ASSIGNMENT 4

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

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

Choose files abalone.csv

- **abalone.csv**(text/csv) - 191962 bytes, last modified: 31/10/2022 - 100% done

```
Saving abalone.csv to abalone (1).csv
```

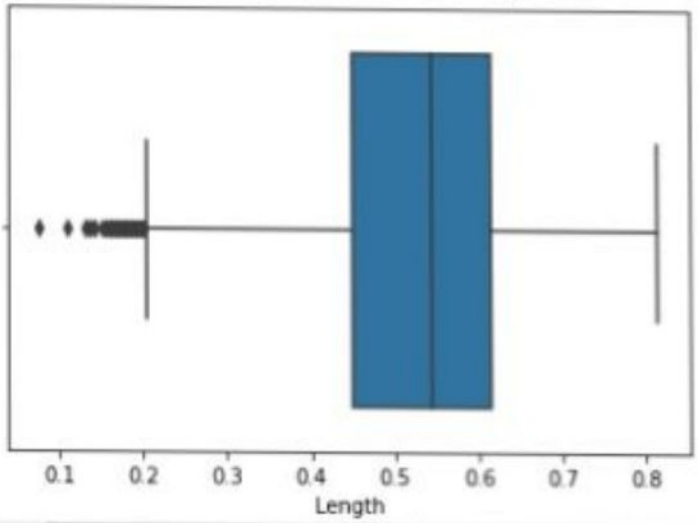
	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0
min	0.075000	0.055000	0.000000	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 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

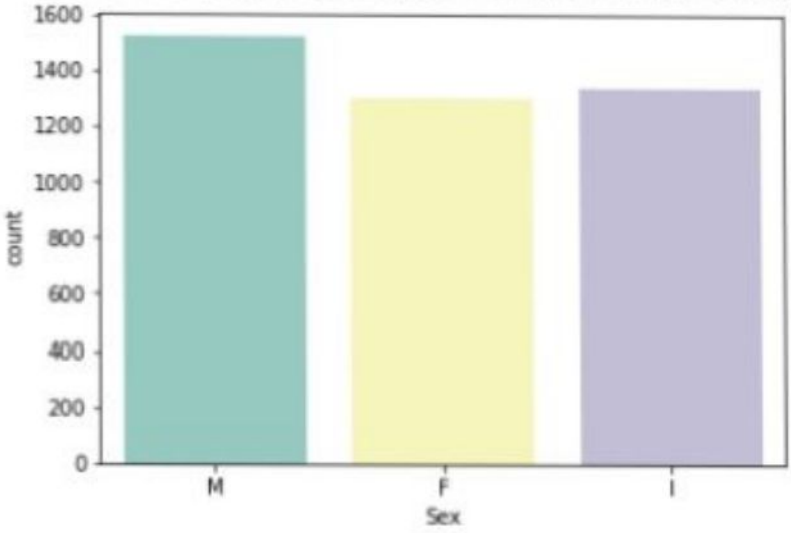
```
sns.boxplot(df.Length)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fc169f7ca50>
```




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

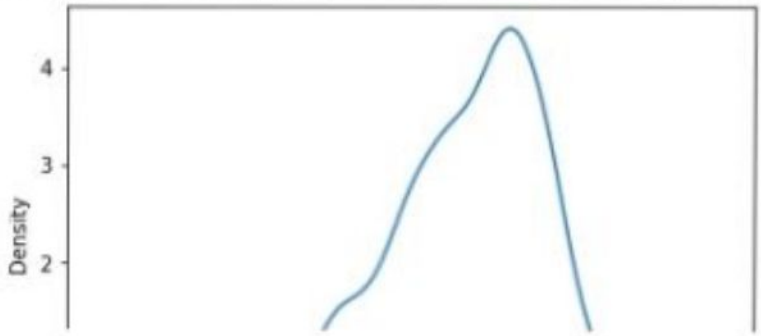
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc169a05750>
```




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

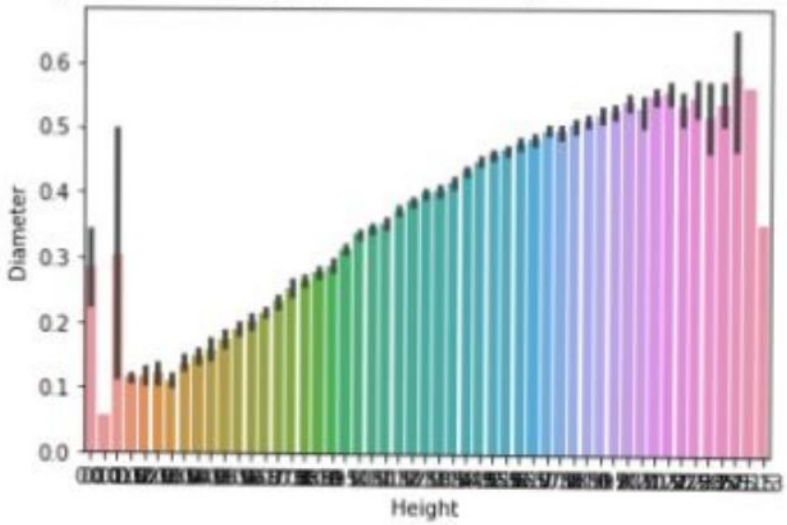
```
a['age']=a['Rings']+1.5
a=a.drop('Rings',axis=1)
sns.kdeplot(a['Diameter'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc169986a90>
```



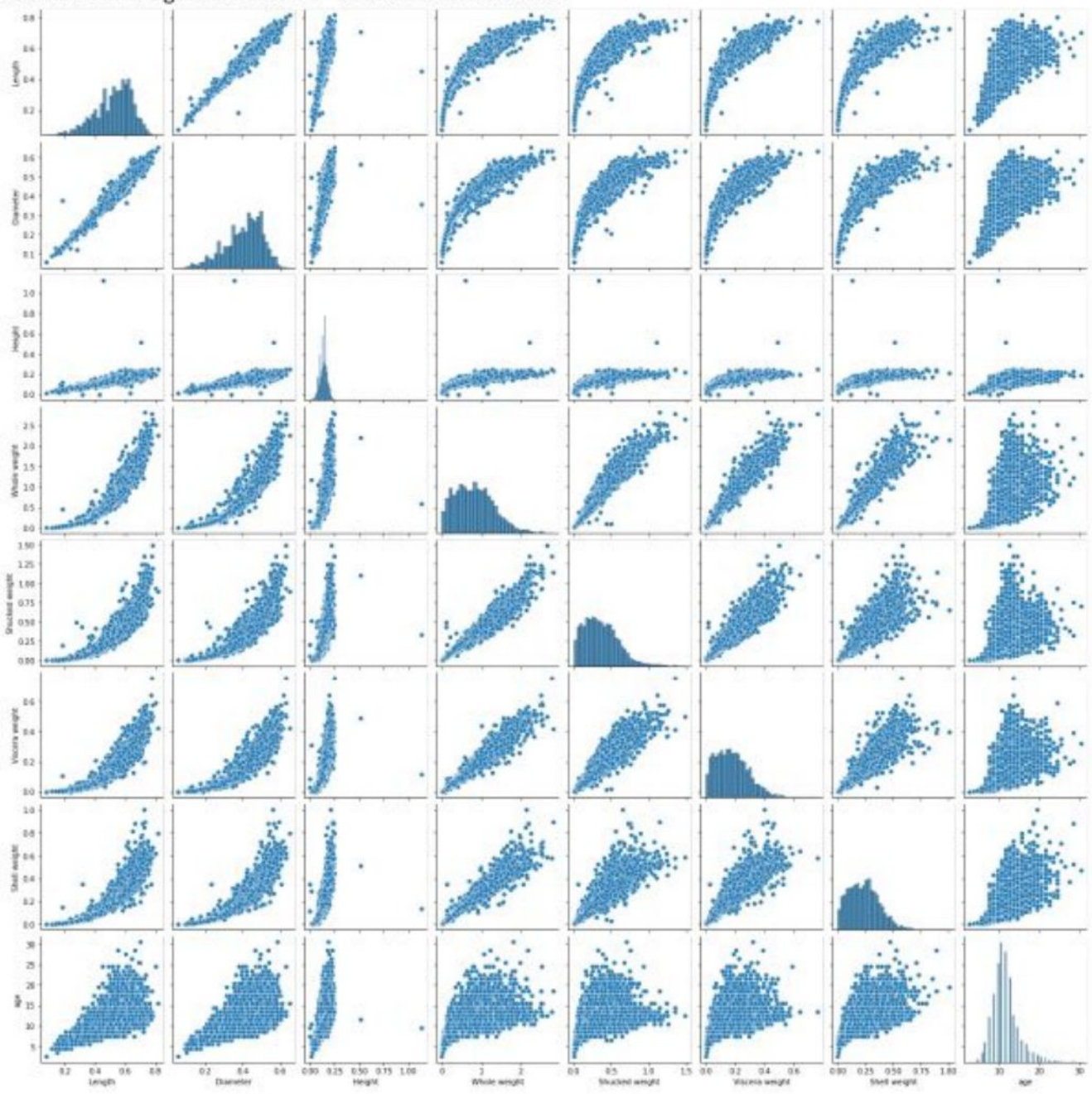

```
sns.barplot(x=df.Height,y=df.Diameter)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc169928fd0>
```




```
sns.pairplot(a)
```

```
<seaborn.axisgrid.PairGrid at 0x7fc1696dbc90>
```



```
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
```

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

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
1257	I	0.430	0.34	0.0	0.428	0.2065	0.0860	0.1150	8
3996	I	0.315	0.23	0.0	0.134	0.0575	0.0285	0.3505	6

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

```
0.425
```

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

```
0.7190979217612694
```

```
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	%
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

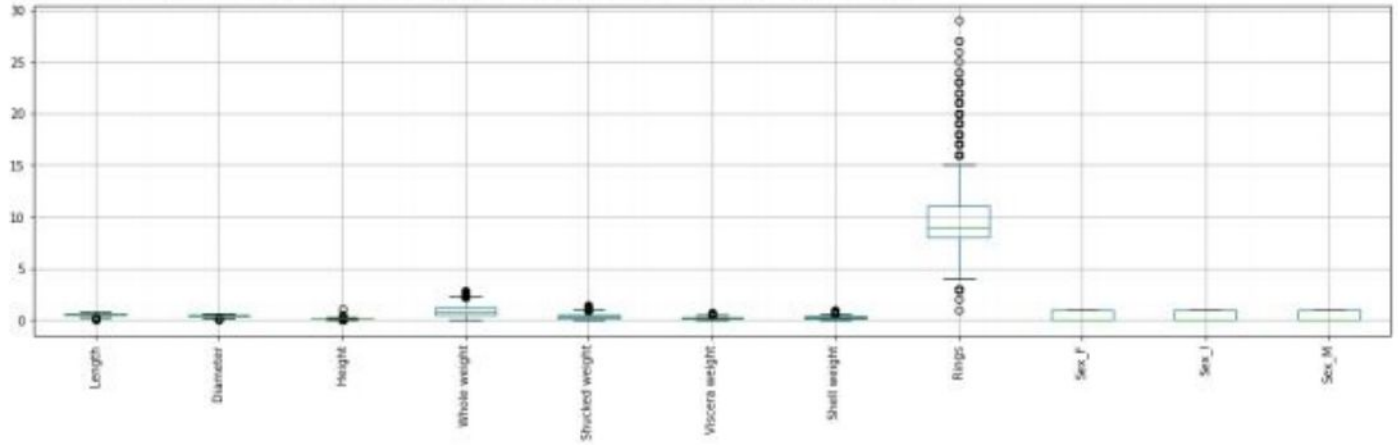
```
q1=df.Rings.quantile(0.25)
q2=df.Rings.quantile(0.75)
iqr=q1-q2
print(iqr)
```

```
-3.0
```

```
df=pd.get_dummies(df)
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc164f09310>
```

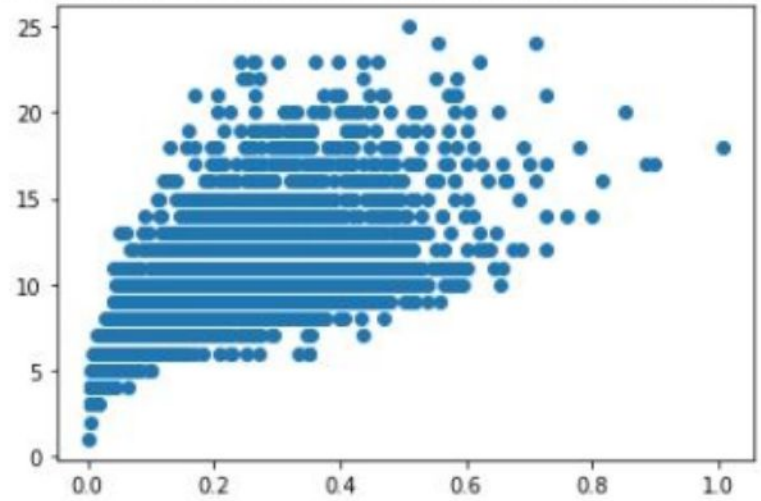



```
df['age']=df['Rings']
df=df.drop('Rings',axis=1)
```

```
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'
plt.scatter(x=df[var],y=df['age'])
```

```
<matplotlib.collections.PathCollection at 0x7fc1634bf0d0>
```



```
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
```

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

```
abalone_numeric.head()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
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: 4150, dtype: float64
```

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

```
 ORIGNAL 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: 4150, 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]]
```

```
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)
```

```
array([[-0.56736455, -0.42395732, -1.05992592, ..., -0.67424712,
        -0.69131775,  1.32156176],
       [-1.44754363, -1.43820927, -1.1801252 , ..., -0.67424712,
        -0.69131775,  1.32156176],
       [ 0.0613348 ,  0.13388126, -0.0983317 , ...,  1.48313573,
        -0.69131775, -0.75668049],
       ...,
       [ 0.64812085,  0.69171983,  1.58445819, ..., -0.67424712,
        -0.69131775,  1.32156176],
       [ 0.8576873 ,  0.79314503,  0.26226613, ...,  1.48313573,
        -0.69131775, -0.75668049],
       [ 1.57021323,  1.50312139,  1.34405963, ..., -0.67424712,
        -0.69131775,  1.32156176]])
```

```
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, 0.425, 0.12 , ..., 0.   , 0.   , 1.   ],
       [0.46 , 0.355, 0.14 , ..., 0.   , 1.   , 0.   ],
       [0.59 , 0.445, 0.13 , ..., 1.   , 0.   , 0.   ],
       ...,
       [0.38 , 0.28 , 0.085, ..., 0.   , 1.   , 0.   ],
       [0.645, 0.49 , 0.19 , ..., 1.   , 0.   , 0.   ],
       [0.465, 0.37 , 0.115, ..., 0.   , 1.   , 0.   ]])
```

```
y_train
```

```
734     18
583     13
766     13
1113     8
2857    11
        ..
1564     7
2886     8
2166     6
3918    18
942      7
Name: age, Length: 3112, dtype: int64
```

```
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: 0.5290674506339392
```

```
from matplotlib.ticker import LinearLocator
lm=LinearRegression()
lm.fit(x_train,y_train)
y_train_pred=lm.predict(x_train)
y_train_pred
```

```
array([12.25  ,  8.375 , 12.5625, ...,  7.25  , 12.6875,  8.0625])
```

```
x_train
```

```
array([[0.525, 0.425, 0.12 , ..., 0.   , 0.   , 1.   ],
       [0.46 , 0.355, 0.14 , ..., 0.   , 1.   , 0.   ],
       [0.59 , 0.445, 0.13 , ..., 1.   , 0.   , 0.   ],
       ...,
       [0.38 , 0.28 , 0.085, ..., 0.   , 1.   , 0.   ],
       [0.645, 0.49 , 0.19 , ..., 1.   , 0.   , 0.   ],
       [0.465, 0.37 , 0.115, ..., 0.   , 1.   , 0.   ]])
```

```
y_train
```

```
734     18
583     13
766     13
1113     8
2857    11
        ..
1564     7
2886     8
2166     6
3918    18
942      7
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:4.773554
```

```
y_train_pred=lm.predict(x_train)
y_test_pred=lm.predict(x_test)
y_test_pred
```

```
array([ 9.375 ,  7.3125, 11.5625, ...,  6.875 , 12.8125,  8.1875])
```

```
x_test
```

```
array([[0.48 , 0.375, 0.105, ..., 1.   , 0.   , 0.   ],
       [0.635, 0.495, 0.015, ..., 1.   , 0.   , 0.   ],
       [0.655, 0.52 , 0.17 , ..., 0.   , 0.   , 1.   ],
       ...,
       [0.34 , 0.26 , 0.08 , ..., 0.   , 1.   , 0.   ],
       [0.5  , 0.4  , 0.165, ..., 0.   , 0.   , 1.   ],
       [0.485, 0.37 , 0.1  , ..., 0.   , 1.   , 0.   ]])
```

```
y_test
```

```
727     12
1174     9
```

```
1963     9
3440     7
2982     9
        ..
4023     6
2921     9
347      6
201     13
3443     7
Name: age, Length: 1038, dtype: int64
```

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

```
Mean Squared error of testing set:4.711923
```

```
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
```