Jamboree Education - Linear Regression

June 7, 2024

[1]:
```
import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import r2_score

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

[2]:
```
df = pd.read_csv("/Users/senth\Downloads/Jamboree_Admission.csv")
```

[3]:
```
df.head()
```

[3]:
```
   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
```

[4]:
```
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
```

There are no missing values present in the dataset.

[12]:
```
cat_cols = ['University Rating', 'SOP', 'LOR ', 'Research']
num_cols = ['GRE Score', 'TOEFL Score', 'CGPA']
target = 'Chance of Admit '
```

[7]:
```
df.describe()
```

[7]:
```
       Serial No.   GRE Score  TOEFL Score  University Rating         SOP  \
count  500.000000  500.000000   500.000000         500.000000  500.000000
mean   250.500000  316.472000   107.192000           3.114000    3.374000
std    144.481833   11.295148     6.081868           1.143512    0.991004
min      1.000000  290.000000    92.000000           1.000000    1.000000
25%    125.750000  308.000000   103.000000           2.000000    2.500000
50%    250.500000  317.000000   107.000000           3.000000    3.500000
75%    375.250000  325.000000   112.000000           4.000000    4.000000
max    500.000000  340.000000   120.000000           5.000000    5.000000

            LOR         CGPA    Research  Chance of Admit
count  500.00000  500.000000  500.000000         500.00000
mean     3.48400    8.576440    0.560000           0.72174
std      0.92545    0.604813    0.496884           0.14114
min      1.00000    6.800000    0.000000           0.34000
25%      3.00000    8.127500    0.000000           0.63000
50%      3.50000    8.560000    1.000000           0.72000
75%      4.00000    9.040000    1.000000           0.82000
max      5.00000    9.920000    1.000000           0.97000
```

[8]:
```
df.isnull().sum()
```

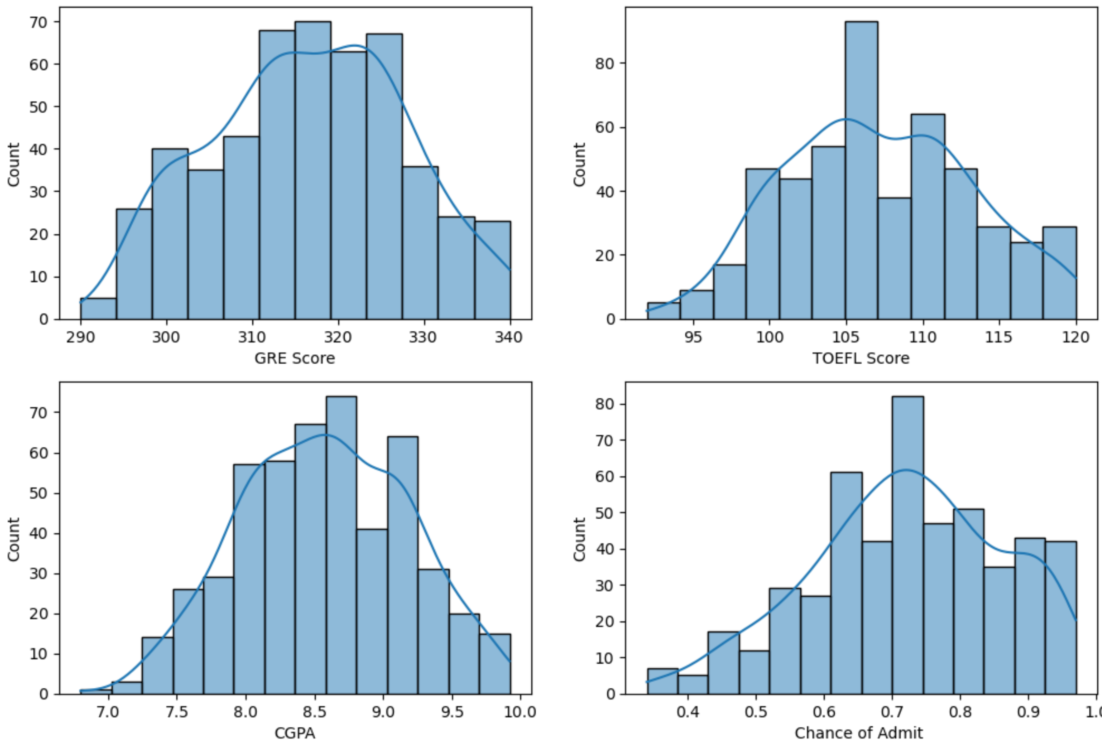
[8]:
```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

[10]:
```
df.shape
```

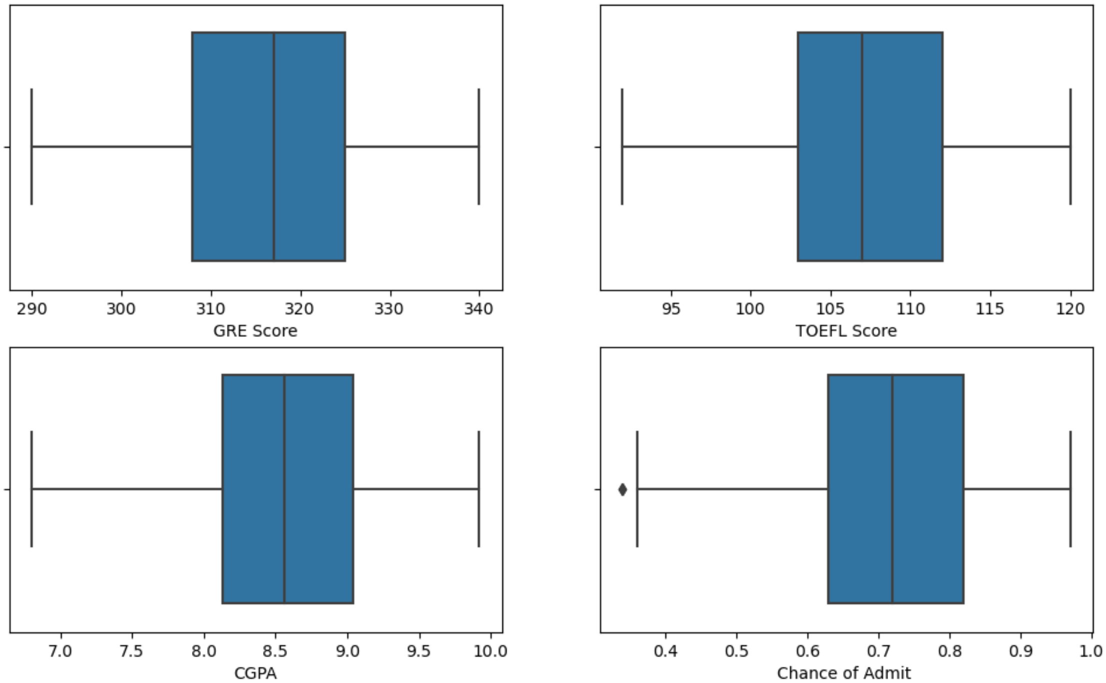
[10]:
```
(500, 9)
```

1 Univariate Analysis

[13]:
```
rows, cols = 2, 2
fig, axs = plt.subplots(rows,cols, figsize=(12, 8))
index = 0
for row in range(rows):
    for col in range(cols):
        sns.histplot(df[num_cols[index]], kde=True, ax=axs[row,col])
        index += 1
    break

sns.histplot(df[num_cols[-1]], kde=True, ax=axs[1,0])
sns.histplot(df[target], kde=True, ax=axs[1,1])
plt.show()
```



[14]:
```
rows, cols = 2, 2
fig, axs = plt.subplots(rows, cols, figsize=(12, 7))

index = 0
for col in range(cols):
    sns.boxplot(x=num_cols[index], data=df, ax=axs[0,index])
    index += 1

sns.boxplot(x=num_cols[-1], data=df, ax=axs[1,0])
sns.boxplot(x=target, data=df, ax=axs[1,1])
plt.show()
```

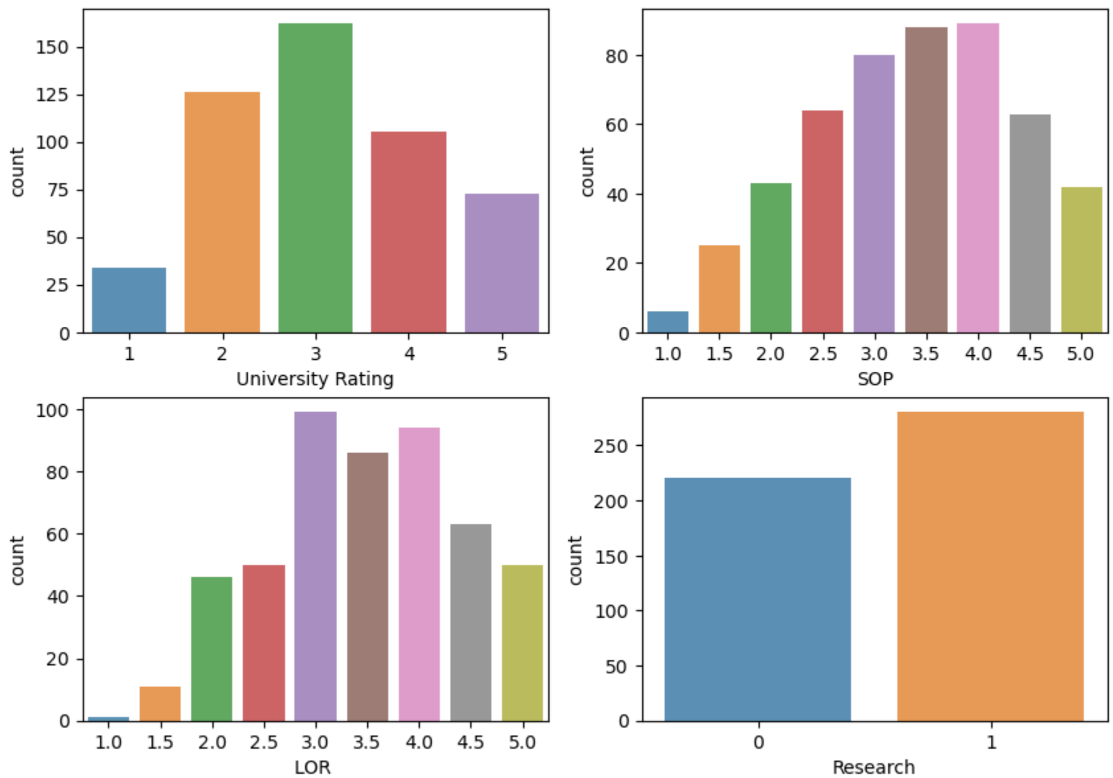


There are no outliers present in the dataset.

[15]:
```
for col in cat_cols:
    print("Column:  {:18}   Unique values: {}".format(col, df[col].nunique()))
```

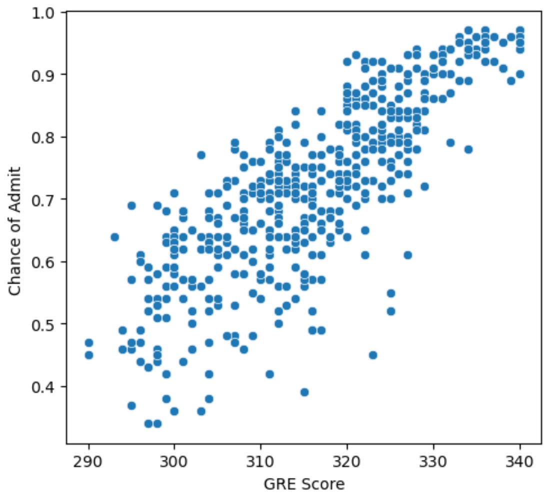
```
Column:  University Rating    Unique values: 5
Column:  SOP                  Unique values: 9
Column:  LOR                  Unique values: 9
Column:  Research             Unique values: 2
```

