Problem statement: With the help of linear regression we are going to help Jamboree in understanding what factors are important in graduate admissions and how these factors are interrelated among themselves. It will also help predict one's chances of admission given the rest of the variables. Using linear regression we can calculate the probability of one getting into the IVY league colleges by using the independent variables as predictors and using modelling.

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
from scipy import stats
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from statsmodels.graphics.gofplots import qqplot
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import statsmodels.stats.api as sms
```

In [3]:
```
jamboree = pd.read_csv('C:/DSML/Jamboree - case study/Jamboree_Admission.csv')
```

In [4]:
```
jamboree.shape
```

Out[4]: (500, 9)

In [5]:
```
jamboree.head()
```

Out[5]:

	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

In [6]:
```
jamboree.columns
```

Out[6]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

```
Basic metrics : We have a total of 500 rows which represent the no. of
students and 9 predictors which we are going to analyze further to see which
can be used for our modelling
```

We are going to split the 500 rows into training and test data to train our model and once the model is trained, test data will be used to check the performance of our model. Whether or not we need to use all 9 atttributes as predictors will be decided further.

Missing value detection

In [7]:
```
jamboree.isna().sum()
```

Out[7]:
```
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
```

Since there are no missing values, we do not need any further action

Statistical summary

In [8]:
```
jamboree.describe()
```

Out[8]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Re:
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.(
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.5
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.4
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.(
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.(
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.(
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.(
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.(

Since we can conclude that the serial no has actual contribution to the prediction of chance of admit, we can drop the column before proceeding further

In [9]:
```
jamboree = jamboree.drop('Serial No.',axis=1)
```

In [10]:
```
jamboree.describe()
```

Out[10]:

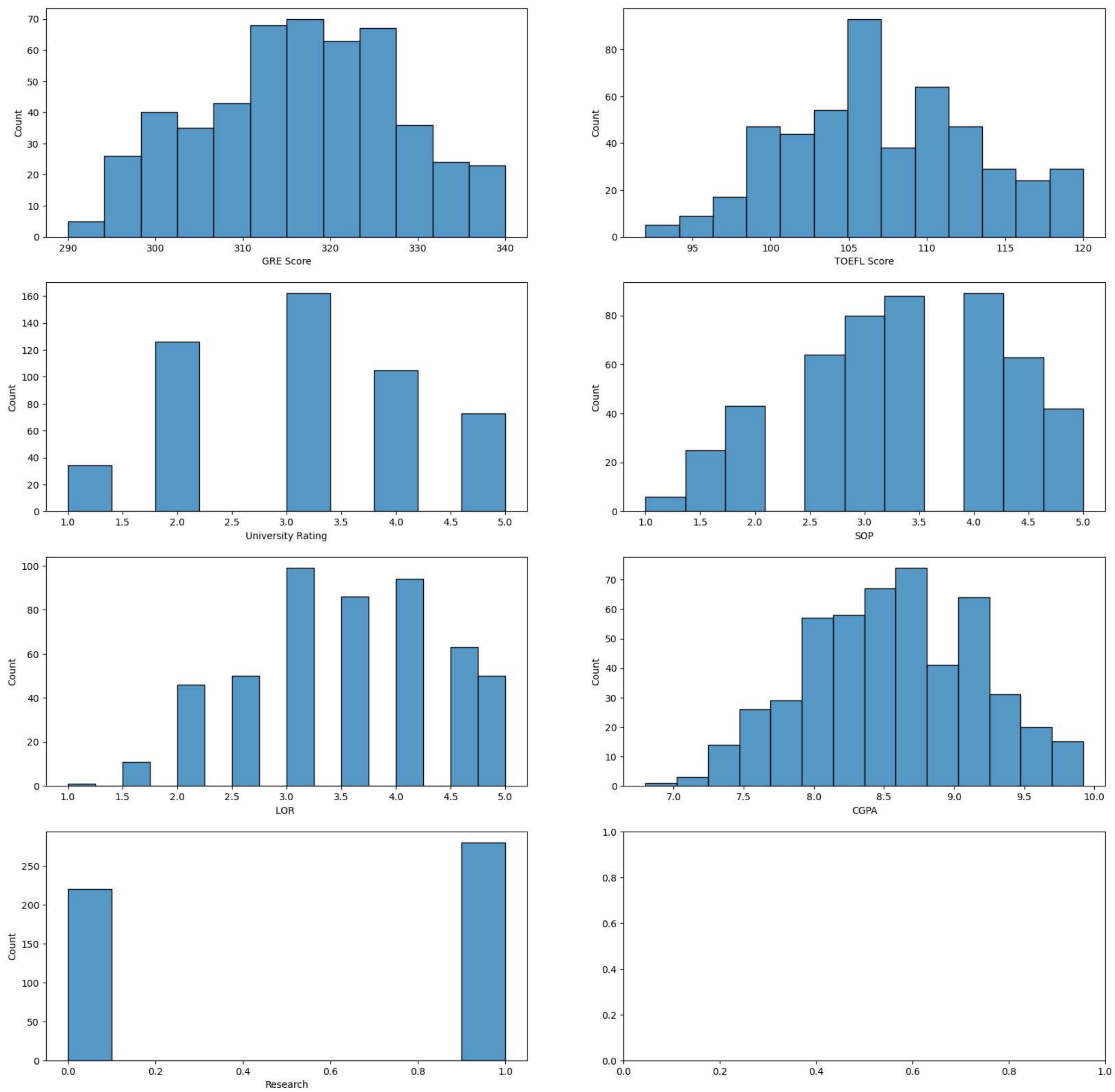
	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Ch of /
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.(
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.7
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.'
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.3
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.6
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.7
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.8
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.9

Visual Analysis

Univariate Analysis

In [11]:
```python
# Hist plot for continuous variables to visualize their distribution
fig, axis = plt.subplots(nrows=4, ncols=2, figsize=(20, 20))
sns.histplot(x = 'GRE Score',data = jamboree,ax=axis[0,0])
sns.histplot(x = 'TOEFL Score',data = jamboree,ax=axis[0,1])
sns.histplot(x = 'University Rating',data = jamboree,ax=axis[1,0])
sns.histplot(x = 'SOP',data = jamboree,ax=axis[1,1])
sns.histplot(x = 'LOR ',data = jamboree,ax=axis[2,0])
sns.histplot(x = 'CGPA',data = jamboree,ax=axis[2,1])
sns.histplot(x = 'Research',data = jamboree,ax=axis[3,0])
```

Out[11]: <AxesSubplot:xlabel='Research', ylabel='Count'>



Observations based on the univariate analysis of the continuous variables

1. GRE Score, TOEFL Score, University Ranking and CGPA follows almost a normal distribution
2. LOR distribution is skewed to the left
3. CGPA also follows almost a normal distribution with some skewness to the lest

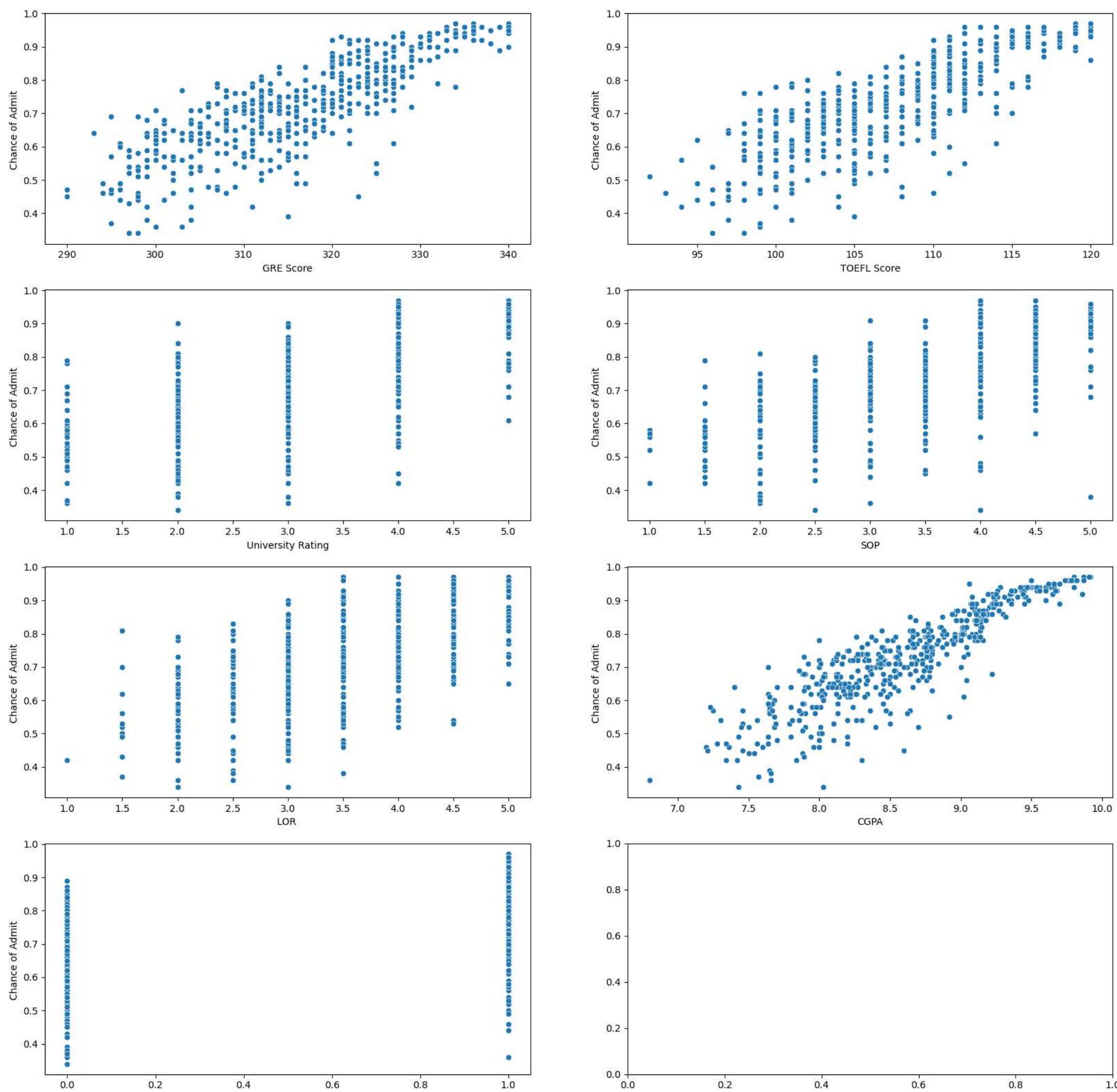
```
4. Since research only 2 values, it does not have a normal distribution
```

Bivariate Analysis

In [12]:

