ASSIGNMET 4

1. Download the dataset.

Importing libraries

```
import numpy as pn
import pandas as dp
import matplotlib.pyplot as tlp
%matplotlib inline
import seaborn as ss
```

2. loading the dataset

```
from google.colab import files
upload = files.upload()
```

```
a = dp.read_csv('abalone.csv')
```

```
a.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

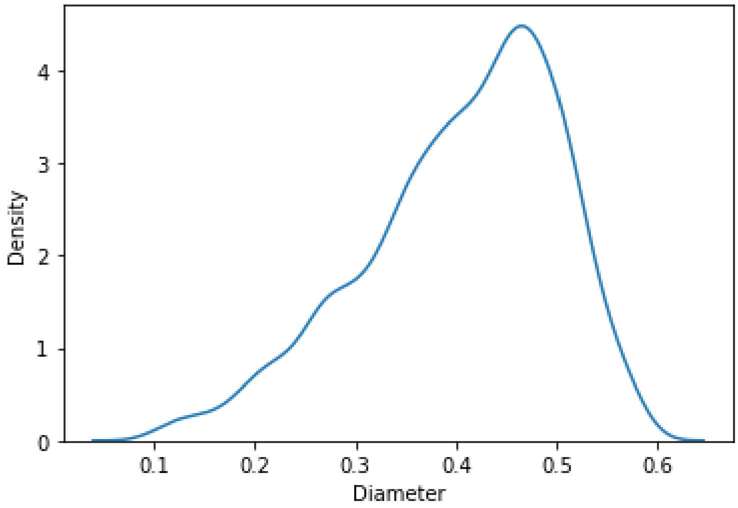
```
a['age'] = a['Rings']+1.5
a = a.drop('Rings',axis = 1)
```

3. Performing Visualizations.

1. univariate Analysis.

```
ss.kdeplot(a['Diameter'])
```

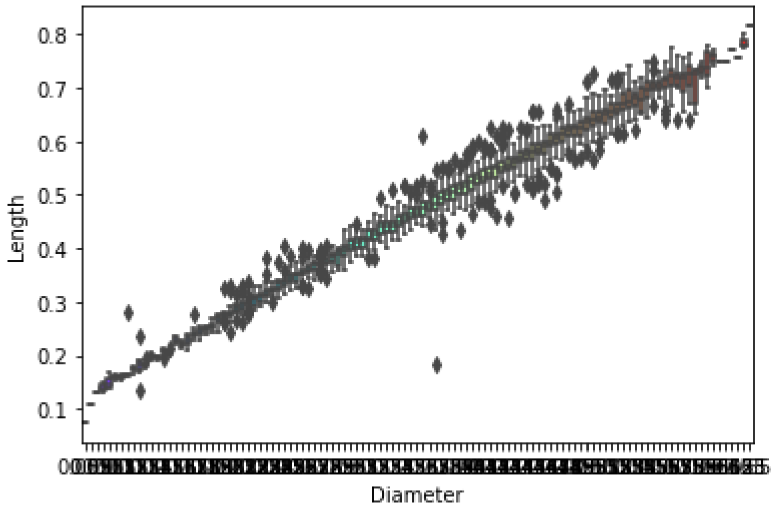
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f36f75625d0>
```



2. Bi-Variate Analysis

```
ss.boxplot(x=a.Diameter,y=a.Length,palette='rainbow')
```

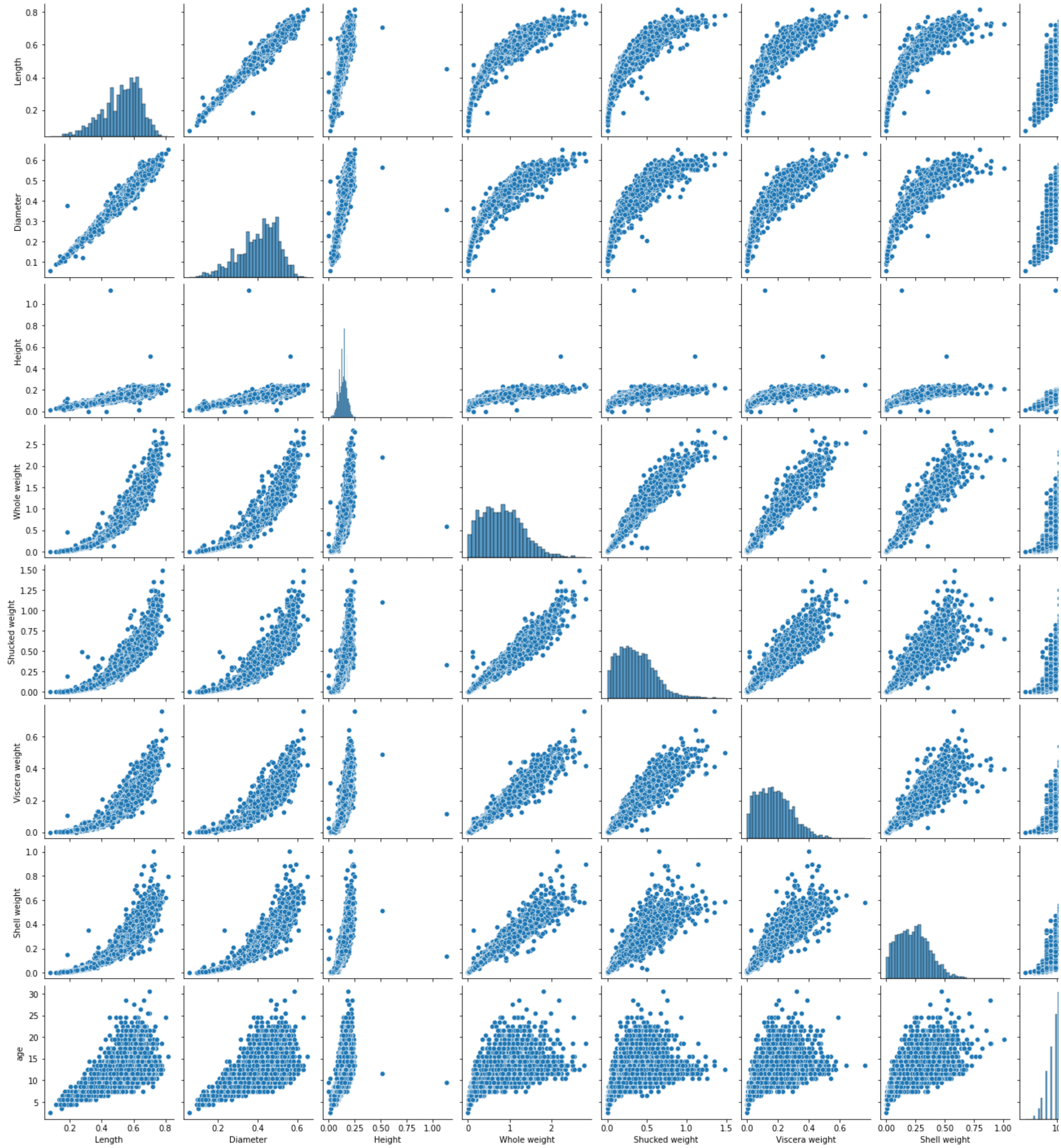
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f36ff791dd0>
```



3. Multi-Variate Analysis

```
ss.pairplot(a)
```

<seaborn.axisgrid.PairGrid at 0x7f36ffc11550>



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

```
a.info()
```

```
<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   age             4177 non-null   float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB
```

```
a['Diameter'].describe()
```

```
count    4177.000000
mean        0.407881
std         0.099240
min         0.055000
25%         0.350000
50%         0.425000
75%         0.480000
max         0.650000
Name: Diameter, dtype: float64
```

