Importing the necessary Libraries

In [212…

```
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.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.stats.diagnostic import het_goldfeldquandt
from scipy import stats
```

In [213…

```
!gdown --id 1XAQerm_u7zfaAdYUOcw0ZCaQ7MF5ssYc
```

```
/usr/local/lib/python3.10/dist-packages/gdown/__main__.py:132: FutureWarning: Option `--id` was deprecated in version 4.3.1 and will be removed in 5.0. You don't need to pass it anymore to use a file ID.
  warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1XAQerm_u7zfaAdYUOcw0ZCaQ7MF5ssYc
To: /content/Jamboree.csv
100% 16.2k/16.2k [00:00<00:00, 37.5MB/s]
```

In [214…

```
df = pd.read_csv("Jamboree.csv")
df.head()
```

Out[214]:

	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

Problem Statement

Jamboree has recently launched a feature that estimates the chances of graduate admission into Ivy League colleges for students, specifically from an Indian perspective. The goal of this analysis is to identify and understand the key factors that influence admission decisions and how these factors interrelate. By analyzing historical admission data, we aim to build a predictive model that estimates a student’s probability of gaining admission based on factors such as academic performance, standardized test scores, extracurricular activities, letters of recommendation, and other relevant metrics.

Understanding the columns:

- **Serial No.**: This column represents the unique row identifier for each applicant in the dataset.
- **GRE Scores**: This column contains the GRE (Graduate Record Examination) scores of the applicants, which are measured on a scale of 0 to 340.
- **TOEFL Scores**: This column includes the TOEFL (Test of English as a Foreign Language) scores of the applicants, which are measured on a scale of 0 to 120.
- **University Rating**: This column indicates the rating or reputation of the university that the applicants are associated with.The rating is based on a scale of 0 to 5, with 5 representing the highest rating.
- **SOP**: This column represents the strength of the applicant's statement of purpose, rated on a scale of 0 to 5, with 5 indicating a strong and compelling SOP.
- **LOR**: This column represents the strength of the applicant's letter of recommendation, rated on a scale of 0 to 5, with 5 indicating a strong and compelling LOR.
- **CGPA**: This column contains the undergraduate Grade Point Average (GPA) of the applicants, which is measured on a scale of 0 to 10.
- **Research**: This column indicates whether the applicant has research experience (1) or not (0).
- **Chance of Admit**: This column represents the estimated probability or chance of admission for each applicant, ranging from 0 to 1.

Exploratory Data Analysis

```
Non Graphical followed by Graphical Analysis
```

In [215…
```
print(df.shape)
print(100*"-")
print(df.dtypes)
print(100*"-")
print(df.isnull().sum()) # checking for nulls
print(100*"-")
print(df.duplicated().any())  # checking for duplicates
print(100*"-")
print(df.columns)
```

```
(500, 9)
----------------------------------------------------------------------------------------------------
Serial No.              int64
GRE Score               int64
TOEFL Score             int64
University Rating       int64
SOP                   float64
LOR                   float64
CGPA                  float64
Research                int64
Chance of Admit       float64
dtype: object
----------------------------------------------------------------------------------------------------
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
----------------------------------------------------------------------------------------------------
False
----------------------------------------------------------------------------------------------------
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [216…
```
df.rename(columns = {'LOR ':'LOR','Chance of Admit ':'Chance of Admit'},inplace = True)
```

Observations:

- Data set that's provided doesn't have any nulls and duplicated rows.
- The 'Chance of Admit' variable is our target feature since it represents the outcome we want to predict (admission likelihood), while all other variables act as predictors.
- We can drop Serial No. column since it is a unique identifier and it doesn't carry any weight in predicting the target varible Chance of Admit.

In [217…

```
# dropping Seria No. column from the dataframe 'df'
df.drop(columns = ['Serial No.'],inplace = True)
```

In [218…

```
# Understanding the value counts of each and every column
for i in df.columns:
  print(df[i].value_counts())
  print()
  print(100*'-')
```

GRE	Score
312	24
324	23
316	18
321	17
322	17
327	17
311	16
320	16
314	16
317	15
325	15
315	13
308	13
323	13
326	12
319	12
313	12
304	12
300	12
318	12
305	11
301	11
310	11
307	10
329	10
299	10
298	10
331	9
340	9
328	9
309	9
334	8
332	8
330	8
306	7
302	7
297	6
296	5
295	5
336	5
303	5
338	4
335	4

```
333     4
339     3
337     2
290     2
294     2
293     1
Name: count, dtype: int64

----------------------------------------------------------------------------------------------------
TOEFL Score
110    44
105    37
104    29
107    28
106    28
112    28
103    25
100    24
102    24
99     23
101    20
111    20
108    19
113    19
109    19
114    18
116    16
115    11
118    10
98     10
119    10
120     9
117     8
97      7
96      6
95      3
93      2
94      2
92      1
Name: count, dtype: int64

----------------------------------------------------------------------------------------------------
University Rating
3    162
```

```
2    126
4    105
5     73
1     34
Name: count, dtype: int64

----------------------------------------------------------------------------------------------------
SOP
4.0    89
3.5    88
3.0    80
2.5    64
4.5    63
2.0    43
5.0    42
1.5    25
1.0     6
Name: count, dtype: int64

----------------------------------------------------------------------------------------------------
LOR
3.0    99
4.0    94
3.5    86
4.5    63
2.5    50
5.0    50
2.0    46
1.5    11
1.0     1
Name: count, dtype: int64

----------------------------------------------------------------------------------------------------
CGPA
8.76    9
8.00    9
8.12    7
8.45    7
8.54    7
       ..
9.92    1
9.35    1
8.71    1
9.32    1
```

```
7.69    1
Name: count, Length: 184, dtype: int64

----------------------------------------------------------------------------------------------------
Research
1    280
0    220
Name: count, dtype: int64

----------------------------------------------------------------------------------------------------
Chance of Admit
0.71    23
0.64    19
0.73    18
0.72    16
0.79    16
        ..
0.38     2
0.36     2
0.43     1
0.39     1
0.37     1
Name: count, Length: 61, dtype: int64

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

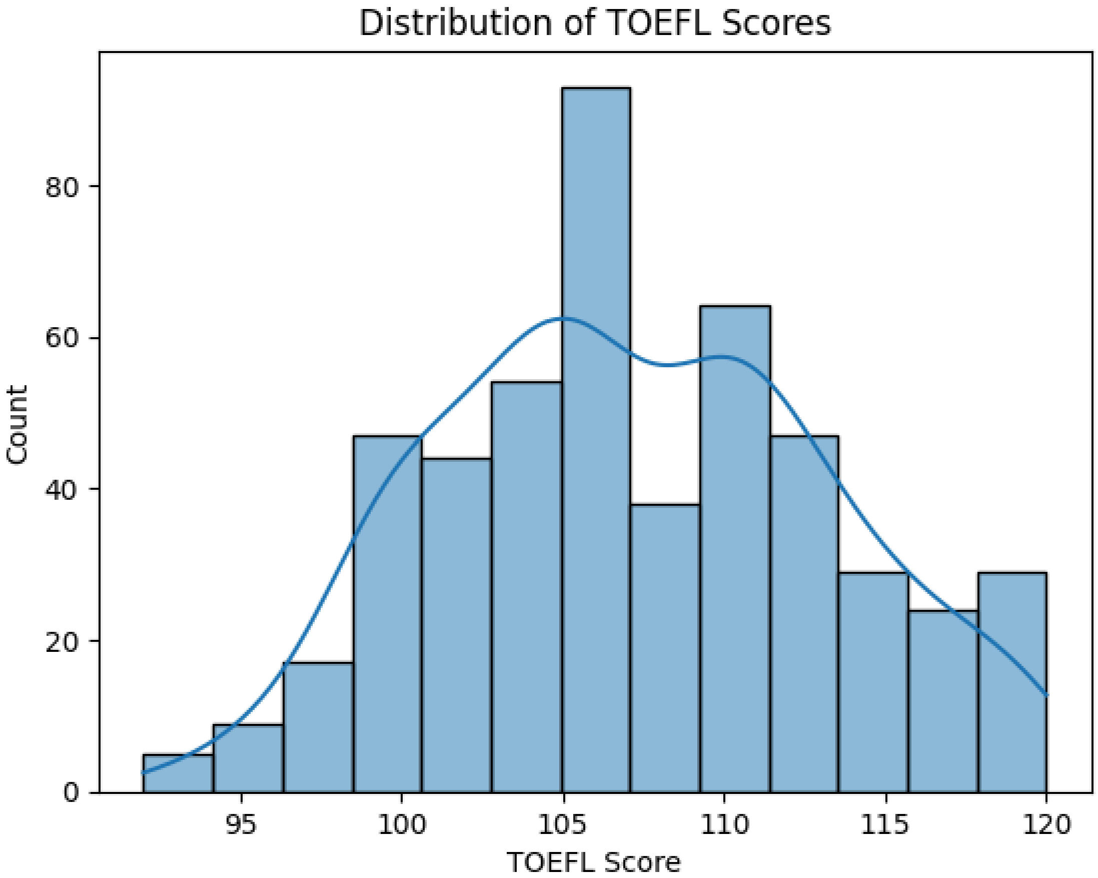
Observations:

- Upon reviewing the dataset, it is observed that columns such as University Rating, SOP, LOR, and Research represent ordinal data, which categorizes ratings or rankings for various features.
- Despite being stored as numerical data types, these columns are fundamentally categorical because they denote ordered categories rather than continuous numerical values.
- In contrast, columns like TOEFL Score, GRE Score, and CGPA contain actual continuous numerical data.

Understanding the distribution of numerical data using hist and kde plots

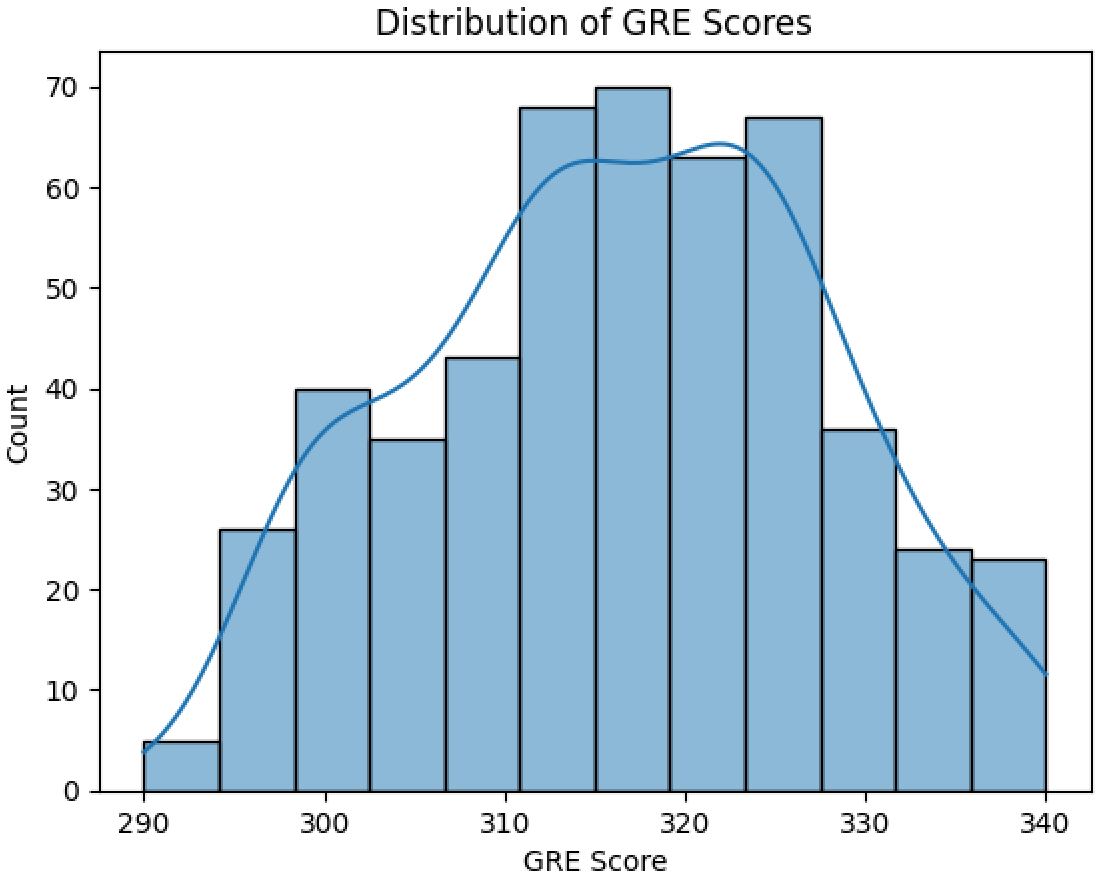
In [219…

```
sns.histplot(data = df, x = 'TOEFL Score',kde  = True)
plt.title("Distribution of TOEFL Scores")
plt.show()
```



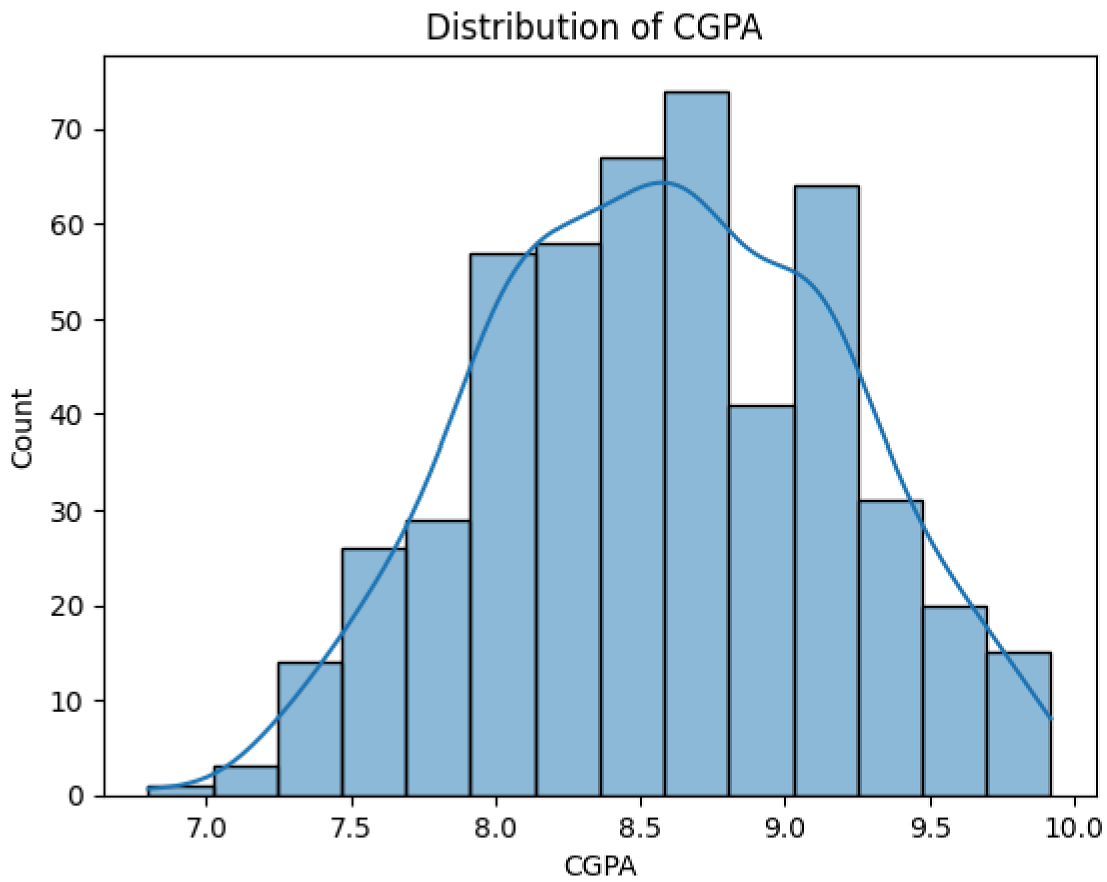
In [220…

```
sns.histplot(data = df , x = 'GRE Score',kde = True)
plt.title("Distribution of GRE Scores")
plt.show()
```



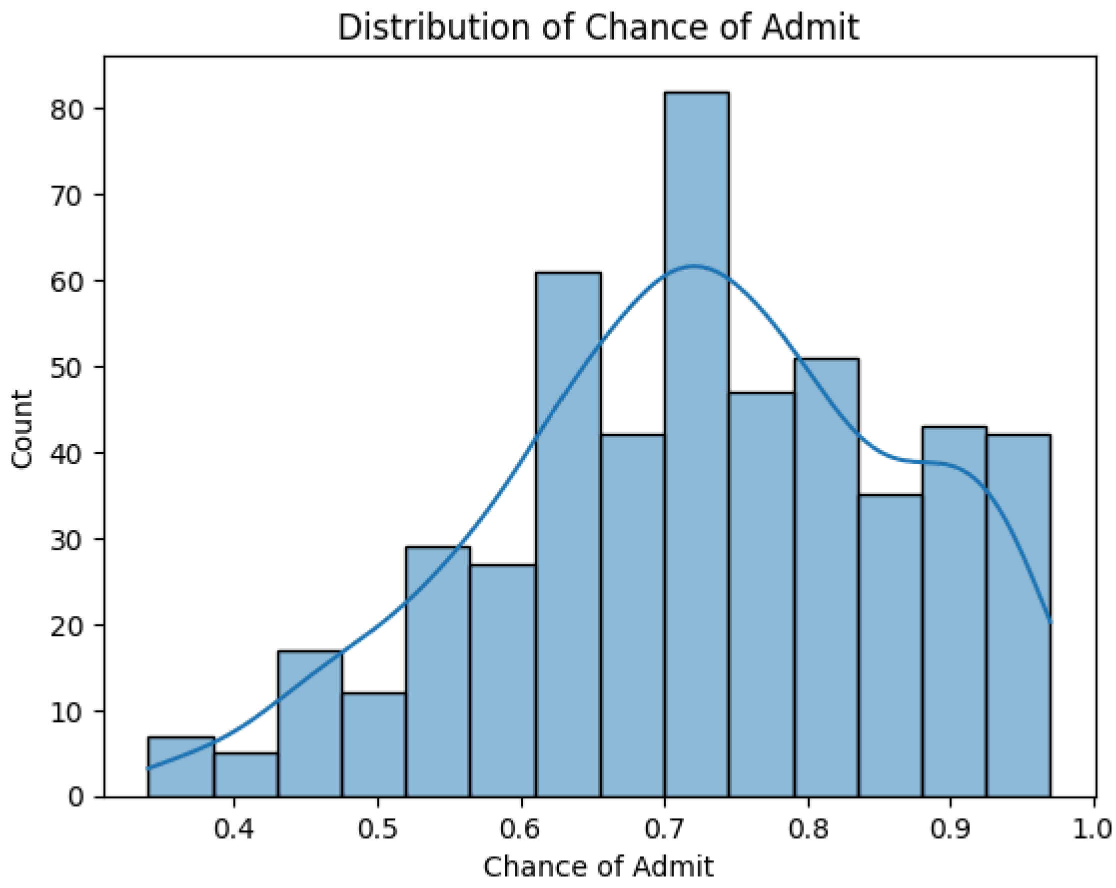
In [221…

```
sns.histplot(data = df , x = 'CGPA',kde = True)
plt.title("Distribution of CGPA ")
plt.show()
```



In [222…

```
sns.histplot(data = df , x = 'Chance of Admit',kde = True)
plt.title("Distribution of Chance of Admit")
plt.show()
```

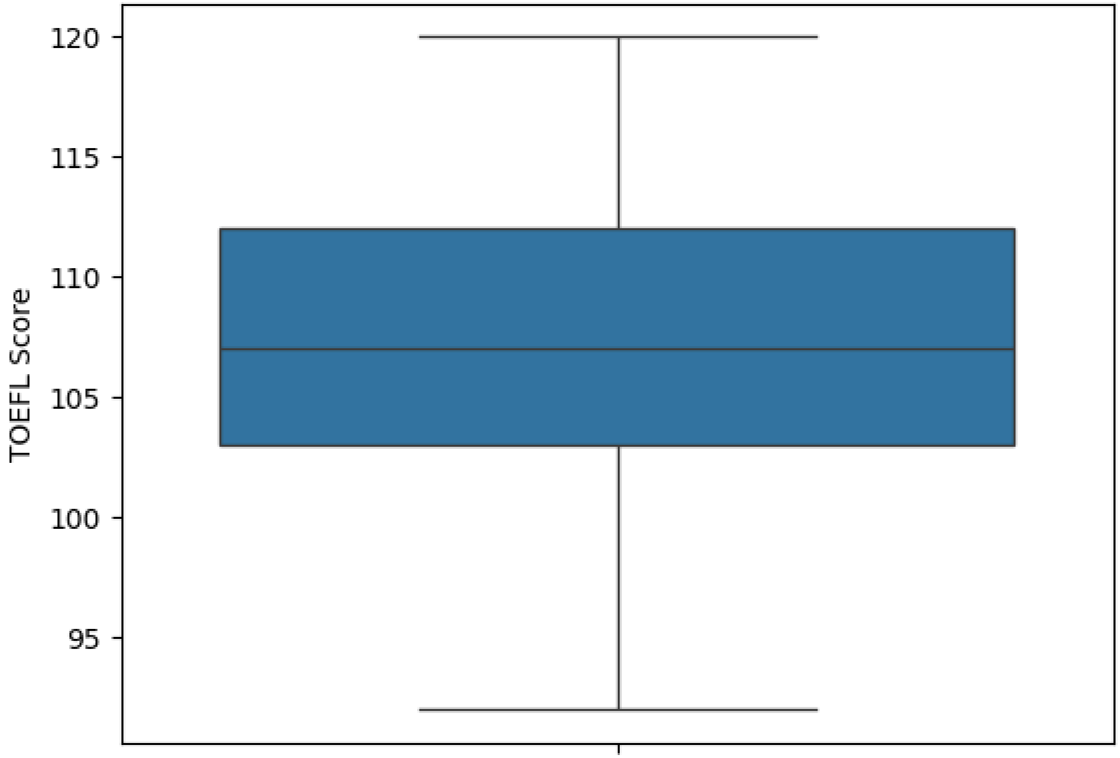


Observations:

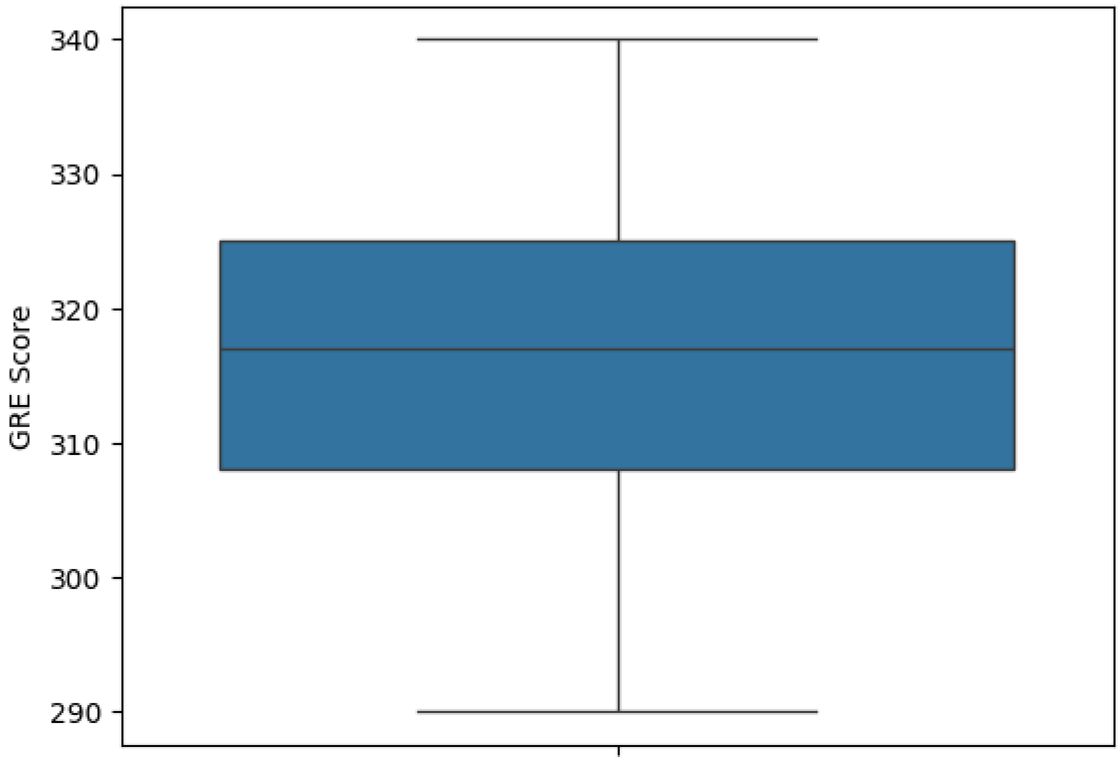
- All the distributions (CGPA,GRE Scores,TOEFL Scores) follows approximately normal distribution.
- Average CGPA is between 8.5-9.0
- Average TOEFL Score is between 105-110
- Average GRE Score is between 310-320
- Average Chance of Admit is between 0.7-0.8 and the distibution is left skewed

Checking for Outliers in the Numerical Data

In [223…
```
sns.boxplot(data = df, y = 'TOEFL Score')
plt.show()
```

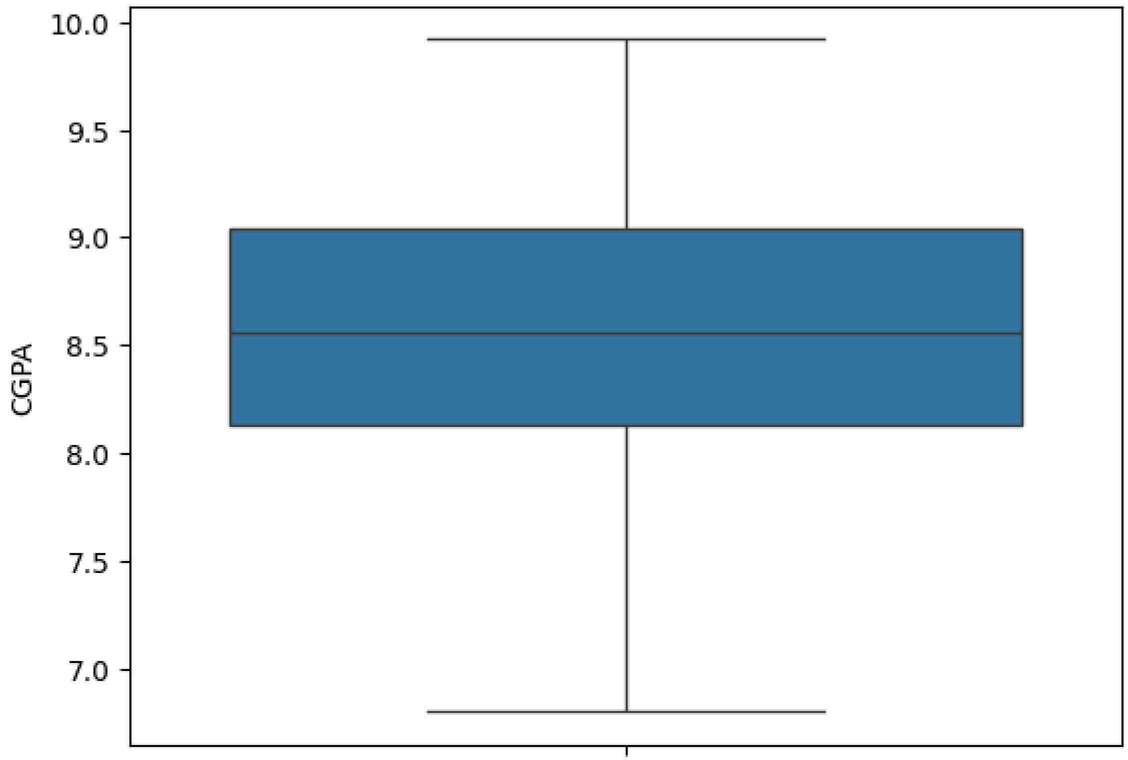


In [224…
```
sns.boxplot(data = df, y = 'GRE Score')
plt.show()
```



In [225…

```
sns.boxplot(data = df, y = 'CGPA')
plt.show()
```



Observation:

- From the above three box plots we can observe that there are no any outliers in TOEFL Score, GRE Score, CGPA

Understanding the frequency of categorical data using pie chart and bar plots

---

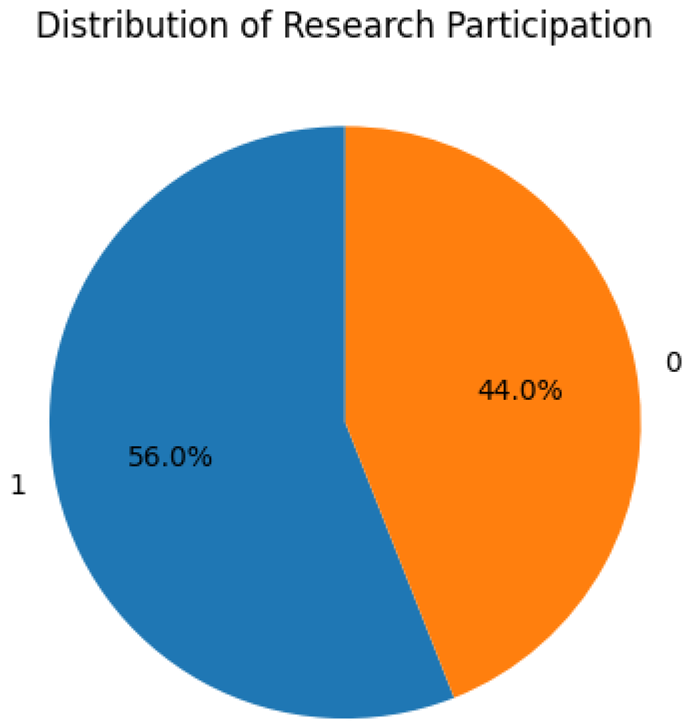
Although the columns - SOP, LOR, University Rating and Research have data types of float64 / int64, they actually represent categorical values

---

Research Participation

In [226…

```
research_counts = df['Research'].value_counts()
plt.pie(research_counts, labels=research_counts.index, autopct='%1.1f%%', startangle = 90)
plt.title("Distribution of Research Participation")
plt.show()
```

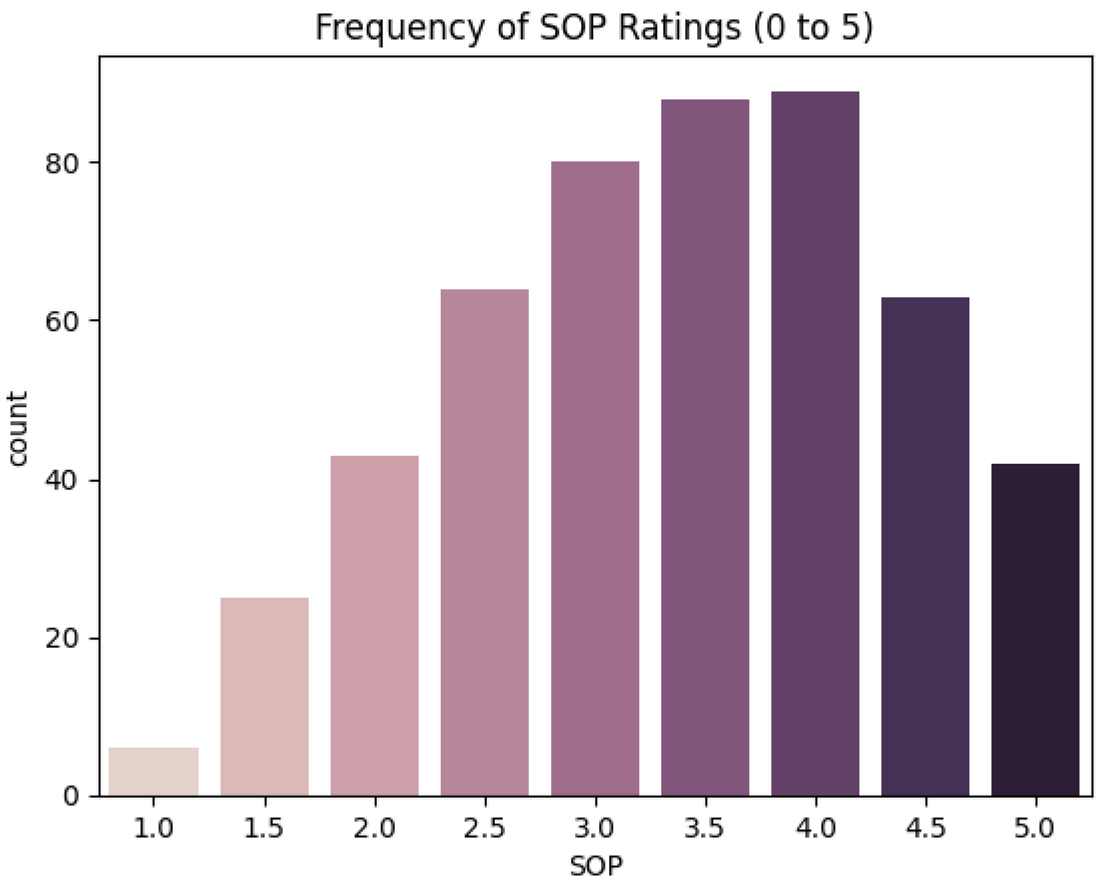


Observation:

- Students with research experience are slightly higher(56%) than stuednts with no experience (44%).

Strength of SOP (statement of purpose)

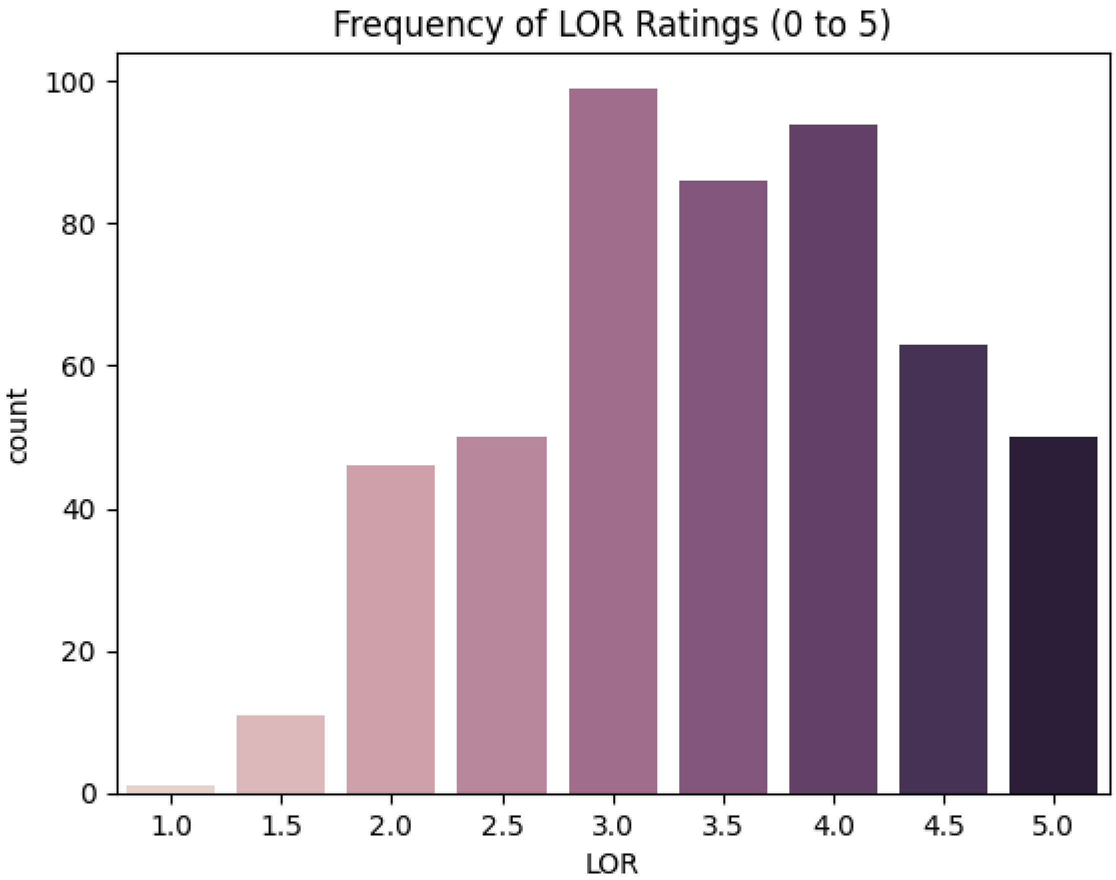
In [227…
```
sns.countplot(data = df , x = 'SOP',hue = 'SOP',legend = False)
plt.title("Frequency of SOP Ratings (0 to 5)")
plt.show()
```



Strength of Letter of Recommendation

In [228…
```
df.rename(columns = {'LOR ':'LOR','Chance of Admit ':'Chance of Admit'},inplace =True)
sns.countplot(data = df , x = 'LOR',hue = 'LOR',legend = False)
```

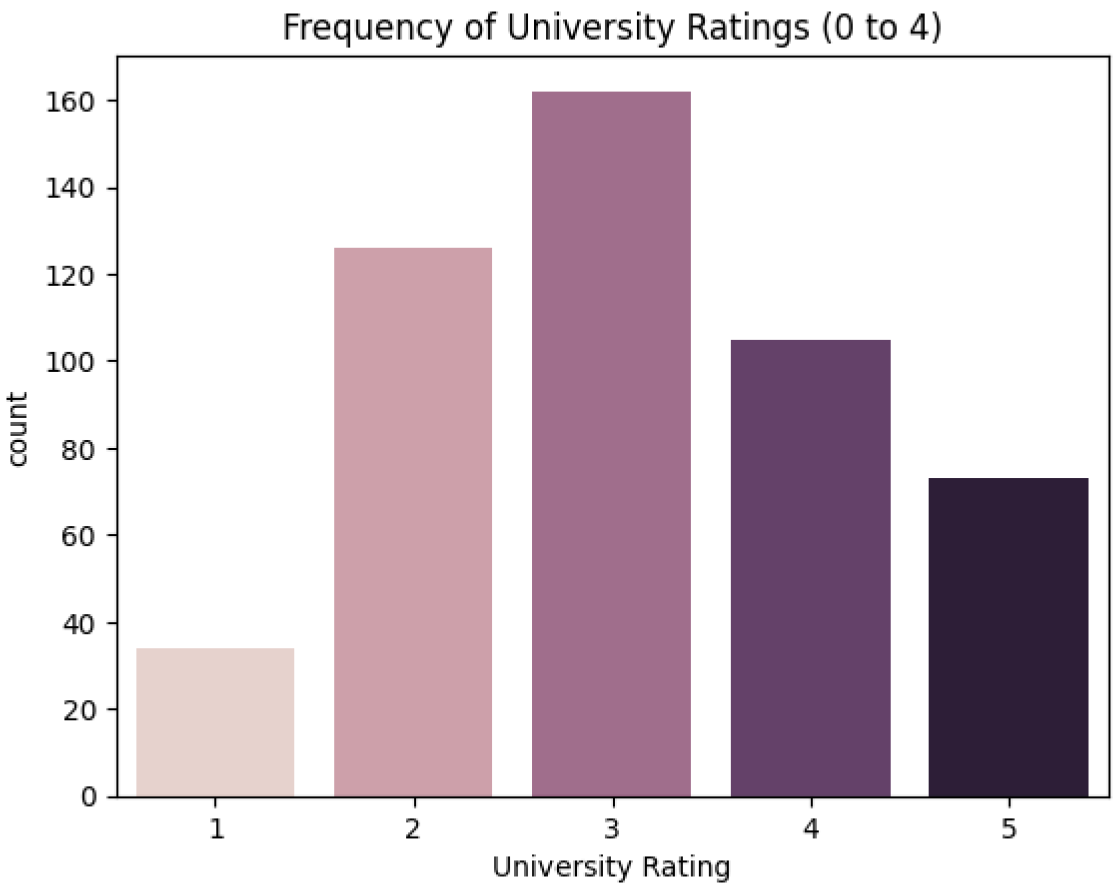
```
plt.title("Frequency of LOR Ratings (0 to 5)")
plt.show()
```



Strength of University Rating

In [229…

```
sns.countplot(data = df , x = 'University Rating',hue = 'University Rating',legend = False)
plt.title("Frequency of University Ratings (0 to 4)")
plt.show()
```

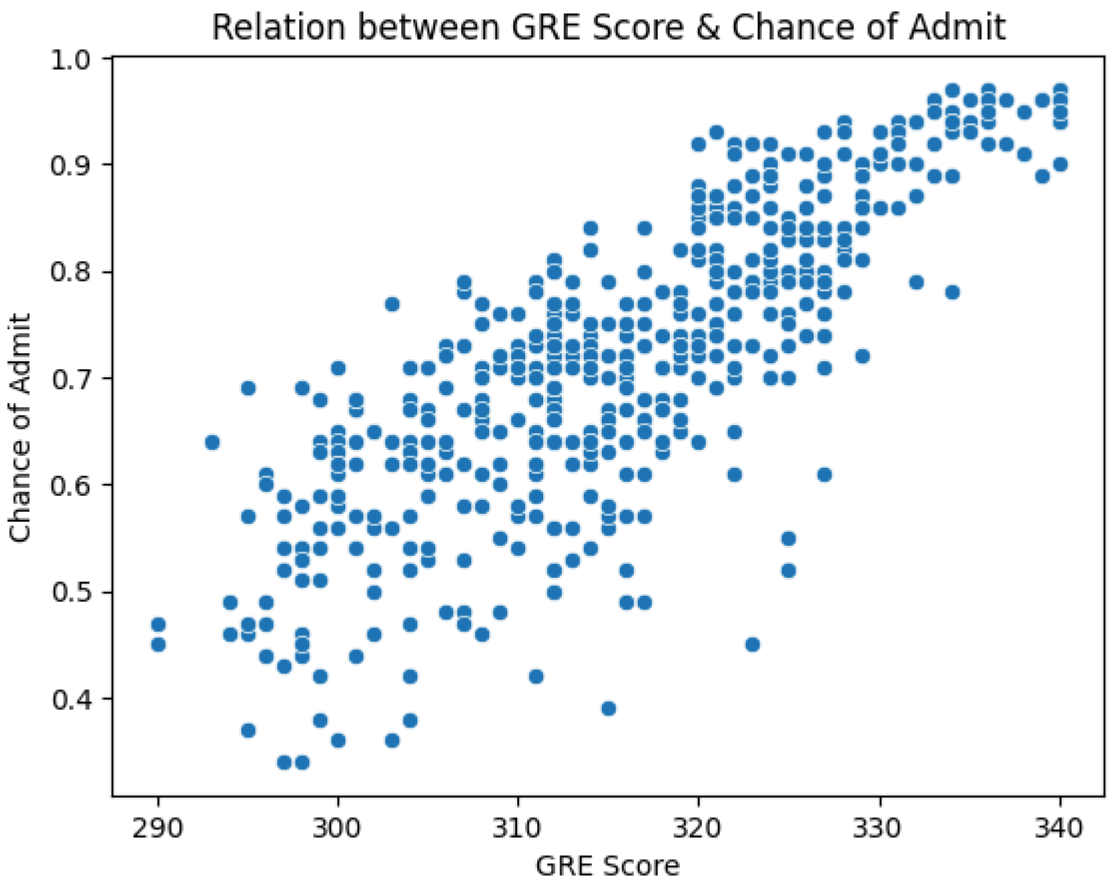


Relationship between two variables using Scatter plots

GRE VS Chance of Admit

In [230…

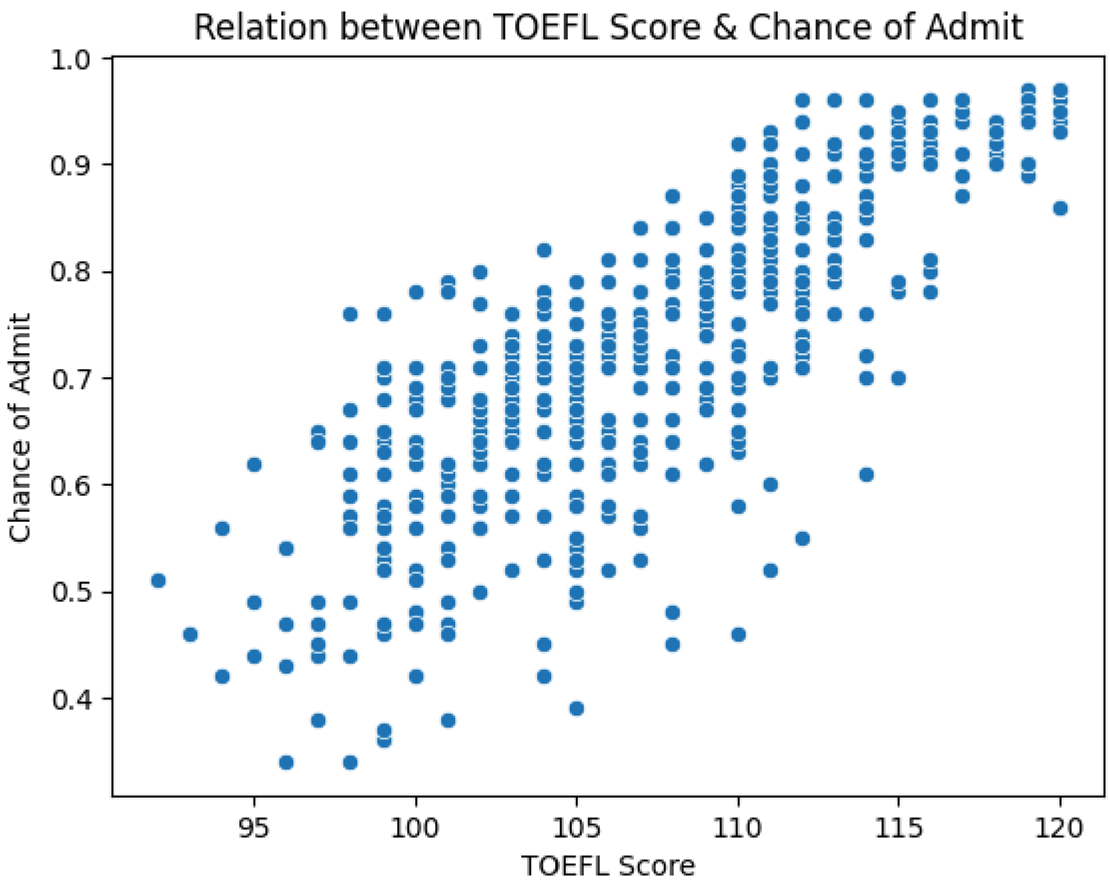
```
sns.scatterplot(x='GRE Score', y='Chance of Admit', data=df)
plt.title("Relation between GRE Score & Chance of Admit")
plt.show()
```



TOEFL Score VS Chance of Admit

In [231…

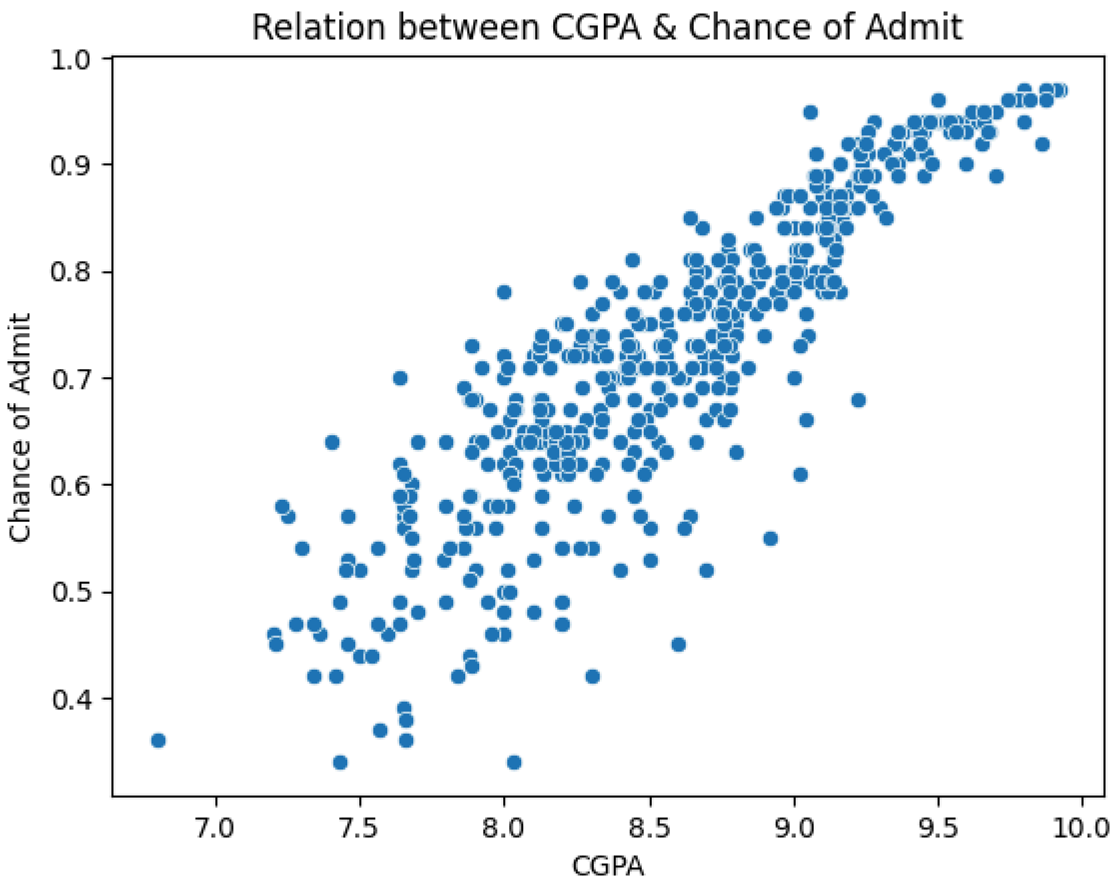
```
sns.scatterplot(x='TOEFL Score', y='Chance of Admit', data=df)
plt.title("Relation between TOEFL Score & Chance of Admit")
plt.show()
```



CGPA VS Chance of Admit

In [232…

```
sns.scatterplot(x='CGPA', y='Chance of Admit', data=df)
plt.title("Relation between CGPA & Chance of Admit")
plt.show()
```



Observation:

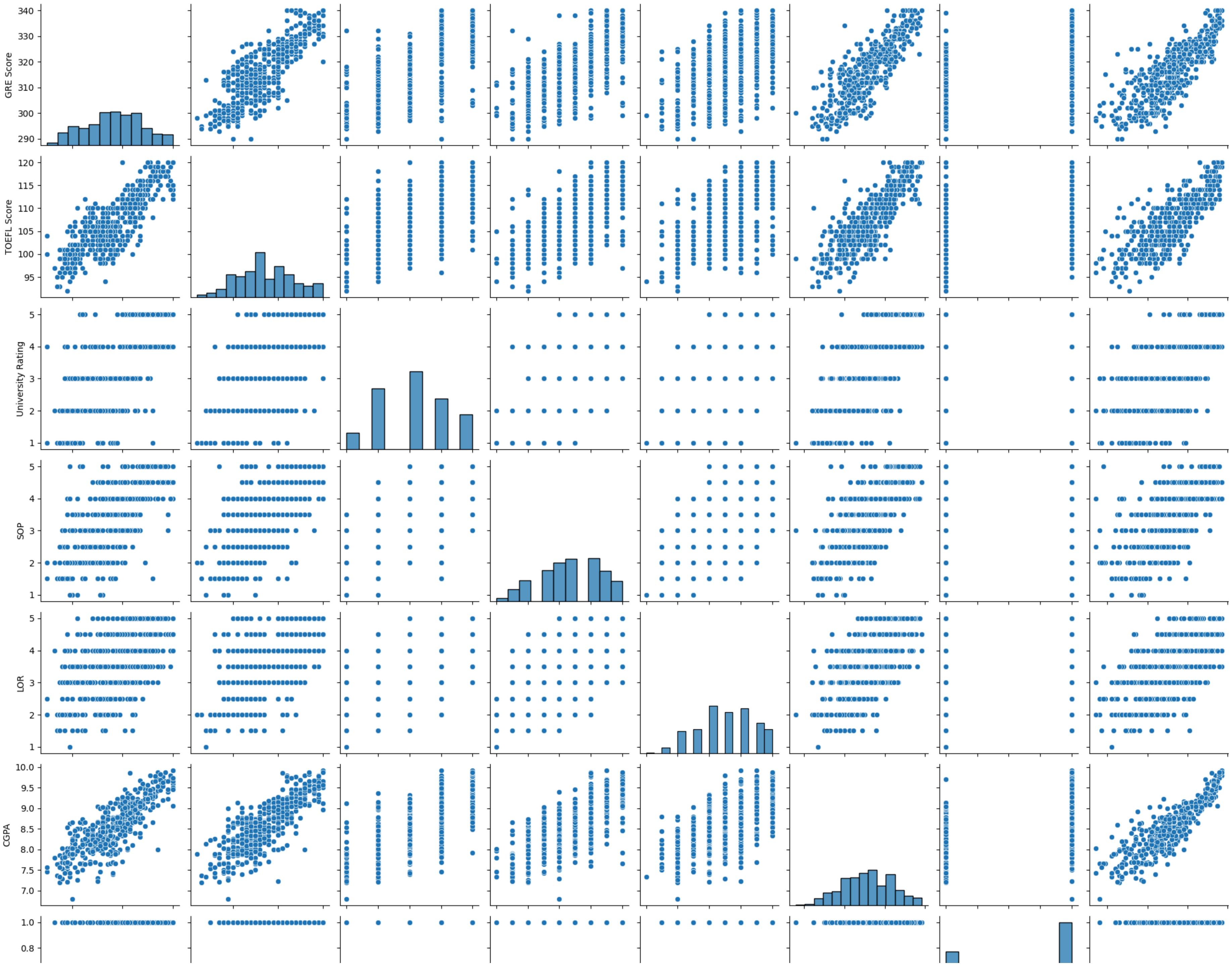
- TOEFL Score, CGPA, and GRE Score exhibit a strong positive correlation with the Chance of Admit.

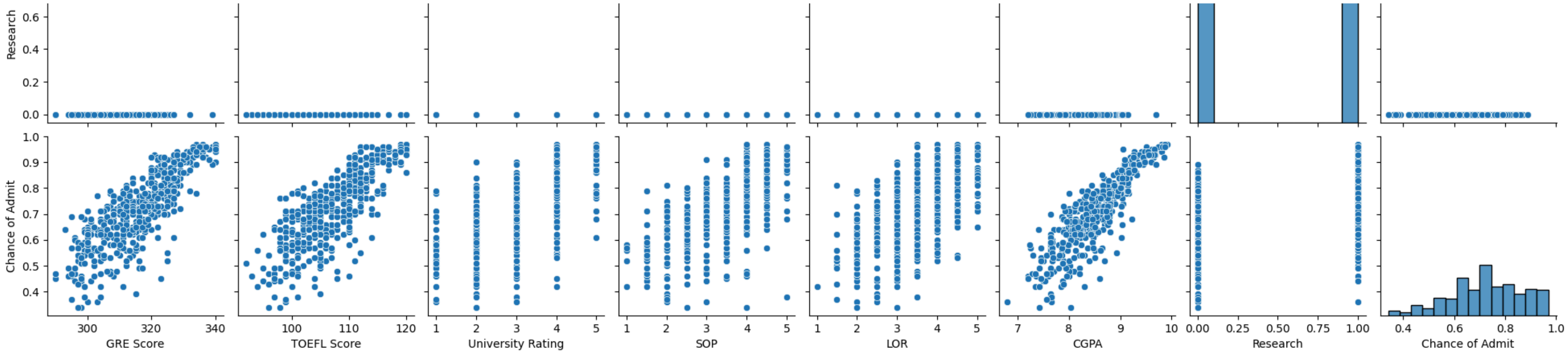
Understanding Relationship between each and every feature using Pair plots

In [233…

```
sns.pairplot(df)
```

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


GRE Score
340
330
320
310
300
290
TOEFL Score
120
115
110
105
100
95
University Rating
5
4
3
2
1
SOP
5
4
3
2
1
LOR
5
4
3
2
1
CGPA
10.0
9.5
9.0
8.5
8.0
7.5
7.0
1.0
0.8



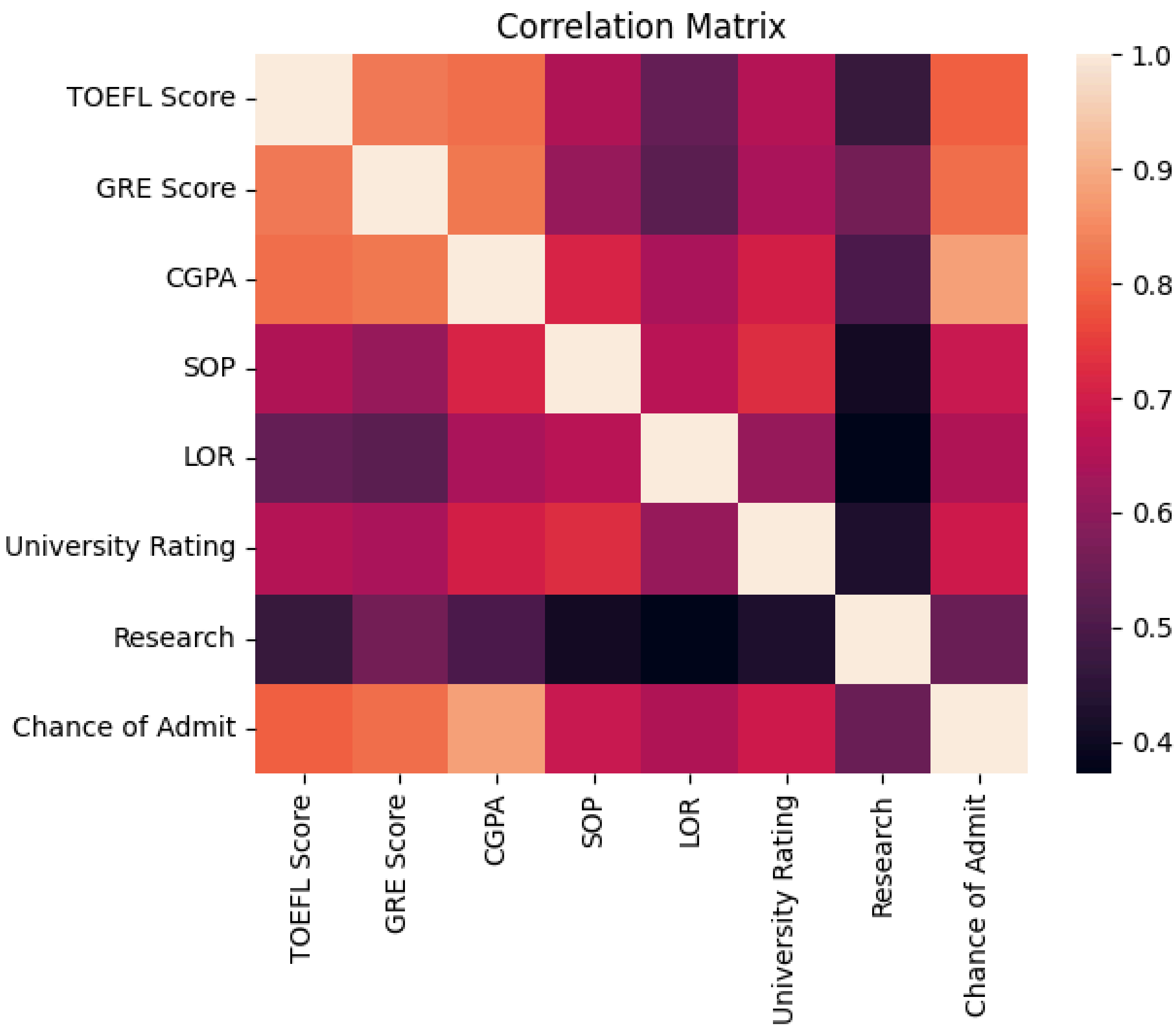
Observations:

- Students with higher GRE scores tend to have higher TOEFL scores and CGPAs, and these students also have a higher Chance of Admit.
- LOR (Letter of Recommendation) and University Rating don't show as strong of a correlation with Chance of Admit compared to GRE, TOEFL, and CGPA. This suggests that while they may have some impact, it is not as direct or linear as other features.
- There's a weak positive trend between University Rating and CGPA, indicating that students from higher-rated universities tend to have slightly higher CGPAs, but this is not a very strong relationship.

Correlation Matrix

In [234…

```
corr_matrix = df[['TOEFL Score','GRE Score','CGPA','SOP','LOR','University Rating','Research','Chance of Admit']].corr
sns.heatmap(corr_matrix)
plt.title('Correlation Matrix')
plt.show()
```



Observation:

- SOP, LOR, University rating have little impact on chance of admit but not as strong as GRE Score, TOEFL Score and CGPA.
- Whereas research participation has least correlation with chance of admit.

Comment on the range of attributes.

In [235…
```
range_of_GRE_Score = df['GRE Score'].min(),df['GRE Score'].max()
print("GRE_Scores ranges from : ",range_of_GRE_Score)
```

```
GRE_Scores ranges from :  (290, 340)
```

In [236…
```
range_of_TOEFL_Score = df['TOEFL Score'].min(),df['TOEFL Score'].max()
print("TOEFL_Scores ranges from : ",range_of_TOEFL_Score)
```

```
TOEFL_Scores ranges from :  (92, 120)
```

In [237…
```
range_of_CGPA = df['CGPA'].min(),df['CGPA'].max()
print("CGPA ranges from : ",range_of_CGPA)
```

```
CGPA ranges from :  (6.8, 9.92)
```

In [238…
```
range_of_University_Rating = df['University Rating'].min(),df['University Rating'].max()
print("University_Rating ranges from : ",range_of_University_Rating)
```

```
University_Rating ranges from :  (1, 5)
```

In [239…
```
range_of_SOP_ratings = df['SOP'].min(),df['SOP'].max()
print("SOP_ratings ranges from : ",range_of_SOP_ratings)
```

```
SOP_ratings ranges from :  (1.0, 5.0)
```

In [240…
```
range_of_LOR_ratings = df['LOR'].min(),df['LOR'].max()
print("LOR_ratings ranges from : ",range_of_LOR_ratings)
```

```
LOR_ratings ranges from :  (1.0, 5.0)
```

Statistical Summary

In [241…
```
df.describe()
```

Out[241]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

Data Preprocessing

Columns such as University Rating, Research, SOP, and LOR are categorical data but they are already represented using numerical data.Therefore, no encoding is necessary for these variables.

Separating features and target

In [242…
```
x = df.drop(columns = ['Chance of Admit']) #---  Features
y = df[['Chance of Admit']] #--- Target Variable
```

In [243…
```
x.head()
```

Out[243]:

	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

In [244…

```
y.head()
```

Out[244]:

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

Performing Test-Train Split

In [245…

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=54)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[245]:

```
((400, 7), (400, 1), (100, 7), (100, 1))
```

In [246…

```
x_train.head()
```

Out[246]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
398	312	103	3	3.5	4.0	8.78	0
146	315	105	3	2.0	2.5	8.48	0
108	331	116	5	5.0	5.0	9.38	1
266	312	105	2	2.0	2.5	8.45	0
462	307	105	4	3.0	3.0	7.94	0

In [247…

```
x_test.head()
```

Out[247]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
80	312	105	3	2.0	3.0	8.02	1
411	313	94	2	2.5	1.5	8.13	0
77	301	99	2	3.0	2.0	8.22	0
126	323	113	3	4.0	3.0	9.32	1
495	332	108	5	4.5	4.0	9.02	1

Performing the feature scaling

In [248…

```
#Initialising object of class StandardScaler() for Standardisation
scaler = StandardScaler()
```

In [249…

```
# Fit the scaler on the training data
scaler.fit(x_train)
# Transform the training data
x_train_scaled = pd.DataFrame(scaler.transform(x_train),columns = x_train.columns)
# Transform the test data using the same scaler
x_test_scaled = pd.DataFrame(scaler.transform(x_test),columns = x_test.columns)
```

After feature Scaling

In [250…
```
x_train_scaled.head()
```

Out[250]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	-0.362690	-0.672091	-0.055273	0.148796	0.581497	0.376090	-1.111142
1	-0.094527	-0.340604	-0.055273	-1.339163	-1.060376	-0.124418	-1.111142
2	1.335674	1.482579	1.713466	1.636755	1.676078	1.377105	0.899975
3	-0.362690	-0.340604	-0.939642	-1.339163	-1.060376	-0.174469	-1.111142
4	-0.809628	-0.340604	0.829096	-0.347190	-0.513085	-1.025331	-1.111142

In [251…
```
x_test_scaled.head()
```

Out[251]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	-0.362690	-0.340604	-0.055273	-1.339163	-0.513085	-0.891863	0.899975
1	-0.273303	-2.163786	-0.939642	-0.843177	-2.154958	-0.708343	-1.111142
2	-1.345954	-1.335067	-0.939642	-0.347190	-1.607667	-0.558191	-1.111142
3	0.620573	0.985347	-0.055273	0.644782	-0.513085	1.277003	0.899975
4	1.425062	0.156628	1.713466	1.140769	0.581497	0.776496	0.899975

Linear Regression Model

In [252…
```
lr_model = LinearRegression()
lr_model.fit(x_train_scaled,y_train)
```

Out[252]:
```
▾ LinearRegression
LinearRegression()
```

In [253…

```
# weights of each independent variables
lr_model.coef_
```

Out[253]:

```
array([[ 0.01439075,  0.0145681 ,  0.00672409, -0.00037411,  0.01473891,
         0.07819229,  0.01512192]])
```

In [254…

```
# bias of the model
lr_model.intercept_
```

Out[254]:

```
array([0.717475])
```

In [255…

```
# Predicting values for the training and test data
y_pred_train = lr_model.predict(x_train_scaled)
y_pred_test = lr_model.predict(x_test_scaled)
```

In [256…

```
# Model Coefficients
for feature,weight in zip(x_train_scaled.columns, lr_model.coef_[0]):
  print(f"Weight of {feature}: {np.round(weight,5)}")
```

```
Weight of GRE Score: 0.01439
Weight of TOEFL Score: 0.01457
Weight of University Rating: 0.00672
Weight of SOP: -0.00037
Weight of LOR: 0.01474
Weight of CGPA: 0.07819
Weight of Research: 0.01512
```

Observation:

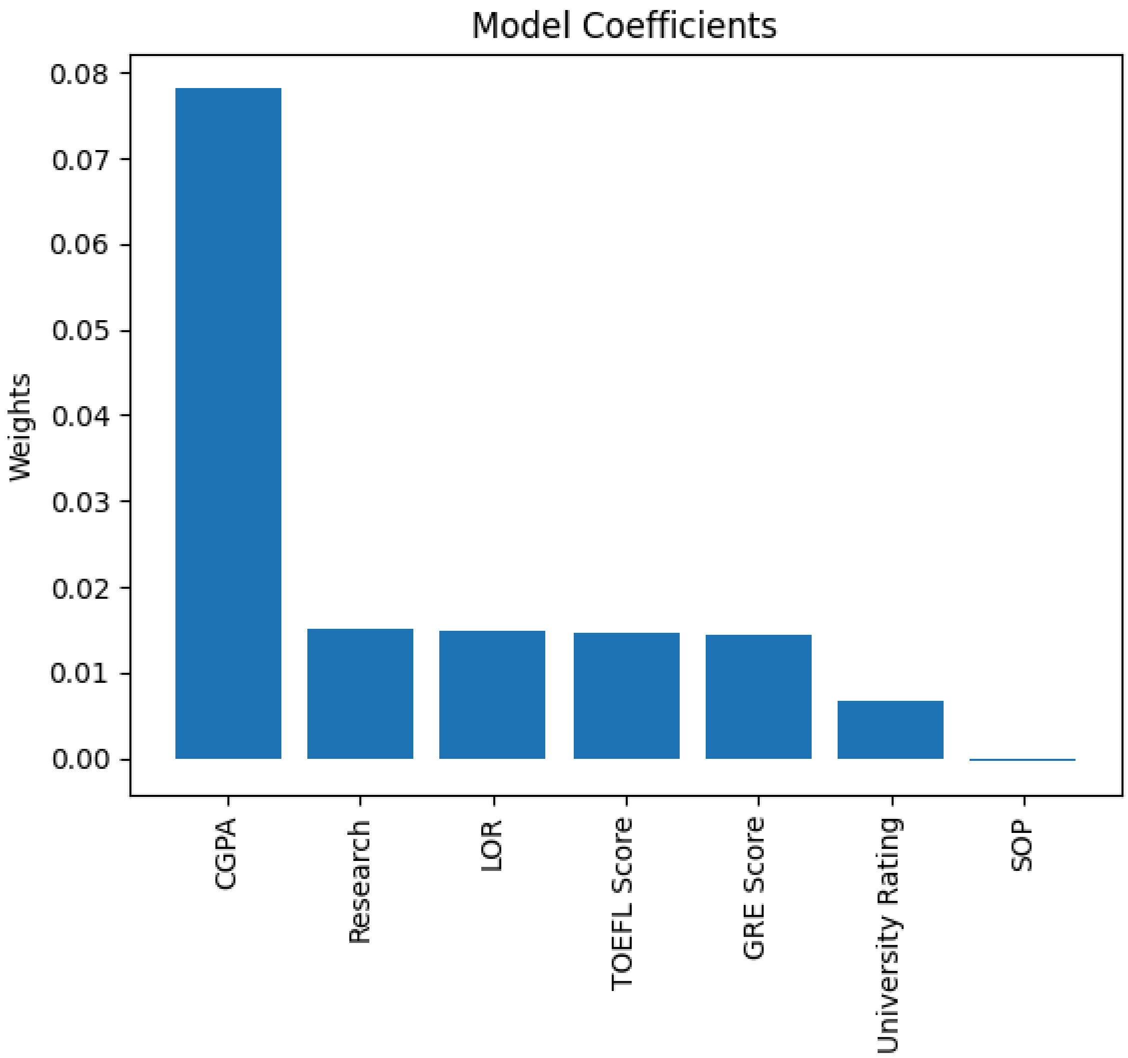
- As per the weights CGPA carries highest weight in predicting the chance of admit
- Followed by Research , LOR, TOEFL Score, GRE Score
- SOP and University Rating have very negligible weight in predicting the chance of admit

In [257…

```
model_weights=list(zip(x_train_scaled.columns, lr_model.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();
```



Using Linear Regression from Statsmodel library.

In [258…

```python
y_train = np.array(y_train)
```

In [259…

```python
x_sm = sm.add_constant(x_train_scaled)  # Statmodels default is without intercept, to add intercept we need to add cor

model = sm.OLS(y_train, x_sm)
results = model.fit()

# Print the summary statistics of the model
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.816
Model:                            OLS   Adj. R-squared:                  0.813
Method:                 Least Squares   F-statistic:                     248.9
Date:                Wed, 11 Sep 2024   Prob (F-statistic):          5.52e-140
Time:                        10:31:00   Log-Likelihood:                 558.73
No. Observations:                 400   AIC:                            -1101.
Df Residuals:                     392   BIC:                            -1070.
Df Model:                           7
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.7175      0.003    237.310      0.000       0.712       0.723
GRE Score             0.0144      0.007      2.178      0.030       0.001       0.027
TOEFL Score           0.0146      0.006      2.394      0.017       0.003       0.027
University Rating     0.0067      0.005      1.394      0.164      -0.003       0.016
SOP                  -0.0004      0.005     -0.072      0.942      -0.011       0.010
LOR                   0.0147      0.004      3.441      0.001       0.006       0.023
CGPA                  0.0782      0.007     11.490      0.000       0.065       0.092
Research              0.0151      0.004      4.150      0.000       0.008       0.022
==============================================================================
Omnibus:                       97.498   Durbin-Watson:                   2.061
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              230.354
Skew:                          -1.216   Prob(JB):                     9.53e-51
Kurtosis:                       5.812   Cond. No.                         5.87
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Observation:

- The model appears to fit the data well with a high $R2$ value.
- Significant predictors include GRE Score, TOEFL Score, LOR, CGPA, and Research.
- Some predictors (University Rating and SOP) are not statistically significant.
- We can drop SOP and University Rating columns as they are not statistically significnat in contributing in predicting the chance of admit.

In [260…

```
# dropping those columns with p > 0.05 from the x_train_scaled data
x_train_scaled.drop(columns = results.params.index[results.pvalues.values > 0.05],inplace = True)
```

In [261…

```
x_train_scaled
```

Out[261]:

	GRE Score	TOEFL Score	LOR	CGPA	Research
0	-0.362690	-0.672091	0.581497	0.376090	-1.111142
1	-0.094527	-0.340604	-1.060376	-0.124418	-1.111142
2	1.335674	1.482579	1.676078	1.377105	0.899975
3	-0.362690	-0.340604	-1.060376	-0.174469	-1.111142
4	-0.809628	-0.340604	-0.513085	-1.025331	-1.111142
...	...	...	...	...	...
395	1.603837	1.979811	1.128787	1.910979	0.899975
396	-1.524729	-1.832299	-1.607667	-1.158800	-1.111142
397	-1.256566	-1.003579	0.034206	-0.991964	-1.111142
398	0.888736	1.482579	0.581497	0.976699	0.899975
399	0.441798	0.322372	0.581497	0.209254	0.899975

400 rows × 5 columns

```
In [262…  # dropping those columns with p > 0.05 from the x_test_scaled data
x_test_scaled.drop(columns = results.params.index[results.pvalues.values > 0.05],inplace = True)
```

```
In [263…  x_test_scaled
```

Out[263]:

	GRE Score	TOEFL Score	LOR	CGPA	Research
0	-0.362690	-0.340604	-0.513085	-0.891863	0.899975
1	-0.273303	-2.163786	-2.154958	-0.708343	-1.111142
2	-1.345954	-1.335067	-1.607667	-0.558191	-1.111142
3	0.620573	0.985347	-0.513085	1.277003	0.899975
4	1.425062	0.156628	0.581497	0.776496	0.899975
...	...	...	...	...	...
95	0.978124	-0.174860	1.128787	0.326039	0.899975
96	1.425062	1.648323	0.581497	0.909964	-1.111142
97	-0.273303	-1.003579	-0.513085	-0.858495	-1.111142
98	-1.077791	-0.340604	-2.154958	-1.759409	-1.111142
99	0.620573	0.488116	1.676078	0.709761	0.899975

100 rows × 5 columns

Retraining the model after removing the SOP and University Ratings column as they have probability > 0.05

```
In [264…  x_sm = sm.add_constant(x_train_scaled)  # Statmodels default is without intercept, to add intercept we need to add cor

model = sm.OLS(y_train, x_sm)
results = model.fit()

# Print the summary statistics of the model
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.815
Model:                            OLS   Adj. R-squared:                  0.813
Method:                 Least Squares   F-statistic:                     347.9
Date:                Wed, 11 Sep 2024   Prob (F-statistic):          4.71e-142
Time:                        10:31:00   Log-Likelihood:                 557.61
No. Observations:                 400   AIC:                            -1103.
Df Residuals:                     394   BIC:                            -1079.
Df Model:                           5
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const           0.7175      0.003    237.252      0.000       0.712       0.723
GRE Score       0.0145      0.007      2.201      0.028       0.002       0.028
TOEFL Score     0.0159      0.006      2.644      0.009       0.004       0.028
LOR             0.0161      0.004      4.054      0.000       0.008       0.024
CGPA            0.0804      0.006     12.396      0.000       0.068       0.093
Research        0.0154      0.004      4.236      0.000       0.008       0.023
==============================================================================
Omnibus:                       95.969   Durbin-Watson:                   2.058
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              223.893
Skew:                          -1.203   Prob(JB):                     2.41e-49
Kurtosis:                       5.765   Cond. No.                         4.94
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Testing the Assumptions of Linear Regression

Multicollinearity check by VIF score

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.

In [265…

```
vif = pd.DataFrame()
x_t = pd.DataFrame(x_train_scaled, columns=x_train_scaled.columns)
vif['Features'] = x_t.columns
vif['VIF'] = [variance_inflation_factor(x_t.values, i) for i in range(x_t.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[265]:

	Features	VIF
0	GRE Score	4.77
3	CGPA	4.60
1	TOEFL Score	3.95
2	LOR	1.73
4	Research	1.45

Observation:

- Since I have set my VIF threshold to 5, all the VIF values for the features are below this threshold, indicating that multicollinearity is not a significant issue among the predictors. Thus, the features are not highly collinear with each other

Mean of residuals should be close to zero.

- The mean of residuals represents the average of residual values in a regression model.
- Residuals are the discrepancies or errors between the observed values and the values predicted by the regression model.

- 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.
- The mean of residuals being close to zero indicates that, on average, the predictions made by the linear regression model are accurate, with an equal balance of overestimations and underestimations. This is a desirable characteristic of a well-fitted regression model.

In [266…

```
residual_train = y_train - y_pred_train
print("Mean of residual_train : ", residual_train.mean())
```

```
Mean of residual_train :  3.3306690738754695e-17
```

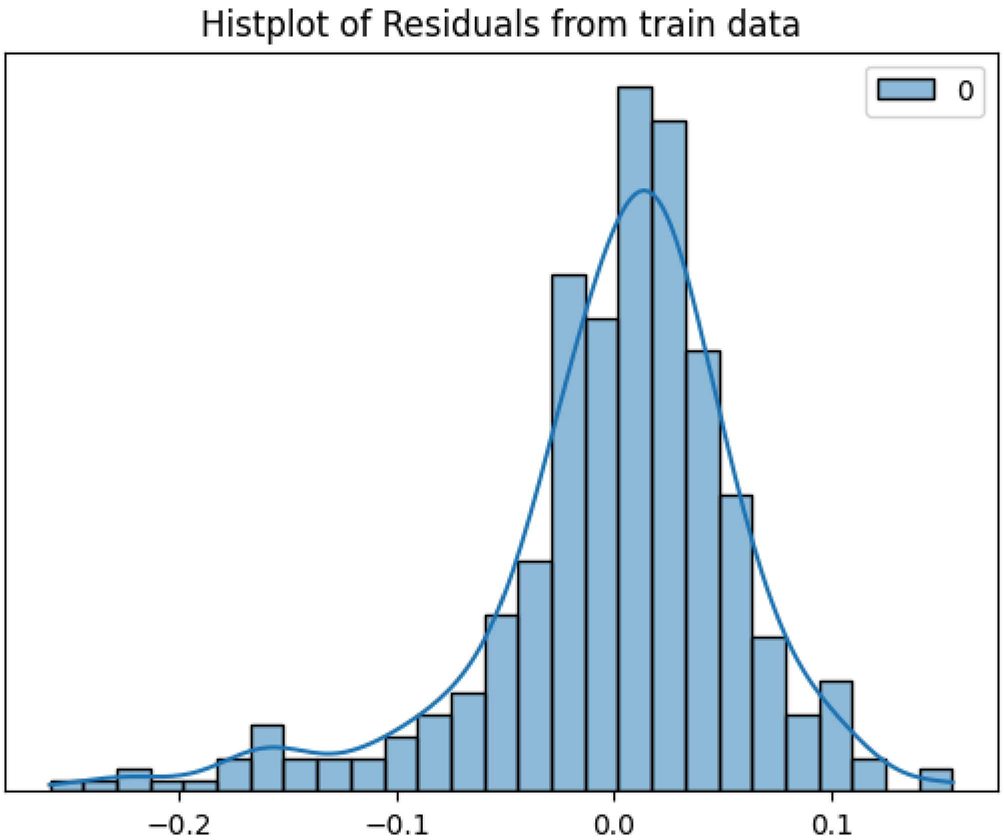
In [267…

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

```
Mean of Residuals:  -0.0011139976940071944
```

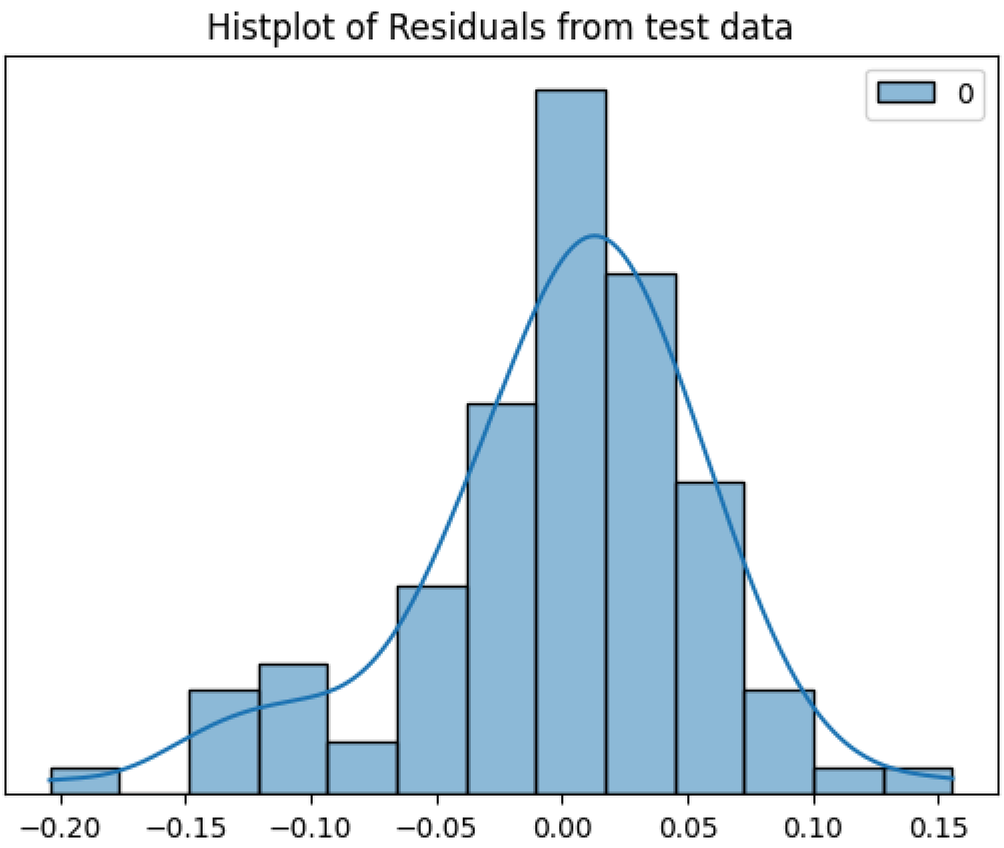
In [268…

```
sns.histplot(residual_train, kde= True)
plt.title('Histplot of Residuals from train data')
plt.ylabel("")
plt.yticks([])
plt.show()
```



In [269…

```
sns.histplot(residuals, kde= True)
plt.title('Histplot of Residuals from test data')
plt.ylabel("")
plt.yticks([])
plt.show()
```



Observation:

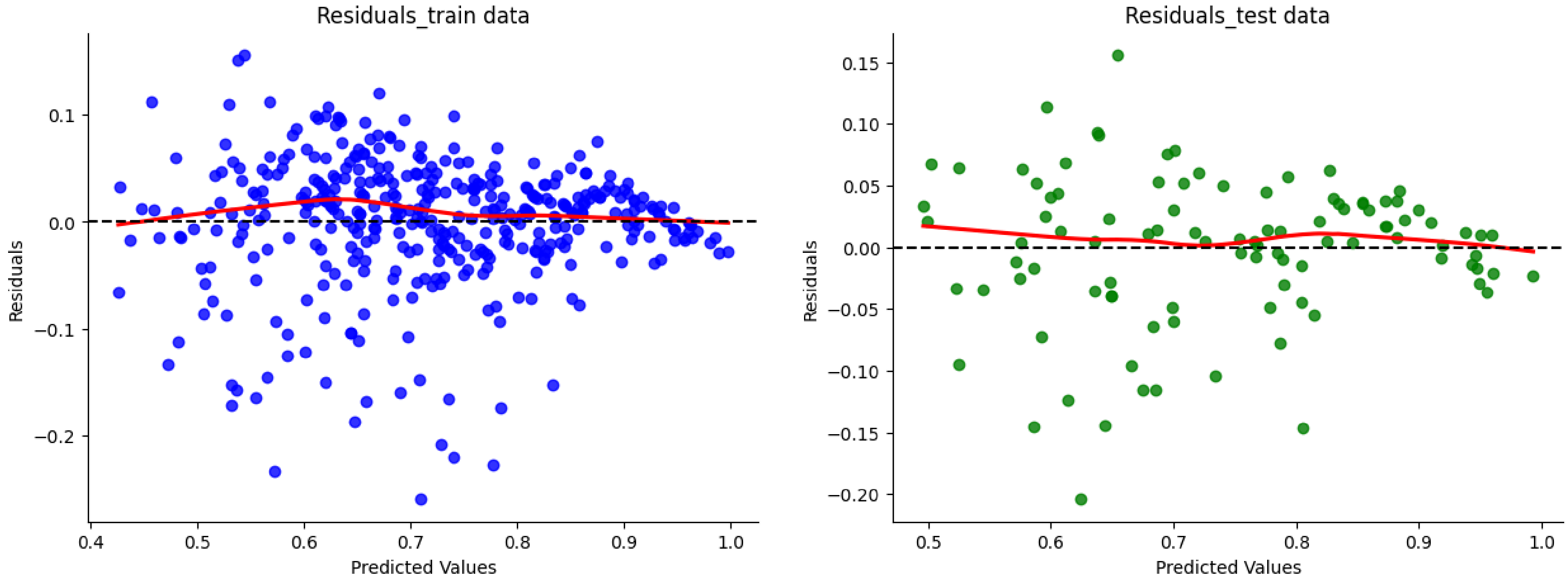
- The mean of residuals is close to zero for both the training and test datasets.
- This indicates that the model is well-calibrated and there is no systematic bias in the predictions.
- The residuals are evenly distributed around zero, suggesting that the model is performing well and not consistently underestimating or overestimating the target variable across both datasets.

Linear relationship between independent & dependent 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.
- When we talk about "no pattern in the residual plot" in the context of linearity, we are referring to the plot of the residuals (the differences between the observed and predicted values of the dependent variable) against the predicted values or the independent variables.
- Ideally, in a linear regression model, the residuals should be randomly scattered around zero, without any clear patterns or trends. This indicates that the model captures the linear relationships well and the assumption of linearity is met.

In [270…

