Assignment 3

1 ASSIGNMENT 3 - Building The Regression model(Abalone Age Prediction)

1.Dataset - “abalone.csv”

2.Load the Dataset.

[1]:
```
#importing   libraries
import pandas as pd
#load the dataset
df=pd.read_csv("abalone.csv")
df
```

[1]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	\
0	M	0.455	0.365	0.095	0.5140	0.2245	
1	M	0.350	0.265	0.090	0.2255	0.0995	
2	F	0.530	0.420	0.135	0.6770	0.2565	
3	M	0.440	0.365	0.125	0.5160	0.2155	
4	I	0.330	0.255	0.080	0.2050	0.0895	
...	..	...	...	...	...	...	
4172	F	0.565	0.450	0.165	0.8870	0.3700	
4173	M	0.590	0.440	0.135	0.9660	0.4390	
4174	M	0.600	0.475	0.205	1.1760	0.5255	
4175	F	0.625	0.485	0.150	1.0945	0.5310	
4176	M	0.710	0.555	0.195	1.9485	0.9455	

	Viscera weight	Shell weight	Rings
0	0.1010	0.1500	15
1	0.0485	0.0700	7
2	0.1415	0.2100	9
3	0.1140	0.1550	10
4	0.0395	0.0550	7
...	...	...	...
4172	0.2390	0.2490	11
4173	0.2145	0.2605	10
4174	0.2875	0.3080	9
4175	0.2610	0.2960	10
4176	0.3765	0.4950	12

[4177 rows x 9 columns]

We are adding ”Age” column using ”Rings” data.

[2]:
```
import pandas as pd
df=pd.read_csv("abalone.csv")
Age=1.5+df.Rings
df["Age"]=Age
df=df.rename(columns = {"Whole weight":"Whole_weight","Shucked weight":
 ↪"Shucked_weight","Viscera weight": "Viscera_weight",
                         "Shell weight": "Shell_weight"})
df=df.drop(columns=["Rings"],axis=1)
df.head()
```

[2]:

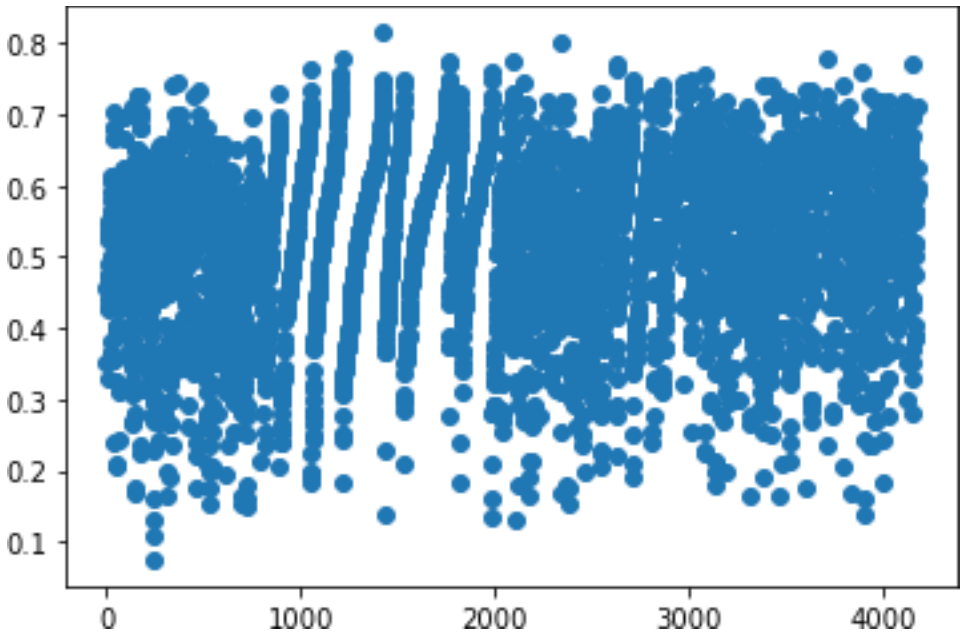
	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	\
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	

	Shell_weight	Age
0	0.150	16.5
1	0.070	8.5
2	0.210	10.5
3	0.155	11.5
4	0.055	8.5

3.Perform Below visualizations

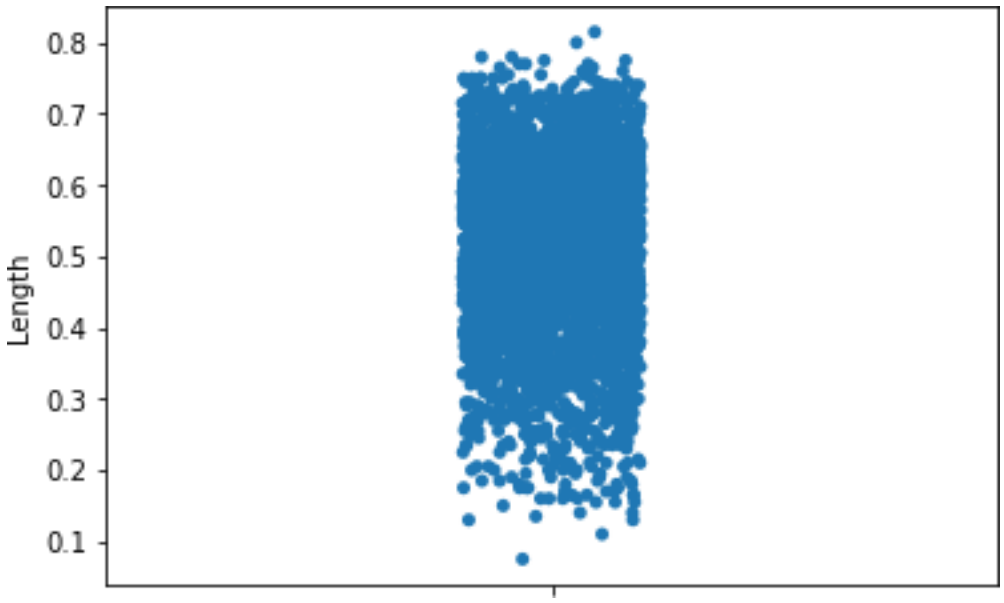
3.1.Univariate analysis

[4]:
```
#scatterplot
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
#load the dataset
df=pd.read_csv("abalone.csv")
plt.scatter(df.index,df["Length"])
plt.show()
```



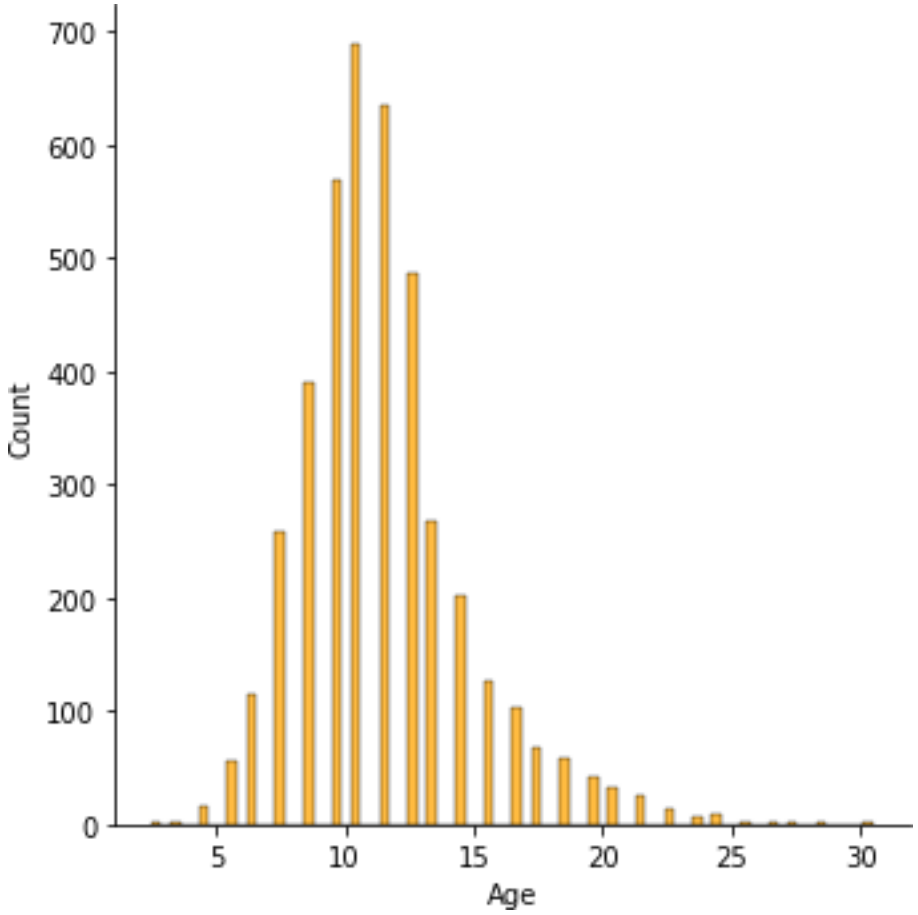
[5]:
```
#strip plot
sns.stripplot(y=df["Length"])
```

[5]: <AxesSubplot:ylabel='Length'>



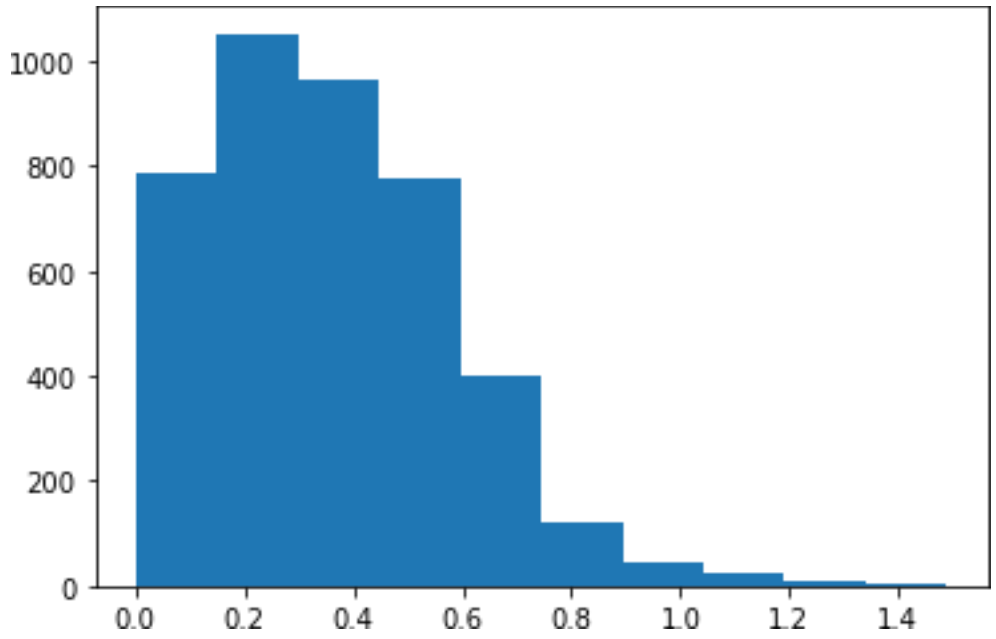
[7]:
```
import seaborn as sns
sns.displot(df["Age"], color='orange')
```

[7]: <seaborn.axisgrid.FacetGrid at 0x18b1f39a730>



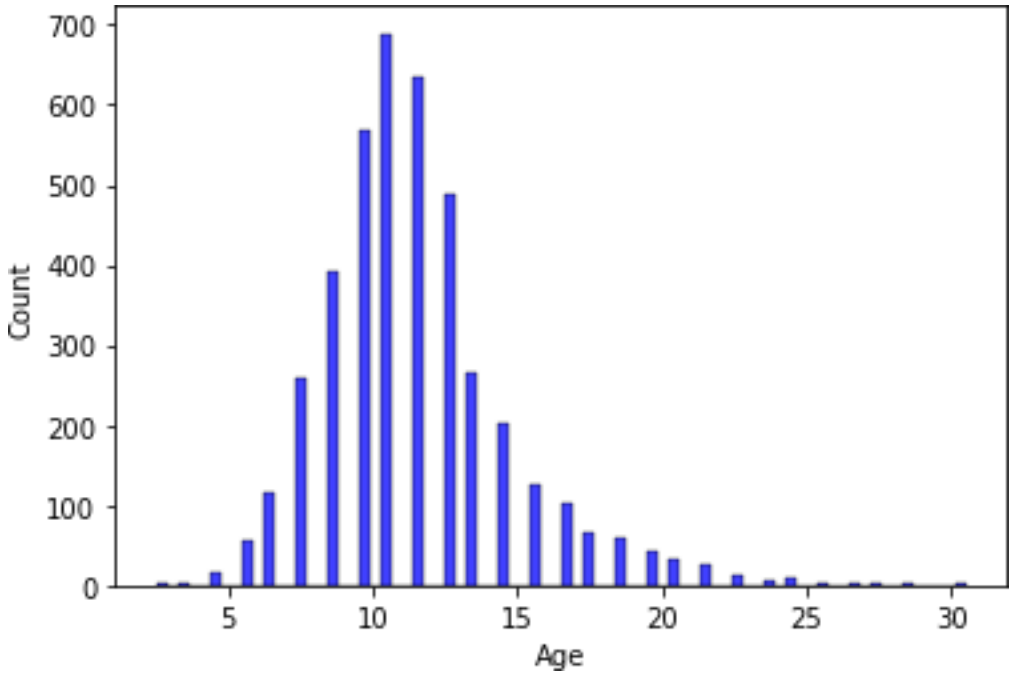
[10]:
```
#histogram
import matplotlib.pyplot as plt
plt.hist(df['Shucked weight'])
```

[10]: (array([786., 1052., 962., 775., 399., 123., 46., 24., 7.,
3.]),
array([1.0000e-03, 1.4970e-01, 2.9840e-01, 4.4710e-01, 5.9580e-01,
7.4450e-01, 8.9320e-01, 1.0419e+00, 1.1906e+00, 1.3393e+00,
1.4880e+00]),
<BarContainer object of 10 artists>)



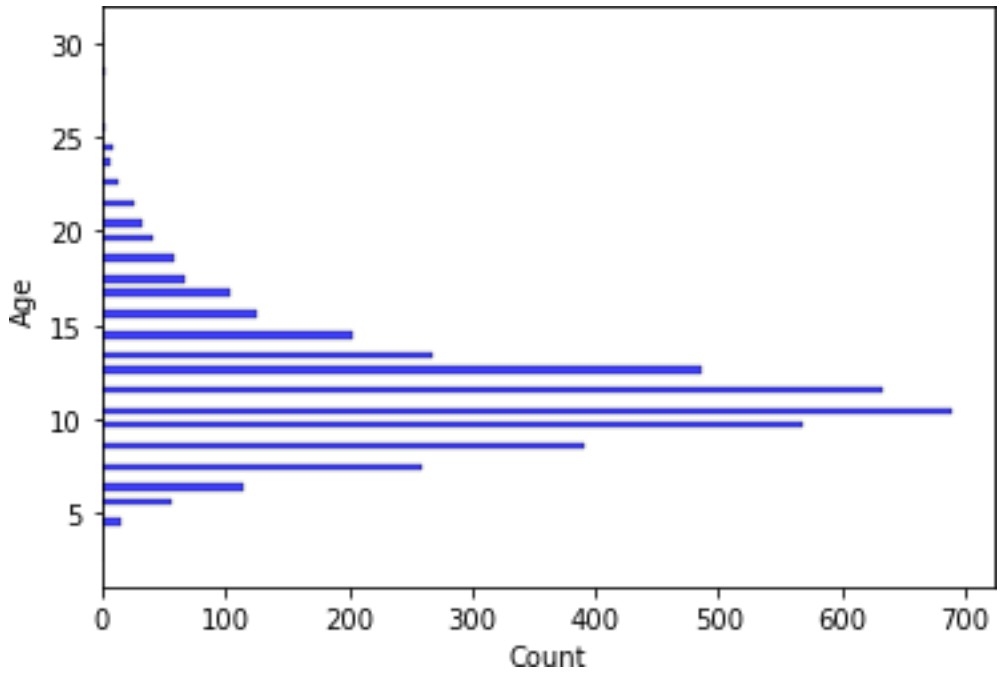
[11]:
```
sns.histplot(x=data.Age,color="blue")
```

[11]: <AxesSubplot:xlabel='Age', ylabel='Count'>



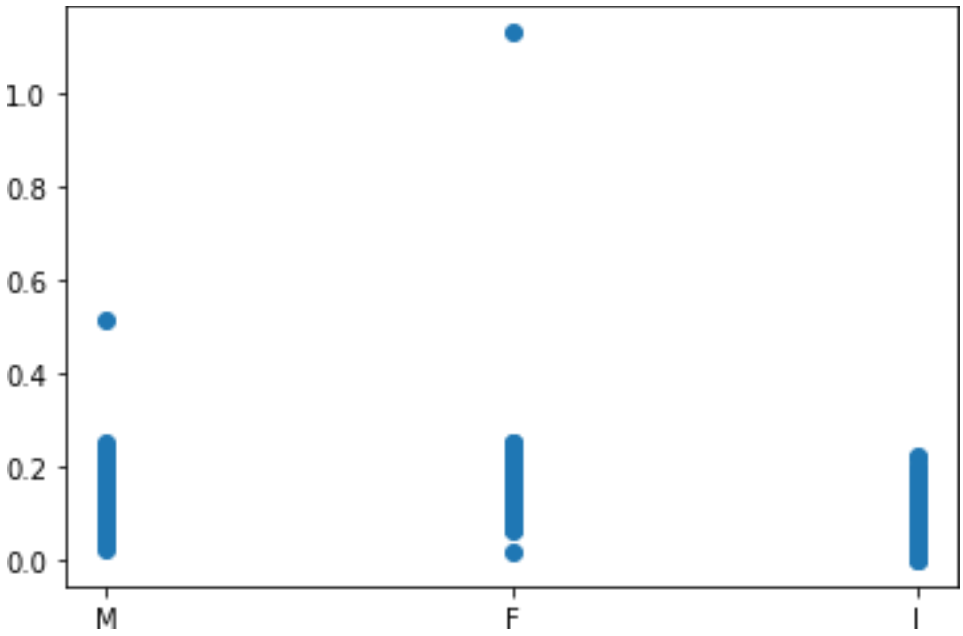
[8]:
```
sns.histplot(y=data.Age,color="Blue")
```

[8]: <AxesSubplot:xlabel='Count', ylabel='Age'>



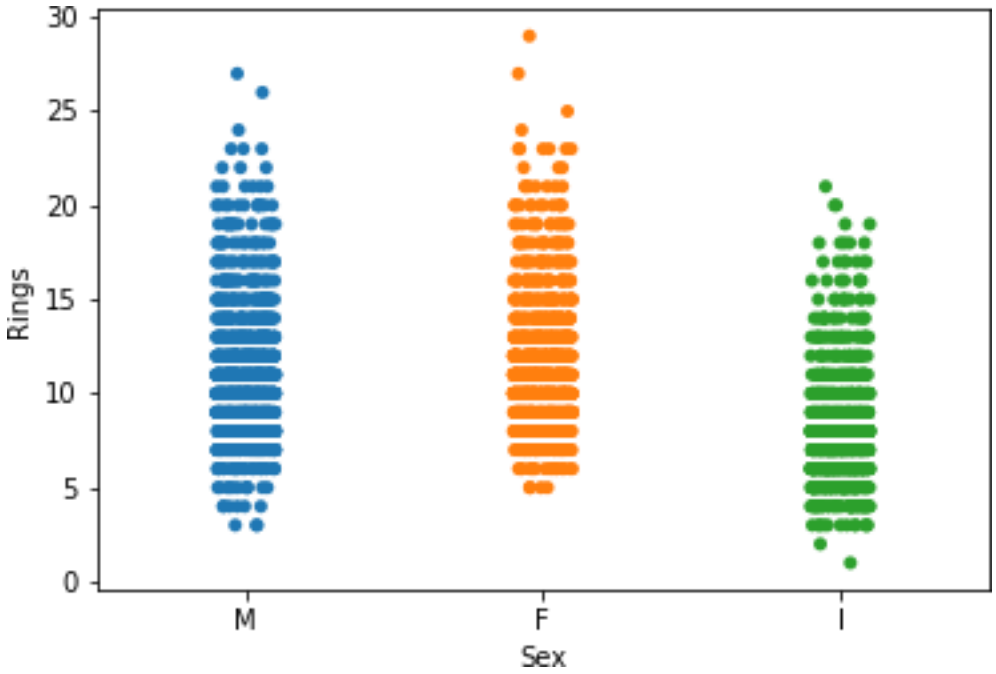
3.2.Bivariate Analysis

[12]:
```
#scatter plot
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
#load the dataset
df=pd.read_csv("abalone.csv")
plt.scatter(df.Sex,df.Height)
plt.show()
```



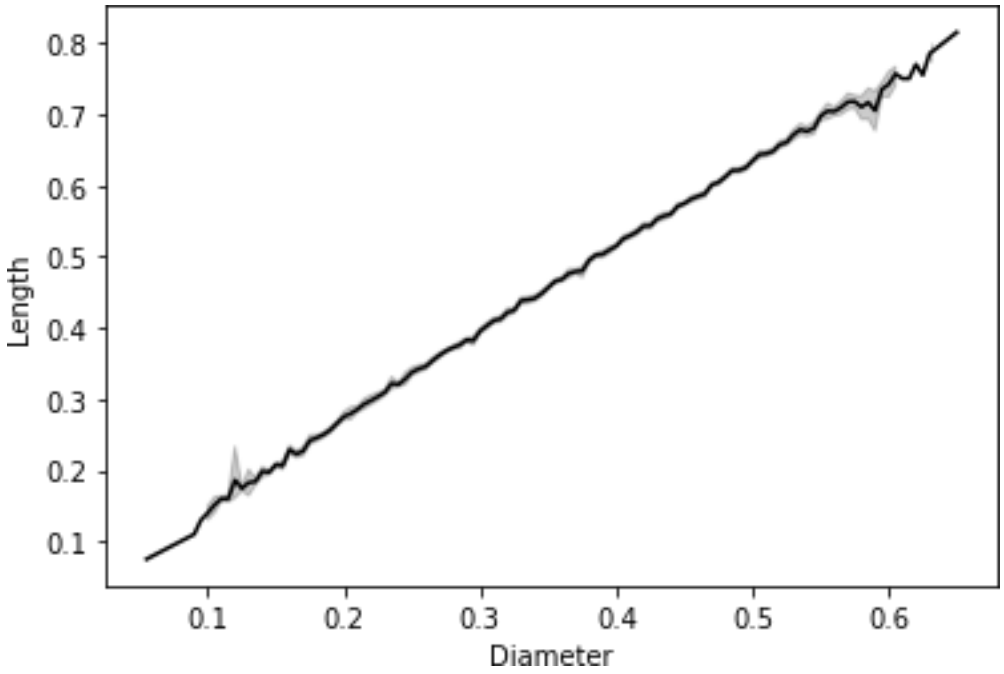
[13]:
```
#strip plot
sns.stripplot(x=df["Sex"],y=df["Rings"])
```

[13]: <AxesSubplot:xlabel='Sex', ylabel='Rings'>



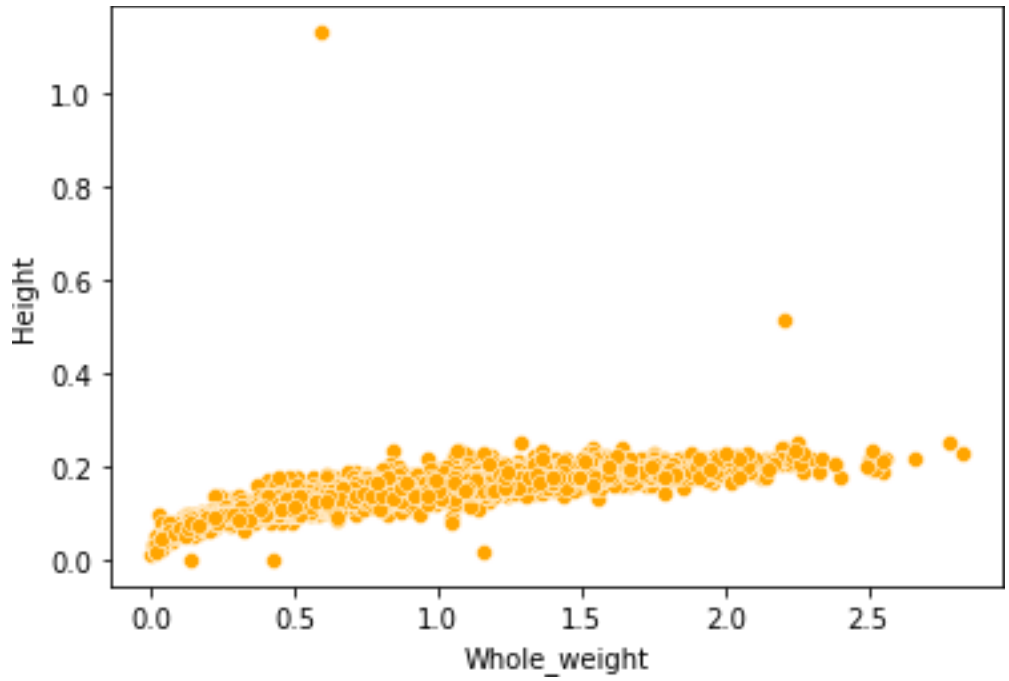
[12]:
```
sns.lineplot(x=data.Diameter,y=data.Length, color="black")
```

[12]: <AxesSubplot:xlabel='Diameter', ylabel='Length'>



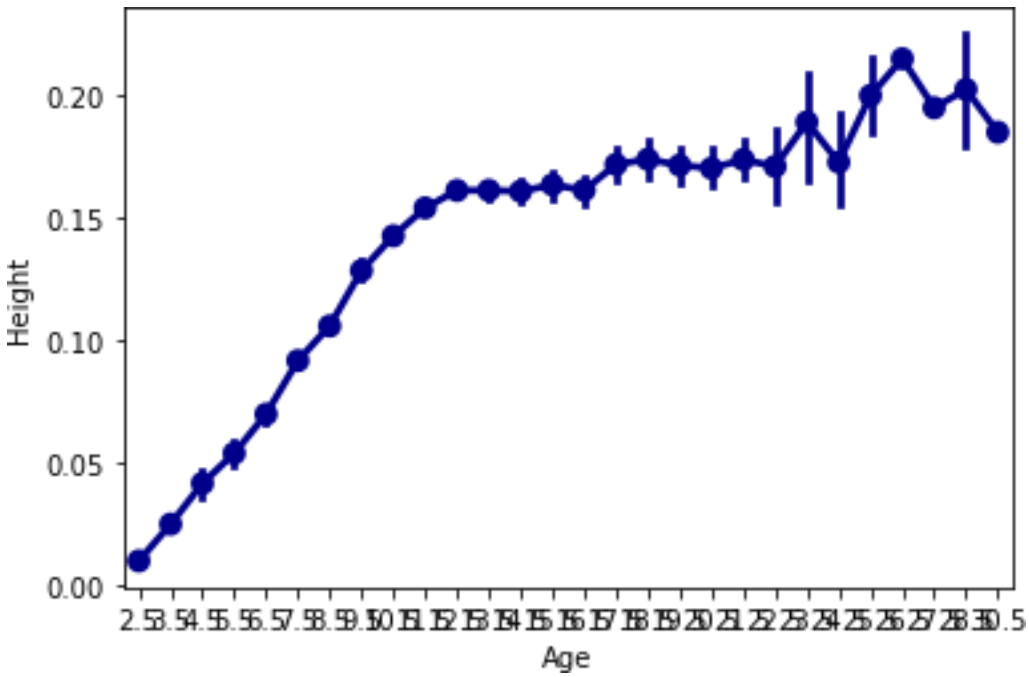
[22]:
```
sns.scatterplot(x=df.Whole_weight,y=df.Height,color="orange")
```

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



[24]:
```
sns.pointplot(x=df.Age, y=df.Height, color="Darkblue")
```

[24]: <AxesSubplot:xlabel='Age', ylabel='Height'>



3.3.Multivariate Analysis

[4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("darkgrid")
sns.set(font_scale=1.3)

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

[4]:

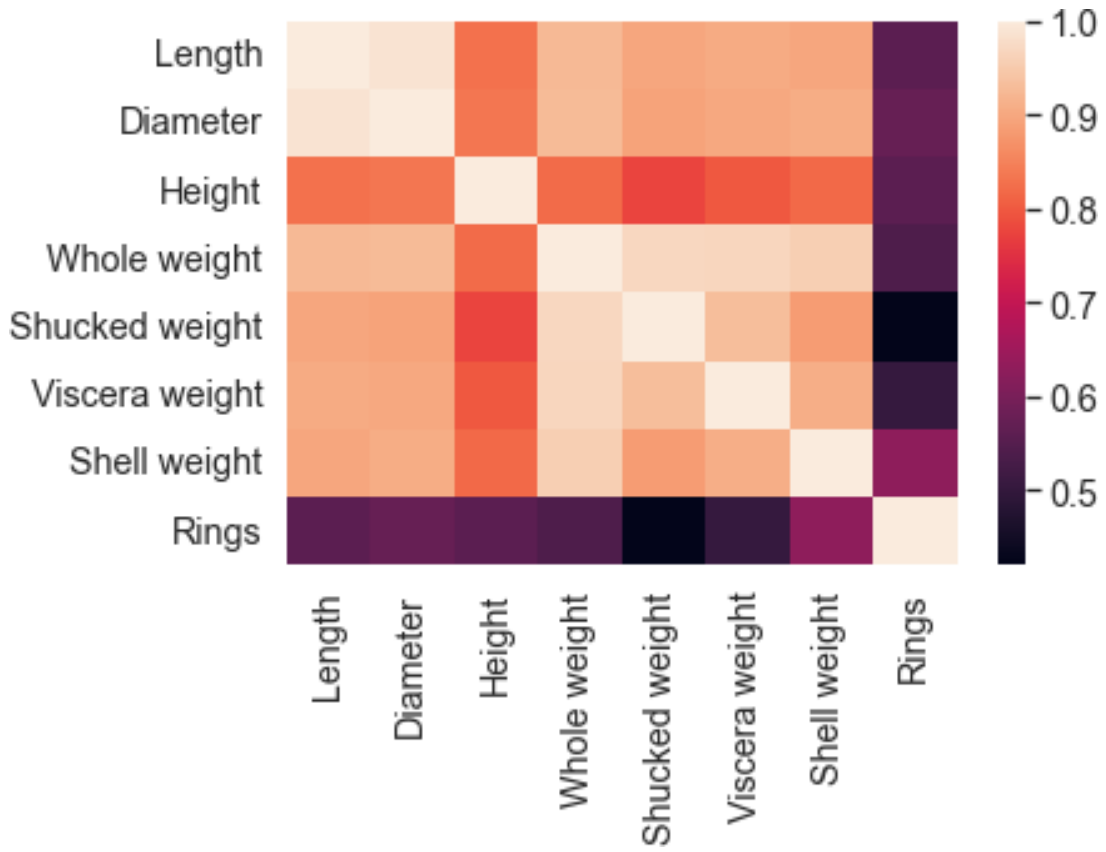
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	\
0	M	0.455	0.365	0.095	0.5140	0.2245	
1	M	0.350	0.265	0.090	0.2255	0.0995	
2	F	0.530	0.420	0.135	0.6770	0.2565	
3	M	0.440	0.365	0.125	0.5160	0.2155	
4	I	0.330	0.255	0.080	0.2050	0.0895	
...	..	...	...	...	...	...	
4172	F	0.565	0.450	0.165	0.8870	0.3700	
4173	M	0.590	0.440	0.135	0.9660	0.4390	
4174	M	0.600	0.475	0.205	1.1760	0.5255	
4175	F	0.625	0.485	0.150	1.0945	0.5310	
4176	M	0.710	0.555	0.195	1.9485	0.9455	

	Viscera weight	Shell weight	Rings
0	0.1010	0.1500	15
1	0.0485	0.0700	7
2	0.1415	0.2100	9
3	0.1140	0.1550	10
4	0.0395	0.0550	7
...	...	...	...
4172	0.2390	0.2490	11
4173	0.2145	0.2605	10
4174	0.2875	0.3080	9
4175	0.2610	0.2960	10
4176	0.3765	0.4950	12

[4177 rows x 9 columns]

[5]:
```
sns.heatmap(df.corr())
```

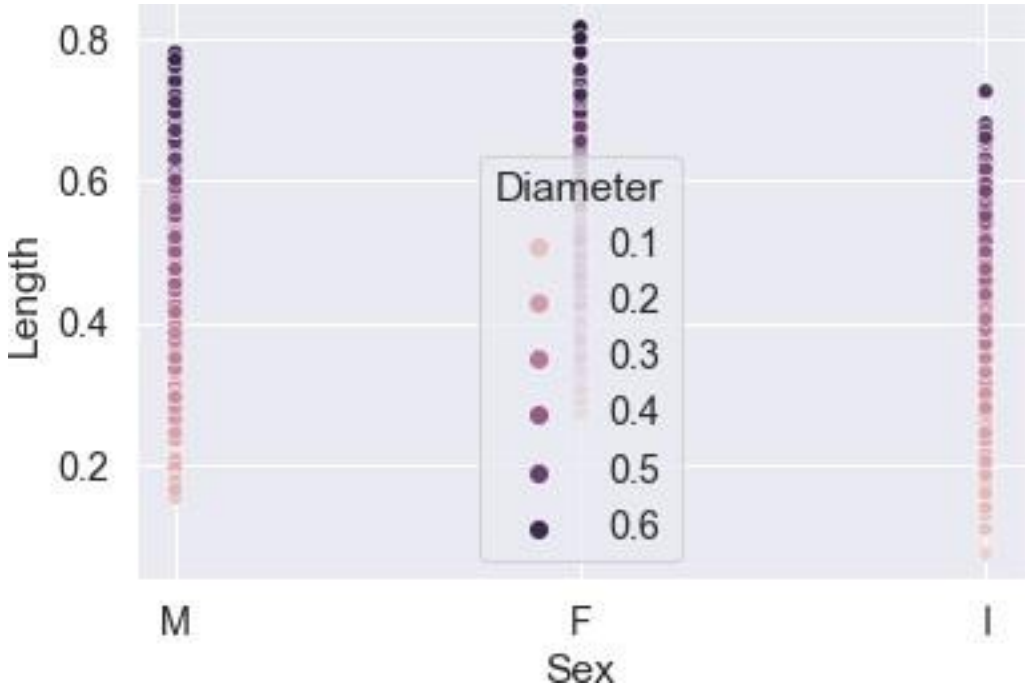
[5]: <AxesSubplot:>



[7]:
```
sns.scatterplot(df["Sex"],df["Length"],df["Diameter"])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y, hue. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

[7]: <AxesSubplot:xlabel='Sex', ylabel='Length'>



4. Perform descriptive statistics on the dataset.

[8]:
```
#load the dataset
import pandas as pd
data=pd.read_csv("abalone.csv")
data.head()
```

[8]:

	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

[9]:
```
df.mean()
```

C:\Users\janar vijay\AppData\Local\Temp\ipykernel_2496\3698961737.py:1:

```
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  df.mean()
```

[9]:
```
Length            0.523992
Diameter          0.407881
Height            0.139516
Whole weight      0.828742
Shucked weight    0.359367
Viscera weight    0.180594
Shell weight      0.238831
Rings             9.933684
dtype: float64
```

[10]:
```
df.describe()
```

[10]:

	Length	Diameter	Height	Whole weight	Shucked weight	\
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	
mean	0.523992	0.407881	0.139516	0.828742	0.359367	
std	0.120093	0.099240	0.041827	0.490389	0.221963	
min	0.075000	0.055000	0.000000	0.002000	0.001000	
25%	0.450000	0.350000	0.115000	0.441500	0.186000	
50%	0.545000	0.425000	0.140000	0.799500	0.336000	
75%	0.615000	0.480000	0.165000	1.153000	0.502000	
max	0.815000	0.650000	1.130000	2.825500	1.488000	

	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000
mean	0.180594	0.238831	9.933684
std	0.109614	0.139203	3.224169
min	0.000500	0.001500	1.000000
25%	0.093500	0.130000	8.000000
50%	0.171000	0.234000	9.000000
75%	0.253000	0.329000	11.000000
max	0.760000	1.005000	29.000000

[11]:
```
df.head(10)
```

[11]:

	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	
5	I	0.425	0.300	0.095	0.3515	0.1410	0.0775	
6	F	0.530	0.415	0.150	0.7775	0.2370	0.1415	
7	F	0.545	0.425	0.125	0.7680	0.2940	0.1495	

8	M	0.475	0.370	0.125	0.5095	0.2165	0.1125
9	F	0.550	0.440	0.150	0.8945	0.3145	0.1510

	Shell weight	Rings
0	0.150	15
1	0.070	7
2	0.210	9
3	0.155	10
4	0.055	7
5	0.120	8
6	0.330	20
7	0.260	16
8	0.165	9
9	0.320	19

[12]:
```
df.tail()
```

[12]:

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

	Viscera weight	Shell weight	Rings
4172	0.2390	0.2490	11
4173	0.2145	0.2605	10
4174	0.2875	0.3080	9
4175	0.2610	0.2960	10
4176	0.3765	0.4950	12

[13]:
```
df.tail(10)
```

[13]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	\
4167	M	0.500	0.380	0.125	0.5770	0.2690	
4168	F	0.515	0.400	0.125	0.6150	0.2865	
4169	M	0.520	0.385	0.165	0.7910	0.3750	
4170	M	0.550	0.430	0.130	0.8395	0.3155	
4171	M	0.560	0.430	0.155	0.8675	0.4000	
4172	F	0.565	0.450	0.165	0.8870	0.3700	
4173	M	0.590	0.440	0.135	0.9660	0.4390	
4174	M	0.600	0.475	0.205	1.1760	0.5255	
4175	F	0.625	0.485	0.150	1.0945	0.5310	
4176	M	0.710	0.555	0.195	1.9485	0.9455	

	Viscera weight	Shell weight	Rings
4167	0.1265	0.1535	9

4168	0.1230	0.1765	8
4169	0.1800	0.1815	10
4170	0.1955	0.2405	10
4171	0.1720	0.2290	8
4172	0.2390	0.2490	11
4173	0.2145	0.2605	10
4174	0.2875	0.3080	9
4175	0.2610	0.2960	10
4176	0.3765	0.4950	12

[14]:
```
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
```

[15]:
```
df.shape
```

[15]: (4177, 9)

[16]:
```
df.median()
```

```
C:\Users\janar vijay\AppData\Local\Temp\ipykernel_2496\530051474.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  df.median()
```

[16]:
Length	0.5450
Diameter	0.4250
Height	0.1400
Whole weight	0.7995
Shucked weight	0.3360
Viscera weight	0.1710
Shell weight	0.2340

```
Rings                9.0000
dtype: float64
```

[17]:
```
df.mode()
```

[17]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	\
0	M	0.550	0.45	0.15	0.2225	0.175	
1	NaN	0.625	NaN	NaN	NaN	NaN	

	Viscera weight	Shell weight	Rings
0	0.1715	0.275	9.0
1	NaN	NaN	NaN

5.Handle the Missing values.

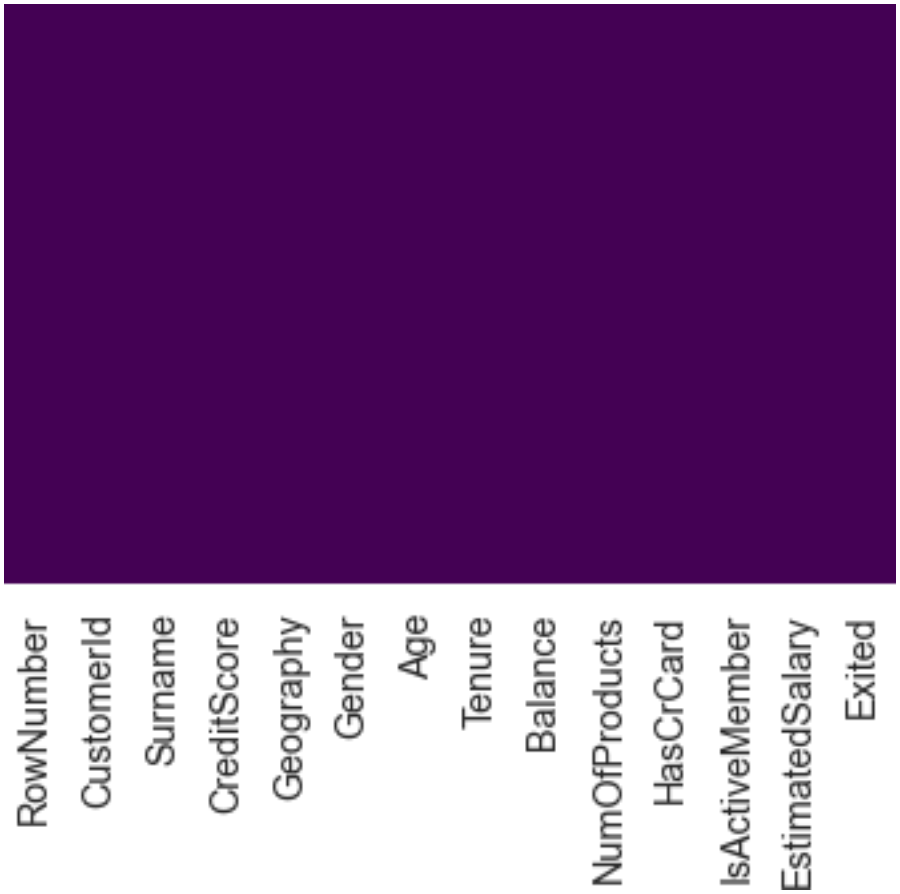
[18]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#read data
train=pd.read_csv("model.csv",sep=",")
```

[19]:
```
df.isnull().any()
```

[19] :
```
Sex                False
Length             False
Diameter           False
Height             False
Whole weight       False
Shucked weight     False
Viscera weight     False
Shell weight       False
Rings              False
dtype: bool
```

[20]:
```
#missing data in model.csv
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

[20] :
```
<AxesSubplot:>
```



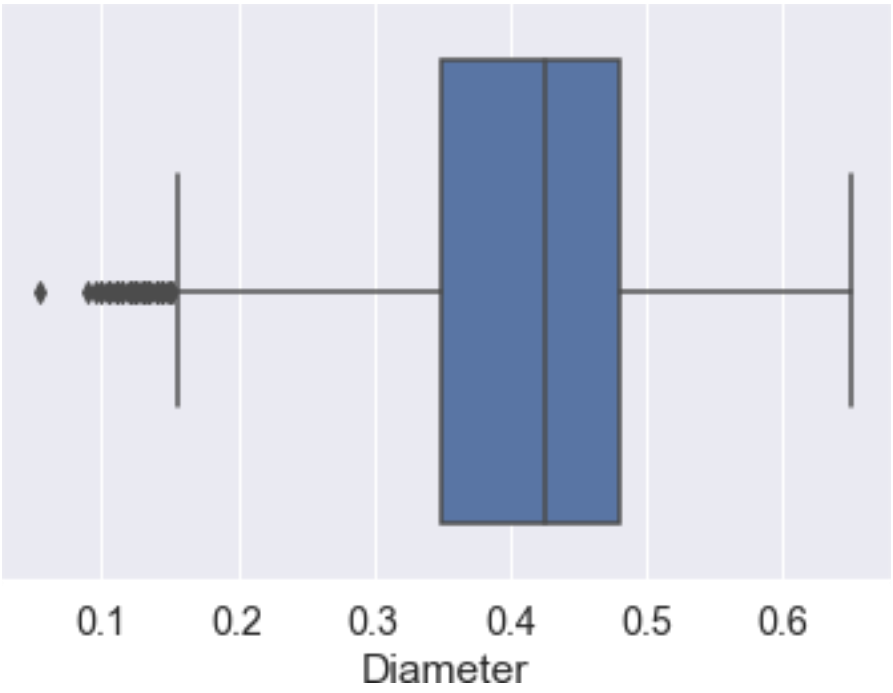
In our data no missing Values .so we have to take titanic data set to perform handling missing values

6. Find the outliers and replace the outliers

[26]:
```
#plotting outliers
sns.boxplot(df["Diameter"])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

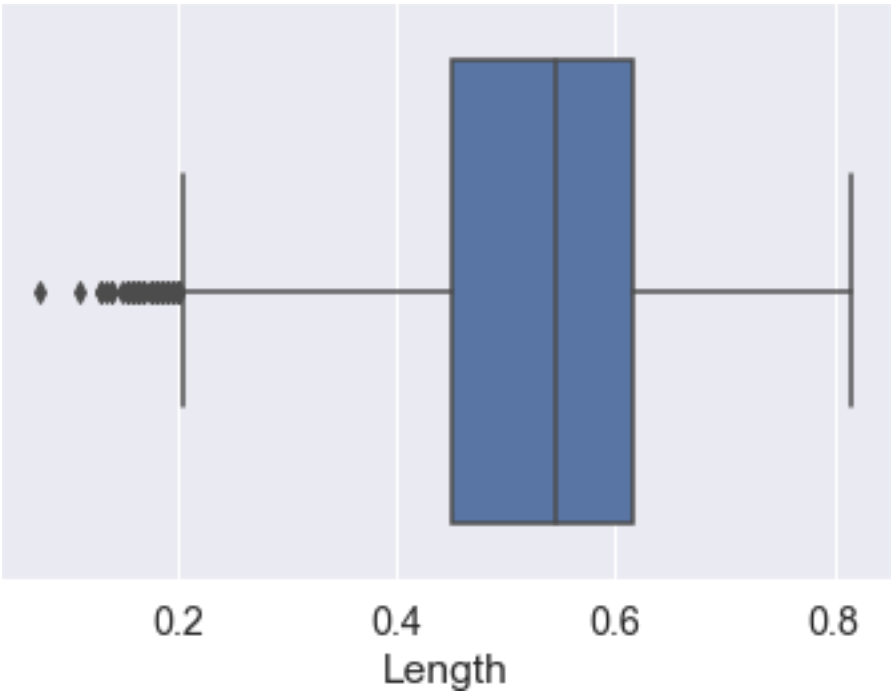
[26] : <AxesSubplot:xlabel='Diameter'>



[27]:
```
sns.boxplot(df["Length"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

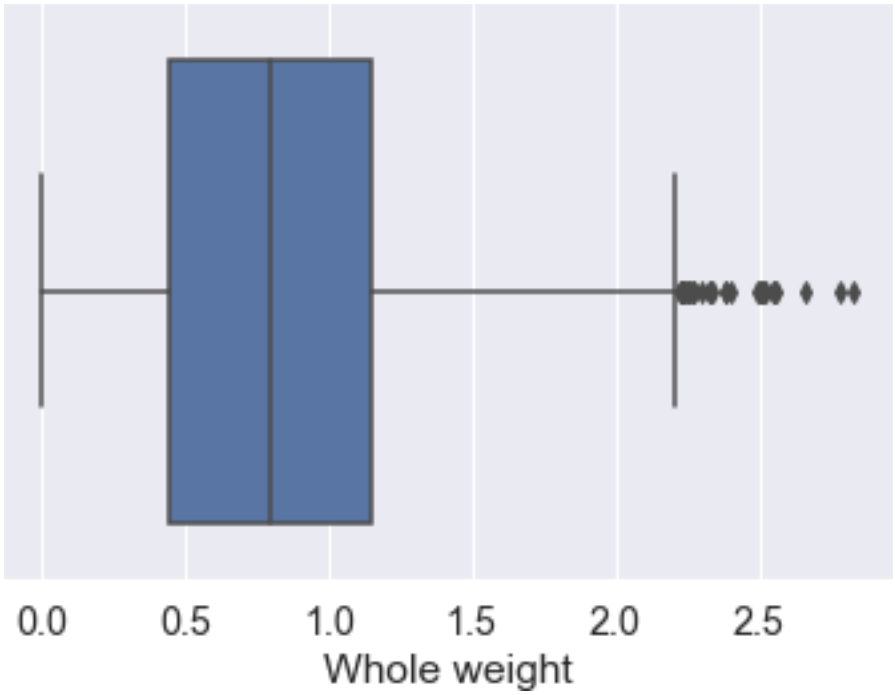
[27] : <AxesSubplot:xlabel='Length'>



[29]:
```
sns.boxplot(df["Whole weight"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

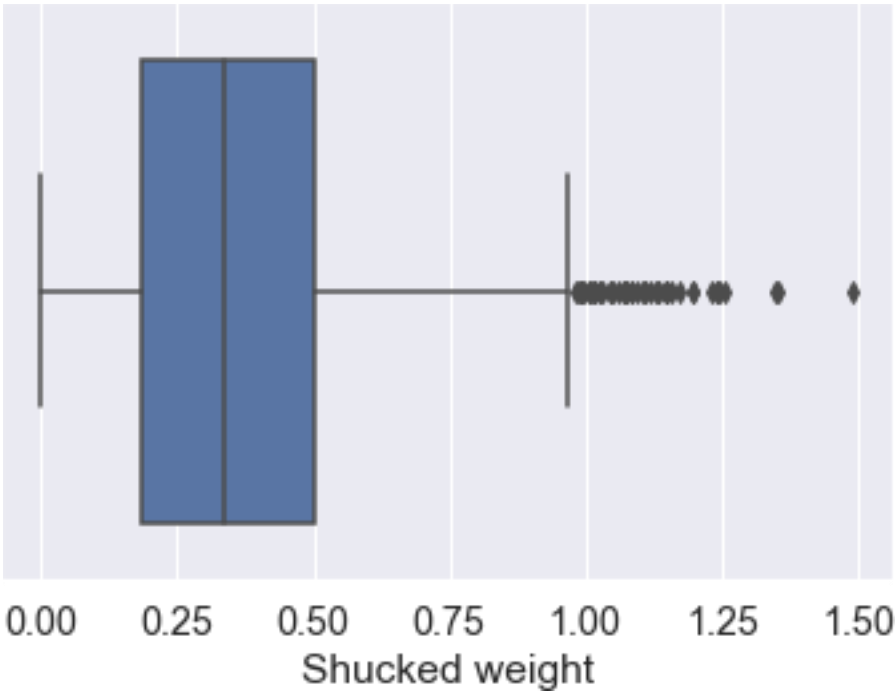
[29] : <AxesSubplot:xlabel='Whole weight'>



[30]:
```
sns.boxplot(df["Shucked weight"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

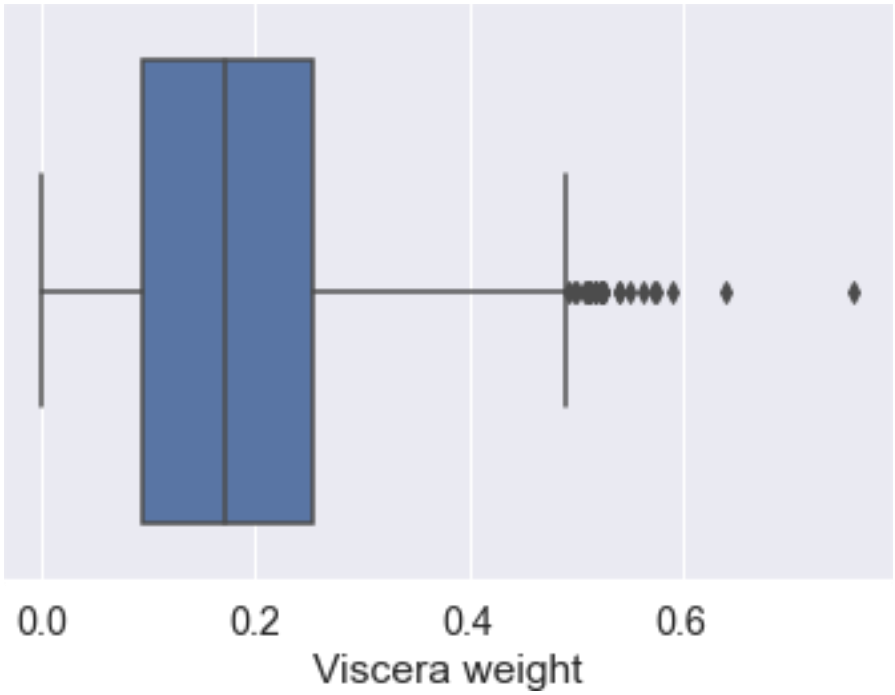
[30] : <AxesSubplot:xlabel='Shucked weight'>



[31]:
```
sns.boxplot(df["Viscera weight"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

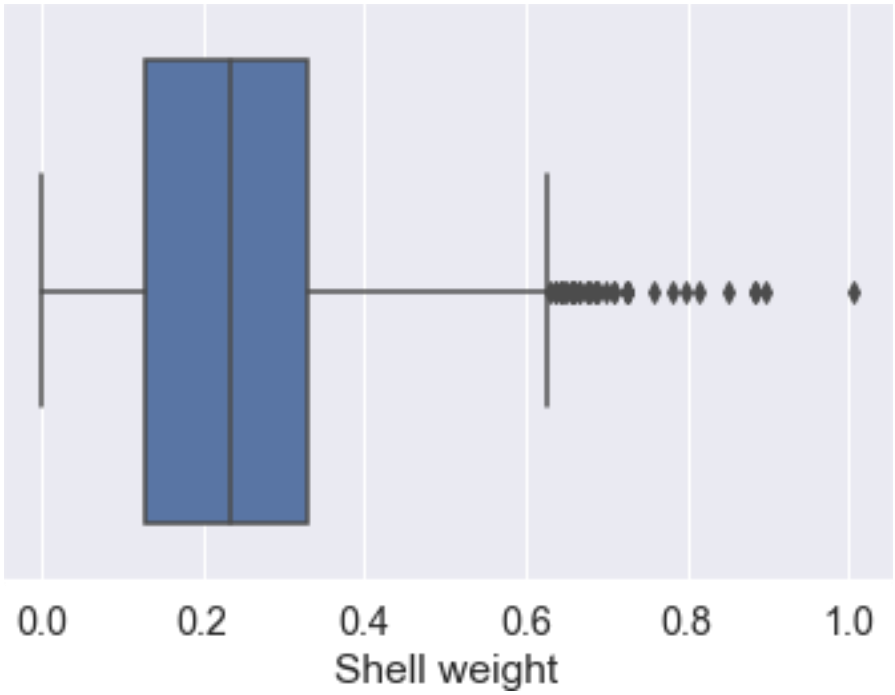
[31]: <AxesSubplot:xlabel='Viscera weight'>



[33]:
```
sns.boxplot(df["Shell weight"])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

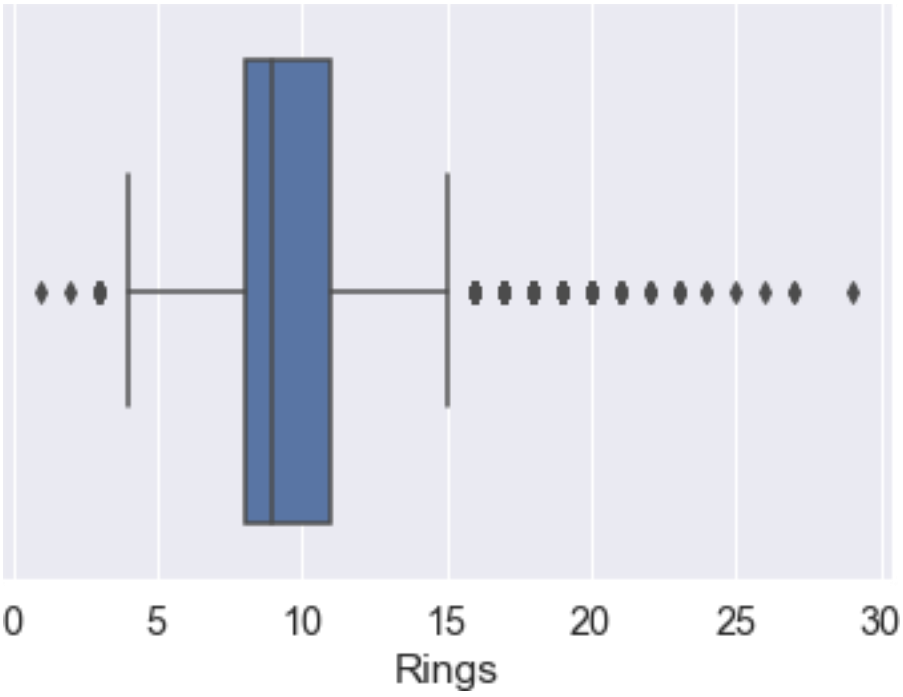
[33] : <AxesSubplot:xlabel='Shell weight'>



[34]:
```
sns.boxplot(df["Rings"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

[34] : <AxesSubplot:xlabel='Rings'>



[35]:
```
qnt=df.quantile(q=(0.75,0.25))

qnt
```

[35] :

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	\
0.75	0.615	0.48	0.165	1.1530	0.502	0.2530	
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	

	Shell weight	Rings
0.75	0.329	11.0
0.25	0.130	8.0

upper=q3+1.5*iqr lower=q1-1.5*iqr iqr=q3-q1

[37]:
```
iqr = qnt.loc[0.75]-qnt.loc[0.25] #iqr calculations
iqr
```

[37]:

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
```

[38]:
```
#lower extreme values
lower=qnt.loc[0.25] - 1.5*iqr
lower
```

[38]:
```
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
```

[39]:
```
#upper extreme values
upper=qnt.loc[0.75] + 1.5*iqr

upper
```

[39]:
```
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
```

[40]:
```
df.mean()
```

```
C:\Users\janar vijay\AppData\Local\Temp\ipykernel_2496\3698961737.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  df.mean()
```

[40]:
```
Length            0.523992
Diameter          0.407881
Height            0.139516
Whole weight      0.828742
Shucked weight    0.359367
Viscera weight    0.180594
Shell weight      0.238831
Rings             9.933684
```

dtype: float64

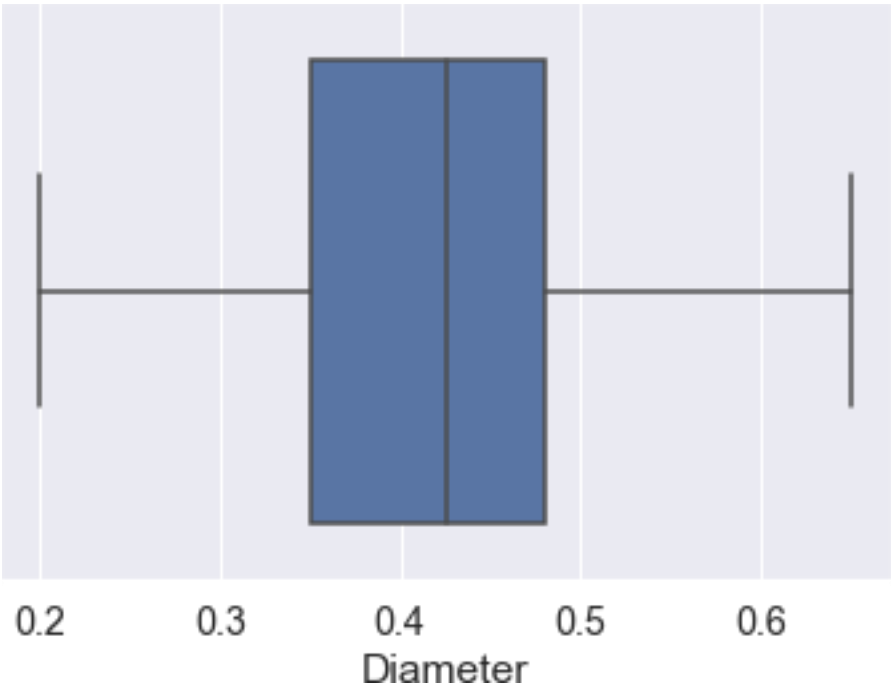
Replacing outlier

[78]:
```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#load the dataset
df=pd.read_csv("abalone.csv")
df["Diameter"]=np.where(df["Diameter"]<0.2,0.25,df["Diameter"])
df["Length"]=np.where(df["Length"]<0.25,0.30,df["Length"])
df["Height"]=np.where(df["Height"]<0,10.20,df["Height"])
df["Whole weight"]=np.where(df["Whole weight"]>2,1.5,df["Whole weight"])
df["Shucked weight"]=np.where(df["Shucked weight"]>0.9,0.5,df["Shucked weight"])
df["Viscera weight"]=np.where(df["Viscera weight"]>0.4,0.3,df["Viscera weight"])
df["Shell weight"]=np.where(df["Shell weight"]>0.6,0.5,df["Shell weight"])
df["Rings"]=np.where(df["Rings"]<5,6,df["Rings"])
df["Rings"]=np.where(df["Rings"]>15,12,df["Rings"])
```

[53]:
```
#remove outlier on the CreditScore column
sns.boxplot(df["Diameter"])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

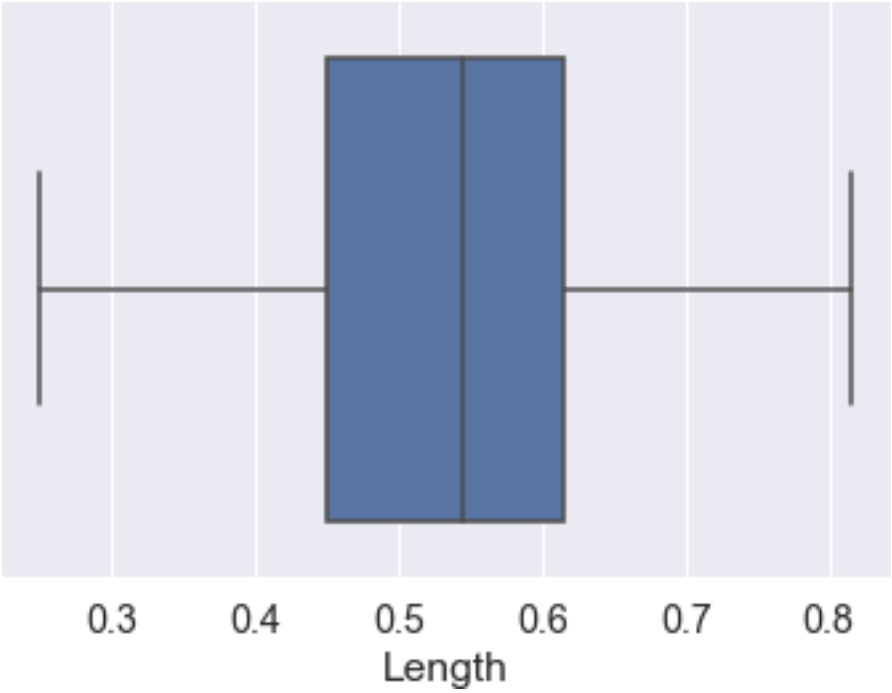
[53] : <AxesSubplot:xlabel='Diameter'>



[54]:
```
sns.boxplot(df["Length"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

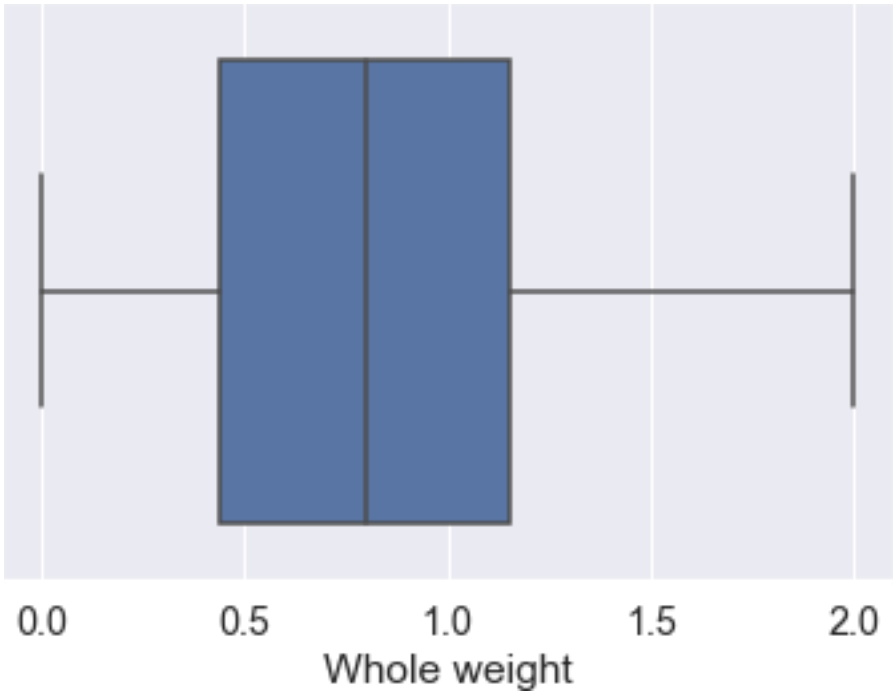
[54] : <AxesSubplot:xlabel='Length'>



[63]:
```
sns.boxplot(df["Whole weight"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

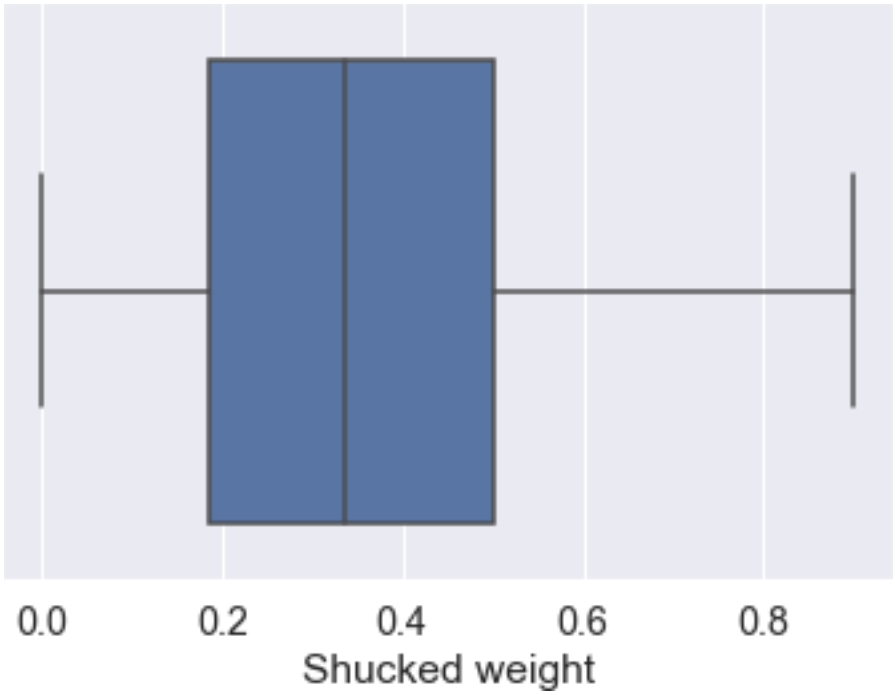
[63]: <AxesSubplot:xlabel='Whole weight'>



[69]:
```
sns.boxplot(df["Shucked weight"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

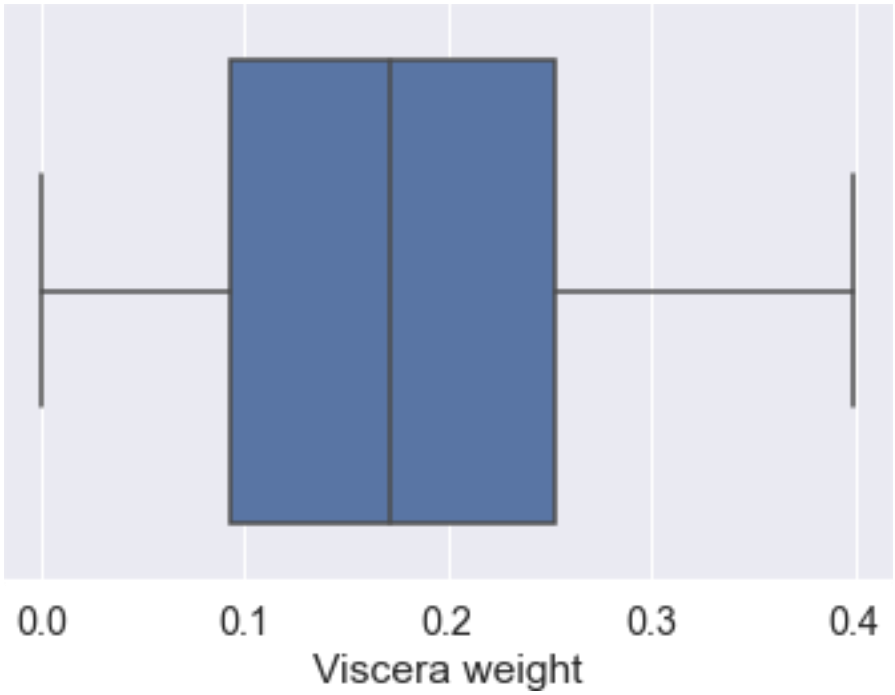
[69]: <AxesSubplot:xlabel='Shucked weight'>



[71]:
```
sns.boxplot(df["Viscera weight"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

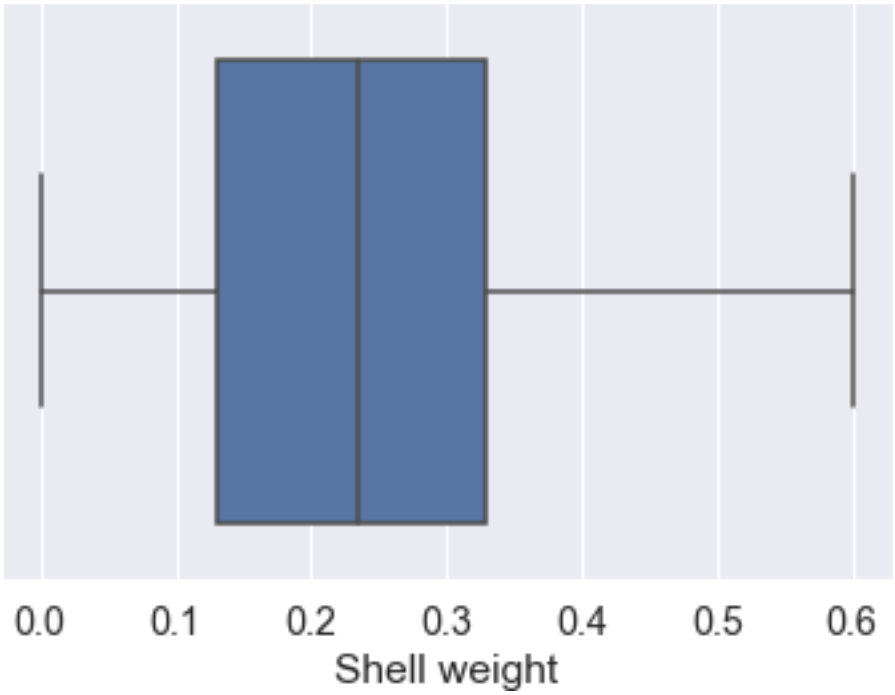
[71]: <AxesSubplot:xlabel='Viscera weight'>



[73]:
```
sns.boxplot(df["Shell weight"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

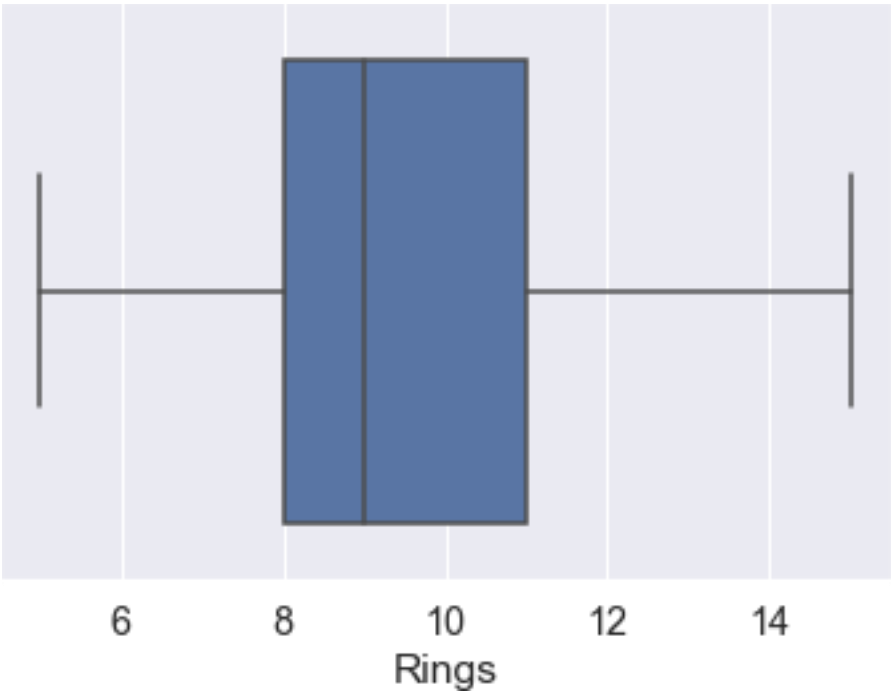
[73]: <AxesSubplot:xlabel='Shell weight'>



[79]:
```
sns.boxplot(df["Rings"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error  or
misinterpretation.
  warnings.warn(
```

[79]: <AxesSubplot:xlabel='Rings'>



7. Check for Categorical columns and perform encoding

[82]:
```
import pandas as pd
df=pd.read_csv("abalone.csv")
df.head()
```

[82]:

	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

encoding

[2]:
```
#manually handling categorincal data
import pandas as pd
df=pd.read_csv('abalone.csv')
df['Sex'].replace({'M':1,'F':2,'I':3},inplace=True)
df.head()
```

[2]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	\
0	1	0.455	0.365	0.095	0.5140	0.2245	
1	1	0.350	0.265	0.090	0.2255	0.0995	
2	2	0.530	0.420	0.135	0.6770	0.2565	
3	1	0.440	0.365	0.125	0.5160	0.2155	
4	3	0.330	0.255	0.080	0.2050	0.0895	

	Viscera weight	Shell weight	Rings
0	0.1010	0.150	15
1	0.0485	0.070	7
2	0.1415	0.210	9
3	0.1140	0.155	10
4	0.0395	0.055	7

[5]:
```
#dummy variable  function
import pandas as pd
df_main=pd.get_dummies(df,columns=['Sex'])
df_main
```

[5]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	\
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	
...	...	...	...	...	...	...	
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	

	Shell weight	Rings	Sex_1	Sex_2	Sex_3
0	0.1500	15	1	0	0
1	0.0700	7	1	0	0
2	0.2100	9	0	1	0
3	0.1550	10	1	0	0
4	0.0550	7	0	0	1
...	...	...	...	...	...
4172	0.2490	11	0	1	0

4173	0.2605	10	1	0	0
4174	0.3080	9	1	0	0
4175	0.2960	10	0	1	0
4176	0.4950	12	1	0	0

[4177 rows x 11 columns]

8. Split the data into dependent and independent variables.

[16]:
```
#target variable or  dependent variable.
X = df.iloc[:, 1:7]
X
```

[16]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
0	0.455	0.365	0.095	0.5140	0.2245	0.1010
1	0.350	0.265	0.090	0.2255	0.0995	0.0485
2	0.530	0.420	0.135	0.6770	0.2565	0.1415
3	0.440	0.365	0.125	0.5160	0.2155	0.1140
4	0.330	0.255	0.080	0.2050	0.0895	0.0395
...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765

[4177 rows x 6 columns]

[15]:
```
#independent variables
y = df.iloc[:, -1]
y
```

[15]:
```
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
```

9. Scale the independent variables

```
from sklearn.preprocessing import scale
```

[8]:

[25]:
```
import numpy as np
import pandas as pd
df=pd.read_csv("abalone.csv")
y = df.iloc[:, -1]
y
```

[25]:
```
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
```

10.Spilt the data into training and testing

```
from sklearn.model_selection import train_test_split
```

[11]:

[12]:
```
X = df.iloc[:, 1:7]
X
```

[12]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
0	0.455	0.365	0.095	0.5140	0.2245	0.1010
1	0.350	0.265	0.090	0.2255	0.0995	0.0485
2	0.530	0.420	0.135	0.6770	0.2565	0.1415
3	0.440	0.365	0.125	0.5160	0.2155	0.1140
4	0.330	0.255	0.080	0.2050	0.0895	0.0395
...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765

[4177 rows x 6 columns]

[13]:
```
y = df.iloc[:, -1]
y
```

[13]:
```
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
```

[14]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state
↪=42)
```

[15]:
```
X_train
```

[15]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
3823	0.615	0.455	0.135	1.0590	0.4735	0.2630
3956	0.515	0.395	0.140	0.6860	0.2810	0.1255
3623	0.660	0.530	0.175	1.5830	0.7395	0.3505
0	0.455	0.365	0.095	0.5140	0.2245	0.1010
2183	0.495	0.400	0.155	0.8085	0.2345	0.1155
...	...	...	...	...	...	...
3444	0.490	0.400	0.115	0.5690	0.2560	0.1325
466	0.670	0.550	0.190	1.3905	0.5425	0.3035
3092	0.510	0.395	0.125	0.5805	0.2440	0.1335
3772	0.575	0.465	0.120	1.0535	0.5160	0.2185
860	0.595	0.475	0.160	1.1405	0.5470	0.2310

[3132 rows x 6 columns]

[16]:
```
y_train
```

[16]:
```
3823     9
3956    12
3623    10
0       15
2183     6
        ..
3444     9
466     12
3092    11
3772     9
860      6
Name: Rings, Length: 3132, dtype: int64
```

[18]:
```
print(X_train.shape, X_test.shape)
```

```
(3132, 6) (1045, 6)
```

[26]:
```
print(y_test,y_test)
```

```
866      9
1483     8
599     16
1702     9
670     14
        ..
532     12
3417     9
1505     8
2245     9
2428    10
Name: Rings, Length: 1045, dtype: int64 866      9
1483     8
599     16
1702     9
670     14
        ..
532     12
3417     9
1505     8
2245     9
2428    10
Name: Rings, Length: 1045, dtype: int64
```

```
print(y_test.shape,y_test.shape)
```

```
(1045,) (1045,)
```

[27]:

11.Build the Model

```
from sklearn.linear_model import LinearRegression
```

```
model=LinearRegression()
```

[19]:

[20]:

[21]:
```
model.fit(X_train,y_train)
```

[21]: LinearRegression()

12.Train the Model

[22]:
```
y_predict_train = model.predict(X_train)
y_predict_train
```

[22]:
```
array([ 9.75888828, 10.45379472, 10.83692259, ...,  9.62903068,
        9.21152746, 10.09516371])
```

13. Test the Model

[23]:
```
y_predict = model.predict(X_test)
y_predict
```

[23]:
```
array([11.5478407 ,  9.93166184, 14.09825921, ..., 12.19440346,
       10.29279231,  9.33037939])
```

14.Measure the performance using Metrics

[24]:
```
from sklearn.metrics import mean_squared_error
import math
print(mean_squared_error(y_test, y_predict))
print(math.sqrt(mean_squared_error(y_test, y_predict)))
```

```
4.862459933051861
2.2050986220692854
```