1.Download the dataset, 2.Load the dataset

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

```
data=pd.read_csv("abalone.csv",encoding='ISO-8859-1')
data.head()
```

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

Saving...

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	

```
data.dtypes
```

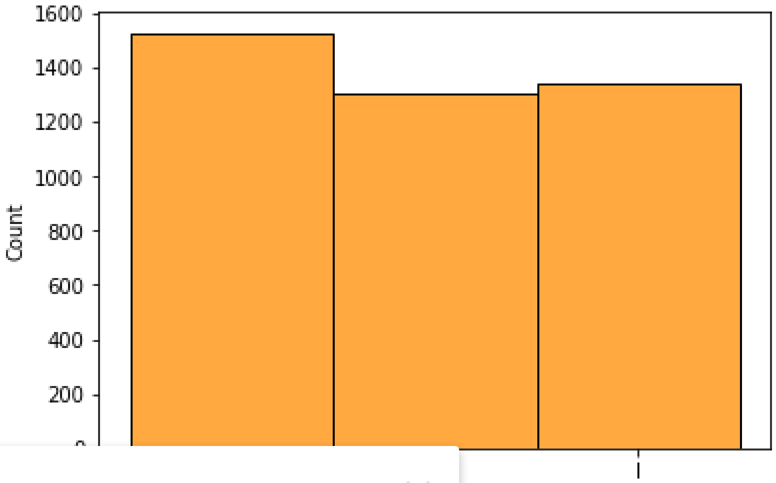
```
Sex              object
Length          float64
Diameter        float64
Height          float64
Whole weight    float64
```

```
Shucked weight    float64
Viscera weight    float64
Shell weight      float64
Rings               int64
Age               float64
dtype: object
```

3.Perform Below Visualizations Univariate Analysis ,Bi - Variate Analysis,Multi - Variate Analysis

```
#univariate analysis "Histogram"
sns.histplot(data["Sex"],color='darkorange')
```

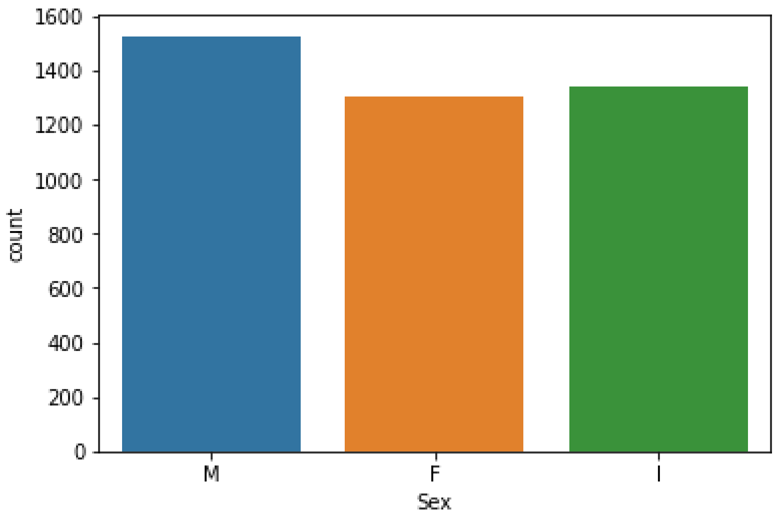
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2997189590>
```



Saving...

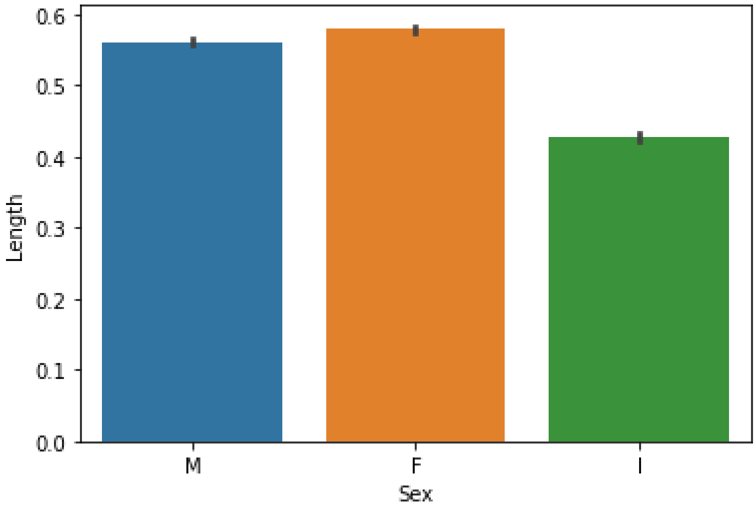
```
#univariate analysis "Countlot"
sns.countplot(data['Sex'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f299706f610>
```



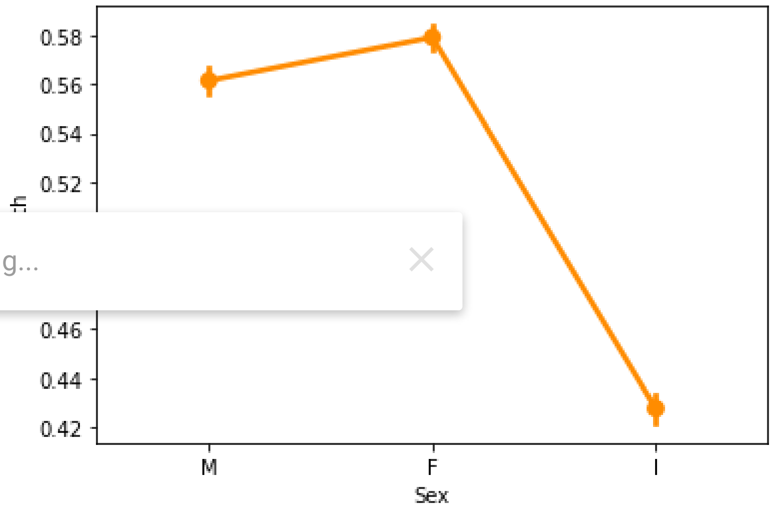

```
#bivariate analysis"Barplot"
sns.barplot(x='Sex',y='Length',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f299708aa10>
```



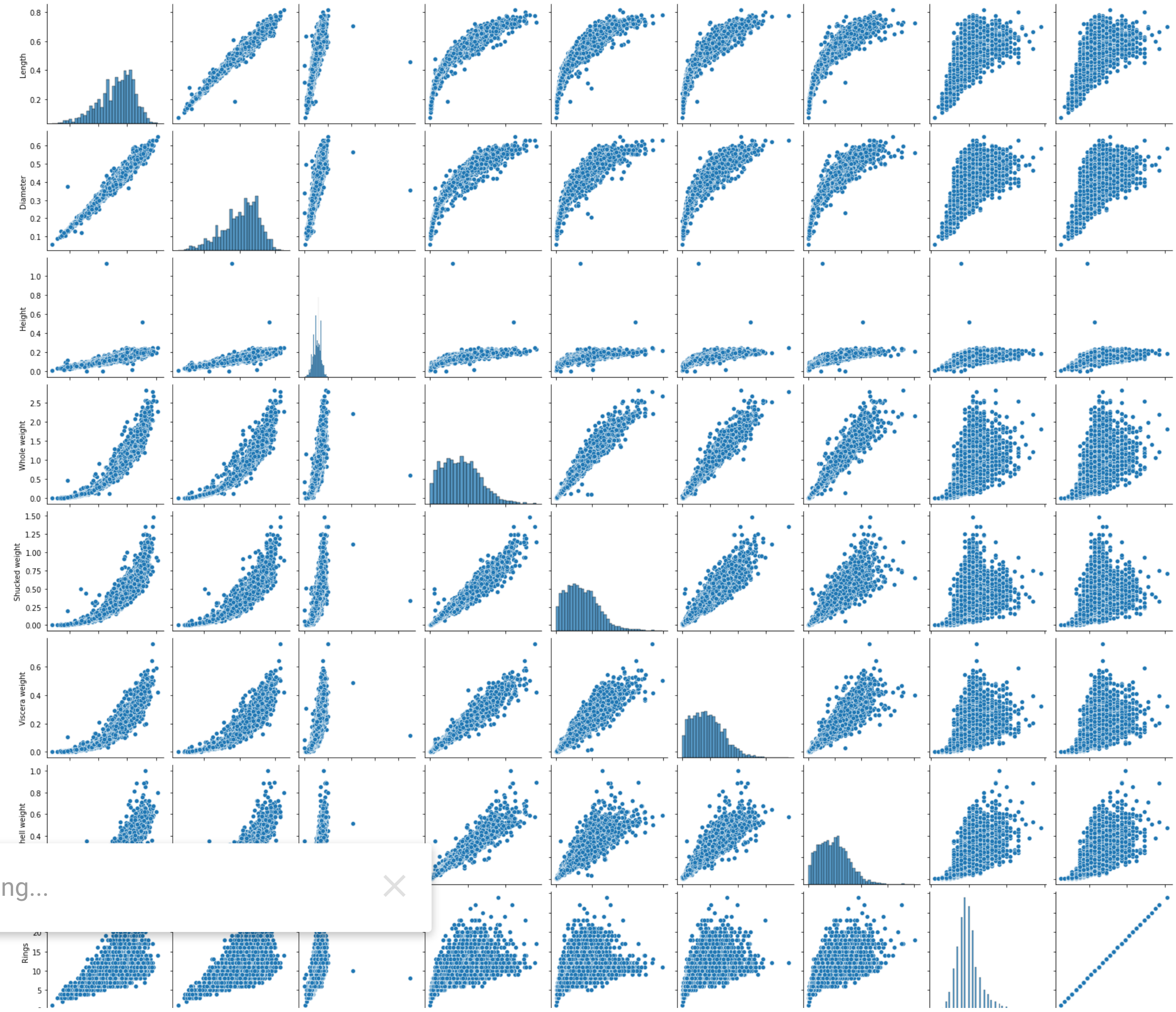

```
#bivariate analysis"Pointplot"
sns.pointplot(x='Sex',y='Length',data=data,color='darkorange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2996b2e9d0>
```




```
#Multivariate analysis"Pairplot"
sns.pairplot(data)
```

<seaborn.axisgrid.PairGrid at 0x7f2996abaa90>



4. Perform descriptive statistics on the dataset.

```
# Descriptive statistics of the data set accessed.
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Length	4177.0	0.523992	0.120093	0.0750	0.4500	0.5450	0.615	0.8150
Diameter	4177.0	0.407881	0.099240	0.0550	0.3500	0.4250	0.480	0.6500
Height	4177.0	0.139516	0.041827	0.0000	0.1150	0.1400	0.165	1.1300
Whole weight	4177.0	0.828742	0.490389	0.0020	0.4415	0.7995	1.153	2.8255
Shucked weight	4177.0	0.359367	0.221963	0.0010	0.1860	0.3360	0.502	1.4880
Viscera weight	4177.0	0.180594	0.109614	0.0005	0.0935	0.1710	0.253	0.7600
Shell weight	4177.0	0.238831	0.139203	0.0015	0.1300	0.2340	0.329	1.0050
Rings	4177.0	9.933684	3.224169	1.0000	8.0000	9.0000	11.000	29.0000
Age	4177.0	11.433684	3.224169	2.5000	9.5000	10.5000	12.500	30.5000

```
data.isnull().any().any()
```

```
False
```

```
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
Age               False
dtype: bool
```

```
df2=data.dropna(how='all')
```

5.Handle the Missing values

```
df2.isnull().sum().sum()
```


This dataset does not contain any missing value.

```
missing_values=data.isnull().sum()
missing_values[missing_values>0]/len(data)*100
```

```
Series([], dtype: float64)
```

6.Find the outliers and replace the outliers

```
sns.boxplot(data['Age'],data=data)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f2991e52710>
```

```
for x in ['Age']:
    q75,q25 = np.percentile(data.loc[:,x],[75,25])
    intr_qr = q75-q25

    max = q75+(1.5*intr_qr)
    min = q25-(1.5*intr_qr)

    data.loc[data[x] < min,x] = np.nan
    data.loc[data[x] > max,x] = np.nan
```

```
data.isnull().sum()
```

```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
```

Saving...

```
Age             278
dtype: int64
```

7.Check for Categorical columns and perform encoding.

```
df2.Sex.value_counts()
```

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

```
from sklearn.preprocessing import LabelEncoder

le= LabelEncoder()
```

```
df2.Sex = le.fit_transform(df2.Sex)
df2.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Age
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	16.5
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	8.5

```
df2.describe()
```

	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

4

Saving...

```
df2.corr()
```

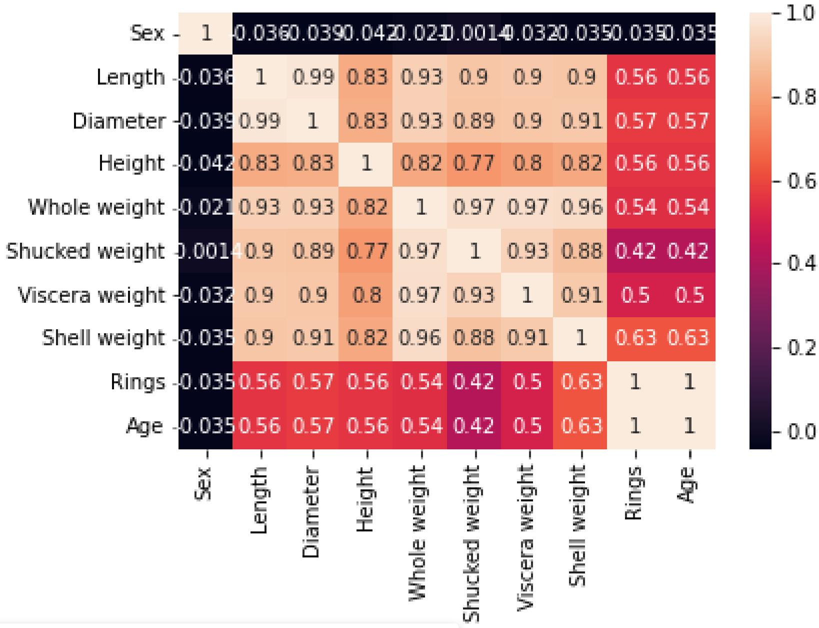
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight

```
import seaborn as sns
```

Length	-0.036066	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.

```
sns.heatmap(df2.corr(),annot = True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f298fbc2f50>



Saving...

```
df2.head()
```

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

```
X = df2.drop(columns = ['Rings'], axis = 1)
X.head()
```

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

```
y = df2.Rings
y.head()
```

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

```
from sklearn.preprocessing import MinMaxScaler
```

```
scale = MinMaxScaler()
```

```
x_scaled = pd.DataFrame(scale.fit_transform(X), columns = X.columns)
```

```
x_scaled.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Age
0	1.0	0.513514	0.521008	0.084071	0.181335	0.150303	0.132324	0.147982	0.500000
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]79646	0.079157	0.066241	0.063199	0.068261	0.214286
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]119469	0.239065	0.171822	0.185648	0.207773	0.285714
3	1.0	0.493243	0.521008	0.110619	0.182044	0.144250	0.149440	0.152965	0.321429

8.Split the data into dependent and independent variables

```
from sklearn.datasets import load_iris
```

```
from sklearn import preprocessing
data = load_iris()

# separate the independent and dependent variables
X_data = data.data
target = data.target
print("Dependent variable")
print(X_data)
print("Independent variable")
print(target)
```

```
[5.1 2.5 3.  1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
```

```
 [6.5 3.  5.8 2.2]
 [7.6 3.  6.6 2.1]
 [4.9 2.5 4.5 1.7]
 [7.3 2.9 6.3 1.8]
 [6.7 2.5 5.8 1.8]
 [7.2 3.6 6.1 2.5]
 [6.5 3.2 5.1 2. ]
 [6.4 2.7 5.3 1.9]
 [6.8 3.  5.5 2.1]
 [5.7 2.5 5.  2. ]
 [5.8 2.8 5.1 2.4]
 [6.4 3.2 5.3 2.3]
 [6.5 3.  5.5 1.8]
 [7.7 3.8 6.7 2.2]
 [7.7 2.6 6.9 2.3]
 [6.  2.2 5.  1.5]
 [6.9 3.2 5.7 2.3]
 [5.6 2.8 4.9 2. ]
 [7.7 2.8 6.7 2. ]
 [6.3 2.7 4.9 1.8]
 [6.7 3.3 5.7 2.1]
 [7.2 3.2 6.  1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3.  4.9 1.8]
 [6.4 2.8 5.6 2.1]
 [7.2 3.  5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2. ]
 [6.4 2.8 5.6 2.2]
```

Saving...

```
 [7.7 3.  6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
Independent variable
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

9.Scale the independent variable

```
# scale of independent variables
standard = preprocessing.scale(target)
print(standard)
```

```
[-1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487  0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          1.22474487  1.22474487
  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487]
```

Saving...

Double-click (or enter) to edit

10. Split the data into training and testing

```
# train test split

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,test_size=0.2,random_state = 1
```

```
x_train.shape
```

```
(3341, 9)
```

```
x_test.shape
```

```
(836, 9)
```

11. Build the Model,
12. Train the Model

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

model.fit(x_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Conver
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression()
```

