Decision Tree

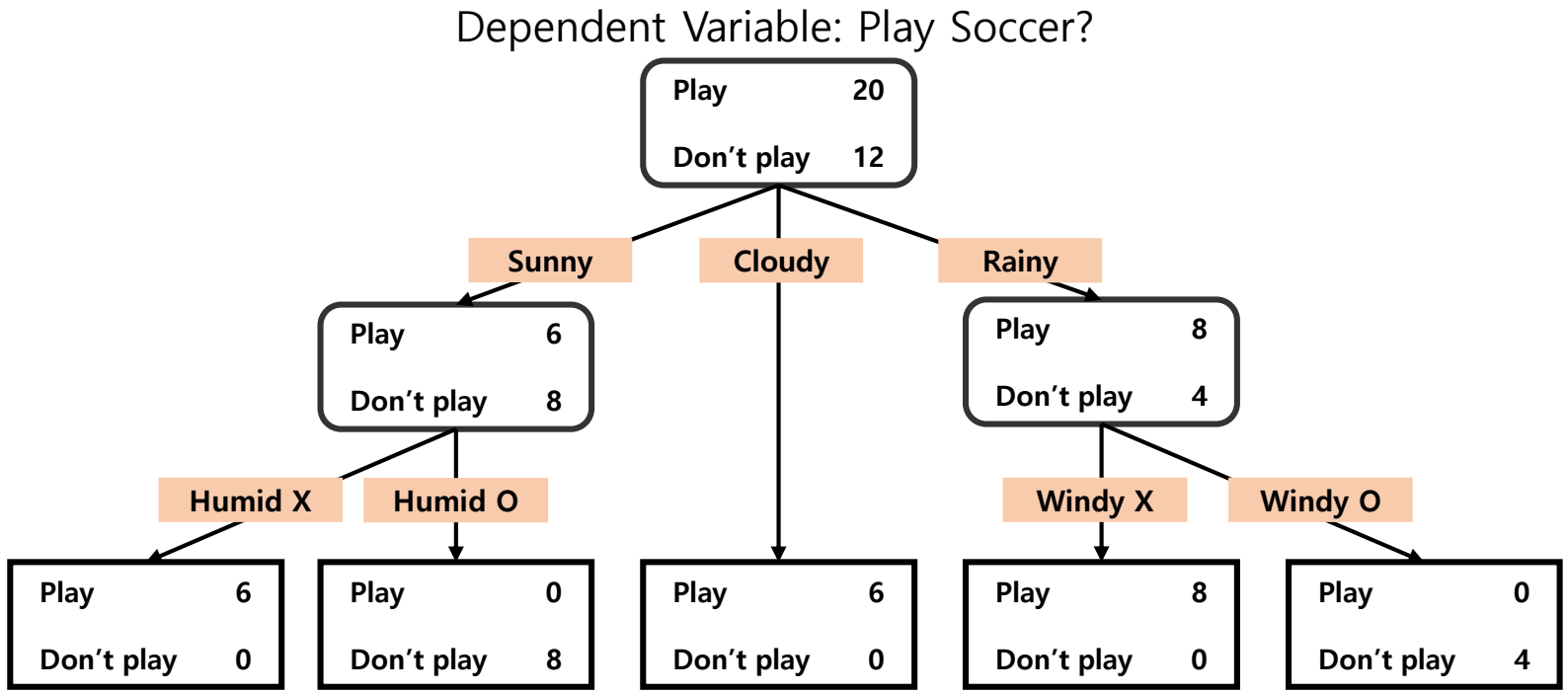
Jehyuk Lee

Department of Data Science

Kookmin University

Decision Tree

- **정의 및 특징**
  - 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내 tree기반의 분류 규칙을 만드는 것
  - 한번에 하나씩의 설명 변수를 사용하여 정확한 예측이 가능한 규칙들의 집합을 생성