```python
fig, axis = plt.subplots(nrows=4, ncols=2, figsize=(20, 20))
sns.scatterplot(x = 'GRE Score',y = 'Chance of Admit ',data = jamboree,ax=axis
sns.scatterplot(x = 'TOEFL Score',y = 'Chance of Admit ',data = jamboree,ax=ax
sns.scatterplot(x = 'University Rating',y = 'Chance of Admit ',data = jamboree
sns.scatterplot(x = 'SOP',y = 'Chance of Admit ',data = jamboree,ax=axis[1,1])
sns.scatterplot(x = 'LOR ',y = 'Chance of Admit ',data = jamboree,ax=axis[2,0]
sns.scatterplot(x = 'CGPA',y = 'Chance of Admit ',data = jamboree,ax=axis[2,1]
sns.scatterplot(x = 'Research',y = 'Chance of Admit ',data = jamboree,ax=axis[
```

Out[12]: <AxesSubplot:xlabel='Research', ylabel='Chance of Admit '>

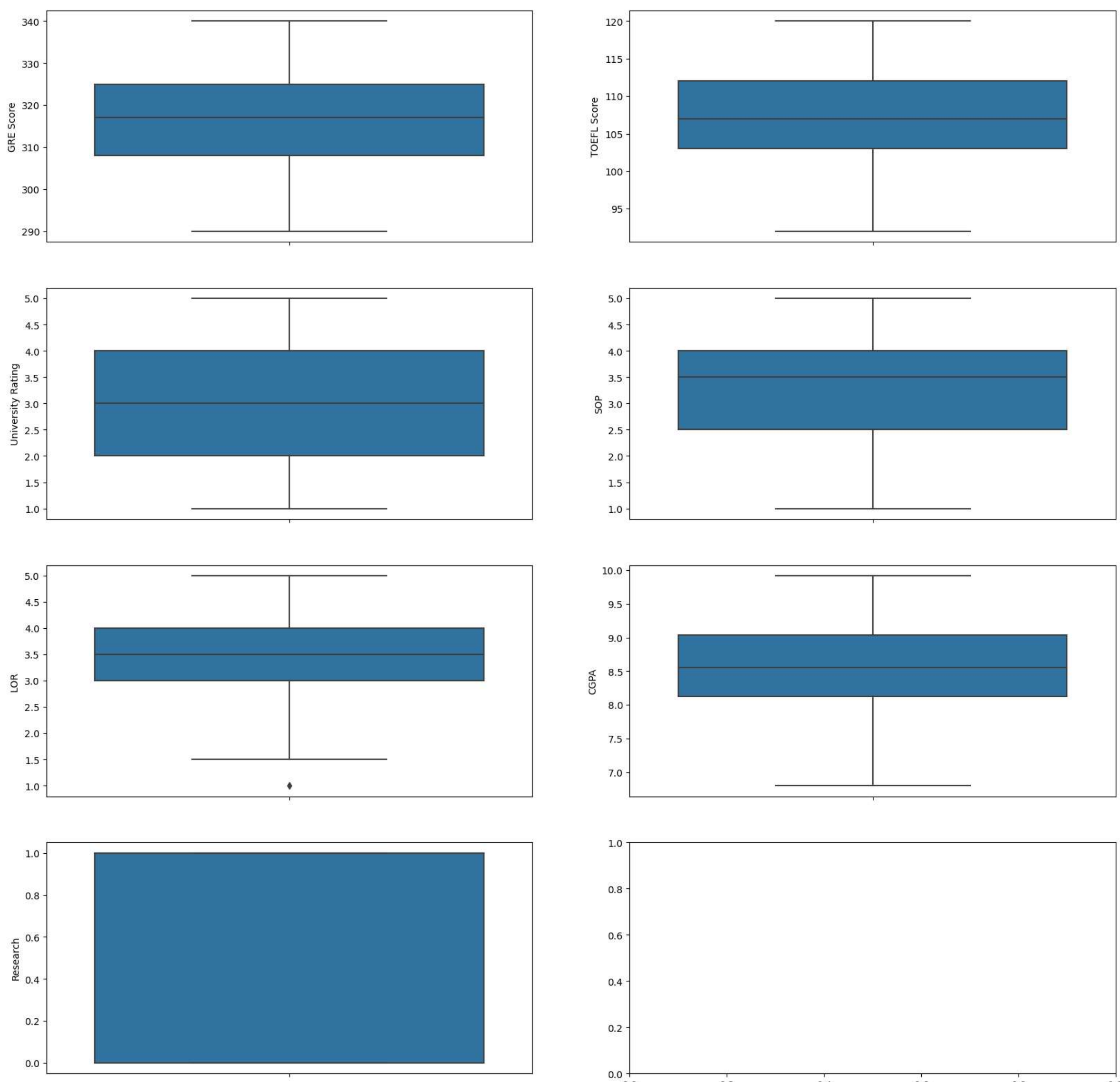



```
Observations based on the bivariate analysis of the continuous variables
```

1. As we can observe, GRE Score, TOEFL Score and CGPA have a linear relationship with the Chance of Admit. Although GRE score and TOEFL score are more scattered, CGPA has a much more more linear relationship with the Chance of Admit.
2. University rating, SOP and LOR have somewhat a linear relationship but very much scattered

In [13]:
```
fig, axis = plt.subplots(nrows=4, ncols=2, figsize=(20, 20))
sns.boxplot(y = 'GRE Score',data = jamboree,ax=axis[0,0])
sns.boxplot(y = 'TOEFL Score',data = jamboree,ax=axis[0,1])
sns.boxplot(y = 'University Rating',data = jamboree,ax=axis[1,0])
sns.boxplot(y = 'SOP',data = jamboree,ax=axis[1,1])
sns.boxplot(y = 'LOR ',data = jamboree,ax=axis[2,0])
sns.boxplot(y = 'CGPA',data = jamboree,ax=axis[2,1])
sns.boxplot(y = 'Research',data = jamboree,ax=axis[3,0])
```

Out[13]: <AxesSubplot:ylabel='Research'>



```
None of the predictor variables have any outliers and hence we do not need
any outlier treatment for the dataset. Though there is an outlier for LOR,
it is very minor and can be ignored.
```

Model building

```
Before we can build the model, we will have to split the dataset into
training and test data. We are going to use the training data to build and
check the performance of the model and the test data will be used to test
the performance of the model
```

In [14]:
```
X = jamboree[jamboree.columns.drop('Chance of Admit ')]
Y = jamboree["Chance of Admit "]
```

In [15]:
```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, shuf
```

In [16]:
```
model = LinearRegression()
model.fit(x_train,y_train)
Y_hat = model.predict(x_train)
```

In [17]:
```
model.intercept_
```

Out[17]:
```
-1.2594324782480177
```

In [18]:
```
model.coef_
```

Out[18]:
```
array([ 0.00173741,  0.00291958,  0.00571666, -0.00330517,  0.02235313,
        0.11893945,  0.02452511])
```

In [19]:
```
model.score(x_train,y_train)
```

Out[19]:
```
0.8034713719824393
```

Linear Regression using statsmodel

In [20]:
```
import statsmodels.api as sm
X_sm = sm.add_constant(x_train)

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

In [21]:
```
print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:      Chance of Admit   R-squared:                       0.80
3
Model:                          OLS   Adj. R-squared:                  0.80
0
Method:               Least Squares   F-statistic:                     228.
9
Date:              Mon, 12 Jun 2023   Prob (F-statistic):          3.12e-13
4
Time:                      00:40:56   Log-Likelihood:                 537.3
7
No. Observations:               400   AIC:                            -105
9.
Df Residuals:                   392   BIC:                            -102
7.
Df Model:                         7
Covariance Type:          nonrobust
==============================================================================
========
                        coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const                -1.2594      0.125    -10.097      0.000      -1.505
-1.014
GRE Score             0.0017      0.001      2.906      0.004       0.001
0.003
TOEFL Score           0.0029      0.001      2.680      0.008       0.001
0.005
University Rating     0.0057      0.005      1.198      0.232      -0.004
0.015
SOP                  -0.0033      0.006     -0.594      0.553      -0.014
0.008
LOR                   0.0224      0.006      4.034      0.000       0.011
0.033
CGPA                  0.1189      0.012      9.734      0.000       0.095
0.143
Research              0.0245      0.008      3.081      0.002       0.009
0.040
==============================================================================
=
Omnibus:                     87.895   Durbin-Watson:                   0.75
9
Prob(Omnibus):                0.000   Jarque-Bera (JB):              181.19
1
Skew:                        -1.159   Prob(JB):                      4.52e-4
0
Kurtosis:                     5.344   Cond. No.                      1.31e+0
4
==============================================================================
=

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

```
[2] The condition number is large, 1.31e+04. This might indicate that there a
re
strong multicollinearity or other numerical problems.
```

```
Model statistics :
1. The trained model has a R-squared value of 0.803 and an adjusted R-
squared value of 0.8. This indicates a good performance of the model
2. the constant or the w0 value is -1.2594
```

```
Coeff with column names :
The weights or the co-efficients of the columns as indicated from the
summary are as below
GRE Score              0.0017
TOEFL Score            0.0029
University Rating      0.0057
SOP                   -0.0033
LOR                    0.0224
CGPA                   0.1189
Research               0.0245
CGPA has the highest weightage and hence the most important predictor
compared to other
This is followed by LOR and Research
The feature with the lowest importance seems to be GRE Score followed by
TOEFL score indicating that these scores don't play a big part in a
candidate's Chance of Admit
```

Testing the assumptions of the linear regression model

Multicollinearity check by VIF score

In [22]:
```
# VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [23]:
```
vif = pd.DataFrame()
X_t = x_train
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shap
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[23]:

	Features	VIF
0	GRE Score	1438.45
1	TOEFL Score	1349.75
5	CGPA	1080.49
4	LOR	38.41
3	SOP	38.05
2	University Rating	22.14
6	Research	2.86

```
As we can see above, GRE Score and TOEFL Score have very high VIF values.
Typically the recommended VIF score should be less than 5. Hence we are
first going to eliminate GRE score since it has the highest VIF score and
check the performance of the model
```

In [24]:
```
X_new = x_train.drop(columns=['GRE Score'])
```

In [25]:
```
X2_sm = sm.add_constant(X_new)

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

In [26]:
```
print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:      Chance of Admit   R-squared:                       0.79
9
Model:                          OLS   Adj. R-squared:                  0.79
6
Method:               Least Squares   F-statistic:                     260.
8
Date:              Mon, 12 Jun 2023   Prob (F-statistic):          1.19e-13
3
Time:                      00:40:56   Log-Likelihood:                 533.1
1
No. Observations:               400   AIC:                            -105
2.
Df Residuals:                   393   BIC:                            -102
4.
Df Model:                         6
Covariance Type:          nonrobust
==============================================================================
========
                        coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const                -0.9804      0.080    -12.202      0.000      -1.138
-0.822
TOEFL Score           0.0044      0.001      4.443      0.000       0.002
0.006
University Rating     0.0064      0.005      1.331      0.184      -0.003
0.016
SOP                  -0.0045      0.006     -0.811      0.418      -0.016
0.006
LOR                   0.0222      0.006      3.961      0.000       0.011
0.033
CGPA                  0.1325      0.011     11.614      0.000       0.110
0.155
Research              0.0313      0.008      4.072      0.000       0.016
0.046
==============================================================================
=
Omnibus:                     79.400   Durbin-Watson:                   0.77
3
Prob(Omnibus):                0.000   Jarque-Bera (JB):              147.28
2
Skew:                        -1.103   Prob(JB):                      1.04e-3
2
Kurtosis:                     4.993   Cond. No.                      2.70e+0
3
==============================================================================
=

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
[2] The condition number is large, 2.7e+03. This might indicate that there ar
```

```
e
strong multicollinearity or other numerical problems.
```

The model performance(R-squared and adjusted r-squared) has reduced very minutely and hence we can continue with out iterations to eliminate predictors with high VIF values

In [27]:

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

Out[27]:

	Features	VIF
4	CGPA	829.46
0	TOEFL Score	731.85
3	LOR	38.36
2	SOP	36.47
1	University Rating	20.27
5	Research	2.86

Now, as we can observe the CGPA now has the highest VIF factor. Let us drop the CGPA feature and test the performance of our model

In [28]:

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

In [29]:

```
vif = pd.DataFrame()
X_t = X_new1
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shap
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[29]:

	Features	VIF
3	LOR	36.63
2	SOP	36.34
0	TOEFL Score	21.87
1	University Rating	20.07
4	Research	2.85

In [30]:
```
X2_sm = sm.add_constant(X_new1)  #Statmodels default is without intercept, to

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

print(sm_model.summary())

```
Although the VIF scores of the remaining features have are lower, we can see
a significant drop in the model's performance and hence we cannot eliminate
the CGPA feature.
Its is also important to notice that the CGPA has the highest weightage and
the most important among all the features. Dropping CGPA would very much
affect the model's performance
```

Check if the mean of residuals is nearly zero

In [31]:
```
X_sm = sm.add_constant(X_new)

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

In [32]:
```
sm_model.resid.mean()
```

Out[32]:
```
-1.065161847613183e-14
```

