Loading the dataset

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

```
d = pd.read_csv('/content/abalone.csv')
```

Perform visualization

Univariate analysis

```
d.head()
```

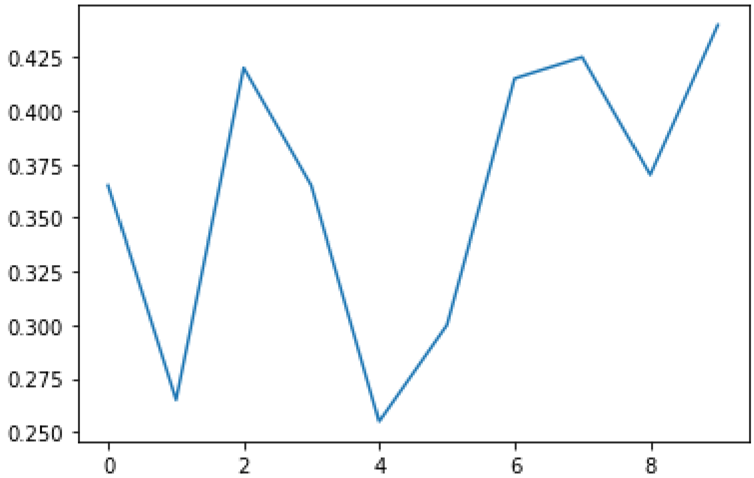
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395

```
plt.hist(d['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>)
```

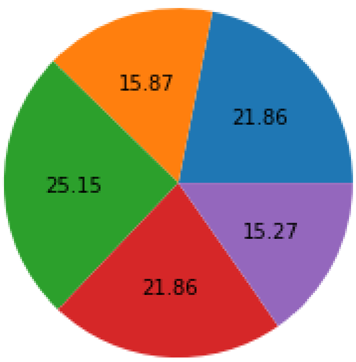
```
plt.plot(d['Diameter'].head(10))
```

```
[<matplotlib.lines.Line2D at 0x7f079f97c790>]
```




```
plt.pie(d['Diameter'].head(),autopct='%.2f')
```

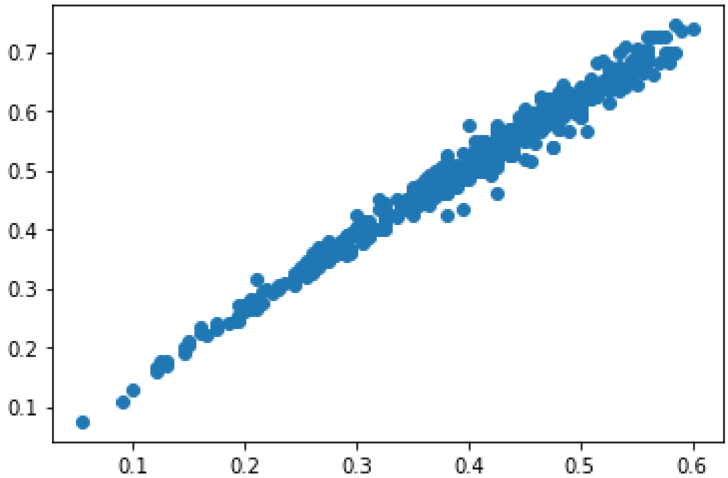
```
([<matplotlib.patches.Wedge at 0x7f079f890150>,
  <matplotlib.patches.Wedge at 0x7f079f890910>,
  <matplotlib.patches.Wedge at 0x7f079f8981d0>,
  <matplotlib.patches.Wedge at 0x7f079f898ad0>,
  <matplotlib.patches.Wedge at 0x7f079f8a4650>],
 [Text(0.8507215626110557, 0.6973326486753676, ''),
  Text(-0.32611344931648134, 1.0505474849691026, ''),
  Text(-1.0998053664078908, -0.02069193128747144, ''),
  Text(-0.08269436219656089, -1.096887251480709, ''),
  Text(0.9758446362287218, -0.5076684409569241, '')],
 [Text(0.46402994324239394, 0.3803632629138369, '21.86'),
  Text(-0.17788006326353525, 0.5730259008922377, '15.87'),
  Text(-0.5998938362224858, -0.011286507974984419, '25.15'),
  Text(-0.045106015743578656, -0.5983021371712958, '21.86'),
  Text(0.5322788924883937, -0.2769100587037768, '15.27')])
```



