# Abalone Age Prediction

**Assignment -3**

| | |
|---|---|
| Assignment Date | 03 October 2022 |
| Team ID | PNT2022TMID27812 |
| Project Name | Smart Lender-Application Credibility Prediction for loan Approval |
| Student Name | S.G.Mydhrayan |
| Student Roll Number | 311519104036 |
| Maximum Marks | 2 Marks |

**Question-1.**Download dataset

**Solution:**

| 1 | Diameter | Height | Whole weight | Shucked weight | Viscera weight | Shell weight | Rings |
|---|---|---|---|---|---|---|---|
| 2 | 0.365 | 0.095 | 0.514 | 0.2245 | 0.101 | 0.15 | 15 |
| 3 | 0.265 | 0.09 | 0.2255 | 0.0995 | 0.0485 | 0.07 | 7 |
| 4 | 0.42 | 0.135 | 0.677 | 0.2565 | 0.1415 | 0.21 | 9 |
| 5 | 0.365 | 0.125 | 0.516 | 0.2155 | 0.114 | 0.155 | 10 |
| 6 | 0.255 | 0.08 | 0.205 | 0.0895 | 0.0395 | 0.055 | 7 |
| 7 | 0.3 | 0.095 | 0.3515 | 0.141 | 0.0775 | 0.12 | 8 |
| 8 | 0.415 | 0.15 | 0.7775 | 0.237 | 0.1415 | 0.33 | 20 |
| 9 | 0.425 | 0.125 | 0.768 | 0.294 | 0.1495 | 0.26 | 16 |
| 10 | 0.37 | 0.125 | 0.5095 | 0.2165 | 0.1125 | 0.165 | 9 |
| 11 | 0.44 | 0.15 | 0.8945 | 0.3145 | 0.151 | 0.32 | 19 |
| 12 | 0.38 | 0.14 | 0.6065 | 0.194 | 0.1475 | 0.21 | 14 |
| 13 | 0.35 | 0.11 | 0.406 | 0.1675 | 0.081 | 0.135 | 10 |
| 14 | 0.38 | 0.135 | 0.5415 | 0.2175 | 0.095 | 0.19 | 11 |
| 15 | 0.405 | 0.145 | 0.6845 | 0.2725 | 0.171 | 0.205 | 10 |
| 16 | 0.355 | 0.1 | 0.4755 | 0.1675 | 0.0805 | 0.185 | 10 |
| 17 | 0.4 | 0.13 | 0.6645 | 0.258 | 0.133 | 0.24 | 12 |
| 18 | 0.28 | 0.085 | 0.2905 | 0.095 | 0.0395 | 0.115 | 7 |
| 19 | 0.34 | 0.1 | 0.451 | 0.188 | 0.087 | 0.13 | 10 |
| 20 | 0.295 | 0.08 | 0.2555 | 0.097 | 0.043 | 0.1 | 7 |
| 21 | 0.32 | 0.1 | 0.381 | 0.1705 | 0.075 | 0.115 | 9 |
| 22 | 0.28 | 0.095 | 0.2455 | 0.0955 | 0.062 | 0.075 | 11 |
| 23 | 0.275 | 0.1 | 0.2255 | 0.08 | 0.049 | 0.085 | 10 |
| 24 | 0.44 | 0.155 | 0.9395 | 0.4275 | 0.214 | 0.27 | 12 |
| 25 | 0.415 | 0.135 | 0.7635 | 0.318 | 0.21 | 0.2 | 9 |

**Question-2.**Load the dataset

**Solution:**

**import numpy as np**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**import sklearn**

**data = pd.read_csv(r"abalone.csv")**

**data.head**

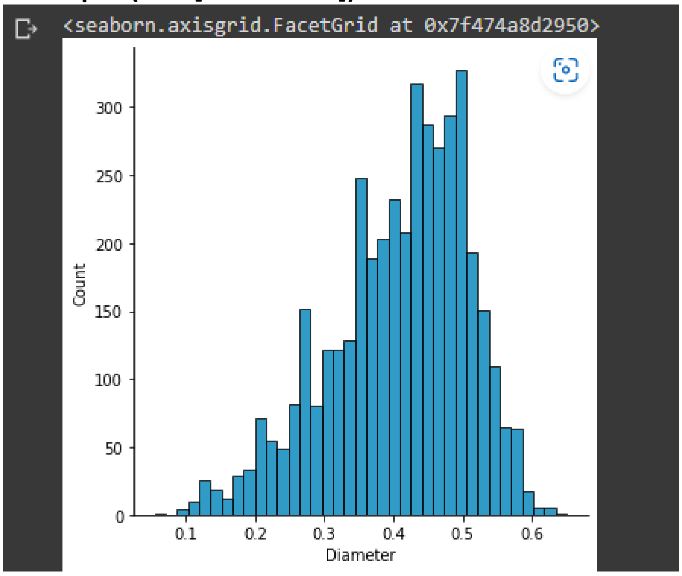
```
<bound method NDFrame.head of      Sex  Length  Diameter  Height  Whole weight  Shucked weight
0      M   0.455     0.365   0.095        0.5140          0.2245
1      M   0.350     0.265   0.090        0.2255          0.0995
2      F   0.530     0.420   0.135        0.6770          0.2565
3      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  Rings
0             0.1010        0.1500     15
1             0.0485        0.0700      7
2             0.1415        0.2100      9
3             0.1140        0.1550     10
4             0.0395        0.0550      7
...              ...           ...    ...
4172          0.2390        0.2490     11
4173          0.2145        0.2605     10
4174          0.2875        0.3080      9
4175          0.2610        0.2960     10
4176          0.3765        0.4950     12
```

**Question-3.**Perform Below Visualizations.

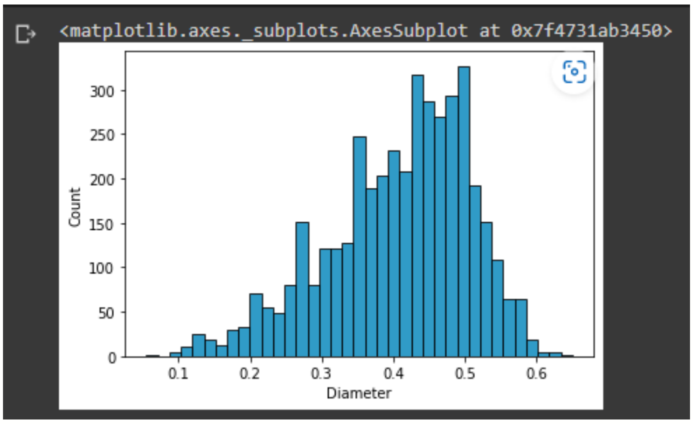
***3.1 Univariate Analysis***

**Solution:**

**sns.displot(data[' Diameter'])**

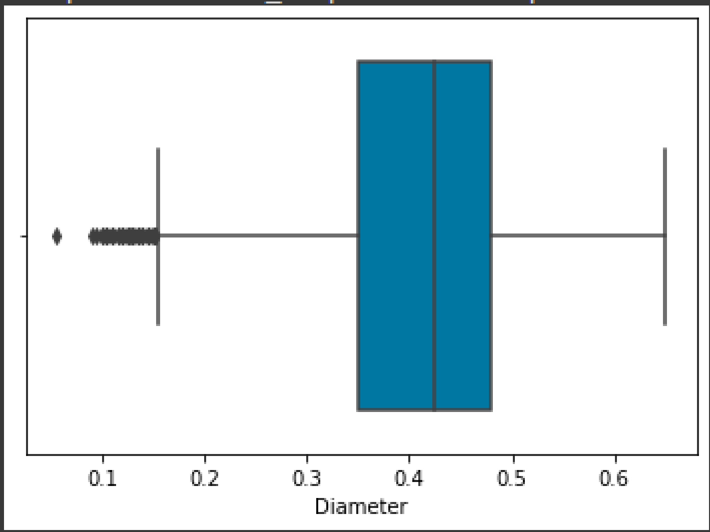



**sns.histplot(data[' Diameter '])**



**sns.boxplot(x = data[' Diameter '])**

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4731587050>
```

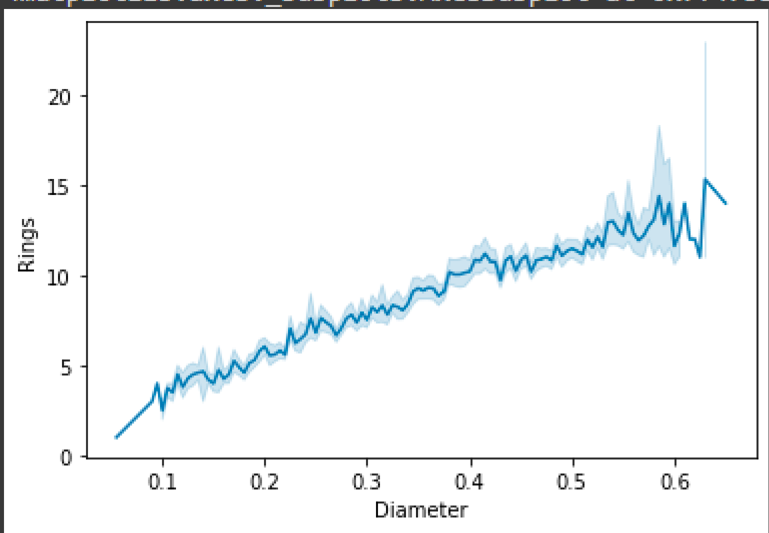



***3.2 Bivariate Analysis***

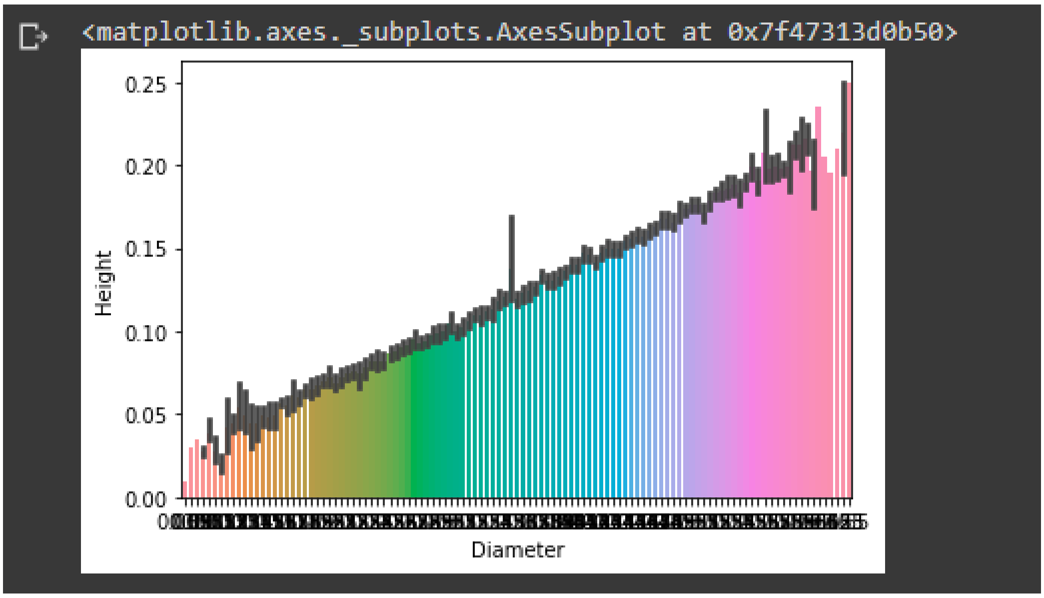
**Solution:**

**sns.lineplot(data = data, x = 'Diameter', y = 'Rings')**

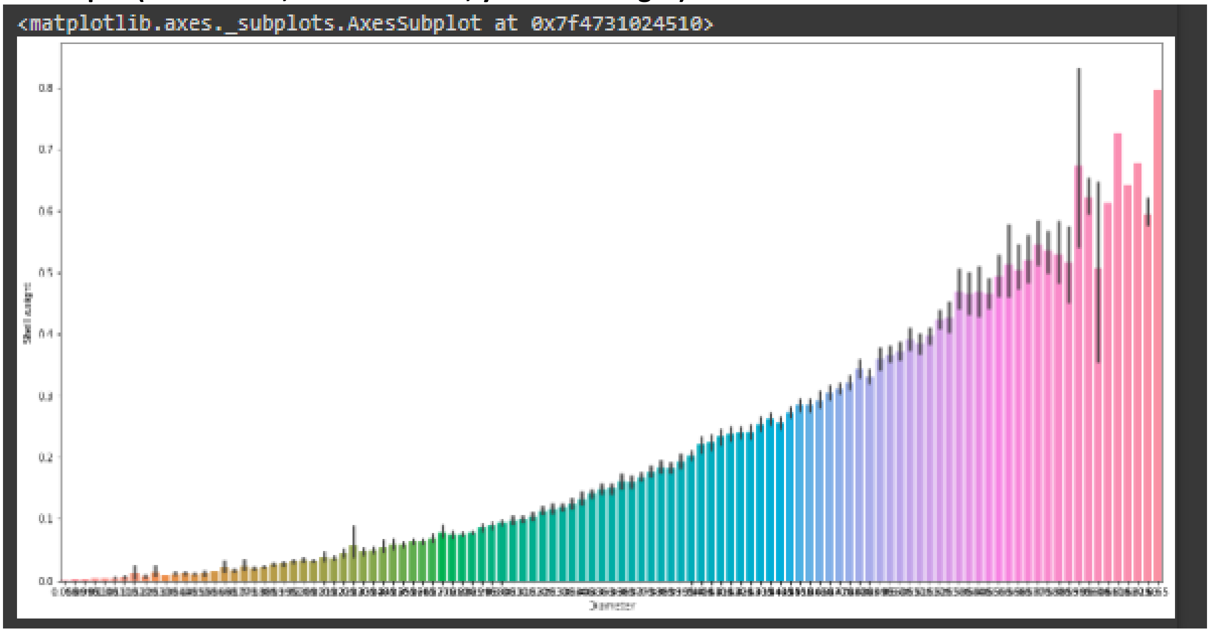
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f47314ec350>
```



