ASSIGNMENT - 4

NAME : RAHUL HARIESH B
REG.NO : 917719IT074

1.Loading Dataset into tool

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

Choose Files abalone.csv

- **abalone.csv**(text/csv) - 191962 bytes, last modified: 10/27/2022 - 100% done

```
Saving abalone.csv to abalone (2).csv
```

2.Performing Visualization

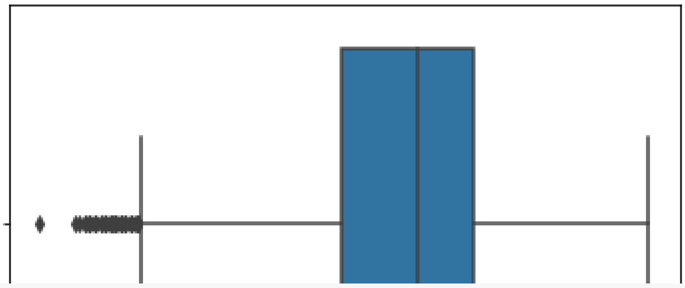
Univariate Analysis

```
data.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shel
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	

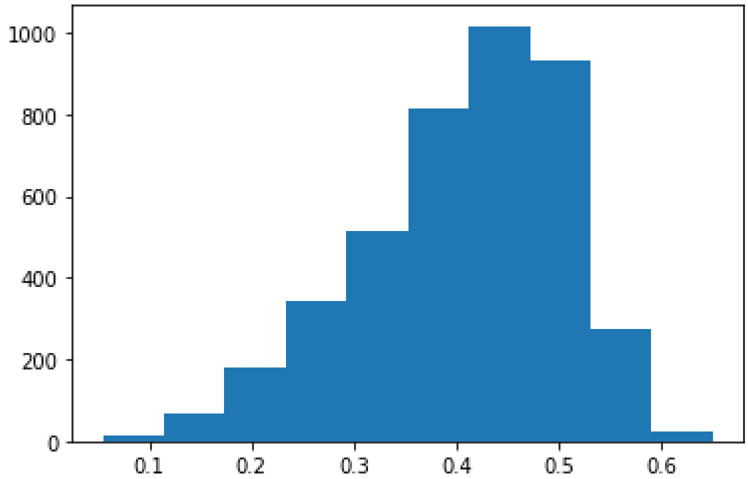
```
sns.boxplot(data['Diameter'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed42e45d0>
```



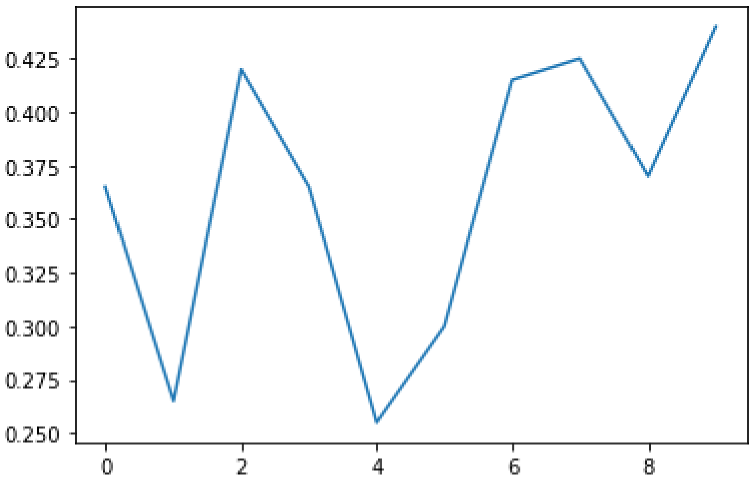
```
plt.hist(data['Diameter'])
```

```
(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>)
```



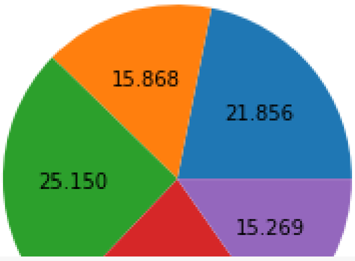

```
plt.plot(data['Diameter'].head(10))
```

```
[<matplotlib.lines.Line2D at 0x7f8ed3c83110>]
```



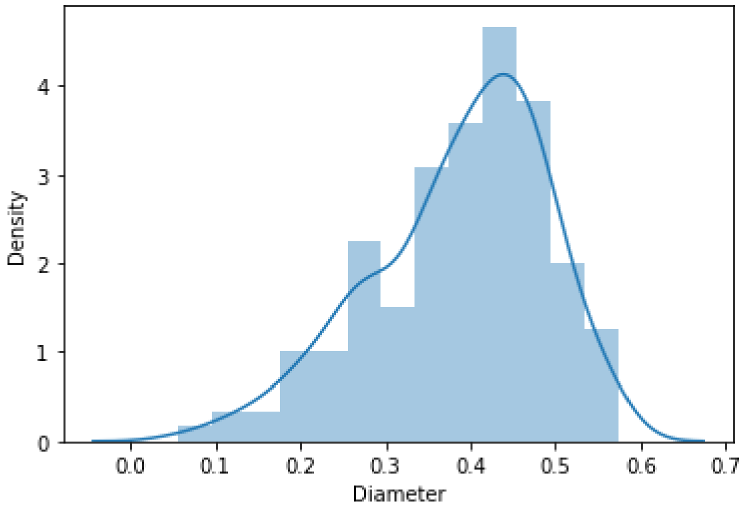

```
plt.pie(data['Diameter'].head(),autopct='%.3f')
```

```
([<matplotlib.patches.Wedge at 0x7f8ed3bec810>,
  <matplotlib.patches.Wedge at 0x7f8ed3becfd0>,
  <matplotlib.patches.Wedge at 0x7f8ed3bf7750>,
  <matplotlib.patches.Wedge at 0x7f8ed3c04050>,
  <matplotlib.patches.Wedge at 0x7f8ed3c04a50>],
 [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')])
```



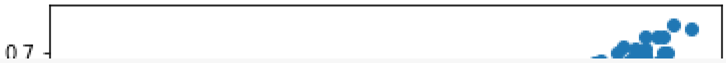

```
sns.distplot(data['Diameter'].head(300))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed3c52410>
```



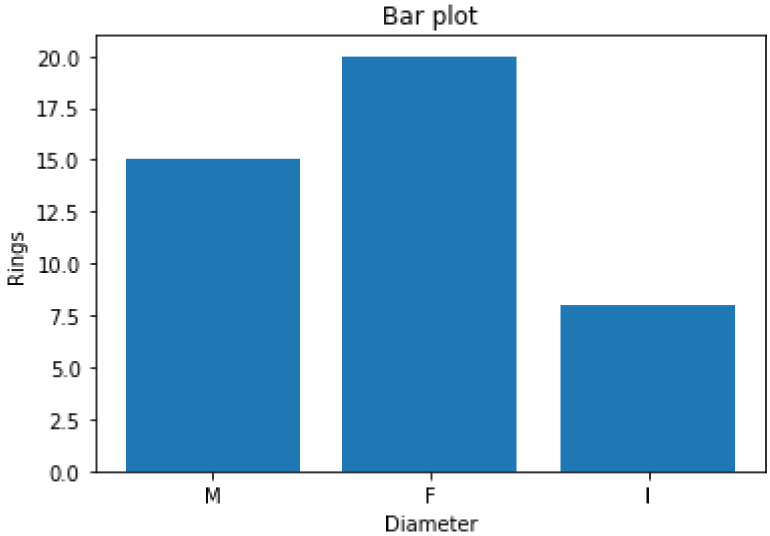

```
plt.scatter(data['Diameter'].head(400),data['Length'].head(400))
```

```
<matplotlib.collections.PathCollection at 0x7f8ed3b52d50>
```



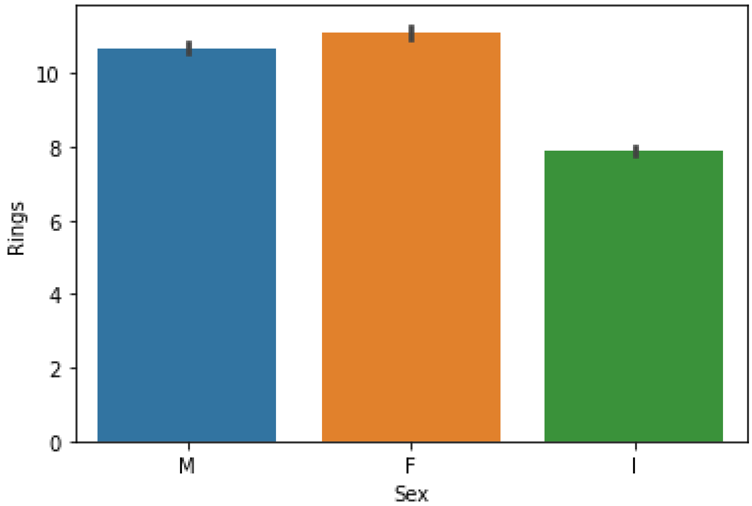

```
plt.bar(data['Sex'].head(20),data['Rings'].head(20))
plt.title('Bar plot')
plt.xlabel('Diameter')
plt.ylabel('Rings')
```

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



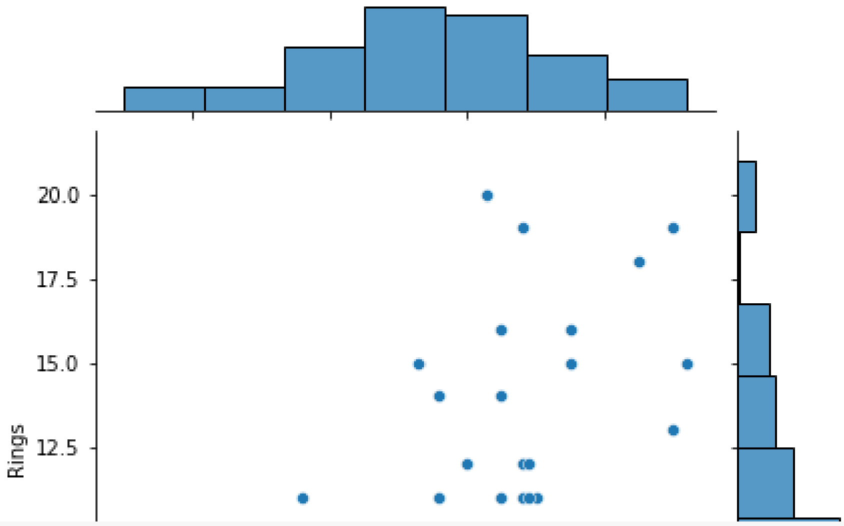

```
sns.barplot(data['Sex'], data['Rings'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed3a3ca50>
```



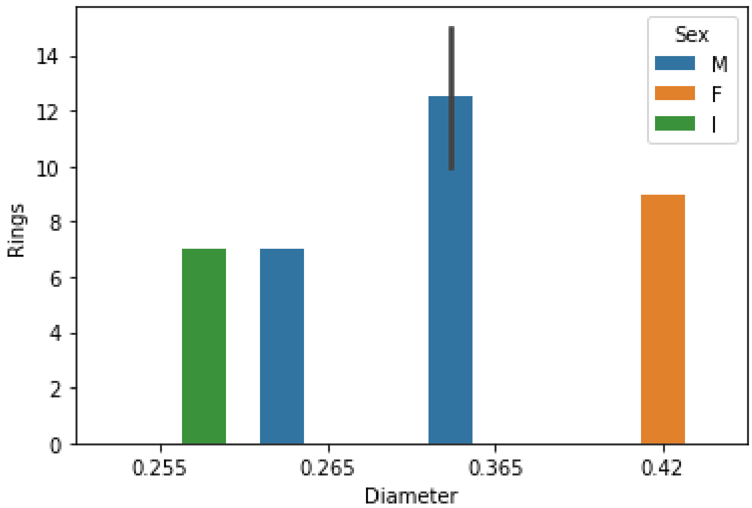

```
sns.jointplot(data['Diameter'].head(50),data['Rings'].head(100))
```

<seaborn.axisgrid.JointGrid at 0x7f8ed3a17450>



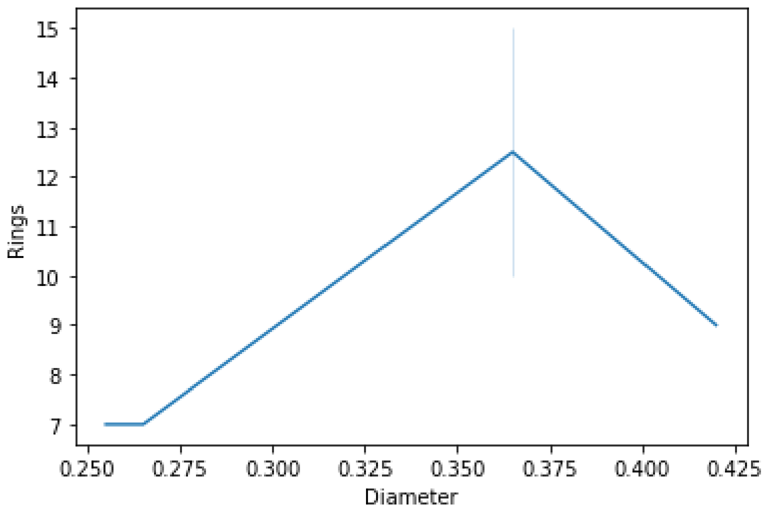

