**1. Download the dataset**

Data set link:[abalone](https://drive.google.com/file/d/1J_n4-xozcKvpl0ovfsM_rAXjfXTrUhp6/view)

from google.colab import drive

drive.mount('/content/drive')

Mounted at /content/drive

**2. Load the dataset into the tool**

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('/content/drive/MyDrive/DataAnalyticsAssignment/abalone.csv')

df.head()

|  | **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked weight** | **Viscera weight** | **Shell weight** | **Rings** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | M | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 15 |
| **1** | M | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 7 |
| **2** | F | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 9 |
| **3** | M | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 10 |
| **4** | I | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 7 |

**3. Perform Visualizations**

List item

* List item
* List item

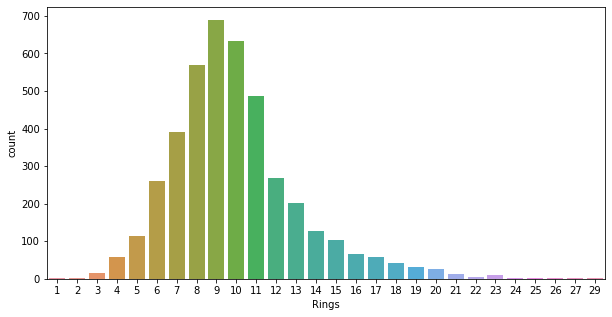
*italicized text*1 Univariate Analysi

#change the size of the figures

plt.rcParams['figure.figsize'] = (10, 5)

# countplot

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



#piechart

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

([,

,

,

,

],

[Text(0.6111272563215626, 0.9146165735327998, ''),

Text(-0.8270237769092663, 0.725280409515335, ''),

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

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

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

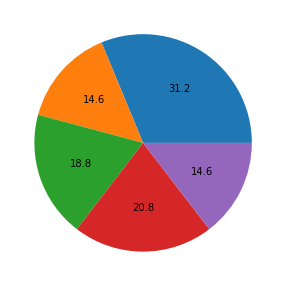
[Text(0.33334213981176136, 0.4988817673815271, '31.2'),

Text(-0.4511038783141452, 0.39560749609927365, '14.6'),

Text(-0.5681580837161301, -0.1928636614502974, '18.8'),

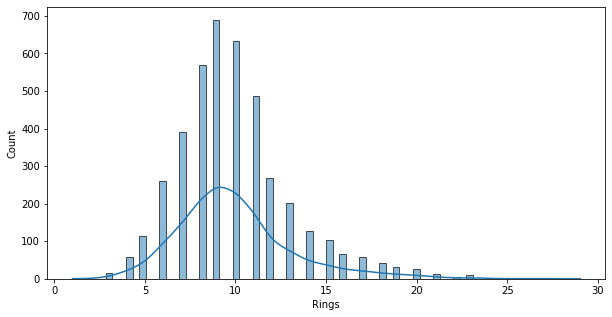
Text(-2.8088027480861175e-08, -0.5999999999999993, '20.8'),

Text(0.5381236242145833, -0.2653732561170434, '14.6')])



#histplot

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



# heatmap

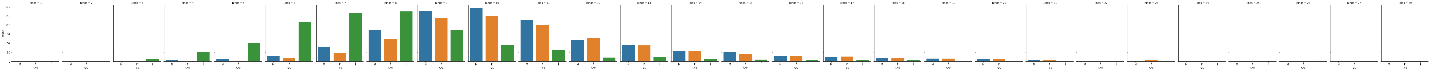
sns.heatmap(df[[ 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',

'Viscera weight', 'Shell weight', 'Rings']])



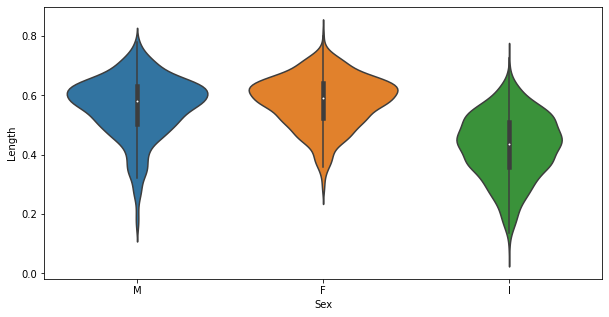
#countplot

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



#violin plot

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



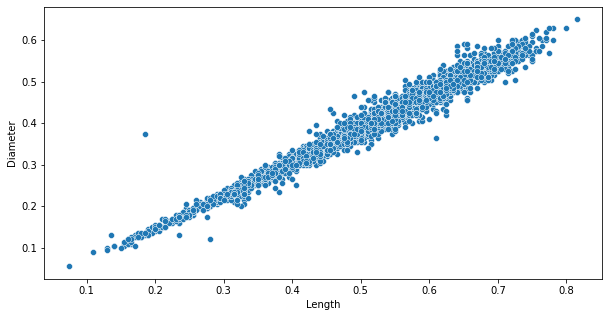
#strip plot

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



#scatter plot

sns.scatterplot(x = df["Length"],y = df["Diameter"])

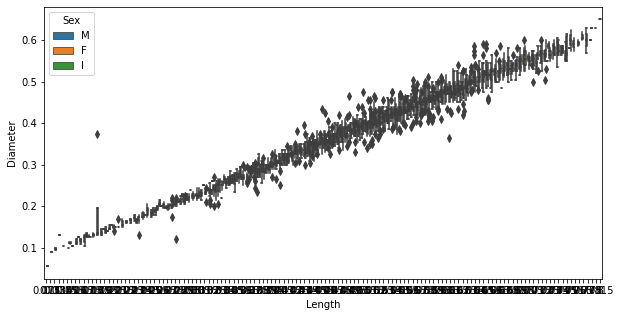


**3.3 Multi-Variate Analysis**

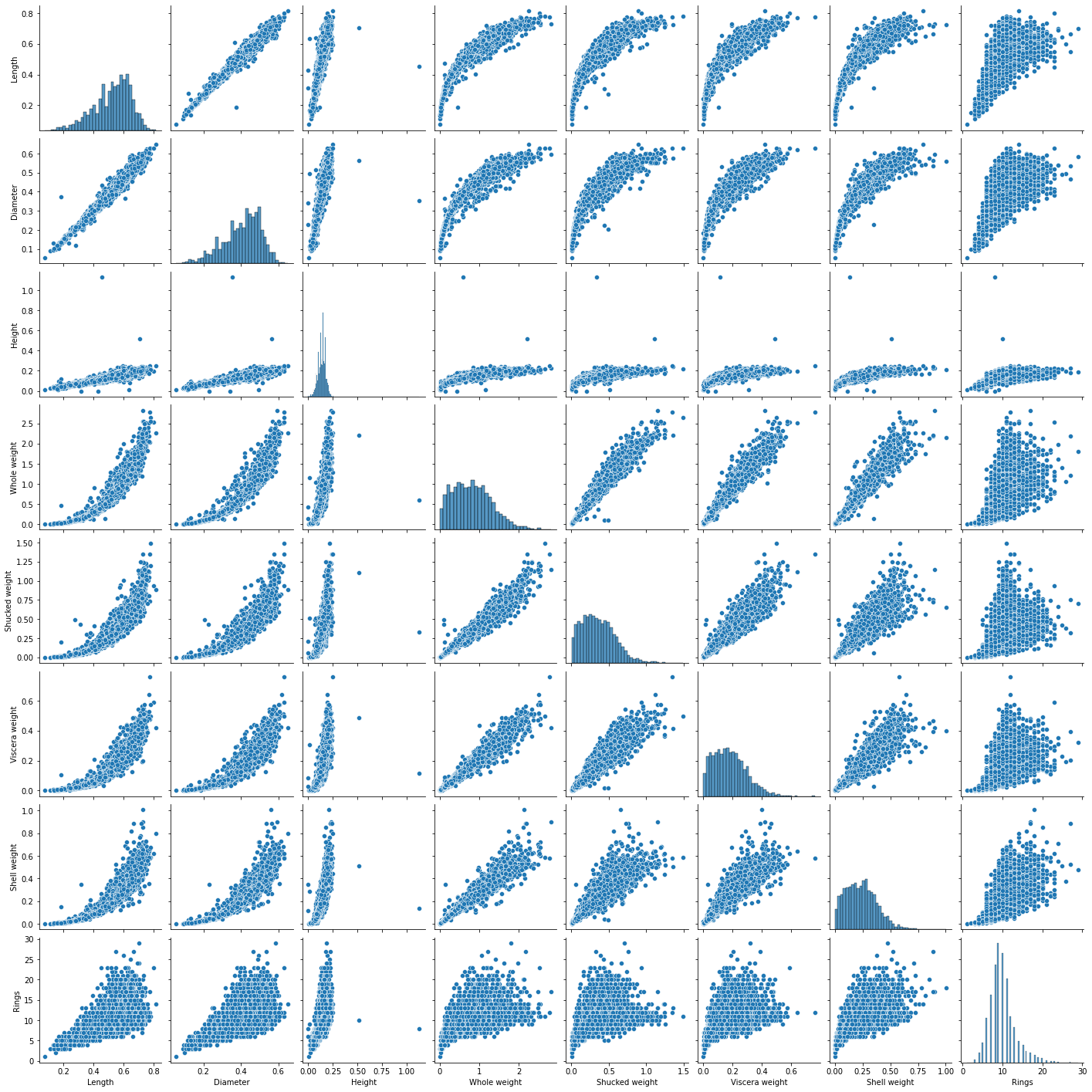
#boxplot

fig, ax1 = plt.subplots(figsize=(10,5))

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

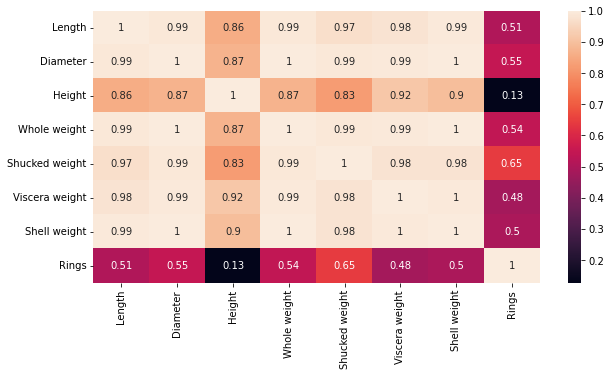


sns.pairplot(df)



fig=plt.figure(figsize=(10,5))

sns.heatmap(df.head().corr(),annot=True)



