Launch Turi Create

In [2]:
```
import turicreate
```

Load house sales data

In [3]:
```
sales = turicreate.SFrame('data.frame_idx')
```

In [4]:
```
sales
```

Out[4]:

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
7129300520	2014-10-13 00:00:00+00:00	221900.0	3.0	1.0	1180.0	5650.0	1.0	0
6414100192	2014-12-09 00:00:00+00:00	538000.0	3.0	2.25	2570.0	7242.0	2.0	0
5631500400	2015-02-25 00:00:00+00:00	180000.0	2.0	1.0	770.0	10000.0	1.0	0
2487200875	2014-12-09 00:00:00+00:00	604000.0	4.0	3.0	1960.0	5000.0	1.0	0
1954400510	2015-02-18 00:00:00+00:00	510000.0	3.0	2.0	1680.0	8080.0	1.0	0
7237550310	2014-05-12 00:00:00+00:00	1225000.0	4.0	4.5	5420.0	101930.0	1.0	0
1321400060	2014-06-27 00:00:00+00:00	257500.0	3.0	2.25	1715.0	6819.0	2.0	0
2008000270	2015-01-15 00:00:00+00:00	291850.0	3.0	1.5	1060.0	9711.0	1.0	0
2414600126	2015-04-15 00:00:00+00:00	229500.0	3.0	1.0	1780.0	7470.0	1.0	0
3793500160	2015-03-12 00:00:00+00:00	323000.0	3.0	2.5	1890.0	6560.0	2.0	0

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat
0	3	7.0	1180.0	0.0	1955.0	0.0	98178	47.51123398
0	3	7.0	2170.0	400.0	1951.0	1991.0	98125	47.72102274
0	3	6.0	770.0	0.0	1933.0	0.0	98028	47.73792661
0	5	7.0	1050.0	910.0	1965.0	0.0	98136	47.52082
0	3	8.0	1680.0	0.0	1987.0	0.0	98074	47.61681228
0	3	11.0	3890.0	1530.0	2001.0	0.0	98053	47.65611835
0	3	7.0	1715.0	0.0	1995.0	0.0	98003	47.30972002
0	3	7.0	1060.0	0.0	1963.0	0.0	98198	47.40949984
0	3	7.0	1050.0	730.0	1960.0	0.0	98146	47.51229381
0	3	7.0	1890.0	0.0	2003.0	0.0	98038	47.36840673

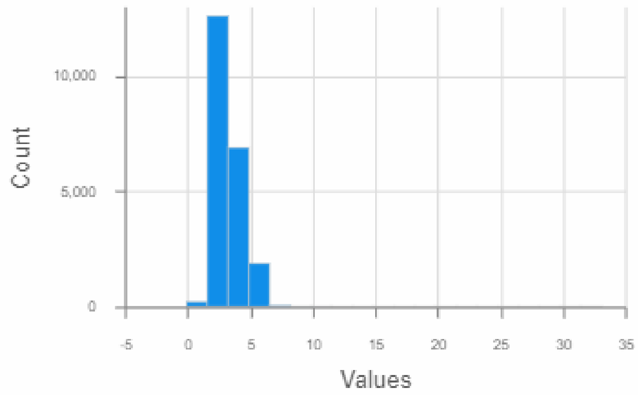
long	sqft_living15	sqft_lot15
-122.25677536	1340.0	5650.0

-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.

Explore

In [60]:
```
sales.show()
```



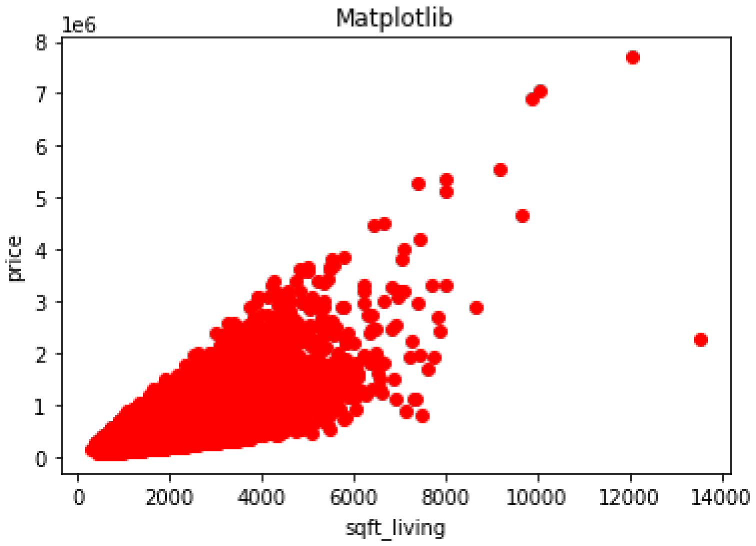
Num. Rows:	21,613
Num. Unique:	13
Missing:	0
Mean:	3.370842
Min:	0
Max:	33
Median:	3
St. Dev:	0.93004

bathrooms float

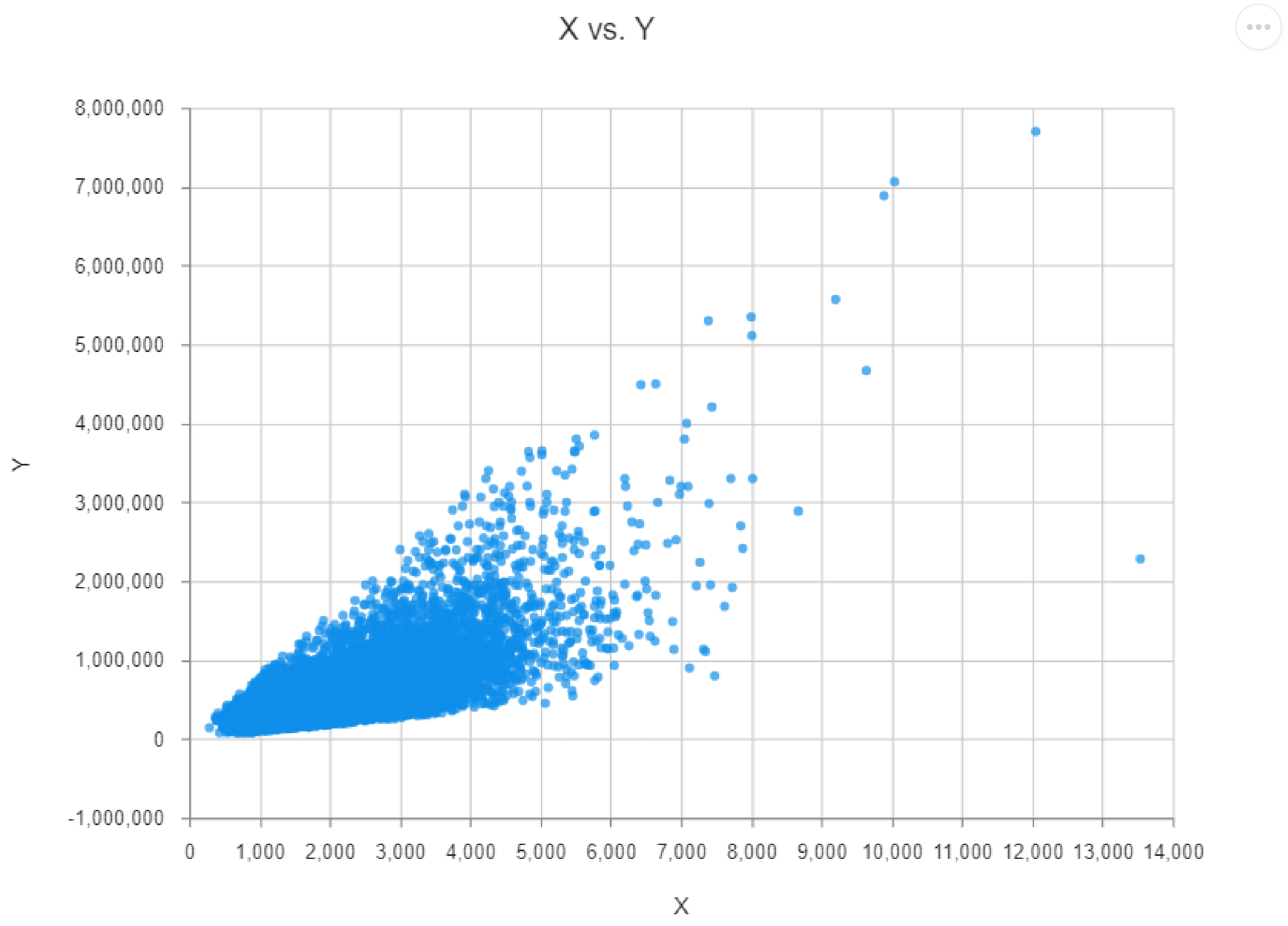
Matplot vs Turi Create's visualization

In [16]:
```
import matplotlib.pyplot as mt
%matplotlib inline

mt.scatter(x= sales['sqft_living'],y=sales['price'],color="r")
mt.xlabel('sqft_living')
mt.ylabel('price')
mt.title("Matplotlib")
mt.show()
```



In [40]:
```
out = turicreate.visualization.scatter(sales['sqft_living'],sales['price'])
out
```

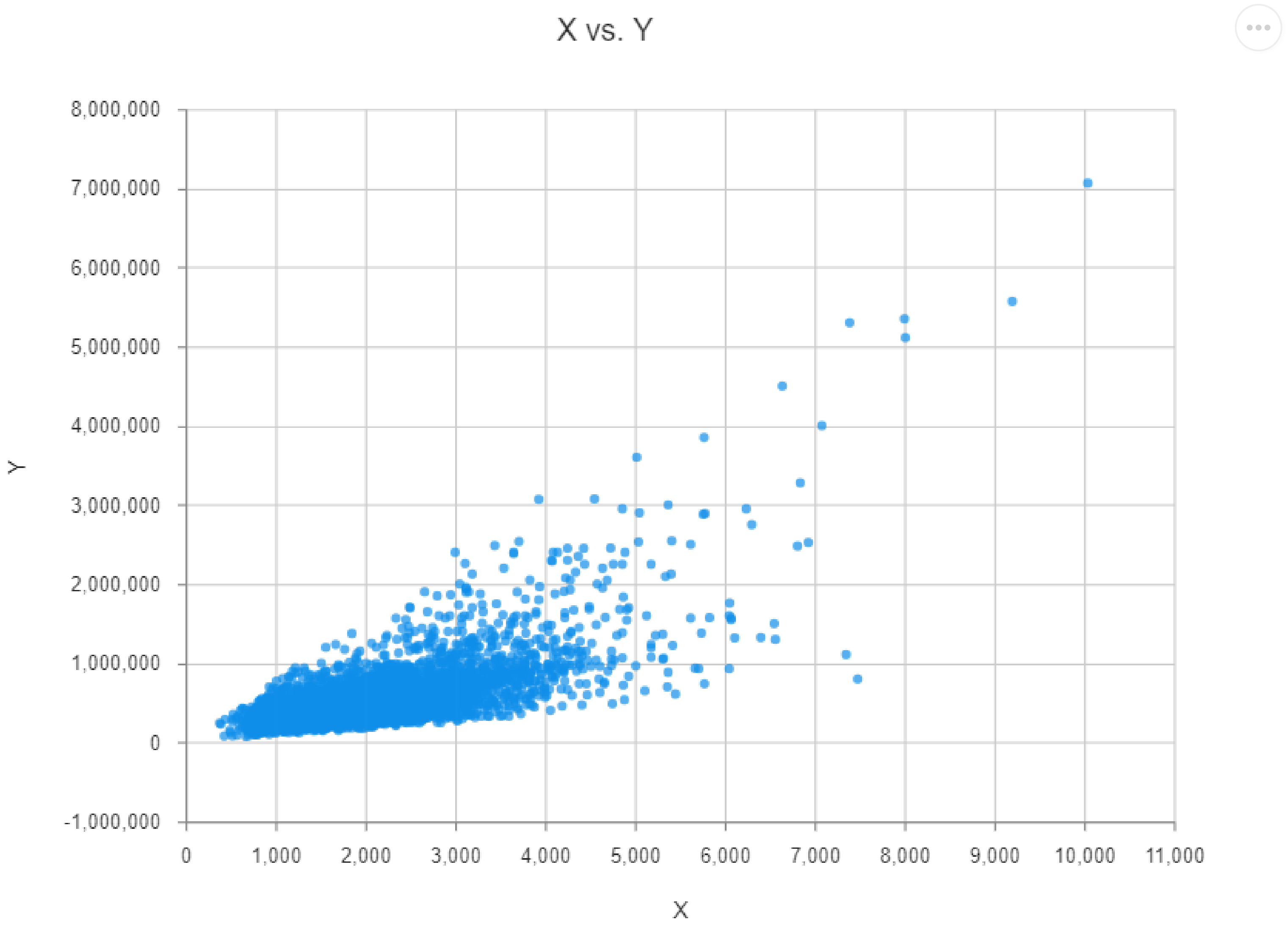


Out[40]: <turicreate.visualization._plot.Plot at 0x7ffcfc0a4e48>

In [9]:
```
turicreate.show(sales[1:5000]['sqft_living'],sales[1:5000]['price'])
```

```
Materializing X axis SArray

Materializing Y axis SArray
```



Simple regression model that predicts price from square feet

In [10]:
```
training_set, test_set = sales.random_split(.8,seed=0)
```

train simple regression model

In [14]:
```
sqft_model = turicreate.linear_regression.create(training_set,target='price',features=['sqft_living'])
```

```
PROGRESS: Creating a validation set from 5 percent of training data. This may take a while.
          You can set ``validation_set=None`` to disable validation tracking.


Linear regression:

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

Number of examples          : 16514

Number of features          : 1

Number of unpacked features : 1

Number of coefficients    : 2

Starting Newton Method

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

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

| Iteration | Passes   | Elapsed Time | Training Max Error | Validation Max Error | Training Root-Mean-Square Error | V
alidation Root-Mean-Square Error |

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

| 1         | 2        | 0.004584     | 4348106.954938     | 2156445.039134       | 263614.586067                   | 2
49870.196532                    |

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

SUCCESS: Optimal solution found.
```

Evaluate the quality of our model

In [17]:
```
print (test_set['price'].mean())
```

```
543054.0425632538
```

In [18]:
```
print (sqft_model.evaluate(test_set))
```

```
{'max_error': 4142375.992218543, 'rmse': 255189.0815203294}
```

Explore model a little further

In [61]:
```
sqft_model.coefficients
```

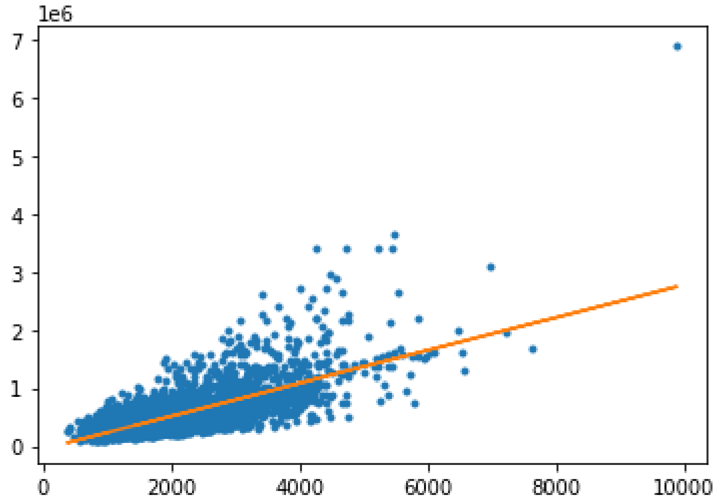
Out[61]:

name	index	value	stderr
(intercept)	None	-47038.38976785168	5067.023999281818
sqft_living	None	282.0689987410828	2.22552235655865

[2 rows x 4 columns]

In [24]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(test_set['sqft_living'],test_set['price'],'.',
        test_set['sqft_living'],sqft_model.predict(test_set),'-')
```

Out[24]:
```
[<matplotlib.lines.Line2D at 0x7ffcfc0f5c50>,
 <matplotlib.lines.Line2D at 0x7ffcfc0f5d30>]
```

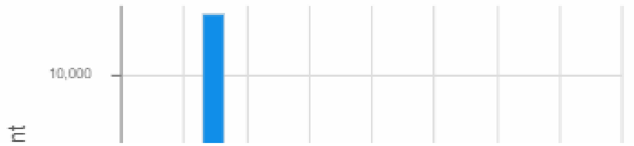


Explore other features of the data

In [25]:
```
my_features = ['bedrooms','bathrooms','sqft_living','sqft_lot','floors','zipcode']
```

In [26]:
```
sales[my_features].show()
```

bedrooms float

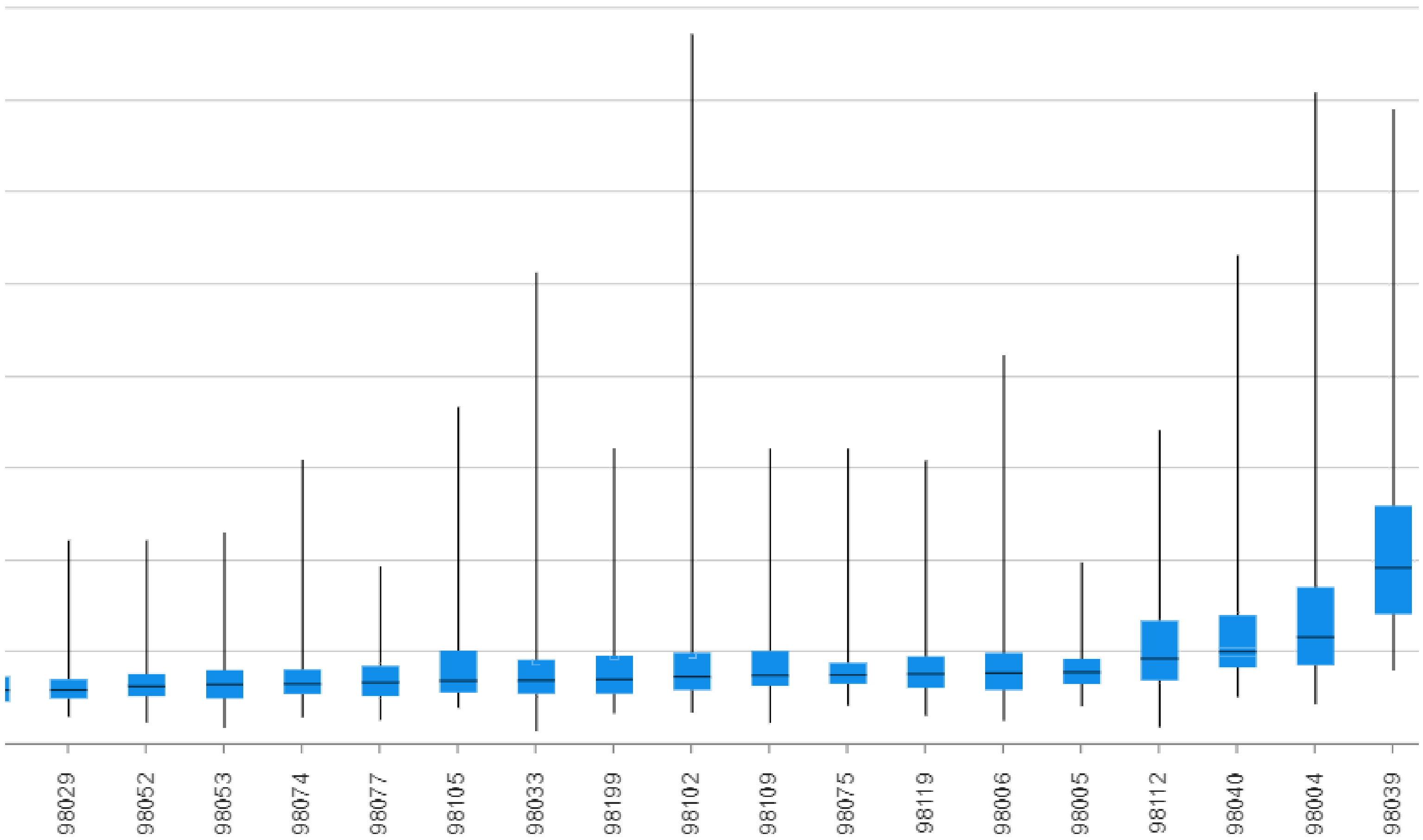


Num. Rows:	21,613
Num. Unique:	13
Missing:	0

In [41]:
```
turicreate.show(sales['zipcode'],sales['price'])
```

```
Materializing X axis SArray

Materializing Y axis SArray
```



Build a model with these additional features

In [99]:
```
my_features_model = turicreate.linear_regression.create(training_set,target='price',features=my_features,validation_set=N
```

```
Linear regression:

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

Number of examples          : 17384

Number of features          : 6

Number of unpacked features : 6

Number of coefficients    : 75

Starting Newton Method

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

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

| Iteration | Passes   | Elapsed Time | Training Max Error | Training Root-Mean-Square Error |

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

| 1         | 2        | 0.030429     | 4086543.315840     | 189216.804808                   |

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

SUCCESS: Optimal solution found.
```

Compare simple model with more complex one

In [100]:
```
print (my_features)
```

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

In [101]:
```
print (sqft_model.evaluate(test_set))
print (my_features_model.evaluate(test_set))
```

```
{'max_error': 4142375.992218543, 'rmse': 255189.0815203294}
{'max_error': 3152242.784868988, 'rmse': 180439.07296640595}
```

Apply learned models to make predictions

In [102]:
```
house1 = sales[sales['id']=='5309101200']
```

In [103]:
```
house1
```

Out[103]:

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
5309101200	2014-06-05 00:00:00+00:00	620000.0	4.0	2.25	2400.0	5350.0	1.5	0

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat
0	4	7.0	1460.0	940.0	1929.0	0.0	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 [104]:
```
print (house1['price'])
```

```
[620000.0, ... ]
```

In [105]:
```
print (sqft_model.predict(house1))
```

```
[629927.2072107471]
```

In [106]:
```
print (my_features_model.predict(house1))
```

```
[729141.9396819306]
```

Prediction for a second house, a fancier one

In [107]:
```
house2 = sales[sales['id']=='1925069082']
```

In [108]:
```
house2
```

Out[108]:

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
1925069082	2015-05-11 00:00:00+00:00	2200000.0	5.0	4.25	4640.0	22703.0	2.0	1

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat
4	5	8.0	2860.0	1780.0	1952.0	0.0	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 [109]:
```
print(house2['price'])
```

```
[2200000.0, ... ]
```

In [110]:
```
print (sqft_model.predict(house2))
```

```
[1261761.7643907724]
```

In [111]:
```
print (my_features_model.predict(house2))
```

```
[1232266.5096878926]
```

Prediction for a super fancy home

In [112]:
```
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 [113]:
```
print (my_features_model.predict(turicreate.SFrame(bill_gates)))
```

```
[15016529.988226019]
```

Assesment

Selection and summary statistics

In [129]:
```
maximum_avg_zip = sales[sales['zipcode']=='98039']
maximum_avg_zip['price'].mean()
```

Out[129]:
```
2160606.5999999996
```

Filtering data

In [115]:
```
filtered = sales[(sales['sqft_living']>2000) & (sales['sqft_living']<4000)]
print("Filtered Count",filtered.shape[0])
print("Actual Count",sales.shape[0])
```

```
Filtered Count 9111
Actual Count 21613
```

In [116]:
```
print("Fraction",filtered.shape[0]/sales.shape[0])
```

```
Fraction 0.4215518437977143
```

Building a regression model with several more features

In [117]:
```
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 [118]:
```
new_model = turicreate.linear_regression.create(training_set,target='price',features=advanced_features,validation_set=No
```

```
Linear regression:

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

Number of examples          : 17384

Number of features          : 18

Number of unpacked features : 18

Number of coefficients    : 87

Starting Newton Method

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

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

| Iteration | Passes   | Elapsed Time | Training Max Error | Training Root-Mean-Square Error |

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

| 1         | 2        | 0.017172     | 4336058.938762     | 162392.982702                 |

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

SUCCESS: Optimal solution found.
```

In [119]:
```
print("RMSE of the new model is ",new_model.evaluate(test_set))
```

```
RMSE of the new model is  {'max_error': 3170363.181382781, 'rmse': 155269.6579279753}
```

In [120]:
```
print("RMSE difference between my_features and advanced_features",
      (new_model.evaluate(test_set)['rmse']-my_features_model.evaluate(test_set)['rmse']))
```

```
RMSE difference between my_features and advanced_features -25169.415038430656
```