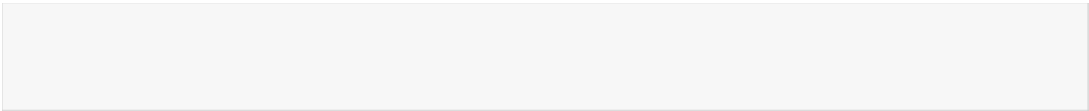
# TEAM ID : PNT2022TMID31193



PROJECT TITLE : RETAIL STORE STOCK INVENTORY ANALYTICS ASSIGNMENT DATE : 01.11.22

STUDENT NAME : GEETHA G

STUDENT ROLL NUMBER :621519106023

# Download the dataset

In [265]:

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

# LOAD THE DATASET

In [266]:

df = pd.read\_csv('abalone.csv')

In [267]:

df.head Out[267]:

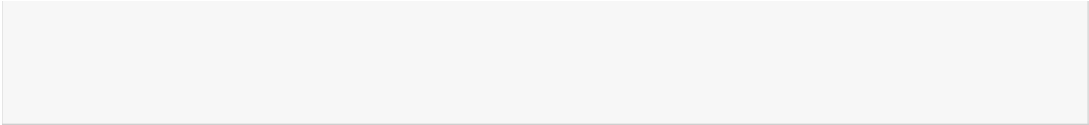
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| <bound  eight | method NDFrame.head  \ | | | | Of | Sex | Length Diameter | Height | Whole | weight | Shucked | w |
| 0 | M | 0.455 | | 0.365 | 0.095 | 0.5140 | | 0.2245 | | | |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

[4177 rows x 9 columns]>

In [268]:

Age=1.5+df.Rings





df["Age"]=Age

df=df.rename(columns = {'whole weight':'whole\_weight','Shucked weight':'Shucked\_weight', 'Viscera weight':'Viscera\_weight','Shell weight':'Shell\_weight'}) df=df.drop(columns=["Rings"],axis=1)

df.head()

Out[268]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked\_weight** | **Viscera\_weight** | **Shell\_weight** | **Age** |
| **0** M | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 16.5 |
| **1** M | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 8.5 |
| **2** F | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 10.5 |
| **3** M | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 11.5 |
| **4** I | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 8.5 |

In [269]:



df.tail()

Out[269]:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked\_weight** | **Viscera\_weight** | **Shell\_weight** | **Age** |
| **4172** | F | 0.565 | 0.450 | 0.165 | 0.8870 | 0.3700 | 0.2390 | 0.2490 | 12.5 |
| **4173** | M | 0.590 | 0.440 | 0.135 | 0.9660 | 0.4390 | 0.2145 | 0.2605 | 11.5 |
| **4174** | M | 0.600 | 0.475 | 0.205 | 1.1760 | 0.5255 | 0.2875 | 0.3080 | 10.5 |
| **4175** | F | 0.625 | 0.485 | 0.150 | 1.0945 | 0.5310 | 0.2610 | 0.2960 | 11.5 |
| **4176** | M | 0.710 | 0.555 | 0.195 | 1.9485 | 0.9455 | 0.3765 | 0.4950 | 13.5 |

# Perform Below Visualizations Univariate Analysis

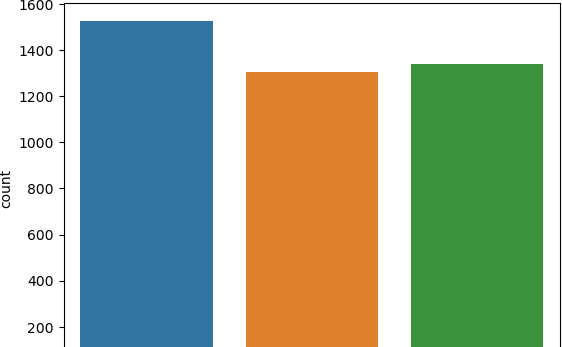
In [270]:



sns.countplot(x='Sex',data=df)

Out[270]:

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







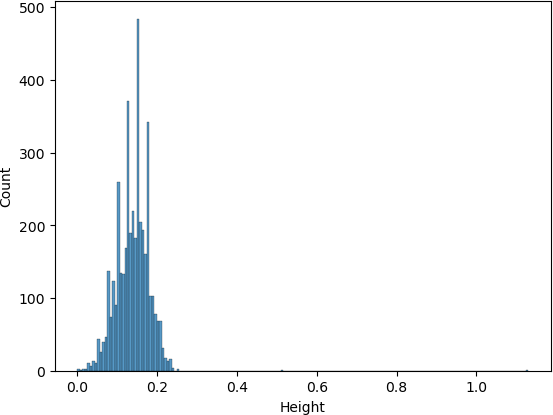
In [271]:



sns.histplot(df["Height"])

Out[271]:

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



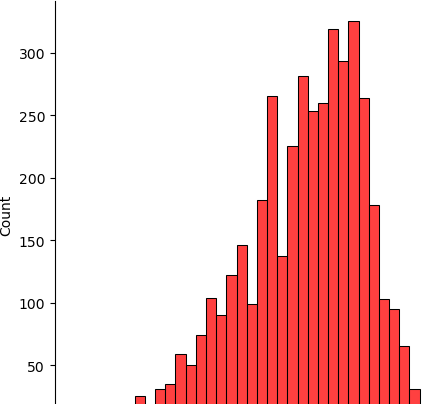
In [272]:

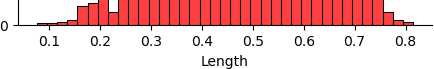


sns.displot(df["Length"],color='red')

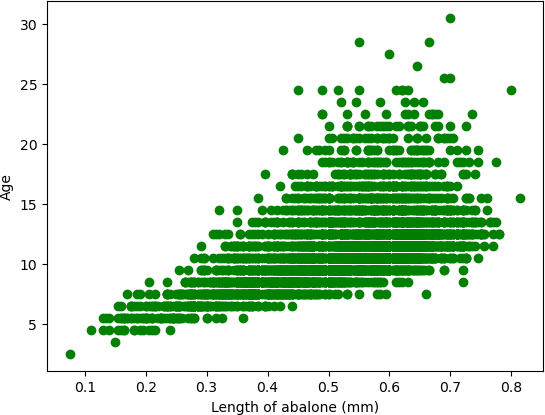
Out[272]:

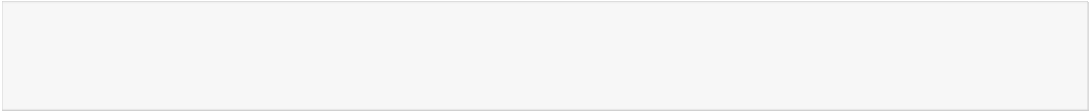
<seaborn.axisgrid.FacetGrid at 0x1af5e2f7820>





# Bi-Variate Analysis

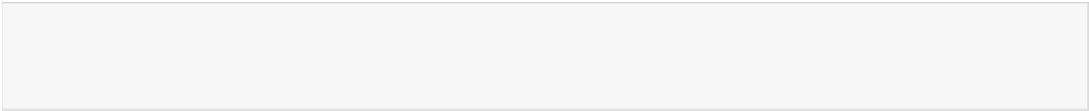
In [273]:



plt.scatter(df['Length'],df['Age'],c='green') plt.xlabel('Length of abalone (mm)') plt.ylabel('Age')

plt.show()

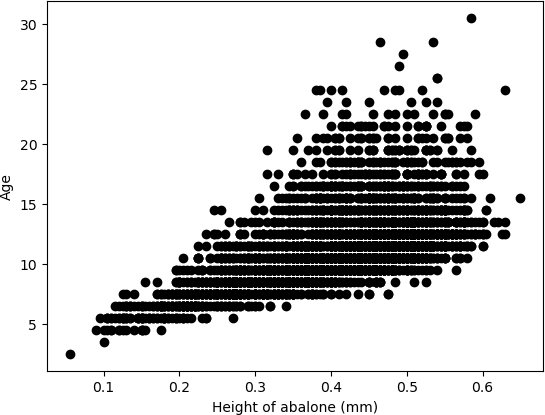
In [274]:

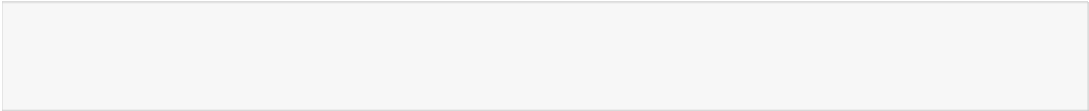


plt.scatter(df['Height'],df['Age'],c='purple') plt.xlabel('Height of abalone (mm)') plt.ylabel('Age')

plt.show()



In [275]:

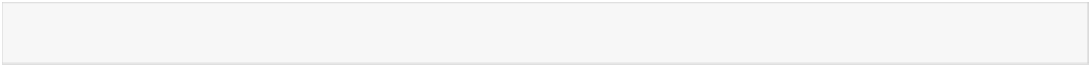


plt.scatter(df['Diameter'],df['Age'],c='black') plt.xlabel('Height of abalone (mm)') plt.ylabel('Age')

plt.show()

# Multi-Variate Analysis

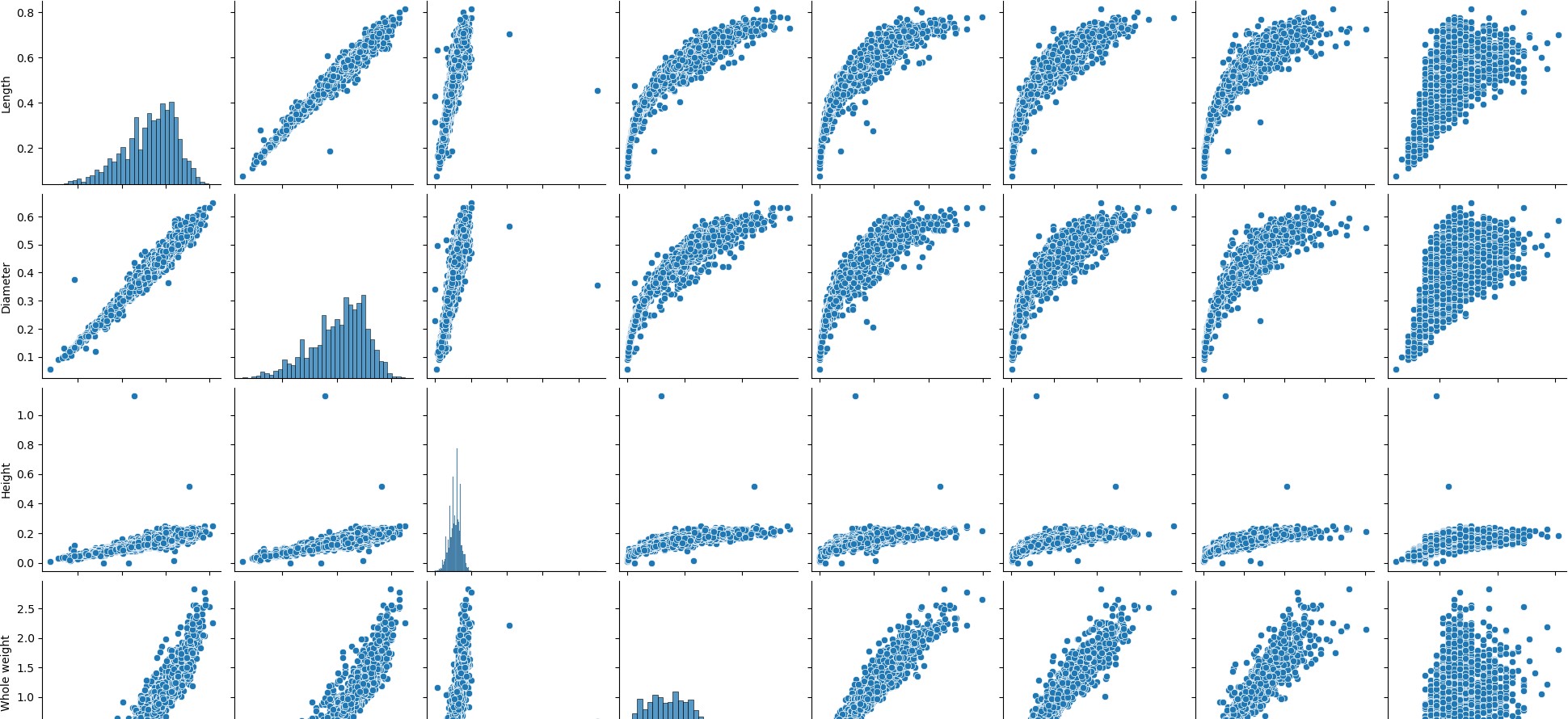
In [276]:

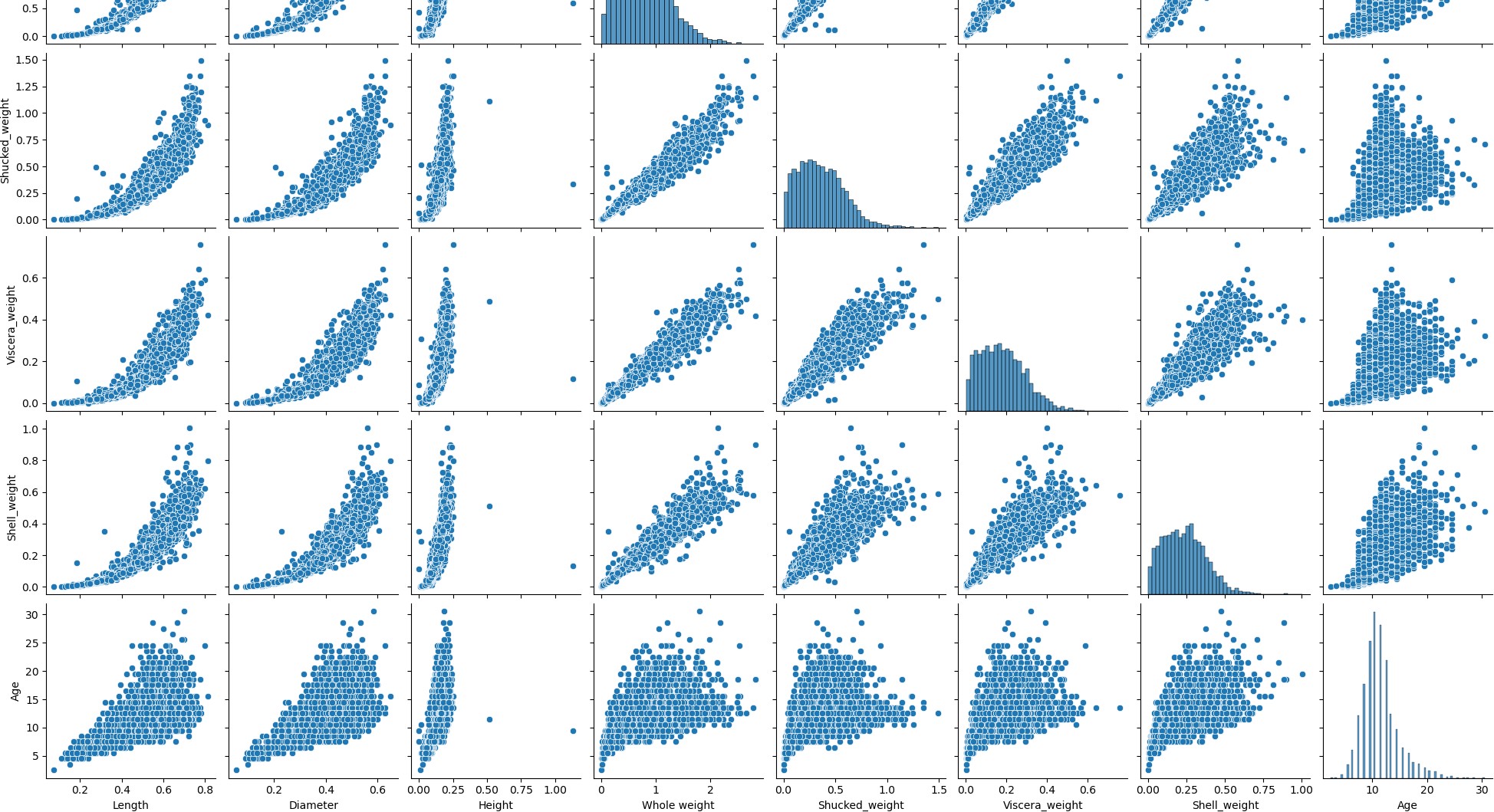


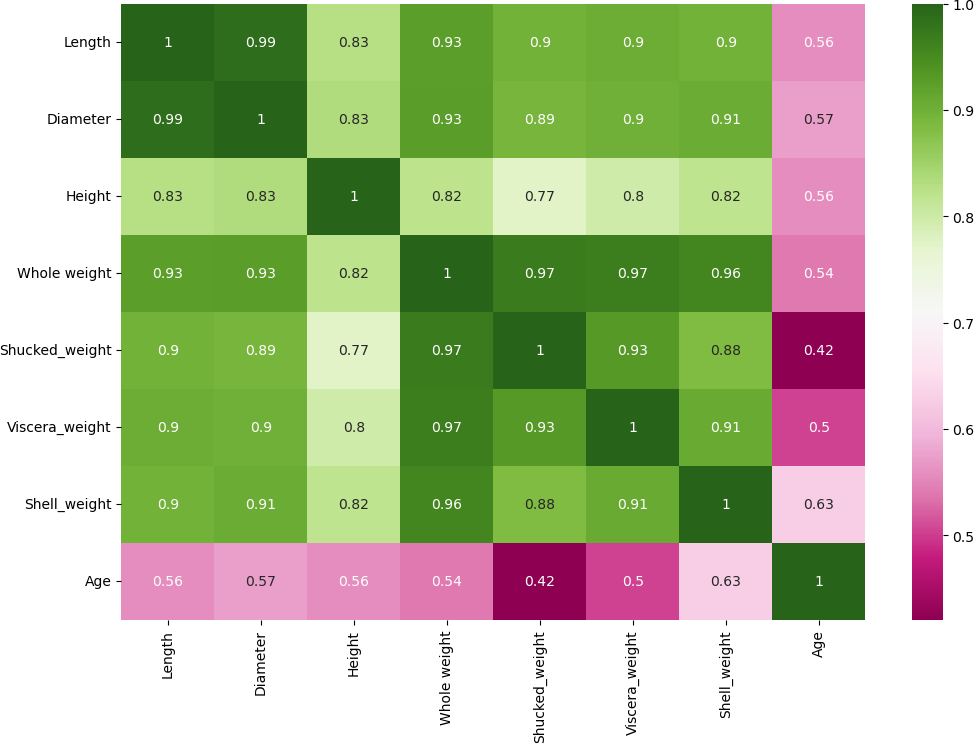
numerical\_features = df.select\_dtypes(include = [np.number]).columns sns.pairplot(df[numerical\_features])

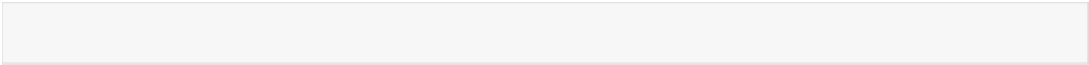
Out[276]:

<seaborn.axisgrid.PairGrid at 0x1af61732d00>





In [277]:



plt.figure(figsize=(12,8)); sns.heatmap(df.corr(),cmap='PiYG',annot=True);

# Perform descriptive statistics on the dataset

In [278]:



df.describe()

Out[278]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked\_weight** | **Viscera\_weight** | **Shell\_weight** | **Age** |
| **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 | 11.433684 |
| **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 | 2.500000 |
| **25%** | 0.450000 | 0.350000 | 0.115000 | 0.441500 | 0.186000 | 0.093500 | 0.130000 | 9.500000 |
| **50%** | 0.545000 | 0.425000 | 0.140000 | 0.799500 | 0.336000 | 0.171000 | 0.234000 | 10.500000 |
| **75%** | 0.615000 | 0.480000 | 0.165000 | 1.153000 | 0.502000 | 0.253000 | 0.329000 | 12.500000 |
| **max** | 0.815000 | 0.650000 | 1.130000 | 2.825500 | 1.488000 | 0.760000 | 1.005000 | 30.500000 |

# Check for Missing values and deal with them

In [279]:



df.isnull().sum()

Out[279]:

Sex 0

Length 0

Diameter 0

Height 0

Whole weight 0

Shucked\_weight 0

Viscera\_weight 0

Shell\_weight 0

Age 0

dtype: int64

# Find the outliers and replace them outliers

In [280]:



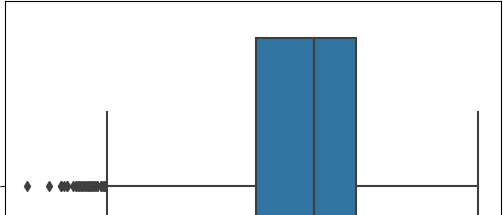
sns.boxplot(df['Length'])

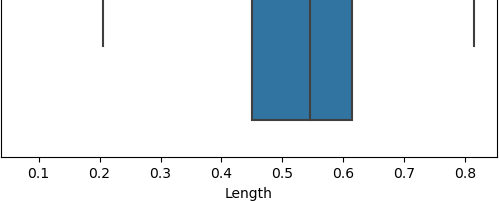
C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res ult in an error or misinterpretation.

warnings.warn(

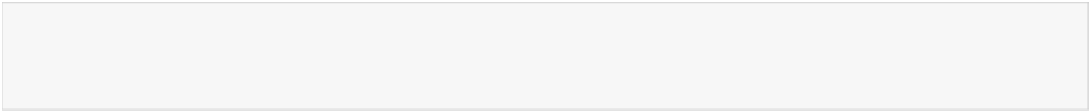
Out[280]:

<AxesSubplot:xlabel='Length'>





In [281]:



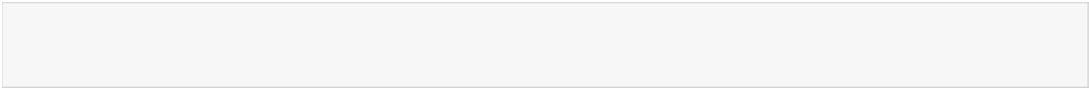
q1 = df['Length'].quantile(0.25) q2 = df['Length'].quantile(0.75) iqr = q2-q1

q1, q2,iqr

Out[281]:

(0.45, 0.615, 0.16499999999999998)

In [282]:

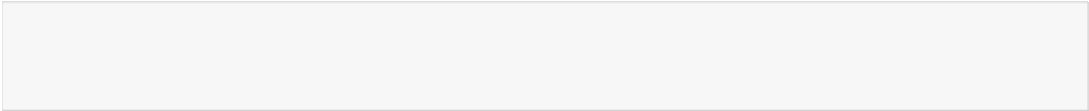


upper\_limit = q2+(1.5\*iqr) lower\_limit = q1-(1.5\*iqr) lower\_limit, upper\_limit

Out[282]:

(0.20250000000000004, 0.8624999999999999)

In [283]:

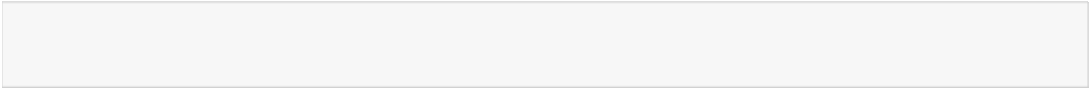


new\_df = df.loc[(df['Length'] <= upper\_limit) & (df['Length'] >= lower\_limit)] print('before removing outliers:', len(df))

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

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

In [284]:



new\_df = df.copy()

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

In [285]:



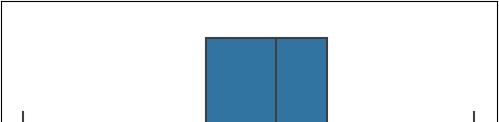
sns.boxplot(new\_df['Length'])

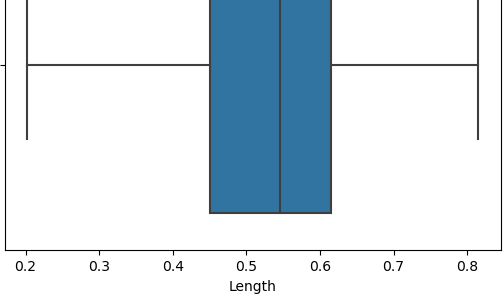
C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res ult in an error or misinterpretation.

warnings.warn(

Out[285]:

<AxesSubplot:xlabel='Length'>





In [286]:



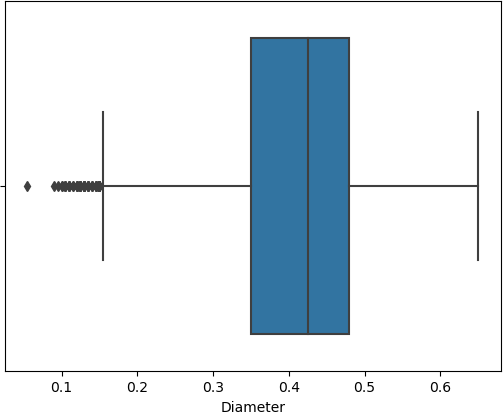
sns.boxplot(df['Diameter'])

C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res ult in an error or misinterpretation.

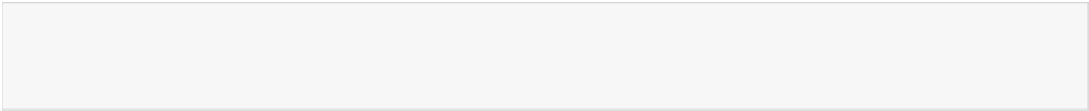
warnings.warn(

Out[286]:

<AxesSubplot:xlabel='Diameter'>



In [287]:



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

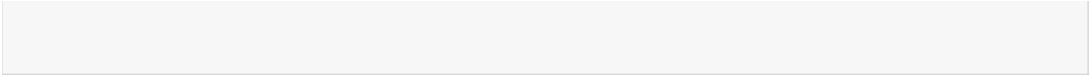
q1, q2, iqr

Out[287]:

(0.35, 0.48, 0.13)

In [288]:

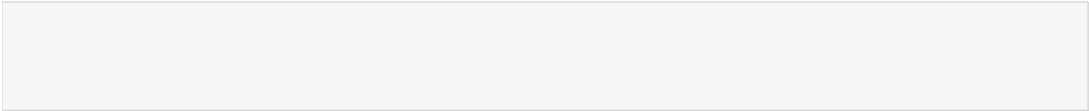




upper\_limit = q2 + (1.5\*iqr) lower\_limit = q1 - (1.5\*iqr) lower\_limit, upper\_limit

Out[288]: (0.15499999999999997, 0.675)

In [289]:

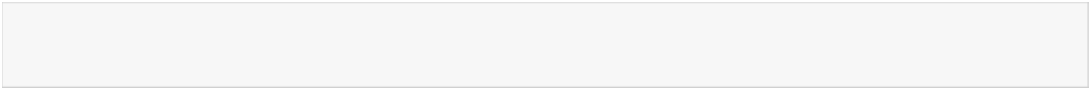


new\_df = df.loc[(df['Diameter'] <= upper\_limit) & (df['Diameter'] >= lower\_limit)] print('before removing outliers :', len(df))

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

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

In [290]:



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

In [291]:



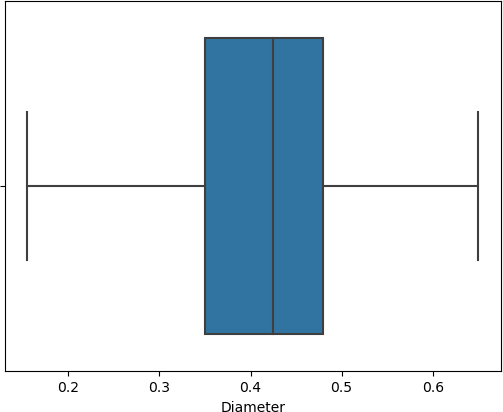
sns.boxplot(new\_df['Diameter'])

C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res ult in an error or misinterpretation.

warnings.warn(

Out[291]:

<AxesSubplot:xlabel='Diameter'>



In [292]:



sns.boxplot(df['Height'])

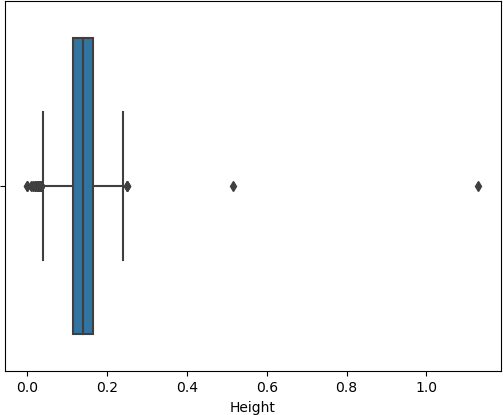
C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res



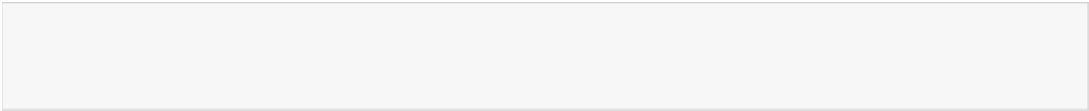
ult in an error or misinterpretation. warnings.warn(

Out[292]:

<AxesSubplot:xlabel='Height'>



In [293]:



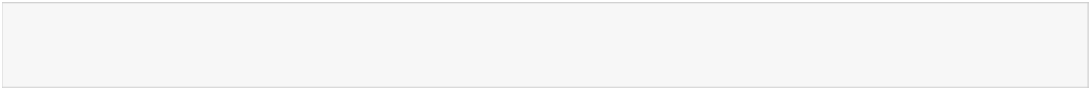
q1 = df['Height'].quantile(0.25) q2 = df['Height'].quantile(0.75) iqr = q2-q1

q1, q2, iqr

Out[293]:

(0.115, 0.165, 0.05)

In [294]:

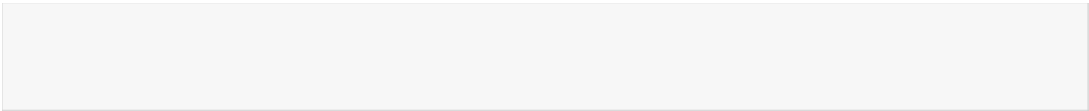


upper\_limit = q2 + (1.5\*iqr) lower\_limit = q1 - (1.5\*iqr) lower\_limit, upper\_limit

Out[294]:

(0.039999999999999994, 0.24000000000000002)

In [295]:

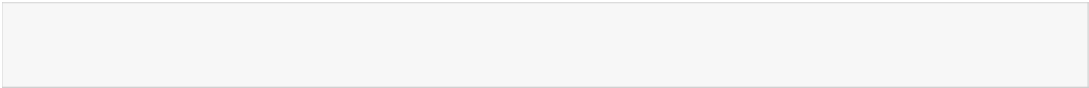


new\_df = df.loc[(df['Height'] <= upper\_limit) & (df['Height'] >= lower\_limit)] print('before removing outliers :', len(df))

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

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

In [296]:



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

In [297]:





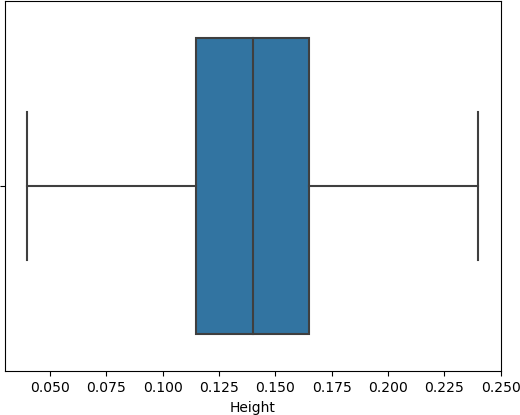
sns.boxplot(new\_df['Height'])

C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res ult in an error or misinterpretation.

warnings.warn(

Out[297]:

<AxesSubplot:xlabel='Height'>



In [298]:



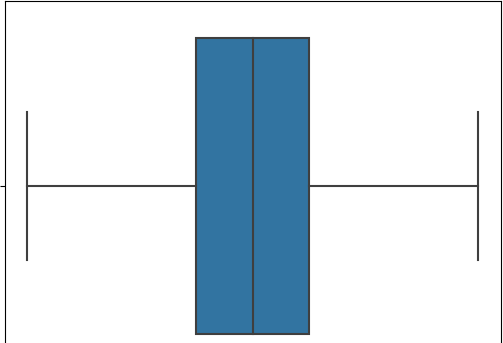
sns.boxplot(new\_df['Height'])

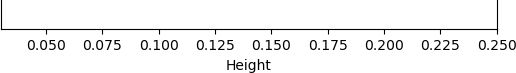
C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res ult in an error or misinterpretation.

warnings.warn(

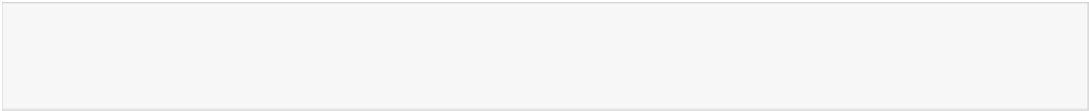
Out[298]:

<AxesSubplot:xlabel='Height'>





In [299]:



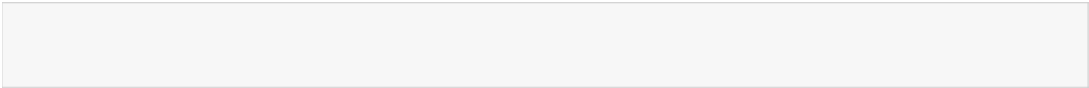
q1 = df['Whole weight'].quantile(0.25) q2 = df['Whole weight'].quantile(0.75) iqr = q2-q1

q1, q2, iqr

Out[299]:

(0.4415, 1.153, 0.7115)

In [300]:

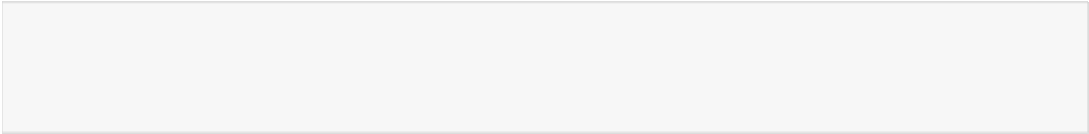


upper\_limit = q2 + (1.5\*iqr) lower\_limit = q1 - (1.5\*iqr) lower\_limit, upper\_limit

Out[300]:

(-0.62575, 2.22025)

In [301]:



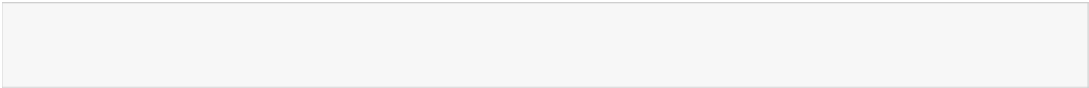
new\_df = df.loc[(df['Whole weight'] <= upper\_limit) & (df['Whole weight'] >= lower\_limit

)]

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

In [302]:



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

In [303]:



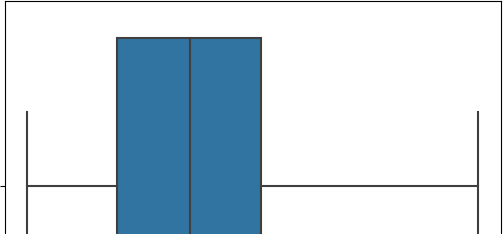
sns.boxplot(new\_df['Whole weight'])

C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res ult in an error or misinterpretation.

warnings.warn(

Out[303]:

<AxesSubplot:xlabel='Whole weight'>





In [304]:



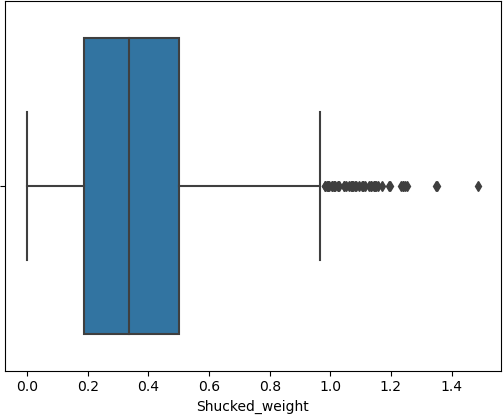
sns.boxplot(df['Shucked\_weight'])

C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res ult in an error or misinterpretation.

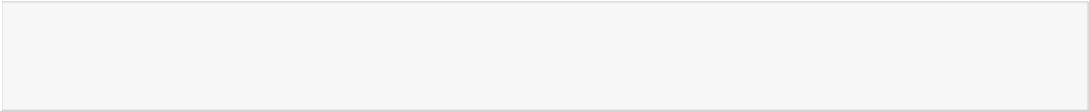
warnings.warn(

Out[304]:

<AxesSubplot:xlabel='Shucked\_weight'>



In [305]:



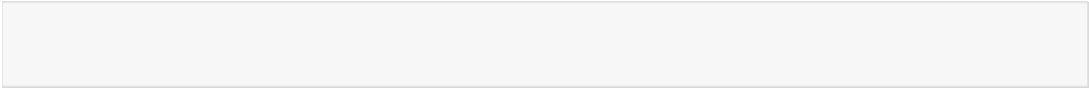
q1 = df['Shucked\_weight'].quantile(0.25) q2 = df['Shucked\_weight'].quantile(0.75) iqr = q2-q1

q1, q2, iqr

Out[305]:

(0.186, 0.502, 0.316)

In [306]:



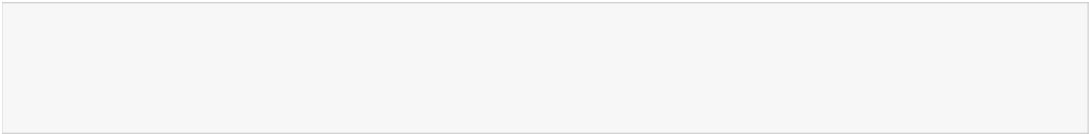
upper\_limit = q2 + (1.5\*iqr) lower\_limit = q1 - (1.5\*iqr) lower\_limit, upper\_limit

Out[306]:

(-0.288, 0.976)



In [307]:

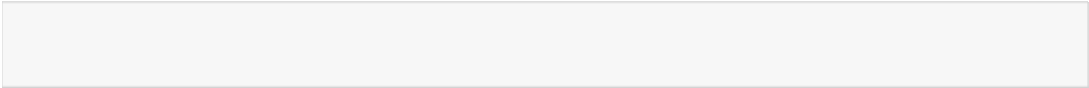


new\_df = df.loc[(df['Shucked\_weight'] <= upper\_limit) & (df['Shucked\_weight'] >= lower\_l imit)]

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

In [308]:



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

In [309]:



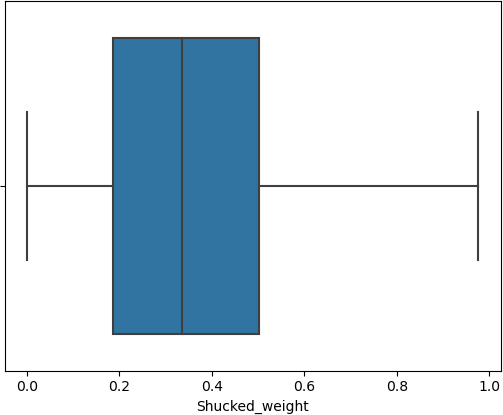
sns.boxplot(new\_df['Shucked\_weight'])

C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res ult in an error or misinterpretation.

warnings.warn(

Out[309]:

<AxesSubplot:xlabel='Shucked\_weight'>



In [310]:



sns.boxplot(df['Viscera\_weight'])

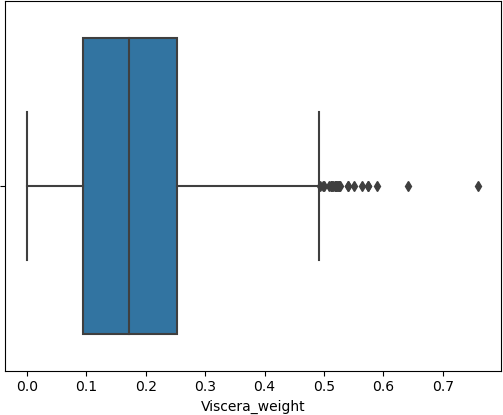
C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will res ult in an error or misinterpretation.

warnings.warn(

Out[310]:

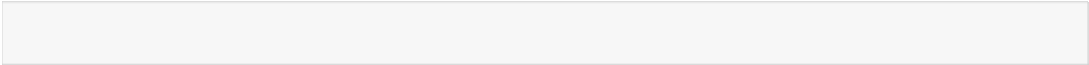
<AxesSubplot:xlabel='Viscera\_weight'>





# Check for Categorical columns and perform encoding

In [311]:



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

Out[311]:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked\_weight** | **Viscera\_weight** | **Shell\_weight** | **Age** |
| **0** | 1 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.1500 | 16.5 |
| **1** | 1 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.0700 | 8.5 |
| **2** | 0 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.2100 | 10.5 |
| **3** | 1 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.1550 | 11.5 |
| **4** | 2 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.0550 | 8.5 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4172** | 0 | 0.565 | 0.450 | 0.165 | 0.8870 | 0.3700 | 0.2390 | 0.2490 | 12.5 |
| **4173** | 1 | 0.590 | 0.440 | 0.135 | 0.9660 | 0.4390 | 0.2145 | 0.2605 | 11.5 |
| **4174** | 1 | 0.600 | 0.475 | 0.205 | 1.1760 | 0.5255 | 0.2875 | 0.3080 | 10.5 |
| **4175** | 0 | 0.625 | 0.485 | 0.150 | 1.0945 | 0.5310 | 0.2610 | 0.2960 | 11.5 |
| **4176** | 1 | 0.710 | 0.555 | 0.195 | 1.9485 | 0.9455 | 0.3765 | 0.4950 | 13.5 |

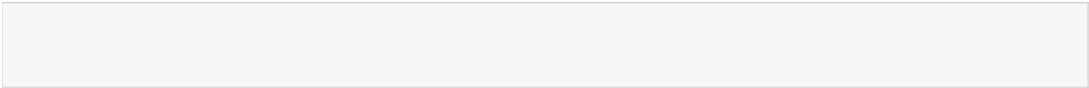
4177 rows × 9 columns

In [312]:



from sklearn.preprocessing import LabelEncoder,OneHotEncoder,StandardScaler

In [313]:



label\_encoder = LabelEncoder()

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

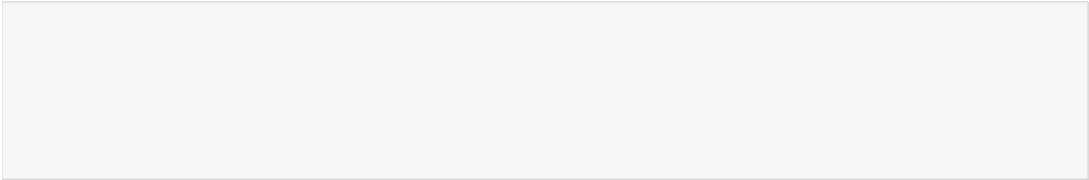
Out[313]:



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked\_weight** | **Viscera\_weight** | **Shell\_weight** | **Age** |
| **0** | 1 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.1500 | 16.5 |
| **1** | 1 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.0700 | 8.5 |
| **2** | 0 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.2100 | 10.5 |
| **3** | 1 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.1550 | 11.5 |
| **4** | 2 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.0550 | 8.5 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4172** | 0 | 0.565 | 0.450 | 0.165 | 0.8870 | 0.3700 | 0.2390 | 0.2490 | 12.5 |
| **4173** | 1 | 0.590 | 0.440 | 0.135 | 0.9660 | 0.4390 | 0.2145 | 0.2605 | 11.5 |
| **4174** | 1 | 0.600 | 0.475 | 0.205 | 1.1760 | 0.5255 | 0.2875 | 0.3080 | 10.5 |
| **4175** | 0 | 0.625 | 0.485 | 0.150 | 1.0945 | 0.5310 | 0.2610 | 0.2960 | 11.5 |
| **4176** | 1 | 0.710 | 0.555 | 0.195 | 1.9485 | 0.9455 | 0.3765 | 0.4950 | 13.5 |

4177 rows × 9 columns

In [314]:



enc = OneHotEncoder(drop='first')

enc\_df = pd.DataFrame(enc.fit\_transform(df[['Sex']]).toarray()) df = df.join(enc\_df)

df.head()

Out[314]:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sex** | **Length** | **Diameter** | **Height** | **Whole weight** | **Shucked\_weight** | **Viscera\_weight** | **Shell\_weight** | **Age** | **0** | **1** |
| **0** 1 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 16.5 | 1.0 | 0.0 |
| **1** 1 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 8.5 | 1.0 | 0.0 |
| **2** 0 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 10.5 | 0.0 | 0.0 |
| **3** 1 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 11.5 | 1.0 | 0.0 |
| **4** 2 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 8.5 | 0.0 | 1.0 |

# Split the data into dependent and independent variables

In [315]:



x.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 4177 entries, 0 to 4176 Data columns (total 11 columns):

# Column Non-Null Count Dtype

* 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. Age 4177 non-null float64
  9. Sex\_F 4177 non-null uint8
  10. Sex\_I 4177 non-null uint8
  11. Sex\_M 4177 non-null uint8 dtypes: float64(8), uint8(3)

memory usage: 273.4 KB



In [316]:



X = x.drop(['Age'], axis = 1)

In [317]:



X.head(2)

Out[317]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Length Diameter** | **Height** | **Whole weight** | **Shucked\_weight** | **Viscera\_weight** | **Shell\_weight** | **Sex\_F** | **Sex\_I** | **Sex\_M** |
| **0** 0.455 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.15 | 0 | 0 | 1 |
| **1** 0.350 0.265  In [318]: | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.07 | 0 | 0 | 1 |



y = x['Age']

In [319]:



y.head(2)

Out[319]:

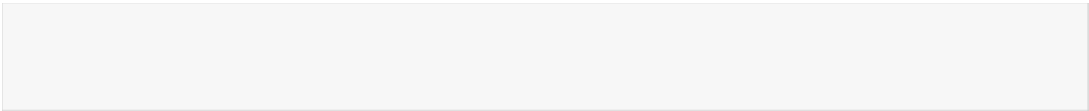
0 16.5

1 8.5

Name: Age, dtype: float64

# Scale the independent variables

In [320]:



scale = StandardScaler()

scaledX = scale.fit\_transform(x)

print(scaledX)

[[-0.57455813 -0.43214879 -1.06442415 ... -0.67483383 -0.68801788

1.31667716]

[-1.44898585 -1.439929 -1.18397831 ... -0.67483383 -0.68801788

1.31667716]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [ 0.05003309 | 0.12213032 | -0.10799087 | ... | 1.48184628 | -0.68801788 |
| -0.75948762] |  |  |  |  |  |
| ... |  |  |  |  |  |
| [ 0.6329849 | 0.67640943 | 1.56576738 | ... | -0.67483383 | -0.68801788 |
| 1.31667716] |  |  |  |  |  |
| [ 0.84118198 | 0.77718745 | 0.25067161 | ... | 1.48184628 | -0.68801788 |
| -0.75948762] |  |  |  |  |  |
| [ 1.54905203 | 1.48263359 | 1.32665906 | ... | -0.67483383 | -0.68801788 |

1.31667716]]

# Split the data into training and testing

In [321]:

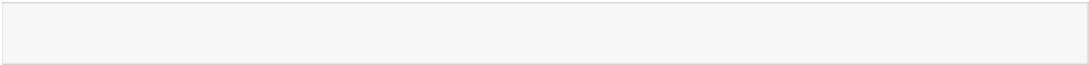


X.shape, y.shape

Out[321]:

((4177, 10), (4177,))

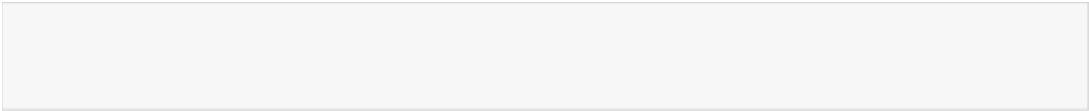
In [322]:



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, random\_state=42)

In [323]:



print(' x\_tain.shape : ',x\_train.shape) print(' y\_tain.shape : ',y\_train.shape) print(' x\_test.shape : ',x\_test.shape) print(' y\_test.shape : ',y\_test.shape)

x\_tain.shape : (3341, 10)

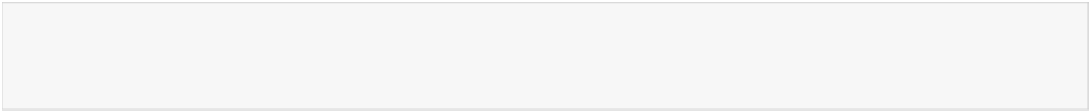
y\_tain.shape : (3341,)

x\_test.shape : (836, 10)

y\_test.shape : (836,)

# 10. Build the Model, 11. Train the Model , 12.Test the Model

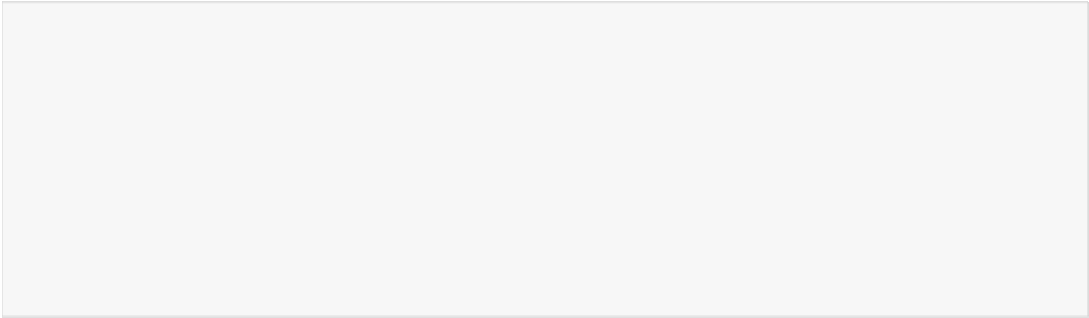
In [324]:



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

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

In [325]:



from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error,make\_scorer from sklearn.model\_selection import RandomizedSearchCV

rf = RandomForestRegressor() param = {

'max\_depth':[3,6,9,12,15],

'n\_estimators':[10,50,100,150,200]

}

rf\_search = RandomizedSearchCV(rf,param\_distributions=param,n\_iter=5,scoring=make\_scorer (mean\_squared\_error),n\_jobs=-1,cv=5,verbose=3)

rf\_search.fit(x\_train, y\_train)

Fitting 5 folds for each of 5 candidates, totalling 25 fits Out[325]:

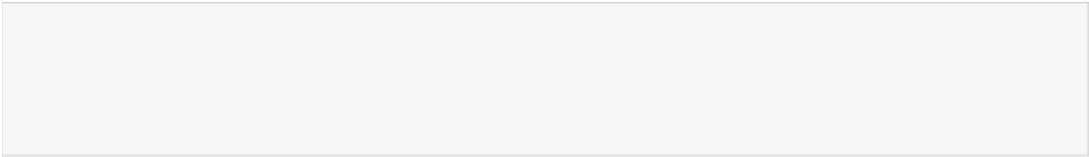
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n\_iter=5, n\_jobs=-1, param\_distributions={'max\_depth': [3, 6, 9, 12, 15],

'n\_estimators': [10, 50, 100, 150,

200]},

scoring=make\_scorer(mean\_squared\_error), verbose=3)

In [326]:



means = rf\_search.cv\_results\_['mean\_test\_score'] params = rf\_search.cv\_results\_['params']

for mean, param in zip(means, params): print("%f with: %r" % (mean,param)) if mean == min(means):

print('Best parameters with the minimum Mean Square Error are:' ,param)

4.664623 with: {'n\_estimators': 200, 'max\_depth': 6}

4.618707 with: {'n\_estimators': 100, 'max\_depth': 15}

4.644619 with: {'n\_estimators': 200, 'max\_depth': 15}

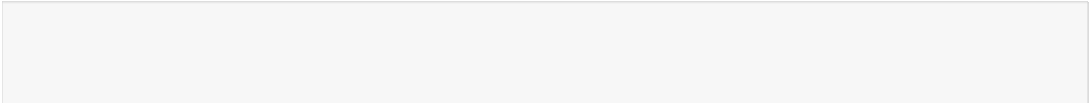
5.677870 with: {'n\_estimators': 150, 'max\_depth': 3}

4.581780 with: {'n\_estimators': 100, 'max\_depth': 9}

Best parameters with the minimum Mean Square Error are: {'n\_estimators': 100, 'max\_depth'

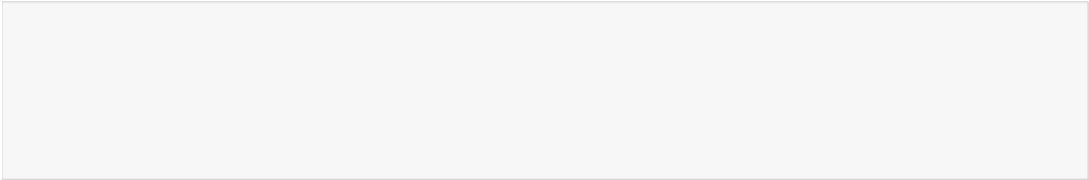
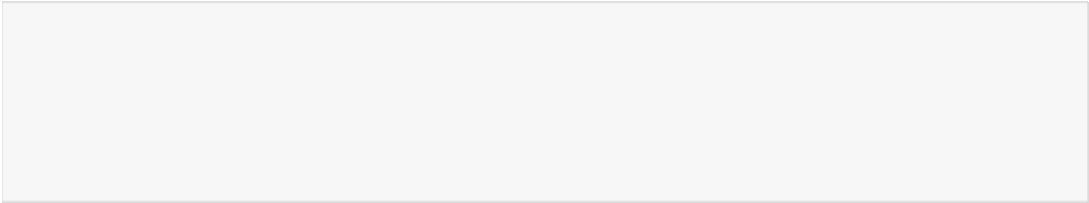
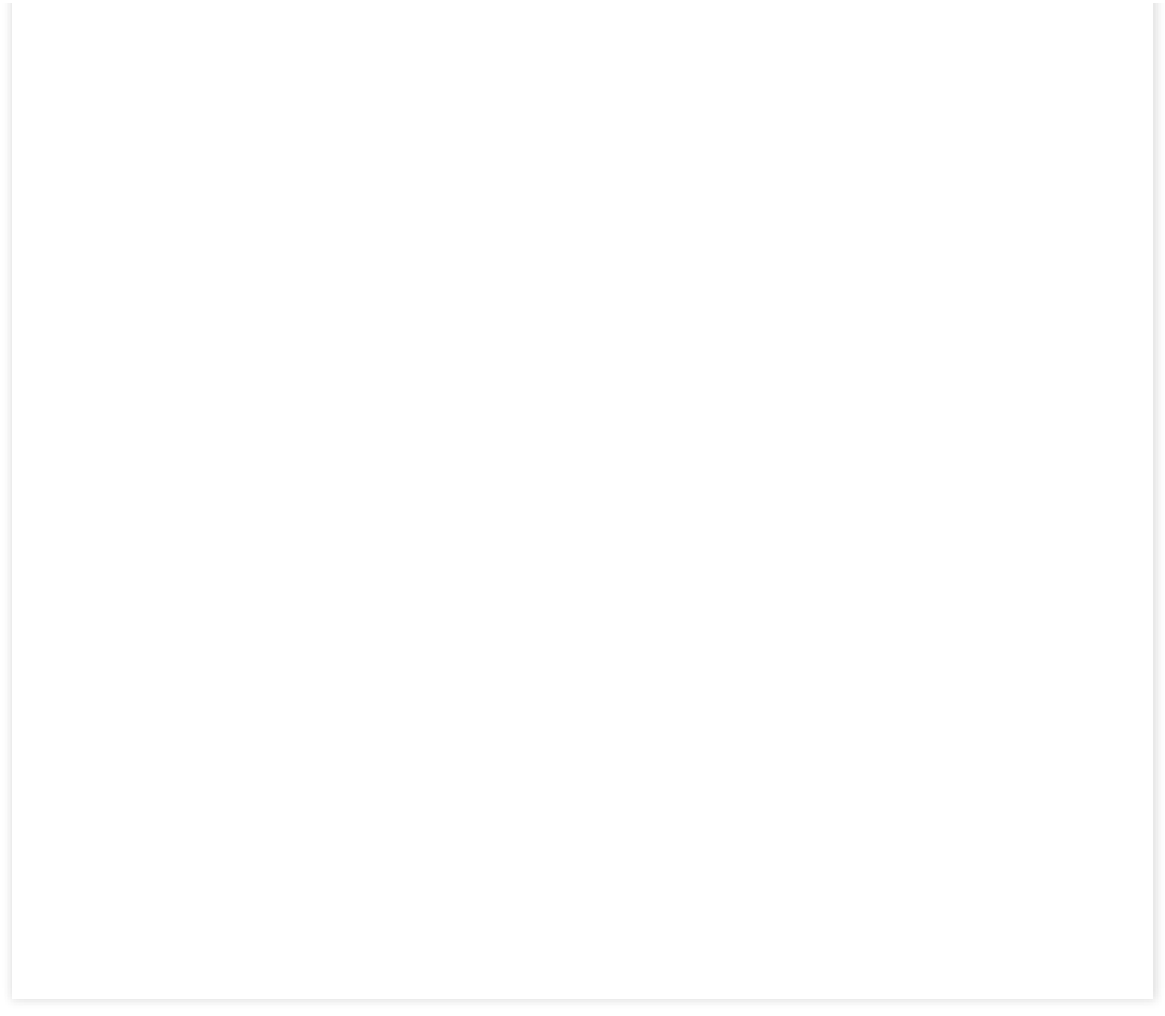
: 9}

In [327]:



rf = RandomForestRegressor(n\_estimators=50,max\_depth=6) rf.fit(x\_train,y\_train)

rf\_pred = rf.predict(x\_test)



# 14. Measure the performance using Metrics

In [328]:

from sklearn import metrics print('Linear Regression :')

print(' ')

print('MAE:',metrics.mean\_absolute\_error(y\_test, lr\_pred)) print('MSE:',metrics.mean\_squared\_error(y\_test, lr\_pred)) print('RMSE:',np.sqrt(metrics.mean\_squared\_error(y\_test, lr\_pred))) print('R2 Score:',metrics.r2\_score(y\_test,lr\_pred))

print('\n\n') Linear Regression :

MAE: 1.5944508821770336

MSE: 4.892375672262822

RMSE: 2.211871531591024

R2 Score: 0.5480572061259404

In [329]:

from sklearn import metrics print('Random Forest Contains:')

print(' ')

print('MAE:',metrics.mean\_absolute\_error(y\_test, rf\_pred)) print('MSE:',metrics.mean\_squared\_error(y\_test, rf\_pred)) print('RMSE:',np.sqrt(metrics.mean\_squared\_error(y\_test, rf\_pred))) print('R2 Score:',metrics.r2\_score(y\_test,rf\_pred))

Random Forest Contains:

MAE: 1.5580369509719958

MSE: 5.025592967383406

RMSE: 2.241783434541215

R2 Score: 0.535750997326301