

TEAM ID	PNT2022TMID27851
STUDENT NAME	R. NIVETHA
DOMAIN NAME	HEALTH CARE
PROJECT NAME	EARLY DETECTION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING
MAXIMUM MARKS	2 MARKS

- LOAD THE DATASET INTO THE DATASET

LOAD DATASET

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

```
✓ [3] df = pd.read_csv('abalone.csv')
```

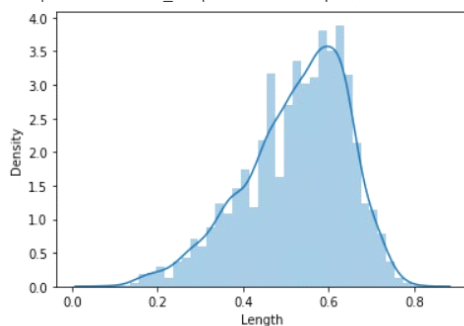
- PERFORM BELOW VISUALIZATIONS
- UNIVARIANT

VISUALIZATIONS

UNIVARIANT ANALYSIS

```
[4] sns.distplot(df.Length)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function  
warnings.warn(msg, FutureWarning)  
<matplotlib.axes._subplots.AxesSubplot at 0x7f61214aa210>
```

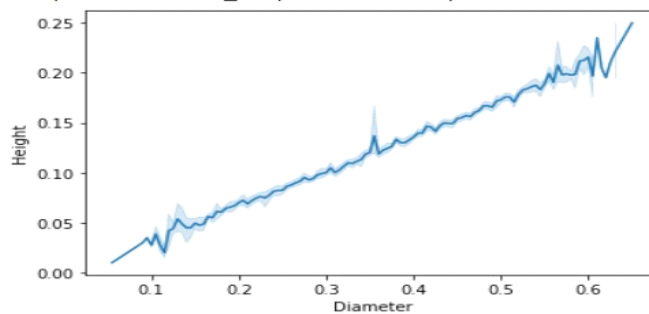


- BIVARIANT

BIVARIANT ANALYSIS

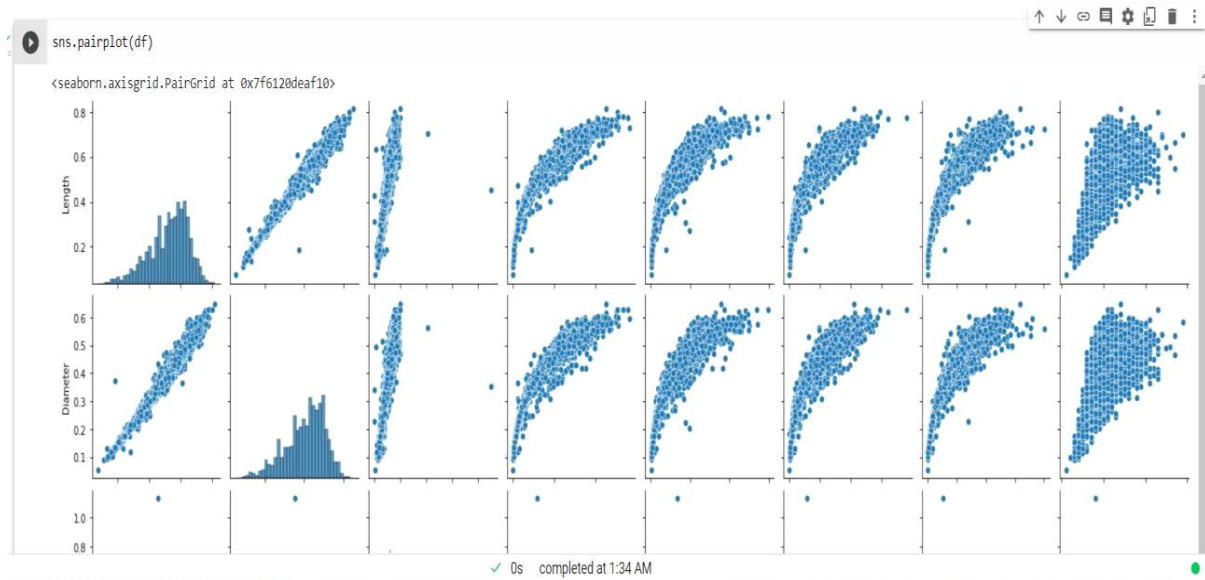
```
[5] sns.lineplot(df.Diameter,df.Height)
```

