Jamboree has helped thousands of students like you make it to top colleges abroad. Be it GMAT, GRE or SAT, their unique problem-solving methods ensure maximum scores with minimum effort.

They 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.

How can you help here?

Your analysis will help Jamboree in understanding what factors are important in graduate admissions and how these factors are interrelated among themselves. It will also help predict one's chances of admission given the rest of the variables.

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import warnings
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import minmax_scale
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
from sklearn import metrics
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import GridSearchCV
```

In [2]:

```
warnings.filterwarnings('ignore')
```

In [3]:

```
df = pd.read_csv('Jamboree_Admission.csv')
```

In [4]:

```
df.head()
```

Out[4]:

	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 [5]:

```
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
```

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

There are no null values in the dataframe

In [7]:

```
## Copy of the dataset for further usage
df_copy = df.copy(deep = True)
```

Taking a copy of df for future evaluations

In [8]:

```
df.columns
```

Out[8]:

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

In [9]:

```
## There is space at the end of the column name so renaming it.
df.rename(columns={'LOR ':'LOR','Chance of Admit ':'Chance of Admit'},inplace = True)
```

In [10]:

```
df_copy = df.copy(deep = True)
```

In [11]:

```
## removin the first column as it has no significance just an index
df.drop(['Serial No.'],axis=1,inplace=True)
```

In [12]:

```
df[df.duplicated()]
```

Out[12]:

GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit

```
There are no duplicated values in the dataset.
```

In [ ]:

```
```

Univariate Analysis

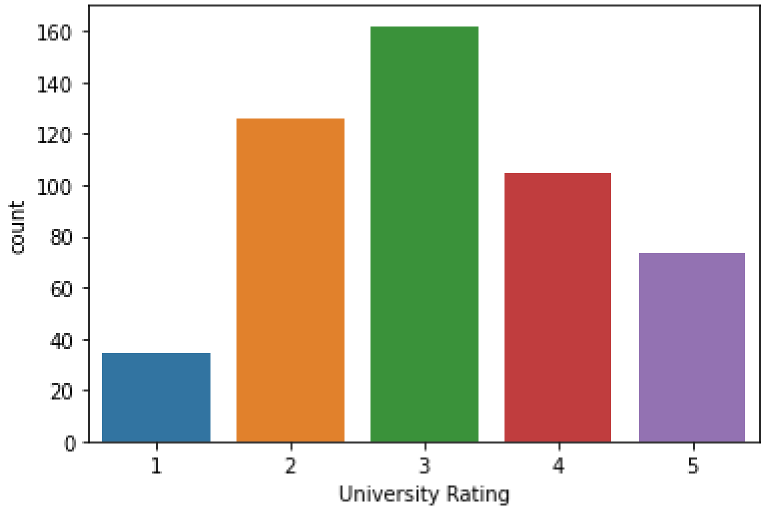
In [13]:

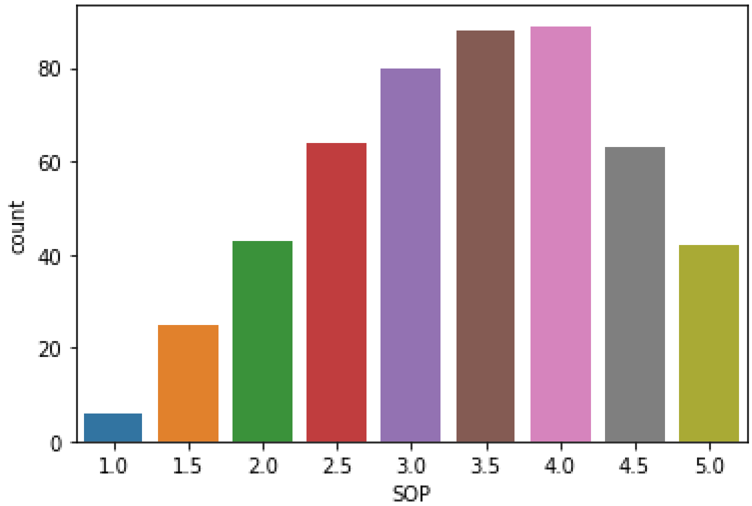
```
cat_cols = []
for i in df.columns:
    if df[i].nunique() < 10:
        cat_cols.append(i)
        print(i + ':' + str(df[i].nunique()))
```

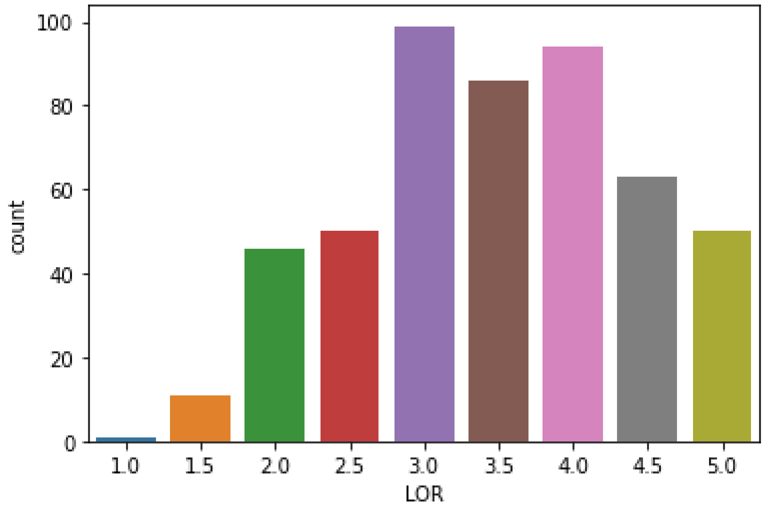
```
University Rating:5
SOP:9
LOR:9
Research:2
```

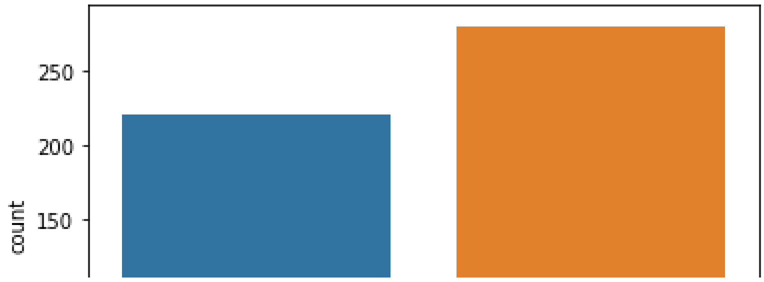
In [14]:

```
for i in range(len(cat_cols)):
    sns.countplot(df[cat_cols[i]])
    plt.show()
```





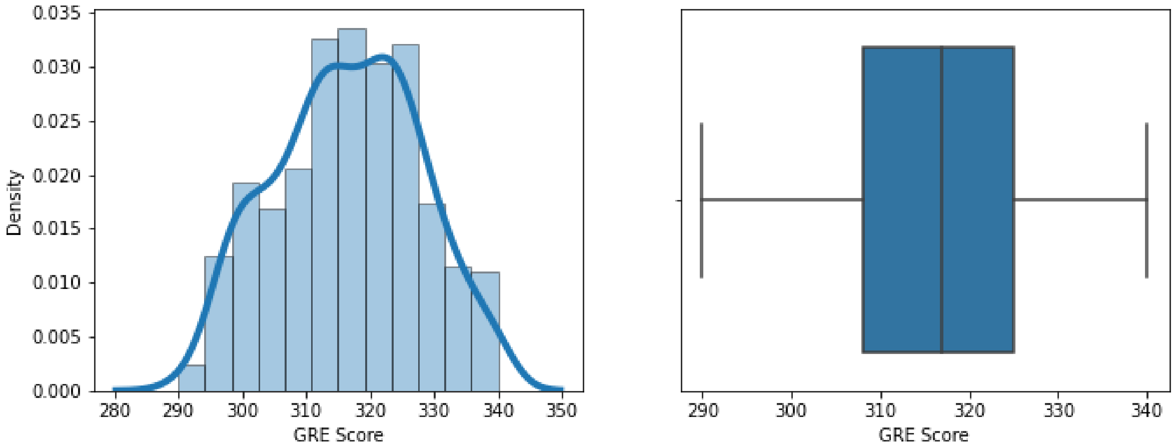


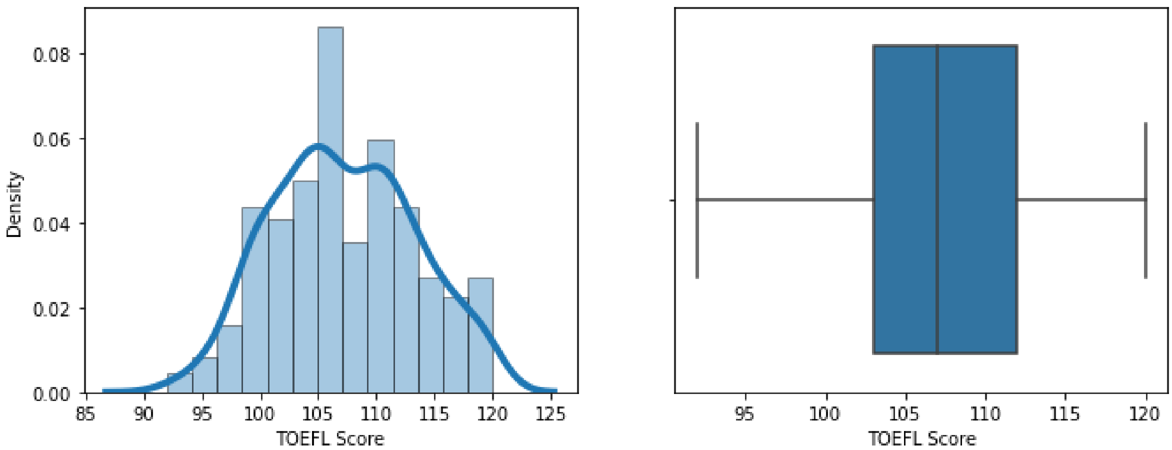


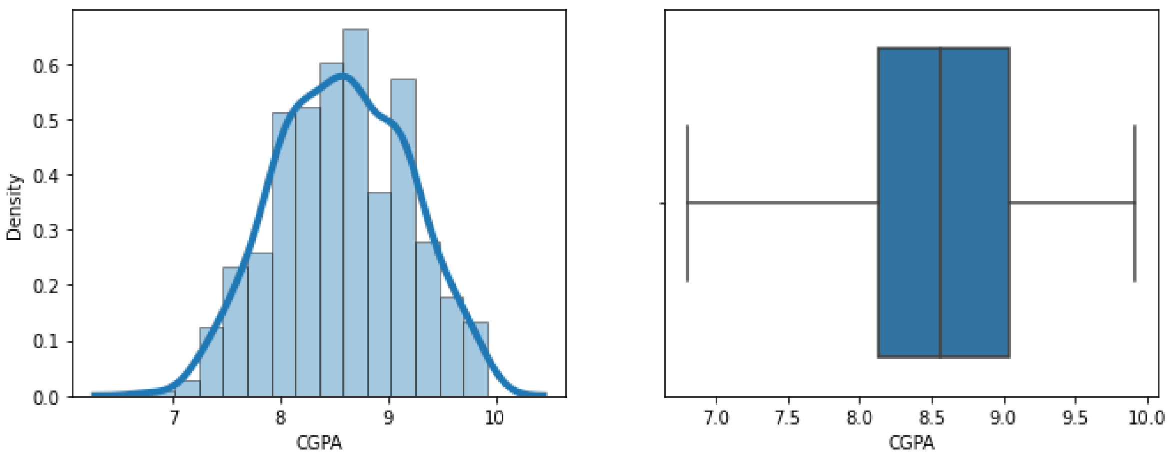
-->Most of the universities are average rated (around 3)
-->SOP of the students is on bit higher rate (4)
-->LOR of students is around 3-4
-->Among all the students most of the students have done research
-->From above observaions most of the students are average students and very few students are good performing and very few students are bad performing

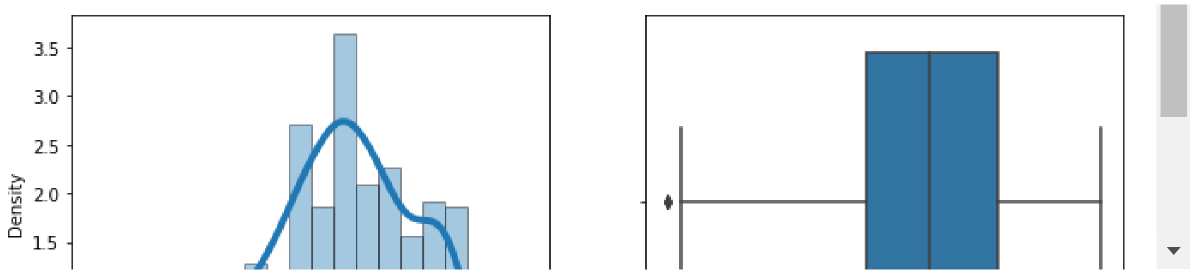
In [15]:

```
con_cols = []
for i in df.columns:
    if df[i].nunique() > 10:
        con_cols.append(i)
        plt.figure(figsize=(11,4))
        plt.subplot(1,2,1)
        sns.distplot(df[i],hist=True,kde=True,hist_kws={'edgecolor':'black'},kde_kws ={"lin
        plt.subplot(1,2,2)
        sns.boxplot(df[i])
        plt.show()
```









From above plots, we can say that most of the data follows normal distribution and there are no outliers in the data.

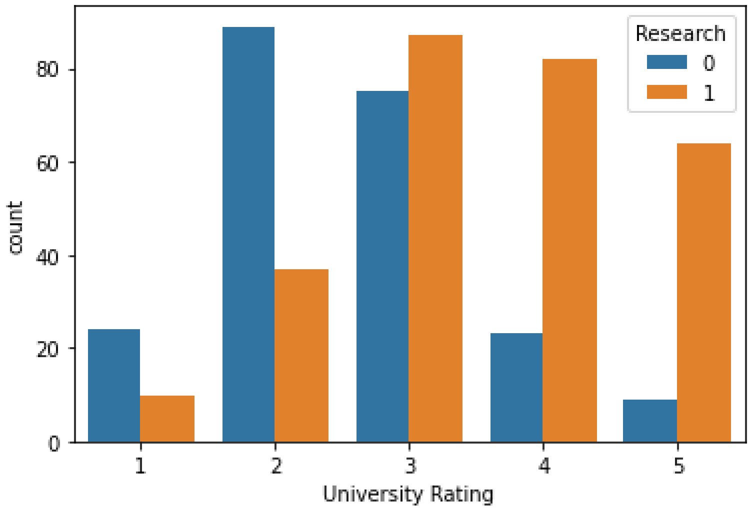
BiVariate Analysis

In [16]:

```
sns.countplot(x=df['University Rating'],hue =df['Research'])
```

Out[16]:

<AxesSubplot:xlabel='University Rating', ylabel='count'>



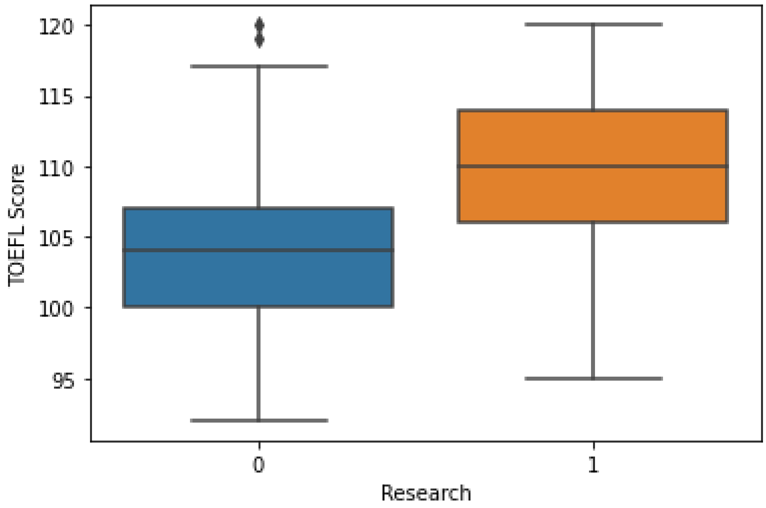
Most of the students who have done research has done research prefers for universities with high rating

In [17]:

```
sns.boxplot(y=df['TOEFL Score'],x=df['Research'])
```

Out[17]:

```
<AxesSubplot:xlabel='Research', ylabel='TOEFL Score'>
```



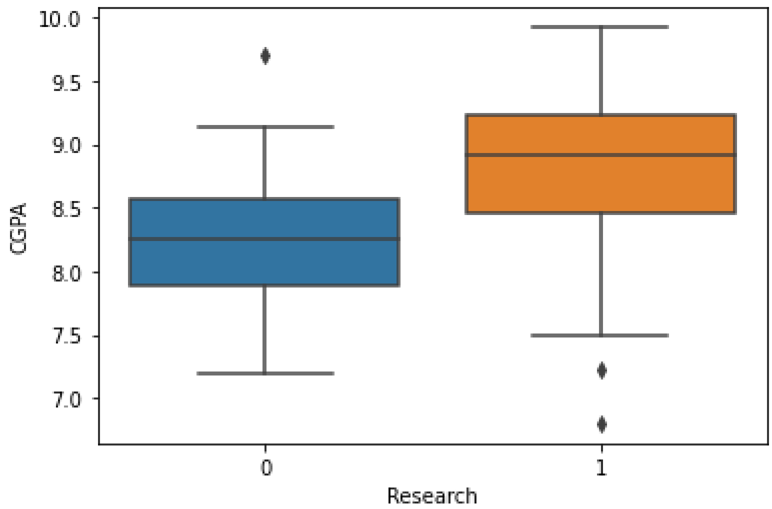

```
Median of TOFEL Scores is higher for students who have research experience
```

In [18]:

```
sns.boxplot(y=df['CGPA'],x=df['Research'])
```

Out[18]:

```
<AxesSubplot:xlabel='Research', ylabel='CGPA'>
```



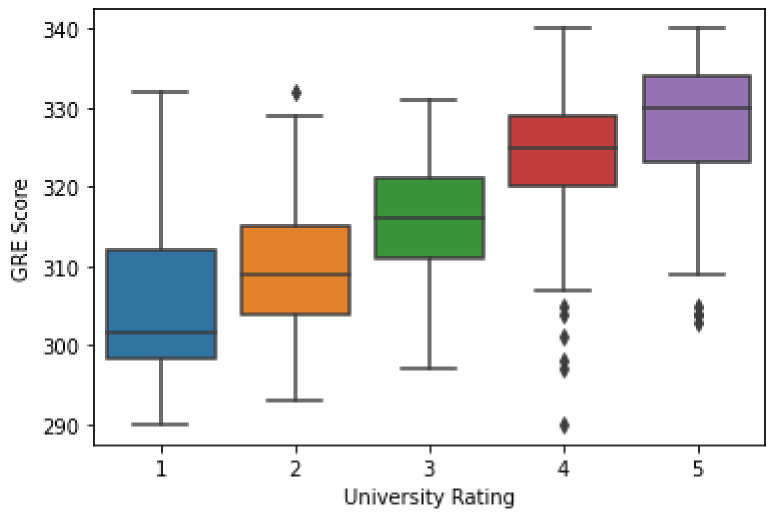

```
Students who have high CGPA are more likely to have research experience
```

In [19]:

```
sns.boxplot(x=df['University Rating'],y=df['GRE Score'])
```

Out[19]:

```
<AxesSubplot:xlabel='University Rating', ylabel='GRE Score'>
```



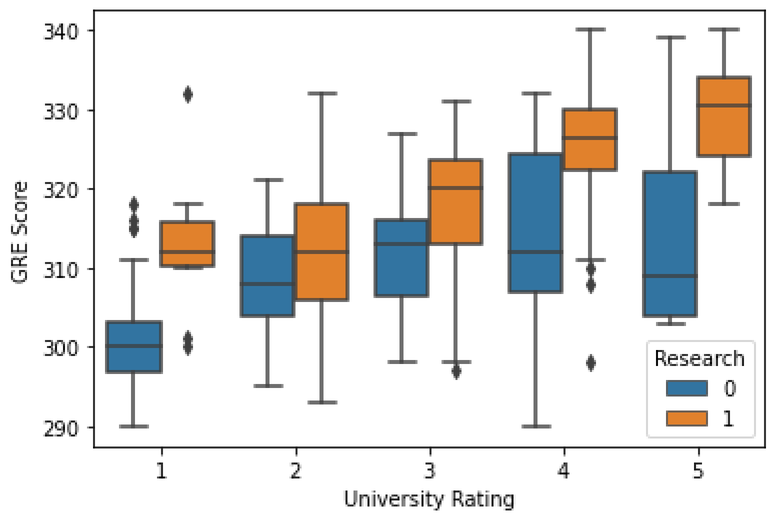

```
As the GRE scores increases the university prefrence of the students also increases
```

In [20]:

```
sns.boxplot(x=df['University Rating'],y=df['GRE Score'],hue = df['Research'])
```

Out[20]:

```
<AxesSubplot:xlabel='University Rating', ylabel='GRE Score'>
```



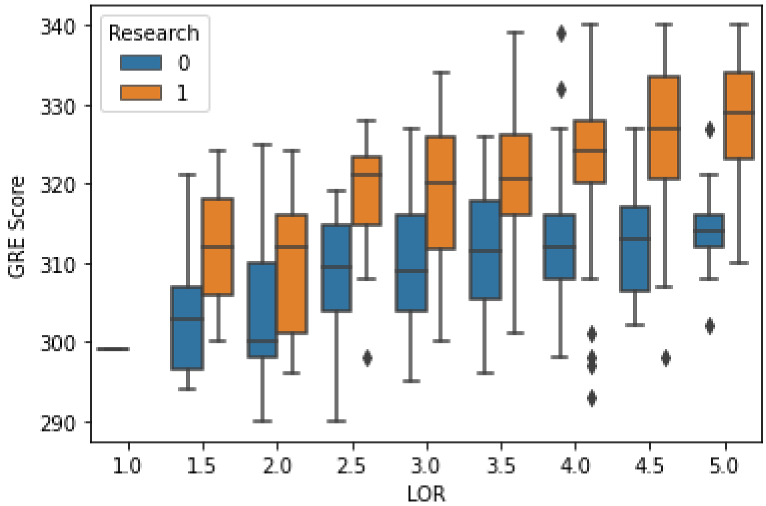

```
Students who have high GRE scores are having research experience
```

In [21]:

```
sns.boxplot(x=df['LOR'],y=df['GRE Score'],hue = df['Research'])
```

Out[21]:

```
<AxesSubplot:xlabel='LOR', ylabel='GRE Score'>
```

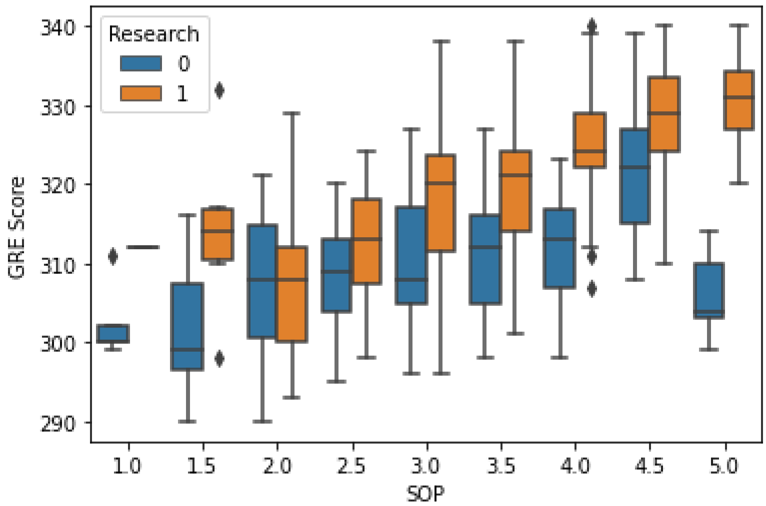


In [22]:

```
sns.boxplot(x=df['SOP'],y=df['GRE Score'],hue = df['Research'])
```

Out[22]:

```
<AxesSubplot:xlabel='SOP', ylabel='GRE Score'>
```



From above plots we can say that students who have research experience gets high GRE scores than others

In [23]:

```
df.corr()
```

Out[23]:

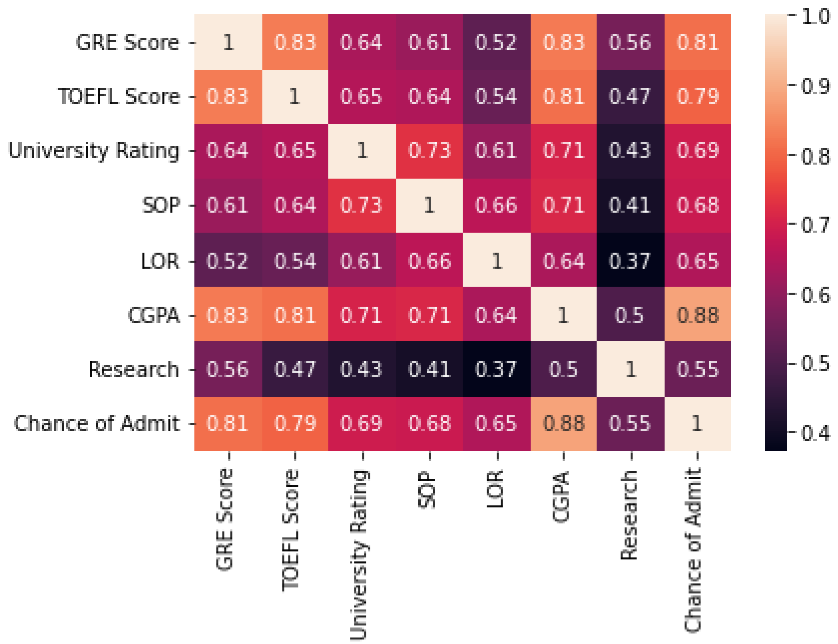
	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1.000000	0.827200	0.635376	0.613498	0.524679	0.825878	0.563398	0.810351
TOEFL Score	0.827200	1.000000	0.649799	0.644410	0.541563	0.810574	0.467012	0.792228
University Rating	0.635376	0.649799	1.000000	0.728024	0.608651	0.705254	0.427047	0.690132
SOP	0.613498	0.644410	0.728024	1.000000	0.663707	0.712154	0.408116	0.684137
LOR	0.524679	0.541563	0.608651	0.663707	1.000000	0.637469	0.372526	0.645365
CGPA	0.825878	0.810574	0.705254	0.712154	0.637469	1.000000	0.501311	0.882413
Research	0.563398	0.467012	0.427047	0.408116	0.372526	0.501311	1.000000	0.545871
Chance of Admit	0.810351	0.792228	0.690132	0.684137	0.645365	0.882413	0.545871	1.000000

In [24]:

```
sns.heatmap(df.corr(), annot = True)
```

Out[24]:

<AxesSubplot:>




```
From above heat map we can infer that
    --> GRE,TOFEL,CGPA and chance of admit are highly correlated
    --> University rating and SOP are highly correlated
```

Linear Regression

Linear Regression

In [25]:

```
df.columns
```

Out[25]:

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

In [26]:

```
df.head()
```

Out[26]:

	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 [27]:

```
## Target Feature
predict = df['Chance of Admit']
```

In [28]:

```
## dropping target column from dataframe
df.drop(['Chance of Admit'],axis=1,inplace=True)
```

In [29]:

```
df.head()
```

Out[29]:

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

In [30]:

```
# Normalisation
df.columns
for i in df.columns:
    df[i] = minmax_scale(df[i])
df.head()
```

Out[30]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	0.94	0.928571	0.75	0.875	0.875	0.913462	1.0
1	0.68	0.535714	0.75	0.750	0.875	0.663462	1.0
2	0.52	0.428571	0.50	0.500	0.625	0.384615	1.0
3	0.64	0.642857	0.50	0.625	0.375	0.599359	1.0
4	0.48	0.392857	0.25	0.250	0.500	0.451923	0.0

In [31]:

```
predict = pd.DataFrame(predict,columns=['Chance of Admit'])
```

In [32]:

```
#Normalisation of target feature
predict['Chance of Admit'] = minmax_scale(predict['Chance of Admit'])
```

In [33]:

```
predict.head()
```

Out[33]:

	Chance of Admit
0	0.920635
1	0.666667
2	0.603175
3	0.730159
4	0.492063

In [34]:

```
#Train and Test data split
x_train,x_test,y_train,y_test = train_test_split(df,predict,train_size=0.30,random_state=1)
```

In [35]:

```
x_train.head()
```

Out[35]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
409	0.20	0.214286	0.00	0.250	0.375	0.391026	0.0
103	0.54	0.428571	0.25	0.875	0.750	0.535256	0.0
220	0.46	0.392857	0.50	0.750	0.750	0.625000	0.0
130	0.98	0.785714	1.00	0.750	0.875	0.948718	1.0
353	0.20	0.357143	0.50	0.625	0.375	0.439103	0.0

In [36]:

```
lr_1 = LinearRegression()
```

In [37]:

```
lr_1.fit(x_train,y_train)
```

Out[37]:

```
LinearRegression()
```

In [38]:

```
y_predict = lr_1.predict(x_test)
```

In [39]:

```
lr_1.intercept_
```

Out[39]:

```
array([0.06052024])
```

In [40]:

```
for idx, col_name in enumerate(x_train.columns):
    print("The coefficient for {} is {}".format(col_name, lr_1.coef_[0][idx]))
```

```
The coefficient for GRE Score is -0.0050978648509186104
The coefficient for TOEFL Score is 0.21669717001842637
The coefficient for University Rating is 0.05851981351758148
The coefficient for SOP is 0.03264096702009038
The coefficient for LOR is 0.03798672416464777
The coefficient for CGPA is 0.591252249663798
The coefficient for Research is 0.05080772261800126
```

In [41]:

```
intercept = lr_1.intercept_[0]
print("The intercept for our model is {}".format(intercept))
```

```
The intercept for our model is 0.060520238439629614
```

Ridge

In [42]:

```
ridge = Ridge() # initializing the model
```

In [43]:

```
params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0, 4.0,5.0]} ## Parameters for alpha t
```

In [44]:

```
folds = 5 #total number of crossvaldations to be done during the model building
```

In [45]:

```
model_cv = GridSearchCV(estimator = ridge,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)
```

In [46]:

```
model_cv.fit(x_train, y_train) # fit the model
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
```

Out[46]:

```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0,
3.0,
                                   4.0, 5.0]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

In [47]:

```
cv_results = pd.DataFrame(model_cv.cv_results_) # to get the output of the model
cv_results = cv_results[cv_results['param_alpha']<=200]
cv_results.head()
```

Out[47]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split
0	0.007201	0.001834	0.004627	0.001250	0.0001	{'alpha': 0.0001}	
1	0.007750	0.005073	0.003419	0.003326	0.001	{'alpha': 0.001}	
2	0.000000	0.000000	0.006252	0.007658	0.01	{'alpha': 0.01}	
3	0.003126	0.006252	0.003125	0.006250	0.05	{'alpha': 0.05}	
4	0.007425	0.005962	0.004122	0.002295	0.1	{'alpha': 0.1}	

5 rows × 21 columns

In [48]:

```
cv_results.columns
```

Out[48]:

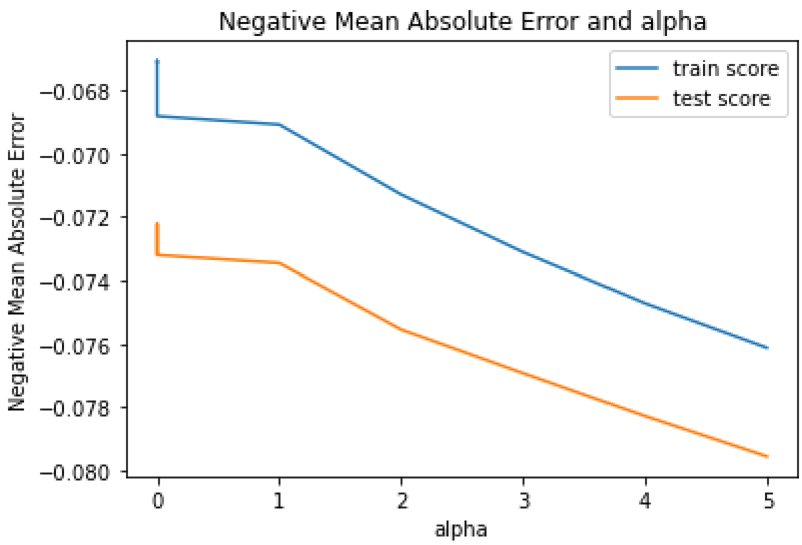
```
Index(['mean_fit_time', 'std_fit_time', 'mean_score_time', 'std_score_time',
       'param_alpha', 'params', 'split0_test_score', 'split1_test_score',
       'split2_test_score', 'split3_test_score', 'split4_test_score',
       'mean_test_score', 'std_test_score', 'rank_test_score',
       'split0_train_score', 'split1_train_score', 'split2_train_score',
       'split3_train_score', 'split4_train_score', 'mean_train_score',
       'std_train_score'],
      dtype='object')
```

```
Here we have the scores for each test and train split, as we gave cross validation as 5
there are 5 train and test splits along with the ranks for each alpha
```

In [49]:

```
# plotting mean test and train scoes with alpha
cv_results['param_alpha'] = cv_results['param_alpha'].astype('int32')

# plotting
plt.plot(cv_results['param_alpha'], cv_results['mean_train_score'])
plt.plot(cv_results['param_alpha'], cv_results['mean_test_score'])
plt.xlabel('alpha')
plt.ylabel('Negative Mean Absolute Error')
plt.title("Negative Mean Absolute Error and alpha")
plt.legend(['train score', 'test score'], loc='upper right')
plt.show()
```



In [50]:

```
alpha = 1
ridge = Ridge(alpha=alpha)

ridge.fit(x_train, y_train)
ridge.coef_
```

Out[50]:

```
array([[0.10754325, 0.20096854, 0.06723524, 0.06535809, 0.0690688 ,
        0.3543325 , 0.0545913 ]])
```

Lasso

In [51]:

```
params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]}
```

In [52]:

```
lasso = Lasso()

# cross validation
model_cv = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)

model_cv.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 14 candidates, totalling 70 fits
```

Out[52]:

```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

In [53]:

```
cv_results = pd.DataFrame(model_cv.cv_results_)
cv_results.head()
```

Out[53]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split0_t
0	0.004125	0.006066	0.007277	0.007109	0.0001	{'alpha': 0.0001}	
1	0.006106	0.005892	0.001620	0.001984	0.001	{'alpha': 0.001}	
2	0.006251	0.007655	0.003125	0.006251	0.01	{'alpha': 0.01}	
3	0.011153	0.008083	0.003000	0.002683	0.05	{'alpha': 0.05}	
4	0.004725	0.005737	0.001600	0.002059	0.1	{'alpha': 0.1}	

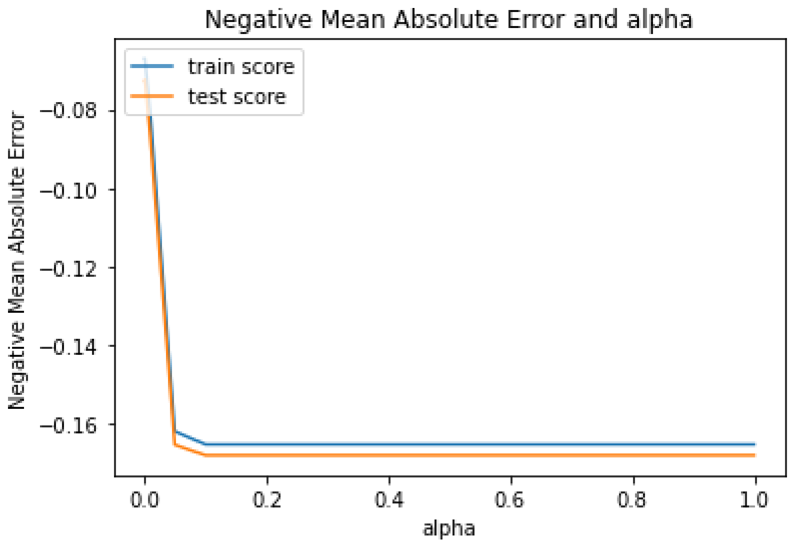
5 rows × 21 columns

In [54]:

```
# plotting mean test and train scoes with alpha
cv_results['param_alpha'] = cv_results['param_alpha'].astype('float32')

# plotting
plt.plot(cv_results['param_alpha'], cv_results['mean_train_score'])
plt.plot(cv_results['param_alpha'], cv_results['mean_test_score'])
plt.xlabel('alpha')
plt.ylabel('Negative Mean Absolute Error')

plt.title("Negative Mean Absolute Error and alpha")
plt.legend(['train score', 'test score'], loc='upper left')
plt.show()
```



In [55]:

```
alpha = 0.1
lasso = Lasso(alpha=alpha)

ridge.fit(x_train, y_train)
ridge.coef_
```

Out[55]:

```
array([[0.10754325, 0.20096854, 0.06723524, 0.06535809, 0.0690688 ,
        0.3543325 , 0.0545913 ]])
```

OLS Regression

In [56]:

```
xtrain_ols1 = sm.add_constant(x_train) ## adding constant to perform OLS regression
ols_regression1 = sm.OLS(y_train,xtrain_ols1).fit() ## fitting model
```

In [57]:

```
print(ols_regression1.summary()) ## Summary of the OLS Model
```

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:        Chance of Admit   R-squared:                       0.7
95
Model:                            OLS   Adj. R-squared:                  0.7
85
Method:                 Least Squares   F-statistic:                     78.
89
Date:                Tue, 08 Nov 2022   Prob (F-statistic):           9.01e-
46
Time:                        17:10:19   Log-Likelihood:                 142.
34
No. Observations:                 150   AIC:                            -26
8.7
Df Residuals:                     142   BIC:                            -24
4.6
Df Model:                           7                                         
Covariance Type:            nonrobust                                         
==============================================================================
=========
                        coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const                 0.0605      0.029      2.089      0.038       0.003
0.118
GRE Score            -0.0051      0.077     -0.066      0.947      -0.158
0.147
TOEFL Score           0.2167      0.073      2.980      0.003       0.073
0.360
University Rating     0.0585      0.048      1.225      0.223      -0.036
0.153
SOP                   0.0326      0.055      0.598      0.551      -0.075
0.141
LOR                   0.0380      0.048      0.785      0.434      -0.058
0.134
CGPA                  0.5913      0.092      6.411      0.000       0.409
0.774
Research              0.0508      0.019      2.700      0.008       0.014
0.088
==============================================================================
==
Omnibus:                       33.098   Durbin-Watson:                   2.1
07
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               54.4
71
Skew:                          -1.085   Prob(JB):                     1.49e-
12
Kurtosis:                       5.001   Cond. No.                         2
5.1
==============================================================================
==

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

In [58]:

```
## VIF of the features present in the model

xtrain_vif1 = x_train
vif_data = pd.DataFrame()
vif_data["feature"] = xtrain_vif1.columns
vif_data["VIF"] = [variance_inflation_factor(xtrain_vif1.values, i) for i in range(xtrain_v
vif_data['VIF'] = round(vif_data['VIF'],2)
vif_data = vif_data.sort_values(by = 'VIF',ascending=False)
vif_data
```

Out[58]:

	feature	VIF
5	CGPA	44.80
0	GRE Score	32.49
1	TOEFL Score	29.84
3	SOP	20.12
4	LOR	16.23
2	University Rating	12.23
6	Research	3.14

```
P-Value and VIF is higher for GRE Score, hence dropping the column and rebuilding the
model
```

In [59]:

```
## Removing the columns in the dataset

xtrain_ols2 = xtrain_ols1.drop('GRE Score',1)
xtrain_vif2 = xtrain_vif1.drop('GRE Score',1)

xtrain_ols2 = sm.add_constant(xtrain_ols2)
ols_regression2 = sm.OLS(y_train,xtrain_ols2).fit()
```

In [60]:

```
print(ols_regression2.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:         Chance of Admit   R-squared:                       0.7
95
Model:                             OLS   Adj. R-squared:                  0.7
87
Method:                  Least Squares   F-statistic:                     92.
68
Date:                 Tue, 08 Nov 2022   Prob (F-statistic):           8.97e-
47
Time:                         17:10:19   Log-Likelihood:                 142.
33
No. Observations:                  150   AIC:                            -27
0.7
Df Residuals:                      143   BIC:                            -24
9.6
Df Model:                            6
Covariance Type:             nonrobust
==============================================================================
=========
                        coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const                 0.0606      0.029      2.101      0.037       0.004
0.118
TOEFL Score           0.2142      0.062      3.440      0.001       0.091
0.337
University Rating     0.0579      0.047      1.240      0.217      -0.034
0.150
SOP                   0.0332      0.054      0.617      0.538      -0.073
0.139
LOR                   0.0385      0.048      0.811      0.419      -0.055
0.132
CGPA                  0.5885      0.082      7.152      0.000       0.426
0.751
Research              0.0505      0.018      2.807      0.006       0.015
0.086
==============================================================================
==
Omnibus:                        33.364   Durbin-Watson:                   2.1
06
Prob(Omnibus):                   0.000   Jarque-Bera (JB):                55.1
59
Skew:                           -1.091   Prob(JB):                     1.05e-
12
Kurtosis:                        5.016   Cond. No.                          2
1.6
==============================================================================
==

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

In [61]:

```
vif_data2 = pd.DataFrame()
vif_data2["feature"] = xtrain_vif2.columns
vif_data2["VIF"] = [variance_inflation_factor(xtrain_vif2.values, i) for i in range(xtrain_
vif_data2['VIF'] = round(vif_data2['VIF'],2)
vif_data2 = vif_data2.sort_values(by = 'VIF',ascending=False)
vif_data2
```

Out[61]:

	feature	VIF
4	CGPA	34.99
0	TOEFL Score	22.07
2	SOP	19.63
3	LOR	15.66
1	University Rating	11.68
5	Research	2.87

```
P-Value and VIF is higher for SOP Score, hence dropping the column and rebuilding the
model
```

In [62]:

```
xtrain_ols3 = xtrain_ols2.drop('SOP',1)
xtrain_vif3 = xtrain_vif2.drop('SOP',1)

xtrain_ols3 = sm.add_constant(xtrain_ols3)
ols_regression3 = sm.OLS(y_train,xtrain_ols3).fit()
```

In [63]:

```
print(ols_regression3.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:        Chance of Admit   R-squared:                       0.7
95
Model:                            OLS   Adj. R-squared:                  0.7
88
Method:                 Least Squares   F-statistic:                     11
1.6
Date:                Tue, 08 Nov 2022   Prob (F-statistic):           9.72e-
48
Time:                        17:10:20   Log-Likelihood:                 142.
13
No. Observations:                 150   AIC:                            -27
2.3
Df Residuals:                     144   BIC:                            -25
4.2
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
=========
                        coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const                 0.0633      0.028      2.227      0.028       0.007
0.120
TOEFL Score           0.2158      0.062      3.476      0.001       0.093
0.339
University Rating     0.0715      0.041      1.737      0.084      -0.010
0.153
LOR                   0.0461      0.046      1.008      0.315      -0.044
0.137
CGPA                  0.5959      0.081      7.334      0.000       0.435
0.756
Research              0.0507      0.018      2.831      0.005       0.015
0.086
==============================================================================
==
Omnibus:                       33.060   Durbin-Watson:                   2.1
02
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               54.9
95
Skew:                          -1.077   Prob(JB):                     1.14e-
12
Kurtosis:                       5.039   Cond. No.                        2
0.1
==============================================================================
==

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

In [64]:

```
vif_data3 = pd.DataFrame()
vif_data3["feature"] = xtrain_vif3.columns
vif_data3["VIF"] = [variance_inflation_factor(xtrain_vif3.values, i) for i in range(xtrain_
vif_data3['VIF'] = round(vif_data3['VIF'],2)
vif_data3 = vif_data3.sort_values(by = 'VIF',ascending=False)
vif_data3
```

Out[64]:

	feature	VIF
3	CGPA	33.10
0	TOEFL Score	22.02
2	LOR	14.05
1	University Rating	9.32
4	Research	2.87

```
P-Value and VIF is higher for LOR Score, hence dropping the column and rebuilding the
model
```

In [65]:

```
xtrain_ols4 = xtrain_ols3.drop('LOR',1)
xtrain_vif4 = xtrain_vif3.drop('LOR',1)

xtrain_ols4 = sm.add_constant(xtrain_ols4)
ols_regression4 = sm.OLS(y_train,xtrain_ols4).fit()
```

In [66]:

```
print(ols_regression4.summary())
```

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:         Chance of Admit   R-squared:                       0.7
93
Model:                             OLS   Adj. R-squared:                  0.7
88
Method:                  Least Squares   F-statistic:                      13
9.3
Date:                 Tue, 08 Nov 2022   Prob (F-statistic):           1.28e-
48
Time:                         17:10:20   Log-Likelihood:                 141.
61
No. Observations:                  150   AIC:                             -27
3.2
Df Residuals:                      145   BIC:                             -25
8.2
Df Model:                            4                                         
Covariance Type:             nonrobust                                         
==============================================================================
=========
                        coef    std err          t      P>|t|      [0.025      
0.975]
------------------------------------------------------------------------------
---------
const                 0.0730      0.027      2.726      0.007       0.020      
0.126
TOEFL Score           0.2138      0.062      3.446      0.001       0.091      
0.337
University Rating     0.0882      0.038      2.343      0.020       0.014      
0.163
CGPA                  0.6164      0.079      7.837      0.000       0.461      
0.772
Research              0.0506      0.018      2.820      0.005       0.015      
0.086
==============================================================================
==
Omnibus:                      32.983   Durbin-Watson:                   2.0
98
Prob(Omnibus):                 0.000   Jarque-Bera (JB):                54.1
20
Skew:                         -1.083   Prob(JB):                     1.77e-
12
Kurtosis:                      4.991   Cond. No.                         1
8.2
==============================================================================
==

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

In [67]:

```
vif_data4 = pd.DataFrame()
vif_data4["feature"] = xtrain_vif4.columns
vif_data4["VIF"] = [variance_inflation_factor(xtrain_vif4.values, i) for i in range(xtrain_
vif_data4['VIF'] = round(vif_data4['VIF'],2)
vif_data4 = vif_data4.sort_values(by = 'VIF',ascending=False)
vif_data4
```

Out[67]:

	feature	VIF
2	CGPA	25.51
0	TOEFL Score	22.01
1	University Rating	8.16
3	Research	2.86

```
-->xtrain_ols4 has the data with features from which we can exactly predict our model,
going further we will be using this data.
-->Eventhough the VIF of the features is high but from OLS we can see that p-value of the
features less than 0.05 so from which we are rejecting the null hypothesis
```

In [68]:

```
ytrain_predict = ols_regression4.predict(xtrain_ols4)
```

In [69]:

```
x_test.head()
```

Out[69]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
304	0.46	0.500000	0.25	0.375	0.250	0.522436	0.0
340	0.44	0.535714	0.50	0.500	0.500	0.532051	1.0
47	0.98	0.964286	1.00	0.875	0.750	0.929487	0.0
67	0.52	0.535714	0.25	0.625	0.625	0.589744	1.0
479	0.70	0.642857	0.75	0.875	0.750	0.692308	1.0

In [70]:

```
xtrain_ols4.columns
```

Out[70]:

```
Index(['const', 'TOEFL Score', 'University Rating', 'CGPA', 'Research'], dty
pe='object')
```

In [71]:

```
x_test = x_test[['TOEFL Score', 'University Rating', 'CGPA', 'Research']]
```

In [72]:

```
x_test.head()
```

Out[72]:

	TOEFL Score	University Rating	CGPA	Research
304	0.500000	0.25	0.522436	0.0
340	0.535714	0.50	0.532051	1.0
47	0.964286	1.00	0.929487	0.0
67	0.535714	0.25	0.589744	1.0
479	0.642857	0.75	0.692308	1.0

In [73]:

```
x_test = sm.add_constant(x_test)
```

In [74]:

```
x_test.head()
```

Out[74]:

	const	TOEFL Score	University Rating	CGPA	Research
304	1.0	0.500000	0.25	0.522436	0.0
340	1.0	0.535714	0.50	0.532051	1.0
47	1.0	0.964286	1.00	0.929487	0.0
67	1.0	0.535714	0.25	0.589744	1.0
479	1.0	0.642857	0.75	0.692308	1.0

In [75]:

```
ytest_predict = ols_regression4.predict(x_test)
```

Linearity Check

In [76]:

```
df_copy.columns
```

Out[76]:

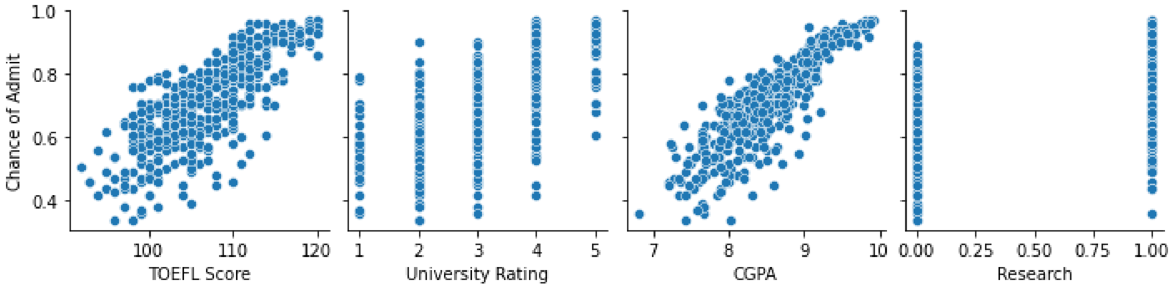
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR', 'CGPA', 'Research', 'Chance of Admit'],
      dtype='object')
```

In [77]:

```
sns.pairplot(df_copy, x_vars=['TOEFL Score', 'University Rating', 'CGPA', 'Research'],y_var
```

Out[77]:

```
<seaborn.axisgrid.PairGrid at 0x2978b1d1f40>
```



Mean of residuals

In [78]:

```
ytest_predict = pd.DataFrame(ytest_predict,columns = ['Chance of Admit'])
```

In [79]:

```
np.mean(ytest_predict-y_test)
```

Out[79]:

```
Chance of Admit    0.012823
dtype: float64
```

Test for Homoscedasticity

In [80]:

```
ytest_predict = pd.DataFrame(ytest_predict,columns=['Chance of Admit'])
```

In [81]:

```
ytest_predict.head()
```

Out[81]:

	Chance of Admit
304	0.524003
340	0.610173
47	0.940345
67	0.623686
479	0.753918

In [82]:

```
y_test.head()
```

Out[82]:

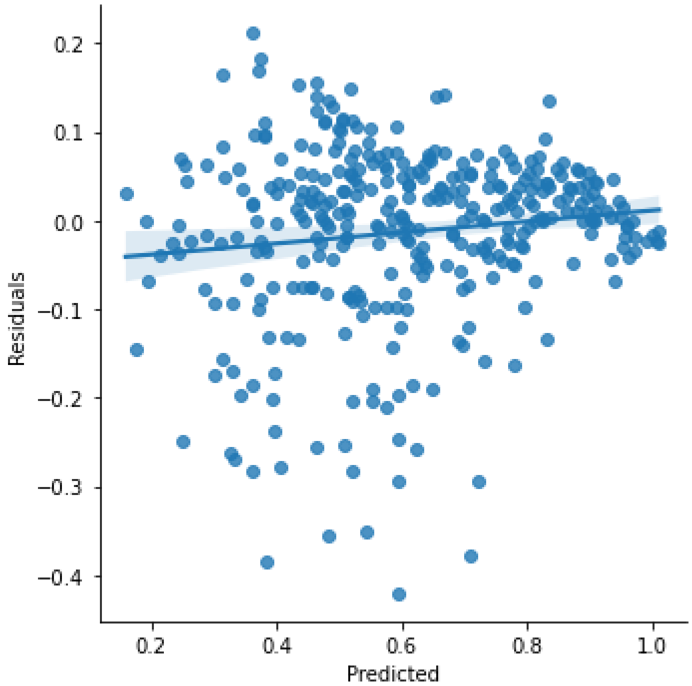
	Chance of Admit
304	0.444444
340	0.650794
47	0.873016
67	0.365079
479	0.714286

In [83]:

```
data = pd.DataFrame()
data['Predicted'] = ytest_predict
data['Residuals'] = y_test - ytest_predict
sns.lmplot(x='Predicted',y='Residuals',data=data)
```

Out[83]:

```
<seaborn.axisgrid.FacetGrid at 0x2978a963190>
```



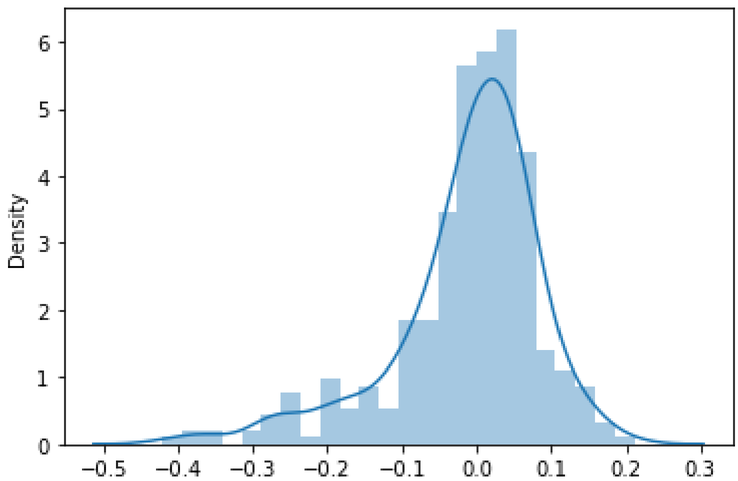
Normality of residuals

In [84]:

```
sns.distplot(y_test-ytest_predict)
```

Out[84]:

```
<AxesSubplot:ylabel='Density'>
```




```
form the above plot we can see that the graph is similar to the normal plot, Hence we can
say that the assumption is True
```

Model Evaluation

In [85]:

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,ytest_predict))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,ytest_predict))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,ytest_predict))
```

```
Mean Absolute Error: 0.06971171640623312
Mean Squared Error: 0.010046336251067323
Root Mean Squared Error: 0.10023141349430988
```

In [86]:

```
## R2 Score
r2 = r2_score(y_test,ytest_predict)
r2
```

Out[86]:

```
0.80978591888805
```

In [87]:

```
## Adjusted R2 Score
m = y_test.shape[0]
d = xtrain_vif4.shape[1]
adj_r2 = 1-((1-r2)*(m-1)/(m-d-1))
adj_r2
```

Out[87]:

```
0.8075805382374766
```

```
Mean errors are nearly equals to zero
R2 score is 80% and adjusted R2 score is also almost same
If there are any other information like the state/region the student belongs to and the
```

Actionable Insights & Recommendations

```
--> Research feature makes more sense in prediction.
--> R2 score is 80 which states that this is a good model
--> Information like state/region or educational qualifications and work experience in the
field of intrest adds more value to the model
-->Few features had multicollinearity which makes the model to predict inappropriate
outputs so they are not much required to predict the output
--> Factors like Research,CGPA,TOFEL Score, University Rating effects the chance of the
student getting seat in the applied college from the given dataset
```