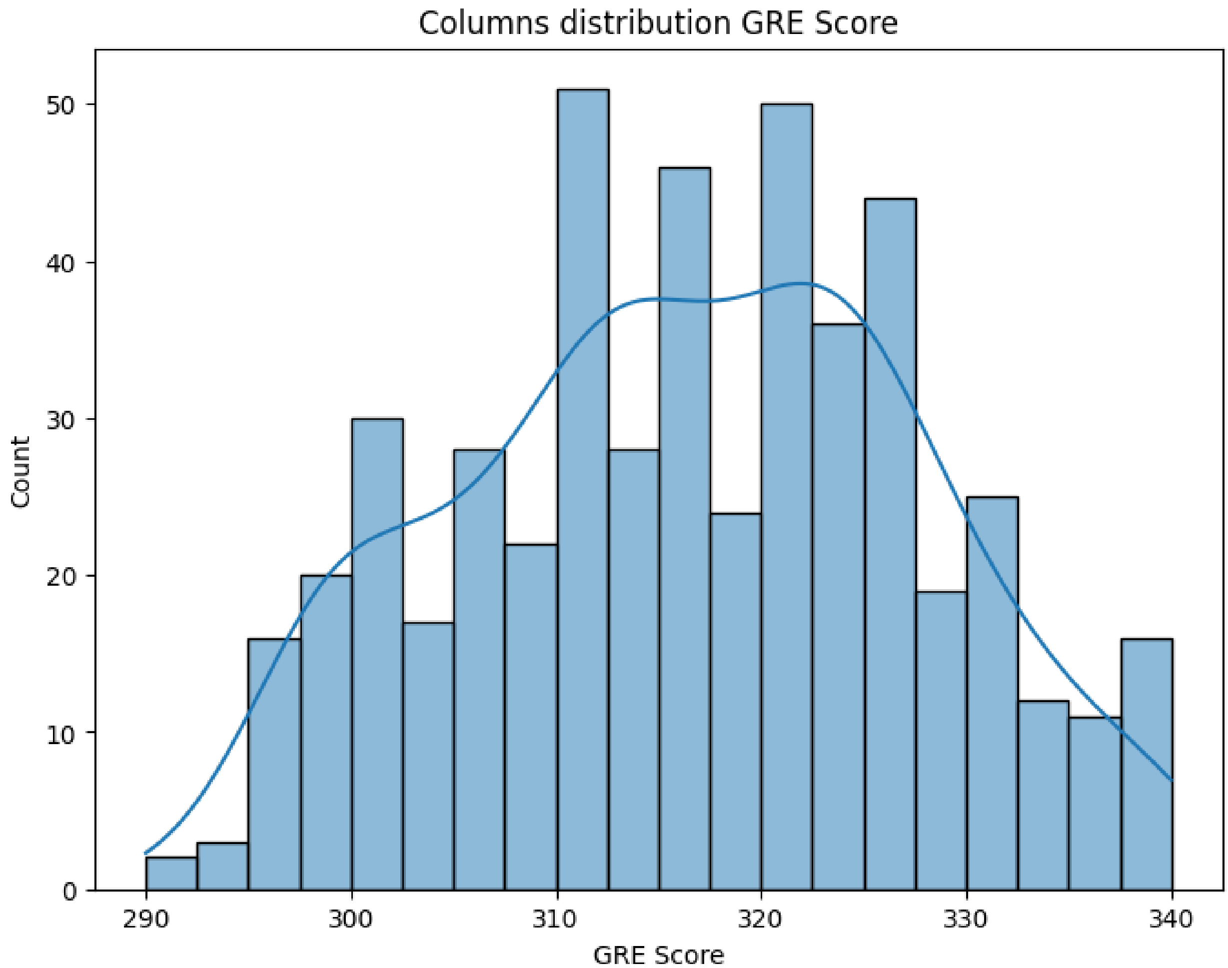
There is no missing values.

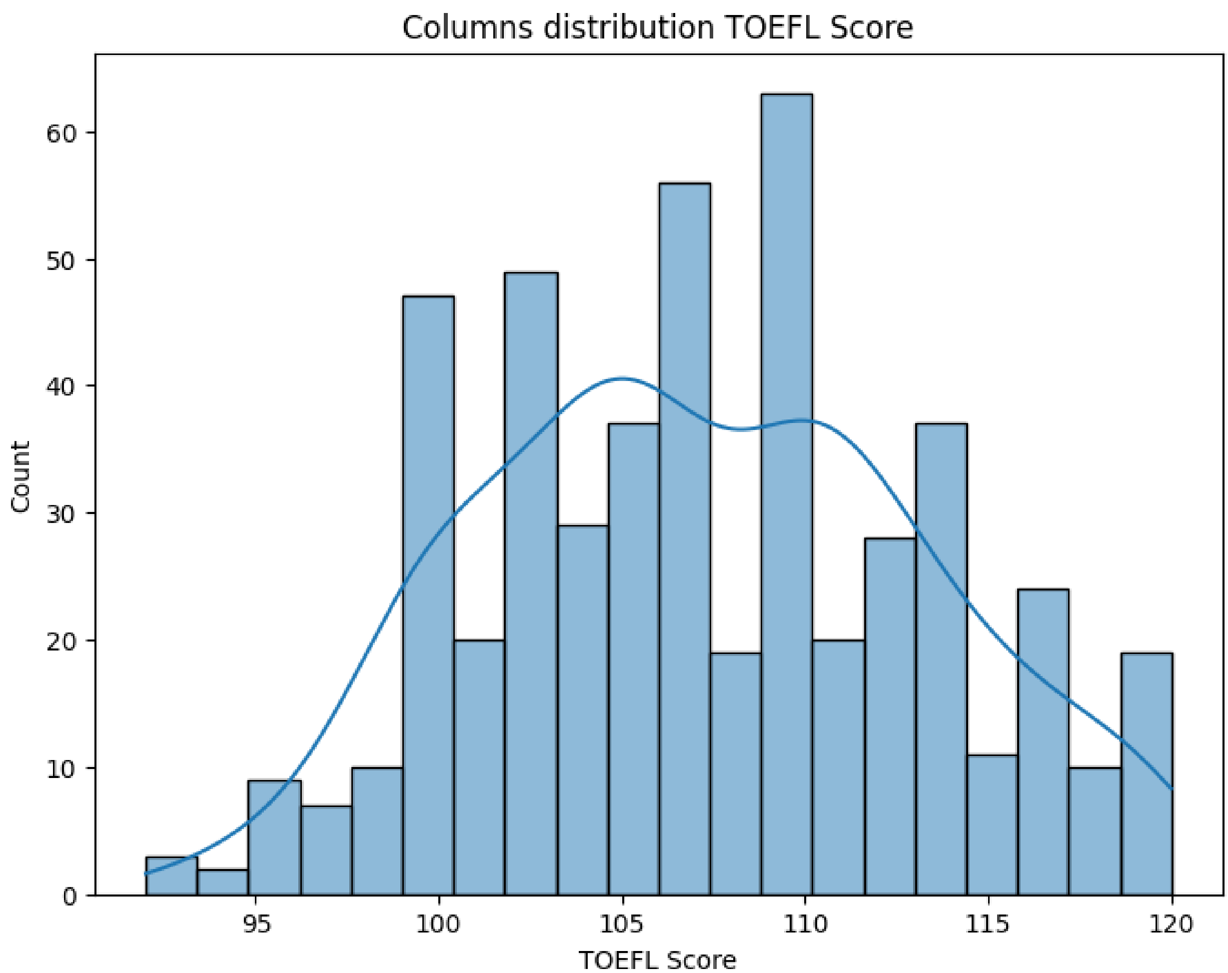
```
df.describe().T
```

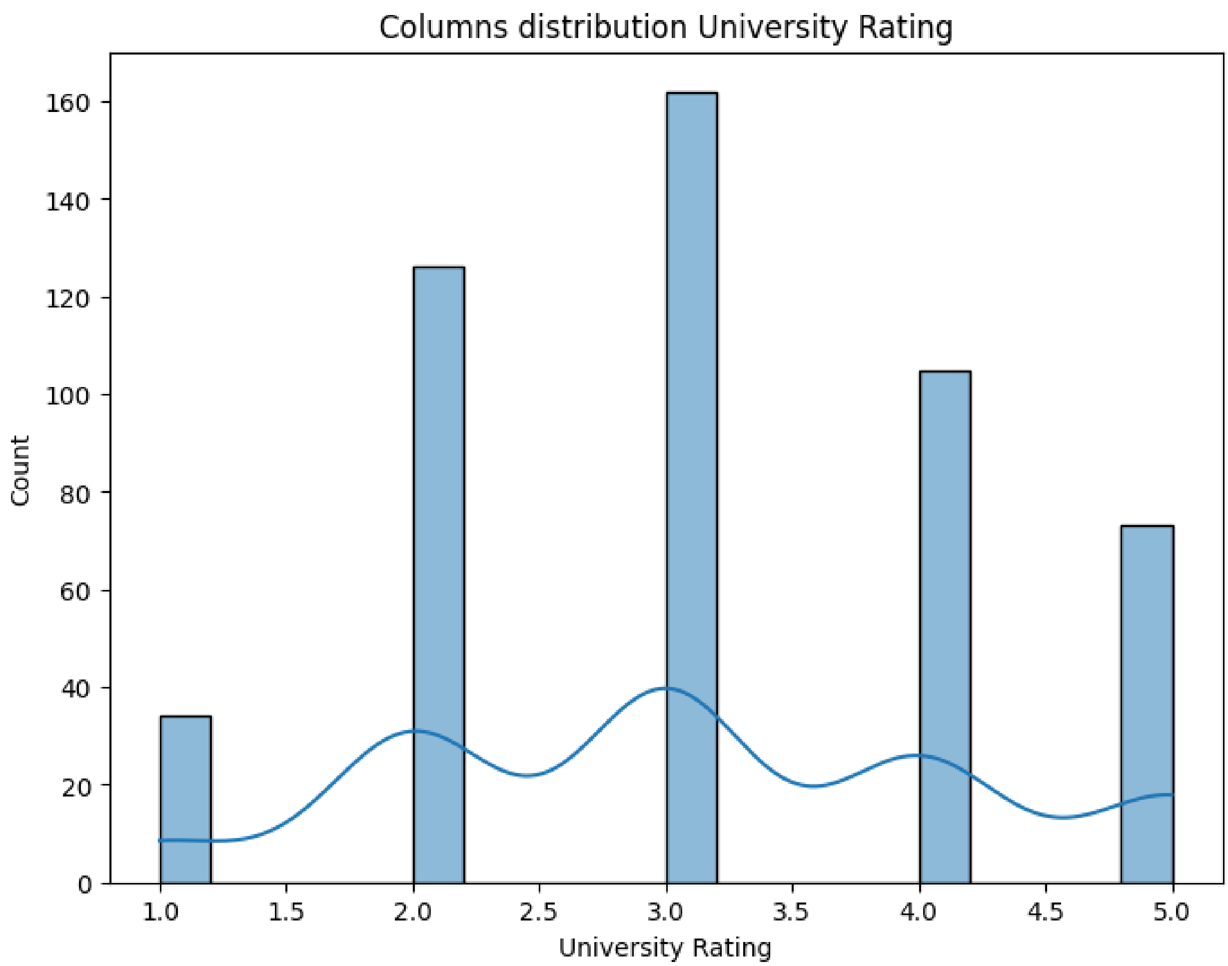
```
                   count       mean        std     min       25%
50%  \
GRE Score          500.0  316.47200  11.295148  290.00  308.0000
317.00
TOEFL Score        500.0  107.19200   6.081868   92.00  103.0000
107.00
University Rating  500.0    3.11400   1.143512    1.00    2.0000
3.00
SOP                500.0    3.37400   0.991004    1.00    2.5000
3.50
LOR                500.0    3.48400   0.925450    1.00    3.0000
3.50
CGPA               500.0    8.57644   0.604813    6.80    8.1275
8.56
Research           500.0    0.56000   0.496884    0.00    0.0000
1.00
Chance of Admit    500.0    0.72174   0.141140    0.34    0.6300
0.72

                      75%     max
GRE Score          325.00  340.00
TOEFL Score        112.00  120.00
University Rating    4.00    5.00
SOP                  4.00    5.00
LOR                  4.00    5.00
CGPA                 9.04    9.92
Research             1.00    1.00
Chance of Admit      0.82    0.97
```

