Jamboree Education!

```
import pandas as pd
import numpy as np

df=pd.read_csv("/content/Jamboree_Admission.csv")
```

Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), missing value detection, statistical summary.

```
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
```

```
df.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
```

```
df.isnull().sum()
```

```
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
```

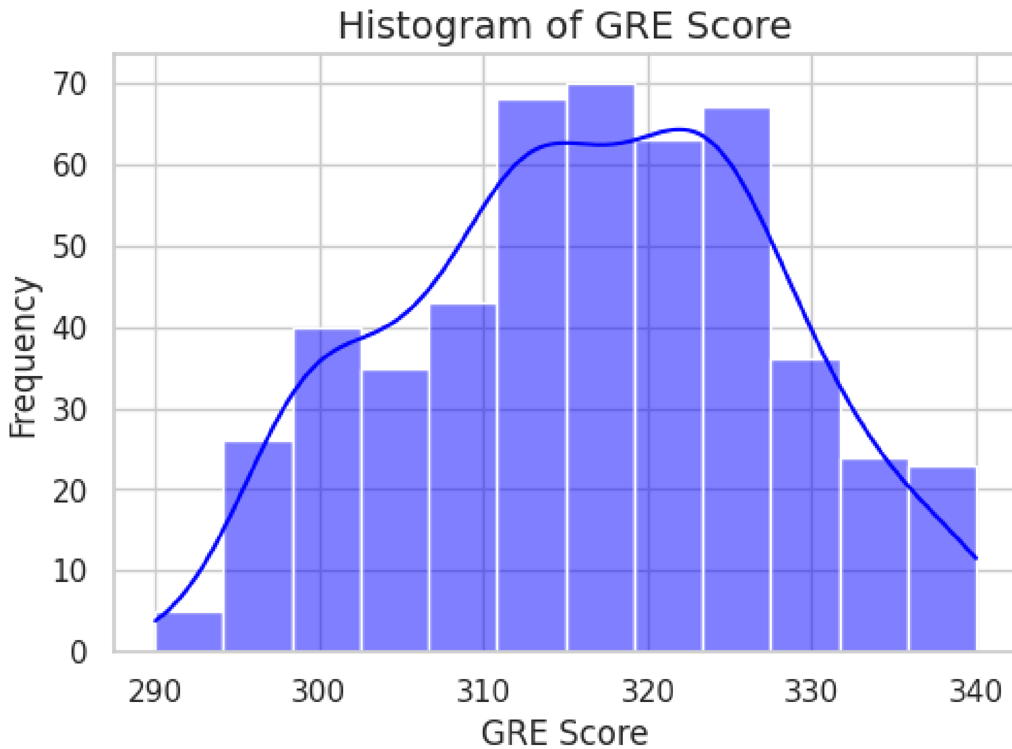
```
df.shape
```

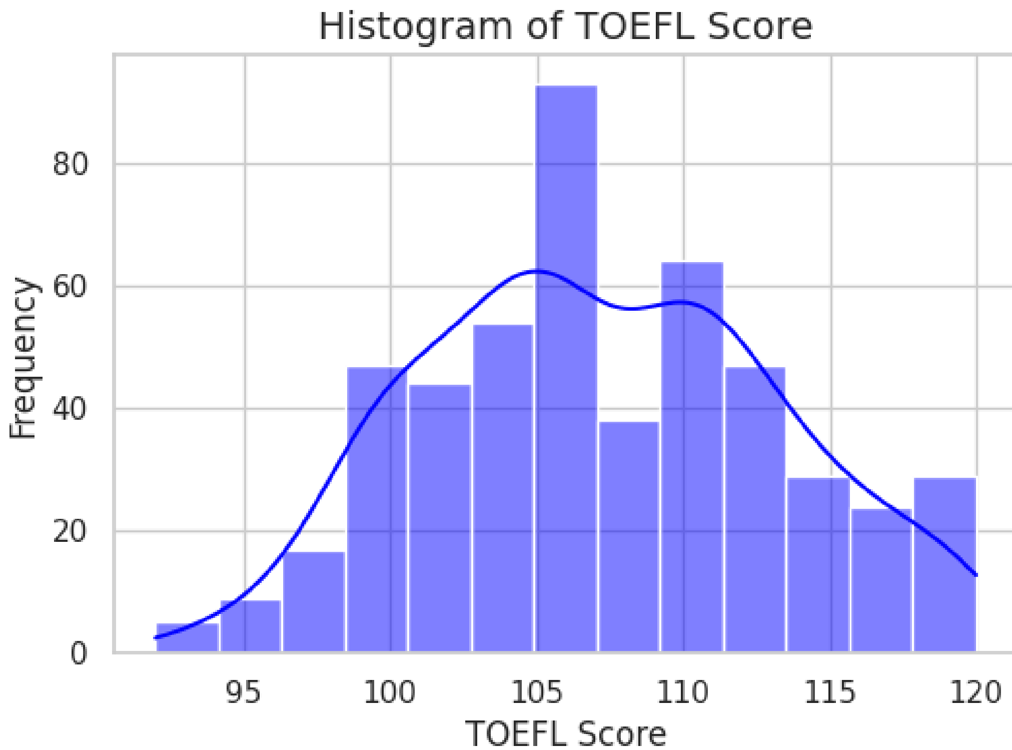
```
(500, 9)
```

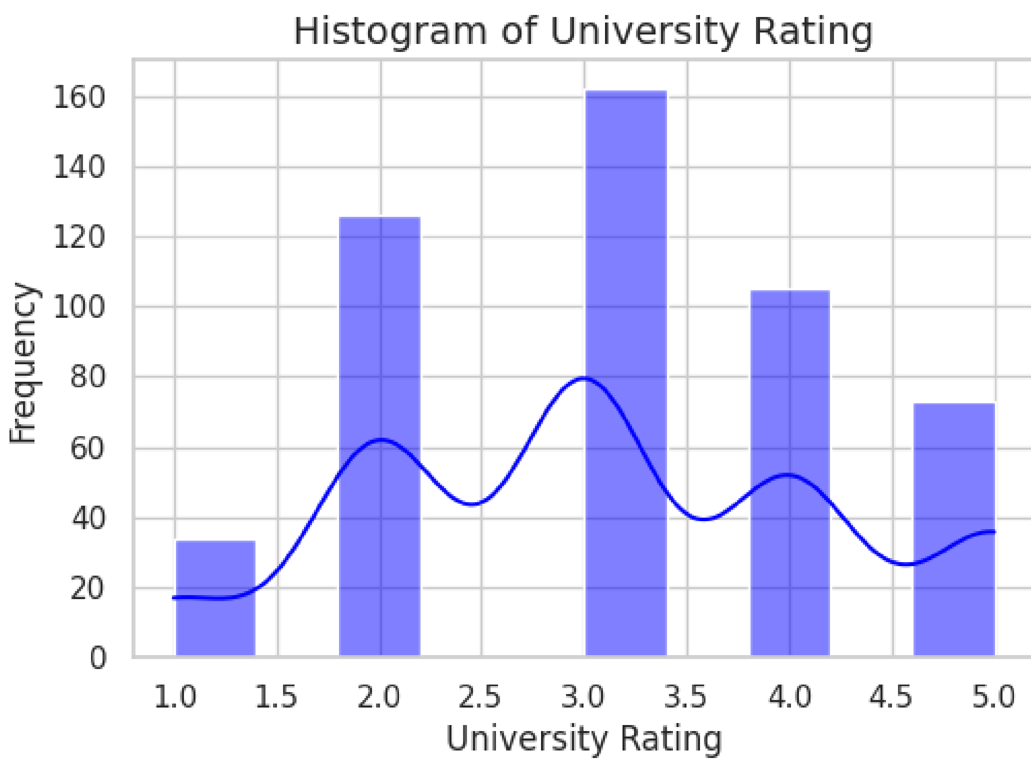
Univariate Analysis (distribution plots of all the continuous variable(s)barplots/countplots of all the categorical variables

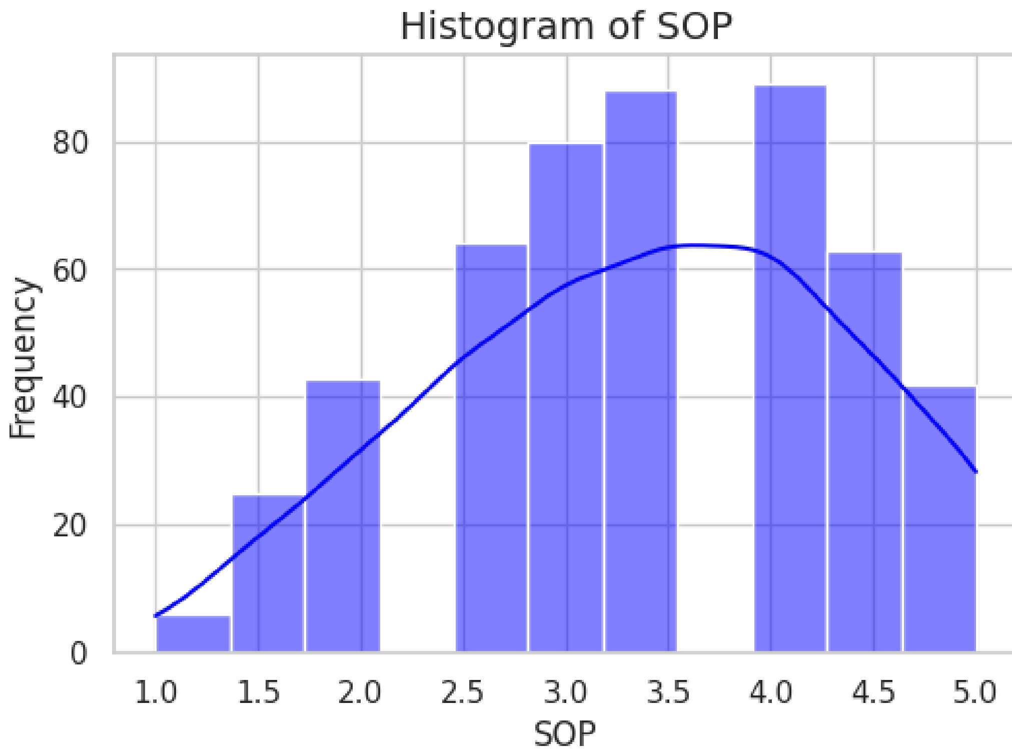
```
df.drop(columns= 'Serial No.', inplace= True)

import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style="whitegrid")
for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[column], kde=True, color='blue')
```

