파머완 5장(Part5.1~5.8): 회귀

상태	진행 중
담당자	시현 이
마감일	@10/13/2025
작업 유형	개념정리 및 필사
설명	파이썬 머신러닝 완벽 가이드_개정2판_제5장 개념 정리+ 필사 링크 + kaggle
업데이트 시간	@October 12, 2025 12:01 AM

01.회귀 소개

회귀 분석이란, 데이터 값이 평균과 같은 일정한 값으로 돌아가려는 경향을 이용한 통계학 기법

⇒갈톤의 유전적 특성을 연구에서 비롯됨→ 사람의 키는 평균 키로 회귀하려는 경향을 가진다는 자연의 법칙

- 통계학적 의미: 여러 개의 독립변수와 한 개의 종속 변수 간의 상관관계를 모델링하는 기법을 통칭
    - ex) 독립변수: 아파트의 방 개수, 방 크기, 주변 학군 등 // 종속변수: 아파트 가격
        - Y = W1*X1 + W2*X2 + W3*X3 + ... + Wn*Xn (선형 회귀식)이 있으면 X 변수는 방 개수, 방크기, 주변 학군 등의 독립변수, Y는 아파트의 가격을 의미하는 종속변수, W 변수는 독립변수의 값에 영향을 미치는 회귀 계수(Regression Coefficients)
- 머신러닝 관점: 독립변수→ feature// 종속 변수 → 결정값
    - 머신러닝 회귀 예측의 핵심은 주어진 피처와 결정 값 데이터 기반에서 학습을 통해 최적의 회귀 계수를 찾아내는 것.

<회귀의 유형>

회귀 계수의 선형성에 따라 선형/비선형 회귀로 분류하며 독립변수에 개수에 따라 단일/다중 회귀로 나눔

독립변수 개수	회귀 계수의 결합
1개: 단일 회귀	선형: 선형회귀
여러 개: 다중 회귀	비선형: 비선형 회귀

<대표적인 선형 회귀 모델>

지도학습의 두가지 유형: 분류(예측 값이 이산형), 회귀(예측 값이 연속형 숫자값)

회귀 중에서 가장 많이 사용되는 것이 '선형 회귀'

선형 회귀란, 실제 값과 예측값의 차이(오류의 제곱값)를 최소화하는 직선형 회귀선을 최적화하는 방식.

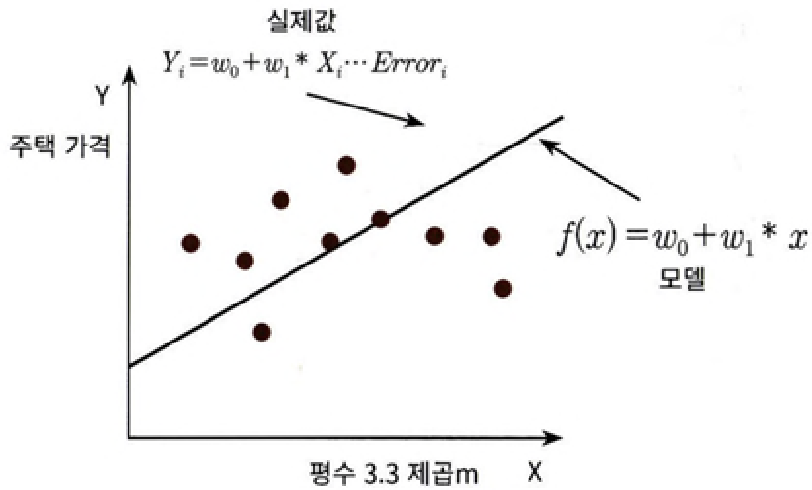
⇒ 규제 방법(과적합 문제 해결을 위해 회귀 계수에 패널티 값 적용)에 따라 별도의 유형으로 나뉠 수 있음

- 일반 선형 회귀: 예측값과 실제값의 RSS(Residual Sum of Squares)를 최소화할 수 있도록 회귀 계수 최적화. 규제 적용X
- 릿지(Ridge): (선형 회귀 + L2 규제) 모델.
  - L2 규제: 상대적으로 큰 회귀 계수 값의 예측 영향도를 감소시키기 위해 회귀 계수값을 더 작게 만드는 규제 모델
- 라쏘(Lasso): (선형 회귀 + L1 규제) 모델.
  - L1 규제: 예측 영향력이 작은 피처의 회귀 계수를 0으로 만들어 회귀 예측 시 피처가 선택되지 않도록함 (= 피처 선택 기능)
- 엘라스틱넷(ElasticNet): (L1+L2) 모델. 주로 피처가 많은 데이터셋에서 적용.
- 로지스틱 회귀(Logistic Regression): 분류에 사용되는 선형 모델. 매우 강력. 일반적으로 이진 분류뿐만 아니라 희소 영역의 분류(ex-텍스트 분류)같은 영역에서 뛰어난 예측 성능을 보임

