Boston Dataset - sklearn

The sklearn Boston dataset is used wisely in regression and is famous dataset from the 1970's. There are 506 instances and 14 attributes

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html

source:https://towardsdatascience.com/machine-learning-project-predicting-boston-house-prices-with-regression-b4e47493633d

In [109]:
```
# print all the outputs in a cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [158]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Loading the Boston dataset from Scikit-learn

In [111]:
```
#  import the Boston dataset and store it in a variable called boston
from sklearn.datasets import load_boston
boston = load_boston()
```

explore the data

In [112]:
```
type(boston)
```

Out[112]:
```
sklearn.utils.Bunch
```

In [113]:
```
boston # boston is a dictionary-like object
```

Out[113]:
```
{'DESCR': "Boston House Prices dataset\n===========================\n\nNote
s\n------\nData Set Characteristics:  \n\n    :Number of Instances: 506 \n
\n    :Number of Attributes: 13 numeric/categorical predictive\n    \n    :
Median Value (attribute 14) is usually the target\n\n    :Attribute Informa
tion (in order):\n        - CRIM     per capita crime rate by town\n
- ZN       proportion of residential land zoned for lots over 25,000 sq.f
t.\n        - INDUS    proportion of non-retail business acres per town\n
- CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 other
wise)\n        - NOX      nitric oxides concentration (parts per 10 millio
n)\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      in
dex of accessibility to radial highways\n        - TAX      full-value prop
erty-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 blacks by town\n
- LSTAT    % lower status of the population\n        - MEDV     Median valu
e of owner-occupied homes in $1000'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.\nhttp://archive.ics.uci.edu/ml/datasets/Housing\n\n\nTh
is dataset was taken from the StatLib library which is maintained at Carneg
ie Mellon University.\n\nThe Boston house-price data of Harrison, D. and Ru
binfeld, D.L. 'Hedonic\nprices and the demand for clean air', J. Environ. E
conomics & Management,\nvol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsc
h, 'Regression diagnostics\n...', Wiley, 1980.   N.B. Various transformatio
ns are used in the table on\npages 244-261 of the latter.\n\nThe Boston hou
se-price data has been used in many machine learning papers that address re
gression\nproblems.   \n     \n**References**\n\n   - Belsley, Kuh & Welsc
h, 'Regression diagnostics: Identifying Influential Data and Sources of Col
linearity', Wiley, 1980. 244-261.\n   - Quinlan,R. (1993). Combining Instan
ce-Based and Model-Based Learning. In Proceedings on the Tenth Internationa
l Conference of Machine Learning, 236-243, University of Massachusetts, Amh
erst. Morgan Kaufmann.\n   - many more! (see http://archive.ics.uci.edu/ml/
datasets/Housing)\n",
 'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.969
0e+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]]),
 'feature_names': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
'DIS', 'RAD',
        'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'),
 '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, 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])}
```

explore the attributes:

- 'data': the data to learn
- 'feature_names': the meaning of the features
- 'target': the regression targets
- 'DESCR': the full description of the dataset
- 'filename':the physical location of boston csv dataset (added in version 0.20).

In [114]:
```
print(boston.keys())
```

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

In [115]:
```
type(boston["data"])
```

Out[115]:
```
numpy.ndarray
```

In [116]:
```
boston["data"].shape #  150 observations, 4 columns
```

Out[116]:
```
(506, 13)
```

In [117]:
```
# print the names of the features
print(boston["feature_names"])
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

In [118]:
```
print (boston.target)
```

```
[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 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]
```

In [119]:
```
# The DESCR method will provide the dataset characteristics for the Boston dat
aset.
print(boston["DESCR"])
```

```
Boston House Prices dataset
===========================

Notes
------
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,00
0 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river;
0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks b
y town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
http://archive.ics.uci.edu/ml/datasets/Housing


This dataset was taken from the StatLib library which is maintained at Carn
egie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnosti
cs
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers t
hat address regression
problems.

**References**

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influentia
l Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning.
In Proceedings on the Tenth International Conference of Machine Learning, 2
36-243, University of Massachusetts, Amherst. Morgan Kaufmann.
   - many more! (see http://archive.ics.uci.edu/ml/datasets/Housing)
```

Pandas Dataframe Conversion

In [120]:
```
df = pd.DataFrame(data = boston["data"], columns=boston["feature_names"])
df.head()
```

Out[120]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LST/
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	4.
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	9.
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	4.
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	2.
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	5.

In [121]:
```
df["target"] = boston["target"] # add the feature "target" as a new column
df.head()
```

Out[121]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LST/
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	4.
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	9.
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	4.
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	2.
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	5.

Export the dataframe to csv

In [122]:
```
# df.to_csv('boston.csv')
```

Explore the Correlation of Features

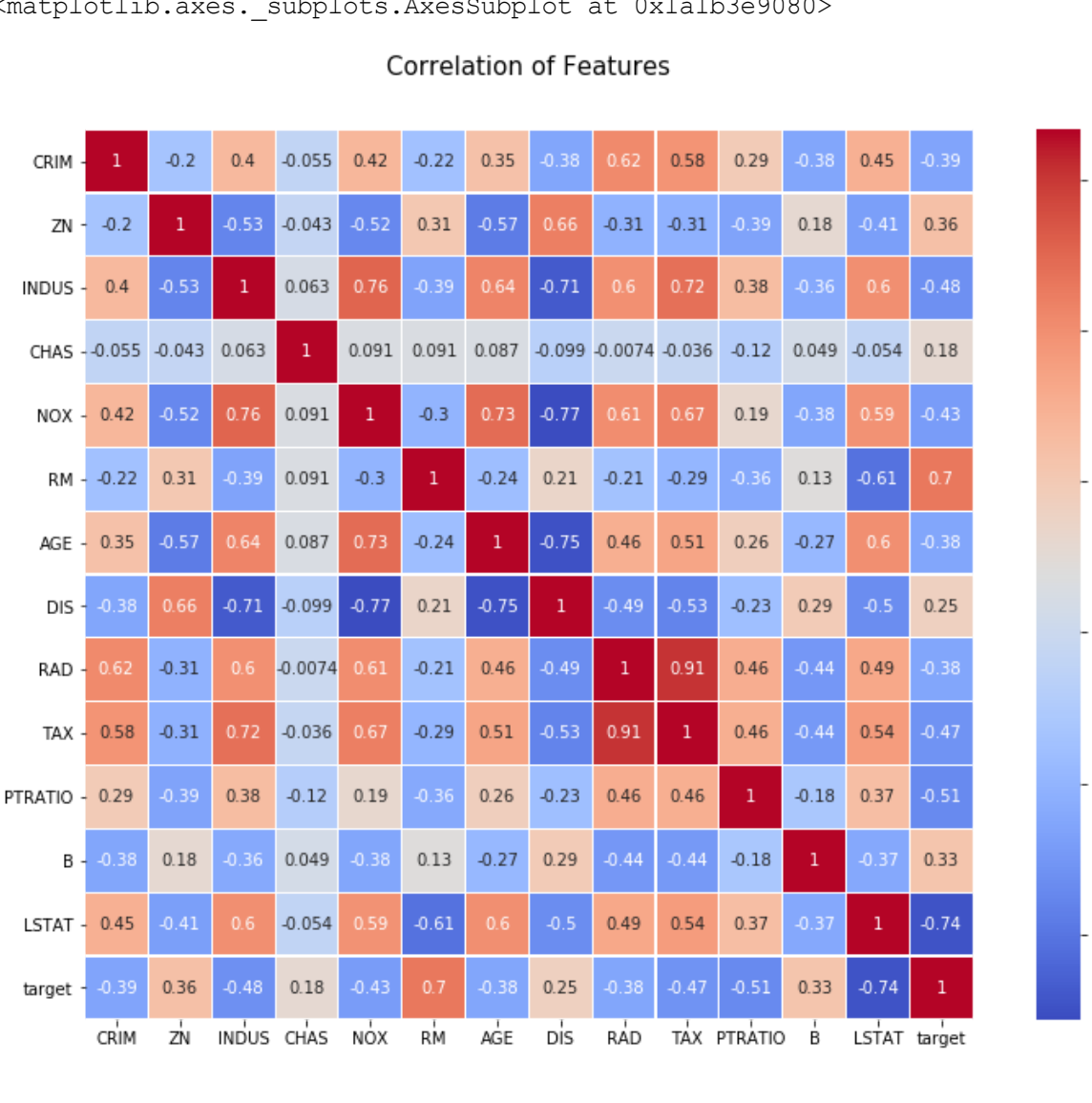
There are different color provided:
https://matplotlib.org/2.0.2/examples/color/colormaps_reference.html

In [123]:
```
plt.figure(figsize=(14,10))
plt.title('Correlation of Features', y=1.05, size=15)
sns.heatmap(df.astype(float).corr(),cmap = "coolwarm"
           ,linewidths=0.1,square=True, linecolor='white', annot=True)
```

Out[123]:
```
<matplotlib.figure.Figure at 0x1a1b3e9978>
```

Out[123]:
```
Text(0.5,1.05,'Correlation of Features')
```

Out[123]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1b3e9080>
```



From above, we can found that there is a high positive correlation between **RM**and **target**,
and there is a high negative correlation between **LSTAT**and **target**

scatter plot

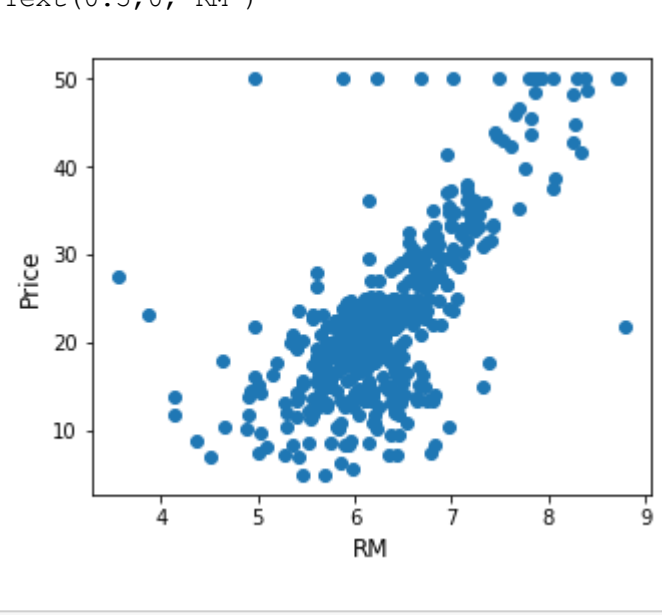
In [137]:
```
plt.figure(figsize=(5, 4))
plt.scatter(df['RM'], boston.target)
plt.ylabel('Price', size=12)
plt.xlabel('RM', size=12)
```

Out[137]:
```
<matplotlib.figure.Figure at 0x10454f780>
```

Out[137]:
```
<matplotlib.collections.PathCollection at 0x1a1a776c88>
```

Out[137]:
```
Text(0,0.5,'Price')
```

Out[137]:
```
Text(0.5,0,'RM')
```



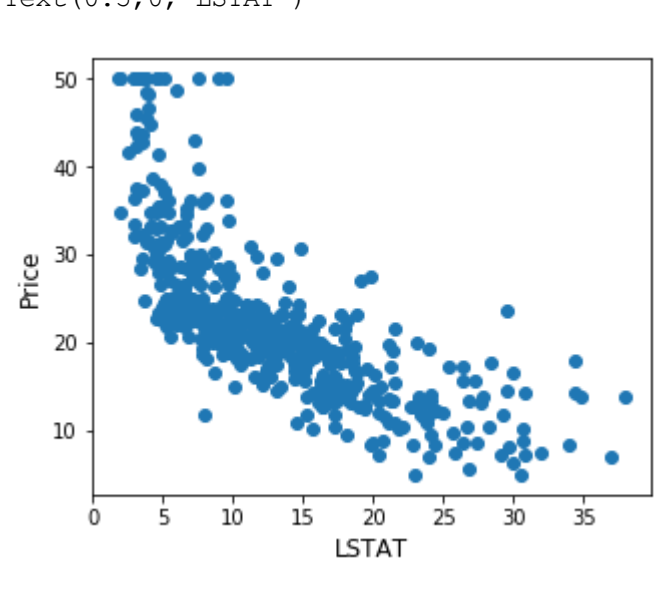
In [138]:
```
plt.figure(figsize=(5, 4))
plt.scatter(df['LSTAT'], boston.target)
plt.ylabel('Price', size=12)
plt.xlabel('LSTAT', size=12)
```

Out[138]:
```
<matplotlib.figure.Figure at 0x1045f4550>
```

Out[138]:
```
<matplotlib.collections.PathCollection at 0x1a1ba84208>
```

Out[138]:
```
Text(0,0.5,'Price')
```

Out[138]:
```
Text(0.5,0,'LSTAT')
```



Split the datasets to training & testing

In [125]:
```
from sklearn.model_selection import train_test_split
```

In [31]:
```
# The ratio of training & testing is 9:1
train_test_split(boston["data"],boston["target"],test_size=0.1)
# Return 90% training data, 10% testing data, 90% label, 10% label
```

Out[31]:
```
[array([[4.87141e+00, 0.00000e+00, 1.81000e+01, ..., 2.02000e+01,
         3.96210e+02, 1.86800e+01],
        [1.81590e-01, 0.00000e+00, 7.38000e+00, ..., 1.96000e+01,
         3.96900e+02, 6.87000e+00],
        [2.49800e-01, 0.00000e+00, 2.18900e+01, ..., 2.12000e+01,
         3.92040e+02, 2.13200e+01],
        ...,
        [4.37900e-02, 8.00000e+01, 3.37000e+00, ..., 1.61000e+01,
         3.96900e+02, 1.02400e+01],
        [3.30600e-02, 0.00000e+00, 5.19000e+00, ..., 2.02000e+01,
         3.96140e+02, 8.51000e+00],
        [1.09600e-02, 5.50000e+01, 2.25000e+00, ..., 1.53000e+01,
         3.94720e+02, 8.23000e+00]]),
 array([[2.07460e-01, 0.00000e+00, 2.77400e+01, 0.00000e+00, 6.09000e-01,
         5.09300e+00, 9.80000e+01, 1.82260e+00, 4.00000e+00, 7.11000e+02,
         2.01000e+01, 3.18430e+02, 2.96800e+01],
        [3.58400e-02, 8.00000e+01, 3.37000e+00, 0.00000e+00, 3.98000e-01,
         6.29000e+00, 1.78000e+01, 6.61150e+00, 4.00000e+00, 3.37000e+02,
         1.61000e+01, 3.96900e+02, 4.67000e+00],
        [1.13081e+00, 0.00000e+00, 8.14000e+00, 0.00000e+00, 5.38000e-01,
         5.71300e+00, 9.41000e+01, 4.23300e+00, 4.00000e+00, 3.07000e+02,
         2.10000e+01, 3.60170e+02, 2.26000e+01],
        [1.77800e-02, 9.50000e+01, 1.47000e+00, 0.00000e+00, 4.03000e-01,
         7.13500e+00, 1.39000e+01, 7.65340e+00, 3.00000e+00, 4.02000e+02,
         1.70000e+01, 3.84300e+02, 4.45000e+00],
        [1.73310e-01, 0.00000e+00, 9.69000e+00, 0.00000e+00, 5.85000e-01,
         5.70700e+00, 5.40000e+01, 2.38170e+00, 6.00000e+00, 3.91000e+02,
         1.92000e+01, 3.96900e+02, 1.20100e+01],
        [2.39120e-01, 0.00000e+00, 9.69000e+00, 0.00000e+00, 5.85000e-01,
         6.01900e+00, 6.53000e+01, 2.40910e+00, 6.00000e+00, 3.91000e+02,
         1.92000e+01, 3.96900e+02, 1.29200e+01],
        [3.56868e+00, 0.00000e+00, 1.81000e+01, 0.00000e+00, 5.80000e-01,
         6.43700e+00, 7.50000e+01, 2.89650e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.93370e+02, 1.43600e+01],
        [2.05500e-02, 8.50000e+01, 7.40000e-01, 0.00000e+00, 4.10000e-01,
         6.38300e+00, 3.57000e+01, 9.18760e+00, 2.00000e+00, 3.13000e+02,
         1.73000e+01, 3.96900e+02, 5.77000e+00],
        [8.24809e+00, 0.00000e+00, 1.81000e+01, 0.00000e+00, 7.13000e-01,
         7.39300e+00, 9.93000e+01, 2.45270e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.75870e+02, 1.67400e+01],
        [2.33099e+00, 0.00000e+00, 1.95800e+01, 0.00000e+00, 8.71000e-01,
         5.18600e+00, 9.38000e+01, 1.52960e+00, 5.00000e+00, 4.03000e+02,
         1.47000e+01, 3.56990e+02, 2.83200e+01],
        [1.51902e+00, 0.00000e+00, 1.95800e+01, 1.00000e+00, 6.05000e-01,
         8.37500e+00, 9.39000e+01, 2.16200e+00, 5.00000e+00, 4.03000e+02,
         1.47000e+01, 3.88450e+02, 3.32000e+00],
        [8.70700e-02, 0.00000e+00, 1.28300e+01, 0.00000e+00, 4.37000e-01,
         6.14000e+00, 4.58000e+01, 4.09050e+00, 5.00000e+00, 3.98000e+02,
         1.87000e+01, 3.86960e+02, 1.02700e+01],
        [5.82401e+00, 0.00000e+00, 1.81000e+01, 0.00000e+00, 5.32000e-01,
         6.24200e+00, 6.47000e+01, 3.42420e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.96900e+02, 1.07400e+01],
        [3.31470e-01, 0.00000e+00, 6.20000e+00, 0.00000e+00, 5.07000e-01,
         8.24700e+00, 7.04000e+01, 3.65190e+00, 8.00000e+00, 3.07000e+02,
         1.74000e+01, 3.78950e+02, 3.95000e+00],
        [1.40507e+01, 0.00000e+00, 1.81000e+01, 0.00000e+00, 5.97000e-01,
         6.65700e+00, 1.00000e+02, 1.52750e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.50500e+01, 2.12200e+01],
        [9.18702e+00, 0.00000e+00, 1.81000e+01, 0.00000e+00, 7.00000e-01,
         5.53600e+00, 1.00000e+02, 1.58040e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.96900e+02, 2.36000e+01],
        [1.00245e+00, 0.00000e+00, 8.14000e+00, 0.00000e+00, 5.38000e-01,
         6.67400e+00, 8.73000e+01, 4.23900e+00, 4.00000e+00, 3.07000e+02,
         2.10000e+01, 3.80230e+02, 1.19800e+01],
        [6.71800e-01, 0.00000e+00, 1.81000e+01, 0.00000e+00, 7.40000e-01,
         6.45900e+00, 9.48000e+01, 1.98790e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 4.30600e+01, 2.39800e+01],
        [5.18800e-02, 0.00000e+00, 4.49000e+00, 0.00000e+00, 4.49000e-01,
         6.01500e+00, 4.51000e+01, 4.42720e+00, 3.00000e+00, 2.47000e+02,
         1.85000e+01, 3.95990e+02, 1.28600e+01],
        [6.39312e+00, 0.00000e+00, 1.81000e+01, 0.00000e+00, 5.84000e-01,
         6.16200e+00, 9.74000e+01, 2.20600e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.02760e+02, 2.41000e+01],
        [4.64689e+00, 0.00000e+00, 1.81000e+01, 0.00000e+00, 6.14000e-01,
         6.98000e+00, 6.76000e+01, 2.53290e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.74680e+02, 1.16600e+01],
        [8.40540e-01, 0.00000e+00, 8.14000e+00, 0.00000e+00, 5.38000e-01,
         5.59900e+00, 8.57000e+01, 4.45460e+00, 4.00000e+00, 3.07000e+02,
         2.10000e+01, 3.03420e+02, 1.65100e+01],
        [7.15100e-02, 0.00000e+00, 4.49000e+00, 0.00000e+00, 4.49000e-01,
         6.12100e+00, 5.68000e+01, 3.74760e+00, 3.00000e+00, 2.47000e+02,
         1.85000e+01, 3.95150e+02, 8.44000e+00],
        [9.91655e+00, 0.00000e+00, 1.81000e+01, 0.00000e+00, 6.93000e-01,
         5.85200e+00, 7.78000e+01, 1.50040e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.38160e+02, 2.99700e+01],
        [9.25200e-02, 3.00000e+01, 4.93000e+00, 0.00000e+00, 4.28000e-01,
         6.60600e+00, 4.22000e+01, 6.18990e+00, 6.00000e+00, 3.00000e+02,
         1.66000e+01, 3.83780e+02, 7.37000e+00],
        [1.40300e-01, 2.20000e+01, 5.86000e+00, 0.00000e+00, 4.31000e-01,
         6.48700e+00, 1.30000e+01, 7.39670e+00, 7.00000e+00, 3.30000e+02,
         1.91000e+01, 3.96280e+02, 5.90000e+00],
        [1.87000e-02, 8.50000e+01, 4.15000e+00, 0.00000e+00, 4.29000e-01,
         6.51600e+00, 2.77000e+01, 8.53530e+00, 4.00000e+00, 3.51000e+02,
         1.79000e+01, 3.92430e+02, 6.36000e+00],
        [4.15292e+01, 0.00000e+00, 1.81000e+01, 0.00000e+00, 6.93000e-01,
         5.53100e+00, 8.54000e+01, 1.60740e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.29460e+02, 2.73800e+01],
        [3.65900e-02, 2.50000e+01, 4.86000e+00, 0.00000e+00, 4.26000e-01,
         6.30200e+00, 3.22000e+01, 5.40070e+00, 4.00000e+00, 2.81000e+02,
         1.90000e+01, 3.96900e+02, 6.72000e+00],
        [9.88430e-01, 0.00000e+00, 8.14000e+00, 0.00000e+00, 5.38000e-01,
         5.81300e+00, 1.00000e+02, 4.09520e+00, 4.00000e+00, 3.07000e+02,
         2.10000e+01, 3.94540e+02, 1.98800e+01],
        [4.66600e-02, 8.00000e+01, 1.52000e+00, 0.00000e+00, 4.04000e-01,
         7.10700e+00, 3.66000e+01, 7.30900e+00, 2.00000e+00, 3.29000e+02,
         1.26000e+01, 3.54310e+02, 8.61000e+00],
        [6.46600e-02, 7.00000e+01, 2.24000e+00, 0.00000e+00, 4.00000e-01,
         6.34500e+00, 2.01000e+01, 7.82780e+00, 5.00000e+00, 3.58000e+02,
         1.48000e+01, 3.68240e+02, 4.97000e+00],
        [1.35472e+00, 0.00000e+00, 8.14000e+00, 0.00000e+00, 5.38000e-01,
         6.07200e+00, 1.00000e+02, 4.17500e+00, 4.00000e+00, 3.07000e+02,
         2.10000e+01, 3.76730e+02, 1.30400e+01],
        [8.30800e-02, 0.00000e+00, 2.46000e+00, 0.00000e+00, 4.88000e-01,
         5.60400e+00, 8.98000e+01, 2.98790e+00, 3.00000e+00, 1.93000e+02,
         1.78000e+01, 3.91000e+02, 1.39800e+01],
        [5.69175e+00, 0.00000e+00, 1.81000e+01, 0.00000e+00, 5.83000e-01,
         6.11400e+00, 7.98000e+01, 3.54590e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.92680e+02, 1.49800e+01],
        [1.95390e-01, 0.00000e+00, 1.08100e+01, 0.00000e+00, 4.13000e-01,
         6.24500e+00, 6.20000e+00, 5.28730e+00, 4.00000e+00, 3.05000e+02,
         1.92000e+01, 3.77170e+02, 7.54000e+00],
        [5.11358e+01, 0.00000e+00, 1.81000e+01, 0.00000e+00, 5.97000e-01,
         5.75700e+00, 1.00000e+02, 1.41300e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 2.60000e+00, 1.01100e+01],
        [1.00623e+01, 0.00000e+00, 1.81000e+01, 0.00000e+00, 5.84000e-01,
         6.83300e+00, 9.43000e+01, 2.08820e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 8.13300e+01, 1.96900e+01],
        [8.26500e-02, 0.00000e+00, 1.39200e+01, 0.00000e+00, 4.37000e-01,
         6.12700e+00, 1.84000e+01, 5.50270e+00, 4.00000e+00, 2.89000e+02,
         1.60000e+01, 3.96900e+02, 8.58000e+00],
        [1.18123e+01, 0.00000e+00, 1.81000e+01, 0.00000e+00, 7.18000e-01,
         6.82400e+00, 7.65000e+01, 1.79400e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 4.84500e+01, 2.27400e+01],
        [5.78900e-02, 1.25000e+01, 6.07000e+00, 0.00000e+00, 4.09000e-01,
         5.87800e+00, 2.14000e+01, 6.49800e+00, 4.00000e+00, 3.45000e+02,
         1.89000e+01, 3.96210e+02, 8.10000e+00],
        [1.95100e-02, 1.75000e+01, 1.38000e+00, 0.00000e+00, 4.16100e-01,
         7.10400e+00, 5.95000e+01, 9.22290e+00, 3.00000e+00, 2.16000e+02,
         1.86000e+01, 3.93240e+02, 8.05000e+00],
        [8.79212e+00, 0.00000e+00, 1.81000e+01, 0.00000e+00, 5.84000e-01,
         5.56500e+00, 7.06000e+01, 2.06350e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.65000e+00, 1.71600e+01],
        [1.30100e-02, 3.50000e+01, 1.52000e+00, 0.00000e+00, 4.42000e-01,
         7.24100e+00, 4.93000e+01, 7.03790e+00, 1.00000e+00, 2.84000e+02,
         1.55000e+01, 3.94740e+02, 5.49000e+00],
        [1.70900e-02, 9.00000e+01, 2.02000e+00, 0.00000e+00, 4.10000e-01,
         6.72800e+00, 3.61000e+01, 1.21265e+01, 5.00000e+00, 1.87000e+02,
         1.70000e+01, 3.84460e+02, 4.50000e+00],
        [4.46200e-02, 2.50000e+01, 4.86000e+00, 0.00000e+00, 4.26000e-01,
         6.61900e+00, 7.04000e+01, 5.40070e+00, 4.00000e+00, 2.81000e+02,
         1.90000e+01, 3.95630e+02, 7.22000e+00],
        [5.20580e-01, 0.00000e+00, 6.20000e+00, 1.00000e+00, 5.07000e-01,
         6.63100e+00, 7.65000e+01, 4.14800e+00, 8.00000e+00, 3.07000e+02,
         1.74000e+01, 3.88450e+02, 9.54000e+00],
        [5.70818e+00, 0.00000e+00, 1.81000e+01, 0.00000e+00, 5.32000e-01,
         6.75000e+00, 7.49000e+01, 3.33170e+00, 2.40000e+01, 6.66000e+02,
         2.02000e+01, 3.93070e+02, 7.74000e+00],
        [2.44668e+00, 0.00000e+00, 1.95800e+01, 0.00000e+00, 8.71000e-01,
         5.27200e+00, 9.40000e+01, 1.73640e+00, 5.00000e+00, 4.03000e+02,
         1.47000e+01, 8.86300e+01, 1.61400e+01],
        [6.16200e-02, 0.00000e+00, 4.39000e+00, 0.00000e+00, 4.42000e-01,
         5.89800e+00, 5.23000e+01, 8.01360e+00, 3.00000e+00, 3.52000e+02,
         1.88000e+01, 3.64610e+02, 1.26700e+01],
        [6.32000e-03, 1.80000e+01, 2.31000e+00, 0.00000e+00, 5.38000e-01,
         6.57500e+00, 6.52000e+01, 4.09000e+00, 1.00000e+00, 2.96000e+02,
         1.53000e+01, 3.96900e+02, 4.98000e+00]]),
 array([16.7, 23.1, 13.3, 19.7, 20.2, 50. , 21.4,  8.3, 17.5, 28.5, 18.8,
        25. , 13.5, 22.2, 24.8, 19.4, 29.6, 32. , 13.3, 50. , 22. ,  7. ,
        10.2, 10.2, 12.8, 17.1,  5.6, 23.3, 19.6, 21.2, 21.6, 25. , 19. ,
        27.1, 14.1, 36.2, 23.1, 16.1, 20.1, 25.2, 24.4, 30.1, 41.7, 25. ,
        19.9, 30.8, 33.8, 11.8, 16.6, 37. , 22.8, 23.2, 17. , 22.7, 20. ,
        21.4, 13.6, 13.6, 24.2, 10.4, 15.2, 50. , 18.9, 42.8, 21.1,  7.4,
        25.3, 24.6, 16.8, 12.6, 14.2, 14.5, 23.7, 24.4, 14.8, 23.8, 19.6,
        22.6, 18.7, 22.8, 12.1, 29.6, 18.2, 36.2, 23.9, 23.6, 19.4, 12.7,
        17.6, 32.2, 13.8, 10.4, 18.6, 29. , 27.9, 36. , 23.9, 22. , 22.9,
        22. , 19.3, 17.5, 26.5, 10.9, 18.9, 20.8, 34.9, 50. , 19.3, 27.5,
        15.7, 19.5, 28.7, 13.4, 17.2, 20.1,  8.8,  7.2, 31.7, 10.5, 41.3,
        18.9, 14.9, 19.2, 50. , 23. , 26.6, 22.6, 18.4, 13.4, 18.4, 20.9,
        23.8, 19.6, 19.5, 36.4, 46.7, 18.3, 23.4, 50. , 13.8, 19.3, 50. ,
        35.4, 20.8, 21. , 12.3, 20.5, 21.1, 21.2, 24.1, 25. , 28.2, 23.3,
        19.8, 19.5, 20.6, 36.5, 15.6, 21. , 11.9, 20.6, 25. , 20.5, 21.7,
         9.7, 22.4, 33.2, 37.2, 12. ,  8.7, 13. , 33.1, 15. ,  9.5, 31. ,
        15.6, 16.5, 21.8, 22.3, 17.1, 30.1, 50. , 19.6,  7. , 13.4, 29.9,
        34.7, 39.8,  8.3, 12.7, 31.5, 21.7, 19.1, 15.4, 16.5, 26.4, 16.1,
        45.4, 17.7, 21.9, 22.6, 16.7, 22.1, 22.8, 43.8, 13.1, 50. , 23.8,
        22.4, 35.4, 15.4, 26.7, 15.2, 22.9,  5. , 37.3, 24.5, 28.1, 20.7,
        24.6, 42.3, 17.8, 18.5, 21.7, 33.2, 18.2,  8.8, 21.7, 10.5, 15.2,
        11. , 17.3, 21.4, 22.8, 44.8, 15.3, 15.6, 29.4, 18.6, 22.9, 23.3,
        22.6, 21.5, 14.4, 27. , 20.4, 13.8, 17.4, 33.4, 17.1, 37.9, 17.4,
        16.6, 14.3, 19.4, 38.7, 24.4,  9.6, 21.7, 20.3, 50. , 23.9, 48.8,
        19.3, 34.9, 23.1, 20.4, 20.9, 15. , 24.7, 27.5, 29.1, 28.7, 34.9,
        21.6, 33.3,  8.4, 22.7, 11.9, 23.1, 22. , 30.7, 18.8, 21.7, 14. ,
        19.6, 32.4, 21.4, 14.9, 20. , 13.2, 23.8, 13.8, 19.9, 28.6, 16.2,
        23.7, 13.5, 14.3, 21.4, 46. , 25. , 28.4, 18.4, 19.9,  5. , 22.3,
        29. , 27.5, 18.5, 23.6, 21.5, 18.5, 19.2, 20.3, 50. , 20.4, 31.2,
        19.9,  7.2, 20.6, 24.3,  8.5, 18.7, 29.1, 27.9, 15.6, 20.6, 16.2,
        12.5, 20.5, 33.4, 24.5, 19.4, 31.1, 31.6, 20.1, 24.1, 19.7, 24.1,
        10.8, 20.7, 20. , 22.2, 24.8, 23. , 20. , 50. , 18.1, 20.6, 21.7,
        37.6, 18.2, 17.8, 34.6,  7.2, 23.1, 24. , 16.4, 20.1, 17.4, 23.1,
        20.2, 26.6, 30.5, 19.5, 26.6, 19.4, 23.2, 23.7, 24.3, 15.6, 14.6,
        19.8, 10.2, 14.9, 19.1, 13.1, 43.1, 19. , 27.5, 22.2, 33.1, 24.3,
        22.6, 21.9, 20.1, 50. , 11.7, 22.5, 18.5, 43.5, 22.9,  7.5, 18.9,
        28. , 14.1, 15.1, 17.5, 21.2, 32. , 23.2, 31.5, 18.3, 19.3, 19.8,
        23. , 50. , 19.1, 32.5, 21.2, 24.8, 28.7, 29.8, 16.8, 20.4, 36.1,
        20. , 13.4, 27.1, 21.9, 13.1, 25. , 14.6, 24.7, 44. , 13.9, 35.2,
        16.1, 11.5, 17.8, 20.3, 13.8, 16.3, 22.2, 50. , 14.4, 26.2, 18. ,
        17.9, 31.6, 28.4, 10.9, 16. , 18.7, 22. , 25. , 35.1, 24.5, 20.3,
        48.5, 19.4, 20.6, 22. ]),
 array([ 8.1, 23.5, 12.7, 32.9, 21.8, 21.2, 23.2, 24.7, 17.8, 17.8, 50. ,
        20.8, 23. , 48.3, 17.2, 11.3, 21. , 11.8, 22.5, 13.3, 29.8, 13.9,
        22.2,  6.3, 23.3, 24.4, 23.1,  8.5, 24.8, 14.5, 30.3, 22.5, 14.5,
        26.4, 19.1, 23.4, 15. , 14.1, 23.9,  8.4, 22. , 33. , 11.7, 32.7,
        30.1, 23.9, 25.1, 23.7, 13.1, 17.2, 24. ])]
```

In [32]:
```
# Save 90% training data, 10% testing data, 90% label, 10% label to four varia
bels
x_train, x_test, y_train, y_test = train_test_split(boston["data"],
                                                    boston["target"],
                                                    test_size=0.1)
```

In [ ]:

Training the model

- Decision Tree

In [92]:
```
from sklearn.tree import DecisionTreeRegressor

# Create a model called reg
reg1 = DecisionTreeRegressor()
```

In [93]:
```
# Train model: Using training data & corresponding labels
reg1.fit(x_train, y_train)
```

Out[93]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

Prediction

In [139]:
```
import numpy as np
```

In [141]:
```
predict = reg1.predict(x_test)
print("Actual Price:", y_test[:5])
print()
print("Predicted Price:", predict[:5])
interval = np.subtract(predict, y_test)
print()
print("Difference:", interval[:5])
```

```
Actual Price: [23.1 20.  12.6 22.7 25.3]

Predicted Price: [24.5 18.9 18.4 20.2 24.7]

Difference: [ 1.4 -1.1  5.8 -2.5 -0.6]
```

R square

R-squared is a statistical measure of how close the data are to the fitted regression line

In [98]:
```
from sklearn.metrics import r2_score
```

In [100]:
```
# Training
R2_training = r2_score(y_train, reg1.predict(x_train))
print("r2 score:", R2_training)
```

```
r2 score: 1.0
```

In [155]:
```
# Testing
R2_testing = r2_score(y_test, reg1.predict(x_test))
print("r2 score:", R2)
```

```
r2 score: 0.8423644079430607
```

In [ ]:

Prevent overfitting: pre-pruning

In [144]:
```
reg2 = DecisionTreeRegressor(max_depth=3) # set the parameter: max_depth=3
reg2.fit(x_train, y_train)
```

Out[144]:
```
DecisionTreeRegressor(criterion='mse', max_depth=3, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

Plot the decision tree: using graphviz

We need to install this third-party libraries first
How To Install graphviz on Mac with Brew?

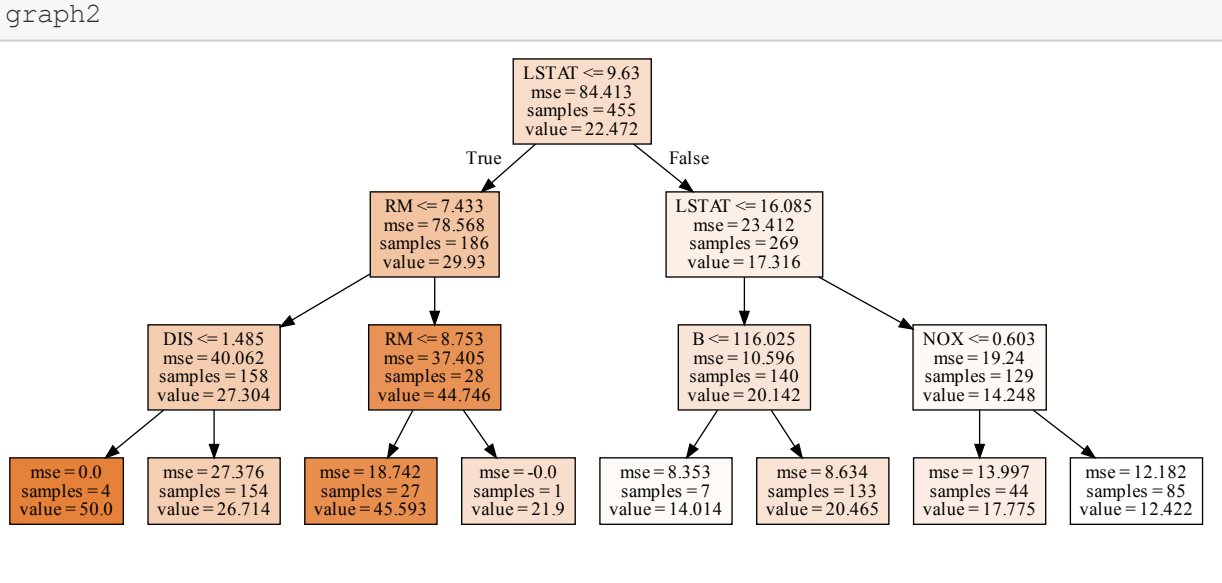
- Launch Terminal
- type "brew install graphviz"

In [145]:
```
import graphviz
from sklearn.tree import export_graphviz
```

In [146]:
```
# Remeber add the parameter:  out_file = None, otherwise it will cause error
g2 = export_graphviz(reg2, feature_names=boston["feature_names"], filled=True,
out_file = None)
```

In [147]:
```
graph2 = graphviz.Source(g2)
graph2
```

Out[147]:



Prediction

In [148]:
```
predict = reg2.predict(x_test)
print("Actual Price:", y_test[:5])
print()
print("Predicted Price:", predict[:5])
interval = np.subtract(predict, y_test)
print()
print("Difference:", interval[:5])
```

```
Actual Price: [23.1 20.  12.6 22.7 25.3]

Predicted Price: [26.71428571 20.46466165 12.42235294 20.46466165 26.714285
71]

Difference: [ 3.61428571  0.46466165 -0.17764706 -2.23533835  1.41428571]
```

R square

In [153]:
```
# Training
R2_training = r2_score(y_train, reg2.predict(x_train))
print("r2 score:", R2_training)
```

```
r2 score: 0.8026471880061703
```

In [154]:
```
# Testing
R2_testing = r2_score(y_test, reg2.predict(x_test))
print("r2 score:", R2)
```

```
r2 score: 0.8423644079430607
```

Measure the importance of every feature

In [94]:
```
print(boston["feature_names"])
reg1.feature_importances_
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

Out[94]:
```
array([0.02436802, 0.00195035, 0.0069382 , 0.00065952, 0.02622918,
       0.29244341, 0.01544867, 0.06654938, 0.00149788, 0.00896696,
       0.00752606, 0.01393326, 0.53348911])
```

export the decision tree picture to pdf file

In [156]:
```
graph2.render("boston") # export the picture to pdf
```

Out[156]:
```
'boston.pdf'
```

In [157]:
```
ls
```

```
Boston_Regression.ipynb  boston              house.png
R2.png                   boston.pdf          house2.png
```