Boston House Price Prediction

March 10, 2019

```
In [2]: %matplotlib inline
        import numpy as np
        from matplotlib import pyplot as plt
        from sklearn.datasets import load_boston
        import pandas as pd
        boston = load_boston()
```

```
In [3]: type(boston)
        type(boston.data)
        boston.feature_names
        print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usu

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 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 by 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.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon Universit

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 diagnostics
...', 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 that address regressio
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the
```

In [4]:
```python
bos = pd.DataFrame(boston.data)
bos.columns = boston.feature_names
bos['PRICE'] = boston.target
```

In [5]:
```python
bos.head()
```

Out[5]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	

	PTRATIO	B	LSTAT	PRICE
0	15.3	396.90	4.98	24.0
1	17.8	396.90	9.14	21.6
2	17.8	392.83	4.03	34.7
3	18.7	394.63	2.94	33.4
4	18.7	396.90	5.33	36.2

In [6]:
```python
bos.corr(method='spearman')
```

Out[6]:
```
             CRIM        ZN     INDUS      CHAS       NOX        RM       AGE  \
CRIM     1.000000 -0.571660  0.735524  0.041537  0.821465 -0.309116  0.704140
ZN      -0.571660  1.000000 -0.642811 -0.041937 -0.634828  0.361074 -0.544423
INDUS    0.735524 -0.642811  1.000000  0.089841  0.791189 -0.415301  0.679487
CHAS     0.041537 -0.041937  0.089841  1.000000  0.068426  0.058813  0.067792
NOX      0.821465 -0.634828  0.791189  0.068426  1.000000 -0.310344  0.795153
RM      -0.309116  0.361074 -0.415301  0.058813 -0.310344  1.000000 -0.278082
AGE      0.704140 -0.544423  0.679487  0.067792  0.795153 -0.278082  1.000000
DIS     -0.744986  0.614627 -0.757080 -0.080248 -0.880015  0.263168 -0.801610
RAD      0.727807 -0.278767  0.455507  0.024579  0.586429 -0.107492  0.417983
TAX      0.729045 -0.371394  0.664361 -0.044486  0.649527 -0.271898  0.526366
PTRATIO  0.465283 -0.448475  0.433710 -0.136065  0.391309 -0.312923  0.355384
B       -0.360555  0.163135 -0.285840 -0.039810 -0.296662  0.053660 -0.228022
LSTAT    0.634760 -0.490074  0.638747 -0.050575  0.636828 -0.640832  0.657071
PRICE   -0.558891  0.438179 -0.578255  0.140612 -0.562609  0.633576 -0.547562

              DIS       RAD       TAX   PTRATIO         B     LSTAT     PRICE
CRIM    -0.744986  0.727807  0.729045  0.465283 -0.360555  0.634760 -0.558891
ZN       0.614627 -0.278767 -0.371394 -0.448475  0.163135 -0.490074  0.438179
INDUS   -0.757080  0.455507  0.664361  0.433710 -0.285840  0.638747 -0.578255
CHAS    -0.080248  0.024579 -0.044486 -0.136065 -0.039810 -0.050575  0.140612
NOX     -0.880015  0.586429  0.649527  0.391309 -0.296662  0.636828 -0.562609
RM       0.263168 -0.107492 -0.271898 -0.312923  0.053660 -0.640832  0.633576
AGE     -0.801610  0.417983  0.526366  0.355384 -0.228022  0.657071 -0.547562
DIS      1.000000 -0.495806 -0.574336 -0.322041  0.249595 -0.564262  0.445857
RAD     -0.495806  1.000000  0.704876  0.318330 -0.282533  0.394322 -0.346776
TAX     -0.574336  0.704876  1.000000  0.453345 -0.329843  0.534423 -0.562411
PTRATIO -0.322041  0.318330  0.453345  1.000000 -0.072027  0.467259 -0.555905
B        0.249595 -0.282533 -0.329843 -0.072027  1.000000 -0.210562  0.185664
LSTAT   -0.564262  0.394322  0.534423  0.467259 -0.210562  1.000000 -0.852914
PRICE    0.445857 -0.346776 -0.562411 -0.555905  0.185664 -0.852914  1.000000
```

In [7]:
```python
X = bos[['CRIM']]
y = bos['NOX']
type(X)
```

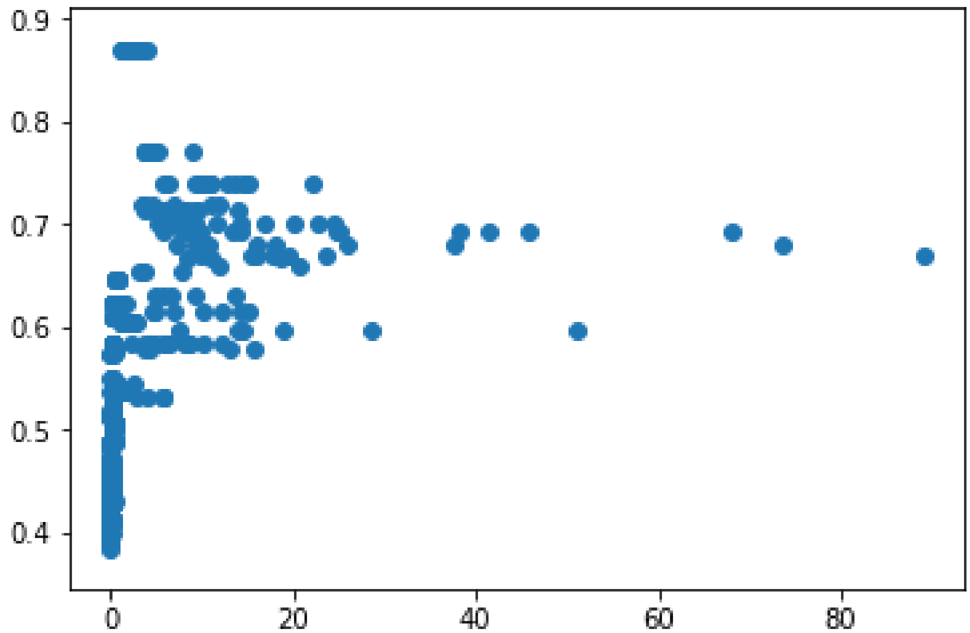
Out[7]:
```
pandas.core.frame.DataFrame
```

In [8]:
```python
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state
#X_train, y_train = X, y

lm = LinearRegression()
lm.fit(X_train, y_train)
```

Out[8]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [9]: plt.scatter(X, y)

Out[9]: <matplotlib.collections.PathCollection at 0x7f20ca7f07b8>




```
In [21]: from sklearn.metrics import r2_score
         from numpy import corrcoef

In [23]: pred = lm.predict(X_test)
         print(r2_score(X_test.values, pred))
         print(corrcoef(y_test, pred)[0])
         from scipy.stats import pearsonr
         r, p  = pearsonr(X_test.values.reshape(len(X_test.values)), pred)
         r

0.98
0.93333333333


Out[23]: 0.99999999999999978

In [ ]:
```