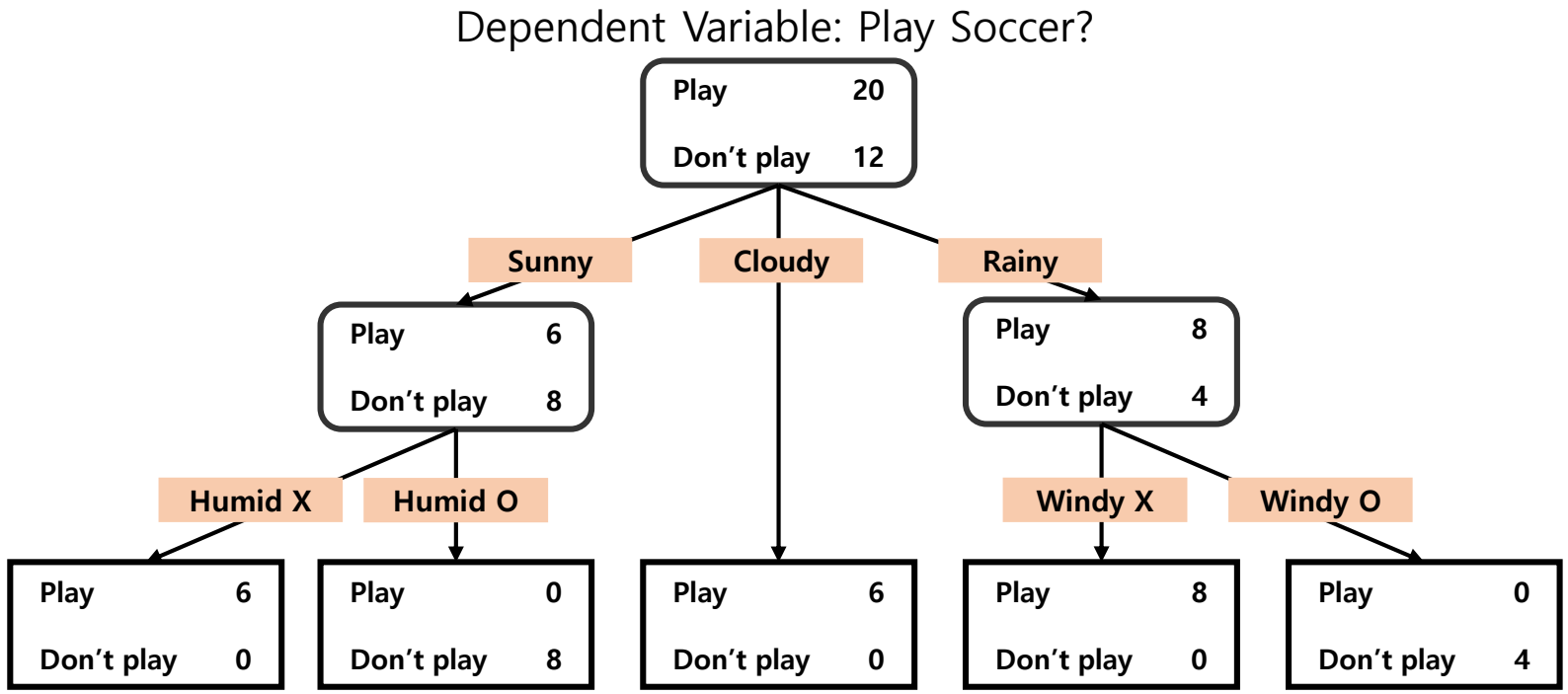
규칙 예시

만약 내일 맑고 습하지 않으면 축구를 할 것이다.

만약 내일 비가 오는데 바람까지 불면 축구를 하지 않을 것이다.

Decision Tree

- **장점**
  - if/then형식의 규칙으로 나타냄 → 사람이 직관적으로 이해하기 편함
  - 데이터 전처리(범주형 변수의 벡터화, 정규화 등)가 상대적으로 적게 요구됨



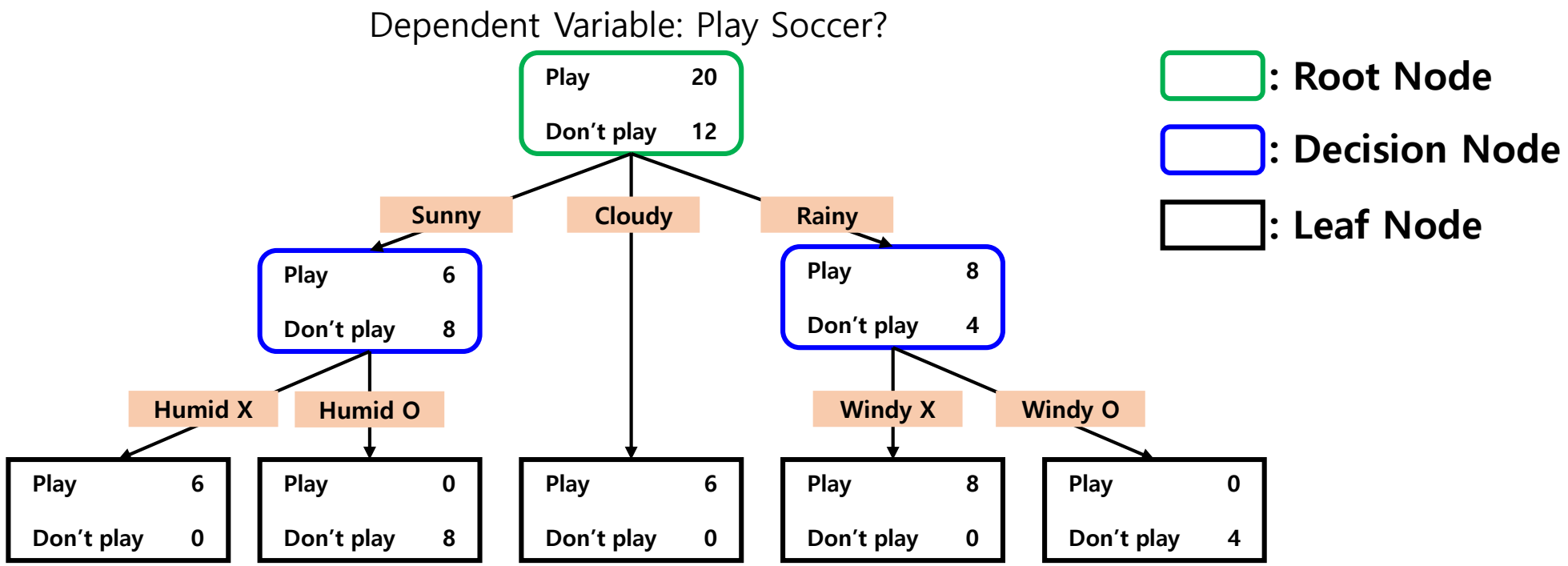
규칙 예시

만약 내일 맑고 습하지 않으면 축구를 할 것이다.

만약 내일 비가 오는데 바람까지 불면 축구를 하지 않을 것이다.

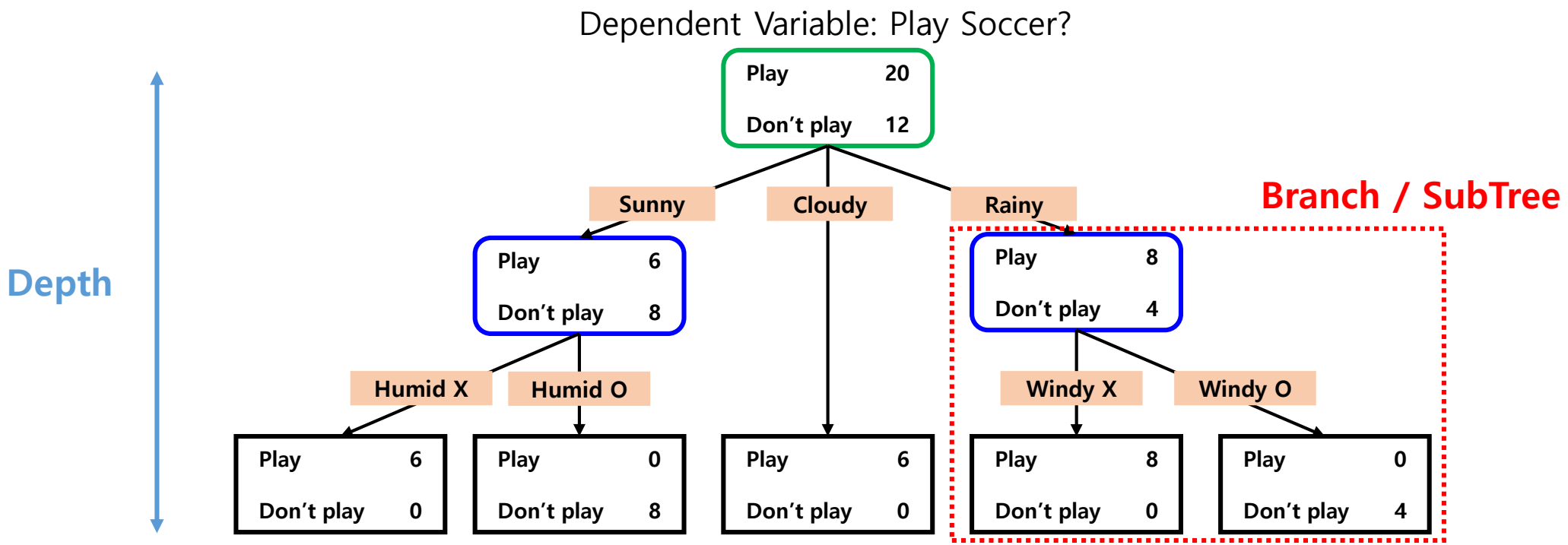
Decision Tree

- 구조
  - **루트 노드(Root Node):** 깊이가 0인 맨 꼭대기의 노드
  - **규칙 노드(Decision Node):** 규칙 조건
  - **리프 노드(Leaf Node):** 자식 노드가 더 없는 노드로, 결정된 클래스 값



Decision Tree

- 구조
  - **SubTree:** 새로운 규칙 조건마다 새로운 subtree가 생성
    - Data에 feature가 있고, 이를 결합하여 규칙 조건을 만들 때마다 규칙 노드 생성
    - 그러나 규칙이 많다는 건 분류를 결정하는 방식이 복잡해짐을 의미 → overfitting
    - 즉, Tree의 깊이(depth)가 깊을 수록 Overfitting 가능성이 높음



CART algorithm

- **CART(Classification And Regression Tree)알고리즘**
  - Decision Tree를 학습시키는 대표적인 알고리즘
  - 개별 변수의 영역을 반복적으로 분할, 전체 영역에서의 규칙을 생성하는 지도학습 기법
  - If/Then 형식으로 표현되는 규칙을 생성 → 결과 예측과 동시에 이유도 설명도 가능
  - 수치형 변수와 범주형 변수에 대한 동시 처리 가능

CART algorithm

- **Key Ideas**
  - **재귀적 분기(Recursive Partitioning)**
    - 입력 변수의 영역을 2개로 구분 → 구분 후, 각 영역의 순도(purity)가 증가하도록 구분
  - **가지치기(Pruning)**
    - 과적합 방지를 위해 자세하게 구분된 영역을 통합

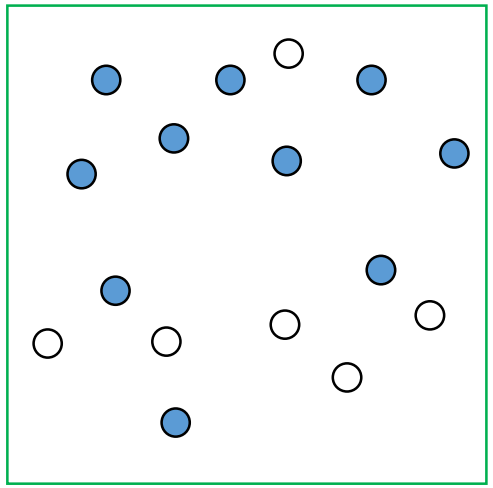
이 용어부터 알아봅시다

Decision Tree

- **Gini Index: 불순도를 나타내는 대표적인 index**
  - 단일 영역에 대한 Gini계수

$$I(A) = 1 - \sum_{k=1}^{m} p_k^2$$

    - 여기서 p_k: A영역에 속한 레코드 중 k범주에 속하는 레코드 비율
  - 예제



$$I(A) = 1 - \sum_{k=1}^{m} p_k^2$$

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

→모든 레코드가 동일 범주: $I(A) = 0$

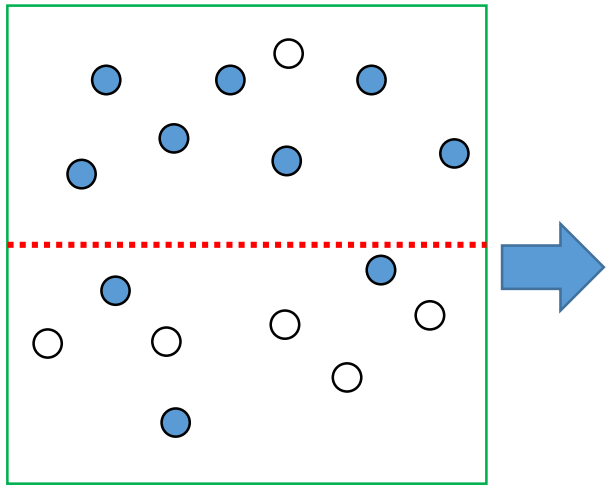
→두 범주에 동일 수의 데이터: $I(A) = 0.5$

Decision Tree

- **Gini Index: 불순도를 나타내는 대표적인 index**
  - 2개 이상의 영역에 대한 Gini계수

$$I(A) = \sum_{i=1}^{d} R_i \left(1 - \sum_{k=1}^{m} p_{ik}^2\right)$$

    - 여기서 R_i: 분할 전 레코드 중, 분할 후 i영역에 속하는 레코드의 비율
  - 예제



$$I(A) = \sum_{i=1}^{d} R_i \left(1 - \sum_{k=1}^{m} p_{ik}^2\right)$$

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

Decision Tree

- **Key Ideas**
  - **재귀적 분기(Recursive Partitioning)**
    - 한 속성의 모든 분기점에 대해서 Cost값을 계산

$$J = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

    - 예시
      - Var2에 대해서 0.6125을 기준으로 두 영역으로 구분시 cost

$$J = \frac{1}{20}\left(1 - \frac{1}{1}\right)^2 + \frac{19}{20}\left(1 - \left(\frac{6}{19}\right)^2 - \left(\frac{13}{19}\right)^2\right) \approx 0.41$$

Var1	Var2	Label
87.0	0.7973	1
82.8	0.7922	1
51.0	0.7917	0
64.8	0.7880	1
52.8	0.7777	0
93.0	0.7666	1
61.5	0.7599	0
43.2	0.7493	1
81.0	0.7445	0
69.0	0.7373	0
75.0	0.7149	1
110.1	0.7028	0
33.0	0.6963	0
66.0	0.6899	0
60.0	0.6600	0
49.2	0.6289	0
84.0	0.6245	0
108.0	0.6162	0
64.8	0.6145	0
85.0	0.6123	0

Decision Tree

- **Key Ideas**
  - **재귀적 분기(Recursive Partitioning)**
    - 한 속성의 모든 분기점에 대해서 Cost값을 계산

$$J = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

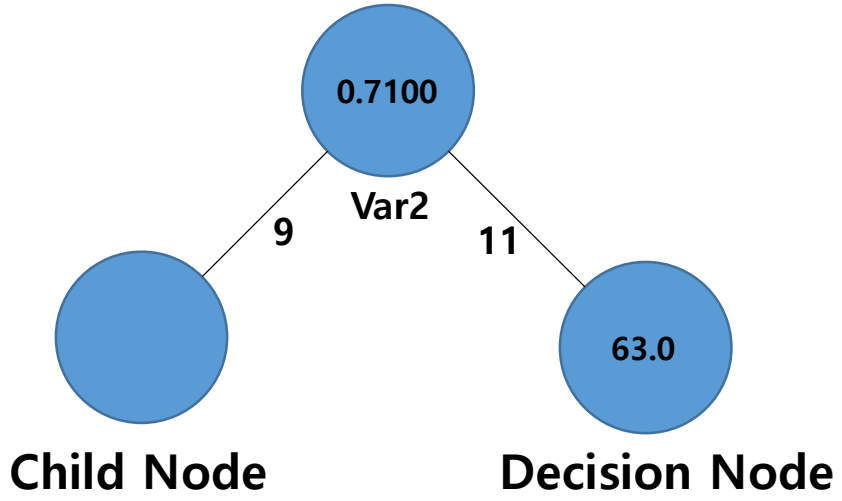
    - 예시
      - Var2에 대해서 0.7100을 기준으로 두 영역으로 구분시 cost

$$J = \frac{9}{20}\left(1 - \frac{9}{9}\right)^2 + \frac{11}{20}\left(1 - \left(\frac{6}{11}\right)^2 - \left(\frac{5}{11}\right)^2\right) \approx 0.23$$

Var1	Var2	Label
87.0	0.7973	1
82.8	0.7922	1
51.0	0.7917	0
64.8	0.7880	1
52.8	0.7777	0
93.0	0.7666	1
61.5	0.7599	0
43.2	0.7493	1
81.0	0.7445	0
69.0	0.7373	0
75.0	0.7149	1
110.1	0.7028	0
33.0	0.6963	0
66.0	0.6899	0
60.0	0.6600	0
49.2	0.6289	0
84.0	0.6245	0
108.0	0.6162	0
64.8	0.6145	0
85.0	0.6123	0

Decision Tree

- **Key Ideas**
  - **재귀적 분기(Recursive Partitioning)**
    - 이러한 과정을 반복해서 한 속성에 대한 최적 분기점 선택
    - Cost 값이 최소가 되는 분기점을 선택



Var1	Var2	Label
87.0	0.7973	1
82.8	0.7922	1
51.0	0.7917	0
64.8	0.7880	1
52.8	0.7777	0
93.0	0.7666	1
61.5	0.7599	0
43.2	0.7493	1
81.0	0.7445	0
69.0	0.7373	0
75.0	0.7149	1
110.1	0.7028	0
33.0	0.6963	0
66.0	0.6899	0
60.0	0.6600	0
49.2	0.6289	0
84.0	0.6245	0
108.0	0.6162	0
64.8	0.6145	0
85.0	0.6123	0

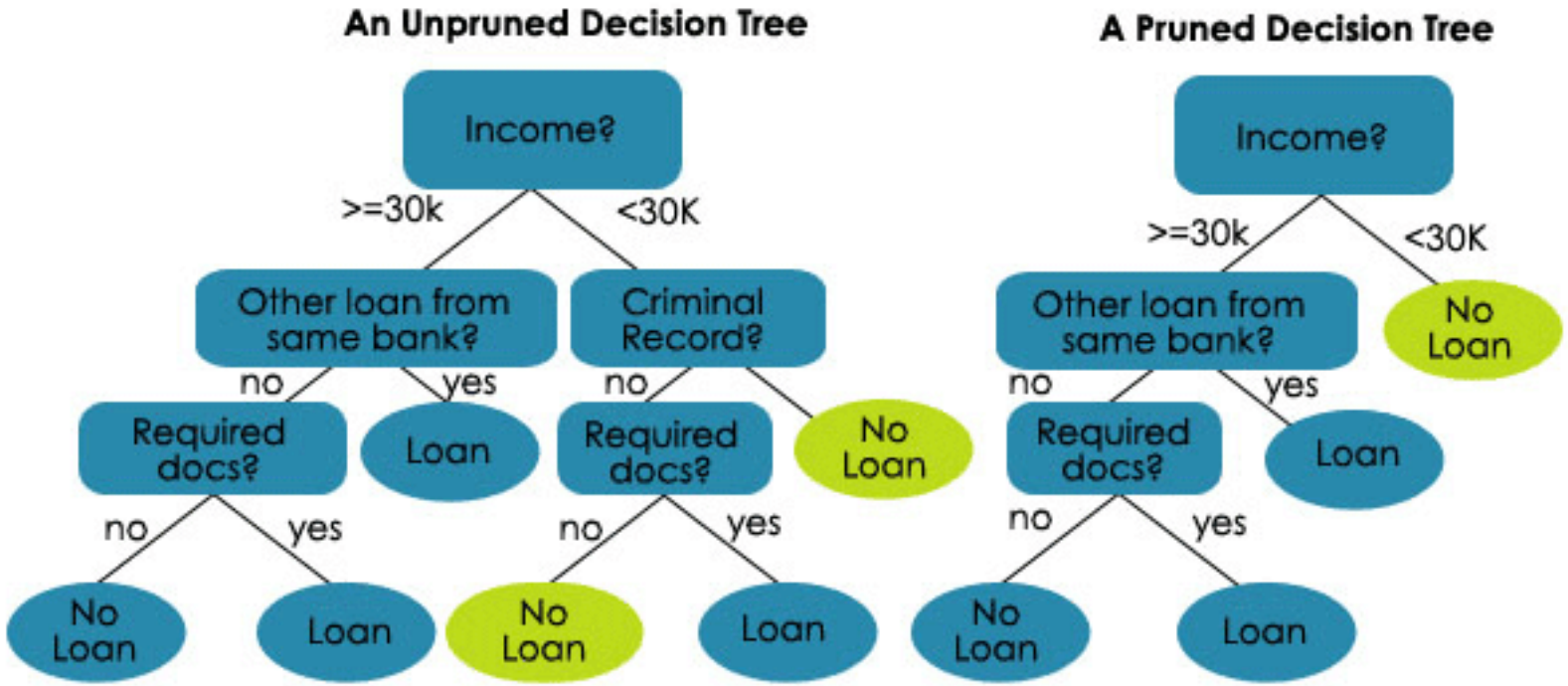
Decision Tree

- **Key Ideas**
  - **재귀적 분기(Recursive Partitioning)**
    - 이러한 과정을 반복해서 한 속성에 대한 최적 분기점 선택
    - Cost 값이 최소가 되는 분기점을 선택
    - 모든 노드의 순도가 100%가 될 때까지 분기 탐색 과정 수행
    - 재귀적 분기 완료
      - 사용자가 지정한 최대 깊이가 되거나
      - 불순도를 줄이는 분할을 찾을 수 없을 때 → 모든 노드 순도 100%

Var1	Var2	Label
87.0	0.7973	1
82.8	0.7922	1
51.0	0.7917	0
64.8	0.7880	1
52.8	0.7777	0
93.0	0.7666	1
61.5	0.7599	0
43.2	0.7493	1
81.0	0.7445	0
69.0	0.7373	0
75.0	0.7149	1
110.1	0.7028	0
33.0	0.6963	0
66.0	0.6899	0
60.0	0.6600	0
49.2	0.6289	0
84.0	0.6245	0
108.0	0.6162	0
64.8	0.6145	0
85.0	0.6123	0

Decision Tree

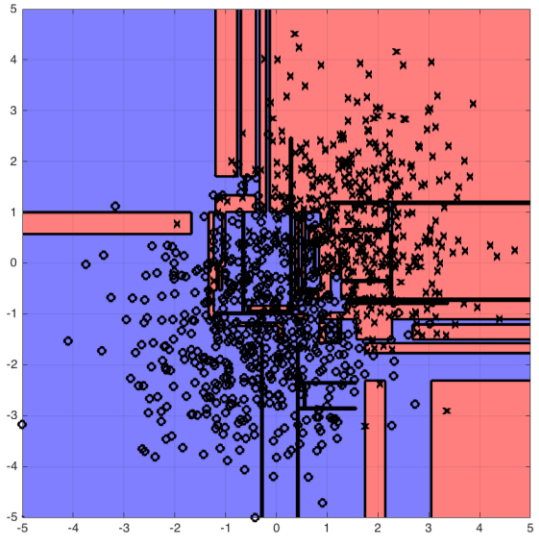
- **Key Ideas**
  - **가지치기(Pruning)**
    - 과적합 방지를 위해 자세하게 구분된 영역을 통합

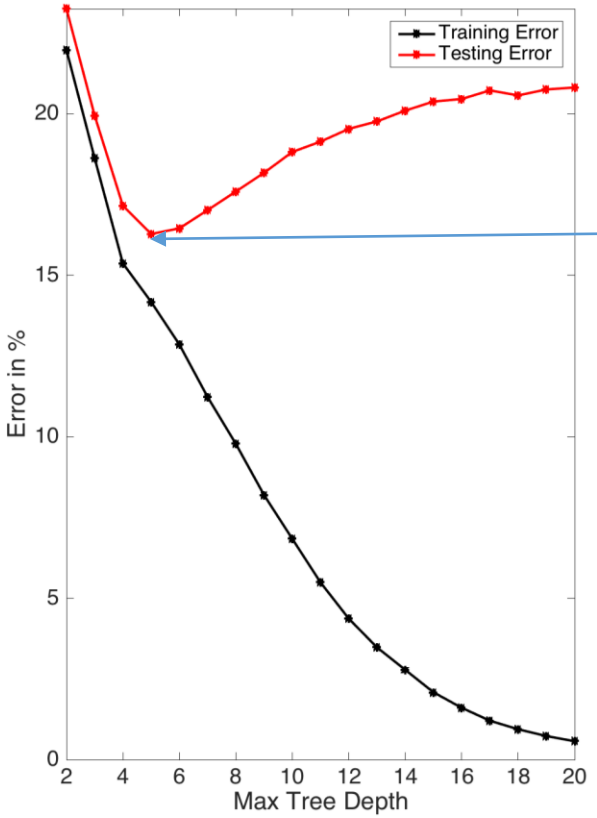


(출처: https://www.cloudyml.com/blog/decision-tree-pruning-techniques-in-python/)

Decision Tree

- **Key Ideas**
  - **가지치기(Pruning)**
    - 과적합 방지를 위해 자세하게 구분된 영역을 통합





최적 트리 깊이

(출처: http://www.cs.cornell.edu/courses/cs4780/2017sp/lectures/lecturenote17.html)

Decision Tree

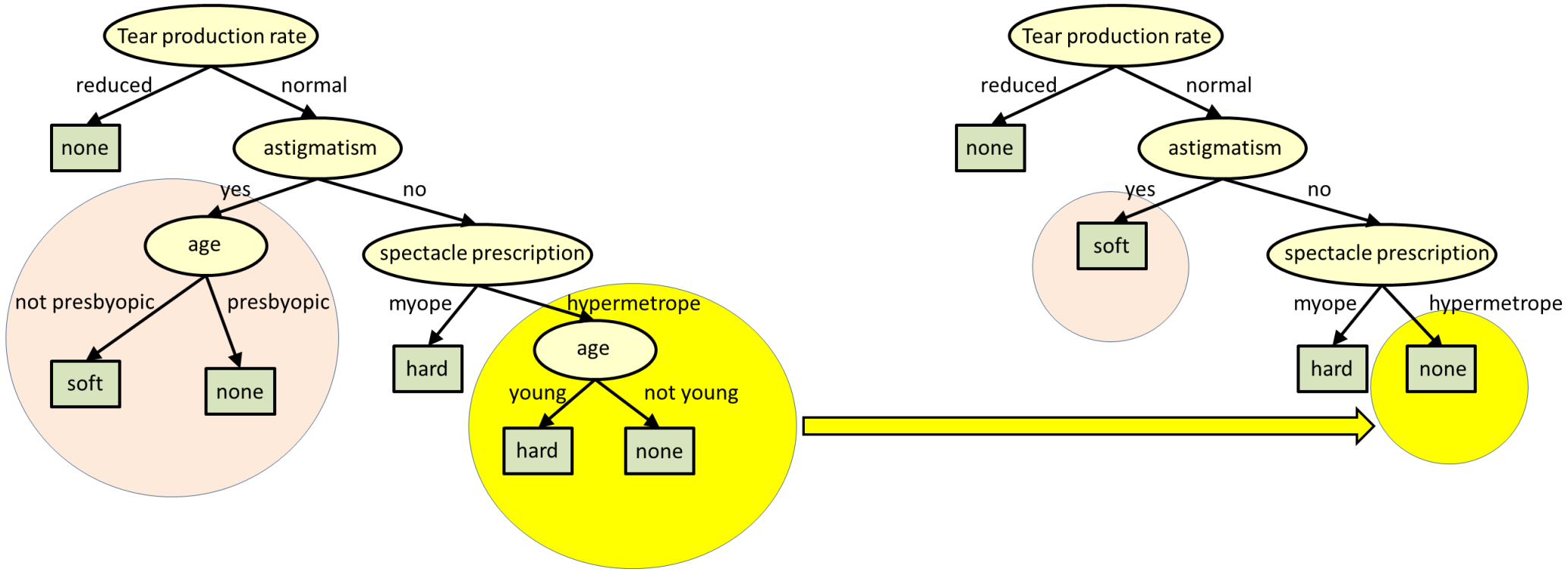
- **Key Ideas**
  - **가지치기(Pruning)**
    - Full Tree를 생성한 뒤, 적절한 수준에서 말단 노드를 결합하는 가지치기 수행
    - Validation data에 대한 오분류율이 증가하는 시점에서 가지치기
    - 다음과 같은 cost function을 최소화 하는 시점에서 pruning

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha|T|$$

$|T|$: Terminal Nodes 수
R_m: m번째 terminal node에 대응하는 predictor space의 subspace
$\hat{y}_{R_m}$: R_m에 속하는 training 관측치의 평균

Decision Tree

- **Key Ideas**
  - **가지치기(Pruning)**
    - 과적합 방지를 위해 자세하게 구분된 영역을 통합



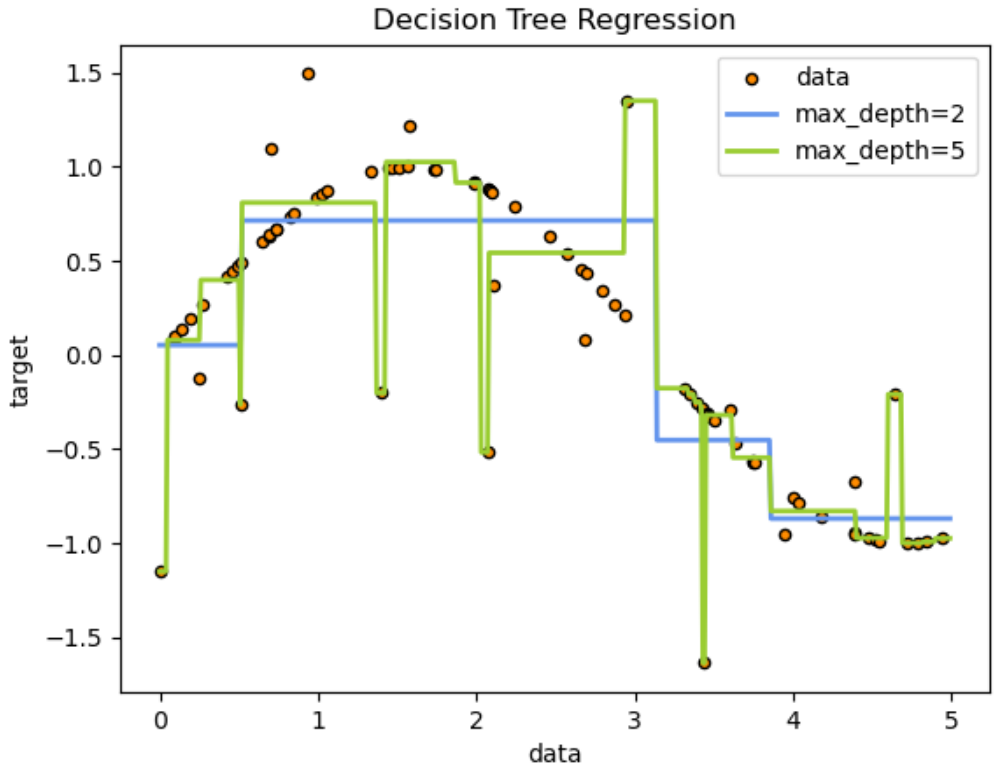
(출처: https://www.cs.cmu.edu/~bhiksha/courses/10-601/decisiontrees/)

Decision Tree

- **Decision Tree를 활용한 예측**
  - Decision Tree가 생성된 뒤, 각 leaf node에 속하는 레코드들은 training data의 레코드
  - 각 leaf node에 해당하는 범주는 해당 노드에 포함된 개체들이 속한 범주 비율로 판단
  - If cut-off = 0.5
    - 해당 leaf node에 속해있는 record들의 다수결 방식으로 범주 예측
  - If cut-off = 0.75
    - 해당 leaf node에 속해있는 record의 비율이 0.75이상일 때만 범주 예측

Decision Tree

- **Decision Tree를 활용한 회귀**
  - 사실 DT는 분류에만 사용되는게 아니라, 회귀(regression)에도 사용할 수 있습니다.
  - 이에 관해서는 '회귀'시간에 다루도록 하겠습니다.



(출처: https://scikit-learn.org/stable/modules/tree.html)

Summary

- **Decision Tree**
  - 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내 tree기반의 분류 규칙을 만드는 것
  - 데이터 전처리(범주형 변수의 벡터화, 정규화 등)가 상대적으로 적게 요구됨
  - Decision Tree를 학습하는 대표적인 방법으로 CART 알고리즘이 있음

- **CART algorithm**
  - 개별 변수의 영역을 반복적으로 분할, 전체 영역에서의 규칙을 생성하는 지도학습 기법
  - If/Then 형식으로 표현되는 규칙을 생성 → 결과 예측과 동시에 이유도 설명도 가능
  - 수치형 변수와 범주형 변수에 대한 동시 처리 가능
  - Key Idea
    - **재귀적 분기(Recursive Partitioning):** 한 속성을 2개의 부분으로 나누되, 순도가 증가하도록
    - **가지치기(Pruning):** 과적합 방지를 위해 자세하게 나눠진 부분을 통합

End of the documents