

## Assignment-3

Assignment Date	07 October 2022
Team ID	PNT2022TMID46353
Student Name	Santhini Devi.S
Student Roll Number	820319104036
Maximum Marks	2 Marks

[1]

```
from google.colab import drive
```

[2]

```
drive.mount('/content/drive')
```

Mounted at /content/drive

[5]

```
import pandas as pd
```

[9]

```
data = pd.read_csv('/content/drive/MyDrive/abalone.csv')
```

[11]

```
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

[12]

```
import pandas as pd
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
sns.set_style('darkgrid')
```

```
sns.set(font_scale=1.3)
```

```
data = pd.read_csv('/content/drive/MyDrive/abalone.csv')
```

[13]

```
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

```
[14]
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
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
```

```
[15]
```

```
data.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
<b>count</b>	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
<b>mean</b>	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
<b>std</b>	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
<b>min</b>	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
<b>25%</b>	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
<b>50%</b>	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
<b>75%</b>	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
<b>max</b>	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

[16]

*#univariate analysis*

cols = 3

rows = 3

num\_cols = data.select\_dtypes(exclude='object').columns

fig = plt.figure( figsize=(cols\*5, rows\*5))

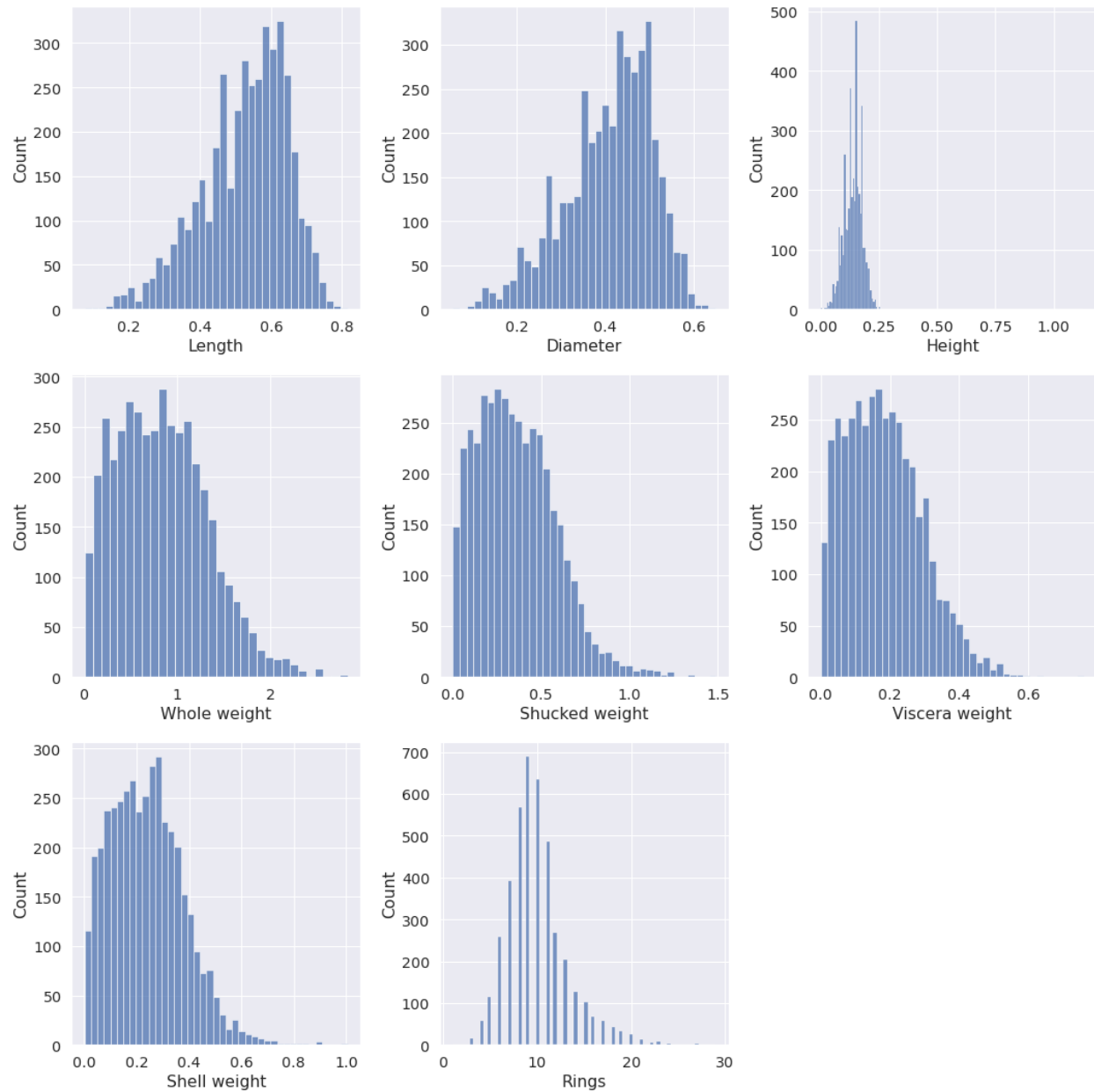
**for** i, col **in** enumerate(num\_cols):

    ax=fig.add\_subplot(rows,cols,i+1)

    sns.histplot(x = data[col], ax = ax)

fig.tight\_layout()

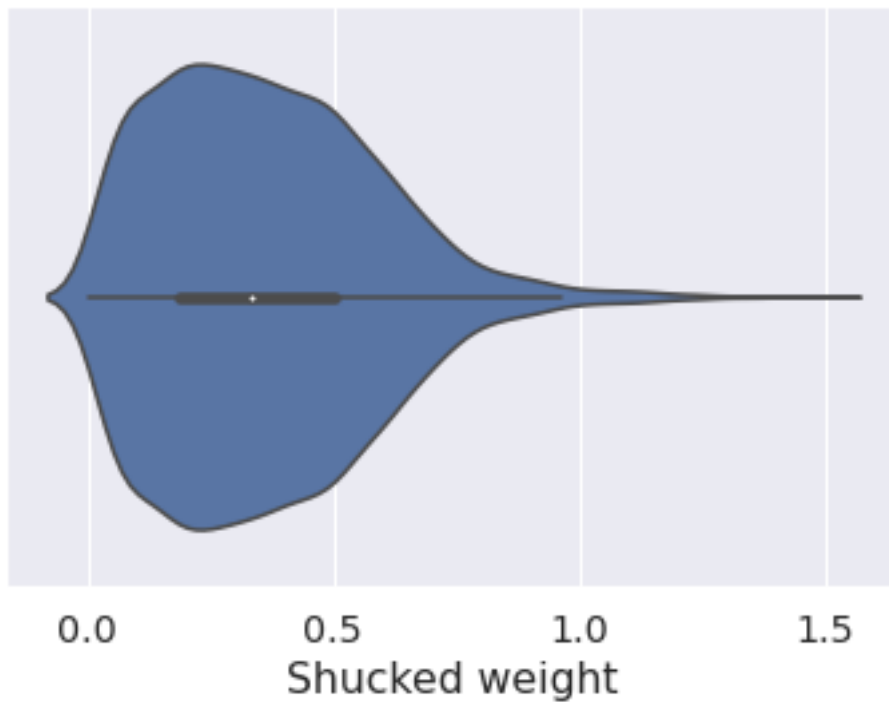
plt.show()



[17]

```
sns.violinplot(x=data["Shucked weight"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5593102910>
```



[18]

*#Bivariate analysis*

**import** matplotlib.pyplot **as** plt

*#create scatterplot of hours vs. score*

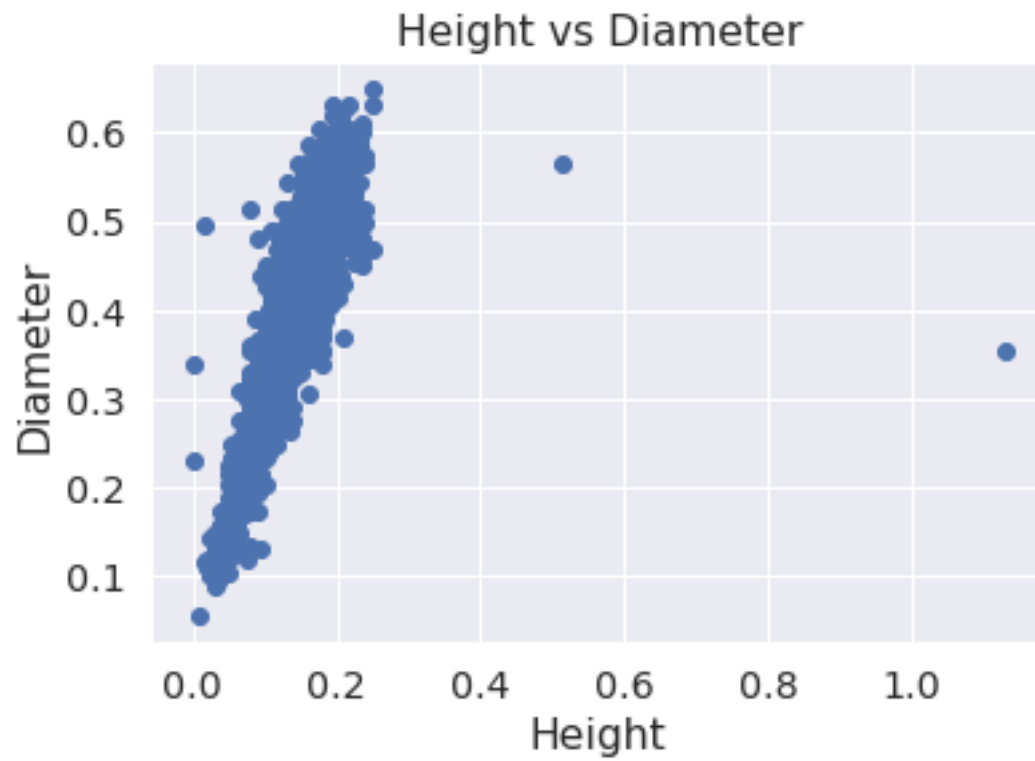
plt.scatter(data.Height, data.Diameter)

plt.title('Height vs Diameter')

plt.xlabel('Height')

plt.ylabel('Diameter')

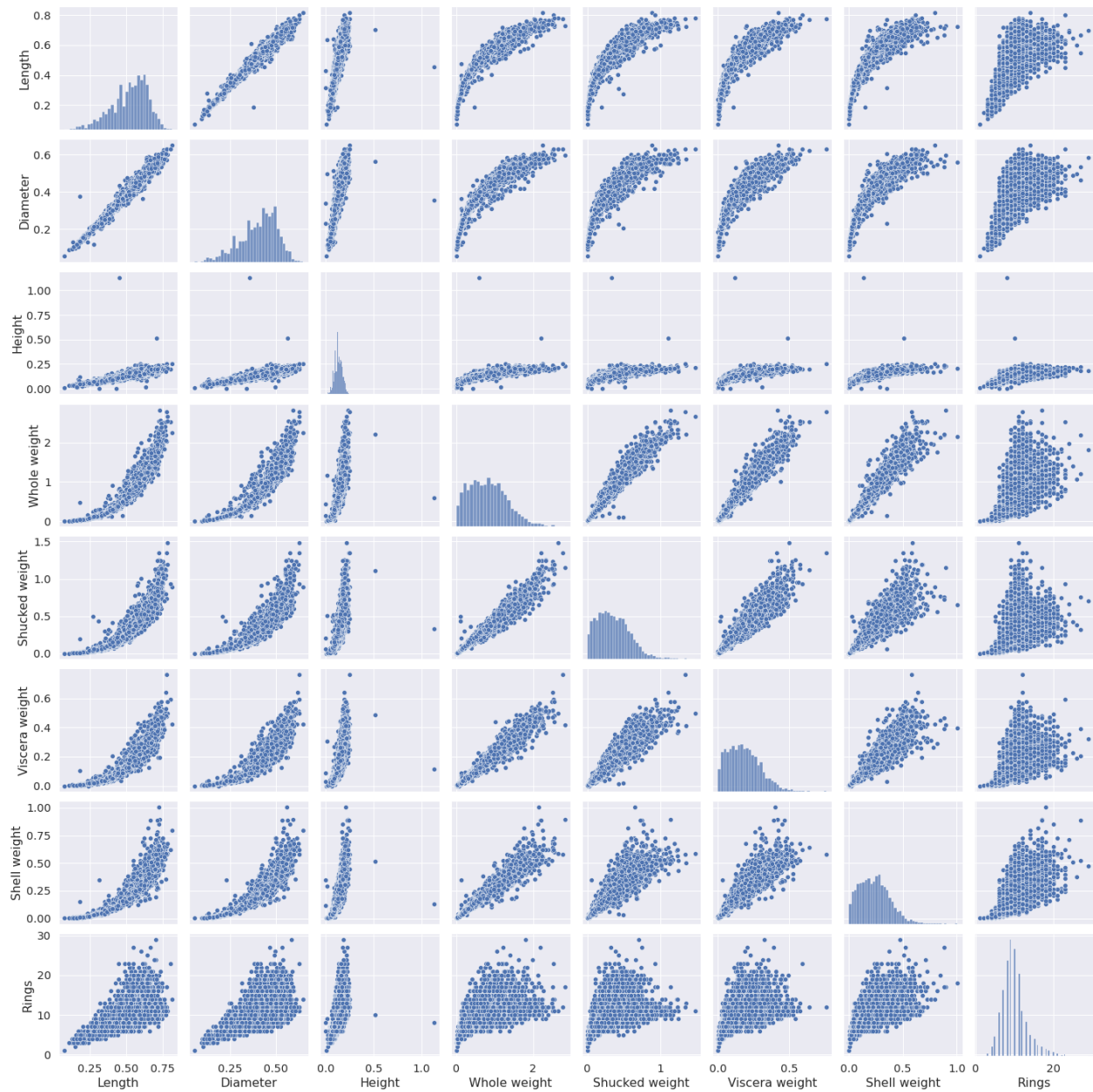
Text(0, 0.5, 'Diameter')



[19]

*#multivariate analysis*

`sns.pairplot(data);`



[20]

```
data.mean()
```

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric\_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

"""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
```

```
[21]
```

```
data.median()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise TypeError.
Select only valid columns before calling the reduction.
```

```
"""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
```

```
[22]
```

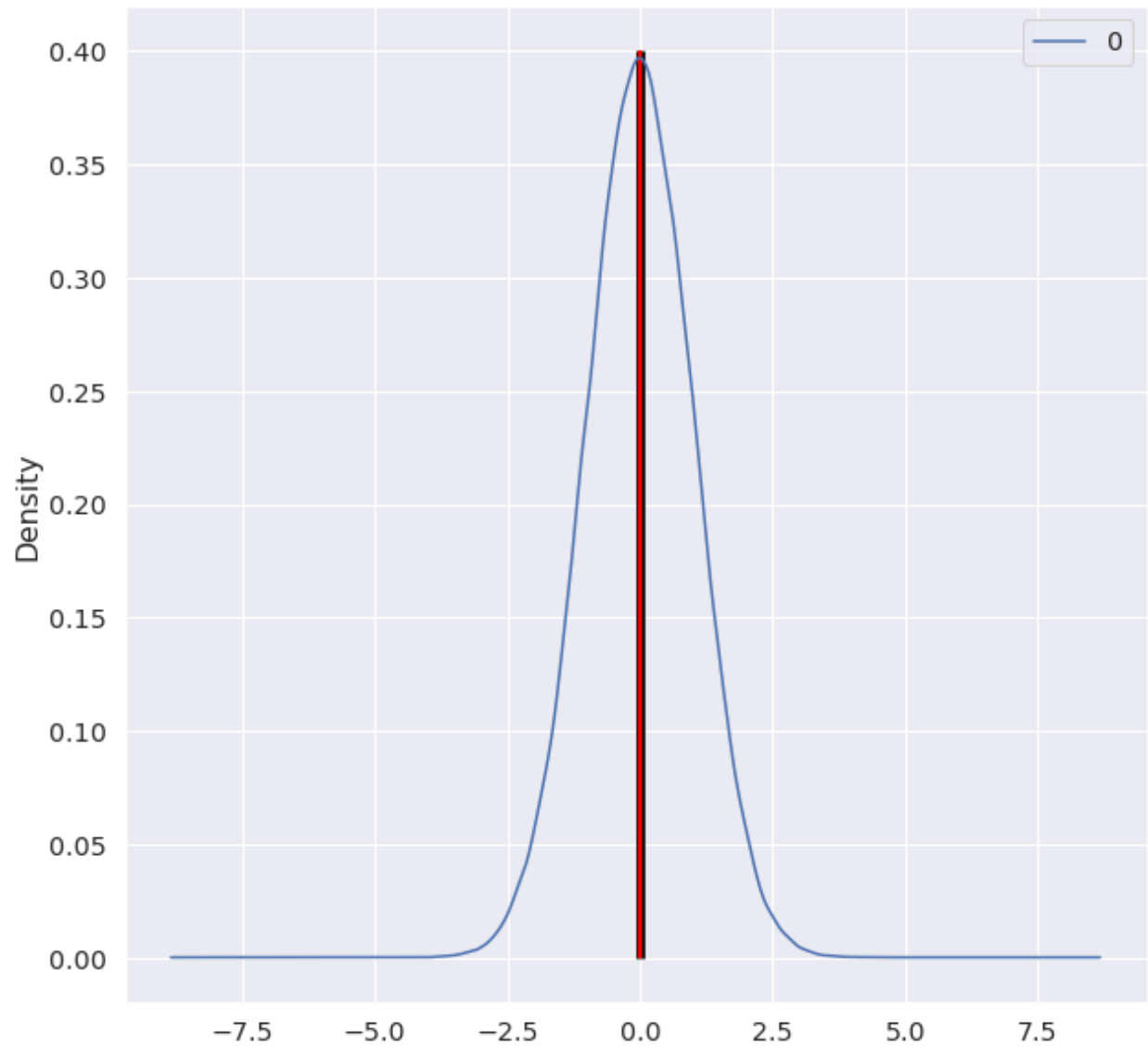
```
norm_data = pd.DataFrame(np.random.normal(size=100000))
```

```
norm_data.plot(kind="density",
                figsize=(10,10));
```

```
plt.vlines(norm_data.mean(),    # Plot black line at mean
            ymin=0,
            ymax=0.4,
            linewidth=5.0);
```

```
plt.vlines(norm_data.median(), # Plot red line at median
            ymin=0,
            ymax=0.4,
            linewidth=2.0,
            color="red");
```





### Identifying the Outliers

```
[23]
```

```
print(data['Shell weight'].skew())
```

```
data['Shell weight'].describe()
```

```
0.6209268251392077
```

```
count    4177.000000
```

```
mean      0.238831
```

```
std       0.139203
```

```
min       0.001500
```

```
25%      0.130000
```

```
50%      0.234000
```

```
75%      0.329000
```

```
max      1.005000
Name: Shell weight, dtype: float64
```

## Replacing the Outliers

```
[24]
print(data['Shell weight'].quantile(0.50))
print(data['Shell weight'].quantile(0.95))
data['Shell weight'] = np.where(data['Shell weight'] > 325, 140, data['Shell weight'])
data.describe()
0.234
0.48
```

	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

## Perform Encoding

```
[25]
from sklearn.compose import make_column_selector as selector

categorical_columns_selector = selector(dtype_include=object)
categorical_columns = categorical_columns_selector(data)
categorical_columns

['Sex']
[26]
data_categorical = data[categorical_columns]
data_categorical.head()
[27]
from sklearn import preprocessing
```

Label\_encoder object knows how to understand word labels.

```
[28]  
label_encoder = preprocessing.LabelEncoder()
```

Encode labels in column 'Species'.

```
[29]  
data['Sex']= label_encoder.fit_transform(data['Sex'])  
data['Sex'].unique()  
array([2, 0, 1])  
[30]  
X= data.iloc[ : , :-1].values  
  
y= data.iloc[ : , 4].values  
print(X,y)  
[[2.    0.455  0.365  ... 0.2245 0.101  0.15  ]  
 [2.    0.35   0.265  ... 0.0995 0.0485 0.07  ]  
 [0.    0.53   0.42   ... 0.2565 0.1415 0.21  ]  
 ...  
 [2.    0.6    0.475  ... 0.5255 0.2875 0.308  ]  
 [0.    0.625  0.485  ... 0.531  0.261  0.296  ]  
 [2.    0.71   0.555  ... 0.9455 0.3765 0.495  ]] [0.514  0.2255 0.677  ... 1.176  
1.0945 1.9485]
```

Import Packages

```
[31]  
import numpy as np  
import pandas as pd  
from sklearn.model_selection import train_test_split
```

Importing Data

```
[32]  
print(data.shape)  
(4177, 9)
```

Head of the Data

```
[33]  
print('Head of the dataframe : ')  
print(data.head())  
  
print(data.columns)
```

```
X= data['Whole weight']
y=data['Shucked weight']
```

Head of the dataframe :

	Sex	Length	Diameter	Height	Whole weight	Shucked weight \
0	2	0.455	0.365	0.095	0.5140	0.2245
1	2	0.350	0.265	0.090	0.2255	0.0995
2	0	0.530	0.420	0.135	0.6770	0.2565
3	2	0.440	0.365	0.125	0.5160	0.2155
4	1	0.330	0.255	0.080	0.2050	0.0895

	Viscera weight	Shell weight	Rings
0	0.1010	0.150	15
1	0.0485	0.070	7
2	0.1415	0.210	9
3	0.1140	0.155	10
4	0.0395	0.055	7

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

### Using the train test split function

```
[34]  
X_train, X_test, y_train, y_test = train_test_split(  
X,y , random_state=104,test_size=0.25, shuffle=True)
```

### Printing out train and test sets

```
[35]  
print('X_train : ')  
print(X_train.head())  
print(X_train.shape)
```

```
print("")  
print('X_test : ')  
print(X_test.head())  
print(X_test.shape)
```

```
print("")  
print('y_train : ')  
print(y_train.head())
```

```
print(y_train.shape)
```

```
print("")  
print('y_test : ')  
print(y_test.head())  
print(y_test.shape)
```

**X\_train :**

```
437    0.2520  
1331    0.8730  
1611    0.7625  
1394    1.5210  
396     0.7155  
Name: Whole weight, dtype: float64  
(3132,)
```

**X\_test :**

```
4087    0.9840  
1699    1.4890  
1868    0.6965  
2984    1.2240  
5        0.3515  
Name: Whole weight, dtype: float64  
(1045,)
```

**y\_train :**

```
437    0.0915  
1331    0.3820  
1611    0.3270  
1394    0.6440  
396     0.3165  
Name: Shucked weight, dtype: float64  
(3132,)
```

**y\_test :**

```
4087    0.4865  
1699    0.7150  
1868    0.3045  
2984    0.6180  
5        0.1410
```

Name: Shucked weight, dtype: float64  
(1045,)

## Scaling

[36]

```
data_scaled = data.copy()
col_names = ['Shucked weight', 'Whole weight']
features = data_scaled[col_names]
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_scaled[col_names] = scaler.fit_transform(features.values)
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(5, 10))
```

```
data_scaled[col_names] = scaler.fit_transform(features.values)
data_scaled
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	2	0.455	0.365	0.095	5.906676	5.751513	0.1010	0.1500	15
1	2	0.350	0.265	0.090	5.395785	5.331204	0.0485	0.0700	7
2	0	0.530	0.420	0.135	6.195325	5.859112	0.1415	0.2100	9
3	2	0.440	0.365	0.125	5.910218	5.721251	0.1140	0.1550	10
4	1	0.330	0.255	0.080	5.359483	5.297579	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	0	0.565	0.450	0.165	6.567204	6.240753	0.2390	0.2490	11
4173	2	0.590	0.440	0.135	6.707101	6.472764	0.2145	0.2605	10
4174	2	0.600	0.475	0.205	7.078980	6.763618	0.2875	0.3080	9
4175	0	0.625	0.485	0.150	6.934656	6.782112	0.2610	0.2960	10
4176	2	0.710	0.555	0.195	8.446963	8.175857	0.3765	0.4950	12

4177 rows × 9 columns

## Testing and Training

[37]

```
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
```

## Split the Dataset

[38]

```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, test_size=0.05, random_state=0)  
print(X_train, X_test, y_train, y_test)
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight \
678	0	0.450	0.380	0.165	0.8165	0.2500
3009	1	0.255	0.185	0.065	0.0740	0.0305
1906	1	0.575	0.450	0.135	0.8245	0.3375
768	0	0.550	0.430	0.155	0.7850	0.2890
2781	2	0.595	0.475	0.140	1.0305	0.4925
...	...	...	...	...	...	...
1033	2	0.650	0.525	0.185	1.6220	0.6645
3264	0	0.655	0.500	0.140	1.1705	0.5405
1653	2	0.595	0.450	0.145	0.9590	0.4630
2607	0	0.625	0.490	0.165	1.1270	0.4770
2732	1	0.410	0.325	0.110	0.3260	0.1325

	Viscera weight	Shell weight
678	0.1915	0.2650
3009	0.0165	0.0200
1906	0.2115	0.2390
768	0.2270	0.2330
2781	0.2170	0.2780
...	...	...
1033	0.3225	0.4770
3264	0.3175	0.2850
1653	0.2065	0.2535
2607	0.2365	0.3185
2732	0.0750	0.1010

[3968 rows x 8 columns]

	Sex	Length	Diameter	Height	Whole weight	Shucked weight \
668	2	0.550	0.425	0.155	0.9175	0.2775
1580	1	0.500	0.400	0.120	0.6160	0.2610
3784	2	0.620	0.480	0.155	1.2555	0.5270
463	1	0.220	0.165	0.055	0.0545	0.0215
2615	2	0.645	0.500	0.175	1.5105	0.6735
...	...	...	...	...	...	...

1670	0	0.610	0.485	0.150	1.2405	0.6025
3055	0	0.610	0.495	0.160	1.0890	0.4690
3366	2	0.280	0.210	0.065	0.0905	0.0350
1410	0	0.665	0.530	0.180	1.4910	0.6345
4035	1	0.520	0.410	0.140	0.5995	0.2420

	Viscera weight	Shell weight
668	0.2430	0.3350
1580	0.1430	0.1935
3784	0.3740	0.3175
463	0.0120	0.0200
2615	0.3755	0.3775
...	...	...
1670	0.2915	0.3085
3055	0.1980	0.3840
3366	0.0200	0.0300
1410	0.3420	0.4350
4035	0.1375	0.1820

[209 rows x 8 columns] 678 23

3009	4
1906	11
768	11
2781	10

..	
1033	10
3264	12
1653	10
2607	9
2732	8

Name: Rings, Length: 3968, dtype: int64 668 13

1580	8
3784	11
463	5
2615	12

..	
1670	12
3055	11
3366	5
1410	10



4035 11

Name: Rings, Length: 209, dtype: int64

[39]

**from** sklearn.linear\_model **import** LogisticRegression

logreg= LogisticRegression()

logreg.fit(X\_train,y\_train)

y\_pred=logreg.predict(X\_test)

print (X\_test) *#test dataset*

print (y\_pred) *#predicted values*

	Sex	Length	Diameter	Height	Whole weight	Shucked weight \
668	2	0.550	0.425	0.155	0.9175	0.2775
1580	1	0.500	0.400	0.120	0.6160	0.2610
3784	2	0.620	0.480	0.155	1.2555	0.5270
463	1	0.220	0.165	0.055	0.0545	0.0215
2615	2	0.645	0.500	0.175	1.5105	0.6735
...	...	...	...	...	...	...
1670	0	0.610	0.485	0.150	1.2405	0.6025
3055	0	0.610	0.495	0.160	1.0890	0.4690
3366	2	0.280	0.210	0.065	0.0905	0.0350
1410	0	0.665	0.530	0.180	1.4910	0.6345
4035	1	0.520	0.410	0.140	0.5995	0.2420

	Viscera weight	Shell weight
668	0.2430	0.3350
1580	0.1430	0.1935
3784	0.3740	0.3175
463	0.0120	0.0200
2615	0.3755	0.3775
...	...	...
1670	0.2915	0.3085
3055	0.1980	0.3840
3366	0.0200	0.0300
1410	0.3420	0.4350
4035	0.1375	0.1820

[209 rows x 8 columns]

[10 9 10 6 10 10 8 8 7 10 8 6 8 9 6 9 7 10 10 8 7 7 8 6  
9 10 5 10 9 10 7 5 10 11 8 9 8 9 8 10 10 8 9 10 10 8 9 10  
9 7 9 7 8 10 8 8 6 7 7 7 10 9 10 8 7 10 11 10 9 11 8 11

```

10 9 9 9 9 9 10 8 9 6 7 10 9 8 9 11 5 7 9 9 8 8 10 6
8 7 10 11 10 8 10 9 6 10 8 7 7 10 11 11 9 9 10 11 10 10 6 9
7 9 7 8 10 10 10 11 7 10 10 8 9 10 11 10 9 9 7 10 9 8 7 9
10 9 8 8 8 6 8 11 7 10 7 7 9 7 10 7 9 7 9 10 6 10 7 6
9 9 9 6 8 10 10 10 6 8 7 10 9 9 9 9 10 9 9 8 6 8 9 8
8 9 10 8 7 9 5 11 8 9 11 9 10 10 6 11 8]

```

/usr/local/lib/python3.7/dist-packages/sklearn/linear\_model/\_logistic.py:818:  
ConvergenceWarning: lbfgs failed to converge (status=1):  
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](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG,

[40]

[X\\_train](#)

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
<b>678</b>	0	0.450	0.380	0.165	0.8165	0.2500	0.1915	0.2650
<b>3009</b>	1	0.255	0.185	0.065	0.0740	0.0305	0.0165	0.0200
<b>1906</b>	1	0.575	0.450	0.135	0.8245	0.3375	0.2115	0.2390
<b>768</b>	0	0.550	0.430	0.155	0.7850	0.2890	0.2270	0.2330
<b>2781</b>	2	0.595	0.475	0.140	1.0305	0.4925	0.2170	0.2780
...	...	...	...	...	...	...	...	...
<b>1033</b>	2	0.650	0.525	0.185	1.6220	0.6645	0.3225	0.4770
<b>3264</b>	0	0.655	0.500	0.140	1.1705	0.5405	0.3175	0.2850
<b>1653</b>	2	0.595	0.450	0.145	0.9590	0.4630	0.2065	0.2535
<b>2607</b>	0	0.625	0.490	0.165	1.1270	0.4770	0.2365	0.3185
<b>2732</b>	1	0.410	0.325	0.110	0.3260	0.1325	0.0750	0.1010

3968 rows × 8 columns

[41]

X\_test

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
668	2	0.550	0.425	0.155	0.9175	0.2775	0.2430	0.3350
1580	1	0.500	0.400	0.120	0.6160	0.2610	0.1430	0.1935
3784	2	0.620	0.480	0.155	1.2555	0.5270	0.3740	0.3175
463	1	0.220	0.165	0.055	0.0545	0.0215	0.0120	0.0200
2615	2	0.645	0.500	0.175	1.5105	0.6735	0.3755	0.3775
...	...	...	...	...	...	...	...	...
1670	0	0.610	0.485	0.150	1.2405	0.6025	0.2915	0.3085
3055	0	0.610	0.495	0.160	1.0890	0.4690	0.1980	0.3840
3366	2	0.280	0.210	0.065	0.0905	0.0350	0.0200	0.0300
1410	0	0.665	0.530	0.180	1.4910	0.6345	0.3420	0.4350
4035	1	0.520	0.410	0.140	0.5995	0.2420	0.1375	0.1820
209 rows × 8 columns								

[42]

y\_train

678 23  
3009 4  
1906 11  
768 11  
2781 10

..  
1033 10  
3264 12  
1653 10  
2607 9  
2732 8

Name: Rings, Length: 3968, dtype: int64

[43]

y\_test

668 13  
1580 8  
3784 11  
463 5

```
2615 12
```

```
..  
1670 12
```

```
3055 11
```

```
3366 5
```

```
1410 10
```

```
4035 11
```

```
Name: Rings, Length: 209, dtype: int64
```

```
[44]
```

```
# Select algorithm
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.metrics import accuracy_score
```

```
model = DecisionTreeClassifier()
```

```
# Fit model to the data
```

```
model.fit(X_train, y_train)
```

```
# Check model performance on training data
```

```
predictions = model.predict(X_train)
```

```
print(accuracy_score(y_train, predictions))
```

```
1.0
```

```
[45]
```

```
# Evaluate the model on the test data
```

```
predictions = model.predict(X_test)
```

```
predictions
```

```
array([19, 12, 14, 5, 10, 13, 8, 8, 8, 9, 8, 5, 8, 9, 6, 13, 10,  
       15, 9, 8, 6, 7, 9, 6, 11, 11, 4, 17, 14, 11, 9, 4, 11, 18,  
       6, 8, 9, 9, 7, 11, 12, 8, 15, 11, 13, 8, 9, 10, 17, 7, 9,  
       6, 9, 17, 10, 6, 7, 7, 6, 6, 9, 8, 9, 7, 7, 13, 10, 13,  
       12, 10, 10, 14, 10, 9, 10, 9, 10, 8, 9, 8, 9, 6, 6, 10, 12,  
       9, 8, 15, 6, 10, 8, 8, 8, 6, 17, 5, 9, 9, 10, 10, 11, 13,  
       10, 12, 5, 11, 8, 6, 10, 20, 10, 11, 9, 9, 16, 9, 9, 12, 5,  
       9, 7, 14, 8, 11, 13, 9, 13, 13, 6, 9, 9, 8, 9, 11, 10, 10,  
       10, 8, 6, 18, 14, 12, 6, 8, 12, 8, 9, 7, 7, 6, 8, 13, 8,  
       8, 9, 8, 15, 7, 10, 7, 9, 10, 9, 9, 6, 20, 7, 6, 10, 11,  
       10, 3, 6, 10, 21, 11, 6, 8, 6, 13, 11, 9, 8, 10, 17, 10, 10,  
       8, 7, 9, 8, 9, 12, 10, 13, 8, 8, 7, 4, 15, 10, 11, 12, 9,  
       8, 8, 5, 10, 10])
```

```
[46]
```

```
print(accuracy_score(y_test, predictions))
```

```
0.20095693779904306
```

```
[47]
df = X_test.copy()
df['Actual'] = y_test
df['Prediction'] = predictions
df
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Actual	Prediction
668	2	0.550	0.425	0.155	0.9175	0.2775	0.2430	0.3350	13	19
1580	1	0.500	0.400	0.120	0.6160	0.2610	0.1430	0.1935	8	12
3784	2	0.620	0.480	0.155	1.2555	0.5270	0.3740	0.3175	11	14
463	1	0.220	0.165	0.055	0.0545	0.0215	0.0120	0.0200	5	5
2615	2	0.645	0.500	0.175	1.5105	0.6735	0.3755	0.3775	12	10
...	...	...	...	...	...	...	...	...	...	...
1670	0	0.610	0.485	0.150	1.2405	0.6025	0.2915	0.3085	12	8
3055	0	0.610	0.495	0.160	1.0890	0.4690	0.1980	0.3840	11	8
3366	2	0.280	0.210	0.065	0.0905	0.0350	0.0200	0.0300	5	5
1410	0	0.665	0.530	0.180	1.4910	0.6345	0.3420	0.4350	10	10
4035	1	0.520	0.410	0.140	0.5995	0.2420	0.1375	0.1820	11	10

209 rows × 10 columns

```
[48]
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin'
```

```
[49]
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import log_loss
X_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]
Y_predic = [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]
results = confusion_matrix(X_actual, Y_predic)
print ('Confusion Matrix :')
print(results)
print ('Accuracy Score is',accuracy_score(X_actual, Y_predic))
print ('Classification Report : ')
print (classification_report(X_actual, Y_predic))
print('AUC-ROC:',roc_auc_score(X_actual, Y_predic))
print('LOGLOSS Value is',log_loss(X_actual, Y_predic))
```

Confusion Matrix :

```
[[3 3]
```

```
[1 3]]
```

Accuracy Score is 0.6

Classification Report :

	precision	recall	f1-score	support
0	0.75	0.50	0.60	6
1	0.50	0.75	0.60	4
accuracy			0.60	10
macro avg	0.62	0.62	0.60	10
weighted avg	0.65	0.60	0.60	10

AUC-ROC: 0.625

LOGLOSS Value is 13.815750437193334