IBM Assignment 3

October 11, 2022

[1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
import seaborn as sns
import math
```

Load dataset

[2]:
```
df = pd.read_csv("abalone.csv")
df = pd.DataFrame(df)
df.head()
```

[2]:
```
  Sex  Length  Diameter  Height  Whole weight  Shucked weight  Viscera weight  \
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

   Shell weight  Rings
0         0.150     15
1         0.070      7
2         0.210      9
3         0.155     10
4         0.055      7
```

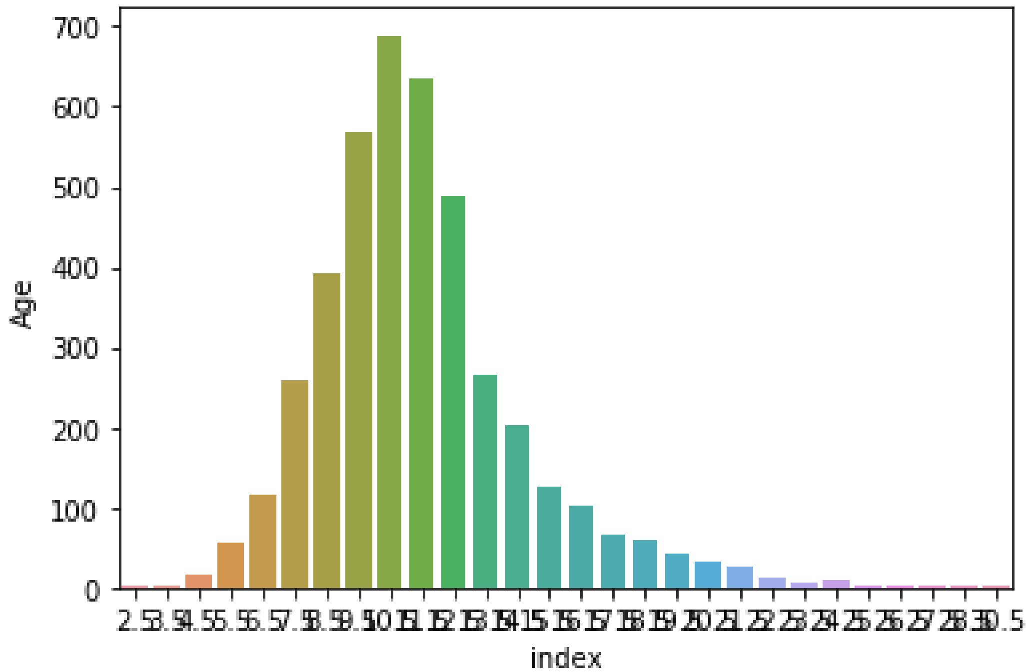
[3]:
```
# created age column
df["Age"] = df["Rings"]+1.5
# dropped rings since unnecessary
df = df.drop("Rings", axis=1)
df
```

```
[3]:       Sex  Length  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   Age
     0             0.1010        0.1500  16.5
     1             0.0485        0.0700   8.5
     2             0.1415        0.2100  10.5
     3             0.1140        0.1550  11.5
     4             0.0395        0.0550   8.5
     …               …            …     …
     4172          0.2390        0.2490  12.5
     4173          0.2145        0.2605  11.5
     4174          0.2875        0.3080  10.5
     4175          0.2610        0.2960  11.5
     4176          0.3765        0.4950  13.5

     [4177 rows x 9 columns]
```

Univariate analysis

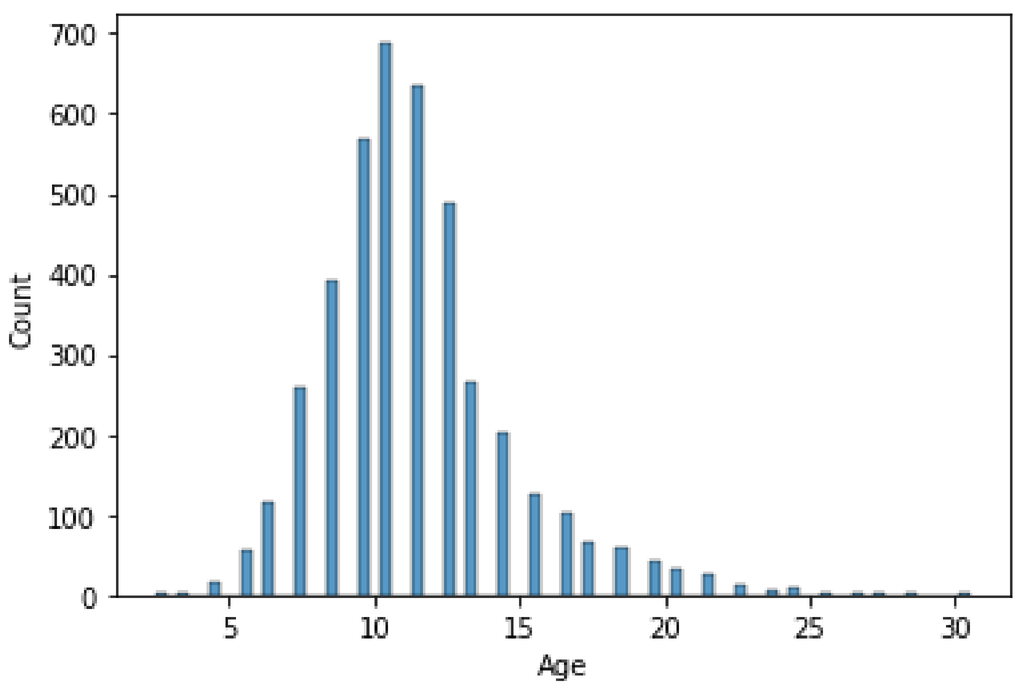
```
[4]: age = df['Age'].value_counts().reset_index()
     # barplot
     sns.barplot(data=age, x='index', y='Age')
```

```
[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffaa3025450>
```



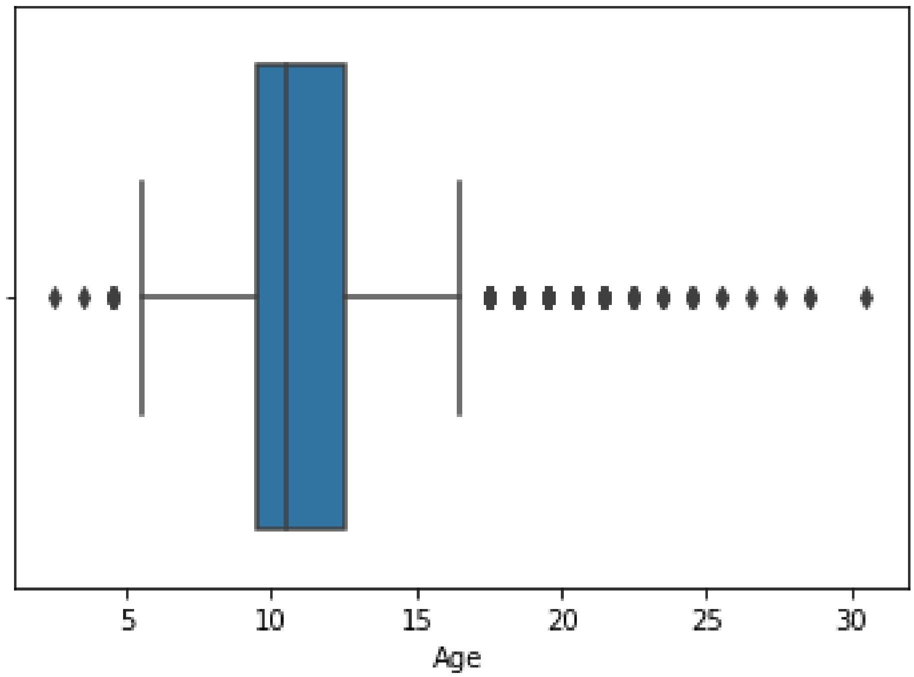

```
[5]: #histplot
     sns.histplot(x=df.Age)
```

[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffaa2ec7150>



[6]:
```
# boxplot
sns.boxplot(x=df.Age)
```

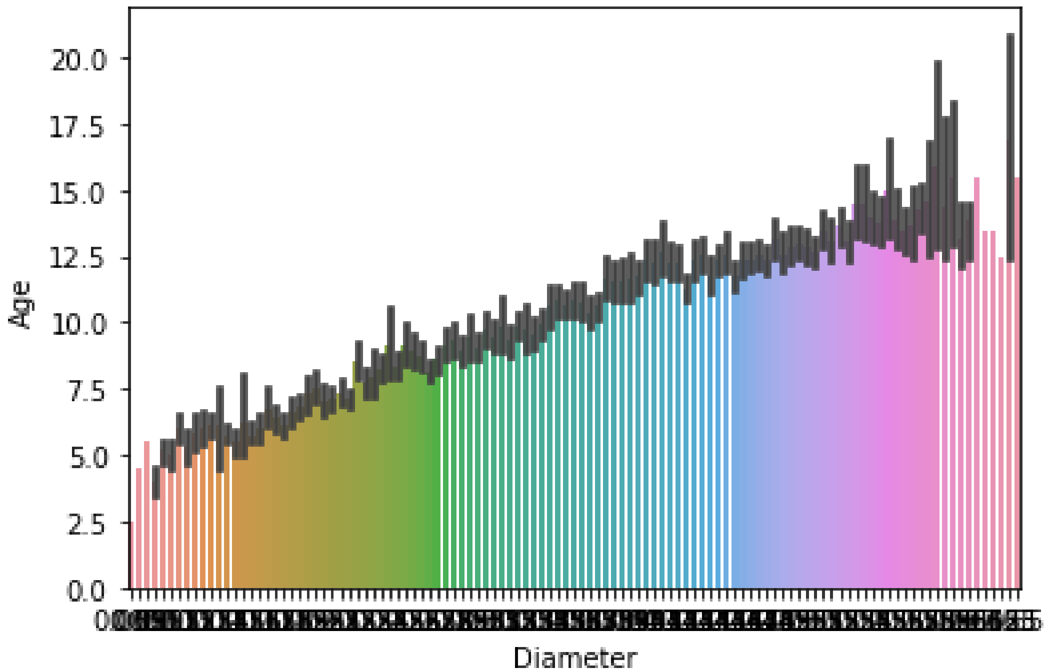
[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffaa29fd110>



Bivariate analysis

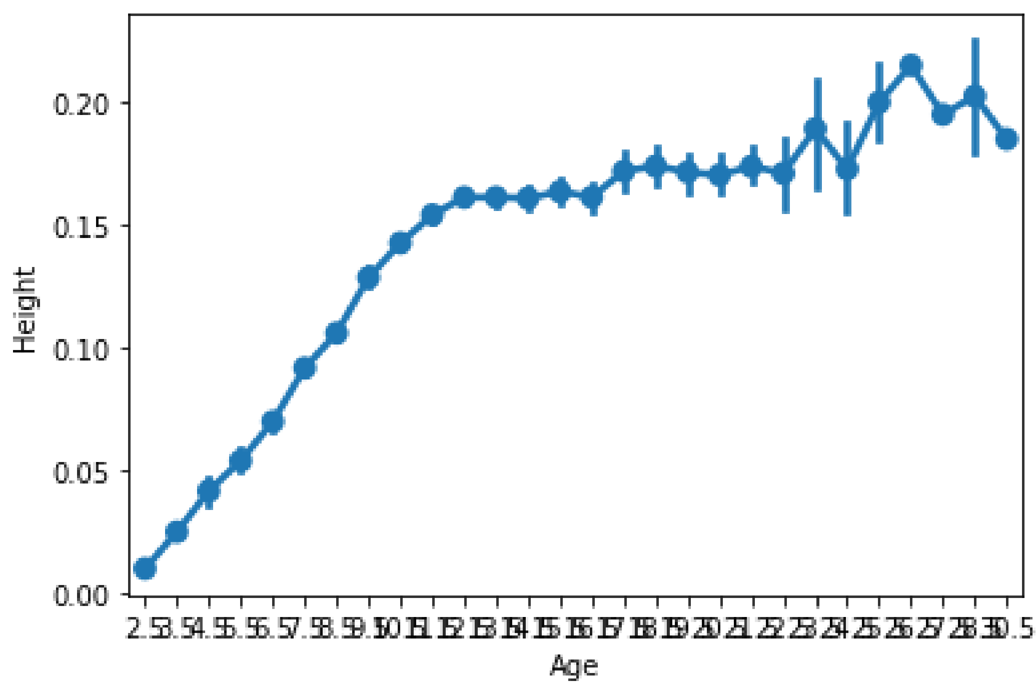
[7]:
```
#barplot
sns.barplot(x=df['Diameter'], y=df.Age)
```

[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffaa2806b10>



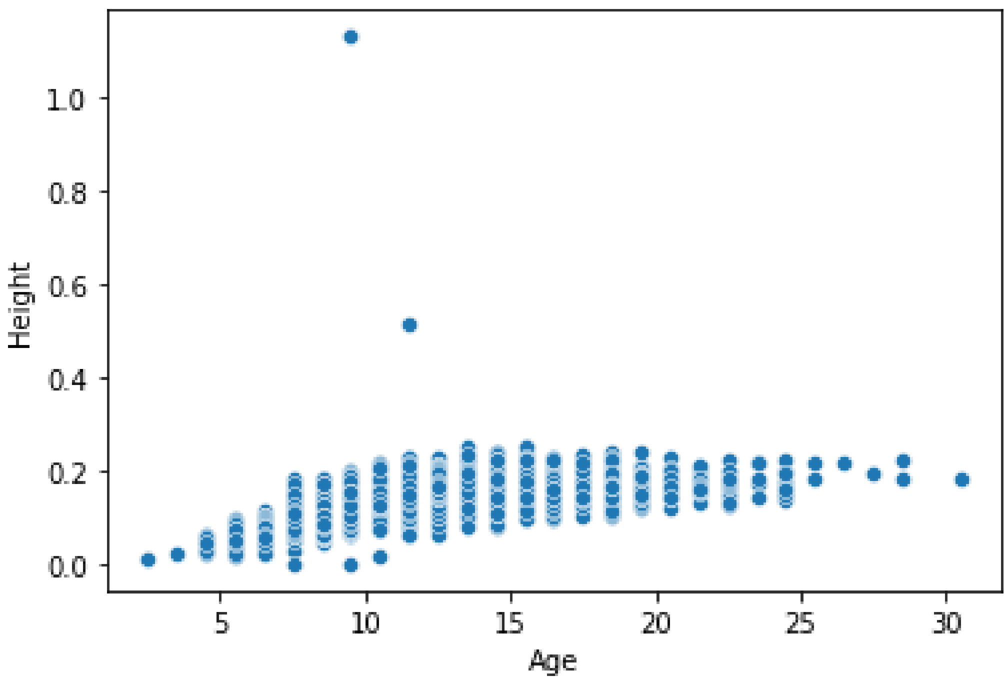
[8]:
```
#pointplot
sns.pointplot(x=df.Age, y=df.Height)
```

[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffaa289c3d0>
```



[9]:
```
#scatter plot
sns.scatterplot(data=df,x="Age",y="Height")
```

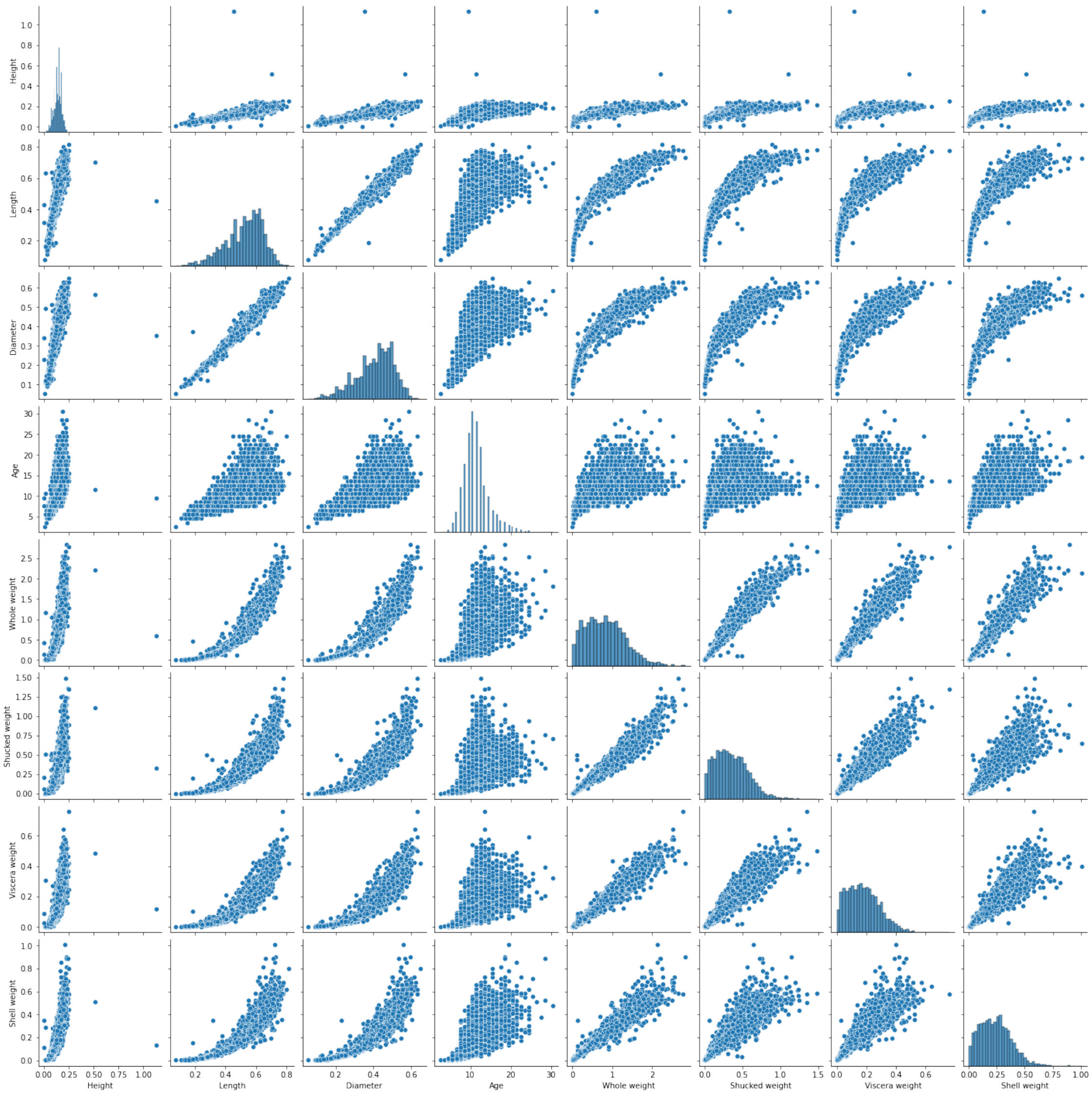
[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffaa29b3f10>



Multivariate analysis

[10]:
```
#pairplot
sns.pairplot(data = df[["Height","Length","Diameter","Age","Whole␣
↪weight","Shucked weight","Viscera weight","Shell weight"]])
```

[10]: <seaborn.axisgrid.PairGrid at 0x7ffaa227b150>



Descriptive statistics

[11]:
```
df.describe()
```

[11]:

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

```
       Viscera weight  Shell weight          Age
count     4177.000000   4177.000000  4177.000000
mean         0.180594      0.238831    11.433684
std          0.109614      0.139203     3.224169
min          0.000500      0.001500     2.500000
25%          0.093500      0.130000     9.500000
50%          0.171000      0.234000    10.500000
75%          0.253000      0.329000    12.500000
max          0.760000      1.005000    30.500000
```

Missing values and how to deal with them

[12]:
```
df.isnull().sum()
```

[12]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Age               0
dtype: int64
```

[13]:
```
df.isna().sum()
# no missing values
```

[13]:
```
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

[14]:
```
# replacing numerical outliers with lower and upper limits respectively

for i in df:
  if df[i].dtype=='int64'or df[i].dtypes=='float64':
    q1=df[i].quantile(0.25)
    q3=df[i].quantile(0.75)
```

```
    iqr=q3-q1
    upper=q3+1.5*iqr
    lower=q1-1.5*iqr
    df[i]=np.where(df[i] >upper, upper, df[i])
    df[i]=np.where(df[i] <lower, lower, df[i])
```

Check for categorical columns and perform encoding

[15]:
```
# identified and encoded the categorical values
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
for i in df:
  if df[i].dtype=='object' or df[i].dtype=='category':
    print(i)
    df[i]=encoder.fit_transform(df[i])
df.head()
```

```
Sex
```

[15]:
```
   Sex  Length  Diameter  Height  Whole weight  Shucked weight  \
0    2   0.455     0.365   0.095        0.5140          0.2245
1    2   0.350     0.265   0.090        0.2255          0.0995
2    0   0.530     0.420   0.135        0.6770          0.2565
3    2   0.440     0.365   0.125        0.5160          0.2155
4    1   0.330     0.255   0.080        0.2050          0.0895

   Viscera weight  Shell weight   Age
0          0.1010         0.150  16.5
1          0.0485         0.070   8.5
2          0.1415         0.210  10.5
3          0.1140         0.155  11.5
4          0.0395         0.055   8.5
```

Split the data into dependent and independent variables.

