In [4]:
```
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sns

%matplotlib inline
```

In [6]:
```
from sklearn.datasets import load_boston
```

In [9]:
```
boston=load_boston()
```

In [10]:
```
boston
```

Out[10]:
```
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+0
1, 3.9690e+02,
         4.9800e+00],
        [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.969
0e+02,
         9.1400e+00],
        [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.928
3e+02,
         4.0300e+00],
        ...,
        [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.969
0e+02,
         5.6400e+00],
        [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.934
5e+02,
         6.4800e+00],
        [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.969
0e+02,
         7.8800e+00]]),
 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 1
6.5, 18.9, 15. ,
        18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6,
19.6,
        15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5,
13.2,
        13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3,
24.7,
        21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4,
18.9,
        35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. ,
23.5,
        19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4,
20. ,
        20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5,
22.2,
        23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7,
43.8,
        33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8,
19.4,
        21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3,
22. ,
        20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2,
19.6,
        23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4,
13.4,
        15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3,
19.4,
        17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. ,
22.7,
        25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6,
29.4,
        23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6,
50. ,
        32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3,
30.3,
        34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5,
24.4,
```

```
        20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5,
23. ,
        26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5,
24.3,
        31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. ,
20.1,
        22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8,
29.6,
        42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8,
31. ,
        36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2,
32.4,
        32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2,
22. ,
        20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6,
27.1,
        20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4,
28.2,
        22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8,
23.1,
        21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3,
22.6,
        19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. ,
18.7,
        32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9,
24.1,
        18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9,
20.8,
        16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. ,
13.8,
        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,
13.1,
        12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. ,
11.9,
        27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5,
10.4,
         8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9,
11. ,
         9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4,
12.8,
        10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. ,
13.4,
        15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4,
17.7,
        19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6,
23.2,
        29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. ,
21.8,
        20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8,
24.5,
        23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. ,
11.9]),
 '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
```

```
/categorical predictive. Median Value (attribute 14) is usually the
target.\n\n    :Attribute Information (in order):\n        - CRIM
per capita crime rate by town\n        - ZN       proportion of res
idential land zoned for lots over 25,000 sq.ft.\n        - INDUS
proportion of non-retail business acres per town\n        - CHAS
Charles River dummy variable (= 1 if tract bounds river; 0 otherwis
e)\n        - NOX      nitric oxides concentration (parts per 10 mi
llion)\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 property-tax rate per $10,000\n        - PTRATIO  pupil-
teacher ratio by town\n        - B        1000(Bk - 0.63)^2 where B
k is the proportion of black people by town\n        - LSTAT    % l
ower status of the population\n        - MEDV     Median value of o
wner-occupied homes in $1000's\n\n    :Missing Attribute Values: No
ne\n\n    :Creator: Harrison, D. and Rubinfeld, D.L.\n\nThis is a c
opy of UCI ML housing dataset.\nhttps://archive.ics.uci.edu/ml/mach
ine-learning-databases/housing/\n\n\nThis dataset was taken from th
e StatLib library which is maintained at Carnegie Mellon Universit
y.\n\nThe Boston house-price data of Harrison, D. and Rubinfeld, D.
L. 'Hedonic\nprices and the demand for clean air', J. Environ. Econ
omics & 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 of the latt
er.\n\nThe Boston house-price data has been used in many machine le
arning papers that address regression\nproblems.   \n     \n.. topi
c:: References\n\n   - Belsley, Kuh & Welsch, 'Regression diagnosti
cs: Identifying Influential Data and Sources of Collinearity', Wile
y, 1980. 244-261.\n   - Quinlan,R. (1993). Combining Instance-Based
and Model-Based Learning. In Proceedings on the Tenth International
Conference of Machine Learning, 236-243, University of Massachusett
s, Amherst. Morgan Kaufmann.\n",
 'filename': 'boston_house_prices.csv',
```

In [11]:
```
boston.keys()
```

Out[11]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename',
'data_module'])
```

In [12]:
```
boston.feature_names
```

Out[12]:
```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'R
AD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

In [13]:
```
boston_data=pd.DataFrame(boston.data,columns=boston.feature_names)
```

In [15]:
```
boston_data
```

Out[15]:

	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 [ ]:

In [20]:
```
boston_data['MEDV']=boston.target
```

In [16]:
```
boston_data.isnull().sum()
```

Out[16]:
```
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
dtype: int64
```

In [21]:

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

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

Out[21]: <AxesSubplot: >

In [18]:

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

In [22]:

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

In [24]:

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

In [25]:

```
from sklearn.model_selection import train_test_split

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

In [26]:

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

In [27]:

```
print(y_pred)
```

```
[17.0158238  20.08439816 21.11751691 18.58939292 31.12819518 12.045
29737
 12.9431592  25.50276159 21.74561491 23.27601559 23.93062596 21.061
18666
 19.58867816 20.46234662 31.48150684 20.72959078 24.03266333 18.735
76774
 32.98494955 24.08521096 10.57428835 25.36644863 13.19678615 37.185
51626
 12.59741851 24.36949922 19.89245798 18.23375385 17.14965076 22.952
07839
 19.67056657 30.0313232  30.17038529 37.93257436  9.30284941 20.486
62281
  5.34999675 24.43137336 31.00014291 12.62200969 30.8651961  26.361
70141
 20.6598945  23.81037058  6.08489992 17.95901801 31.9584166  24.659
03946
 21.05852524 37.60630974 18.77956209 19.4868534  28.66262663 27.492
66514
 26.22125145 18.70012112 24.17112829 29.85598722 13.02713362 28.103
20961
 18.03122555 25.20240732 21.9972604  23.58681623 18.66979028 25.919
59349
 17.77080018 18.66268664 21.79121748 21.62667517 35.01946209 22.708
267
 26.24544482 21.80678248 19.19747175 18.89110492 17.1180871  26.390
7959
 25.2323417  28.15007514 22.01458447 27.7842493  23.07381288 23.412
32014
 24.76563554 23.0739883  18.67273166 28.03823062 20.44958968 35.747
64613
 25.79564455 22.88953979  2.51093527 32.96101931 28.09585126 18.297
96382
 22.89488788 19.53149959 24.20072379 27.45040041 35.34351225 22.412
84753]
```

In [28]:

```
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:
4.753323260570139
```

In [ ]: