권도형

2016-2-22

머신 러닝 실습

주택 가격 예측

회귀 분석 모델

크게 두 가지 경향이 있다. simple한 모델 이용하기와 다양한 feature을 이용한 모델 이용하기

주택 데이터 읽어들이기

```
import graphlab

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

- ✓ # SFrame은 graphlab create에서 표 형태의 데이터를 담는 자료구조이다. 여기서는 주택 정보를 읽어올 것이다.

[INFO] GraphLab Create v1.8.2 started. Logging:
C:₩Users₩admin₩AppData₩Local₩Temp₩graphlab_server_1456107403.log.0

```
sales
```

- ✓ # 데이터가 자료구조에 담겨진 채로 보여진다.

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
7129300520	2014-10-13 00:00:00+00:00	221900	3	1	1180	5650	1	0
6414100192	2014-12-09 00:00:00+00:00	538000	3	2.25	2570	7242	2	0
5631500400	2015-02-25 00:00:00+00:00	180000	2	1	770	10000	1	0
2487200875	2014-12-09 00:00:00+00:00	604000	4	3	1960	5000	1	0
1954400510	2015-02-18 00:00:00+00:00	510000	3	2	1680	8080	1	0
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

권도형

2016-2-22

머신 러닝 실습

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.

권도형

2016-2-22

머신 러닝 실습

산점도를 통해 데이터의 생김새 관찰하기

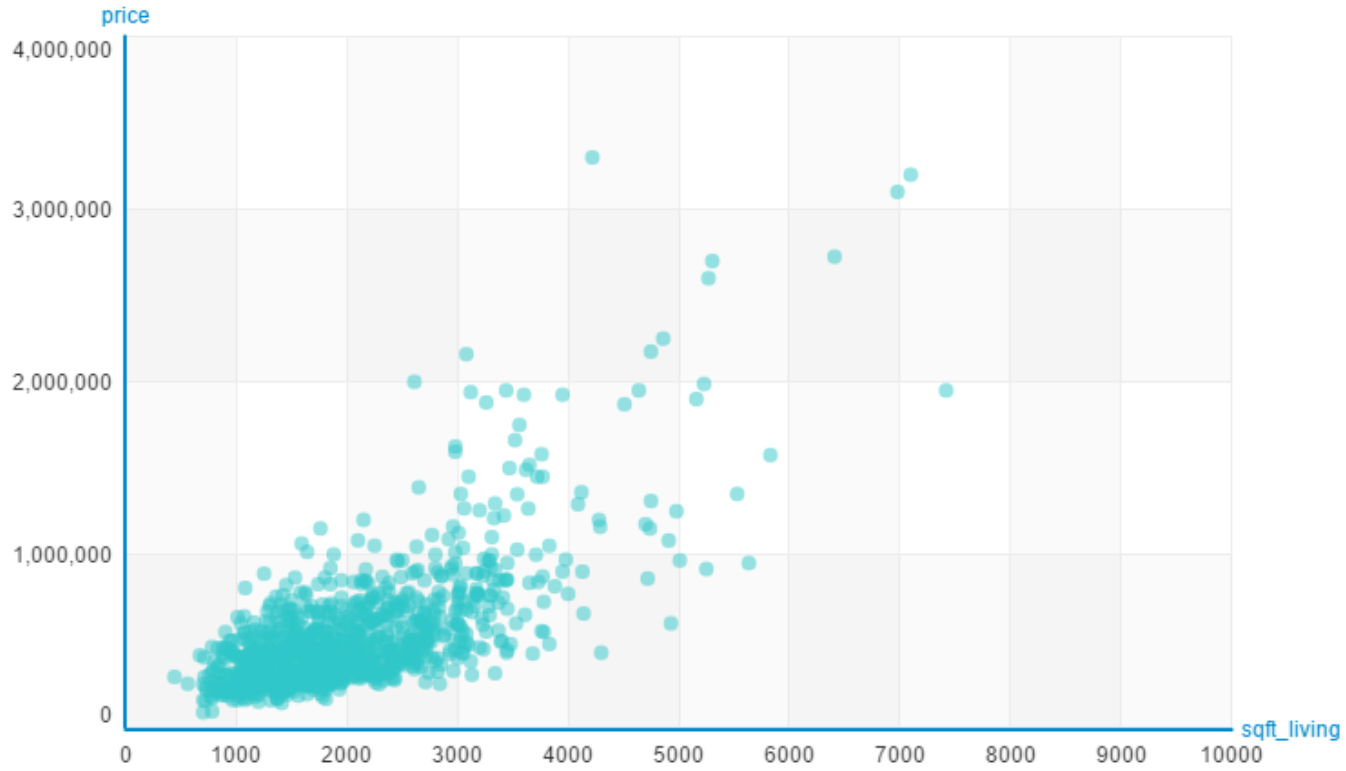
이제 graphlab canvas를 써서 시각화 해본다.

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

- ✓ # graphlab canvas는 새로운 탭에 결과를 보여주려고 하는데 그렇게 하지 않으려면 이렇게 한다. 대상이 기본 브라우저로 되어 있는데 이를 IPython 노트북으로 바꾼다.

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

- ✓ # graphlab이 모든 것을 처리하도록 두지 말고 산점도를 그리도록 지시하는 명령이다. 두 변수의 관계를 나타내는데, X변수는 전용면적을, Y변수는 가격을 의미한다.



권도형

2016-2-22

머신 러닝 실습

전용 면적과 가격 간의 간단한 회귀 모델 만들기

훈련 set와 테스트 set로 나눠야 한다고 배웠다.

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

- ✓ # .8의 의미는 80%가 훈련 set, 20%가 테스트 set라는 의미다. random split함수를 써서 훈련셋과 테스트 셋을 나눈다.

