In [139]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(context="notebook", style = 'darkgrid' , color_codes=True)
from scipy import stats
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
import warnings
warnings.filterwarnings('ignore')
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error,mean_absol
ute_percentage_error
```

In [ ]:
```
```

In [140]:
```
df=pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/00
0/001/839/original/Jamboree_Admission.csv')
df.head()
```

Out[140]:

	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

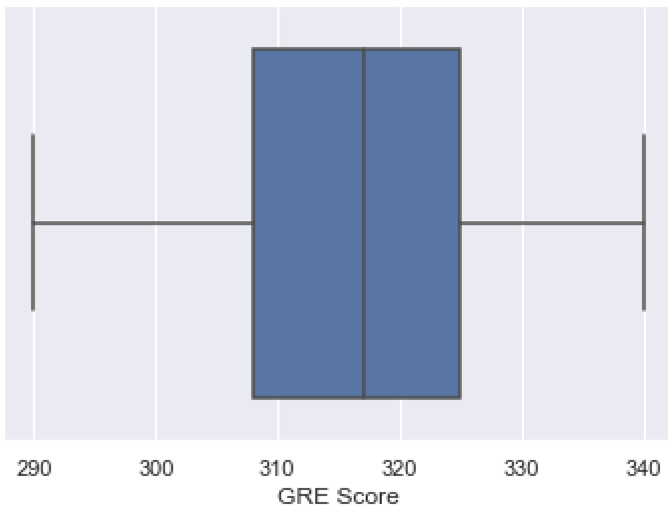
In [141]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

Univariate Analysis

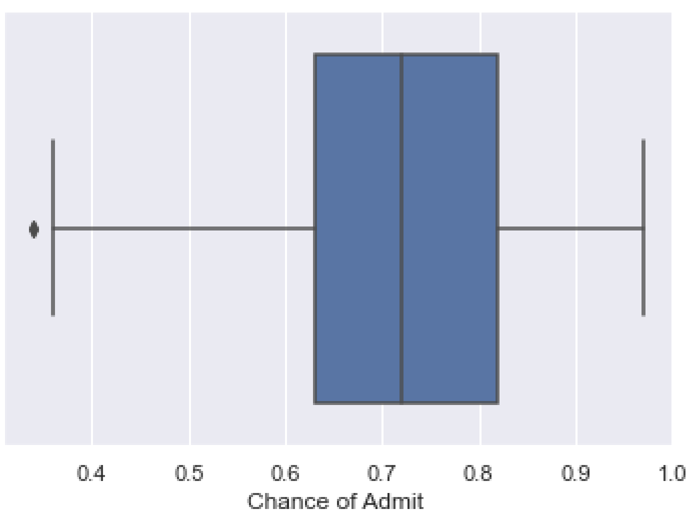
In [142]:
```
sns.boxplot(df['GRE Score'])
```

Out[142]: <matplotlib.axes._subplots.AxesSubplot at 0x1feb5874940>



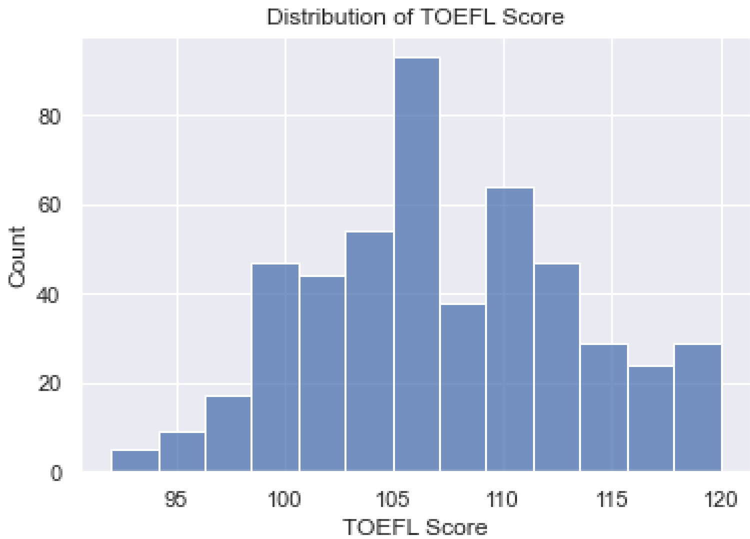
In [143]:
```
sns.boxplot(df['Chance of Admit '])
```

Out[143]: <matplotlib.axes._subplots.AxesSubplot at 0x1feb5999ac0>



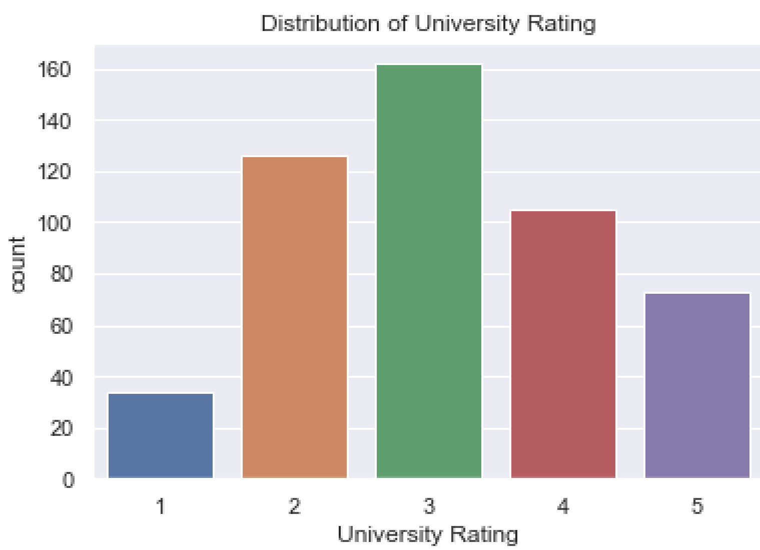
In [144]:
```
sns.histplot(df['TOEFL Score'])
plt.title('Distribution of TOEFL Score')
```

Out[144]: Text(0.5, 1.0, 'Distribution of TOEFL Score')



In [145]:
```
sns.countplot(x='University Rating',data=df)
plt.title('Distribution of University Rating')
```

Out[145]: Text(0.5, 1.0, 'Distribution of University Rating')

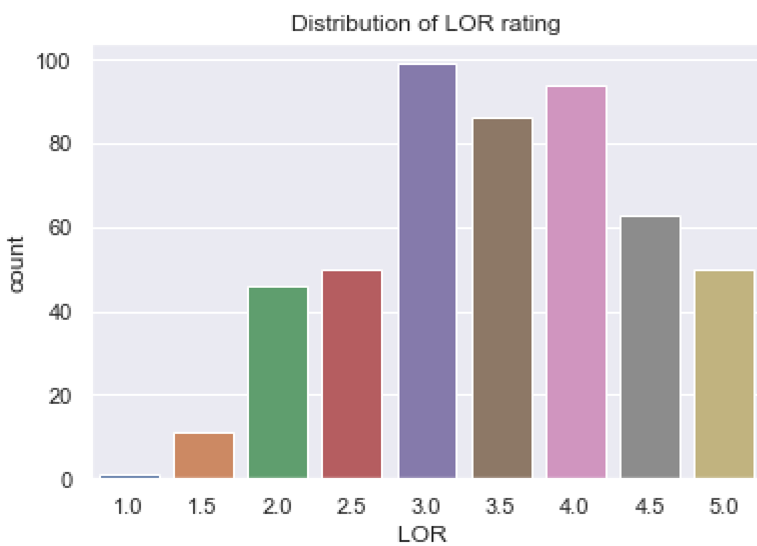


In [146]:
```
sns.countplot(x='SOP',data=df)
plt.title('Distribution of SOP Rating')
```

Out[146]: Text(0.5, 1.0, 'Distribution of SOP Rating')

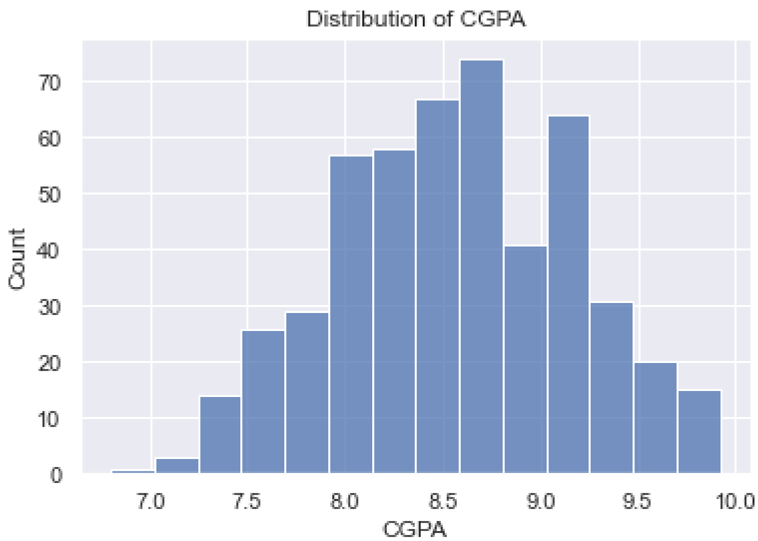
In [147]:
```
sns.countplot(x='LOR ',data=df)
plt.title('Distribution of LOR rating')
```

Out[147]: Text(0.5, 1.0, 'Distribution of LOR rating')



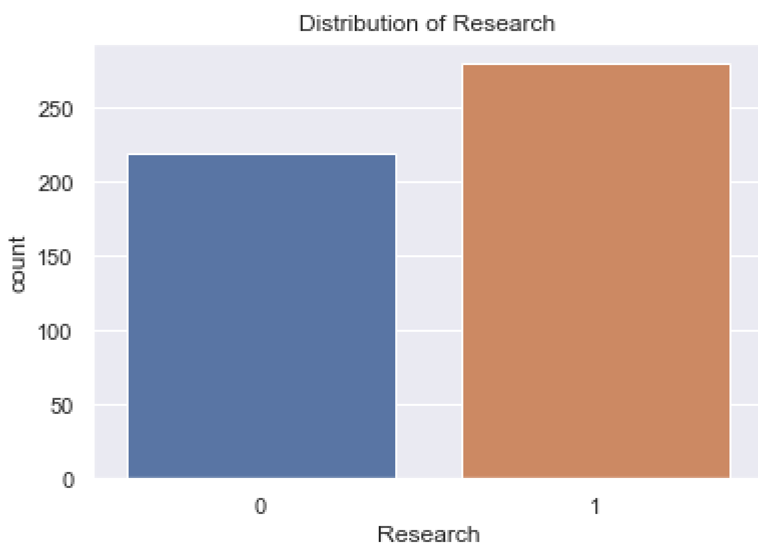
In [148]:
```python
sns.histplot(df['CGPA'])
plt.title('Distribution of CGPA')
```

Out[148]:
```
Text(0.5, 1.0, 'Distribution of CGPA')
```



In [149]:
```python
sns.countplot(x='Research',data=df)
plt.title('Distribution of Research')
```

Out[149]:
```
Text(0.5, 1.0, 'Distribution of Research')
```

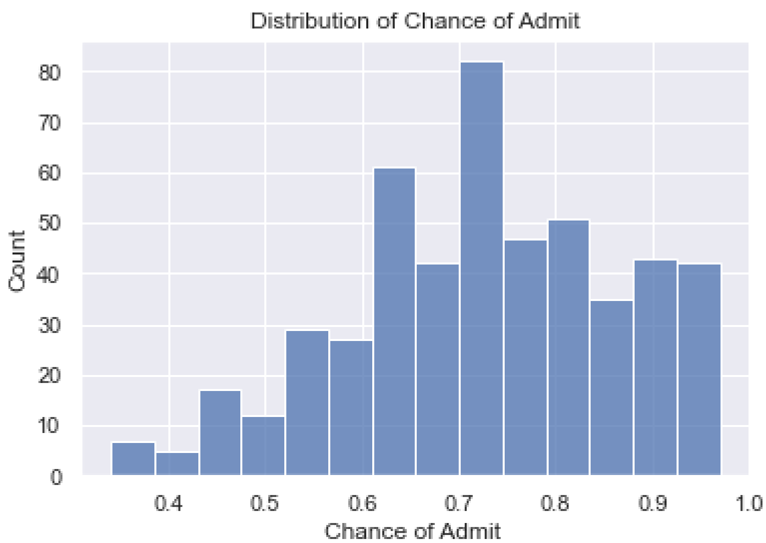


In [150]:
```python
df['Research'].value_counts()
```

Out[150]:
```
1    280
0    220
Name: Research, dtype: int64
```

In [151]:
```
sns.histplot(df['Chance of Admit '])
plt.title('Distribution of Chance of Admit')
```

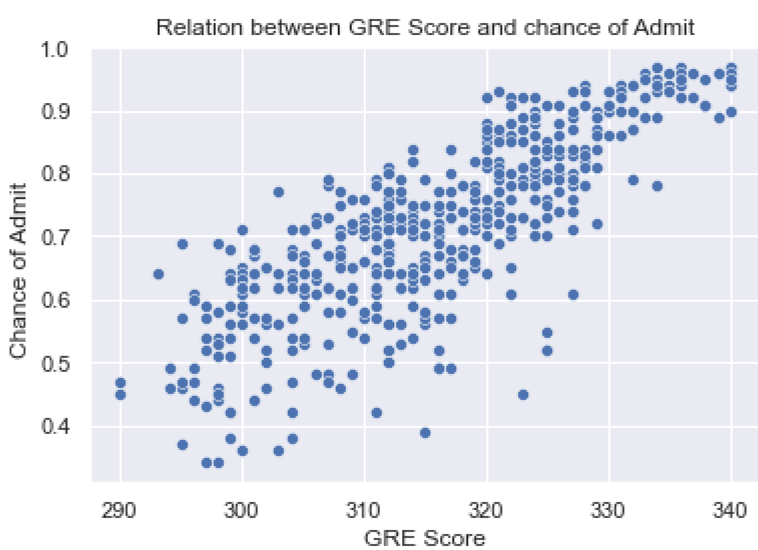
Out[151]: Text(0.5, 1.0, 'Distribution of Chance of Admit')



Bivariate Analysis

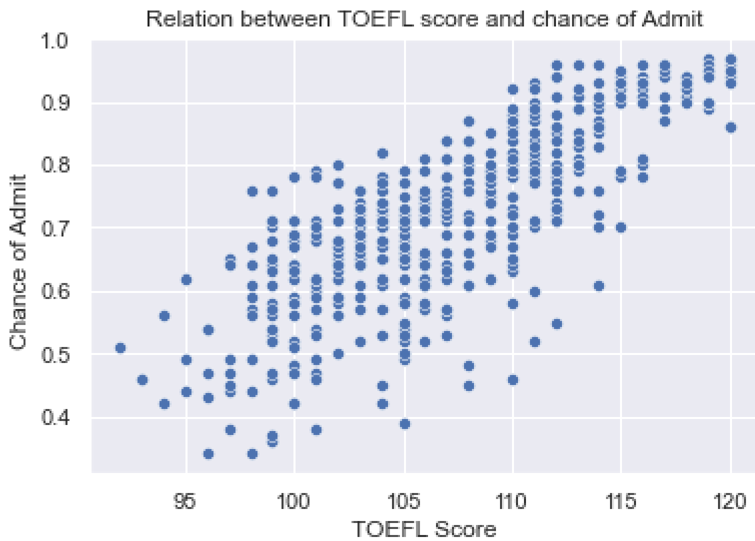
In [152]:
```
sns.scatterplot(x='GRE Score',y='Chance of Admit ',data=df)
plt.title('Relation between GRE Score and chance of Admit')
```

Out[152]: Text(0.5, 1.0, 'Relation between GRE Score and chance of Admit')



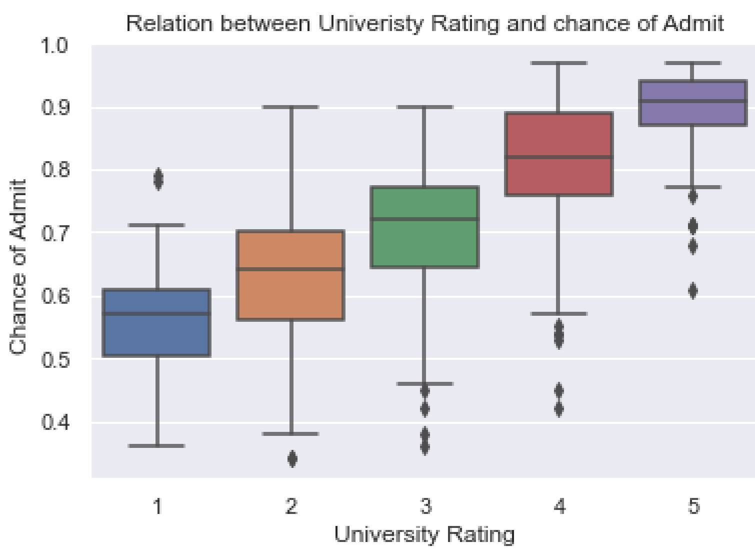
In [153]:
```
sns.scatterplot(x='TOEFL Score',y='Chance of Admit ',data=df)
plt.title('Relation between TOEFL score and chance of Admit')
```

Out[153]: Text(0.5, 1.0, 'Relation between TOEFL score and chance of Admit')

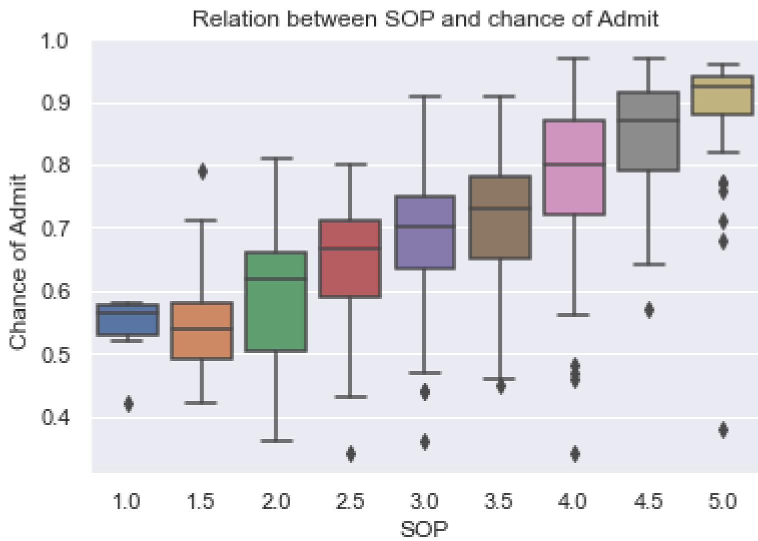


In [154]:
```
sns.boxplot(x='University Rating',y='Chance of Admit ',data=df)
plt.title('Relation between Univeristy Rating and chance of Admit')
```

Out[154]: Text(0.5, 1.0, 'Relation between Univeristy Rating and chance of Admit')

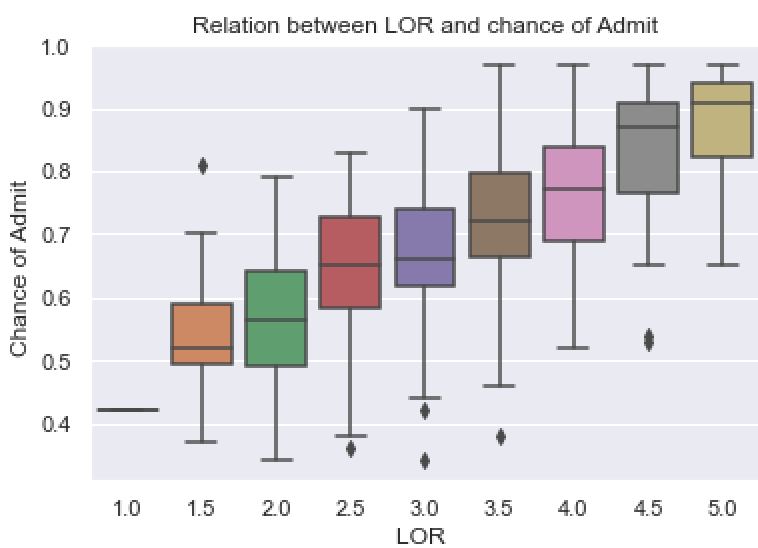


In [156]:
```
sns.boxplot(x='SOP',y='Chance of Admit ',data=df)
plt.title('Relation between SOP and chance of Admit')
```
Out[156]: Text(0.5, 1.0, 'Relation between SOP and chance of Admit')



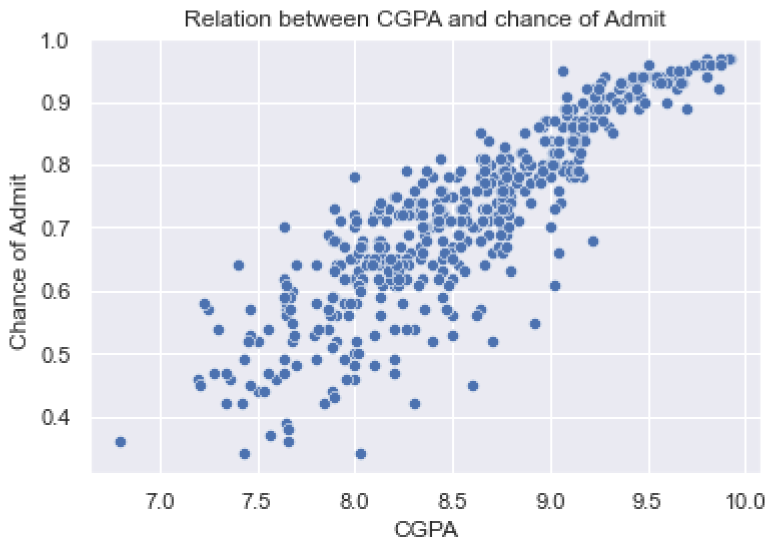
In [ ]:

In [157]:
```
sns.boxplot(x='LOR ',y='Chance of Admit ',data=df)
plt.title('Relation between LOR and chance of Admit')
```
Out[157]: Text(0.5, 1.0, 'Relation between LOR and chance of Admit')



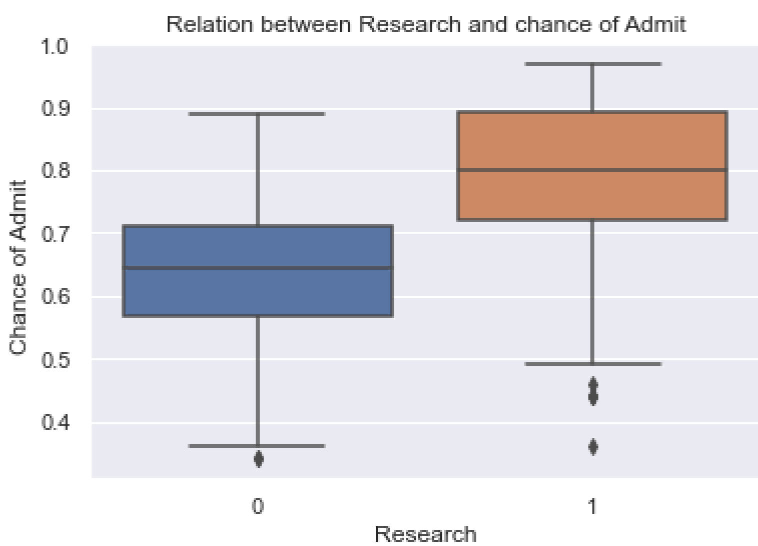
In [158]:
```
sns.scatterplot(x='CGPA',y='Chance of Admit ',data=df)
plt.title('Relation between CGPA and chance of Admit')
```

Out[158]: Text(0.5, 1.0, 'Relation between CGPA and chance of Admit')



In [159]:
```
sns.boxplot(x='Research',y='Chance of Admit ',data=df)
plt.title('Relation between Research and chance of Admit')
```

Out[159]: Text(0.5, 1.0, 'Relation between Research and chance of Admit')



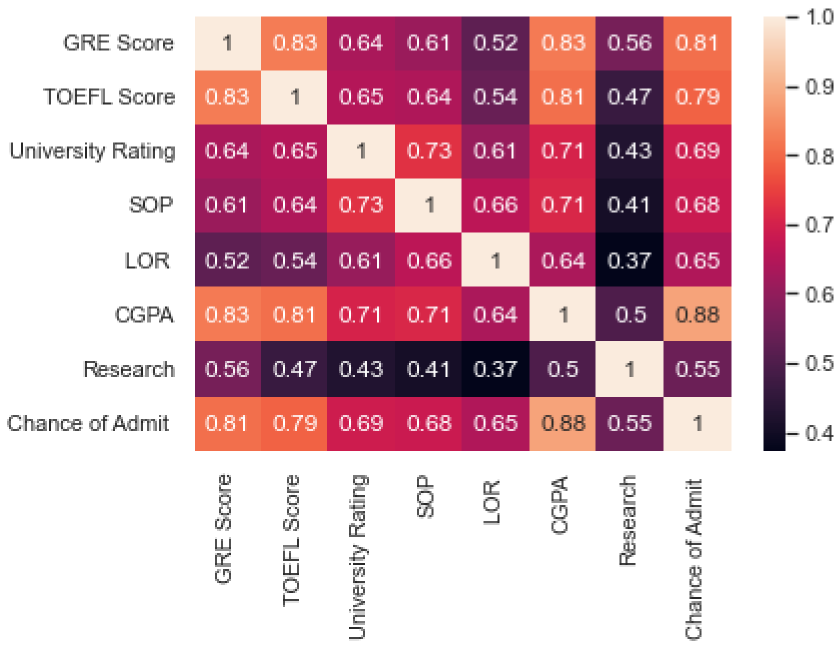
In [160]:
```
df.drop(columns=['Serial No.'],axis=1,inplace=True)
df.head()
```

Out[160]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

In [161]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[161]: <matplotlib.axes._subplots.AxesSubplot at 0x1feb6362640>



In [162]:
```
df.shape
```

Out[162]: (500, 8)

In [163]:
```
df.columns
```

Out[163]:
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGP
A',
       'Research', 'Chance of Admit '],
      dtype='object')
```

Through Trail and error, decided these are the columns that help me in uilding a better model, as there is no much difference in the model with all variables and model with the below mentioned variables

In [164]:
```
col=['GRE Score',
'TOEFL Score',
'LOR ',
'CGPA',
'SOP',
'Research'
]
```

Scaling the data

In [165]:
```
from sklearn.preprocessing import StandardScaler
df1=df
for i in col:
    df1[i] = StandardScaler().fit_transform(df1[[i]])
```

In [166]:
```
df1.head()
```

Out[166]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1.819238	1.778865	4	1.137360	1.098944	1.776806	0.886405	0.92
1	0.667148	-0.031601	4	0.632315	1.098944	0.485859	0.886405	0.76
2	-0.041830	-0.525364	3	-0.377773	0.017306	-0.954043	0.886405	0.72
3	0.489904	0.462163	3	0.127271	-1.064332	0.154847	0.886405	0.80
4	-0.219074	-0.689952	2	-1.387862	-0.523513	-0.606480	-1.128152	0.65

In [167]:
```
X=df1[col]
Y=df1['Chance of Admit ']
```

In [ ]:

In [168]:
```
print(X.shape, Y.shape)
```

```
(500, 6) (500,)
```

Performing Linear Regression with Statsmodel

In [169]:
```
import statsmodels.api as sm
```

In [170]:
```
X_sm = sm.add_constant(X)

sm_model = sm.OLS(Y, X_sm).fit()
```

Summary of Stats Model

In [171]:

```
print(sm_model.summary())
```

```
                            OLS Regression Results                            
==============================================================================
=
Dep. Variable:       Chance of Admit   R-squared:                       0.82
1
Model:                           OLS   Adj. R-squared:                  0.81
9
Method:                Least Squares   F-statistic:                     376.
9
Date:               Tue, 15 Mar 2022   Prob (F-statistic):          1.37e-18
0
Time:                       23:29:46   Log-Likelihood:                 700.1
5
No. Observations:                500   AIC:                            -138
6.
Df Residuals:                    493   BIC:                            -135
7.
Df Model:                          6                                         
Covariance Type:           nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const          0.7217      0.003    268.647      0.000       0.716       0.7
27
GRE Score      0.0214      0.006      3.775      0.000       0.010       0.0
33
TOEFL Score    0.0176      0.005      3.329      0.001       0.007       0.0
28
LOR            0.0164      0.004      4.333      0.000       0.009       0.0
24
CGPA           0.0728      0.006     12.531      0.000       0.061       0.0
84
SOP            0.0042      0.004      0.991      0.322      -0.004       0.0
12
Research       0.0123      0.003      3.767      0.000       0.006       0.0
19
==============================================================================
=
Omnibus:                     112.527   Durbin-Watson:                   0.78
9
Prob(Omnibus):                 0.000   Jarque-Bera (JB):              260.84
9
Skew:                         -1.158   Prob(JB):                     2.28e-5
7
Kurtosis:                      5.675   Cond. No.                         5.2
5
==============================================================================
=

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
```

Checking VIF Score

In [172]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [173]:
```
vif = pd.DataFrame()
X_t = X
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shap
e[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[173]:

	Features	VIF
3	CGPA	4.68
0	GRE Score	4.45
1	TOEFL Score	3.87
4	SOP	2.45
2	LOR	1.99
5	Research	1.49

Linear Regression Using Scikit Learn

In [174]:
```
from sklearn.linear_model import LinearRegression
rm=LinearRegression()
rm.fit(X,Y)
```

Out[174]:
```
LinearRegression()
```

In [175]:
```
print(rm.intercept_)
```

```
0.7217399999999997
```

In [189]:
```
[print(f"The coefficient of {col[i]} is {rm.coef_[i]}") for i in range(len(col
))]
```

```
The coefficient of GRE Score is 0.021404695703631405
The coefficient of TOEFL Score is 0.01760495005366209
The coefficient of LOR  is 0.016432612597028093
The coefficient of CGPA is 0.07283664907806046
The coefficient of SOP is 0.004168721982917806
The coefficient of Research is 0.012349581379967556
```

Out[189]:
```
[None, None, None, None, None, None]
```

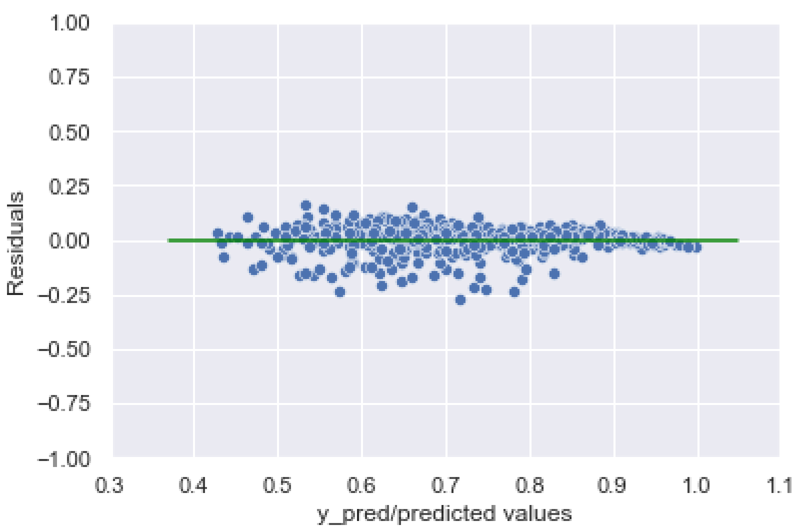
In [177]:
```
rm.score(X,Y)
```
Out[177]: 0.8210166861582956

Mean of Resiuals

In [178]:
```
Y_pred = rm.predict(X)
residuals = Y.values-Y_pred
mean_residuals = np.mean(residuals)
print("Mean of Residuals {}".format(round(mean_residuals,5)))
```
```
Mean of Residuals 0.0
```

Test of Homoscedasticity

In [179]:
```
sns.scatterplot(Y_pred,residuals)
plt.xlabel('y_pred/predicted values')
plt.ylabel('Residuals')
plt.xlabel('y_pred/predicted values')
plt.ylabel('Residuals')
plt.ylim(-1,1)
plt.xlim(0.3,1.10)
p = sns.lineplot([0.37,1.05],[0,0],color='green')
```



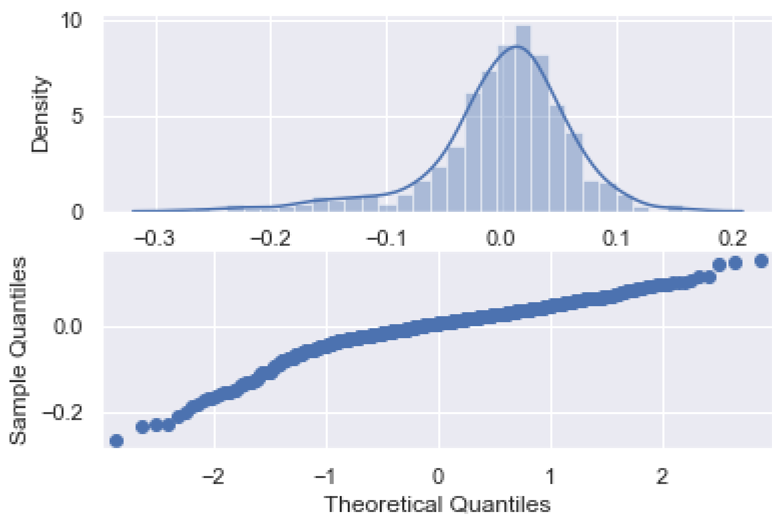
Goldfeld Quandt Test to check Homoscedasticity

In [180]:
```
import statsmodels.stats.api as sms
from statsmodels.compat import lzip
name=['F_stat','P_value']
GQT = sms.het_goldfeldquandt(residuals, X)
lzip(name,GQT)
```
Out[180]: [('F_stat', 0.45996749764483563), ('P_value', 0.9999999989223426)]

Normality of Residuals

In [181]:
```
fig,axes=plt.subplots(2)
sns.distplot(residuals,ax=axes[0])
sm.qqplot(residuals,ax=axes[1])
p=stats.normaltest(residuals).pvalue
if p>0.05:
    print('Residuals are Normally Distributed as p value is  '+str(p)+' which
 is greater than 0.05')
else:
    print('Residuals are not Normally Distributed as p value is  '+str(p)+' wh
ich is less than 0.05')
```

```
Residuals are not Normally Distributed as p value is  3.6739042749043505e-25
which is less than 0.05
```



Performance checks on Model

In [182]:
```
print("mean absolute error : ",mean_absolute_error(Y_pred,Y))
print("Mean Squared error : ", mean_squared_error(Y_pred,Y))
print("Root Mean squared error : ", np.sqrt(mean_squared_error(Y_pred,Y)))
print("Mean absolute Percentage error : ",mean_absolute_percentage_error(Y_pre
d,Y))
print("R-squared : ",rm.score(X,Y))
print("Adjusted R-squared : ", 1 - (1-rm.score(X, Y))*(len(Y)-1)/(len(Y)-X.sha
pe[1]-1))
```

```
mean absolute error :  0.04276860707461887
Mean Squared error :  0.003558326525884695
Root Mean squared error :  0.059651710167309496
Mean absolute Percentage error :  0.0641908511172667
R-squared :  0.8210166861582956
Adjusted R-squared :  0.8188383902494716
```

Linear Regression with Train and Test data

In [183]:
```
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size = 0.2,random_st
ate=1 )
```

In [184]:
```
fm=LinearRegression()
fm.fit(x_train,y_train)
```

Out[184]:
```
LinearRegression()
```

In [185]:
```
print(fm.intercept_,fm.coef_)
```

```
0.7226452460475254 [0.02142235 0.01981272 0.01430457 0.07165618 0.00570992 0.
01020389]
```

In [186]:
```
y_pred = fm.predict(x_test)
```

In [187]:
```
print("mean absolute error : ",mean_absolute_error(y_pred,y_test))
print("Mean Squared error : ", mean_squared_error(y_pred,y_test))
print("Root Mean squared error : ", np.sqrt(mean_squared_error(y_pred,y_test
)))
print("Mean absolute Percentage error : ",mean_absolute_percentage_error(y_pre
d,y_test))
print("R-squared : ",rm.score(x_train,y_train))
print("Adjusted R-squared : ", 1 - (1-rm.score(x_train, y_train))*(len(y_train
)-1)/(len(y_train)-x_train.shape[1]-1))
```

```
mean absolute error :  0.04006870393111951
Mean Squared error :  0.0034716563533543753
Root Mean squared error :  0.058920763346670714
Mean absolute Percentage error :  0.05874548625702469
R-squared :  0.8202210391178049
Adjusted R-squared :  0.8174763221577713
```

In [190]:
```
[print(f"The coefficient of {col[i]} is {fm.coef_[i]}") for i in range(len(col
))]
```

```
The coefficient of GRE Score is 0.021422347634471903
The coefficient of TOEFL Score is 0.019812718838837576
The coefficient of LOR  is 0.014304568752076963
The coefficient of CGPA is 0.07165617630233034
The coefficient of SOP is 0.005709916757106009
The coefficient of Research is 0.010203891926236885
```

Out[190]:
```
[None, None, None, None, None, None]
```

For all the models which e built using different techniques, there is a good value of R-squared and adjusted R- Square value. So we can consider our model as good and the Root Mean Squared Errors for Scikit Learn Models (one is with direct data and other with train,test data) are also pretty low