ASSIGNMENT 3

Assignment Date	21 /10/2022
Student Name	ANUVITHA G
Student Roll Number	61771921002
Maximum Marks	2 Marks

Abalone Age Prediction

Description: - Predicting the age of abalone from physical measurements. The age of abalone is determined by

cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring

and time-consuming task. Other measurements, which are easier to obtain, are used to predict age. Further

information, such as weather patterns and locations (hence food availability) may be required to solve the

problem.

Attribute Information:

Given is the attribute name, attribute type, measurement unit, and a brief description. The number of

rings is the value to predict: either as a continuous value or as a classification problem.

Name / Data Type / Measurement Unit / Description

1- Sex / nominal / -- / M, F, and I (infant)

2- Length / continuous / mm / Longest shell measurement

3- Diameter / continuous / mm / perpendicular to length

4- Height / continuous / mm / with meat in shell

5- Whole weight / continuous / grams / whole abalone

6- Shucked weight / continuous / grams / weight of meat

7- Viscera weight / continuous / grams / gut weight (after bleeding)

8- Shell weight / continuous / grams / after being dried

9- Rings / integer / -- / +1.5 gives the age in years

Building a Regression Model

1. Download the dataset:

2. Load the dataset into the tool.

3. Perform Below Visualizations.

· Univariate Analysis

· Bi-Variate Analysis

· Multi-Variate Analysis

4. Perform descriptive statistics on the dataset.

5. Check for Missing values and deal with them.

6. Find the outliers and replace them outliers

7. Check for Categorical columns and perform encoding.

8. Split the data into dependent and independent variables.

9. Scale the independent variables

10. Split the data into training and testing

11. Build the Model

12. Train the Model

13. Test the Model

14. Measure the performance using Metrics.

SOLUTION:

1.

Downloading dataset

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

Choose Files No file chosen 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
```

2.

Loading dataset

```
ab=pd.read_csv('abalone.csv')
ab.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

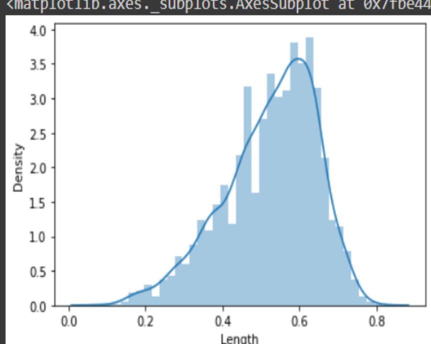
3.

Visualizations

Univariate Analysis

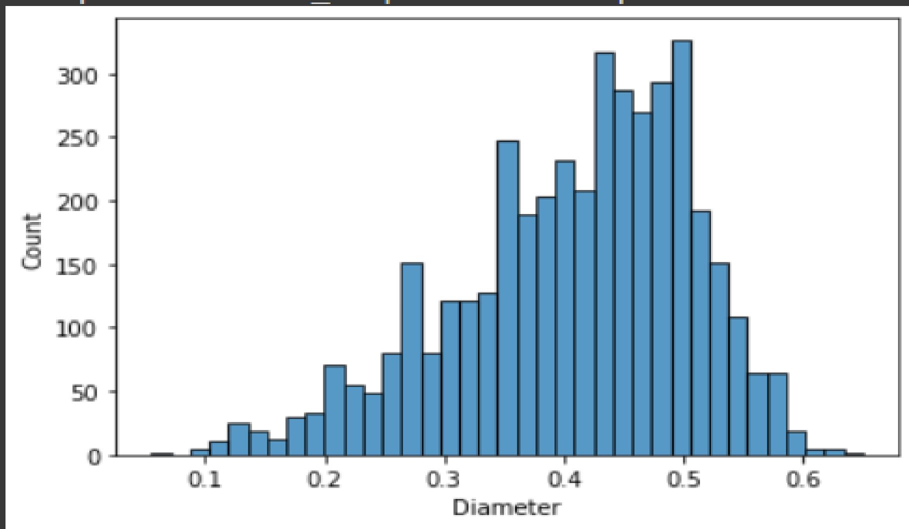
```
sb.distplot(ab.Length)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe44b41710>
```



```
sb.histplot(ab.Diameter)
```

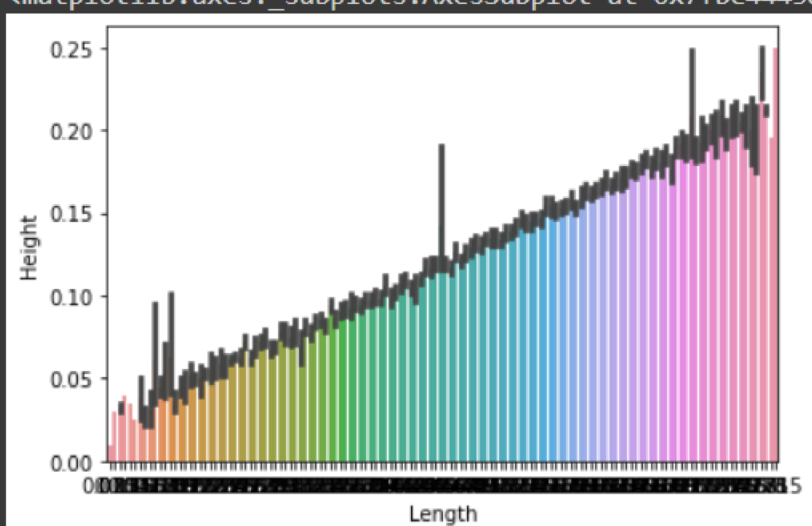
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe4452a7d0>
```



Bivariate Analysis

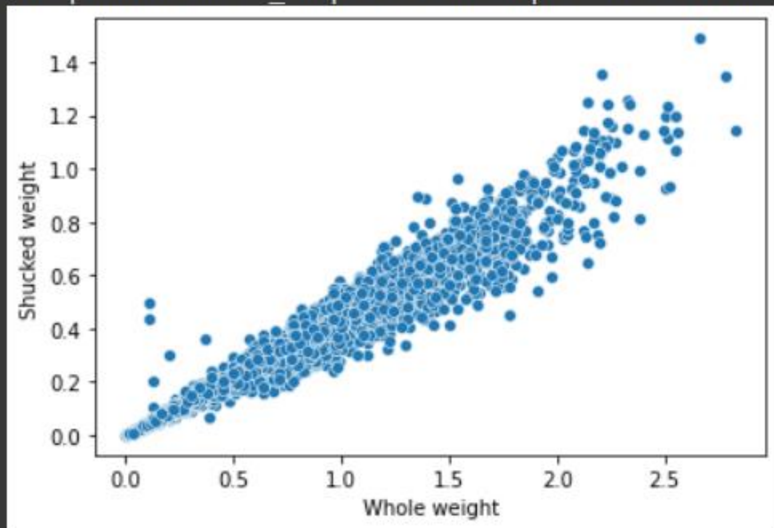
```
sb.barplot(ab.Length,ab.Height)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe44438e90>
```



```
sb.scatterplot(data=ab, x="Whole weight", y="Shucked weight")
```

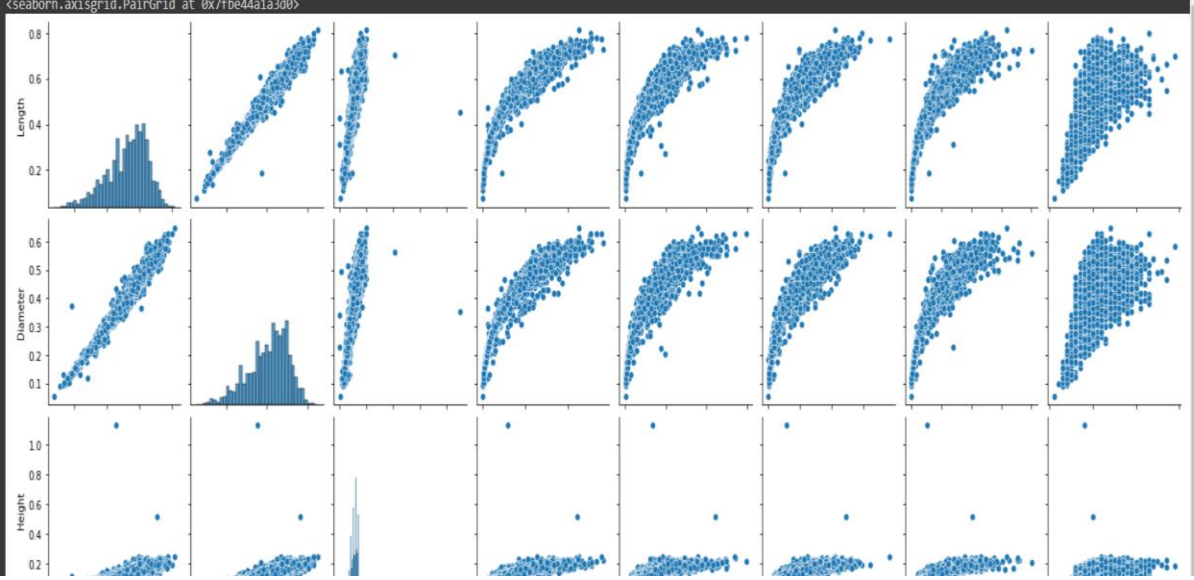
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe43e4ac50>
```

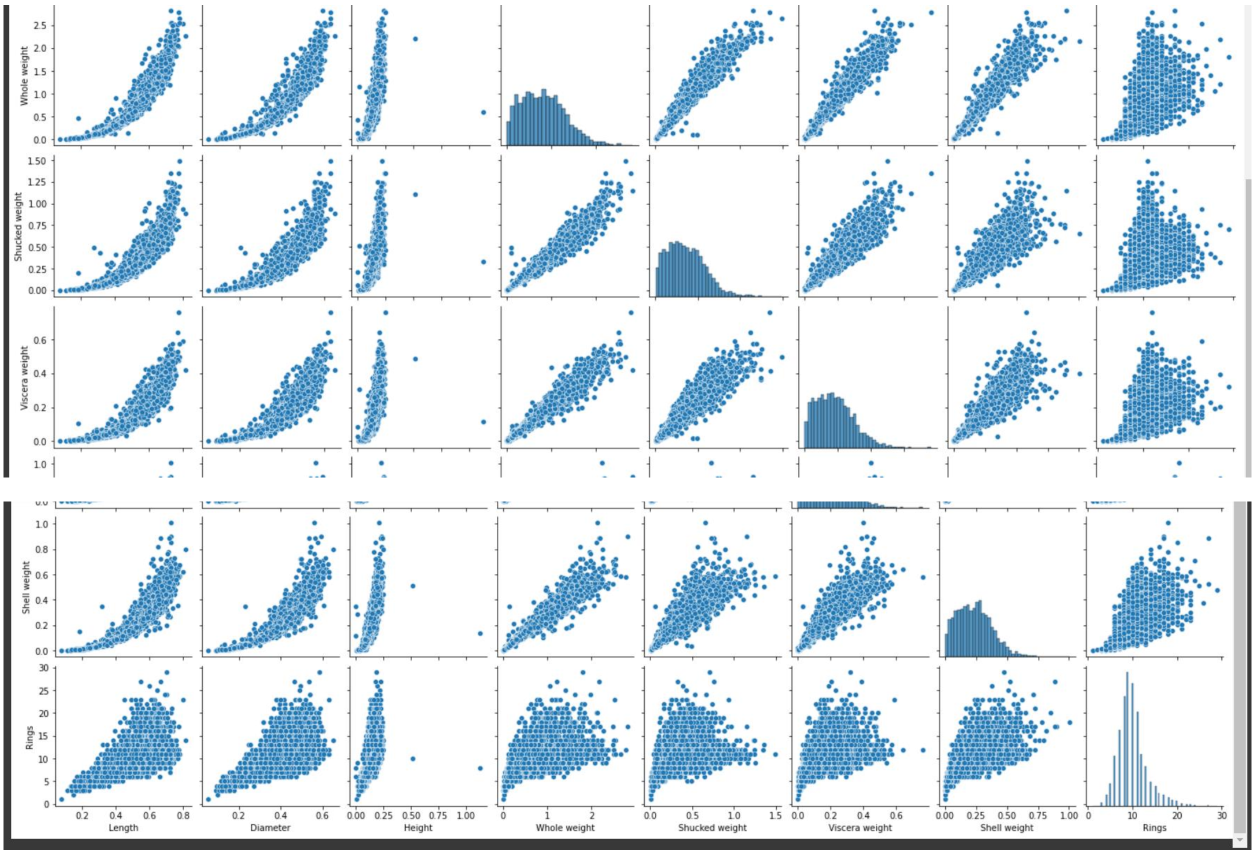


Multivariate Analysis

```
sb.pairplot(ab)
```

```
<seaborn.axisgrid.PairGrid at 0x7fbe44a1a3d0>
```





5.

Descriptive statistics

+ Code   + Text

```
[ ]  ab.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

```
] ab.mean()

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in DataFrame rec
  """Entry point for launching an IPython kernel.
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
```

```
ab.median()

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in DataFrame rec
  """Entry point for launching an IPython kernel.
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
```

```
[ ] ab.mode()
```

	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

6.

Checking for Missing values

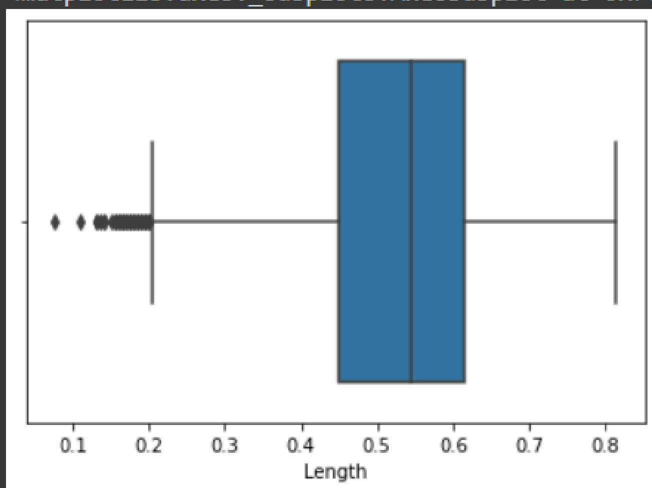
```
ab.isnull().any()

Sex               False
Length            False
Diameter          False
Height            False
Whole weight      False
Shucked weight    False
Viscera weight    False
Shell weight      False
Rings             False
dtype: bool
```

7.

Finding the outliers and replacing them

```
sb.boxplot(ab.Length)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: F
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe3f686290>
```

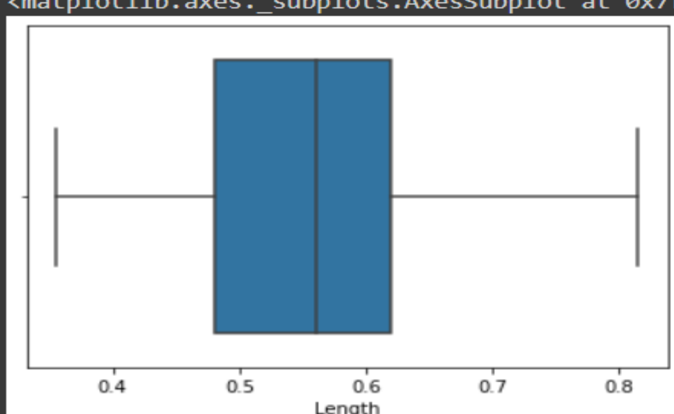



```
a=ab.Length.quantile(0.1)
a
```

```
0.355
```

```
ab=ab[ab.Length>=a]
sb.boxplot(ab.Length)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe3ddd3a10>
```



8.

Checking for Categorical columns and performing encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
ab.Sex=le.fit_transform(ab.Sex)
ab.head()
```

	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
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
5	1	0.425	0.300	0.095	0.3515	0.1410	0.0775	0.120	8
6	0	0.530	0.415	0.150	0.7775	0.2370	0.1415	0.330	20

9.

Spliting the data

```
x=ab.iloc[:,:-1]
x.head()
```

	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
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
5	1	0.425	0.300	0.095	0.3515	0.1410	0.0775	0.120
6	0	0.530	0.415	0.150	0.7775	0.2370	0.1415	0.330

```
[ ] y=ab.iloc[:,-1]
y.head()
```

```
0    15
2     9
3    10
5     8
6    20
Name: Rings, dtype: int64
```

10.

Scaling the independent variables

```
from sklearn.preprocessing import scale
scale=pd.DataFrame(scale(x),columns=x.columns)
scale.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	1.124209	-1.025403	-0.829824	-1.424341	-0.860909	-0.808566	-0.939748	-0.857452
1	-1.222589	-0.217386	-0.119841	-0.328202	-0.502083	-0.654555	-0.544014	-0.391780
2	1.124209	-1.187006	-0.829824	-0.602237	-0.856506	-0.851881	-0.812723	-0.818646
3	-0.049190	-1.348609	-1.668896	-1.424341	-1.218635	-1.210437	-1.169372	-1.090287
4	-1.222589	-0.217386	-0.184384	0.082850	-0.280843	-0.748405	-0.544014	0.539562

11.

Spliting the data into training and testing

```
[ ] x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
[ ] x_train.shape
```

```
(2638, 8)
```

```
[ ] x_test.shape
```

```
(1131, 8)
```

```
[ ] y_train.shape
```

```
(2638,)
```

```
y_test.shape
```

```
(1131,)
```

12.

Building the Model

```
[ ] from sklearn.linear_model import LinearRegression
    model=LinearRegression()
```

13.

Training the Model

```
[ ] model.fit(x_train,y_train)
    LinearRegression()

    LinearRegression()
```

14.

Testing the Model

```
[ ] p=model.predict(x_test)
    p

    array([ 7.81182283,  7.55408012, 10.85103291, ...,  9.6826651 ,
           11.54475891, 10.66776086])
```

15.

Measuring the performance using Metrics.

```
from sklearn.metrics import mean_squared_error
import math
print(math.sqrt(mean_squared_error(y_test,p)))

2.2683508786406725
```