```
Here the mean of residuals(errors) is almost zero and this assumption is
successfully met
```

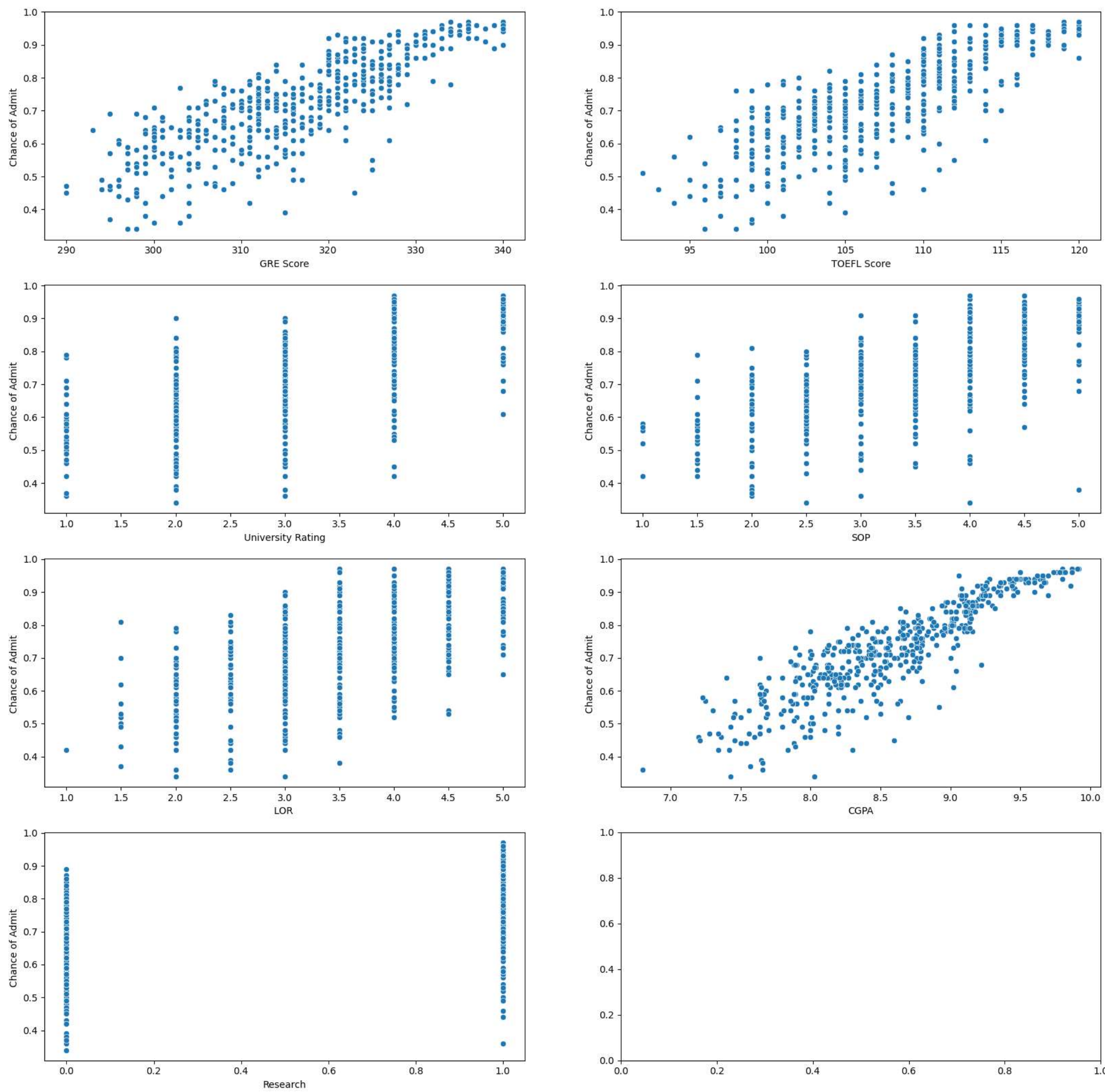
Linearity of variables

```
We can directly check the linearity between the predictor and target
variables using scatter plot
```

In [33]:

```
fig, axis = plt.subplots(nrows=4, ncols=2, figsize=(20, 20))
sns.scatterplot(x = 'GRE Score',y = 'Chance of Admit ',data = jamboree,ax=axis
sns.scatterplot(x = 'TOEFL Score',y = 'Chance of Admit ',data = jamboree,ax=ax
sns.scatterplot(x = 'University Rating',y = 'Chance of Admit ',data = jamboree
sns.scatterplot(x = 'SOP',y = 'Chance of Admit ',data = jamboree,ax=axis[1,1])
sns.scatterplot(x = 'LOR ',y = 'Chance of Admit ',data = jamboree,ax=axis[2,0]
sns.scatterplot(x = 'CGPA',y = 'Chance of Admit ',data = jamboree,ax=axis[2,1]
sns.scatterplot(x = 'Research',y = 'Chance of Admit ',data = jamboree,ax=axis[
```

Out[33]: <AxesSubplot:xlabel='Research', ylabel='Chance of Admit '>



1. As we can observe, GRE Score, TOEFL Score and CGPA have a linear relationship with the Chance of Admit. Although GRE score and TOEFL score are more scattered, CGPA has a much more more linear relationship with the Chance of Admit.
2. University rating, SOP and LOR have somewhat a linear relationship but very much scattered

We can also use the Pearson's 'r' to check the linear relationship between all the predictor and target variables

In [34]:
```
jamboree.corr()['Chance of Admit ']
```

Out[34]:
```
GRE Score            0.810351
TOEFL Score          0.792228
University Rating    0.690132
SOP                  0.684137
LOR                  0.645365
CGPA                 0.882413
Research             0.545871
Chance of Admit      1.000000
Name: Chance of Admit , dtype: float64
```

As evident from the visual analysis above and also from the Pearson's 'r' value, CGPA, GRE Score and TOEFL Score has a high linear relationship with Chance of Admit

Test for Homoscedasticity(minimal to no heteroscadasticity)

In [35]:
```
predicted = sm_model.predict()
residuals = sm_model.resid
```

Breusch-Pagan test for homoscedasticity

Null Hypothesis H0 : Homoscedasticity is present
Alternate Hypothesis Ha : Heteroscedasticity is present
alpha : 0.05

In [36]:
```
 bp_test = pd.DataFrame(sms.het_breuschpagan(residuals, sm_model.model.exog),
                          columns=['value'],
                          index=['Lagrange multiplier statistic', 'p-value',
```

In [37]:
```
bp_test
```

Out[37]:

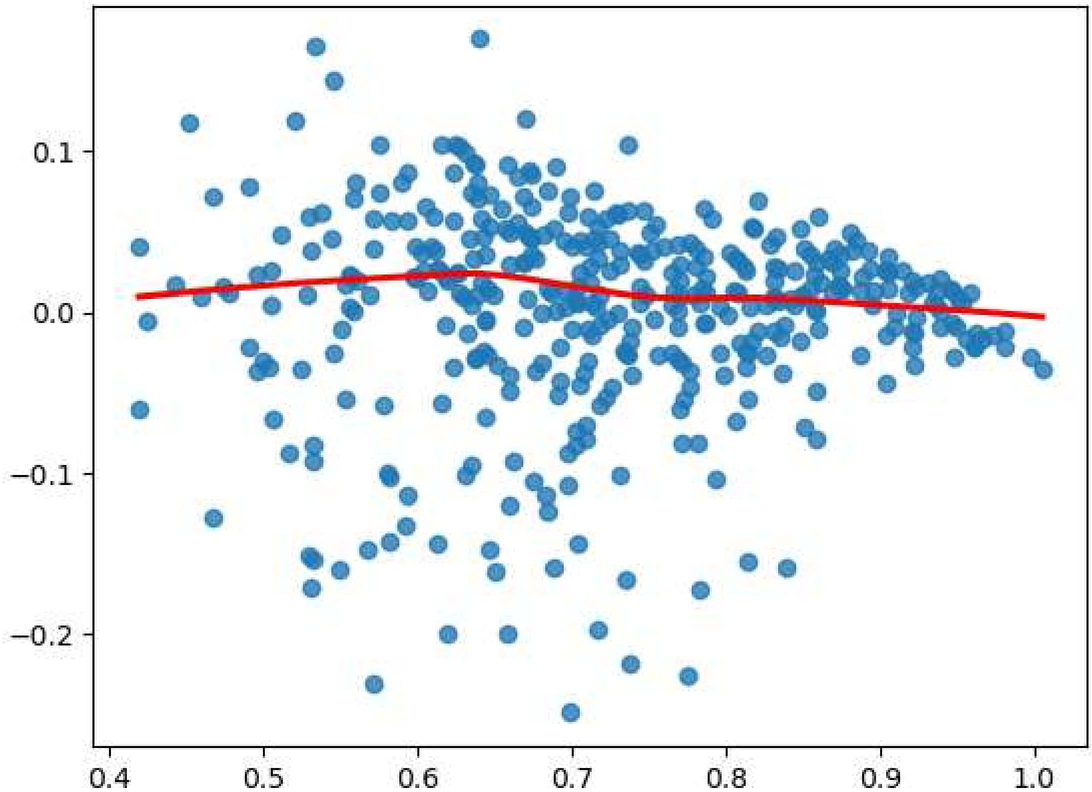
	value
Lagrange multiplier statistic	28.461274
p-value	0.000077
f-value	5.017548
f p-value	0.000057

```
Since the p-value is much lower than the alpha value, we can reject the null
hypothesis and conclude that Heteroscedasticity is present
```

Regplot for visualization of homoscedasticity

In [38]:
```
sns.regplot(x=predicted, y=residuals, lowess=True, line_kws={'color': 'red'})
```

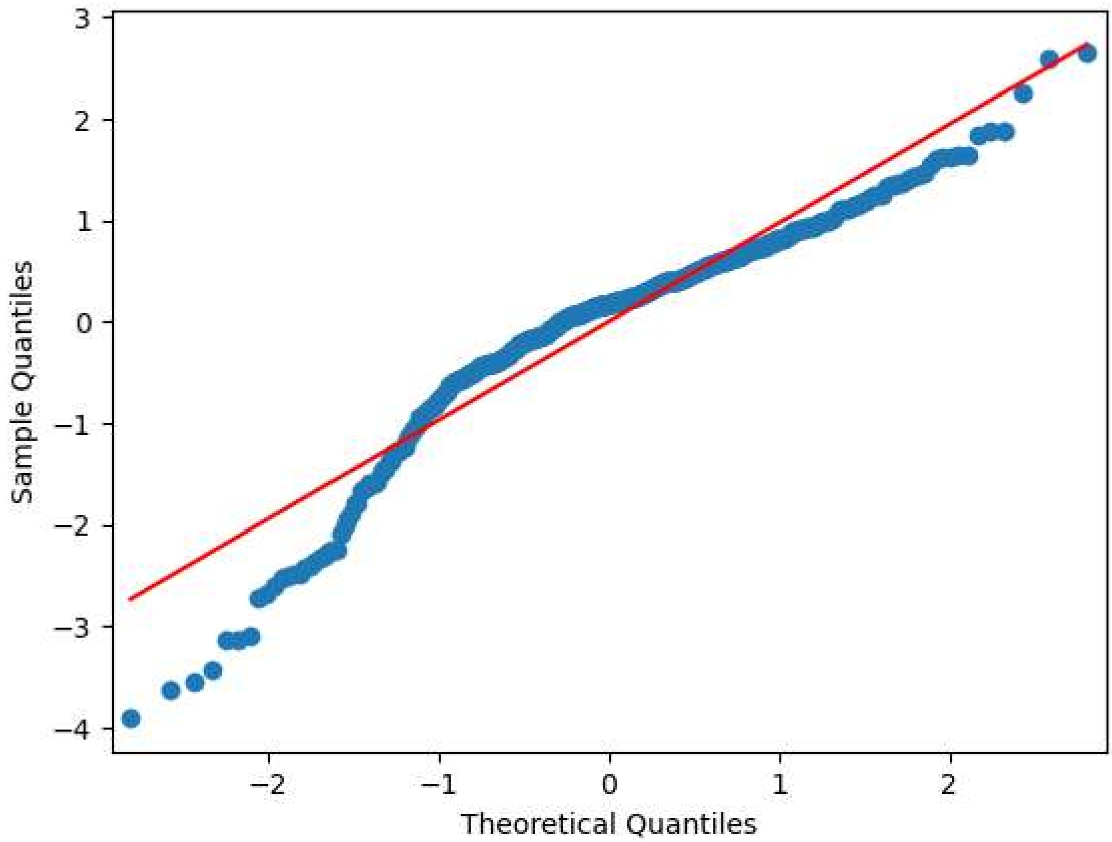
Out[38]: <AxesSubplot:>




