

# ASSIGNMENT 4

Name	Ranjith B S
Student Roll No	731619205040
Team ID	PNT2022TMID32070
Maximum Marks	2 marks

## ASSIGNMENT 4

### Problem Statement: 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 location (hence food availability) may be required to solve the problem.

### 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 relations 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.

```
20 [1]: #import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import plotly.express as px
```

### 2. Load the dataset into the tool

```
20 [1]: data = pd.read_csv('/content/drive/My Drive/Machine Learning/abalone.csv')
data
```

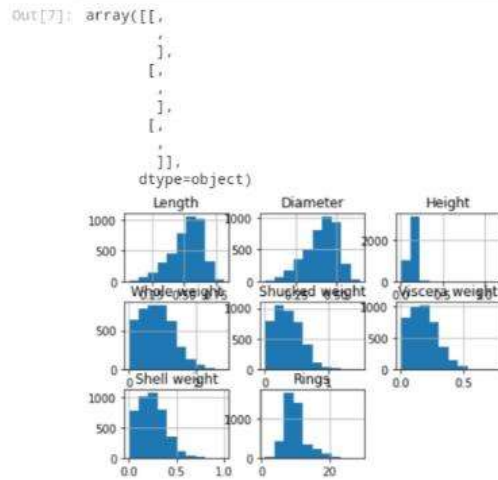
```
Out[1]:
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.7240	0.2245	0.090	0.0500	15
1	M	0.350	0.285	0.080	0.2250	0.0905	0.0405	0.0700	7
2	F	0.330	0.420	0.175	0.6770	0.2500	0.045	0.0900	9
3	M	0.440	0.385	0.105	0.5800	0.2050	0.040	0.0750	10
4	I	0.330	0.255	0.080	0.2050	0.0905	0.035	0.0750	7
...	...	...	...	...	...	...	...	...	...
4072	F	0.365	0.450	0.105	0.8070	0.3240	0.2300	0.2400	9
4073	M	0.390	0.440	0.125	0.8000	0.4000	0.245	0.2005	10
4074	M	0.600	0.475	0.205	1.7000	0.5255	0.2875	0.3000	9
4075	F	0.625	0.485	0.230	1.8945	0.5300	0.2900	0.2900	10
4076	M	0.700	0.505	0.305	1.9405	0.9455	0.3005	0.4050	12
4077	new	0	columns						

### 3. Perform Below Visualizations.

• Univariate Analysis

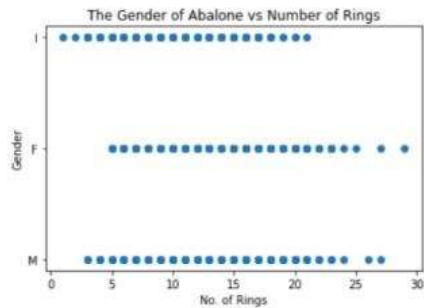
```
In [7]: data['Rings'].value_counts()  
data.hist()
```



• Bi-Variate Analysis

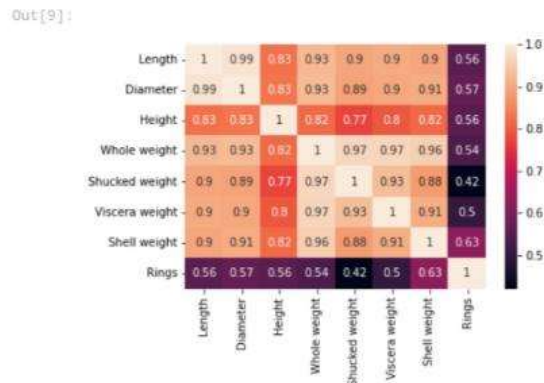
```
In [8]: plt.scatter(data.Rings, data.Sex)  
plt.title('The Gender of Abalone vs Number of Rings')  
plt.xlabel('No. of Rings')  
plt.ylabel('Gender')
```

```
Out[8]: Text(0, 0.5, 'Gender')
```



• Multi-Variate Analysis

```
In [9]: sb.heatmap(data.corr(),annot=True)
```



#### 4. Perform descriptive statistics on the dataset.

```
In [10]: 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
```

```
In [11]: data.describe()
```

```
Out[11]:
```

	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	0.048827	0.490389	0.228963	0.109014	0.139203	3.224109
min	0.075000	0.055000	0.000000	0.002000	0.000000	0.000500	0.005000	1.000000
25%	0.450000	0.350000	0.150000	0.445000	0.180000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.330000	0.170000	0.234000	9.000000
75%	0.650000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.850000	0.650000	0.300000	2.825500	1.488000	0.760000	1.005000	29.000000

#### 5. Check for Missing values and deal with them.

There is no missing values

```
In [12]: data.isnull().any()
```

```
Out[12]: Sex                False
Length              False
Diameter            False
Height              False
Whole weight        False
Shucked weight      False
Viscera weight      False
Shell weight        False
Rings               False
dtype: bool
```

#### 6. Find the outliers and replace them outliers

The dataset does not have a outliers

```
In [54]: fig = px.histogram(data, x='Whole weight')
fig.show()
```

#### 7. Check for Categorical columns and perform encoding.

There is one Categorical column SEX is replaced by an integer

```
In [16]: from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data["Sex"] = le.fit_transform(data["Sex"])
data["Sex"]
```

```
Out[16]: 0      2
1      2
2      0
3      2
4      1
...
4172   0
4173   2
4174   2
4175   0
4176   2
Name: Sex, Length: 4177, dtype: int64
```

## 8. Split the data into dependent and independent variables.

```
In [26]: x=data.iloc[:,0:8].values
         y=data.iloc[:,8:9].values

In [27]: x

Out[27]: array([[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 ]])

In [28]: y

Out[28]: array([[15],
                [ 7],
                [ 9],
                ...,
                [ 9],
                [10],
                [12]])
```

## 9. Scale the independent variables

```
In [50]: x=data.iloc[:,0:8]
         print(x.head())
```

	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
0	0.1010	0.150
1	0.0485	0.070
2	0.1415	0.210
3	0.1140	0.155
4	0.0395	0.055

## 10. Split the data into training and testing

```
In [31]: 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_state=0)

In [31]: x_train.shape

Out[31]: (2923, 8)

In [27]: x_test.shape

Out[27]: (836, 8)
```

## 11. Build the Model

```
In [36]: from sklearn.linear_model import LinearRegression
         lr = LinearRegression()
```

## 12. Train the Model

```
In [38]: lr.fit(x_train, y_train)

Out[38]: LinearRegression()
```

## 13. Test the Model

```
In [40]: y_pred = lr.predict(x_test)
         print(y_test[0:6])
         print((y_pred)[0:6])

[[13]
 [ 8]
 [11]
 [ 5]
 [12]
 [11]]
[[13.11640829]
 [ 9.65691091]
 [10.35350972]
 [ 5.63648715]
 [10.67436485]
 [11.95341338]]
```

## 14. Measure the performance using Metrics.

```
In [41]: # RMSE(Root Mean Square Error)

from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("RMSE value : {:.2f}".format(rmse))
```

RMSE value : 2.26

```
In [43]: from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(lr, x, y, cv=5)
sco=cv_scores.round(4)
print(cv_scores.round(4))
print("Average",sco.sum()/5)
```

[0.4113 0.1574 0.4807 0.5046 0.4362]  
Average 0.39803999999999995