

Linear Regression w/sklearn

Linear Regression

**Director of TEAMLAB
Sungchul Choi**



Boston House Price Dataset

- 머신 러닝 등 데이터 분석을 처음 배울 때,
가장 대표적으로 사용하는 **Example Dataset**
- 1978년에 발표된 데이터로, 미국 인구통계 조사 결과
미국 보스턴 지역의 주택 가격에 영향 요소들을 정리함

<http://lib.stat.cmu.edu/datasets/boston>

Boston House Price Dataset

X 변수
13개

Y 변수

[01]	CRIM	자치시(town) 별 1인당 범죄율
[02]	ZN	25,000 평방피트를 초과하는 거주지역의 비율
[03]	INDUS	비소매상업지역이 점유하고 있는 토지의 비율
[04]	CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
[05]	NOX	10ppm 당 농축 일산화질소
[06]	RM	주택 1가구당 평균 방의 개수
[07]	AGE	1940년 이전에 건축된 소유주택의 비율
[08]	DIS	5개의 보스턴 직업센터까지의 접근성 지수
[09]	RAD	방사형 도로까지의 접근성 지수
[10]	TAX	10,000 달러 당 재산세율
[11]	PTRATIO	자치시(town)별 학생/교사 비율
[12]	B	$1000(Bk-0.63)^2$, 여기서 Bk는 자치시별 흑인의 비율을 말함.
[13]	LSTAT	모집단의 하위계층의 비율(%)
[14]	MEDV	본인 소유의 주택가격(중앙값) (단위: \$1,000)

<http://www.dator.co.kr/ctg258/textyle/1721307>

<http://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html>

데이터 로딩

```
from sklearn.datasets import load_boston
import matplotlib.pyplot as plt
import numpy as np
```

```
boston = load_boston()

x_data = boston.data
y_data = boston.target.reshape(boston.target.size,1)

x_data[:3]
```

데이터 로딩

```
array([[ 6.32000000e-03,  1.80000000e+01,  2.31000000e+00,
        0.00000000e+00,  5.38000000e-01,  6.57500000e+00,
        6.52000000e+01,  4.09000000e+00,  1.00000000e+00,
        2.96000000e+02,  1.53000000e+01,  3.96900000e+02,
        4.98000000e+00],
       [ 2.73100000e-02,  0.00000000e+00,  7.07000000e+00,
        0.00000000e+00,  4.69000000e-01,  6.42100000e+00,
        7.89000000e+01,  4.96710000e+00,  2.00000000e+00,
        2.42000000e+02,  1.78000000e+01,  3.96900000e+02,
        9.14000000e+00],
       [ 2.72900000e-02,  0.00000000e+00,  7.07000000e+00,
        0.00000000e+00,  4.69000000e-01,  7.18500000e+00,
        6.11000000e+01,  4.96710000e+00,  2.00000000e+00,
        2.42000000e+02,  1.78000000e+01,  3.92830000e+02,
        4.03000000e+00]])
```

데이터 스케일링

```
from sklearn import preprocessing

minmax_scale = preprocessing.MinMaxScaler().fit(x_data)
x_scaled_data = minmax_scale.transform(x_data)

x_scaled_data[:3]
```

데이터 스케일링

```
array([[ 0.00000000e+00,  1.80000000e-01,  6.78152493e-02,
         0.00000000e+00,  3.14814815e-01,  5.77505269e-01,
         6.41606591e-01,  2.69203139e-01,  0.00000000e+00,
         2.08015267e-01,  2.87234043e-01,  1.00000000e+00,
         8.96799117e-02],
       [ 2.35922539e-04,  0.00000000e+00,  2.42302053e-01,
         0.00000000e+00,  1.72839506e-01,  5.47997701e-01,
         7.82698249e-01,  3.48961980e-01,  4.34782609e-02,
         1.04961832e-01,  5.53191489e-01,  1.00000000e+00,
         2.04470199e-01],
       [ 2.35697744e-04,  0.00000000e+00,  2.42302053e-01,
         0.00000000e+00,  1.72839506e-01,  6.94385898e-01,
         5.99382080e-01,  3.48961980e-01,  4.34782609e-02,
         1.04961832e-01,  5.53191489e-01,  9.89737254e-01,
         6.34657837e-02]])
```

Train-Test Split

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(x_scaled_data, y_data, test_size=0.33)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((339, 13), (167, 13), (339, 1), (167, 1))
```


Linear regression fitting

```
from sklearn import linear_model

regr = linear_model.LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=1)
regr.fit(x_scaled_data, y_data)

## The coefficients
print('Coefficients: ', regr.coef_)
print('intercept: ', regr.intercept_)
```

```
Coefficients: [[ -9.53495156   4.63952195   0.56906733   2.6885614  -8.64873871
 19.85700309   0.07292809 -16.22877191   7.03006588  -6.46057746
 -8.96255741   3.7248827  -19.04291078]]
intercept: [ 26.61291386]
```

$$y = w_1x_1 + w_2x_2 + \beta_3x_3 + w_4x_4 + w_5x_5 \\ + w_6x_6 + w_7x_7 + \cdots w_{13}x_{13} + w_0 \cdot 1$$

수식 결과비교

```
regr.predict(x_data[0].reshape(1,-1))  
array([[ -483.19114376]])
```

```
x_data[0].dot(regr.coef_.T) + regr.intercept_  
array([ -483.19114376])
```

$$\sum_{i=0}^{13} w_i x_i = \mathbf{w}^T \cdot \mathbf{x}$$

Metric 측정

```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
```

```
y_true = y_test
y_hat = regr.predict(X_test)
```

```
r2_score(y_true, y_hat), mean_absolute_error(y_true, y_hat), mean_squared_error(y_true, y_hat)

(0.6835404325597263, 3.5517583251413343, 27.19728458900094)
```



Human knowledge belongs to the world.