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

df

|  | **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

df.head()

|  | **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked weight** | **Viscera weight** | **Shell weight** | **Rings** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | M | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 15 |
| **1** | M | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 7 |
| **2** | F | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 9 |
| **3** | M | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 10 |
| **4** | I | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 7 |

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

df.describe()

|  | **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 |

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

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

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

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].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** | **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 |

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

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** | **Viscera weight** | **Shell weight** | **Rings** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | False | False | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False | False | False | False |
| **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 |

4177 rows × 9 columns

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

df.isnull().sum()

Sex 0

Length 0

Diameter 0

Height 0

Whole weight 0

Shucked weight 0

Viscera weight 0

Shell weight 0

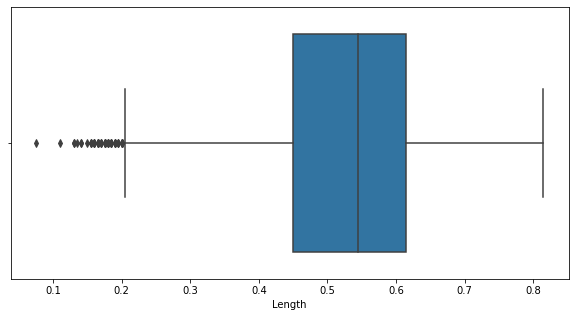
Rings 0

dtype: int64

**6. Find the outliers and replace them outliers**

#length

sns.boxplot(x=df['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

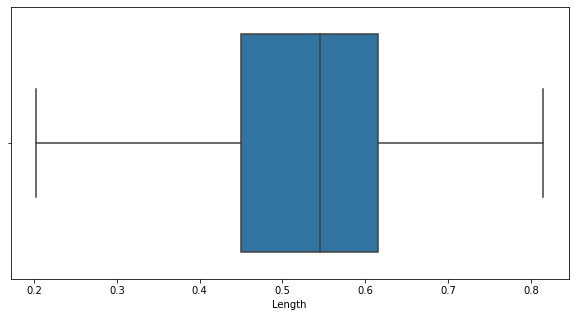
outliers: 49

new\_df = df.copy()

new\_df.loc[(new\_df['Length']>upper\_limit), 'Length'] = upper\_limit

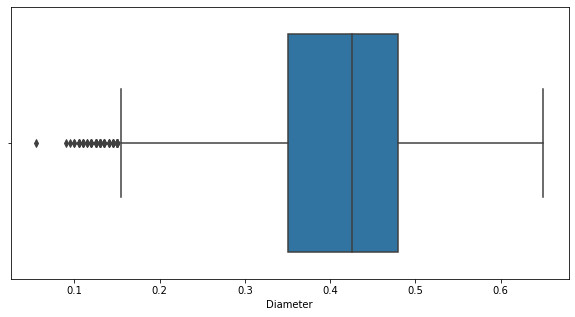
new\_df.loc[(new\_df['Length']<lower\_limit), 'Length'] = lower\_limit

sns.boxplot(x=new\_df['Length'])



#Diameter

sns.boxplot(x=df['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

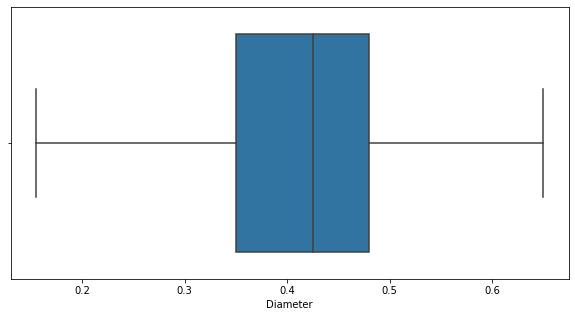
outliers: 59

new\_df = df.copy()

new\_df.loc[(new\_df['Diameter']>upper\_limit), 'Diameter'] = upper\_limit

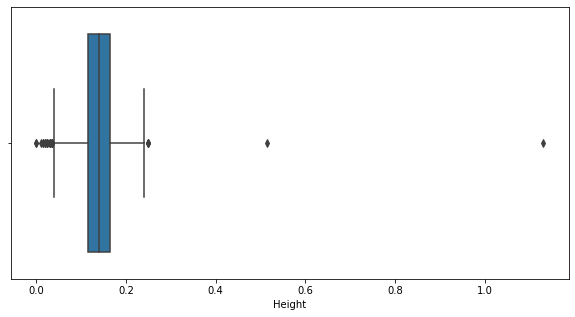
new\_df.loc[(new\_df['Diameter']<lower\_limit), 'Diameter'] = lower\_limit

sns.boxplot(x=new\_df['Diameter'])



#Height

sns.boxplot(x=df['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

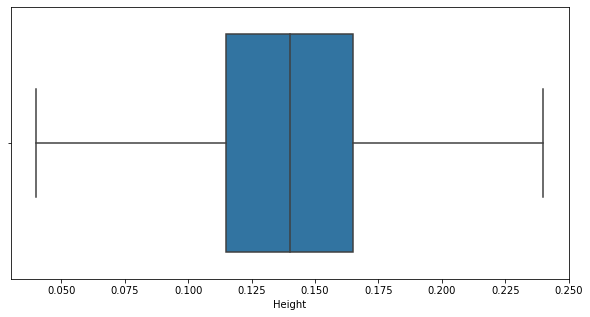
outliers: 29

new\_df = df.copy()

new\_df.loc[(new\_df['Height']>upper\_limit), 'Height'] = upper\_limit

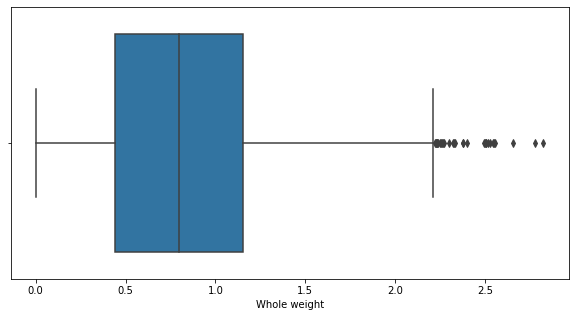
new\_df.loc[(new\_df['Height']<lower\_limit), 'Height'] = lower\_limit

sns.boxplot(x=new\_df['Height'])



#Whole Weight

sns.boxplot(x=df['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

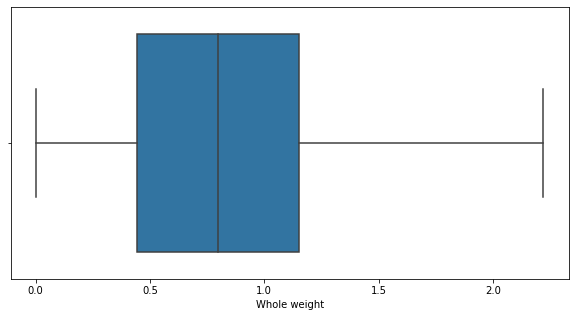
outliers: 30

new\_df = df.copy()

new\_df.loc[(new\_df['Whole weight']>upper\_limit), 'Whole weight'] = upper\_limit

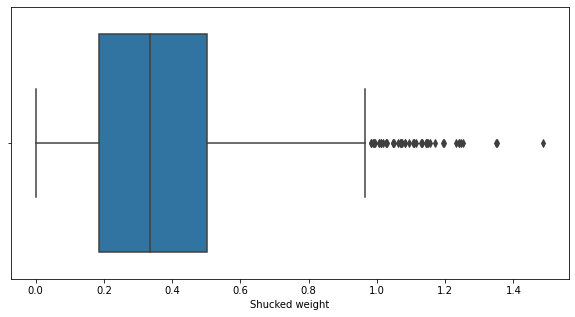
new\_df.loc[(new\_df['Whole weight']<lower\_limit), 'Whole weight'] = lower\_limit

sns.boxplot(x=new\_df['Whole weight'])



#Shucked weight

sns.boxplot(x=df['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

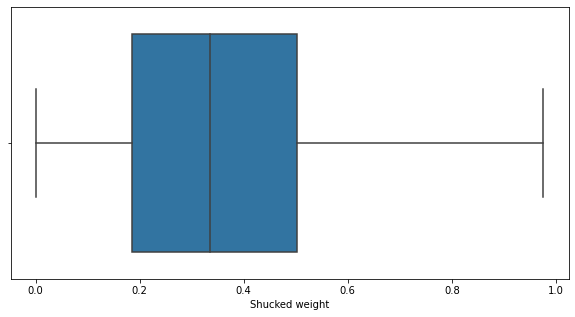
outliers: 48

new\_df = df.copy()

new\_df.loc[(new\_df['Shucked weight']>upper\_limit), 'Shucked weight'] = upper\_limit

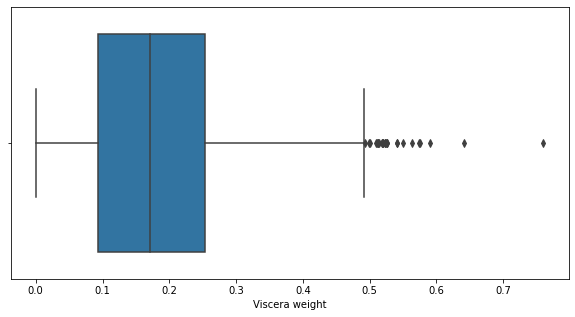
new\_df.loc[(new\_df['Shucked weight']<lower\_limit), 'Shucked weight'] = lower\_limit

sns.boxplot(x=new\_df['Shucked weight'])



#Viscera weight

sns.boxplot(x=df['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

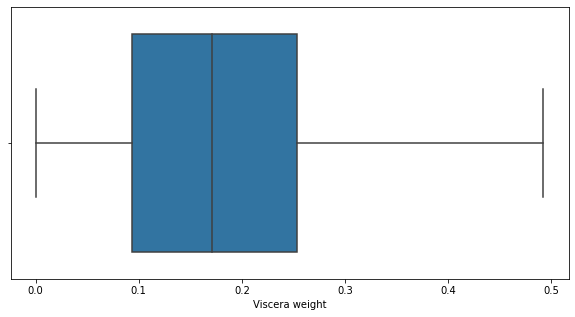
outliers: 26

new\_df = df.copy()

new\_df.loc[(new\_df['Viscera weight']>upper\_limit), 'Viscera weight'] = upper\_limit

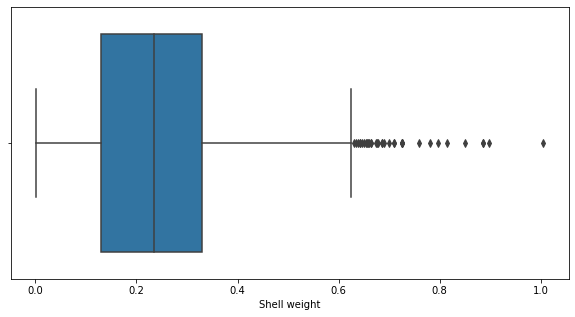
new\_df.loc[(new\_df['Viscera weight']<lower\_limit), 'Viscera weight'] = lower\_limit

sns.boxplot(x=new\_df['Viscera weight'])



#shell weight

sns.boxplot(x=df['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

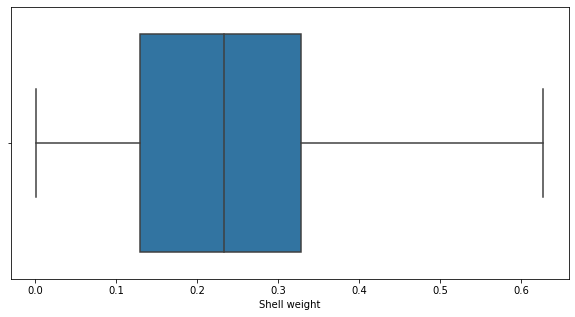
outliers: 35

new\_df = df.copy()

new\_df.loc[(new\_df['Shell weight']>upper\_limit), 'Shell weight'] = upper\_limit

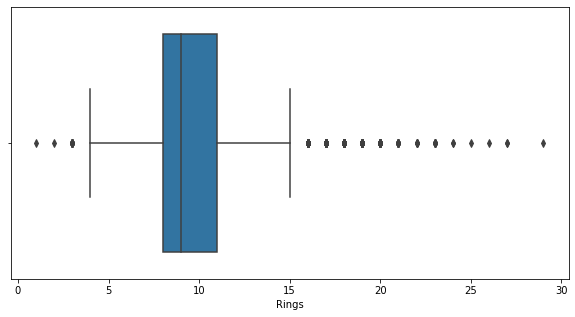
new\_df.loc[(new\_df['Shell weight']<lower\_limit), 'Shell weight'] = lower\_limit

sns.boxplot(x=new\_df['Shell weight'])



#Rings

sns.boxplot(x=df['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.loc[(df['Rings'] <= upper\_limit) & (df['Rings'] >= lower\_limit)]

print('before removing outliers:', len(df))

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

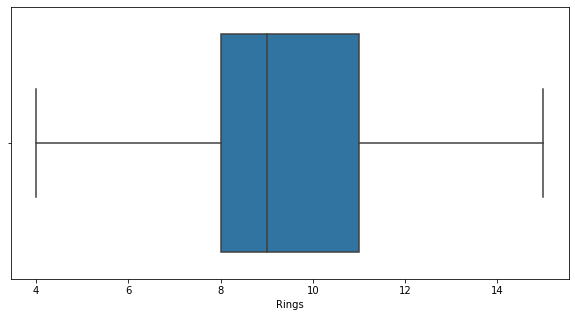
print('outliers:', len(df)-len(new\_df))

before removing outliers: 4177

after removing outliers: 3899

outliers: 278

sns.boxplot(x=new\_df['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** | **Viscera weight** | **Shell weight** | **Rings** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.1500 | 15 |
| **1** | 1 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.0700 | 7 |
| **2** | 0 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.2100 | 9 |
| **3** | 1 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.1550 | 10 |
| **4** | 2 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.0550 | 7 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4172** | 0 | 0.565 | 0.450 | 0.165 | 0.8870 | 0.3700 | 0.2390 | 0.2490 | 11 |
| **4173** | 1 | 0.590 | 0.440 | 0.135 | 0.9660 | 0.4390 | 0.2145 | 0.2605 | 10 |
| **4174** | 1 | 0.600 | 0.475 | 0.205 | 1.1760 | 0.5255 | 0.2875 | 0.3080 | 9 |
| **4175** | 0 | 0.625 | 0.485 | 0.150 | 1.0945 | 0.5310 | 0.2610 | 0.2960 | 10 |
| **4176** | 1 | 0.710 | 0.555 | 0.195 | 1.9485 | 0.9455 | 0.3765 | 0.4950 | 12 |

4177 rows × 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** | **Viscera weight** | **Shell weight** | **Rings** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.1500 | 15 |
| **1** | 1 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.0700 | 7 |
| **2** | 0 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.2100 | 9 |
| **3** | 1 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.1550 | 10 |
| **4** | 2 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.0550 | 7 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4172** | 0 | 0.565 | 0.450 | 0.165 | 0.8870 | 0.3700 | 0.2390 | 0.2490 | 11 |
| **4173** | 1 | 0.590 | 0.440 | 0.135 | 0.9660 | 0.4390 | 0.2145 | 0.2605 | 10 |
| **4174** | 1 | 0.600 | 0.475 | 0.205 | 1.1760 | 0.5255 | 0.2875 | 0.3080 | 9 |
| **4175** | 0 | 0.625 | 0.485 | 0.150 | 1.0945 | 0.5310 | 0.2610 | 0.2960 | 10 |
| **4176** | 1 | 0.710 | 0.555 | 0.195 | 1.9485 | 0.9455 | 0.3765 | 0.4950 | 12 |

4177 rows × 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** | **Viscera weight** | **Shell weight** | **Rings** | **0** | **1** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 15 | 1.0 | 0.0 |
| **1** | 1 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 7 | 1.0 | 0.0 |
| **2** | 0 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 9 | 0.0 | 0.0 |
| **3** | 1 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 10 | 1.0 | 0.0 |
| **4** | 2 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 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** | **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 |
| **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

y=df.iloc[:,8]

y

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**

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 LinearRegression

linearmodel = 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([10.17397542, 10.07068143, 8.67134702, 12.71828702, 8.86787867,

10.75020563, 13.81975514, 9.3096892 , 5.87779411, 7.63321116,

10.3846552 , 10.97183695, 9.08525726, 9.41456742, 7.03254741,

9.26266303, 7.98789822, 9.58057684, 6.90047509, 13.20121889,

12.31827093, 6.32982348, 6.93276273, 9.82100727, 6.89363451,

11.75279639, 12.40782101, 11.42741142, 6.17935212, 10.58353429,

5.73047254, 10.13685152, 8.2577295 , 10.50566987, 13.35578547,

11.97989071, 8.10446134, 9.39036207, 14.94288966, 9.48787719,

6.84291307, 8.72349593, 11.15558658, 7.91090618, 7.56937702,

10.81845142, 11.45602571, 6.52755349, 7.54769416, 13.37564367,

11.21365421, 11.33219466, 10.33833187, 8.97306333, 7.64224419,

12.34919834, 11.23908478, 8.29052292, 9.61979896, 12.16774129,

8.14726141, 7.86928166, 8.379765 , 8.21480518, 10.67368872,

9.08489685, 10.30109851, 9.61691359, 16.38370773, 10.38658295,

7.60433846, 8.91135057, 10.23679762, 9.68643202, 10.58887912,

14.09672862, 7.75396252, 9.38286525, 8.09019702, 6.70653863,

14.13250104, 10.94701043, 8.60106706, 10.55121131, 10.79580376,

8.62721105, 10.11423972, 9.80501137, 11.84720976, 8.86276973,

9.44337233, 11.75612497, 7.78851464, 7.50147585, 11.47768384,

8.06885032, 9.15504967, 7.21961486, 11.58946404, 8.74369597,

7.36918806, 7.23939635, 8.36582551, 16.31886394, 9.13027804,

10.04964164, 12.34827063, 7.92254209, 9.74825822, 9.24864352,

11.27226984, 7.60364506, 9.23331985, 9.56454156, 10.64353064,

9.62725603, 10.70957373, 9.46708597, 10.22589621, 5.35276609,

6.08220464, 10.06445933, 7.49186721, 5.905933 , 7.54578731,

7.19099017, 10.83549612, 9.23313769, 9.86779332, 11.15379941,

9.07336003, 14.99738757, 12.25181359, 9.94037845, 7.90403809,

9.85599078, 10.07807767, 14.0604697 , 9.03156801, 8.37773133,

14.58389859, 8.78667178, 12.76998234, 12.72708632, 9.08441782,

10.29168203, 9.15756652, 7.68305322, 12.96880044, 8.7975219 ,

11.21885759, 8.28789489, 12.13333445, 11.22596526, 8.99826017,

13.79588856, 13.46445746, 10.23862132, 10.32981686, 7.78587509,

11.44360059, 11.46190162, 10.71239955, 8.63350174, 11.8020593 ,

10.89779026, 7.45929232, 8.09751252, 8.61057936, 8.88657995,

6.8642686 , 7.89290115, 9.25728487, 10.17200214, 10.89536487,

9.31969189, 11.50812191, 10.36656963, 9.76111692, 13.81407369,

10.03392886, 10.04604909, 7.63318277, 11.83195646, 6.60618029,

9.92010927, 9.01730645, 13.84773421, 9.8166853 , 7.2201233 ,

11.06637665, 8.49137437, 10.02030329, 9.28863143, 10.08683779,

11.19695092, 13.87268294, 9.37431071, 8.19908208, 9.53377207,

4.42573307, 8.45210797, 10.56674365, 9.28466476, 12.54980798,

12.24104201, 10.71455522, 9.59895402, 7.24616938, 13.40651785,

12.19495086, 9.62779018, 10.38986657, 8.36734183, 8.40968821,

8.62161717, 9.9165741 , 11.69919037, 8.78071656, 17.70783782,

6.28747179, 6.60158198, 10.29637943, 9.91656486, 5.73605306,

9.96533837, 10.61629247, 6.1268223 , 7.21523919, 9.52603926,

11.07711147, 10.85882985, 16.31228355, 10.09815977, 6.99977395,

14.30253069, 11.34052186, 9.44471804, 9.60774992, 8.63354615,

11.12457954, 9.41696539, 7.15187159, 10.72504389, 9.18033076,

13.72223946, 10.48664851, 7.53704542, 11.70285227, 6.71622008,

9.31401174, 9.49632063, 14.30216128, 8.63837237, 10.21667701,

8.96829488, 12.59042533, 10.35039589, 9.75285273, 10.95971148,

8.79977768, 8.17789946, 8.00791705, 8.47518242, 8.14317763,

10.56949186, 8.19974679, 9.95488703, 8.38776052, 9.1675797 ,

10.74999782, 8.16995006, 10.04370958, 9.40427953, 10.7947037 ,

9.08379978, 8.69663582, 9.79058816, 9.52958313, 10.63558435,

10.53644573, 10.47595022, 8.79524302, 10.32008808, 11.65544496,

12.68157799, 9.82289102, 10.51327521, 12.32173905, 11.78354233,

10.57360346, 10.69520152, 10.11492664, 11.3382128 , 14.46564446,

12.48647971, 14.46425733, 8.15955755, 7.41738396, 11.44371933,

12.29297755, 11.78152324, 9.64085199, 6.68190225, 9.78463752,

10.15529128, 13.34990057, 7.78798847, 9.45126717, 9.08420505,

13.60154678, 19.9074499 , 7.99920078, 7.9877291 , 12.22407467,

5.23061326, 9.03185859, 9.84236027, 16.24078395, 7.72943272,

5.20272241, 10.08637807, 9.95608141, 9.62580214, 10.384239 ,

13.4927171 , 9.75642436, 11.02057169, 12.29295389, 8.03468201,

10.19702269, 10.39366802, 12.48699151, 6.92783091, 10.74280001,

8.3486369 , 8.61778065, 8.45543388, 8.00677636, 9.71675823,

16.92550805, 10.57953196, 15.90536436, 9.50462285, 9.39988304,

10.32470506, 12.86534747, 12.29079639, 14.74918872, 9.1639895 ,

6.20630441, 13.00872207, 7.54953187, 10.00180916, 9.083734 ,

11.13982846, 10.40601532, 6.04155134, 9.07858903, 12.23620916,

6.89145492, 7.1780126 , 6.69541463, 10.67061781, 8.83159662,

11.00954948, 11.18723069, 9.58537449, 11.6606398 , 10.99727223,

15.94197138, 11.86295159, 10.55422747, 11.84734619, 7.34152748,

11.09309102, 4.65257582, 9.68205642, 10.86540308, 7.02324895,

9.39411833, 11.41375515, 6.32025608, 9.23713509, 9.41794234,

10.08955103, 6.07584093, 12.16137614, 13.98070402, 7.50574666,

11.63939066, 9.87978312, 7.98519933, 7.86956897, 9.09066842,

11.33150975, 5.8932936 , 7.38592484, 8.16104742, 7.13985192,

12.12374737, 9.49445256, 12.6608272 , 10.92822679, 9.64579814,

11.37740016, 13.35655501, 12.03431308, 8.29052643, 15.07767248,

16.97826247, 8.49546962, 9.14262023, 10.89505792, 8.94620052,

8.10005376, 8.2741099 , 9.04242193, 10.80323642, 10.33724187,

10.7569934 , 11.25459692, 7.79289162, 11.82360347, 9.36253889,

9.20535978, 11.33237436, 11.90082395, 7.21549415, 8.98121219,

9.18751815, 10.85940902, 10.89884311, 8.11486849, 11.83199882,

10.81757103, 10.92910856, 12.08135246, 10.71535444, 9.71625622,

9.01781515, 10.95518072, 8.34046289, 10.17978723, 9.2628684 ,

6.27546838, 13.01673973, 9.29323555, 10.88277688, 10.65023675,

7.79813885, 11.3480945 , 13.44888209, 15.81779561, 7.9295 ,

10.48476882, 5.59038454, 8.40947994, 17.62563803, 6.81638847,

7.21469929, 12.84101012, 9.32876132, 18.90467618, 11.63015657,

10.99326775, 12.40854784, 9.10945045, 12.9140317 , 7.8017051 ,

9.90552646, 8.3464963 , 12.88270016, 8.25612957, 9.34584491,

11.65653916, 12.39564448, 10.2772439 , 9.3555932 , 9.34328984,

16.64921469, 11.26978487, 12.18802387, 10.34557299, 10.59100518,

10.179332 , 6.73708151, 12.92492286, 7.80725467, 9.92395643,

10.06050323, 12.38547025, 10.07510473, 9.65976876, 11.51754953,

7.4877671 , 10.78351855, 14.60826713, 19.17650316, 7.60163495,

9.76719767, 11.36471963, 9.02854673, 11.17540692, 12.78651789,

11.96011417, 7.25130786, 10.5343224 , 9.76155361, 13.64379351,

11.22547096, 8.27602732, 15.78794218, 11.02301369, 8.91859135,

10.87261659, 5.83492447, 8.6425455 , 11.97076478, 7.93997707,

12.26801825, 11.96747073, 15.00085408, 10.75919008, 7.84940305,

12.56772493, 9.44675979, 12.64068414, 9.77515558, 7.1381343 ,

11.05597325, 10.1280931 , 11.16712892, 7.8402234 , 5.01479697,

9.35977556, 7.89993382, 11.7162187 , 13.96597243, 8.99518238,

8.83630485, 9.74606374, 7.98103739, 8.55583733, 9.11193498,

9.84097136, 14.59619824, 11.08122319, 5.26162661, 9.36987379,

17.68746302, 9.19687317, 6.97536302, 15.40254755, 10.94042066,

8.11738171, 12.2744849 , 9.18371758, 8.6557194 , 6.09011959,

11.40208474, 7.68337939, 12.49805976, 10.85065987, 7.37385646,

11.63132344, 11.15538826, 9.4669364 , 9.62369785, 5.79947458,

11.51770471, 8.78142722, 9.68851357, 7.86140492, 10.11144881,

8.61144343, 8.41492208, 7.77951366, 9.61522353, 10.4936152 ,

12.59717806, 8.49685449, 13.74886065, 10.38168 , 10.58203164,

8.96082279, 7.09674879, 9.76569836, 6.63030467, 8.77986736,

18.81281935, 11.99051575, 7.83652082, 9.226031 , 6.16416627,

10.94455769, 13.14586874, 11.66712638, 11.30215648, 9.25448888,

7.26095181, 11.2501062 , 7.64633084, 8.60575428, 9.98687751,

6.70614951, 11.20633964, 9.05142488, 6.11967056, 10.48426928,

9.70077869, 9.4378062 , 5.26350078, 8.46643588, 12.08022381,

11.54467485, 5.37634476, 9.96782175, 10.60255086, 8.92740429,

9.41477621, 11.20247256, 11.18792496, 8.93558577, 7.08613717,

11.0009911 , 10.92435509, 9.63686967, 10.33608233, 11.34742638,

9.66588006, 11.44518221, 9.52292647, 13.22462549, 11.29187232,

8.65570115, 8.1475651 , 6.45539076, 11.15749119, 12.21130351,

9.61298 , 13.15361983, 11.12240773, 8.98401594, 9.30281553,

8.7786281 , 20.823057 , 9.77559007, 6.97664278, 8.02486076,

5.72071161, 5.25241178, 9.15342537, 8.7139211 , 7.38888852,

10.77588248, 12.55690906, 10.56627908, 10.00154538, 7.51361598,

10.5813927 , 10.11686683, 7.22117183, 7.13356489, 4.5775051 ,

10.68352071, 6.99739725, 8.41520956, 5.44565067, 11.80265579,

7.9881176 , 12.67148822, 10.90697897, 9.08923154, 10.14366775,

13.41589911, 6.57982745, 10.67212561, 7.96560911, 8.82358657,

4.77901726, 6.15115267, 11.24701873, 10.65799593, 6.12410353,

11.85295131, 14.13405726, 8.21581124, 10.38155575, 7.52322851,

9.65919407, 10.50570684, 10.53815013, 11.03105652, 14.94420309,

8.1629979 , 13.20620937, 13.76807838, 10.06643372, 6.08667484,

7.96640174, 9.39281163, 8.9050128 , 9.1292816 , 9.45801676,

9.42507151, 10.73687597, 10.14317479, 8.55714844, 11.25360358,

9.96226645, 8.66100729, 10.86426286, 7.30857096, 12.59277927,

7.46577829, 8.93329457, -3.4842505 , 9.99035938, 8.63918657,

12.32485852, 12.1632845 , 10.44428429, 10.67203842, 8.89267787,

9.78142809, 13.97144567, 8.24050808, 8.71524195, 12.38640756,

11.18986679, 8.18474635, 6.60605893, 9.26527952, 9.51550508,

4.77282357, 12.47965749, 12.34329938, 8.38642711, 11.82341819,

10.18619148, 9.86825241, 9.36663152, 15.78962765, 11.07421522,

13.98760595, 10.22976508, 9.30838983, 13.96711829, 11.82822129,

19.02810829, 9.65306974, 9.46423954, 14.92680808, 11.64748294,

7.94239092, 6.98331916, 8.8509933 , 11.85385203, 8.2656887 ,

11.36743669, 8.90454322, 10.38098675, 7.70083638, 9.66900187,

8.30679606, 7.56821231, 10.15969955, 9.49467134, 8.16319961,

12.84097277, 8.10548003, 10.62392686, 11.86885538, 10.2362116 ,

11.83076288, 9.32783045, 10.34572183, 10.62568974, 8.97676071,

9.13508074, 17.69207229, 6.99754881, 7.85010695, 10.90888864,

13.18201489, 12.00545114, 9.27436134, 9.17594523, 9.5500056 ,

8.63693289, 10.13830113, 8.77430855, 6.70722936, 9.76476193,

10.97166462, 6.2434988 , 10.55199796, 13.55224476, 8.5792102 ,

9.18759628, 9.44127845, 9.8394762 , 8.44224265, 11.36534893,

8.83745096, 8.39855326, 9.70755212, 14.5637696 , 10.55770228,

8.64156549, 7.95558695, 8.27908153, 9.07302743, 7.6430385 ,

12.5282226 , 10.60750098, 8.56517665, 12.41248015, 9.12142511,

8.03734112, 15.08509198, 6.05827171, 9.02007634, 6.02558078,

11.63680507, 9.97324886, 15.03187662, 8.51771273, 10.04145538,

13.51888784, 9.10029217, 9.42349388, 9.04886862, 8.52978087,

10.72524213])

**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)

Mean Squared error of training set :4.949028

Mean Squared error of testing set :4.785948

# 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)

0.41877128928053997

**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)

0.48693687494342397

**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)

#Mesure the performance using Metrics

dtr.score(x\_train, y\_train)

0.07943400002124779

**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)

0.512187188782296

**1. Download the dataset**

Data set link:[abalone](https://drive.google.com/file/d/1J_n4-xozcKvpl0ovfsM_rAXjfXTrUhp6/view)

from google.colab import drive

drive.mount('/content/drive')

Mounted at /content/drive

**2. Load the dataset into the tool**

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('/content/drive/MyDrive/DataAnalyticsAssignment/abalone.csv')

df.head()

|  | **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked weight** | **Viscera weight** | **Shell weight** | **Rings** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | M | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 15 |
| **1** | M | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 7 |
| **2** | F | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 9 |
| **3** | M | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 10 |
| **4** | I | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 7 |

**3. Perform Visualizations**

List item

* List item
* List item

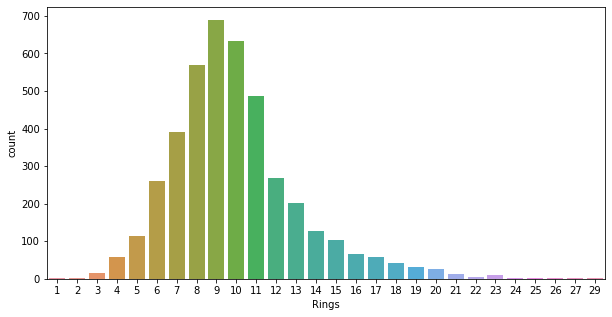
*italicized text*1 Univariate Analysi

#change the size of the figures

plt.rcParams['figure.figsize'] = (10, 5)

# countplot

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



#piechart

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

([,

,

,

,

],

[Text(0.6111272563215626, 0.9146165735327998, ''),

Text(-0.8270237769092663, 0.725280409515335, ''),

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

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

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

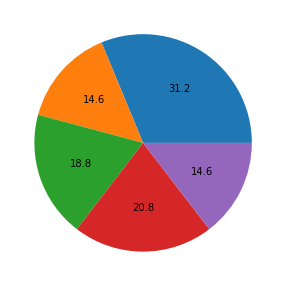
[Text(0.33334213981176136, 0.4988817673815271, '31.2'),

Text(-0.4511038783141452, 0.39560749609927365, '14.6'),

Text(-0.5681580837161301, -0.1928636614502974, '18.8'),

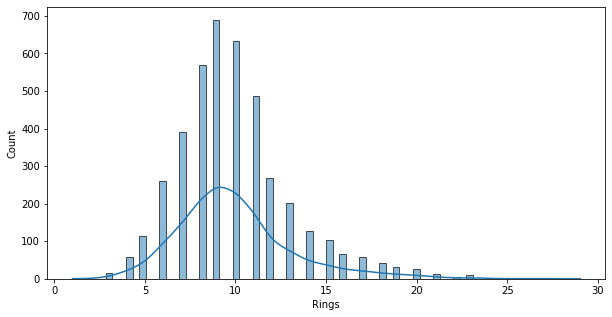
Text(-2.8088027480861175e-08, -0.5999999999999993, '20.8'),

Text(0.5381236242145833, -0.2653732561170434, '14.6')])



#histplot

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



# heatmap

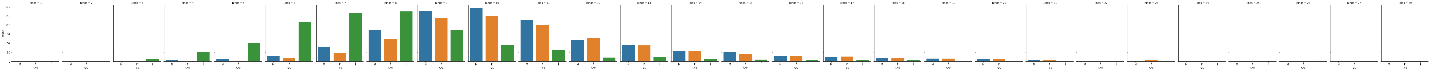
sns.heatmap(df[[ 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',

'Viscera weight', 'Shell weight', 'Rings']])



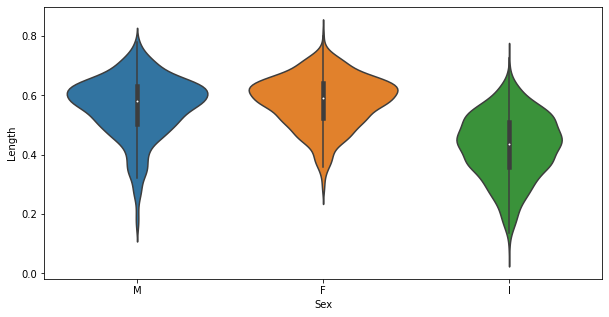
#countplot

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



#violin plot

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



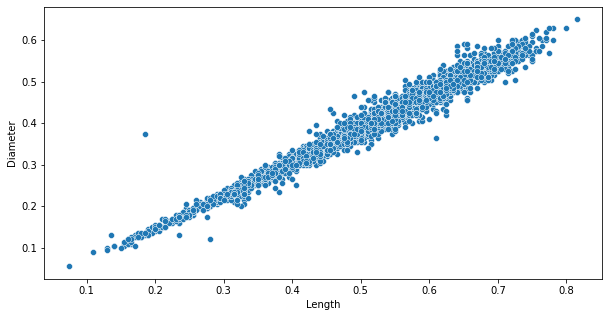
#strip plot

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



#scatter plot

sns.scatterplot(x = df["Length"],y = df["Diameter"])

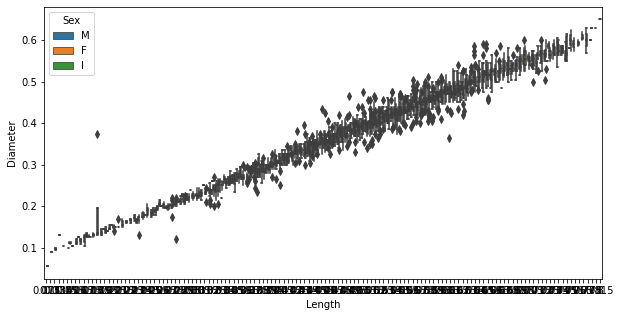


**3.3 Multi-Variate Analysis**

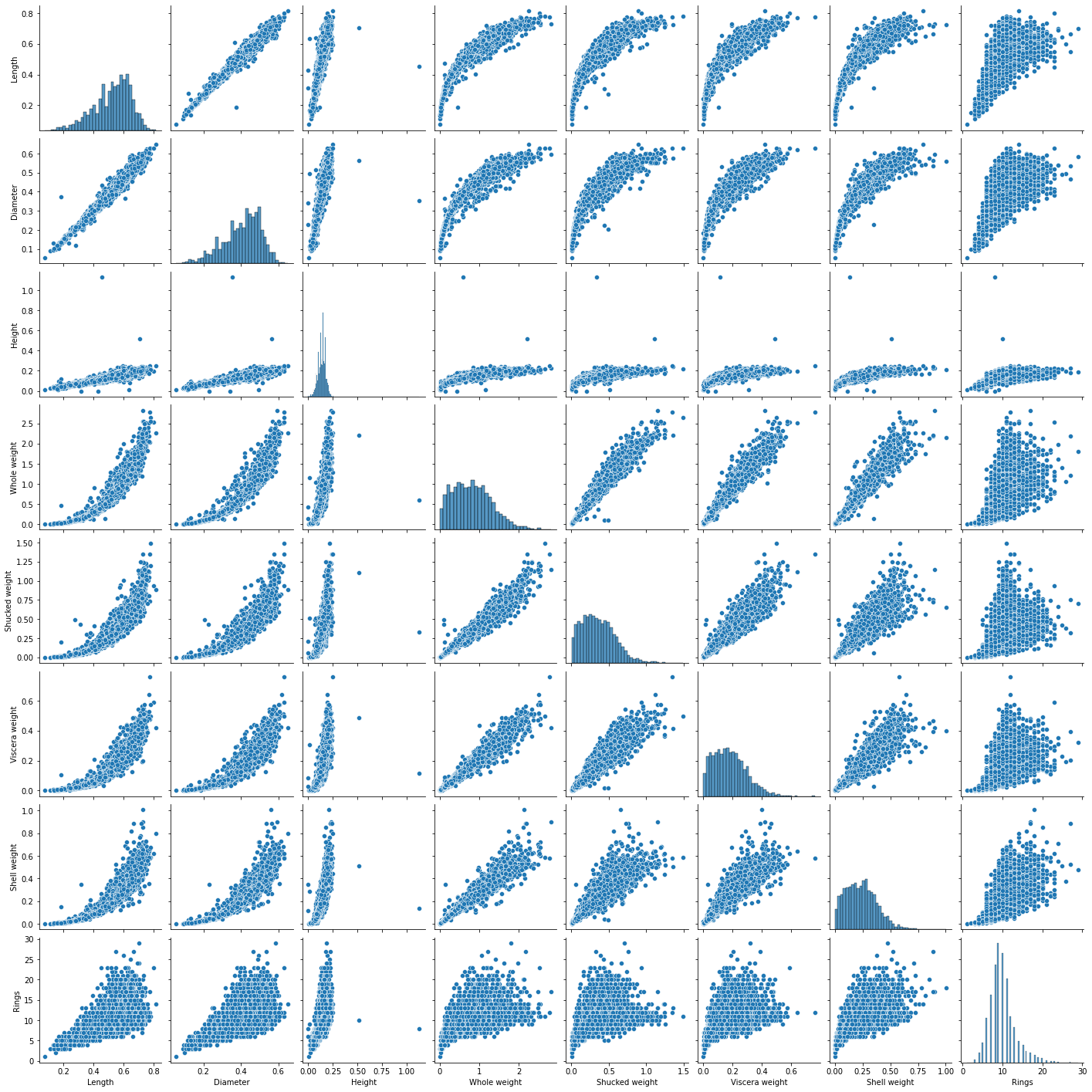
#boxplot

fig, ax1 = plt.subplots(figsize=(10,5))

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

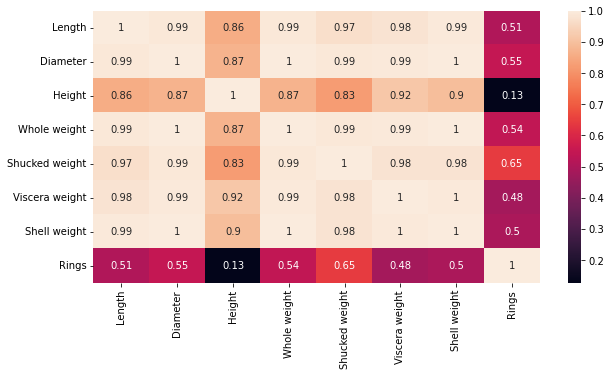


sns.pairplot(df)



fig=plt.figure(figsize=(10,5))

sns.heatmap(df.head().corr(),annot=True)



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

df

|  | **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

df.head()

|  | **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked weight** | **Viscera weight** | **Shell weight** | **Rings** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | M | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 15 |
| **1** | M | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 7 |
| **2** | F | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 9 |
| **3** | M | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 10 |
| **4** | I | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 7 |

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

df.describe()

|  | **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 |

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

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

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

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].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** | **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 |

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

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** | **Viscera weight** | **Shell weight** | **Rings** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | False | False | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False | False | False | False |
| **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 |

4177 rows × 9 columns

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

df.isnull().sum()

Sex 0

Length 0

Diameter 0

Height 0

Whole weight 0

Shucked weight 0

Viscera weight 0

Shell weight 0

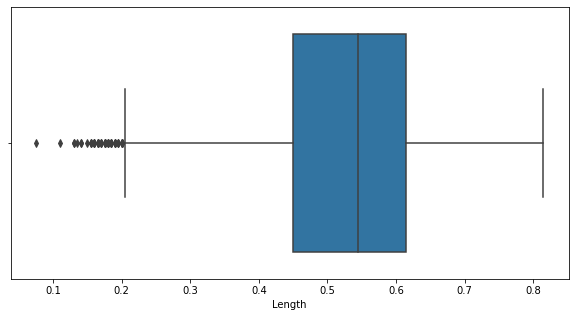
Rings 0

dtype: int64

**6. Find the outliers and replace them outliers**

#length

sns.boxplot(x=df['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

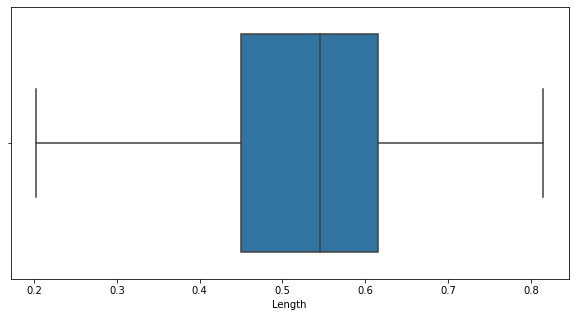
outliers: 49

new\_df = df.copy()

new\_df.loc[(new\_df['Length']>upper\_limit), 'Length'] = upper\_limit

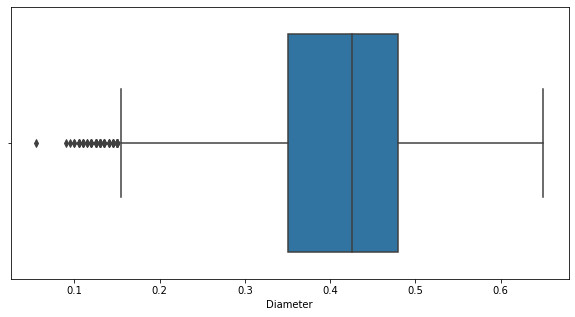
new\_df.loc[(new\_df['Length']<lower\_limit), 'Length'] = lower\_limit

sns.boxplot(x=new\_df['Length'])



#Diameter

sns.boxplot(x=df['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

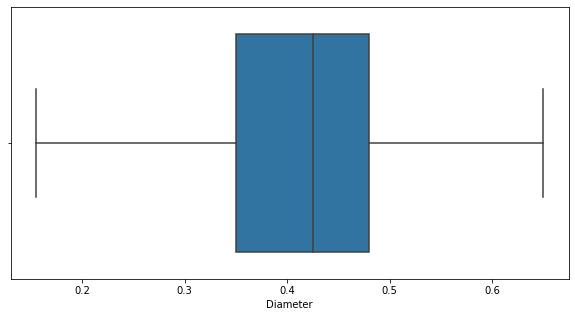
outliers: 59

new\_df = df.copy()

new\_df.loc[(new\_df['Diameter']>upper\_limit), 'Diameter'] = upper\_limit

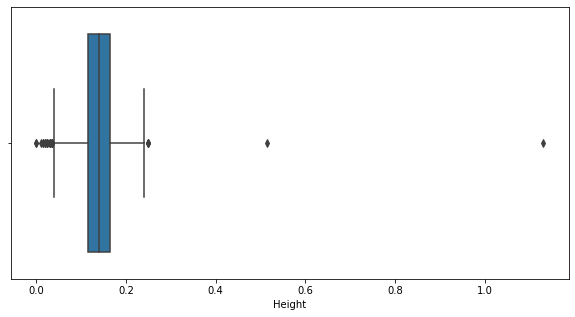
new\_df.loc[(new\_df['Diameter']<lower\_limit), 'Diameter'] = lower\_limit

sns.boxplot(x=new\_df['Diameter'])



#Height

sns.boxplot(x=df['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

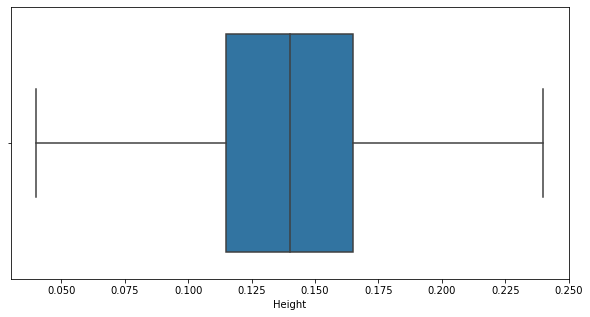
outliers: 29

new\_df = df.copy()

new\_df.loc[(new\_df['Height']>upper\_limit), 'Height'] = upper\_limit

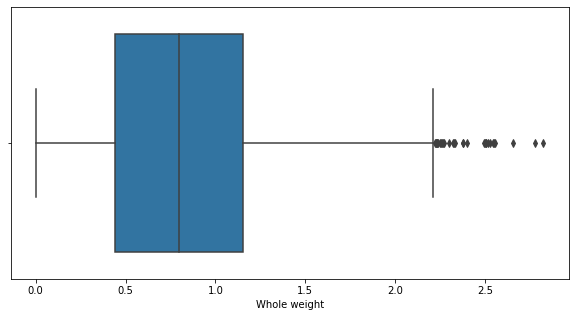
new\_df.loc[(new\_df['Height']<lower\_limit), 'Height'] = lower\_limit

sns.boxplot(x=new\_df['Height'])



#Whole Weight

sns.boxplot(x=df['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

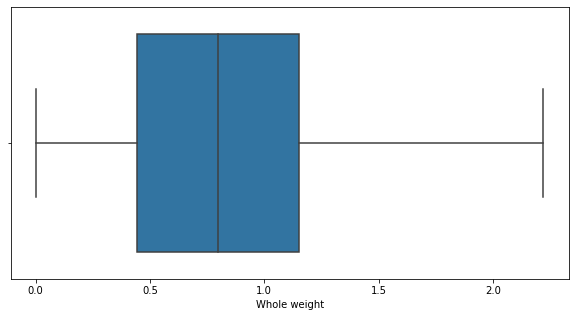
outliers: 30

new\_df = df.copy()

new\_df.loc[(new\_df['Whole weight']>upper\_limit), 'Whole weight'] = upper\_limit

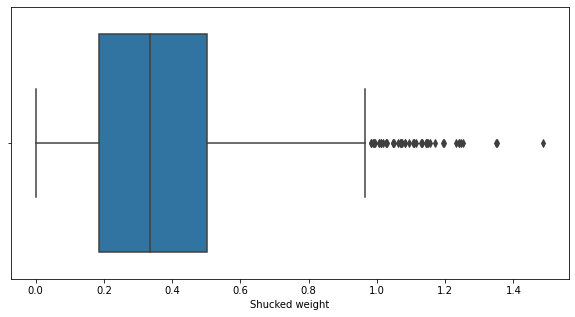
new\_df.loc[(new\_df['Whole weight']<lower\_limit), 'Whole weight'] = lower\_limit

sns.boxplot(x=new\_df['Whole weight'])



#Shucked weight

sns.boxplot(x=df['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

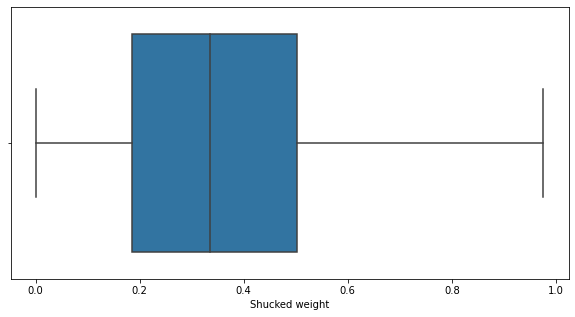
outliers: 48

new\_df = df.copy()

new\_df.loc[(new\_df['Shucked weight']>upper\_limit), 'Shucked weight'] = upper\_limit

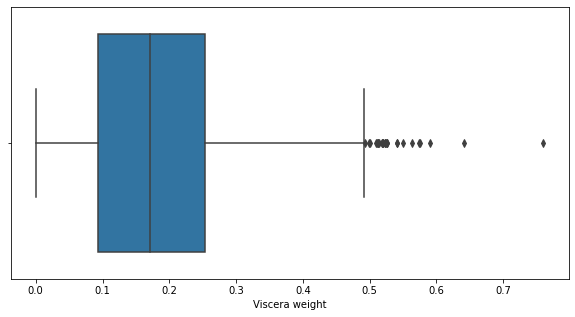
new\_df.loc[(new\_df['Shucked weight']<lower\_limit), 'Shucked weight'] = lower\_limit

sns.boxplot(x=new\_df['Shucked weight'])



#Viscera weight

sns.boxplot(x=df['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

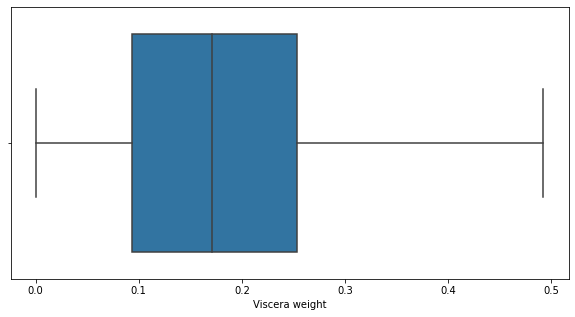
outliers: 26

new\_df = df.copy()

new\_df.loc[(new\_df['Viscera weight']>upper\_limit), 'Viscera weight'] = upper\_limit

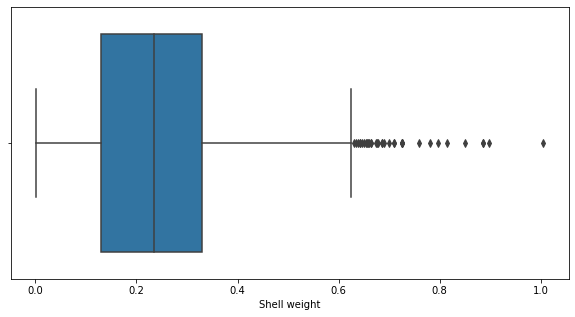
new\_df.loc[(new\_df['Viscera weight']<lower\_limit), 'Viscera weight'] = lower\_limit

sns.boxplot(x=new\_df['Viscera weight'])



#shell weight

sns.boxplot(x=df['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

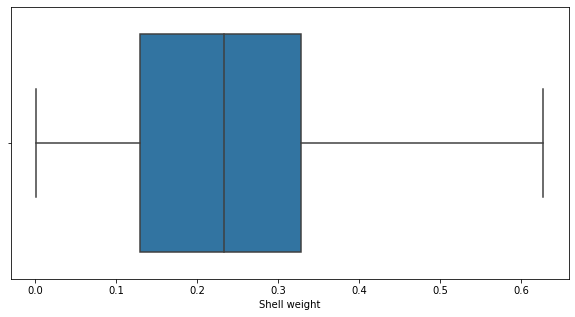
outliers: 35

new\_df = df.copy()

new\_df.loc[(new\_df['Shell weight']>upper\_limit), 'Shell weight'] = upper\_limit

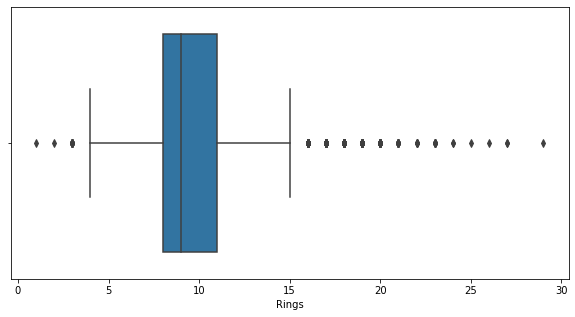
new\_df.loc[(new\_df['Shell weight']<lower\_limit), 'Shell weight'] = lower\_limit

sns.boxplot(x=new\_df['Shell weight'])



#Rings

sns.boxplot(x=df['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.loc[(df['Rings'] <= upper\_limit) & (df['Rings'] >= lower\_limit)]

print('before removing outliers:', len(df))

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

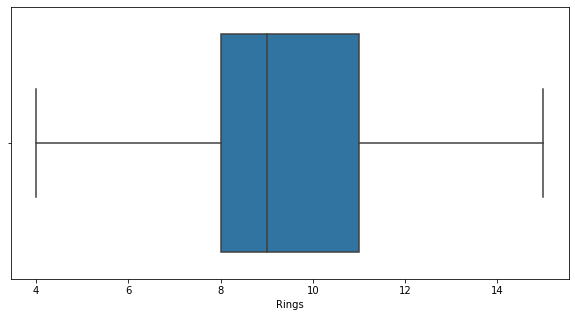
print('outliers:', len(df)-len(new\_df))

before removing outliers: 4177

after removing outliers: 3899

outliers: 278

sns.boxplot(x=new\_df['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** | **Viscera weight** | **Shell weight** | **Rings** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.1500 | 15 |
| **1** | 1 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.0700 | 7 |
| **2** | 0 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.2100 | 9 |
| **3** | 1 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.1550 | 10 |
| **4** | 2 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.0550 | 7 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4172** | 0 | 0.565 | 0.450 | 0.165 | 0.8870 | 0.3700 | 0.2390 | 0.2490 | 11 |
| **4173** | 1 | 0.590 | 0.440 | 0.135 | 0.9660 | 0.4390 | 0.2145 | 0.2605 | 10 |
| **4174** | 1 | 0.600 | 0.475 | 0.205 | 1.1760 | 0.5255 | 0.2875 | 0.3080 | 9 |
| **4175** | 0 | 0.625 | 0.485 | 0.150 | 1.0945 | 0.5310 | 0.2610 | 0.2960 | 10 |
| **4176** | 1 | 0.710 | 0.555 | 0.195 | 1.9485 | 0.9455 | 0.3765 | 0.4950 | 12 |

4177 rows × 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** | **Viscera weight** | **Shell weight** | **Rings** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.1500 | 15 |
| **1** | 1 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.0700 | 7 |
| **2** | 0 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.2100 | 9 |
| **3** | 1 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.1550 | 10 |
| **4** | 2 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.0550 | 7 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4172** | 0 | 0.565 | 0.450 | 0.165 | 0.8870 | 0.3700 | 0.2390 | 0.2490 | 11 |
| **4173** | 1 | 0.590 | 0.440 | 0.135 | 0.9660 | 0.4390 | 0.2145 | 0.2605 | 10 |
| **4174** | 1 | 0.600 | 0.475 | 0.205 | 1.1760 | 0.5255 | 0.2875 | 0.3080 | 9 |
| **4175** | 0 | 0.625 | 0.485 | 0.150 | 1.0945 | 0.5310 | 0.2610 | 0.2960 | 10 |
| **4176** | 1 | 0.710 | 0.555 | 0.195 | 1.9485 | 0.9455 | 0.3765 | 0.4950 | 12 |

4177 rows × 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** | **Viscera weight** | **Shell weight** | **Rings** | **0** | **1** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 15 | 1.0 | 0.0 |
| **1** | 1 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 7 | 1.0 | 0.0 |
| **2** | 0 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 9 | 0.0 | 0.0 |
| **3** | 1 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 10 | 1.0 | 0.0 |
| **4** | 2 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 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** | **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 |
| **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

y=df.iloc[:,8]

y

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**

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 LinearRegression

linearmodel = 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([10.17397542, 10.07068143, 8.67134702, 12.71828702, 8.86787867,

10.75020563, 13.81975514, 9.3096892 , 5.87779411, 7.63321116,

10.3846552 , 10.97183695, 9.08525726, 9.41456742, 7.03254741,

9.26266303, 7.98789822, 9.58057684, 6.90047509, 13.20121889,

12.31827093, 6.32982348, 6.93276273, 9.82100727, 6.89363451,

11.75279639, 12.40782101, 11.42741142, 6.17935212, 10.58353429,

5.73047254, 10.13685152, 8.2577295 , 10.50566987, 13.35578547,

11.97989071, 8.10446134, 9.39036207, 14.94288966, 9.48787719,

6.84291307, 8.72349593, 11.15558658, 7.91090618, 7.56937702,

10.81845142, 11.45602571, 6.52755349, 7.54769416, 13.37564367,

11.21365421, 11.33219466, 10.33833187, 8.97306333, 7.64224419,

12.34919834, 11.23908478, 8.29052292, 9.61979896, 12.16774129,

8.14726141, 7.86928166, 8.379765 , 8.21480518, 10.67368872,

9.08489685, 10.30109851, 9.61691359, 16.38370773, 10.38658295,

7.60433846, 8.91135057, 10.23679762, 9.68643202, 10.58887912,

14.09672862, 7.75396252, 9.38286525, 8.09019702, 6.70653863,

14.13250104, 10.94701043, 8.60106706, 10.55121131, 10.79580376,

8.62721105, 10.11423972, 9.80501137, 11.84720976, 8.86276973,

9.44337233, 11.75612497, 7.78851464, 7.50147585, 11.47768384,

8.06885032, 9.15504967, 7.21961486, 11.58946404, 8.74369597,

7.36918806, 7.23939635, 8.36582551, 16.31886394, 9.13027804,

10.04964164, 12.34827063, 7.92254209, 9.74825822, 9.24864352,

11.27226984, 7.60364506, 9.23331985, 9.56454156, 10.64353064,

9.62725603, 10.70957373, 9.46708597, 10.22589621, 5.35276609,

6.08220464, 10.06445933, 7.49186721, 5.905933 , 7.54578731,

7.19099017, 10.83549612, 9.23313769, 9.86779332, 11.15379941,

9.07336003, 14.99738757, 12.25181359, 9.94037845, 7.90403809,

9.85599078, 10.07807767, 14.0604697 , 9.03156801, 8.37773133,

14.58389859, 8.78667178, 12.76998234, 12.72708632, 9.08441782,

10.29168203, 9.15756652, 7.68305322, 12.96880044, 8.7975219 ,

11.21885759, 8.28789489, 12.13333445, 11.22596526, 8.99826017,

13.79588856, 13.46445746, 10.23862132, 10.32981686, 7.78587509,

11.44360059, 11.46190162, 10.71239955, 8.63350174, 11.8020593 ,

10.89779026, 7.45929232, 8.09751252, 8.61057936, 8.88657995,

6.8642686 , 7.89290115, 9.25728487, 10.17200214, 10.89536487,

9.31969189, 11.50812191, 10.36656963, 9.76111692, 13.81407369,

10.03392886, 10.04604909, 7.63318277, 11.83195646, 6.60618029,

9.92010927, 9.01730645, 13.84773421, 9.8166853 , 7.2201233 ,

11.06637665, 8.49137437, 10.02030329, 9.28863143, 10.08683779,

11.19695092, 13.87268294, 9.37431071, 8.19908208, 9.53377207,

4.42573307, 8.45210797, 10.56674365, 9.28466476, 12.54980798,

12.24104201, 10.71455522, 9.59895402, 7.24616938, 13.40651785,

12.19495086, 9.62779018, 10.38986657, 8.36734183, 8.40968821,

8.62161717, 9.9165741 , 11.69919037, 8.78071656, 17.70783782,

6.28747179, 6.60158198, 10.29637943, 9.91656486, 5.73605306,

9.96533837, 10.61629247, 6.1268223 , 7.21523919, 9.52603926,

11.07711147, 10.85882985, 16.31228355, 10.09815977, 6.99977395,

14.30253069, 11.34052186, 9.44471804, 9.60774992, 8.63354615,

11.12457954, 9.41696539, 7.15187159, 10.72504389, 9.18033076,

13.72223946, 10.48664851, 7.53704542, 11.70285227, 6.71622008,

9.31401174, 9.49632063, 14.30216128, 8.63837237, 10.21667701,

8.96829488, 12.59042533, 10.35039589, 9.75285273, 10.95971148,

8.79977768, 8.17789946, 8.00791705, 8.47518242, 8.14317763,

10.56949186, 8.19974679, 9.95488703, 8.38776052, 9.1675797 ,

10.74999782, 8.16995006, 10.04370958, 9.40427953, 10.7947037 ,

9.08379978, 8.69663582, 9.79058816, 9.52958313, 10.63558435,

10.53644573, 10.47595022, 8.79524302, 10.32008808, 11.65544496,

12.68157799, 9.82289102, 10.51327521, 12.32173905, 11.78354233,

10.57360346, 10.69520152, 10.11492664, 11.3382128 , 14.46564446,

12.48647971, 14.46425733, 8.15955755, 7.41738396, 11.44371933,

12.29297755, 11.78152324, 9.64085199, 6.68190225, 9.78463752,

10.15529128, 13.34990057, 7.78798847, 9.45126717, 9.08420505,

13.60154678, 19.9074499 , 7.99920078, 7.9877291 , 12.22407467,

5.23061326, 9.03185859, 9.84236027, 16.24078395, 7.72943272,

5.20272241, 10.08637807, 9.95608141, 9.62580214, 10.384239 ,

13.4927171 , 9.75642436, 11.02057169, 12.29295389, 8.03468201,

10.19702269, 10.39366802, 12.48699151, 6.92783091, 10.74280001,

8.3486369 , 8.61778065, 8.45543388, 8.00677636, 9.71675823,

16.92550805, 10.57953196, 15.90536436, 9.50462285, 9.39988304,

10.32470506, 12.86534747, 12.29079639, 14.74918872, 9.1639895 ,

6.20630441, 13.00872207, 7.54953187, 10.00180916, 9.083734 ,

11.13982846, 10.40601532, 6.04155134, 9.07858903, 12.23620916,

6.89145492, 7.1780126 , 6.69541463, 10.67061781, 8.83159662,

11.00954948, 11.18723069, 9.58537449, 11.6606398 , 10.99727223,

15.94197138, 11.86295159, 10.55422747, 11.84734619, 7.34152748,

11.09309102, 4.65257582, 9.68205642, 10.86540308, 7.02324895,

9.39411833, 11.41375515, 6.32025608, 9.23713509, 9.41794234,

10.08955103, 6.07584093, 12.16137614, 13.98070402, 7.50574666,

11.63939066, 9.87978312, 7.98519933, 7.86956897, 9.09066842,

11.33150975, 5.8932936 , 7.38592484, 8.16104742, 7.13985192,

12.12374737, 9.49445256, 12.6608272 , 10.92822679, 9.64579814,

11.37740016, 13.35655501, 12.03431308, 8.29052643, 15.07767248,

16.97826247, 8.49546962, 9.14262023, 10.89505792, 8.94620052,

8.10005376, 8.2741099 , 9.04242193, 10.80323642, 10.33724187,

10.7569934 , 11.25459692, 7.79289162, 11.82360347, 9.36253889,

9.20535978, 11.33237436, 11.90082395, 7.21549415, 8.98121219,

9.18751815, 10.85940902, 10.89884311, 8.11486849, 11.83199882,

10.81757103, 10.92910856, 12.08135246, 10.71535444, 9.71625622,

9.01781515, 10.95518072, 8.34046289, 10.17978723, 9.2628684 ,

6.27546838, 13.01673973, 9.29323555, 10.88277688, 10.65023675,

7.79813885, 11.3480945 , 13.44888209, 15.81779561, 7.9295 ,

10.48476882, 5.59038454, 8.40947994, 17.62563803, 6.81638847,

7.21469929, 12.84101012, 9.32876132, 18.90467618, 11.63015657,

10.99326775, 12.40854784, 9.10945045, 12.9140317 , 7.8017051 ,

9.90552646, 8.3464963 , 12.88270016, 8.25612957, 9.34584491,

11.65653916, 12.39564448, 10.2772439 , 9.3555932 , 9.34328984,

16.64921469, 11.26978487, 12.18802387, 10.34557299, 10.59100518,

10.179332 , 6.73708151, 12.92492286, 7.80725467, 9.92395643,

10.06050323, 12.38547025, 10.07510473, 9.65976876, 11.51754953,

7.4877671 , 10.78351855, 14.60826713, 19.17650316, 7.60163495,

9.76719767, 11.36471963, 9.02854673, 11.17540692, 12.78651789,

11.96011417, 7.25130786, 10.5343224 , 9.76155361, 13.64379351,

11.22547096, 8.27602732, 15.78794218, 11.02301369, 8.91859135,

10.87261659, 5.83492447, 8.6425455 , 11.97076478, 7.93997707,

12.26801825, 11.96747073, 15.00085408, 10.75919008, 7.84940305,

12.56772493, 9.44675979, 12.64068414, 9.77515558, 7.1381343 ,

11.05597325, 10.1280931 , 11.16712892, 7.8402234 , 5.01479697,

9.35977556, 7.89993382, 11.7162187 , 13.96597243, 8.99518238,

8.83630485, 9.74606374, 7.98103739, 8.55583733, 9.11193498,

9.84097136, 14.59619824, 11.08122319, 5.26162661, 9.36987379,

17.68746302, 9.19687317, 6.97536302, 15.40254755, 10.94042066,

8.11738171, 12.2744849 , 9.18371758, 8.6557194 , 6.09011959,

11.40208474, 7.68337939, 12.49805976, 10.85065987, 7.37385646,

11.63132344, 11.15538826, 9.4669364 , 9.62369785, 5.79947458,

11.51770471, 8.78142722, 9.68851357, 7.86140492, 10.11144881,

8.61144343, 8.41492208, 7.77951366, 9.61522353, 10.4936152 ,

12.59717806, 8.49685449, 13.74886065, 10.38168 , 10.58203164,

8.96082279, 7.09674879, 9.76569836, 6.63030467, 8.77986736,

18.81281935, 11.99051575, 7.83652082, 9.226031 , 6.16416627,

10.94455769, 13.14586874, 11.66712638, 11.30215648, 9.25448888,

7.26095181, 11.2501062 , 7.64633084, 8.60575428, 9.98687751,

6.70614951, 11.20633964, 9.05142488, 6.11967056, 10.48426928,

9.70077869, 9.4378062 , 5.26350078, 8.46643588, 12.08022381,

11.54467485, 5.37634476, 9.96782175, 10.60255086, 8.92740429,

9.41477621, 11.20247256, 11.18792496, 8.93558577, 7.08613717,

11.0009911 , 10.92435509, 9.63686967, 10.33608233, 11.34742638,

9.66588006, 11.44518221, 9.52292647, 13.22462549, 11.29187232,

8.65570115, 8.1475651 , 6.45539076, 11.15749119, 12.21130351,

9.61298 , 13.15361983, 11.12240773, 8.98401594, 9.30281553,

8.7786281 , 20.823057 , 9.77559007, 6.97664278, 8.02486076,

5.72071161, 5.25241178, 9.15342537, 8.7139211 , 7.38888852,

10.77588248, 12.55690906, 10.56627908, 10.00154538, 7.51361598,

10.5813927 , 10.11686683, 7.22117183, 7.13356489, 4.5775051 ,

10.68352071, 6.99739725, 8.41520956, 5.44565067, 11.80265579,

7.9881176 , 12.67148822, 10.90697897, 9.08923154, 10.14366775,

13.41589911, 6.57982745, 10.67212561, 7.96560911, 8.82358657,

4.77901726, 6.15115267, 11.24701873, 10.65799593, 6.12410353,

11.85295131, 14.13405726, 8.21581124, 10.38155575, 7.52322851,

9.65919407, 10.50570684, 10.53815013, 11.03105652, 14.94420309,

8.1629979 , 13.20620937, 13.76807838, 10.06643372, 6.08667484,

7.96640174, 9.39281163, 8.9050128 , 9.1292816 , 9.45801676,

9.42507151, 10.73687597, 10.14317479, 8.55714844, 11.25360358,

9.96226645, 8.66100729, 10.86426286, 7.30857096, 12.59277927,

7.46577829, 8.93329457, -3.4842505 , 9.99035938, 8.63918657,

12.32485852, 12.1632845 , 10.44428429, 10.67203842, 8.89267787,

9.78142809, 13.97144567, 8.24050808, 8.71524195, 12.38640756,

11.18986679, 8.18474635, 6.60605893, 9.26527952, 9.51550508,

4.77282357, 12.47965749, 12.34329938, 8.38642711, 11.82341819,

10.18619148, 9.86825241, 9.36663152, 15.78962765, 11.07421522,

13.98760595, 10.22976508, 9.30838983, 13.96711829, 11.82822129,

19.02810829, 9.65306974, 9.46423954, 14.92680808, 11.64748294,

7.94239092, 6.98331916, 8.8509933 , 11.85385203, 8.2656887 ,

11.36743669, 8.90454322, 10.38098675, 7.70083638, 9.66900187,

8.30679606, 7.56821231, 10.15969955, 9.49467134, 8.16319961,

12.84097277, 8.10548003, 10.62392686, 11.86885538, 10.2362116 ,

11.83076288, 9.32783045, 10.34572183, 10.62568974, 8.97676071,

9.13508074, 17.69207229, 6.99754881, 7.85010695, 10.90888864,

13.18201489, 12.00545114, 9.27436134, 9.17594523, 9.5500056 ,

8.63693289, 10.13830113, 8.77430855, 6.70722936, 9.76476193,

10.97166462, 6.2434988 , 10.55199796, 13.55224476, 8.5792102 ,

9.18759628, 9.44127845, 9.8394762 , 8.44224265, 11.36534893,

8.83745096, 8.39855326, 9.70755212, 14.5637696 , 10.55770228,

8.64156549, 7.95558695, 8.27908153, 9.07302743, 7.6430385 ,

12.5282226 , 10.60750098, 8.56517665, 12.41248015, 9.12142511,

8.03734112, 15.08509198, 6.05827171, 9.02007634, 6.02558078,

11.63680507, 9.97324886, 15.03187662, 8.51771273, 10.04145538,

13.51888784, 9.10029217, 9.42349388, 9.04886862, 8.52978087,

10.72524213])

**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)

Mean Squared error of training set :4.949028

Mean Squared error of testing set :4.785948

# 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)

0.41877128928053997

**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)

0.48693687494342397

**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)

#Mesure the performance using Metrics

dtr.score(x\_train, y\_train)

0.07943400002124779

**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)

0.512187188782296