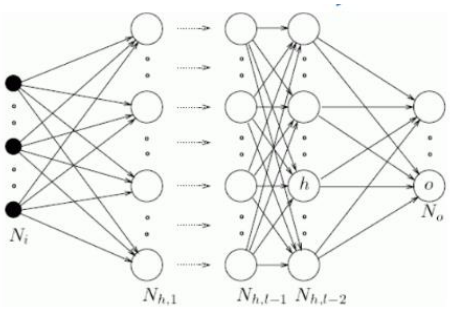
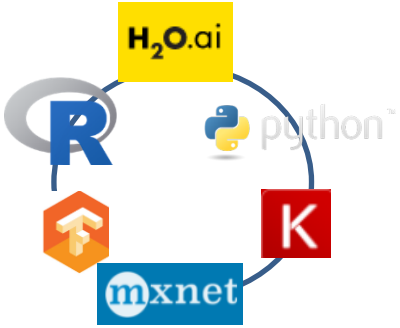

Machine Learning
2023

# 1. Ensemble 개요

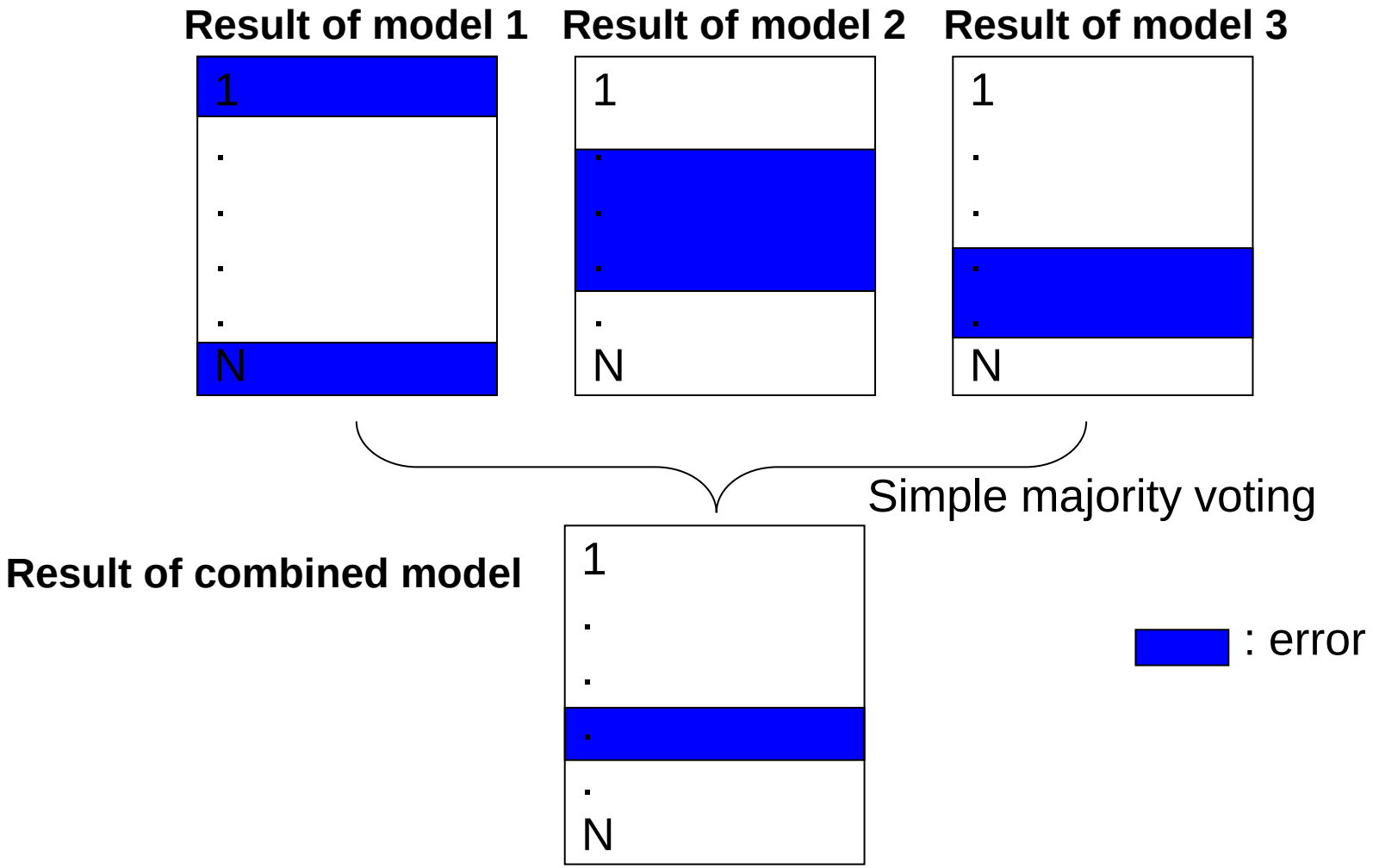
**AI와 머신러닝**

- **Machine = 컴퓨터**
- **Learning = 분석**
- **"컴퓨터가 데이터를 분석하는 알고리즘과 기술"**
- **1) Study of algorithms 2) improve their performance 3) at some task 4) with experience**



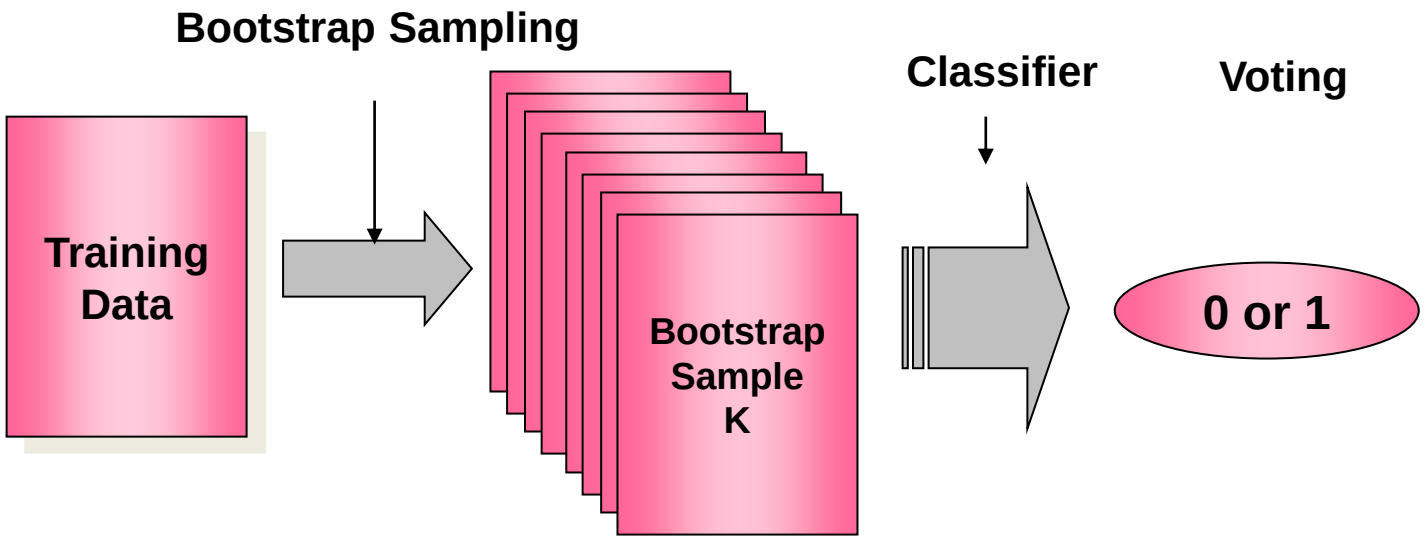



# Concept

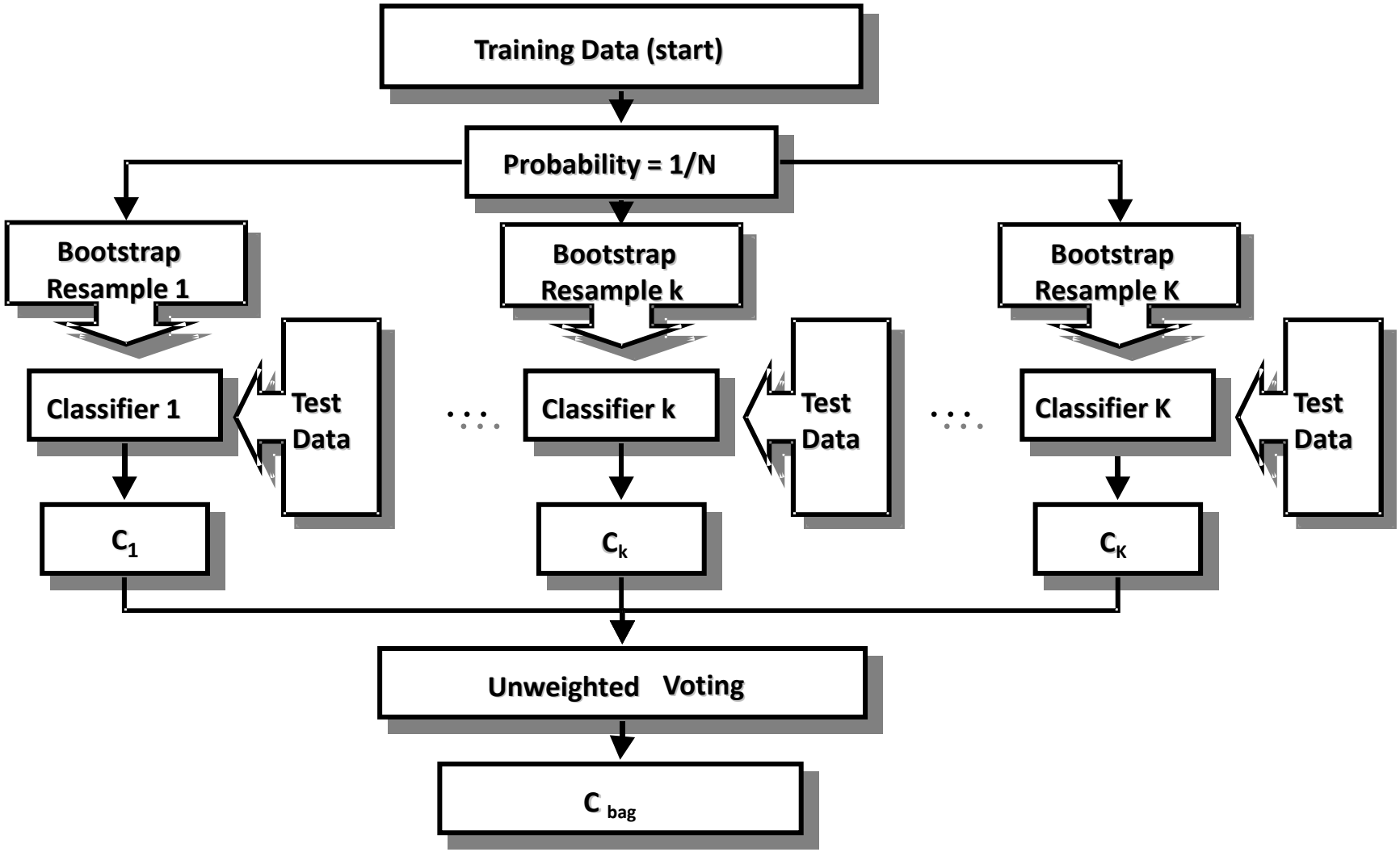

Result of model 1
Result of model 2
Result of model 3
1
N
Simple majority voting
Result of combined model
: error

# 1. Ensemble 개요

- **Combine classification results using multiple classifiers**
  - Extract multiple datasets from a raw dataset using Bootstrap resampling method
  - Combine the results using simple majority voting and weighted voting



# Bagging (Bootstrap Aggregating)



# 1. Ensemble 개요

- 특징

① Bootstrap resample을 K 개 만들어 원래의 데이터를 대체하여 하나의 분류기를 적용하여 resample수 만큼 각각 학습

② $x$ 와 출력 $y$ 로 구성된 Bootstrap resample $k$ 에 대한 분류기 $C_k(x)$의 가능한 예측 범주를 class 0, class 1 로 지정

③ Threshold value issue!

④ K개의 Bootstrap resample의 개수만큼 되풀이하여 아래 식과 같이 $C_{bag}$를 구한 후 $C_{bag}$가 0.5이상이면 해당 관찰치를 class 1 이라 하고 0.5이하이면 class 0 이라 분류

$$C_{bag} = \frac{1}{K}\sum_{k=1}^{K} C_k(x)$$

# 1. Ensemble 개요

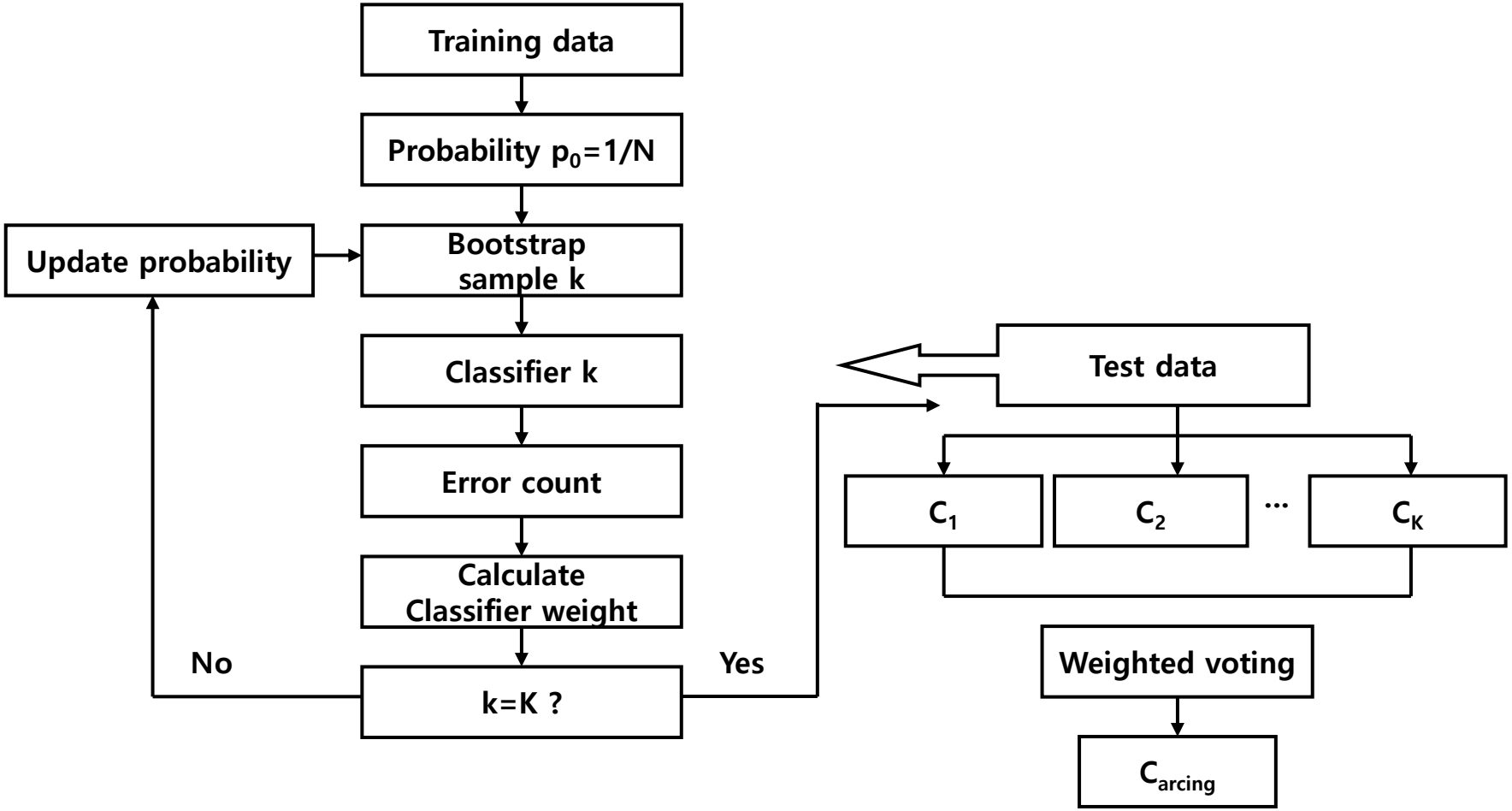
- **Random forest: 대표적인 앙상블 분류모형**
  - **다수의 DT(**randomly selected m inputs per tree)
  - 예측: 각 tree 예측에 대한 mode
  - Random decision forests에서 시작(Tin Kam Ho of Bell Labs in 1995)
  - Breiman's "bagging" idea와 random selection of features (2001)와 결합
  - Bagging (Breiman, 1996), Random Forests (Breiman, 2001), Extremely randomized trees (Geurts et al., 2006)

- **장점**
  - 높은 정확성, 효율적인 수행, 변수 제거 없이도 다수의 변수 활용, 변수 중요도 제공, Voting을 통한 Unbiased 결과, Missing 값이 있어도 좋은 성능

## Boosting, Arcing
### (Adaptively Resample and Combine)

- 기본 아이디어
  - 분류기 성능에 따라 weighted voting
  - 이전 bootstrap sample에서 오분류된 observation은 다음 bootstrap sample 시에 선택되도록 하는 확률을 높여줌으로써 이전에 오분류된 observation 을 정분류할 가능성이 높은 분류기를 생성하도록 함.

