Importing of Libraries

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

Dataset Loading

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 [ ]:

```
#adding age
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)

Visualization

In [ ]:

```
#univarient analysis

df.groupby('Sex')[['Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight',
       'Viscera_weight', 'Shell_weight', 'Age']].mean().sort_values('Age')
```

Out[ ]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_we
Sex							
I	0.427746	0.326494	0.107996	0.431363	0.191035	0.092010	0.128
M	0.561391	0.439287	0.151381	0.991459	0.432946	0.215545	0.281
F	0.579093	0.454732	0.158011	1.046532	0.446188	0.230689	0.302

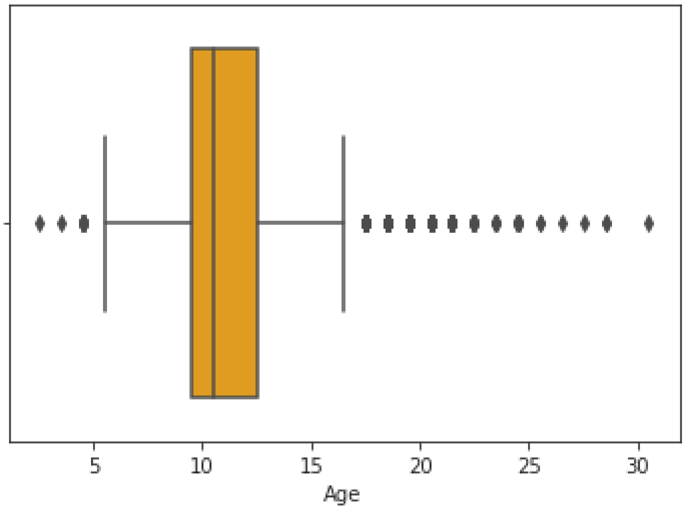
In [ ]:

```
#boxplot

sns.boxplot(x=df.Age,color='orange')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe456598810>
```

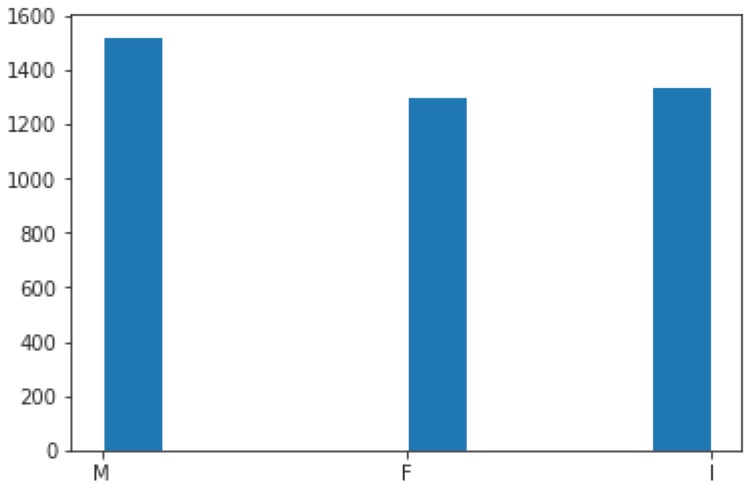


In [ ]:

```
plt.hist(df['Sex'])
```

Out[ ]:

```
(array([1528.,    0.,    0.,    0.,    0., 1307.,    0.,    0.,    0.,
        1342.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
 <a list of 10 Patch objects>)
```

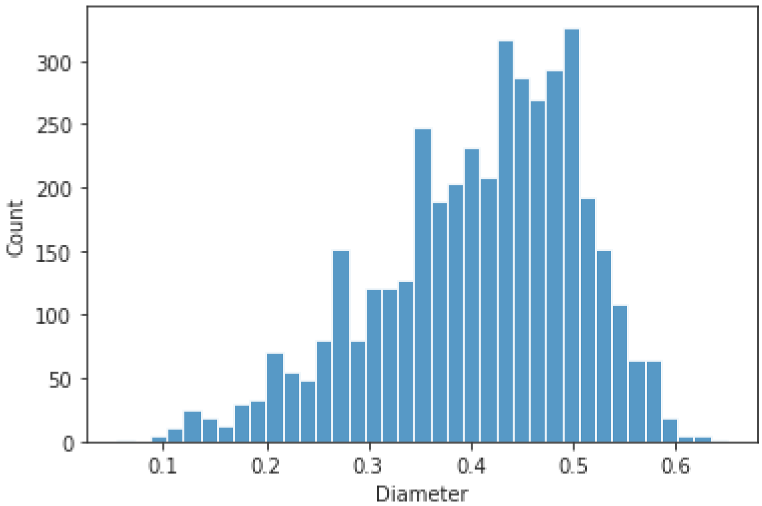


In [ ]:

```
sns.histplot(x=df.Diameter,palette='Rainbow')
```

Out[ ]:

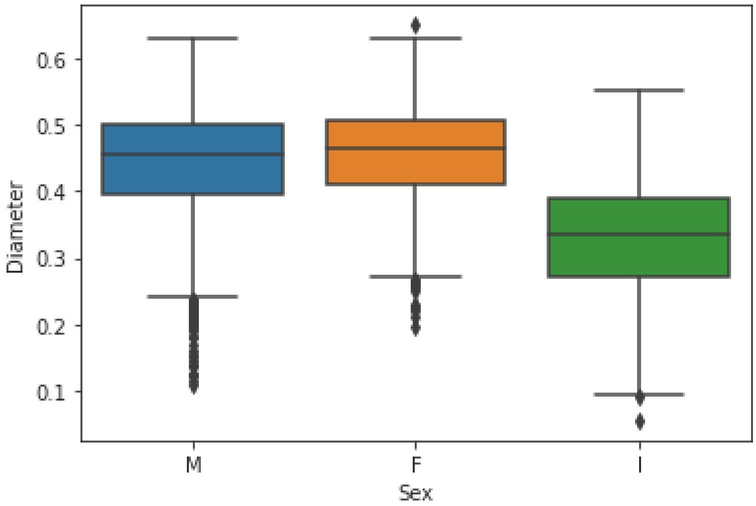
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe45a5fe690>
```



In [ ]:

```
#bi-varient analysis
#boxplot

sns.boxplot(x=df.Sex,y=df.Diameter,data=df)
plt.show()
```



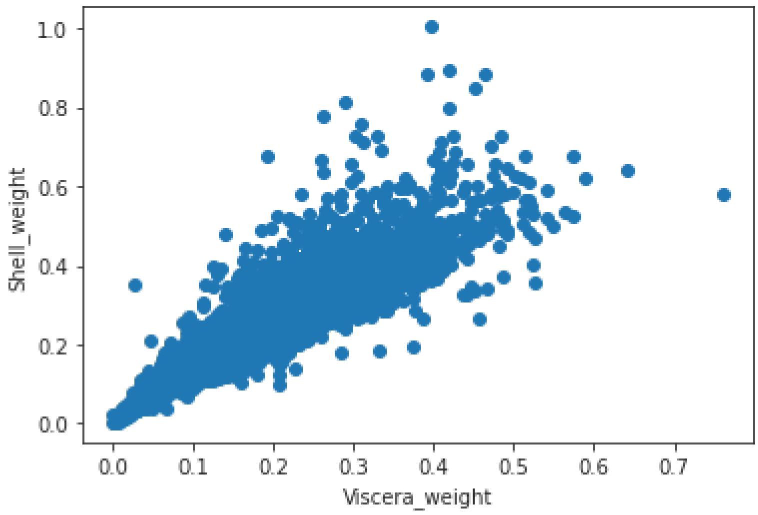
In [ ]:

```
#scatter plot

plt.scatter(df.Viscera_weight,df.Shell_weight)
plt.xlabel("Viscera_weight")
plt.ylabel("Shell_weight")
```

Out[ ]:

```
Text(0, 0.5, 'Shell_weight')
```

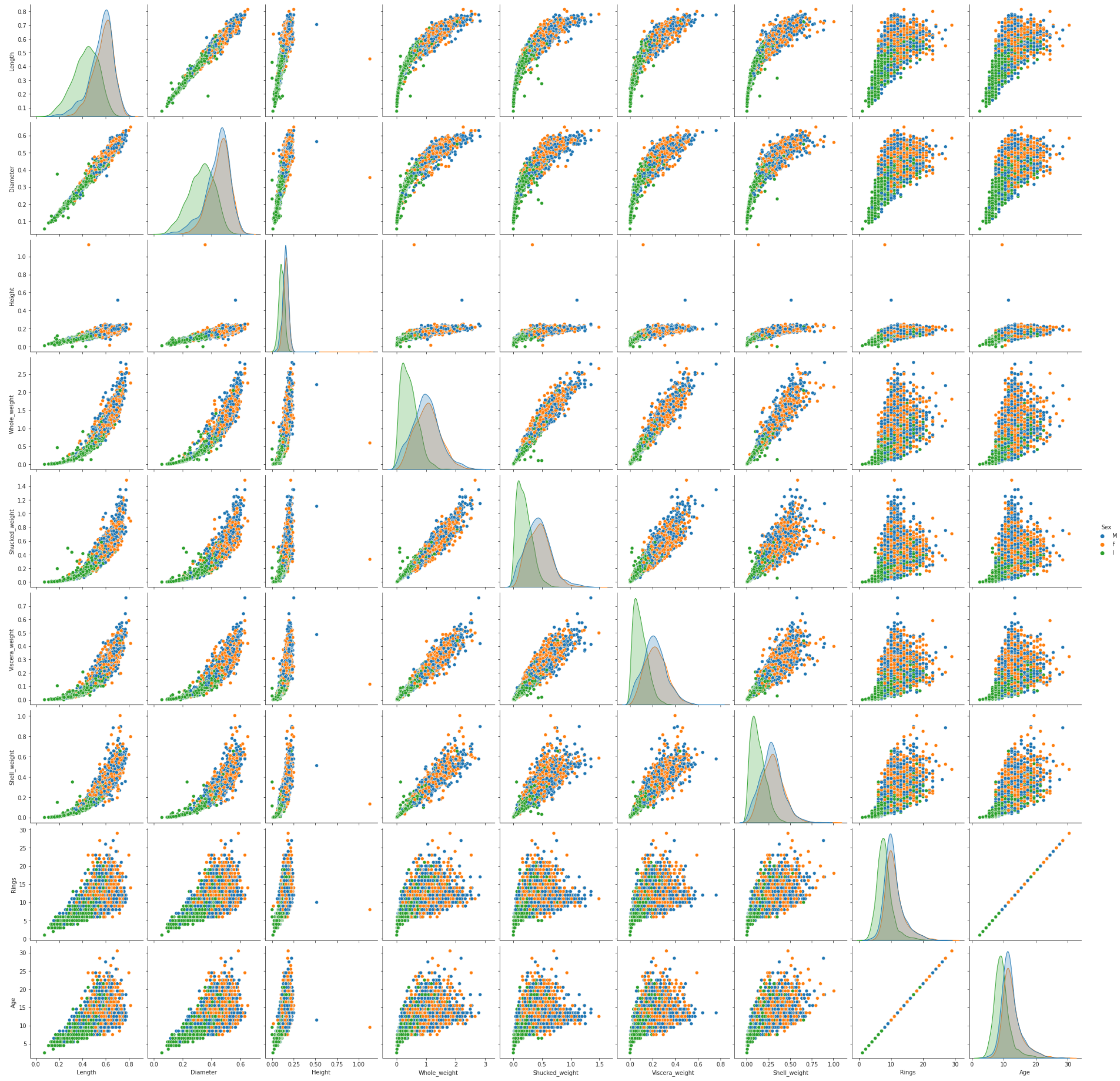


In [ ]:

```
#Multi-varient analysis

sns.pairplot (df, hue="Sex", size=3)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:2076: UserWarni
ng: The `size` parameter has been renamed to `height`; please update your
code.
  warnings.warn(msg, UserWarning)
```



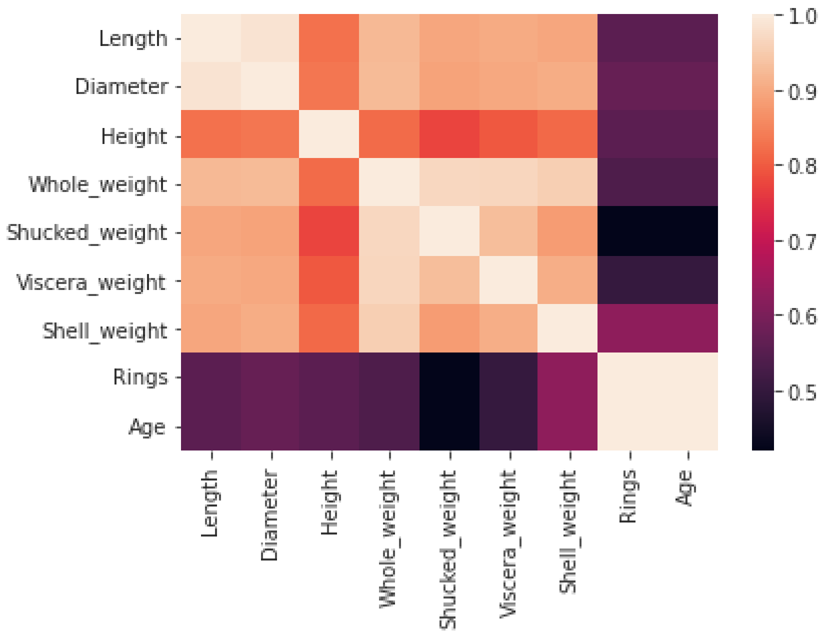
In [ ]:

```
#heatmap

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

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe459e6d210>
```



Statistics

In [ ]:

```
df.describe()
```

Out[ ]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	1.052909	0.523992	0.407881	0.139516	0.828742	0.359367
std	0.822240	0.120093	0.099240	0.041827	0.490389	0.221963
min	0.000000	0.075000	0.055000	0.000000	0.002000	0.001000
25%	0.000000	0.450000	0.350000	0.115000	0.441500	0.186000
50%	1.000000	0.545000	0.425000	0.140000	0.799500	0.336000
75%	2.000000	0.615000	0.480000	0.165000	1.153000	0.502000
max	2.000000	0.815000	0.650000	1.130000	2.825500	1.488000

In [ ]:

```
df.mean()
```

Out[ ]:

```
Sex                1.052909
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
Age               11.444577
dtype: float64
```

In [ ]:

```
df.mode()
```

Out[ ]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_wei
0	2.0	0.550	0.45	0.15	0.2225	0.175	0.1715	0.
1	NaN	0.625	NaN	NaN	NaN	NaN	NaN	N

In [ ]:

```
df.median()
```

Out[ ]:

```
Sex                1.0000
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 [ ]:

```
#Checking of Null values

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
```

In [ ]:

```
#Encoding

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 [ ]:

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

In [ ]:

```
print("Independent variable")
x
```

Independent variable

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 [ ]:

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

In [ ]:

```
print("Dependent variable")
y
```

Dependent variable

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

Handling of outliers

In [ ]:

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

Out[ ]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_v
0.25	0.0	0.450	0.35	0.115	0.4415	0.186	0.0935	
0.75	2.0	0.615	0.48	0.165	1.1530	0.502	0.2530	

In [ ]:

```
a=df.Age.quantile(0.25)
b=df.Age.quantile(0.75)
c=b-a
lower_limit = a-1.5*c
df.median(numeric_only=True)
```

Out[ ]:

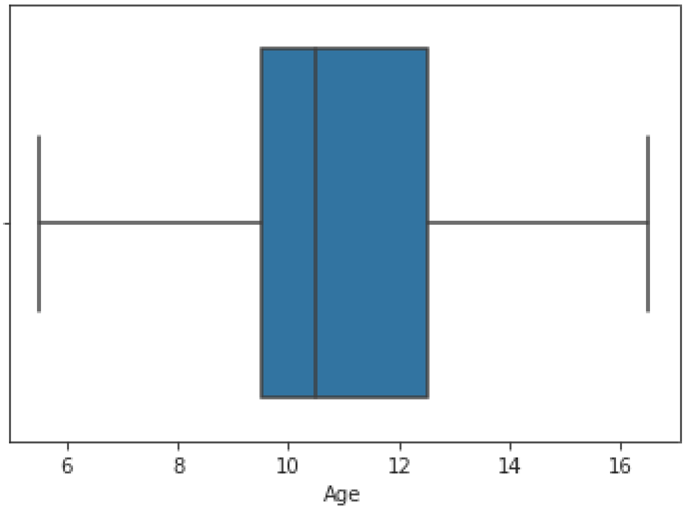
```
Sex                1.0000
Length             0.5450
Diameter           0.4250
Height             0.1400
Whole_weight       0.7995
Shucked_weight     0.3360
Viscera_weight     0.1710
Shell_weight       0.2340
Rings              9.0000
Age               10.5000
dtype: float64
```

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 0x7fe4513ba890>
```



Feature 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]]
```

Splitting the data *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)
```

Building the model

In [ ]:

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

Out[ ]:

```
LinearRegression()
```

In [ ]:

```
#training and testing

x_test[0:5]
```

Out[ ]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_
1957	0	0.645	0.520	0.210	1.5535	0.6160	0.3655	
1238	1	0.375	0.280	0.080	0.2025	0.0825	0.0480	
3277	0	0.465	0.390	0.140	0.5555	0.2130	0.1075	
2111	1	0.455	0.355	0.080	0.4520	0.2165	0.0995	
2649	2	0.505	0.400	0.135	0.7230	0.3770	0.1490	

In [ ]:

```
y_test[0:5]
```

Out[ ]:

	Age
1957	17.5
1238	9.5
3277	16.5
2111	10.5
2649	8.5

In [ ]:

```
mlr.predict(x_test[0:10])
```

Out[ ]:

```
array([[15.53831908],
       [ 8.69250731],
       [12.33347255],
       [ 8.94053076],
       [ 9.85983137],
       [13.11162417],
       [ 8.5778736 ],
       [12.08984937],
       [10.91753115],
       [10.64247071]])
```

In [ ]:

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

Out[ ]:

```
0.10551972009791755
```