

```
In [1]:
import pandas as pd
from pandas import DataFrame

import numpy as np

# visualization
import seaborn as sns
import matplotlib.pyplot as plt

# this allows plots to appear directly in the notebook
%matplotlib inline
sns.set(style='whitegrid', context='notebook')
```

Load some house value vs. crime rate data

Dataset is from Philadelphia, PA and includes average house sales price in a number of neighborhoods. The attributes of each neighborhood we have include the crime rate ('CrimeRate'), miles from Center City ('MilesPhila'), town name ('Name'), and county name ('County').

```
In [2]:
# read the housing data
sales = pd.read_csv('Philadelphia_Crime_Rate_noNA.csv')
sales.head(10)
```

```
Out[2]:
```

	HousePrice	HsPrc (\$10,000)	CrimeRate	MilesPhila	PopChg	Name	County
0	140463	14.0463	29.7	10	-1.0	Abington	Montgome
1	113033	11.3033	24.1	18	4.0	Ambler	Montgome
2	124186	12.4186	19.5	25	8.0	Aston	Delaware
3	110490	11.0490	49.4	25	2.7	Bensalem	Bucks
4	79124	7.9124	54.1	19	3.9	Bristol B.	Bucks
5	92634	9.2634	48.6	20	0.6	Bristol T.	Bucks
6	89246	8.9246	30.8	15	-2.6	Brookhaven	Delaware
7	195145	19.5145	10.8	20	-3.5	Bryn Athyn	Montgome
8	297342	29.7342	20.2	14	0.6	Bryn Mawr	Montgome
9	264298	26.4298	20.4	26	6.0	Buckingham	Bucks

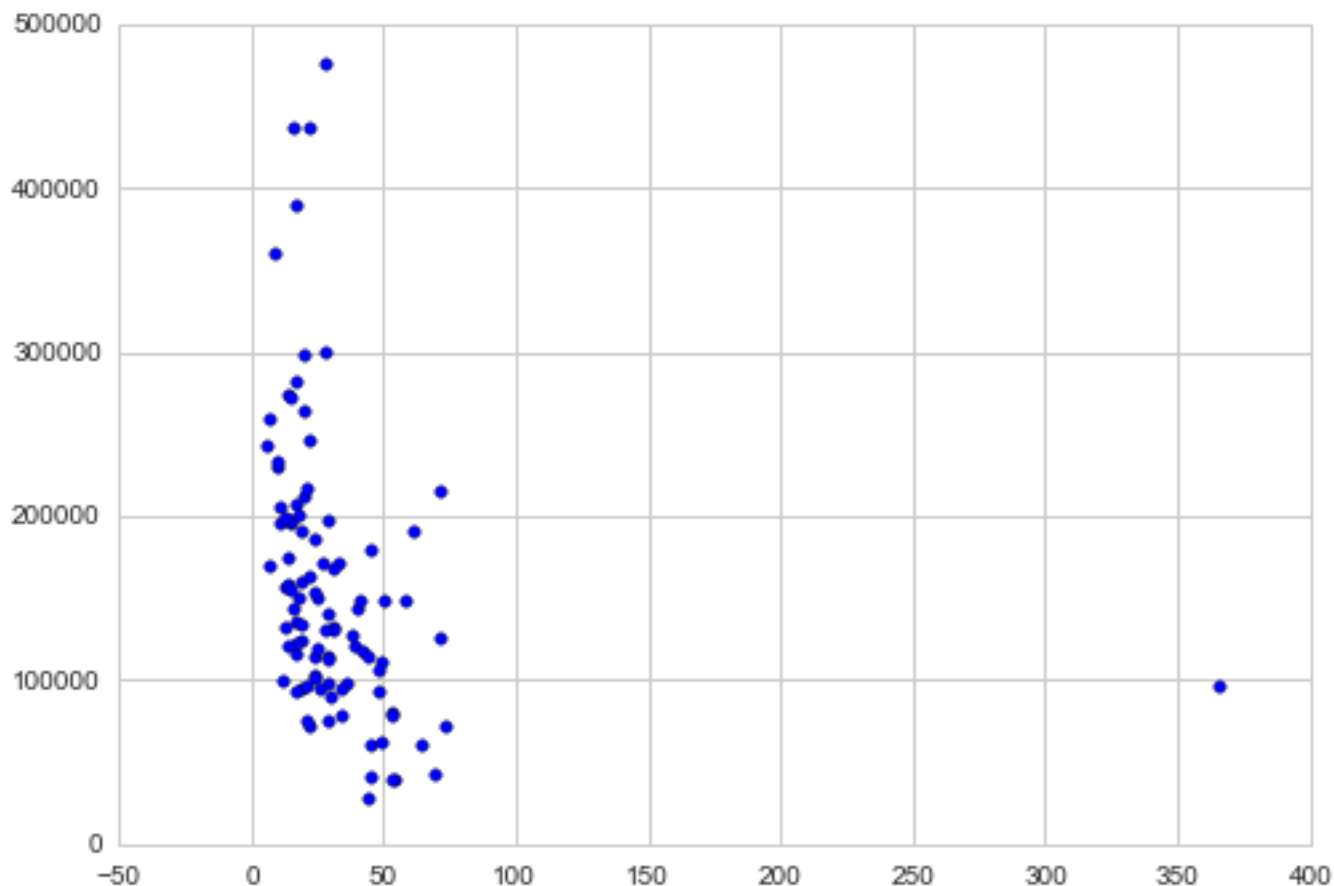
Exploring the data

The house price in a town is correlated with the crime rate of that town. Low crime towns tend to be associated with higher house prices and vice versa.

```
In [3]:
plt.figure()
plt.scatter(x=sales['CrimeRate'], y=sales['HousePrice'], marker='o',
c='b')
```

```
Out[3]:
<matplotlib.collections.PathCollection at 0x90332e8>
```

```
D:\Work\Anaconda3\lib\site-packages\matplotlib\collections.py:590:
FutureWarning: elementwise comparison failed; returning scalar instead,
but in the future will perform elementwise comparison
    if self._edgecolors == str('face'):
```



```
In [4]:
def model_coeffs_matrix(features, clf):
    coeffs_matrix = DataFrame(columns=('Feature', 'Coefficients'))
    coeffs_matrix = coeffs_matrix.append([{'Feature': 'Intercept',
'Coefficients': clf.intercept_}])

    for feature, coeff in zip(features, clf.coef_):
        coeffs_matrix = coeffs_matrix.append([{'Feature': feature,
'Coefficients':coeff}])
    coeffs_matrix.reset_index(drop=True, inplace=True)
    return coeffs_matrix
```

Fit the regression model using crime as the feature

```
In [5]:
from sklearn.linear_model import LinearRegression

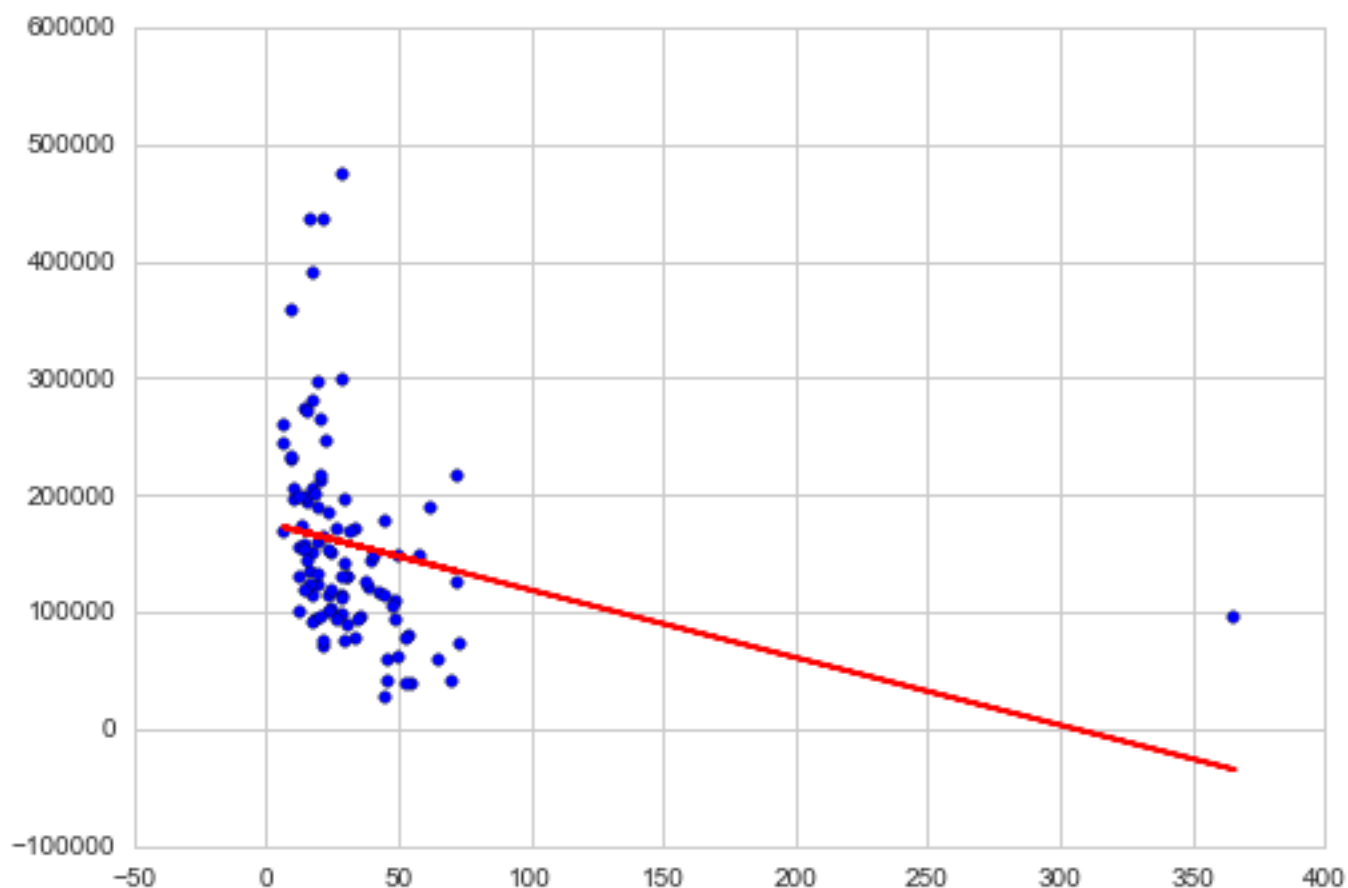
features = ['CrimeRate']
In [6]:
crime_model = LinearRegression()
crime_model.fit(sales[features], sales['HousePrice'])
Out[6]:
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1,
normalize=False)
In [7]:
y_pred = crime_model.predict(sales[features])
```

Let's see what our fit looks like

```
In [8]:  
plt.figure()  
plt.scatter(sales[features], sales['HousePrice'], c='b', marker='o')  
plt.plot(sales[features], y_pred, color='red')
```

```
Out[8]:  
[<matplotlib.lines.Line2D at 0xa4b9dd8>]
```

```
D:\Work\Anaconda3\lib\site-packages\matplotlib\collections.py:590:  
FutureWarning: elementwise comparison failed; returning scalar instead,  
but in the future will perform elementwise comparison  
    if self._edgecolors == str('face'):
```



Above: blue dots are original data, green line is the fit from the simple regression.

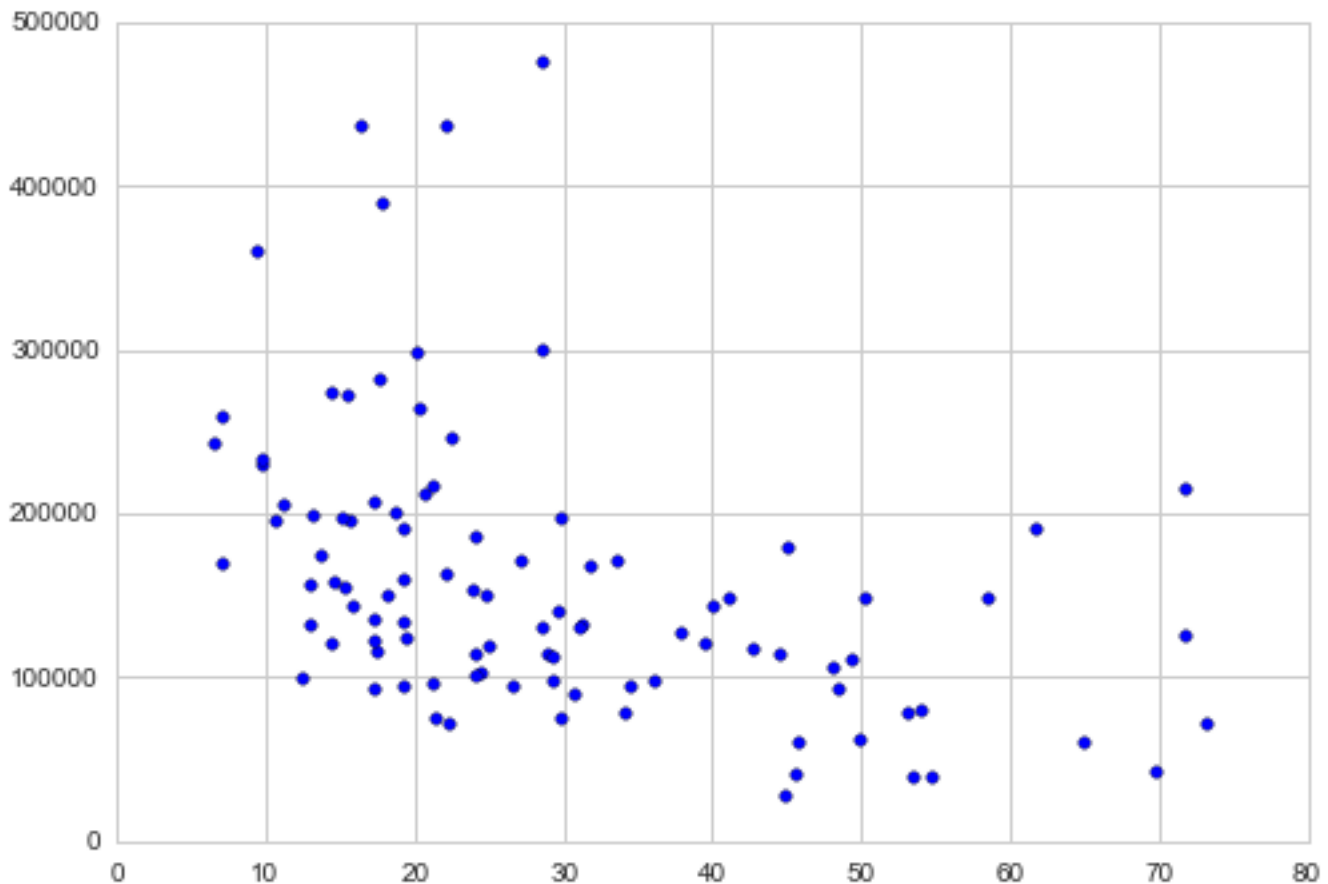
Remove Center City and redo the analysis

Center City is the one observation with an extremely high crime rate, yet house prices are not very low. This point does not follow the trend of the rest of the data very well. A question is how much including Center City is influencing our fit on the other datapoints. Let's remove this datapoint and see what happens.

```
In [9]:  
sales_noCC = sales[sales['MilesPhila'] != 0.0]  
In [10]:
```

```
plt.figure()
plt.scatter(x=sales_noCC[features], y=sales_noCC['HousePrice'],
marker='o', c='b')
Out[10]:
<matplotlib.collections.PathCollection at 0xa53e278>
```

D:\Work\Anaconda3\lib\site-packages\matplotlib\collections.py:590:
FutureWarning: elementwise comparison failed; returning scalar instead,
but in the future will perform elementwise comparison
if self._edgecolors == str('face'):



Refit our simple regression model on this modified dataset:

```
In [11]:
crime_model_noCC = LinearRegression()
crime_model_noCC.fit(sales_noCC[features], sales_noCC['HousePrice'])
Out[11]:
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1,
normalize=False)
In [12]:
y_pred = crime_model_noCC.predict(sales_noCC[features])
```

Look at the fit:

```
In [13]:
plt.figure()
plt.scatter(sales_noCC[features], sales_noCC['HousePrice'], c='b',
marker='o')
```

```
plt.plot(sales_noCC[features], y_pred, color='red')
```

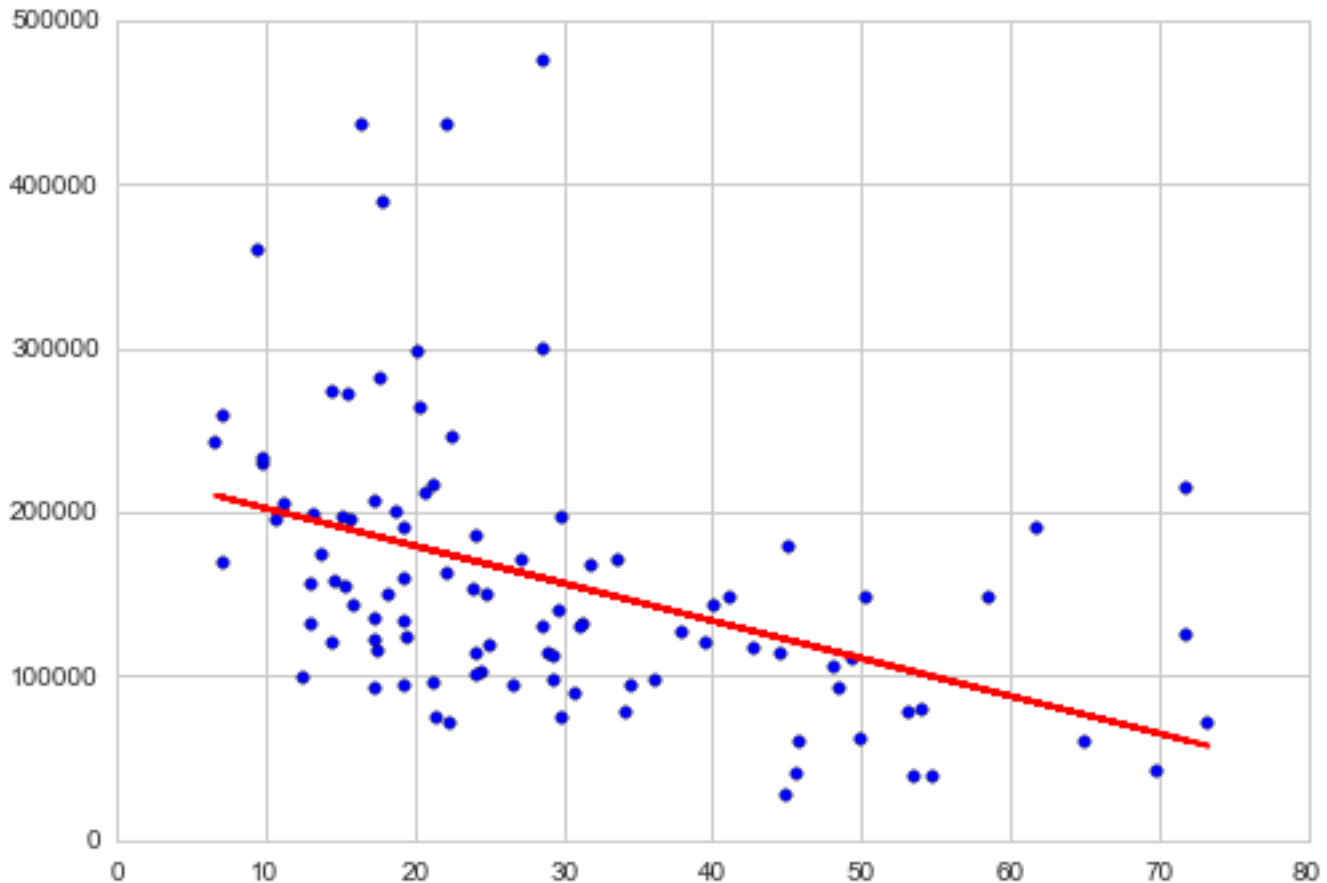
```
Out[13]:
```

```
[<matplotlib.lines.Line2D at 0xa5a3908>]
```

```
D:\Work\Anaconda3\lib\site-packages\matplotlib\collections.py:590:
```

```
FutureWarning: elementwise comparison failed; returning scalar instead,  
but in the future will perform elementwise comparison
```

```
if self._edgecolors == str('face'):
```



Compare coefficients for full-data fit versus no-Center-City fit

Visually, the fit seems different, but let's quantify this by examining the estimated coefficients of our original fit and that of the modified dataset with Center City removed.

```
In [14]:
```

```
model_coeffs_matrix(features, crime_model)
```

```
Out[14]:
```

	Feature	Coefficient
0	Intercept	176629.408107
1	CrimeRate	-576.908128

```
In [15]:
```

```
model_coeffs_matrix(features, crime_model_noCC)
```

```
Out[15]:
```

	Feature	Coefficient
0	Intercept	225233.551839
1	CrimeRate	-2288.689430

Above: We see that for the "no Center City" version, per unit increase in crime, the predicted decrease in house prices is 2,287. In contrast, for the original dataset, the drop is only 576 per unit increase in crime. This is significantly different!

High leverage points:

Center City is said to be a "high leverage" point because it is at an extreme x value where there are not other observations. As a result, recalling the closed-form solution for simple regression, this point has the *potential* to dramatically change the least squares line since the center of x mass is heavily influenced by this one point and the least squares line will try to fit close to that outlying (in x) point. If a high leverage point follows the trend of the other data, this might not have much effect. On the other hand, if this point somehow differs, it can be strongly influential in the resulting fit.

Influential observations:

An influential observation is one where the removal of the point significantly changes the fit. As discussed above, high leverage points are good candidates for being influential observations, but need not be. Other observations that are *not* leverage points can also be influential observations (e.g., strongly outlying in y even if x is a typical value).

Remove high-value outlier neighborhoods and redo analysis

Based on the discussion above, a question is whether the outlying high-value towns are strongly influencing the fit. Let's remove them and see what happens.

In [16]:

```
sales_nohighend = sales_noCC[sales_noCC['HousePrice'] < 350000]
crime_model_nohighend = LinearRegression()
crime_model_nohighend.fit(sales_nohighend[features],
sales_nohighend['HousePrice'])
```

Out[16]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1,
normalize=False)
```

Do the coefficients change much?

In [17]:

```
model_coeffs_matrix(features, crime_model_noCC)
```

Out[17]:

	Feature	Coefficients
0	Intercept	225233.551839
1	CrimeRate	-2288.689430

In [18]:

```
model_coeffs_matrix(features, crime_model_nohighend)
```

Out[18]:

	Feature	Coefficients
0	Intercept	199098.852670
1	CrimeRate	-1838.562649

Above: We see that removing the outlying high-value neighborhoods has *some* effect on the fit, but not nearly as much as our high-leverage Center City datapoint.