Predicting house prices

August 30, 2017

```
In [ ]: Aluno: Marco Antonio Santo
```

1 Fire up libraries

```
In [1]: import matplotlib
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        %matplotlib inline
        from sklearn import linear_model
```

2 Load some house sales data

Dataset is from house sales in King County, the region where the city of Seattle, WA is located.

```
In [2]: sales = pd.read_csv('home_data.csv')

In [3]: sales.head()

Out[3]:            id             date   price  bedrooms  bathrooms  sqft_living  \
        0  7129300520  20141013T000000  221900         3       1.00         1180
        1  6414100192  20141209T000000  538000         3       2.25         2570
        2  5631500400  20150225T000000  180000         2       1.00          770
        3  2487200875  20141209T000000  604000         4       3.00         1960
        4  1954400510  20150218T000000  510000         3       2.00         1680

           sqft_lot  floors  waterfront  view     ...      grade  sqft_above  \
        0      5650     1.0           0     0     ...          7        1180
        1      7242     2.0           0     0     ...          7        2170
        2     10000     1.0           0     0     ...          6         770
        3      5000     1.0           0     0     ...          7        1050
        4      8080     1.0           0     0     ...          8        1680

           sqft_basement  yr_built  yr_renovated  zipcode      lat     long  \
        0              0      1955             0    98178  47.5112 -122.257
        1            400      1951          1991    98125  47.7210 -122.319
```

```
2             0      1933             0     98028  47.7379 -122.233
3           910      1965             0     98136  47.5208 -122.393
4             0      1987             0     98074  47.6168 -122.045

   sqft_living15  sqft_lot15
0           1340        5650
1           1690        7639
2           2720        8062
3           1360        5000
4           1800        7503

[5 rows x 21 columns]
```

In [4]:
```
sales[sales['id']==1839920160]
```

Out[4]:
```
               id             date   price  bedrooms  bathrooms  sqft_living  \
11860  1839920160  20140714T000000  432000         3        2.0         1870

       sqft_lot  floors  waterfront  view     ...      grade  sqft_above  \
11860      7080     1.0           0     0     ...          7        1210

       sqft_basement  yr_built  yr_renovated  zipcode      lat     long  \
11860            660      1969             0    98034  47.7244 -122.179

       sqft_living15  sqft_lot15
11860           1620        8000

[1 rows x 21 columns]
```

In [5]:
```
sales.keys()
```

Out[5]:
```
Index([u'id', u'date', u'price', u'bedrooms', u'bathrooms', u'sqft_living',
       u'sqft_lot', u'floors', u'waterfront', u'view', u'condition', u'grade',
       u'sqft_above', u'sqft_basement', u'yr_built', u'yr_renovated',
       u'zipcode', u'lat', u'long', u'sqft_living15', u'sqft_lot15'],
      dtype='object')
```

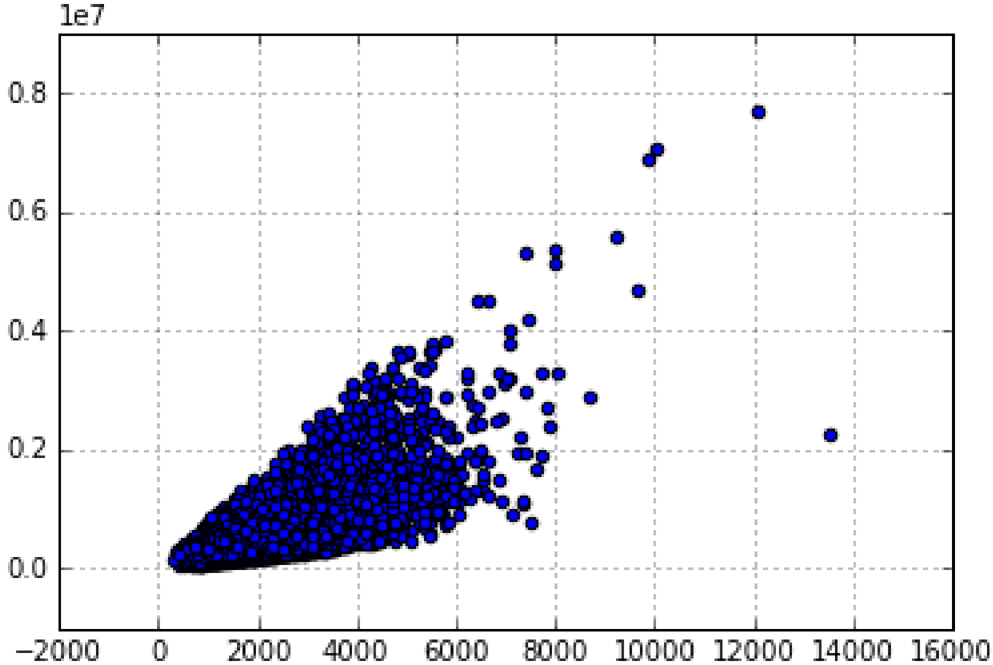
In [6]:
```
sales.shape
```

Out[6]:
```
(21613, 21)
```

3 Exploring the data for housing sales

The house price is correlated with the number of square feet of living space.

In [7]:
```
plt.grid('on')
plt.scatter(sales['sqft_living'], sales['price'])
plt.show()
```



4 Create a simple regression model of sqft_living to price

Split data into training and testing.
We use random_state=200 so that everyone running this notebook gets the same results. In practice, you may set a random seed.

In [8]:
```
train_data = sales.sample(frac=0.8, random_state=200)
test_data  = sales.drop(train_data.index)
print(train_data.shape, test_data.shape)
```

```
((17290, 21), (4323, 21))
```

In [9]:
```
train_data.head()
```

Out[9]:
```
               id             date   price  bedrooms  bathrooms  sqft_living  \
11860  1839920160  20140714T000000  432000         3       2.00         1870
12446  6705850140  20141009T000000  750000         4       2.75         3170
10556   924069190  20140819T000000  440000         3       1.75         2000
4828   3211270170  20140523T000000  404000         4       3.00         4060
3502   9523103001  20141013T000000  389000         2       1.00          850

       sqft_lot  floors  waterfront  view     ...      grade  sqft_above  \
11860      7080     1.0           0     0     ...          7        1210
12446      7634     2.0           0     0     ...         10        3170
```

```
10556     11880     2.0            0      0      ...          8         2000
4828      35621     1.0            0      0      ...          9         2030
3502       3276     1.0            0      0      ...          6          850

       sqft_basement  yr_built  yr_renovated  zipcode      lat     long  \
11860            660      1969             0    98034  47.7244 -122.179
12446              0      1992             0    98075  47.5774 -122.054
10556              0      1979             0    98075  47.5882 -122.052
4828            2030      1989             0    98092  47.3059 -122.108
3502               0      1910             0    98103  47.6742 -122.350

       sqft_living15  sqft_lot15
11860           1620        8000
12446           2940        7846
10556           1820       15120
4828            2950       35259
3502            1460        4100

[5 rows x 21 columns]
```

In [10]:
```python
test_data.head()
```

Out[10]:
```
            id             date    price  bedrooms  bathrooms  sqft_living  \
3   2487200875  20141209T000000   604000         4        3.0         1960
4   1954400510  20150218T000000   510000         3        2.0         1680
5   7237550310  20140512T000000  1225000         4        4.5         5420
17  6865200140  20140529T000000   485000         4        1.0         1600
18    16000397  20141205T000000   189000         2        1.0         1200

    sqft_lot  floors  waterfront  view     ...      grade  sqft_above  \
3       5000     1.0           0     0     ...          7        1050
4       8080     1.0           0     0     ...          8        1680
5     101930     1.0           0     0     ...         11        3890
17      4300     1.5           0     0     ...          7        1600
18      9850     1.0           0     0     ...          7        1200

    sqft_basement  yr_built  yr_renovated  zipcode      lat     long  \
3             910      1965             0    98136  47.5208 -122.393
4               0      1987             0    98074  47.6168 -122.045
5            1530      2001             0    98053  47.6561 -122.005
17              0      1916             0    98103  47.6648 -122.343
18              0      1921             0    98002  47.3089 -122.210

    sqft_living15  sqft_lot15
3            1360        5000
4            1800        7503
5            4760      101930
17           1610        4300
```

```
18              1060         5095

[5 rows x 21 columns]
```

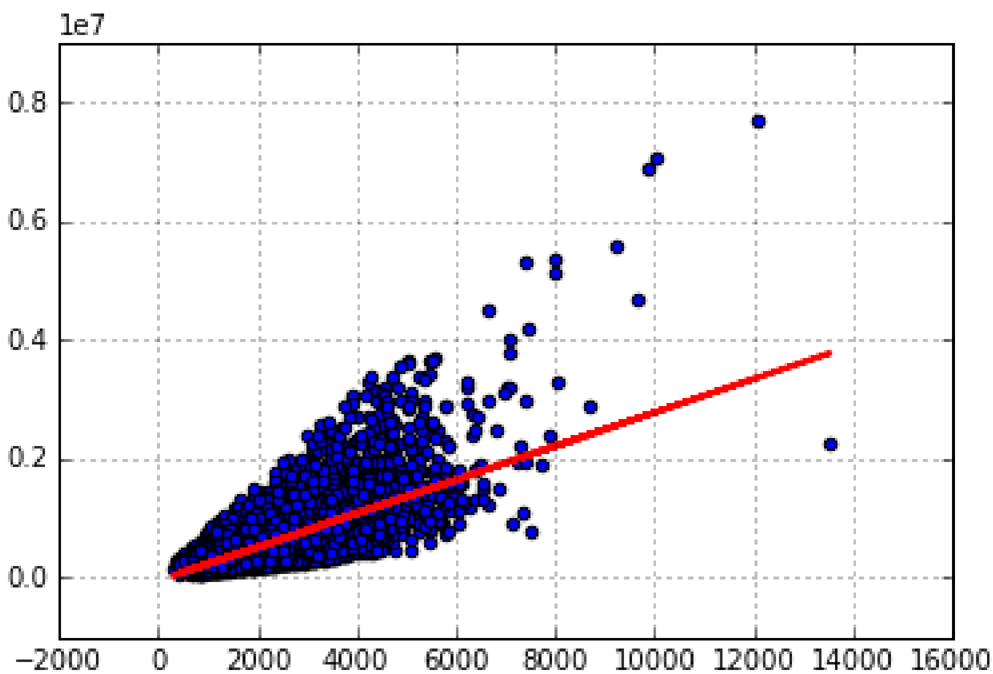
4.1 Build the regression model using only sqft_living as a feature

In [11]:
```
x_train = train_data['sqft_living'].values.reshape(-1,1)
y_train = train_data['price'].values.reshape(-1,1)
```

In [12]:
```
simple_model = linear_model.LinearRegression()
simple_model.fit(x_train, y_train)
```

Out[12]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [13]:
```
plt.grid('on')
plt.scatter(x_train, y_train)
plt.plot(x_train, simple_model.predict(x_train), color='red', linewidth=2)
plt.show()
```

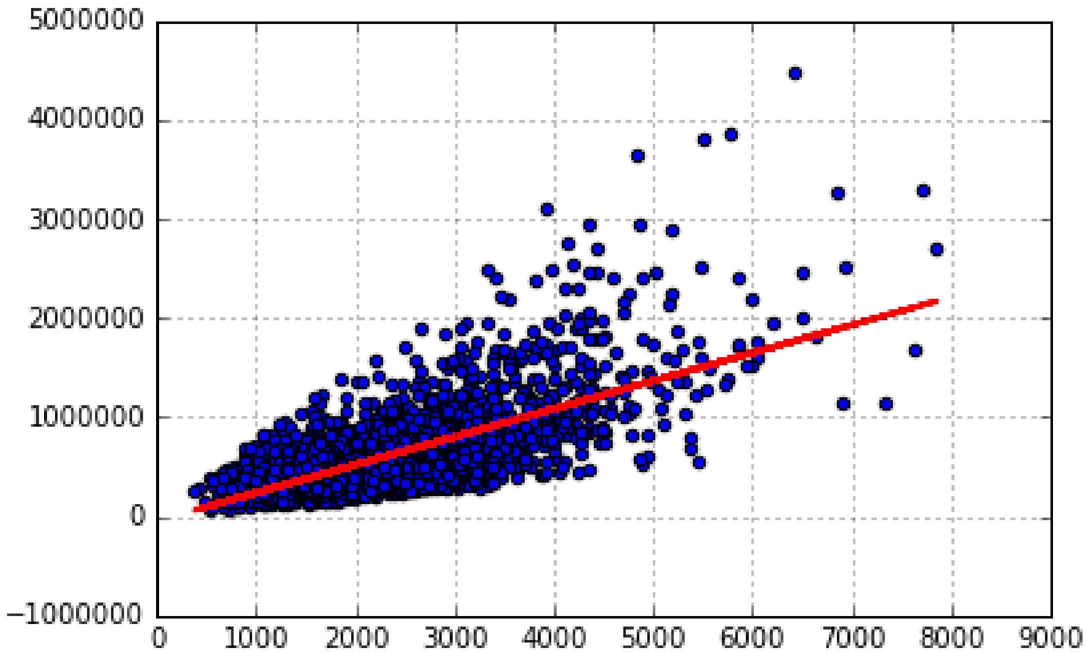


5 Let's show what our predictions look like

In [14]:
```
x_test = test_data['sqft_living'].values.reshape(-1,1)
y_test = test_data['price'].values.reshape(-1,1)

y_pred = simple_model.predict(x_test)
```

```
In [15]: plt.grid('on')
         plt.scatter(x_test, y_test)
         plt.plot(x_test,y_pred, color='red', linewidth=2)
         plt.show()
```



6 Evaluate the simple model

```
In [16]: def rmse(predictions, targets):
             return np.sqrt(((predictions - targets) ** 2).mean())
```

```
In [17]: print('intercept:', simple_model.intercept_, 'coefficients:', simple_model.coef_)
         # The mean squared error
         print("RMSE: %.2f" % (rmse(y_pred, y_test)))
```

```
('intercept:', array([-46493.04519733]), 'coefficients:', array([[ 282.27187583]]))
RMSE: 254323.39
```

RMSE of about $254.323,39

7 Explore other features in the data

To build a more elaborate model, we will explore using more features.

```
In [18]: my_features = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'zipcode']
```

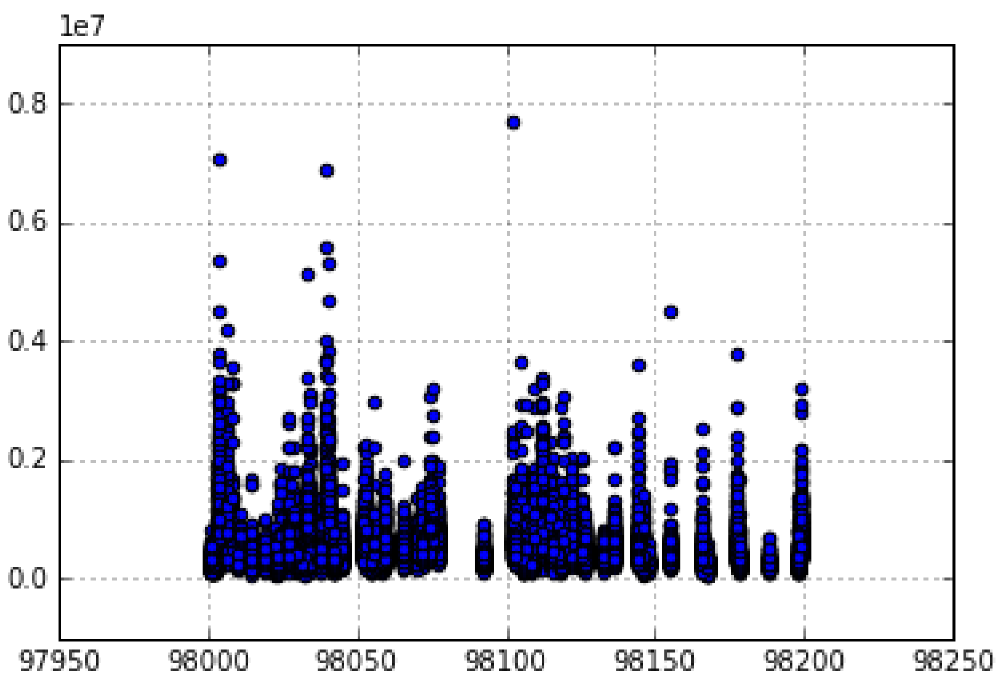
In [19]:
```
sales[my_features].describe()
```

Out[19]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors
count	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000
mean	3.370842	2.114757	2079.899736	1.510697e+04	1.494309
std	0.930062	0.770163	918.440897	4.142051e+04	0.539989
min	0.000000	0.000000	290.000000	5.200000e+02	1.000000
25%	3.000000	1.750000	1427.000000	5.040000e+03	1.000000
50%	3.000000	2.250000	1910.000000	7.618000e+03	1.500000
75%	4.000000	2.500000	2550.000000	1.068800e+04	2.000000
max	33.000000	8.000000	13540.000000	1.651359e+06	3.500000

	zipcode
count	21613.000000
mean	98077.939805
std	53.505026
min	98001.000000
25%	98033.000000
50%	98065.000000
75%	98118.000000
max	98199.000000

In [20]:
```
#sales.show(view='BoxWhisker Plot', x='zipcode', y='price')
plt.grid('on')
plt.scatter(sales['zipcode'], sales['price'])
plt.show()
```



98039 is the most expensive zip code.

8 Build a regression model with more features

In [21]:
```
#my_features_model = (train_data,target='price',features=my_features,validation_set=Non
x_train = train_data[my_features].values.reshape(-1,len(my_features))
y_train = train_data['price'].values.reshape(-1,1)
```

In [22]:
```
mult_model = linear_model.LinearRegression()
mult_model.fit(x_train, y_train)
```

Out[22]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

8.1 Comparing the results of the simple model with adding more features

In [23]:
```
x_test = test_data[my_features].values.reshape(-1,len(my_features))
y_test = test_data['price'].values.reshape(-1,1)

y_pred = mult_model.predict(x_test)
```

In [24]:
```
print('intercept:', mult_model.intercept_, 'coefficients:', mult_model.coef_)
# The mean squared error
print("RMSE: %.2f" % (rmse(y_pred, y_test)))
```

```
('intercept:', array([-56348418.9495597]), 'coefficients:', array([[ -6.17445455e+04,   1.880533
        -2.88006880e-01,  -9.68870610e+03,   5.75218487e+02]]))
RMSE: 249311.90
```

The RMSE goes down from $254.323,39 to $228.024,43 with more features.

9 Apply learned models to predict prices of 3 houses

The first house we will use is considered an "average" house in Seattle.

In [25]:
```
house1 = sales[sales['id']==5309101200]
```

In [26]:
```
house1
```

Out[26]:
```
              id             date   price  bedrooms  bathrooms  sqft_living  \
1054  5309101200  20140605T000000  620000         4       2.25         2400

      sqft_lot  floors  waterfront  view     ...      grade  sqft_above  \
1054      5350     1.5           0     0     ...          7        1460

      sqft_basement  yr_built  yr_renovated  zipcode      lat     long  \
1054            940      1929             0    98117  47.6763 -122.37
```

```
      sqft_living15  sqft_lot15
1054           1250        4880

[1 rows x 21 columns]
```

```
In [27]: print house1['price']
```

```
1054    620000
Name: price, dtype: int64
```

```
In [28]: print simple_model.predict(house1['sqft_living'])
```

```
[[ 630959.4568039]]
```

```
/home/aluno/anaconda3/envs/gl-env/lib/python2.7/site-packages/sklearn/utils/validation.py:386: D
  DeprecationWarning)
```

```
In [29]: print mult_model.predict(house1[my_features])
```

```
[[ 630924.33807747]]
```

In this case, the model with more features provides a worse prediction than the simpler model with only 1 feature. However, on average, the model with more features is better.

9.1 Prediction for a second, fancier house

We will now examine the predictions for a fancier house.

```
In [30]: house2 = sales[sales['id']==1925069082]
```

```
In [31]: house2
```

```
Out[31]:              id             date    price  bedrooms  bathrooms  sqft_living  \
         1361  1925069082  20150511T000000  2200000         5       4.25         4640

               sqft_lot  floors  waterfront  view     ...      grade  sqft_above  \
         1361     22703     2.0           1     4     ...          8        2860

               sqft_basement  yr_built  yr_renovated  zipcode      lat     long  \
         1361           1780      1952             0    98052  47.6393 -122.097

               sqft_living15  sqft_lot15
         1361           3140       14200

         [1 rows x 21 columns]
```

```
In [32]: print house2['price']
```

```
1361    2200000
Name: price, dtype: int64
```

```
In [33]: print simple_model.predict(house2['sqft_living'].values.reshape(-1,1))
```

```
[[ 1263248.45867172]]
```

```
In [34]: print mult_model.predict(house2[my_features])
```

```
[[ 1270172.16078085]]
```

In this case, the model with more features provides a better prediction. This behavior is expected here, because this house is more differentiated by features that go beyond its square feet of living space, especially the fact that it's a waterfront house.

9.2 Last house, super fancy

Our last house is a very large one owned by a famous Seattleite.

```
In [35]: bill_gates = {'bedrooms':[8],
                       'bathrooms':[25],
                       'sqft_living':[50000],
                       'sqft_lot':[225000],
                       'floors':[4],
                       'zipcode':['98039'],
                       'condition':[10],
                       'grade':[10],
                       'waterfront':[1],
                       'view':[4],
                       'sqft_above':[37500],
                       'sqft_basement':[12500],
                       'yr_built':[1994],
                       'yr_renovated':[2010],
                       'lat':[47.627606],
                       'long':[-122.242054],
                       'sqft_living15':[5000],
                       'sqft_lot15':[40000]}
```

```
In [36]: print simple_model.predict(pd.DataFrame(bill_gates)['sqft_living'].values.reshape(-1,1)
```

```
[[ 14067100.74649496]]
```

The model predicts a price of over $14M for this house! But we expect the house to cost much more. (There are very few samples in the dataset of houses that are this fancy, so we don't expect the model to capture a perfect prediction here.)

```
In [37]: print mult_model.predict(pd.DataFrame(bill_gates)[my_features])
```

```
[[ 15779944.98847022]]
```

10 Answers

11 1 - Selection and summary statistics

```
In [38]: sales[sales['zipcode']==98039]
```

Out[38]:

	id	date	price	bedrooms	bathrooms	sqft_living \
2974	3625049014	20140829T000000	2950000	4	3.50	4860
3761	2540700110	20150212T000000	1905000	4	3.50	4210
4077	3262300940	20141107T000000	875000	3	1.00	1220
4078	3262300940	20150210T000000	940000	3	1.00	1220
4149	6447300265	20141014T000000	4000000	4	5.50	7080
4411	2470100110	20140804T000000	5570000	5	5.75	9200
4791	2210500019	20150324T000000	937500	3	1.00	1320
5178	6447300345	20150406T000000	1160000	4	3.00	2680
5589	6447300225	20141106T000000	1880000	3	2.75	2620
5880	2525049148	20141007T000000	3418800	5	5.00	5450
6868	3262300235	20141126T000000	1555000	5	2.50	2870
7501	2525049133	20150402T000000	1398000	5	2.25	2640
8241	3262301355	20140725T000000	1320000	3	2.75	2680
9254	9208900037	20140919T000000	6885000	6	7.75	9890
9694	3262301610	20141118T000000	865000	3	1.50	1530
9809	5426300060	20141008T000000	1000000	3	2.25	2300
11278	3025300226	20140515T000000	2100000	4	1.75	3550
11952	2260300060	20150410T000000	2575000	5	3.00	4780
12295	3738000070	20150309T000000	1712750	5	2.50	2660
12811	5425700150	20140804T000000	787500	4	1.75	1580
13235	3262300322	20150408T000000	1651000	4	3.25	3640
13267	5425700205	20140520T000000	1800000	4	3.50	4460
13419	2525049246	20141017T000000	1550000	2	2.25	2950
13621	2525049266	20140821T000000	1762000	3	2.25	3060
13988	5427110040	20140609T000000	1225000	4	2.50	2740
14052	7397300220	20140529T000000	2750000	4	3.25	4430
14254	2425049107	20150305T000000	1950000	4	3.75	4150
14385	2425049061	20140825T000000	2200000	3	2.00	3570
14803	3835502815	20140925T000000	1260000	3	2.50	3110
15022	2210500010	20140930T000000	2450000	7	4.25	4670
15255	2425049063	20140911T000000	3640900	4	3.25	4830
15632	3625049088	20140702T000000	2271150	4	3.25	4040
16268	3025300250	20150513T000000	1620000	4	2.25	2350
16302	7397300170	20140530T000000	3710000	4	3.50	5550
16377	3262300920	20150408T000000	1200000	4	3.00	2150

```
16825  3025300095  20141009T000000  2500000         4       4.50          4300
17001  2525049259  20140812T000000  2187730         4       4.50          4240
17209  3025300225  20141031T000000  1450000         5       2.75          3090
17230  2470200020  20140514T000000  1880000         4       2.75          3260
17899  3262300555  20140708T000000  2458000         4       5.25          6500
17930  3625049079  20140801T000000  1350000         3       2.00          2070
18793  2525049263  20140709T000000  2680000         5       3.00          4290
18892  5427100150  20140626T000000  1410000         4       2.25          3250
18912  2425049066  20140616T000000  1920000         4       2.50          3070
19148  3625049042  20141011T000000  3635000         5       6.00          5490
19236  2525049086  20141003T000000  2720000         4       3.25          3990
19351  2525049113  20140725T000000  1950000         4       3.50          4065
20096  3262300485  20150421T000000  2250000         5       5.25          3410
21040  6447300365  20141113T000000  2900000         5       4.00          5190
21514  3262300818  20150227T000000  1865000         4       3.75          3790

       sqft_lot  floors  waterfront  view     ...      grade  sqft_above  \
2974      23885     2.0           0     0     ...         12        4860
3761      18564     2.0           0     0     ...         11        4210
4077       8119     1.0           0     0     ...          7        1220
4078       8119     1.0           0     0     ...          7        1220
4149      16573     2.0           0     0     ...         12        5760
4411      35069     2.0           0     0     ...         13        6200
4791       8500     1.0           0     0     ...          7        1320
5178      15438     2.0           0     2     ...          8        2680
5589      17919     1.0           0     1     ...          9        2620
5880      20412     2.0           0     0     ...         11        5450
6868      16238     2.0           0     0     ...          8        2870
7501      14959     1.0           0     0     ...          7        1770
8241      20104     1.0           0     0     ...          9        1820
9254      31374     2.0           0     4     ...         13        8860
9694      10827     1.0           0     0     ...          8        1530
9809      15952     1.0           0     0     ...          8        1150
11278     19865     2.0           0     0     ...          9        3550
11952     20440     1.0           0     0     ...         10        3660
12295      6572     1.0           0     0     ...          9        1960
12811      9382     1.0           0     0     ...          7        1080
13235     13530     1.0           0     0     ...          9        2570
13267     16953     1.0           0     0     ...          9        2550
13419     15593     1.0           0     0     ...          8        1560
13621     16000     2.0           0     0     ...         10        3060
13988     16007     2.0           0     0     ...          9        2740
14052     21000     2.0           0     0     ...         10        4430
14254     17424     1.0           0     0     ...          9        3130
14385     30456     1.0           0     1     ...          8        2070
14803      9930     1.0           0     1     ...          8        1640
15022     23115     2.0           0     2     ...         11        4670
15255     22257     2.0           1     4     ...         11        4830
```

```
15632   18916     1.0           0     0    ...          9          4040
16268   17709     2.0           0     0    ...          9          2350
16302   28078     2.0           0     2    ...         12          3350
16377    8119     2.0           0     0    ...          8          2150
16825   19844     2.0           0     0    ...         11          4300
17001   13162     2.0           0     0    ...         10          4240
17209   19865     1.0           0     0    ...          9          3090
17230   19542     1.0           0     0    ...         10          2170
17899   14986     2.0           0     0    ...         11          5180
17930    9600     1.0           0     1    ...          7          1590
18793   20445     2.0           0     0    ...         11          4290
18892   16684     2.0           0     0    ...          9          3250
18912   34412     1.0           0     3    ...          9          2070
19148   19897     2.0           0     0    ...         12          5490
19236   18115     2.0           0     0    ...         11          3990
19351   18713     2.0           0     0    ...         10          4065
20096    8118     2.0           0     0    ...         11          3410
21040   14600     2.0           0     1    ...         11          5190
21514    8797     2.0           0     0    ...         11          3290

       sqft_basement  yr_built  yr_renovated  zipcode      lat     long  \
2974               0      1996             0    98039  47.6172 -122.230
3761               0      2001             0    98039  47.6206 -122.225
4077               0      1955             0    98039  47.6328 -122.236
4078               0      1955             0    98039  47.6328 -122.236
4149            1320      2008             0    98039  47.6151 -122.224
4411            3000      2001             0    98039  47.6289 -122.233
4791               0      1954             0    98039  47.6187 -122.226
5178               0      1902          1956    98039  47.6109 -122.226
5589               0      1949             0    98039  47.6144 -122.228
5880               0      2014             0    98039  47.6209 -122.237
6868               0      1962             0    98039  47.6308 -122.238
7501             870      1929             0    98039  47.6191 -122.234
8241             860      1964             0    98039  47.6304 -122.234
9254            1030      2001             0    98039  47.6305 -122.240
9694               0      1955             0    98039  47.6354 -122.234
9809            1150      1963             0    98039  47.6322 -122.232
11278              0      1962          2002    98039  47.6236 -122.235
11952           1120      1975             0    98039  47.6242 -122.239
12295            700      1959             0    98039  47.6176 -122.223
12811            500      1963             0    98039  47.6353 -122.232
13235           1070      1924          2000    98039  47.6293 -122.238
13267           1910      1962          1994    98039  47.6338 -122.232
13419           1390      1942          1986    98039  47.6209 -122.236
13621              0      1988             0    98039  47.6189 -122.230
13988              0      1984             0    98039  47.6353 -122.229
14052              0      1952          2007    98039  47.6398 -122.237
14254           1020      1963          2000    98039  47.6390 -122.236
```

14385	1500	1946	1982	98039	47.6413	-122.240
14803	1470	1954	0	98039	47.6112	-122.226
15022	0	1992	0	98039	47.6183	-122.227
15255	0	1990	0	98039	47.6409	-122.241
15632	0	1954	0	98039	47.6155	-122.238
16268	0	1977	0	98039	47.6232	-122.236
16302	2200	2000	0	98039	47.6395	-122.234
16377	0	1953	2004	98039	47.6335	-122.236
16825	0	1985	1999	98039	47.6218	-122.237
17001	0	2004	0	98039	47.6193	-122.229
17209	0	1953	0	98039	47.6232	-122.235
17230	1090	1968	0	98039	47.6245	-122.236
17899	1320	2001	0	98039	47.6304	-122.236
17930	480	1946	0	98039	47.6160	-122.239
18793	0	1985	0	98039	47.6217	-122.239
18892	0	1979	0	98039	47.6334	-122.229
18912	1000	1950	0	98039	47.6400	-122.240
19148	0	2005	0	98039	47.6165	-122.236
19236	0	1989	0	98039	47.6177	-122.229
19351	0	1987	0	98039	47.6209	-122.237
20096	0	2006	0	98039	47.6295	-122.236
21040	0	2013	0	98039	47.6102	-122.225
21514	500	2006	0	98039	47.6351	-122.236

	sqft_living15	sqft_lot15
2974	3580	16054
3761	3520	18564
4077	1910	8119
4078	1910	8119
4149	3140	15996
4411	3560	24345
4791	2790	10800
5178	4480	14406
5589	3400	14400
5880	3160	17825
6868	2870	16238
7501	3240	17904
8241	3060	19837
9254	4540	42730
9694	2050	10827
9809	2200	14284
11278	3000	19862
11952	4660	20440
12295	3960	14595
12811	2010	9382
13235	2760	15000
13267	1980	13370
13419	2060	19855

```
13621           3510      13162
13988           2760      16008
14052           3930      20000
14254           3930      21420
14385           3570      27418
14803           3650      14399
15022           3240      13912
15255           3820      25582
15632           3000      18831
16268           3360      19855
16302           2980      19602
16377           1590       8119
16825           3070      19845
17001           3010      12163
17209           2970      19862
17230           3480      19863
17899           2270       8119
17930           3000      16215
18793           3620      22325
18892           2890      16927
18912           3780      27940
19148           2910      17600
19236           3450      16087
19351           3070      18713
20096           3410      16236
21040           3840      19250
21514           2660      12150

[50 rows x 21 columns]
```

In [39]:
```
sales[sales['zipcode']==98039]['price'].mean()
```

Out[39]:
```
2160606.6
```

12 2 - Filtering data

In [40]:
```
num = len(sales[(sales['sqft_living']>2000) & (sales['sqft_living']<4000)])
num
```

Out[40]:
```
9111
```

In [41]:
```
num/float(len(sales))
```

Out[41]:
```
0.4215518437977143
```

13 3 - Building a regression model with several more features

In [42]:
```
advanced_features = ['bedrooms','bathrooms','sqft_living','sqft_lot','floors','zipcode'
                     'waterfront','view','sqft_above','sqft_basement','yr_built','yr_renovate
```

```
                  'sqft_living15','sqft_lot15']
x_train2 = train_data[advanced_features].values.reshape(-1,len(advanced_features))
y_train2 = train_data['price'].values.reshape(-1,1)
```

In [43]:
```
mult_model = linear_model.LinearRegression()
mult_model.fit(x_train2, y_train2)
```

Out[43]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [44]:
```
x_test2 = test_data[advanced_features].values.reshape(-1,len(advanced_features))
y_test2 = test_data['price'].values.reshape(-1,1)

y_pred2 = mult_model.predict(x_test2)
```

In [45]:
```
print('intercept:', simple_model.intercept_, 'coefficients:', simple_model.coef_)
# The mean squared error
print("RMSE: %.2f" % (rmse(y_pred2, y_test2)))
```

```
('intercept:', array([-46493.04519733]), 'coefficients:', array([[ 282.27187583]]))
RMSE: 193713.58
```

In [46]:
```
print('intercept:', mult_model.intercept_, 'coefficients:', mult_model.coef_)
# The mean squared error
print("RMSE: %.2f" % (rmse(y_pred, y_test)))
```

```
('intercept:', array([ 7220465.95346491]), 'coefficients:', array([[ -3.85547540e+04,   4.429332
          1.59043347e-01,   4.14361852e+03,  -6.01513553e+02,
          2.44382135e+04,   9.34950344e+04,   6.12210257e+05,
          5.06076502e+04,   6.98882157e+01,   4.40768912e+01,
         -2.62156390e+03,   1.96225739e+01,   6.03747455e+05,
         -2.25535679e+05,   2.24949849e+01,  -4.25865615e-01]]))
RMSE: 249311.90
```

In [52]:
```
print('Difference: ',(rmse(y_pred, y_test))-(rmse(y_pred2, y_test2)))
```

```
('Difference: ', 55598.322088692774)
```