# Assignment: 04

**STUDENT NAME**: Karishma O

**BATCH NO**:09

**STUDENT ROLL NO**:921319104088

**TEAM ID:**PNT2022TMID04978

**PROJECT NAME:** Retail Store Inventory

# Problem Statement: Abalone Age Prediction

Description:- Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings througha microscope a boring and time-consuming task. Other measurements, which are easier to obtain,are used to predict age. Further information, such as weather patterns and location (hence foodavailability) may be required to solve the problem.

## 1. Download the dataset

Data set link: [abalone](https://drive.google.com/file/d/1sIv-7x7CE0zAPAt0Uv-6pbO2ST2LVp5u/view)

**2. Loading the dataset** import numpy as np import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline import seaborn as sns

*#Load the dataset*

df=pd.read\_csv('"C:\Users\Kiruthika\Downloads\abalone.csv"') df.head()

Sex Length Diameter Height Whole weight Shucked weight Viscera weight \

1. 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

1. F 0.530 0.420 0.135 0.6770 0.2565

0.1415

1. M 0.440 0.365 0.125 0.5160 0.2155

0.1140

1. I 0.330 0.255 0.080 0.2050 0.0895

0.0395

Shell weight Rings

1. 0.150 15
2. 0.070 7
3. 0.210 9
4. 0.155 10
5. 0.055 7

**3. Perform Visualizations.**

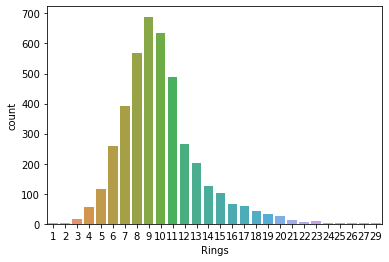
1. Univariate Analysis
2. Bi-Variate Analysis
3. Multi-Variate Analysis

## 1. Univariate Analysis

*# countplot*

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

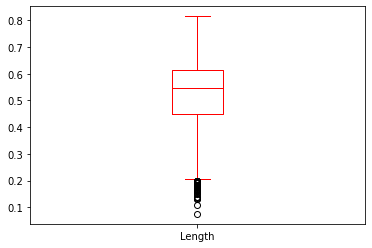
<AxesSubplot: xlabel='Rings', ylabel='count'>



*#boxplot*

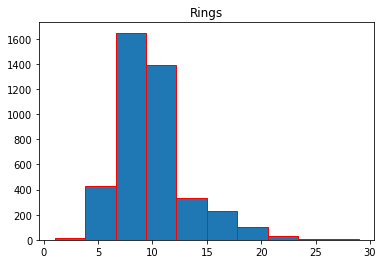
df.boxplot(column=['Length'], grid=False, color='Red')

<AxesSubplot: >



*#histogram*

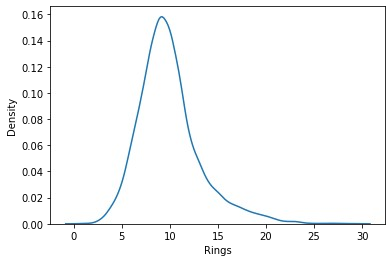
df.hist(column='Rings', grid=False, edgecolor='Red') array([[<AxesSubplot: title={'center': 'Rings'}>]], dtype=object)



*#kdeplot*

sns.kdeplot(df['Rings'])

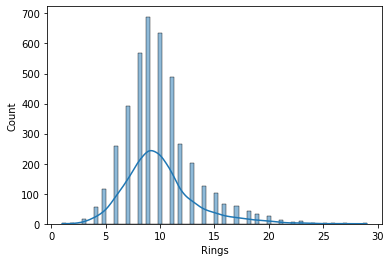
<AxesSubplot: xlabel='Rings', ylabel='Density'>



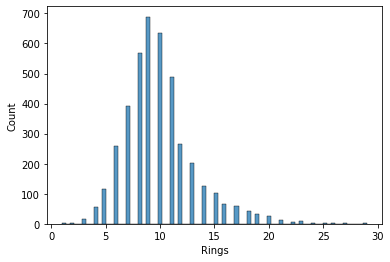
*#histplot*

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

<AxesSubplot: xlabel='Rings', ylabel='Count'>

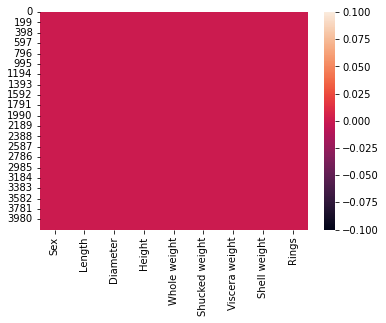


*#histplot* sns.histplot(df['Rings']) <AxesSubplot: xlabel='Rings', ylabel='Count'>



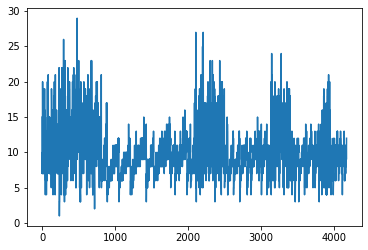
*#heatmap* sns.heatmap(df.isnull())

<AxesSubplot: >



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

[<matplotlib.lines.Line2D at 0x12a04bb20>]



*#piechart* plt.pie(df['Rings'].head(),autopct='%.1f')

([<matplotlib.patches.Wedge at 0x12a0d0b80>,

<matplotlib.patches.Wedge at 0x12a0d1090>,

<matplotlib.patches.Wedge at 0x12a0d1780>,

<matplotlib.patches.Wedge at 0x12a0d1e10>,

<matplotlib.patches.Wedge at 0x12a0d24a0>],

[Text(0.6111272563215626, 0.9146165735327997, ''), Text(-0.8270237769092664, 0.725280409515335, ''),

Text(-1.041623153479572, -0.35358337932554523, ''),

Text(-5.149471704824549e-08, -1.0999999999999988, ''),

Text(0.9865599777267362, -0.4865176362145796, '')],

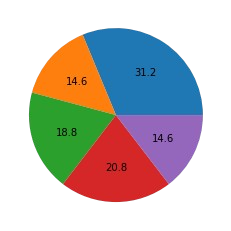
[Text(0.33334213981176136, 0.49888176738152706, '31.2'),

Text(-0.4511038783141453, 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')])

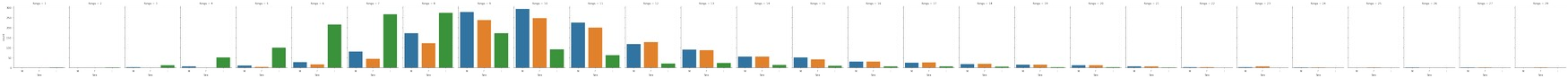


# 2. Bi-Variate Analysis

*#countplot*

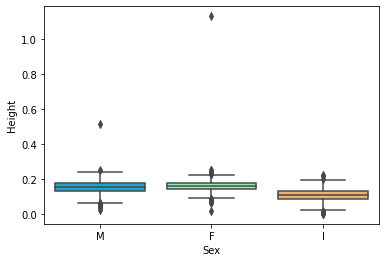
sns.catplot(x="Sex",col="Rings",data=df, kind="count",height=4, aspect=.7)

<seaborn.axisgrid.FacetGrid at 0x129df3850>



*#boxplot*

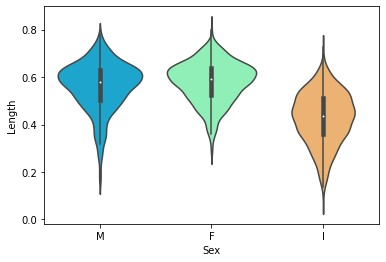
sns.boxplot(x='Sex',y='Height',data=df,palette='rainbow') <AxesSubplot: xlabel='Sex', ylabel='Height'>



*#violin plot*

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

<AxesSubplot: xlabel='Sex', ylabel='Length'>



*#strip plot*

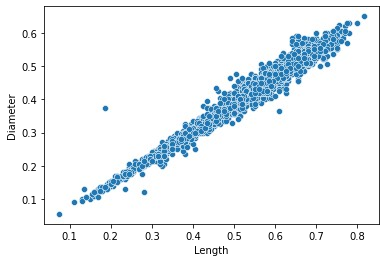
sns.stripplot(x="Sex", y="Length", data=df)

<AxesSubplot: xlabel='Sex', ylabel='Length'>



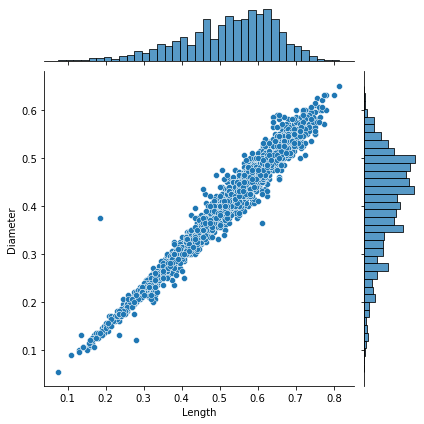
*#scatter plot*

sns.scatterplot(x =df["Length"],y =df["Diameter"]) <AxesSubplot: xlabel='Length', ylabel='Diameter'>



*#joint\_plot*

sns.jointplot(x="Length",y="Diameter",data=df) <seaborn.axisgrid.JointGrid at 0x12ad280d0>

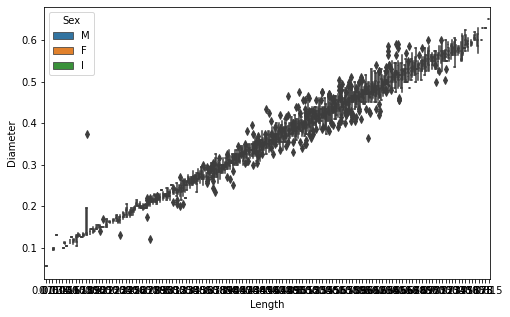


# 3. Multi-Variate Analysis

*#boxplot*

fig, ax1 =plt.subplots(figsize=(8,5))

testPlot=sns.boxplot(ax=ax1, x='Length', y='Diameter', hue='Sex', data=df)

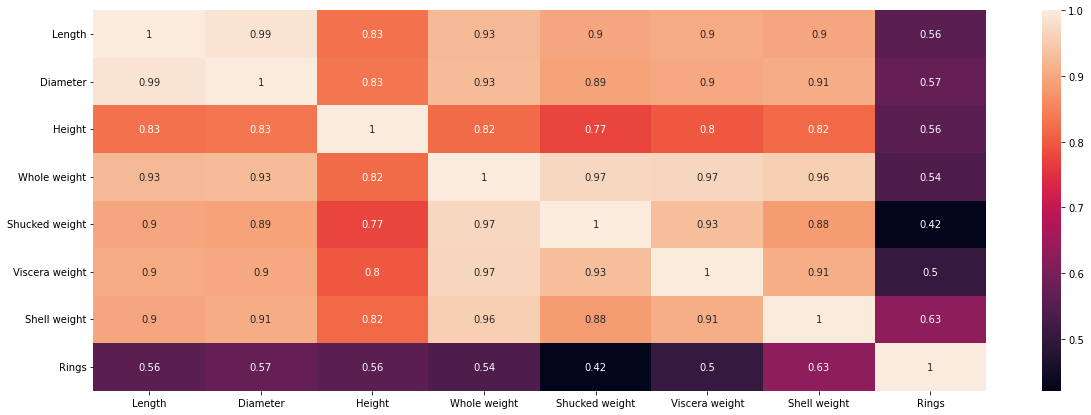


numerical\_features=df.select\_dtypes(include = [np.number]).columns categorical\_features=df.select\_dtypes(include = [object]).columns

plt.figure(figsize= (20,7))

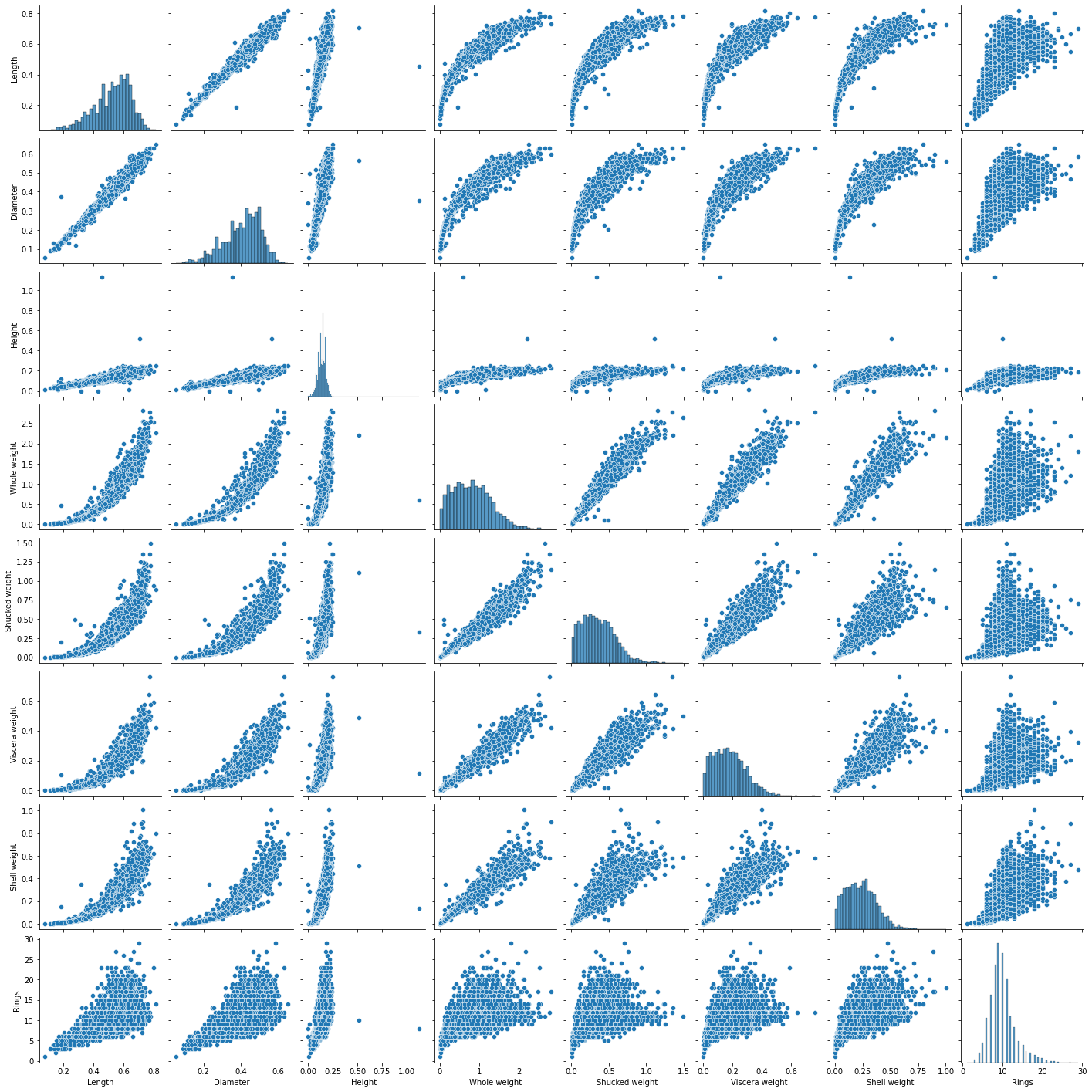
sns.heatmap(df[numerical\_features].corr(),annot=True)

<AxesSubplot: >



sns.pairplot(df)

<seaborn.axisgrid.PairGrid at 0x12c97a290>



**4. Perform descriptive statistics on the dataset.**

df

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

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

1. F 0.565 0.450 0.165 0.8870 0.3700
2. M 0.590 0.440 0.135 0.9660 0.4390
3. M 0.600 0.475 0.205 1.1760 0.5255
4. F 0.625 0.485 0.150 1.0945 0.5310
5. M 0.710 0.555 0.195 1.9485 0.9455 Viscera weight Shell weight Rings 0 0.1010 0.1500 15
6. 0.0485 0.0700 7
7. 0.1415 0.2100 9
8. 0.1140 0.1550 10
9. 0.0395 0.0550 7 ... ... ... ... 4172 0.2390 0.2490 11
10. 0.2145 0.2605 10
11. 0.2875 0.3080 9
12. 0.2610 0.2960 10
13. 0.3765 0.4950 12

[4177 rows x 9 columns] df.head()

Sex Length Diameter Height Whole weight Shucked weight Viscera weight \

1. 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

1. F 0.530 0.420 0.135 0.6770 0.2565

0.1415

1. M 0.440 0.365 0.125 0.5160 0.2155

0.1140

1. I 0.330 0.255 0.080 0.2050 0.0895

0.0395

Shell weight Rings

1. 0.150 15
2. 0.070 7
3. 0.210 9
4. 0.155 10 4 0.055 7 df.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 df.describe()

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 df[numerical\_features].mean()

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

df[numerical\_features].median()

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

df[numerical\_features].value\_counts()

Length Diameter Height Whole weight Shucked weight Viscera weight

Shell weight Rings

0.075 0.055 0.010 0.0020 0.0010 0.0005

0.0015 1 1

0.590 0.465 0.155 1.1360 0.5245 0.2615

0.2750 11 1

0.165 1.1150 0.5165 0.2730

0.2750 10 1

0.170 1.0425 0.4635 0.2400

0.2700 10 1

0.195 1.0885 0.3685 0.1870

0.3750 17 1

..

0.485 0.370 0.155 0.9680 0.4190 0.2455

0.2365 9 1

0.375 0.110 0.4640 0.2015 0.0900

0.1490 8 1

0.125 0.5620 0.2505 0.1345

0.1525 8 1

0.130 0.5535 0.2660 0.1120

0.1570 8 1

0.815 0.650 0.250 2.2550 0.8905 0.4200

0.7975 14 1 Length: 4177, dtype: int64 df[numerical\_features].mode()

Length Diameter Height Whole weight Shucked weight Viscera weight \

1. 0.550 0.45 0.15 0.2225 0.175 0.1715
2. 0.625 NaNNaNNaNNaNNaN

Shell weight Rings

1. 0.275 9.0
2. NaNNaN

df[numerical\_features].std()

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

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

[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] 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()

Sex Length Diameter Height Whole weight Shucked weight \

1. False FalseFalseFalseFalseFalse
2. False FalseFalseFalseFalseFalse
3. False FalseFalseFalseFalseFalse
4. False FalseFalseFalseFalseFalse
5. False FalseFalseFalseFalseFalse ... ... ... ... ... ... ... 4172 FalseFalseFalseFalseFalseFalse
6. False FalseFalseFalseFalseFalse
7. False FalseFalseFalseFalseFalse
8. False FalseFalseFalseFalseFalse
9. False FalseFalseFalseFalseFalse

Viscera weight Shell weight Rings 0 False FalseFalse

1. False FalseFalse
2. False FalseFalse
3. False FalseFalse
4. False FalseFalse ... ... ... ... 4172 False FalseFalse
5. False FalseFalse
6. False FalseFalse
7. False FalseFalse
8. False FalseFalse

[4177 rows x 9 columns] df.shape (4177, 9) df.isnull().any()

Sex False Length False Diameter False Height False

Whole weight False

Shucked weight False

Viscera weight False

Shell weight False Rings False dtype: bool df.isnull().sum()

Sex 0 Length 0 Diameter 0 Height 0

Whole weight 0

Shucked weight 0

Viscera weight 0

Shell weight 0 Rings 0 dtype: int64

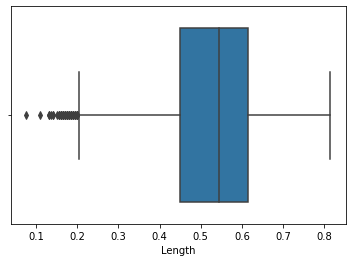
# 6. Find the outliers and replace them outliers

*#length*

sns.boxplot(df['Length'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Length'>



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

(0.45, 0.615, 0.16499999999999998)

upper\_limit= q2 + (1.5\*iqr) lower\_limit= q1 - (1.5\*iqr) lower\_limit, upper\_limit

(0.20250000000000004, 0.8624999999999999)

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))

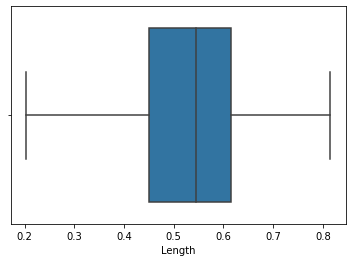
before removing outliers: 4177 after removing outliers: 4128 outliers: 49

new\_df=df.copy()

new\_df.loc[(new\_df['Length']>upper\_limit), 'Length'] =upper\_limitnew\_df.loc[(new\_df['Length']<lower\_limit), 'Length'] =lower\_limitsns.boxplot(new\_df['Length'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Length'>

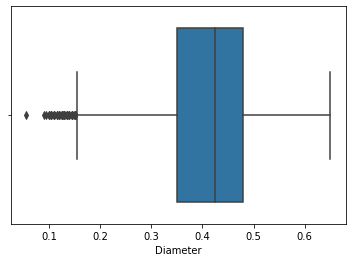


*#Diameter*

sns.boxplot(df['Diameter'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Diameter'>



q1 =df['Diameter'].quantile(0.25) q2 =df['Diameter'].quantile(0.75) iqr= q2-q1 q1, q2, iqr (0.35, 0.48, 0.13)

upper\_limit= q2 + (1.5\*iqr) lower\_limit= q1 - (1.5\*iqr) lower\_limit, upper\_limit (0.15499999999999997, 0.675)

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))

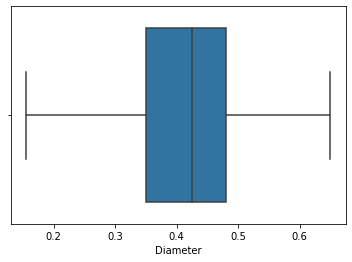
before removing outliers: 4177 after removing outliers: 4118 outliers: 59

new\_df=df.copy()

new\_df.loc[(new\_df['Diameter']>upper\_limit), 'Diameter'] =upper\_limitnew\_df.loc[(new\_df['Diameter']<lower\_limit), 'Diameter'] =lower\_limitsns.boxplot(new\_df['Diameter'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Diameter'>

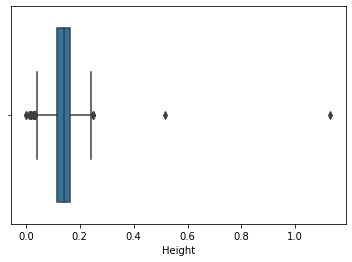


*#Height*

sns.boxplot(df['Height'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Height'>



q1 =df['Height'].quantile(0.25) q2 =df['Height'].quantile(0.75) iqr= q2-q1 q1, q2, iqr (0.115, 0.165, 0.05)

upper\_limit= q2 + (1.5\*iqr) lower\_limit= q1 - (1.5\*iqr) lower\_limit, upper\_limit

(0.039999999999999994, 0.24000000000000002)

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))

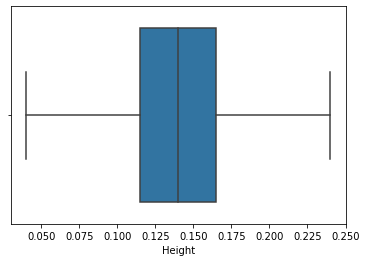
before removing outliers: 4177 after removing outliers: 4148 outliers: 29

new\_df=df.copy()

new\_df.loc[(new\_df['Height']>upper\_limit), 'Height'] =upper\_limitnew\_df.loc[(new\_df['Height']<lower\_limit), 'Height'] =lower\_limitsns.boxplot(new\_df['Height'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Height'>

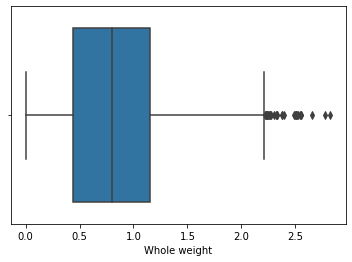


*#Whole Weight*

sns.boxplot(df['Whole weight'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Whole weight'>



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

(0.4415, 1.153, 0.7115)

upper\_limit= q2 + (1.5\*iqr) lower\_limit= q1 - (1.5\*iqr) lower\_limit, upper\_limit (-0.62575, 2.22025)

new\_df=df.loc[(df['Whole weight'] <=upper\_limit) & (df['Whole weight'] >=lower\_limit)]

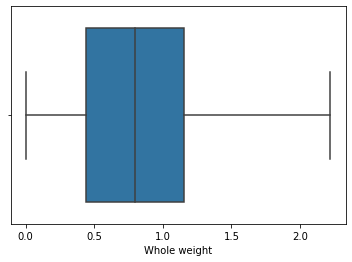
print('before removing outliers:', len(df)) print('after removing outliers:',len(new\_df)) print('outliers:', len(df)-len(new\_df))

before removing outliers: 4177 after removing outliers: 4147 outliers: 30

'Whole weight']>upper\_limit), 'Whole weight'] ='Whole weight']<lower\_limit), 'Whole weight'] =

'Whole weight'])

<AxesSubplot: xlabel='Whole weight'>

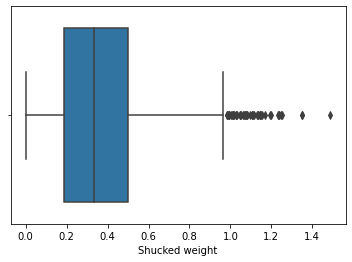


*#Shucked weight*

sns.boxplot(df['Shucked weight'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Shucked weight'>



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

(0.186, 0.502, 0.316)

upper\_limit= q2 + (1.5\*iqr) lower\_limit= q1 - (1.5\*iqr) lower\_limit, upper\_limit (-0.288, 0.976)

new\_df=df.loc[(df['Shucked weight'] <=upper\_limit) & (df['Shucked weight'] >=lower\_limit)]

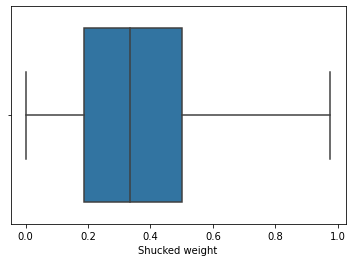
print('before removing outliers:', len(df)) print('after removing outliers:',len(new\_df)) print('outliers:', len(df)-len(new\_df))

before removing outliers: 4177 after removing outliers: 4129 outliers: 48

'Shucked weight']>upper\_limit), 'Shucked weight'] = 'Shucked weight']<lower\_limit), 'Shucked weight'] =

'Shucked weight'])

<AxesSubplot: xlabel='Shucked weight'>

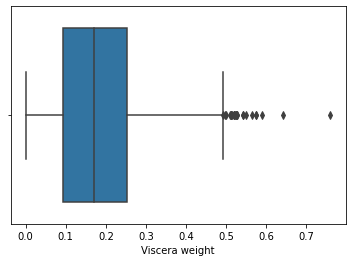


*#Viscera weight*

sns.boxplot(df['Viscera weight'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Viscera weight'>



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

(0.0935, 0.253, 0.1595)

upper\_limit= q2 + (1.5\*iqr) lower\_limit= q1 - (1.5\*iqr) lower\_limit, upper\_limit (-0.14575000000000002, 0.49225)

new\_df=df.loc[(df['Viscera weight'] <=upper\_limit) & (df['Viscera weight'] >=lower\_limit)]

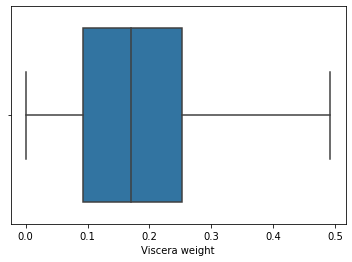
print('before removing outliers:', len(df)) print('after removing outliers:',len(new\_df)) print('outliers:', len(df)-len(new\_df))

before removing outliers: 4177 after removing outliers: 4151 outliers: 26

'Viscera weight']>upper\_limit), 'Viscera weight'] = 'Viscera weight']<lower\_limit), 'Viscera weight'] =

'Viscera weight'])

<AxesSubplot: xlabel='Viscera weight'>

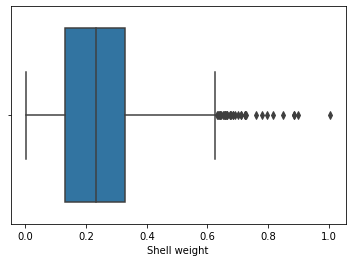


*#shell weight*

sns.boxplot(df['Shell weight'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Shell weight'>



q1 =df['Shell weight'].quantile(0.25) q2 =df['Shell weight'].quantile(0.75) iqr= q2-q1 q1, q2, iqr (0.13, 0.329, 0.199)

upper\_limit= q2 + (1.5\*iqr) lower\_limit= q1 - (1.5\*iqr) lower\_limit, upper\_limit (-0.16849999999999998, 0.6275)

new\_df=df.loc[(df['Shell weight'] <=upper\_limit) & (df['Shell weight'] >=lower\_limit)]

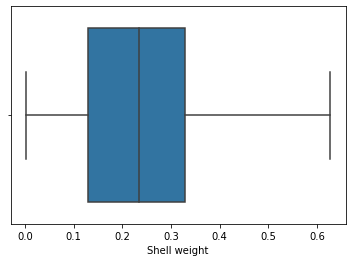
print('before removing outliers:', len(df)) print('after removing outliers:',len(new\_df)) print('outliers:', len(df)-len(new\_df))

before removing outliers: 4177 after removing outliers: 4142 outliers: 35

'Shell weight']>upper\_limit), 'Shell weight'] ='Shell weight']<lower\_limit), 'Shell weight'] =

'Shell weight'])

<AxesSubplot: xlabel='Shell weight'>

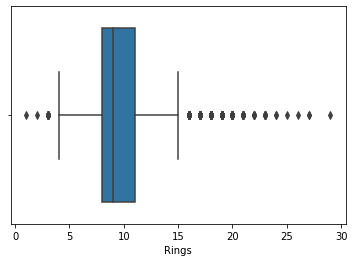


*#Rings*

sns.boxplot(df['Rings'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Rings'>



q1 =df['Rings'].quantile(0.25) q2 =df['Rings'].quantile(0.75) iqr= q2-q1 q1, q2, iqr (8.0, 11.0, 3.0)

upper\_limit= q2 + (1.5\*iqr) lower\_limit= q1 - (1.5\*iqr) lower\_limit, upper\_limit

(3.5, 15.5)

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))

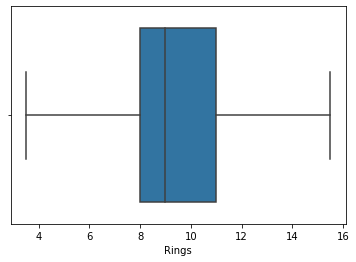
before removing outliers: 4177 after removing outliers: 3899 outliers: 278

new\_df=df.copy()

new\_df.loc[(new\_df['Rings']>upper\_limit), 'Rings'] =upper\_limitnew\_df.loc[(new\_df['Rings']<lower\_limit), 'Rings'] =lower\_limitsns.boxplot(new\_df['Rings'])

/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/ 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(

<AxesSubplot: xlabel='Rings'>



**7. Check for Categorical columns and perform encoding.**

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

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. 0 0.530 0.420 0.135 0.6770 0.2565
3. 1 0.440 0.365 0.125 0.5160 0.2155
4. 2 0.330 0.255 0.080 0.2050 0.0895 ... ... ... ... ... ... ... 4172 0 0.565 0.450 0.165 0.8870 0.3700
5. 1 0.590 0.440 0.135 0.9660 0.4390
6. 1 0.600 0.475 0.205 1.1760 0.5255
7. 0 0.625 0.485 0.150 1.0945 0.5310
8. 1 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
5. 0.2145 0.2605 10
6. 0.2875 0.3080 9
7. 0.2610 0.2960 10
8. 0.3765 0.4950 12

[4177 rows x 9 columns]

from sklearn.preprocessing import LabelEncoder,OneHotEncoder,StandardScaler

label\_encoder=LabelEncoder()

df['Sex']=label\_encoder.fit\_transform(df['Sex']) df

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. 0 0.530 0.420 0.135 0.6770 0.2565
3. 1 0.440 0.365 0.125 0.5160 0.2155
4. 2 0.330 0.255 0.080 0.2050 0.0895 ... ... ... ... ... ... ... 4172 0 0.565 0.450 0.165 0.8870 0.3700
5. 1 0.590 0.440 0.135 0.9660 0.4390
6. 1 0.600 0.475 0.205 1.1760 0.5255
7. 0 0.625 0.485 0.150 1.0945 0.5310
8. 1 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
5. 0.2145 0.2605 10
6. 0.2875 0.3080 9
7. 0.2610 0.2960 10
8. 0.3765 0.4950 12

[4177 rows x 9 columns] enc =OneHotEncoder(drop='first') enc\_df=pd.DataFrame(enc.fit\_transform(df[['Sex']]).toarray()) df=df.join(enc\_df) df.head()

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. 0 0.530 0.420 0.135 0.6770 0.2565
3. 1 0.440 0.365 0.125 0.5160 0.2155
4. 2 0.330 0.255 0.080 0.2050 0.0895

Viscera weight Shell weight Rings 0 1 0 0.1010 0.150 15 1.0 0.0

1. 0.0485 0.070 7 1.0 0.0
2. 0.1415 0.210 9 0.0 0.0
3. 0.1140 0.155 10 1.0 0.0
4. 0.0395 0.055 7 0.0 1.0 **8. Split the data into dependent and independent variables.**

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

Length Diameter Height Whole weight Shucked weight Viscera weight \

1. 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

1. 0.530 0.420 0.135 0.6770 0.2565

0.1415

1. 0.440 0.365 0.125 0.5160 0.2155

0.1140

1. 0.330 0.255 0.080 0.2050 0.0895

0.0395

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

1. 0.565 0.450 0.165 0.8870 0.3700

0.2390

1. 0.590 0.440 0.135 0.9660 0.4390

0.2145

1. 0.600 0.475 0.205 1.1760 0.5255

0.2875

1. 0.625 0.485 0.150 1.0945 0.5310

0.2610

1. 0.710 0.555 0.195 1.9485 0.9455

0.3765

Shell weight

1. 0.1500
2. 0.0700
3. 0.2100
4. 0.1550
5. 0.0550 ... ... 4172 0.2490
6. 0.2605
7. 0.3080
8. 0.2960
9. 0.4950

[4177 rows x 7 columns]

y=df.iloc[:,8] y

0 15 1 7

2 9 3 10

4 7 ..

1. 11
2. 10 4174 9 4175 10

4176 12

Name: Rings, Length: 4177, dtype: int64 **9. Scale the independent variables** 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** from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test=train\_test\_split(x,y, test\_size=

0.2)

print(x.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

from sklearn.linear\_model import LinearRegressionlinearmodel=LinearRegression()

**12. Train the Model** linearmodel.fit(x\_train, y\_train)

LinearRegression() **13. Test the Model**

y\_train\_pred=linearmodel.predict(x\_train) y\_test\_pred=linearmodel.predict(x\_test) y\_test\_pred

array([ 9.47089211, 11.21761752, 12.45746154, 13.98475412,

11.63205706,

15.06852206, 11.28509789, 11.26611379, 10.478623 ,

9.54255691,

7.51806127, 7.89502413, 12.09714369, 7.01694889,

10.06344914,

7.22113973, 10.20974354, 12.05453296, 9.0630022 ,

6.78718126,

12.51004135, 12.1303277 , 10.23427085, 11.89467971,

9.88868997,

5.24327598, 9.0577592 , 12.90664974, 8.72754612,

13.11399541,

12.64202957, 12.92400961, 9.08706747, 12.45461196,

12.48634564,

10.07804014, 8.15574878, 10.68425155, 12.1020984 ,

6.1915207 ,

15.28797415, 10.29658112, 7.48991773, 20.53951935,

9.20518963,

12.5702666 , 10.28211228, 7.70483528, 9.28458802, 7.36047611,

8.10324679, 10.49662349, 11.41112739, 12.39594156,

8.08189753,

9.50162052, 9.3714665 , 12.23298058, 5.79118394,

13.78655815,

8.83881939, 10.06882824, 8.96482586, 6.77622691,

10.79485462,

9.13979642, 10.99012474, 9.16701246, 10.81862043,

7.56404085,

11.04745719, 10.4442233 , 8.35127257, 6.37009807,

10.18826912,

7.78642557, 9.08415208, 10.78255159, 13.72111937,

8.12140416,

8.31279753, 7.05604339, 12.31071417, 10.75239544,

11.59825028,

8.64683427, 7.65929047, 13.63529525, 8.07865886,

10.05455816,

9.39283204, 13.0189663 , 18.87898094, 8.78586552,

13.66491309,

10.50370946, 13.34022206, 7.54283174, 10.61600711,

7.50713307,

11.30096403, 9.0457258 , 6.67702671, 10.78458937,

6.54135176,

14.97611904, 7.29969348, 6.62461389, 13.07844976,

10.75691202,

9.21010972, 7.59935478, 13.03527214, 7.21049877,

8.58047994,

7.66895733, 8.53196107, 9.87084709, 8.13760243,

8.15111319,

11.00659114, 8.88769594, 5.25567913, 9.32059787,

8.47010909,

8.37825326, 8.63213494, 10.78447946, 8.73498862,

7.19639423,

11.29983658, 10.69329407, 8.7277503 , 9.23775692,

9.59204441,

5.91545727, 7.62019108, 8.46140939, 10.66534273,

6.29641466,

10.85414379, 8.87317836, 5.48747812, 11.7367539 ,

8.44541517,

9.91033873, 12.43326809, 11.02611969, 9.20558187,

12.45870849,

7.29384561, 6.33000643, 7.65858619, 10.18764679,

7.47816728,

8.45825764, 10.3117139 , 11.87728433, 11.0473399 ,

11.17474725,

7.85013858, 10.90262254, 9.81507119, 8.6042659 ,

10.78742379,

11.5973733 , 10.19487071, 8.36355166, 5.99634943,

10.86416144,

10.24752585, 11.80207144, 5.66035619, 10.1793873 , 16.43421175,

11.05022237, 12.2987536 , 9.48888781, 7.82783885,

9.40662503,

9.31973992, 10.22550972, 9.93989723, 11.73170818,

11.24552877,

8.14677228, 13.51851237, 9.57797989, 10.41739489,

6.39771924,

11.02329551, 8.62998271, 9.31347138, 13.14400156,

11.78684795,

10.77622753, 9.9488686 , 13.21995198, 6.83500562,

6.87708928,

9.29043964, 11.45428215, 9.34658903, 8.82665639,

8.9367785 ,

14.80395325, 11.8063895 , 10.27295828, 13.93703673,

9.52253126,

11.8577913 , 10.60958394, 8.58332241, 10.39675 ,

17.58505802,

6.25861316, 7.42245657, 7.76030375, 7.57137613,

8.91858079,

9.7380453 , 11.83929401, 13.06789847, 9.72482201,

8.42848118,

11.11610524, 9.86958653, 9.4131896 , 8.23090887,

9.83153404,

12.413822 , 9.31143396, 10.18573205, 10.34633553,

11.25645281,

9.47599684, 8.82081377, 9.42282538, 12.56733651,

9.07633939,

7.72005367, 16.965691 , 8.70158457, 13.37433829,

11.3533882 ,

7.56486806, 8.42948098, 14.04646224, 11.19511922,

11.29599298,

10.76926226, 10.59709887, 7.4304641 , 9.19831695,

9.6640906 ,

12.38453516, 8.68136607, 7.50167601, 10.17044267,

8.96489708,

9.71155464, 5.34824193, 7.92922826, 9.64047698,

7.83011443,

9.46358522, 11.27860973, 8.13247976, 11.05838039,

8.05015768,

10.50177439, 7.70220884, 12.33984129, 5.17176515,

7.13845794,

12.87948503, 13.3744418 , 6.73738104, 5.52907695,

10.12242977,

7.89936832, 9.40044464, 7.4036428 , 12.69685458,

12.90070308,

8.79763903, 7.23831657, 6.9893129 , 10.02829136,

9.20175443,

9.59196775, 9.79103472, 11.38883663, 12.1957921 ,

9.34988884,

10.14343928, 9.41681938, 7.05131894, 13.10795052, 10.1315401 ,

10.24224026, 6.9409188 , 12.40875052, 7.38668254,

11.69855438,

9.31817821, 12.19296756, 9.079605 , 8.4540433 ,

4.49384329,

5.78757926, 7.50457939, 8.37742322, 10.04974345,

12.39639756,

10.08677028, 10.94880527, 9.33665567, 15.32450072,

6.17543896,

11.45626028, 9.30746316, 11.0050131 , 10.4651411 ,

10.70093879,

5.10979224, 9.14257081, 11.21975354, 9.83750409,

9.79912476,

7.71146613, 5.33702132, 9.69587138, 8.05466991,

6.79799963,

8.72884406, 12.56424859, 16.8570055 , 11.33010552,

9.79402352,

7.13475673, 11.59907561, 7.51361365, 7.24763752,

9.86913934,

12.91579197, 10.51246866, 9.18447239, 10.32669974,

10.87136275,

11.56608868, 8.67180517, 7.08106214, 14.52757786,

7.90627503,

9.92844006, 11.08671104, 10.95125646, 8.92218286,

11.63264258,

9.04689851, 10.70118933, 7.7063996 , 8.93253215,

11.25237885,

10.03559659, 7.81781351, 10.9318696 , 11.63248986,

10.1439026 ,

7.25076743, 5.4576721 , 12.38369861, 11.29693124,

10.45808148,

8.50526188, 12.87958889, 6.53551463, 10.72878317,

10.05923938,

9.13440282, 8.84724635, 7.69073338, 8.80995922,

13.21042855,

10.2487618 , 10.61404565, 8.83898392, 7.25110201,

10.48011159,

10.59906963, 15.68205081, 8.09294038, 9.80938001,

11.00145096,

12.1583954 , 8.42453402, 9.3123793 , 11.14804446,

15.98092246,

8.85994567, 8.82219643, 15.93729164, 8.83482954,

10.77577904,

14.40167534, 8.65336523, 15.58566757, 7.61930991,

12.58608177,

10.38010051, 7.07535296, 10.44305474, 6.14659107,

7.231232 ,

7.39307045, 9.07097932, 10.34982481, 8.74800324,

10.78043109,

10.18960188, 11.49784145, 4.97373256, 9.57000006, 10.06381202,

8.46933175, 10.31164919, 10.40408677, 9.67252184,

6.09977546,

7.65417737, 11.49765082, 10.29306777, 10.96453093,

7.22745744,

8.41274997, 17.38164233, 12.50233778, 12.10904078,

8.92767174,

8.54590195, 12.73432169, 7.84474796, 7.57954376,

12.18211771,

10.33878652, 11.90813108, 9.33992114, 7.56691557,

8.00997981,

10.12151393, 9.2084682 , 8.82039 , 9.63938474,

11.93871372,

7.81691717, 9.25984125, 7.74340654, 11.43228783,

8.0585123 ,

8.63778265, 8.46315526, 9.85530938, 8.86588842,

8.92382498,

6.92447417, 13.78076934, 12.13309397, 8.87582642,

5.19291986,

9.01173165, 9.93968809, 11.80336874, 11.29051863,

10.85700288,

9.28057209, 10.12905656, 9.30708367, 8.56678609,

10.87091367,

10.55395861, 6.41544068, 9.96085089, 9.92043219,

9.92476448,

10.61698454, 10.82817573, 9.37738646, 12.56217709,

13.79954677,

10.72922022, 10.93695107, 9.26441584, 12.35101924,

8.83579381,

13.99188855, 8.79464613, 10.31897205, 9.38211873,

13.17622984,

10.51387315, 9.97147732, 7.24660484, 11.482972 ,

8.14209735,

9.28107877, 8.49220466, 6.94117866, 5.10898516,

10.38009594,

10.02777226, 8.03243799, 5.98791187, 12.805406 ,

12.19302904,

14.29612103, 7.12368777, 9.81630291, 14.38644246,

8.75981531,

9.15213739, 8.7270633 , 13.89135822, 10.10486528,

10.74955375,

6.98953457, 10.68529115, 10.56764732, 10.52894541,

8.62079692,

8.54339975, 8.92527775, 9.06274977, 11.32921967,

9.27939503,

9.31449004, 10.91359603, 11.61683081, 14.91175937,

9.84124369,

6.73230997, 12.86594861, 9.56801822, 8.40218641,

10.58796996,

11.34244412, 12.40146002, 6.53019187, 8.98600686, 6.8992399 ,

7.21182213, 6.38539517, 7.92352508, 8.86956192,

11.53888805,

8.60140616, 5.90459251, 10.66178744, 9.96969262,

9.7482581 ,

10.53936162, 8.20601253, 10.2345791 , 6.69289025,

10.67398901,

10.931215 , 5.8865841 , 8.2526372 , 12.17335418,

7.00812114,

7.74232498, 8.96812632, 9.89283073, 7.46012713,

9.67398753,

9.77950524, 9.51000752, 13.17109924, 14.56423553,

11.69019201,

6.33672744, 5.87059279, 9.83656189, 6.43206964,

5.43035662,

6.18773777, 10.7370586 , 10.86968908, 5.54168742,

10.80290679,

8.82641387, 10.72085629, 12.37262003, 8.60241474,

12.2039746 ,

11.9973331 , 10.79214342, 12.10242814, 13.19886559,

8.94256994,

10.18768992, 13.17326324, 10.93149331, 6.29099388,

11.22478803,

10.10871546, 11.80095142, 14.41565468, 9.13449064,

12.24949617,

11.66760468, 9.91274238, 10.33603778, 10.32691333,

6.59015518,

10.21042869, 9.97631227, 7.47446004, 7.27693069,

13.68496923,

7.88543458, 12.0109855 , 9.68640693, 7.71817341,

8.7225773 ,

8.39141268, 7.87576801, 6.8912282 , 13.51822253,

12.22908549,

9.65272115, 11.42431543, 9.28351175, 9.51783881,

8.98320483,

10.33165813, 11.04338744, 12.06641403, 10.47986743,

8.15814354,

14.17828525, 9.48474158, 10.62445119, 10.77125158,

8.99791232,

7.44696096, 10.59911987, 8.73005498, 7.47086785,

9.42779714,

10.22411029, 11.37479164, 11.10576208, 8.31614191,

12.77741904,

9.02295626, 11.08136402, 9.98725469, 10.12718192,

7.49626494,

18.07886078, 7.96862931, 6.6176935 , 5.90074735,

9.37264624,

7.67694195, 11.40662682, 10.60774466, 20.90436471,

10.81936541,

10.23418575, 7.94977428, 9.35604065, 9.58406053, 14.3079444 ,

8.98357669, 18.20049048, 11.11996036, 8.80983361,

8.21662339,

7.64139708, 11.77021183, 7.98288456, 9.8881865 ,

7.15992031,

11.0385808 , 8.13592452, 10.34984281, 9.90412942,

13.98484024,

8.22090577, 7.03000865, 8.25436555, 10.10939119,

11.5686925 ,

8.51703842, 10.93399137, 10.83109454, 12.74512244,

7.83042841,

8.11262693, 13.5978928 , 9.10989043, 7.16323107,

8.95073982,

7.47677887, 8.37251111, 7.86866912, 10.91328935,

6.62946909,

11.45327091, 11.24301157, 11.45044931, 9.30705273,

8.28750292,

10.43753088, 9.18094447, 7.51563017, 10.11168315,

10.12897077,

7.13772736, 12.18964556, 7.44576755, 12.11511597,

13.15948746,

9.07996498, 7.40926099, 9.3843992 , 10.60908287,

13.44706066,

10.52298199, 7.72647256, 8.34151096, 12.74344249,

8.14229844,

11.15985971, 10.18733866, 6.3535048 , 9.38935602,

7.57984435,

9.68865206, 8.32618096, 12.41753787, 6.49327307,

7.27118609,

10.06064498, 12.5045186 , 7.43840021, 9.91393225,

6.33296049,

7.30111054, 12.74145358, 8.29077959, 8.54825484,

18.67890375,

7.77104979, 5.92982385, 11.13669645, 10.35307734,

10.17614447,

11.78064362, 9.36936526, 7.07983165, 11.98251823,

6.03624751,

6.34160809, 7.51393039, 6.96474143, 15.35557452,

8.9811335 ,

12.55741663, 13.63295629, 8.21396573, 10.60416458,

8.74242956,

9.59644707, 10.65693619, 8.65541125, 8.82169607,

10.74806659,

10.74246353, 9.82857434, 9.98711359, 11.31163199,

8.96399085,

11.179814 , 12.42542346, 7.36200175, 6.19182742,

7.96272963,

6.33624015, 10.23367674, 8.47063207, 11.24789801,

9.7495214 ,

7.64293847, 7.47162597, 7.2902786 , 9.30923846, 11.3498611 ,

7.25017867, 7.40948037, 6.29918922, 6.95290024,

10.40630663,

10.39430584, 11.34543927, 11.40078708, 7.29598983,

12.3382395 ,

14.50848261, 8.23495736, 9.21346851, 9.71071461,

9.93904526,

9.72851223, 6.60124358, 10.05023211, 13.47060722,

9.42880635,

10.7496388 , 8.74031215, 10.37464781, 7.01095385,

11.43985422,

10.96488596, 11.28331786, 12.07164148, 10.88580584,

18.5412797 ,

11.87987996, 13.92778842, 14.62009509, 12.42371898,

11.27238956,

9.68599955]) **14. Measure the performance using Metrics.**

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)

p =mean\_squared\_error(y\_test, y\_test\_pred) print('Mean Squared error of testing set :%2f'%p)

----------------------------------------------------------------------

-----

NameError Traceback (most recent call last)

<ipython-input-1-9eb84f83135f> in <module>

1 from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

----> 2 s = mean\_squared\_error(y\_train, y\_train\_pred)

3 print('Mean Squared error of training set :%2f'%s)

4

5 p = mean\_squared\_error(y\_test, y\_test\_pred)

NameError: name 'y\_train' is not defined

# Random Forest Regression

*# Build the Model*

from sklearn.ensemble import RandomForestRegressor

rfr=RandomForestRegressor(max\_depth=2, random\_state=0, n\_estimators=100)

*#Train the model* 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)

0.44962462796743885

# K Neighbors Regression

*#Build the model*

from sklearn.neighbors import KNeighborsRegressorknr=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)

0.4765852196766659

# Decision Tree Regression

*#Build the model*

from sklearn.tree import DecisionTreeRegressordtr=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) *#Mesure the performance using Metrics* dtr.score(x\_train, y\_train)

0.10583634509810536

# Lasso Regression

*#Build the model* from sklearn.linear\_model import Lasso lr=Lasso(alpha=0.01)

*#Train the model* lr.fit(x\_train,y\_train) Lasso(alpha=0.01)

y\_train\_pred=lr.predict(x\_train) y\_test\_pred=lr.predict(x\_test)

*#Measure the performance using Metrics* lr.score(x\_train, y\_train)

0.5033325783616354

# ------------------------------------------------------------