파머완 05 회귀

01 회귀 소개

회귀 분석

: 데이터 값이 평균과 같은 일정한 값으로 돌아가려는 경향을 이용한 통계학 기법

ex) 부모의 키가 클 경우: 세대를 이어가면서 자식들의 키가 무한정 커지는 것이 아님

회귀란?

: 독립변수와 한 개의 종속변수 간의 상관관계를 모델링하는 기법

ex) 독립변수: 아파트의 방 개수, 방 크기, 주변 학군 등 ⇒ 종속변수: 아파트 가격

Y=W1*X1+W2*X1+...+Wn*Xn

- Y: 종속변수 ⇒ 결정 값
- X1,X2,...,Xn: 독립변수 ⇒ feature
- W1,W2,...,Wn: 회귀 계수 ⇒ feature와 결정 값 기반의 학습을 통해 최적의 회귀 계수를 찾아야함!

회귀 유형 구분

독립 변수 개수	회귀 계수의 결합
1개: 단일 회귀	선형: 선형 회귀
여러 개: 다중 회귀	비선형: 비선형 회귀

지도학습

분류(Classification)

예측 값: 이산형 클래스 값 ex) 카테고리

회귀(Regression)

예측 값: 연속형 숫자 값

선형 회귀

: 실제 값과 예측 값의 차이를 최소화하는 직선형 회귀선을 최적화하는 방식

- 오류의 제곱 값

규제방법(Regularization)

선형 회귀의 과적합 문제를 해결하기 위해 회귀 계수에 페널티 값을 적용함

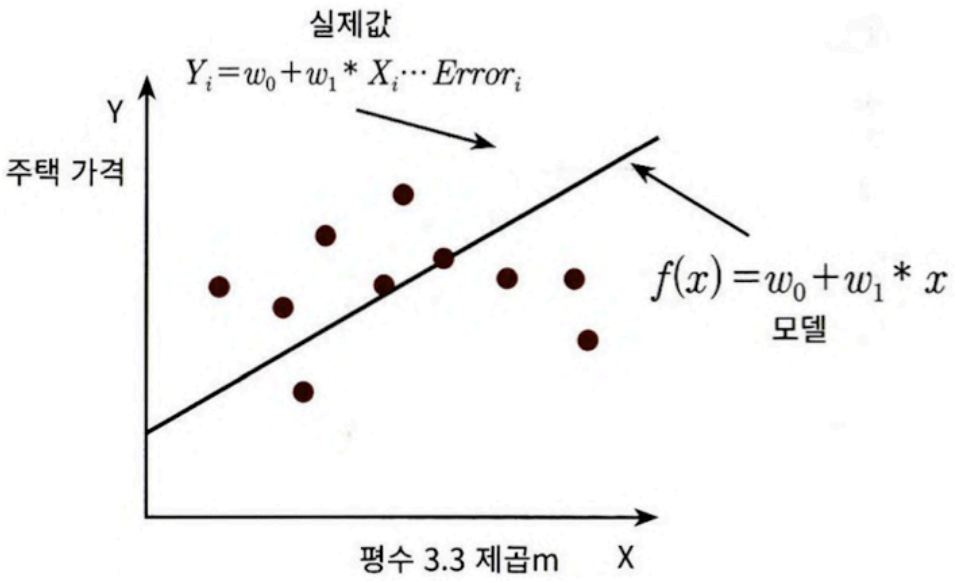
일반 선형 회귀	예측값과 실제값의 RSS를 최소화하도록 회귀 계수 최적화 (without 규제)
릿지(Ridge)	선형 회귀 + L2 규제 ; L2: 상대적으로 큰 회귀 계수 값의 예측 영향도를 감소시키기 위해 회귀 계수값을 더 작게 만드는 규제 모델
라쏘(Lasso)	선형 회귀 + L1 규제; L1: 예측 영향력이 작은 피처의 회귀 계수를 0으로 만들어 회귀 예측 시 피처가 선택되지 않게 하는 모델
엘라스틱넷 (ElasticNet)	L2 + L1 결합: L1 규제로 피처의 개수를 줄이며 L2 규제로 계수 값의 크기 조정 (피처가 많은 데이터 세트에 적용함)
로지스틱 회귀	분류에 사용되는 선형 모델

02 단순 선형 회귀를 통한 휘귀 이해

단순 선형 회귀

:독립변수 1개 + 종속변수 1개

ex) 주택의 가격이 주택의 크기로만 결정될 경우



회귀 계수: w1, w0

잔차: | 실제 값 - 회귀 모델 오류 값 |

⇒ 잔차의 합이 최소가 되는 모델을 만드는 것이 최적의 회귀 모델을 만드는 것!

오류 합 계산

1. MAE: 절댓값을 취해서 더함
2. RSS: 오류 값의 제곱을 구해서 더함 = 비용함수 = 손실 함수 (loss function)

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

03. 비용 최소화하기 = 경사 하강법(Gradient Descent) 소개

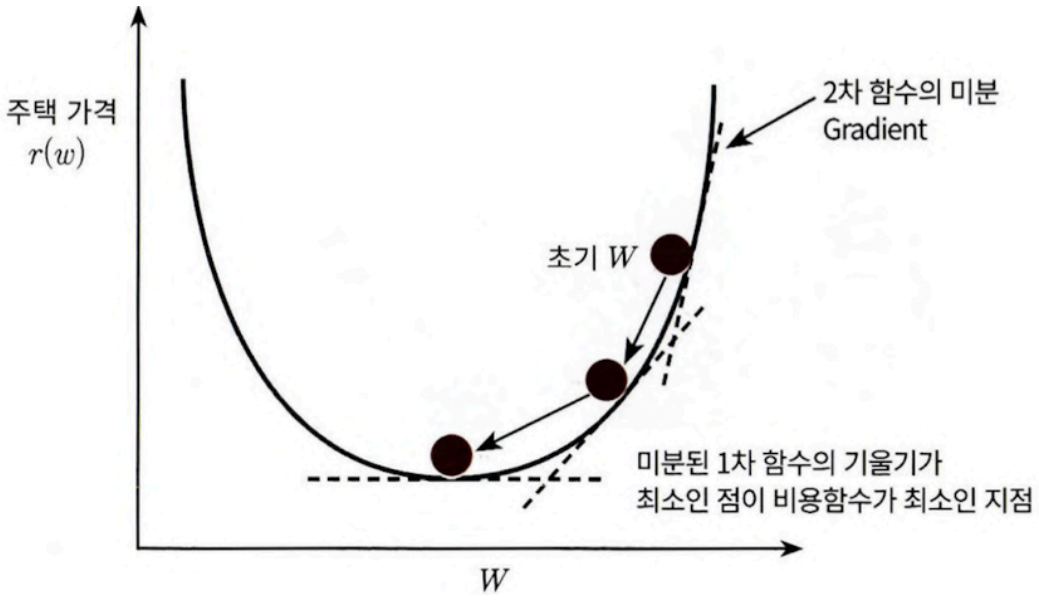
비용 함수가 최소가 되는 W 파라미터를 구하는 법?

- W 파라미터가 적을 경우: 고차원 방정식을 활용
- W 파라미터가 많을 경우: 고차원 방정식으로 해결하기 어려움 → gradient descent 사용

Gradient Descent

: 점진적으로 반복적인 계산 → W 파라미터 업데이트 → 오류 값이 최소가 되는 W 파라미터를 구하는 방식

- 예측 값과 실제 값의 차이가 작아지는 방향성으로 W 파라미터를 보정해나감



how?

1. 최초 w에서 미분 적용
2. 미분 값이 계속 감소하는 방향으로 순차적으로 w 업데이트
3. 더이상 미분된 1차 함수의 기울기가 감소 하지 않는 지점 → 비용 함수가 최소인 지점으로 설정 ⇒ 그때의 w 반환

RSS(w0,w1) 미분

: 두개의 파라미터 → 각 변수에 편미분 적용

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

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

⇒ 비용 함수 R(w)가 최소가 되는 w1,w0 값 구함

새로운 w1: 이전 w1 사용

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

새로운 w0: 이전 w0 사용

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

how?

1. w1,w0를 임의의 값으로 설정 → 첫 비용 함수 값 계산
2. w1과 w0을 업데이트 후, 다시 비용 함수 값 계산
3. 비용 함수가 감소하는 방향성으로 주어진 횟수만큼 반복하며 w1,w0 =업데이트

경사 하강법 단점

: 모든 학습 데이터에 대해 반복적으로 비용함수 최소화를 위한 값을 업데이트 함

→ 수행 시간이 매우 오래 걸림

⇒ 확률적 경사 하강법 사용!

확률적 경사 하강법(Stochastic Gradient Descent)

: 일부 데이터로 w가 업데이트되는 값을 계산함 ⇒ 속도가 빠름

- 대용량 데이터의 경우 사용 (+ 미니 배치 확률적 경사 하강법)

피처가 몇 개인가?

피처가 1개인 경우:

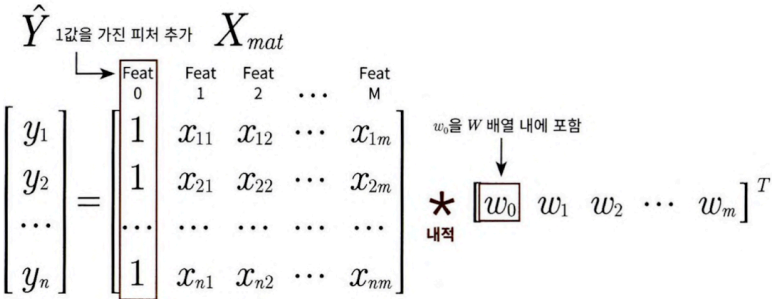
$$\hat{Y} = w_0 + w_1 * X$$

피처가 M개인 경우: 회귀계수 M+1개 도출 (w0,w1,...,w100)

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

→ 회귀 계수가 많아지더라도, 선형대수를 이용해 예측값 도출 가능!

예측 행렬 Y 구하는 식



⬇️

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

04 사이킷런 LinearRegression을 이용한 보스턴 주택 가격 예측

LinearRegression 클래스

: 예측값과 실제 값의 RSS(Residual Sum of Squares)를 최소화하여 OLS(Ordinary Least Squares) 추정 방식으로 구현한 클래스

- fit(X,y) ⇒ 회귀 계수 W를 coef_ 속성에 저장!

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

입력 파라미터

fit_intercept

- 불린 값, default=True
- intercept 값 계산 여부를 정함
- False → intercept가 사용되지 않고 0으로 지정됨

normalize

- 불린 값, default=False
- fit_intercept=False → 무시
- True → 회귀 수행 전에 입력 데이터 세트 정규화

속성

coef_

- fit() 수행 시, 회귀 계수가 배열 형태로 저장하는 속성
- Shape: (Target 값 개수, 피처 개수)

intercept_

- intercept 값

다중공선성(multi-collinearity) 문제

OLS 기반 회귀 계수 계산 → 입력 피처의 독립성에 많은 영향을 받음

ex) 피처 간 상관관계가 매우 높은 경우 → 분산이 매우 커져서 오류에 민감해짐

💡 solution?

상관관계가 높은 피처가 많을 경우: 독립적인 중요한 피처만 남기고 제거 or 규제

매우 많은 피처가 문제를 가지고 있을 경우: PCA를 통해 차원 축소

회귀 평가 지표

: 실제 값과 회귀 예측값의 차이 값을 기반으로 지표를 설정함

평가 지표	설명
MAE	Mean Absolute Error: 실제 값과 예측 값의 차이를 절댓값으로 변환해 평균함
MSE	Mean Squared Error: 실제 값과 예측값의 차이를 제곱해 평균함
RMSE	MSE에 루트를 씌워, 실제 오류 평균보다 더 커지는 특성을 방지함
R^2	분산 기반으로 예측 성능을 평가함; 예측값 분산 / 실제값 분산 → 1에 가까울수록 정확도가 높음

+) MSLE(log 적용), RMSLE(root + log 적용)

사이킷런 평가 지표

평가 방법	API	Scoring 함수 적용 값
MAE	metrics.mean_absolute_error	‘neg_mean_absolute_error’
MSE	metrics.mean_sqaured_error	‘neg_mean_squared_error’

평가 방법	API	Scoring 함수 적용 값
RMSE	metrics.mean_squared_error를 그대로 사용하되, squared=False 설정	'neg_root_mean_squared_error'
MSLE	metrics.mean_squared_log_error	'neg_mean_squared_log_error'
R^2	metrics.r2_score	'r2'

음수값을 반환하는 이유?

- Scoring 함수: score가 클수록 좋은 평가 결과를 평가함

  → 회귀 평가 지표 값이 커질 경우 보정이 필요함
- 10 > 1 ⇒ -1 > -10 로 1이 더 좋은 값임!

---

05 다항 회귀와 과(대)적합/과소적합 이해

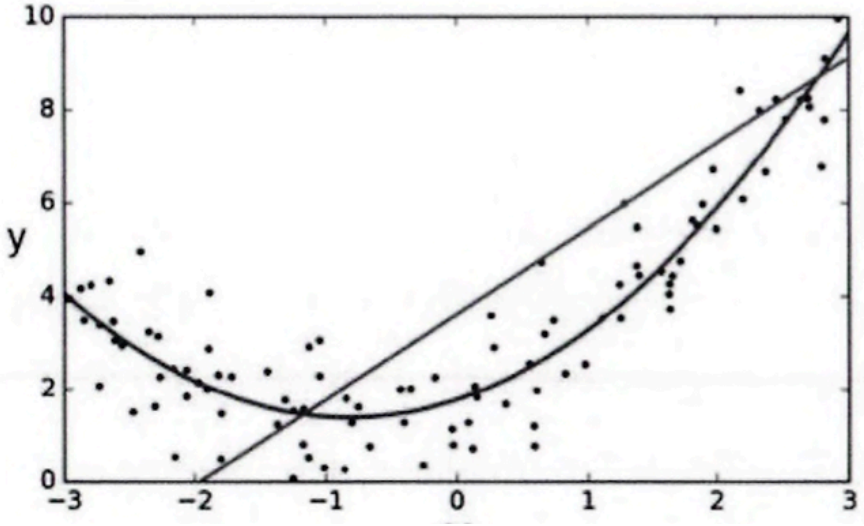
다항 회귀

: 독립변수의 단항식이 아닌 2차, 3차 방정식과 같은 다항식으로 표현되는 회귀

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

다항 회귀는 선형 회귀이다 ⭐

Z = [x1,x2,x1*x2,x1^2,x2^2] → y=w0 + w1*z1 + w2*z2 + ... 표현 가능



선형 회귀 직선형보다 다항 회귀 곡선형이 예측 성능이 높음

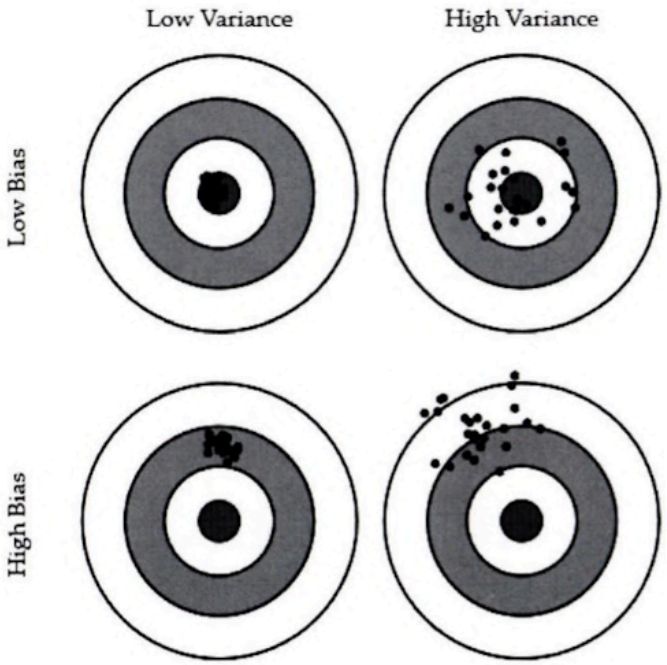
다항 회귀를 이용한 과소적합 및 과적합 이해

: 다항 회귀의 차수가 높아질수록

→ 학습 데이터에만 맞춘 학습이 이루어져 테스트 데이터에서는 정확도가 떨어지는 과적합 문제 발생

편향 - 분산 트레이드 오프

- 매우 단순화된 모델이 지나치게 한 방향성으로 치우칠 경우 ⇒ 고편향성
- 학습 데이터의 특성을 민감하게 반영한 매우 복잡한 모델이 지나치게 높은 변동성을 가질 경우 ⇒ 고분산성



저편향/저분산 (2사분면)

- 예측 결과가 실제 결과와 매우 근접함
- 예측 변동이 특정 부분에 집중돼있음 → 성능이 뛰어남!

저편향/고분산 (1사분면)

- 예측 결과가 실제 결과가 근접함
- 예측 결과가 실제 결과를 중심으로 넓은 부분에 분포되어 있음

고편향/저분산 (3사분면)

- 정확한 결과에서 벗어남
- 예측이 특정 부분에 집중됨

고편향/고분산 (4사분면)

- 정확한 예측 결과를 벗어남
- 넓은 부분에 분포됨

편향과 분산은 한쪽이 높으면, 한쪽이 낮아지는 경향!

편향이 높을 경우 → 분산이 낮아짐 (과소적합)

분산이 높을 경우 → 편향이 낮아짐 (과적합)

편향이 너무 높을 경우 ⇒ 전체 오류가 높아짐

편향을 낮추면, 분산이 높아지고 전체 오류도 낮아짐 → 골디락스 지점

- 골디락스 지점을 지나치고 지속적으로 분산을 높일 경우 전체 오류가 오히려 증가함

---

06 규제 선형 모델 - 릿지, 라쏘, 엘라스틱넷

규제 선형 모델의 개요

좋은 회귀 모델?

1. 적절히 데이터에 적합하면서
2. 회귀 계수가 기하급수적으로 커지는 것을 제어!

RSS를 최소화하는 것만 고려하면: 학습 데이터에 지나치게 맞추게 되고 회귀 계수가 쉽게 커짐

⇒ 테스트 데이터 예측 성능 저하

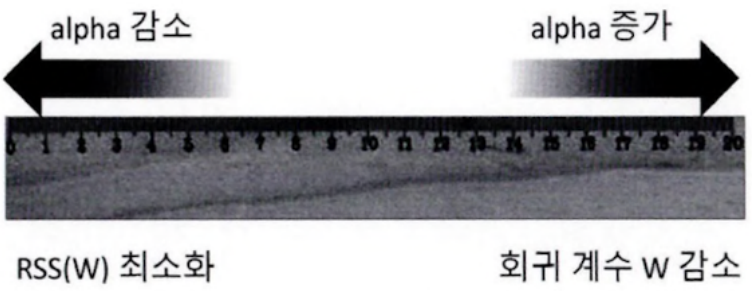
RSS 최소화 방법 + 회귀 계수 크기 제어 균형을 이뤄야함

비용 함수 목표 = Min(RSS(W) + alpha * ||W||2)

alpha?

: 학습 데이터 적합 정도 & 회귀 계수 값을 제어하는 튜닝 파라미터

- alpha가 클 경우: 비용함수는 회귀 계수 W 값을 작게 하여 과적합 개선
- alpha가 작을 경우: 회귀 계수 W 값이 커져도 상쇄 가능하므로 학습 데이터 적합 개선



규제

: 비용함수에 alpha로 페널티를 부여하여 회귀 계수 값의 크기를 감소시켜 과적합을 개선하는 방식

Ridge 회귀(L2)

: W의 제곱에 대해 페널티 부여

→ 회귀 계수를 0으로 만들지는 않음

Lasso 회귀(L1)

: W의 절댓값에 대해 페널티 부여

→ 영향력이 크지 않은 회귀 계수 값을 0으로 변환함

ElasticNet 회귀(L1+L2)

Lasso: 상관관계가 높은 피처들의 경우, 이들 중에서 중요 피처만을 셀렉하고 다른 피처들을 모두 회귀 계수를 0으로 만드는 성향이 강함

→ alpha에 따라 회귀 계수 값이 급격히 변동할 가능성 o

⇒ 이를 완화하고자 L2 규제를 추가함

단점: 수행시간이 오래걸림

정의: a*L1 + b*L2 (a: L1의 alpha, b: L2의 alpha)

주요 파라미터	정의	
alpha	a+b	1일 경우: L1
l1_ratio	a/(a+b)	0일 경우: L2

선형 회귀 모델을 위한 데이터 변환

선형 회귀 모델

: 피처와 타깃값 간에 선형 관계가 있다고 가정 → 최적의 선형함수를 찾아내 결과 예측

- 피처값과 타깃값의 분포가 정규 분포 형태를 선호함
- 왜곡된 형태일 경우, 예측 성능에 부정적인 영향을 미침

피처 데이터 세트에 적용하는 변환 작업

1. Scaler 클래스
- StandardScaler 클래스: 평균이 0, 분산이 1인 표준 정교 분포를 가진 데이터 세트로 변환
- MinMaxScaler 클래스: 최솟값이 0, 최댓값이 1인 값으로 정규화
2. 스케일링/정규화한 데이터 세트에 다항 특성을 적용하여 변환
3. 로그 변환: 원래 값에 log 함수 적용 → 정규 분포에 가깝게 분포됨

⭐ Scaler를 사용한 변환은 성능이 크게 향상되지 않고

다항 특성을 적용한 경우는 과적합의 문제가 발생할 수 있기 때문에

로그 변환을 많이 씀!

07 로지스틱 회귀

가중치 변수가 선형인지 아닌지에 따라 선형/비선형 회귀가 결정됨

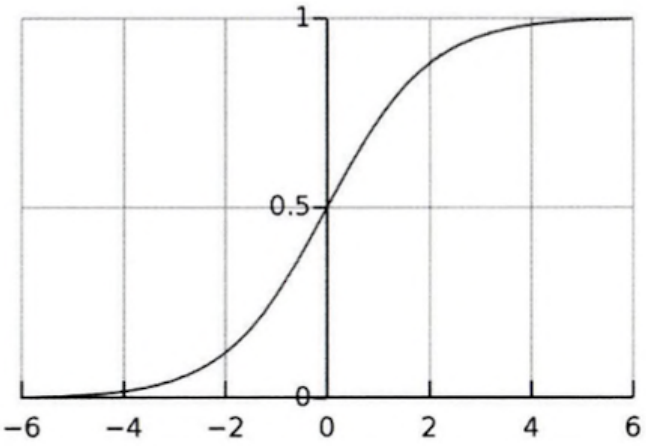
⇒ 로지스틱 회귀: 분류에 사용되는 선형 회귀 계열

로지스틱 회귀

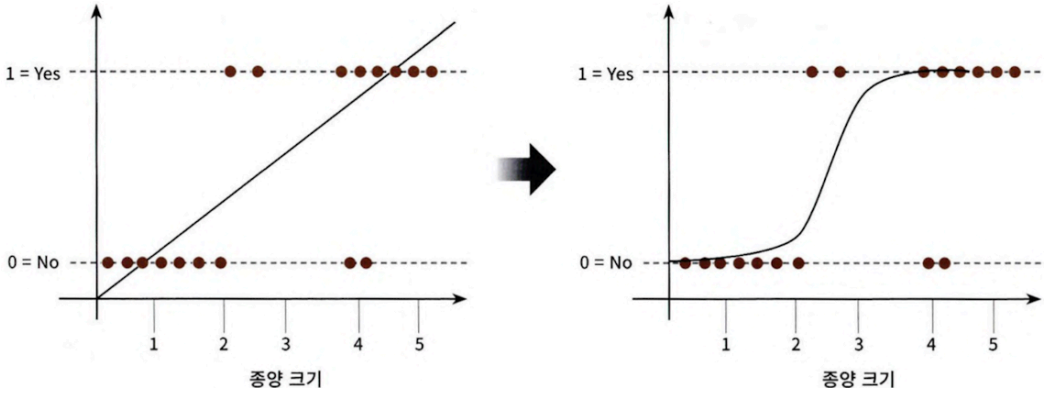
: 선형 함수가 아닌 Sigmoid 함수 최적선을 찾고,

이 함수의 반환 값을 확률로 간주하여 확률에 따라 분류를 결정함!

- 시그모이드 함수: y=1/(1+e^-x)



⇒ |x|가 커져도 항상 0과 1 사이의 값을 반환함



로지스틱 회귀 파라미터

solver 파라미터

lbfgs	기본 설정값, 메모리 공간 절약 가능, 병렬 수행 가능
liblinear	다차원이고 작은 데이터 세트에서 효과적 but 국소 최적화 이슈 & 병렬 불가능
newton-cg	더 정교한 최적화 가능 but 느림
sag	경사 하강법 기반의 최적화, 대용량의 데이터에서 빠름
saga	sag와 유사, L1 정규화 가능

max_iter 파라미터

: solver로 지정된 알고리즘이 최적 수렴할 수 있는 최대 반복 횟수

penalty 파라미터

: 규제의 유형 설정

C 파라미터

: alpha 값의 역수

---

08 회귀 트리

선형 회귀	비선형 회귀
회귀 계수의 관계를 모두 선형으로 가정하는 방정식	비선형 회귀 함수를 통해 결괏값 예측
회귀 계수를 선형으로 결합하는 회귀 함수를 구해, 독립변수를 입력하여 결괏값 예측	회귀 계수의 결합이 비선형!

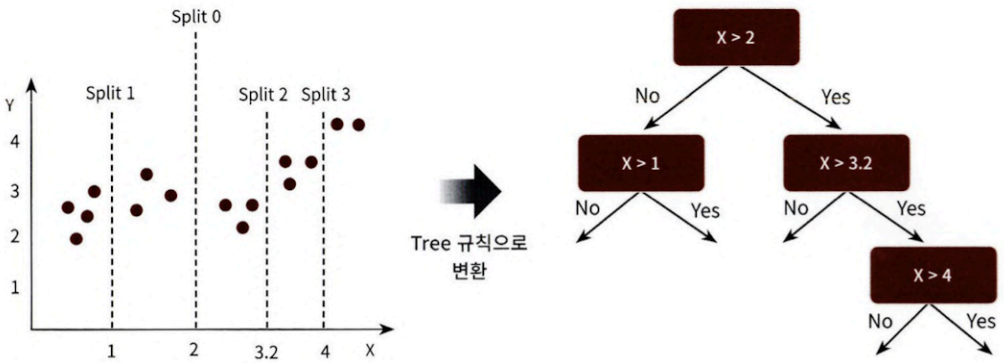
⇒ 머신 러닝 기반 회귀의 목표: 회귀 계수를 기반으로 하는 최적 회귀 함수를 도출하는 것

회귀 트리

: 회귀를 위한 트리를 생성하고 이를 기반으로 회귀 예측을 함

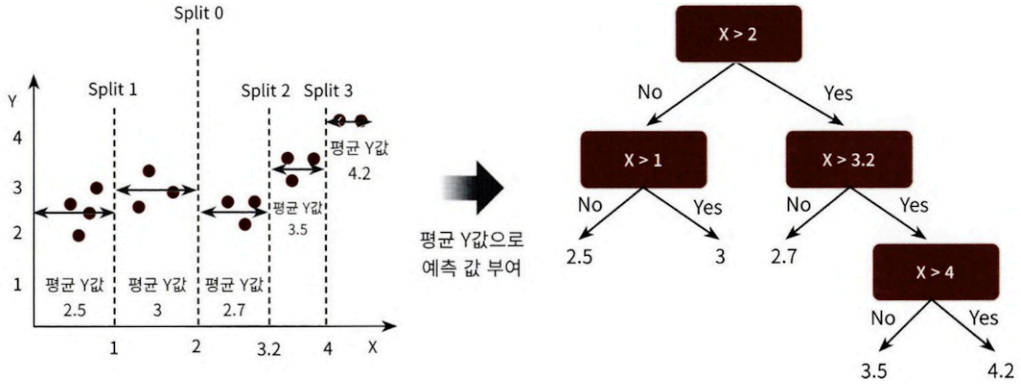
- 리프 노드에 속한 데이터 값의 평균값을 구해 회귀 예측값을 계산

1. 결정 트리 기반으로 분할하기



- X값의 균일도를 반영한 지니 계수에 따라 분할
- 루트 노드를 split 0 기준으로 분할하고 재귀적으로 split 3 노드까지 분할하여 트리 규칙으로 변환

2. 리프 노드에 결정 값 할당



- 리프 노드에 소속된 데이터 값의 평균값으로 결정 값 할당