DATA
SCIENCE LAB

Decision Tree & Ensemble

23.07.20 / 9기 신소연

CONTENTS

DATA
SCIENCE LAB

Decision Tree

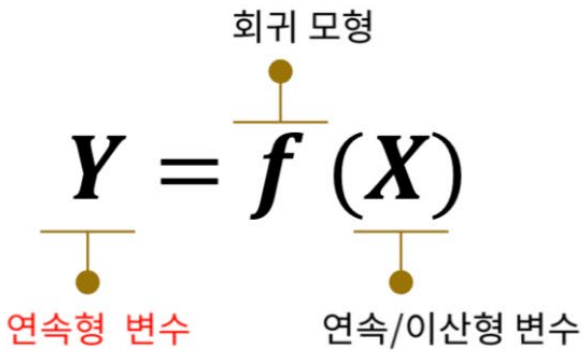
0. INTRO

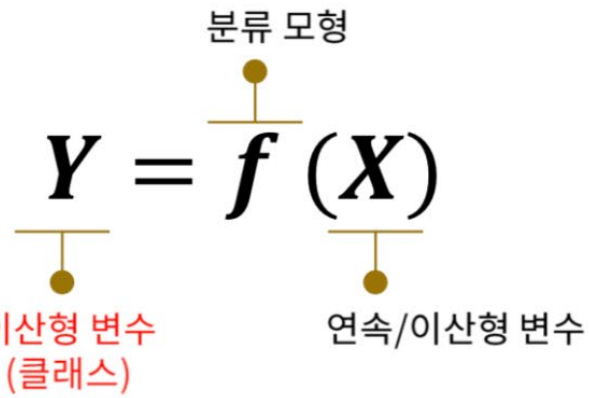
Decision Tree

지도학습의 종류 : 분류(Classification) / 회귀(Regression)

- Decision Tree는 Rule Based Classifier : 기본적으로 Classification Task를 위한 방법론
- Regression Tree로 확장 가능

=> Classification, Regression 모두 가능한 지도학습 모델



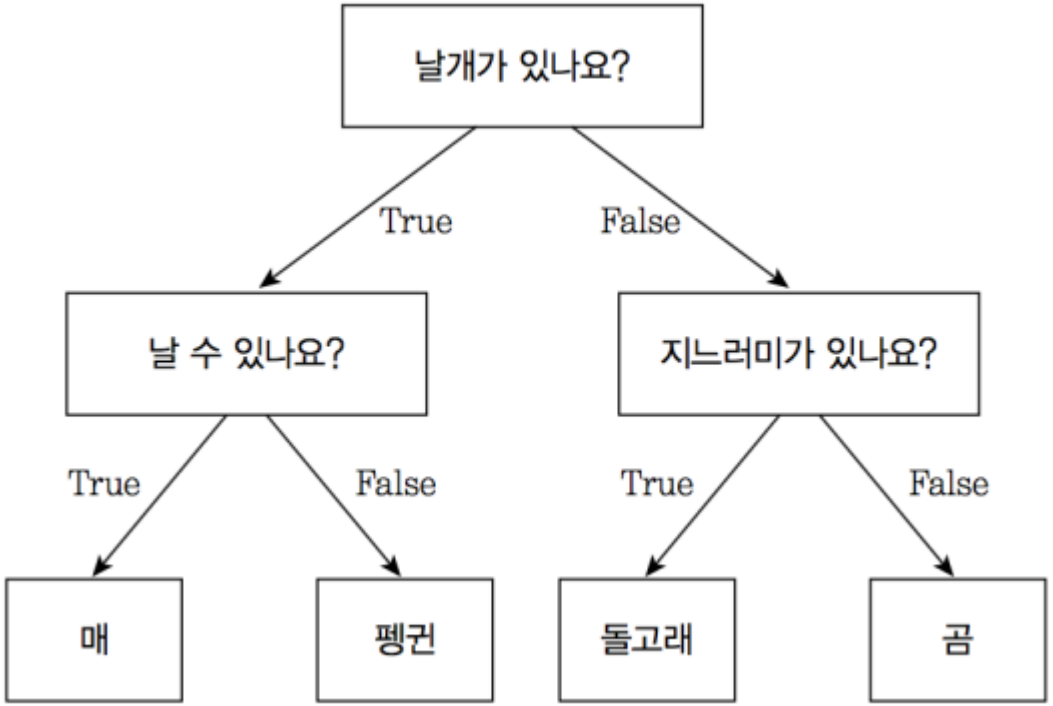


1. Decision Tree

Decision Tree란?

: 일련의 질문에 대한 답에 근거하여 데이터를 분류하는 모델

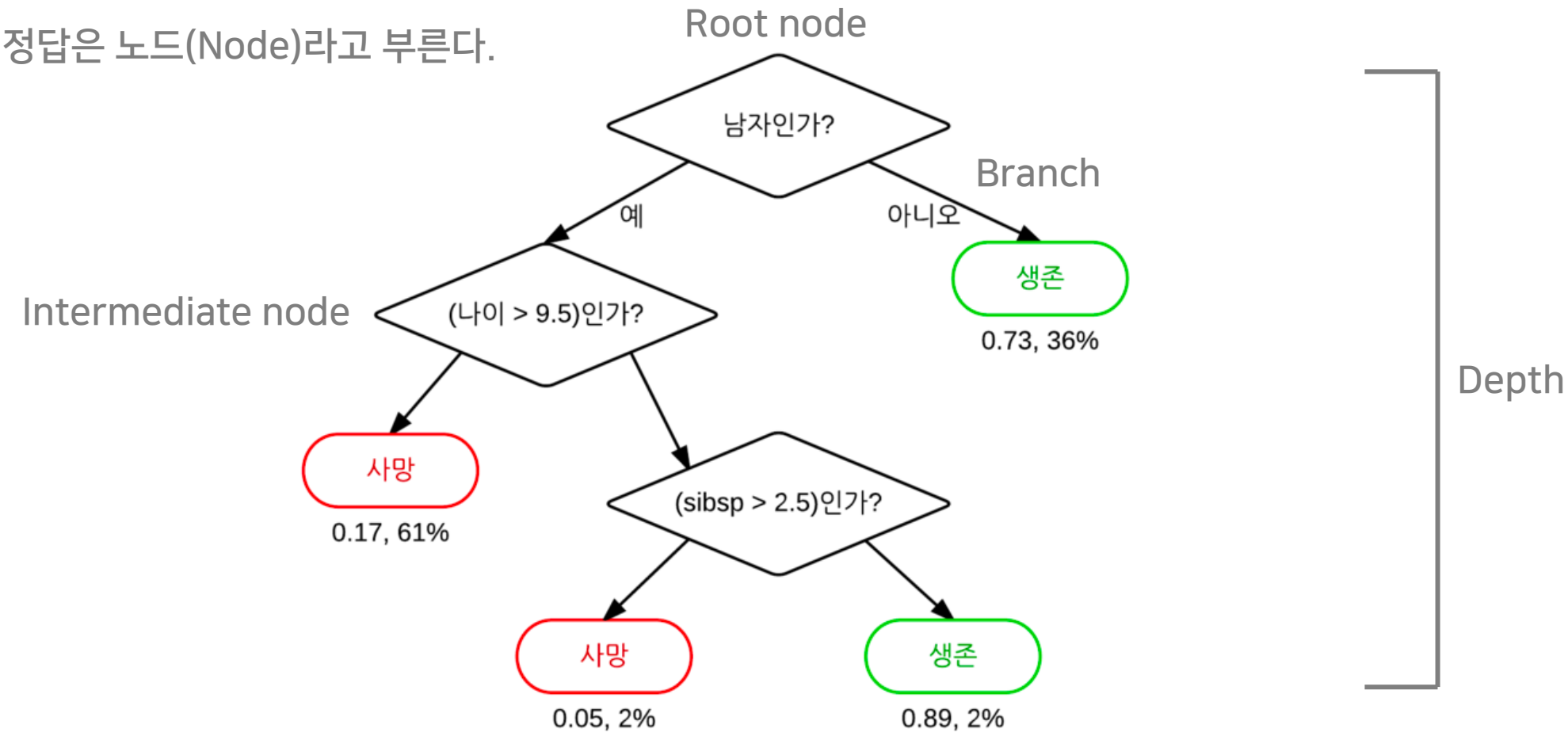
- 한 번의 분기마다 변수 영역을 두 개로 구분
- 질문 = 속성 = Attribute
- Human interpretable : 높은 표현력
- 필터링/스무고개의 원리



1. Decision Tree

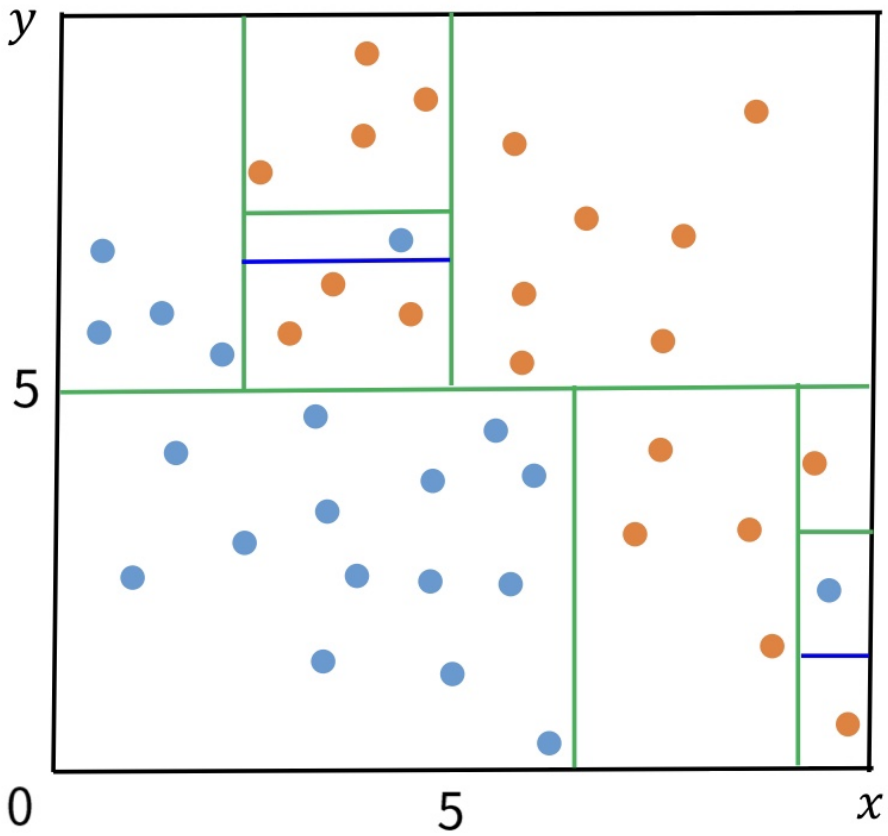
구성

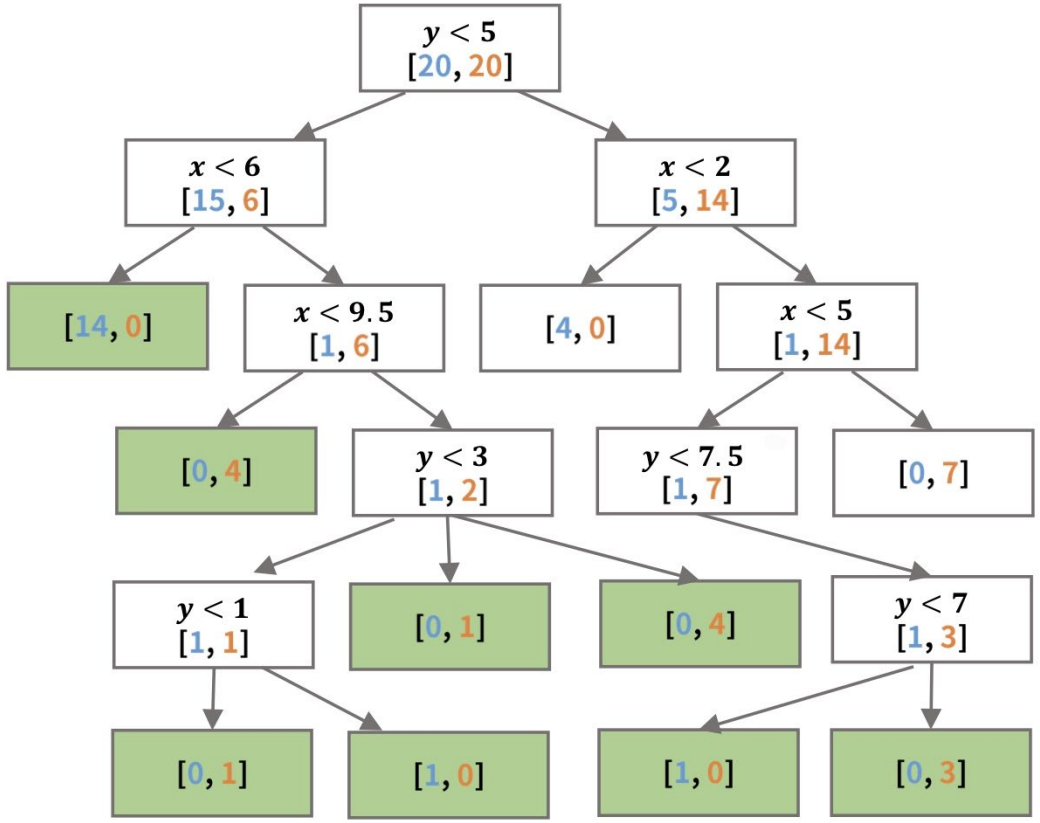
- 질문이나 정답은 노드(Node)라고 부른다.



1. Decision Tree

Overfitting





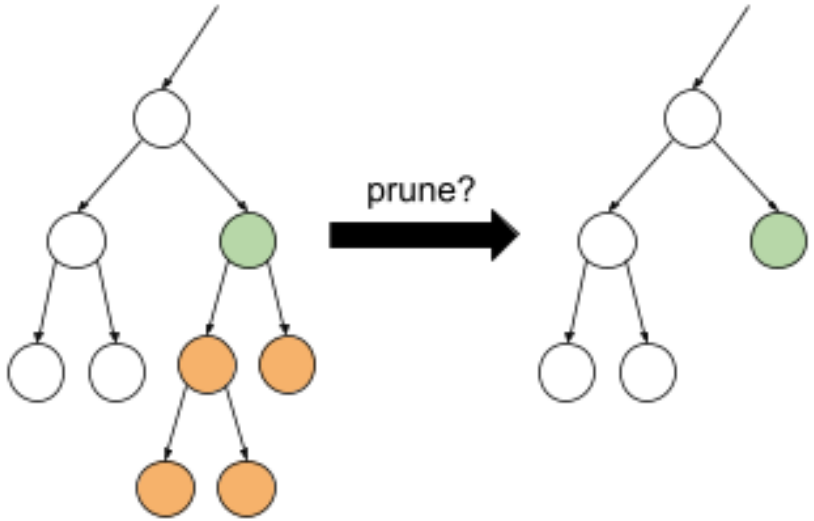
=> Overfitting!

1. Decision Tree

가지치기 (Pruning)

트리에 가지가 너무 많으면 Overfitting 문제 발생 – 해결 방법!

- 사전 가지치기: 나무가 완성되기 전에 **특정 조건을 만족하면** 알고리즘 중단
  - max_depth, min_sample_split
  - ex. 한 노드에 들어있는 데이터가 10개 이하면 해당 노드는 더 이상 분기를 하지 않는다.
- 사후 가지치기: 나무가 완성된 후 하단 노드부터 유의미하지 않은 sub tree를 끝 노드로 변환



1. Decision Tree

사후 가지치기

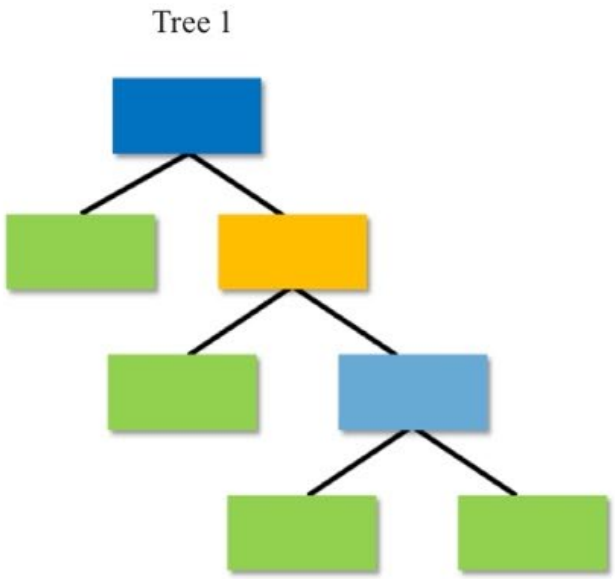
비용-복잡도 가지치기(Cost-Complexity Pruning)을 수행하며 아래 함수를 최소화하는 방향

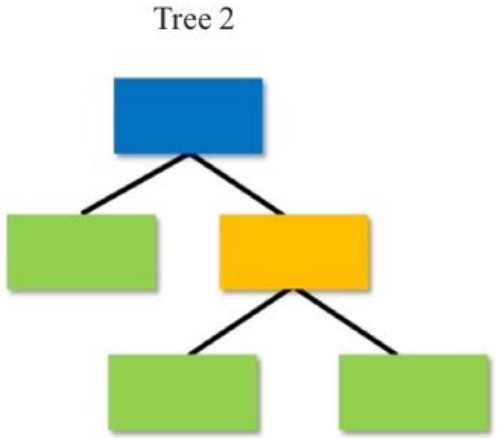
$$CC(T) = Err(T) + \alpha * L(T)$$

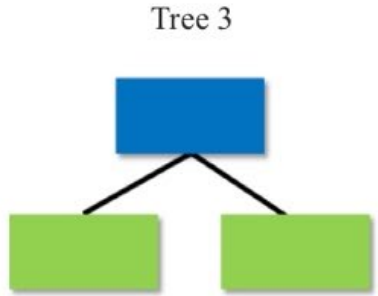
- CC(T) : 나무 모델의 비용-복잡도 지표
- Err(T) : 검증 데이터에 대한 오분류율
- L(T) : Leaf Node의 수 (= 구조 복잡도)
- α ($\alpha > 0$) : Err(T)와 L(T)를 결합하는 가중치(= Complexity Parameter)
  - 나무 복잡도의 비중을 나타내는 하이퍼 파라미터
  - 값이 커질수록 간결한 모델이 된다.

1. Decision Tree

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





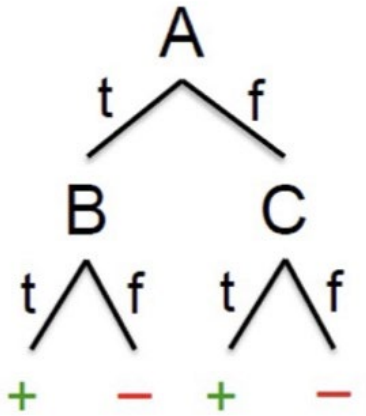


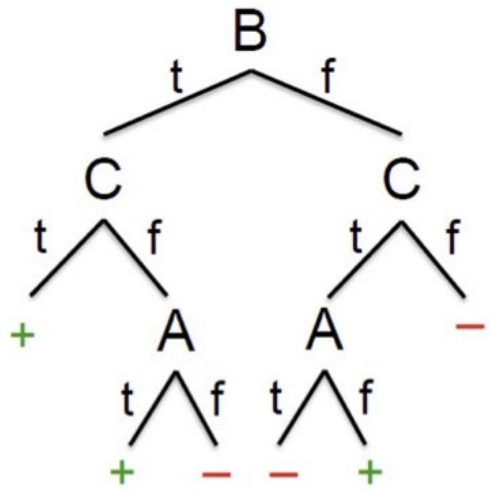
	Tree 1	Tree 2	Tree 3
	Err(T): 0.15 L(T): 4	Err(T): 0.13 L(T): 3	Err(T): 0.18 L(T): 2
① α = 0	CC(T) = 0.15 + 0 * 4 = 0.15	CC(T) = 0.13	CC(T) = 0.18
② α = 0.1	CC(T) = 0.15 + 0.1 * 4 = 0.55	CC(T) = 0.43	CC(T) = 0.38

1. Decision Tree

Best Decision Tree?

- 한 데이터를 분류하기 위해 나올 수 있는 트리의 개수는 지수함수적으로 많다.





Which tree do we prefer?

- 각 attribute에 의해 분할된 영역에, 동일한 클래스 데이터가 최대한 많이 존재하도록 기준 설정
- **"변별력이 좋은 질문"** 선정 필요

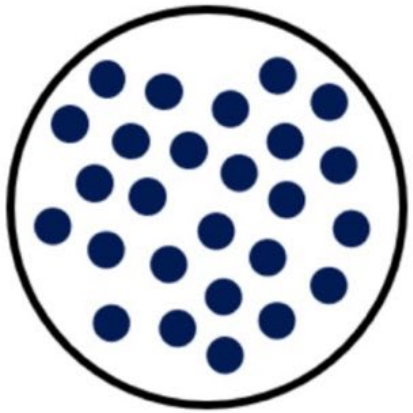
1. Decision Tree

불순도 (Impurity)

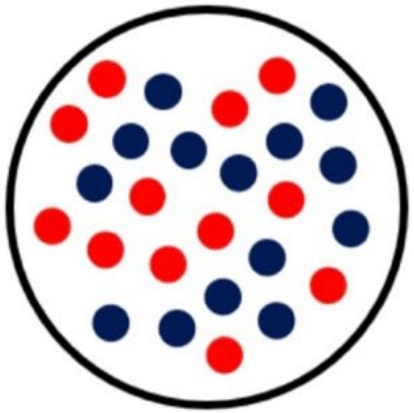
: 해당 범주 안에 서로 다른 데이터가 얼마나 섞여 있는가

Decision Tree는 **불순도를 최소화하는 방향**으로 학습을 진행한다.

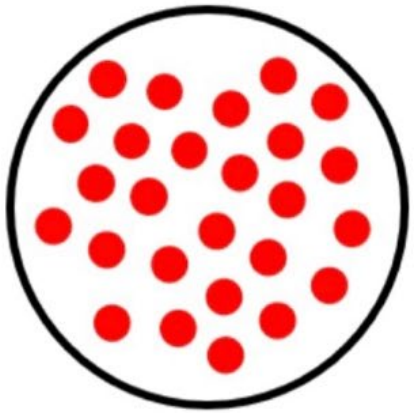
불순도를 측정하는 지표: Entropy, Information Gain, Gini Index



불순도 = 0



불순도 > 0



불순도 = 0

2. DT Algorithm

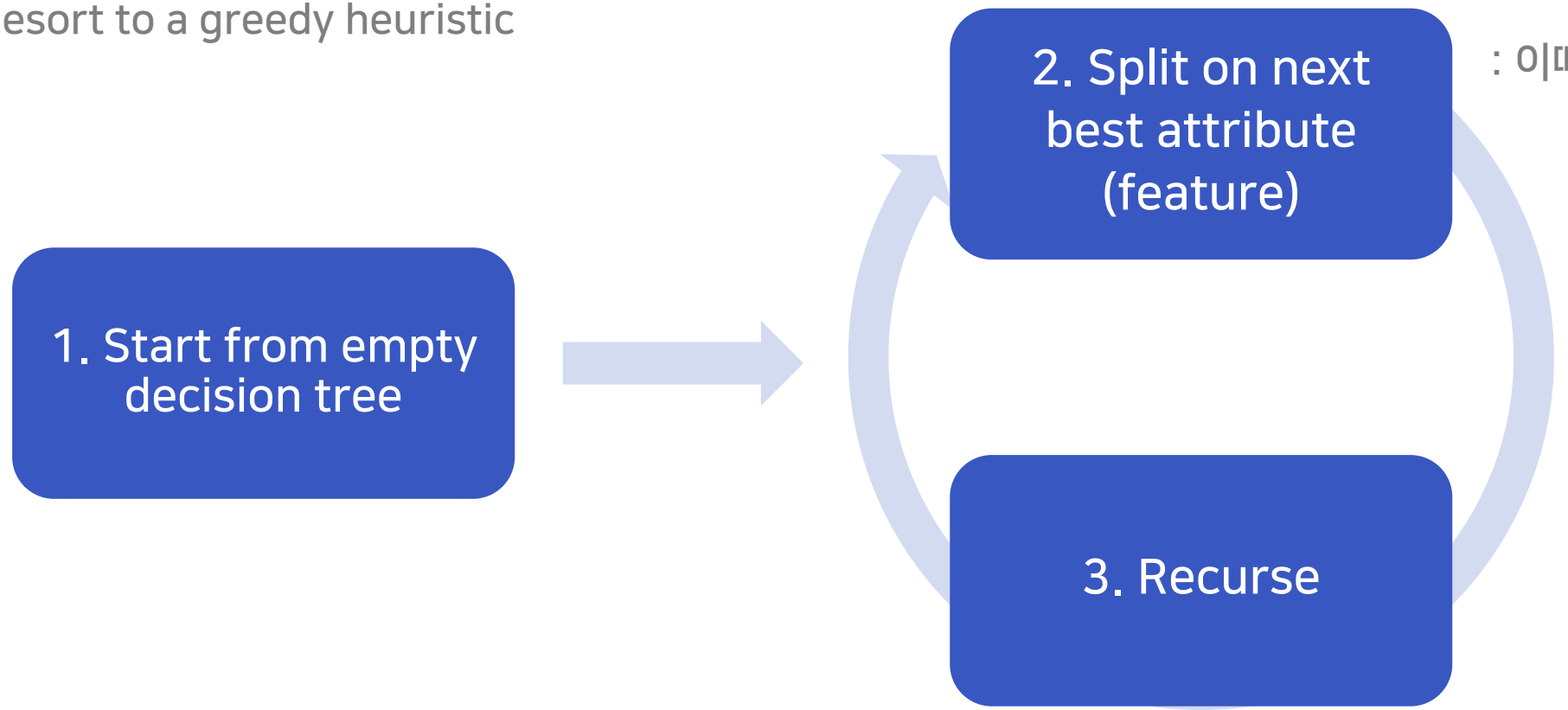
알고리즘 비교

알고리즘	예측변수	불순도 평가지수	분리
CHAID	범주형, 연속형	Chi-square(범주) F검정(연속)	Multiple Split
ID3	범주형	Entropy Information Gain	Multiple Split
C4.5, C5.0	범주형, 연속형	Entropy Information Gain Information Gain Ratio	Multiple Split
CART	범주형, 연속형	Gini Index(범주) 분산의 감소량(연속)	Binary Split

2. DT Algorithm

Algorithm Process

Resort to a greedy heuristic



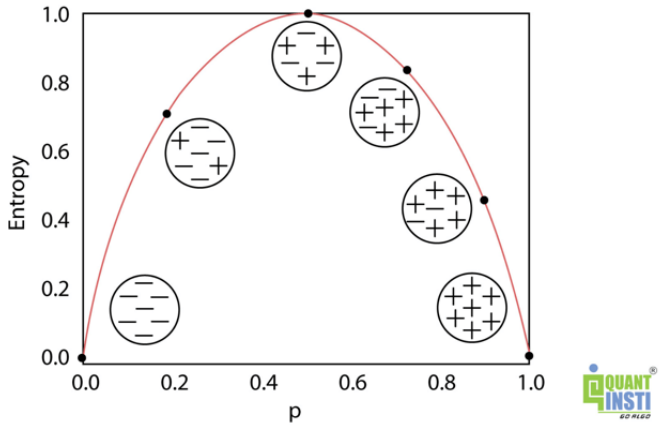
: 이때 불순도 평가지수를 사용

2. DT Algorithm

Entropy

: 불확실성(Uncertainty)

- High Entropy = High Impurity
- MAX = 1 , MIN = 0



High Entropy

- Y from a uniform like distribution
- Values less predictable

Low Entropy

- Y from a varied (peaks and valleys) distribution
- Values more predictable

$$Entropy = H(S) = \sum_{i=1}^{c} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{c} p_i \log_2 p_i$$

2. DT Algorithm

Entropy

$$H(S) = \sum_{i=1}^{c} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{c} p_i \log_2 p_i$$

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$$P(Y = T) = 5/6$$

$$P(Y = F) = 1/6$$

$$H(Y) = -5/6 \log_2 5/6 \; - 1/6 \log_2 1/6$$

$$= 0.65$$

2. DT Algorithm

Conditional Entropy

$$\mathrm{H}(Y|X) \equiv \sum_{x \in \mathcal{X}} p(x)\,\mathrm{H}(Y|X=x)$$
$$= -\sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x)\,\log_2\,p(y|x)$$

-> Attribute X_1이 좋을지 X_2이 좋을지 결정 가능!

ex.

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

P(X_1 = T) = 4/6

P(X_1 = F) = 2/6

H(Y |X_1) = -4/6(1log2(1) + 0log2(0)) – 2/6(1/2log2(1/2) + 1/2log2(1/2)) = 0.33

P(X_2 = T) = 3/6

P(X_2 = F) = 3/6

H(Y |X_2) = -3/6(1log2(1) + 0log2(0)) – 3/6(2/3log2(2/3) + 1/3log2(1/3)) = 0.46

2. DT Algorithm

Information Gain

: 분기 전후 Entropy의 변화량(감소량)

$$IG(\mathrm{X}) = \mathrm{H}(\mathrm{Y}) - \mathrm{H}(\mathrm{Y}|\mathrm{X})$$

X_1 의 경우: 0.65 - 0.33 = 0.32

- IG(X) > 0 이어야 해당 분기가 유의미하다.
- IG(X)가 **가장 큰** attribute를 선택해야 한다.
- Entropy를 Gini Index로 대체 가능

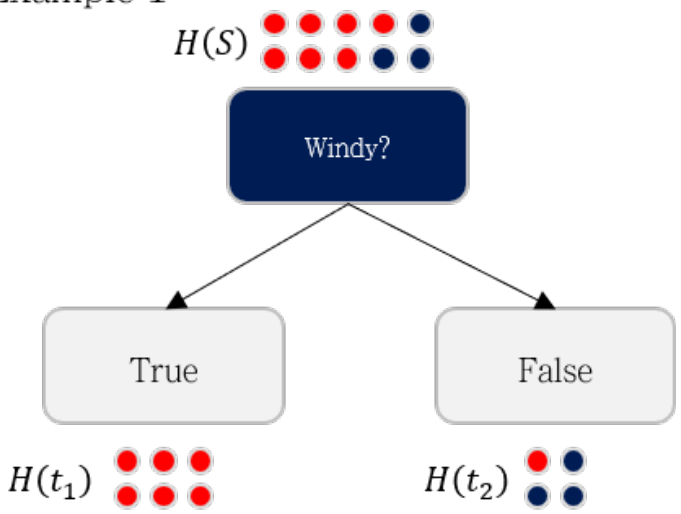
<ID3 적용>

2. Split on next best attribute (feature)
   - ex. Using information gain : $arg\max_i IG(X_i) = arg\max_i H(Y) - H(Y|X_i)$

2. DT Algorithm

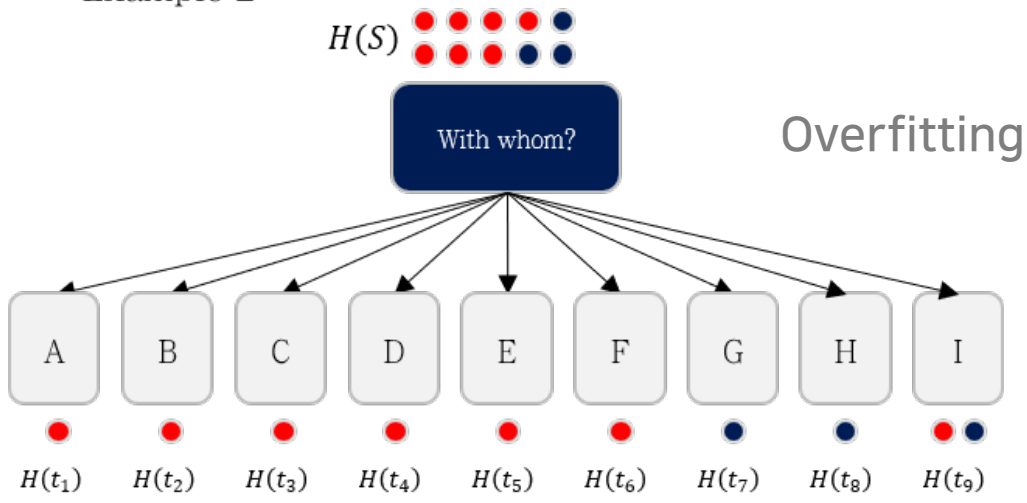
Information Gain의 한계점

Example 1



$$Information\ Gain = H(7,3) - (\frac{6}{10}H(6,0) + \frac{4}{10}H(1,3))$$
$$= 0.8813 - (\frac{6}{10} \times 0 + \frac{4}{10} \times 0.8113)$$
$$= 0.5568$$

Example 2



Overfitting?

$$Information\ Gain = H(7,3) - (\frac{1}{10}H(1,0) + \frac{1}{10}H(1,0) + \ldots + \frac{1}{10}H(0,1) + \frac{2}{10}H(1,1))$$
$$= 0.8813 - (\frac{1}{10} \times 0 + \ldots + \frac{2}{10} \times 1)$$
$$= 0.6813$$

https://tyami.github.io/machine%20learning/decision-tree-3-c4_5/

2. DT Algorithm

Information Gain Ratio

: 정보이득률

Information Gain: split을 더 많이 할수록 값이 커지는 한계 (실제로는 Overfitting 때문에 성능이 더 낮아짐)

-> 이를 보완하기 위해 Information Gain Ratio 개념 도입

$$Information\ Gain\ Ratio = \frac{IG}{IV}$$

$$Intrinsic\ Value = IV = -\sum_{i}^{N} p_i log_2 p_i$$

N : 특정 지표로 분기했을 때 생성되는 가지의 수

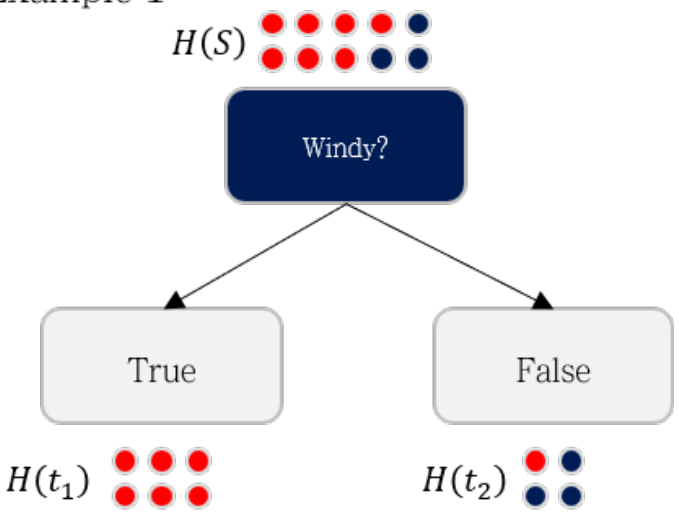
p_i: i번째 가지에 해당하는 확률

https://tyami.github.io/machine%20learning/decision-tree-3-c4_5/

2. DT Algorithm

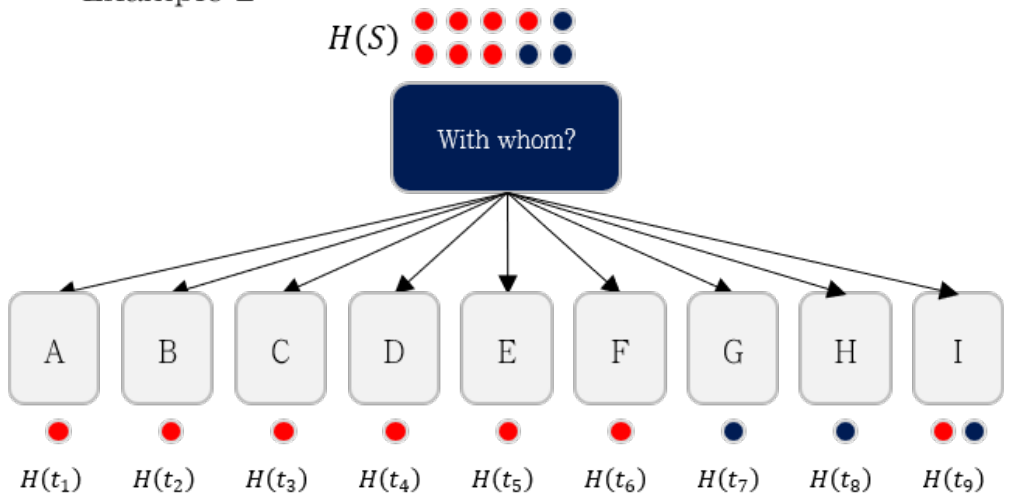
Information Gain Ratio

Example 1



$$Information\ Gain\ Ratio = IG/IV$$
$$= \frac{0.5568}{-(\frac{6}{10}log_2\frac{6}{10} + \frac{4}{10}log_2\frac{4}{10})}$$
$$= \frac{0.5568}{0.9701}$$
$$= 0.5739$$

Example 2



$$Information\ Gain\ Ratio = IG/IV$$
$$= \frac{0.6813}{3.1219}$$
$$= 0.2182$$

https://tyami.github.io/machine%20learning/decision-tree-3-c4_5/

2. DT Algorithm

Gini Index (= Gini Impurity)

- 두 개의 데이터를 랜덤하게 골랐을 때 **두 데이터의 클래스가 서로 다를 확률**을 의미한다.
- High Gini Index = High Impurity
- 출력변수의 클래스가 c개 존재할 때:

$$Gini(S) = \sum_{i=1}^{c} \sum_{i' \neq i} p_i p_{i'} = \sum_{i=1}^{c} p_i \sum_{i' \neq i} p_{i'} = \sum_{i=1}^{c} p_i(1 - p_i)$$

$$= 1 - \sum_{i=1}^{c} p_i^2$$

- 모든 조합에 대해 Gini Index를 계산한 후, 가장 낮은 지표를 찾아 분기한다.
- MAX = 1 – 1/n , MIN = 0

2. DT Algorithm

Gini Index (= Gini Impurity)

ex. 이등 분할

- 분리 전 부모 노드의 Gini Index

Class A	Class B	Gini Index
0	10	0
1	9	0.18
2	8	0.32
3	7	0.42
4	6	0.48

$1-\{(\frac{0}{10})^2+(\frac{10}{10})^2\}=0$

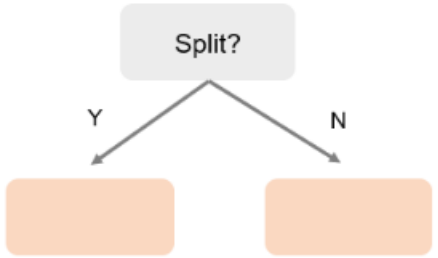
$1-\{(\frac{1}{10})^2+(\frac{9}{10})^2\}=0.18$

$1-\{(\frac{2}{10})^2+(\frac{8}{10})^2\}=0.32$

$1-\{(\frac{3}{10})^2+(\frac{7}{10})^2\}=0.42$

$1-\{(\frac{4}{10})^2+(\frac{6}{10})^2\}=0.48$

- 분리 후 자식 노드의 Gini Index



Class A	Class B	Gini Index
5	5	0.5

$1-\{(\frac{5}{10})^2+(\frac{5}{10})^2\}=0.5$

Class A	Class B	Gini Index
2	3	0.48
5	0	0

$1-\{(\frac{2}{5})^2+(\frac{3}{5})^2\}=0.48$

$1-\{(\frac{5}{5})^2+(\frac{0}{5})^2\}=0$

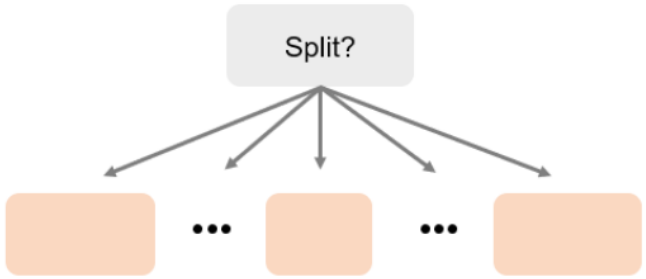
$\text{Gini(Children)} = 0.48 * (\frac{5}{10}) + 0 * (\frac{5}{10}) = 0.24$

https://lucy-the-marketer.kr/ko/growth/decision-tree-and-impurity/

2. DT Algorithm

Gini Index (= Gini Impurity)

ex. 다중 분할



annoyed	Y	Y	N	N	Y	Y
	-5°C	0°C	10°C	18°C	28°C	35°C
Split position	-7°C	-2.5°C	5°C	14°C	23°C	31.5°C
	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	2 1	3 0	3 0
No	0 7	2 5	3 4	3 4	3 4	5 2
Gini	0.42	0.375	0.343	0.400	0.300	0.375

$1-\{(\frac{0}{2})^2+(\frac{2}{2})^2\}=0$ $1-\{(\frac{3}{8})^2+(\frac{5}{8})^2\}=0.47$

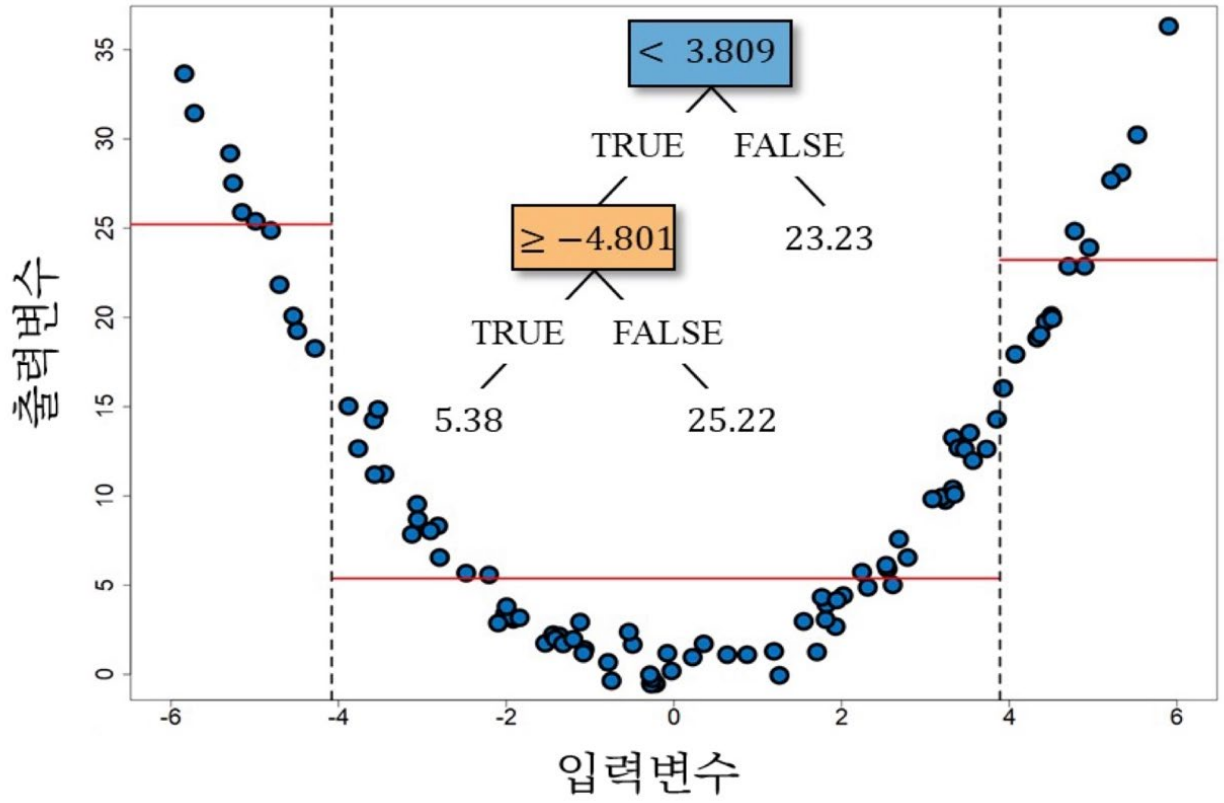
$$\text{Gini(Children)} = 0*(\frac{(0+2)}{(0+2+3+5)}) + 0.47*(\frac{(3+5)}{(0+2+3+5)}) = 0.375$$

https://lucy-the-marketer.kr/ko/growth/decision-tree-and-impurity/

2. DT Algorithm

Regression Tree

- 분기 지표를 선택하는 기준으로 불순도 대신 **분산의 감소량**을 사용한다.



$$Residual\ Sum\ of\ Squares\ (RSS) = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

실제값과 예측값의 오차(=분산)을 사용

DATA
SCIENCE LAB

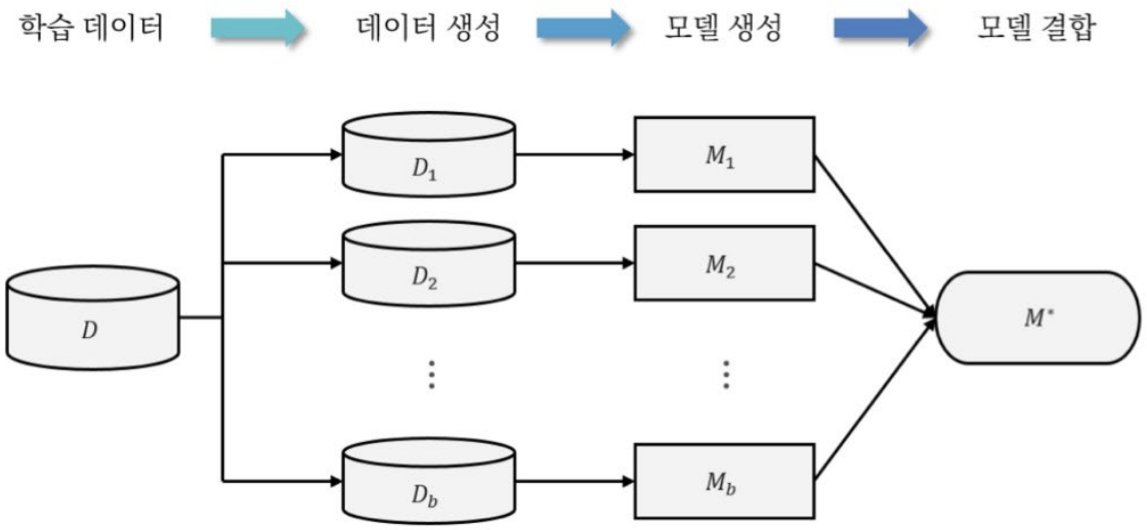
Ensemble

3. Ensemble

Ensemble 기법이란?

: 다수의 학습 모델을 **결합**하여 최적의 결과를 찾아내는 기법

- Decision Tree, SVM, Deep Learning 등 모든 종류의 학습 모델이 사용될 수 있다.
- 대표적인 앙상블 기법: Voting, Bagging, Boosting, Stacking



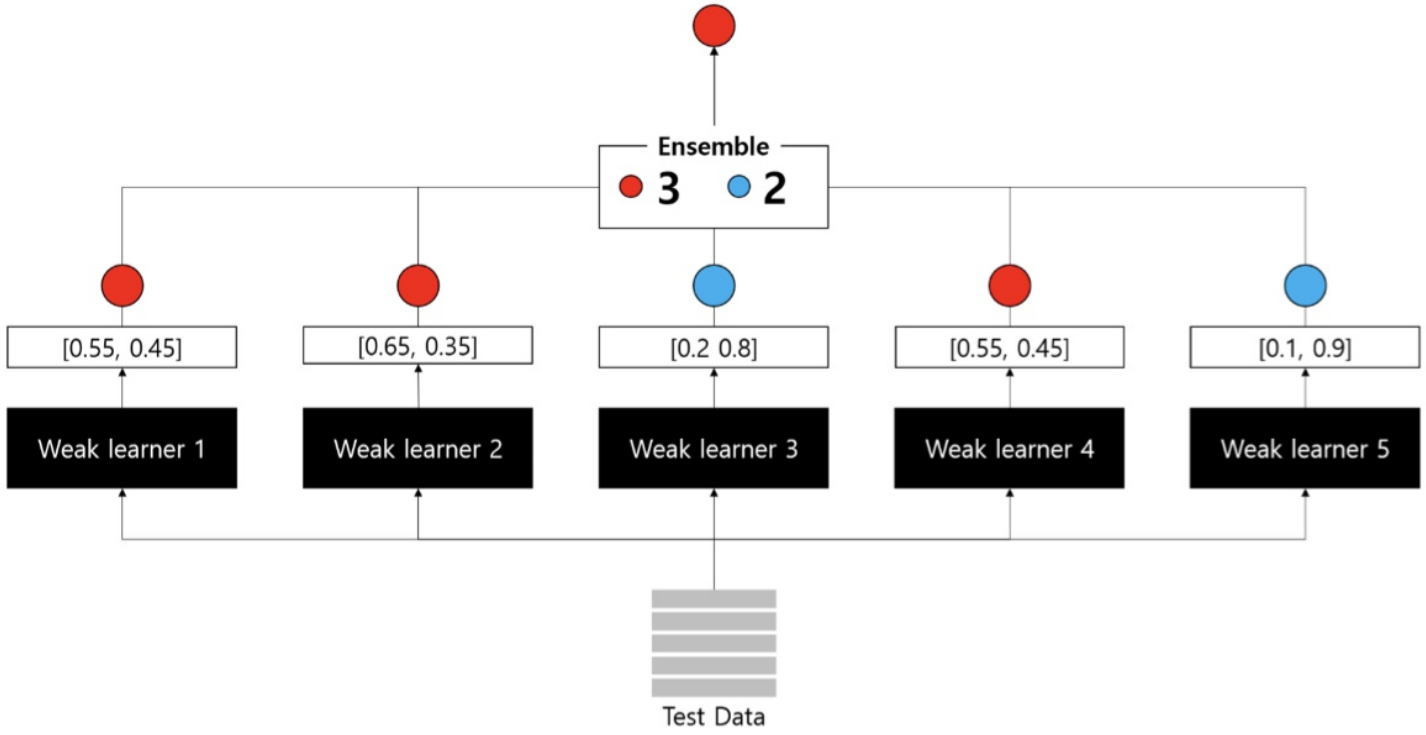
3. Ensemble

Voting

: 투표를 통해 결과 도출

Hard Voting

- Majority voting: 다수결로 선택



3. Ensemble

Soft Voting

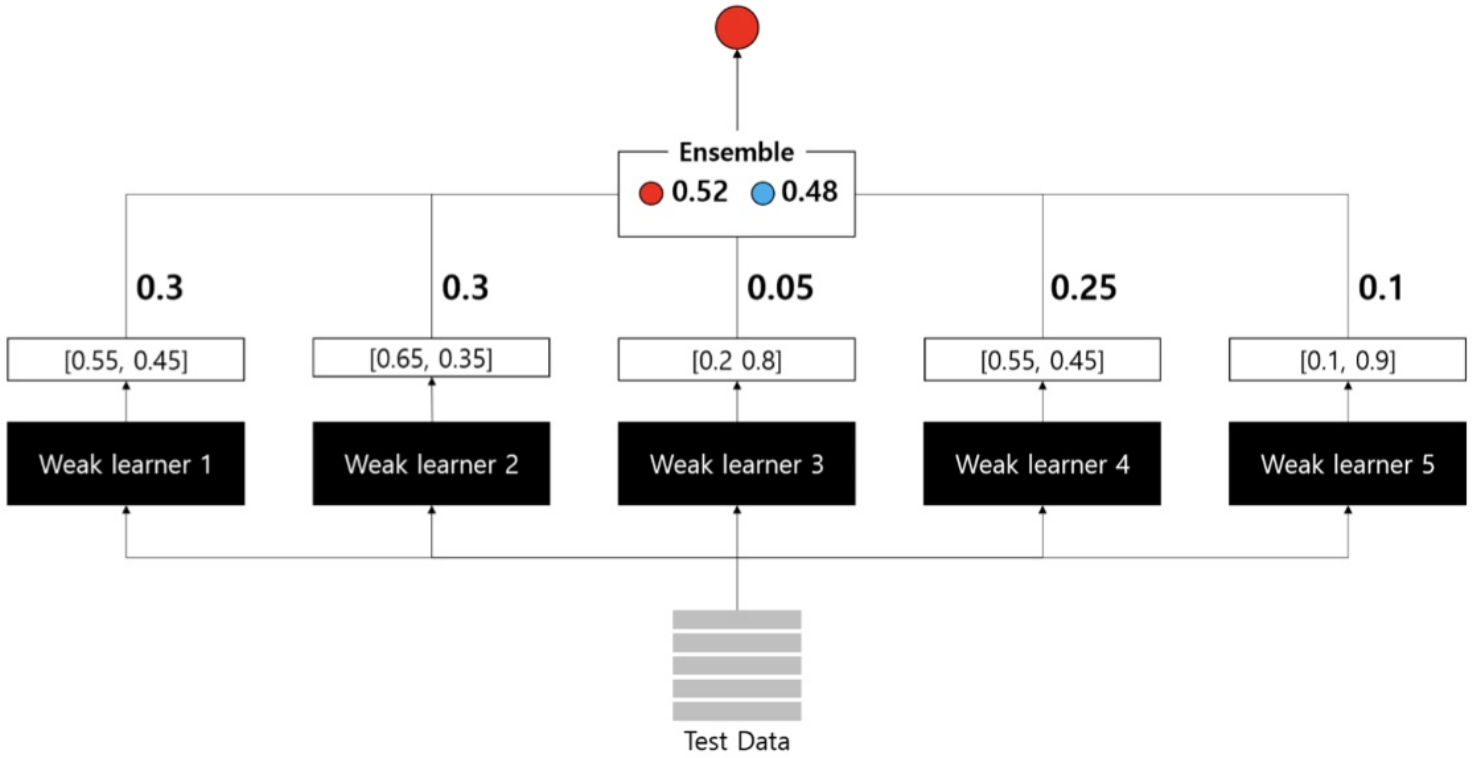
: 각 class별로 예측 확률의 평균값을 계산하여 가장 높은 class 선택

- Weak Learner 개별의 예측값은 중요하지 않다.

3. Ensemble

Weighted Voting

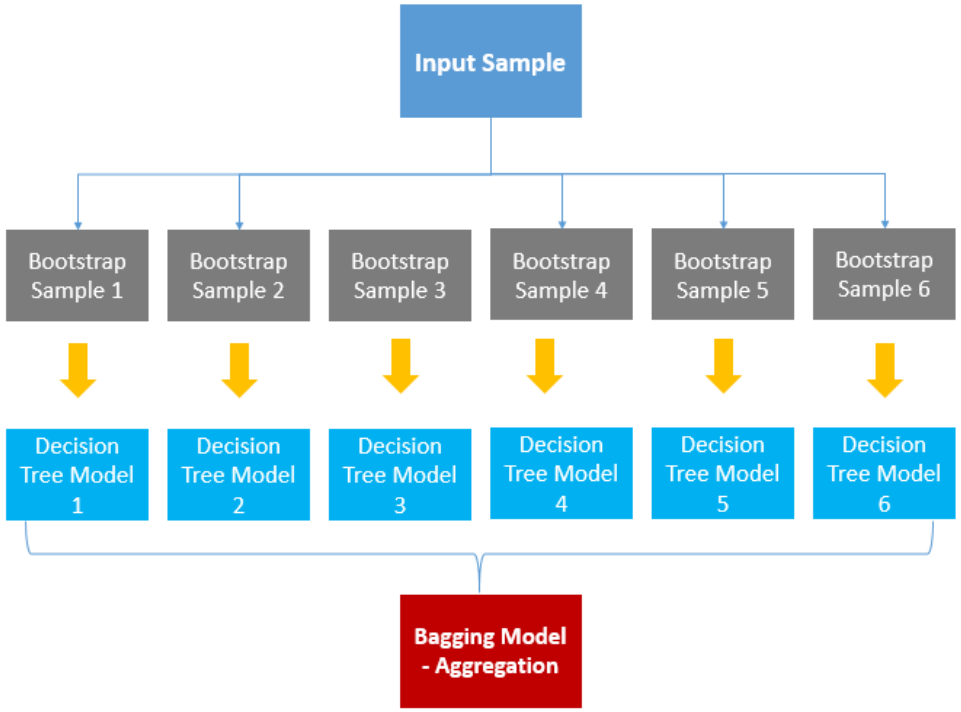
: Weak Learner의 신뢰도가 다를 경우, 각 모델별로 class에 대한 가중치를 부여하여 가중치 합을 사용한다.



4. Bagging

Bagging이란?

- Bootstrap Aggregation의 약자
- 샘플을 여러 번 뽑아(Bootstrap) 각 모델을 학습시켜 결과물을 집계(Aggregation)
- 범주형 데이터: Voting, 연속형 데이터: Average



Bootstrap Sampling

Bootstrap(복원추출): 주어진 데이터셋에서 Random Sampling하여 원하는 크기의 새로운 데이터셋을 만들어낸다.

- Low variance (Less sensitive)

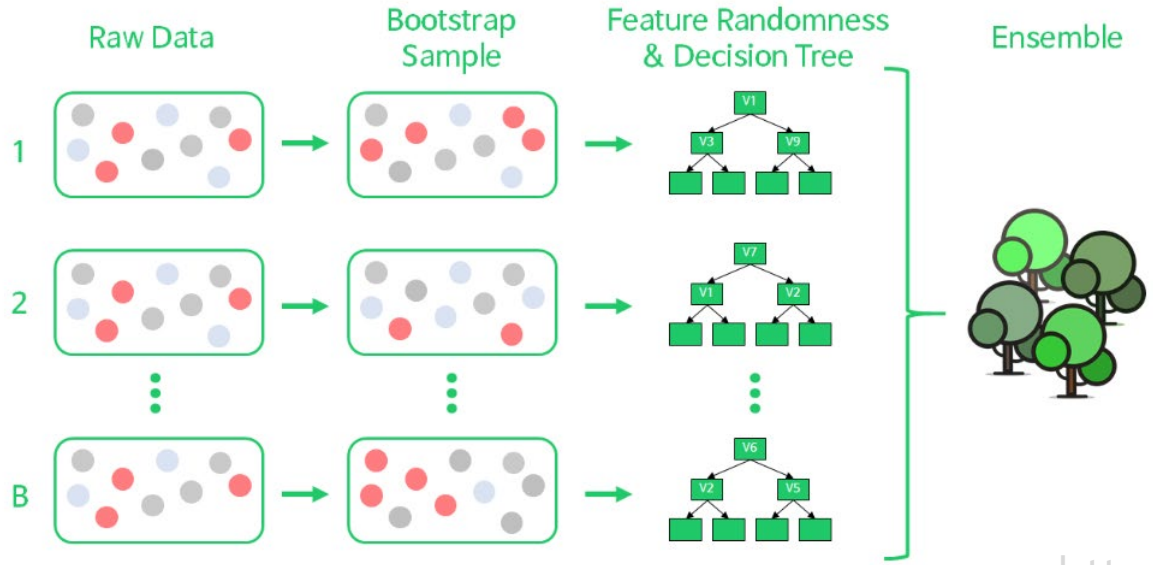
4. Bagging

Random Forest

: Decision Tree의 Overfitting 문제를 해결하기 위해 여러 개의 트리를 취합한 Ensemble Model

Diversity & Randomness! >> 각 트리가 uncorrelated model(≒ 독립적)

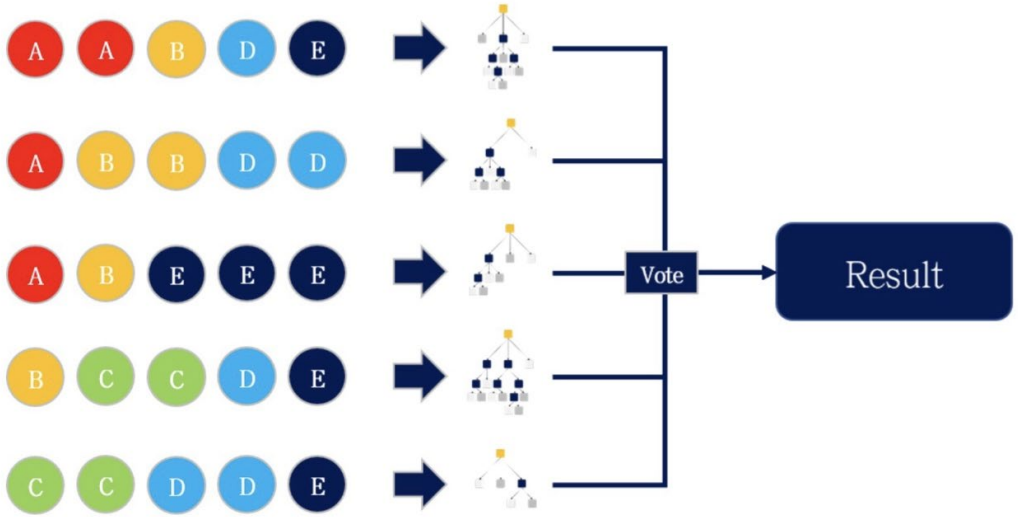
1. Bootstrap Sampling : N개의 sample을 **복원추출**
2. Feature Random Selection : 각 노드에서 중복을 허용하지 않고 랜덤하게 d개의 특성을 선택



4. Bagging

Random Forest Process

1. N개의 랜덤한 부트스트랩 샘플 추출(중복 허용)
2. 각 노드에서 중복을 허용하지 않고 랜덤하게 d개의 특성을 선택
3. IG와 같은 목적 함수를 기준으로 최선의 분할을 만드는 특성을 사용해 노드를 분할 [**불순도 감소량**이 가장 큰 특성]
4. 1~3 단계를 k번 반복
5. 다수결 투표 또는 평균에 따라 클래스 레이블 할당

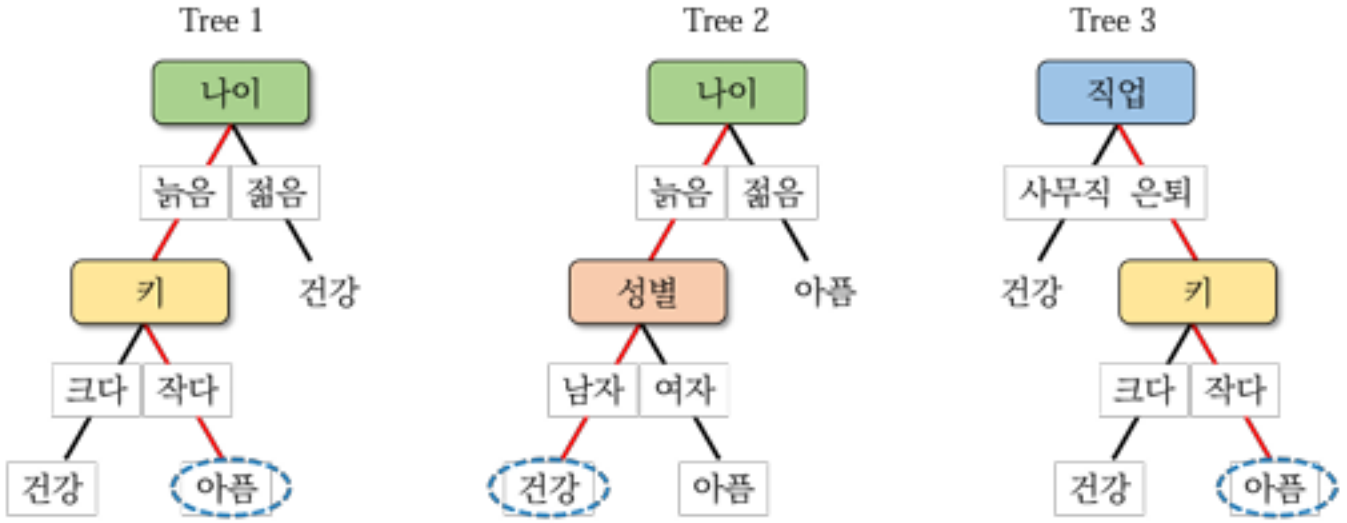


4. Bagging

Random Forest 예시

나이, 직업유무, 키, 성별을 이용해 건강한지 아닌지 판단하는 상황

- 총 3번의 표본을 추출하였고, 각각 {나이, 키}, {나이, 성별}, {직업, 키}를 입력변수로 선택
- 나이가 늙고, 키가 작고, 남성이고, 은퇴한 사람은 세 의사결정나무에 의해 건강하지 않다고 판별



4. Bagging

Variable Importance

Random Forest는 **변수의 중요도**(Variable Importance)를 산출하여 제공할 수 있다.

: 어느 변수가 예측 성능에 중요한 역할을 하는지를 추정함으로써 설명변수와 반응변수의 설명력을 확보

⇒ "해당 변수가 상대적으로 얼마만큼 종속변수에 영향을 주는가?"

- Mean Decrease in Impurity Importance / Permutation Importance / Drop Column Importance

Drop Column Importance

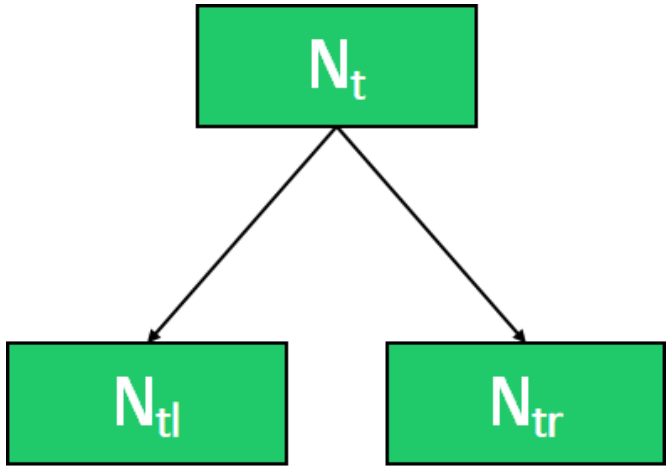
: (모든 변수를 사용하는 경우의 성능) – (특정 변수를 빼고 재학습한 경우의 성능)으로 변수 중요도 정의

- 특정 변수를 빼고 재학습했을 경우 성능이 많이 떨어진다면 해당 변수는 중요한 변수
- 개념이 매우 직관적이지만, 변수의 개수만큼 모델 재학습이 필요하기 때문에 매우 비효율적

4. Bagging

Mean Decrease in Impurity (MDI) Importance

: 각 변수가 분할될 때 **불순도 감소량의 평균**을 변수 중요도로 정의



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

$i(t)$: t노드의 impurity (entropy, gini index ...)

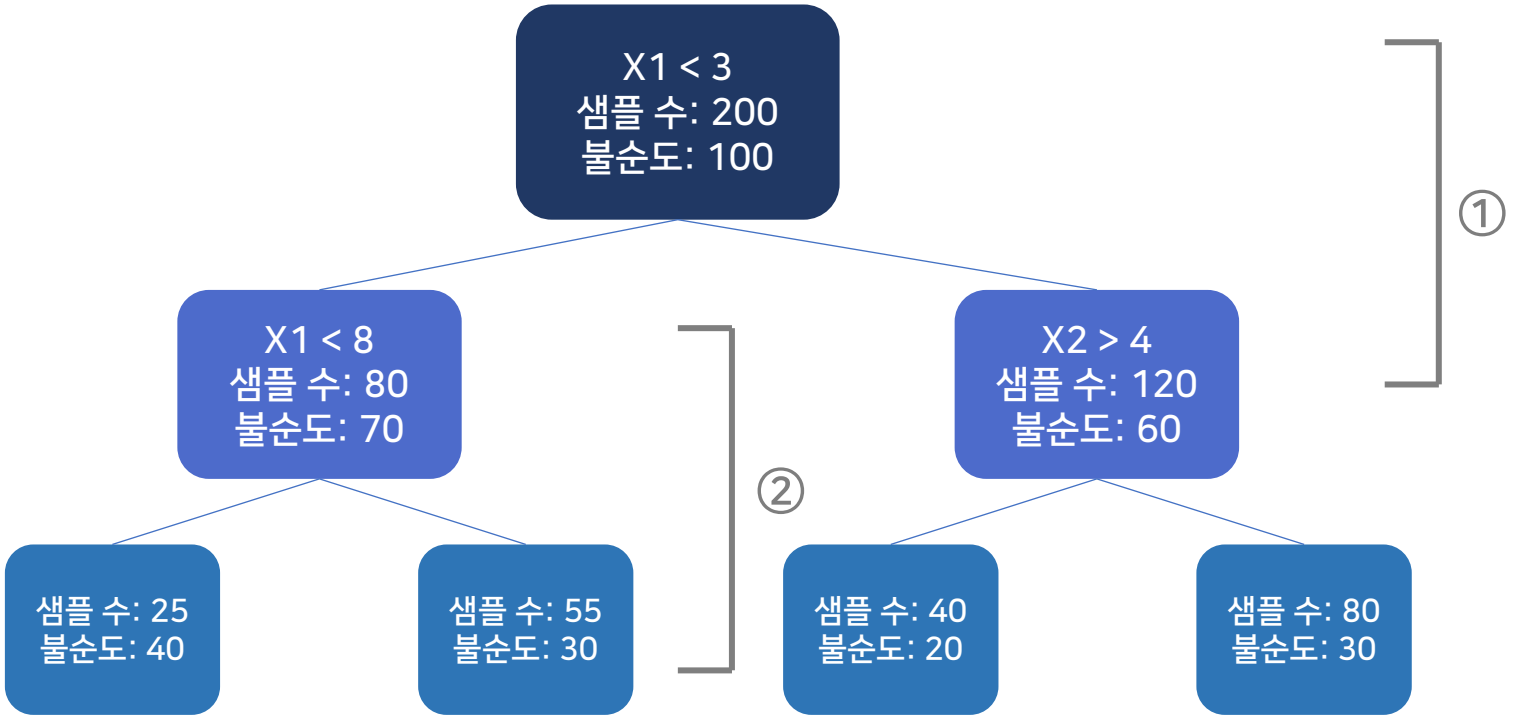
N_t : t노드의 샘플 수

4. Bagging

MDI Importance 예시

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

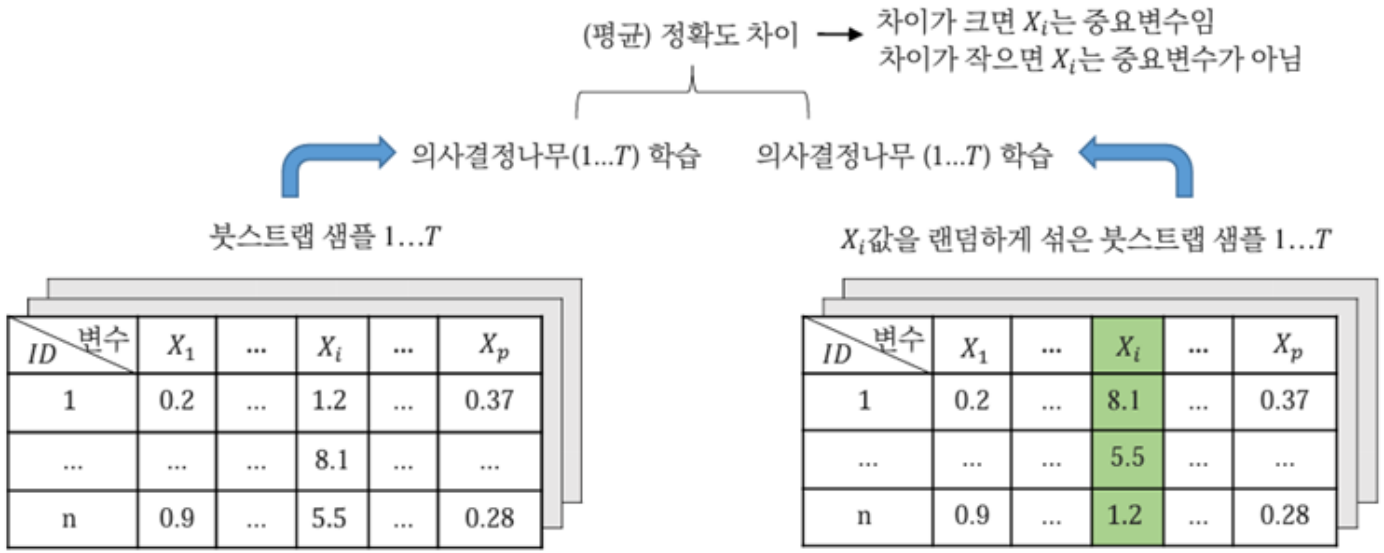
<변수 X1의 중요도 계산>



4. Bagging

Permutation Importance

1. 데이터셋에 의사결정나무를 학습시킨 후 정확도 산출
2. 데이터셋에서 입력변수 X_i의 값을 무작위로 섞은 후, 의사결정나무를 학습시킨 후 정확도 산출
3. 두 정확도의 차이가 크다면 입력변수 X_i의 중요도는 해당 의사결정나무에서 높음
4. 1~3번을 의사결정나무의 개수만큼 반복 후 평균으로 변수 중요도 정의



ID \ 변수	X_1	...	X_i	...	X_p
1	0.2	...	1.2	...	0.37
...	...	...	8.1	...	...
n	0.9	...	5.5	...	0.28

ID \ 변수	X_1	...	X_i	...	X_p
1	0.2	...	8.1	...	0.37
...	...	...	5.5	...	...
n	0.9	...	1.2	...	0.28

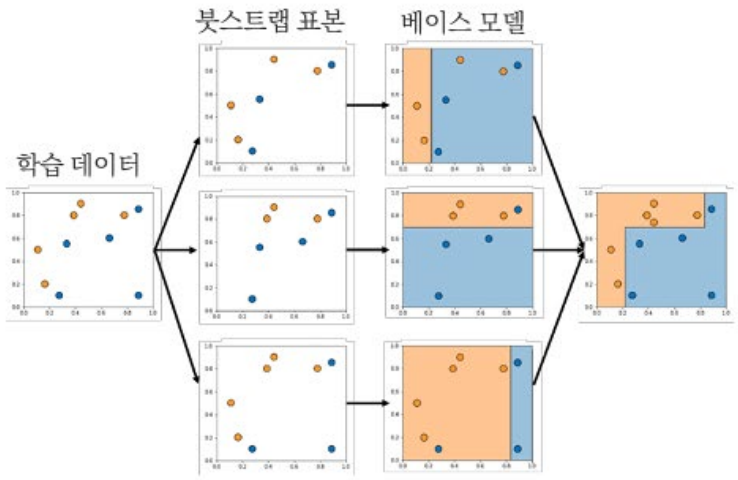
5. Boosting

Boosting

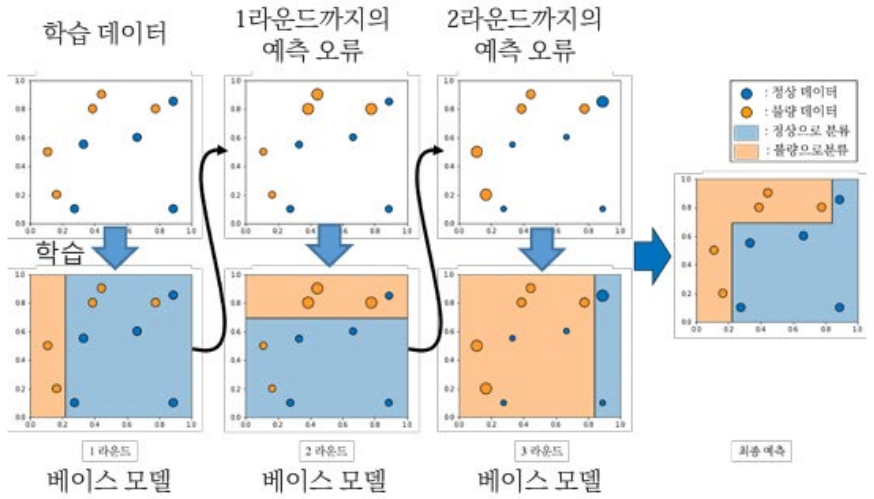
: 베이스 모델을 각 학습 라운드마다 **순차적으로 학습**시키고, 이를 결합하여 예측

- 각 라운드의 모델들은 이전 모델들이 잘못 예측한 부분에 대해 중점적으로 학습
- Adaptive Boosting, Gradient Boosting Machine(XGBoost, LightGBM, Catboost)

Bagging vs Boosting



Bagging: Parallel



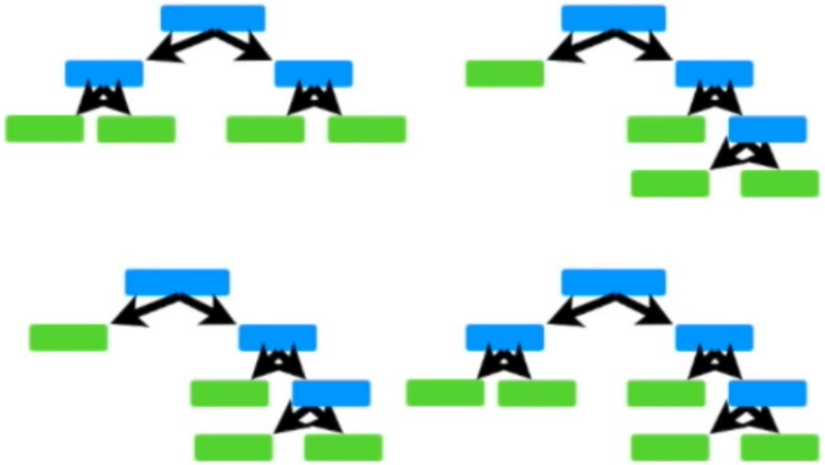
Boosting: Sequential

5. Boosting

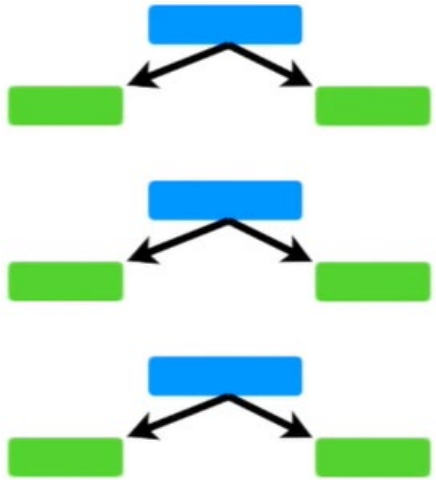
AdaBoost

: 다수의 Stump(Weak Classifier)로 구성된 알고리즘

- Weak Classifier를 모아서 최종적으로 Strong Classifier를 생성
- Random Forest와 달리 각 분류기의 중요도에 따라 **가중치 부여**



Random Forest



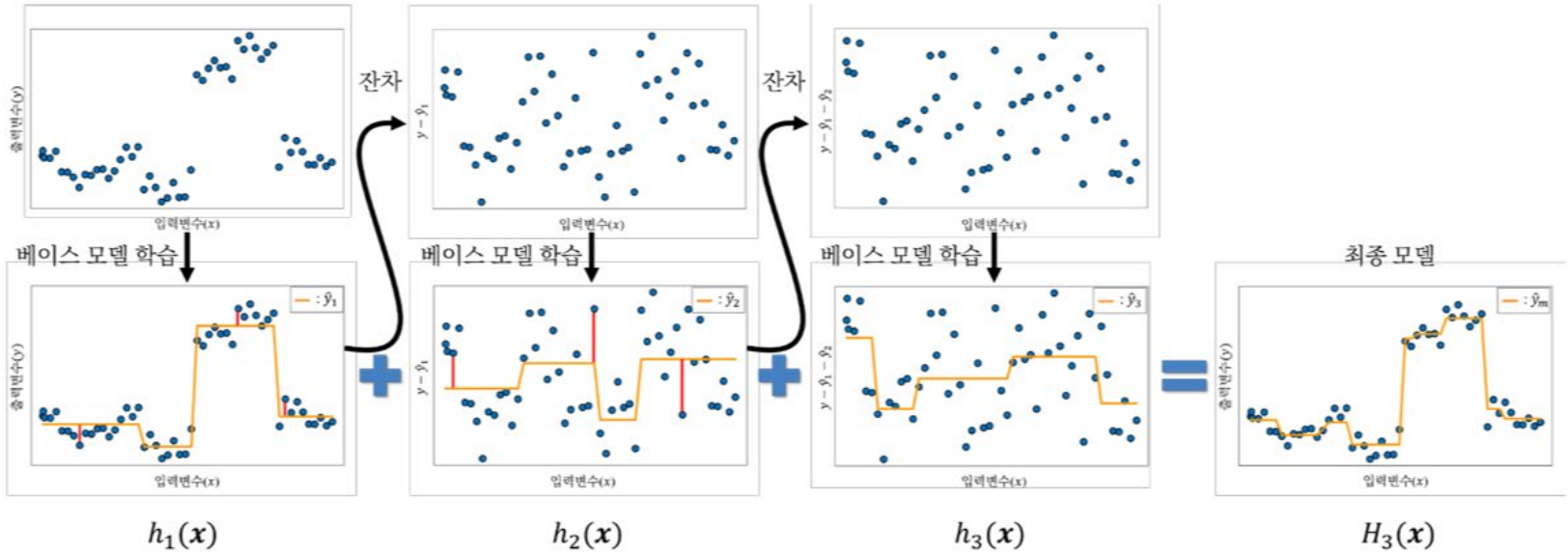
AdaBoost

5. Boosting

GBM: 잔차를 지속적으로 학습

: Residual Fitting!

- m라운드의 베이스모델 h_m이 (m−1)라운드까지의 베이스 모델을 결합한 $H_m = h_1 + \cdots + h_{m-1}$의 **잔차**를 학습
- 이전결합모델 H_{m-1}의 예측 오류는 아직까지 학습 못한 특징이라고 가정

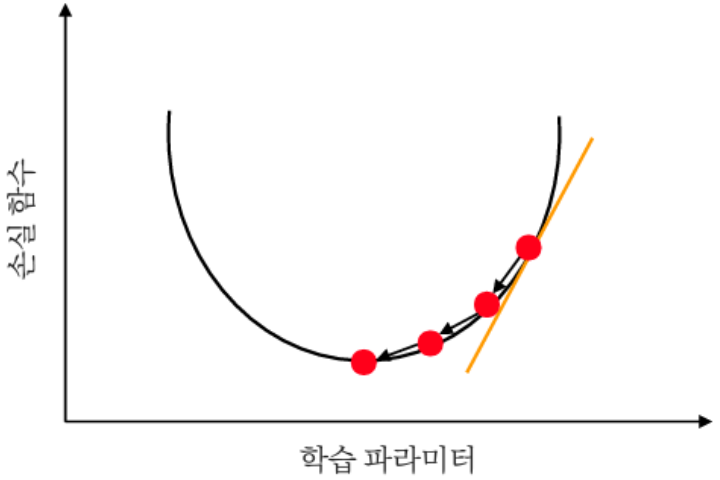


5. Boosting

회귀 문제에서의 Gradient Boosting Machine

- m라운드의 베이스 모델 h_m은 m라운드까지의 **누적된 잔차** $(y - H_m(x))$를 최소화하는 방향으로 학습
- 손실함수를 최소화하기 위해 일반적으로 Gradient Descent 방법으로 베이스 모델의 학습 파라미터를 조절

=> Gradient Descent + Boosting



n 개 데이터에 대한 평균 오류자승

예측한 출력변수

$$MSE = \frac{1}{n}\sum_{i=1}^{n}[y_i - H_{m-1}(x_i)]^2$$

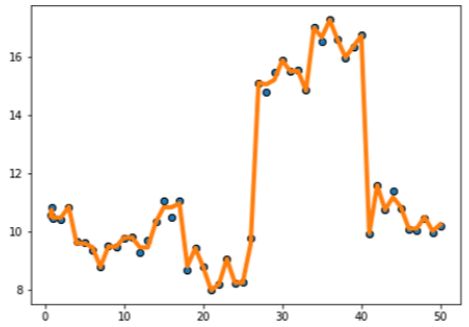
Minimize

h_m 학습

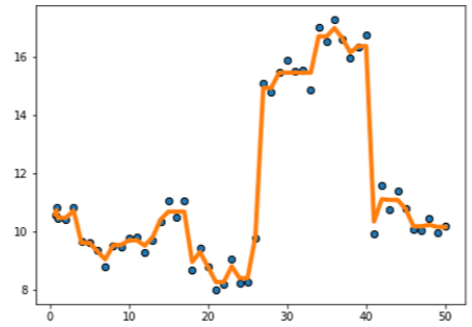
데이터 개수

실제 출력변수

Gradient: $$\frac{\partial L(y, H_{m-1})}{\partial H_{m-1}} = \frac{2}{n}\sum_{i=1}^{n}[H_{m-1}(x_i) - y_i]$$



Learning Rate=1.0 (M=10)



Learning Rate=0.5 (M=10)

5. Boosting

GBM 회귀 모델

- 각각의 베이스 모델이 학습하는 입력변수와 출력변수:

X_1	...	X_p	출력변수
$x_{1,1}$		$x_{1,p}$	y_1
$x_{2,1}$		$x_{2,p}$	y_2
⋮		⋮	⋮
$x_{n-1,1}$		$x_{n-1,p}$	y_{n-1}
$x_{n,1}$		$x_{n,p}$	y_n

h_0 학습

X_1	...	X_p	출력변수
$x_{1,1}$		$x_{1,p}$	$y_1 - h_0(\boldsymbol{x}_1)$
$x_{2,1}$		$x_{2,p}$	$y_2 - h_0(\boldsymbol{x}_2)$
⋮		⋮	⋮
$x_{n-1,1}$		$x_{n-1,p}$	$y_{n-1} - h_0(\boldsymbol{x}_{n-1})$
$x_{n,1}$		$x_{n,p}$	$y_n - h_0(\boldsymbol{x}_n)$

h_1 학습

X_1	...	X_p	출력변수
$x_{1,1}$		$x_{1,p}$	$y_1 - h_0(\boldsymbol{x}_1) - \nu h_1(\boldsymbol{x}_1)$
$x_{2,1}$		$x_{2,p}$	$y_2 - h_0(\boldsymbol{x}_2) - \nu h_1(\boldsymbol{x}_2)$
⋮		⋮	⋮
$x_{n-1,1}$		$x_{n-1,p}$	$y_{n-1} - h_0(\boldsymbol{x}_{n-1}) - \nu h_1(\boldsymbol{x}_{n-1})$
$x_{n,1}$		$x_{n,p}$	$y_n - h_0(\boldsymbol{x}_n) - \nu h_1(\boldsymbol{x}_n)$

h_2 학습

- h_0은 초기값으로, 일반적으로 회귀 문제에서는 **출력변수의 평균**을 사용
- 베이스 모델로는 회귀 트리 알고리즘인 CART를 주로 사용

5. Boosting

GBM 회귀 모델 예시

- 특정 제품의 두께를 예측하는 예제 (v(learning rate)=0.8)

설비	온도	재료	제품 두께	Error1
A	18	C	15	4
B	20	D	9	-2
A	22	C	14	3
B	19	D	8	-3
B	17	C	9	-2

h_0는 제품 두께의 평균을 예측하는 회귀 트리 모델

$$\frac{15+9+14+8+9}{5} = 11$$

Error1 계산

1번 데이터: $15 - 11 = 4$

2번 데이터: $9 - 11 = -2$

3번 데이터: $14 - 11 = 3$

4번 데이터: $8 - 11 = -3$

5번 데이터: $9 - 11 = -2$

5. Boosting

GBM 회귀 모델 예시

설비	온도	재료	제품 두께	Error1
A	18	C	15	4
B	20	D	9	-2
A	22	C	14	3
B	19	D	8	-3
B	17	C	9	-2

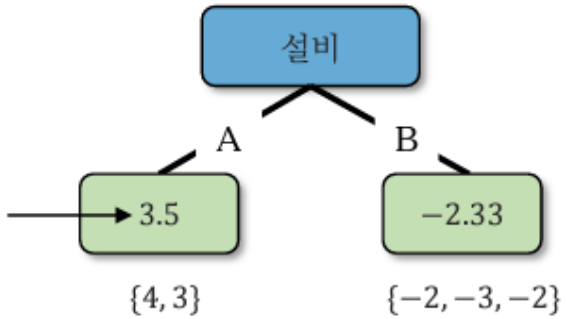
h_0는 제품 두께의 평균을 예측하는 회귀 트리 모델

$$\frac{15+9+14+8+9}{5} = 11$$

h_1은 Error1을 출력변수로 두고 회귀 트리 모델 생성

$$\text{끝노드 출력값} = \frac{j\text{노드 내 잔차합}}{j\text{ 노드 내 데이터 개수}}$$

$$= \frac{4+3}{2} = 3.5$$



5. Boosting

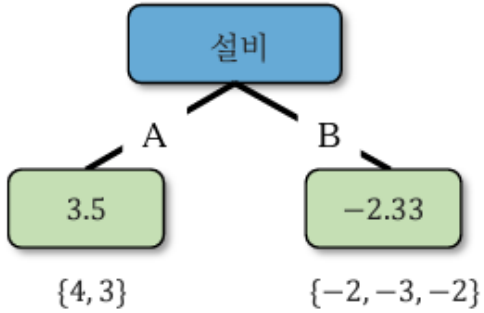
GBM 회귀 모델 예시

설비	온도	재료	제품 두께	Error1	예측값
A	18	C	15	4	13.8
B	20	D	9	-2	9.14
A	22	C	14	3	13.8
B	19	D	8	-3	9.14
B	17	C	9	-2	9.14

h_0는 제품 두께의 평균을 예측하는 회귀 트리 모델

$$\frac{15+9+14+8+9}{5} = 11$$

h_1은 Error1을 출력변수로 두고 회귀 트리 모델 생성



$H_1(h_0 + \nu h_1)$ 예측 (ν)

1번 데이터: $11 + 0.8 \times 3.5 = 13.8$

2번 데이터: $11 + 0.8 \times -2.33 = 9.14$

3번 데이터: $11 + 0.8 \times 3.5 = 13.8$

4번 데이터: $11 + 0.8 \times -2.33 = 9.14$

5번 데이터: $11 + 0.8 \times -2.33 = 9.14$

5. Boosting

XGBoost

: Gradient Boost 알고리즘을 병렬 학습이 지원되도록 구현한 라이브러리

- GBM과 동일하게 이전 라운드에서의 예측 오류를 다음 라운드의 모델 학습에 반영시킴
- **규제항 $\Omega(h)$: 자체적으로 트리의 복잡성에 패널티를 부여하여 Overfitting 규제**

Minimize m 라운드까지의 예측 오류$(y - H_m(x))$ + 트리 규제항 $\Omega(h_m)$

h_m 학습

T: 트리 끝노드 수, γ: 트리 복잡도에 대한 페널티

$$\Omega(h_m) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

베이스 모델 / 출력값 페널티 / 트리의 j 번째 끝노드에서의 출력값

① ②

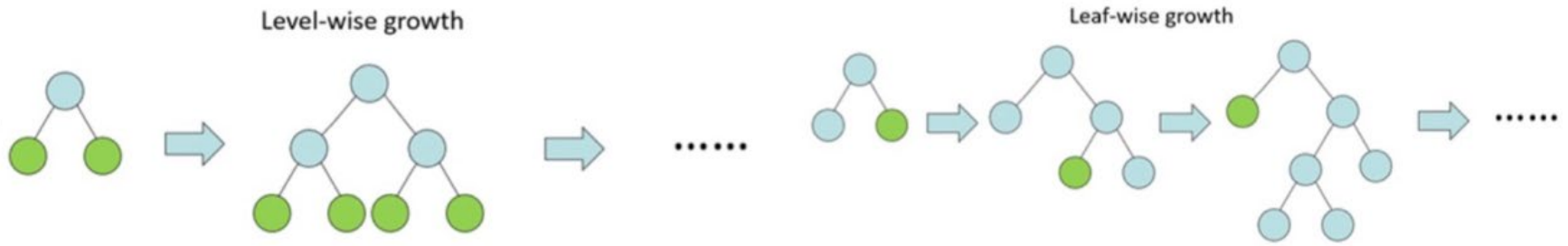
① : 트리가 커짐에 따른 패널티

② : 특정 끝노드의 출력값이 커지는 것에 대한 패널티

5. Boosting

LightGBM

- 속도가 빠르다
- 트리 구조가 수평적으로 확장하는 다른 알고리즘과 달리 수직적으로 확장(leaf-wise) => 손실을 줄임



Categorical Boosting (Catboost)

- Overfitting 해결 + 학습 속도 개선 + 범주형 입력변수 처리에 유용

Summary

Decision Tree

- 입력변수 공간을 수직 또는 수평으로 나누는 작업을 반복하며 분류 규칙을 만들어내는 알고리즘
- 불순도 평가지수: Entropy, IG, GR, Gini Index
- Overfitting의 위험이 크다 – Pruning/Ensemble로 개선
- 하이퍼 파라미터 튜닝을 통해 최적의 의사결정나무 모델을 찾아야 한다.

Ensemble

- 집단 지성의 원리
- Random Forest: Bootstrap Sampling 기반의 앙상블 알고리즘
- Bagging: Parallel – Random Forest
- Boosting: Sequential – AdaBoost, GBM, XGBoost, LightGBM, CatBoost

Reference

- 8기 장준혁님 <DT & Ensemble>
- 박노성 교수님 <데이터사이언스개론> 5강
- 김창욱 교수님 <머신러닝과 산업응용> 4강, 6강
- https://velog.io/@vvakki_
- https://tyami.github.io/

발표자 : 신소연
E-mail: shin020810@yonsei.ac.kr