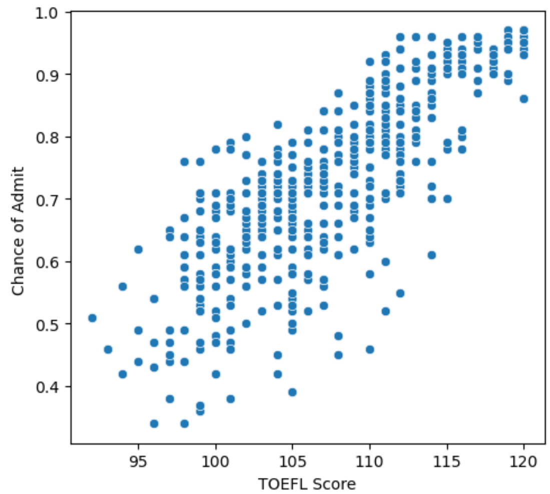
[16]:
```
cols, rows = 2, 2
fig, axs = plt.subplots(rows, cols, figsize=(10, 7))

index = 0
for row in range(rows):
    for col in range(cols):
        sns.countplot(x=cat_cols[index], data=df, ax=axs[row, col], alpha=0.8)
        index += 1
plt.show()
```

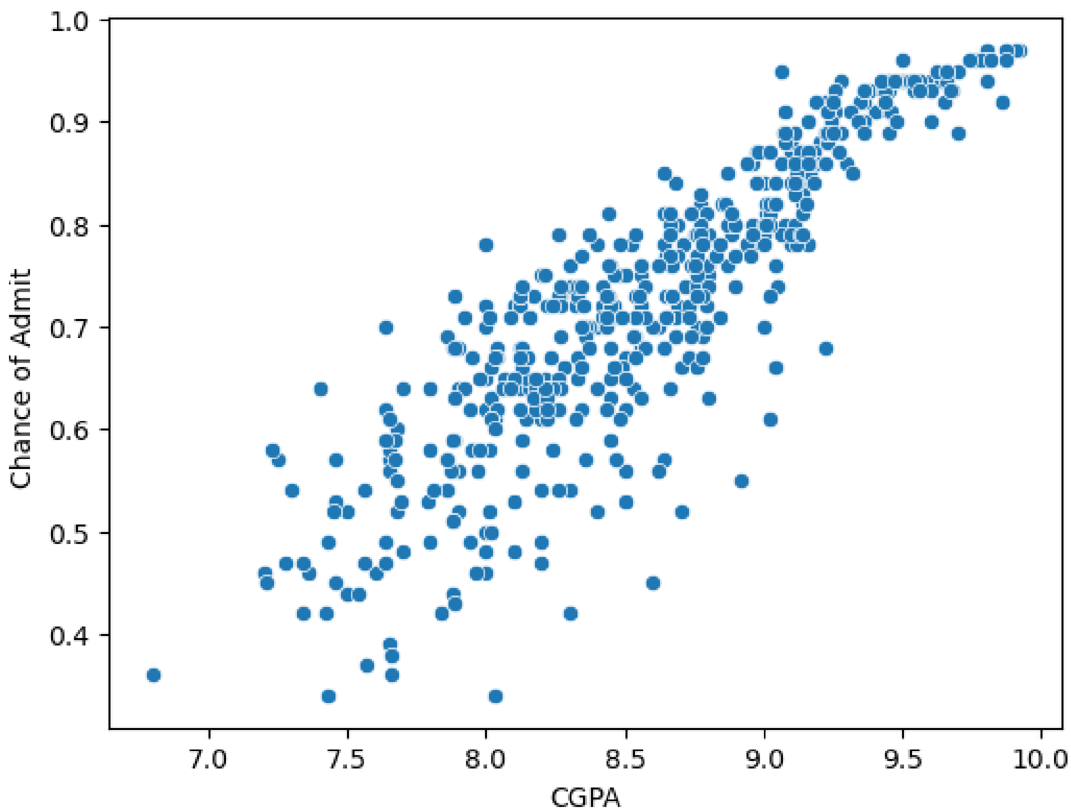


2 Bivariate Analysis

[17]:
```
fig, axs = plt.subplots(1, 2, figsize=(12,5))

sns.scatterplot(x=num_cols[0], y=target, data=df, ax=axs[0])
sns.scatterplot(x=num_cols[1], y=target, data=df, ax=axs[1])
plt.show()
sns.scatterplot(x=num_cols[2], y=target, data=df)
plt.show()
```