**sns.barplot(data = data, x = 'Diameter',y = 'Height')**




**plt.figure(figsize=(20,20))**
**sns.barplot(data = data, x = 'Diameter', y = 'Shell weight)**

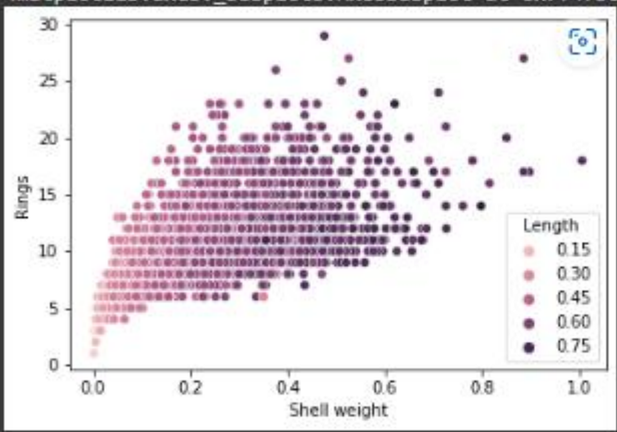



### *3.3 Multivariate Analysis*

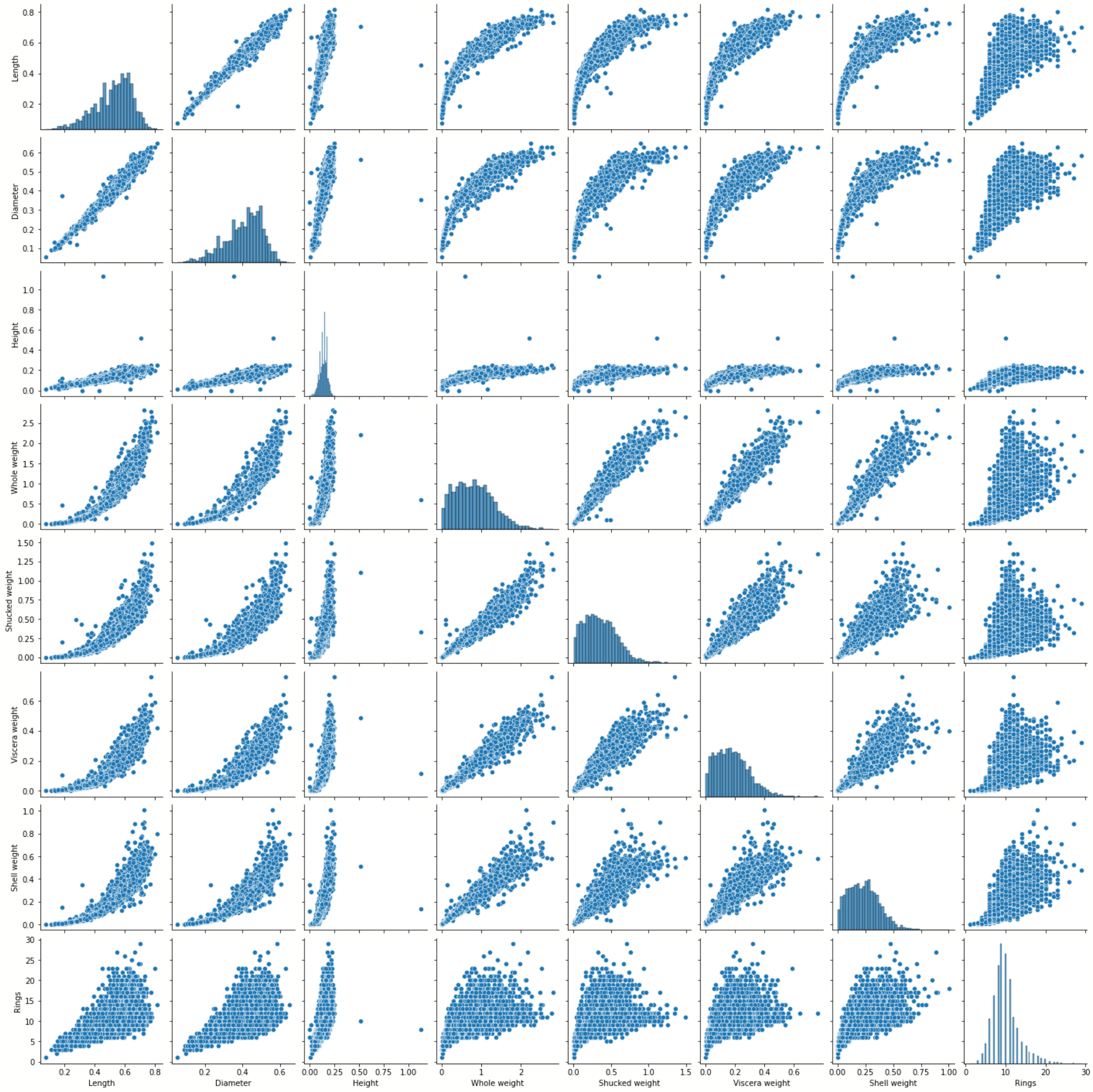
**Solution:**

**sns.scatterplot(data = data x = 'Shell weight', y = 'Rings', hue = 'Length')**

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4730b25210>
```



**sns.pairplot(data)**



**Question-4.**Perform descriptive statistics on the dataset.

**Solution:**

**data.mean(numeric_only = True)**

```
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
```

**data.median(numeric_only = True)**

```
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
dtype: float64
```

**data['Whole weight'].mode()**

```
0 0.2225 dtype: float64
```

**data['Length'].mode()**

```
0 0.550 1 0.625 dtype: float64
```

**data['Rings'].unique()**

```
array([15, 7, 9, 10, 8, 20, 16, 19, 14, 11, 12, 18, 13, 5, 4, 6, 21,
17, 22, 1, 3, 26, 23, 29, 2, 27, 25, 24])
```

**data.std(numeric_only=True)**

```
Length            0.120093
Diameter          0.099240
Height            0.041827
Whole weight      0.490389
Shucked weight    0.221963
Viscera weight    0.109614
Shell weight      0.139203
Rings             3.224169
dtype: float64
```

**data.describe()**

| | 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 |

**data['Whole weight'].value_counts()**

