In [1]:
```
!gdown 'https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/839/original/Jamboree_Admission.csv'
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/839/original/Jamboree_Admission.csv
 (https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/839/original/Jamboree_Admission.csv)
To: /Users/apple/Documents/Jamboree_Admission.csv
100%|████████████████████████████████████████| 16.2k/16.2k [00:00<00:00, 11.9MB/s]
```

In [104]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

In [107]:
```
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from scipy import stats
from statsmodels.compat import lzip
import statsmodels
import matplotlib.pyplot as plt
```

Problem statement --

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.

In [70]:
```
df=pd.read_csv('Jamboree_Admission.csv', None)
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/IPython/core/interactiveshell.py:3369: FutureWarning:
In a future version of pandas all arguments of read_csv except for the argument 'filepath_or_buffer' will be
keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/var/folders/2m/svsbyfss4h53t29lk1vcnvf40000gn/T/ipykernel_15147/1936837999.py:1: ParserWarning: Falling back
to the 'python' engine because the 'c' engine does not support sep=None with delim_whitespace=False; you can
avoid this warning by specifying engine='python'.
  df=pd.read_csv('Jamboree_Admission.csv', None)
```

In [71]:
```
df.head()
```

Out[71]:

	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 [72]:
```
cols=['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR', 'CGPA', 'Research']
```

In [6]:
```
df.shape
```

Out[6]:
```
(500, 9)
```

In [7]:
```
df.dtypes
```

Out[7]:
```
Serial No.             int64
GRE Score              int64
TOEFL Score            int64
University Rating      int64
SOP                  float64
LOR                  float64
CGPA                 float64
Research               int64
Chance of Admit      float64
dtype: object
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

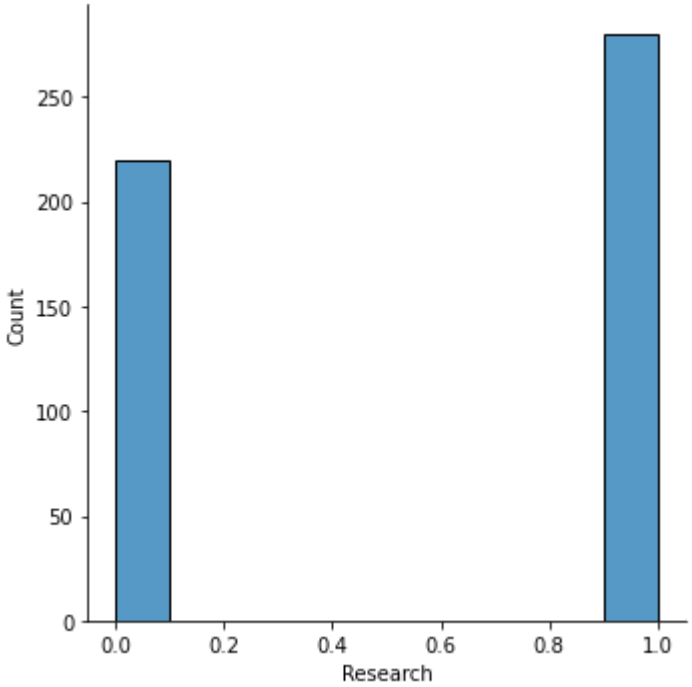
In [9]:
```
df.describe(include='all')
```

Out[9]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

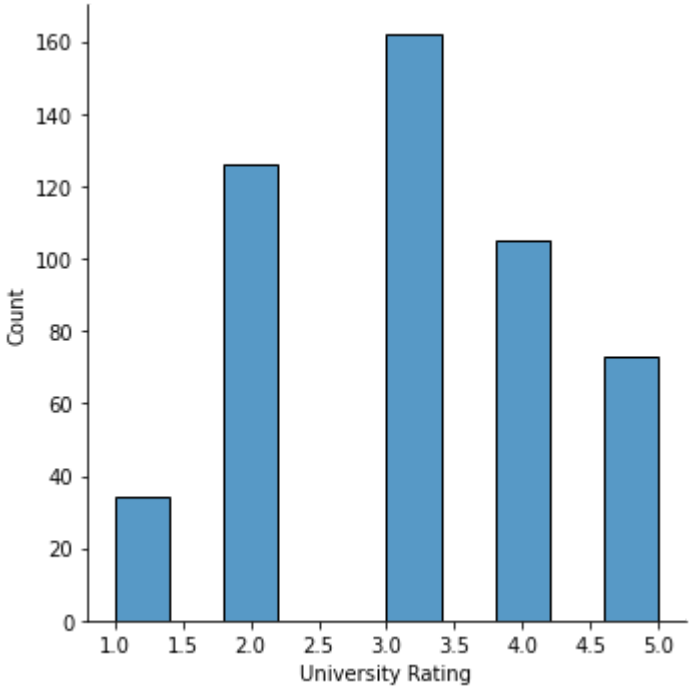
In [10]:
```
sns.displot(df['Research'] )
```

Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x7fd473710910>
```



In [12]:
```
sns.displot(df['University Rating'] )
```

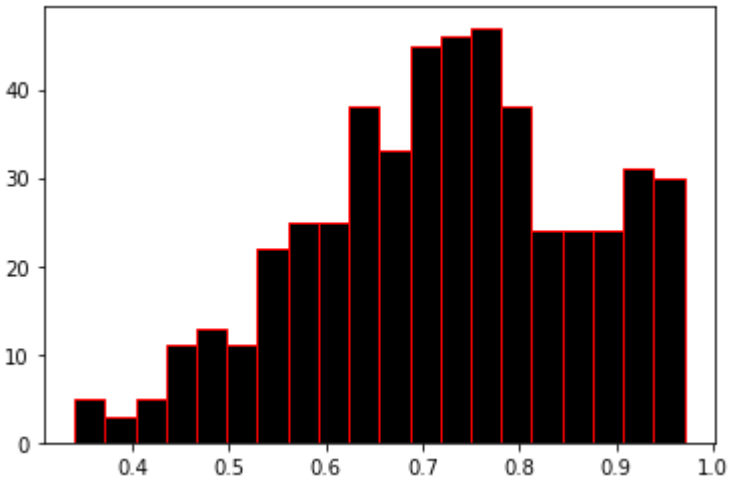
Out[12]:
```
<seaborn.axisgrid.FacetGrid at 0x7fd47758e1c0>
```



In [ ]:
```
plt.hist(df['SOP'], color = 'black', edgecolor = 'red',
         bins = int(20))
```

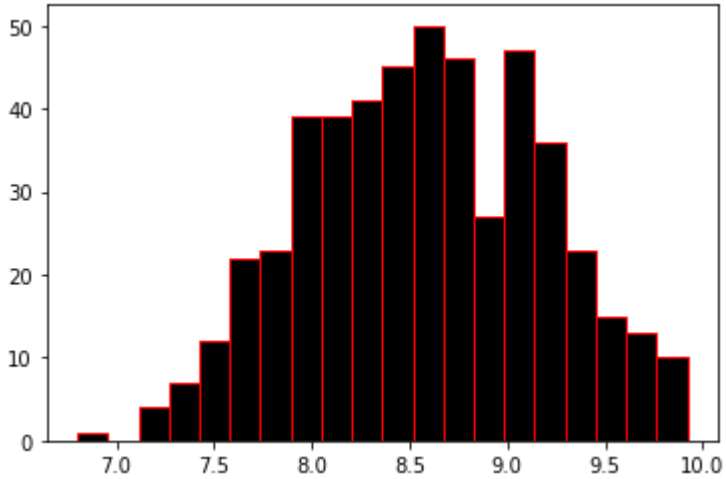
In [65]:
```
plt.hist(df['Chance of Admit'], color = 'black', edgecolor = 'red',
         bins = int(20))
```

Out[65]:
```
(array([ 5.,  3.,  5., 11., 13., 11., 22., 25., 25., 38., 33., 45., 46.,
        47., 38., 24., 24., 24., 31., 30.]),
 array([0.34  , 0.3715, 0.403 , 0.4345, 0.466 , 0.4975, 0.529 , 0.5605,
        0.592 , 0.6235, 0.655 , 0.6865, 0.718 , 0.7495, 0.781 , 0.8125,
        0.844 , 0.8755, 0.907 , 0.9385, 0.97  ]),
 <BarContainer object of 20 artists>)
```



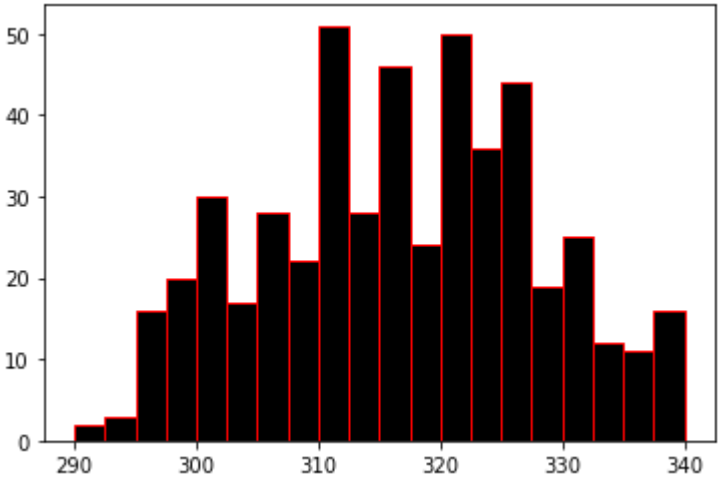
In [56]:
```
plt.hist(df['CGPA'], color = 'black', edgecolor = 'red',
         bins = int(20))
```

Out[56]:
```
(array([ 1.,  0.,  4.,  7., 12., 22., 23., 39., 39., 41., 45., 50., 46.,
        27., 47., 36., 23., 15., 13., 10.]),
 array([6.8  , 6.956, 7.112, 7.268, 7.424, 7.58 , 7.736, 7.892, 8.048,
        8.204, 8.36 , 8.516, 8.672, 8.828, 8.984, 9.14 , 9.296, 9.452,
        9.608, 9.764, 9.92 ]),
 <BarContainer object of 20 artists>)
```



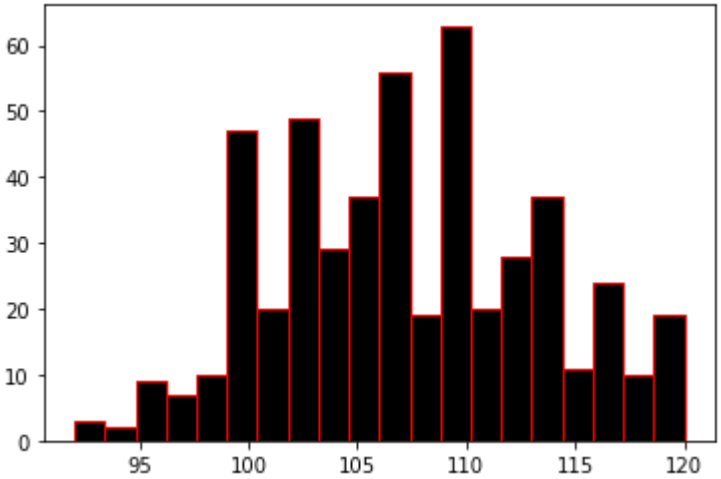
In [43]:
```
plt.hist(df['GRE Score'], color = 'black', edgecolor = 'red',
         bins = int(20))
```

Out[43]:
```
(array([ 2.,  3., 16., 20., 30., 17., 28., 22., 51., 28., 46., 24., 50.,
        36., 44., 19., 25., 12., 11., 16.]),
 array([290. , 292.5, 295. , 297.5, 300. , 302.5, 305. , 307.5, 310. ,
        312.5, 315. , 317.5, 320. , 322.5, 325. , 327.5, 330. , 332.5,
        335. , 337.5, 340. ]),
 <BarContainer object of 20 artists>)
```



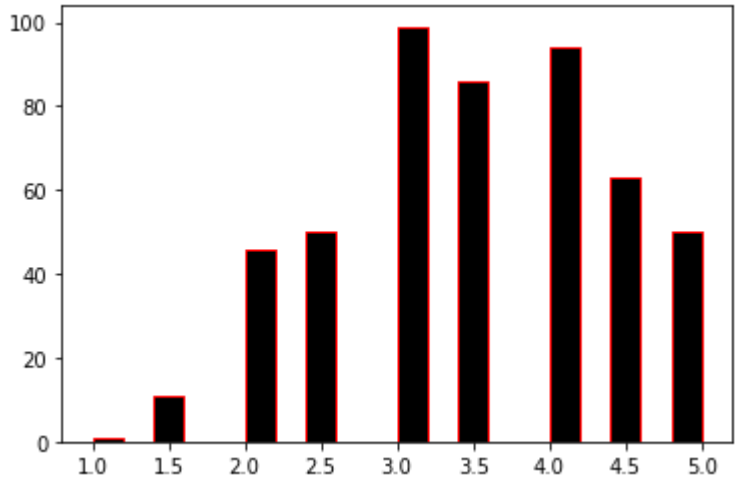
In [42]:
```
plt.hist(df['TOEFL Score'], color = 'black', edgecolor = 'red',
         bins = int(20))
```

Out[42]:
```
(array([ 3.,  2.,  9.,  7., 10., 47., 20., 49., 29., 37., 56., 19., 63.,
        20., 28., 37., 11., 24., 10., 19.]),
 array([ 92. ,  93.4,  94.8,  96.2,  97.6,  99. , 100.4, 101.8, 103.2,
        104.6, 106. , 107.4, 108.8, 110.2, 111.6, 113. , 114.4, 115.8,
        117.2, 118.6, 120. ]),
 <BarContainer object of 20 artists>)
```



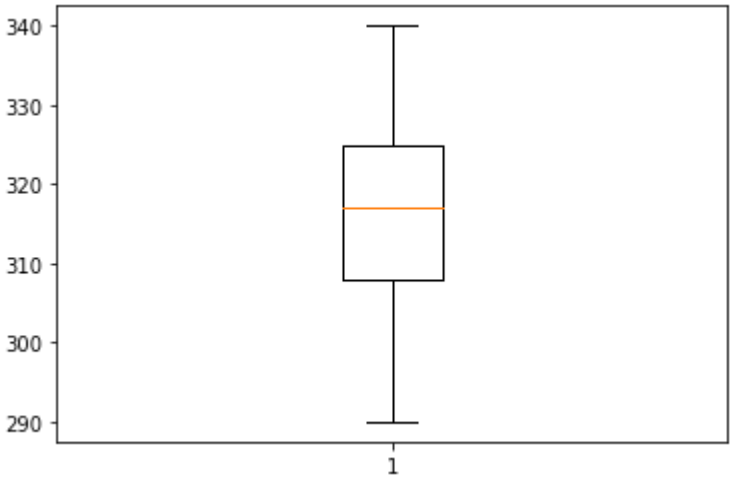
In [64]:
```
plt.hist(df['LOR'], color = 'black', edgecolor = 'red',
         bins = int(20))
```

Out[64]:
```
(array([ 1.,  0., 11.,  0.,  0., 46.,  0., 50.,  0.,  0., 99.,  0., 86.,
         0.,  0., 94.,  0., 63.,  0., 50.]),
 array([1. , 1.2, 1.4, 1.6, 1.8, 2. , 2.2, 2.4, 2.6, 2.8, 3. , 3.2, 3.4,
        3.6, 3.8, 4. , 4.2, 4.4, 4.6, 4.8, 5. ]),
 <BarContainer object of 20 artists>)
```



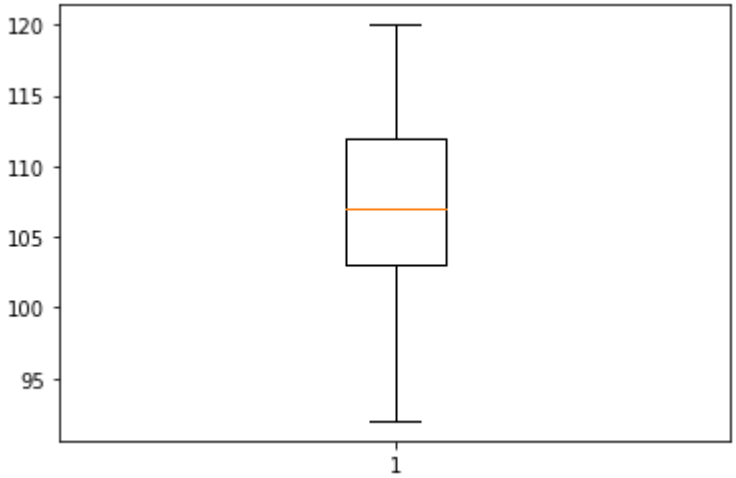
In [52]:
```
plt.boxplot(df['GRE Score'])
```

Out[52]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7fd45f0c3b20>,
  <matplotlib.lines.Line2D at 0x7fd45f0c3df0>],
 'caps': [<matplotlib.lines.Line2D at 0x7fd45f0c3ca0>,
  <matplotlib.lines.Line2D at 0x7fd45f0c3fd0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fd45f0c38e0>],
 'medians': [<matplotlib.lines.Line2D at 0x7fd45f0d02e0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fd45f0d05b0>],
 'means': []}
```



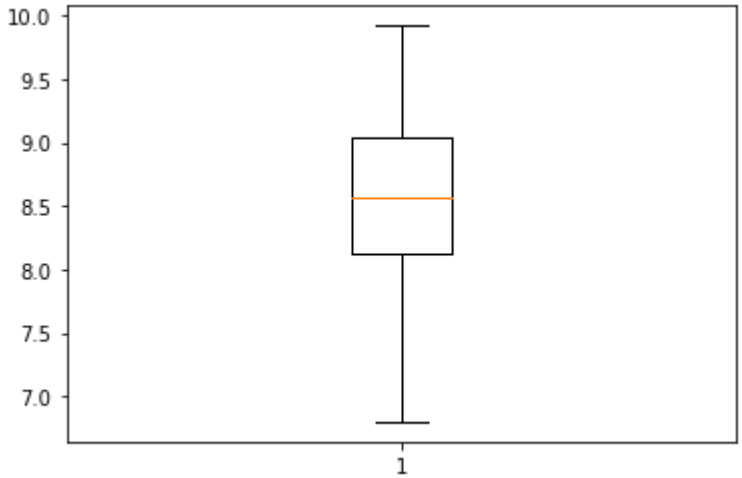
In [48]:
```
plt.boxplot(df['TOEFL Score'])
```

Out[48]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7fd45e294bb0>,
  <matplotlib.lines.Line2D at 0x7fd45e294e80>],
 'caps': [<matplotlib.lines.Line2D at 0x7fd45e29d160>,
  <matplotlib.lines.Line2D at 0x7fd45e29d430>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fd45e2948e0>],
 'medians': [<matplotlib.lines.Line2D at 0x7fd45e29d700>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fd45e29d970>],
 'means': []}
```



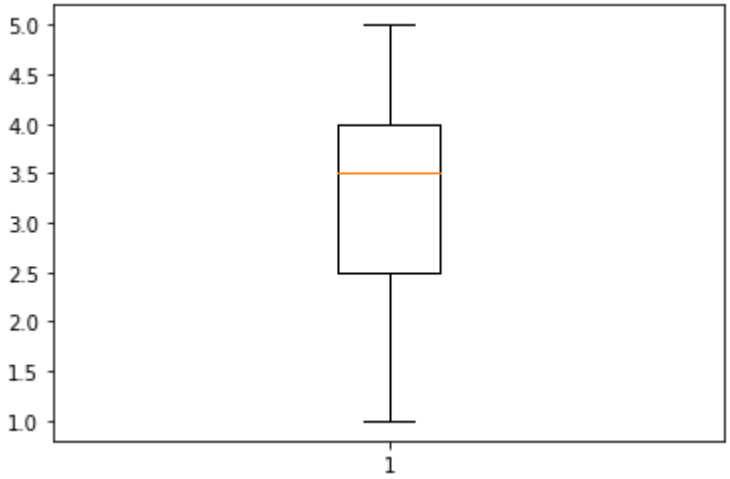
In [49]:
```
plt.boxplot(df['CGPA'])
```

Out[49]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7fd45e2d4730>,
  <matplotlib.lines.Line2D at 0x7fd45e2d4ac0>],
 'caps': [<matplotlib.lines.Line2D at 0x7fd45e2d4d30>,
  <matplotlib.lines.Line2D at 0x7fd45e2d4e50>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fd45e2d4460>],
 'medians': [<matplotlib.lines.Line2D at 0x7fd45e2df160>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fd45e2df430>],
 'means': []}
```



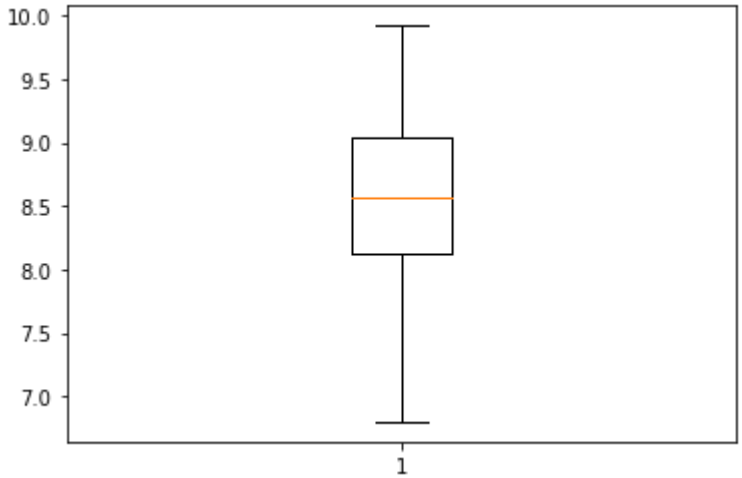
In [57]:
```
plt.boxplot(df['SOP'])
```

Out[57]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7fd45f5ce400>,
  <matplotlib.lines.Line2D at 0x7fd45f5ce6d0>],
 'caps': [<matplotlib.lines.Line2D at 0x7fd45f5ce9a0>,
  <matplotlib.lines.Line2D at 0x7fd45f5cec70>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fd45f5b2f70>],
 'medians': [<matplotlib.lines.Line2D at 0x7fd45f5cef40>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fd45f635250>],
 'means': []}
```



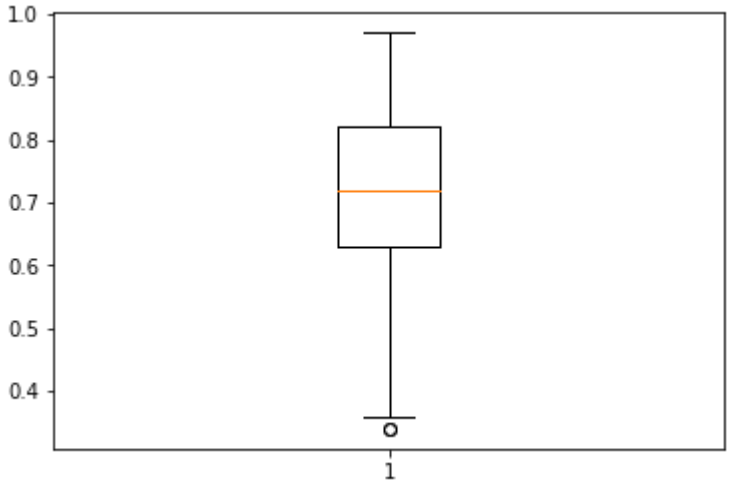
In [59]:
```
plt.boxplot(df['CGPA'])
```

Out[59]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7fd45f7a0c10>,
  <matplotlib.lines.Line2D at 0x7fd45f7a0ee0>],
 'caps': [<matplotlib.lines.Line2D at 0x7fd45f7ad1f0>,
  <matplotlib.lines.Line2D at 0x7fd45f7ad4c0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fd45f7978b0>],
 'medians': [<matplotlib.lines.Line2D at 0x7fd45f7ad790>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fd45f7ada60>],
 'means': []}
```



In [62]:
```
plt.boxplot(df['Chance of Admit'])
```

Out[62]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7fd45f8b19a0>,
  <matplotlib.lines.Line2D at 0x7fd45f8b1c70>],
 'caps': [<matplotlib.lines.Line2D at 0x7fd45f8b1f70>,
  <matplotlib.lines.Line2D at 0x7fd45f8bc280>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fd45f8b16d0>],
 'medians': [<matplotlib.lines.Line2D at 0x7fd45f8bc550>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fd45f8bc820>],
 'means': []}
```



Linear Regression

In [84]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
```

In [73]:
```
X_train, X_test, y_train, y_test = train_test_split(df[cols], df['Chance of Admit'], test_size = 0.20)
```

In [87]:
```
pipeline = make_pipeline(StandardScaler(),LinearRegression())
pipeline.fit(X_train, y_train)
```

Out[87]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('linearregression', LinearRegression())])
```

In [88]:
```
print(regr.score(X_test, y_test))
```

```
0.8070648243322358
```

In [90]:
```
y_train_pred = pipeline.predict(X_train)
y_test_pred = pipeline.predict(X_test)
#
# Mean Squared Error
#
print('MSE train: %.3f, test: %.3f' % (mean_squared_error(y_train, y_train_pred),
                mean_squared_error(y_test, y_test_pred)))
#
# R-Squared
#
print('R^2 train: %.3f, test: %.3f' % (r2_score(y_train, y_train_pred), r2_score(y_test, y_test_pred)))
```

```
MSE train: 0.003, test: 0.004
R^2 train: 0.825, test: 0.807
```

In [76]:
```
regr.coef_
```

Out[76]:
```
array([0.00199163, 0.0026226 , 0.00418216, 0.00316408, 0.01696329,
       0.12168425, 0.02508302])
```

In [78]:
```
regr.intercept_
```

Out[78]:
```
-1.3315348237710944
```

Ridge and Lasso regression

In [92]:
```
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
```

In [91]:
```
pipeline = make_pipeline(StandardScaler(), Ridge(alpha=1.0))
pipeline.fit(X_train, y_train)
#
# Calculate the predicted value for training and test dataset
#
y_train_pred = pipeline.predict(X_train)
y_test_pred = pipeline.predict(X_test)
#
# Mean Squared Error
#
print('MSE train: %.3f, test: %.3f' % (mean_squared_error(y_train, y_train_pred),
                mean_squared_error(y_test, y_test_pred)))
#
# R-Squared
#
print('R^2 train: %.3f, test: %.3f' % (r2_score(y_train, y_train_pred), r2_score(y_test, y_test_pred)))
```

```
MSE train: 0.003, test: 0.004
R^2 train: 0.825, test: 0.807
```

In [93]:
```
lasso = Lasso(alpha=1.0)
#
# Fit the Lasso model
#
lasso.fit(X_train, y_train)
#
# Create the model score
#
lasso.score(X_test, y_test), lasso.score(X_train, y_train)
```

Out[93]:
```
(0.26174317424038895, 0.26129026401064237)
```

Testing the assumptions of the linear regression model (50 Points)

Multicollinearity check by VIF score

In [99]:
```
# compute the vif for all given features
def compute_vif(considered_features):

    X = df[considered_features]
    # the calculation of variance inflation requires a constant
    X['intercept'] = 1

    # create dataframe to store vif values
    vif = pd.DataFrame()
    vif["Variable"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    vif = vif[vif['Variable']!='intercept']
    return vif
```

In [100]:
```
# features to consider removing
considered_features = ['Serial No.','GRE Score','TOEFL Score','University Rating','SOP','LOR','CGPA','Research',

# compute vif
compute_vif(considered_features).sort_values('VIF', ascending=False)
```

Out[100]:

	Variable	VIF
6	CGPA	6.234806
8	Chance of Admit	5.861208
1	GRE Score	4.606284
2	TOEFL Score	4.057830
4	SOP	2.899715
3	University Rating	2.635335
5	LOR	2.111309
7	Research	1.536124
0	Serial No.	1.100970

Removing feature CGPA

In [102]:
```
# features to consider removing
considered_features = ['Serial No.','GRE Score','TOEFL Score','University Rating','SOP','LOR','Research','Chance

# compute vif
compute_vif(considered_features).sort_values('VIF', ascending=False)
```

Out[102]:

	Variable	VIF
7	Chance of Admit	4.510142
1	GRE Score	4.295721
2	TOEFL Score	3.942884
4	SOP	2.839826
3	University Rating	2.613175
5	LOR	2.092097
6	Research	1.531172
0	Serial No.	1.098885

mean of residuals

In [115]:
```
mean_res=np.mean(y_test-y_test_pred)
```

In [123]:
```
res1=y_test-y_test_pred
```

In [116]:
```
res=np.sum(y_test-y_test_pred)
```

In [ ]:
```

```

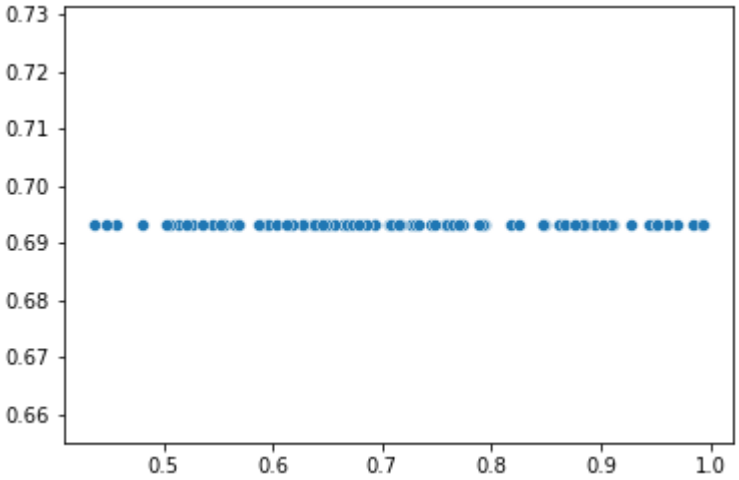
In [117]:
```
mean_res, res
```

Out[117]:
```
(0.006932114226125306, 0.6932114226125305)
```

Test for Homoscedasticity

In [118]:
```
p = sns.scatterplot(y_test_pred,res)
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



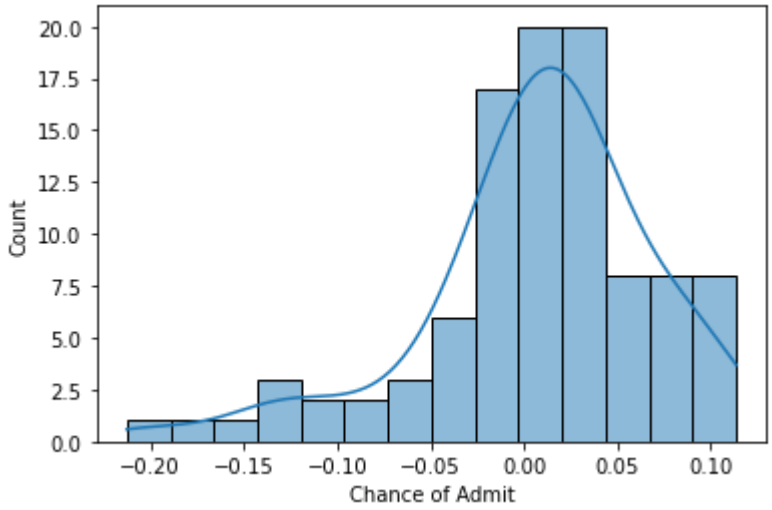
Checking heteroscedasticity : Using Goldfeld Quandt we test for heteroscedasticity.

Null Hypothesis: Error terms are homoscedastic Alternative Hypothesis: Error terms are heteroscedastic.

Normality of residuals

In [126]:
```
sns.histplot(res1, kde=True)
```

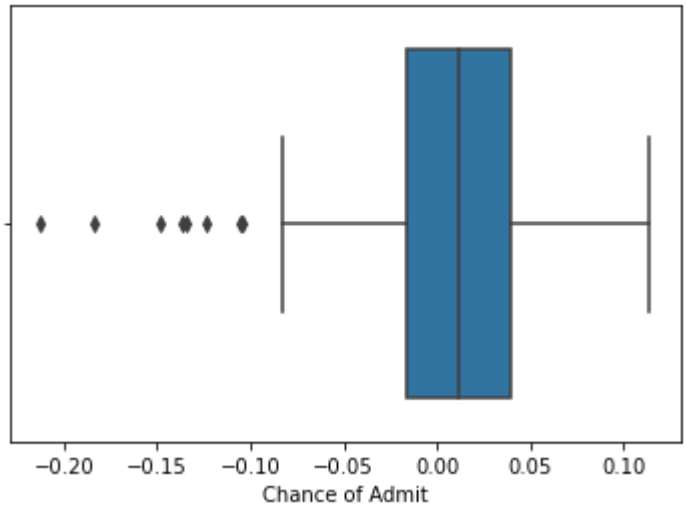
Out[126]:
```
<AxesSubplot:xlabel='Chance of Admit', ylabel='Count'>
```



In [127]:
```
sns.boxplot(res1)
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[127]:
```
<AxesSubplot:xlabel='Chance of Admit'>
```



Model performance evaluation

MAE, RMSE, R2, Adj R2

In [129]:
```
MAE= print('MSE train: %.3f, test: %.3f' % (mean_squared_error(y_train, y_train_pred),
                mean_squared_error(y_test, y_test_pred)))
#
```

```
MSE train: 0.003, test: 0.004
```

In [132]:
```
RMSE=print('RMSE train: %.3f, test: %.3f' % (np.sqrt(mean_squared_error(y_train, y_train_pred)),
                np.sqrt(mean_squared_error(y_test, y_test_pred))))
```

```
RMSE train: 0.059, test: 0.062
```

In [133]:
```
# R-Squared
#
print('R^2 train: %.3f, test: %.3f' % (r2_score(y_train, y_train_pred), r2_score(y_test, y_test_pred)))
```

```
R^2 train: 0.825, test: 0.807
```

Actionable Insights & Recommendations

```
1- model train and test scores are very good
2- dataset do not have any outliers
3- dateset does to contain null values
4- CGPA is most significant and GRE score is least significant feature
5- data is normally distributed.
```