In [1]:**import** pandas **as** pd **import**
numpy **as** np **import**
matplotlib.pyplot **as** plt **import**
seaborn **as** sns **import** warnings
warnings.filterwarnings('ignore')

1--Dataset Downloaded

2--Load the dataset

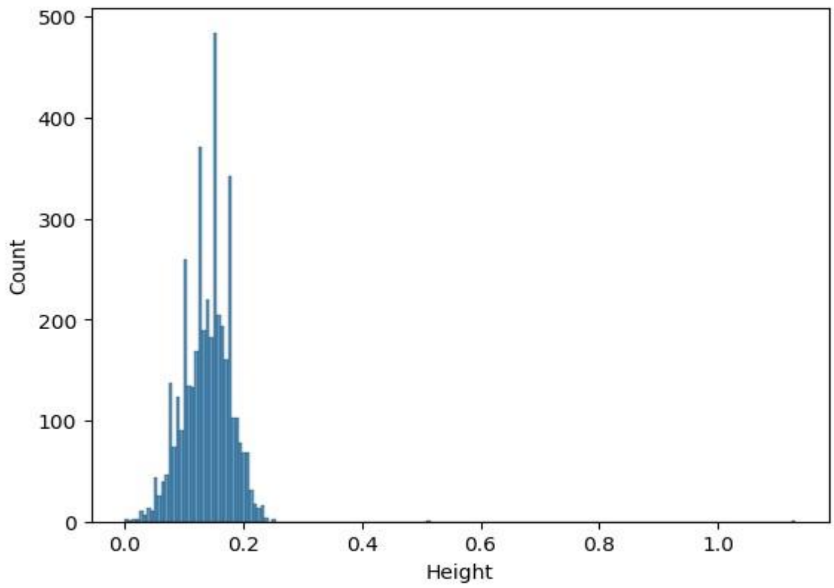
In [2]:data=pd.read_csv(r"D:\Chrome_Downloads\abalone.csv")

3--Visuslization

Univariate analysis

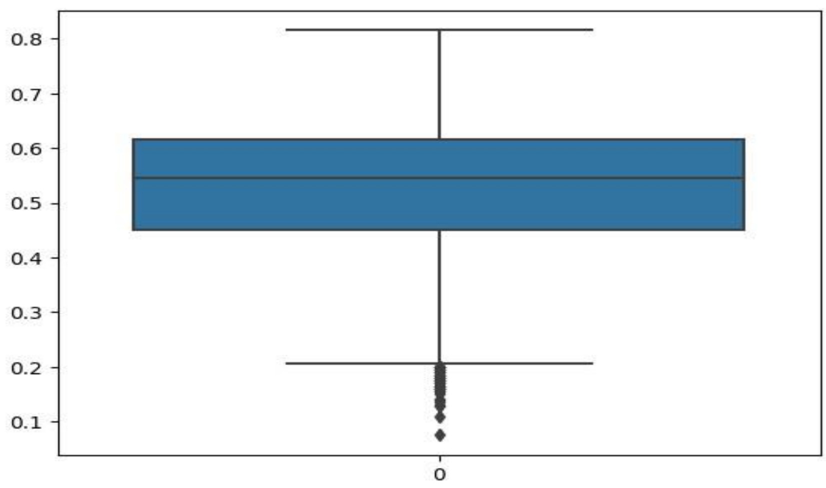
In [3]:sns.histplot(data['Height'])

Out[3]:<AxesSubplot: xlabel='Height', ylabel='Count'>

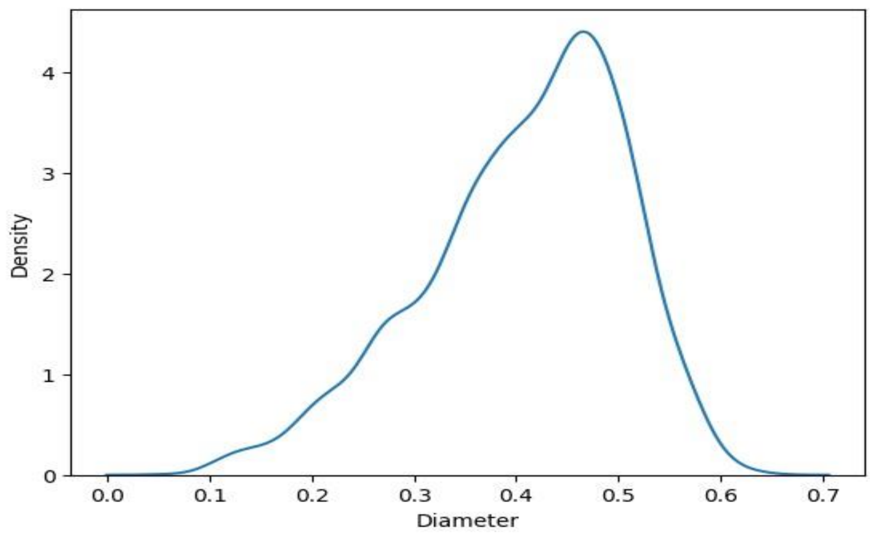


In [4]:sns.boxplot(data['Length'])

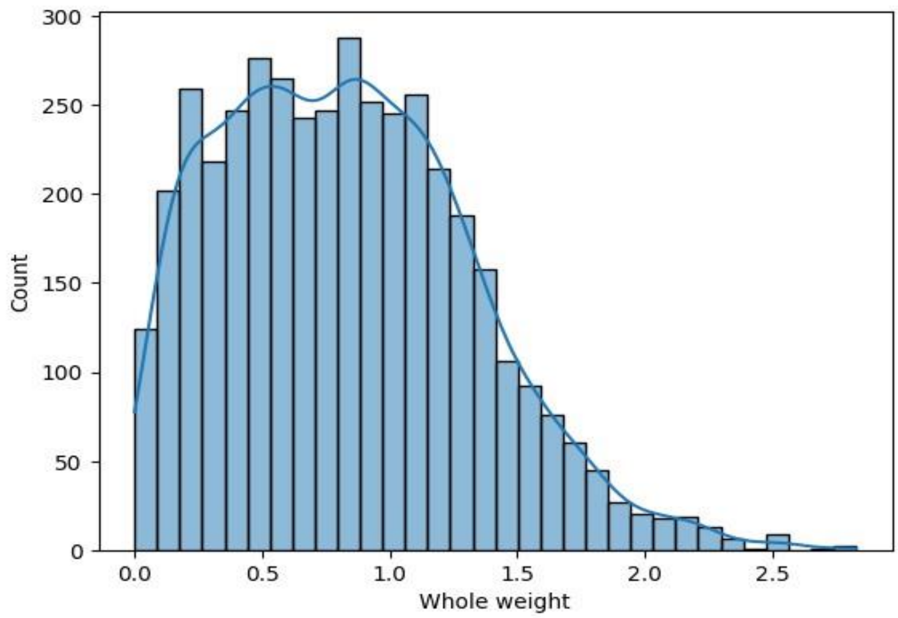
Out[4]:<AxesSubplot: >



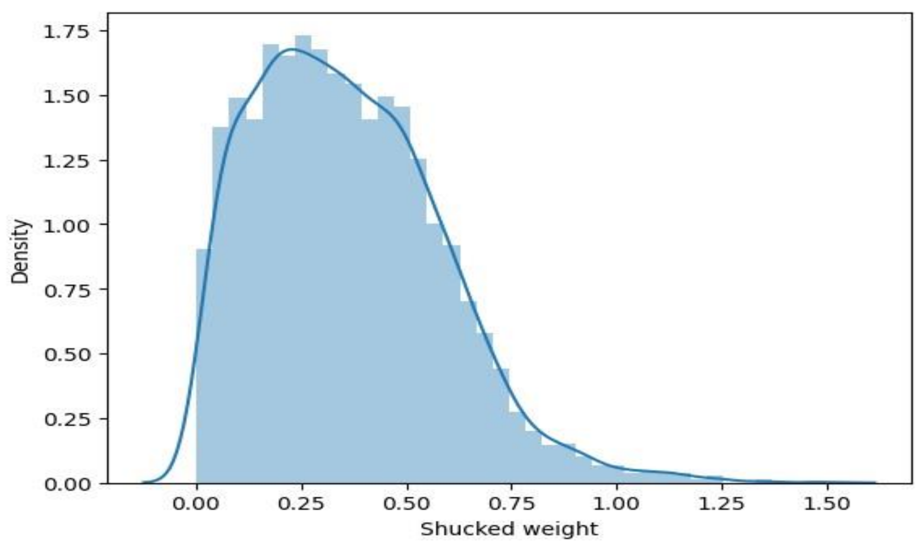
```
In [5]:sns.kdeplot(data['Diameter'])
Out[5]:<AxesSubplot: xlabel='Diameter', ylabel='Density'>
```




```
In [6]:sns.histplot(data['Whole weight'],kde=True)
Out[6]:<AxesSubplot: xlabel='Whole weight', ylabel='Count'>
```



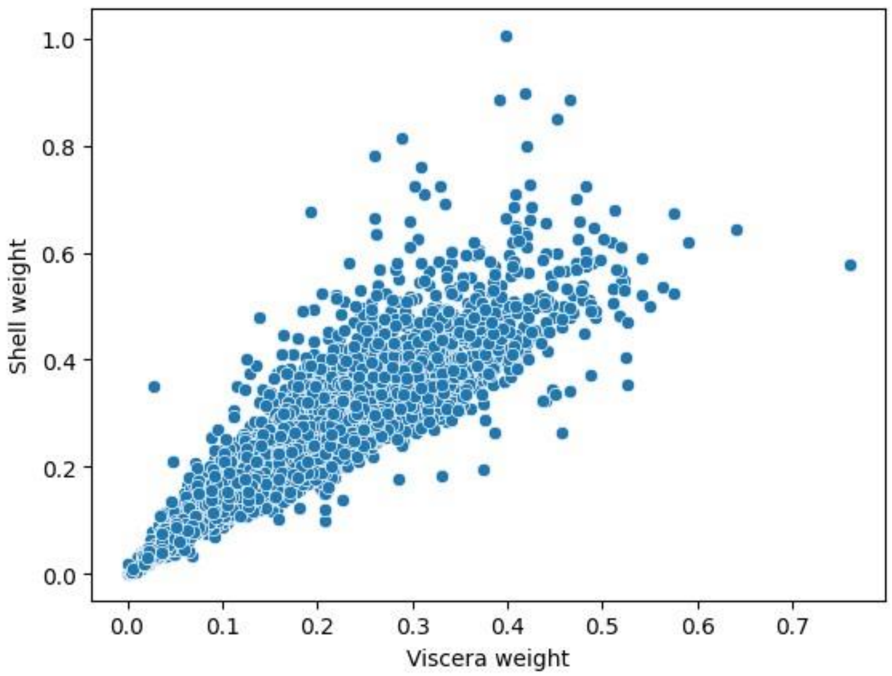

```
In [7]:sns.distplot(data['Shucked weight'],kde=True)
Out[7]:<AxesSubplot: xlabel='Shucked weight', ylabel='Density'>
```



Bivariate Analysis

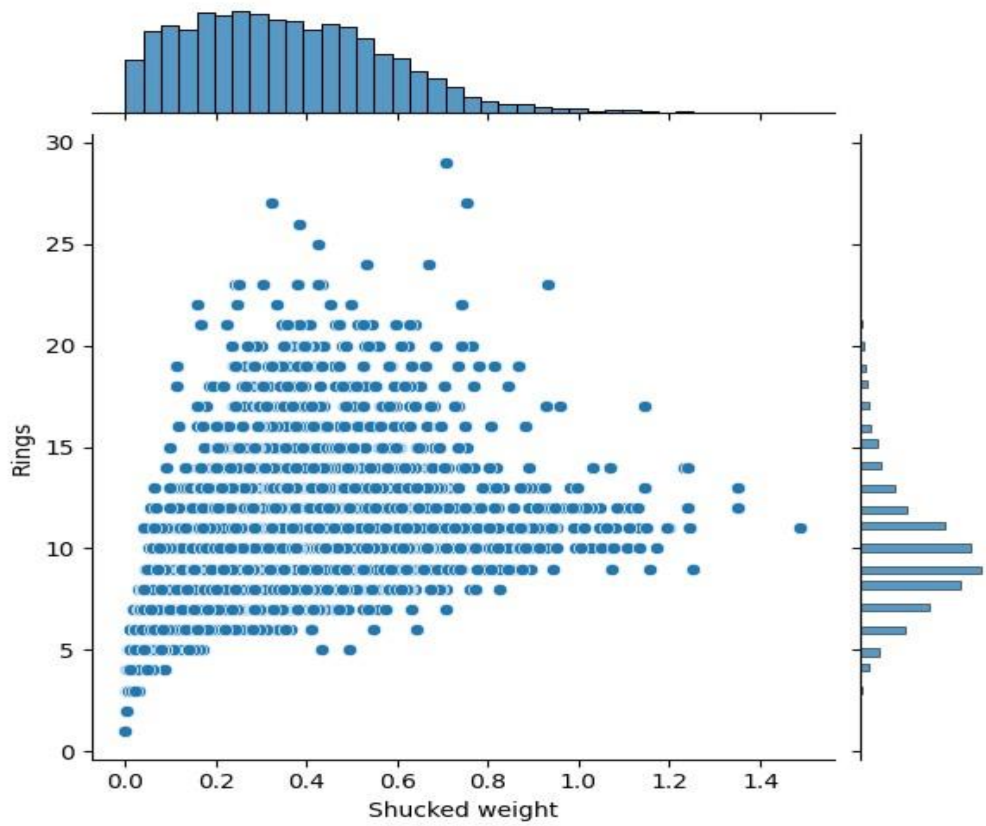
In [8]:sns.scatterplot(data=data,x='Viscera weight', y='Shell weight')

Out[8]:<AxesSubplot: xlabel='Viscera weight', ylabel='Shell weight'>



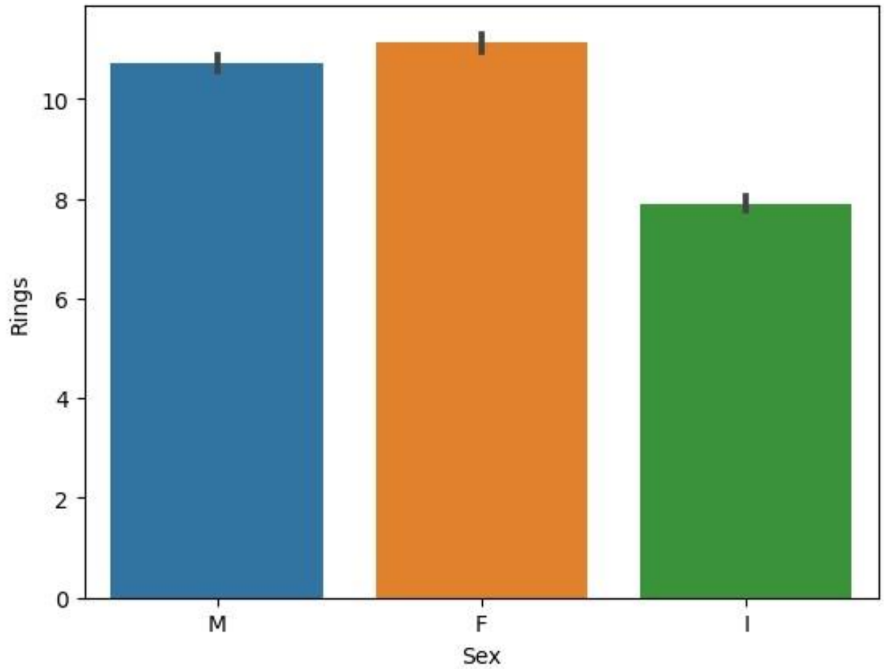
In [9]:sns.jointplot(data=data,x='Shucked weight',y='Rings')

Out[9]:<seaborn.axisgrid.JointGrid at 0x26d098190c0>



In [10]:sns.barplot(data=data,x='Sex',y='Rings')

Out[10]:<AxesSubplot: xlabel='Sex', ylabel='Rings'>



Multivariate Analysis

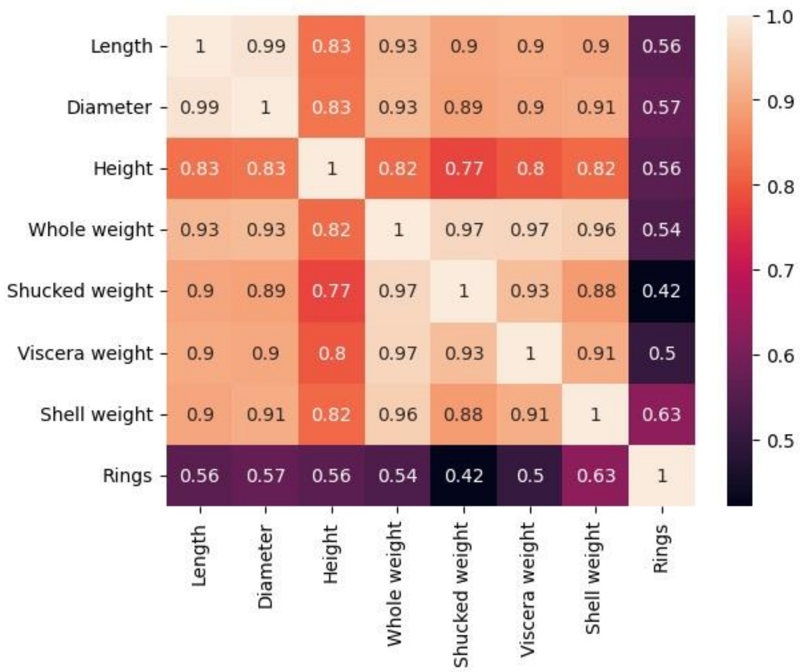
In [11]:data.corr()

Out[11]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Length	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
Diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
Height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
Whole weight	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
Shucked weight	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
Viscera weight	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
Shell weight	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
Rings	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

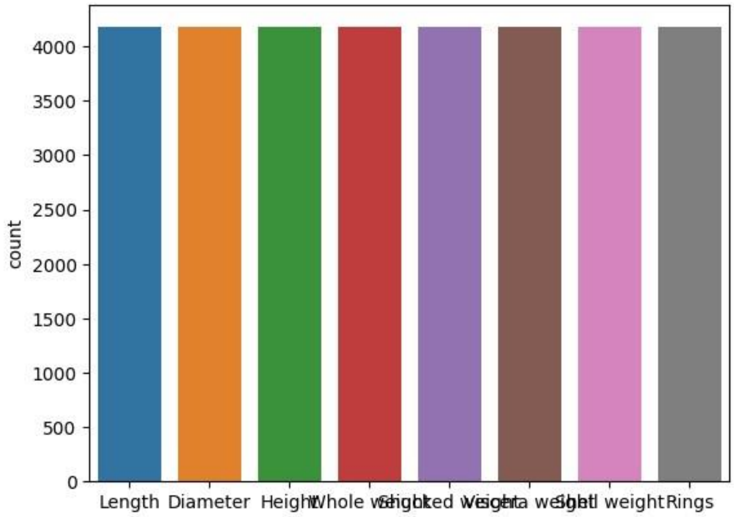
In [12]:sns.heatmap(data.corr(),annot=**True**)

Out[12]:<AxesSubplot: >



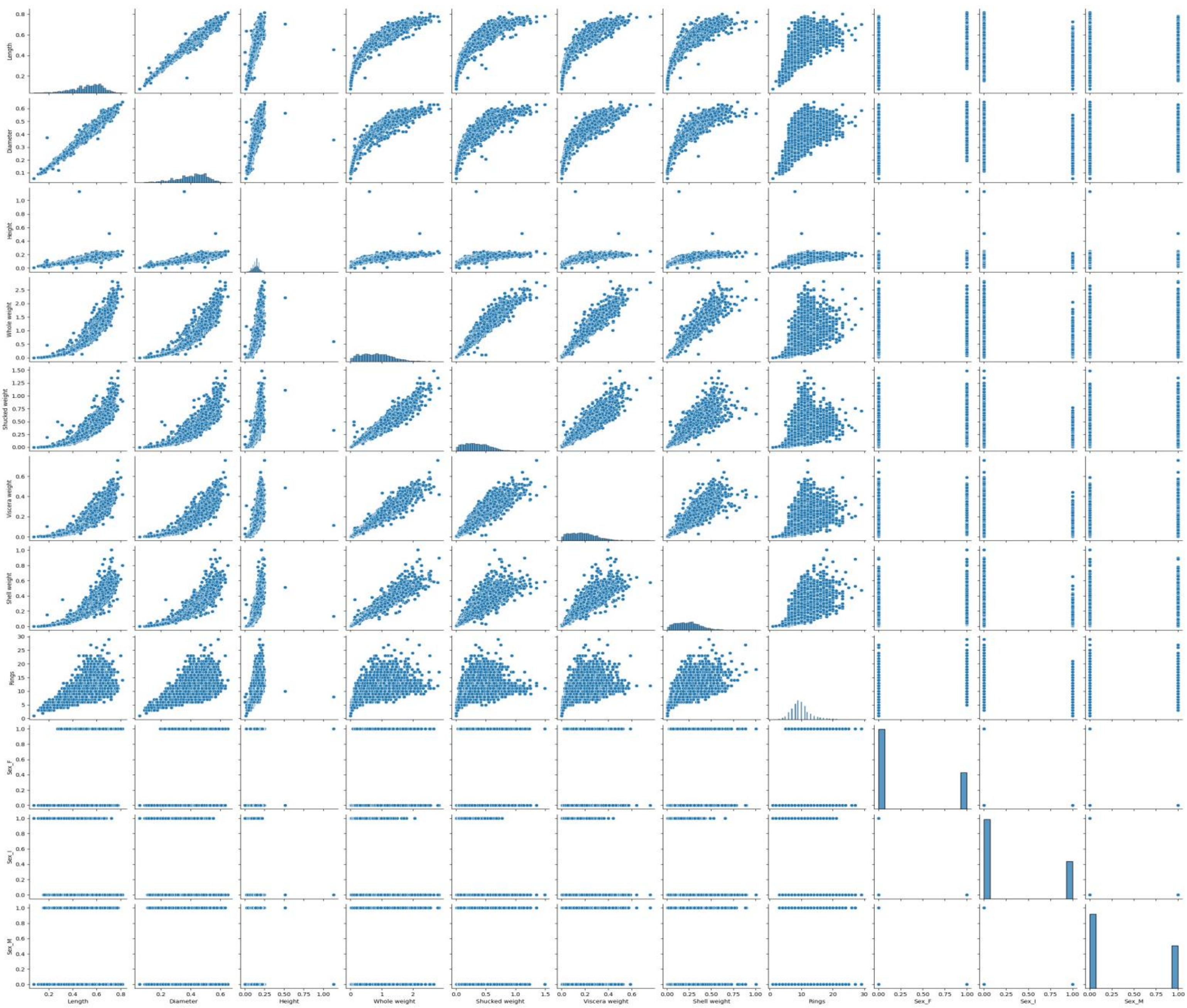
In [13]:sns.countplot(data)

Out[13]:<AxesSubplot: ylabel='count'>



In [13]:sns.pairplot(data)

Out[13]:<seaborn.axisgrid.PairGrid at 0x250074f0ee0>



4--Descriptive Statistics

In [14]:data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176 Data
columns (total 9 columns):
 #   Column   Non-Null Count Dtype
--- ------   -------------- -----
0   Sex   4177 non-null   object
1   Length   4177 non-null   float64
2   Diameter   4177 non-null   float64
3   Height   4177 non-null   float64
4   Whole weight   4177 non-null   float64
5   Shucked weight 4177 non-null   float64
6   Viscera weight 4177 non-null   float64
7   Shell weight   4177 non-null   float64 8   Rings   4177 non-null   int64 dtypes: float64(7), int64(1),
object(1) memory usage: 293.8+ KB

In [15]:data.describe()

Out[15]:

	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

5--Handling missing values

In [16]:data.isna().sum()

Out[16]:Sex   0
Length   0
Diameter   0
Height   0
Whole weight   0
Shucked weight   0
Viscera weight   0
Shell weight   0 Rings
0 dtype: int64

There is no missing values in this dataset

6--Handling Outliers

IQR

In [3]:quant=data.quantile(q=[0.75,0.25]) quant

Out[3]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0.75	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329	11.0
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130	8.0

In [4]:iqr=quant.loc[0.75]-quant.loc[0.25] iqr

Out[4]:Length   0.1650
Diameter   0.1300
Height   0.0500
Whole weight   0.7115
Shucked weight 0.3160

Viscera weight 0.1595
Shell weight 0.1990 Rings
3.0000 dtype: float64

UPPER BOUND

In [5]:upper=quant.loc[0.75]+(1.5*iqr) print(upper)

Length 0.86250
Diameter 0.67500
Height 0.24000
Whole weight 2.22025
Shucked weight 0.97600
Viscera weight 0.49225
Shell weight 0.62750 Rings
15.50000 dtype: float64

LOWER BOUND

In [6]:lower=quant.loc[0.25]-(1.5*iqr) lower

Out[6]:Length 0.20250
Diameter 0.15500
Height 0.04000
Whole weight -0.62575
Shucked weight -0.28800
Viscera weight -0.14575
Shell weight -0.16850 Rings
3.50000 dtype: float64

MEAN

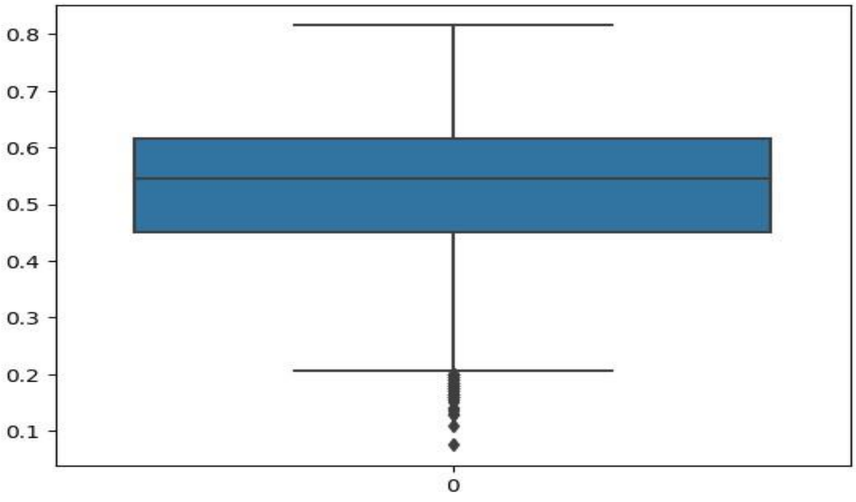
In [7]:data.mean()

Out[7]: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

Removing outlier

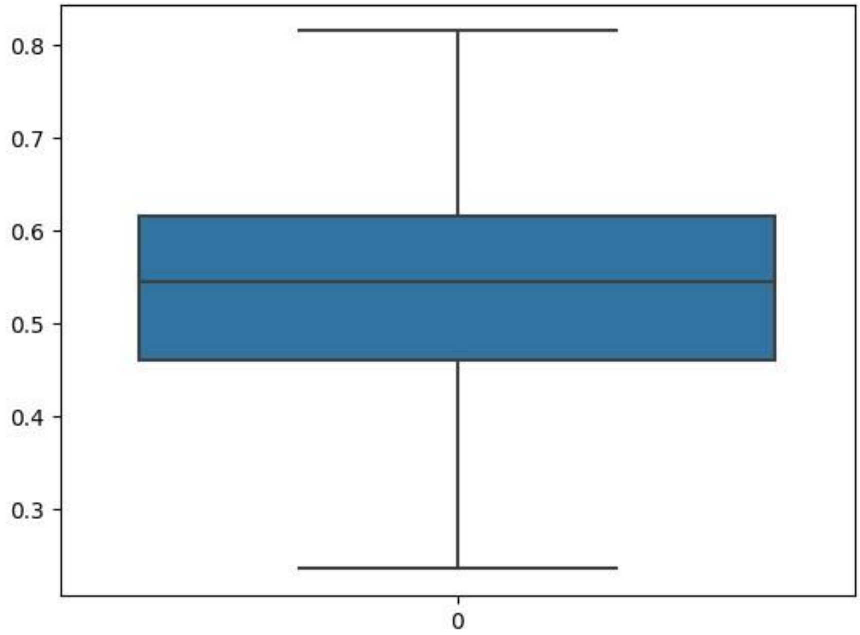
In [17]:sns.boxplot(data['Length'])

Out[17]:<AxesSubplot: >



In [18]:data['Length']=np.where(data['Length']< 0.23350,0.523992,data['Length']) sns.boxplot(data['Length'])

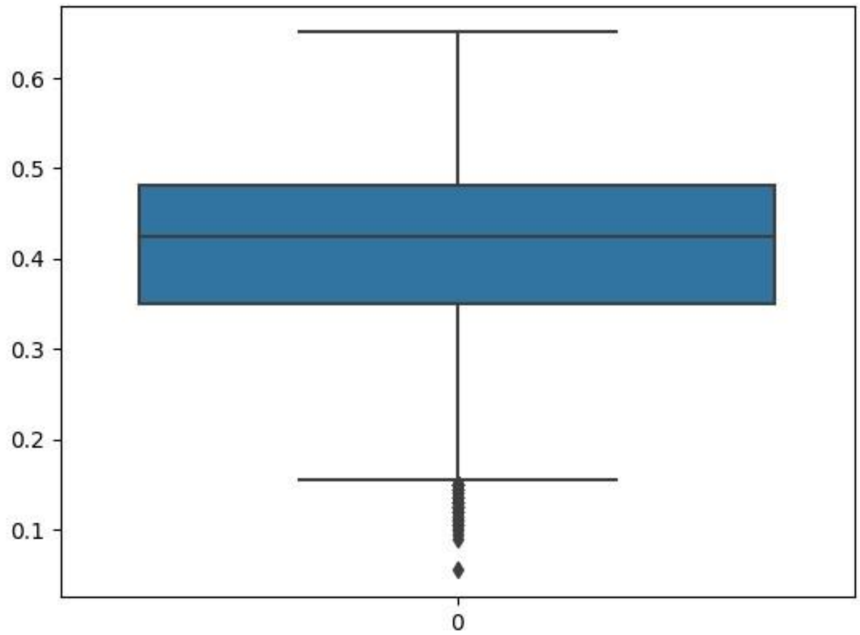
Out[18]:



Outlier removed for Length

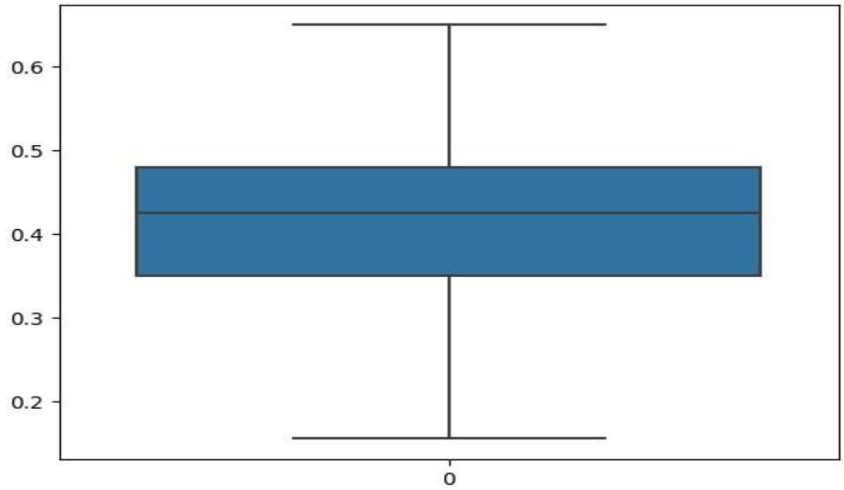
In [19]:sns.boxplot(data['Diameter'])

Out[19]:<AxesSubplot: >



In [20]:data['Diameter']=np.where(data['Diameter']<0.15500,0.407881,data['Diameter'])
sns.boxplot(data['Diameter'])

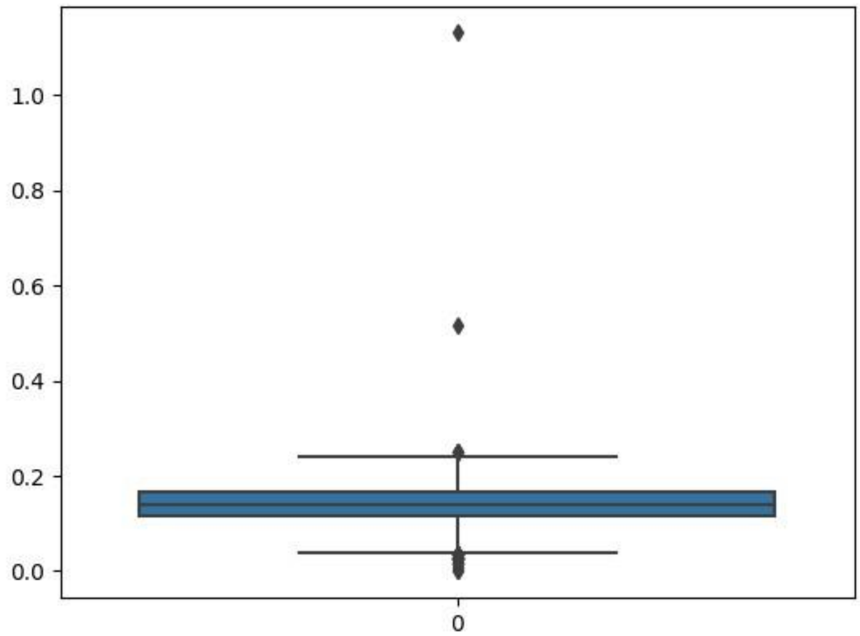
Out[20]:



Outlier removed for diameter

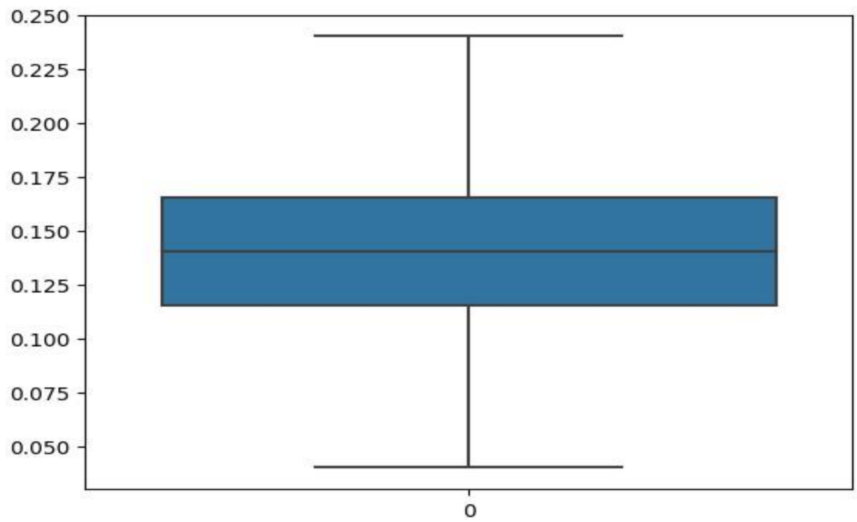
In [21]:sns.boxplot(data['Height'])

Out[21]:<AxesSubplot: >



In [22]:data['Height']=np.where(data['Height']< 0.04000,0.139516,data['Height'])
data['Height']=np.where(data['Height']> 0.24000,0.139516,data['Height']) sns.boxplot(data['Height'])

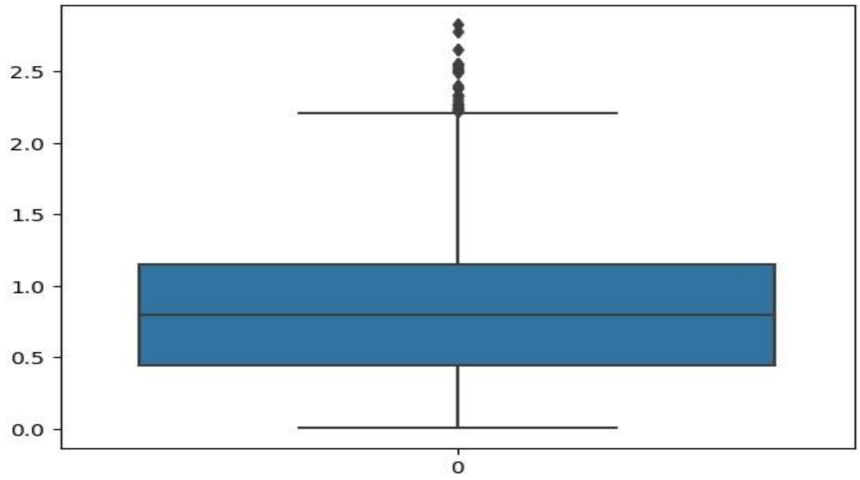
Out[22]:



Outlier removed for Height

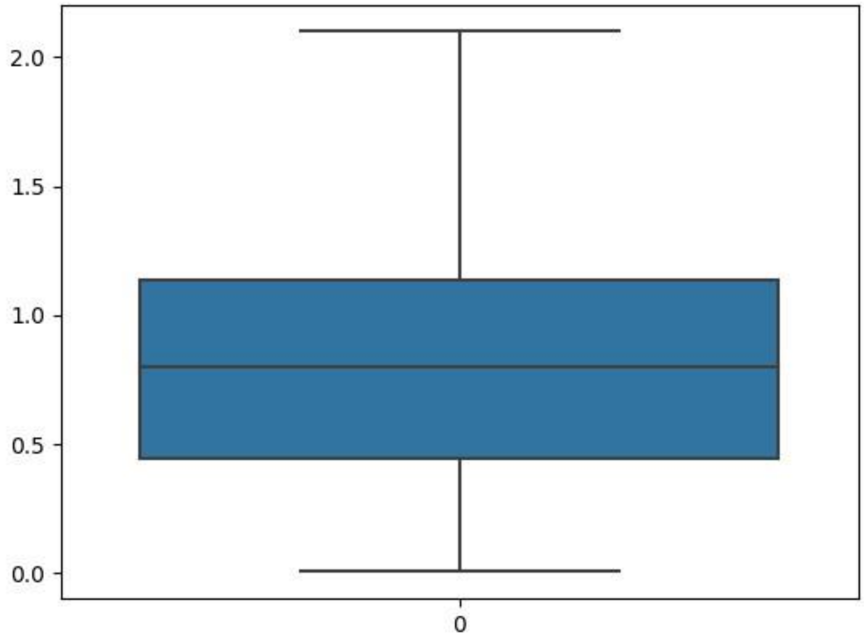
In [23]:sns.boxplot(data['Whole weight'])

Out[23]:<AxesSubplot: >



In [24]:data['Whole weight']=np.where(data['Whole weight']> 2.10022,0.828742,data['Whole weight'])
sns.boxplot(data['Whole weight'])

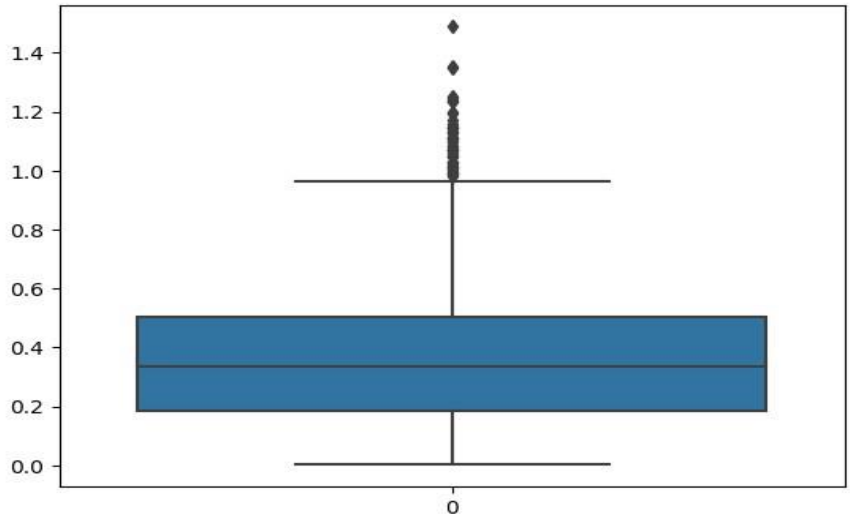
Out[24]:



In [25]:*# Outlier removed for Whole weightt*

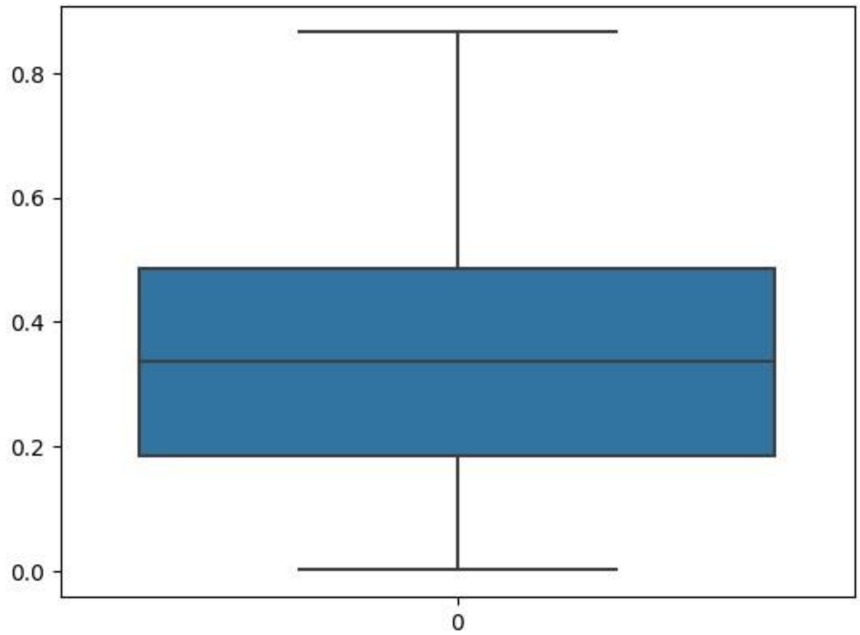
In [26]:sns.boxplot(data['Shucked weight'])

Out[26]:<AxesSubplot: >



In [27]:data['Shucked weight']=np.where(data['Shucked weight']> 0.86600,0.3593676,data['Shucked weight'])
sns.boxplot(data['Shucked weight'])

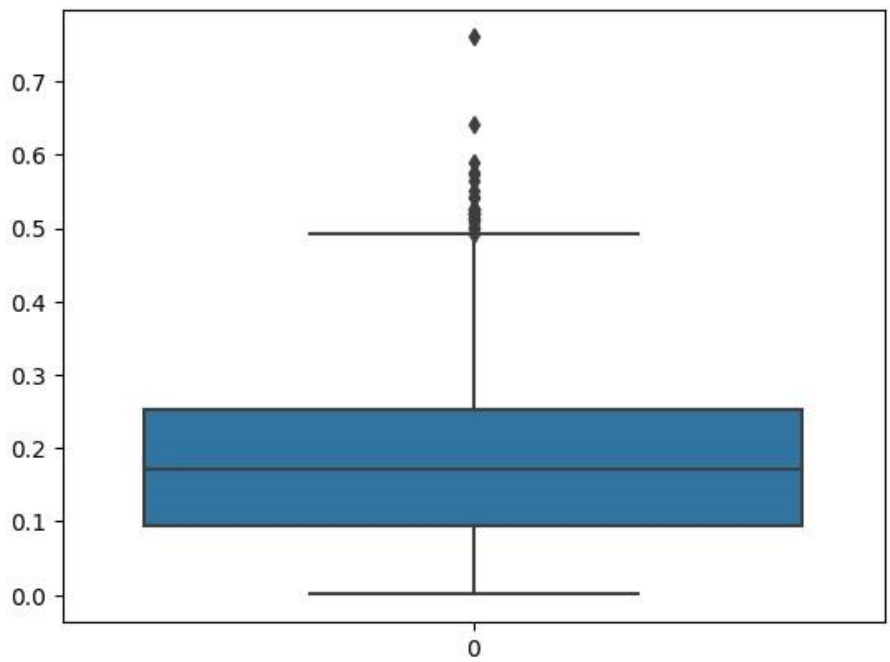
Out[27]:<AxesSubplot: >



In [28]:*# Outlier removed for Shucked weight*

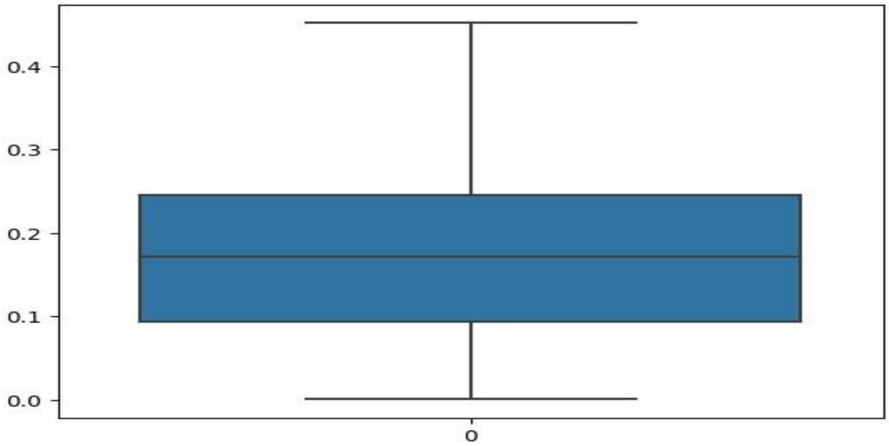
In [29]:sns.boxplot(data['Viscera weight'])

Out[29]:<AxesSubplot: >



In [30]:data['Viscera weight']=np.where(data['Viscera weight']> 0.45225,0.180594,data['Viscera weight'])
sns.boxplot(data['Viscera weight'])

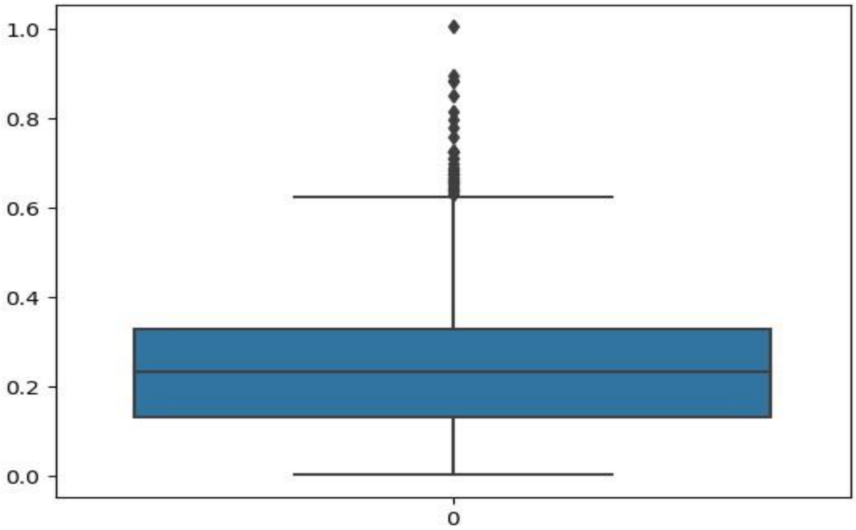
Out[30]:<AxesSubplot: >



In [31]:*# Outlier removed for Viscera weight*

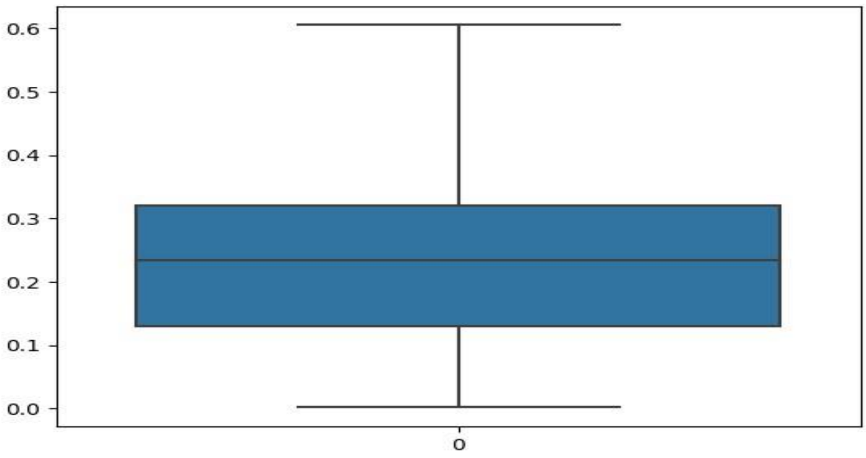
In [32]:sns.boxplot(data['Shell weight'])

Out[32]:<AxesSubplot: >



In [33]:data['Shell weight']=np.where(data['Shell weight']> 0.60750,0.238831,data['Shell weight'])
sns.boxplot(data['Shell weight'])

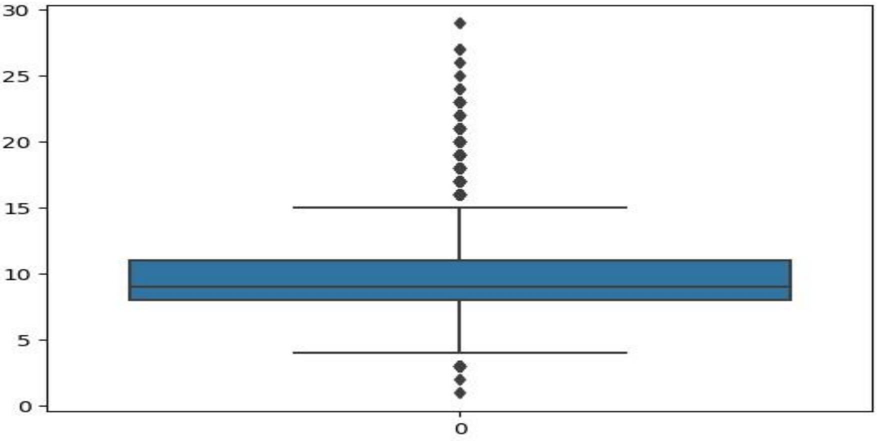
Out[33]:<AxesSubplot: >



In [34]:*# Outlier removed for Shell weight*

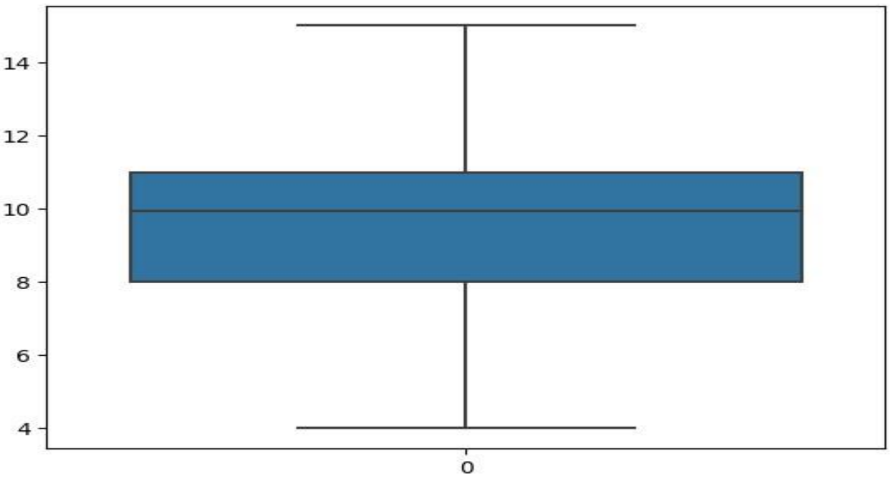
In [35]:sns.boxplot(data['Rings'])

Out[35]:<AxesSubplot: >



In [8]:data['Rings']=np.where(data['Rings']> 15.50000,9.933684,data['Rings'])
data['Rings']=np.where(data['Rings']< 3.50000,9.933684,data['Rings']) sns.boxplot(data['Rings'])

Out[8]:<AxesSubplot: >



In [37]:*# Outlier removed for Ring*

7--Encoding

In [3]:data=pd.get_dummies(data,columns=['Sex'])

In [10]:data.head()

Out[10]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Sex_F	Sex_I	Sex_M
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15.0	0	0	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7.0	0	0	1
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9.0	1	0	0
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10.0	0	0	1
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7.0	0	1	0

8--Splitting dependant and independant variables

In [4]:x=data.drop(columns=['Rings']).values
y=data.Rings.values x

Out[4]:array([[0.455, 0.365, 0.095, ..., 0. , 0. , 1.],
[0.35 , 0.265, 0.09 , ..., 0. , 0. , 1.], [0.53
, 0.42 , 0.135, ..., 1. , 0. , 0.],
...,
[0.6 , 0.475, 0.205, ..., 0. , 0. , 1.],
[0.625, 0.485, 0.15 , ..., 1. , 0. , 0.],
[0.71 , 0.555, 0.195, ..., 0. , 0. , 1.]])

In [12]:y

Out[12]:array([15., 7., 9., ..., 9., 10., 12.])

9--Scaling independant variables

In [13]:**from** sklearn.preprocessing **import** scale
```
x=scale(x) x
```
```
Out[13]:array([[-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]])
```

10--splitting the data into training and testing

In [5]:**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=0)
```
In [44]:x_train**.**shape

Out[44]:(3341, 10)

In [45]:x_test**.**shape

Out[45]:(836, 10)

In [46]:x**.**shape

Out[46]:(4177, 10)

In [47]:y_train**.**shape

Out[47]:(3341,)

In [48]:y_test**.**shape

Out[48]:(836,)

In [49]:y_train**.**shape

Out[49]:(3341,)

11-Building the model

In [6]:**from** sklearn.linear_model **import** LinearRegression reg**=**LinearRegression()
```
reg.fit(x_train,y_train)
```
Out[6]:

LinearRegression

LinearRegression()

12--Training the model

In [7]:*#prediction on training data*
```
pred=reg.predict(x_train) pred
```
```
Out[7]:array([ 5.32778396,  5.34787166, 13.87574587, ...,  9.71432748,
       11.7565958 ,  8.03917066])
```
In [8]:*#prediction on testing data*
```
y_pred=reg.predict(x_test) y_pred
```
```
Out[8]:array([13.17014483,  9.10922796, 10.35985364,  5.54120008, 10.65916033,
       12.05100358,  8.26779233, 10.09973617,  8.07089377, 12.28730504,
        8.22228608,  6.26339268,  8.4766956 ,  9.30261309,  5.70767624,
        9.45939464,  8.32687786, 13.85702971, 11.09090306,  7.77985409,
        7.44406172,  6.94780239,  8.88503259,  7.9654523 ,  9.51969794,
       11.56414001,  5.59707836, 13.11441854, 10.10204559, 11.59519979,
        8.45138755,  4.66462354, 11.30148049, 12.76604688,  7.15709165,
        8.63988457,  8.5200469 , 10.25020425,  8.16602336, 11.70289969,
       11.60141065,  9.56418323, 12.04986801, 11.89787512, 12.49372844,
        9.76276335,  9.2051602 , 11.83360392, 11.42857215,  7.90264847,
```

11.93961315, 7.13983251, 9.40292052, 13.71258569, 9.35573394,
8.09482073, 6.88614845, 7.8807578 , 7.3433024 , 7.10793707,
9.7911508 , 9.14297012, 10.43650314, 7.60734099, 8.01110075,
12.3951208 , 12.398778 , 12.34386704, 8.53070865, 14.31136822,
9.164063 , 18.80234388, 10.85486878, 10.62353632, 10.09018517,
8.90345615, 10.26383911, 10.1508343 , 11.30742361, 8.5655046 ,
9.62245825, 6.12382531, 7.17042395, 12.21957718, 9.45685753,
8.02722065, 10.11019131, 12.4580311 , 5.03646764, 7.00715717,
9.71770007, 10.62680397, 8.37143 , 2.50393825, 11.98492521,
6.25260615, 10.0485812 , 8.14000991, 14.19423453, 9.54563988,
10.01505727, 11.24031021, 10.00665829, 11.15604241, 5.65409689,
10.3605766 , 7.32736447, 6.994029 , 7.51308546, 13.5993784 ,
9.06757741, 11.00923226, 11.24404651, 8.60387213, 14.38612508,
10.2577704 , 11.7119004 , 13.23129132, 5.21937368, 9.28352027,
7.20128047, 11.72547647, 6.87725702, 9.13839505, 11.03963916,
13.27387874, 10.61175995, 10.79014221, 7.7722332 , 10.716316 ,
10.45506787, 8.41743889, 9.84185901, 11.2842196 , 11.41959736,
11.0808743 , 10.49707272, 8.82320871, 7.20728334, 13.88165762,
10.87036059, 11.54690512, 7.16806741, 8.53621066, 11.51270352,
10.27343561, 9.12768361, 7.27469893, 8.0704399 , 6.65863828,
8.7182523 , 17.15062041, 7.1208007 , 11.09105929, 8.21477651,
6.68016316, 10.76651969, 7.03804216, 13.17375097, 8.47241615,
10.41950876, 7.86988347, 9.82248106, 10.80970665, 5.38752726,
12.27147414, 7.11609533, 6.50365269, 11.30095645, 9.35235169,
8.30256607, 6.16510119, 8.75977964, 10.50787414, 12.77043643,
10.38850716, 5.83610865, 8.24339804, 8.56011322, 12.12312429,
9.18028358, 10.64936179, 8.75963674, 8.67743839, 11.66214092,
10.79306807, 8.90205159, 8.36978049, 6.74152012, 9.14536818,
10.82026096, 10.11732009, 9.18956686, 10.47231965, 10.4259198 ,
9.48658012, 7.81386013, 8.99043021, 5.16333508, 15.44583586,
9.76944005, 11.16296023, 14.46118322, 9.92808161, 9.44561919,
12.53927229, 7.07888171, 12.39701983, 9.56904244, 12.27378782,
5.78989131, 9.83047807, 10.9974863 , 10.12277916, 11.67297645,
10.18254011, 9.92466066, 11.13740707, 5.74691472, 12.50820342,
11.87344753, 11.78457642, 8.05833466, 9.7806431 , 11.59998052,
14.15436655, 11.1711593 , 9.81575233, 12.00624932, 8.21180642,
9.63078885, 8.87790108, 6.1383025 , 11.44202708, 7.46966705,
12.27290928, 7.97320661, 5.64564762, 7.96442446, 10.41666077,
9.34143048, 10.14108468, 11.90000571, 9.8923703 , 10.04004317,
9.51553187, 11.43394113, 7.90477517, 11.06689539, 14.62927303,
12.1791369 , 8.37204219, 12.58063474, 8.08039535, 10.67836046,
11.40246175, 7.34468299, 9.15317562, 9.93415493, 15.3163193 ,
10.56095213, 9.6325556 , 10.8650268 , 7.79141397, 10.31722098,
8.27193829, 9.4763941 , 6.91123369, 7.91185634, 7.87308293,
11.343783 , 9.26138239, 8.82288098, 8.9330123 , 6.4174903 ,
10.83769016, 13.53828814, 9.14344418, 10.21206111, 9.94911202,
7.04427649, 11.23582732, 7.44996267, 9.72693198, 10.24505415,
12.06835237, 6.91645284, 10.90442016, 10.579799 , 13.12895141,
7.69024935, 8.56109819, 6.17597945, 9.61165531, 7.21035733,
8.04762806, 6.78811105, 6.16074879, 9.78424022, 15.46518548,
9.22025194, 8.87815086, 12.65505125, 9.50789119, 9.30721901,
8.65302689, 9.91356157, 10.3483697 , 11.55645638, 9.44063129,
9.21575415, 10.85316777, 8.0415159 , 10.22884147, 17.13427908,
11.84745077, 8.76164804, 6.72407822, 8.47621399, 10.6602947 ,
11.22188929, 10.20508595, 10.13124066, 9.82883865, 12.29073279,
9.91332839, 8.99631725, 9.62988741, 15.3886821 , 9.79276556,
8.36742244, 11.40193289, 5.21439082, 10.62175021, 9.95195519,
10.89018456, 7.47871677, 9.66952186, 11.31920416, 10.66604485,
9.4082212 , 10.30497792, 10.63949851, 11.60890333, 12.38327886,
6.91990564, 6.15057396, 12.59320021, 13.17764304, 7.42708886,
6.3545605 , 8.41353413, 12.32717561, 10.03760242, 10.96100113,
10.09455307, 10.87676124, 8.6667971 , 8.06636985, 10.54783533,
9.14978821, 10.27113242, 7.81724471, 9.12798102, 8.83487398,
6.80139517, 6.4161699 , 9.35595493, 8.3157471 , 10.4598419 ,
10.64561231, 7.37349644, 11.68881568, 7.54718552, 12.63345119,
13.28423389, 13.09668922, 8.05848399, 11.08459918, 5.03698129,
9.27399861, 10.65312838, 9.97673686, 12.7343062 , 7.14852342,

11.05285878, 15.53227364, 9.46472551, 7.64100654, 6.53251258,
14.93051178, 10.36089623, 11.51222023, 10.08033707, 18.00816097,
7.43595027, 10.99184888, 9.15181407, 11.31938908, 10.64308615,
8.51706808, 8.77997645, 6.5746879 , 13.25302798, 9.15291019,
10.11517066, 7.12147978, 11.60947652, 15.73141063, 10.21928393,
10.26415852, 7.38043073, 10.27642275, 10.53869559, 13.76713077,
10.69689048, 12.8206553 , 11.77290059, 14.56722515, 10.02074911,
9.46302853, 8.93879607, 10.89993525, 11.59264571, 6.14039636,
8.98729731, 10.79404257, 11.68120274, 11.62295474, 13.87910146,
9.32338195, 10.47763045, 9.22667014, 13.13476922, 8.69426892,
9.83463083, 11.95025184, 10.10749788, 10.00445195, 10.61017126,
8.14285584, 6.88616414, 13.38955712, 8.15132996, 6.69389695,
7.26484177, 13.49684563, 5.70653872, 10.79720829, 10.87691419,
5.63351722, 7.00868772, 9.80953478, 6.92167271, -2.95486979,
9.00020272, 11.57445581, 9.88473056, 9.39150796, 12.10765656,
7.10957348, 15.3801368 , 8.70961218, 10.74956657, 9.86327737,
12.33146667, 7.64333524, 19.78140958, 10.59503125, 9.25921662,
8.11377995, 10.83605757, 7.77621921, 14.8909173 , 8.53939048,
9.53669585, 7.9281739 , 7.87690416, 8.17984939, 11.02521423,
10.70090972, 11.95313521, 9.29647895, 11.96229214, 13.56693829,
12.12509661, 10.28005974, 10.24750158, 7.83955625, 10.90534537,
5.00924596, 9.53146933, 12.99768913, 11.02158266, 7.00832919,
7.48338897, 8.4380444 , 6.44530822, 10.45082447, 9.72212336,
7.9456168 , 12.64352459, 6.99834135, 12.12782633, 10.75800038,
10.67537673, 11.57343334, 14.28242229, 7.73261079, 5.99940295,
10.15228159, 9.17818542, 14.41629654, 14.48409168, 9.85923459,
8.14030564, 10.76332056, 9.93303692, 13.99555973, 10.59100334,
9.89811805, 10.26385635, 10.98800278, 10.03749712, 7.23285132,
12.35297969, 9.43113086, 10.15037024, 8.33824004, 8.37110858,
8.77101405, 7.37992479, 9.03564707, 10.3857669 , 5.40363239,
9.90869428, 12.30630856, 12.15401425, 8.04522886, 13.51533782,
7.72499911, 10.20752978, 8.11626291, 12.66000991, 7.78850711,
10.81088463, 7.66310022, 9.03306044, 10.6929132 , 10.02764463,
11.3392804 , 13.6111223 , 7.86586402, 13.34870745, 11.16080787,
7.65058919, 8.831394 , 9.78640297, 10.45801218, 10.37531118,
12.1295994 , 9.60585313, 11.92176647, 9.32447596, 10.76371391,
6.78669655, 8.90700543, 7.58247733, 10.60973707, 7.81917943,
11.27394006, 8.83409484, 8.08463747, 9.77880395, 8.62895541,
7.86940516, 11.36876388, 10.14797018, 11.23679848, 4.93643115,
13.33465099, 6.28777892, 5.83423924, 6.39045317, 8.84940625,
9.34597466, 20.5275568 , 10.76395549, 8.6606327 , 12.49698476,
6.00841222, 9.29698924, 10.43921745, 7.68221312, 11.50082812,
10.3825819 , 4.65163958, 8.16469213, 6.57772975, 10.12708023,
8.40201877, 13.04169834, 10.55243517, 9.31239983, 11.99547908,
10.22571064, 10.00485939, 7.75168597, 8.95661875, 11.83692745,
7.28180802, 13.14387783, 10.34036308, 8.67880792, 11.21691482,
13.56067497, 8.58752807, 7.65527671, 14.02321063, 10.81180581,
10.18796925, 12.07443076, 7.22794319, 10.13912855, 11.27161945,
11.93724129, 8.48668725, 12.78449546, 9.51011904, 10.80452408,
10.47371776, 7.22054747, 8.80801932, 8.5323248 , 8.82922965,
9.54156721, 5.46622121, 14.5612365 , 10.05646653, 10.14340123,
10.38192662, 10.01105774, 11.40187832, 9.9018419 , 10.03864136,
10.06123171, 7.46164285, 9.21049206, 11.95434707, 10.61356678,
8.47610545, 8.35462937, 9.11398832, 9.8449566 , 6.98309669,
12.74468301, 7.47401314, 8.09057119, 10.79311758, 13.60804213,
11.69767985, 7.66159854, 13.53733476, 11.60222588, 12.24649105,
8.35703988, 8.69429908, 6.00880527, 6.67609329, 11.52967319,
9.76582811, 11.44363028, 15.45123941, 12.09178189, 5.75070956,
7.48752584, 9.2516493 , 6.19773885, 7.59862879, 15.40867958,
13.44355497, 9.73975357, 10.63151344, 10.40306162, 9.65897878,
7.41210479, 11.70578143, 11.34338122, 10.23576643, 16.8112554 ,
9.43658 , 11.46697648, 11.88026147, 13.9906949 , 7.38250752,
10.31187707, 15.13318446, 13.04857472, 8.36924114, 8.69171024,
6.93579773, 11.05364791, 9.49013012, 13.29726777, 9.63611855,
11.70290951, 10.19282933, 8.10692402, 12.89537993, 11.30785064,
6.3257887 , 5.28555582, 6.04752728, 11.20700827, 7.20511778,
9.62401869, 11.39500126, 8.38036493, 7.31755538, 8.29492917,

```
13.2538857 ,  5.30453205,  9.71222001,  6.91730128, 13.46024912,
 5.98907129,  8.48648297,  8.49513495, 12.17664325, 11.17305268,
 8.56706169, 11.37085312, 11.97599514, 12.64655226,  8.55949895,
10.65291861,  8.03712261,  8.14192944,  8.71027982,  6.26470647,
12.45630579,  8.99928223,  7.03926853,  6.74544361, 10.64830056,
 7.27835623,  9.53179166,  8.7812475 ,  8.58862035,  9.61156007,
 9.45580203,  9.25164308,  8.35263242, 11.01554311,  8.78284278,
11.94990673, 10.13880754,  7.2306203 ,  9.21757781, 14.55390007,
 6.87475459,  6.68663408,  7.57098621,  6.13663789, 11.16482928,
 8.91745923, 10.47144936,  7.01059119, 11.85817086,  8.49112774,
 7.07943473,  9.69011526, 13.70522141,  9.38973206,  9.15129583,
10.09901384,  7.21604273, 13.44693909,  7.40404536, 10.36139035,
14.7101407 , 12.53138476,  7.51662026,  7.92928767,  8.23364917,
 9.22134658, 10.0367933 ,  8.32253057,  9.89085283, 11.11840966,
10.0080677 , 10.73235242, 13.85655801, 10.61114551, 13.20763459,
 9.08040636, 10.93792739, 10.25376778,  4.78164155, 11.22029536,
10.44484521,  7.94961688,  9.15100996,  6.26065873,  6.76090596,
 9.03804389,  9.64078406, 11.11226901,  7.63075893,  9.02905811,
 8.69196297,  7.93082363, 10.07981061, 12.47395894, 10.74586699,
 9.93373911,  9.70992672, 12.01120954, 12.69429573,  8.98929011,
 6.05385557, 10.28822631, 11.13840504,  6.72911336,  7.90067854,
 9.08107423, 11.97300616, 10.32523818,  6.62532275,  6.84965725,
 7.96279994, 10.7008404 ,  9.10178821,  8.13580849, 12.18420246,
 5.69113674])
```

13--Testing the model

In [9]:reg.predict([[0.530,0.139516,0.135,0.6770,0.2565,0.1415,0.210,1,0,0]])

Out[9]:array([7.93478621])

14--Performance using metrices

In [10]:**from** sklearn.metrics **import** r2_score print(f'Training accuracy:{(r2_score(y_train,pred))*100:.2f}%') print(f'Testing accuracy:{(r2_score(y_test,y_pred))*100:.2f}%')

Training accuracy:53.70%
Testing accuracy:53.90%