Assignment-4

Name : Kishore Kumar.P

Roll No : 19CS039

1.Loading Dataset into tool

```
from google.colab import files
uploaded = files.upload()
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving abalone.csv to abalone.csv

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv("abalone.csv")
```

2.Performing Visualization

Univariate Analysis

```
data.head()
```

	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

```
sns.boxplot(data['Diameter'])
```

```
plt.hist(data['Diameter'])
(array([  13.,   66.,  180.,  344.,  513.,  812., 1017.,  934.,  275.,
          23.]),
 array([0.055 , 0.1145, 0.174 , 0.2335, 0.293 , 0.3525, 0.412 , 0.4715,
        0.531 , 0.5905, 0.65  ]),
 )

plt.plot(data['Diameter'].head(10))
[]

plt.pie(data['Diameter'].head(),autopct='%.3f')
([,
  ,
  ,
  ,
  ],
 [Text(0.8507215626110557, 0.6973326486753676, ''),
  Text(-0.32611344931648134, 1.0505474849691026, ''),
  Text(-1.0998053664078908, -0.02069193128747144, ''),
  Text(-0.08269436219656089, -1.096887251480709, ''),
  Text(0.9758446362287218, -0.5076684409569241, '')],
 [Text(0.46402994324239394, 0.3803632629138369, '21.856'),
  Text(-0.17788006326353525, 0.5730259008922377, '15.868'),
  Text(-0.5998938362224858, -0.011286507974984419, '25.150'),
  Text(-0.045106015743578656, -0.5983021371712958, '21.856'),
  Text(0.5322788924883937, -0.2769100587037768, '15.269')])

sns.distplot(data['Diameter'].head(300))

plt.scatter(data['Diameter'].head(400),data['Length'].head(400))
```

```
plt.bar(data['Sex'].head(20),data['Rings'].head(20))
plt.title('Bar plot')
plt.xlabel('Diameter')
plt.ylabel('Rings')
```

Text(0, 0.5, 'Rings')

```
sns.barplot(data['Sex'], data['Rings'])
```

```
sns.jointplot(data['Diameter'].head(50),data['Rings'].head(100))
```

```
sns.barplot('Diameter','Rings',hue='Sex',data=data.head())
```

```
sns.lineplot(data['Diameter'].head(),data['Rings'].head())
```

```
sns.boxplot(data['Sex'].head(10),data['Diameter'].head(10),data['Rings'].head(10))
```

```
fig=plt.figure(figsize=(8,5))
sns.heatmap(data.head().corr(),annot=True)
```

```
sns.pairplot(data.head(),hue='Height')
```

```
sns.pairplot(data.head())
```

3.Perform Descriptive Statistics on the dataset

```
data.head()
```

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

data.tail()

	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

data.info()

RangeIndex: 4177 entries, 0 to 4176

Data columns (total 9 columns):

```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
```

dtypes: float64(7), int64(1), object(1)

memory usage: 293.8+ KB

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.mode().T

	0	1
Sex	M	NaN
Length	0.55	0.625
Diameter	0.45	NaN
Height	0.15	NaN
Whole weight	0.2225	NaN
Shucked weight	0.175	NaN
Viscera weight	0.1715	NaN
Shell weight	0.275	NaN
Rings	9.0	NaN

data.shape

(4177, 9)

data.kurt()

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

data.skew()

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

data.var()

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

data.nunique()

Sex 3

Length 134

Diameter 111

Height 51

Whole weight 2429

Shucked weight 1515

Viscera weight 880

Shell weight 926

Rings 28

dtype: int64

4.Check for missing values and deal with them

data.isna()

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
4172	False	False	False	False	False	False	False	False	False
4173	False	False	False	False	False	False	False	False	False
4174	False	False	False	False	False	False	False	False	False
4175	False	False	False	False	False	False	False	False	False
4176	False	False	False	False	False	False	False	False	False

4177 rows × 9 columns

data.isna().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.isna().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

data.isna().any().sum()

0

5.Find the outliers and replace them outliers

sns.boxplot(data['Diameter'])

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

quant

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130	8.0
0.75	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329	11.0

iqr=quant.loc[0.75]-quant.loc[0.25]

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
low=quant.loc[0.25]-(1.5*iqr)
low
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
up=quant.loc[0.75]+(1.5*iqr)
up
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
data['Diameter']=np.where(data['Diameter']<0.155,0.4078,data['Diameter'])
sns.boxplot(data['Diameter'])

sns.boxplot(data['Length'])
```

```
data['Length']=np.where(data['Length']<0.23,0.52, data['Length'])
sns.boxplot(data['Length'])

sns.boxplot(data['Height'])

data['Height']=np.where(data['Height']<0.04,0.139, data['Height'])
data['Height']=np.where(data['Height']>0.23,0.139, data['Height'])
sns.boxplot(data['Height'])

sns.boxplot(data['Whole weight'])

data['Whole weight']=np.where(data['Whole weight']>0.9,0.82, data['Whole weight'])
sns.boxplot(data['Whole weight'])

sns.boxplot(data['Shucked weight'])

data['Shucked weight']=np.where(data['Shucked weight']>0.93,0.35, data['Shucked weight'])
sns.boxplot(data['Shucked weight'])

sns.boxplot(data['Viscera weight'])

data['Viscera weight']=np.where(data['Viscera weight']>0.46,0.18, data['Viscera weight'])
sns.boxplot(data['Viscera weight'])

sns.boxplot(data['Shell weight'])

data['Shell weight']=np.where(data['Shell weight']>0.61,0.2388, data['Shell weight'])
sns.boxplot(data['Shell weight'])
```

6.Check for Categorical columns and perform encoding.

```
data['Sex'].replace({'M':1,'F':0,'I':2},inplace=True)
data
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	1	0.590	0.440	0.135	0.8200	0.4390	0.2145	0.2605	10
4174	1	0.600	0.475	0.205	0.8200	0.5255	0.2875	0.3080	9
4175	0	0.625	0.485	0.150	0.8200	0.5310	0.2610	0.2960	10
4176	1	0.710	0.555	0.195	0.8200	0.3500	0.3765	0.4950	12

4177 rows × 9 columns

7.Split the data into dependent and independent variables.

```
x=data.drop(columns= ['Rings'])
y=data['Rings']
x
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550
...	...	...	...	...	...	...	...	...
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490

4173	1	0.590	0.440	0.135	0.8200	0.4390	0.2145	0.2605
4174	1	0.600	0.475	0.205	0.8200	0.5255	0.2875	0.3080
4175	0	0.625	0.485	0.150	0.8200	0.5310	0.2610	0.2960
4176	1	0.710	0.555	0.195	0.8200	0.3500	0.3765	0.4950

4177 rows × 8 columns

```
y
0      15
1       7
2       9
3      10
4       7
       ..
4172   11
4173   10
4174    9
4175   10
4176   12
Name: Rings, Length: 4177, dtype: int64
```

8.Scale the independent variables

```
from sklearn.preprocessing import scale
x = scale(x)
x
array([[-0.0105225 , -0.67088921, -0.50179694, ..., -0.61037964,
        -0.7328165 , -0.64358742],
       [-0.0105225 , -1.61376082, -1.57304487, ..., -1.22513334,
        -1.24343929, -1.25742181],
       [-1.26630752,  0.00259051,  0.08738942, ..., -0.45300269,
        -0.33890749, -0.18321163],
       ...,
```

```
      [-0.0105225 ,  0.63117159,  0.67657577, ...,  0.86994729,
        1.08111018,  0.56873549],
      [-1.26630752,  0.85566483,  0.78370057, ...,  0.89699645,
        0.82336724,  0.47666033],
      [-0.0105225 ,  1.61894185,  1.53357412, ...,  0.00683308,
        1.94673739,  2.00357336]])
```

9.Split the data into training and testing

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2)
print(x_train.shape, x_test.shape)
(3341, 8) (836, 8)
```

10.Build the Model

```
from sklearn.linear_model import LinearRegression
MLR=LinearRegression()
```

