Assignment-3

Assignment Date	06 October 2022
Student Name	Vaishnavi R
Student Roll Number	820419106068
Maximum Marks	2 Marks

1.Download the Dataset

2.Load the dataset into the tool

In [2]:
```
import pandas as pd
data = pd.read_csv("abalone.csv")
data.head()
```

Out[2]:

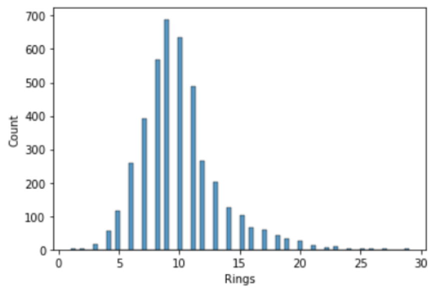
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

3.Perform Below Visualizations

Univariate Analysis

In [3]:
```
import seaborn as sns
sns.histplot(data, x="Rings")
```

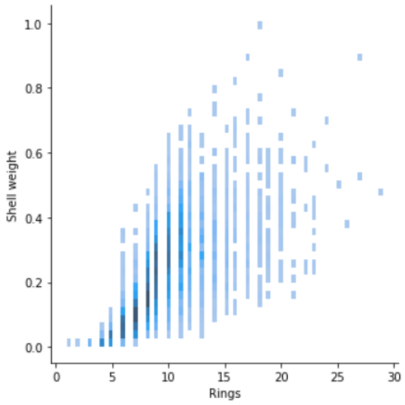
Out[3]: <AxesSubplot:xlabel='Rings', ylabel='Count'>



In [ ]:
```
Bi-variate Analysis
```

In [4]:
```
sns.displot(data, x="Rings" , y="Shell weight")
```

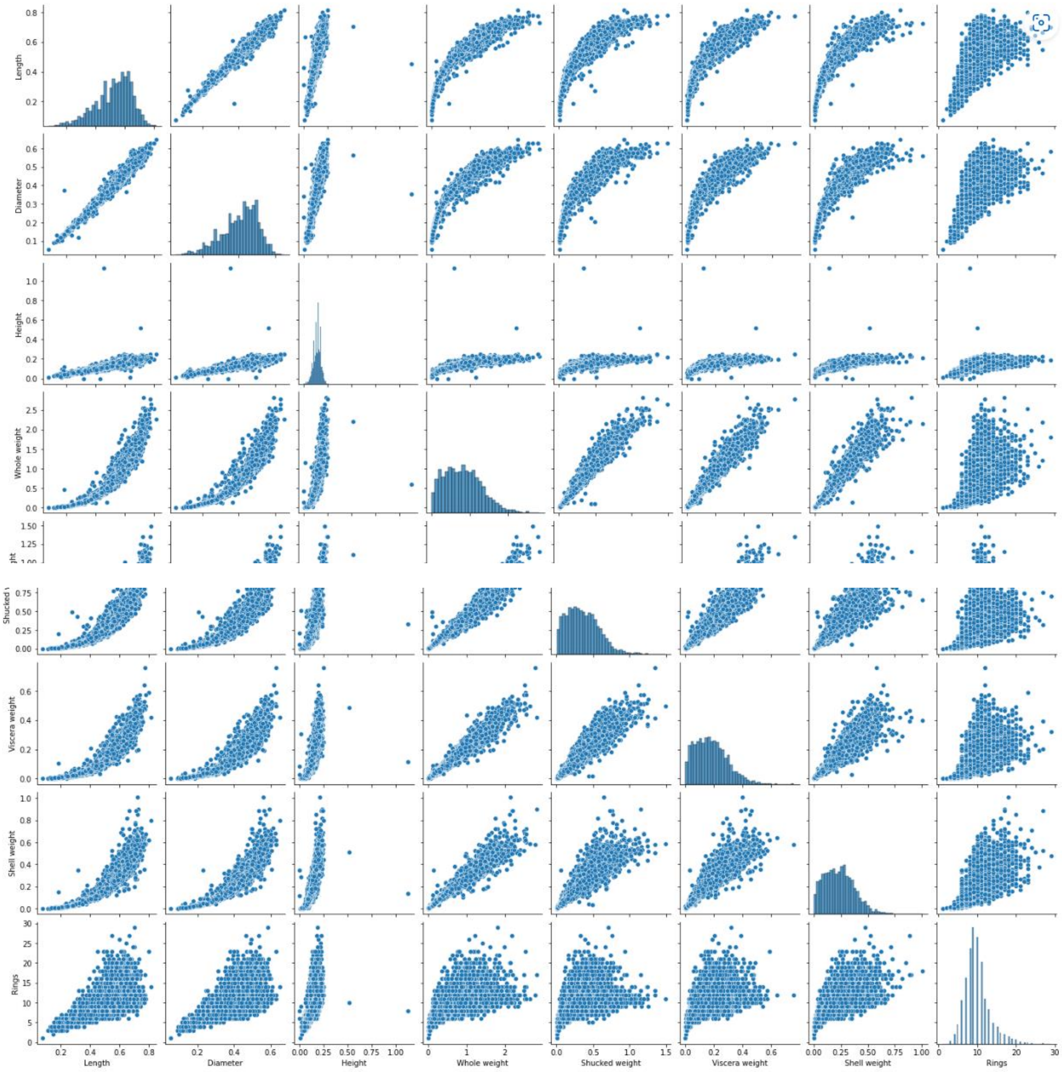
Out[4]: <seaborn.axisgrid.FacetGrid at 0x1bd656a1130>



