Euron_ML_Week8

생성일	@November 7, 2024 1:52 PM

5. 회귀

01. 회귀 소개

💡 데이터 값이 평균과 같은 일정한 값으로 돌아가려는 경향을 이용한 통계학 기법
여러 개의 독립변수와 한 개의 종속변수 간의 상관관계를 모델링하는 기법

> 머신러닝 관점에서 보면..
>
> 독립변수→ 피처
>
> 종속변수→ 결정값
>
> ⇒ 머신러닝 회귀 예측: 주어진 피처와 결정 값 데이터 기반에서 학습을 통해 **최적의 회귀 계수**를 찾아냄

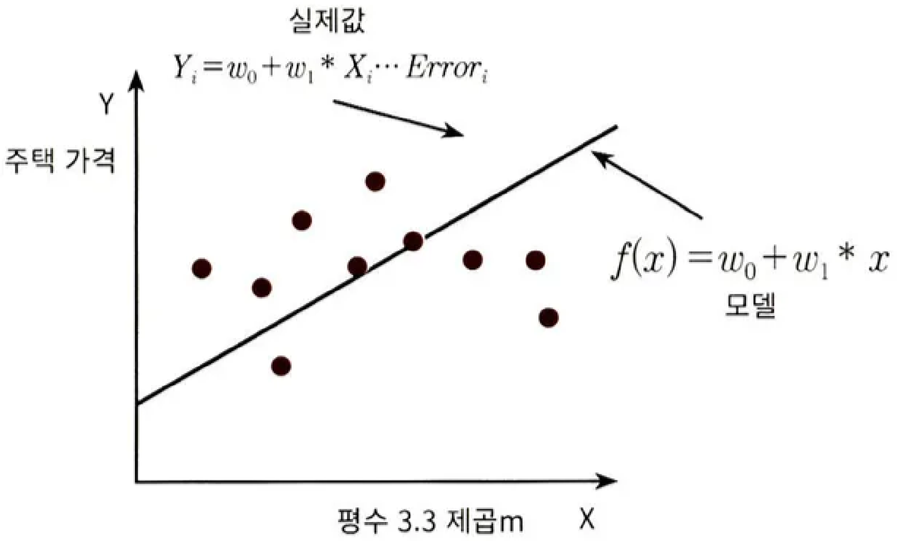
독립변수 개수	회귀 계수의 결합
1개: 단일 회귀	선형: 선형 회귀
여러 개: 다중 회귀	비선형: 비선형 회귀

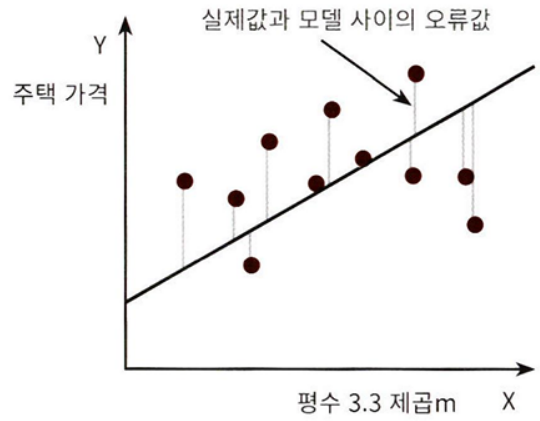
- 선형 회귀
  : 실제 값과 예측값의 차이(오류의 제곱 값)를 최소화하는 직선형 회귀선을 최적화하는 방식
  : 여러 가지 회귀 중 가장 많이 사용됨
  - 규제 방법에 따라 별도의 유형으로 나뉠 수 있는데,
    - 일반 선형 회귀: 규제 적용 x
    - 릿지: 선형 회귀에 L2 규제를 추가한 회귀 모델
    - 라쏘: 선형 회귀에 L1 규제를 적용한 회귀 모델
    - 엘라스틱넷: L2, L1 규제를 함께 결합한 모델
    - 로지스틱 회귀: 분류에 사용되는 선형 모델(매우 강력한 분류 알고리즘)

---

02. 단순 선형 회귀를 통한 회귀 이해

💡 독립변수도 하나, 종속변수도 하나인 선형 회귀





잔차= 실제 값과 회귀 모델 차이에 따른 오류값(남은 오류)

기울기 w1과 절편 w0을 회귀 계수로 지정

- 최적의 회귀 모델을 만든다

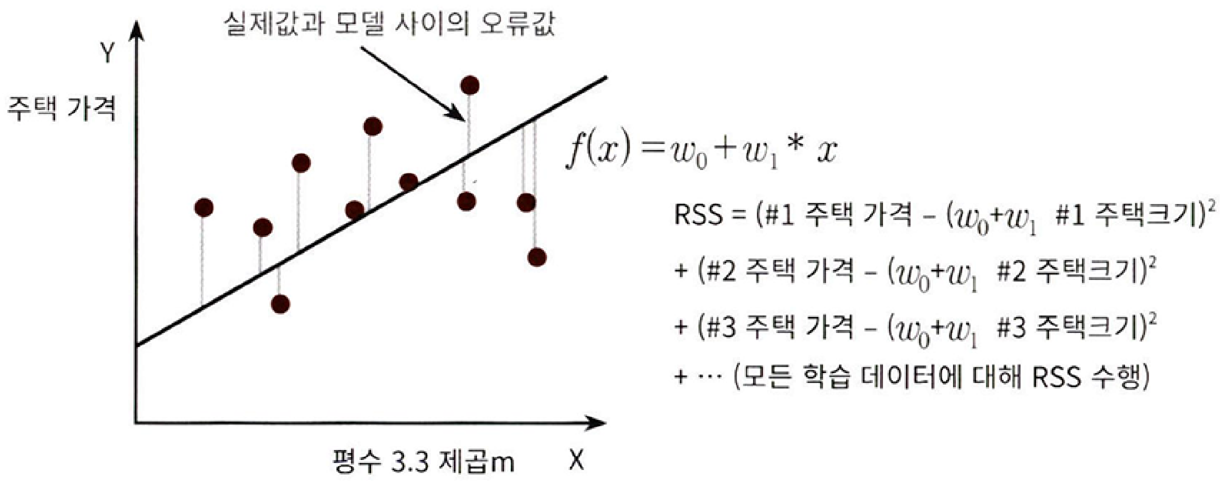
  = 데이터의 잔차(오류 값) 합이 최소가 되는 모델을 만든다.

  = 오류 값 합이 최소가 될 수 있는 최적의 회귀 계수를 찾는다.

- 오류 값이 +,- 모두 될 수 있기 때문에

  → 절댓값을 취해 더하거나 (Mean Absolute Error)

  → 오류 값의 제곱을 구해서 더함 (RSS, Residual Sum of Square) * 미분 등의 계산 시 편리



RSS는 변수가 w0, w1인 식으로 표현 가능

- 머신러닝 기반 회귀의 핵심 사항

  : RSS를 최소로 하는 회귀 계수(w0, w1)를 학습을 통해 찾는 것

  : RSS의 중심 변수는 회귀식의 독립변수 X, 종속변수 Y가 아니라 **w 변수(회귀계수)** 이다!

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

[i는 1부터 학습 데이터의 총 건수 N까지]

- 머신러닝 회귀 알고리즘

  : RSS는 비용

  : w변수(회귀변수)로 구성되는 RSS는 비용 함수

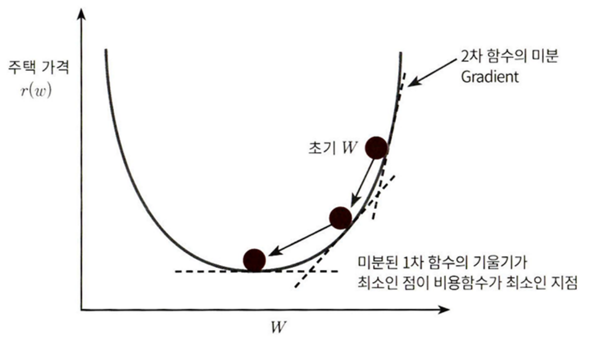
→ 데이터를 계속 학습하면서 비용 함수가 반환하는 값(오류값)을 지속해서 감소시키고 최종적으로는 더 이상 감소하지 않는 최소의 오류 값을 구하는 것

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

[비용 함수가 최소가 되는 W파라미터 구하기]

> 💡 경사하강법(Gradient Descent)
>
> : '점진적으로' 반복적인 계산을 통해 W파라미터 값을 업데이트하면서 오류 값이 최소가 되는 W 파라미터를 구하는 방식
>
> : 지속해서 오류를 감소시키는 방향으로 W 값을 계속 업데이트→ 오류 값이 더 이상 작아지지 않으면 그 오류 값을 최소 비용으로 판단→ 그 때의 W값을 최적 파라미터로 반환

- 그렇다면 어떻게 오류가 작아지는 방향으로 W값을 보정할 수 있을까?



미분된 1차 함수의 기울기가 감소하지 않는 지점= 비용 함수가 최소인 지점으로 간주

RSS를 R(w)로 지칭

- R(w)를 미분해서 미분 함수의 최솟값을 구해야 하는데, R(W)는 두 개의 파라미터 (w0,w1)를 가지고 있어 일반적인 미분 적용 불가

  → w0, w1 각 변수에 편미분 적용 (각각 r(w)를 w0, w1으로 순차적으로 편미분 수행)

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

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

- 위 식의 결괏값인

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

을 반복적으로 보정하면서 w0,w1 값을 업데이트

업데이트는 새로운 w1을 이전w1에서 편미분 결괏값을 마이너스하면서 적용

즉, 새로운 w1

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

- 위 편미분 값이 너무 클 수 있기 때문에 보정계수를 곱해줌

[경사하강법의 일반적인 프로세스]

- Step 1: w_1, w_0를 임의의 값으로 설정하고 첫 비용 함수의 값을 계산합니다.
- Step 2: w_1을 $w_1 + \eta\frac{2}{N}\sum_{i=1}^{N} x_i * (\text{실제값}_i - \text{예측값}_i)$, w_0을 $w_0 + \eta\frac{2}{N}\sum_{i=1}^{N} (\text{실제값}_i - \text{예측값}_i)$으로 업데이트한 후 다시 비용 함수의 값을 계산합니다.
- Step 3: 비용 함수의 값이 감소했으면 다시 Step 2를 반복합니다. 더 이상 비용 함수의 값이 감소하지 않으면 그때의 w_1, w_0를 구하고 반복을 중지합니다.

- 피처가 여러 개인 경우

: 1개인 경우를 확장해 유사하게 도출

$\hat{Y} = w_0 + w_1 * X_1 + w_2 * X_2 + \cdots + w_{100} * X_{100}$과 같이 예측 회귀식을 만들 수 있습니다.

- 예측 행렬 y_pred

  : 기존→ np.dot(X, w1.T) + w0

  : `y_pred = np.dot(Xmat, w.T) + w0`

$$\underset{\hat{Y}}{\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}} = \underset{X_{mat}}{\begin{array}{cccc} \text{Feature 1} & \text{Feature 2} & \cdots & \text{Feature M} \\ \hline x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{array}} \underset{\text{내적}}{*} \begin{bmatrix} w_1 & w_2 & \cdots & w_m \end{bmatrix}^T + w_0$$

w_0를 Weight의 배열인 W안에 포함시키기 위해서 Xmat의 맨 처음 열에 모든 데이터의 값이 1인 피처 Feat 0을 추가하겠습니다. 이제 회귀 예측값은 $\hat{Y} = X_{mat} * W^T$와 같이 도출할 수 있습니다.

1값을 가진 피처 추가

$$\underset{\hat{Y}}{\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}} = \underset{X_{mat}}{\begin{array}{ccccc} \text{Feat 0} & \text{Feat 1} & \text{Feat 2} & \cdots & \text{Feat M} \\ \hline \boxed{1} & x_{11} & x_{12} & \cdots & x_{1m} \\ \boxed{1} & x_{21} & x_{22} & \cdots & x_{2m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \boxed{1} & x_{n1} & x_{n2} & \cdots & x_{nm} \end{array}} \underset{\text{내적}}{*} \begin{bmatrix} \boxed{w_0} & w_1 & w_2 & \cdots & w_m \end{bmatrix}^T$$

w_0을 W 배열 내에 포함

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

04. 사이킷런 LinearRegression을 이용한 보스턴 주택 가격 예측

- LinearRegression클래스- Ordinary Least Squares

> 예측값과 실제 값의 RSS를 최소화해 OLS(Ordinary Least Squares) 추정 방식으로 구현한 클래스

> LinearRegression 클래스: fit() 메서드

> X, y 배열을 입력 받으면 회귀 계수 (Coefficients)인 W를 coef_속성에 저장

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

입력 파라미터	
fit_intercept	절편 계산 유무 (default=True) False 지정 시 intercept가 사용되지 않고 0으로 지정
normalize	데이터 세트 정규화 (default=False) fit_intercept=False인 경우, 이 파라미터 무시

속성	
coef_	회귀 계수의 배열
intercept_	intercept 값

- 회귀 평가 지표

평가 지표	설명	사이킷런 평가 지표 API	Scoring 함수 적용 값
MAE (Mean Absolute Error)	실제 값과 예측값의 차이의 절댓값의 평균	metrics.mean_absolute_error	'neg_mean_absolute_error'
MSE (Mean Squared Error)	실제 값과 예측값의 차이의 제곱의 평균	metrics.mean_squared_error	'neg_mean_squared_error'
RMSE (Root Mean Squared Error)	MSE의 루트값	RMSE를 제공x→ MSE에 제곱근을 씌워 계산하는 함수를 직접 만들어야 함	
R^2	실제 값의 분산 대비 예측값의 분산 비율 * 1에 가까울수록 예측 정확도 높음	metrics.r2_score	'r2'

💡 **Scoring 함수에 음수값을 반환하는 이유(neg=negative)**

사이킷런의 Scoring 함수가 score값이 클수록 좋은 평가 결과로 자동 평가하기 때문

but 회귀 평가 지표의 경우 값이 커지면 오히려 나쁜 모델이라는 의미이기에 사이킷런의 Scoring 함수에 반영하려면 보정이 필요한 것

'neg_mean_absolute_error'의 의미

: -1 * metrics.mean_absolute_error()

- LinearRegression을 이용해 보스턴 주택 가격 회귀 구현

05. 다항 회귀와 과(대)적합/과소적합 이해

- 다항 회귀 이해

💡 회귀가 독립변수의 단항식이 아닌 2차, 3차 방정식과 같은 다항식으로 표현되는 것

> 주의: 다항 회귀는 선형 회귀 (독립변수의 선형/비선형 여부와는 무관)

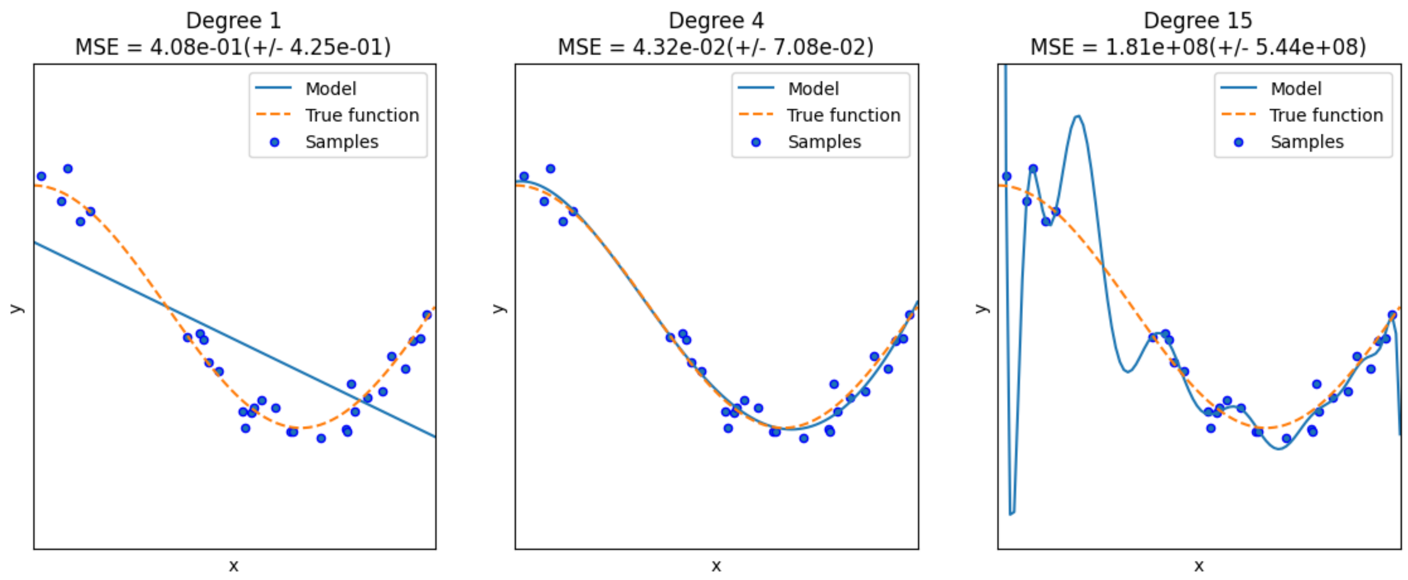
Colab으로 예제 실행

- 다항 회귀를 이용한 과소적합 및 과적합 이해

다항 회귀의 차수를 높일수록 학습 데이터에만 맞춘 학습이 이뤄짐

→ 테스트 데이터 환경에서 예측 정확도가 떨어짐, 즉 **차수가 높아질수록 과적합 문제 발생**

Colab으로 예제 실행



1. 단순한 직선, 단순 선형 회귀와 똑같음

   실제 데이터 세트인 코사인 데이터 세트를 직선으로 예측하기에 너무 단순

   예측 곡선이 학습 데이터의 패턴을 제대로 반영하지 못하고 있는 과소적합 모델

2. 실제 데이터 세트와 유사, 학습 데이터 세트를 비교적 잘 반영

   코사인 곡선 기반으로 테스트 데이터를 잘 예측한 곡선을 가진 모델

   가장 뛰어난 예측 성능

3. MSE 값이182815432

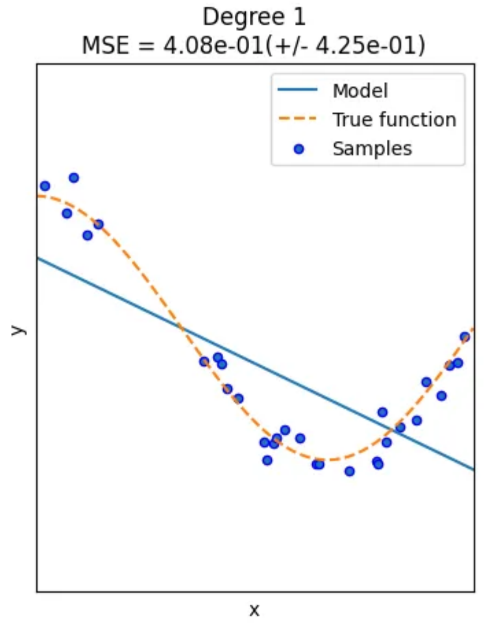
   데이터 세트의 변동 잡음값까지 지나치게 반영한 결과, 학습 데이터 세트만 정확히 예측하고 테스트 값의 실제 곡선과는 완전히 다른 형태의 예측 곡선이 만들어짐

   학습 데이터에 너무 충실하게 맞춘 과적합이 심한 모델

   💡 좋은 예측 모델

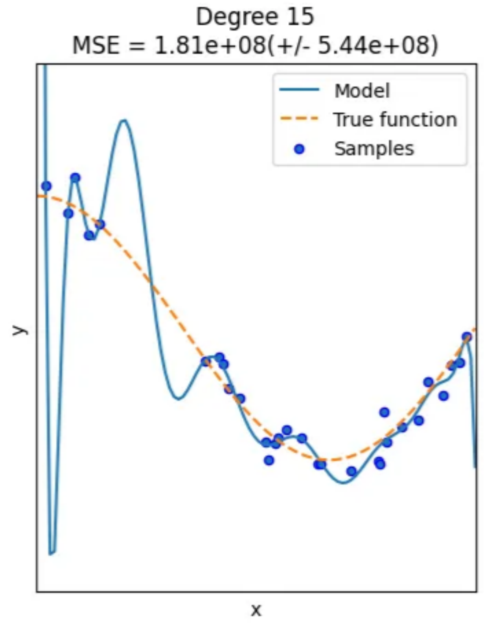
   : 학습 데이터의 패턴을 잘 반영하면서도 복잡하지 않은 균형 잡힌 모델

- 편향-분산 트레이드오프(Bias-Variance Trade off)



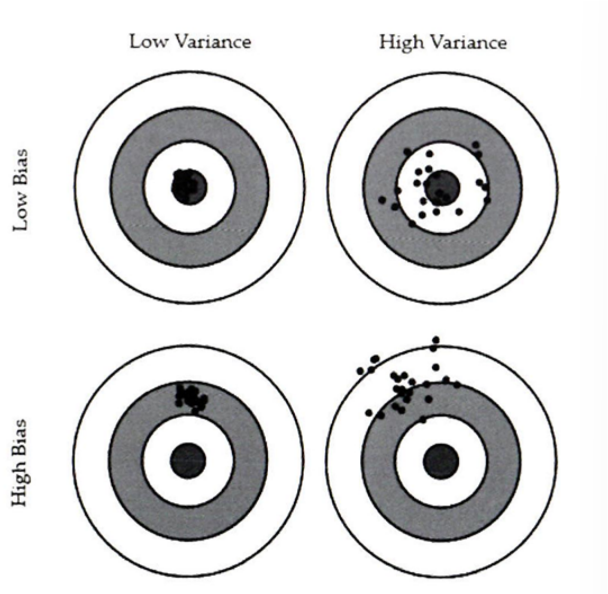
매우 단순화된 모델

→ 지나치게 한 방향성으로 치우침(고편향성)



매우 복잡한 모델

→ 지나치게 높은 변동성(고분산성)



1. 저편향/저분산

   : 예측 결과가 실제 결과에 매우 잘 근접

   : 예측 변동이 크지 않고 특정 부분에 집중

2. 저편향/고분산

   : 예측 결과가 실제 결과에 비교적 근접

   : 예측 결과가 실제 결과를 중심으로 꽤 넓은 부분에 분포

3. 고편향/저분산

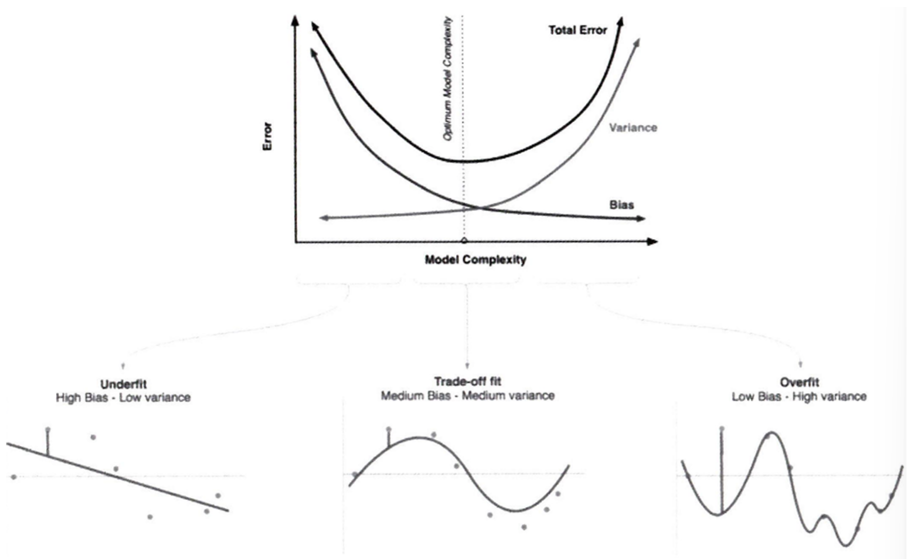
   : 정확한 결과에서 벗어남

   : 예측이 특정 부분에 집중됨

4. 고편향/고분산

   : 정확한 예측 결과를 벗어남

   : 넓은 부분에 분포



- 일반적으로 편향과 분산은 한 쪽이 높으면 한 쪽이 낮아짐

  → 과소 적합(편향 높음, 분산 낮음)

  → 과적합(편향 낮음, 분산 높음)

💡 편향 분산이 서로 트레이드오프를 이루면서 오류 Cost 값이 최대로 낮아지는 모델을 구축하는 것이 가장 효율적인 머신러닝 예측 모델을 만드는 방법

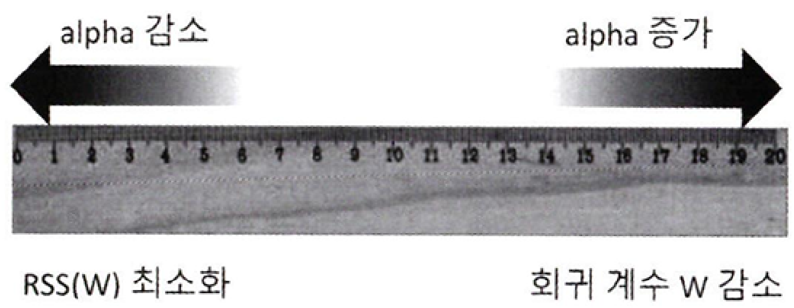
06. 규제 선형 모델- 릿지, 라쏘, 엘라스틱넷

- 규제 선형 모델의 개요

: 회귀 계수의 크기를 제어해 과적합을 개선하기 위해 비용 함수의 목표 변경

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

- alpha
  : 학습 데이터 적합 정도와 회귀 계수 값의 크기 제어를 수행하는 튜닝 파라미터
  - 역할
    : alpha 값을 크게 하면 비용 함수는 회귀 계수 W의 값을 작게 해 과적합 개선
    : alpha 값을 작게 하면 회귀 계수 W의 값이 커져도 어느 정도 상쇄가 가능하므로 학습 데이터 적합을 더 개선



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

💡 **규제(Regularization)**
: 비용 함수에 alpha값으로 패널티를 부여해 회귀 계수 값의 크기를 감소시켜 과적합을 개선하는 방식

- L1 방식
  → 라쏘(Lasso) 회귀
- L2 방식; W의 제곱에 대해 패널티 부여
  → 릿지(Ridge) 회귀

- 릿지 회귀

L2 규제는 회귀 계수의 크기를 감소시킴

Colab으로 예제 실행

- 라쏘 회귀

L1 규제는 불필요한 회귀 계수를 급격하게 감소시켜 0으로 만들고 제거
→ 적절한 피처만 회귀에 포함시킴(피처 선택의 특성)

Colab으로 예제 실행

- 엘라스틱넷 회귀

L2 규제와 L1 규제를 결합한 회귀

→

$$\text{목표는 } \mathrm{RSS}(W) + \mathrm{alpha2} * \|W\|_2^2 + \mathrm{alpha1} * \|W\|_1 \text{ 식을 최소화하는 W를 찾는 것}$$

→ 라쏘 회귀의 성향(중요 피처만을 셀렉션하고 다른 피처들은 회귀 계수를 0으로 만듦)으로 인해 alpha값에 따라 회귀 계수의 값이 급격히 변동하는 문제 완화

단점) 수행시간이 상대적으로 오래 걸림

Colab으로 예제 실행

- 선형 회귀 모델을 위한 데이터 변환

💡 선형 회귀 모델

: 일반적으로 타깃값 간에 선형의 관계가 있다고 가정 (선형 모델의 특징)

: 피처값과 타깃값의 분포가 정규분포인 형태를 선호

왜곡된 분포도로 인해 예측 성능에 부정적 영향

→ 선형 회귀 모델 적용 전 데이터에 대한 스케일링/정규화 작업 수행

- 사이킷런을 이용해 피처 데이터 세트에 적용하는 변환 작업

1. StandardScaler 클래스를 이용해 평균이 0, 분산이 1인 표준 정규 분포를 가진 dataset으로 변환하거나 MinMaxScaler 클래스를 이용해 최솟값이 0이고, 최대값이 1인 값으로 정규화를 수행.

   : 예측 성능 향상을 크게 기대하기 어려운 경우가 많음

2. 스테일링/정규화를 수행한 dataset에 다시 다항 특성을 적용하여 변환.

   : 1번 방법을 통해 예측 성능에 향상이 없을 경우

   : feature의 개수가 매우 많을 경우에는 다향 변환으로 생성되는 feature의 개수가 기하급수로 늘어나서 과적합의 이슈가 발생

3. 로그 변환(Log Transformation)

   : 원래 값에 log 함수를 적용하여 보다 정규 분포에 가까운 형태로 값이 분포하게 함.

   실제로 선형 회귀에서 다 로그 변환을 훨씬 많이 사용.

Colab으로 예제 실행

알 수 있는 결과: 선형 회귀를 적용하려는 데이터 세트에 데이터 값의 분포가 심하게 왜곡되어 있을 경우에 로그 변환을 적용하는 것이 좋은 결과를 불러옴

07. 로지스틱 회귀

💡 로지스틱 회귀

: (선형 회귀 계열) 선형 회귀 방식을 분류에 적용한 알고리즘→ 분류에 사용

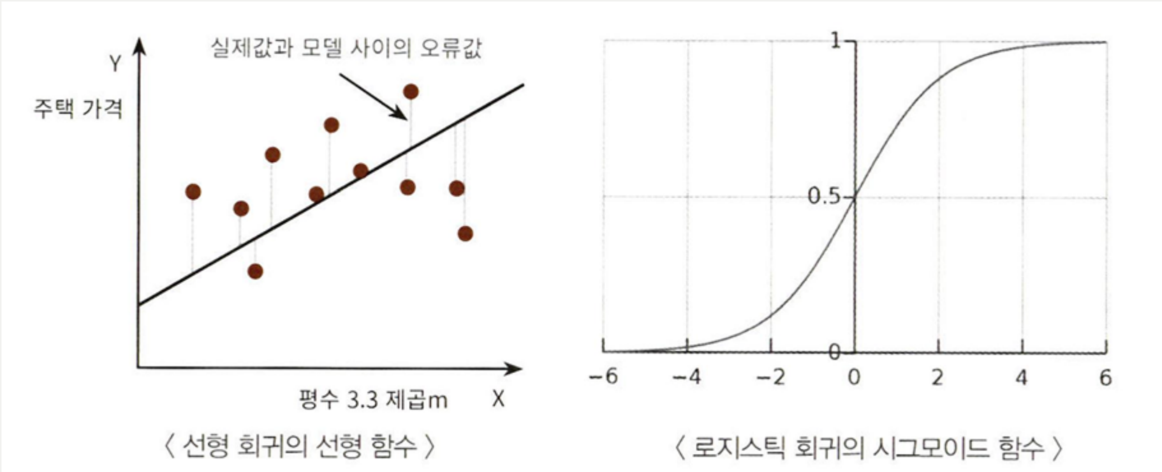
: 가볍고 빠르며 이진 분류 예측 성능이 뛰어남

→ 이진 분류의 기본 모델로 사용하는 경우가 많음

: 희소한 데이터 세트 분류에도 뛰어난 성능을 보임→ 텍스트 분류에서도 사용

선형 회귀와 다른 점

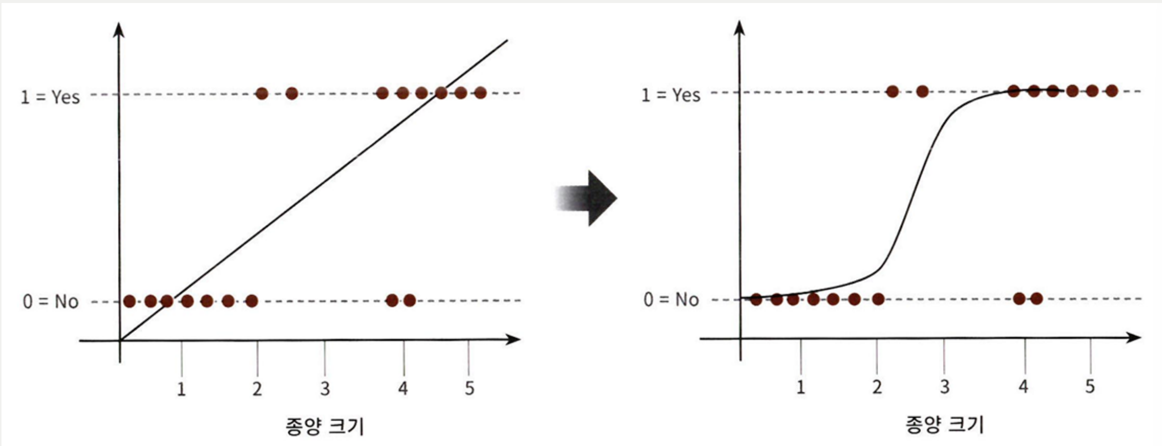
: 학습이 아니라 **시그모이드(Sigmoid)함수 최적선**을 찾고 이 시그모이드 함수의 반환 값을 확률로 간주해 확률에 따라 분류를 결정



종양 예시

: 선형 회귀를 적용하면 데이터가 모여 있는 곳으로 선형 회귀 선을 그릴 수 있지만 , 회귀 라인이 0과 1을 제대로 분류하지 못함

: 시그모이드 함수를 이용하면 좀 더 정확하게 0과 1에 대해 분류할 수 있음



Colab으로 예제 실행

08. 회귀 트리

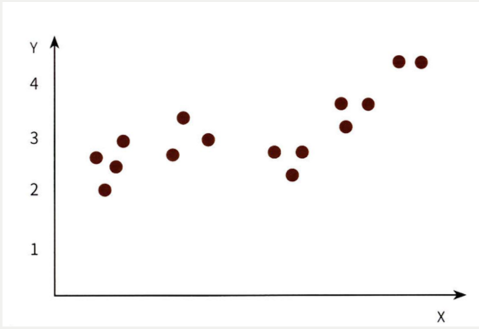
회귀 함수 기반이 아닌, 트리 기반의 회귀 방식

분류 트리와의 차이점

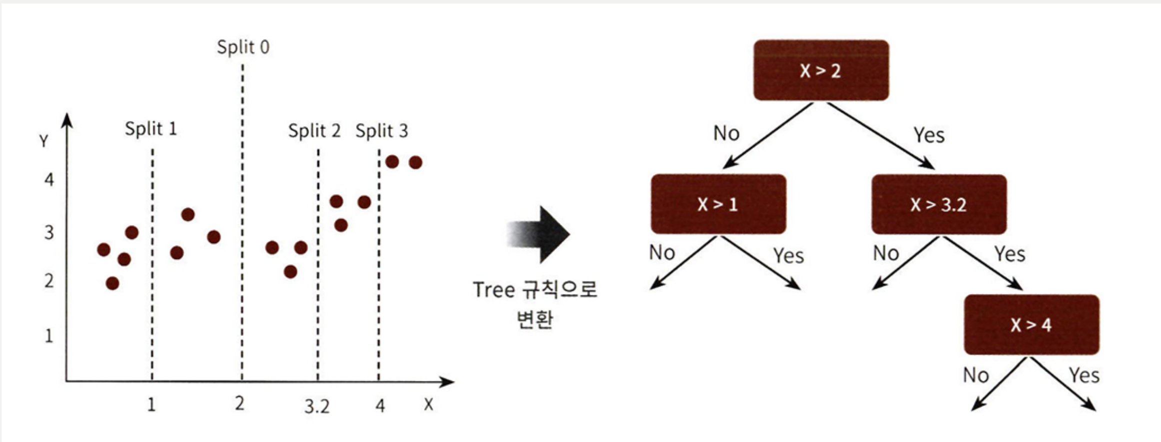
: 리프 노드에서 예측 결정 값을 만드는 과정

→ 회귀 트리는 리프 노드에 속한 데이터 값의 평균값을 구해 회귀 예측값을 계산

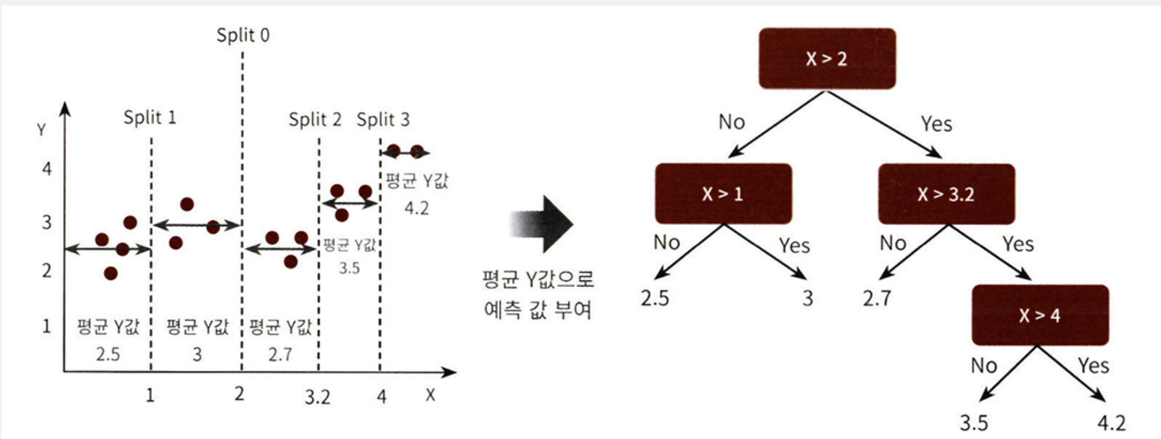
예시



피처가 단 하나인 X피처 데이터 세트와 결정값 Y



X 피처를 결정 트리 기반으로 분할(X값의 균일도를 반영한 지니 계수에 따라 분할)



트리 분할이 완료되면 리프 노드에 소속된 데이터 값의 평균값을 구해 최종적으로 리프 노드에 결정 값으로 할당

- 모든 트리 기반의 알고리즘은 분류 뿐 아니라 회귀도 가능

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

Colab으로 예제 실행