[16]:
```
# independent variables
X = df.iloc[:, :-1].values
X
```

[16]:
```
array([[2.    , 0.455 , 0.365 , …, 0.2245, 0.101 , 0.15  ],
       [2.    , 0.35  , 0.265 , …, 0.0995, 0.0485, 0.07  ],
       [0.    , 0.53  , 0.42  , …, 0.2565, 0.1415, 0.21  ],
       …,
       [2.    , 0.6   , 0.475 , …, 0.5255, 0.2875, 0.308 ],
       [0.    , 0.625 , 0.485 , …, 0.531 , 0.261 , 0.296 ],
       [2.    , 0.71  , 0.555 , …, 0.9455, 0.3765, 0.495 ]])
```

[17]:
```python
# dependent variables
Y = df.iloc[:, -1].values
Y
df.head()
```

[17]:
```
   Sex  Length  Diameter  Height  Whole weight  Shucked weight  \
0    2   0.455     0.365   0.095        0.5140          0.2245
1    2   0.350     0.265   0.090        0.2255          0.0995
2    0   0.530     0.420   0.135        0.6770          0.2565
3    2   0.440     0.365   0.125        0.5160          0.2155
4    1   0.330     0.255   0.080        0.2050          0.0895

   Viscera weight  Shell weight   Age
0          0.1010         0.150  16.5
1          0.0485         0.070   8.5
2          0.1415         0.210  10.5
3          0.1140         0.155  11.5
4          0.0395         0.055   8.5
```

Scale independent variables

[18]:
```python
x = scale(df[["Viscera weight", "Length", "Diameter", "Height", "Whole weight",␣
 ↪"Shucked weight", "Shell weight"]])
x
```

[18]:
```
array([[-0.73030425, -0.58311728, -0.44088378, …, -0.6447403 ,
        -0.61498531, -0.64518445],
       [-1.21388983, -1.46569411, -1.45976205, …, -1.23820752,
        -1.1916374 , -1.23138964],
       [-0.35725252,  0.04729474,  0.11949927, …, -0.30943646,
        -0.46736237, -0.20553056],
       …,
       [ 0.98757595,  0.63567929,  0.67988232, …,  0.71704585,
         0.77359293,  0.51257079],
       [ 0.74348037,  0.84581663,  0.78177015, …,  0.54939393,
         0.79896563,  0.42464001],
       [ 1.80736865,  1.56028358,  1.49498494, …,  2.30613921,
         2.71114397,  1.88282541]])
```

Split the data into training and testing

[19]:
```python
X = df.iloc[:, 1:7]
X
```

[19]:
```
      Length  Diameter  Height  Whole weight  Shucked weight  Viscera weight
0      0.455     0.365   0.095        0.5140          0.2245          0.1010
1      0.350     0.265   0.090        0.2255          0.0995          0.0485
2      0.530     0.420   0.135        0.6770          0.2565          0.1415
```

```
3     0.440   0.365   0.125        0.5160         0.2155         0.1140
4     0.330   0.255   0.080        0.2050         0.0895         0.0395
…     …     …    …            …            …             …
4172  0.565   0.450   0.165        0.8870         0.3700         0.2390
4173  0.590   0.440   0.135        0.9660         0.4390         0.2145
4174  0.600   0.475   0.205        1.1760         0.5255         0.2875
4175  0.625   0.485   0.150        1.0945         0.5310         0.2610
4176  0.710   0.555   0.195        1.9485         0.9455         0.3765

[4177 rows x 6 columns]
```

[20]:
```
Y = df.iloc[:, -1]
Y
```

[20]:
```
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
Name: Age, Length: 4177, dtype: float64
```

[21]:
```
# splitting to train and test data
x_train,x_test,y_train,y_test = train_test_split(X, Y, test_size=0.
 ↪25,random_state =42)
```

Build the Model

[22]:
```
# linear regression model
model = LinearRegression()
```

Train the model

[23]:
```
model.fit(x_train,y_train)
```

[23]:
```
LinearRegression()
```

Test the model

[24]:
```
y_predict = model.predict(x_test)
y_predict
```

[24]:
```
array([12.80459703, 11.53324712, 15.38929967, …, 13.49727819,
       11.82966664, 10.72212477])
```

[25]:
```python
print("Mean Squared Error: ",mean_squared_error(y_test, y_predict))
print("Root Mean Squared Error: ",math.sqrt(mean_squared_error(y_test,␣
 ↪y_predict)))
```

```
Mean Squared Error:  3.4158644158621536
Root Mean Squared Error:  1.848205728771057
```