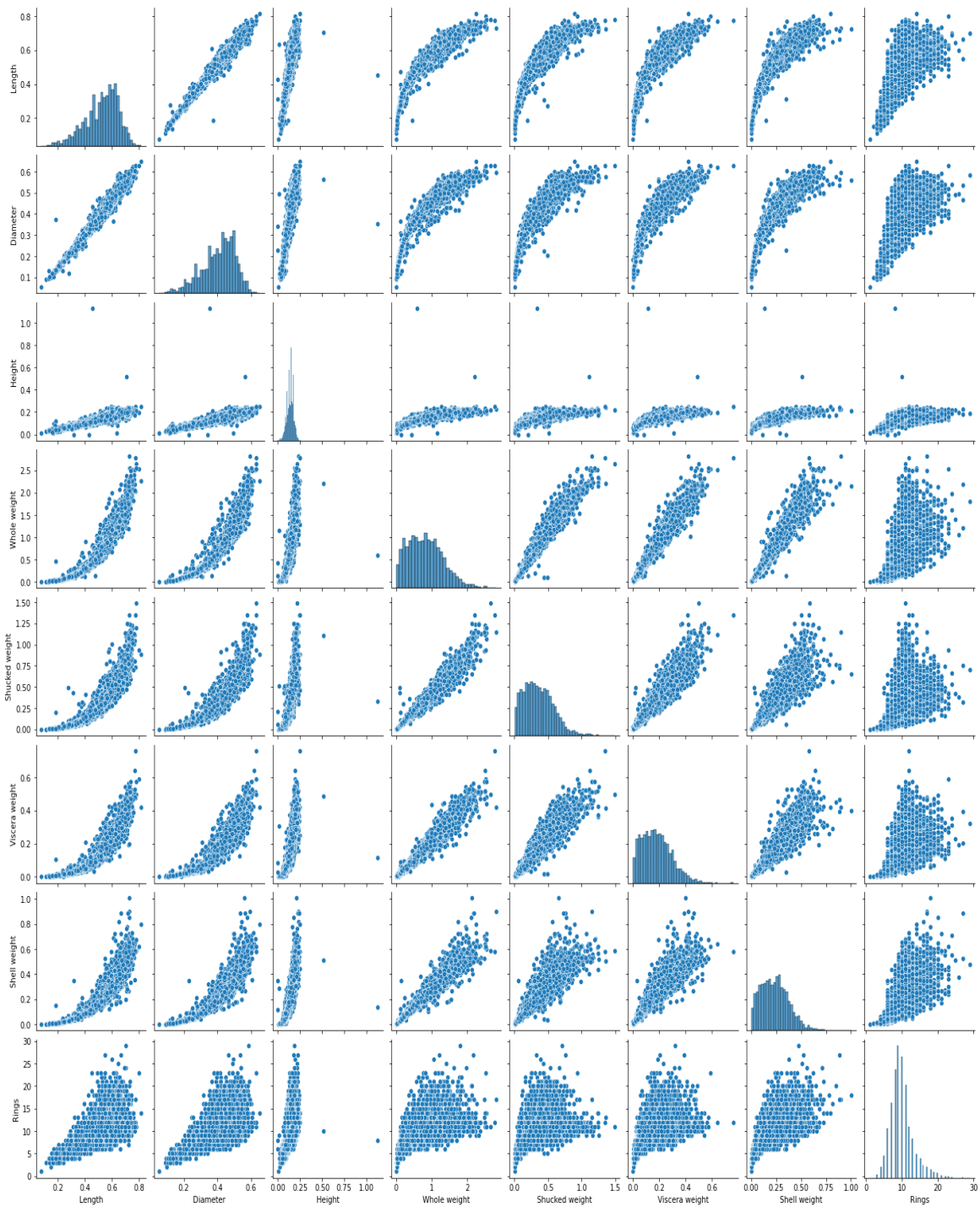
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass  
FutureWarning  
<matplotlib.axes._subplots.AxesSubplot at 0x7f6120e37150>
```



- MULTIVARIANT

MULTIVARIANT ANALYSIS





- PERFORM DESCRIPTIVE STATISTICS ON THE DATASET

DESCRIPTIVE STATISTICS

✓ [7] df.describe()

0s

	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

✓ [8] df.shape

0s

(4177, 9)

✓ [9] df.info()

0s

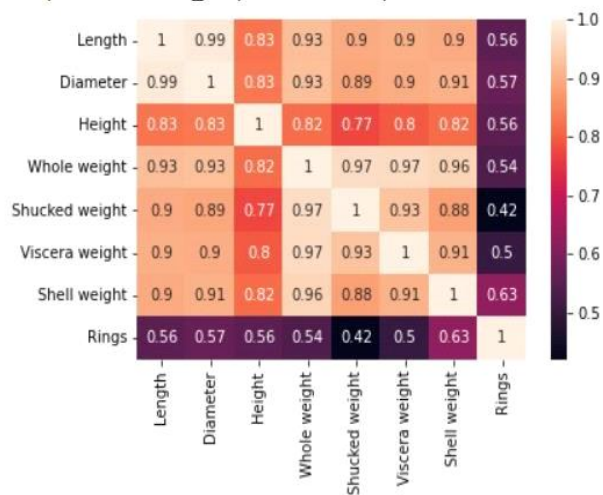
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   Sex              4177 non-null   object
1   Length           4177 non-null   float64
2   Diameter         4177 non-null   float64
3   Height           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
```

```
✓ [11] df.corr()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Length	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
Diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
Height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
Whole weight	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
Shucked weight	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
Viscera weight	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
Shell weight	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
Rings	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

