

## ASSIGNMENT -3

Assignment Date	15 October 2022
Student Name	Mohammed Safwan S
Student Roll Number	727719EUCS088
Maximum Marks	2 Marks

**Dataset:** <https://drive.google.com/file/d/1sIv-7x7CE0zAPAt0Uv-6pbO2ST2LVp5u/view>

### Loading the dataset:

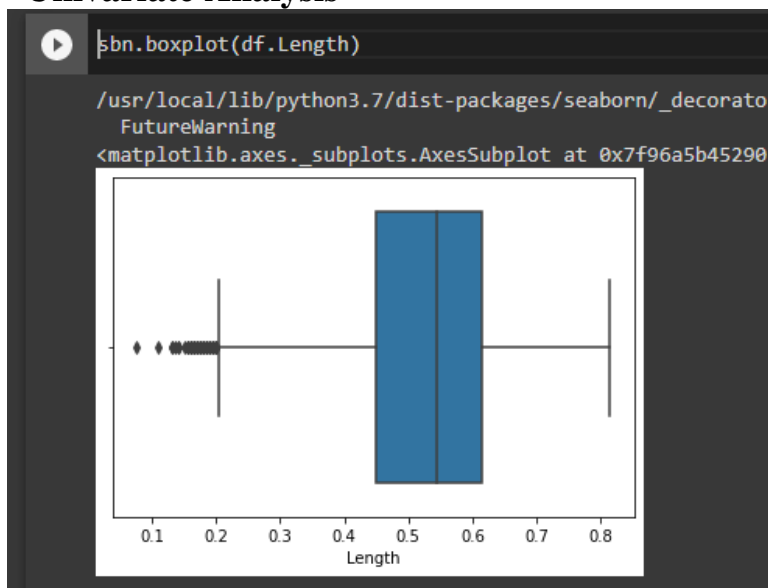
```
[1] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sbn

[84] df=pd.read_csv("abalone.csv")
df.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

### Perform Below Visualizations.

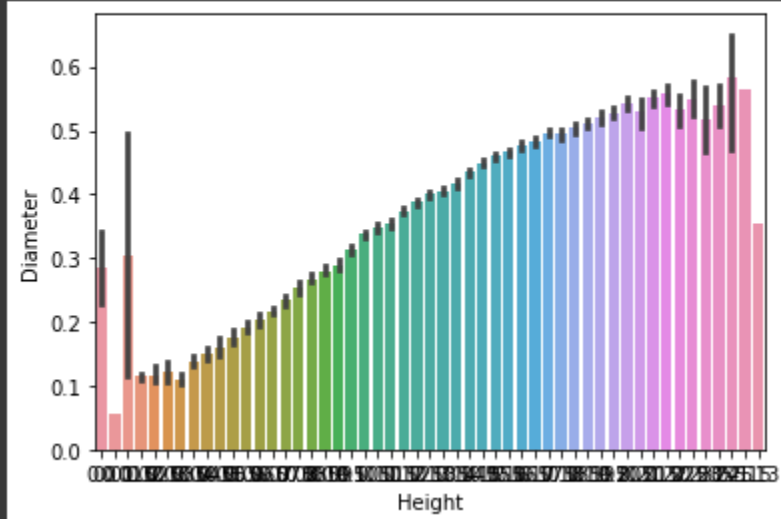
#### • Univariate Analysis



## Bi-Variate Analysis

```
[ ] sbn.barplot(x=df.Height,y=df.Diameter)
```

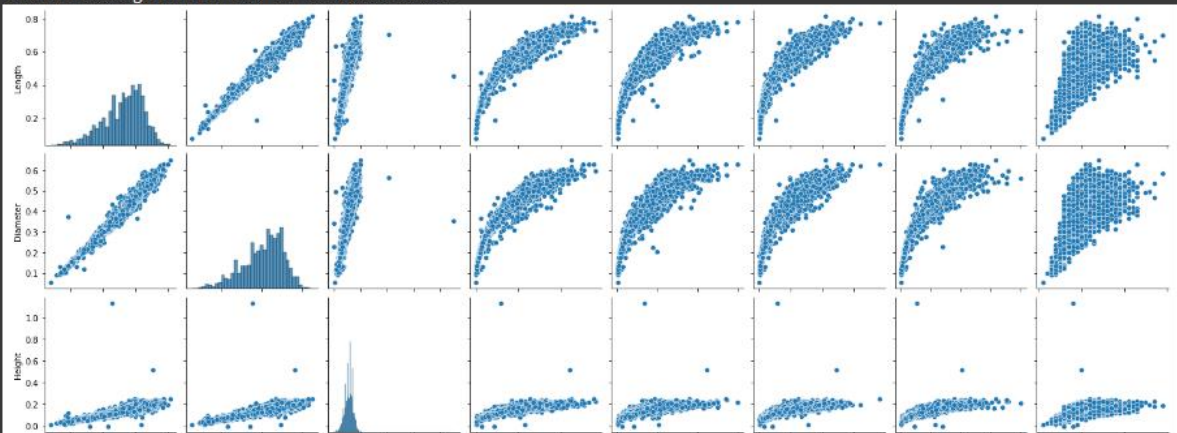
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f96a5571dd0>
```