Bivariate Analysis

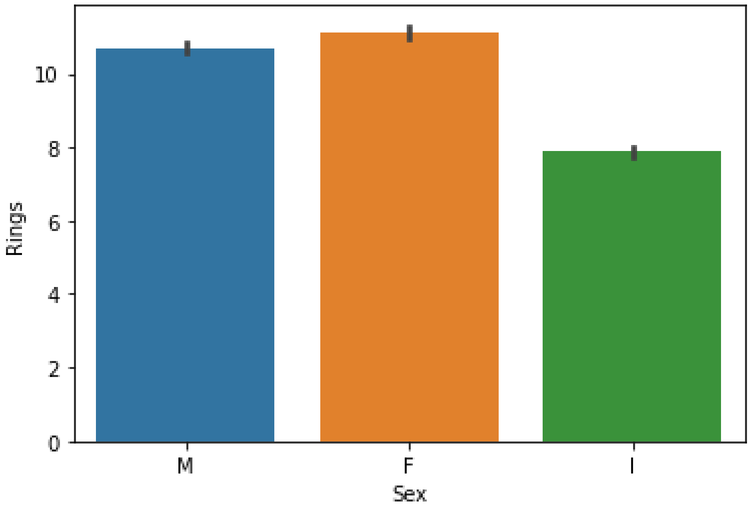
```
plt.scatter(d['Diameter'].head(500),d['Length'].head(500))
```

```
<matplotlib.collections.PathCollection at 0x7f079f869fd0>
```



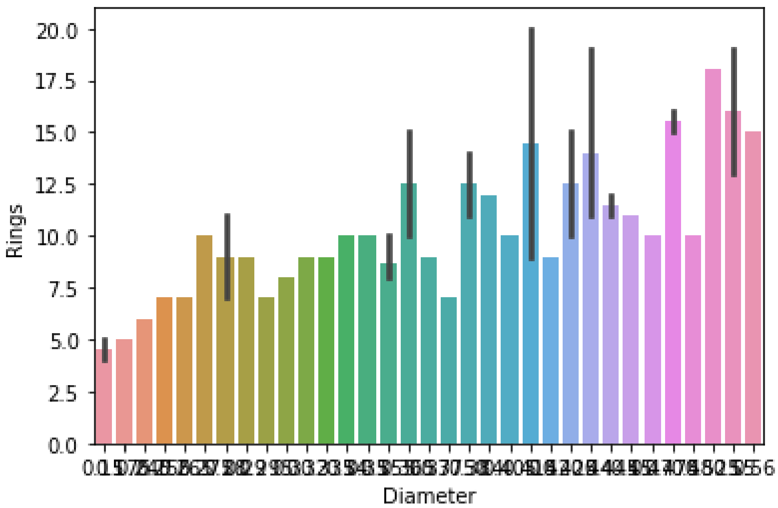

```
sns.barplot(d['Sex'], d['Rings'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f079f7be650>
```



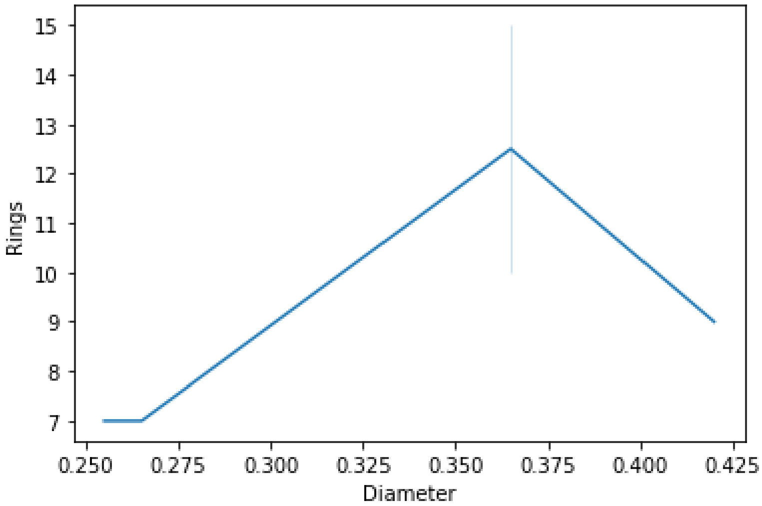

```
sns.barplot(d['Diameter'].head(50),d['Rings'].head(50))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f079f7b0290>
```