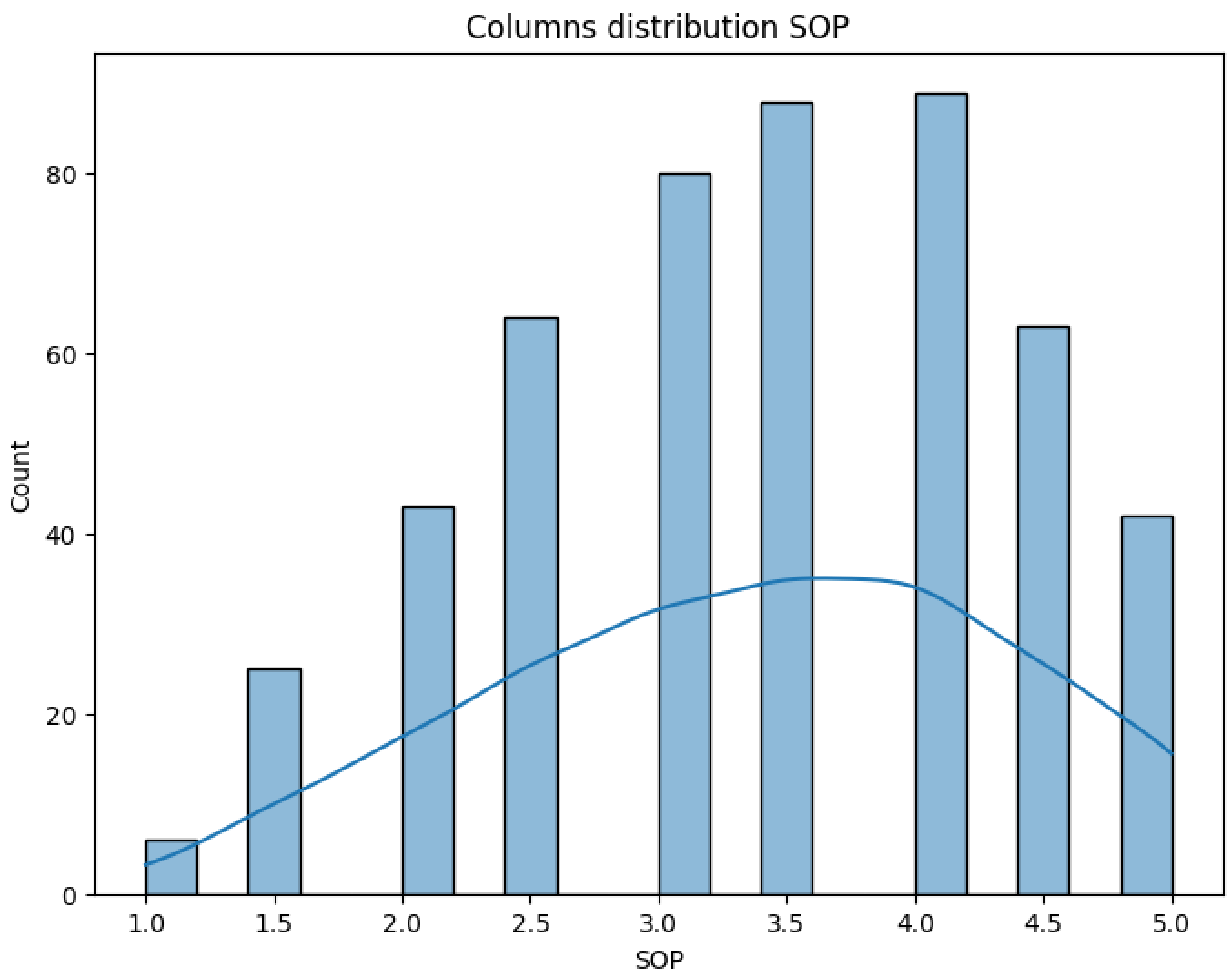
Analysing all numerical columns.

```
for col in ['GRE Score','TOEFL Score','University Rating','SOP','LOR
', 'CGPA']:
  plt.figure(figsize= (8, 6))
  sns.histplot(df[col], bins = 20, kde = True)
  plt.title(f'Columns distribution {col}')
  plt.show()
```

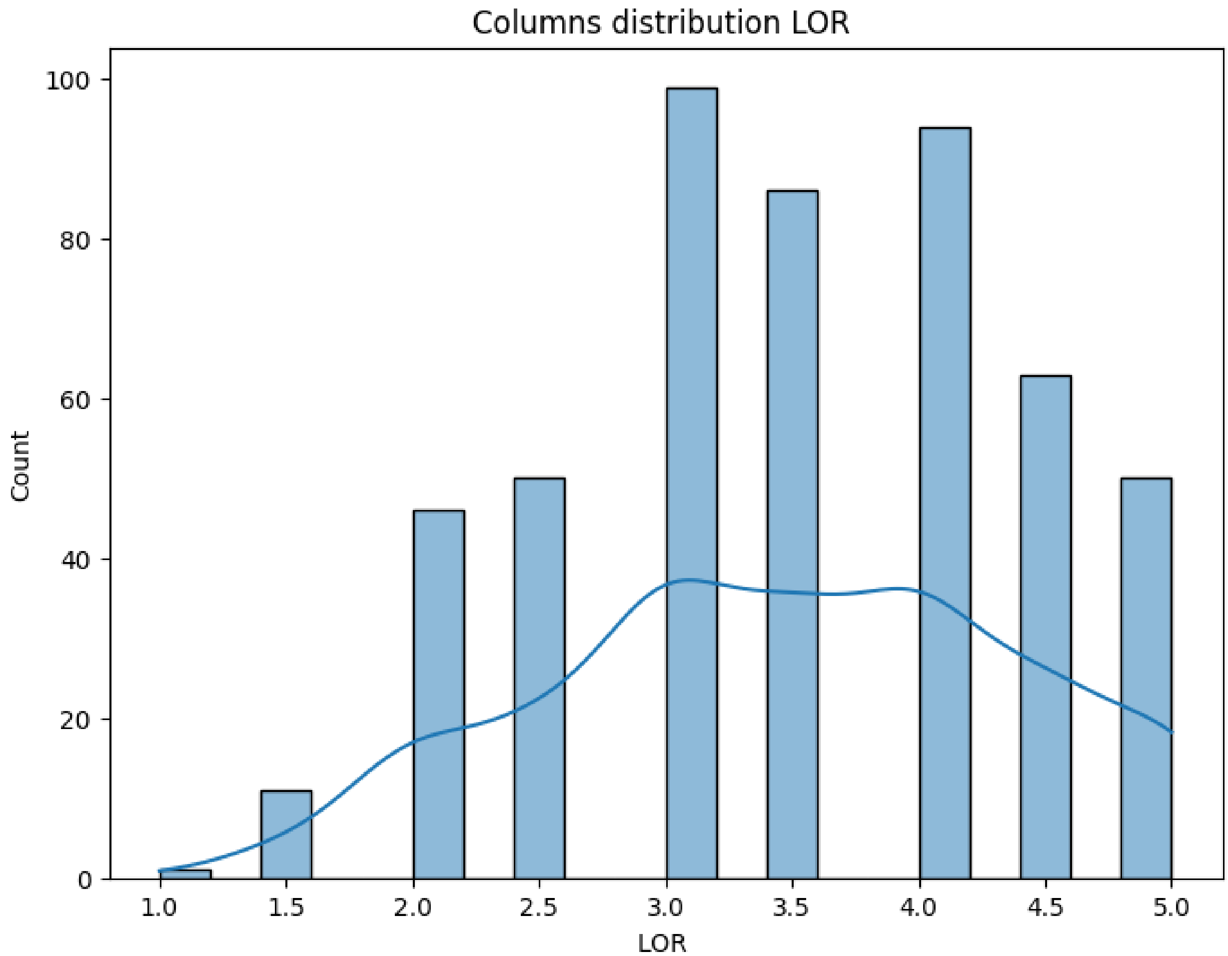

Columns distribution GRE Score
Count
GRE Score
0
10
20
30
40
50
290
300
310
320
330
340


Columns distribution TOEFL Score
Count
TOEFL Score
0
10
20
30
40
50
60
95
100
105
110
115
120

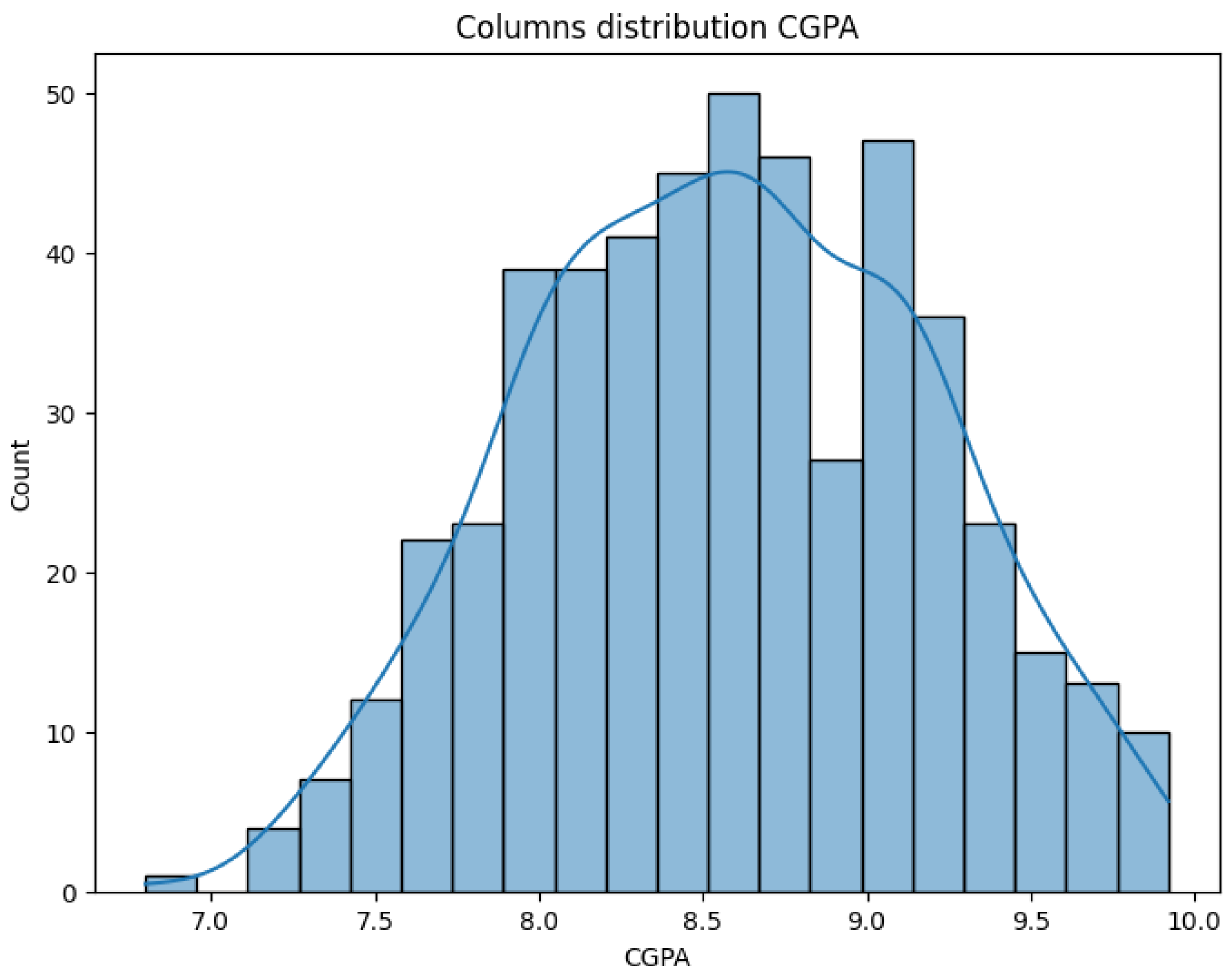

Columns distribution University Rating
Count
University Rating
0
20
40
60
80
100
120
140
160
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



Columns distribution SOP
Count
SOP
0
20
40
60
80
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



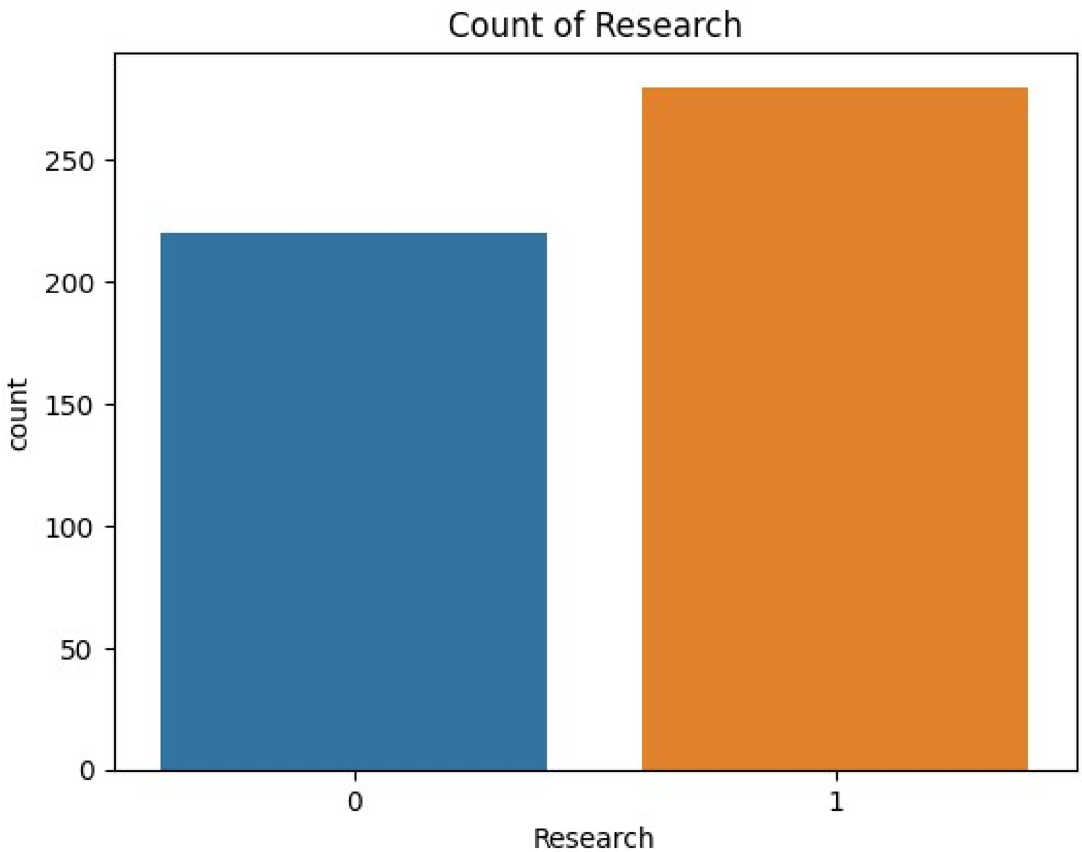
Columns distribution LOR
Count
LOR
0
20
40
60
80
100
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



1. Different graph shows the distribution of all numerical columns.

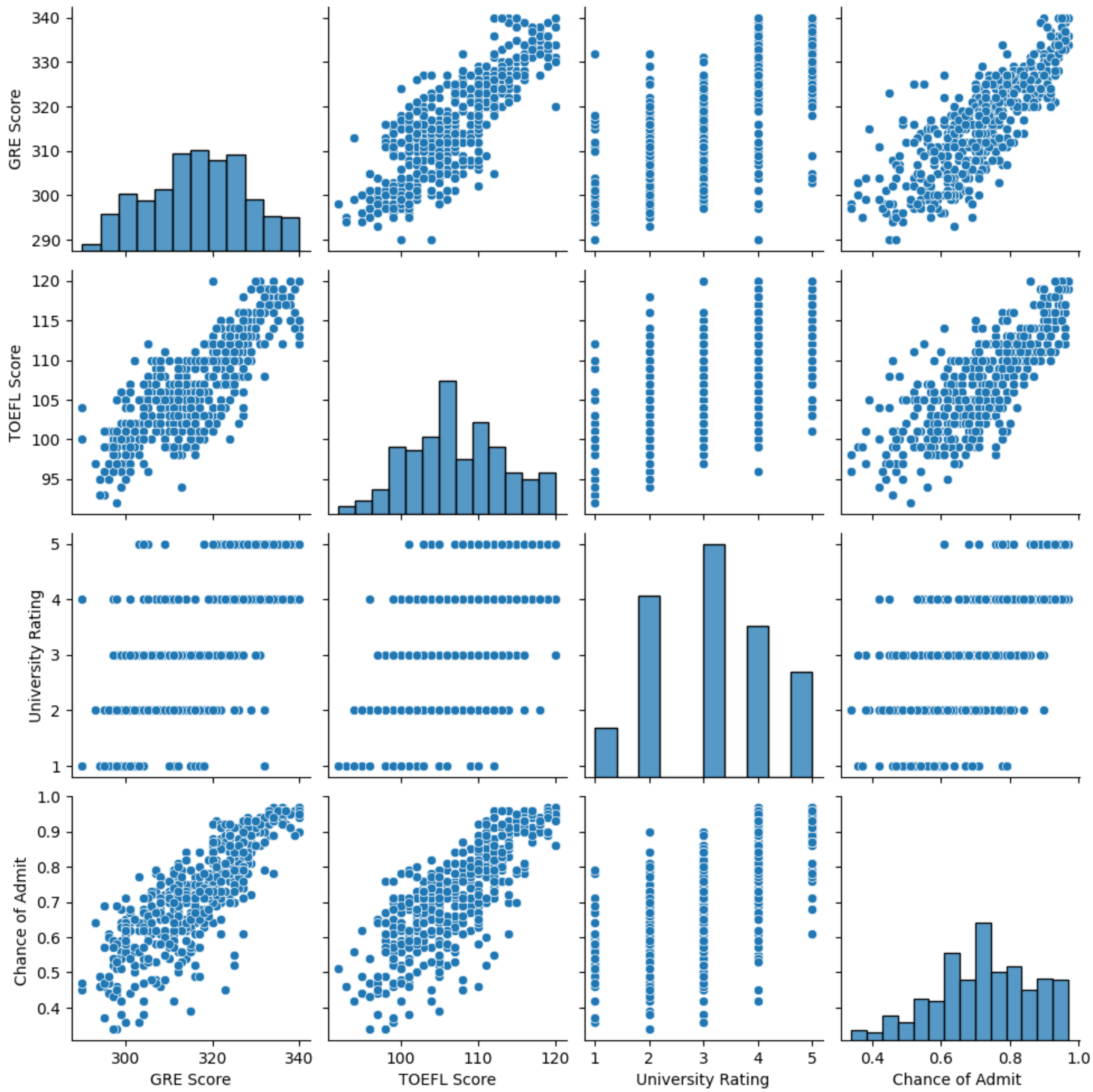
2. Graph shows GRE Score, TOEFL Score and CGPA column are normally distributed which indicates them being correlated.

```
df['Research'].value_counts()

1    280
0    220
Name: Research, dtype: int64

for col in ['Research']:
    sns.countplot(data=df, x=col)
    plt.title(f'Count of {col}')
    plt.show()
```



Using count plot to Research feature which shows 280 candidate with 1 year research experience and 220 candidate with no research experience

```
sns.pairplot(df[['GRE Score','TOEFL Score', 'University Rating',
'Chance of Admit ']])
plt.show()
```



1. Using pairplot to graphically analyse the correlation between different features.

2. Plot shows high correlation between chance of admit and gre score feature also with toefl score

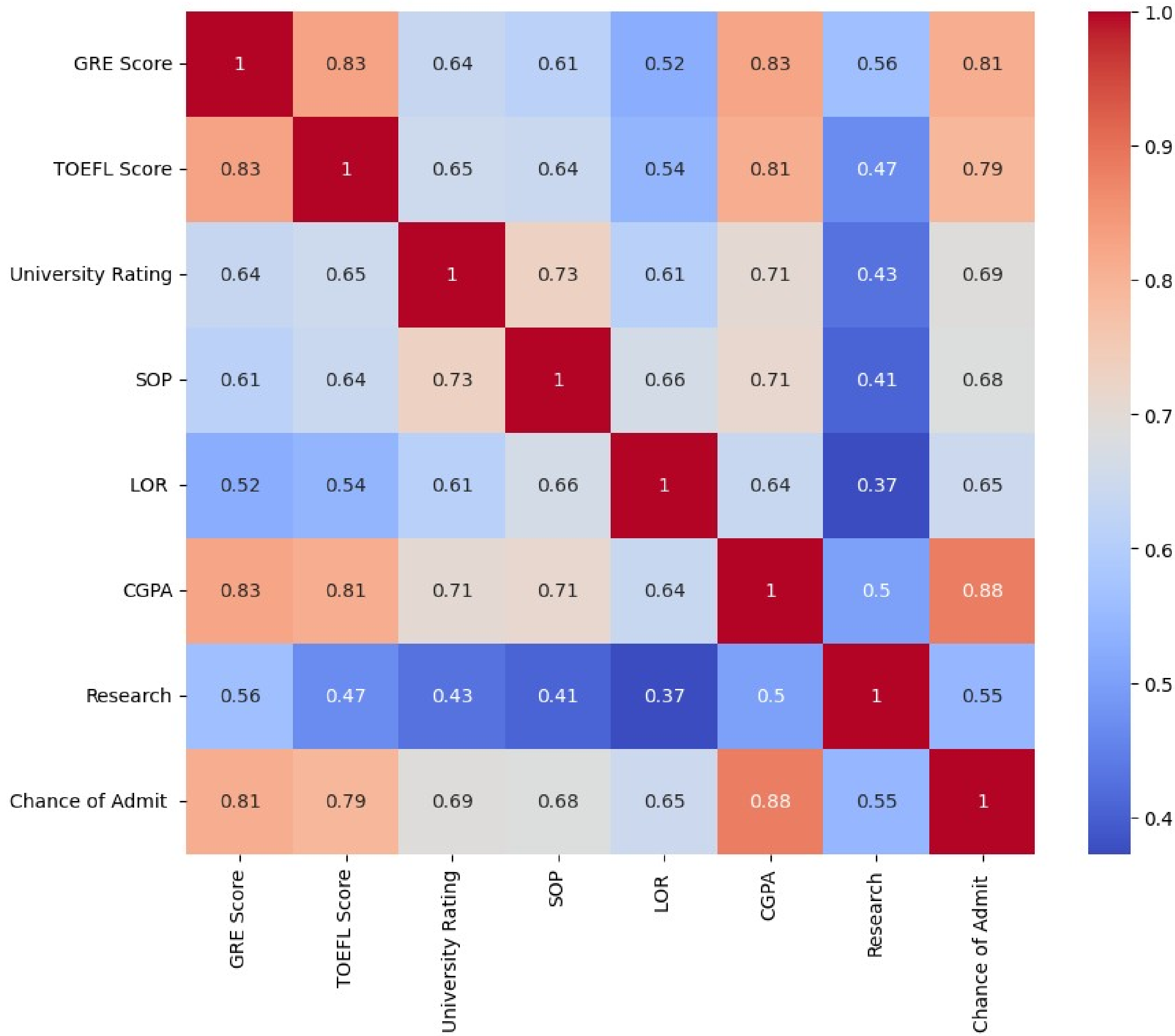
3. TOEFL and GRE score are highly related indicating greater score between anyone of them indicates that candidate will score high in the other feature also.

```
df.corr()

                    GRE Score  TOEFL Score  University Rating       SOP  \
GRE Score            1.000000     0.827200           0.635376  0.613498
```

