Date : 10 October 2022.

Team ID : PNT2022TMID38676.

Project Name :Exploratory Analysis of RainFall Data in India for Agriculture.

Name : Logeshwaran.v (TM 2)

1.Download the dataset

The dataset was download and some changes applied in this dataset.

In [122…

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

2.Load the dataset

In [123…

```
k1 = pd.read_csv("abalone.csv")
```

In [124…

```
k1.head()
```

Out[124]:

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

In [125…

```
k1.columns
```

Out[125]:

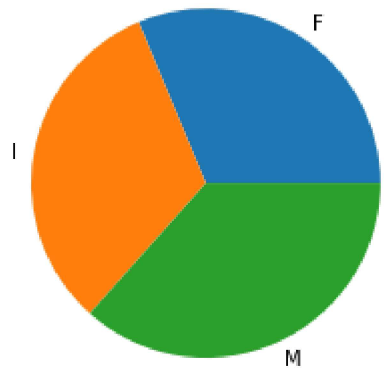
```
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'Age'],
      dtype='object')
```

3. perform Visualization

Univariate Analysis

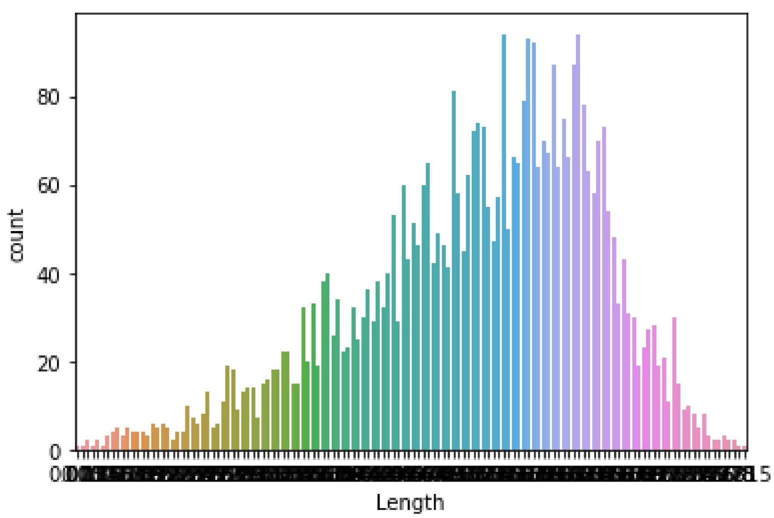
In [126…

```
#pie chart
data1=k1.groupby("Sex",axis=0)
plt.pie(data1.count()["Length"], labels=data1.indices)
plt.show()
```



In [127…

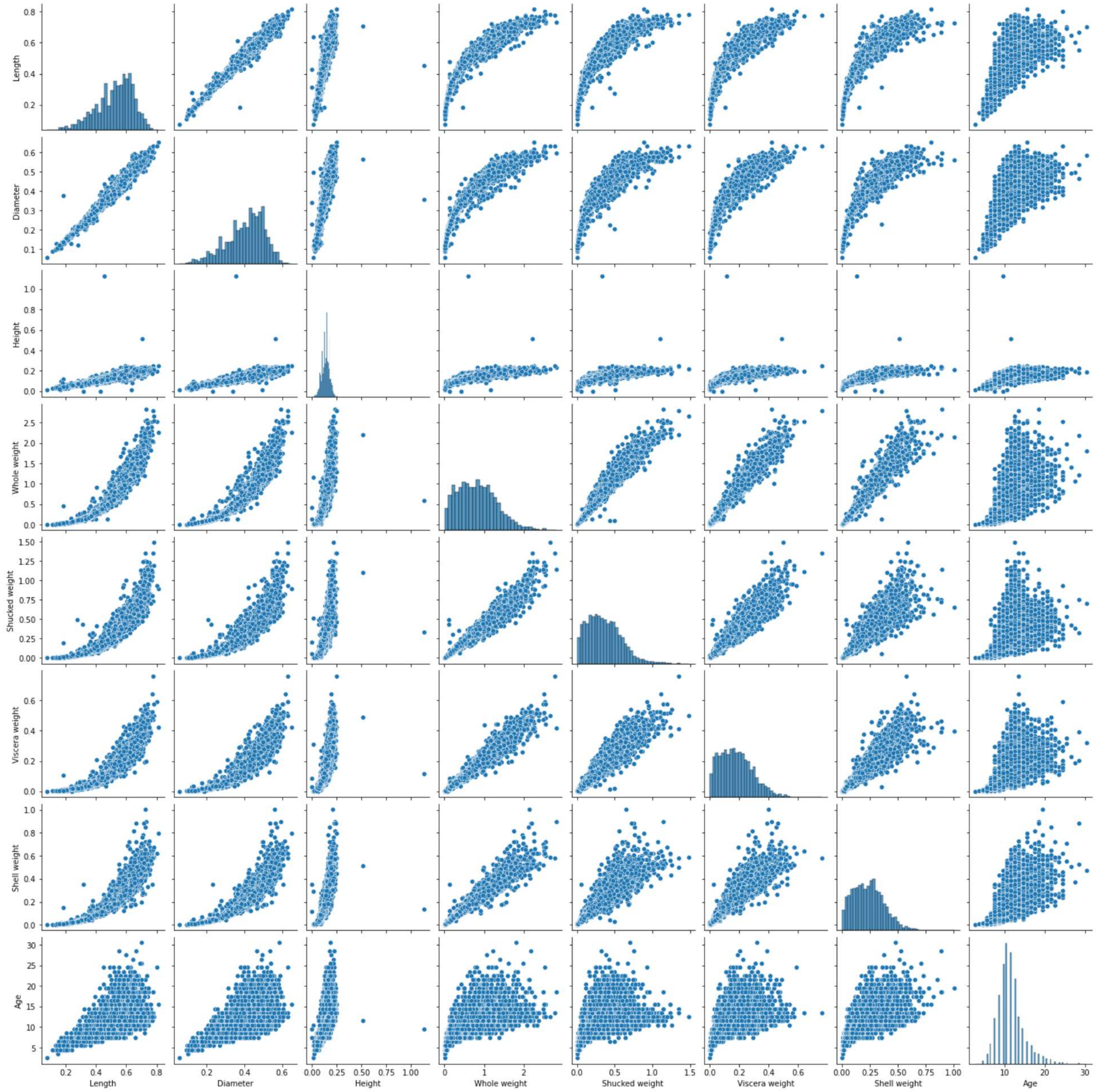
```
#countplot
sns.countplot(x=k1["Length"])
plt.show()
```



Bi-Variate Analysis

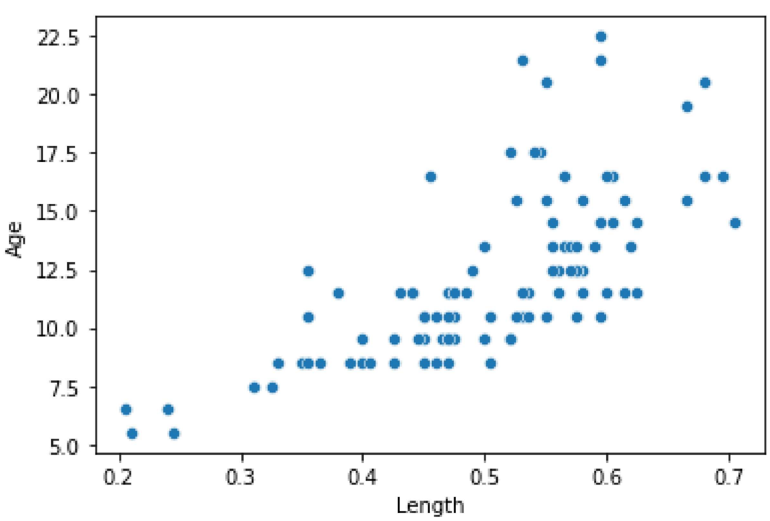
In [128…

```
#pairplot
sns.pairplot(k1)
plt.show()
```



