

ASSIGNMENT 4

NAME : JEYA GANESH A V

REG.NO : 917719IT041

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.csv

2.Performing Visualization

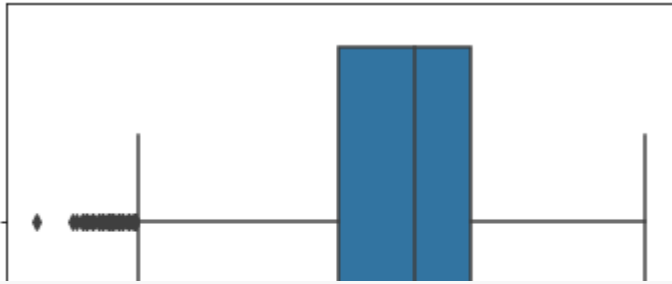
Univariate Analysis

```
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

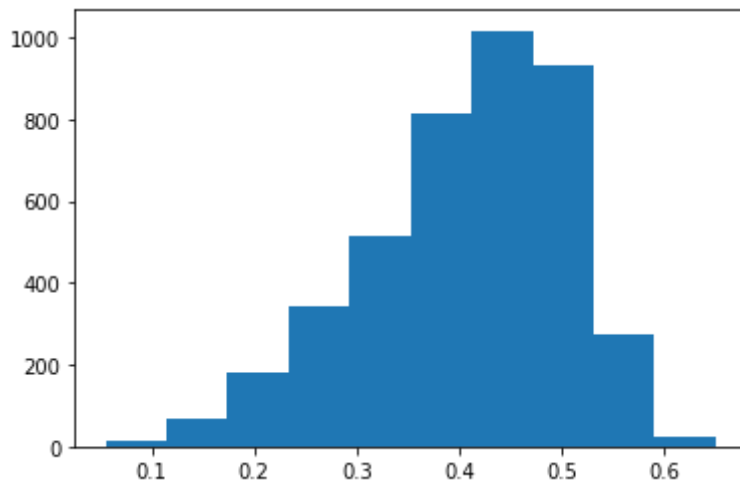
```
sns.boxplot(data['Diameter'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa784d90410>
```



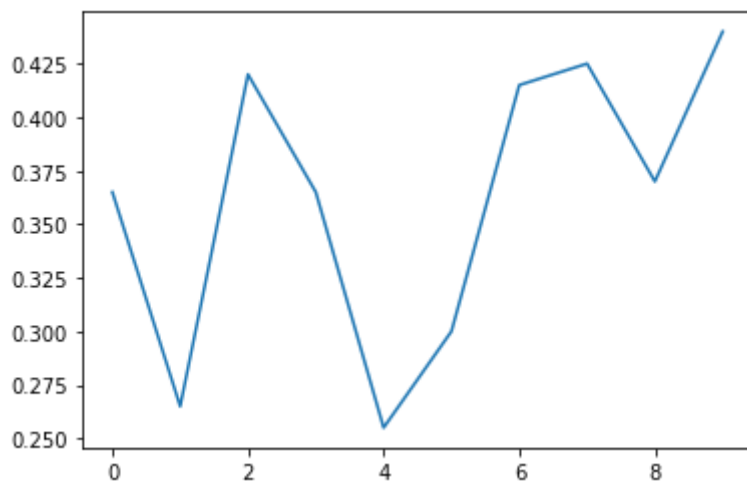
```
plt.hist(data['Diameter'])
```

```
(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>)
```



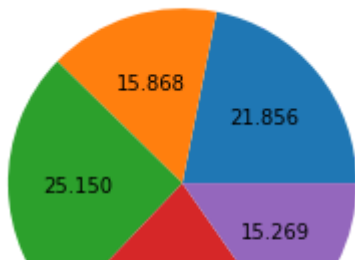
```
plt.plot(data['Diameter'].head(10))
```

```
[<matplotlib.lines.Line2D at 0x7fa78477b050>]
```



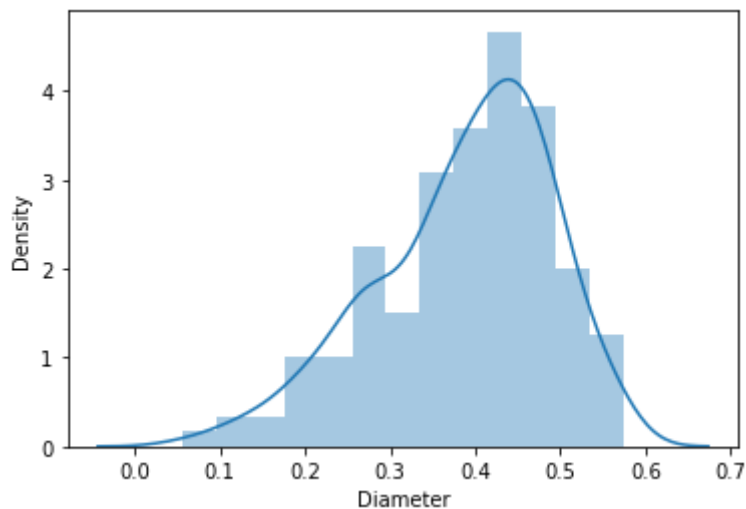
```
plt.pie(data['Diameter'].head(), autopct='%.3f')
```

```
([<matplotlib.patches.Wedge at 0x7fa7846f8290>,
 <matplotlib.patches.Wedge at 0x7fa7846f8950>,
 <matplotlib.patches.Wedge at 0x7fa784702150>,
 <matplotlib.patches.Wedge at 0x7fa784702b50>,
 <matplotlib.patches.Wedge at 0x7fa78470d6d0>],
 [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 0x7fa78470dcd0>
```



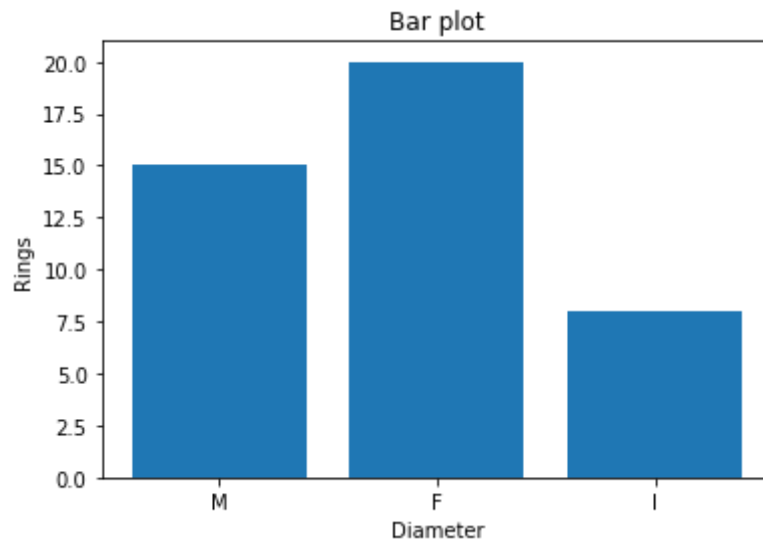
```
plt.scatter(data['Diameter'].head(400), data['Length'].head(400))
```

