In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("Jamboree_Admission.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	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 [4]:
```
df.shape
```

Out[4]: (500, 9)

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.describe()
```

Out[6]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Resea
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000

In [7]:
```
df.isna().sum()
# no null values in the dataset
```

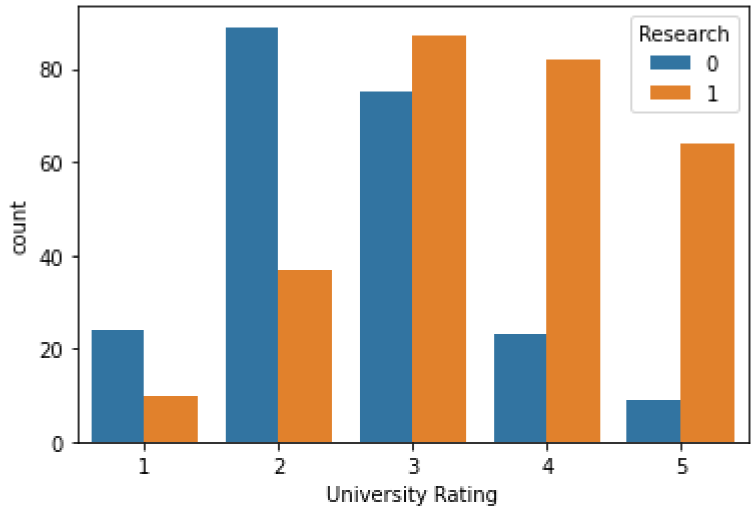
Out[7]:
```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [8]:
```
for i in df.columns:
 print(i,":", df[i].nunique())
```

```
Serial No. : 500
GRE Score : 49
TOEFL Score : 29
University Rating : 5
SOP : 9
LOR  : 9
CGPA : 184
Research : 2
Chance of Admit  : 61
```

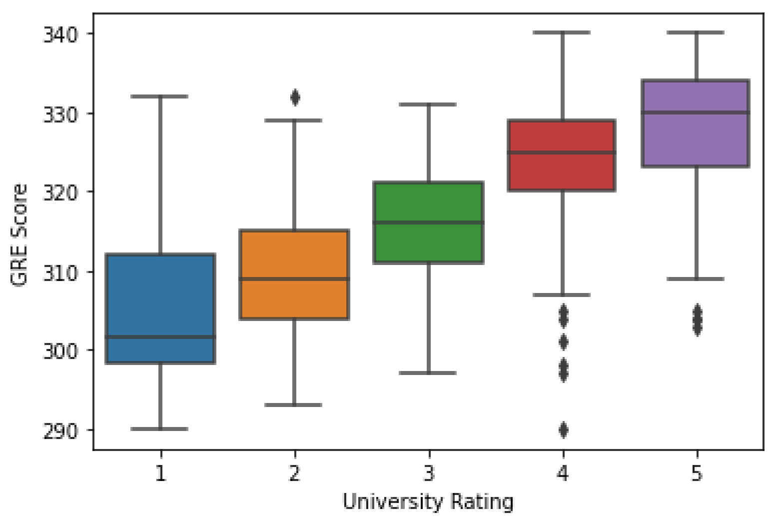
In [9]:
```
sns.countplot(data=df,x=df['University Rating'],hue=df['Research'])
```

Out[9]: <AxesSubplot:xlabel='University Rating', ylabel='count'>



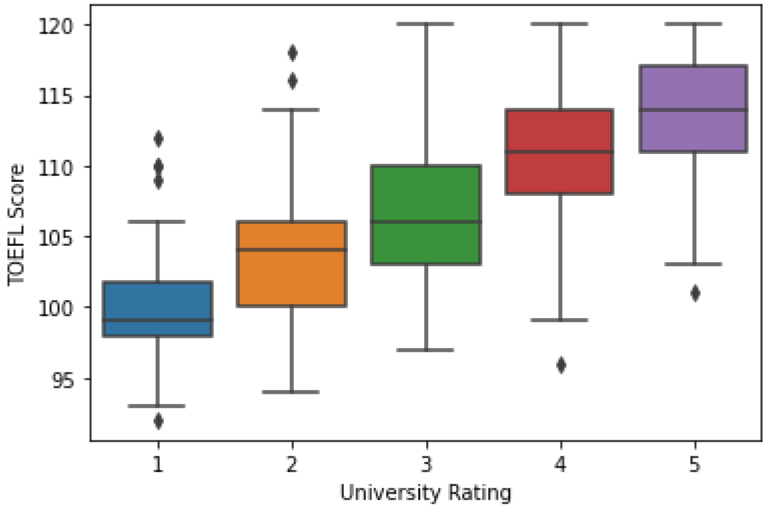
In [10]:
```
sns.boxplot(data=df,x=df['University Rating'],y=df['GRE Score'])
```

Out[10]: <AxesSubplot:xlabel='University Rating', ylabel='GRE Score'>



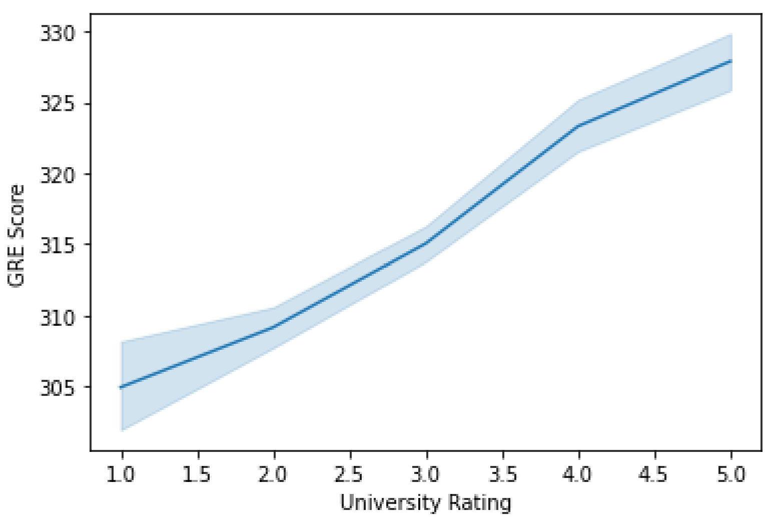
In [11]:
```
sns.boxplot(data=df,x=df['University Rating'],y=df['TOEFL Score'])
```

Out[11]: <AxesSubplot:xlabel='University Rating', ylabel='TOEFL Score'>



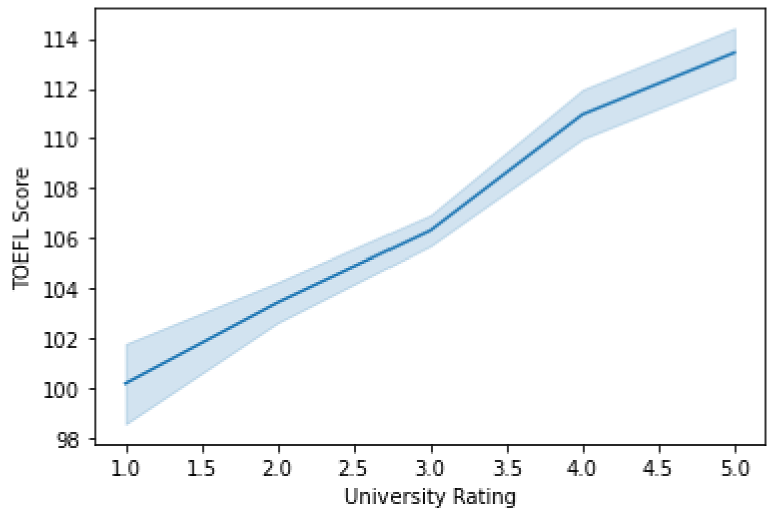
In [12]:
```
sns.lineplot(data=df,x=df['University Rating'],y=df['GRE Score'])
```

Out[12]: <AxesSubplot:xlabel='University Rating', ylabel='GRE Score'>



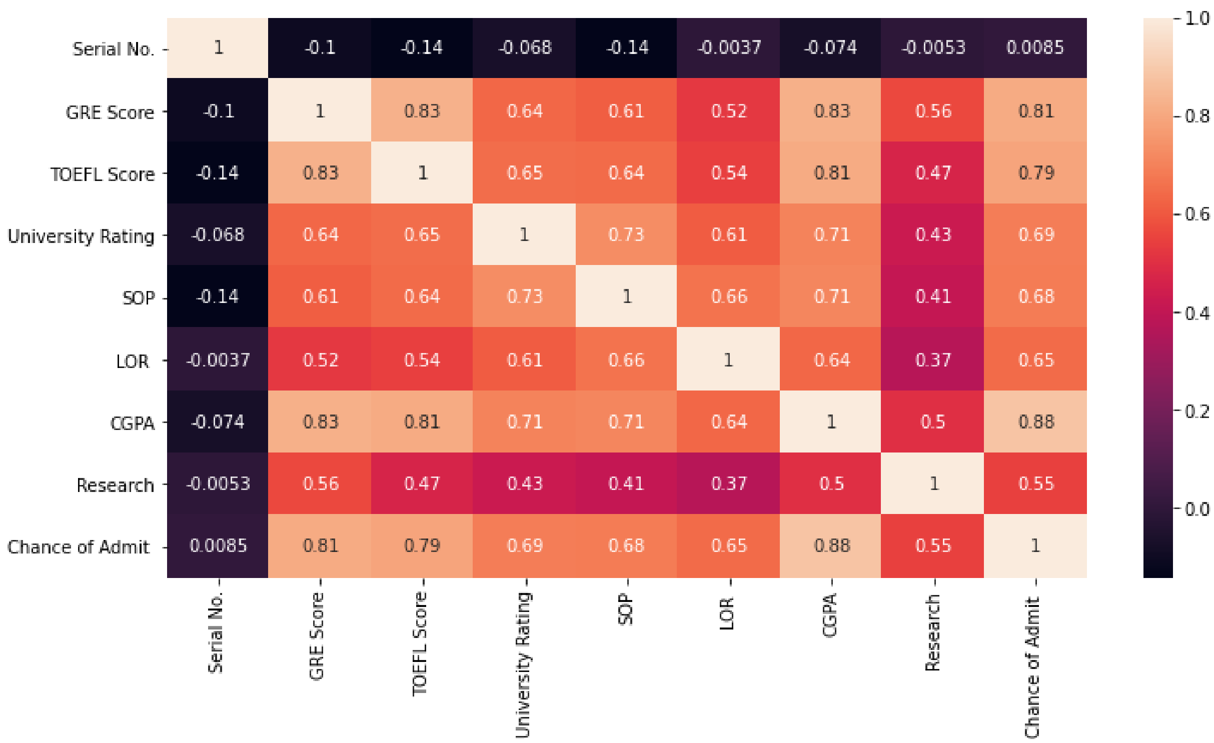
In [13]:
```
sns.lineplot(data=df,x=df['University Rating'],y=df['TOEFL Score'])
```

Out[13]: <AxesSubplot:xlabel='University Rating', ylabel='TOEFL Score'>



In [14]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df.corr(),annot=True)
```

Out[14]: <AxesSubplot:>



In [15]:
```
df[df.duplicated()]
#no duplicates
```

Out[15]:

Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit

In [16]:
```
# creating a Regression model
```

In [17]:
```
X=df[df.columns.drop(['Chance of Admit ','Serial No.'])]
X
```

Out[17]:

	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
...	...	...	...	...	...	...	...
495	332	108	5	4.5	4.0	9.02	1
496	337	117	5	5.0	5.0	9.87	1
497	330	120	5	4.5	5.0	9.56	1
498	312	103	4	4.0	5.0	8.43	0
499	327	113	4	4.5	4.5	9.04	0

500 rows × 7 columns

In [18]:
```
Y=df['Chance of Admit ']
```

In [19]:
```
from sklearn.model_selection import train_test_split
x_train , x_test, y_train, y_test = train_test_split(X,Y, test_size=0.2, random_s
```

In [20]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

In [21]:
```
#standarizing the data
```

In [22]:
```
standard_scaler=StandardScaler()
```

In [23]:
```
standard_scaler.fit(x_train)
```

Out[23]:
```
StandardScaler()
```

In [24]:
```
x_train=standard_scaler.transform(x_train)
```

In [25]:
```
x_test=standard_scaler.transform(x_test)
```

In [26]:
```
x_train.std(),x_test.std()
```

Out[26]:
```
(1.0, 0.9441314548393193)
```

In [27]:
```
X_=df[df.columns.drop(['Chance of Admit ','Serial No.'])]
X
```

Out[27]:

	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
...	...	...	...	...	...	...	...
495	332	108	5	4.5	4.0	9.02	1
496	337	117	5	5.0	5.0	9.87	1
497	330	120	5	4.5	5.0	9.56	1
498	312	103	4	4.0	5.0	8.43	0
499	327	113	4	4.5	4.5	9.04	0

500 rows × 7 columns

In [28]:
```
lr = LinearRegression()
```

In [29]:
```
lr.fit(x_train, y_train)
```

Out[29]:
```
LinearRegression()
```

In [30]:
```
lr.intercept_
```

Out[30]:
```
0.7209250000000001
```

In [31]:
```
x_train.columns
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-31-4391aa2eb072> in <module>
----> 1 x_train.columns

AttributeError: 'numpy.ndarray' object has no attribute 'columns'
```

In [32]:
```
lr.coef_
```

Out[32]:
```
array([0.02091007, 0.01965792, 0.00701103, 0.00304937, 0.01352815,
       0.07069295, 0.00988992])
```

In [33]:
```
lr.score(x_train, y_train)
```

Out[33]:
```
0.8215099192361265
```

In [34]:
```
y_hat = lr.predict(x_test)
```

In [35]:
```
lr.score(x_test, y_test)
```

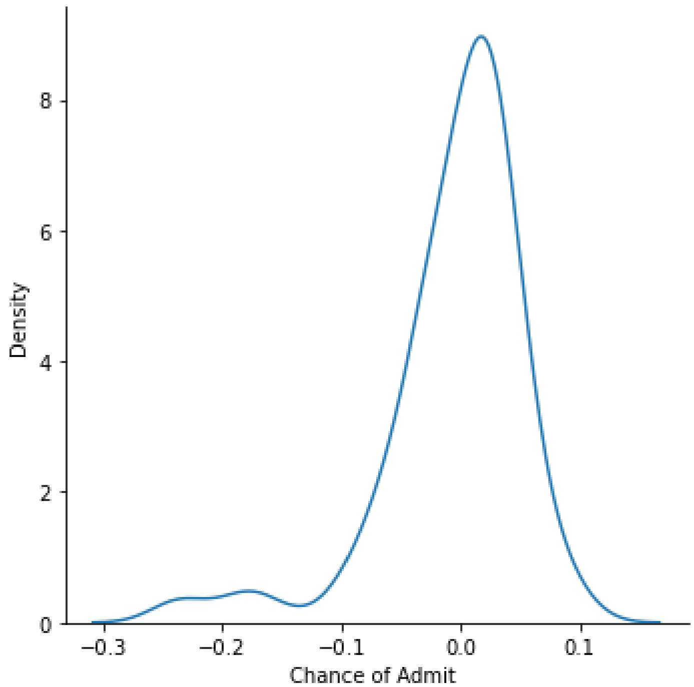
Out[35]:
```
0.8208741703103732
```

In [36]:
```
error = y_test - y_hat
# mean of residuals
np.mean(error)
```

Out[36]:
```
-0.005706590389232276
```

In [37]:
```
# plot of residual errors is following bell shaped distribution
# normality of residuals
sns.displot(error, kind = 'kde')
```

Out[37]: <seaborn.axisgrid.FacetGrid at 0x20fb58ec3d0>



In [38]:
```
#mean of residual errors
np.mean(error)
```

Out[38]: -0.005706590389232276

In [39]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_hat)
```

Out[39]: 0.003459098897136383

In [40]:
```
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
```

In [41]:
```
# building a Ridge regression model
```

In [42]:
```
ridge_lr=Ridge(alpha=0.1)
```

In [43]:
```
ridge_lr.fit(x_train,y_train)
```

Out[43]:
```
Ridge(alpha=0.1)
```

In [45]:
```
ridge_lr.coef_
```

Out[45]:
```
array([0.02093043, 0.01967003, 0.00701935, 0.00306466, 0.01353252,
       0.07062809, 0.00989163])
```

In [46]:
```
ridge_lr.intercept_
```

Out[46]:
```
0.7209250000000001
```

In [47]:
```
ridge_lr.score(x_train,y_train)
```

Out[47]:
```
0.821509872593231
```

In [48]:
```
y_hat=ridge_lr.predict(x_test)
```

In [49]:
```
ridge_lr.score(x_test, y_test)
```

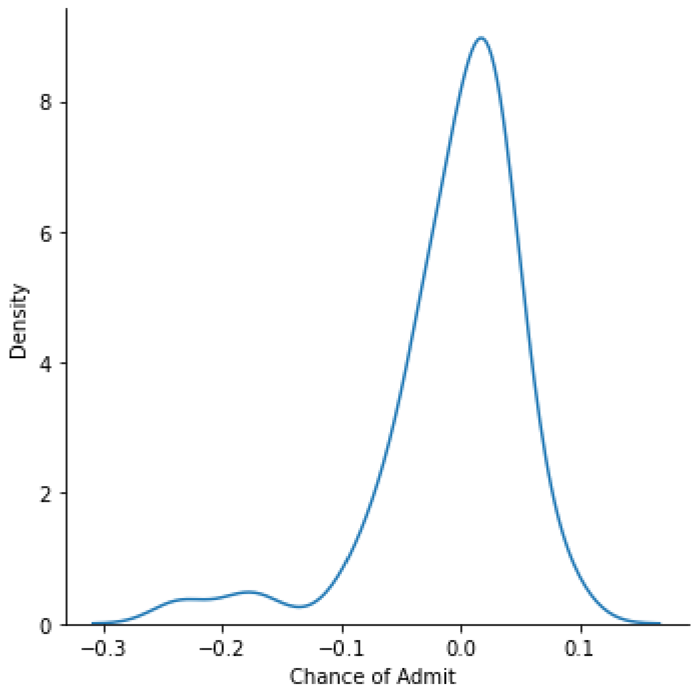
Out[49]:
```
0.8208640275953533
```

In [50]:
```
error=y_test-y_hat
#mean of residuals
np.mean(error)
```

Out[50]:
```
-0.005707834022735487
```

In [51]:
```
# normality of residuals
sns.displot(error, kind = 'kde')
```

Out[51]:
```
<seaborn.axisgrid.FacetGrid at 0x20fb6015040>
```



In [52]:
```
# building a lasso regression model
```

In [53]:
```
lasso_lr=Lasso(0.0001)
```

In [54]:
```
lasso_lr.fit(x_train,y_train)
```

Out[54]:
```
Lasso(alpha=0.0001)
```

In [56]:
```
lasso_lr.coef_
```

Out[56]:
```
array([0.02089882, 0.01962888, 0.006992  , 0.00302528, 0.01348122,
       0.07071527, 0.00983233])
```

In [57]:
```
lasso_lr.intercept_
```

Out[57]: 0.7209250000000001

In [58]:
```
lasso_lr.score(x_train,y_train)
```

Out[58]: 0.8215090781660604

In [59]:
```
y_hat=lasso_lr.predict(x_test)
```

In [60]:
```
lasso_lr.score(x_test, y_test)
```

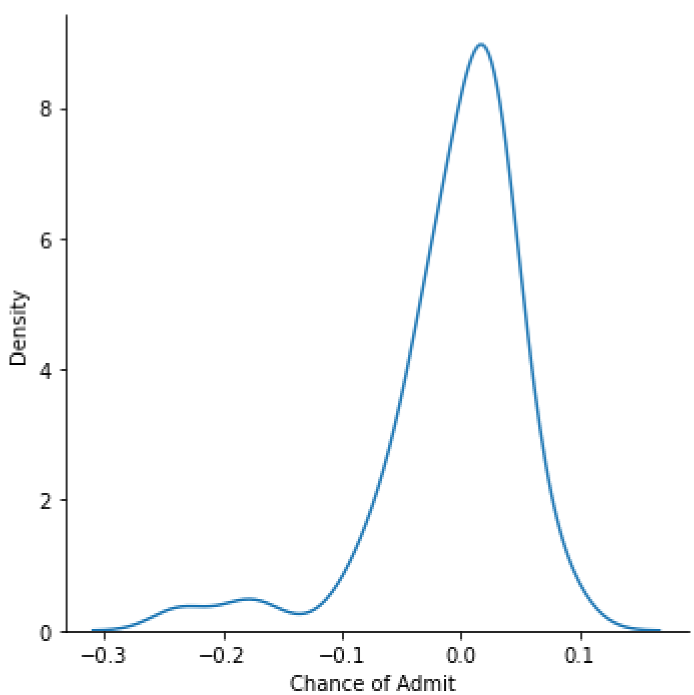
Out[60]: 0.8207818227394215

In [61]:
```
error=y_test-y_hat
#mean of residuals
np.mean(error)
```

Out[61]: -0.005698330605372871

In [62]:
```
# normality of residuals
sns.displot(error, kind = 'kde')
```

Out[62]: <seaborn.axisgrid.FacetGrid at 0x20fb6015850>



In [63]:
```
x_train=pd.DataFrame(data=x_train,columns=X.columns)
```

In [64]:
```
# Multicollinearity check by VIF score

from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
X_t = x_train
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

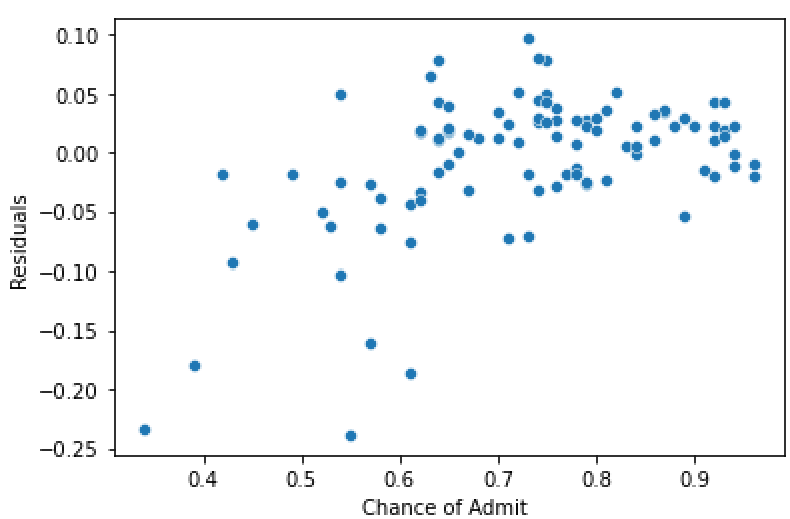
Out[64]:

	Features	VIF
0	GRE Score	4.88
5	CGPA	4.75
1	TOEFL Score	4.26
3	SOP	2.92
2	University Rating	2.80
4	LOR	2.08
6	Research	1.51

In [65]:
```
# The VIF score in less than 5 for all the variables so not rejecting any variabl
```

In [66]:
```
# Test for Homoscedasticity
sns.scatterplot(x = y_test, y=error)
plt.ylabel("Residuals")
```

Out[66]: Text(0, 0.5, 'Residuals')



In [67]:
```
# CGPA, Toefl score and GRE score are having the highest weights among all the fe
#so high value of this features may result in chance of admit
# There is no multicoleanrity, no pattern in residual errors, mean of residuals i
# The model accuracy is 82% while using Linear regression, Ridge or Lasso.
# some more fields like scholarship, test conducted by jamboree to evaluate the c
```

In [ ]: