Fire up graphlab create

In [3]:

```
import graphlab
graphlab.canvas.set_target('ipynb')
```

Load some house sales data

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

In [4]:

```
sales = graphlab.SFrame('home_data.gl/')
```

```
/home/aluno/.virtualenvs/gl-env/local/lib/python2.7/site-packages/requests/packages/urllib3/util/ssl_.py:315:
 SNIMissingWarning: An HTTPS request has been made, but the SNI (Subject Name Indication) extension to TLS is
 not available on this platform. This may cause the server to present an incorrect TLS certificate, which can
 cause validation failures. For more information, see https://urllib3.readthedocs.org/en/latest/security.html#
snimissingwarning.
  SNIMissingWarning
/home/aluno/.virtualenvs/gl-env/local/lib/python2.7/site-packages/requests/packages/urllib3/util/ssl_.py:120:
 InsecurePlatformWarning: A true SSLContext object is not available. This prevents urllib3 from configuring SS
L appropriately and may cause certain SSL connections to fail. For more information, see https://urllib3.readt
hedocs.org/en/latest/security.html#insecureplatformwarning.
  InsecurePlatformWarning
```

```
This non-commercial license of GraphLab Create for academic use is assigned to alainandre@decom.cefetmg.br and
 will expire on July 19, 2017.
```

```
[INFO] graphlab.cython.cy_server: GraphLab Create v2.0.1 started. Logging: /tmp/graphlab_server_1490113953.log
```

In [5]:

```
sales
```

Out[5]:

	00:00:00+00:00							
7237550310	2014-05-12 00:00:00+00:00	1225000	4	4.5	5420	101930	1	0
1321400060	2014-06-27 00:00:00+00:00	257500	3	2.25	1715	6819	2	0
2008000270	2015-01-15 00:00:00+00:00	291850	3	1.5	1060	9711	1	0
2414600126	2015-04-15 00:00:00+00:00	229500	3	1	1780	7470	1	0
3793500160	2015-03-12 00:00:00+00:00	323000	3	2.5	1890	6560	2	0

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat
0	3	7	1180	0	1955	0	98178	47.51123398
0	3	7	2170	400	1951	1991	98125	47.72102274
0	3	6	770	0	1933	0	98028	47.73792661
0	5	7	1050	910	1965	0	98136	47.52082
0	3	8	1680	0	1987	0	98074	47.61681228
0	3	11	3890	1530	2001	0	98053	47.65611835
0	3	7	1715	0	1995	0	98003	47.30972002
0	3	7	1060	0	1963	0	98198	47.40949984
0	3	7	1050	730	1960	0	98146	47.51229381
0	3	7	1890	0	2003	0	98038	47.36840673

long	sqft_living15	sqft_lot15
-122.25677536	1340.0	5650.0
-122.3188624	1690.0	7639.0
-122.23319601	2720.0	8062.0
-122.39318505	1360.0	5000.0
-122.04490059	1800.0	7503.0
-122.00528655	4760.0	101930.0
-122.32704857	2238.0	6819.0
-122.31457273	1650.0	9711.0
-122.33659507	1780.0	8113.0
-122.0308176	2390.0	7570.0

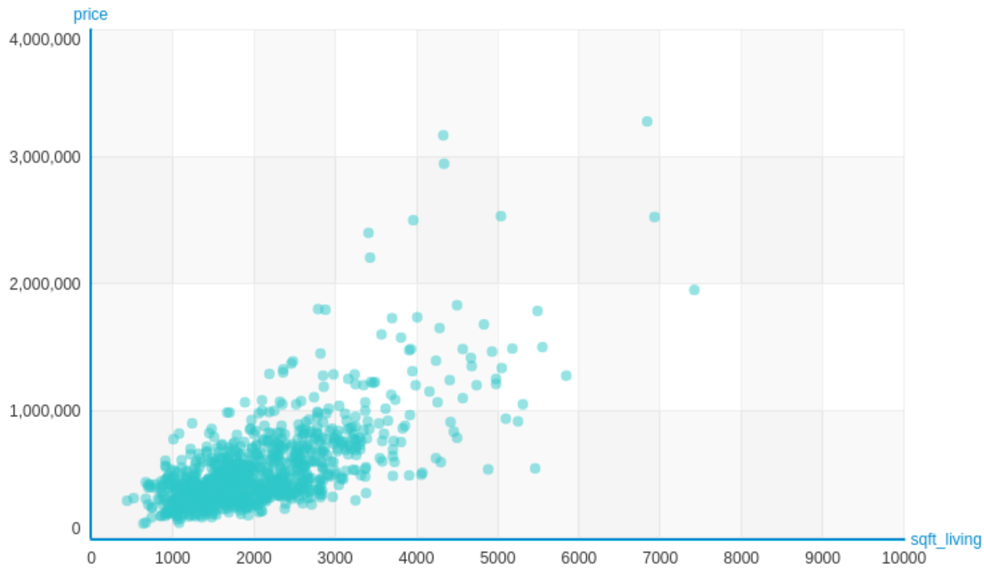
[21613 rows x 21 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.

Exploring the data for housing sales

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

In [6]:

```
sales.show(view="Scatter Plot", x="sqft_living", y="price")
```



Create a simple regression model of sqft_living to price

Split data into training and testing.
We use seed=0 so that everyone running this notebook gets the same results. In practice, you may set a random seed (or let GraphLab Create pick a random seed for you).

In [7]:

```
train_data,test_data = sales.random_split(.8,seed=0)
```

Build the regression model using only sqft_living as a feature

In [8]:

```
sqft_model = graphlab.linear_regression.create(train_data, target='price', features=['sqft_living'],validation_set=None)
```

```
Linear regression:

--------------------------------------------------------

Number of examples          : 17384

Number of features          : 1

Number of unpacked features : 1

Number of coefficients    : 2

Starting Newton Method

--------------------------------------------------------

+-----------+----------+--------------+--------------------+---------------+

| Iteration | Passes   | Elapsed Time | Training-max_error | Training-rmse |

+-----------+----------+--------------+--------------------+---------------+

| 1         | 2        | 1.007346     | 4349521.926170     | 262943.613754 |

+-----------+----------+--------------+--------------------+---------------+

SUCCESS: Optimal solution found.
```

Evaluate the simple model

In [9]:

```
print test_data['price'].mean()
```

```
543054.042563
```

In [10]:

```
print sqft_model.evaluate(test_data)
```

```
{'max_error': 4143550.8825285966, 'rmse': 255191.0287052736}
```

RMSE of about $255,170!

Let's show what our predictions look like

Matplotlib is a Python plotting library that is also useful for plotting. You can install it with:

'pip install matplotlib'

In [15]:

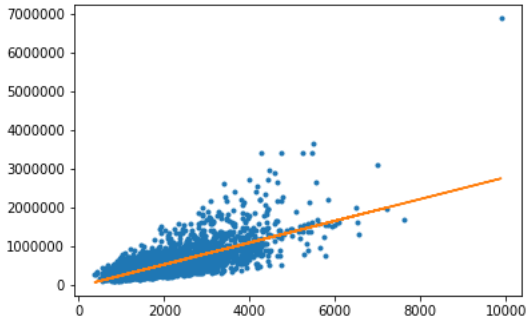
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [16]:

```
plt.plot(test_data['sqft_living'],test_data['price'],'.',
        test_data['sqft_living'],sqft_model.predict(test_data),'-')
```

Out[16]:

```
[<matplotlib.lines.Line2D at 0x7f85545c8650>,
 <matplotlib.lines.Line2D at 0x7f85545c8750>]
```



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

Below: we can view the learned regression coefficients.

In [17]:

```
sqft_model.get('coefficients')
```

Out[17]:

name	index	value	stderr
(intercept)	None	-47114.0206702	4923.34437753
sqft_living	None	281.957850166	2.16405465323

[2 rows x 4 columns]

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].show()
```

bedrooms

dtype:	**str**
num_unique (est.):	**13**
num_undefined:	**0**

frequent items:

3
4
2
5
6
1
7
0
8
9
10
11

bathrooms

dtype:	**str**
num_unique (est.):	**30**
num_undefined:	**0**

frequent items:

2.5
1
1.75
2.25
2
1.5
2.75
3
3.5
3.25
3.75
4

sqft_living

dtype:	**int**
num_unique (est.):	**1.036**
num_undefined:	**0**
min:	**290**
max:	**13.540**
median:	**1.910**
mean:	**2.079,9**
std:	**918,42**

distribution of values:

sqft_lot

dtype:	**int**
num_unique (est.):	**9.747**
num_undefined:	**0**
min:	**520**
max:	**1.651.359**
median:	**7.617**
mean:	**15.106,968**
std:	**41.419,553**

distribution of values:

floors

dtype:	**str**
num_unique (est.):	**6**
num_undefined:	**0**

frequent items:

1
2
1.5
3
2.5
3.5

zipcode

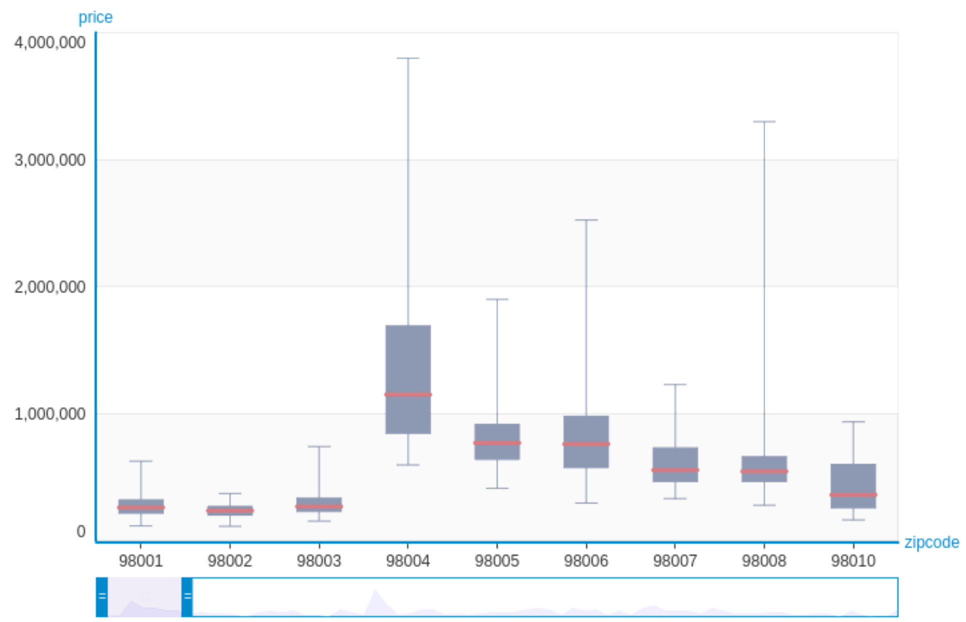
dtype:
num_uniq
num_unde

frequent it

98103
98038
98115
98052
98117
98042
98034
98118
98023
98006
98133
98059

In [20]:

```
sales.show(view='BoxWhisker Plot', x='zipcode', y='price')
```



Pull the bar at the bottom to view more of the data.

98039 is the most expensive zip code.

Build a regression model with more features

In [78]:

```
my_features_model = graphlab.linear_regression.create(train_data,target='price',features=my_features,validation_set=None)
```

```
Linear regression:

--------------------------------------------------------

Number of examples          : 17384

Number of features          : 6

Number of unpacked features : 6

Number of coefficients    : 115

Starting Newton Method

--------------------------------------------------------

+-----------+----------+--------------+--------------------+---------------+

| Iteration | Passes   | Elapsed Time | Training-max_error | Training-rmse |

+-----------+----------+--------------+--------------------+---------------+

| 1         | 2        | 0.044407     | 3763208.270523     | 181908.848367 |

+-----------+----------+--------------+--------------------+---------------+

SUCCESS: Optimal solution found.
```

In [22]:

```
print my_features
```

```
['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'zipcode']
```

Comparing the results of the simple model with adding more features

In [23]:

```
print sqft_model.evaluate(test_data)
print my_features_model.evaluate(test_data)
```

```
{'max_error': 4143550.8825285966, 'rmse': 255191.0287052736}
{'max_error': 3486584.509381705, 'rmse': 179542.4333126903}
```

The RMSE goes down from $255,170 to $179,508 with more features.

Apply learned models to predict prices of 3 houses

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

In [24]:

```
house1 = sales[sales['id']=='5309101200']
```

In [25]:

```
house1
```

Out[25]:

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
5309101200	2014-06-05 00:00:00+00:00	620000	4	2.25	2400	5350	1.5	0

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat
0	4	7	1460	940	1929	0	98117	47.67632376

long	sqft_living15	sqft_lot15
-122.37010126	1250.0	4880.0

[? rows x 21 columns]
Note: Only the head of the SFrame is printed. This SFrame is lazily evaluated.
You can use sf.materialize() to force materialization.

In [26]:

```
print house1['price']
```

```
[620000, ... ]
```

In [27]:

```
print sqft_model.predict(house1)
```

[629584.8197281543]

In [28]:

```
print my_features_model.predict(house1)
```

[721918.9333272863]

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.

Prediction for a second, fancier house

We will now examine the predictions for a fancier house.

In [29]:

```
house2 = sales[sales['id']=='1925069082']
```

In [30]:

```
house2
```

Out[30]:

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
1925069082	2015-05-11 00:00:00+00:00	2200000	5	4.25	4640	22703	2	1

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat
4	5	8	2860	1780	1952	0	98052	47.63925783

long	sqft_living15	sqft_lot15
-122.09722322	3140.0	14200.0

[? rows x 21 columns]
Note: Only the head of the SFrame is printed. This SFrame is lazily evaluated.
You can use sf.materialize() to force materialization.

In [31]:

```
print house2['price']
```

[2200000, ...]

In [32]:

```
print sqft_model.predict(house2)
```

[1261170.404099967]

In [33]:

```
print my_features_model.predict(house2)
```

[1446472.4690774973]

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.

Last house, super fancy

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

In [34]:

```
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 [35]:

```
print my_features_model.predict(graphlab.SFrame(bill_gates))
```

```
[13749825.525719076]
```

The model predicts a price of over $13M 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 [36]:

```
print sqft_model.predict(graphlab.SFrame(bill_gates))
```

```
[14050778.487629175]
```

Answers

1 - Selection and summary statistics

In [44]:

```
filtered = [ x for x in sales if x['zipcode'] == '98004']
```

In [47]:

```
price_total = 0
for x in filtered:
    price_total += x['price']

average = price_total / len(filtered)
```

```
1355927
```

2 - Filtering data

In [51]:

```
houses = [x for x in sales if x['sqft_living']>=2000 and x['sqft_living'] <= 4000]
```

In [64]:

```
len(houses)/(len(sales)*1.0)*100.0
```

Out[64]:

```
42.66413732475825
```

3 - Building a regression model with several more features

In [65]:

```
advanced_features = [
  'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'zipcode',
  'condition', # condition of house
  'grade', # measure of quality of construction
  'waterfront', # waterfront property
  'view', # type of view
  'sqft_above', # square feet above ground
  'sqft_basement', # square feet in basement
  'yr_built', # the year built
  'yr_renovated', # the year renovated
  'lat', 'long', # the lat-long of the parcel
  'sqft_living15', # average sq.ft. of 15 nearest neighbors
  'sqft_lot15', # average lot size of 15 nearest neighbors
]
```

In [76]:

```
advanced_features_model = graphlab.linear_regression.create(train_data,target='price',features=advanced_features,validatio
n_set=None)
```

```
Linear regression:

--------------------------------------------------------

Number of examples          : 17384

Number of features          : 18

Number of unpacked features : 18

Number of coefficients    : 127

Starting Newton Method

--------------------------------------------------------

+-----------+----------+--------------+--------------------+---------------+

| Iteration | Passes   | Elapsed Time | Training-max_error | Training-rmse |

+-----------+----------+--------------+--------------------+---------------+

| 1         | 2        | 0.081864     | 3469012.450686     | 154580.940736 |

+-----------+----------+--------------+--------------------+---------------+

SUCCESS: Optimal solution found.
```

In [79]:

```
print my_features_model.evaluate(test_data)
print advanced_features_model.evaluate(test_data)
```

```
{'max_error': 3486584.509381705, 'rmse': 179542.4333126903}
{'max_error': 3556849.413858208, 'rmse': 156831.1168021901}
```

In [ ]:

```

```