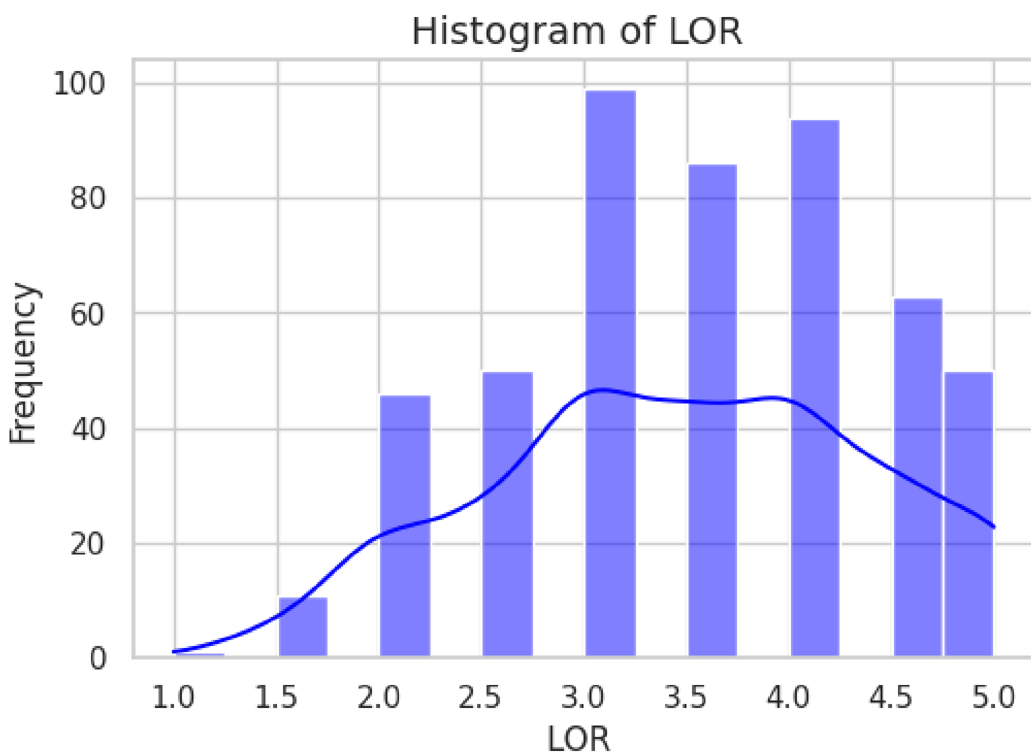
```
    plt.title(f'Histogram of {column}', fontsize=14)
    plt.xlabel(column, fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.show()
```

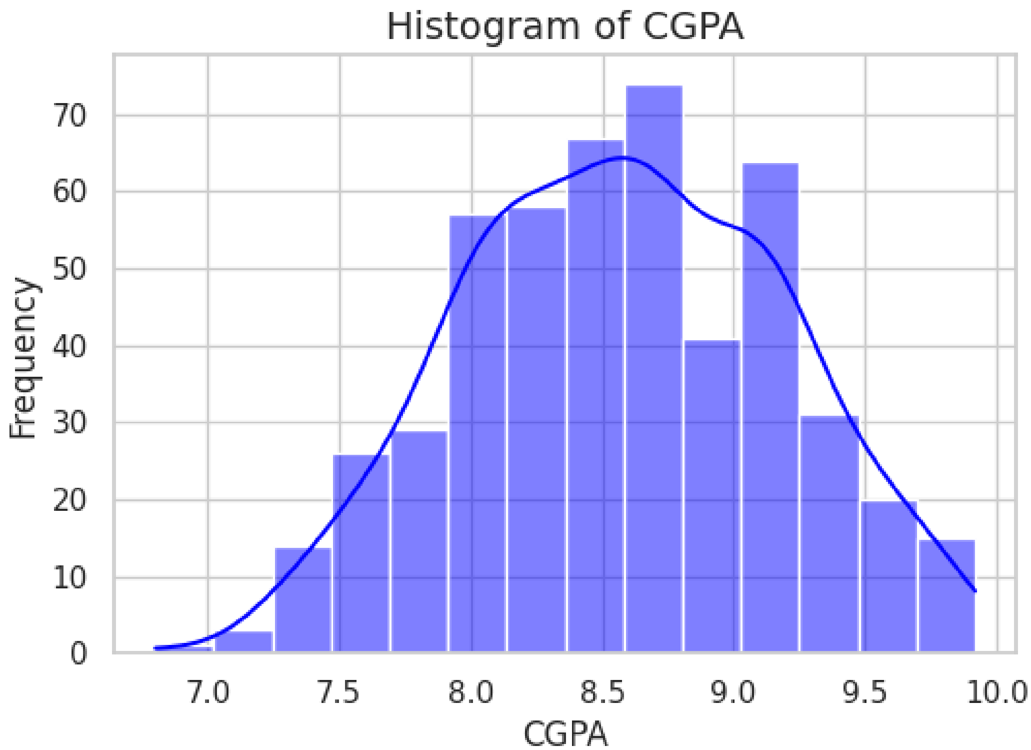


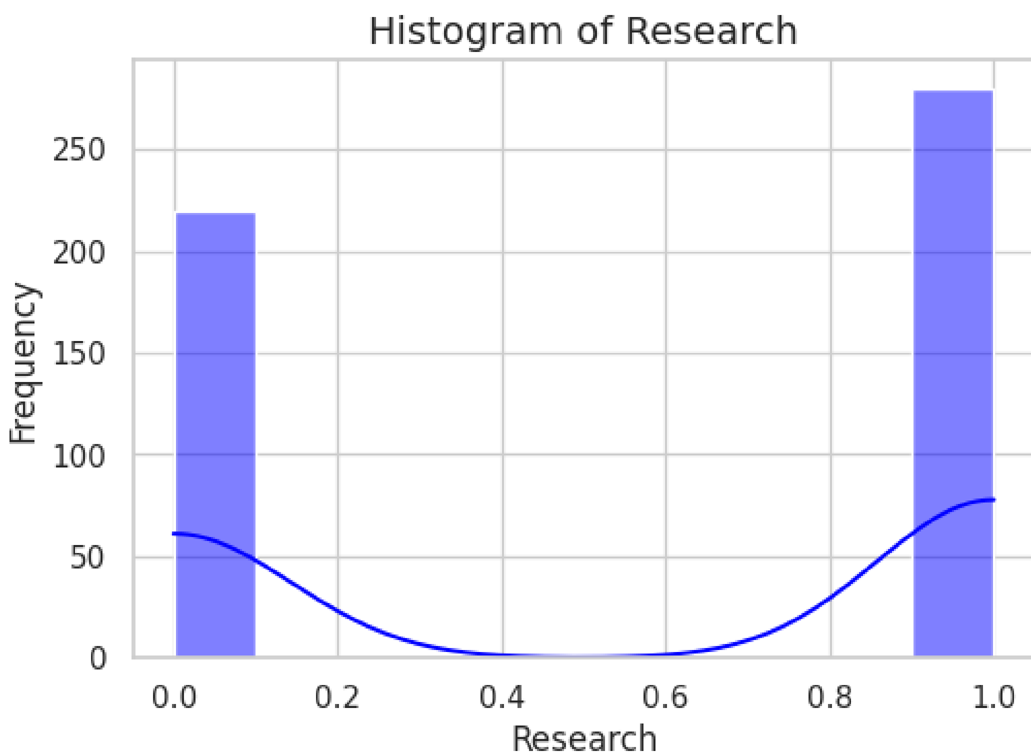

Histogram of TOEFL Score
Frequency
0
20
40
60
80
95
100
105
110
115
120
TOEFL Score

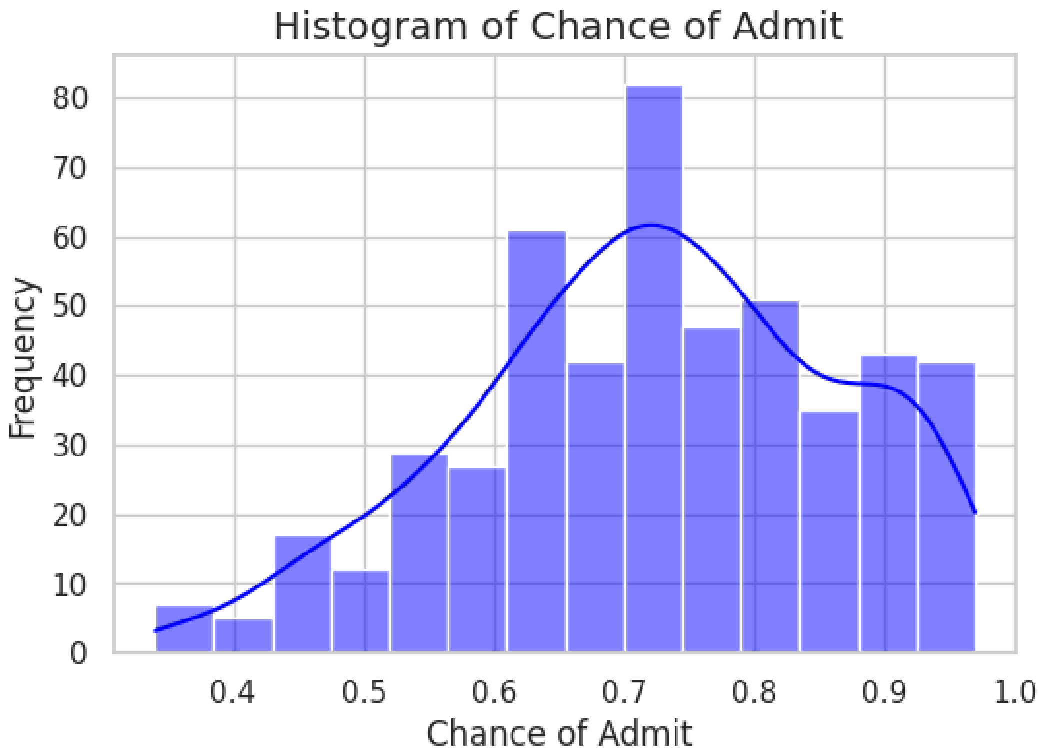

Histogram of University Rating
Frequency
0
20
40
60
80
100
120
140
160
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
University Rating


Histogram of SOP
Frequency
SOP
0
20
40
60
80
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

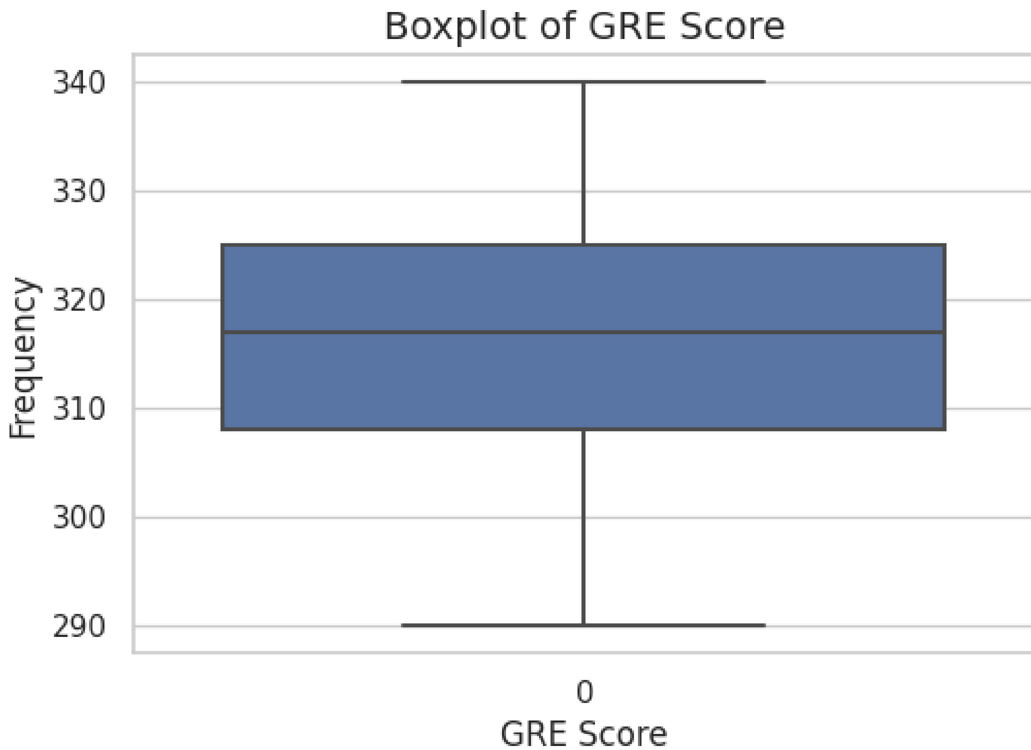

Histogram of LOR
Frequency
LOR
0
20
40
60
80
100
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

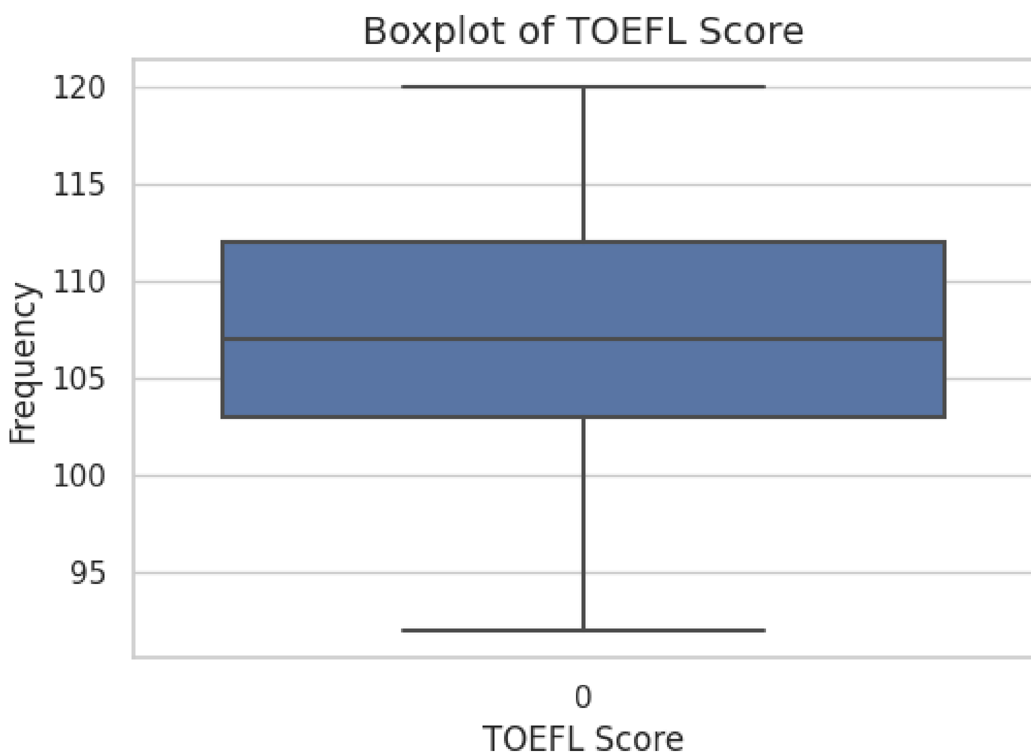

Histogram of CGPA
Frequency
0
10
20
30
40
50
60
70
7.0
7.5
8.0
8.5
9.0
9.5
10.0
CGPA

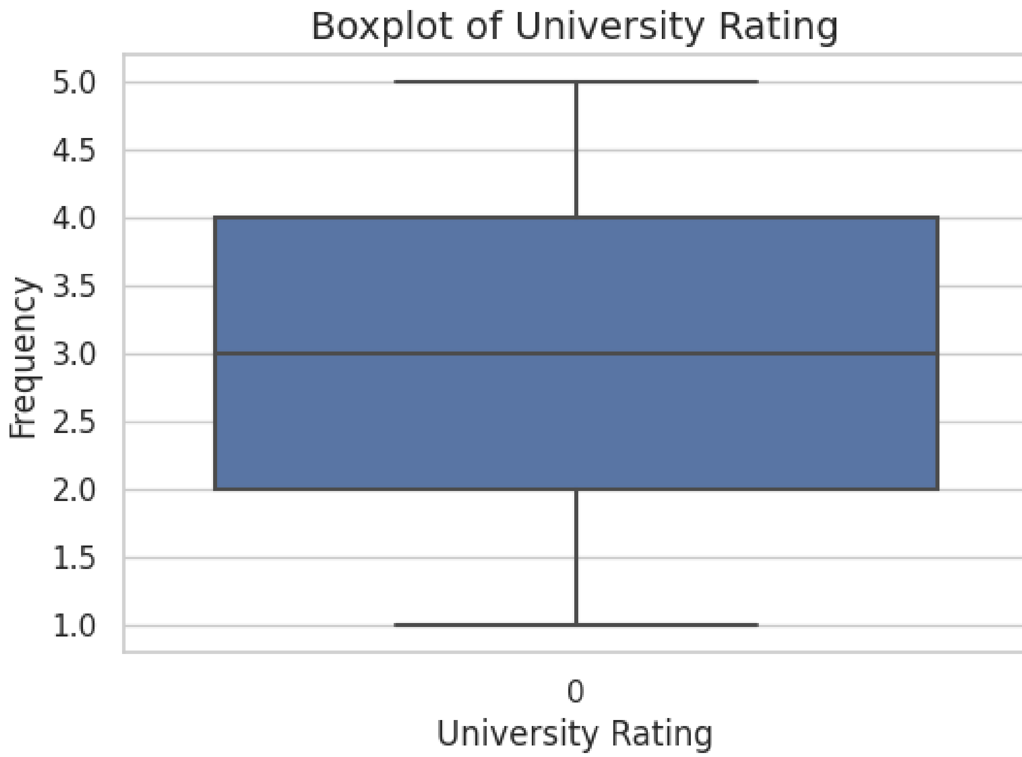

Histogram of Research
Frequency
0
50
100
150
200
250
0.0
0.2
0.4
0.6
0.8
1.0
Research

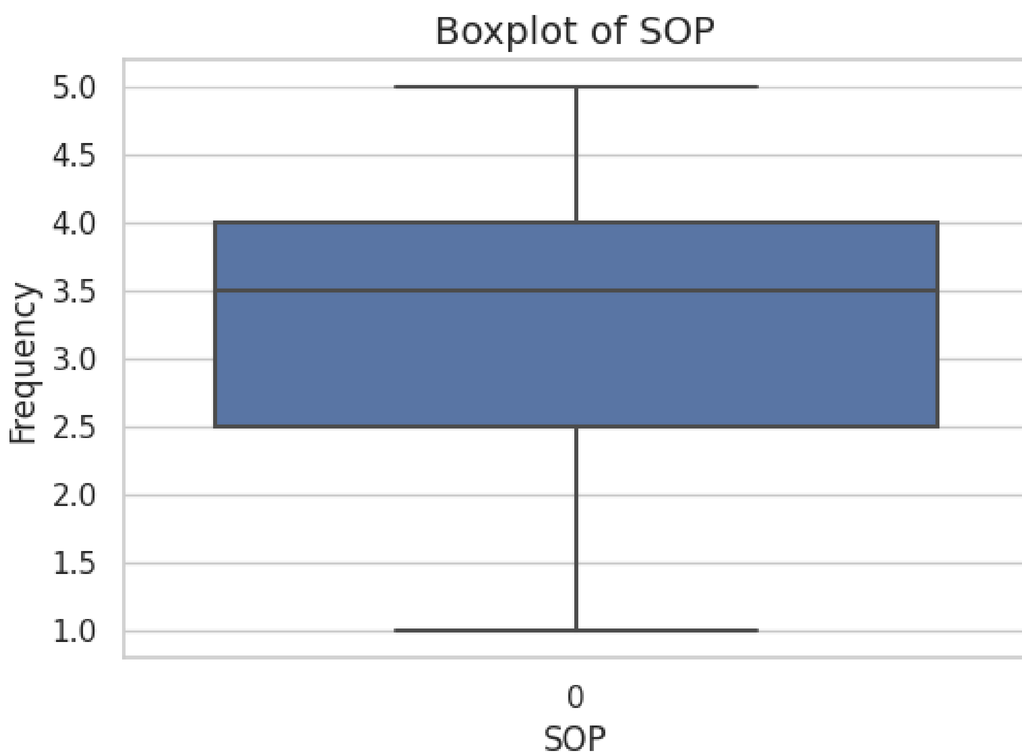


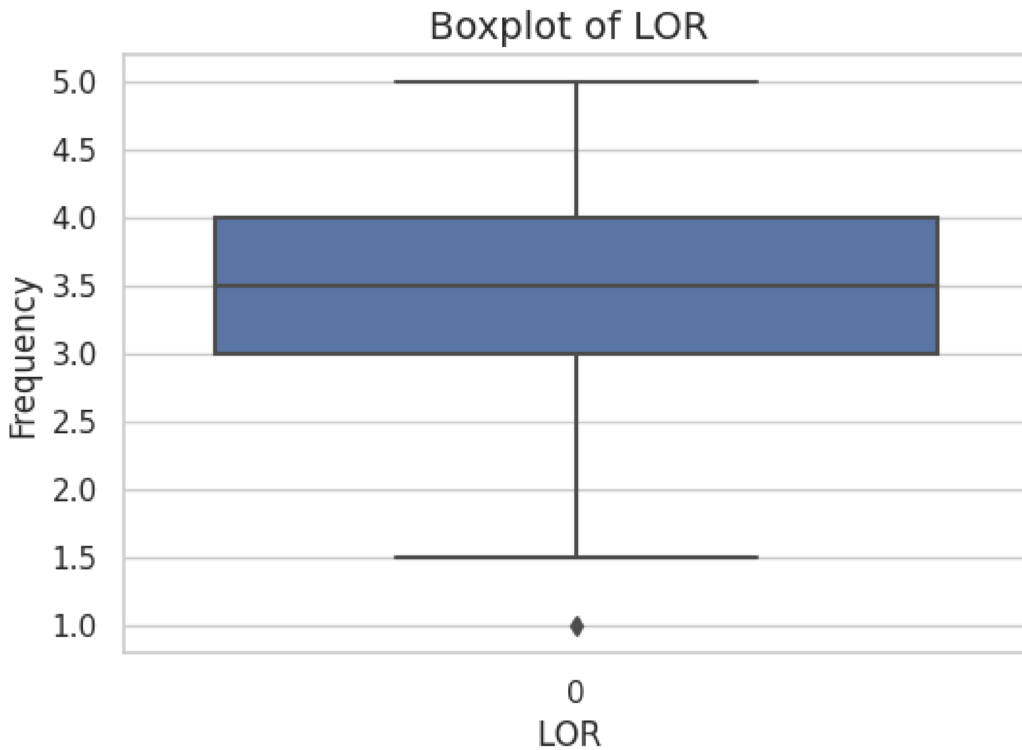

```
for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.boxplot(df[column])
    plt.title(f'Boxplot of {column}', fontsize=14)
    plt.xlabel(column, fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.show()
```

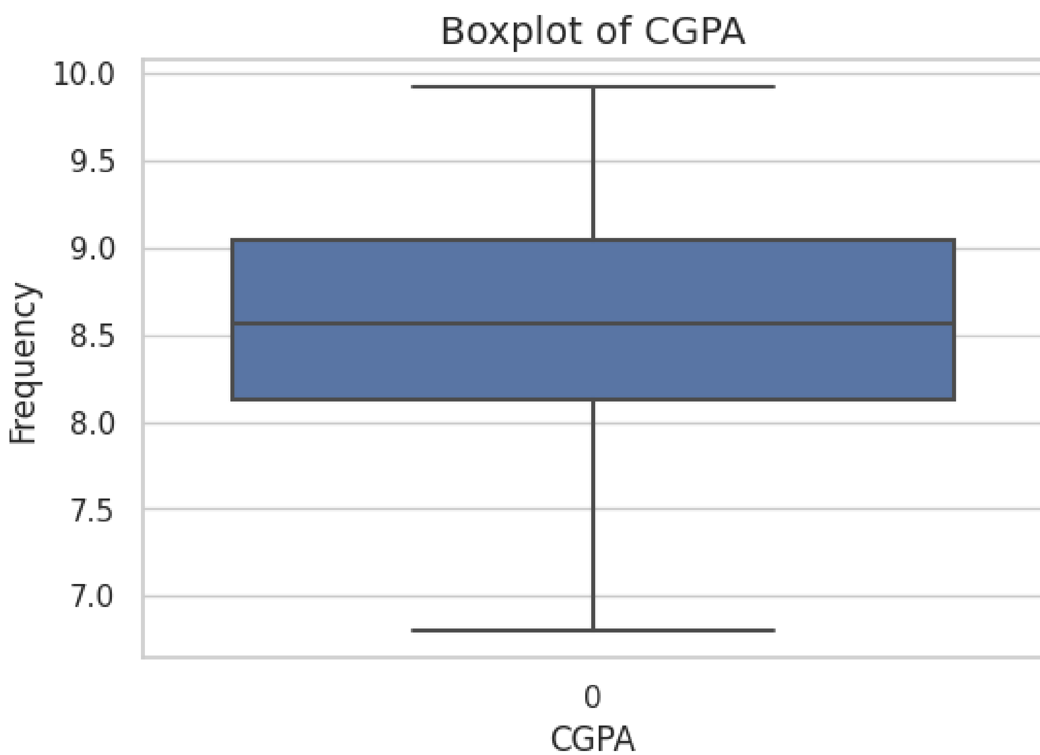

Boxplot of GRE Score
340
330
320
310
300
290
Frequency
0
GRE Score

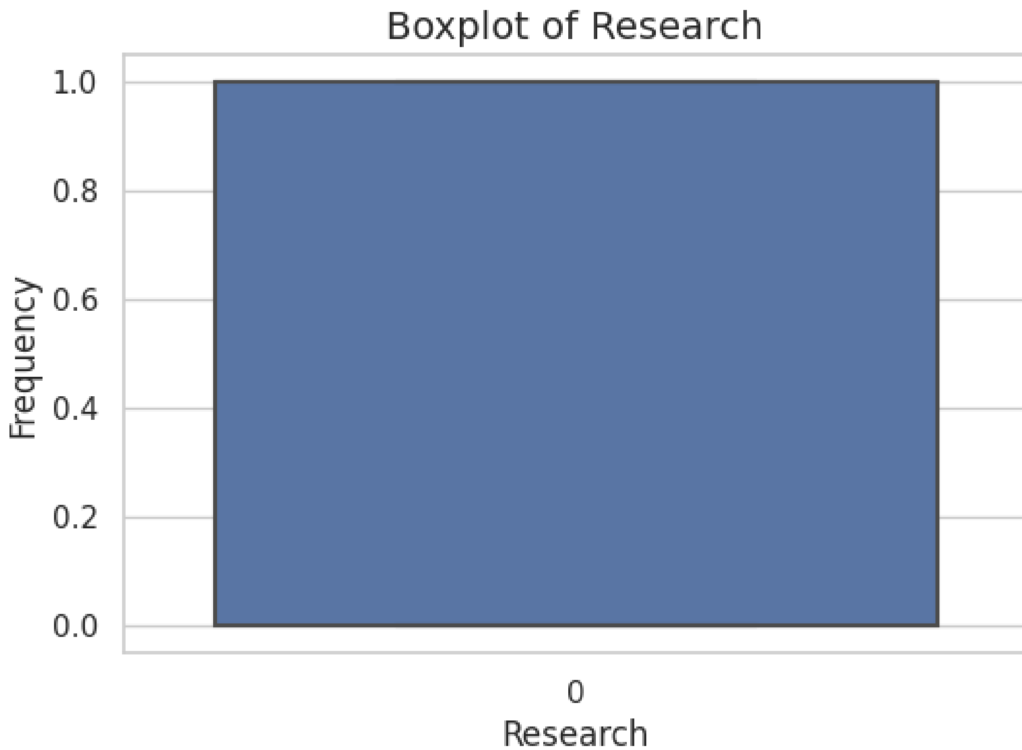

Boxplot of TOEFL Score
120
115
110
105
100
95
Frequency
0
TOEFL Score

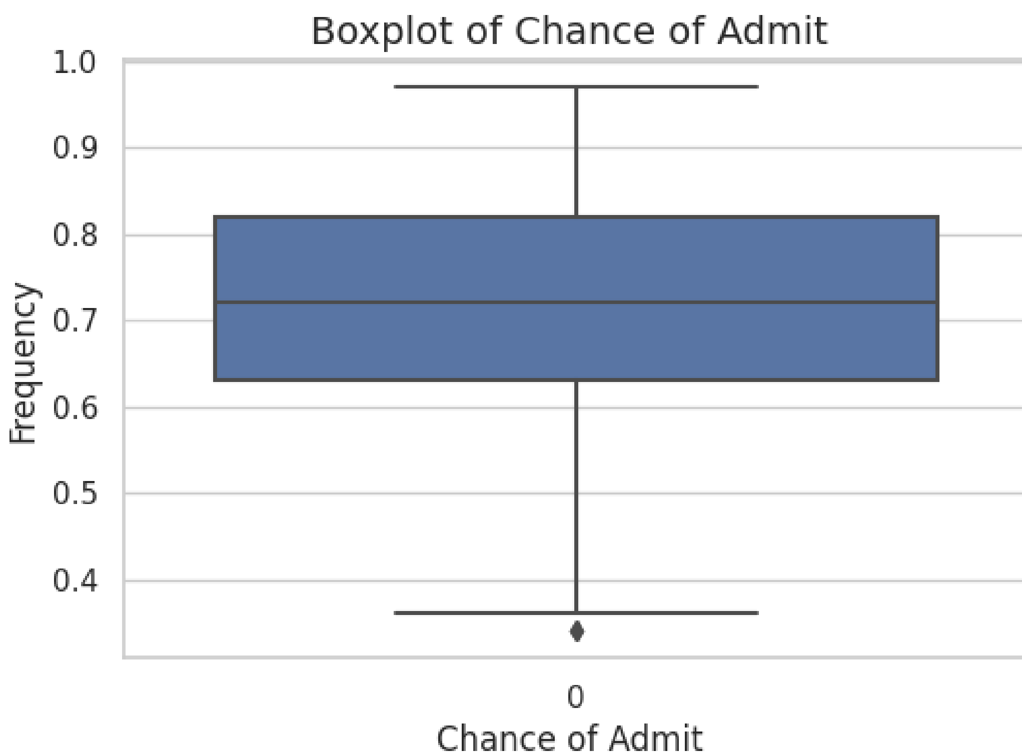

Boxplot of University Rating
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Frequency
0
University Rating


Boxplot of SOP
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Frequency
0
SOP


Boxplot of LOR
Frequency
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
LOR


Boxplot of CGPA
Frequency
10.0
9.5
9.0
8.5
8.0
7.5
7.0
0
CGPA





outliers found in LOR and ADMIT columns

Bivariate Analysis (Relationships between important variables

1. Data Preprocessing (10 Points)

a. Duplicate value check

b. Missing value treatment

c. Outlier treatment

d. Feature engineering

e. Data preparation for modeling

```
df.columns
```

```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ',
'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

```
sc= ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ',
'CGPA']
X= df.drop(columns='Chance of Admit ')
Y= df["Chance of Admit "]

from sklearn.preprocessing import MinMaxScaler
scaler= MinMaxScaler()
X[sc]= scaler.fit_transform(X[sc])

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=2)
```

Model building :

a. Build the Linear Regression model and comment on the model statistics

b. Display model coefficients with column names

```
import statsmodels.api as sm
X_sm= sm.add_constant(X_train)
model= sm.OLS(Y_train, X_sm)
result= model.fit()
print(result.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:                Sun, 07 Jan 2024   Prob (F-statistic):
3.33e-146
Time:                        04:01:35   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                 0.3450      0.010     34.259      0.000
0.325       0.365
GRE Score             0.1067      0.027      3.893      0.000
0.053       0.161
TOEFL Score           0.0826      0.027      3.024      0.003
0.029       0.136
University Rating     0.0194      0.016      1.185      0.237      -
0.013       0.052
SOP                   0.0084      0.020      0.428      0.669      -
0.030       0.047
LOR                   0.0744      0.018      4.131      0.000
0.039       0.110
CGPA                  0.3537      0.033     10.633      0.000
0.288       0.419
Research              0.0247      0.007      3.476      0.001
0.011       0.039
======================================================================
========
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.
23.4
======================================================================
```

```
========

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

Testing the assumptions of linear regression model (50 Points)

a. Multicollinearity check by VIF score (variables are dropped one-by-one till none has VIF>5) (10 Points)

b. Mean of residuals is nearly zero (10 Points)

c. Linearity of variables (no pattern in residual plot) (10 Points)

d. Test for Homoscedasticity (10 Points)

e. Normality of residuals (almost bell-shaped curve in residuals distribution, points in QQ plot are almost all on the line) (10 Points)

```
from statsmodels.stats.outliers_influence import
variance_inflation_factor

vif=pd.DataFrame()
vif["features"]= X_sm.columns
vif['VIF_score'] = [variance_inflation_factor(X_sm, i) for i in
range(X_sm.shape[1])]
vif["VIF_score"]= round(vif["VIF_score"], 2)
vif.sort_values(by='VIF_score', ascending=False)
```

	features	VIF_score
0	const	11.94
6	CGPA	4.77
1	GRE Score	4.24
2	TOEFL Score	4.06
4	SOP	2.71
3	University Rating	2.59
5	LOR	1.98
7	Research	1.47

Mean of residuals is nearly zero

On Test data

```
X_test_sm= sm.add_constant(X_test)
Y_test_pred= result.predict(X_test_sm)
error_test= Y_test-Y_test_pred

error_test.mean()
```

```
-0.006100917484112264
```

On train data(Mean of residuals)

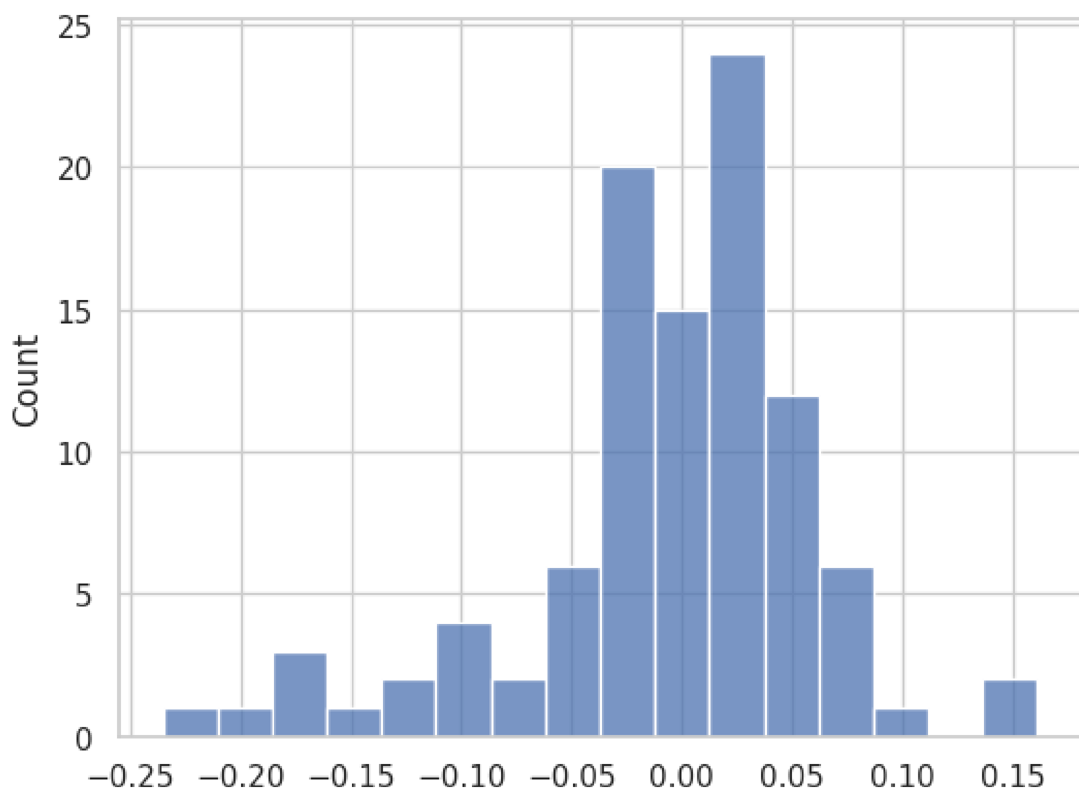
```
Y_train_pred= result.predict(X_sm)
error_train= Y_train-Y_train_pred
error_train.mean()
```

```
-1.6597834218146091e-16
```

Linearity of variables (no pattern in residual plot)

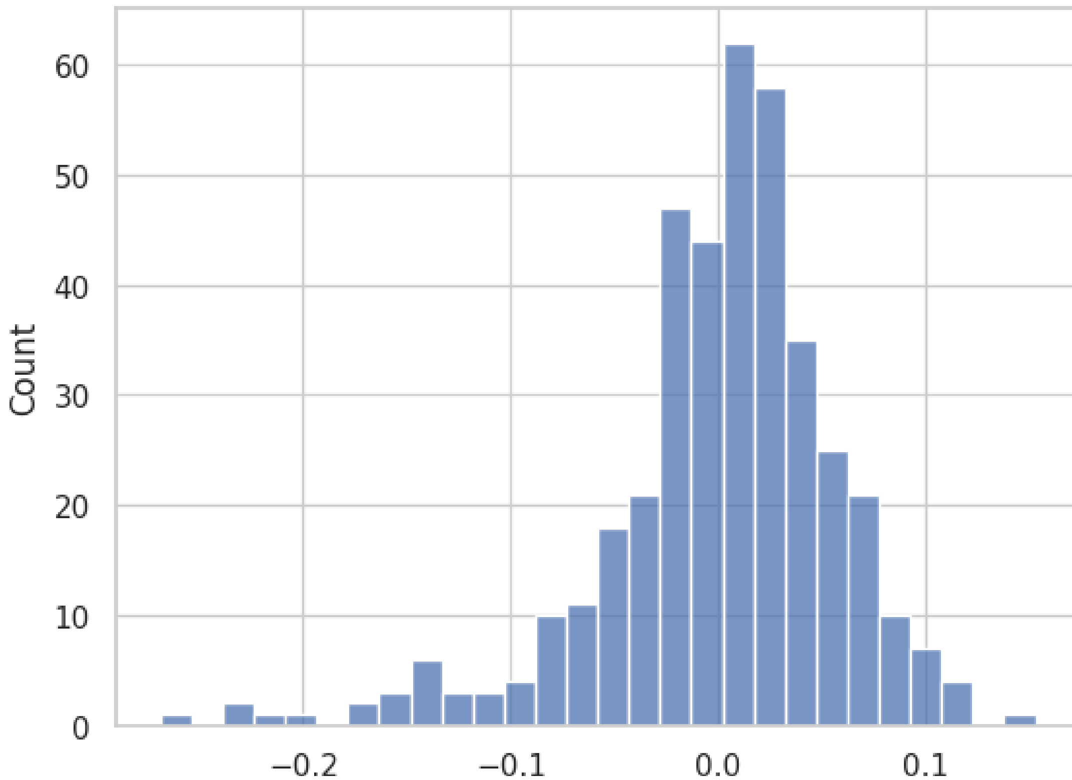
```
sns.histplot(error_test)
```

```
<Axes: ylabel='Count'>
```



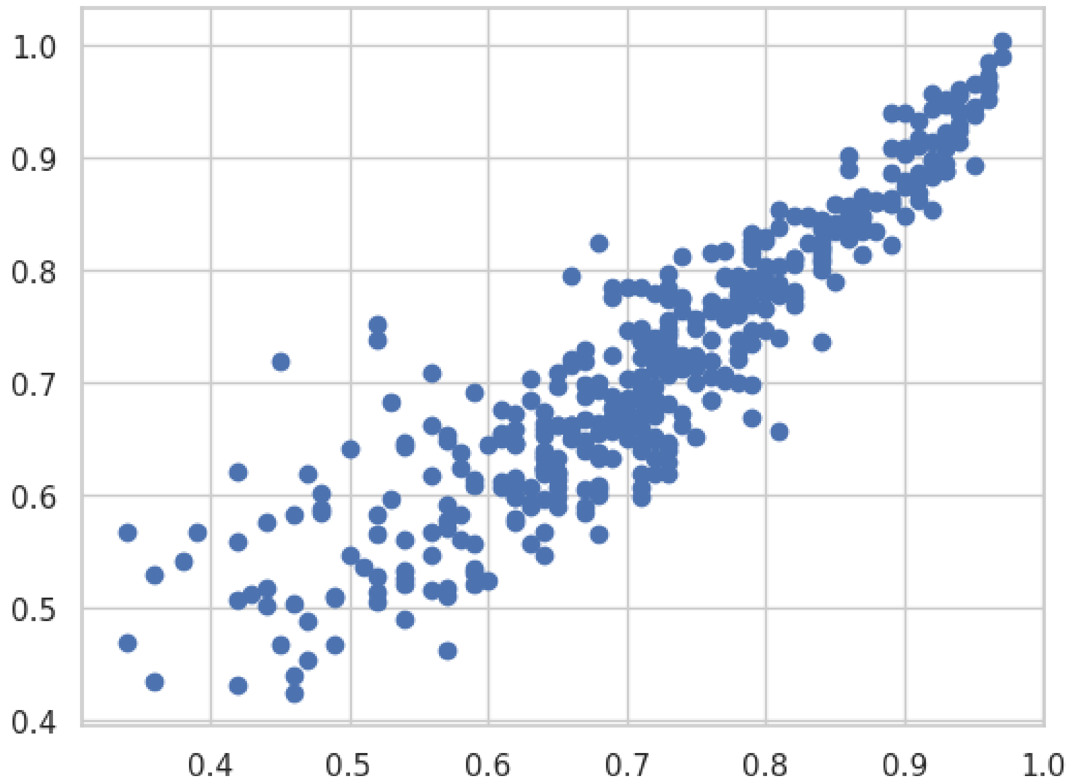

```
sns.histplot(error_train)
```

```
<Axes: ylabel='Count'>
```



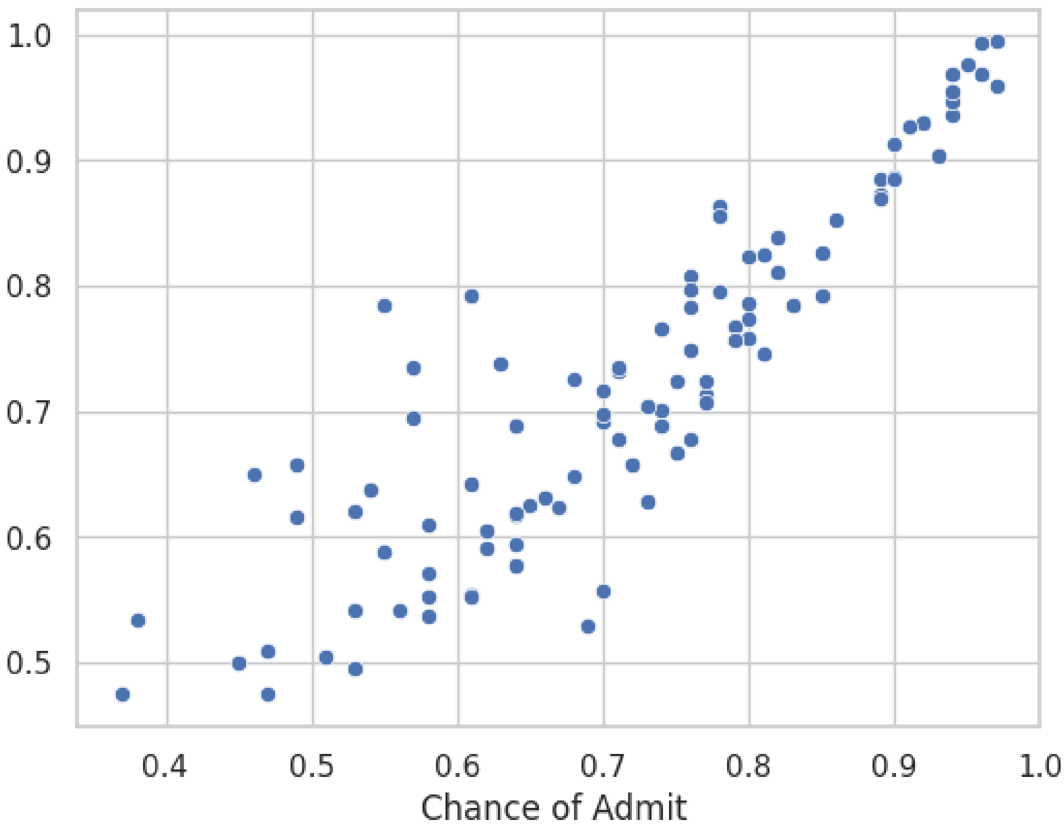

```
plt.scatter(Y_train, Y_train_pred)
```

```
<matplotlib.collections.PathCollection at 0x7b86dbc57b80>
```