## Multi-Variate Analysis

```
sbn.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f96a5b8d050>
```



## Perform descriptive analytics on the dataset

```
[ ] df['Height'].mean()
```

```
0.13951639932966242
```

```
[ ] df['Diameter'].median()
```

```
0.425
```

```
[ ] df['Length'].mode()
```

```
0    0.550
```

```
1    0.625
```

```
dtype: float64
```



```
df.max()
```



```
Sex    M
```

```
Length 0.815
```

```
Diameter 0.65
```

```
Height 1.13
```

```
Whole weight 2.8255
```

```
Shucked weight 1.488
```

```
Viscera weight 0.76
```

```
Shell weight 1.005
```

```
Rings 29
```

```
dtype: object
```

```
[ ] df.min()
```

```
Sex    F
```

```
Length 0.075
```

```
Diameter 0.055
```

```
Height 0.0
```

```
Whole weight 0.002
```

```
Shucked weight 0.001
```

```
Viscera weight 0.0005
```

```
Shell weight 0.0015
```

```
Rings 1
```

## Check for Missing values and deal with them.

```
df.isna().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

## Find the outliers and replace them outliers

```
[3] q1=df.Rings.quantile(0.25)
     q3=df.Rings.quantile(0.75)
     iqr=q3-q1

[4] print(iqr)

3.0

df=df[~((df.Rings<(q1-1.5*iqr))|(df.Rings>(q3+1.5*iqr)))]
```

## Check for Categorical columns and perform encoding.

```
[6] df['Sex'].replace({'M':1,'F':0,'I':2},inplace=True)
```

/usr/local/lib/python3.7/dist-packages/pandas/core/generic.py:6619: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame  
  
See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#)  
return self.\_update\_inplace(result)

```
df.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

**Split the data into dependent and independent variables.**

```
✓ 0s x=df.iloc[:, :-1].values
✓ 0s [50] y=df.iloc[:, -1].values
```

**Scale the independent variables**

```
[39] from sklearn.preprocessing import StandardScaler
std=StandardScaler()
x=std.fit_transform(x)
x

array([[ -0.03822742, -0.55104264, -0.40422906, ..., -0.58564588,
        -0.69758868, -0.60447624],
       [ -0.03822742, -1.4332      , -1.42309849, ..., -1.14600915,
        -1.17989471, -1.21362086],
       [ -1.2907376 ,  0.07906976,  0.15614912, ..., -0.44219288,
        -0.32552403, -0.14761778],
       ...,
       [ -0.03822742,  0.66717467,  0.71652731, ...,  0.76370889,
         1.01574608,  0.59858438],
       [ -1.2907376 ,  0.87721213,  0.81841425, ...,  0.78836487,
         0.77229637,  0.50721269],
       [ -0.03822742,  1.59133952,  1.53162285, ...,  2.64652949,
         1.83336964,  2.02245992]])
```

## Split the data into training and testing

```
[60] from sklearn.model_selection import train_test_split
      x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

```
[61] x_train
      array([[0.    , 0.695 , 0.53  , ..., 0.75  , 0.4195, 0.6095],
             [0.    , 0.525 , 0.41  , ..., 0.4065, 0.198 , 0.177 ],
             [1.    , 0.64  , 0.485 , ..., 0.456 , 0.2245, 0.2835],
             ...,
             [0.    , 0.595 , 0.47  , ..., 0.4515, 0.178 , 0.155 ],
             [1.    , 0.555 , 0.46  , ..., 0.3345, 0.1935, 0.275 ],
             [2.    , 0.36  , 0.27  , ..., 0.097 , 0.0405, 0.065 ]])
```

```
[62] y_train
      array([14,  8,  9, ..., 11, 10,  6])
```

```
[63] x_test
      array([[1.    , 0.7   , 0.565 , ..., 0.895 , 0.3355, 0.446 ],
             [0.    , 0.735 , 0.6   , ..., 1.1335, 0.44  , 0.6   ],
             [0.    , 0.61  , 0.495 , ..., 0.3705, 0.3135, 0.33  ],
             ...,
             [0.    , 0.66  , 0.53  , ..., 0.493 , 0.245 , 0.49  ],
             [1.    , 0.555 , 0.435 , ..., 0.341 , 0.1645, 0.214 ],
             [1.    , 0.505 , 0.39  , ..., 0.2595, 0.18  , 0.19  ]])
```

```
[64] y_test
      array([ 9, 11, 12, 15,  9,  7,  9,  9,  9, 11, 10,  9,  7, 11,  8, 12, 10,
             10,  8,  8,  6,  5, 10,  9,  9, 15,  9, 10, 13,  6, 10,  7, 11,  9,
             10, 11, 11, 10,  9,  9,  7,  7, 14,  9, 15, 10,  9,  9,  4,  8, 11,
             5,  9, 15,  9, 11, 11,  8, 13,  9, 11, 11, 10,  9, 12, 15, 11,  8,
             9,  7, 11, 14,  6, 13, 10,  8, 10,  8,  5,  6, 10, 10, 12,  8, 11,
             11, 12, 10,  6, 13, 10,  8,  8,  7, 10, 10,  4,  8, 10,  7,  5,  8,
             13,  6,  9, 11,  7, 11,  9, 11, 10,  9, 10, 13,  8, 11,  9, 15, 13,
             6, 10,  8, 11,  6, 11, 10, 10, 10,  7, 14, 11,  8,  9, 10, 15,  9,
             9, 11, 15,  8, 10,  8, 15, 10, 14, 12,  9, 10, 14,  9, 10,  5,  7,
             10, 11, 13,  9,  9, 13,  7, 11,  9, 10, 10, 13,  8,  9,  8,  9,  7,
             7,  8, 11,  8,  4, 11,  7,  9,  8, 11, 10, 10, 14,  6,  6,  4, 11,
             10,  8,  7,  6, 12, 12, 11, 13, 11, 10, 10, 12,  5, 11, 13,  9, 12,
             10, 10, 11, 10,  9,  8, 11, 14, 11,  9,  6,  7,  9,  7,  6, 11,  9,
             11,  7, 14,  8, 10, 13, 15,  5,  7,  9,  5, 11,  4, 10, 10, 12, 11,
             13,  5, 10,  9,  9,  9, 12,  9,  9,  8, 11, 11, 10,  7, 11,  8, 11,
             9,  6,  8, 13,  9,  9, 11, 10, 11, 10,  4, 15, 13,  9,  9, 11, 11,
             11, 11, 11, 12, 12,  5,  9,  9, 11,  8,  6, 10,  9, 11,  9,  7,  7,
             10, 12,  8, 11,  9, 12, 11,  8, 11, 10, 12,  9,  9, 10,  9,  9, 15,
             4, 14,  9,  7, 10, 11,  5,  9,  8,  8,  8, 10, 12, 13, 12, 11, 10,
             15,  9,  9,  9,  9, 13,  6,  8, 11, 11, 11,  9,  8,  9, 10,  7,  9,
             5,  8, 12, 11,  9,  8,  9, 10, 11,  7,  6,  4, 12,  9,  6,  7,  8,
             13, 12, 12, 10, 14, 10, 12,  9,  9, 13,  9, 10, 13,  8, 15,  8, 10,
             13,  5, 10,  6,  8,  9, 12, 14, 10, 14, 11, 10,  9,  9, 10, 11,  8,
             12, 11, 10,  5, 11, 11, 15, 14, 13, 12,  7, 11, 10, 13,  9,  6, 15,
```

## Build the Model

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators = 1000, oob_score = True, n_jobs=-1, min_samples_split = 6, min_samples_leaf = 4, max_features = 'sqrt', max_depth = 120, bootstrap = True)
```

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators = 1000, oob_score = True, n_jobs=-1, min_samples_split = 6, min_samples_leaf = 4, max_features = 'sqrt', max_depth = 120, bootstrap = True)
```

## Train the Model

```
model.fit(x_train, y_train)

RandomForestRegressor(max_depth=120, max_features='sqrt', min_samples_leaf=4, min_samples_split=6, n_estimators=1000, n_jobs=-1, oob_score=True)
```

## Test the Model

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

9.23052686,	6.76527568,	6.27337663,	9.7808718	10.46575533,
10.39856318,	9.92302597,	7.03874443,	9.28506128,	4.8144354
8.51898345,	9.44591446,	10.50450779,	10.28790825,	10.1401078
7.95223754,	5.30119942,	9.96964081,	6.82311145,	6.29814986,
8.68373737,	8.21113623,	10.6245237	10.77857176,	11.17060581,
9.16360497,	10.28201394,	6.6367132	10.49952107,	8.41476732,
9.11490296,	10.11751273,	8.49518805,	4.88652692,	10.28148647,
10.94575126,	11.71629647,	9.46380019,	9.44207265,	10.21271332,
9.14684877,	9.86565957,	8.92327854,	10.88901169,	10.58669074,
8.954949	12.25015427,	10.70193653,	11.64170245,	8.81236519,
8.06411968,	5.5665906	8.73177525,	11.59118191,	10.65204263,
9.18393415,	11.58186427,	6.54125027,	10.43332356,	6.94692004,
11.27852383,	9.31304977,	8.40214749,	6.02948651,	12.03950182,
6.58799368,	11.31287941,	11.37077235,	4.7255203	11.15012629,
10.0408263	7.73944001,	6.9423391	4.90132305,	10.40211536,
10.04235146,	6.96710608,	11.05620166,	11.35397795,	10.22259343,
11.63211032,	9.39309664,	8.88237849,	10.83092528,	6.6303001
11.52583068,	10.787237	9.93738872,	11.74766958,	10.45900969,
7.60619186,	9.82836881,	9.69601129,	10.5296791	9.20391431,
9.00121742,	9.79719374,	10.45730253,	8.39235724,	7.41134463,

## Measure the performance using Metrics.

```
✓ [93] from sklearn.metrics import r2_score  
0s acc=r2_score(y_test,predictions)  
acc
```

```
0.5902139902351261
```