02. 단순 선형 회귀를 통한 회귀 이해

단순 선형 회귀: 독립변수 1개, 종속변수 1개

ex) Y = 주택 가격, X = 주택의 크기 로만 결정된다고 할 때

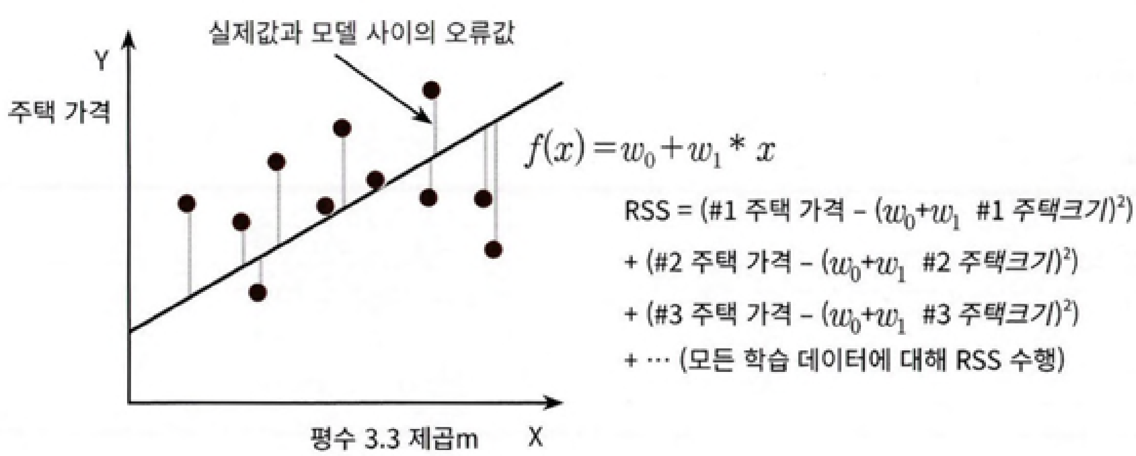


다음과 같이 선형 관계로 표현할 수 있음

예측값 Y(bar) = w0 + w1*X로 계산할 수 있음 ⇒ 회귀 계수: w0, w1

실제 주택값 = 예측값 + 오류값

⇒ 여기서 오류값을 '잔차'라고 부름. 최적의 회귀 모델은 데이터 잔차 합이 최소가 되는 모델



<오류 값 계산>

오류값은 +/- 가능하기 때문에 단순히 더하면 X

⇒ 절댓값을 취해서 더하거나(Mean Absolute Error), 오류 값의 제곱을 구해서 더하는 방식(RSS)을 취함

일반적으로 RSS 방식을 취함

$$RSS(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - (w_0 + w_1 * x_i)\right)^2$$

(i는 1부터 학습 데이터의 총 건수 N까지)

RSS는 w변수(회귀 계수)가 중심 변수

학습 데이터로 입력되는 독립변수와 종속 변수는 RSS에서 모두 상수로 간주

회귀에서 RSS는 비용(Cost), w변수(회귀 계쑤)로 구성되는 RSS를 비용 함수라고 함

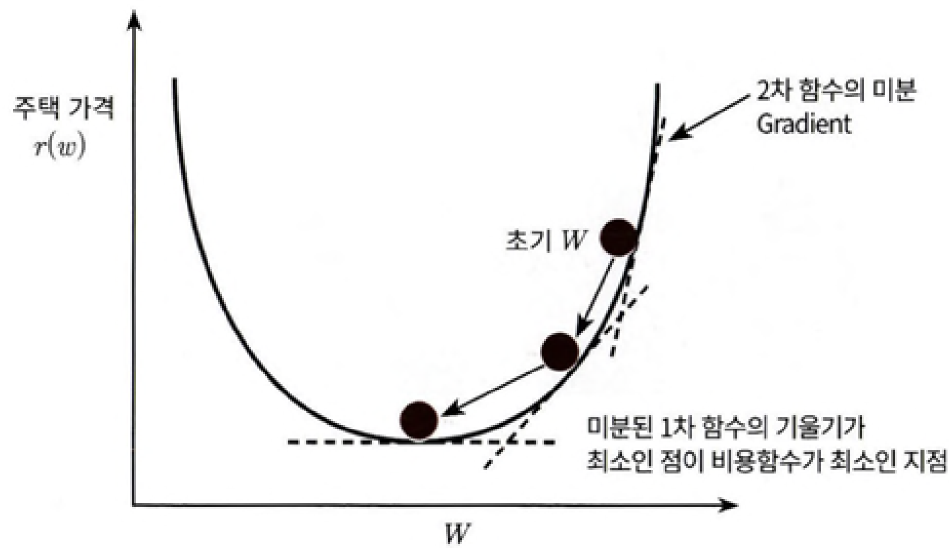
⇒ 머신러닝 회귀 알고리즘은 데이터를 계속 학습하며 이 비용함수가 반환하는 값(오류값)을 지속해서 감소시키고 최종적으로는 더 이상 감소하지 않는 최소의 오류값을 구함

(비용 함수 == 손실함수)

03. 비용 최소화하기-경사 하강법 소개

경사하강법(Gradient Descent)은 w 파라미터가 많은 경우 최소 값을 구하기 어려운 문제를 해결해주면서 비용 함수 RSS를 최소화하는 방법을 직관적으로 제공하는 방식

- 데이터를 기반으로 알고리즘이 스스로 학습한다는 머신러닝의 개념을 가능하게 한 핵심 기법의 하나
- '점진적으로' 반복적인 계산(비용함수의 반환값 계산)을 통해 w 파라미터 값을 업데이트하면서 오류 값이 최소가 되는 w 파라미터를 구하는 방식
    - 오류 값이 더 이상 작아지지 않으면 그 오류값을 최소 비용으로 판단하고 그 W 값을 최적 파라미터로 반환



비용함수가 포물선 형태의 2차 함수라면, 경사하강법은 최초 w에서 미분을 적용 한 뒤 이 미분 값이 계쏙 감소하는 방향으로 순차적으로 w를 업데이트함

더 이상 미분된 1차함수의 기울기가 감소하지 않는 지점 = 비용함수가 최소인 지점 ⇒ 이때의 w를 반환

<수식>

RSS(w0, w1)를 R(w)로 지칭하면 비용함수의 수식은 다음과 같다.

$$\mathrm{R(w)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + w_1 * x_i))^2$$

w0, w1 각각 파라미터에 대한 편미분을 적용

$$\frac{\partial R(w)}{\partial w_1} = \frac{2}{N}\sum_{i=1}^{N} -x_t * (y_i - (w_0 + w_1 x_i)) = -\frac{2}{N}\sum_{i=1}^{N} x_i * (\text{실제값}_i - \text{예측값}_i)$$

$$\frac{\partial R(w)}{\partial w_0} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (w_0 + w_1 x_i)) = -\frac{2}{N}\sum_{i=1}^{N} (\text{실제값}_i - \text{예측값}_i)$$

편미분의 결과값을 반복적으로 보정하면서 w0, w1 값을 업데이트*하면 R(w)가 최소가 되는 w1, w0 값을 구할 수 있음

*업데이트 : 새로운 w1을 이전 w1에서 편미분 결과값을 마이너스(-)하면서 적용

$$w_1 - \left(-\frac{2}{N}\sum_{i=1}^{N} x_i * \left(\text{실제값}_i - \text{예측값}_i\right) \right)$$

⇒새로운 w1에 대한 수식

이때, 편미분 값이 너무 큰 것을 고려해 보정 계수 η을 곱하는데, 이를 ‘학습률’이라고 함

$$\text{새로운 } w_0 = \text{이전 } w_0 + \eta \frac{2}{N}\sum_{i=1}^{N}\left(\text{실제값}_i - \text{예측값}_i\right)\cdot$$

(w1도 수식이 동일)

이 수식을 반복적으로 적용하며 비용함수가 최소가 되는 값을 찾음

—

일반적으로 경사하강법은 모든 학습 데이터에 대해 반복적으로 비용함수 최소화를 위한 값을 업데이트하기에 수행 시간이 매우 오래 걸림

⇒ 실전에서는 확률적 경사 하강법(Stochastic Gradient Descent) or 미니 배치 확률적 경사하강을 이용

- 확률적 경사 하강법: 일부 데이터만 이용해 w가 업데이트 되는 값을 계산하므로 빠른 속도 보장.

<feature가 여러 개인 회귀>

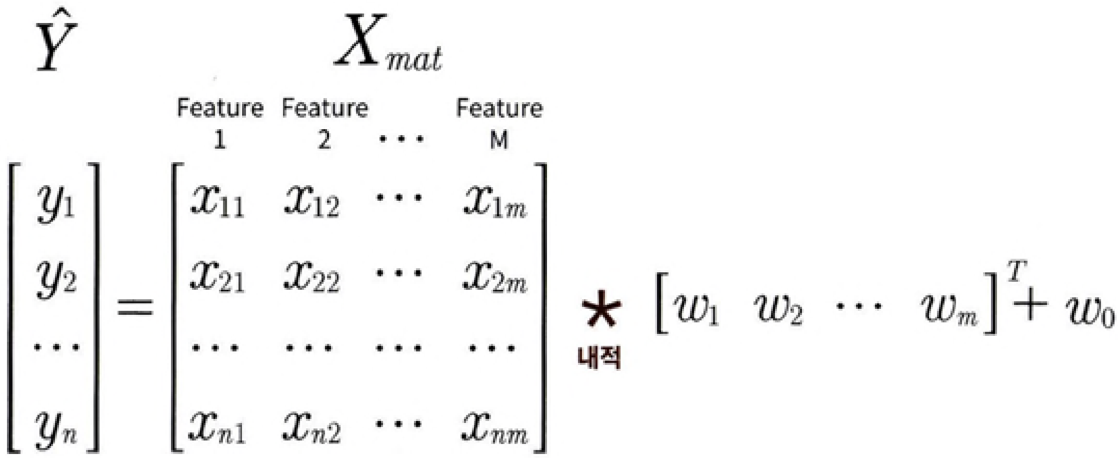
피처가 M개 있다면 회귀 계수도 (M+1)개

$$\hat{Y} = w_0 + w_1 \star X_1 + w_2 \star X_2 + \cdots + w_{100} \star X_{100}$$

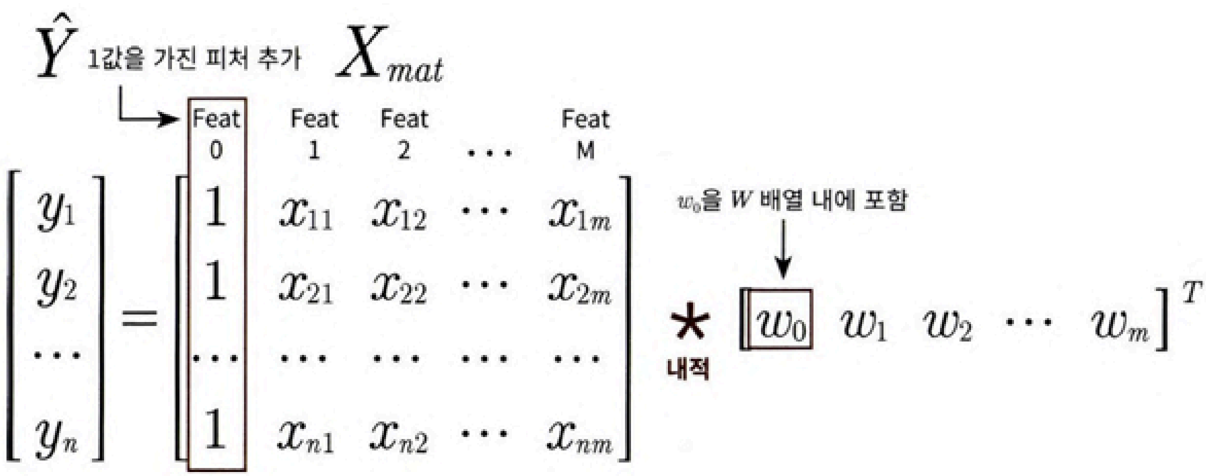
회귀 계수가 많아져도 선형대수를 이용해 간단하게 예측값 도달 가능

데이터 개수가 N이고 피처 M개의 입력 행렬을 X(mat), 회귀 계수 w1, w2, …,w100을 W 배열로 표기하면

예측 행렬 Y(hat) = np.dot(X(mat), W.T) + w0로 구할 수 있음



w0을 Weight인 배열인 W 안에 포함 시키기 위해 X(mat)의 첫 열에 모든 데이터 값이 1인 피처 feat0 추가



$$\hat{Y} = X_{mat} * W^T$$

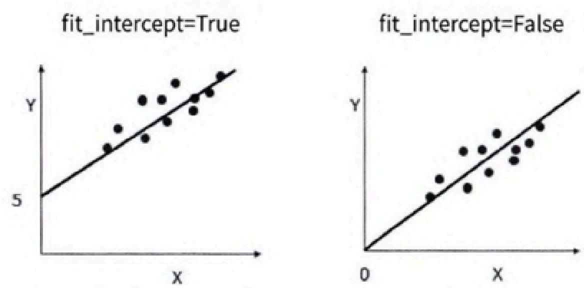
04.사이킷런 LinearRegression을 이용한 보스턴 주택 가격 예측

LinearRegression 클래스 - Ordinary Least Squares

LinearRegression 클래스는 예측값과 실제 값의 RSS(Residual Sum of Squares)를 최소화해 OLS(Ordinary Least Square) 추정 방식으로 구현한 클래

```
class sklearn.linear_model.LinearRegression(fit_intercept = True, normalize=False, copy_X = True, n_jobs = 1)
```

LinearRegression 클래스는 fit() 메서드로 X, y 배열을 입력받으면 회귀 계수(Cofficients)인 W를 coef_ 속성에 저장

입력 파라미터	**fit_intercept**: 불린 값으로, 디폴트는 True입니다. Intercept(절편) 값을 계산할 것인지 말지를 지정합니다. 만일 False로 지정하면 intercept가 사용되지 않고 0으로 지정됩니다.  **normalize**: 불린 값으로 디폴트는 False입니다. fit_intercept가 False인 경우에는 이 파라미터가 무시됩니다. 만일 True이면 회귀를 수행하기 전에 입력 데이터 세트를 정규화합니다.
속성	**coef_**: fit() 메서드를 수행했을 때 회귀 계수가 배열 형태로 저장하는 속성. Shape는 (Target 값 개수, 피처 개수). **intercept_**: intercept 값

OLS 기반의 회귀 계수 계산은 입력 피처의 독립성에 많은 영향을 받음

피처 간의 상관관계가 매우 높은 경우 분산이 매우 커져서 오류에 매우 민감해짐

⇒'다중 공선성(multi-collinearity)문제'

일반적으로 상관관계가 높은 피처가 많은 경우 독립적인 중요한 피처만 남기고 제거하거나 규제를 적용. 또한 매우 많은 피처가 다중 공선성 문제를 가지고 있다면 PCA를 통해 차원 축소를 수행하는 것도 고려해볼 수 있음

회귀 평가 지표

평가 지표	설명	수식
MAE	Mean Absolute Error(MAE)이며 실제 값과 예측값의 차이를 절댓값으로 변환해 평균한 것입니다.	$MAE = \frac{1}{n}\sum_{i=1}^{n}\lvert Yi - \hat{Y}i \rvert$
MSE	Mean Squared Error(MSE)이며 실제 값과 예측값의 차이를 제곱해 평균한 것입니다.	$MSE = \frac{1}{n}\sum_{i=1}^{n}(Yi - \hat{Y}i)^2$
RMSE	MSE 값은 오류의 제곱을 구하므로 실제 오류 평균보다 더 커지는 특성이 있으므로 MSE에 루트를 씌운 것이 RMSE(Root Mean Squared Error)입니다.	$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Yi - \hat{Y}i)^2}$
R^2	분산 기반으로 예측 성능을 평가합니다. 실제 값의 분산 대비 예측값의 분산 비율을 지표로 하며, 1에 가까울수록 예측 정확도가 높습니다.	$R^2 = \frac{\text{예측값 } Variance}{\text{실제값 } Variance}$

이 외에도 MSE나 RMSE에 로그를 적용한 MSLE(Mean Squared Log Error)와 RMSLE(Root Mean Squared Log Error)도 사용

→사이킷런은 RMSE 제공X→0.22버전부터는 제공

→ MSE를 위한 metrics.mean_squared_error() 함수를 그대로 사용하되, squared 파라미터를 False로 지정해 사용.

MSE는 사이킷런에서 mean_squared_error(실제값, 예측값, squared = True)

RMSE는 mean_squared_error(실제값, 예측값, squared = False)

평가 방법	사이킷런 평가 지표 API	Scoring 함수 적용 값
MAE	metrics.mean_absolute_error	'neg_mean_absolute_error'
MSE	metrics.mean_squared_error	'neg_mean_squared_error'
RMSE	metrics.mean_squared_error를 그대로 사용하되 squared 파라미터를 False로 설정.	'neg_root_mean_squared_error'
MSLE	metrics.mean_squared_log_error	'neg_mean_squared_log_error'
R^2	metrics.r2_score	'r2'

cross_val_score, GridSearchCV와 같은 Scoring 함수에 회귀 평가 지표를 적용 시 유의점

ex) MAE의 scoring 파라미터 값: 'neg_mean_absolute_error'와 같이 'neg'라는 접두어가 붙어있음.negative(음수) 값을 가진다는 의미인데, MAE는 절대값의 합이기에 음수X

- Scoring 함수에 음수값을 반환하는 이유: 사이킷런의 Scoring 함수가 score값이 클수록 좋은 평가 결과로 자동 평가
- But, 실제 값과 예측 값의 오류 차이를 기반으로 하는 회귀 평가 지표의 경우 값이 커지면 오히려 나쁜 모델 ⇒ -1을 원래의 평가 지표 값에 곱해 작은 오류 값이 더 큰 숫자로 인식하게 함

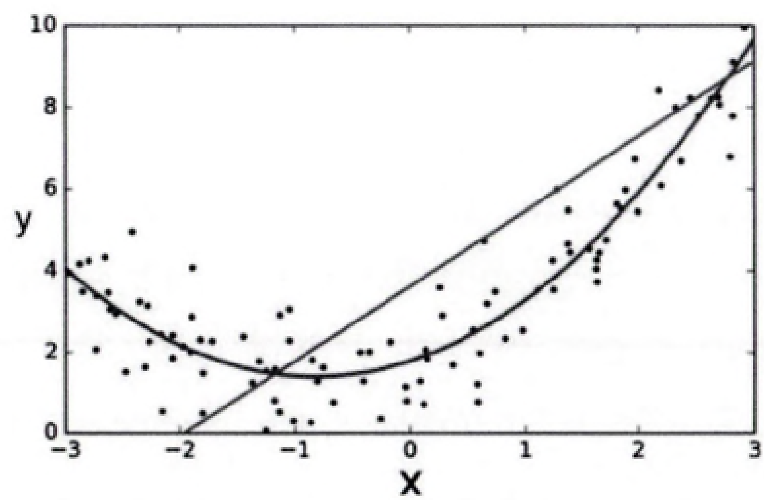
05. 다항 회귀와 과(대)적합/과소적합 이해

다항 회귀:

$$y = w0 + w1 * x1 + w2 * x2 + w3 * x1 * X2 + w4 * x1^2 + w5 * x2^2$$

다항 회귀도 선형 회귀라는 점 주의**

-회귀에서 선형 회귀/비선형 회귀를 나누는 기준은 회귀 계수가 선형/비선형인지에 따른 것이지 독립변수의 선형/비선형 여부와는 무관



〈 주어진 데이터 세트에서 다항 회귀가 더 효과적임 〉

사이킷런은 다항 회귀를 위한 클래스를 명시적으로 제공X

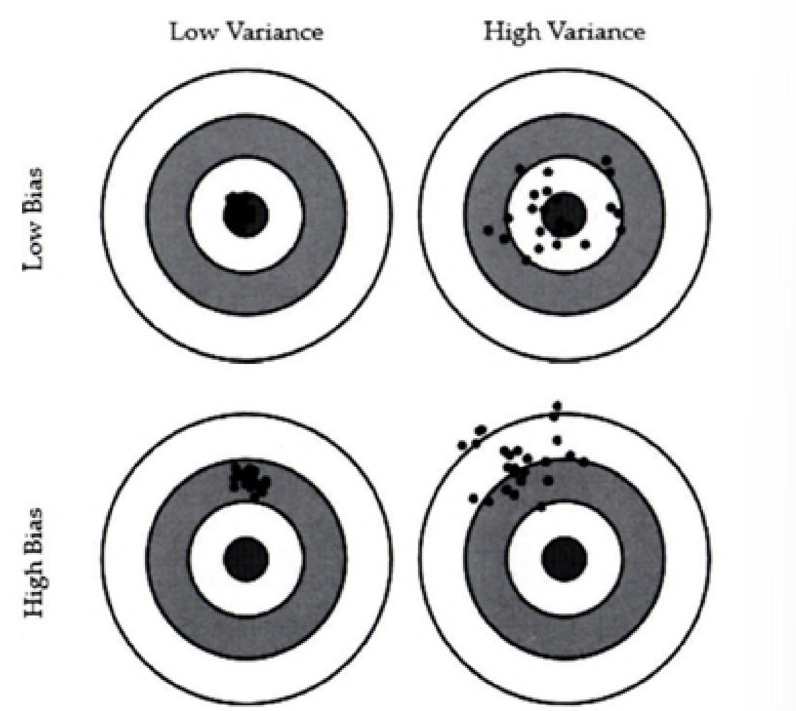
→다항 회귀 역시 선형 회귀이기에 비선형 함수를 선형 모델에 적용시키는 방법을 사용해 구현

편향-분산 트레이드오프(Bias-Variance Trade off)

편향-분산 트레이드오프는 머신러닝이 극복해야할 가장 중요한 이슈 중 하나

앞의 Degree1 ⇒ 과소적합 경향 ⇒ 고편향(High Bias)성을 가짐

반대로 Degree15 ⇒ 과적합 모델 ⇒ 지나치게 높은 변동성. 고분산(High Variance)성을 가짐



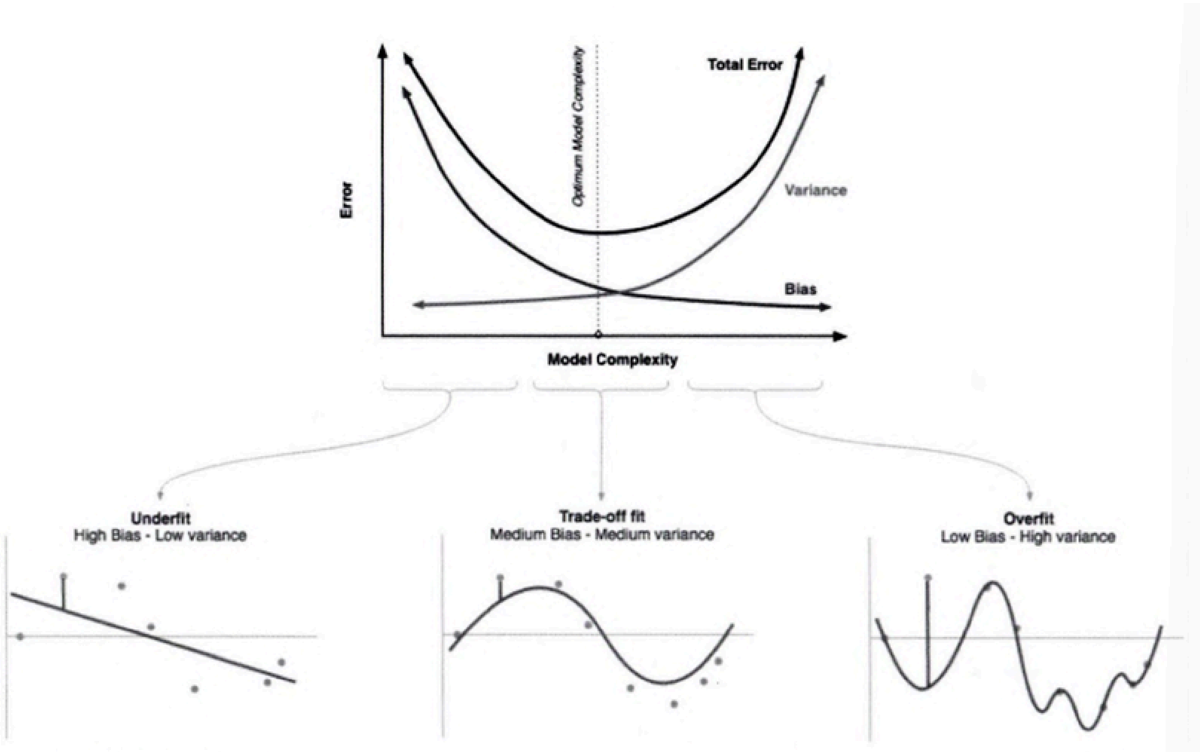
〈 편향과 분산의 고/저에 따른 표현.
http://scott.fortmann-roe.com/docs/BiasVariance.html에서 발췌 〉

- 상단 왼쪽: 저편향/저분산(Low Bias/Low Variance)은 예측 결과가 실제 결과에 매우 근접. 예측 변동 크지 않고 특정 부분에 집중돼 있는 아주 뛰어난 성능
- 상단 오른쪽: 저편향/고분산(Low Bias/High Variance)은 예측 결과가 실제 결과에 비교적 근접하지만, 예측 결과가 실제 경과를 중심으로 꽤 넓은 부분에 분포됨.
- 하단 왼쪽: 고편향/저분산(High Bias/Low Variance)은 정확한 결과에서 벗어나면서도 예측이 특정 부분에 집중돼 있음

- 하단 오른쪽: 고편향/고분산(High Bias/High Variance)은 정확한 예측 결과를 벗어나면서도 넓은 부분에 분포됨

⇒편향이 높으면 분산이 낮아지고(과소적합)

⇒분산이 높으면 편향이 낮아짐(과적합)



〈 편향과 분산에 따른 전체 오류 값(Total Error) 곡선. http://scott.fortmann-roe.com/docs/BiasVariance.html에서 발췌. 〉

⇒ 편향과 분산의 관계에 따른 전체 오류값(Total Error)의 변화

편향이 너무 높으면 전체 오류가 높음

편향을 너무 낮추면 동시에 분산이 높아지고 전체 오류도 낮아짐

→전체 오류가 가장 낮아지는 '골디락스'지점을 통과하면서 분산을 지속적으로 높이면 전체 오류값이 오히려 증가하면서 예측 성능이 다시 저하됨

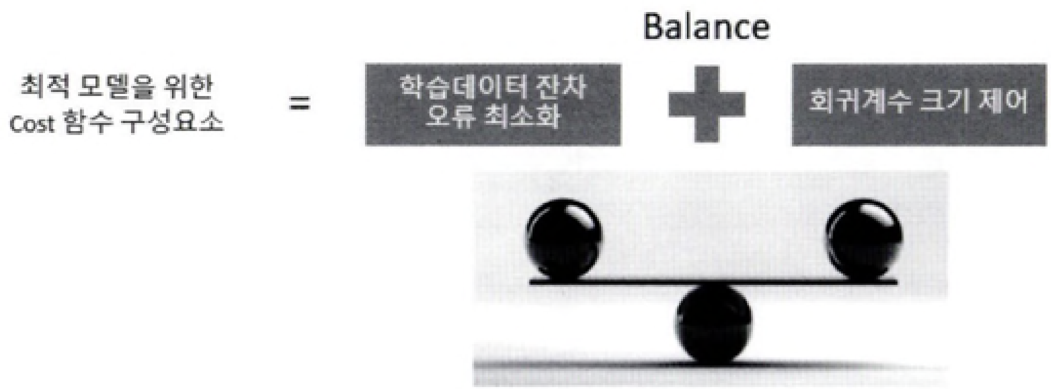
06.규제 선형 모델-릿지, 라쏘, 엘라스틱넷

규제 선형 모델의 개요

회귀 모델 -적절히 데이터에 적합하면서도 회귀 계수가 기하급수적으로 커지는 것을 제어할 수 있어야함

이전까지 선형 모델의 비용 함수는 RSS 최소화만 고려→ 학습 데이터에 지나치게 맞추게 되고, 회귀 계수가 쉽게 커짐→ 변동성이 오히려 심해져서 테스트 데이터 세트에서는 예측 성능이 저하될 수 있음

⇒ 비용 함수는 학습 데이터의 잔차 오류 값을 최소로 하는 RSS 최소화 방법과 과적합을 방지하기 위해 회귀 계수 값이 커지지 않도록 균형을 이뤄야함



⇒ 비용함수 변경

$$\textbf{비용 함수 목표} = \mathrm{Min}(\mathrm{RSS(W)} + \mathrm{alpha} * \| W \|_2^2)$$

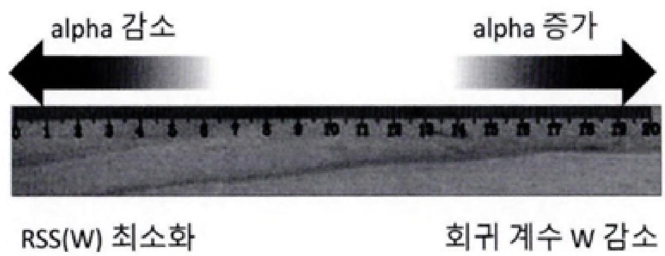
alpha: 학습 데이터 적합 정도와 회귀 계수 값의 크기 제어를 수행하는 튜닝 파라미터

→ alpha = 0 : 비용 함수 식은 기존과 동일한 Min(RSS(W) +0)

→alpha = 무한대: alpha*|W|값이 너무 커지게 됨 → W 값을 0으로 만들어야 Cost가 최소화되는 비용 함수 목표를 달성할 수 있음

alpha 값을 크게하면 비용 함수는 회귀 계수 W 값을 작게 해 과적합 개선

- alpha = 0인 경우는 W가 커도 $alpha * \|W\|_2^2$가 0이 되어 비용 함수는 Min(RSS(W))
- alpha = 무한대인 경우 $alpha * \|W\|_2^2$도 무한대가 되므로 비용 함수는 W를 0에 가깝게 최소화 해야 함.



〈 alpha 튜닝 파라미터를 통한 RSS 최소화와 회귀 계수 크기 감소의 균형 조정 〉

비용함수에 alpha 값으로 패널티를 부여해 회귀 계수 값의 크기를 감소시켜 과적합을 개선하는 방식을 '규제'

→L2 규제 : alpha*||W||와 같이 W의 제곱에 대해 패널티를 부여하는 방식 ⇒ '릿지 회귀'

→L1 규제: alpha *||W||와 같이 W의 절댓값에 패널티 부여, 영향력 크지 않은 회귀 계수 값을 0으로 변환 ⇒'라쏘 회귀'

릿지 회귀

사이킷런은 Ridge 클래스를 통해 릿지 회귀를 구현

-주요 생성 파라미터: alpha(릿지 회귀의 alpha L2 규제 계수)

선형 회귀 모델을 위한 데이터 변환

타깃값의 경우 정규 분포 형태가 아니라 특정값의 분포가 치우친 왜곡(skew)된 형태의 분포도일 경우 예측 성능에 부정적인 영향을 미칠 가능성이 높음

→ 선형 회귀 모델을 적용하기 전에 먼저 데이터에 대한 스케일링/정규화 작업을 수행하는 것이 일반적임 but! 이런 작업을 선행한다고 해서 무조건 예측 성능이 향상되는 것X

일반적으로 중요 피처들이나 타깃값의 분포도가 심하게 왜곡됐을 경우에 이러한 변환 작업을 수행

<사이킷런을 이용해 피처 데이터 세트에 적용하는 변환 작업>

1. StandardScaler 클래스를 이용해 평균이 0, 분산이 1인 표준 정규 분포를 가진 데이터 세트로 변환하거나 MinMaxScaler 클래스를 이용해 최솟값이 0이고 최댓값이 1인 값으로 정규화 수행

   → 예측 성능 향상을 크게 기대하기 어려움

2. 스케일링/정규화를 수행한 데이터 세트에 다시 다항 특성을 적용해 변환하는 방법. 보통 1번 방법을 통해 예측 성능에 향상이 없을 경우 이와 같은 방법을 적용

   → 피처의 개수 多, 다항 변환으로 생성되는 피처의 개수가 기하급수로 늘어나서 과적합 발

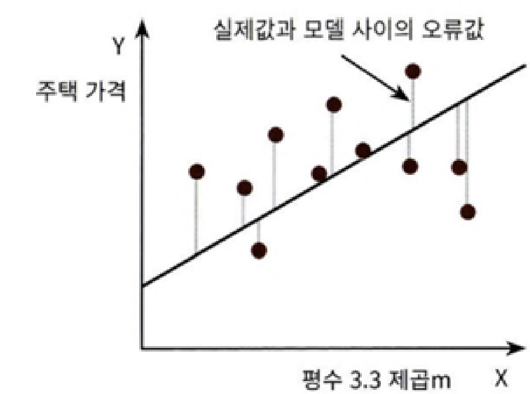
3. 원래 값에 log 함수를 적용하면 보다 정규 분포에 가까운 형태로 값이 분포됨. ⇒ 로그 변환 (log Transformation)

   제일 많이 사용됨.

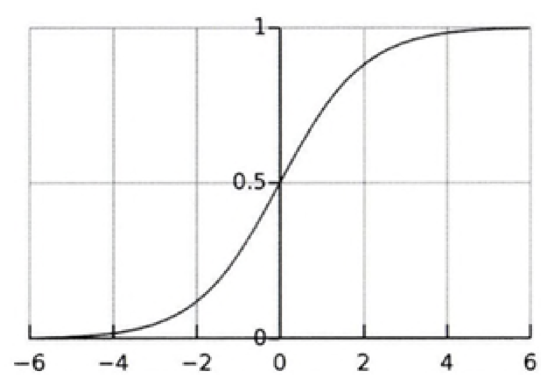
07.로지스틱 회귀

로지스틱 회귀는 선형 회귀 방식을 분류에 적용한 알고리즘

학습을 통해서 선형 함수의 회귀 최적선을 찾는 것이 아니라 시그모이드 함수* 최적선을 찾고 이 시그모이드 함수의 반환 값을 확률로 간주해 확률에 따라 분류를 결정한다는 것

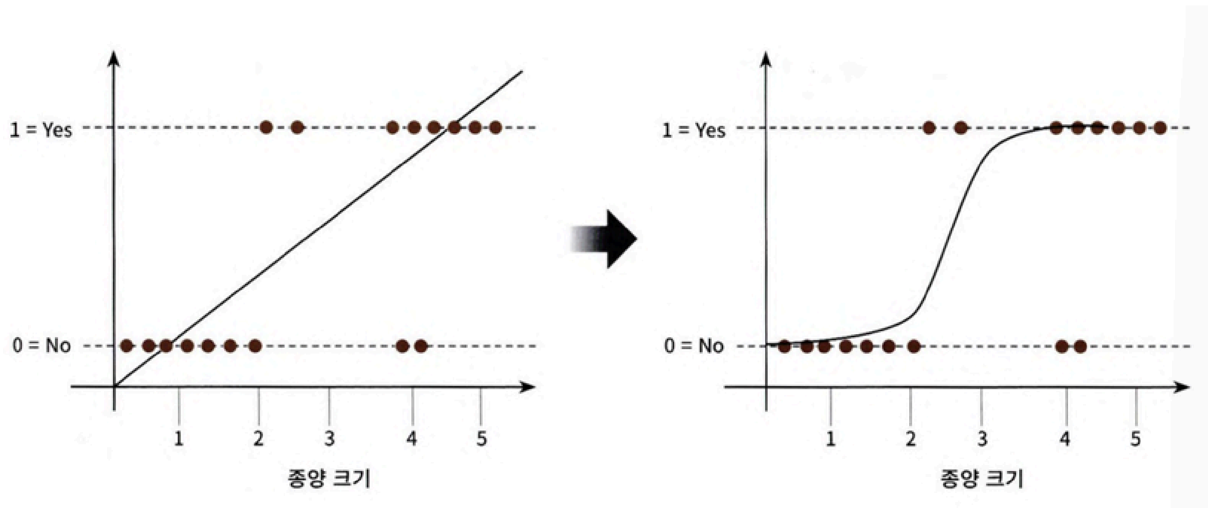


〈 선형 회귀의 선형 함수 〉



〈 로지스틱 회귀의 시그모이드 함수 〉

*시그모이드 함수 $y = 1/(1 + e^(- x))$ → 아무리 커지거나 작아져도 y값은 항상 0과 1 사이 값을 반환(x 값이 커지면 1에 근사, x값이 작아지면 0에 근사, x가 0일 때 0.5)



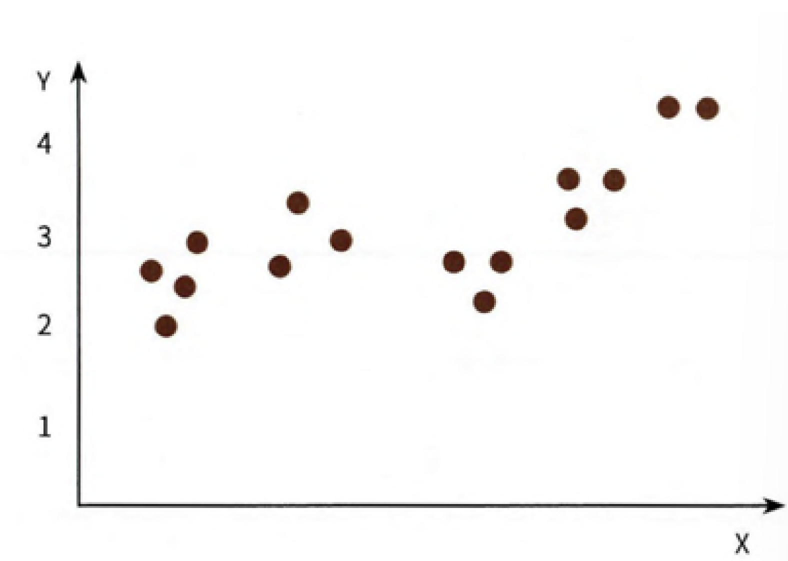
→ 시그모이드 함수를 이용하면 좀 더 정확하게 0과 1에 대해 분류를 할 수 있

08.회귀 트리

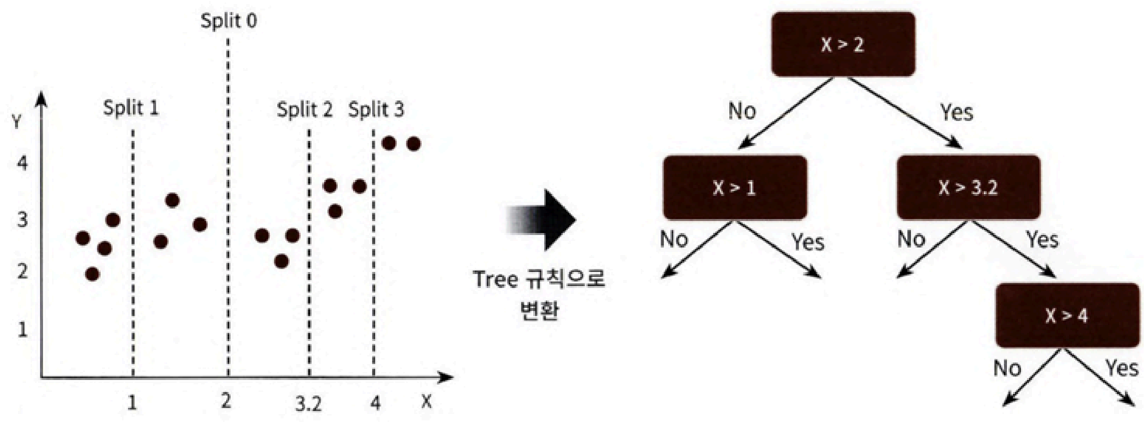
회귀함수를 기반으로 하지 않고 결정 트리와 같이 트리를 기반으로 하는 회귀 방식

트리 기반의 회귀는 회귀 트리를 이용하는 것. 즉, 회귀를 위한 트리를 생성하고 이를 기반으로 회귀 예측을 함

→분류 트리와의 차이점: 분류트리가 특정 클래스 레이블을 결정하는 것과는 달리 회귀트리는 리프 노트에 속한 데이터 값의 평균값을 구해 회귀 예측값을 계산

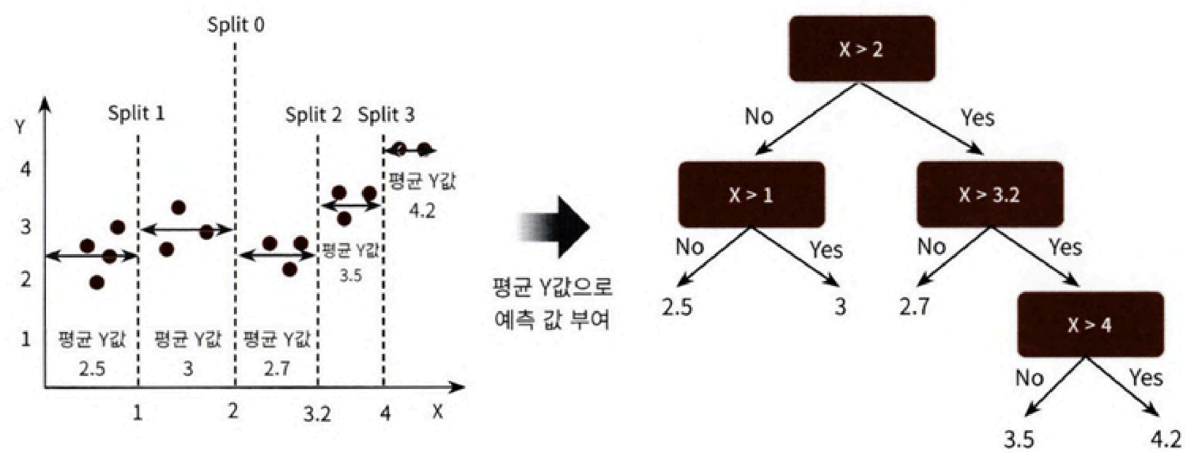


이 데이터 세트의 X 피처를 결정 트리 기반으로 분할하면 X값의 균일도를 반영한 지니계수에 따라



이처럼 분할 할 수 있음

리프 노드 생성 기준에 부합하는 트리 분할이 완료됐다면 리프 토노드에 소속된 데이터 값의 평균값을 구해서 최종적으로 리프 노드에 결정 값으로 할당



트리 생성이 CART 알고리즘*에 기반하고 있기에 결정트리, 랜덤 포레스트, GBM, XGBoost, LightBFM 등 모든 트리 기반 알고리즘은 분류뿐만 아니라 회귀도 가능

*CART(Classification And Regression Trees)는 분류와 회귀 모두 가능하게 해주는 트리 생성 알고리즘

<사이킷런의 트리 기반 회귀와 분류의 Estimator 클래스>

알고리즘	회귀 Estimator 클래스	분류 Estimator 클래스
Decision Tree	DecisionTreeRegressor	DecisionTreeClassifier
Gradient Boosting	GradientBoostingRegressor	GradientBoostingClassifier
XGBoost	XGBRegressor	XGBClassifier
LightGBM	LGBMRegressor	LGBMClassifier

지원 파일

https://colab.research.google.com/drive/1pLTN7uFKSCurVHPeRM7_mt6WZKTv8ynf?usp=sharing