Date : 09 October 2022.

Team ID : PNT2022TMID38676

Project Name : Exploratory Analysis of Rain Fall Data In India For Agriculture.

Name : Vikram P (T L)

1.Download the dataset

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

In [14]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2.Load the dataset

In [15]:
```
f1=pd.read_csv("abalone.csv")
```

In [16]:
```
f1.head()
```

Out[16]:

	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

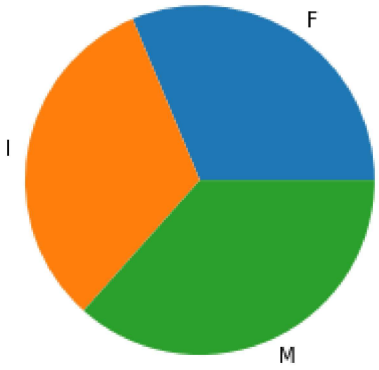
In [17]:
```
f1.columns
```

Out[17]:
```
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'Age'],
      dtype='object')
```

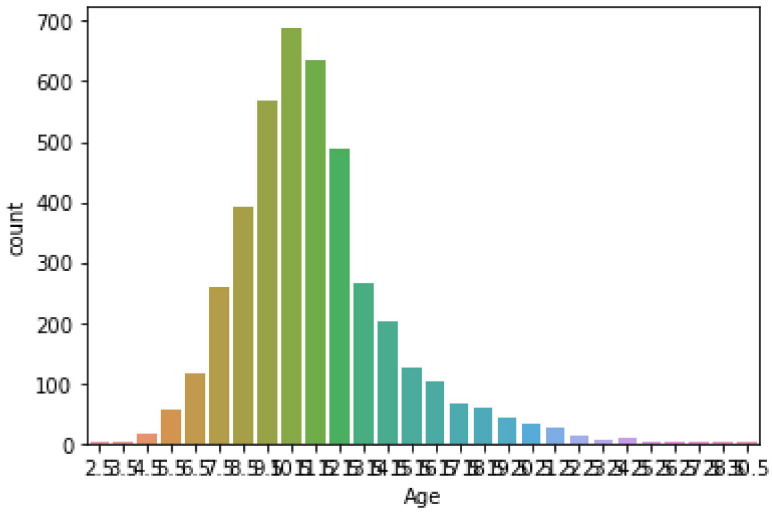
3. perform Visualization

Univariate Analysis

In [18]:
```
#pie chart
data1=f1.groupby("Sex",axis=0)
plt.pie(data1.count()["Age"], labels=data1.indices)
plt.show()
```