```
a['Sex'].value_counts()
```

```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
```

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

```
a.isnull()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	..
4172	False	False	False	False	False	False	False	False
4173	False	False	False	False	False	False	False	False
4174	False	False	False	False	False	False	False	False
4175	False	False	False	False	False	False	False	False
4176	False	False	False	False	False	False	False	False

4177 rows × 9 columns

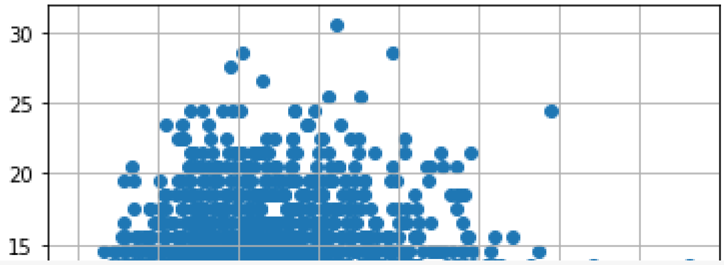
```
a.isnull().sum()
```

```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
age               0
dtype: int64
```

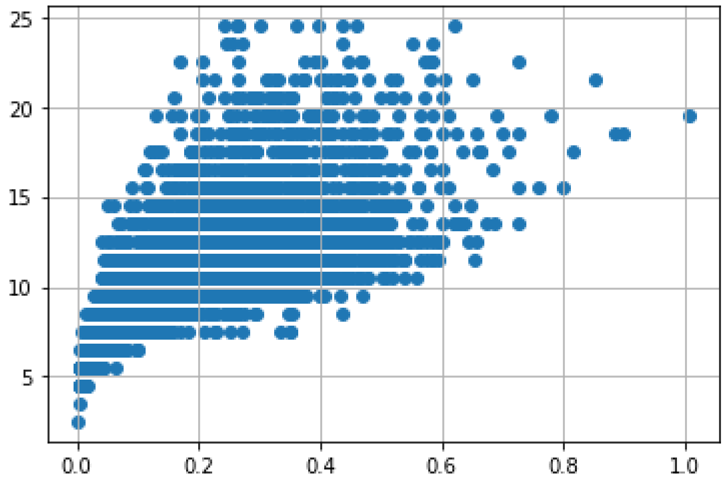
6. Find the outliers and replace them outliers.

```
# outlier handling
a = dp.get_dummies(a)
dummy_a = a
```

```
var = 'Viscera weight'
tlp.scatter(x = a[var], y = a['age'])
tlp.grid(True)
```

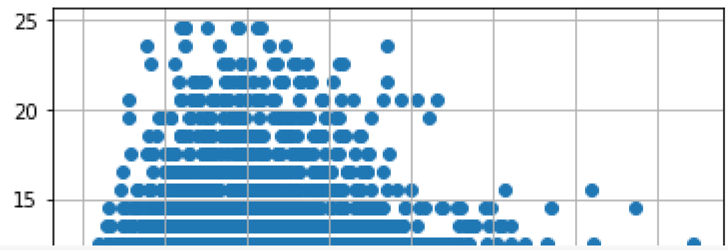



```
a.drop(a[(a['Viscera weight'] > 0.5) &
          (a['age'] < 20)].index, inplace = True)
a.drop(a[(a['Viscera weight']<0.5) & (
```



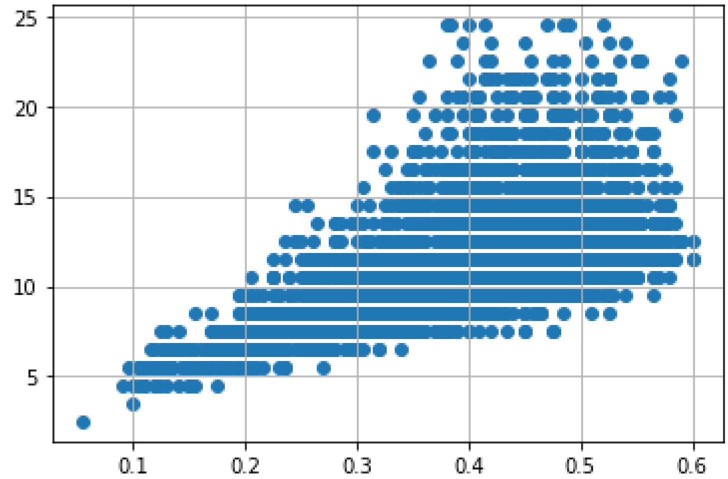

```
a.drop(a[(a['Shell weight'] > 0.6) &
          (a['age'] < 25)].index, inplace = True)
a.drop(a[(a['Shell weight']<0.8) & (
a['age'] > 25)].index, inplace = True)
```

```
var = 'Shucked weight'
tlp.scatter(x = a[var], y =a['age'])
tlp.grid(True)
```




```
a.drop(a[(a['Whole weight'] >= 2.5) &
          (a['age'] < 25)].index, inplace = True)
a.drop(a[(a['Whole weight']<2.5) & (
a['age'] > 25)].index, inplace = True)
```

```
var = 'Diameter'
tlp.scatter(x = a[var], y = a['age'])
tlp.grid(True)
```

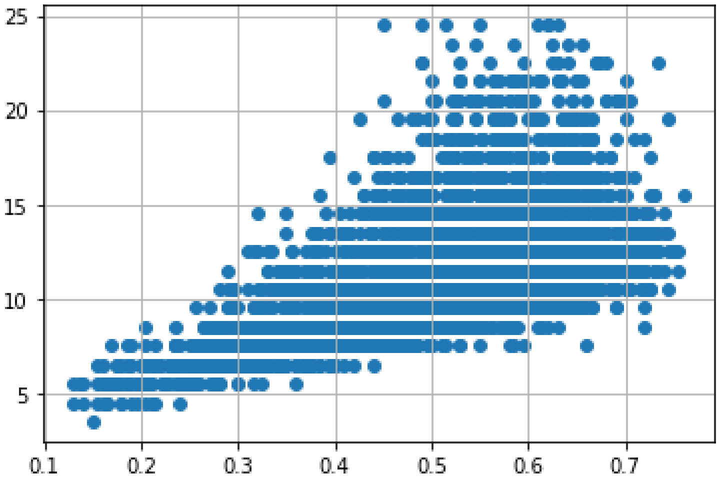



```
a.drop(a[(a['Diameter'] <0.1) &
          (a['age'] < 5)].index, inplace = True)
a.drop(a[(a['Diameter']<0.6) & (
a['age'] > 25)].index, inplace = True)
a.drop(a[(a['Diameter']>=0.6) & (
a['age'] < 25)].index, inplace = True)
```

```
var = 'Height'
tlp.scatter(x = a[var], y = a['age'])
tlp.grid(True)
```

```
a.drop(a[(a['Height'] > 0.4) &
          (a['age'] < 15)].index, inplace = True)
a.drop(a[(a['Height']<0.4) & (
a['age'] > 25)].index, inplace = True)
```

```
var = 'Length'
tlp.scatter(x = a[var], y = a['age'])
tlp.grid(True)
```




```
a.drop(a[(a['Length'] <0.1) &
          (a['age'] < 5)].index, inplace = True)
a.drop(a[(a['Length']<0.8) & (
a['age'] > 25)].index, inplace = True)
a.drop(a[(a['Length']>=0.8) & (a['age'] < 25)].index, inplace = True)
```

7. Check for Categorical columns and perform encoding.

```
numerical_features = a.select_dtypes(include = [pn.number]).columns
categorical_features = a.select_dtypes(include = [pn.object]).columns
```

```
numerical_features
```

```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age', 'Sex_F', 'Sex_I', 'Sex_M'],
      dtype='object')
```

```
categorical_features
```

```
Index([], dtype='object')
```

Encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
print(a.Length.value_counts())
```

```
0.550    93
0.575    93
0.625    93
0.580    92
0.600    86
         ..
0.755     2
0.220     2
0.150     1
0.135     1
0.760     1
Name: Length, Length: 126, dtype: int64
```

8. Spliting the data into dependent and independent variables.

```
x=a.iloc[:,:5]
x
```

	Length	Diameter	Height	Whole weight	Shucked weight
0	0.455	0.365	0.095	0.5140	0.2245
1	0.350	0.265	0.090	0.2255	0.0995
2	0.530	0.420	0.135	0.6770	0.2565
3	0.440	0.365	0.125	0.5160	0.2155
4	0.330	0.255	0.080	0.2050	0.0895
...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700
4173	0.590	0.440	0.135	0.9660	0.4390
4174	0.600	0.475	0.205	1.1760	0.5255
4175	0.625	0.485	0.150	1.0945	0.5310
4176	0.710	0.555	0.195	1.9485	0.9455

4096 rows × 5 columns

```
y=a.iloc[:,:5]
y
```

	Length	Diameter	Height	Whole weight	Shucked weight
0	0.455	0.365	0.095	0.5140	0.2245
1	0.350	0.265	0.090	0.2255	0.0995
2	0.530	0.420	0.135	0.6770	0.2565
3	0.440	0.365	0.125	0.5160	0.2155
4	0.330	0.255	0.080	0.2050	0.0895
...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700
4173	0.590	0.440	0.135	0.9660	0.4390
4174	0.600	0.475	0.205	1.1760	0.5255

9. Scale the independent variables.

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x_train=ss.fit_transform(x_train)
```

```
mlrpred=mlr.predict(x_test[0:9])
```

```
mlrpred
```

```
array([[0.41  , 0.31  , 0.125 , 0.3595, 0.1415],
       [0.585 , 0.435 , 0.14  , 0.6955, 0.3085],
       [0.575 , 0.43  , 0.13  , 0.7425, 0.2895],
       [0.67  , 0.525 , 0.165 , 1.6085, 0.682 ],
       [0.645 , 0.51  , 0.2   , 1.5675, 0.621 ],
       [0.7   , 0.535 , 0.16  , 1.7255, 0.63  ],
       [0.41  , 0.325 , 0.1   , 0.3245, 0.132 ],
       [0.58  , 0.425 , 0.15  , 0.844 , 0.3645],
       [0.465 , 0.375 , 0.135 , 0.6   , 0.2225]])
```

Double-click (or enter) to edit

10. Spliting the data into training and testing

```
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)
```

11. Building the model.

```
from sklearn.linear_model import LinearRegression
mlr=LinearRegression()
mlr.fit(x_train,y_train)
```

```
LinearRegression()
```

12. Training the model

13. Testing the model

```
x_test[0:5]
```

	Length	Diameter	Height	Whole weight	Shucked weight
3268	0.410	0.310	0.125	0.3595	0.1415
2668	0.585	0.435	0.140	0.6955	0.3085
3042	0.575	0.430	0.130	0.7425	0.2895
1040	0.670	0.525	0.165	1.6085	0.6820
184	0.645	0.510	0.200	1.5675	0.6210

```
y_test[0:5]
```

	Length	Diameter	Height	Whole weight	Shucked weight
3268	0.410	0.310	0.125	0.3595	0.1415
2668	0.585	0.435	0.140	0.6955	0.3085
3042	0.575	0.430	0.130	0.7425	0.2895
1040	0.670	0.525	0.165	1.6085	0.6820
184	0.645	0.510	0.200	1.5675	0.6210

14. Measure the performance using Metrics.

```
from sklearn.metrics import r2_score
r2_score(mlr.predict(x_test),y_test)
```

```
1.0
```