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

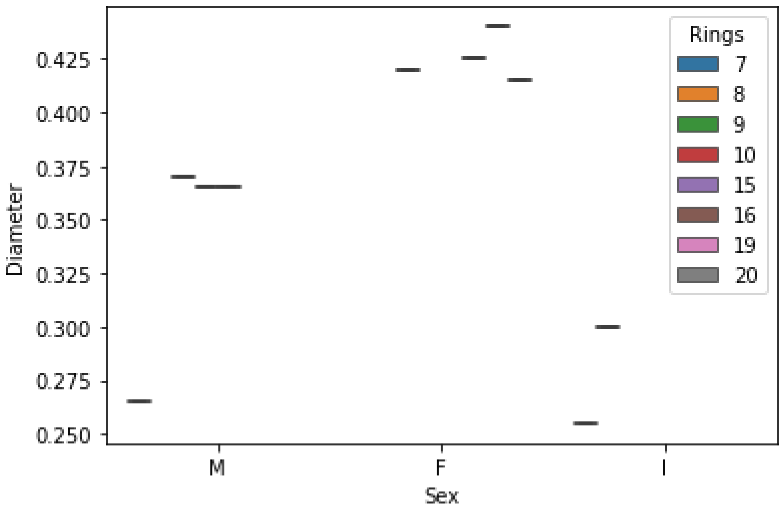
<matplotlib.axes._subplots.AxesSubplot at 0x7f079f7b3990>



Multivariate Analysis

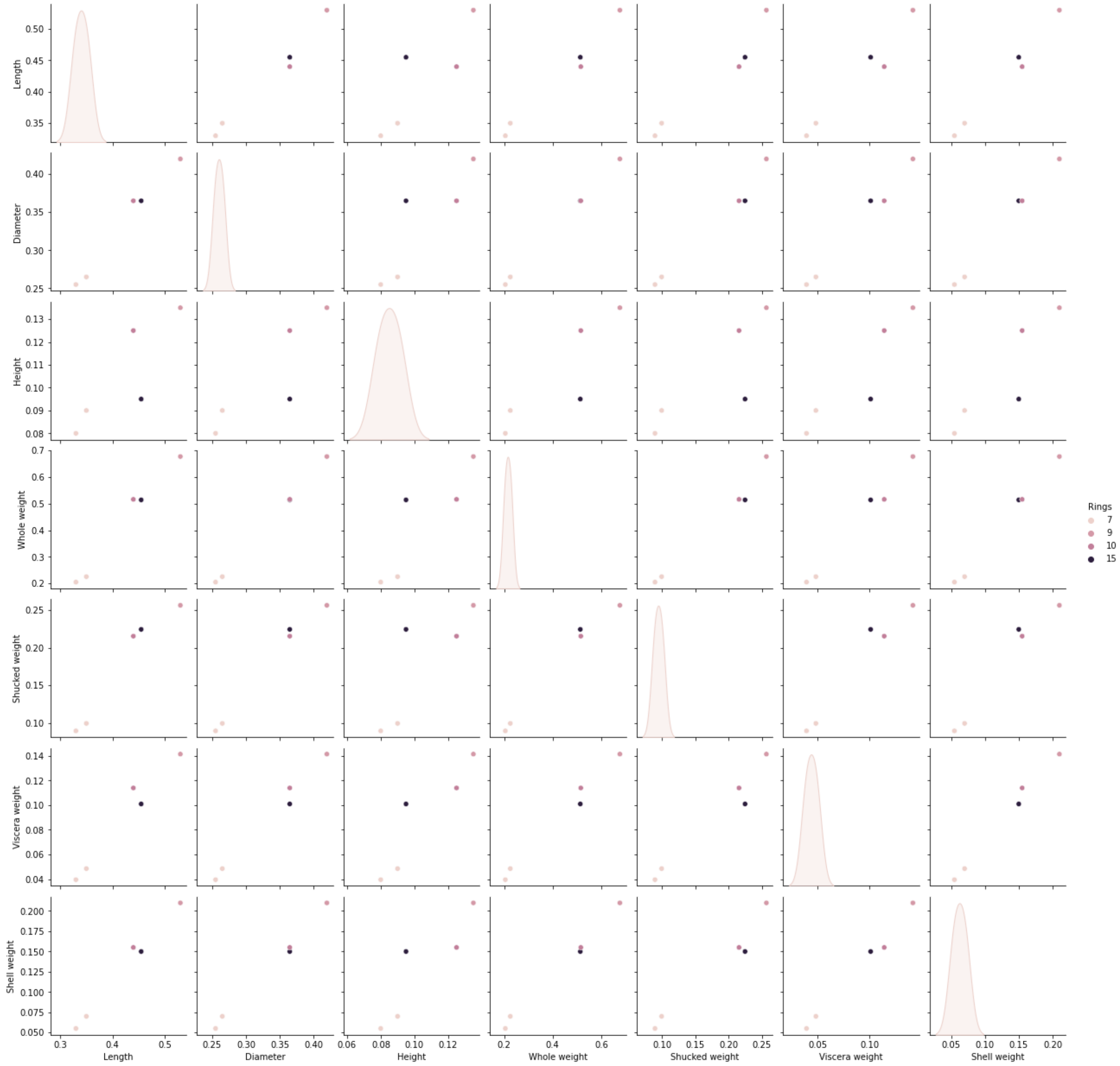
```
sns.boxplot(d['Sex'].head(10),d['Diameter'].head(10),d['Rings'].head(10))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f079fef4550>




