ASSIGNMENT-4

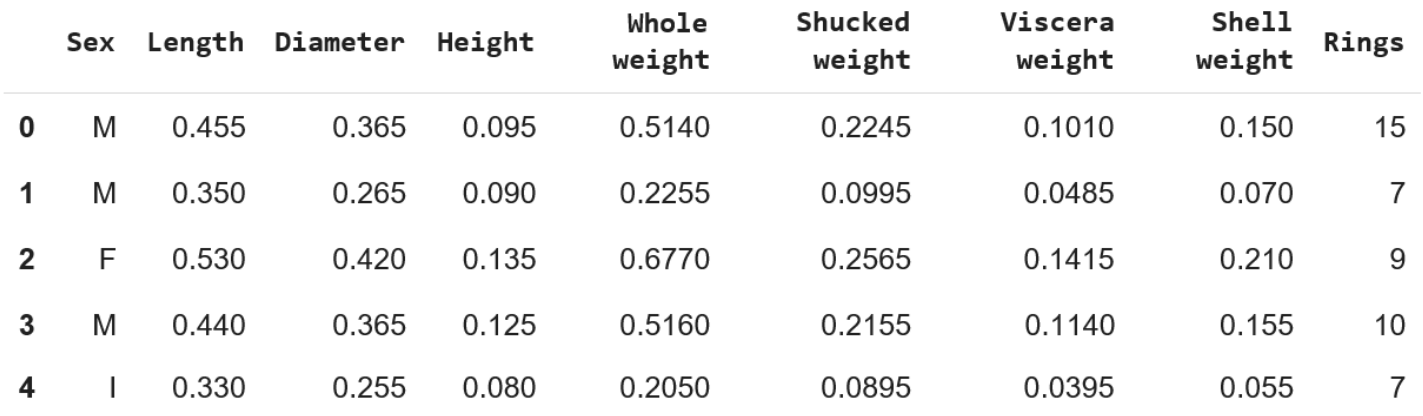
(ABALONE AGE PREDICTION)

NAME : SARAVANASETHU G

REG : 2019503046

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
data=pd.read_csv("abalone.csv")
```

```
data.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

```
data.shape
```

```
                  (4177, 9)
```

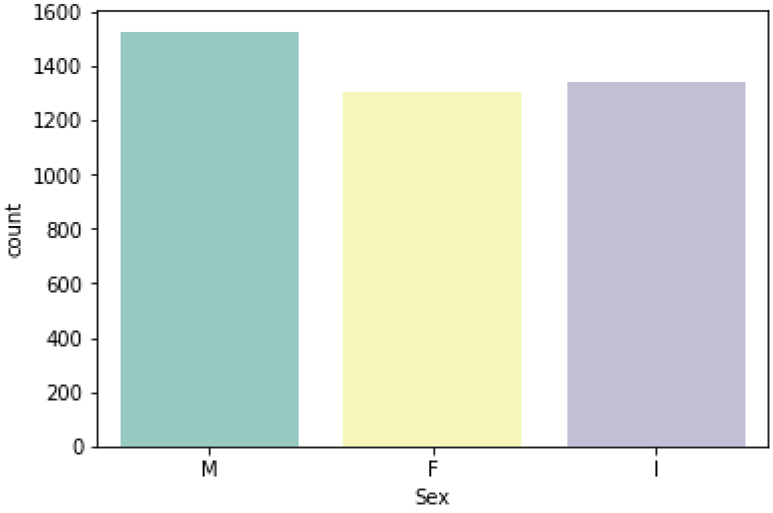
Univariate analysis

```
sns.boxplot(data.Length)
```

```
     usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
       FutureWarning
```

```
sns.countplot(x = 'Sex', data = data, palette = 'Set3')
```

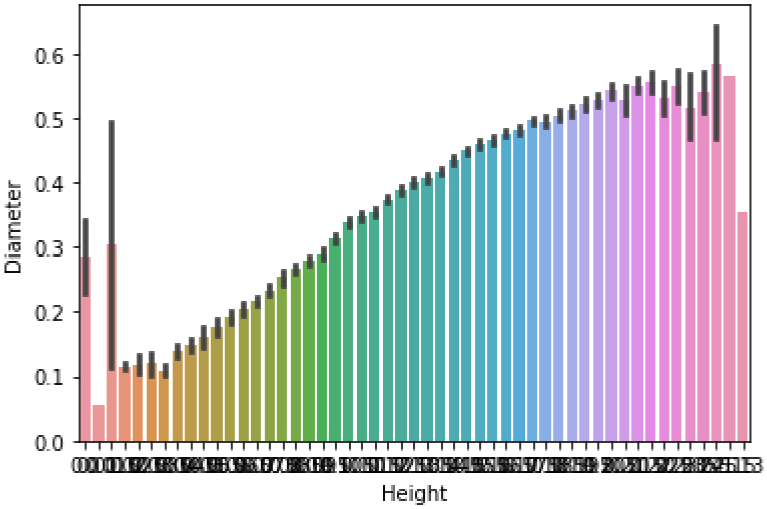
```
     <matplotlib.axes._subplots.AxesSubplot at 0x7f26821807d0>
```



Bivariate analysis

```
sns.barplot(x=data.Height,y=data.Diameter)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2682167990>
```

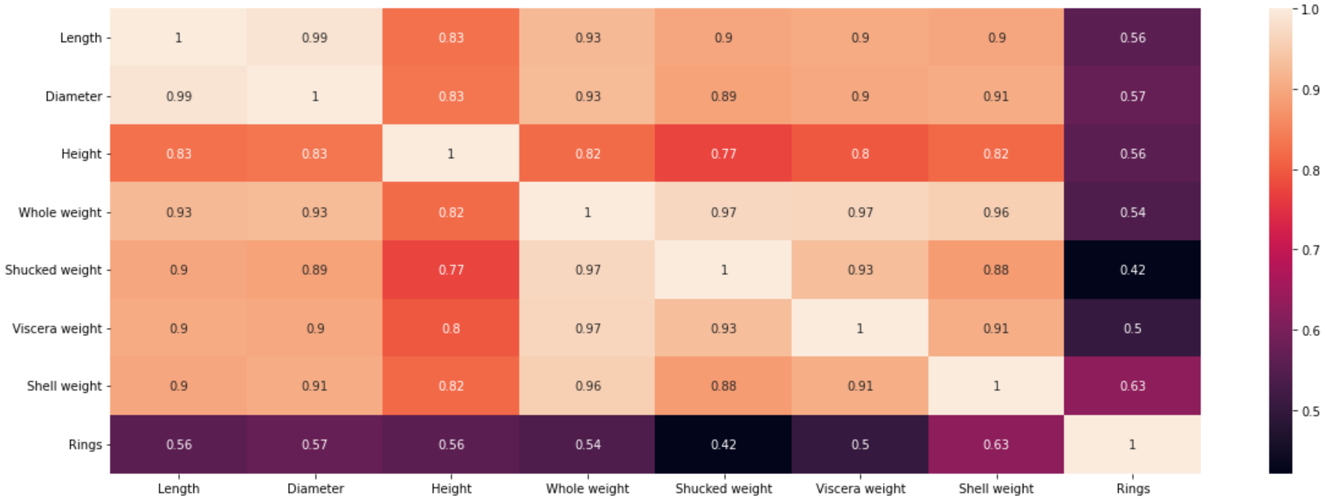


```
numerical_features = data.select_dtypes(include = [np.number]).columns
categorical_features = data.select_dtypes(include = [np.object]).columns
```

```
 /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: `np
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/rele
```

```
plt.figure(figsize = (20,7))
sns.heatmap(data[numerical_features].corr(),annot = True)
```

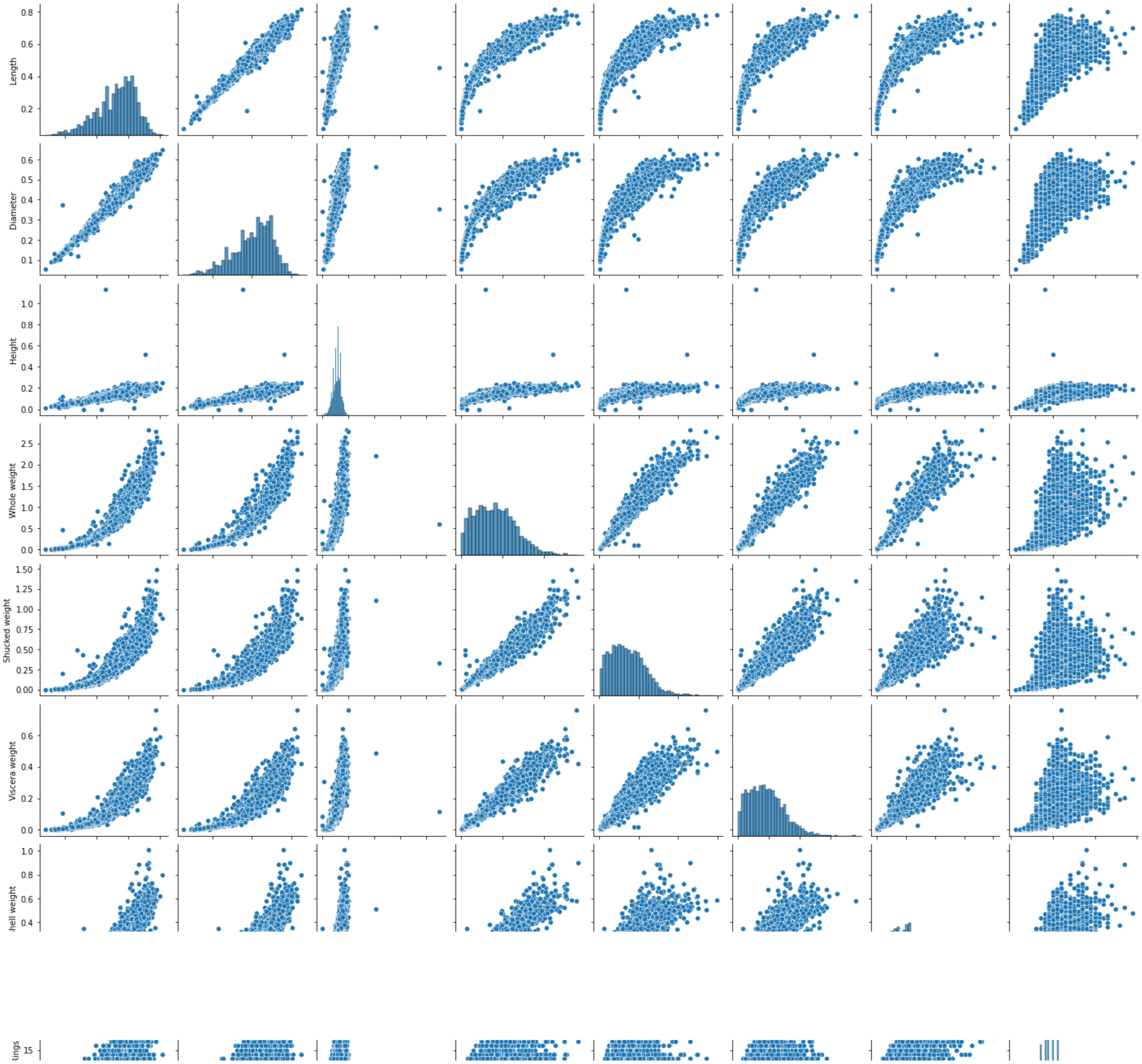
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f268401db50>
```



Multivariate analysis

```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7f26843078d0>
```



Performing descriptive statistics on the dataset.

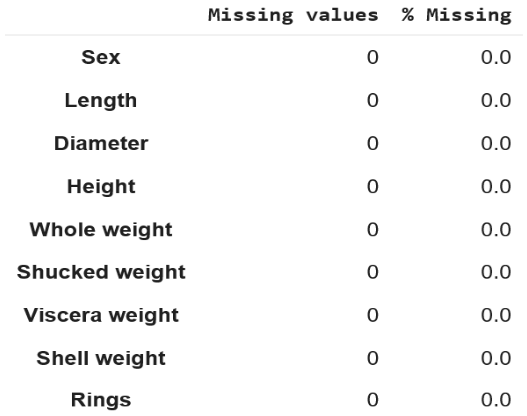
```
data['Height'].describe()

       count    4177.000000
       mean        0.139516
       std         0.041827
       min         0.000000
       25%         0.115000
       50%         0.140000
       75%         0.165000
       max         1.130000
       Name: Height, dtype: float64


  data['Height'].mean()

       0.13951639932966242
  data.max()

       Sex                     M
       Length              0.815
       Diameter             0.65
       Height               1.13
       Whole weight       2.8255
       Shucked weight      1.488
       Viscera weight       0.76
       Shell weight        1.005
       Rings                  29
       dtype: object

  data['Sex'].value_counts()

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

  data['Shucked weight'].kurtosis()

       0.5951236783694207


  data['Diameter'].median()

       0.425


  data['Shucked weight'].skew()

       0.7190979217612694
```

Missing Values

```
data.isna().any()
```

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

```
missing_values = data.isnull().sum().sort_values(ascending = False)
percentage_missing_values = (missing_values/len(data))*100
pd.concat([missing_values, percentage_missing_values], axis = 1, keys= ['Missing values', '%
```



	Missing values	% Missing
Sex	0	0.0
Length	0	0.0
Diameter	0	0.0
Height	0	0.0
Whole weight	0	0.0
Shucked weight	0	0.0
Viscera weight	0	0.0
Shell weight	0	0.0
Rings	0	0.0

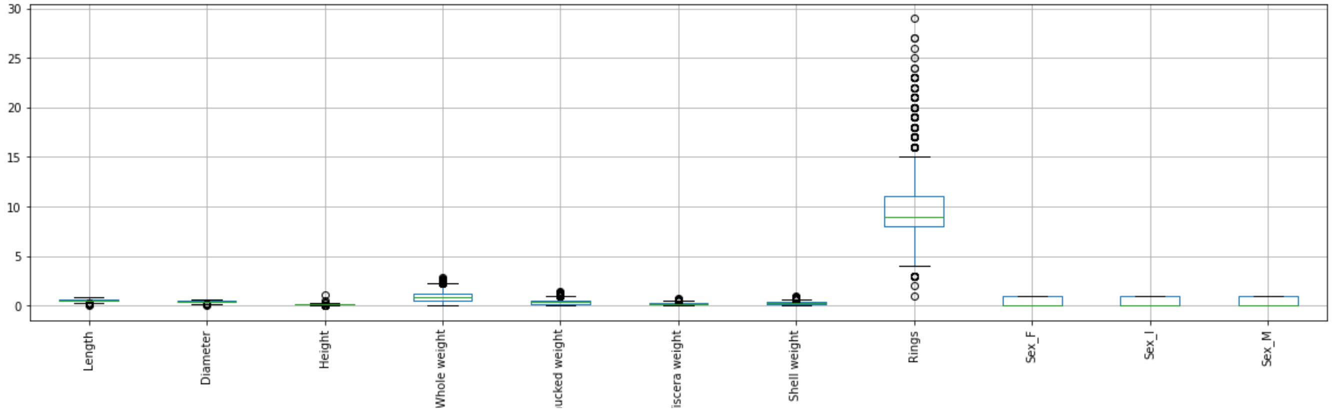
Find the outliers

```
q1=data.Rings.quantile(0.25)
q2=data.Rings.quantile(0.75)
iqr=q2-q1
print(iqr)
```

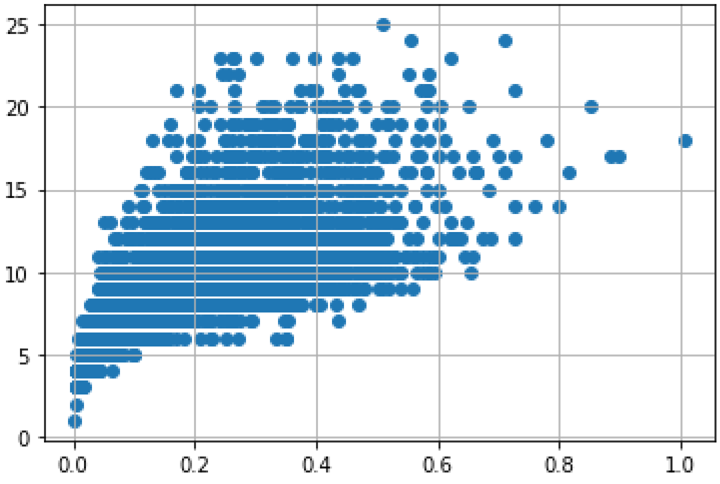
```
    3.0
```

```
data = pd.get_dummies(data)
dummy_data = data
data.boxplot( rot = 90, figsize=(20,5))
```

```
    <matplotlib.axes._subplots.AxesSubplot at 0x7f267fe90810>
```



```
data['age'] = data['Rings']
data = data.drop('Rings', axis = 1)


data.drop(data[(data['Viscera weight']> 0.5) & (data['age'] < 20)].index, inplace=True)
data.drop(data[(data['Viscera weight']<0.5) & (data['age'] > 25)].index, inplace=True)


var = 'Shell weight'
plt.scatter(x = data[var], y = data['age'])
plt.grid(True)
```



Check for Categorical columns and perform encoding.

```
numerical_features = data.select_dtypes(include = [np.number]).columns
categorical_features = data.select_dtypes(include = [np.object]).columns
```

```
    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: `np
    Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/rele
```

```
abalone_numeric = data[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight','Vis
abalone_numeric.head()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	0	0	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	0	0	1
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	1	0	0
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	0	0	1

```
x = data.iloc[:, 0:1].values
y = data.iloc[:, 1]
y
```

```
     0       0.365
     1       0.265
     2       0.420
     3       0.365
     4       0.255
             ...
     4172    0.450
     4173    0.440
     4174    0.475
     4175    0.485
     4176    0.555
     Name: Diameter, Length: 4150, dtype: float64
```

Scale the independent variables

```
print ("ORIGINAL VALUES: \n", x,y)
```

```
ORIGINAL VALUES:

      [[0.455]
      [0.35 ]
      [0.53 ]
      ...
      [0.6  ]
      [0.625]
      [0.71 ]] 0       0.365
     1       0.265
     2       0.420
     3       0.365
     4       0.255
             ...
     4172    0.450
     4173    0.440
     4174    0.475
```

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

```
     VALUES AFTER MIN MAX SCALING: 
      [[0.51351351]
      [0.37162162]
      [0.61486486]
      ...
      [0.70945946]
      [0.74324324]
      [0.85810811]]
```

Split the data into training and testing

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

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectKBest
standardScale = StandardScaler()
standardScale.fit_transform(X)

selectkBest = SelectKBest()
X_new = selectkBest.fit_transform(X, y)

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size = 0.25)
X_train
```

```
     array([[0.73 , 0.55 , 0.205, ..., 1.   , 0.   , 0.   ],
            [0.395, 0.3  , 0.09 , ..., 0.   , 1.   , 0.   ],
            [0.625, 0.495, 0.175, ..., 0.   , 0.   , 1.   ],
            ...,
            [0.645, 0.51 , 0.18 , ..., 1.   , 0.   , 0.   ],
            [0.71 , 0.56 , 0.18 , ..., 1.   , 0.   , 0.   ],
            [0.46 , 0.36 , 0.135, ..., 0.   , 0.   , 1.   ]])
```

Build the Model

```
from sklearn import linear_model as lm
from sklearn.linear_model import LinearRegression
model=lm.LinearRegression()
results=model.fit(X_train,y_train)
accuracy = model.score(X_train, y_train)
print('Accuracy of the model:', accuracy)
```

```
    Accuracy of the model: 0.5305010084253585
```

```
Train the Model
```

Train the Model

```
lm = LinearRegression()
lm.fit(X_train, y_train)
y_train_pred = lm.predict(X_train)
y_train_pred
```

```
    array([19.9375,  6.9375, 10.625 , ..., 10.9375, 12.125 , 11.6875])
```

```
X_train
```

```
    array([[0.73 , 0.55 , 0.205, ..., 1.   , 0.   , 0.   ],
           [0.395, 0.3  , 0.09 , ..., 0.   , 1.   , 0.   ],
           [0.625, 0.495, 0.175, ..., 0.   , 0.   , 1.   ],
           ...,
           [0.645, 0.51 , 0.18 , ..., 1.   , 0.   , 0.   ],
           [0.71 , 0.56 , 0.18 , ..., 1.   , 0.   , 0.   ],
           [0.46 , 0.36 , 0.135, ..., 0.   , 0.   , 1.   ]])
```

```
y_train
```

```
    3081    14
    3602     5
    1805     9
    3898    10
    2555     6
            ..
    3913    11
    594     12
    1955    12
    1196    11
    2484    14
    Name: age, Length: 3112, dtype: int64
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
s = mean_squared_error(y_train, y_train_pred)
print('Mean Squared error of training set :%2f'%s)
```

```
     Mean Squared error of training set :4.775387
```

Test the Model

```
y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
y_test_pred
```

```
     array([10.875 , 10.375 ,  9.8125, ..., 12.6875, 10.375 ,  8.0625])
```

```
p = mean_squared_error(y_test, y_test_pred)
print('Mean Squared error of testing set :%2f'%p)
```

```
     Mean Squared error of testing set :4.556061
```

Measure the performance using Metrics.

```
from sklearn.metrics import r2_score
s = r2_score(y_train, y_train_pred)
print('R2 Score of training set:%.2f'%s)
```

```
     R2 Score of training set:0.53
```

```
p = r2_score(y_test, y_test_pred)
print('R2 Score of testing set:%.2f'%p)
```

```
     R2 Score of testing set:0.55
```