```
✓ [12] sns.heatmap(df.corr(),annot=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f611d1ecc90>



- CHECK FOR MISSING VALUES AND DEAL WITH THEM

CHECK FOR MISSING VALUES AND DEAL WITH THEM

```
✓ [13] df.isnull().any()
```

0s

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

```
✓ [14] df['Length'].fillna(df['Length'].mean(),inplace=True)
```

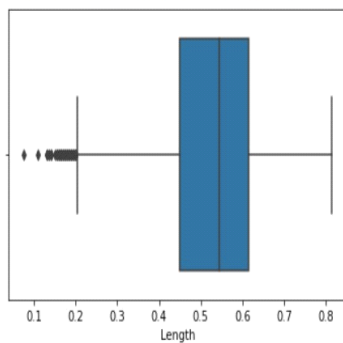
• FIND THE OUTLIERS AND REPLACETHEM OUTLIERS

FIND THE OUTLIERS AND REPLACE THEM OUTLIERS

```
✓ [15] sns.boxplot(df.Length)
```

0s

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From v
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f612059ac90>
```



4

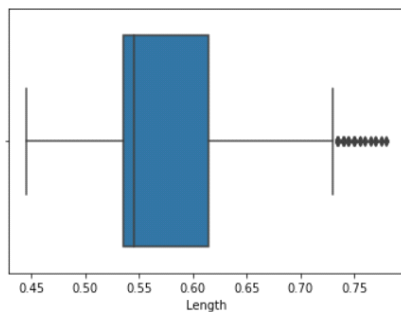
✓
0s

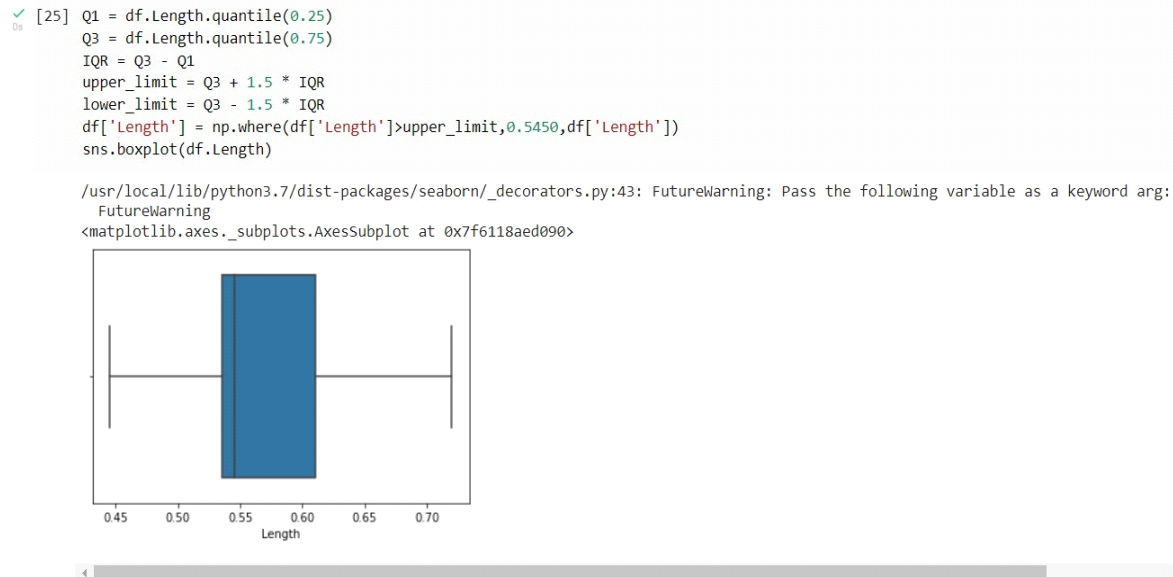
```
[16] df.median()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
    """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
```

```
[23] Q1 = df.Length.quantile(0.25)
Q3 = df.Length.quantile(0.75)
IQR = Q3 - Q1
upper_limit = Q3 + 1.5 * IQR
lower_limit = Q3 - 1.5 * IQR
df['Length'] = np.where(df['Length'] < lower_limit, 0.5450, df['Length'])
sns.boxplot(df.Length)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword argument to the function:
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f6118b5d090>
```





• CHECK FOR CATEGORICAL COLUMN AND PERFORM ENCODING

CHECK FOR CATEGORICAL COLUMN AND PERFORM ENCODING

```
[30] from sklearn.preprocessing import LabelEncoder
```

```
[32] le = LabelEncoder()
      df.Sex = le.fit_transform(df.Sex)
      df.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
1	2	0.545	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	2	0.545	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	1	0.545	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

• SPLIT THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

SPLIT THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

```
✓ [33] x = df.drop(columns=['Rings'],axis=1)
0s 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
1	2	0.545	0.265	0.090	0.2255	0.0995	0.0485	0.070
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210
3	2	0.545	0.365	0.125	0.5160	0.2155	0.1140	0.155
4	1	0.545	0.255	0.080	0.2050	0.0895	0.0395	0.055

```
✓ [34] y = df.Rings
0s y.head()
```

```
0    15
1     7
2     9
3    10
4     7
Name: Rings, dtype: int64
```

• SCALE THE INDEPENDENT VARIABLES

SCALE THE INDEPENDENT VARIABLES

```
✓ [38] from sklearn.preprocessing import MinMaxScaler
```


```
✓ [39] scale = MinMaxScaler()  
      X_scaled = pd.DataFrame(scale.fit_transform(X), columns=X.columns)  
      X_scaled
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	1.0	0.036364	0.521008	0.084071	0.181335	0.150303	0.132324	0.147982
1	1.0	0.363636	0.352941	0.079646	0.079157	0.066241	0.063199	0.068261
2	0.0	0.309091	0.613445	0.119469	0.239065	0.171822	0.185648	0.207773
3	1.0	0.363636	0.521008	0.110619	0.182044	0.144250	0.149440	0.152965
4	0.5	0.363636	0.336134	0.070796	0.071897	0.059516	0.051350	0.053313
...
4172	0.0	0.436364	0.663866	0.146018	0.313441	0.248151	0.314022	0.246637
4173	1.0	0.527273	0.647059	0.119469	0.341420	0.294553	0.281764	0.258097
4174	1.0	0.563636	0.705882	0.181416	0.415796	0.352724	0.377880	0.305431
4175	0.0	0.654545	0.722689	0.132743	0.386931	0.356422	0.342989	0.293473
4176	1.0	0.963636	0.840336	0.172566	0.689393	0.635171	0.495063	0.491779

4177 rows × 8 columns

- SPLIT THE DATA INTO TRAINING AND TESTING

SPLIT THE DATA INTO TRAINING AND TESTING

```
✓  from sklearn.model_selection import train_test_split
```

```
✓ [41] X_train,X_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.3,random_state = 0)
```

```
✓ [42] X_train.shape  
0s (2923, 8)
```

```
✓ [43] X_test.shape  
0s (1254, 8)
```

```
✓ [44] y_train.shape  
0s (2923,)
```

```
✓ [45] y_test.shape  
0s (1254,)
```

• BUILD THE MODEL

BUILD THE MODEL

```
✓ [49] from sklearn.linear_model import LinearRegression  
0s model = LinearRegression()
```

• TRAIN THE MODEL

TRAIN THE MODEL


```
✓ [51] model.fit(X_train,y_train)
0s
      LinearRegression()
```

- TEST THE MODEL

TEST THE MODEL

```
✓ [52] y_predict = model.predict(X_test)
0s
```

```
✓ [53] pd.DataFrame({'Actual':y_test,'Predicted':y_predict})
0s
```




	Actual	Predicted
668	13	13.287283
1580	8	9.927522
3784	11	10.338571
463	5	5.422012
2615	12	10.627746
...
1052	12	14.794857
3439	8	8.409200
1174	9	8.971563
2210	18	18.703214
2408	15	11.648978

1254 rows × 2 columns

- MEASURE THE PERFORMANCE USING METRICS

MEASURE THE PERFORMANCE USING METRICS

✓ 0s  `from sklearn import metrics`

✓ 0s `[55] metrics.r2_score(y_test,y_predict)`

`0.5202338594368163`