inesscase-linear-regression-model

April 11, 2024

#Jamboore Education - BusinessCase

[337]:
```
# importing libraries -
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
from statsmodels.api import OLS
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


# reading the data file -
df=pd.read_csv('Jamboree_Admission.csv')
df.head()
```

[337]:
```
   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
```

[338]:
```
df.shape
```

[338]:
```
(500, 9)
```

[339]:
```
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
```

[340]:
```
#Dropping the unique row identifier
df.drop(columns=['Serial No.'], inplace=True)
```

[341]:
```
#Checking any duplicate data is present.
df.duplicated().sum()
```

[341]: 0

Let's check the statistical information of the data

There are no NULL values in the data.

[342]:
```
df.describe()
```

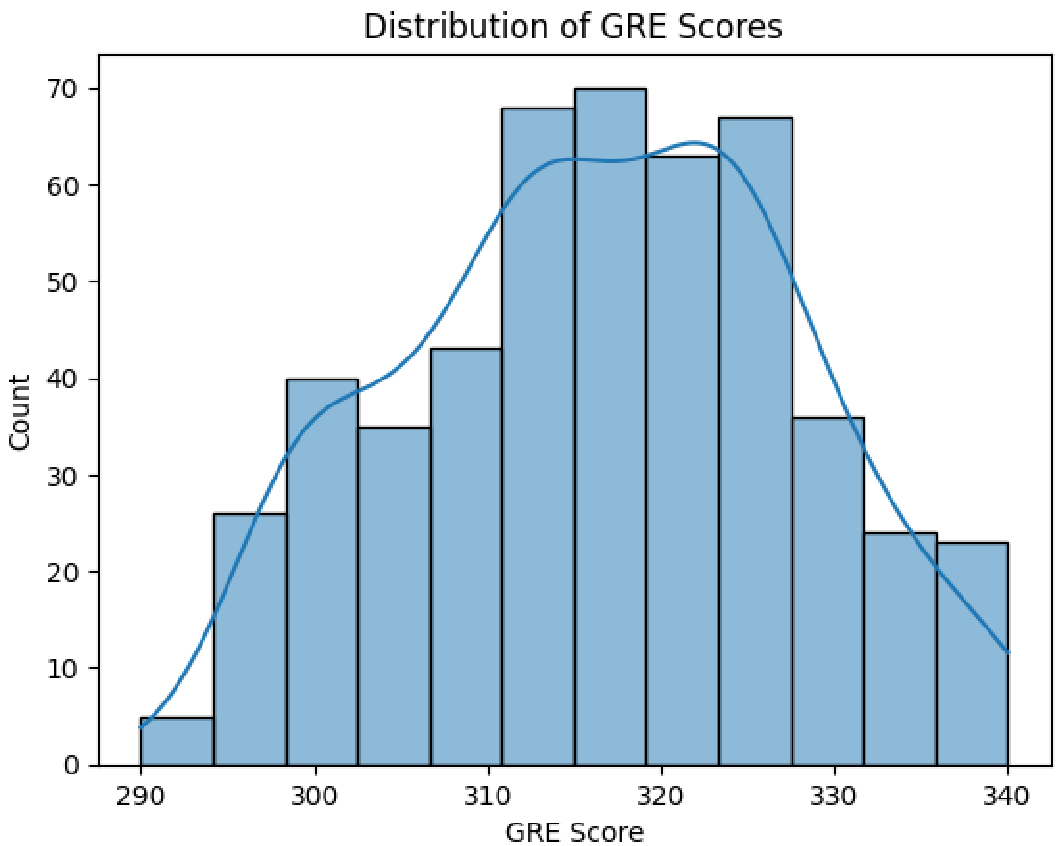
[342]:
```
        GRE Score  TOEFL Score  University Rating         SOP        LOR  \
count  500.000000   500.000000         500.000000  500.000000  500.00000   
mean   316.472000   107.192000           3.114000    3.374000    3.48400   
std     11.295148     6.081868           1.143512    0.991004    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  
```

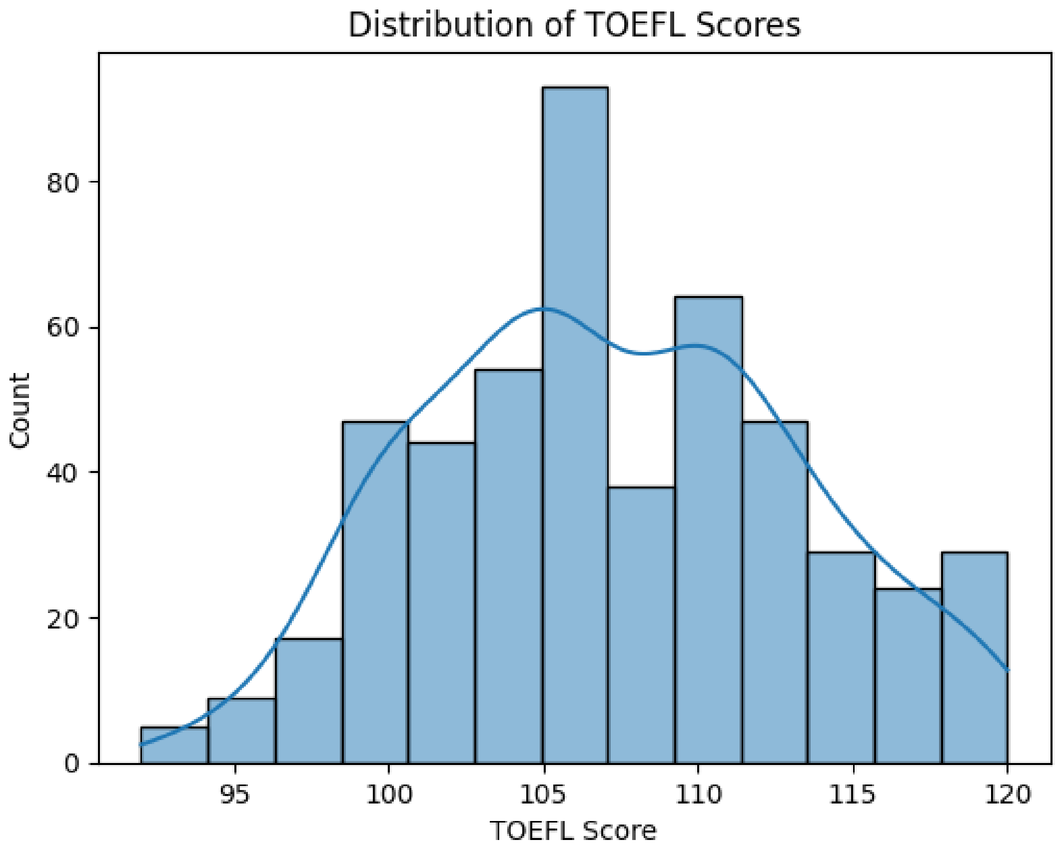
```
75%       9.040000    1.000000              0.82000
max       9.920000    1.000000              0.97000
```

As we can see the mean values and 50% percentile(i.e.,median) values are almost same, then we can say there might be no outliers in the data.

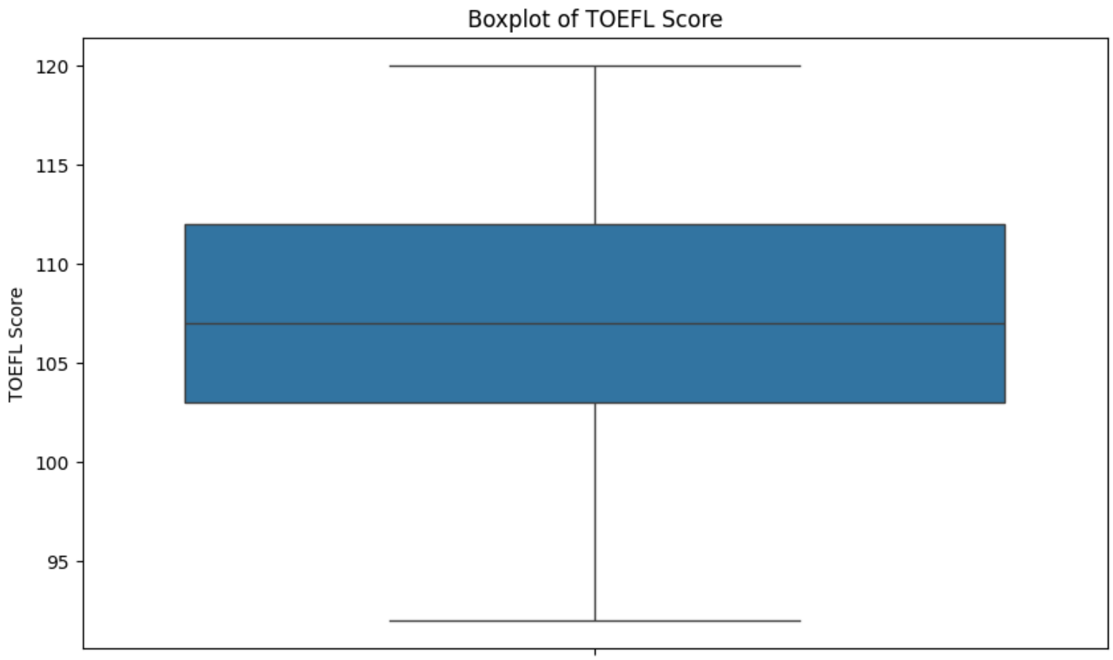
[343]:
```
sns.histplot(df['GRE Score'], kde=True)
plt.title('Distribution of GRE Scores')
plt.show()
```



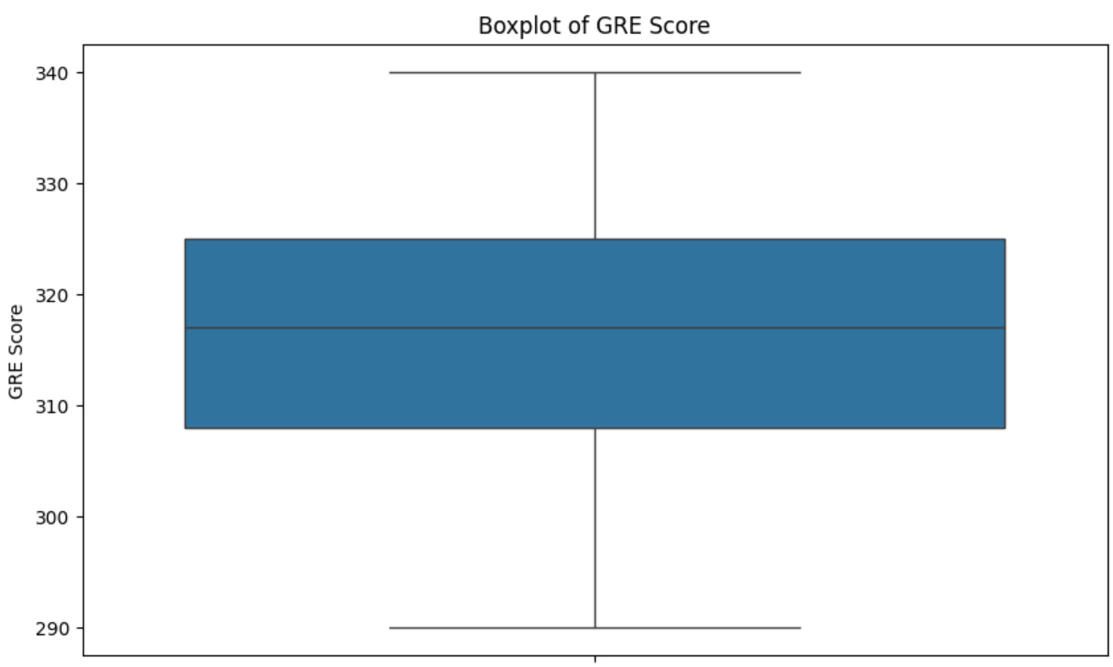
[344]:
```
sns.histplot(df['TOEFL Score'], kde=True)
plt.title('Distribution of TOEFL Scores')
plt.show()
```



[345]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(df['TOEFL Score'])
plt.title('Boxplot of TOEFL Score')
plt.show()
```



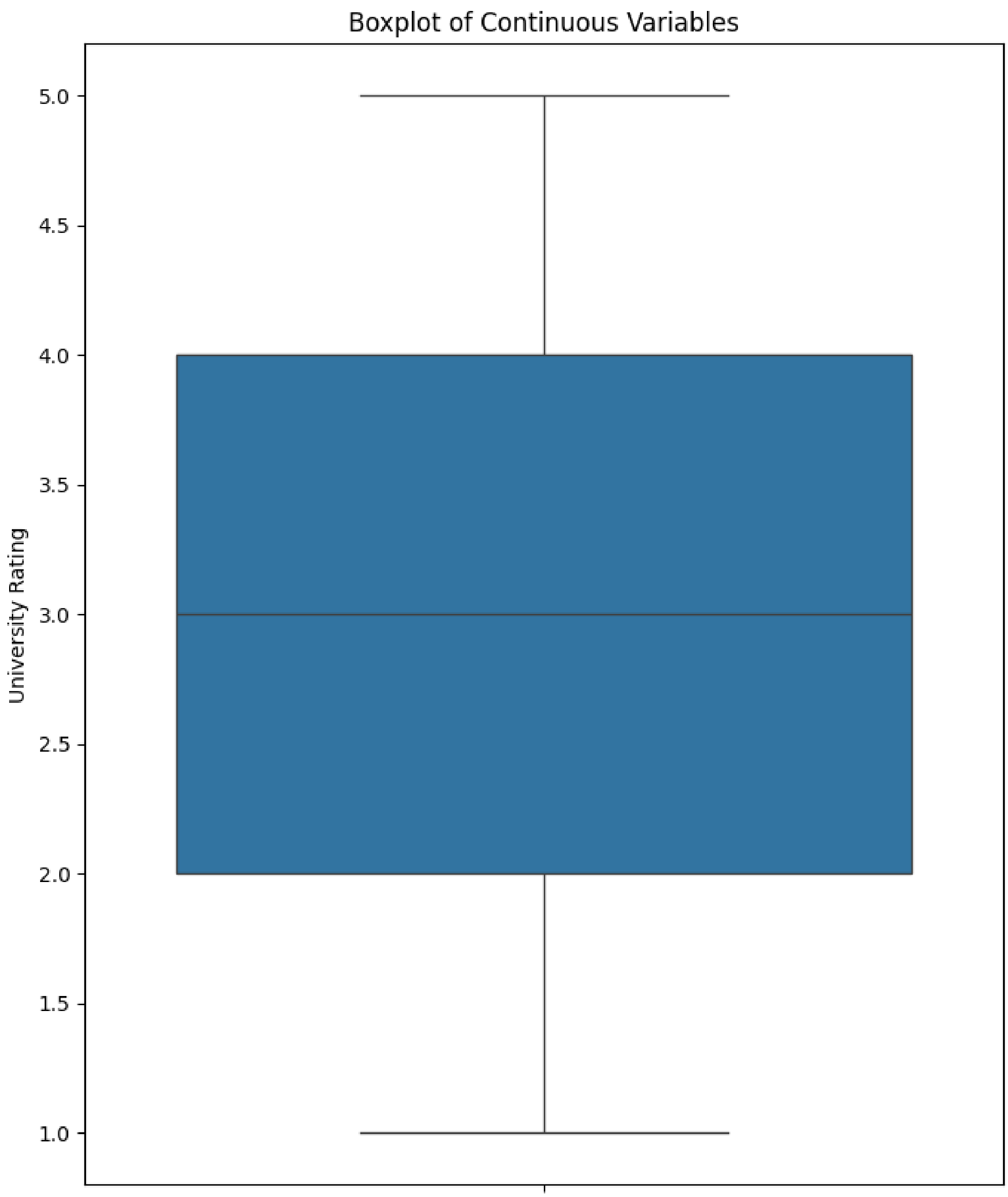
[346]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(df['GRE Score'])
plt.title('Boxplot of GRE Score')
plt.show()
```



[347]:
```
df.info()
```

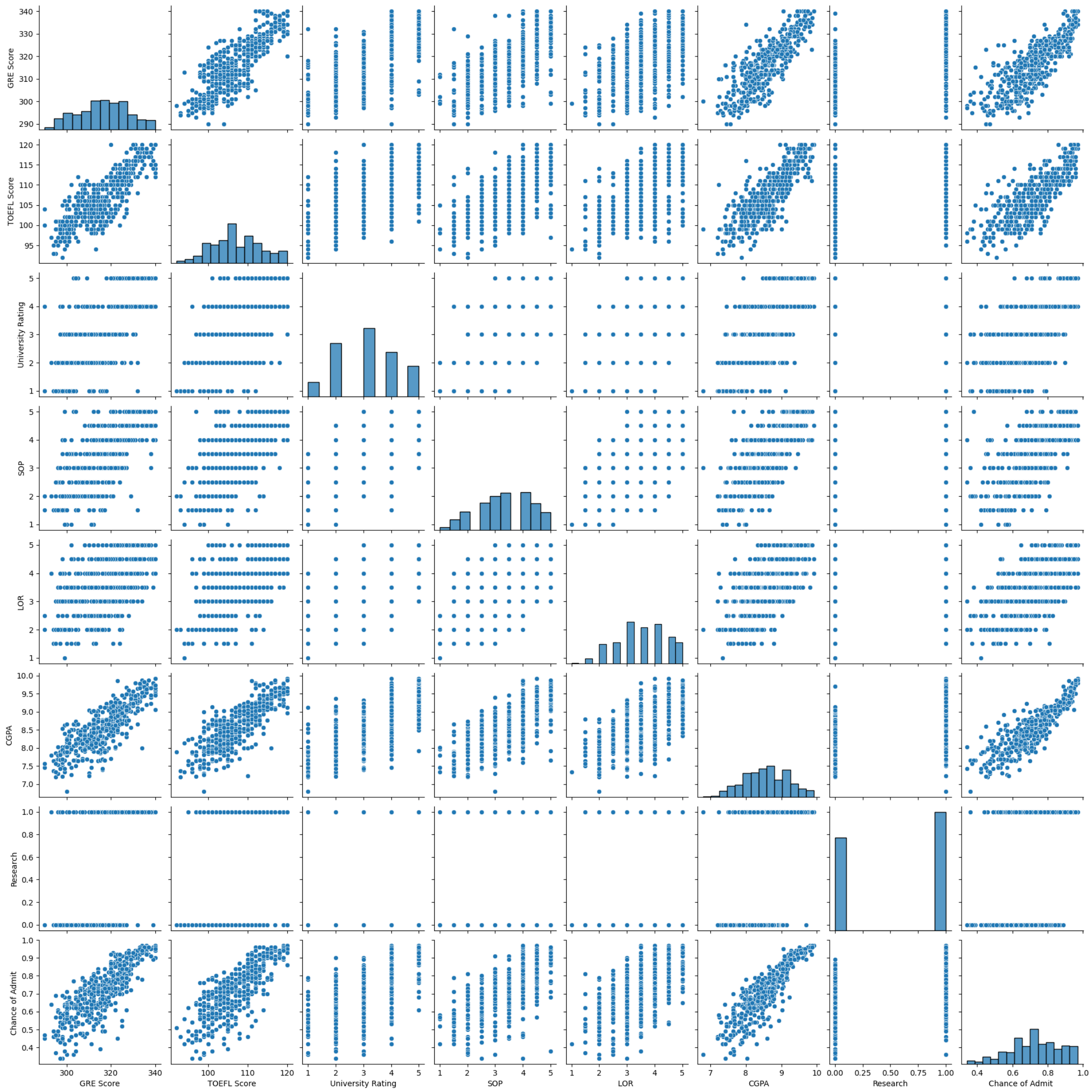
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   GRE Score          500 non-null    int64
 1   TOEFL Score        500 non-null    int64
 2   University 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
```

[348]:
```
plt.figure(figsize=(8,10))
sns.boxplot(df['University Rating'])
plt.title('Boxplot of Continuous Variables')
plt.show()
```

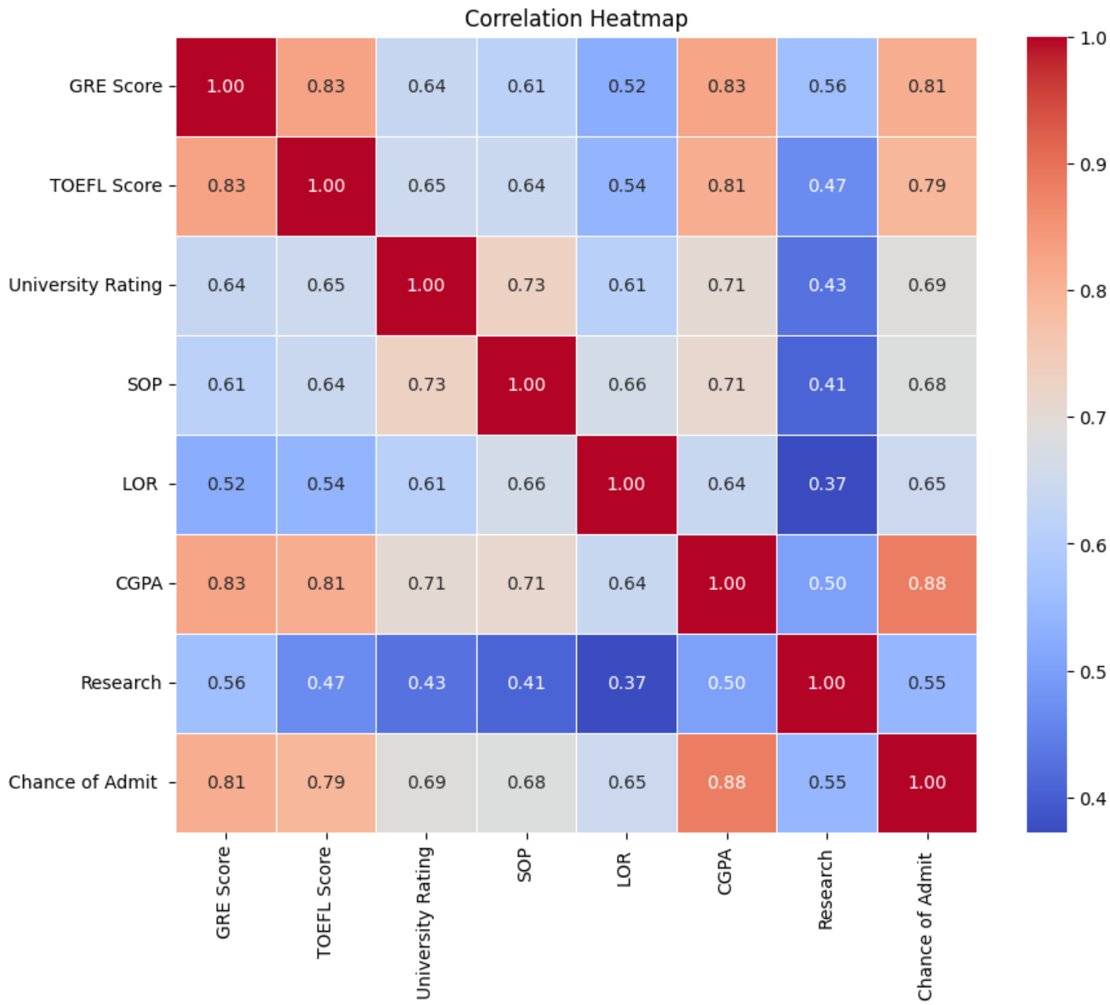


From the visual analysis also, we can conclude that there are no outliers in the data.

[349]:
```
sns.pairplot(df)
plt.show()
```



[350]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



From the above plots, we can see many features are correlated highly,i.e., 1)GRE Score vs TOEFL Score, 2)GRE Score vs CGPA, 3)TOEFL Score vs CGPA, 4)CGPA vs Chance of Admit

[351]:
```
df.head()
```

[351]:
```
   GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  Research  \
0        337          118                  4  4.5   4.5  9.65         1
1        324          107                  4  4.0   4.5  8.87         1
2        316          104                  3  3.0   3.5  8.00         1
3        322          110                  3  3.5   2.5  8.67         1
4        314          103                  2  2.0   3.0  8.21         0

   Chance of Admit
0             0.92
1             0.76
2             0.72
```

```
3               0.80
4               0.65
```

#Creating LinearRegression Model

Splitting the training and testing dataset

[352]:
```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)
```

[353]:
```
# Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)
```

[353]:
```
LinearRegression()
```

[354]:
```
# Get the model coefficients
coefficients = model.coef_

# Get the column names
column_names = X_train.columns

# Create a DataFrame to display coefficients with column names
coefficients_df = pd.DataFrame({'Feature': column_names, 'Coefficient': coefficients})

print("Model Coefficients:")
print(coefficients_df)
```

```
Model Coefficients:
             Feature  Coefficient
0          GRE Score     0.002004
1        TOEFL Score     0.003258
2  University Rating     0.005675
3                SOP     0.002400
4               LOR      0.016894
5               CGPA     0.111939
6           Research     0.019461
```

[355]:
```
model.score(X_train,y_train)
```

[355]:
```
0.8279743197944157
```

[359]:
```
def adj_r2(y_true, y_pred, n, p):
    r_squared = r2_score(y_true, y_pred)
    adjusted_r_squared = 1 - ((1 - r_squared) * (n - 1) / (n - p - 1))
```

```
    return adjusted_r_squared
adj_r2(y_test,y_test_pred,len(y_test),X_train.shape[1])
```

[359]: 0.7762693627975559

[360]:
```
print("Train MAE:", mean_absolute_error(y_train, y_train_pred))
print("Test MAE:", mean_absolute_error(y_test, y_test_pred))

print("Train RMSE:", np.sqrt(mean_squared_error(y_train, y_train_pred)))
print("Test RMSE:", np.sqrt(mean_squared_error(y_test, y_test_pred)))

print("Train R2 Score:", r2_score(y_train, y_train_pred))
print("Test R2 Score:", r2_score(y_test, y_test_pred))

print("Train Adjusted R2 Score:", adj_r2(y_train,␣
 ↪y_train_pred,len(y_train),X_train.shape[1]))
print("Test Adjusted R2 Score:", adj_r2(y_test, y_test_pred,len(y_test),X_test.
 ↪shape[1]))
```

```
Train MAE: 0.041995840770610984
Test MAE: 0.045225659905628775
Train RMSE: 0.058708507866976706
Test RMSE: 0.0629453227125008
Train R2 Score: 0.8279743197944157
Test R2 Score: 0.7920887007815671
Train Adjusted R2 Score: 0.8249024326478874
Test Adjusted R2 Score: 0.7762693627975559
```

Assumptions of Linear Regression

1 Checking Multilinearity and removing the featrues having high VIF Score by setting threshold for R2 Score and VIF score.

[362]:
```
vif_thr = 5
r2_thr = 0.75
feat_removed=[]

while True:
  vif_list = [variance_inflation_factor(X_train,i) for i in range(0,len(X_train.
 ↪columns))]
  vif_zip = list(zip(X_train.columns,vif_list))
  vif = pd.DataFrame(vif_zip,columns = ["feature","vif"])
  vif.sort_values(by = "vif",ascending=False,inplace=True)

  cols2 = vif.feature.values
  model.fit(X_train[cols2],y_train)
```

```
    r2_score = model.score(X_train[cols2],y_train)

    vif_sc = vif.iloc[0]["vif"]
    fea = vif.iloc[0]["feature"]

    if (vif_sc<vif_thr) or (r2_score<r2_thr):
      break
    else:
      X_train.drop(columns = fea,inplace=True)
      feat_removed.append(fea)
```

[363]:
```
r2_score
```

[363]: 0.7483007447755545

[364]:
```
vif
```

[364]:

	feature	vif
2	SOP	33.009489
3	LOR	29.564419
0	TOEFL Score	21.498294
1	University Rating	19.372848
4	Research	2.896241

[365]:
```
feat_removed
```

[365]: ['GRE Score', 'CGPA']

Removed the features of GRE Score, CGPA as their VIF Score is high.

[366]:
```
vif.feature.values
```

[366]:
```
array(['SOP', 'LOR ', 'TOEFL Score', 'University Rating', 'Research'],
      dtype=object)
```

[367]:
```
vif
```

[367]:

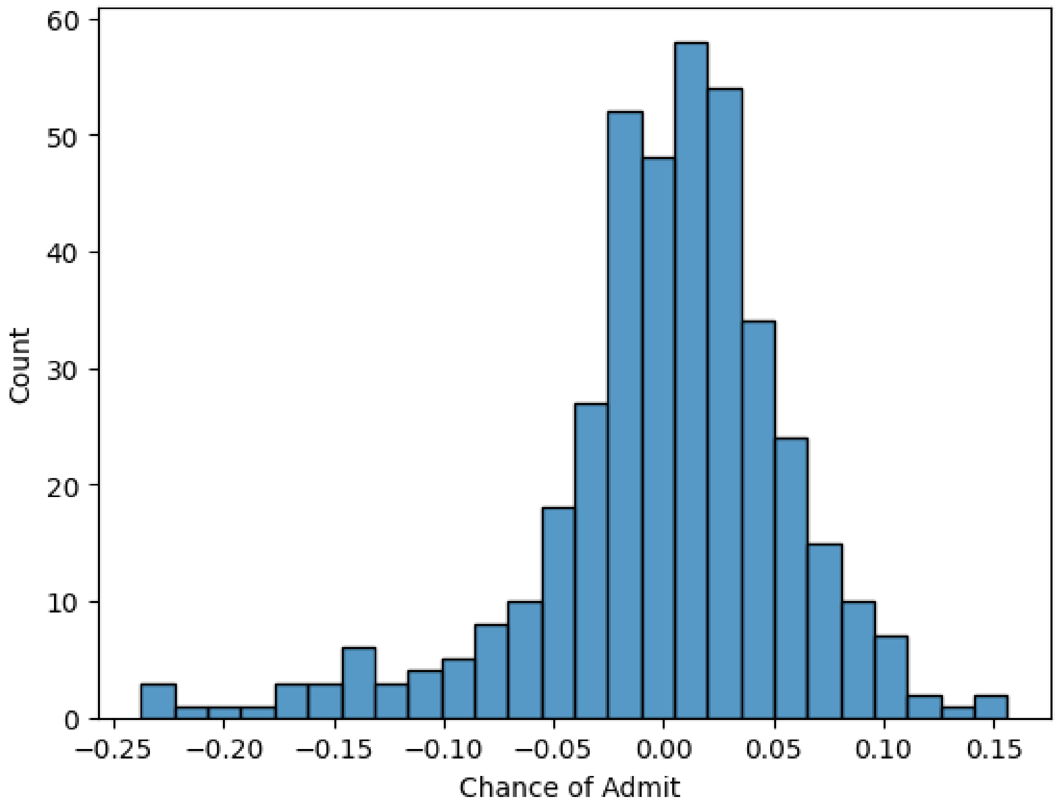
	feature	vif
2	SOP	33.009489
3	LOR	29.564419
0	TOEFL Score	21.498294
1	University Rating	19.372848
4	Research	2.896241

#Checking Normality of Residuals

[368]:
```
residuals = y_train - y_train_pred
```

[369]:
```
import seaborn as sns

sns.histplot(residuals)
plt.show()
```



[370]:
```
#USing Shapiro test to check the normality
from scipy.stats import shapiro
res = shapiro(residuals)
res.statistic
```

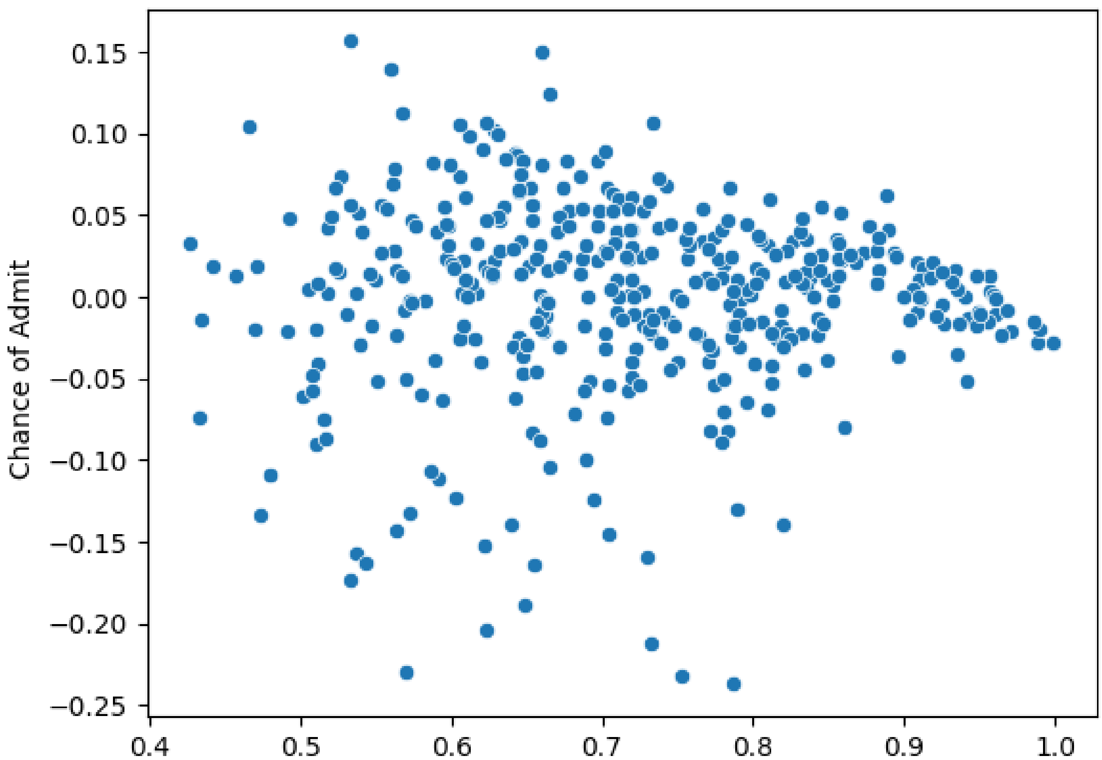
[370]: 0.9305607676506042

As the statistic is very high, the residuals are normally distributed.

#Check for Heteroskedasticity

[371]:
```
sns.scatterplot(x = y_train_pred , y = residuals)
```

[371]: <Axes: ylabel='Chance of Admit '>



[372]:
```
#Goldfeld-Quandt Test #check for heteroskadasticity
#HO = Data is Homoskadastic
from statsmodels.stats.api import het_goldfeldquandt
het_goldfeldquandt(y_train,X_train)
```

[372]: (0.9449806443259853, 0.653427792165528, 'increasing')

As p_value is greater than 0.05, we cannot reject null. Hence, Data is Homoskadastic.

#The mean of residuals should be nearly zero

[373]:
```
residuals.mean()
```

[373]: 1.9831358777366858e-16

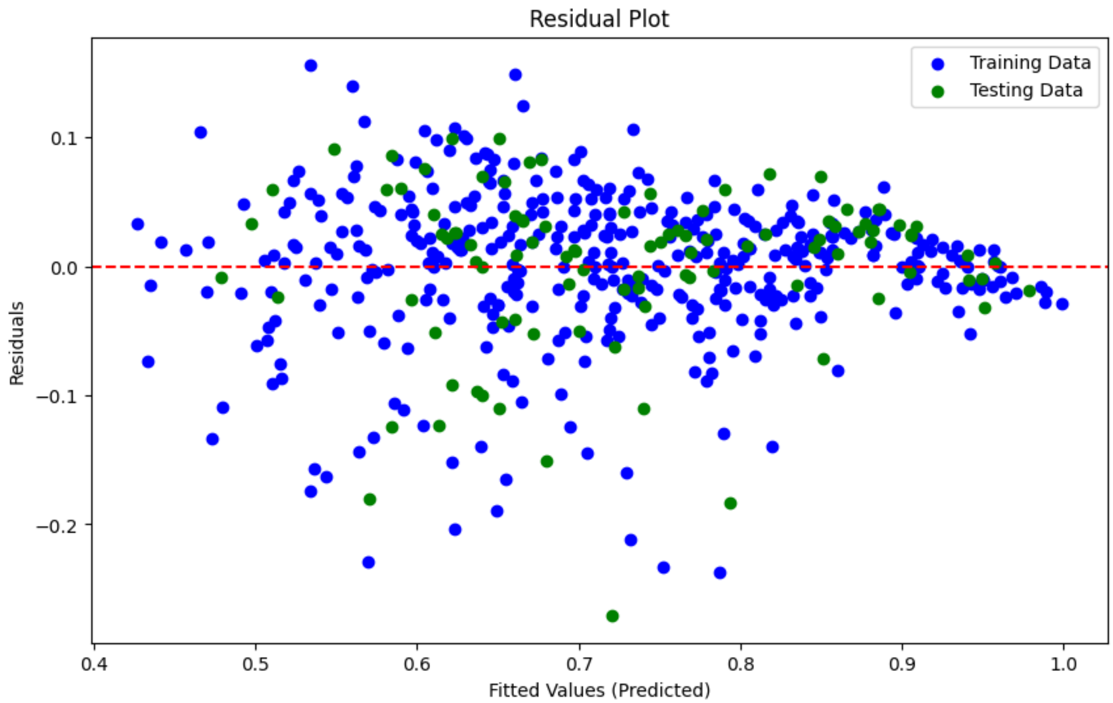
We can see that the mean of the residuals is nearly 0.

#Checking Linearity of variables (no pattern in the residual plot)

[374]:
```
# Calculate residuals
train_residuals = y_train - y_train_pred
test_residuals = y_test - y_test_pred

# Plot residual plots
```

```
plt.figure(figsize=(10, 6))
plt.scatter(y_train_pred, train_residuals, color='blue', label='Training Data')
plt.scatter(y_test_pred, test_residuals, color='green', label='Testing Data')
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Residual Plot')
plt.xlabel('Fitted Values (Predicted)')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```



We can see the residuals do not follow any pattern

Actionable Insights:

1. Variables with higher coefficients have a stronger impact on the admission decision. For example, As GRE scores have a high coefficient, it suggests that higher GRE scores significantly increase the chances of admission.

2. We should consider incorporating additional relevant data sources to improve the model's predictive performance. This could include data on extracurricular activities, internships, personal statements, or letters of recommendation.

Recommendations:

1. We can implement the model on Jamboree's website to provide students with personalized insights into their chances of admission to IVY league colleges.

2. We can offer a user-friendly interface where students can input their academic credentials and receive an estimated probability of admission based on the model's predictions.
3. We can also rovide explanations and recommendations to students on how they can improve their chances of admission based on the model's insights.