In [ ]:
```
Multi-variate Analysis
```

In [5]:
```
sns.pairplot(data)
```

Out[5]: <seaborn.axisgrid.PairGrid at 0x1bd68d0aca0>



4.Perform Descriptive Statistics on the Dataset

In [6]:
```
data.mean()
```

```
C:\Users\Raju\AppData\Local\Temp\ipykernel_6340\531903386.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reduct
ions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before
calling the reduction.
  data.mean()
```

Out[6]:
```
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
```

In [7]:
```
data.median()
```

```
C:\Users\Raju\AppData\Local\Temp\ipykernel_6340\4184645713.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reduc
tions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns befor
e calling the reduction.
  data.median()
```

Out[7]:
```
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
```

In [8]:
```
data.mode()
```

Out[8]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.550	0.45	0.15	0.2225	0.175	0.1715	0.275	9.0
1	NaN	0.625	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [9]:
```
data.skew()
```

```
C:\Users\Raju\AppData\Local\Temp\ipykernel_6340\1188251951.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reduc
tions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns befor
e calling the reduction.
  data.skew()
```

Out[9]:
```
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 [10]:
```
data.kurt()
```

```
C:\Users\Raju\AppData\Local\Temp\ipykernel_6340\2907027414.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reduc
tions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns befor
e calling the reduction.
  data.kurt()
```

Out[10]:
```
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 [11]:
```
data.std()
```

```
C:\Users\Raju\AppData\Local\Temp\ipykernel_6340\2723740006.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reduc
tions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns befor
e calling the reduction.
  data.std()
```

Out[11]:
```
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
```

In [13]:
```
data.var()
```

```
C:\Users\Raju\AppData\Local\Temp\ipykernel_6340\445316826.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reduct
ions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before
calling the reduction.
  data.var()
```

Out[13]:
```
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
```

5.Check for Missing values and deal with them

In [14]:
```
data.isna().sum()
```

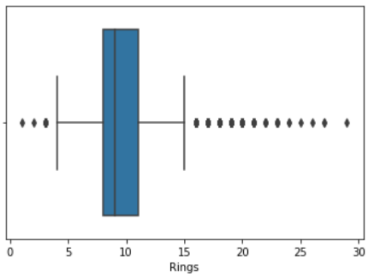
Out[14]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

6.Find the outliers and replace them with outliers

In [15]:
```
sns.boxplot(data['Rings'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  warnings.warn(
```

Out[15]: <AxesSubplot:xlabel='Rings'>

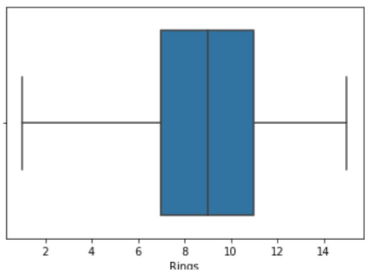


In [16]:
```
import numpy as np
data['Rings']=np.where(data['Rings']>15,5,data['Rings']) #replacing
```

In [17]:
```
sns.boxplot(data['Rings'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  warnings.warn(
```

Out[17]: <AxesSubplot:xlabel='Rings'>



7.Check for categorical columns and perform encoding

In [18]:
```
data.tail()
```

Out[18]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
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

In [19]:
```
from sklearn.preprocessing import LabelEncoder
```

In [20]:
```
le=LabelEncoder()
```

In [21]:
```
data['Sex']=le.fit_transform(data['Sex'])
```

In [22]:
```
data.head()
```

Out[22]:

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

8.Split the data into Dependent and Independent variables

In [23]:
```
y=data['Rings']
y.head()
```

Out[23]:
```
0    15
1     7
2     9
3    10
4     7
Name: Rings, dtype: int64
```

In [24]:
```
x=data.drop(columns=['Rings'])
x.head()
```

Out[24]:

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

9.Scale theIndependent variables

In [25]:
```
from sklearn.preprocessing import scale
x=scale(x)
x
```

Out[25]:
```
array([[ 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]])
```

10.Split the data into Training and Testing

In [26]:
```
from sklearn.model_selection import train_test_split
```

In [27]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [28]:
```
print(x_train.shape,y_train.shape)
```

```
(3341, 8) (3341,)
```

In [29]:
```
print(x_test.shape,y_test.shape)
```

```
(836, 8) (836,)
```

11.Build the Model

In [32]:
```
from sklearn.tree import DecisionTreeRegressor
model=DecisionTreeRegressor()
```

12.Train the Model

In [33]:
```
model.fit(x_train,y_train)
```

Out[33]:
```
DecisionTreeRegressor()
```

13.Test the Model

In [35]:
```
pred2=model.predict(x_test)
pred2
```

Out[35]:
```
array([13.,  5.,  9., 10., 14.,  9.,  8.,  9.,  4.,  8., 13.,  9., 13.,
        6.,  7., 12., 10., 11., 11.,  9.,  5.,  9., 10.,  9.,  9.,  6.,
        8.,  6.,  9.,  5.,  8., 14.,  6., 12.,  8., 11.,  6.,  1.,  7.,
        7., 10.,  5., 12.,  9., 13., 10.,  8.,  8., 14.,  5.,  6.,  5.,
        9.,  3.,  5., 12.,  9.,  5., 12., 10., 11.,  5., 11.,  8.,  8.,
       11.,  4.,  6.,  8.,  9., 14., 10.,  8.,  8.,  9.,  9.,  5., 11.,
        9.,  8., 14., 14., 14.,  8.,  6., 13.,  9.,  9., 11., 12.,  5.,
       14., 10., 13., 11.,  9.,  8., 10.,  7.,  6.,  5., 10., 11., 11.,
        9.,  8.,  8., 10., 10.,  8., 10.,  9., 10., 10., 10.,  5.,  8.,
        7.,  9., 15., 11., 10.,  8., 11., 11., 13., 11., 10., 10.,  5.,
       10.,  8.,  9.,  4.,  4.,  9.,  9., 10.,  6., 11.,  9., 11.,  8.,
       11., 10., 13.,  5.,  5.,  8., 11.,  9.,  9.,  7., 15.,  8., 11.,
        9.,  5., 10.,  7., 10.,  6.,  8., 13.,  7., 12., 11., 10., 12.,
        9., 10., 11., 10.,  8.,  5., 10.,  8.,  8., 10.,  7., 14., 12.,
        6.,  8.,  6., 12.,  9., 11.,  9.,  5.,  9., 12., 11., 10., 11.,
        8., 11.,  5.,  9., 12.,  8., 13., 13., 11.,  7., 10., 10.,  9.

        7.,  6.,  8., 11.,  8., 14., 11., 13., 14.,  6., 10.,  5.,  6.,
        5.,  9., 11., 11.,  9., 13., 12., 12.,  8., 15., 10.,  9., 15.,
       10.,  9., 13., 10.,  5., 12.,  7., 12.,  6.,  7.,  8.,  6., 12.,
       11.,  9., 11.,  9.,  7.,  8., 14.,  9.,  7., 10.,  6.,  8., 10.,
       12.,  8., 11.,  5.,  6.,  8., 14., 12.,  9.,  8.,  6.,  9.,  9.,
        8., 10., 10., 11.,  7., 13., 11.,  7.,  5.,  5.,  8.,  9.,  9.,
       12.,  6., 15.,  9., 15.,  9.,  8., 10.,  6., 12.,  5., 11.,  8.,
       13., 12., 13., 11.,  9.,  9.,  8.,  9.,  5., 10.,  8.,  6.,  7.,
        9., 11., 11.,  7., 10., 11., 12.,  9.,  9.,  4., 10.,  9., 11.,
        7.,  7.,  5.,  7.,  6.,  9.,  9., 12.,  5., 12.,  8., 10.,  4.,
       15.,  5.,  8., 15.,  9.,  9.,  9.,  9.,  5.,  9., 11.,  5.,  7.,
        8., 15., 10.,  8.,  8.,  9., 14., 11.,  6.,  6.,  9.,  6.,  9.,
        9., 11., 12., 10.,  5.,  7., 13., 11., 12.,  8., 11.,  8.,  8.,
        8.,  8.,  9., 10.,  7.,  8.,  7., 10.,  8.,  6.,  8.,  7.,  5.,
       10., 10.,  9., 13.,  9., 15.,  8.,  8.,  8., 12.,  8.,  7.,  9.,
        9., 10.,  6., 14., 11., 12., 15., 10., 14.,  6., 11.,  7.,  7.,
        6.,  3.,  9.,  8., 12., 10., 14.,  7., 10.,  9.,  8.,  8.,  6.,
       10., 11.,  9.,  5., 11., 14.,  8.,  8., 10.,  6., 11., 10.,  9.,
       15.,  9., 11.,  8.,  5.,  6.,  5.,  8.,  8., 13.,  8.,  8.,  5.,
       12.,  6.,  5., 12.,  5., 10.,  5.,  7., 12., 14.,  8.,  8.,  5.,
        5.,  8.,  8., 12., 12.,  4., 11.,  6.,  9.,  9.,  9., 12.,  9.,
        6., 12., 12., 10.,  9.,  9.,  5.,  9., 13.,  5., 11., 11.,  9.,
        7.,  9.,  9., 12., 11., 14.,  9.,  9., 13., 11.,  9.,  5., 13.,
       10.,  9.,  5., 10.,  7.,  7.,  7.,  6., 10.,  5., 10.,  4., 12.,
        8.,  5., 13.,  9.,  9., 11., 11.,  8., 11., 11.,  8.,  9.,  8.,
        7.,  6., 13.,  9.,  9.,  9., 11.,  4.,  8., 10.,  9.,  8., 10.,
       13.,  8., 13., 15., 15.,  6.,  8.,  8., 11.,  7., 13.,  5., 11.,
       12., 10., 11.,  8.,  9., 10., 10., 15., 12., 12.,  9.,  5.,  7.,
       11.,  9.,  8.,  5.,  9., 11.,  7., 14., 15.,  4., 10.,  5., 12.,
       14., 12., 12., 11., 12., 12., 10.,  7.,  5.,  9.,  4.,  5.,  5.,
       14., 10.,  5., 13., 11., 11.,  3.,  5.,  5., 13., 10., 10.,  5.,
       10., 10., 15.,  9., 11.,  9., 12.,  9., 10., 10., 10., 11.,  8.,
        8.,  8., 10., 11.,  8.,  6.,  8.,  5.,  7.,  8.,  9., 10., 10.,
       11.,  9.,  8.,  7.])
```

