고려대학교
빅데이터 연구회

KU-BIG

Decision Tree & Ensemble

정석원, 박인성

목차

PART. I Decision Tree

1 Decision Tree란?

2 Split Rule

3 장단점

Decision Tree

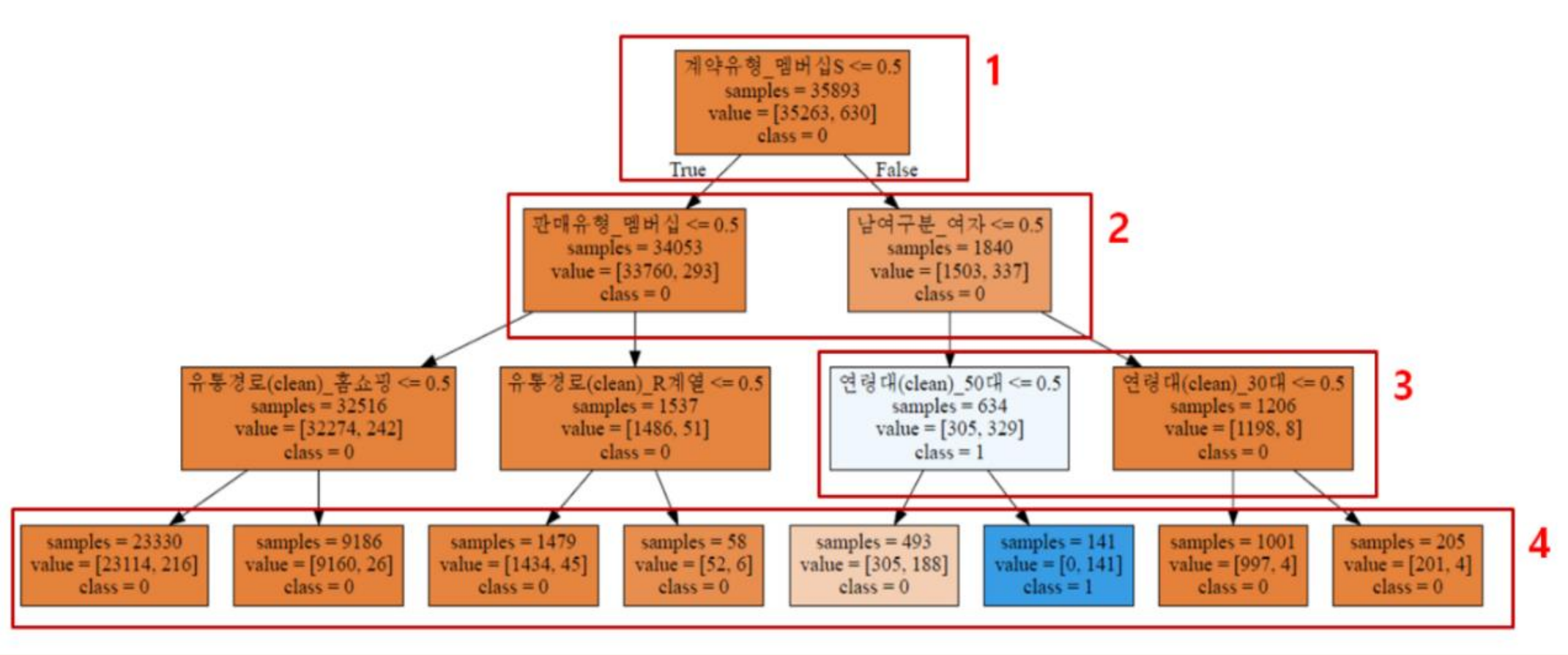
"Decision Tree"란?

의사결정 규칙을 나무구조로 나타내어 **전체 자료**를
몇 개의 **소집단으로 분류**(classification)하거나
예측(prediction)을 수행하는 분석방법

특징

- 머신러닝 알고리즘 중 하나인 Random Forest의 기본 구성요소
- 다른 머신러닝 모델들에 비해 이해하기가 비교적 쉬우며, 정확도도 높은 편이라 자주 사용되는 모델 중 하나

구조


계약유형_멤버십S <= 0.5
samples = 35893
value = [35263, 630]
class = 0
True
False
1
관매유형_멤버십 <= 0.5
samples = 34053
value = [33760, 293]
class = 0
남여구분_여자 <= 0.5
samples = 1840
value = [1503, 337]
class = 0
2
유통경로(clean)_홈쇼핑 <= 0.5
samples = 32516
value = [32274, 242]
class = 0
유통경로(clean)_R계열 <= 0.5
samples = 1537
value = [1486, 51]
class = 0
연령대(clean)_50대 <= 0.5
samples = 634
value = [305, 329]
class = 1
연령대(clean)_30대 <= 0.5
samples = 1206
value = [1198, 8]
class = 0
3
samples = 23330
value = [23114, 216]
class = 0
samples = 9186
value = [9160, 26]
class = 0
samples = 1479
value = [1434, 45]
class = 0
samples = 58
value = [52, 6]
class = 0
samples = 493
value = [305, 188]
class = 0
samples = 141
value = [0, 141]
class = 1
samples = 1001
value = [997, 4]
class = 0
samples = 205
value = [201, 4]
class = 0
4

분류방법(Split rule)

그렇다면..

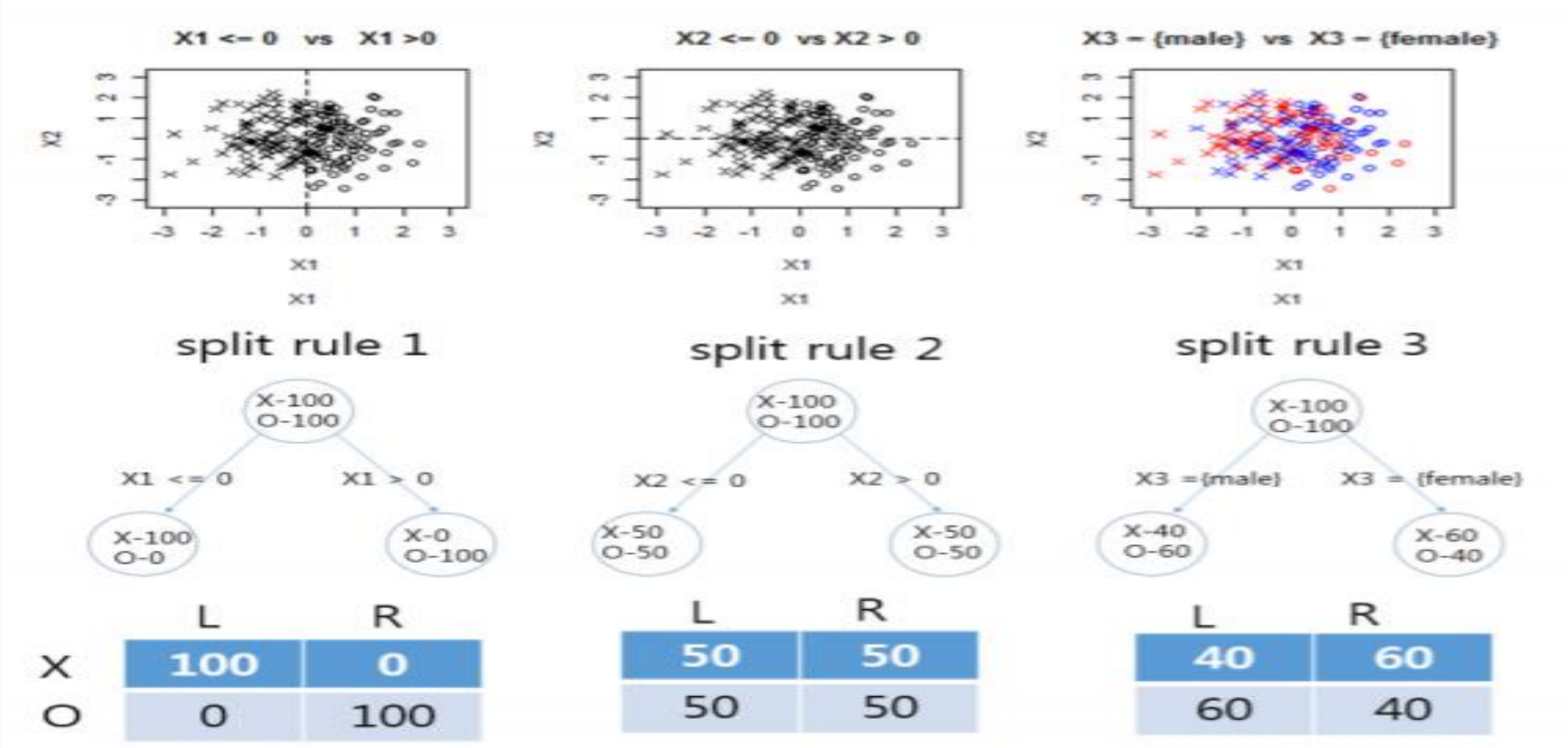
Decision Tree에서 분류는 어떻게 진행될까?

분류방법(Split rule)

순도(homogeneity) ↑ (= 불순도(impurity) ↓)

- 지표 : 지니계수(Gini index), 엔트로피(Entropy), 카이제곱 통계량(Chi-square statistic), 분산(Deviance), 오분류오차(Misclassification error)

예시


X1 <= 0 vs X1 >0
X2 <= 0 vs X2 > 0
X3 = {male} vs X3 = {female}
X1
X2
split rule 1
split rule 2
split rule 3
X-100
O-100
X1 <= 0
X1 > 0
X-100
O-0
X-0
O-100
X2 <= 0
X2 > 0
X-50
O-50
X-50
O-50
X3 ={male}
X3 = {female}
X-40
O-60
X-60
O-40
L
R
X
O
100
0
0
100
50
50
50
50
40
60
60
40

지니계수(Gini index)

i번째 노드의 지니계수 정의 :

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

$P_{i,k}$=i번째 노드에 있는 훈련 샘플 중 클래스 k에 속한 샘플의 비율

예시 - 지니계수(Gini index)

Rule 1

Root node의 지니계수

$$1 - \left[\left(\frac{100}{200}\right)^2 + \left(\frac{100}{200}\right)^2\right] = 0.5$$

왼쪽 Child node의 지니계수

$$1 - \left[\left(\frac{100}{100}\right)^2 + \left(\frac{0}{100}\right)^2\right] = 0$$

오른쪽 Child node의 지니계수

$$1 - \left[\left(\frac{0}{100}\right)^2 + \left(\frac{100}{100}\right)^2\right] = 0$$

→ $\Delta(t) = 0.5 - \left[\left(\frac{100}{200}\right) * 0 + \left(\frac{100}{200}\right) * 0\right] = 0.5$

예시 - 지니계수(Gini index)

Rule 2

Root node의 지니계수

$$1 - \left[\left(\frac{100}{200}\right)^2 + \left(\frac{100}{200}\right)^2\right] = 0.5$$

왼쪽 Child node의 지니계수

$$1 - \left[\left(\frac{50}{100}\right)^2 + \left(\frac{50}{100}\right)^2\right] = 0.5$$

오른쪽 Child node의 지니계수

$$1 - \left[\left(\frac{50}{100}\right)^2 + \left(\frac{50}{100}\right)^2\right] = 0.5$$

→ $\Delta(t) = 0.5 - \left[\left(\frac{100}{200}\right) * 0.5 + \left(\frac{100}{200}\right) * 0.5\right] = 0$

엔트로피(Entropy)

i번째 노드의 엔트로피 정의 :

$$H_i = -\sum_{k=1}^{n} p_{i,k} \log_2(p_{i,k})$$

$P_{i,k}$=i번째 노드에 있는 훈련 샘플 중 클래스 k에 속한 샘플의 비율

예시 – 엔트로피(Entropy)

Rule 1

Root node의 엔트로피

$$-\left[\left(\frac{100}{200}\right)log_2\left(\frac{100}{200}\right)+\left(\frac{100}{200}\right)log_2\left(\frac{100}{200}\right)\right]=1$$

왼쪽 Child node의 엔트로피

$$-\left[\left(\frac{100}{100}\right)log_2\left(\frac{100}{100}\right)+\left(\frac{0}{100}\right)log_2\left(\frac{0}{100}\right)\right]=0$$

오른쪽 Child node의 엔트로피

$$-\left[\left(\frac{0}{100}\right)log_2\left(\frac{0}{100}\right)+\left(\frac{100}{100}\right)log_2\left(\frac{100}{100}\right)\right]=0$$

→ $\Delta(t) = 1-\left[\left(\frac{100}{200}\right)*0+\left(\frac{100}{200}\right)*0\right]=1$

카이제곱 통계량

$$\chi^2 = \sum (\text{관측값} - \text{기댓값})^2 / \text{기댓값}$$

가지치기(Pruning)

“가지치기”란?

Decision tree에서 적합성에 큰 영향을 주는

과적합(overfitting)을 방지하고 적합성을 가진 모델

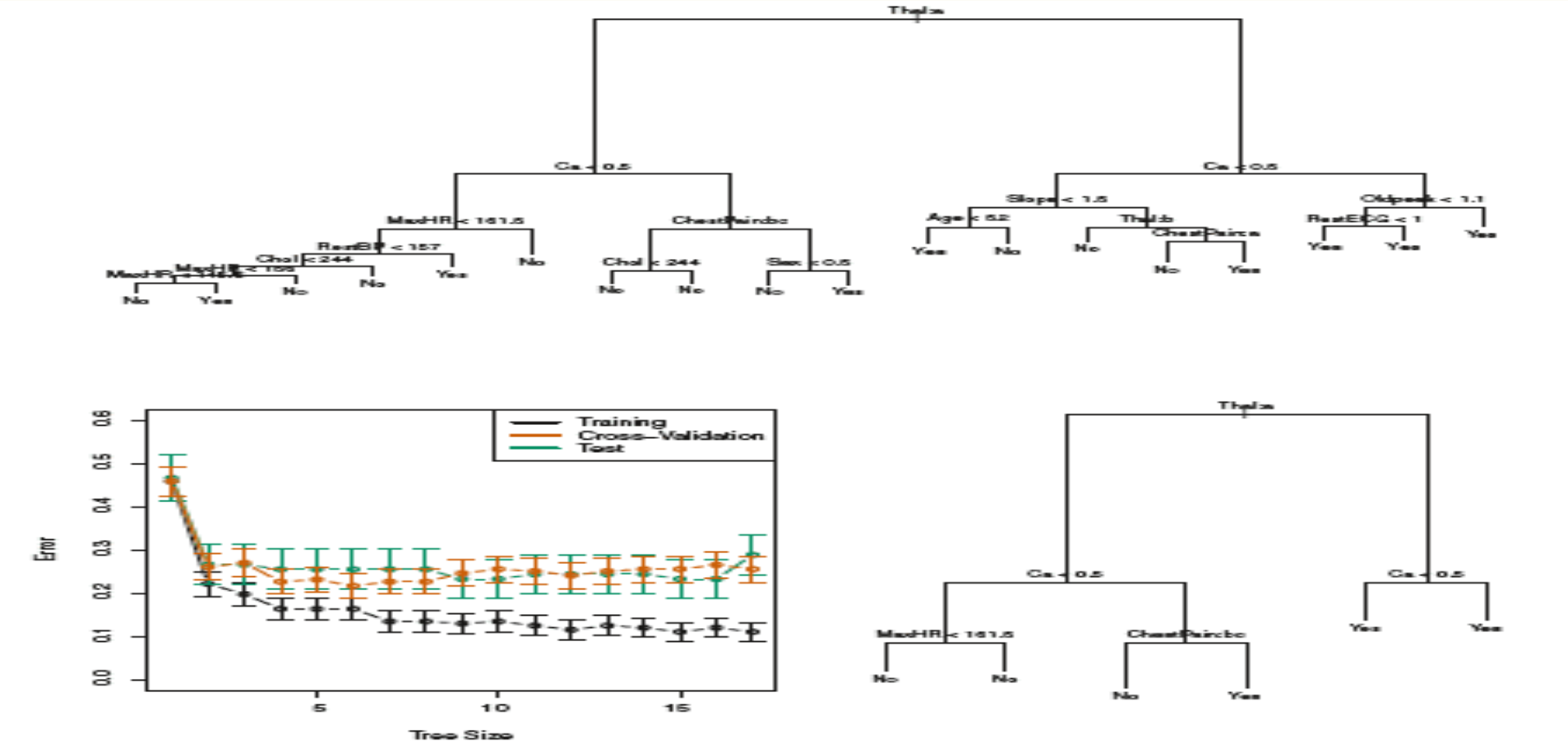
을 만들기 위한 방법

가지치기 방법

- 사전 가지치기 : Decision tree의 분류정지 조건을 사전에 설정하여, 분할을 멈추도록 하는 방식

- 사후 가지치기 : Full tree를 먼저 생성한 뒤, 모델에 대한 해석과 평가가 완화되는 방향으로 tree의 branch를 쳐내는 방식

가지치기 방법


Training
Cross-Validation
Test
Error
Tree Size
5
10
15
0.0
0.1
0.2
0.3
0.4
0.5
0.6
No
Yes

장점과 단점

장점

- 구조가 단순하여 결과해석이 쉬움
- 선형성, 정규성, 등분산성 가정이 불필요한 비모수적 모형

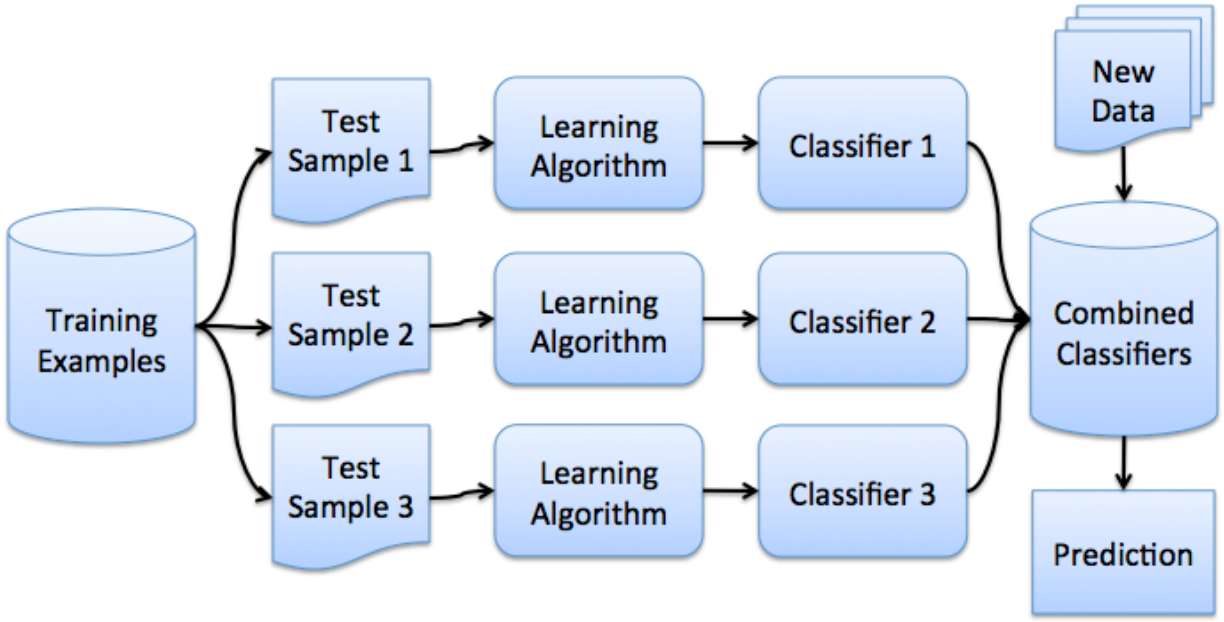
단점

- 기준값의 경계선 근방의 자료 값에 대해서는 오차가 클 수 있음
- 새로운 자료에 대한 예측이 불안정
- 선형성이 미흡해 모델의 안정성이 낮음

PART. Ⅱ Ensemble

1. 앙상블의 개념
2. 앙상블의 장단점
3. 앙상블의 종류

1. 앙상블의 개념



여러 개의 예측 모형을 만든 후 그 모형들을 종합하여 하나의 최종 예측 모형을 만드는 방법

1. 앙상블의 개념

앙상블의 원리

Ex) 오분류율이 0.05인 분류기 5개 있을 때

1. 5개 분류기 오분류율의 평균

$$E_{Avg} = 0.05$$

2. 앙상블 모형의 오분류율 (절반이상이 오분류할 때)

$$E_{ensemble} = \sum_{i=3}^{5} (0.05)^i (1 - 0.05)^{5-i} = 0.0001$$

1. 앙상블의 개념

앙상블 최종 모델 결정 방법

1. 평균

Model1	Model2	Model3	Average Prediction
45	55	60	50

2. 다수결

Model1	Model2	Model3	Voting Prediction
1	0	0	0

3. 가중 평균

	Model1	Model2	Model3	Weight Average Prediction
Weight	0.4	0.3	0.3	
Prediction	45	55	60	52.5

2. 앙상블의 장단점

장점	단점
- 이상치에 대한 대응력이 높아진다. - Variance를 감소시키거나 (Bagging) Bias를 감소시키거나 (Boosting) 예측력을 높인다. (Stacking) - Data가 너무 작거나 클 때 특히 유용하다. - Overfitting의 가능성을 줄여준다.	- 모형의 투명성이 떨어지게 되어 현상에 대한 원인을 분석할 때 적합하지 않다. - 앙상블을 만들기 위해 모델을 선택하는 것이 매우 어렵다. - 시간이 오래 걸리고 해석이 어렵다.

3. 앙상블의 종류

1) 학습 데이터의 다양화 ex) Bagging, Boosting

2) 데이터와 변수의 다양화 ex) Random Forest

3) 분류기의 다양화 ex) Stacking

PART. Ⅲ Boosting

1 Boosting이란?

2 작동 단계

3 특징

Boosting

"Boosting"이란?

간단한 약분류기(weak classifier)들이 상호보완 하도록 단계적으로 학습하고, 이를 조합하여 만들어진 최종 강분류기(strong classifier)의 성능을 증폭시키는 원리

Boosting 종류

- Ada Boost
- Gradient Boost
- XG Boost

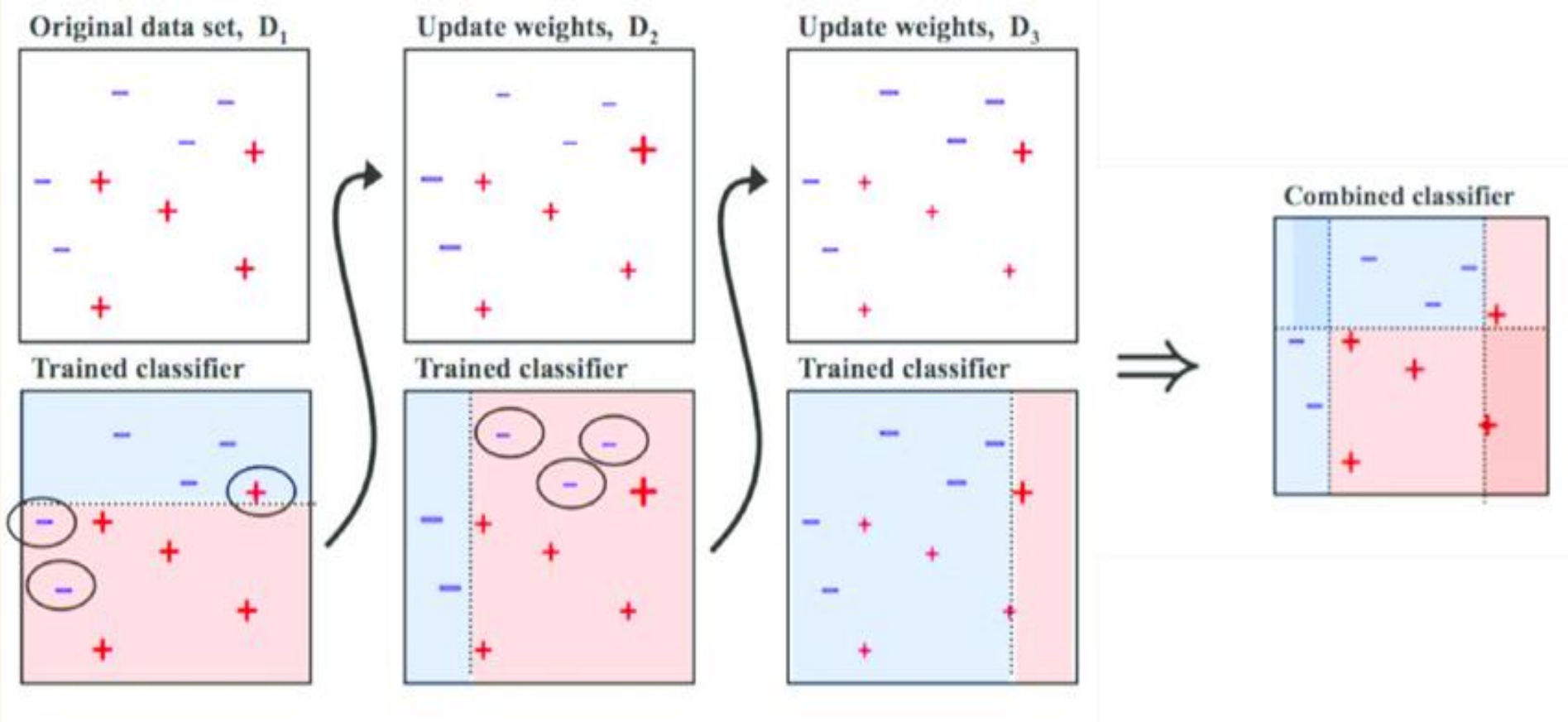
Ada Boost

"Ada Boost"란?

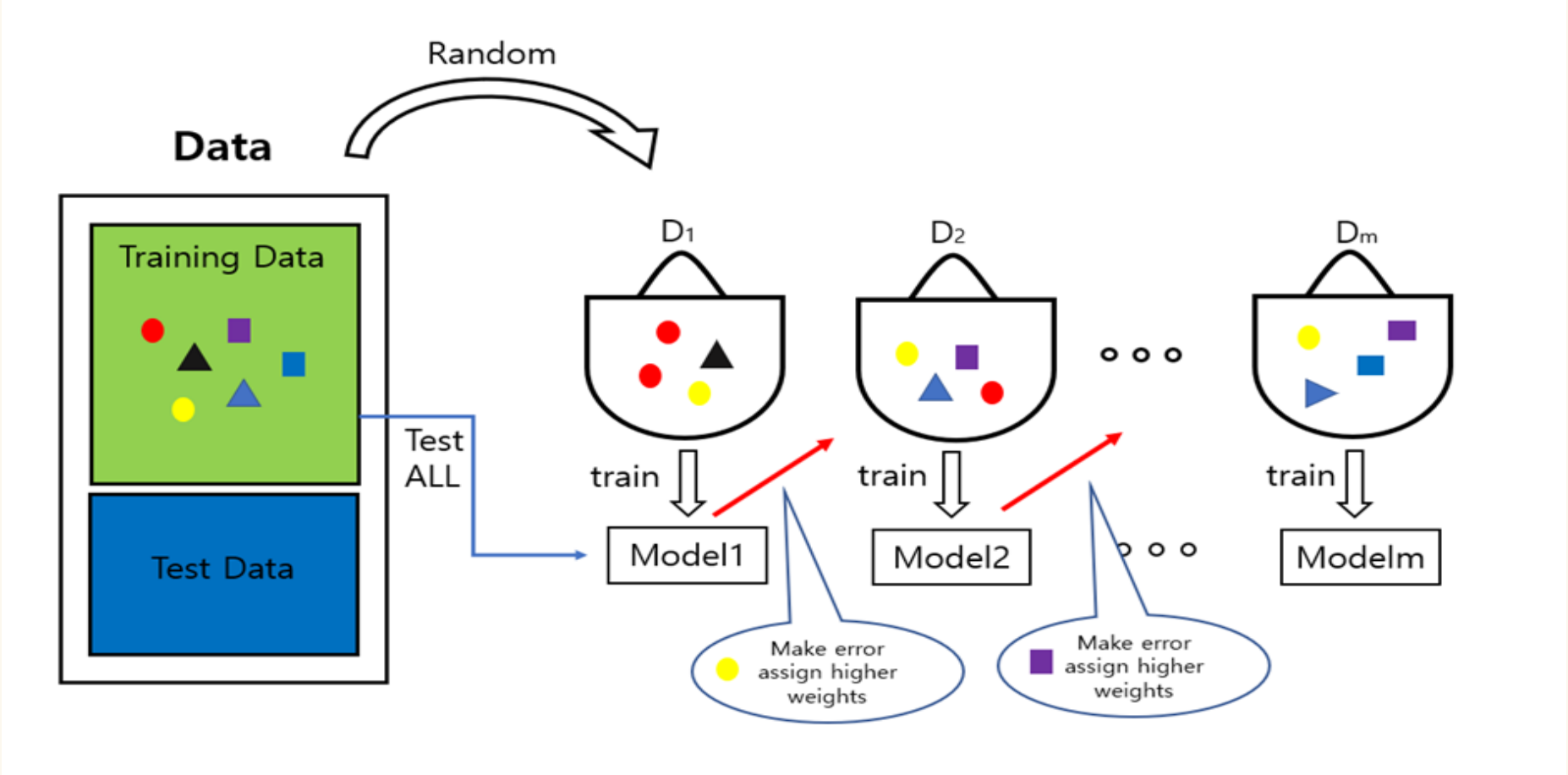
Adaptive Boosting의 약자.

이전의 분류기가 잘 분류하지 못한 것들을 이어지는 약한 학습기들이 수정해줄 수 있다는 점에서 다양한 상황에 적용할 수 있다(adaptive).

Ada Boost


Original data set, D_1
Update weights, D_2
Update weights, D_3
Trained classifier
Trained classifier
Trained classifier
Combined classifier

Ada Boost 작동단계



Ada Boost 작동단계

현재 주어진 데이터셋에 대해 상대적으로 단순한 모델을 이용하여 학습

학습오류가 큰 개체의 선택확률을 증가, 오류가 작은 개체의 선택확률을 감소

→ 반복적으로 오류가 큰 데이터에 집중

일반적으로 분류모델의 경우는 Weighted Majority Vote,

회귀모델의 경우에는 Weighted Sum을 사용하여 최종 모델을 생성

Ada Boost 특징

- Ada Boosting은 Bias를 줄이려는 것에 초점을 맞춘 기법

  → Outlier에 취약하다

(WHY?? 오류가 크게 발생하는 Outlier에 반복적으로 집중하기때문)

- 과적합(Overfitting)의 문제가 발생 할 수 있다.

  → 모델의 모수를 적절하게 조정하는 것이 중요!!

Gradient Boost

"Gradient Boost"란?

경사하강법을 사용해 손실함수를 최소화 하는 가중치를 적용하여 Ada Boosting보다 성능을 개선한 Boosting기법

Gradient Boost 특징

- 특성의 스케일을 조정하지 않아도 되고(outlier에 덜 민감하게 반응), 연속적인 데이터 특성에서도 잘 작동
- Ada Boosting에서 보다 학습훈련의 시간이 길게 걸리므로 경사 하강법 적용 시 적절한 변수 조정이 필요함
- 고차원 데이터에서는 잘 작동하지 않을 수 있음
- Regression과 Classification 문제에 모두 적용 가능

XG Boost

"XG Boost"란?

Extreme Gradient Boosting의 약자

기존의 Gradient Boosting의 속도문제를 해결하기 위해 전산속도와 모델의 성능에 초점을 맞춘 기법

XG Boost 특징

- 기존 Gradient Boosing 비해 연산속도가 빠르고 모델의 성능이 향상된다
- 과적합(Overfitting)이 잘 일어나지 않는다
- 훈련하는 동안 컴퓨터의 사용 가능한 모든 CPU를 사용함으로써 트리의 구조를 병렬화 한다
- 다른 알고리즘과 연계활용성이 좋다
- 최근의 앙상블 모델 중에서 가장 우수한 알고리즘으로 평가받아 각종 대회에서 사용된다

PART. IV Bagging

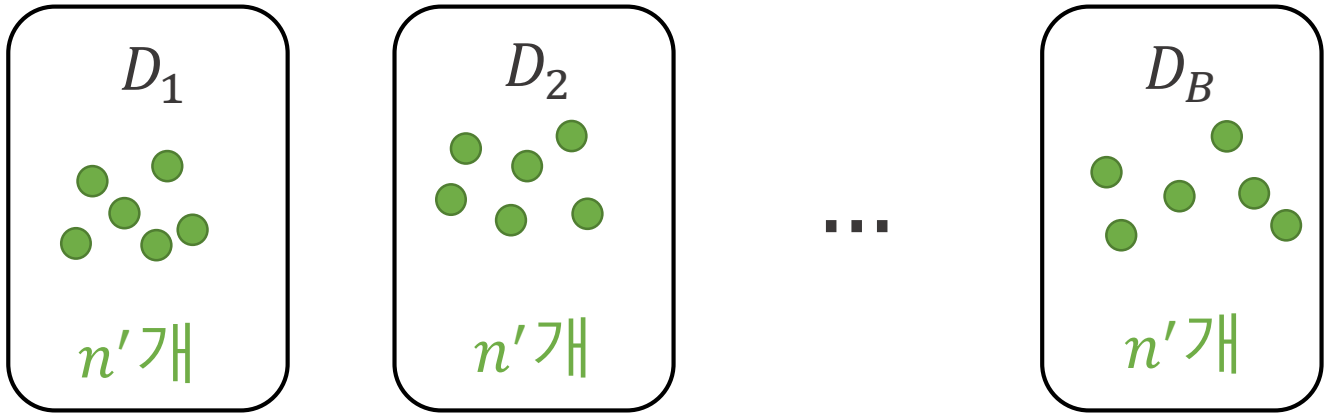
1. Bagging의 개념
2. Bagging의 알고리즘
3. Bagging의 장단점

1. Bagging의 개념

: Bootstrap Aggregating 의 줄임말

원 데이터 집합으로부터 크기가 같은 표본을 여러 번 단순임의 복원추출하여 각 표본(Bootstrap Sample)에 대해 분류기(classifiers)를 생성한 후 그 결과를 앙상블하는 방법

1. Bagging의 개념



n : 전체 관측치 개수

n′ : bag 안의 관측치 개수

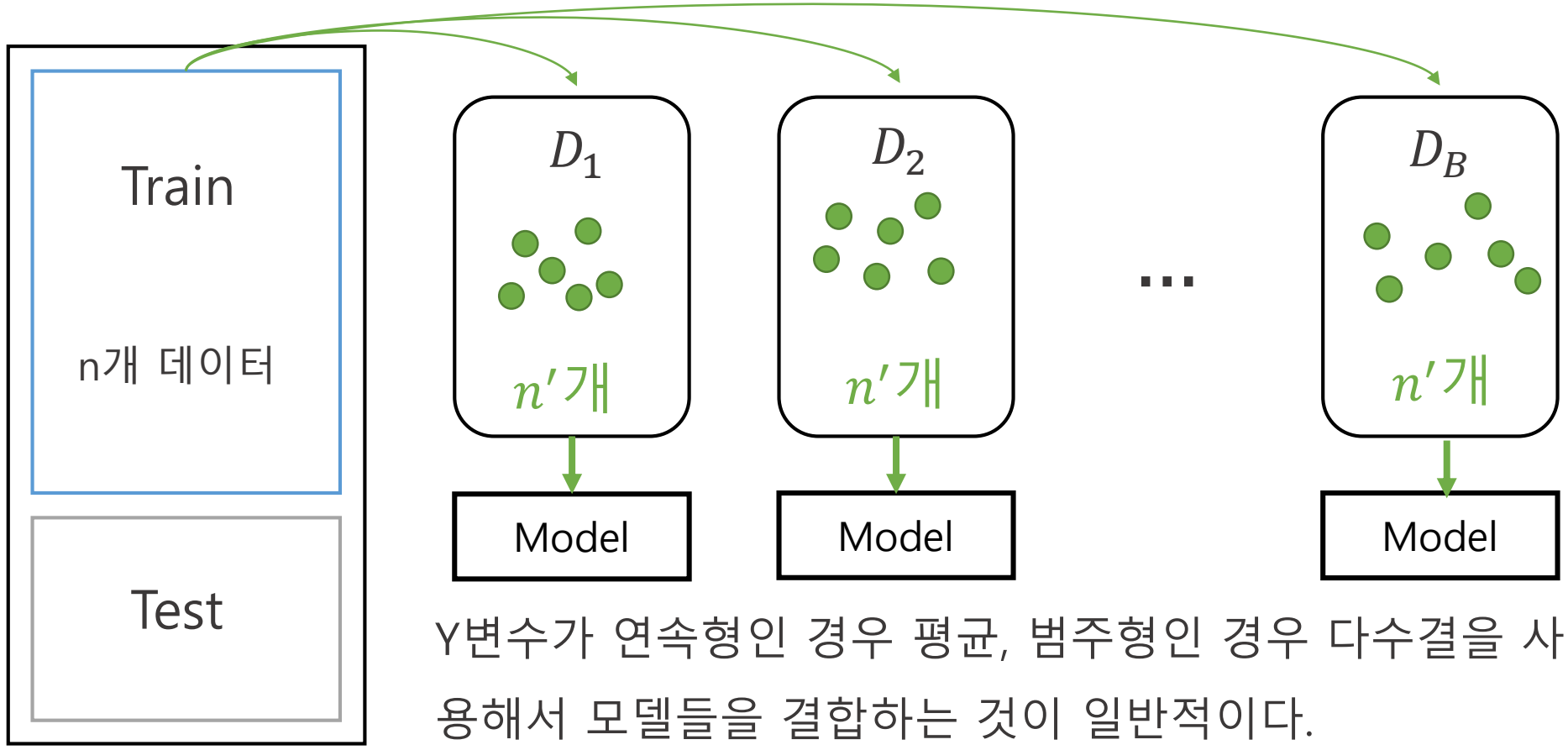
B : bag의 개수

일반적으로 n′=n

단순임의복원추출시 중복값 제외하면 약 60%

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n} = e^{-1} = 0.3678$$

1. Bagging의 개념



Y변수가 연속형인 경우 평균, 범주형인 경우 다수결을 사용해서 모델들을 결합하는 것이 일반적이다.

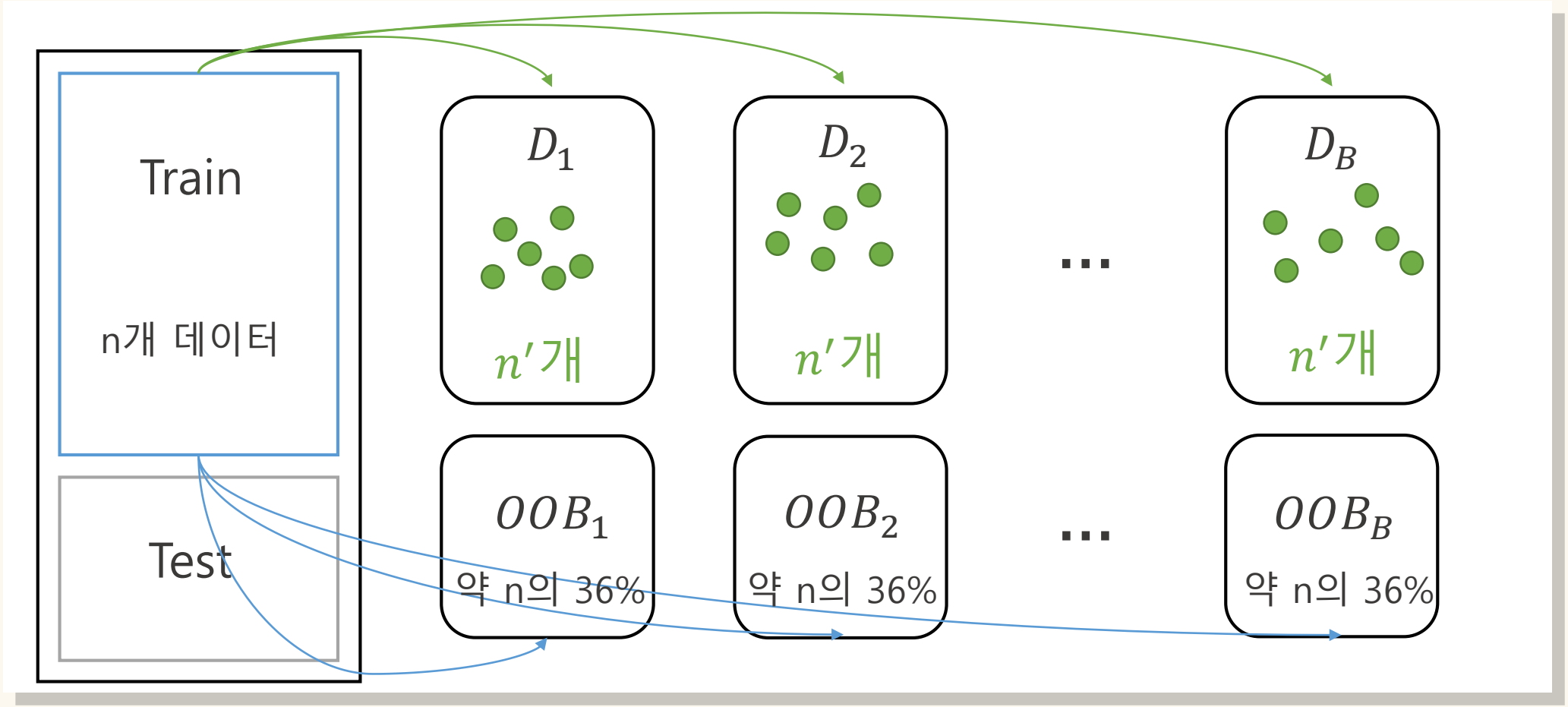
Out-of-Bag(OOB) error

: Bootstrap Sampling 과정에서 표본으로 뽑히지 않은 데이터

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n} = e^{-1} = 0.3678$$

따로 test data를 구성할 필요 없이 OOB 데이터를 이용해서 test error를 구하거나 변수의 중요도를 측정할 수 있다.

Out-of-Bag(OOB) error



2. Bagging의 알고리즘

1) n개의 training data에서 크기 n의 bootstrap sample을 B개($D_1, D_2, \ldots, D_B$) 생성한다.

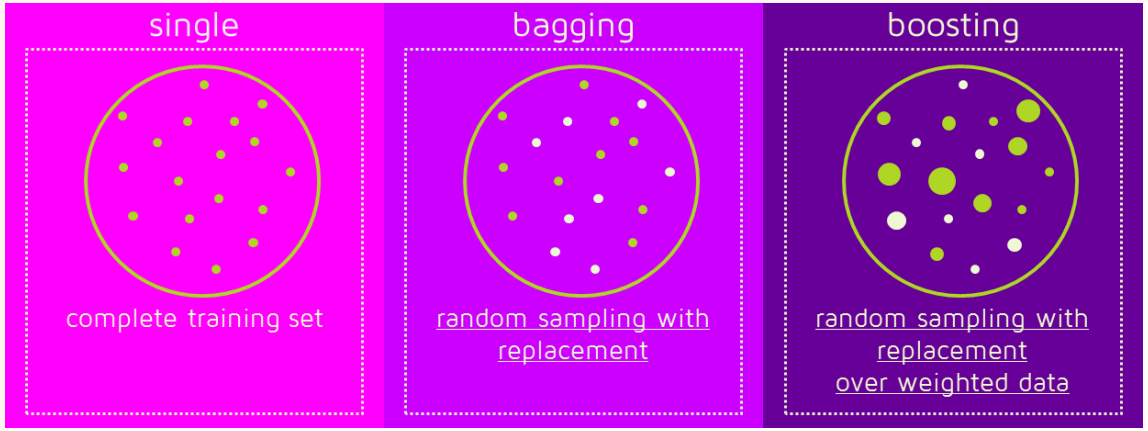
2) 각각의 bootstrap sample들로 B개의 분류기를 학습시킨다.

3) 모델들을 결합해 최종 모델 산출

2. Bagging의 알고리즘

- 주로 decision tree에 대해 많이 쓰이지만 어떤 모델에 대해서도 사용 가능하다.
- 가지치기(pruning)를 하지 않고 최대로 성장한 tree들을 활용한다.
- 최대로 성장한 각 tree들의 variance는 크고 bias는 작다.
- Bagging을 통해 최종 모델의 variance는 줄어들게 된다.
- $Z_1, Z_2, \ldots, Z_n$의 n개의 독립적인 관측치가 있을 때 각각의 분산이 σ^2 이라 하면 $\bar{Z}$의 분산은 σ^2/n 이 되는것과 비슷한 원리

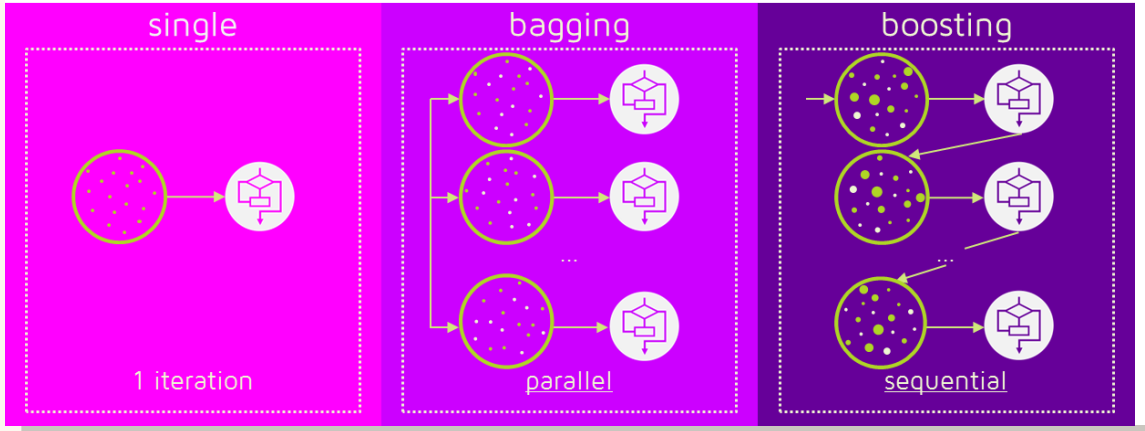
2. Bagging vs Boosting



학습데이터 생성

Bagging : 단순임의복원추출

Boosting : 가중치 부여된 데이터셋에서 단순임의 복원 추출



모델과 학습데이터와의 관계

Bagging : 병렬적 학습 (분류기 상호 연관 x)

Boosting : 연속적 학습 (이전 분류기 학습 다음에 영향)

2. Bagging vs Boosting

적용 목적

Bagging : Variance를 줄여 Overfitting 방지하는 것이 목적

-> 단일 모델의 Variance는 크고, Bias는 작을 때 적절하다.

Boosting : Bias를 줄여 Underfitting 방지하는 것이 목적

-> 단일 모델의 Variance는 작고, Bias는 클 때 적절하다.

3. Bagging의 장단점

장점

- 변동성이 큰 모델의 Variance를 줄일 수 있다.
- 과적합을 방지할 수 있다.
- 데이터 양이 적어도 모델 생성이 가능하다.

단점

- Tree에서 활용되는 독립변수 선정에 대한 고려가 없기 때문에 tree간의 높은 상관성이 생기게 된다.

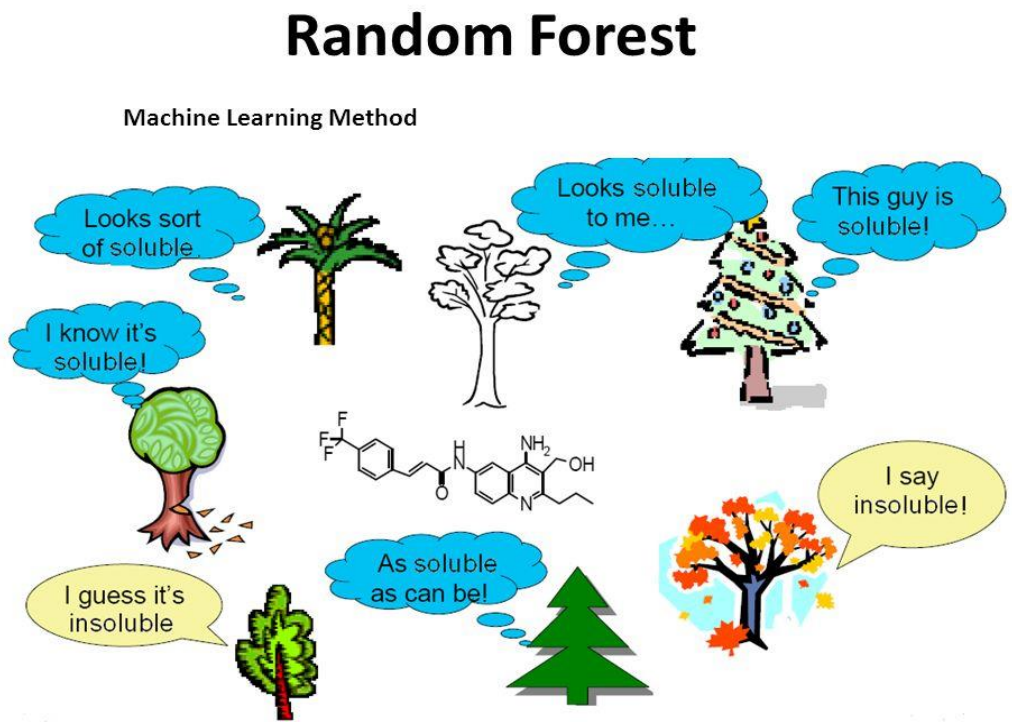
=> 이 문제점을 해결하는 방법이 Random Forest

PART. V Random Forest

1. Random Forest 의 개념
2. Random Forest 의 알고리즘
3. 결과 분석 방법

1. Random Forest의 개념

: 다수의 Decision tree를 결합하여 하나의 모형을 생성하는 방법



Bagging

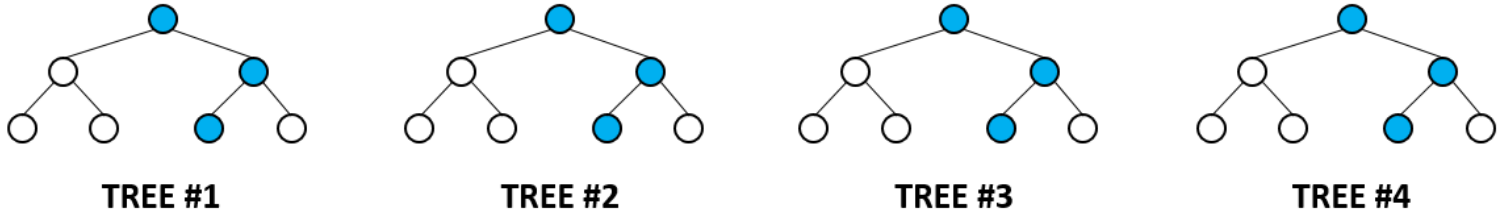
+

Random Subspace method

(변수 선택)

Random Subspace Method

: 기계학습에서 변수를 전부 이용하는 것 대신 몇 가지를 임의 추출하여 이용하는 것을 의미한다.

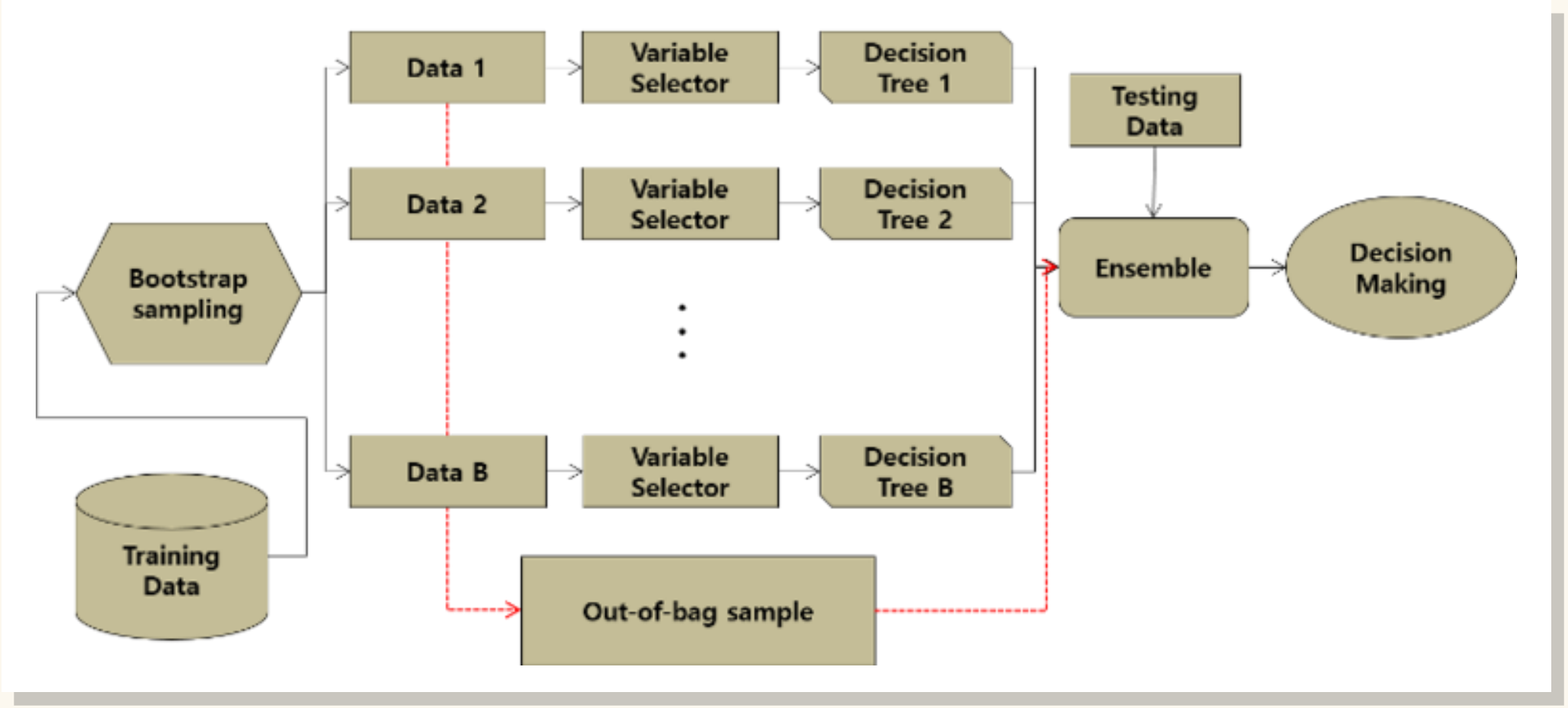


- 극소수의 변수들이 강한 영향력을 가진다면 여러 tree들에 그 변수들이 중복되어 선택되고 tree들이 상관화된다.
- 이를 방지하기 위해 변수도 임의 추출한다.

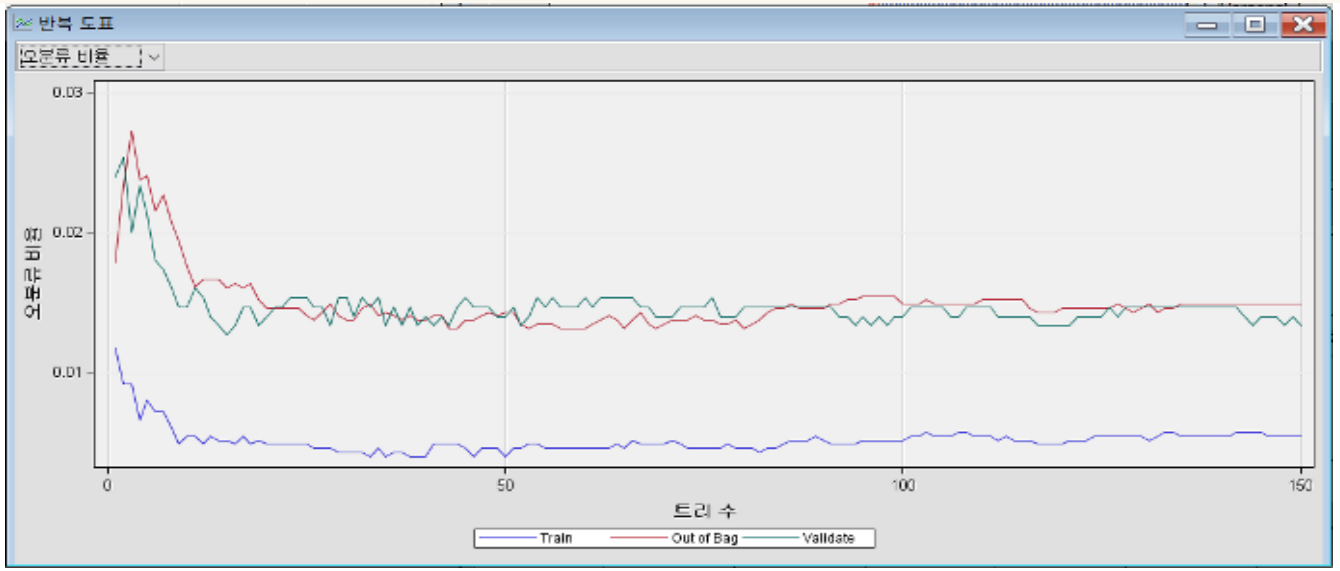
2. Random Forest의 알고리즘

1) n개의 training data에서 크기 n의 bootstrap sample을 B개 $(D_1, D_2, \ldots, D_B)$ 생성한다.
2) 각 sample에서 전체 p개 변수 중 p/3개(회귀 tree)나 $\sqrt{p}$개(분류 tree)의 변수를 임의 추출한다.
3) 각각의 bootstrap sample들로 B개의 분류기(decision tree)를 학습시킨다.
4) 모델들을 결합해 최종 모델 산출

2. Random Forest의 알고리즘



3. 결과 분석 방법 – 적정 Tree 개수



- 모형을 학습 데이터로 적합했으므로 Training set의 오분류율은 비교적 낮다. 반대로 OOB 오분류율은 높게 나타난다. 실제 평가용 데이터에 대한 오분류율은 그 둘 사이에 존재하게 된다.
- 오분류율이 안정화되는 지점(대략 50개)의 tree 개수를 선택하면 된다.

3. 결과 분석 방법 – 변수 중요도

- Bagging이나 Random Forest는 다수의 tree를 결합하여 만들었으므로 어떤 변수가 예측과정에서 얼마나 중요한지 정확히 알 순 없지만 전체적인 중요도를 다음 방법을 통해 알 수 있다.

① RSS(Residual Sum of Squares) - 회귀 tree

② Gini 계수 - 분류 tree

③ Permutation Importance

3. 변수 중요도 – RSS

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

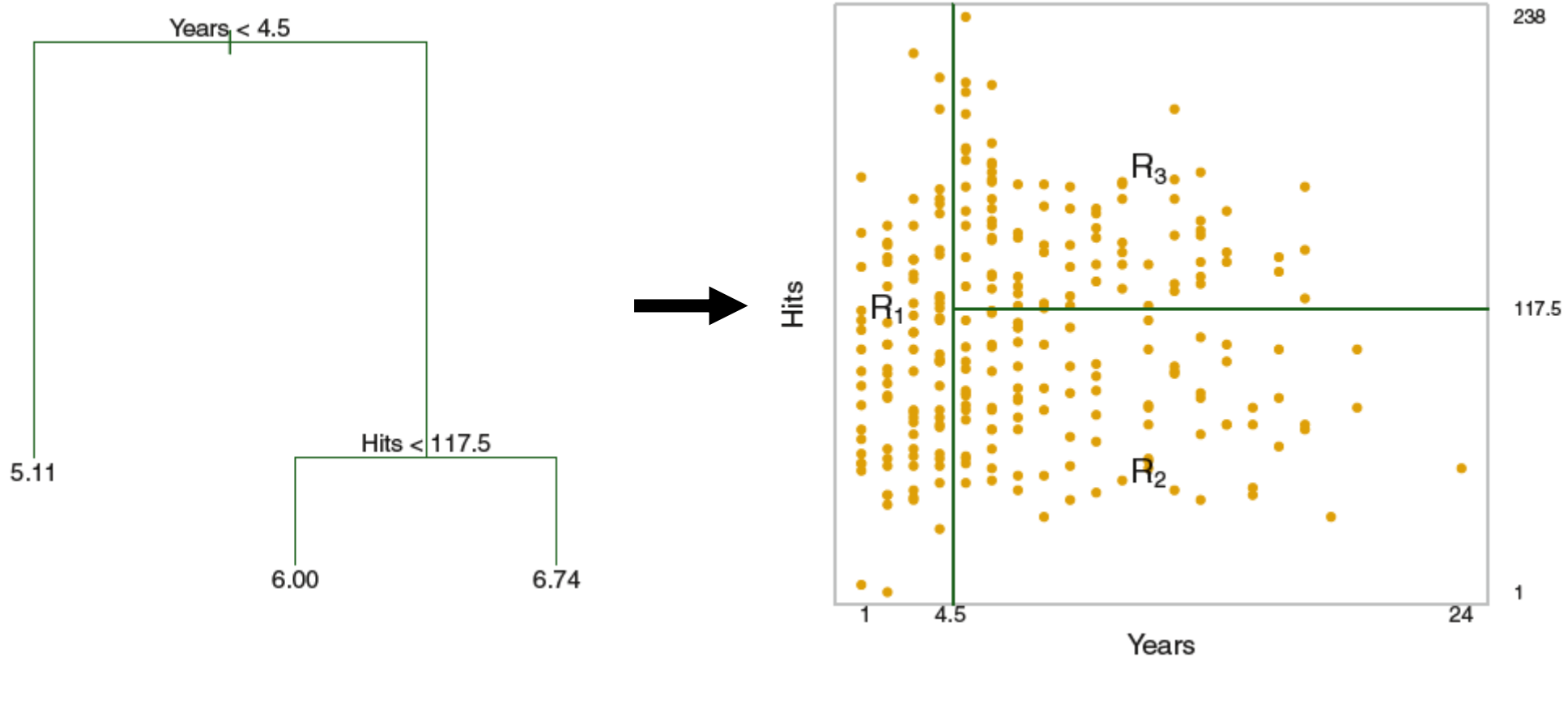
R_j : x 변수에 대해 겹치지 않는 j번째 영역

$\hat{y}_{R_j}$: R_j의 평균 y값

RSS : 실제 y값들과 $\hat{y}_{R_j}$ 값들의 차이의 제곱합

1) x 변수에 대해 J개의 겹치지 않는 영역을 만든다.
2) B개의 tree에 대해 각 변수에서의 split으로 인해 RSS가 감소된 정도를 측정하고 평균을 낸다.
3) 해당 변수의 RSS 감소량이 클수록 중요한 변수임을 의미한다.

3. 변수 중요도 – RSS


Years < 4.5
5.11
Hits < 117.5
6.00
6.74
Hits
238
117.5
1
1
4.5
24
Years
R_1
R_2
R_3

3. 변수 중요도 – Gini 계수

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

K : y 범주의 개수

$\hat{p}_{mk}$:m영역에 속하는 레코드 중 k범주에 속하는 레코드의 비율

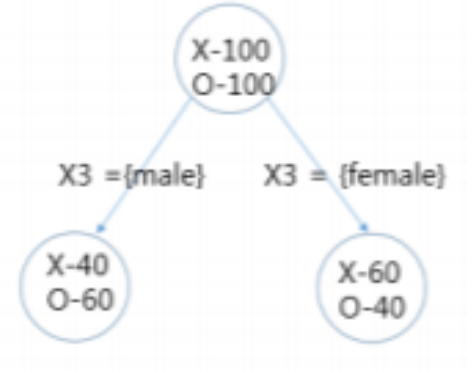
1) Split 되기 전의 불순도(Gini 계수)를 구한다.
2) 왼쪽으로 Split된 node의 Gini 계수, 오른쪽으로 Split된 node의 Gini 계수를 구하고 각 방향으로 갈 확률을 곱해준 뒤 더한다. (Split 후의 Gini 계수)
3) 각 변수마다 Split 전후의 Gini 계수를 계산한다. Gini 계수의 감소량이 클수록 중요한 변수임을 의미한다..

3. 변수 중요도 – Gini 계수 ?

Split 후의 Gini 계수 : $\mathrm{p_L} * i(t_L) + p_R * i(t_R)$

$\mathrm{p_L}$: 왼쪽 node로 분류될 확률 $i(t_\mathrm{L})$: 왼쪽 node의 Gini 계수

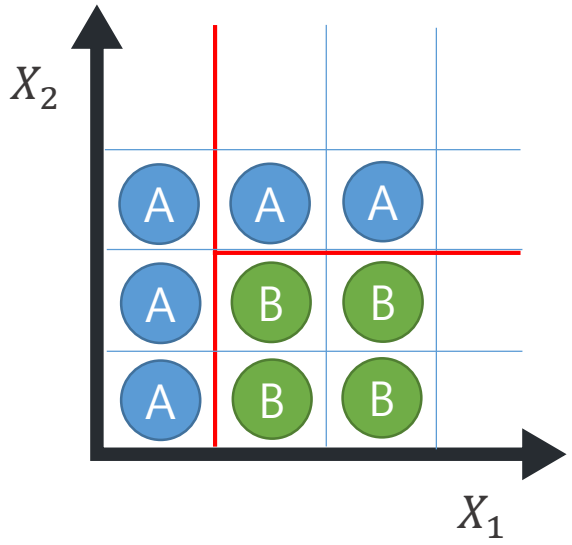
p_R : 오른쪽 node로 분류될 확률 $i(t_R)$: 오른쪽 node의 Gini 계수



Ex)

$$\frac{1}{2}\left(\frac{4}{10} * \frac{6}{10} + \frac{6}{10} * \frac{4}{10}\right) + \frac{1}{2}\left(\frac{4}{10} * \frac{6}{10} + \frac{6}{10} * \frac{4}{10}\right) = 0.48$$

3. 변수 중요도 – Gini 계수

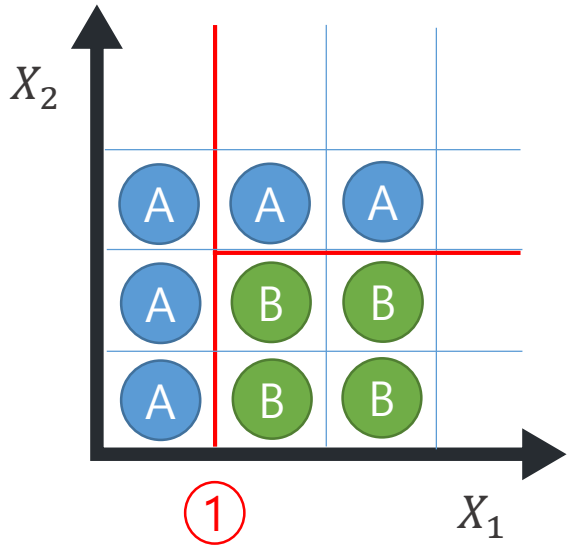


만약 split rule1이 $X_1 > 1$, split rule2가 $X_2 > 2$ 일 때

Root node의 Gini 계수 :

$$\frac{5}{9} * \frac{4}{9} + \frac{4}{9} * \frac{5}{9} = \frac{40}{81}$$

3. 변수 중요도 – Gini 계수

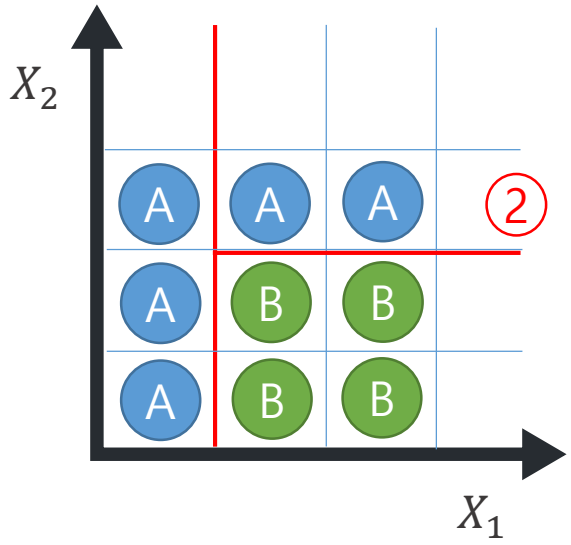


split rule1 이후의 Gini 계수 :

$$\frac{3}{9}\left(\frac{3}{3}*\frac{0}{3}+\frac{0}{3}*\frac{3}{3}\right)+\frac{6}{9}\left(\frac{2}{6}*\frac{4}{6}+\frac{4}{6}*\frac{2}{6}\right)=\frac{8}{27}$$

$\frac{40}{81}-\frac{8}{27}=\frac{16}{81}$ 만큼 Gini 계수 감소

3. 변수 중요도 – Gini 계수



split rule2 이후의 Gini 계수 :

$$\frac{2}{6}\left(\frac{0}{2}*\frac{2}{2}+\frac{2}{2}*\frac{0}{2}\right)+\frac{4}{6}\left(\frac{0}{4}*\frac{4}{4}+\frac{4}{4}*\frac{0}{4}\right)=0$$

$\frac{8}{27}-0=\frac{8}{27}$ 만큼 Gini 계수 감소

X_2의 Gini 계수 감소량이 더 크므로 이 tree에선 X_2가 더 중요한 변수이다.

3. 변수 중요도 – Gini 계수

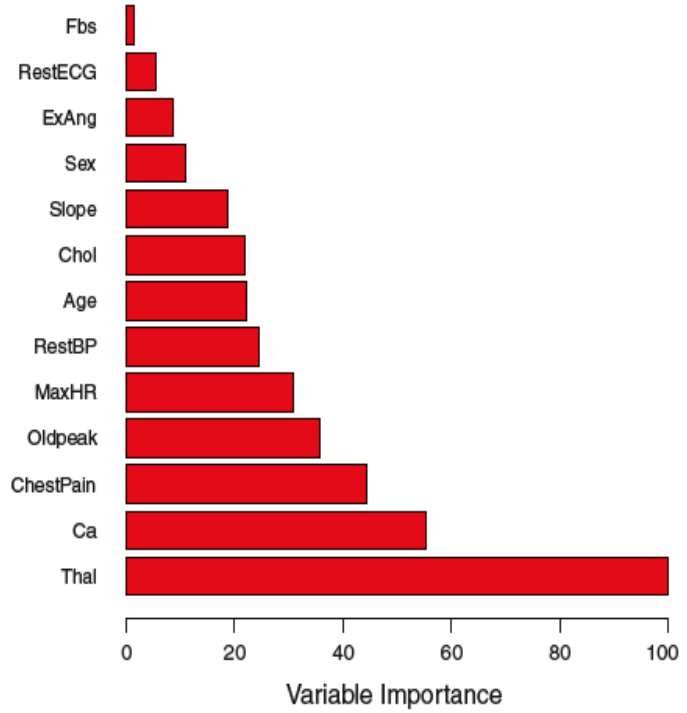
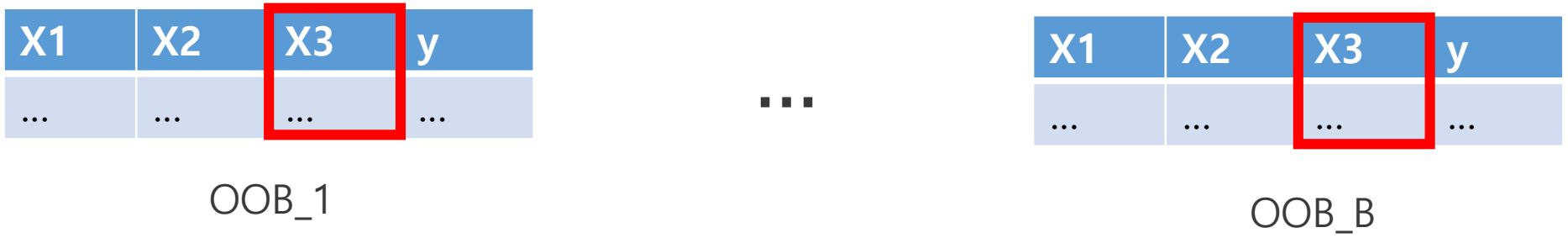


FIGURE 8.9. *A variable importance plot for the* `Heart` *data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.*

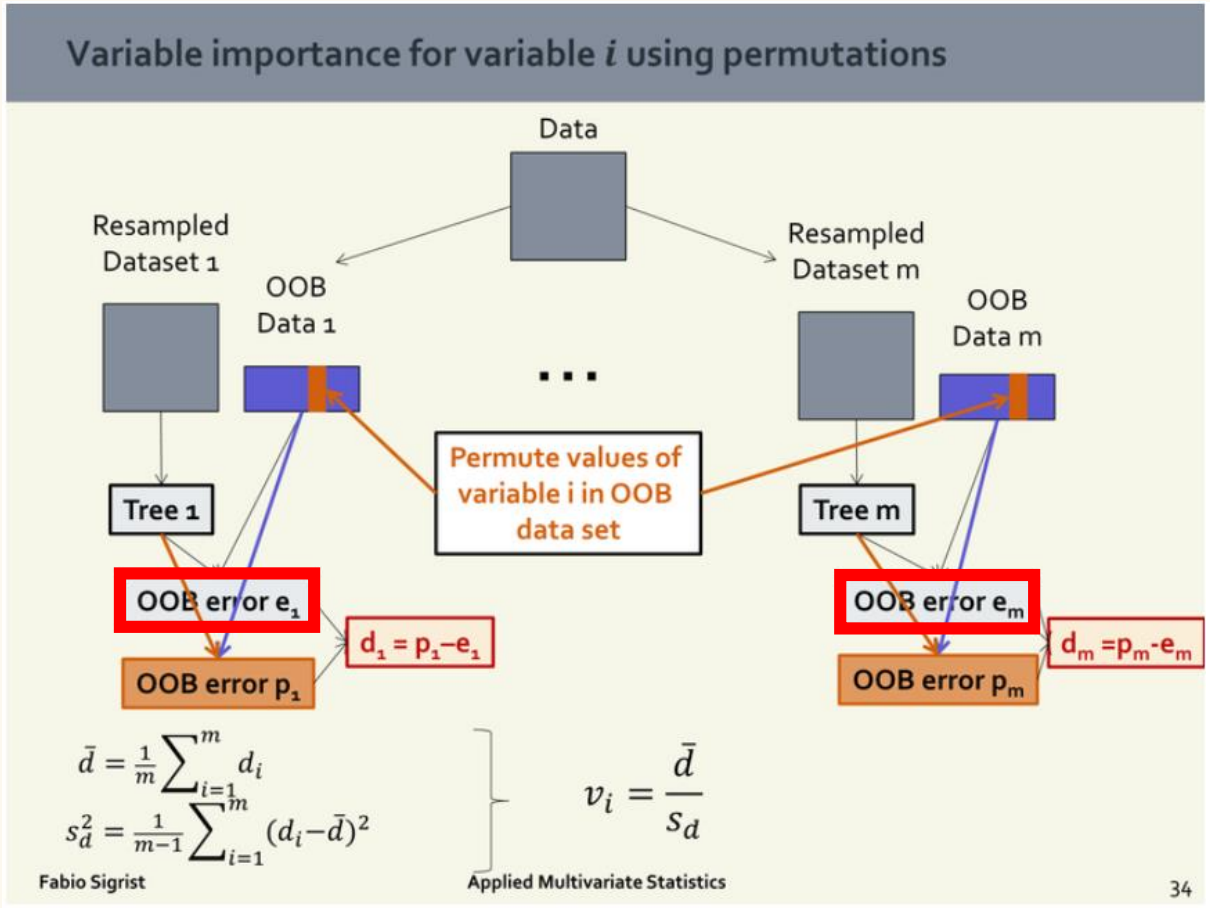
변수별 평균적인 Gini 계수 감소량을 min-max scaling 하여 상대적인 중요도를 알 수 있다.

3. 변수 중요도 – Permutation Importance



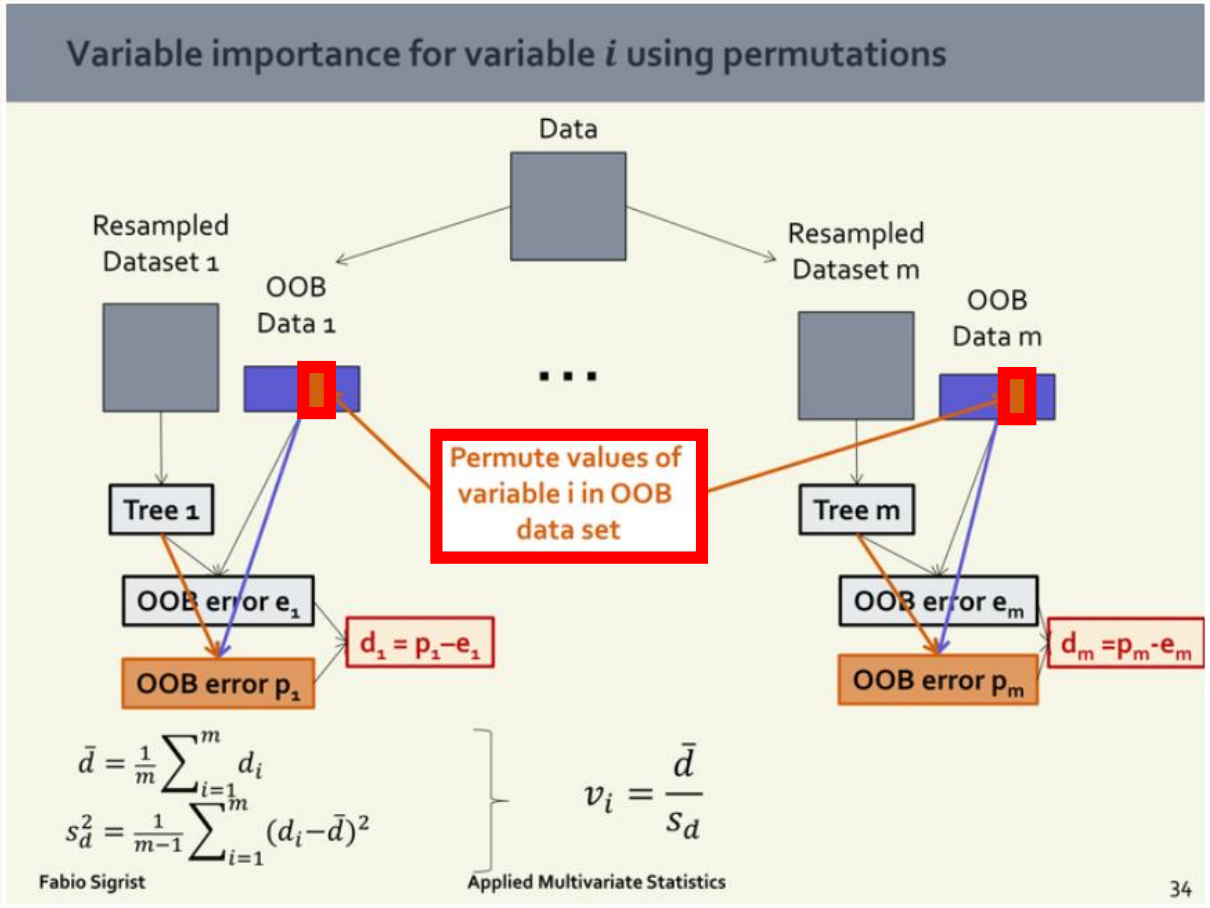
1) B개 tree에 대해 OOB error를 구한다.
2) OOB 데이터의 특정 변수를 선택한 후 그 변수의 값들을 재조합(permutation)한다. (값들의 위치를 서로 무작위로 바꾼다.)
3) 그 후에 OOB error를 다시 구하고 원래 OOB error와의 차이를 구한다. 차이가 클수록 중요한 변수이다.

3. 변수 중요도 – Permutation Importance



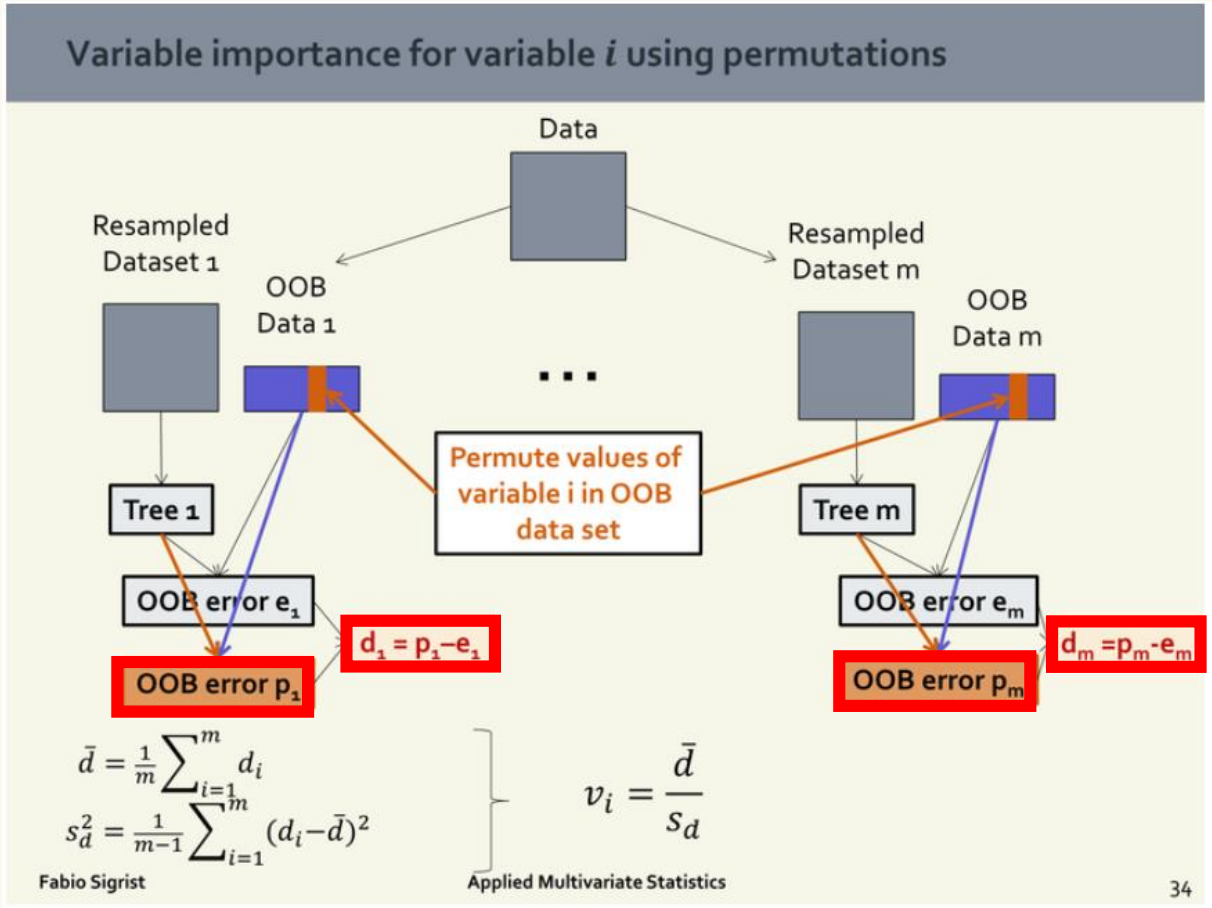
m개 tree에 대해 OOB error를 구한다.

3. 변수 중요도 – Permutation Importance



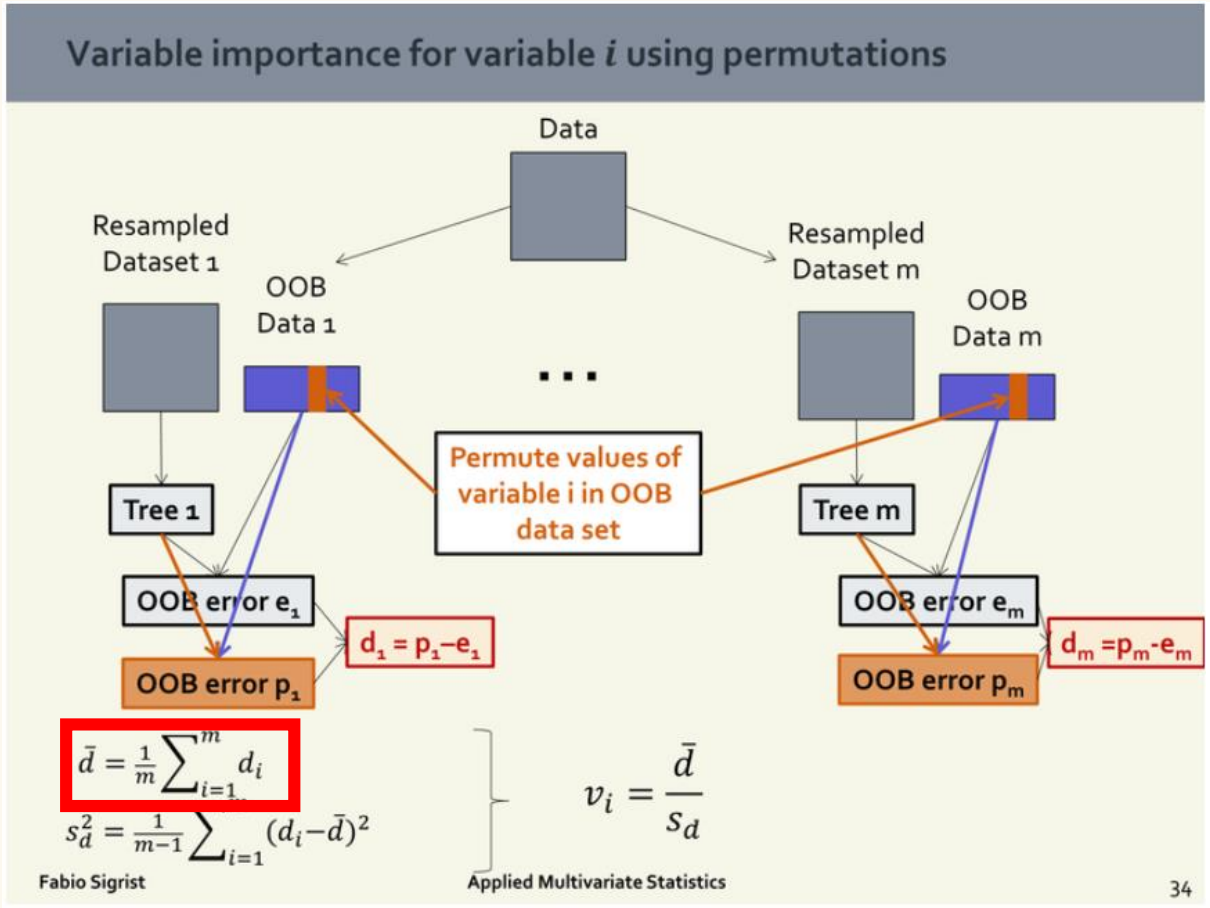
OOB 데이터의 특정 변수를 선택한 후 그 변수의 값들을 재조합(permutation)한다.

3. 변수 중요도 – Permutation Importance



그 후에 OOB error를 다시 구하고 원래 OOB error와의 차이를 구한다.

3. 변수 중요도 – Permutation Importance



$d_1, \ldots, d_m$ 의 평균 $\bar{d}$ 가 한 변수의 중요도가 된다.

3. Stacking(= Meta Ensemble)

서로 다른 모델들을 조합해서 최고의 성능을 내는 모델을 생성한다.

-> 모델 간의 장점은 취하고 단점은 상호 보완

SVM, Random Forest, KNN 등 다양한 알고리즘 사용 가능하다.

하나의 예측 모델 계층 위에 또 하나의 모델이 설정된다.

-> 하위모델은 training set으로만 분석

-> 상위계층의 모델은 하위 단계를 거쳐서 도출된 결과들을 토대로 결론을 도출한다.


Do you
have any
question?
Thank you
for your attention.