In [129…

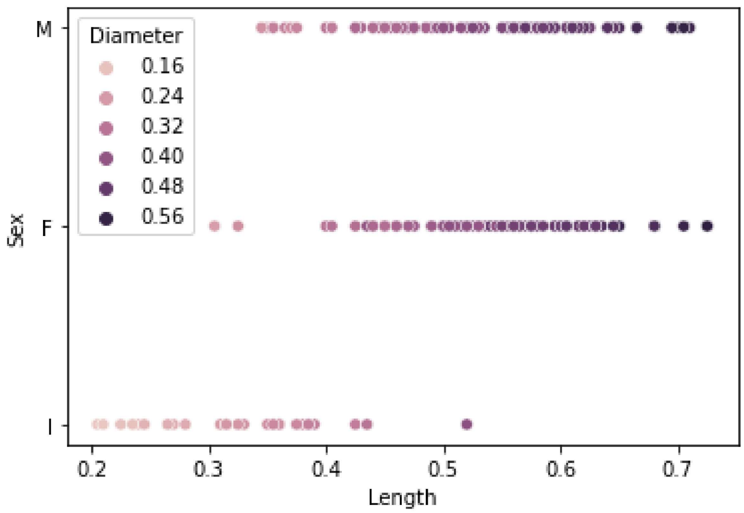
```
#scatterplot for Length and Age
sns.scatterplot(x=k1.iloc[:100,:]["Length"],y=f1.iloc[:100,:]["Age"])
plt.show()
```



Multi-Variate Analysis

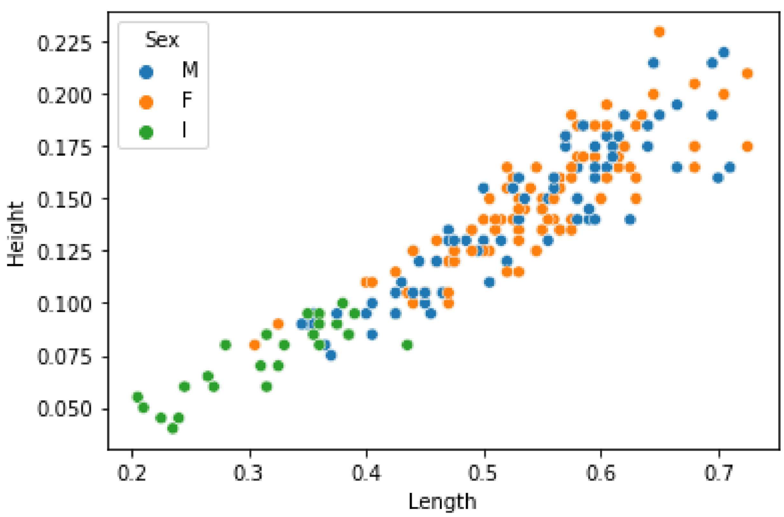
In [158…

```
#scatterplot for Sex, Length and Diameter
sns.scatterplot(x=k1.iloc[:200,:]["Length"],y=f1.iloc[:200,:]["Sex"],hue=f1.iloc[:200,
plt.show()
```



In [159…

```
#scatterplot for Length, Height and Age
sns.scatterplot(x=k1.iloc[:200,:]["Length"],y=f1.iloc[:200,:]["Height"],hue=f1.iloc[:2
plt.show()
```



4.Perform the descriptive statistics on the datase

In [132…

```
k1.describe()
```

Out[132]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
count	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
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000

In [133…

```
k1.mode(numeric_only=True)
```

Out[133]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Age
0	0.550	0.45	0.15	0.2225	0.175	0.1715	0.275	10.5
1	0.625	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [134…

```
k1.median(numeric_only=True)
```

Out[134]:

```
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
Age               10.5000
dtype: float64
```

In [135…

```
k1.skew(numeric_only=True)
```

Out[135]:

```
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
Age               1.114102
dtype: float64
```

In [136…

```
k1.kurt(numeric_only=True)
```

Out[136]:
```
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
Age               2.330687
dtype: float64
```

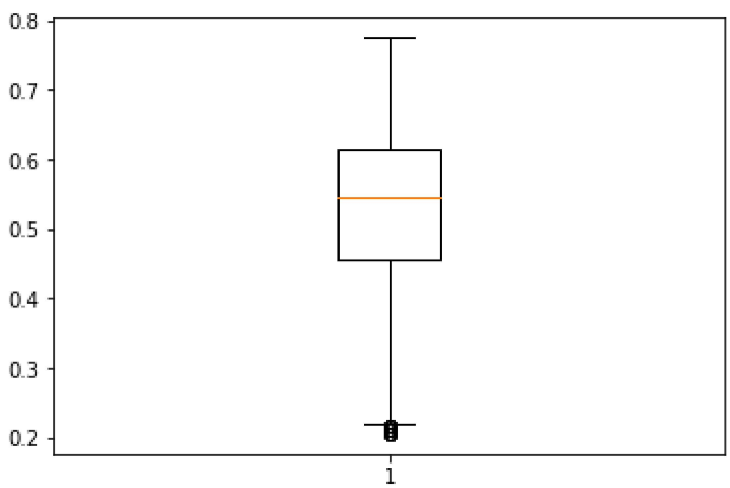
5. Handle the missing values

In [137…
```
#find the null columns
k1.isnull().sum()
```

Out[137]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Age               0
dtype: int64
```

6. Find the outliers and replace the outliers

In [162…
```
#find outliers
plt.boxplot(k1["Length"])
plt.show()
```



In [163…
```
#handling outliers: InterQuartile Range(IQR)
Q3=np.percentile(f1["Length"],75,interpolation='midpoint')
Q1=np.percentile(f1["Length"],25,interpolation='midpoint')
IQR=Q3-Q1
print("Q1: ", Q1)
print("Q3:", Q3)
print("IQR: ", IQR)
```

```
Q1:  0.45
Q3: 0.615
IQR:  0.16499999999999998
```

In [164…

```
upperOutlayers=Q3+1.5*IQR
lowerOutlayers=Q1-1.5*IQR
print(upperOutlayers)
print(lowerOutlayers)
```

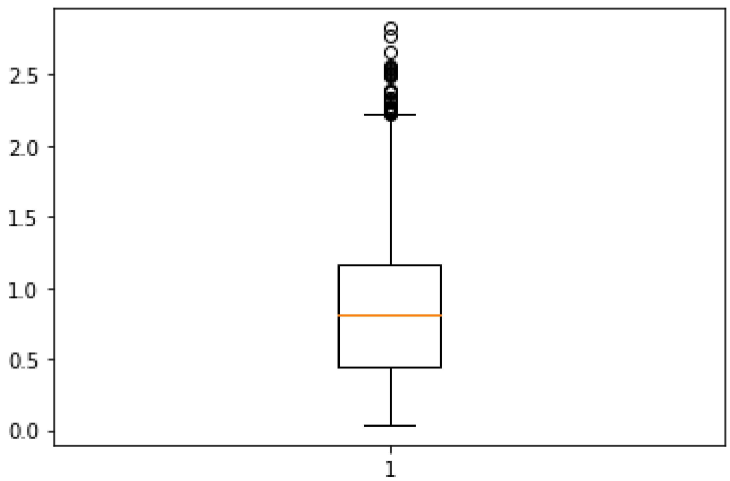
```
0.8624999999999999
0.20250000000000004
```

In [141…

```
k1.drop(np.where(k1["Length"]>=upperOutlayers)[0],inplace=True)
k1.drop(np.where(k1["Length"]<=lowerOutlayers)[0],inplace=True)
```

In [142…

```
#find outliers
plt.boxplot(k1["Whole weight"])
plt.show()
```



In [143…

```
#handling outliers: InterQuartile Range(IQR)
Q3=np.percentile(f1["Whole weight"],75,interpolation='midpoint')
Q1=np.percentile(f1["Whole weight"],25,interpolation='midpoint')
IQR=Q3-Q1
print("Q1: ", Q1)
print("Q3:", Q3)
print("IQR: ", IQR)
```

```
Q1:  0.4415
Q3: 1.153
IQR:  0.7115
```

In [144…

```
upperOutlayers=Q3+1.5*IQR
lowerOutlayers=Q1-1.5*IQR
print(upperOutlayers)
print(lowerOutlayers)
```

```
2.22025
-0.62575
```

In [145…

```
k1.drop(np.where(f1["Whole weight"]>=upperOutlayers)[0],inplace=True)
k1.drop(np.where(f1["Whole weight"]<=lowerOutlayers)[0],inplace=True)
```

7.Check the categorical columns and perform encoding

In [146…

```
k1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4098 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4098 non-null   object 
 1   Length          4098 non-null   float64
 2   Diameter        4098 non-null   float64
 3   Height          4098 non-null   float64
 4   Whole weight    4098 non-null   float64
 5   Shucked weight  4098 non-null   float64
 6   Viscera weight  4098 non-null   float64
 7   Shell weight    4098 non-null   float64
 8   Age             4098 non-null   float64
dtypes: float64(8), object(1)
memory usage: 320.2+ KB
```

In [147…

```
from sklearn.preprocessing import LabelEncoder
k1["Sex"]=LabelEncoder().fit_transform(k1["Sex"])
print(k1["Sex"].unique())
```

```
[2 0 1]
```

In [148…

```
k1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4098 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4098 non-null   int32  
 1   Length          4098 non-null   float64
 2   Diameter        4098 non-null   float64
 3   Height          4098 non-null   float64
 4   Whole weight    4098 non-null   float64
 5   Shucked weight  4098 non-null   float64
 6   Viscera weight  4098 non-null   float64
 7   Shell weight    4098 non-null   float64
 8   Age             4098 non-null   float64
dtypes: float64(8), int32(1)
memory usage: 304.1 KB
```

8.Split the dataset into ipdendent and dependent variables.

In [165…

```
X=k1.iloc[:,:-2].values #independent variables
Y=k1.iloc[:,-2].values #dependent variable

X
```

Out[165]:
```
array([[2.    , 0.455 , 0.365 , ..., 0.514 , 0.2245, 0.101 ],
       [2.    , 0.35  , 0.265 , ..., 0.2255, 0.0995, 0.0485],
       [0.    , 0.53  , 0.42  , ..., 0.677 , 0.2565, 0.1415],
       ...,
       [2.    , 0.6   , 0.475 , ..., 1.176 , 0.5255, 0.2875],
       [0.    , 0.625 , 0.485 , ..., 1.0945, 0.531 , 0.261 ],
       [2.    , 0.71  , 0.555 , ..., 1.9485, 0.9455, 0.3765]])
```

In [166…
```
Y
```

Out[166]:
```
array([0.15 , 0.07 , 0.21 , ..., 0.308, 0.296, 0.495])
```

9.Scale the independent variable

In [151…
```
from sklearn.preprocessing import StandardScaler
std_scaler=StandardScaler().fit_transform(X)
std_scaler
```

Out[151]:
```
array([[ 1.15199292, -0.63240755, -0.4775494 , -1.12098668, -0.66806029],
       [ 1.15199292, -1.55987395, -1.54217646, -1.24510998, -1.28439797],
       [-1.27331764,  0.03006846,  0.10799548, -0.12800026, -0.31983485],
       ...,
       [ 1.15199292,  0.64837939,  0.69354036,  1.60972597,  0.74620502],
       [-1.27331764,  0.86920473,  0.80000306,  0.24436964,  0.5720923 ],
       [ 1.15199292,  1.62001086,  1.54524201,  1.36147936,  2.39653728]])
```

10.Split the data into training and testing.

In [152…
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_stat
X_train.shape
```

Out[152]:
```
(2868, 5)
```

In [153…
```
X_test.shape
```

Out[153]:
```
(1230, 5)
```

11. Build the Model

In [154…
```
#Linear Regression
from sklearn.linear_model import LinearRegression
reg= LinearRegression()
reg.fit(X_train, y_train)
```

Out[154]:
```
LinearRegression()
```

12. Train the Model

In [155…
```
y_pred = reg.predict(X_test)
```

13. Test the model And

14. Measure the performance using Metrics

In [156…
```
from sklearn import metrics
```

In [157…
```
print('Mean absolute error(MAE): {}'.format(metrics.mean_absolute_error(y_test, y_pred
print('Mean squared error(MSE): {}'.format(metrics.mean_squared_error(y_test, y_pred))
print("Intercept: {}".format(reg.intercept_))
print('R2 Score: {}'.format(metrics.r2_score(y_test, y_pred)))
```

```
Mean absolute error(MAE): 0.035513348487061835
Mean squared error(MSE): 0.0027850443765074177
Intercept: -0.02261289768697705
R2 Score: 0.9420641022563094
```