14.Measure the performance using Metrics

In [36]:
```
from sklearn import metrics
```

In [37]:
```
metrics.confusion_matrix(y_test,pred2)
```

Out[37]:
```
array([[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 1,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  1,  5,  6,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  2,  6, 22,  9,  1,  6,  4,  7, 10,  7,  4,  5,  3],
       [ 0,  0,  1,  7, 11, 15,  8,  4,  0,  1,  1,  0,  0,  0],
       [ 0,  0,  1,  4, 19, 14, 25, 10,  4,  2,  2,  2,  0,  1],
       [ 0,  0,  1,  3,  6, 15, 24, 24, 13,  7,  5,  0,  0,  1],
       [ 0,  0,  0,  5,  3,  8, 26, 30, 28, 19, 12,  7,  3,  1],
       [ 0,  0,  0,  9,  1,  7, 22, 29, 15, 21, 16,  8,  7,  4],
       [ 0,  0,  0,  9,  1,  1,  9, 19, 20, 12,  7,  7,  3,  5],
       [ 0,  0,  0,  5,  0,  1,  5,  6, 12,  9,  8,  4,  0,  1],
       [ 0,  0,  0,  3,  2,  0,  1,  3,  7,  6,  1,  3,  2,  3],
       [ 0,  0,  0,  3,  0,  0,  4,  5,  1,  4,  0,  1,  6,  2],
       [ 0,  0,  0,  2,  0,  0,  0,  3,  2,  4,  3,  6,  1,  0]],
      dtype=int64)
```

In [39]:
```
print('DT model Accuracy Score:',metrics.accuracy_score(y_test,pred2))
```

```
DT model Accuracy Score: 0.17942583732057416
```

In [40]:
```
acc=metrics.accuracy_score(y_test,pred2)
acc
```

Out[40]:
```
0.17942583732057416
```

In [41]:
```
1-acc
```

Out[41]:
```
0.8205741626794258
```

In [42]:
```
data.head()
```

Out[42]:

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