Intermediate_Python_Notes

November 11, 2018

1 BUDSA Intermediate Python Tutorial

Covering some key functionalities of Python libraries like `numpy`, `pandas`, and `sklearn`. Also covers some details of putting Python code into clean/reusable functions.

1.1 Load Boston House Price Data from `sklearn`

```
In [49]: from sklearn.datasets import load_boston
         boston = dict(load_boston())
         print(boston.keys())
```

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

```
In [47]: 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 25,000 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.
http://archive.ics.uci.edu/ml/datasets/Housing


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

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 regress
problems.

**References**

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources o
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on th
   - many more! (see http://archive.ics.uci.edu/ml/datasets/Housing)
```

1.2 Manipulate data with numpy and View data in a pandas DataFrame

In [50]:
```
import numpy as np
import pandas as pd
rows = np.concatenate((boston['data'], boston['target'].reshape(-1, 1)), axis=1)
columns = np.append(boston['feature_names'], 'Price')
df = pd.DataFrame(rows, columns=columns)
df
```

Out[50]:
```
         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
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0
10	0.22489	12.5	7.87	0.0	0.524	6.377	94.3	6.3467	5.0	311.0
11	0.11747	12.5	7.87	0.0	0.524	6.009	82.9	6.2267	5.0	311.0
12	0.09378	12.5	7.87	0.0	0.524	5.889	39.0	5.4509	5.0	311.0
13	0.62976	0.0	8.14	0.0	0.538	5.949	61.8	4.7075	4.0	307.0
14	0.63796	0.0	8.14	0.0	0.538	6.096	84.5	4.4619	4.0	307.0
15	0.62739	0.0	8.14	0.0	0.538	5.834	56.5	4.4986	4.0	307.0
16	1.05393	0.0	8.14	0.0	0.538	5.935	29.3	4.4986	4.0	307.0
17	0.78420	0.0	8.14	0.0	0.538	5.990	81.7	4.2579	4.0	307.0
18	0.80271	0.0	8.14	0.0	0.538	5.456	36.6	3.7965	4.0	307.0
19	0.72580	0.0	8.14	0.0	0.538	5.727	69.5	3.7965	4.0	307.0
20	1.25179	0.0	8.14	0.0	0.538	5.570	98.1	3.7979	4.0	307.0
21	0.85204	0.0	8.14	0.0	0.538	5.965	89.2	4.0123	4.0	307.0
22	1.23247	0.0	8.14	0.0	0.538	6.142	91.7	3.9769	4.0	307.0
23	0.98843	0.0	8.14	0.0	0.538	5.813	100.0	4.0952	4.0	307.0
24	0.75026	0.0	8.14	0.0	0.538	5.924	94.1	4.3996	4.0	307.0
25	0.84054	0.0	8.14	0.0	0.538	5.599	85.7	4.4546	4.0	307.0
26	0.67191	0.0	8.14	0.0	0.538	5.813	90.3	4.6820	4.0	307.0
27	0.95577	0.0	8.14	0.0	0.538	6.047	88.8	4.4534	4.0	307.0
28	0.77299	0.0	8.14	0.0	0.538	6.495	94.4	4.4547	4.0	307.0
29	1.00245	0.0	8.14	0.0	0.538	6.674	87.3	4.2390	4.0	307.0
..	...	...	...	...	...	...	...	...	...	...
476	4.87141	0.0	18.10	0.0	0.614	6.484	93.6	2.3053	24.0	666.0
477	15.02340	0.0	18.10	0.0	0.614	5.304	97.3	2.1007	24.0	666.0
478	10.23300	0.0	18.10	0.0	0.614	6.185	96.7	2.1705	24.0	666.0
479	14.33370	0.0	18.10	0.0	0.614	6.229	88.0	1.9512	24.0	666.0
480	5.82401	0.0	18.10	0.0	0.532	6.242	64.7	3.4242	24.0	666.0
481	5.70818	0.0	18.10	0.0	0.532	6.750	74.9	3.3317	24.0	666.0
482	5.73116	0.0	18.10	0.0	0.532	7.061	77.0	3.4106	24.0	666.0
483	2.81838	0.0	18.10	0.0	0.532	5.762	40.3	4.0983	24.0	666.0
484	2.37857	0.0	18.10	0.0	0.583	5.871	41.9	3.7240	24.0	666.0
485	3.67367	0.0	18.10	0.0	0.583	6.312	51.9	3.9917	24.0	666.0
486	5.69175	0.0	18.10	0.0	0.583	6.114	79.8	3.5459	24.0	666.0
487	4.83567	0.0	18.10	0.0	0.583	5.905	53.2	3.1523	24.0	666.0
488	0.15086	0.0	27.74	0.0	0.609	5.454	92.7	1.8209	4.0	711.0
489	0.18337	0.0	27.74	0.0	0.609	5.414	98.3	1.7554	4.0	711.0
490	0.20746	0.0	27.74	0.0	0.609	5.093	98.0	1.8226	4.0	711.0
491	0.10574	0.0	27.74	0.0	0.609	5.983	98.8	1.8681	4.0	711.0
492	0.11132	0.0	27.74	0.0	0.609	5.983	83.5	2.1099	4.0	711.0
493	0.17331	0.0	9.69	0.0	0.585	5.707	54.0	2.3817	6.0	391.0
494	0.27957	0.0	9.69	0.0	0.585	5.926	42.6	2.3817	6.0	391.0
495	0.17899	0.0	9.69	0.0	0.585	5.670	28.8	2.7986	6.0	391.0

496	0.28960	0.0	9.69	0.0	0.585	5.390	72.9	2.7986	6.0	391.0
497	0.26838	0.0	9.69	0.0	0.585	5.794	70.6	2.8927	6.0	391.0
498	0.23912	0.0	9.69	0.0	0.585	6.019	65.3	2.4091	6.0	391.0
499	0.17783	0.0	9.69	0.0	0.585	5.569	73.5	2.3999	6.0	391.0
500	0.22438	0.0	9.69	0.0	0.585	6.027	79.7	2.4982	6.0	391.0
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0

	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
5	18.7	394.12	5.21	28.7
6	15.2	395.60	12.43	22.9
7	15.2	396.90	19.15	27.1
8	15.2	386.63	29.93	16.5
9	15.2	386.71	17.10	18.9
10	15.2	392.52	20.45	15.0
11	15.2	396.90	13.27	18.9
12	15.2	390.50	15.71	21.7
13	21.0	396.90	8.26	20.4
14	21.0	380.02	10.26	18.2
15	21.0	395.62	8.47	19.9
16	21.0	386.85	6.58	23.1
17	21.0	386.75	14.67	17.5
18	21.0	288.99	11.69	20.2
19	21.0	390.95	11.28	18.2
20	21.0	376.57	21.02	13.6
21	21.0	392.53	13.83	19.6
22	21.0	396.90	18.72	15.2
23	21.0	394.54	19.88	14.5
24	21.0	394.33	16.30	15.6
25	21.0	303.42	16.51	13.9
26	21.0	376.88	14.81	16.6
27	21.0	306.38	17.28	14.8
28	21.0	387.94	12.80	18.4
29	21.0	380.23	11.98	21.0
..	...	...	...	...
476	20.2	396.21	18.68	16.7
477	20.2	349.48	24.91	12.0
478	20.2	379.70	18.03	14.6
479	20.2	383.32	13.11	21.4
480	20.2	396.90	10.74	23.0

```
481      20.2  393.07   7.74   23.7
482      20.2  395.28   7.01   25.0
483      20.2  392.92  10.42   21.8
484      20.2  370.73  13.34   20.6
485      20.2  388.62  10.58   21.2
486      20.2  392.68  14.98   19.1
487      20.2  388.22  11.45   20.6
488      20.1  395.09  18.06   15.2
489      20.1  344.05  23.97    7.0
490      20.1  318.43  29.68    8.1
491      20.1  390.11  18.07   13.6
492      20.1  396.90  13.35   20.1
493      19.2  396.90  12.01   21.8
494      19.2  396.90  13.59   24.5
495      19.2  393.29  17.60   23.1
496      19.2  396.90  21.14   19.7
497      19.2  396.90  14.10   18.3
498      19.2  396.90  12.92   21.2
499      19.2  395.77  15.10   17.5
500      19.2  396.90  14.33   16.8
501      21.0  391.99   9.67   22.4
502      21.0  396.90   9.08   20.6
503      21.0  396.90   5.64   23.9
504      21.0  393.45   6.48   22.0
505      21.0  396.90   7.88   11.9

[506 rows x 14 columns]
```

1.3 Visualize Data with `matplotlib`

1.3.1 Create reusable functions

In [71]:
```
def plot_subplots(boston):
    import matplotlib.pyplot as plt
    fig, axis_array = plt.subplots(4, 4)
    index = 0
    for row in axis_array:
        for col in row:
            if index < len(boston['feature_names']):
                feature = boston['feature_names'][index]
                col.set_title('{} vs. Price'.format(feature))
                col.set_xlabel(feature)
                col.set_ylabel('Price')
                col.scatter(df[feature], df['Price'])
                index += 1
            else:
                break
    fig.set_size_inches(20, 20)
```

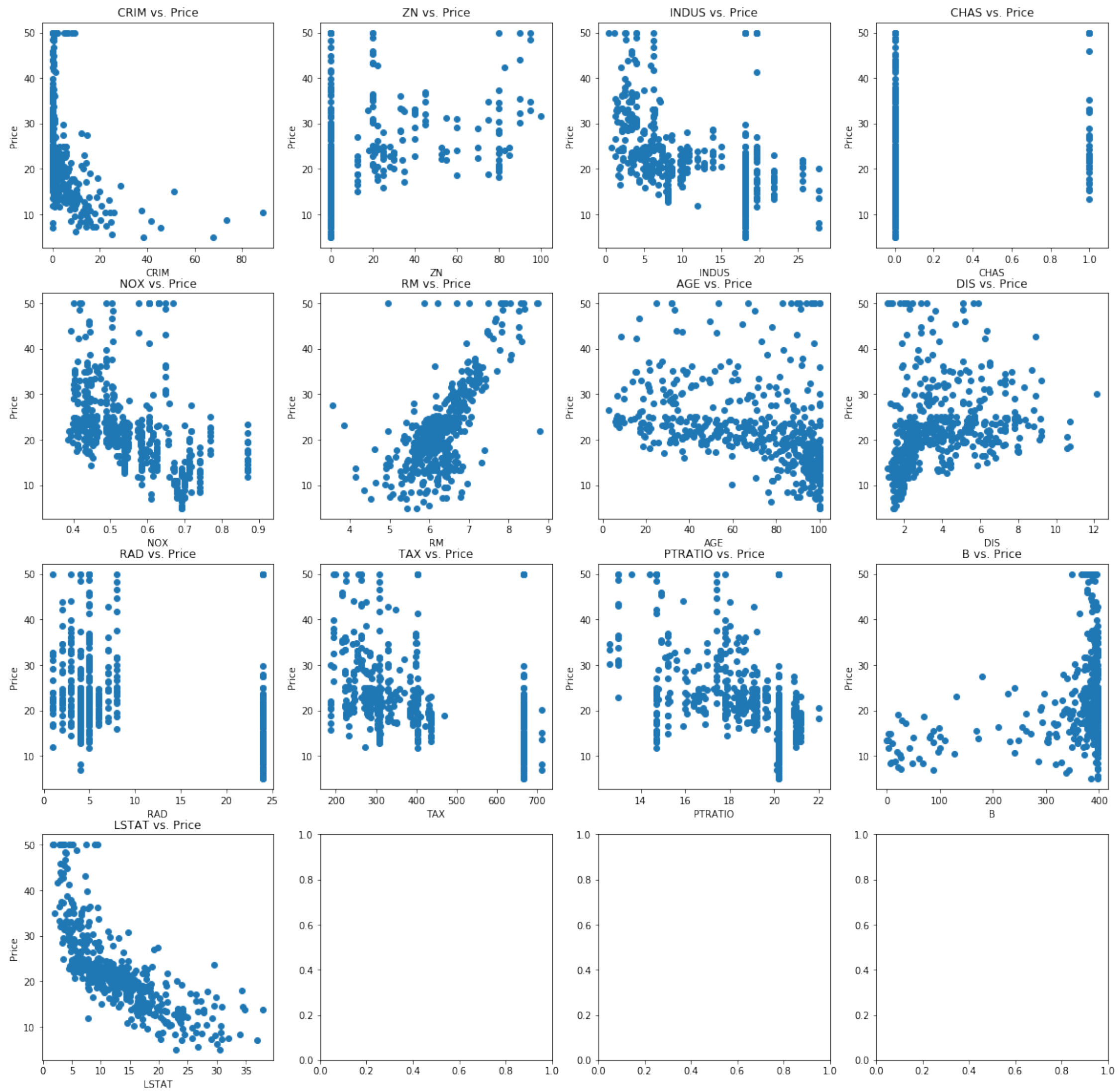
```
        fig.show()

    def plot_separate_plots(boston):
        for feature in boston['feature_names']:
            plt.figure()
            plt.title('{} vs. Price'.format(feature))
            plt.xlabel(feature)
            plt.ylabel('Price')
            plt.scatter(df[feature], df['Price'])
            plt.show()
```

1.4 Plot features vs. price in multiple subplots within a figure

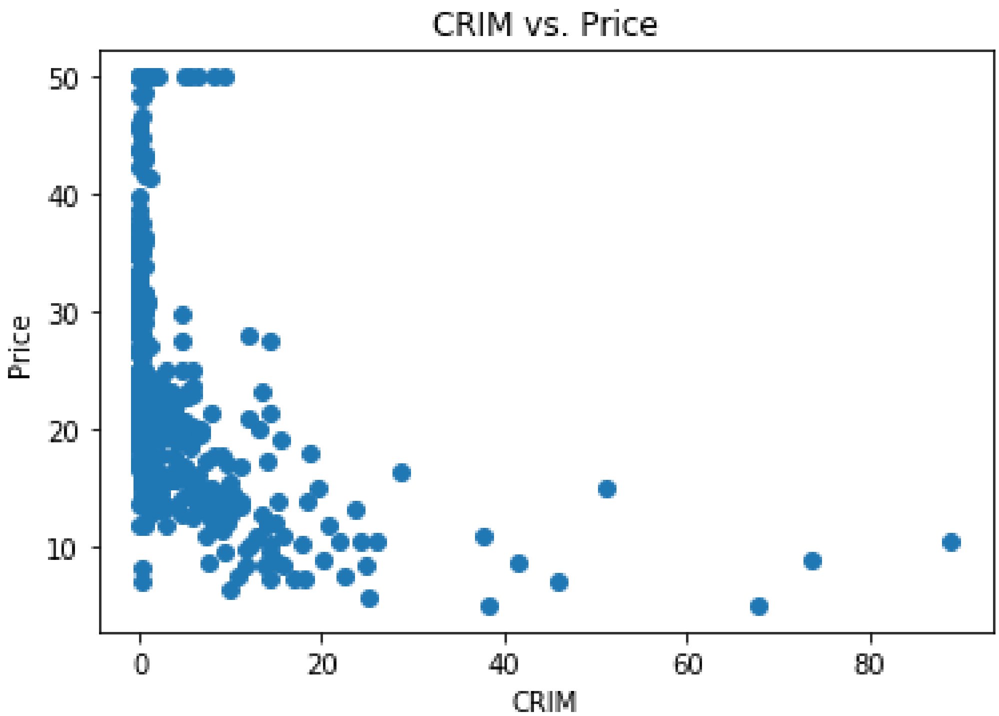
```
In [70]: plot_subplots(boston)
```

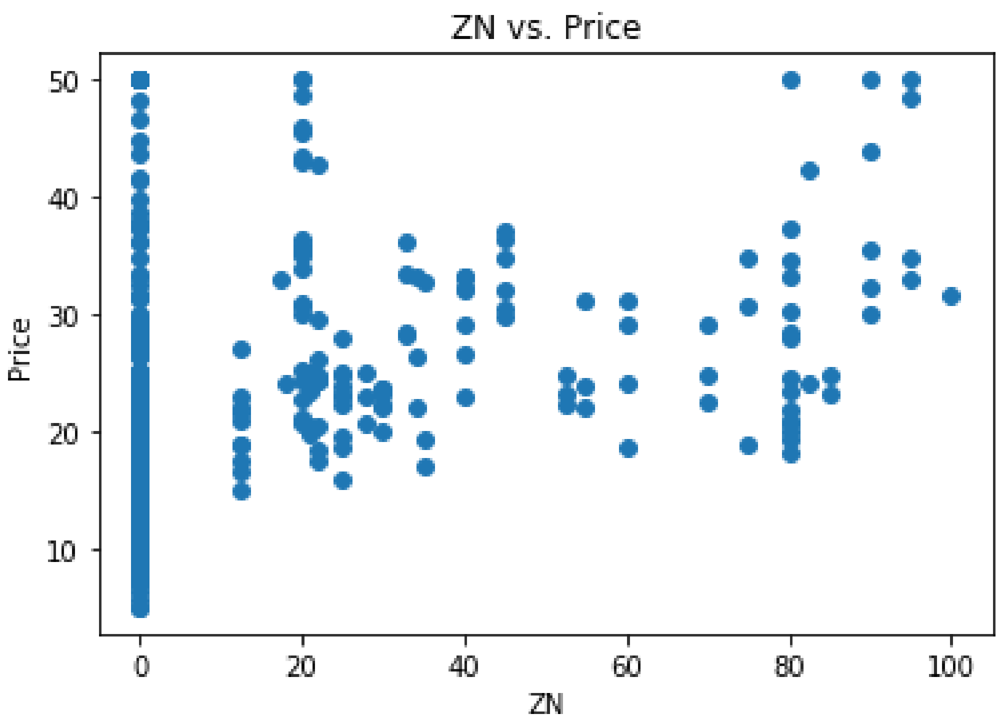
```
/home/ryan/.local/lib/python3.6/site-packages/matplotlib/figure.py:457: UserWarning: matplotlil
  "matplotlib is currently using a non-GUI backend, "
```

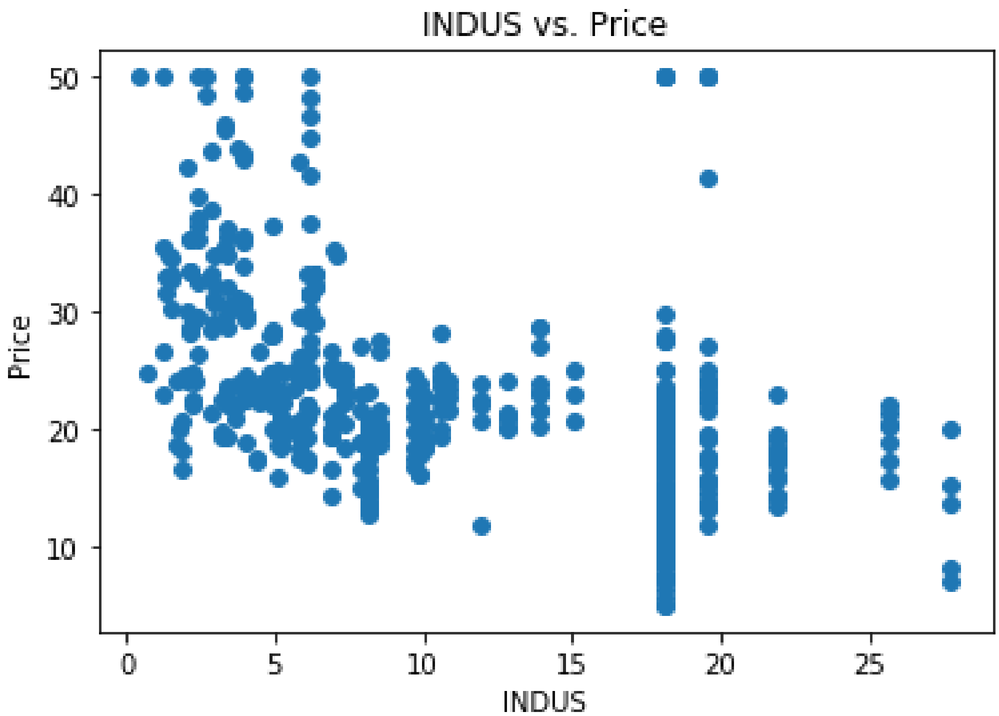


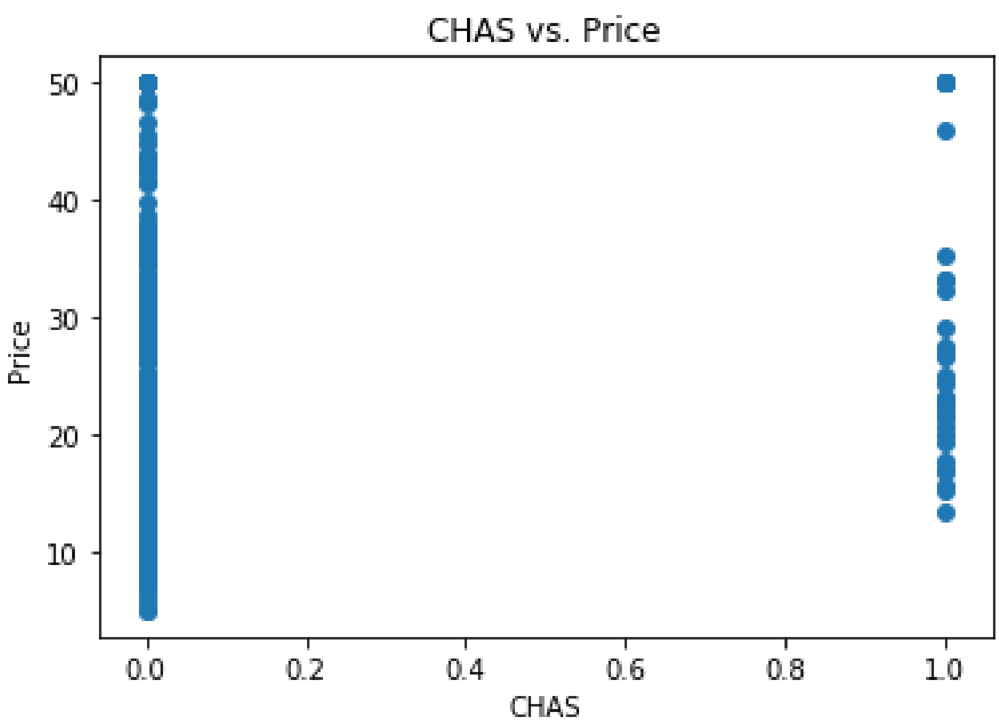
1.5 Plot features vs. price in separate plots

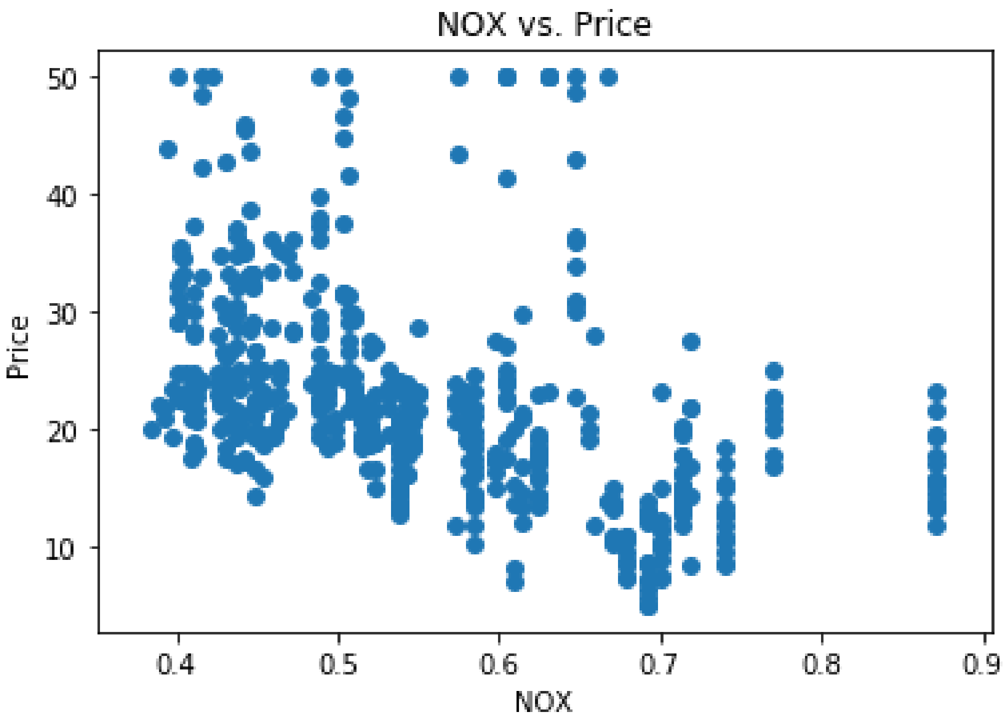
```
In [72]: plot_separate_plots(boston)
```


CRIM vs. Price
Price
CRIM
50
40
30
20
10
0
20
40
60
80

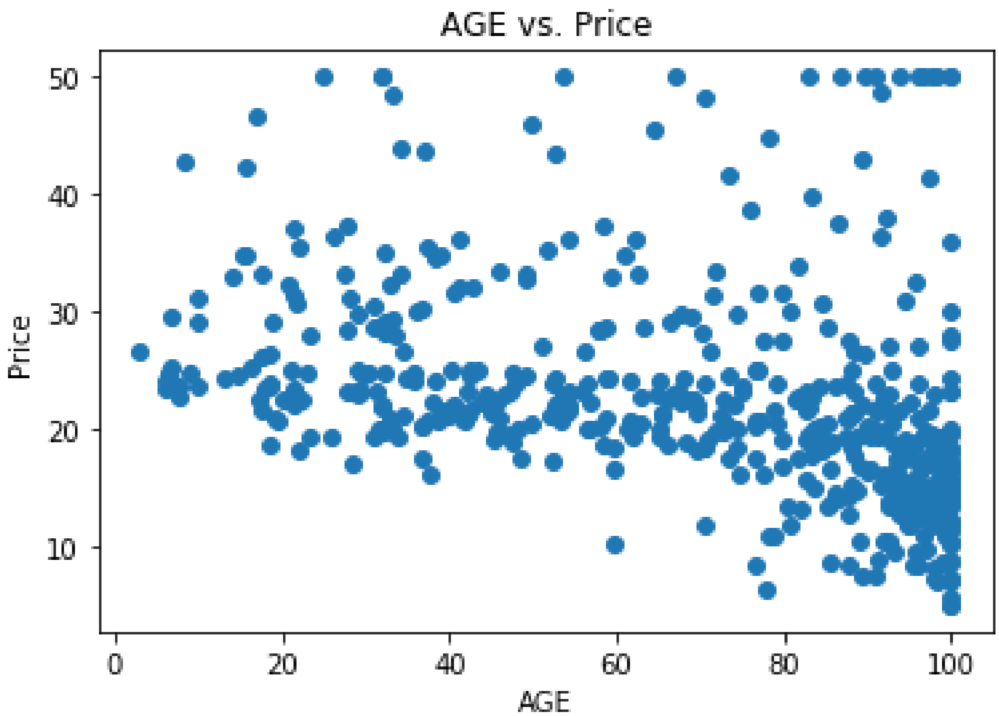

ZN vs. Price
Price
ZN
50
40
30
20
10
0
20
40
60
80
100

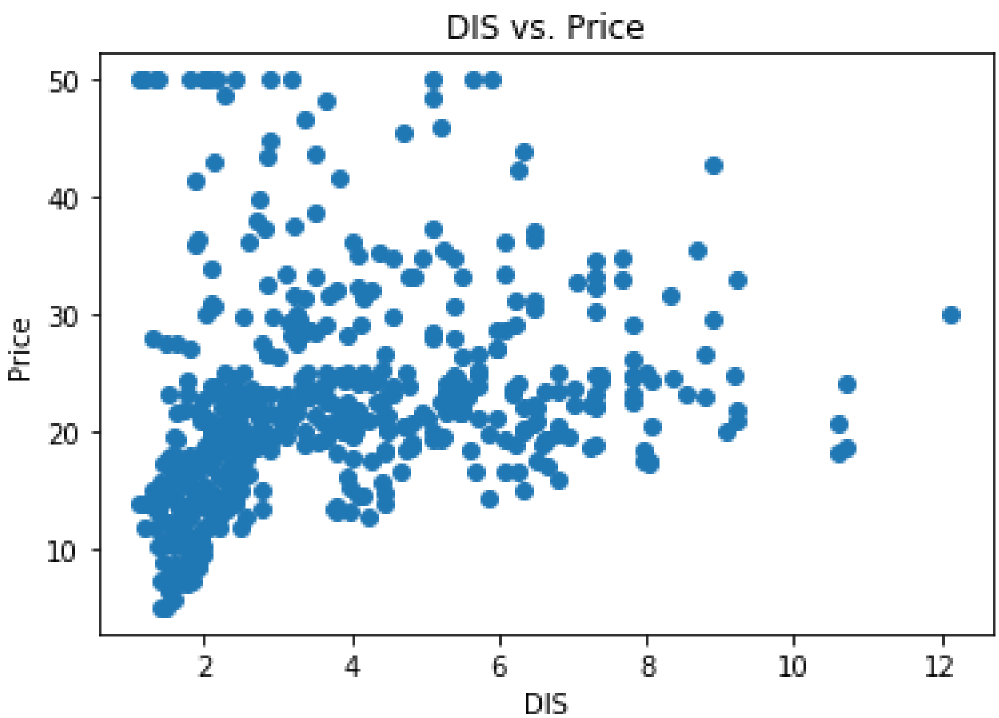

INDUS vs. Price
Price
INDUS
50
40
30
20
10
0
5
10
15
20
25

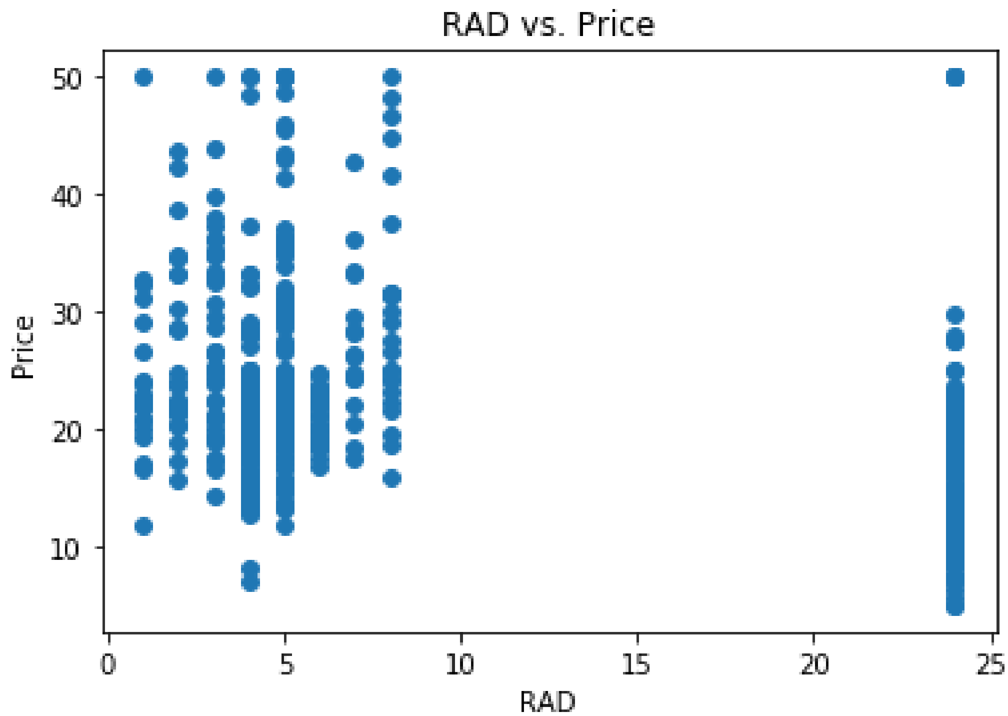

CHAS vs. Price
Price
CHAS
50
40
30
20
10
0.0
0.2
0.4
0.6
0.8
1.0

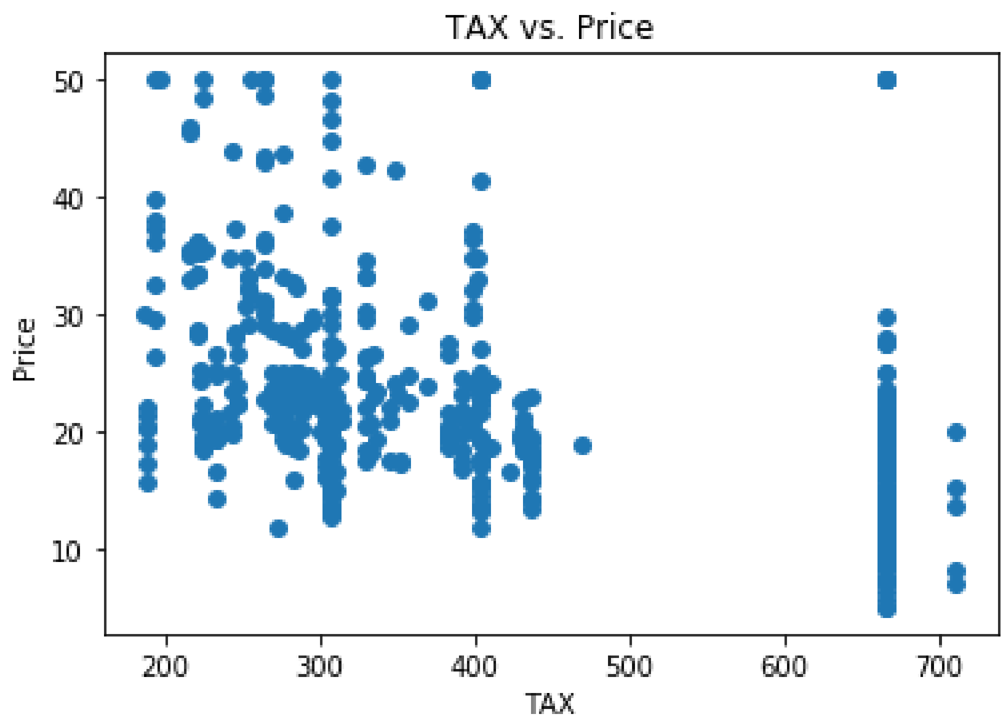

NOX vs. Price
Price
NOX
50
40
30
20
10
0.4
0.5
0.6
0.7
0.8
0.9

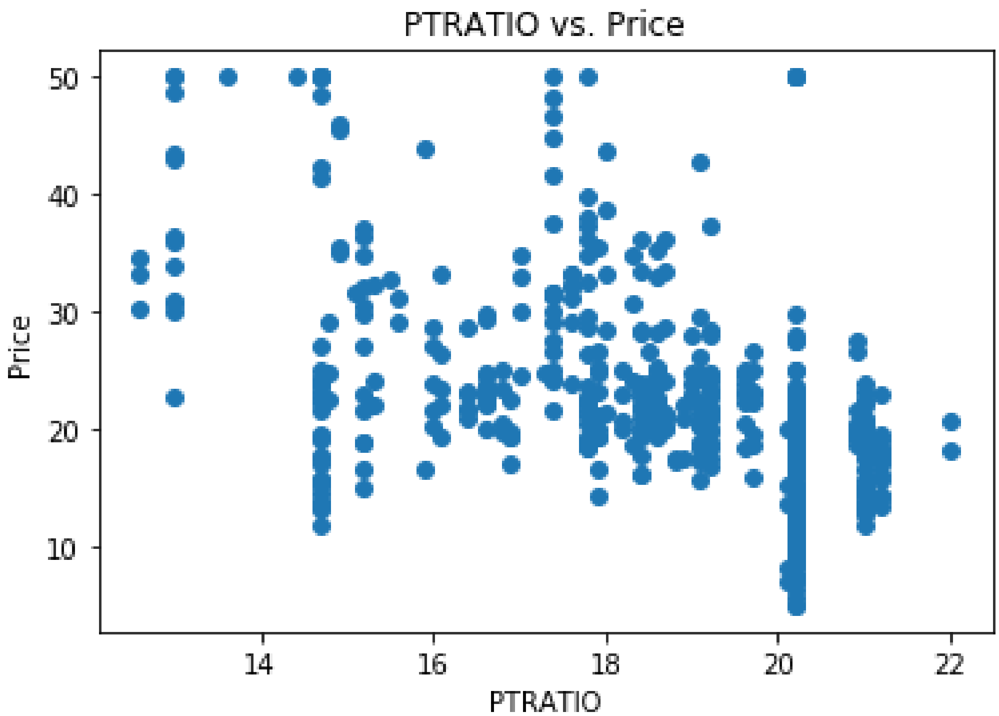

RM vs. Price
Price
RM
50
40
30
20
10
4
5
6
7
8
9

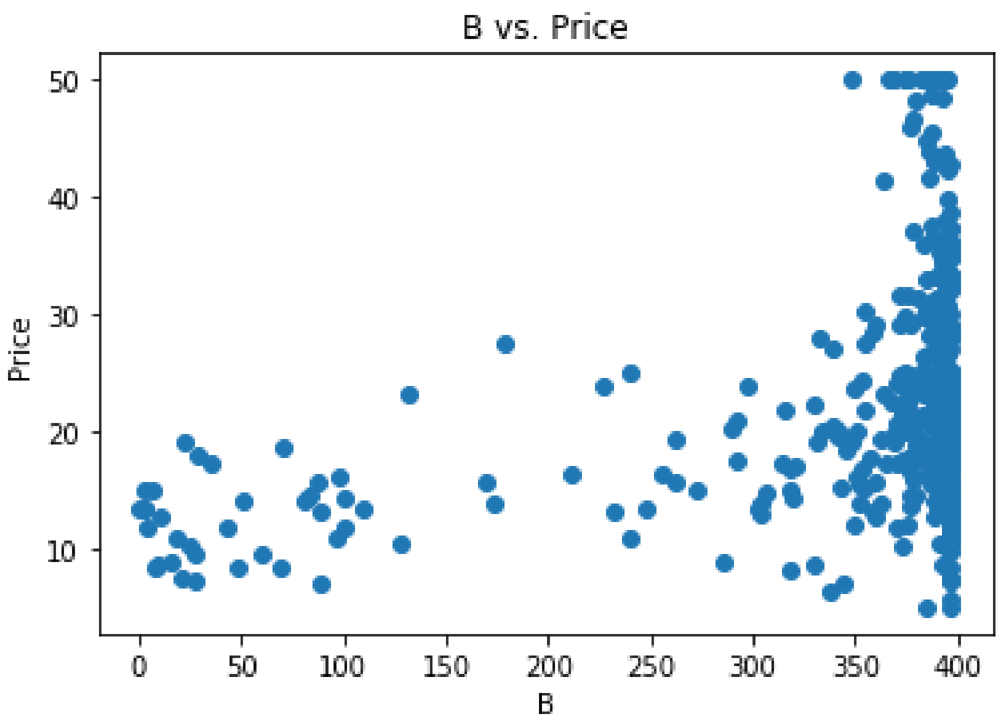

AGE vs. Price
Price
AGE
50
40
30
20
10
0
20
40
60
80
100

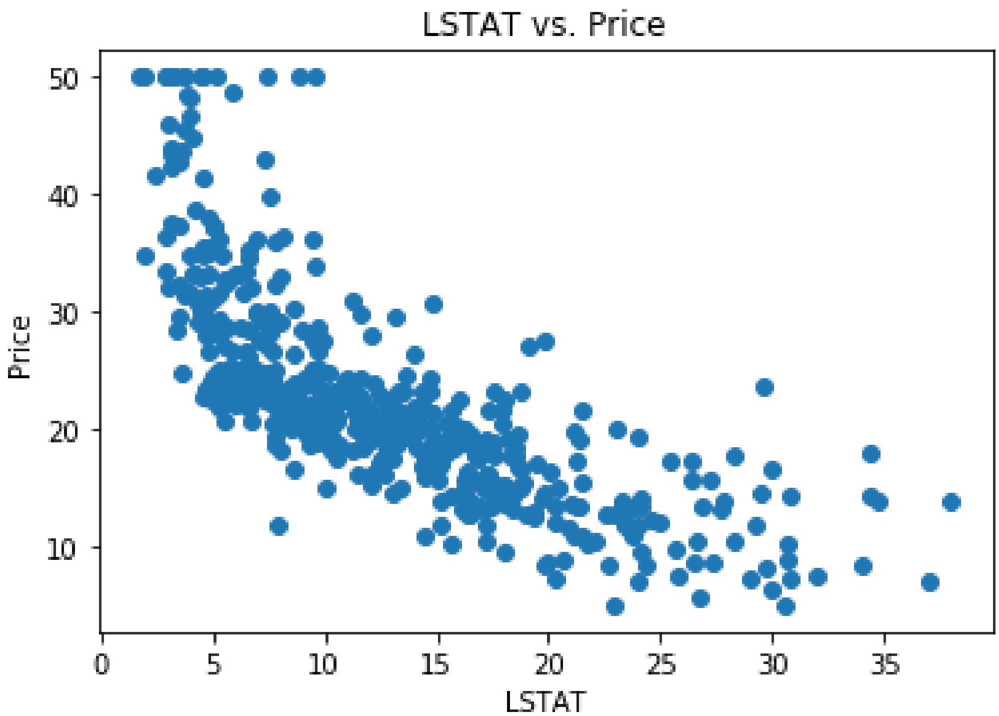

DIS vs. Price
Price
DIS
50
40
30
20
10
2
4
6
8
10
12


RAD vs. Price
Price
RAD
50
40
30
20
10
0
5
10
15
20
25


TAX vs. Price
Price
TAX
50
40
30
20
10
200
300
400
500
600
700


PTRATIO vs. Price
Price
PTRATIO
50
40
30
20
10
14
16
18
20
22


B vs. Price
Price
B
50
40
30
20
10
0
50
100
150
200
250
300
350
400



1.6 Try to predict house prices with Linear Regression using `sklearn`

`sklearn` machine learning models usually looking like this:

```
from sklearn import some_model
my_model = some_model()
my_model.fit(Xtrain, Ytrain)
my_model.predict(Xtest)
```

For the sake of a simpler tutorial, we aren't going to split our data into training and testing sets, but you should always do this when doing machine learning and data science!

```
In [116]: from sklearn.linear_model import LinearRegression
          # Write a reusable function to try LinearRegression models on different data
          def fit_and_test_lr_model(data, labels):
              model = LinearRegression()
              model.fit(data, labels)
              train_accuracy = model.score(data, labels)
              print('Accuracy: {}%'.format(train_accuracy*100.0))
```

1.6.1 Using all features

```
In [99]: fit_and_test_lr_model(boston['data'], boston['target'])
```

```
Accuracy: 74.06077428649428%
```

1.6.2 Hand-picking some good looking features

```
In [100]: fit_and_test_lr_model(df['RM'].values.reshape(-1, 1), df['Price'].values)
```

```
Accuracy: 48.35254559913343%
```

```
In [101]: fit_and_test_lr_model(df['LSTAT'].values.reshape(-1, 1), df['Price'].values)
```

```
Accuracy: 54.41462975864797%
```

```
In [115]: from IPython.display import display
          two_features_df = pd.DataFrame({'RM':df['RM'], 'LSTAT':df['LSTAT']})
          display(two_features_df.head(5))
          fit_and_test_lr_model(two_features_df.values, df['Price'].values)
```

	RM	LSTAT
0	6.575	4.98
1	6.421	9.14
2	7.185	4.03
3	6.998	2.94
4	7.147	5.33

```
Accuracy: 63.85616062603403%
```