11.Train the model

```
MLR.fit(x_train,y_train)
LinearRegression()
```

12.Test the model

```
y_pred=MLR.predict(x_test)
y_pred
array([ 6.3204331 , 10.41671748, 13.91911179, 12.29316277,  8.7273177 ,
      11.04369928, 12.40210281, 11.6992544 , 12.01785949,  6.57983392,
      11.91353764, 10.79661591, 11.56560952, 10.14326497, 13.16762604,
       9.34621768, 10.76904478, 11.88283609,  9.34461447, 10.08802992,
      12.80140942,  9.58177975, 11.20908126, 10.3662699 , 10.0168299 ,
      15.92815446, 15.93700213,  7.36066362, 13.2889134 , 10.1579858 ,
      11.62833855, 11.08597007, 11.60253151, 11.74194458,  9.75151497,
```

9.16685512, 7.93960537, 10.04563481, 10.81773394, 10.55133893,
7.19389026, 9.30303442, 10.83957317, 10.63432914, 10.19371808,
13.47423856, 9.06825076, 6.69843582, 13.38213142, 9.62823486,
8.20174551, 7.79183041, 9.3338472 , 11.08195328, 11.25321895,
6.11231204, 10.6960639 , 9.23348159, 7.76425036, 11.65342323,
12.6024271 , 7.49694081, 9.71678931, 7.41119139, 6.94925679,
6.34706174, 9.99734923, 6.70117631, 10.71374432, 9.59457302,
7.07847213, 6.6940933 , 9.30356123, 13.66698224, 9.71369221,
17.36952958, 7.81225327, 8.86909973, 9.29540502, 11.03405521,
12.90720962, 13.03952065, 4.90843127, 9.50619996, 10.09434256,
8.67296752, 9.03746047, 8.33310609, 10.60445018, 9.66636969,
7.67351279, 8.74447193, 12.37470593, 7.70552082, 11.35599144,
11.25726129, 10.02276461, 8.01953433, 11.39538114, 7.92288557,
11.02588274, 7.02530311, 10.80014326, 13.22266766, 11.41469264,
7.5577235 , 6.83654146, 6.97820486, 10.29150052, 9.1851768 ,
9.72122817, 9.29569276, 11.98122676, 9.87982582, 8.55374278,
7.67912597, 10.93152036, 11.90656204, 11.93625854, 12.59760271,
11.87092001, 5.99671728, 9.20248712, 11.18185068, 11.13316757,
12.85726928, 9.50993961, 9.39438115, 10.55793101, 8.61221838,
7.12344177, 7.0075169 , 7.56528442, 14.02672909, 13.39176121,
10.27099354, 13.04124533, 9.72264547, 11.63284409, 3.06922786,
8.60297955, 10.80917425, 11.27118306, 6.4757245 , 10.27830248,
10.17409116, 10.39178358, 6.11330216, 8.27295199, 12.0413644 ,
10.43536813, 11.12820999, 10.56478101, 12.12900686, 9.0459273 ,
6.50569617, 8.65471113, 11.17391657, 4.17641665, 6.45933408,
10.94174559, 10.56404265, 7.32806471, 10.90718067, 8.76983179,
9.54866214, 9.71969088, 9.19215908, 11.19107958, 9.95023994,
10.33050587, 11.98860703, 5.76011208, 8.82560871, 8.26963359,
6.41006108, 7.62776781, 7.77958091, 10.53587014, 8.89399096,
11.50322847, 6.46552063, 6.62035734, 11.27313616, 8.28747988,
12.05544015, 11.6973709 , 12.73972343, 11.36996234, 7.97256548,

9.42073857, 11.25296103, 8.05208624, 10.99827477, 8.28671759,

11.9443616 , 11.82872121, 9.74400382, 8.90145486, 8.57310105,

7.40827472, 11.17489105, 10.0697987 , 9.82070981, 7.33964403,

14.9428325 , 7.76026974, 12.77292992, 6.50073351, 11.29473941,

11.88889387, 7.67192672, 11.10156897, 12.84247625, 6.80849608,