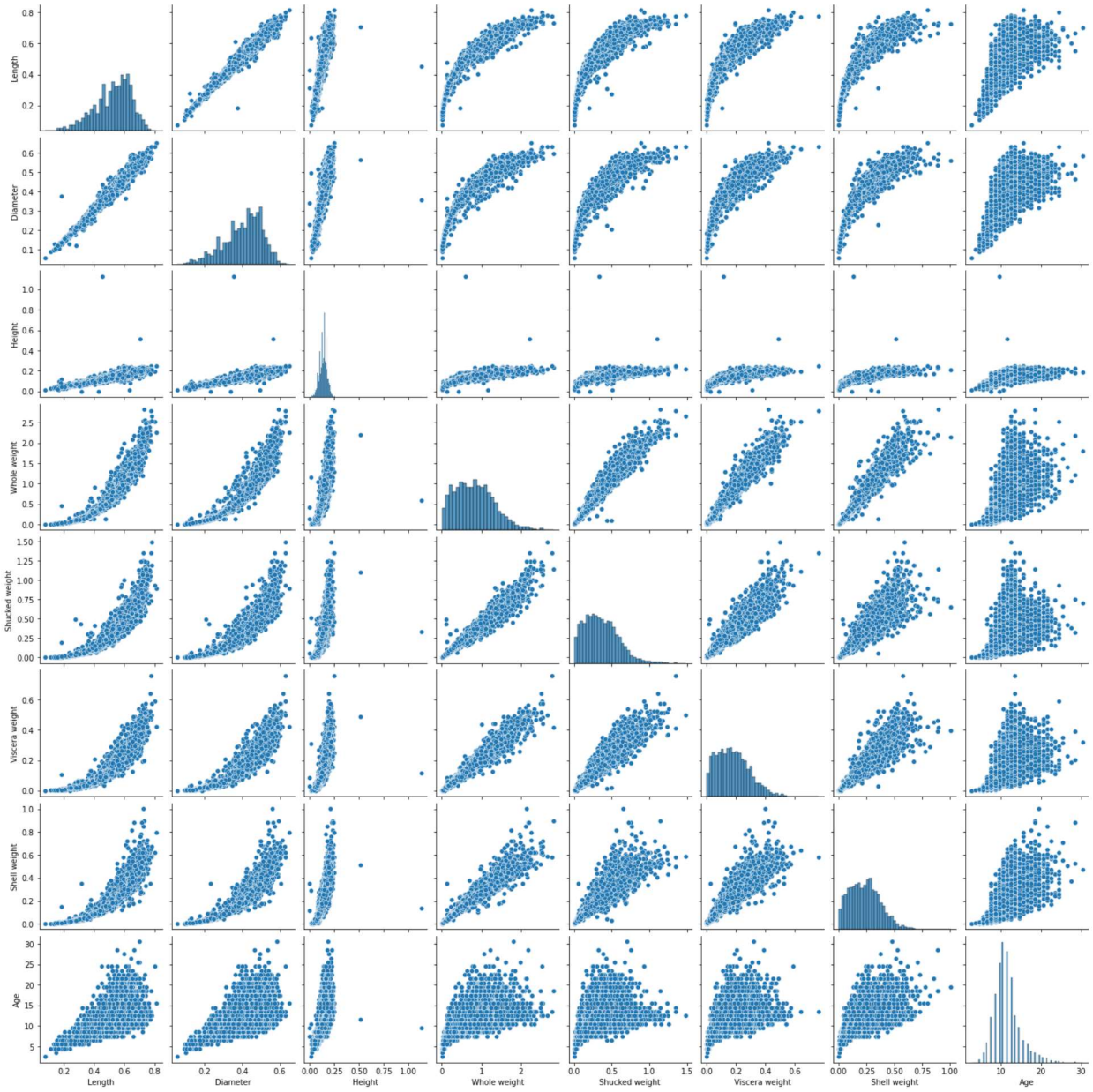


In [19]:
```
#countplot
sns.countplot(x=f1["Age"])
plt.show()
```

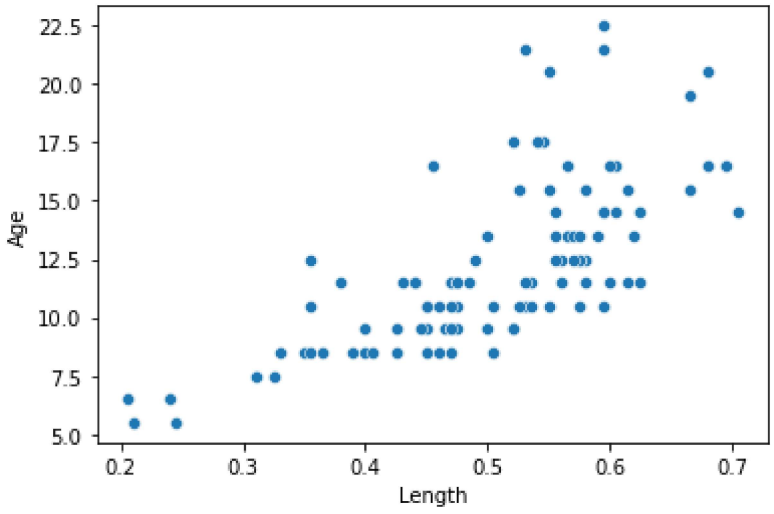


Bi-Variate Analysis

In [20]:
```
#pairplot
sns.pairplot(f1)
plt.show()
```

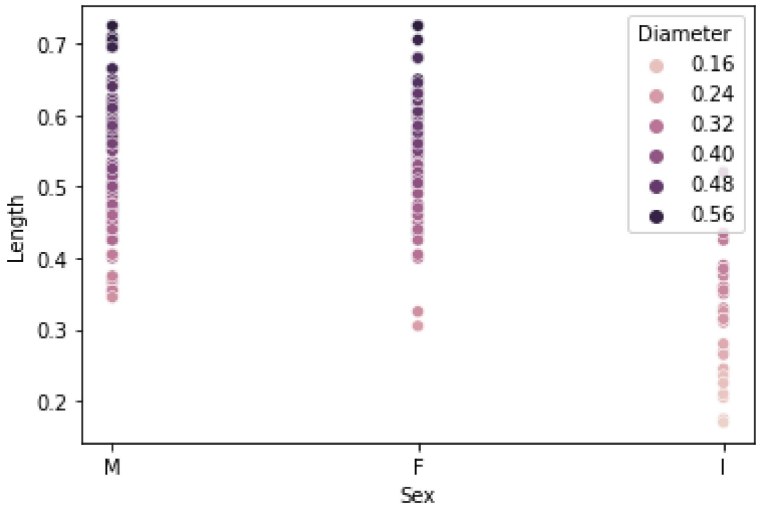


In [21]:
```
#scatterplot for Length and Age
sns.scatterplot(x=f1.iloc[:100,:]["Length"],y=f1.iloc[:100,:]["Age"])
plt.show()
```

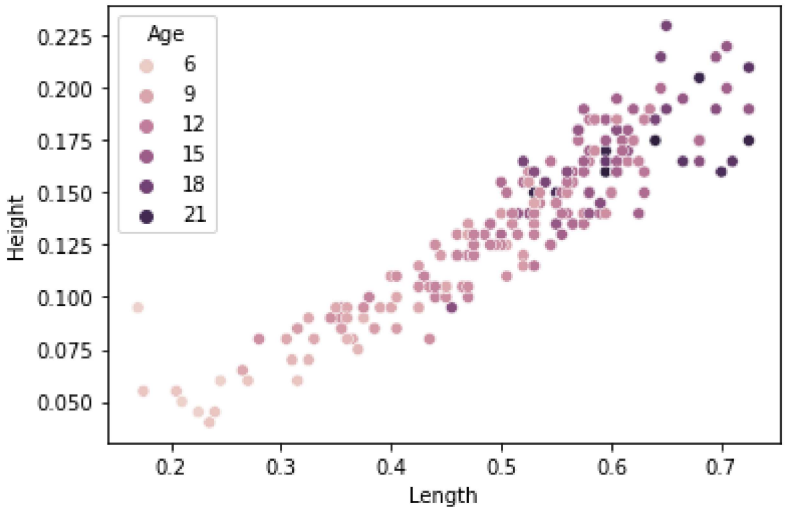


Multi-Variate Analysis

In [22]:
```
#scatterplot for Sex, Length and Diameter
sns.scatterplot(x=f1.iloc[:200,:]["Sex"],y=f1.iloc[:200,:]["Length"],hue=f1.iloc[:200,
plt.show()
```



In [23]:
```
#scatterplot for Length, Height and Age
sns.scatterplot(x=f1.iloc[:200,:]["Length"],y=f1.iloc[:200,:]["Height"],hue=f1.iloc[:2
plt.show()
```



4.Perform the descriptive statistics on the datase

In [24]:
```
f1.describe()
```

Out[24]:

	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	4
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 [25]:
```
f1.mode(numeric_only=True)
```

Out[25]:

	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 [26]:
```
f1.median(numeric_only=True)
```