```
0.2225    8
1.1345    7
0.9700    7
0.4775    7
0.1960    7
         ..
0.0475    1
1.8930    1
1.8725    1
2.1055    1
1.9485    1
Name: Whole weight, Length: 2429, dtype: int64
```

**Question-5.**Handle the Missing values.

**Solution:**

**data.isnull().any()**

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

**data.isnull().sum()**

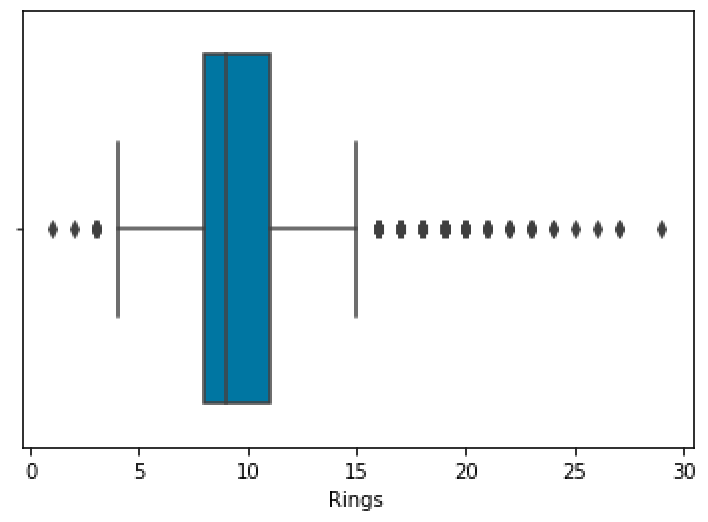
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

**Question-6.**Find the outliers and replace the outliers

**Solution:**

**sns.boxplot(x = data['Rings'])**

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f472d3e1090>
```




**q = data.quantile([0.75,0.25])**
**q**

| | 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 |

**iqr =  q.iloc[0] - q.iloc[1]**
**iqr**

```
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
```

**u = q.iloc[0] + (1.5*iqr)**
**u**

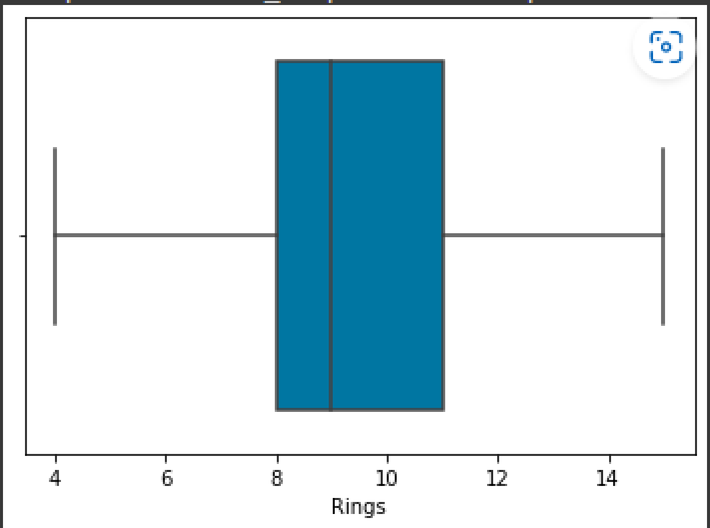
```
Length            0.86250
Diameter          0.67500
Height            0.24000
Whole weight      2.22025
Shucked weight    0.97600
Viscera weight    0.49225
Shell weight      0.62750
Rings            15.50000
dtype: float64
```

**l = q.iloc[1] - (1.5*iqr)**
**l**

```
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
```

**data['Rings'] = np.where(np.logical_or(data['Rings']>15, data['Rings']<4), 9, data['Rings'])**
**sns.boxplot(x = data['Rings'])**

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f472d26c310>
```



**Question-7.** Check for Categorical columns and perform encoding

**Solution:**

**data['Age'] = data['Rings'] + 1.5**
**data['Sex'].value_counts()**

```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
```

**from sklearn.preprocessing import LabelEncoder, OneHotEncoder**
**le = LabelEncoder()**
**data = pd.get_dummies(data)**
**data.head()**

| | Length | Diameter | Height | Whole weight | Shucked weight | Viscera weight | Shell weight | Rings | Age | Sex_F | Sex_I | Sex_M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 15 | 16.5 | 0 | 0 | 1 |
| 1 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 7 | 8.5 | 0 | 0 | 1 |
| 2 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 9 | 10.5 | 1 | 0 | 0 |
| 3 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 10 | 11.5 | 0 | 0 | 1 |
| 4 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 7 | 8.5 | 0 | 1 | 0 |

**Question-8.**Split the data into dependent and independent variables split the data in X and Y

**Solution:**

**y = data['Age']**
**data = data.drop(['Rings','Age'], axis=1)**
**x = data**
**data.head**

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

y

```
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
```

**Question-9.**Scale the independent variables

**Solution:**

**from sklearn.preprocessing import StandardScaler, MinMaxScaler**
**sc = StandardScaler()**
**x_scaled = sc.fit_transform(x)**
**x_scaled**

```
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]])
```

**Question-10.**Split data into Training and Testing

**Solution:**

**from sklearn.model_selection import train_test_split**
**x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size = 0.3, random_state = 0)**
**x_train**

```
array([[ 0.79954256,  1.0291325 ,  0.84844242, ..., -0.67483383,
        -0.68801788,  1.31667716],
       [-1.49062526, -1.54070702, -1.30353247, ..., -0.67483383,
         1.45345059, -0.75948762],
       [-1.24078877, -1.33915098, -1.06442415, ..., -0.67483383,
         1.45345059, -0.75948762],
       ...,
       [ 0.59134549,  0.42446438,  0.13111745, ..., -0.67483383,
        -0.68801788,  1.31667716],
       [ 0.84118198,  0.82757646,  0.6093341 , ...,  1.48184628,
        -0.68801788, -0.75948762],
       [-0.94931287, -0.83526087, -0.70576167, ..., -0.67483383,
         1.45345059, -0.75948762]])
```

**x_train.shape**

```
(2923, 10)
```

**x_test**

```
array([[ 0.21659075,  0.17251933,  0.37022577, ..., -0.67483383,
        -0.68801788,  1.31667716],
       [-0.1998034 , -0.07942572, -0.46665335, ..., -0.67483383,
         1.45345059, -0.75948762],
       [ 0.79954256,  0.72679844,  0.37022577, ..., -0.67483383,
        -0.68801788,  1.31667716],
       ...,
       [ 0.92446081,  0.87796547, -2.97729071, ...,  1.48184628,
        -0.68801788, -0.75948762],
       [ 1.13265788,  0.97874349,  1.44621322, ..., -0.67483383,
        -0.68801788,  1.31667716],
       [ 0.79954256,  0.77718745,  0.72888826, ..., -0.67483383,
         1.45345059, -0.75948762]])
```

**x_test.shape**

```
(1254, 10)
```

**y_train**

```
1376    11.5
1225     6.5
2722     8.5
3387    10.5
2773    12.5
        ...
1033    11.5
3264    13.5
1653    11.5
2607    10.5
2732     9.5
Name: Age, Length: 2923, dtype: float64
```

**y_test**

```
668     14.5
1580     9.5
3784    12.5
463      6.5
2615    13.5
        ...
1052    13.5
3439     9.5
1174    10.5
2210    10.5
2408    16.5
Name: Age, Length: 1254, dtype: float64
```

**Question-11.** Build the model

**Solution:**

**from sklearn.linear_model import LinearRegression**
**lr = LinearRegression()**
**lr.fit(x_train,y_train)**

```
LinearRegression()
```

**predict = lr.predict(x_test)**
**predict**

```
array([12.32892845, 10.26905996, 11.99728864, ..., 11.36906433,
       13.83936664, 11.45100404])
```

**Question-12.** Train the model

**Solution:**

**y_train**

```
1376    11.5
1225     6.5
2722     8.5
3387    10.5
2773    12.5
        ...
1033    11.5
3264    13.5
1653    11.5
2607    10.5
2732     9.5
Name: Age, Length: 2923, dtype: float64
```

**Question-13.** Test the model

**Solution:**

**y_test**

```
668     14.5
1580     9.5
3784    12.5
463      6.5
2615    13.5
        ...
1052    13.5
3439     9.5
1174    10.5
2210    10.5
2408    16.5
Name: Age, Length: 1254, dtype: float64
```

**Question-14.** Measure the performance using Metrics

**Solution:**

```
from sklearn.metrics import r2_score, mean_squared_error
mse = mean_squared_error(y_test, predict)
rmse = np.sqrt(mse)
print("mse = ", mse)
print("rmse = ", rmse)
r2_score(y_test,predict)
```

```
mse =  2.9684942599827626
rmse =  1.7229318790894672
0.4607933491755676
```