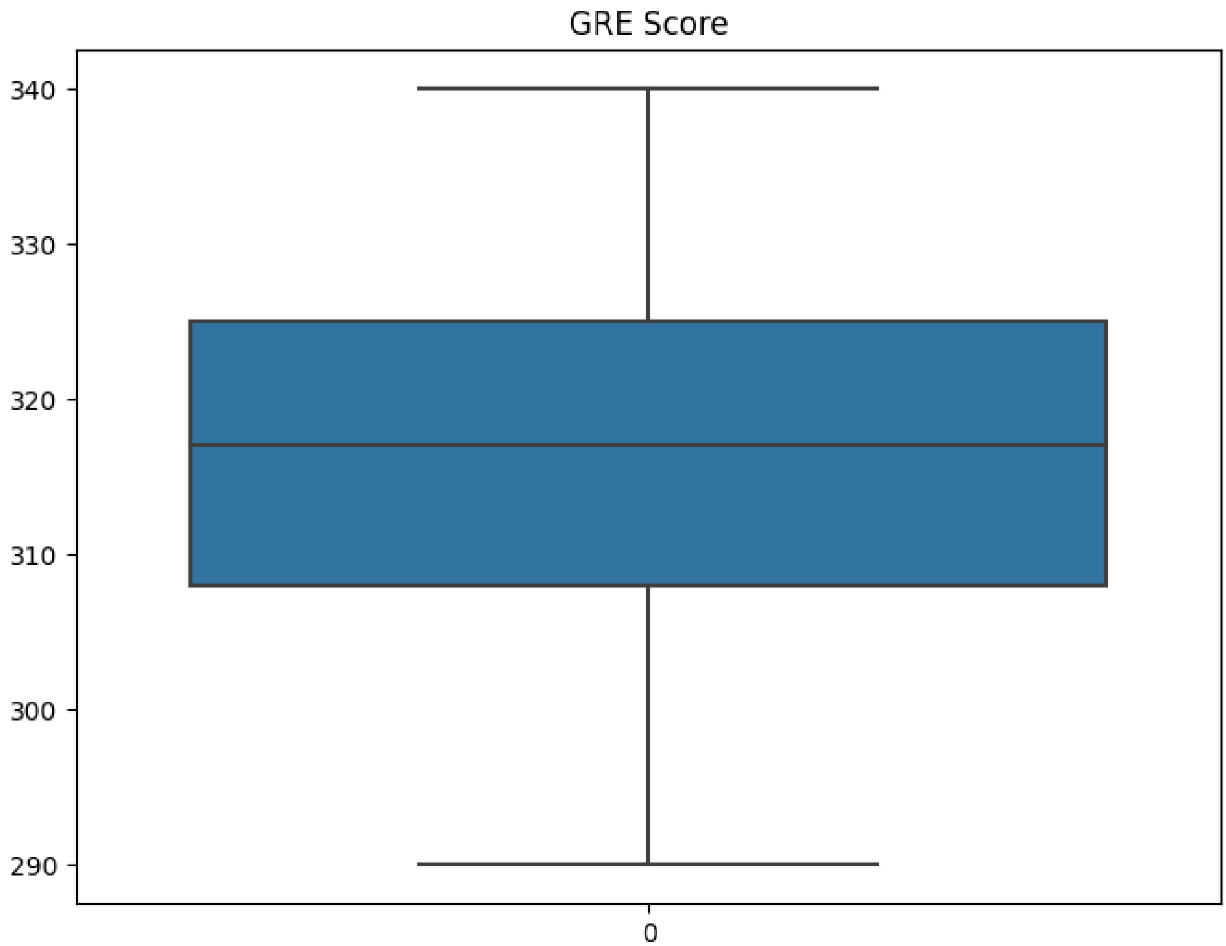
```
TOEFL Score           0.827200    1.000000          0.649799  0.644410
University Rating     0.635376    0.649799          1.000000  0.728024
SOP                   0.613498    0.644410          0.728024  1.000000
LOR                   0.524679    0.541563          0.608651  0.663707
CGPA        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

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

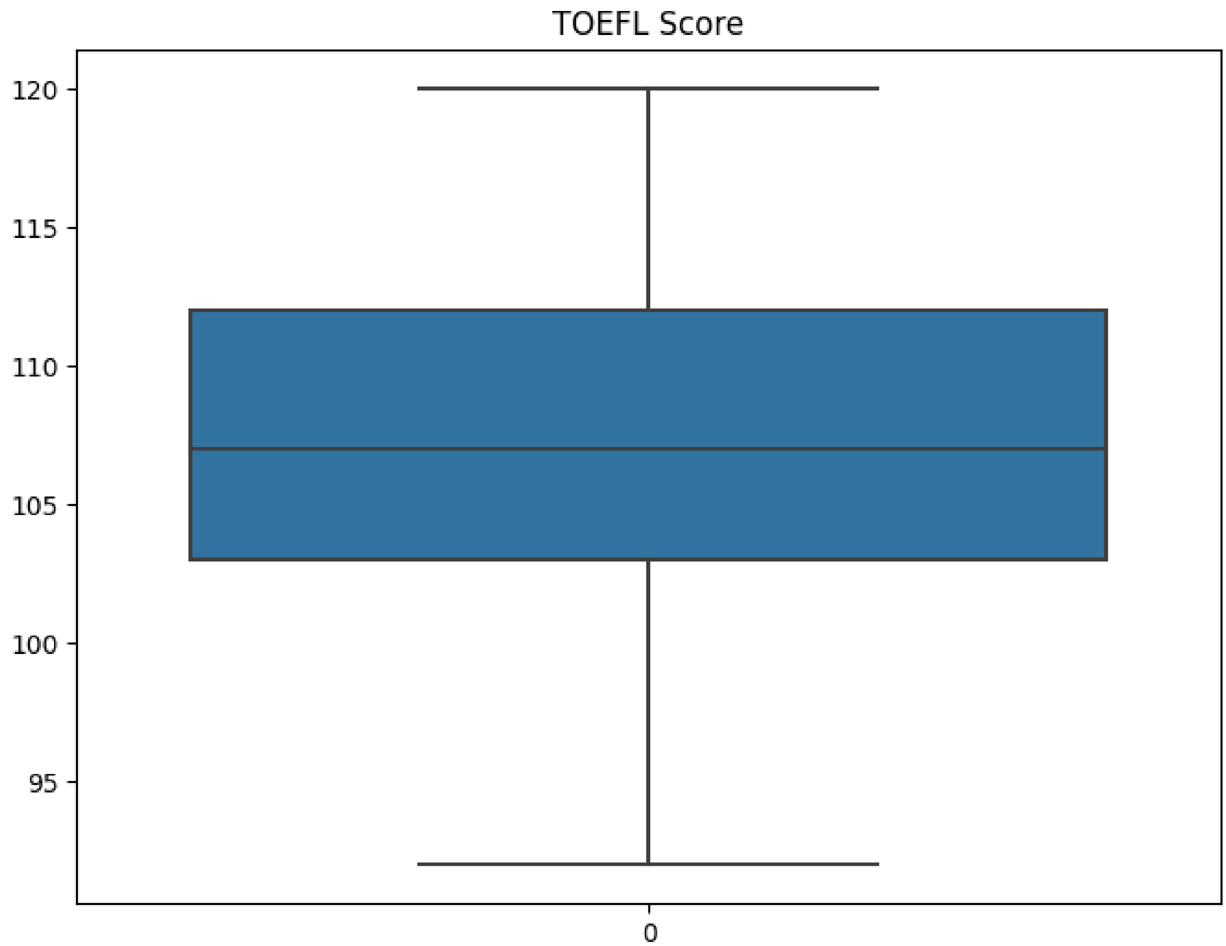
Shows correlation between all features.

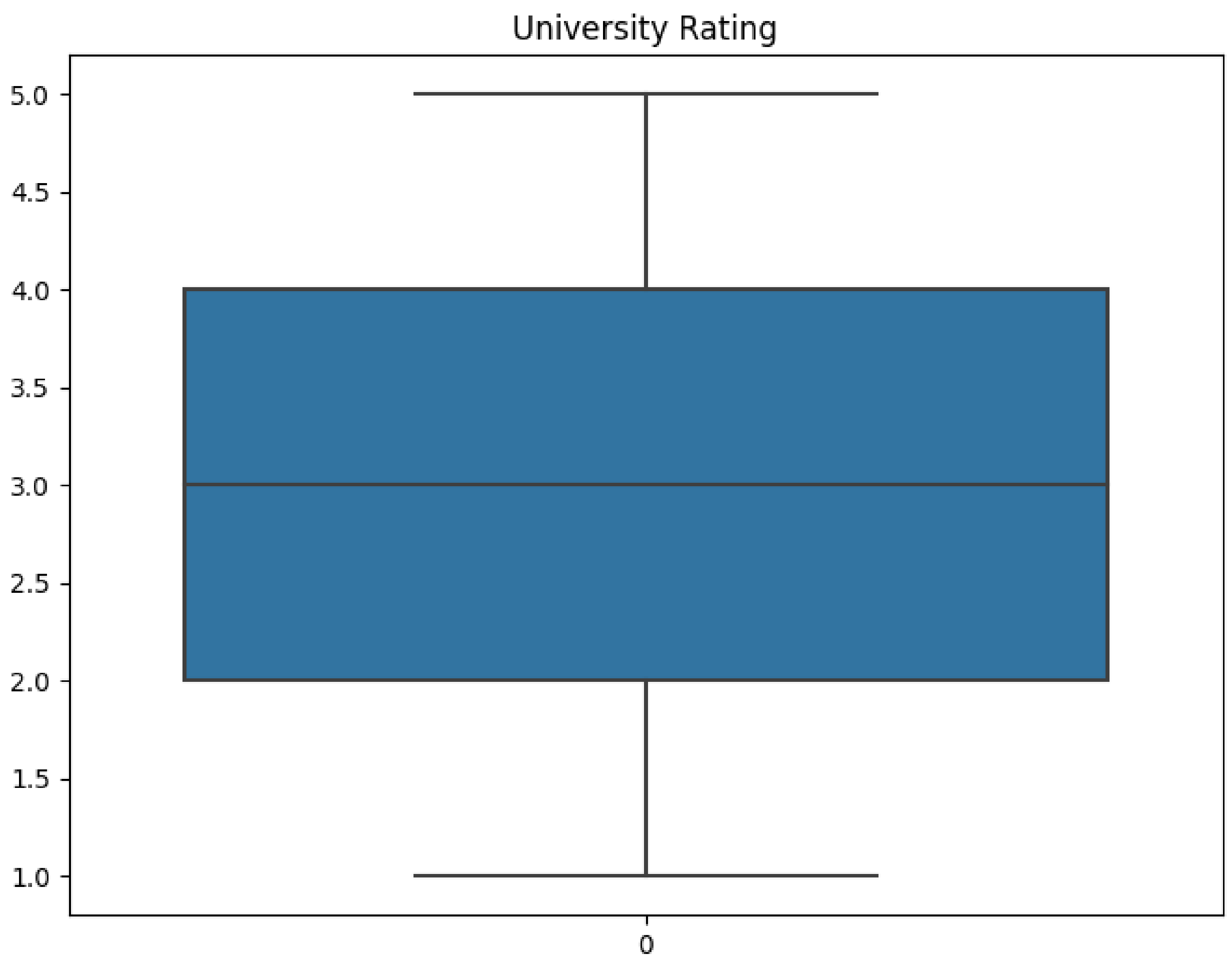
```
plt.figure(figsize = (10, 8))
sns.heatmap(df.corr(),cmap="coolwarm", annot=True)
plt.show()
```

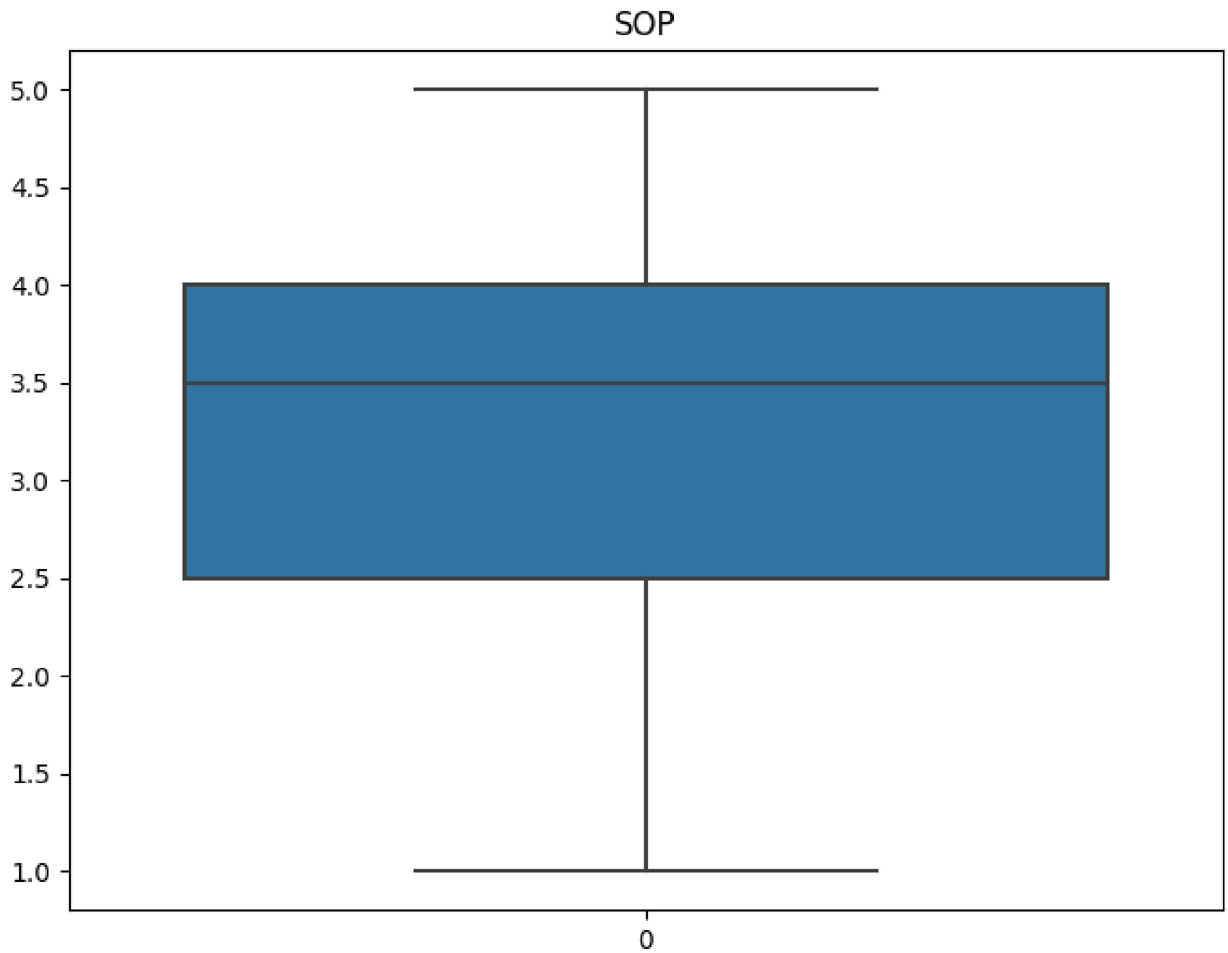


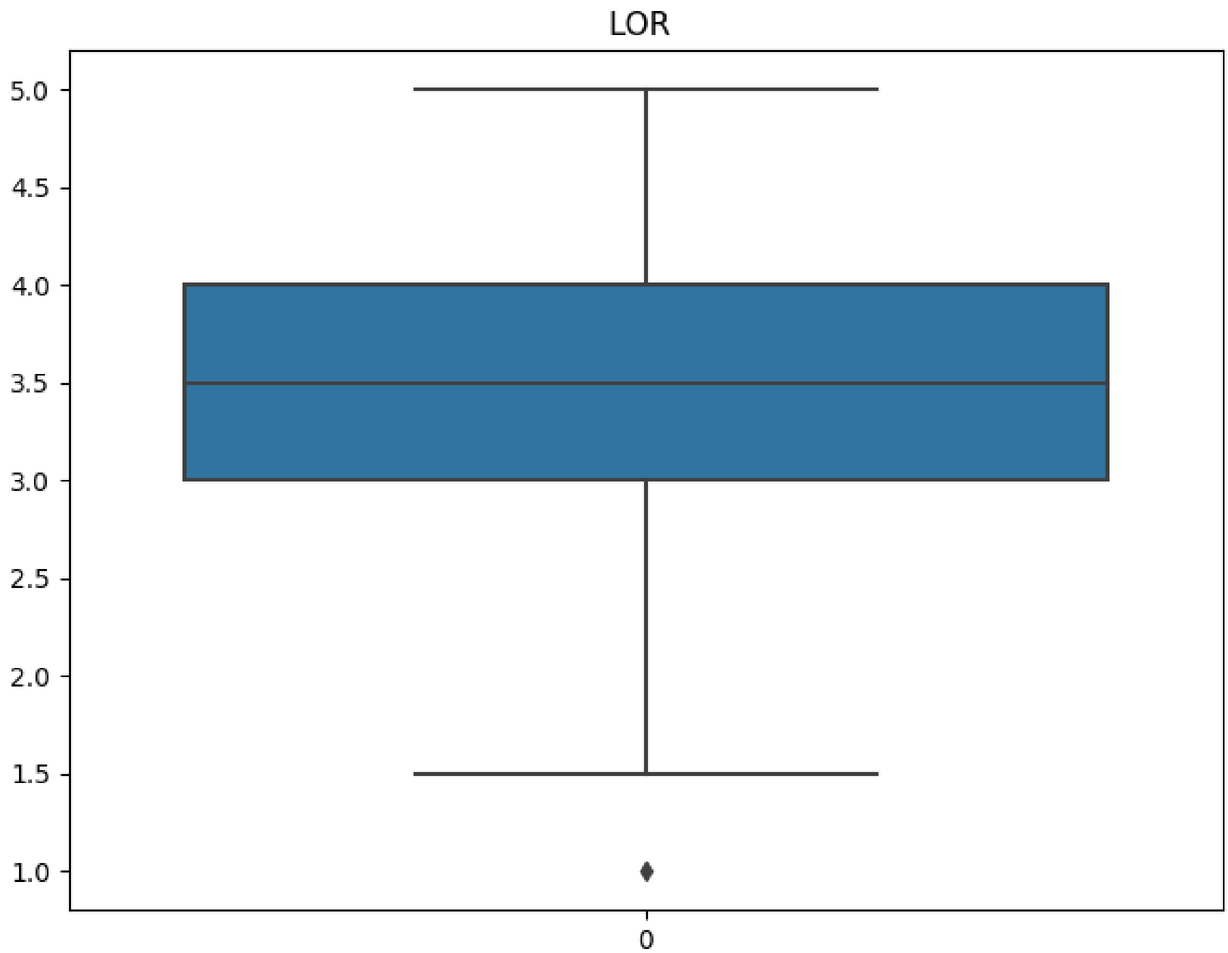

