Jamboree Education - Linear Regression

Context

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.

Link to google Drive

Jamboree Education.ipynb - Colab (google.com)

1) **Define Problem Statement and perform Exploratory Data Analysis** (10 points)
   Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
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 statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
```

Read csv file and top 10 rows and bottom 10 rows

```
[57] df = pd.read_csv("Jamboree_Admission.csv")
```

```
df.head()
```

	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

Next steps: Generate code with df   View recommended plots

```
df.tail()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
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

Information on the data

```
[60] df.shape
```

```
(500, 9)
```

```
[61] df.ndim
```

```
2
```

```
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
```

Check for null values in the data

```
df.isna().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
```

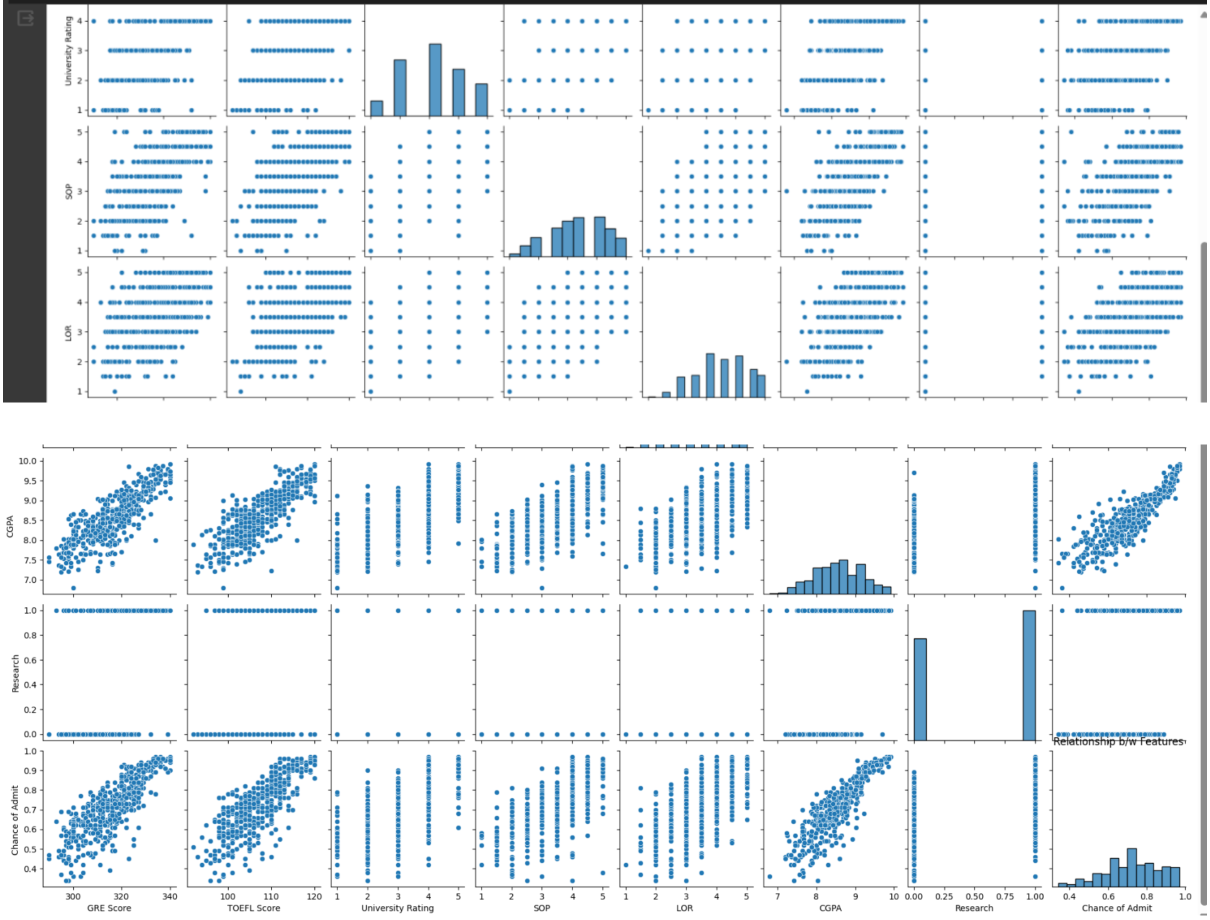
Since Serial no. column is not required this column can be dropped.

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

Univariate/Bivariate analysis

Pair plot of all numerical columns

```
[65] sns.pairplot(df)
     plt.title('Relationship b/w Features')
     plt.show();
```



- Exam scores (GRE, TOEFL and CGPA) have a high positive correlation with chance of admit
- 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. We can convert these columns to categorical variables

Rename incorrect column names

```
[66] df.rename(columns={'LOR ':'LOR', 'Chance of Admit ':'Chance of Admit'}, inplace=True)
```

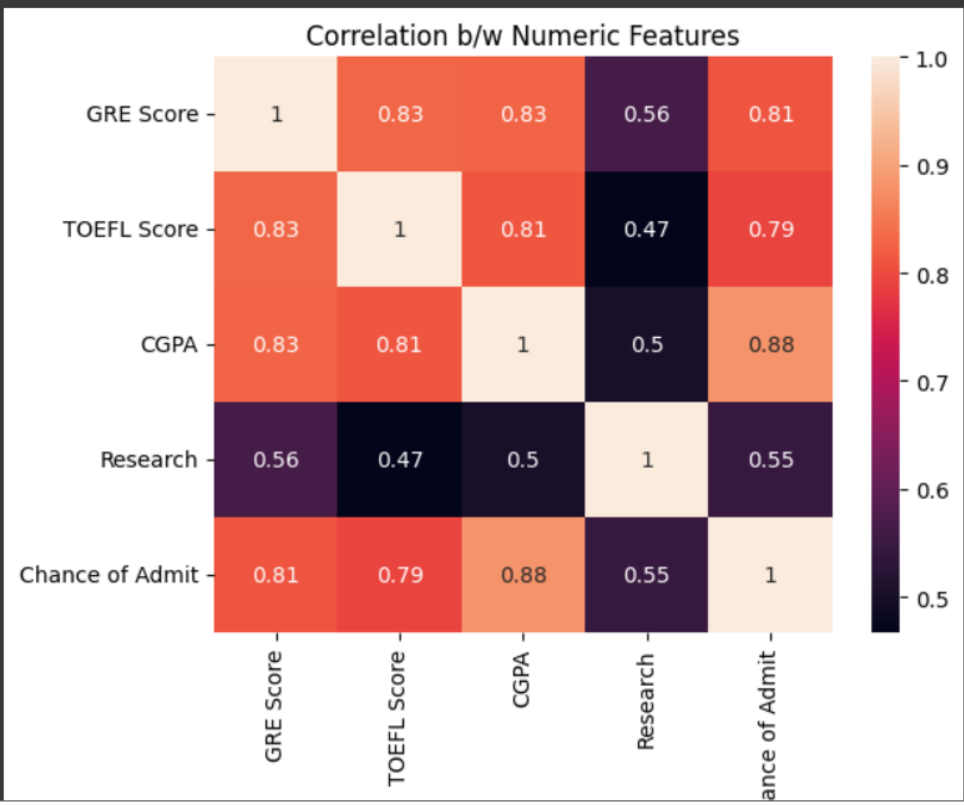
Group university rating, SOP, LOR as category columns and Research as Boolean

```
df[['University Rating', 'SOP', 'LOR']] = df[['University Rating', 'SOP', 'LOR']].astype('category')
df['Research'] = df['Research'].astype('bool')
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    category
 3   SOP                500 non-null    category
 4   LOR                500 non-null    category
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    bool
 7   Chance of Admit    500 non-null    float64
dtypes: bool(1), category(3), float64(2), int64(2)
memory usage: 18.6 KB
```

Corelation between all numerical columns using heat map

```
df_corr = df.corr(numeric_only=True)
sns.heatmap(df_corr, annot=True)
plt.title('Correlation b/w Numeric Features')
plt.show();
```

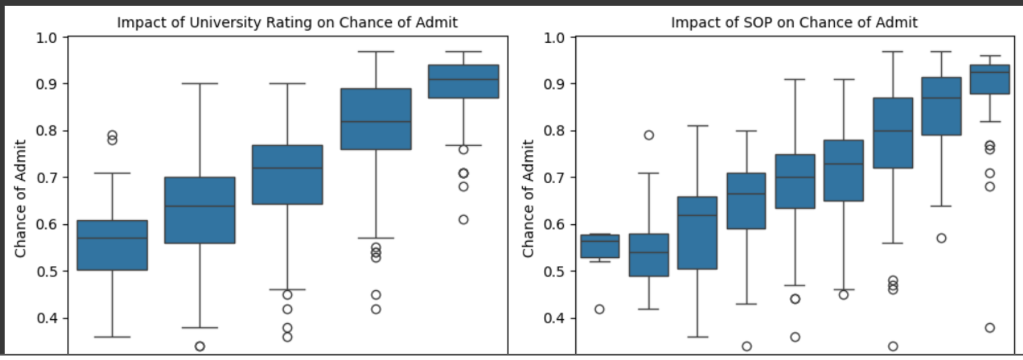


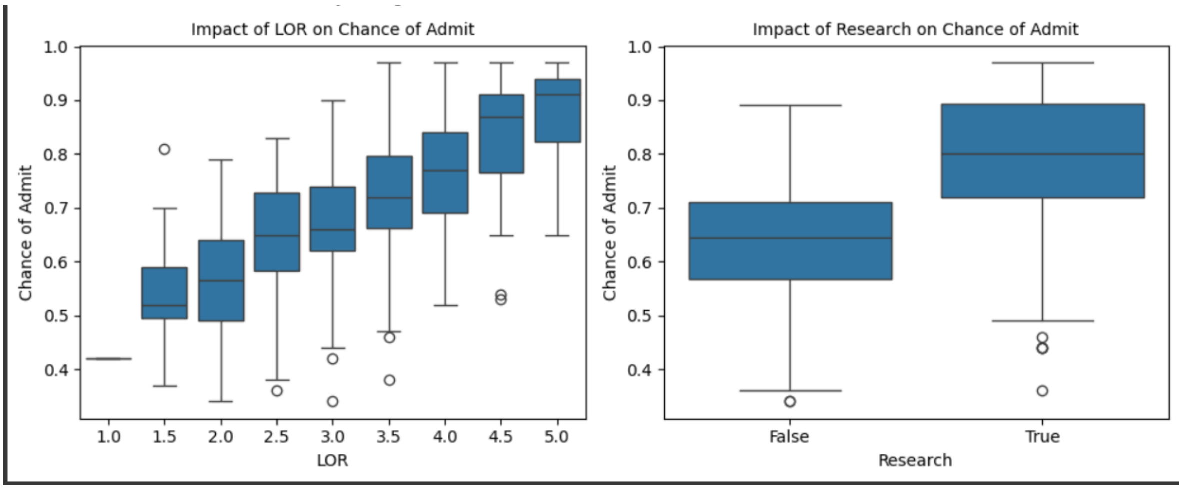
- Confirming the inferences from pairplot, the correlation matrix also shows that **exam scores (CGPA/GRE/TOEFL)** have a strong positive correlation with chance of admit
- Infact, they are also highly correlated amongst themselves

Box plot for all category columns

```
cat_cols = df.select_dtypes(include=['bool','category']).columns.tolist()
plt.figure(figsize=(10,8))
i=1
for col in cat_cols:
  ax = plt.subplot(2,2,i)
  sns.boxplot(data = df, x=col, y='Chance of Admit')
  plt.title(f"Impact of {col} on Chance of Admit", fontsize=10)
  plt.xlabel(col)
  plt.ylabel('Chance of Admit')
  i+=1

plt.tight_layout()
plt.show();
```



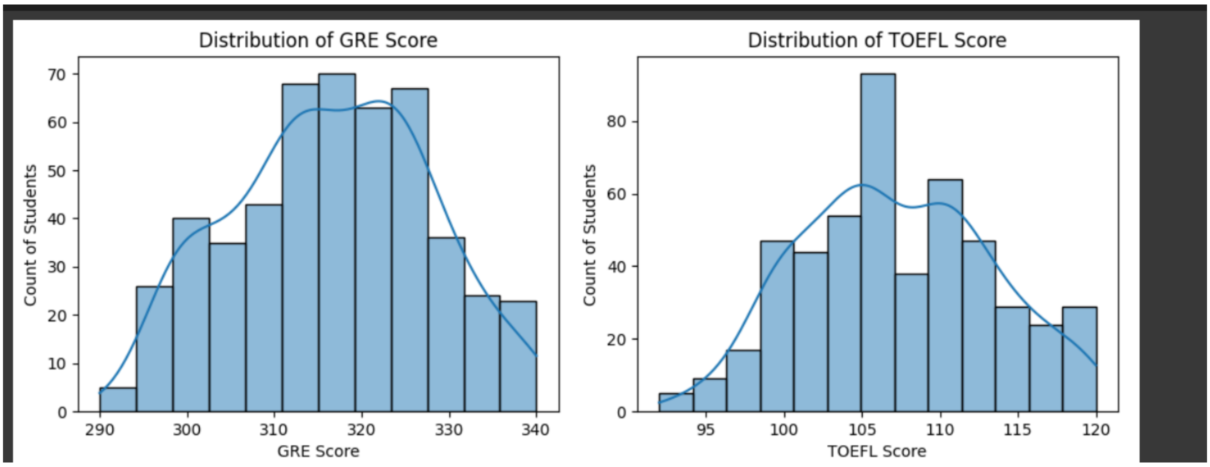


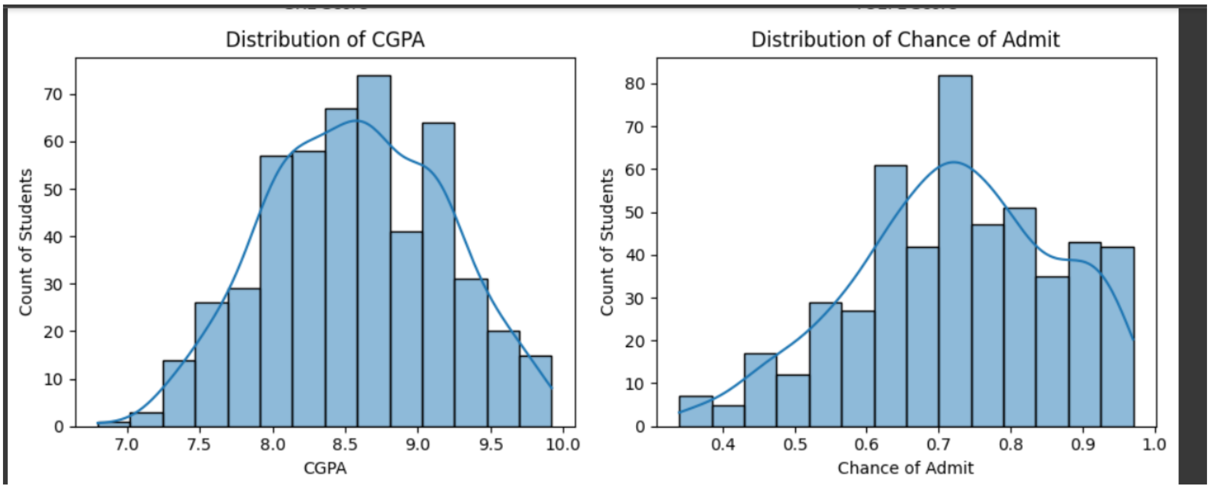
Distribution of numerical columns

```
numeric_cols = df.select_dtypes(include=['float','int']).columns.tolist()

plt.figure(figsize=(10,8))
i=1
for col in numeric_cols:
  ax=plt.subplot(2,2,i)
  sns.histplot(data=df[col], kde=True)
  plt.title(f'Distribution of {col}')
  plt.xlabel(col)
  plt.ylabel('Count of Students')
  i += 1

plt.tight_layout()
plt.show();
```





We can see the range of all the numerical attributes:

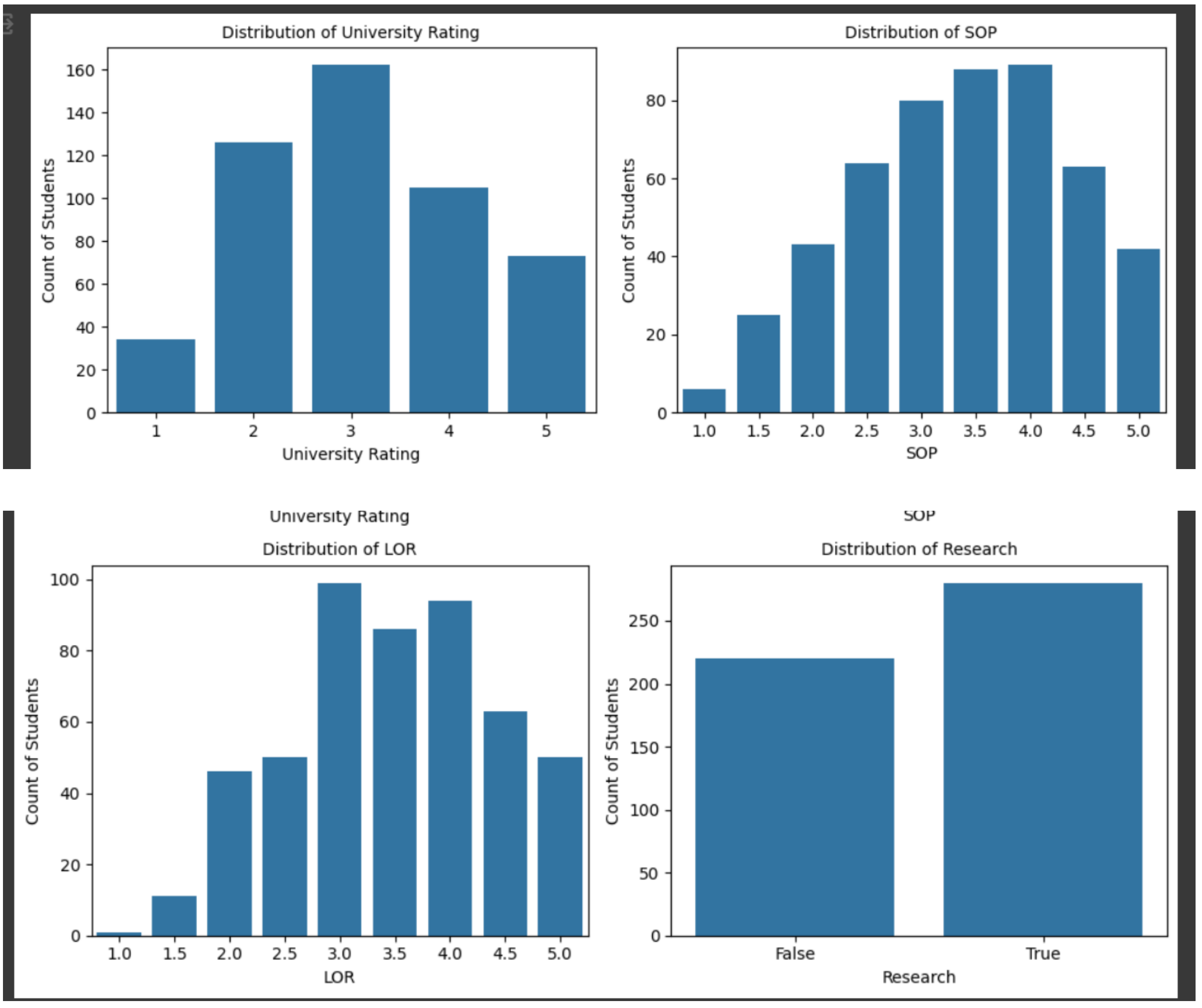
- GRE scores are between 290 and 340, with maximum students scoring in the range 310-330
- TOEFL scores are between 90 and 120, with maximum students scoring around 105
- CGPA ranges between 7 and 10, with maximum students scoring around 8.5
- Chance of Admit is a probability percentage between 0 and 1, with maximum students scoring around 70%-75%

Count plot for all categorical columns

```
[71] plt.figure(figsize=(10,8))
     i=1

     for col in cat_cols:
       ax = plt.subplot(2,2,i)
       sns.countplot(x=df[col])
       plt.title(f'Distribution of {col}', fontsize=10)
       plt.xlabel(col)
       plt.ylabel('Count of Students')
       i+=1

     plt.tight_layout()
     plt.show();
```



It can be observed that the most frequent value of categorical features is as following:

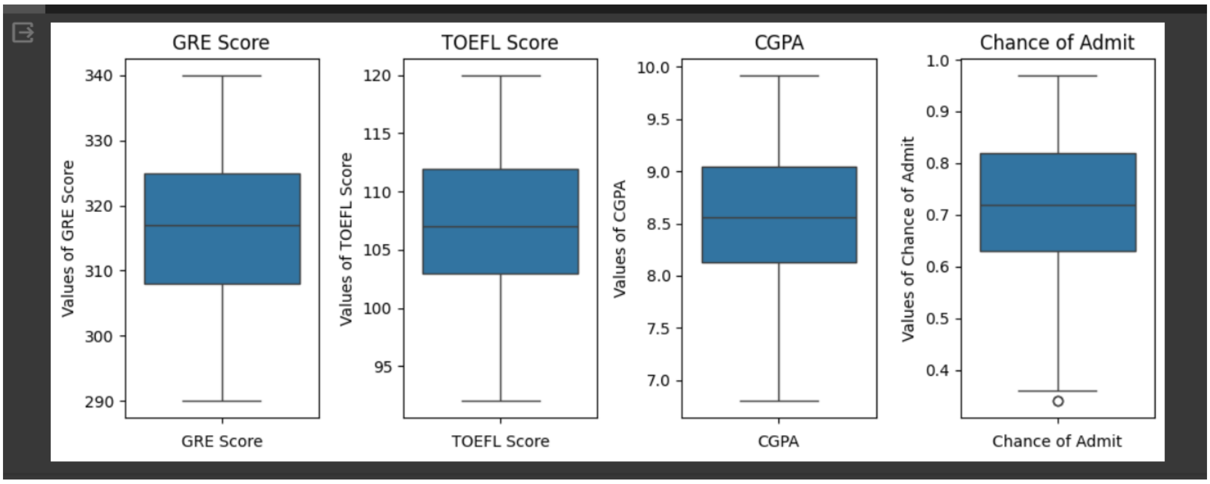
- University Rating: 3
- SOP: 3.5 & 4
- LOR: 3
- Research: True

Box plot for all numeric cols

```
plt.figure(figsize=(10,4))
i=1

for col in numeric_cols:
  ax = plt.subplot(1,4,i)
  sns.boxplot(df[col])
  plt.title(col)
  plt.xlabel(col)
  plt.ylabel(f'Values of {col}')
  i+=1

plt.tight_layout()
plt.show()
```



It can be observed that there are no outliers in the numeric columns (all the observations are within the whiskers which represent the mimimum and maximum of the range of values)

2) **Data Preprocessing (10 Points)**

Check for duplicate rows

```
[73] df[df.duplicated()].shape

(0, 8)
```

Splitting teat and train data

```
[74] numeric_cols.remove('Chance of Admit')
```

```
x = df[numeric_cols + cat_cols]
y = df[['Chance of Admit']]
```

```
[76] x.head()
```

	GRE Score	TOEFL Score	CGPA	University Rating	SOP	LOR	Research
0	337	118	9.65	4	4.5	4.5	True
1	324	107	8.87	4	4.0	4.5	True
2	316	104	8.00	3	3.0	3.5	True
3	322	110	8.67	3	3.5	2.5	True
4	314	103	8.21	2	2.0	3.0	False

```
y.head()
```

	Chance of Admit
0	0.92
1	0.76
2	0.72
3	0.80
4	0.65

```
[78] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
                                                         random_state=42)

     print(f'Shape of x_train: {x_train.shape}')
     print(f'Shape of x_test: {x_test.shape}')
     print(f'Shape of y_train: {y_train.shape}')
     print(f'Shape of y_test: {y_test.shape}')

     Shape of x_train: (400, 7)
     Shape of x_test: (100, 7)
     Shape of y_train: (400, 1)
     Shape of y_test: (100, 1)
```

Label encoding and standardization

```python
[79] label_encoders = {}
     for col in cat_cols:
         label_encoders[col] = LabelEncoder()
```

```python
[80] for col in cat_cols:
       label_encoders[col].fit(x[col])
```

```python
[81] for col in cat_cols:
       x_train[col] = label_encoders[col].transform(x_train[col])
       x_test[col] = label_encoders[col].transform(x_test[col])
```

```python
[82] x_cat_encoded = pd.concat([x_train, x_test])
     x_cat_encoded.head(10)
```

	GRE Score	TOEFL Score	CGPA	University Rating	SOP	LOR	Research
249	321	111	8.83	2	5	6	1
433	316	111	8.54	3	6	8	0
19	303	102	8.50	2	5	4	0
322	314	107	8.27	1	3	6	0
332	308	106	8.21	2	5	3	1
56	316	102	7.40	2	2	4	0
301	319	108	8.76	1	3	4	0
229	324	111	9.01	3	4	4	1
331	311	105	8.12	1	4	2	1
132	309	105	8.56	4	5	5	0

Next steps: Generate code with x_cat_encoded | View recommended plots

```
[83] scaler_x = MinMaxScaler()
```

```
[84] scaler_x.fit(x_cat_encoded)
```

```
▾ MinMaxScaler
MinMaxScaler()
```

```
[85] all_cols = x_train.columns
```

```
[86] x_train[all_cols]=scaler_x.transform(x_train[all_cols])
     x_test[all_cols]=scaler_x.transform(x_test[all_cols])
```

```
x_test.head()
```

	GRE Score	TOEFL Score	CGPA	University Rating	SOP	LOR	Research
361	0.88	0.857143	0.878205	0.75	0.750	0.625	1.0
73	0.48	0.571429	0.717949	0.75	0.875	0.750	1.0
374	0.50	0.464286	0.272436	0.25	0.250	0.375	0.0
155	0.44	0.607143	0.605769	0.50	0.500	0.500	0.0
104	0.72	0.714286	0.721154	0.50	0.625	0.500	1.0

3) Model building (10 Points)

Building linear regression model

```
[ ] model_lr = LinearRegression()
```

```
[ ] model_lr.fit(x_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
[ ] y_pred_train = model_lr.predict(x_train)
    y_pred_test = model_lr.predict(x_test)
```

Calculate MAE,RMSE,R2 score, Adj R2 score for test and train data.

```
[ ] def model_evaluation(y_actual, y_forecast, model):
      n = len(y_actual)
      if len(model.coef_.shape)==1:
        p = len(model.coef_)
      else:
        p = len(model.coef_[0])
      MAE = np.round(mean_absolute_error(y_true=y_actual, y_pred=y_forecast),2)
      RMSE = np.round(mean_squared_error(y_true=y_actual,
                                         y_pred=y_forecast, squared=False),2)
      r2 = np.round(r2_score(y_true=y_actual, y_pred=y_forecast),2)
      adj_r2 = np.round(1 - ((1-r2)*(n-1)/(n-p-1)),2)
      return print(f"MAE: {MAE}\nRMSE: {RMSE}\nR2 Score: {r2}\nAdjusted R2: {adj_r2}")
```

```
[ ] model_evaluation(y_train.values, y_pred_train, model_lr)

    MAE: 0.04
    RMSE: 0.06
    R2 Score: 0.82
    Adjusted R2: 0.82
```

```
[ ] model_evaluation(y_test.values, y_pred_test, model_lr)

    MAE: 0.04
    RMSE: 0.06
    R2 Score: 0.82
    Adjusted R2: 0.81
```

Since there is no difference in the loss scores of training and test data, 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 the prediction

Calculate weights and intercept.

```
[ ] for feature,weight in zip(x_train.columns, model_lr.coef_[0]):
      print(f"Weight of {feature}: {np.round(weight,2)}")
```

```
Weight of GRE Score: 0.12
Weight of TOEFL Score: 0.08
Weight of CGPA: 0.35
Weight of University Rating: 0.01
Weight of SOP: 0.01
Weight of LOR: 0.07
Weight of Research: 0.02
```

```
[ ] model_lr.intercept_
```

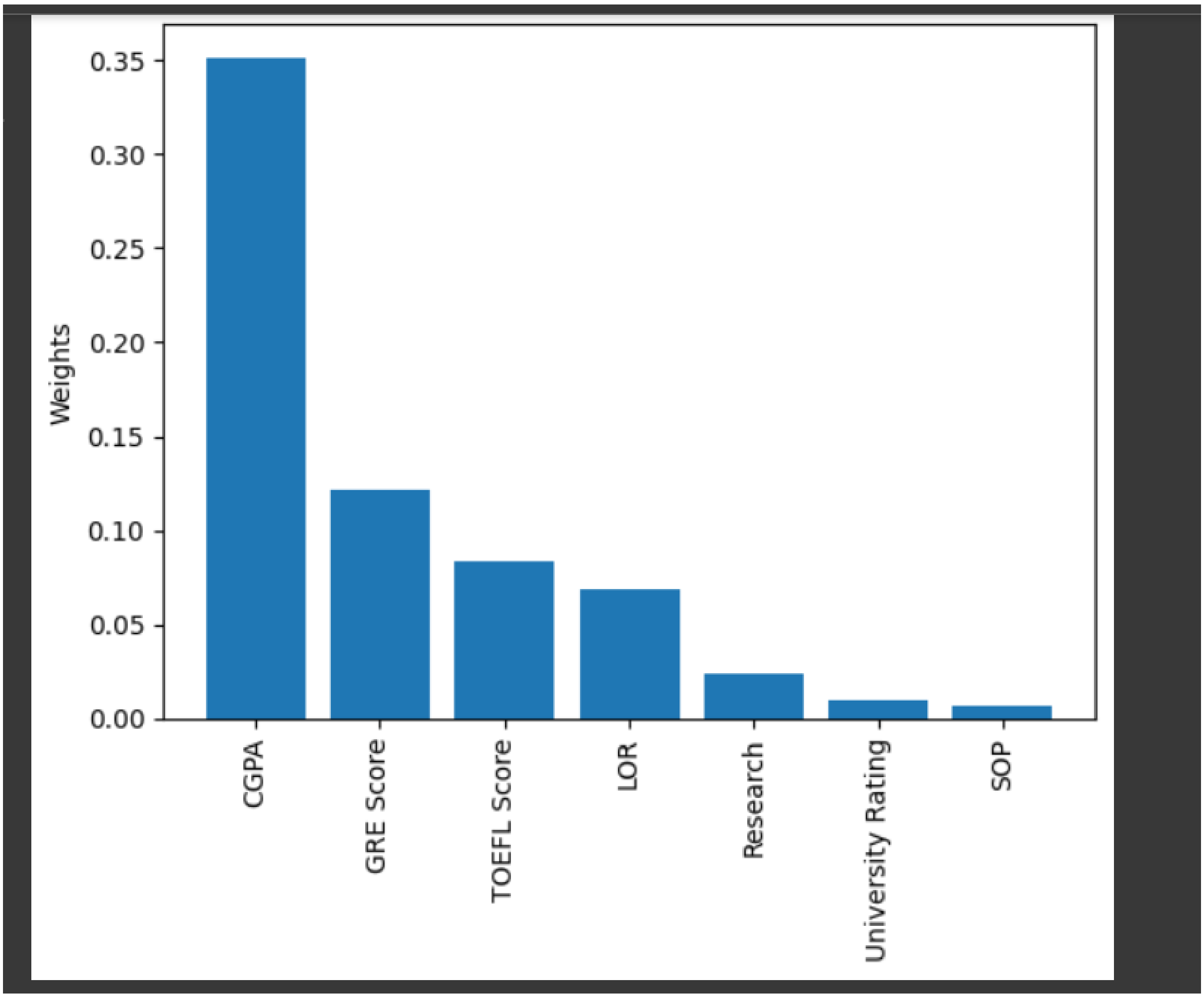
```
array([0.34696506])
```

Plot features and weights

```
model_weights=list(zip(x_train.columns, model_lr.coef_[0]))
model_weights.sort(key=lambda x:x[1], reverse=True)

features = [i[0] for i in model_weights]
weights = [i[1] for i in model_weights]

plt.bar(x=features, height=weights)
plt.title('Model Coefficients')
plt.ylabel('Weights')
plt.xticks(rotation=90)
plt.show();
```



- CGPA & GRE scores have the highest weight
- SOP, University rating, and research have the lowest weights

4) Testing the assumptions of the linear regression model (50 Points)

Multicolinearity Check

VIF (Variance Inflation Factor) is a measure that quantifies the severity of multicollinearity in a regression analysis. It assesses how much the variance of the estimated regression coefficient is inflated due to collinearity.

The formula for VIF is as follows:

VIF(j) = 1 / (1 - R(j)^2)

Where:

j represents the jth predictor variable. R(j)^2 is the coefficient of determination (R-squared) obtained from regressing the jth predictor variable on all the other predictor variables.

```
[ ] vif = pd.DataFrame()
    vif['Variable'] = x_train.columns
    vif['VIF'] = [variance_inflation_factor(x_train.values, i) for i in range(x_train.shape[1])]
    vif
```

	Variable	VIF
0	GRE Score	31.185925
1	TOEFL Score	26.753950
2	CGPA	41.732265
3	University Rating	10.837374
4	SOP	18.864173
5	LOR	14.657099
6	Research	3.366187

We see that almost all the variables (excluding research) have a very high level of colinearity. This was also observed from the correlation heatmap which showed strong positive correlation between GRE score, TOEFL score and CGPA.

Mean of residuals

Mean of Residuals

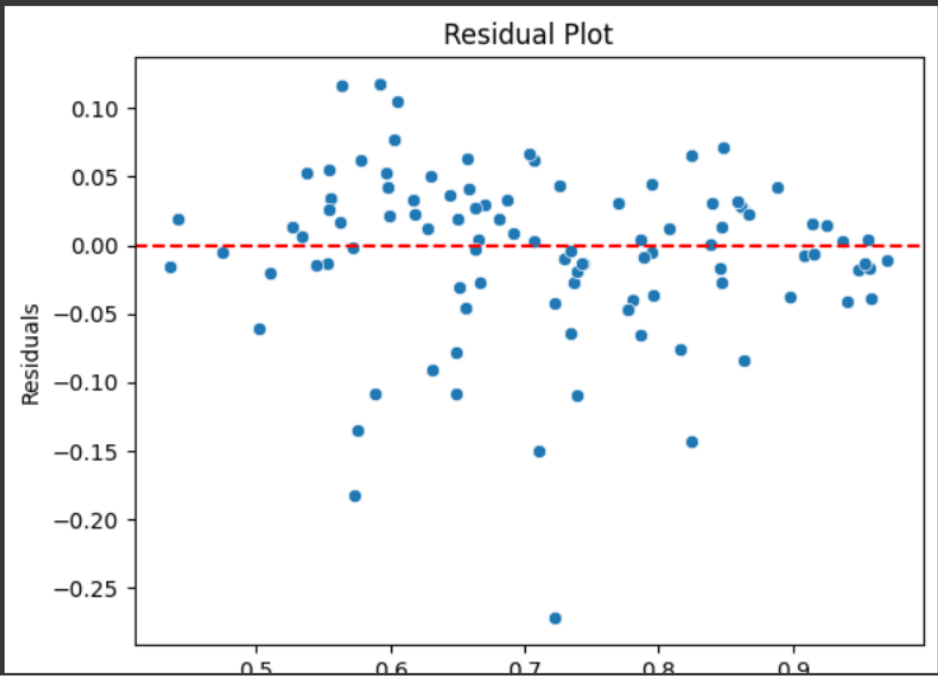
```
[ ] residuals = y_test.values - y_pred_test
    residuals.reshape((-1,))
    print('Mean of Residuals: ', residuals.mean())
```

```
Mean of Residuals:  -0.005453623717661285
```

Mean of Residuals: -0.005453623717661285
Since the mean of residuals is very close to 0, we can say that the model is unbiased

Linearity of variables

```python
sns.scatterplot(x = y_pred_test.reshape((-1,)), y=residuals.reshape((-1,)))
plt.title('Residual Plot')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.axhline(y=0, color='r', linestyle='--')
plt.show();
```

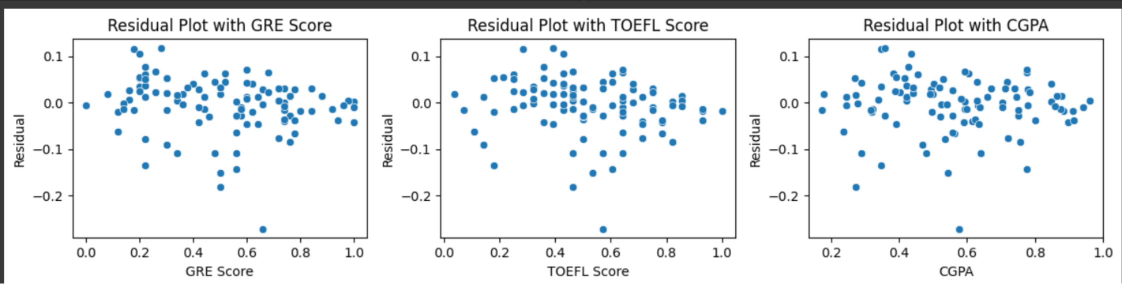


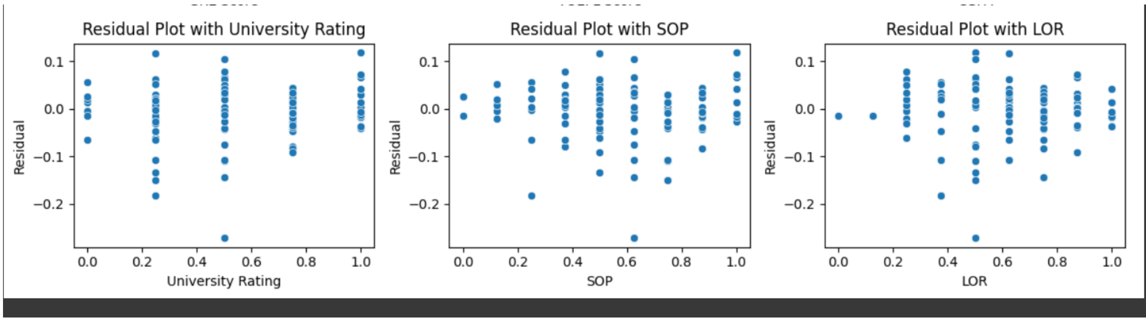
Since the residual plot shows no clear pattern or trend in residuals, we can conclude that linearity of variables exists

Homoscedasticity

```python
plt.figure(figsize=(12,6))
i=1
for col in x_test.columns[:-1]:
  ax = plt.subplot(2,3,i)
  sns.scatterplot(x=x_test[col].values.reshape((-1,)), y=residuals.reshape((-1,)))
  plt.title(f'Residual Plot with {col}')
  plt.xlabel(col)
  plt.ylabel('Residual')
  i+=1

plt.tight_layout()
plt.show();
```

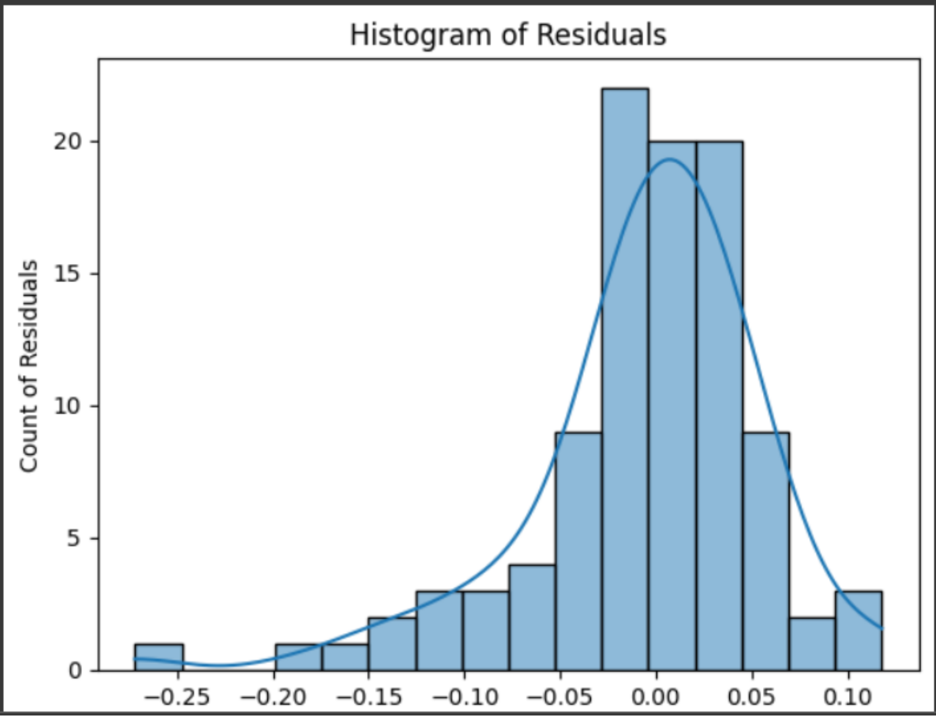




Since we do not see any significant change in the spread of residuals with respect to change in independent variables, we can conclude that homoscedasticity is met.

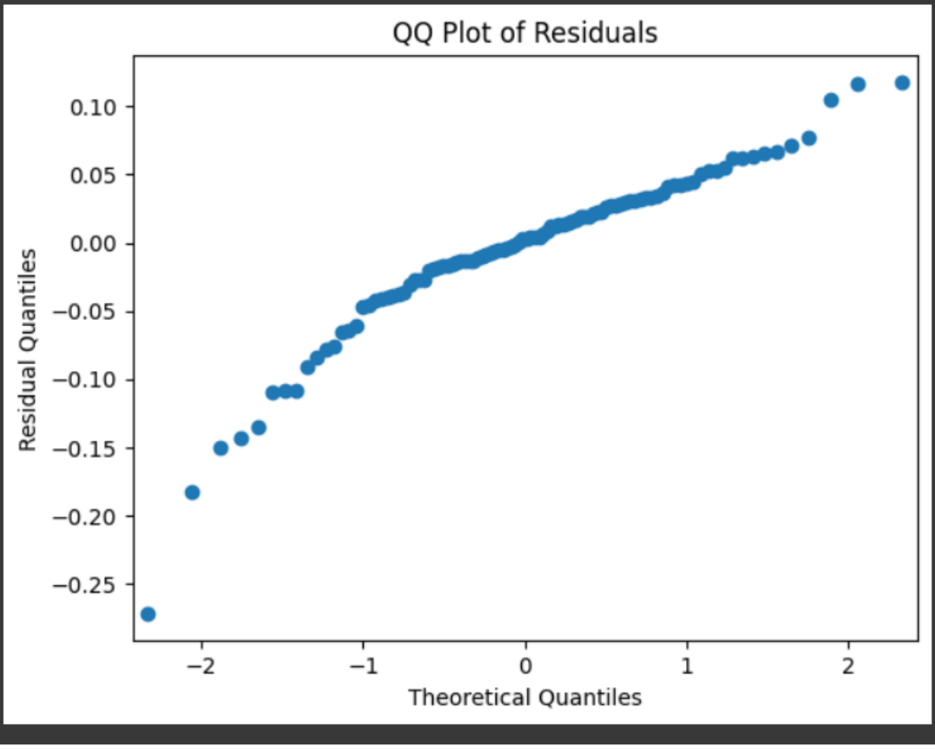
Normality of residuals

```
sns.histplot(residuals.reshape((-1,)), 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

```
[ ] sm.qqplot(residuals.reshape((-1,)))
    plt.title('QQ Plot of Residuals')
    plt.ylabel('Residual Quantiles')
    plt.show();
```



The QQ plot shows that residuals are slightly deviating from the straight diagonal.

5) **Model performance evaluation (10 Points)**
Lasso and ridge regression

```
[ ] model_ridge = Ridge()
    model_lasso = Lasso()
```

```
[ ] model_ridge.fit(x_train, y_train)
    model_lasso.fit(x_train, y_train)
```

```
▾ Lasso
Lasso()
```

```
[ ] y_train_ridge = model_ridge.predict(x_train)
    y_test_ridge = model_ridge.predict(x_test)

    y_train_lasso = model_lasso.predict(x_train)
    y_test_lasso = model_lasso.predict(x_test)
```

```
[ ] print('Ridge Regression Training Accuracy\n')
    model_evaluation(y_train.values, y_train_ridge, model_ridge)
    print('\n\nRidge Regression Test Accuracy\n')
    model_evaluation(y_test.values, y_test_ridge, model_ridge)
    print('\n\nLasso Regression Training Accuracy\n')
    model_evaluation(y_train.values, y_train_lasso, model_lasso)
    print('\n\nLasso Regression Test Accuracy\n')
    model_evaluation(y_test.values, y_test_lasso, model_lasso)
```

```
Ridge Regression Training Accuracy

MAE: 0.04
RMSE: 0.06
R2 Score: 0.82
Adjusted R2: 0.82


Ridge Regression Test Accuracy

MAE: 0.04
RMSE: 0.06
R2 Score: 0.82
Adjusted R2: 0.81


Lasso Regression Training Accuracy

MAE: 0.11
RMSE: 0.14
R2 Score: 0.0
Adjusted R2: -0.02


Lasso Regression Test Accuracy

MAE: 0.12
RMSE: 0.14
R2 Score: -0.01
Adjusted R2: -0.09
```

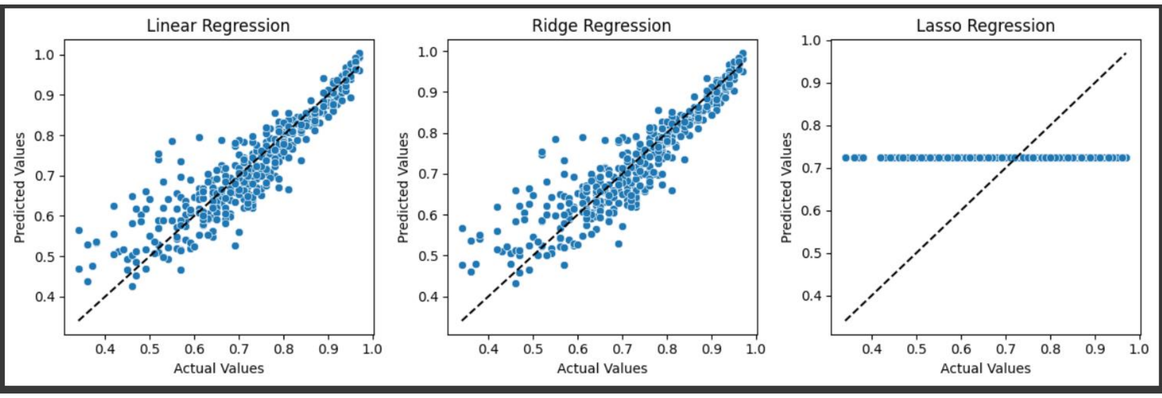
While Linear Regression and Ridge regression have similar scores, Lasso regression has not performed well on both training and test data

Actual v/s Predicted values for training data

```python
actual_values = y_train.values.reshape((-1,))
predicted_values = [y_pred_train.reshape((-1,)), y_train_ridge.reshape((-1,)), y_train_lasso.reshape((-1,))]
model = ['Linear Regression', 'Ridge Regression', 'Lasso Regression']

plt.figure(figsize=(12,4))
i=1
for preds in predicted_values:
  ax = plt.subplot(1,3,i)
  sns.scatterplot(x=actual_values, y=preds)
  plt.plot([min(actual_values),max(actual_values)], [min(actual_values),max(actual_values)], 'k--')
  plt.xlabel('Actual Values')
  plt.ylabel('Predicted Values')
  plt.title(model[i-1])
  i+=1

plt.tight_layout()
plt.show();
```



We can observe that both Linear Regression and Ridge Regression have similar accuracy while Lasso regression has oversimplified the model.

This is the reason that the r2 score of Lasso regression is 0. It doesn't capture any variance in the target variable. It has predicted the same value across all instances.

6) Actionable Insights & Recommendations (10 Points)

Insights:

- 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