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


import matplotlib.pyplot as tlp
%matplotlib inline
import seaborn as ss


from google.colab import files
upload=files.upload()
df = pd.read_csv('abalone.csv')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving abalone.csv to abalone.csv

```
df.describe()
```

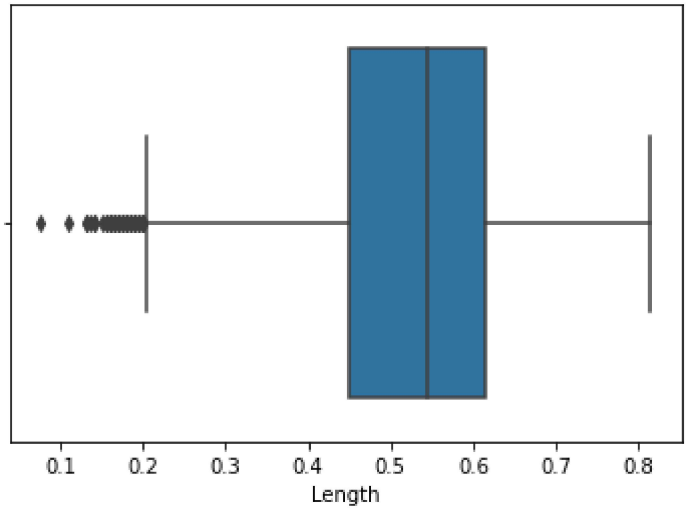
	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.

```
df.head()
```

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

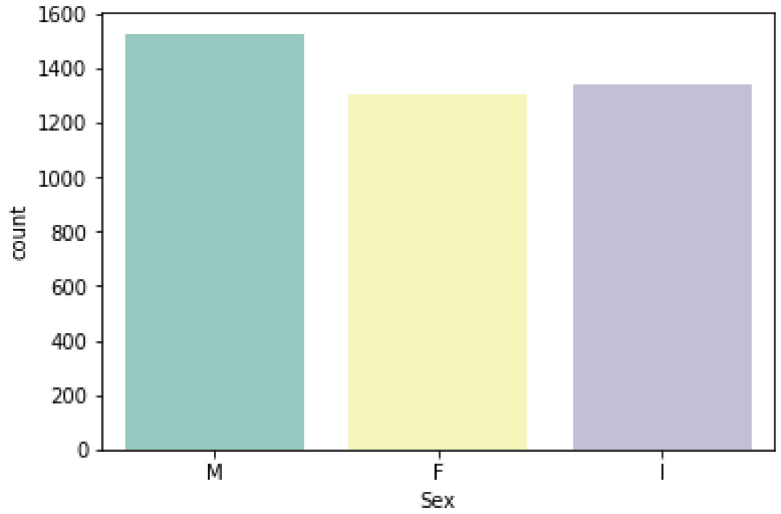
```
df['age'] = df['Rings']+1.5
df = df.drop('Rings',axis = 1)


#Perform visualisations
#Univariate analysis


sns.boxplot(df.Length)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f74c87523d0>
```



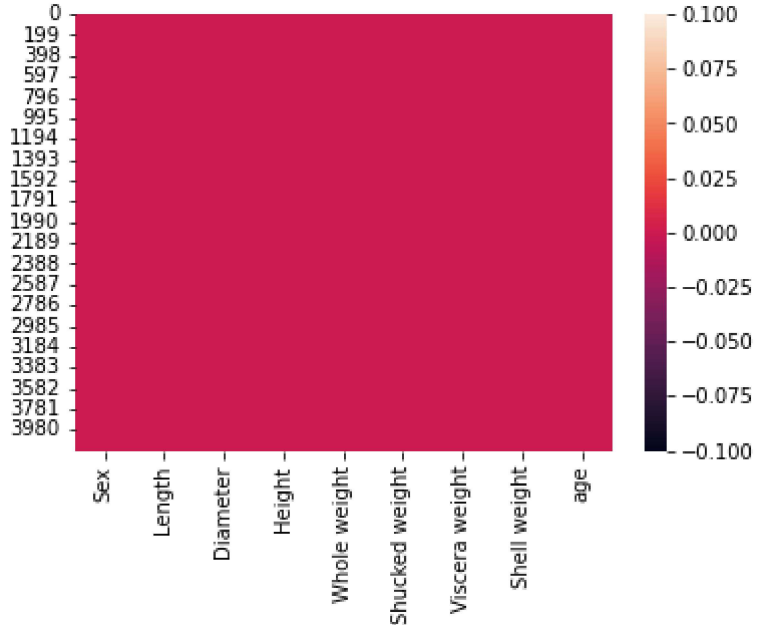

```
sns.countplot(x = 'Sex', data = df, palette = 'Set3')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f74c87b6d50>
```



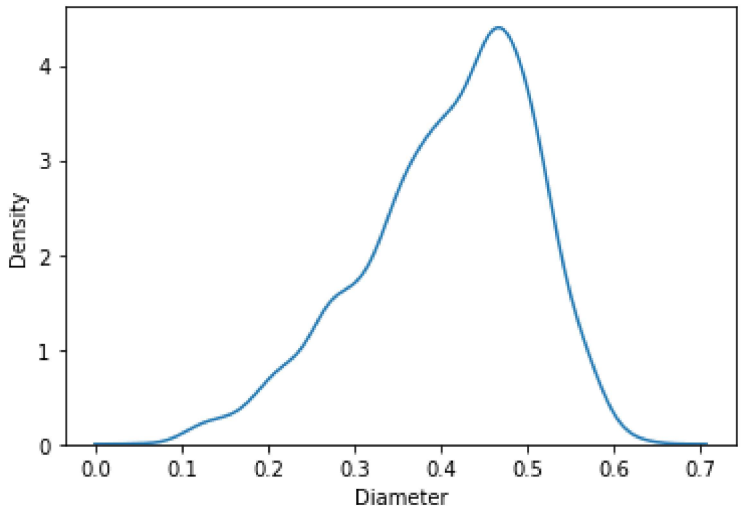

```
sns.heatmap(df.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f74c88a6cd0>
```



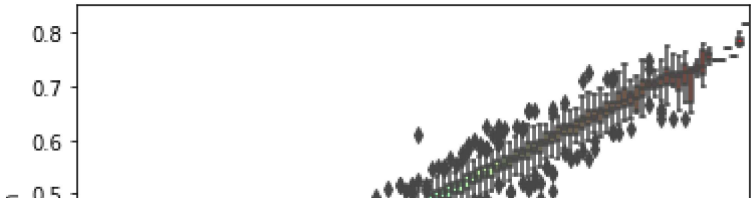

```
ss.kdeplot(df['Diameter'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f74c589b410>
```



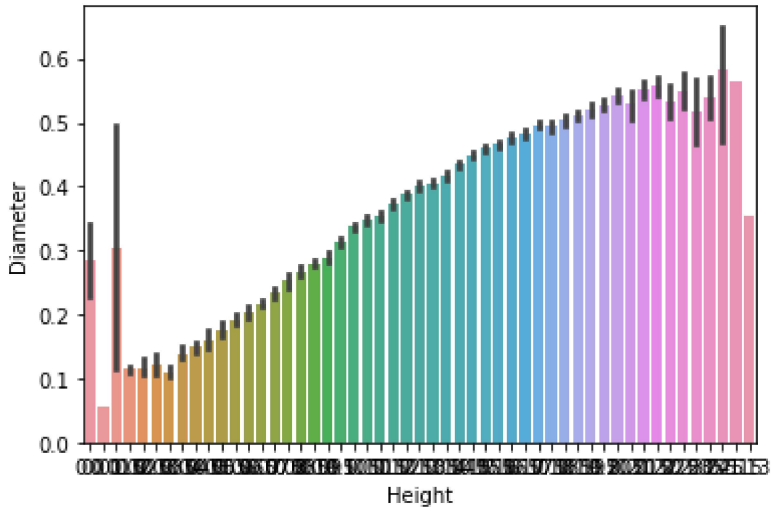

```
#Bivariate analysis


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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f74c57d5e50>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f74c4e19790>
```




```
numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.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/relea
```

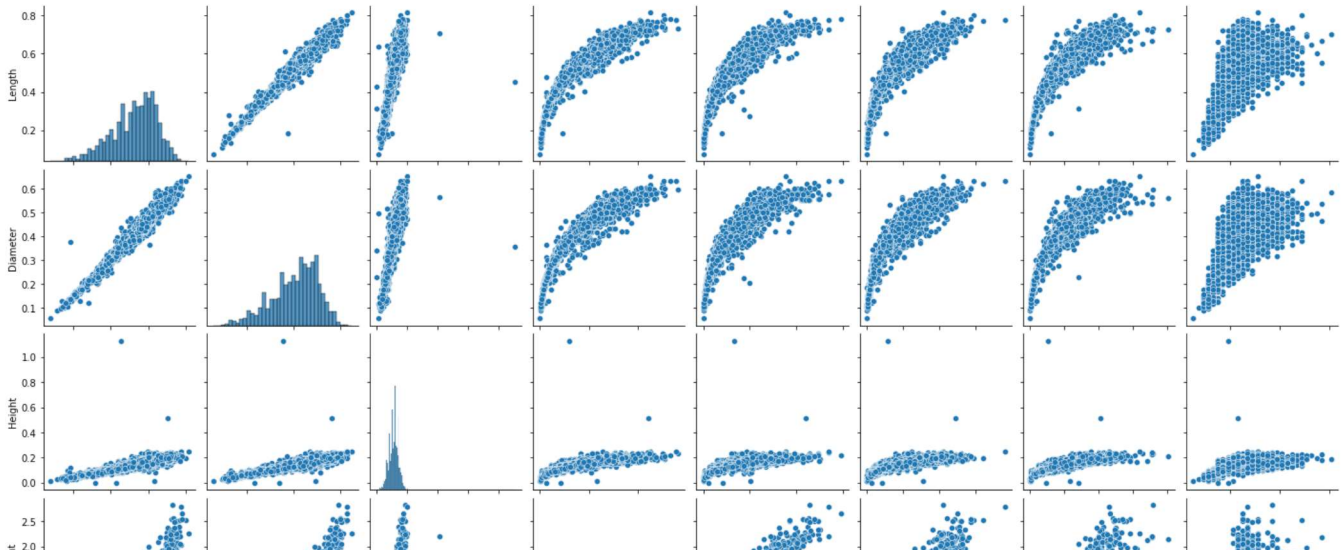
```
plt.figure(figsize = (20,7))
sns.heatmap(df[numerical_features].corr(),annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f74c5881f90>
```

#Multivariate Analysis


```
ss.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f74c4942390>
```




```
#Perform descriptive model on the dataset
```

```
df['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
```

```
df['Height'].mean()
```

```
0.13951639932966242
```

```
df.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
age                30.5
dtype: object
```

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

```
M    1528
```

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

```
df[df.Height == 0]
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
1257	I	0.430	0.34	0.0	0.428	0.2065	0.0860	0.1150	9.5
3996	I	0.315	0.23	0.0	0.134	0.0575	0.0285	0.3505	7.5

```
df['Shucked weight'].kurtosis()
```

```
0.5951236783694207
```

```
df['Diameter'].median()
```

```
0.425
```

```
df['Shucked weight'].skew()
```

```
0.7190979217612694
```

```
#Missing values
```

```
df.isna().any()
```

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

```
missing_values = df.isnull().sum().sort_values(ascending = False)
percentage_missing_values = (missing_values/len(df))*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

```
df.isnull()
```

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

4177 rows × 9 columns

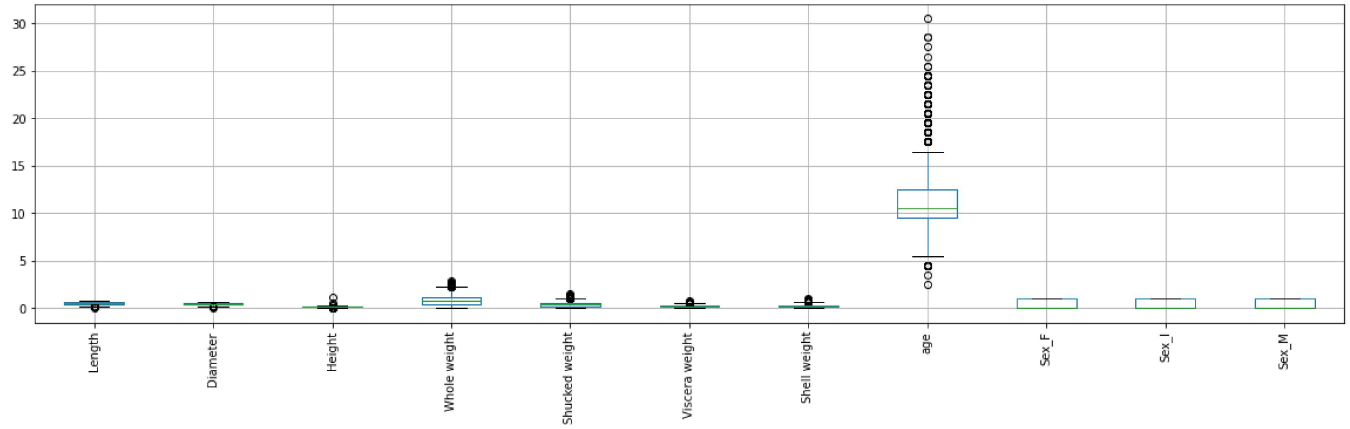
```
df.isnull().sum()
```

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

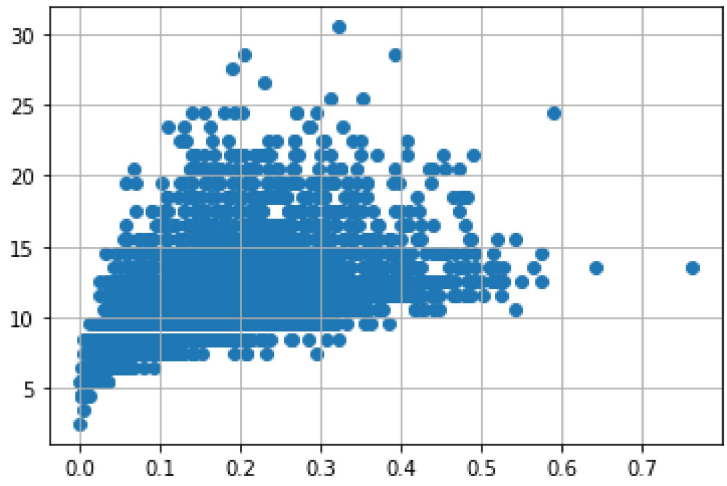
```
#Find the outliers
```

```
df = pd.get_dummies(df)
dummy_df = df
df.boxplot( rot = 90, figsize=(20,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f74c2a1ce90>
```

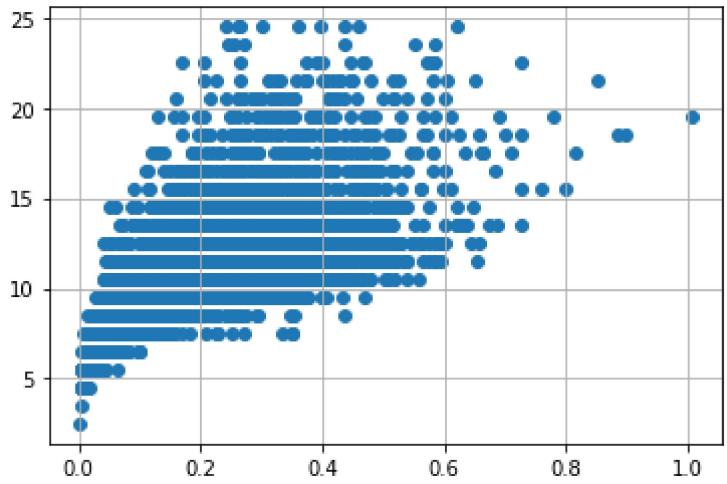



```
var = 'Viscera weight'
tlp.scatter(x = df[var], y = df['age'])
tlp.grid(True)
```




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

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




```
#Check for categorical columns and perform encoding


numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.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/relea
```

```
numerical_features
categorical_features
```

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

```
abalone_numeric = df[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight','Visce


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	16.5	0	0	1
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5	1	0	0

```
#Dependent and Independent Variables
x = df.iloc[:, 0:1].values
```

```
y = df.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: 4147, dtype: float64
```

```
#Scaling the Independent Variables
print ("\n ORIGINAL VALUES: \n\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
4175    0.485
4176    0.555
Name: Diameter, Length: 4147, dtype: float64
```

```
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:

[[0.51351351]
[0.37162162]
[0.61486486]
...
[0.70945946]
[0.74324324]
[0.85810811]]
```

```
#Split the data into Training and Testing
X = df.drop('age', axis = 1)
y = df['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([[0.525, 0.4  , 0.155, ..., 0.   , 0.   , 1.   ],
       [0.275, 0.2  , 0.075, ..., 0.   , 1.   , 0.   ],
       [0.32 , 0.24 , 0.07 , ..., 0.   , 1.   , 0.   ],
       ...,
       [0.63 , 0.505, 0.18 , ..., 0.   , 1.   , 0.   ],
       [0.35 , 0.26 , 0.095, ..., 0.   , 1.   , 0.   ],
       [0.52 , 0.405, 0.12 , ..., 1.   , 0.   , 0.   ]])
```

```
y_train
```

```
1595    10.5
2227     8.5
1995     7.5
2843    11.5
3785    10.5
        ...
3443     8.5
3028     8.5
1385    12.5
```

```
3813     9.5
473     12.5
Name: age, Length: 3110, dtype: float64
```

```
# Build the model
# Linear Regression
```

```
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: 0.5336629122727419
```

```
#Training the model
lm = LinearRegression()
lm.fit(X_train, y_train)
y_train_pred = lm.predict(X_train)
y_train_pred
```

```
array([12.34375  ,  7.6171875,  7.9140625, ..., 10.9296875,  8.4609375,
       11.46875  ])
```

```
X_train
```

```
array([[0.525, 0.4  , 0.155, ..., 0.   , 0.   , 1.   ],
       [0.275, 0.2  , 0.075, ..., 0.   , 1.   , 0.   ],
       [0.32 , 0.24 , 0.07 , ..., 0.   , 1.   , 0.   ],
       ...,
       [0.63 , 0.505, 0.18 , ..., 0.   , 1.   , 0.   ],
       [0.35 , 0.26 , 0.095, ..., 0.   , 1.   , 0.   ],
       [0.52 , 0.405, 0.12 , ..., 1.   , 0.   , 0.   ]])
```

```
y_train
```

```
1595    10.5
2227     8.5
1995     7.5
2843    11.5
3785    10.5
        ...
3443     8.5
3028     8.5
1385    12.5
3813     9.5
```

```
     473     12.5
     Name: age, Length: 2110, dtype: float64
```

```
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 :4.694526
```

```
#Testing the model
```

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

```
X_test
```

```
     array([[0.445, 0.355, 0.095, ..., 0.   , 1.   , 0.   ],
            [0.43 , 0.35 , 0.105, ..., 0.   , 1.   , 0.   ],
            [0.595, 0.475, 0.165, ..., 0.   , 0.   , 1.   ],
            ...,
            [0.445, 0.33 , 0.12 , ..., 0.   , 1.   , 0.   ],
            [0.475, 0.35 , 0.115, ..., 1.   , 0.   , 0.   ],
            [0.42 , 0.325, 0.1  , ..., 0.   , 1.   , 0.   ]])
```

```
y_test
```

```
     3540     9.5
     3253     7.5
     1001    10.5
     399     12.5
     1756    15.5
             ...
     4023     7.5
     1390    10.5
     645     12.5
     562     12.5
     1839    12.5
     Name: age, Length: 1037, dtype: float64
```

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

```
     Mean Squared error of testing set :4.623368
```

```
#Measure the performance using metrices
```

```
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
```

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

```
R2 Score of testing set:0.53
```