Assignment -3

Assignment Date	01 October 2022
Student Name	B. Gayathiri
Student Roll Number	811519104030
Maximum Marks	2 Marks

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

```python
[2]: d=pd.read_csv("abalone.csv")
     d.head()
```

```
[2]:   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

        Shell weight  Rings
     0        0.150     15

     1        0.070      7

     2        0.210      9

     3        0.155     10

     4        0.055      7
```

```python
[3]: 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-    float64
     null
 6   Viscera weight 4177 non-    float64
     null
 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
[7]: for i in d.columns:
         print(d[i].value_counts())
```

```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
0.625 94
0.550 94
0.575 93
0.580 92
0.600    87
         ..
0.075 1 0.815
 1 0.110     1
0.150 1
0.800    1
Name: Length, Length: 134, dtype: int64
0.450 139
0.475 120
0.400 111
0.500 110
0.470    100
         …
0.610 1 0.650
 1 0.620     1
0.095 1
0.615      1
Name: Diameter, Length: 111, dtype: int64
0.150 267
0.140 220
0.155 217
0.175 211
0.160 205
0.125 202
0.165 193
0.135 189
0.145 182
0.130 169
0.120 169
0.170 160
0.100 145
0.110    135
0.115 133
0.180 131
0.090 124
0.105 114
0.185 103
0.190 103
0.095 91 0.195
 78 0.080    76
0.085 74 0.200
```

```
 68 0.075    61
0.070 47 0.205
 45 0.065    39
0.215 31 0.060
 26 0.055    25
0.210 23 0.050
 18 0.220    17
0.040 13 0.225
 13 0.045    11
0.230 10
0.030 6 0.035
 6 0.235     6
0.025 5 0.240
 4 0.250     3
0.020 2 0.015
 2 0.000     2
0.010 1 0.515
 1
1.130     1
Name: Height, dtype: int64
0.2225    8

1.1345    7

0.9700    7

0.4775    7

0.1960    7

..

0.0475    1

1.8930    1

1.8725    1

2.1055    1

1.9485    1
Name: Whole weight, Length: 2429, dtype: int64
0.1750    11

0.2505    10

0.0970     9

0.0960     9

0.4190     9

..

0.4175     1

0.1935     1
```

```
0.1790    1

0.1275    1

0.9455    1

Name: Shucked weight, Length: 1515, dtype: int64
0.1715    15

0.1960    14

0.0575    13

0.0610    13

0.0370    13

..

0.4270    1

0.4075    1

0.4920    1

0.4650    1

0.5260    1

Name: Viscera weight, Length: 880, dtype: int64
0.2750    43

0.2500    42

0.2650    40

0.3150    40

0.1850    40

..

0.0060    1

0.6460    1

0.5010    1

0.3295    1

0.0920    1

Name: Shell weight, Length: 926, dtype: int64
9     689
10    634
8     568
11    487
7     391
12    267
6     259
```

```
13    203
14    126
5     115
15      103
16       67
17       58
4      57
18       42
19       32
20       26
3      15
21       14
23        9
22        6
27        2
24        2
1       1
26        1
29        1
2       1
25        1
Name: Rings, dtype: int64
```

```
[8]: d.isnull().sum()
```

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

```
[9]: d.duplicated().value_counts()
```

```
[9]: False4177
     dtype: int64
```

1 Data visualization(EDA Analysis)

```
[12]: sns.boxplot(data=d,x="Length")
```

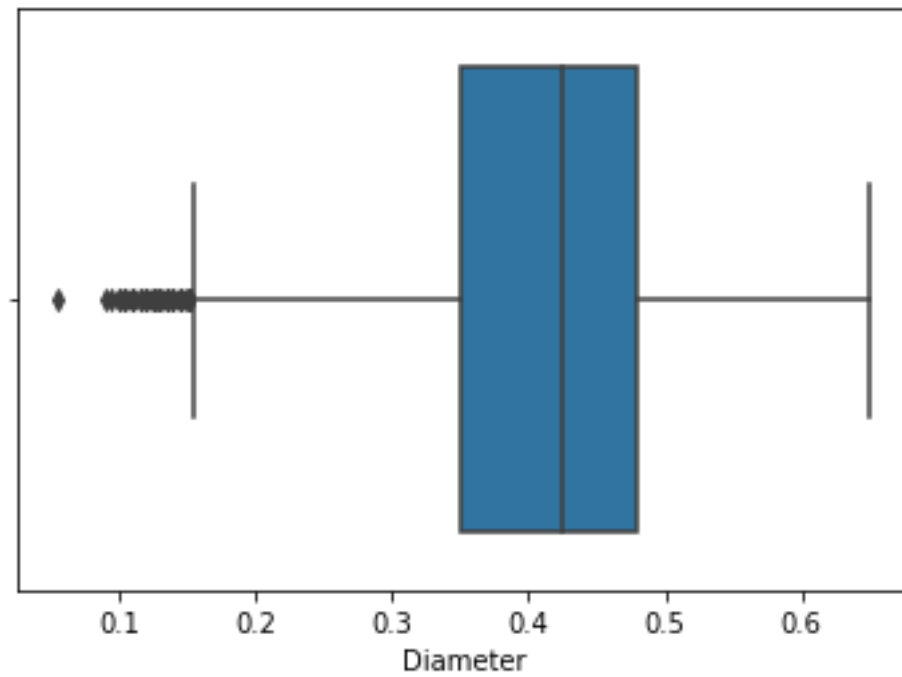[12]: <AxesSubplot:xlabel='Length'>
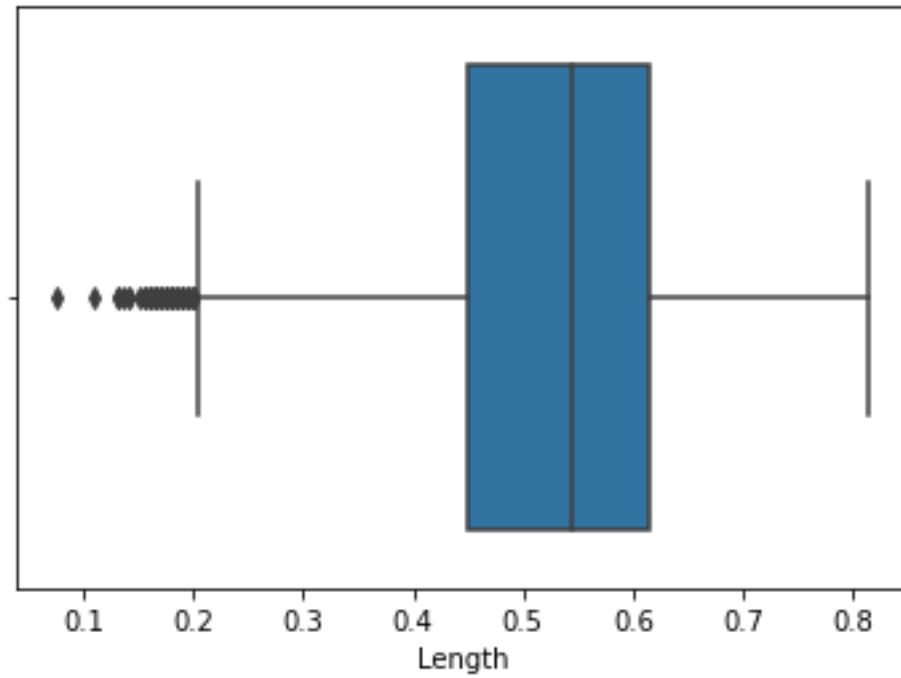
[13]: ```
sns.boxenplot(data=d,x="Length")
```

[13]: <AxesSubplot:xlabel='Length'>

```
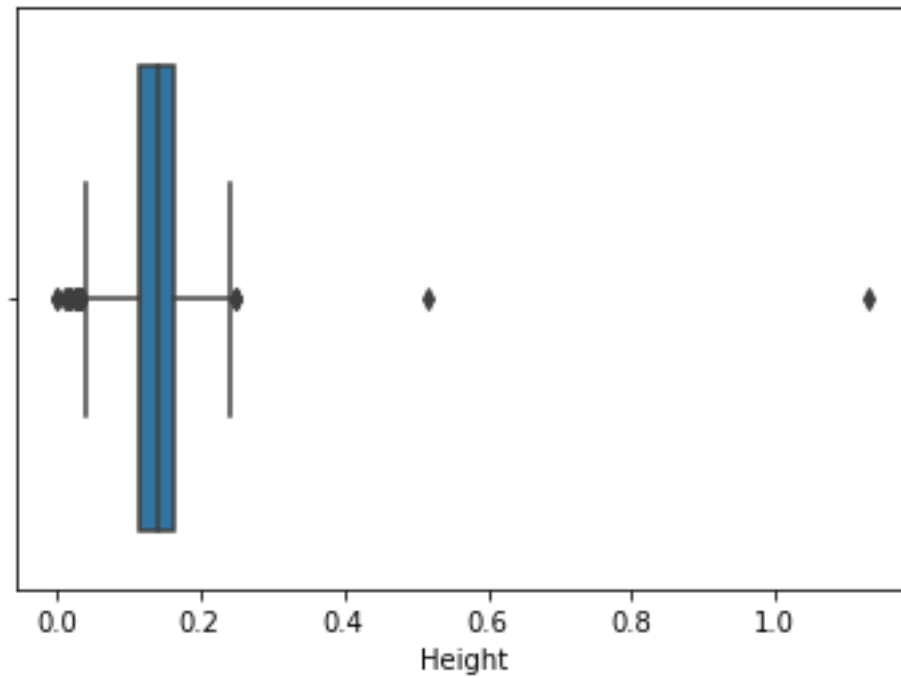[16]: sns.boxplot(data=d,x="Diameter")
```

```
[16]: <AxesSubplot:xlabel='Diameter'>
```



```
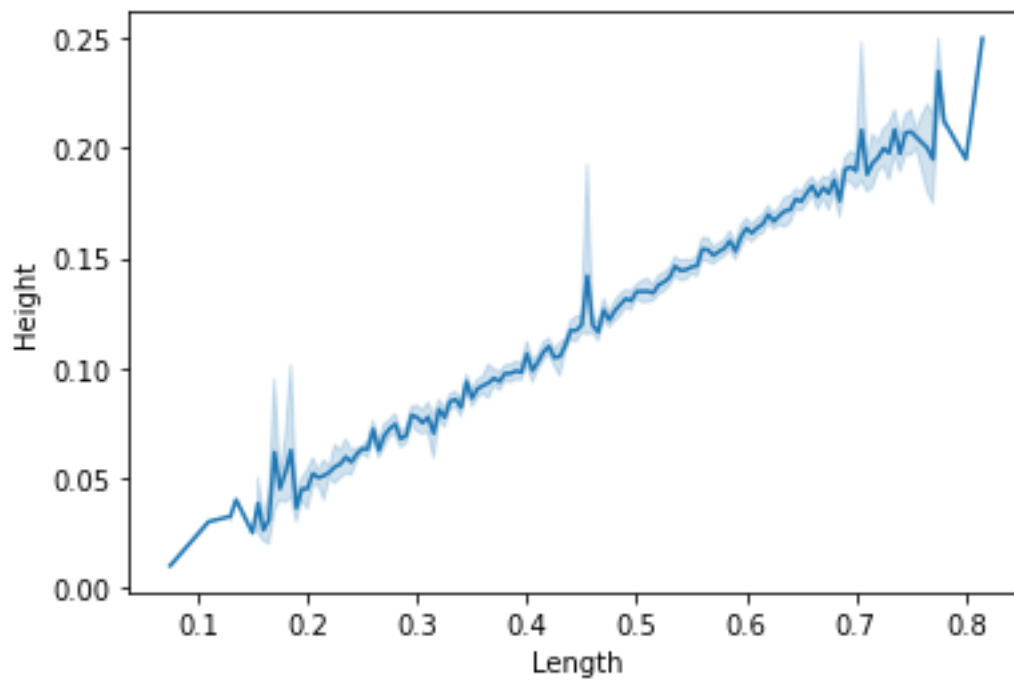[18]: sns.boxplot(data=d,x="Height")
```

```
[18]: <AxesSubplot:xlabel='Height'>
```

```
[20]: sns.lineplot(data=d,x="Length",y="Height")
```

[20]: <AxesSubplot:xlabel='Length', ylabel='Height'>

```
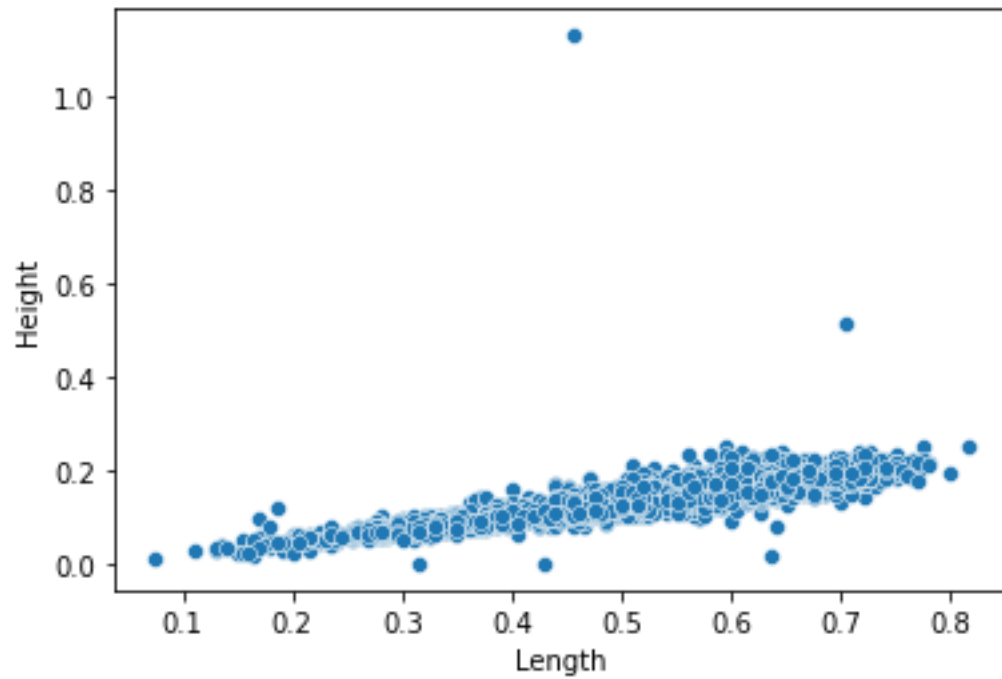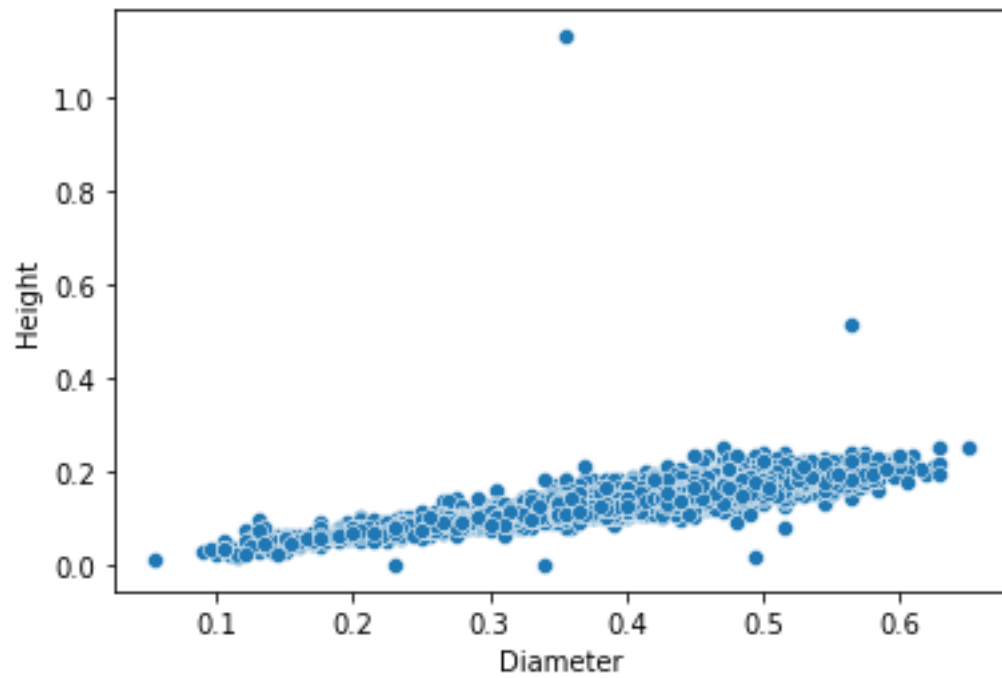[21]: sns.scatterplot(data=d,x="Length",y="Height")
```

```
[21]: <AxesSubplot:xlabel='Length', ylabel='Height'>
```



```
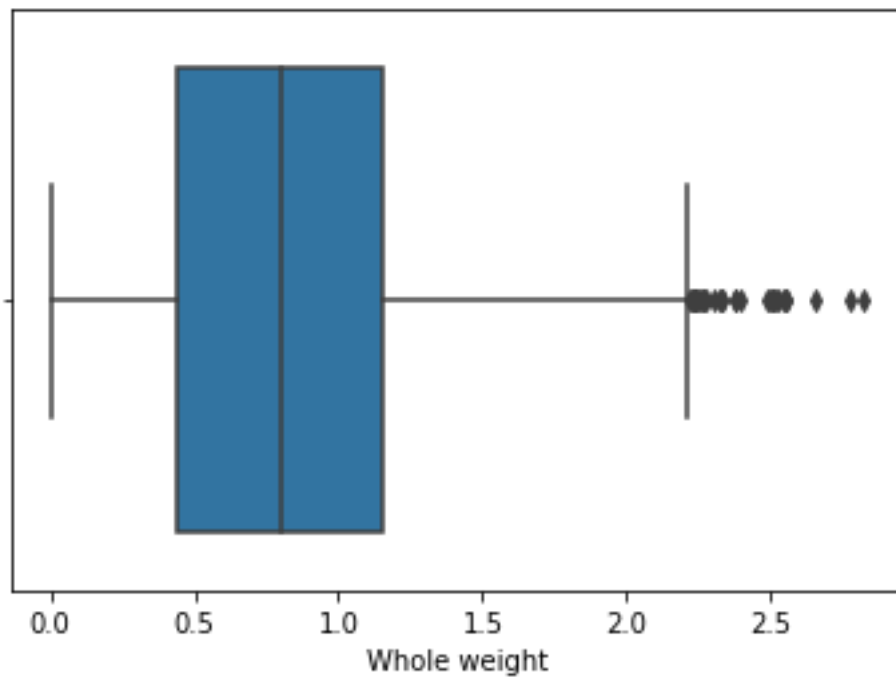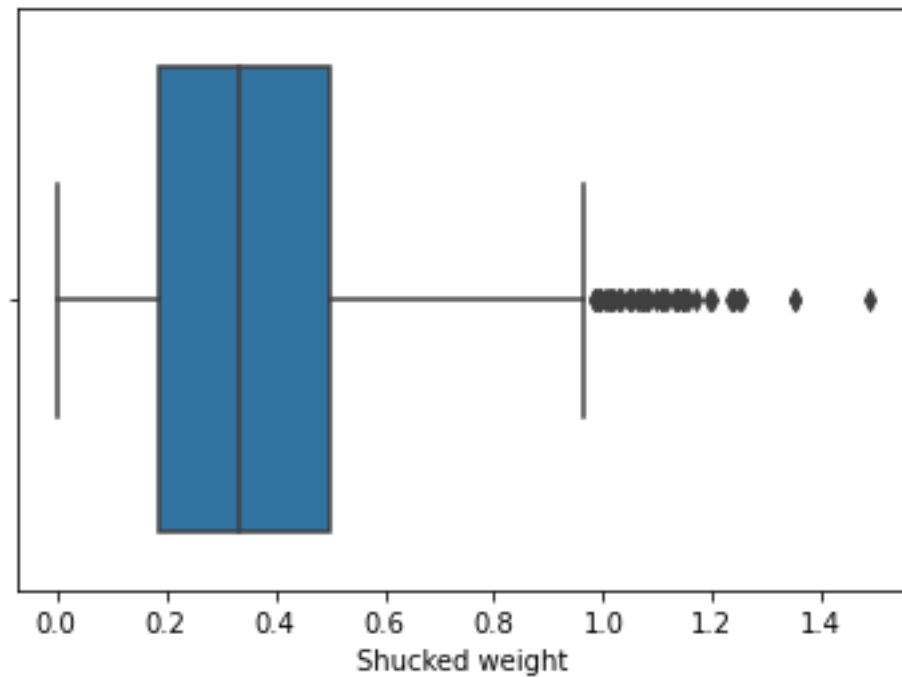[22]: sns.scatterplot(data=d,x="Diameter",y="Height")
```

```
[22]: <AxesSubplot:xlabel='Diameter', ylabel='Height'>
```

```
[23]: sns.boxplot(data=d,x="Whole weight")
```

```
[23]: <AxesSubplot:xlabel='Whole weight'>
```


10

```
[24]: sns.boxplot(data=d,x="Shucked weight")
```

```
[24]: <AxesSubplot:xlabel='Shucked weight'>
```



```
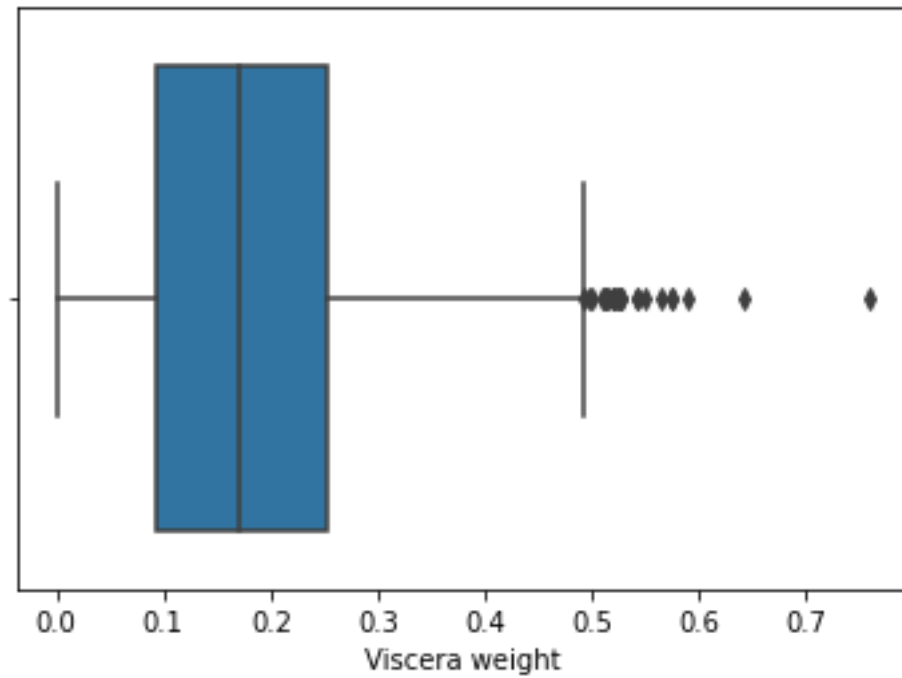[25]: d.columns
```

```
[25]: Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight',
'Shucked weight',
        'Viscera weight', 'Shell weight', 'Rings'],
       dtype='object')
```

```
[27]: sns.boxplot(data=d,x="Viscera weight")
```

```
[27]: <AxesSubplot:xlabel='Viscera weight'>
```

[28]: `sns.boxplot(data=d,x="Shell weight")`

[28]: `<AxesSubplot:xlabel='Shell weight'>`

12

```
[29]: sns.boxplot(data=d,x="Rings")
```

```
[29]: <AxesSubplot:xlabel='Rings'>
```



```
[31]: sns.scatterplot(data=d,x="Diameter",y="Whole weight")
```

```
[31]: <AxesSubplot:xlabel='Diameter', ylabel='Whole weight'>
```

```
[33]: sns.catplot(x="Shell weight",y="Sex",data=d,kind='box')
```

```
[33]: <seaborn.axisgrid.FacetGrid at 0x1ca496c8970>
```

2 Removing Outliners

```
[35]: data1=d[~(d["Height"]>0.4)]
```

```
[36]: data1=data1[~(data1["Length"]<0.15)]
```

```
[37]: data1=data1[~(data1["Shell weight"]>0.8)]
```

```
[38]: data1=data1[~(data1["Whole weight"]>2.5)]
```

```
[40]: data1=data1[~(data1["Shucked weight"]>1.2)]
```

```
[42]: data1.shape,d.shape
```

```
[42]: ((4148, 9), (4177, 9))
```

```
[43]: data1["Age"]=data1["Rings"]+1.5
```

```
[44]: data1.head()
```

```
[44]:   Sex Length Diameter Height Whole weight Shucked weight Viscera weight
\
0    M 0.455 0.365 0.095 0.5140      0.2245      0.1010
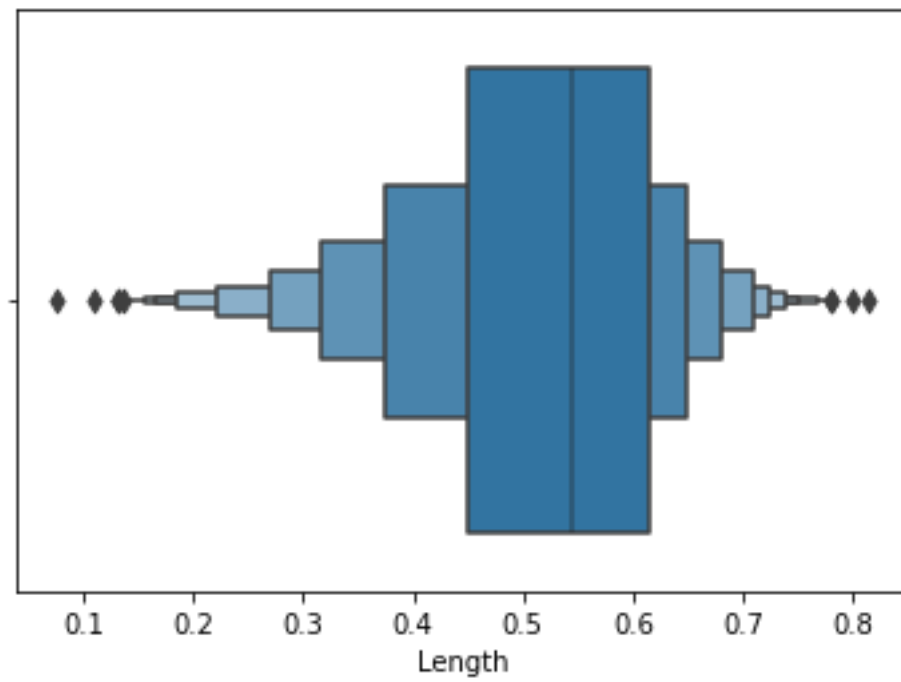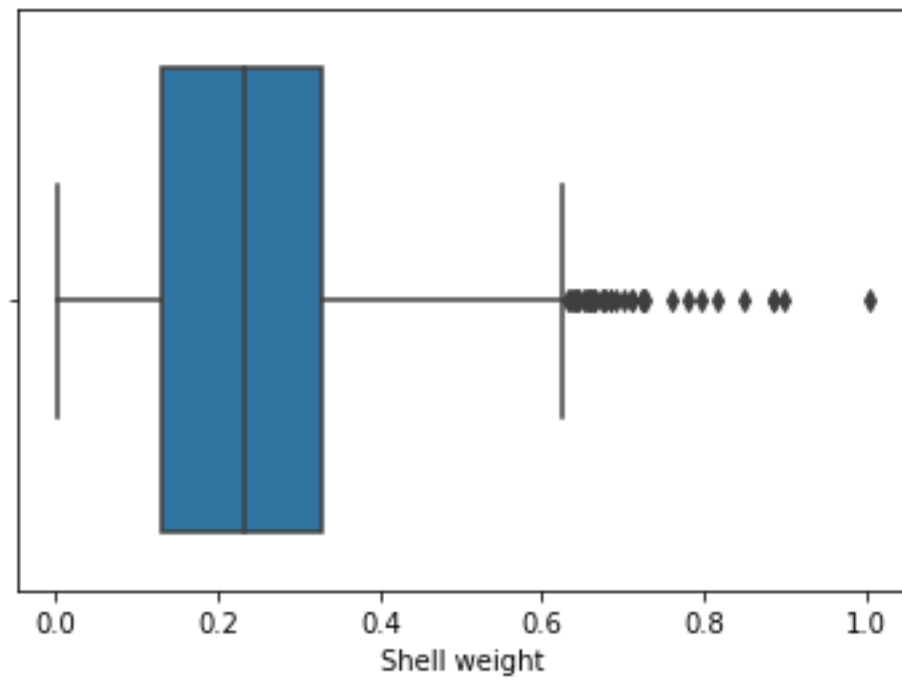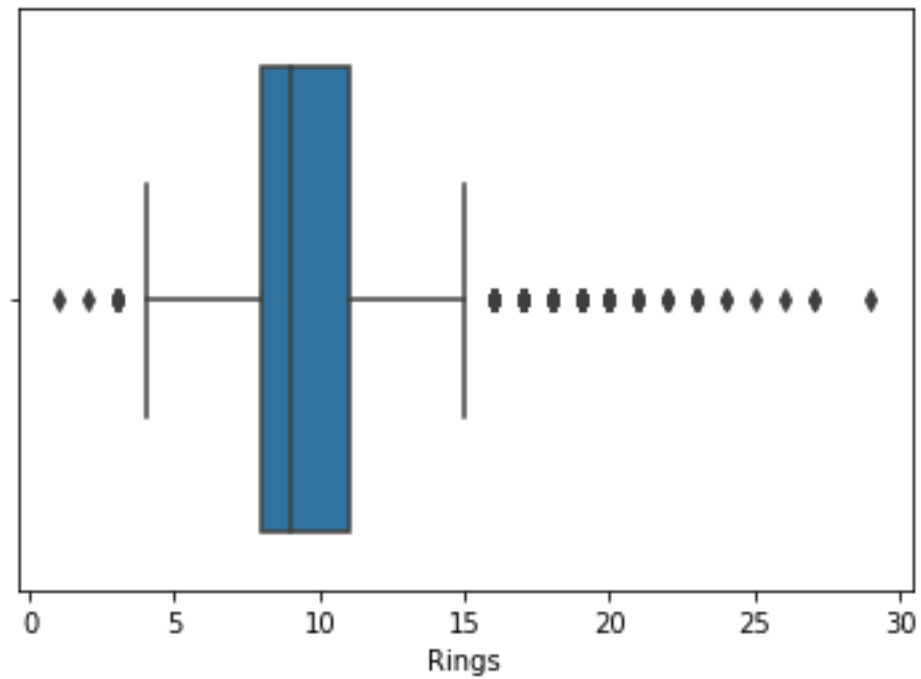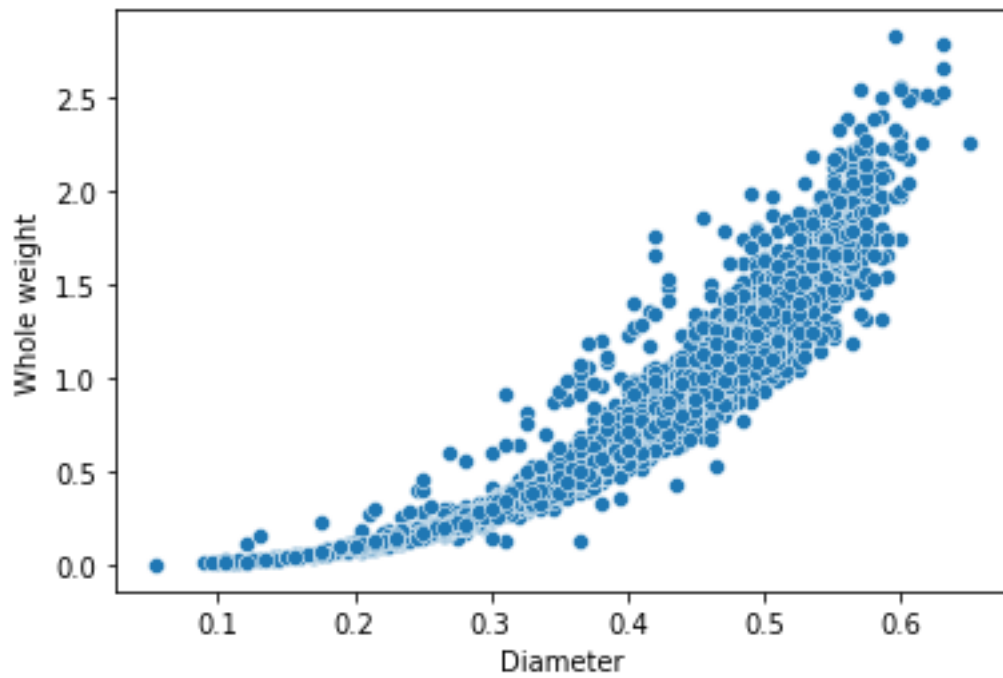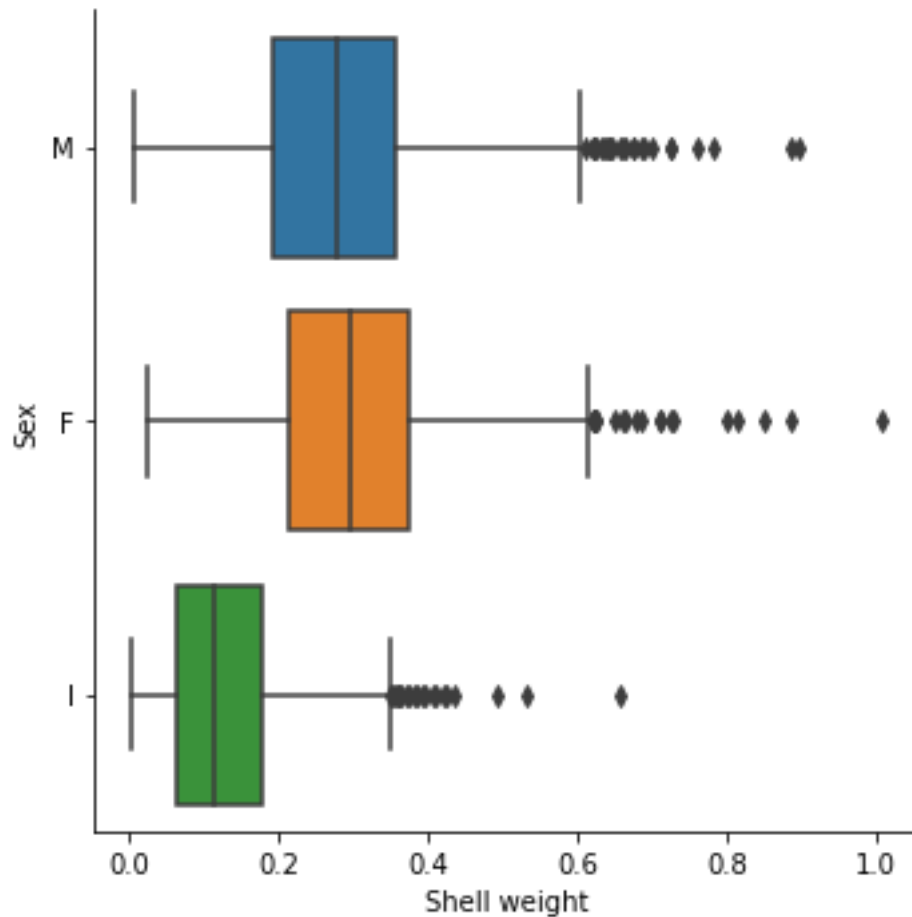
1    M 0.350 0.265 0.090 0.2255      0.0995      0.0485

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

      Shell weight Rings  Age

0        0.150    15 16.5

1        0.070    7 8.5

2        0.210    9 10.5

3        0.155    10 11.5

4        0.055    7 8.5
```

```
[45]: plt.figure(figsize=(15,9))
      sns.heatmap(data1.corr(),annot=True)
```

```
[45]: <AxesSubplot:>
```

16

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Age
Length	1	0.99	0.9	0.93	0.9	0.91	0.9	0.55	0.55
Diameter	0.99	1	0.9	0.93	0.9	0.9	0.91	0.57	0.57
Height	0.9	0.9	1	0.89	0.84	0.87	0.89	0.6	0.6
Whole weight	0.93	0.93	0.89	1	0.97	0.97	0.96	0.54	0.54
Shucked weight	0.9	0.9	0.84	0.97	1	0.93	0.89	0.42	0.42
Viscera weight	0.91	0.9	0.87	0.97	0.93	1	0.91	0.5	0.5
Shell weight	0.9	0.91	0.89	0.96	0.89	0.91	1	0.62	0.62
Rings	0.55	0.57	0.6	0.54	0.42	0.5	0.62	1	1
Age	0.55	0.57	0.6	0.54	0.42	0.5	0.62	1	1

```
[46]: q1=data1["Height"].quantile(0.25)
      q3=data1["Height"].quantile(0.75)
      iq=q3-q1
      data2=data1[~((data1["Height"]<(q1-1.5*iq))|(data1["Height"]>(q3+1.5*iq)))]
```

```
[47]: q1=data2["Length"].quantile(0.25)
      q3=data2["Length"].quantile(0.75)
      iq=q3-q1
      data2=data2[~((data2["Length"]<(q1-1.5*iq))|(data2["Length"]>(q3+1.5*iq)))]
```

```
[48]: q1=data2["Whole weight"].quantile(0.25)
      q3=data2["Whole weight"].quantile(0.75)
      iq=q3-q1
      data2=data2[~((data2["Whole weight"]<(q1-1.5*iq))|(data2["Whole weight"]>(q3+1.
       ↪5*iq)))]
      data2.shape
```

```
[48]: (4084, 10)
```

17

3 Split the data into dependent and independent variables.Check for Categorical columns and perform encoding

```
[49]: x=data1.drop(columns=["Age","Rings"])
      x["Sex"].replace({'M':2,'F':1,'I':0},inplace=True)
      y=data1["Age"]
```

4 Scale the independent

variables

```
[50]: from sklearn.preprocessing import StandardScaler
      sc=StandardScaler()
      x1=sc.fit_transform(x)
      x1
```

```
[50]: array([[ 1.15517188, -0.57962568, -0.43426571, …, -
              0.61179664, -0.72983642, -0.64298618],
             [ 1.15517188, -1.46650489, -1.4556039 , …, -
              1.19199725, -1.21854392, -1.23398399],
             [-0.05270635, 0.05385947, 0.12747029, …, -
              0.46326529,
              -0.35283349, -0.19973781],
             …,
             [ 1.15517188, 0.64511228, 0.6892063 , …, 0.78532642,
               1.00623881, 0.52423451],
             [-0.05270635, 0.856274 , 0.79134012, …, 0.81085524,
              0.75955788, 0.43558484],
            [ 1.15517188, 1.57422383, 1.50627685, …, 2.73480045,
              1.83471439, 1.90569191]])
```

5 Model Building

6 Linear Regression

```
[51]: from sklearn.model_selection import train_test_split
      from sklearn import metrics
      x_train,x_test,y_train,y_test=train_test_split(x1,y,tes
      t_size=0.
       ↪2,random_state=42)
      x_train.shape,x_test.shape
```

```
[51]: ((3318, 8), (830, 8))
```

```
[52]: from sklearn.linear_model import LinearRegression
      lr=LinearRegression()
      lr.fit(x_train,y_train)
      lr.score(x_test,y_test)
```

```
[52]: 0.5676481741929682
```

7   Lasso

```
[53]: from sklearn.linear_model import Lasso
      lr1=Lasso(alpha=0.001)
      lr1.fit(x_train,y_train)
      lr1.score(x_test,y_test)
```

```
[53]: 0.5672651558727646
```

8   Ridge

```
[54]: from sklearn.linear_model import Ridge
      r1=Ridge(alpha=0.01)
      r1.fit(x_train,y_train)
      r1.score(x_test,y_test)
```

```
[54]: 0.5676440857767044
```

9   Prediction

```
[55]: x_test[231],y_test[231]
```

```
[55]: (array([1.15517188, 0.56064759, 0.94454085, 0.28941484,
      0.66861919,
               0.59734142, 0.79679274, 0.797571 ]),
         15.5)
```

```
[56]: x_test[23],y_test[23]
```

```
[56]: (array([1.15517188, 0.89850634, 1.09774158, 0.68448704,
      1.10789396,
               0.61126624, 1.57872474, 1.57325563]),
         10.5)
```

```
[57]: lr.predict([x_test[231]])
```

```
[57]: array([13.06004481])
```

```
[58]: lr.predict([x_test[23]])
```

[58]: array([15.2756354])