```
sns.pairplot(d.head(),hue='Rings')
```

```
<seaborn.axisgrid.PairGrid at 0x7f079f382a50>
```



Perform the descriptive stastics on the dataset

```
d.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

```
d.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

```
d.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
```

```
d.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

```
d.shape
```

```
(4177, 9)
```

```
d.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
```

```
d.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
```

Check for Missing values and deal with them

```
d.isna()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	False	False	False	False	False	False	False	False	False
4174	False	False	False	False	False	False	False	False	False

```
d.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
```

```
d.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
```

```
d.isna().any().sum()
```

```
0
```

Find the outliers and Replace the Outlier

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f079b641210>
```

```
qnt=d.quantile(q=[0.25,0.75])
qnt
```

	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=qnt.loc[0.75]-qnt.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
```

```
lower=qnt.loc[0.25]-(1.5*iqr)
lower
```

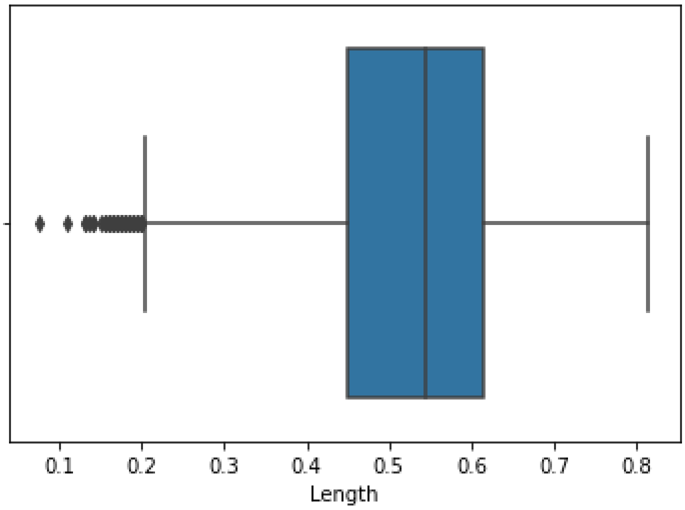
```
Length             0.20250
Diameter           0.15500
Height             0.04000
Whole weight      -0.62575
Shucked weight    -0.28800
Viscera weight    -0.14575
Shell weight      -0.16850
Rings              3.50000
dtype: float64
```

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

<matplotlib.axes._subplots.AxesSubplot at 0x7f079b74cd10>

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

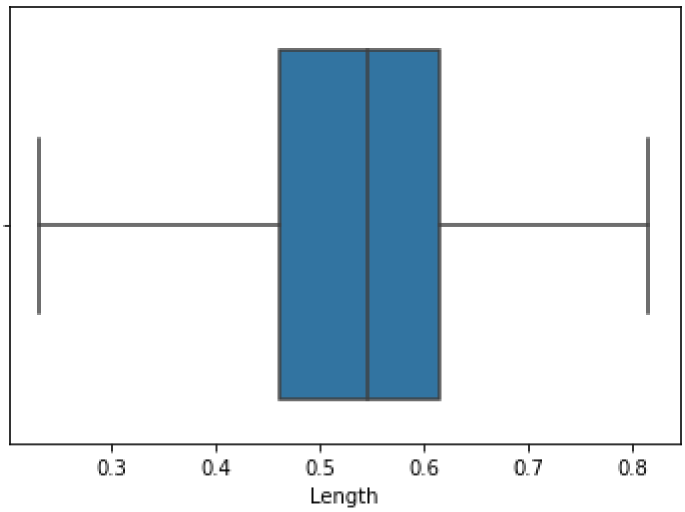
<matplotlib.axes._subplots.AxesSubplot at 0x7f0799d3d490>




```
d['Length']=np.where(d['Length']<0.23,0.52, d['Length'])
```

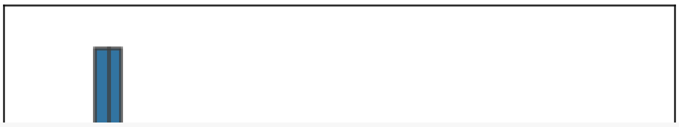
```
sns.boxplot(d['Length'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0799cbb850>




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

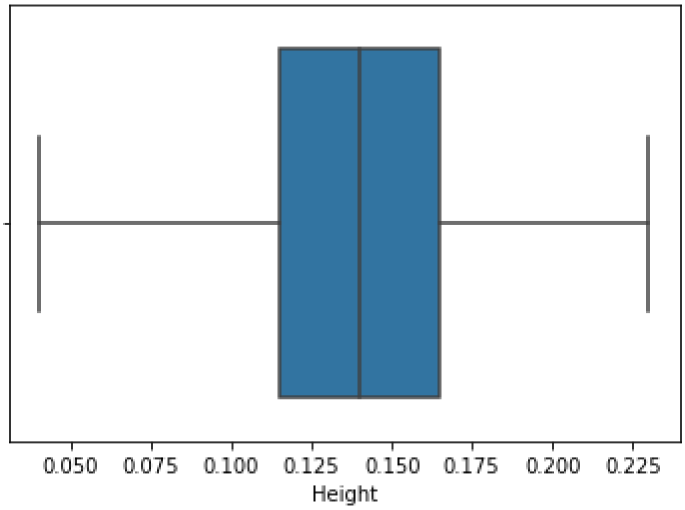
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0799c99ed0>
```



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

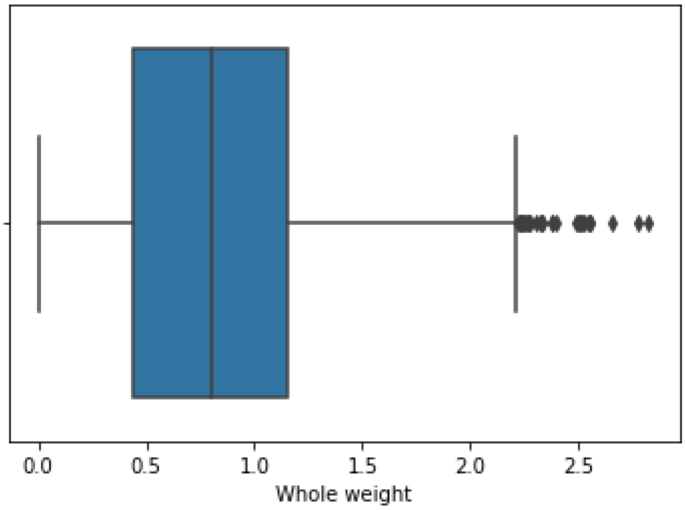
```
sns.boxplot(d['Height'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0799c0f350>
```




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

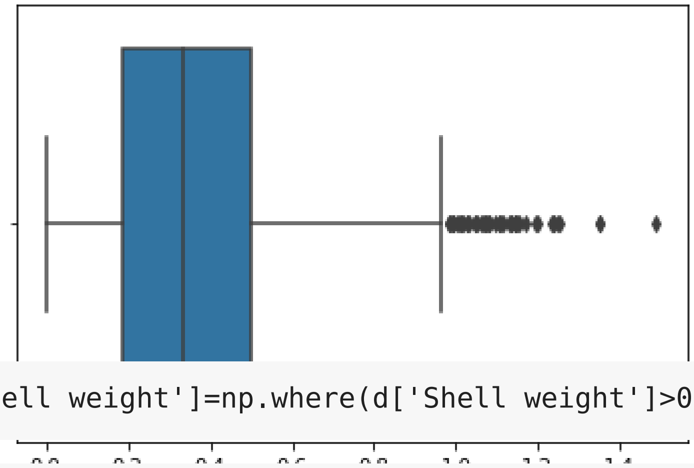
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0799be3250>
```




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

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

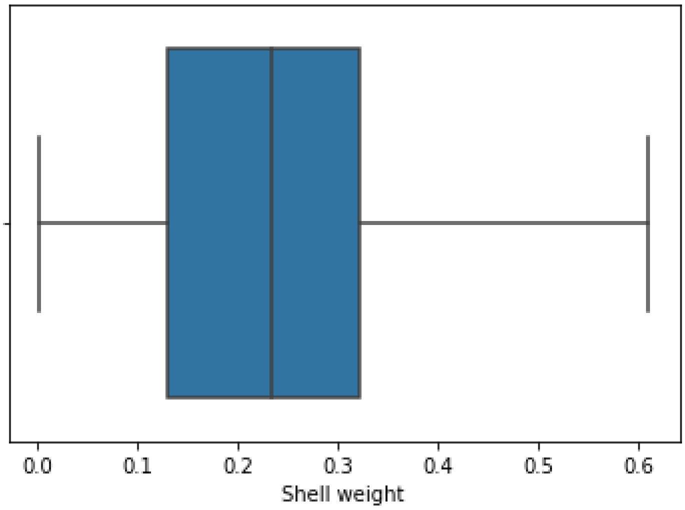
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0799b69ed0>
```



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

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0799ac9610>
```



Check for Categorical column and perform encoding

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

	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

Split the data into dependent and independent variables

[ ] ↳ 2 cells hidden

Scale the independent variables

[ ] ↳ 2 cells hidden

4177 rows × 9 columns

Split the Data into training and testing

[ ] ↳ 2 cells hidden

Build the model

[ ] ↳ 1 cell hidden

Train the model

[ ] ↳ 1 cell hidden

Test the model

[ ] ↳ 4 cells hidden

Measure the performance using metrics

[ ] ↳ 2 cells hidden

LASSO

[ ]

RIDGE

```
rg=Ridge(alpha=0.01,normalize=True)
```

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

```
Ridge(alpha=0.01, normalize=True)
```

```
rg_pred=rg.predict(x_test)
rg_pred
```

```
array([11.64700359, 11.42842481,  9.43499713,  9.32392864, 12.15627403,
       10.66262315,  6.0328879 ,  9.35015904,  8.52722423, 10.70641973,
       12.41676889, 13.50364278,  6.40640722,  6.74314259, 10.69745163,
       11.31553953, 10.05981979, 11.03324423, 13.34618397, 12.86909971,
       10.19139966,  9.81882077, 10.64152475,  8.23405669,  7.89182332,
        7.84325901, 12.3049092 ,  9.29951043,  9.02940379,  9.4215392 ,
        8.40589358,  9.64898959, 12.50446709, 11.43285486,  6.77439574,
       10.75463325,  8.04377959,  5.76360466, 14.74176604, 11.47168951,
       11.51879226,  5.92003071,  7.10373386, 10.17072827,  9.0707484 ,
        7.90799393, 12.00612941, 10.80666312,  7.92802412,  9.86222749,
        9.48957167, 13.46591111, 13.12701692,  7.88454519, 11.72413619,
       12.91893542,  9.9872087 , 10.44183445,  6.92813703,  8.65511369,
       12.05837405, 11.05798956,  7.12153746,  9.48436077, 11.50002986,
        9.82396061,  9.14607247, 11.96893408,  6.6514697 ,  9.08308591,
       14.11860806, 12.05999709,  9.21826407,  9.80361134, 11.58771694,
       10.79341009, 14.08574183, 12.58847017, 10.16712085,  6.48724656,
       11.4412772 ,  8.83393988, 11.38619917, 10.61345475,  7.97225769,
        9.61800105,  8.02381322,  6.95030699,  7.07461767, 10.10500748,
        6.80171758, 10.95838156,  7.11471292,  9.64942587,  7.64927651,
       11.46799064, 12.83229226, 11.61819942,  8.37186011, 11.70731818,
       11.55607298,  7.60587665,  7.09956462,  8.23529485,  8.8343559 ,
        8.83069065, 11.3283293 , 10.77124467,  9.41579082, 11.0428856 ,
        9.64644229,  8.16268378,  8.67489686, 10.62081839, 10.08633742,
       10.97770913, 12.66186474, 10.61314616,  6.03676936,  8.30668633,
       10.86376571, 12.8020696 ,  5.91698459, 10.160743  ,  8.22492202,
       10.88506839,  7.15722565, 11.38999961,  8.08406878, 10.04198306,
       10.61462702,  6.87763414, 12.10327142, 10.4364496 ,  8.63537291,
        7.61146399,  9.30509569,  9.31823422,  8.9155313 ,  5.78226259,
        4.91995226,  9.1473559 , 11.60601171,  9.11035745,  8.95885363,
        5.12241776, 10.6631954 , 12.05054039, 12.34658321, 10.78410736,
       10.4896159 ,  9.59251965, 10.15367433,  9.9914951 , 12.71079208,
        9.76779331,  5.6324325 ,  6.54279012,  7.80323273, 12.43336795,
        9.33472824, 10.22294201, 11.21312496, 12.60261756,  7.88062178,
       10.50855939, 11.1487998 , 11.70937657, 11.78808402, 10.38717833,
        8.69704591,  6.32698706, 12.56652788,  9.85913316, 10.15742855,
       10.84398055,  8.50732366,  7.7502726 , 14.05569056,  9.76533475,
        7.74578462,  9.03837552, 12.35542301, 13.74987929,  7.67174586,
        6.40608353,  6.5860754 , 11.12995536, 11.79652774, 12.50074109,
        8.23103724, 14.76924037, 10.1471261 ,  9.37788943,  7.29893411,
```

```
        8.89624114, 14.76351705,  8.79684319, 12.79443562, 11.75577235,
        6.75188629, 11.08634507, 12.21751   , 12.27266551, 10.36130386,
       12.26162097, 15.75670394,  5.61229575,  5.36044997,  7.44423989,
        9.97921082, 11.76463271, 12.11739638, 12.4871439 , 13.09112428,
        6.78073143,  7.47797529,  6.32351034, 11.14633149,  9.22529064,
       11.52155941, 11.03903576, 11.22296093, 10.77208672,  8.7782633 ,
        9.18161649,  9.55926771, 10.52363855, 10.3501496 ,  8.01031509,
       11.44072487, 13.50779466,  7.40142993,  7.55755412,  8.88144185,
        4.57893315,  9.39709337,  9.03111482, 11.48432858,  6.56458861,
       11.3825303 ,  9.31036182, 10.34900556,  5.9780508 ,  5.99561561,
       13.16434388,  7.85265927, 11.37483624, 12.43031006, 11.32049169,
        6.02664947,  8.81847124, 11.00656069, 11.75424398,  9.48337478,
       13.4867177 ,  8.40227002,  7.64766384,  8.07276202, 11.14796801,
        7.6618817 , 11.60947783,  4.01231736,  8.69720556, 11.2113068 ,
       10.81846734,  9.72856703,  6.23782707,  8.56416621,  9.10602092,
       16.30089766, 12.61879688,  9.81101747, 10.60896902, 11.86558917,
        7.97155832, 11.02158753, 11.02573458, 10.08517598, 10.17328851,
        6.32373852,  8.78270762,  9.64519434,  9.61061123,  6.35367976,
        9.02908934, 13.10147448, 12.19155567,  7.9908791 , 11.8937656 ,
```

```
rg.coef_
```

```
array([-0.22437736, -0.70969267,  0.4418621 ,  0.88051191,  0.66291868,
       -1.70529939,  0.53128287,  1.72113842])
```

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

```
0.45198921381183166
```

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

```
2.5477847772203694
```

0s completed at 10:08 PM