ASSIGNMENTS - 4

Assignment Date	17 October 2022
Student Name	Abdullah S
Student Roll Number	210519205001
Maximum Marks	2 Marks

Question:
Problem Statement: Abalone Age Prediction

Building a Regression Mode

l 1. Download the dataset: Dataset

2. Load the dataset into the tool.

3. Perform Below Visualizations. · Univariate Analysis · Bi-Variate Analysis Multi-Variate Analysis

4. Perform descriptive statistics on the dataset.

5. Check for Missing values and deal with them.

6. Find the outliers and replace them outliers

7. Check for Categorical columns and perform encoding.

8. Split the data into dependent and independent variables.

9. Scale the independent variables

10. Split the data into training and testing

11. Build the Model 12. Train the Model

13. Test the Model

14. Measure the performance using Metrics

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sb
```

```
ak=pd.read_csv("/content/abalone.csv")
```

```
ak.info
```

Out[32]: <bound method DataFrame.info of   Sex Length Diameter Height Whole weigh
t Shucked weight \
0   M   0.455   0.365   0.095   0.5140   0.2245
1   M   0.350   0.265   0.090   0.2255   0.0995
2   F   0.530   0.420   0.135   0.6770   0.2565
3   M   0.440   0.365   0.125   0.5160   0.2155
4   I   0.330   0.255   0.080   0.2050   0.0895
...   .
4172   F   0.565   0.450   0.165   0.8870   0.3700
4173   M   0.590   0.440   0.135   0.9660   0.4390
4174   M   0.600   0.475   0.205   1.1760   0.5255
4175   F   0.625   0.485   0.150   1.0945   0.5310
4176   M   0.710   0.555   0.195   1.9485   0.9455

      Viscera weight Shell weight Rings
0   0.1010   0.1500   15
1   0.0485   0.0700   7
2   0.1415   0.2100   9
3   0.1140   0.1550   10
4   0.0395   0.0550   7
...   .
4172   0.2390   0.2490   11
4173   0.2145   0.2605   10
4174   0.2875   0.3080   9
4175   0.2610   0.2960   10
4176   0.3765   0.4950   12

[4177 rows x 9 columns]>

```
ak.isnull().sum()
```
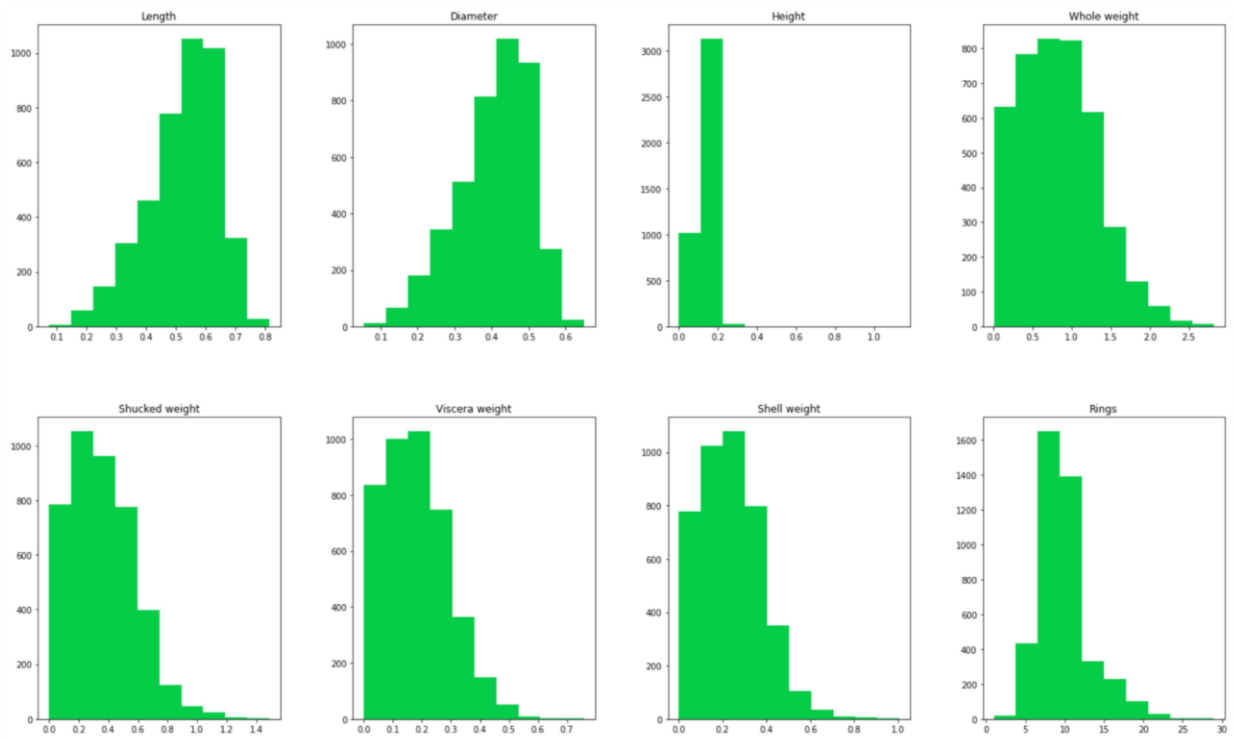
Out[33]: Sex   0
        Length   0
        Diameter   0
        Height   0
        Whole weight   0
        Shucked weight   0
        Viscera weight   0
        Shell weight   0
        Rings   0
        dtype: int64

UNI-VARIATE ANALYSIS

```
In [ ]: ak.hist(figsize=(25,15),grid=False , color="#05cc46" , layout=(2,4))
```

```
Out[34]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f98071de550>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7f98072cc1d0>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7f98070ce510>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7f9807084b10>],
               [<matplotlib.axes._subplots.AxesSubplot object at 0x7f9807046150>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7f980707d750>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7f9807034dd0>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7f9806ff8350>]],
              dtype=object)
```



```
In [ ]: numerical_features=ak.select_dtypes(include=[np.number]).columns
        categorical_features = ak.select_dtypes(include=[np.object]).columns
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarn
ing: `np.object` is a deprecated alias for the builtin `object`. To silence thi
s warning, use `object` by itself. Doing this will not modify any behavior and
is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devd
ocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdocs/release/
1.20.0-notes.html#deprecations)
```

```
In [ ]: print(numerical_features)
        categorical_features
```

Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'Rings'],
      dtype='object')

Out[36]: Index(['Sex'], dtype='object')

```
In [ ]: ak['age']=ak['Rings']+1.5
```

```
In [ ]: ak.age
```
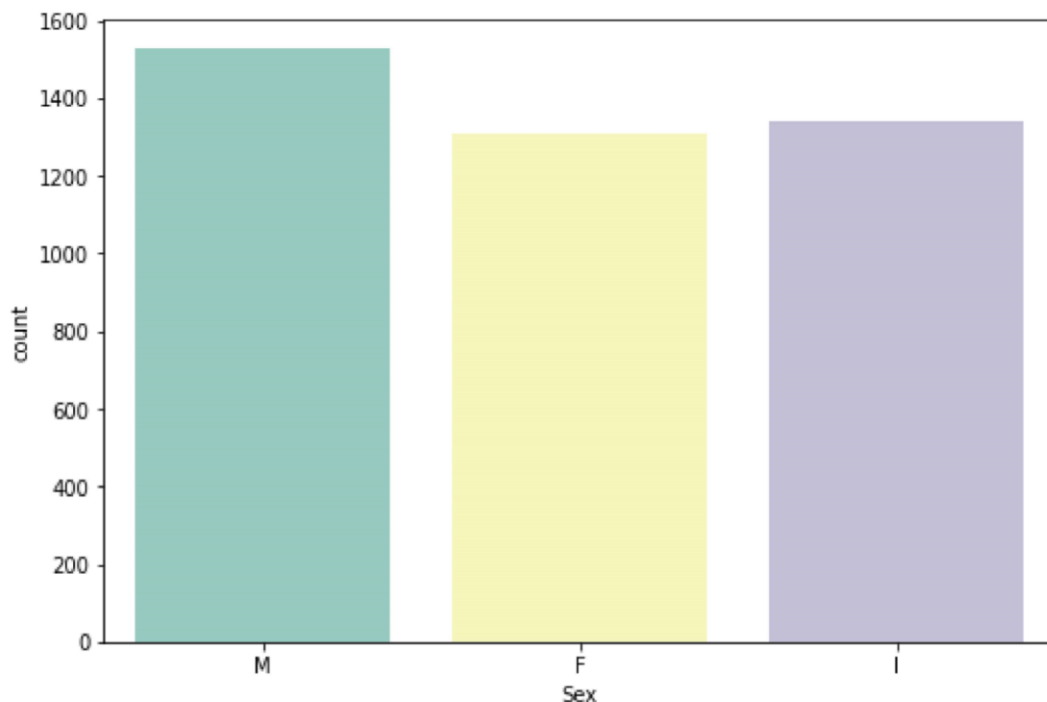
Out[38]: 0   16.5
        1   8.5
        2   10.5
        3   11.5
        4   8.5
                 ...
        4172   12.5
        4173   11.5
        4174   10.5
        4175   11.5
        4176   13.5
        Name: age, Length: 4177, dtype: float64

```
In [ ]: sb.countplot(x="Sex" , data=ak , palette='Set3')
```

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9806cc67d0>

BI-VARIATE ANALYSIS

```
In [ ]:   plt.figure(figsize = (20,7))
          sb.heatmap(ak[numerical_features].corr(),annot=True)
```
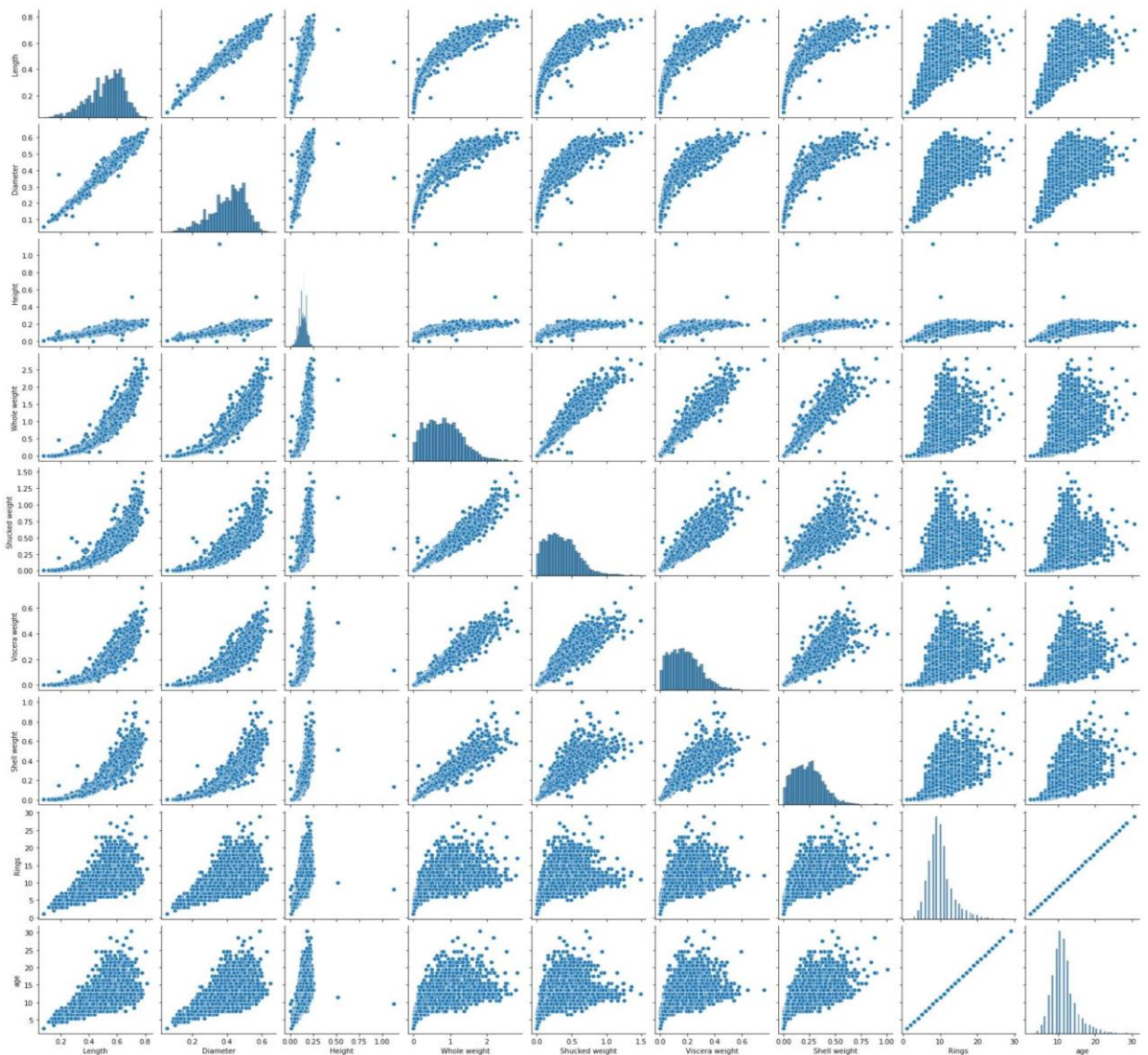
Out[40]:   <matplotlib.axes._subplots.AxesSubplot at 0x7f98097d8e90>

MULTI-VARIAT ANALYSIS

```
In [ ]:   sb.pairplot(ak)
```

Out[41]:   <seaborn.axisgrid.PairGrid at 0x7f98083d8c50>

Descriptive Statistics

```
In [ ]:  ak.describe
```

```
Out[42]:  <bound method NDFrame.describe of        Sex   Length   Diameter   Height   Whole wei
          ght   Shucked weight   \
          0       M    0.455    0.365    0.095       0.5140              0.2245
          1       M    0.350    0.265    0.090       0.2255              0.0995
          2       F    0.530    0.420    0.135       0.6770              0.2565
          3       M    0.440    0.365    0.125       0.5160              0.2155
          4       I    0.330    0.255    0.080       0.2050              0.0895
          ...    ..    ...      ...      ...         ...                 ...
          4172    F    0.565    0.450    0.165       0.8870              0.3700
          4173    M    0.590    0.440    0.135       0.9660              0.4390
          4174    M    0.600    0.475    0.205       1.1760              0.5255
          4175    F    0.625    0.485    0.150       1.0945              0.5310
          4176    M    0.710    0.555    0.195       1.9485              0.9455

                  Viscera weight   Shell weight   Rings    age
          0              0.1010         0.1500       15    16.5
          1              0.0485         0.0700        7     8.5
          2              0.1415         0.2100        9    10.5
          3              0.1140         0.1550       10    11.5
          4              0.0395         0.0550        7     8.5
          ...            ...            ...         ...     ...
          4172           0.2390         0.2490       11    12.5
          4173           0.2145         0.2605       10    11.5
          4174           0.2875         0.3080        9    10.5
          4175           0.2610         0.2960       10    11.5
          4176           0.3765         0.4950       12    13.5

          [4177 rows x 10 columns]>
```

MISSING VALUES

```
In [ ]:  Missing_Values=ak.isnull().sum()
```
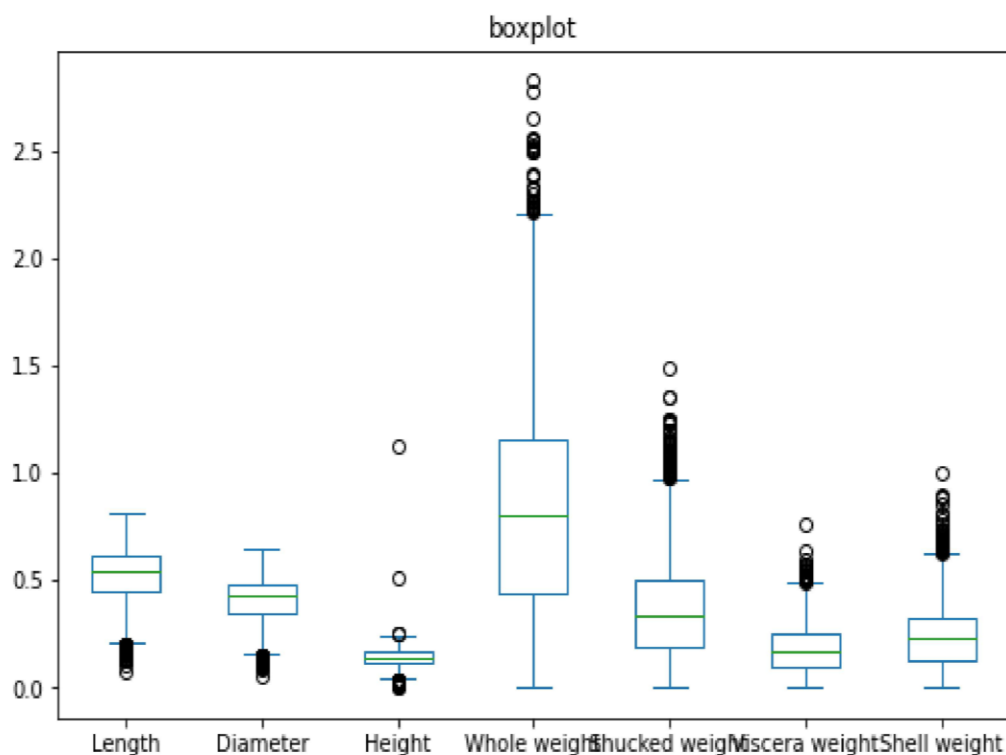
```
In [ ]:  Missing_Values
```

```
Out[46]:  Sex                0
          Length             0
          Diameter           0
          Height             0
          Whole weight       0
          Shucked weight     0
          Viscera weight     0
          Shell weight       0
          Rings              0
          age                0
          dtype: int64
```

FINDING OUTLIERS AND REPLACING THEM

In [ ]:
```python
plt.rcParams['figure.figsize']=[8.50,5.50]
ax = ak[['Length','Diameter','Height','Whole weight','Shucked weight','Viscera we
```


boxplot

In [ ]:
```python
for i in range(1,8):
    Q1 = ak.iloc[:,i].quantile(0.25)
    Q3 = ak.iloc[:,i].quantile(0.75)
    IQR = Q3 - Q1
    whisker_width = 1.5
    lower_whisker = Q1 - (whisker_width*IQR)
    upper_whisker = Q3 + (whisker_width*IQR)
    ak.iloc[:,i] = np.where(ak.iloc[:,i]>upper_whisker,upper_whisker,np.where(ak.
```

```
In [ ]: plt.rcParams["figure.figsize"] = [8.50, 5.50]
        ay = ak[['Length','Diameter','Height','Whole weight','Shucked weight','Viscera we
```


boxplot
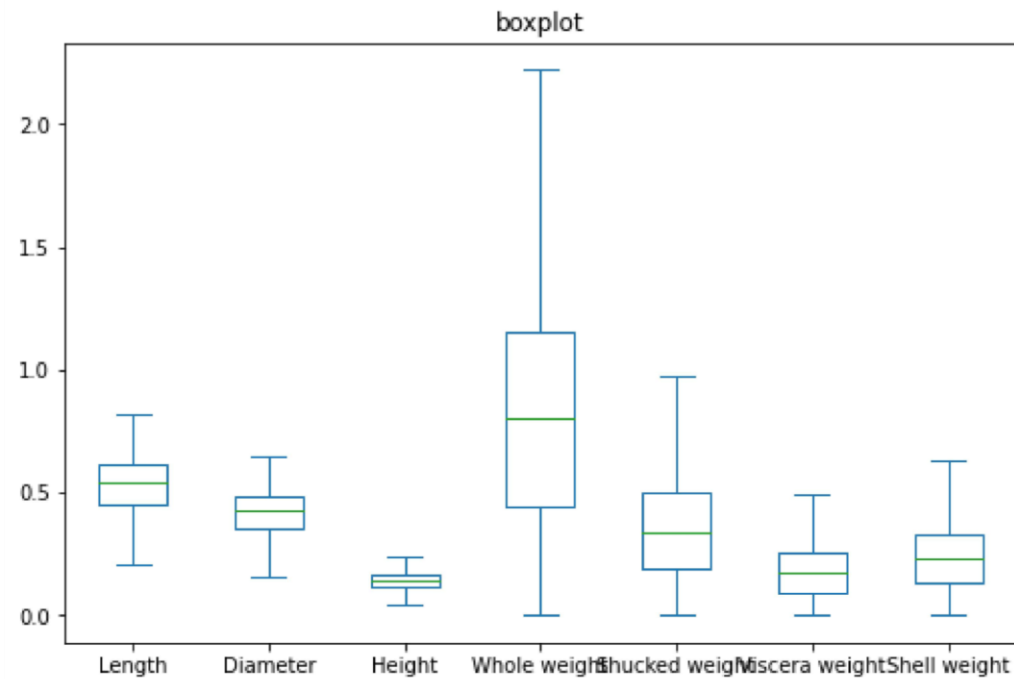
Checking for Categorical columns and performing encoding

```
In [ ]: cate_data = ak.select_dtypes(include=['object']).copy()
```

```
In [ ]: from sklearn.preprocessing import LabelBinarizer

        lb = LabelBinarizer()
        lb_results = lb.fit_transform(cate_data['Sex'])
        lb_results_df = pd.DataFrame(lb_results, columns=lb.classes_)

        print(lb_results_df.head())
```

```
     F  I  M
0    0  0  1
1    0  0  1
2    1  0  0
3    0  0  1
4    0  1  0
```

```
In [ ]: result_df = pd.concat([cate_data, lb_results_df], axis=1)

        print(result_df.head())
```

```
    Sex  F  I  M
0     M  0  0  1
1     M  0  0  1
2     F  1  0  0
3     M  0  0  1
4     I  0  1  0
```

8

Splitting into dependent and independent variables

```
In [ ]: x=ak.iloc[:,1:2].values
        x
```

```
Out[64]: array([[0.455],
                [0.35 ],
                [0.53 ],
                ...,
                [0.6  ],
                [0.625],
                [0.71 ]])
```

```
In [ ]: y=ak.age
        y
```

```
Out[54]: 0        16.5
         1         8.5
         2        10.5
         3        11.5
         4         8.5
                 ...
         4172     12.5
         4173     11.5
         4174     10.5
         4175     11.5
         4176     13.5
         Name: age, Length: 4177, dtype: float64
```

```
In [ ]: x.shape
```

```
Out[55]: (4177, 7)
```

```
In [ ]: y.shape
```

```
Out[56]: (4177,)
```

Scaling the independent variables

```
In [ ]: print ("\n ORIGIONAL VALUES: \n\n", x,y)
```

```
         ORIGIONAL VALUES:

         [[0.455]
         [0.35 ]
         [0.53 ]
         ...
         [0.6  ]
         [0.625]
         [0.71 ]] 0        16.5
         1         8.5
         2        10.5
         3        11.5
         4         8.5
                 ...
         4172     12.5
         4173     11.5
         4174     10.5
         4175     11.5
         4176     13.5
         Name: age, Length: 4177, dtype: float64
```

```python
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler(feature_range =(0, 1))
new_y= min_max_scaler.fit_transform(x,y)
print ("\n VALUES AFTER MIN MAX SCALING: \n\n", new_y)
```

```
VALUES AFTER MIN MAX SCALING:

[[0.4122449 ]
 [0.24081633]
 [0.53469388]
 ...
 [0.64897959]
 [0.68979592]
 [0.82857143]]
```

SPLITING THE DATA

```python
X = ak.drop('age', axis = 1)
y = ak['age']
```

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
x_train, x_test, y_train, y_test = train_test_split(new_y,x,test_size=0.2)
```

```python
x_train.shape,x_test.shape
```

Out[78]: ((3341, 1), (836, 1))

```python
y_train.shape,y_test.shape
```

Out[79]: ((3341, 1), (836, 1))

Building and training the model

```python
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[80]: LinearRegression()

Testing the model

```python
y_pred=model.predict(x_test)
```

Measure the performance using Metrics

```python
print('R Squared value:', r2_score(y_test,y_pred))
print('Mean Absolute Error:', mean_absolute_error(y_test,y_pred))
print('Mean Squared Error:', mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
R Squared value: 1.0
Mean Absolute Error: 2.905039313836758e-17
Mean Squared Error: 2.853878008501086e-33
Root Mean Squared Error: 5.342169979045113e-17
```