ASSIGNMENT - 3

DATE	9 October 2022
TEAM ID	PNT2022TMID38667
PROJECT NAME	Early Detection of Chronic Kidney Disease using Machine Learning
MAXIMUM MARKS	2 Marks

1. Download the dataset

	A	B	C	D	E	F	G	H	I	J
1	Sex	Length	Diameter	Height	Whole wei	Shucked w	Viscera we	Shell weigh	Rings	
2	M	0.455	0.365	0.095	0.514	0.2245	0.101	0.15	15	
3	M	0.35	0.265	0.09	0.2255	0.0995	0.0485	0.07	7	
4	F	0.53	0.42	0.135	0.677	0.2565	0.1415	0.21	9	
5	M	0.44	0.365	0.125	0.516	0.2155	0.114	0.155	10	
6	I	0.33	0.255	0.08	0.205	0.0895	0.0395	0.055	7	
7	I	0.425	0.3	0.095	0.3515	0.141	0.0775	0.12	8	
8	F	0.53	0.415	0.15	0.7775	0.237	0.1415	0.33	20	
9	F	0.545	0.425	0.125	0.768	0.294	0.1495	0.26	16	
10	M	0.475	0.37	0.125	0.5095	0.2165	0.1125	0.165	9	
11	F	0.55	0.44	0.15	0.8945	0.3145	0.151	0.32	19	
12	F	0.525	0.38	0.14	0.6065	0.194	0.1475	0.21	14	
13	M	0.43	0.35	0.11	0.406	0.1675	0.081	0.135	10	
14	M	0.49	0.38	0.135	0.5415	0.2175	0.095	0.19	11	
15	F	0.535	0.405	0.145	0.6845	0.2725	0.171	0.205	10	
16	F	0.47	0.355	0.1	0.4755	0.1675	0.0805	0.185	10	
17	M	0.5	0.4	0.13	0.6645	0.258	0.133	0.24	12	
18	I	0.355	0.28	0.085	0.2905	0.095	0.0395	0.115	7	
19	F	0.44	0.34	0.1	0.451	0.188	0.087	0.13	10	
20	M	0.365	0.295	0.08	0.2555	0.097	0.043	0.1	7	
21	M	0.45	0.32	0.1	0.381	0.1705	0.075	0.115	9	
22	M	0.355	0.28	0.095	0.2455	0.0955	0.062	0.075	11	
23	I	0.38	0.275	0.1	0.2255	0.08	0.049	0.085	10	
24	F	0.565	0.44	0.155	0.9395	0.4275	0.214	0.27	12	
25	F	0.55	0.415	0.135	0.7635	0.318	0.21	0.2	9	
26	F	0.615	0.48	0.165	1.1615	0.513	0.301	0.305	10	

2. Load the dataset

In [1]:
```python
## import required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

## 2.loading dataset

df=pd.read_csv('abalone.csv')
df.head()
```

Out[1]:

	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. Perform Below Visualizations.

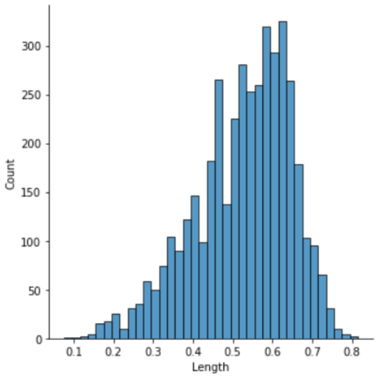
i. Univariate Analysis

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

## 3i.univariate analysis

df=pd.read_csv('abalone.csv')
df.head()
sns.displot(df.Length)
```

Out[5]: <seaborn.axisgrid.FacetGrid at 0x1b9e248e880>



ii.Bi - Variate Analysis

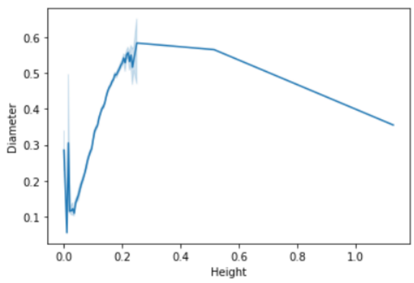
In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

## 3ii.bivariate analysis

df=pd.read_csv('abalone.csv')
df.head()
sns.lineplot(df.Height,df.Diameter)
```

```
C:\Users\sahan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword ar
gs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[4]: <AxesSubplot:xlabel='Height', ylabel='Diameter'>



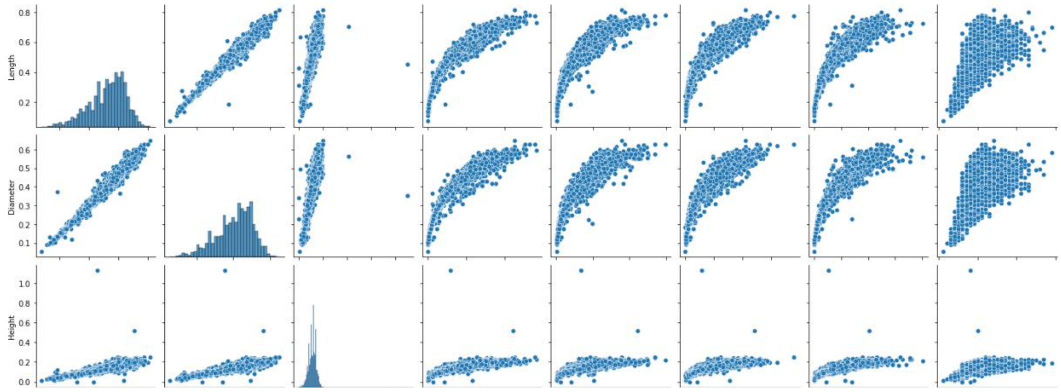
iii.Multi - Variate Analysis

```
In [6]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from matplotlib import rcParams

        ## 3iii.multi-variate analysis

        df=pd.read_csv('abalone.csv')
        df.head()
        sns.pairplot(df)
```

```
Out[6]: <seaborn.axisgrid.PairGrid at 0x1b9e4e53250>
```



4. Perform descriptive statistics on the dataset

```
In [7]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from matplotlib import rcParams

        ## 4.descriptive analysis

        df=pd.read_csv('abalone.csv')
        df.head()
        df.describe()
```

Out[7]:

	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

5. Handle the Missing values.

In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

## 5.no missing values

df=pd.read_csv('abalone.csv')
df.head()
df.isnull().any()
```

Out[8]:
```
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 the outliers

In [10]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

## 6.find the outlier

df=pd.read_csv('abalone.csv')
df.head()
Q1=df.Length.quantile(0.25)
Q3=df.Length.quantile(0.75)
Q1,Q3
```

Out[10]:
```
(0.45, 0.615)
```

In [12]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

## 6.replace the outlier

df=pd.read_csv('abalone.csv')
df.head()
Q1=df.Length.quantile(0.25)
Q3=df.Length.quantile(0.75)
Q1,Q3
IQR=Q3-Q1
IQR
lower_limit = Q1-1.5*IQR
upper_limit = Q3+1.5*IQR
lower_limit, upper_limit
df_no_outlier = df[(df.Length>lower_limit)&(df.Length<upper_limit)]
df_no_outlier
```

Out[12]:

	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	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4128 rows × 9 columns

7. Check for Categorical columns and perform encoding.

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
from sklearn.preprocessing import LabelEncoder

## 7.categorical columns encoding

df=pd.read_csv('abalone.csv')
le=LabelEncoder()
df.Gender=le.fit_transform(df.Length)
df.Geographyr=le.fit_transform(df.Diameter)
df.head()
```

```
C:\Users\sahan\AppData\Local\Temp\ipykernel_20396\2379657326.py:12: UserWarning: Pandas doesn't allow columns to be created via
a new attribute name - see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access
  df.Gender=le.fit_transform(df.Length)
C:\Users\sahan\AppData\Local\Temp\ipykernel_20396\2379657326.py:13: UserWarning: Pandas doesn't allow columns to be created via
a new attribute name - see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access
  df.Geographyr=le.fit_transform(df.Diameter)
```

Out[13]:

	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

8. Split the data into dependent and independent variables

Split the data into dependent and independent variables ¶

In [24]:
```
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]
```

In [25]:
```
from sklearn.model_selection import train_test_split

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

# using the train test split function
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

print('X_train : ')
print(X_train.head())
print(X_train.shape)

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

```
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,)
```

9. Scale the independent variables

Scale the independent variables

In [27]:
```python
#scaling
df_scaled =data.copy()
col_names = ['Shucked weight', 'Whole weight']
features = df_scaled[col_names]
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_scaled[col_names] = scaler.fit_transform(features.values)
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(5, 10))

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

Out[27]:

	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

10. Split the data into training and testing

Split the data into training and testing

In [46]:
```
#testing and training
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

# split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)
print(X_train, X_test, y_train, y_test)
```

```
      Sex  Length  Diameter  Height  Whole weight  Shucked weight  \
1794    1   0.575     0.450   0.130        0.8145          0.4030
1466    2   0.515     0.425   0.145        0.9365          0.4970
2275    2   0.655     0.525   0.185        1.2590          0.4870
3929    0   0.650     0.515   0.215        1.4980          0.5640
1955    0   0.645     0.510   0.180        1.6195          0.7815
...   ...     ...       ...     ...           ...             ...
2103    2   0.375     0.290   0.100        0.2760          0.1175
3603    1   0.420     0.325   0.110        0.3250          0.1245
3340    1   0.540     0.435   0.145        0.9700          0.4285
3064    2   0.635     0.500   0.180        1.1540          0.4405
3398    2   0.365     0.285   0.085        0.2205          0.0855

      Viscera weight  Shell weight
1794          0.1715        0.2130
1466          0.1810        0.2185
2275          0.2215        0.4450
3929          0.3230        0.4250
1955          0.3220        0.4675
...              ...           ...
2103          0.0565        0.0850
3603          0.0755        0.1025
3340          0.2200        0.2640
3064          0.2315        0.3870
3398          0.0515        0.0700
```

In [47]:
```
X_train
```

Out[47]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
1794	1	0.575	0.450	0.130	0.8145	0.4030	0.1715	0.2130
1466	2	0.515	0.425	0.145	0.9365	0.4970	0.1810	0.2185
2275	2	0.655	0.525	0.185	1.2590	0.4870	0.2215	0.4450
3929	0	0.650	0.515	0.215	1.4980	0.5640	0.3230	0.4250
1955	0	0.645	0.510	0.180	1.6195	0.7815	0.3220	0.4675
...	...	...	...	...	...	...	...	...
2103	2	0.375	0.290	0.100	0.2760	0.1175	0.0565	0.0850
3603	1	0.420	0.325	0.110	0.3250	0.1245	0.0755	0.1025
3340	1	0.540	0.435	0.145	0.9700	0.4285	0.2200	0.2640
3064	2	0.635	0.500	0.180	1.1540	0.4405	0.2315	0.3870
3398	2	0.365	0.285	0.085	0.2205	0.0855	0.0515	0.0700

3341 rows × 8 columns

In [48]:
```
y_train
```

Out[48]:
```
1794    10
1466     8
2275    20
3929    16
1955    12
        ..
2103     9
3603     7
3340    17
3064     9
3398     9
```

In [49]:
```
X_test
```

Out[49]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
341	0	0.620	0.510	0.205	1.3475	0.4775	0.2565	0.480
3413	2	0.490	0.395	0.120	0.6740	0.3325	0.1235	0.185
1088	1	0.450	0.340	0.120	0.4925	0.2410	0.1075	0.120
98	2	0.470	0.370	0.130	0.5225	0.2010	0.1330	0.165
3661	1	0.550	0.415	0.150	0.7915	0.3535	0.1760	0.236
...	...	...	...	...	...	...	...	...
1662	0	0.605	0.455	0.145	0.9775	0.4680	0.1775	0.275
2656	2	0.545	0.420	0.120	0.7865	0.4030	0.1850	0.170
2185	2	0.445	0.340	0.120	0.4475	0.1930	0.1035	0.130
727	0	0.480	0.375	0.105	0.5250	0.2185	0.1195	0.155
3796	0	0.675	0.540	0.210	1.5930	0.6860	0.3180	0.450

836 rows × 8 columns

In [50]:
```
y_test
```

Out[50]:
```
341     14
3413     9
1088     6
98       7
3661    10
        ..
1662     9
2656     7
2185     9
727     12
3796    11
```

11. Build the Model

12. Train the Model

13. Test the Model

Build the model

In [61]:
```
logreg = LogisticRegression()
```

Train and Test the model

In [53]:
```
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.3f}'. format(logreg.score (X_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.261
```

Measure the performance using metrics

In [59]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix\n\n', cm)
```

```
Confusion matrix

 [[ 0  5  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  2 15  5  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  2  9 30  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  7 32 27  7  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  2 22 43 36 12  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  2 11 25 70 30  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  9 12 37 47 14  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  5  9 29 32 15  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  3 10 13 20 15  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  2  5  9 20 13  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  5  6 12  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
```

14. Measure the performance using Metrics

In [55]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           4       0.00      0.00      0.00         9
           5       0.22      0.09      0.12        23
           6       0.23      0.20      0.21        46
           7       0.27      0.43      0.33        74
           8       0.30      0.37      0.33       115
           9       0.32      0.49      0.39       143
          10       0.23      0.39      0.29       119
          11       0.16      0.17      0.16        90
          12       0.00      0.00      0.00        62
          13       0.00      0.00      0.00        49
          14       0.00      0.00      0.00        32
          15       0.00      0.00      0.00        14
          16       0.00      0.00      0.00        14
          17       0.00      0.00      0.00        12
          18       0.00      0.00      0.00         9
          19       0.00      0.00      0.00         9
          20       0.00      0.00      0.00         5
          21       0.00      0.00      0.00         4
          22       0.00      0.00      0.00         1
          23       0.00      0.00      0.00         4
          26       0.00      0.00      0.00         1
          27       0.00      0.00      0.00         1

    accuracy                           0.26       836
   macro avg       0.08      0.10      0.08       836
weighted avg       0.19      0.26      0.21       836
```