In [1]:

```
from sklearn.datasets import load_boston
```

In [2]:

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

In [3]:

```
sns.set(rc={'figure.figsize':(11.7,8.27)})
```

In [4]:

```
boston=load_boston()
```

```
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the origina
l
    source::

        import pandas as pd
        import numpy as np

        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, heade
r=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2,
:2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.
e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames
housing
```

In [5]:

```
boston
```

Out[5]:

```
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01,
3.9690e+02,
         4.9800e+00],
        [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+
02,
         9.1400e+00],
        [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+
02,
         4.0300e+00],
        ...,
        [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+
02,
         5.6400e+00],
        [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+
02,
         6.4800e+00],
        [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+
02,
         7.8800e+00]]),
 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.
5, 18.9, 15. ,
        18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 1
9.6,
        15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 1
3.2,
        13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 2
4.7,
        21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 1
8.9,
        35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 2
3.5,
        19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 2
0. ,
        20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 2
2.2,
        23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 4
3.8,
        33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 1
9.4,
        21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 2
2. ,
        20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 1
9.6,
        23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 1
3.4,
        15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 1
9.4,
        17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 2
2.7,
        25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 2
9.4,
        23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 5
0. ,
        32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 3
0.3,
```

```
        34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 2
4.4,
        20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 2
3. ,
        26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 2
4.3,
        31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 2
0.1,
        22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 2
9.6,
        42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 3
1. ,
        36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 3
2.4,
        32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 2
2. ,
        20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 2
7.1,
        20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 2
8.2,
        22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 2
3.1,
        21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 2
2.6,
        19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 1
8.7,
        32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 2
4.1,
        18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 2
0.8,
        16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 1
3.8,
        13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,
8.8,
         7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 1
3.1,
        12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 1
1.9,
        27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 1
0.4,
         8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 1
1. ,
         9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 1
2.8,
        10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 1
3.4,
        15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 1
7.7,
        19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 2
3.2,
        29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 2
1.8,
        20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 2
4.5,
        23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 1
1.9]),
 'feature_names': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM',
'AGE', 'DIS', 'RAD',
        'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'),
 'DESCR': ".. _boston_dataset:\n\nBoston house prices dataset\n-------
--------------------\n\n**Data Set Characteristics:**  \n\n    :Number
```

```
of Instances: 506 \n\n    :Number of Attributes: 13 numeric/categorica
l predictive. Median Value (attribute 14) is usually the target.\n\n
:Attribute Information (in order):\n        - CRIM     per capita crim
e rate by town\n        - ZN       proportion of residential land zone
d for lots over 25,000 sq.ft.\n        - INDUS    proportion of non-re
tail business acres per town\n        - CHAS     Charles River dummy v
ariable (= 1 if tract bounds river; 0 otherwise)\n        - NOX      n
itric oxides concentration (parts per 10 million)\n        - RM       
average number of rooms per dwelling\n        - AGE      proportion of
owner-occupied units built prior to 1940\n        - DIS      weighted 
distances to five Boston employment centres\n        - RAD      index 
of accessibility to radial highways\n        - TAX      full-value pro
perty-tax rate per $10,000\n        - PTRATIO  pupil-teacher ratio by 
town\n        - B        1000(Bk - 0.63)^2 where Bk is the proportion 
of black people by town\n        - LSTAT    % lower status of the popu
lation\n        - MEDV     Median value of owner-occupied homes in $10
00's\n\n    :Missing Attribute Values: None\n\n    :Creator: Harrison,
D. and Rubinfeld, D.L.\n\nThis is a copy of UCI ML housing dataset.\nh
ttps://archive.ics.uci.edu/ml/machine-learning-databases/housing/\n\n
\nThis dataset was taken from the StatLib library which is maintained 
at Carnegie Mellon University.\n\nThe Boston house-price data of Harri
son, D. and Rubinfeld, D.L. 'Hedonic\nprices and the demand for clean 
air', J. Environ. Economics & Management,\nvol.5, 81-102, 1978.   Used
in Belsley, Kuh & Welsch, 'Regression diagnostics\n...', Wiley, 1980. 
N.B. Various transformations are used in the table on\npages 244-261 o
f the latter.\n\nThe Boston house-price data has been used in many mac
hine learning papers that address regression\nproblems.   \n     \n.. 
topic:: References\n\n   - Belsley, Kuh & Welsch, 'Regression diagnost
ics: Identifying Influential Data and Sources of Collinearity', Wiley,
1980. 244-261.\n   - Quinlan,R. (1993). Combining Instance-Based and M
odel-Based Learning. In Proceedings on the Tenth International Confere
nce of Machine Learning, 236-243, University of Massachusetts, Amhers
t. Morgan Kaufmann.\n",
 'filename': 'boston_house_prices.csv',
 'data_module': 'sklearn.datasets.data'}
```

In [6]:

```
boston.keys()
```

Out[6]:

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'da
ta_module'])
```

In [7]:

```
boston.DESCR
```

Out[7]:

```
".. _boston_dataset:\n\nBoston house prices dataset\n-----------------
----------\n\n**Data Set Characteristics:**  \n\n    :Number of Instan
ces: 506 \n\n    :Number of Attributes: 13 numeric/categorical predict
ive. Median Value (attribute 14) is usually the target.\n\n    :Attrib
ute Information (in order):\n        - CRIM     per capita crime rate 
by town\n        - ZN       proportion of residential land zoned for l
ots over 25,000 sq.ft.\n        - INDUS    proportion of non-retail bu
siness acres per town\n        - CHAS     Charles River dummy variable
(= 1 if tract bounds river; 0 otherwise)\n        - NOX      nitric ox
ides concentration (parts per 10 million)\n        - RM       average 
number of rooms per dwelling\n        - AGE      proportion of owner-o
ccupied units built prior to 1940\n        - DIS      weighted distanc
es to five Boston employment centres\n        - RAD      index of acce
ssibility to radial highways\n        - TAX      full-value property-t
ax rate per $10,000\n        - PTRATIO  pupil-teacher ratio by town\n 
- B        1000(Bk - 0.63)^2 where Bk is the proportion of black peopl
e by town\n        - LSTAT    % lower status of the population\n      
- MEDV     Median value of owner-occupied homes in $1000's\n\n    :Mis
sing Attribute Values: None\n\n    :Creator: Harrison, D. and Rubinfel
d, D.L.\n\nThis is a copy of UCI ML housing dataset.\nhttps://archive.
ics.uci.edu/ml/machine-learning-databases/housing/\n\n\nThis dataset w
as taken from the StatLib library which is maintained at Carnegie Mell
on University.\n\nThe Boston house-price data of Harrison, D. and Rubi
nfeld, D.L. 'Hedonic\nprices and the demand for clean air', J. Enviro
n. Economics & Management,\nvol.5, 81-102, 1978.   Used in Belsley, Ku
h & Welsch, 'Regression diagnostics\n...', Wiley, 1980.   N.B. Various
transformations are used in the table on\npages 244-261 of the latte
r.\n\nThe Boston house-price data has been used in many machine learni
ng papers that address regression\nproblems.   \n     \n.. topic:: Ref
erences\n\n   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identi
fying Influential Data and Sources of Collinearity', Wiley, 1980. 244-
261.\n   - Quinlan,R. (1993). Combining Instance-Based and Model-Based
Learning. In Proceedings on the Tenth International Conference of Mach
ine Learning, 236-243, University of Massachusetts, Amherst. Morgan Ka
ufmann.\n"
```

In [8]:

```
boston.feature_names
```

Out[8]:

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RA
D',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

In [9]:

```
len(boston.feature_names)
```

Out[9]:

```
13
```

In [10]:

```
boston_data=pd.DataFrame(boston.data,columns=boston.feature_names)
```

In [11]:

```
boston_data
```

Out[11]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90

506 rows × 13 columns

In [12]:

```
boston_data['MEDV']=boston.target
```

In [13]:

```
boston_data
```

Out[13]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90

506 rows × 14 columns

In [14]:

```
boston_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(14)
```

In [15]:

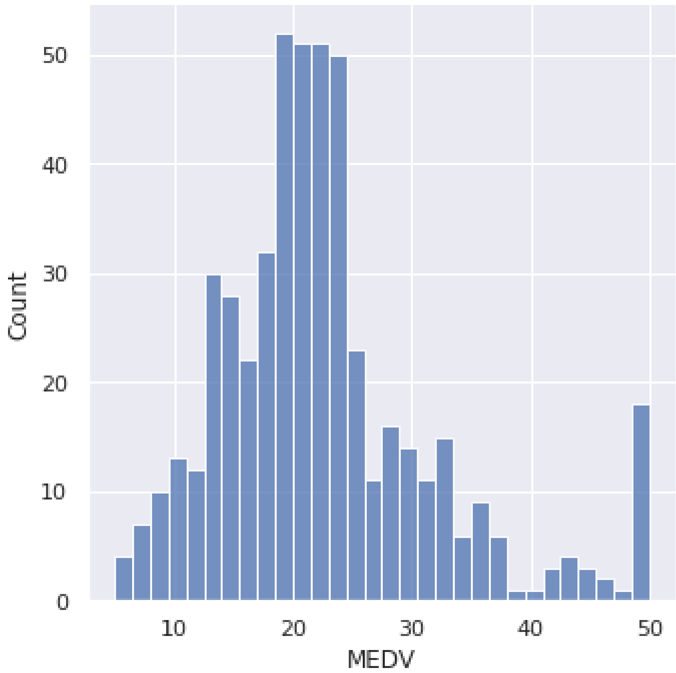
```
boston_data.isnull().sum()
```

Out[15]:

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [16]:

```
sns.displot(boston_data['MEDV'],bins=30)
plt.show()
```

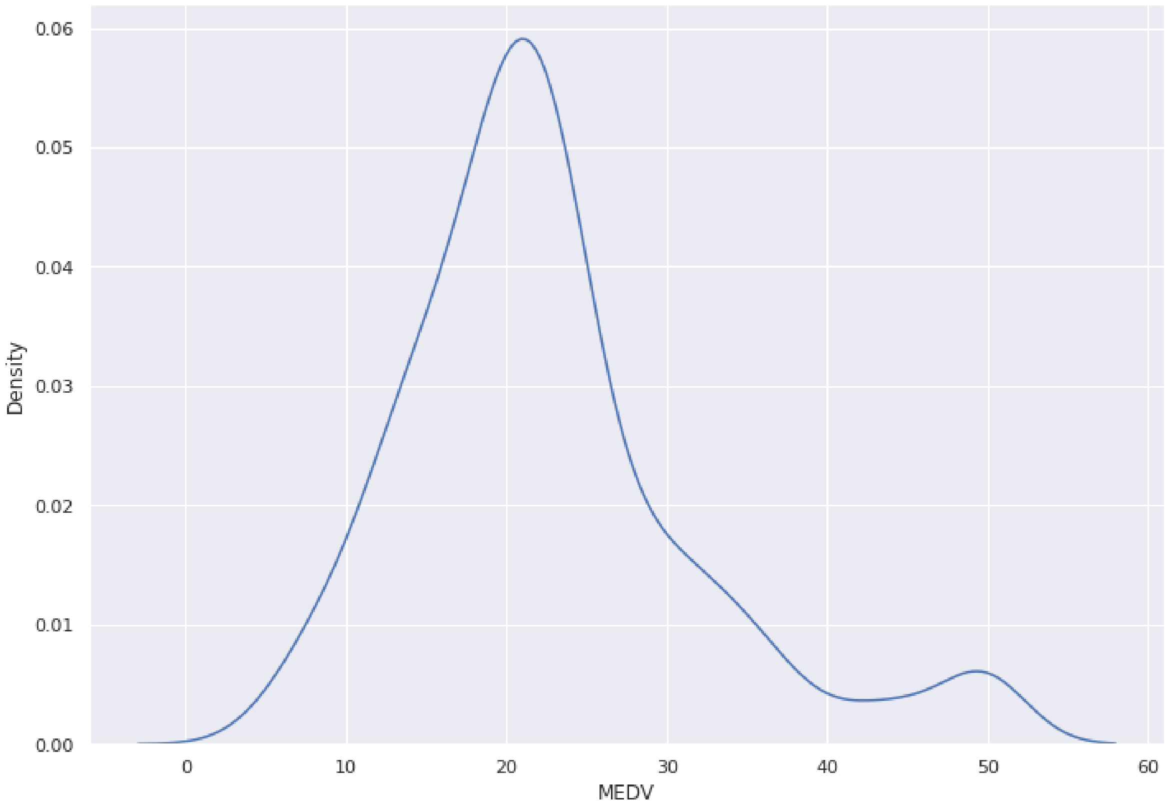


In [17]:

```
sns.kdeplot(data=boston_data,x='MEDV')
```

Out[17]:

```
<AxesSubplot:xlabel='MEDV', ylabel='Density'>
```



In [18]:

```
corr_matrix=boston_data.corr().round(2)
```

In [19]:

```
sns.heatmap(data=corr_matrix,annot=True)
```

Out[19]:

```
<AxesSubplot:>
```

RM ----> average number of rooms per dwelling

In [20]:

```
X = pd.DataFrame(np.c_[boston_data['LSTAT'], boston_data['RM']], columns = ['LSTAT'
```

In [21]:

```
Y = boston_data['MEDV']
```

In [22]:

```
X
```

Out[22]:

	LSTAT	RM
0	4.98	6.575
1	9.14	6.421
2	4.03	7.185
3	2.94	6.998
4	5.33	7.147
...	...	...
501	9.67	6.593
502	9.08	6.120
503	5.64	6.976
504	6.48	6.794
505	7.88	6.030

506 rows × 2 columns

In [23]:

```
Y
```

Out[23]:

```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

In [24]:

```
from sklearn.model_selection import train_test_split
```

In [25]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

In [26]:

```
X
```

Out[26]:

	LSTAT	RM
0	4.98	6.575
1	9.14	6.421
2	4.03	7.185
3	2.94	6.998
4	5.33	7.147
...	...	...
501	9.67	6.593
502	9.08	6.120
503	5.64	6.976
504	6.48	6.794
505	7.88	6.030

506 rows × 2 columns

In [27]:

```
Y_train.shape
```

Out[27]:

(404,)

In [28]:

```
X_train.shape
```

Out[28]:

(404, 2)

In [ ]:

```

```

In [36]:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, Y_train)
y_pred = lr.predict(X_test)
```

In [37]:

```
print(y_pred)
```

```
[18.4785072  28.9993276  19.34419211 14.52059333 20.96645744 18.908852
95
 25.97663035 24.04164894 30.54128281 21.76725649 18.34854405 30.139584
41
 30.89087819 24.11999908  5.56488904 22.30616847 34.38373135 20.205913
35
 20.7528009  23.91070427 30.64994705 36.62423805 21.19044152 28.193241
15
 13.49957862 20.65741102 16.73935708 18.30961587 26.43136276 24.453824
15
 22.77519663  9.18330071 31.44727016 22.6444472  19.54119711 22.772845
41
 13.64477005 26.21423255  9.83900633 15.87181018 32.24709149 29.095752
55
 22.36400141 19.33493648 20.17314927 22.0062284  21.21140504 20.356540
71
 13.28519316 19.66822332 36.69847677 23.79185493 22.2811433  32.101260
79
 20.82919789 35.37326129 18.97819246 13.49468321 31.59819455 21.158808
23
 27.16580899 27.79673292 19.93936752 26.71558608 22.90986148 19.377198
15
 17.18057012 31.26676666 20.99735726 24.50720268 15.03628934 22.824948
42
 22.42922419 30.61978147 18.63085334 12.48115938 32.90592549 16.325320
63
 26.27490444 18.5363409  19.63977121 18.22166635 21.23642489 21.705946
85
 21.94530924 22.75888885 22.69052781 20.32955783 24.10398529 26.029373
4
 26.25938369 25.50417297 28.99178773 15.20210432 23.17263281 25.542663
95
  5.53056152 26.84638223 23.90668857 29.34030284 21.81691027 26.396595
76]
```

In [39]:

```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(Y_test, y_pred))
print("Root Mead squared Error is:")
print(rmse)
```

```
Root Mead squared Error is:
5.091246859885894
```

In [41]:

```
lr.score(X_train, Y_train)
```

Out[41]:

```
0.6262834852468848
```

In [43]:

```
lr.score(X_test, Y_test)
```

Out[43]:

0.682624165203414

In [ ]:

```
```