회귀, 분류 모델

ML Session 3차시

CONTENTS.

01. 회귀 모델

- 선형 회귀 분석
- 다항회귀
- 규제가 있는 선형 회귀
- 로지스틱 회귀

02. 분류 모델

- SVM
- KNN
- Decision Tree

회귀 모델

선형 회귀 분석

회귀 분석이란?

독립 변수 x에 대응하는 종속 변수 y와 가장 비슷한 값 $\hat{y}$ 을 출력하는 함수 f(x)를 찾는 과정을 말함

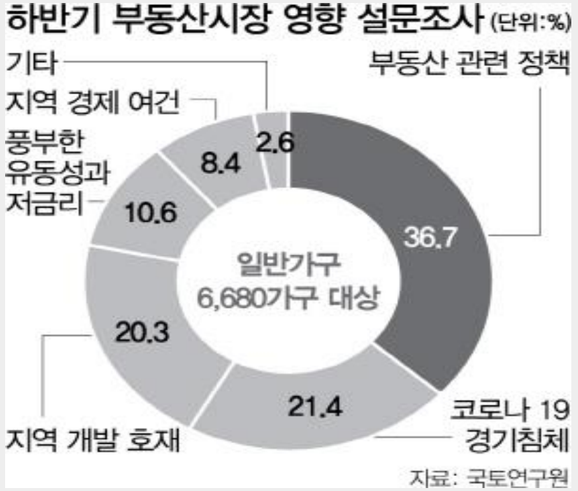
$$\hat{y} = f(x) \approx y$$

→ 목적: 독립 변수 x에 대응하는 종속 변수 y와 가장 비슷한 $\hat{y}$ 을 출력하는 가중치를 찾아야 함

선형 회귀 분석이란?

$$\hat{y} = \theta_0 + \theta_1 \cdot x_1 + \cdots + \theta_n \cdot x_n$$

- 예측하고자 하는 변수인 "집의 가격"을 타겟(target)이라고 하고, 나머지를 데이터의 특성(feature)이라고 할 때…
- f(x)가 선형 함수인 회귀 모형이면 선형 회귀 분석



선형 회귀 분석

X y

	roof	yard	bathroom	livingroom	room	price
0	1	481	5	1	10	55400
1	1	139	2	3	17	23400
2	1	576	5	1	11	65200
3	1	551	2	2	11	
4	0	40	2	2	18	

$$\hat{y} = \theta_0 + \theta_1 \cdot x_1 + \cdots + \theta_n \cdot x_n$$

Θ는 각 독립변수 X에 곱해지는 가중치로
학습을 통해 찾아야 하는 값

X에 어떤 Θ를 곱해야 y와 가장 비슷할까?

회귀 모델

선형 회귀 분석

X

y

	roof	yard	bathroom	livingroom	room	price
0	1	481	5	1	10	55400
1	1	139	2	3	17	23400
2	1	576	5	1	11	65200
3	1	551	2	2	11	
4	0	40	2	2	18	

$$\hat{y} = \theta_0 + \theta_1 \cdot x_1 + \cdots + \theta_n \cdot x_n$$

1. y가 있는 데이터로 y를 가장 잘 설명하는 ŷ 을 완성시켜 최적의 Θ를 찾는다.

2. 찾은 회귀식에 x 값만 있는 데이터를 적용시켜 y를 예측한다.

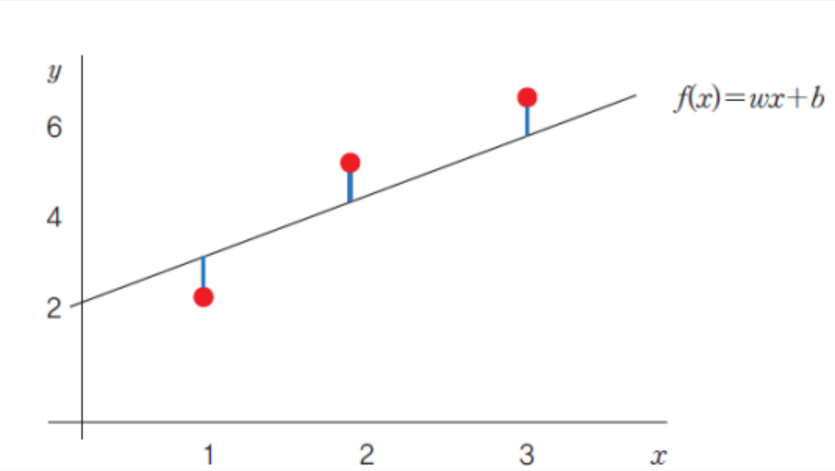
회귀 모델

선형 회귀 분석

Loss function

손실함수라고 하며, 실제값과 예측값의 차이를 특정 함수로

나타내 함수를 최소화 시키는 방향으로 모델의 학습이 진행됨



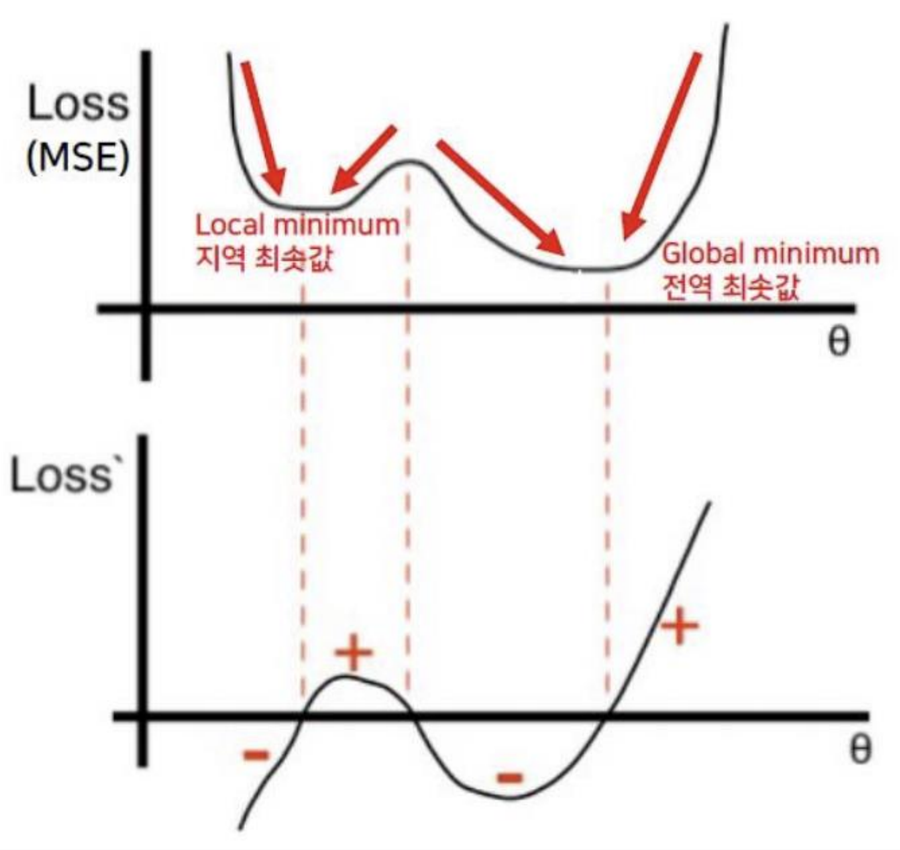
MSE

- 회귀 모델에서 주로 사용하는 Loss function
- 평균 제곱 오차
- 각 실제값과 예측값의 오차를 제곱한 값들을 평균한 값

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

선형 회귀 분석

경사하강법 (이론)



→ Loss의 미분값이 음수일 때 Θ가 커지게, 양수일 때는 Θ가 작아지게 학습!

기울기 반대 방향으로 η 배 만큼 이동

$$\theta_{i,j+1} = \theta_{i,j} - \eta \frac{\partial}{\partial \theta_{i,j}} J(\theta)$$

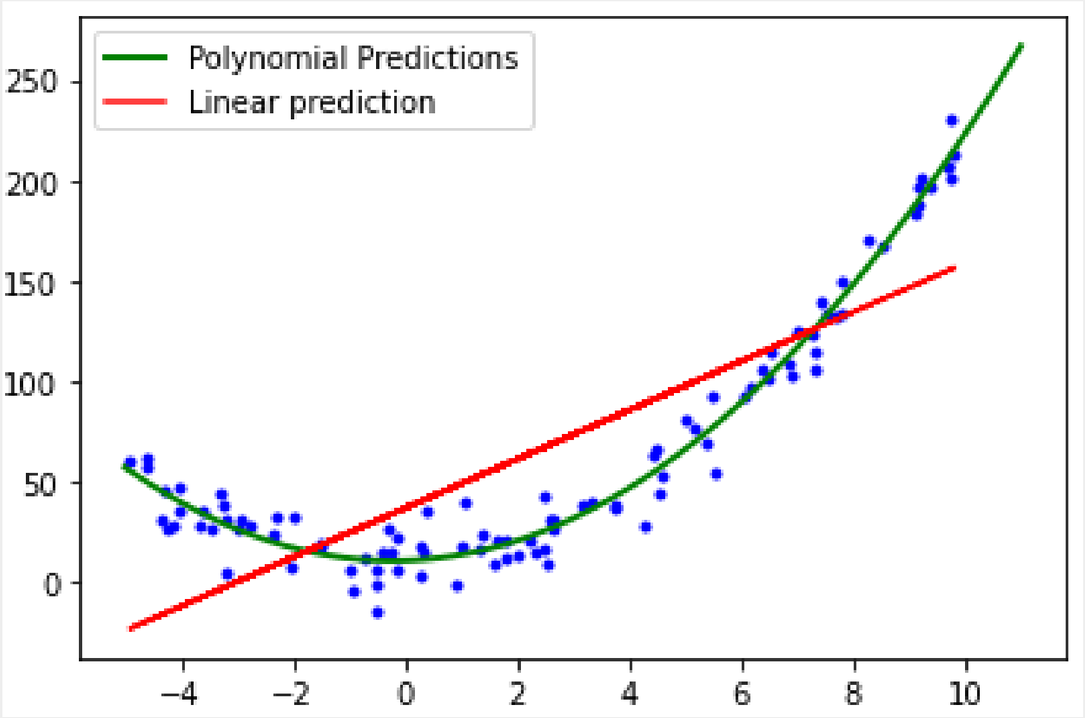
- $\theta_{i,j+1}$: 새롭게 갱신된 가중치
- $\theta_{i,j}$: 갱신 전 가중치
- η: 학습률
- $\frac{\partial}{\partial \theta_{i,j}} J(\theta)$: 해당 지점에서의 기울기

→ Θ를 업데이트할 때에는 Loss의 미분값을 활용

다항회귀

다항 회귀

독립 변수가 2개 이상인 다항식을 사용해 회귀 분석을 수행하는 것

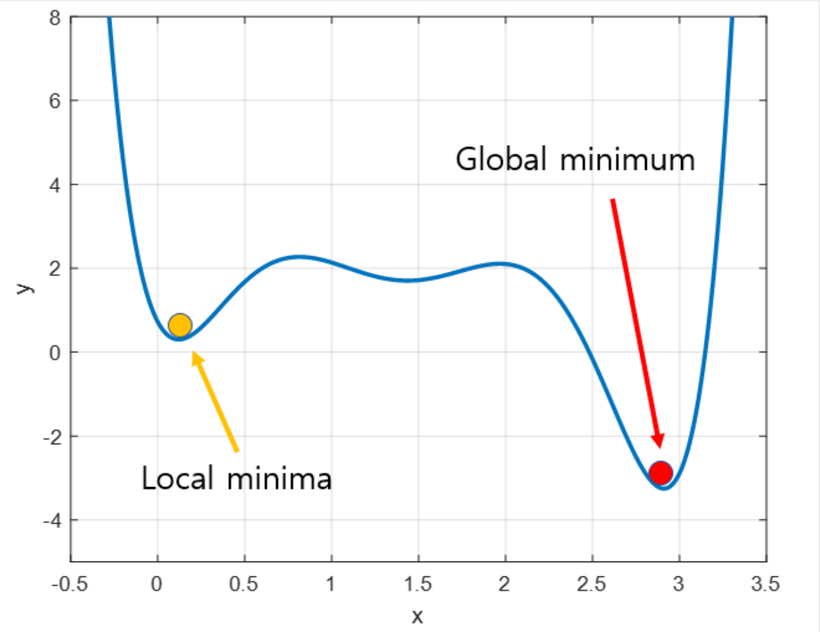


- 독립 변수의 일차항만을 가진 선형 함수는
  곡선 형태의 값들을 잘 표현할 수 없음
- 직선만으로 표현할 수 없는 복잡한 데이터에 대해서
  다항 회귀가 더 잘 적합하는 경우가 많음

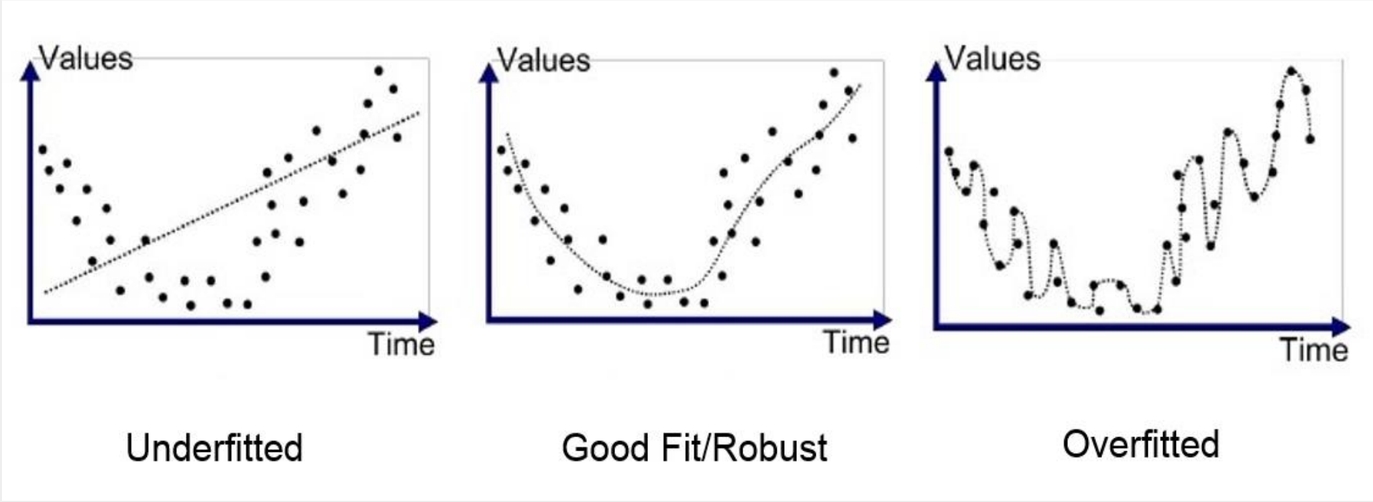
규제가 있는 선형 회귀

회귀 모델의 규제

학습 중에 전역 최솟값을 찾았다!
완전한 성공일까?



다 맞춰버리겠어!
저 실제 값을 다 이어버리는 모델을 만들면 쉽잖아?



이러한 문제들을 해결하기 위해 독립 변수의 영향력과 개수에 규제를 주어 Overfitting을 감소시킴!

회귀 모델

규제가 있는 선형 회귀

Ridge 회귀 모형

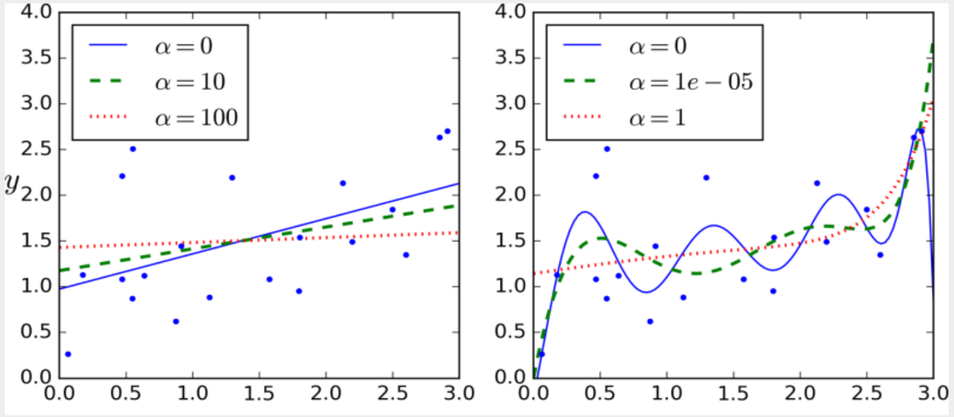
기존 손실 함수 + Θ에 대한 L2 규제 추가

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} \theta_i^2$$

α : 규제 정도를 조절하는 하이퍼 파라미터

Θ의 전체적인 크기에 제약을 줌

→ 기울기 값을 작게 해 급격한 변화를 줄임 (0이 되지 않음)



Lasso 회귀 모형

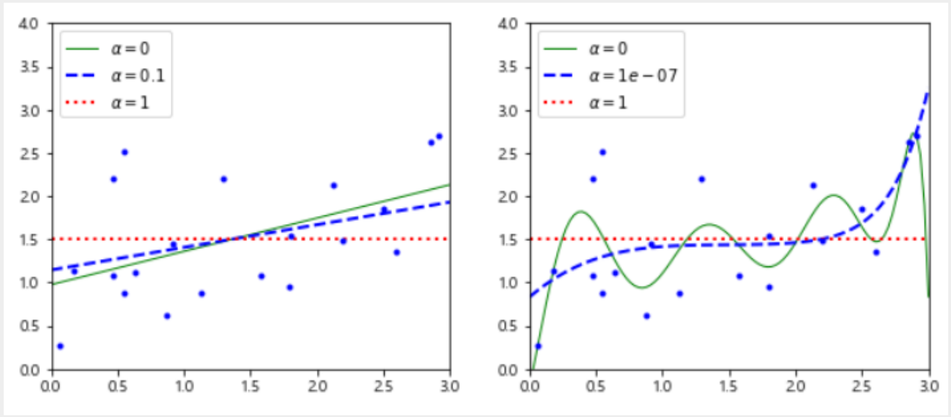
기존 손실 함수 + Θ에 대한 L1 규제 추가

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$$

α : 규제 정도를 조절하는 하이퍼 파라미터

Θ의 전체적인 크기에 제약을 줌

→ 기울기 값을 작게 해 급격한 변화를 줄임 (0도 가능!)

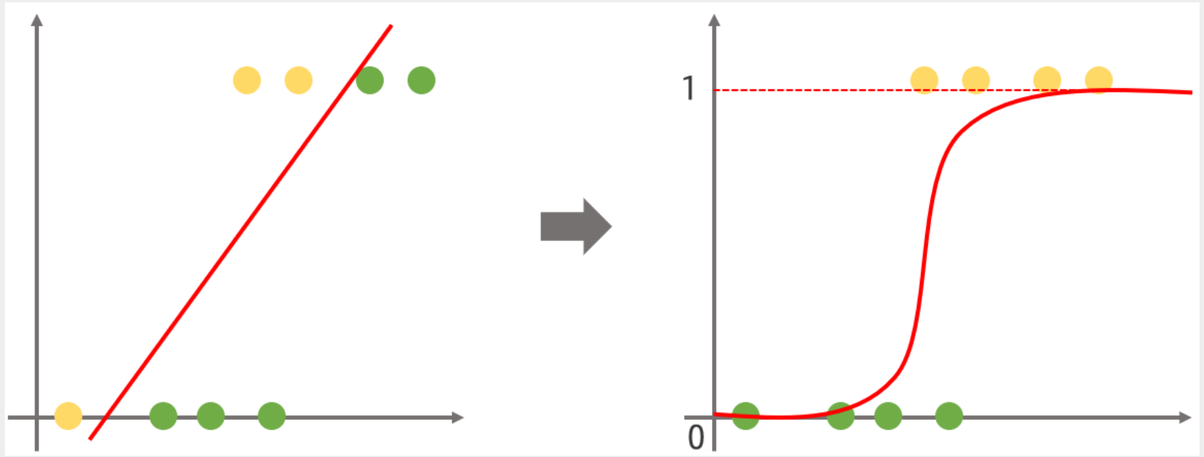


로지스틱 회귀

로지스틱 회귀 (Logistic Regression)

회귀를 사용해 데이터가 어떤 범주에 속할 확률을 0에서 1 사이의 값으로 예측하고 그 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해주는 지도 학습 알고리즘

Ex) 스팸 메일일 확률이 0.5 이상이면 스팸메일로, 그 미만이면 일반 메일로 분류

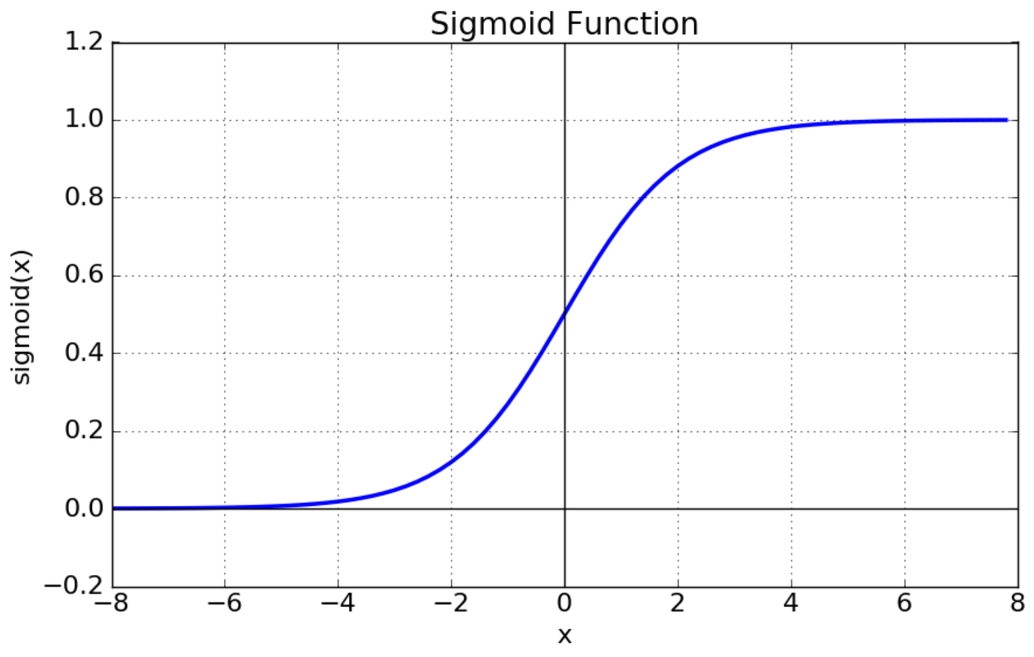


- 선형함수로 이를 표현하게 된다면 그림과 같이 확률이 1을 넘어가거나 확률이 음수로 표현되는 값이 존재
- 이를 방지하기 위해 로지스틱 회귀를 사용하고 이때 사용되는 함수는 시그모이드 함수

회귀 모델

로지스틱 회귀

시그모이드 함수 (Sigmoid function)



$$\sigma(t) = \frac{1}{1+\exp(-t)}$$

$$\hat{p} = h_\theta(x) = \sigma(\theta^T x) = \sigma(\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n)$$ 일 때…

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases}$$

→ 0.5를 기준으로
2개의 클래스로 분류

회귀 모델

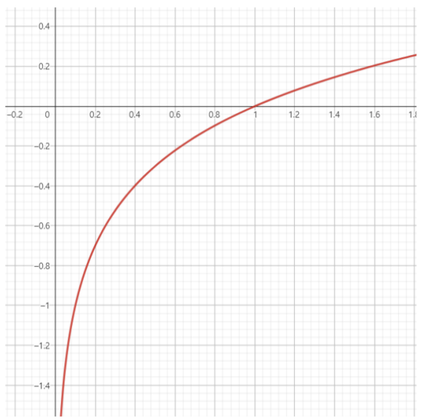
로지스틱 회귀

로지스틱 회귀의 손실 함수 (Log loss)

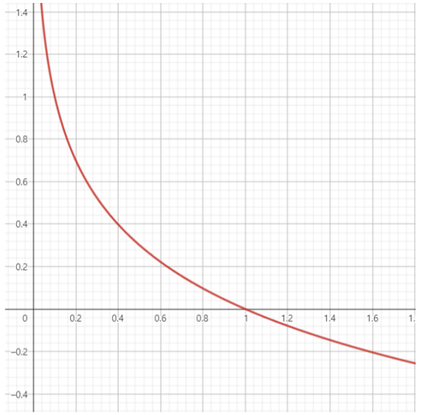
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{p}^{(i)}) + (1-y^{(i)})\log(1-\hat{p}^{(i)})\right]$$

→ y: 실제 y값 (0 또는 1)
→ p: 시그모이드 함수를 통해 계산된 확률

Log function 그래프



- Log function 그래프



If) y = 1 이면 첫 번째 항만 계산 → log 값이 1에 가까울수록 전체 loss가 감소하기 때문에 p가 커지도록 학습

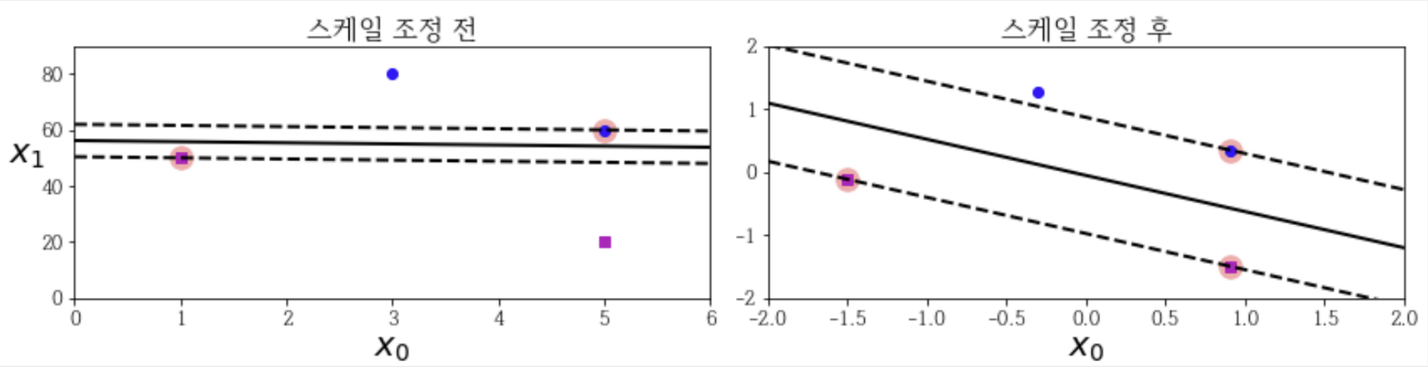
If) y = 0 이면 두 번째 항만 계산 → log 값이 1에 가까울수록 전체 loss가 감소하기 때문에 p가 작아지도록 학습

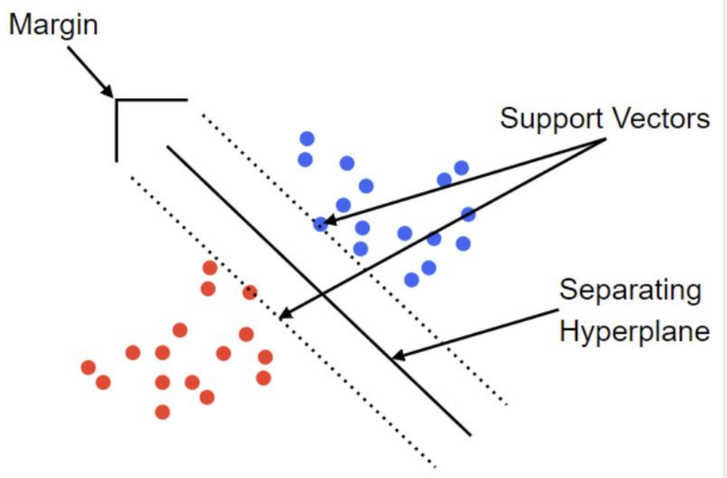
분류 모델

SVM (Support Vector Machine)

SVM이란?

- 선형 분류, 비선형 분류, 회귀, 이상치 탐색에도 사용할 수 있는 다목적 머신 러닝 모델
- 클래스가 다른 데이터들을 가장 큰 마진 (Margin) 으로 분리해내는 선 또는 면으로 찾아내는 것
- 기본 아이디어: 클래스 사이에 가장 폭 넓은 간격을 찾는 것
- 특히 복잡한 분류 문제에 잘 들어맞으며 작거나 중간 크기의 데이터셋에 적합
- 특성이 비슷하고 스케일이 비슷할 때 성능이 좋음



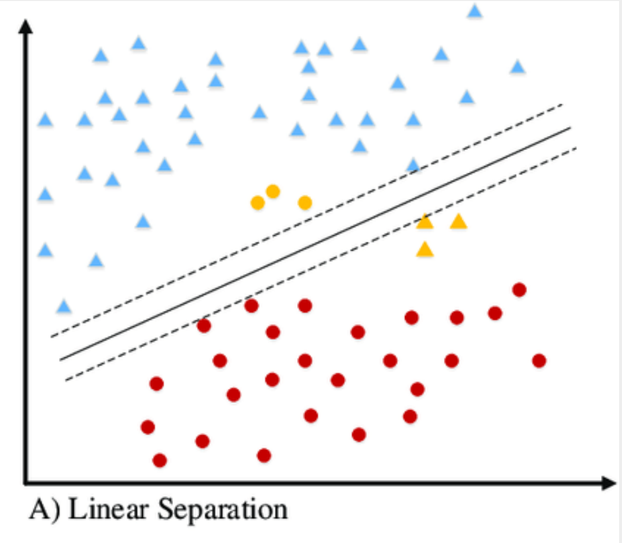


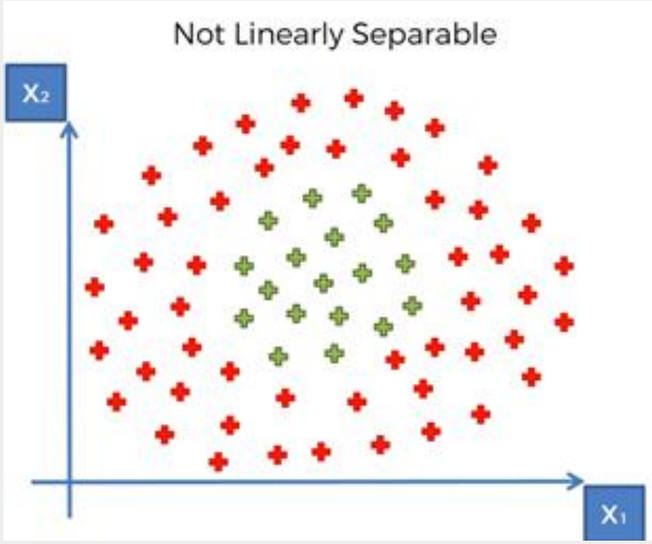
분류 모델

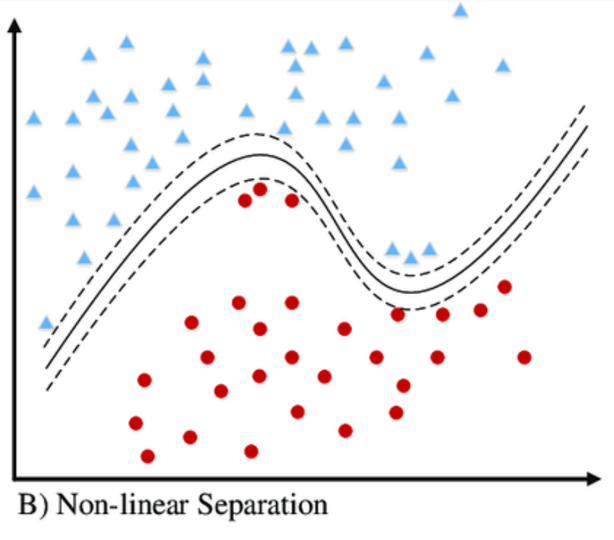
SVM (Support Vector Machine)

SVM 종류

- 선형 SVM
- 비선형 SVM → 다항 특성을 사용한 선형 SVM / Kernel - SVM





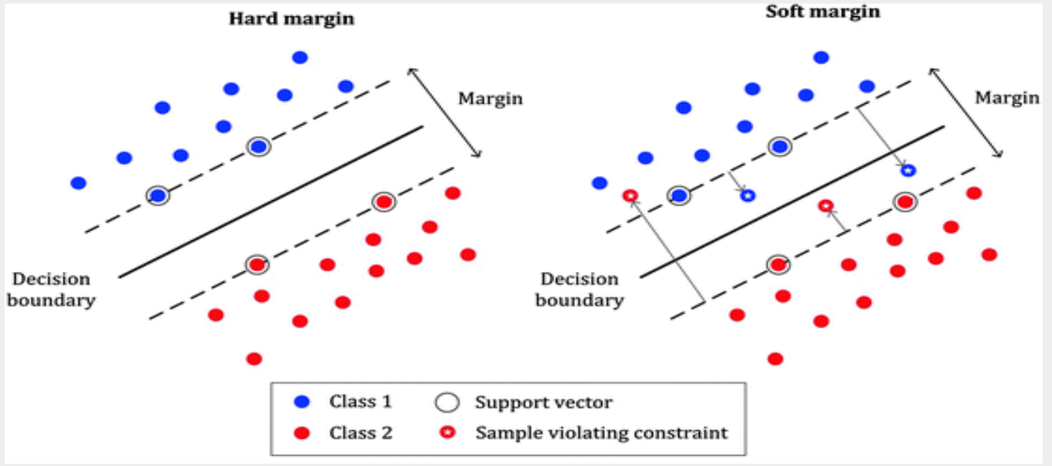


분류 모델

SVM (Support Vector Machine)

결정 경계 (Decision Boundary)

분류를 위한 기준선



Hard Margin

모든 샘플이 올바르게 분류된 경우

문제점

- 데이터가 선형적으로 구분될 수 있어야 제대로 작동
- 이상치에 민감

Soft Margin

좀 더 유연한 모델

도로의 폭을 가능한 넓게 유지하는 것과 마진 오류

사이에 적절한 균형을 잡는 것

하이퍼파라미터 C를 조절해 마진 오류를 조절할 수 있음

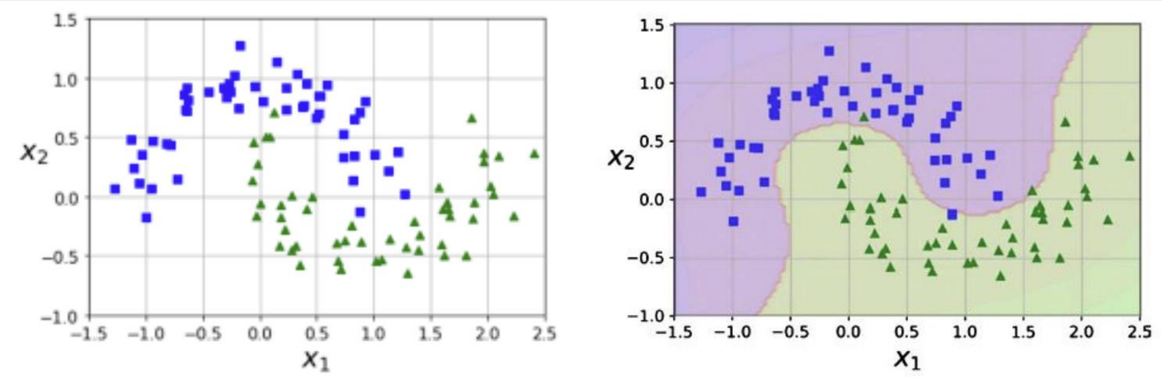
분류 모델

SVM (Support Vector Machine)

비선형 SVM

다항 특성을 활용한 선형 SVM

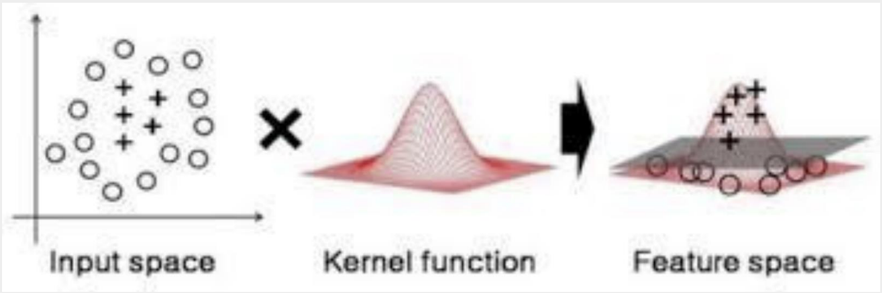
→ Polynomial Features를 이용



Kernal - SVM

Kernal 기법: 주어진 데이터를 고차원 특징 공간으로 사상해주는 것

→ 다항식 커널, 가우시안 RBF 커널 등이 존재



분류 모델

SVM (Support Vector Machine)

Hyperparameter

SVC: 분류 / SVR: 회귀

Parameter	Input 값	설명
C	float, default = 1.0	마진 오류를 얼마나 허용할 것인지 지정. 클수록 hard margin, 작을수록 soft margin. 반드시 양수.
kernel	‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’	알고리즘에 사용할 커널 유형을 지정.
degree	int, default = 3	다항식 커널 함수(‘poly’)의 차수. 다른 커널에서는 무시됨.
gamma	‘scale’, ‘auto’ (or float)	‘rbf’, ‘poly’ 및 ‘sigmoid’에 대한 커널 계수. 결정 경계를 얼마나 유연하게 그릴지 결정. 클수록 overfitting될 가능성이 높아짐.
coef0	float, default = 0.0	커널 함수의 독립 항. ‘poly’ 와 ‘sigmoid’에서만 적용됨.

분류 모델

SVM (Support Vector Machine)

장점

- 범주나 수치 예측 문제에 사용 가능
- 오류 데이터에 대한 영향이 적음
- 과적합되는 경우가 적음
- 신경망보다 사용하기 쉬움

단점

- 최적의 모델을 찾기 위해 커널과 모델에서 다양한 테스트 필요
- 여러 연산이 필요하고 입력 데이터셋이 많을 경우 학습 속도가 느림
- 해석이 어렵고 복잡한 블랙박스 형태로 되어있음

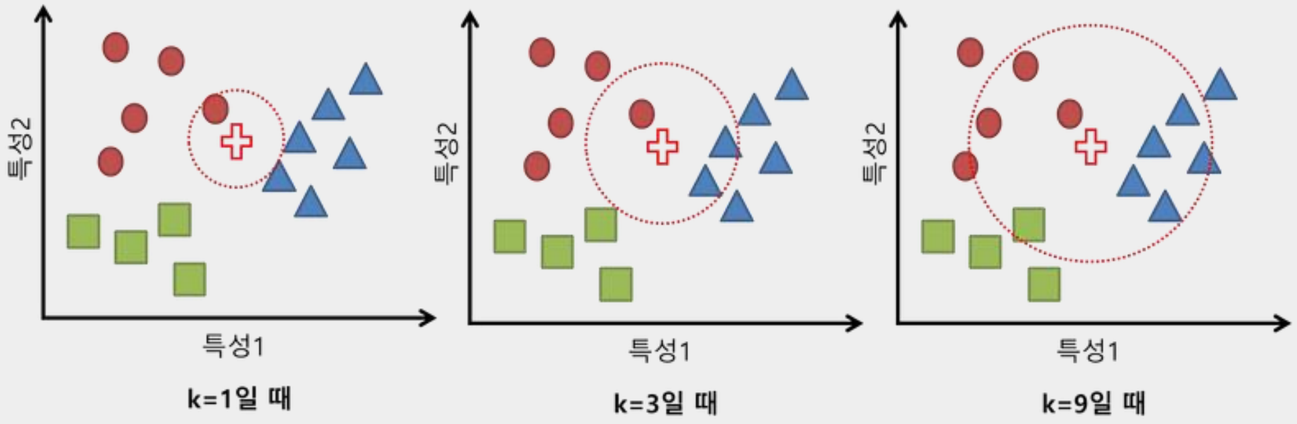
분류 모델

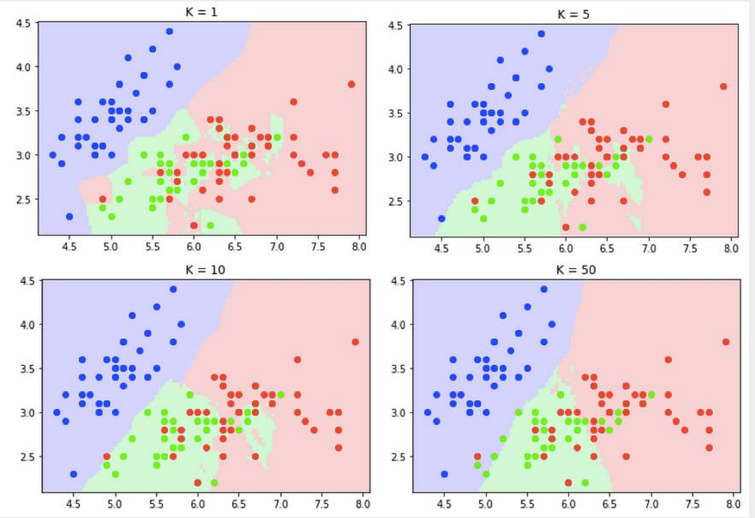
KNN

KNN이란?

K-Nearest Neighbor, K-최근접 이웃

- 굉장히 직관적이고 간단함
- 새로운 데이터가 주어지면 가장 가까운 K개 이웃의 데이터를 살펴본 뒤 더 많은 데이터가 포함되어 있는 범주로 데이터를 예측하는 방식





K 값이 커질수록

Overfitting을 방지하는 형태를 띈다.

분류 모델

KNN

Hyperparameter

Parameter	Input 값	설명
n_neighbors	int, default = 5	이웃 수, k를 의미.
weights	callable, ‘uniform’, ‘distance’	예측에 사용되는 가중치 함수. uniform : 가중치가 모두 균일 / distance : 거리에 따라 가중치 부여
algorithm	‘auto’, ‘ball_tree’, ‘kd_tree’, ‘brute’	가장 가까운 이웃을 계산하는 데 사용되는 알고리즘
P	int, default = 2	Minkowski 메트릭에 대한 검정력 매개변수. P = 1은 맨하튼 거리 / P = 2는 유클리드 거리
metric	str or callable, default = ‘minkowski’	거리 계산에 사용할 미터법. 디폴트 값에서 P = 2일 때 표준 유클리드 거리.

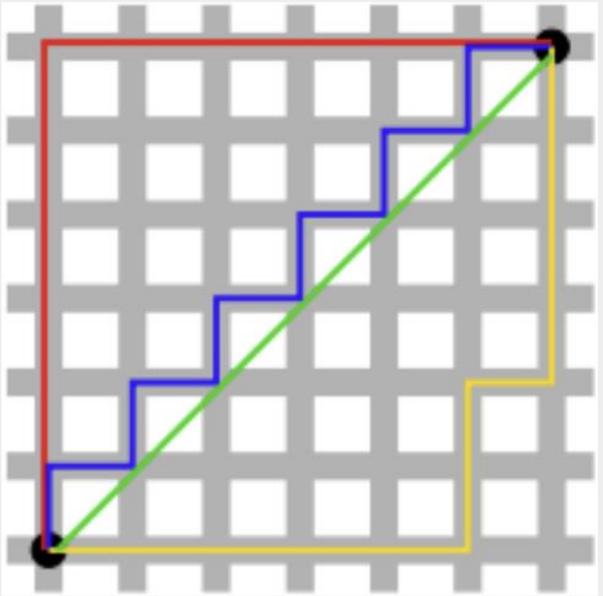
분류 모델

KNN

각종 거리 계산

Euclidean Distance (유클리드 거리)

- 두 관측치 사이의 직선 최단 거리를 의미
- 가장 흔히 사용되는 거리 척도
- 공식: $d(x,y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_p - y_p)^2}$



Manhattan Distance (맨해튼 거리)

- A에서 B로 이동할 때 각 좌표축 방향으로만 이동할 경우에 계산되는 거리
- 아래의 세 경로 (빨, 파, 노) 는 맨해튼 거리 기준으로는 같은 거리
- 공식: $d(x,y) = \sum_{i=1}^{p} |x_i - y_i|$

분류 모델

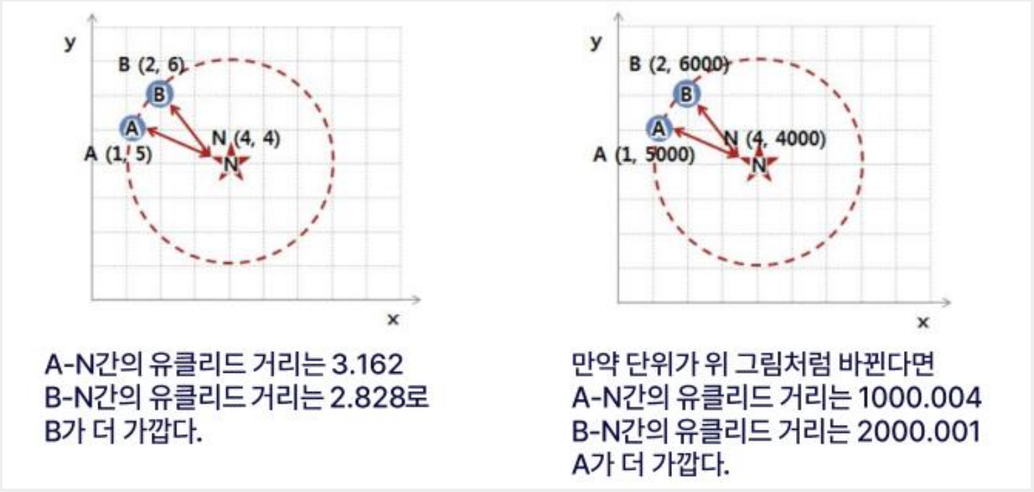
KNN

각종 거리 계산

Minkowski Distance (민코프스키 거리)

- M 차원 민코프스키 공간에서의 거리
- M = 1일 때 맨해튼 거리와 같고 m = 2일 때 유클리드 거리와 같음

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^m\right)^{\frac{1}{m}}$$



- 변수 별 단위가 무엇인지에 따라 가까운 순서가 달라지게 됨
- 가까움의 정도가 달라지면 KNN의 분류 결과도 달라지기 때문에 KNN을 사용할 때는 데이터에 표준화를 해야 함

분류 모델

KNN

장점

- 알고리즘과 원리가 간단해 구현하기 쉬움
- 알고리즘이 이해하기 쉬운 모델
- 수치 기반 데이터 분류 작업에서 성능이 좋음
- 모델 구축이 빠름

단점

- K와 어떤 거리 척도가 분석에 적합한지 불분명해 데이터 각각의 특성에 맞게 사용자가 임의로 선정해야 함
- Feature 의 수가 크면 계산량이 떨어지고 변수 간 거리가 멀어지기 때문에 정확도가 떨어짐
- 학습 데이터셋이 커질수록 느려짐
- Sparse dataset 과 같은 특정 데이터셋에서는 잘 작동하지 않음
- 범주형 Feature 는 차원 간의 거리를 찾기 어려워 잘 작동하지 않음

분류 모델

Decision Tree

Decision Tree (결정 트리) 란?

특정 규칙에 따라 레이블을 분류하는 모델

- 데이터의 어떤 기준을 바탕으로 규칙을 만드느냐가 성능을 결정하는 중요한 요소
- 분류와 회귀, 다중 출력이 가능한 머신러닝 알고리즘
- Random Forest 의 기본 구성 요소
- 지나치게 많은 규칙으로 분류할 경우, Overfitting 발생 가능성이 높음

가지치기 (Pruning)

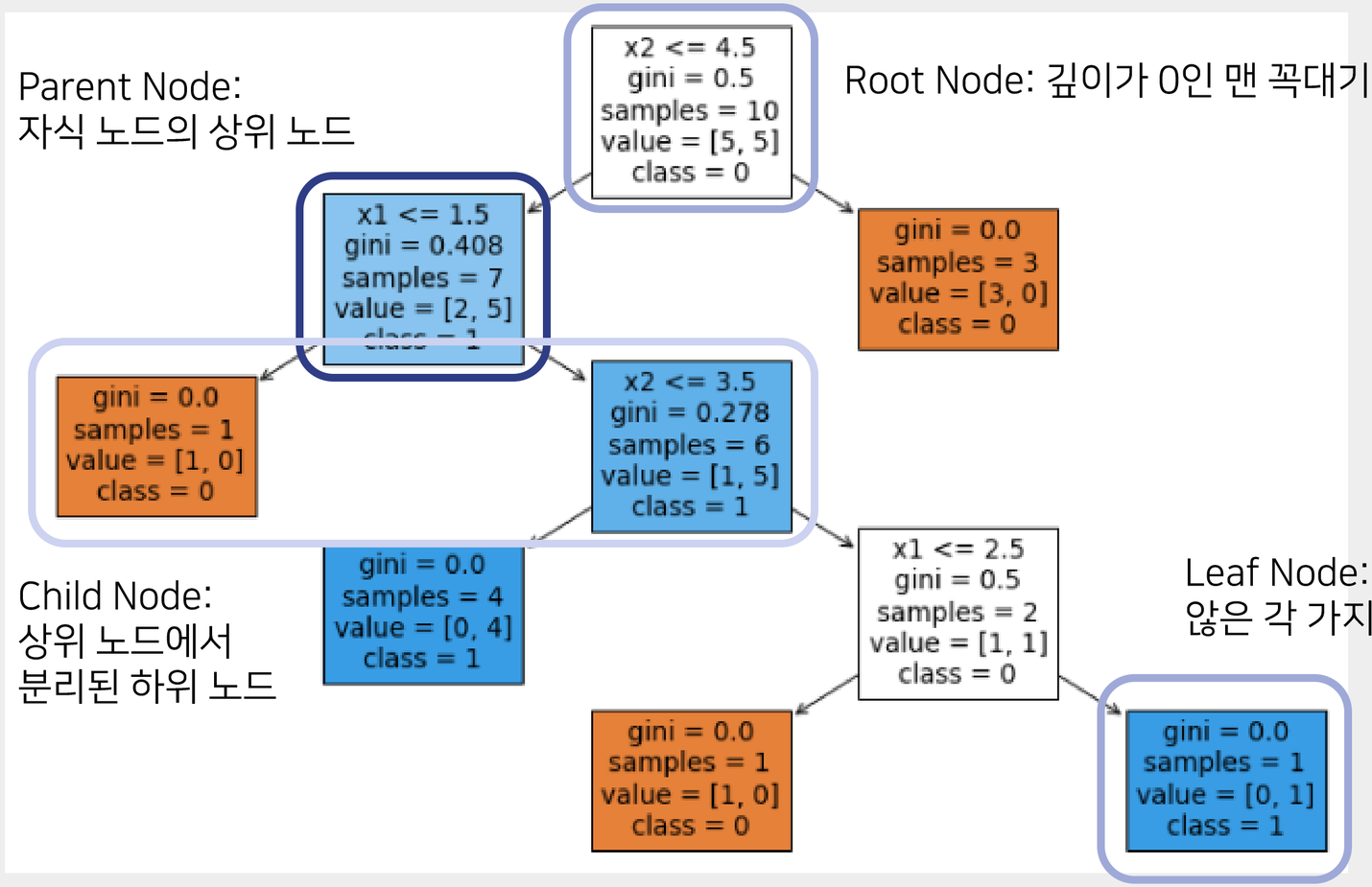
과적합을 막기 위한 전략

- 최대 깊이나 리프 노드의 최대 개수
- 한 노드가 분할하기 위한 최소 데이터 수를 제한하는 것

분류 모델

Decision Tree

깊이: 가지를 이루고 있는 노드의 분리 층수

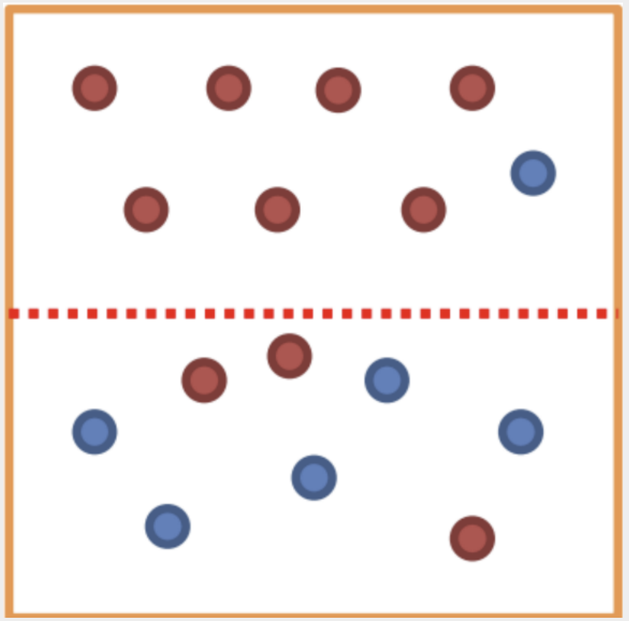


분류 모델

Decision Tree

불순도 (Impurity) 란?

- 해당 범주 안에 서로 다른 데이터가 얼마나 섞여있는지를 의미
- 위쪽 범주는 불순도가 낮고 (= 순도가 높고),
  아래쪽 범주는 불순도가 높음 (= 순도가 낮음)
- 이 불순도를 최소화하는 방향으로 학습을 진행



정보의 불순도 측정 방법

- 엔트로피 (Entropy) 를 이용한 정보 이득 (Information gain)
- 지니 계수 (Gini index)

분류 모델

Decision Tree

엔트로피 (Entropy)

- 불순도를 수치적으로 나타낸 척도
- 엔트로피가 높다 = 불순도가 높다

$$\text{Entropy} = -\sum_i (p_i \log_2(p_i))$$

지니 계수 (Gini index)

- 지니 계수가 낮을수록 데이터 균일도가 높다고 해석
- 0일 때 가장 균일, 1에 가까울수록 균일하지 않음
- 지니 계수가 낮은 속성을 기준으로 분할

$$G,\ I(A) = \sum_{i=1}^{d} \left(R_i \left(1 - \sum_{k=1}^{m} p_{ik}^2 \right) \right)$$

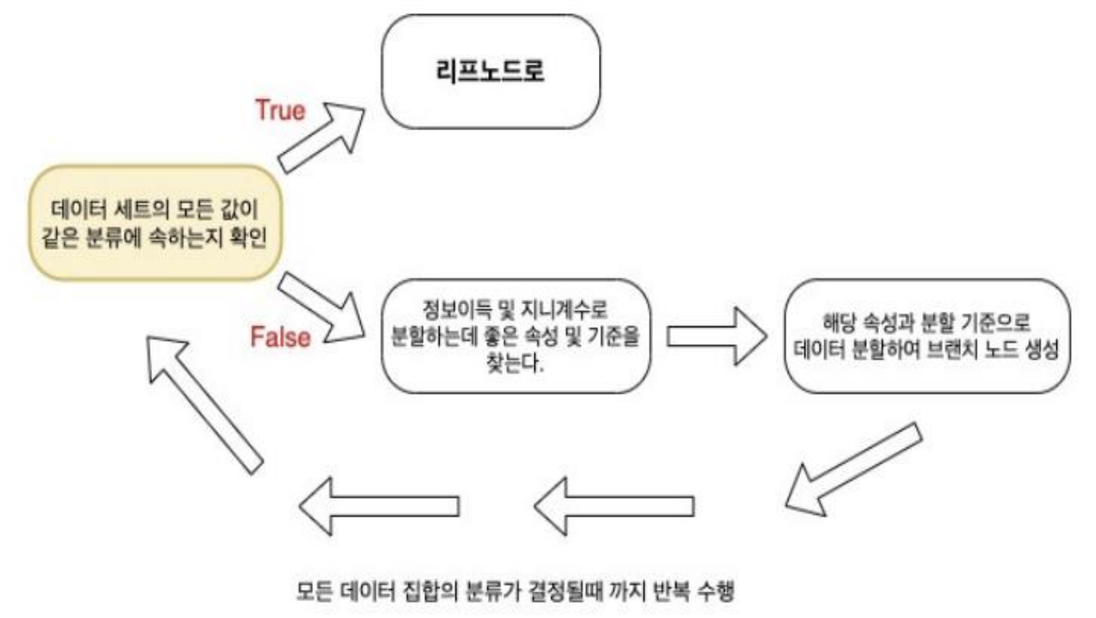
정보 이득 (Information gain)

- 데이터를 분리할 때 특정 노드 이전, 이후에 엔트로피 차이를 측정하는 척도
- 모든 Feature의 정보 이득 계산 후 정보 이득이 높은 Feature를 기준으로 분리

분류 모델

Decision Tree

결정 트리 과정



CART 훈련 알고리즘

Classification And Regression Tree

- 이진 트리만 만드는 알고리즘
- Scikit-learn 에서 Decision Tree를 훈련 시키기 위해 사용

CART 알고리즘이 Subset을 나누는 과정

- 정의된 최대 깊이가 되면 중지 OR 불순도를 줄이는
  분할을 찾을 수 없을 때 중지

분류 모델

Decision Tree

Hyperparameter

Parameter	Input 값	설명
max_depth	int, default = None	최대 깊이 지정 default 시, 노드의 데이터 수가 min_samples_split보다 작을 때까지 계속 분할
min_samples_split	int or float, default = 2	분할되기 위해 노드가 가져야 하는 최고 샘플 수 작게 설정할수록 분할되는 노드가 많아져 과적합 가능성 증가
min_samples_leaf	int or float, default = 1	리프 노드가 가지고 있어야 할 최소 샘플 수 if 비대칭적 데이터, 특정 클래스의 데이터가 극도로 작을 수 있어 이 경우 작게 설정
max_features	int, float or {'auto', 'sqrt', 'log2'}	각 노드에서 분할에 사용할 feature의 최대 수 int : feature 수, float : feature 비율 sqrt : $\sqrt{feature}$ 수, auto : sqrt와 동일, log2 : log2(feature 수)
max_leaf_nodes	int, default = None	리프 노드의 최대 수
min_weight_fraction_leaf	float, default=0.0	min_samples_leaf와 같지만 가중치가 부여된 전체 샘플 수에서의 비율

분류 모델

Decision Tree

장점

- 이해하고 해석하기 쉬우며 Tree를 시각화할 수 있음
- 다중 출력 문제를 처리할 수 있음
- 범주형, 연속형 수치 모두 예측 가능
- Feature 의 스케일링이나 정규화 등의 데이터 전처리가 필요하지 않음

단점

- 데이터 수가 적을 경우 불안정
- 복잡한 구조를 가지면 Overfitting 가능성이 높아짐

THANK YOU

ML Session 3차시

빅데이터 분석 학회 D&A