

데이터마이닝팀

4팀

김현우
김준서
서희나
김수빈
변석주



CONTENTS

1. 트리 기반 모델

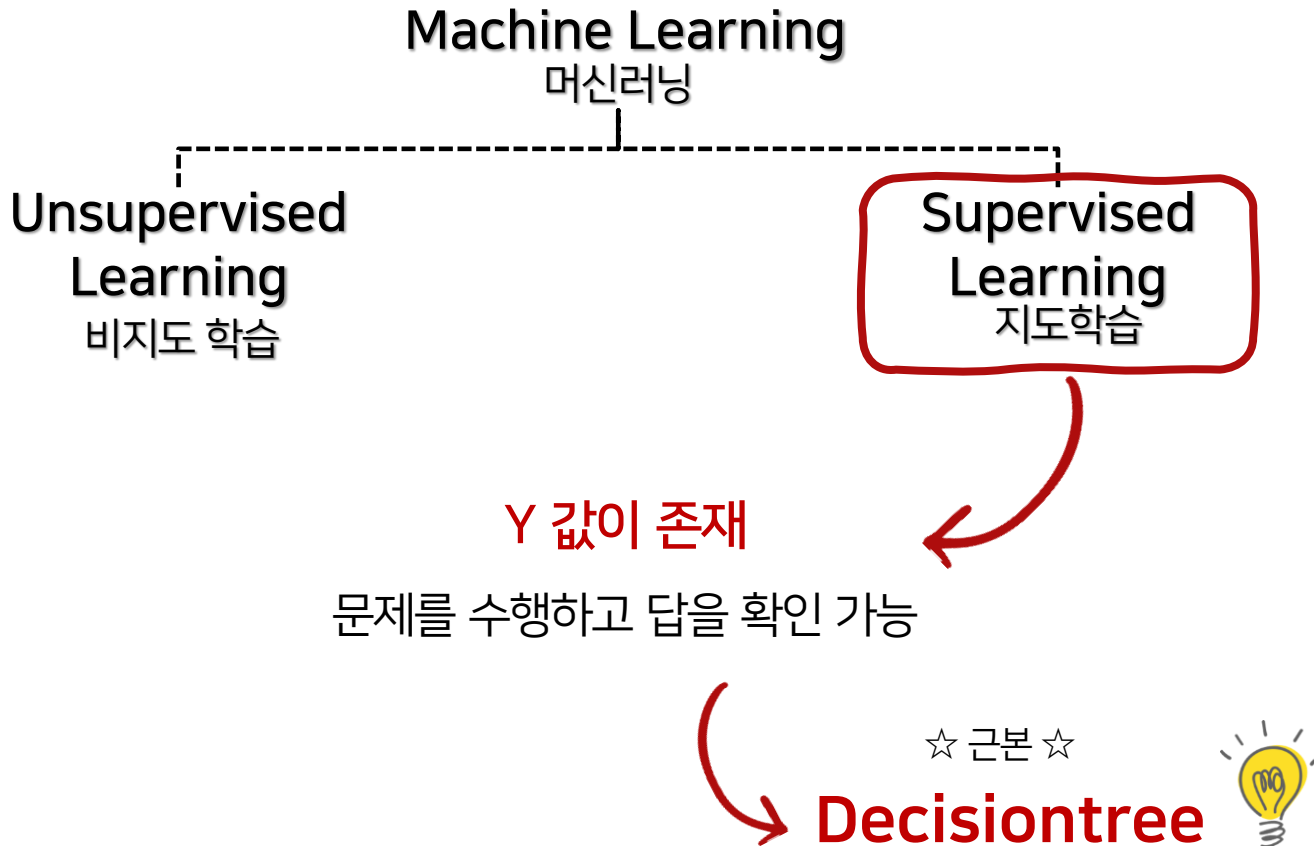
2. 생성 모델

1

트리 기반 모델

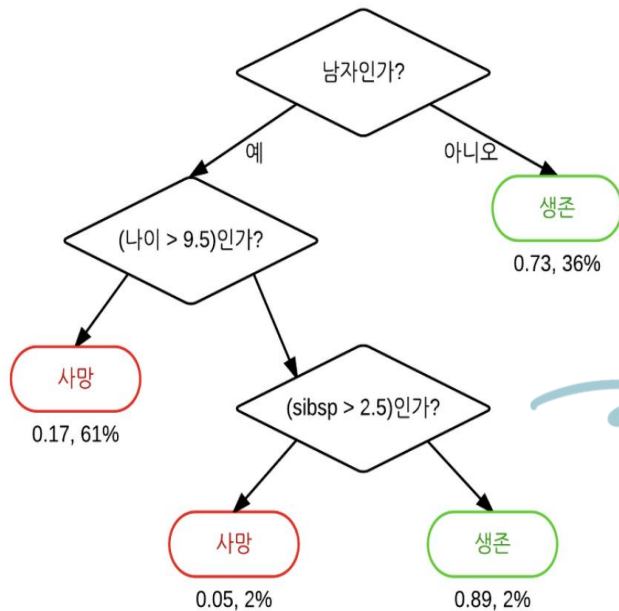
트리 기반 모델

지도 학습의 대표적 모델



트리 기반 모델

의사 결정 나무



질문에 대한 대답 → 영역 분할

이후 다른 변수를 이용해 질문

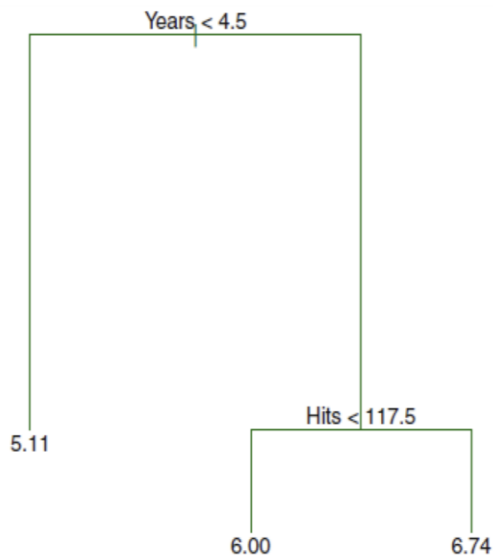
타이타닉 프로젝트의 Decision Tree 구현 예시

각 질문에 대한 대답으로 예측값이 결정되는 로직

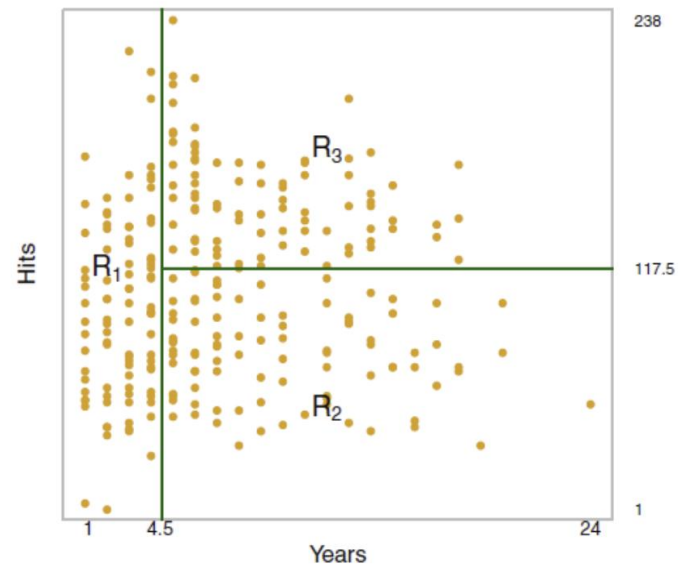
트리 기반 모델

트리 분기의 과정

“ 독립변수들이 차지하는 공간을 일정한 영역으로 분할 ”



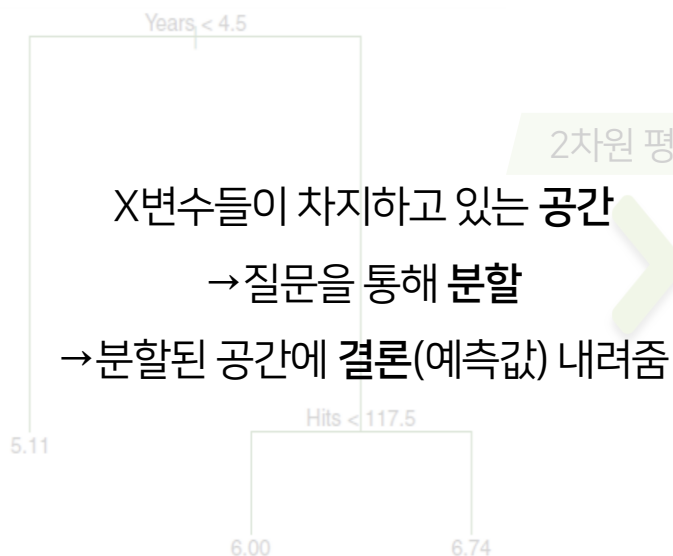
2차원 평면



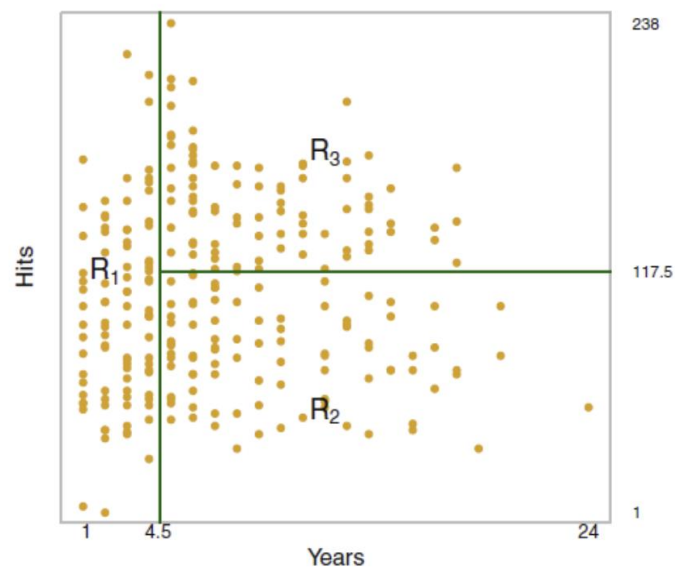
트리 기반 모델

트리 분기의 과정

“ 독립변수들이 차지하는 공간을 일정한 영역으로 분할 ”

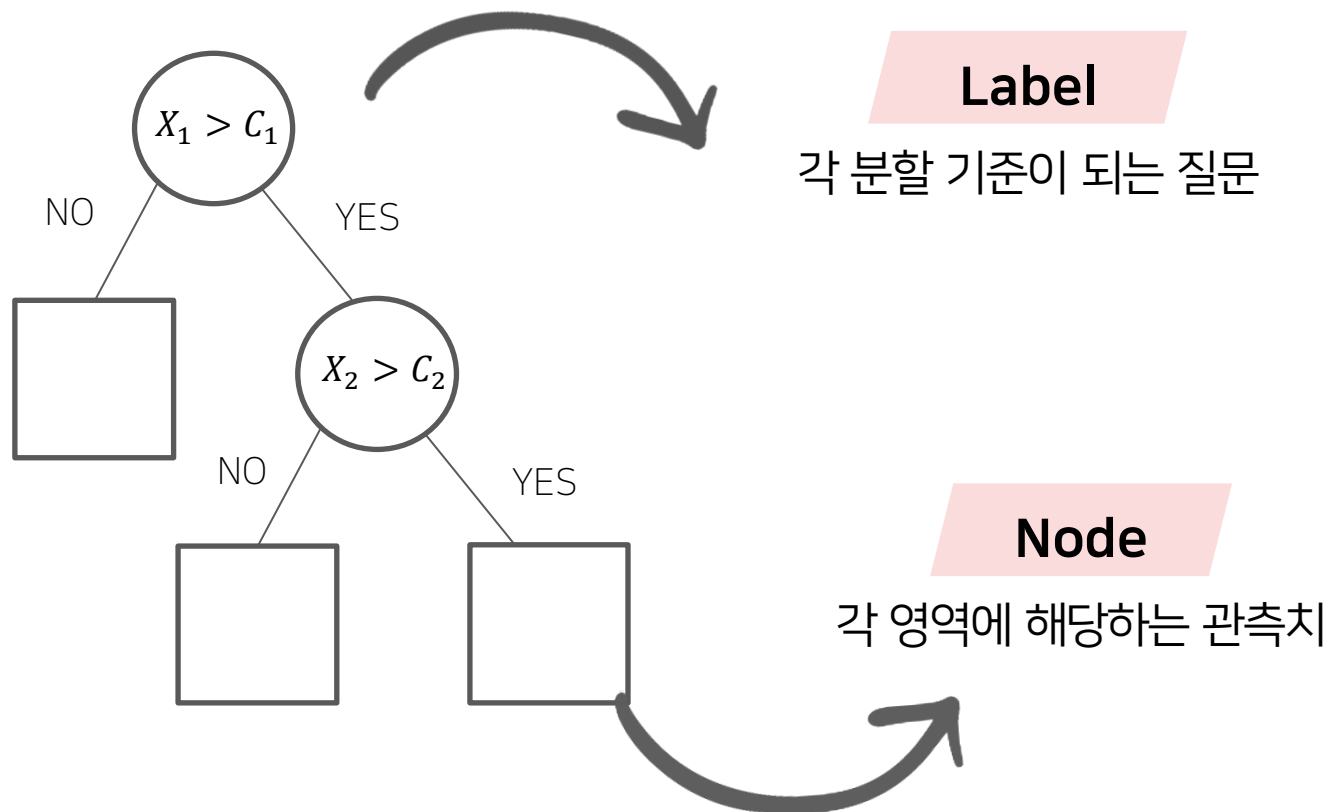


2차원 평면



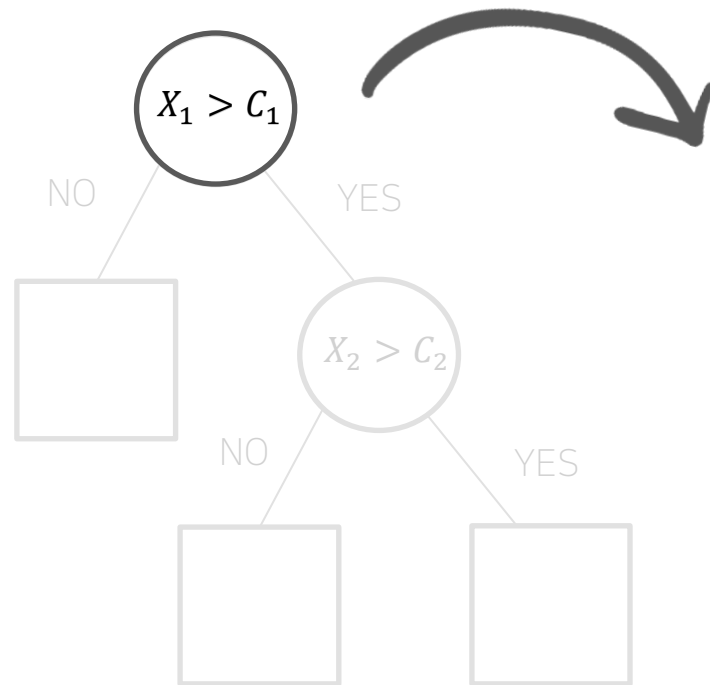
트리 기반 모델

용어 정리



트리 기반 모델

용어 정리

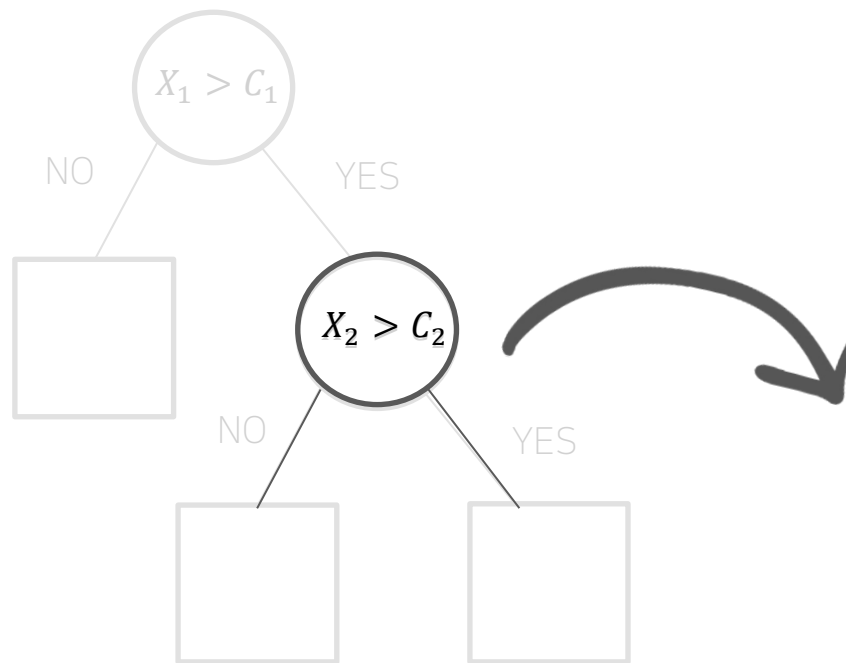


Root Node

영역이 나뉘지지 않았을 때 항목

트리 기반 모델

용어 정리



Intermediate Node

더 나뉘질 수 있는 항목

트리 기반 모델

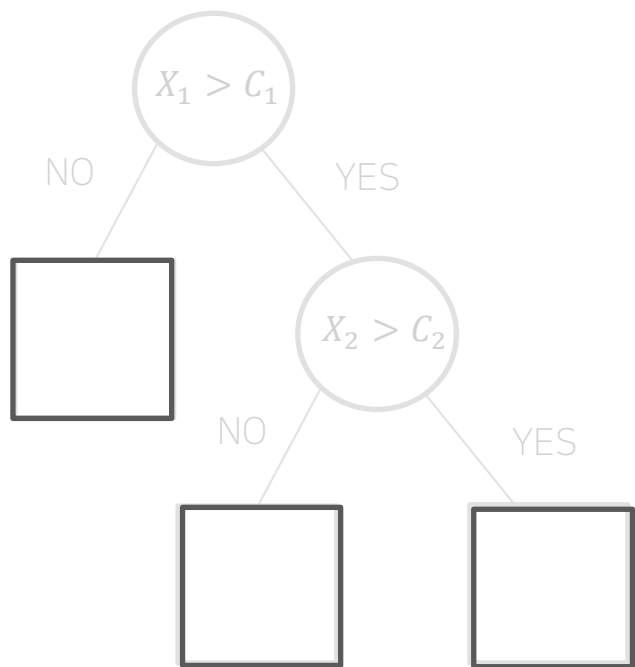
용어 정리

Terminal Node

더 이상 하위의 항목을 갖지 않는 노드

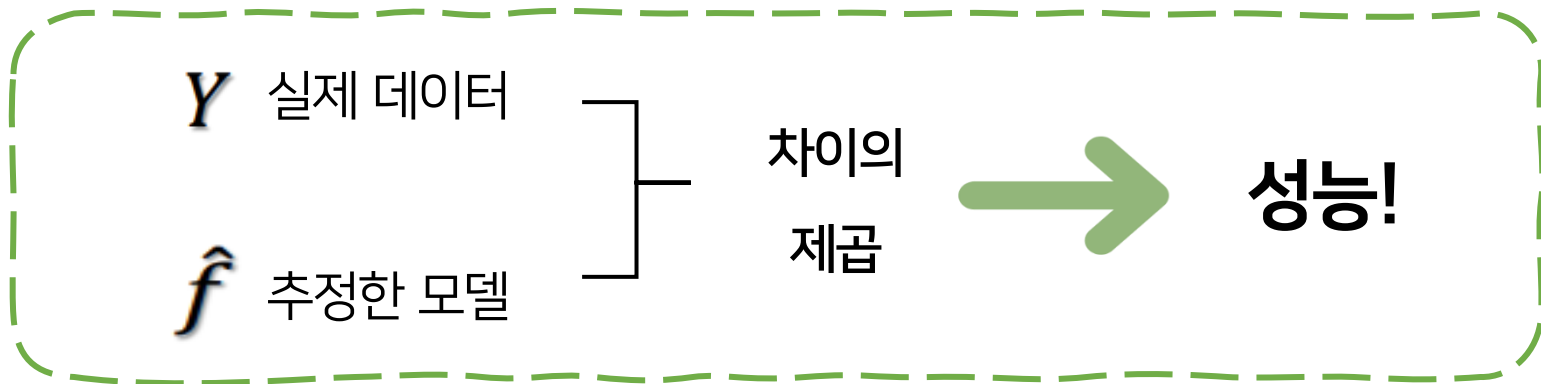
모델 적합 과정이 끝난 후

결과값이 들어가게 됨



회귀 모델

Decision Tree Regressor



$$\text{MSE} = \frac{\text{RSS}}{\text{자유도}}$$

(Mean Square Error)

(Residual Sum of Square)

Decision Tree의 모델학습은 RSS를 줄이는 방식으로 작동

회귀 모델

Decision Tree Regressor

Objective Function

$$\min c_m \sum_{i=1}^N \{y_i - f(x_i)\}^2 = \min c_m \sum_{i=1}^N \left[y_i - \sum_{m=1}^M c_m I(X \in R_m) \right]^2$$

c_m

해당 노드의 관측값들의 평균

N

전체 관측값의 개수

M

전체 node의 개수

회귀 모델

Decision Tree Regressor

Objective Function

$$\min c_m \sum_{i=1}^N \{y_i - f(x_i)\}^2 = \min c_m \sum_{i=1}^N \left[y_i - \sum_{m=1}^M c_m I(X \in R_m) \right]^2$$



i 번째 관측치의 실제값에 i 번째 관측치의
예측값을 뺀 오차의 제곱을 최소화

회귀 모델

Decision Tree Regressor

분기 기준

 $\text{Price} \leq 13$

Price	Review Rating
10.0	10
11.0	10
12.0	10
13.0	10
14.0	13
...	...
18.5	52
19.0	55
21.0	80
...	...
31.0	100

R1

R2

회귀 모델

Decision Tree Regressor

Price	Review Rating
10.0	10
11.0	10
12.0	10
13.0	10
14.0	13
...	...
18.5	52
19.0	55
21.0	80
...	...
31.0	100

R1

Review Rating의 평균

10**R2**

Review Rating의 평균

70.625

회귀 모델

Decision Tree Regressor

Price	Review Rating
10.0	10
11.0	10
12.0	10
13.0	10
14.0	13
...	...
18.5	52
19.0	55
21.0	80
...	...
31.0	100

R1

R1 영역의 RSS

$$(10 - 10)^2 + (10 - 10)^2 + \dots + (10 - 10)^2$$

→ 0

R2

R2 영역의 RSS

$$(10 - 70.625)^2 + (13 - 70.625)^2 + \dots + (100 - 70.625)^2$$

→ 13455.75

회귀 모델

Decision Tree Regressor

R1

R1 영역의 RSS

$$(10 - 10)^2 + (10 - 10)^2 + \dots + (10 - 10)^2$$

 $\rightarrow 0$ **R2**

R2 영역의 RSS

$$(10 - 70.625)^2 + (13 - 70.625)^2 + \dots \\ + (100 - 70.625)^2$$

 $\rightarrow 13455.75$ **전체 RSS** 13455.75

회귀 모델

Decision Tree Regressor



R1

R1 영역의 RSS

$$(10 - 10)^2 + (10 - 10)^2 + \dots + (10 - 10)^2$$

→ 0

R2

R2 영역의 RSS

$$(12 - 70.625)^2 + (13 - 70.625)^2 + \dots + (100 - 70.625)^2$$

→ 13455.75

분기 조건에 따라 달라지는 RSS 값



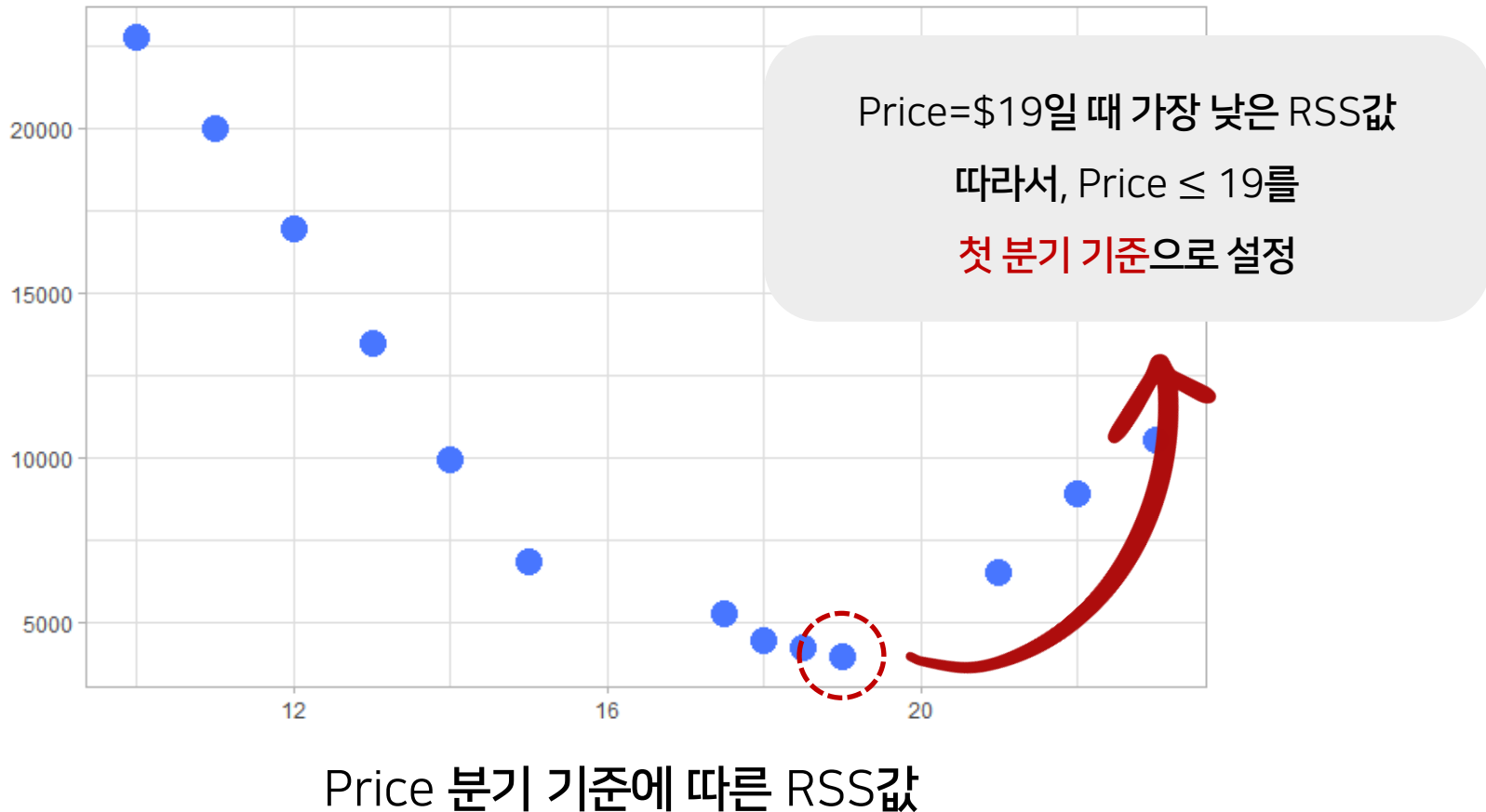
RSS 값이 가장 작은 조건을
의사결정나무의 분기 기준으로

채택!

전체 RSS 13455.75

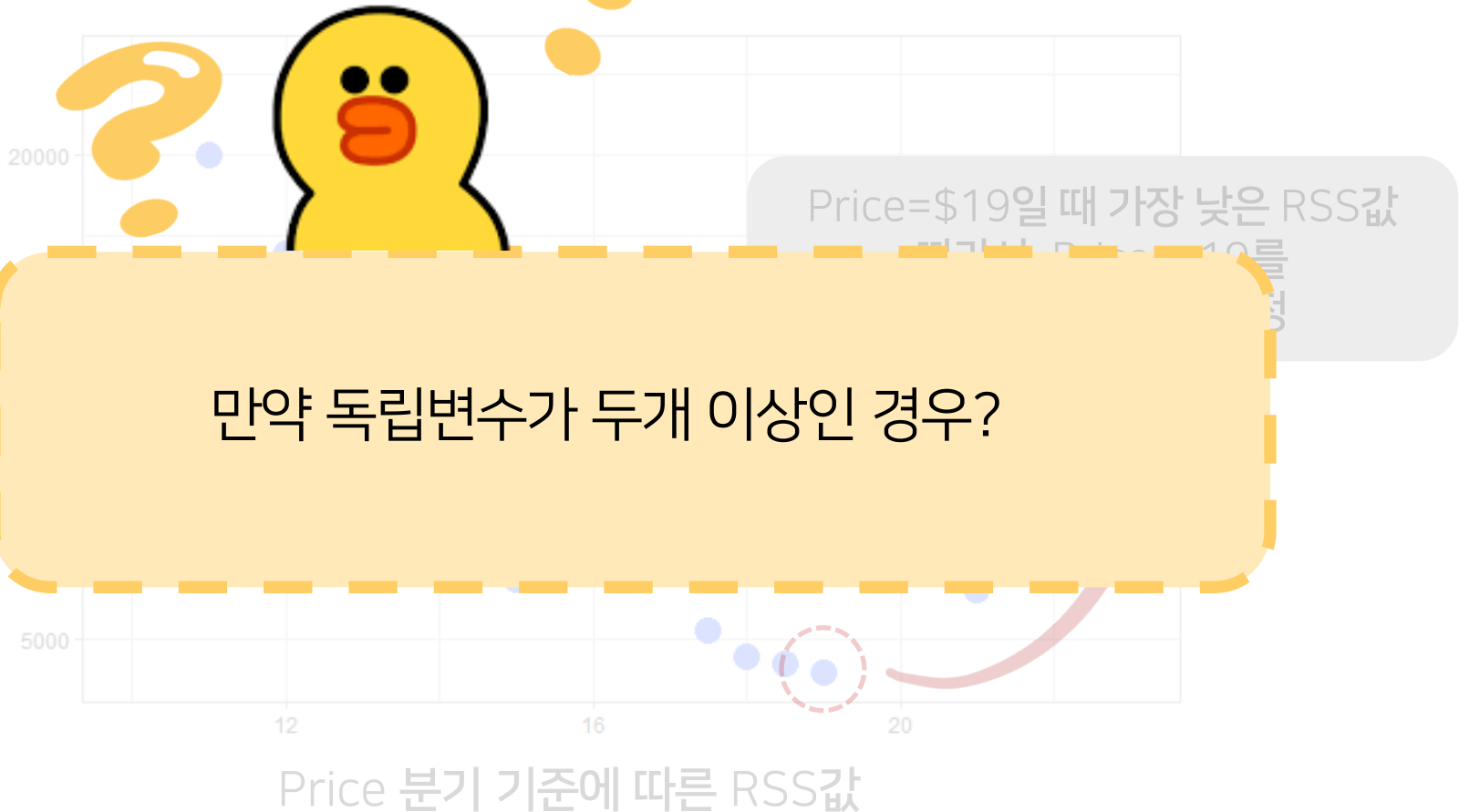
Decision Tree Regressor

탐욕적 알고리즘(Greedy Algorithm)



회귀 모델 | Decision Tree Regressor

탐욕적 알고리즘(Greedy Algorithm)




회귀 모델 | Decision Tree Regressor

Price	Parking fee	Foreign	Review Rating
10.0	0.0	0	10
11.0	1.0	0	10
...
17.5	7.5	1	35
18.0	8.0	1	44
18.5	8.5	1	52
19.0	9.0	0	55
21.0	11.0	1	80
22.0	12.0	0	83
...
24.0	14.0	1	83

회귀 모델 | Decision Tree Regressor

각 독립변수별로 **RSS값**이 **최소**로 만드는 기준 찾기

Price	Parking fee	Foreign	Review Rating
17.5	7.5	1	35
18.0	8.0	1	44
18.5	8.5	1	52
19.0	9.0	0	55
21.0	11.0	1	80
3961.8	64214.8	11685	



회귀 모델 | Decision Tree Regressor

각 기준에 따른 RSS값을 비교

Price	Parking fee	Foreign	Review Rating
17.5	7.5	1	35
18.0	8.0	1	44
18.5	8.5	1	52
19.0	9.0	0	55
21.0	11.0	1	80
19\$	7.5	1	최적기준
3961.8	64214.8	11685	RSS

회귀 모델 | Decision Tree Regressor

가장 작은 RSS값의 독립변수를 첫번째 분기기준으로

- ① 나뉜 R1, R2에서 다시 최적 기준 찾기
- ② RSS값이 더 이상 작아지지 않을 때까지 분할
- ③ 공간에 각 상황 별 예측값의 평균이 배치됨

회귀문제 ----> 오차의 제곱을 최소화 하는 것이 목표

R1	19.0	9.0	0	55
R2	21.0	11.0	1	80
	19\$	7.5	1	최적기준
	3961.8	64214.8	11685	RSS

회귀 모델 | Decision Tree Regressor

가장 작은 RSS값의 독립변수를 첫번째 분기기준으로

- ① 나뉜 R1, R2에서 다시 최적 기준 찾기
- ② RSS값이 더 이상 작아지지 않을 때까지 분할
- ③ 공간에 각 상황 별 예측값의 평균이 배치됨

회귀문제 ----> 오차의 제곱을 최소화 하는 것이 목표

분류문제 ----> 영역 내 output들의 class가
잘 분리되게 하는 것이 목표

3961.8

64214.8

11685

RSS

회귀 모델 | Decision Tree Regressor

가장 작은 RSS값의 독립변수를 첫번째 분기기준으로

- ① 나뉜 R1, R2에서 다시 최적 기준 찾기
- ② RSS값이 더 이상 작아지지 않을 때까지 분할
- ③ 공간에 각 상황 별 예측값의 평균이 배치됨

회귀문제 ----> 오차의 제곱을 최소화 하는 것이 목표

분류문제 ----> 영역 내 output들의 class가
잘 분리되게 하는 것이 목표

어떤 기준으로 분기할까?

3961.8

64214.8

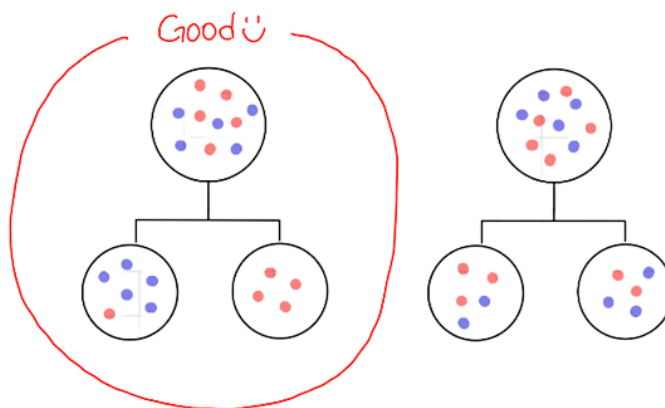
11685

RSS

분류 모델 | Decision Tree Classifier

불순도

DecisionTree Classifier의 경우 불순도를 기준으로 분기



불순도

분기 된 영역(ex. R1,R2)내에
여러가지의 클래스가 섞여 있는 정도

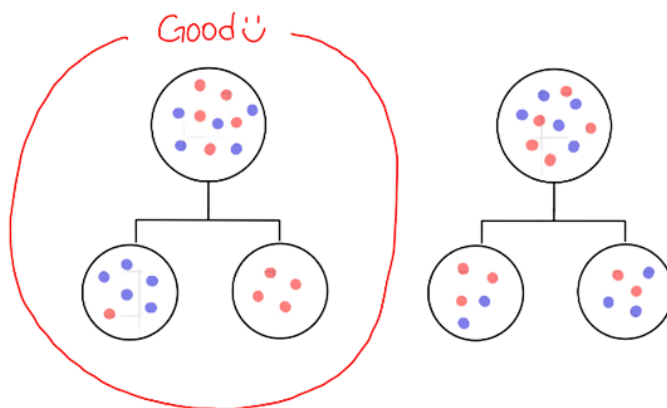


분류 모델 | Decision Tree Classifier

불순도



DecisionTree Classifier의 경우 **불순도**를 기준으로 분기



잘 분류된

잘 분류되지 않은

경우의 불순도(낮음)

경우의 불순도(높음)

불순도 자체가 **낮아지도록** 학습하는 것이 핵심!

분류 모델 | Decision Tree Classifier

불순도

불순도의 지표

Ex) Misclassification Rate, Gini Index, Entropy

CART -----> Gini index

ID3 -----> Entropy

분류 모델 | Decision Tree Classifier

불순도

불순도의 지표

Ex) Misclassification Rate, Gini Index, Entropy

CART -----> Gini index

ID3 -----> Entropy

분류 모델 | Decision Tree Classifier

Entropy

발생확률 \hat{p}_{mk} 를 한 영역 내의 특정 class의 비율이라고
생각하고 엔트로피 수식을 보자!

$$Entropy = - \sum_{k=1}^K \hat{p}_{mk} \log_2(\hat{p}_{mk})$$

\hat{p}_{mk} 분기된 영역 내의 k번째 class

k 영역 내의 class의 종류

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

분류 모델 | Decision Tree Classifier

Entropy



발생확률 \hat{p}_{mk} 를 한 영역 내의 특정 class의 비율이라고

생각하고 엔트로피 수식을 보자!

$$Entropy = - \sum_{k=1}^K \hat{p}_{mk} \log_2(\hat{p}_{mk})$$

영역 내의 \hat{p}_{mk} 들이 비슷하다면
해당 영역은 특정 class로 규정짓기 힘들다

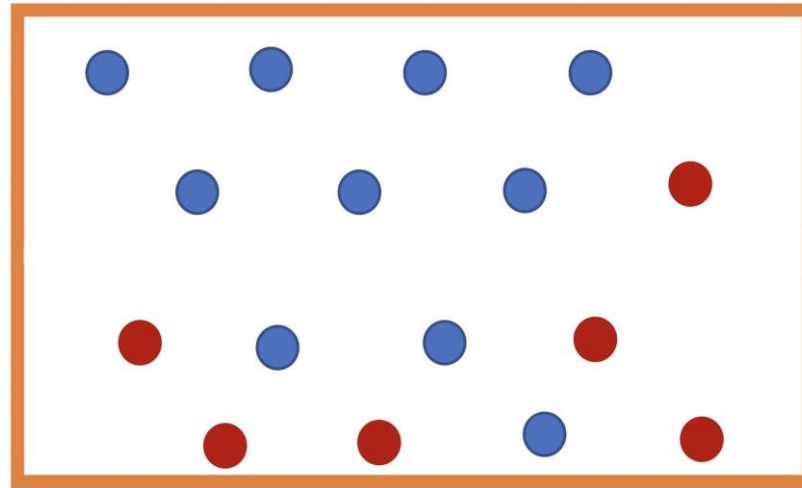
➔ 해당 영역의 불순도(=entropy)가 높다

k 영역 내의 class의 종류

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

분류 모델

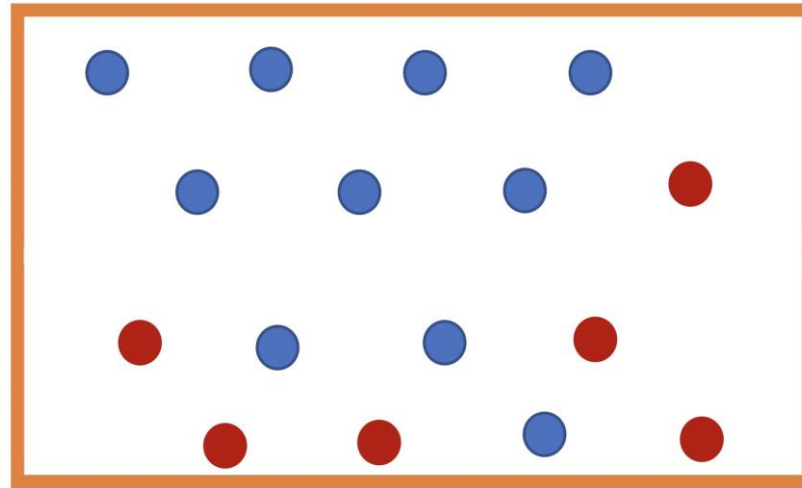
Decision Tree Classifier



다음과 같은 데이터가 주어졌다고 가정
트리 모델이 영역을 2개로 분할하려고 함

분류 모델

Decision Tree Classifier

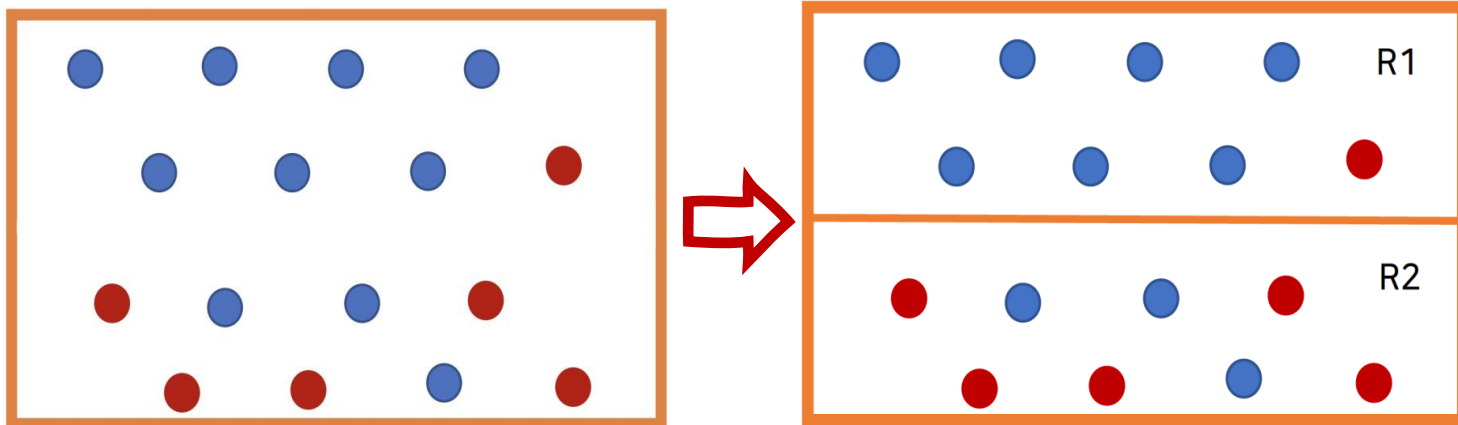


분기 전 Entropy

$$Entropy(X) = -\frac{10}{16} \log_2 \left(\frac{10}{16} \right) - \frac{6}{16} \log_2 \left(\frac{6}{16} \right) \approx 0.95$$

분류 모델

Decision Tree Classifier



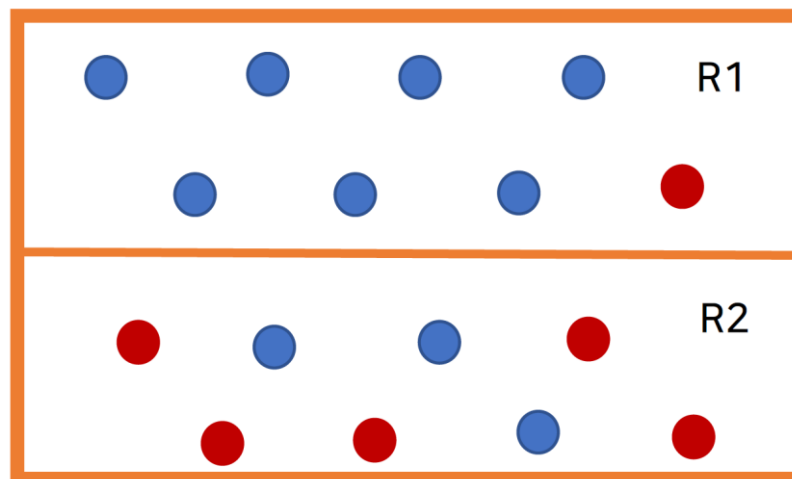
분기 된 영역의 Entropy는 가중평균을 통해 계산

$$\triangleright E(S, P) = \sum_p \frac{|P|}{|S|} Entropy(P)$$

S: 분기 전 영역 P: 분기 후 각 영역

분류 모델

Decision Tree Classifier



분기 후 Entropy

$$E(S, P) = -\frac{8}{16} \text{Entropy}(R_1) + \frac{8}{16} \text{Entropy}(R_2)$$

$$= 0.5 \times \left(-\frac{7}{8} \log_2 \left(\frac{7}{8} \right) - \frac{1}{8} \log_2 \left(\frac{1}{8} \right) \right) + 0.5 \times \left(-\frac{3}{8} \log_2 \left(\frac{3}{8} \right) - \frac{5}{8} \log_2 \left(\frac{5}{8} \right) \right) \approx 0.75$$

분류 모델

Decision Tree Classifier



분기 전: 0.95 → 분기 후: 0.75

트리 분기 후 엔트로피 0.2 정도 감소



트리 기반 분류 모델은 전체 불순도 합을 줄이는 방향으로 분류



더 이상 불순도 줄어들지 않을 때 분기를 멈춤

분류 모델

Decision Tree Classifier



분기 전: 0.9 → 분기 후: 0.75

트리 분기 후 엔트로피 0.15 정도 감소

Decision Tree에서

오차 또는 불순도가 줄어들지 않을 때

분기를 멈춘다고 했는데,

과연 분기를 계속 하는 것이 문제가 없을까?

다 이상 불순도 줄어들지 않을 때 분기를 멈춤

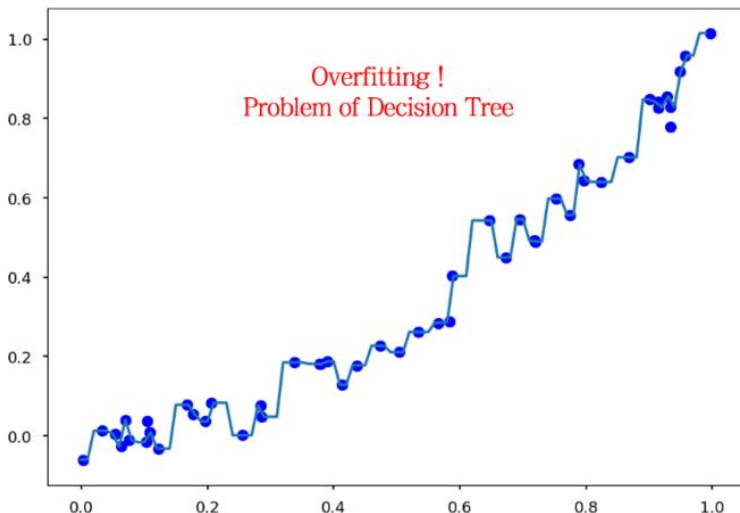
과적합 방지법

Avoid Overfitting in Tree-Based Models

분기를 하면 할수록 불순도는 계속 줄어듦



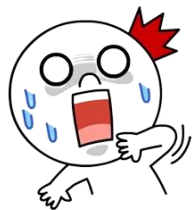
Overfitting이 발생!



Bias ↓
Variance ↑

“Non-Robust”

새로운 데이터에 강건하지 못하다



트리 모델의 과적합 방지법

Avoid Overfitting in Tree-Based Models

사전 가지치기

트리 모델의
하이퍼파라미터인
Tree의 깊이를
사전에 제한두는 것

V/S

사후 가지치기

Full Tree를 만든 후
적절한 수준에서 **Terminal**
Node를 **결합**하는 방법

트리 모델의 특성으로 인한 과적합을 방지하기 위한 방법으로
사전 가지치기 와 사후 가지치기가 있음

트리 모델의 과적합 방지법

사전 가지치기

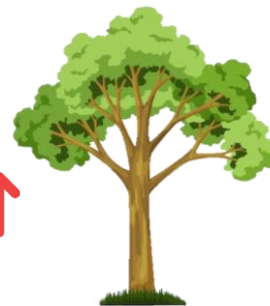
Max_depth

트리 기반 모델로 작업할 때 조정할 수 있는 **대표적인 하이퍼파라미터** 중 하나

큰 값

깊게 분기

모델 복잡도 ↑



작은 값

얕게 분기

모델 복잡도 ↓



트리 모델의 과적합 방지법

사전 가지치기

Max_depth

트리 기반 모델로 작업할 때 조정할 수 있는 **대표적인 하이퍼파라미터** 중 하나

큰 값

하이퍼파라미터 튜닝 등을 통해

모델 복잡도 ↑

적절히 값을 설정해주는 것이 중요!

작은 값

얇게 분기

모델 복잡도 ↓



트리 모델의 과적합 방지법

사후 가지치기

데이터에 대한 충분한 설명력을 가지며,
동시에 input space가 지나치게 세세하게 분류되는 것을 방지

트리를 **끝까지 분기**



밑에서부터 하나씩 **부모 노드의 Error** 값과 **자식 노드의 Error** 값을 비교



부모 노드 Error 보다 자식 노드 Error가 클 때 자식 노드 제거

트리 모델의 특성

가지치기의 한계

데이터의 작은 변화에도 tree의
최종 예측 값이 크게 변동함

트리 모델의 Non-Robust하고
Variance가 크다는 특성

모델이 과적합할 가능성이 큼

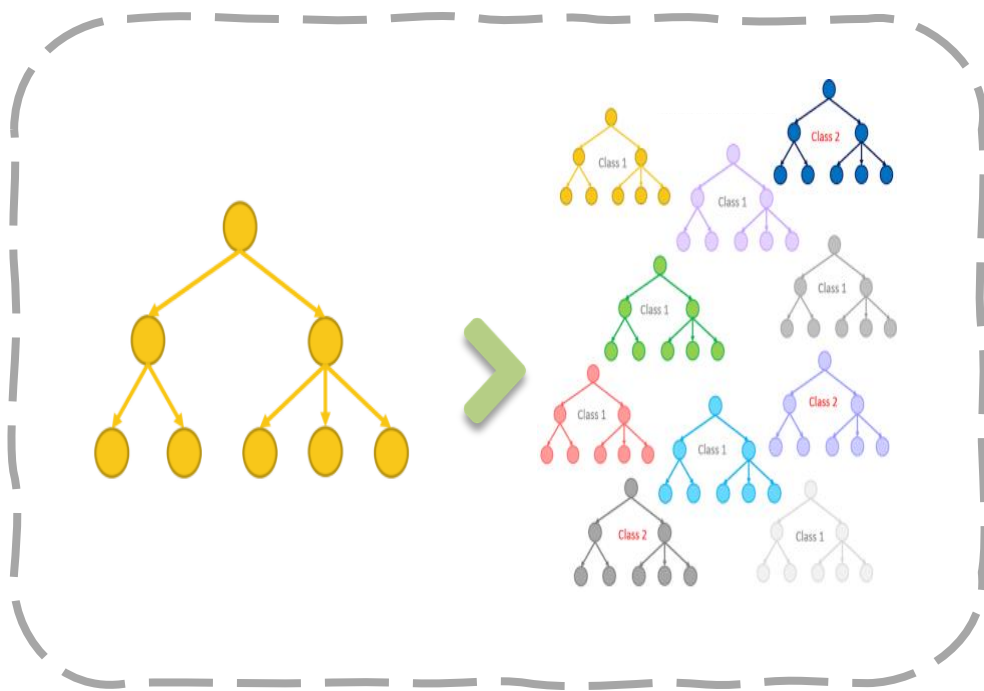


따라서, 단순히 가지치기를 한다고 해도 모델이 과적합에서 자유로울 수 없음

앙상블 기법(Ensemble Method)

앙상블 기법이란?

여러 개의 의사 결정 나무를 합쳐서 하나의 강한 모델을 만들어내는 방식



Bagging

RandomForest

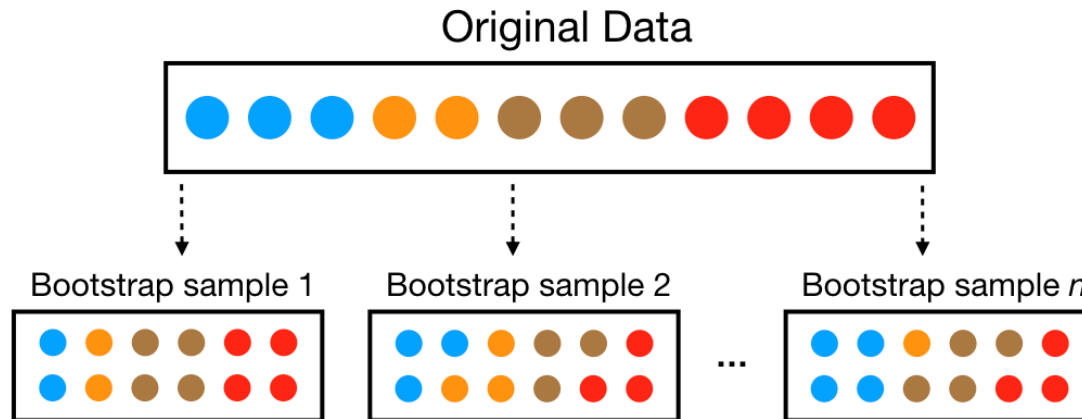
Boosting

GBM

AdaBoost

배깅 기법(Bagging Method)

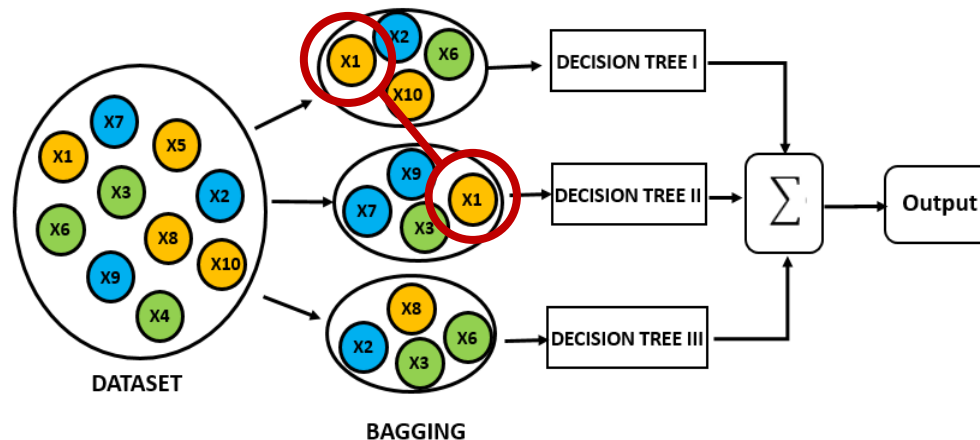
부트스트랩(Bootstrap)이란?



샘플을 추출할 때 복원추출이 가능하게 하여
샘플마다 중복 값이 존재하도록 만들어 줌

배깅 기법(Bagging Method)

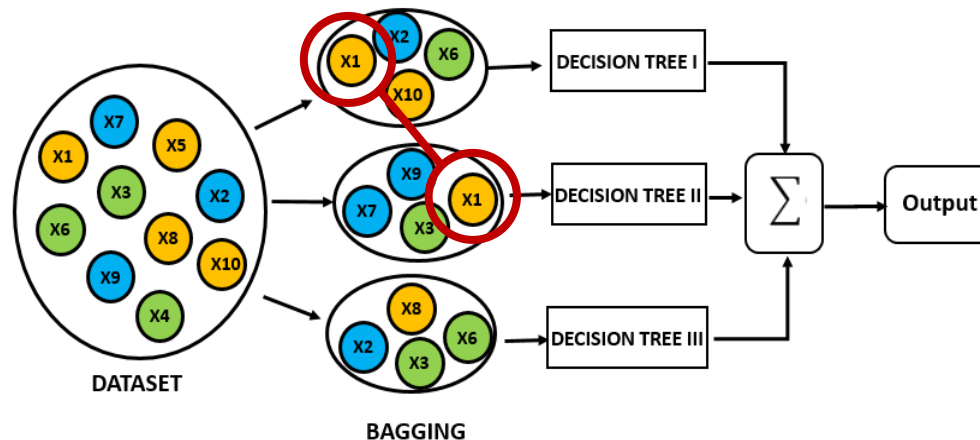
배깅(Bagging)이란?



여러 개의 의사결정 나무를 만들되, 각각의 N개의 의사결정 나무를
부트스트랩 기법으로 추출된 N개의 데이터 셋으로 학습을 시키는 기법

배깅 기법(Bagging Method)

배깅(Bagging)이란?



효과

각각 데이터셋에 중복의 값을 허용
→ N개의 모델들 간 분산을 줄임



한계

각각 데이터셋에 중복의 값을 허용
→ 최종 모델의 분산은 크게 줄어들지 않게 됨

배깅 기법(Bagging Method)

배깅(Bagging)이란?



각 모델의 분산을 줄이기 위해서
표본의 중복을 허용하는 것이
문제가 되는 이유??



효과

각각 데이터셋에 중복의 값을 허용

→ N개의 모델들 간 분산을 줄임



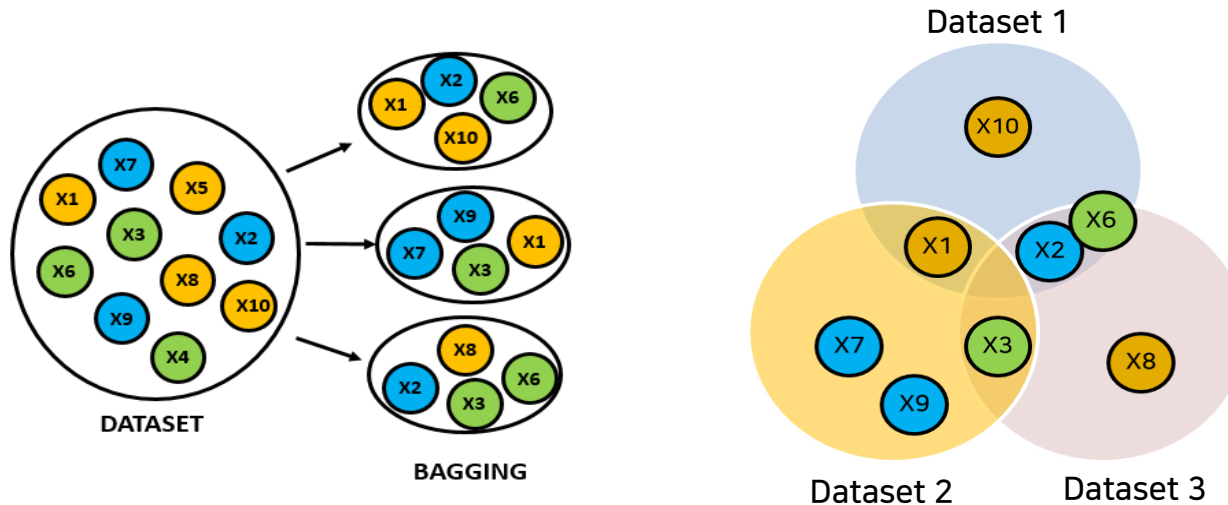
문제

각각 데이터셋에 중복의 값을 허용

→ 최종 모델의 분산은 크게 줄어들지 않게 됨

배깅 기법(Bagging Method)

배깅 기법의 한계



데이터의 중복 값을 허용하게 되면 생성된 데이터끼리 **Correlated** 됨

배깅 기법(Bagging Method)

배깅 기법의 한계

이 데이터를 학습한 N개의 모델들도 결국 유사한 형태



모델의 높은 Covariance를 야기

의도한 만큼 Variance가 줄어들지 않음

데이터의 중복 값을 허용하게 되면 생성된 데이터끼리 Correlated 됨

배깅 기법(Bagging Method)

배깅 기법의 한계

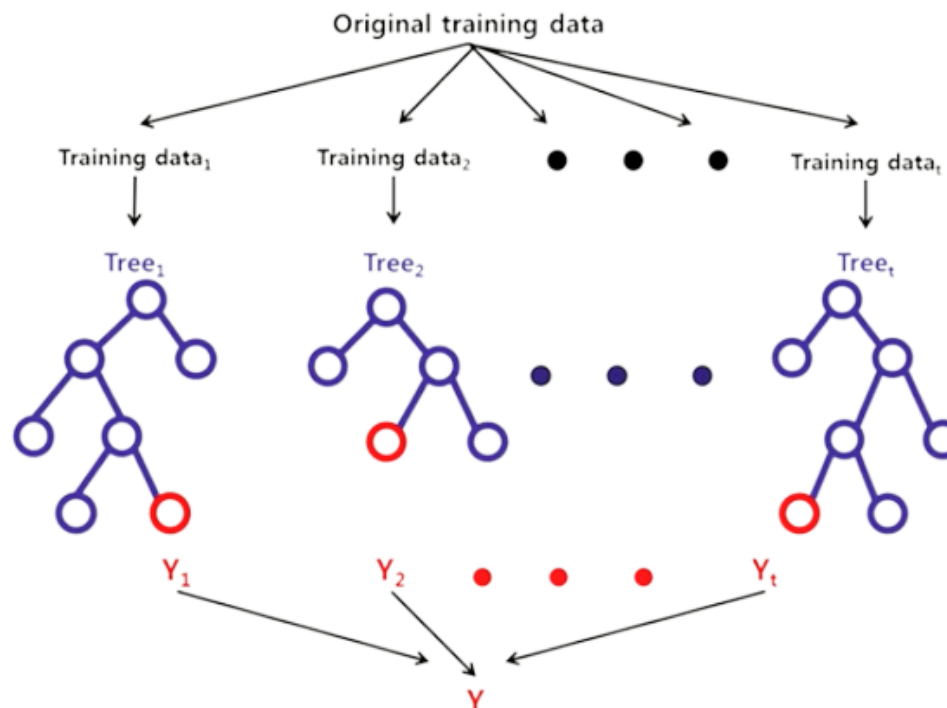


“ RandomForest ”

데이터의 중복 값을 허용하게 되면 생성된 데이터끼리 **Correlated** 됨

배깅 기법(Bagging Method)

랜덤포레스트(RandomForest) 모델

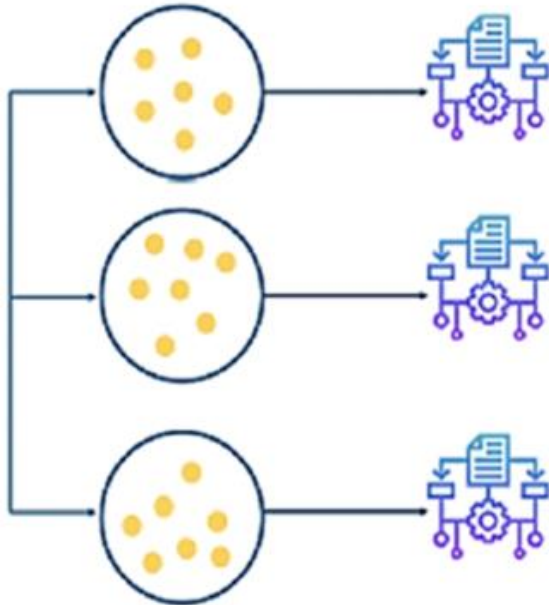


각 모델링마다 사용되는 Feature의 개수를 랜덤하게 선택함

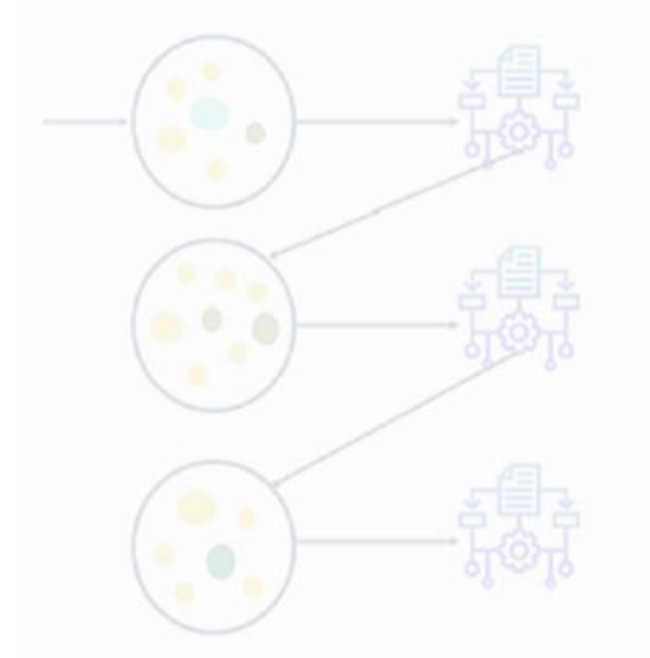
부스팅 기법(Boosting Method)

부스팅 기법 Vs. 배깅 기법

Bagging



Boosting

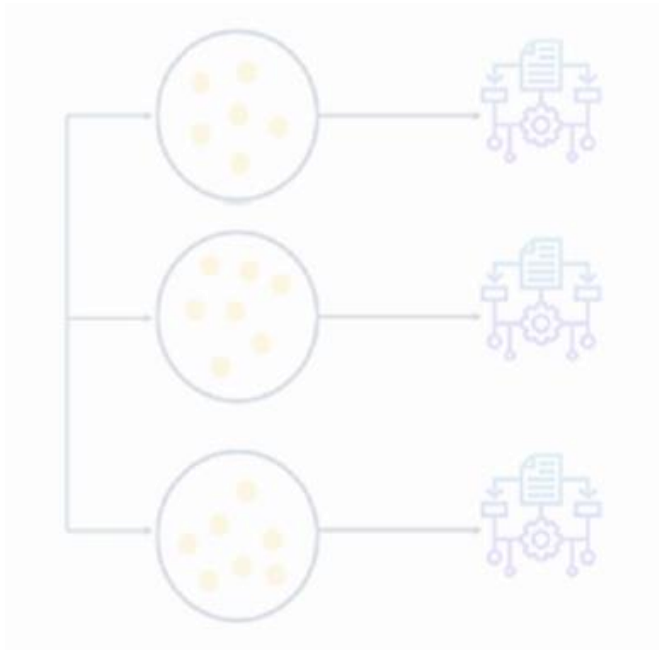


병렬적으로 생성된 각각의 트리 모델에 각기 다른 부분 데이터 셋을 적합

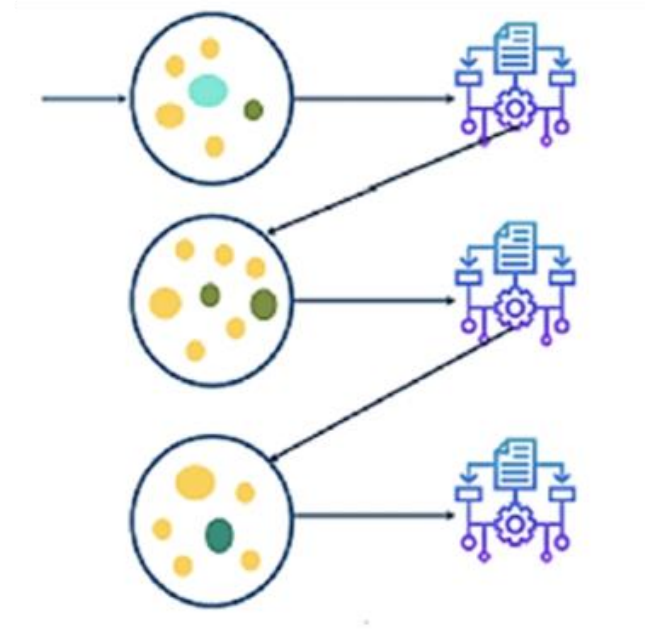
부스팅 기법(Boosting Method)

부스팅 기법 Vs. 배깅 기법

Bagging



Boosting



순차적으로 생성되는 트리 모델에 전체 데이터 셋을 적합

부스팅 기법(Boosting Method)

부스팅(Boosting)이란?



여러 개의 약한 트리(Weak Tree) 모델을 모아서
하나의 강한 모델을 만들어내는 기법

부스팅 기법(Boosting Method)

부스팅(Boosting)이란?



여러 개의 **약한 트리(Weak Tree) 모델**을 모아서
하나의 강한 모델을 만들어내는 기법

부스팅 기법(Boosting Method)

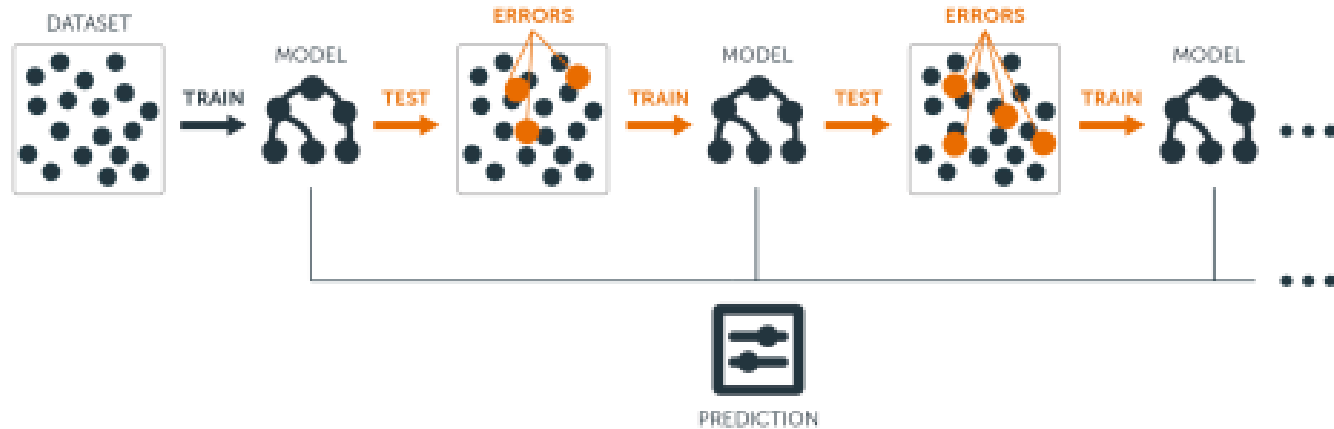
부스팅(Boosting)이란?



여러 개의 **약한 트리(Weak Tree) 모델**을 모아서
하나의 강한 모델을 만들어내는 기법

GBM

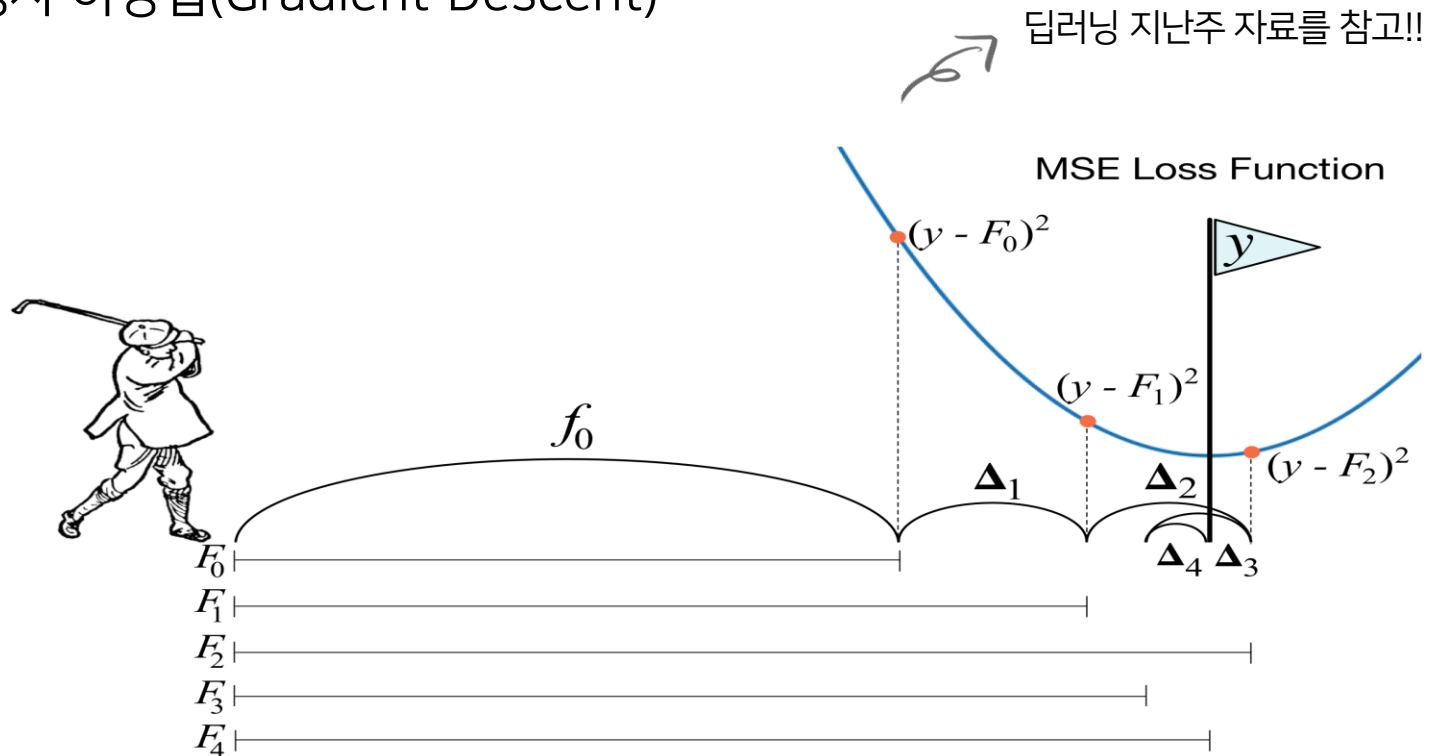
GBM(Gradient Boosting Method)이란?



- 경사하강법을 사용하여 **weak model들을 생성**하는 기법
 - 목적함수를 최소화 하는 방향으로 학습을 진행함

GBM

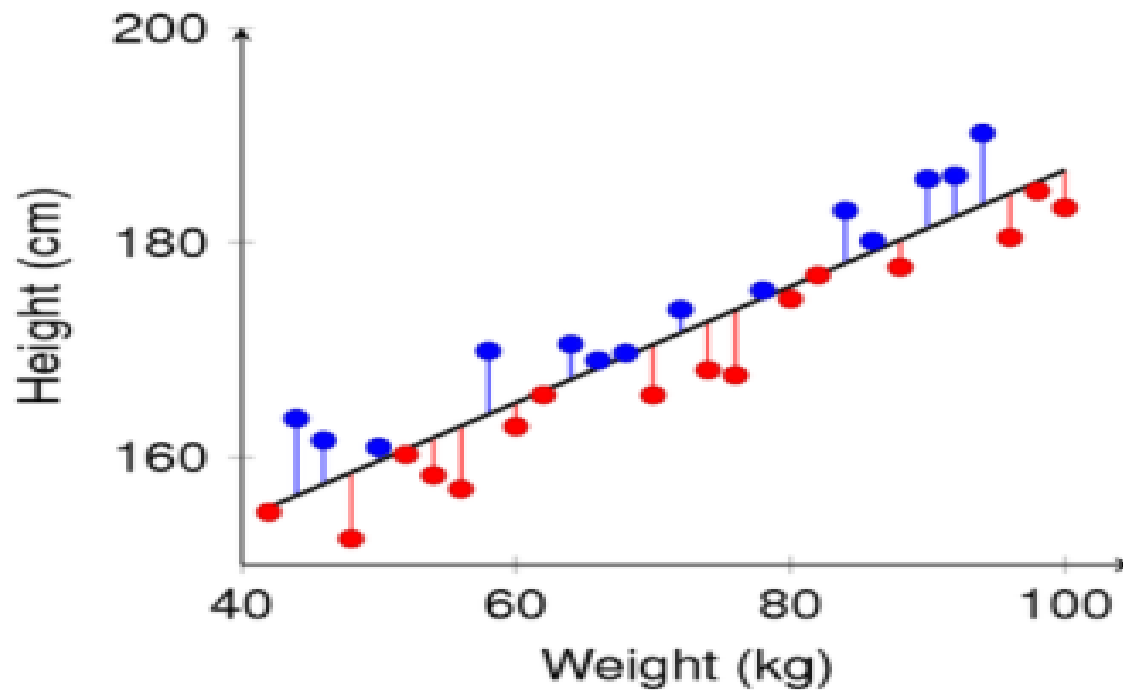
경사 하강법(Gradient Descent)



목적함수의 최솟값을 찾아내는 수치 해석적인 방법

GBM

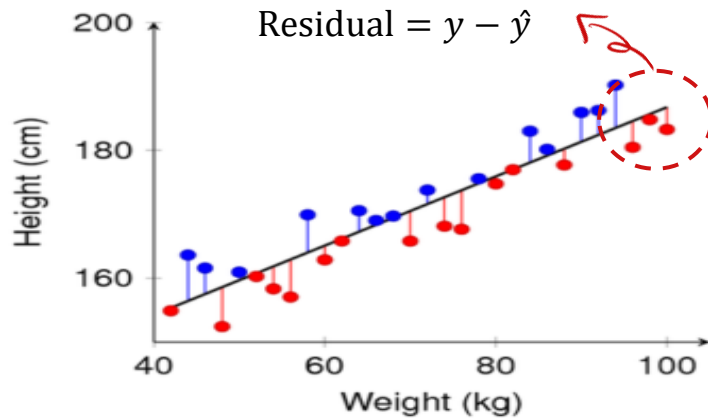
Negative Gradient



몸무게로 키를 예측하는 단순 회귀 모델을 생각해보자

GBM

Objective Function과 Negative Gradient



$$y = f_1(x) + r_1$$

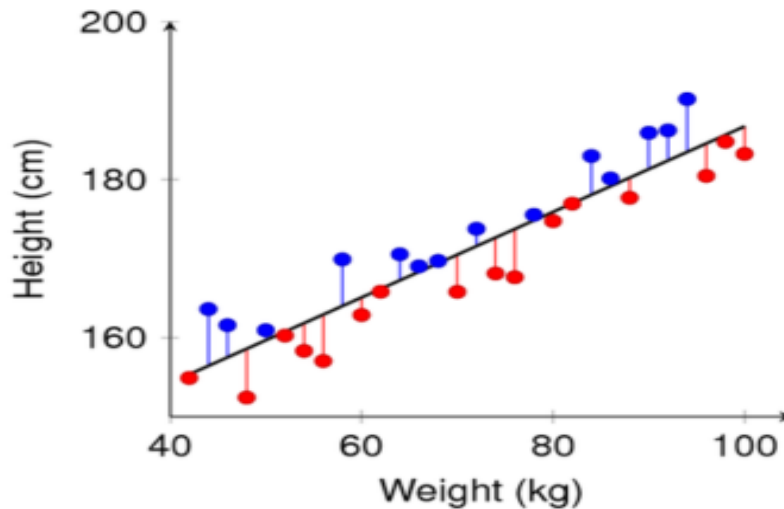
“Residual(잔차)”

$f_1(x)$ 모델이 미처 예측하지 못한 부분

- $f_i(x)$: 예측 함수(모델)의 예측값
 - r_i : i 번째 예측모델의 잔차
- 실제값 : $y = f_1(x) + r_1$

GBM

Negative Gradient



$$y = f_1(x) + r_1$$

$$r_1 = f_2(x) + r_2$$

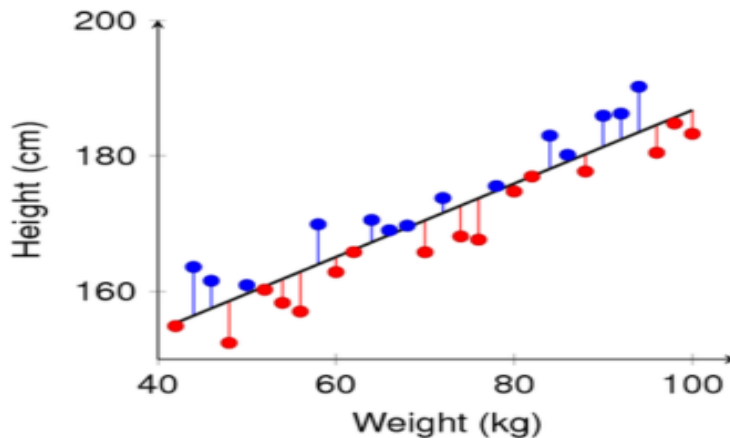
잔차 r_1 을 종속변수로 둔

새로운 모델 $f_2(x)$

미처 예측하지 못한 부분인
잔차도 예측해 내기 위해 모델을 추가 생성

GBM

Negative Gradient



$$y = f_1(x) + r_1$$

$$r_1 = f_2(x) + r_2$$

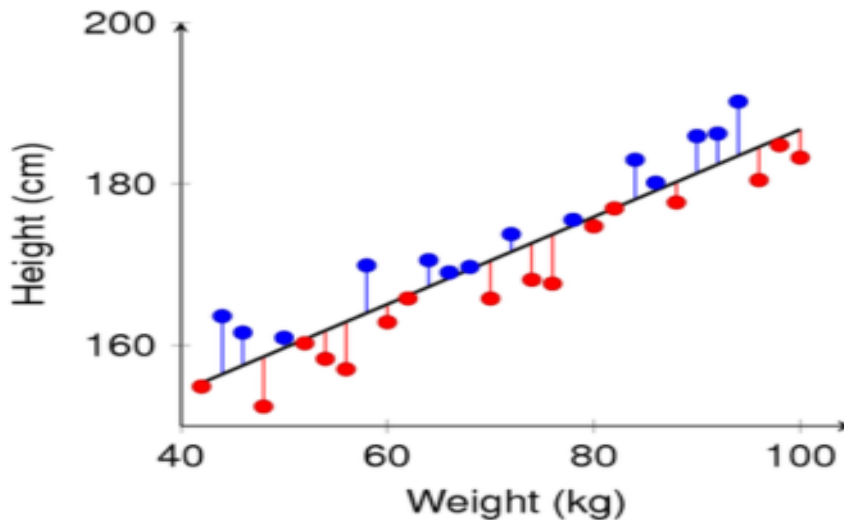


$$y = f_1(x) + f_2(x) + r_2$$

이 두 모델을 합치게 되면,
최종적으로 실제값은 생성된 두 개 모델의
예측값의 합에 잔차가 더해진 꼴로 모델로 구성됨

GBM

Negative Gradient



$$y = f_1(x) + f_2(x) + r_2$$

$$r_2 = f_3(x) + r_3$$

$$y = f_1(x) + f_2(x) + f_3(x) + r_3$$

그럼에도 불구하고 여전히 잔차는 존재하므로
해당 잔차를 예측하는 모델을 재생성하고 이 과정을 반복

GBM

Negative Gradient

$$y = f_1(x) + \text{Residual}_1$$

$$y = f_1(x) + f_2(x) + \text{Residual}_2$$

$$y = f_1(x) + f_2(x) + f_3(x) + \text{Residual}_3$$

...

$$y = f_1(x) + f_2(x) + \cdots + f_n(x) + \text{Residual}_n$$



굉장히 작은
잔차만 생존

잔차들을 모델 학습마다 구해내는게 힘들

→ 잔차를 경사하강법을 통해 구해낸

Negative Gradient로 대체해서 사용

GBM

Negative Gradient

$$y = f_1(x) + \text{Residual}_1$$

$$y = f_1(x) + f_2(x) + \text{Residual}_2$$

그렇다면 잔차는 경사하강법을 통해서 구해진

Negative Gradient와 같다고 말할 수 있을까?

$$y = f_1(x) + f_2(x) + \dots + f_n(x) + \text{Residual}_n$$

잔차들을 모델 학습마다 구해내는게 힘들

→ 잔차를 경사하강법을 통해 구해낸

Negative Gradient로 대체해서 사용

GBM

Objective Function과 Negative Gradient

$$\text{loss function} = \frac{1}{2} (y_i - f(x_i))^2$$

“실제값”

“함수 $f(x_i)$ 를 통해 예측한 값”

목적함수로 **Squared Error**를 사용한다고 가정

GBM

Objective Function과 Negative Gradient

$$\text{loss function} = \frac{1}{2} (y_i - f(x_i))^2$$

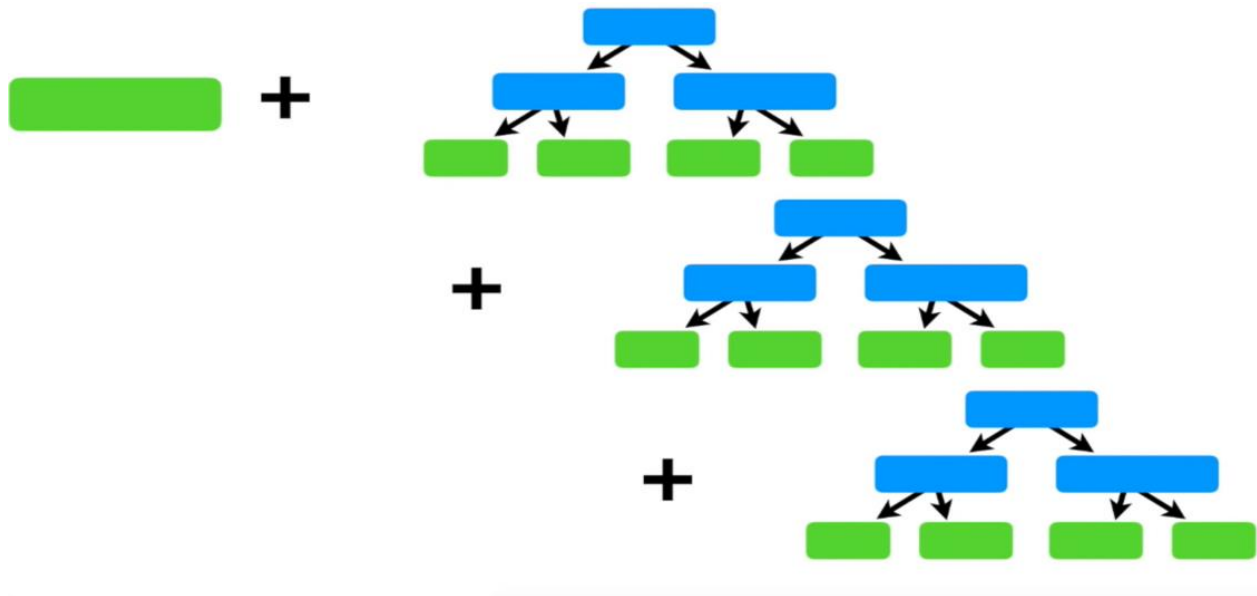
$$\begin{aligned} \text{negative gradient} &= - \frac{\partial(\text{loss function})}{\partial f(x_i)} = - (f(x_i) - y_i) \\ &= \text{residual} \end{aligned}$$

목적함수를 편미분하여 Negative Gradient를 구하면

Residual, 즉 잔차가 나오는 것을 확인 가능

GBM

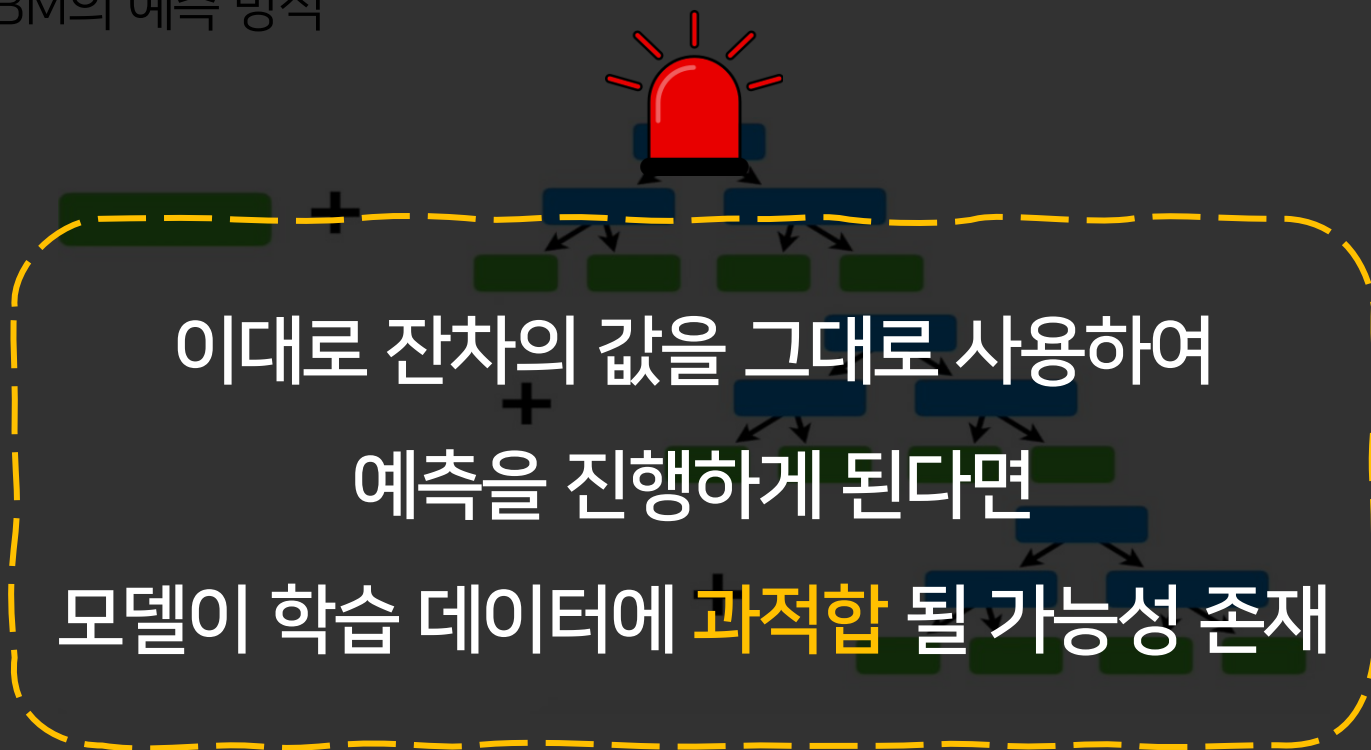
GBM의 예측 방식



경사 하강법으로 생성된 여러 개의 Weak Tree에
전체 데이터를 순차적으로 적합시켜 예측 값을 출력

GBM

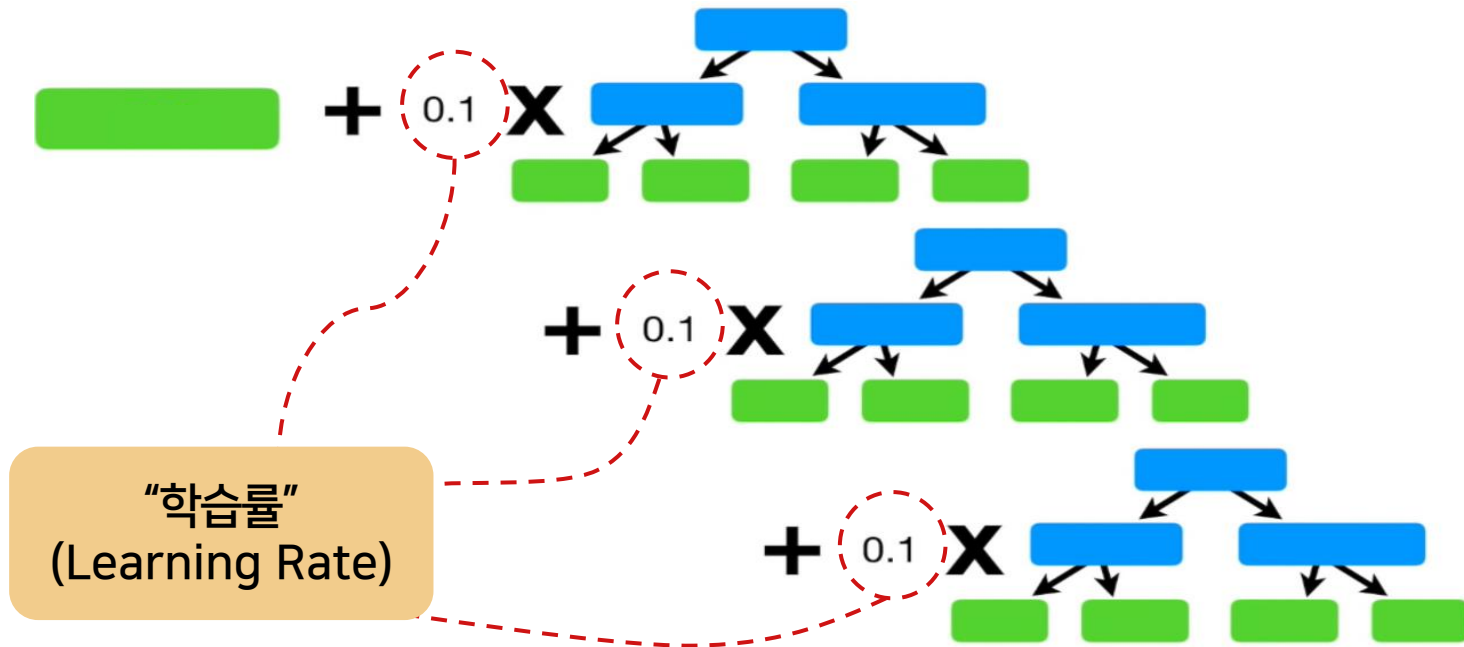
GBM의 예측 방식



경사 하강법으로 생성된 여러 개의 Weak Tree에
전체 데이터를 순차적으로 적합시켜 예측 값을 출력

GBM

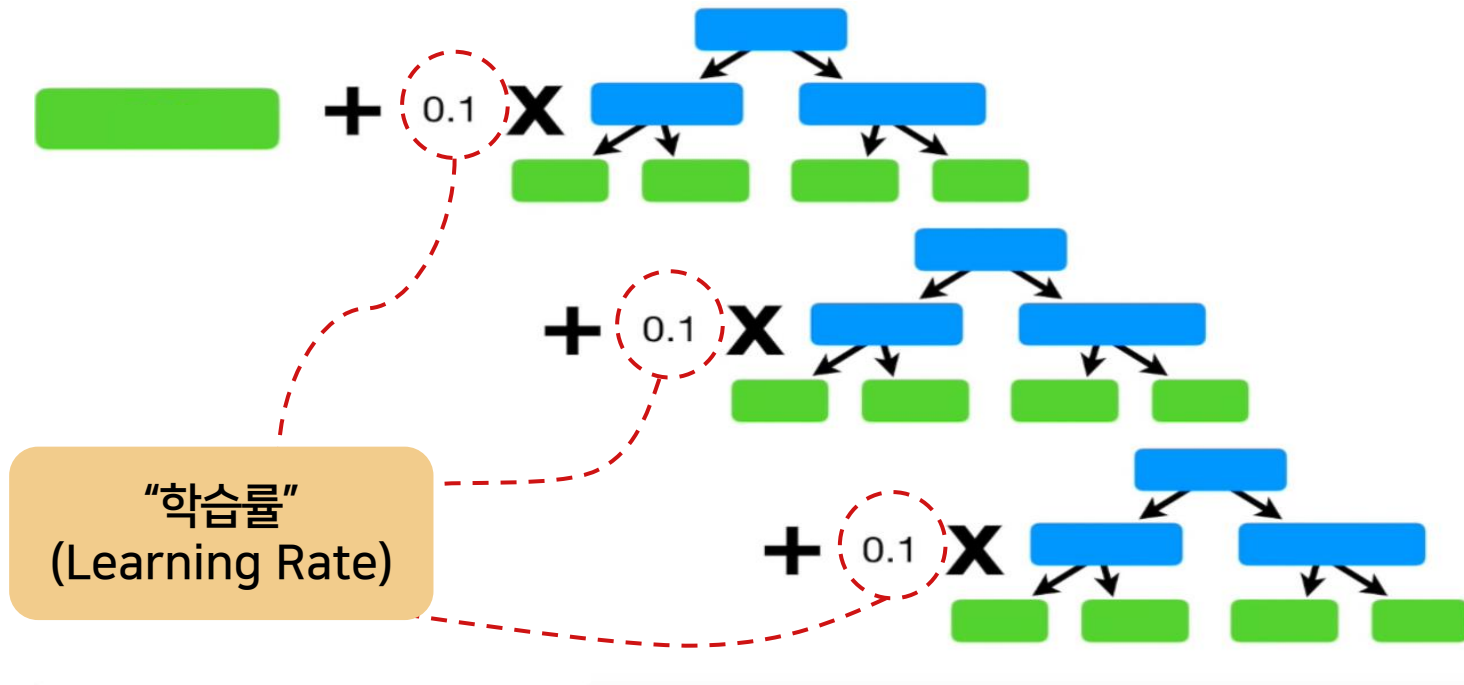
GBM의 예측 방식



따라서, 예측 값에서 잔차 값을 그대로 사용하는 것이 아닌
잔차에 **학습률**을 곱한 값을 사용하여 과적합을 방지

GBM

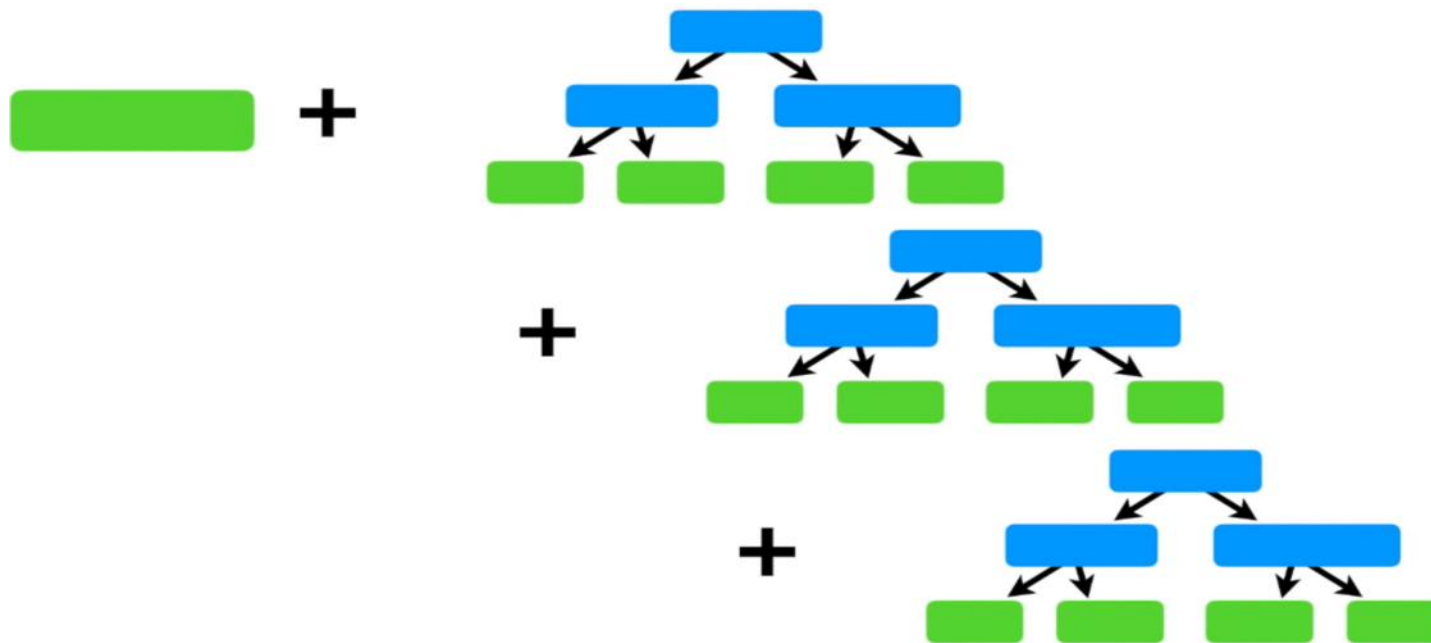
GBM의 예측 방식



학습률의 값에 따라 모델의 성능이 달라짐

GBM

GBM의 예측 방식



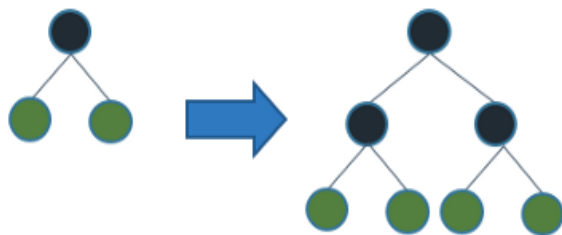
목적 함수를 미분가능한 다른 Metric으로 설정하면
회귀 문제가 아닌 분류 문제의 경우에도 사용 가능

GBM

GBM의 예측 방식

XGBOOST

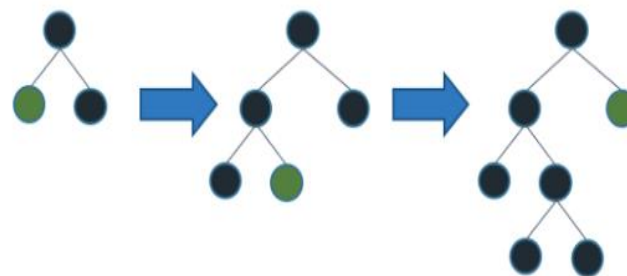
Level-wise tree growth



트리가 균형적인 형태로
분할하는 방식을 채택

LightGBM

Leaf-wise tree growth



트리가 불균형적인 형태로
분할하는 방식을 채택

2

생성 모델

Review: 판별 모델 vs 생성 모델

판별 모델

Discriminative Model

$P(y = j|X)$ 를 직접 구하여 이를 바탕으로 분류를 진행

생성 모델

Generative Model

$P(y = j|X)$ 를
 $P(y = j)$ 와 $P(X|y = j)$ 를 통해
간접적으로 구하여
이를 바탕으로 분류를 진행

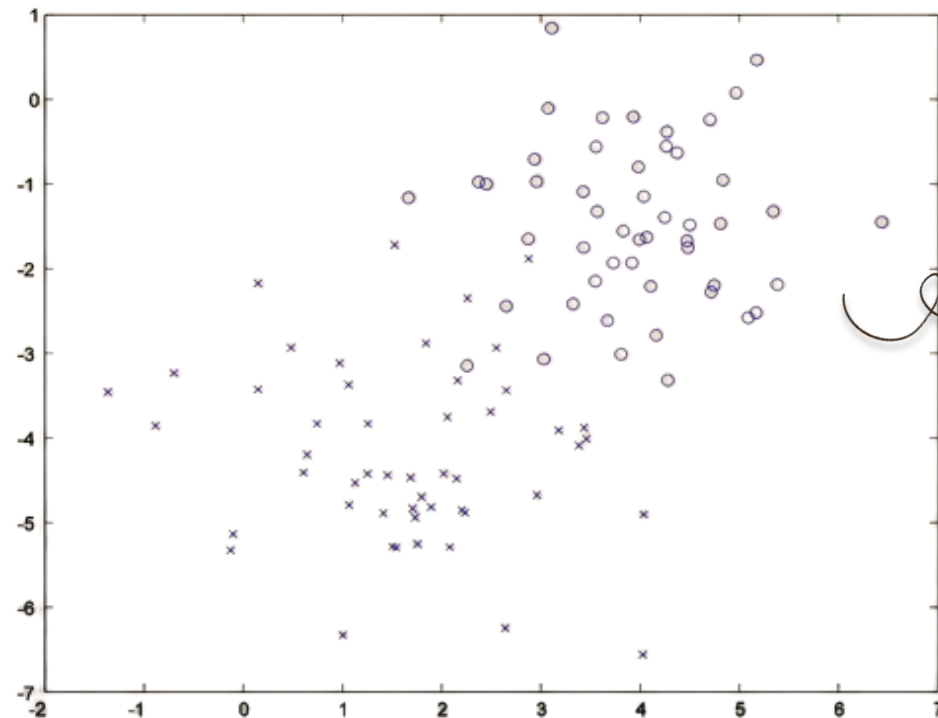


이 때 생성모델에서는 베이즈 정리를 통해 확률을 **간접적**으로 계산!

이번 주차: 데이터에 **라벨링이 되어있는** 경우 (지도학습)

다음 주차: 데이터에 **라벨링이 되어있지 않은** 경우(비지도학습)

GDA



X=0, O=1로
인코딩!

다음과 같은 데이터가 주어졌다고 가정
이 데이터를 분류하기 위해 **선형의 결정경계**를 얻어내야 함

GDA

선형의 결정경계를 형성하기 위해

① $P(y)$: y 가 0 또는 1일 확률

② $P(X|y)$: y 가 0 또는 1일 때,
그 때의 X 의 분포 계산!

$X=0, 0=1$ 로
인코딩!



간접적으로 $P(y|X)$ 를 구할 수 있음

다음과 같은 데이터가 주어졌다고 가정

이 데이터를 분류하기 위해 선형의 결정경계를 얻어내야 함

GDA

GDA의 기본 가정

GDA의 분포가정

$$y \sim \text{Bernoulli}(\phi)$$

$$X|y = 1 \sim \text{MVN}(\mu_1, \Sigma)$$

$$X|y = 0 \sim \text{MVN}(\mu_0, \Sigma)$$

$P(y)$ 와 $P(X|y)$ 를 계산하기 위해서

y 는 베르누이 분포, $X|y$ 는 다변량 정규분포를 따른다고 가정


GDA

GDA의 기본 가정

GDA의 분포가정

$$y \sim \text{Bernoulli}(\phi)$$

$$X|y = 1 \sim \text{MVN}(\mu_1, \Sigma)$$

$$X|y = 0 \sim \text{MVN}(\mu_0, \Sigma)$$


이때 선형의 결정 경계를 형성하기 위해서
두 다변량 정규분포의 **공분산은 동일**하다고 가정

GDA

GDA의 파라미터 추정

GDA의 MLE 추정량

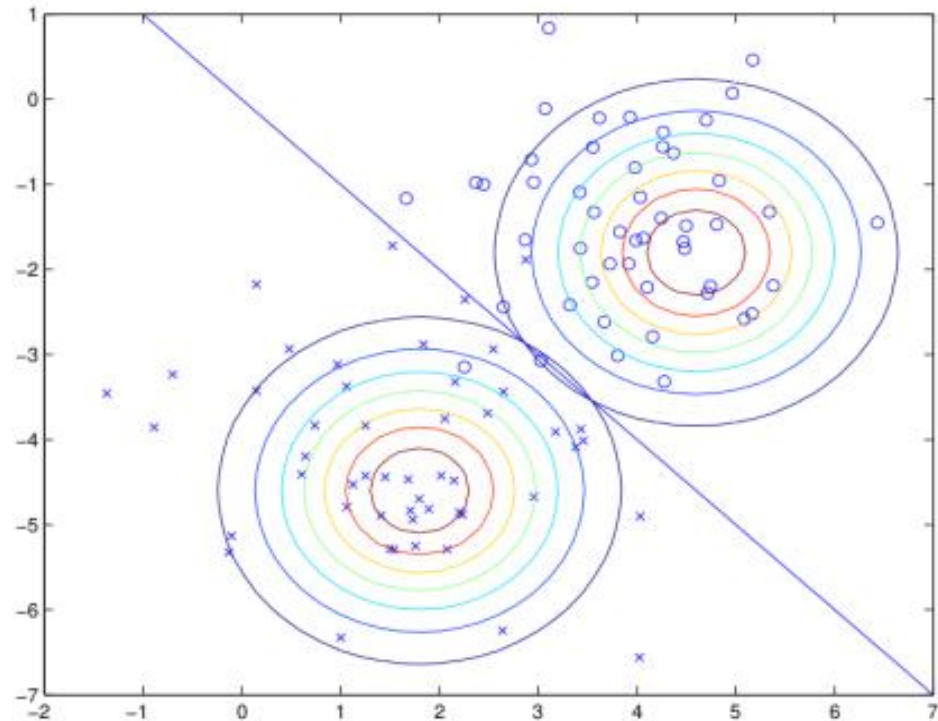
$$\begin{aligned}\phi: \frac{\sum_{i=1}^m I(y^{(i)} = 1)}{m} &= \frac{\sum_{i=1}^m y^{(i)}}{m} = \bar{Y} \\ \mu_1 &= \frac{\sum_{i=1}^m I(y^{(i)} = 1) x^{(i)}}{\sum_{i=1}^m I(y^{(i)} = 1)} & \mu_0 &= \frac{\sum_{i=1}^m I(y^{(i)} = 0) x^{(i)}}{\sum_{i=1}^m I(y^{(i)} = 0)} \\ \Sigma &= \frac{\sum_{i=1}^m (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T}{m}\end{aligned}$$

Likelihood를 최대화하는 방향으로 학습하면

다음과 같은 MLE 추정량을 얻어낼 수 있음

GDA

GDA와 선형 결정 경계

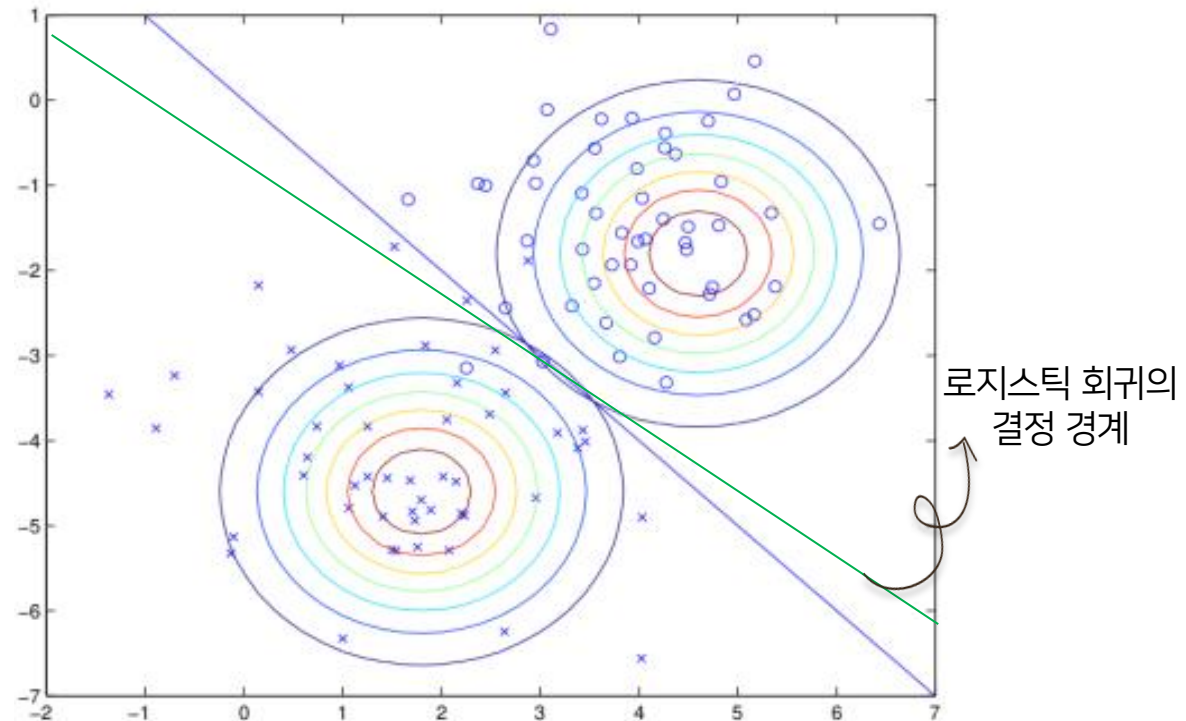


파라미터를 추정하고 나면, 2개의 다변량 정규분포를 통해

선형의 결정 경계를 얻어낼 수 있음

GDA

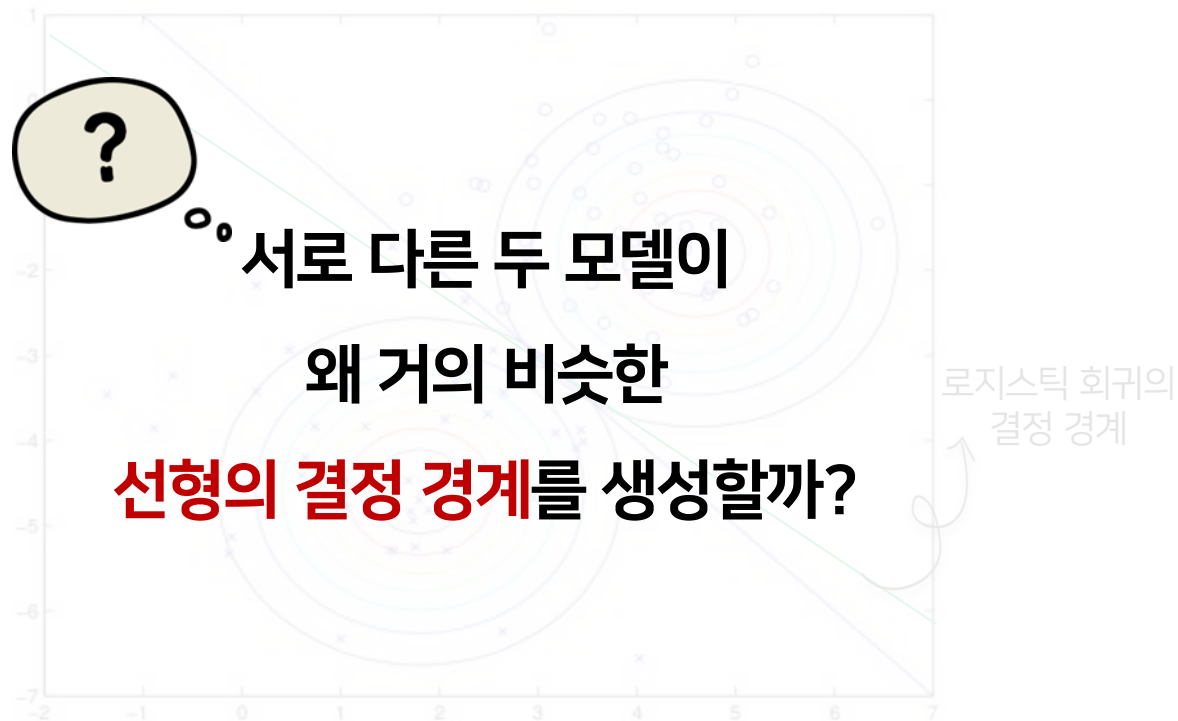
GDA와 선형 결정 경계



이를 로지스틱 회귀를 통해 얻은 결정경계와 비교해보면
두 모델의 결정 경계가 **거의 일치하는 것**을 확인할 수 있음

GDA

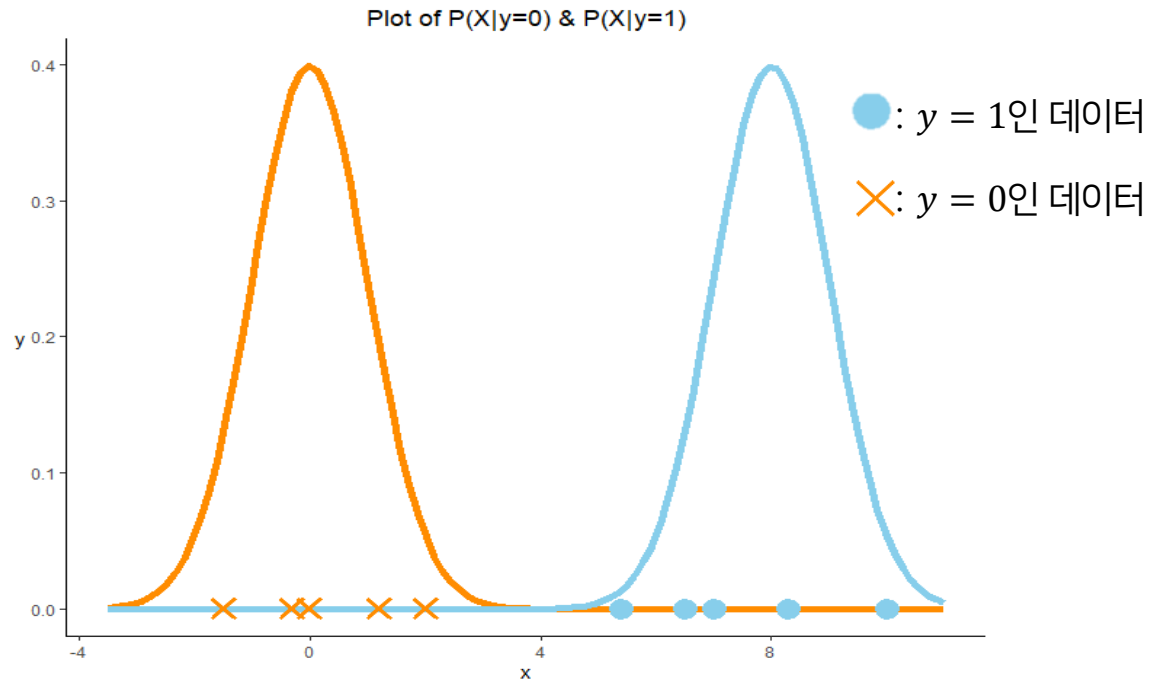
GDA와 선형 결정 경계



이를 로지스틱 회귀를 통해 얻은 결정경계와 비교해보면
두 모델의 결정 경계가 거의 일치하는 것을 확인할 수 있음

GDA

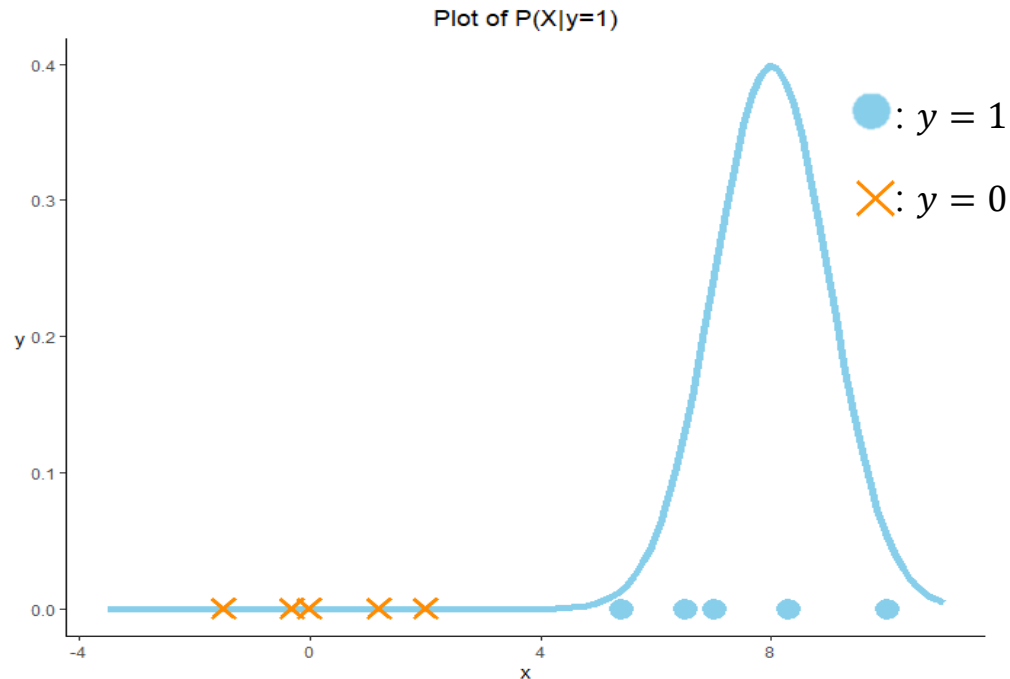
GDA와 선형 결정 경계



데이터를 1차원에서 바라보았을 때,
 $X|y = 0$ 과 $X|y = 1$ 는 각각 정규분포를 통해
다음과 같이 데이터를 생성 해냄

GDA

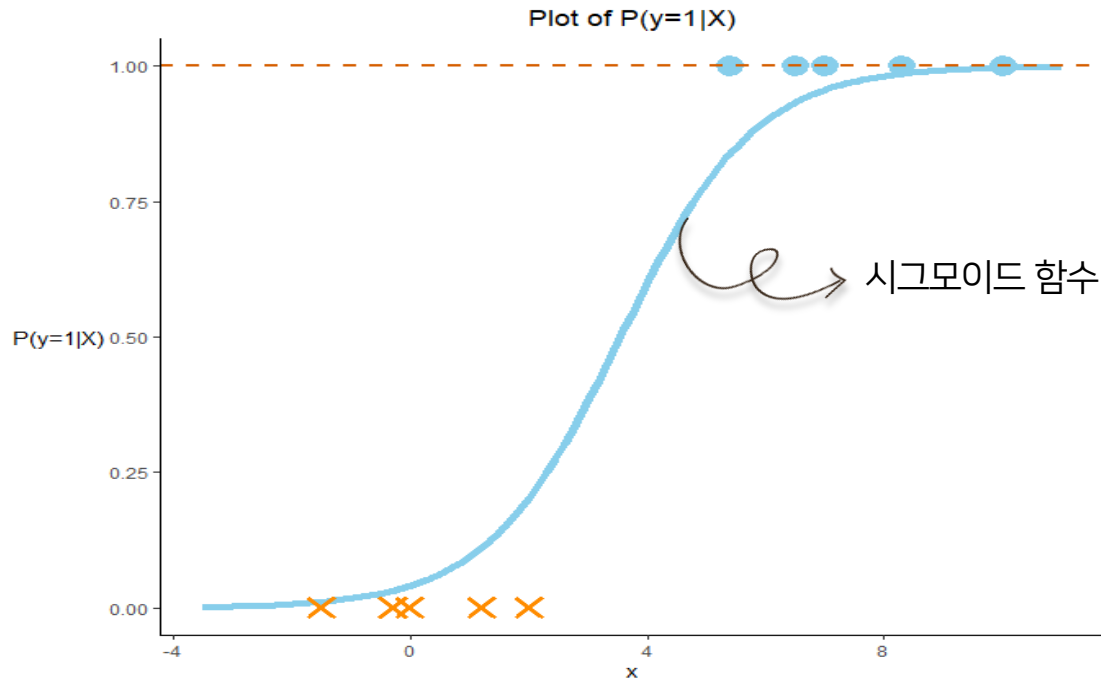
GDA와 선형 결정 경계



$X|y = 1$ 의 분포를 살펴보면,
 $y = 0$ 인 데이터보다 $y = 1$ 인 데이터가
생성될 확률이 좀더 높게 나타남

GDA

GDA와 선형 결정 경계

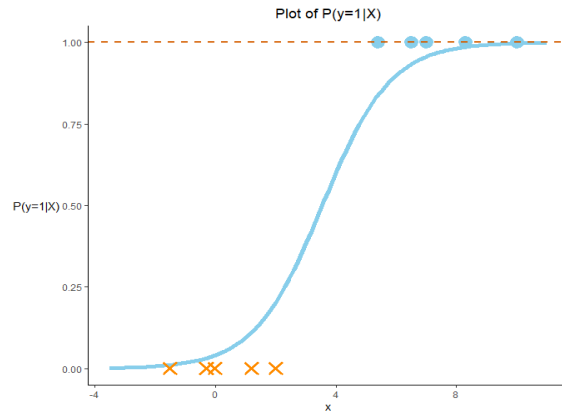


$P(y = 1)$ 와 $P(X|y = 1)$ 를 통해 $P(y = 1|X)$ 를 계산하면
로지스틱 회귀와 같은 방식으로 확률을 계산됨을 확인 가능

➡ 로지스틱 회귀와 같이 선형의 결정경계가 생성!

GDA

GDA와 선형 결정 경계



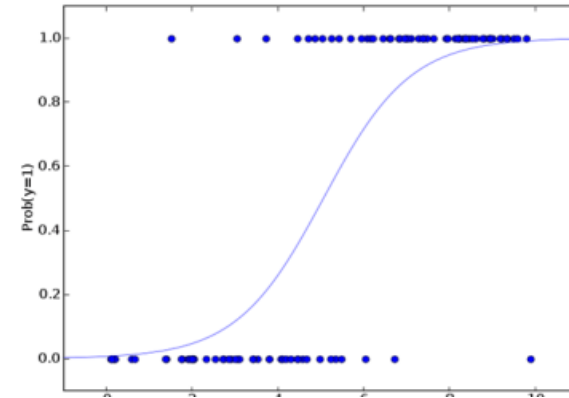
GDA

$$y \sim \text{Bernoulli}(\phi)$$

$$x|y = 1 \sim \text{MVN}(\mu_1, \Sigma)$$

$$x|y = 0 \sim \text{MVN}(\mu_0, \Sigma)$$

implies


Logistic
Regression

$$P(y = 1|X) = \phi = \frac{1}{1 + e^{X\beta}}$$

따라서 다음과 같은 명제가 성립하게 됨

GDA

GDA의 장점과 한계

장점

분포 가정이 맞을 때 뛰어난 성능을 보임

→ y 와 $X|y$ 의 분포 가정만 맞다면 학습 데이터 수가 적을 때
Logistic Regression보다 훨씬 좋은 성능을 보임

한계

① 데이터가 정규분포를 통해 생성되지 않았을 때 성능이 저하됨

→ 일반적으로는 Logistic Regression이 좀 더 Robust한 모델임

② 모든 독립변수들이 연속형 자료일 때만 사용 가능

Naive Bayes

Naive Bayes

The diagram shows the Naive Bayes formula $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ with four blue arrows pointing to its components: 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Likelihood

Class Prior Probability

Posterior Probability

Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

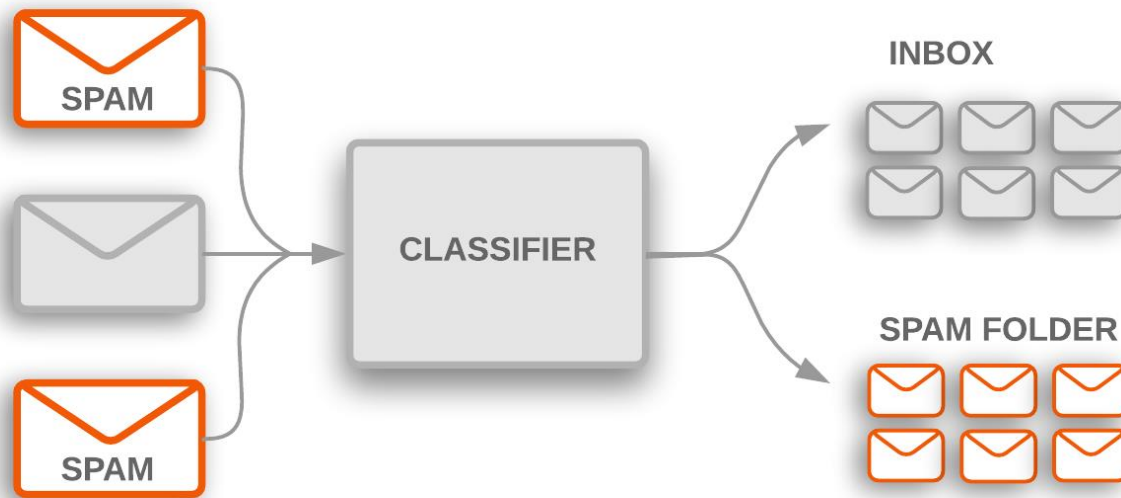
베이즈 정리를 바탕으로, **각 변수들 간의 조건부 독립**을 가정하여

베이즈 정리의 복잡한 조건을 완화한 생성 모델

주로 텍스트 데이터 분류에 많이 사용됨

Naive Bayes

Naive Bayes



스팸메일을 받으면 이를 자동으로 분류하는 모델을 만든다고 가정
이를 위해 어떤 단어들이 등장할 때, 그 메일이 **스팸메일일 확률**을 계산

Naive Bayes

Naive Bayes

인덱싱 된 단어 10000개

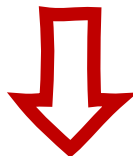
m개의 이메일		스팸여부(y)	강의(x_1)	광고(x_2)	...	보험(x_{5714})	...
	이메일 1	1(스팸)	0	1	...	1	...
	이메일 2	0(정상)	1	0	...	1	...
	⋮	⋮	⋮	⋮	...	⋮	⋮
	이메일 m	0	1	1	...	0	...

이를 위해 나이브 베이즈 모델은 m개의 이메일에서
10000개 단어의 출현 여부를 학습하게 됨

Naive-Bayes

$$P(y|X) = E[I(y|X)], \text{ where } X = [x_1, x_2, \dots, x_{10000}]$$

이를 로지스틱 회귀로 분류한다면, 설명변수(10000개)가 많아
예측을 위해 많은 파라미터를 필요로 함 → 효과적으로 분류를 진행하기 어려움



$$\operatorname{argmax}_y P(y|X) = \operatorname{argmax}_y P(y)P(X|y) = \operatorname{argmax}_y P(X, y)$$

Generative하게 접근한다면
좀더 쉽고 효과적으로 분류 가능!

Naive-Bayes

여기서 문제!



$P(y|X) = E[I(y|X)]$, where $X = [x_1, x_2, \dots, x_{10000}]$
 $X = [x_1, x_2, \dots, x_{10000}]$ 였기 때문에, **베이즈 정리**에 의해 $P(X|y)$ 는

이를 로지스틱 회귀로 분류한다면, 설명변수(10000개)가 많아
 $P(X, y) = P(y)P(x_1|y)P(x_2|y, x_1) \cdots P(x_{10000}|x_{9999}, \dots, x_2, x_1, y)$
 예측을 위한 **파라미터**가 필요함 → 효과적으로 분류를 진행하기 어려움



$$\operatorname{argmax}_y P(y|X) = \operatorname{argmax}_y P(y)P(X|y) = \operatorname{argmax}_y P(X, y)$$

이를 계산하는 것은 아까 말했던 Logistic Regression으로 이 이메일들을

분류하는 것처럼 **효과적이지 못하고**, 매우 복잡한 연산과정을 요구할 것

Generative하게 접근한다면
 좀더 쉽고 효과적으로 분류 가능!

Naive-Bayes

Naive-Bayes

$$\begin{aligned} p(y|X_1, X_2, \dots, X_n) \\ &\propto p(y, X_1, X_2, \dots, X_n) \\ &\propto p(y)p(X_1|y)p(X_2|y)p(X_3|y) \cdots \\ &\propto p(y) \prod_{i=1}^n p(X_i|y) \end{aligned}$$

단순히 y 가 주어졌을 때, x_j 끼리는 독립이라는 조건부 독립을 가정

→ 베이즈 법칙의 조건을 완화하여

기존의 매우 복잡한 연산을 비교적 간단하게 바꿀 수 있음

"Naive"
(순진한)



Naive-Bayes

Naive-Bayes의 기본 가정

Naive-Bayes의 분포가정

$$\phi_y = P(y = 1) \rightarrow y \sim \text{Bernoulli}(\phi_y)$$

$$\phi_{j|y=1} = P(x_j = 1|y = 1) \rightarrow (x_j|y = 1) \sim \text{Bernoulli}(\phi_{j|y=1})$$

$$\phi_{j|y=0} = P(x_j = 1|y = 0) \rightarrow (x_j|y = 0) \sim \text{Bernoulli}(\phi_{j|y=0})$$

$P(y)$ 와 $P(x_j|y)$ 를 계산하기 위해서 분포를 가정할 때,

y 는 스팸메일 여부, $x_j|y$ 는 스팸메일 여부가 주어졌을 때 단어의 출현 여부이므로

y 와 $x_j|y$ 의 분포 모두 베르누이 분포를 가정함

Naive-Bayes

Naive-Bayes의 파라미터 추정

Naive-Bayes 의 MLE 추정량

$$\widehat{\phi_y^{MLE}} = \frac{\sum_{i=1}^m I(y^{(i)} = 1)}{m}$$

$$\widehat{\phi_{j|y=1}^{MLE}} = \frac{\sum_{i=1}^m I(x_j^{(i)} = 1, y^{(i)} = 1)}{\sum_{i=1}^m I(y^{(i)} = 1)}$$

$$\widehat{\phi_{j|y=0}^{MLE}} = \frac{\sum_{i=1}^m I(x_j^{(i)} = 1, y^{(i)} = 0)}{\sum_{i=1}^m I(y^{(i)} = 0)}$$

Likelihood를 최대화하는 방향으로 학습하면

다음과 같은 MLE 추정량을 얻어낼 수 있음

Naive-Bayes

Naive-Bayes의 장점

Naive-Bayes 의 MLE 추정량

각 파라미터의 추정치가 단순히

- ① 전체 이메일 수 (m)
- ② 스팸메일 수 ($\sum_{i=1}^m I(y^{(i)} = 1)$)
- ③ 스팸메일에서 해당 단어 j 가 등장한 이메일 수 ($\sum_{i=1}^m I(x_j^{(i)} = 1, y^{(i)} = 1)$)
- ④ 정상메일에서 해당 단어 j 가 등장한 이메일 수 ($\sum_{i=1}^m I(x_j^{(i)} = 1, y^{(i)} = 0)$)

이는 단순히 숫자만 세어주면 되기 때문에

학습속도도 빠르고, 적용하기도 쉽다는 장점을 가지게됨



THANK YOU