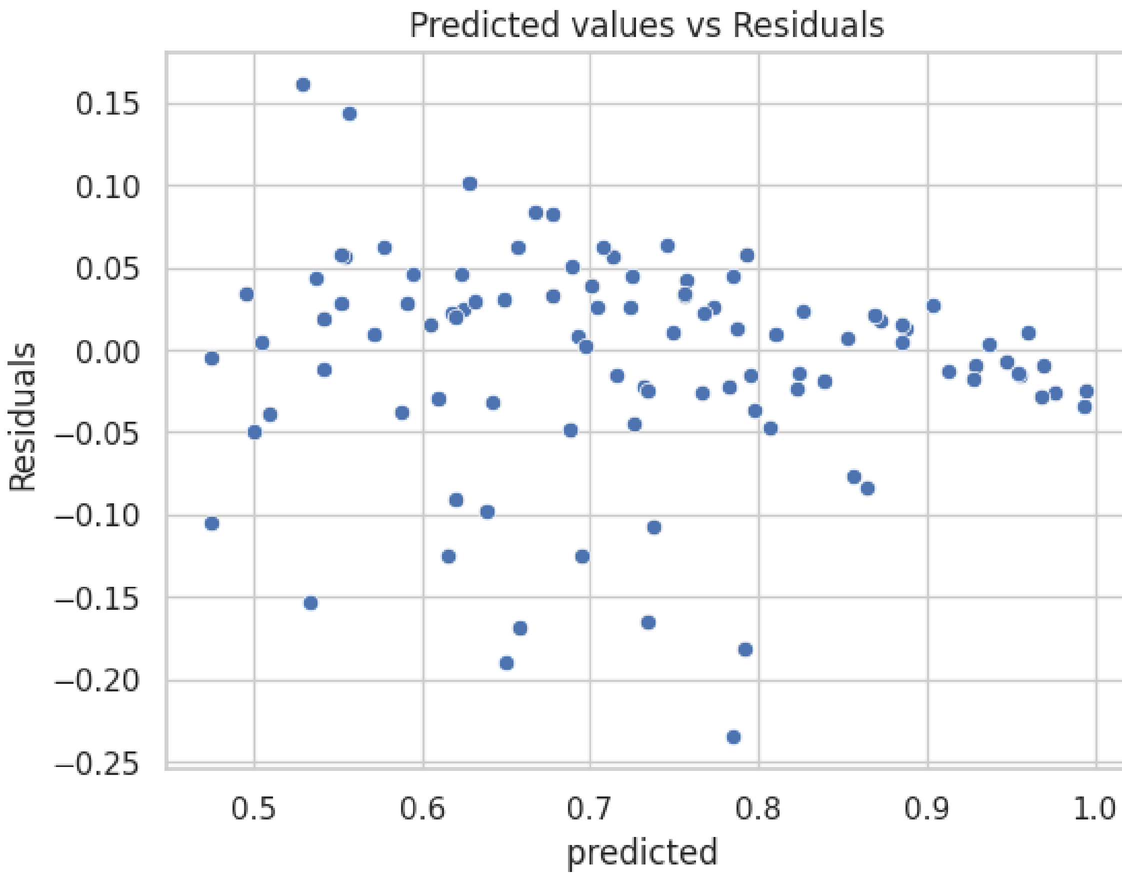



```
sns.scatterplot(x=Y_test, y=Y_test_pred)
```

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

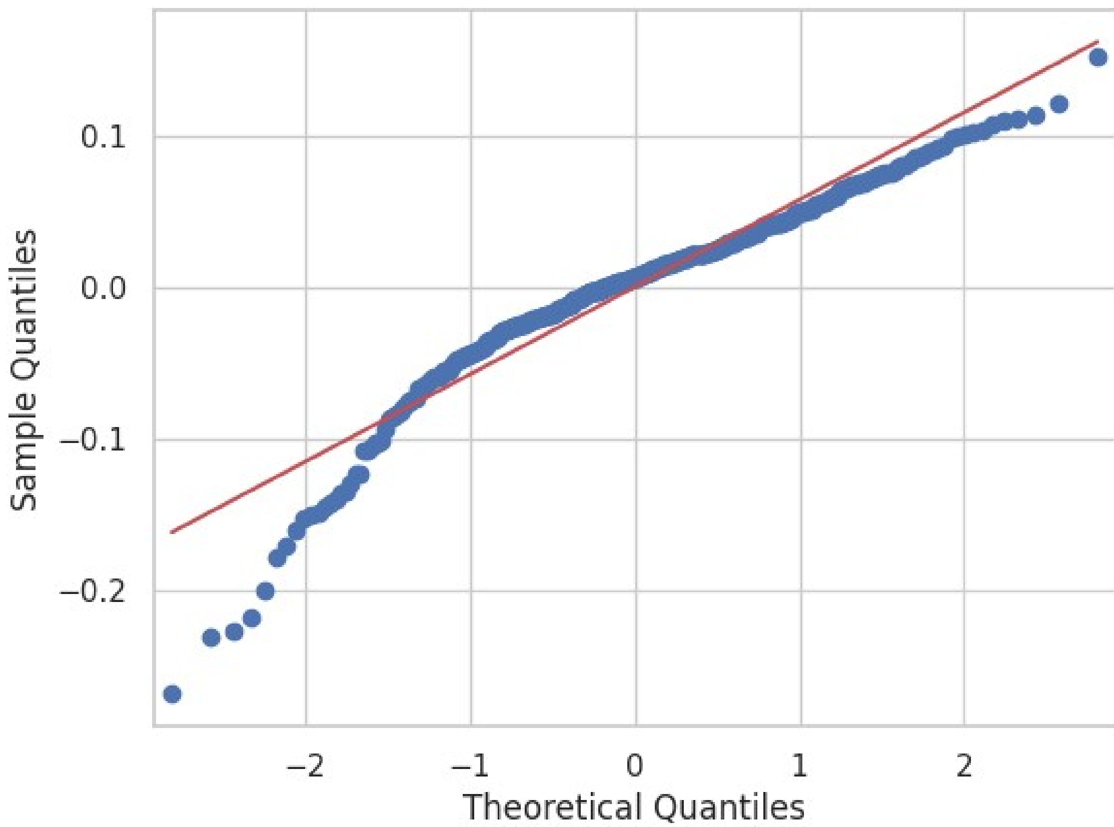



```
sns.scatterplot(x= Y_test_pred,y=error_test)
plt.xlabel("predicted ")
plt.ylabel("Residuals")
plt.title("Predicted values vs Residuals")

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

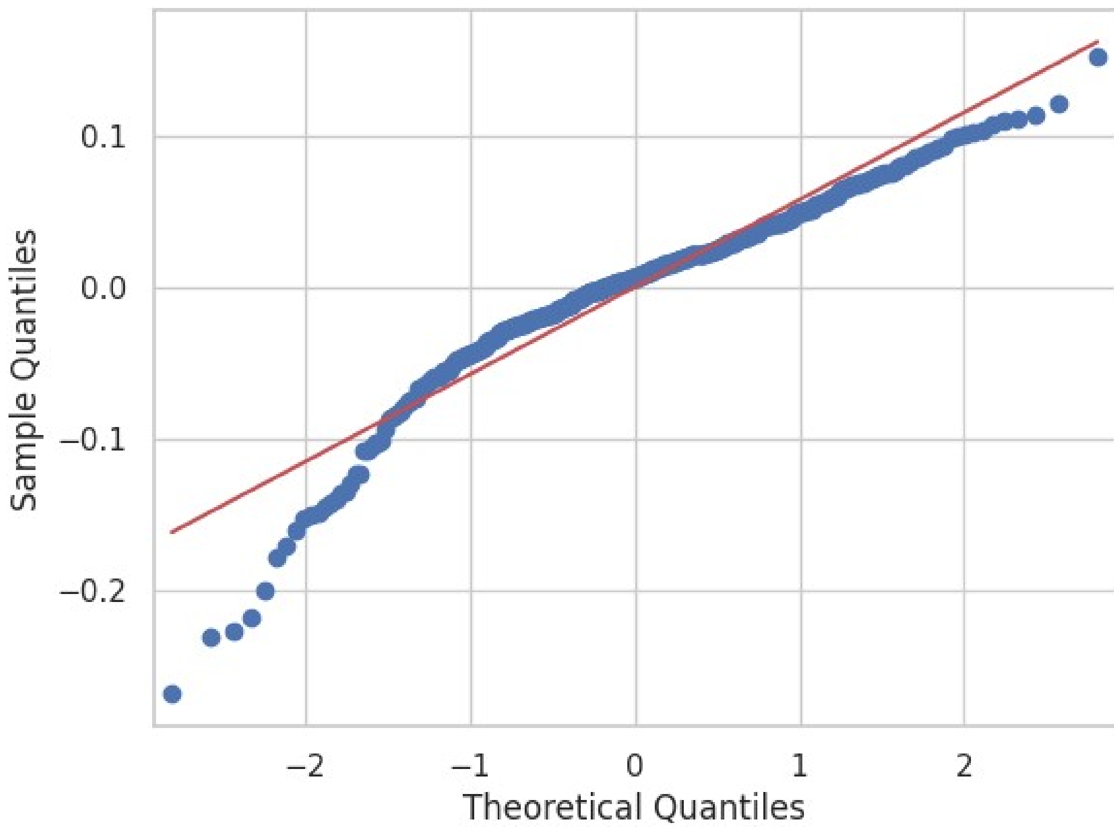


Q-Q plot (residuals on train data)

```
from statsmodels.graphics.gofplots import qqplot
qqplot( error_train, line='s')
```

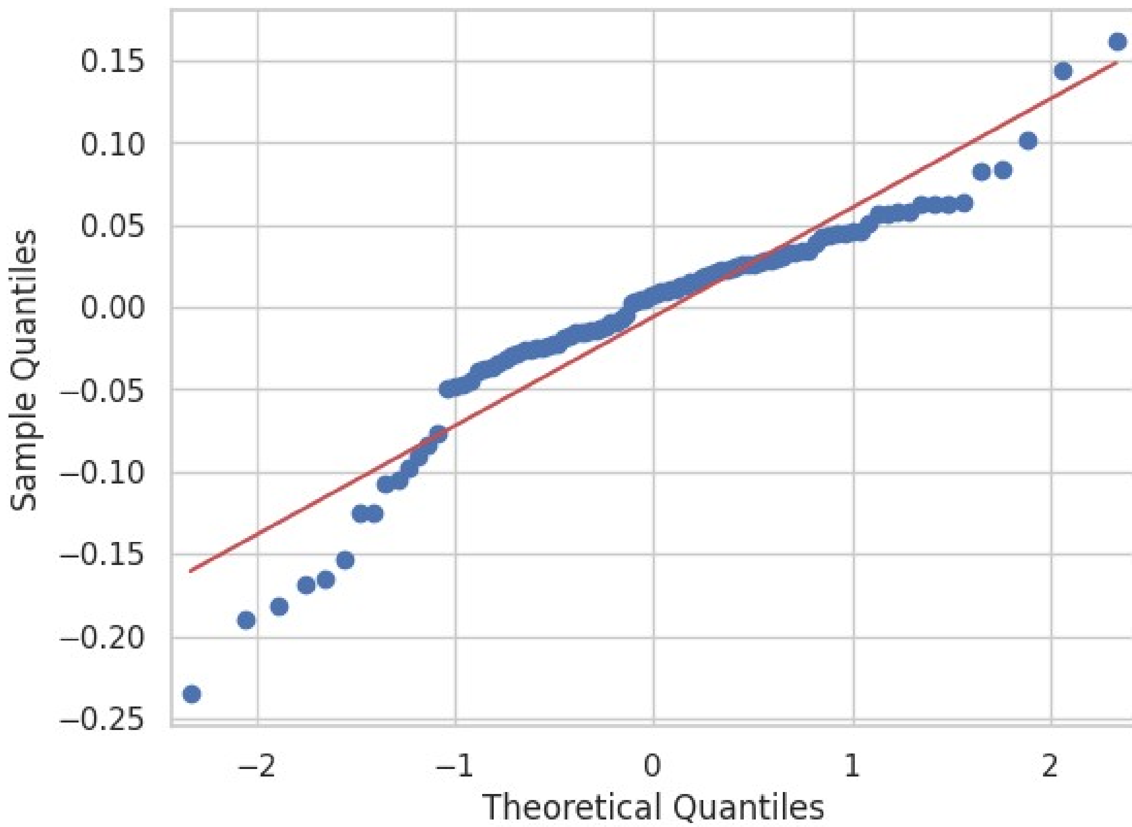


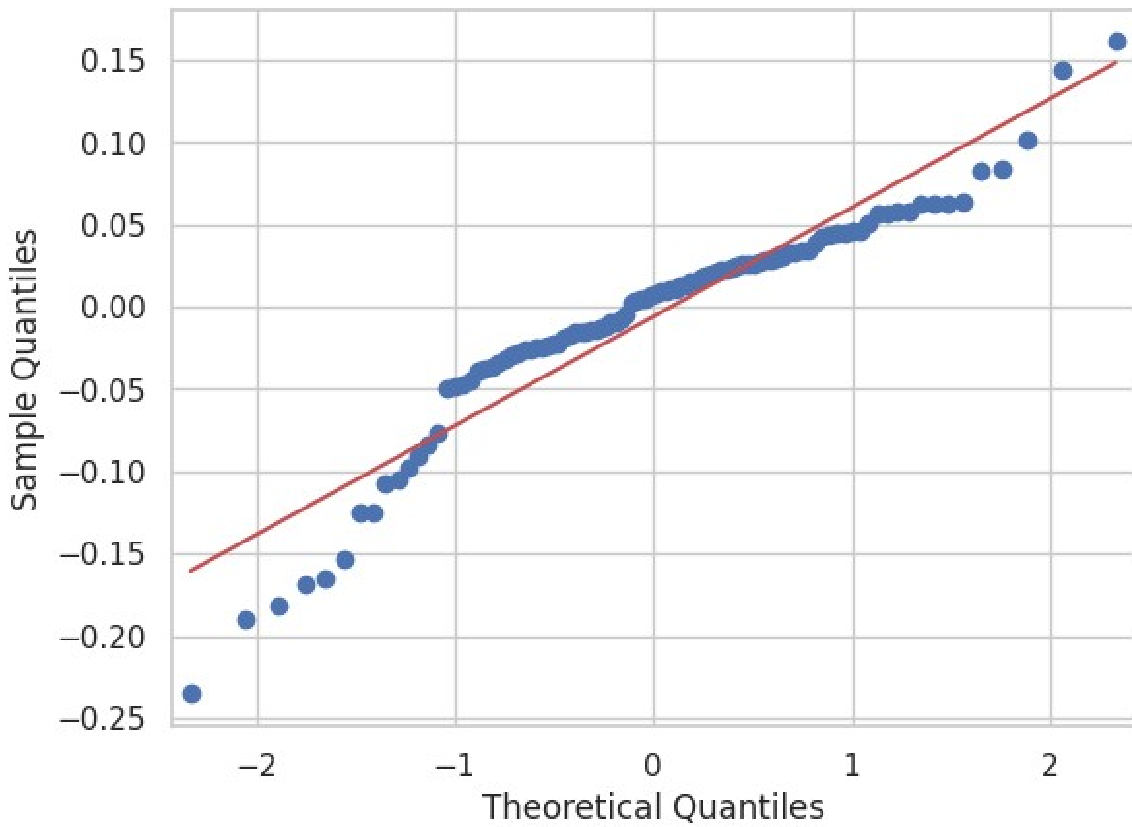
0.1
0.0
−0.1
−0.2
Sample Quantiles
−2
−1
0
1
2
Theoretical Quantiles



Q-Q plot (residuals on test data)

```
qqplot(error_test, line='s')
```


0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
Sample Quantiles
−2
−1
0
1
2
Theoretical Quantiles



Test for Homoscedasticity

- Null Hypothesis: Heteroscedasticity is not present.
- Alternate Hypothesis: Heteroscedasticity is present.

```
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', 1.0772994279987238), ('p-value',
0.30323276479815664)]
```

Model performance evaluation

- Metrics checked - MAE, RMSE, R2, Adj R2

```
from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error
```

For Test data

```
print("r2 score: ", r2_score(Y_test,Y_test_pred))
print("mean squared error: ", mean_squared_error(Y_test,Y_test_pred))
print("mean absolute error: ",
mean_absolute_error(Y_test,Y_test_pred))

r2 score:  0.7927524897595928
mean squared error:  0.004429285498957571
mean absolute error:  0.04730057428620608
```

Train data

```
print(result.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:               Sun, 07 Jan 2024   Prob (F-statistic):
3.33e-146
Time:                       04:48:29   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                 0.3450      0.010     34.259      0.000
0.325       0.365
GRE Score             0.1067      0.027      3.893      0.000
0.053       0.161
TOEFL Score           0.0826      0.027      3.024      0.003
```

```
0.029       0.136
University Rating        0.0194      0.016      1.185      0.237      -
0.013       0.052
SOP                      0.0084      0.020      0.428      0.669      -
0.030       0.047
LOR                      0.0744      0.018      4.131      0.000
0.039       0.110
CGPA                     0.3537      0.033     10.633      0.000
0.288       0.419
Research                 0.0247      0.007      3.476      0.001
0.011       0.039
======================================================================
========
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.
23.4
======================================================================
========

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

Based on significance level we can drop SOP and University rating

```
X_sm_SOP_Urating= X_sm.drop(columns=["SOP", "University Rating"])

model1= sm.OLS(Y_train, X_sm_SOP_Urating)
result1=model1.fit()
print(result1.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:               Sun, 07 Jan 2024   Prob (F-statistic):
2.65e-148
Time:                       04:58:07   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           0.3425      0.010     34.923      0.000       0.323
0.362
GRE Score       0.1086      0.027      3.970      0.000       0.055
0.162
TOEFL Score     0.0887      0.027      3.298      0.001       0.036
0.142
LOR             0.0836      0.017      5.050      0.000       0.051
0.116
CGPA            0.3667      0.032     11.465      0.000       0.304
0.430
Research        0.0254      0.007      3.573      0.000       0.011
0.039
======================================================================
========
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.
20.6
======================================================================
========

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

Metric values after dropping SOP and University rating

```
Y_pred_new= result1.predict(X_sm_SOP_Urating)
print("r2 score: ", r2_score(Y_train,Y_pred_new))
print("mean squared error: ", mean_squared_error(Y_train,Y_pred_new))
```

```
print("mean absolute error: ",
mean_absolute_error(Y_train,Y_pred_new))

r2 score:  0.8283544154206002
mean squared error:  0.003348330647824238
mean absolute error:  0.04166067850716051
```

Recommendations

- Important features to increase chances of admit are a good CGPA, GRE, TOEFL score respectively
- The model predicts with an accurancy of 82%
- University rating and SOP are insignificant features

```
```