Jamboree Education

Jamboree has helped thousands of students like you make it to top colleges abroad. Be it GMAT, GRE or SAT, their unique problem-solving methods ensure maximum scores with minimum effort. They recently launched a feature where students/learners can come to their website and check their probability of getting into the IVY league college. This feature estimates the chances of graduate admission from an Indian perspective.

Column Profiling:

- Serial No. (Unique row ID)
- GRE Scores (out of 340)
- TOEFL Scores (out of 120)
- University Rating (out of 5)
- Statement of Purpose and Letter of Recommendation Strength (out of 5)
- Undergraduate GPA (out of 10)
- Research Experience (either 0 or 1)
- Chance of Admit (ranging from 0 to 1)

Problem Statement: Analysis will 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.

```python
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import r2_score
```

```
df = pd.read_csv("Jamboree_Admission.csv")
print(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 = df.drop(columns = ["Serial No."])
Since Serial No. is unnecessary field, we can drop this column.

print(df.shape)
```

```
(500, 8)
```

```
print(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
```

```
print(df.describe())
```

```
        GRE Score  TOEFL Score  University Rating         SOP        LOR   \
count  500.000000   500.000000         500.000000  500.000000  500.00000
mean   316.472000   107.192000           3.114000    3.374000    3.48400
std     11.295148     6.081868           1.143512    0.991004    0.92545
min    290.000000    92.000000           1.000000    1.000000    1.00000
25%    308.000000   103.000000           2.000000    2.500000    3.00000
50%    317.000000   107.000000           3.000000    3.500000    3.50000
75%    325.000000   112.000000           4.000000    4.000000    4.00000
max    340.000000   120.000000           5.000000    5.000000    5.00000

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

```
print(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 are no missing values present in the dataset.

```
print(df.duplicated().sum())
```

```
0
```

- There are no duplicate values in the data

```
all_num = df.keys()
print(all_num)
```

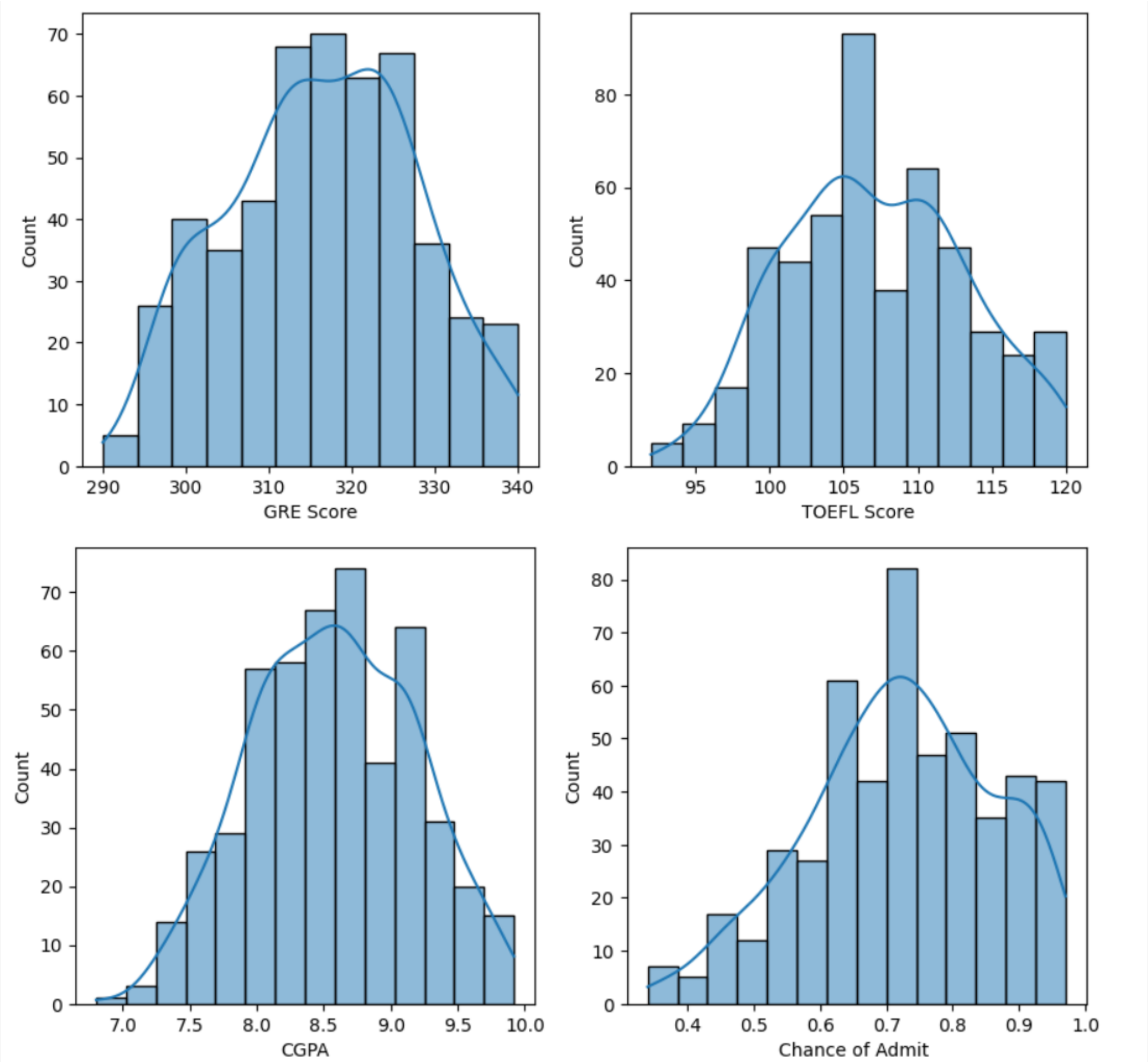
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

```
num = ["GRE Score" , "TOEFL Score" , "CGPA" , "Chance of Admit "]
cat = ["University Rating" , "SOP" , "LOR " , "Research"]

for i in all_num :
   if i in (df.iloc[: , [2,3,4,6]].columns) :
      print()
      print("Unique categories in" ,i,": " , df[i].unique())

   print("number of unique values in" ,i,": " , df[i].nunique())
```

```
number of unique values in GRE Score :  49
number of unique values in TOEFL Score :  29

Unique categories in University Rating :  [4 3 2 5 1]
number of unique values in University Rating :  5

Unique categories in SOP :  [4.5 4.  3.  3.5 2.  5.  1.5 1.  2.5]
number of unique values in SOP :  9

Unique categories in LOR  :  [4.5 3.5 2.5 3.  4.  1.5 2.  5.  1. ]
number of unique values in LOR  :  9
number of unique values in CGPA :  184

Unique categories in Research :  [1 0]
number of unique values in Research :  2
number of unique values in Chance of Admit  :  61
```

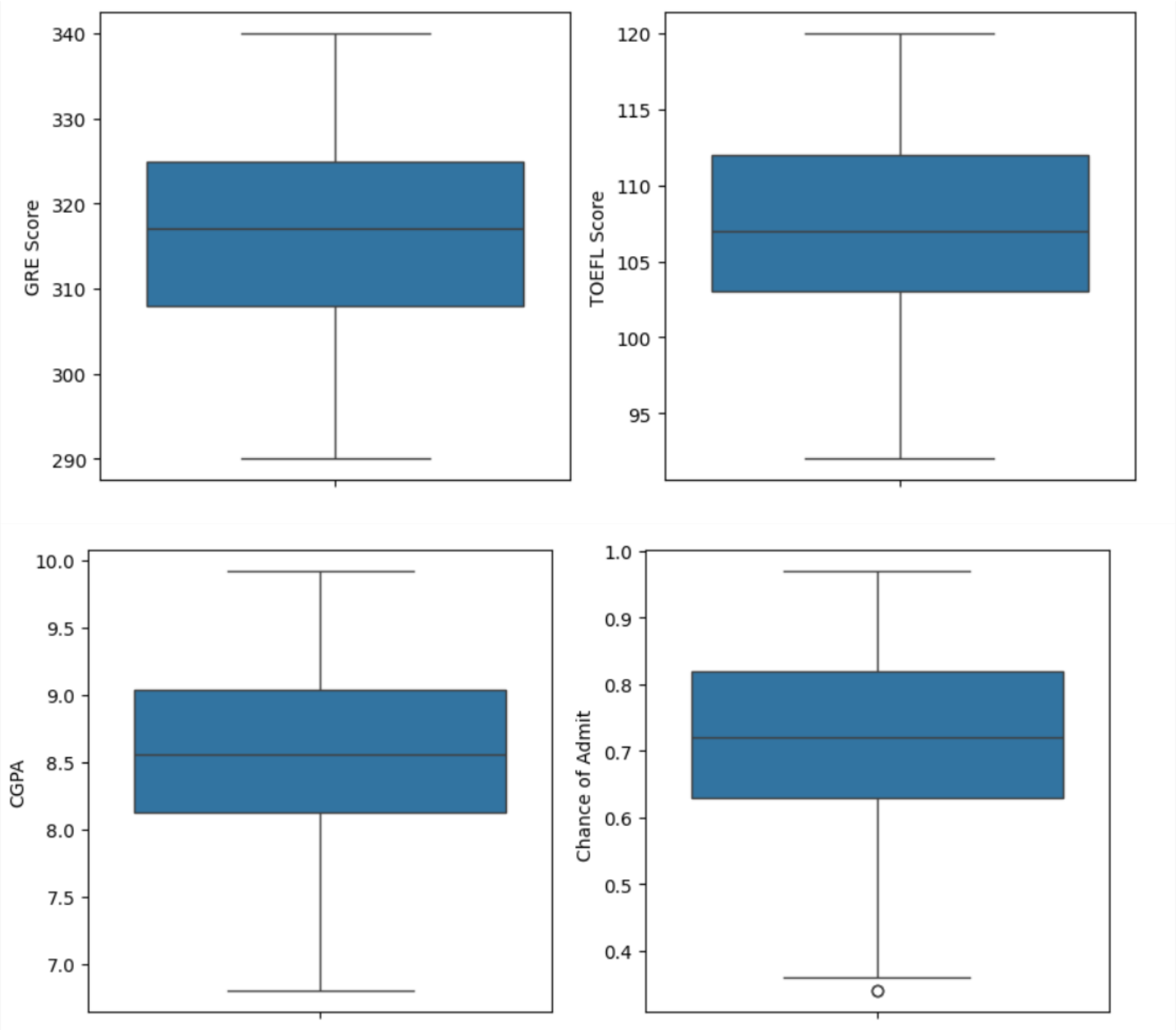
Univariate Analysis:

```
# Histplot
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(10, 10))
count = 0

for i in range(2) :
   for j in range(2) :
      sns.histplot(x = df[num[count]] , ax = ax[i , j] , kde = True)
      count += 1
plt.show()
```



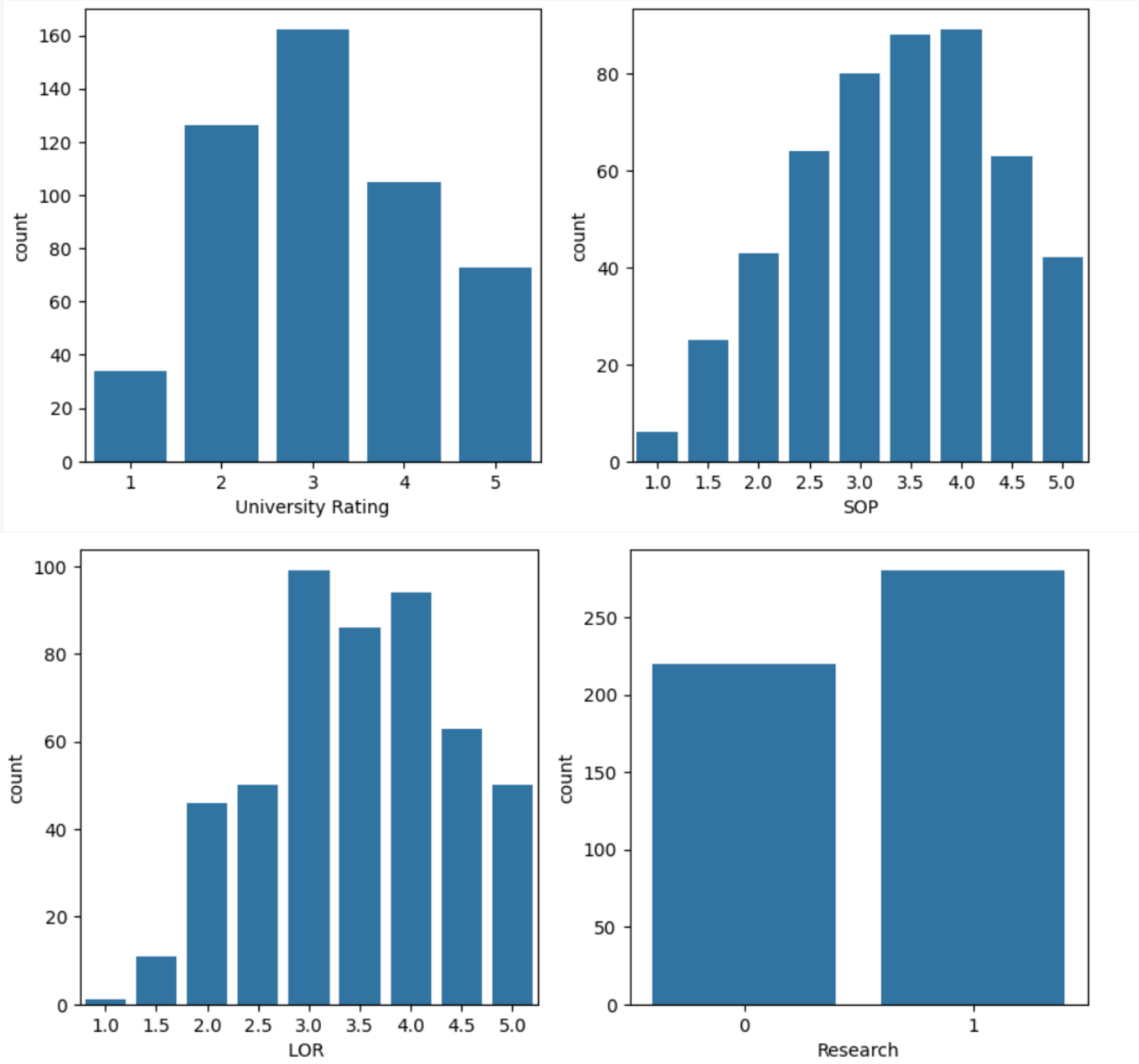
- As we can see numerical variables are following normal distribution.

```
# Boxplot
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(10, 10))
count = 0
for i in range(2) :
    for j in range(2) :
        sns.boxplot(y = df[num[count]] , ax = ax[i , j])
        count += 1
plt.show()
```



- There are no outliers in the data.

```
# Countplot
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(10, 10))
count = 0
for i in range(2) :
   for j in range(2) :
      sns.countplot(x = df[cat[count]] , ax = ax[i , j])
      count += 1
plt.show()
```

