Import the required libraries

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading of dataset

In [ ]:

```
df = pd.read_csv("/content/abalone.csv")
df
```

Out[ ]:

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

4177 rows × 9 columns

In [ ]:

```
#modifying the dataset

df['Age'] = df.Rings + 1.5

df=df.rename(columns = {'Whole weight':'Whole_weight','Shucked weight': 'Shucked_weigh
t','Viscera weight': 'Viscera_weight',
                            'Shell weight': 'Shell_weight'})
df
```

Out[ ]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	
...	...	...	...	...	...	...	...	
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	

4177 rows × 10 columns

In [ ]:

```
df.shape
```

Out[ ]:

```
(4177, 10)
```

In [ ]:

```
df.mean
```

Out[ ]:

```
<bound method NDFrame._add_numeric_operations.<locals>.mean of       Sex  L
ength  Diameter  Height  Whole_weight  Shucked_weight  \
0      M   0.455     0.365   0.095        0.5140          0.2245
1      M   0.350     0.265   0.090        0.2255          0.0995
2      F   0.530     0.420   0.135        0.6770          0.2565
3      M   0.440     0.365   0.125        0.5160          0.2155
4      I   0.330     0.255   0.080        0.2050          0.0895
...   ..     ...       ...     ...           ...             ...
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   Age
0             0.1010        0.1500     15  16.5
1             0.0485        0.0700      7   8.5
2             0.1415        0.2100      9  10.5
3             0.1140        0.1550     10  11.5
4             0.0395        0.0550      7   8.5
...              ...           ...    ...   ...
4172          0.2390        0.2490     11  12.5
4173          0.2145        0.2605     10  11.5
4174          0.2875        0.3080      9  10.5
4175          0.2610        0.2960     10  11.5
4176          0.3765        0.4950     12  13.5

[4177 rows x 10 columns]>
```

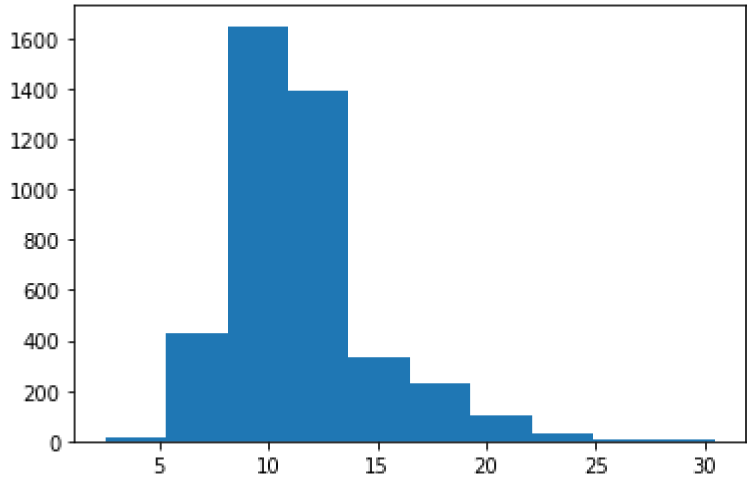
Visualization

In [ ]:

```
#Univarient analysis
plt.hist(df['Age'])
```

Out[ ]:

```
(array([  17.,  431., 1648., 1388.,  329.,  228.,  100.,   29.,    4.,
           3.]),
 array([ 2.5,  5.3,  8.1, 10.9, 13.7, 16.5, 19.3, 22.1, 24.9, 27.7, 30.
5]),
 <a list of 10 Patch objects>)
```



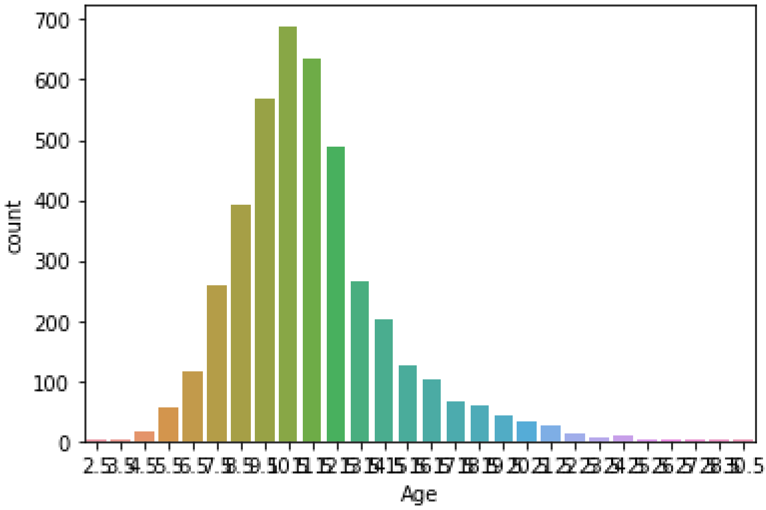
In [ ]:

```
sns.countplot(df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa5afa97750>
```

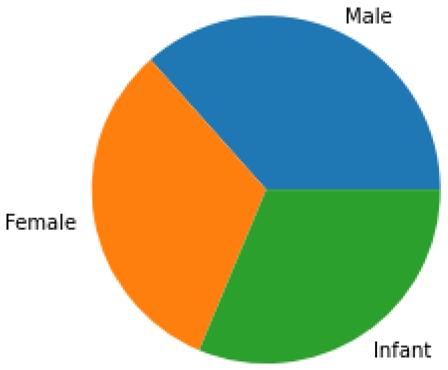


In [ ]:

```
plt.pie(df['Sex'].value_counts(),labels=['Male','Female','Infant'])
```

Out[ ]:

```
([<matplotlib.patches.Wedge at 0x7fa5af4df050>,
  <matplotlib.patches.Wedge at 0x7fa5af560310>,
  <matplotlib.patches.Wedge at 0x7fa5af4df910>],
 [Text(0.45010440780275796, 1.0036961801643607, 'Male'),
  Text(-1.0848393519507589, -0.18199884741134378, 'Female'),
  Text(0.6099659291018239, -0.9153914820091724, 'Infant')])
```

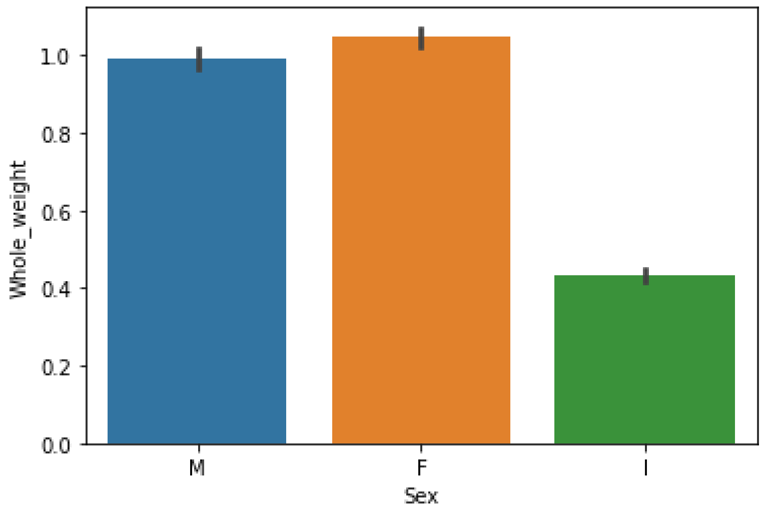


In [ ]:

```
#Bi-varient analysis:
sns.barplot(x = df.Sex,y = df.Whole_weight)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa5af4f8890>
```

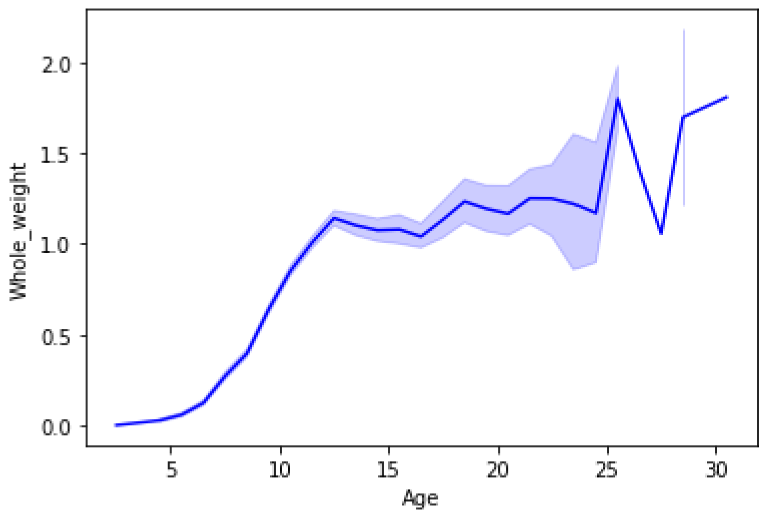


In [ ]:

```
sns.lineplot(x = df.Age, y = df.Whole_weight, color = 'blue')
```

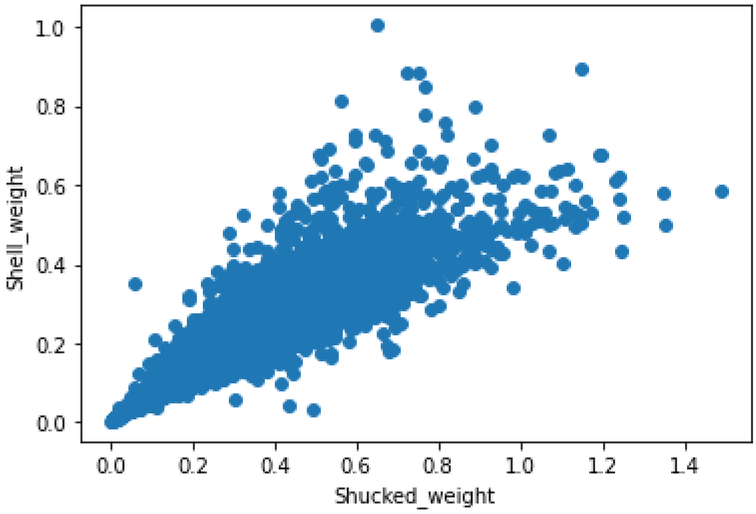
Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa5af458590>
```



In [ ]:

```
plt.scatter(df.Shucked_weight,df.Shell_weight)
plt.xlabel("Shucked_weight")
plt.ylabel("Shell_weight")
sns.set_style('ticks')
```



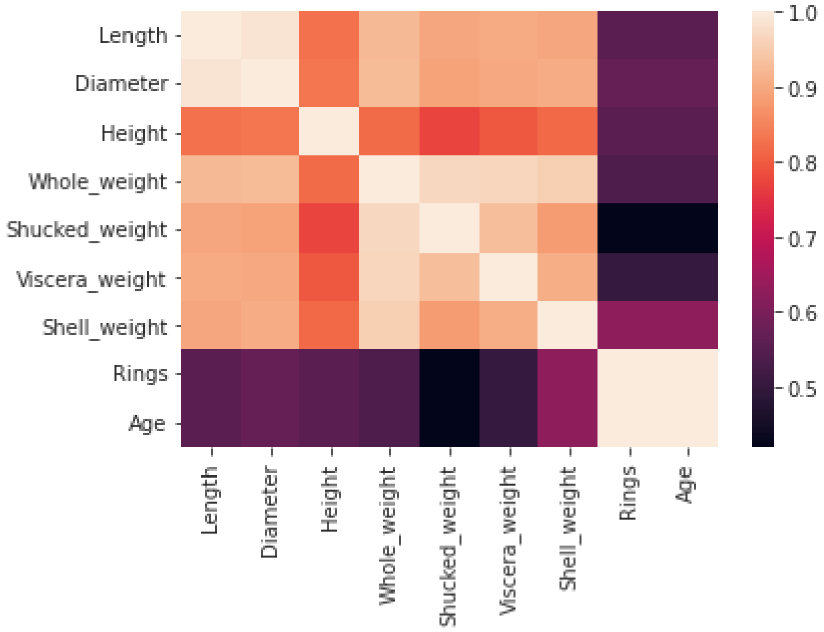
In [ ]:

```
#multi varient analysis

corr = df.corr()
sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns)
```

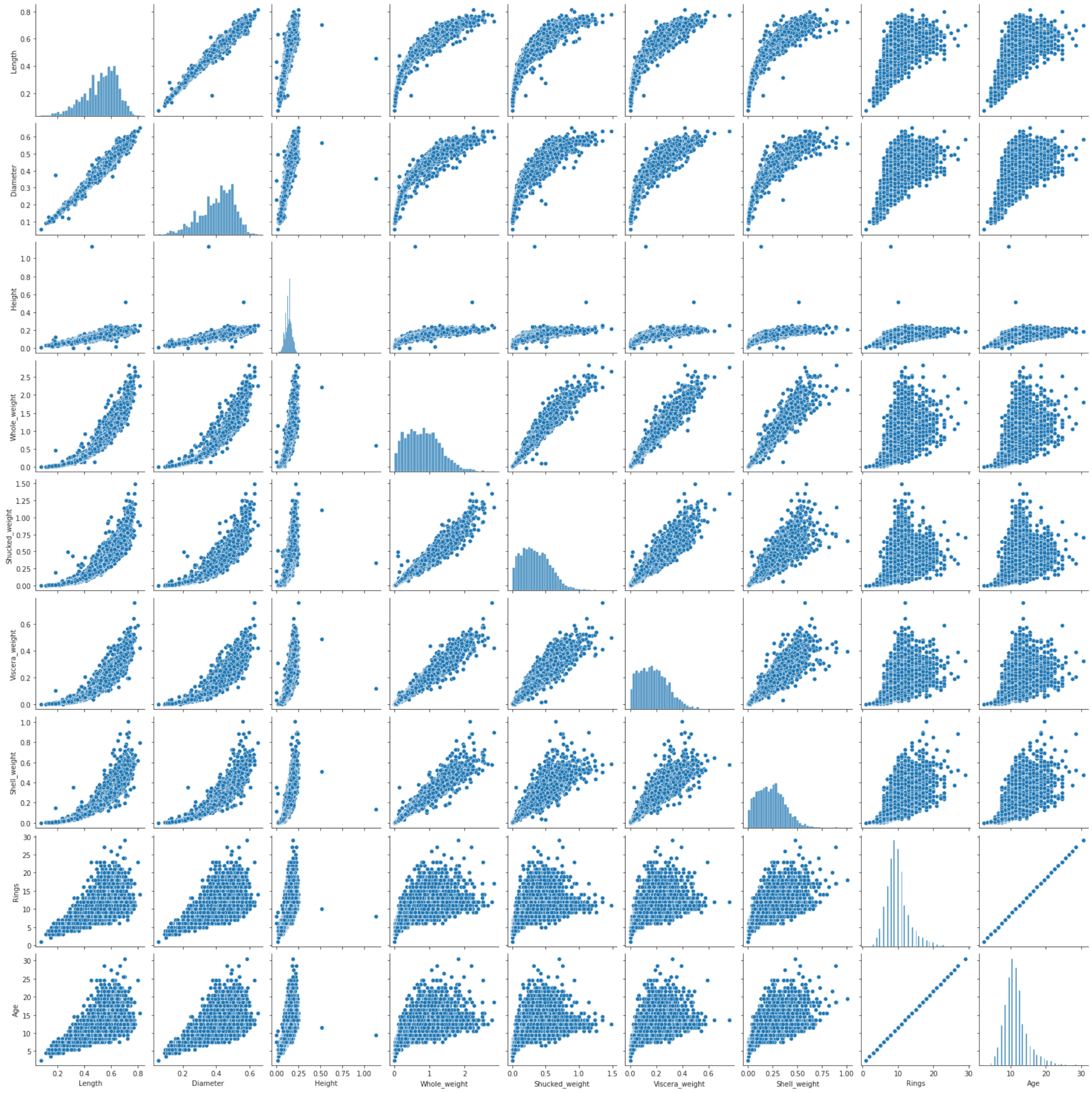
Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa5af36fe50>
```



In [ ]:

```
sns.pairplot (df)
plt.show()
```



Descriptive statistics

In [ ]:

```
df.describe()
```

Out[ ]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weigh
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.00000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.18059
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.10961
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.00050
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.09350
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.17100
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.25300
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.76000

Handling missing values

In [ ]:

```
df.isnull().sum()
```

Out[ ]:

```
Sex               0
Length            0
Diameter          0
Height            0
Whole_weight      0
Shucked_weight    0
Viscera_weight    0
Shell_weight      0
Rings             0
Age               0
dtype: int64
```

Outliers handling

In [ ]:

```
outliers=df.quantile(q=(0.25,0.75))
outliers
```

Out[ ]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130
0.75	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329

In [ ]:

```
q1 = df.Age.quantile(0.25)
q2 = df.Age.quantile(0.75)
q3 = q2 - q1
lower_limit = q1 - 1.5 * q3
df.median(numeric_only=True)
```

Out[ ]:

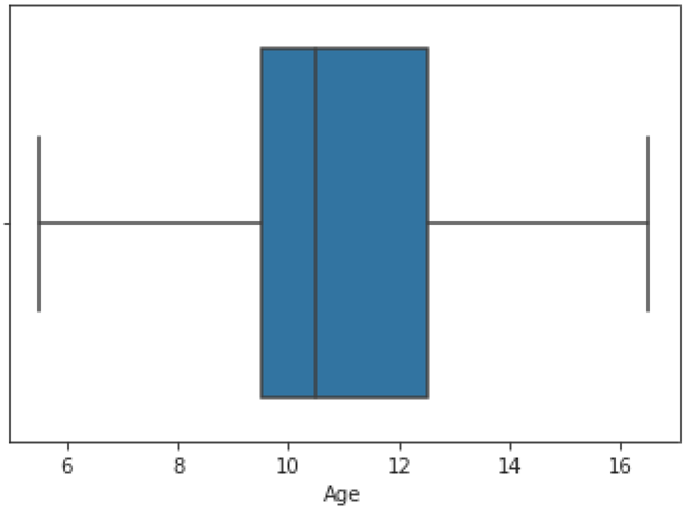
```
Length             0.5450
Diameter           0.4250
Height             0.1400
Whole_weight       0.7995
Shucked_weight     0.3360
Viscera_weight     0.1710
Shell_weight       0.2340
Rings              9.0000
Age               10.5000
dtype: float64
```

In [ ]:

```
df['Age'] = np.where(df['Age'] < lower_limit, 7, df['Age'])
sns.boxplot(x=df.Age,showfliers = False)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa5ab05f090>
```



Encoding

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
encode = LabelEncoder()
df.Sex = encode.fit_transform(df.Sex)
df
```

Out[ ]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	
...	...	...	...	...	...	...	...	
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	
4173	2	0.590	0.440	0.135	0.9660	0.4390	0.2145	
4174	2	0.600	0.475	0.205	1.1760	0.5255	0.2875	
4175	0	0.625	0.485	0.150	1.0945	0.5310	0.2610	
4176	2	0.710	0.555	0.195	1.9485	0.9455	0.3765	

4177 rows × 10 columns

Independent and Dependent Variables

In [ ]:

```
#independent variable

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

Out[ ]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	
...	...	...	...	...	...	...	...	
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	
4173	2	0.590	0.440	0.135	0.9660	0.4390	0.2145	
4174	2	0.600	0.475	0.205	1.1760	0.5255	0.2875	
4175	0	0.625	0.485	0.150	1.0945	0.5310	0.2610	
4176	2	0.710	0.555	0.195	1.9485	0.9455	0.3765	

4177 rows × 8 columns

In [ ]:

```
#dependent variable

y = df.iloc[:,9:]
y
```

Out[ ]:

	Age
0	16.5
1	8.5
2	10.5
3	11.5
4	8.5
...	...
4172	12.5
4173	11.5
4174	10.5
4175	11.5
4176	13.5

4177 rows × 1 columns

Scaling

In [ ]:

```
from sklearn import preprocessing
standardisation = preprocessing.StandardScaler()
new_x = standardisation.fit_transform(x)
print(new_x)
```

```
[[ 1.15198011 -0.57455813 -0.43214879 ... -0.60768536 -0.72621157
  -0.63821689]
 [ 1.15198011 -1.44898585 -1.439929   ... -1.17090984 -1.20522124
  -1.21298732]
 [-1.28068972  0.05003309  0.12213032 ... -0.4634999  -0.35668983
  -0.20713907]
 ...
 [ 1.15198011  0.6329849   0.67640943 ...  0.74855917  0.97541324
   0.49695471]
 [-1.28068972  0.84118198  0.77718745 ...  0.77334105  0.73362741
   0.41073914]
 [ 1.15198011  1.54905203  1.48263359 ...  2.64099341  1.78744868
   1.84048058]]
```

In [ ]:

```
min_max_scaler = preprocessing.MinMaxScaler()
new_x = min_max_scaler.fit_transform(x)
print(new_x)
```

```
[[1.         0.51351351 0.5210084  ... 0.15030262 0.1323239  0.14798206]
 [1.         0.37162162 0.35294118 ... 0.06624075 0.06319947 0.06826109]
 [0.         0.61486486 0.61344538 ... 0.17182246 0.18564845 0.2077728 ]
 ...
 [1.         0.70945946 0.70588235 ... 0.3527236  0.37788018 0.30543099]
 [0.         0.74324324 0.72268908 ... 0.35642233 0.34298881 0.29347285]
 [1.         0.85810811 0.84033613 ... 0.63517149 0.49506254 0.49177877]]
```

Split the data into training and testing

In [ ]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y)
```

Model building

In [ ]:

```
from sklearn.linear_model import LinearRegression
mlr=LinearRegression()
mlr.fit(x_train,y_train)
```

Out[ ]:

```
LinearRegression()
```

Training and testing

In [ ]:

```
x_test
```

Out[ ]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_
564	0	0.470	0.355	0.130	0.5465	0.2005	0.1260	
1584	0	0.515	0.375	0.110	0.6065	0.3005	0.1310	
3613	0	0.610	0.490	0.170	1.3475	0.7045	0.2500	
885	0	0.670	0.585	0.160	1.3090	0.5445	0.2945	
3106	1	0.300	0.220	0.065	0.1235	0.0590	0.0260	
...	...	...	...	...	...	...	...	
56	2	0.445	0.350	0.120	0.4425	0.1920	0.0955	
1890	2	0.565	0.455	0.155	0.9355	0.4210	0.1830	
1299	1	0.530	0.415	0.110	0.5745	0.2525	0.1235	
3392	2	0.645	0.515	0.185	1.4605	0.5835	0.3155	
195	2	0.500	0.405	0.155	0.7720	0.3460	0.1535	

1045 rows × 8 columns

In [ ]:

```
y_test
```

Out[ ]:

	Age
564	15.5
1584	7.5
3613	12.5
885	11.5
3106	6.5
...	...
56	9.5
1890	12.5
1299	10.5
3392	20.5
195	13.5

1045 rows × 1 columns

In [ ]:

```
from sklearn.metrics import r2_score
r2_score(mlr.predict(x_test),y_test)
```

Out[ ]:

```
0.0557067898421757
```