Introduction to Artificial Intelligence [AICS223]

Supervised Learning-Classification (W05)

Prof. Mee Lan Han (aeternus1203@gmail.com)

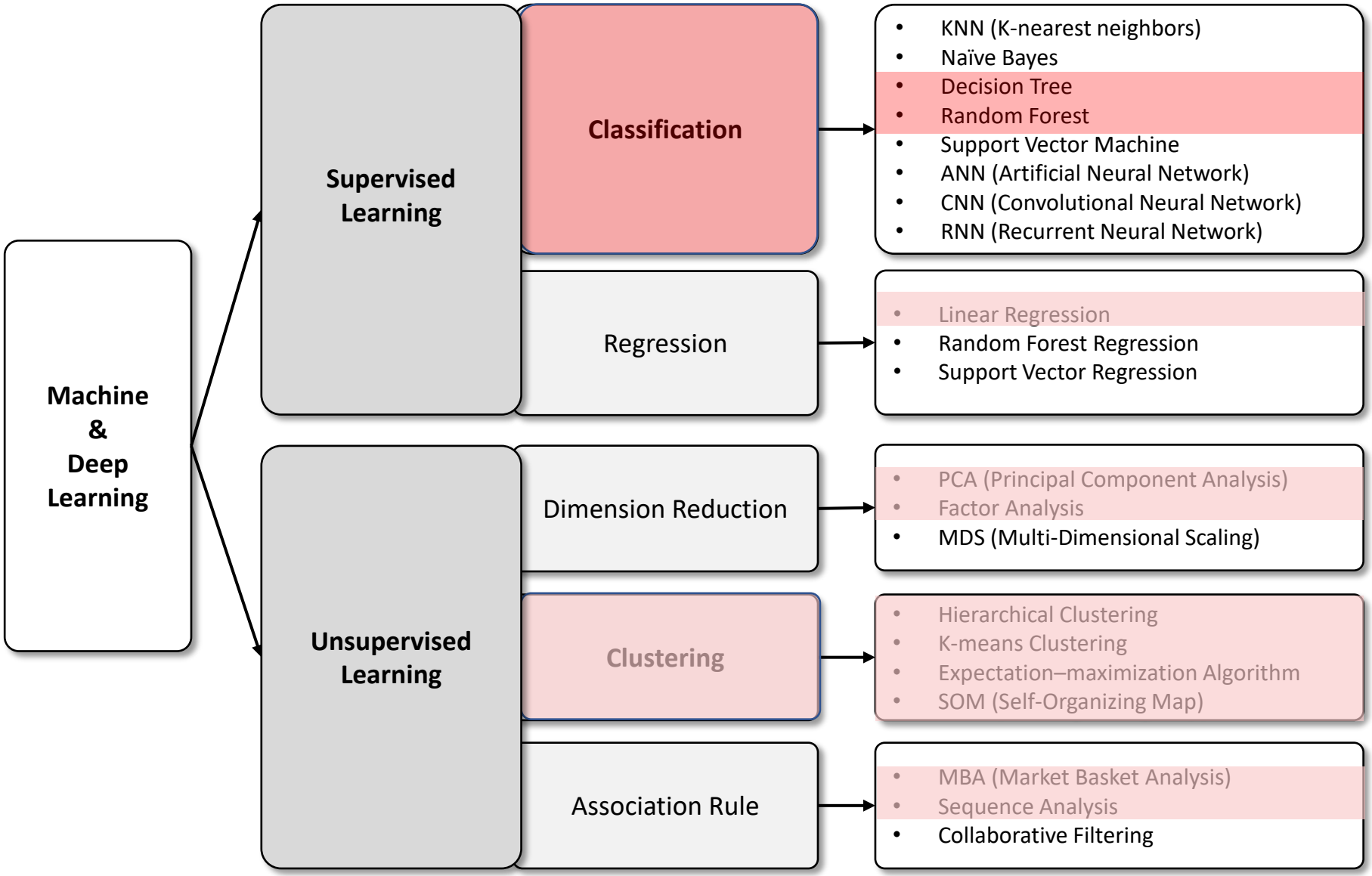
고려대학교

인공지능사이버보안학과

KOREA UNIVERSITY

CONTENTS

Machine Learning



Review

■ KNN (K-nearest neighbors)

□ 특정공간 내 새로 들어온 입력값이 어떤 그룹의 데이터와 가장 가까운가 분류

□ K의 역할은 몇 번째로 가까운 데이터까지 살펴볼 것인가를 설정하는 수치

- 장점
  - 높은 정확도
  - 단순하며 효율적 (모델 생성을 미리 하지 않음)
- 단점
  - 데이터가 많을 수록 처리 시간이 증가
  - 모델 생성이 미리 되지 않아 새로운 데이터에 대한 학습 시간보다 분류/예측 시간이 더 걸림

Review

■ SVM(Support Vector Machine)

□ 주어진 데이터가 어느 카테고리에 속할지 판단하는 이진 선형 분류 모델

□ 결정 경계 (Decision Boundary), 즉 분류를 위한 기준선을 정의하는 모델

□ **Support Vectors**: 결정 경계와 가까이 분포된 데이터를 의미

- n개의 Feature (속성)을 가진 데이터에는 최소 n+1개의 서포트 벡터가 존재

□ **Margin**: 결정 경계와 서포트 벡터 사이의 거리를 의미

□ **커널 (Kernel)**

- 선형으로 분류할 수 없는 데이터 특성을 갖고 있는 경우, 원래 가지고 있는 데이터를 더 높은 차원의 데이터로 변환하는 역할

□ 장점

- SVM은 결정 경계를 정의하는 서포트 벡터만 잘 골라내면 됨
- 모든 데이터를 확인할 필요가 없어 속도가 매우 빠름

Review

■ One-class Classification

□ 정상 샘플만을 이용해 모델을 학습

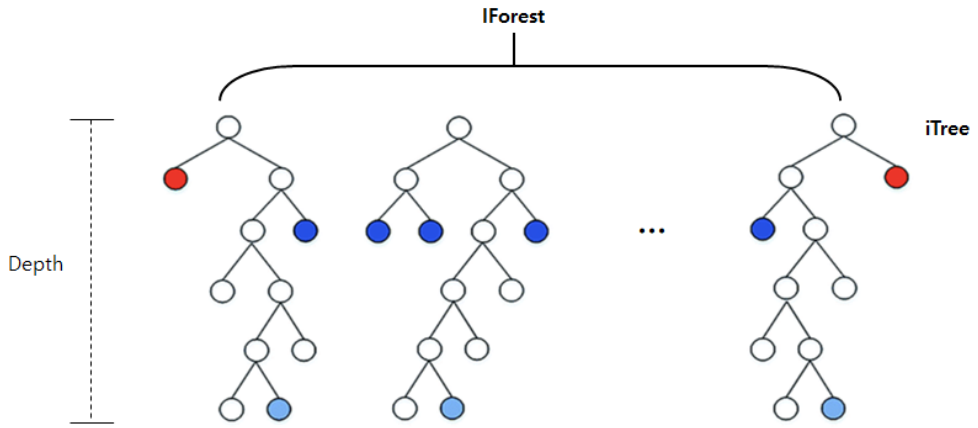
□ 정상 샘플과 거리가 먼 샘플을 이상치로 탐지하는 준지도(Semi-supervised) 학습 방법

□ One-class SVM

- One-class SVM은 SVM과는 달리 **비지도 학습 (unsupervised learning)**

□ Isolation Forest

- 기본적으로 데이터셋을 의사결정나무 (Decision Tree) 형태로 표현
- 랜덤으로 데이터를 split하여 모든 관측치를 고립 (분리) 시키며 구현
- 데이터의 Depth를 기준으로 정상값과 이상값을 분리
  - **정상 데이터:** tree의 terminal node와 근접하며, 경로길이 (Depth)가 큼
  - **이상치**: **tree의 root node와 근접**하며, **경로길이 (Depth)가 작음**



여러개의 iTree로 구성

Decision Tree

■ 의사결정나무 (Decision Tree)

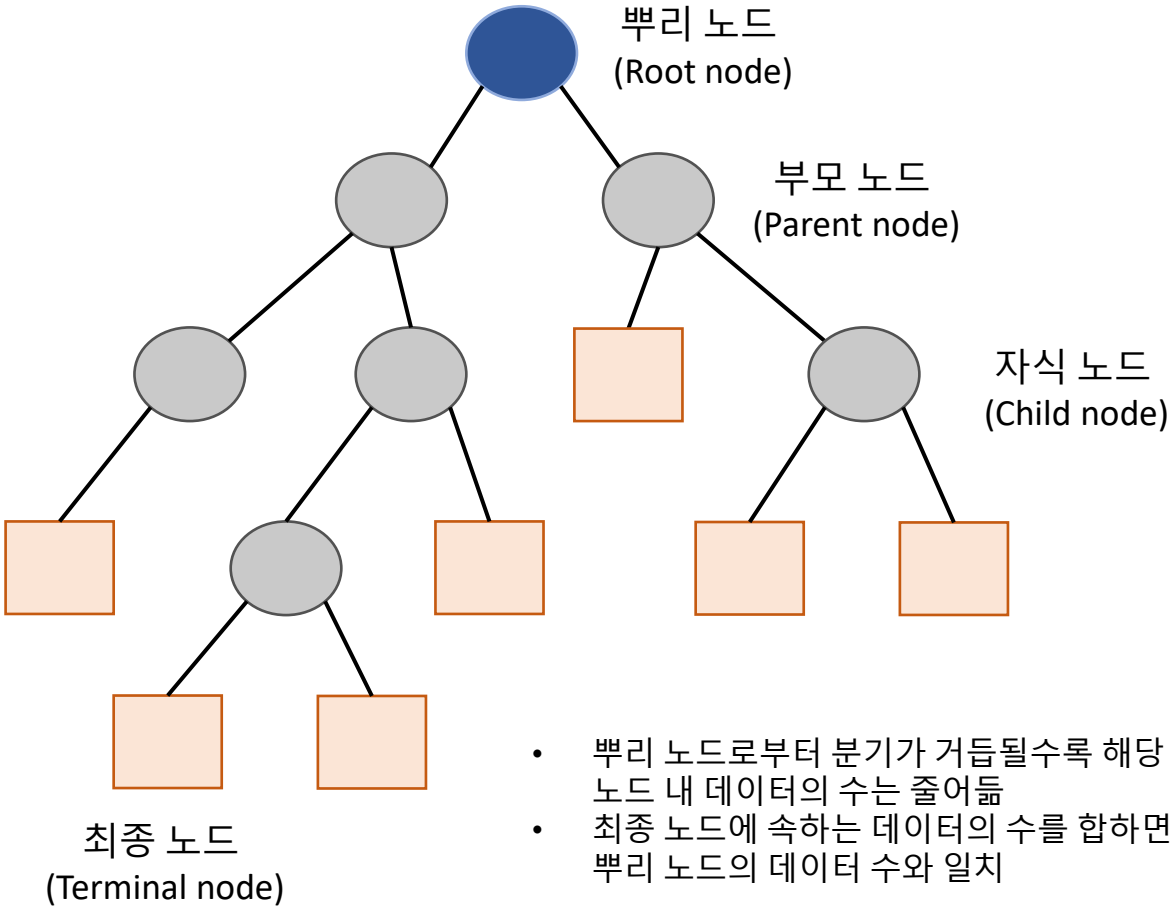
□ 의사결정규칙 (Decision Rule)을 나무구조로 도표화하여 분류와 예측을 수행하는 분석 방법

나무의 가지를 생성하는 과정

분할 (Split)

생성된 가지를 잘라 모형을 단순화하는 과정

가지치기 (Pruning)



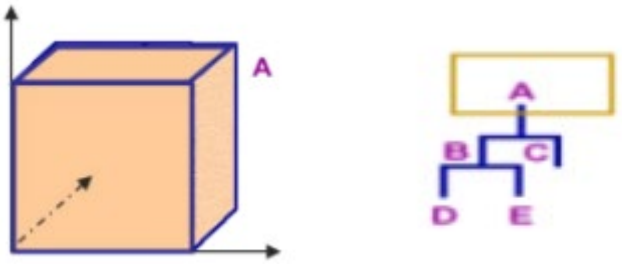
- 뿌리 노드로부터 분기가 거듭될수록 해당 노드 내 데이터의 수는 줄어듦
- 최종 노드에 속하는 데이터의 수를 합하면 뿌리 노드의 데이터 수와 일치

Decision Tree

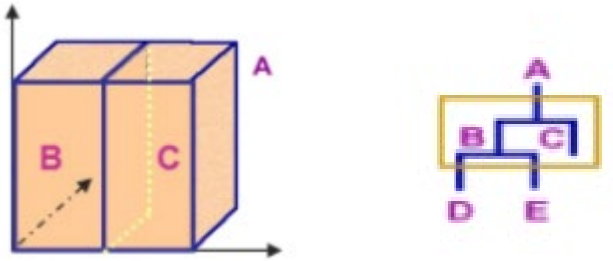
■ 의사결정나무 (Decision Tree)

□ 의사결정규칙 (Decision Rule)을 나무구조로 도표화하여 분류와 예측을 수행하는 분석 방법

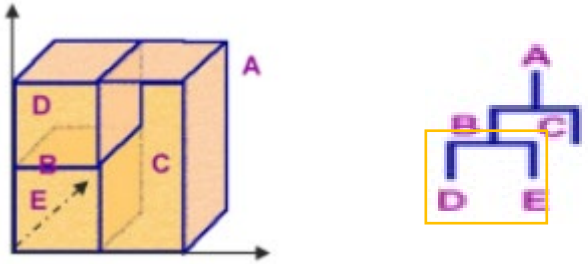
- 전체 A 데이터



- 전체 A 데이터 가 B와 C 데이터로 분할



- B 데이터가 D와 E 데이터로 분할



Decision Tree

■ 의사결정나무 (Decision Tree)

□ 의사결정규칙 (Decision Rule)을 나무구조로 도표화하여 분류와 예측을 수행하는 분석 방법

■ 장점

□ 학습된 결과에 대한 **해석 가능**
□ 데이터를 분석하고 표현하는데 있어 **매우 직관적인 방식**
□ 범주형 자료와 수치형 자료 모두 수행 가능
□ 정규화 또는 더미 변수 생성 없이 처리 가능

■ 단점

□ 데이터의 특성이 특정 변수에 **수직/수평적으로 구분되지 못할 때 분류율이 떨어짐**
□ **연속형 변수를 비연속적 값으로 취급**
-> 분리 경계점에서 예측 오류가 발생할 가능성 존재
-> 분할해야 하는 곳이 많아 분할할 최적 값을 검색하는데 많은 시간이 소요

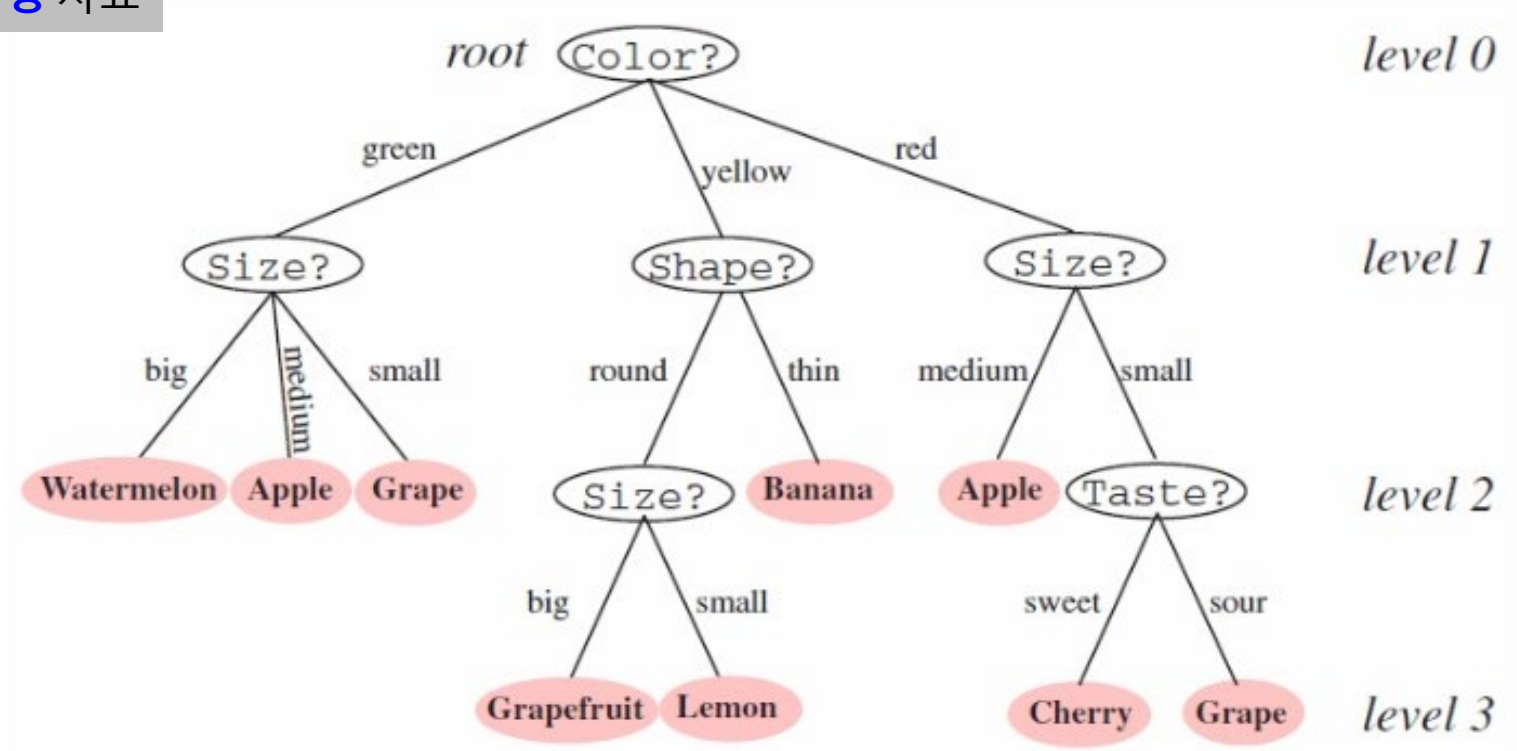
□ Training data가 적으면 overfitting, over-classified 발생

Decision Tree

■분류변수와 분류 기준값

□ 목표변수의 형태에 따라 상위노드에서 가지분할을 수행할 시 분류 (기준) 변수와 분류 기준값의 선택

범주형 자료

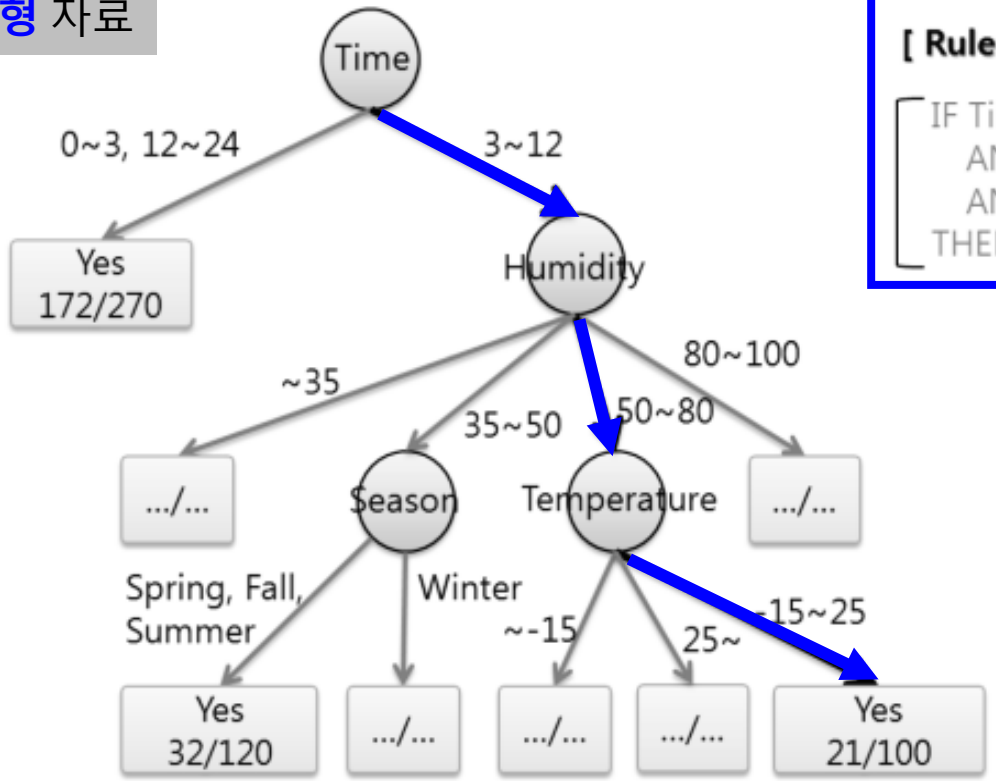


Decision Tree

■분류변수와 분류 기준값

□ 목표변수의 형태에 따라 상위노드에서 가지분할을 수행할 시 분류 (기준) 변수와 분류 기준값의 선택

수치형 자료



[Rule]

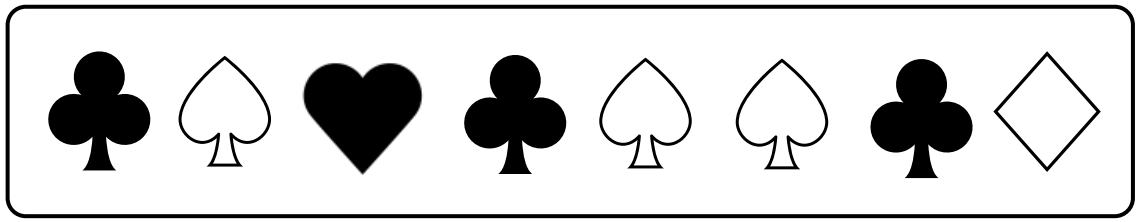
IF Time in [3~12]
AND Humidity in [50~80]
AND Temperature in [-15~25]
THEN **Class=Yes(0.21)**

Decision Tree

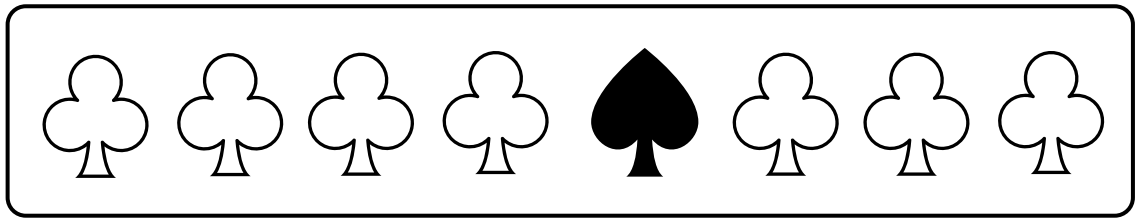
■ 의사결정 트리의 분할 속성

□ 분할된 데이터의 불순도 제거 정도에 따라 속성(트리)과 속성값(기준값) 결정

- **순수도** : 특정 범주의 개체들이 포함되어 있는 정도
- **불순도 (불확실성)** : 다양한 범주들의 개체들이 포함되어 있는 정도



높은 이질성 -> 낮은 순수도



낮은 이질성 => 높은 순수도

□ **의사결정 트리의 분할 속성 선택**

- 부모노드의 순수도 대비 자식노드들의 순수도가 증가하도록 노드를 형성
- **순수도 증가, 불순도가 최대치로 감소하는 방향으로 학습을 진행**

Decision Tree

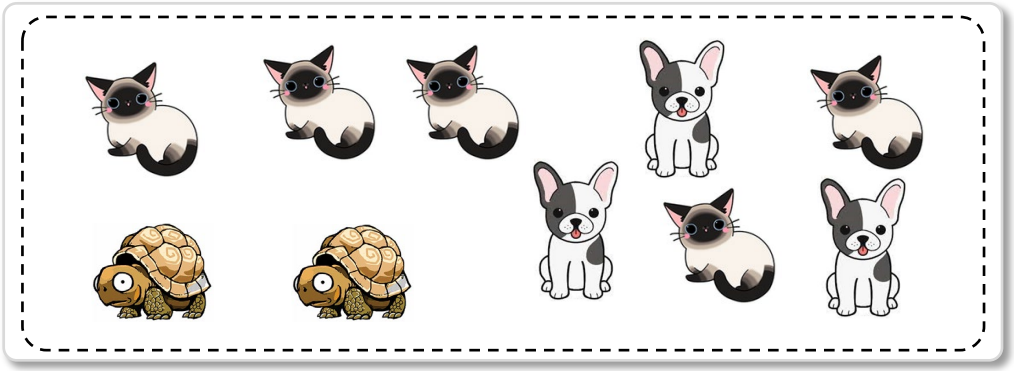
■분할 기준 (순도 측정) 방법

(1) 지니 지수 (Gini index)

- **불순도를 측정**하는 하나의 지수
- 0 (하나의 클래스로만 구성된 상태)에서 1/m(가장 혼합도가 높은 상태의 값)의 범위를 갖음
- 두 개의 범주 개체가 50대 50으로 구성될 때, 최대치의 불순도 값 측정

$$G(A) = 1 - \sum_{i=1}^{m} {P_i}^2$$

P_i는 데이터 A에서 i class 에 속하는 관측 비율 $(0 \leq G \leq 1/2)$



[Class 3개]

2 turtles 3 dogs 5 cats

자식 노드 생성을 위한 기준
Ex. 포유류인가 아닌가?

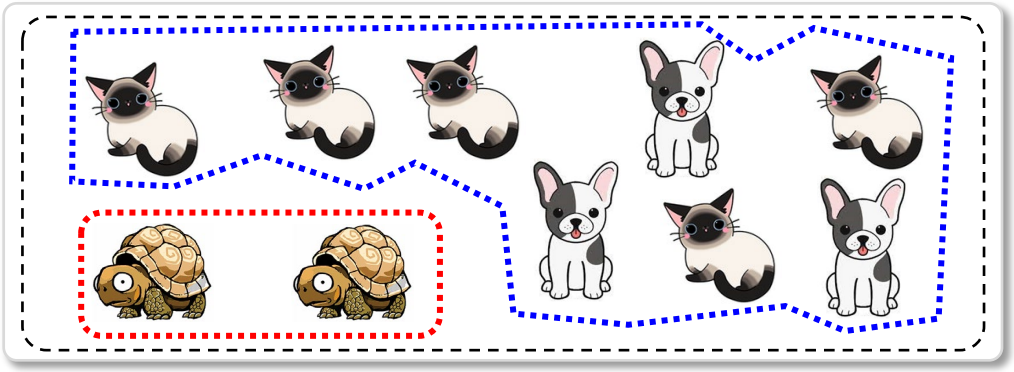
Decision Tree

■분할 기준 (순도 측정) 방법

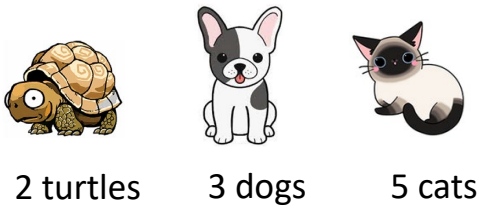
(1) 지니 지수 (Gini index)

- **불순도를 측정**하는 하나의 지수
- 0 (하나의 클래스로만 구성된 상태)에서 1/m(가장 혼합도가 높은 상태의 값)의 범위를 갖음
- 두 개의 범주 개체가 50대 50으로 구성될 때, 최대치의 불순도 값 측정

$$G(A) = 1 - \sum_{i=1}^{m} {P_i}^2$$ P_i는 데이터 A에서 i class 에 속하는 관측 비율 $(0 \leq G \leq 1/2)$



[Class 3개]



2 turtles 3 dogs 5 cats

자식 노드 생성을 위한 기준
Ex. 포유류인가 아닌가?

$$G(A) = 1 - \sum_{i}^{m} {P_i}^2 = 1 - \left[\left(\frac{8}{10}\right)^2 + \left(\frac{2}{10}\right)^2\right] = 1 - (0.64 + 0.04) = 0.32$$

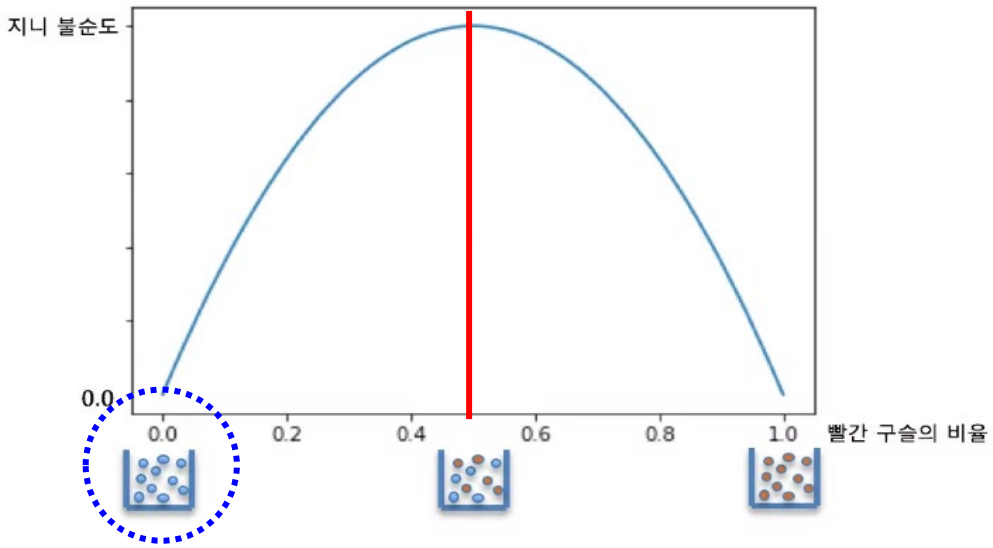
Decision Tree

■ 분할 기준 (순도 측정) 방법

(1) 지니 지수 (Gini index)

- **불순도를 측정**하는 하나의 지수
- 0 (하나의 클래스로만 구성된 상태)에서 1/m(가장 혼합도가 높은 상태의 값)의 범위를 갖음
- 두 개의 범주 개체가 50대 50으로 구성될 때, 최대치의 불순도 값 측정

$$G(A) = 1 - \sum_{i=1}^{m} {P_i}^2$$ P_i는 데이터 A에서 i class 에 속하는 관측 비율 $(0 \leq G \leq 1/2)$

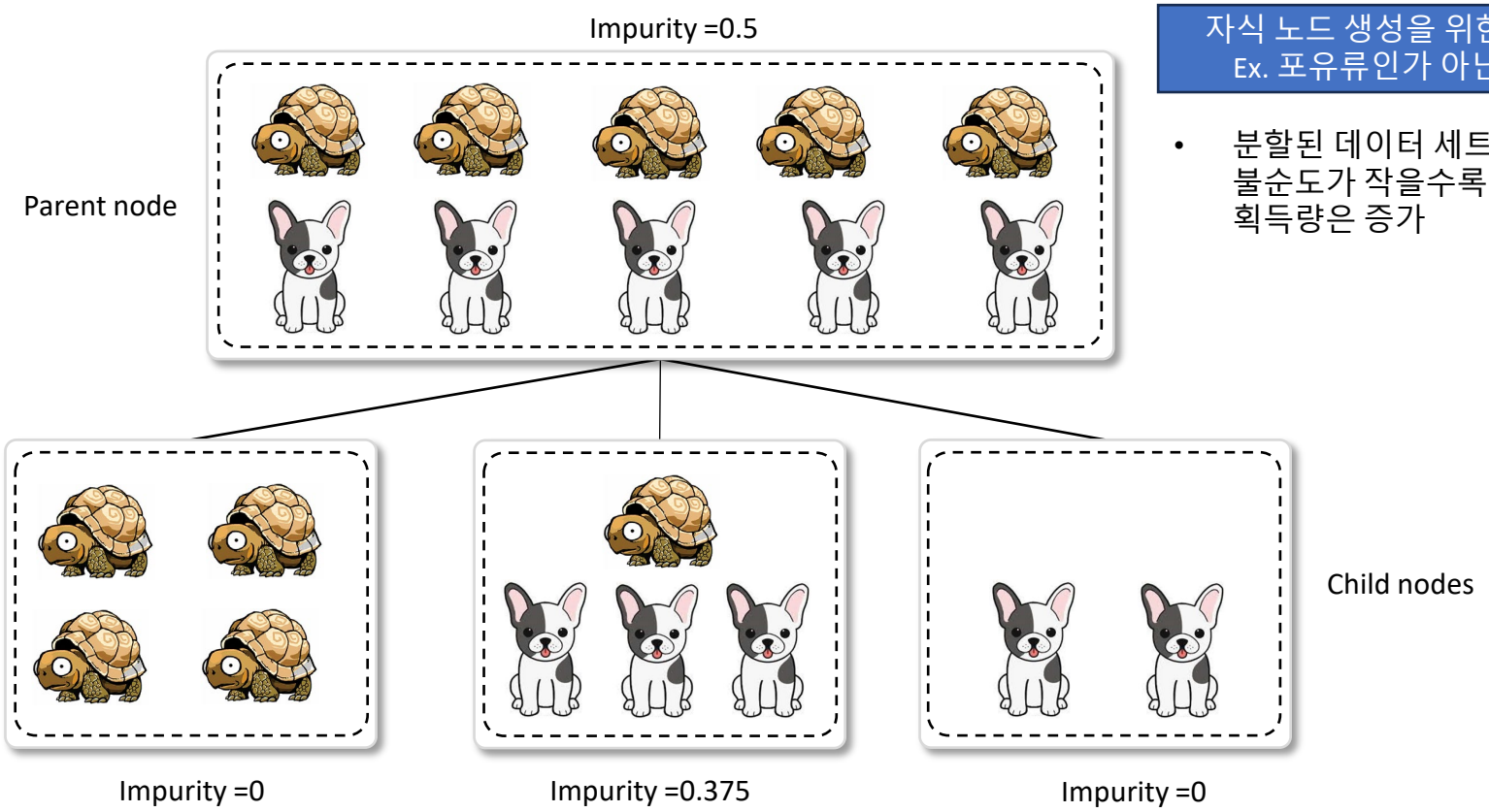


Decision Tree

■분할 기준 (순도 측정) 방법

(2) 정보 획득량 (Information gain)

- 지니지수를 구할 수 있다면 정보 획득량도 계산이 가능
- Information Gain = 0.5 – (0 + 0.375 + 0) = 0.125
- **이전 단계 (level) 불순도 값에서 다음 단계 (level) 불순도 값의 합을 뺀 수치**



자식 노드 생성을 위한 기준
Ex. 포유류인가 아닌가?

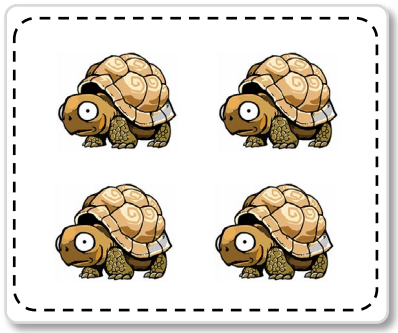
- 분할된 데이터 세트들의 불순도가 작을수록 정보 획득량은 증가

Decision Tree

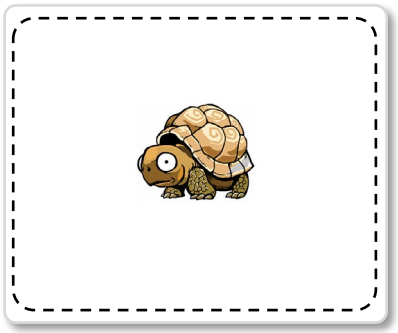
■분할 기준 (순도 측정) 방법

(2) 정보 획득량 (Information gain)

- 하위 데이터 셋들의 불순도만으로 정보 획득량을 설명하기 부족한 측면이 존재
  -> 가중치 (weight) 적용



불순도=0



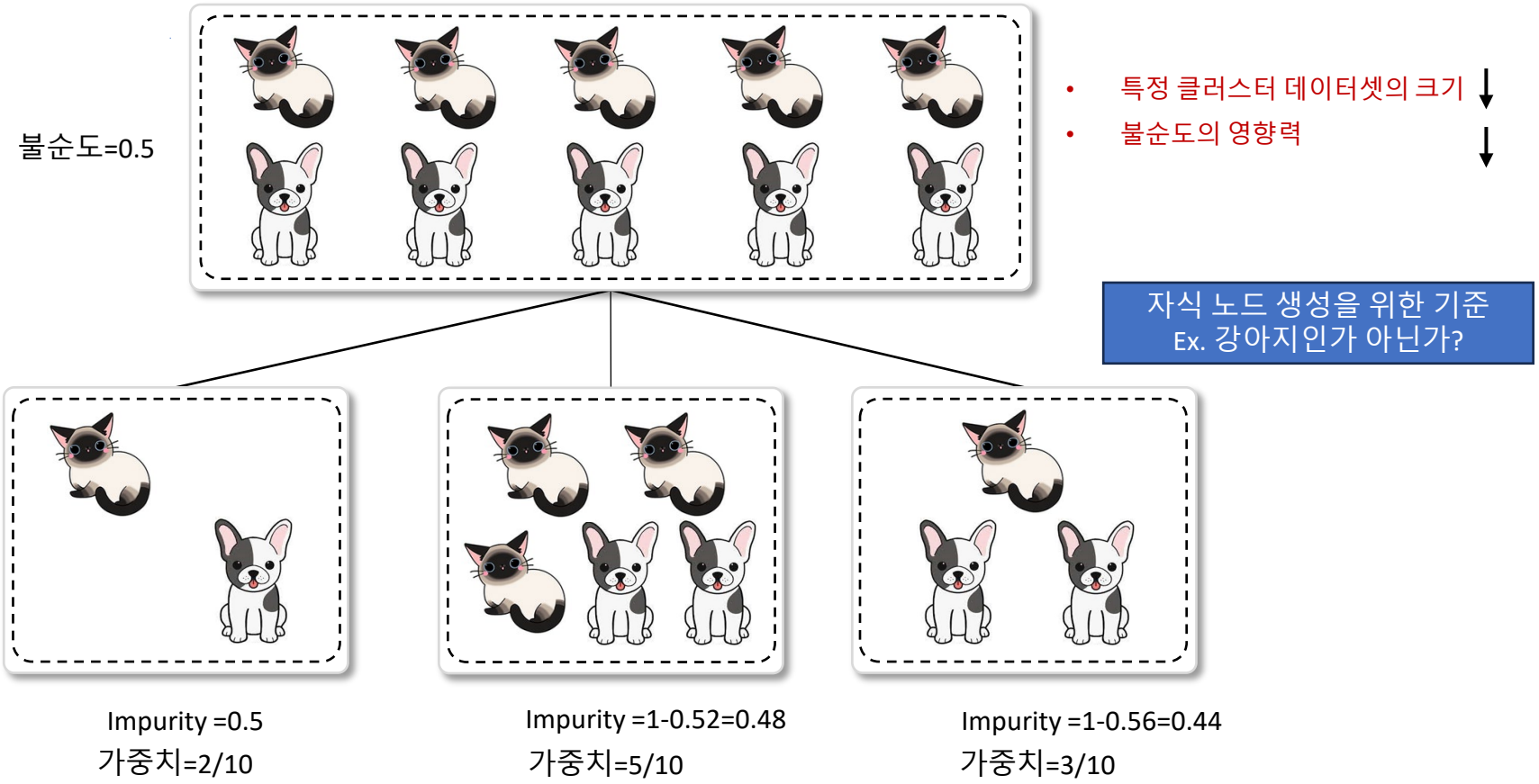
불순도=0

- 두 개의 데이터 셋 모두 불순도 0
- **왼쪽 데이터셋의 분류가 더 의미가 있어 보임**
- **불순도 값 뿐만 아니라 데이터셋 크기도 중요**
- 데이터셋의 크기에 따라 가중치 적용하여 **Weighted Information Gain** 을 측정해 볼 수 있음

Decision Tree

■분할 기준 (순도 측정) 방법

(2) 정보 획득량 (Information gain)



Information Gain = 0.5 – ((2/10)*0.5 + (5/10)*0.48 + (3/10)*0.44) = 0.026

Decision Tree

■분할 기준 (순도 측정) 방법

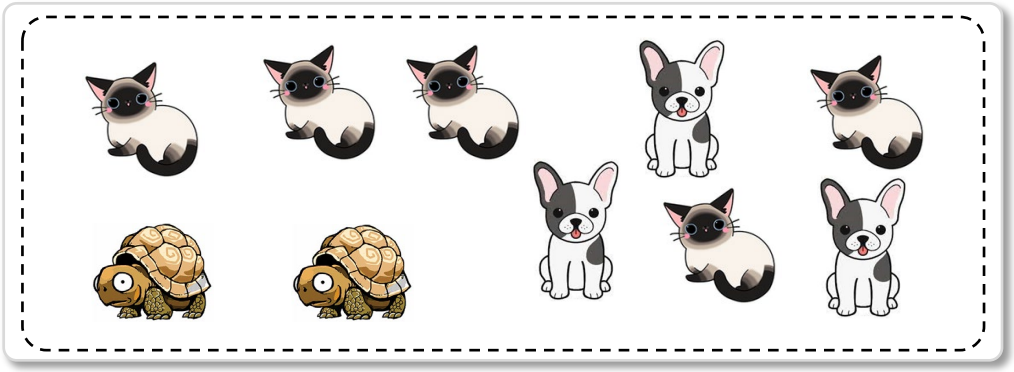
(3) 엔트로피 지수 (Entropy index)

- 열역학에서 나온 개념으로써 **무질서도 (분산도)** 의미
- 지니 지수와 비슷하지만, log를 취함으로써 정규화 과정을 거치게 됨
- 의사 결정 트리는 분기 뒤 각 영역의 **순수도가 증가 / 불확실성(엔트로피)가 최대한 감소**하도록 하는 방향으로 학습을 진행

$$Entropy(A) = -\sum_{i=1}^{m} P_i \, log_2(P_i)$$

P_i는 데이터 A에서 i class 에 속하는 관측 비율

$(\mathbf{0} \leq \boldsymbol{E} \leq \mathbf{1})$



[Class 3개]

2 turtles　3 dogs　5 cats

자식 노드 생성을 위한 기준?
포유류인가 아닌가?

Decision Tree

■분할 기준 (순도 측정) 방법

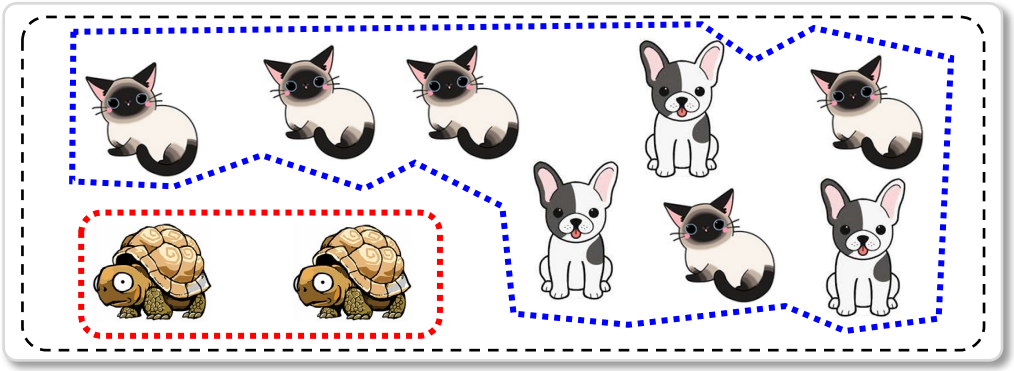
(3) 엔트로피 지수 (Entropy index)

- 열역학에서 나온 개념으로써 **무질서도 (분산도)** 의미
- 지니 지수와 비슷하지만, log를 취함으로써 정규화 과정을 거치게 됨
- 의사 결정 트리는 분기 뒤 각 영역의 **순수도가 증가 / 불확실성(엔트로피)가 최대한 감소**하도록 하는 방향으로 학습을 진행

$$Entropy(A) = -\sum_{i=1}^{m} P_i \, log_2(P_i)$$

P_i는 데이터 A에서 i class 에 속하는 관측 비율

$(\mathbf{0 \le E \le 1})$



[Class 3개]

2 turtles　3 dogs　5 cats

자식 노드 생성을 위한 기준
Ex. 포유류인가 아닌가?

$$Entropy(A) = -\sum_{i=1}^{m} P_i \, log_2(P_i) = [-0.8 * log_2(0.8)] + [-0.2 * log_2(0.2)] = 0.721928095$$

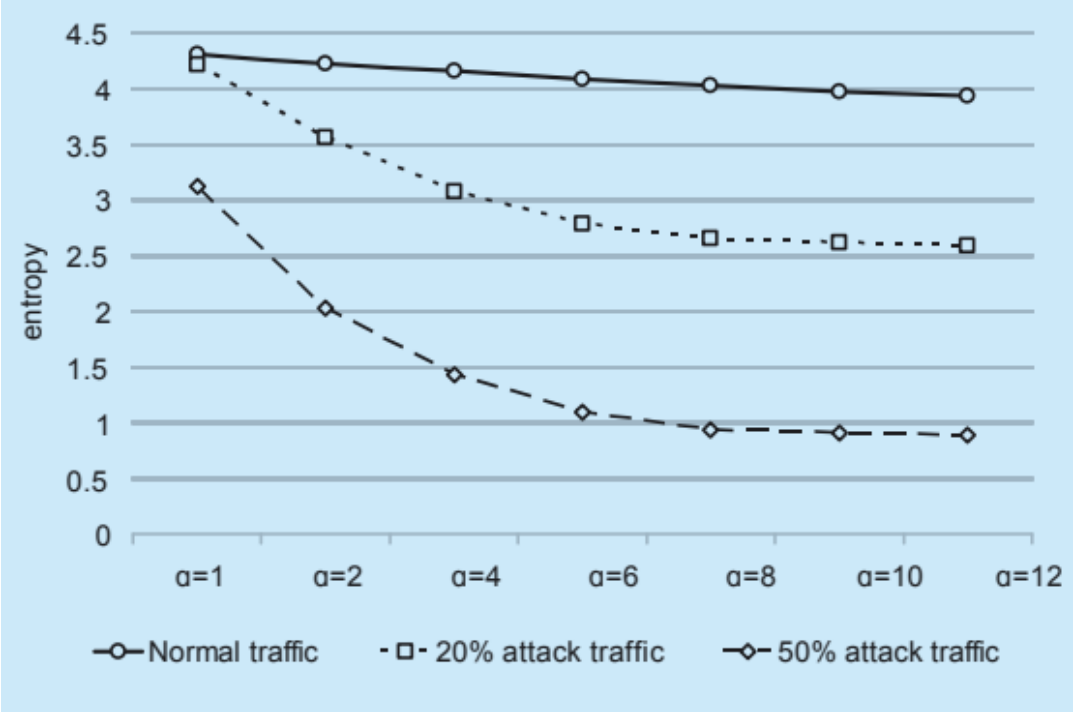
Gini index 최고값이 나오는 상태와 유사하게 데이터 셋 구성 (5:5)에 가까울수록, Entropy 값은 높아짐

Decision Tree

■분할 기준 (순도 측정) 방법

(3) 엔트로피 지수 (Entropy index)

- 열역학에서 나온 개념으로써 **무질서도 (분산도)** 의미
- 지니 지수와 비슷하지만, log를 취함으로써 정규화 과정을 거치게 됨
- 의사 결정 트리는 분기 뒤 각 영역의 **순수도가 증가 / 불확실성(엔트로피)가 최대한 감소**하도록 하는 방향으로 학습을 진행



Normal Traffic
&
Attack Traffic
??

Decision Tree

■ 의사결정나무 (Decision Tree) 의 속성

□ 재귀적 분기 (Recursive partitioning)

- 입력 변수 영역을 두 개로 구분하는 속성

□ 가지치기 (Pruning)

- 자세하게 분류된 최종 노드의 영역을 다시 통합하는 속성

Decision Tree

■ 재귀적 분기 (Recursive partitioning)

□ 의사결정 트리에서는 정보 획득량 (Information Gain)이 큰 순서대로 질문을 배치

□ 질문 속성에 대해 어떤 기준으로 나누는 게 좋을지 반복적으로 적용 -> **최적의 트리 검색**

(1)

1 / 23

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

(2)

2 / 22

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

Input 값을 2분할 한 모든 조합에 대해 엔트로피/지니지수를 재귀적으로 측정
→ 트리를 만들기 위한 과정

- Lot size (대지 면적) 기준으로 정렬
- 데이터 2등분 (노랑:빨강)
- 각 분기의 엔트로피 계산

- 2) 와 같이 shifting 하여 데이터 2등분
- 각 분기의 엔트로피 계산
- ....
- 마지막 23)과 같이 shifting 하여 데이터 2등분
- 각 분기의 엔트로피 계산

(23)

23 / 1

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

Decision Tree

■ 재귀적 분기 (Recursive partitioning)

□ 의사결정 트리에서는 정보 획득량 (Information Gain)이 큰 순서대로 질문을 배치

□ 질문 속성에 대해 어떤 기준으로 나누는 게 좋을지 반복적으로 적용 -> **최적의 트리 검색**

총 행	좌측 행	우측 행	좌측 Non-owner	좌측 Owner	우측 Owner	우측 Non-owner	엔트로피
24	1	23	1	0	12	11	0.96
24	2	22	2	0	12	10	0.91
24	3	21	3	0	12	9	0.86
24	4	20	4	0	12	8	0.81
24	5	19	4	1	11	8	0.84
24	6	18	5	1	11	7	0.79
24	7	17	5	2	10	7	0.83
24	8	16	6	2	10	6	0.77
24	9	15	7	2	10	5	0.71
24	10	14	7	3	9	5	0.76
24	11	13	8	3	9	4	0.69
24	12	12	9	3	9	3	0.61
24	14	10	10	4	8	2	0.58
24	15	9	10	5	7	2	0.65
24	16	8	10	6	6	2	0.72
24	17	7	11	6	6	1	0.62
24	18	6	11	7	5	1	0.70
24	19	5	11	8	4	1	0.77
24	20	4	12	8	4	0	
24	21	3	12	9	3	0	
24	22	2	12	10	2	0	
24	23	1	12	11	1	0	

- 엔트로피 값이 가장 낮음

  → 즉, 순수도가 가장 높음
- 14/10 등분일 때 이 지점을 기준으로 데이터를 split 하여 새로운 노드를 생성함
- 분기된 노드들은 다시 재귀적 분기를 반복하면서 최종 트리 구조를 생성함

Decision Tree

■ 가지치기 (Pruning)

□ **재귀적 분귀를 통해 모든 노드를 분리한 후 분기를 적절히 합치는 과정**

□ 분기가 많아질 경우 학습데이터에 과대적합 (Overfitting)이 발생할 우려 존재

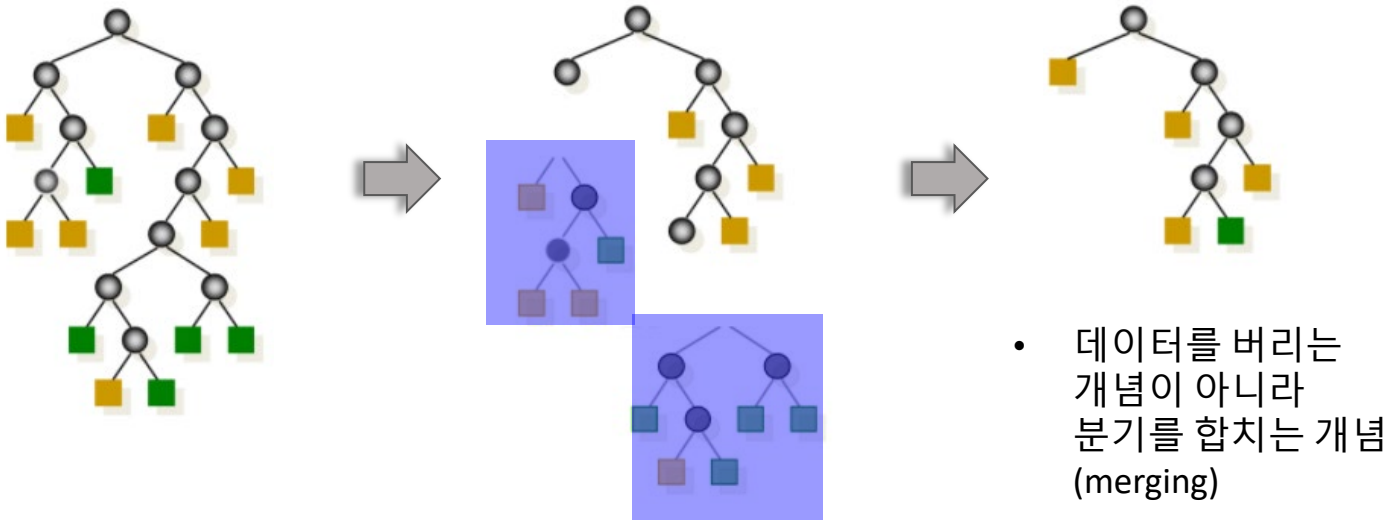
□ 분기수가 증가할 때 초기에는 오분류표 (confusion matrix: TP, TN, FP, FN) 증가

- **Accuracy, Precision, Recall 값이 증가**

□ 이후 분기수가 증가할 때마다 오분류표 (confusion matrix: TP, TN, FP, FN) 증가

- **Accuracy, Precision, Recall 값이 감소**

⇒ 적절한 가지치기 수행이 요구됨 (Reduced error pruning, Rule post pruning)



- 데이터를 버리는 개념이 아니라 분기를 합치는 개념 (merging)

Decision Tree

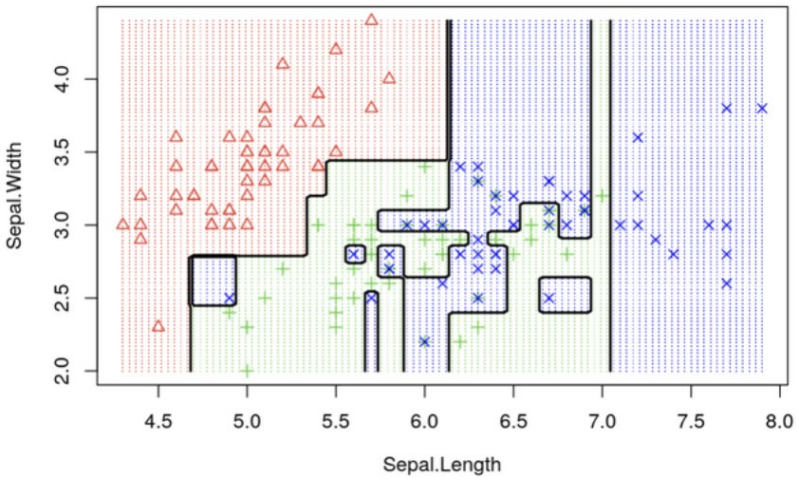
■ 가지치기 (Pruning)

□ Reduced error pruning

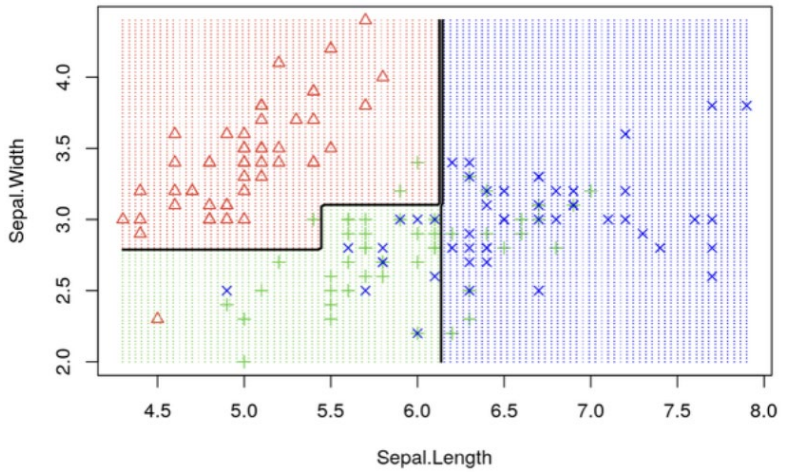
- 노드 가지치기 전의 error와 노드의 아래 부분을 자르거나 결합한 뒤 error를 비교
- 오류가 더 이상 감소하지 않을 때까지 반복하는 방법

□ Rule post pruning

- Rule: 뿌리 노드부터 최종 노드까지의 경로
- 트리를 Rule 형태로 변환한 뒤 각 Rule 의 Accuracy 를 구하고 낮은 순으로 제거하는 방법



Full tree



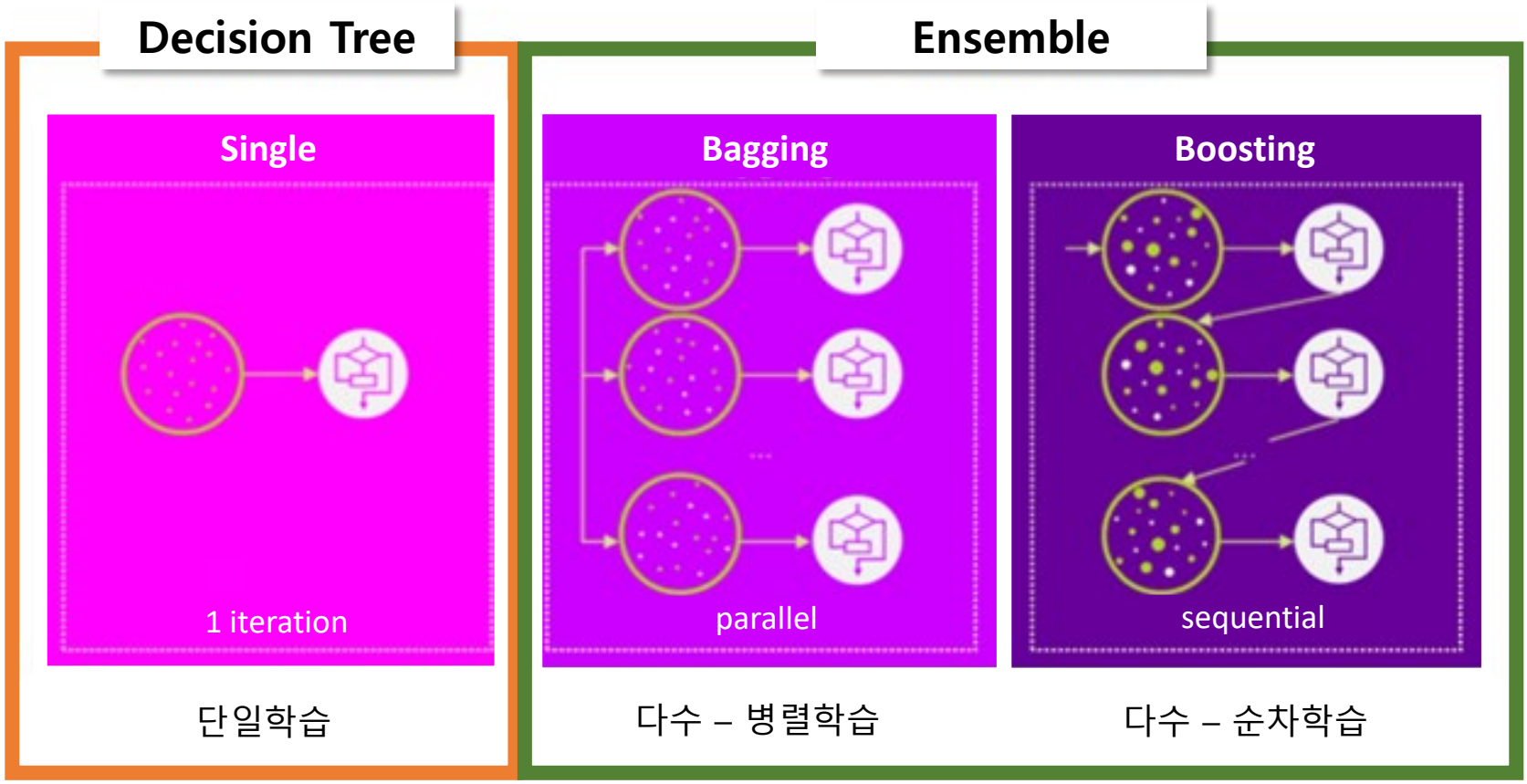
적절한 가지치기를 한 뒤 Tree

Ensemble 알고리즘 (RF & XGBoost)

■ Ensemble 알고리즘

□ 여러 가지 우수한 학습 모델을 조합해 분류&예측 결과를 향상시키는 모델

□ 배깅 (Bagging), 부스팅 (Boosting)



Random Forest

■ Random Forest (랜덤포레스트)

□ 분류, 회귀 분석 등에 사용되는 앙상블 학습 방법의 일종

□ 훈련 과정에서 구성한 다수의 결정 트리로부터 분류 또는 평균 예측치 (회귀 분석)를 출력함으로써 동작

□ 다수의 결정 트리들을 학습하는 앙상블 방법

■ Random Forest 핵심!!!

□ **Diversity (다양성)**

- 여러 개의 Training Data를 생성해 각 데이터마다 개별 Decision Tree를 구축

□ **Random Subspace**

- Decision Tree 구축 시 변수를 무작위로 선택

Random Forest

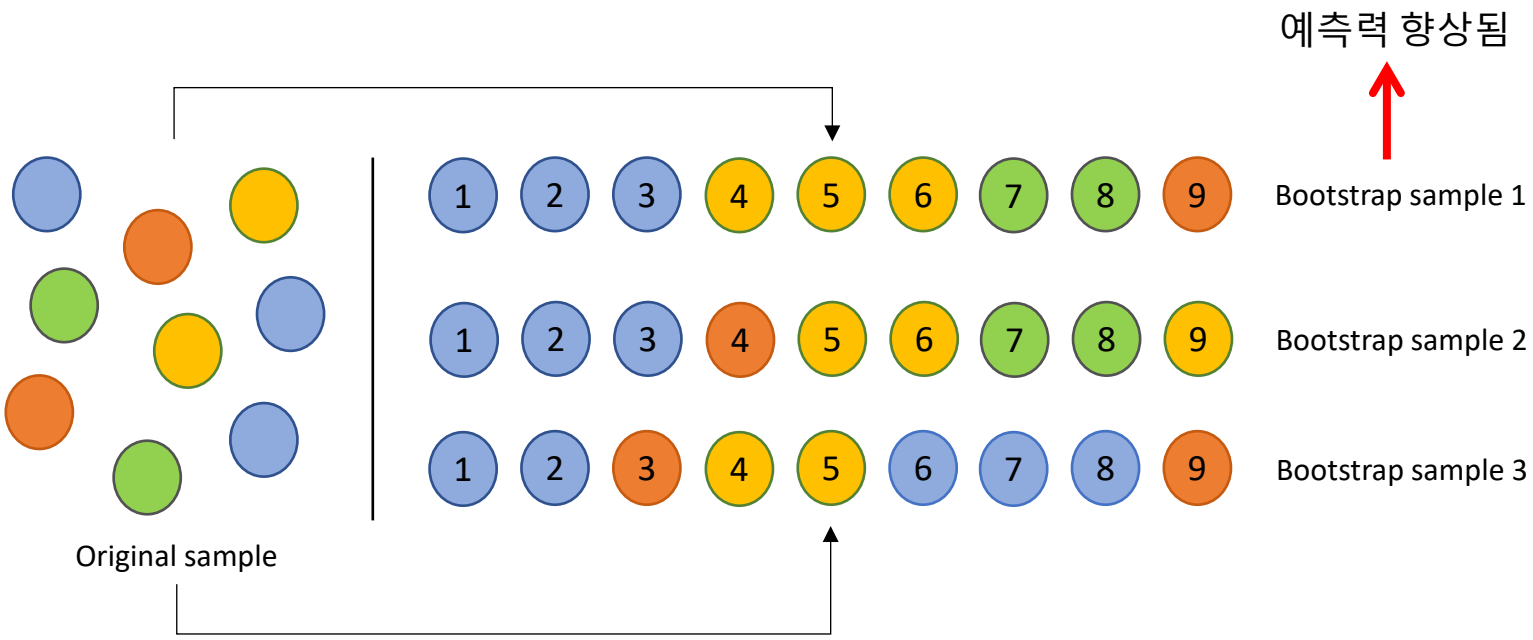
■ 배깅 (Bagging = Bootstrap + Aggregating)

□ **Bootstrap**: 통계적 재표본추출(resampling) 방법

□ Original sample 데이터로부터 다수의 Bootstrap 자료를 생성하고 모델링한 후 결합하여 예측 모델을 산출

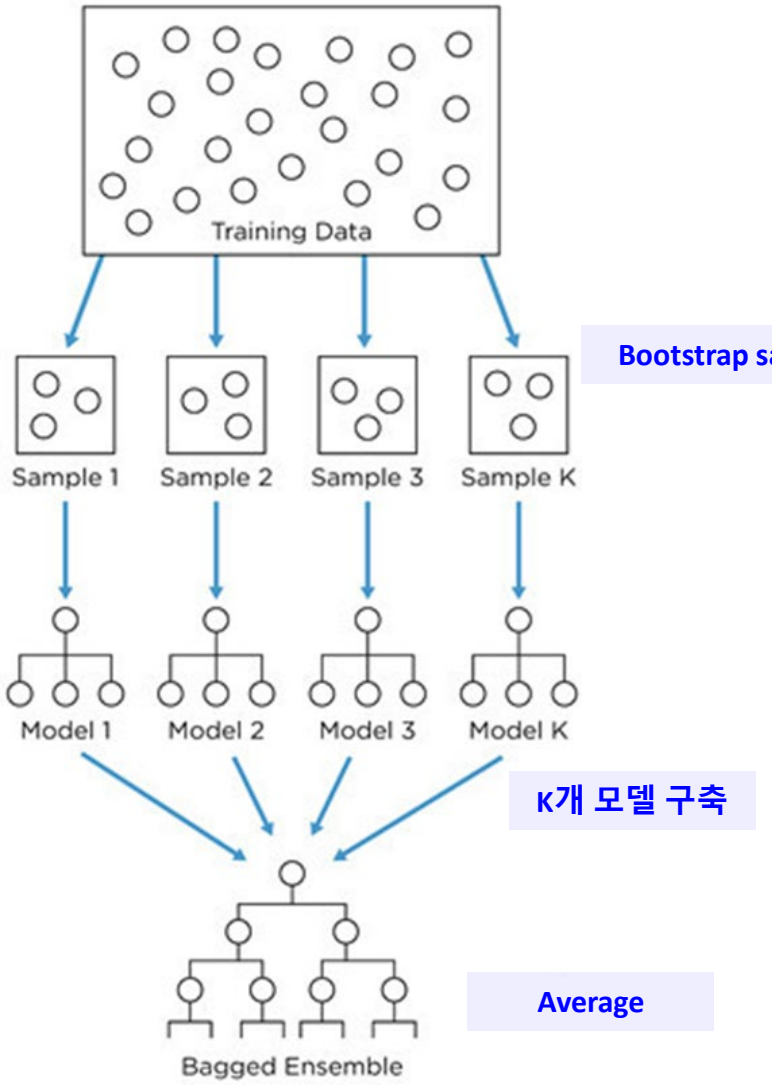
□ Bootstrap 자료

- 랜덤 샘플링을 통해 원시자료 (Raw data)로부터 크기가 동일한 여러 표본 자료를 의미함



Random Forest

■ 배깅 (Bagging = Bootstrap + Aggregating)



특징

- 여러 개의 결정트리 (Decision Tree)를 활용한 배깅 방식의 대표적인 알고리즘
- 각 트리 마다 랜덤하게 데이터를 샘플링

장점/단점

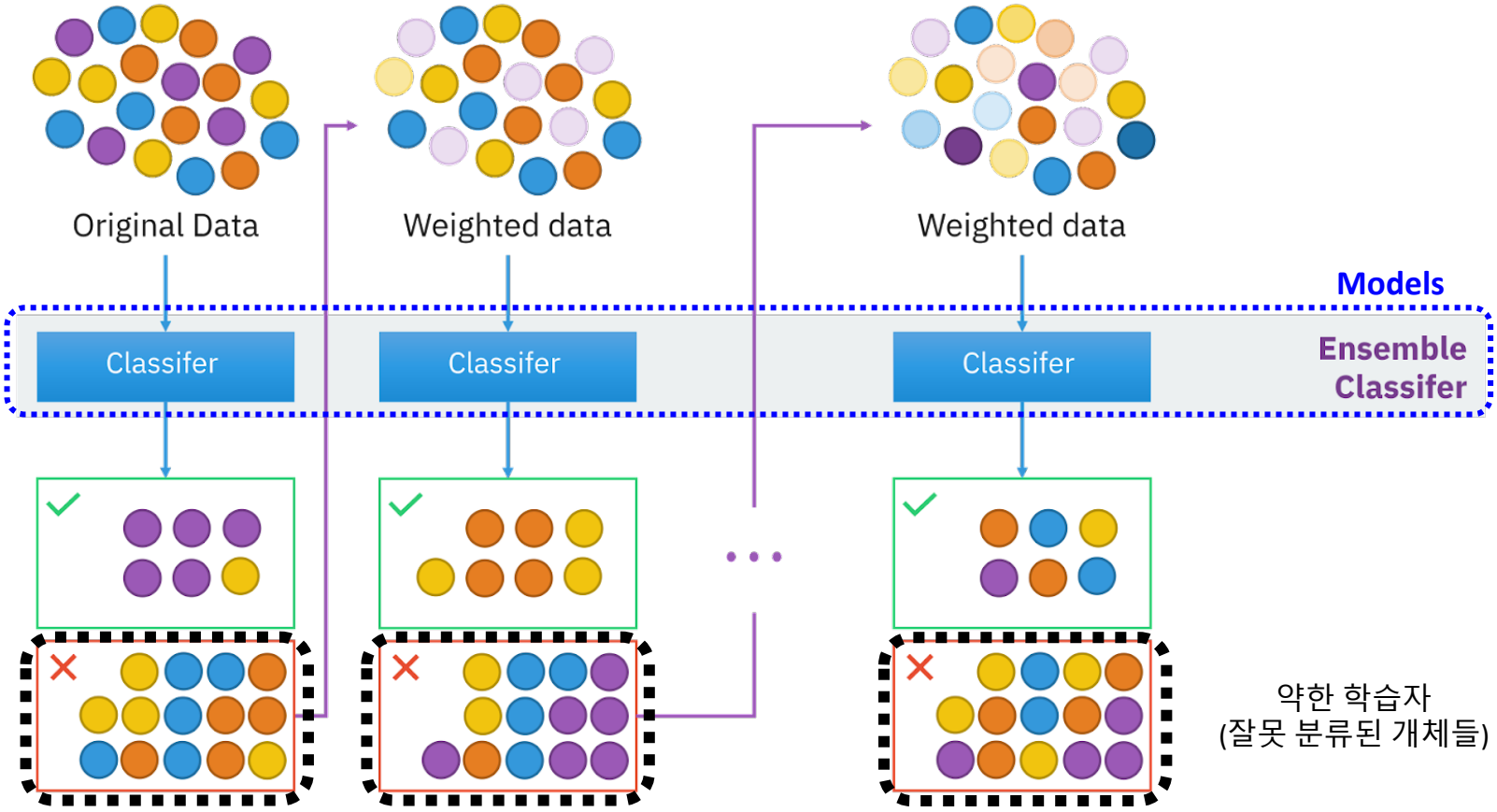
- 결정 트리의 쉽고 직관적인 장점을 그대로 가지고 있음
- 앙상블 알고리즘 중 비교적 빠른 수행 속도를 가지고 있음
- 다양한 분야에서 좋은 성능을 나타냄
- 각각의 개별 트리는 과적합 (Overfitting)될 수 있음
- 하이퍼 파라미터가 많아 튜닝을 위한 시간이 많이 소요됨

Boosting

■ 부스팅 (Boosting)

□ 순차적으로 **약한 학습자 (잘못 분류된 개체들)**를 추가 결합하여 모델을 만드는 방법

□ "무작위로 선택" 의미보다는 **약한 것들을 여러 개 결합시켜 강한 모델을 생성**



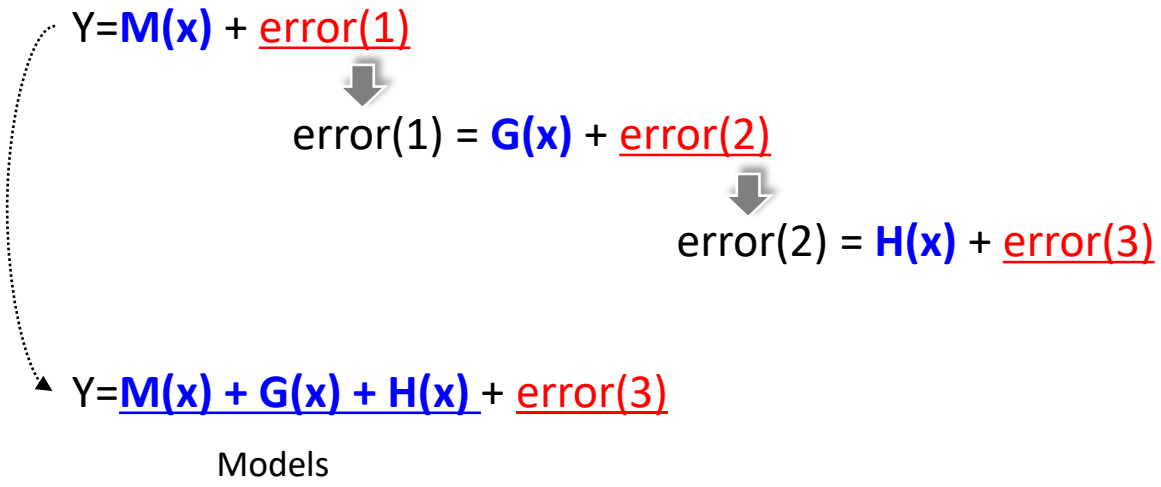
- [Reference] https://en.wikipedia.org/wiki/Boosting_(machine_learning)#/media/File:Ensemble_Boosting.svg

Boosting

■ 부스팅 (Boosting)

□ 순차적으로 **약한 학습자 (잘못 분류된 개체들)**를 추가 결합하여 모델을 만드는 방법

□ "무작위로 선택" 의미보다는 **약한 것들을 여러 개 결합시켜 강한 모델을 생성**



① Raw data 의 개체에는 동일한 가중치로 시작

② 모델링을 통한 예측변수에 의해 **잘못 분류된 개체들에는 높은 가중치 부여**

③ **제대로 분류된 개체들에는 낮은 가중치 부여**

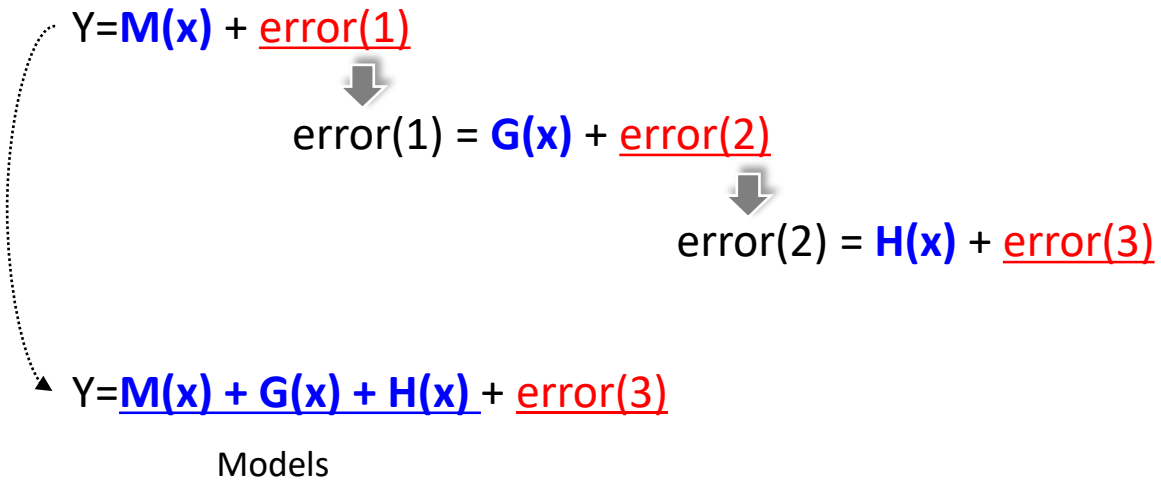
④ 잘못 분류된 개체들을 더욱 잘 분류되도록 하는 방법

Boosting

■ 부스팅 (Boosting)

□ 순차적으로 **약한 학습자 (잘못 분류된 개체들)**를 추가 결합하여 모델을 만드는 방법

□ "무작위로 선택" 의미보다는 **약한 것들을 여러 개 결합시켜 강한 모델을 생성**

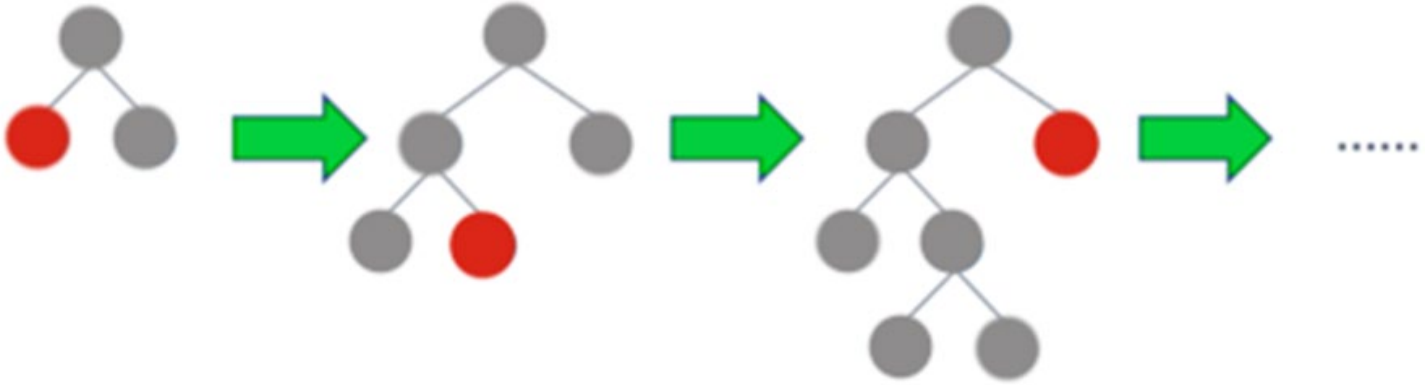


- 학습 모델 M, G, H 각각의 성능이 다를 수 있지 않을까?
- 각 학습 모델에 **Weights 를 두어 최적의 학습 모델을 도출**
- 학습 모델 M만을 단독 사용하는 것 보다는 최종 모델 (M+ G + H)이 정확도 높음

$$Y=\alpha*M(x) + \beta*G(x) + \gamma*H(x) + error(3)$$

XGBoost & LightGBM

XGBoost: Level-wise tree growth



LightGBM: Leaf-wise tree growth

XGBoost

■ Hyperparameter tuning

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.1, max_delta_step=0, max_depth=5,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

□ General parameter
- General parameters determine whether to use a tree or a linear model when performing boosting.

□ Boost parameter
- It is parameters related to tree optimization, boosting, regularization, etc.

□ Parameter for Learning
- It determine the learning scenario.

XGBoost

■ Hyperparameter tuning

□ General parameter

- booster [Default = **gbtree**]
  - 의사결정기반모형 (gbtree)
  - 선형모형 (gblinear)

- n_jobs
  - Number of parallel threads used to run XGBoost

- verbosity [**Default = 1**]
  - Valid values: 0 (silent), 1 (warning), 2 (information), 3 (debug)

XGBoost

■ Hyperparameter tuning

□ Boost parameter (for gbtree Booster)

파라미터 명 (파이썬 래퍼)	파라미터명 (사이킷런 래퍼)	설명
eta (0.3)	**learning rate** (0.1)	- GBM의 learning rate와 같은 파라미터 - 범위: 0 ~ 1
num_boost_around (10)	**n_estimators** (100)	- 생성할 weak learner의 수
min_child_weight (1)	**min_child_weight** (1)	- GBM의 min_samples_leaf와 유사 - 관측치에 대한 가중치 합의 최소를 말하지만 GBM에서는 관측치 수에 대한 최소를 의미 - 과적합 조절 용도 - 범위: 0 ~ ∞
gamma (0)	**min_split_loss** (0)	- 리프노드의 추가분할을 결정할 최소손실 감소값 - 해당값보다 손실이 크게 감소할 때 분리 - 값이 클수록 과적합 감소효과 - 범위: 0 ~ ∞
max_depth (6)	**max_depth** (3)	- 트리 기반 알고리즘의 max_depth와 동일 - 0을 지정하면 깊이의 제한이 없음 - 너무 크면 과적합(통상 3~10정도 적용) - 범위: 0 ~ ∞
sub_sample (1)	**subsample** (1)	- GBM의 subsample과 동일 - 데이터 샘플링 비율 지정(과적합 제어) - 일반적으로 0.5~1 사이의 값을 사용 - 범위: 0 ~ 1
colsample_bytree (1)	**colsample_bytree** (1)	- GBM의 max_features와 유사 - 트리 생성에 필요한 피처의 샘플링에 사용 - 피처가 많을 때 과적합 조절에 사용 - 범위: 0 ~ 1
lambda (1)	**reg_lambda** (1)	- L2 Regularization 적용 값 - 피처 개수가 많을 때 적용을 검토 - 클수록 과적합 감소 효과
alpha (0)	**reg_alpha** (0)	- L1 Regularization 적용 값 - 피처 개수가 많을 때 적용을 검토 - 클수록 과적합 감소 효과
scale_pos_weight (1)	**scale_pos_weight** (1)	- 불균형 데이터셋의 균형을 유지

XGBoost

■ Hyperparameter tuning

□ Parameter for Learning

학습 태스크 파라미터

: 학습 수행 시의 객체함수, 평가를 위한 지표 등을 설정하는 파라미터

파라미터 명	설명
objective	- 'reg:linear' : 회귀 - binary:logistic : 이진분류 - multi:softmax : 다중분류, 클래스 반환 - multi:softprob : 다중분류, 확률반환
eval_metric	- 검증에 사용되는 함수정의 - 회귀 분석인 경우 'rmse'를, 클래스 분류 문제인 경우 'error' ---------- - rmse : Root Mean Squared Error - mae : mean absolute error - logloss : Negative log-likelihood - error : binary classification error rate - merror : multiclass classification error rate - mlogloss: Multiclass logloss - auc: Area Under Curve

Thank you

KOREA
UNIVERSITY