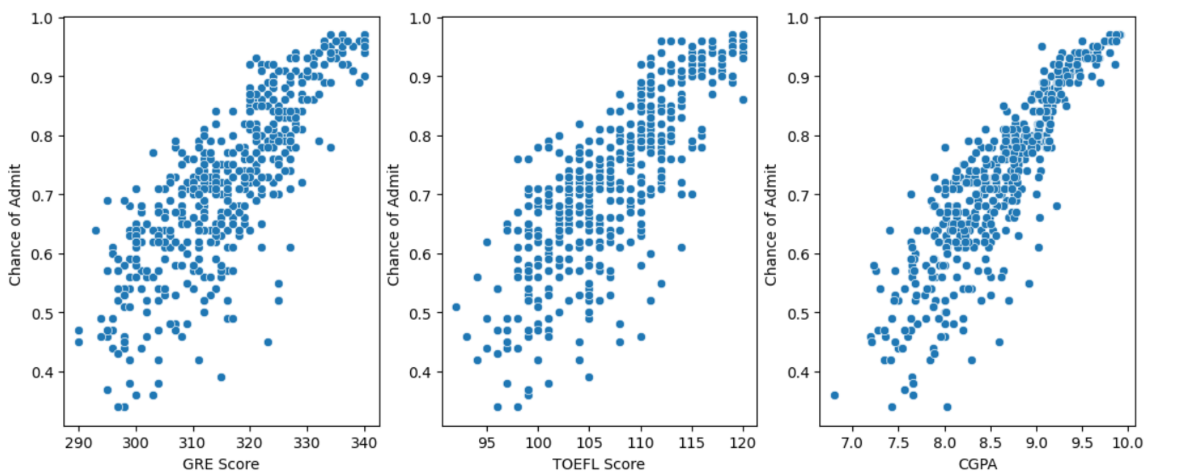


- More data is of university rating 3, SOP 4, LOR 3 and Research 1.

Bivariate Analysis

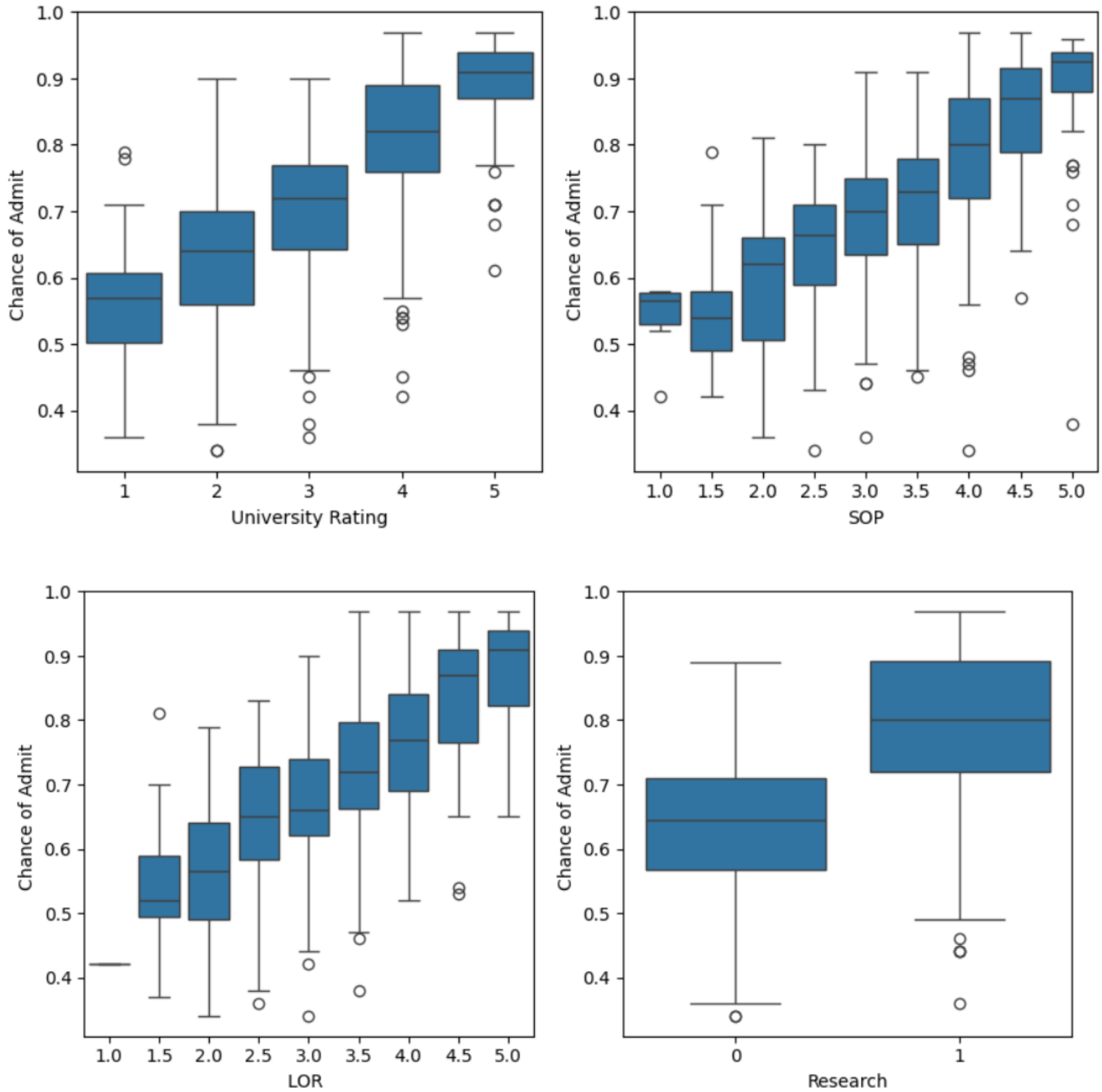
```
num = ["GRE Score" , "TOEFL Score" , "CGPA"]
cat = ["University Rating" , "SOP" , "LOR " , "Research"]

plt.figure(figsize = (13 , 5))
plt.subplot(1,3,1)
sns.scatterplot(x = df["GRE Score"] , y = df["Chance of Admit "])

plt.subplot(1,3,2)
sns.scatterplot(x = df["TOEFL Score"] , y = df["Chance of Admit "])
```

```
plt.subplot(1,3,3)
sns.scatterplot(x = df["CGPA"] , y = df["Chance of Admit "])
plt.show()
```



- Independent variables and dependent variables collinear with each other.

```
# Boxplot
fig , ax = plt.subplots(nrows = 2 , ncols = 2 , figsize = (10, 10))
count = 0
for i in range(2) :
  for j in range(2) :
    sns.boxplot(x = df[cat[count]] , y = df["Chance of Admit "] , ax =
ax[i , j])
    count += 1
plt.show()
```

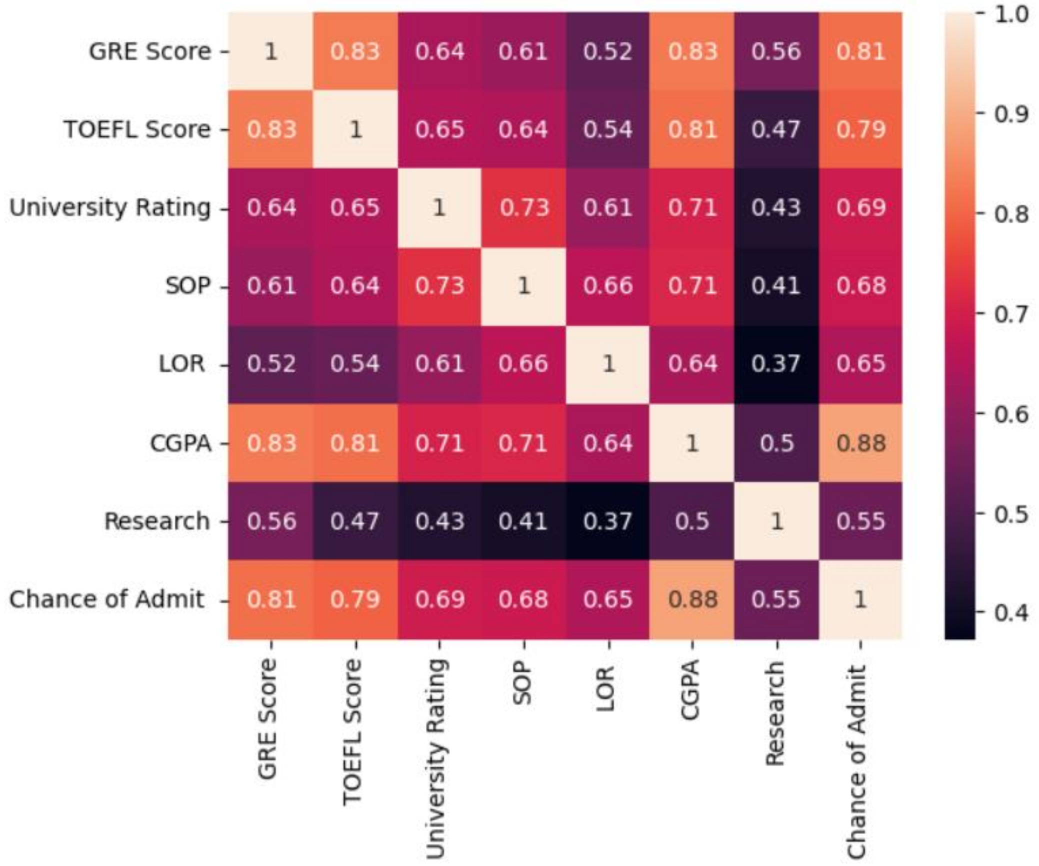


- As independent variables increase chances of Admit also increase.