```
sns.barplot('Diameter','Rings',hue='Sex',data=data.head())
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed38fa290>



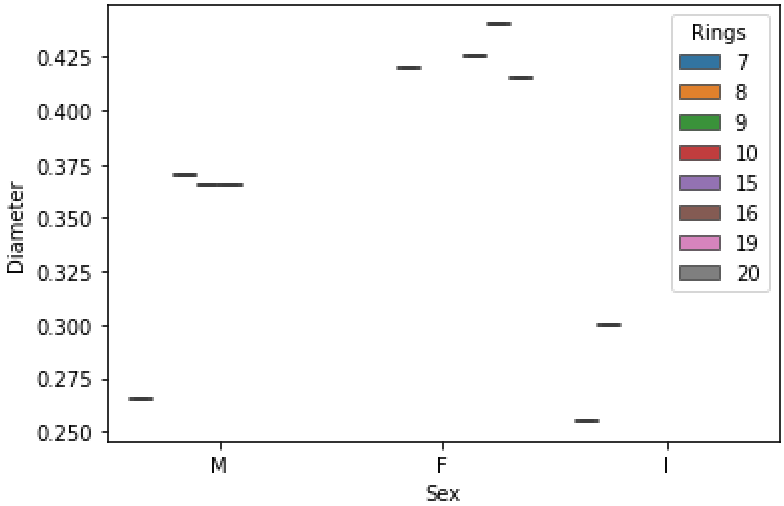

```
sns.lineplot(data['Diameter'].head(),data['Rings'].head())
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed3a17290>



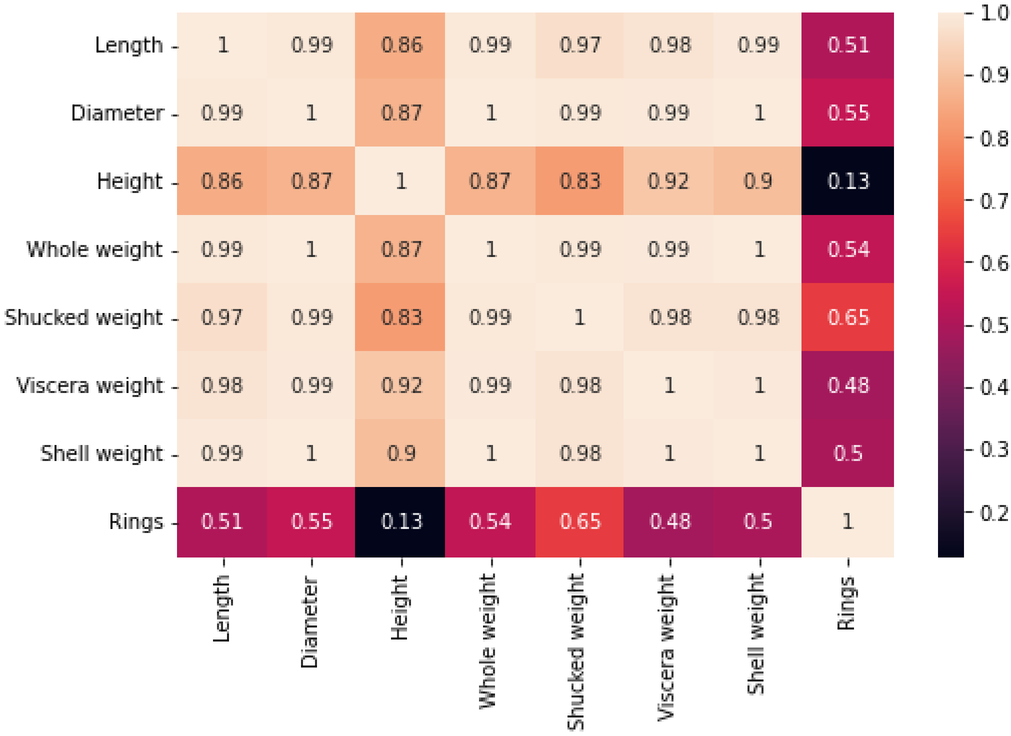

```
sns.boxplot(data['Sex'].head(10),data['Diameter'].head(10),data['Rings'].head(10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed3794a90>
```



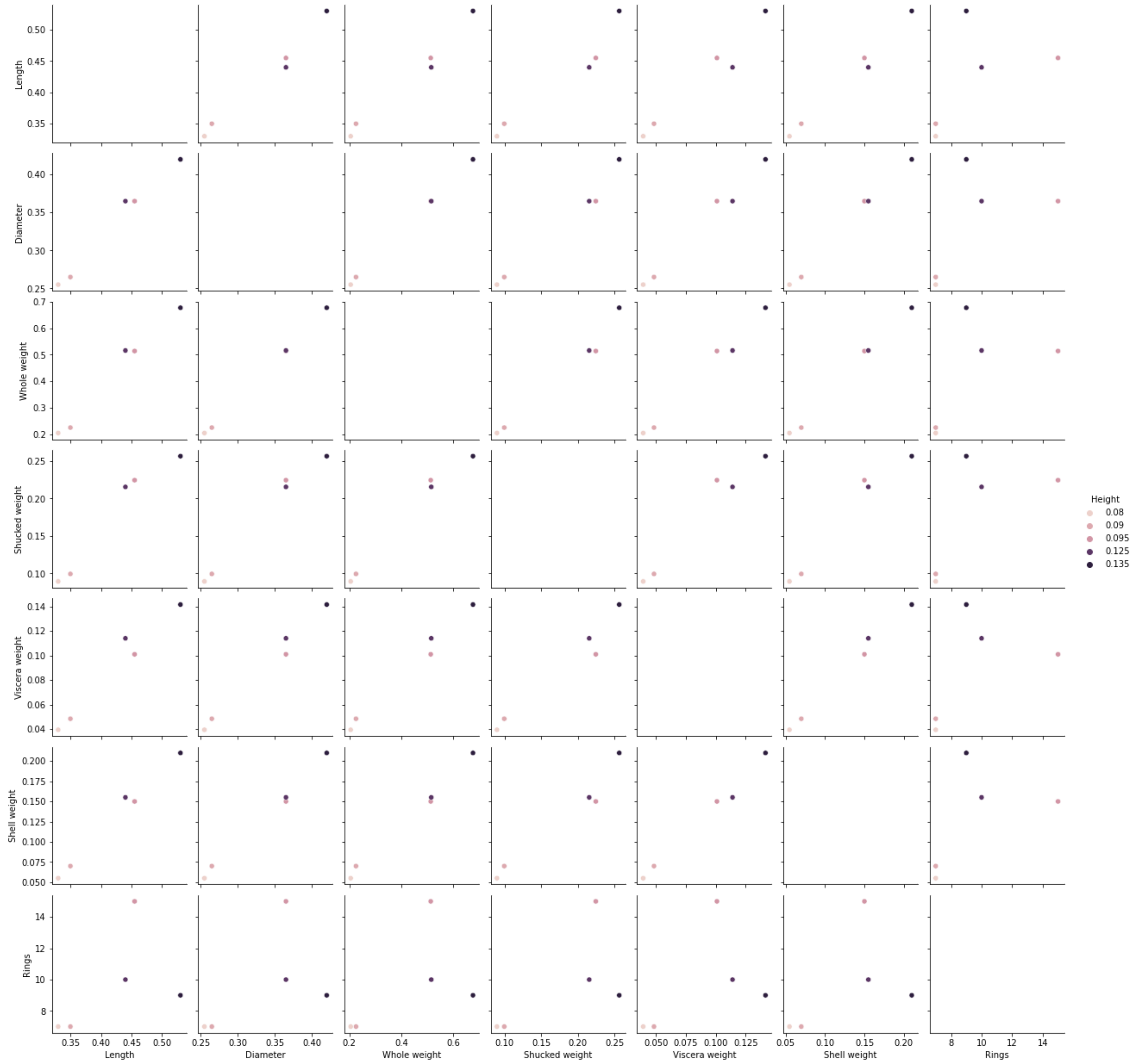

```
fig=plt.figure(figsize=(8,5))
sns.heatmap(data.head().corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed363ed50>
```



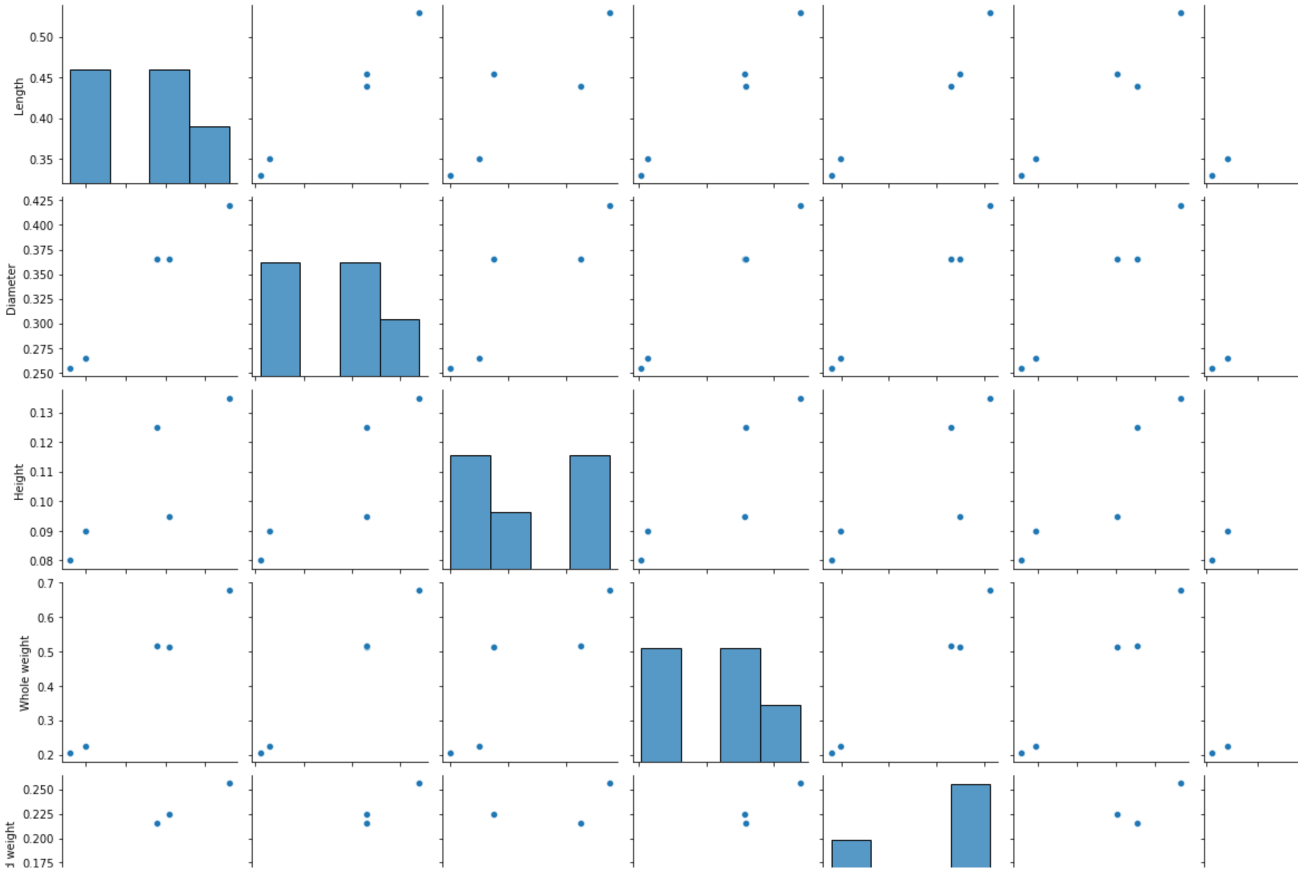

```
sns.pairplot(data.head(),hue='Height')
```

```
<seaborn.axisgrid.PairGrid at 0x7f8ed35890d0>
```



```
sns.pairplot(data.head())
```

```
<seaborn.axisgrid.PairGrid at 0x7f8ed266fed0>
```



3.Perform Descriptive Statistics on the dataset

```
data.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shel
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	

```
data.tail()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765

```python
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
```

```python
data.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	41
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	

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

```
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
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
4172	False	False	False	False	False	False	False
4173	False	False	False	False	False	False	False
4174	False	False	False	False	False	False	False
4175	False	False	False	False	False	False	False
4176	False	False	False	False	False	False	False

4177 rows × 9 columns

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

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

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

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

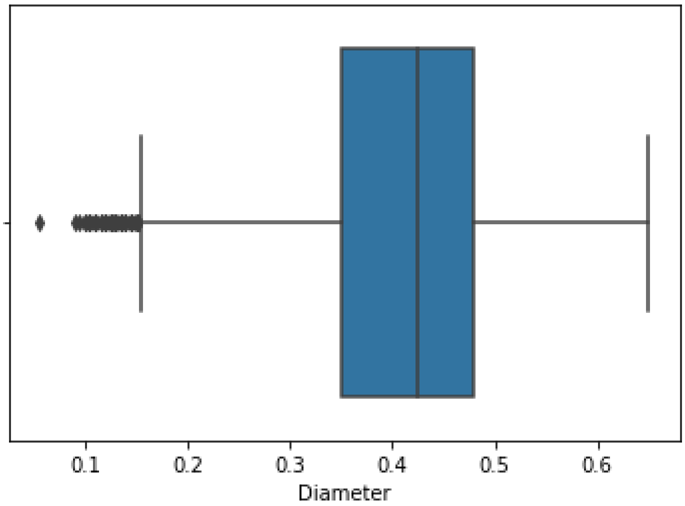
```
data.isna().any().sum()
```

```
0
```

5.Find the outliers and replace them outliers

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

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x7f8ed2fe8950&gt;
```




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

	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
```

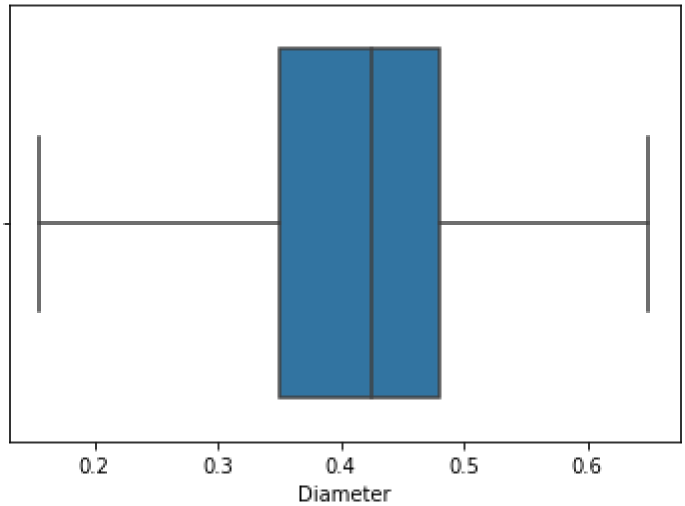
```
Whole weight     -0.62575
Shucked weight   -0.28800
Viscera weight   -0.14575
Shell weight     -0.16850
Rings             3.50000
dtype: float64
```

```
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 0x7f8ed05a5a90>
```

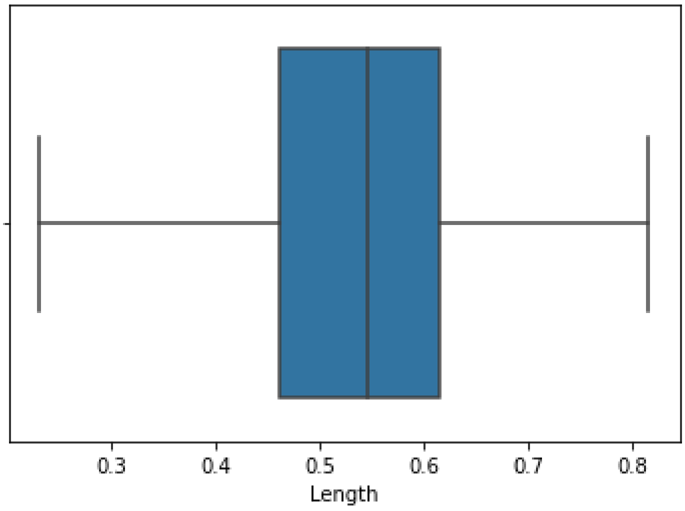



```
sns.boxplot(data['Length'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed0504d50>

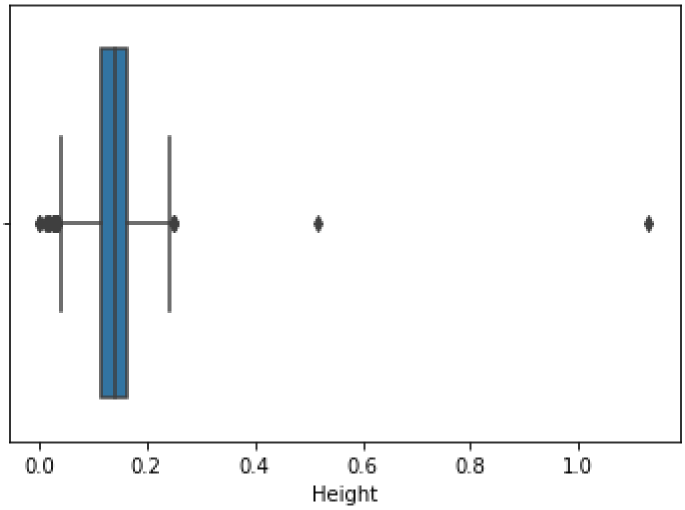
```
data['Length']=np.where(data['Length']<0.23,0.52, data['Length'])
sns.boxplot(data['Length'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed04c5510>




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

<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed03ec050>

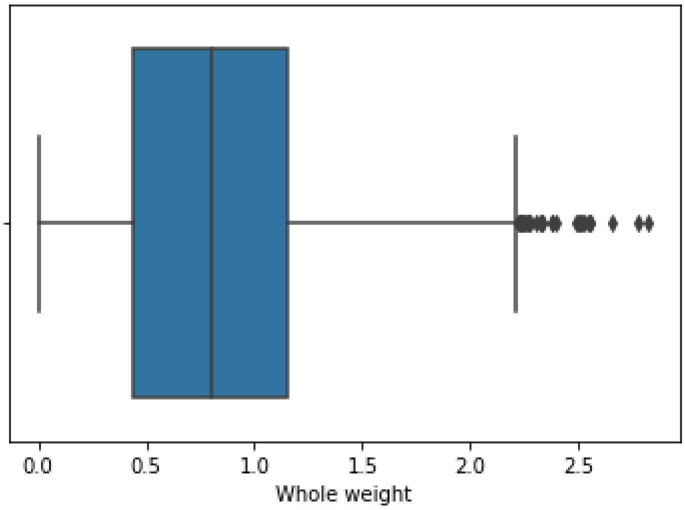



```
data['Height']=np.where(data['Height']<0.04,0.139, data['Height'])
data['Height']=np.where(data['Height']>0.23,0.139, data['Height'])
sns.boxplot(data['Height'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed03bdfd0>
```

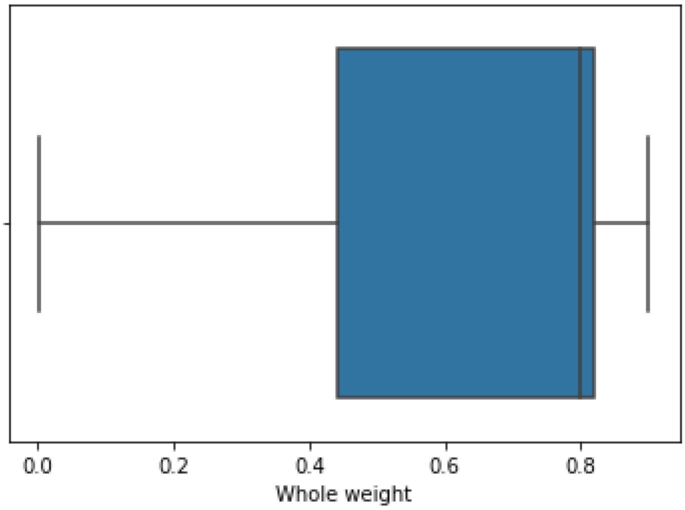
```
sns.boxplot(data['Whole weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed0334750>
```



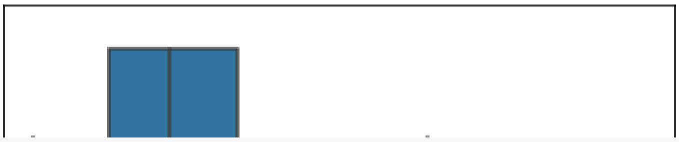

```
data['Whole weight']=np.where(data['Whole weight']>0.9,0.82, data['Whole weight'])
sns.boxplot(data['Whole weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed02a3610>
```



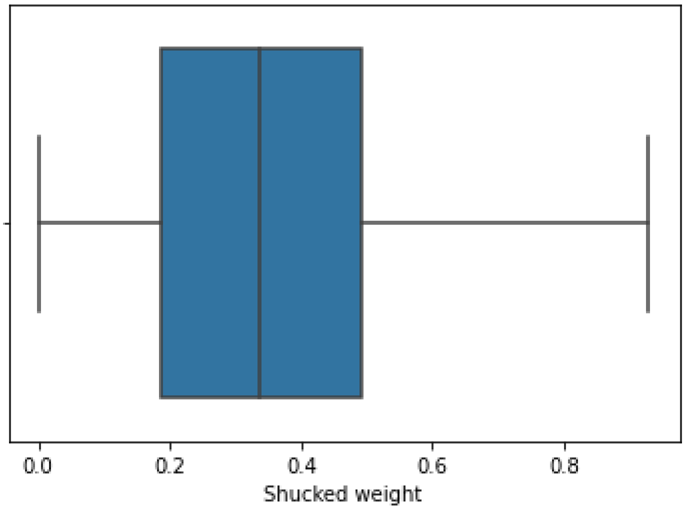

```
sns.boxplot(data['Shucked weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed0292e50>
```



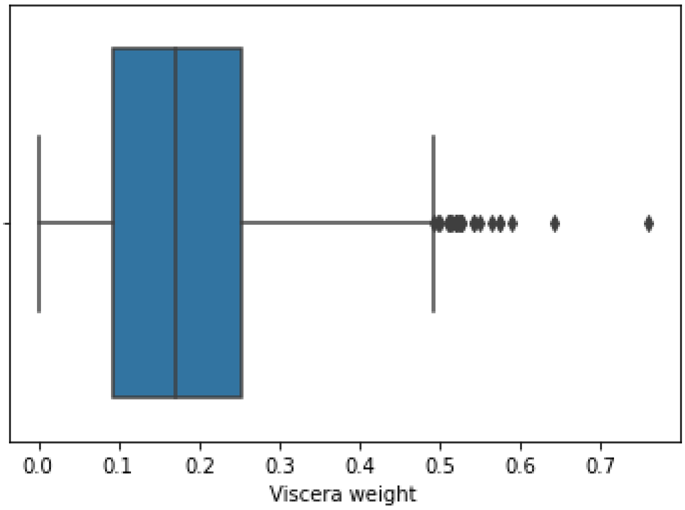
```
data['Shucked weight']=np.where(data['Shucked weight']>0.93,0.35, data['Shucked weight'])
sns.boxplot(data['Shucked weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed01fadd0>
```



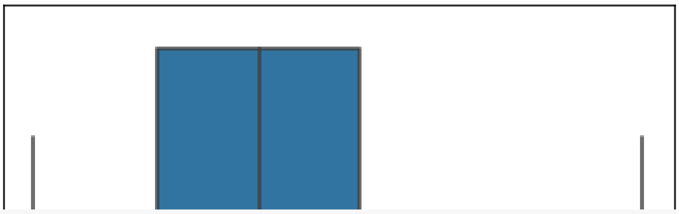

```
sns.boxplot(data['Viscera weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed0176610>
```



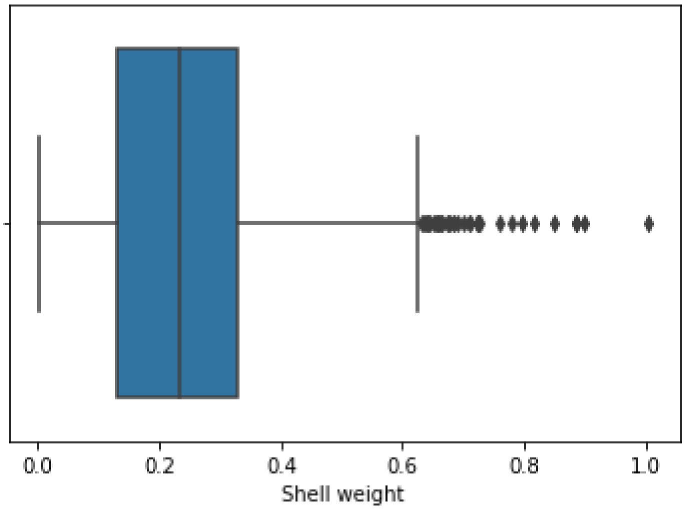

```
data['Viscera weight']=np.where(data['Viscera weight']>0.46,0.18, data['Viscera weight'])
sns.boxplot(data['Viscera weight'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed00e17d0>



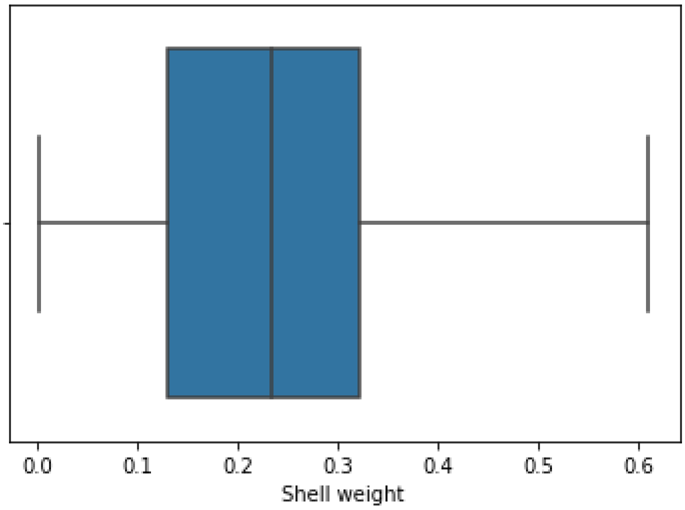
```
sns.boxplot(data['Shell weight'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8ed00c6b10>




```
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 0x7f8ed0083450>



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

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 columns

```
y
```

```
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
```

```
        8.28302381,  6.80703473, 10.14622512,  8.80659289,  8.67300268,
       11.64525322, 10.62947375, 12.41131129, 10.67906878, 10.62931222,
        8.08207895,  7.23201843,  7.23454285,  9.1121437 ,  6.91087923,
        8.3031377 , 10.2415864 , 10.97090534,  9.40846073,  9.71199549,
       10.96843135, 10.95582949,  7.78158587,  6.52329076, 11.03709441,
       11.07639127, 12.80788972, 13.55538906, 11.89986487,  9.94390459,
       10.66224031, 11.71106846, 10.73611615,  9.91789824,  8.75063832,
        9.33180575, 10.8514073 , 12.73438985, 11.04863175,  8.54523508,
       12.90767261, 14.00488364, 10.46639039,  8.22111291,  7.0316113 ,
        8.83730693,  9.73247054, 10.53720865,  6.75273473,  9.83718438,
       12.59967536,  8.56989844, 14.61831869, 12.46070633, 13.00629932,
       12.24306204, 13.73334628, 12.30583728,  6.85250431, 11.41155001,
       10.81489055, 12.56395171, 10.86778438,  7.1516642 ,  7.20151922,
       11.88434155,  9.41248327, 10.08775632, 10.15608315,  7.27407599,
       10.80447349, 11.20569057, 11.9122926 , 11.8119424 , 12.44464725,
        9.43520375,  8.84020036, 12.94674703,  8.30873796,  8.51163199,
        7.70669946, 11.99864847,  6.85874138, 14.50281209, 11.21059094,
        6.69606389,  9.0897566 ,  9.06627113, 13.08740844,  7.39980906,
        9.74394267, 11.95384079,  9.10068791, 11.07523904,  8.01632588,
        8.62320466,  7.96385688,  6.84722731,  9.54046652, 10.17306675,
        7.06997655,  9.94293828, 12.23792609, 11.09187927,  7.06075364,
       10.10025445,  8.9812512 ,  6.81804015, 12.27540751, 12.88938924,
       11.45457568,  6.07447188, 11.26190814,  5.77662195, 11.57028684,
       15.53422036,  6.81430978, 12.50357518, 12.01251715, 13.73182302,
        8.45142136,  6.62643104, 10.65395715,  8.37096633, 13.09711736,
       10.27634662,  8.63032238, 10.275617  , 13.28123427, 11.64728785,
        9.58645516, 11.12120888, 16.11259521, 12.66144875,  7.12379028,
        6.50959605, 11.44751098, 12.77045143,  7.71357878,  9.00045675,
       11.49878276,  8.42755739, 12.25501677,  9.73744105, 10.82595179,
       11.12555266, 10.4434275 , 10.03897825, 10.50706749,  9.32807249,
        9.79246434,  9.59763315,  9.84309603, 11.54734512,  9.65218684,
        9.28898019, 12.74238923,  9.0278447 , 12.23370855, 11.69034311,
        9.42455531, 11.14718653,  8.92937894, 13.09767783, 11.17630726,
       10.88155909, 10.17553512, 10.77809751, 10.78832233,  6.59680499,
        6.69324962, 10.88352585, 10.0784974 , 11.16347347, 11.50610012,
        9.27596305,  8.16432929, 12.97248157, 10.86952699,  8.82508666,
       10.11311737, 11.2027162 , 13.23084954,  6.90444362,  6.60466576,
        6.48704208,  8.5283003 , 11.5652372 , 12.85740913,  9.02692544,
       11.40393323,  8.39333887,  5.79220227, 13.56199222, 10.1412087 ,
        7.88681927,  7.26749514,  8.91161678, 12.95059963, 12.43183382,
        9.59418144,  7.09186839, 10.24451646, 10.88836121,  7.31080274,
       14.77987462,  8.89238508, 11.26476367, 13.33002841, 10.81570312,
       11.96354332,  9.7114161 , 10.66479017, 14.0120255 ,  9.30046649,
        7.24969947,  9.44885318, 12.0500118 , 10.19263865, 11.80173049,
       10.64511947, 10.35462376, 11.04073452, 11.54270178, 12.55582421,
       13.89663466,  9.74743711, 10.61808063, 11.12256538,  9.73461396,
        7.20578585,  9.42259302,  6.55805808,  6.94619707, 13.59208686,
        8.02987978, 12.08106226,  6.98621944, 10.07291212, 15.00009599,
        8.02880097,  9.88710237,  6.32041777, 10.90479422,  8.10485288,
       10.2082919 ,  8.36562117,  8.92196388, 10.31006884, 10.0984584 ,
       10.25163804, 11.82392451,  6.3942556 ,  9.29334161,  7.75402583,
       11.89060658,  9.80152061, 12.44009651, 11.69353927,  8.53029606,
        8.23821649,  6.55343407,  6.95371647,  9.09760269,  6.87117008,
        8.82139125,  9.00133505, 10.66129416, 10.39555104,  9.73837274,
        6.43469811, 11.87126895,  9.50241489, 10.08611136,  7.39922025,
       12.46075335, 10.73802676,  7.17022511, 12.02261681,  6.5785987 ,
       11.18762102,  7.86268622, 10.74849877,  7.2900424 , 11.27089541,
```

```
       8.88007503])
```

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

```
array([10.2918066 ,  9.43514005, 11.04680197, ...,  9.38397647,
        8.11755911, 11.99777509])
```

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

```
0.4495174776679036
```

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

```
array([9.9428431])
```

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))
```

```
2.3848161286712557
```

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.        ,  0.44104319,  0.22522348,
        0.        ,  0.        ,  0.80188509])
```

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

```
0.33219490497803317
```

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

```
2.6266850174643404
```

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
```

```
        8.26214566,  6.83598346, 10.2634771 ,  8.77914039,  8.93440102,
       11.71399493, 10.54523229, 12.3562332 , 10.6812123 , 10.68364519,
        8.12889285,  7.21812511,  7.25172368,  9.20839698,  6.90773494,
        8.36324462, 10.27320209, 10.99105821,  9.42750383,  9.68164727,
       11.01953264, 10.8526412 ,  7.86300153,  6.50011149, 10.978223  ,
       11.09206189, 12.83429806, 13.50421029, 11.86241699, 10.03903156,
       10.67373152, 11.70240498, 10.7070267 , 10.07339448,  8.6659436 ,
        9.39676967, 10.7845043 , 12.56112643, 11.06899003,  8.57146322,
       12.86806103, 13.88991195, 10.44039359,  8.21821906,  7.06724205,
        8.778853  ,  9.73697157, 10.54280011,  6.74013139,  9.79066371,
       12.45416218,  8.59537244, 14.39437772, 12.41423314, 12.89349322,
       12.14359631, 13.62190296, 12.20493316,  6.86053073, 11.41242636,
       10.72733528, 12.63097701, 10.84139347,  7.15843918,  7.20969665,
       11.93711708,  9.45154328, 10.08748729, 10.12178256,  7.23388368,
       10.82722284, 11.13866215, 11.90210343, 11.80780188, 12.35957016,
        9.48728927,  8.85892074, 12.78309893,  8.25174642,  8.46290663,
        7.74083379, 11.87792609,  6.85990403, 14.32361984, 11.27131705,
        6.70201327,  9.13155032,  9.0705427 , 13.02529653,  7.42917473,
        9.73812696, 11.92277596,  9.13952998, 11.01680934,  8.04941216,
        8.686199  ,  7.97455119,  6.85579231,  9.64463724, 10.3178143 ,
        7.07905178,  9.99154376, 12.21823088, 11.14836533,  7.01935213,
       10.17879581,  9.02509479,  6.82545318, 12.26044194, 12.77178103,
       11.34223882,  6.07574009, 11.20149921,  5.75010483, 11.59046927,
       15.30121937,  6.80557939, 12.41514111, 12.02921782, 13.54620663,
        8.48117642,  6.6245008 , 10.65723842,  8.65888824, 13.08354902,
       10.30955264,  8.67238548, 10.25562541, 13.10862194, 11.64634739,
        9.55470753, 11.06247069, 15.91097617, 12.59172479,  7.14679493,
        6.50902842, 11.46063508, 12.76366928,  7.72575376,  8.95004232,
       11.55005252,  8.42974303, 12.06936418,  9.83167693, 10.85408567,
       11.16839543, 10.48526578, 10.04259555, 10.63755821,  9.32881265,
        9.81297957,  9.56979318,  9.86931868, 11.4409946 ,  9.78828174,
        9.29656253, 12.57051765,  9.04521346, 12.13413114, 11.76294544,
        9.43466522, 11.10088632,  8.88809268, 13.020723  , 11.16829919,
       10.80502847, 10.23566895, 10.81596222, 10.78270741,  6.63281769,
        6.69354154, 10.81669543, 10.21245775, 11.09048021, 11.45461581,
        9.3319393 ,  8.19531483, 12.95867252, 10.85221434,  8.90323983,
       10.11311796, 11.21617055, 13.14768566,  6.92271688,  6.60113124,
        6.46773661,  8.5228144 , 11.56089094, 12.75433962,  9.04540261,
       11.30720841,  8.39964602,  5.77401258, 13.44973258, 10.17575446,
        7.96768197,  7.2778146 ,  8.91138896, 12.81818389, 12.41978901,
        9.62939487,  7.08054904, 10.2957178 , 10.92359257,  7.33376326,
       14.61924411,  9.00188642, 11.27531447, 13.25818553, 10.7606288 ,
```

```
11.9736864 ,  9.86595551, 10.58742932, 13.85166074,  9.25429151,
 7.23917133,  9.39434424, 12.02572072, 10.29376802, 11.73238963,
10.78636094, 10.3028947 , 11.11621434, 11.46324185, 12.47715316,
13.74459879,  9.76510768, 10.65407953, 11.15394561,  9.68301897,
 7.22215224,  9.42569246,  6.57780434,  6.97671904, 13.44777834,
 8.07544782, 11.97829587,  6.96273172, 10.10431947, 14.7535265 ,
 8.08209103,  9.8518559 ,  6.3231934 , 11.00307396,  8.26943073,
10.33953552,  8.41457457,  8.89848421, 10.38109938, 10.30666658,
10.23140237, 11.82981844,  6.3883851 ,  9.28958284,  7.80379187,
11.73186337,  9.80775659, 12.42763329, 11.63488101,  8.55712126,
 8.26169337,  6.66885565,  6.98995444,  9.10533014,  6.83369853,
 8.81714466,  9.11481822, 10.68733908, 10.31653973,  9.81076448,
 6.46221449, 11.81520831,  9.53798307, 10.14077356,  7.4148972 ,
12.36306211, 10.72427387,  7.19510671, 11.98506299,  6.56559339,
11.27443449,  7.90831624, 10.79332911,  7.29335713, 11.31161072,
 8.85998991])
```

```
rg.coef_
```

```
array([-0.29908686, -0.71684578,  0.35136926,  0.93249058,  0.96444086,
       -1.38925399, -0.04131207,  1.7059293 ])
```

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

```
0.44751749074992275
```

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

```
2.3891444011828984
```