Jamboree Education

Introduction

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.

Problem Statement

Data 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.

In [1]:
```
# Importing required libraries

import numpy as np
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt
%matplotlib inline
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm


from scipy.stats import norm
```

In [2]:
```
# Importing the dataset and reading the dataset using Pandas
df=pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000
```

In [3]:
```
df
```

Out[3]:

	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
...	...	...	...	...	...	...	...	...	...
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 9 columns

Data Exploration to draw insights from the given data set

In [4]:
```
# shape of the data set
df.shape
```

Out[4]:
```
(500, 9)
```

In [5]:
```
# General information about the dataset about the data type and any null value
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
```

In [6]:
```
df.head()
```

Out[6]:

	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

Descriptive Statistics

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

Out[7]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Res
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.0
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	0.0
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.0
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.0
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.0
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.0

Observations:

- There are 500 student records
- There are 8 features
- Chances of admit is a probability measure, which is within 0 to 1 which is good (as no outliers are present)
- GRE score range is 290 to 340.
- Range of TOEFL score is in between 92 to 120.
- University rating, SOP and LOR are distributed in between range of 1 to 5.
- Cumulative GPA is in the range of 6.8 to 9.92.
- Mean research score is 0.56 while median research score is 1.0, but it is a binary variable so it doesn't give much meaning.

Data pre-processing and cleaning

In [8]:
```
#checking for null values
df.isna().sum()
```

Out[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
```

Observations:

- There are no null values

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
column_names = df.columns
column_names
```

Out[10]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

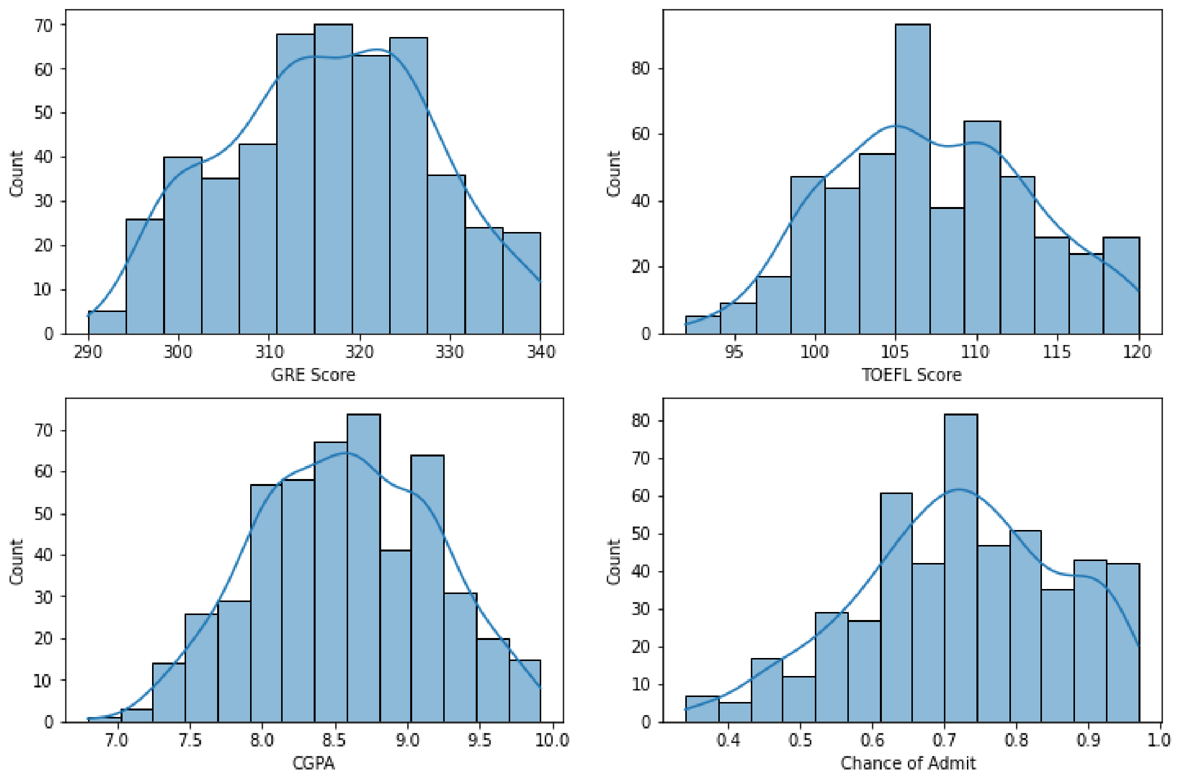
In [11]:
```
#Remove serial number as it is not a feature required for prediction

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

Exploratory Data Analysis

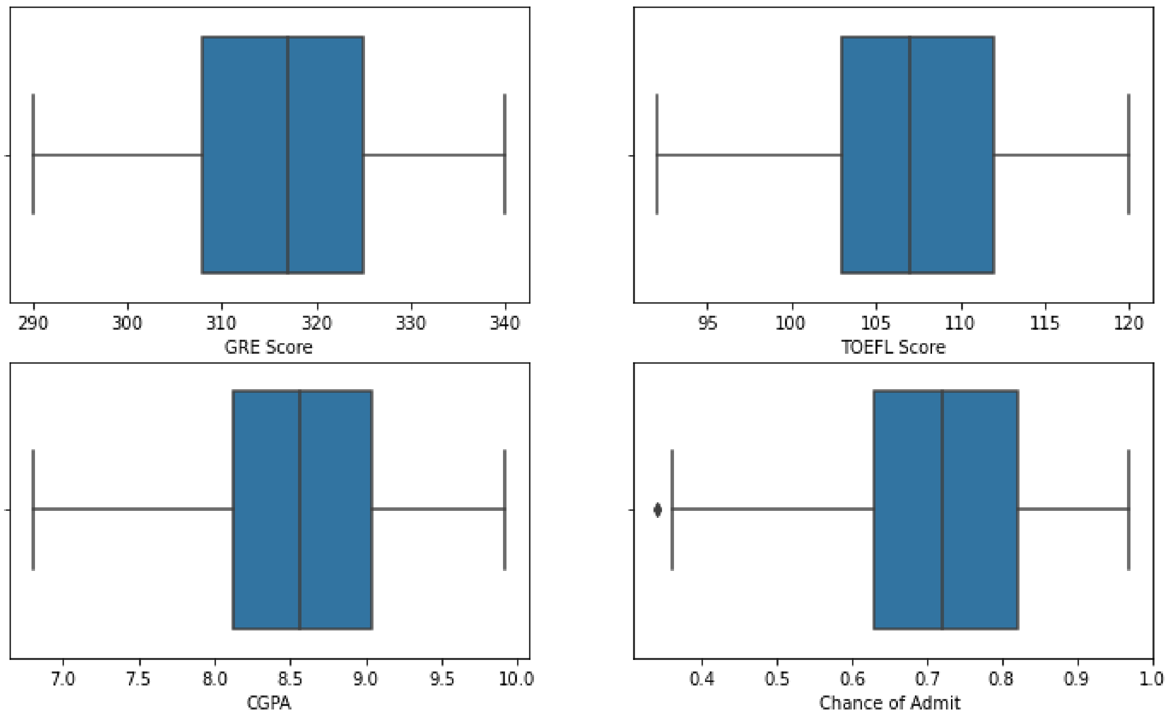
In [12]:
```
cat_cols = ['University Rating', 'SOP', 'LOR ', 'Research']
num_cols = ['GRE Score', 'TOEFL Score', 'CGPA']
target = 'Chance of Admit '
```

Univariate Analysis

In [13]:
```
# check distribution of each numerical variable
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()
```



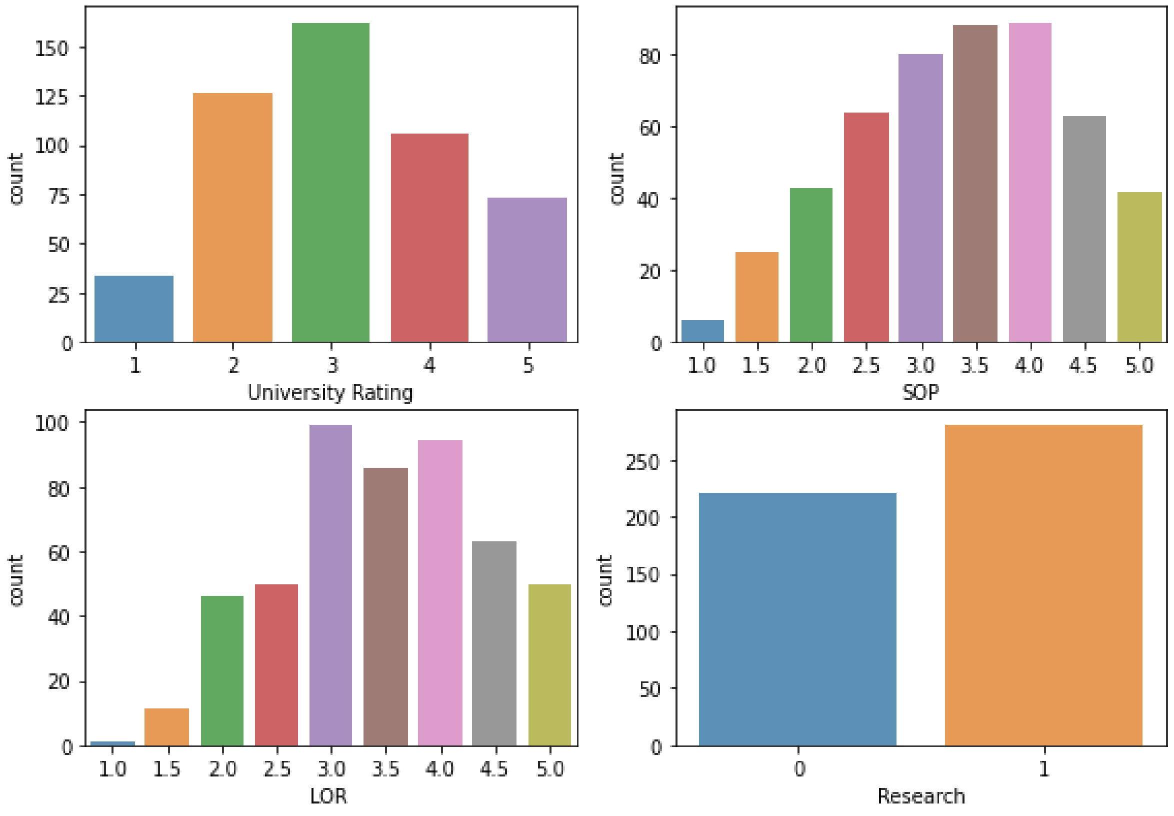
In [14]:
```
# check for outliers using boxplots
rows, cols = 2, 2
fig, axs = plt.subplots(rows, cols, figsize=(12, 7))

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()
```



Observations

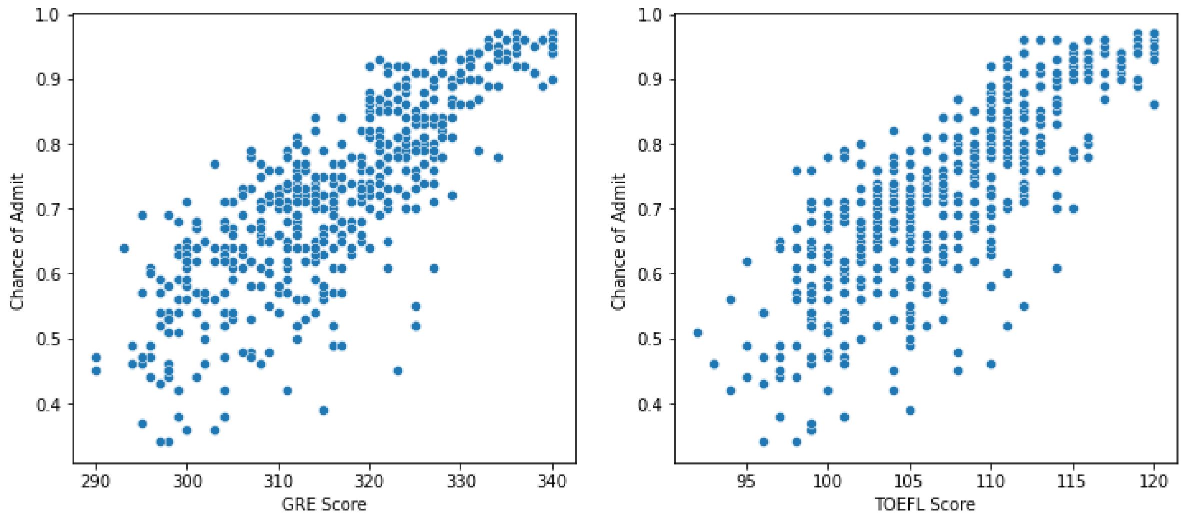
- There are no outliers in the dataset.

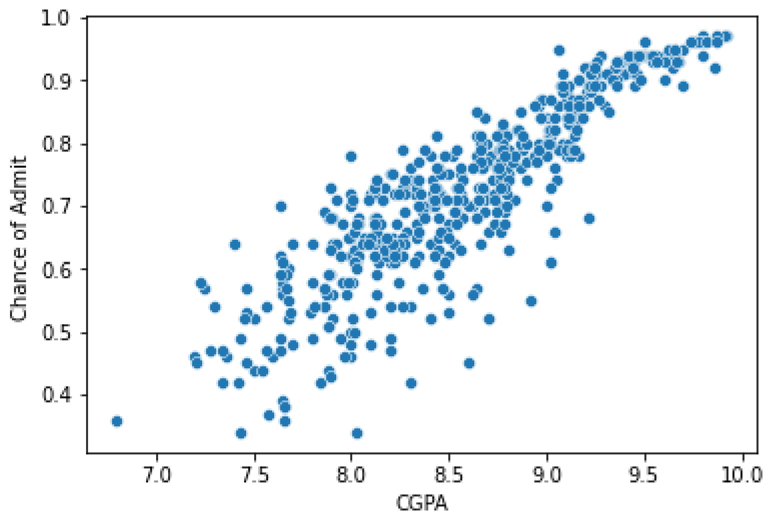
In [15]:
```
# countplots for categorical variables
cols, rows = 2, 2
fig, axs = plt.subplots(rows, cols, figsize=(10, 7))

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()
```



- Observations:
- Most frequent values are University Rating: 3 SOP: 3.5 & 4 LOR: 3 Research: True

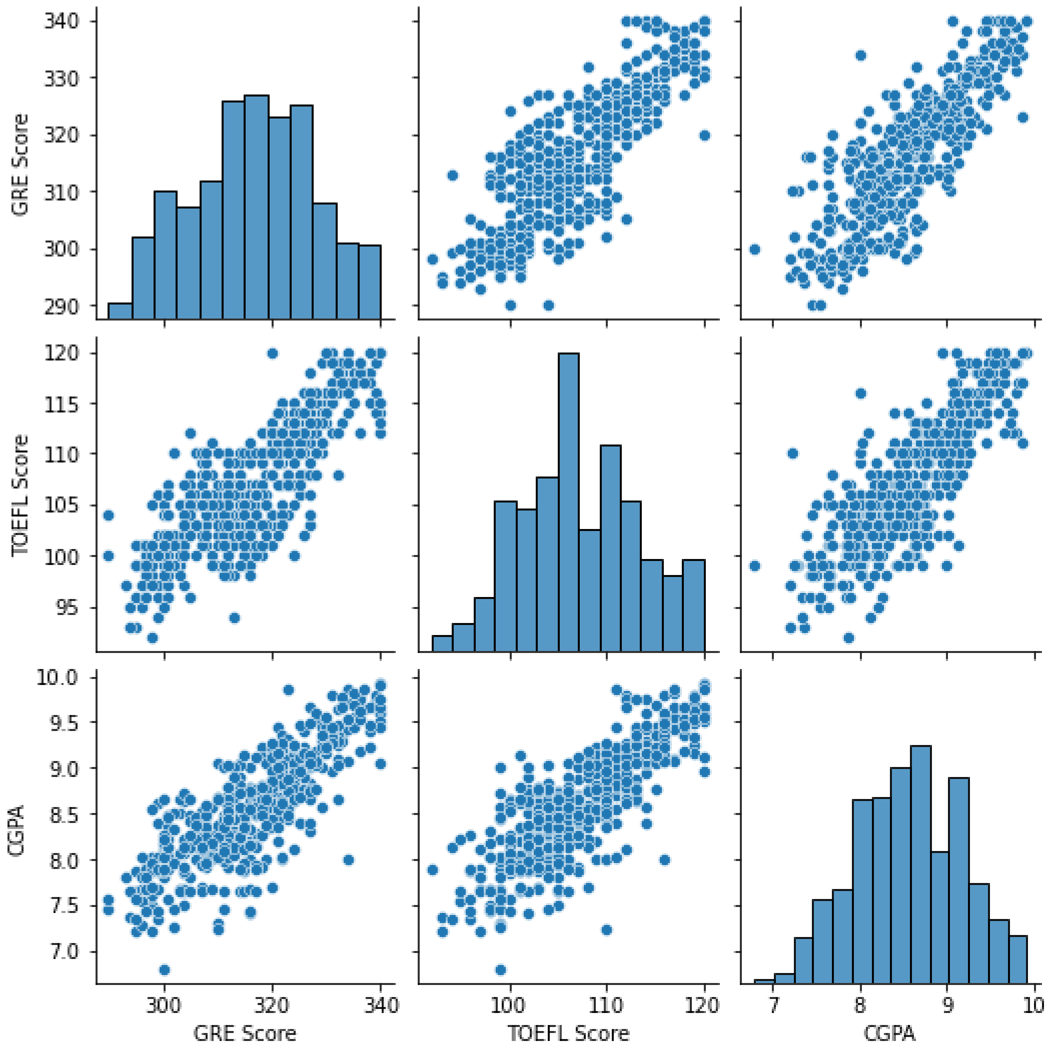
In [16]:
```
# check relation bw continuous variables & target variable
fig, axs = plt.subplots(1, 2, figsize=(12,5))

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()
```





Multivariate Analysis

```
In [17]: sns.pairplot(df[num_cols])
         plt.show()
```

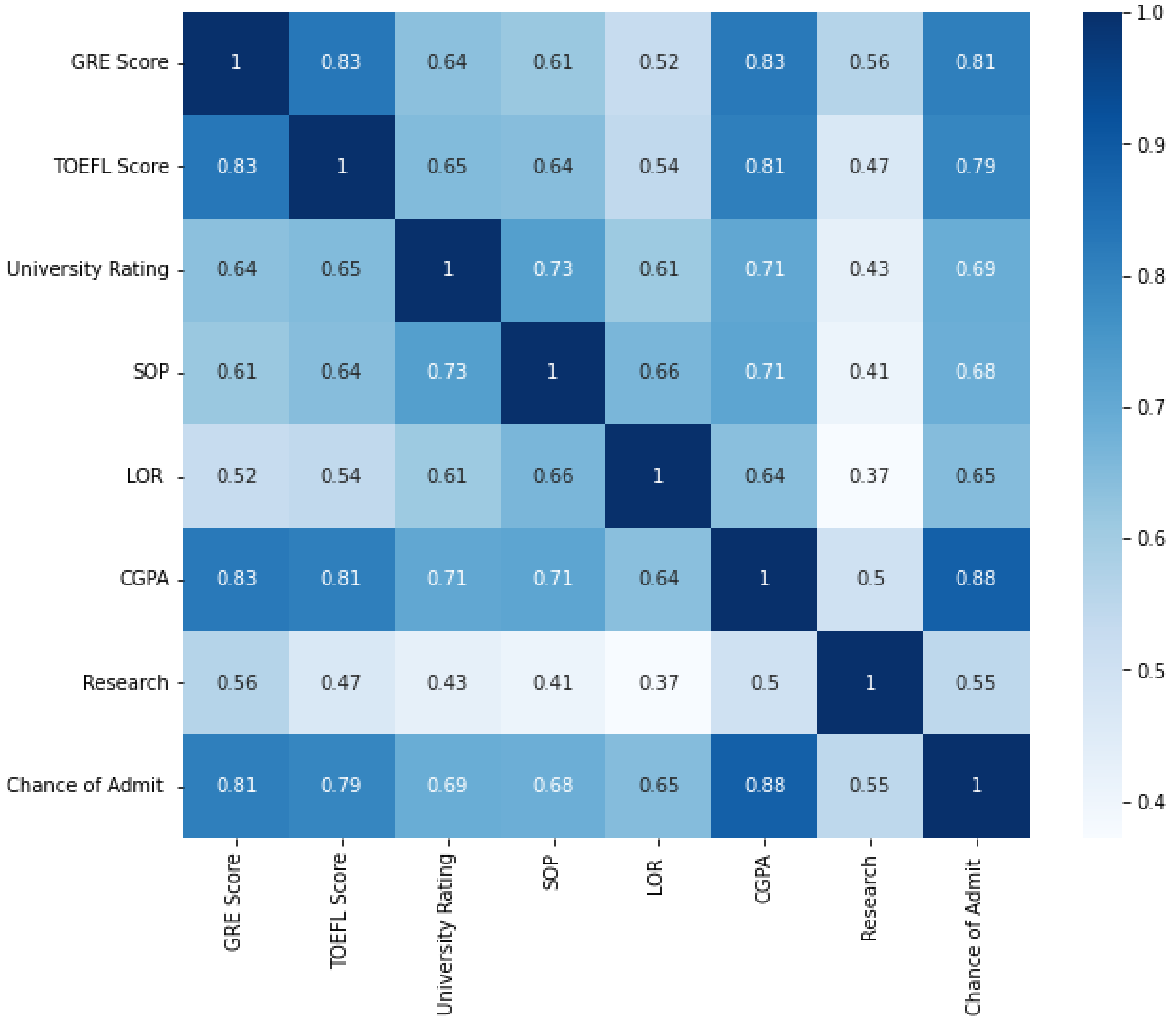


Observations:

- While university ranking, rating of SOP and LOR also have an impact on chances of admit, research is the only variable which doesn't have much of an impact.
- We can see from the scatterplot that the values of university ranking, SOP, LOR and research are not continuous so we can convert them into categorical variables.

Correlation Analysis

In [18]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), annot=True,cmap='Blues')
plt.show()
```



	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1	0.83	0.64	0.61	0.52	0.83	0.56	0.81
TOEFL Score	0.83	1	0.65	0.64	0.54	0.81	0.47	0.79
University Rating	0.64	0.65	1	0.73	0.61	0.71	0.43	0.69
SOP	0.61	0.64	0.73	1	0.66	0.71	0.41	0.68
LOR	0.52	0.54	0.61	0.66	1	0.64	0.37	0.65
CGPA	0.83	0.81	0.71	0.71	0.64	1	0.5	0.88
Research	0.56	0.47	0.43	0.41	0.37	0.5	1	0.55
Chance of Admit	0.81	0.79	0.69	0.68	0.65	0.88	0.55	1

Observations

- Chance of Admit shows strong correlation with GRE Score, TOEFL Score and CGPA.
- Research shows 0.5 correlation with GRE Score, CGPA as well as chance of admit implying students with research component have 50% chance of getting admission.
- Statement of purpose, SOP is correlated with LOR? Letter of Recommendations.
- GRE Score and TOEFL Score share a strong correlation meaning that students who are performing well in GRE are also doing doing well in TOEFL.
- CGPA shares a strong correlation with GRE as well as TOEFL Score implying that the student is good in his college subjects as well.

Inference

- Correlation matrix shows that exam scores (GRE, TOEFL) have a strong positive correlation with chance of admission and they are also strongly correlated with CGPA of the student.

Data preparation

In [19]:
```
# Feature selection and extracting into Column X and target y
X = df.drop(columns=[target])
y = df[target]
```

In [22]:
```
# standardize the dataset
sc = StandardScaler()
X = sc.fit_transform(X)
```

In [23]:
```
# Splitting into Training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando
```

In [24]:
```
scaler_X = MinMaxScaler()
```

In [25]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

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

In [26]:
```
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)

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

In [28]:
```
def train_model(X_train, y_train, X_test, y_test,cols, model_name="linear", al
    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[
    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['
    print(f"Train Adjusted_R2: {train_res['adj_r2']} Test Adjusted_R2: {test_r
    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
```

In [29]:
```
train_model(X_train, y_train, X_test, y_test,df.columns[:-1], "linear")
```

```
----   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: 0.7249781214769964
              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
--------------------------------------------------
```

Out[29]:
```
LinearRegression()
```

- Observations:
- There is no difference in the training and test scores of the model so we can conclude that there is no overfitting of the model.

- Mean Absolute Error of 0.04 shows that on an average, the absolute difference between the actual and predicted values of chance of admit is 4%
- Root Mean Square Error of 0.06 means that on an average, the root of squared difference between the actual and predicted values is 6%
- R2 score of 0.82 means that our model captures 82% variance in the data.
- Adjusted R2 is an extension of R2 which shows how the number of features used changes the accuracy of prediction

In [ ]:

In [30]:
```
train_model(X_train, y_train, X_test, y_test,df.columns[:-1], "ridge")
```

```
----   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: 0.7249823645841699
              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
--------------------------------------------------
```

Out[30]:
```
Ridge()
```

In [31]:
```
train_model(X_train, y_train, X_test, y_test,df.columns[:-1], "lasso", 0.001)
```

```
----   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: 0.7249659139557144
              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
--------------------------------------------------
```

Out[31]:
```
Lasso(alpha=0.001)
```

Linear Regression model Assumption test

Checking for Multi-collinearity

In [32]:
```
def vif(newdf):
    # VIF dataframe
    vif_data = pd.DataFrame()
    vif_data["feature"] = newdf.columns

    # calculating VIF for each feature
    vif_data["VIF"] = [variance_inflation_factor(newdf.values, i)
                              for i in range(len(newdf.columns))]
    return vif_data
```

In [33]:
```
res = vif(df.iloc[:,:-1])
res
```

Out[33]:

	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

In [34]:
```
# drop GRE Score and again calculate the VIF
res = vif(df.iloc[:, 1:-1])
res
```

Out[34]:

	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

- Observation:
- Dropping GRE scores changed the VIF scores

In [35]:
```
#  drop TOEFL Score and again calculate the VIF
res = vif(df.iloc[:,2:-1])
res
```

Out[35]:

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

In [36]:
```
# Now lets drop the SOP and again calculate VIF
res = vif(df.iloc[:,2:-1].drop(columns=['SOP']))
res
```

Out[36]:

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

In [37]:
```
# lets drop the LOR as well
newdf = df.iloc[:,2:-1].drop(columns=['SOP'])
newdf = newdf.drop(columns=['LOR '], axis=1)
res = vif(newdf)
res
```

Out[37]:

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

In [38]:
```
# drop the University Rating
newdf = newdf.drop(columns=['University Rating'])
res = vif(newdf)
res
```

Out[38]:

	feature	VIF
0	CGPA	2.455008
1	Research	2.455008

Observations:

- Almost all the variables excluding research have a very high level of collinearity. This was also observed from the correlation heat map which showed strong positive correlation

In [39]:
```
# now again train the model with these only two features
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, rando
```

- Observations:
- 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.

Mean of residuals

- The mean of residuals is useful to assess the overall bias in the regression model. If the mean of residuals is close to zero, it indicates that the model is unbiased on average. However, if the mean of residuals is significantly different from zero, it suggests that the model is systematically overestimating or underestimating the observed values.

- Observations:
- Since the mean of residuals is very close to 0, we can say that the model is unbiased.

Linearity of variables

- Linearity of variables refers to the assumption that there is a linear relationship between the independent variables and the dependent variable in a regression model. It means that the effect of the independent variables on the dependent variable is constant across different levels of the independent variables.

In [40]:
```
sns.scatterplot(x = y_pred_test, y=residuals)
plt.title('Residual Plot')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.axhline(y=0, color='r', linestyle='--')
plt.show();
```

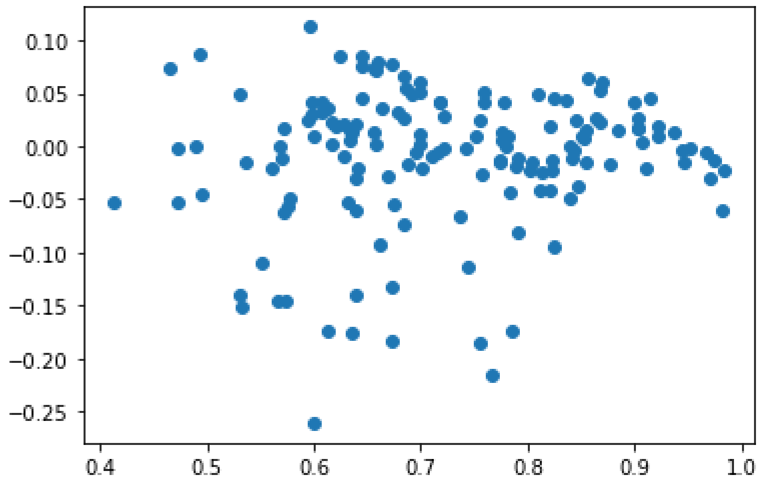
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_15676/3399749213.py in <module>
----> 1 sns.scatterplot(x = y_pred_test, y=residuals)
      2 plt.title('Residual Plot')
      3 plt.xlabel('Predicted Values')
      4 plt.ylabel('Residuals')
      5 plt.axhline(y=0, color='r', linestyle='--')

NameError: name 'y_pred_test' is not defined
```

- Observations: Since the residual plot shows no clear pattern or trend in residuals, we can conclude that linearity of variables exists

Test for Homoscedasticity

- Homoscedasticity refers to the assumption in regression analysis that the variance of the residuals (or errors) should be constant across all levels of the independent variables. In simpler terms, it means that the spread of the residuals should be similar across different values of the predictors.
- When homoscedasticity is violated, it indicates that the variability of the errors is not consistent across the range of the predictors, which can lead to unreliable and biased regression estimates.

In [65]:
```
# Scatter  plot: Plot the residuals against the predicted values
plt.scatter(y_pred, residuals)
plt.show()
```



- Since we do not see any significant change in the spread of residuals, we can conclude that homoscedasticity is met.

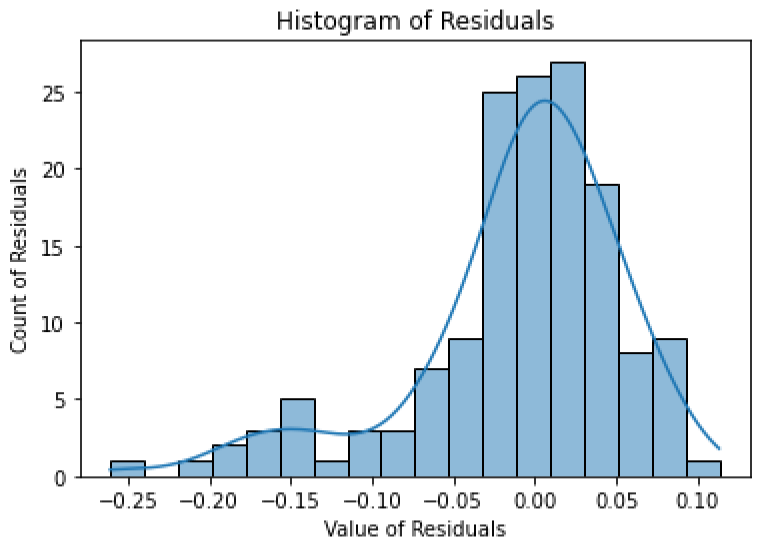
Normality of Residuals:

The assumption of normality is important in many statistical analyses because it allows for the application of certain statistical tests and the validity of confidence intervals and hypothesis tests. When residuals are normally distributed, it implies that the errors are random, unbiased, and have consistent variability.

To check for the normality of residuals, you can follow these steps:

Residual Histogram: Create a histogram of the residuals and visually inspect whether the shape of the histogram resembles a bell-shaped curve. If the majority of the residuals are clustered around the mean with a symmetric distribution, it suggests normality.

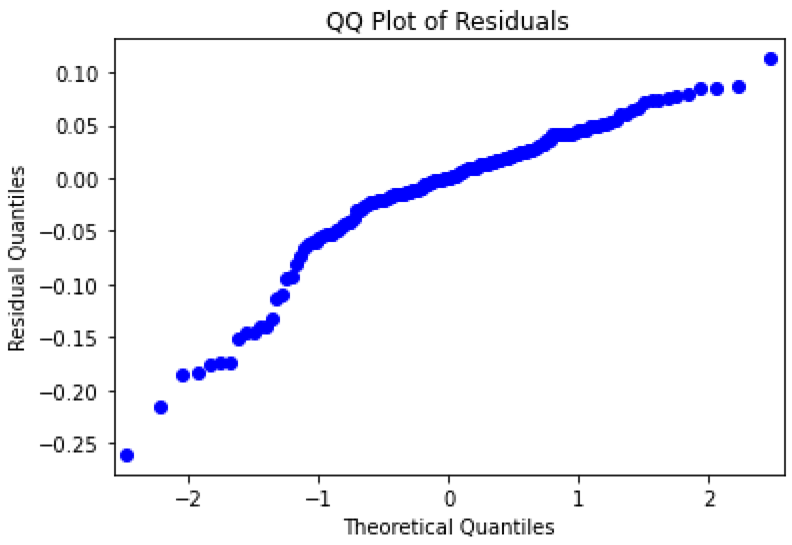
```
In [78]: #Histogram of Residuals
         sns.histplot((residuals), kde=True)
         plt.title('Histogram of Residuals')
         plt.xlabel('Value of Residuals')
         plt.ylabel('Count of Residuals')
         plt.show();
```



- The histogram shows that there is a negative skew in the distribution of residuals but it is close to a normal distribution

In [80]:
```
# QQ-Plot of residuals
sm.qqplot(residuals)
plt.title('QQ Plot of Residuals')
plt.ylabel('Residual Quantiles')
plt.show();
```

```
C:\Users\Dell\anaconda3\lib\site-packages\statsmodels\graphics\gofplots.py:99
3: UserWarning: marker is redundantly defined by the 'marker' keyword argumen
t and the fmt string "bo" (-> marker='o'). The keyword argument will take pre
cedence.
  ax.plot(x, y, fmt, **plot_style)
```



Lasso and Ridge Regression

Ridge and Lasso regression are both regularization techniques used to prevent overfitting in linear regression models. They work by adding a penalty term to the cost function, which helps to control the complexity of the model by shrinking the coefficient values.

Ridge Regression: Ridge regression uses L2 regularization, where the penalty term is the squared sum of the coefficients multiplied by a regularization parameter (lambda or alpha). The regularization term helps to reduce the impact of less important features on the model and prevents them from dominating the model. Ridge regression can help in reducing the variance of the model and is particularly useful when dealing with multicollinearity (high correlation between independent variables).

Lasso Regression: Lasso regression uses L1 regularization, where the penalty term is the sum of the absolute values of the coefficients multiplied by a regularization parameter (lambda or alpha). Lasso regression has the ability to shrink some coefficients to exactly zero, effectively performing feature selection. This makes Lasso regression useful when dealing with high-dimensional data where only a few variables are relevant.

The main differences between Ridge and Lasso regression are:

Ridge regression tends to shrink all coefficient values towards zero, but it rarely makes them exactly zero. On the other hand, Lasso regression can make coefficient values exactly zero, performing variable selection. Ridge regression is suitable when dealing with multicollinearity

- Observation:
- While Linear Regression and Ridge regression have similar scores, Lasso regression has not performed well on both training and test data

Observations:

- There are 500 student records
- There are 8 features
- Chances of admit is a probability measure, which is within 0 to 1 which is good (as no outliers are present)
- GRE score range is 290 to 340.
- Range of TOEFL score is in between 92 to 120.
- University rating, SOP and LOR are distributed in between range of 1 to 5.
- Cumulative GPA is in the range of 6.8 to 9.92.
- Mean research score is 0.56 while median research score is 1.0, but it is a binary variable so it doesn't give much meaning.
- No outliers found in the data.
- Almost all the variables excluding research have a very high level of collinearity. This was also observed from the correlation heat map which showed strong positive correlation between GRE, TOEFL scores and CGPA.

** LinearRegression()

- There is no difference in the training and test scores of the model so we can conclude that there is no overfitting of the model.
- Mean Absolute Error of 0.04 shows that on an average, the absolute difference between the actual and predicted values of chance of admit is 4%
- Root Mean Square Error of 0.06 means that on an average, the root of squared difference between the actual and predicted values is 6%
- R2 score of 0.82 means that our model captures 82% variance in the data.
- Adjusted R2 is an extension of R2 which shows how the number of features used changes the accuracy of prediction.
- While Linear Regression and Ridge regression have similar scores, Lasso regression has not performed well on both training and test data.

Inference:

- While university ranking, rating of SOP and LOR also have an impact on chances of admit, research is the only variable which doesn't have much of an impact.
- We can see from the scatterplot that the values of university ranking, SOP, LOR and research are not continuous so we can convert them into categorical variables.
- Chance of Admit shows strong correlation with GRE Score, TOEFL Score and CGPA.
- Research shows 0.5 correlation with GRE Score, CGPA as well as chance of admit implying students with research component have 50% chance of getting admission.

- Statement of purpose, SOP is correlated with LOR, Letter of Recommendations.
- GRE Score and TOEFL Score share a strong correlation meaning that students who are performing well in GRE are also doing doing well in TOEFL.
- CGPA shares a strong correlation with GRE as well as TOEFL Score implying that the student is good in his college subjects as well.
- Correlation matrix shows that exam scores (GRE, TOEFL) have a strong positive correlation with chance of admission and they are also strongly correlated with CGPA of the student.
- Almost all the variables excluding research have a very high level of collinearity. This was also observed from the correlation heat map which showed strong positive correlation between GRE, TOEFL scores and CGPA.
-
- Mean of Residuals It is clear from RMSE that Mean of Residuals is almost zero.
- Linearity of variables It is quite clear from EDA that independent variables are linearly dependent on the target variables.
- While Linear Regression and Ridge regression have similar scores, Lasso regression has not performed well on both training and test data
- The distribution of target variable (chances of admit) is left-skewed
- Exam scores (CGPA/GRE/TOEFL) have a strong positive correlation with chance of admit. These variables are also highly correlated amongst themselves the categorical variables such as university ranking, research, quality of SOP and LOR also show an upward trend for chances of admit.
- From the model coefficients (weights), we can conclude that CGPA is the most significant predictor variable while SOP/University Rating are the least significant
- Both Linear Regression and Ridge Regression models, which are our best models, have captured upto 82% of the variance in the target variable (chance of admit). Due to high colinearity among the predictor variables, it is difficult to achieve better results.
- Other than multicolinearity, the predictor variables have met the conditions required for Linear Regression - mean of residuals is close to 0, linearity of variables, normality of residuals and homoscedasticity is established.

Recommendations:

- Since all the exam scores are highly correlated, it is recommended to add more independent features for better prediction.
- Examples of other independent variables could be work experience, internships, mock interview performance, extracurricular activities or diversity variables

In [ ]: