Machine Learning Case Study (University of Washington) Notes #2

Regression ML workflow:

1. Acquire and organize training data (crawling), split the data into **training data** and **test data**
2. Feature extraction - decide on which features you'd like to feed into the ML model
3. ML model - run linear regression with given features to predict houseing prices (Y)
4. Use the **test data set** to quality check ML error - RSS (residual sum of square)
5. Use the error to update ML algorithm

Practice with ipython and graphlab

Start graphlab create

```
import graphlab
```

Load some house sales

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

sales
```

id	date	price	bedrooms	bathrooms	sqft_living	sqf
7129300520	2014-10-13 00:00:00+00:00	221900	3	1	1180	56
6414100192	2014-12-09 00:00:00+00:00	538000	3	2.25	2570	72
5631500400	2015-02-25 00:00:00+00:00	180000	2	1	770	10
2487200875	2014-12-09 00:00:00+00:00	604000	4	3	1960	50

1954400510	2015-02-18 00:00:00+00:00	510000	3	2	1680	8
7237550310	2014-05-12 00:00:00+00:00	1225000	4	4.5	5420	10
1321400060	2014-06-27 00:00:00+00:00	257500	3	2.25	1715	6
2008000270	2015-01-15 00:00:00+00:00	291850	3	1.5	1060	9
2414600126	2015-04-15 00:00:00+00:00	229500	3	1	1780	7
3793500160	2015-03-12 00:00:00+00:00	323000	3	2.5	1890	6

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated
0	3	7	1180	0	1955	0
0	3	7	2170	400	1951	1991
0	3	6	770	0	1933	0
0	5	7	1050	910	1965	0
0	3	8	1680	0	1987	0
0	3	11	3890	1530	2001	0
0	3	7	1715	0	1995	0
0	3	7	1060	0	1963	0
0	3	7	1050	730	1960	0

Exploring data for housing

```
graphlab.canvas.set_target('ipynb')
sales.show(view = "Scatter Plot", x = "sqft_living", y = "price")
```

```
train_data, test_data = sales.random_split(.8, seed = 0) #random split at 80% training and
```

Build regression model

```
sqft_model = graphlab.linear_regression.create(train_data, target = 'price', features = ['s
```

```
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          : 16535
```

```
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-rmse | Validation-rmse |
```

```
+-----------+----------+--------------+--------------------+----------------------
+---------------+-----------------+
```

```
| 1         | 2        | 1.032983     | 4330297.944904     | 2146825.443090
| 263793.497730 | 245938.878190   |
```

```
+-----------+----------+--------------+--------------------+----------------------
+---------------+-----------------+
```

```
SUCCESS: Optimal solution found.
```

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

```
543054.042563
```

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

```
2079.36628044
```

```
sqft_model.evaluate(test_data) # feed testing data into regression model
```

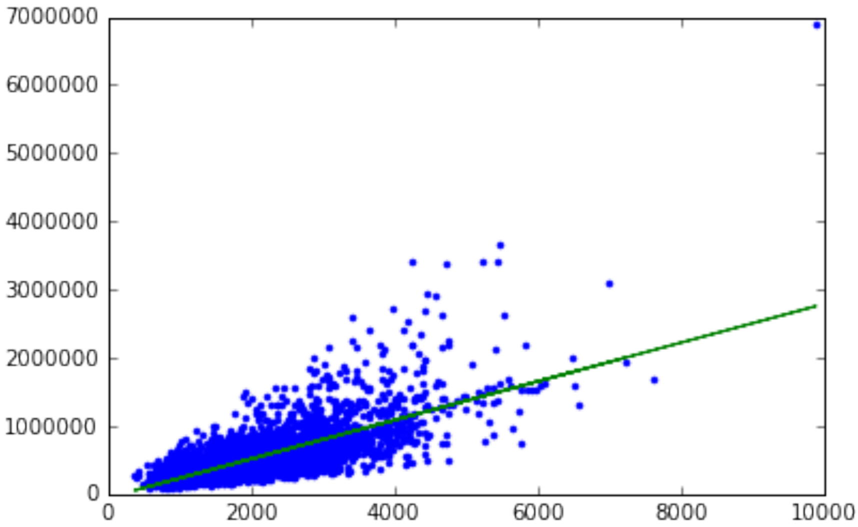
```
{'max_error': 4128404.1045744056, 'rmse': 255233.4942645685}
```

Visualize predictions

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

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

```
[<matplotlib.lines.Line2D at 0x11c970e90>,
 <matplotlib.lines.Line2D at 0x11c970f90>]
```




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

name	index	value	stderr
(intercept)	None	-50635.5486219	5083.29518928
sqft_living	None	283.845444292	2.23423178874

[2 rows x 4 columns]

Explore other features in the data

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

```
sales[my_features].show()
```

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

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

Build regression with more features

```
my_features_model = graphlab.linear_regression.create(train_data, target = 'price', feature
```

```
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          : 16500
```

```
Number of features          : 6
```

```
Number of unpacked features : 6
```

```
Number of coefficients    : 115
```

```
Starting Newton Method
```

```
--------------------------------------------------------
```

```
+-----------+----------+--------------+--------------------+---------------------
+---------------+-----------------+
```

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

```
+-----------+----------+--------------+--------------------+---------------------
+---------------+-----------------+
```

```
| 1         | 2        | 0.041547     | 3812222.718931     | 2513073.167157
| 181701.161639 | 187938.281345   |
```

```
+-----------+----------+--------------+--------------------+---------------------
+---------------+-----------------+
```

```
SUCCESS: Optimal solution found.
```

```python
print my_features
```

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

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

```
{'max_error': 4128404.1045744056, 'rmse': 255233.4942645685}
```

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

```
{'max_error': 3501734.2616914595, 'rmse': 179395.79874371667}
```

Apply learnt model to predict pieces of 3 houses

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

```python
house1
```

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lo
5309101200	2014-06-05 00:00:00+00:00	620000	4	2.25	2400	5350

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	z
0	4	7	1460	940	1929	0	

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.

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

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

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

```
[630593.5176787783]
```

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

```
[718872.7830475004]
```

Prediction for a second fancier house

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

```
house2
```

id	date	price	bedrooms	bathrooms	sqft_living	sqft_l
1925069082	2015-05-11 00:00:00+00:00	2200000	5	4.25	4640	2270

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	z
4	5	8	2860	1780	1952	0	

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.

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

```
[1266407.3128927795]
```

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

```
[1448493.2503331956]
```

Bill Gate's house

```
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]}
```

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

```
[13673416.515746258]
```

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

```
[14141636.6659763]
```

Problem set1

Question 1

```
sales[sales['zipcode'] == '98039'].show()
# Question 1 - find average price of highest average price district by zipcode
# Answer - 2,160,607
```

Question 2

```
sales[(sales['sqft_living'] > 2000) & (sales['sqft_living'] < 4000)].show()

# Number of houses with 2000 < sqft_living <4000 as a percentage to the total house
# 9065/21509 = 0.4215
```

```
sales.show()
```

Question 3

```
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
]
```

```
my_features_model = graphlab.linear_regression.create(train_data, target = 'price',
                                                      features = my_features, validation_se
```

```
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        | 0.043309     | 3763208.270523     | 181908.848367 |
```

```
+-----------+----------+--------------+--------------------+---------------+
```

```
SUCCESS: Optimal solution found.
```

```python
my_features_eval = my_features_model.evaluate(test_data)
print my_features_eval
# Find out RMSE using my_features
```

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

```python
advanced_features_model = graphlab.linear_regression.create(
    train_data, target = 'price', features = advanced_features, validation_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        | 0.062253     | 3469012.450686     | 154580.940736 |
```

```
+-----------+----------+--------------+--------------------+---------------+
```

```
SUCCESS: Optimal solution found.
```

```
advanced_features_eval =  advanced_features_model.evaluate(test_data)
print advanced_features_eval
# Find out RMSE using advanced_features
```

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

```
# Difference between two RMSEs:
print (my_features_eval['rmse'] - advanced_features_eval['rmse'])
```

```
22711.3165105
```