```
for col in ['GRE Score','TOEFL Score','University Rating','SOP','LOR
', 'CGPA', 'Chance of Admit ']:
  plt.figure(figsize = (8, 6))
  sns.boxplot(df[col])
  plt.title(f'{col}')
  plt.show()
```


GRE Score
340
330
320
310
300
290
0

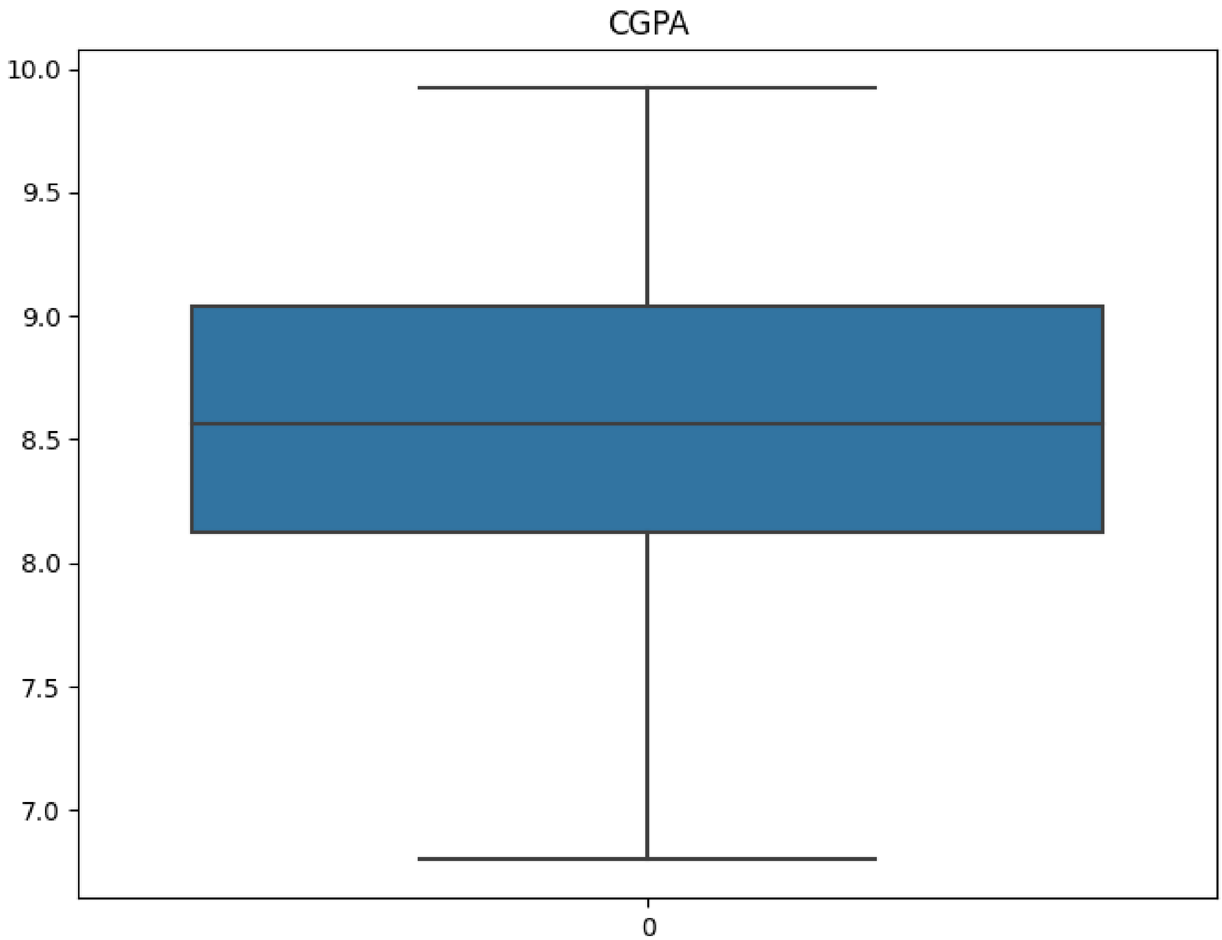

TOEFL Score
120
115
110
105
100
95
0


University Rating
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0

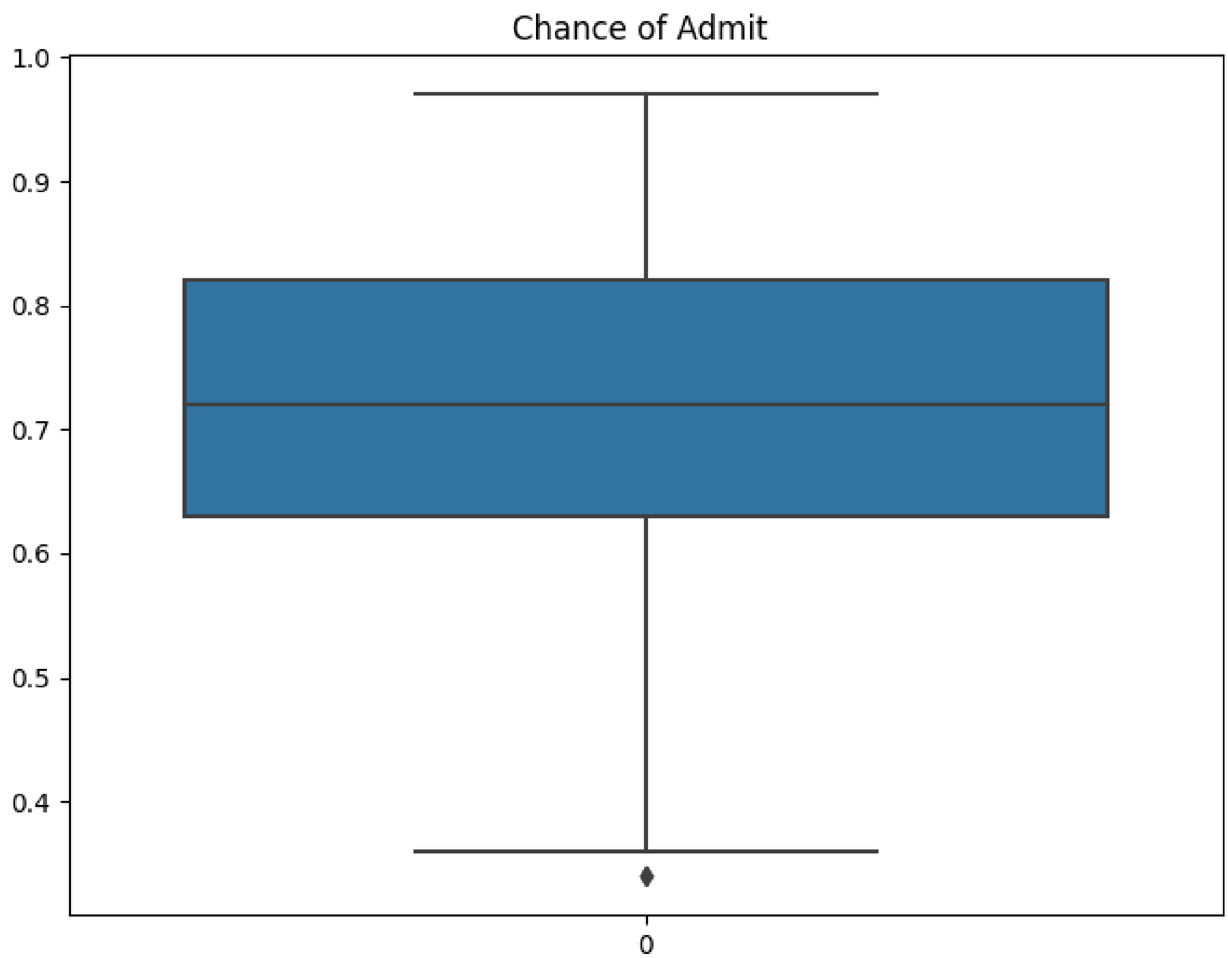

SOP
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0



LOR
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0



CGPA
10.0
9.5
9.0
8.5
8.0
7.5
7.0
0



The box plot for all features show that there are no outliers among any feature.

There is no need for outlier treatment.

Model Building: Linear Regression

```
X = df.drop('Chance of Admit ', axis = 1)
y = df['Chance of Admit ']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state= 2, shuffle = True)
```

Using train and test split to select data for training and testing.

```
X_train
X_col = X_train.columns

y_train.shape
```

```
(400,)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

Scaling the model and fitting the model.

```
X_train = pd.DataFrame(X_train_scaled, columns= X_col)
X_train
```

```
     GRE Score  TOEFL Score  University Rating       SOP       LOR      
CGPA  \
0    -0.040169    -0.679294          -0.966900 -1.396982  1.114121  
0.281025
1    -0.861817    -0.347121          -0.966900 -0.885735  1.114121 -
0.757792
2     0.690184     0.815485           0.775263  0.648005 -1.083899 -
0.791302
3     0.233713     0.483312          -0.095819 -0.374488 -1.083899  
0.364801
4     0.872772     0.151139          -0.095819 -0.374488  0.015111  
0.532352
..         ...          ...                ...       ...       ...      
...
395   1.055361     1.479832           1.646344  1.670499  1.663626  
1.554413
396   0.416301     0.649399           1.646344  1.670499  1.663626  
1.470638
397  -1.500876    -2.007987          -0.966900 -0.374488 -2.182909 -
0.590241
398  -0.222758    -0.347121          -0.095819  0.136759 -1.083899 -
0.456200
399  -2.139935    -1.675814          -0.966900 -1.396982  0.564616 -
1.293955

     Research  
0   -1.128152  
1    0.886405  
2    0.886405  
3   -1.128152  
4   -1.128152  
..        ...  
395  0.886405  
396  0.886405  
397  0.886405  
398 -1.128152  
399  0.886405  

[400 rows x 7 columns]
```

```
X_test_scaled = scaler.transform(X_test)

lr = LinearRegression()
lr.fit(X_train, y_train)
print(f'Train_R2_score: {lr.score(X_train_scaled, y_train)}')
print(f'Test_R2_score: {lr.score(X_test_scaled, y_test)}')

Train_R2_score: 0.829322723369172
Test_R2_score: 0.7927524897595929
```

Linear Regression shows Train R2_score = 0.83 and Test R2_score = 0.80.

The obtained score are almost similar which shows model is working well on both for train and test data but score shows model can be improved more.

```
degree = 2
poly = PolynomialFeatures(degree = degree)
X_train_pol = poly.fit_transform(X_train_scaled)
X_test_pol = poly.fit_transform(X_test_scaled)

poly_lr = LinearRegression()
poly_lr.fit(X_train_pol, y_train)
print("Train r2_score:",poly_lr.score(X_train_pol,y_train))
print("Test r2_score:",poly_lr.score(X_test_pol,y_test))

Train r2_score: 0.8467241531716378
Test r2_score: 0.7901333886055277
```

Using Polynomial Regression to improve the r2 score.

But still scores are almost similar to linear regression.

Some other method can be used to improve the r2_score.

```
X_train.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

model = sm.OLS(y_train, sm.add_constant(X_train)).fit()
```

Using stats model to improve r2_score.

```
print(model.summary())

                            OLS Regression Results

======================================================================
========
Dep. Variable:       Chance of Admit    R-squared:
0.829
Model:                            OLS   Adj. R-squared:
```

```
0.826
Method:                 Least Squares   F-statistic:
272.1
Date:                Sat, 30 Dec 2023   Prob (F-statistic):
3.33e-146
Time:                        14:37:16   Log-Likelihood:
573.41
No. Observations:                 400   AIC:
-1131.
Df Residuals:                     392   BIC:
-1099.
Df Model:                           7

Covariance Type:            nonrobust

======================================================================
===============
                        coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
---------------
const                 0.7221      0.003    247.782      0.000
0.716       0.728
GRE Score             0.0234      0.006      3.893      0.000
0.012       0.035
TOEFL Score           0.0178      0.006      3.024      0.003
0.006       0.029
University Rating     0.0056      0.005      1.185      0.237      -
0.004       0.015
SOP                   0.0020      0.005      0.428      0.669      -
0.007       0.011
LOR                   0.0169      0.004      4.131      0.000
0.009       0.025
CGPA                  0.0677      0.006     10.633      0.000
0.055       0.080
Research              0.0123      0.004      3.476      0.001
0.005       0.019
======================================================================
========
Omnibus:                       94.166   Durbin-Watson:
1.943
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
231.309
Skew:                          -1.158   Prob(JB):
5.92e-51
Kurtosis:                       5.918   Cond. No.
5.47
======================================================================
========
```

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

The summary shows r2 score and adjusted r2_score are similar i.e 0.829 and 0.826

H0: Feature is likely to become zero

Ha: Feature will not become zero

In this summary Univerity rating and SOP has pvalue greater tha 0.05hence they are not likely to become zero which can impact our model hence we will use modelling by removing these features.

```
X_train_d = X_train.drop(columns = 'SOP')
model = sm.OLS(y_train, sm.add_constant(X_train_d)).fit()
print(model.summary())

                            OLS Regression Results

======================================================================
========
Dep. Variable:       Chance of Admit    R-squared:
0.829
Model:                           OLS    Adj. R-squared:
0.827
Method:                Least Squares    F-statistic:
318.1
Date:               Sat, 30 Dec 2023    Prob (F-statistic):
1.96e-147
Time:                       14:37:16    Log-Likelihood:
573.32
No. Observations:                400    AIC:
-1133.
Df Residuals:                    393    BIC:
-1105.
Df Model:                          6

Covariance Type:           nonrobust

======================================================================
===============
                        coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
---------------
const                 0.7221      0.003    248.040      0.000
0.716       0.728
GRE Score             0.0233      0.006      3.883      0.000
0.011       0.035
```

```
TOEFL Score              0.0181      0.006      3.114      0.002
0.007       0.030
University Rating        0.0063      0.004      1.430      0.153      -
0.002       0.015
LOR                      0.0174      0.004      4.452      0.000
0.010       0.025
CGPA                     0.0681      0.006     10.835      0.000
0.056       0.080
Research                 0.0123      0.004      3.483      0.001
0.005       0.019
======================================================================
========
Omnibus:                       92.863   Durbin-Watson:
1.939
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
226.009
Skew:                          -1.146   Prob(JB):
8.37e-50
Kurtosis:                       5.883   Cond. No.
5.05
======================================================================
========

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

```
X_train_d2 = X_train_d.drop(columns = 'University Rating')
model = sm.OLS(y_train, sm.add_constant(X_train_d2)).fit()
print(model.summary())
```

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:       Chance of Admit   R-squared:
0.828
Model:                           OLS   Adj. R-squared:
0.826
Method:                Least Squares   F-statistic:
380.3
Date:               Sat, 30 Dec 2023   Prob (F-statistic):
2.65e-148
Time:                       14:37:17   Log-Likelihood:
572.28
No. Observations:                400   AIC:
-1133.
Df Residuals:                    394   BIC:
-1109.
Df Model:                          5
```

```
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.7221      0.003    247.712      0.000       0.716       0.728
GRE Score      0.0238      0.006      3.970      0.000       0.012       0.036
TOEFL Score    0.0191      0.006      3.298      0.001       0.008       0.030
LOR            0.0190      0.004      5.050      0.000       0.012       0.026
CGPA           0.0702      0.006     11.465      0.000       0.058       0.082
Research       0.0126      0.004      3.573      0.000       0.006       0.020
==============================================================================
Omnibus:                       90.913   Durbin-Watson:                   1.941
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              218.765
Skew:                          -1.127   Prob(JB):                     3.13e-48
Kurtosis:                       5.837   Cond. No.                         4.63
==============================================================================

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

Summary shows still no effect on r2_score which indicates some different method might improve the score i.e imoroving the complexity of model.

```
ridge_model = Ridge(alpha=0.1)
ridge_model.fit(X_train_pol, y_train)
print(f'Ridge Coefficients: {ridge_model.coef_}')

print("*"*50)
print("Degree: ", degree)
print("Train r2_score:",ridge_model.score(X_train_pol,y_train))
print("Test r2_score:",ridge_model.score(X_test_pol,y_test))
```

```
Ridge Coefficients: [ 0.          0.01690838  0.02074965  0.0072882
0.0059243   0.01569447
  0.06390535  0.01197212 -0.00105736 -0.01266717  0.00257734 -
0.00021989
  0.0156172  -0.0023023  -0.00488274  0.00122205  0.00790986
0.01389655
 -0.01196011 -0.00418238  0.00501263 -0.00228258  0.02676843 -
0.00457004
 -0.0121225   0.0043675  -0.01542188  0.00272575 -0.00067111 -
0.00113057
  0.00342725 -0.01268339 -0.00465992  0.00765165  0.00469252 -
0.00289422]
**************************************************
Degree:  2
Train r2_score: 0.8467243297179153
Test r2_score: 0.7901645207851495
```

Using ridge method to improve the model.

The score shows same value as previous model.

```
lasso_model = Lasso(alpha=0.1)
lasso_model.fit(X_train_pol, y_train)
print(f'Ridge Coefficients: {ridge_model.coef_}')

print("*"*50)
print("Degree: ", degree)
print("Train r2_score:",lasso_model.score(X_train_pol,y_train))
print("Test r2_score:",lasso_model.score(X_test_pol,y_test))
```

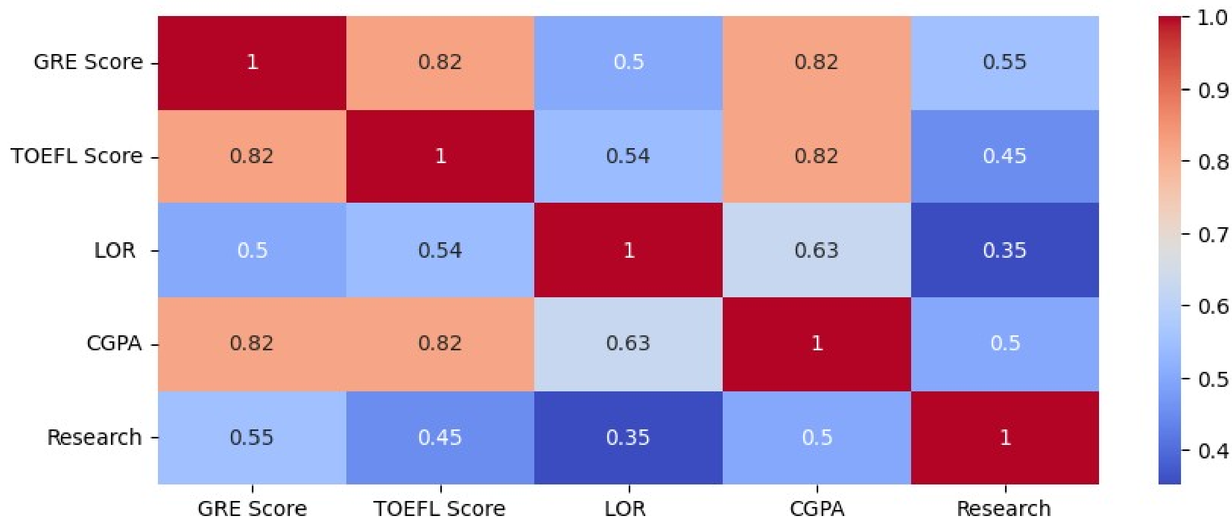
```
Ridge Coefficients: [ 0.          0.01690838  0.02074965  0.0072882
0.0059243   0.01569447
  0.06390535  0.01197212 -0.00105736 -0.01266717  0.00257734 -
0.00021989
  0.0156172  -0.0023023  -0.00488274  0.00122205  0.00790986
0.01389655
 -0.01196011 -0.00418238  0.00501263 -0.00228258  0.02676843 -
0.00457004
 -0.0121225   0.0043675  -0.01542188  0.00272575 -0.00067111 -
0.00113057
  0.00342725 -0.01268339 -0.00465992  0.00765165  0.00469252 -
0.00289422]
**************************************************
Degree:  2
Train r2_score: 0.2687752696330791
Test r2_score: 0.26929314054867604
```

This model gives bad score so cannot be used.

Test of the Assumptions

Assumption 1 : Multicollinearity

```
plt.figure(figsize=(10,4))
sns.heatmap(X_train_d2.corr(),cmap="coolwarm", annot=True)
plt.show()
```



Shows high collinearity among features.

Assumption 2 : VIF

```
X_train_d2
```

	GRE Score	TOEFL Score	LOR	CGPA	Research
0	-0.040169	-0.679294	1.114121	0.281025	-1.128152
1	-0.861817	-0.347121	1.114121	-0.757792	0.886405
2	0.690184	0.815485	-1.083899	-0.791302	0.886405
3	0.233713	0.483312	-1.083899	0.364801	-1.128152
4	0.872772	0.151139	0.015111	0.532352	-1.128152
..	...	...	...	...	...
395	1.055361	1.479832	1.663626	1.554413	0.886405
396	0.416301	0.649399	1.663626	1.470638	0.886405
397	-1.500876	-2.007987	-2.182909	-0.590241	0.886405
398	-0.222758	-0.347121	-1.083899	-0.456200	-1.128152
399	-2.139935	-1.675814	0.564616	-1.293955	0.886405

```
[400 rows x 5 columns]
```

```
vif = pd.DataFrame()
X_t = pd.DataFrame(X_train_d2, columns=X_train_d2.columns)
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in
range(X_t.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by="VIF", ascending=False)
vif
```

```
      Features   VIF
3         CGPA  4.41
0    GRE Score  4.22
1  TOEFL Score  3.94
2          LOR   1.67
4     Research  1.46
```

Here VIF value is <5 for all fetures.

Obtained VIF shows there is no Variance Inflation Factor

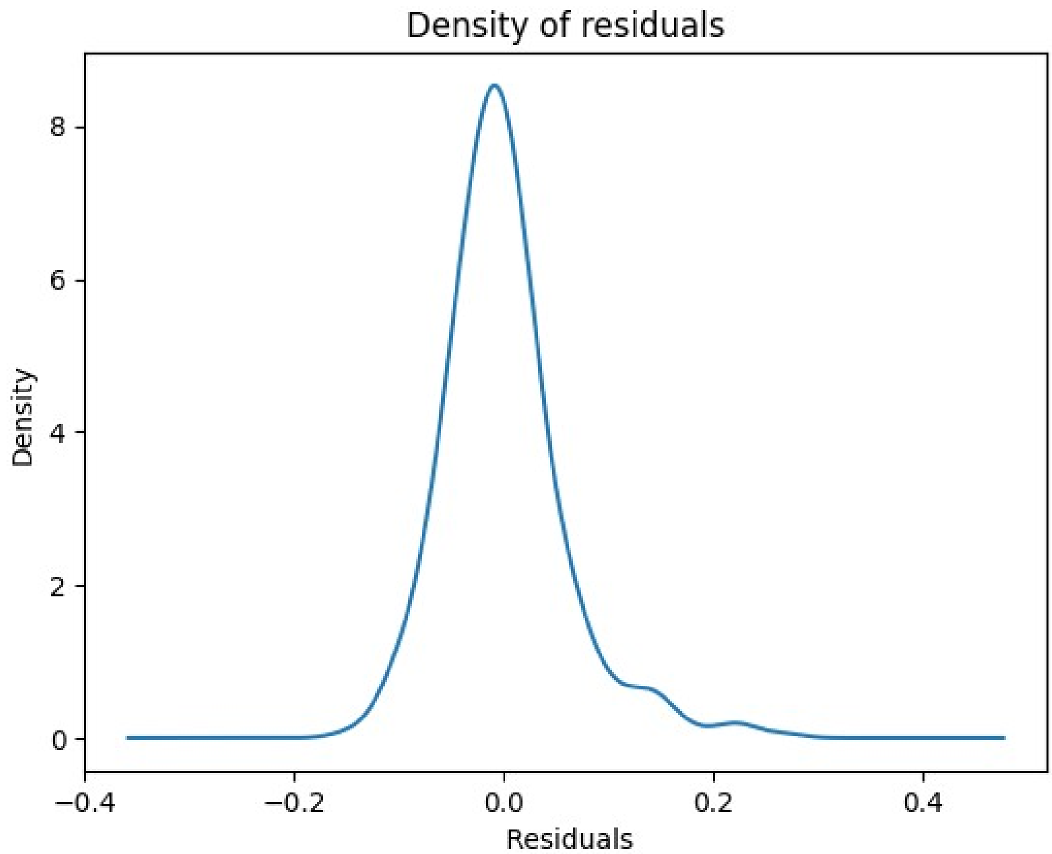
No VIF assumption hold true.

Assumption 3: Residual Error(Error must be normally distributed)

```
X_sm = sm.add_constant(X_train_d2)
sm_model = sm.OLS(y_train, X_sm).fit()

y_train.shape
```

```
(400,)
```

```
Y_hat = sm_model.predict(X_sm)
errors = Y_hat - y_train

errors.plot(kind = 'kde')
plt.xlabel(" Residuals")
plt.title("Density of residuals")
plt.show()
```



Errors are normally distributed.

Graph shows errors are centred around zero which is mean value and also less variance.

```
from scipy import stats
res = stats.shapiro(errors)
res.statistic
```

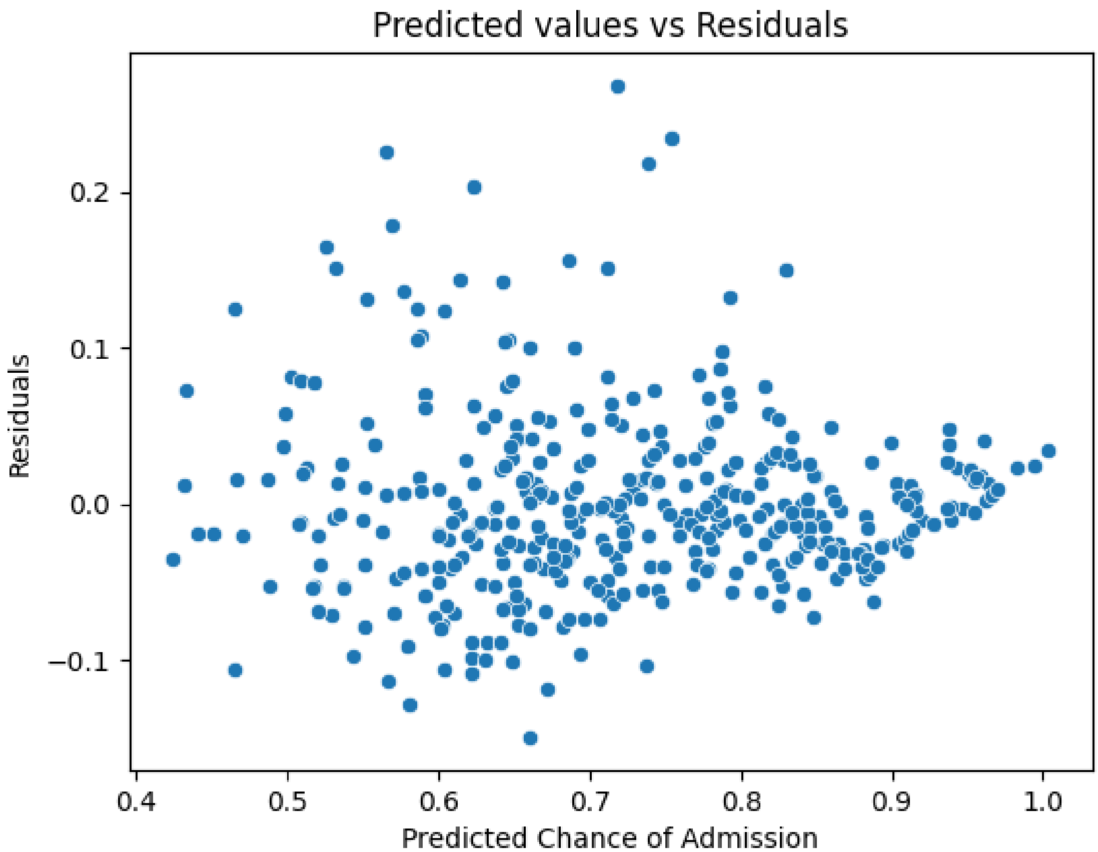
```
0.9354482293128967
```

Test for normality of distribution obtained value is 0.935 which shows graph is normally distributed.

Assumption 4: Homoskedacity

```
sns.scatterplot(x=Y_hat,y=errors)
plt.xlabel("Predicted Chance of Admission")
plt.ylabel("Residuals")
plt.title("Predicted values vs Residuals")
```

```
Text(0.5, 1.0, 'Predicted values vs Residuals')
```



H0: Homoskedacity exist

Ha: Model is heteroskedic

```
from statsmodels.compat import lzip
import statsmodels.stats.api as sms

name = ['F statistic', 'p-value']
test = sms.het_goldfeldquandt(y_train, X_sm)
lzip(name, test)

[('F statistic', 1.0695200311468724), ('p-value',
0.32010091217135295)]
```

From the goldfeld-quandt test:

F Statistic comes out to be 1.06 => Implying minimal difference in variance between groups.

p-value of 0.320 indicates that this difference is statistically significant at conventional levels of significance (e.g., 0.05).

Therefore, we accept the null hypothesis of homoscedasticity, and conclude that there is no strong evidence of heteroscedasticity in the data.

```
X_test_scaled = pd.DataFrame(X_test_scaled, columns = X_col)
X_test_scaled = sm.add_constant(X_test_scaled)
X_test_scaled = X_test_scaled.drop(columns = ['University Rating',
'SOP'])
X_test_scaled
```

```
    const  GRE Score  TOEFL Score       LOR      CGPA  Research
0     1.0   1.511831     1.812005  1.663626  1.303087  0.886405
1     1.0  -0.496640    -0.845381  0.564616  0.113474  0.886405
2     1.0  -1.044405    -0.513208 -2.182909 -1.310710 -1.128152
3     1.0   2.150891     0.981572  1.663626  1.956536  0.886405
4     1.0  -1.866053    -1.343641 -1.083899 -0.908588 -1.128152
..    ...        ...          ...       ...       ...       ...
95    1.0  -0.861817    -0.347121  0.015111 -0.791302 -1.128152
96    1.0  -2.413818    -0.513208 -1.083899 -1.863629 -1.128152
97    1.0   0.598890     0.483312  1.663626  0.515597  0.886405
98    1.0   0.690184     0.483312  0.015111  0.783678  0.886405
99    1.0  -0.131464    -0.347121 -1.083899 -0.154608 -1.128152

[100 rows x 6 columns]
```

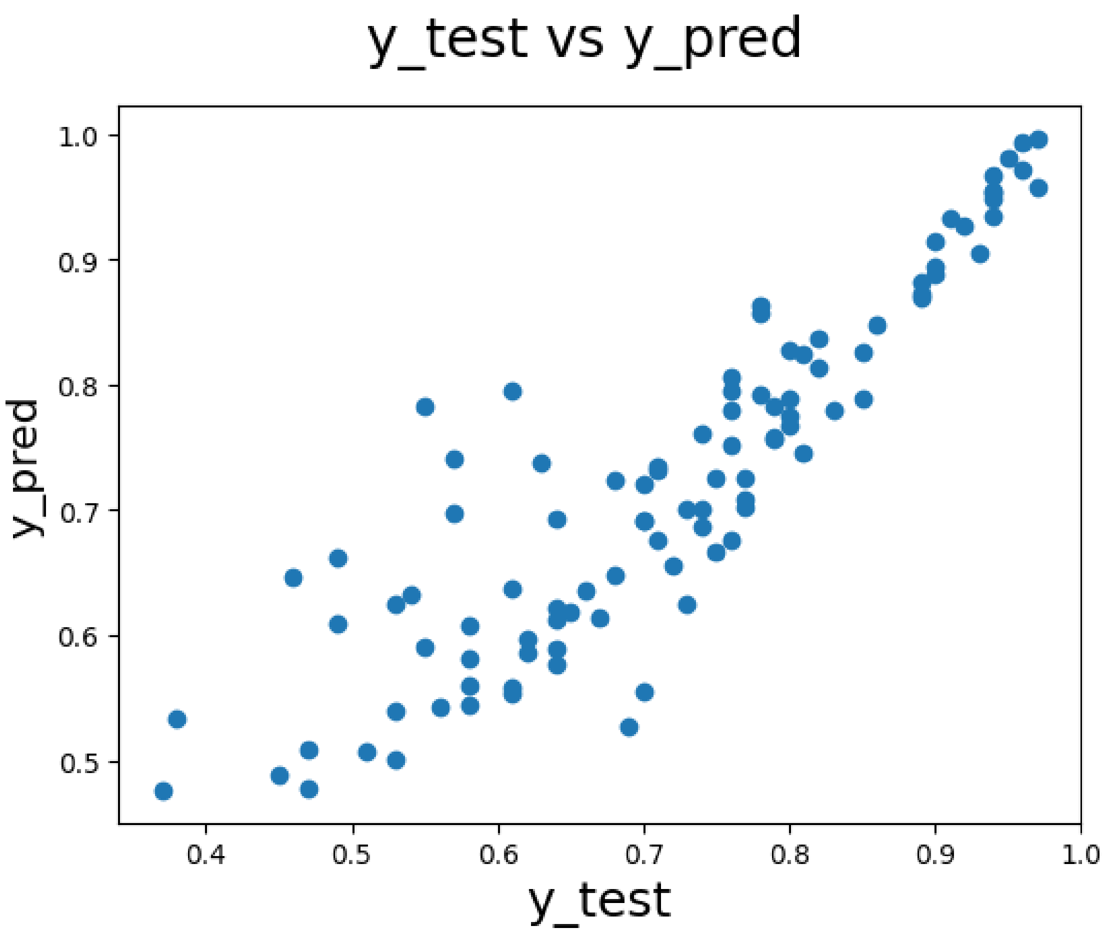
```
y_pred = model.predict(X_test_scaled)
print(f'MAE: {mean_absolute_error(y_test, y_pred)}')
print(f'RMSE: {np.sqrt(mean_squared_error(y_test, y_pred))}')
print(f'R2 Score: {r2_score(y_test, y_pred)}')
print(f'Adjusted R2 Score: {1 - (1-r2_score(y_test,
y_pred))*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1)}')
```

```
MAE: 0.047347725647416475
RMSE: 0.06690331530026929
R2 Score: 0.790564197286199
Adjusted R2 Score: 0.7746288644710184
```

Mean absolute error and root mean square error is close to zero.

R2_score is 0.79 on test data shows model can be improved further.

```
fig = plt.figure()
plt.scatter(y_test.values, y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20) # Plot head
plt.xlabel('y_test', fontsize=18) # X-label
plt.ylabel('y_pred', fontsize=16)
plt.show()
```




```
print(model.params[:-1])
```

```
const           0.722125
GRE Score       0.023782
TOEFL Score     0.019073
LOR             0.019006
CGPA            0.070156
dtype: float64
```

Weights and constant value.

Actionable Insights

1. Data is simple with no outliers.

2. Features are correlated may require some feature engineering.

3. GRE Scores and TOEFL Scores seem to have a strong positive correlation with the Chance of Admit.

4. Research Experience might be a significant predictor, but it depends on the coefficient and p-value.

5. There is a strong variance in test and predcited value when prediction value(Chance of Admit) is low but variance reduces when value predicted is high.

Recommendations:

1. More complex method required to improve the model.

2. Hyperparameter and regularization method can be used to improve the score.

3. Some other modelling method can be used to improve the model.

4. Cross validation method can be applied.