

### ASSIGNMENT 3

<b>Date</b>	10 October 2022
<b>Team ID</b>	PNT2022TMID38667
<b>Project Name</b>	Project – Early Detection of Chronic Kidney Disease using Machine Learning
<b>Maximum Marks</b>	2 Marks

1.Download the dataset: [Dataset](#)

	A	B	C	D	E	F	G	H	I
1	Sex	Length	Diameter	Height	Whole we	Shucked w	Viscera we	Shell weig	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

## 2. Load the dataset into the tool.

```
In [1]: ## 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.Univariate Analysis

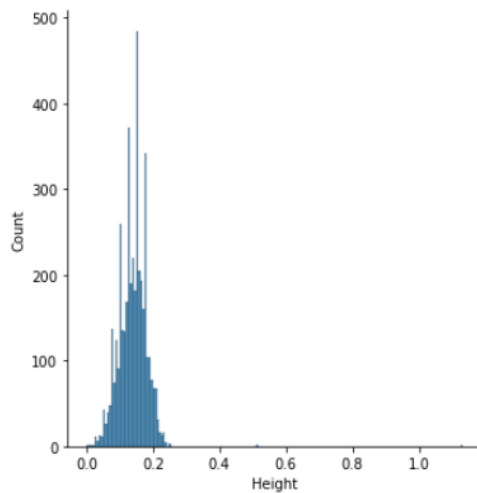
```
In [2]: ## import required libraries

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

## 3.univariate analysis

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

```
Out[2]: <seaborn.axisgrid.FacetGrid at 0x18bba4e4340>
```



### 3.Bi-Variate Analysis

In [8]: *## import required libraries*

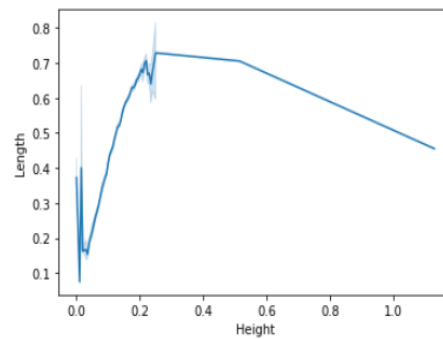
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
```

*## 3.bi-variate analysis*

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

C:\Users\ELCOT\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword arguments: 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[8]: <AxesSubplot:xlabel='Height', ylabel='Length'>



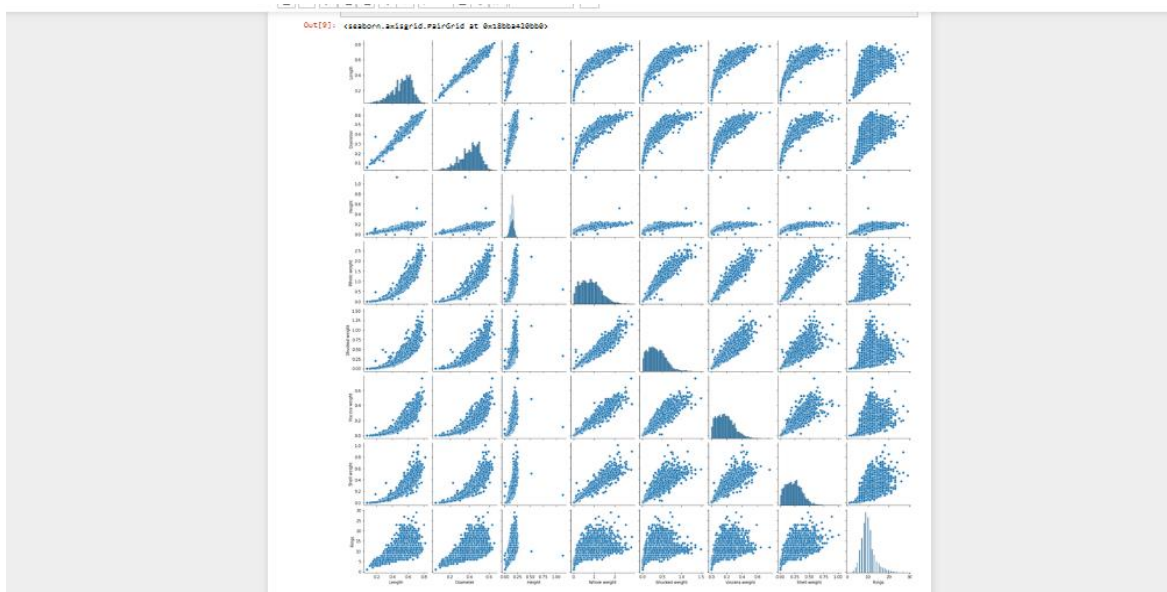
### 3.Multi-Variate Analysis

In [9]: *## import required libraries*

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

*## 3.multi-variate analysis*

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



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

```
In [10]: 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[10]:
```

	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. Check for Missing values and deal with them.

```

In [11]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

## 5.no missing value

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

Out[11]: 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.

```

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

## 6.Find outliers

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

```

Out[13]: (0.115, 0.165)

```

In [14]: 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.Height.quantile(0.25)
Q3=df.Height.quantile(0.75)
Q1,Q3
IQR=Q3-Q1
lower_limit =Q1-1.5*IQR
upper_limit =Q1+1.5*IQR
lower_limit, upper_limit
df_no_outlier = df[(df.Height>lower_limit)&(df.Height< upper_limit)]
df_no_outlier

```

## 7. Check for Categorical columns and perform encoding

```
In [17]: 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.Surname=le.fit_transform(df.Length)
df.Gender=le.fit_transform(df.Height)
df.head()
```

C:\Users\ELCOT\AppData\Local\Temp\ipykernel\_3712\3809871668.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.Surname=le.fit\_transform(df.Length)

C:\Users\ELCOT\AppData\Local\Temp\ipykernel\_3712\3809871668.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.Gender=le.fit\_transform(df.Height)

Out[17]:

	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.

```
In [26]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

## 8.independent variable-x

df_main=pd.read_csv('abalone.csv')
df_main.head()
x=df_main.drop(columns=['Rings'],axis=1)
x.head()
```

Out[26]:

•

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055

```
In [25]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

## 8.dependent variable-y

df_main=pd.read_csv('abalone.csv')

df_main.head()
x=df_main.drop(columns=['Rings'],axis=1)
x.head()
y=df_main.Rings
y
```

Out[25]:

0	15
1	7
2	9
3	10
4	7
...	...
4172	11
4173	10
4174	9
4175	10
4176	12

Name: Rings, Length: 4177, dtype: int64

## 9. Scale the independent variables.

```
In [27]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

df_main=pd.read_csv('abalone.csv')
df_main.head()
X=df_main.drop(columns=['Shell weight'],axis=1)
X.head()

##9.Scaling

X_train = pd.DataFrame(X)
X_train.head()
```

```
Out[27]:
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	7

## 10. Split the data into training and testing.

```
In [28]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

##10.Training and Testing
y=df_main.Length
y
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
print('x_train.shape:',X_train.shape)
print('y_train.shape:',y_train.shape)
print('x_test.shape:',X_test.shape)
print('y_test.shape:',y_test.shape)
```

```
x_train.shape: (3132, 8)
y_train.shape: (3132,)
x_test.shape: (1045, 8)
y_test.shape: (1045,)
```

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

```
Out[47]:
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
1794	1	0.575	0.460	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
```

	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  2 15  5  1  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  7 32 27  7  1  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  2 11 25 70 30  5  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  5  9 29 32 15  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  2  5  9 20 13  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]]
```



## 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