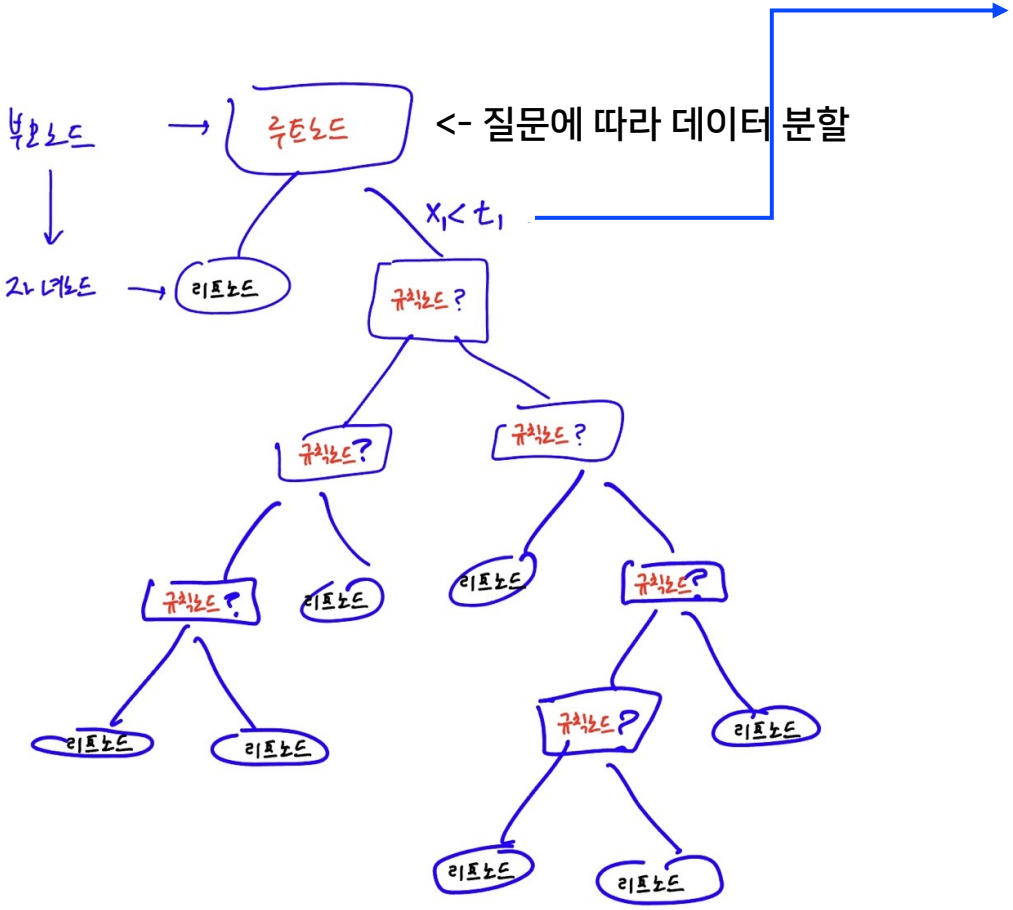
11주차 복습세션 1부

3조 윤희재 이승재 이원재 장윤서

목차

01. 트리모델과 하이퍼 파라미터 튜닝

결정트리



◆ 분류기준: 불순도

(t1를 정하는 기준)

1) 지니 불순도

1-(음성클래스 비율^2+양성클래스 비율^2)

Ex) 와인분류

노드 데이터 중 레드 20% 화이트 80%

: 지니불순도 = 1-(0.04+0.64) = 0.32

Ex) 다중분류도 똑같은 원리

노드 데이터 중 레드 10% 화이트 80% 로제 10%

:지니불순도=1-(0.01+0.01+0.64)=0.34

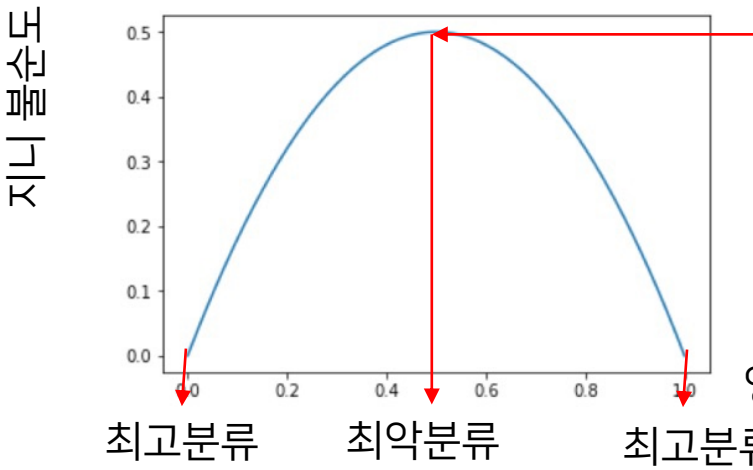
2) 엔트로피 : 제곱대신 log 사용

1-(log2(음성클래스 비율)x음성클래스 비율

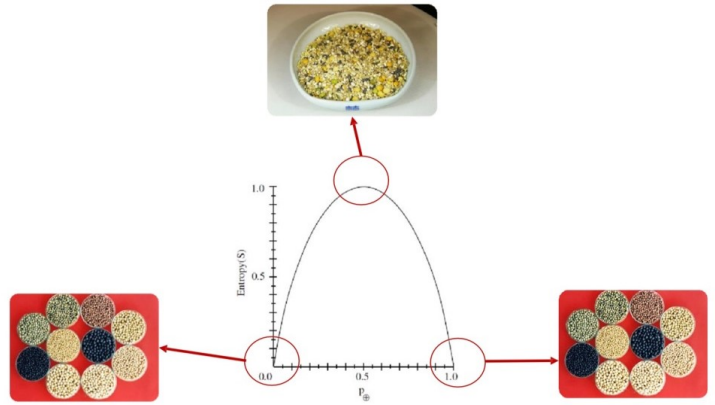
+log2(양성클래스 비율)x양성클래스 비율)

불순도

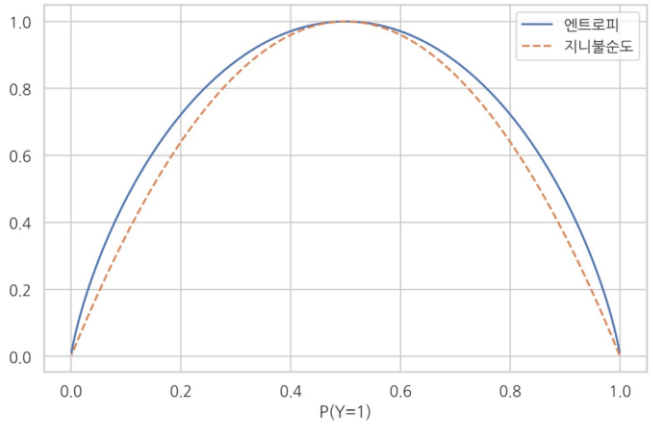
◆ 지니불순도



◆ 엔트로피 불순도



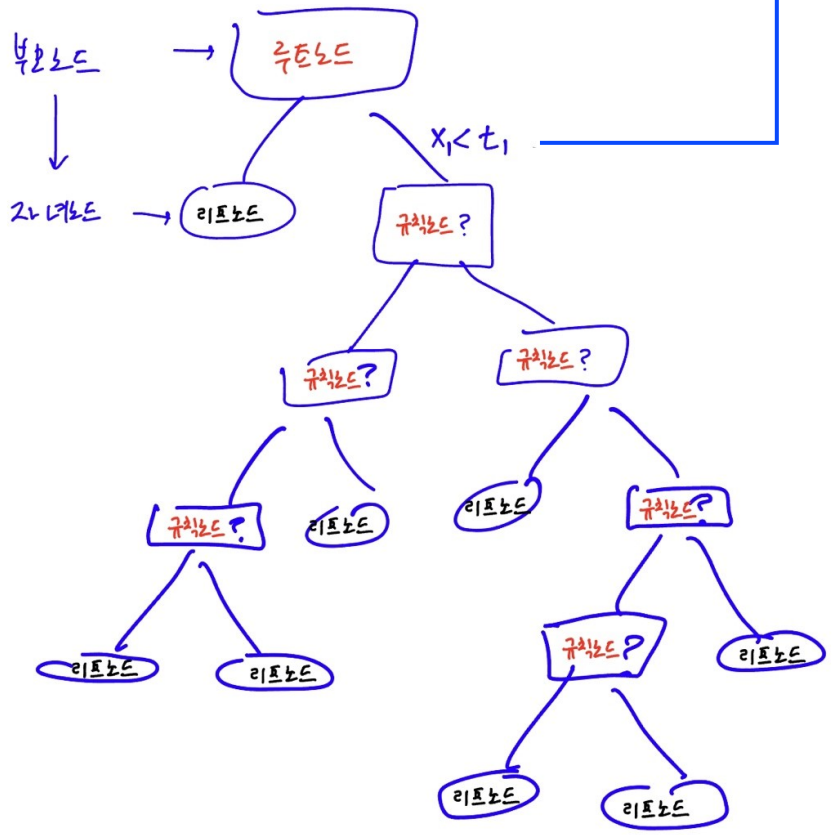
◆ 엔트로피 vs 지니



엔트로피는 시간이 오래 걸리는 대신 성능이 좋다

하지만 결과의 차이는 크지 않음

결정트리의 분기 기준



◆ T1를 정하는 기준

트리모델은 부모 노드 - > 자식 노드로 갈수록 불순도를 줄여감

T1은 부모-자식의 불순도 차이가 가장 큰 방향으로 결절됨

= 정보이득 :

$$\text{부모노드 불순도} - \sum \frac{\text{자녀노드 데이터 수}}{\text{부모노드 데이터 수}} \times \text{자녀노드 불순도}$$

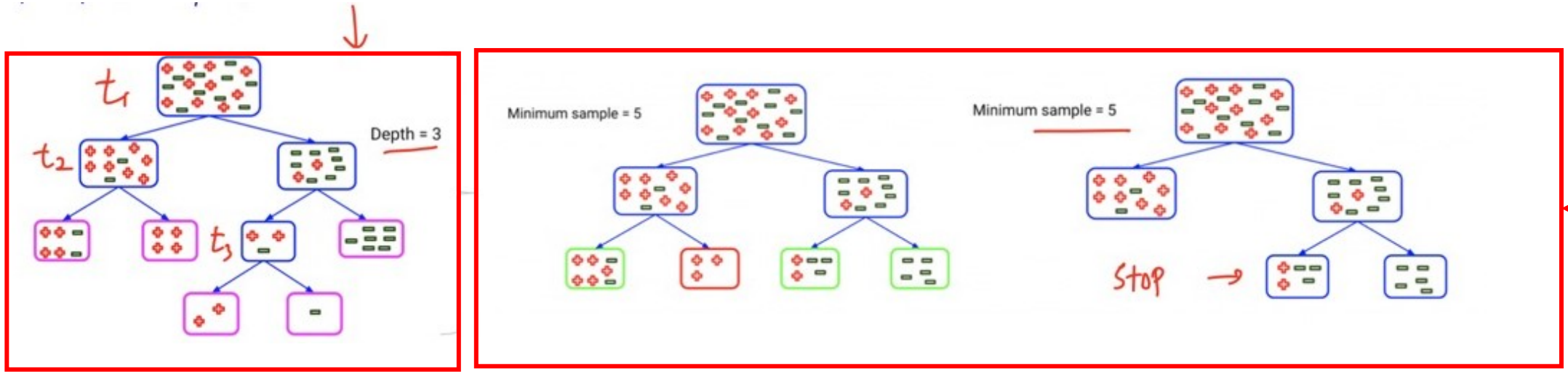
가중평균

◆ ! 트리 무제한 분기시, 과적합 문제 발생

결정트리의 규제(가지치기)

◆ 사전규제 (트리 생성 과정에서 제한)

Max_depth : 트리의 최대 깊이 제한



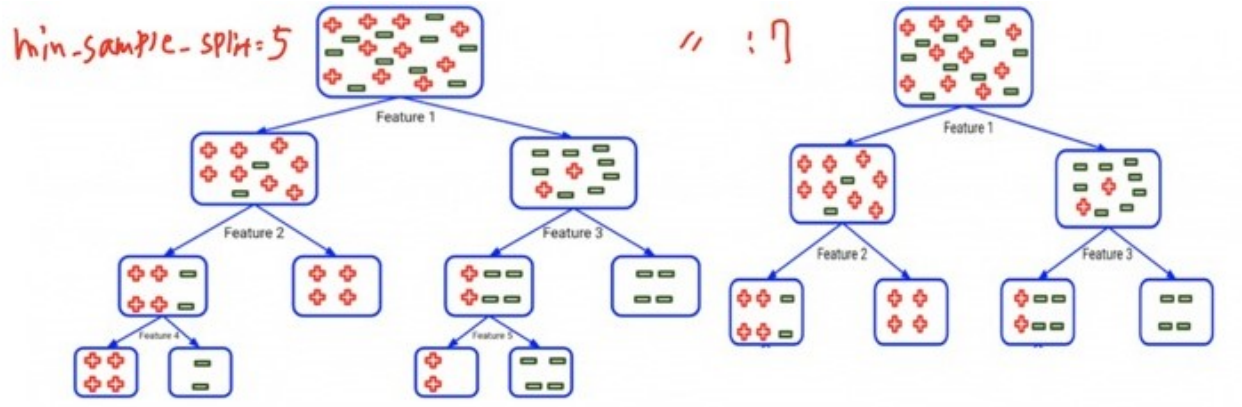
Min_sample_leaf : 리프노드가 되기 위한 최소 data 개수 (leaf 노드 개수 제한)

Max_leaf_nodes : 리프노드 자체의 최대 개수

결정트리의 규제(가지치기)

◆ 사전규제 (트리 생성 과정에서 제한)

Min_sample_split : 분기하기 위한 최소 샘플 개수



Max_features : 고려할 최대 feature 개수

-> 모델 설정 시 입력해야하는 설정값, hyper parameter

결정트리의 규제(가지치기)

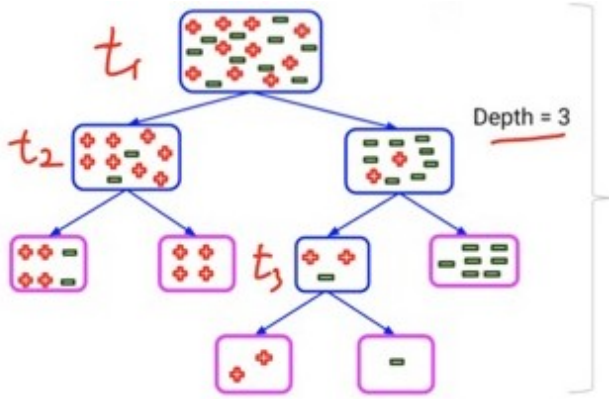
◆ 사후규제 (트리 생성 후 노드 삭제, 병합)

기준: ccp (cos complexity planing)

t: 트리 사이즈 (=트리 길이)

Cost complexity : cc(t)= (회귀 : SSE(t) / 분류: ERR(t) 오분류율) + alpha x L(t) 가중치 Leaf 노드 개수

Ex) T=3



트리모델의 예측 & 하이퍼 파라미터 튜닝

◆ 분류모델의 경우 예측

노드 내에 더 많은 class 값이 fitted y가 됨 (다수결)

ex) 노드 내 레드 20% 화이트 80% 일 경우 fitted y=화이트

◆ 회귀모델의 경우 예측

노드 훈련 data들의 y값의 평균

◆ 하이퍼 파라미터 튜닝

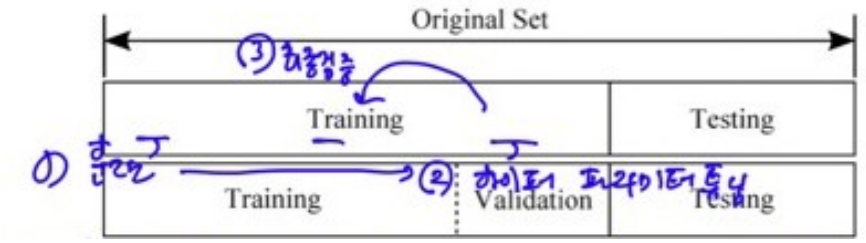
목표: 최선의 hyper parameter(alpha, tree size 등) 조합 찾기

! Test data로 하이퍼 파라미터 튜닝 시, 이번에는 test data에 과적합되는 문제가 발생

교차검증법

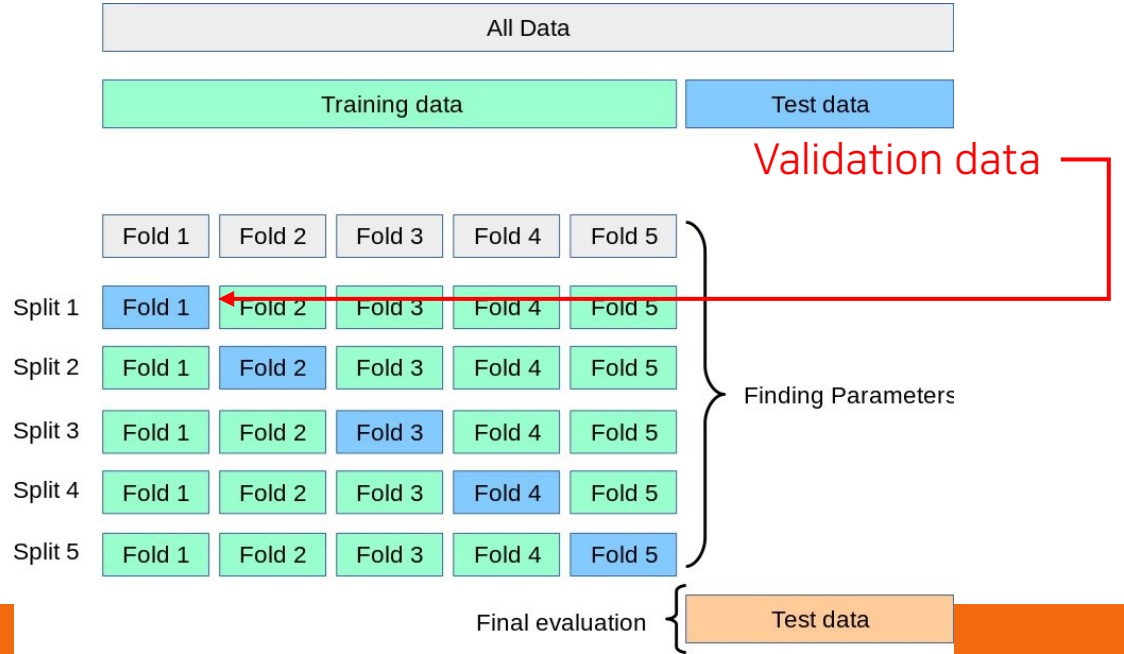
◆ 하이퍼 파라미터 튜닝 시 과적합을 피하기 위한 기법

1) 검증세트를 따로 분리(validation set)



! 문제점: 훈련 데이터가 적어짐 -> 데이터의 차이에 따라 결과가 들쭉날쭉해져 비교가 쉽지 않음

2) k-fold 교차검증법



Test data는 유지

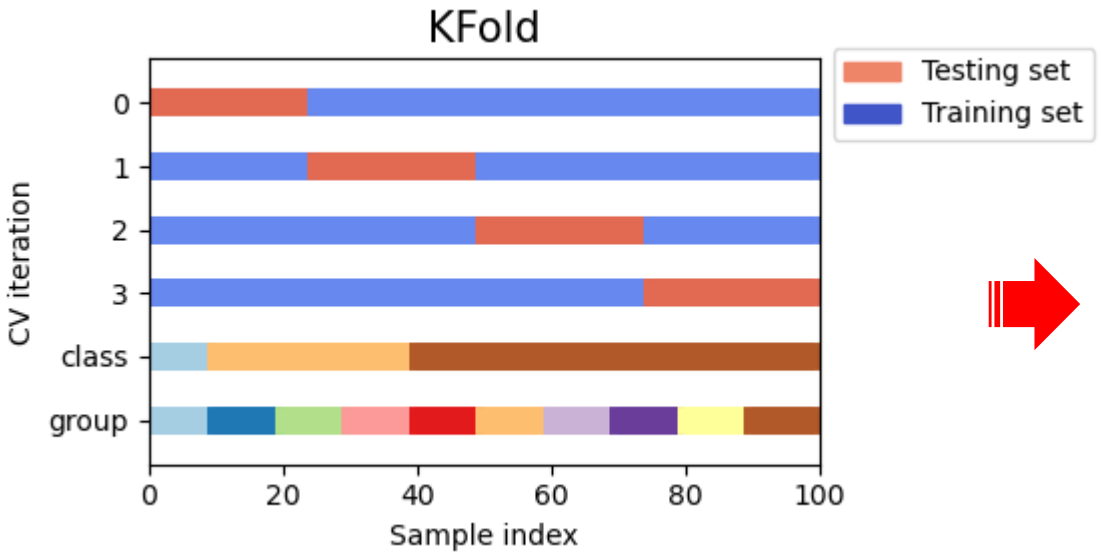
Train data를 k개 그룹으로 나누어 각각의 그룹을 한번씩 validation data로 테스트하는 방법

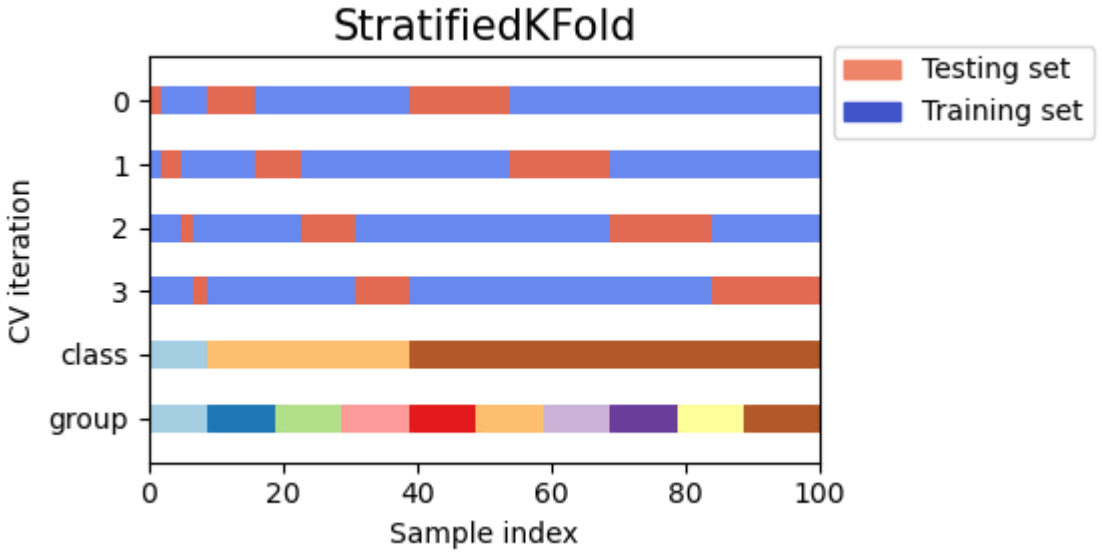
단, 소요시간이 늘어나고 k가 커질수록 데이터가 일반화되지 못한다는 단점이 있음

교차검증법

◆ 하이퍼 파라미터 튜닝 시 과적합을 피하기 위한 기법

3) stratified k-fold 교차검증법

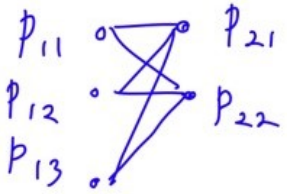




! 각 fold마다 class의 비율이 크게 차이남

Fold내에 클래스별 비율을 일정하게 맞춰주는 방법

하이퍼 파라미터 튜닝



서치된 하이퍼 파라미터 p1, p2에 대해 모든 조합(p1, p2)를 테스트해 최적의 조합을 찾음

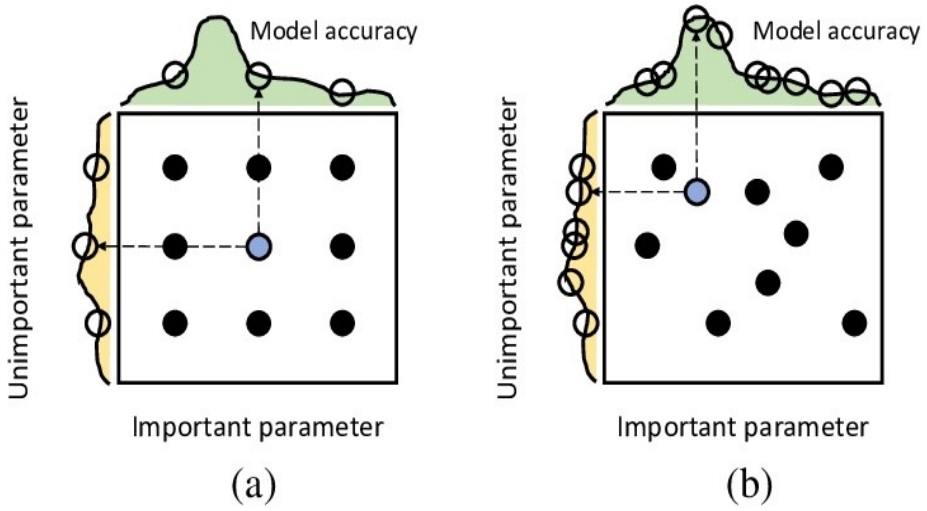
-> 하이퍼 파라미터의 서치 방법:

◆ 그리드 서치(일정간격을 두고 찾기)

정한 범위 내에서 일정한 간격으로 매개변수 탐색

! 단 매개변수의 범위를 미리 정하는게 어려울 수 있고 시간이 오래 걸림

그리드 서치 랜덤서치



◆ 랜덤서치(랜덤으로 찾기)

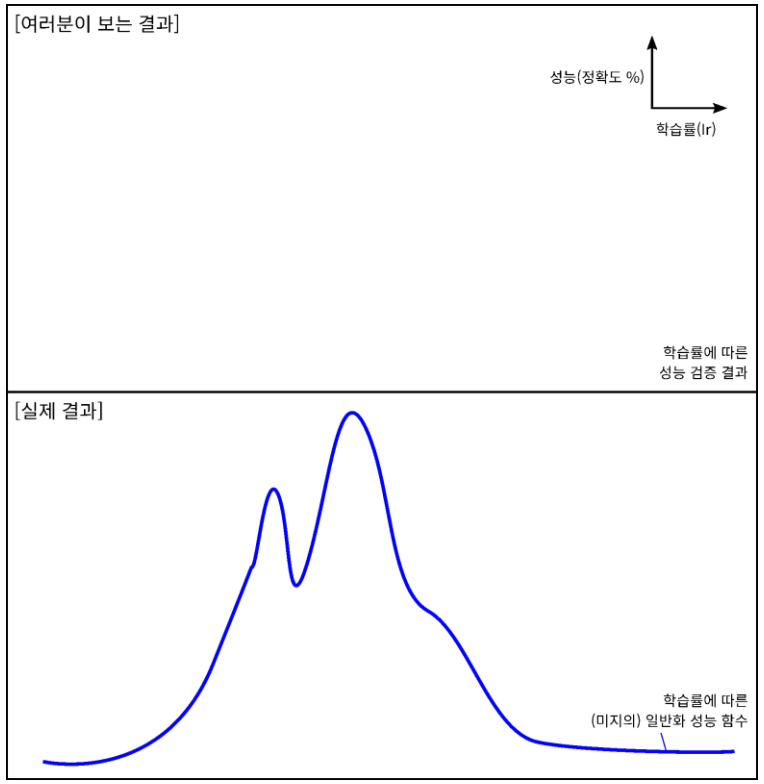
매개변수가 따르는 확률분포를 지정, 해당 분포에서 매개변수를 랜덤으로 추출

-> 시간에 비해 더 나은 조합을 발견할 수도 있음

01. 트리모델과 하이퍼 파라미터 튜닝

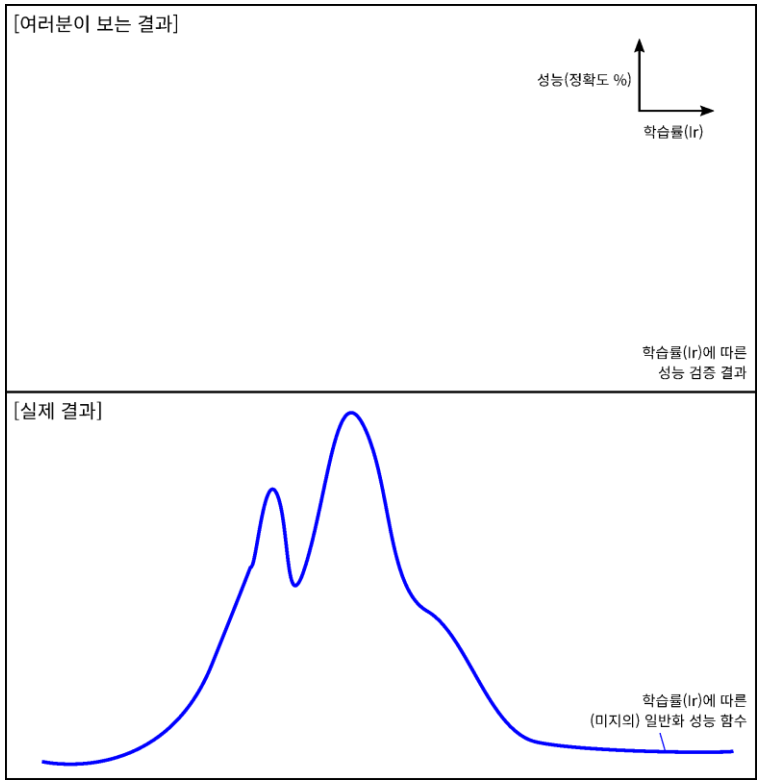
하이퍼 파라미터 튜닝 - 서치 방법의 비교:

일반적으로 최적 파라미터를 더 빨리 찾을 수 있는 방법



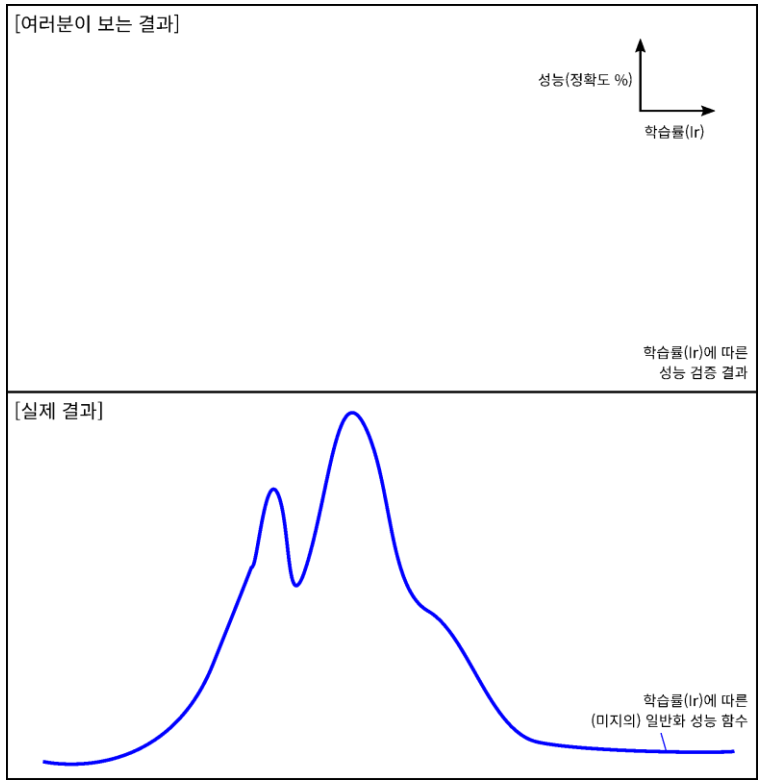
매뉴얼 서치

찾은 파라미터가 최선의 결과가 아닐 확률이 큼



그리드 서치

지정한 그리드 상에서의 최선의 결과 획득



랜덤 서치

그리드 서치에 비해 불필요한 반복 횟수를 줄이면서 정해진 그리드 사이에 위치한 파라미터에 대해서도 확률적으로 탐색이 가능

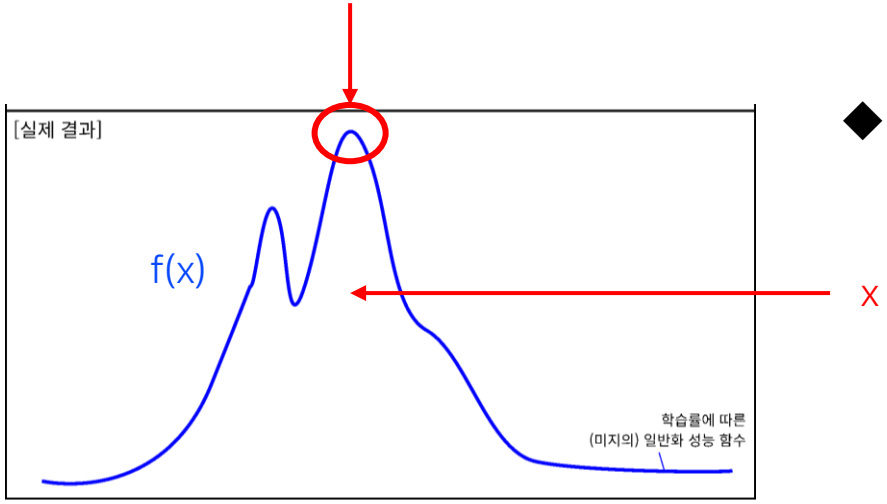
베이지안 최적화 기법

그리드 서치와 랜덤 서치 모두 매뉴얼 서치보다는 나아졌지만 여전히 이전 시도에서 얻은 사전지식(하이퍼 파라미터의 성능 결과)가 다음 시도에 반영되지 않기 때문에 불필요한 탐색을 반복할 가능성이 있음

➢ 이를 보완한 방법: **베이지안 최적화 기법**

어느 입력값 x; hyper parameter 에 대한 미지의 목적함수 f(x); 모델 점수 (ex.정확도) 를 상정

목표: f(x)를 최대로 만드는 최적해 x를 찾기



◆ Surrogate model

현재까지 조사된 (xi, f(xi))을 바탕으로 f(x)를 확률적으로 추정한 모델, Gaussian process를 주로 사용

Gaussian process(GP) : 특정 변수가 아닌 모종의 함수들에 대한 확률분포를 나타내는 확률 모델로 구성요소들 간의 결합분포(joint distribution)이 가우시안 분포를 따름

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$$

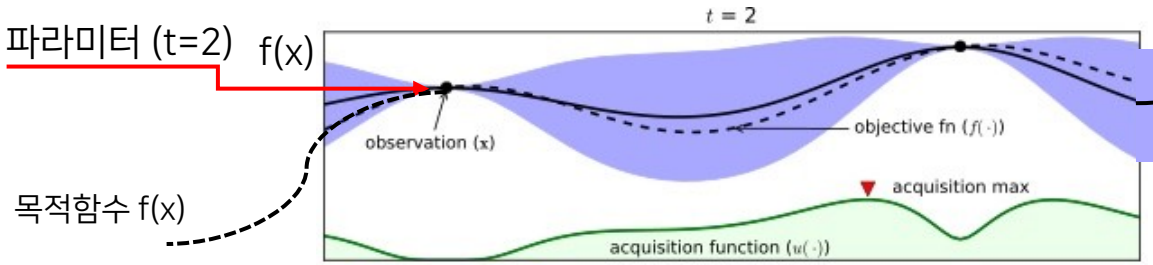
표현수식, k는 공분산 함수

01. 트리모델과 하이퍼 파라미터 튜닝

베이지안 최적화 기법

◆ Surrogate model이 f(x)를 추정하는 원리

1. 두개의 조사된 파라미터 (t=2)



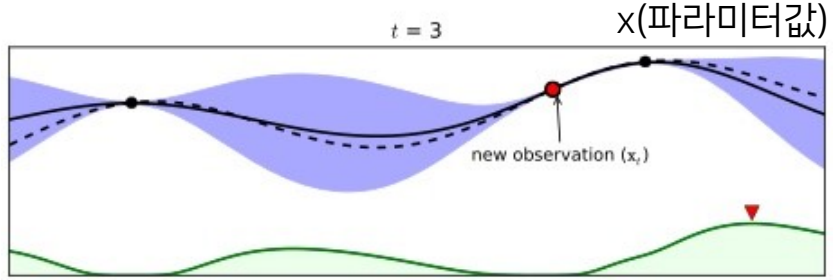
각 x 위치별 평균 μ(x)

각 x 위치별 표준편차 σ(x)

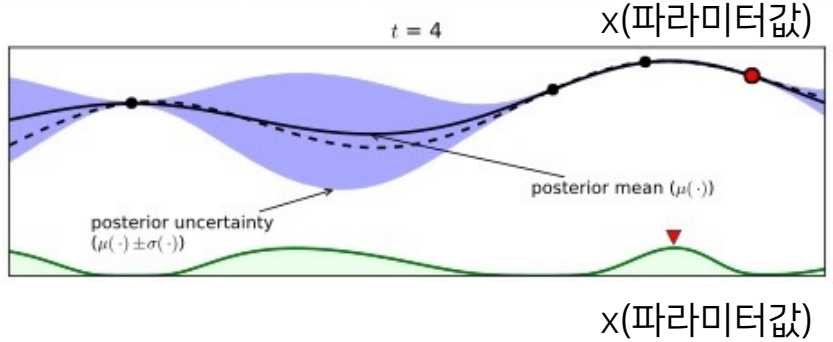
조사된 점에서 멀어질수록 표준편차가 커짐

-> 거리가 멀수록 이 지점에서 추정된 평균값에 대한 불확실성이 커짐

2. 파라미터 하나 더 조사 (t=3)



2. 파라미터 하나 더 조사 (t=4)



조사된 파라미터가 추가될수록 표준편차 영역이 줄어드며 실제

목적 함수에 대한 추정 범위가 점차적으로 압축됨

-> 조사된 점이 증가할수록 목적함수의 추정 결과에 대한 불확실성 감소

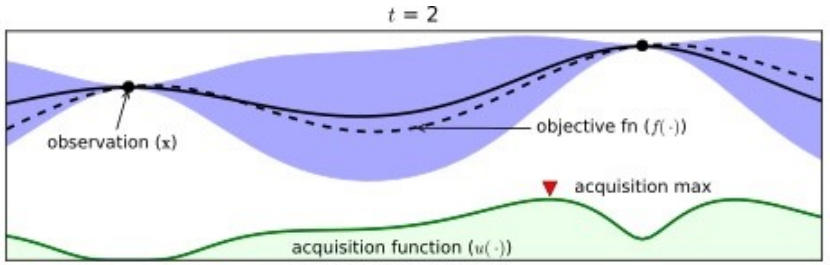
베이지안 최적화 기법

◆ Acquisition Function

주로 쓰이는 함수:

Surrogate model이 현재까지 목적함수에 대해 추정한 결과를 바탕으로 다음번에 조사하기에 가장 유용할 만할 파라미터 후보를 추천해주는 함수

f(x)



x(파라미터값)

Trade-off 관계에 있는 두 전략

1. 함수값이 더 큰 점 근방에서 최적 입력값 x를 찾을 확률이 높을 것이다

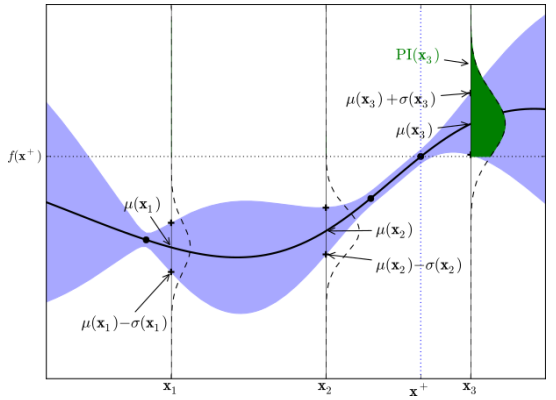
   -> Exploitation(착취, 수탈)전략

2. 불확실한 영역(표준편차가 큰 영역)에 최적 입력값이 존재할 가능성이 있으므로 이 부분을 추가로 탐색해야 할 것이다

   -> Exploration(탐색)전략

◆ Expected Improvement(EI)

Trade-off 관계의 두 전략을 모두 일정 수준 포함하도록 설계된 함수



02. 앙상블 기법

02. 앙상블 기법

앙상블 기법의 배경

◆ 결정트리 모델의 장단점

장점 : 수치형/범주형 변수 모두 다룰 수 있다
모델 설명이 쉽다
속도가 빠르다

가장 중요한 점을 만족 x
단점 : 예측 성능이 떨어진다
데이터의 미세한 노이즈나 변동에 크게 반응한다

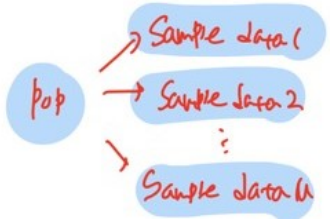
◆ 앙상블 학습 기법

Bagging / boosting

Bagging

◆ 이상적 샘플링

처음부터 모집단에서 여러 개의 sample data를 추출



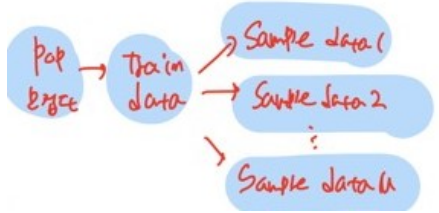
현실적으로 데이터 수집에 한계가 있음

대안으로 부트스트랩 방법 사용

◆ Bootstrap sample

모집단에서 뽑힌 하나의 train data에서 데이터 크기(n)만큼

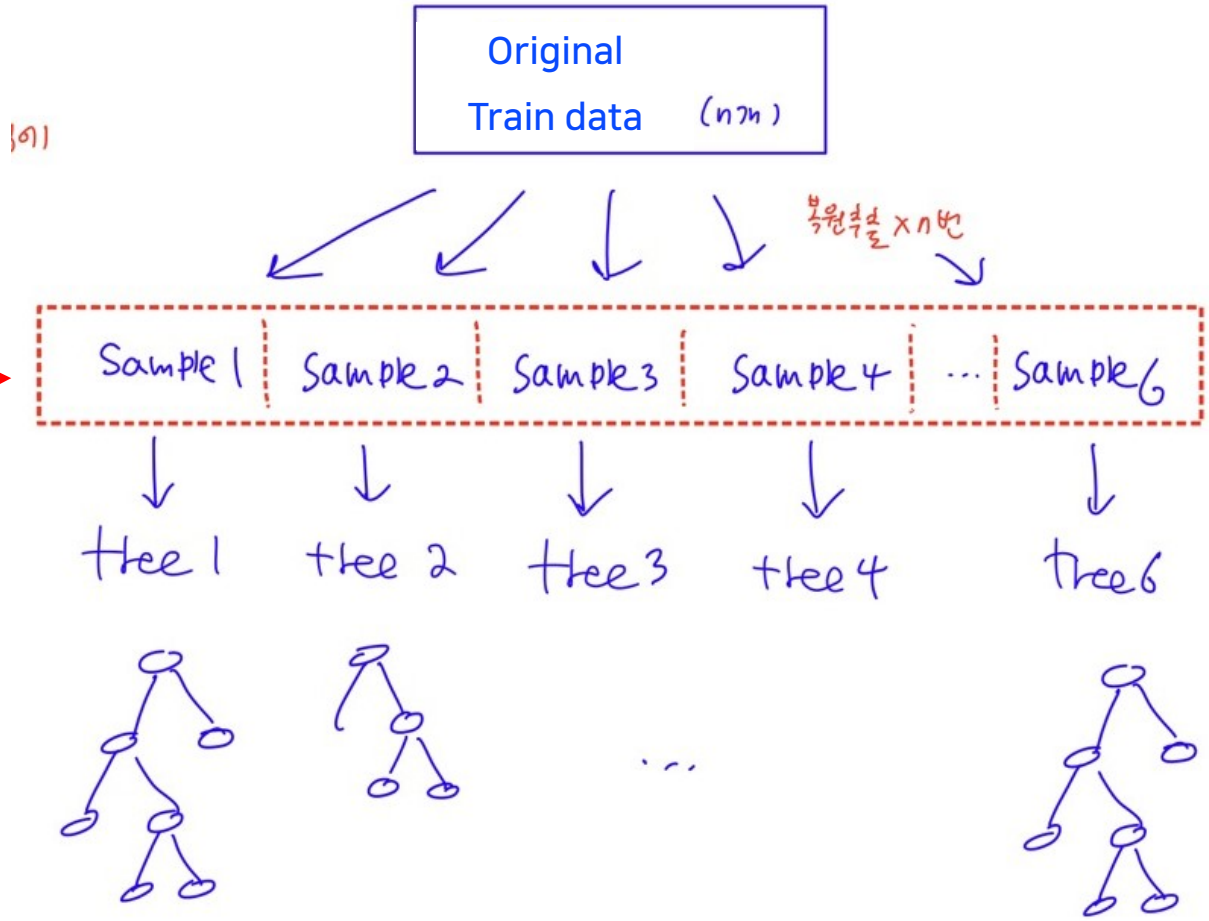
복원추출을 반복해 sample data 생성



이상적 방법과 크게 차이 x

데이터셋 크기를 늘리고 분포를 고르게 만듬

랜덤 포레스트 모델



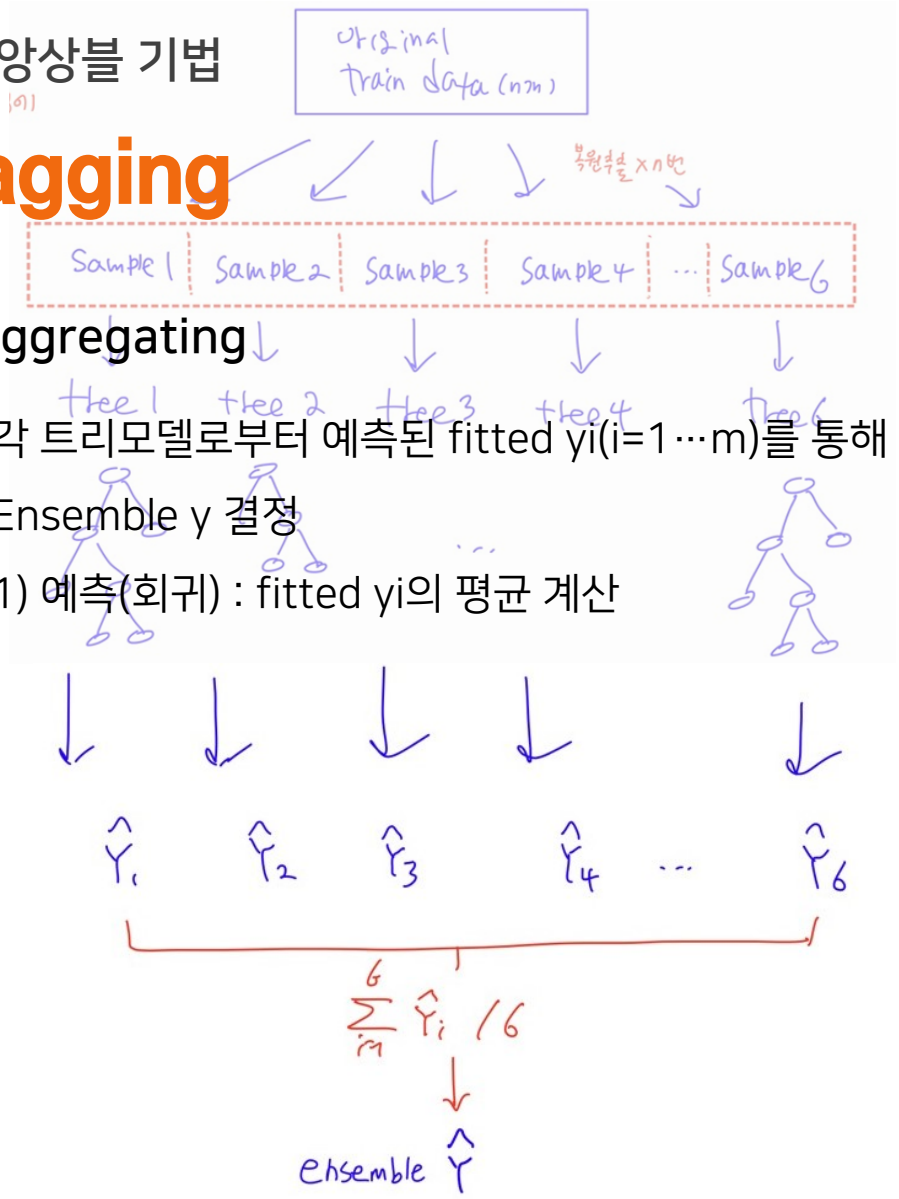
02. 앙상블 기법

Bagging

◆ Aggregating

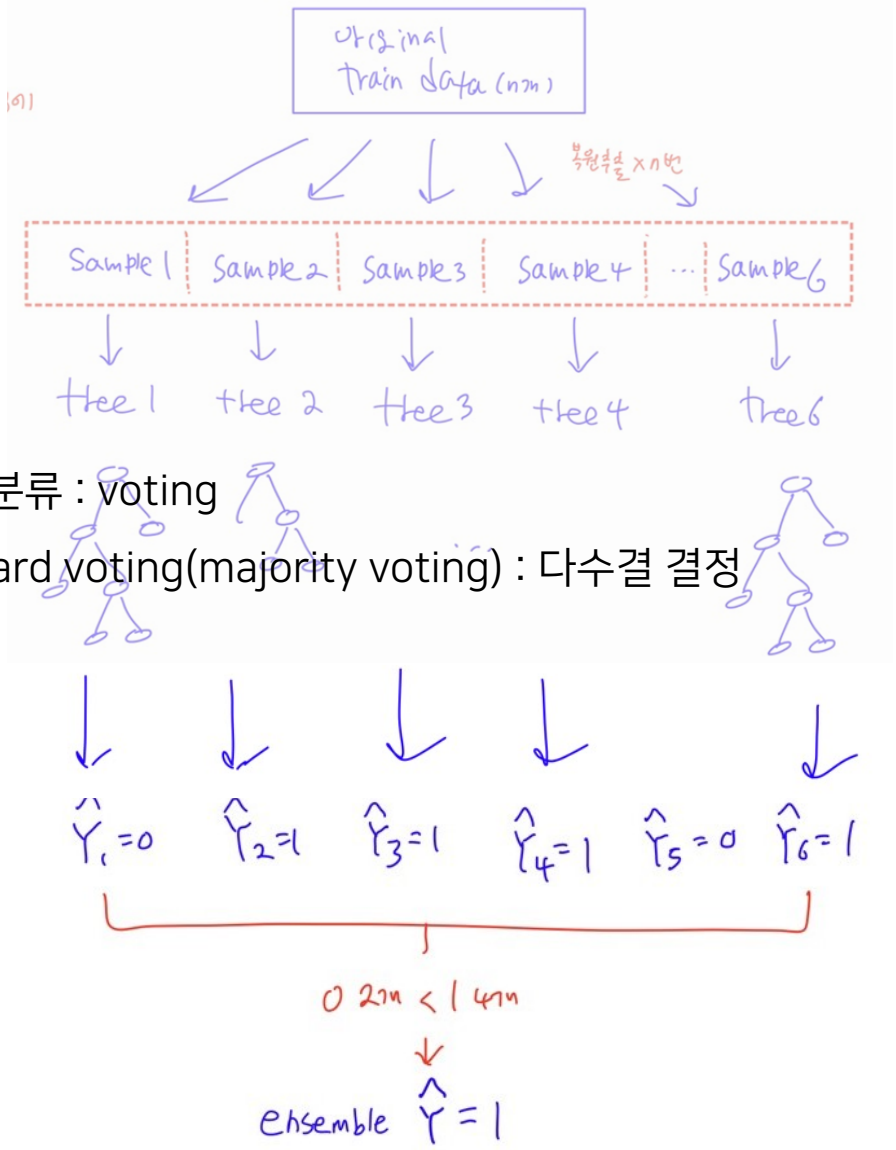
각 트리모델로부터 예측된 fitted yi(i=1…m)를 통해 하나의 Ensemble y 결정

1) 예측(회귀) : fitted yi의 평균 계산



2) 분류 : voting

- hard voting(majority voting) : 다수결 결정



02. 앙상블 기법

Bagging

◆ Aggregating

2) 분류 : voting

- Soft voting(predict_prova)

  : class 별 확률의 평균을 계산해 크기 비교

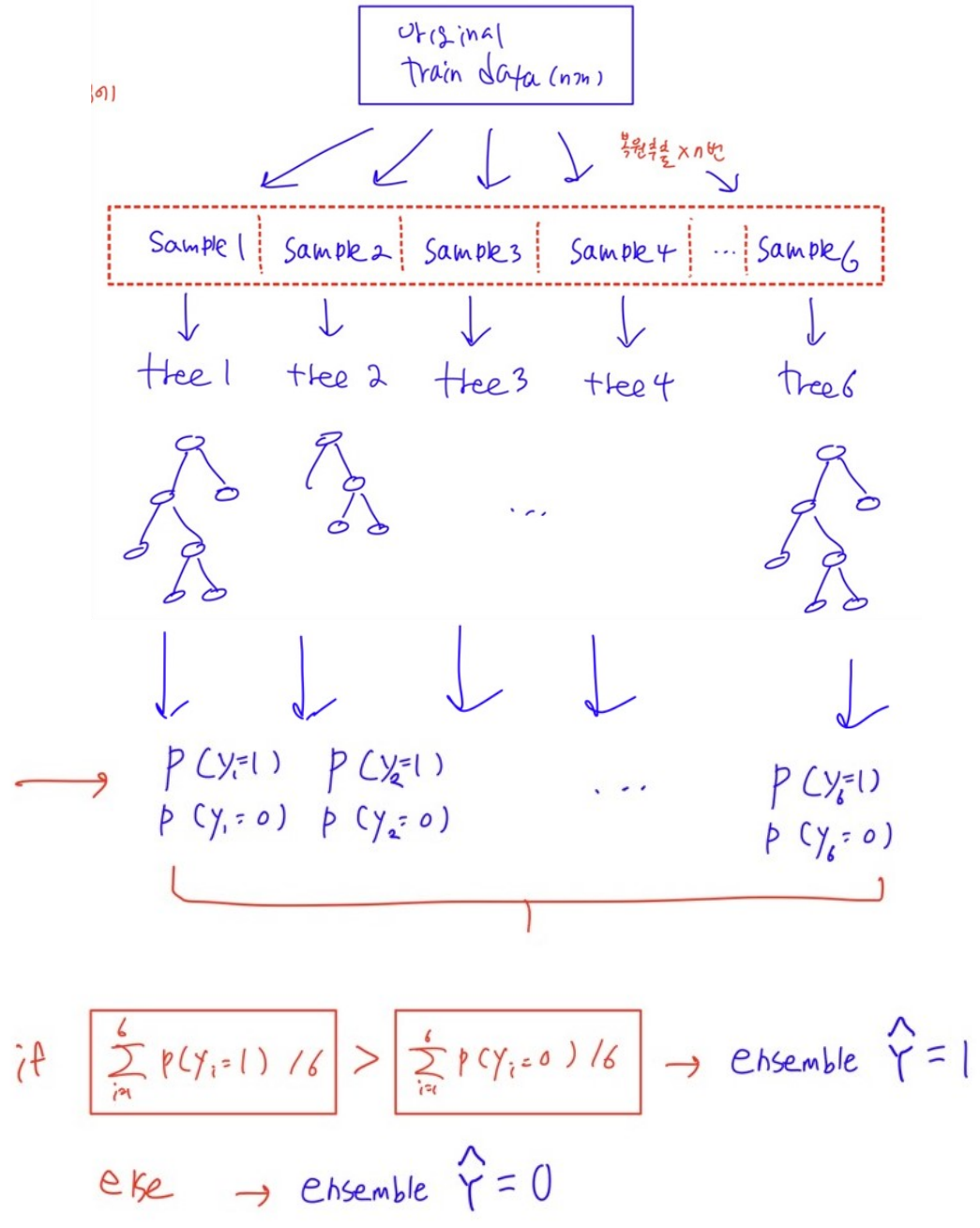
- Weighted voting (옵션)

  : 지정한 특정 array 값들로 가중평균을 취해줌

  :hard/soft votin에 옵션으로 더해질 수 있음

  Sklearn.ensemble.VotingClassifier 기준

  Ex. Traing accuracy



Bagging

◆ Random Subspace

랜덤포레스트 모델의 성능이 좋으려면 개별 base 모델이 독립

적이어야함, 이를 가능케 하는 것이 random subspace 방식

Tree 모델이 분기할때 원래 변수보다 더 적은 수의 변수를 임의로 선택해 고려함

◆ 랜덤포레스트의 하이퍼 파라미터들

(1) n_estimators

사용할 tree 수

(2) max_features

노드 분할 시 무작위로 선택되는 변수의 수

◆ 랜덤포레스트의 장단점

1) 장점

분류 및 회귀 문제에 모두 사용 가능

대용량 데이터 처리에 효과적

과대적합 문제를 해결할 수 있음

2) 단점

시간이 오래 걸림

모든 트리 모델을 다 확인하기 힘들어 모델 설명이 어려움

02. 앙상블 기법

엑스트라 트리

◆ 특성

랜덤 포레스트와 비슷하지만 부트스트랩 샘플을 사용하지 않고 전체 훈련 세트를 사용한다는 점에서 차이가 있음

노드를 무작위하게 분할하여 편향성을 증가시키고 분산을 감소시킴

◆ 장점

과대 적합을 막고 test 점수를 높일 수 있음

랜덤하게 노드를 분할해 속도가 빠름

◆ 단점

랜덤 포레스트 모델보다 성능이 낮음

일반화 성능을 높이기 위해서 많은 트리를 만들어야 함

◆ 랜덤 포레스트 vs 엑스트라 트리

랜덤 포레스트는 주어진 모든 변수에 대한 정보이득을 게산하고 이 중 가장 최선의 featur을 선택

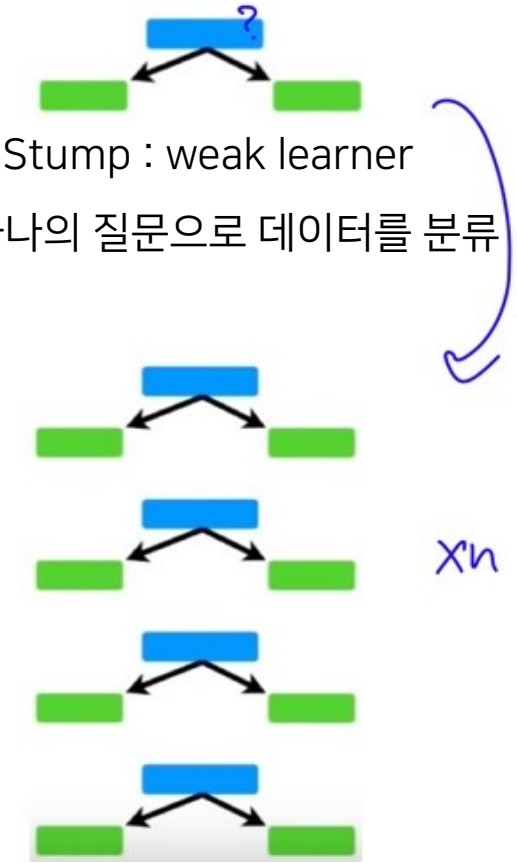
엑스트라 트리는 무작위로 feature을 선택하고 선택된 feature 중에서 최적의 분기를 고름

랜덤 포레스트 보다 성능은 낮아지지만 생각보다 준수한 성능을 보임

02. 앙상블 기법

Boosting

◆ Ada boost

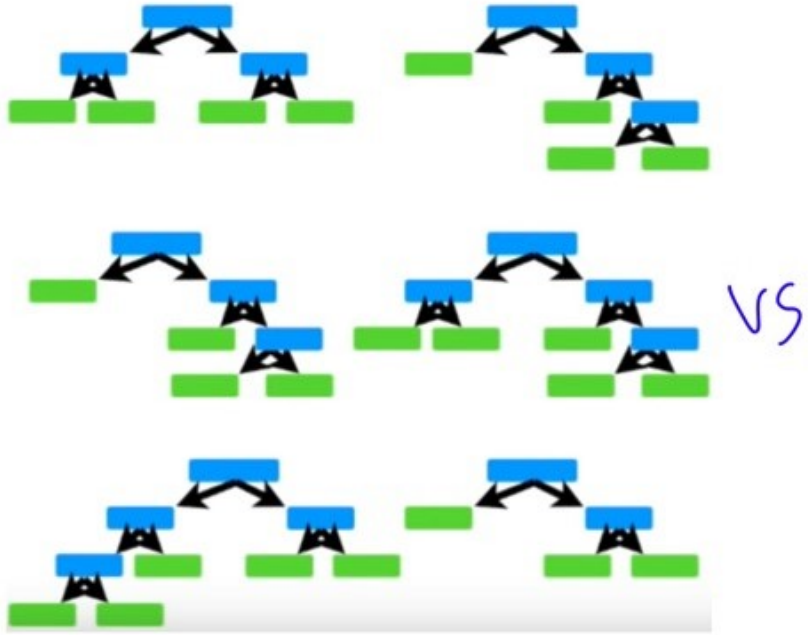


Stump : weak learner

단 하나의 질문으로 데이터를 분류

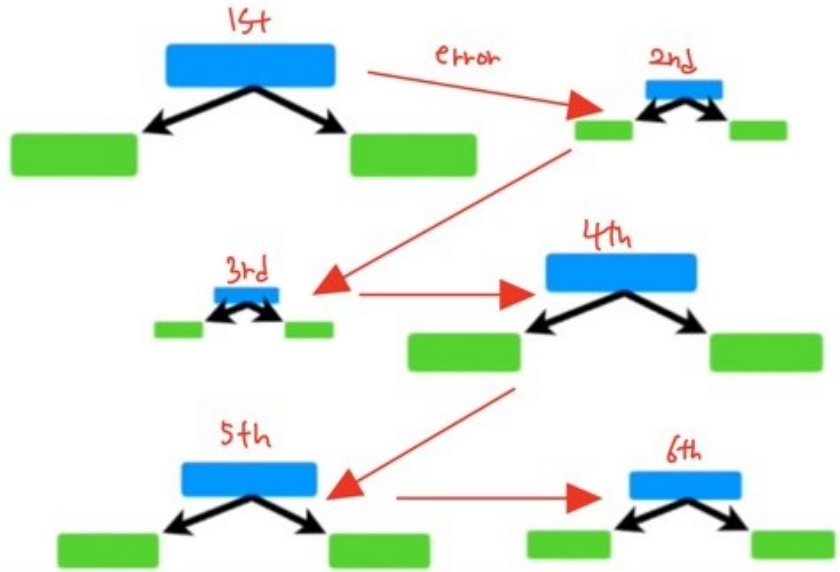
Forest of stumps

랜덤 포레스트 모델



각각의 트리가 동일한 가중치를 가짐
(민주주의 voting)

Ada boost 모델



각각의 트리가 상이한 가중치를 가짐

이전 stump에서 발생한 에러를

다음 학습에 반영

Ada boost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

지니계수가 가장 작은 weight stump를 첫 stump로 지정

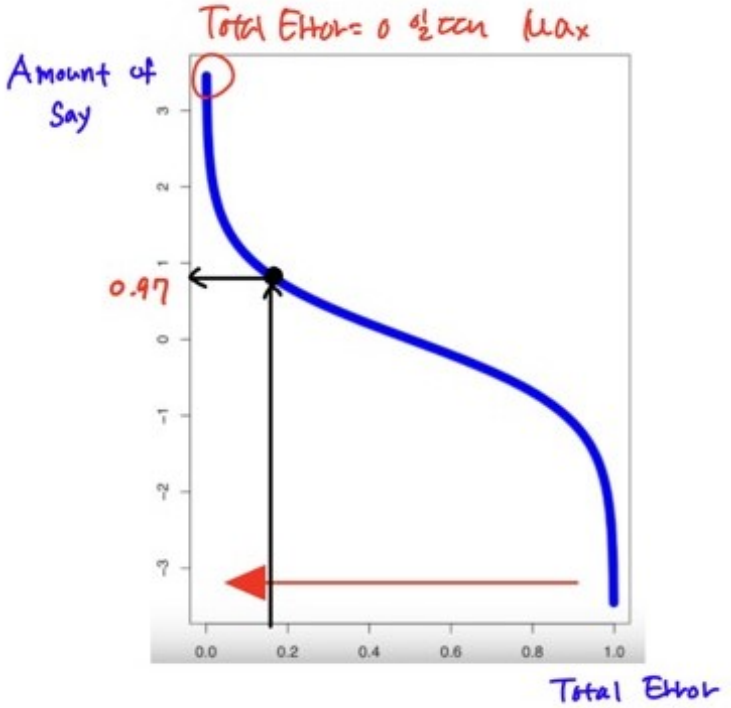
-> 오분류율이 1개(sample weight 1/8)

Total Error = 1/8

◆ amount of say

Total error= 1/8

이를 대입해 amount of say 계산



$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - 1/8}{1/8}\right) = 0.97$$

Ada boost

◆ 오분류된 데이터

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

0.33 = (1/8) × $e^{0.97}$

◆ 정분류된 데이터

$$\text{New Sample Weight} = \text{sample weight} \times e^{-\text{amount of say}}$$

0.05 = (1/8) × $e^{-0.97}$

위에서 계산된 Amount of say를 사용해 sample weight 조정

Sample Weight	New Weight
1/8	0.05
1/8	0.05
1/8	0.05
1/8	0.33
1/8	0.05
1/8	0.05
1/8	0.05
1/8	0.05
Sum	1 / 0.68

정분류

오분류

weight / 0.68

정규화

Norm. Weight
0.07
0.07
0.07
0.49
0.07
0.07
0.07
0.07

합을 1로 맞춰주기 위해 정규화 과정을 거침

Ada boost

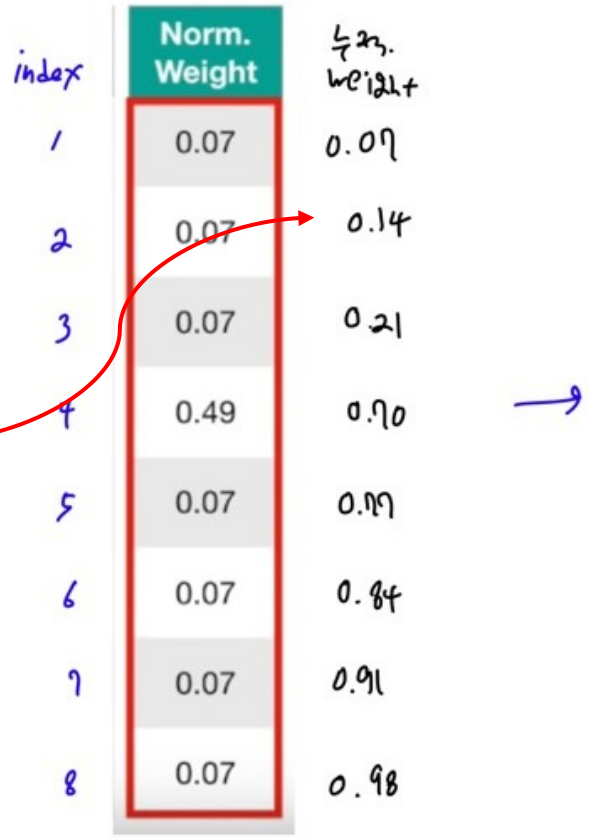
◆ Train data 리샘플링

0에서 1까지 범위 중 랜덤넘버 x를 데이터 수만큼 추출

(x1, x2, ⋯ , x8)

누적 weight 를 계산해 xi가 속하는 범위의 data를 가져옴 (중복허용)

Ex. Xi=0.1 이면 2번 데이터 추출



◆ 리샘플링된 train data

: 1st stump에서 오분류된 데이터가 4번이나 뽑힘

norm.weight가 가중치로 작용

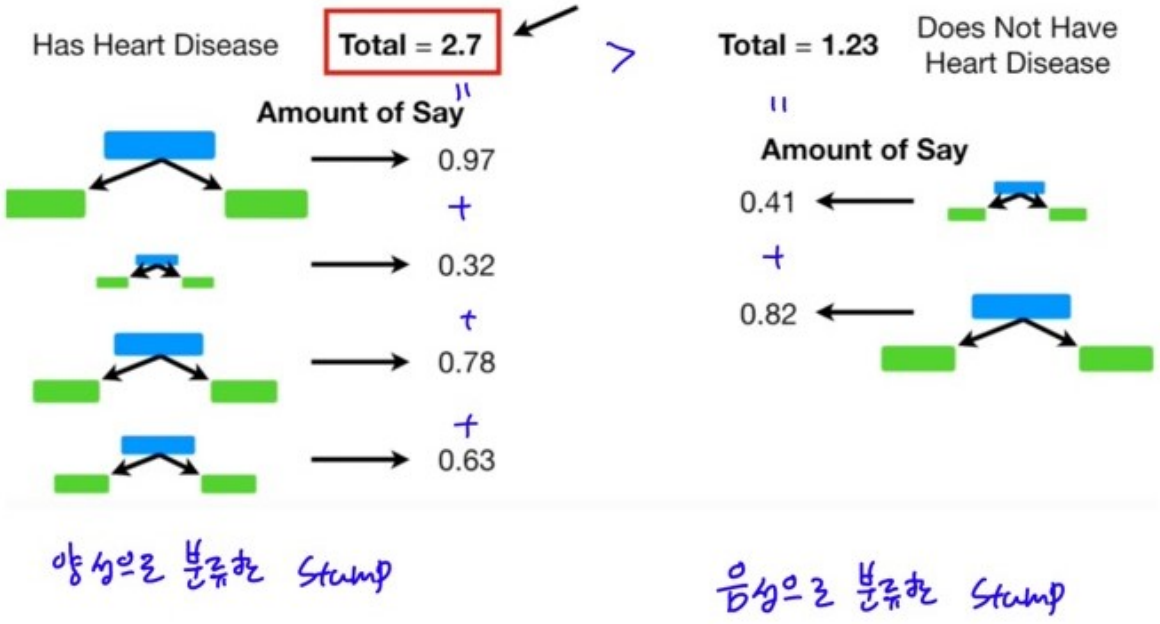
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
No	Yes	156	No	1/8
Yes	Yes	167	Yes	1/8
No	Yes	125	No	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	172	No	1/8
Yes	Yes	205	Yes	1/8
Yes	Yes	167	Yes	1/8

