강원대학교
AI 소프트웨어학과

머신러닝2

- 지도학습 –

Decision Tree

KNU 강원대학교

Decision Tree

기계학습의 방법(지도학습)

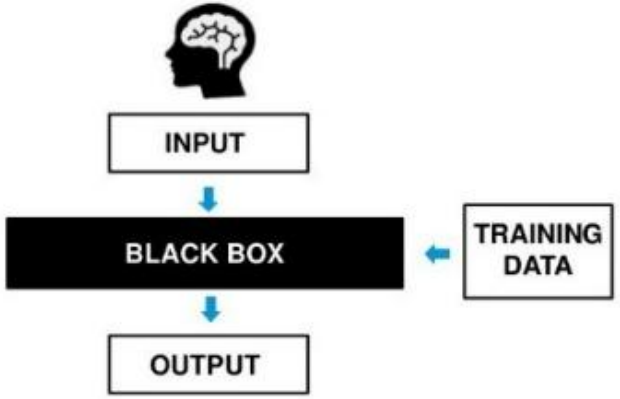
- 다중 회귀분석(Regression)
- 로지스틱 회귀분석(Logistic Regression)
- 신경망(Atificial Neural Network)
- 서포트 백터 머신(Support Vector Machine)
- 의사결정나무(Decision Tree)
- 앙상블(Ensemble)
- K-근접 이웃기법(k-Nearest Neighbor)

Decision Tree

모델 해석 : Black box vs White box

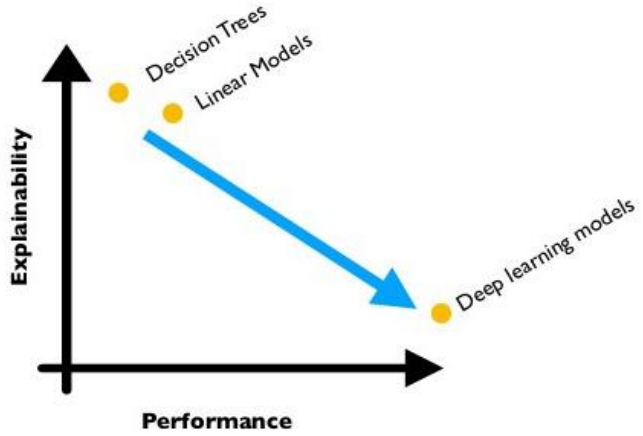
Black box

- 아웃풋은 원하는 형태로 도출 할 수 있지만 '무엇을', '왜', '어떻게' 등 특정 근거를 들어 결과가 나온 이유를 알 수 없는 것을 의미함
- 한계 : AI 의사가 어떤 환자를 암이라고 진단하는데 '왜 이 환자가 암인가'를 설명하지 못하게 되면 신뢰도를 잃게 됨



White box

- 아웃풋이 원하는 형태로 도출 할 수 있으며 예측을 만드는 연산 과정을 쉽게 확인할 수 있음 즉 판단의 근거를 제시할 수 있음
- 한계 : 모델의 성능과 설명 가능성 사이는 trade-off관계로 설명이 가능한 모델은 성능이 낮을 수 있음



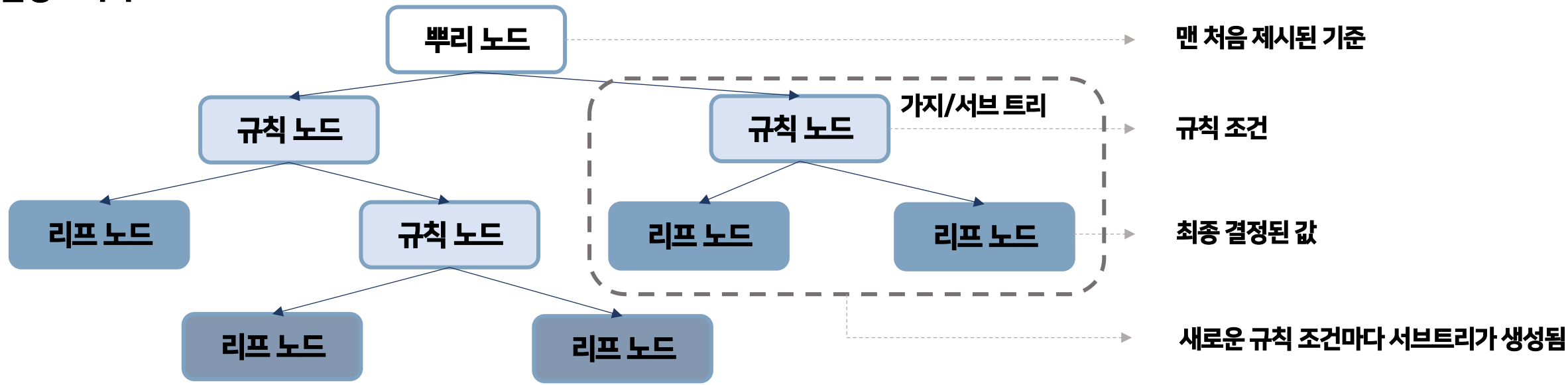
어떤 것이 더 좋다 나쁘다 등을 평가할 수 없으며, 머신 러닝의 목적성에 따라 적절한 모델을 고르는 것이 중요함

Decision Tree

결정트리 개념

- 결정트리(Decision Tree)는 데이터에 있는 규칙(Rule)을 학습을 통해 자동으로 찾아내 트리(Tree)기반의 규칙을 만드는 모델임
- 다양한 범주형 독립 변수들(조합을 인지해)을 활용해 모델링 가능함
- 데이터들이 가진 속성들로 분할 기준 속성을 판별하고, 나무 형태로 모델링하는 모델

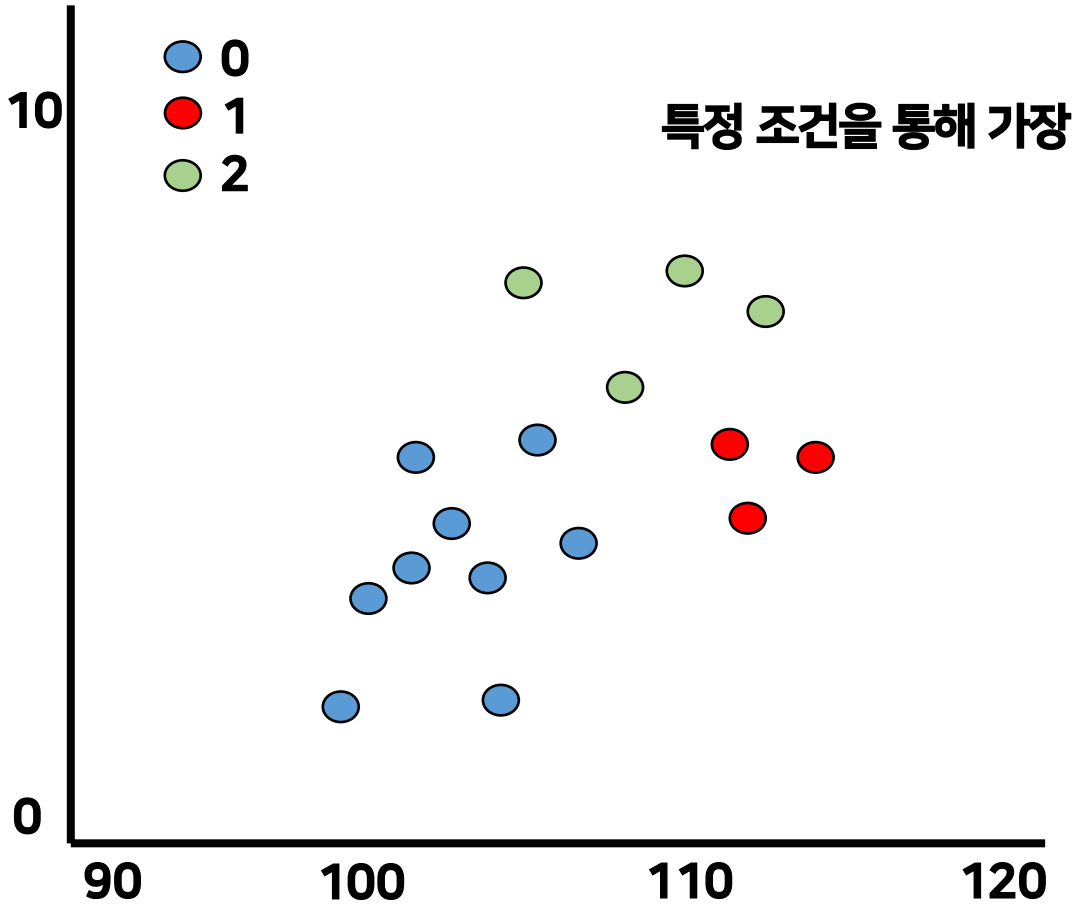
결정트리 구조



Decision Tree

결정트리 개념 – 재귀적 분할(Recursive Partitioning)

- Tree를 만드는 과정
- 그룹이 최대한 동질 하도록 반복적으로 하위 그룹으로 분리

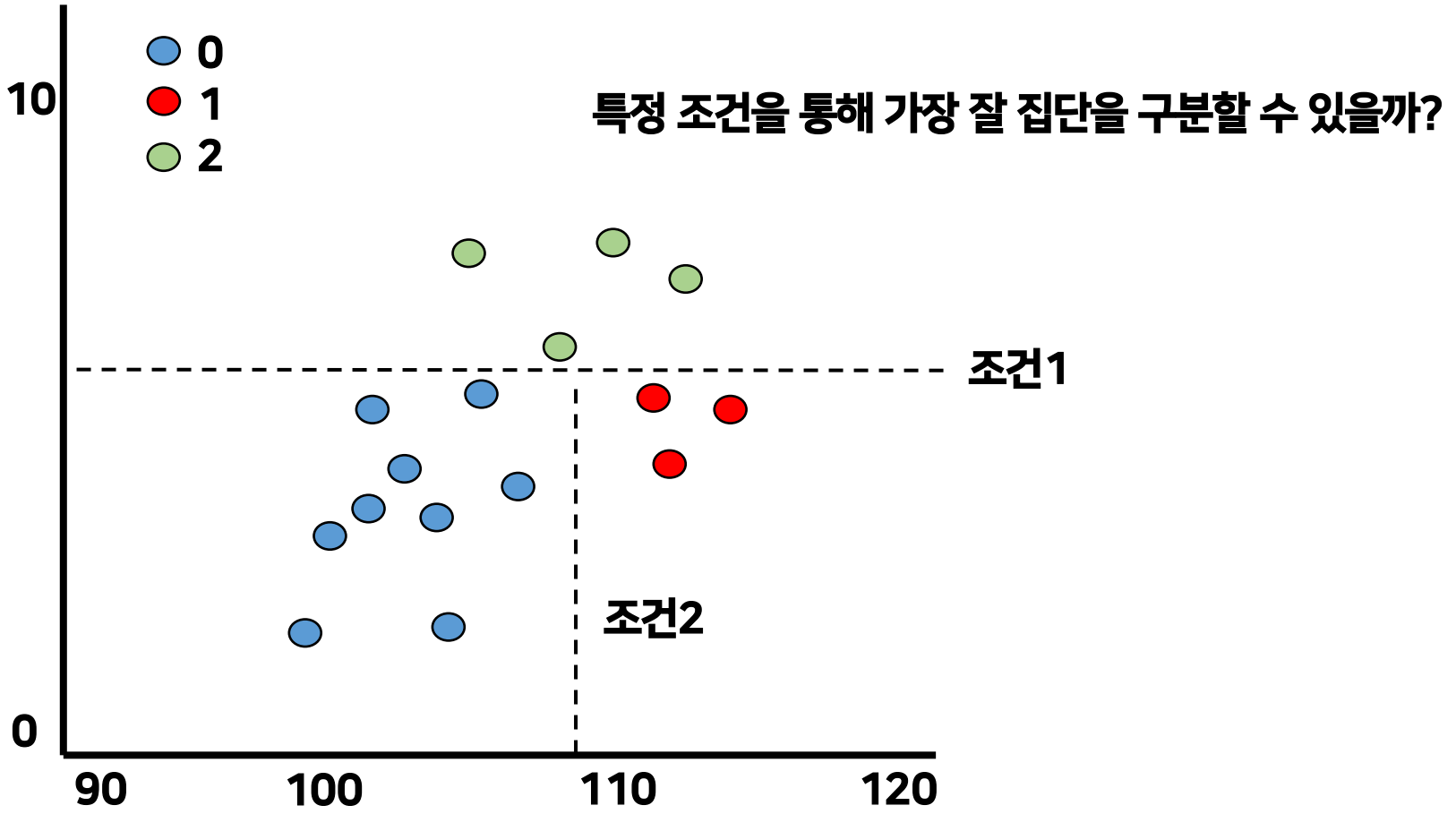


특정 조건을 통해 가장 잘 집단을 구분할 수 있을까?

Decision Tree

결정트리 개념 – 재귀적 분할(Recursive Partitioning)

- Tree를 만드는 과정
- 그룹이 최대한 동질 하도록 반복적으로 하위 그룹으로 분리



Decision Tree

결정트리 특징

- 분류와 회귀모델 둘 다 가능한 머신러닝 모델임(어떤 알고리즘을 사용 하느냐에 따라 다름)
- 결정규칙(Decision Rule)을 나무구조(Tree)로 도표화해 분류 및 예측을 수행하는 방법
- 피처의 스케일링이나 정규화 등 사전 가공 영향도가 크지 않음

결정트리 장점

- 불순도 룰을 기반으로 하고 있어 알고리즘을 쉽고 직관적으로 이해하기 좋은 알고리즘임

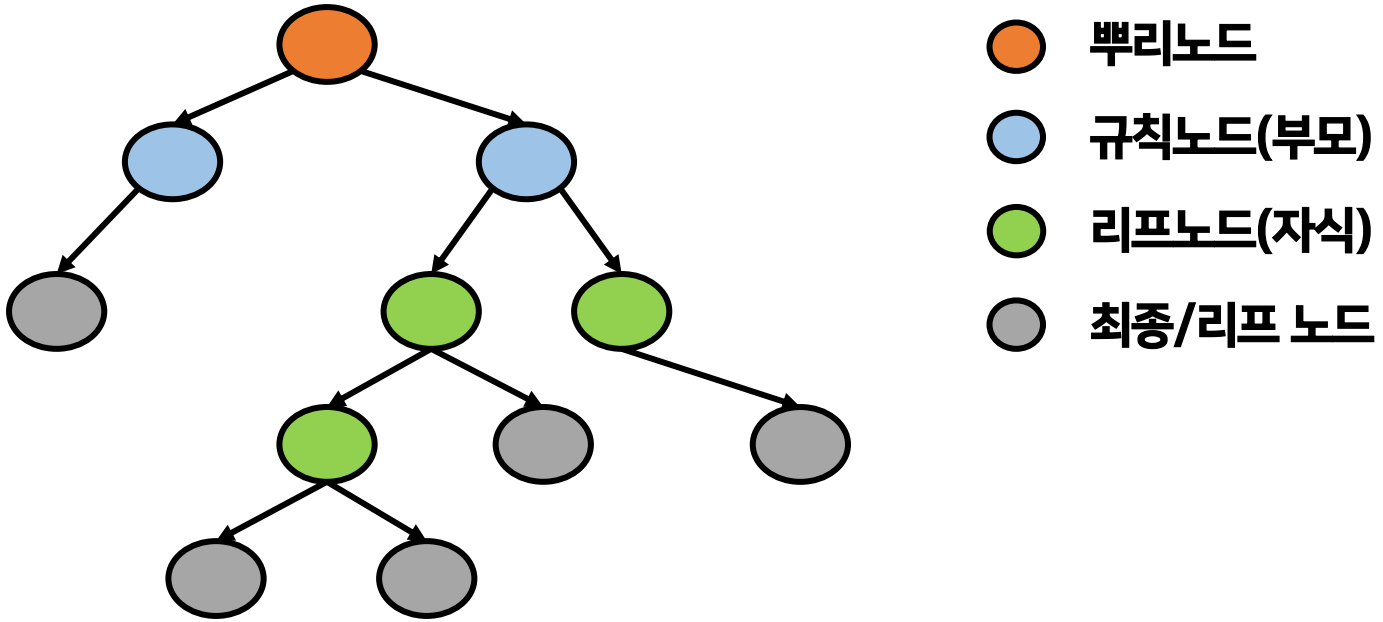
결정트리의 종류

- 분류나무(Classification Tree) : 목표 변수가 범주형 변수
- 회귀나무(Regression Tree) : 목표 변수가 수치형 변수(잘 사용하지 않음)

Decision Tree

분류나무(Classification Tree)의 프로세스

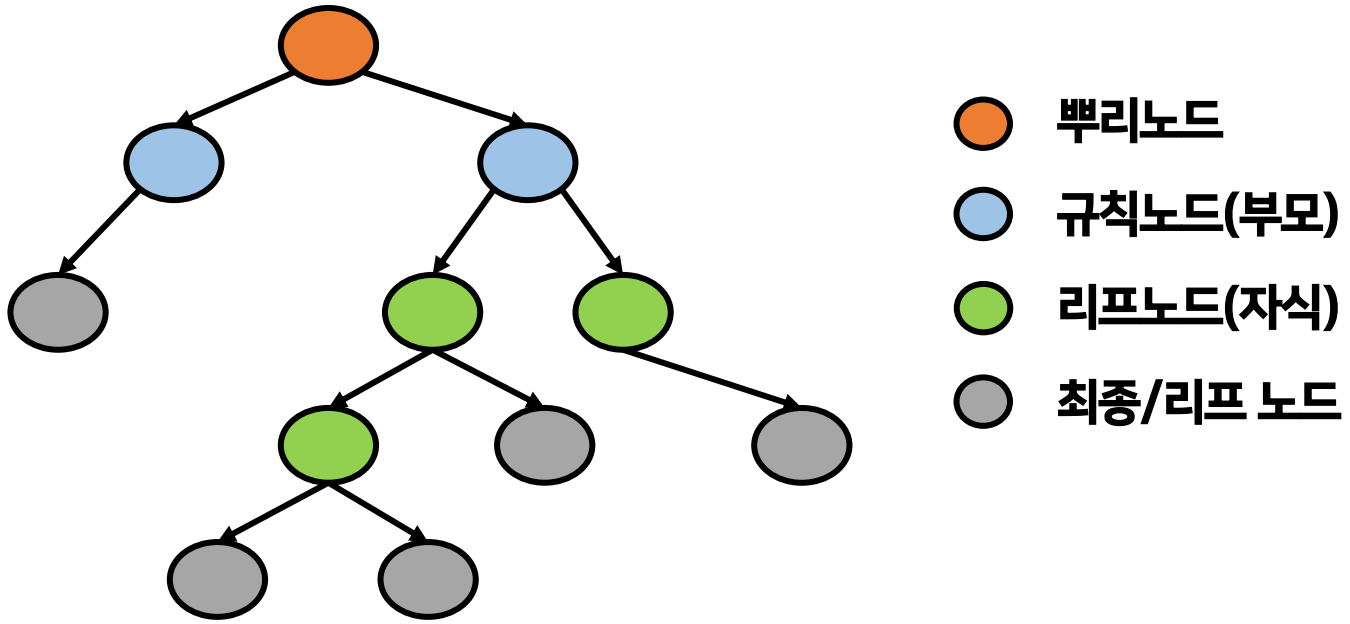
- Step 1(변수 선택) : 종속변수와 관계가 있는 독립변수 선택
- Step 2(성장) : 분석목적과 데이터의 적절한 분리 기준과 규칙을 통해 의사결정 나무 구조 작성
- Step 3(가지치기) : 일반화 가능성을 증대 시키는 트리를 제외한 서브트리를 제거해 효율성을 높임
- Step 4(타당성 평가) : 모델평가를 통해 최종모델을 선정
- Step 5(해석 및 예측) : 새로운 데이터에 대해서 분류(Classification), 예측(Prediction)을 수행함



Decision Tree

분류나무(Classification Tree)

- 불순도(Impurity) → 다양한 범주의 데이터들을 포함하고 있는지에 따라 노드의 분리의 기준을 세움
- 분류 알고리즘과 불순수도 지표
  - CART : 지니 지수(Gini Index) 이분류(클래스의 비율이 비슷해야 함)에 특화됨, 다중분류도 가능
  - C4.5 : 엔트로피(Entropy Index), 정보이익비율(Information Gain Ratio), 이익비율(Gain Ratio) – 다분류(클래스의 비율이 차이가 나도 됨)



Decision Tree – CART(Classification And Regression Tree)

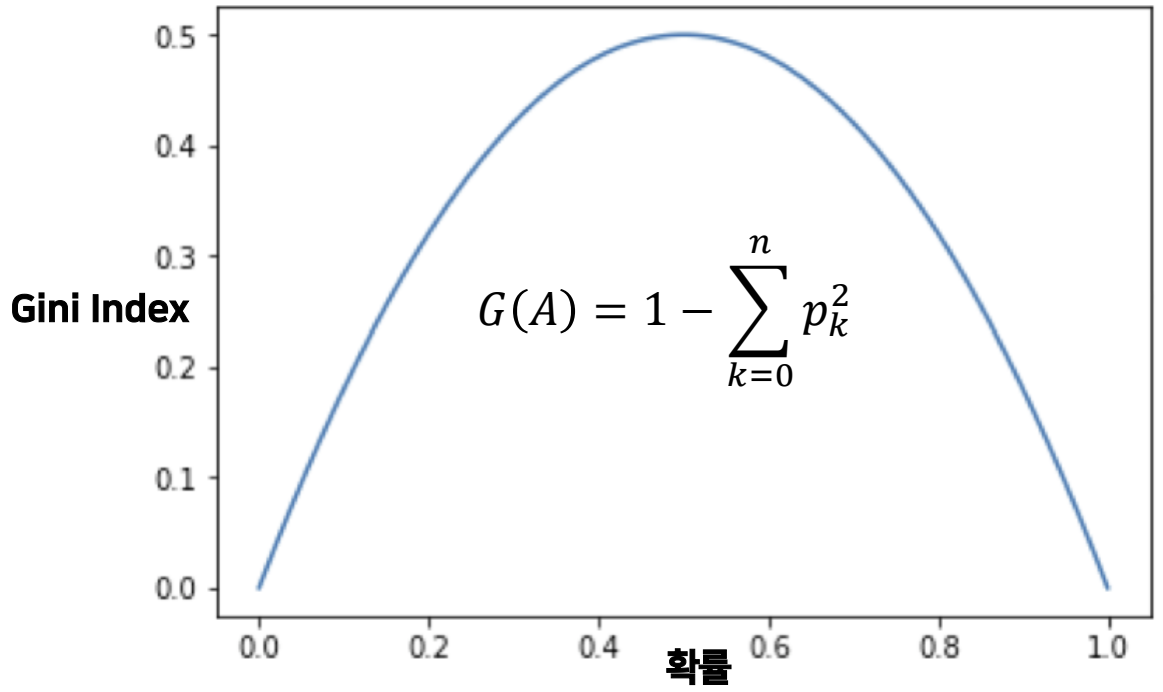
CART 알고리즘

- 불순도 알고리즘 : 지니 계수(Gini Index) – 불확실성이 작거나 크거나
- 이분류에 주로 사용(Binary Split)
- 학습 데이터를 이용하여 나무를 성장시키고, 검증 데이터를 이용하여 가지치기를 함

p_k^2 : 집단의 확률의 제곱

$$G(A) = 1 - \sum_{k=1}^{n} p_k^2$$

집단의 확률에 따라 지니 계수가 변함



Decision Tree – CART(Classification And Regression Tree)

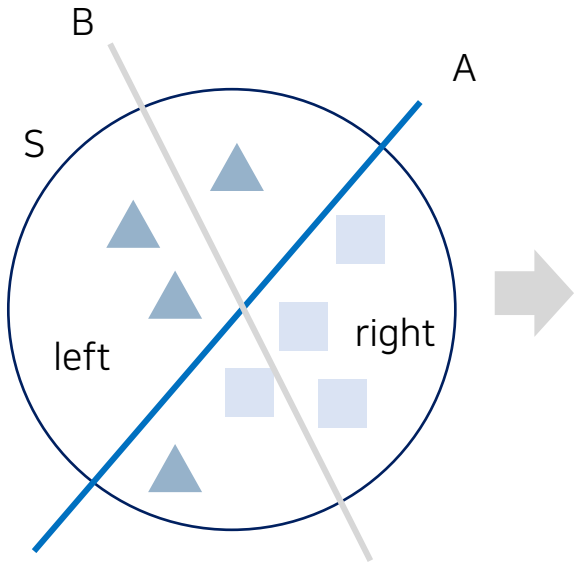
- 경제학자인 코라도 지니(Corrado Gini)의 이름을 딴 계수로 원래 경제학에서 불평등 지수(불순도)를 나타낼 때 사용하는 계수임
- 서로 다른 값이 섞여 있으면(혼잡할 시) 지니 계수가 높고, 같은 값이 많을 시(균일할 시) 지니 계수가 낮음
- 즉, 결정트리에서 지니 계수가 낮은 속성을 기준으로 분할함

$$G = 1 - \sum_{k=1}^{c} p_k^2$$

$m_{left/right}$는 왼쪽/오른쪽 샘플 수

$G_{left/right}$는 왼쪽/오른쪽 지니 계수

ΔG : **불순도 감소량 (얼마나 불순도가 줄었는지)**



$$G = 1 - \left(\frac{4}{8}\right)^2 - \left(\frac{4}{8}\right)^2 = 0.5$$

$$G_{left} = 1 - \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2 = 0$$

$$G_{right} = 1 - \left(\frac{1}{5}\right)^2 - \left(\frac{4}{5}\right)^2 = 0.31$$

$$\Delta G = G - \left(\frac{3}{8} \times 0 + \frac{5}{8} \times 0.31\right)$$

$$\text{Cost 함수} = \frac{m_{left}}{m} \times G_{left} + \frac{m_{right}}{m} \times G_{right}$$

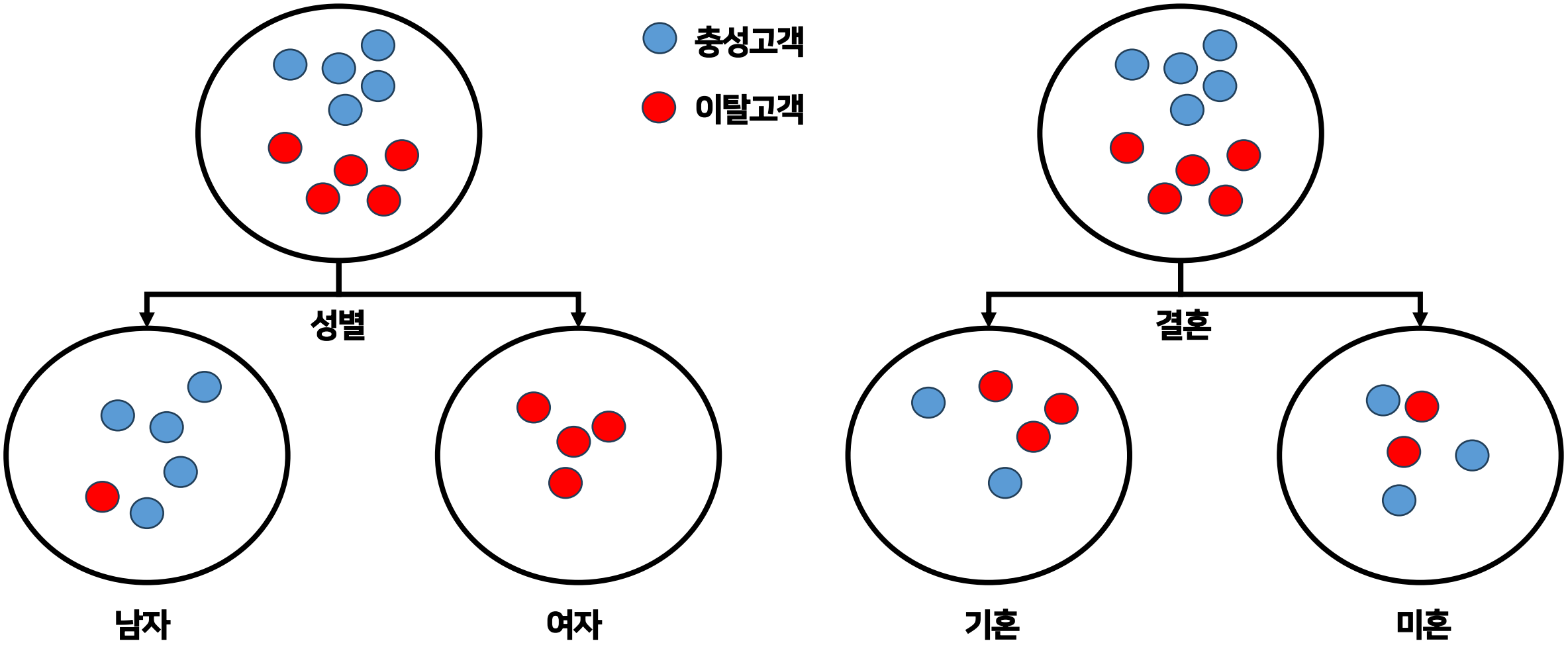
$$\text{Cost} = \left(\frac{3}{8}\right) \times 0 + \left(\frac{5}{8}\right) \times 0.31 = 0.193$$

불순도가 줄어듦 : $0.5 \rightarrow 0.193$

Decision Tree – CART(Classification And Regression Tree)

CART 알고리즘

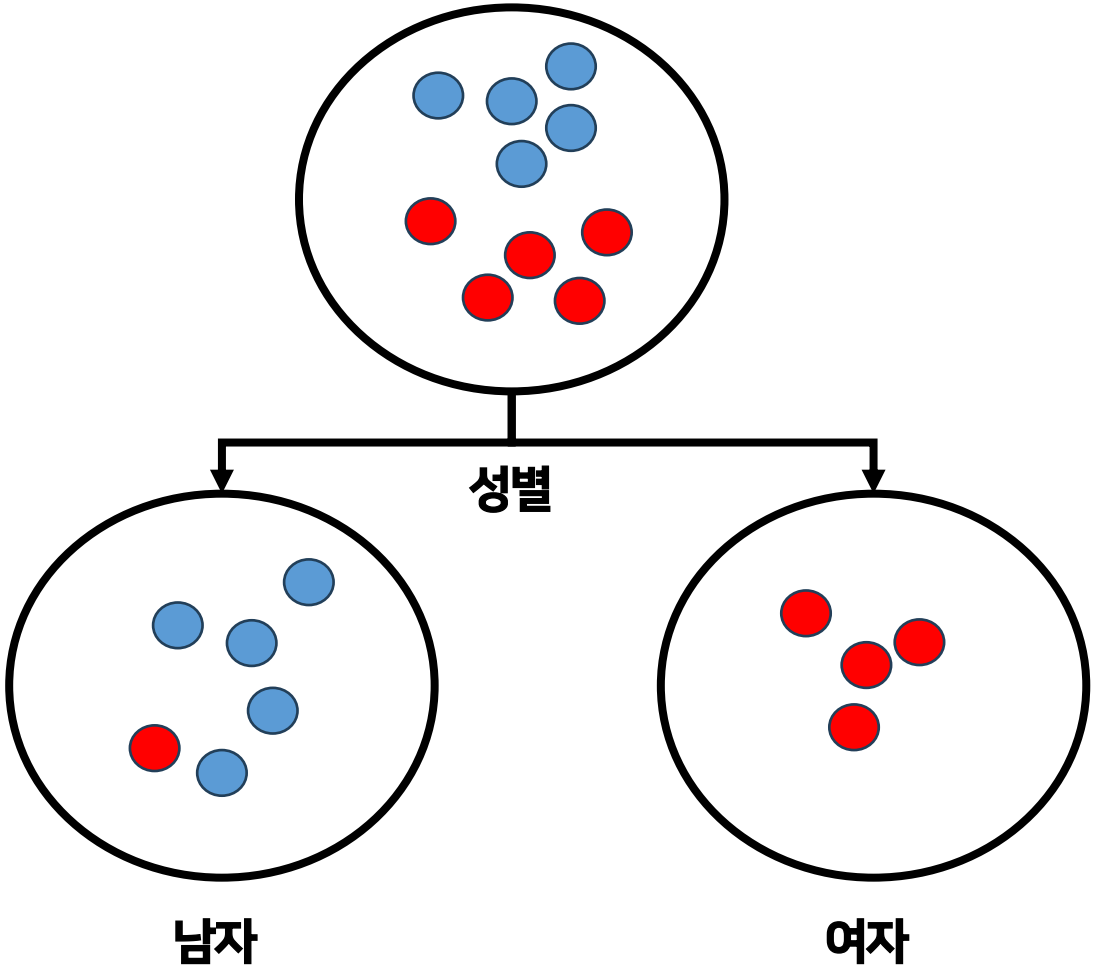
- 홈쇼핑의 충성고객과 이탈고객에 대한 특징을 분류해보자



Decision Tree – CART(Classification And Regression Tree)

CART 알고리즘

- 홈쇼핑의 충성고객과 이탈고객에 대한 특징을 분류해보자



$$G(\text{분할 전}) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2 = 0.5$$

$$G(\text{남}) = 1 - \left(\frac{5}{6}\right)^2 - \left(\frac{1}{6}\right)^2 = 0.278$$

$$G(\text{여}) = 1 - \left(\frac{0}{4}\right)^2 - \left(\frac{4}{4}\right)^2 = 0$$

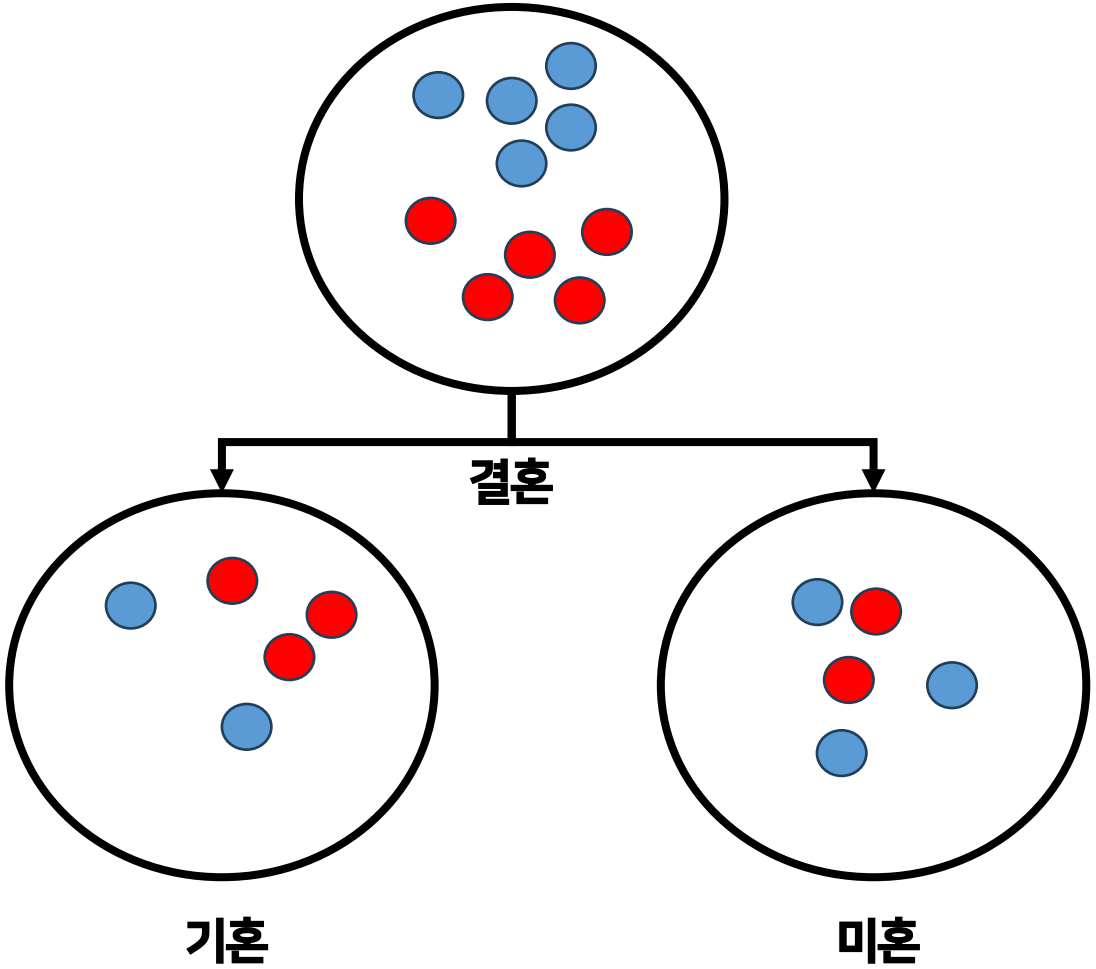
$$G(\text{성별}) = \left(\frac{5}{10}\right)(0.5) + \left(\frac{5}{10}\right)(0) = 0.167$$

$$\Delta G = G - cost = 0.5 - 0.167 = 0.333$$

Decision Tree – CART(Classification And Regression Tree)

CART 알고리즘

- 홈쇼핑의 충성고객과 이탈고객에 대한 특징을 분류해보자



$$G(\text{분할 전}) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2 = 0.5$$

$$G(\text{기혼}) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$$

$$G(\text{미혼}) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 0.48$$

$$G(\text{결혼}) = \left(\frac{5}{10}\right)(0.48) + \left(\frac{5}{10}\right)(0.48) = 0.48$$

$$\Delta G = G - cost = 0.5 - 0.48 = 0.02$$

Decision Tree – CART(Classification And Regression Tree)

CART 알고리즘

- **CART 알고리즘의 경우 이진분류만 가능함 따라서 다중항목이 있을 경우 한 기준에 따라 이진분류해서 보아야 함**
- **분류의 경우 지니 계수(Gini index)를 이용하며, 예측의 경우 분산의 감소량을 이용함**
- **분류에 대한 CART 비용함수** $J(k, t_k) = \frac{m_{left}}{m} \times G_{left} + \frac{m_{right}}{m} \times G_{right} = G - \Delta G$

$m_{left/right}$는 왼쪽/오른쪽 서브셋의 샘플 수

$G_{left/right}$는 왼쪽/오른쪽 서브셋의 지니 계수

Temperature	Humidity	Windy	Class(y)
Hot	High	False	N
Hot	High	True	N
Hot	High	False	P
Mild	High	False	P
Cold	Normal	False	P
Cold	Normal	True	N
Cold	Normal	True	P
Mild	High	False	N
Cold	Normal	False	N
Mild	Normal	False	P
Mild	Normal	True	P
Mild	High	True	P
Hot	Normal	False	N
Mild	High	True	P

ex) Temperature의 경우 Hot, Mild, Cold 3가지가 존재함. L={Hot}, R={Mild, Cold}로 먼저 봄

	N	P	Total
Left(Hot)	3	1	4
Right(Mild, Cold)	3	7	10
Total	6	8	14

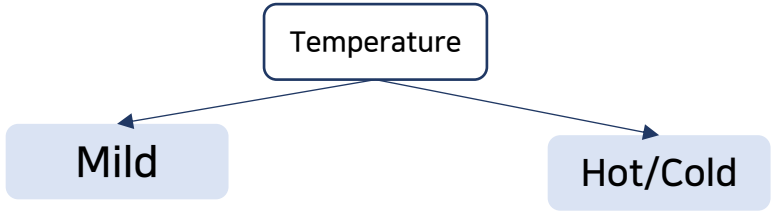
$$G = 1 - (\frac{6}{14})^2 - (\frac{8}{14})^2 = 0.4898$$

$$G_L = 1 - (\frac{3}{4})^2 - (\frac{1}{4})^2 = 0.375$$

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

$$\Delta G = 0.4898 - (\frac{4}{14} \times 0.375 + \frac{10}{14} \times 0.42) = 0.0827$$

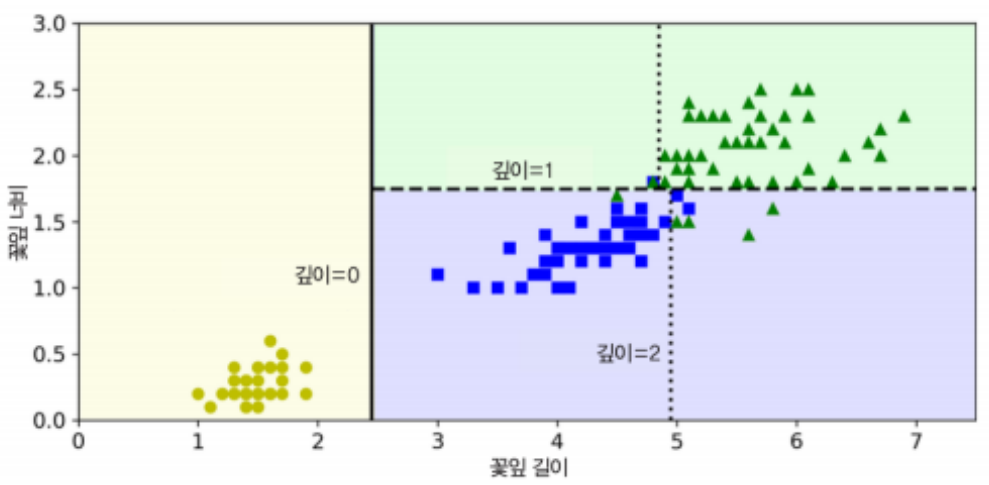
기준	Left	Right	ΔG (최대 선택)	G-ΔG (최소 선택)
Temperature	Hot	Mild, Cold	0.0827	0.4071
Temperature	Mild	Hot, Cold	0.1029	0.3869
Temperature	Cold	Hot, Mild	0.0041	0.4857
Humidity	High	Normal	0.0000	0.4898
Windy	False	True	0.0136	0.4762

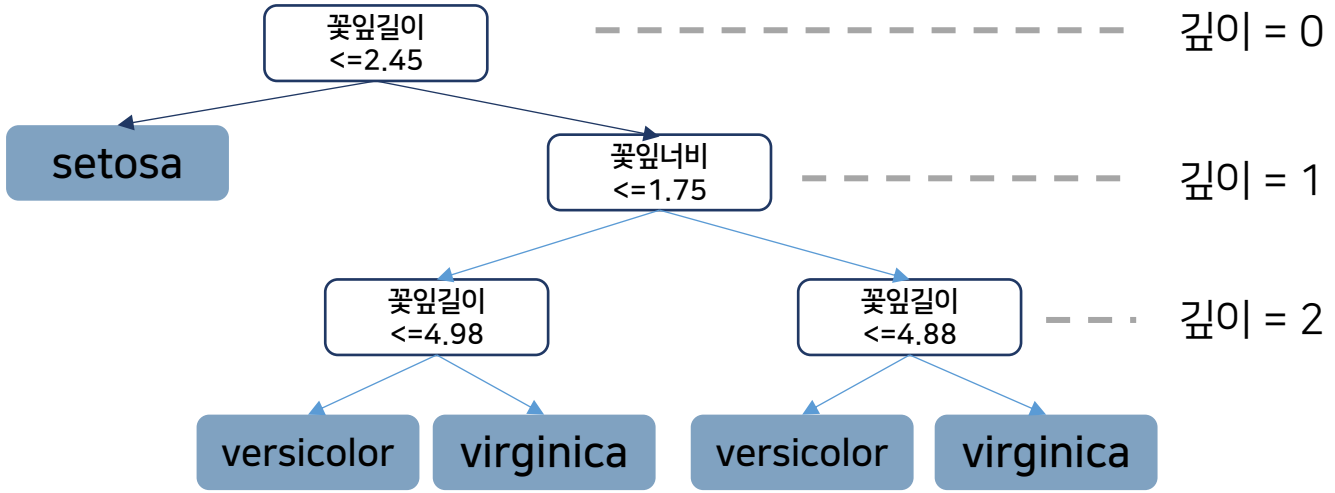


Decision Tree – CART(Classification And Regression Tree)

CART 알고리즘

- **분류 CART 비용함수 =** $\frac{m_{left}}{m} \times G_{left} + \frac{m_{right}}{m} \times G_{right} = G - \Delta G$





Decision Tree – C4.5(Quinlan)

C4.5-정보량(Self-Information)

- 이분류 이외에 클래스가 많아도 다중 분류가 가능함
- 정보량을 활용해 정보 이득지수(Gain Ratio)를 도출하고 이를 활용해 가지 구성
- 정보 이득지수가 크면서도 과도하게 가지를 치지 않고 변수를 선택 가능함

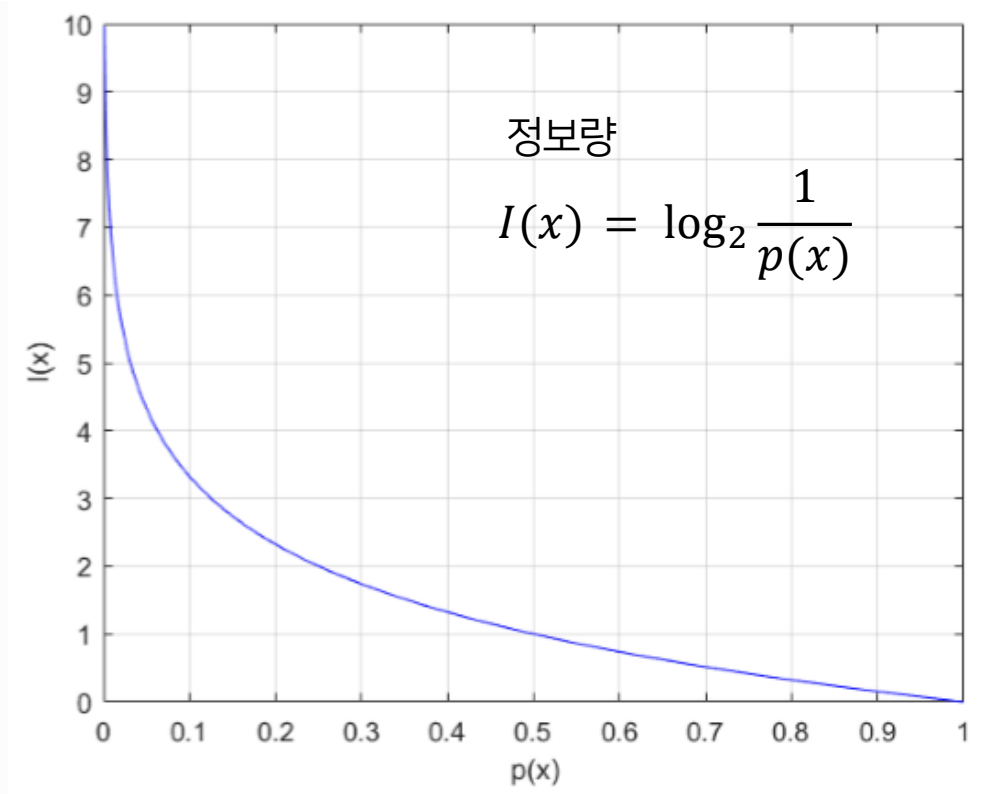
정보량 : 사건 x가 일어났을 때 얻는 정보량(Self-Information)

$$I(x) = \log_2 \frac{1}{p(x)} = -\log_2 p(x)$$

$p(x)$ ↓ 정보량은 커짐

$p(x)$: 사건 x가 일어날 확률

$p(x)$ ↑ 정보량은 작아짐

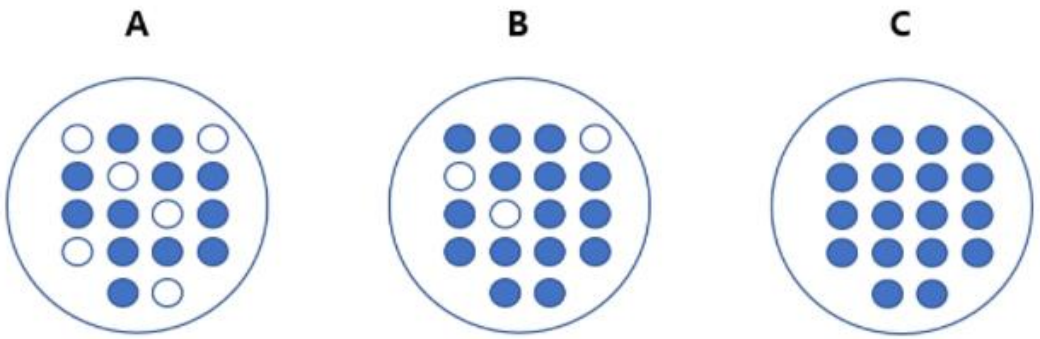


Decision Tree – C4.5(Quinlan)

정보의 균일도란?

- 적은 결정노드로 높은 예측 정확도를 가지려면, 가능한 결정노드로 나뉜 데이터들의 값들이 균일한 데이터 셋을 구성할 수 있도록 해야함
- 결과가 나오는 확률이 얼마나 비슷하게(균일하게) 퍼져 있는가
- 데이터셋의 균일도는 데이터를 구분하는데 필요한 정보의 양에 영향을 미침
- 해당 사건이 얼마나 희귀한가 → 놀람 정도

정보의 균일도 예시



- 균일한 데이터 A, B, C에서 공의 색 분포가 각각 다음과 같을 경우
- A: 여러 색의 공이 비슷한 개수로 섞여 있음(가장균일)
- B : 한 색이 많긴 하지만, 다른 색도 섞여 있음(중간)
- C : 모든 공이 파란공이고, 다른색은 없음(불균일)
- C는 불확실성이 없고, 색을 확인해도 새로운 정보가 없음
- A는 불확실성이 크고, 색을 많이 확인해야 정보를 알 수 있음

  → 불확실성이 커질수록 정보량이 커짐

Decision Tree – C4.5(Quinlan)

C4.5-엔트로피(Entropy)

- 정보량과 정보의 균일도를 하나로 묶어주는 개념
- 무작위 변수가 여러 값 x를 가질 때, 평균적으로 얼마나 많은 정보가 필요한가(불확실한가)를 찾는것 → 불순도

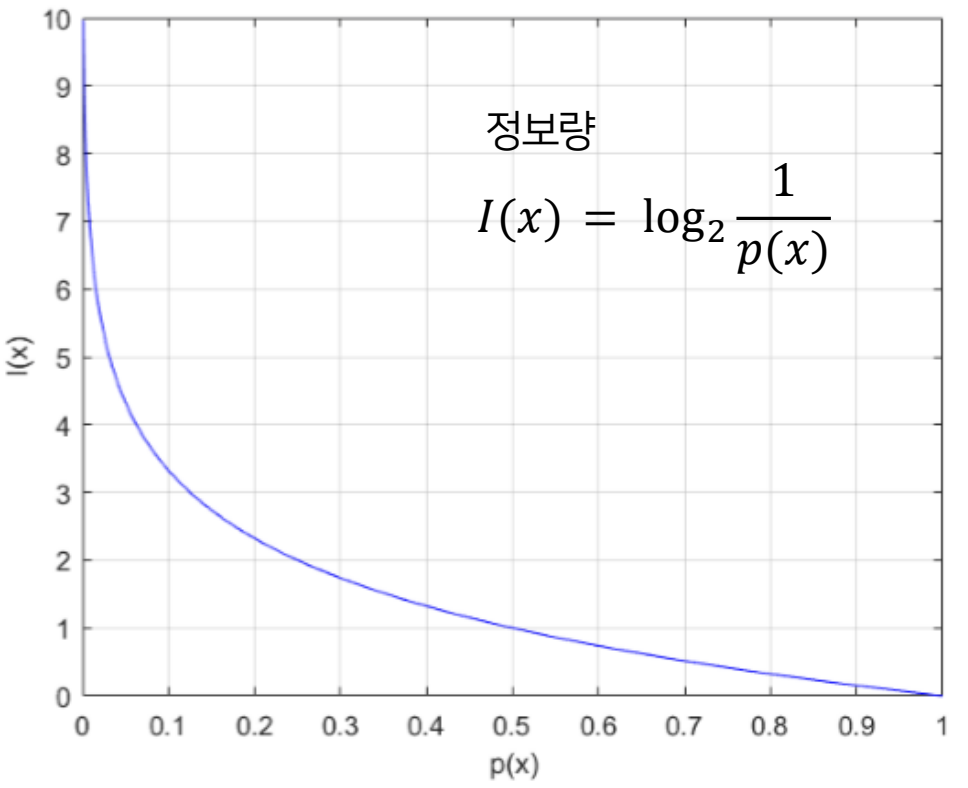
$$I(x) = \log_2 \frac{1}{p(x)} = -\log_2 p(x)$$

$p(x)$: 사건 x가 일어날 확률

$$E[I(x)] = H(x) = \sum_x p(x)I(x) = -\sum_x p(x)\log_2 p(x)$$

$$H(S) = -\sum_{x=1} p(x)\log_2 p(x)$$

각 사건의 정보량X그 사건이 일어날 확률의 합
→ 해당 데이터의 불확실 성의 속성을 보자!



Decision Tree – C4.5(Quinlan)

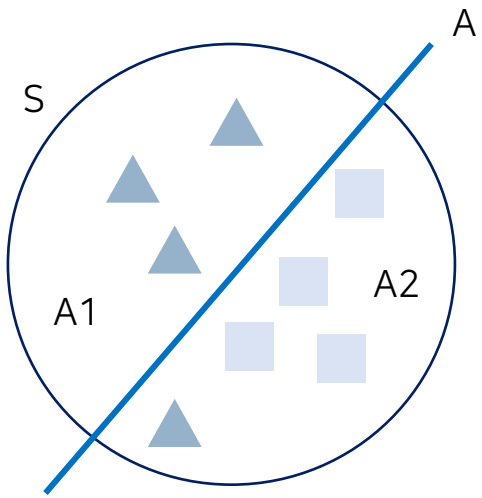
C4.5-엔트로피(Entropy)

- 속성을 본 후의 불확실성 → 조건부 엔트로피

$$H(S) = -\sum_{x=1} p(x)\log_2 p(x)$$

$$H(S|A) = \sum_{x=1} p(x)H(x)$$

A의 값을 알고 난 뒤에도 남아 있는 클래스 불확실성



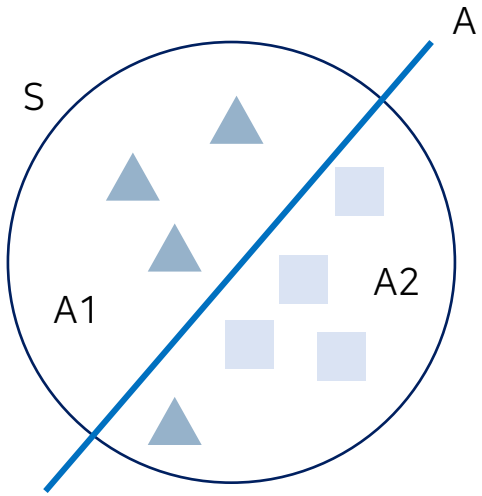
Decision Tree – C4.5(Quinlan)

C4.5-정보이득 지수(Information Gain)

- 정보 이득 지수가 크다는 것은 분할 전보다 분할 후에 엔트로피(불순도)가 많이 줄었다는 의미
- 정보 이득 지수가 크다는 것은 자식 노드들이 더 순수해졌다는 의미

$$H(S) = -\sum_{x=1} p(x)\log_2 p(x)$$

$$H(S|A) = \sum_{x=1} p(x)H(x)$$ A의 값을 알고 난 뒤에도 남아 있는 클래스 불확실성



$$IG(S,A) = H(S) - H(S|A)$$

속성 A로 분할했을 때, 얼마나 불확실성이 줄었는가를 보기 위해

Decision Tree – C4.5(Quinlan)

엔트로피

- 주어진 데이터 집합의 혼잡도를 의미함
- 서로 다른 값이 섞여 있으면(혼잡할 시) 엔트로피가 높고, 같은 값이 많을 시(균일할 시) 엔트로피가 낮음

$$\mathrm{H}(S) = -\sum_i p(x)\log_2 p(x)$$

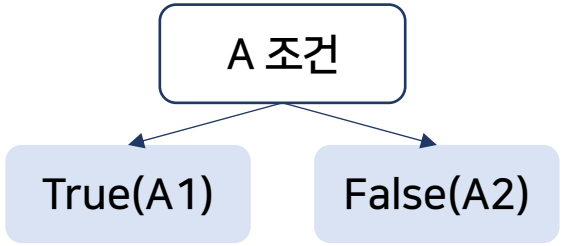
정보이득(Information gain)지수

- 주어진 데이터 집합의 균일한 정도를 나타냄
- 서로 다른 값이 섞여 있으면(혼잡할 시) 정보이득지수가 낮고, 같은 값이 많을 시(균일할 시) 정보이득지수가 높음
- 즉, 결정 트리에서는 정보이득지수가 높은 속성을 기준으로 분할함

$$\mathrm{IG}(S, A) = \mathrm{H}(S) - \sum_t p(t)H(t)$$

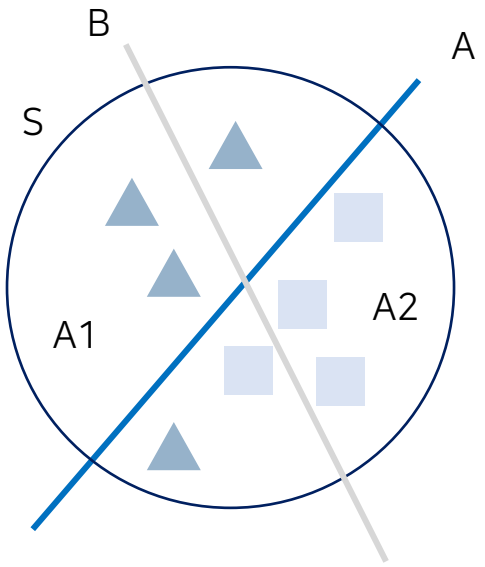
$\sum_t p(t)H(t)$ → H(S|A)

p(t) : 양쪽 가지로 나뉘는 확률 → 가중치로 사용



Entropy at parent level

H(t)



$$\mathrm{H}(S) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$\mathrm{H}(A1) = -\left(\frac{3}{3}\log_2\frac{3}{3} + \frac{0}{3}\log_2\frac{0}{3}\right) = 0$$

$$\mathrm{H}(A2) = -\left(\frac{1}{5}\log_2\frac{1}{5} + \frac{4}{5}\log_2\frac{4}{5}\right) = 0.721$$

$$\mathrm{H}(S|A) = -\left(\frac{3}{8}\times 0 + \frac{5}{8}\times 0.721\right) = 0.451$$

$$\mathrm{IG}(S, A) = 1 - \left(\frac{3}{8}\times 0 + \frac{5}{8}\times 0.721\right) = 0.549$$

불순도가 줄어듦 : 1 → 0.451

Decision Tree

CART 알고리즘

- 분류와 회귀 두가지 다 가능함
- 지니 계수(Gini Index)를 이용해 데이터를 분류함
- 가지 분기 시, 이진 분류로 두개의 자식 노드만 가능함

C4.5 알고리즘

- 회귀 모형을 만들 수 없음
- 정보이득 지수 (Information Gain)를 이용해 분류함
- 결측치가 포함된 데이터에도 사용 가능함

Decision Tree – C4.5(Quinlan)

C4.5 – 정보이득지수의 장점

o 연속형 독립변수 일 때도 사용이 가능함

Temperature	Play
36	No
35	No
22	Yes
24	Yes
20	Yes
35	No
31	Yes
30	No
29	Yes
28	Yes
25	Yes
26	Yes
29	Yes
27	No

Method 1) 값이 바뀌는 모든 지점을 Point로 설정

Method 2) Output이 바뀔 때만 Point로 설정

Method 3) Q1, Median, Q3값을 Point로 설정

$H(Play) = -(\frac{9}{14}\log_2\frac{9}{14} + \frac{5}{14}\log_2\frac{5}{14})$

$H(Play, Temperature(21)) = -\frac{13}{14}(\frac{8}{13}\log_2\frac{8}{13} + \frac{5}{13}\log_2\frac{5}{13})$

$IG(Play, Temperature(21)) = H(Play) - H(Play, Temperature(21))$

Method 1

	Temperature	Play
	20	Yes
21	22	Yes
23	24	Yes
24.5	25	Yes
25.5	26	Yes
26.5	27	No
27.5	28	Yes
28.5	29	Yes
	29	Yes
29.5	30	No
30.5	31	Yes
33	35	No
	35	No
35.5	36	No

Method 2

	Temperature	Play
	20	Yes
	22	Yes
	24	Yes
	25	Yes
	26	Yes
26.5	27	No
27.5	28	Yes
	29	Yes
	29	Yes
29.5	30	No
30.5	31	Yes
33	35	No
	35	No
	36	No

Method 3

	Temperature	Play
	20	Yes
	22	Yes
	24	Yes
	25	Yes
25.25	26	Yes
	27	No
	28	Yes
28.5	29	Yes
	29	Yes
	30	No
30.75	31	Yes
	35	No
	35	No
	36	No

$IG(Play, Temperature(21)) = 0.04742$

$IG(Play, Temperature(23)) = 0.1001$

$IG(Play, Temperature(24.5)) = 0.1590$

$IG(Play, Temperature(25.5)) = 0.2257$

$IG(Play, Temperature(26.5)) = 0.3029$

$IG(Play, Temperature(27.5)) = 0.09000$

$IG(Play, Temperature(28.5)) = 0.1516$

$IG(Play, Temperature(29.5)) = 0.3586$

$IG(Play, Temperature(30.5)) = 0.1925$

$IG(Play, Temperature(33)) = 0.4009$

$IG(Play, Temperature(35.5)) = 0.2863$

Decision Tree – C4.5(Quinlan)

C4.5 – 정보이득지수의 장점

o 결측치가 포함된 데이터에도 사용이 가능함

Outlook	Play
Sunny	No
	No
	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
	No
	Yes
	Yes
	Yes
Overcast	Yes
Overcast	Yes
Rain	No

1단계 : Entropy는 Non-missing value로만 계산함

$$H(Play) = -\sum_{i=1}^{C} p_i log_2 p_i$$
$$= -(\frac{3}{8} log_2 \frac{3}{8} + \frac{5}{8} log_2 \frac{5}{8}) \quad = 0.9544$$

$$H(Play, Outlook) = \sum p(t) H(t)$$
$$= \frac{1}{8} H(1,0) + \frac{4}{8} H(2,2) + \frac{3}{8} H(0,3)$$
$$= 0.5$$

$$IG(Play, Outlook) = H(Play) - H(Play, Outlook)$$
$$= 0.4544$$

2단계 : Information Gain → Weighted Information Gain

$F = \frac{8}{14}$ → 전체 데이터 중에 결측치가 아닌 데이터의 비율

$$Weighted\ Information\ Gain = \frac{8}{14} \times 0.4544 \ = 0.2597$$

3단계 : 결측치를 하나의 클래스로 보고 IV 및 Information Gain Ratio을 계산함

$$IV = -(\frac{1}{14} log_2 \frac{1}{14} + \frac{4}{14} log_2 \frac{4}{14} + \frac{3}{14} log_2 \frac{3}{14} + \frac{6}{14} log_2 \frac{6}{14}) = 1.659$$

$$Information\ Gain\ Ratio = \frac{0.2597}{1.659} \ = 0.1565$$

Decision Tree – C4.5(Quinlan)

C4.5 – 정보이득지수의 장점

o 이득율을 이용한 희귀 집단 판단

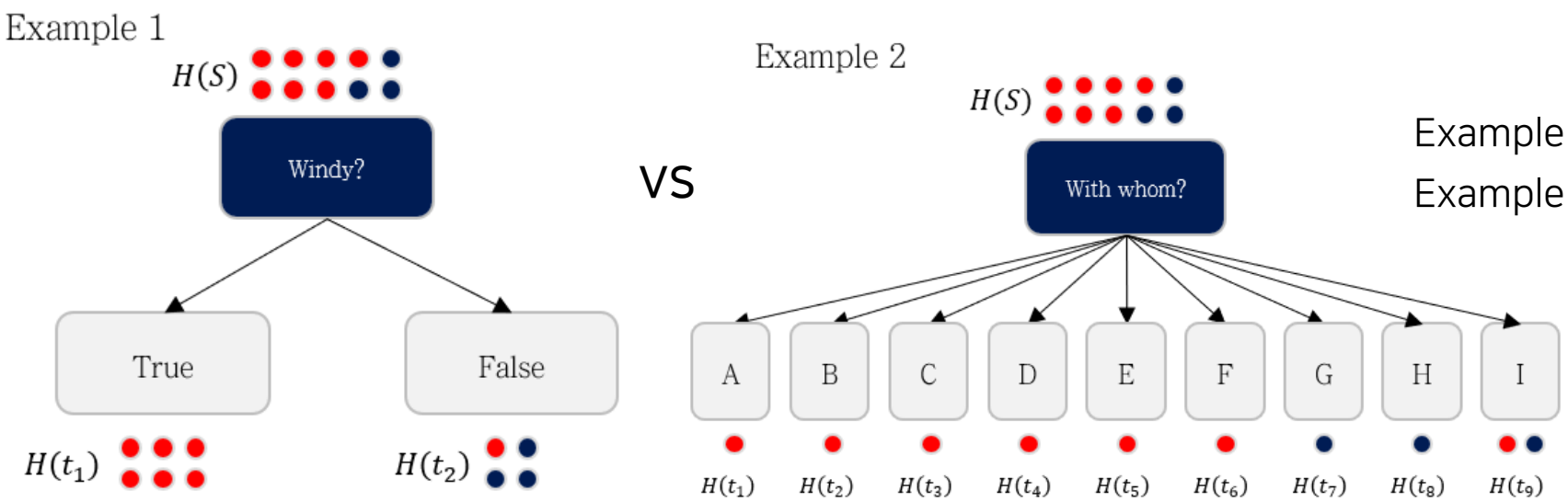
- 따라서, '이득율(Gain ratio)'를 이용하고자 함 → 희귀질환 판단 1:99와 같은 민감한 클래스에 더 잘 반응함
- 특정 지표로 분기했을 때, 생성되는 가지의 수를 N이라고 하고, i번째 가지에 해당하는 확률을 $p(i)$ 라고 함

$$\text{IV} = \text{entropy} = -\sum_{i} p(i)\log_2 p(i)$$

- 이득율(Gain ratio) = IG / IV

Example 1 IV = $-(\frac{6}{10}\log_2\frac{6}{10}+\frac{4}{10}\log_2\frac{4}{10})$ = 0.9701

Example 2 IV = $-(\frac{1}{10}\log_2\frac{1}{10}+\frac{1}{10}\log_2\frac{1}{10}+\ \ldots+\frac{2}{10}\log_2\frac{2}{10})$ = 3.1219



Example 1) IG/IV = 0.5568/0.9701 = 0.5739

Example 2) IG/IV = 0.6818/3.1219 = 0.2182

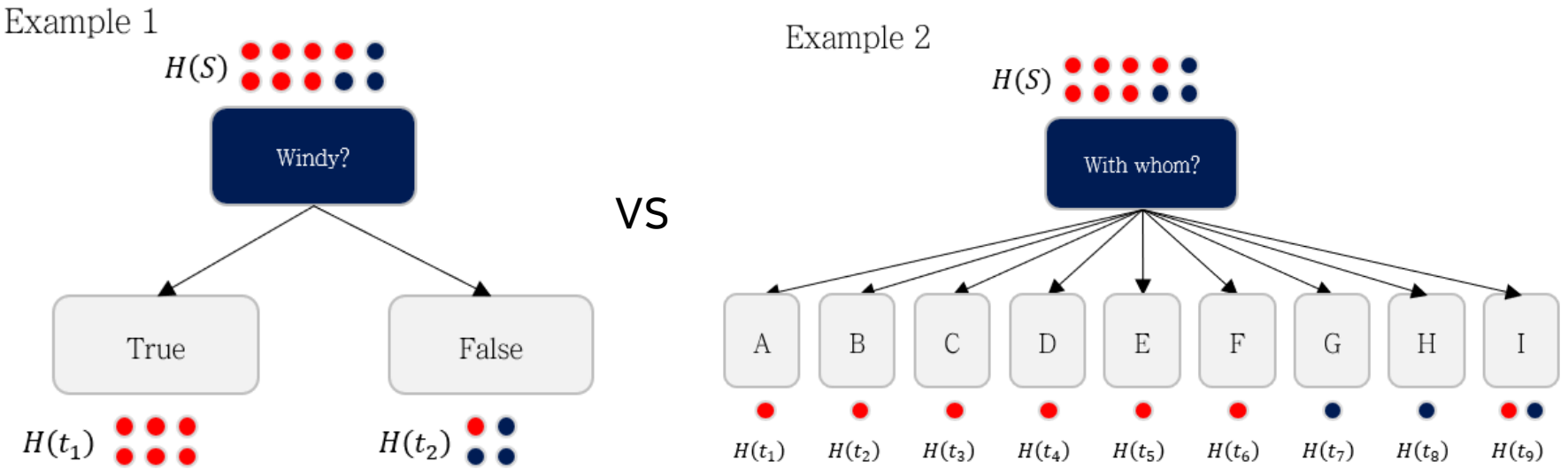
Decision Tree – C4.5(Quinlan)

C4.5 – 정보이득지수의 단점

o 정보이득지수의 단점

- 정보이득지수(Information gain)의 한계가 존재함
- 많은 범주를 가지는 요소에 치중되는 경향이 존재함
- 중요하지 않은 분할이 발생할 수 있음

o 정보이득지수의 한계 $IG(S,A) = H(S) - \sum_t p(t)H(t)$



Example 1) IG = 0.5568

Example 2) IG = 0.6818

진짜 'with whom?'을 기준으로 나누는 것이 더 맞는가?

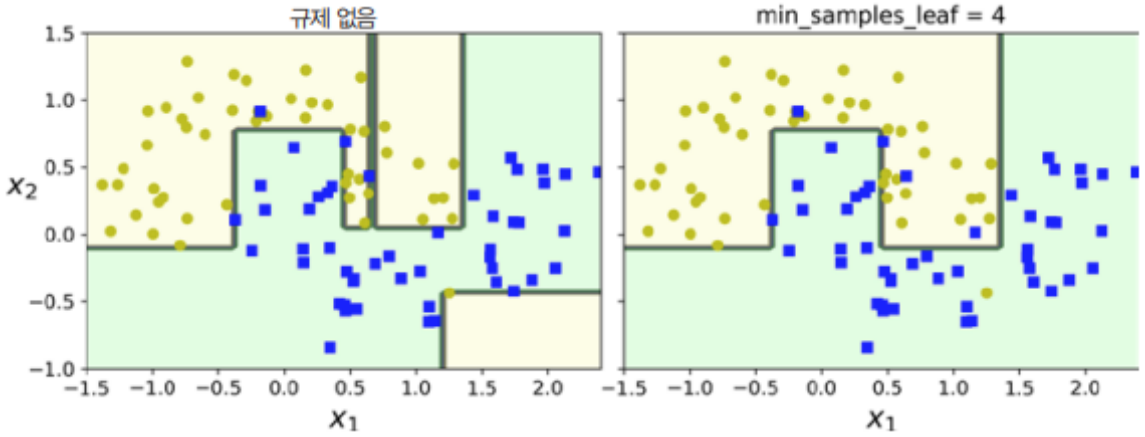
Decision Tree – C4.5(Quinlan)

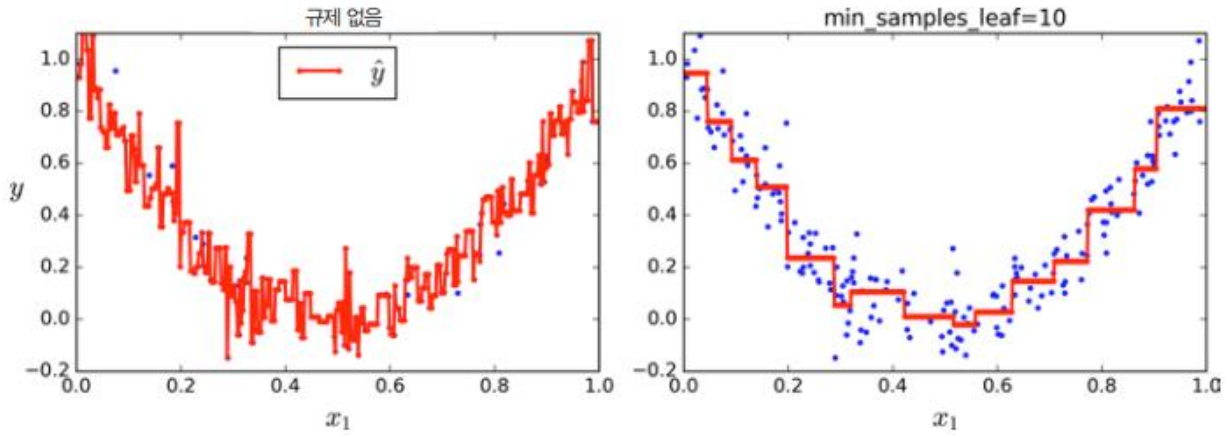
결정트리 특징 및 한계

o Overfitting(과적합)

- 가지수를 너무 많이 나누게 되면, overfitting 가능성이 높음
- Training set은 잘 설명 가능하지만, 새로운 데이터가 들어왔을 시 설명을 잘 못하는 경우가 존재함
- min_samples_split, min_samples_leaf, max_depth, max_leaf_node 등 하이퍼 파라미터 조정 필요함

- Min_samples_split : 노드를 분할하기 위한 최소한의 샘플 데이터 수
- Min_samples_leaf : 리프노드가 되기위한 최소한의 샘플 데이터 수
- Max_depth : 트리의 최대 깊이
- max_leaf_node : 리프 노드의 최대 개수





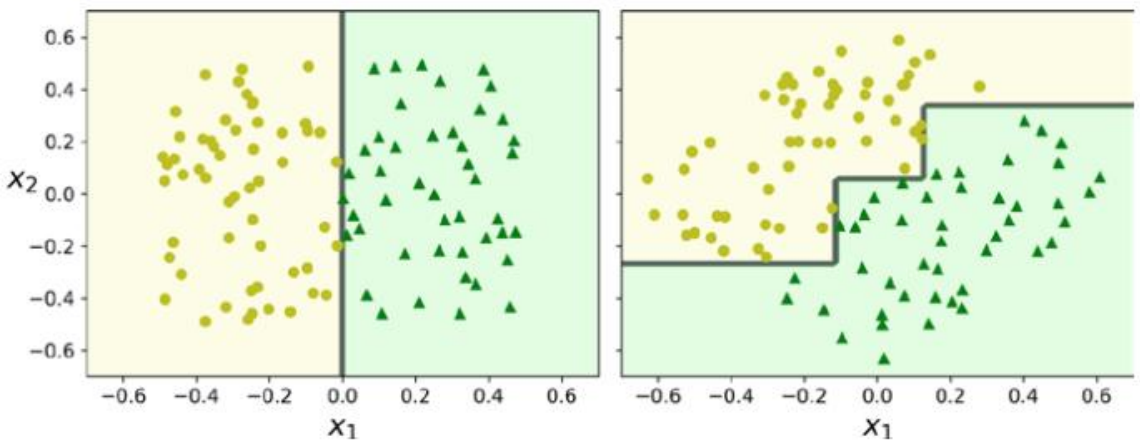
Decision Tree – C4.5(Quinlan)

결정트리 특징 및 한계

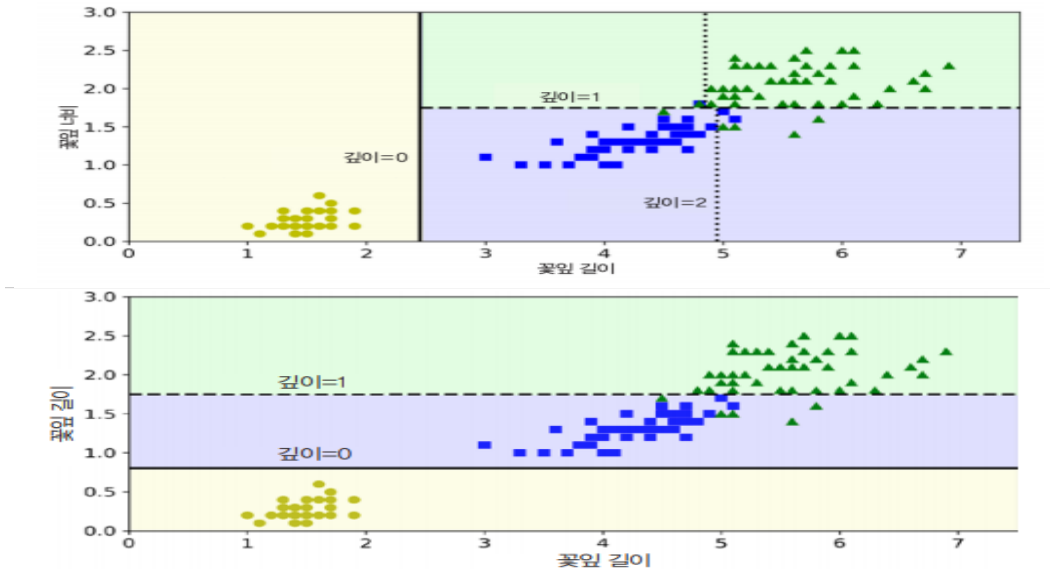
o 불안정성

- 결정트리의 경우 계단모양의 결정 경계를 만듦(x축에 수평 및 수직으로만 표현 가능)
- 1. 훈련 세트의 회전에 민감함 → 훈련 데이터를 더 좋은 방향으로 회전시키는 PCA 기법 사용
- 2. 훈련 데이터의 작은 변화에도 매우 민감함

1. 왼쪽 데이터 셋을 45도 회전함



2. 꽃잎 길이 4.8cm, 너비1.8cm 데이터 제거



Decision Tree – C4.5(Quinlan)

C4.5

o 가지치기(pruning) 과정이 존재함

- 사후 가지치기(post-pruning)로 결정트리를 완성한 다음 가지치기를 수행함

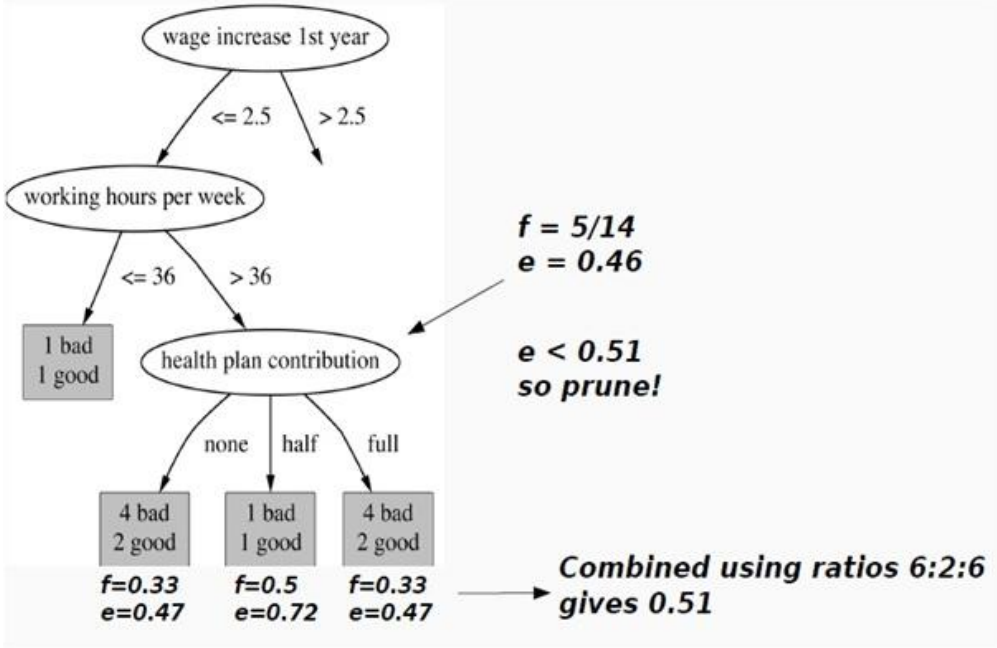
  - 0단계 : 기존 데이터를 나눌 때, Training/test 데이터로 나누는 것이 아닌 Training/Validation(pruning)/test로 나눔
  - 1단계 : Training 데이터만 이용하여 결정트리를 구성함
  - 2단계 : pruning 데이터를 이용해 가지치기를 수행함
    자식노드의 e값 weight sum > 부모노드의 e값 비교 ➔ 가지치기 수행

$$e = \frac{f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}$$

$N = sample\ size$

$f = Error\ rate$

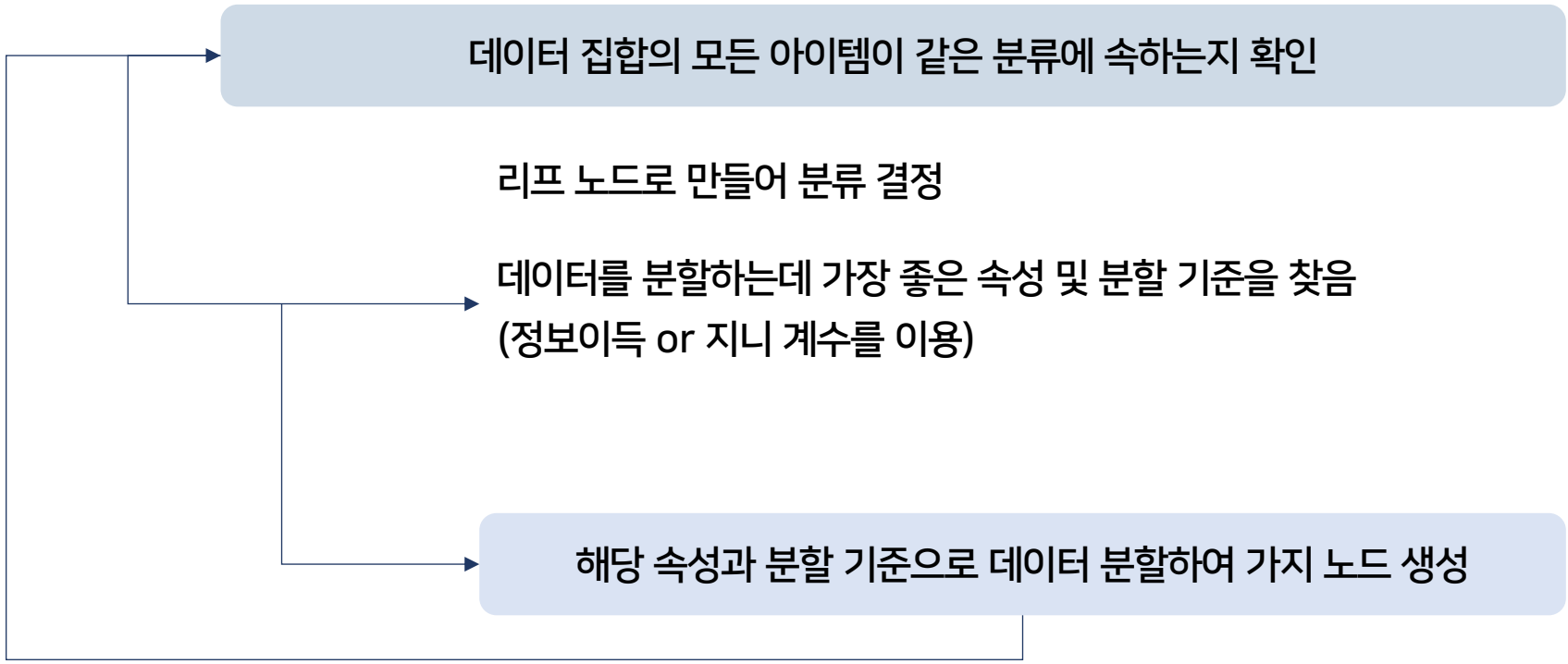
$z = z\ score\ (default\ z = 0.69)$



$\frac{6}{14}$ X 0.47 + $\frac{2}{14}$X0.72 + $\frac{6}{14}$X0.33

Decision Tree

결정트리 분류 메커니즘



Decision Tree(이분류)

```
library(caret)
library(rpart) #CART모델
library(rpart.plot)
library(skimr)

data(mtcars)
skim(mtcars)

mtcars[sapply(mtcars, is.character)] <- lapply(mtcars[sapply(mtcars, is.character)], as.factor)
mtcars$am <- as.factor(mtcars$am)

train_indices <- createDataPartition(mtcars$am, p = 0.8, list = FALSE)
train_data <- mtcars[train_indices, ]
test_data <- mtcars[-train_indices, ]
```

Decision Tree(이분류)

```
#Hyper parameters 설정
control_params <- rpart.control(cp = 0.01, maxdepth = 5, minsplit = 10)
#cp(분할될 때 마다 모델이 최소 1%는 개선이 되어야 함)
#maxdepth(트리의 깊이가 최대값), minsplit(각각의 노드에 최소 10개의 관측치가 있도록 분할)

#모델 생성
tree_model <- rpart(am ~ ., data = train_data, method = "class", control = control_params)

#모델 결과
summary(tree_model)
rpart.plot(tree_model)

predicted_probs <- predict(tree_model, test_data, type = "prob")[,2]
predicted_probs

predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)
predicted_classes <- as.factor(predicted_classes)
predicted_classes
```

Decision Tree(이분류)

```
Call:
rpart(formula = am ~ ., data = train_data, method = "class",
    control = control_params)
  n= 27

          CP nsplit  rel error    xerror       xstd
1 0.81818182      0 1.00000000 1.0000000 0.2321035
2 0.09090909      1 0.18181818 0.5454545 0.1963861
3 0.01000000      2 0.09090909 0.3636364 0.1678106

Variable importance
  wt disp drat  mpg  cyl   hp qsec
  20   17   15   15   14   14    4
```

Nsplit : 몇 번을 나눴을 때, xerror : 에러가 가장 작은것이 좋은 나눔

어떤 변수가 가장 중요하게(많이) 사용되었는지를 나타내는 지표로 값이 클수록 많이 사용

Decision Tree(이분류)

```
Node number 1: 27 observations,    complexity param=0.8181818
  predicted class=0  expected loss=0.4074074  P(node) =1
    class counts:    16    11
   probabilities: 0.593 0.407
  left son=2 (14 obs) right son=3 (13 obs)
  Primary splits:
      wt   < 3.18   to the right, improve=9.652422, (0 missing)
      drat < 3.75   to the left,  improve=8.322751, (0 missing)
      gear < 3.5    to the left,  improve=7.170370, (0 missing)
      disp < 163.8  to the right, improve=6.564510, (0 missing)
      mpg  < 20.1   to the left,  improve=5.476597, (0 missing)
  Surrogate splits:
      disp < 163.8  to the right, agree=0.926, adj=0.846, (0 split)
      mpg  < 20.1   to the left,  agree=0.889, adj=0.769, (0 split)
      drat < 3.695  to the left,  agree=0.889, adj=0.769, (0 split)
      cyl  < 5      to the right, agree=0.852, adj=0.692, (0 split)
      hp   < 118    to the right, agree=0.852, adj=0.692, (0 split)

Node number 2: 14 observations
  predicted class=0  expected loss=0  P(node) =0.5185185
    class counts:    14     0
   probabilities: 1.000 0.000

Node number 3: 13 observations,    complexity param=0.09090909
  predicted class=1  expected loss=0.1538462  P(node) =0.4814815
    class counts:     2    11
   probabilities: 0.154 0.846
  left son=6 (3 obs) right son=7 (10 obs)
  Primary splits:
      qsec < 19.685 to the right, improve=2.0512820, (0 missing)
      drat < 4      to the left,  improve=0.7179487, (0 missing)
      wt   < 2.3925 to the right, improve=0.7179487, (0 missing)
      mpg  < 24.4   to the left,  improve=0.5274725, (0 missing)
      disp < 114.05 to the right, improve=0.5274725, (0 missing)

Node number 6: 3 observations
  predicted class=0  expected loss=0.3333333  P(node) =0.1111111
    class counts:     2     1
   probabilities: 0.667 0.333

Node number 7: 10 observations
  predicted class=1  expected loss=0  P(node) =0.3703704
    class counts:     0    10
   probabilities: 0.000 1.000
```

왼쪽, 오른쪽 순으로 각 노드별로 어떤 변수들이 활용 됐고, 몇 개의 가지와 노드가 존재하는지 알려줌

Decision Tree(이분류)

의사결정 나무-이분류

```
library(pROC)

#Confusion Matrix 도출
confusion_matrix <- confusionMatrix(predicted_classes, test_data$am)
print(confusion_matrix)

#ROC Curve
roc_curve <- roc(test_data$am, predicted_probs)
plot(roc_curve)
auc(roc_curve)
```

Decision Tree(다중분류-C4.5)

의사결정 나무-다중분류

```
df=read.csv("customer.csv")

# 1) 문자형 컬럼 공백 제거 후 factor로 변환
char_cols <- sapply(df, is.character)
df[char_cols] <- lapply(df[char_cols], function(x) {
  x <- trimws(x)        # 앞뒤 공백 제거
  x[x == ""] <- NA      # 완전 빈 문자열은 NA로 처리
  factor(x)
})

# 2) factor 레벨을 make.names 로 안전하게 변환
for (v in names(df)) {
  if (is.factor(df[[v]])) {
    lv <- levels(df[[v]])
    lv[lv == ""] <- "missing"
    lv <- make.names(lv)
    levels(df[[v]]) <- lv
  }
}
```

..High..	High

데이터에 콤마가 들어가는 문제

Decision Tree(다중분류-C4.5)

의사결정 나무-다중분류

```
# 3) 컬럼 이름도 안전하게
names(df) <- make.names(names(df), unique = TRUE)
```

변수 이름에 공백, 괄호, 하이픈, 한글+특수문자 조합 등이 있는 경우

```
# 4) 타겟(종속변수)에 NA 있으면 제거
df <- df[!is.na(df$Segmentation), ]
df$Segmentation <- droplevels(df$Segmentation)
```

독립변수의 값이 없으면 상관없지만 종속변수의 값이 없으면 오류 발생

Decision Tree(다중분류-C4.5)

의사결정 나무-다중분류

```
# 5) 모델 및 예측

library(C50)     # C5.0 (C4.5 후속)
library(caret)
library(pROC)

skim(df)

df[sapply(df, is.character)] <- lapply(df[sapply(df, is.character)], as.factor)

train_indices <- createDataPartition(df$Segmentation, p = 0.8, list = FALSE)
train_data <- df[train_indices, ]
test_data <- df[-train_indices, ]
```

Decision Tree(다중분류-C4.5)

의사결정 나무-다중분류

```
#C5.0 (C4.5 계열) 하이퍼파라미터 설정
# - trials: boosting 횟수 (1이면 순수 트리: C4.5 스타일)
# - CF (confidence factor): 가지치기 강도 (0~1, 작을수록 강한 가지치기, 기본 0.25)
# - minCases: 리프 노드 최소 샘플 수

ctrl_c50 <- C5.0Control(
  CF       = 0.25,   # 비관적 오류 기반 가지치기 강도
  minCases = 2,
  winnow   = FALSE   # 변수 선택 사용 여부 (TRUE면 자동 변수선택)
)

#모델 학습
c50_model <- C5.0(Segmentation ~ ., data = train_data, control = ctrl_c50, trials = 1)
```

Decision Tree(다중분류-C4.5)

의사결정 나무-다중분류

```
# 모델 결과 확인
summary(c50_model)  # 규칙, 노드 수, 오류율, 사용된 변수 등 출력
#plot(c50_model)     # 트리 시각화(복잡하면 생성속도가 느림)
```

```
Evaluation on training data (8558 cases):

        Decision Tree
      ----------------
      Size      Errors

       543 3395(39.7%)   <<


     (a)   (b)   (c)   (d)    <-classified as
    ----  ----  ----  ----
    1325   277   198   455    (a): class A
     374   887   374   292    (b): class B
     248   281  1169   256    (c): class C
     377   139   124  1782    (d): class D


Attribute usage:

100.00% Age
 88.63% Profession
 79.57% Spending
 64.57% Graduated
 63.40% Var
 35.48% Married
 31.82% Gender
 31.32% Family
 18.82% Work
```

Size : 리프 갯수

사용된 변수의 빈도

Decision Tree(다중분류-C4.5)

의사결정 나무-다중분류

```
# 예측 (확률 & 클래스)
pred_prob  <- predict(c50_model, newdata = test_data, type = "prob")
pred_prob
pred_class <- predict(c50_model, newdata = test_data, type = "class")
pred_class <- as.factor(pred_class)
pred_class

# Confusion Matrix & 정확도
conf_mat <- confusionMatrix(pred_class, test_data$Segmentation)
print(conf_mat)

# ROC Curve
mc_roc <- multiclass.roc(response  = test_data$Segmentation,
                         predictor = pred_prob)

mc_roc$auc      # 다중클래스 전체 AUC 값
```

Decision Tree(다중분류-C4.5)

의사결정 나무-다중분류

```
# 6) 클래스별 ROC 커브
classes <- levels(test_data$Segmentation)
roc_list <- list()

for (cl in classes) {
  response_bin <- factor(test_data$Segmentation == cl,
                         levels = c(FALSE, TRUE),
                         labels = c("other", cl))
  prob_cl <- pred_prob[, cl]
  roc_list[[cl]] <- roc(response_bin, prob_cl, levels = c("other", cl))
}

plot(roc_list[[1]], col = 1, lwd = 2,
     main = "One-vs-Rest ROC Curves (C5.0, 4-class)")
if (length(classes) > 1) {
  for (i in 2:length(classes)) {
    plot(roc_list[[i]], col = i, lwd = 2, add = TRUE)
  }
}
legend("bottomright",
       legend = paste0(classes, " (AUC=", round(sapply(roc_list, auc), 3), ")"),
       col    = seq_along(classes),
       lwd    = 2, cex = 0.8)
```