```
<matplotlib.collections.PathCollection at 0x7fa78460cf50>
```



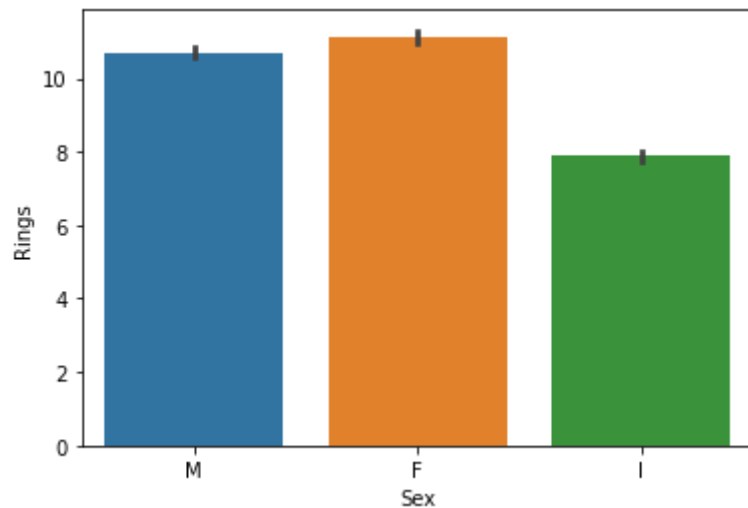
```
plt.bar(data['Sex'].head(20),data['Rings'].head(20))  
plt.title('Bar plot')  
plt.xlabel('Diameter')  
plt.ylabel('Rings')
```

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



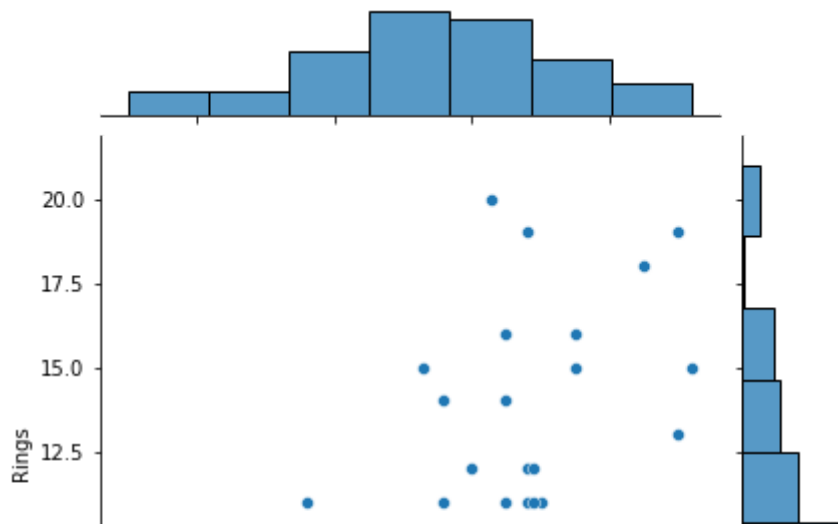
```
sns.barplot(data['Sex'], data['Rings'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa7844ee550>
```



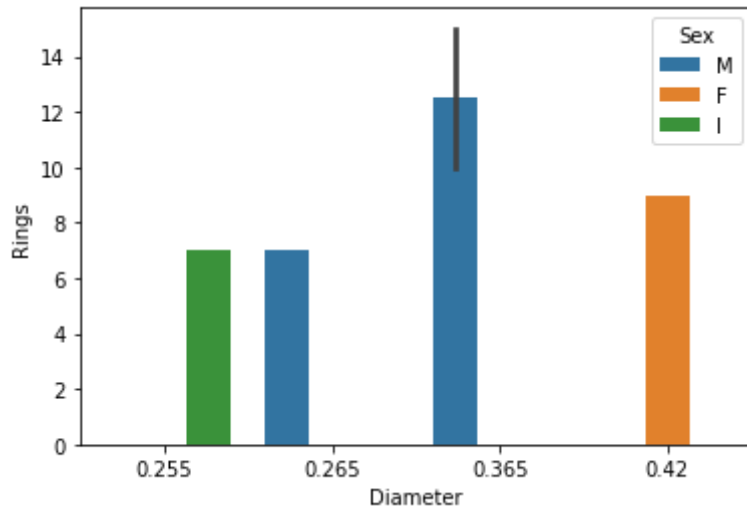
```
sns.jointplot(data['Diameter'].head(50),data['Rings'].head(100))
```

```
<seaborn.axisgrid.JointGrid at 0x7fa78447aa90>
```



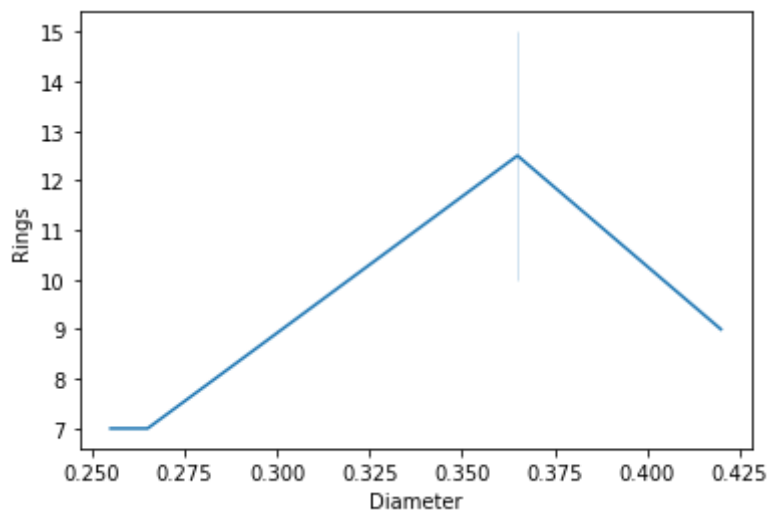
```
sns.barplot('Diameter', 'Rings', hue='Sex', data=data.head())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa781aea150>
```



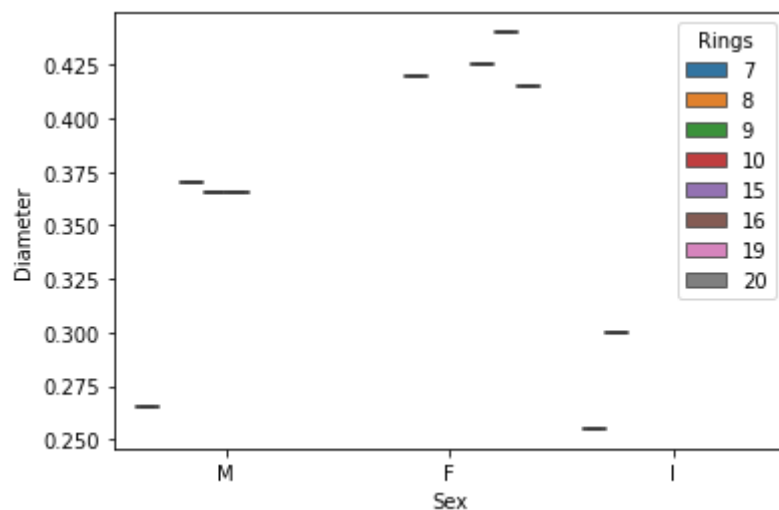
```
sns.lineplot(data['Diameter'].head(), data['Rings'].head())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa781a99950>
```



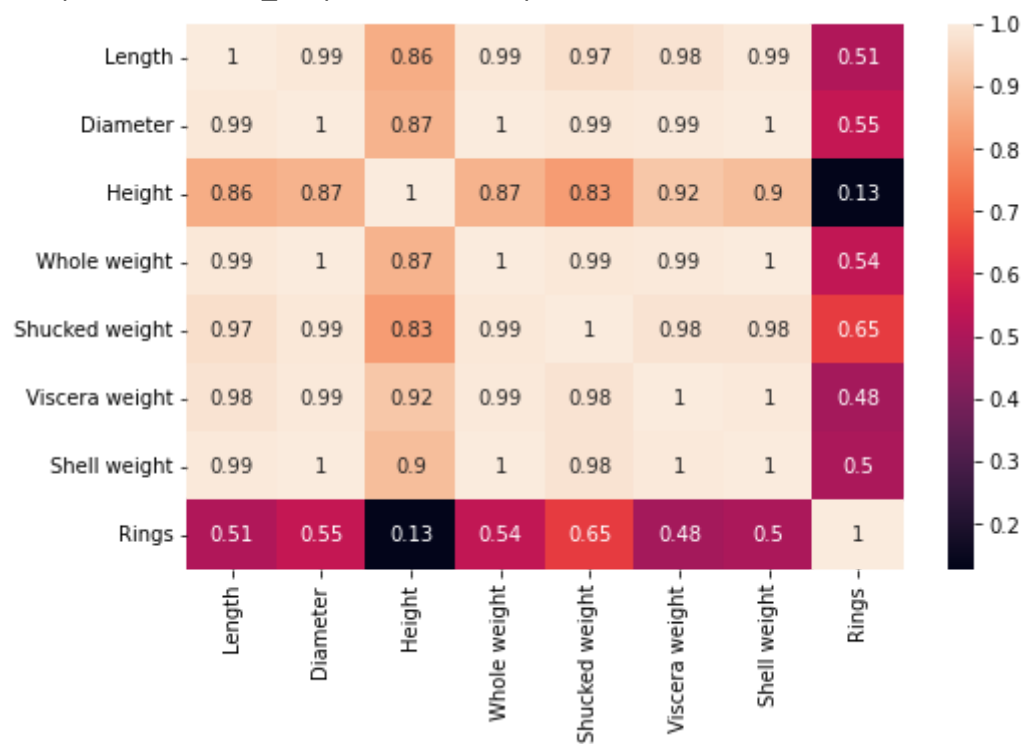
```
sns.boxplot(data['Sex'].head(10), data['Diameter'].head(10), data['Rings'].head(10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa7819ace50>
```



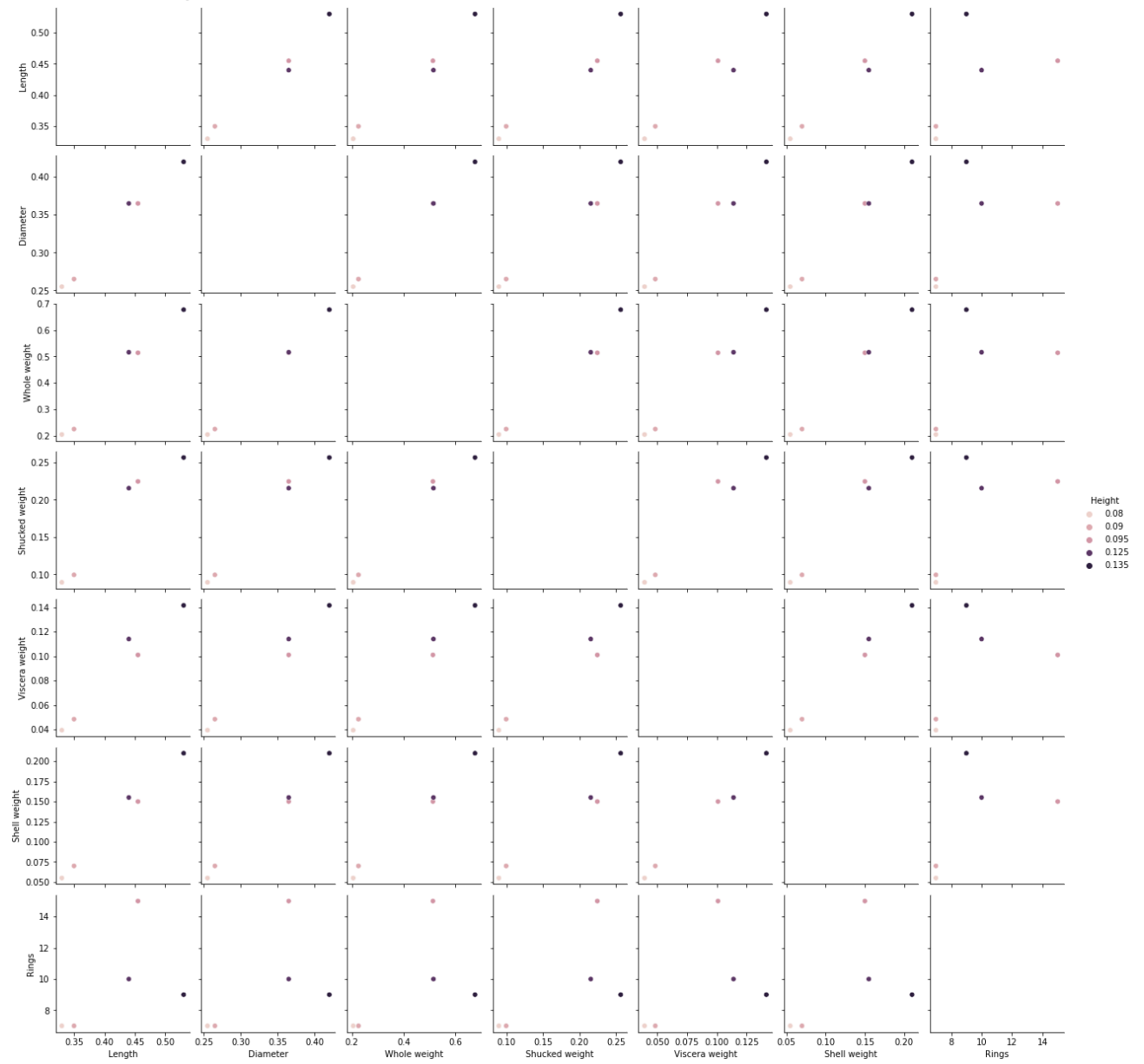
```
fig=plt.figure(figsize=(8,5))  
sns.heatmap(data.head().corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa781872110>
```



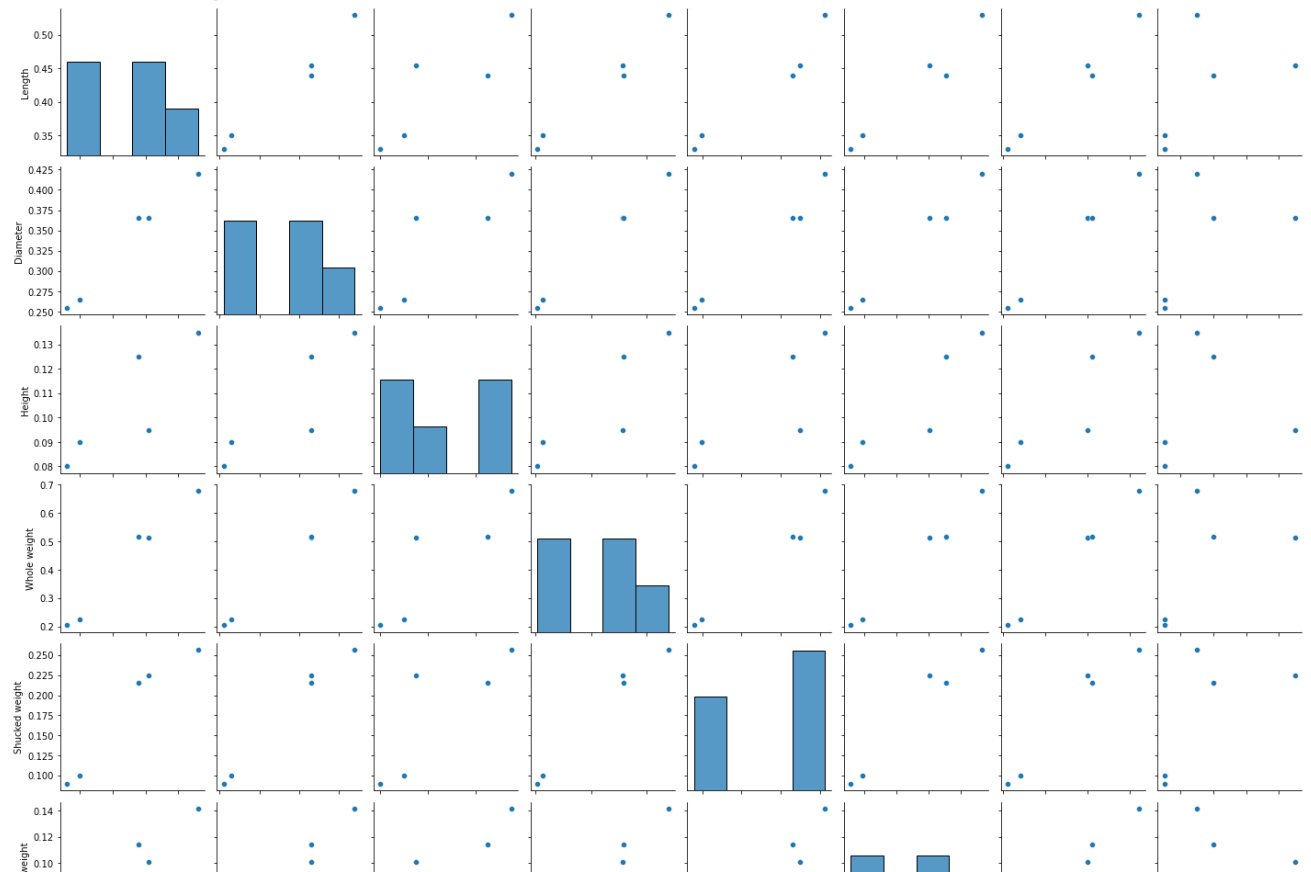
```
sns.pairplot(data.head(),hue='Height')
```

<seaborn.axisgrid.PairGrid at 0x7fa78173e990>



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


<seaborn.axisgrid.PairGrid at 0x7fa780284550>



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	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	



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

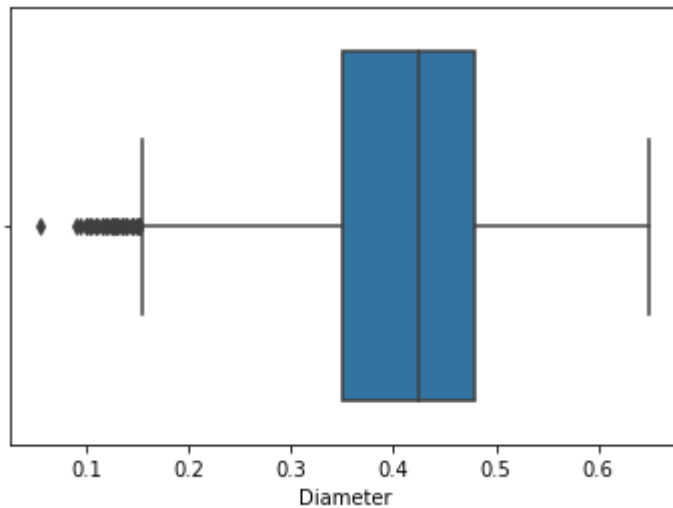
```
data.isna().any().sum()
```

0

5.Find the outliers and replace them outliers

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

<matplotlib.axes._subplots.AxesSubplot at 0x7fa77d0be450>



```
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
```

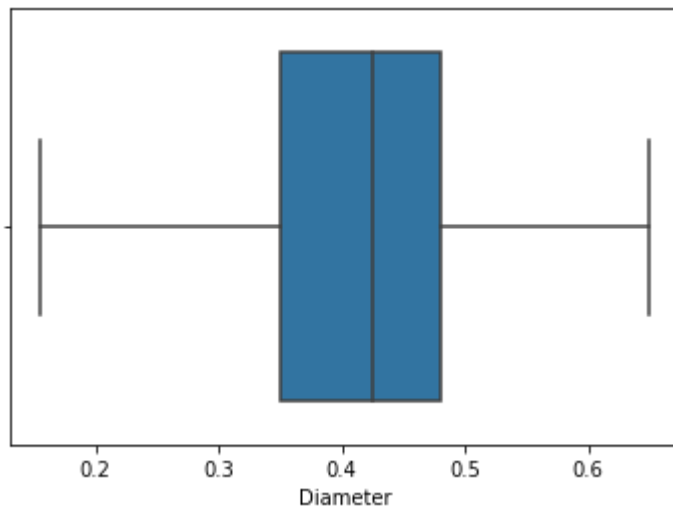
```
Whole weight    -0.62575
Shucked weight  -0.28800
Viscera weight  -0.14575
Shell weight    -0.16850
Rings           3.50000
dtype: float64
```

```
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 0x7fa77cfd8490>
```

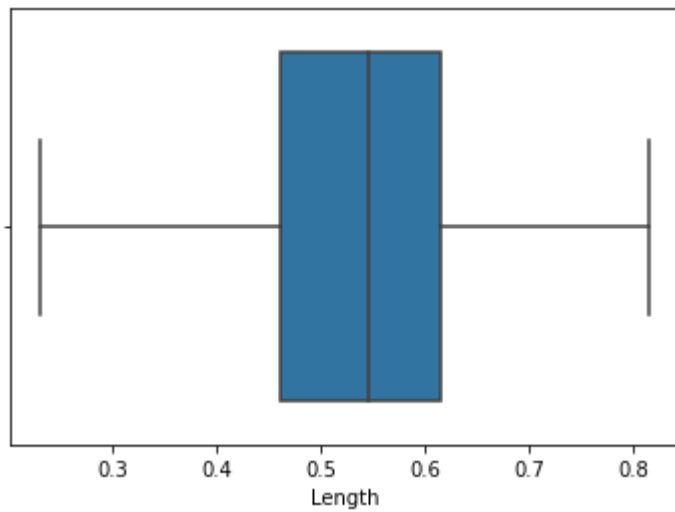


```
sns.boxplot(data['Length'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa77cea9b10>
```

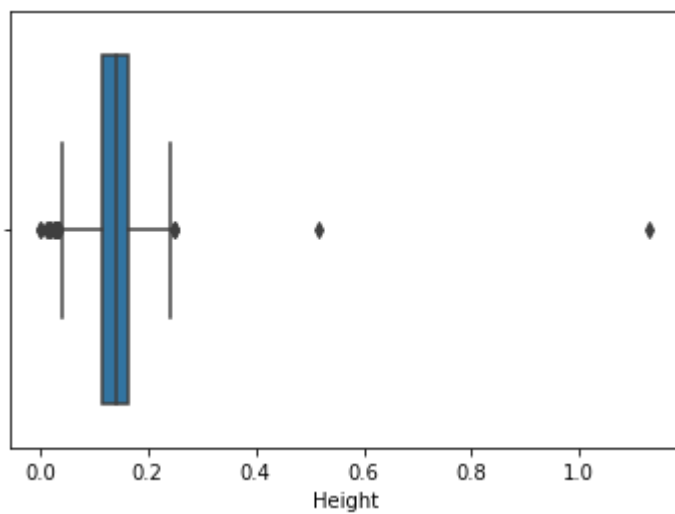
```
data['Length']=np.where(data['Length']<0.23,0.52, data['Length'])  
sns.boxplot(data['Length'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa77ce927d0>
```



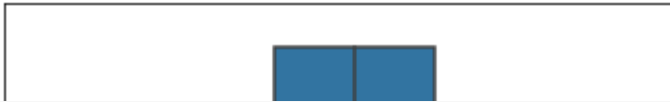
```
sns.boxplot(data['Height'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa77cdfb490>
```



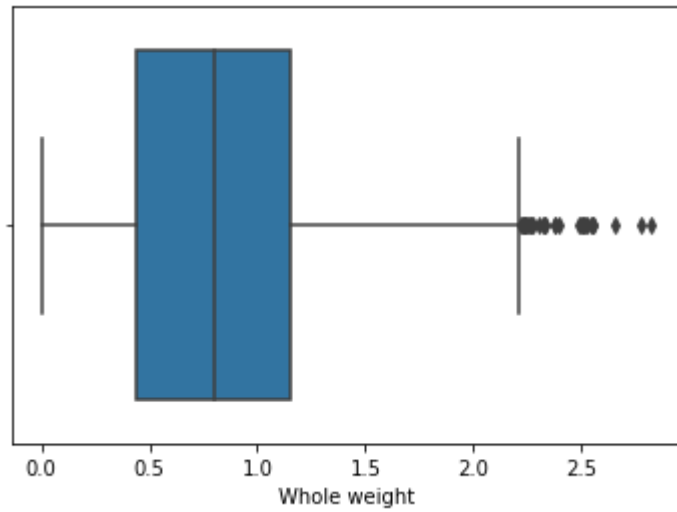
```
data['Height']=np.where(data['Height']<0.04,0.139, data['Height'])  
data['Height']=np.where(data['Height']>0.23,0.139, data['Height'])  
sns.boxplot(data['Height'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa77cf18950>
```



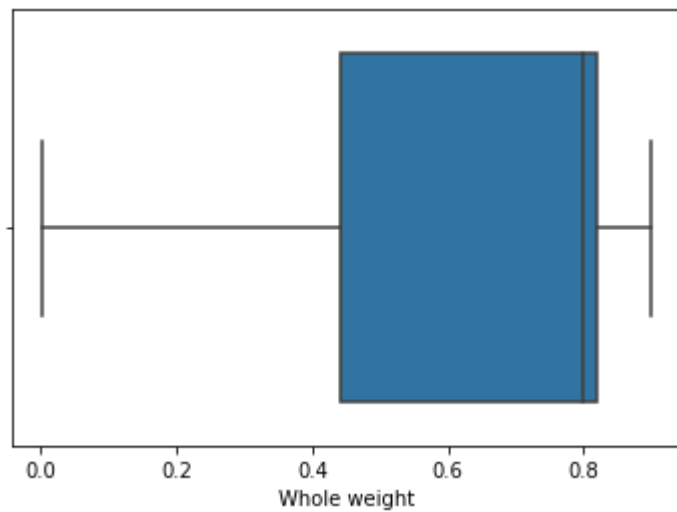
```
sns.boxplot(data['Whole weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa77ce97610>
```



```
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 0x7fa77ccc6f50>
```



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

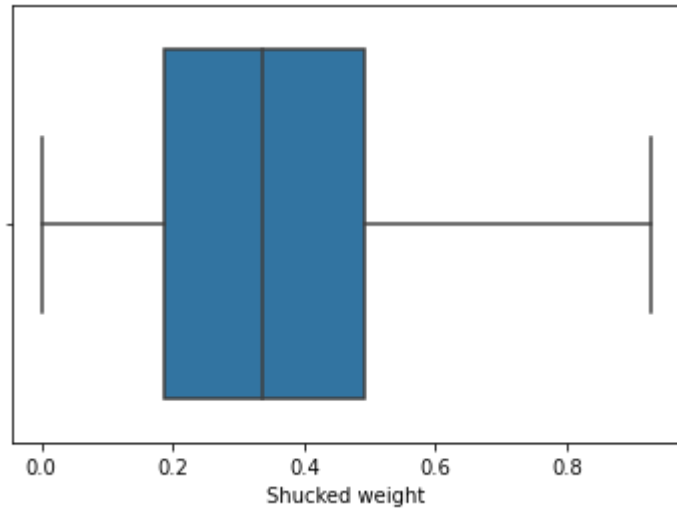


```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa77cc37310>
```



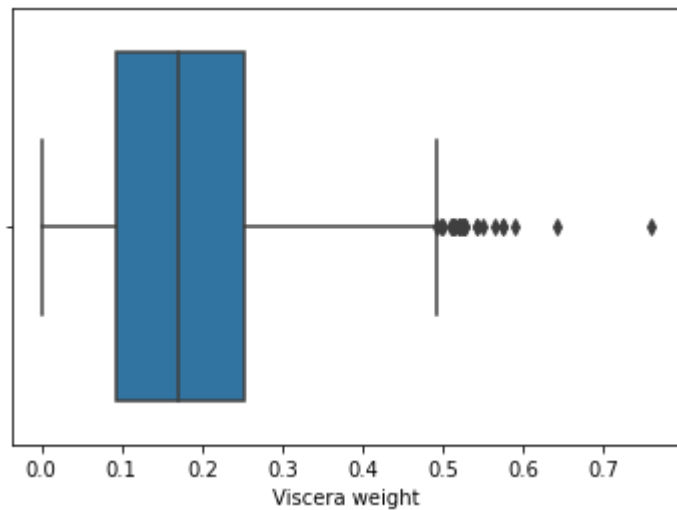
```
data['Shucked weight']=np.where(data['Shucked weight']>0.93,0.35, data['Shucked weight'])  
sns.boxplot(data['Shucked weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa784d100d0>
```



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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa77cb90b50>
```



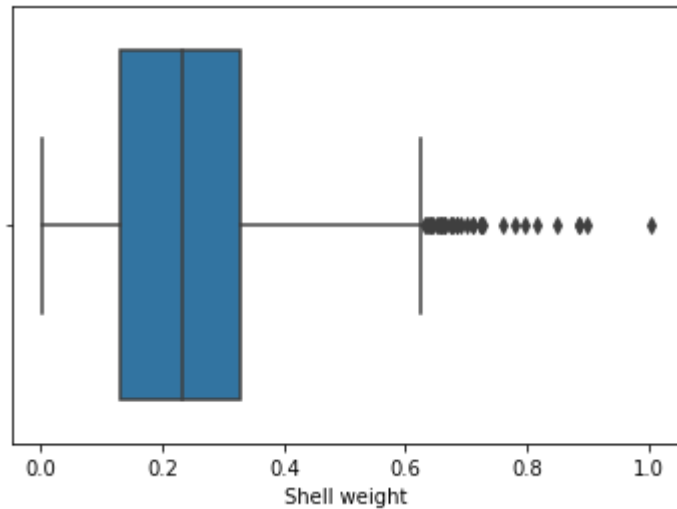
```
data['Viscera weight']=np.where(data['Viscera weight']>0.46,0.18, data['Viscera weight'])  
sns.boxplot(data['Viscera weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa77caf9fd0>
```



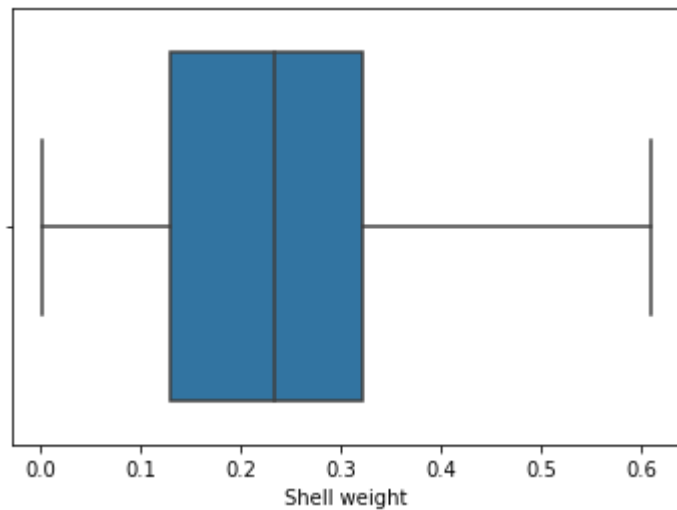
```
sns.boxplot(data['Shell weight'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa77cae3e90>
```



```
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 0x7fa77ca35150>
```



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
```

```
array([ 6.89310026,  7.8621574 ,  8.21495215, 13.37309216,  9.65584062,
        12.20165104,  8.21651582, 11.77358737, 11.76995239,  7.44646711,
        10.2512613 ,  4.19579579, 11.61306338,  8.99017945, 11.02269645,
         7.77815382, 10.9607397 ,  9.02190258,  7.61848022, 11.78776462,
        13.3839753 ,  7.93191395, 11.05668381,  9.27106421,  9.39666981,
        10.33776995,  8.08788808, 15.44875403, 10.14028923, 10.72717577,
        10.1931032 , 10.40597977,  7.48595575, 13.42151489, 11.80377497,
         8.23675034,  9.47824234,  8.77194434,  8.3854369 , 10.90889449,
         7.84067434,  7.74346362, 13.67871226,  8.99442625, 12.61179873,
        16.4351814 , 12.11796025,  8.41355311, 10.97581463, 12.20254156,
        10.62456104,  9.51963038,  8.64719271,  6.31918772,  7.82617854,
         8.84844505, 10.22316825,  7.91045699,  9.62457444,  8.67299711,
         8.7320808 , 14.17107906,  6.6678798 , 10.60525928, 11.48591442,
        11.64605847,  7.06752784,  7.02963973, 11.71583659, 13.01636665,
        10.02431728,  9.2953508 ,  7.02009252, 12.0066483 , 13.03968128,
         8.30800966, 11.07094159,  7.15490175,  8.48325782,  7.77061409,
        14.28768805, 11.24860903,  6.99406171, 11.12451347,  9.53806931,
        11.74363606, 10.2011371 , 13.09604009, 13.22621589, 11.20571195,
        10.2053655 , 12.47864984,  9.19574761,  7.48092132, 11.57576077,
         9.80241825, 10.85378643, 10.16140043, 10.07129986,  9.78348322,
        15.55646264,  9.29858777, 11.43154782, 10.34887403,  9.71496659,
        11.448179 ,  5.71351557,  6.53325773,  8.63951036, 10.54198637,
        12.06248746,  9.63918492, 11.12362647,  9.78767248, 10.43160779,
        12.6329735 ,  7.98527512, 12.27542368, 13.13960393, 14.04192582,
         8.92876112,  8.46323673, 11.87728639, 12.38779341,  9.19092103,
         6.26142836,  8.09959674,  9.52562812,  9.55079728, 11.93635623,
         5.49404371,  7.61040965,  8.40153419,  9.09728126,  7.06997294,
         6.56278952, 13.73671364,  7.14175946,  8.10537519, 10.93298148,
        11.34464912,  6.62227606, 12.36046281, 11.31070968,  9.18440622,
         8.89366662, 12.29178648,  8.18494928,  9.57994848,  8.14180698,
        12.5079604 , 10.69918584,  9.9876957 ,  9.8419938 , 10.77113233,
         7.40745431,  7.4920157 ,  6.71011404,  7.78675843, 11.89022356,
        10.11150983,  8.74055899, 10.33367221, 11.59328155,  4.21481826,
        12.31744447,  7.33780366,  8.78456574, 11.74790031,  9.59214565,
        12.26820435, 12.91146585, 14.18136571, 10.52954174, 10.77483346,
        11.10136547,  7.01218334,  9.84713121,  7.05979373, 12.22300459,
         8.24382819, 12.97486122,  6.06518427, 12.54209032, 11.67160281,
         8.19326621,  9.45111312, 10.98196637, 10.73171399, 13.12492142,
         7.83496851, 15.6849385 , 13.55703529, 10.2683613 , 10.86089773,
         7.72463922,  9.51691785, 13.03228426, 12.88252233, 10.12865026,
        11.36419516, 13.21103713,  9.22874758, 10.02129909,  9.90381913,
         9.45393537,  8.71574104,  8.94463987, 12.66290643,  9.91974598,
        10.34078253, 12.42239384,  8.48789762, 13.71025566,  8.76923656,
         8.92521157, 11.38173454,  7.02029061,  8.25083891, 11.66847087,
        11.72925841,  4.23350467,  7.98334934,  9.30322968,  8.43694041,
         8.96561827, 10.89786631, 12.23793448, 13.76137558,  9.13187444,
         8.68847423,  9.44412153, 10.86104017,  9.72091889,  7.07247467,
         8.42690025, 14.95838489,  6.8623433 , 11.45128063, 10.74600759,
        10.09632315, 10.81586231, 11.23345971,  4.52517363, 11.22323888,
         7.88265507, 10.35452817, 13.74279854,  8.73451754, 11.43935825,
         6.01875721, 10.4541057 , 12.97049986, 10.63118735,  8.93129116,
        10.01932768, 12.48295869, 10.66278766, 11.49252353, 10.90024949,
         8.99345374,  7.60812167,  6.64626315, 11.30507237, 13.6369384 ,
        10.05957262, 10.97457512,  9.98773446, 10.78269826,  9.03293993,
        10.71606107,  6.11196372,  6.92890642,  8.63951586,  6.49864545,
        12.2834418 ,  9.39811138, 10.79711174,  9.32409614, 13.77196045,
```

9.99745819, 4.83274398, 6.80342624, 8.75308531, 10.65546138,
11.58563111, 11.78820502, 7.17715716, 12.11560626, 10.72121578

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

```
array([10.46279564, 12.32100071, 9.83144512, ..., 10.07006072,
       10.81868708, 10.55692281])
```

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

```
0.4702855002721834
```

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

```
array([9.94159016])
```

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))
```

```
2.2354202875485663
```

LASSO

```
from sklearn.linear_model import Lasso, Ridge
#initialising 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.          ,  0.51718379,  0.15649197,
        0.          ,  0.          ,  0.78958185])
```

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

```
0.37071187099571534
```

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

2.436480989658682

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
```

```
10.5584285, 10.52275008, 8.05225785, 7.84880414, 11.27405424,
10.28190334, 13.8097873 , 11.57303296, 10.0378216 , 9.30810872,
7.87803342, 11.68425437, 13.53213016, 9.42683282, 11.4272792 ,
10.6072671 , 8.81038885, 7.02103587, 6.26249572, 11.27367769,
13.29858161, 6.69776306, 8.64379091, 11.96695658, 8.54732142,
6.06415474, 12.78393363, 10.75465789, 10.07475088, 7.58614034,
10.64717155, 9.24366721, 12.93853439, 10.91732646, 11.30288588,
9.75150107, 11.34815162, 11.65616651, 6.97103497, 11.2194177 ,
8.34055844, 11.35754141, 12.94964242, 6.40057426, 11.79456835,
10.02095852, 6.47284569, 10.84473998, 9.06568977, 11.39889183,
9.35441388, 12.15106278, 9.23601994, 6.01944187, 6.89756207,
11.48256515, 9.87539764, 10.16254993, 11.37367698, 13.83296928,
11.39913526, 10.04453458, 7.31126743, 8.93019277, 8.373329 ,
12.4657147 , 13.50082847, 9.96400656, 11.48090869, 12.30579678,
10.39847479, 11.84351948, 7.58481946, 9.89197913, 11.16307988,
8.00936217, 11.34660719, 9.57055004, 11.23453597, 8.76532697,
9.54352474, 11.2172432 , 11.92830244, 6.65589801, 10.37107509,
10.23485661, 8.20183051, 14.91932956, 10.58829331, 10.50085114,
10.81188346, 10.8666431 , 10.75393834, 10.81042592, 8.65342235,
13.94722357, 10.61686848, 9.16861323, 8.97011326, 6.65748205,
7.86418399, 9.81607979, 9.03162654, 7.50790447, 12.81933189,
7.53997238, 11.11309321, 5.71614854, 10.1875337 , 10.46100302,
10.85425494, 11.26251017, 10.54581988, 13.00242389, 10.00898975,
6.49229514, 9.86911974, 10.05560188, 7.18307085, 11.66197245,
10.95853233, 7.20683116, 9.73497608, 11.95299317, 12.3284704 ,
11.14674777, 9.84263098, 9.857322 , 6.89351245, 9.67601842,
8.86834711, 6.39184361, 10.32749453, 9.25303516, 11.11359605,
12.08559246, 7.93020502, 13.5598376 , 10.47429098, 9.31350932,
6.63584897, 10.8705064 , 9.64368446, 10.26394622, 10.2360594 ,
7.4966218 , 13.49487618, 8.57207231, 11.28140441, 10.05945935,
12.45105082, 7.01640402, 10.76769808, 7.85430124, 8.77635567,
7.53717564, 8.11589484, 10.80288825, 9.3651218 , 11.32268917,
11.33376913, 10.77837482, 10.24671732, 9.16034865, 10.19270066,
9.27510346, 6.30721074, 11.56149751, 4.56866994, 12.82678405,
10.83235195, 13.90182724, 11.05711349, 13.00158477, 10.68357141,
8.74497553, 11.52351577, 10.3536453 , 16.58116289, 9.83799412,
11.68359358, 6.81968291, 11.92486797, 11.0101736 , 13.75554249,
8.06601627, 9.15394781, 7.97798682, 11.99009622, 12.08608797,
8.15189916, 9.60061663, 12.26760197, 11.74758619, 10.89867171,
12.83082472, 9.17695072, 10.57750322, 8.80840622, 7.89089606,
11.90311481, 11.85786736, 7.58720788, 9.7962279 , 9.92251594,
12.97906537, 10.02685044, 12.55833557, 11.13999039, 8.61496753,
7.6659044 , 10.70592109, 6.80026644, 10.83600453, 10.18950615,
```

```

9.13756879, 12.52572879, 10.05484218, 12.95213451, 9.81219632,
10.86655715, 8.95787582, 9.33415363, 11.6466516 , 7.28192684,
12.47396775, 11.13223364, 7.43857685, 7.48265491, 11.42958845,
9.46785255, 6.79580251, 9.07110024, 5.9075498 , 10.43218759,
10.50208924, 10.20587372, 8.64854445, 12.44158425, 8.3592836 ,
10.65238816, 10.95426484, 10.24088813, 6.88718401, 10.16672307,
12.12961804, 8.88539444, 5.92958015, 9.32232018, 12.27709724,
7.6662387 , 9.29647413, 9.66928994, 8.03987937, 10.89794725,
10.92384176, 10.58607159, 11.97342471, 6.34364149, 13.3020735 ,
11.259069 , 7.66626477, 10.52182446, 8.06961982, 8.13916524,
6.63874985, 9.33170693, 11.42403733, 12.41235388, 8.71674282,
9.86961137, 9.48153448, 7.83644164, 13.32562235, 11.36146913,
4.53688218, 10.37898537, 6.08212191, 7.02657522, 10.40645716,
11.65282887, 12.6642297 , 11.18731043, 10.16416161, 6.54818793,
14.76013648, 8.14465949, 12.94715727, 11.75570227, 8.71598662,
11.72349248])

```

```
rg.coef_
```

```

array([-0.2971817 , -0.71714541,  0.32134212,  1.00119834,  0.9368208 ,
       -1.49109993, -0.02497205,  1.76455729])

```

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

```
0.47109389479038755
```

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

```
2.2337139046896075
```