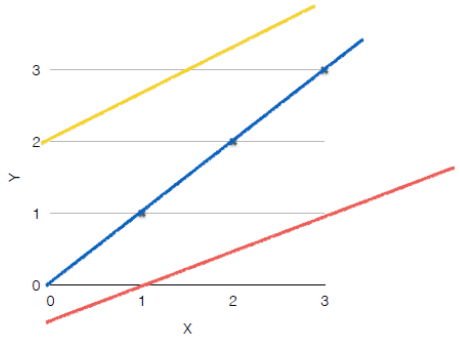
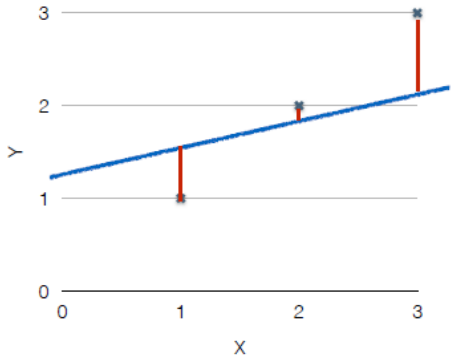
Chapter 1	machine Learning with Scikit-Learn	regression

[선형 회귀]

H(x) = Wx + b



Cost function (Loss)



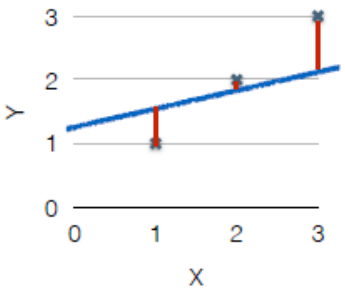
어떤 것이 좋은 가설인가? 실제 값과 거리가 먼 것은 나쁘고, 가까운 것은 좋은 것.

Cost function (Loss) : 우리가 세운 가설과 실제 데이터가 얼마나 가까운가를 구하는 함수.

가설을 세운 H(x) 에서 실제 값 y의 차를 구한 후 제곱의 합

$$\frac{(H(x^{(1)})-y^{(1)})^2+(H(x^{(2)})-y^{(2)})^2+(H(x^{(3)})-y^{(3)})^2}{3}$$

$$H(x)=Wx+b$$



$$cost=\frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)})-y^{(i)})^2$$

머신 러닝의 목표 : minimize cost

회귀분석(regression analysis)은 D차원 벡터 독립 변수 x와 이에 대응하는 스칼라 종속 변수 y 간의 관

계를 정량적으로 찾아내는 작업이다.
회귀분석에는 결정론적 모형(deterministic Model)과 확률적 모형(probabilistic Model)이 있다.
결정론적 회귀분석 모형은 독립 변수 x에 대해 대응하는 종속 변수 y와 가장 비슷한 값 y^를 출력하는 함수 H(x)를 찾는 과정이다.
만약 독립 변수 x와 이에 대응하는 종속 변수 y간의 관계가 다음과 같은 선형 함수 H(x)이면
선형 회귀분석(linear regression analysis)이라고 한다.

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = w_0 + w^T x$$

위 식에서 w0,⋯,wD를 함수 H(x)의 계수(coefficient)이자 이 선형 회귀모형의 parameter 라고 한다.

(핵심 함수)

```
model = LinearRegression()      # 학습 모델 선택
model = model.fit(x, y)         # 학습
result =model.predict([[7]]) # 예측
```

ex07_regression.py (단순 선형 회귀) x: [공부시간]

```
from sklearn.linear_model import LinearRegression

x = [[10],[5],[9],[7]]      #공부시간 10시간  5시간, 9시간, 7시간
y = [[100],[50],[90],[77]]  #시험점수 100점   50점,  90점   77점

model = LinearRegression()

model = model.fit(x, y)
result =model.predict([[7]])

print(result)
```

실행 결과:

[[72.01694915]]

ex08_regression.py (다중 선형 회귀) x: [공부시간, 학년]

```
from sklearn.linear_model import LinearRegression

x = [[10,3],[5,2],[9,3],[7,3]]      #공부시간,학년: 10시간,3  5시간,2, 9시간,3, 7시간,3
y = [[100],[50],[90],[77]]          #시험점수:      100점      50점,    90점      77점

model = LinearRegression()

model = model.fit(x, y)
result =model.predict([[7,2]])   #7시간공부,  2학년

print(result)
```

실행 결과:

```
[[65.]]
```

(데이터에 대한 사전 조사)

데이터에 결측치 또는 이상한 값(outlier) 이 있는지 확인
각 데이터가 연속적인 실수값인지 범주형 값인지 확인
실수형 데이터의 분포가 정규 분포인지 확인
실수형 데이터에 양수 혹은 범위 등으로 제한 조건이 있는지 확인
범주형 데이터의 경우 범주의 값이 어떤 값 혹은 숫자로 표현되어 있는지 확인
데이터 간의 상관관계를 확인

sklearn에 예제 데이터 중 보스턴 집값 예측 예

scikit-learn 이 제공하는 회귀 분석용 예제 데이터 중 하나인 보스턴 주택 가격 데이터에 대해 소개한다. 이 데이터는 다음과 같이 구성되어 있다.

- 타겟 데이터
  - 1978 보스턴 주택 가격
  - 506 타운의 주택 가격 중앙값 (단위 1,000 달러)

- 특징 데이터
  - CRIM: 범죄율
  - INDUS: 비소매상업지역 면적 비율
  - NOX: 일산화질소 농도
  - RM: 주택당 방 수
  - LSTAT: 인구 중 하위 계층 비율
  - B: 인구 중 흑인 비율
  - PTRATIO: 학생/교사 비율
  - ZN: 25,000 평방피트를 초과 거주지역 비율
  - CHAS: 찰스강의 경계에 위치한 경우는 1, 아니면 0
  - AGE: 1940 년 이전에 건축된 주택의 비율
  - RAD: 방사형 고속도로까지의 거리
  - DIS: 직업센터의 거리
  - TAX: 재산세율

ex09_regression.py

```
from sklearn.datasets import load_boston
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

boston = load_boston()
dfX = pd.DataFrame(boston.data, columns=boston.feature_names)
dfy = pd.DataFrame(boston.target, columns=["MEDV"])
df = pd.concat([dfX, dfy], axis=1)
print( df.head())
```

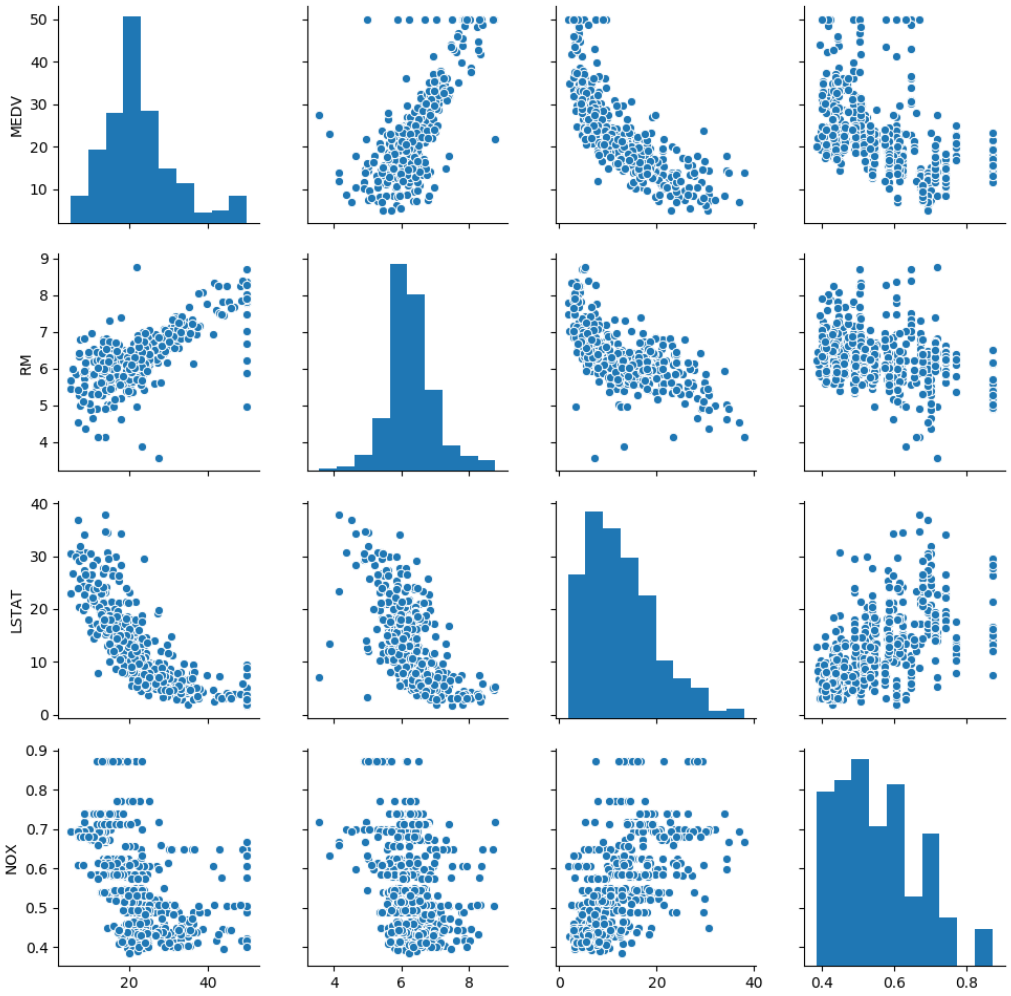
```
cols = ["MEDV", "RM", "LSTAT", "NOX"]
sns.pairplot(df[cols])
plt.show()
"""
가격(MEDV)과 RM 데이터가 강한 양의 상관관계,
LSTAT, NOX 데이터와 강한 음의 상관관계
"""
data = df[["RM","LSTAT","NOX"]]
label = df["MEDV"]

model = LinearRegression( )
model = model.fit(data, label)

predict = model.predict([[6, 9.67, 0.573 ]]) # RM(방 수):6개,   LSTAT:9.67   NOX: 0.573
print("예측 집값 : ", predict)
```

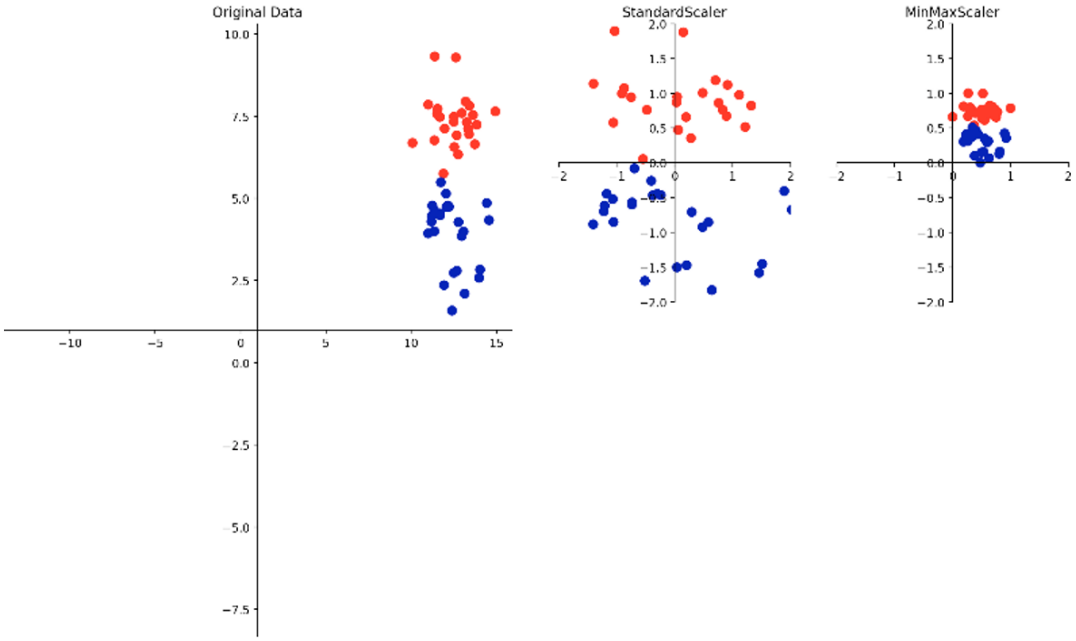
실행결과 :

	CRIM	ZN	INDUS	CHAS	NOX	...	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	...	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	...	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	...	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	...	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	...	222.0	18.7	396.90	5.33	36.2



예측 집값 : [22.89854372]

Chapter 1	machine Learning with Scikit-Learn	Scaler



MinMaxSclaer는 매우 다른 스케일의 범위를 0과 1사이로 변환. (정규화)
StandardScalar는 각 특성의 평균을 0, 분산을 1로 변경하여 모든 특성이 같은 크기를 가지게 함. 이 방법은 특성의 최솟값과 최댓값 크기를 제한하지 않음. (표준화)

ex10_minmaxScaler.py

```
from sklearn import preprocessing
scaler = preprocessing.MinMaxScaler() # 데이터 정규화

after_Normalization = scaler.fit_transform([[100],[1],[200],[1]])
print( after_Normalization )

reverseData = scaler.inverse_transform( after_Normalization )
print( reverseData )
```

실행결과

```
[[0.49748744]
[0. ]
[1. ]
[0. ]]
[[100.]
[ 1.]
```

[200.]

[1.]]