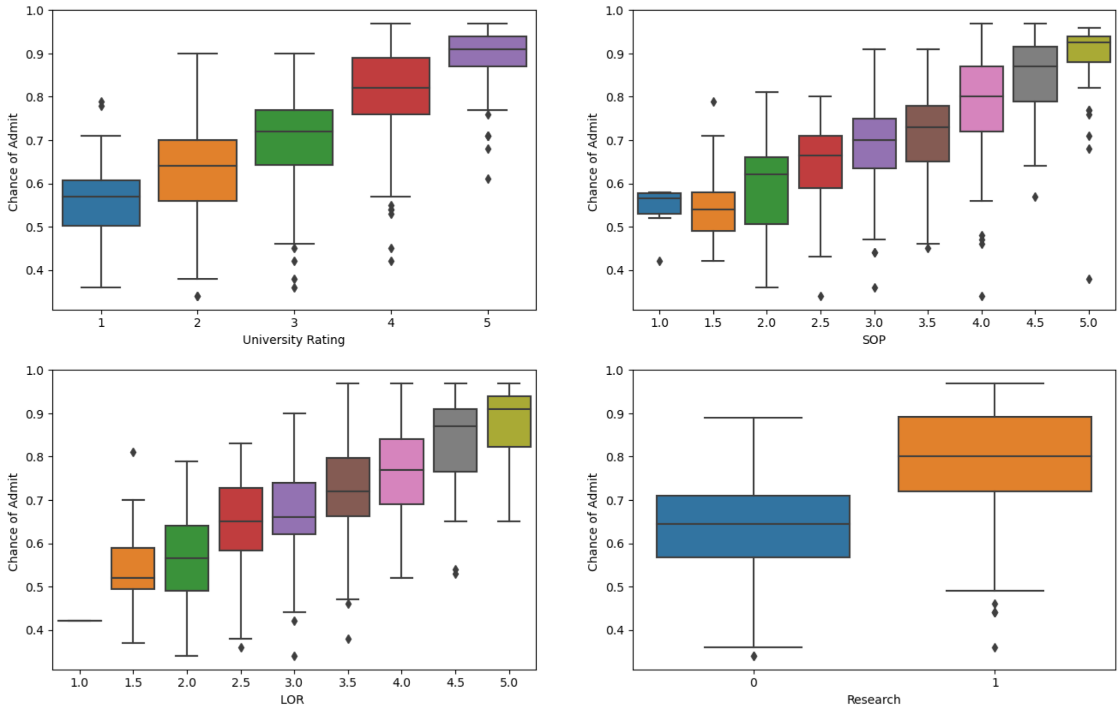




Seems like there is a linear correlation between the continuous variables and the target variable.

[18]:
```
rows, cols = 2,2
fig, axs = plt.subplots(rows, cols, figsize=(16,10))
```

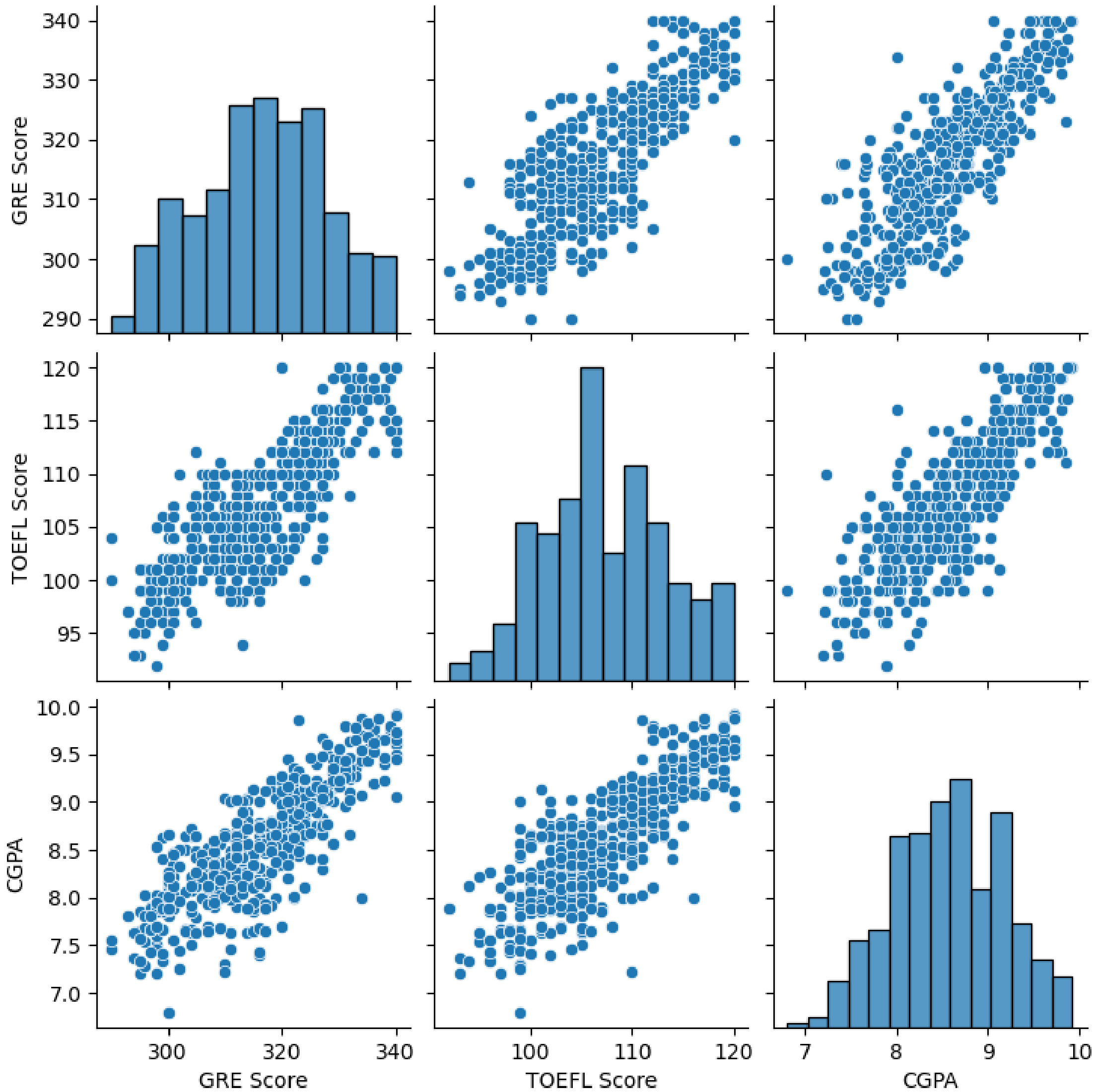
```
index = 0
for row in range(rows):
    for col in range(cols):
        sns.boxplot(x=cat_cols[index], y=target, data=df, ax=axs[row,col])
        index += 1
```



- As you can see from the graphs, as tge rating increases the Chance of Admit also increases.
- Students who have the research experience have more chances of Admin as compared to other students who don't have the research experience.

3 Multivariate Analysis

[19]:
```
sns.pairplot(df[num_cols])
plt.show()
```



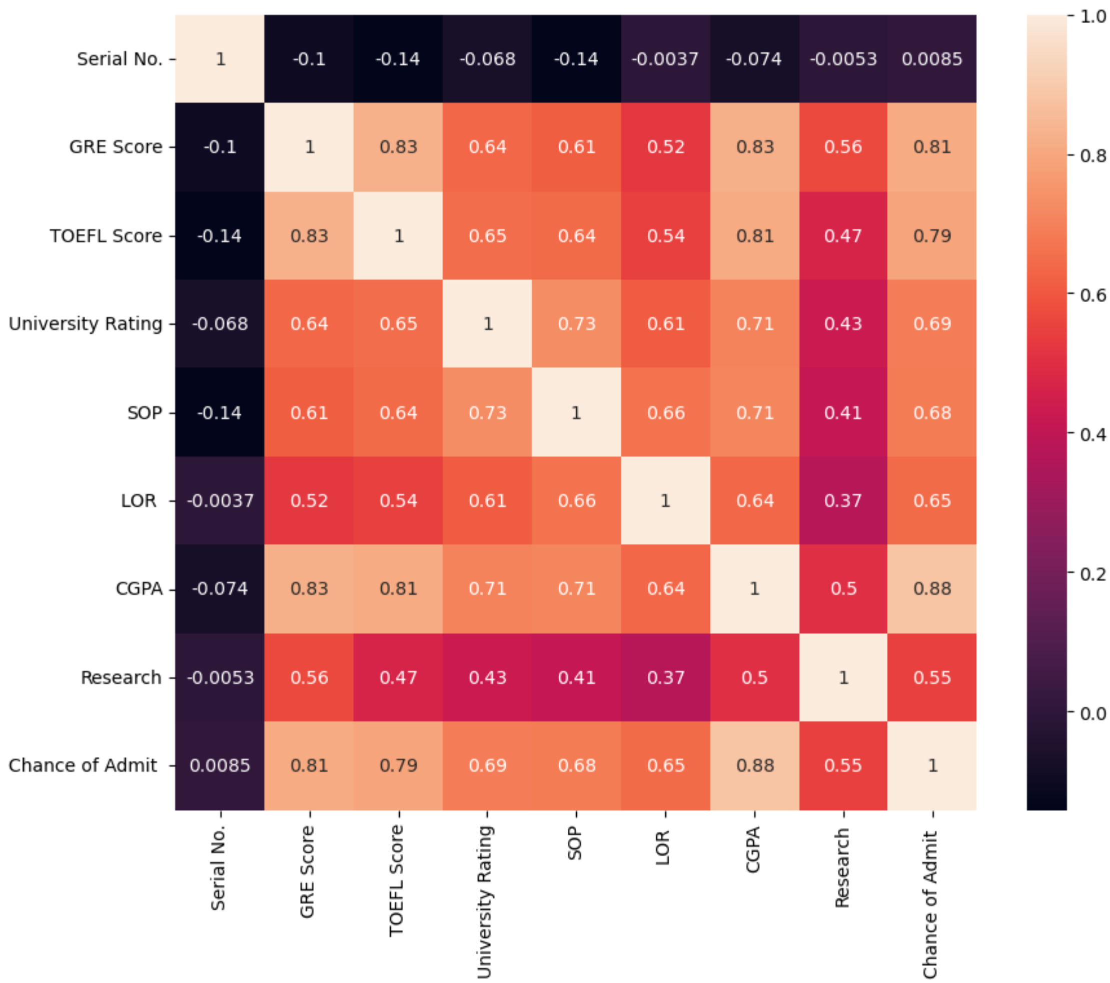
Independent continuous variables are also correlated with each other.

[20]:
```
df.corr()
```

[20]:
```
                   Serial No.  GRE Score  TOEFL Score  University Rating  \
Serial No.           1.000000  -0.103839    -0.141696          -0.067641
GRE Score           -0.103839   1.000000     0.827200           0.635376
TOEFL Score         -0.141696   0.827200     1.000000           0.649799
University Rating   -0.067641   0.635376     0.649799           1.000000
SOP                 -0.137352   0.613498     0.644410           0.728024
LOR                 -0.003694   0.524679     0.541563           0.608651
CGPA                -0.074289   0.825878     0.810574           0.705254
Research            -0.005332   0.563398     0.467012           0.427047
```

```
Chance of Admit      0.008505   0.810351     0.792228          0.690132

                          SOP       LOR      CGPA  Research  Chance of Admit
Serial No.          -0.137352 -0.003694 -0.074289 -0.005332         0.008505
GRE Score            0.613498  0.524679  0.825878  0.563398         0.810351
TOEFL Score          0.644410  0.541563  0.810574  0.467012         0.792228
University Rating    0.728024  0.608651  0.705254  0.427047         0.690132
SOP                  1.000000  0.663707  0.712154  0.408116         0.684137
LOR                  0.663707  1.000000  0.637469  0.372526         0.645365
CGPA                 0.712154  0.637469  1.000000  0.501311         0.882413
Research             0.408116  0.372526  0.501311  1.000000         0.545871
Chance of Admit      0.684137  0.645365  0.882413  0.545871         1.000000
```

[23]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



4 Data Preprocessing

[25]:
```
df = df.drop(columns=['Serial No.'], axis=1)
```

[26]:
```
df.duplicated().sum()
```

[26]: 0

There are no missing values, outliers and duplicates present in the dataset.

5 Data preparation for model building

[27]:
```
X = df.drop(columns=[target])
y = df[target]
```

[28]:
```
sc = StandardScaler()
X = sc.fit_transform(X)
```

[29]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=1)
```

[30]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(350, 7) (350,)
(150, 7) (150,)
```

5.1 Model Building

[31]:
```
def adjusted_r2(r2, p, n):
    """
    n: no of samples
    p: no of predictors
    r2: r2 score
    """
    adj_r2 = 1 - ((1-r2)*(n-1) / (n-p-1))
    return adj_r2

def get_metrics(y_true, y_pred, p=None):
    n = y_true.shape[0]
    mse = np.sum((y_true - y_pred)**2) / n
    rmse = np.sqrt(mse)
    mae = np.mean(np.abs(y_true - y_pred))
    score = r2_score(y_true, y_pred)
    adj_r2 = None
    if p is not None:
        adj_r2 = adjusted_r2(score, p, n)
```

```python
    res = {
        "mean_absolute_error": round(mae, 2),
        "rmse": round(rmse, 2),
        "r2_score": round(score, 2),
        "adj_r2": round(adj_r2, 2)
    }
    return res
```

[32]:
```python
def train_model(X_train, y_train, X_test, y_test,cols, model_name="linear",␣
 ↪alpha=1.0):
    model = None
    if model_name == "lasso":
        model = Lasso(alpha=alpha)
    elif model_name == "ridge":
        model = Ridge(alpha=alpha)
    else:
        model = LinearRegression()

    model.fit(X_train, y_train)
    y_pred_train = model.predict(X_train)
    y_pred_test = model.predict(X_test)
    p = X_train.shape[1]
    train_res = get_metrics(y_train, y_pred_train, p)
    test_res = get_metrics(y_test, y_pred_test, p)

    print(f"\n----   {model_name.title()} Regression Model  ----\n")
    print(f"Train MAE: {train_res['mean_absolute_error']} Test MAE:␣
 ↪{test_res['mean_absolute_error']}")
    print(f"Train RMSE: {train_res['rmse']} Test RMSE: {test_res['rmse']}")
    print(f"Train R2_score: {train_res['r2_score']} Test R2_score:␣
 ↪{test_res['r2_score']}")
    print(f"Train Adjusted_R2: {train_res['adj_r2']} Test Adjusted_R2:␣
 ↪{test_res['adj_r2']}")
    print(f"Intercept: {model.intercept_}")
    #print(len(df.columns), len(model.coef_))
    coef_df = pd.DataFrame({"Column": cols, "Coef": model.coef_})
    print(coef_df)
    print("-"*50)
    return model
```

[33]:
```python
train_model(X_train, y_train, X_test, y_test,df.columns[:-1], "linear")
train_model(X_train, y_train, X_test, y_test,df.columns[:-1], "ridge")
train_model(X_train, y_train, X_test, y_test,df.columns[:-1], "lasso", 0.001)
```

```
----   Linear Regression Model  ----
```

```
Train MAE: 0.04 Test MAE: 0.04
Train RMSE: 0.06 Test RMSE: 0.06
Train R2_score: 0.82 Test R2_score: 0.82
Train Adjusted_R2: 0.82 Test Adjusted_R2: 0.81
Intercept: 0.724978121476996
              Column      Coef
0          GRE Score  0.018657
1        TOEFL Score  0.023176
2  University Rating  0.011565
3                SOP -0.000999
4               LOR   0.012497
5               CGPA  0.064671
6           Research  0.013968
--------------------------------------------------

----   Ridge Regression Model  ----

Train MAE: 0.04 Test MAE: 0.04
Train RMSE: 0.06 Test RMSE: 0.06
Train R2_score: 0.82 Test R2_score: 0.82
Train Adjusted_R2: 0.82 Test Adjusted_R2: 0.81
Intercept: 0.7249823645841696
              Column      Coef
0          GRE Score  0.018902
1        TOEFL Score  0.023252
2  University Rating  0.011594
3                SOP -0.000798
4               LOR   0.012539
5               CGPA  0.064004
6           Research  0.013990
--------------------------------------------------

----   Lasso Regression Model  ----

Train MAE: 0.04 Test MAE: 0.04
Train RMSE: 0.06 Test RMSE: 0.06
Train R2_score: 0.82 Test R2_score: 0.82
Train Adjusted_R2: 0.82 Test Adjusted_R2: 0.81
Intercept: 0.7249659139557142
              Column      Coef
0          GRE Score  0.018671
1        TOEFL Score  0.022770
2  University Rating  0.010909
3                SOP  0.000000
4               LOR   0.011752
5               CGPA  0.064483
6           Research  0.013401
--------------------------------------------------
```

[33]: Lasso(alpha=0.001)

- Since model is not overfitting, Results for Linear, Ridge and Lasso are the same.
- R2_score and Adjusted_r2 are almost the same. Hence there are no unnecessary independent variables in the data.

6 Linear Regression Model - Assumption Test

Mutlicollinearity Check

[34]:
```
def vif(newdf):
    vif_data = pd.DataFrame()
    vif_data["feature"] = newdf.columns
    vif_data["VIF"] = [variance_inflation_factor(newdf.values, i)
                          for i in range(len(newdf.columns))]
    return vif_data
```

[35]:
```
res = vif(df.iloc[:,:-1])
res
```

[35]:

	feature	VIF
0	GRE Score	1308.061089
1	TOEFL Score	1215.951898
2	University Rating	20.933361
3	SOP	35.265006
4	LOR	30.911476
5	CGPA	950.817985
6	Research	2.869493

[36]:
```
res = vif(df.iloc[:, 1:-1])
res
```

[36]:

	feature	VIF
0	TOEFL Score	639.741892
1	University Rating	19.884298
2	SOP	33.733613
3	LOR	30.631503
4	CGPA	728.778312
5	Research	2.863301

[37]:
```
res = vif(df.iloc[:,2:-1])
res
```

[37]:

	feature	VIF
0	University Rating	19.777410
1	SOP	33.625178
2	LOR	30.356252
3	CGPA	25.101796

```
4           Research   2.842227
```

[38]:
```
res = vif(df.iloc[:,2:-1].drop(columns=['SOP']))
res
```

[38]:

	feature	VIF
0	University Rating	15.140770
1	LOR	26.918495
2	CGPA	22.369655
3	Research	2.819171

[39]:
```
newdf = df.iloc[:,2:-1].drop(columns=['SOP'])
newdf = newdf.drop(columns=['LOR '], axis=1)
res = vif(newdf)
res
```

[39]:

	feature	VIF
0	University Rating	12.498400
1	CGPA	11.040746
2	Research	2.783179

[40]:
```
newdf = newdf.drop(columns=['University Rating'])
res = vif(newdf)
res
```

[40]:

	feature	VIF
0	CGPA	2.455008
1	Research	2.455008

[41]:
```
X = df[['CGPA', 'Research']]
sc = StandardScaler()
X = sc.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=1)
```

[42]:
```
model = train_model(X_train, y_train, X_test, y_test, ['CGPA', 'Research'],␣
 ↪"linear")
train_model(X_train, y_train, X_test, y_test, ['CGPA', 'Research'], "ridge")
train_model(X_train, y_train, X_test, y_test, ['CGPA', 'Research'], "lasso", 0.
 ↪001)
```

```
----   Linear Regression Model  ----

Train MAE: 0.05 Test MAE: 0.05
Train RMSE: 0.06 Test RMSE: 0.07
Train R2_score: 0.78 Test R2_score: 0.81
```

```
Train Adjusted_R2: 0.78 Test Adjusted_R2: 0.81
Intercept: 0.7247774222727991
     Column      Coef
0      CGPA  0.112050
1  Research  0.020205
--------------------------------------------------

----   Ridge Regression Model  ----

Train MAE: 0.05 Test MAE: 0.05
Train RMSE: 0.06 Test RMSE: 0.07
Train R2_score: 0.78 Test R2_score: 0.81
Train Adjusted_R2: 0.78 Test Adjusted_R2: 0.81
Intercept: 0.7247830300095277
     Column      Coef
0      CGPA  0.111630
1  Research  0.020362
--------------------------------------------------

----   Lasso Regression Model  ----

Train MAE: 0.05 Test MAE: 0.05
Train RMSE: 0.06 Test RMSE: 0.07
Train R2_score: 0.78 Test R2_score: 0.81
Train Adjusted_R2: 0.78 Test Adjusted_R2: 0.81
Intercept: 0.7247713356661623
     Column      Coef
0      CGPA  0.111344
1  Research  0.019571
--------------------------------------------------
```

[42]:

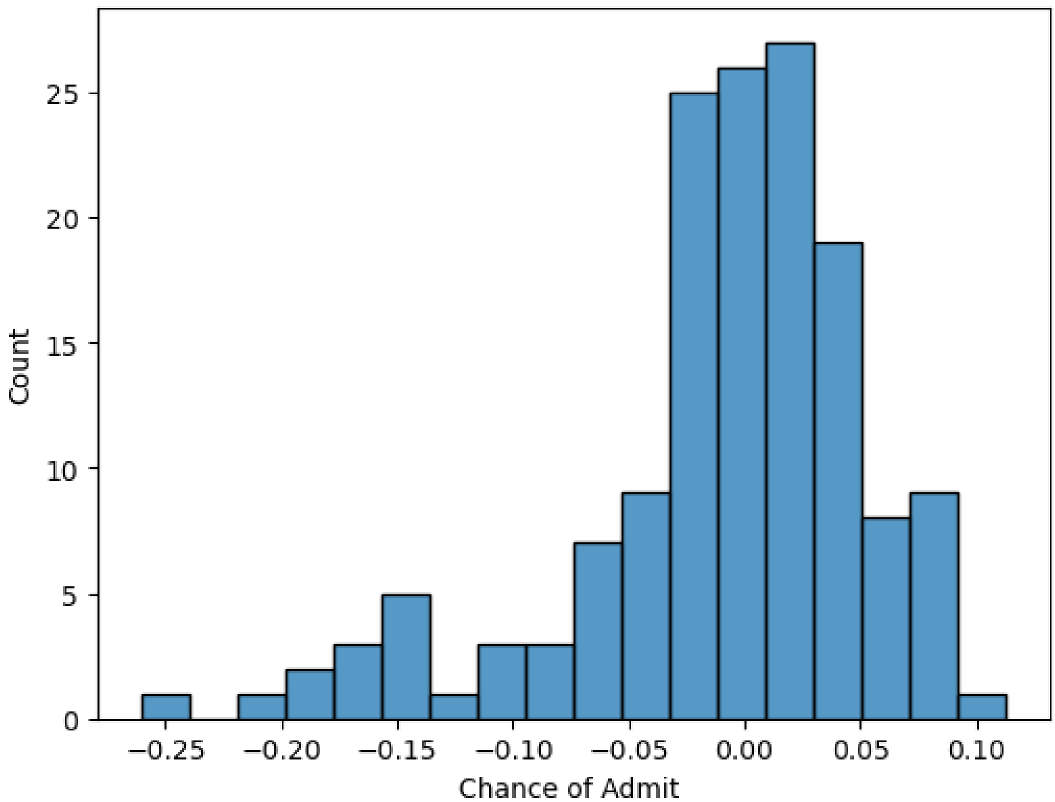
```
Lasso(alpha=0.001)
```

After removing collinear features using VIF and using only two features. R2_score and Adjusted_r2 are still the same as before the testing dataset.

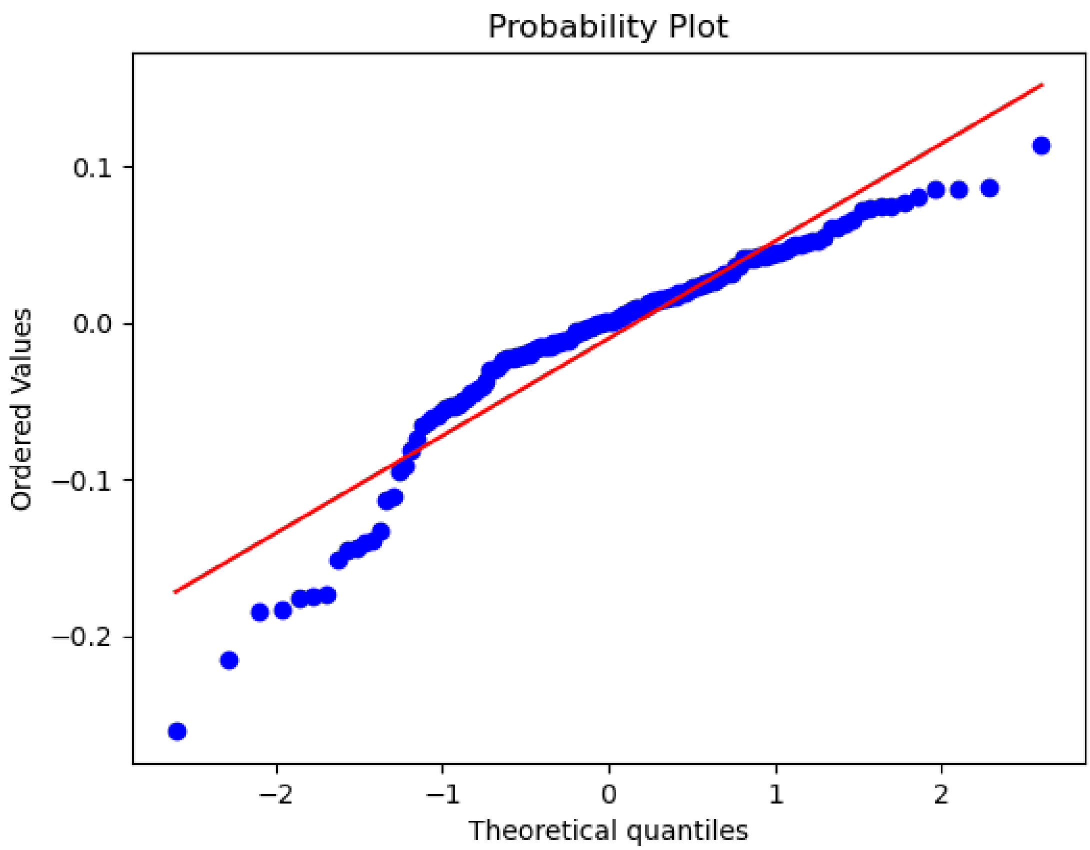
Normality of Residuals

[43]:

```python
y_pred = model.predict(X_test)
residuals = (y_test - y_pred)
sns.histplot(residuals)
plt.show()
```

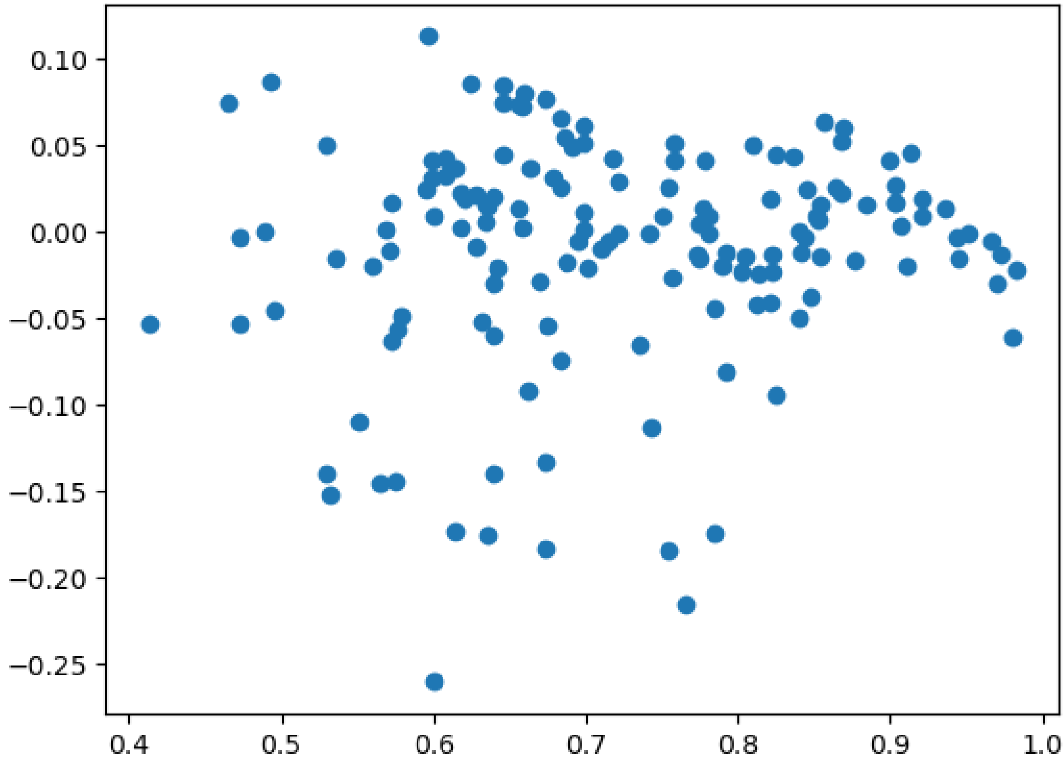


[44]:
```
stats.probplot(residuals,plot=plt)
plt.show()
```



Test for Homoscedasticity

[45]:
```
plt.scatter(y_pred,residuals)
plt.show()
```



- Since the plot is not creating a cone type shape. Hence there is no homoscedasticity present in the data.

7 Insights

- Multicollinearity is present in the data.
- After removing collinear features there are only two variables which are important in making predictions for the target variables.
- Indepedent variables are linearly correlated with dependent variables.

8 Recommendations

1. CGPA and Research are the only two variables which are important in making the prediction for Chance of Admit.
2. CGPA is the most important varibale in making the prediction for the Chance of Admit.
3. Following are the final model results on the test data:
   - RMSE: 0.07
   - MAE: 0.05 -R2_score: 0.81
   - Adjusted_R2: 0.81

[ ]: