load_boston

November 24, 2018

```
In [1]: import pandas as pd
        import numpy as np
```

```
In [3]: import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [4]: %matplotlib inline
```

```
In [47]: from sklearn.datasets import load_boston
```

```
In [48]: boston = load_boston()
```

```
In [57]: boston.keys()
```

```
Out[57]: ['data', 'feature_names', 'DESCR', 'target']
```

```
In [50]: boston.data , boston.feature_names,
```

```
Out[50]: (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]]),
          array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
                 'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='|S7'))
```

```
In [103]: 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 Universi

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)
```

In [52]: boston.target

```
Out[52]: 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])
```

```
In [53]: bos = pd.DataFrame(boston['data'])
         bos.columns = boston[('feature_names')]
         bos['Price'] = boston['target']
         bos.head()
```

```
Out[53]:       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 [54]: bos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
CRIM       506 non-null float64
ZN         506 non-null float64
INDUS      506 non-null float64
CHAS       506 non-null float64
NOX        506 non-null float64
RM         506 non-null float64
AGE        506 non-null float64
DIS        506 non-null float64
RAD        506 non-null float64
TAX        506 non-null float64
PTRATIO    506 non-null float64
B          506 non-null float64
LSTAT      506 non-null float64
Price      506 non-null float64
dtypes: float64(14)
memory usage: 55.4 KB
```

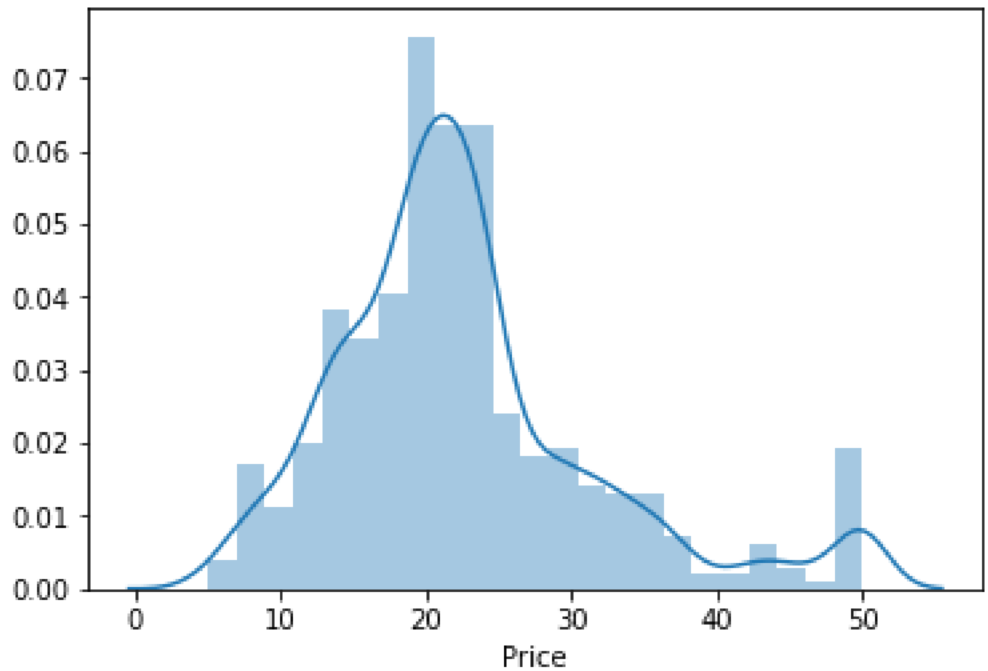
```
In [55]: bos.describe()
```

Out[55]:
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.593761   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.596783   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.647423   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT       Price
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

In [56]:
```python
sns.distplot(bos['Price'])
```

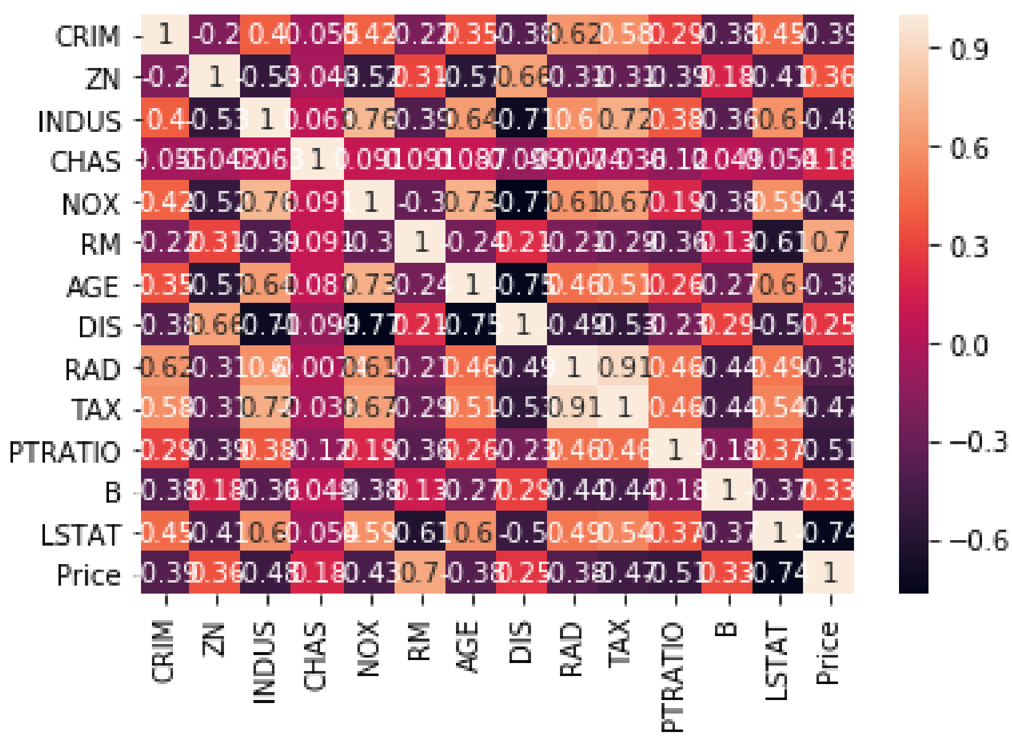
```
/anaconda2/lib/python2.7/site-packages/matplotlib/axes/_axes.py:6462: UserWarning: The 'normed
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1192efa50>
```




```
In [59]: sns.heatmap(bos.corr(), annot=True)
```

```
Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x11904bdd0>
```



```
In [60]: bos.columns

Out[60]: Index([u'CRIM', u'ZN', u'INDUS', u'CHAS', u'NOX', u'RM', u'AGE', u'DIS',
                u'RAD', u'TAX', u'PTRATIO', u'B', u'LSTAT', u'Price'],
               dtype='object')

In [61]: X = bos[[u'CRIM', u'ZN', u'INDUS', u'CHAS', u'NOX', u'RM', u'AGE', u'DIS',
                u'RAD', u'TAX', u'PTRATIO', u'B', u'LSTAT']]

In [68]: y = bos['Price']

In [70]: from sklearn.model_selection import train_test_split

In [85]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=

In [86]: from sklearn.linear_model import LinearRegression

In [87]: linear_model = LinearRegression()

In [88]: linear_model.fit(X_train,y_train)

Out[88]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [89]: linear_model.coef_

Out[89]: array([-6.71330988e-02,  4.00218287e-02, -1.46541639e-02,  2.59467565e+00,
                -1.57288770e+01,  3.73001907e+00, -8.98578854e-03, -1.33542114e+00,
                 2.80805340e-01, -1.18721183e-02, -8.89263070e-01,  1.05946405e-02,
                -4.88277689e-01])

In [90]: X_train.columns

Out[90]: Index([u'CRIM', u'ZN', u'INDUS', u'CHAS', u'NOX', u'RM', u'AGE', u'DIS',
                u'RAD', u'TAX', u'PTRATIO', u'B', u'LSTAT'],
               dtype='object')

In [92]: bosDF = pd.DataFrame(linear_model.coef_, X.columns, columns = ['Coef'])

In [93]: bosDF

Out[93]:              Coef
         CRIM     -0.067133
         ZN        0.040022
         INDUS    -0.014654
         CHAS      2.594676
         NOX     -15.728877
         RM        3.730019
         AGE      -0.008986
```

```
DIS       -1.335421
RAD        0.280805
TAX       -0.011872
PTRATIO   -0.889263
B          0.010595
LSTAT     -0.488278
```

In [94]:
```
predictions = linear_model.predict(X_test)
```

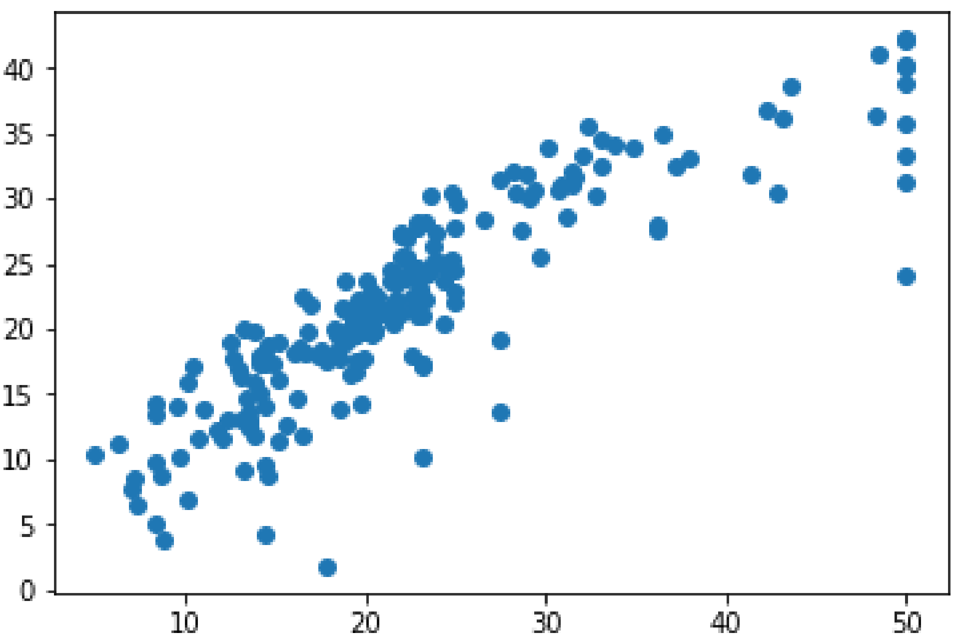
In [95]:
```
predictions
```

Out[95]:
```
array([17.88754504, 19.74749906, 16.02652993, 31.34192695, 11.52082795,
       30.14024503, 25.05274508, 11.62938272, 23.48782641,  3.91610379,
       11.1599503 , 32.16197185, 11.30618688, 18.82882352, 27.39797443,
       21.32651585, 26.34104432, 41.11454443, 27.84786841, 30.6709252 ,
       17.0687279 , 24.64494776, 35.74160515, 13.45339548, 25.40952107,
       27.16281402, 13.67212233,  5.12969405, 21.5473272 ,  8.7434751 ,
       24.58544379, 31.12209676, 13.8074543 , 13.00306968, 20.08552557,
       19.74539172, 25.59948378, 17.38709659, 17.6371365 , 24.6360056 ,
       17.97781597, 34.07685818, 22.2054959 , 21.15893523, 19.2366249 ,
        1.81025188, 22.53066692, 17.75606052, 14.10879532, 40.28690088,
        9.23927801, 33.27858104, 15.8117349 , 20.47515208, 22.39869851,
       24.437373  , 19.89550216, 19.46311289, 19.55294393, 31.06124288,
       20.95164093, 17.29154137, 17.19926229, 20.52414559, 13.10690674,
       12.96140931, 18.97420324, 11.77735963, 17.65357798, 27.75515952,
       36.4263965 ,  4.18868838, 30.33853851, 17.80396827, 10.07910482,
       20.47595652, 25.05576825, 17.54690656, 19.53079951, 21.78850844,
       24.75308208, 21.83817931, 17.32611859, 16.41852786, 21.62602186,
       18.51766648, 17.32988097, 15.94386786, 31.8342253 , 21.37664674,
       23.61547904, 21.4565251 , 10.44524028, 21.18180577, 28.23195596,
       33.20595452, 23.9200377 , 22.19725698, 22.95372273, 24.0586839 ,
       19.50742307, 42.23283927, 35.51894278, 33.127117  , 22.27585894,
       38.84205581, 23.63327501, 25.57125286, 20.00131471, 18.14985347,
       21.23335072, 12.73476562, 21.58630484, 30.41400281, 30.02957636,
       22.92224303, 28.55169364, 19.52916582, 25.48149903, 20.62294162,
       16.22960064,  8.5711739 , 30.44103734, 32.55773978,  6.86015134,
       36.69003204, 23.75377518, 14.34551562, 10.19557327, 28.261591  ,
       39.96937382, 13.87476435,  9.66392158, 21.16714611, 19.37532242,
       32.53087808, 25.03358586, 22.4091751 , 24.36419616, 17.88900902,
        8.66924919, 18.94136264, 42.19020136, 11.83621448, 14.24873601,
       38.56596368, 31.85558573, 21.12732785, 24.24038442, 12.36034763,
       31.67242202, 29.63452687, 28.06283346, 19.84477484,  6.52742424,
       18.24289471, 27.45279292, 17.9076568 , 30.26732609, 18.34453205,
       13.70734132, 23.93798477, 32.06959172, 27.00965611, 27.603045  ,
       21.99659412, 21.62571188,  9.64518085, 31.54169596, 12.29997212,
       12.71731265, 13.99176898, 34.91787959, 16.77095401, 21.61132654,
       25.03879469,  7.7654817 , 14.72206753, 30.56579364, 14.57890072,
       27.53300353, 15.11303702, 34.42714537, 19.69525524, 19.91236677,
```

```
33.99861761, 36.24514595, 19.68510586, 23.61243425, 20.36185263,
24.30000642, 22.23340604, 28.35576193, 21.01600673, 22.9822916 ,
33.9182534 , 17.73316135, 19.73534561, 24.08273044, 16.86023203,
18.38817912, 22.11103012, 18.8931337 ])
```

In [96]:
```
plt.scatter(y_test, predictions)
```

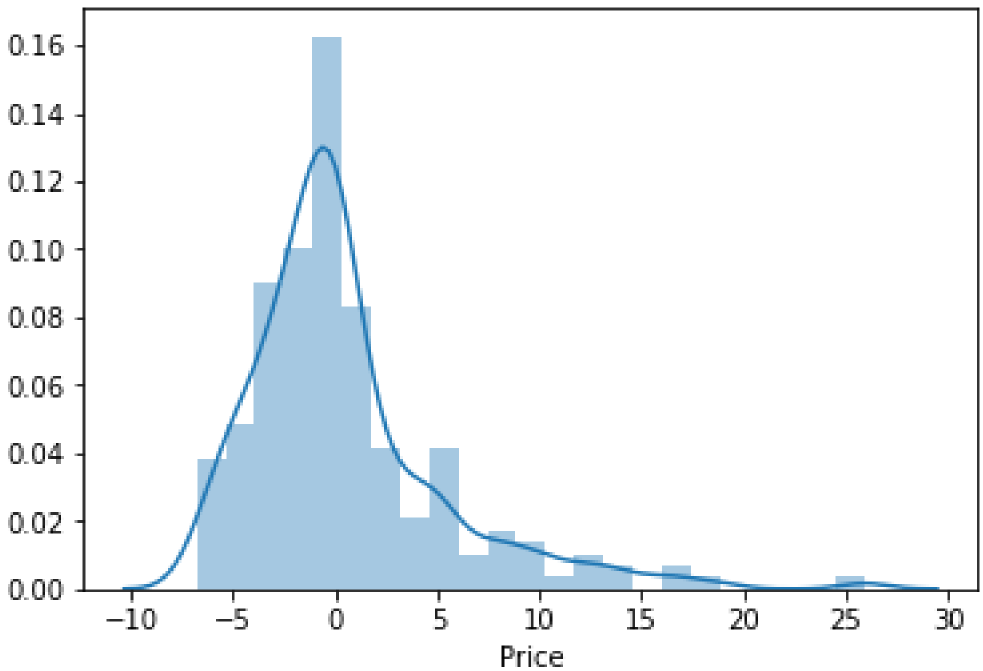
Out[96]:
```
<matplotlib.collections.PathCollection at 0x1a2415b350>
```



In [102]:
```
#After training the set
sns.distplot(y_test - predictions)
```

```
/anaconda2/lib/python2.7/site-packages/matplotlib/axes/_axes.py:6462: UserWarning: The 'normed
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[102]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a27bdadd0>
```


0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−10
−5
0
5
10
15
20
25
30
Price