```
sqft_model=graphlab.linear_regression.create(train_data,target='price',features=['sqft_living'])
```

- ✓ # train_data : 훈련 data
- ✓ target='price' : 예측할 대상
- ✓ features=[...] : 특징. 아무것도 넣지 않으면 모든 특징과 모든 열의 데이터를 사용.
- ✓ sqft_model은 graphlab.linear_regression.create에 훈련 data를 넣고, 전용 면적 특징만으로 대상 가격을 예측한다.
- ✓ validation_set을 적지 않은 채로 엔터를 입력하도록 한다.
- ✓ newton's method를 이용하여 회귀 모델을 세운 후 데이터에 대한 예측을 할 수 있게 하는 것을 볼 수 있다.(to make some predictions on the data using Newton's Method)

```
Linear regression:
--------------------------------------------------------
Number of examples          : 16439
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        | 0.007005     | 4350384.034876     | 2899954.371335       | 262836.210167 | 264805.245553   |
+-----------+----------+--------------+--------------------+----------------------+---------------+-----------------+
SUCCESS: Optimal solution found.

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.
```

권도형

2016-2-22

머신 러닝 실습

Evaluating Error[RMSE;root mean square error] of the simple model

```
print test_data['price'].mean()
```
543054.042563

- ✓ # 선형회귀모델을 훈련시켰으므로 평가해볼 단계다.(테스트 데이터를 이용해볼 단계다.)
- ✓ 테스트 데이터의 기준을 먼저 알아보기 위해 위의 명령을 출력해본다.
- ✓ "테스트 data의 가격(price)열에서 평균 가격은 얼마인가?"

```
print sqft_model.evaluate(test_data)
```
{'max_error': 4144231.541497713, 'rmse': 255189.51977487374}

- ✓ # 만든 sqft_model에 평가 함수를 호출하여 테스트 데이터 set를 입력한다.
- ✓ 결과에 대한 해석 : 테스트 data에 대한 최고 오류는 410만, 주택 하나가 이상점인 것으로 보인다. RMSE 도 높은 수치다.(안 좋다는 소리)

권도형

2016-2-22

머신 러닝 실습

모델을 이용하여 시각화하기

시각화의 방법은 두 가지. graphlab.canvas를 이용하는 것과 matplotlib을 이용하는 것

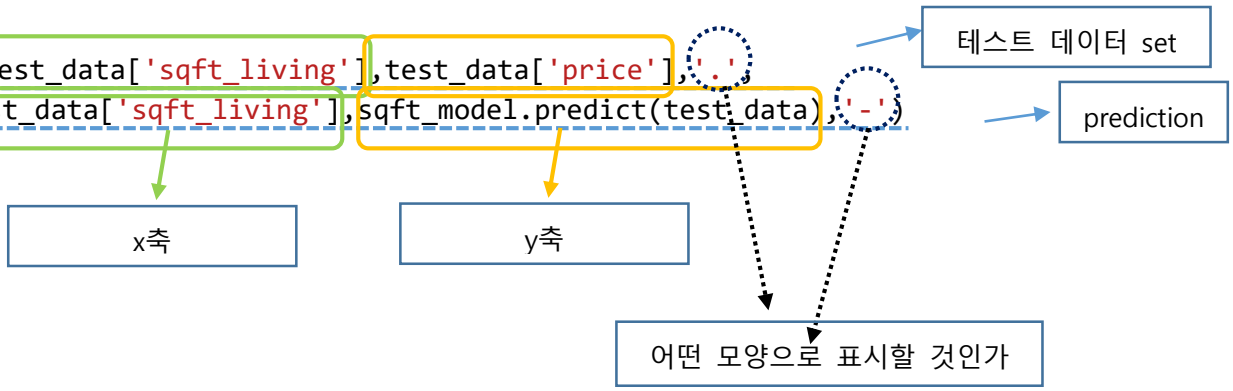
```
import matplotlib.pyplot as plt
```

- ✓ # pyplot을 이용하여 그래프를 쉽게 그리고 싶다.
- ✓ plt는 예명임

```
%matplotlib inline
```

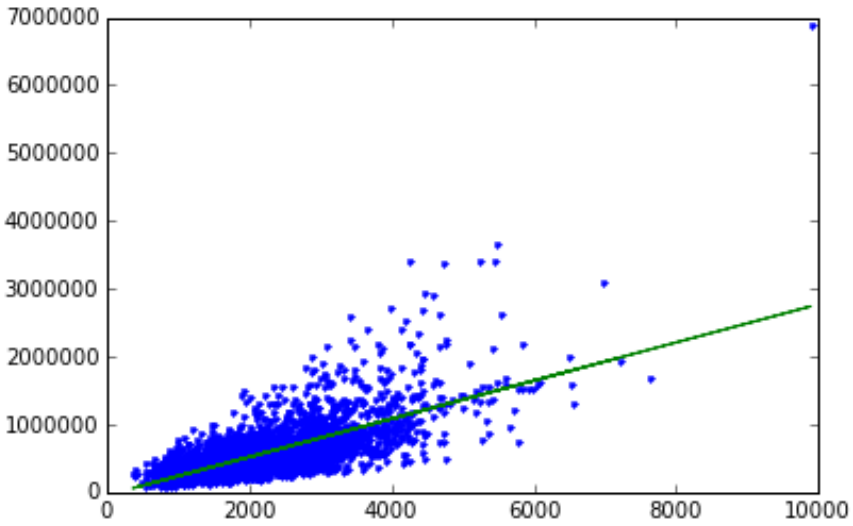
- ✓ # 노트북에 직접 출력하도록 지시한다.

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



- ✓ 테스트 데이터를 두 가지 방법으로 그리고 겹칠 것이다.

[<matplotlib.lines.Line2D at 0x2bf68128>,
 <matplotlib.lines.Line2D at 0x2dcf9470>]



- ✓ 파란점이 test data, 초록색선이 회귀 모델에 의한 예측 가격
- ✓

권도형

2016-2-22

머신 러닝 실습

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

- ✓ # 직선 모델의 계수를 살펴보고자 한다.
- ✓ coefficient : 계수, 가중치

name	index	value	stderr
(intercept)	None	-46963.8750567	5064.91812434
sqft_living	None	281.873845658	2.22434930554

[2 rows x 4 columns]

- ✓ w_0 : y절편. : -44000달러
- ✓ w_1 : 기울기,각도 ; 주택 한 평당 가격은 280달러

권도형

2016-2-22

머신 러닝 실습

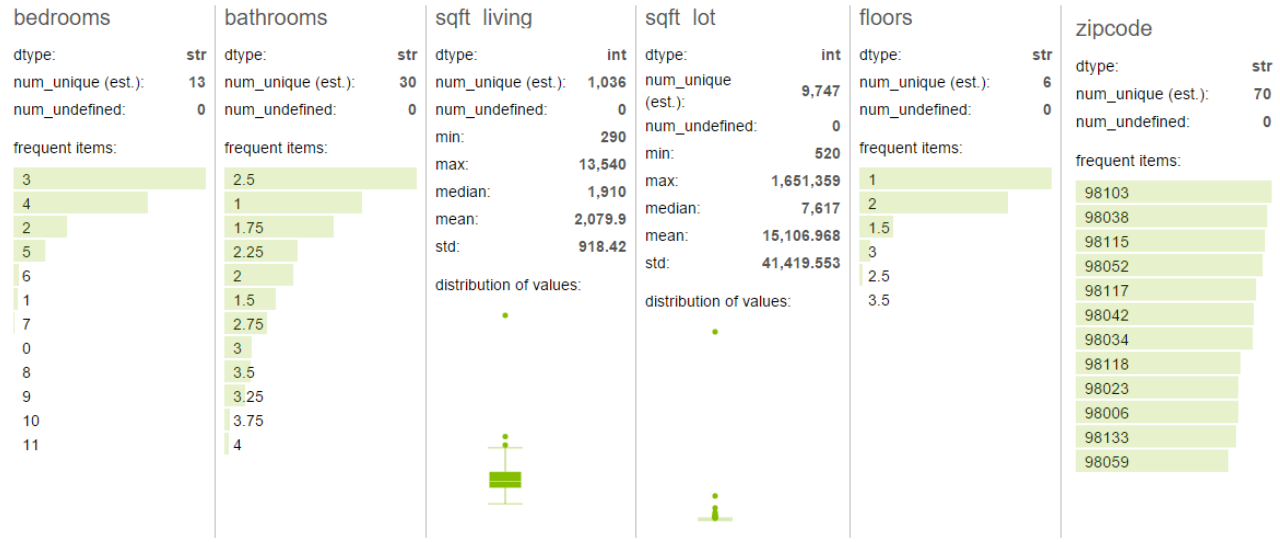
다른 특징들도 고려한 회귀 분석

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

✓ 탐구해볼 다른 특징들의 목록화

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

✓ 특징을 시각화한다.



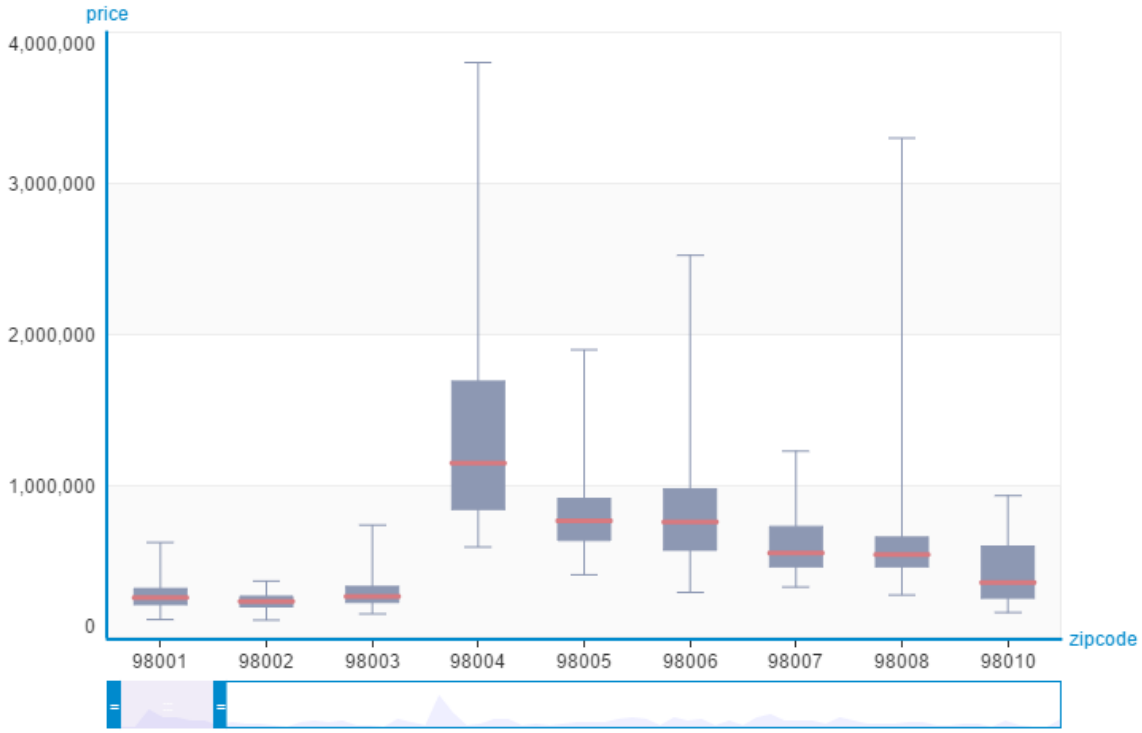

```
sales.show(view='BoxWhisker Plot',x='zipcode',y='price')
```

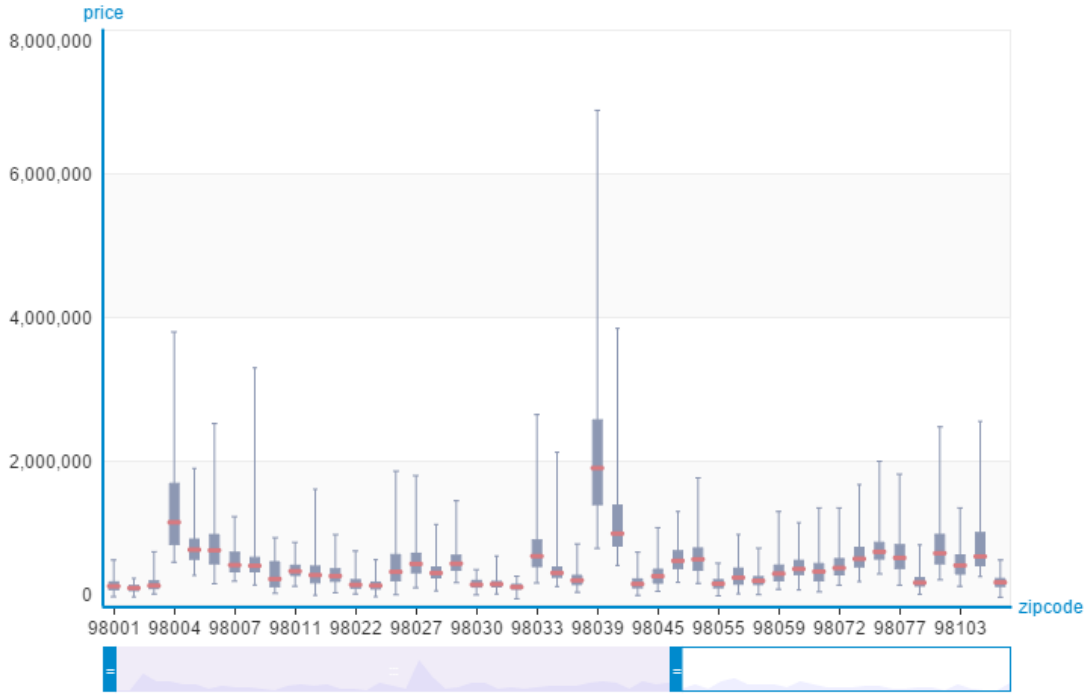
✓ 또 다른 시각화. 상자 그림으로 표현하기. 우편 번호로 표현되는 위치와 가격의 관련성

권도형

2016-2-22

머신 러닝 실습





- ✓ 98003의 주택가격은 상당히 낮고 변동성 또한 크지 않다.
- ✓ 98039는 가장 높은 가격을 나타내고 있으며 변동성 또한 엄청나다.

권도형

2016-2-22

머신 러닝 실습

특징을 더 추가함으로써 모델이 더 향상되었는지 검증해보기

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

- ✓ train_data : 훈련 데이터
- ✓ target='price' : 대상
- ✓ features = [...] : 우리가 생성한 특징 리스트를 받게 한다.

```
Linear regression:
--------------------------------------------------------
Number of examples          : 16528
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        | 0.033005     | 3762726.291708     | 1417911.299029       | 182701.629923 | 167431.372780   |
+-----------+----------+--------------+--------------------+----------------------+---------------+-----------------+
SUCCESS: Optimal solution found.

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.
```

- ✓ 역시 뉴튼법을 사용중인 것을 볼 수 있다.

```
print my_features
```

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

어떤 특징들이 있었는지 확인해보기 위해 출력하기

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

{'max_error': 4144231.541497713, 'rmse': 255189.51977487374}
{'max_error': 3480580.765453957, 'rmse': 179615.3617833967}

- ✓ simple model의 경우 가장 큰 오류가 400만인 반면 더 많은 특징을 추가한 모델은 348만을 나타냄(약간 내려간 수치. 긍정적)
- ✓ RMSE 즉 평균오류는 25.5만에서 17.9만으로 내려감
- ✓ 결론 : 특징을 추가할수록 오류가 줄어들 확률 또한 높아진다. 단지 몇몇 특징을 추가하는 것만으로 성능은 개선된다.

권도형

2016-2-22

머신 러닝 실습

실제 주택 가격 예측에 적용하기

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

✓ 특정 집에 대한 정보를 불러내어 house1에 저장한다.

```
house1
```

✓ 그 결과를 출력한다.

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 len(sf) to force materialization.

esc+m & 〈 img src="house-5309101200.jpg" 〉

✓ ipython에선 html도 사용 가능하다.

권도형

2016-2-22

머신 러닝 실습

```
print house1['price']
```
[620000L, ...]

- ✓ 주택 가격만을 출력해본다. 이 가격은 실제 거래된 가격이다. 비교의 기준이 된다.

```
print sqft_model.predict(house1)
```
[629533.3545228285]

- ✓ simple모델로 예측한 집 값이다. 상당히 근접한 수치다.

```
print my_features_model.predict(house1)
```
[722351.0862722755]

- ✓ 두 번째 모델로 예측한 집 값이다. 상대적으로 빗나갔다.

- ✓ 평균적으로는 feature를 추가할수록 예측이 정확해지지만 예외는 항상 존재한다.

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

house2

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 len(sf) to force materialization.

권도형

2016-2-22

머신 러닝 실습

```
print house2['price']
```
[2220000L, ...]

- ✓ 주택 가격만을 출력해본다. 이 가격은 실제 거래된 가격이다. 비교의 기준이 된다.

```
print sqft_model.predict(house2)
```
[1260930.768797059]

```
print my_features_model.predict(house2)
```
[1460521.2987769814]

- ✓ 더 나은 결과물이다.

권도형

2016-2-22

머신 러닝 실습

매우 다양한 집의 특성 : 사전을 이용한다.

```
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]}
```

✓ 집을 묘사(추상화, 모델링)한 '사전'이다.

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

[13801760.587000746]

사전을 SFrame 형식으로 변환하여 결과값을 출력한다.