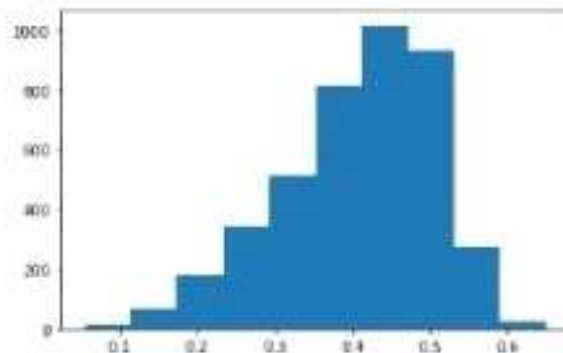


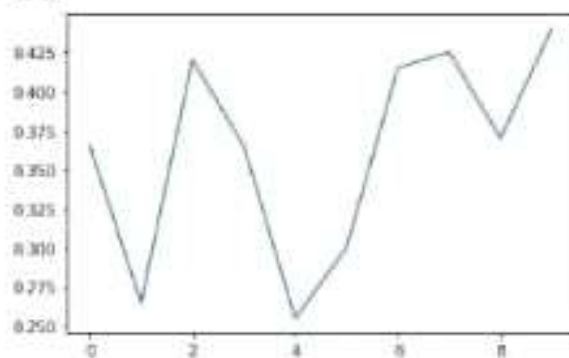
Assignment date	07 November 2022
Student Name	Ms.K.Gayathri
Student Roll Number	821719106011
Maximum Marks	2 Marks

```
Out[ ]: (array([ 13.,  66., 180., 344.,
        812., 1017.,  934., 275.,
         23.]),
        array([0.055 , 0.1145, 0.174 , 0.233
        93 , 0.3525, 0.412 , 0.4715,
         0.531 , 0.5905, 0.65  ]),
        )
```



```
In [ ]: plt.plot(data['Diameter'].head(10))
```

```
Out[ ]: []
```

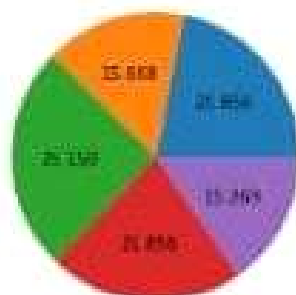


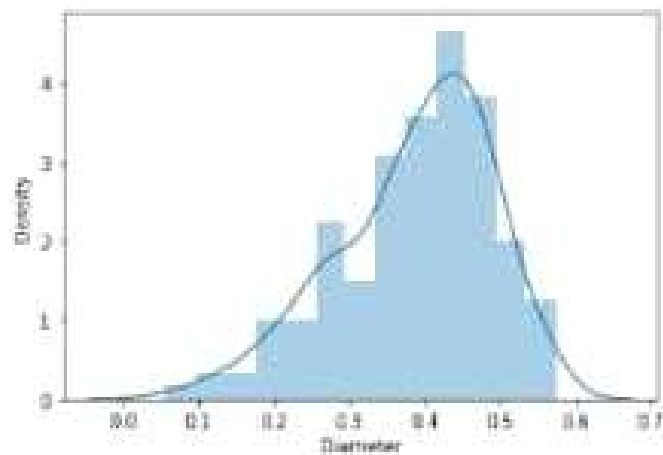
```
In [ ]: plt.pie(data['Diameter'].head(), autopct='%0.3f')
```

```

(1,
 ,
 ,
 ,
 ],
 [Text(0.8507215626110557, 0.6973
76, ''),
 Text(-0.32611344931648134, 1.05
1026, ''),
 Text(-1.0998053664078908, -0.02
47144, ''),
 Text(-0.08269436219656089, -1.0
0709, ''),
 Text(0.9758446362287218, -0.507
241, '')] ,
 [Text(0.46402994324239394, 0.380
369, '21.856'),
 Text(-0.17788006326353525, 0.57
2377, '15.868'),
 Text(-0.5998938362224858, -0.01
984419, '25.150'),
 Text(-0.045106015743578656, -0.
712958, '21.856'),
 Text(0.5322788924883937, -0.276
768, '15.269')])

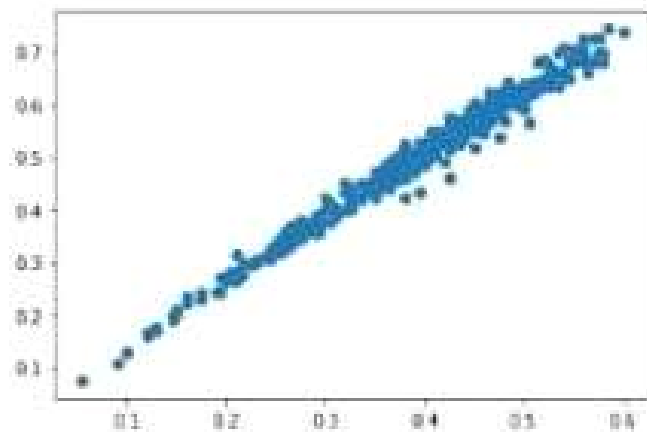
```





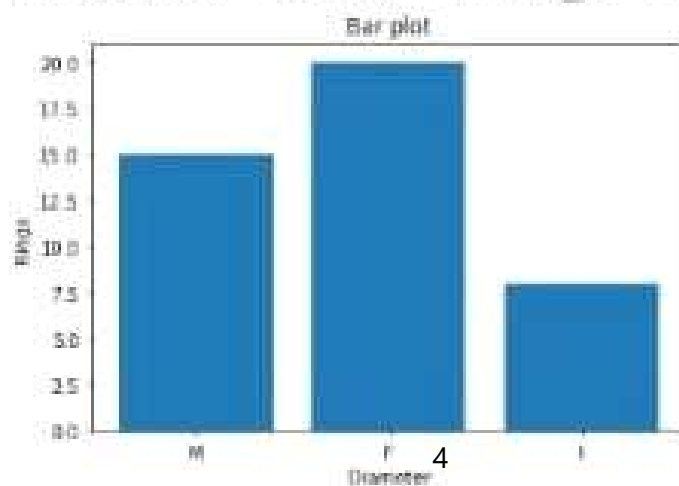
```
In [1]: plt.scatter(data['Diameter'].head(400),data['Length'].head(400))
```

Out[1]:



```
In [1]: plt.bar(data['Sex'].head(20),data['Rings'].head(20))
plt.title('Bar plot')
plt.xlabel('Diameter')
plt.ylabel('Rings')
```

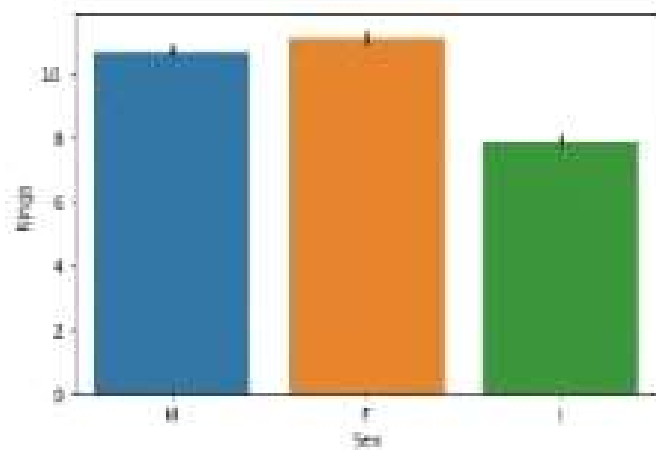
Out[1]: Text(0, 0.5, 'Rings')



```
In [1]: sns.barplot(data['Sex'], data['Rings'])
```

Out[1]:

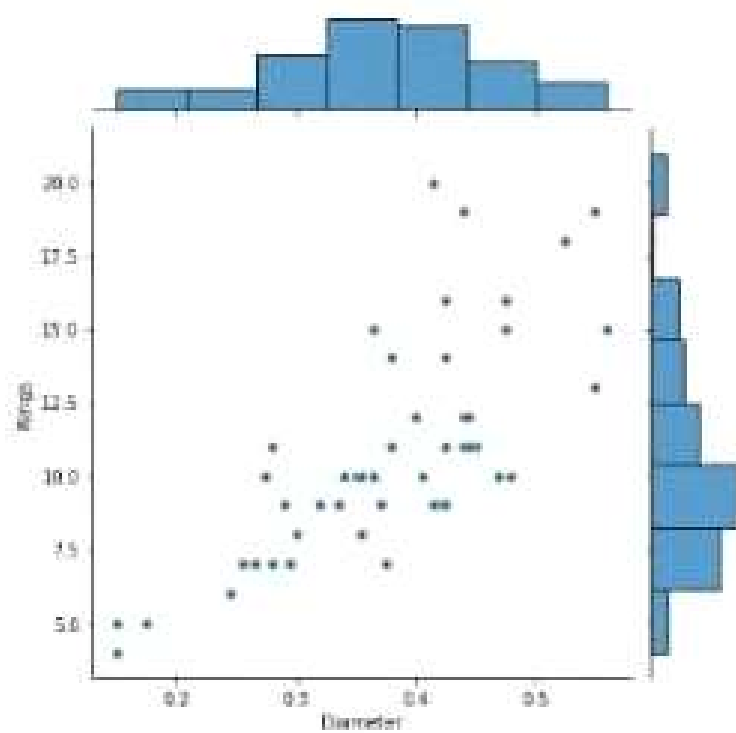
(Out):



(In):

```
sns.jointplot(data['Diameter'],head(50),data['Rings'],head(100))
```

(Out):

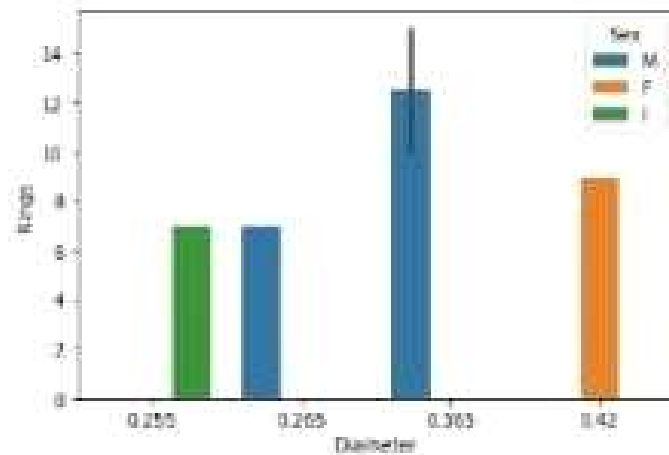


(In):

```
sns.barplot('Diameter','Rings',hue='Sex',data=data.head())
```

(Out):

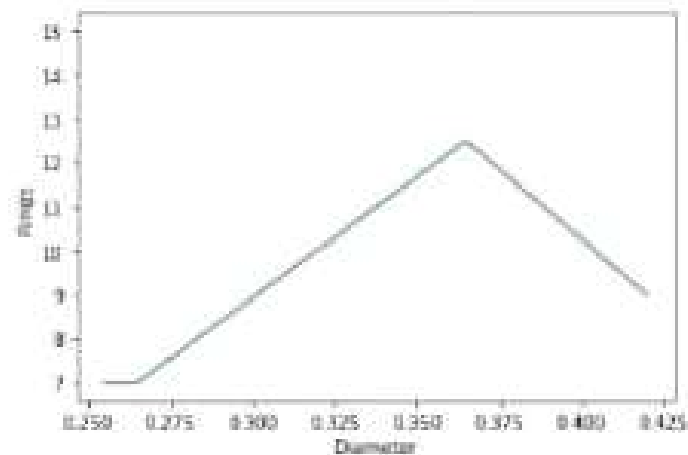
```
In [1]:
```



```
In [2]:
```

```
sns.lineplot(data['Diameter'].head(), data['Rings'].head())
```

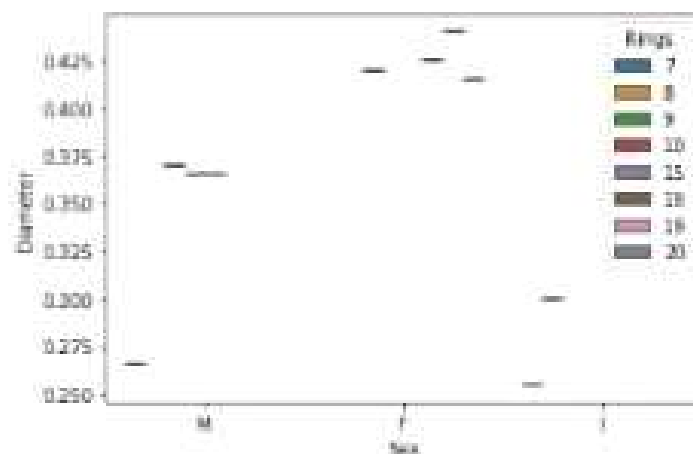
```
Out[2]:
```



```
In [3]:
```

```
sns.boxplot(data['Sex'].head(10), data['Diameter'].head(10), data['Rings'].head(10))
```

```
Out[3]:
```



```
In [4]:
```

```
fig=plt.figure(figsize=(8,5))  
sns.heatmap(data.head(),corr(),annot=True)
```



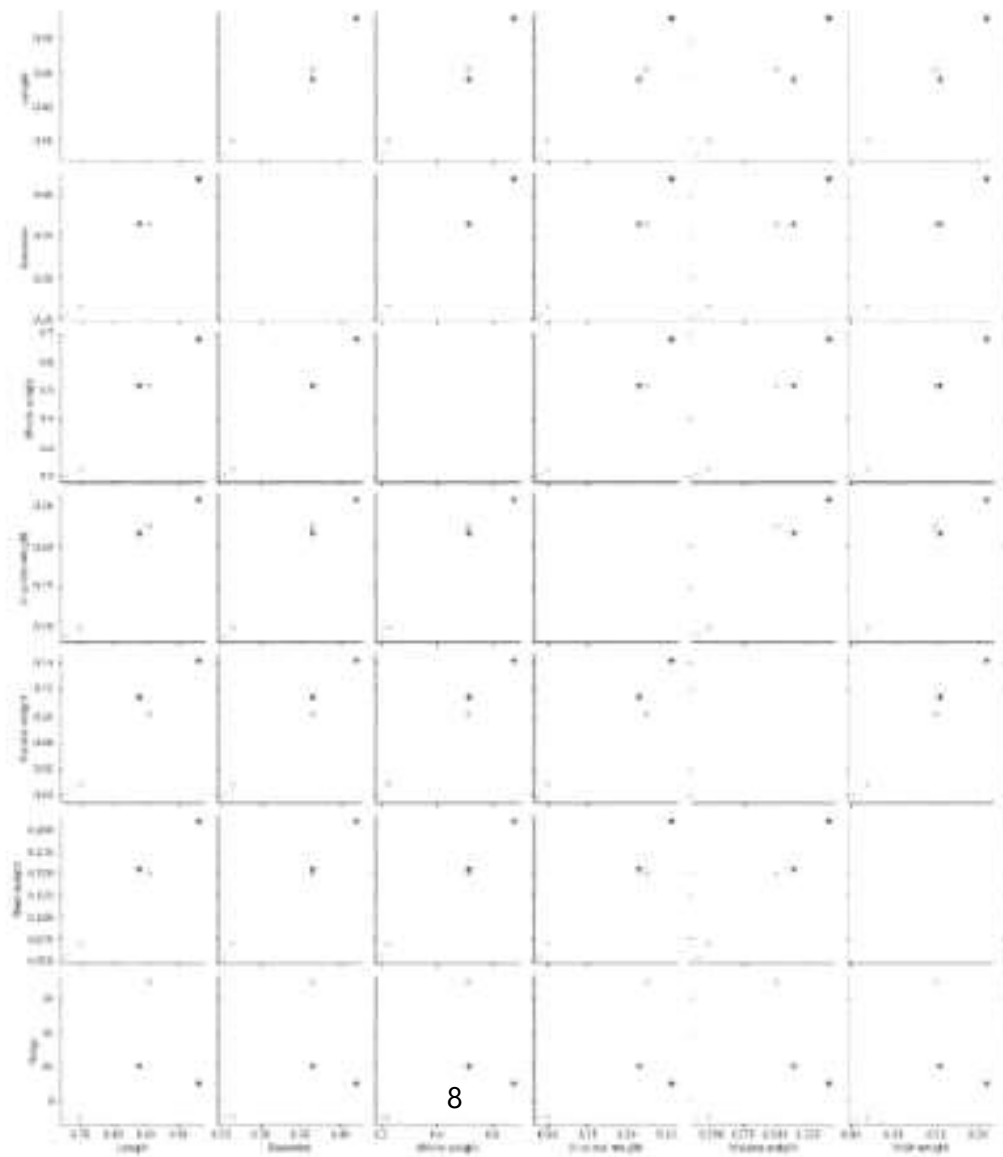
```
sns.heatmap
```



```
In [1]:
```

```
sns.pairplot(data.head(), hue='Height')
```

```
Out[1]:
```

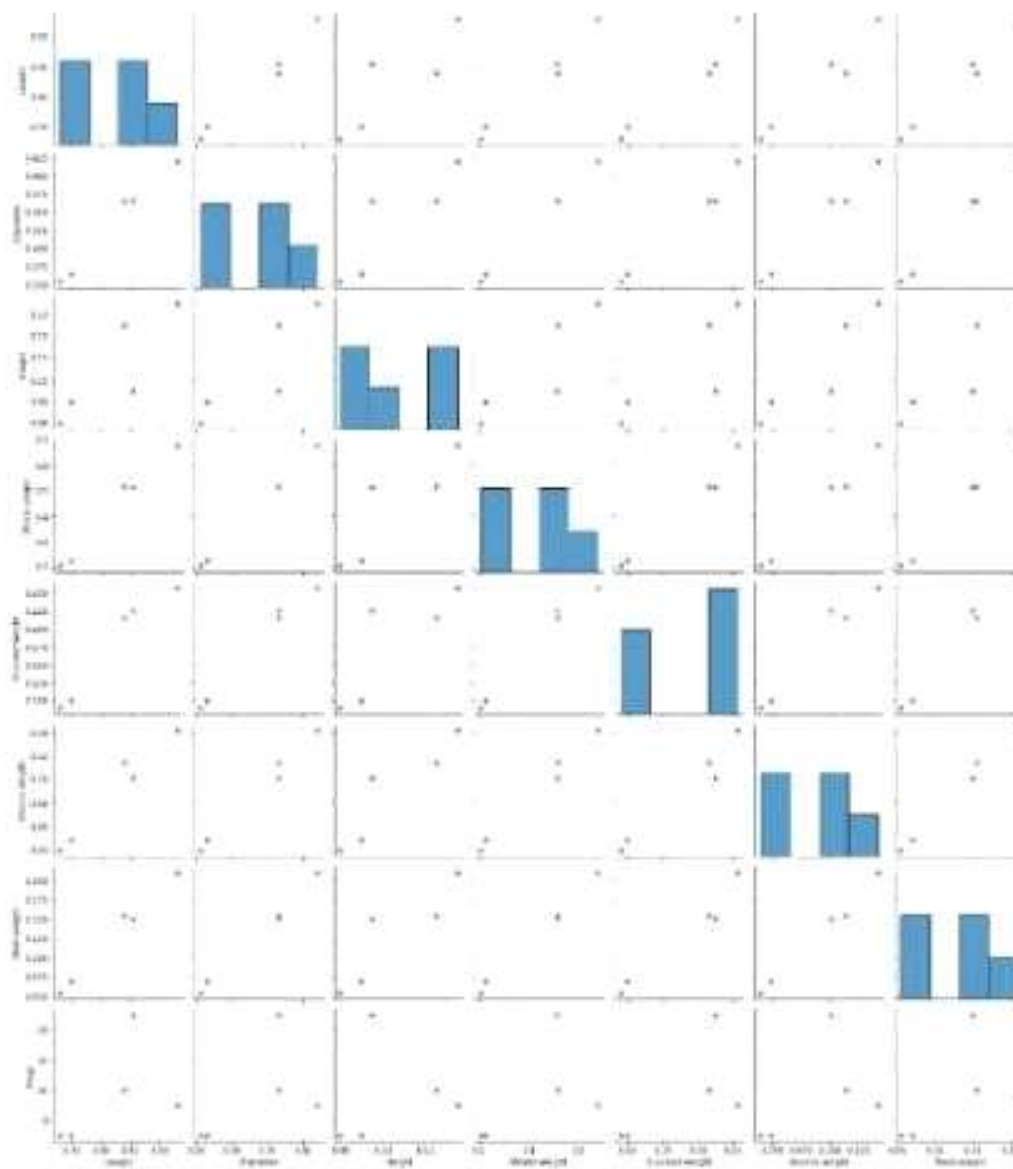


```
In [1]:
```

```
sns.pairplot(data.head())
```

```
Out[1]:
```


Out[]:



3 Perform Descriptive Statistics on the dataset

In []:

```
data.head()
```

Out[]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.435	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	F	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In []:

```
data.tail()
```

Out[]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
4172	F	0.565	0.450	0.185	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.480	0.135	0.9680	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	1.0540	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.6485	0.9455	0.3765	0.4950	12

In []:

```
data.info()
```

RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):

#	Column	Non-Null Count
0	Sex	4177 non-null
1	Length	4177 non-null
4		
2	Diameter	4177 non-null
4		
3	Height	4177 non-null
4		
4	Whole weight	4177 non-null
4		
5	Shucked weight	4177 non-null
4		
6	Viscera weight	4177 non-null
4		
7	Shell weight	4177 non-null
4		
8	Rings	4177 non-null

dtypes: float64(7), int64(1), objec
memory usage: 293.8+ KB

```
In [1]: data.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera w
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.00
mean	0.523992	0.407881	0.139316	0.828742	0.359367	0.16
std	0.120893	0.099240	0.041827	0.490389	0.221963	0.10
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.00
25%	0.450000	0.350000	0.115000	0.441500	0.166000	0.09
50%	0.545000	0.425000	0.140000	0.789500	0.306000	0.17
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.25
max	0.815000	0.650000	1.130000	2.825500	1.428000	0.76

```
In [2]: data.mode().T
```

	0	1
Sex	M	NaN
Length	0.55	0.625
Diameter	0.45	NaN
Height	0.15	NaN
Whole weight	0.2225	NaN
Shucked weight	0.175	NaN
Viscera weight	0.1715	NaN
Shell weight	0.275	NaN
Rings	9.0	NaN

```
In [3]: data.shape
```

```
Out[3]: (4177, 9)
```

```
In [4]: data.kurt()
```

```
Out[4]: 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
```



```
In [1]: data.skew()
```

```
Out[1]: 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
```

```
In [2]: data.var()
```

```
Out[2]: 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
```

```
In [3]: data.nunique()
```

```
Out[3]: Sex              3
Length             134
Diameter           111
Height              51
Whole weight       2429
Shucked weight     1515
Viscera weight     880
Shell weight       926
Rings              28
dtype: int64
```


4. Check for missing values and deal with them

```
In [ ]: data.isna()
```

```
Out[ ]:
```

	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

```
In [ ]: data.isna().any()
```

```
Out[ ]:
```

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

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

```
Out[ ]:
```

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

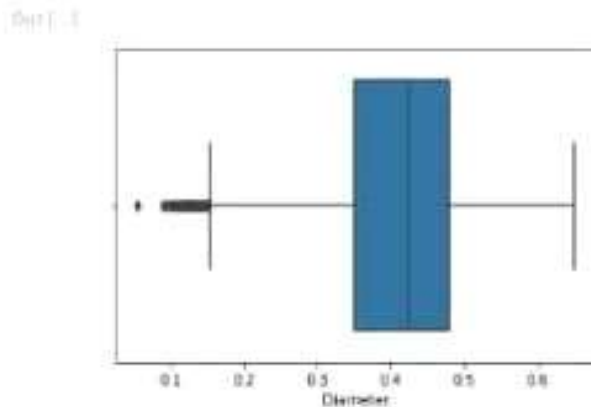
16

```
In [ ]: data.isna().any().sum()
```

```
Out[ ]: 0
```


5 Find the outliers and replace them outliers

```
In [ ]: sns.boxplot(data["Diameter"])
```



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

Out []:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130	8.0
0.75	0.615	0.40	0.165	1.1530	0.502	0.2030	0.329	11.0

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

Out []:

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

```
In [ ]: low=quant.loc[0.25]-(1.5*iqr)
low
```

Out []:

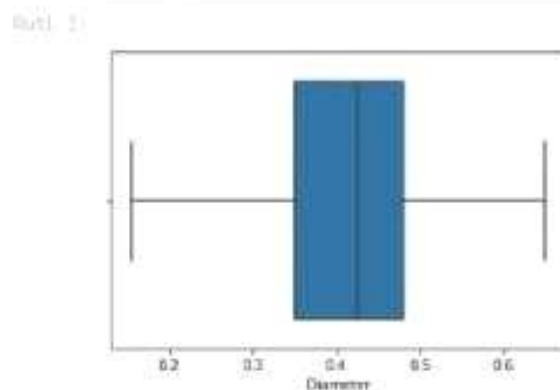
Length	0.20250
Diameter	0.15500
Height	0.04000
Whole weight	-0.62575
Shucked weight	-0.28800
Viscera weight	0.14575
Shell weight	-0.16850
Rings	3.50000
dtype:	float64


```
Out[ ]: 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
```

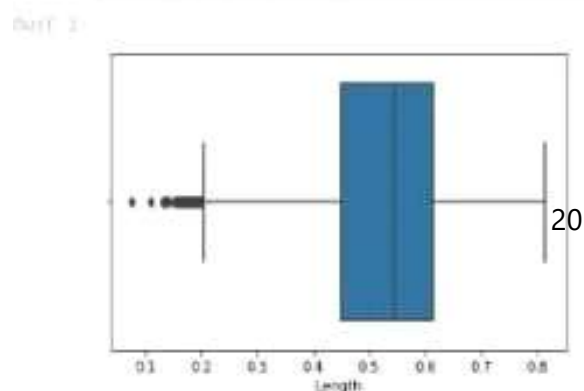
```
In [ ]: up=quant,loc[0.75]*(1.5*iqr)
up:
```

```
Out[ ]: 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
```

```
In [ ]: data['Diameter']=np.where(data['Diameter']<0.155,0.4078,data['Diameter'])
sns.boxplot(data['Diameter'])
```

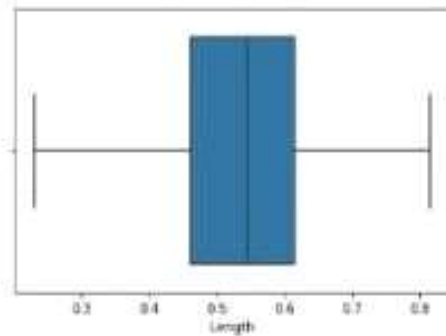


```
In [ ]: sns.boxplot(data['Length'])
```



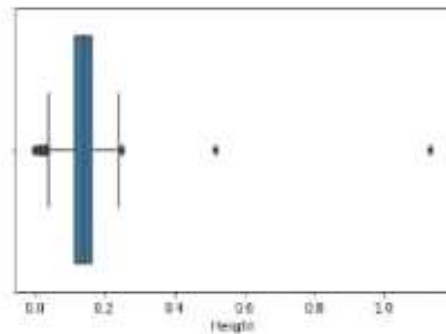