```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.title('Residuals_train data',fontsize=12)
sns.regplot(x=y_pred_train, y=residual_train, lowess=True, color='b',line_kws={'color': 'red'})
plt.axhline(y=0, color='k', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.subplot(122)
plt.title('Residuals_test data',fontsize=12)
sns.regplot(x=y_pred_test, y=residuals, lowess=True,color='g' ,line_kws={'color': 'red'})
plt.axhline(y=0, color='k', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
sns.despine()
plt.show()
```



Observations:

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

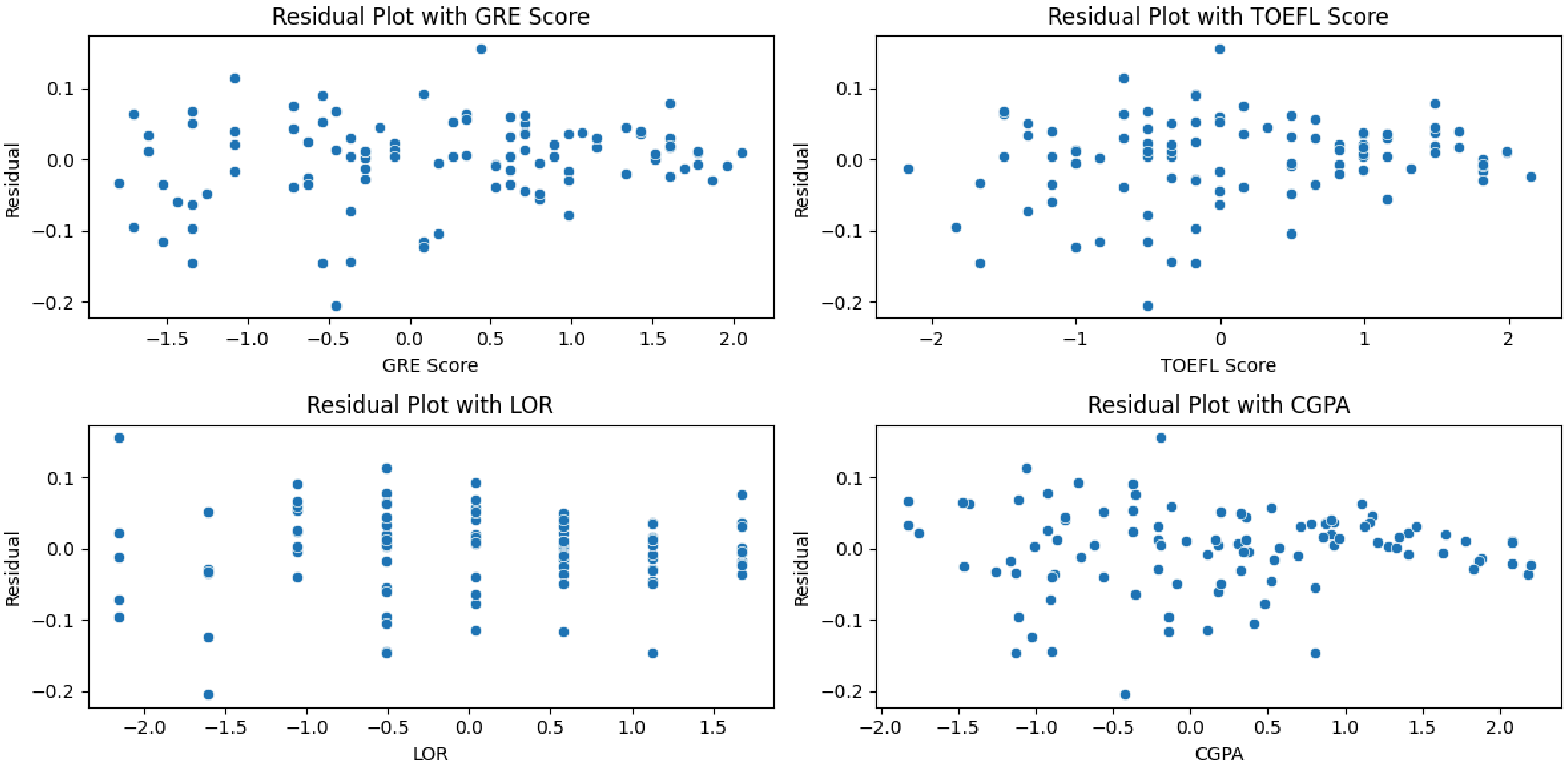
Test for Homoscedasticity

- 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 [271…

```
# Scatterplot of residuals with each independent variable to check for Homoscedasticity
plt.figure(figsize=(12,6))
i=1
for col in x_test_scaled.columns[:-1]:
  ax = plt.subplot(2,2,i)
  sns.scatterplot(x=x_test_scaled[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();
```



Observation:

- 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.

In [272…

```
# Add constant to the features (intercept)
x_sm = sm.add_constant(x_train_scaled)

# Fit the OLS regression model
model = sm.OLS(y_train, x_sm)
results = model.fit()

# Perform the Goldfeld-Quandt test
gq_test = het_goldfeldquandt(results.resid, x_sm)

# The Goldfeld-Quandt test returns three values:
# 1. F-statistic
# 2. p-value
# 3. 'two-sided' or 'increasing' or 'decreasing'
f_stat, p_value, alternative = gq_test

print("Goldfeld-Quandt F-statistic:", f_stat)
print("Goldfeld-Quandt p-value:", p_value)
print("Test alternative hypothesis:", alternative)
```

```
Goldfeld-Quandt F-statistic: 1.0549083552206973
Goldfeld-Quandt p-value: 0.35503510350188705
Test alternative hypothesis: increasing
```

Observations:

- The p-value (0.3550) is much higher than typical significance levels (e.g., 0.05).
- We fail to reject the null hypothesis.
- There is no significant evidence of increasing variance in the residuals. Heteroscedasticity is not a major concern in the model.

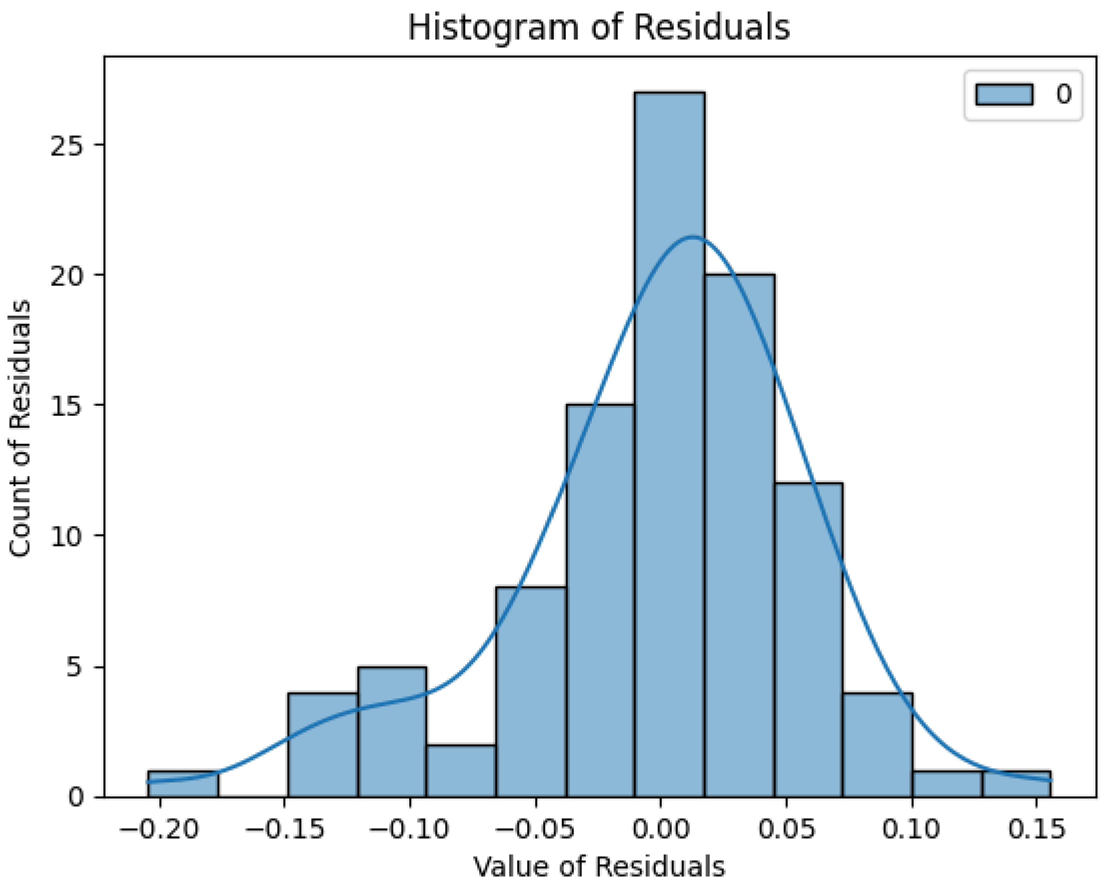
Normality of residuals

Normality of residuals refers to the assumption that the residuals (or errors) in a statistical model are normally distributed. Residuals are the differences between the observed values and the predicted values from the model.

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.

In [273…

```
sns.histplot(residuals, kde=True)
plt.title('Histogram of Residuals')
plt.xlabel('Value of Residuals')
plt.ylabel('Count of Residuals')
plt.show();
```



Shapiro wilk's test to check the normality of residuals

In [274…

```
# Assuming 'errors' is the array of residuals (errors = y_train - y_pred_train)
shapiro_stat,p_val = stats.shapiro(residuals)
alpha= 0.05
if p_val > alpha:
  print(f"Shapiro-Wilk Test p-value: {p_val} : Residuals are normally distributed" )
else:
  print(f"Shapiro-Wilk Test p-value: {p_val} : Residuals are not normally distributed")
```

```
Shapiro-Wilk Test p-value: 0.001853847562560806 : Residuals are not normally distributed
```

Evaluating the models performance

In [275…

```
# Evaluating the model using multiple loss functions
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}")
```

In [276…

```
#Metrics for test data
model_evaluation(y_test.values, y_pred_test,lr_model)
```

```
MAE: 0.04
RMSE: 0.06
R2 Score: 0.83
Adjusted R2: 0.82
```

In [277…

```
#Metrics for train data
model_evaluation(y_train, y_pred_train,lr_model)
```

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

Observation:

- Error Metrics: Both MAE and RMSE values are consistent across training and test data, indicating that the model's prediction errors are similar in both datasets. This suggests a well-calibrated model with a low level of prediction error.

- R2 Score: The R2 Score is 0.83 for the test data and 0.82 for the training data. This indicates that the model explains around 82-83% of the variance in the target variable, which is quite strong and suggests a good fit.

- Adjusted R2: The Adjusted R2 values are identical (0.82) for both datasets. This confirms that the model accounts for the number of predictors appropriately and indicates that the model is not overfitting.

- Model Performance: The consistency of metrics between training and test data, along with strong R2 and Adjusted R2 values, suggests that the model performs well and generalizes effectively to unseen data. There is no significant overfitting or underfitting observed.

Insights:

- Distribution of Chance of Admit is left-skewed, meaning most applicants have a higher likelihood of admission.
- Exam scores (CGPA, GRE, TOEFL) show a strong positive correlation with the Chance of Admit.
- Research experience and LOR positively influence admission chances.
- CGPA is the most significant predictor, while SOP and University Rating have lesser impact.
- The model explains about 82% of the variance in admission chances, indicating a good fit.
- High collinearity among predictors exists, but the model performs well.
- The model meets most Linear Regression assumptions, except for the normality of residuals.

Recommendations:

- Improve Key Metrics: Students should focus on boosting their GRE scores, CGPA, and the quality of their Letters of Recommendation (LOR) to increase their chances of admission.

- Add Diverse Data: Expand the data collected to include extracurricular activities, work experience, and personal statements for a more holistic view of applicants.