Jamboree Education - Linear Regression

Objective

The objective of conducting linear regression analysis for Jamboree is to identify the significant factors influencing graduate admissions, such as GRE scores, CGPA, and Letters of Recommendation (LOR), while exploring the interrelationships among these features and assessing any multicollinearity issues. By developing a predictive model to estimate admission probabilities based on the identified features, we aim to provide prospective students with valuable insights into their chances of admission. Additionally, we will evaluate the model's assumptions, particularly regarding the normality and homoscedasticity of residuals, to ensure its robustness. Ultimately, this analysis will empower Jamboree to enhance its advising strategies, helping students focus on improving their performance in critical areas for better admission outcomes.

Dataset

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

In [175]:
```
#importing packages for eda
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from scipy.stats import shapiro
warnings.filterwarnings('ignore')
```

In [174]:
```
df=pd.read_csv(r"C:\Users\varun\Desktop\jambotree.csv")
```

In [3]:
```
jambotree_df=df.copy()
```

Data Preprocessing

In [4]:
```
jambotree_df.columns
```

Out[4]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [5]:
```
jambotree_df.info()     #checking datatype and null values
```

```
<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
```

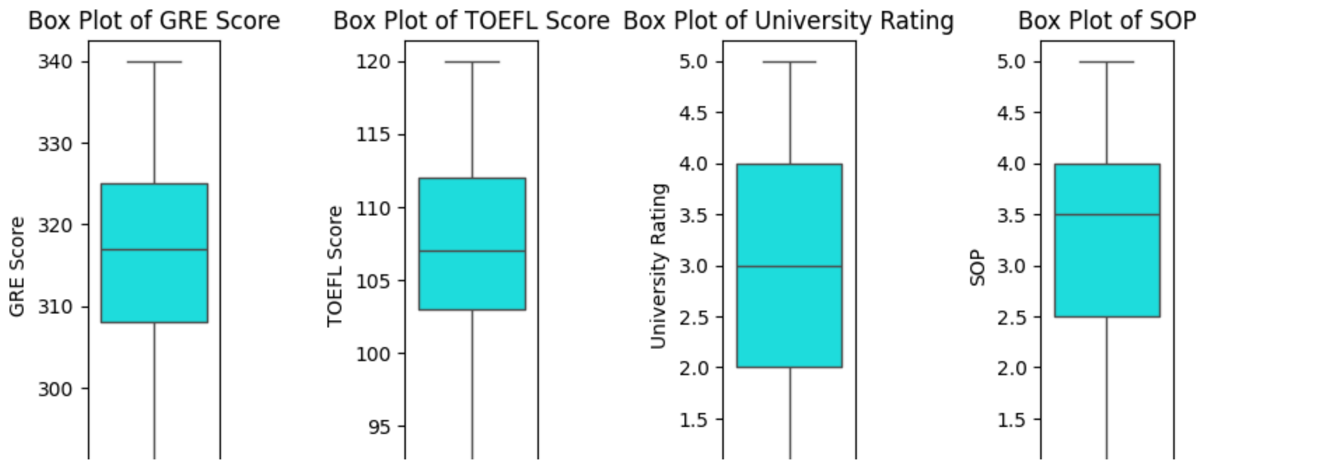
In [6]:
```
jambotree_df.drop(columns={'Serial No.'},inplace=True) # removing unwanted columns
```

In [7]:
```
jambotree_df.rename(columns={'Chance of Admit ':'Chance of Admit'},inplace=True)
jambotree_df.rename(columns={'LOR ':'LOR'},inplace=True)# Renaming the columns
```

In [8]:
```
jambotree_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
```

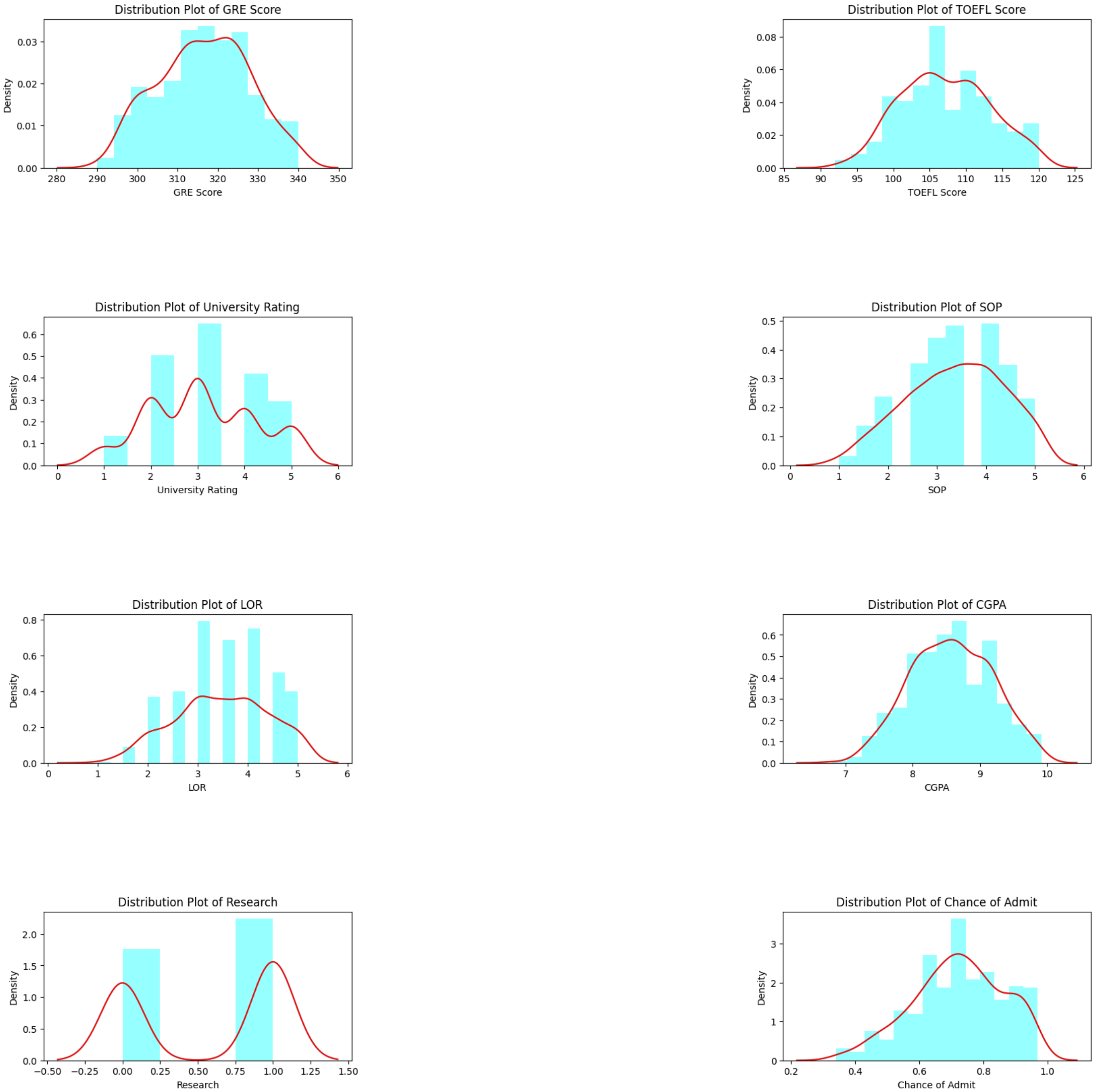
In [9]:
```
plt.figure(figsize=(10,10))
plt.subplots_adjust(wspace=1.4,hspace=0.4)
for i,val in enumerate(jambotree_df.columns):
    plt.subplot(2,4,i+1)
    sns.boxplot(jambotree_df[val],color='cyan')
    plt.title(f"Box Plot of {val}")
plt.show()
```



We can see that there is no problem with the outliers here so there is no need to do outlier handling

In [10]:

```
plt.figure(figsize=(20,20))
plt.subplots_adjust(wspace=1.4,hspace=1.0)
for i,val in enumerate(jambotree_df.columns):
    plt.subplot(4,2,i+1)
    sns.distplot(jambotree_df[val],color='cyan')
    sns.kdeplot(jambotree_df[val],color='red')
    plt.title(f"Distribution Plot of {val}")
plt.show()
```



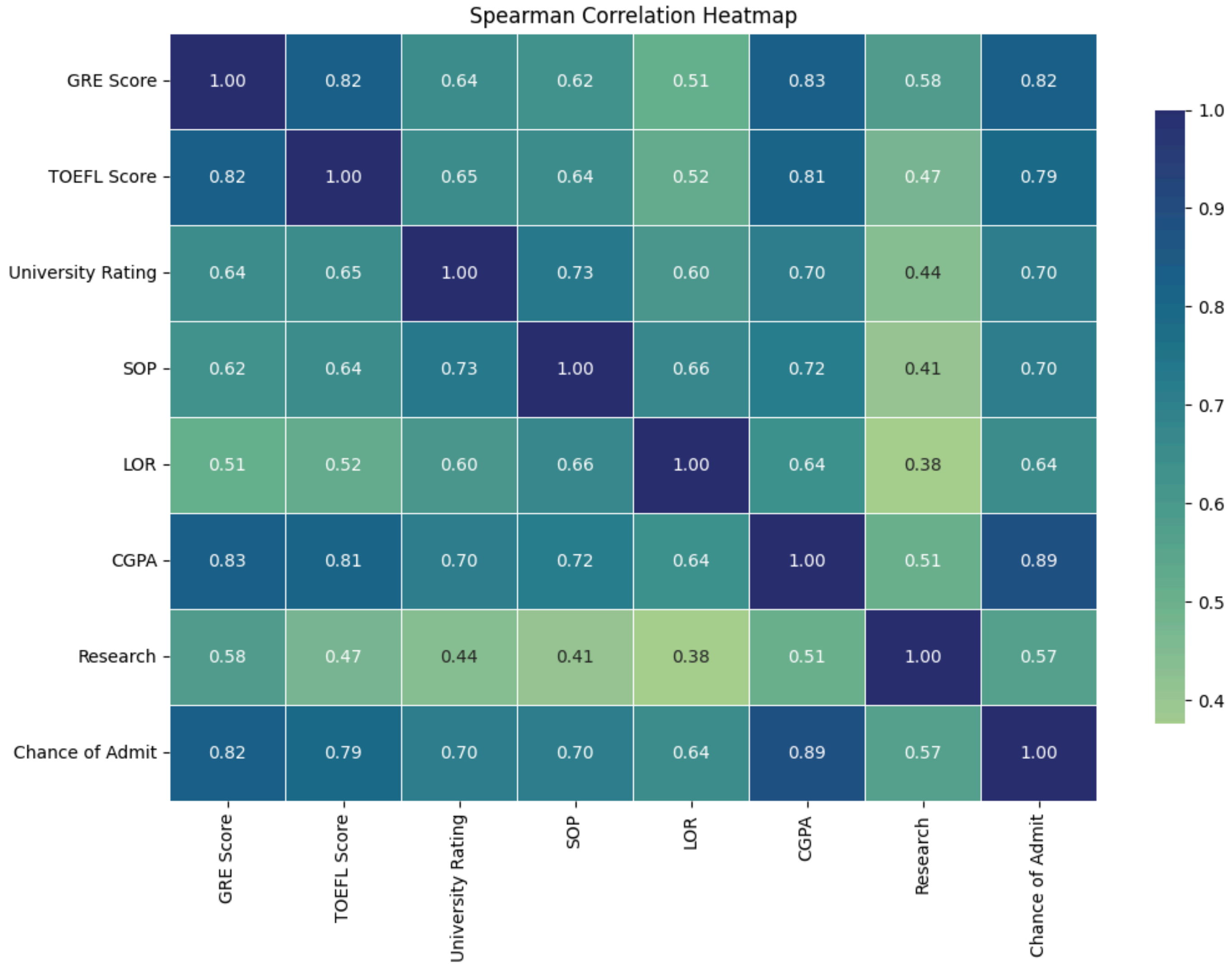
In [ ]:

```

```

EDA

In [11]:

```python
plt.figure(figsize=(12, 8))

# Create a heatmap for the Spearman correlation matrix
sns.heatmap(
    jambotree_df.corr(method='spearman'),
    annot=True,
    cmap="crest",  # You can change this to another palette if you prefer
    fmt='.2f',     # Format for the annotations
    linewidths=.5, # Lines between cells
    cbar_kws={"shrink": .8}  # Color bar adjustments
)

plt.title('Spearman Correlation Heatmap')
plt.show()
```

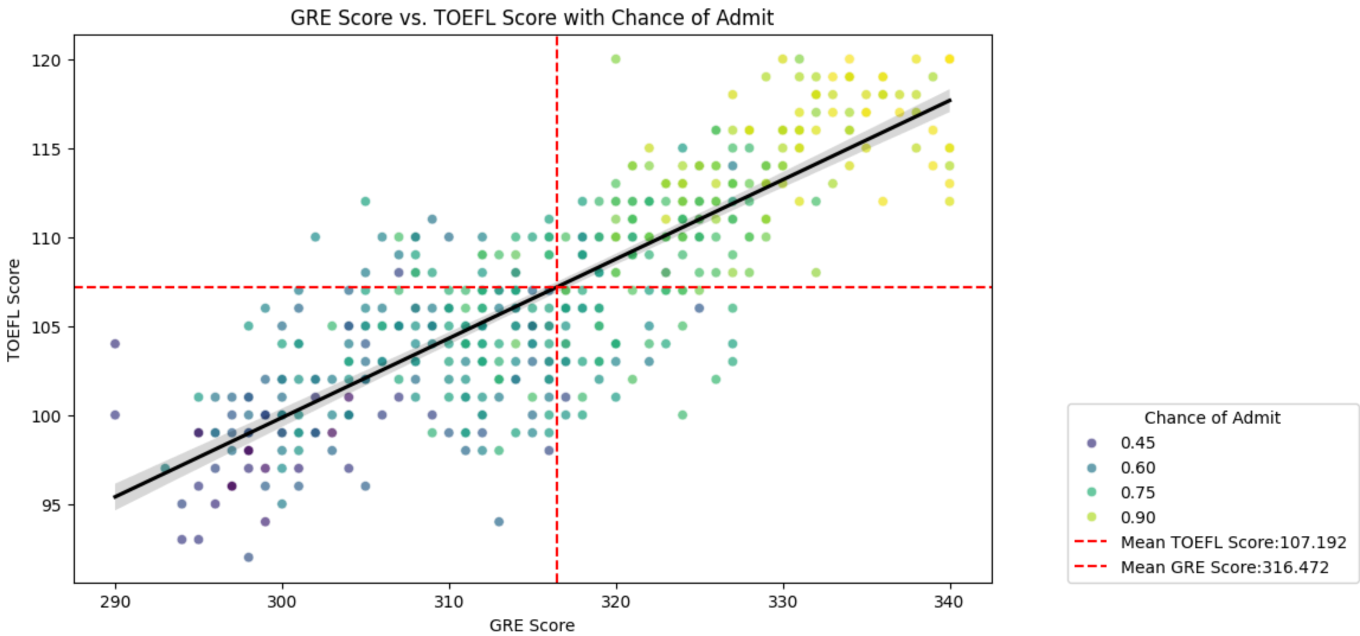


In [12]:

```python
def plot_graph(df, col1, col2,col3):
    plt.figure(figsize=(10, 6))

    # Scatter plot with hue
    sns.scatterplot(
        data=df,
        x=col1,
        y=col2,
        hue=col3,
        palette='viridis',
        alpha=0.7
    )

    # Regression line
    sns.regplot(
        data=df,
        x=col1,
        y=col2,
        scatter=False,
        color='black'
    )

    # Horizontal line at the mean TOEFL score
    plt.axhline(df[col2].mean(), color='red', linestyle='--', label=f'Mean {col2}:{df[col2].mean()} ')
    plt.axvline(df[col1].mean(), color='red', linestyle='--', label=f'Mean {col1}:{df[col1].mean()}')

    # Adding title and labels
    plt.title(f'{col1} vs. {col2} with {col3}')
    plt.xlabel(col1)
    plt.ylabel(col2)
    plt.legend(title=f"{col3}", loc='lower right', bbox_to_anchor=(1.4, 0), borderaxespad=0)   # Adjust legend position
    plt.show()

# Example usage:
# plot_scores_with_regression(jambotree_df, 'GRE Score', 'TOEFL Score')
```

In [13]:

```python
plot_graph(jambotree_df,'GRE Score','TOEFL Score','Chance of Admit')
```



- The observation that higher GRE scores correlate with higher TOEFL scores suggests a positive relationship between academic ability and language proficiency.
- This trend indicates that students excelling in the GRE, which assesses analytical and quantitative skills, often also perform well in the TOEFL, measuring English proficiency.
- Consequently, higher scores in these tests enhance a candidate's application, reflecting strong academic preparedness and increasing their chances of admission to competitive programs, as many institutions use these scores as critical components of their evaluation criteria.

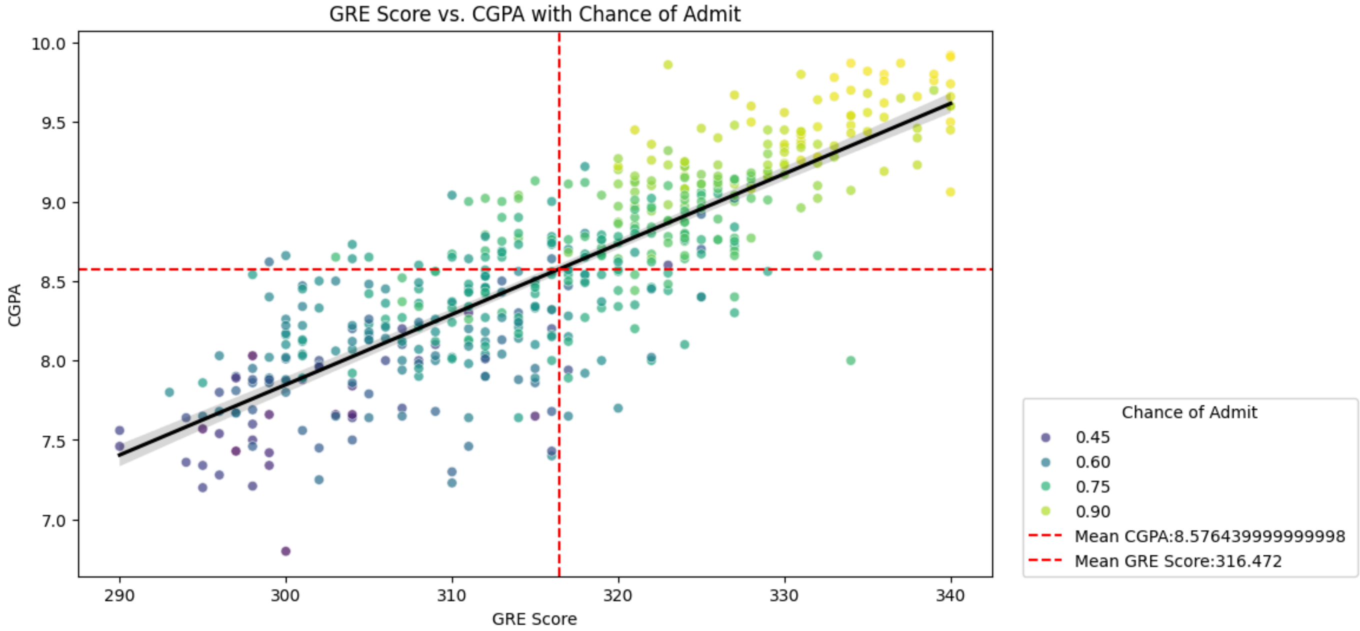
In [14]:

```python
jambotree_df.columns
```

Out[14]:

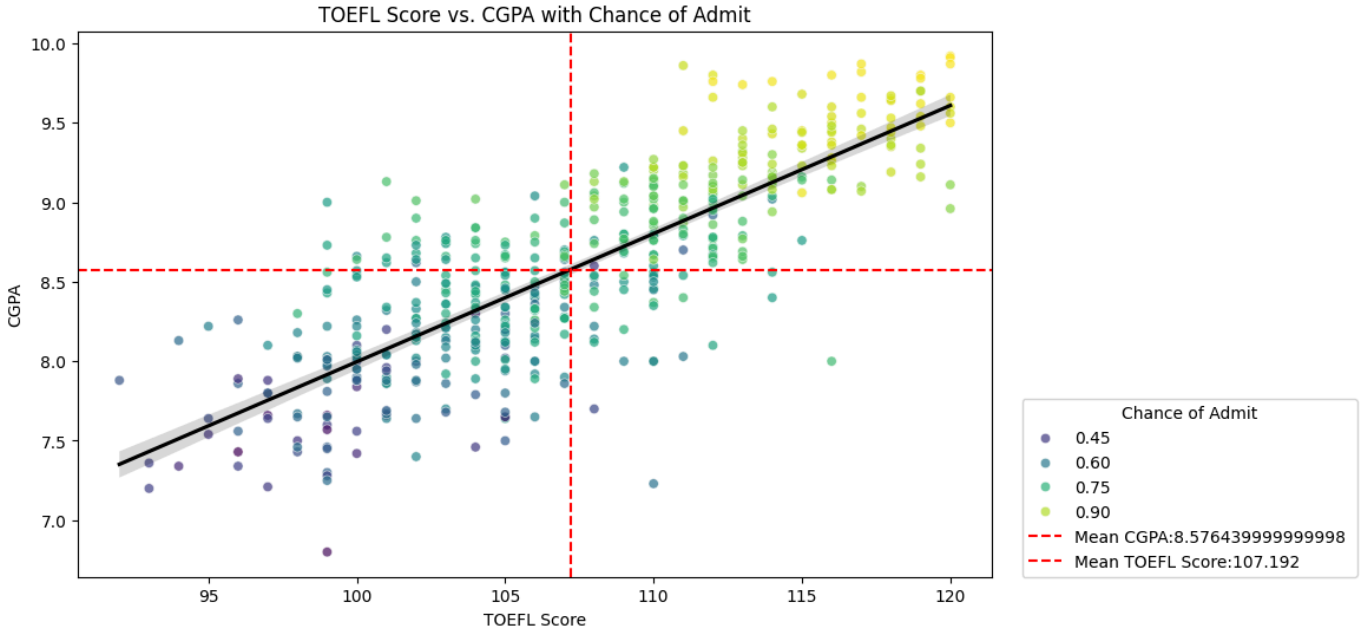
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR', 'CGPA',
       'Research', 'Chance of Admit'],
      dtype='object')
```

In [15]:
```
plot_graph(jambotree_df,'GRE Score','CGPA','Chance of Admit')
```



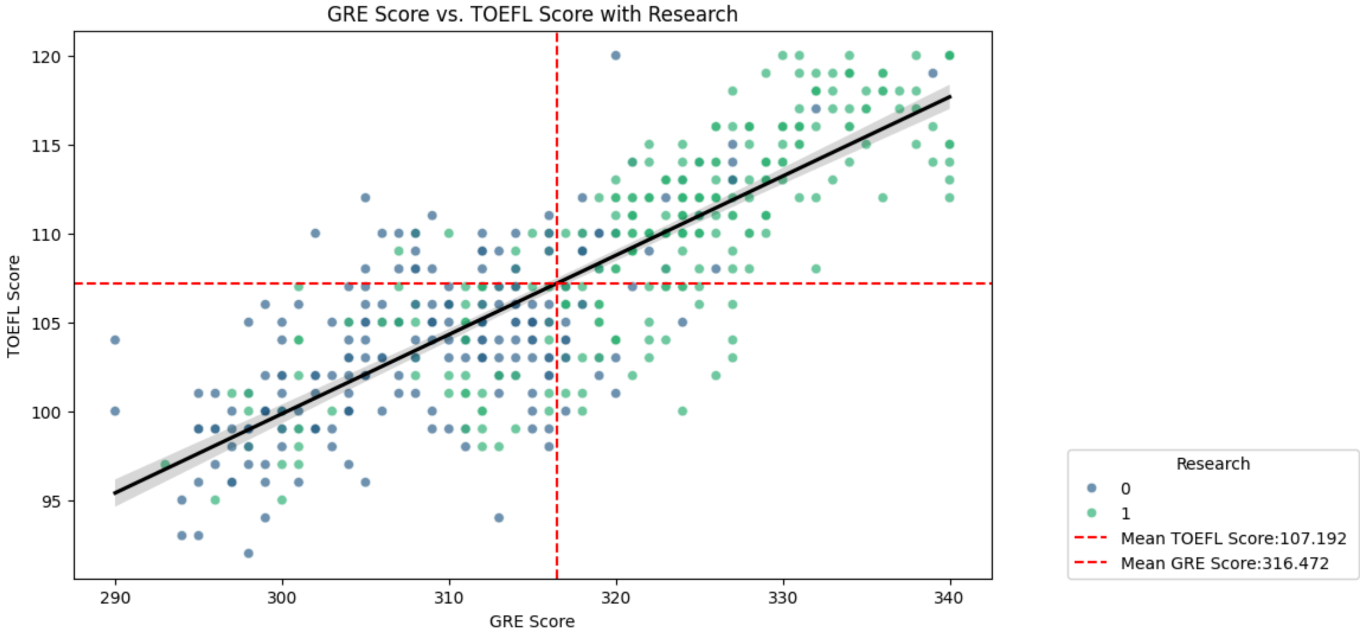
- Analysis of the data from the graph reveals a clear correlation between higher GRE scores and CGPA with increased chances of admission to graduate programs.
- As the graph illustrates, applicants with elevated GRE scores and CGPAs consistently have higher acceptance rates, suggesting that admissions committees prioritize these metrics as indicators of academic readiness.
- This trend underscores the importance of a strong academic foundation and test performance in a competitive admissions landscape.
- However, it also highlights the need for applicants to present a comprehensive profile, as other factors like research experience and personal statements can complement these quantitative measures.
- Ultimately, while high GRE scores and CGPAs significantly boost admission prospects, a well-rounded application remains essential for standing out in a crowded field.

In [16]:
```
plot_graph(jambotree_df,'TOEFL Score','CGPA','Chance of Admit')
```



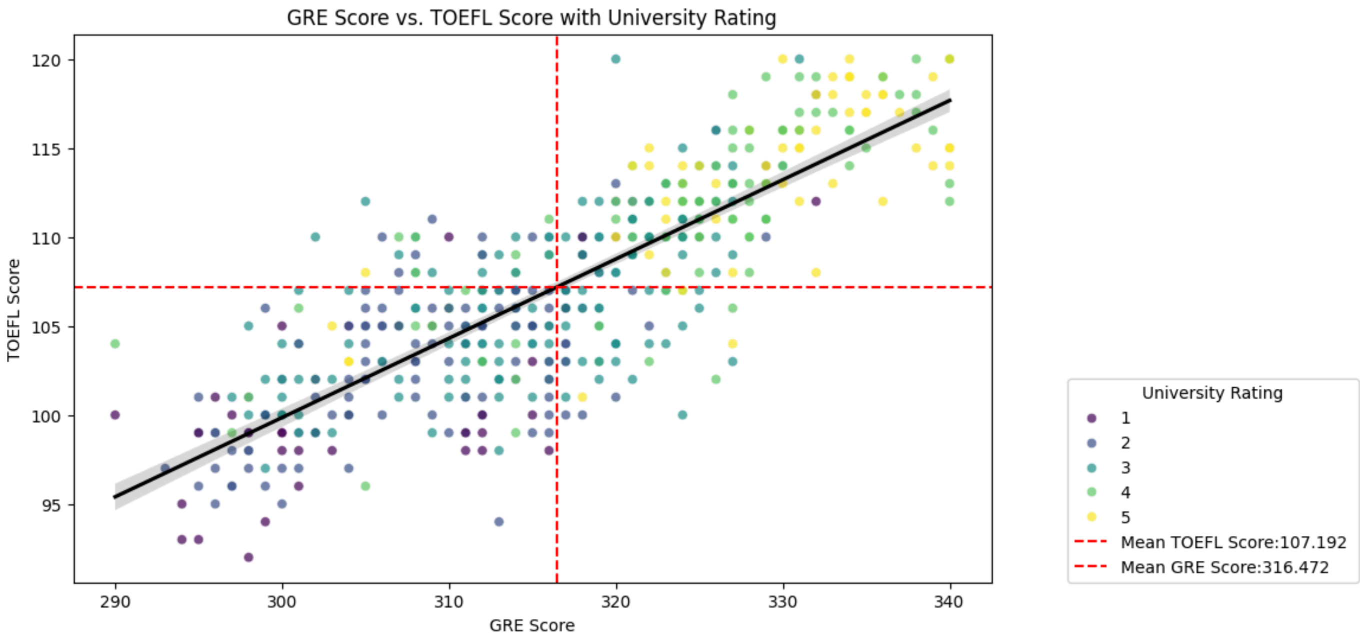
- Analysis of the data from the graph reveals a clear **correlation between higher TOEFL scores and CGPA with increased chances of admission to graduate programs**.
- As the graph illustrates, applicants with elevated TOEFL scores and CGPAs consistently have higher acceptance rates, suggesting that admissions committees prioritize these metrics as indicators of academic readiness.
- This trend underscores the importance of a strong academic foundation and test performance in a competitive admissions landscape.
- However, it also highlights the need for applicants to present a comprehensive profile, as other factors like research experience and personal statements can complement these quantitative measures.
- Ultimately, while high TOEFL scores and CGPAs significantly boost admission prospects, a well-rounded application remains essential for standing out in a crowded field.

In [17]:
```
plot_graph(jambotree_df,'GRE Score','TOEFL Score','Research')
```



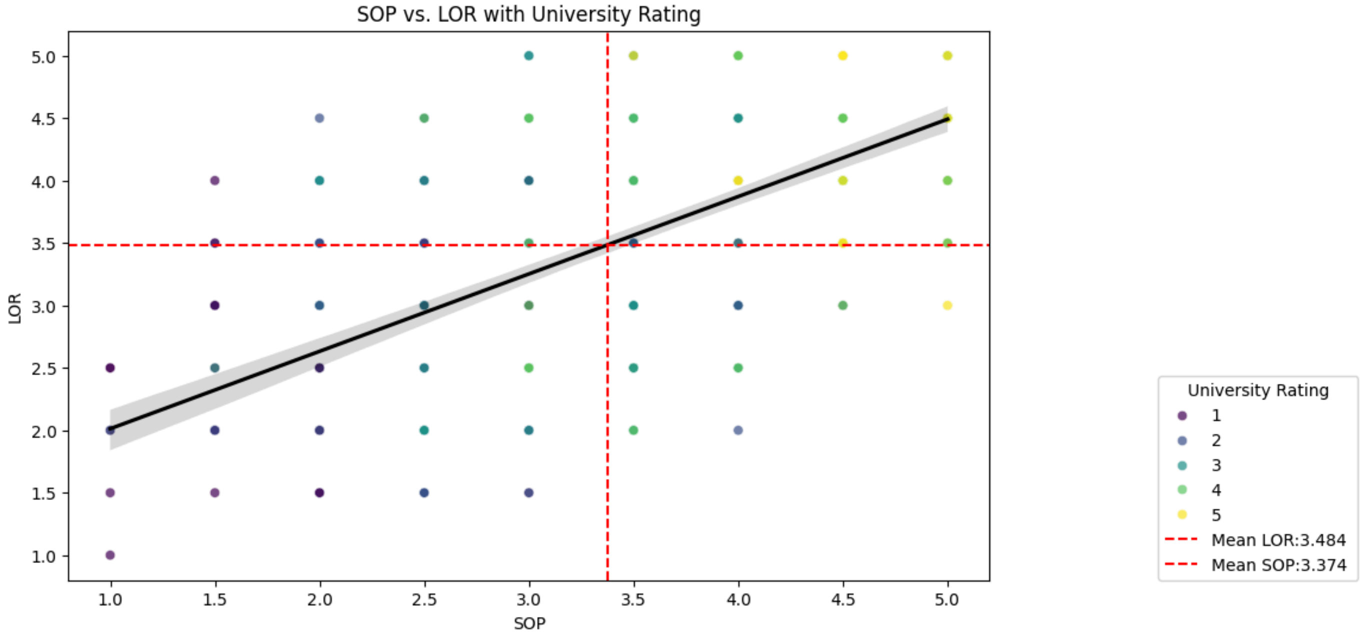
- There is a clear **positive correlation between high GRE and TOEFL scores**, suggesting that candidates who excel in one area tend to perform well in the other, reflecting strong analytical and language skills.
- Furthermore, the prevalence of research experience among these applicants likely contributes to their elevated scores, indicating that engagement in research enhances critical thinking and communication abilities essential for success in standardized testing.
- This combination of strong academic performance and research involvement provides a competitive advantage in the admissions process, underscoring the importance of both credentials for prospective graduate students.
- Ultimately, the data reinforces the notion that research experience is a valuable asset that can significantly bolster an applicant's profile.

In [18]:
```
plot_graph(jambotree_df,'GRE Score','TOEFL Score','University Rating')
```



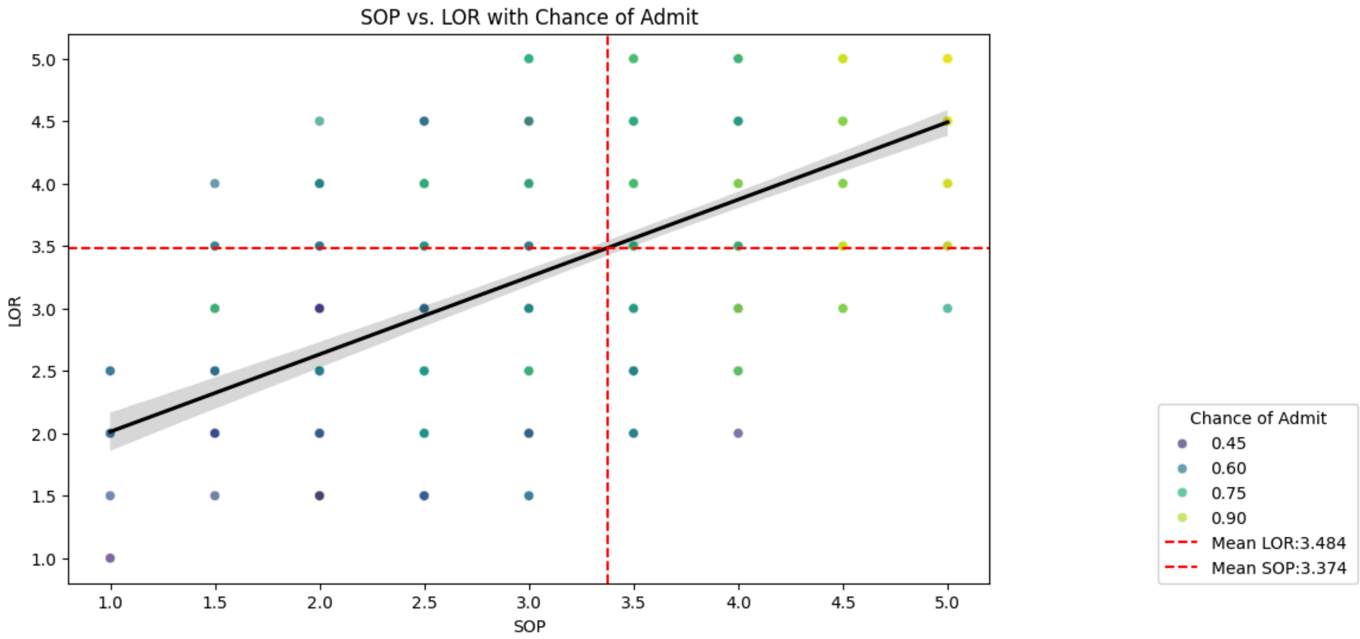
- The plot shows a strong positive correlation between GRE and TOEFL scores, with **higher university ratings** (ratings 4 and 5) clustering at the **top-right, indicating higher test scores**.
- Most applicants with better university ratings score above the mean GRE (316.472) and TOEFL (107.192) lines, while lower ratings (ratings 1 and 2) are associated with lower scores and more spread-out data points.
- The clear trend suggests that higher test scores are linked to better-rated universities, with a few outliers showing uneven performance between the two tests

In [19]:
```
plot_graph(jambotree_df,'SOP','LOR','University Rating')
```



- The plot shows a **positive correlation between SOP and LOR scores, with higher university ratings (4 and 5) associated with stronger scores**.
- Most top-rated universities have SOP and LOR scores above the mean values (3.374 for SOP and 3.484 for LOR), while lower-rated universities show more variation and tend to fall below these averages.
- This suggests that stronger SOP and LOR scores are linked to higher university ratings.

In [20]:
```
plot_graph(jambotree_df,'SOP','LOR','Chance of Admit')
```

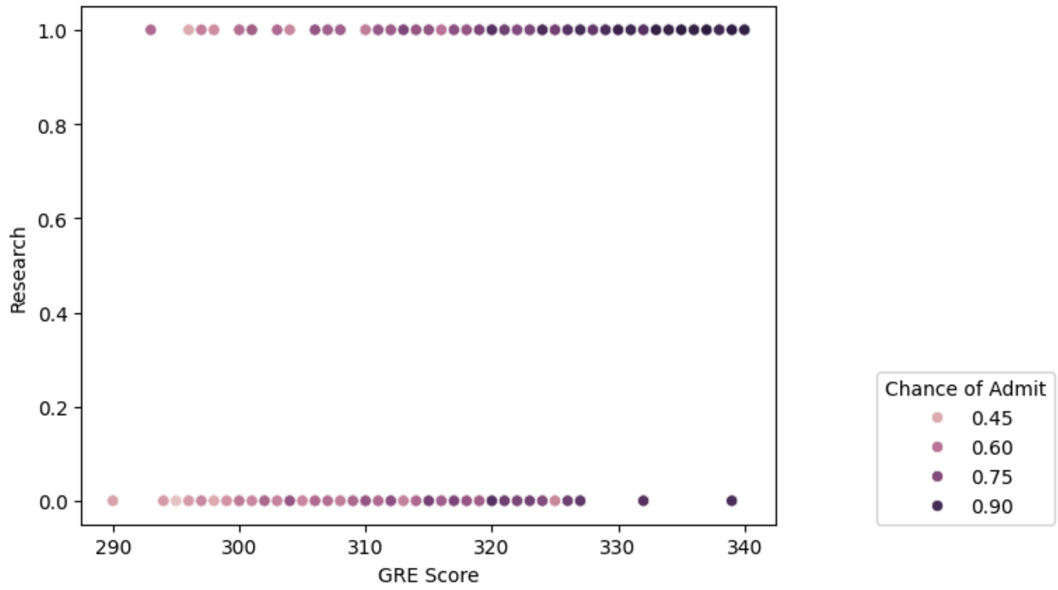


- This scatter plot shows the relationship between SOP and LOR scores in relation to the chance of admission. Higher SOP and LOR scores, particularly above the mean values (3.374 for SOP and 3.484 for LOR), are associated with a higher chance of admission, as indicated by the yellow and light green points (chances of 0.75 to 0.90).
- The black trend line suggests a positive correlation between SOP and LOR, meaning that applicants with strong scores in both areas tend to have a better chance of being admitted. Lower chances of admission (0.45 and 0.60) are more common among those with lower SOP and LOR scores.

In [21]:

```
sns.scatterplot(data=jambotree_df,x='GRE Score',y='Research',hue='Chance of Admit')
plt.legend(title='Chance of Admit', loc='lower right', bbox_to_anchor=(1.4, 0), borderaxespad=0)
```

Out[21]: <matplotlib.legend.Legend at 0x20230ae0a30>

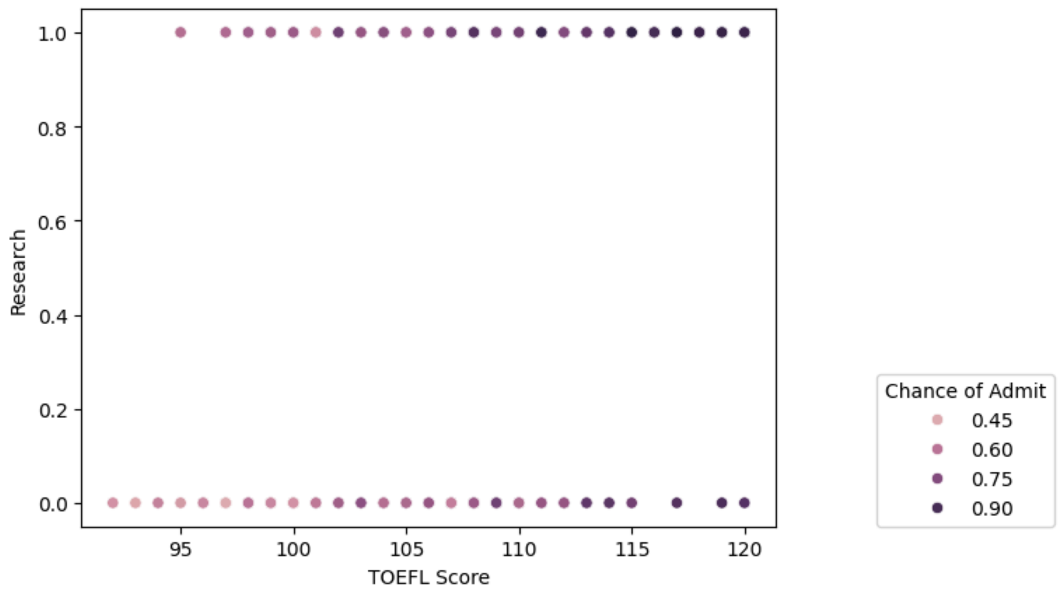


- The plot highlights the positive impact of both GRE scores and research experience on the likelihood of admission.
- Higher GRE scores consistently correlate with a greater chance of admission, but having research experience further amplifies this effect.
- Applicants with research experience generally have better chances of acceptance, even if their GRE scores are moderate. - - This suggests that both academic performance (as indicated by GRE scores) and research experience are significant factors in the admissions process, with research experience acting as a potential differentiator for candidates.

In [22]:

```
sns.scatterplot(data=jambotree_df,x='TOEFL Score',y='Research',hue='Chance of Admit')
plt.legend(title='Chance of Admit', loc='lower right', bbox_to_anchor=(1.4, 0), borderaxespad=0)
```

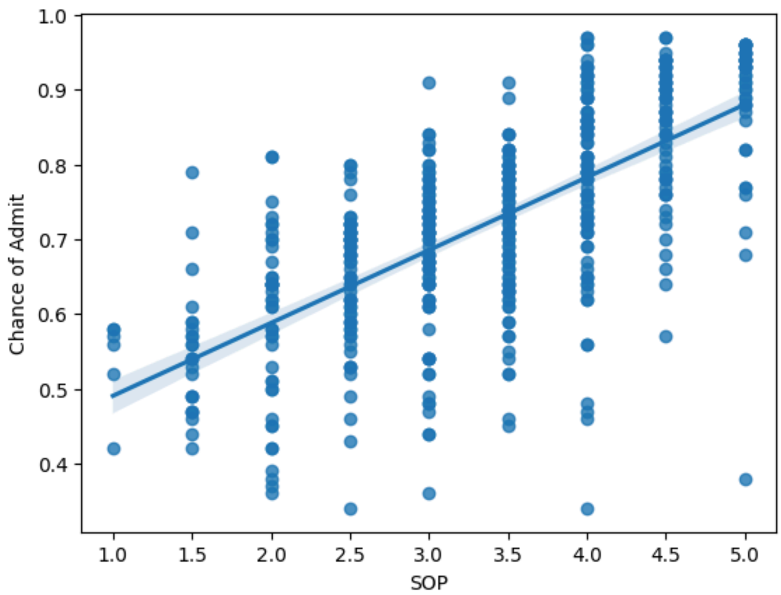
Out[22]: <matplotlib.legend.Legend at 0x20232af9270>



- The plot highlights the positive impact of both TOEFL scores and research experience on the likelihood of admission.
- Higher TOEFL scores consistently correlate with a greater chance of admission, but having research experience further amplifies this effect.
- Applicants with research experience generally have better chances of acceptance, even if their TOEFL scores are moderate. This suggests that both academic performance (as indicated by TOEFL scores) and research experience are significant factors in the admissions process, with research experience acting as a potential differentiator for candidates.

In [23]:
```
sns.regplot(data=jambotree_df,x="SOP",y="Chance of Admit")
```

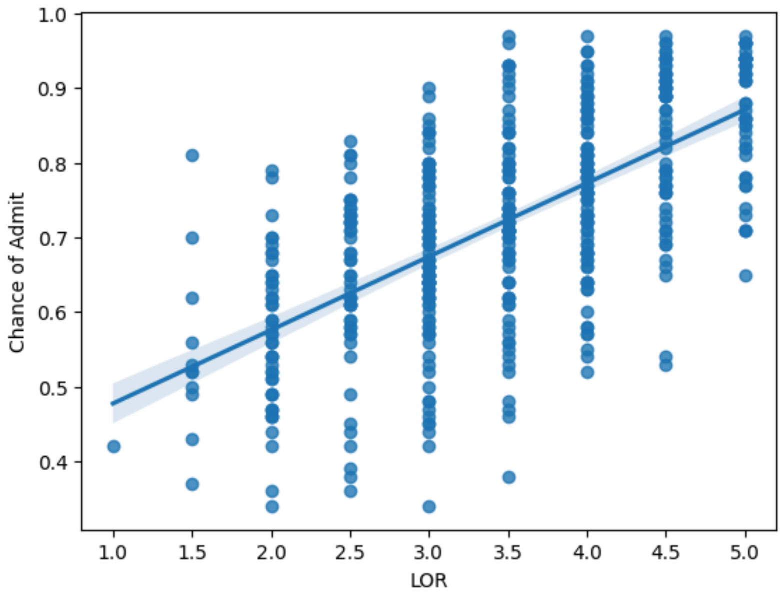
Out[23]: <Axes: xlabel='SOP', ylabel='Chance of Admit'>



- It shows a clear positive correlation between the two variables, with higher SOP scores generally corresponding to a greater chance of admission. As the SOP score increases from 1.0 to 5.0, there is a noticeable upward trend in the chance of admit, suggesting that a stronger SOP positively influences the likelihood of acceptance.
- However, there is some variability, especially at higher SOP scores, where applicants with the same SOP score (e.g., 4.0 or 5.0) still display varying chances of admission, indicating that other factors beyond SOP likely play a role in the overall decision.

In [24]:
```
sns.regplot(data=jambotree_df,x="LOR",y="Chance of Admit")
```

Out[24]: <Axes: xlabel='LOR', ylabel='Chance of Admit'>



- **A positive correlation is observed, where higher LOR ratings are generally associated with a greater chance of admission. - As the LOR score increases from 1.0 to 5.0, the likelihood of admission also tends to rise, following a clear upward trend**.
- However, there is noticeable variability at each LOR rating, particularly at higher scores (e.g., 4.0 or 5.0), where applicants with the same LOR score have varying chances of admission.
- This suggests that while strong LORs are important, other factors likely influence the final admission decision.

Linear Regression

In [25]:
```
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error,r2_score
from sklearn.linear_model import LinearRegression,Ridge,Lasso
import statsmodels.api as sm
```

In [26]:
```
x=jambotree_df.iloc[:,:-1]
y=jambotree_df.iloc[:,-1:]

```

In [27]:
```
x.shape,y.shape
```

Out[27]:
```
((500, 7), (500, 1))
```

In [28]:
```
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.20, random_state=42)
```

In [29]:
```
scaler=StandardScaler()
```

In [30]:
```
X_train_std=scaler.fit_transform(X_train)
```

In [31]:
```
X_train=pd.DataFrame(X_train_std, columns=X_train.columns)

```

In [32]:
```
X_test=scaler.transform(X_test)
```

Using Sklearn

In [33]:
```
model=LinearRegression()
```

In [34]:
```
model.fit(X_train,Y_train)
```

Out[34]:
```
▾ LinearRegression
LinearRegression()
```

In [35]:
```
prediction=model.predict(X_test)
train_pred=model.predict(X_train)
```

In [ ]:
```

```

In [36]:
```
def error_calculation(data1,data2):
    print(f"MAE : {mean_absolute_error(data1,data2)}")
    print(f"MSE : {mean_squared_error(data1,data2)}")
    print(f"RMSE : {np.sqrt(mean_squared_error(data1,data2))}")
def scores(data1,data2):
    r_squared=model.score(data1,data2)
    print(f"R_squred : {r_squared}")
    n =x.shape[0]  # number of observations
    p = x.shape[1]  # number of predictors
    adjusted_r_squared = 1 - (1 -r_squared) * (n - 1) / (n - p - 1)
    print(f"Adjusted_R_squred : {adjusted_r_squared}")


```

In [37]:
```
print("-----------------------------Train--------------------------------")
error_calculation(Y_train,train_pred)
scores(X_train,Y_train)
print("-----------------------------Test---------------------------------")
error_calculation(Y_test,prediction)
scores(X_test,Y_test)

```

```
-----------------------------Train--------------------------------
MAE : 0.042533340611643135
MSE : 0.003526555478455758
RMSE : 0.05938480848210052
R_squred : 0.8210671369321554
Adjusted_R_squred : 0.8185213441649299
-----------------------------Test---------------------------------
MAE : 0.04272265427705366
MSE : 0.003704655398788409
RMSE : 0.0608658804157831
R_squred : 0.8188432567829629
Adjusted_R_squred : 0.816265823444509
```

Using StatsModel

Hypothesis Testing

H0: Feature has no Significance Difference

H1:Feature has Significant difference

In [38]:
```python
X_sm=sm.add_constant(X_train)
model1=sm.OLS(Y_train.values,X_sm)
result=model1.fit()

```

In [39]:
```python
print(result.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       0.821
Model:                            OLS   Adj. R-squared:                  0.818
Method:                 Least Squares   F-statistic:                     257.0
Date:                Sun, 13 Oct 2024   Prob (F-statistic):          3.41e-142
Time:                        20:08:46   Log-Likelihood:                 561.91
No. Observations:                 400   AIC:                            -1108.
Df Residuals:                     392   BIC:                            -1076.
Df Model:                           7                                         
Covariance Type:            nonrobust                                         
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.7242      0.003    241.441      0.000       0.718       0.730
GRE Score             0.0267      0.006      4.196      0.000       0.014       0.039
TOEFL Score           0.0182      0.006      3.174      0.002       0.007       0.030
University Rating     0.0029      0.005      0.611      0.541      -0.007       0.012
SOP                   0.0018      0.005      0.357      0.721      -0.008       0.012
LOR                   0.0159      0.004      3.761      0.000       0.008       0.024
CGPA                  0.0676      0.006     10.444      0.000       0.055       0.080
Research              0.0119      0.004      3.231      0.001       0.005       0.019
==============================================================================
Omnibus:                       86.232   Durbin-Watson:                   2.050
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              190.099
Skew:                          -1.107   Prob(JB):                     5.25e-42
Kurtosis:                       5.551   Cond. No.                         5.65
==============================================================================

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

We can see that for **University Ranking and SOP p value** is much **larger** than **significance value(0.05)**

SO there is those features **have no significance**

Removing SOP (SInce SOP has the highest p value

In [40]:
```python
X_train_new=X_train.drop(columns={'SOP'})
```

In [41]:
```python
x_sm_new=sm.add_constant(X_train_new)
model_sop_drop=sm.OLS(Y_train.values,x_sm_new)
result=model_sop_drop.fit()
```

In [42]:
```python
print(result.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       0.821
Model:                            OLS   Adj. R-squared:                  0.818
Method:                 Least Squares   F-statistic:                     300.4
Date:                Sun, 13 Oct 2024   Prob (F-statistic):          2.01e-143
Time:                        20:08:46   Log-Likelihood:                 561.85
No. Observations:                 400   AIC:                            -1110.
Df Residuals:                     393   BIC:                            -1082.
Df Model:                           6                                         
Covariance Type:            nonrobust                                         
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.7242      0.003    241.710      0.000       0.718       0.730
GRE Score             0.0266      0.006      4.192      0.000       0.014       0.039
TOEFL Score           0.0185      0.006      3.240      0.001       0.007       0.030
University Rating     0.0035      0.005      0.779      0.437      -0.005       0.012
LOR                   0.0163      0.004      4.056      0.000       0.008       0.024
CGPA                  0.0680      0.006     10.730      0.000       0.056       0.080
Research              0.0120      0.004      3.240      0.001       0.005       0.019
==============================================================================
Omnibus:                       85.621   Durbin-Watson:                   2.047
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              188.163
Skew:                          -1.101   Prob(JB):                     1.38e-41
Kurtosis:                       5.539   Cond. No.                         5.19
==============================================================================

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

In [43]:
```python
X_train_new=X_train_new.drop(columns={'University Rating'})
x_sm_new=sm.add_constant(X_train_new)
result=sm.OLS(Y_train.values,x_sm_new).fit()

print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.821
Model:                            OLS   Adj. R-squared:                  0.818
Method:                 Least Squares   F-statistic:                     360.8
Date:                Sun, 13 Oct 2024   Prob (F-statistic):         1.36e-144
Time:                        20:08:46   Log-Likelihood:                 561.54
No. Observations:                 400   AIC:                            -1111.
Df Residuals:                     394   BIC:                            -1087.
Df Model:                           5
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const           0.7242      0.003    241.830      0.000       0.718       0.730
GRE Score       0.0269      0.006      4.245      0.000       0.014       0.039
TOEFL Score     0.0191      0.006      3.391      0.001       0.008       0.030
LOR             0.0172      0.004      4.465      0.000       0.010       0.025
CGPA            0.0691      0.006     11.147      0.000       0.057       0.081
Research        0.0122      0.004      3.328      0.001       0.005       0.019
==============================================================================
Omnibus:                       84.831   Durbin-Watson:                   2.053
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              185.096
Skew:                          -1.094   Prob(JB):                     6.41e-41
Kurtosis:                       5.514   Cond. No.                         4.76
==============================================================================

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

Testing Assumptions of Linear Regression Model

No MultiCollinearity ---- Using VIF

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 [44]:
```python
from statsmodels.stats.outliers_influence import variance_inflation_factor

```

In [45]:
```python
#checking vif
def calculate_vif(dataset,col):
 dataset=dataset.drop(columns=col,axis=1)
 vif=pd.DataFrame()
 vif['features']=dataset.columns
 vif['VIF_Value']=[variance_inflation_factor(dataset.values,i) for i in range(dataset.shape[1])]
 return vif
calculate_vif(X_train_new,[])

```

Out[45]:

	features	VIF_Value
0	GRE Score	4.471557
1	TOEFL Score	3.540082
2	LOR	1.655867
3	CGPA	4.281365
4	Research	1.504670

Since VIF is less than 5 there is no multicollinarity

In [50]:
```
X_test = pd.DataFrame(X_test, columns=X_train.columns)  # Ensure column names match
X_test_new = X_test.drop(columns=['SOP', 'University Rating'])  # Drop the same columns
X_test_new = sm.add_constant(X_test_new)  # Add constant term

# Ensure the model's parameters are aligned with the new test set
predictions = result.predict(X_test_new)


```

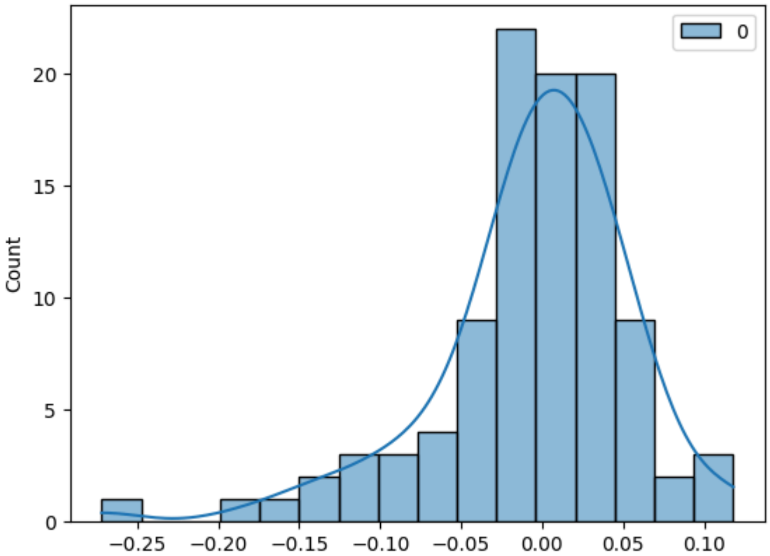
In [51]:
```
predictions=predictions.values.reshape(-1, 1)
```

Mean of Residuals:

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 [53]:
```
residuals=Y_test.values-prediction
```

In [54]:
```
sns.histplot(residuals,kde=True,color='blue')
plt.show()
```



In [55]:
```
residuals.mean()
```

Out[55]:
```
-0.005453623717661262
```

Since the mean of residuals is very close to 0, we can say that the model is UnBiased .

Normality of Residuals

In [56]:
```
shapiro(residuals)
```

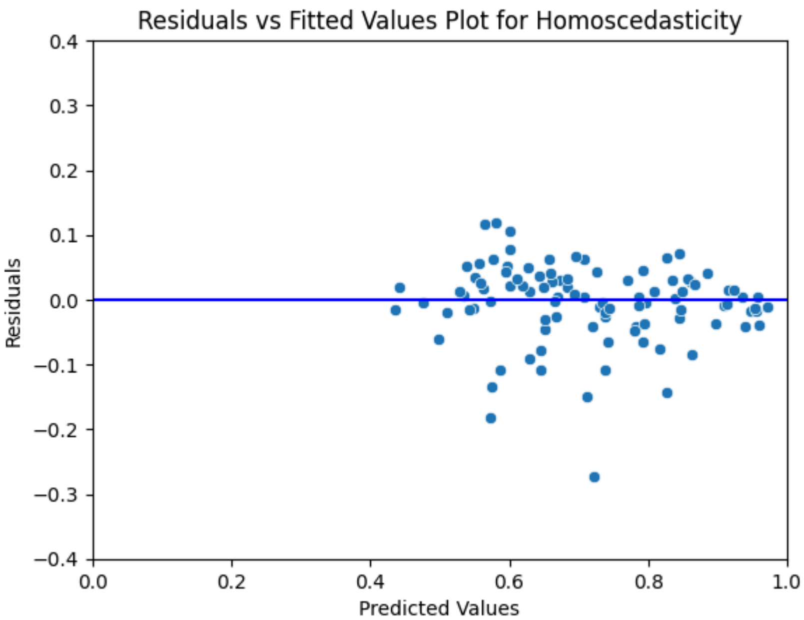
Out[56]:
```
ShapiroResult(statistic=0.9178698658943176, pvalue=1.0869382094824687e-05)
```

Since the statistics are almost same euqal to 1 but p value is <0.05 so **not normal**

No Heteroskedasticity

In [57]:

```
# Convert prediction and residuals to 1D arrays if not already
prediction_1d = predictions.ravel()  # Converts to 1D array
residuals_1d = residuals.ravel()    # Converts to 1D array

# Create the scatter plot
p = sns.scatterplot(x=prediction_1d, y=residuals_1d)
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')

# Add a horizontal reference line at y=0 (for homoscedasticity)
sns.lineplot(x=[0, 1], y=[0, 0], color='blue')

# Add the title
plt.title('Residuals vs Fitted Values Plot for Homoscedasticity')
plt.ylim(-0.4,0.4)
plt.xlim(0,1)
# Show the plot
plt.show()

```



In [58]:

```
# Import necessary libraries
from statsmodels.compat import lzip
from statsmodels.stats.diagnostic import het_goldfeldquandt

# Perform the Goldfeld-Quandt test
name = ['F-statistic', 'p-value']
test = het_goldfeldquandt(Y_train, x_sm_new)

# Print the results
if lzip(name, test)[1][1]>0.05:
    print(f'Homocedasticity  with p_value : {lzip(name, test)[1][1]}')
else:
    print(f'Heterocedasticity with p_value of {lzip(name, test)[1][1]}')

```

```
Homocedasticity  with p_value : 0.6139024845884404
```

In [73]:

```
model.coef_
```

Out[73]:

```
array([[0.02667052, 0.01822633, 0.00293995, 0.001788  , 0.0158655 ,
        0.06758106, 0.01194049]])
```

In [78]:

```
coef_df_LR = pd.DataFrame({
    'Features': result.params.index,  # Access coefficients through result.params
    'Coefficients': result.params.values
})
intercept_df_coef_ = pd.DataFrame({'Features': ['Intercept'], 'Coefficients': model.intercept_})
coef_df_LR = pd.concat([coef_df_LR, intercept_df_coef_], ignore_index=True)
coef_df_LR=coef_df_LR.loc[1:,:]
```

In [ ]:

```

```

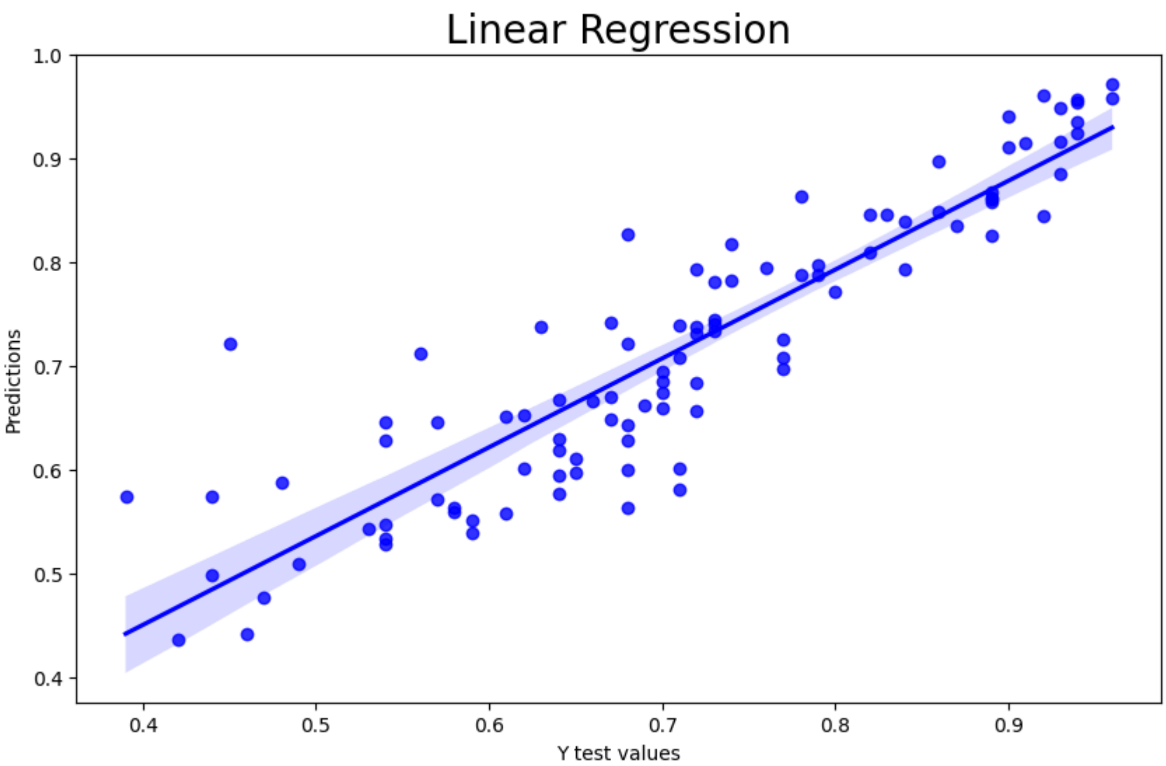
In [79]:

```
coef_df_LR.reset_index(drop=True, inplace=True)
```

In [80]:
```
coef_df_LR
```
Out[80]:

	Features	Coefficients
0	GRE Score	0.026879
1	TOEFL Score	0.019106
2	LOR	0.017207
3	CGPA	0.069066
4	Research	0.012226
5	Intercept	0.724175

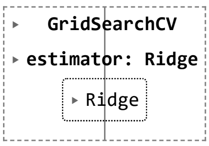
In [81]:
```
plt.figure(figsize=(10,6))
sns.regplot(x=Y_test.values.ravel(),y=predictions.ravel(),color='b')
plt.title('Linear Regression', fontsize=20)
plt.ylabel('Predictions')
plt.xlabel('Y test values')
```
Out[81]: Text(0.5, 0, 'Y test values')



In [126]:
```
error_calculation(Y_test,predictions)
scores(X_test,Y_test)
```
```
MAE : 0.04292345578265783
MSE : 0.0037730207651168967
RMSE : 0.061424919740418846
R_squred : 0.8188432567829629
Adjusted_R_squred : 0.816265823444509
```

Regularization

In [127]:
```
#checking the best alpha
ridge = Ridge()
param_grid = {'alpha': [0.001, 0.01, 0.1, 1, 10]}
grid_search = GridSearchCV(ridge, param_grid, cv=5)
grid_search.fit(X_train, Y_train)
```
Out[127]:

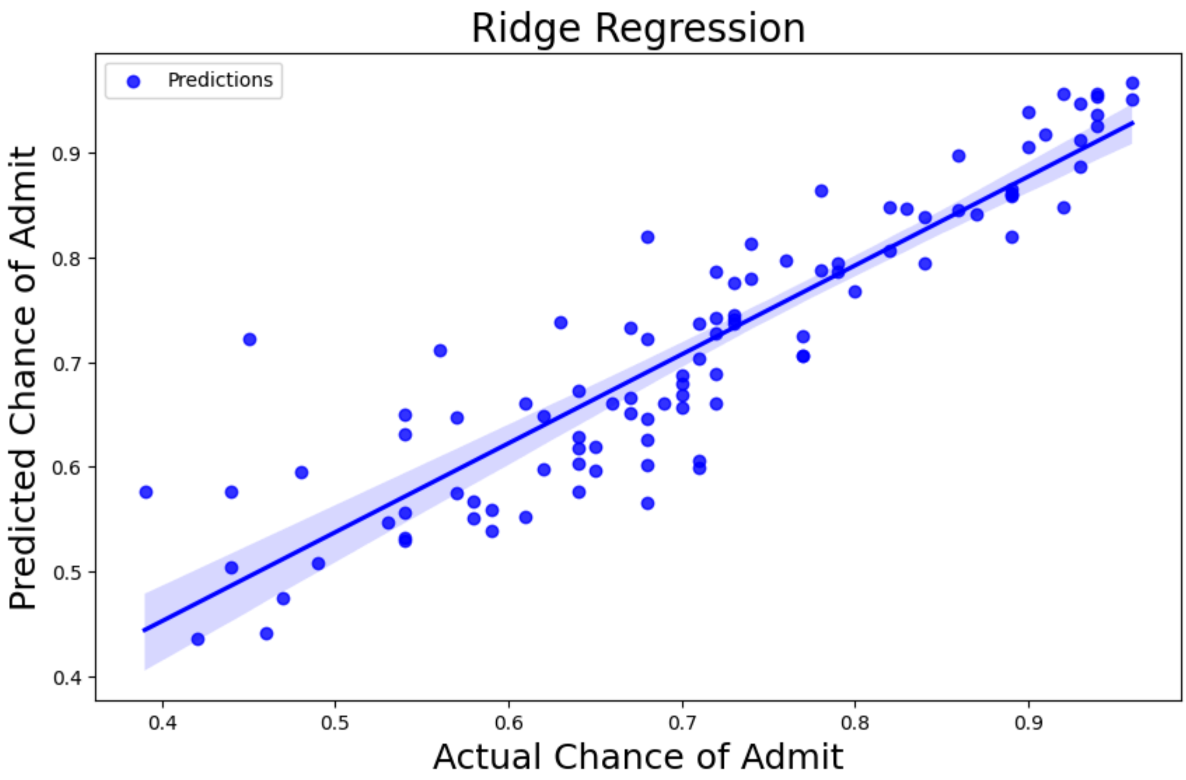


In [128]:
```
print(f"Best alpha: {grid_search.best_params_['alpha']}")
```
```
Best alpha: 10
```

In [ ]:
```

```

In [129]:
```
ridge = Ridge(alpha=grid_search.best_params_['alpha'])  # alpha is the regularization strength
ridge.fit(X_train, Y_train)
y_pred_ridge = ridge.predict(X_test)
```

In [130]:
```
#plotting ridge
plt.figure(figsize=(10,6))
sns.regplot(x=Y_test, y=y_pred_ridge, label='Predictions', color='blue')
plt.title('Ridge Regression', fontsize=20)
plt.xlabel('Actual Chance of Admit', fontsize=18)
plt.ylabel('Predicted Chance of Admit', fontsize=18)
plt.legend()
plt.show()
```



In [131]:
```
error_calculation(Y_test,y_pred_ridge)
scores(X_test,Y_test)
```

```
MAE : 0.04291617789042092
MSE : 0.003716136548366631
RMSE : 0.06096012260787072
R_squred : 0.8188432567829629
Adjusted_R_squred : 0.816265823444509
```

In [132]:
```
ridge.coef_,X_train.columns,model.coef_
```

Out[132]:
```
(array([[0.02760072, 0.01935237, 0.00393214, 0.00314397, 0.01607462,
         0.06250732, 0.01204283]]),
 Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR', 'CGPA',
        'Research'],
       dtype='object'),
 array([[0.02667052, 0.01822633, 0.00293995, 0.001788  , 0.0158655 ,
         0.06758106, 0.01194049]]))
```

In [133]:
```
ridge_coef_LR = pd.DataFrame(
    {'Features': X_train.columns,  # Use the columns from your training set as column names
     'Coefficients': ridge.coef_[0]  # Convert coefficients to a list
    }
)
# Create DataFrame for intercept
intercept_df_ridge = pd.DataFrame({'Features': ['Intercept'], 'Coefficients': [ridge.intercept_[0]]})  # Ensure it's a list

# Concatenate the coefficients DataFrame and the intercept DataFrame
ridge_coef_LR = pd.concat([ridge_coef_LR, intercept_df_ridge], ignore_index=True)
```

In [ ]:
```

```

In [134]:
```
ridge_coef_LR
```
Out[134]:

	Features	Coefficients
0	GRE Score	0.027601
1	TOEFL Score	0.019352
2	University Rating	0.003932
3	SOP	0.003144
4	LOR	0.016075
5	CGPA	0.062507
6	Research	0.012043
7	Intercept	0.724175

In [145]:
```
#checking the best alpha
lasso = Lasso()
param_grid = {'alpha': [0.001, 0.01, 0.1, 1, 10]}
grid_search = GridSearchCV(lasso, param_grid, cv=5)
grid_search.fit(X_train, Y_train)
```
Out[145]:
```
▸ GridSearchCV
▸ estimator: Lasso
  ▸ Lasso
```

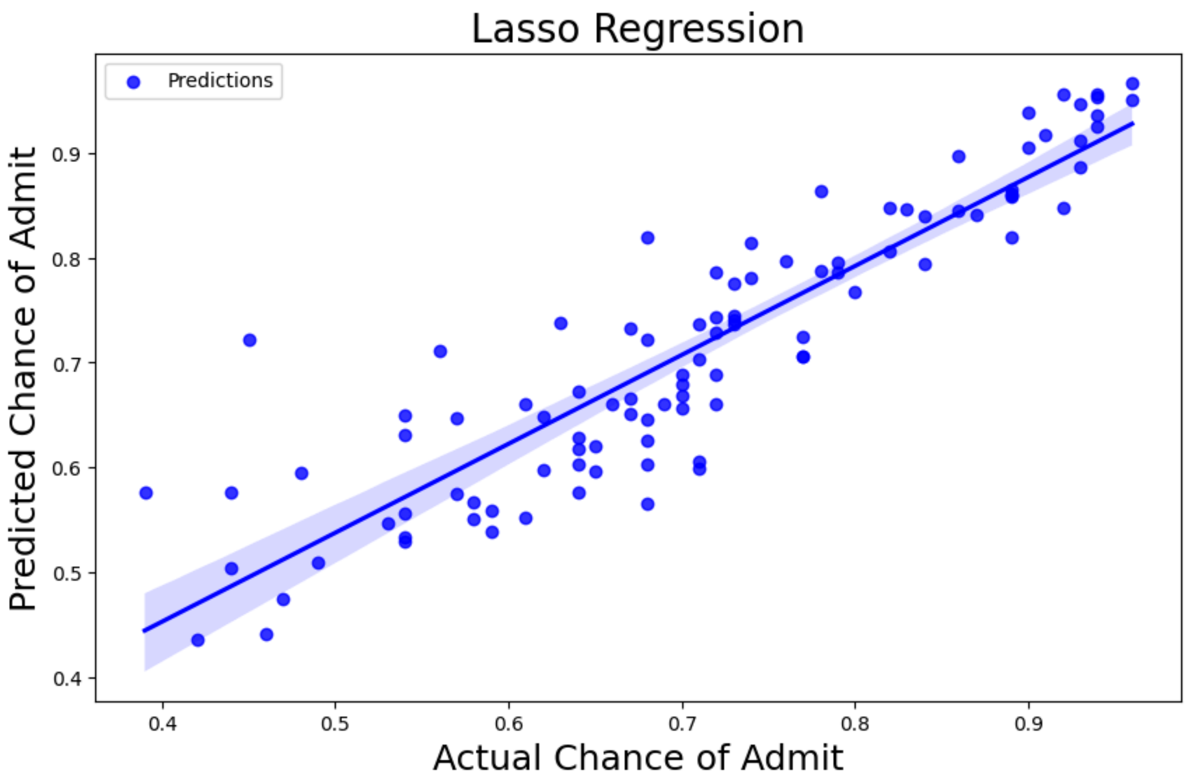
In [ ]:
```

```

In [146]:
```
print(f"Best alpha: {grid_search.best_params_['alpha']}")
```
```
Best alpha: 0.001
```

In [147]:
```
lasso = Lasso(alpha=grid_search.best_params_['alpha'])  # alpha is the regularization strength
lasso.fit(X_train, Y_train)
y_pred_lasso = ridge.predict(X_test)
```

In [148]:
```
lasso.coef_
```
Out[148]:
```
array([0.02661236, 0.01791855, 0.00274603, 0.00159205, 0.01539165,
       0.06773355, 0.01135986])
```

In [149]:
```
#plotting lasso
plt.figure(figsize=(10,6))
sns.regplot(x=Y_test, y=y_pred_lasso, label='Predictions', color='blue')
plt.title('Lasso Regression', fontsize=20)
plt.xlabel('Actual Chance of Admit', fontsize=18)
plt.ylabel('Predicted Chance of Admit', fontsize=18)
plt.legend()
plt.show()
```



In [150]:
```
error_calculation(Y_test,y_pred_lasso)
scores(X_test,Y_test)
```

```
MAE : 0.04291617789042092
MSE : 0.003716136548366631
RMSE : 0.06096012260787072
R_squred : 0.8188432567829629
Adjusted_R_squred : 0.816265823444509
```

In [156]:
```
lasso_coef_LR = pd.DataFrame(
    {'Features': X_train.columns,  # Use the columns from your training set as column names
     'Coefficients': lasso.coef_  # Convert coefficients to a list
    }
)
# Create DataFrame for intercept
intercept_df_ridge = pd.DataFrame({'Features': ['Intercept'], 'Coefficients': [lasso.intercept_[0]]})  # Ensure it's a list

# Concatenate the coefficients DataFrame and the intercept DataFrame
lasso_coef_LR = pd.concat([lasso_coef_LR , intercept_df_ridge], ignore_index=True)
```

In [ ]:
```

```

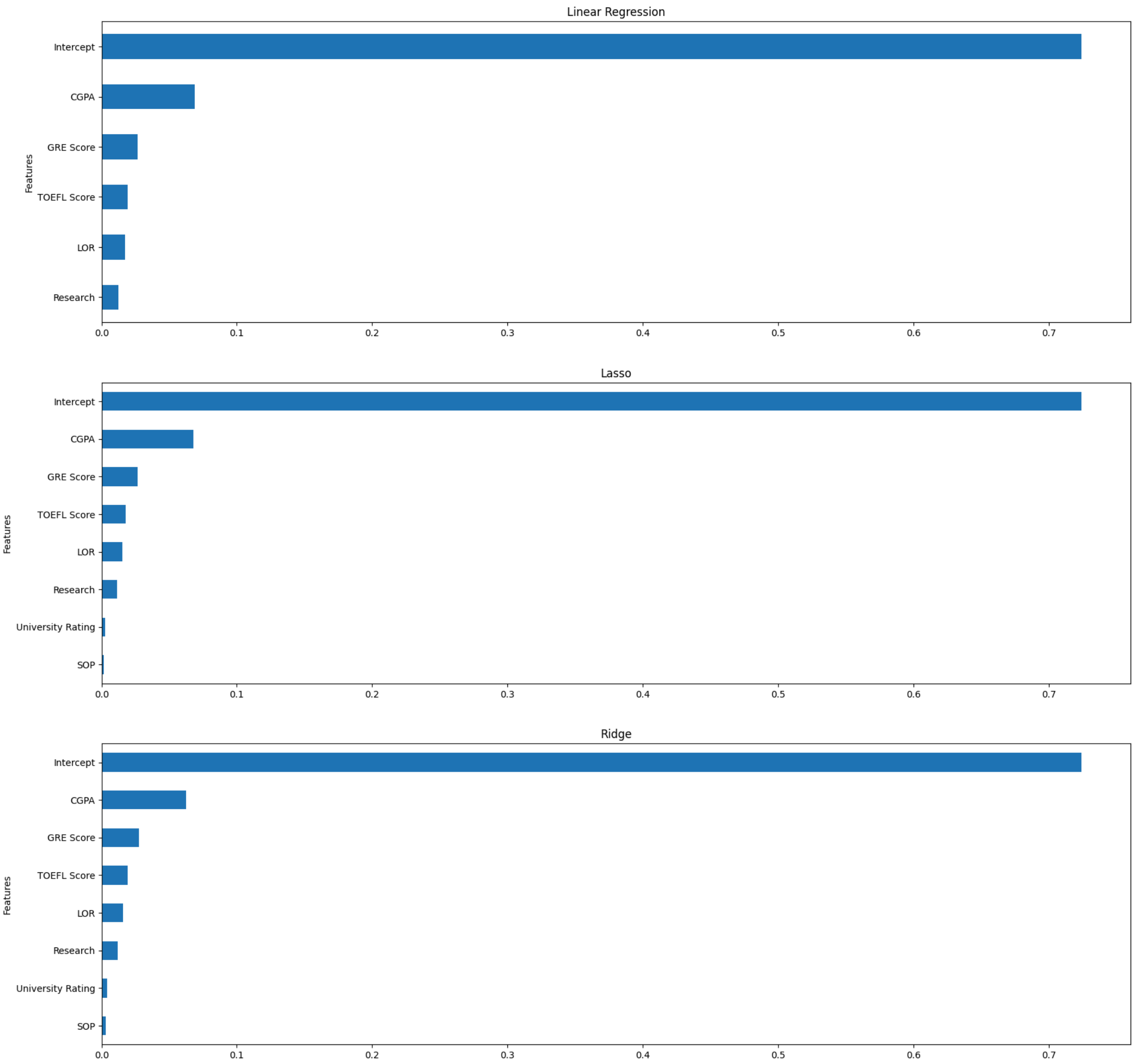
In [157]:
```
lasso_coef_LR
```

Out[157]:

	Features	Coefficients
0	GRE Score	0.026612
1	TOEFL Score	0.017919
2	University Rating	0.002746
3	SOP	0.001592
4	LOR	0.015392
5	CGPA	0.067734
6	Research	0.011360
7	Intercept	0.724175

In [189]:

```python
plt.figure(figsize=(20,20))
plt.subplot(3,1,1)
coef_df_LR.groupby('Features')['Coefficients'].sum().sort_values(ascending=True).plot(kind='barh')
plt.title('Linear Regression')
plt.subplot(3,1,2)
lasso_coef_LR.groupby('Features')['Coefficients'].sum().sort_values(ascending=True).plot(kind='barh')
plt.title('Lasso')
plt.subplot(3,1,3)
ridge_coef_LR.groupby('Features')['Coefficients'].sum().sort_values(ascending=True).plot(kind='barh')
plt.title('Ridge')
plt.show()
```



Key Insights from Linear Regression

```
- Upon conducting regression analysis, it's evident that **CGPA** emerges as the **most influential feature** in predicting admission chance.

- Additionally, **GRE and TOEFL** scores also exhibit significant importance in the predictive model

- Here’s a concise bullet-point summary of your findings:

- Initial regression model through **OLS** revealed **University Rating** and **SOP** as non-relevant features.
- **Multicollinearity Check**:
  - VIF scores consistently below **5**, indicating low multicollinearity among predictors.
- **Residual Analysis**:
  - **Residuals do not follow a normal distribution**.
  - Presence of **Homoscedasticity** in residual plots.
- **Regularized Models**:
  - **Ridge and Lasso regression** results were comparable to the Linear Regression Model.
- Overall, the features demonstrated strong predictive capabilities.
```

Recommendations

- **Feature Enhancement**:

- Focus Areas: Encourage students to prioritize improving their GRE scores,Cumulative Grade Point Average (CGPA),TOEFL and the quality of Letters of Recommendation (LOR). These three factors have been identified as having a significant impact on admission chances, and enhancing them can greatly improve overall application competitiveness.

- **Data Augmentation**:

  - Holistic Profiles: Advocate for the collection of a broader range of data that goes beyond traditional academic metrics. This should include extracurricular achievements, personal statements, and diversity factors. By capturing a more comprehensive view of applicants' backgrounds and experiences, admissions committees can better assess the overall potential of each candidate.

- **Additional Features**:

  - Correlation Insights: Given the strong correlation among CGPA, it would be beneficial to enrich the predictive model with a variety of diverse features. These may include:

    -- Research Experience: Highlighting involvement in research projects or publications can showcase an applicant's commitment and analytical skills.

    -- Work Experience: Relevant professional experiences can demonstrate practical application of knowledge and skills acquired during academic training.

    -- Internships: Participation in internships provides valuable real-world exposure, making candidates more attractive to admissions committees.

    -- Extracurricular Activities: Involvement in clubs, sports, or volunteer work reflects a well-rounded individual and can be a differentiating factor in applications.

In [ ]:

```
1
```