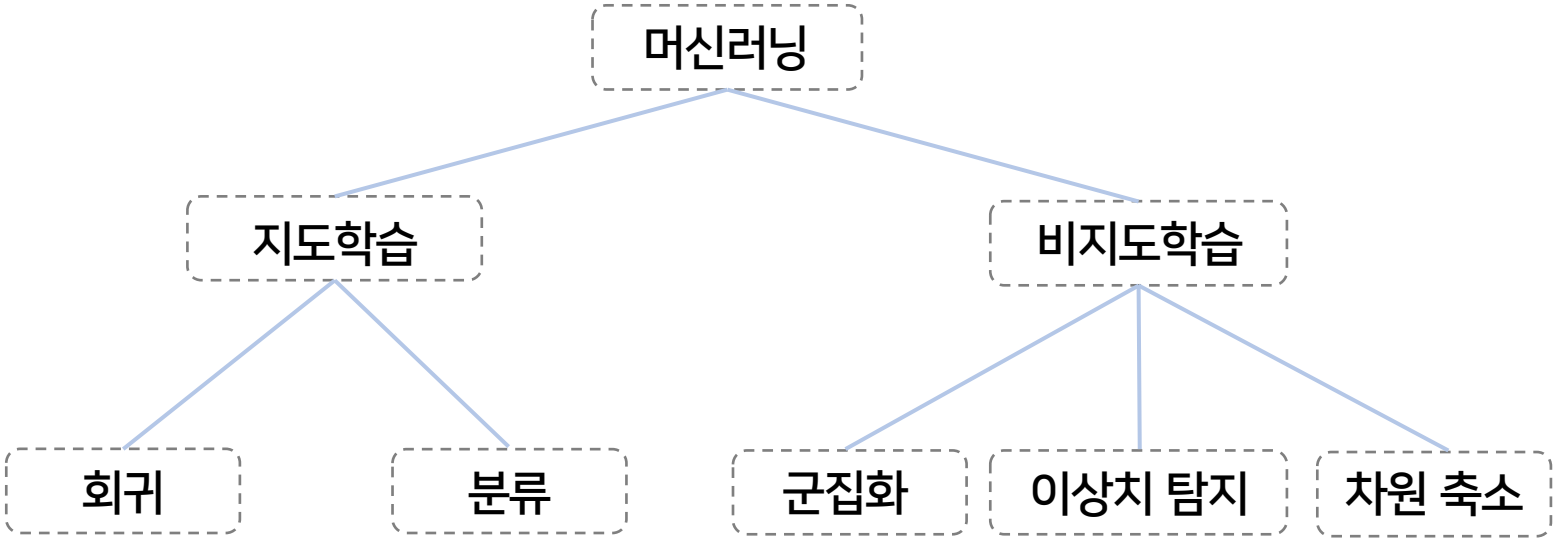
데이터마이닝팀

4팀
오주원
이동기
김형석
이경미
최종혁

INDEX

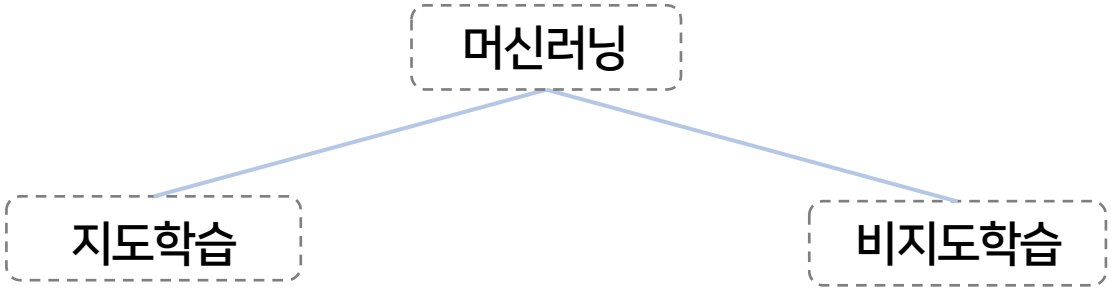
1 트리 기반 모델

개념



1 트리 기반 모델

개념



회귀 분류 군집화 이상치 탐지 차원 축소

트리 기반 모델은 지도 학습의 대표적인 모델임

그 중에서도 가장 대표적인 모델은 **의사결정나무(Decision Tree)**

1 트리 기반 모델

의사결정나무(Decision Tree)

의사결정나무

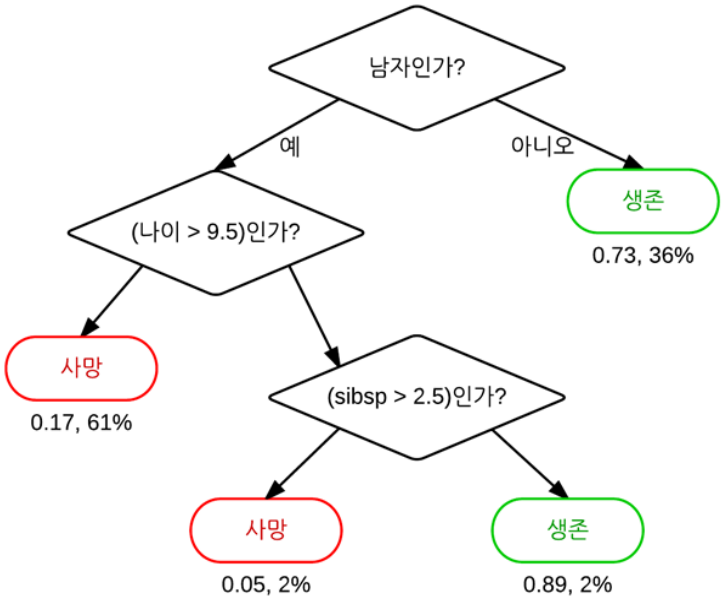
데이터에 내재되어 있는 패턴을 **예측 가능한 규칙들의 조합**으로 나타냄

흔히 아는 스무고개 놀이와 비슷한 개념

1 트리 기반 모델

의사결정나무(Decision Tree)

Ex) 타이타닉 프로젝트



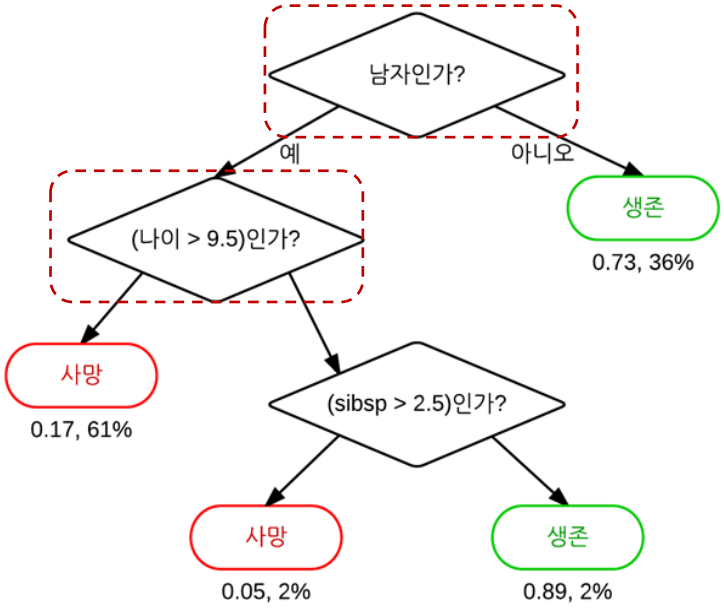
타이타닉 탑승자들의
정보를 바탕으로
생존 여부를 예측

질문을 던져서 대상을
좁혀나감

1 트리 기반 모델

의사결정나무(Decision Tree)

Ex) 타이타닉 프로젝트

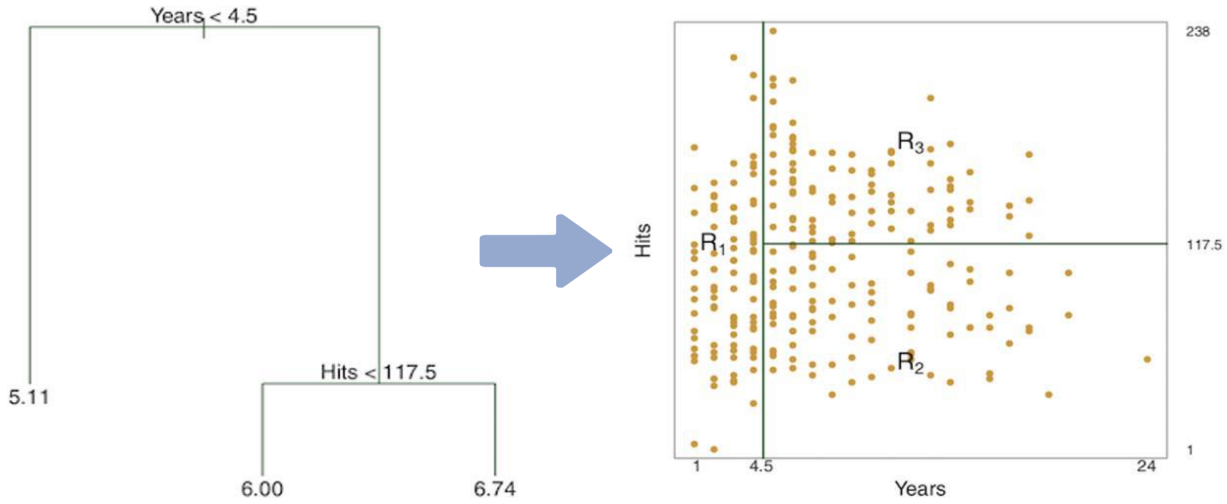


'남자인가?'라는 질문에 대해서 '예' 와 '아니오'에 따라서 **영역이 분할**

'나이>9.5?'라는 질문에 대해 '예'와 '아니오'에 따라 다시 영역이 분할

1 트리 기반 모델

의사결정나무(Decision Tree)

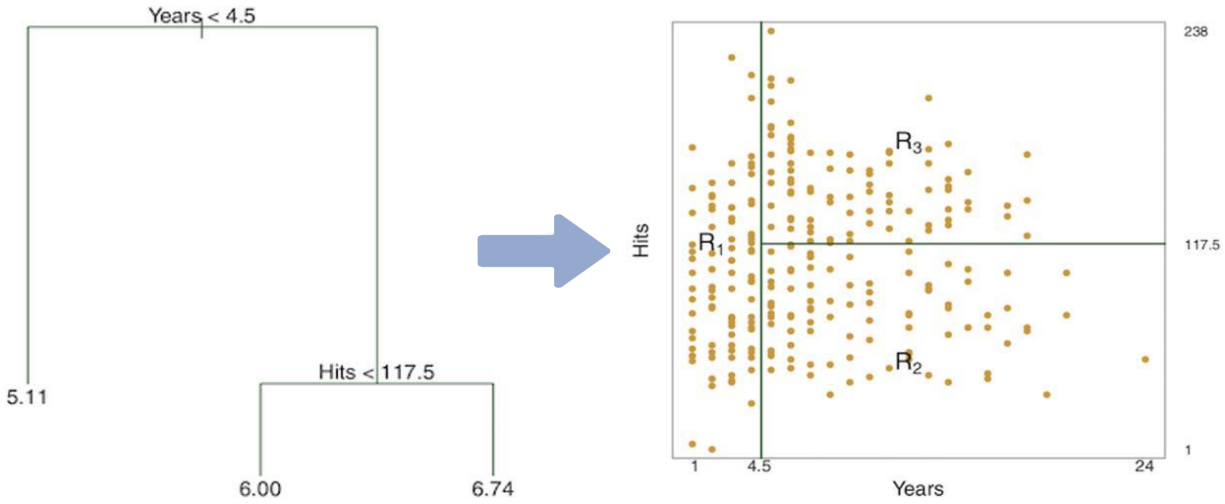


이러한 과정은 **순차적으로 독립변수의 공간을 분할**한다고

표현할 수 있음

1 트리 기반 모델

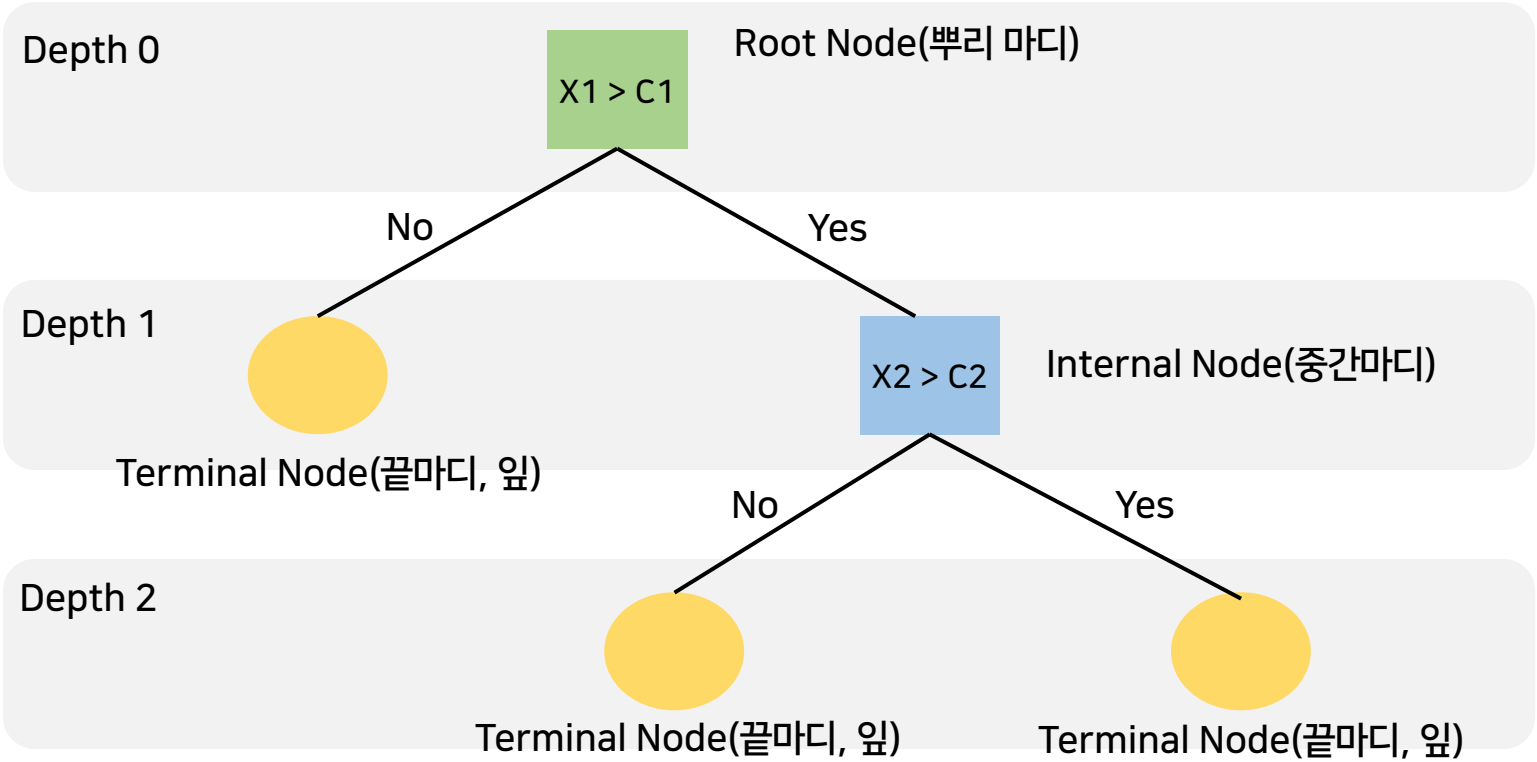
의사결정나무(Decision Tree)



독립변수(X)들이 차지하고 있는 공간을 질문을 통해 분할해 나가고,
그렇게 분할된 공간에 **적절한 결론**을 내려주는 것이 핵심

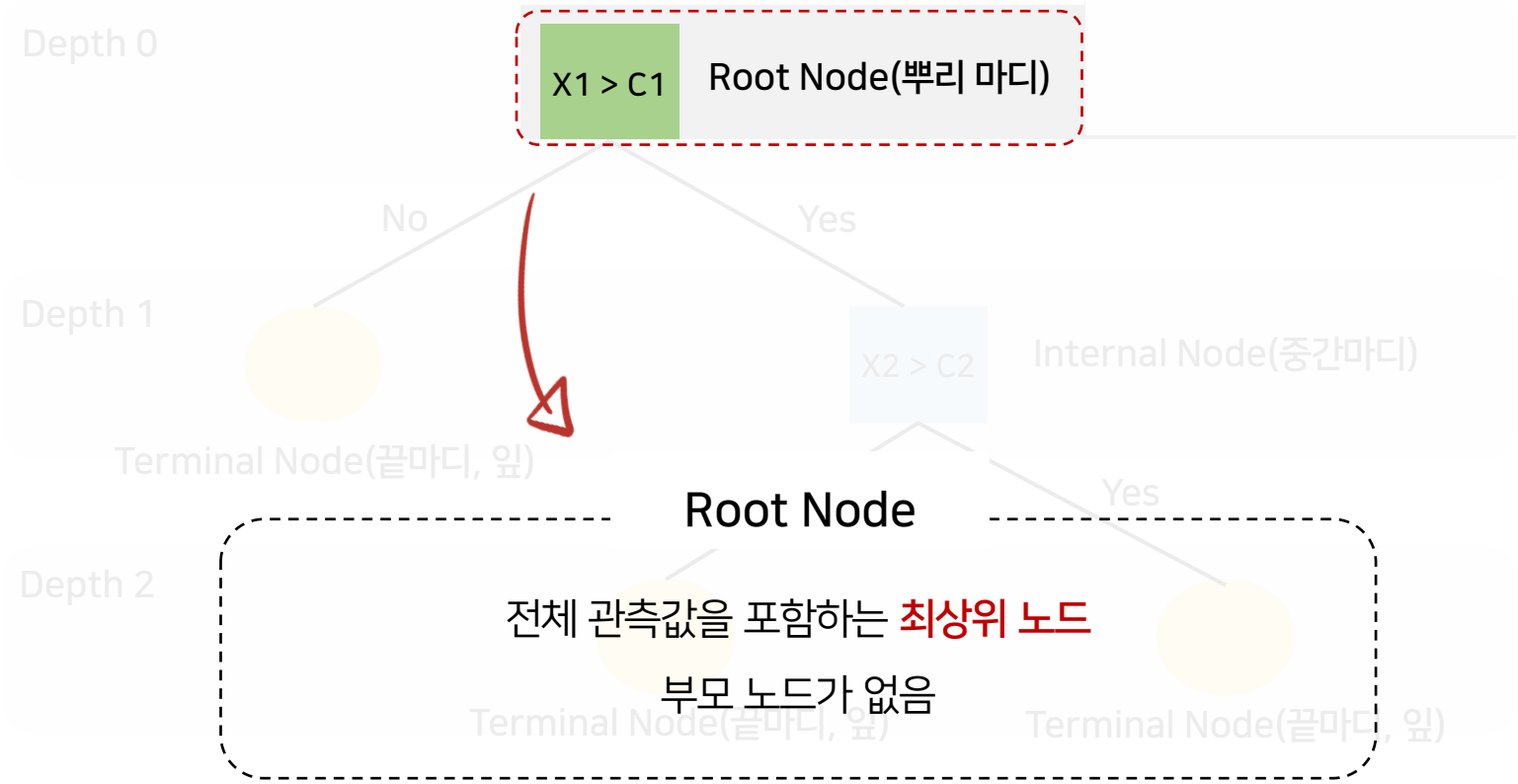
1 트리 기반 모델

의사결정나무(Decision Tree)



1 트리 기반 모델

의사결정나무(Decision Tree)



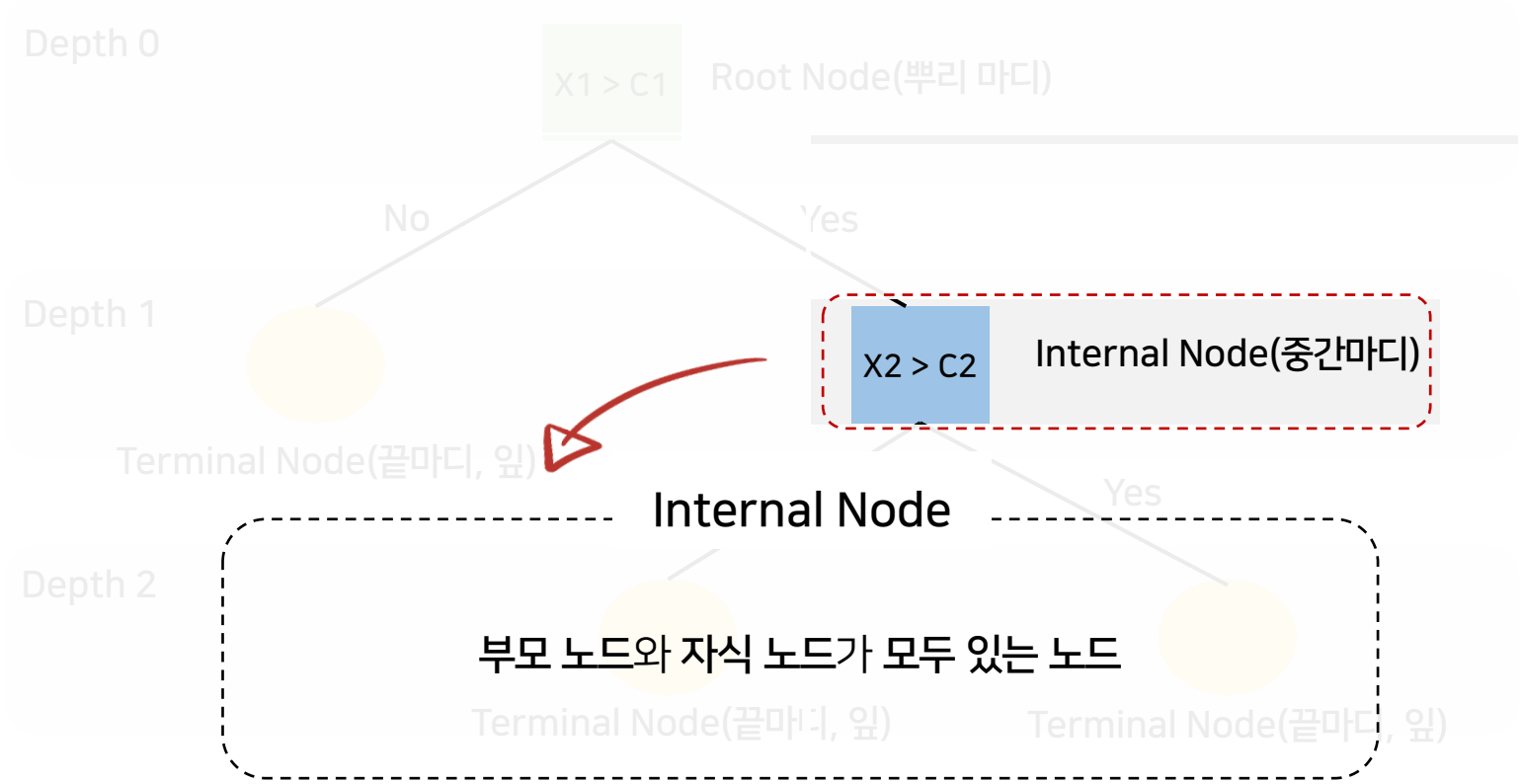
Root Node

전체 관측값을 포함하는 **최상위 노드**

부모 노드가 없음

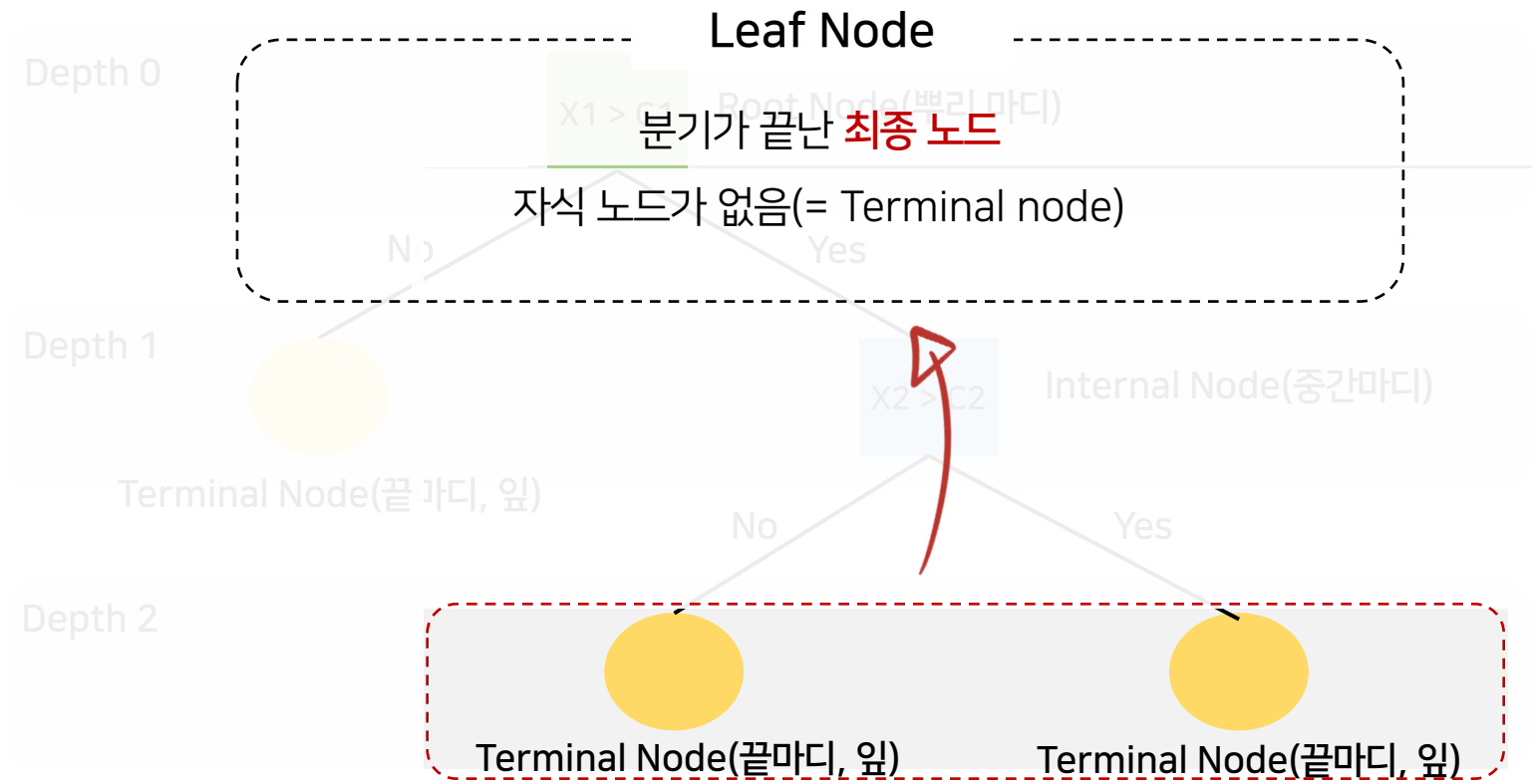
1 트리 기반 모델

의사결정나무(Decision Tree)



1 트리 기반 모델

의사결정나무(Decision Tree)

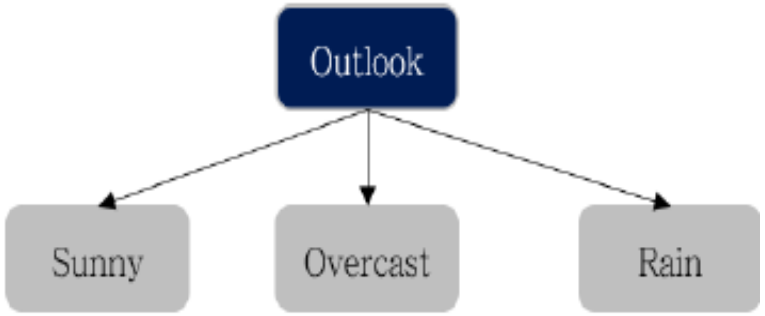


1 트리 기반 모델

의사결정나무(Decision Tree)

본 클린업에서는 회귀 모델로는 CART,
분류 모델로는 ID3의 관점을 바라보고 전개

ID3

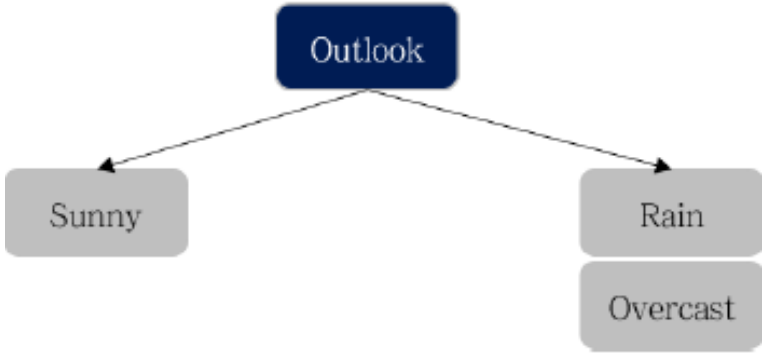


독립변수가 **범주형**인 경우에만 사용

한 노드를 **여러 개로 분기** 가능

분기 기준은 **엔트로피(Entropy)**

CART



독립변수는 **연속형**과 **범주형** 모두 가능

한 노드를 **두 가지로만 분기** 가능

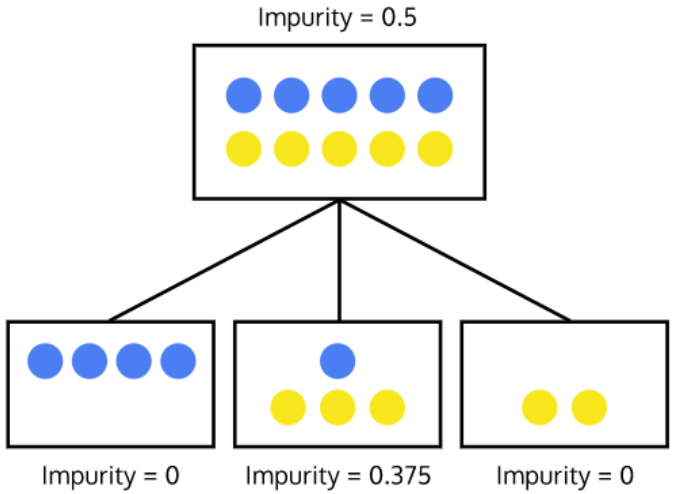
분기 기준은 **지니 계수(Gini index)**

1 트리 기반 모델

Decision Tree Classifier

분기 기준: 불순도

Decision Tree Classifier는 **영역들의 불순도(Impurity)**를 기준으로 분기



분기된 영역 내에서 **서로 다른**
범주들의 개체들이 얼마나 포함되어 있는가,
불순도가 높을수록 분류가 잘 되지 않았다고 판단

1 트리 기반 모델

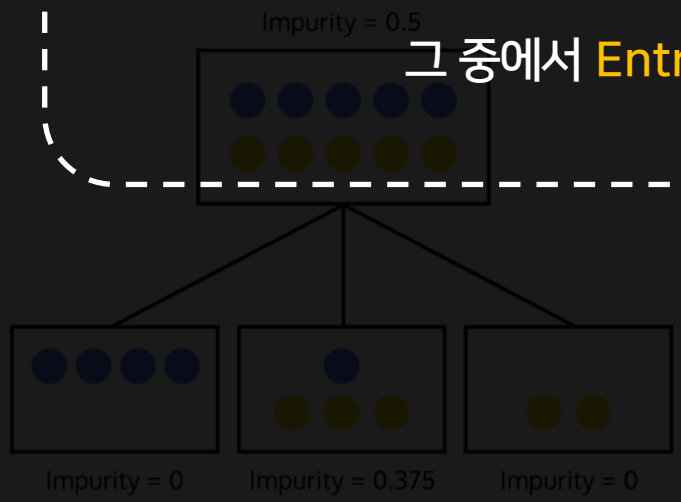
Decision Tree Classifier

분기 기준: 불순도

Decision Tree Classifier는 영역들의 불순도(Impurity)를 기준으로 분기

불순도를 측정하는 방법으로는 Misclassification Rate, Gini Index, Entropy 등 여러가지 방법이 존재함

그 중에서 Entropy에 대해서 알아볼 예정!



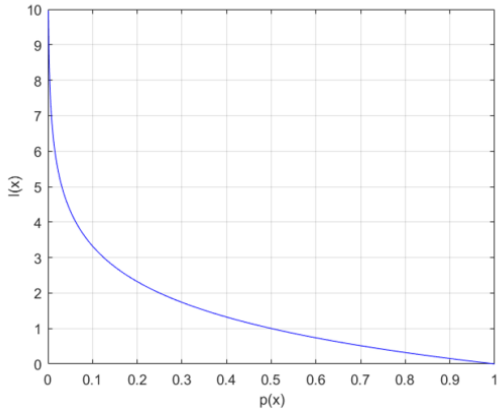
분기된 영역 내에서 서로 다른 범주들의 개체들이 얼마나 혼합되어 있는가,

불순도가 높을수록 분류가 잘 되지 않았다고 할 수 있다.

Decision Tree Classifier

Entropy

확률분포가 가지는 정보량을 수치로 표현한 것



특정 사건이 발생할 **확률이 작아질수록** 정보량이 높음

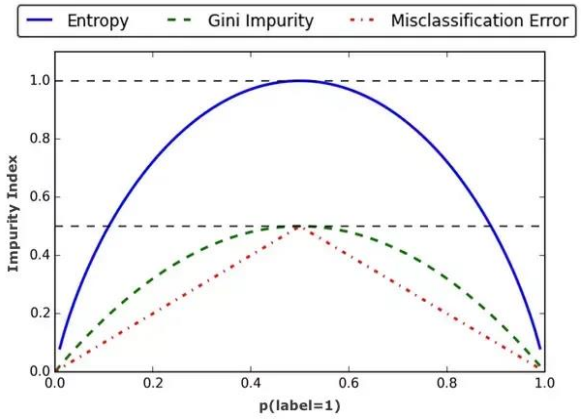
$$I(x) = -\log_b P(x)$$

$P(x)$: 사건 X가 발생할 확률

Decision Tree Classifier

Entropy

확률분포가 가지는 정보량을 수치로 표현한 것



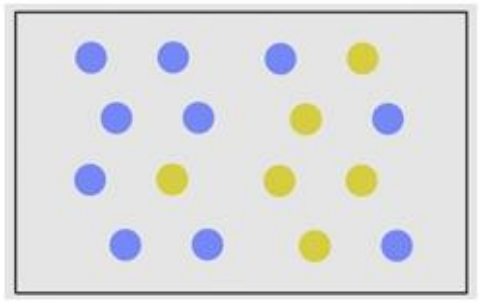
정보량을 **놀람의 정도**로 생각해보자!

발생 확률이 비슷한 경우 정보의 불확실성 증가

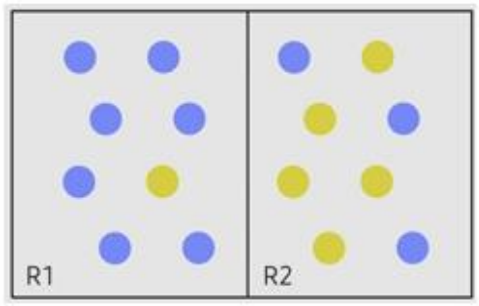
-> 엔트로피 값 증가

$$Entropy = E[I(X)] = -\sum_{i=1}^{n} P(x_i) logb(P(x_i))$$

Decision Tree Classifier



0.95

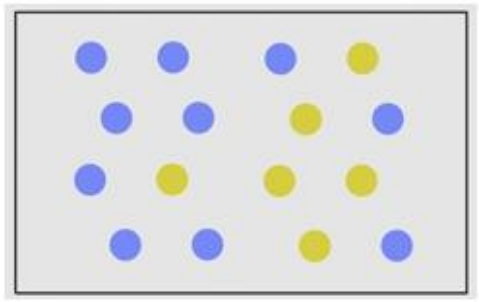


Decision Tree Classifier의 핵심은

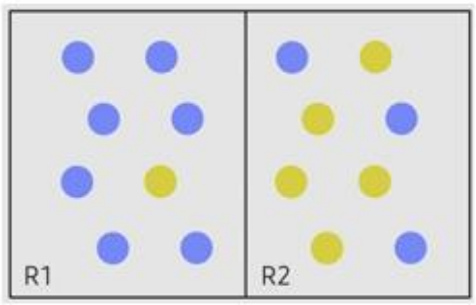
불순도가 낮아지는 방향으로 학습을 진행하는것

분기 이전 엔트로피 : $Entropy = -\frac{10}{16}\log_2\left(\frac{10}{16}\right) - \frac{6}{16}\log_2\left(\frac{6}{16}\right) \approx 0.95$

Decision Tree Classifier



0.95



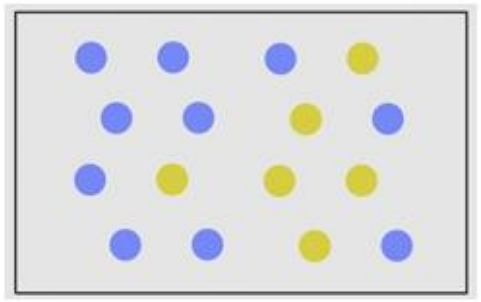
Decision Tree Classifier의 핵심은

불순도가 낮아지는 방향으로 학습을 진행하는것

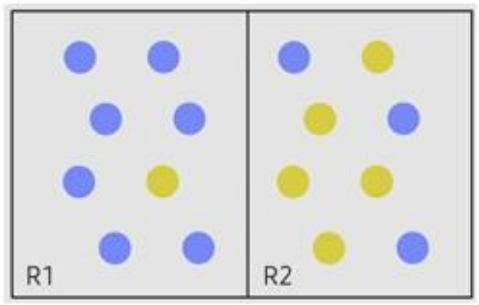
분기 이후 엔트로피 R_1: $Entropy = -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right)$

분기 이후 엔트로피 R_2 : $Entropy = -\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right)$

Decision Tree Classifier



0.95



0.75

분기 후 엔트로피 :

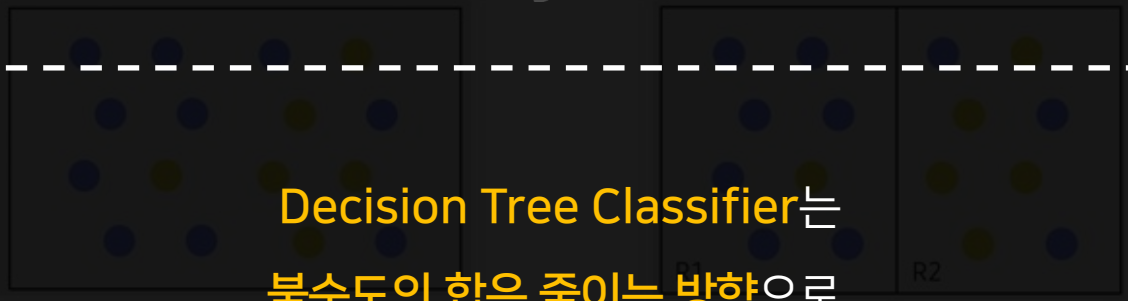
$$E(S,P) = \frac{8}{16} * Entropy(R_1) + \frac{8}{16} * Entropy(R_2) \approx 0.75$$

분기 이후의 불순도가 분기 이전에 비해 감소

(0.95 에서 0.75로 정보 이득)

1 트리 기반 모델

Decision Tree Classifier



Decision Tree Classifier는

불순도의 합은 줄이는 방향으로,

정보이득은 늘리는 방향으로 학습을 진행함!

이제 학습 프로세스를 살펴보자!

분기 후 엔트로피 :

$$E(S,P) = \frac{8}{16} * Entropy(R_1) + \frac{8}{16} * Entropy(R_2) \approx 0.75$$

분기 이후의 불순도가 분기 이전에 비해 감소(정보 이득)

1 트리 기반 모델

Decision Tree Classifier | 학습 프로세스

Income	Lot Size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
47.4	16.4	Non-owner
…	…	…
33.0	18.8	Non-owner
110.1	19.2	Owner
…	…	…
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

24개 가구에 대해서 잔디깎이 기계가 있는지 여부를 분류

독립변수로는 수입 (Income)과 마당 크기 (Lot Size) 사용함

Decision Tree Classifier | 학습 프로세스

Income	Lot Size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
47.4	16.4	Non-owner
…	…	…
33.0	18.8	Non-owner
110.1	19.2	Owner
…	…	…
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

1. 관측값들을 하나의 변수를 기준으로 오름차순 **정렬**

여기서는 Lot Size

Decision Tree Classifier | 학습 프로세스

Income	Lot Size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
47.4	16.4	Non-owner
…	…	…
33.0	18.8	Non-owner
110.1	19.2	Owner
…	…	…
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

$$\text{분기 전 엔트로피} = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

$$\text{분기 후 엔트로피} = \frac{1}{24}(-\log_2 1) +$$

$$\frac{23}{24}\left(-\frac{12}{23}\log_2\left(\frac{12}{23}\right) - \frac{11}{23}\log_2\left(\frac{11}{23}\right)\right) \approx 0.96$$

Information Gain= 1-0.96=0.04

2. Split 가능한 모든 경우의 불순도를 탐색하고 **Information Gain**을 계산

Ex(Split Point = 14.4)

Decision Tree Classifier | 학습 프로세스

Income	Lot Size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
47.4	16.4	Non-owner
…	…	…
33.0	18.8	Non-owner
110.1	19.2	Owner
…	…	…
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

불순도가 분기 이전에 비해 0.04만큼 감소

변수가 여러 개이기 때문에 **모든 경우를 탐색하여**

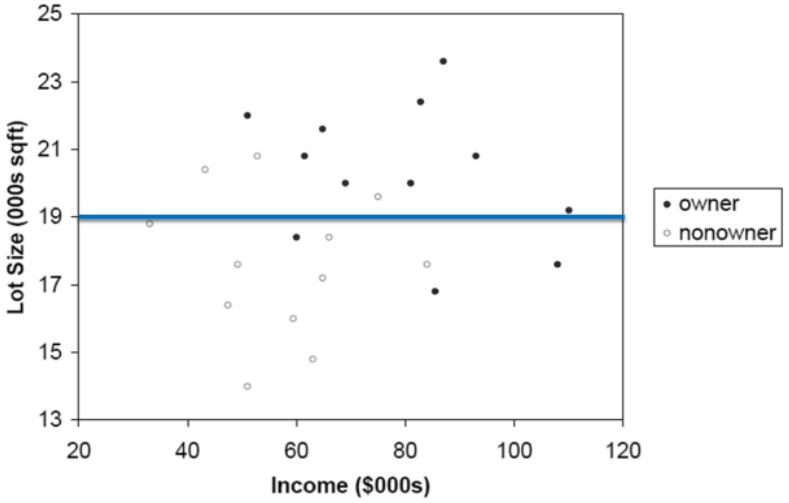
최선의 결과를 찾음

2. Split 가능한 모든 경우의 불순도를 탐색하고 **Information Gain**을 계산

Ex(Split Point = 14.4)

Decision Tree Classifier | 학습 프로세스

Income	Lot Size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
47.4	16.4	Non-owner
…	…	…
33.0	18.8	Non-owner
110.1	19.2	Owner
…	…	…
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

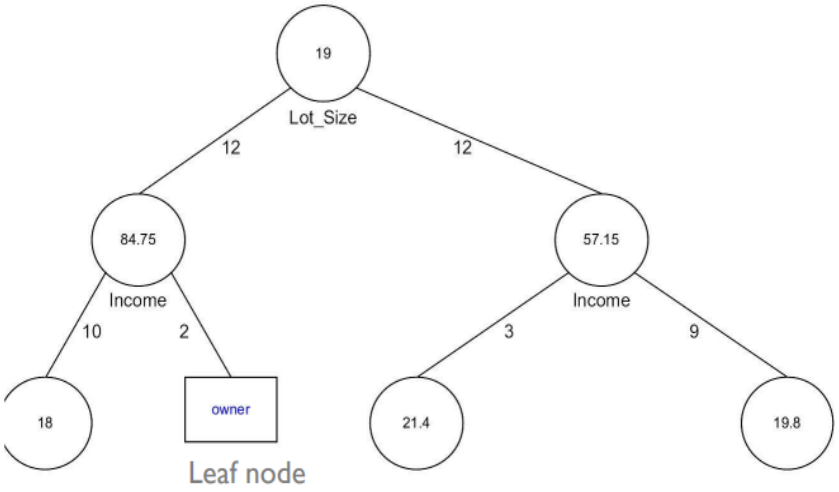


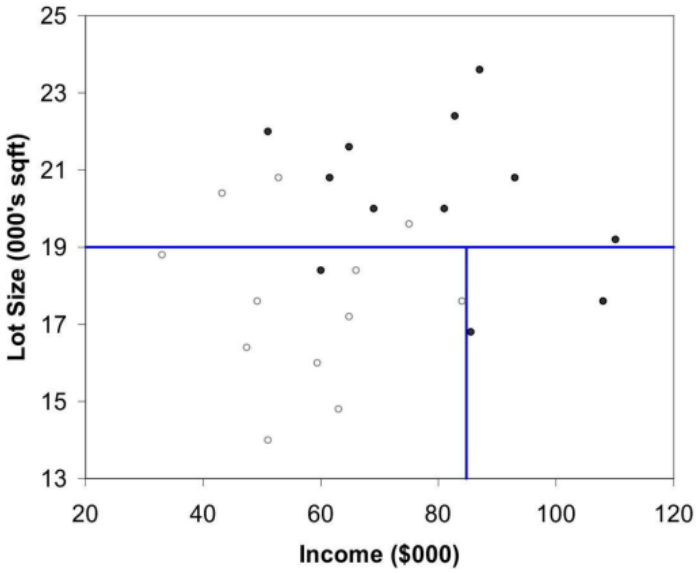
Lot size라는 독립변수

영역을 19를 기준으로 **분할**

3. Information Gain이 **가장 큰 경우에** 따라 분기

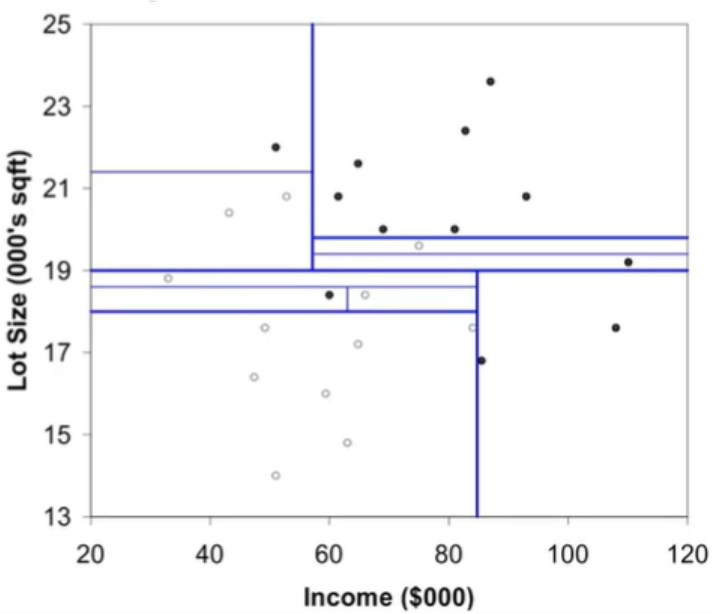
Decision Tree Classifier | 학습 프로세스





4. 분기되는 노드를 따라 내려가면서, 위 **과정을 반복**

Decision Tree Classifier | 학습 프로세스



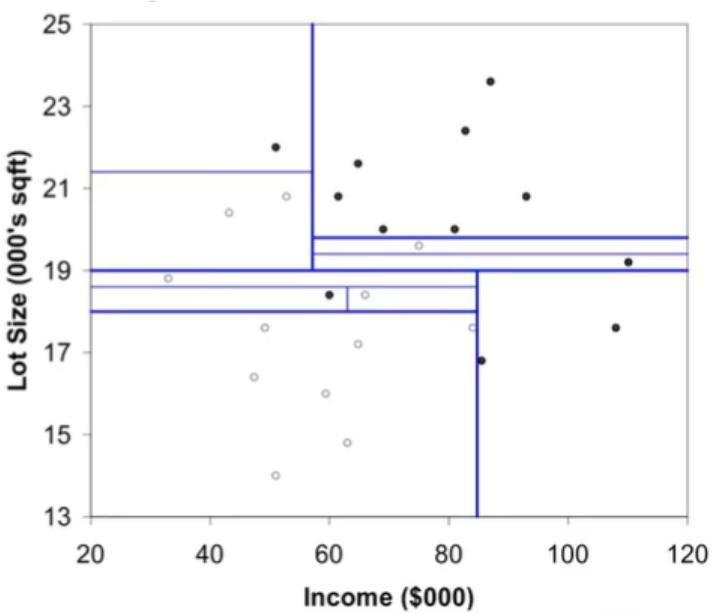
서로 다른 클래스의 관측값이
완전히 겹쳐있지 않은 이상, 하나씩 분기해나가면
학습 데이터를 100% 구분 가능

이를 Full-Tree라고 함

5. Information Gain=0이 되면 분기를 종료

1 트리 기반 모델

Decision Tree Classifier | 학습 프로세스



서로 다른 클래스의 관측값이

완전히 겹쳐있지 않은 이상, 하나씩 분기해나가면

학습 데이터를 100% 구분 가능

이를 Full-Tree라고 함

6. 이렇게 트리가 완성되고 나면, 새로운 객체는 만들어둔 트리를 따라 내려가서

Leaf Node의 비율대로 판단해 분류

Ex) 해당 영역에 A Class 10개, B Class 3개가 있었다면 A Class로 분류!

Decision Tree Regressor

Decision Tree Regressor

Decision Tree를 **회귀** 문제에 적용할 수 있도록 변형시킨 것으로,

분기하는 기준이 달라짐

⋮

RSS (Residual Sum of Squares)

실제값과 추정값의 차이

$$MSE(Mean\ Square\ Error) = \frac{RSS}{Degree\ of\ Freedom}$$

RSS를 최소화하는 방향으로 학습

Decision Tree Regressor

목적함수

$$\min_{c_m} \sum_{i=1}^{n} \{y_{\bar{l}} - f(x_i)\}^2 = \min_{c_m} \sum_{i=1}^{N} \left\{ y_i - \sum_{m=1}^{M} C_m I(x \in R_m) \right\}^2$$

⋮

c_m : m번째 노드의 예측값 (해당 노드 관측값의 평균)

R_m : m번째 노드

M : 전체 노드 개수

N : 전체 관측치 개수

Decision Tree Regressor

목적함수

$$\min_{c_m} \sum_{i=1}^{n} \{y_{\bar{i}} - f(x_i)\}^2 = \min_{c_m} \sum_{i=1}^{N} \left\{ y_i - \sum_{m=1}^{M} C_m I(x \in R_m) \right\}^2$$

⋮

C_m : m번째 노드의 예측값 (해당 노드 관측값의 평균)

M : 전체 노드 개수

N : 전체 관측치 개수

i번째 관측치의 실제값에 i번째 관측치의 예측값을 뺀

오차의 제곱합을 최소화시켜야 한다는 의미

1 트리 기반 모델

Decision Tree Regressor | 학습 프로세스

Price	Parking fee	Foreign	Review Rating
10.0	0.0	0	10
11.0	1.0	0	10
12.0	2.0	1	10
13.0	3.0	0	10
14.0	4.0	1	13
15.0	5.0	0	20
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
23.0	13.0	1	80
24.0	14.0	1	83

지불 금액(Price), 주차 비용(Parking fee), 외국인 여부(Foreign)
을 독립변수로 리뷰 평점을 예측하려고 함

Decision Tree Regressor | 학습 프로세스

Price	Parking fee	Foreign	Review Rating
10.0	0.0	0	10
11.0	1.0	0	10
12.0	2.0	1	10
13.0	3.0	0	10
14.0	4.0	1	13
15.0	5.0	0	20
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
23.0	13.0	1	80
24.0	14.0	1	83

1. 관측값들을 한 변수를 기준으로 오름차순 정렬

여기서는 Price

Decision Tree Regressor | 학습 프로세스

Price	Parking fee	Foreign	Review Rating
10.0	0.0	0	10
11.0	1.0	0	10
12.0	2.0	1	10
13.0	3.0	0	10
14.0	4.0	1	13
15.0	5.0	0	20
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
23.0	13.0	1	80
24.0	14.0	1	83

R1의 예측값 : $\frac{10+10+10+10}{4} = 10$

R2의 예측값 : $\frac{13+20+\cdots+83}{10} = 54.5$

RSS = $(10-10)^2+(10-10)^2+\cdots+(83-54.5)^2$

만약 'Price ≤ 13.5'를 기준으로 Split한 경우,

RSS 값은 위와 같이 계산됨

2. Split 가능한 모든 경우의 RSS를 탐색한다.

Ex(Split Point = 13.5)

1 트리 기반 모델

Decision Tree Regressor | 학습 프로세스

Price	Parking fee	Foreign	Review Rating
10.0	0.0	0	10
11.0	1.0	0	10
12.0	2.0	1	10
13.0	3.0	0	10
14.0	4.0	1	13
15.0	5.0	0	20
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
23.0	13.0	1	80
24.0	14.0	1	83

R1의 예측값 : $\frac{10+10+10+10}{4} = 10$

R2의 예측값 : $\frac{13+20+\cdots+83}{10} = 54.5$

RSS : $(10-10)^2+(10-10)^2+\cdots+$

분기 조건에 따라

RSS는 달라짐

만약 'Price ≤ 13.5'을

기준으로 Split한 경우,

RSS 값은 위와 같이 계산됨

2. Split 가능한 모든 경우의 RSS를 탐색한다.

Ex(Split Point = 13.5)

1 트리 기반 모델

Decision Tree Regressor | 학습 프로세스

Price	Parking fee	Foreign	Review Rating
10.0	0.0	0	10
11.0	1.0	0	10
12.0	2.0	1	10
13.0	3.0	0	10
14.0	4.0	1	13
15.0	5.0	0	20
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
23.0	13.0	1	80
24.0	14.0	1	83

일일이 계산하며 탐색해 본 결과,

Price = 20 일 때 **RSS가 가장 작게** 나옴

따라서 첫 번째로 **Price**라는

독립변수 영역을 20을 기준으로 분할

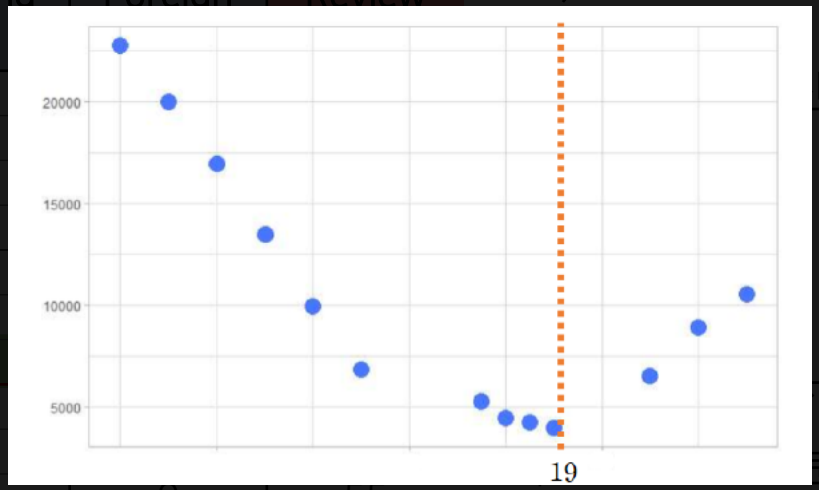
3. RSS가 최소인 Split point를 기준으로 분기

1 트리 기반 모델

분기 기준

Decision Tree Regressor | 학습 프로세스

Price	Parking fee	Foreign	Review
10.0	0.0		
11.0	1.0		
12.0	2.0		
13.0	3.0		
14.0	4.0		
15.0	5.0		
17.5	7.5		
18.0	8.0		
18.5	8.5		
19.0	9.0	0	55
21.0	11.0	1	90
22.0	12.0	0	83
23.0	13.0	1	80
24.0	14.0	1	83



하며 탐색해 본 결과,

때 RSS가 가장 작게 나옴

트리의 첫 번째

분기 기준을

Price <= 19로 채택

각 변수별로 최소 RSS가 나오는 지점을 찾고,

그 RSS 값들 중 가장 작은 경우를 Split point로 채택

3) RSS가 최소인 Split point를 기준으로 분기

Decision Tree Regressor | 학습 프로세스

4. 분기되는 노드를 따라 내려가면서, 위 과정을 반복

5. RSS 값이 더이상 작아지지 않으면 분기를 종료

6. 이렇게 트리가 완성되고 나면, 새로운 객체는 만들어둔 트리를 따라 내려가서
Leaf node의 예측값 (평균값)대로 판단해 예측

➡ ex. 해당 영역에 10, 11, 12, 9, 8 이런 값들이 있었다면, 예측값은 $\frac{10+11+12+9+8}{5}$ = 10이 됨!

1 트리 기반 모델

Decision Tree Regressor | 학습 프로세스

어디까지 분기할 것인가

4) 분기되는 노드를 따라 루트에서 자식 방향으로 분기 과정을 반복

기본적으로 Decision Tree는

불순도 또는 RSS가 더 이상 줄어들지 않을 때까지 분기

5) RSS 값이 더이상 작아지지 않으면 분기를 종료

하지만 이러한 Full Tree는 노이즈까지 학습

6) 이렇게 트리가 완성되고 나면, 새로운 객체는 만들어둔 트리를 따라 내려가서

Leaf node의 예측값 (평균값) 대로 판단해 예측

ex. 해당 영역에 10, 11, 12 이런 값들이 있었다면 예측값은 $\frac{10+11+12}{3}$ 이렇게!

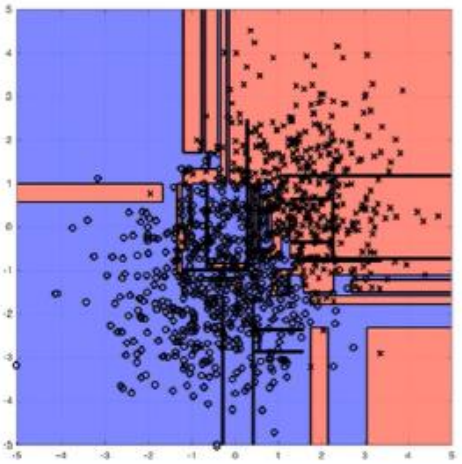
과대적합 문제 야기

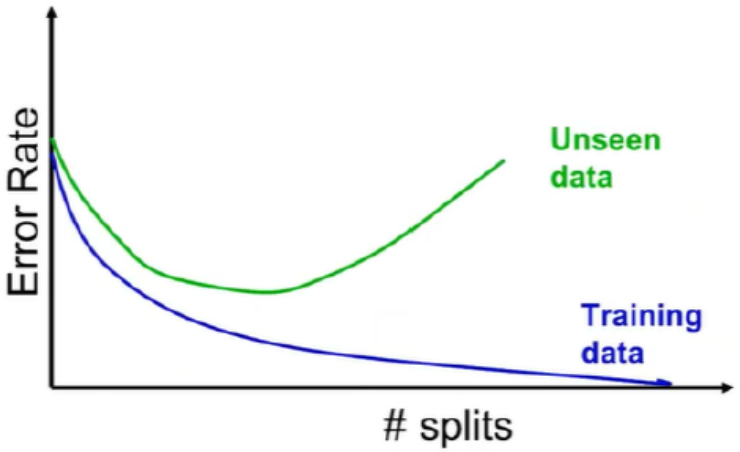
트리 기반 모델의 과적합

과대적합 (Overfitting)

새로운 데이터에 강건하지 못함 (non-robust)

Bias ↓ Variance ↑



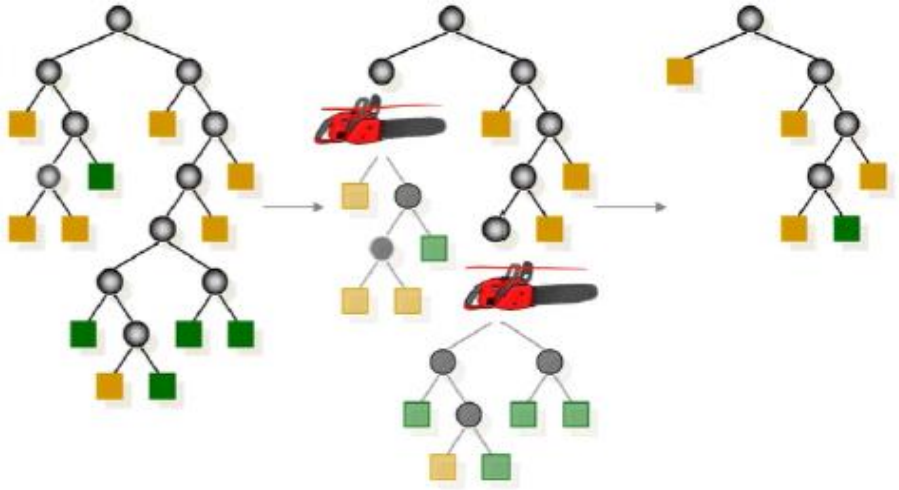


트리 기반 모델의 과적합

가지치기 (Pruning)

Decision의 **분기 깊이**를 적정 수준에서 **제한**

Full Tree까지 다 만들지 않음



트리 기반 모델의 과적합

가지치기 (Pruning)

Decision의 **분기 깊이**를 적정 수준에서 **제한**

Full Tree까지 다 만들지 않음

적당히 학습량을 조절하여

일반화 성능을 확보하자는 것이 **핵심 !**

1 트리 기반 모델

트리 기반 모델의 과적합

가지치기 (Pruning)

Decision의 **분기 깊이**를 적정 수준에서 **제한**

Full Tree까지 다 만들지 않음

사전 가지치기

트리의 깊이나 리프 노드 내

최소 관측값 수를 미리 정해

(**하이퍼파라미터 제한**)

Tree가 일정 수준 이상 자라지

못하도록 하는 방법

현재 많은 소프트웨어에서 구현되고 있음

사후 가지치기

Full Tree를 만든 후,

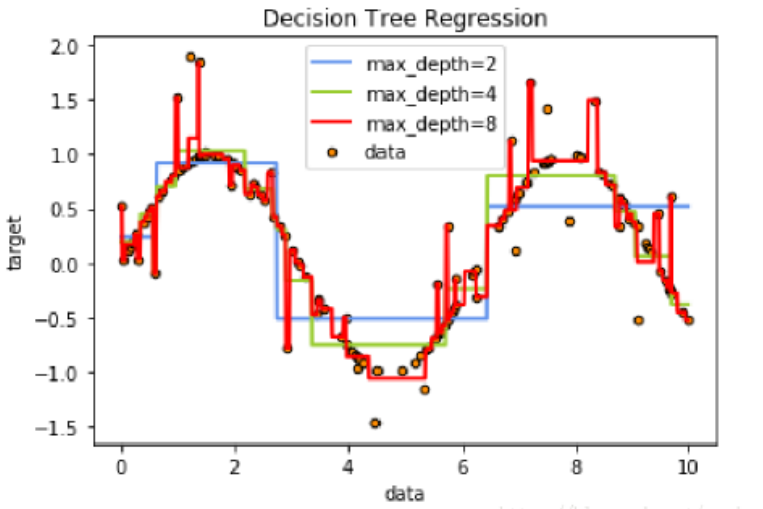
적절한 수준에서

종단 노드를 결합하는 방법

사전 가지치기 (pre-pruning)

Max_depth

트리가 분기하는 **최대 깊이를 제한**하는 파라미터



max_depth=2일 때 가장 단순,

max_depth=8일 때 가장 복잡

모델의 복잡도가 높아지면

설명력은 높아지지만,

새로운 데이터를 예측하는 성능이 저하됨

1 트리 기반 모델

사전 가지치기 (pre-pruning)

Max_depth

트리가 분기하는 **최대 깊이를 제한**하는 파라미터

Max_depth를 포함한 여러 **파라미터** 값을
적절하게 **조정**하고 **교차검증**을 통해 비교하여
가장 좋은 성능을 도출하는 모델을 만드는 것이 중요

1 트리 기반 모델

사전 가지치기 (pre-pruning)

Min_samples_split

노드를 분할하기 위한 최소 샘플 데이터 개수

"한 노드에 최소한 이 개수보다는 많이 들어있어야 Split 하겠다"

작게 설정할수록 과적합 가능성 증가 (default = 2)

Min_impurity_decrease

노드를 분할하기 위한 최소 불순도 감소량

"한 노드에 최소한 이 정도는 줄어들어야 Split 하겠다"

작게 설정할수록 과적합 가능성 증가

1 트리 기반 모델

사전 가지치기 (pre-pruning)

Min_samples_leaf

리프 노드가 되기 위한 최소 샘플 데이터 개수

"리프 노드에 최소한 이 개수의 샘플은 들어있어야 한다"

작게 설정할수록 과적합 가능성 증가 (default = 1)

불균형 데이터의 경우에는 작게 설정해야 할 수도 있음

Max_leaf_nodes

리프 노드의 최대 개수

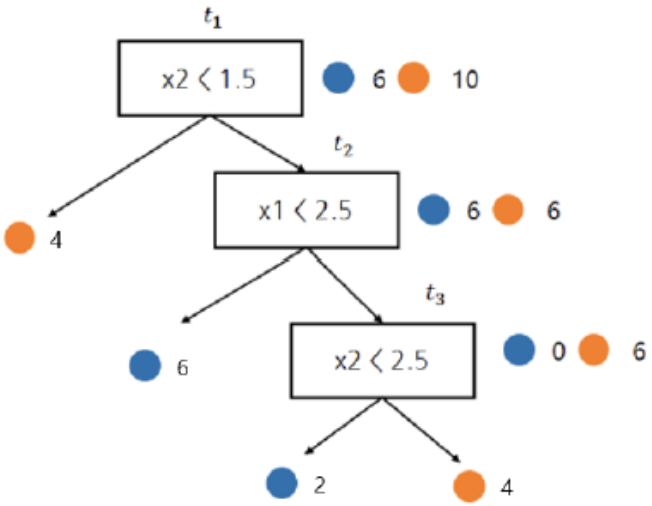
"리프 노드가 아무리 많아도 이 정도까지만 있어야 한다"

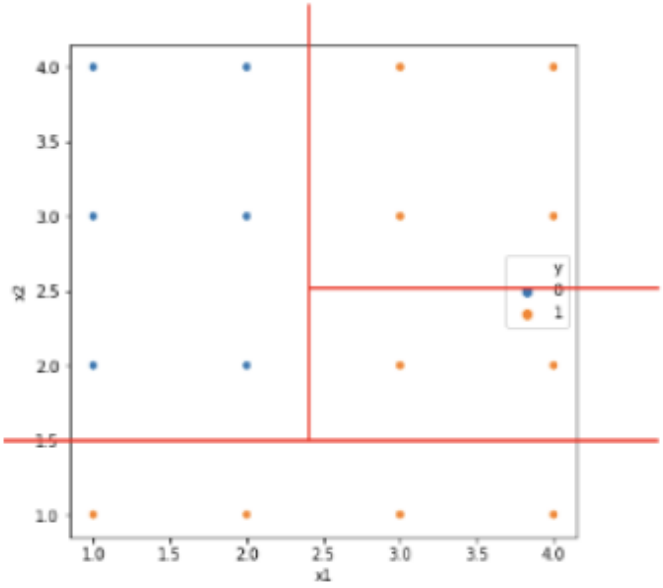
크게 설정할수록 과적합 가능성 증가

사후 가지치기 (post-pruning)

REP (Reduce Error Pruning)

Full Tree의 밑에서부터 분할 이전과 이후의 **불순도를 비교**하고,
분할 이전의 **불순도가 더 낮다**면 해당 **자식 노드를 쳐내는** 방식

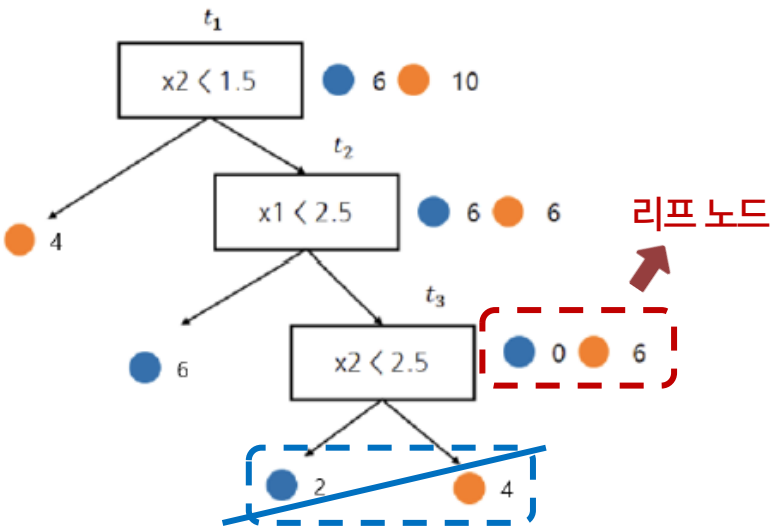




사후 가지치기 (post-pruning)

REP (Reduce Error Pruning)

Full Tree의 밑에서부터 분할 이전과 이후의 **불순도를 비교**하고,
분할 이전의 **불순도가 더 낮다**면 해당 **자식 노드를 쳐내는** 방식



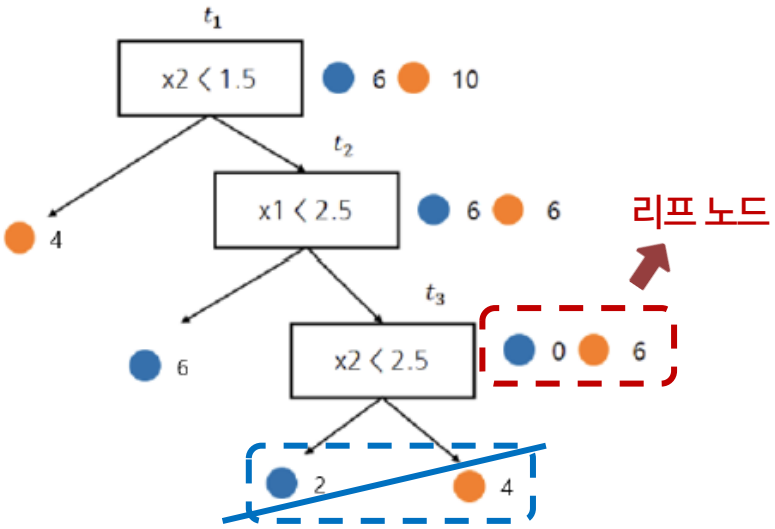
t_3에서 분할 이후의 **불순도**가
분할 이전의 부모 노드보다 커짐

t_3의 **자식노드**를 쳐내고,
주황색 공이 6개 들어있는 노드를
리프 노드로 바꿔줌

사후 가지치기 (post-pruning)

REP (Reduce Error Pruning)

Full Tree의 밑에서부터 분할 이전과 이후의 **불순도를 비교**하고,
분할 이전의 **불순도가 더 낮다**면 해당 **자식 노드를 쳐내는** 방식



"불순도가 감소하지 않은 분할"을
다시 결합해주는 방식

사후 가지치기 (post-pruning)

REP (Reduce Error Pruning)

Full Tree의 밑에서부터 분할 이전과 이후의 **불순도를 비교**하고,
분할 이전의 **불순도가 더 낮다**면 해당 **자식 노드를 쳐내는** 방식

원리가 단순해 구현이 쉽지만, 단순히 불순도만으로 pruning 여부를 판단하기 때문에
일반화 성능 향상에는 제한적
검증 데이터가 적은 경우 가지치기가 과하게 될 수 있음

⋮

Cost Complexity (비용 복잡도) 활용

사후 가지치기 (post-pruning)

CCP (Cost Complexity Pruning)

트리의 복잡도를 함께 고려하는

Cost Complexity (비용 복잡도)를 활용하여 더욱 안정적인 Tree를 얻을 수 있음

$$CC(T) = Err(T) + \alpha * L(T)$$

⋮

$CC(T)$: Tree의 비용 복잡도

$Err(T)$: 오분류율 (불순도)

$L(T)$: terminal node의 개수 (=구조 복잡도)

α : $Err(T)$와 $L(T)$를 결합하는 가중치

(=Complexity Parameter)

사후 가지치기 (post-pruning)

CCP (Cost Complexity Pruning)

트리의 복잡도를 함께 고려하는

Cost Complexity (비용 복잡도)를 활용하여 더욱 안정적인 Tree를 얻을 수 있음

$$CC(T) = Err(T) + \alpha * L(T)$$

여기서 α는 트리의 복잡도에 따라 적절한 수준의 **페널티**를 부여하는 것

최적의 α는 .cost_complexity_pruning_path를 통해 구할 수 있음

여러 후보들 가운데 **교차검증**을 실시하여 valid score가 가장 좋은 것 선택

2

앙상블 기법

앙상블 기법

앙상블 (Ensemble)

여러 개의 **모델**들을 합쳐서 **하나의 강한 모델**을 만들어내는 기법

Error를 줄이는 것이 **목표**

2 앙상블 기법

앙상블 기법

앙상블 (Ensemble)

여러 개의 모델들을 합쳐서 **하나의 강한 모델**을 만들어내는 기법

배깅 (Bagging)

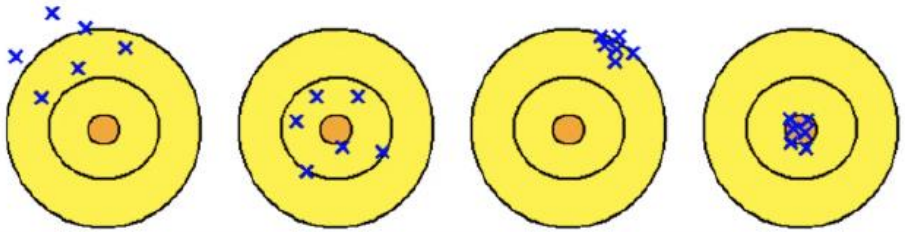
분산이 큰 모델들을 모아

분산을 줄이는 방식

부스팅 (Boosting)

편향이 큰 모델들을 모아

편향을 줄이는 방식



Bias	High	Low	High	Low
Variance	High	High	Low	Low

앙상블 주요 Idea

앙상블 기법

개별 Tree들이
어떻게 충분한
다양성을 얻을 것인가?

개별 Tree들이
어떻게 잘 결합할 것인가?

배깅

부스팅

By data

By model

편향을 줄이는 방식

Majority Voting,
Weighted Voting,
Targetting Residual
(boosting)

앙상블 효과

ϵ의 평균은 0이고, 서로 독립적

$$y_m(x) = f(x) + \epsilon_m(x)$$

$$E_x[\{y_m(x) - f(x)\}^2] = E_x[\epsilon_m(x)^2]$$

개별 모델 M개의 평균적인 Error

$$E_{Avg} = \frac{1}{M}\sum_{m=1}^{M} E_x[\epsilon_m(x)^2]$$

앙상블한 경우 평균적인 Error

$$E_{Ensemble} = E_x\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(x) - f(x)\right\}^2\right]$$

2 앙상블 기법

앙상블 효과

$$y_m(x) = f(x) + \epsilon_m(x)$$

$$E_x[\{y_m(x) - f(x)\}^2] = E_x[\epsilon_m(x)^2]$$

$$f(x) = \frac{1}{M} \times M \times f(x)$$

$$E_x\left[\left\{\frac{1}{M}\sum_{m=1}^{n}\epsilon_m(x)\right\}^2\right] = \frac{1}{M^2}\sum_{m=1}^{M}E_x[\epsilon_m(x)^2]$$

개별 모델 M개의 평균적인 Error

$$E_{Avg} = \frac{1}{M}\sum_{m=1}^{M}E_x[\epsilon_m(x)^2]$$

앙상블한 경우 평균적인 Error

$$E_{Ensemble} = E_x[\quad]$$

$$E_{Ensemble} = \frac{1}{M} \times E_{Avg}$$

앙상블의 에러가 개별 에러들의 $\frac{1}{M}$까지 감소

앙상블 효과

$$y_m(x) = f(x) + \epsilon_m(x)$$

$$E_x[\{y_m(x) - f(x)\}^2] = E_x[\epsilon_m(x)^2]$$

현실에서는 개별 ϵ끼리 완전히 독립이기 힘들기 때문에

초기 식의 가정을 충족하기 어려움

$$(a^2 + b^2)(x^2 + y^2) \geq (ax + by)^2$$

코시 - 슈바르츠 부등식 활용

$$\left[\sum_{m=1}^{M} \epsilon_m(x)\right]^2 \leq M \sum_{m=1}^{M} \epsilon_m(x)^2 \quad \Rightarrow \quad \left[\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(x)\right]^2 \leq \frac{1}{M}\sum_{m=1}^{M} \epsilon_m(x)^2$$

$$E_{Ensemble} \leq E_{Avg}$$

2 앙상블 기법

앙상블 효과

결론

$$y_m(x) = f(x) + \epsilon_m(x)$$

$$E_x[\{y_m(x) - f(x)\}^2] = E_x[\epsilon_m(x)^2]$$

현실에서는 개별 ϵ끼리 완전 독립이기 힘들기 때문에

처음 식의 가정을 충족하기 어려움

- ϵ이 이상적인 조건을 만족할 경우, 앙상블 Error가 개별 모델 Error의 $\frac{1}{M}$ 까지 줄어들 수 있음

$(a^2 + b^2)(x^2 + y^2) \geq (ax + by)^2$ 활용

- ϵ이 이상적인 조건을 만족하지 못하더라도, 앙상블 Error가 개별 모델 Error보다는 낮음

$$\left[\sum_{m=1}^{M} \epsilon_m(x)\right]^2 \leq M \sum_{m=1}^{M} \epsilon_m(x)^2 \quad \Rightarrow \quad \left[\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(x)\right]^2 \leq \frac{1}{M}\sum_{m=1}^{M} \epsilon_m(x)^2$$

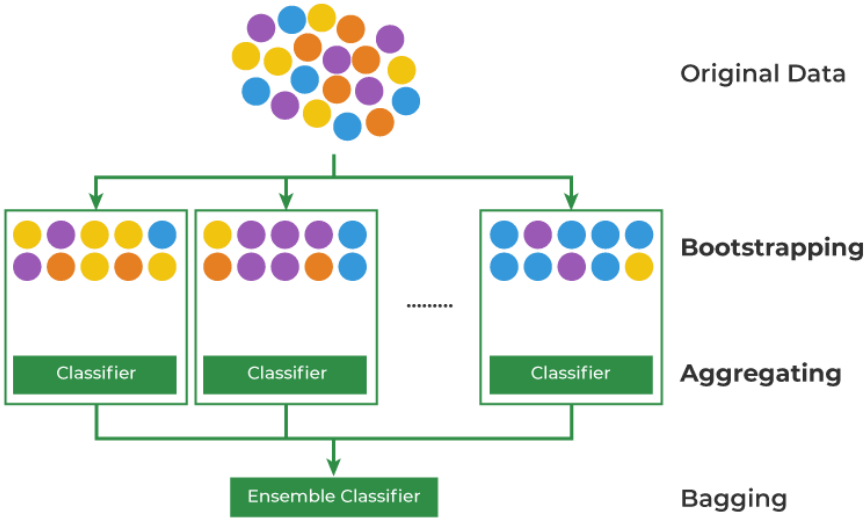
2 앙상블 기법

배깅

배깅 (Bagging)

전체 데이터에서 샘플을 **복원추출**한 데이터셋을 만들고

여러 개의 Decision Tree를 학습시키는 방법



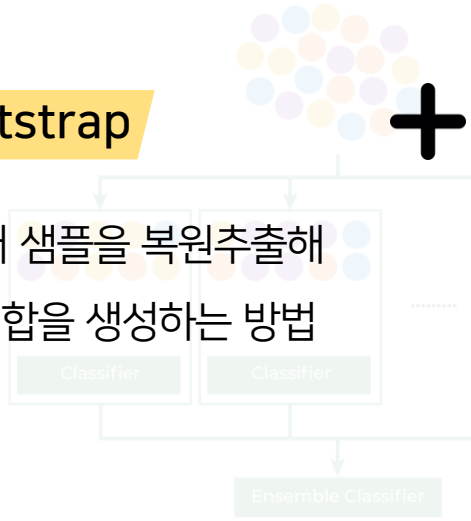
2 앙상블 기법

배깅

배깅 (Bagging)

전체 데이터에서 샘플을 **복원추출**한 데이터셋을 만들고
여러 개의 Decision Tree를 학습시키는 방법

Bootstrap + Aggregating의 합성어!



Bootstrap

데이터셋으로부터 샘플을 복원추출해
여러 개의 부분집합을 생성하는 방법

Aggregating

Bootstrap Data를 학습한
모델들의 결과를 집계하는 방법

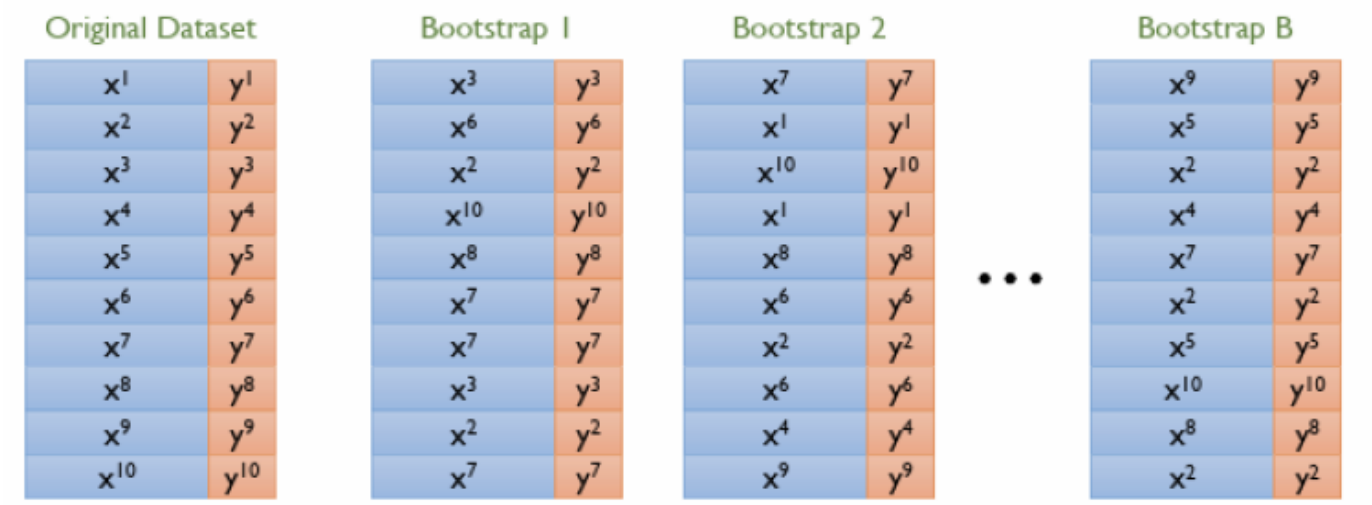
2 앙상블 기법

배깅 | Bootstrap

부트스트랩 (Boostrap)

데이터셋으로부터 샘플을 랜덤으로 **복원추출**하여

여러 개의 데이터셋 (부분집합) 을 만들어내는 방법

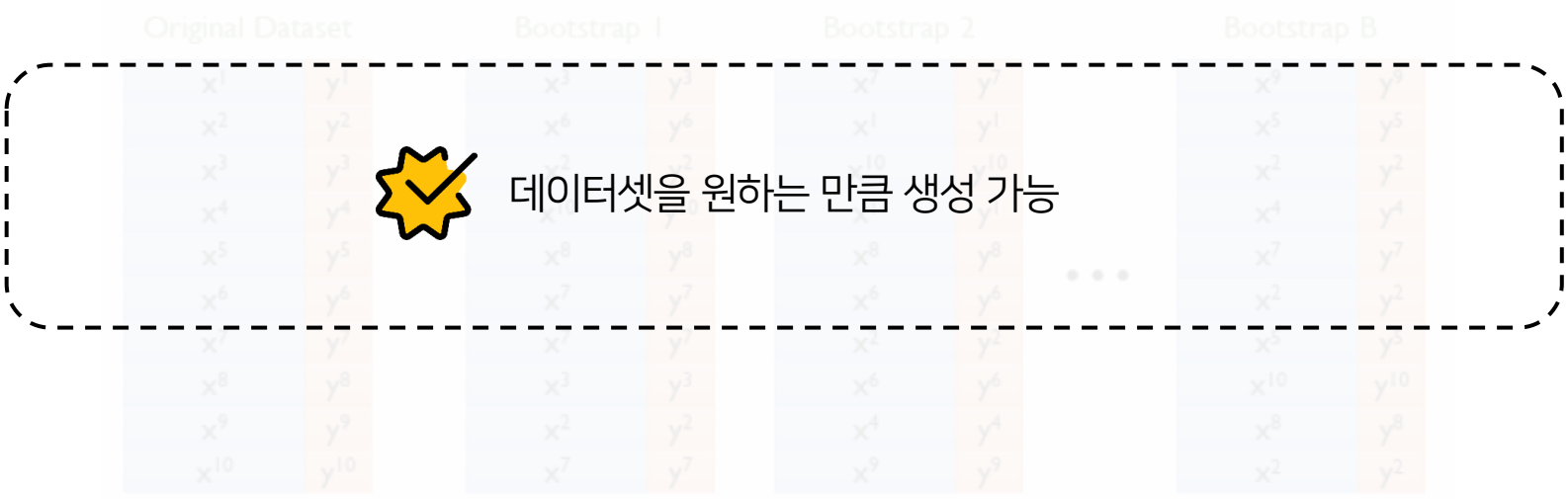


2 앙상블 기법

배깅 | Bootstrap

부트스트랩 (Boostrap)

데이터셋으로부터 샘플을 랜덤으로 **복원추출**하여
여러 개의 데이터셋 (부분집합) 을 만들어내는 방법



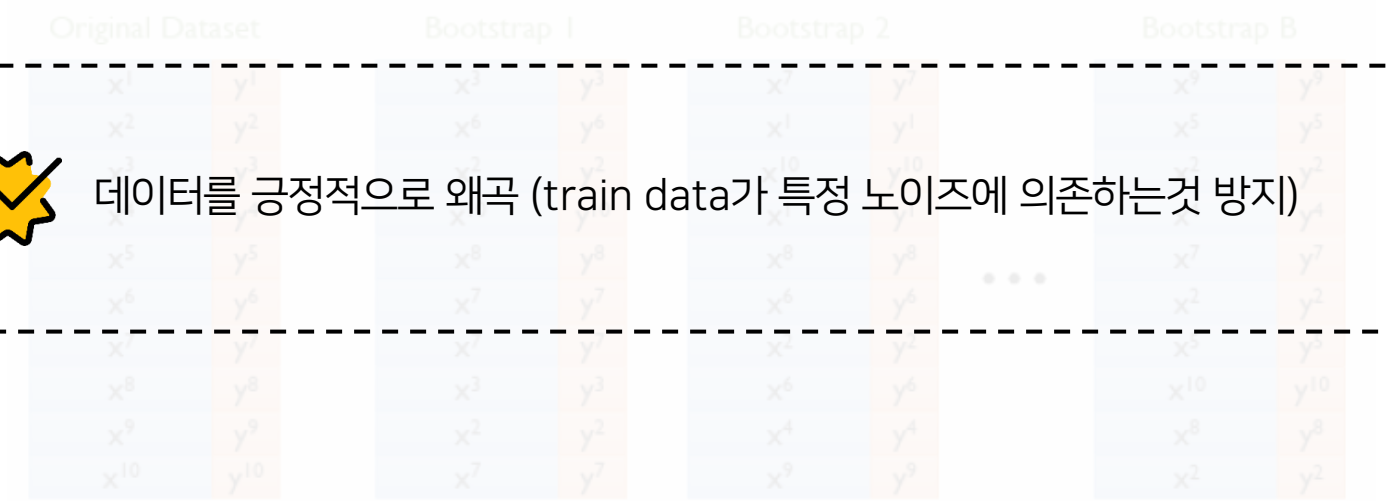
데이터셋을 원하는 만큼 생성 가능

2 앙상블 기법

배깅 | Bootstrap

부트스트랩 (Boostrap)

데이터셋으로부터 샘플을 랜덤으로 **복원추출**하여
여러 개의 데이터셋 (부분집합) 을 만들어내는 방법



데이터를 긍정적으로 왜곡 (train data가 특정 노이즈에 의존하는것 방지)

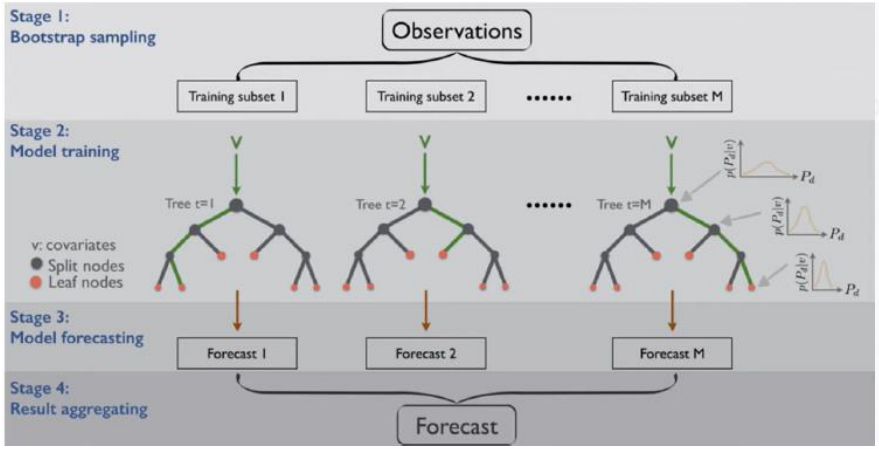
2 앙상블 기법

배깅 | Bootstrap

부트스트랩 (Boostrap)

데이터셋으로부터 샘플을 랜덤으로 **복원추출**하여

여러 개의 데이터셋 (부분집합) 을 만들어내는 방법



여러 데이터셋을 학습하므로 복잡한

모델의 분산을 줄일 수 있음

Low bias & high variance

모델 복잡도가 높은 모델과 잘 맞음

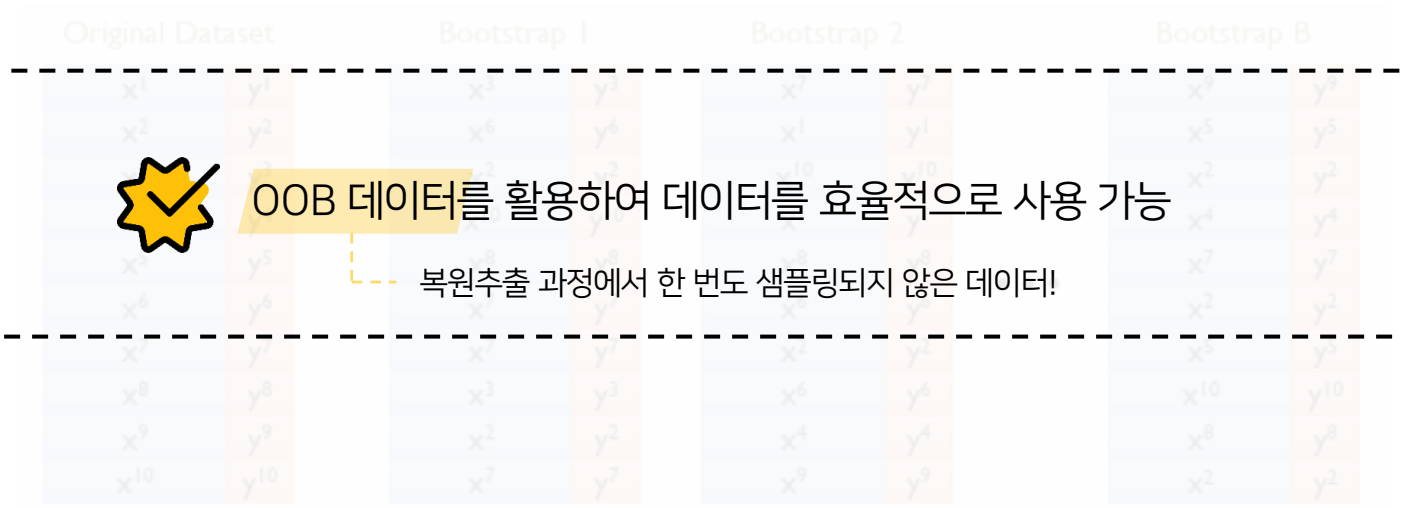
2 앙상블 기법

배깅 | Bootstrap

부트스트랩 (Boostrap)

데이터셋으로부터 샘플을 랜덤으로 **복원추출**하여

여러 개의 데이터셋 (부분집합) 을 만들어내는 방법



OOB 데이터를 활용하여 데이터를 효율적으로 사용 가능

복원추출 과정에서 한 번도 샘플링되지 않은 데이터!

2 앙상블 기법

배깅 | Aggregating

다수결 보팅 (Major Voting)

예측 결과들을 **다수결로 집계**해서 최종 결과를 예측하는 방식

아래의 경우 예측 값을 1로 결정

Training Accuracy	Ensemble population	P(y=1) for a test instance	Predicted class label
0.80	Model 1	0.90	1
0.75	Model 2	0.92	1
0.88	Model 3	0.87	1
0.91	Model 4	0.34	0
0.77	Model 5	0.41	0
0.65	Model 6	0.84	1
0.95	Model 7	0.14	0
0.82	Model 8	0.32	0
0.78	Model 9	0.98	1
0.83	Model 10	0.57	1

$$\sum_{j=1}^{n} I(\hat{y}_j = 0) = 4$$

$$<$$

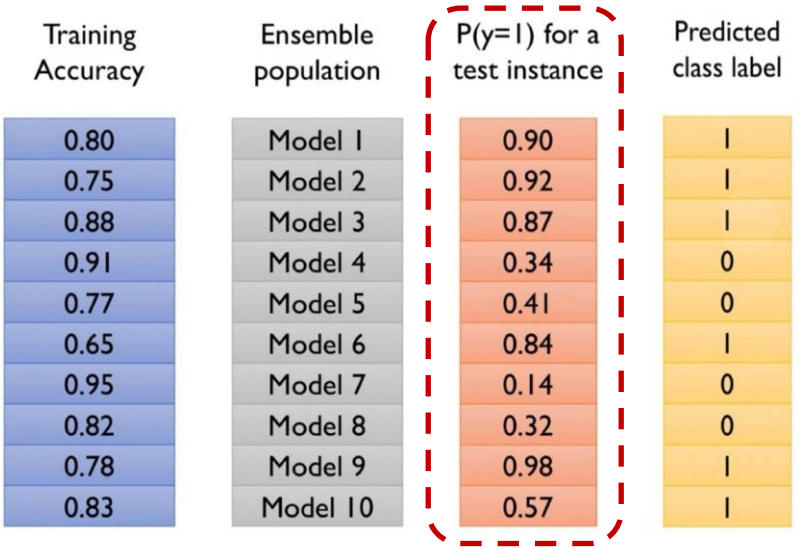
$$\sum_{j=1}^{n} I(\hat{y}_j = 1) = 6$$

배깅 | Aggregating

가중치 보팅 (Weighted Voting)

예측 확률 값 또는 성능의 평균을 계산 후 가중치로 반영하여 최종 결과를 예측하는 방식

아래의 경우는 **예측 확률 값을 이용**한 경우로, 예측 값을 1로 결정



Training Accuracy	Ensemble population	P(y=1) for a test instance	Predicted class label
0.80	Model 1	0.90	1
0.75	Model 2	0.92	1
0.88	Model 3	0.87	1
0.91	Model 4	0.34	0
0.77	Model 5	0.41	0
0.65	Model 6	0.84	1
0.95	Model 7	0.14	0
0.82	Model 8	0.32	0
0.78	Model 9	0.98	1
0.83	Model 10	0.57	1

$$\frac{1}{n}\sum_{j=1}^{n} P(y=0) = 0.375$$

$$<$$

$$\frac{1}{n}\sum_{j=1}^{n} P(y=1) = 0.375$$

배깅 | Aggregating

가중치 보팅 (Weighted Voting)

예측 확률 값 또는 성능의 평균을 계산 후 가중치로 반영하여 최종 결과를 예측하는 방식

아래의 경우는 **예측 성능을 이용**한 경우로, 예측 값을 1로 결정

Training Accuracy	Ensemble population	P(y=1) for a test instance	Predicted class label
0.80	Model 1	0.90	1
0.75	Model 2	0.92	1
0.88	Model 3	0.87	1
0.91	Model 4	0.34	0
0.77	Model 5	0.41	0
0.65	Model 6	0.84	1
0.95	Model 7	0.14	0
0.82	Model 8	0.32	0
0.78	Model 9	0.98	1
0.83	Model 10	0.57	1

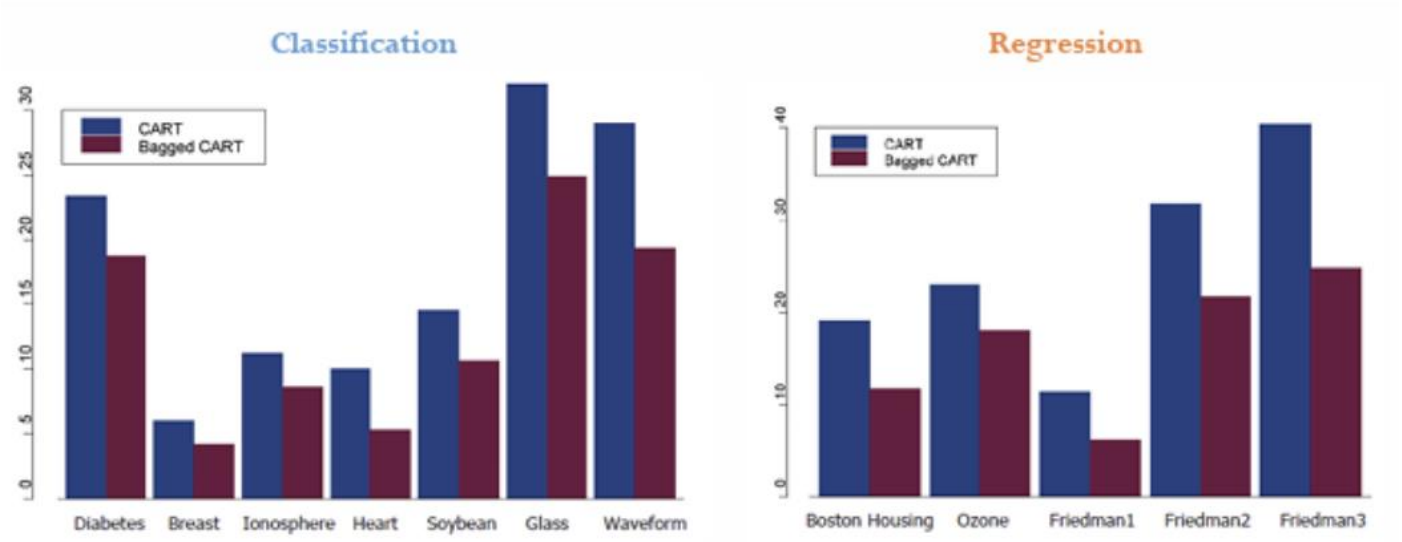
$$\frac{\sum_{j=1}^{n}(TA_j)\cdot I(\hat{y}_j=0)}{\sum_{j=1}^{n}(TA_j)}=0.424$$

$$<$$

$$\frac{\sum_{j=1}^{n}(TA_j)\cdot I(\hat{y}_j=1)}{\sum_{j=1}^{n}(TA_j)}=0.576$$

2 앙상블 기법

배깅 | Aggregating



실제로 bagging 기법을 사용하여 훈련 시,

단일 CART 모델보다 더 좋은 성능을 보임

Y축 값은 Valid Error 를 의미

2 앙상블 기법

배깅 | Aggregating

복원 추출로 데이터들을 생성해내는 데 그치는 일반적인 배깅 방식은
데이터들 간에 **상관성**을 가지게 해 모델의 분산을 제대로 못줄일 수 있음

⋮

Random Forest 모델을 이용해 이러한 문제점을 어느정도 해결 가능 !

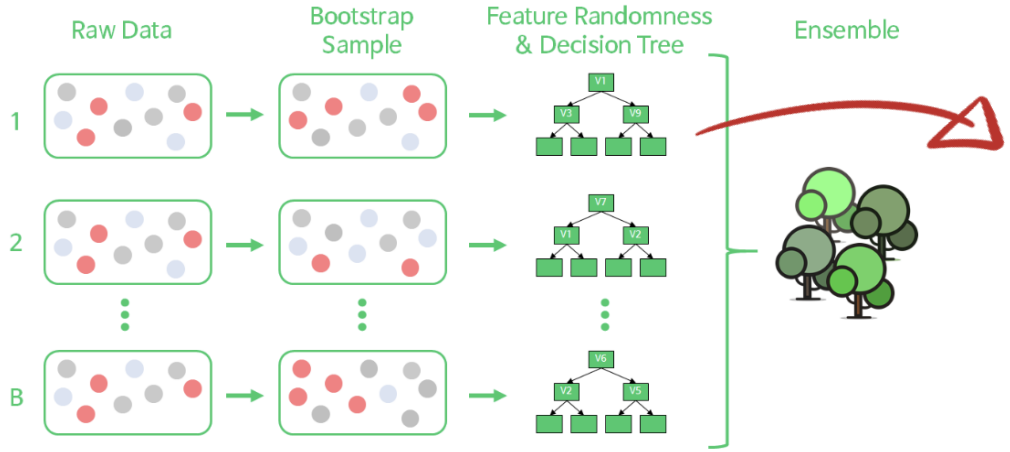
2 앙상블 기법

배깅 | Random Forest

랜덤 포레스트 (Random Forest)

분기 시 사용되는 **변수를 랜덤하게 선택**하게 끔 하여,

일반적인 배깅 방식에 비해 분산을 더 줄일 수 있는 모델



분기 시 변수 선택 개수

분류 문제: 전체 변수 개수의 **제곱근** 만큼

예측 문제: 전체 변수 개수의 **1/3** 만큼 선택

일반적으로 이미 나와 있음

2 앙상블 기법

배깅 | Random Forest

랜덤 포레스트 (Random Forest)

분기 시 사용되는 **변수를 랜덤하게 선택**하게 끔 하여,
일반적인 배깅 방식에 비해 분산을 더 줄일 수 있는 모델

Random Forest는 **예측력이 좋고 분산을 줄여줄 수 있지만,**
모형의 해석이 어렵다는 단점이 존재

⋮

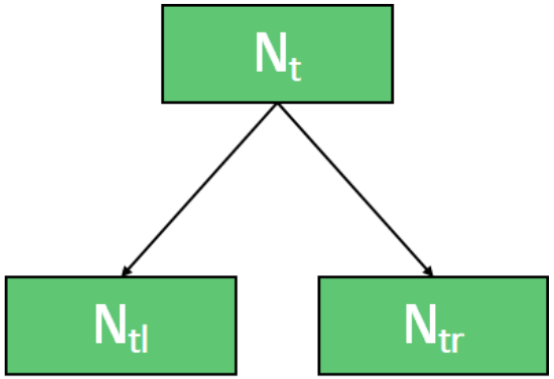
변수 중요도를 계산하여 이용함으로써 이러한 단점을 보완

배깅 | Random Forest

변수 중요도 (Feature Importance)

선택한 변수로 분류를 진행한 뒤 **중요도가 높은 변수**부터 순서대로 정렬한 것

중요도를 계산하는 기준에는 여러가지가 존재하며, 아래는 **불순도 감소분**을 기준으로 하였음



$$\Delta i(t) = i(t) - \frac{N_{tl}}{N_t} i(t_l) - \frac{N_{tr}}{N_t} i(t_r)$$

- $i(t)$: t노드의 impurity (entropy, gini index, variance, ...)
- N_t : t노드의 관측치 개수

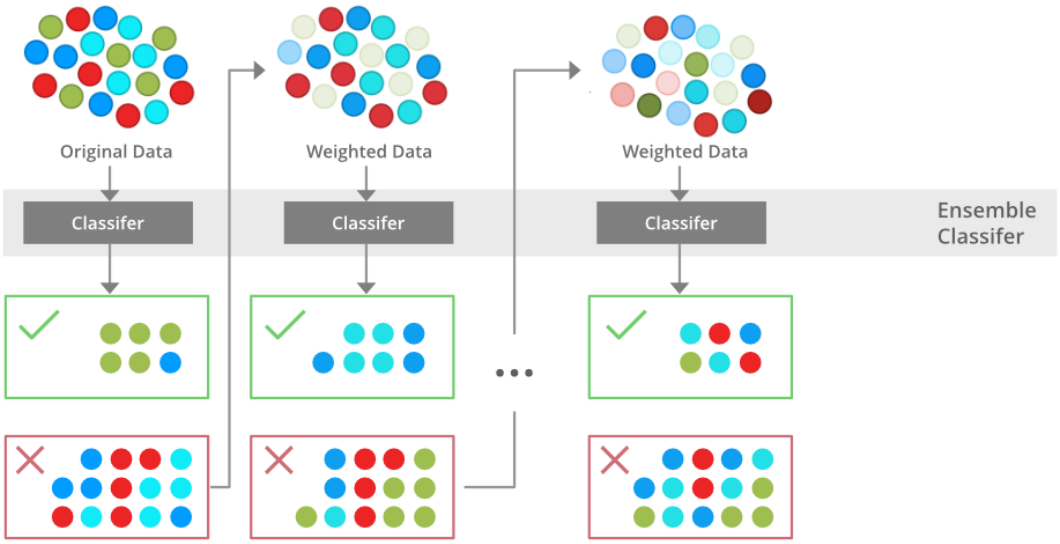
불순도 감소분(= $\Delta i(t)$) 이 클수록

중요도가 큰 변수로 간주

부스팅

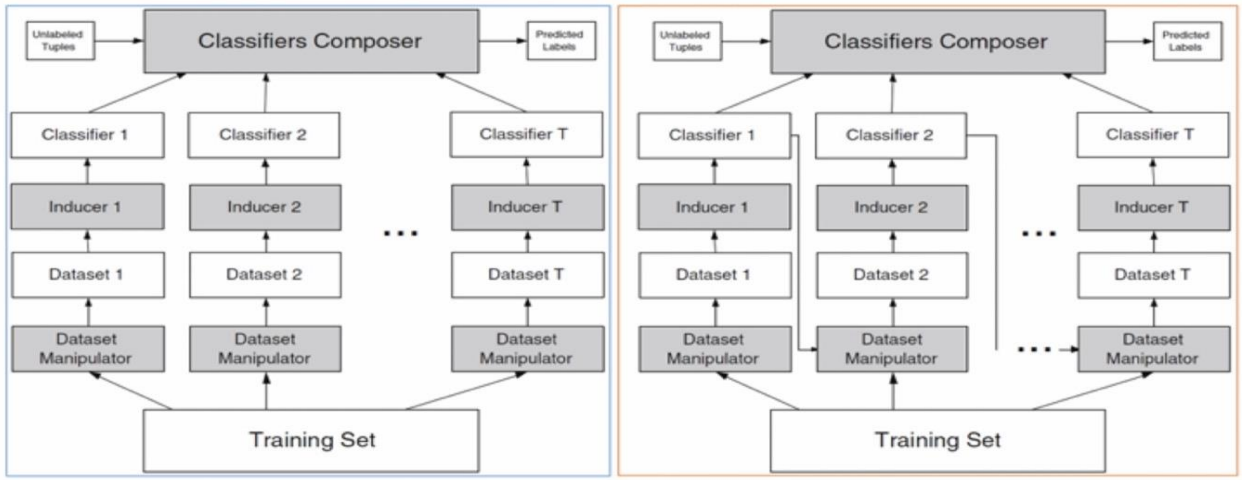
부스팅 (Boosting)

여러 개의 **약한 모델들을 모아**서 하나의 강한 모델을 만드는 기법



2 앙상블 기법

부스팅



배깅과 달리 부스팅은, Bootstrap 샘플 대신 **전체 샘플을 학습에 이용**하며

이전에 학습된 모델의 결과물이 다음 모델의 학습에 영향을 미침

왼쪽 그림은 배깅을, 오른쪽 그림은 부스팅을 의미

2 앙상블 기법

부스팅

부스팅의 종류

AdaBoost

기존 weak learner의 단점이

다음 학습의 **데이터에 반영**되는 방식

GBM

기존 weak learner의 단점이

다음 학습의 **Gradient에 반영**되는 방식

⋮

XGBoost, LightGBM 모델의 기반

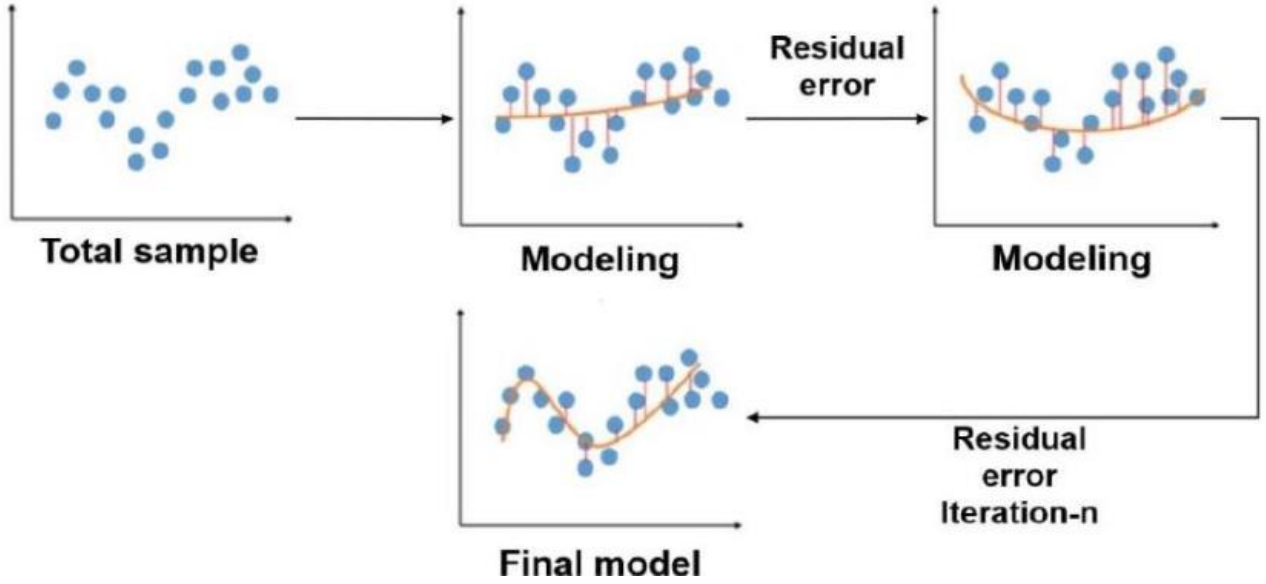
부스팅 | GBM

GBM (Gradient Boosting Machine)

Gradient Descent 의 원리를 이용해 약점을 보완함으로써

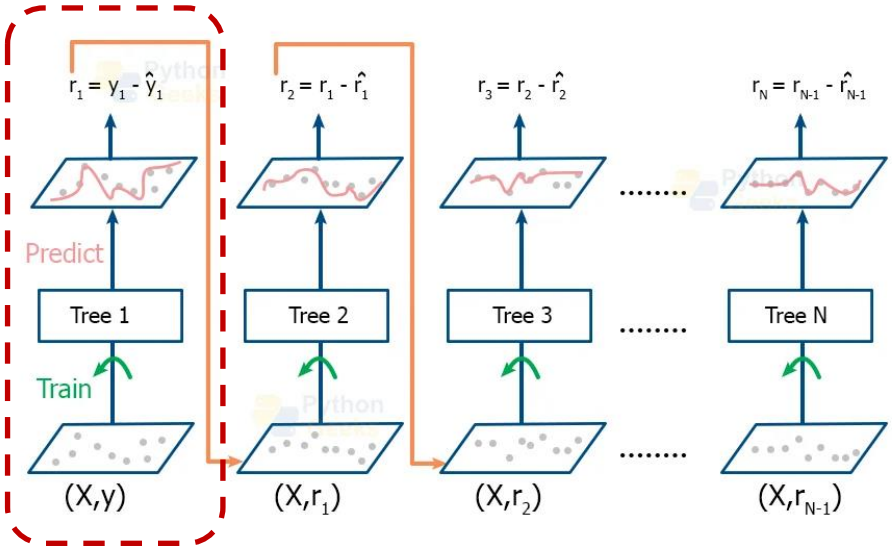
강력한 모델을 만드는 방법으로, 분류 및 회귀 문제에 모두 이용 가능

Gradient Descent 에 대한 내용은 24-1 리드오프 자료 참고



부스팅 | GBM

선형회귀에서의 GBM 작동원리



회귀 fitting 시 아래와 같이 잔차가 발생

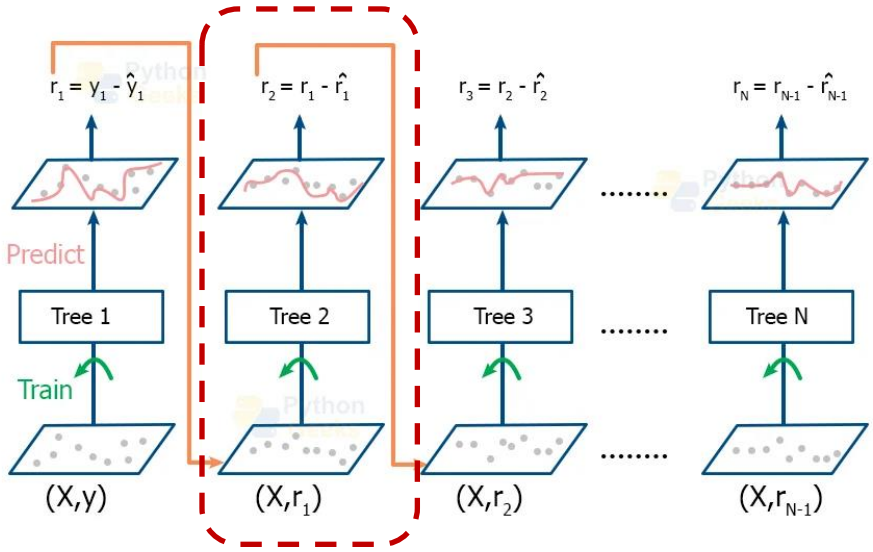
$$r_1 = y - \hat{y}_1$$

y 값 대신

이전 모델이 찾지 못한 오차 r_1 을 추정

부스팅 | GBM

선형회귀에서의 GBM 작동원리



추정한 $\hat{r}_1$ 을 이용하여 두번째 잔차를 계산

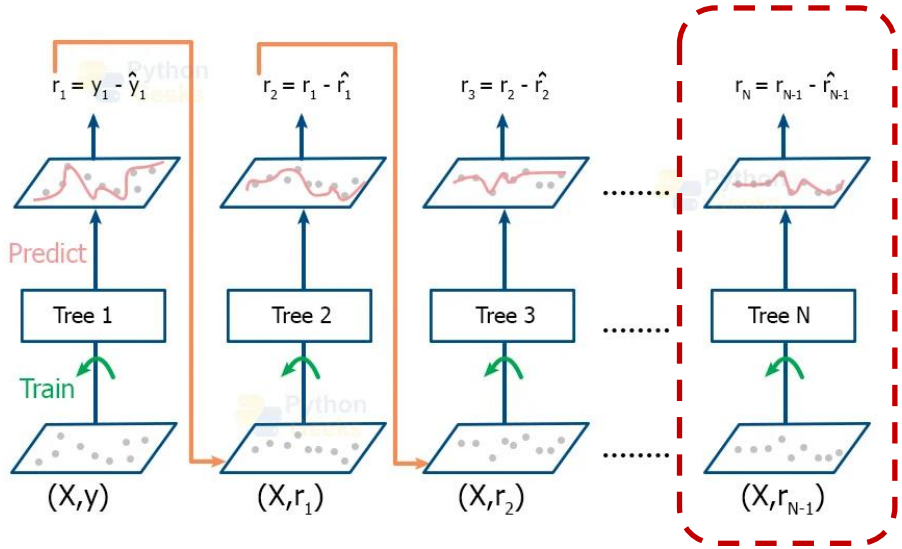
$$r_2 = r_1 - \hat{r}_1$$

첫번째 단계와 마찬가지로

이전 모델이 찾지 못한 오차 r_2 를 추정

부스팅 | GBM

선형회귀에서의 GBM 작동원리



이 과정을 n번 반복했을 때

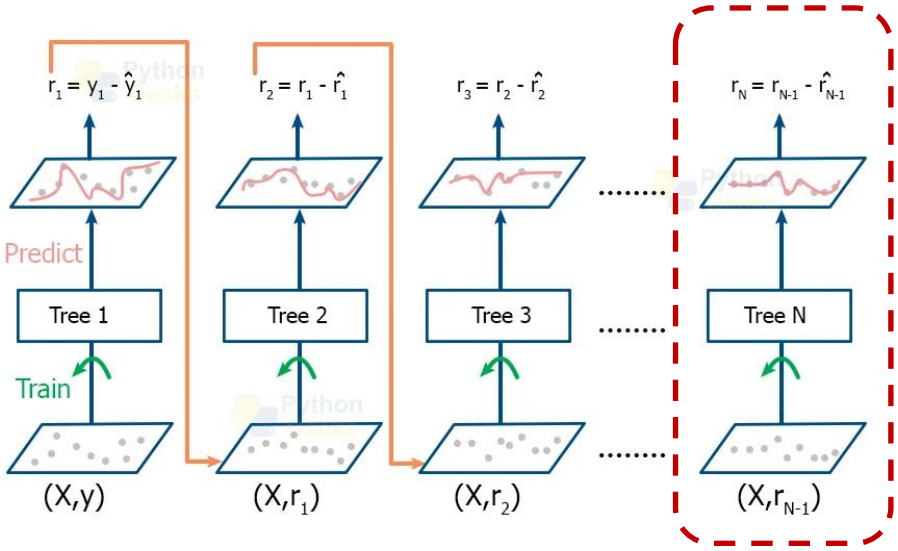
$f_1(x) = \hat{y}_1,\ f_k(x) = \hat{r}_{k-1}$ 이라고 두면,

$y = f(x)$ 는 아래와 같이 나타낼 수 있음

$$\boldsymbol{f(x) = f_1(x) + \cdots + f_n(x) + r_n}$$

부스팅 | GBM

선형회귀에서의 GBM 작동원리



결과적으로 약한 모델 $f_k(x)$ 들을 누적해가면서
잔차가 매우 작아진 매우 강력한 모델을 생성

부스팅 | GBM

GBM 모델의 이름에는 왜 Gradient 라는 단어가 붙은 걸까?

선형회귀에서의 GBM 작동원리

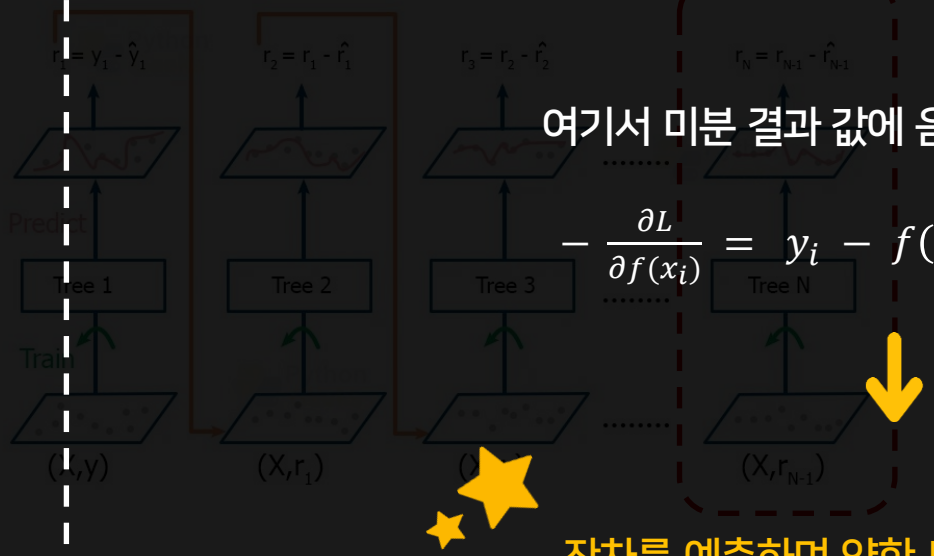
회귀 문제의 손실함수인 OLS 를 미분하면

$$min\ L = \frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i))^2 \quad \rightarrow \quad \frac{\partial L}{\partial f(x_i)} = f(x_i) - y_i$$

여기서 미분 결과 값에 음수 부호를 취해주면

$$-\frac{\partial L}{\partial f(x_i)} = y_i - f(x_i) = \text{Residual}$$

결과적으로 약한 모델 $f_k(x)$ 들을 누적해가면서 잔차가 매우 작아진 매우 강력한 모델을 생성



잔차를 예측하며 약한 모델을 쌓는다는 건

결국 Negative Gradient 방향으로 학습을 시킨다는 것과 같은 의미

2 앙상블 기법

GBM | 학습 프로세스

Height (m)	Gender	Age	Weight (kg)	Residual
1.8	Male	27	88	23.8
1.7	Male	44	68	3.8
1.7	Male	58	76	11.8
1.5	Female	15	35	-29.2
1.6	Female	25	54	-10.2

$76 - 64.2 = 11.8$

Average Weight

64.2

Y 값들(= Weight)의 **평균**을 계산해

Single leaf 값으로 할당해주고,

이를 이용하여 첫번째 잔차값을 계산

GBM | 학습 프로세스

Height (m)	Gender	Age	Weight (kg)	Residual
1.8	Male	27	88	23.8
1.7	Male	44	68	3.8
1.7	Male	58	76	11.8
1.5	Female	15	35	-29.2
1.6	Female	25	54	-10.2

64.2

첫번째 잔차값(= Residual)을
예측하는 하나의 **Tree를 생성**

만약 Leaf node에 값이 여러 개 있을 경우,
이들의 **평균값**으로 예측값을 대체

GBM | 학습 프로세스

Height (m)	Gender	Age	Weight (kg)	Residual
1.8	Male	27	88	23.8
1.7	Male	44	68	3.8
1.7	Male	58	76	11.8
1.5	Female	15	35	-29.2
1.6	Female	25	54	-10.2

64.2

$$Predicted\ Weight = 64.2 + Learning\ Rate \times (-10.2) = 54$$

생성된 모델(= Single leaf + Tree) 을
이용하여 **결과값을 예측**

2 앙상블 기법

GBM | 학습 프로세스

Height (m)	Gender	Age	Weight (kg)	Residual
1.8	Male	27	88	23.8
1.7	Male	44	68	3.8
1.7	Male	58	76	11.8
1.5	Female	15	35	-29.2
1.6	Female	25	54	-10.2

64.2

$$Predicted\ Weight = 64.2 + Learning\ Rate \times (-10.2) = 54$$

Single leaf 에 잔차를 그냥 더하는 게 아닌,

일정 학습률을 곱해준 뒤 더해야 함

잔차를 그냥 더하게 되면,

훈련데이터를 완벽히 예측해 **Overfitting 발생**

학습률에 대한 내용은 24-1 리드오프 자료 참고

GBM | 학습 프로세스

Height (m)	Gender	Age	Weight (kg)	Residual	Residual	Residual
1.8	Male	27	88			
1.7	Male	44	68			
1.7	Male	58	76			
1.5	Female	15	35			
1.6	Female	25	54			

...

앞의 과정의 반복을 통해

잔차를 예측하는 **Tree들을 계속 쌓아나가**며,

성능이 높아진 하나의 강력한 모델을 생성

2 앙상블 기법

GBM | XGBoost, LightGBM

XGBoost, LightGBM

수행시간이 느리고 **과적합**에 취약하다는 기존 GBM 모델의 단점을 개선시킨 모델

XGBoost

데이터를 분할하여 bucket 단위로
split point 탐색

결측치 처리 방향을 미리 정해
Sparse 한 변수의 처리 시간 단축

Column-wise 로 **사전에 sort**하여
처리 시간 단축

하드웨어 개선을 통한 계산 속도 증대

LightGBM

Gradient가 큰 객체들에 집중하고,
작은 객체들은 랜덤하게 drop 처리

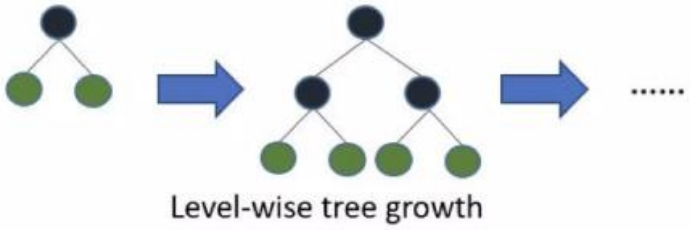
서로 배타적인 변수들은
하나의 bundle로 묶어서 처리

2 앙상블 기법

GBM | XGBoost, LightGBM

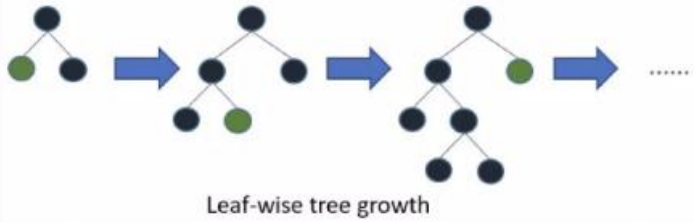
두 모델 간에는 **트리를 분할하는 방식**에서 차이점이 존재

XGBoost:



트리 깊이를 최소화하면서
균형 잡힌 트리를 만들어 감

LightGBM:



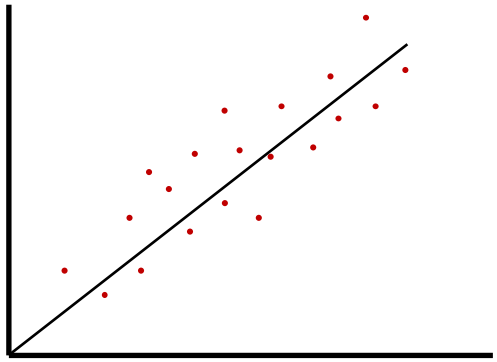
손실이 큰 트리에 대해서
지속적으로 분할해 나감

3

비선형 모델

비선형 모델

선형 모델



해석 & 추론 강점

하지만, 예측 면에서는 제한적임

현실의 데이터는 선형적이지 않은 경우가 많음

비선형 모델

해석력은 최대한 유지,

선형성 완화 및 성능 향상

비선형 모델

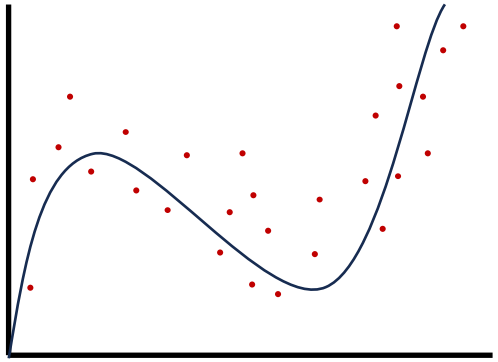
선형 모델

해석 & 추론 강점

하지만, 예측 면에서는 제한적임

현실의 데이터는 선형적이지 않은 경우가 많음.

비선형 모델



해석력은 최대한 유지,

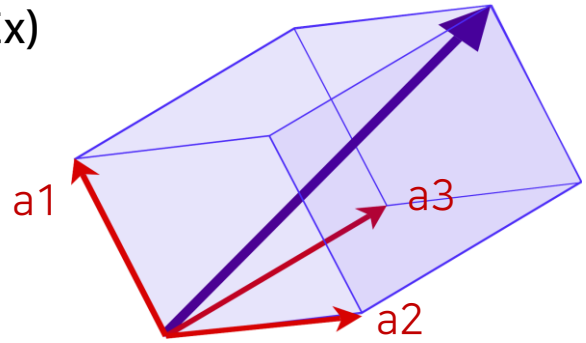
선형성 완화 및 성능 향상

용어 정리

기저 (Basis)

어떤 공간 내 **모든 벡터들을 나타낼 수 있게** 해주는 벡터들의 집합

Ex)



[a1, a2, a3] 는 R^3 의 **기저**

이를 구성하는 각각은 R^3 의 **기저벡터**

n 개의 선형 독립 axis

(Red line)

↓

n 차원 벡터 및 벡터 공간 생성

(Purple line)

용어 정리

기저함수 (Basis Function)

함수 공간의 기저를 이루는 함수

무한 차원 벡터 공간에서의 기저벡터와 유사한 개념으로,

모든 연속함수들은 **기저 함수들의 선형결합**으로 표시할 수 있음

Ex)

$$f(x)$$
$$= a\varphi_1(x) + b\varphi_2(x) + c\varphi_3(x)$$
$$= ax^2 + bx + c$$

↓

$[\varphi_1(x),\ \varphi_2(x),\ \varphi_3(x)]$ 는

함수 공간의 기저에 포함되며,

이들 각각은 2차 다항식의 **기저함수**

3 비선형 모델

비선형 모델

비선형 모델의 Key idea

X와 Y가 비선형 관계에 있으므로, input matri에 해당하는 X를 비선형적으로 transform

$$X \rightarrow Z$$

변환된 input matrix Z와 Y를 linear한 모델로 학습

➔ X : Y의 공간에서는 non-linear fit이지만, Z : Y의 공간에서는 linear fit이 적용됨.

$$Y = \beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \beta_3 Z_3 + \ \ldots + \beta_n Z_n$$

비선형 모델

비선형 모델의 Key idea

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X)$$

X와 Y가 비선형 관계에 있으므로, input matri에 해당하는 X를 비선형적으로 transform

이때 선형결합이 가능한 형태로 transform 해주는 것이

기저함수(Basis Function)

Linear Basis Expansion of X

변환된 input matrix Z와 Y를 linear한 모델로 학습

➔ X : Y의 공간에서는 non-linear fit이지만, Z : Y의 공간에서는 linear fit이 적용됨.

$$Y = \beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \beta_3 Z_3 + \ \ldots + \beta_n Z_n$$

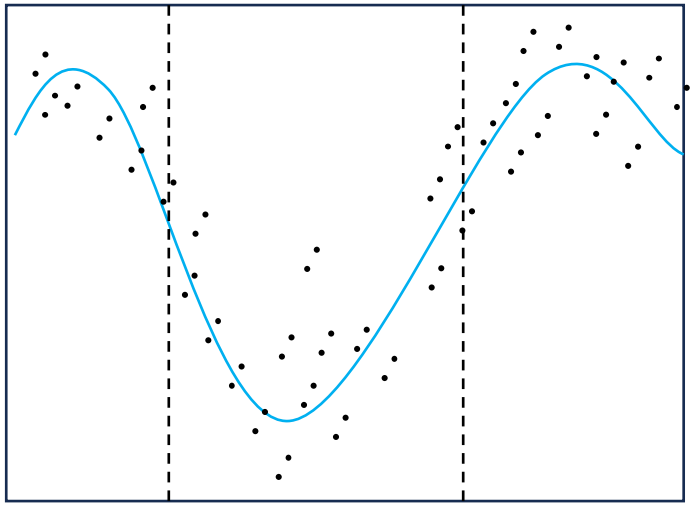
기저함수 기반 모델

Piecewise Polynomial

설명변수의 공간을 매듭점(knot)으로 분할하고,

각 공간의 다항 함수를 추정하는 회귀 방법

원활한 이해를 돕기 위해, X 벡터를 1-Dimensional 벡터라고 가정



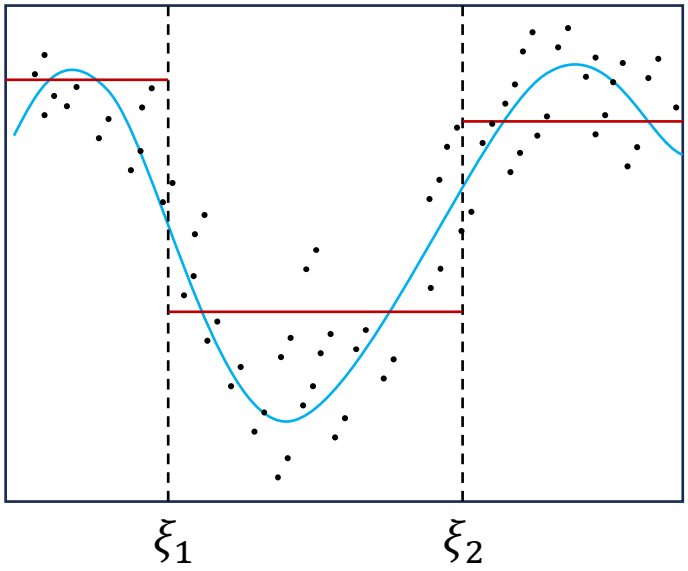
기저함수 기반 모델

Piecewise Constant Model

Knot으로 분할된 각 공간에 Constant 함수를 적합한 경우

$$h_1(x) = I(X < \xi_1), \qquad h_2(x) = I(\xi_1 < X < \xi_2), \qquad h_3(x) = I(X \geq \xi_2)$$

I : indicator function



$$f(X) = \sum_{m=1}^{3} \beta_m h_m(X)$$

기저함수 기반 모델

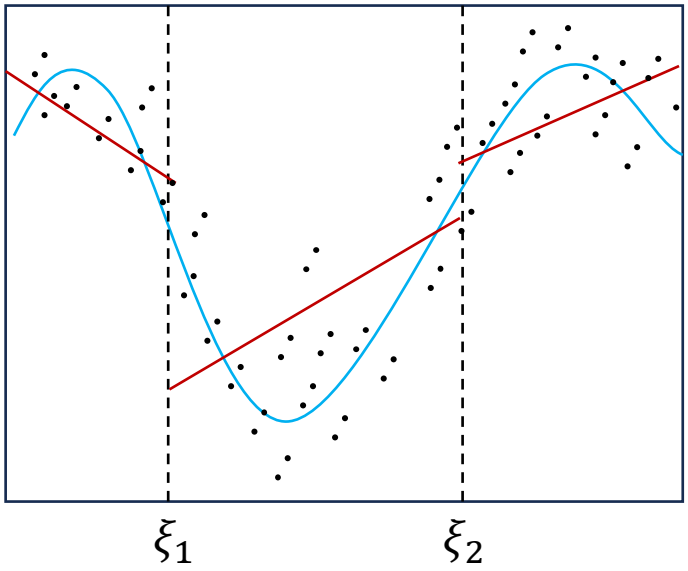
Piecewise Linear Model

Knot으로 분할된 각 공간에 Linear 함수를 적합한 경우

$$\beta_1 I(x < \xi_1), \quad \beta_2 I(\xi_1 < x < \xi_2), \quad \beta_3 I(x \geq \xi_2)$$

$$\beta_4 XI(x < \xi_1), \quad \beta_2 XI(\xi_1 < x < \xi_2), \quad \beta_3 XI(x \geq \xi_2)$$

I : indicator function



$$f(X) = \sum_{m=1}^{6} \beta_m h_m(X)$$

기저함수 기반 모델

Piecewise Linear Model

행렬 $[h_1(X), \cdots, h_6(X)]$ 에 대해

LSE를 이용하여 $f(x)$ 를 추정

$$X = \begin{bmatrix} 5 \\ 3 \\ 11 \\ 7 \end{bmatrix} \quad \cdots\cdots \quad H = \begin{array}{c} \begin{matrix} h1 & h2 & h3 & h4 & h5 & h6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 11 \\ 0 & 1 & 0 & 0 & 7 & 0 \end{bmatrix} \end{array}$$

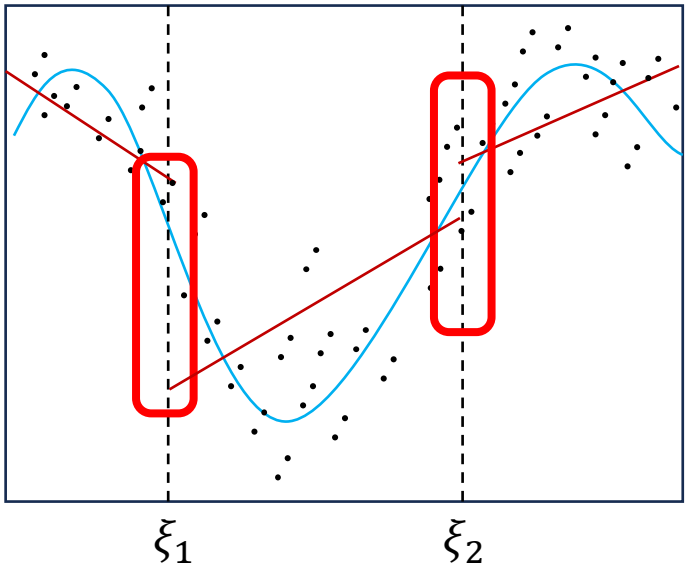
Matrix H가 x 역할을 하는 셈!

$$\hat{\beta} = \begin{bmatrix} \widehat{\beta_1} \\ : \\ : \\ \widehat{\beta_6} \end{bmatrix} \quad \cdots\cdots \quad (H^T H)^{-1} H^T y \text{ 로 쉽게 추정 가능!}$$

3 비선형 모델

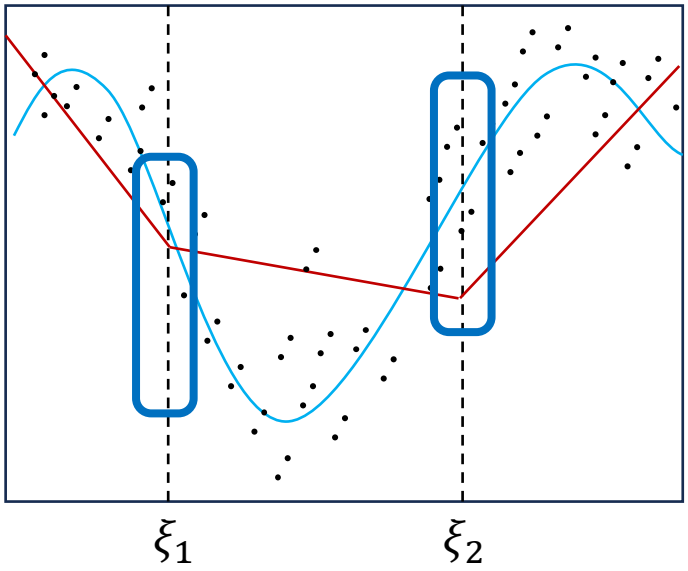
기저함수 기반 모델

Knot 지점에서
불연속성(Discontinuity) 문제 발생
비합리적인 예측을 할 수 있음



기저함수 기반 모델

Knot를 기준으로
양쪽 함수의 좌극한과 우극한이
같도록 만들어주는 제약식 설정



$$f(\xi_1^-) = f(\xi_1^+)$$
$$f(\xi_2^-) = f(\xi_2^+)$$

제약식으로 연속성 확보!

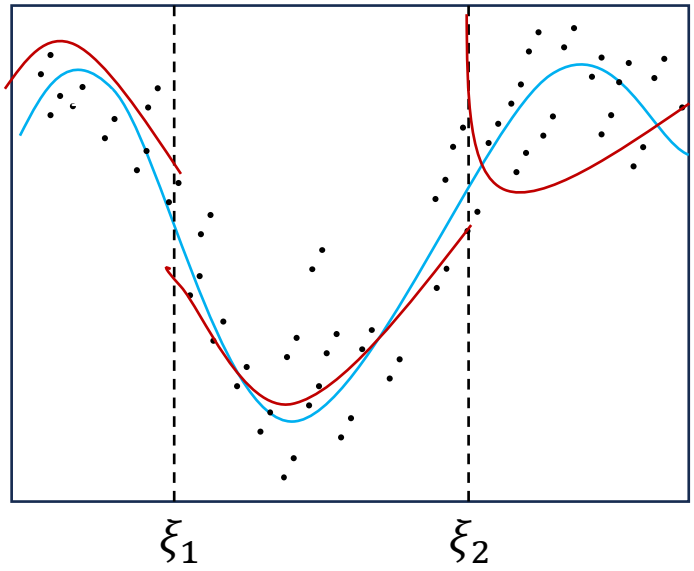
기저함수 기반 모델

Cubic Spline

Piecewise 모델의 일종으로, 각 공간에 **3차(Cubic) 다항함수**를 적합한 경우

$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \epsilon_i & if\ x_i < \xi_1 \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i & if\ \xi_1 < x_i \le \xi_2 \\ \beta_{03} + \beta_{13}x_i + \beta_{23}x_i^2 + \beta_{33}x_i^3 + \epsilon_i & if\ x_i \ge \xi_2 \end{cases}$$

I : indicator function



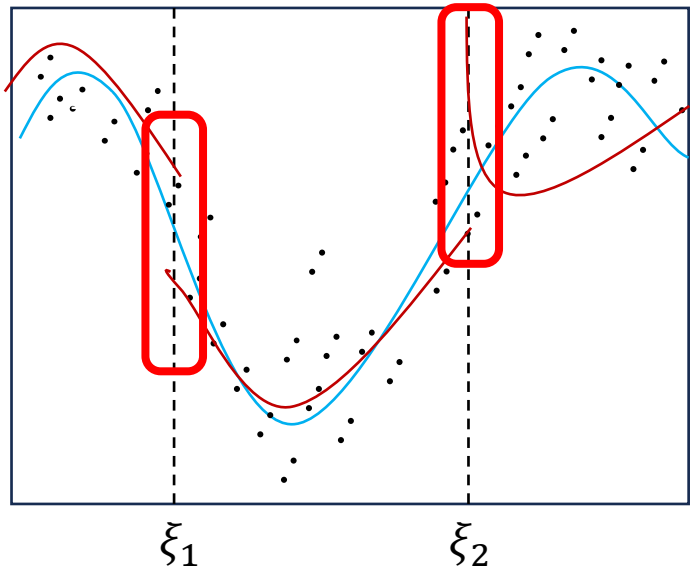
3차 함수 사용이유

3차 다항식만 고려해도

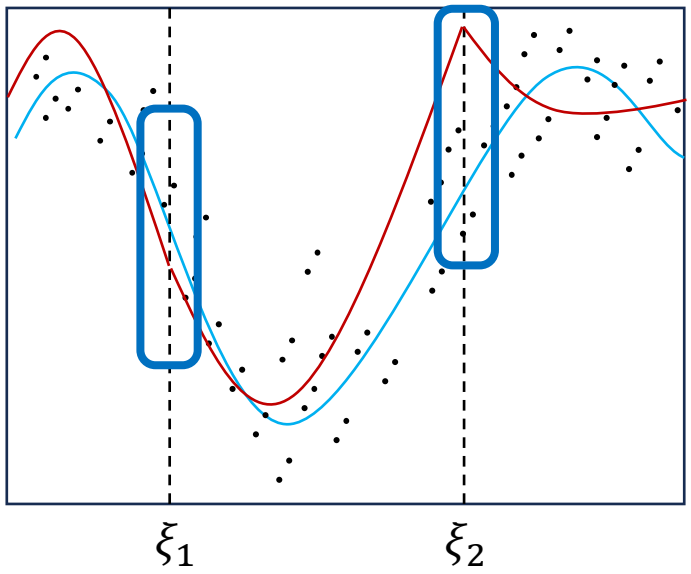
현실의 X와 Y의 관계 고려 가능함.

기저함수 기반 모델

Discontinuous



Continuous



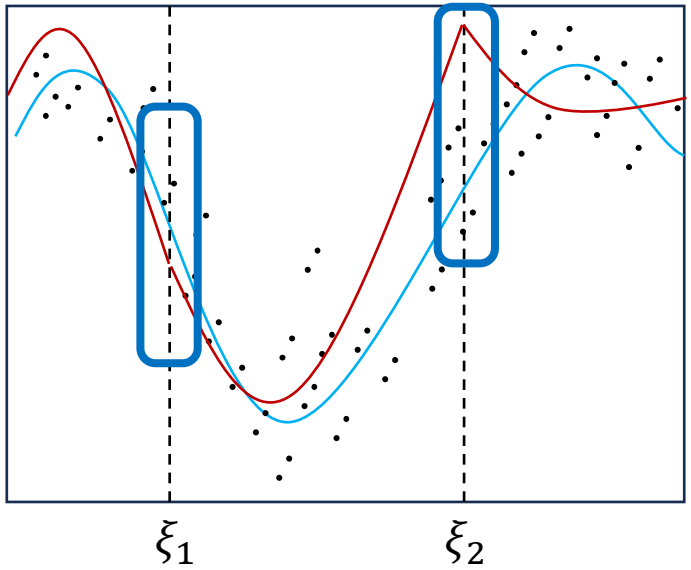
Piecewise linear 모델에서처럼

제약식을 추가해서 불연속성 문제 해결

기저함수 기반 모델

Discontinuous

Continuous



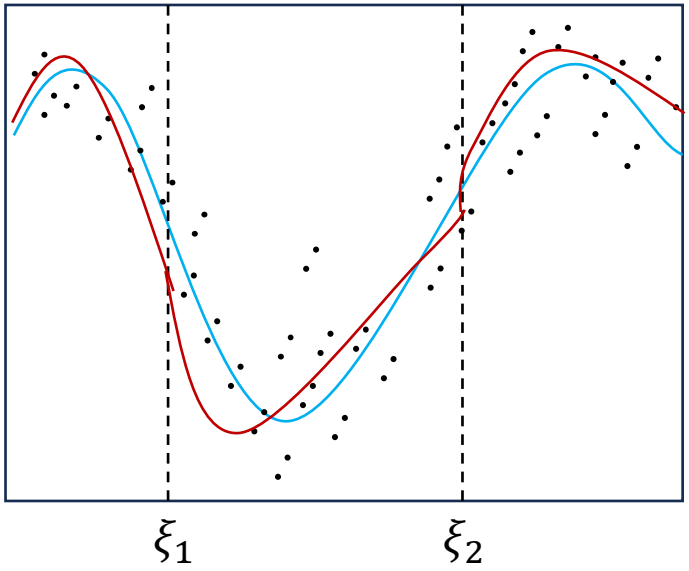
하지만, 연속성을 부여해도

부자연스럽게 꺾인 지점이 생겨 미분불가능의 문제 발생

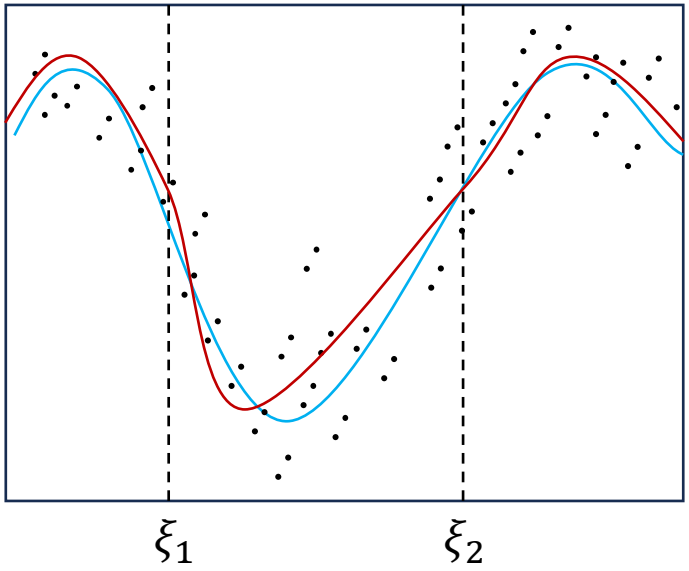
3 비선형 모델

기저함수 기반 모델

Continuous First Derivative



Continuous Second Derivative



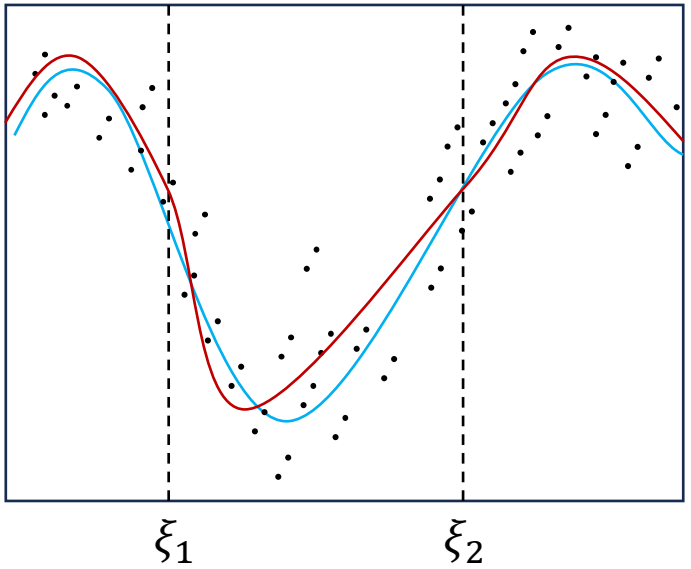
Knot를 기준으로 좌미분계수와 우미분계수가 같다고 설정

미분가능성에 대한 제약 추가!

기저함수 기반 모델

Continuous First Derivative

Continuous Second Derivative

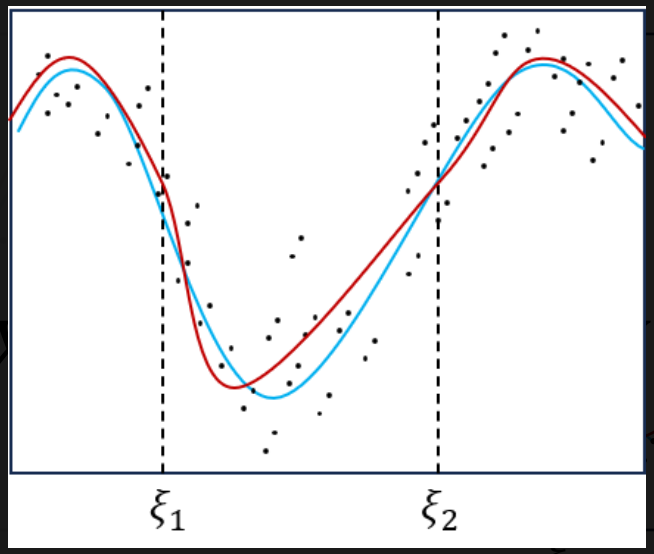


2차 미분계수까지 같다는 제약을 추가하여
곡선을 더욱 매끄럽게 만들 수 있음!

3 비선형 모델

기저함수 기반 모델

Cubic Spline의 Basis function 형태의 표현



$$f(X) = \sum_{m=1}^{6} \beta_m h_m(X)$$

$$h_1(x) = 1, \; h_2(x) = x, \; h_3(x) = x^2, \; h_4(x) = x^3,$$

$$h_5(x) = (x - \xi_1)^3, \; h_6(x) = (x - \xi_2)^3$$

3 비선형 모델

기저함수 기반 모델

장점

- 복잡한 데이터에 적합하기 위해 차수를 높이는 게 아니라,
  Knot를 늘려 **단순한 다항식으로도 유연하게 표현**할 수 있음
- **데이터의 보간법(interpolation)**으로도 사용할 수 있음

단점

- Knot 개수와 위치를 산정하는 기준이 명확하지 않음
- **독립변수가 3개 이상**으로 늘어날 경우
  모델이 심각하게 복잡해져서 사용하기 어려움

단점을 보완하고자 Natural Spline, Smoothing Spline 등이 있음

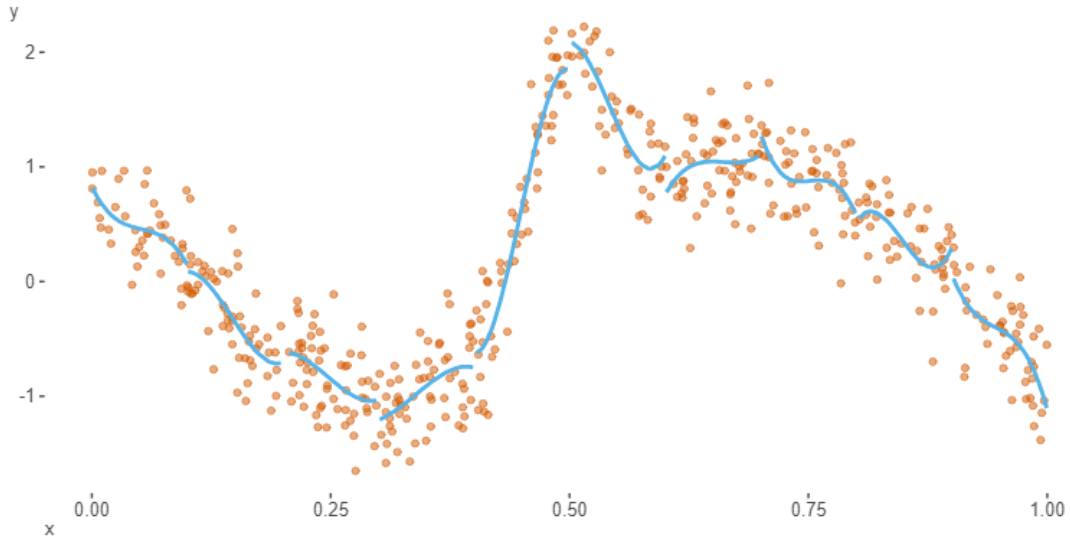
3 비선형 모델

GAM

GAM (Generalized Additive Model)

기존의 선형 모델에서 **가법성은 유지**하면서도

각 변수에 Non - linear한 적합을 가능하게 한 방법



GAM

GAM (Generalized Additive Model)

$$y = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon$$

$$y = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon$$

다중선형회귀 모델을 확장시킨 것으로 이해할 수 있음.

$\beta_j x_{ij}$ 의 선형결합을 **비선형함수 $f_j(x_{ij})$ 들의 선형결합**으로 확장

GAM

GAM (Generalized Additive Model)

각각의 Y에 대한 각 예측변수들의
기여를 '더하여' 표현함

$$y = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon$$

⋮

여전히 각 변수 x_{ij} 을 각각의 함수 f_j에 부과하고 있기에 가법성이 유지됨.

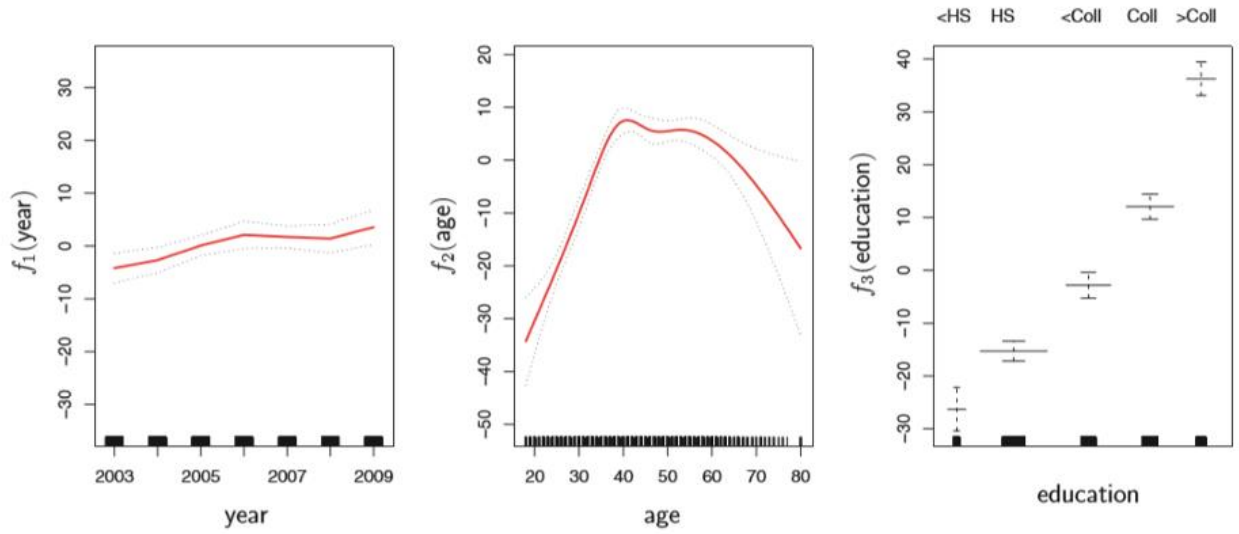
다중선형회귀 모델을 확장시킨 것으로 이해할 수 있음.

$\beta_j x_{ij}$ 의 선형결합을 **비선형함수 $f_j(x_{ij})$ 들의 선형결합**으로 확장

3 비선형 모델

GAM

GAM (Generalized Additive Model)



각 변수별로 다양한 선형/비선형 함수를 적합하여

보다 유연한 관계에 fitting 가능

3 비선형 모델

GAM

장점

- 일반적인 선형모델이 놓칠 수 있는 비선형 관계를 파악할 수 있음
- 개별 변수의 영향력을 확인 할 수 있어 해석력 & 예측 성능 확보 가능

단점

- 모델이 가법적인게 단점이 되기도 함
- 변수의 개수가 많은 경우 변수 간 중요한 상호작용을 놓칠 수 있음

직접 상호작용 항을 넣어줌으로써 일부 완화시킬 수 있음

다음 주 예고

1. 클러스터링

2. 추천 시스템

THANK YOU