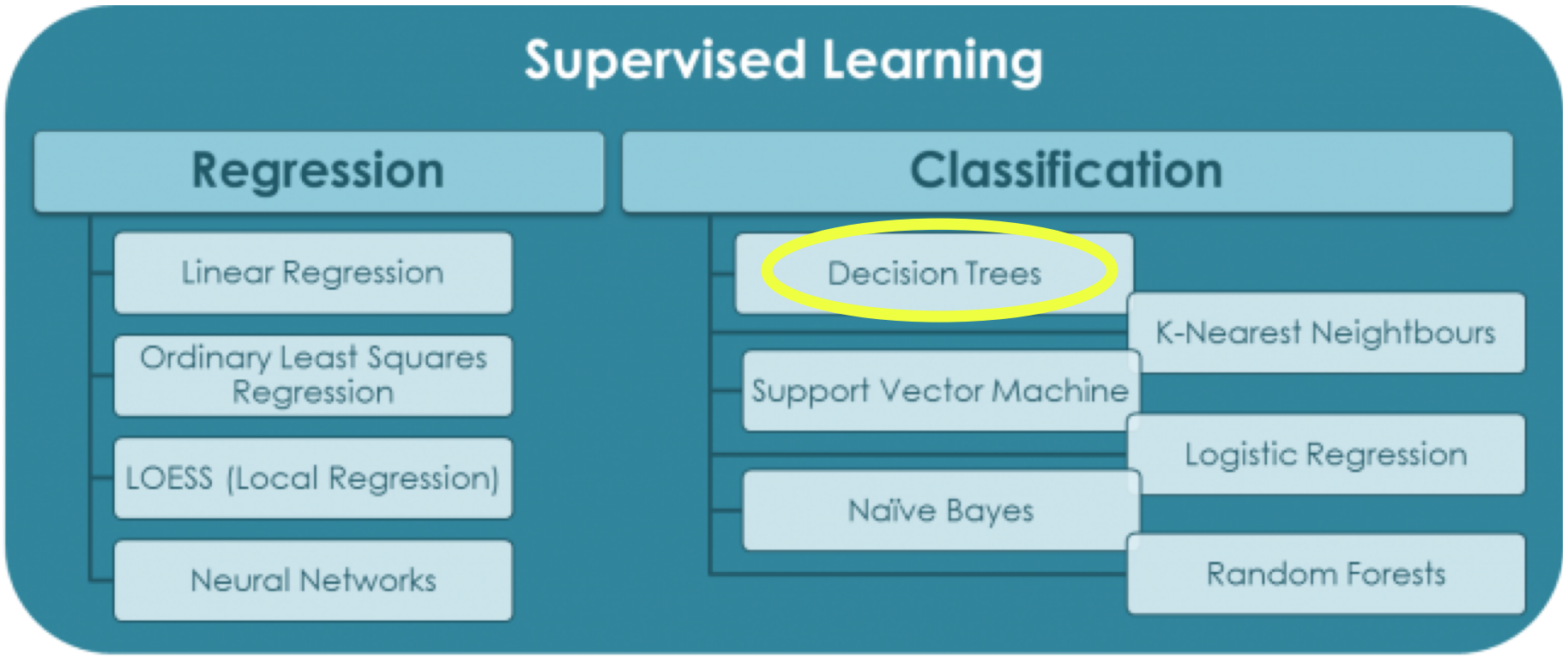
Ⅲ. 지도학습 (Supervised Learning)

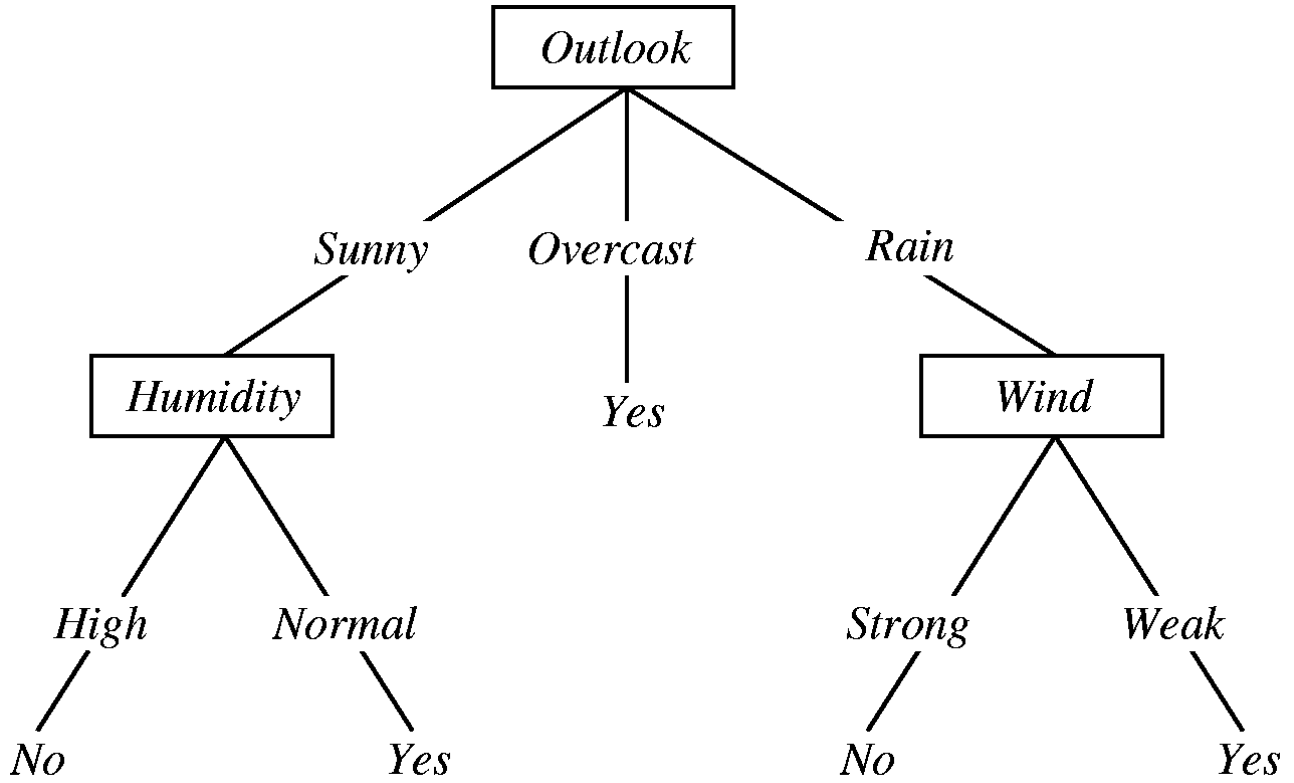
- Classification

❸ Decision Tree (의사결정 나무)


Supervised Learning
Regression
Linear Regression
Ordinary Least Squares Regression
LOESS (Local Regression)
Neural Networks
Classification
Decision Trees
K-Nearest Neightbours
Support Vector Machine
Logistic Regression
Naïve Bayes
Random Forests

Decision Tree (의사결정나무) 란?

- 가장 단순한 classifier 중 하나
- 아래 그림은 오늘 외출을 할까 말까 를 결정하는 decision tree
- 이렇듯 decision tree는 tree구조를 가지고 있으며, root에서부터 적절한 node를 선택하면서 진행하다가 최종 결정을 내리게 되는 model

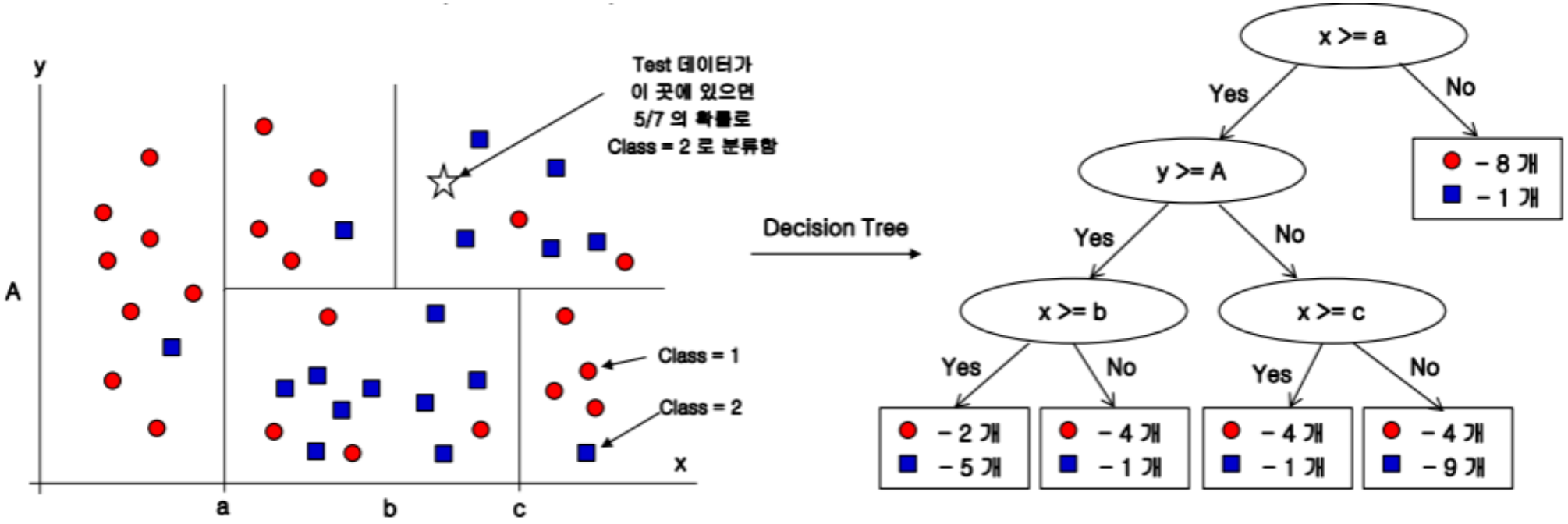


Decision tree의 장점

- 단순하다 -> 누구나 쉽게 이해할 수 있고, 그렇기 때문에 쉽게 디버깅할 수 있다.
- 예를 들어 위의 예시에서 습도가 높아도 나갈 만하다는 생각이 든다면 맨 왼쪽의 No를 Yes로 바꾸기만 하면 간단하게 로직을 바꿀 수 있다. 그러나 다른 모델들은 그런 점들이 비교적 어렵다.
- Machine Learning에서 말하는 decision tree는 decision tree learning으로, 일일이 node마다 로직을 사람이 써넣어 만드는 것을 의미하는게 아니라, node 개수, depth, 각각의 node에서 내려야하는 결정 등을 데이터를 통해 learning하는 algorithm들을 사용해 만든 decision tree를 의미한다.
- 많이 쓰이는 알고리즘들로는 ID3, C4.5, CART, CHAID, MARS 등이 있으며, 보통 C4.5를 가장 많이 사용한다.

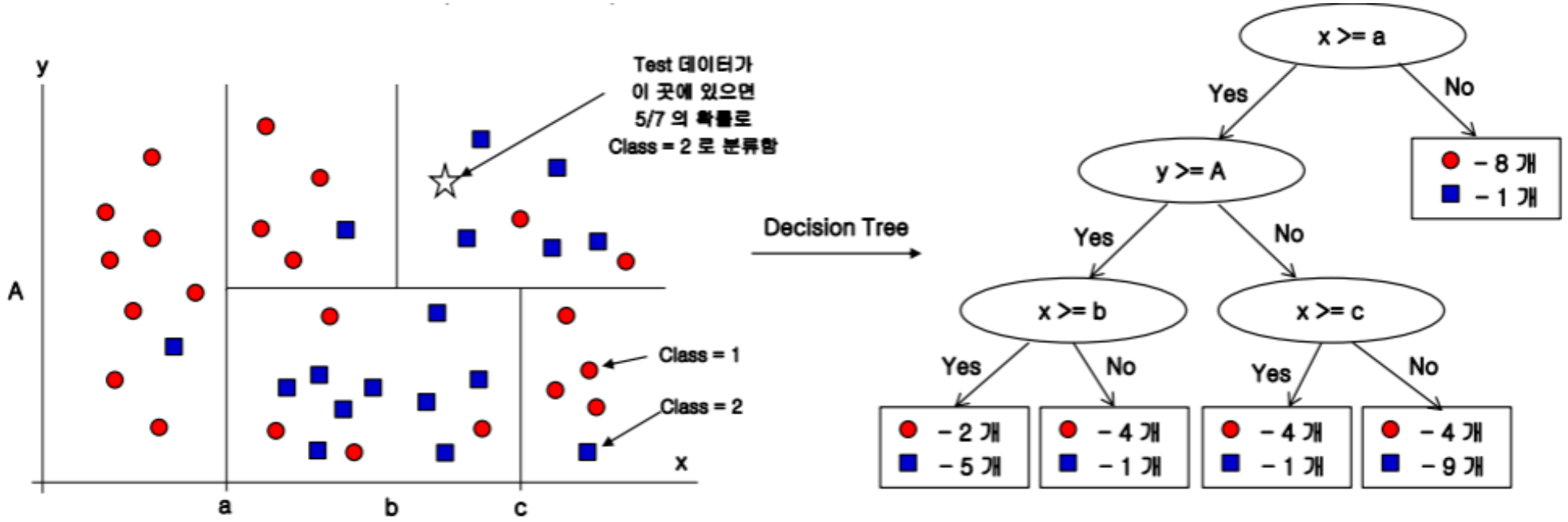
Decision Tree (의사결정나무) 란?

- 훈련 데이터가 왼쪽 그림처럼 구성된 경우 분류를 위한 경계선은 오른쪽의 **트리 형식**으로 표현할 수 있음.
- 아래 예시는 입력 변수가 2개 (x, y) 인 경우이며 2차원 평면상에 배치된 데이터들을 4개의 직선으로 분류한 경우임.
- 입력 변수가 3개 이상인 다차원 구조에서는 직선 대신 (초) 평면으로 구분함.



Decision Tree (의사결정나무) 란?

- 트리 구조로 경계선을 구분하면 **차원이 높아져도 직관적으로 이해하기 쉬우므로**, 분류 기법으로 많이 사용됨.
- 경계선을 너무 많이 사용하면 트리 구조가 복잡해지고, 과잉 적합의 우려가 있음.
- 과잉 적합 문제를 해결하기 위한 방편으로 트리가 너무 복잡해지지 않도록 **사전/사후 가지치기 (Pruning) 방법**이 사용됨.
- 일반화 특성과 정확도를 향상시키기 위해서는 훈련 데이터를 여러 개로 분할하고, 분할된 데이터 마다 트리를 구성한 후 결과를 종합하는 **랜덤 포레스트 (Random Forest) 방법**이 사용됨.

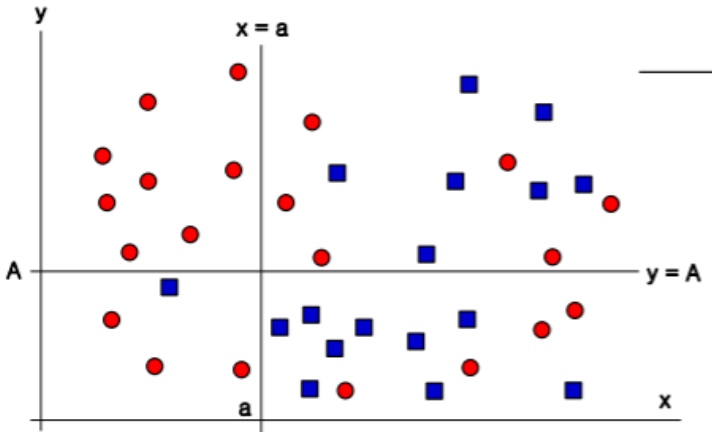


Decision Tree (의사결정 나무) 알고리즘 : 불순척도(Inpurity)와 최적의 분할 선택 기준 (분리 기준)

- 각 노드의 **불순척도** : 엔트로피, 지니지수 등
- 불순척도가 작아지도록 (순도가 높은) 분할 기준을 선택함.
- 분할 전 부모 노드의 불순척도보다 분할 후 자식 노드의 불순척도가 높을수록 좋음. (Information gain)

$$H(t) = -\sum_{i=1}^{c} p(i|t)\log_2 p(i|t)$$ ← 엔트로피

$$Gini(t) = 1 - \sum_{i=1}^{c} p(i|t)^2$$ ← 지니 지수



1. 초기분할 : x = a 인 경우 Gini 지수

Left: ● – 11 개, ■ – 1 개

Right: ● – 10 개, ■ – 16 개

$$G_L = 1 - \left(\frac{11}{12}\right)^2 - \left(\frac{1}{12}\right)^2 = 0.153$$

$$G_R = 1 - \left(\frac{10}{26}\right)^2 - \left(\frac{16}{26}\right)^2 = 0.473$$

$$E_a = \left(\frac{12}{38}\right)*0.153 + \left(\frac{26}{38}\right)*0.473 = 0.372$$ 가중평균

2. 초기분할 : y = A 인 경우 Gini 지수

Upper: ● – 14 개, ■ – 7 개

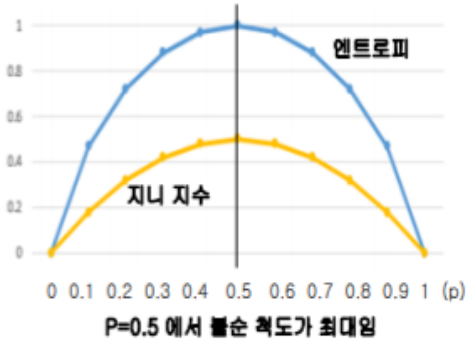
Below: ● – 7 개, ■ – 10 개

$$G_U = 1 - \left(\frac{14}{21}\right)^2 - \left(\frac{7}{21}\right)^2 = 0.444$$

$$G_B = 1 - \left(\frac{7}{17}\right)^2 - \left(\frac{10}{17}\right)^2 = 0.484$$

$$E_A = \left(\frac{21}{38}\right)*0.444 + \left(\frac{17}{38}\right)*0.484 = 0.462$$

불순도가 더 낮음
x = a 로 분할하는
것이 좋음



P=0.5 에서 불순 척도가 최대임

Decision Tree의 특징

- 분류나 예측의 근거를 알 수 있으므로 결정 과정을 쉽게 이해할 수 있음.
- 데이터의 차원이 높아져도 (Feature가 많아져도) 분류에 덜 중요한 Feature들은 분류 기준에서 제외되므로 Feature 선정에 크게 신경 쓸 필요가 없음.
- Feature 마다 분류에 영향을 미치는 정도를 파악할 수 있음. 중요한 Feature들을 찾아볼 수 있음. -> **중요도 분석**
- Decision Tree를 과도하게 분할할 경우 **Tree가 복잡해지고 과잉적합 (Overfitting) 문제가 발생**함.

Decision Tree의 특징

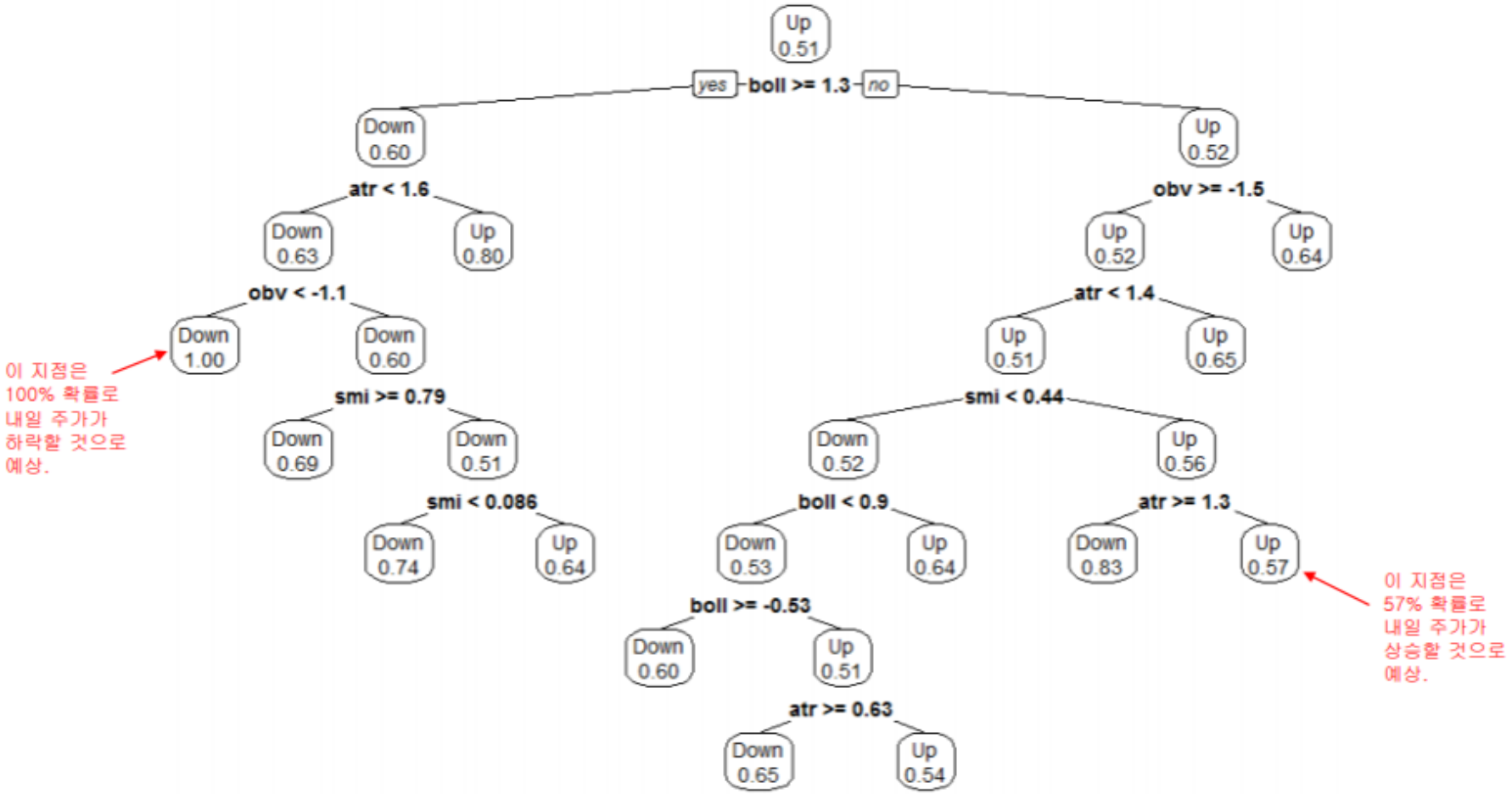
- 트리가 복잡해 질수록 아래 쪽 노드에 포함되는 데이터가 작아지므로 통계적으로 의미 있는 결정을 내리기 어려움. **-> 데이터 단편화 (Data fragmentation)**
- 트리 안에 동일한 서브 트리 (Sub Tree)가 반복해서 복제되어 나타날 수 있음. **서브 트리는 Decision Tree를 복잡하게 만들고 이해하기 어렵게 만듦.**
- 트리가 복잡해지고, 과잉적합 문제를 개선하기 위해서는 트리가 더 이상 커지지 않도록 하는 **정지기준 (Stopping rule)과 가지치기 (Pruning) 방법**이 있음.
- 정지기준은 트리의 깊이 (Depth)를 지정하거나, 마지막 노드의 데이터 수가 임계치 이하로 떨어지지 않도록 지정하는 방법이 있음.
- 가지치기는 사전 가지치기와 사후 가지치기 방법이 있음.

가지치기 (pruning)

- 트리가 너무 복잡해지지 않도록 단순화 시킴 -> 일반화 특성을 향상 시킴.
- 평가용 데이터 (Validation Data)를 이용하여 일반화 특성이 좋아지는 지점에서 트리 성장을 멈추거나, 데이터 전문가에 의해 타당성이 없는 규칙은 제거함.
- **사전 가지치기 (Pre-pruning)**
  - 조기 정지 규칙에 의해 트리 성장을 멈춤. 정지 규칙으로는 트리의 깊이, 마지막 노드의 최소 데이터 수, 불순 척도 등의 임계치를 이용하는 방법 등이 있음.
- **사후 가지치기 (Post-pruning)**
  - 초기에는 트리를 최대 크기로 만듦. 마지막 노드의 불순 척도가 최소가 되도록 분할함. 복잡도 (Complexity)가 최대가 됨.
  - 완전히 성장한 트리를 위쪽 방향으로 다듬어가는 절차를 수행함. -> Trimming
  - Cross Validatation (CV) 시험 오차가 최소가 되는 분할 수준으로 트리를 줄임.

Decision Tree 예시

- (cp를 높여서 더 단순하게 생성한 트리. Underfitting)



Regression Tree

- Decision tree는 output value가 반드시 binary여야 한다는 제약조건이 있기 때문에 스팸 필터 등에서는 사용할 수 있지만, 실제 모든 데이터가 binary만을 output으로 가지지 않으므로 모든 데이터에 사용하려면 변형이 필요하다.
- **Regression tree**는 binary가 아니라 real value를 output으로 가지는 모델로, learning하는 방법은 크게 다르지 않다.
- 가끔은 하나의 decision tree를 사용하는 것이 아니라 한 번에 여러 개의 decision tree들을 만들어서 각각의 decision tree들이 내리는 결정을 종합적으로 판단하여 (ensemble) 결정을 내리기도 한다. **Bagging decision tree, random forest등**이 이에 속한다.
- 이런 식으로 여러 개의 classifier를 사용해 decision을 내리는 방법을 **ensemble method**라고 하는데, industry에서는 machine learning algorithm의 성능을 높이기 위해서 여러 개의 알고리즘들을 ensemble method를 사용하여 한 번에 같이 사용하기도 한다. 대표적인 예로 **AdaBoost** 등이 있다.