```
In [ ]: data['Length']=np.where(data['Length']<0.23,0.52, data['Length'])
sns.boxplot(data['Length'])
```

Out[]:



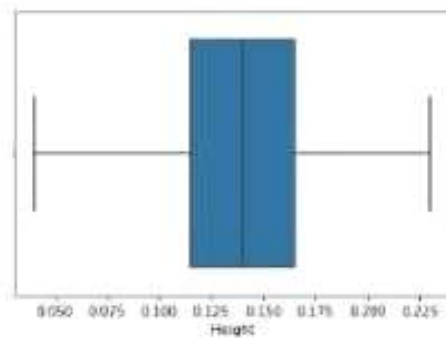
```
In [ ]: sns.boxplot(data['Height'])
```

Out[]:



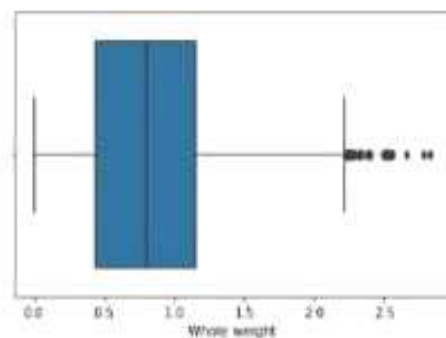
```
In [ ]: data['Height']=np.where(data['Height']<0.04,0.139, data['Height'])
data['Height']=np.where(data['Height']>0.23,0.139, data['Height'])
sns.boxplot(data['Height'])
```

Out[]:



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

Out[]:

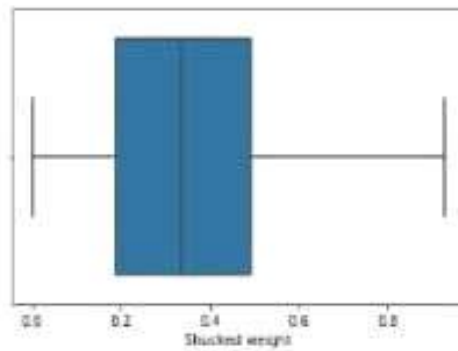


```
In [ ]: data['Whole weight']=np.where(data['Whole weight']>0.9,0.82, data['Whole weight'])
sns.boxplot(data['Whole weight'])
```



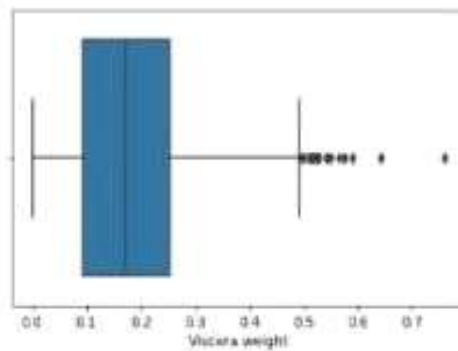
```
In [1]: data['Shucked weight']=np.where(data['Shucked weight']>0.93,0.35, data['Shu
sns.boxplot(data['Shucked weight'])
```

Out[1]:



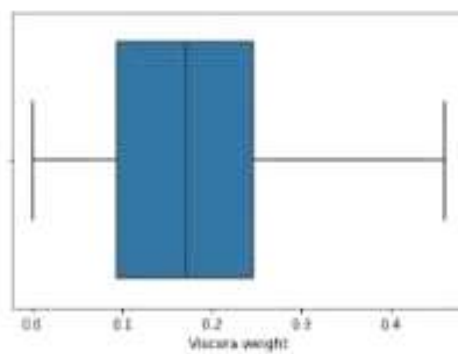
```
In [2]: sns.boxplot(data['Viscera weight'])
```

Out[2]:



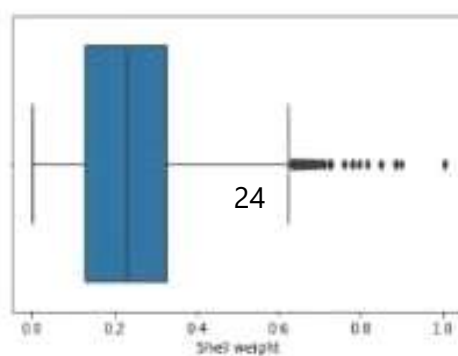
```
In [3]: data['Viscera weight']=np.where(data['Viscera weight']>0.46,0.19, data['Vis
sns.boxplot(data['Viscera weight'])
```

Out[3]:



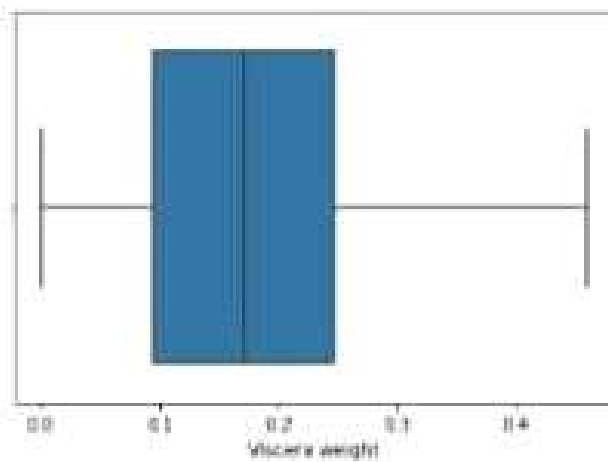
```
In [4]: sns.boxplot(data['Shell weight'])
```

Out[4]:



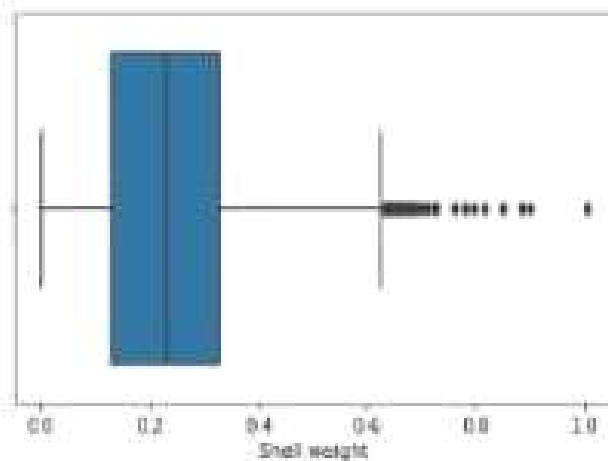

```
In [ ]: data['Viscera weight']=np.where(data['Viscera weight']>0.46,0.18, data)
sns.boxplot(data['Viscera weight'])
```

Out[]:



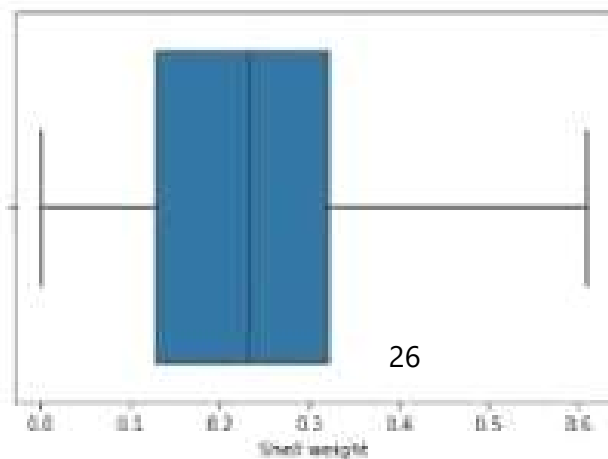
```
In [ ]: sns.boxplot(data['Shell weight'])
```

Out[]:



```
In [ ]: data['Shell weight']=np.where(data['Shell weight']>0.61,0.2388, data)
sns.boxplot(data['Shell weight'])
```

Out[]:



6. Check for Categorical columns and perform encoding.

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

```
Out[ ]:
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Sho
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	1	0.350	0.265	0.090	0.2255	0.0965	0.0485	
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	2	0.330	0.255	0.080	0.2050	0.0665	0.0395	
...
4172	0	0.565	0.450	0.165	0.6670	0.3700	0.2390	
4173	1	0.590	0.440	0.135	0.6200	0.4390	0.2145	
4174	1	0.600	0.475	0.205	0.6200	0.5255	0.2675	
4175	0	0.625	0.485	0.150	0.6200	0.5310	0.2610	
4176	1	0.710	0.555	0.195	0.6200	0.3580	0.3765	

4177 rows x 9 columns

7. Split the data into dependent and independent variables.

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

```
Out[ ]:
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Sho
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	1	0.350	0.265	0.090	0.2255	0.0965	0.0485	
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	2	0.330	0.255	0.080	0.2050	0.0665	0.0395	
...
4172	0	0.565	0.450	0.165	0.6670	0.3700	0.2390	
4173	1	0.590	0.440	0.135	0.6200	0.4390	0.2145	
4174	1	0.600	0.475	0.205	0.6200	0.5255	0.2675	
4175	0	0.625	0.485	0.150	0.6200	0.5310	0.2610	
4176	1	0.710	0.555	0.195	0.6200	0.3580	0.3765	

4177 rows × 8 columns

```
In [1]: y:
```

```
Out[1]:
```

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

8. Scale the independent variables

```
In [1]: from sklearn.preprocessing import scale
x = scale(x)
x
```

```
Out[1]:
```

```
array([[ -0.0105225 , -0.67088921, -0.501796
94, ..., -0.61037964,
        -0.7328165 , -0.64358742],
       [ -0.0105225 , -1.61376082, -1.573044
87, ..., -1.22513334,
        -1.24343929, -1.25742181],
       [ -1.26630752,  0.00259051,  0.087389
42, ..., -0.45300269,
        -0.33890749, -0.18321163],
       ...,
       [ -0.0105225 ,  0.63117159,  0.676575
77, ...,  0.86994729,
         1.08111018,  0.56873549],
       [ -1.26630752,  0.85566483,  0.783700
57, ...,  0.89699645,
         0.82336724,  0.47666033],
       [ -0.0105225 ,  1.61894185,  1.533574
12, ...,  0.00683308,
         1.94673739,  2.00357336]])
```

9. Split the data into training and testing

```
In [1]: 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_train.shape, x_test.shape)
```

(3341, 8) (836, 8)

10. Build the Model

```
In [1]: from sklearn.linear_model import LinearRegression
MLR=LinearRegression()
```

11. Train the model

```
In [1]: MLR.fit(x_train,y_train)
```

```
Out[1]: LinearRegression()
```


12. Test the model

```
In [15]: y_pred=MLR.predict(x_test)
         y_pred
```

```
Out[15]: array([ 6.27730521,  5.11464173, 11.2906194
7,   8.84719371, 11.31342551,
        14.27587505, 11.89677849, 12.3964822
5,   8.55248601,  8.08961834,
        12.09449868, 10.56528709,  9.7895849
9,   8.59686646,  7.76585939,
        8.47357248, 11.36977123,  9.5280555
6, 12.36997291,  6.51973298,
        6.71785594, 11.05744841, 11.6901007
4, 10.75739263,  6.5544077 ,
        6.82824096,  9.5306839 ,  7.5119168
9,   5.82377217, 10.47024617,
        13.13730038, 10.34700988, 11.4119617
7, 10.59789269, 13.25077032,
        14.82997416, 12.28691696, 10.9214164
, 12.87901037, 11.59049406,
        8.5462146 ,  8.52536272,  9.9537730
9,   7.94745203,  6.85150487,
        9.45338836,  8.86394805, 11.5806935
8,   6.06270743,  4.07194007,
        10.72813151,  8.62455986, 10.9224726
4,   8.31707157,  3.31458267,
        10.83423943,  9.36311705,  9.9259695
7,   7.17213853, 11.12938437,
        13.91273686,  7.42159167,  9.9633253
4, 13.92006698, 11.33472246,
        9.06493075, 10.20822237,  9.2684450
5, 10.24458569,  8.00893436,
        6.65277356,  9.9852585 ,  8.5866735
2, 11.43900078,  9.16014079,
        9.50575436, 10.974906 ,  9.3115556
1, 10.85487744, 10.47876918,
        10.89867355,  9.57567238,  7.3775531
6, 10.26968745,  8.68813991,
        9.66582988,  3.98888101,  6.6801349
2,   9.98844442,  8.20535208,
        14.65659649, 11.55465815, 10.8217179
7,   6.76120381,  8.9003516 ,
        13.21613708, 10.1018605 ,  8.2471845
3,   7.45995921, 10.21992407,
        11.59425676, 10.66513659, 13.3792795
6, 10.94076906, 10.60418916,
        11.27500775,  8.12004033, 13.0246637
```


4, 10.75778470, 11.49138144,
 6.20410519, 8.62148935, 12.5795445
 , 6.69882956, 12.00242043,
 9.3306056 , 9.98680774, 8.9656372
 5, 10.93545618, 6.79192911,
 9.70880805, 10.70932137, 8.9335340
 2, 9.50496905, 8.07991477,
 8.95737614, 10.93209418, 7.7644968
 , 8.13978903, 10.95470452,
 10.31933582, 13.16703248, 10.1301335
 8, 9.83546825, 9.90633339,
 7.01510185, 7.90500802, 11.0718584
 9, 9.29098045, 11.97645598,
 10.72209476, 8.01227134, 12.5263092
 , 12.14710337, 7.46207783,
 6.40507134, 11.11733617, 7.3223731
 3, 11.74140484, 10.62073403,
 6.7070387 , 8.26673972, 8.0359016
 5, 11.01898624, 6.86794371,
 10.81252025, 10.428314 , 8.7164497
 8, 11.55298684, 10.54060743,
 7.8005991 , 7.77758003, 12.8386618
 , 8.93468911, 11.39965158,
 14.96420885, 3.44108358, 12.1693500
 9, 5.50036395, 11.35043959,
 11.89531817, 10.4970892 , 10.3551774
 8, 12.2719997 , 8.58613842,
 11.96040691, 7.91529244, 9.9775962
 9, 12.80717361, 7.84490672,
 11.63929871, 10.9799557 , 13.2329783
 9, 7.46809636, 8.32165892,
 7.57491949, 9.60052813, 9.3036159
 3, 9.48560973, 9.0381744 ,
 11.86024904, 10.77236488, 10.1931920
 8, 5.88606055, 8.97293908,
 9.02729706, 11.00195158, 10.7400269
 1, 8.01759076, 13.44832209,
 11.04252635, 10.85167432, 6.9659662
 5, 6.49943453, 11.22159222,
 8.02736878, 6.63174433, 10.2306847
 1. 7.84382717. 11.44647106.

In []:

```
#initializing model
rg=Ridge(alpha=0.01,normalize=True)
#fit the model
rg.fit(x_train,y_train)
Ridge(alpha=0.01,normalize=True)
#prediction
rg_pred=rg.predict(x_test)
rg_pred
```

Out []:

```
array([ 6.30300957,  5.24101358, 11.239
9,  8.80569939, 11.23497312,
        14.13797601, 11.92160215, 12.311
,  8.52939227,  8.10668036,
        12.02589171, 10.60144613,  9.846
9,  8.56291595,  7.78357519,
        8.5060612 , 11.3805872 ,  9.716
2, 12.23013361,  6.56583238,
        6.71061615, 11.09881244, 11.685
3, 10.75006504,  6.53321358,
        6.80701928,  9.50457155,  7.506
9,  5.82174793, 10.46987678,
        13.0187986 , 10.34574355, 11.401
, 10.68002897, 13.15798051,
        14.66038279, 12.27424653, 10.896
7, 12.88649665, 11.51460947,
        8.5070967 ,  8.71115424,  9.976
3,  7.86515354,  6.87069919,
        9.45904819,  8.91527508, 11.550
1,  6.06653215,  4.20261672,
        10.70874698,  8.71861134, 10.886
9,  8.34049892,  3.81754284,
        10.82059178,  9.39860756,  9.947
9,  7.193782 , 11.14099571,
        13.90129992,  7.40095334, 10.006
7, 13.80959351, 11.32553828,
        9.12191088, 10.23254861,  9.373
3, 10.25176744,  8.00046819,
        6.66525407, 10.11599264,  8.595
3, 11.47038858,  9.13708106,
        9.51026325, 10.89937697,  9.486
2, 10.97606174, 10.53874014,
        11.01871713,  9.60450515,  7.367
8, 10.25527342,  8.68133152,
        9.75982871,  4.12374073,  6.743
5, 10.01097393,  8.23515612.
```



```

11.25513880, 12.49279281, 10.9107993
9, 6.84011069, 12.48921874,
    7.81304895, 8.95960113, 10.39999996
2, 5.43945682, 10.50917718,
    12.07995019, 8.87089164, 10.2151565
, 12.71058466, 7.23101045,
    7.50114506, 9.58725943, 11.3352038
9, 11.7963565 , 11.35488843,
    13.01264902, 10.2918353 , 12.8157642
5, 7.07085019, 12.48308462,
    11.34631202, 8.28239564, 6.5224844
4, 10.25674811, 10.83809242,
    10.03350984, 7.88923102, 9.657072
, 7.05413921, 10.20044713,
    9.81331394, 9.69054416, 8.0905216
1, 12.74964657, 11.20405624,
    9.74627941, 11.47087059, 10.8641454
2, 7.95846639, 10.36850946,
    12.08673065, 5.60767597, 9.7008085
1, 8.63857986, 6.62962512,
    10.75725404, 6.67523629, 9.5239507
4, 12.82208937, 7.0549315 ,
    9.94693633, 11.17738827, 11.1924380
4, 10.21193312, 11.70131552,
    8.11838748, 9.96414209, 6.4883403
4, 10.00112698, 10.31351969,
    9.32608025, 10.23928034, 11.0186451
1, 10.69788454, 10.01110364,
    8.61391512, 11.67185473, 11.5202983
8, 11.52419116, 12.49302801,
    4.26263982])

```

```
In [1]: pred=NLR.predict(x_train)
pred
```

```
Out[1]: array([13.90916896,  7.94417688, 10.9917352
9, ...,  9.21077865,
        6.26813443,  9.88590822])
```

```
In [2]: from sklearn.metrics import r2_score
accuracy=r2_score(y_test,y_pred)
accuracy
```

```
Out[2]: 0.4482390430138421
```

```
In [3]: NLR.predict([[1,0.455,0.365,0.095,0.5140,0.2245,0.1010,0.150]])
```

```
Out[3]: array([9.8732734])
```


3, 13.23913023, 11.30083311,
 10.3368754 , 7.10758876,
 1, 7.85402001, 10.00198933,
 9.70641069, 11.0597554 , 1
 , 10.58800852, 10.66081442,
 9.05442746, 8.60070572, 1
 4, 12.19261652, 11.13426365,
 13.48329978, 6.76362585, 1
 , 10.55683449, 10.08735129,
 10.20261149, 7.83069389, 1
 8, 7.92121782, 11.19802171,
 6.39283708, 11.14220166, 1
 9, 7.12056043, 8.62559626,
 9.17090339, 10.60670375, 1
 3, 12.78044861, 10.9109979 ,
 9.85095938, 10.53183932,
 9, 11.73066479, 11.54927977,
 11.44445225, 10.72689747,
 2, 7.72639271, 9.85787237,
 9.47315769, 16.27772218,
 , 11.19679134, 6.65814151,
 9.45046154, 12.56182568, 1
 8, 11.3506093 , 11.8577869 ,
 8.76211858, 7.98252027,
 6, 9.95275773, 6.64554143,
 3.86887792, 11.5768315 , 1
 9, 11.04695105, 8.70438878,
 12.42880884, 12.48815433, 1
 1, 9.98550377, 10.05265494,
 9.98508902, 6.81325368, 1
 6 13.4379969 8.19654646

1, 9.67780978, 9.889152 ,
 12.55693421, 7.12772136, 10.292630
 3, 7.06043327, 10.59688463,
 9.76670251, 12.15490598, 7.829933
 3, 8.85667009, 5.68322798,
 6.23088727, 12.24204903, 9.125320
 2, 13.43394753, 13.49074696,
 8.15668467, 7.68124075, 8.178077
 , 4.34582983, 7.44964188,
 9.25941614, 10.55019847, 8.330454
 2, 13.43850814, 7.20071343,
 12.84542293, 12.56332314, 7.675610
 2, 10.31351961, 9.88236469,
 7.12065454, 9.46403006, 10.560374
 3, 9.79024532, 10.83257724,
 11.19582412, 7.46628468, 8.651299
 1, 11.49306076, 12.34360091,
 10.23316195, 7.38545576, 12.695238
 7, 7.25072387, 9.51996597,
 9.23589149, 10.28516715, 12.254872
 4, 11.70938634, 5.89869058,
 10.27829349, 9.84350759, 11.078654
 3, 11.69545111, 7.97940467,
 10.05127456, 12.95302658, 12.711692
 2, 7.11295539, 15.32010872,
 10.57859706, 10.52385061, 11.325927
 5, 9.68231005, 13.92790455,
 9.81344796, 10.30727476, 11.038678
 9, 6.48670499, 8.26933443,
 11.31510793, 10.18991685, 6.451202
 1, 7.52095425, 10.73988282,
 10.91873047, 15.62162486, 9.598457
 2, 11.19155549, 12.38342095,
 12.59833132, 6.12248511, 6.789396
 4, 10.72678692, 11.41095463,
 9.3126676 , 10.76734849, 10.688672
 4, 5.99287938, 12.31962914,


