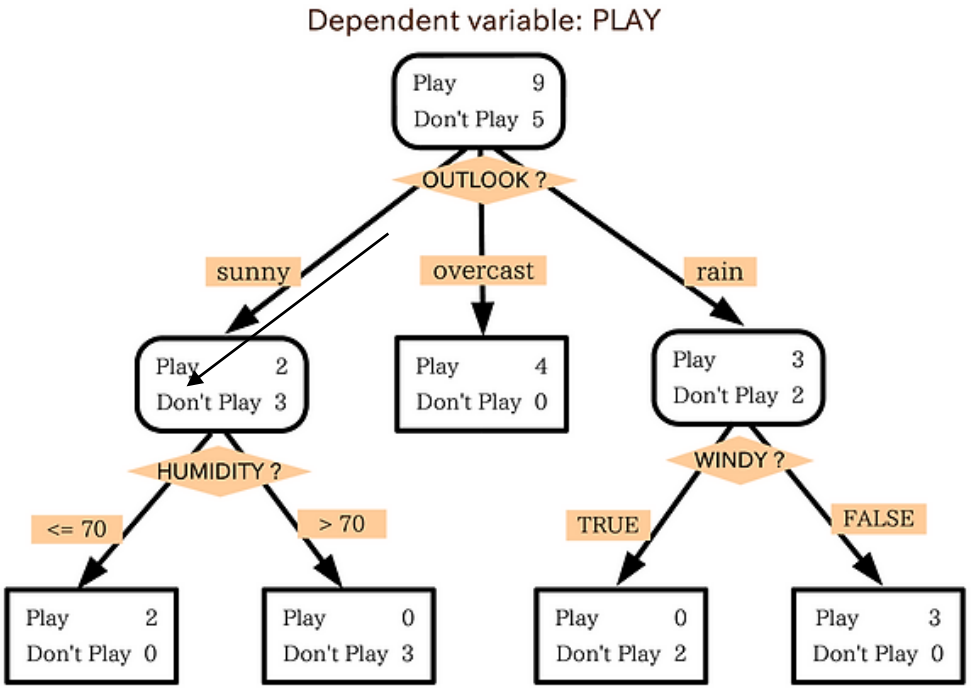
의사결정나무 (Decision Tree)

의사결정나무

- 질문을 던져서 맞고 틀리는 것에 따라 우리가 생각하고 있는 대상을 좁혀나가는 방법과 유사
- “스무고개” 놀이와 비슷한 개념

의사결정나무

- If-then 형식으로 알고리즘이 작동하여, 형성된 결과 물은 나무의 형태로 표현됨



규칙 예시

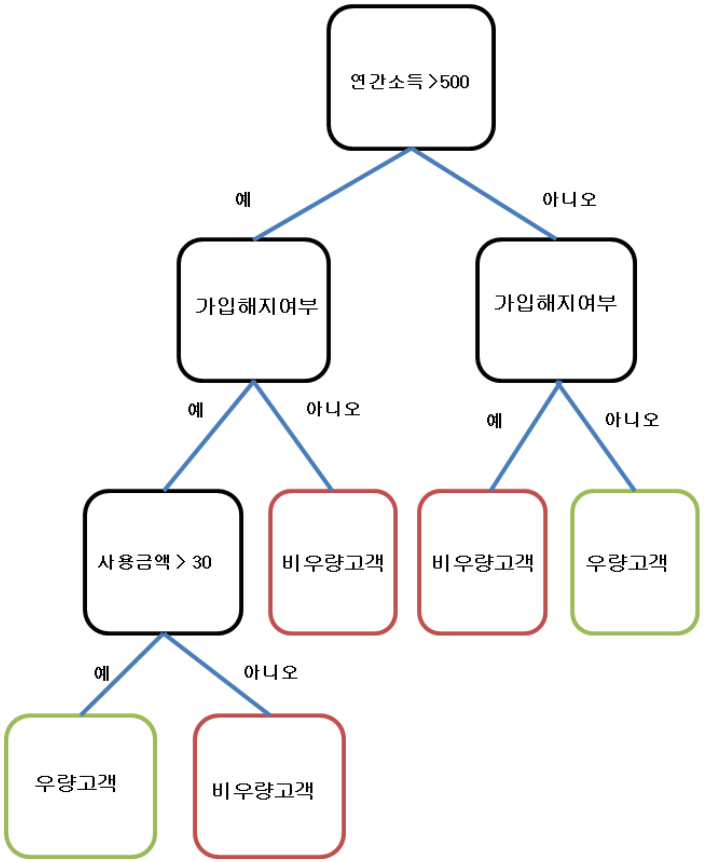
만일 내일 날씨가 맑고 습도가 70% 이하이면 아이는 밖에 나가서 놀 것이다.

or

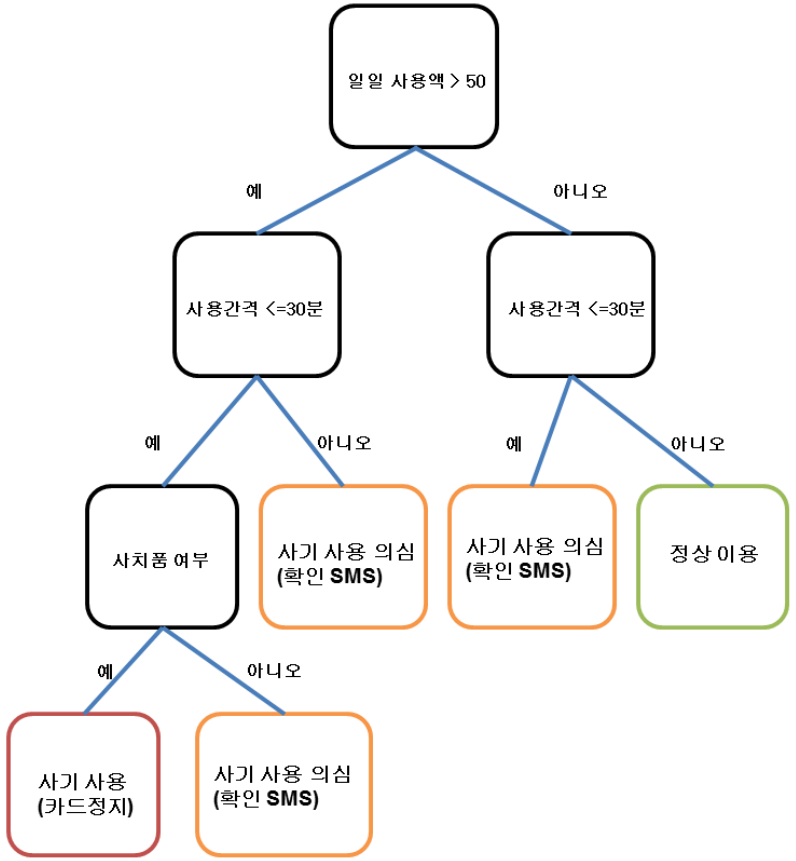
만일 내일 비가 오고 바람이 불면 아이는 밖에 나가 놀지 않을 것이다.

활용 예시 – 서비스 업체

1) 우량고객 평가 모델(Customer value evaluation)



2) 카드 사기 방지 모델 (Fraud detection)



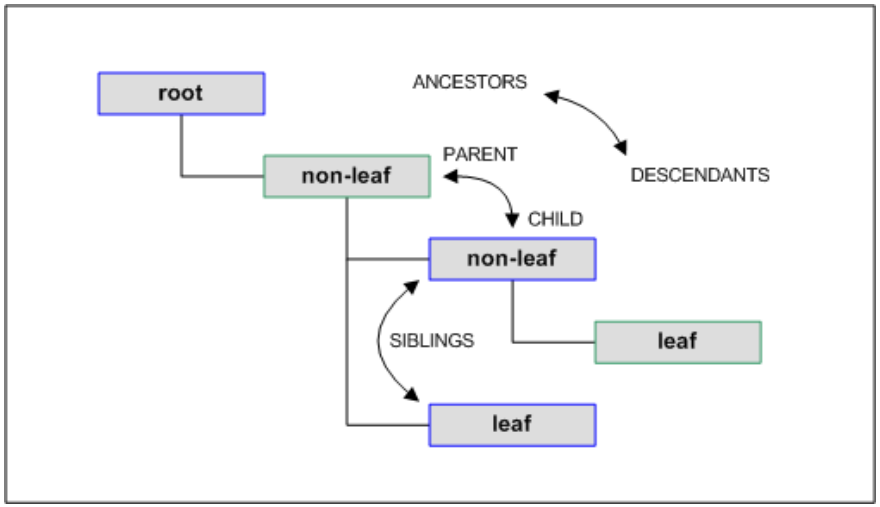
의사결정나무 유형

- 의사결정나무 모델은 분류(Classification)/회귀(Regression) 문제에 모두 사용 가능하다

- 분류 의사결정나무(Classification Tree)는 범주형 반응변수를 예측하는데 사용

- 회귀 의사결정나무(Regression Tree)는 연속형 반응변수를 예측하는데 사용

의사결정나무 용어



- Parent node: 분할 전의 노드
- Child node: 분할 후의 노드
- Root node: 처음 시작 노드 - child node만 가지고 있음
- Leaf nodes: 마지막 노드 - parent node만 가지고 있음
- Split criterion: 노드를 분할하는데 사용한 변수값

  →예) 사용간격 = 30분

왜 사용할까?

- 간단하며 직관적이고 결과에 대한 해석이 가능하다

  → If ~ then ~

- 모델 학습을 위한 데이터 전처리가 비교적 간단하다

- Numerical, Categorical 데이터를 모두 다룰 수 있다

  * 하지만, scikit-learn의 경우에는 categorical 변수를 더미변수로 만들어줘야 한다

Key Ideas

1. Recursive Partitioning

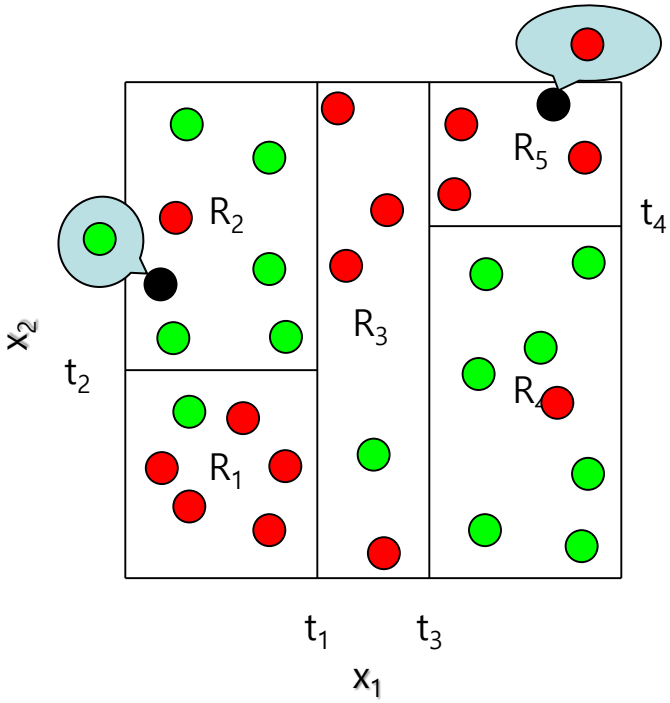
- 데이터 공간을 반복적으로 2개로 나눈다
- 새롭게 나뉘어진 공간 안에 있는 데이터들의 동종성 (homogeneity)를 최대화 할 수 있는 방향으로 나눈다

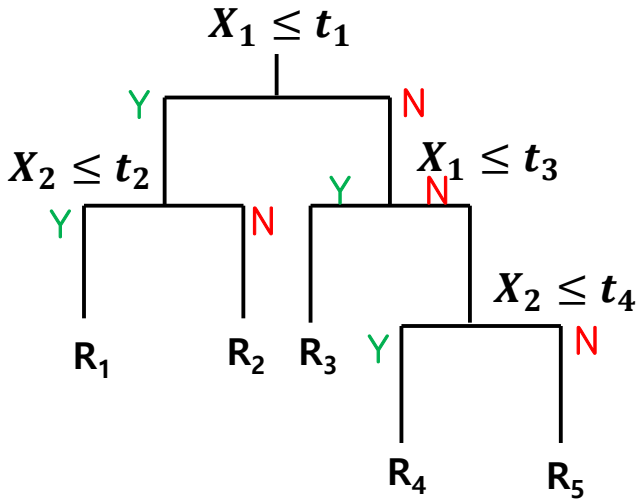
2. Pruning Tree

- 과적합을 막기 위한 방법
- 학습, 구축된 의사결정나무 모델을 보다 단순화하여 불필요하게 '과하게 학습'되는 것을 방지하는 방법

Recursive Partitioning

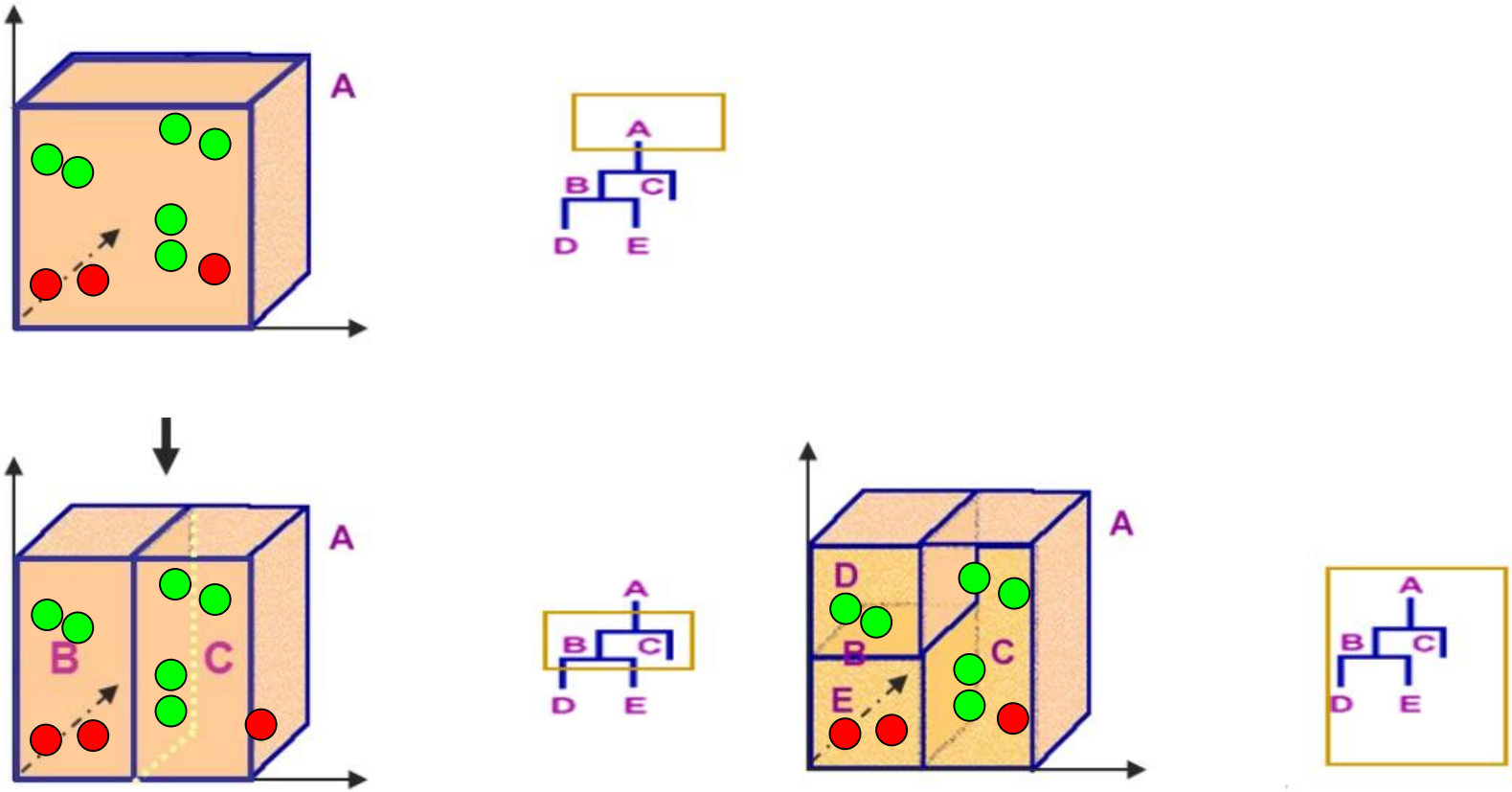
- 입력 변수의 영역을 두 개로 구분
- 구분하기 전보다 구분된 뒤에 각 영역의 순도(purity, homogeneity)가 증가하도록





Recursive Partitioning

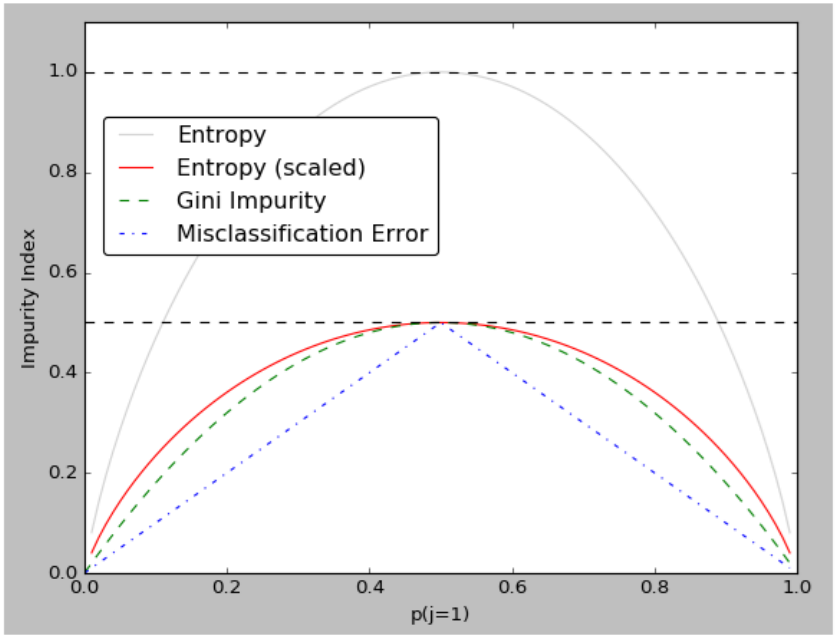
- 입력 변수의 영역을 두 개로 구분
- 구분하기 전보다 구분된 뒤에 각 영역의 순도(purity, homogeneity)가 증가하도록



Classification Tree – Impurity Measure

- 필요한 것: 영역의 순도(purity, homogeneity)를 측정하기 위한 지표
- 계산을 편하게 하기 위하여 영역의 불순도(impurity)를 측정한다

- p_k - class k에 속할 확률
- Misclassification Error $= 1 - \max_k p_k$
- Gini Index $= 1 - \sum_k p_k^2$
- Entropy $= -\sum_k p_k \log_2 p_k$



Note that we introduced a scaled version of the entropy (entropy/2) to emphasize that the **Gini index** is an intermediate measure between **entropy** and the **classification error**.

Classification Tree – Impurity Measure

- Information Gain을 이용해서 분할 전과 분할 후의 불순도 차이를 측정
- Information Gain이 클수록 불순도를 많이 낮추는 분할 지점을 찾은 것이며, 불순도를 많이 낮출수록 좋은 분할이다

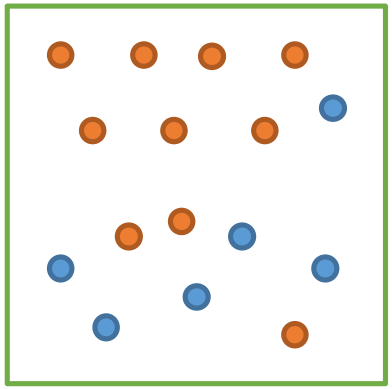
- $$IG(D_p) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

Impurity Measure - Entropy

- **Step 1.** '분할 전' 전체 관측치 영역 A에 대한 불순도를 계산

$$entropy(A) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

- $p_k = A$영역에 속한 관측치 중 k 범주에 속하는 관측치 수 비율



$$entropy(A) = -\frac{10}{16} log_2\left(\frac{10}{16}\right) - \frac{6}{16} log_2\left(\frac{6}{16}\right) = 0.95$$

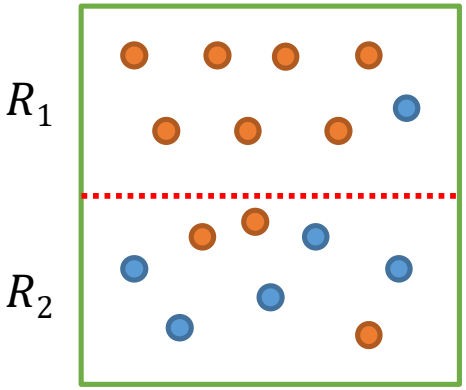
- 모든 관측치가 동일한 범주에 속할 경우 $entropy(A) = 0$
- 두 개의 범주에 속하는 개체의 수가 동일할 경우 $entropy(A) = 1$

Impurity Measure - Entropy

- Step 2. '분할 후' 분할된 영역들에 대한 불순도를 계산

$$entropy(A) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

- R_i = 분할 전 관측치 중 분할 후 영역 i에 속하는 관측치의 비율



$$entropy(R_1 + R_2) = -\sum_{i=1}^{2} P(R_i) p_k \log_2 p_k$$

$$entropy(R_1) = -\frac{7}{8}\log_2\frac{7}{8} - \frac{1}{8}\log_2\frac{1}{8} = 0.54$$

$$entropy(R_2) = -\frac{3}{8}\log_2\frac{3}{8} - \frac{5}{8}\log_2\frac{5}{8} = 0.94$$

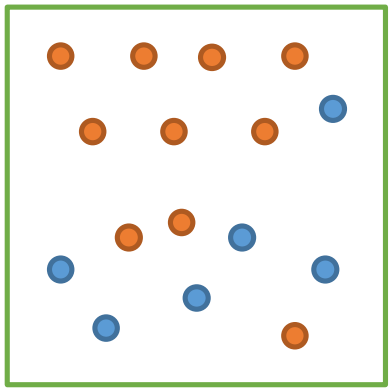
$$entropy(R_1 + R_2) = \frac{8}{16}0.54 + \frac{8}{16}0.94 = 0.61$$

- 분기 후의 **정보 획득(Information gain)**: 0.95-0.61=0.34

Impurity Measure – Gini Index

- **Step 1.** '분할 전' 전체 관측치 영역 A에 대한 불순도를 계산

- $p_k = A$영역에 속한 관측치 중 k 범주에 속하는 관측치 수 비율



$$GINI(A) = 1 - \left(\frac{6}{16}\right)^2 - \left(\frac{10}{16}\right)^2 \approx 0.47$$

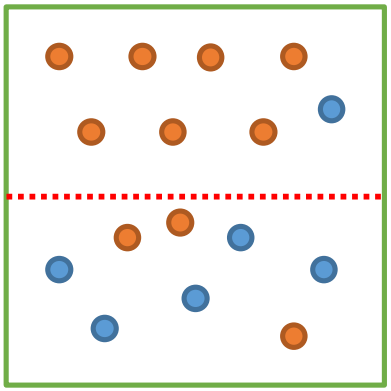
- 모든 관측치가 동일한 범주에 속할 경우 $GINI(A) = 0$
- 두 개의 범주에 속하는 개체의 수가 동일할 경우 $GINI(A) = 0.5$

Impurity Measure – Gini Index

- Step 2. '분할 후' 분할된 영역들에 대한 불순도를 계산

- R_i = 분할 전 관측치 중 분할 후 영역 i에 속하는 관측치의 비율

$$GINI(A) = \sum_{i=1}^{2} P(R_i)(1 - \sum_{K=1}^{M} (p_k)^2)$$



$$GINI(A)$$
$$= 0.5 \times \left(1 - \left(\frac{7}{8}\right)^2 - \left(\frac{1}{8}\right)^2\right) + 0.5 \times \left(1 - \left(\frac{3}{8}\right)^2 - \left(\frac{5}{8}\right)^2\right)$$
$$= 0.34$$

- 분기 후의 **정보 획득(Information gain)**: 0.47-0.34=0.13

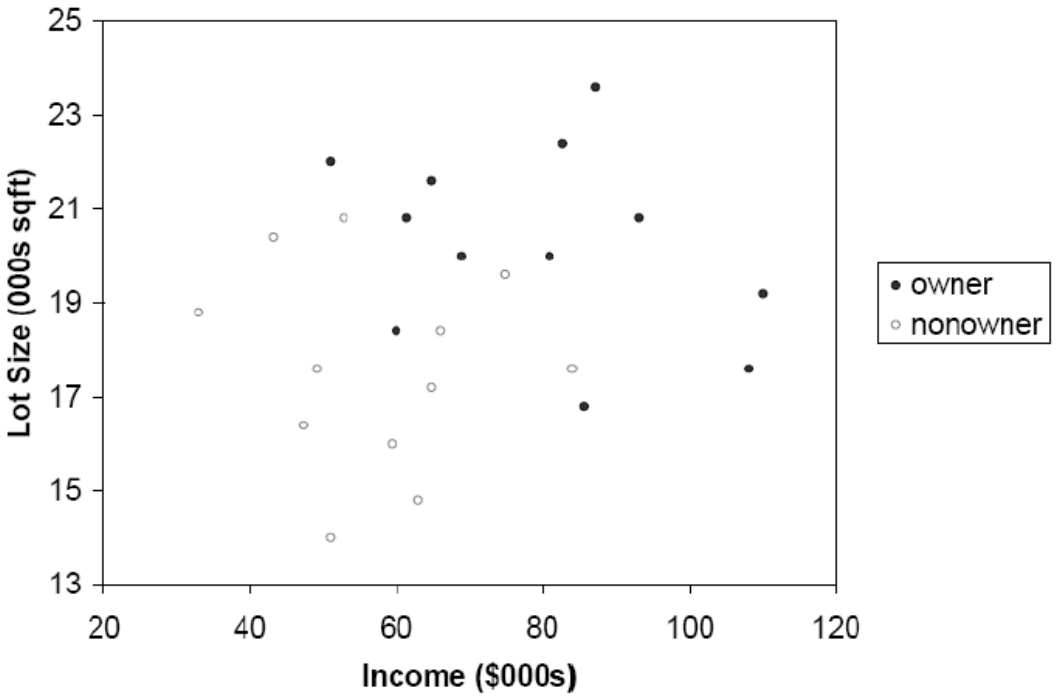
Recursive Partitioning - Example

- Income, Lot size 변수를 가지고 잔디깎이 기계를 갖고 있는 사람(Ownership) 인가를 분류하는 의사결정나무 모델을 구축

Income	Lot size	Ownership	Income	Lot size	Ownership
60.0	18.4	Owner	75.0	19.6	Non-owner
85.5	16.8	Owner	52.8	20.8	Non-owner
64.8	21.6	Owner	64.8	17.2	Non-owner
61.5	20.8	Owner	43.2	20.4	Non-owner
87.0	23.6	Owner	84.0	17.6	Non-owner
110.1	19.2	Owner	49.2	17.6	Non-owner
108.0	17.6	Owner	59.4	16.0	Non-owner
82.8	22.4	Owner	66.0	18.4	Non-owner
69.0	20.0	Owner	47.4	16.4	Non-owner
93.0	20.8	Owner	33.0	18.8	Non-owner
51.0	22.0	Owner	51.0	14.0	Non-owner
81.0	20.0	Owner	63.0	14.8	Non-owner

Recursive Partitioning - Example

- Step 1. 하나의 변수에 대해서 정렬한다 (lot size)



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
85.5	16.8	Owner
64.8	17.2	Non-owner
108.0	17.6	Owner
84.0	17.6	Non-owner
49.2	17.6	Non-owner
60.0	18.4	Owner
66.0	18.4	Non-owner
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner
69.0	20.0	Owner
81.0	20.0	Owner
43.2	20.4	Non-owner
61.5	20.8	Owner
93.0	20.8	Owner
52.8	20.8	Non-owner
64.8	21.6	Owner
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

Recursive Partitioning - Example

- Step 2. 정렬한 변수에 대해서 하나의 분할 지점을 선정한다

  (두 값의 중간지점을 선정)

  그 후, Information Gain을 산출한다

- $\text{midpoint} = 0.5 \times (14.0 + 14.8) = 14.4$

- 분할전

  $1 - \left(\frac{12}{24}\right)^2 - \left(\frac{12}{24}\right)^2 = 0.5$

- 분할후

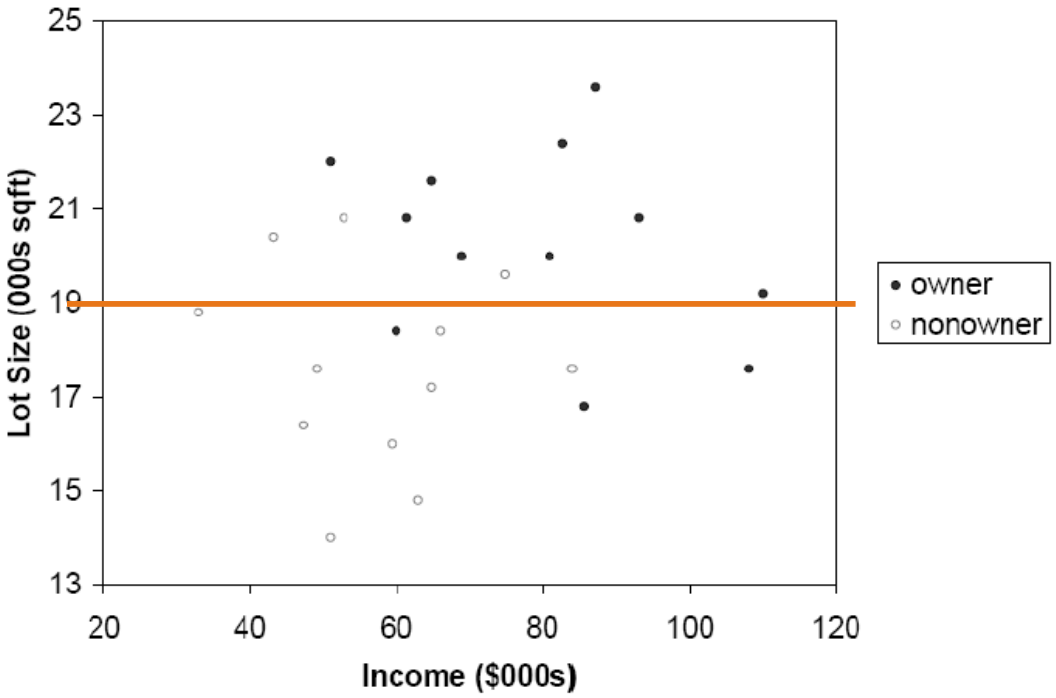
  $\frac{1}{24}\left(1 - \left(\frac{1}{1}\right)^2\right) + \frac{23}{24}\left(1 - \left(\frac{12}{23}\right)^2 - \left(\frac{11}{23}\right)^2\right) \approx 0.48$

- $\text{Information Gain} = 0.50 - 0.48 = 0.02$

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
85.5	16.8	Owner
64.8	17.2	Non-owner
108.0	17.6	Owner
84.0	17.6	Non-owner
49.2	17.6	Non-owner
60.0	18.4	Owner
66.0	18.4	Non-owner
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner
69.0	20.0	Owner
81.0	20.0	Owner
43.2	20.4	Non-owner
61.5	20.8	Owner
93.0	20.8	Owner
52.8	20.8	Non-owner
64.8	21.6	Owner
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

Recursive Partitioning - Example

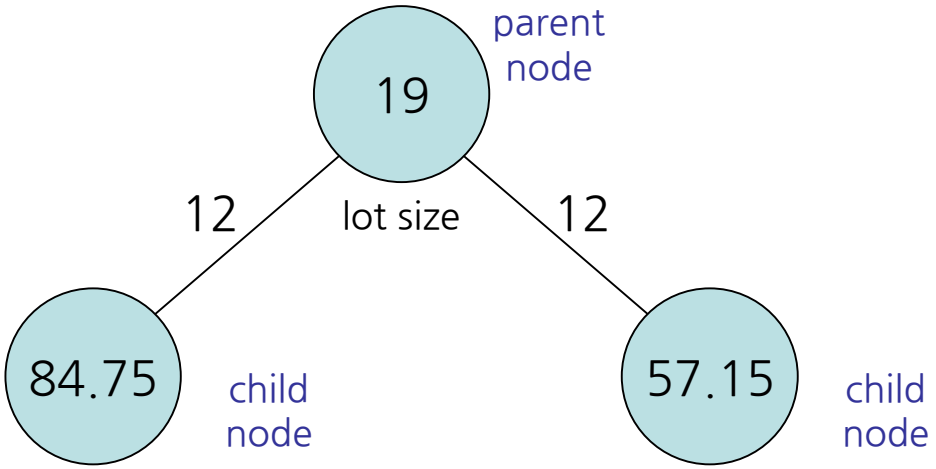
- Step 3. Step 2를 반복적으로 수행하여, best split을 찾는다



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
85.5	16.8	Owner
64.8	17.2	Non-owner
108.0	17.6	Owner
84.0	17.6	Non-owner
49.2	17.6	Non-owner
60.0	18.4	Owner
66.0	18.4	Non-owner
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner
69.0	20.0	Owner
81.0	20.0	Owner
43.2	20.4	Non-owner
61.5	20.8	Owner
93.0	20.8	Owner
52.8	20.8	Non-owner
64.8	21.6	Owner
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

Recursive Partitioning - Example

- Step 3. Step 2를 반복적으로 수행하여, best split을 찾는다
- 주의: income에 대해서도 동일한 과정을 수행해야 함



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
85.5	16.8	Owner
64.8	17.2	Non-owner
108.0	17.6	Owner
84.0	17.6	Non-owner
49.2	17.6	Non-owner
60.0	18.4	Owner
66.0	18.4	Non-owner
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner
69.0	20.0	Owner
81.0	20.0	Owner
43.2	20.4	Non-owner
61.5	20.8	Owner
93.0	20.8	Owner
52.8	20.8	Non-owner
64.8	21.6	Owner
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

Recursive Partitioning - Example

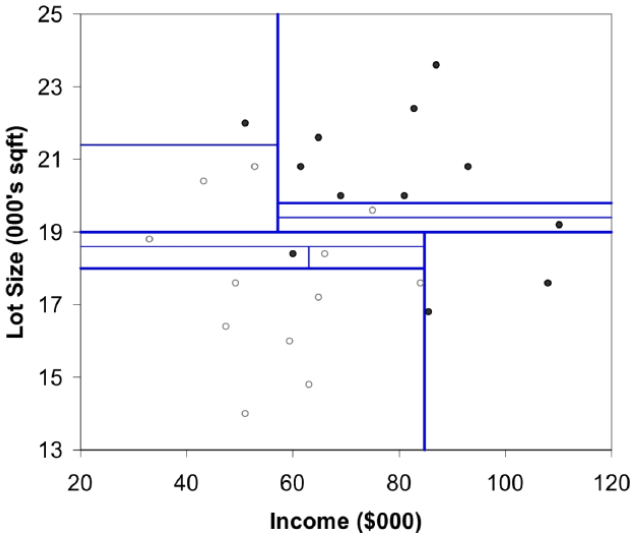
- Recursively repeat

- K개의 변수와 N개의 관측치가 있을 경우,

  한 iteration 당 계산해야 하는 information gain의 수

  $= K \times (N - 1)$

- 모든 분할된 구역에 한 종류의 클래스만 포함될 때까지 반복



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
85.5	16.8	Owner
64.8	17.2	Non-owner
108.0	17.6	Owner
84.0	17.6	Non-owner
49.2	17.6	Non-owner
60.0	18.4	Owner
66.0	18.4	Non-owner
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner
69.0	20.0	Owner
81.0	20.0	Owner
43.2	20.4	Non-owner
61.5	20.8	Owner
93.0	20.8	Owner
52.8	20.8	Non-owner
64.8	21.6	Owner
51.0	22.0	Owner
82.8	22.4	Owner
87.0	23.6	Owner

Impurity Measure – Categorical Variables

- 명목형 변수에 대한 불순도 계산 방법은?

- 카테고리를 구분할 수 있는 모든 방법에 대해서 불순도를 계산해야 한다

- Example - A, B, C 3개의 카테고리가 있는 변수 X_3에 대해서

1. $\{A\}$ vs $\{B, C\}$ → $X_3 = A \ or \ not$
2. $\{B\}$ vs $\{A, C\}$ → $X_3 = B \ or \ not$
3. $\{C\}$ vs $\{A, B\}$ → $X_3 = C \ or \ not$

가장 좋은 split point

(A, B, C 중 하나)를 탐색

의사결정나무 예측 단계

- 각 leaf node의 label 혹은 클래스는 포함된 관측치들의 'voting'에 의해 결정된다

- Voting - 사전에 정의된 cutoff value보다 높으면 해당 클래스를 할당

- 예) 타겟 클래스가 1이며, 전체 클래스가 1 & 0인 이진 분류 문제에서

  cutoff value = 0.7로 사전에 정의가 되어 있다면,

  1) leaf node #3에서 클래스 1 비율이 0.75 → leaf node 클래스는 1

  2) leaf node #7에서 클래스 1 비율이 0.40 → leaf node 클래스는 0

- multiclass 문제 → 가장 높은 비율을 갖는 클래스로 할당

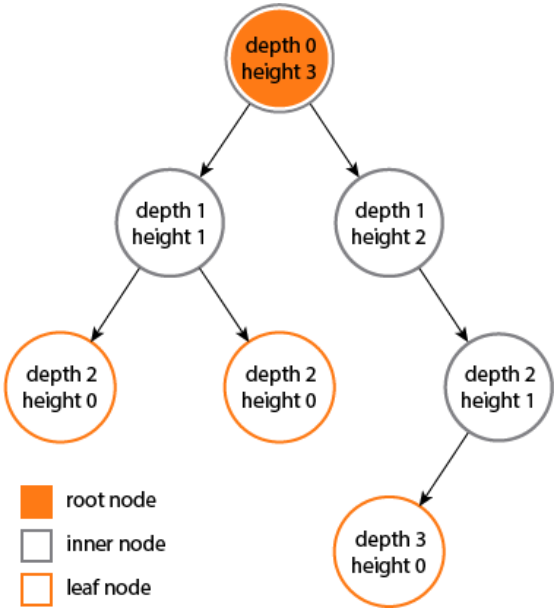
Key Ideas

1. Recursive Partitioning
   - 데이터 공간을 반복적으로 2개로 나눈다
   - 새롭게 나뉘어진 공간 안에 있는 데이터들의 동종성 (homogeneity)를 최대화 할 수 있는 방향으로 나눈다

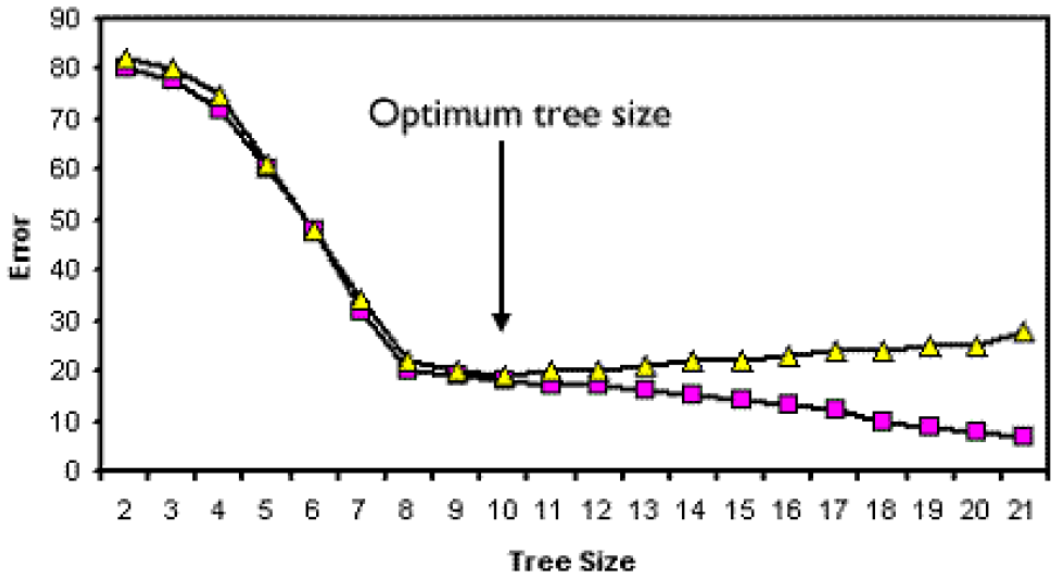
2. Pruning Tree
   - 과적합을 막기 위한 방법
   - 학습, 구축된 의사결정나무 모델을 보다 단순화하여 불필요하게 '과하게 학습'되는 것을 방지하는 방법

Overfitting - 과적합

- Recursive Partitioning은 모든 terminal node의 순도가 100%일 때,
  즉 각 노드에 한 종류의 클래스만 존재할 때 종료됨
- 이를 full tree라고 부르며 일반적으로 과적합에 대한 위험성이 있고,
  generalization 성능이 저하되는 현상이 발생함



[tree depth 개념도]



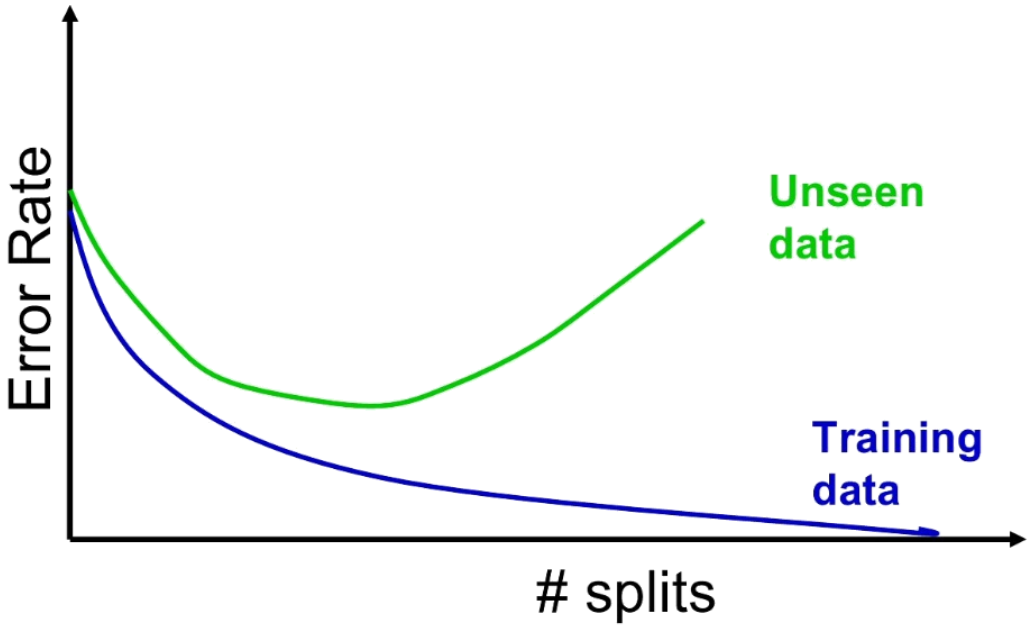
Overfitting - 과적합

- 과적합이란?

  Memorize the training dataset, rather than discovering significant patterns

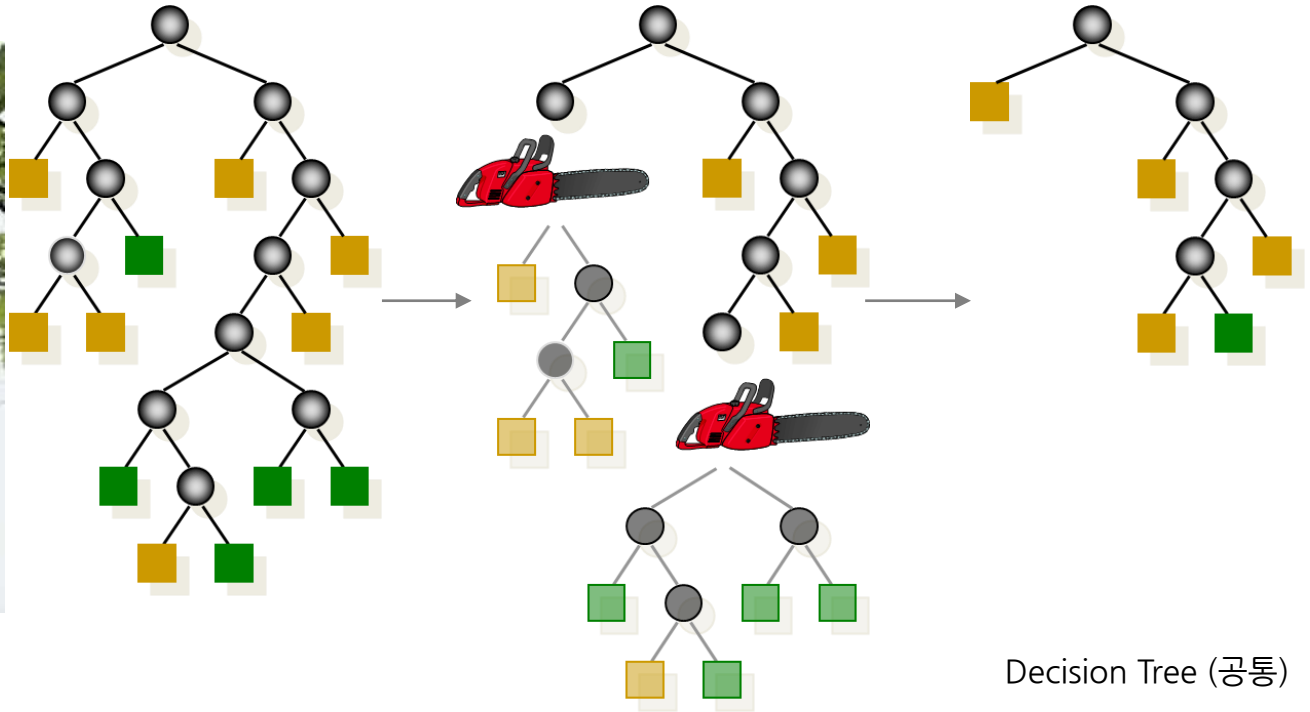
- 데이터에 포함되어 있는 noise까지 학습해서, 정작 예측 성능은 저하되는 현상

  → poor generalization performance



Pruning Tree

- Full tree를 구축한 뒤, 적절한 수준에서 검증데이터에 대한 오분류율을 줄일 수 있도록 pruning을 수행
- 비교적 단순한 구조의 의사결정나무를 구축할 수 있음
- Cost complexity, maximum depth 등을 조절하여 최적 구조를 탐색 (다양한 접근 방법이 존재)



Pruning Tree

- Cost complexity

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

  - $CC(T)$ - 생성된 의사결정나무가 갖고 있는 전체 복잡도
  - $Err(T)$ - 검증(validation) 데이터에 대한 오분류율
  - $L(T)$ - leaf node의 개수
  - α - 사용자 지정 파라미터

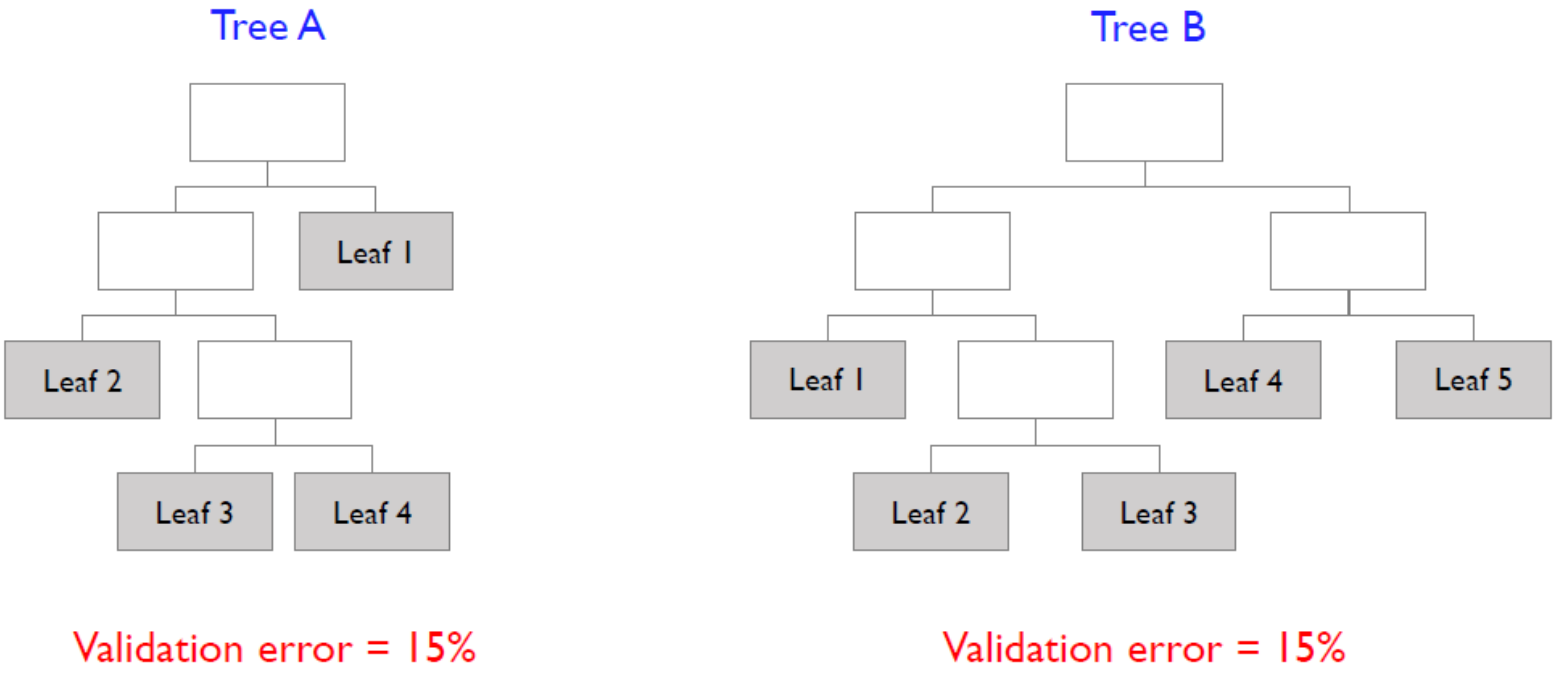
- α가 클수록

  → leaf node에 대한 penalty가 커진다

  → leaf node 개수가 적은, 단순한 구조의 의사결정 나무를 만들게 된다

Pruning Tree

- 이왕이면 단순한 (leaf node의 수가 적은) 모델을 구축하는 것이 좋다



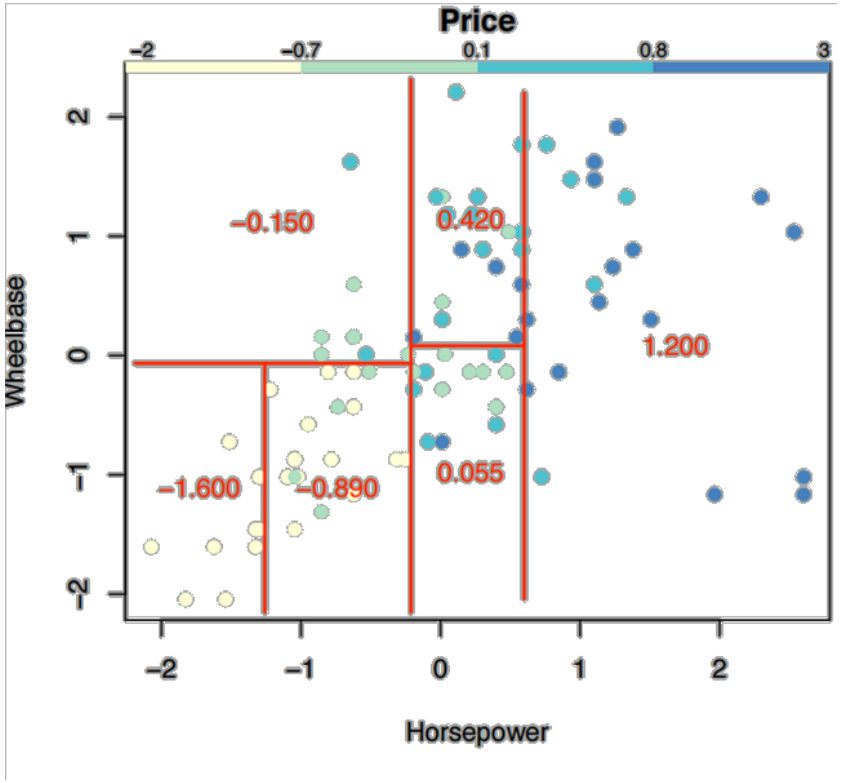
- 만약, validation error가 동일하다면 단순한 모델인 Tree A를 선택

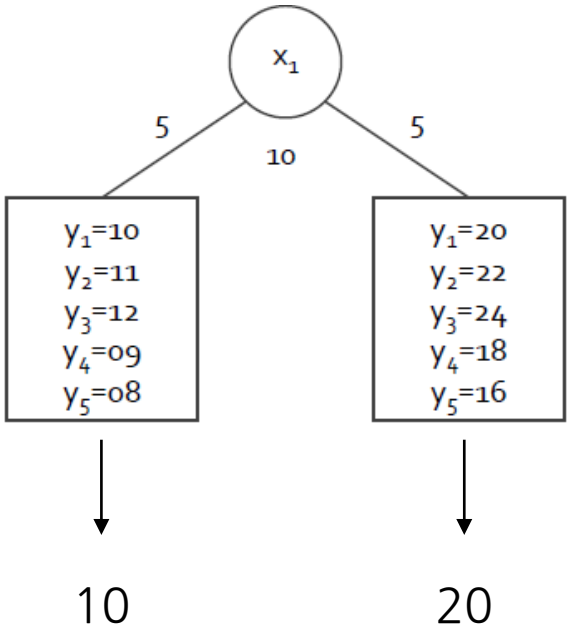
Pruning Tree

- Scikit-learn…
  - ✓ max_depth
  - ✓ min_samples_split - the number of samples to split an internal node
  - ✓ min_samples_leaf - the number of samples required to be at a leaf node

Regression Tree

- Recursive partitioning, pruning은 분류 의사결정나무와 동일
- 차이점 1 - prediction of the node
  - ✓ 분류 의사결정나무 → voting을 사용해서 클래스를 지정
  - ✓ 예측 의사결정나무 → 노드에 속한 관측치들의 타겟 변수 평균값을 사용

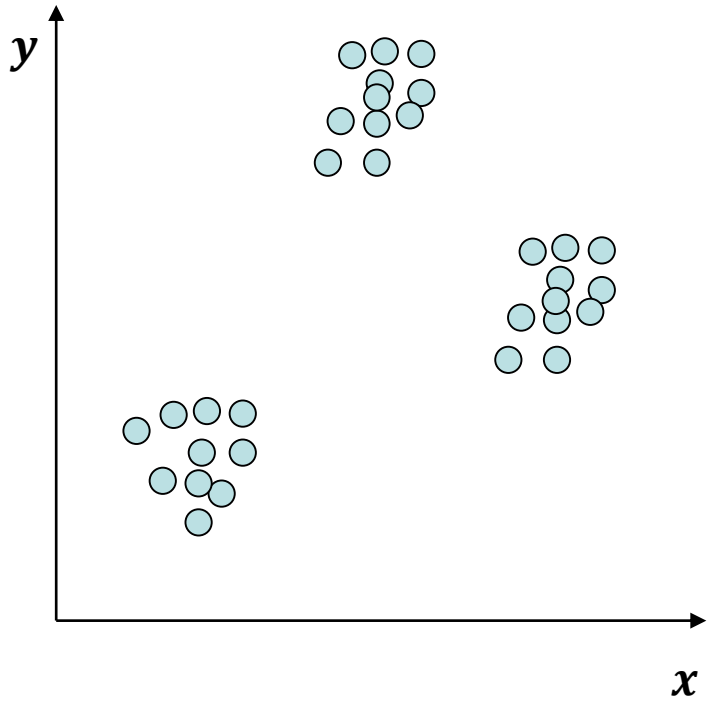


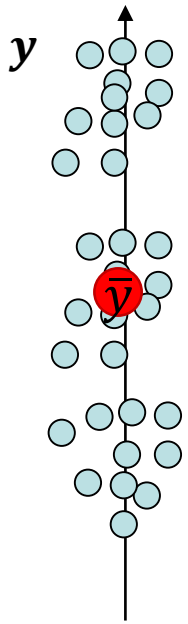


Regression Tree

Regression Tree

- 차이점 2 - Impurity measure
  - ✓ 각 노드에서 데이터가 흩어진 정도(분산)를 측정
  - ✓ Sum of Squared Error (SSE) = $\sum_{i=1}^{n}(y_i - \bar{y})^2$

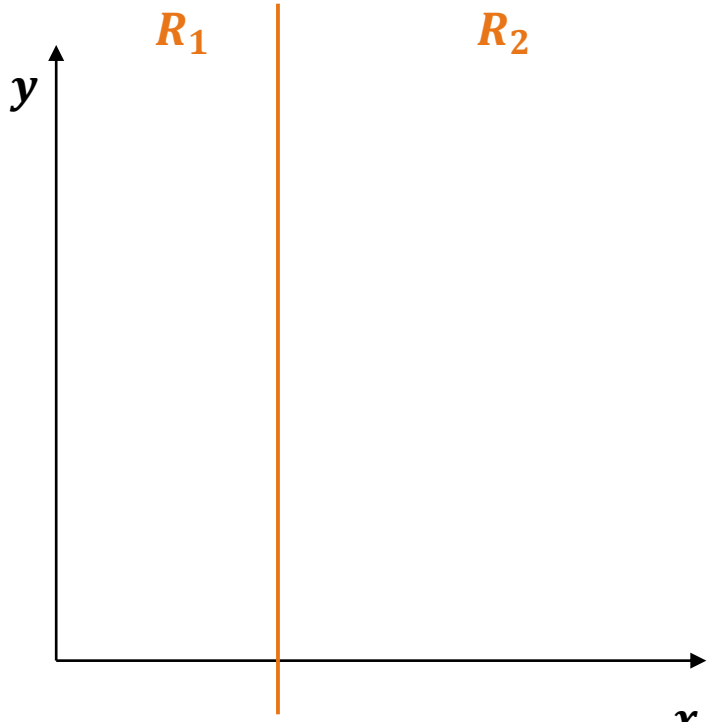


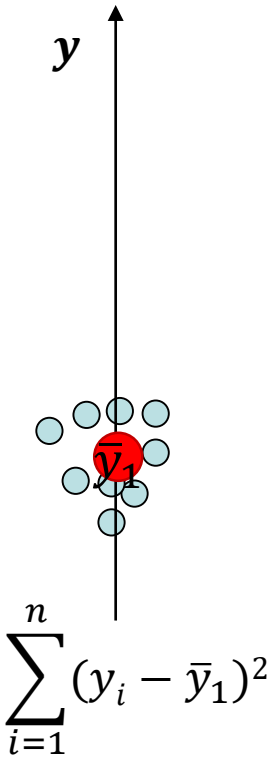


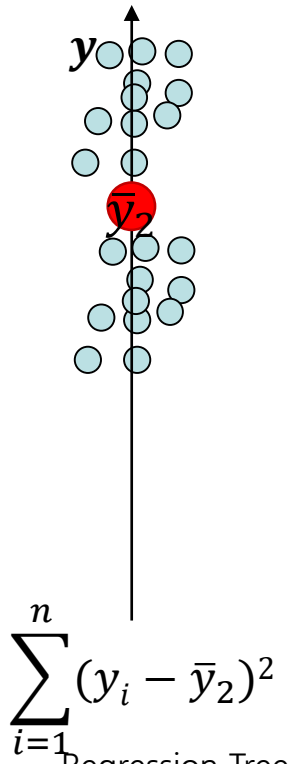
$$\sum_{i=1}^{n}(y_i - \bar{y})^2$$

Regression Tree

- 차이점 2 - Impurity measure
  - ✓ 각 노드에서 데이터가 흩어진 정도(분산)를 측정
  - ✓ Sum of Squared Error (SSE) = $\sum_{i=1}^{n}(y_i - \bar{y})^2$

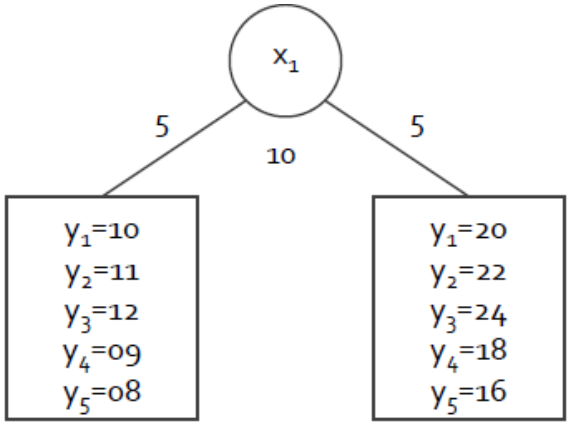






Regression Tree

- 차이점 2 - Impurity measure
  - ✓ 각 노드에서 데이터가 흩어진 정도(분산)를 측정
  - ✓ Sum of Squared Error (SSE) = $\sum_{i=1}^{n}(y_i - \bar{y})^2$



- ✓ $SSE(parent) = 300$
- ✓ $SSE(left) = 10$
- ✓ $SSE(right) = 40$
  - ➢ $GAIN = 300 - (10 + 40) = 250$

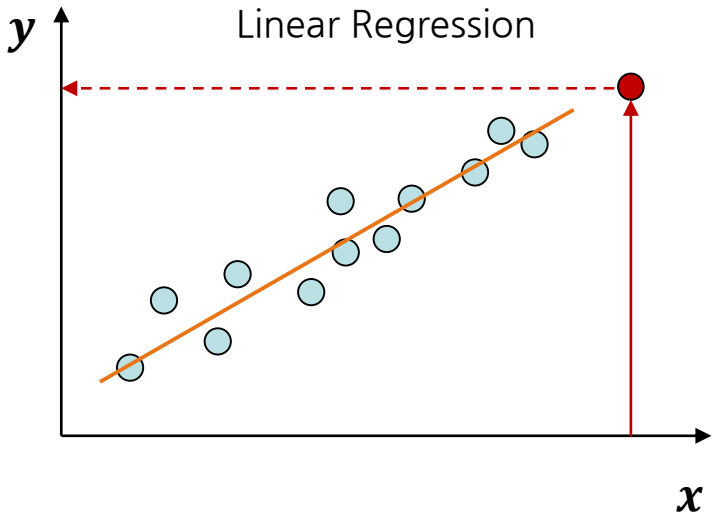
Regression Tree

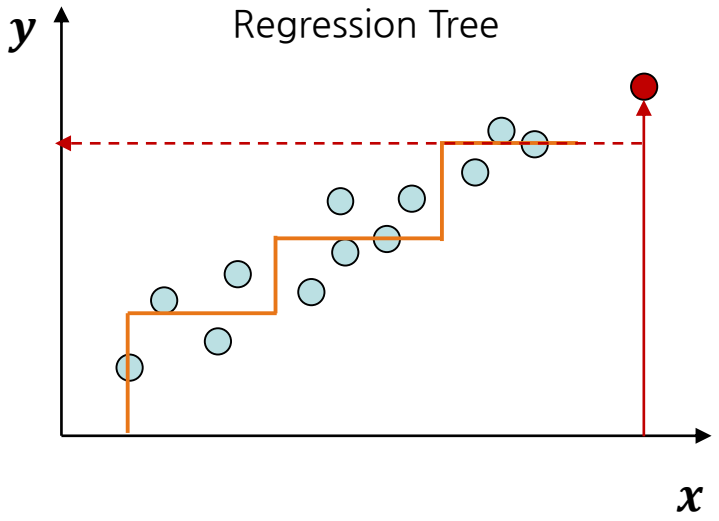
- Linear Regression vs Regression Tree

① Regression tree의 예측 가능한 값의 종류는 leaf node의 수와 같다

→ 그 영역의 평균값으로 예측이 수행되기 때문

② Train data가 갖고 있는 y 값의 범위 내에서만 예측이 가능하다



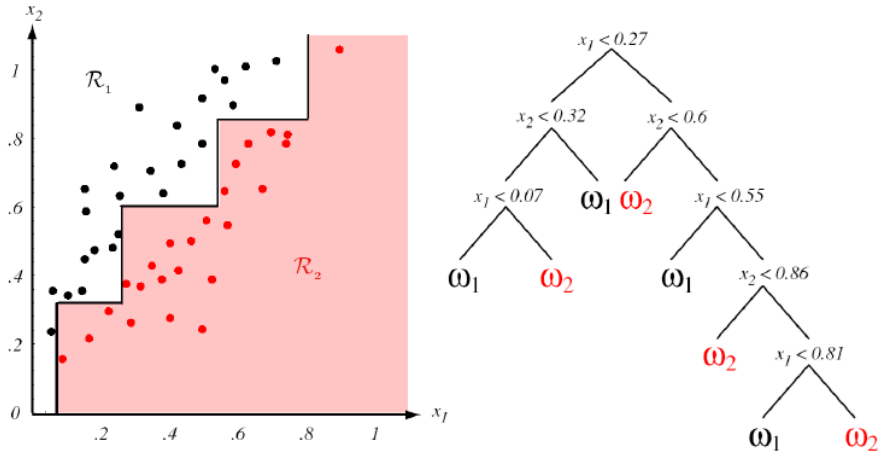


의사결정나무의 장점

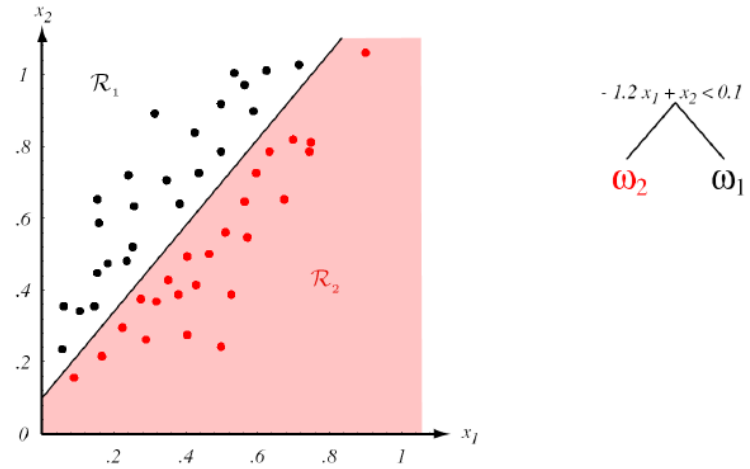
- 의사결정나무는 사용하고 이해, 해석하기 편한 알고리즘
- Decision rule까지 얻을 수 있다
- 분류와 예측 모델로써 모두 사용이 가능하다
- Split 때 사용된 변수를 살펴봄으로써 간접적으로 변수 중요도를 산출할 수 있다
- 통계적 가정이 필요하지 않다

의사결정나무의 단점

- Vertical or horizontal split으로 잘 구분되지 않는 데이터에서는 상대적으로 낮은 성능을 보일 가능성이 있다



- 하나의 변수에 대해서 split을 반복하기 때문에, 변수간 관계를 고려하지 못한다



EOD