```
test_pred= model.predict(x_test)
test_pred
```

```
array([ 9,  9,  7,  9, 13,  6,  9,  9, 11,  8, 10,  6,  9, 10,  7,  9, 11,
       11, 10,  9,  9, 11,  7, 11,  9,  9, 11, 13, 10, 10, 10,  9,  9,  8,
        7,  7, 10, 11,  9, 10,  9, 11, 11, 10,  8,  8, 10,  9, 10,  9, 10,
        9, 10,  6, 11,  9,  9,  7,  8,  7,  8,  9, 11,  9,  8,  8, 10, 10,
        9,  8,  9,  9,  7,  9,  8,  7, 11, 10,  9,  9,  8,  8,  9,  8, 11,
        8, 11,  7, 11,  9, 10,  8,  8,  9,  7,  9,  9, 10,  6, 13,  8, 10,
        7,  8, 11,  8, 13,  9, 11,  9,  7, 10,  9,  7,  7, 10,  8,  7, 10,
       [illegible]  9, 10, 10,  8,  9,  8,  7,  8,  8, 10,  7,
       [illegible]  9, 11,  8,  9, 10, 11,  9,  9, 10,  7,  9,
       10,  9,  9, 10, 10,  9, 11,  9, 10, 10,  6,  8,  7, 11,  9,  9, 13,
        7, 11, 11,  8,  8, 10, 11, 11,  7,  8,  9,  9,  9,  7,  7, 10, 12,
        9, 10, 11,  6,  8,  6, 11,  8, 11,  8,  5,  7,  9,  5,  8, 11,  9,
        8,  8,  8,  8,  9,  7, 10,  8,  9,  8,  8,  8, 10,  9, 10,  8, 13,
        8, 10, 10,  8, 11,  9, 11, 10,  9,  9,  9, 10, 10,  8,  6, 10, 10,
       10,  9, 11,  9,  9,  9,  7,  9, 10,  7,  8, 10,  9, 11, 11, 11,  6,
       12, 10, 13,  7, 10, 10, 11,  8,  9,  8,  9, 11,  9,  9, 11,  9, 12,
       11,  9, 11, 13,  7,  9, 10, 11,  9,  8, 10,  7,  7,  8,  6,  7,  7,
        9,  8,  8, 10,  9,  8, 10,  7,  9, 11,  9, 11,  8,  8,  9, 11,  9,
       13,  8,  8,  9,  8, 10,  9,  7,  7, 13, 11, 11,  9,  5,  9, 11, 10,
        9, 13, 10,  8, 10,  9, 11, 10,  8, 10,  7,  8,  9,  9,  5, 11, 11,
        7,  7, 11, 11,  9, 13, 13,  9, 11,  9,  9,  6, 11,  8,  6,  9,  8,
       10, 11,  9,  9,  6, 10,  8,  8, 10,  6,  8, 11,  7,  9, 10, 10, 10,
        9,  9,  8, 11,  8,  6, 11,  7,  7, 13, 11,  7,  8, 11,  8,  9,  7,
       10, 10,  8,  9,  7, 10, 10, 11, 11,  8,  7,  9, 10, 11,  7, 10, 13,
        9, 11,  9, 10,  9,  9, 12,  7,  9,  9,  8,  8,  8,  9, 11,  8, 13,
        5,  6, 10,  9,  9, 10,  8, 11,  8,  8, 10, 12,  8,  8,  9, 11,  8,
        8,  9, 10,  8, 10, 11,  8,  8, 11,  9,  5,  7,  8, 11,  9,  8, 11,
       10, 10,  7,  9, 11,  9, 11, 10, 12, 10,  9, 10,  9,  6,  8,  9,  9,
        8, 11, 11,  5,  7,  7, 10, 10,  7,  8,  7,  9,  9, 10,  8,  9,  8,
        7,  6,  7,  7, 10,  9,  8, 11,  6,  9,  8, 11, 11,  9,  9, 10, 11,
        9,  9, 13,  7,  6,  8,  7, 11,  9,  8,  8, 11,  9, 10,  8, 11,  9,
        5, 10, 13, 11, 10,  9, 11, 10, 11, 13, 11,  7,  7, 11, 11,  9,  9,
        7, 10,  8,  9,  9, 10,  9, 11,  9, 11, 10, 10,  8, 10,  8,  8,  9,
        8,  8,  9,  8, 10, 13, 10, 11,  8, 10,  7,  9,  9, 11, 10,  8, 11,
        8, 10,  9,  9, 10,  5, 13, 11,  8,  9,  5, 11,  8, 13,  8,  9,  8,
        6,  9, 10,  5,  9,  9,  8, 10, 10,  8, 10, 11, 10, 11, 11,  9,  9,
        9,  7, 11,  7,  7,  9,  9,  8,  9, 11,  7,  9, 11, 10,  8,  9,  8,
        7,  7,  9,  9, 10,  9,  7, 10,  7,  8,  9,  9,  6, 10,  8,  8, 10,
```

```
        7, 11,  8,  7, 13,  9, 11, 10,  8, 13,  8,  9, 11,  7,  9,  9,  8,
        9,  9,  9,  9,  7,  6,  6,  9, 11,  6,  7, 10, 11,  8,  9,  8, 13,
        7,  9,  9,  8,  7,  7, 10,  7,  8, 10,  9, 11, 10, 11,  9,  9,  7,
        8,  9,  6, 10,  8,  7, 11, 10, 10, 10, 11,  7,  7,  7,  9,  8, 11,
       10, 10,  8, 13,  9,  8,  9, 12, 10,  9, 11, 11, 11,  9, 13,  8,  9,
       11,  9,  8,  9, 10, 13,  9, 11,  8,  9,  8, 11,  9, 11, 10,  9, 10,
        7,  8,  7, 10,  9,  7, 10,  8,  7,  7,  9,  6,  7,  7,  8,  9,  9,
        7,  9,  8,  8, 11,  9,  7,  9, 10, 10,  8,  7,  8,  8,  8, 11, 10,
       10,  9, 10, 11, 11, 11,  8, 10,  9, 10, 11, 10,  8, 11, 10,  6,  7,
        9, 10, 10, 10, 13,  8,  7,  9, 10, 10,  9,  7,  8, 10,  6,  9,  9,
        9,  8,  7,  9, 11, 11,  7, 10, 11,  6,  9,  9,  8,  7, 10,  9, 11,
        9, 10,  9])
```

```
y_test
```

```
17      10
1131     8
299      9
1338    10
2383    16
        ..
1787     8
3075    11
2766     8
1410    10
2529     8
Name: Rings, Length: 836, dtype: int64
```

Saving...

```
train_pred
```

```
array([9, 6, 8, ..., 7, 9, 6])
```

13. Test the Model

```
# testing on a random value

model.predict([[1,20,3000]])
```

14. Measure the performance using Metrics.

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_curv
```

```
accuracy_score(y_test,test_pred)
```

```
0.2631578947368421
```

```
accuracy_score(y_train,train_pred)
```

```
0.30859024244238253
```

```
confusion_matrix(y_test,test_pred)
```

```
array([[ 0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  4,  6,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  3, 11,  6,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  1,  8, 28, 16,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  4, 25, 43,  8,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 17, 44, 45,  1,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 10, 35, 70, 16,  1,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  1, 10, 10, 65, 48,  9,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1,  5, 22, 41, 22,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  5,  2,  8, 28, 20,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  2,  0,  0, 12, 25,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  4, 20,  0,  0,  0,  0,  0,  0,
         [illegible]],
       [illegible],  0,  0,  0, 17,  0,  4,  0,  0,  0,  0,
         [illegible]],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  6,  1,  3,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  5,  2,  5,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  6,  1,  3,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  6,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  3,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  2,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  2,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0]])
```

Saving... ×

```
pd.crosstab(y_test,test_pred)
```

col_0	5	6	7	8	9	10	11	12	13
Rings									
2	1	0	0	0	0	0	0	0	0
3	2	0	0	0	0	0	0	0	0
4	4	6	1	0	0	0	0	0	0
5	3	11	6	0	0	0	0	0	0
6	1	8	28	16	0	0	0	0	0
7	0	4	25	43	8	0	0	0	0
8	0	0	17	44	45	1	0	0	0
9	0	0	10	35	70	16	1	0	0
10	0	1	10	10	65	48	9	0	0
11	0	0	1	5	22	41	22	0	0
12	0	0	5	2	8	28	20	0	0
13	0	0	2	0	0	12	25	0	0
14	0	0	0	0	0	4	20	0	0
15	0	0	0	0	0	0	17	0	4
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	6	1	3
17	0	0	0	0	0	0	5	2	5
18	0	0	0	0	0	0	6	1	3
19	0	0	0	0	0	0	0	1	6

Saving...

```
print(classification_report(y_test,test_pred))
```

```
              precision    recall  f1-score   support

           2       0.00      0.00      0.00         1
           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00        11
           5       0.27      0.15      0.19        20
           6       0.27      0.15      0.19        53
           7       0.24      0.31      0.27        80
           8       0.28      0.41      0.34       107
           9       0.32      0.53      0.40       132
          10       0.32      0.34      0.33       143
          11       0.17      0.24      0.20        91
          12       0.00      0.00      0.00        63
          13       0.00      0.00      0.00        39
          14       0.00      0.00      0.00        24
          15       0.00      0.00      0.00        21
          16       0.00      0.00      0.00        10
          17       0.00      0.00      0.00        12
          18       0.00      0.00      0.00        10
          19       0.00      0.00      0.00         7
          20       0.00      0.00      0.00         4
```

```
          21       0.00      0.00      0.00         3
          22       0.00      0.00      0.00         1
          23       0.00      0.00      0.00         2

    accuracy                           0.26       836
   macro avg       0.09      0.10      0.09       836
weighted avg       0.21      0.26      0.23       836

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Unde
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Unde
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Unde
  _warn_prf(average, modifier, msg_start, len(result))
```

Saving...

Colab paid products - Cancel contracts here

✓ 0s completed at 8:23 PM