반복

Ada boost

◆ 최종분류

양성의 amount of say가 더 크므로
양성으로 predict 결정



양성/ 음성으로 분류한 stump의 amount of say를 각각 합쳐준다

◆ Ada boost VS Gradient boost

Stump에서 시작 단일 leaf 노드로 시작

타겟값의 초기추정값으로

주로 평균을 이용

◆ Gradient boost

Leaf(평균)과의
오차를 구함

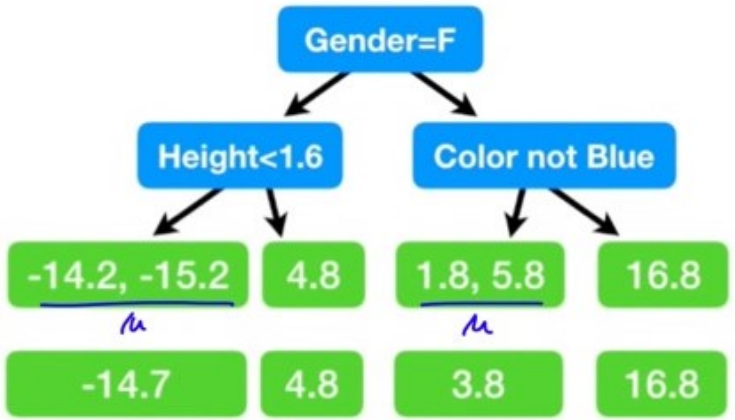
Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

오차 → Average Weight 71.2 (Single leaf)

Gradient boost

◆ Gradient node의 residual 예측 tree

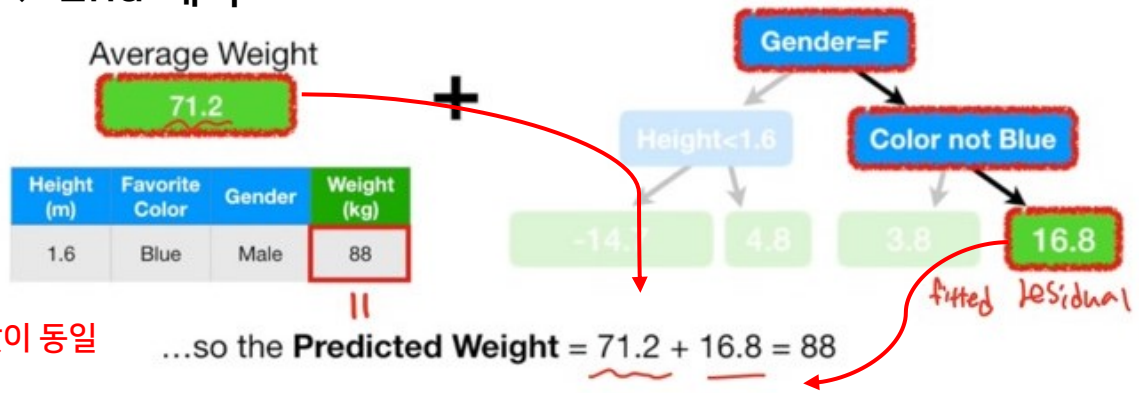
타겟에 대한 예측이 아닌 residual 예측 tree를 만들어준다



Residual 값이 두개 이상인 leaf노드의 경우 평균을 사용

◆ 1st 예측

◆ 2nd 예측



! 예측값과 원 데이터값이 동일

-> 과적합 문제 발생

단일 leaf 노드와 residual tree를 이용해 몸무게를 예측한다

과적합을 피하기 위해 learning rate을 설정해줌

1st 예측보다 실제 값과 가까워짐

Gradient boost

◆ New residual

2nd예측값으로 residual 다시 계산 88-72.9=15.1 kg

new

Residual	Residual
16.8	15.1
4.8	4.3
-15.2	-13.7
1.8	1.4
5.8	5.4
-14.2	-12.7

new prediction

new tree

◆ 3rd 예측 수행

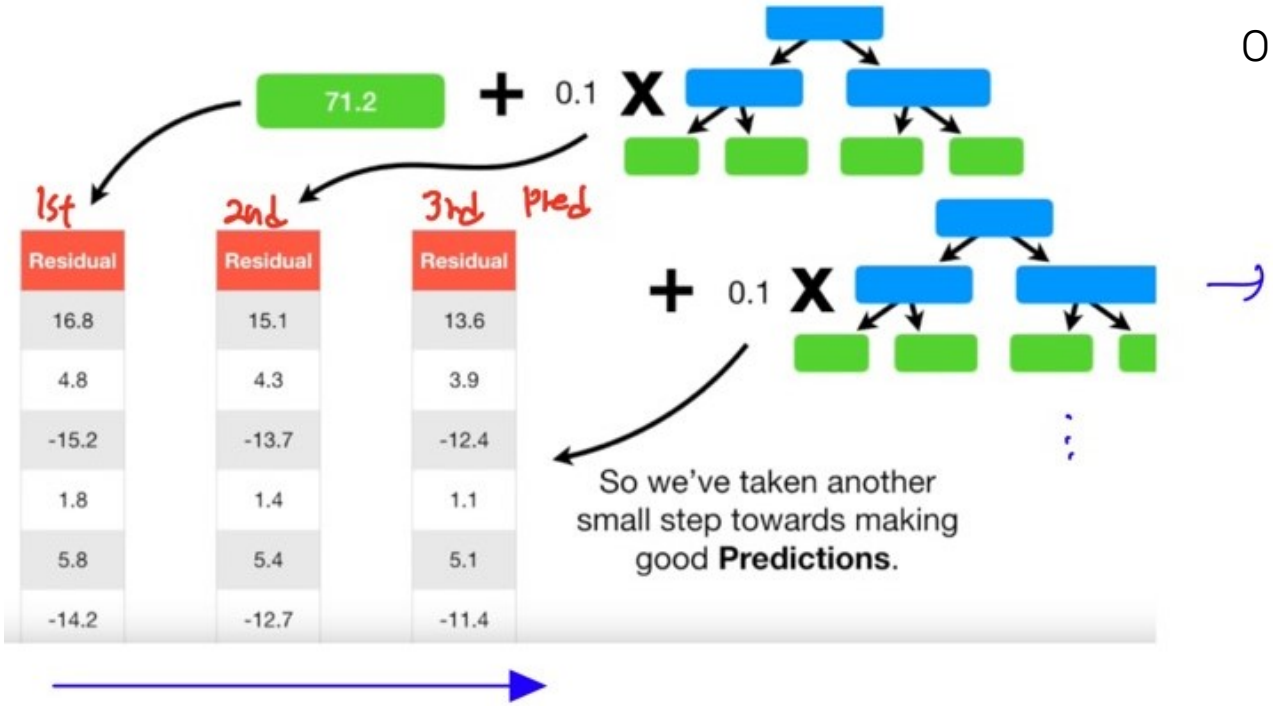
learning rate

$$71.2 + (0.1 \times 16.8) + (0.1 \times 15.1)$$

Leaf 노드(평균) 1nd resd 2nd resd

Gradient boost

◆ 3rd residual 계산



Residual이 점차 감소함:

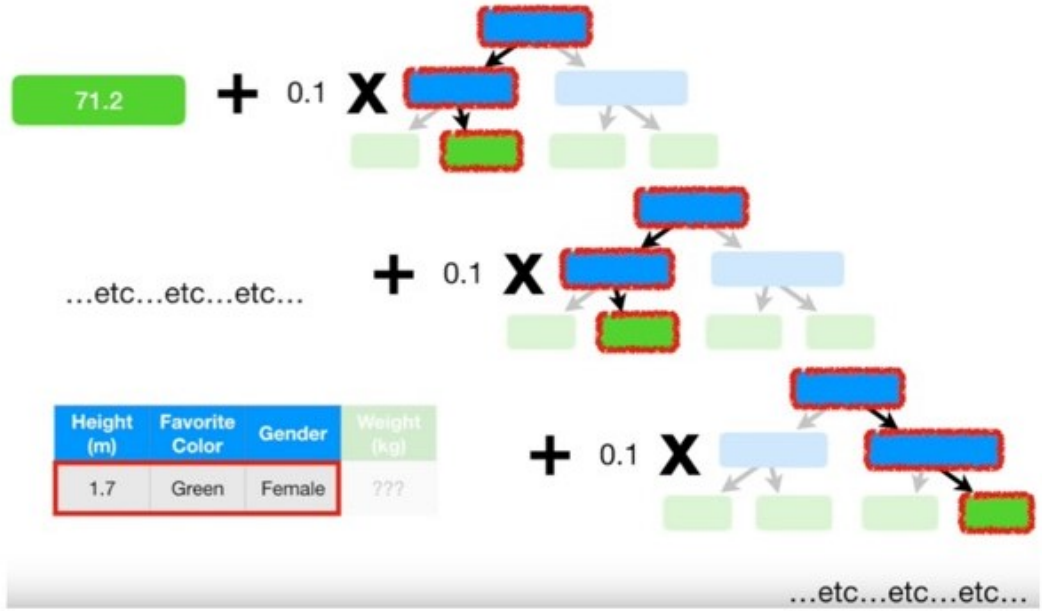
실제 값과 점점 동일해짐, 모델이 발전함을 의미

◆ 반복

언제까지?

Hyper parameter 인 iteration(반복수)에 도달할때까지

Or residual이 더 이상 줄어들지 않을 때까지



감사합니다