```
We can also simply visualize the heteroscedasticity between the different
values of Y
```

Normality of residuals

In [39]:
```
import statsmodels.api as sm
qqplot(residuals,stats.norm,fit=True,line='r')
plt.show()
```




```
As we can see from the above plot, the residuals are not exactly normally
distributed and follows a somewhat normal distribution. This assumption is
not exactly met.
```

Model performance evaluation

Mean Absolute Error(MAE)

In [40]:
```
error = mae(y_train, predicted)
error
```

Out[40]: 0.046195490173396914

Root Mean Squared Error(RMSE)

In [41]:
```
rmse = np.sqrt(mean_squared_error(y_train, predicted, squared = False))
rmse
```

Out[41]:
```
0.2526228350601312
```

R-squared and Adjusted R-squared

In [42]:
```
sm_model.summary()
```

Out[42]:

OLS Regression Results

Dep. Variable:	Chance of Admit	**R-squared:**	0.799
Model:	OLS	**Adj. R-squared:**	0.796
Method:	Least Squares	**F-statistic:**	260.8
Date:	Mon, 12 Jun 2023	**Prob (F-statistic):**	1.19e-133
Time:	00:40:58	**Log-Likelihood:**	533.11
No. Observations:	400	**AIC:**	-1052.
Df Residuals:	393	**BIC:**	-1024.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.9804	0.080	-12.202	0.000	-1.138	-0.822
TOEFL Score	0.0044	0.001	4.443	0.000	0.002	0.006
University Rating	0.0064	0.005	1.331	0.184	-0.003	0.016
SOP	-0.0045	0.006	-0.811	0.418	-0.016	0.006
LOR	0.0222	0.006	3.961	0.000	0.011	0.033
CGPA	0.1325	0.011	11.614	0.000	0.110	0.155
Research	0.0313	0.008	4.072	0.000	0.016	0.046

Omnibus:	79.400	**Durbin-Watson:**	0.773
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	147.282
Skew:	-1.103	**Prob(JB):**	1.04e-32
Kurtosis:	4.993	**Cond. No.**	2.70e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.7e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Train and test performances are checked

In [43]:
```
model = LinearRegression().fit(x_train, y_train)
print("intercept w0 : ",model.intercept_)
print("co efficients : ",model.coef_)
```

```
intercept w0 :  -1.2594324782480177
co efficients :  [ 0.00173741  0.00291958  0.00571666 -0.00330517  0.02235313
0.11893945
  0.02452511]
```

In [44]:
```
ypred = model.predict(x_test)
```

In [45]:
```
# model score after training
model.score(x_train, y_train)
```

Out[45]:
```
0.8034713719824393
```

In [46]:
```
# model score with test data
model.score(x_test, y_test)
```

Out[46]:
```
0.898286909853386
```

```
Here we can see that the model is trained well and also the test data has a
very good performance
```

Comments on the performance measures

In [47]:
```
sm_model.summary()
```

Out[47]:

OLS Regression Results

Dep. Variable:	Chance of Admit	**R-squared:**	0.799
Model:	OLS	**Adj. R-squared:**	0.796
Method:	Least Squares	**F-statistic:**	260.8
Date:	Mon, 12 Jun 2023	**Prob (F-statistic):**	1.19e-133
Time:	00:40:58	**Log-Likelihood:**	533.11
No. Observations:	400	**AIC:**	-1052.
Df Residuals:	393	**BIC:**	-1024.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.9804	0.080	-12.202	0.000	-1.138	-0.822
TOEFL Score	0.0044	0.001	4.443	0.000	0.002	0.006
University Rating	0.0064	0.005	1.331	0.184	-0.003	0.016
SOP	-0.0045	0.006	-0.811	0.418	-0.016	0.006
LOR	0.0222	0.006	3.961	0.000	0.011	0.033
CGPA	0.1325	0.011	11.614	0.000	0.110	0.155
Research	0.0313	0.008	4.072	0.000	0.016	0.046

Omnibus:	79.400	**Durbin-Watson:**	0.773
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	147.282
Skew:	-1.103	**Prob(JB):**	1.04e-32
Kurtosis:	4.993	**Cond. No.**	2.70e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.7e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

```
The performance measures indicate that the performance of the model is good.
Although if we want to improve the model performance we can introduce more
relevant features with linear relationship and drop some not so relevant
columns which do not contribute much to the Chance of Admit.
```