Out[26]:
```
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 [27]:
```
f1.skew(numeric_only=True)
```

Out[27]:
```
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 [28]:
```
f1.kurt(numeric_only=True)
```

Out[28]:
```
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
```

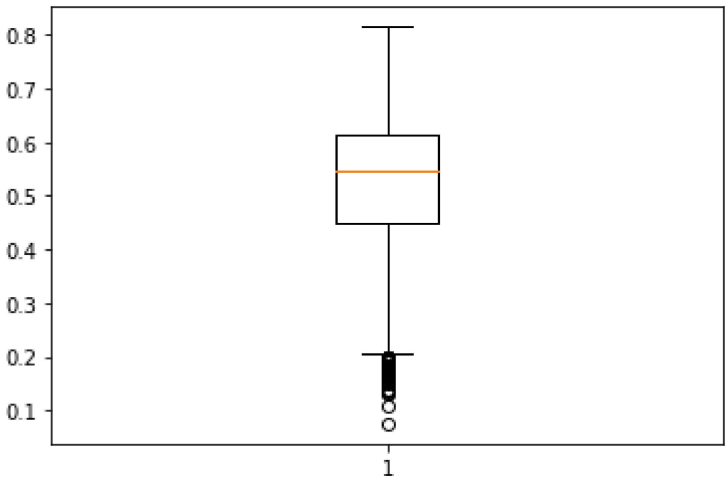
5. Handle the missing values

In [29]:
```
#find the null columns
f1.isnull().sum()
```

Out[29]:
```
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 [30]:
```
#find outliers
plt.boxplot(f1["Length"])
plt.show()
```



In [31]:
```
#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)
```

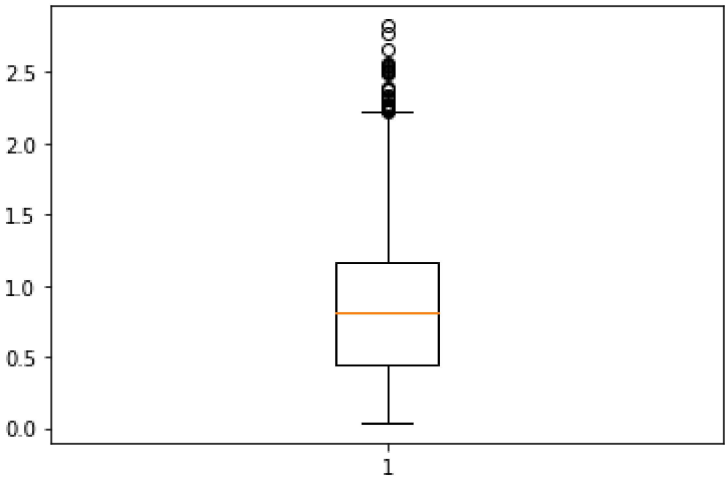
```
Q1:  0.45
Q3: 0.615
IQR:  0.16499999999999998
```

In [32]:
```
upperOutlayers=Q3+1.5*IQR
lowerOutlayers=Q1-1.5*IQR
print(upperOutlayers)
print(lowerOutlayers)
```

```
0.8624999999999999
0.20250000000000004
```

In [33]:
```
f1.drop(np.where(f1["Length"]>=upperOutlayers)[0],inplace=True)
f1.drop(np.where(f1["Length"]<=lowerOutlayers)[0],inplace=True)
```

In [34]:
```
#find outliers
plt.boxplot(f1["Whole weight"])
plt.show()
```



In [35]:
```
#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:  0.451
Q3: 1.1592500000000001
IQR:  0.70825
```

In [36]:
```
upperOutlayers=Q3+1.5*IQR
lowerOutlayers=Q1-1.5*IQR
print(upperOutlayers)
print(lowerOutlayers)
```

```
2.2216250000000004
-0.611375
```

In [37]:
```
f1.drop(np.where(f1["Whole weight"]>=upperOutlayers)[0],inplace=True)
f1.drop(np.where(f1["Whole weight"]<=lowerOutlayers)[0],inplace=True)
```

7.Check the categorical columns and perform encoding

In [38]:
```
f1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4099 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4099 non-null   object 
 1   Length          4099 non-null   float64
 2   Diameter        4099 non-null   float64
 3   Height          4099 non-null   float64
 4   Whole weight    4099 non-null   float64
 5   Shucked weight  4099 non-null   float64
 6   Viscera weight  4099 non-null   float64
 7   Shell weight    4099 non-null   float64
 8   Age             4099 non-null   float64
dtypes: float64(8), object(1)
memory usage: 320.2+ KB
```

In [39]:
```
from sklearn.preprocessing import LabelEncoder
f1["Sex"]=LabelEncoder().fit_transform(f1["Sex"])
print(f1["Sex"].unique())
```

```
[2 0 1]
```

In [40]:
```
f1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4099 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4099 non-null   int32  
 1   Length          4099 non-null   float64
 2   Diameter        4099 non-null   float64
 3   Height          4099 non-null   float64
 4   Whole weight    4099 non-null   float64
 5   Shucked weight  4099 non-null   float64
 6   Viscera weight  4099 non-null   float64
 7   Shell weight    4099 non-null   float64
 8   Age             4099 non-null   float64
dtypes: float64(8), int32(1)
memory usage: 304.2 KB
```

8.Split the dataset into ipdendent and dependent variables.

In [53]:
```
X=f1.iloc[:,:-4].values #independent variables
Y=f1.iloc[:,-4].values #dependent variable

X
```

Out[53]:
```
array([[2.    , 0.455 , 0.365 , 0.095 , 0.514 ],
       [2.    , 0.35  , 0.265 , 0.09  , 0.2255],
       [0.    , 0.53  , 0.42  , 0.135 , 0.677 ],
       ...,
       [2.    , 0.6   , 0.475 , 0.205 , 1.176 ],
       [0.    , 0.625 , 0.485 , 0.15  , 1.0945],
       [2.    , 0.71  , 0.555 , 0.195 , 1.9485]])
```

In [54]:
```
Y
```

Out[54]:
```
array([0.2245, 0.0995, 0.2565, ..., 0.5255, 0.531 , 0.9455])
```

9.Scale the independent variable

In [46]:
```
from sklearn.preprocessing import StandardScaler
std_scaler=StandardScaler().fit_transform(X)
std_scaler
```

Out[46]:
```
array([[ 1.14871275, -0.63674095, -0.48321942, ..., -0.62772927,
        -0.74834141, -0.6627956 ],
       [ 1.14871275, -1.55583249, -1.53757714, ..., -1.19721813,
        -1.23272027, -1.24632486],
       [-1.27463958,  0.01975301,  0.09667732, ..., -0.48194013,
        -0.37467771, -0.22514866],
       ...,
       [ 1.14871275,  0.63248071,  0.67657407, ...,  0.7435999 ,
         0.97235685,  0.48967468],
       [-1.27463958,  0.85131203,  0.78200984, ...,  0.76865741,
         0.72786085,  0.4021453 ],
       [ 1.14871275,  1.59533851,  1.52006024, ...,  2.65708247,
         1.79349436,  1.85367433]])
```

10.Split the data into training and testing.

In [49]:
```
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[49]:
```
(2869, 8)
```

In [50]:
```
X_test.shape
```

Out[50]:
```
(1230, 8)
```

11. Build the Model

In [55]:
```
#Linear Regression
from sklearn.linear_model import LinearRegression
reg= LinearRegression()
reg.fit(X_train, y_train)
```

Out[55]:
```
LinearRegression()
```

12. Train the Model

In [59]:
```
y_pred = reg.predict(X_test)
```

13. Test the model And

14. Measure the performance using Metrics

In [57]:
```
from sklearn import metrics
```

In [79]:
```
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): 1.5633101963051235
Mean squared error(MSE): 4.5383618102508
Intercept: 5.081086538611191
R2 Score: 0.5261530646361188
```