Multivariate Analysis

```
# Heatmap


df.corr()
```

```python
sns.heatmap(df.corr() , annot = True)
plt.show()
```



Pre-processing:

```python
print(df.duplicated().sum())
```

```
0
```

```python
print(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
```

```python
X = df.drop(columns = ["Chance of Admit "] )
y = df["Chance of Admit "]
```

```
scale = StandardScaler()

X = scale.fit_transform(X)

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

print(f"Shape of train data is: {X_train.shape}")
 Shape of train data is: (400, 7)


print(f"Shape of test data is: {X_test.shape}")
  Shape of test data is: (100, 7)
```

Model Building

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

def get_metrics(y_true, y_pred, d=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 = 1 - (np.sum((y_true - y_pred)**2) / np.sum((y_true -
np.mean(y_pred))**2))
    score = r2_score(y_true, y_pred)
    adj_r2 = None
    if d is not None:
        adj_r2 = adjusted_r2(score, d, n)
```

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


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)
    d = X_train.shape[1]
    train_res = get_metrics(y_train, y_pred_train, d)
    test_res = get_metrics(y_test, y_pred_test, d )

    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



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: 0.05
Train RMSE: 0.06 & Test RMSE: 0.07
Train R2_score: 0.83 & Test R2_score: 0.79
Train Adjusted_R2: 0.83 & Test Adjusted_R2: 0.78
Intercept: 0.7229601834968221
              Column      Coef
0          GRE Score  0.024081
1        TOEFL Score  0.017928
2  University Rating  0.005532
3                SOP  0.002075
4               LOR   0.017196
5               CGPA  0.068494
6           Research  0.012267
--------------------------------------------------

 ----   Ridge Regression Model  ----

 Train MAE: 0.04 & Test MAE: 0.05
 Train RMSE: 0.06 & Test RMSE: 0.07
 Train R2_score: 0.83 & Test R2_score: 0.79
 Train Adjusted_R2: 0.83 & Test Adjusted_R2: 0.78
 Intercept: 0.7229606078169396
              Column      Coef
0          GRE Score  0.024203
1        TOEFL Score  0.018104
2  University Rating  0.005639
3                SOP  0.002210
4               LOR   0.017221
5               CGPA  0.067878
6           Research  0.012292
 --------------------------------------------------

 ----   Lasso Regression Model  ----

 Train MAE: 0.04 & Test MAE: 0.05
 Train RMSE: 0.06 & Test RMSE: 0.07
 Train R2_score: 0.83 & Test R2_score: 0.79
 Train Adjusted_R2: 0.83 & Test Adjusted_R2: 0.78
 Intercept: 0.7229556414439673
              Column      Coef
0          GRE Score  0.023837
1        TOEFL Score  0.017683
2  University Rating  0.005358
3                SOP  0.001827
4               LOR   0.016727
5               CGPA  0.068736
6           Research  0.011718
 --------------------------------------------------
```

- Since model is not overfitting, Results for Linear, Ridge and Lasso are the same.
- R2_score and Adj_r2 are almost the same. Hence there are no unnecessary independent variables in the data.

Testing the assumptions of Linear Regression Model

1. Multicollinearity Check:

```
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))]
    vif_data = vif_data.sort_values(["VIF"] , ascending = False)
    return vif_data


result = vif(df.iloc[: , :-1])
print(result)
```

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

```
result = vif(df.iloc[: , 1:-1])
print(result)
```

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

```
result = vif(df.iloc[: , 2:-1])
print(result)
```

```
                feature        VIF
1                   SOP  33.625178
2                   LOR  30.356252
3                  CGPA  25.101796
0  University Rating  19.777410
4              Research   2.842227
```

```
result = vif(df.iloc[: , 3:-1])
print(result)
```

```
    feature        VIF
1       LOR  29.358881
0       SOP  25.742050
2      CGPA  25.012564
3  Research   2.744550
```

```
result = vif(df.iloc[: , 4:-1])
print(result)
```

```
    feature        VIF
0       LOR  22.220673
1      CGPA  20.791852
2  Research   2.624493
```

```
result = vif(df.iloc[: , 5:-1])
print(result)
```

```
    feature       VIF
0      CGPA  2.455008
1  Research  2.455008
```

```
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)

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
--------------------------------------------------
```

- After removing collinear features using VIF and using only two features. R2_score and Adj_r2 are still the same as before the testing dataset.

Mean of residuals

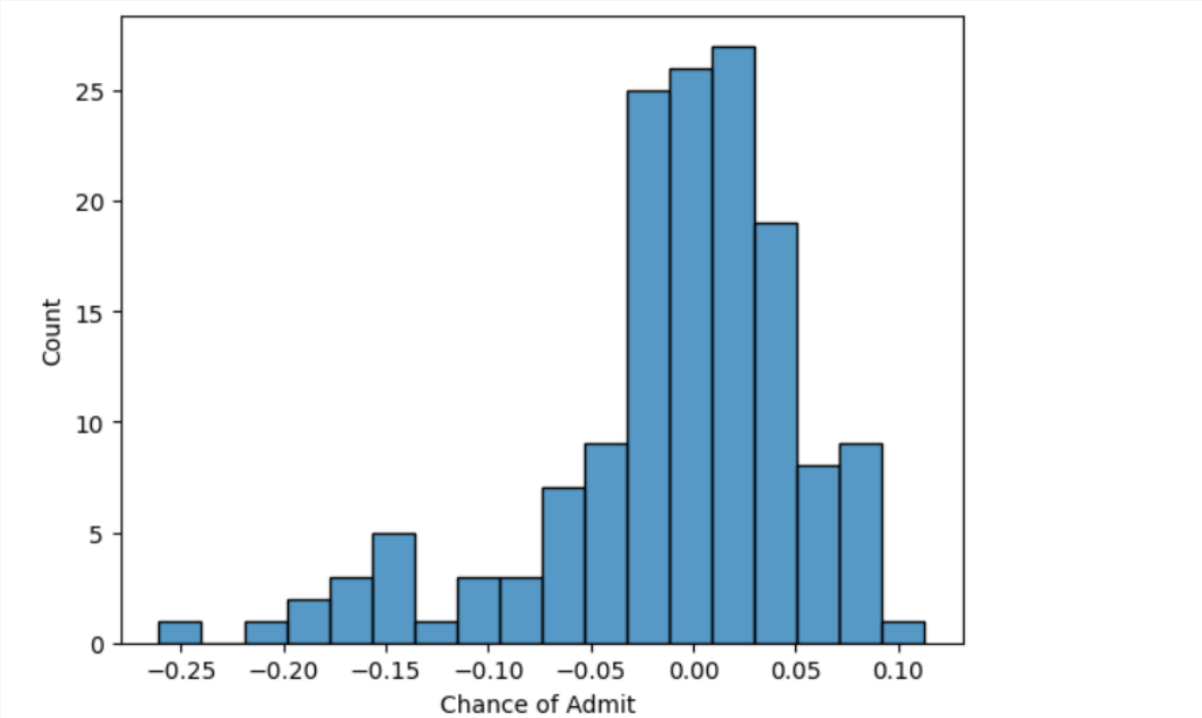
It is clear from RMSE that mean of residuals is almost zero.

Linearity of Variables

It is clear from the EDA that independent variables are linearly dependent on target variables.

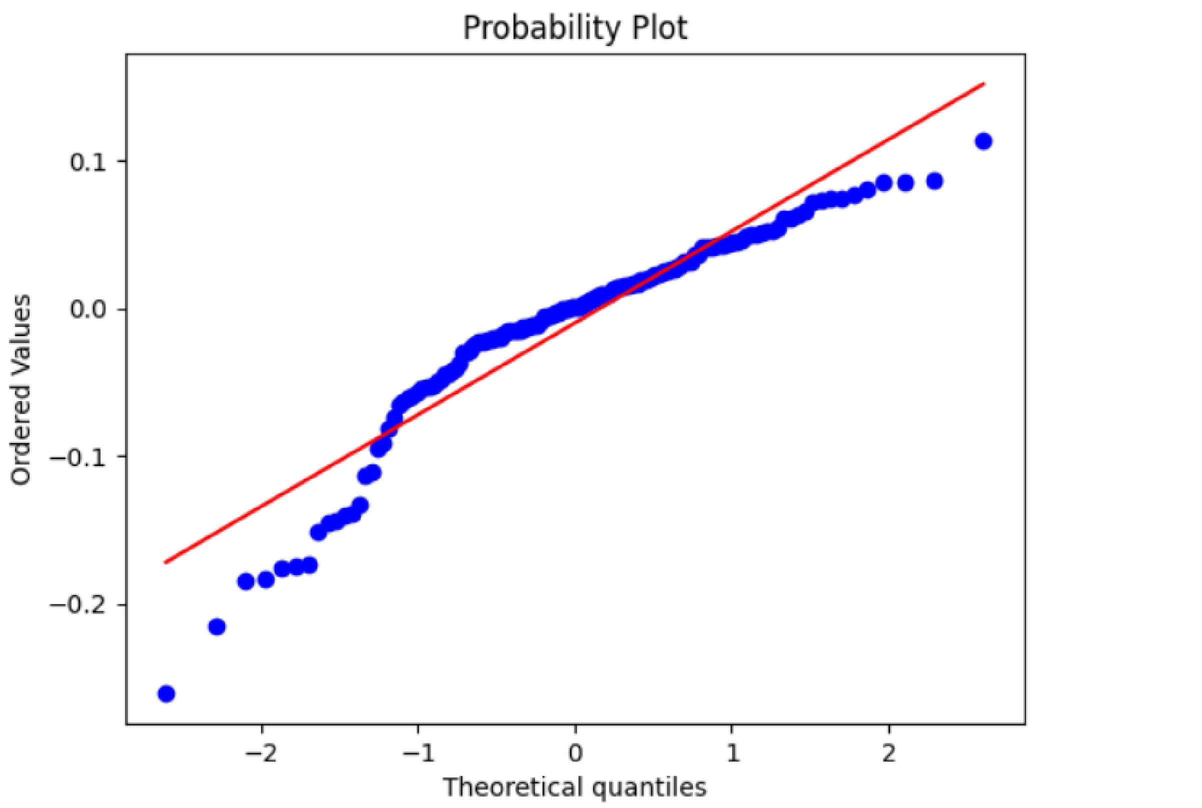
Normality of Residuals

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



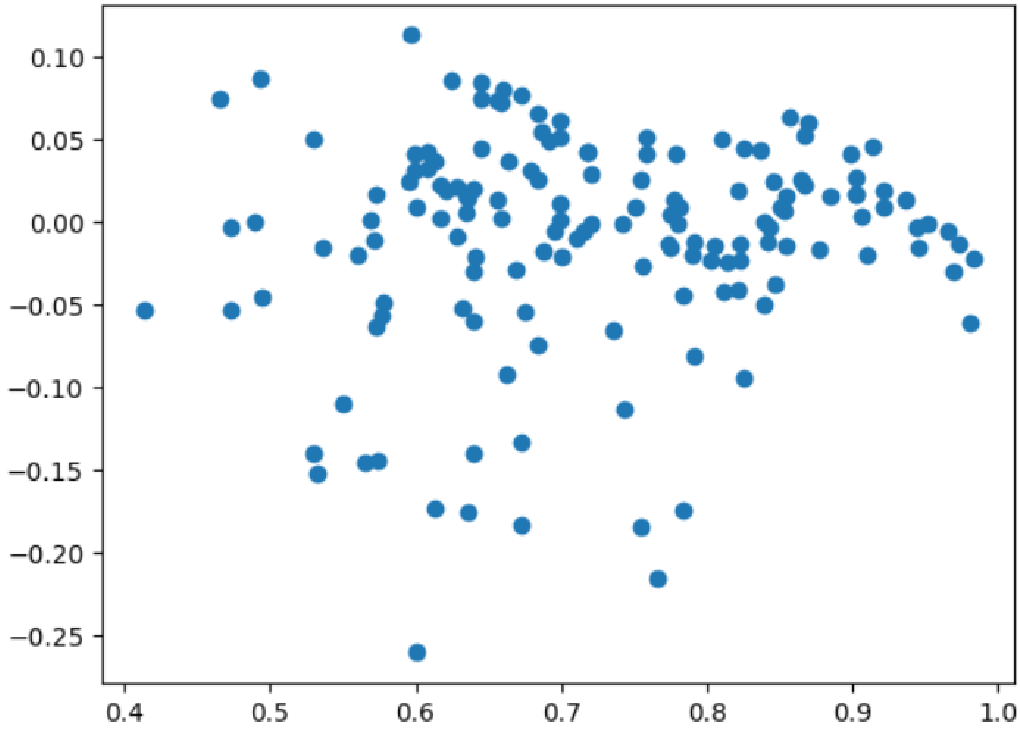
- From above graph we can say that residuals are almost normally distributed.

```
stats.probplot(residuals, plot=plt)
plt.show()
```



Test of Homoscedasticity

```
plt.scatter(y_pred, residuals)
plt.show()
```



- We can see from the above graph that there is a less variance between actual target and residuals as we wanted our prediction to be.

Insights, Feature Importance and Interpretations and Recommendations:

- fist column was observed as unique row identifier which was dropped and was not required for model building.
- University Rating, SOP and LOR strength and research are seeming to be discrete random Variables, but also ordinal numeric data.
- all the other features are numeric, ordinal and continuous.
- No null values were present in data.
- No Significant number of outliers were found in data.

- Chance of admission(target variable) and GRE score(an independent feature) are nearly normally distributed.
- Independent Variables (Input data): GRE Score, TOEFL Score, University Rating, SOP, LOR, CGPA, Research
- Target/Dependent Variable: Chance of Admit (the value we want to predict) from correlation heatmap, we can observe GRE score, TOEFL score and CGPA have very high correlation with Change of admission.
- University rating, SOP, LOR and Research have comparatively slightly less correlated than other features.
- chances of admit is a probability measure, which is within 0 to 1 which is good (no outliers or misleading data in column).
- Range of GRE score looks like between 290 to 340.
- range of TOEFL score is between 92 to 120.
- university rating, SOP and LOR are distributed between range of 1 to 5.
- CGPA range is between 6.8 to 9.92.
- From boxplots (distribution of chance of admission(probability of getting admission) as per GRE score) : with higher GRE score , there is high probability of getting an admission. Students having high TOEFL score, has higher probability of getting admission.
- from count plots, we can observe, statement of purpose SOP strength is positively correlated with Chance of Admission.
- we can also similarly pattern in Letter of Recommendation Strength and University rating, have positive correlation with Chances of admission.

Recommendations

1. Following are the final model results on the test data:
    - **RMSE:** 0.07
    - **MAE:** 0.05
    - **R2_score:** 0.81
    - **Adjusted_R2:** 0.81
2. education institute cannot just help student to improve their CGPA score but also assist them writing good LOR and SOP thus helping them admit to better university.
3. The education institute cannot just help student to improve their GRE Score but can also assist them writing good LOR and SOP thus helping them admit to a better University.
4. Awareness of CGPA and Research Capabilities: Seminars can be organised to increase the awareness regarding CGPA and Research Capabilities to enhance the chance of admit.
5. Any student can never change their current state of attributes so awareness and marketing campaign need to surveyed hence creating a first impression on student at undergraduate level, which won’t just increase company's popularity but will also help student get prepared for future plans in advance.

6. A dashboard can be created for students whenever they logged in into your website, hence allowing a healthy competition also to create a progress report for students.
7. Additional features like number of hours they put in studying, watching lectures, assignments solved percentage, marks in mock test can result a better report for every student to judge themselves and improve on their own.