Jamboree Case Study - Linear Regression

Problem Statement

Jamboree recently launched a feature where students/learners can come to their website and check their probability of getting into the IVY league college. This feature estimates the chances of graduate admission from an Indian perspective.

Additional View

- Lin Reg. will also help predict one's chances of admission given the rest of the variables.
- GRE Score, TOEFL Score & CGPA are most important attributes as per Indian Perspective.

Installing Dependencies

In [709]:

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

Loading Dataset

In [710]:

```
jamboree = pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/0
```

In [711]:

```
jamboree.head(5)
```

Out[711]:

	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

In [712]:

```
jamboree.drop(["Serial No."], axis = 1, inplace = True)
```

In [713]:

```
jamboree.shape
```

Out[713]:

```
(500, 8)
```

In [714]:

```
jamboree.dtypes
```

Out[714]:

```
GRE Score              int64
TOEFL Score            int64
University Rating      int64
SOP                  float64
LOR                  float64
CGPA                 float64
Research               int64
Chance of Admit      float64
dtype: object
```

- **All the features are numerical**

In [715]:

```
jamboree.isnull().sum()
```

Out[715]:

```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

- **There is no null values**

In [716]:

```
jamboree.describe().T
```

Out[716]:

	count	mean	std	min	25%	50%	75%	max
GRE Score	500.0	316.47200	11.295148	290.00	308.0000	317.00	325.00	340.00
TOEFL Score	500.0	107.19200	6.081868	92.00	103.0000	107.00	112.00	120.00
University Rating	500.0	3.11400	1.143512	1.00	2.0000	3.00	4.00	5.00
SOP	500.0	3.37400	0.991004	1.00	2.5000	3.50	4.00	5.00
LOR	500.0	3.48400	0.925450	1.00	3.0000	3.50	4.00	5.00
CGPA	500.0	8.57644	0.604813	6.80	8.1275	8.56	9.04	9.92
Research	500.0	0.56000	0.496884	0.00	0.0000	1.00	1.00	1.00
Chance of Admit	500.0	0.72174	0.141140	0.34	0.6300	0.72	0.82	0.97

- **While Observing the mean and 50% percentile of data there is no significant difference observed**
- **We can conclude there are no outliers in the dataset.**

In [717]:

```
jamboree.duplicated().sum()
```

Out[717]:

```
0
```

- **There is no duplicated values in the dataset**

In [718]:

```
jamboree.columns = map(lambda x: x.strip(), jamboree.columns)
```

Non-Graphical Analysis

In [719]:

```
jamboree["University Rating"].value_counts(normalize=True)
```

Out[719]:

```
3    0.324
2    0.252
4    0.210
5    0.146
1    0.068
Name: University Rating, dtype: float64
```

- **While observing the university rating. Most of universities average rated.**

In [720]:

```
jamboree["SOP"].value_counts(normalize=True)
```

Out[720]:

```
4.0    0.178
3.5    0.176
3.0    0.160
2.5    0.128
4.5    0.126
2.0    0.086
5.0    0.084
1.5    0.050
1.0    0.012
Name: SOP, dtype: float64
```

In [721]:

```
jamboree["Research"].value_counts(normalize=True)
```

Out[721]:

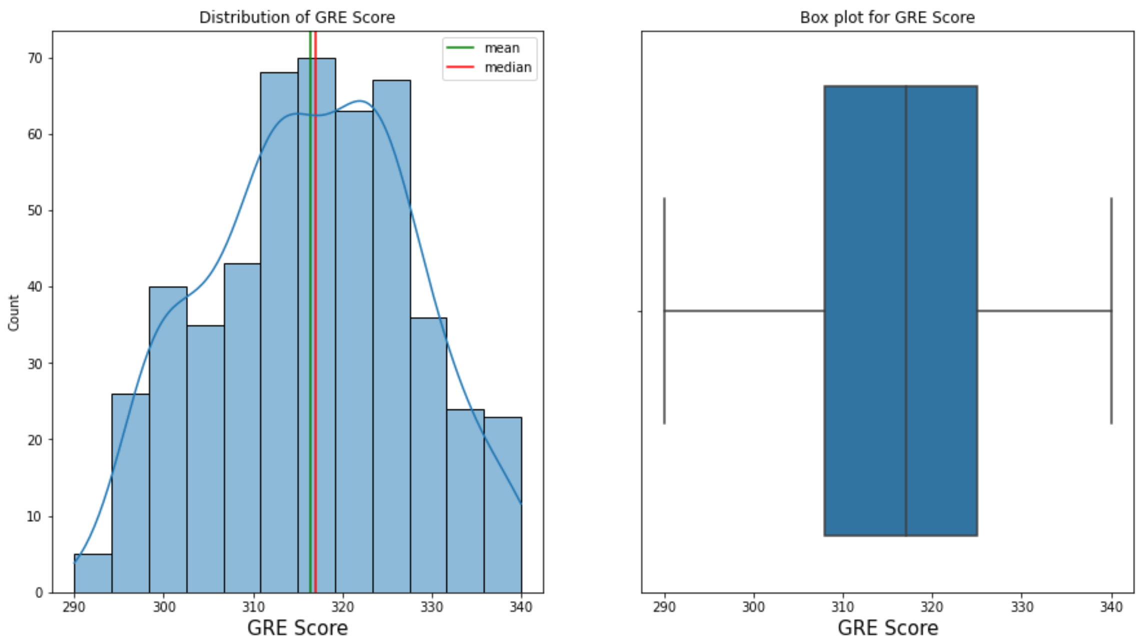
```
1    0.56
0    0.44
Name: Research, dtype: float64
```

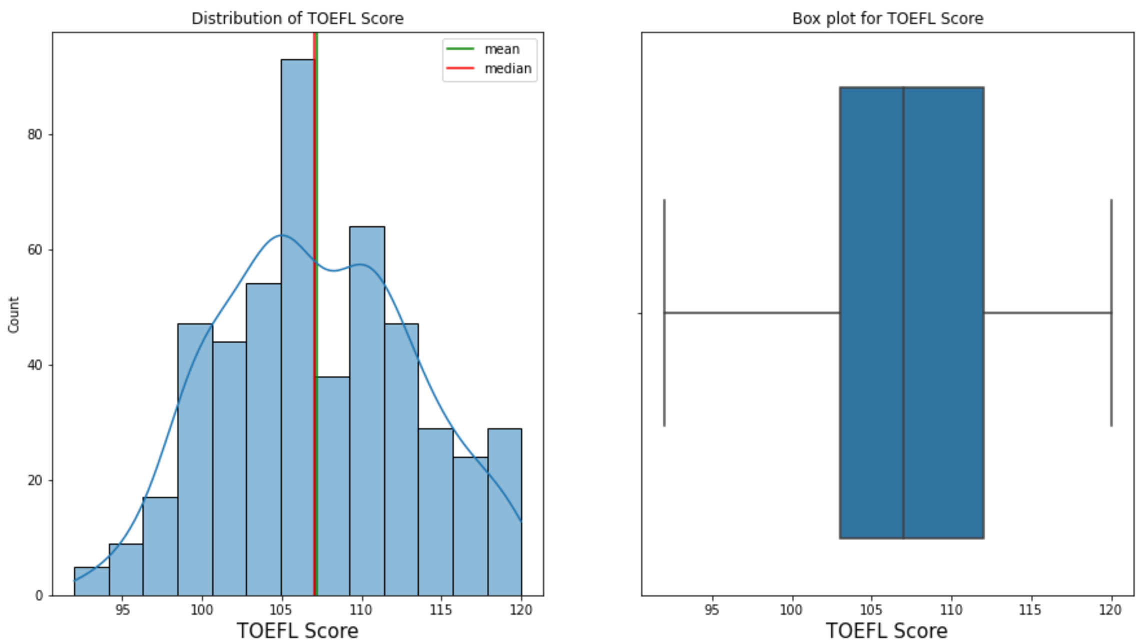
- **Above stats shows there are almost equal distribution among students who did research**

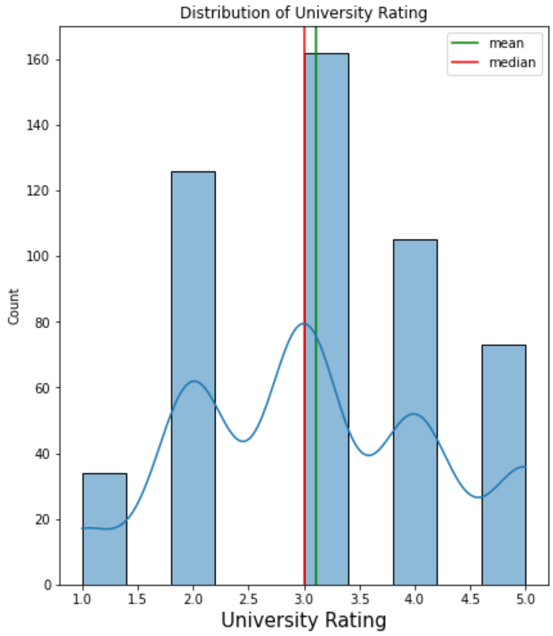
Graphical Analysis: Univariate

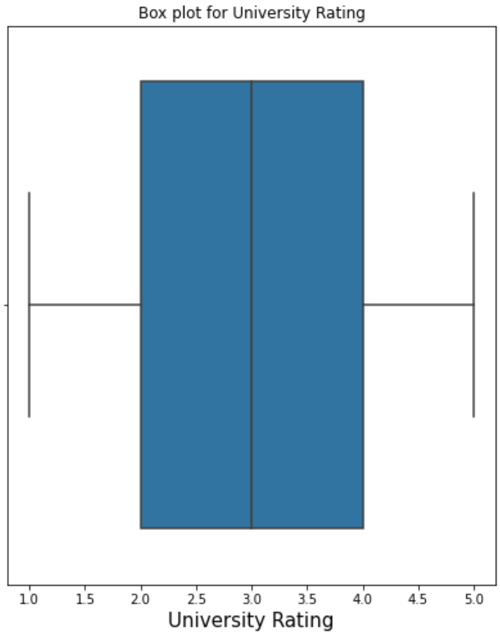
In [722]:

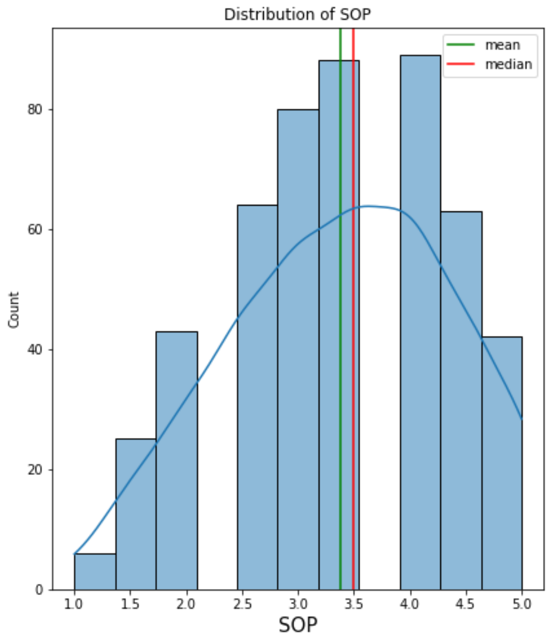
```
num_cat = ["GRE Score", "TOEFL Score", "University Rating", "SOP", "LOR", "CGPA", "Resea

for i in range(len(num_cat)):
    fig = plt.figure(figsize = (15, 8))

    ax1 = plt.subplot2grid((1, 2), (0, 0))
    ax1.set_title(f"Distribution of {num_cat[i]}")
    ax1.set_xlabel(ax1.get_xlabel(), fontsize = 15)
    ax1.axvline(jamboree[num_cat[i]].mean(),color="green", label = "mean")
    ax1.axvline(jamboree[num_cat[i]].median(),color="red", label = "median")
    ax1.legend(loc = "best")
    sns.histplot(data=jamboree, x=num_cat[i], ax=ax1, kde=True)

    ax2 = plt.subplot2grid((1, 2), (0, 1))
    ax2.set_title(f"Box plot for {num_cat[i]}")
    ax2.set_xlabel(ax1.get_xlabel(), fontsize = 15)
    sns.boxplot(data = jamboree, x=num_cat[i], ax=ax2)

plt.show()
```

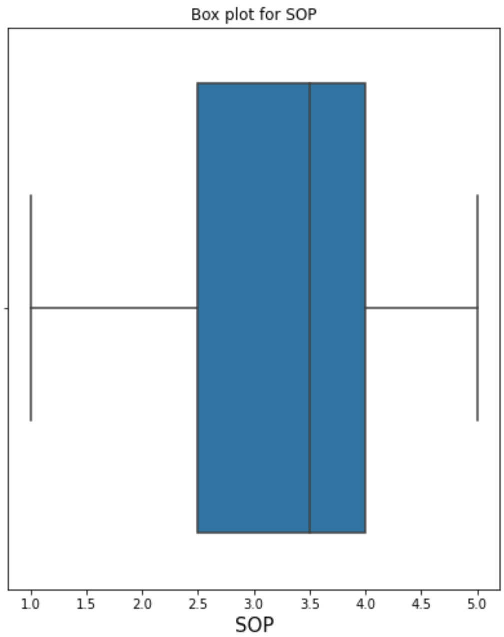


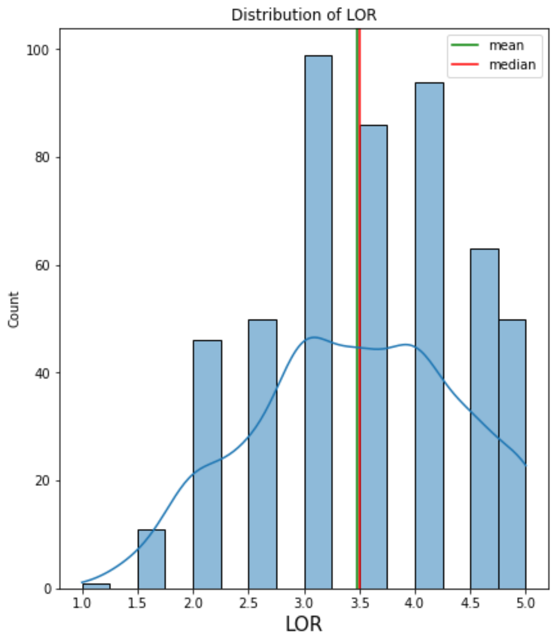


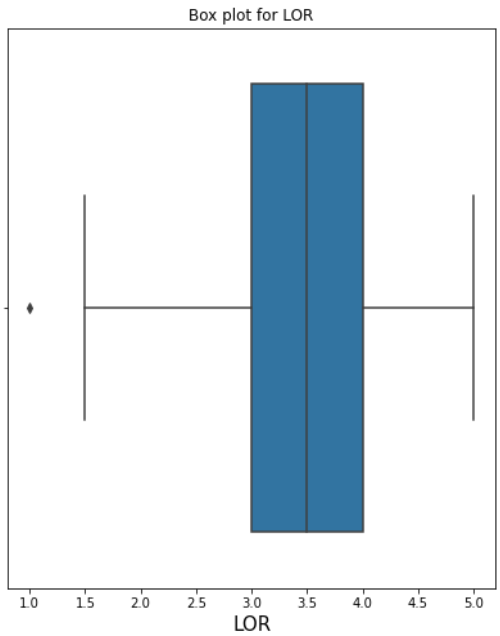

Distribution of University Rating
mean
median
Count
University Rating

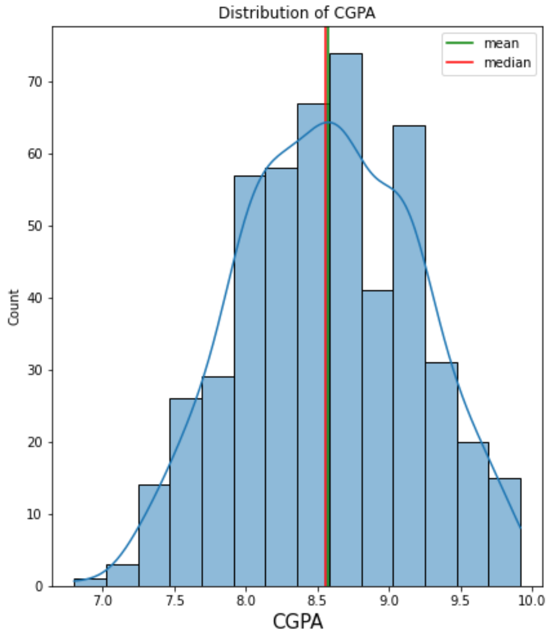

Box plot for University Rating
University Rating

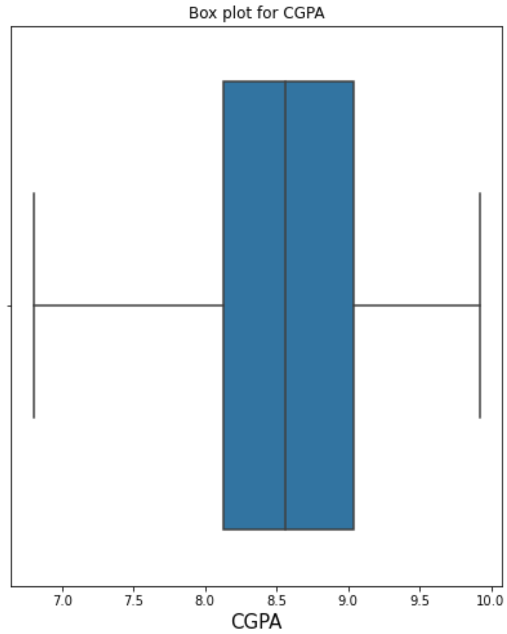

Distribution of SOP
mean
median
Count
SOP

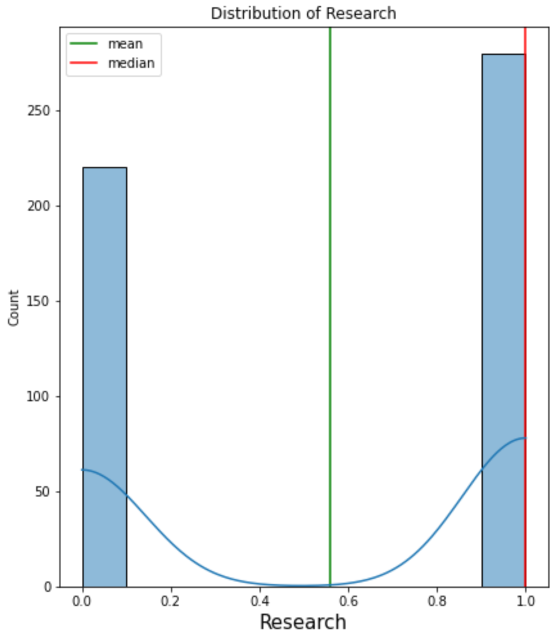

Box plot for SOP
SOP

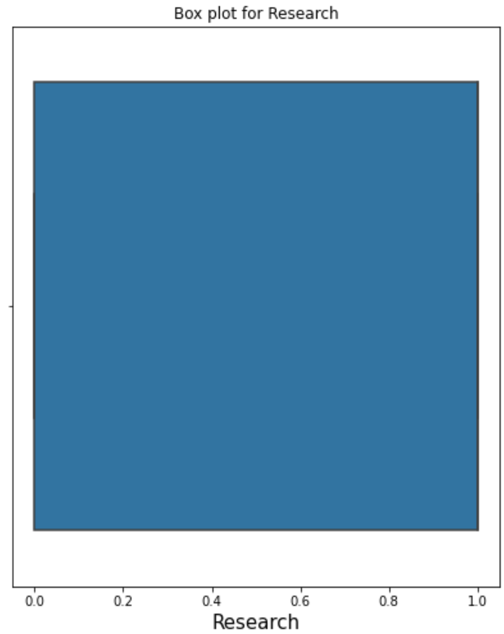

Distribution of LOR
mean
median
Count
LOR

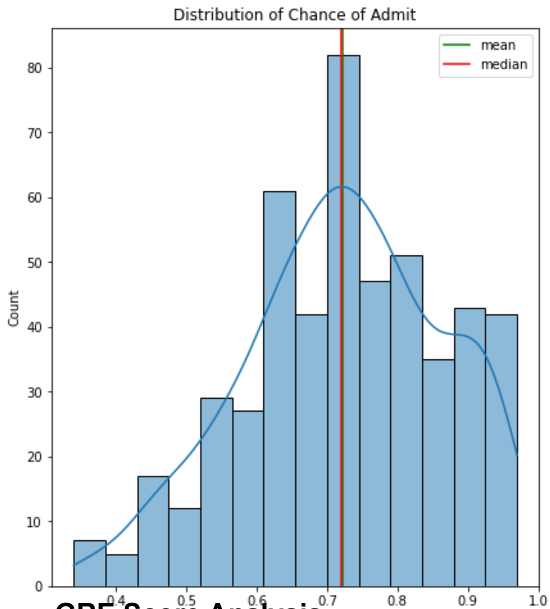

Box plot for LOR
LOR

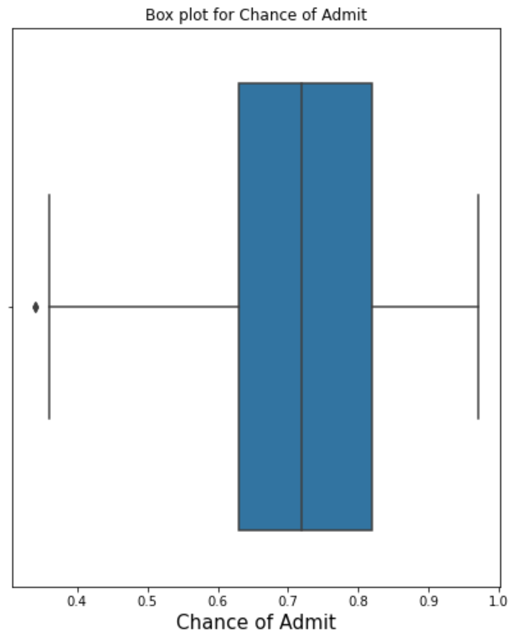












- **GRE Score Analysis**
  - Distribution of GRE resembles like Gaussian
  - Mean of GRE Score is approx 315

    - There is no outliers detected as mean and median overlaps
- **TOEFL Score Analysis**
    - Distribution of TOEFL somewhat resembles like Gaussian
    - Mean of TOEFL Score is approx 108
    - There is no outliers detected as mean and median overlaps
- **LOR Analysis**
    - Most of the students gets 3.5 out 5
- **CGPA Analysis**
    - Distribution of CGPA resembles like Gaussian
    - Mean of CGPA Score is approx 8.5
    - There is no outliers detected as mean and median overlaps
- **Chance of Admit**
    - Mean of chance of admission is 0.72

Outliers Detection

- **From the above observation, There is no outliers detected in the dataset**

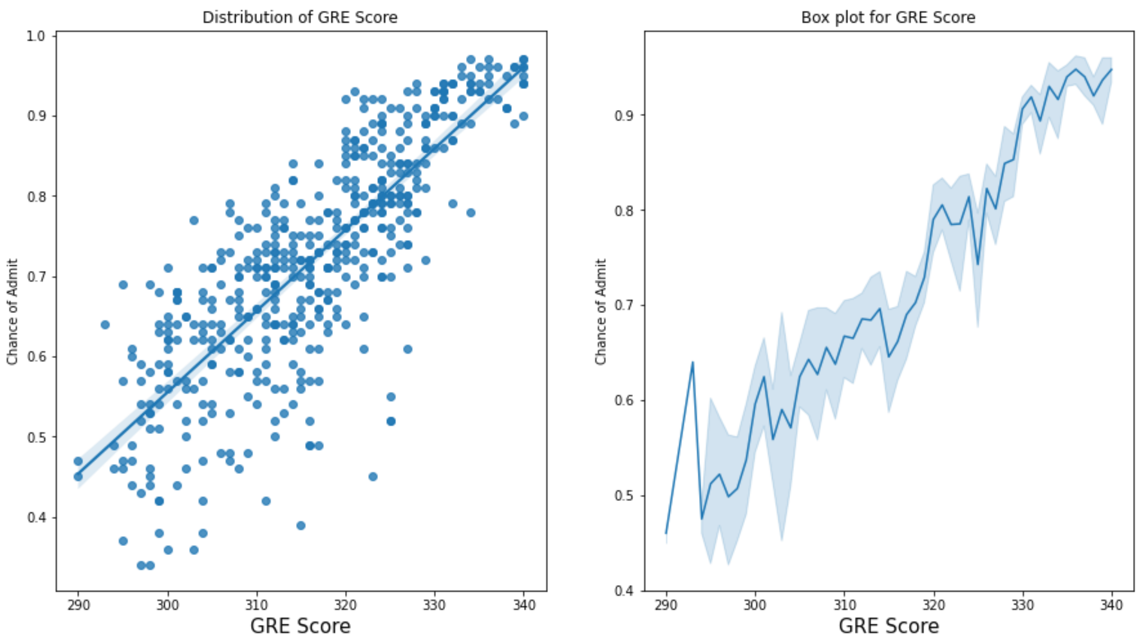
In [ ]:

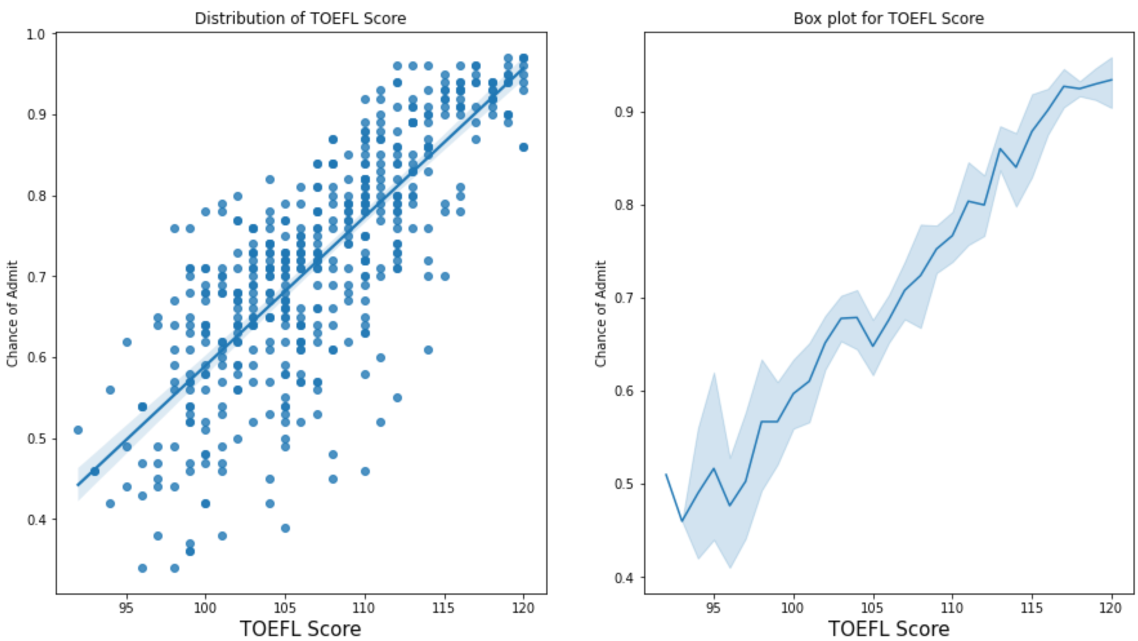
```

```

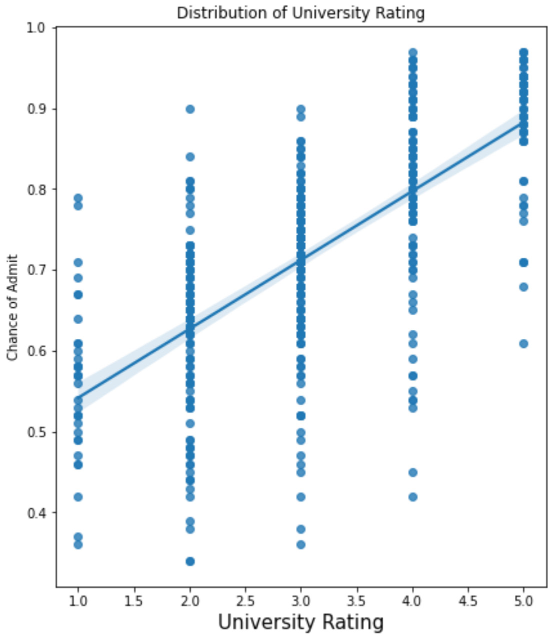
Graphical Analysis: Bivariate

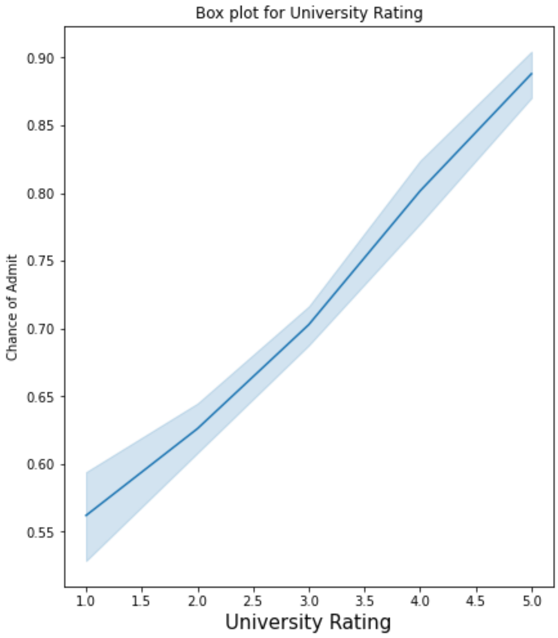
In [723]:

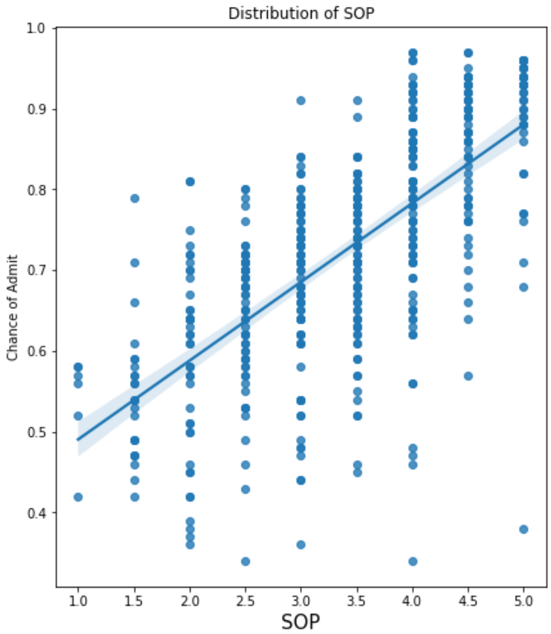
```
num_cat = ["GRE Score", "TOEFL Score", "University Rating", "SOP", "LOR", "CGPA"]
target_cat = "Chance of Admit"

for i in range(len(num_cat)):
    fig = plt.figure(figsize = (15, 8))

    ax1 = plt.subplot2grid((1, 2), (0, 0))
    ax1.set_title(f"Distribution of {num_cat[i]}")
    ax1.set_xlabel(ax1.get_xlabel(), fontsize = 15)
#     sns.histplot(data=jamboree, x=num_cat[i], ax=ax1, kde=True)
    sns.regplot(data=jamboree, ax=ax1, x=num_cat[i], y=target_cat)

    ax2 = plt.subplot2grid((1, 2), (0, 1))
    ax2.set_title(f"Box plot for {num_cat[i]}")
    ax2.set_xlabel(ax1.get_xlabel(), fontsize = 15)
#     sns.boxplot(data = jamboree, x=num_cat[i], ax=ax2)
    sns.lineplot(data=jamboree, x=num_cat[i], y=target_cat, ax=ax2)

plt.show()
```

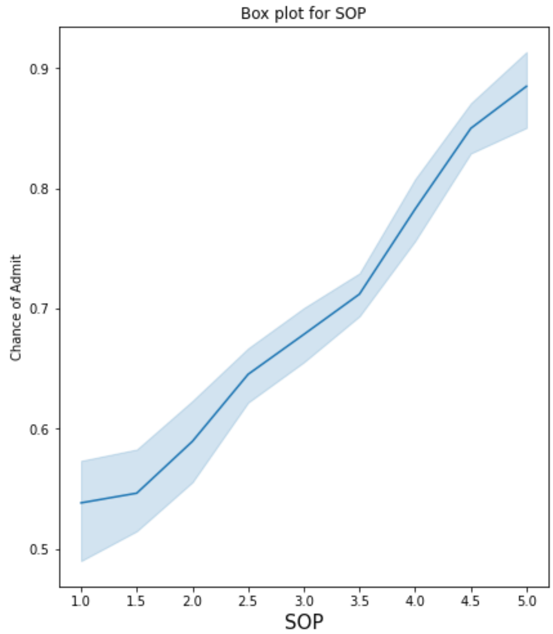


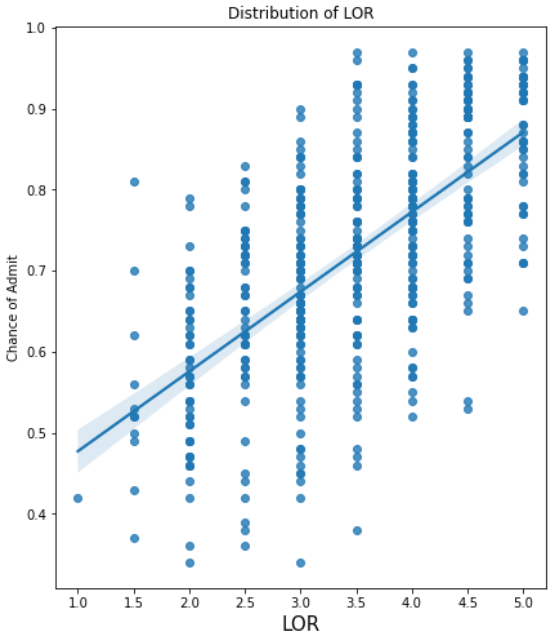


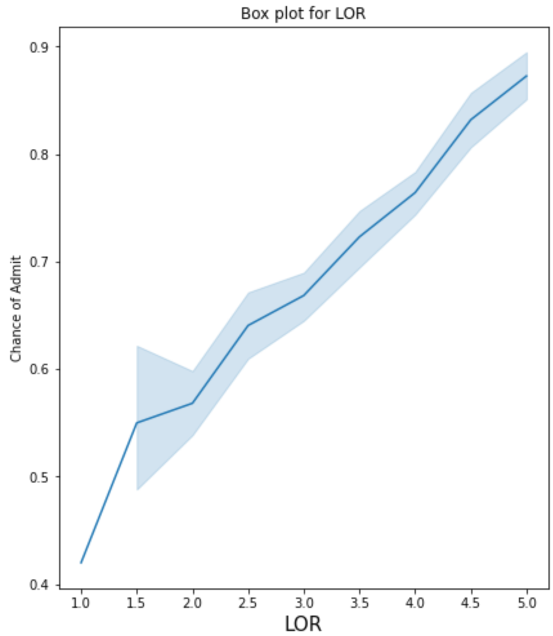

Distribution of University Rating
Chance of Admit
University Rating

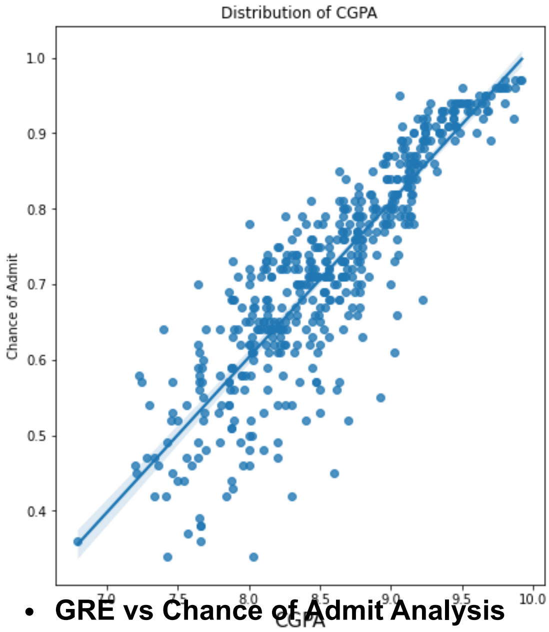

Box plot for University Rating
Chance of Admit
University Rating

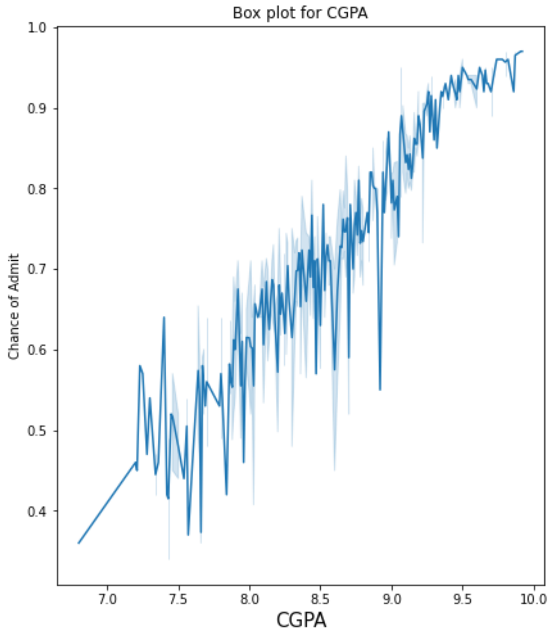

Distribution of SOP
Chance of Admit
SOP


Box plot for SOP
Chance of Admit
SOP


Distribution of LOR
Chance of Admit
LOR


Box plot for LOR
Chance of Admit
LOR

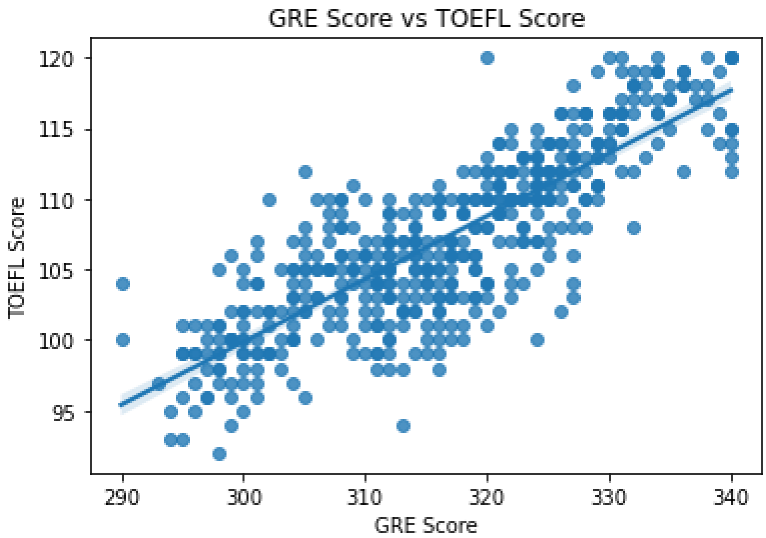


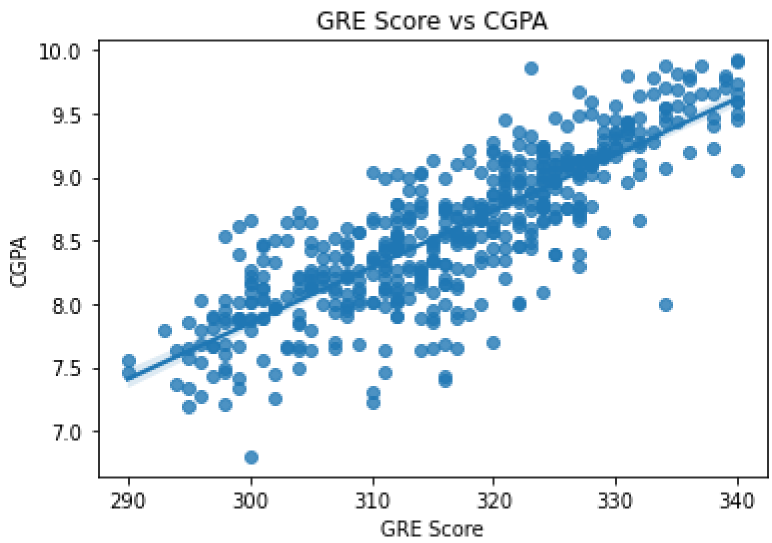


- **GRE vs Chance of Admit Analysis**
    - There is linear relationship between GRE and Chance of Admission
    - Higher the GRE -> Higher the chance of admission
- **TOEFL vs Chance of Admit Analysis**
    - There is linear relationship between TOEFL and Chance of Admission
    - Higher the TOEFL -> Higher the chance of admission
- **LOR / SOP / University Rating vs Chance of Admit Analysis**
    - There is no significant linear relationship between TOEFL and Chance of Admission
- **CGPA vs Chance of Admit Analysis**
    - There is linear relationship between TOEFL and Chance of Admission
    - Higher the CGPA -> Higher the chance of admission

In [724]:

```
fig = sns.regplot(x="GRE Score",y="TOEFL Score",data=jamboree)
plt.title("GRE Score vs TOEFL Score")
plt.show()

fig = sns.regplot(x="GRE Score",y="CGPA",data=jamboree)
plt.title("GRE Score vs CGPA")
plt.show()
```

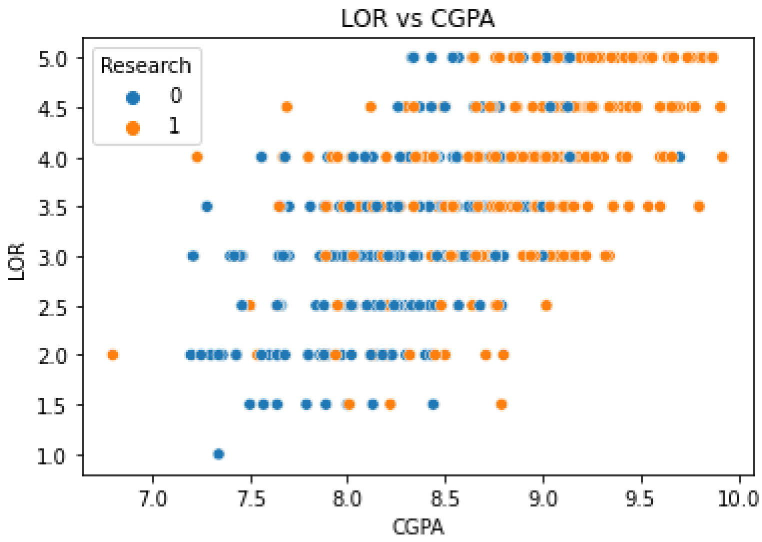




- **People with higher GRE Scores also have higher TOEFL Scores which is justified because both TOEFL and GRE have a verbal section which although not similar are relatable**
- **Although there are exceptions, people with higher CGPA usually have higher GRE scores maybe because they are smart or hard working**

In [725]:

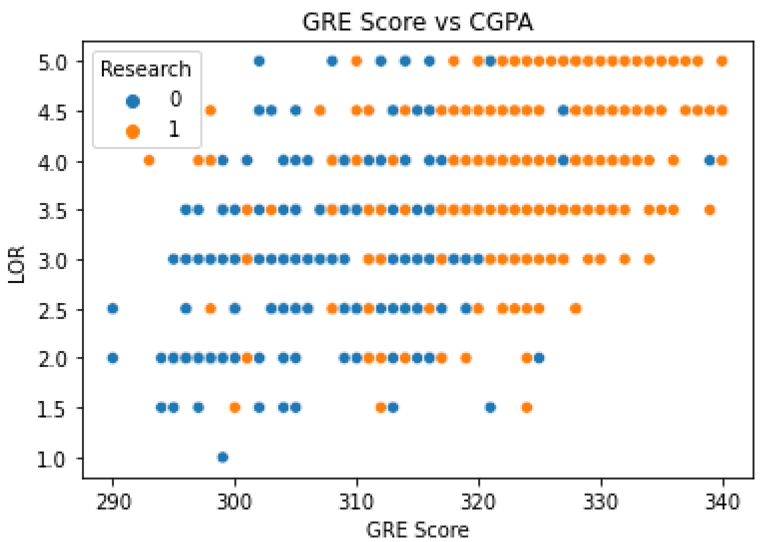
```
fig = sns.scatterplot(x="CGPA", y="LOR", data=jamboree, hue="Research")
plt.title("LOR vs CGPA")
plt.show()
```



- **LORs are not that related with CGPA so it is clear that a persons LOR is not dependent on that persons academic excellence.**
- **Having research experience is usually related with a good LOR which might be justified by the fact that supervisors have personal interaction with the students performing research which usually results in good LORs**

In [726]:

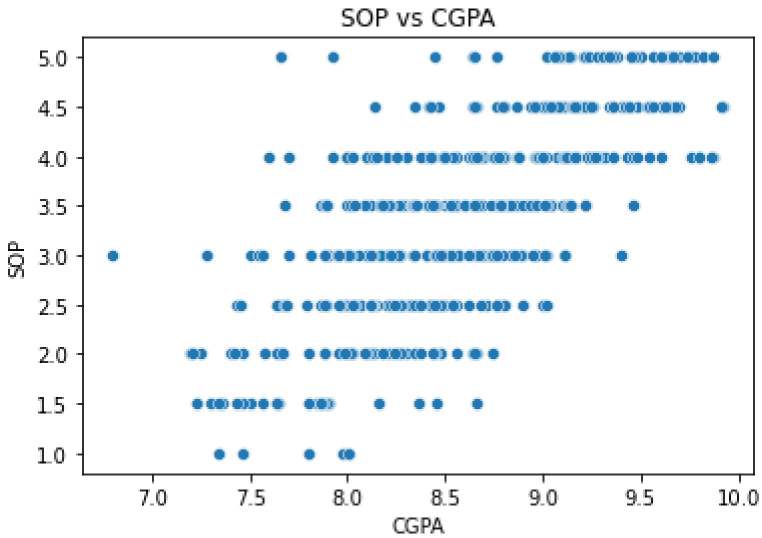
```
fig = sns.scatterplot(x="GRE Score", y="LOR", data=jamboree, hue="Research")
plt.title("GRE Score vs CGPA")
plt.show()
```

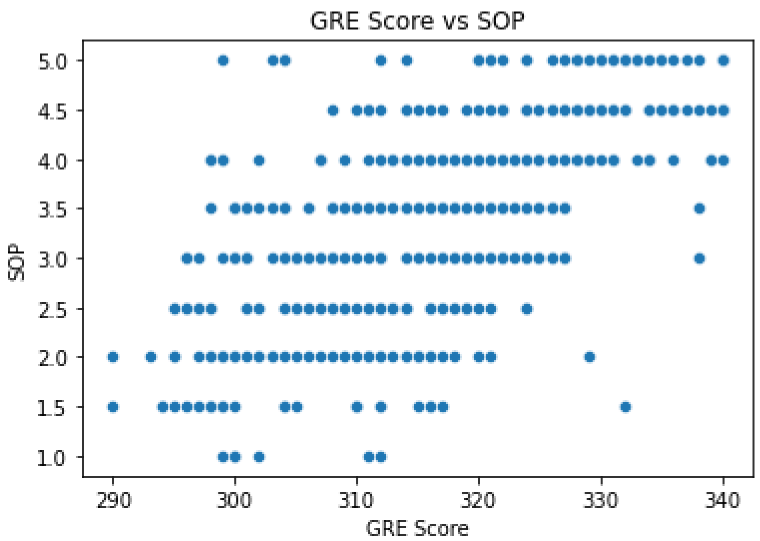


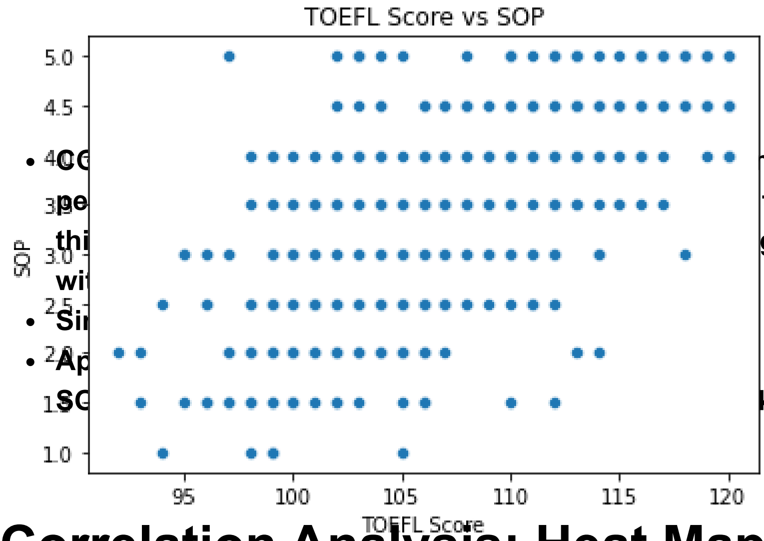
- **GRE scores and LORs are also not that related. People with different kinds of LORs have all kinds of GRE scores**

In [727]:

```
fig = sns.scatterplot(x="CGPA", y="SOP", data=jamboree)
plt.title("SOP vs CGPA")
plt.show()

fig = sns.scatterplot(x="GRE Score", y="SOP", data=jamboree)
plt.title("GRE Score vs SOP")
plt.show()

fig = sns.scatterplot(x="TOEFL Score", y="SOP", data=jamboree)
plt.title("TOEFL Score vs SOP")
plt.show()
```





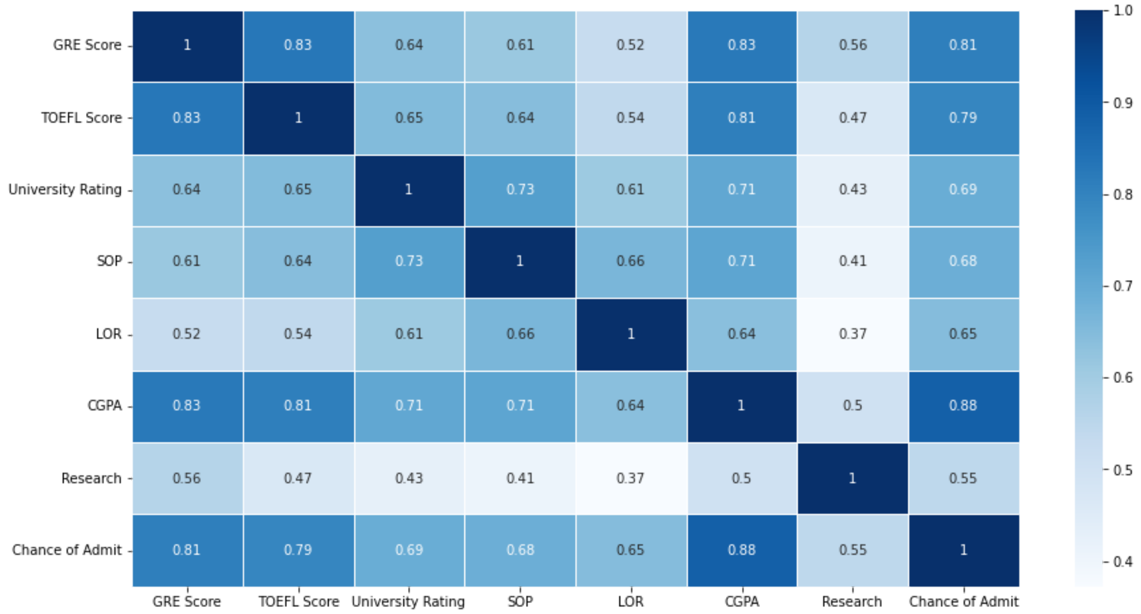


- **CG... nt of Purpose is related to academic pe... to be more hard working so they have good thi... ght move towards higher CGPA as along wi...**
- **Si...**
- **Ap... kinds of TOEFL Score. So the quality of SO... kills.**

Correlation Analysis: Heat Map

In [728]:

```
fig = plt.figure(figsize = (15, 8))
corr = jamboree.corr()
sns.heatmap(corr, linewidths=.5, annot=True, cmap="Blues")
plt.show()
```



- **High Correlation**

    1. GRE Score vs TOEFL Score
    2. CGPA vs TOEFL Score
    3. CGPA vs GRE Score
    4. Chance of Admit vs CGPA
    5. GRE Score vs Chance of Admit

Data Preprocessing

Duplicate Value Check

In [729]:

```
np.any(jamboree.duplicated())
```

Out[729]:

```
False
```

Missing Value Check

In [730]:

```
jamboree.isnull().sum()
```

Out[730]:

```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

Outlier Check

In [731]:

```
for i in data.columns:
    print("======" * 10)
    print("Mean of {}: ".format(i), jamboree[i].mean())
    print("Median of {}: ".format(i), jamboree[i].median())
```

```
============================================================
Mean of GRE Score:  316.472
Median of GRE Score:  317.0
============================================================
Mean of TOEFL Score:  107.192
Median of TOEFL Score:  107.0
============================================================
Mean of University Rating:  3.114
Median of University Rating:  3.0
============================================================
Mean of SOP:  3.374
Median of SOP:  3.5
============================================================
Mean of LOR:  3.484
Median of LOR:  3.5
============================================================
Mean of CGPA:  8.576440000000003
Median of CGPA:  8.56
============================================================
Mean of Research:  0.56
Median of Research:  1.0
============================================================
Mean of Chance of Admit:  0.7217399999999996
Median of Chance of Admit:  0.72
```

- **No outliers detected. As each and every feature overlaps its mean and median**

Feature Engineering & Data Modelling

In [732]:

```
from sklearn.model_selection import train_test_split

X = jamboree.drop(['Chance of Admit'], axis=1)
y = jamboree['Chance of Admit']

print("X shape: {}".format(X.shape))
print("y shape: {}".format(y.shape))
```

```
X shape: (500, 7)
y shape: (500,)
```

Train & Test Split

In [733]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)

print("X_train shape: {}".format(X_train.shape))
print("X_test shape: {}".format(X_test.shape))
print("y_train shape: {}".format(y_train.shape))
print("y_test shape: {}".format(y_test.shape))
```

```
X_train shape: (400, 7)
X_test shape: (100, 7)
y_train shape: (400,)
y_test shape: (100,)
```

In [734]:

```
X_train.head(5)
```

Out[734]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
312	311	107	4	4.5	4.5	9.00	1
32	338	118	4	3.0	4.5	9.40	1
194	316	109	3	3.5	3.0	8.76	0
490	307	105	2	2.5	4.5	8.12	1
470	320	110	5	4.0	4.0	9.27	1

In [735]:

```
y_train
```

Out[735]:

```
312    0.78
32     0.91
194    0.77
490    0.67
470    0.87
       ...
145    0.81
492    0.53
59     0.42
339    0.81
88     0.64
Name: Chance of Admit, Length: 400, dtype: float64
```

Feature standardization

In [736]:

```
# Standarization
from sklearn.preprocessing import StandardScaler
X_train_columns=X_train.columns
std=StandardScaler()
X_train_std=std.fit_transform(X_train)
```

In [737]:

```
X_train_std
```

Out[737]:

```
array([[-0.48601649, -0.0276501 ,  0.79650712, ...,  1.11439835,
         0.695962  ,  0.86413245],
       [ 1.87625988,  1.761474  ,  0.79650712, ...,  1.11439835,
         1.36072183,  0.86413245],
       [-0.0485579 ,  0.29764519, -0.08361124, ..., -0.53047007,
         0.2971061 , -1.15723001],
       ...,
       [-0.48601649, -0.51559304, -0.96372961, ..., -1.62704901,
        -0.46736771, -1.15723001],
       [ 0.65137584, -0.0276501 ,  1.67662549, ...,  0.56610888,
         0.13091614,  0.86413245],
       [-0.22354134,  0.13499755, -0.08361124, ...,  0.01781941,
        -0.73327164, -1.15723001]])
```

In [738]:

```
X_train = pd.DataFrame(data = X_train_std, columns = X_train_columns)
X_train.head(5)
```

Out[738]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	-0.486016	-0.027650	0.796507	1.119664	1.114398	0.695962	0.864132
1	1.876260	1.761474	0.796507	-0.379886	1.114398	1.360722	0.864132
2	-0.048558	0.297645	-0.083611	0.119964	-0.530470	0.297106	-1.157230
3	-0.835983	-0.352945	-0.963730	-0.879736	1.114398	-0.766510	0.864132
4	0.301409	0.460293	1.676625	0.619814	0.566109	1.144675	0.864132

Model Building: Lin Reg, Lasso Reg, Ridge Reg

In [739]:

```
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import mean_squared_error
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm

lm = LinearRegression()
lm.fit(X_train.values, y_train)
predictions = lm.predict(std.transform(X_test))
print("RMSE Linear Regression: ", np.sqrt(mean_squared_error(y_test, predictions)))
list(zip(X_train.columns, lm.coef_))
```

```
RMSE Linear Regression:  0.06568274378853765
```

Out[739]:

```
[('GRE Score', 0.02279984761710703),
 ('TOEFL Score', 0.018155915077019393),
 ('University Rating', 0.008078145789796546),
 ('SOP', 0.0009158722244926622),
 ('LOR', 0.012516168309954182),
 ('CGPA', 0.07159838907779793),
 ('Research', 0.009193614982627534)]
```

In [740]:

```
lm = Lasso()
lm.fit(X_train.values, y_train)
predictions = lm.predict(std.transform(X_test))
print("RMSE Lasso Regression: ", np.sqrt(mean_squared_error(y_test, predictions)))
list(zip(X_train.columns, lm.coef_))
```

```
RMSE Lasso Regression:  0.1448401877933055
```

Out[740]:

```
[('GRE Score', 0.0),
 ('TOEFL Score', 0.0),
 ('University Rating', 0.0),
 ('SOP', 0.0),
 ('LOR', 0.0),
 ('CGPA', 0.0),
 ('Research', 0.0)]
```

In [741]:

```
lm = Ridge()
lm.fit(X_train.values, y_train)
predictions = lm.predict(std.transform(X_test))
print("RMSE Lasso Regression: ", np.sqrt(mean_squared_error(y_test, predictions)))
list(zip(X_train.columns, lm.coef_))
```

```
RMSE Lasso Regression:  0.0656776692949077
```

Out[741]:

```
[('GRE Score', 0.022981409017396162),
 ('TOEFL Score', 0.018297984338214893),
 ('University Rating', 0.008146094998667114),
 ('SOP', 0.0010764691675055765),
 ('LOR', 0.01256553212079061),
 ('CGPA', 0.07094749270353783),
 ('Research', 0.009223428627942506)]
```

Model Summary using stats model library

In [742]:

```
def build_model(X,y):
    X = sm.add_constant(X) #Adding the constant
    lm = sm.OLS(y.values,X).fit() # fitting the model
    print(lm.summary()) # model summary
    return X

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

Model 1

In [743]:

```
X_train_new = build_model(X_train, y_train)
```

```
                            OLS Regression Results                            
==============================================================================
====
Dep. Variable:                      y   R-squared:                            
0.828
Model:                            OLS   Adj. R-squared:                       
0.825
Method:                 Least Squares   F-statistic:                         2
70.2
Date:                Thu, 24 Aug 2023   Prob (F-statistic):          1.01e
-145
Time:                        22:43:25   Log-Likelihood:                    57
1.18
No. Observations:                 400   AIC:                               -1
126.
Df Residuals:                     392   BIC:                               -1
094.
Df Model:                           7                                         
Covariance Type:            nonrobust                                         
==============================================================================
===========
                        coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----------
const                 0.7244      0.003    247.175      0.000       0.719
0.730
GRE Score             0.0228      0.006      3.676      0.000       0.011
0.035
TOEFL Score           0.0182      0.006      3.207      0.001       0.007
0.029
University Rating     0.0081      0.005      1.656      0.099      -0.002
0.018
SOP                   0.0009      0.005      0.186      0.853      -0.009
0.011
LOR                   0.0125      0.004      2.995      0.003       0.004
0.021
CGPA                  0.0716      0.006     11.183      0.000       0.059
0.084
Research              0.0092      0.004      2.551      0.011       0.002
0.016
==============================================================================
====
Omnibus:                       91.100   Durbin-Watson:                        
1.968
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                  21
6.020
Skew:                          -1.135   Prob(JB):                         1.24
e-47
Kurtosis:                       5.794   Cond. No.                             
5.69
==============================================================================
====

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

In [744]:

```
checkVIF(X_train_new)
```

Out[744]:

	Features	VIF
6	CGPA	4.77
1	GRE Score	4.48
2	TOEFL Score	3.73
4	SOP	2.82
3	University Rating	2.77
5	LOR	2.03
7	Research	1.51
0	const	1.00

Inference

- p-vale of SOP seems to be higher than the significance value of 0.05, hence dropping it as it is insignificant in presence of other variables.

In [745]:

```
X_train_new = X_train_new.drop(["SOP"], axis = 1)
```

Model 2

In [746]:

```
X_train_new = build_model(X_train_new, y_train)
```

```
                            OLS Regression Results                            
==============================================================================
====
Dep. Variable:                      y   R-squared:                       
0.828
Model:                            OLS   Adj. R-squared:                  
0.826
Method:                 Least Squares   F-statistic:                     3
16.1
Date:                Thu, 24 Aug 2023   Prob (F-statistic):          5.55e
-147
Time:                        22:43:25   Log-Likelihood:                 57
1.16
No. Observations:                 400   AIC:                            -1
128.
Df Residuals:                     393   BIC:                            -1
100.
Df Model:                           6                                         
Covariance Type:            nonrobust                                         
==============================================================================
===========
                        coef    std err          t      P>|t|      [0.025      
0.975]
------------------------------------------------------------------------------
-----------
const                 0.7244      0.003    247.479      0.000       0.719      
0.730
GRE Score             0.0228      0.006      3.678      0.000       0.011      
0.035
TOEFL Score           0.0182      0.006      3.224      0.001       0.007      
0.029
University Rating     0.0084      0.004      1.893      0.059      -0.000      
0.017
LOR                   0.0127      0.004      3.162      0.002       0.005      
0.021
CGPA                  0.0718      0.006     11.427      0.000       0.059      
0.084
Research              0.0092      0.004      2.562      0.011       0.002      
0.016
==============================================================================
====
Omnibus:                       90.434   Durbin-Watson:                   
1.968
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              21
3.539
Skew:                          -1.129   Prob(JB):                     4.27
e-47
Kurtosis:                       5.778   Cond. No.                        
5.26
==============================================================================
====

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

In [747]:

```
checkVIF(X_train_new)
```

Out[747]:

	Features	VIF
5	CGPA	4.61
1	GRE Score	4.48
2	TOEFL Score	3.72
3	University Rating	2.32
4	LOR	1.89
6	Research	1.51
0	const	1.00

In [748]:

```
X_train_new = X_train_new.drop(["University Rating"], axis = 1)
```

Inference

- p-vale of University Rating seems to be higher than the significance value of 0.05, hence dropping it as it is insignificant in presence of other variables.

MultiCollinearity Check using VIF

In [749]:

```
checkVIF(X_train_new)
```

Out[749]:

	Features	VIF
1	GRE Score	4.47
4	CGPA	4.38
2	TOEFL Score	3.65
3	LOR	1.71
5	Research	1.50
0	const	1.00

Inferences

- VIF looks fine and hence, we can go ahead with the predictions

In [750]:

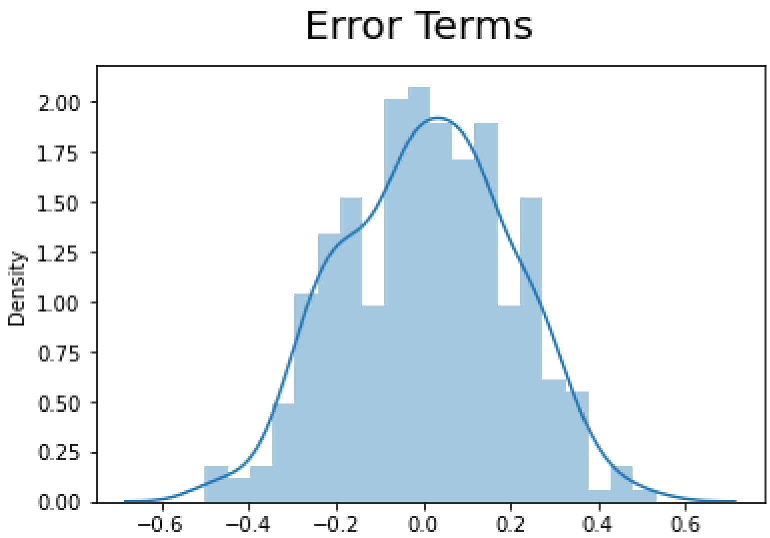
```
lm = sm.OLS(y_train.values,X_train_new).fit()
y_train_admit = lm.predict(X_train_new)
```

Residual Analysis of the Model

In [751]:

```
# Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((y_train - y_train_admit), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)     # Plot heading
plt.show()
```

```
C:\Users\91798\miniconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```



Inferences

- Error terms seem to be approximately normally distributed, so the assumption on the linear modeling seems to be fulfilled.

Mean of Residuals

In [752]:

```
residuals = (y_train - y_train_admit)
np.mean(residuals)
```

Out[752]:

```
0.008274902905840436
```

In [753]:

```
X_train_new = X_train_new.drop('const',axis=1)
```

Mean of Residuals: **-0.002773051612968889**

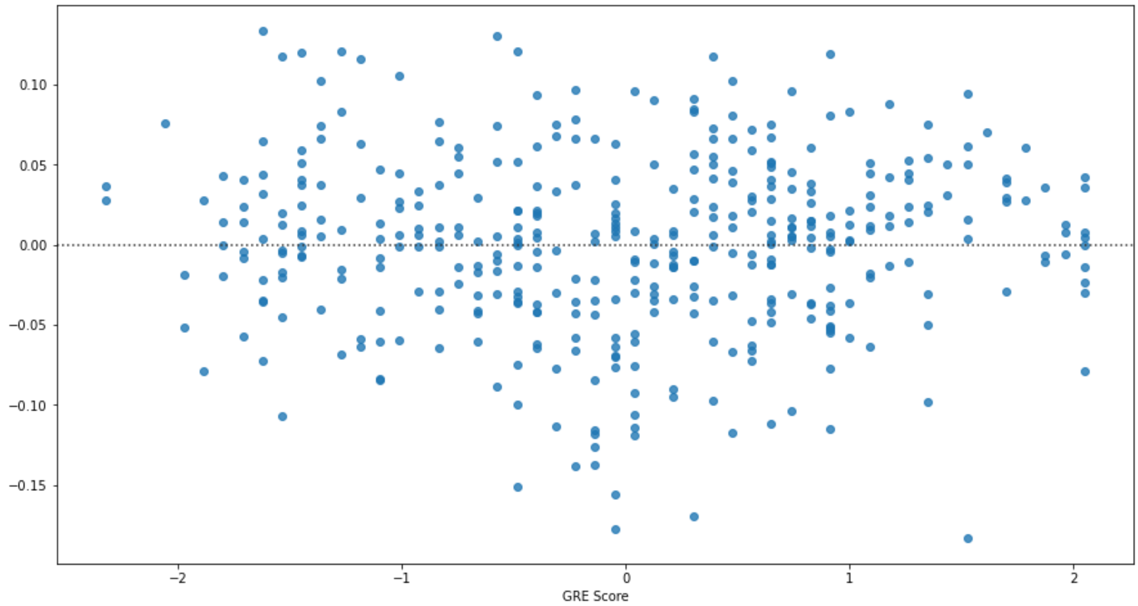
In [754]:

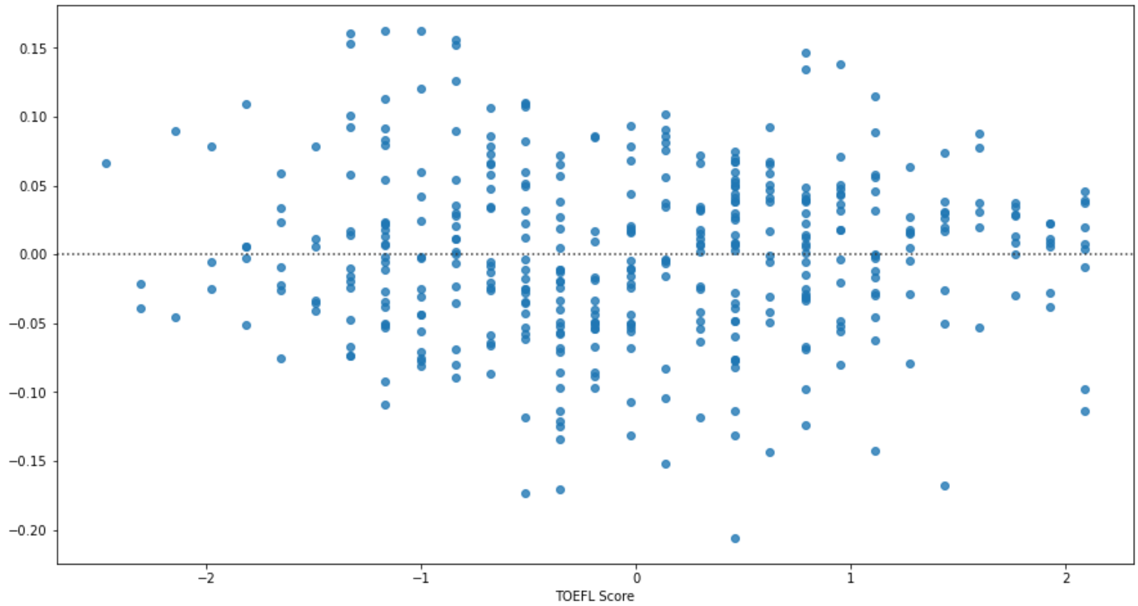
```
# Now let's use our model to make predictions.
# Creating X_test_new dataframe by dropping variables from X_test
X_test_new = X_test[X_train_new.columns]

# Adding a constant variable
X_test_new = sm.add_constant(std.fit_transform(X_test_new))
# Making predictions
y_pred = lm.predict(X_test_new)
```

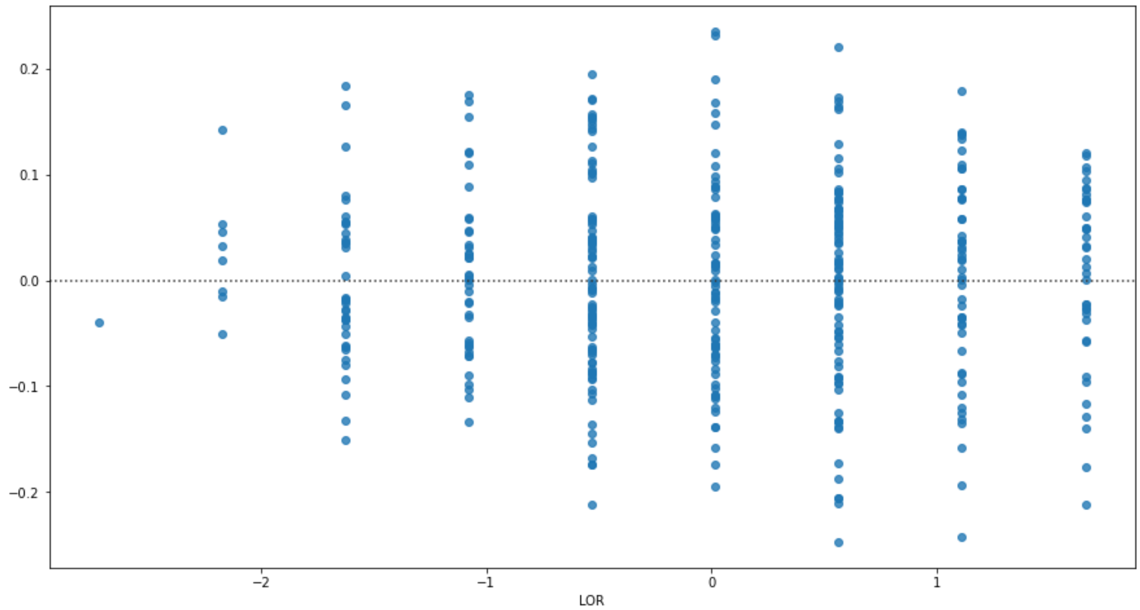
Linearity of Variables: Residual Plot

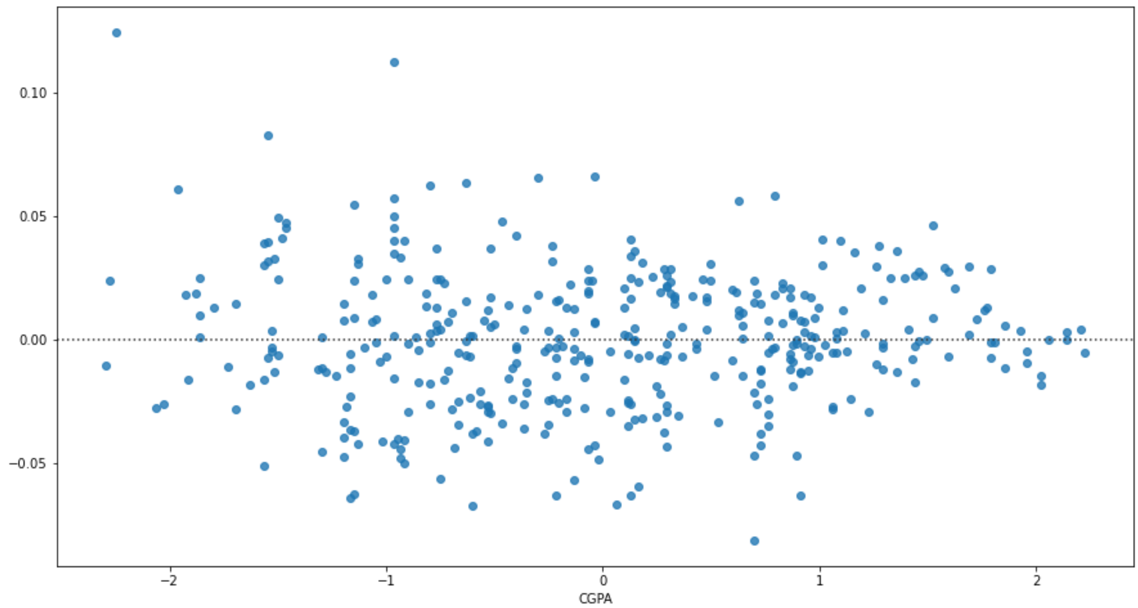
In [755]:

```
for i in X_train_new.columns:
    plt.figure(figsize = (15, 8))
    sns.residplot(x=X_train_new[i], y=y_train_admit)
```

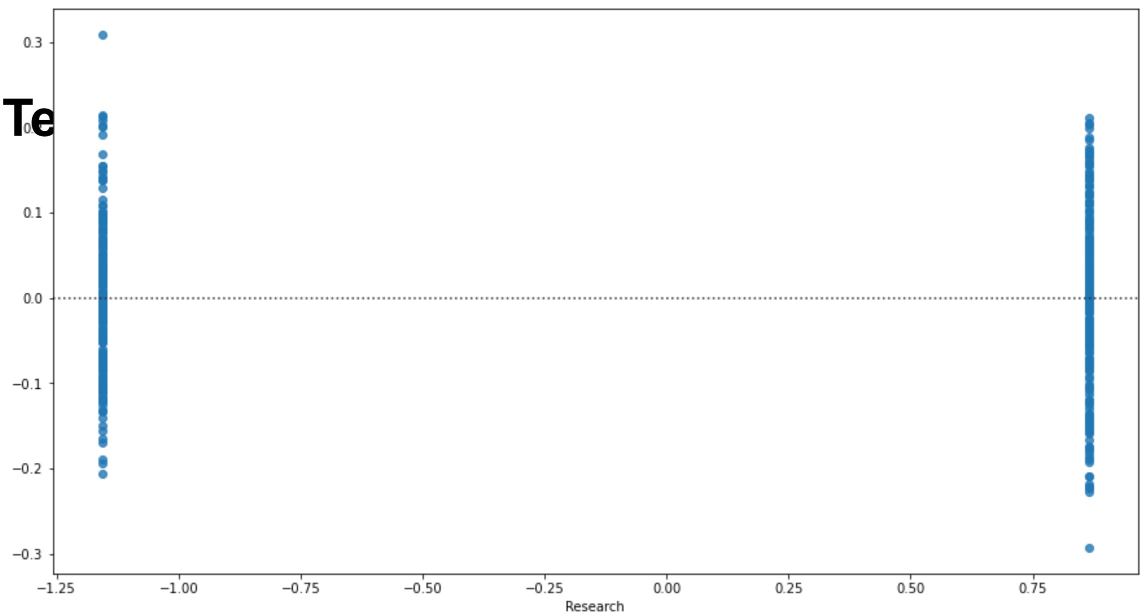





0.2
0.1
0.0
−0.1
−0.2
−2
−1
0
1
LOR


0.10
0.05
0.00
−0.05
−2
−1
0
1
2
CGPA

Te

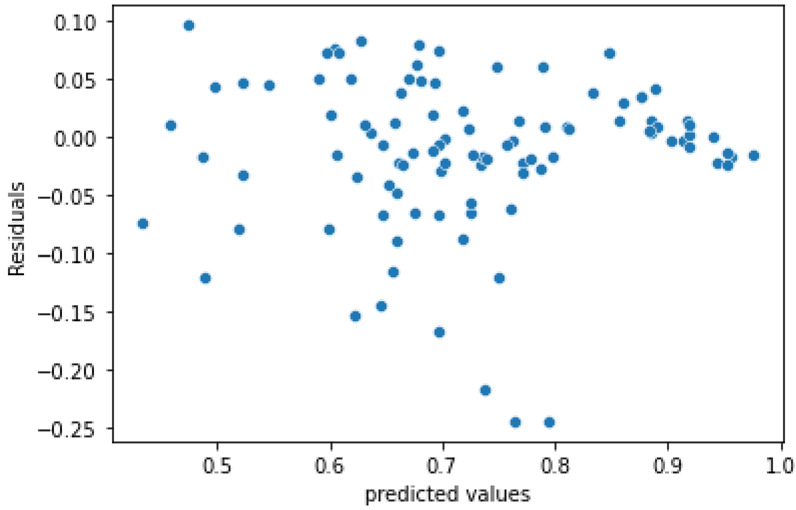

0.3
0.1
0.0
−0.1
−0.2
−0.3
−1.25
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
Research

In [756]:

```
residuals = y_test - y_pred
p = sns.scatterplot(x=y_pred,y=residuals)
plt.xlabel('predicted values')
plt.ylabel('Residuals')
# plt.ylim(-0.4,0.4)
# plt.xlim(0,1)
```

Out[756]:

```
Text(0, 0.5, 'Residuals')
```



In [757]:

```
import statsmodels.stats.api as sas
from statsmodels.compat import lzip
name=['F statistics','p-value']
test=sas.het_goldfeldquandt(residuals,X_test)
lzip(name,test)
```

Out[757]:

```
[('F statistics', 1.1495491734096537), ('p-value', 0.32487005690653364)]
```

Inferences

- Here null hypothesis is - error terms are homoscedastic and since p-values >0.05, we fail to reject the null hypothesis

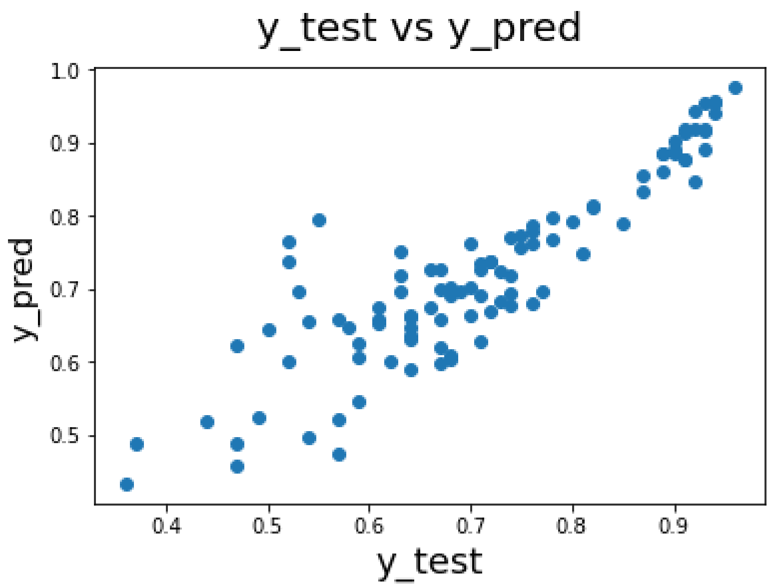
Normality of Residual

In [758]:

```
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test.values, y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20)                # Plot heading
plt.xlabel('y_test', fontsize=18)                            # X-label
plt.ylabel('y_pred', fontsize=16)
```

Out[758]:

```
Text(0, 0.5, 'y_pred')
```



Inferences

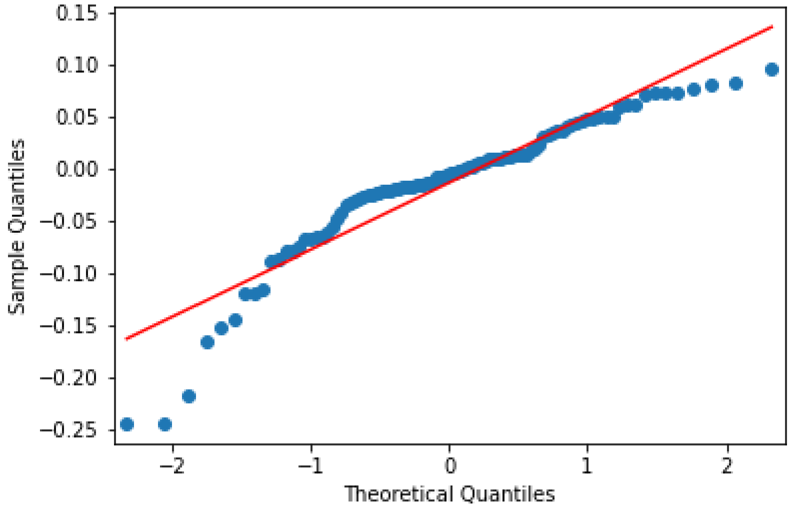
- y_test and y_pred overlaps for the most of the datapoints

In [ ]:

```
fig = plt.figure()
sns.distplot(residuals, bins = 20)
fig.suptitle('Distribution of Residuals', fontsize = 20)
plt.show()
```

In [760]:

```
import matplotlib.pyplot as plt

#create Q-Q plot with 45-degree line added to plot
fig = sm.qqplot(residuals, line = "r")
plt.show()
```



Inferences

- QQ Plots suggest majority of the data points fit the regression line.

Model performance evaluation

Metrics checked - MAE, RMSE, R2, Adj R2

In [761]:

```
from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_error

print("======" * 10)
print('Mean Absolute Error: ',mean_absolute_error(y_test.values,pred))
print("======" * 10)
print('Root Mean Square Error: ',np.sqrt(mean_squared_error(y_test.values,pred)))
print("======" * 10)
r2Score = r2_score(y_test, y_pred)
print('R2 Score: ', r2Score)
print("======" * 10)
aR2Score = 1 - (1-r2Score/(len(y_test)-X_test_new.shape[1]-1))
print('Adjusted. R2 Score: ', r2Score)
print("======" * 10)
```

```
============================================================
Mean Absolute Error:  0.6962010480812433
============================================================
Root Mean Square Error:  0.7210195623986372
============================================================
R2 Score:  0.7867445264177775
============================================================
Adjusted. R2 Score:  0.7867445264177775
============================================================
```

Performance test Train & Test Dataset

In [762]:

```
print("======" * 10)
Trainr2Score = r2_score(y_train, y_train_admit)
print('Train R2 Score: ', Trainr2Score)
print("======" * 10)
Testr2Score = r2_score(y_test, y_pred)
print('Test R2 Score: ', Testr2Score)
```

```
============================================================
Train R2 Score:  0.8267706543222355
============================================================
Test R2 Score:  0.7867445264177775
```

In [763]:

```
print(lm.summary())
```

```
                            OLS Regression Results
==========================================================================
====
Dep. Variable:                      y   R-squared:
0.827
Model:                            OLS   Adj. R-squared:
0.825
Method:                 Least Squares   F-statistic:                     3
76.1
Date:                Thu, 24 Aug 2023   Prob (F-statistic):          1.61e
-147
Time:                        22:43:27   Log-Likelihood:                 56
9.34
No. Observations:                 400   AIC:                            -1
127.
Df Residuals:                     394   BIC:                            -1
103.
Df Model:                           5
Covariance Type:            nonrobust
==========================================================================
=====
                 coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------
-----
const          0.7244      0.003    246.672      0.000       0.719
0.730
GRE Score      0.0233      0.006      3.752      0.000       0.011
0.035
TOEFL Score    0.0197      0.006      3.509      0.001       0.009
0.031
LOR            0.0150      0.004      3.911      0.000       0.007
0.023
CGPA           0.0745      0.006     12.119      0.000       0.062
0.087
Research       0.0096      0.004      2.676      0.008       0.003
0.017
==========================================================================
====
Omnibus:                       86.890   Durbin-Watson:
1.980
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              20
1.597
Skew:                          -1.093   Prob(JB):                     1.67
e-44
Kurtosis:                       5.705   Cond. No.
4.82
==========================================================================
====

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

Actionable Insights and Recommendations

1. R-sqaured and Adjusted R-squared (extent of fit) - 0.83 and 0.82 - 85% variance explained.
2. F-stats and Prob(F-stats) (overall model fit) - 387.9 and 1.03e-149(approx. 0.0) - Model fit is significant and explained 82% variance is just not by chance.
3. p-values - p-values for all the coefficients seem to be less than the significance level of 0.05. - meaning that all the predictors are statistically significant.
4. There is lot of chance for the model improvement by tunining the parameters.
5. Currently this models attains accuracy around 80%. This can be improved further by doing some feature engg.
6. As the dataset is strictly provided for the Indian perspective. This model is not generalized, there is scope for the generalization of this model.
7. LogLikelihood is around 570 which indicates model is significantly fit.
8. Performance of trainning and test data is almost same indicates the model will work significantly on unseen data.
9. While observing the model and according to test assumptions - We can infer errors are homoscedasticity according to p-value
10. While observing the linearity of residual there is no significant pattern found which indicates the residual plots are not correlated
11. While observing the normality of residual - the distribution resembles like bell-shaped and the reg. line fits almost every point

In [ ]: