

**NAME : FARZANA.J REG NO :512219205002 PROJECT : RETAIL STORE STOCK INVENTORY
ANALYTICS ASSIGNMENT: 04 1. DOWNLOADING DATASET**

Dataset Link : [Abalone](#)

2. LOADING THE DATASET

```
import numpy as np import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline import
seaborn as sns
```

In []:

```
df = pd.read_csv('/content/drive/MyDrive/abalone.csv')
```

In []:

```
df.head()
```

In []:

```
Out[ ]:
```

	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

3.PERFORMING VISUALIZATION'S

(i)Univariate Analysis

(ii)Bi-Variate Analysis

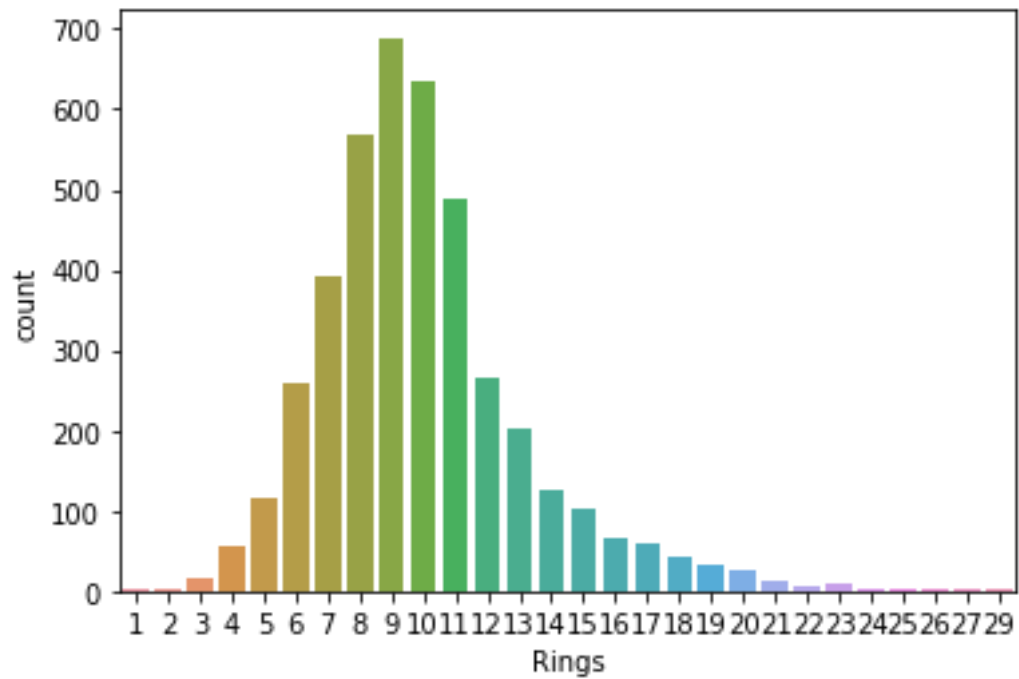
(iii)Multi-Variate Analysis

(i) UNIVARIATE

```
# countplot
sns.countplot(data=df,x="Rings")
```

In []:

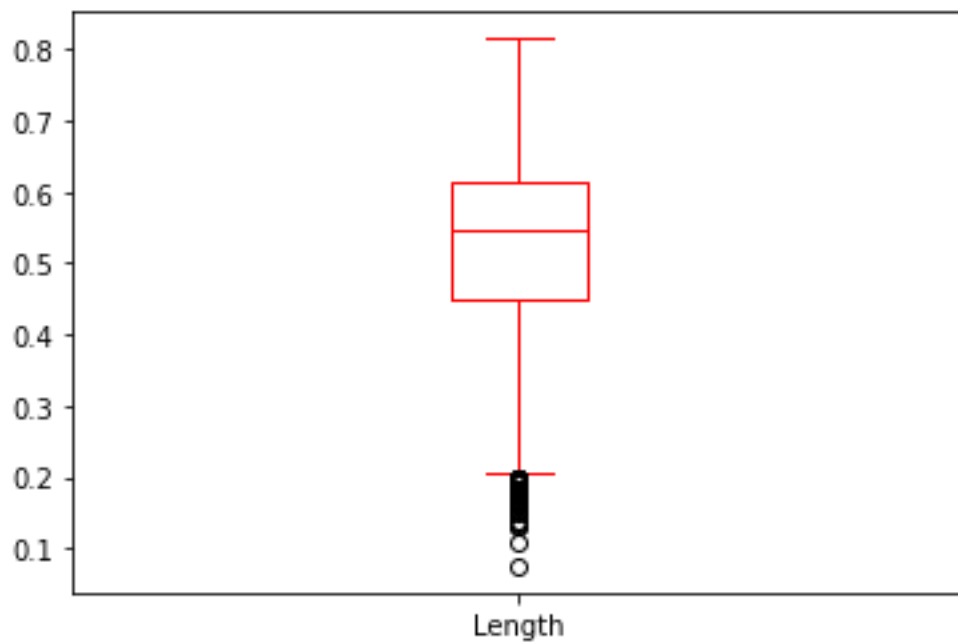
Out[]:



In []:

```
#boxplot
df.boxplot(column=['Length'], grid=False, color='Red')
```

Out[]:

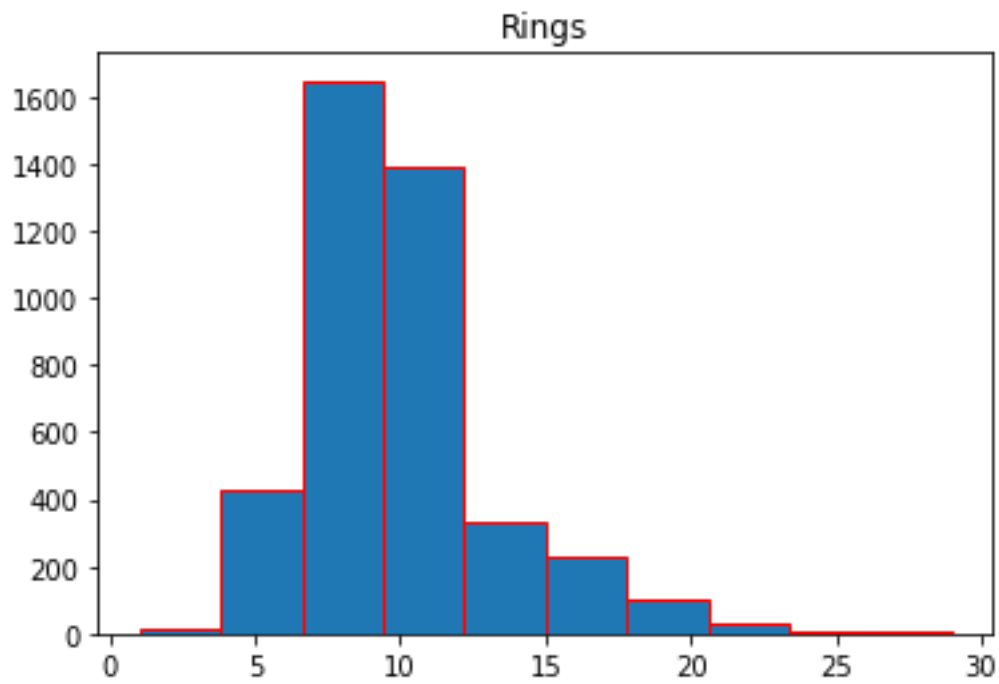


In []:

```
#histogram
df.hist(column='Rings', grid=False, edgecolor='Red')
```

Out[]:

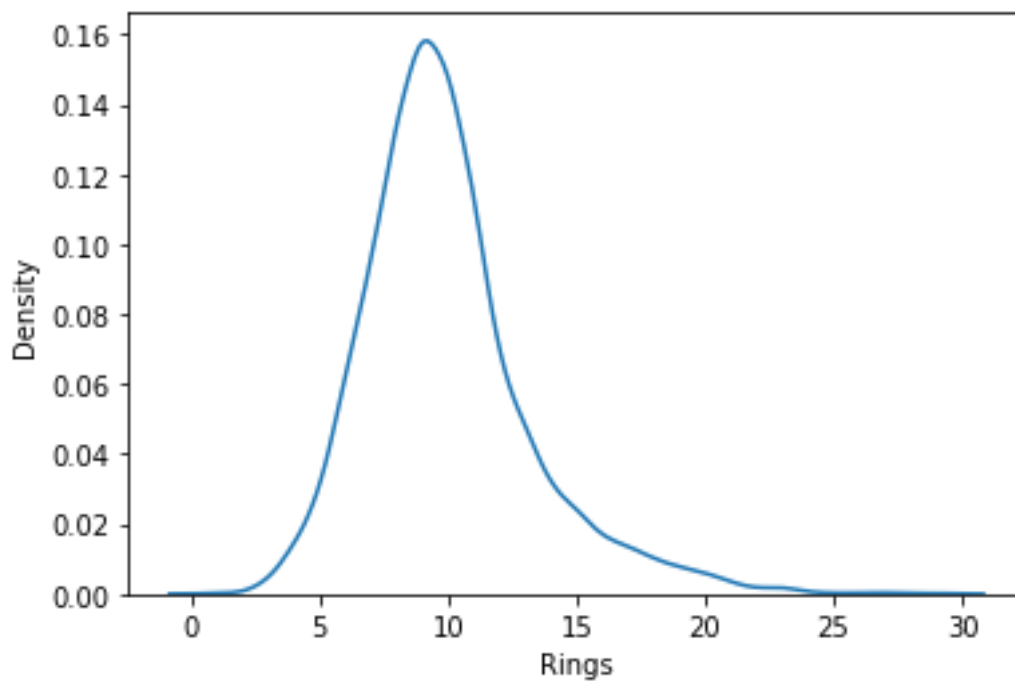
```
array([[[]],          dtype=object)
```



In []:

```
#kdeplot  
sns.kdeplot(df['Rings'])
```

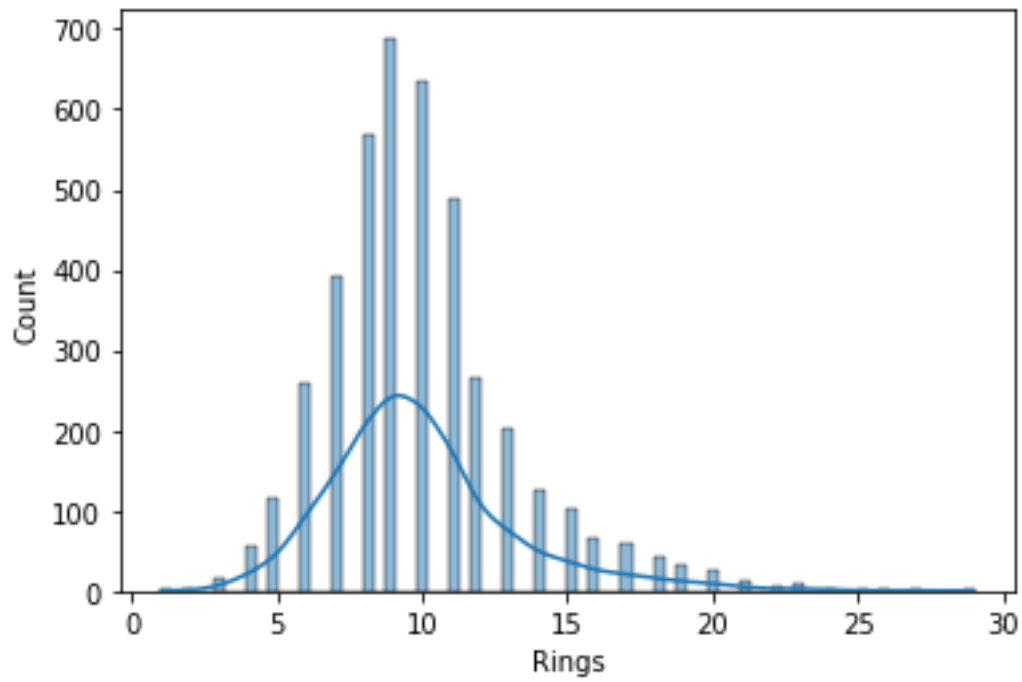
Out[]:



In []:

```
#histplot  
sns.histplot(df.Rings,kde=True)
```

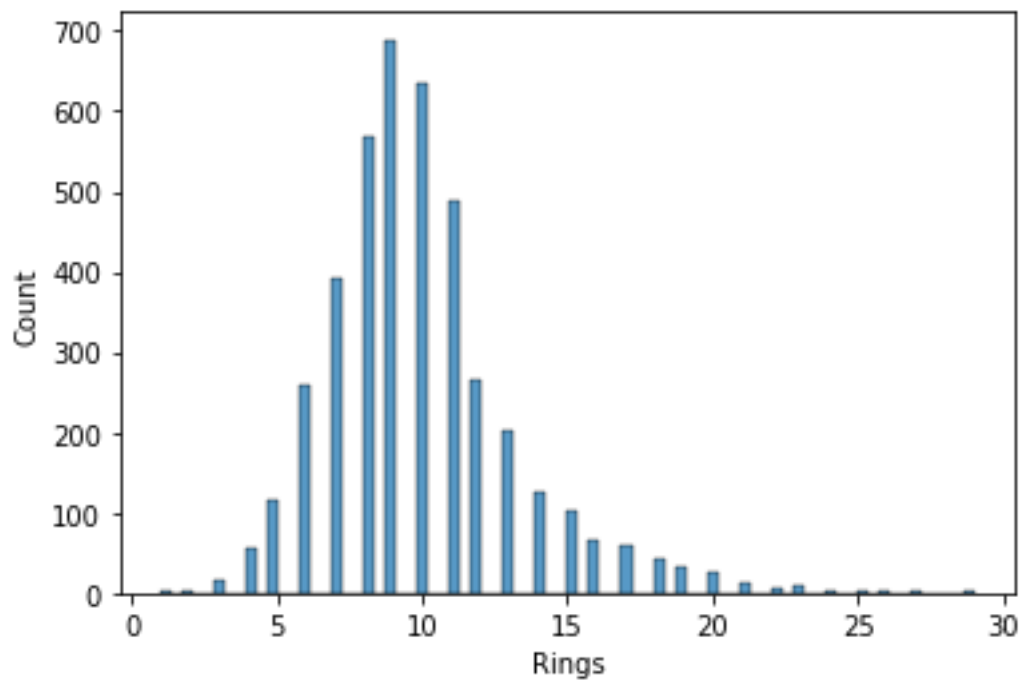
Out[]:



In []:

```
#histplot  
sns.histplot(df['Rings'])
```

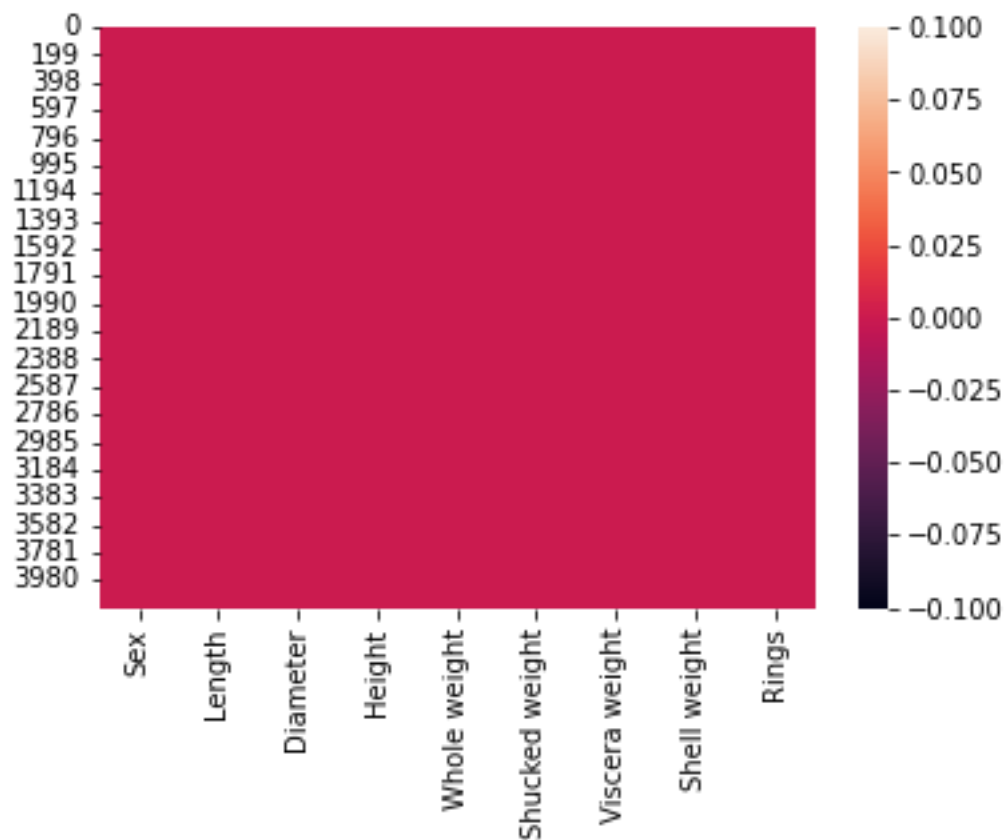
Out[]:



In []:

```
#heatmap  
sns.heatmap(df.isnull())
```

Out[]:

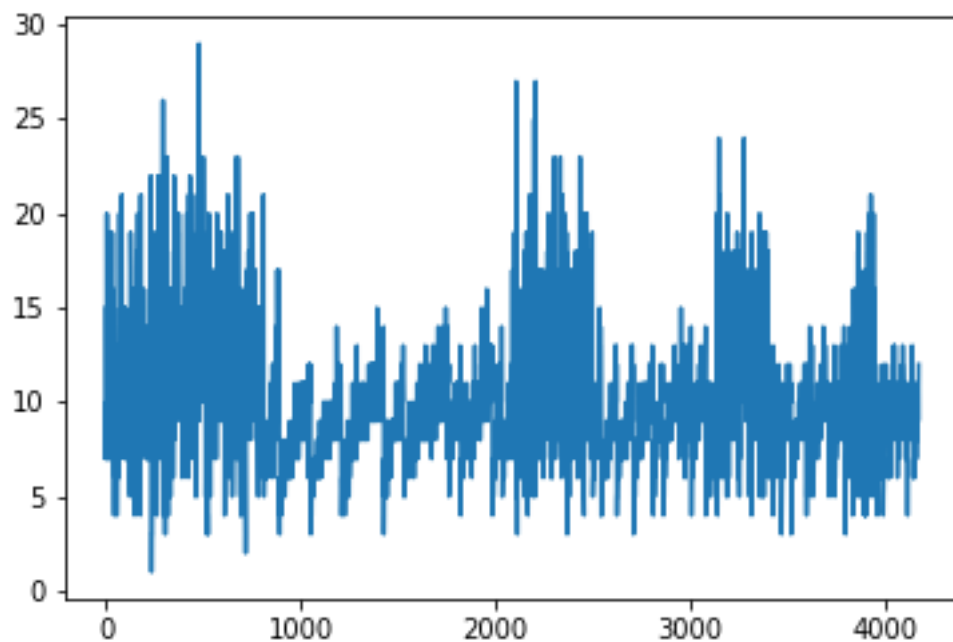


In []:

```
#line plot plt.plot(df['Rings'])
```

Out[]:

[]

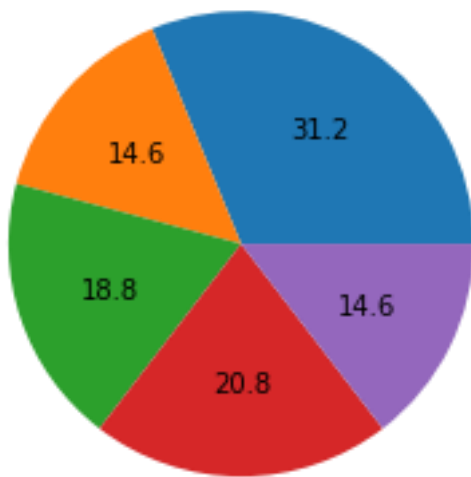


In []:

```
#piechart plt.pie(df['Rings'].head(), autopct='%0.1f')
```

Out[]:

```
([,
,
,
,
],
[Text(0.6111272563215626, 0.9146165735327998, ''),
Text(-0.8270237769092663, 0.725280409515335, ''),
Text(-1.041623153479572, -0.35358337932554523, ''),
Text(-5.149471704824549e-08, -1.0999999999999998, ''),
Text(0.9865599777267362, -0.4865176362145796, '')],
[Text(0.33334213981176136, 0.4988817673815271, '31.2'),
Text(-0.4511038783141452, 0.39560749609927365, '14.6'),
Text(-0.5681580837161301, -0.1928636614502974, '18.8'),
Text(-2.8088027480861175e-08, -0.5999999999999993, '20.8'),
Text(0.5381236242145833, -0.2653732561170434, '14.6')])
```

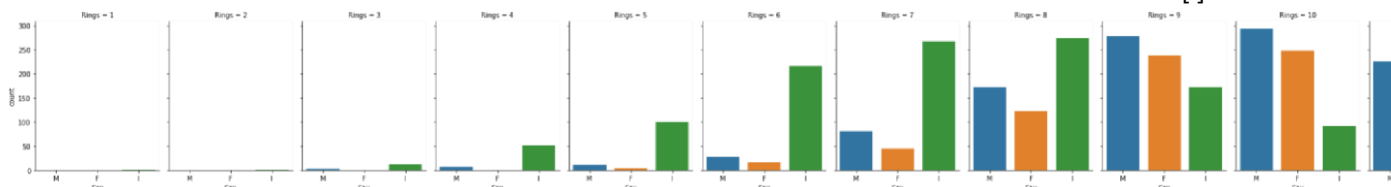


(ii) BIVARIATE

In []:

```
#countplot
sns.catplot(x="Sex",col="Rings",data=df, kind="count",height=4, aspect=.7)
```

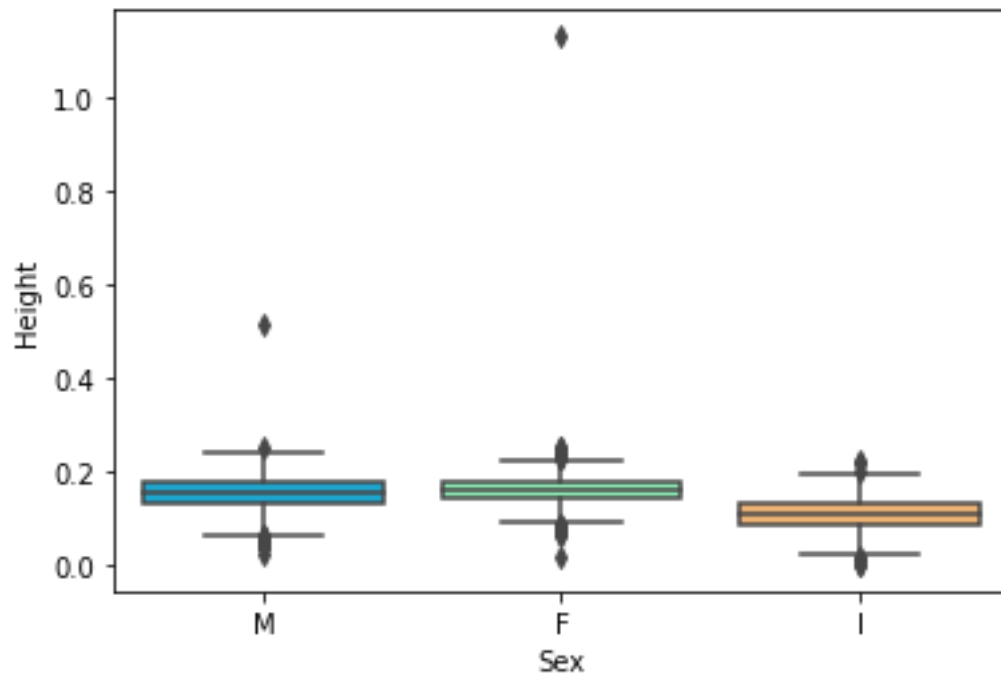
Out []:



In []:

```
#boxplot
sns.boxplot(x='Sex',y='Height',data=df,palette='rainbow')
```

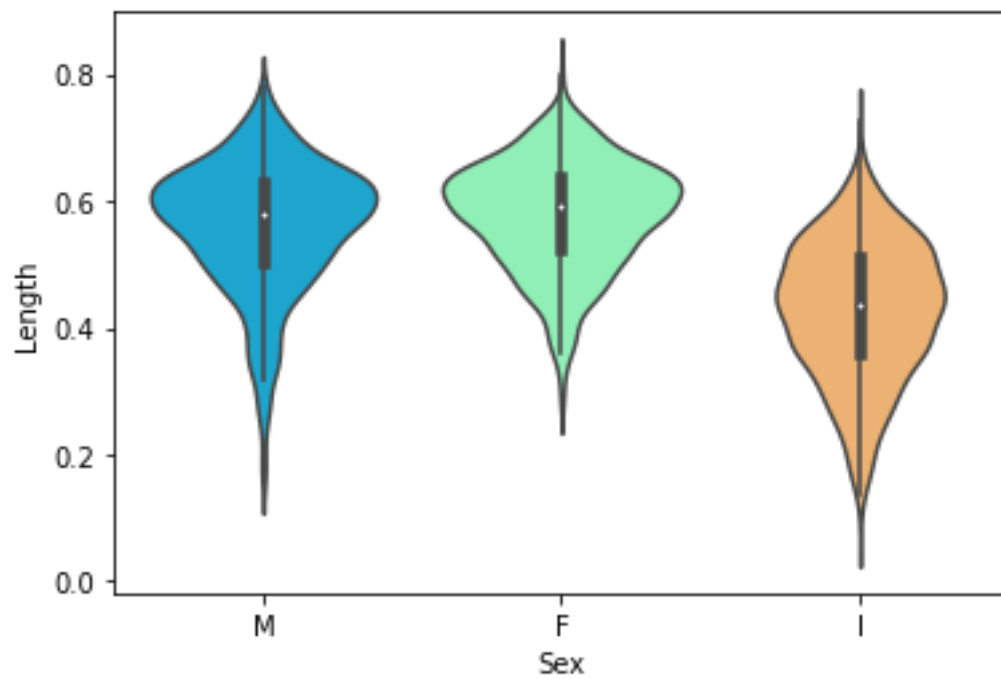
Out []:



In []:

```
#violin plot
sns.violinplot(x="Sex", y="Length", data=df,palette='rainbow')
```

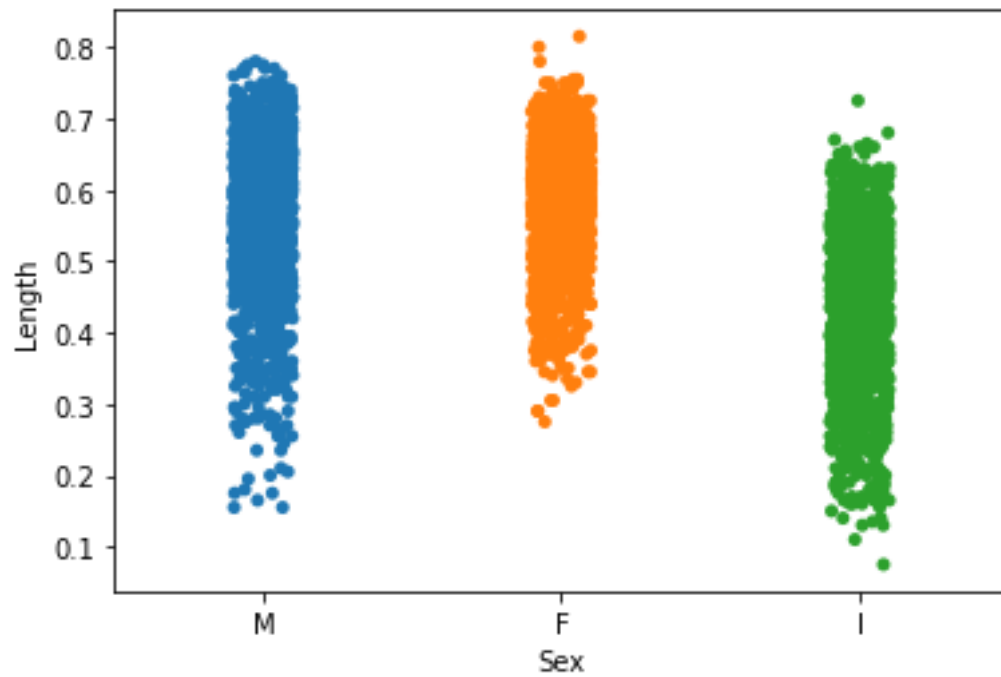
Out[]:



In []:

```
#strip plot
sns.stripplot(x="Sex", y="Length", data=df)
```

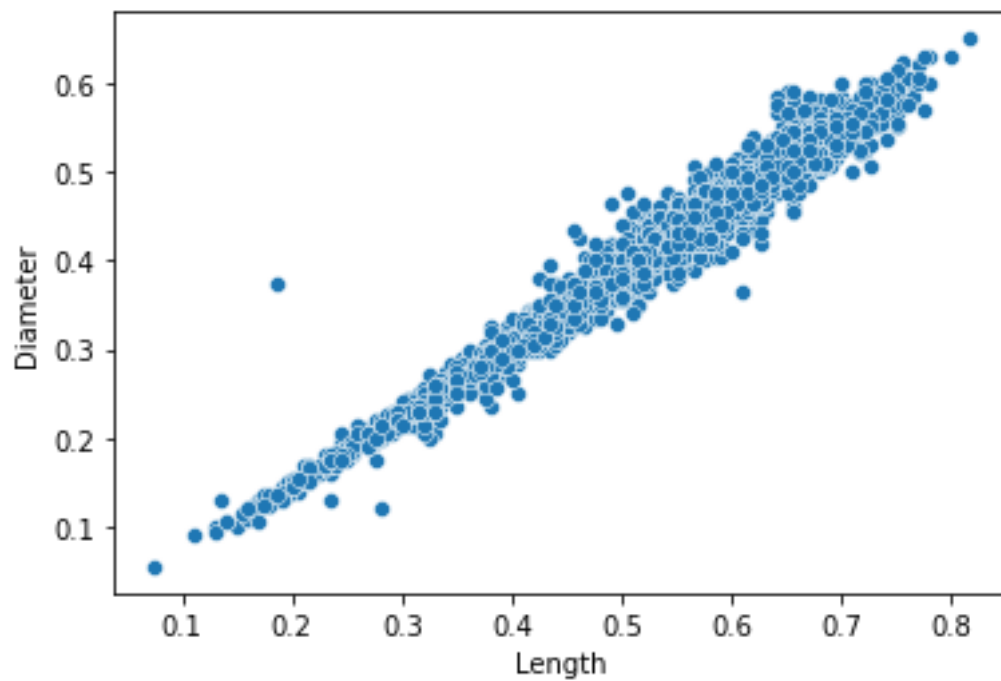
Out[]:



In []:

```
#scatter plot  
sns.scatterplot(x = df["Length"],y = df["Diameter"])
```

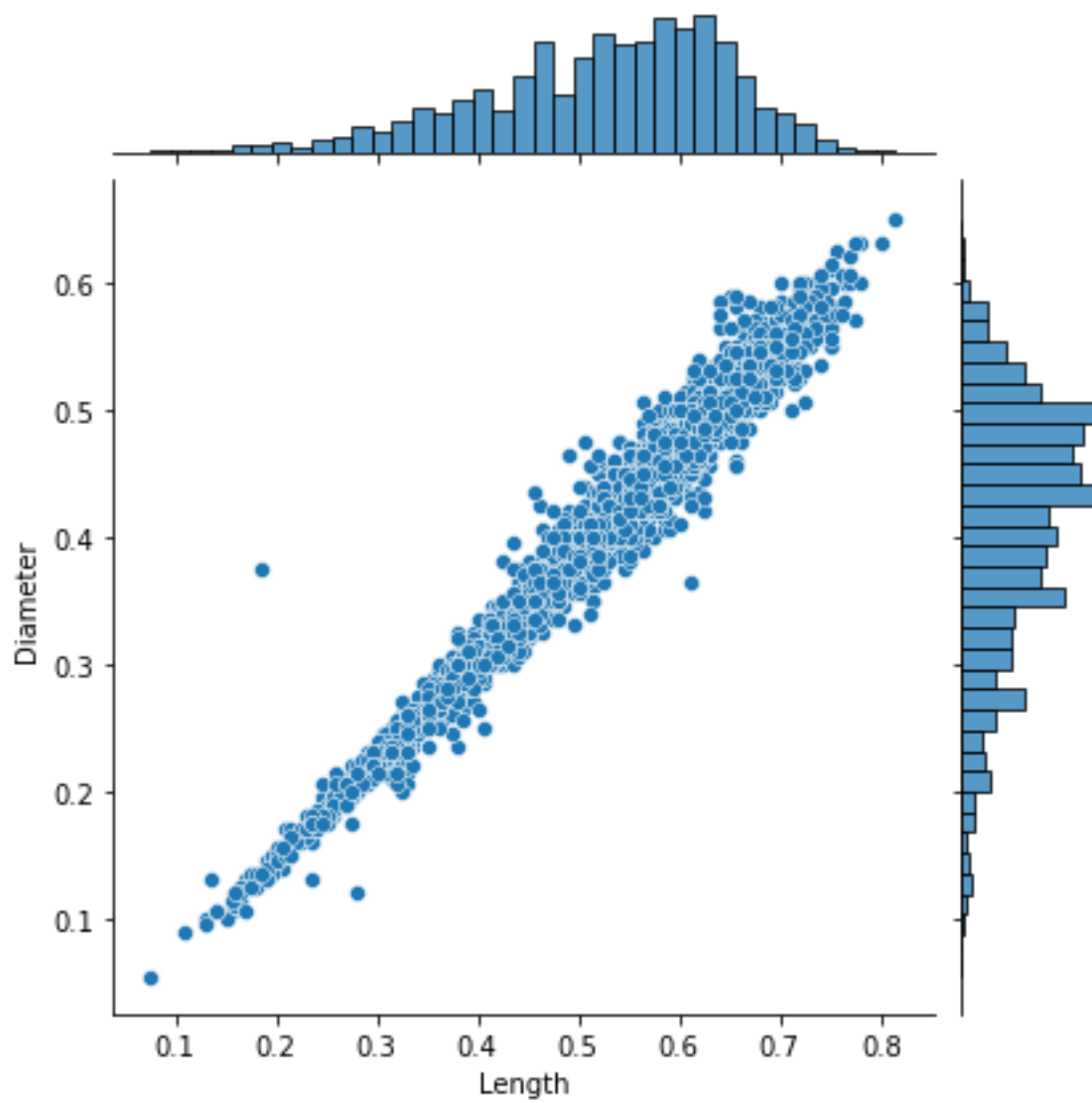
Out[]:



In []:

```
#joint_plot  
sns.jointplot(x="Length",y="Diameter",data=df)
```

Out[]:



(III) MULTI-VARIATE

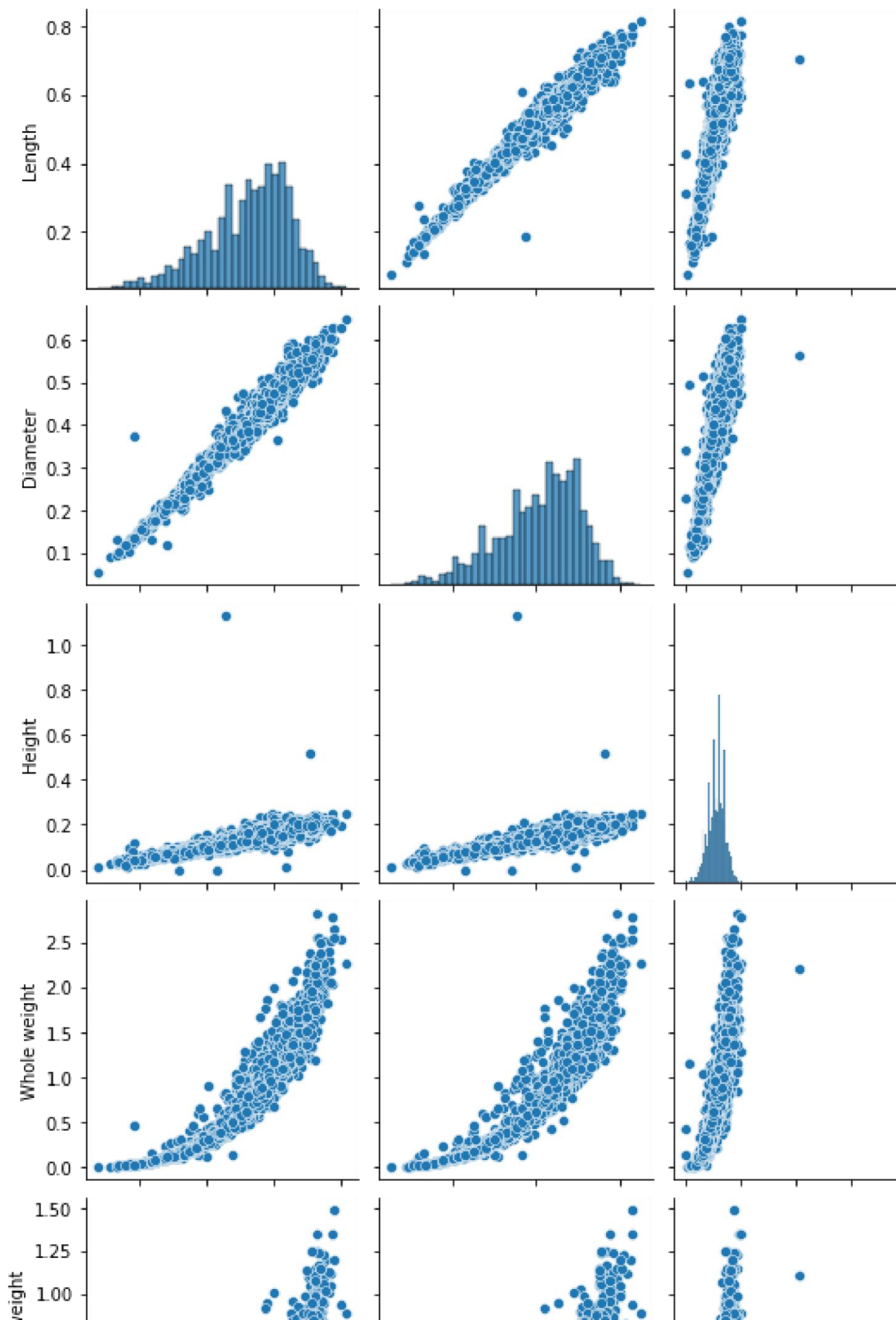
```
#Boxplot fig, ax1 = plt.subplots(figsize=(8,5)) testPlot =  
sns.boxplot(ax=ax1, x='Length', y='Diameter', hue='Sex', data=df)
```

In []:

```
sns.pairplot(df)
```

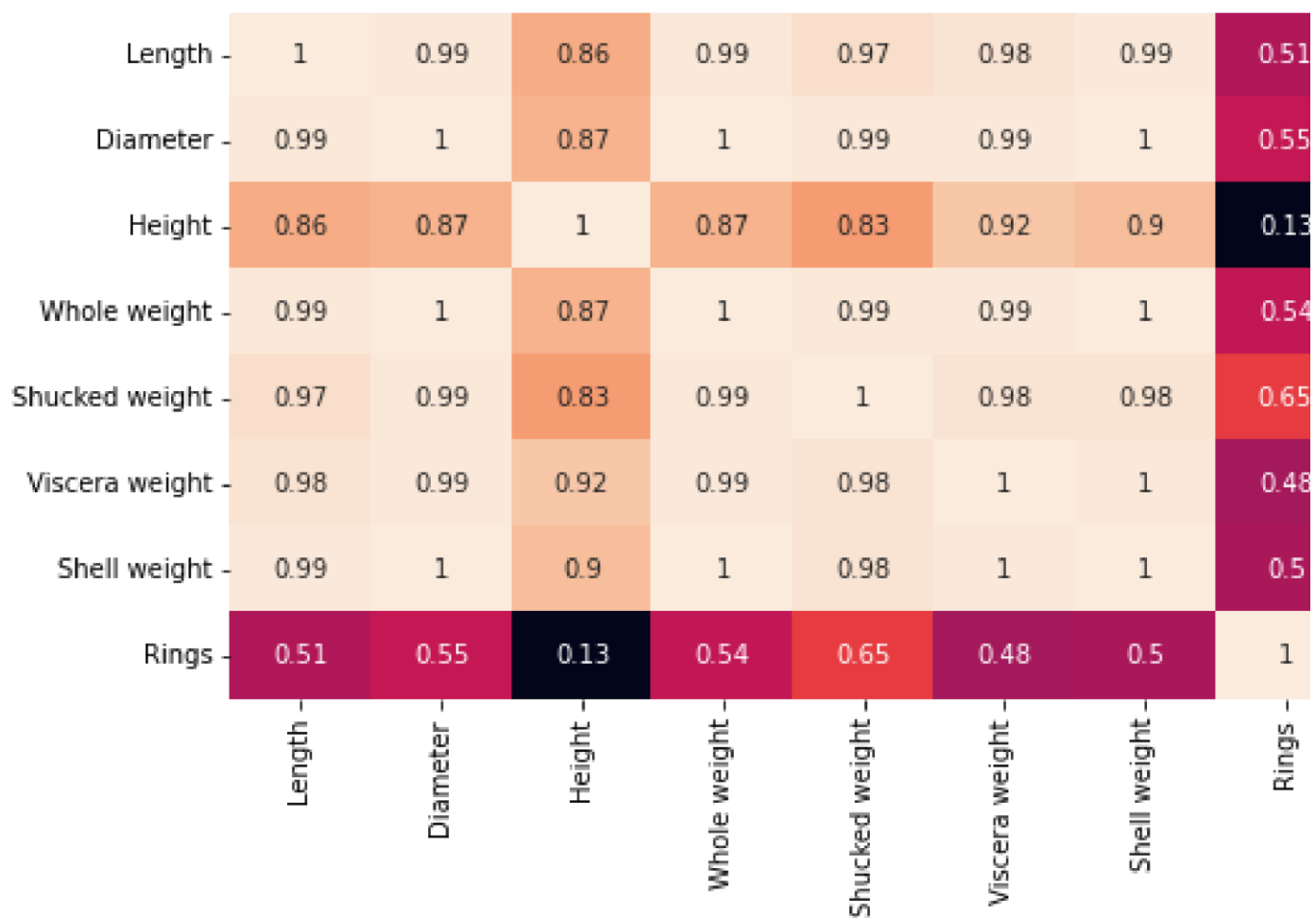
In []:

Out[]:



```
In [ ]:
fig=plt.figure(figsize=(10,5)) sns.heatmap(df.head().corr(),annot=True)
```

Out[]:



4. Perform descriptive statistics on the dataset

```
df
```

Out[]:

	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	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10

```

4      I      0.330      0.255      0.080      0.2050      0.0895      0.0395      0.0550      7
...
...      ...      ...      ...      ...      ...      ...      ...      ...

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

```

4177 rows × 9 columns

```

In [ ]:
df.head()

```

```

Out[ ]:

```

	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

```

In [ ]:
df.info()
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
```

In []:

```
df.describe()
```

Out[]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

In []:

```
numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.select_dtypes(include = [object]).columns
```

In []:

```
df[numerical_features].mean()
```

Out[]:

```
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
```

In []:

```
df[numerical_features].median()
```

Out[]:

```
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
```

In []:

```
percentage = [df[numerical_features].quantile(0),
df[numerical_features].quantile(0.25),
df[numerical_features].quantile(0.50),
df[numerical_features].quantile(0.75),
df[numerical_features].quantile(1)] percentage
```

Out[]:

```
[Length      0.0750
Diameter     0.0550
Height       0.0000
Whole weight 0.0020
Shucked weight 0.0010
Viscera weight 0.0005
Shell weight 0.0015
Rings        1.0000
Name: 0.0, dtype: float64, Length      0.4500
Diameter     0.3500
Height       0.1150
Whole weight 0.4415
Shucked weight 0.1860
Viscera weight 0.0935
Shell weight 0.1300
Rings        8.0000
Name: 0.25, dtype: float64, 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
Name: 0.5, dtype: float64, Length      0.615
Diameter     0.480
Height       0.165
Whole weight 1.153
Shucked weight 0.502
Viscera weight 0.253
Shell weight 0.329
Rings       11.000
Name: 0.75, dtype: float64, Length      0.8150
Diameter     0.6500
Height       1.1300
Whole weight 2.8255
Shucked weight 1.4880
Viscera weight 0.7600
Shell weight 1.0050
Rings       29.0000
```

```
Name: 1.0, dtype: float64]
```

In []:

```
df[numerical_features].value_counts()
```

Out[]:

Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0.075	0.055	0.010	0.0020	0.0010	0.0005	0.0	0.0
015	1	1					
0.590	0.465	0.155	1.1360	0.5245	0.2615	0.2	0.2
750	11	1					
		0.165	1.1150	0.5165	0.2730	0.2	0.2
750	10	1					
		0.170	1.0425	0.4635	0.2400	0.2	0.2
700	10	1					
		0.195	1.0885	0.3685	0.1870	0.3	0.3
750	17	1					
..							
0.485	0.370	0.155	0.9680	0.4190	0.2455	0.2	0.2
365	9	1					
	0.375	0.110	0.4640	0.2015	0.0900	0.1	0.1
490	8	1					
		0.125	0.5620	0.2505	0.1345	0.1	0.1
525	8	1					
		0.130	0.5535	0.2660	0.1120	0.1	0.1
570	8	1					
0.815	0.650	0.250	2.2550	0.8905	0.4200	0.7	0.7
975	14	1					

Length: 4177, dtype: int64

In []:

```
df[numerical_features].mode()
```

Out[]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	0.550	0.45	0.15	0.2225	0.175	0.1715	0.275	9.0
1	0.625	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In []:

```
df[numerical_features].std()
```

Out[]:

Length	0.120093
Diameter	0.099240
Height	0.041827
Whole weight	0.490389
Shucked weight	0.221963
Viscera weight	0.109614
Shell weight	0.139203
Rings	3.224169 dtype:
	float64

```
df[numerical_features].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
```

```
df[numerical_features].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
```

```
df[numerical_features].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
```

5. Check for Missing values and deal with them

```
df.isnull()
```

[illegible]

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	False	False

4177 rows × 9 columns

```
df.isnull().any()
```

In []:

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

Out[]:

```
df.isnull().sum()
```

In []:

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

Out[]:

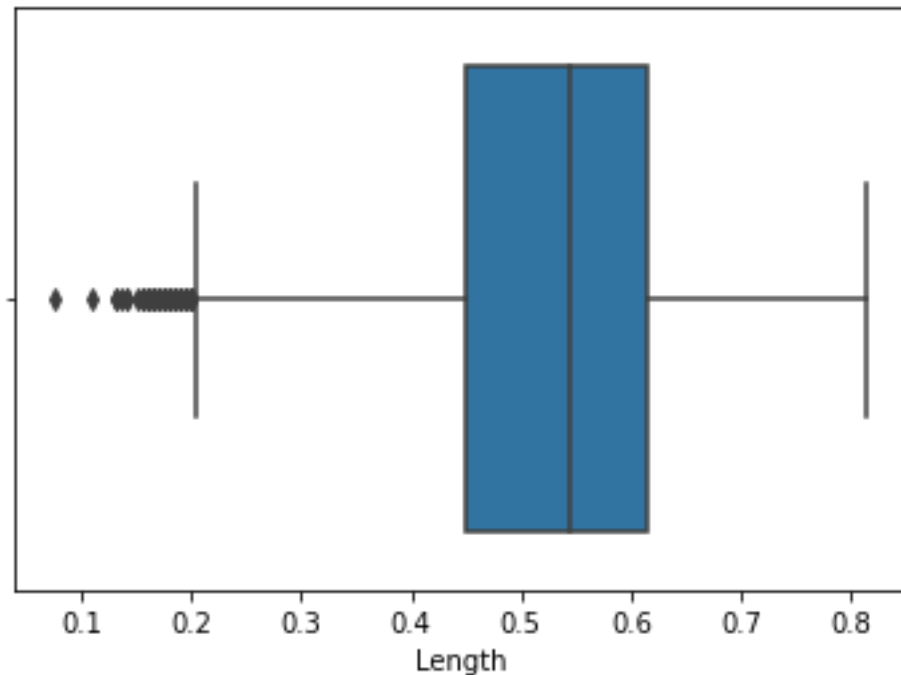
6. Find the outliers and replace them outliers

```
#length
```

In []:

```
sns.boxplot(x=df['Length'])
```

Out[]:



```
In [ ]:  
q1 = df['Length'].quantile(0.25) q2 = df['Length'].quantile(0.75) iqr = q2-  
q1 q1, q2, iqr
```

Out[]:

```
(0.45, 0.615, 0.16499999999999998)
```

```
In [ ]:  
upper_limit = q2 + (1.5 * iqr) lower_limit = q1 - (1.5 * iqr) lower_limit,  
upper_limit
```

Out[]:

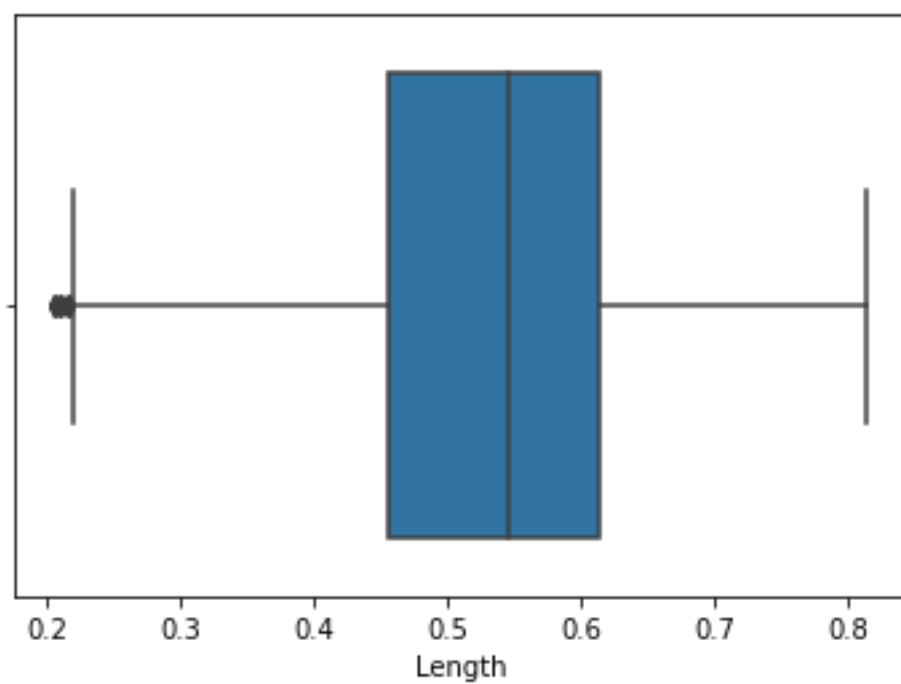
```
(0.202500000000000004, 0.8624999999999999)
```

```
In [ ]:  
new_df = df.loc[(df['Length'] <= upper_limit) & (df['Length'] >=  
lower_limit)]  
print('before removing outliers:', len(df))  
print('after removing outliers:', len(new_df))  
print('outliers:', len(df)-len(new_df))  
before removing outliers: 4177 after removing  
outliers: 4128 outliers: 49
```

```
In [ ]:  
new_df = df.copy() new_df.loc[(new_df['Length']>upper_limit), 'Length'] =  
upper_limit new_df.loc[(new_df['Length']<lower_limit), 'Length'] =  
lower_limit
```

```
In [ ]:  
sns.boxplot(x=new_df['Length'])
```

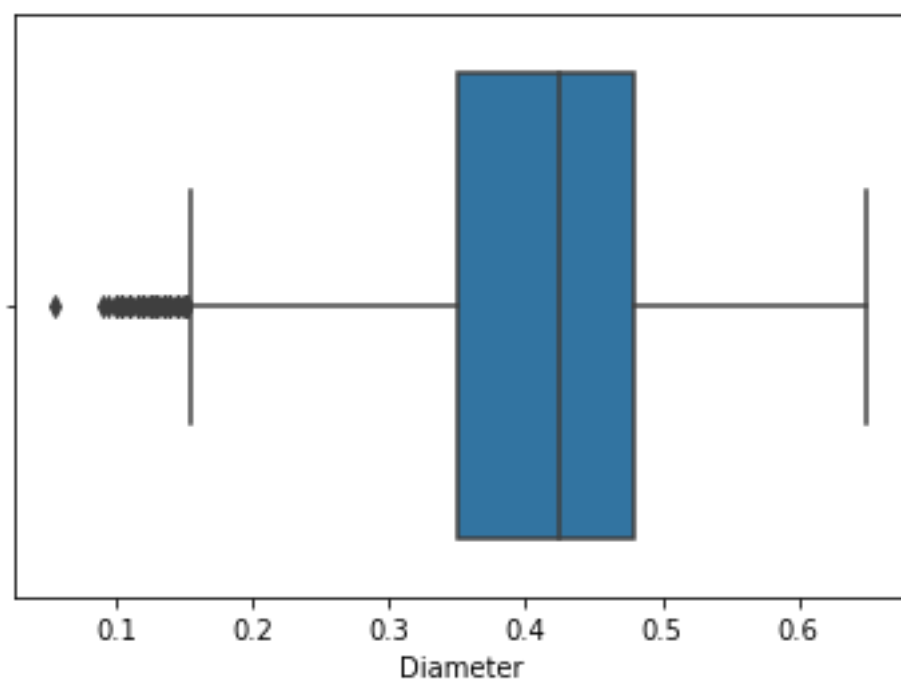
Out[]:



In []:

```
#Diameter
sns.boxplot(x=df['Diameter'])
```

Out[]:



In []:

```
q1 = df['Diameter'].quantile(0.25) q2 = df['Diameter'].quantile(0.75) iqr =
q2-q1
q1, q2, iqr
```

Out[]:

```
(0.45, 0.615, 0.16499999999999998)
```

```
In [ ]:
upper_limit = q2 + (1.5 * iqr) lower_limit = q1 - (1.5 * iqr) lower_limit,
upper_limit
```

```
Out[ ]:
(0.20250000000000004, 0.8624999999999999)
```

```
In [ ]:
new_df = df.loc[(df['Diameter'] <= upper_limit) & (df['Diameter'] >=
lower_limit)] print('before removing outliers:', len(df)) print('after
removing outliers:', len(new_df)) print('outliers:', len(df)-len(new_df))
before removing outliers: 4177 after removing outliers: 4027 outliers: 150
```

```
In [ ]:
new_df = df.copy() new_df.loc[(new_df['Diameter']>upper_limit), 'Diameter']
= upper_limit new_df.loc[(new_df['Diameter']<lower_limit), 'Diameter'] =
lower_limit
```

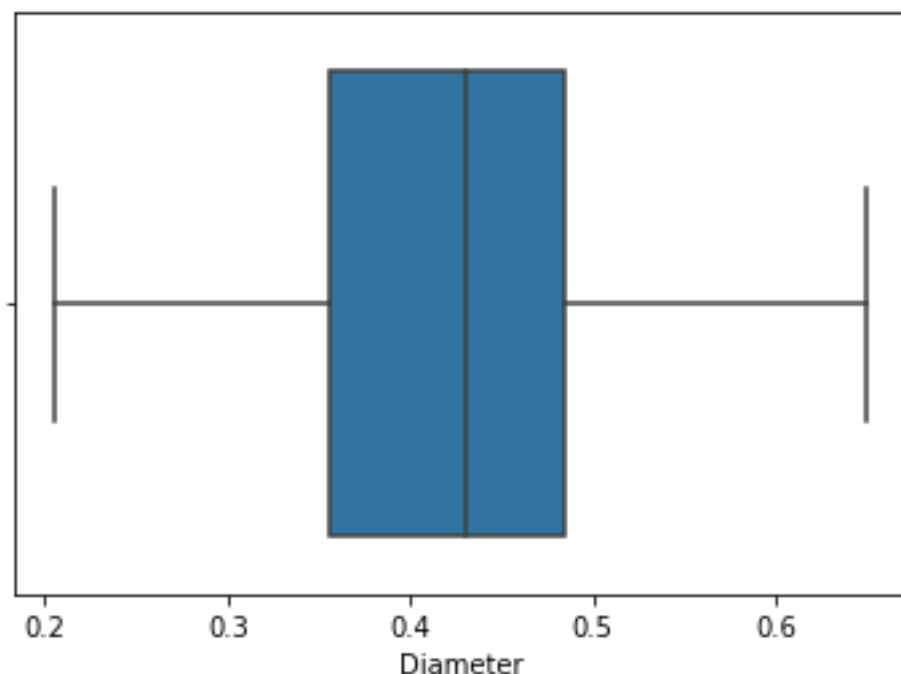
```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1817: Settin
gWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead
```

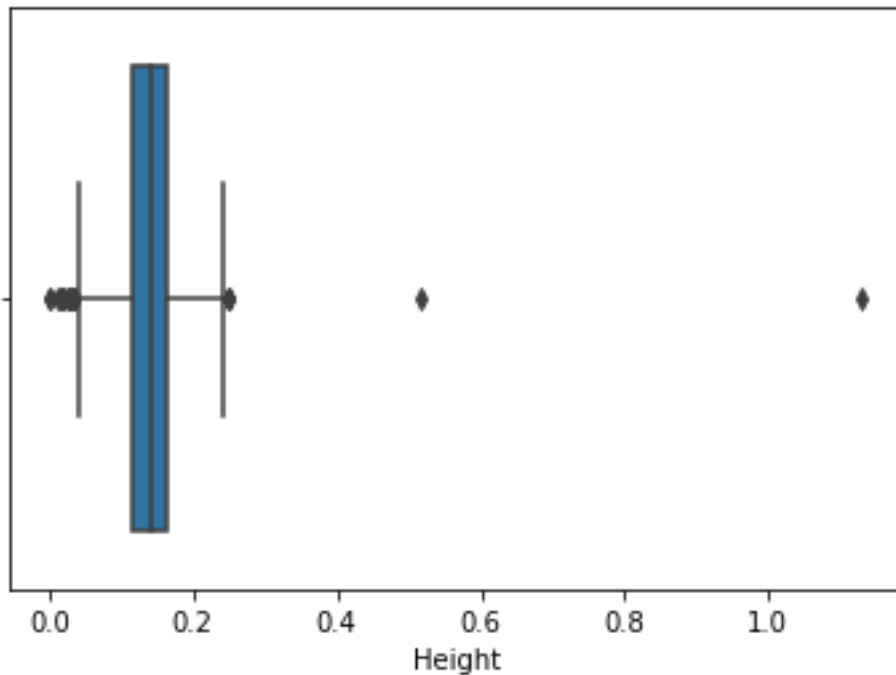
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
self._setitem_single_column(loc, value, pi)
```

```
In [ ]:
sns.boxplot(x=new_df['Diameter'])
```



```
In [ ]:
#Height
sns.boxplot(x=df['Height'])
```

```
Out[ ]:
```



```
In [ ]:
q1 = df['Height'].quantile(0.25) q2 = df['Height'].quantile(0.75) iqr = q2-q1
q1, q2, iqr
```

```
Out[ ]:
(0.45, 0.615, 0.16499999999999998)
```

```
In [ ]:
upper_limit = q2 + (1.5 * iqr) lower_limit
= q1 - (1.5 * iqr) lower_limit,
upper_limit
```

```
Out[ ]:
(0.202500000000000004, 0.8624999999999999)
```

```
In [ ]:
new_df = df.loc[(df['Height'] <= upper_limit) & (df['Height'] >=
lower_limit)] print('before removing outliers:', len(df)) print('after
removing outliers:', len(new_df)) print('outliers:', len(df)-len(new_df))
before removing outliers: 4177 after removing outliers: 153 outliers: 4024
```

```
In [ ]:
new_df = df.copy() new_df.loc[(new_df['Height']>upper_limit), 'Height'] =
upper_limit new_df.loc[(new_df['Height']<lower_limit), 'Height'] =
lower_limit
```

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1817: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

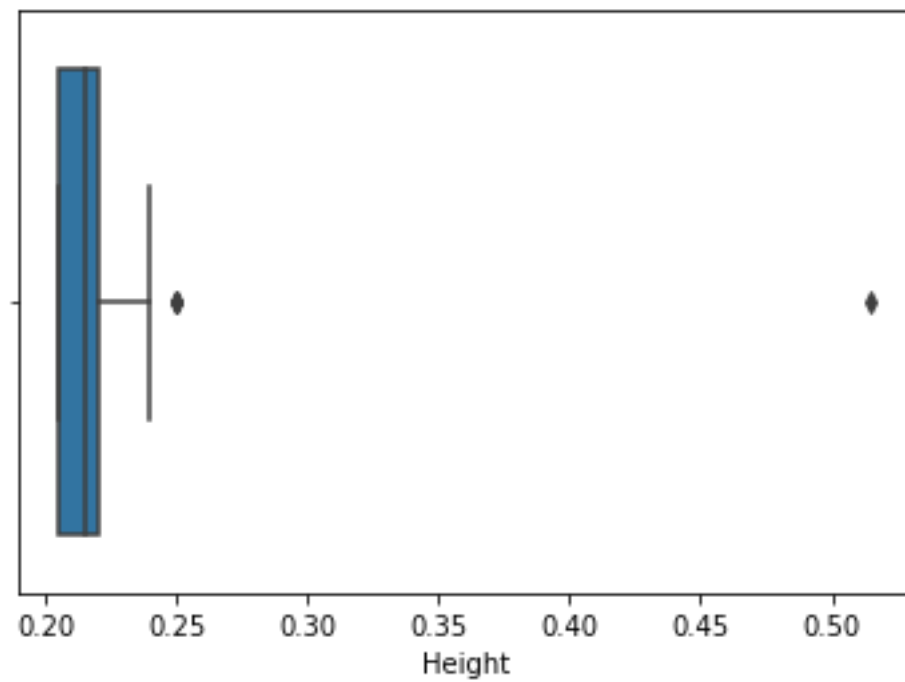
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

self._setitem_single_column(loc, value, pi)

```
In [ ]:
sns.boxplot(x=new_df['Height'])
```

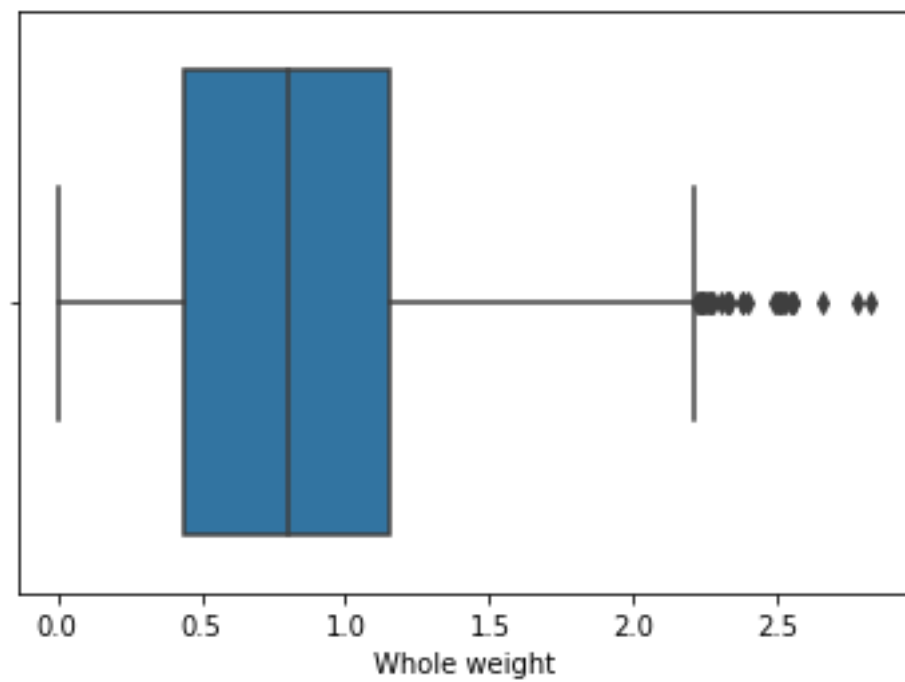
Out[]:



In []:

```
#Whole Weight sns.boxplot(x=df['Whole  
weight'])
```

Out[]:



In []:

```
q1 = df['Whole weight'].quantile(0.25) q2 = df['Whole weight'].quantile(0.75)  
iqr = q2-q1 q1, q2, iqr
```

Out[]:

```
(0.45, 0.615, 0.16499999999999998)
```

```
In [ ]:
upper_limit = q2 + (1.5 * iqr) lower_limit = q1 - (1.5 * iqr) lower_limit,
upper_limit
```

```
Out[ ]:
(0.20250000000000004, 0.8624999999999999)
```

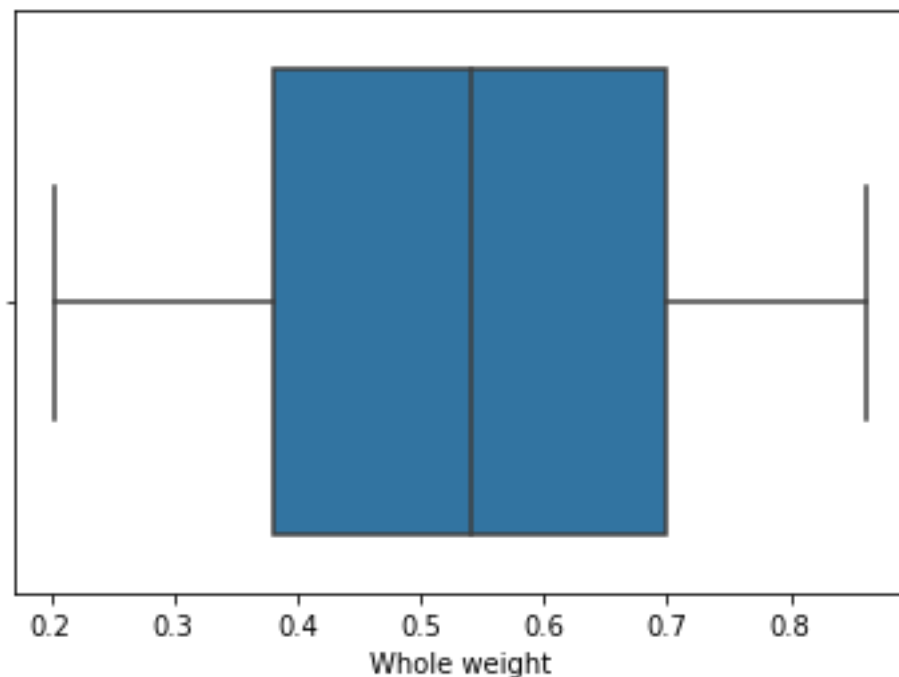
```
In [ ]:
new_df = df.loc[(df['Whole weight'] <= upper_limit) & (df['Whole weight']
>= lower_limit)] print('before removing
outliers:', len(df)) print('after removing
outliers:', len(new_df)) print('outliers:',
len(df)-len(new_df)) before removing
outliers: 4177 after removing outliers: 1872
outliers: 2305
```

```
In [ ]:
new_df = df.copy() new_df.loc[(new_df['Whole weight']>upper_limit), 'Whole
weight'] = upper_limit new_df.loc[(new_df['Whole weight']<lower_limit),
'Whole weight'] = lower_limit

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1817: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame. Try
using .loc[row_indexer,col_indexer] = value instead

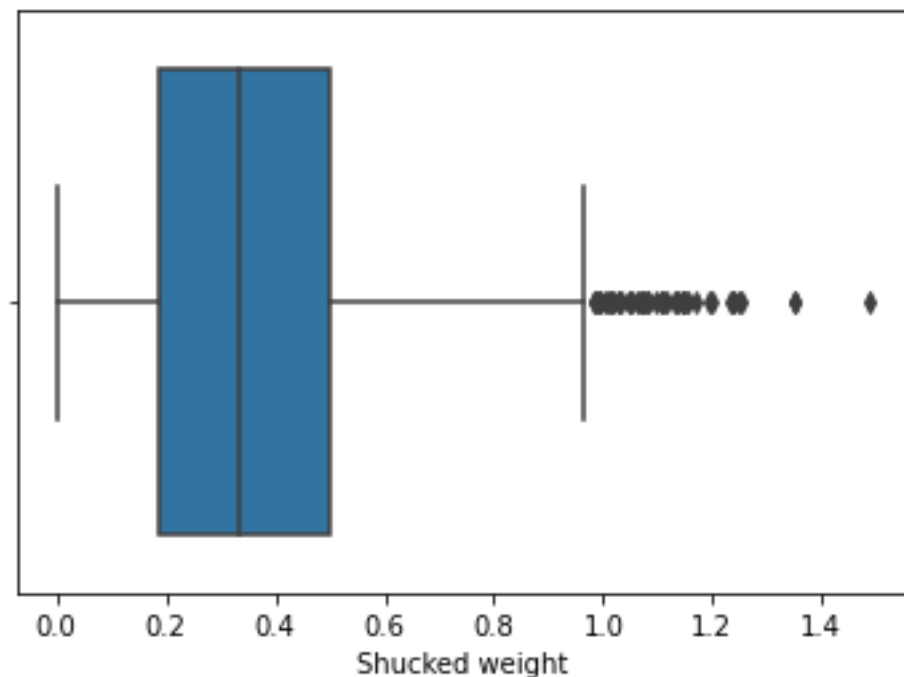
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
self._setitem_single_column(loc, value, pi)
```

```
In [ ]:
sns.boxplot(x=new_df['Whole weight'])
```



```
In [ ]:
#Shucked weight
sns.boxplot(x=df['Shucked weight'])
```

Out[]:



In []:

```
q1 = df['Shucked weight'].quantile(0.25)
q2 = df['Shucked weight'].quantile(0.75)
iqr = q2-q1 q1, q2, iqr
```

Out[]:

```
(0.45, 0.615, 0.16499999999999998)
```

In []:

```
upper_limit = q2 + (1.5 * iqr) lower_limit = q1 - (1.5 * iqr) lower_limit,
upper_limit
```

Out[]:

```
(0.202500000000000004, 0.8624999999999999)
```

In []:

```
new_df = df.loc[(df['Shucked weight'] <= upper_limit) & (df['Shucked weight']
>= lower_limit)] print('before removing outliers:', len(df)) print('after
removing outliers:',len(new_df)) print('outliers:', len(df)-len(new_df))
before removing outliers: 4177 after removing outliers: 2900 outliers: 1277
```

In []:

```
new_df = df.copy() new_df.loc[(new_df['Shucked weight']>upper_limit),
'Shucked weight'] = upper_limit new_df.loc[(new_df['Shucked
weight']<lower_limit), 'Shucked weight'] = lower_limit
```

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1817: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

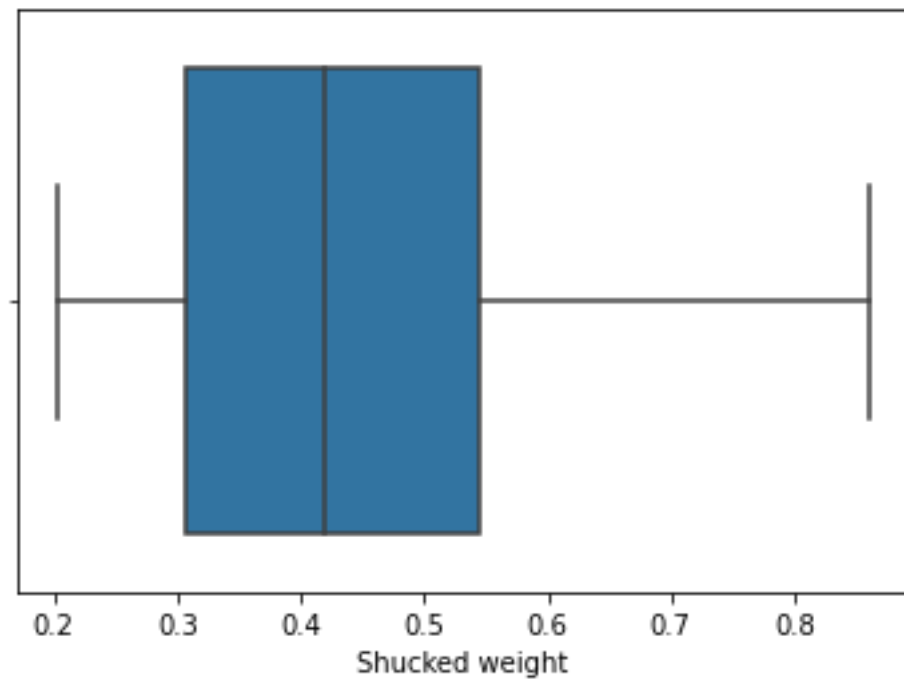
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
self._setitem_single_column(loc, value, pi)
```

In []:


```
sns.boxplot(x=new_df['Shucked weight'])
```

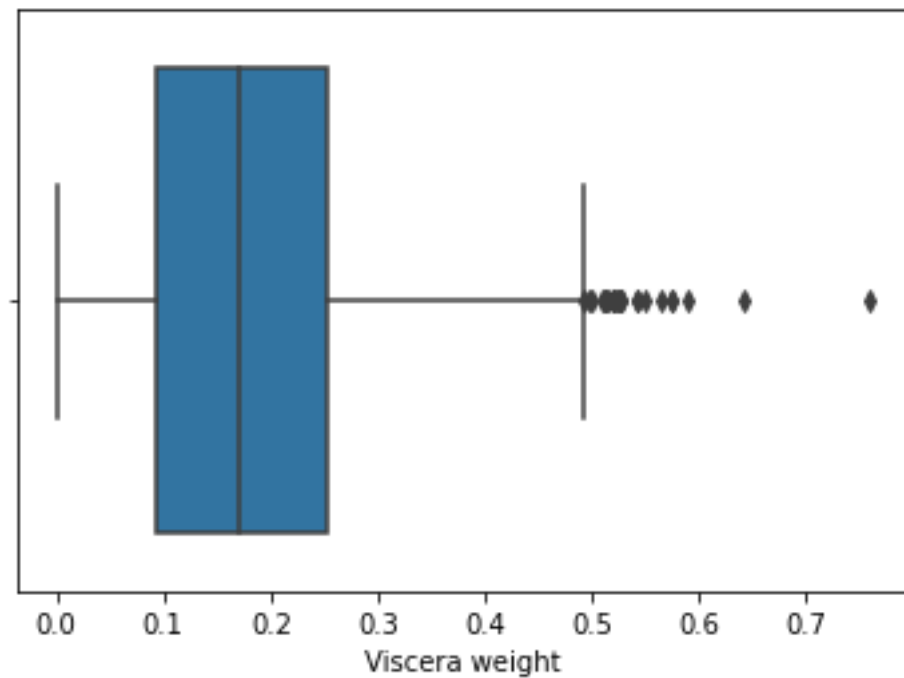
Out[]:



```
#Viscera weight  
sns.boxplot(x=df['Viscera weight'])
```

In []:

Out[]:



```
q1    = df['Viscera weight'].quantile(0.25)    q2    = df['Viscera  
weight'].quantile(0.75) iqr = q2-q1 q1, q2, iqr
```

In []:

Out[]:

```
(0.45, 0.615, 0.16499999999999998)
```

In []:

```
upper_limit = q2 + (1.5 * iqr) lower_limit
= q1 - (1.5 * iqr) lower_limit,
upper_limit
```

Out []:

```
(0.202500000000000004, 0.8624999999999999)
```

In []:

```
new_df = df.loc[(df['Viscera weight'] <= upper_limit) & (df['Viscera weight']
>= lower_limit)] print('before removing outliers:', len(df)) print('after
removing outliers:', len(new_df)) print('outliers:', len(df)-len(new_df))
before removing outliers: 4177 after removing outliers: 1646 outliers: 2531
```

In []:

```
new_df = df.copy() new_df.loc[(new_df['Viscera weight']>upper_limit),
'Viscera weight'] = upper_limit new_df.loc[(new_df['Viscera
weight']<lower_limit), 'Viscera weight'] = lower_limit
```

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1817: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

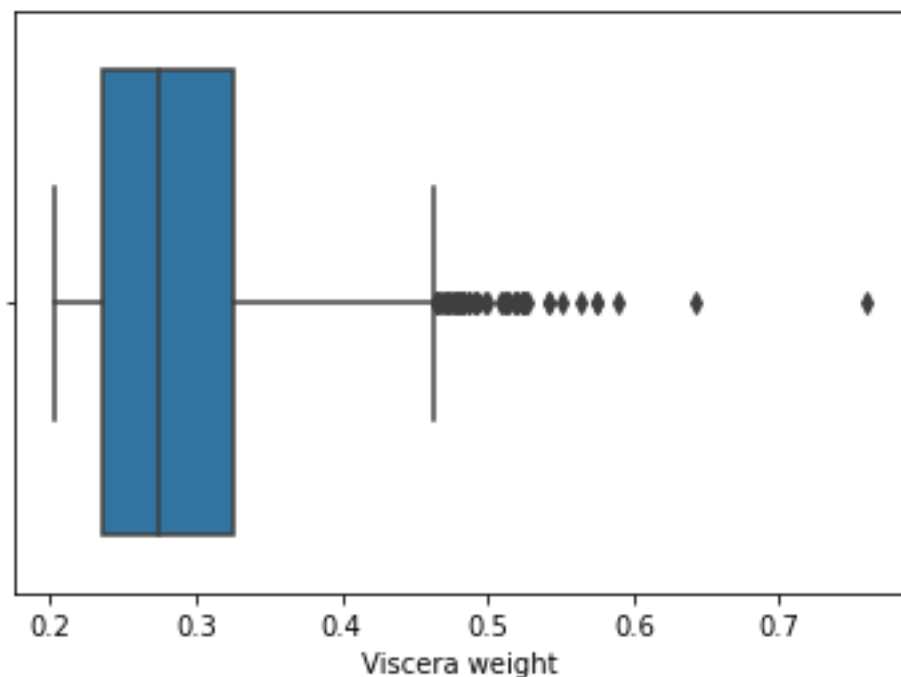
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
self._setitem_single_column(loc, value, pi)
```

In []:

```
sns.boxplot(x=new_df['Viscera weight'])
```

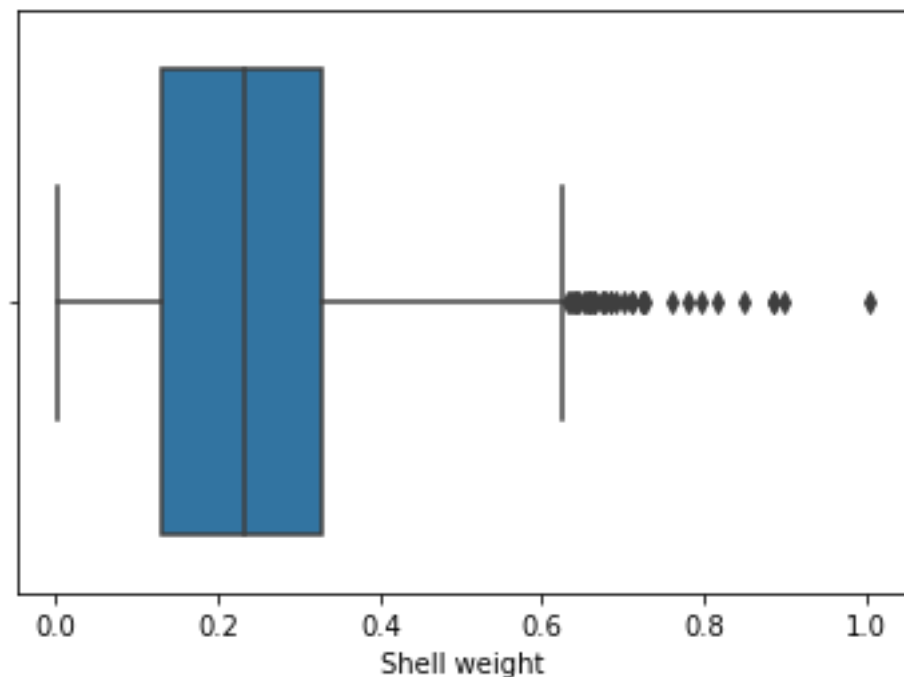
Out []:



In []:

```
#shell weight
sns.boxplot(x=df['Shell weight'])
```

Out[]:



In []:

```
q1 = df['Shell weight'].quantile(0.25) q2 = df['Shell weight'].quantile(0.75)
iqr = q2-q1 q1, q2, iqr
```

Out[]:

```
(0.45, 0.615, 0.16499999999999998)
```

In []:

```
upper_limit = q2 + (1.5 * iqr) lower_limit = q1 - (1.5 * iqr) lower_limit,
upper_limit
```

Out[]:

```
(0.202500000000000004, 0.8624999999999999)
```

In []:

```
new_df = df.loc[(df['Shell weight'] <= upper_limit) & (df['Shell weight']
>= lower_limit)] print('before removing
outliers:', len(df)) print('after removing
outliers:', len(new_df)) print('outliers:',
len(df)-len(new_df)) before removing
outliers: 4177 after removing outliers: 2373
outliers: 1804
```

In []:

```
new_df = df.copy() new_df.loc[(new_df['Shell weight']>upper_limit), 'Shell
weight'] = upper_limit new_df.loc[(new_df['Shell weight']<lower_limit),
'Shell weight'] = lower_limit
```

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1817: SettingWithCopyWarning:

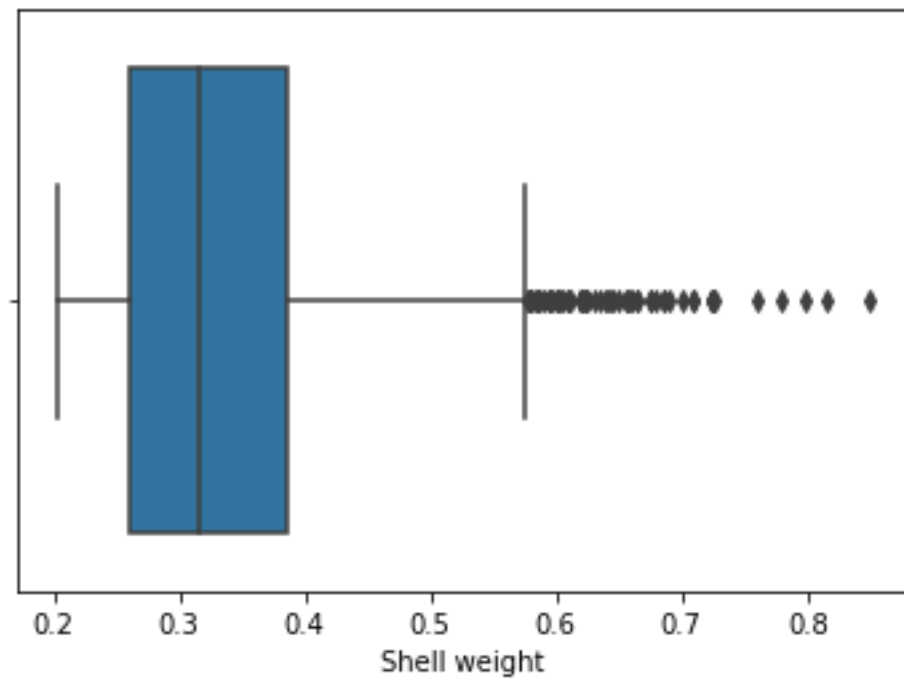
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
self._setitem_single_column(loc, value, pi)

In []:

```
sns.boxplot(x=new_df['Shell weight'])
```

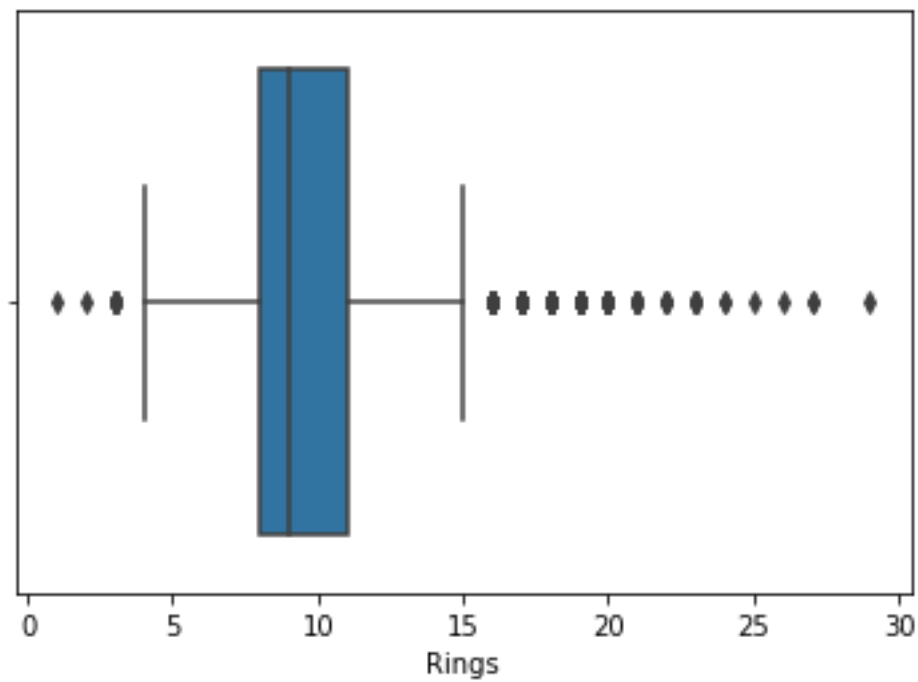
Out[]:



In []:

```
#Rings  
sns.boxplot(x=df['Rings'])
```

Out[]:



In []:

```
q1 = df['Rings'].quantile(0.25) q2 = df['Rings'].quantile(0.75) iqr = q2-q1  
q1, q2, iqr
```

Out[]:

```
(0.45, 0.615, 0.16499999999999998)
```

In []:

```
upper_limit = q2 + (1.5 * iqr) lower_limit = q1 - (1.5 * iqr) lower_limit,
upper_limit
```

Out[]:

```
(0.202500000000000004, 0.8624999999999999)
```

In []:

```
new_df = df.loc[(df['Rings'] <= upper_limit) & (df['Rings'] >= lower_limit)]
print('before removing outliers:', len(df)) print('after removing
outliers:',len(new_df)) print('outliers:', len(df)-len(new_df)) before
removing outliers: 4177 after removing outliers: 0 outliers: 4177
```

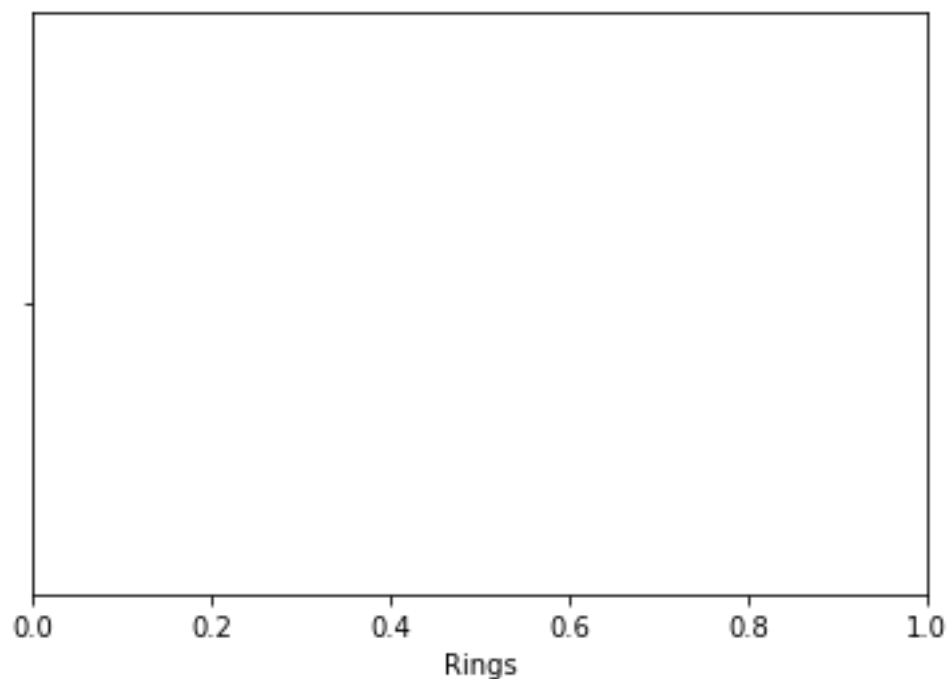
In []:

```
new_df = df.copy() new_df.loc[(new_df['Rings']>upper_limit), 'Rings'] =
upper_limit new_df.loc[(new_df['Rings']<lower_limit), 'Rings'] = lower_limit
```

In [219]:

```
sns.boxplot(x=new_df['Rings'])
```

Out[219]:



7. Check for Categorical columns and perform encoding

In []:

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

df

Out[]:

	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	0.1500	15

[illegible]

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

4177 rows × 9 columns

```
In [ ]:
enc = OneHotEncoder(drop='first')
enc_df =
pd.DataFrame(enc.fit_transform(df[['Sex']]).toarray())
df
=df.join(enc_df)
df.head()
```

```
Out[ ]:
```

	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

8. Split the data into dependent and independent variables

```
In [ ]:
x= df.iloc[:,1:8] x
```

```
Out[ ]:
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100

```

3    0.440  0.365   0.125   0.5160  0.2155  0.1140  0.1550

4    0.330  0.255   0.080   0.2050  0.0895  0.0395  0.0550

...    ...    ...    ...    ...    ...    ...    ...

4172   0.565   0.450   0.165   0.8870   0.3700   0.2390   0.2490

```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950

4177 rows × 7 columns

```
In [ ]:
```

```
y=df.iloc[:,8] y
```

```
Out[ ]:
```

```

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

```
In [ ]:
```

```

scale = StandardScaler() scaledX = scale.fit_transform(x)
print(scaledX)
[[-0.57455813 -0.43214879 -1.06442415 ... -0.60768536 -0.72621157 -
 0.63821689]
 [-1.44898585 -1.439929   -1.18397831 ... -1.17090984 -1.20522124 -
 1.21298732]
 [ 0.05003309  0.12213032 -0.10799087 ... -0.4634999  -0.35668983 -
 0.20713907]
 ...
 [ 0.6329849  0.67640943  1.56576738 ...  0.74855917  0.97541324
 0.49695471]

```



```
[ 0.84118198    0.77718745    0.25067161 ...    0.77334105    0.73362741
0.41073914]
[ 1.54905203  1.48263359  1.32665906 ...  2.64099341  1.78744868
 1.84048058]]
```

10. Split the data into training and testing

```
In [ ]:
from sklearn.model_selection import train_test_split

In [ ]:
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2)

In [ ]:
print(x.shape, x_train.shape, x_test.shape,y_train.shape, y_test.shape)
(4177, 7) (3341, 7) (836, 7) (3341,) (836,)
```

11. Build the Model

```
In [ ]:
from sklearn.linear_model import LinearRegression

In [ ]:
linearmodel = LinearRegression()
```

12. Train the Model

```
In [ ]:
linearmodel.fit(x_train, y_train)
```

```
Out[ ]:
LinearRegression()
```

13. Test the Model

```
In [ ]:
y_train_pred = linearmodel.predict(x_train)    y_test_pred =
linearmodel.predict(x_test)
```

```
In [ ]:
y_test_pred
```

```
Out[ ]:
array([ 8.70365574, 10.39057789,  9.40293106, 10.68158892,  7.57464889,
 4.79636131,    8.67332668, 14.02754984,    9.87864789,  7.25750569,
10.85233616,    8.50778462,  7.15078854,    9.32393986,  5.76619464,
 7.49797457,    5.76688568,  6.2241946 ,    6.18696811,  9.25884721,
11.5681706 ,   12.13604097, 10.98303848, 11.69986211,  7.83702624,
 9.31462136,   10.40327259,  6.96378017,    5.81839663, 12.26690446,
10.86817082,    9.02369275,  8.12760588,    8.83313399,  7.73292169,
 6.91592262,    6.07309496,  7.88643423,    9.63507119,  4.7209354 ,
11.34436294,   10.57283751, 10.49665213, 12.88894543, 12.28423666,
 8.12974709,    7.58999374,  9.08527348, 10.6411015 ,  5.89349286,
 8.30141881,    7.24999833,  8.25347176,    9.08328759,  8.99010706,
 9.10730271,   14.52308851,  9.41403346, 10.23775522,  8.0477514 ,
 9.70375626,   11.03640036,  9.5435852 , 10.8850895 ,  8.08267684,
10.57414181,   11.41222985,  7.2778873 ,    6.70117135,  9.80118094,
 7.10712737,    6.50572362, 11.70751204,    8.76972122,  6.96654951,
 6.47886711,    6.93951927,  7.458727 ,   11.01554321, 10.36006982,
 7.61624674,    9.16579141,  8.04761481, 12.29038942,  6.20248568,
10.8963334 ,   10.28472972, 14.8411801 , 13.2740434 ,  7.49258174,
```

13.25230426, 10.63162313, 11.48753783, 9.12967209, 9.03018002,
10.45067022, 8.82633807, 8.36808156, 8.1866462 , 10.24960421,
10.21056392, 10.72719629, 9.7728592 , 7.38834192, 8.51824041,
10.75769027, 11.85414492, 12.87270151, 9.16270242, 9.16333672,
7.67461895, 11.76874831, 10.16330431, 10.05005681, 9.51670308,
9.62637671, 9.1479746 , 10.34868794, 10.84331115, 7.357215 ,
12.41355527, 14.38136393, 11.14666246, 12.38530452, 7.54358072,
9.40835468, 9.88315837, 12.11159133, 7.94462274, 7.47162288,
8.90032367, 10.43913282, 9.07962681, 9.73850771, 5.83375835,
8.36362646, 10.9307715 , 6.82609483, 8.29060331, 9.69949734,
8.89587123, 5.05902985, 12.66596704, 9.28710853, 12.5326091 ,
8.79671925, 9.49859385, 11.96474424, 11.52223318, 9.60091683,
8.53619584, 15.32591088, 14.00301768, 8.97484674, 13.05508044,
6.2030136 , 9.21207997, 7.936351 , 10.16988917, 11.47842671,
7.46631576, 9.75499741, 10.27332107, 9.23986585, 5.9322595 ,
12.28408611, 7.92151313, 15.25635443, 8.21425868, 5.09784281,
8.64249299, 11.10284085, 7.78901197, 8.83090234, 9.1646292 ,
7.70480134, 9.88123776, 8.25763834, 10.09331237, 9.23862615,
7.90180974, 13.23613166, 9.26144527, 13.1751266 , 10.4482136 ,
8.92745413, 11.50202025, 8.43899771, 5.35863306, 11.15256721,
12.0758802 , 9.72580288, 8.61229334, 10.91349829, 14.60692034,
9.87282727, 9.54625507, 8.67259558, 11.2890254 , 13.33908019,
11.02243873, 8.69347318, 9.8305109 , 10.23686184, 11.2406998 ,
10.17177327, 13.30614216, 7.20501867, 11.59622094, 6.78619841,
11.79121352, 8.89281728, 8.17336191, 6.66181377, 7.88222639,
11.5961669 , 5.62055863, 10.95213761, 10.42565767, 6.91986363,
8.22808635, 9.13631704, 10.60651687, 10.57288312, 12.48198032,
10.72400654, 7.94057143, 8.13184549, 9.72402567, 6.08561859,
10.35685691, 11.45185208, 6.12270113, 7.96196924, 9.67026482,
8.41039006, 7.86133336, 13.70955472, 9.71358231, 12.30386605,
9.45850521, 8.75621392, 7.18052467, 11.26230795, 10.62351901,
13.39193134, 10.66694535, 11.24857396, 7.31369597, 11.34385404,
5.25197892, 9.59055659, 16.84346196, 11.16831052, 11.62804782,
9.53833939, 10.24000367, 8.6230046 , 9.30733633, 12.46773085,
13.64994481, 9.82838202, 10.0234437 , 12.99484459, 10.45651057,
13.40761272, 8.77910533, 12.33978409, 10.67709288, 5.63536671,
9.94034672, 7.14256339, 9.12794164, 9.21566578, 5.80513006,
10.6905346 , 11.97328277, 8.30889084, 10.60075702, 14.67068785,
11.09720556, 8.4771601 , 6.21293923, 8.24916844, 10.21094152,
7.51466157, 11.06498401, 7.13865753, 15.10717164, 10.62721341,
10.7494065 , 9.36798918, 10.84697015, 6.19124128, 9.81320126,
8.87149442, 12.25522958, 10.0180955 , 10.25697378, 10.63826128,
8.83487496, 8.92686532, 10.00754949, 7.30269568, 12.07507174,
11.93297256, 10.38317051, 7.18057693, 8.41307958, 8.63722592,
7.14923596, 9.24164041, 6.20876584, 13.52351759, 10.87929999,
10.40607483, 13.05707136, 12.39494182, 11.75230309, 12.42380122,
9.49893263, 9.37433398, 7.94309167, 8.96190867, 7.62962106,
10.85825406, 11.77951614, 9.16274124, 12.18049248, 7.75075404,
7.18890683, 7.57054635, 13.37668116, 7.99159672, 11.59558707,
9.33023381, 8.65031741, 11.19050069, 12.56526943, 10.34615758,
10.14710918, 13.6822781 , 12.17950611, 5.05668754, 10.10464104,
9.53755259, 9.472615 , 10.20387222, 6.9805034 , 10.36745308,
8.5797176 , 10.0305631 , 9.85976851, 14.7588884 , 10.33370292,
11.95220189, 10.73191172, 6.24084469, 7.94205021, 13.2574999 ,
7.4230356 , 9.69102779, 10.19306655, 7.69531367, 8.78069463,

5.98531412, 9.64295733, 10.25986894, 10.43382077, 7.21469625,
11.13358543, 9.44012819, 9.03507192, 10.31114951, 6.41797086,
7.45384674, 10.59417698, 8.63488075, 14.02982444, 10.65890459,
9.86162823, 11.31451071, 12.59497748, 10.03074457, 7.334356 ,
11.35510583, 11.90079632, 8.12093928, 10.94441475, 9.51383607,
8.61736287, 7.69098049, 10.88432917, 27.92119074, 7.30574213,
12.16580075, 9.99772607, 12.01395884, 8.46372125, 7.9846562 ,
9.20938127, 10.81645842, 7.48697286, 7.84853701, 9.87249706,
12.42241391, 11.34890737, 10.50687099, 10.8923027 , 12.4523061 ,
7.71112748, 6.89139266, 9.07208527, 13.33541534, 8.6100613 ,
9.73762238, 8.02254027, 16.21679373, 6.7788581 , 9.61586488,
8.47538654, 12.61070772, 7.38898974, 7.94305825, 7.7085148 ,
10.61500173, 11.34651659, 13.33513797, 7.60626229, 20.23054277,
8.24406194, 10.52400655, 9.0689655 , 10.45315667, 12.47231165,
8.91146384, 8.67365154, 9.72583367, 12.30204789, 13.6415676 ,
7.32722295, 8.25658547, 8.17855799, 11.49287729, 8.80588627,
12.25548643, 11.55860069, 10.76620005, 9.16792817, 13.80701665,
8.25280724, 8.35282716, 7.55768412, 8.22063676, 9.44288372,
10.01885335, 7.49525516, 8.37458402, 10.92937369, 10.67946102,
13.86376797, 11.71283279, 5.9284063 , 9.59255804, 7.57785783,
11.75865845, 10.29701554, 8.54738043, 7.89933844, 8.3008639 ,
12.80214221, 7.16951618, 11.69354963, 9.23287449, 10.60070989,
9.42484143, 13.39247208, 9.8366729 , 9.79308612, 13.00791904,
8.63395954, 8.02483998, 10.992329 , 14.02775855, 10.6388897 ,
8.69134756, 9.46875832, 10.11163209, 9.7990431 , 13.32735769,
7.15836617, 9.26216515, 10.95117648, 7.64664774, 9.26343679,
9.24247868, 8.72541669, 8.42067421, 11.30720778, 10.19609633,
8.51792098, 13.03157897, 8.10294818, 9.67571952, 10.06232263,
8.75697612, 8.05067885, 7.67606892, 7.37637401, 8.98490732,
10.67321551, 12.18030627, 9.19320718, 8.25611386, 9.18938137,
9.80104014, 10.7614641 , 14.97174085, 9.90863455, 8.49861268,
12.80830169, 9.12888331, 13.22419566, 9.38141091, 9.03719535,
11.28676378, 9.57368493, 14.15810132, 13.15299064, 13.64289594,
9.14271472, 12.52896848, 9.85253211, 10.90763996, 11.91247162,
6.9643063 , 7.04873926, 13.51089032, 8.52621657, 11.92054006,
12.85548554, 9.35432381, 6.75579742, 6.92536014, 11.06993434,
5.09516196, 11.14055981, 10.02254804, 11.46143106, 11.6157383 ,
10.70537981, 11.34252457, 10.32964891, 6.40872558, 9.5230551 ,
9.8754752 , 6.64957542, 11.08423954, 6.27526737, 8.93050687,
7.0837962 , 10.98569946, 9.07710537, 6.54388388, 6.50684135,
13.92665521, 10.73419412, 9.62810419, 9.81485944, 14.28187178,
9.74851442, 7.62655355, 8.06624002, 9.51162406, 9.37527001,
6.28845065, 9.05904232, 12.31612411, 15.07632548, 11.07945243,
8.14721185, 8.86368526, 10.75579173, 6.46670957, 11.46459375,
7.89802408, 10.07334973, 7.59767402, 9.94794478, 7.25564624,
11.34876863, 9.26227404, 7.63153484, 8.78362146, 16.25490427,
7.57095315, 13.33206439, 9.76916846, 5.65234174, 9.98095345,
5.76281777, 5.75660541, 10.65527206, 9.52082376, 9.3706415 ,
11.20512028, 7.20216322, 13.40362951, 12.6176809 , 14.53097291,
7.3174256 , 11.16593252, 12.91316997, 7.21956333, 5.52607635,
10.70583405, 9.26550355, 11.67839413, 8.83346747, 9.35654145,
11.97737999, 10.96765731, 7.08959727, 11.88322075, 11.06567064,
9.14221778, 10.10220632, 10.70128798, 6.76121354, 7.20030452,
9.38367229, 7.60807592, 11.11285058, 7.55727594, 11.97387227,
10.70204789, 10.6971933 , 8.84485399, 9.69472299, 10.25602577,

```

14.82340733, 10.38255133, 9.34222692, 9.31530758, 9.13635225,
11.96187819, 6.01899495, 11.05520058, 8.73866757, 7.23769593,
10.9548119 , 10.0750424 , 10.74864677, 8.25101715, 9.48470442,
5.00488978, 6.63414798, 6.53765429, 9.08694898, 8.49684584,
10.89150004, 7.36918906, 9.06168902, 9.62601824, 14.60100849,
8.75334046, 11.68484866, 13.58737376, 9.39286917, 11.51447553,
8.715134 , 13.98844699, 4.86099936, 11.82027244, 10.50426586,
10.45477875, 7.12948081, 9.17697908, 8.66316832, 11.29164479,
15.08124346, 11.80449459, 10.35175054, 10.5595011 , 9.76572868,
9.65425563, 9.12745562, 11.3516656 , 15.99314921, 5.89453211,
10.87219476, 8.92167285, 10.87637108, 13.90625374, 10.72846676,
10.04208441, 10.24592078, 9.80027847, 13.27309825, 12.05243747,
10.63385363, 13.083792 , 7.60719341, 8.99745186, 13.20899991,
12.5576303 , 11.40667332, 11.23953261, 17.23896893, 7.8531285 ,
8.191757 , 9.48583202, 15.80286428, 11.94151263, 8.67917291,
7.38030287, 10.56161759, 9.5067371 , 10.90686841, 9.52354238,
10.54064953, 12.76740763, 8.52309521, 2.53887209, 10.16614663,
7.92323818, 9.19001237, 8.68823114, 8.63952788, 8.6022473 ,
12.43962035, 7.73106395, 6.83365054, 14.94736901, 8.61041768,
10.17368467, 12.07550847, 10.94804622, 8.71201489, 7.51064335,
11.60606233, 10.05915732, 7.69003357, 10.48004644, 10.8305406 ,
7.22124017, 12.05438669, 8.47989195, 14.50815848, 6.31579116,
9.10492219, 10.63327037, 7.24350042, 11.10935413, 12.34589553,
6.37506037, 5.24088619, 10.72498267, 7.9826483 , 7.85438097,
15.50153207, 8.46753475, 8.42673243, 9.36593052, 9.47760882,
10.90846718, 9.48600976, 11.37847124, 6.34064151, 12.22938785,
9.44902029, 16.78152113, 12.63696271, 6.47978491, 11.74947815,
9.10400676, 14.6031103 , 16.66448936, 8.64201185, 12.33070086,
12.51770165, 11.29289171, 10.53877233, 12.81078631, 11.8829119 ,
10.06782342, 7.75963449, 10.65794249, 4.91459767, 9.56125876,
10.68400691, 7.92487953, 8.61887237, 8.37453113, 10.68875889,
7.62101595, 5.40351361, 6.13674007, 8.55510195, 8.26475414,
11.17925161, 9.78725504, 12.07803016, 7.56853355, 6.76528288,
12.84633503, 12.42805699, 11.76702009, 8.48417017, 13.91588076,
9.28636427, 9.17181811, 12.22486462, 10.46930302, 10.88293772,
8.06717046, 8.04262297, 9.49841394, 9.72765238, 11.91317785,
9.48780322, 12.57409792, 8.56749496, 12.93945663, 8.40072518,
9.26340918, 7.15536791, 7.33131294, 9.40044306, 13.30425143,
7.52907068, 12.11786607, 9.44871531, 12.64268462, 7.73734429,
10.80140706])

```

14. Measure the performance using Metrics

In []:

```

from sklearn.metrics import mean_absolute_error, mean_squared_error s =
mean_squared_error(y_train, y_train_pred) print('Mean Squared error of
training set :%2f'%s)

```

In []:

```

p = mean_squared_error(y_test, y_test_pred) print('Mean Squared error of
testing set :%2f'%p)

```

Mean Squared error of testing set :4.869245

In []:

```

# Build the Model
from sklearn.ensemble import RandomForestRegressor

```

```

rfr = RandomForestRegressor(max_depth=2, random_state=0,
n_estimators=100)
```

```

rfr.fit(x_train, y_train) rfr.fit(x_test, y_test)
```

```

RandomForestRegressor(max_depth=2, random_state=0)
```

```

#Test the model y_train_pred =
rfr.predict(x_train) y_test_pred =
rfr.predict(x_test)
```

```

#measure the performance using metrics rfr.score(x_test,
y_test)
```

```

0.4497726034378102
```

K Neighbors Regression

```

#Build the model
from sklearn.neighbors import KNeighborsRegressor
```

```

knr = KNeighborsRegressor(n_neighbors =4 )
```

```

#Train the model knr.fit(x_train,
y_train) knr.fit(x_test, y_test)
```

```

KNeighborsRegressor(n_neighbors=4)
```

```

#Test the model y_train_pred =
knr.predict(x_train) y_test_pred =
knr.predict(x_test)
```

```

#Measure the performance using Metrics
knr.score(x_train, y_train)
```

```

0.458628955466746
```

Decision Tree Regression

```

#Build the model
from sklearn.tree import DecisionTreeRegressor
```

```

dtr = DecisionTreeRegressor(random_state=0)
```

```

#Train the model dtr.fit(x_test,y_test)
```

```

DecisionTreeRegressor(random_state=0)
```

Test the model

```
y_train_pred = dtr.predict(x_train) y_test_pred = dtr.predict(x_test)
```

In []:

```
#Measure the performance using Metrics dtr.score(x_train,  
y_train)
```

Out[]:

```
0.15715160117393523
```

Lasso Regression

In [214]:

```
#Build the model  
from sklearn.linear_model import Lasso
```

In [215]:

```
lr=Lasso(alpha=0.01)
```

In [216]:

```
#Train the model lr.fit(x_train,y_train)
```

Out[216]:

```
Lasso(alpha=0.01)
```

In [217]:

```
y_train_pred = lr.predict(x_train) y_test_pred = lr.predict(x_test)
```

In [218]:

```
#Measure the performance using Metrics lr.score(x_train,  
y_train)
```

Out[218]: 0.5098141532900928