1.Loading Dataset into tool

```
from google.colab import files uploaded
= files.upload()
```

Choose Files No file chosen

```
import pandas as pd  import numpy
as np import matplotlib.pyplot
as plt import seaborn as sns
import warnings
warnings.filterwarnings('ignore'
)
```

```
data = pd.read_csv("abalone.csv")
```

2.Performing Visualization Univariate

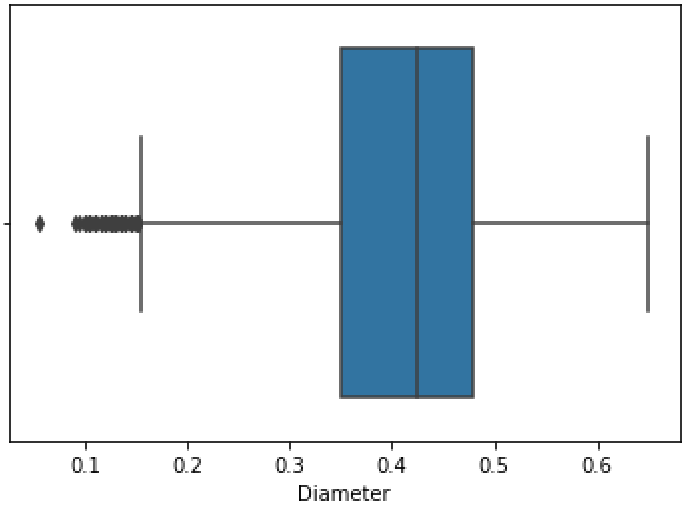
Analysis

```
data.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

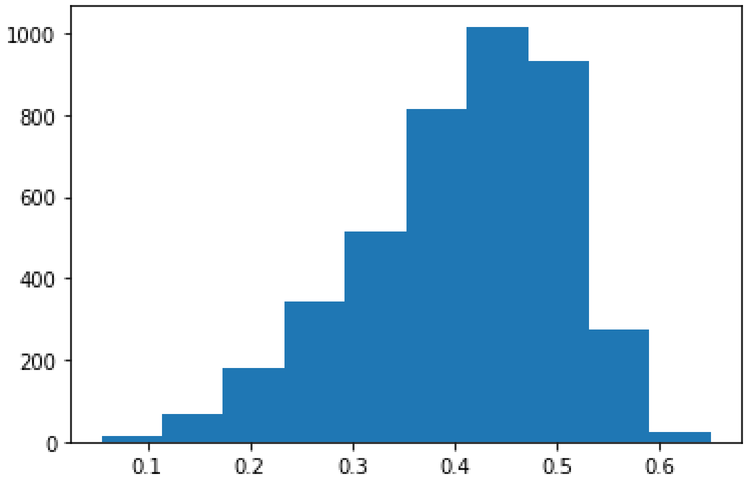
```
sns.boxplot(data['Diameter'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6fb14bf10>
```



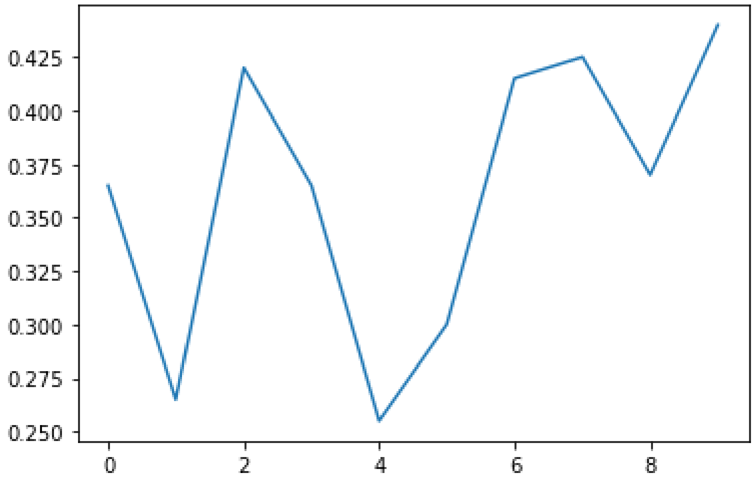

```
plt.hist(data['Diameter'])
```

```
(array([  13.,   66.,  180.,  344.,  513.,  812., 1017.,  934.,  275.,
          23.]),
 array([0.055 , 0.1145, 0.174 , 0.2335, 0.293 , 0.3525, 0.412 , 0.4715,
        0.531 , 0.5905, 0.65  ]),
<a list of 10 Patch objects>)
```



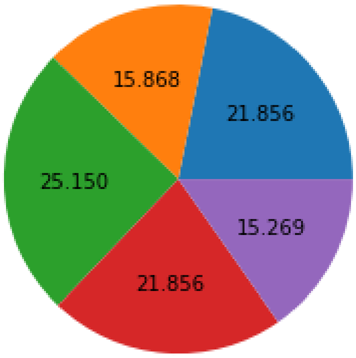

```
plt.plot(data['Diameter'].head(10))
```

```
[<matplotlib.lines.Line2D at 0x7fc6fac11c10>]
```



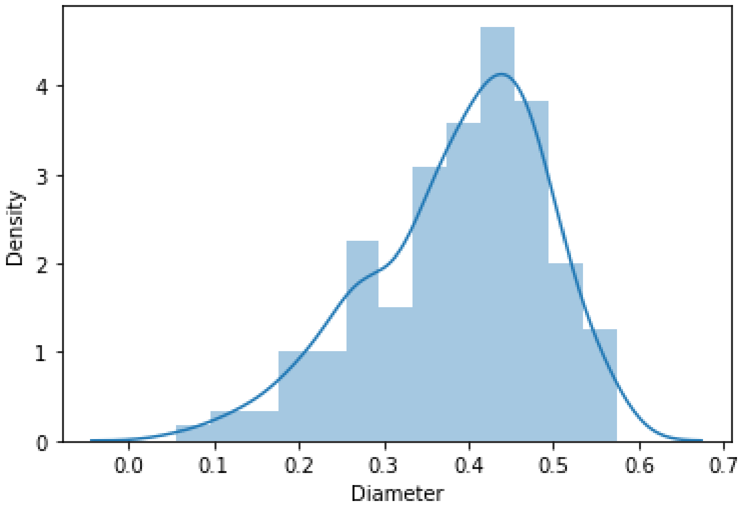

```
plt.pie(data['Diameter'].head(),autopct='%.3f')
```

```
([<matplotlib.patches.Wedge at 0x7fc6fab2a110>,
  <matplotlib.patches.Wedge at 0x7fc6fab2a950>,
  <matplotlib.patches.Wedge at 0x7fc6fab32210>,
  <matplotlib.patches.Wedge at 0x7fc6fab32b10>,
  <matplotlib.patches.Wedge at 0x7fc6faabd690>],
 [Text(0.8507215626110557, 0.6973326486753676, ''),
  Text(-0.32611344931648134, 1.0505474849691026, ''),
  Text(-1.0998053664078908, -0.02069193128747144, ''),
  Text(-0.08269436219656089, -1.096887251480709, ''),
  Text(0.9758446362287218, -0.5076684409569241, '')],
 [Text(0.46402994324239394, 0.3803632629138369, '21.856'),
  Text(-0.17788006326353525, 0.5730259008922377, '15.868'),
  Text(-0.5998938362224858, -0.011286507974984419, '25.150'),
  Text(-0.045106015743578656, -0.5983021371712958, '21.856'),
Text(0.5322788924883937, -0.2769100587037768, '15.269')])
```

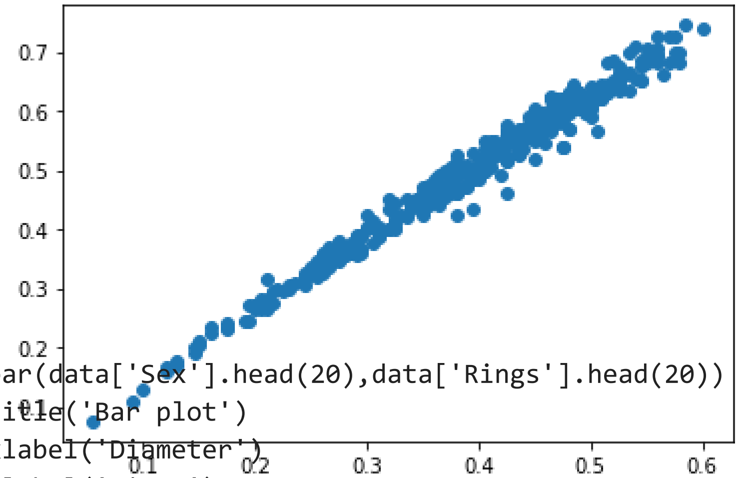



```
sns.distplot(data['Diameter'].head(300))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6fab6e6d0>
```

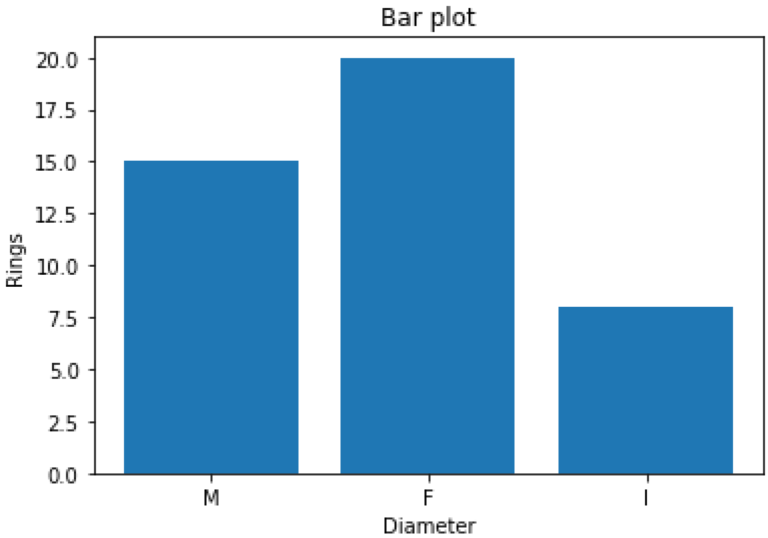



```
plt.scatter(data['Diameter'].head(400),data['Length'].head(400))
<matplotlib.collections.PathCollection at 0x7fc6fa9c7750>
```



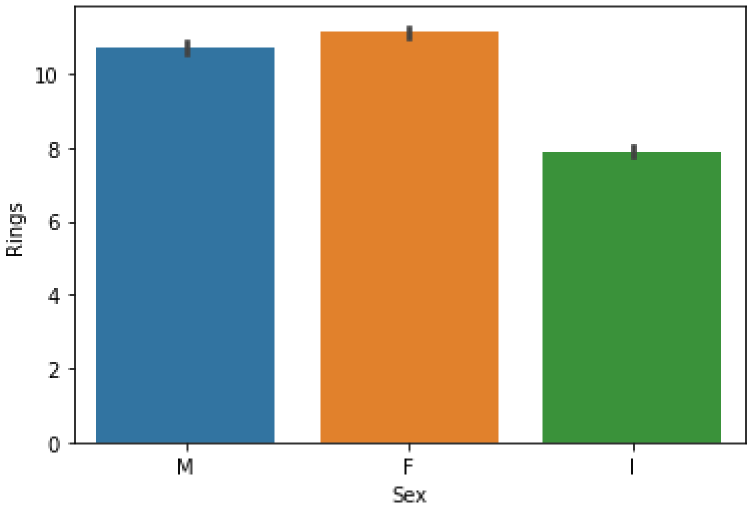

```
plt.bar(data['Sex'].head(20),data['Rings'].head(20))
plt.title('Bar plot')
plt.xlabel('Diameter')
plt.ylabel('Rings')
```

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

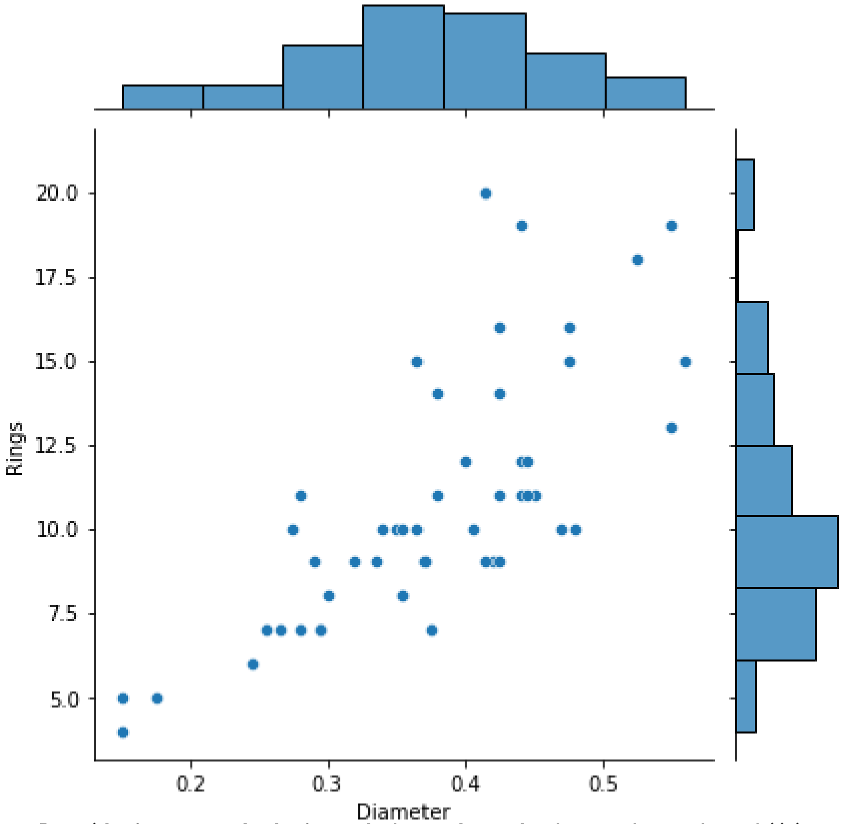



```
sns.barplot(data['Sex'], data['Rings'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6fab01c50>
```

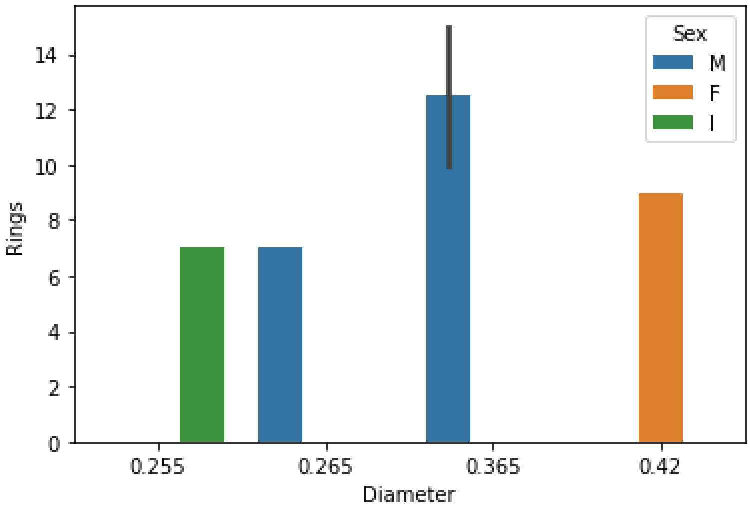



```
sns.jointplot(data['Diameter'].head(50),data['Rings'].head(100))
<seaborn.axisgrid.JointGrid at 0x7fc6fa886b90>
```




```
sns.barplot('Diameter','Rings',hue='Sex',data=data.head())
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f7f29510>

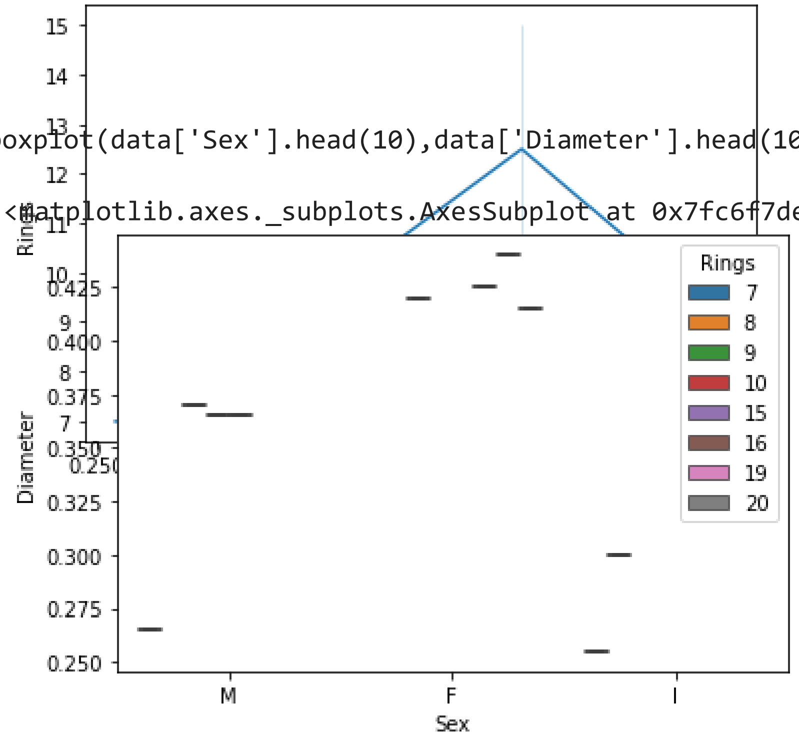



```
sns.lineplot(data['Diameter'].head(),data['Rings'].head())
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f7e5a150>

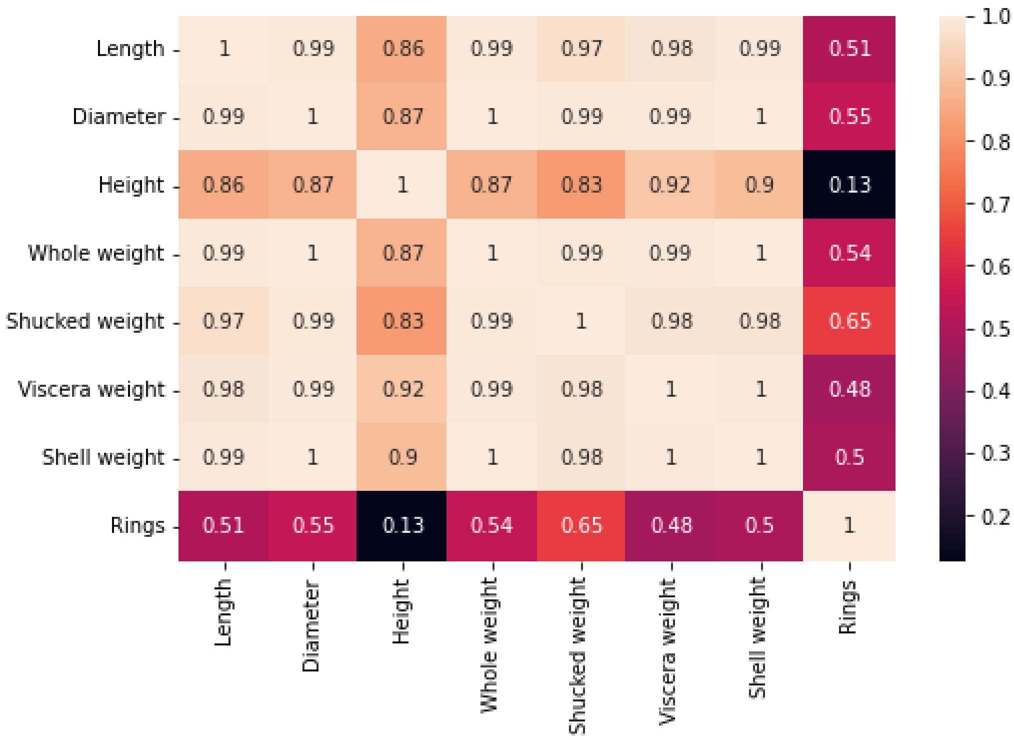
```
sns.boxplot(data['Sex'].head(10),data['Diameter'].head(10),data['Rings'].head(10))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f7de4050>

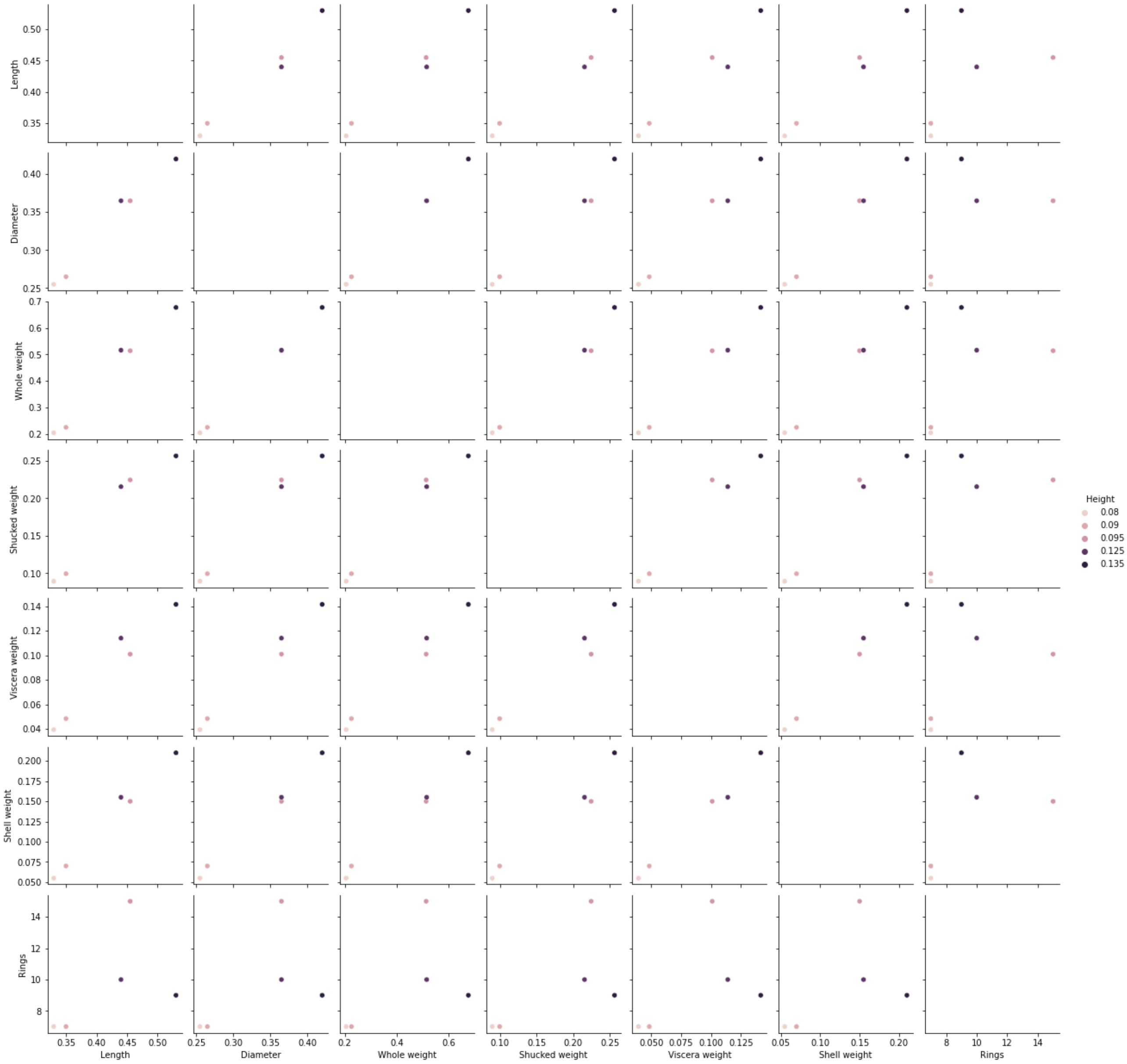



```
fig=plt.figure(figsize=(8,5)) sns.heatmap(data.head().corr(),annot=True)
```

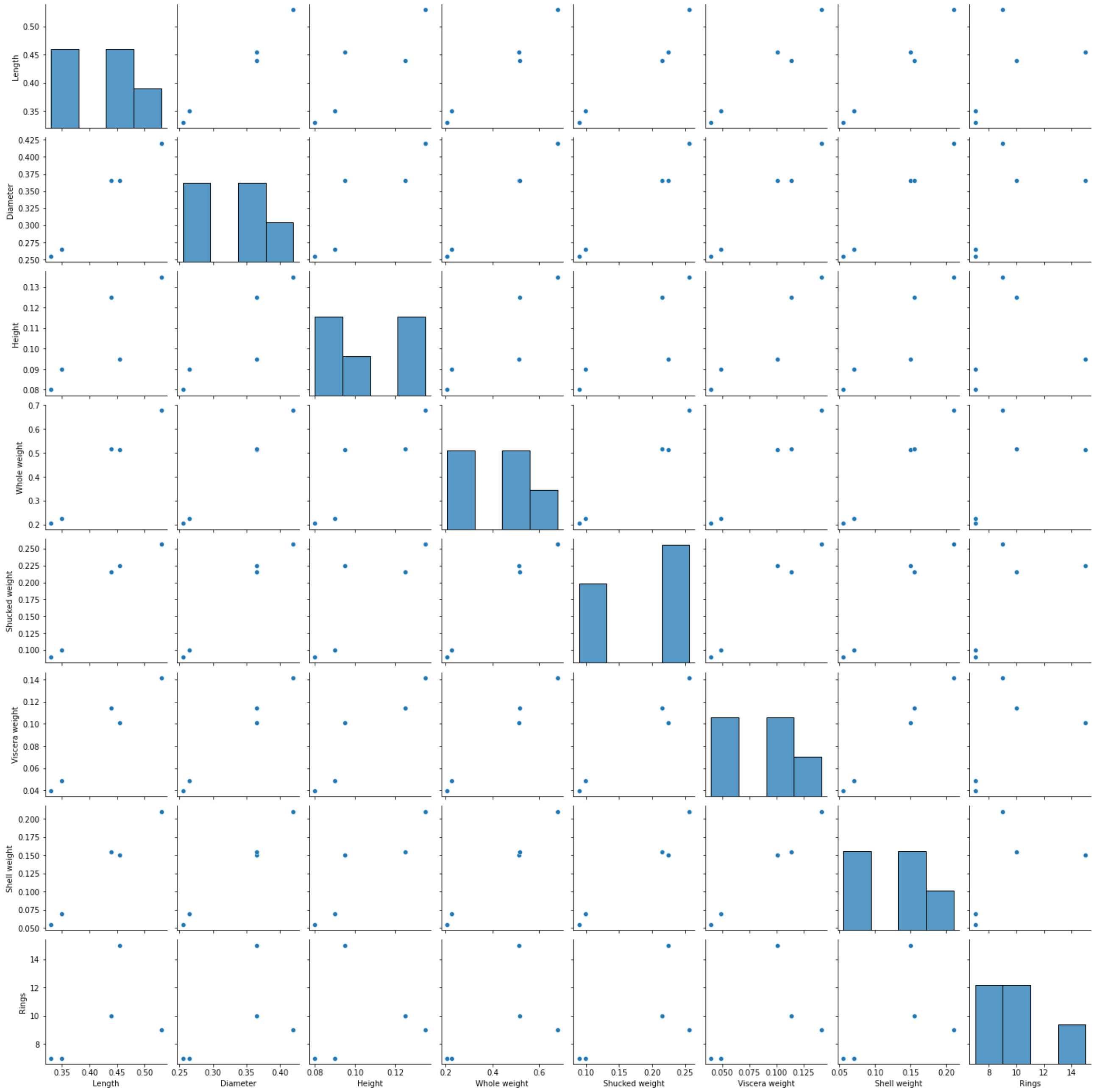
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6fb08a190>




```
sns.pairplot(data.head(),hue='Height') <seaborn.axisgrid.PairGrid at 0x7fc6f7b0fd10>
```



sns.pairplot(data.head()) <seaborn.axisgrid.PairGrid at 0x7fc6f653ef10>



3.Perform Descriptive Statistics on the dataset

```
data.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
data.tail()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings

4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
0   Sex             4177 non-null   object
1   Length          4177 non-null   float64
2   Diameter        4177 non-null   float64
3   Height          4177 non-null   float64
4   Whole weight    4177 non-null   float64
5   Shucked weight  4177 non-null   float64
6   Viscera weight  4177 non-null   float64
7   Shell weight    4177 non-null   float64  8   Rings           4177 non-
    null   int64   dtypes: float64(7), int64(1), object(1) memory usage:
    293.8+ KB
```

```
data.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	

```
data.mode().T
```

	0	1
Sex	M	NaN
Length	0.55	0.625
Diameter	0.45	NaN
Height	0.15	NaN
Whole weight	0.2225	NaN
Shucked weight	0.175	NaN
Viscera weight	0.1715	NaN
Shell weight	0.275	NaN
Rings	9.0	NaN

```
data.shape
```

```
(4177, 9)
```

```
data.kurt()
```

```
Length             0.064621
Diameter          -0.045476
Height            76.025509
Whole weight      -0.023644
Shucked weight     0.595124
Viscera weight     0.084012
Shell weight       0.531926
Rings              2.330687
dtype: float64
```

```
data.skew()
```

```
Length           -0.639873
```

```
    Diameter         -0.609198
    Height            3.128817
    Whole weight      0.530959
    Shucked weight    0.719098
    Viscera weight    0.591852
    Shell weight      0.620927
Rings           1.114102 dtype:
float64 data.var()

    Length             0.014422
    Diameter           0.009849
    Height             0.001750
    Whole weight       0.240481
    Shucked weight     0.049268
    Viscera weight     0.012015
    Shell weight       0.019377 Rings
    10.395266 dtype: float64
```

```
data.nunique()

    Sex                  3
    Length             134
    Diameter           111
    Height              51
    Whole weight      2429
    Shucked weight    1515
    Viscera weight     880
    Shell weight       926 Rings
    28 dtype: int64
```

4.Check for missing values and deal with them

```
data.isna()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False

2 False False False

```
data.isna().any()
```

3 False False False

```
Sex               False
Length            False
Diameter          False
Height            False
                    ...
Whole weight      False
Shucked weight    False
Viscera weight    False
Shell weight      False
Rings             False
dtype: bool
```

4	False	False	False	False	False	False	False	False
	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
4172	False	False	False	False	False	False	False	False
4173	False	False	False	False	False	False	False	False
	False	False	False	False	False	False	False	False
4174	False	False	False	False	False	False	False	False
	False	False	False	False	False	False		
4175	False	False	False					
4176	False	False	False					

```
data.isna().sum()
```

4177 rows × 9 columns

```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0 Rings
0 dtype: int64
```

```
data.isna().any().sum()
```

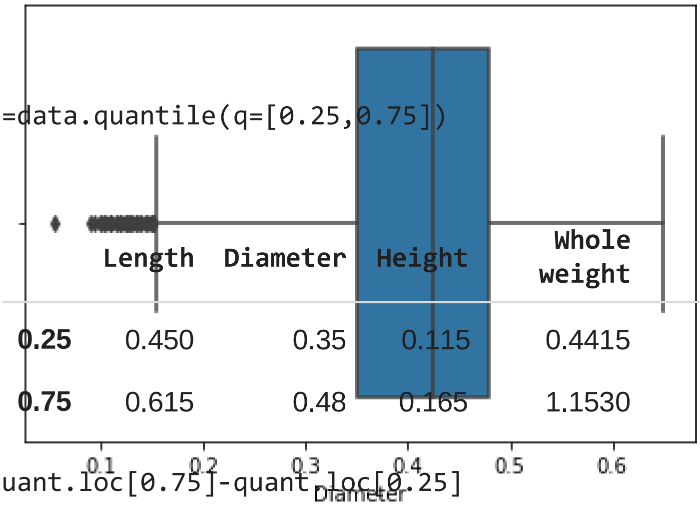
```
0
```

5.Find the outliers and replace them outliers

```
sns.boxplot(data['Diameter'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f3629790>
```

```
quant
```




```
quant=data.quantile(q=[0.25,0.75])
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130	8.0
0.75	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329	11.0

```
iqr=quant.loc[0.75]-quant.loc[0.25]
iqr
```

```
Length            0.1650
Diameter          0.1300
Height            0.0500
Whole weight      0.7115
Shucked weight    0.3160
Viscera weight    0.1595
Shell weight      0.1990 Rings
3.0000 dtype: float64
```

```
low=quant.loc[0.25]-(1.5*iqr)
low
```

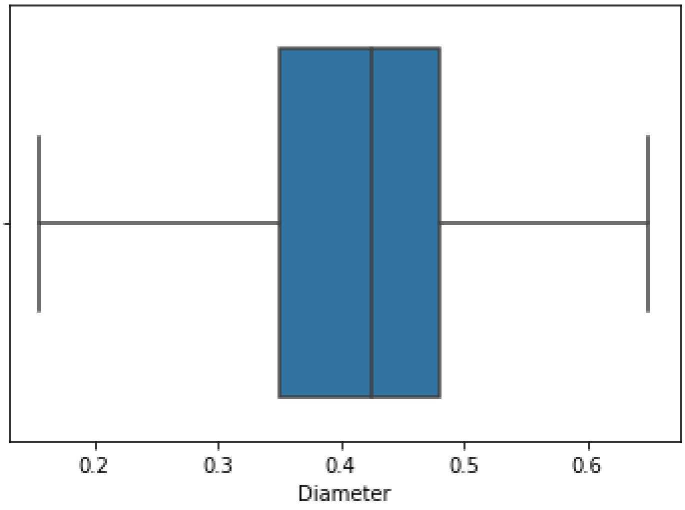
```
Length             0.20250
Diameter           0.15500
Height             0.04000
Whole weight      -0.62575
Shucked weight    -0.28800
Viscera weight    -0.14575
Shell weight      -0.16850 Rings
3.50000 dtype: float64
```

```
up=quant.loc[0.75]+(1.5*iqr)
up
```

```
Length              0.86250
Diameter            0.67500
Height              0.24000
Whole weight        2.22025
Shucked weight      0.97600
Viscera weight      0.49225
Shell weight        0.62750 Rings
15.50000 dtype: float64
```

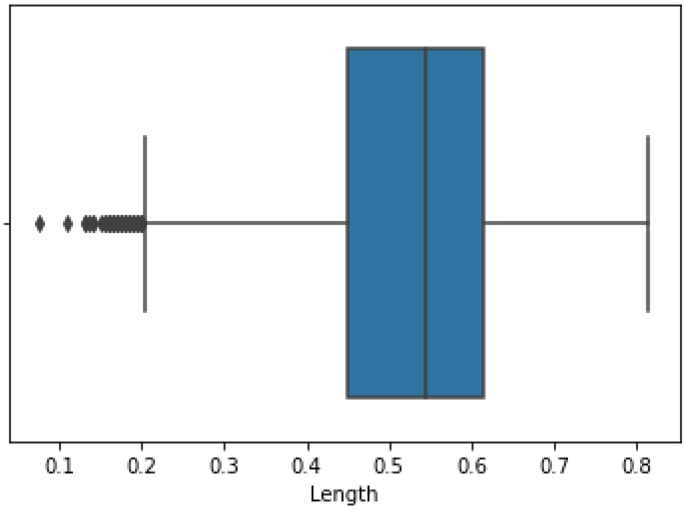
```
data['Diameter']=np.where(data['Diameter']<0.155,0.4078,data['Diameter'])
sns.boxplot(data['Diameter'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f32fb0d0>
```



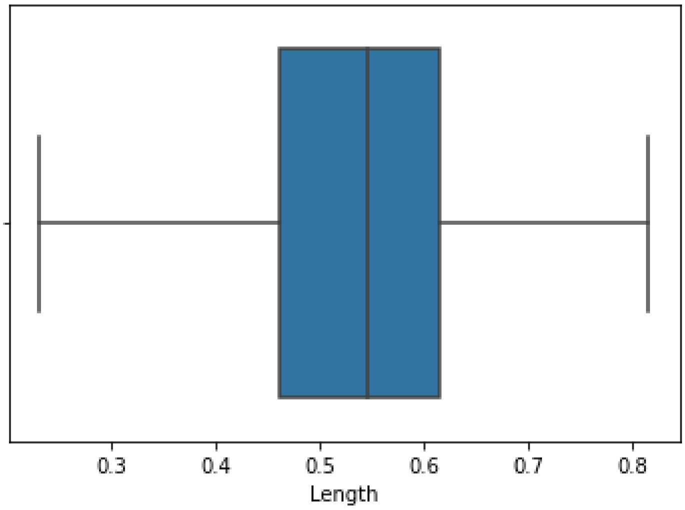

```
sns.boxplot(data['Length'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f32f9b10>

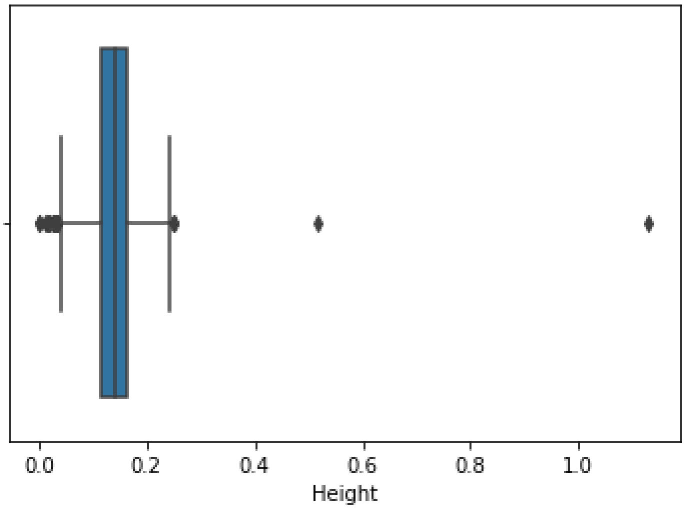



```
data['Length']=np.where(data['Length']<0.23,0.52, data['Length'])
sns.boxplot(data['Length'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f323e710>

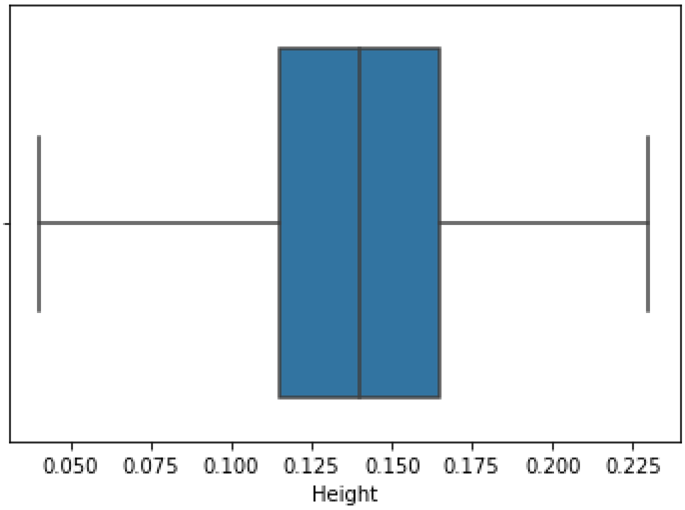



```
sns.boxplot(data['Height']) <matplotlib.axes._subplots.AxesSubplot at 0x7fc6f320f310>
```



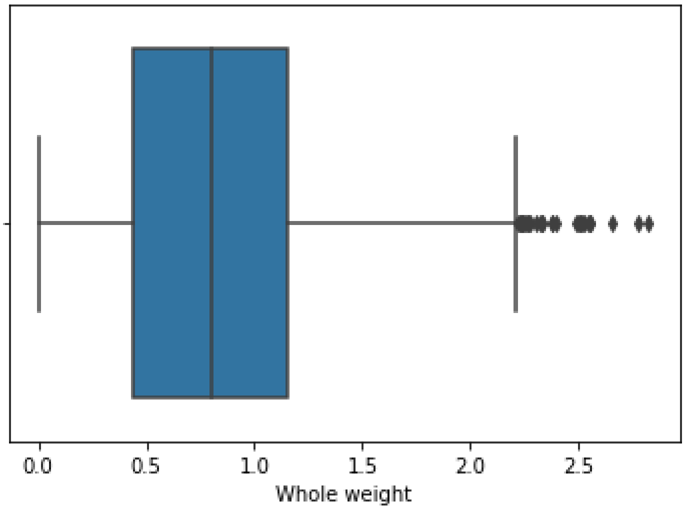

```
data['Height']=np.where(data['Height']<0.04,0.139, data['Height'])
data['Height']=np.where(data['Height']>0.23,0.139, data['Height'])
sns.boxplot(data['Height'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f31a1090>



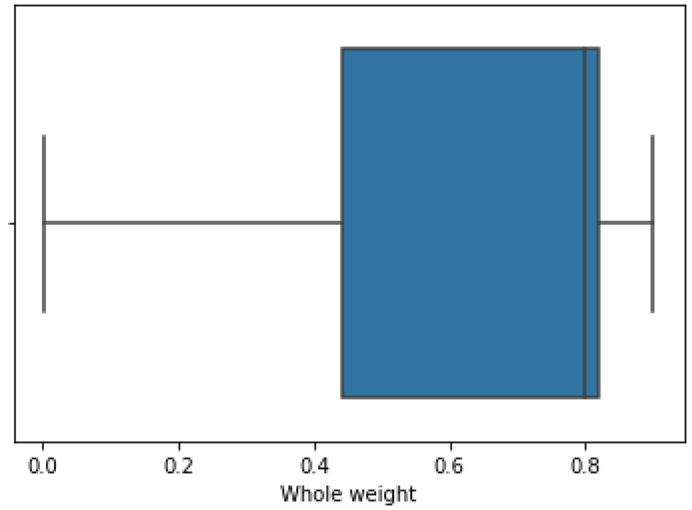

```
sns.boxplot(data['Whole weight'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f31a1050>



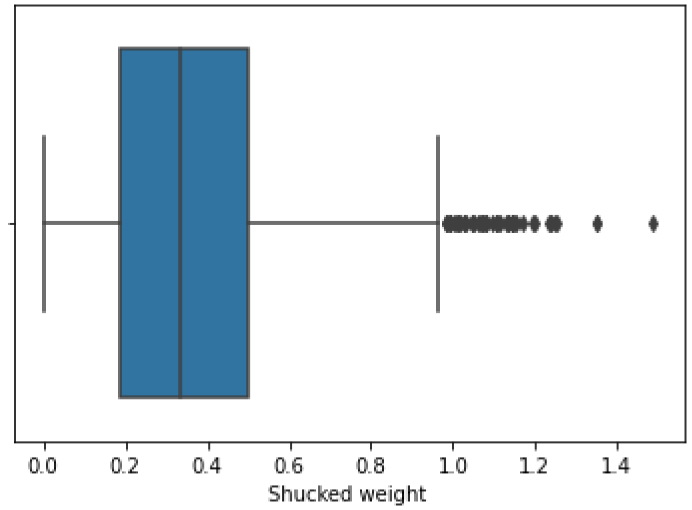

```
data['Whole weight']=np.where(data['Whole weight']>0.9,0.82, data['Whole weight'])
sns.boxplot(data['Whole weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f30ea050>
```




```
sns.boxplot(data['Shucked weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f304f110>
```

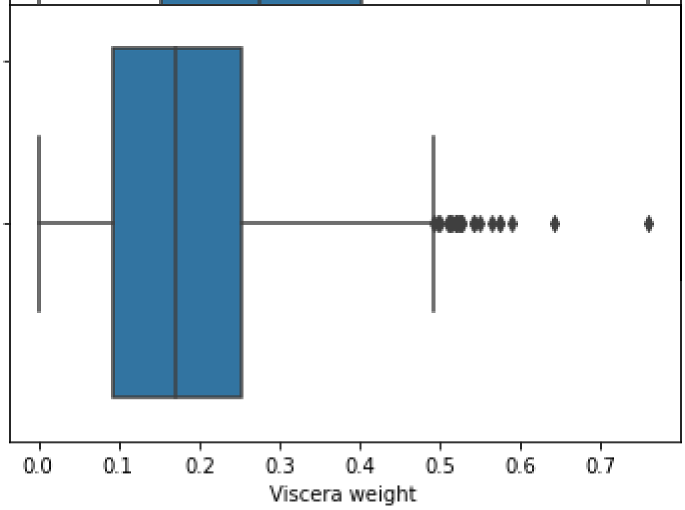



```
data['Shucked weight']=np.where(data['Shucked weight']>0.93,0.35, data['Shucked weight'])
sns.boxplot(data['Shucked weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f30d3e10>
```

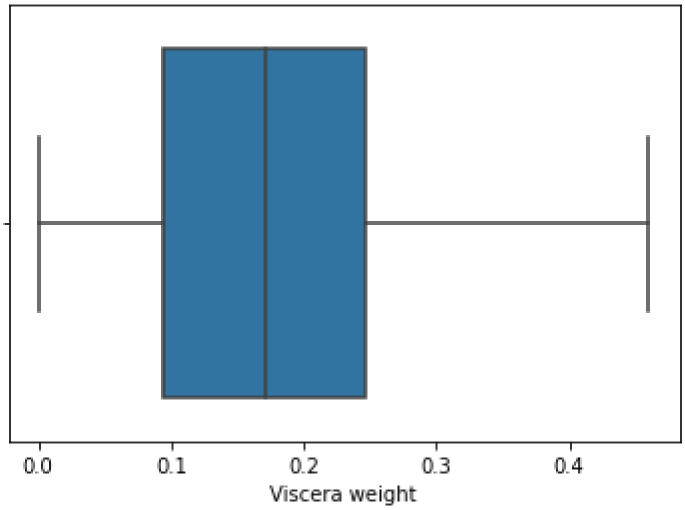
```
sns.boxplot(data['Viscera weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f2f81810>
```




```
data['Viscera weight']=np.where(data['Viscera weight']>0.46,0.18, data['Viscera weight'])
sns.boxplot(data['Viscera weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f2f1a310>
```

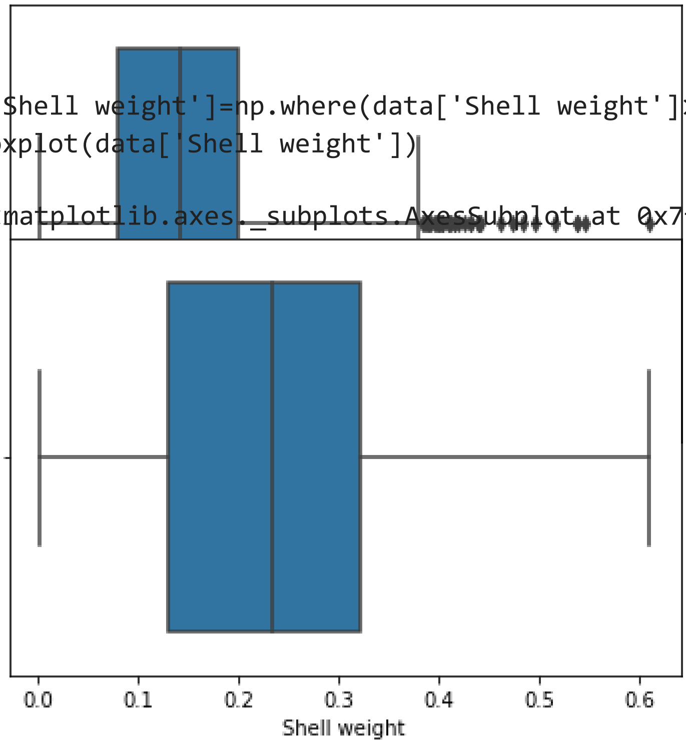



```
sns.boxplot(data['Shell weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f2e7eb50>
```

```
data['Shell weight']=np.where(data['Shell weight']>0.61,0.2388, data['Shell weight'])
sns.boxplot(data['Shell weight'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc6f2e5ae50>



6.Check for Categorical columns and perform encoding.

```
data['Sex'].replace({'M':1,'F':0,'I':2},inplace=True)
```

```
data
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	1	0.590	0.440	0.135	0.8200	0.4390	0.2145	0.2605	10
4174	1	0.600	0.475	0.205	0.8200	0.5255	0.2875	0.3080	9
4175	0	0.625	0.485	0.150	0.8200	0.5310	0.2610	0.2960	10
4176	1	0.710	0.555	0.195	0.8200	0.3500	0.3765	0.4950	12

4177 rows × 9 columns

7.Split the data into dependent and independent variables.

```
x=data.drop(columns= ['Rings'])
y=data['Rings'] x
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550
...	...	...	...	...	...	...	...	...
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490
4173	1	0.590	0.440	0.135	0.8200	0.4390	0.2145	0.2605
4174	1	0.600	0.475	0.205	0.8200	0.5255	0.2875	0.3080
4175	0	0.625	0.485	0.150	0.8200	0.5310	0.2610	0.2960
4176	1	0.710	0.555	0.195	0.8200	0.3500	0.3765	0.4950

4177 rows × 8 columnsy

```
0       15
1       7
2       9
3       10
4       7        ..
4172    11
4173    10
4174    9
4175    10
4176    12
Name: Rings, Length: 4177, dtype: int64
```

8.Scale the independent variables

```
from sklearn.preprocessing import scale
x = scale(x) x
```

```
array([[-0.0105225 , -0.67088921, -0.50179694, ..., -0.61037964,
        -0.7328165 , -0.64358742],
       [-0.0105225 , -1.61376082, -1.57304487, ..., -1.22513334,         -
1.24343929, -1.25742181],
       [-1.26630752,  0.00259051,  0.08738942, ..., -0.45300269,
        -0.33890749, -0.18321163],
       ...,
       [-0.0105225 ,  0.63117159,  0.67657577, ...,  0.86994729,
1.08111018,  0.56873549],
```

```
        [-1.26630752,  0.85566483,  0.78370057, ...,  0.89699645,
 0.82336724,  0.47666033],
        [-0.0105225 ,  1.61894185,  1.53357412, ...,  0.00683308,
          1.94673739,  2.00357336]])
```

9.Split the data into training and testing

```
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)
print(x_train.shape, x_test.shape)
```

```
(3341, 8) (836, 8)
```

10.Build the Model

```
from sklearn.linear_model import LinearRegression
MLR=LinearRegression()
```

11.Train the model

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

```
LinearRegression()
```

12.Test the model

```
y_pred=MLR.predict(x_test) y_pred
```

```
array([10.91653685,  12.55744321,  14.1783292  ,   9.35563082,   7.43211414,
10.7774108   ,  13.38144253,    6.66911369,  11.76080726,    7.53439641,
7.24781575, 13.77037419,  4.91015581, 10.01260524,  8.7898418 ,    6.28146017,
12.2936104  , 11.06755699, 10.4933959  ,  7.17898304,            9.95242944,
6.51892481,  9.82454574,  7.7532194 , 10.08484009,     7.37144014,  3.97011024,
9.32347883,  6.51173485,  9.78511259,    10.48817652,  7.02717025, 12.23620869,
11.92733123, 11.9889763  ,             9.48480236,   9.03522977,   9.37259489,
15.81723261, 10.96336071,            11.21773345,   5.67381233, 12.6814806  ,
13.28242539,   9.59962488,           11.22923672,   7.68329353,   8.9789218  ,
9.12587129,  9.91327472,      9.27332845, 11.40691421,  7.44863498,  9.70226101,
10.25461244,      11.0386109 ,  7.08784785,  8.6607063 ,  9.63829063,  8.34731917,
9.9980484 , 11.37837552,  7.13776877, 13.3970746 ,  6.6005046 ,     13.80133122,
14.00433854, 12.55379762, 12.71515009,   8.76783583,            9.67712105,
11.74556463,   7.0481293  , 13.49328878, 14.86989189,           11.62352645,
5.85855574, 12.90002072,  7.11057015, 10.23755979,      11.63716039,  7.47199396,
7.73538948,  6.60751726, 10.71168715,      12.94158748,  9.31771432, 10.73992843,
12.24071745,   9.25418102,             6.62257397,   7.68153848,  14.49979284,
10.00129705, 14.10599593,
        8.50606455, 12.43504595, 13.34149143,  6.26697755, 12.89253409,
```

```
        8.68848384,    8.54572413,  11.99055703,    7.85121139,    9.56163179,
 13.36290365,   10.78403953,      9.30937974,   11.4594138   ,      4.16215923,
 12.59767471,  6.35526601,  8.4916346 ,  6.19986428,  8.54794644,       7.58145793,
 9.94107683, 10.87611465,  7.44113331, 10.61185092,       8.71409967, 10.1066312
 , 12.86689392,   8.20778755,  10.18096541,          11.56939474,  10.7563318  ,
 11.12633881,  10.96626657,   7.64746453,            13.41757501,  11.51575829,
 11.98897165,   9.18701964,   9.85267727,            10.45124876,   6.64057572,
 12.04431206,  10.11158134,   5.82165217,            12.18607365,   8.60423064,
 12.18718685,  11.16390278,   8.72233716,             8.49650453,   7.86996467,
 8.84657686,  9.66808405,  8.73615923,       9.82792655,  9.6680999 , 10.72665502,
 4.2663484 , 11.37121585,      11.8368494 , 10.91344912,  8.22206639,  9.40991284,
 10.35826288,             10.79331777,  12.12182431,  10.48082557,  11.13014672,
 9.60915214,       10.85040523, 11.94158594,  7.53904291, 16.32969502, 10.0599576
 ,            10.55489637,  12.97231242,   9.82115735,  11.54038743,  10.06498328,
 13.98278119,    11.2312891    ,    12.16565027,    13.25618319,       6.84026022,
 7.98855866,    11.33819098,    13.36212886,    11.37696088,        9.75802861,
 12.49069465,      6.61857583,      9.63995432,   10.89865051,       6.39186652,
 11.38711198,      7.26224245,      9.62199106,      8.9485977   ,   11.94126235,
 9.90815329,         8.29264567,    13.05631009,     11.15914804,    11.48347318,
 11.69346289,      8.3393152   ,   11.07612004,      9.27797441,      8.57070474,
 9.26448714,         9.68189512,    10.79726608,    13.61959519,    11.58284241,
 12.55935344,      9.63744335,      6.75192207,      8.4483055   ,   10.16571898,
 9.74165129,    11.19062134,    10.10241387,    12.51229664,        6.06419923,
 11.43234574,         6.91497992,    10.01392441,    10.12005954,    11.70438295,
 10.22112735,  7.6260467 ,  8.3323689 ,  7.68556049, 14.06249379,       6.52822499,
 12.86749644,   9.4623157  ,  11.96287382,   7.29195347,           10.51326401,
 11.49069921,   6.50913087,  11.00058015,   9.76280581,           10.97974981,
 10.04155794,  11.02494154,  14.1809637  ,  11.40345707,           11.96717269,
 11.06662751,   7.48584774,  10.27747979,   9.75011268,           12.59168959,
 6.71190392,  7.33721287, 13.85825682, 10.19252804,       6.89308223, 10.7374843
 , 11.10384361,   8.56803225,  11.13275726,           8.33529863,  11.49423452,
 15.07212069,   9.39948746,  10.84711907,            11.50568861,  11.57638575,
 17.16819123,  12.31621799,   9.19030825,            10.73585786,  10.44817603,
 8.11579752, 11.76465115, 10.5625014 ,      7.66488272, 10.71086045, 11.67141199,
 8.76119503,  8.58315673,      7.21100131, 10.02118595, 10.17595407,  9.07289132,
 7.07286171,
                 7.85996632,  8.6740366 ,  8.5630248 ,  7.85467058,  8.15100189,
```

```
pred=MLR.predict(x_train) pred
```

```
array([10.95753403, 12.51075719, 12.40779919, ...,  9.07933866,
11.56316712, 13.05582865])
```

```
from sklearn.metrics import r2_score
accuracy=r2_score(y_test,y_pred)
accuracy
```

```
0.41382564717781056
```

```
MLR.predict([[1,0.455,0.365,0.095,0.5140,0.2245,0.1010,0.150]])
```

```
array([9.91859626])
```

13.Measure the performance using Metrics

```
from sklearn import metrics
```

```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
2.5137607628127996
```

LASSO

```
from sklearn.linear_model import Lasso, Ridge
#intialising model lso=Lasso(alpha=0.01,normalize=True)
#fit the model lso.fit(x_train,y_train)
Lasso(alpha=0.01, normalize=True) #prediction
on test data lso_pred=lso.predict(x_test)
#coef
coef=lso.coef_
coef
```

```
array([-0.        ,  0.        ,  0.        ,  0.39518116,  0.14092063,
0.        ,  0.        ,  0.91967498])
```

```
from sklearn import metrics
from sklearn.metrics import mean_squared_error
metrics.r2_score(y_test,lso_pred)
```

```
0.32329491047485237
```

```
np.sqrt(mean_squared_error(y_test,lso_pred))
```

```
2.7009109039758865
```

RIDGE

```
#initialising model rg=Ridge(alpha=0.01,normalize=True)
#fit the model rg.fit(x_train,y_train)
Ridge(alpha=0.01, normalize=True)
#prediction
rg_pred=rg.predict(x_test) rg_pred
```

```
array([10.94284897,  12.55766909,  14.10332643,    9.46590794,    7.46924252,
10.84374888,    13.31875055,       6.65332469,    11.69140963,       7.50414008,
7.24475396, 13.60946869,  5.0169634 , 10.02178785,  8.74951312,       6.37222921,
12.31857953,  11.03870168,  10.49466108,    7.57409317,               9.98428586,
6.49266759,  9.84030265,  7.78278199, 10.19140983,
        7.39626063,  4.1188049 ,  9.31722402,  6.4991906 ,  9.87481467,
```

```
       10.45452133,     7.05292159,   12.19226412,   11.88058887,   11.94176185,
    9.45558448,  9.06859106,  9.4380396 , 15.54387597, 10.87360643,      11.20175072,
    5.63482434, 12.59657969, 13.11927611,  9.55165419,      11.2562201 ,  7.64193974,
    9.03206614,  9.15363186,  9.94026085,       9.25301756, 11.41125447,  7.42301827,
    9.70191307, 10.26573426,      11.08877416,  7.07965658,  8.69369519,  9.63125155,
    8.35676932,      9.93919976, 11.31014253,  7.12871125, 13.27672043,  6.63861343,
    13.67994264,   13.90203172,   12.44586244,   12.76817899,      8.8034988   ,
    9.71751121,    11.82803549,        7.03886776,    13.40189233,    14.74059217,
    11.61507368,      5.83139586,   12.79745889,      7.1515056   ,   10.34467008,
    11.63301292,      7.47023681,      7.71387173,      6.57316872,   10.73587515,
    12.87928907,       9.34315194,   10.78785181,   12.26796854,      9.37745276,
    6.78922748,  7.68734781, 14.34194069,  9.98310639, 13.96447937,      8.53787527,
    12.42359455,  13.34447633,   6.26339643,  12.58179517,             8.69409607,
    8.54383129, 11.94010954,  8.1946398 ,  9.68244419,      13.26318783, 10.82008989,
    9.36552525, 11.4907786 ,  4.30726312,      12.44922397,  6.35701734,  8.68860271,
    6.20646585,  8.59848595,       7.63009379,  9.94773274, 10.98012079,  7.46937269,
    10.62344088,              8.88228329,  10.19672373,  12.86160653,   8.19526719,
    10.23768626,              11.60327133,  10.79218988,  11.02069875,  11.00699291,
    7.60610003,      13.36361692, 11.5001681 , 11.8862438 ,  9.3112327 ,  9.80266528,
    10.4561747   ,      6.61208891,   12.0188652   ,   10.07950545,      5.83817825,
    12.10925296,       8.58950503,   12.21901876,   11.14230194,       8.80164233,
    8.47970329,  7.868846  ,  8.84692525,  9.71954025,  8.80133574,       9.94539604,
    9.66443074,  10.67620234,   4.41311216,  11.38000599,            11.82690585,
    11.01848448,   8.25251755,   9.4282695  ,  10.30600249,            10.83507059,
    12.09765308,  10.50555409,  11.1258249  ,   9.64946864,            10.86249846,
    11.87144177,   7.56481939,  16.09047806,  10.12645866,            10.55382485,
    12.90061432,   9.76527987,  11.50905489,  10.09195624,            13.84575226,
    11.24850011, 12.16689801, 13.17750628,  6.82202086,       7.98381948, 11.3172104
    , 13.20797834,  11.33163535,   9.72350255,          12.43979567,   6.62481266,
    9.59872613, 10.82060265,  6.3839089 ,      11.51849218,  7.27601802,  9.59291787,
    8.93509554, 11.78079094,       9.93000685,  8.369916  , 12.94527836, 11.17246295,
    11.52651747,              11.78751614,   8.32318974,  11.1001638  ,   9.31644338,
    8.59486775,      9.30284616,  9.84731307, 10.90224953, 13.54597748, 11.51809613,
    12.6077395   ,      9.61396008,      6.73559313,      8.43671541,   10.13183962,
    9.71644509, 11.13047849, 10.11506636, 12.5431072 ,  6.02801715,      11.42567449,
    7.29787387,  9.95811538, 10.14655093, 11.68566229,       10.23824892,  7.66218
    ,   8.35979732,   7.72654777,  13.94137525,          6.50124904,  12.78422455,
    9.41120031, 11.99078295,  7.5690498 ,      10.59963443, 11.45096506,  6.55795791,
    11.08043936,   9.90628875,            11.05943396,  10.12296683,  11.06098623,
    14.00848616,  11.37825823,            11.87240228,  11.06310896,   7.47483432,
    10.31308671,   9.7785197  ,            12.48134598,   6.68424648,   7.33411183,
    13.75537421, 10.16269344,       6.91382568, 10.74358733, 11.0538112 ,  8.6157287
    ,  11.13847756,            8.31586114,  11.42146266,  14.91667617,   9.36837014,
    10.81938363,              11.5305361  ,  11.54418845,  16.93766664,  12.18356495,
    9.31738089,      10.7555976 , 10.50502172,  8.09538677, 11.73799171, 10.77361763,
    7.67458578, 10.66594777, 11.67663851,  9.05482079,  8.583721  ,       7.24914069,
    9.98945701, 10.25967048,  9.15198791,  7.04613797,
           7.8642291 ,  8.67148719,  8.61567912,  7.85428151,  8.13317584,
```

```
rg.coef_
```

```
    array([-0.27998868, -0.781349  ,  0.23517567,  0.93481937,  0.97644297,
    -1.41006666, -0.09708902,  1.9317592 ])
```

```
metrics.r2_score(y_test,rg_pred)
```

```
    0.41685467364102824
```

```
np.sqrt(mean_squared_error(y_test,rg_pred))
```

```
2.5072574844843873
```


0s completed at 12:00 PM