```

11.52420400, 8.20500050,
7, 10.39422657, 10.88813401,
10.05576314, 7.90682188,
, 7.05636532, 10.30757546,
9.83786911, 9.74111195,
3, 12.7508793 , 11.18544708,
9.8280501 , 11.5146657 ,
6, 7.98652396, 10.38069133,
12.05590344, 5.6183766 ,
5, 8.77410128, 6.61246081,
10.77826513, 6.65080491,
9, 12.74366316, 7.066261 ,
10.08086815, 11.12085743,
2, 10.12093457, 11.66397296,
8.11851979, 10.10293128,
6, 9.98127971, 10.33284013,
9.35406561, 10.28532244,
6, 10.79017291, 10.13515047,
8.65638785, 11.64237009,
9, 11.63134495, 12.43684096,
4.39564549])

```

```
rg.coef_
```

```

array([-0.34874321, -0.70989254,
5, 1.02984113, 0.94593211,
-1.45851724, -0.14684477,
4])

```

```
metrics.r2_score(y_test,rg_pred)
```

```
0.4493030433197964
```

44

```
np.sqrt(mean_squared_error(y_test,rg_pred))
```

```
2.401672354777648
```


8.70681169, 10.35531895, 11.82
 , 10.69012783, 10.40917452,
 9.00251064, 10.12524793, 8.00
 4, 7.01803139, 9.89605901,
 10.95301139, 10.11983309, 13.31
 9, 9.3845876 , 10.72954916,
 7.80515437, 8.7176603 , 9.04
 9, 11.95110897, 6.27650075,
 11.19367599, 12.39295814, 10.90
 , 6.8782812 , 12.47837181,
 7.82297172, 8.94264056, 10.44
 8, 5.41903381, 10.46294082,
 12.05447988, 8.87387261, 10.29
 8, 12.63557146, 7.25245407,
 7.47893592, 9.61199559, 11.26
 1, 11.79322916, 11.31154571,
 12.94035422, 10.28883135, 12.72
 1, 7.10712989, 12.4468316 ,
 11.32426408, 8.28568658, 6.53
 7, 10.39422657, 10.88813401,
 10.05576314, 7.90682188, 9.60
 , 7.05636532, 10.30757546,
 9.83786911, 9.74111195, 8.15
 3, 12.7508793 , 11.18544708,
 9.8280501 , 11.5146657 , 10.93
 6, 7.98652396, 10.38069133,
 12.05590344, 5.6183766 , 9.70
 5, 8.77410128, 6.61246081,
 10.77826513, 6.65080491, 9.63
 9, 12.74366316, 7.066261 ,
 10.08086815, 11.12085743, 11.24
 2, 10.12093457, 11.66397296,
 8.11851979, 10.10293128, 6.48
 6, 9.00107071, 10.33304013

Out[1]: 2.403991367956563

LASSO

```
In [1]: from sklearn.linear_model import Lasso, Ridge
#initialising model
lso=Lasso(alpha=0.01,normalize=True)
#fit the model
lso.fit(x_train,y_train)
Lasso(alpha=0.01, normalize=True)
#prediction on test data
lso_pred=lso.predict(x_test)
#coef
coef=lso.coef_
coef
```

Out[1]: array([-0.01293987, 0. , 0. ,
 0.50666281, 0.15925177,
 0. , 0. , 0.7739190
3])

```
In [2]: from sklearn import metrics
from sklearn.metrics import mean_squared_error
metrics.r2_score(y_test,lso_pred)
```

Out[2]: 0.36871210321772163

```
In [3]: np.sqrt(mean_squared_error(y_test,lso_pred))
```

Out[3]: 2.571408956644621

RIDGE

```
In [4]: #initialising model
rg=Ridge(alpha=0.01,normalize=True)
#fit the model
rg.fit(x_train,y_train)
Ridge(alpha=0.01, normalize=True)
#prediction
rg_pred=rg.predict(x_test)
rg_pred
```

Out[4]: array([6.30300957, 5.24101358, 11.2391992
9, 8.80569939, 11.23497312,
 14.13797601, 11.92160215, 12.3113006
 , 8.52939227, 8.10668036,
 12.02589171, 10.60144613, 9.8464354
9, 8.56291595, 7.78357519,
 8.5060612 , 11.3805872 , 9.7168214
2, 12.23013361, 6.56583238,
 6.71061615, 11.09881244, 11.6851442
3, 10.75006504, 6.53321358,
 6.80701928, 9.50457155, 7.5064711
9, 5.82174793, 10.46987678,
 13.0187986 , 10.34574355, 11.4015058
 , 10.68002897, 13.15798051,
 14.66038279, 12.27424653, 10.8962319
7, 12.88649665, 11.51460947,
 8.5070967 , 8.71115424, 9.9760314
3, 7.86515354, 6.87069919,
 9.45904819, 8.91527508, 11.5507249
1, 6.06653215, 4.20261672,
 10.70874698, 8.71861134, 10.8866879
9, 8.34049892, 3.81754284,
 10.82850178, 0.32860756, 0.04720228

6, 10.47965546, 7.37206624,
 14.30853246, 11.17685101, 6.3923162
 6, 13.29016329, 11.38371139,
 10.23085817, 7.06496936, 6.2948497
 6, 7.89811317, 10.02660635,
 9.74137391, 11.04422851, 11.9662958
 2, 10.49765854, 10.61100165,
 9.07926388, 8.59550758, 11.9706804
 3, 12.1800007, 11.15351352,
 13.41804188, 6.7462837, 14.3025865
 7, 10.50088566, 10.14606343,
 10.07070828, 7.79519418, 12.3649476
 5, 7.91204745, 11.19841461,
 6.30886416, 11.09355661, 10.9570427
 3, 7.0782196, 8.56331832,
 9.15812866, 10.58092144, 10.3240889
 , 12.87637462, 10.8709905,
 9.86989967, 10.53119798, 7.3565180
 9, 11.71944329, 11.59601023,
 11.43064552, 10.79847017, 9.7888680
 9, 7.66885054, 9.85334605,
 9.42365792, 16.51180007, 9.9426521
 4, 11.28639786, 6.66873002,
 9.42050294, 12.5832539, 15.9618033
 4, 11.29038383, 11.8568694,
 8.77587437, 7.96089774, 7.7164742
 8, 10.00735736, 6.64929106,
 3.36917219, 11.57092155, 10.1219976
 7, 11.1501544, 8.73720749,
 12.53011009, 12.63591866, 11.6942705
 4, 10.01649554, 9.97193676,
 9.96563833, 6.69621385, 12.5773895
 8, 13.53572912, 8.12958148,
 9.04406337, 14.69199149, 8.0065457
 8, 10.05883603, 9.49369772,
 9.65283002, 10.88384891, 6.2928482
 2, 12.5853, 5.77106864,
 6.84710247, 13.39385554, 6.5972879
 , 10.28184012, 8.77071666,