# 1. Ensemble 개요



# Arcing (arc-fs)

## Freund & Schapire (1996) AdaBoost

1) Training 데이터에 있는 *N* 개의 관측치에서 각 관측치가 추출될 확률을 *P(i)=1/N*로 같은 값을 적용하여 Bootstrap resample을 실시(i=1,...,N).
2) N개의 관측치를 가지는 k번째 Bootstrap resample로 학습된 분류기 $C_k$를 형성
3) i 번째 경우에 대해서 $C_k$를 이용하여 분류한 결과가 오분류 되었을 때는 1의 값을, 정분류 되었을 때는 0의 값을 갖는 더미변수 d(i)를 정의
4) 분류기 $C_k$의 오분류율 $\varepsilon_k$와 P(i)를 갱신하는데 필요한 $\beta_k$를 계산

$$\varepsilon_k = \sum_{i=1}^{N} P(i)d(i), \qquad \beta_k = \frac{(1-\varepsilon_k)}{\varepsilon_k}$$

- 만일 k번째 step의 $\varepsilon_k$가 0이거나 ½보다 크면 p(i)=1/N 로 하고 1st step에서 다시 시작

# 1. Ensemble 개요

5) $\beta_k$ 를 바탕으로 k+1 번째 Bootstrap resample에서 관측치 i 가 샘플링될 확률을 다음과 같이 갱신한다.

$$P_{k+1}(i) = \frac{P_k(i)\beta_k^{d(i)}}{\sum P_k(i)\beta_k^{d(i)}}$$

6) 이와 같은 과정을 K 번 반복 한 후, 각 분류기의 분류 결과에 $\log(\beta_k)$의 가중치를 주어 가중 평균한 값을 취하게 된다.

$$C_{arc-fs} = \frac{\sum_{k=1}^{K} w_k C_k(x)}{\sum_{k=1}^{K} w_k}$$

$$where \quad w_k = \log(\beta_k)$$

# 1. Ensemble 개요

Gradient Boosting = Gradient Descent + Boosting

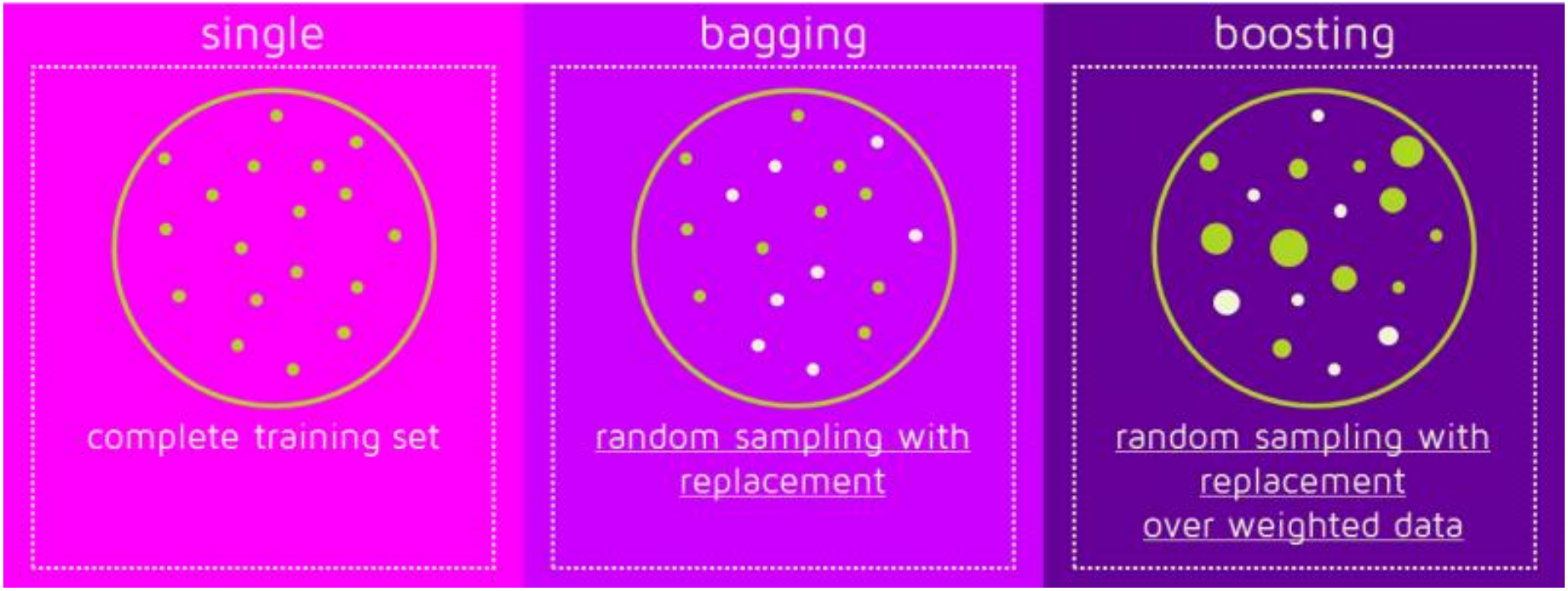
- Fit an additive model (ensemble) $\sum_t \alpha_t h_t$ in a forward stage-wise manner.
- In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.

- In Gradient Boosting,"shortcomings" are identified by gradients.
- In Adaboost,"shortcomings" are identified by high-weight data points.

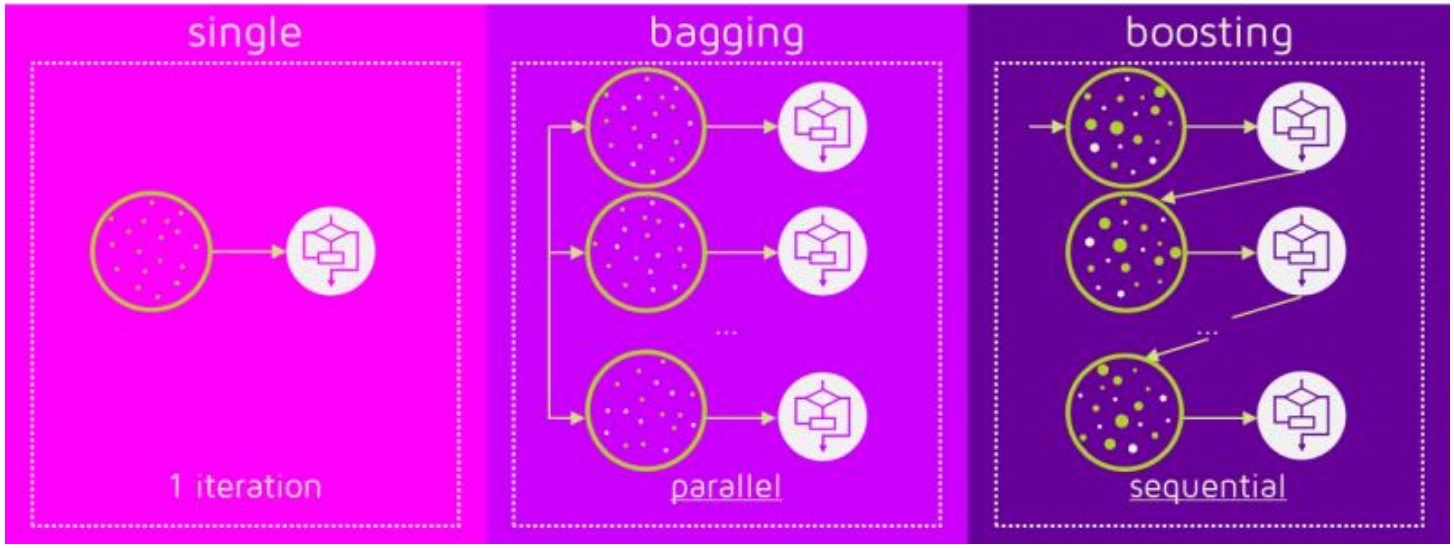
- Formulate Adaboost as gradient descent with a special loss function.
- Generalize Adaboost to Gradient Boosting in order to handle a variety of loss functions.

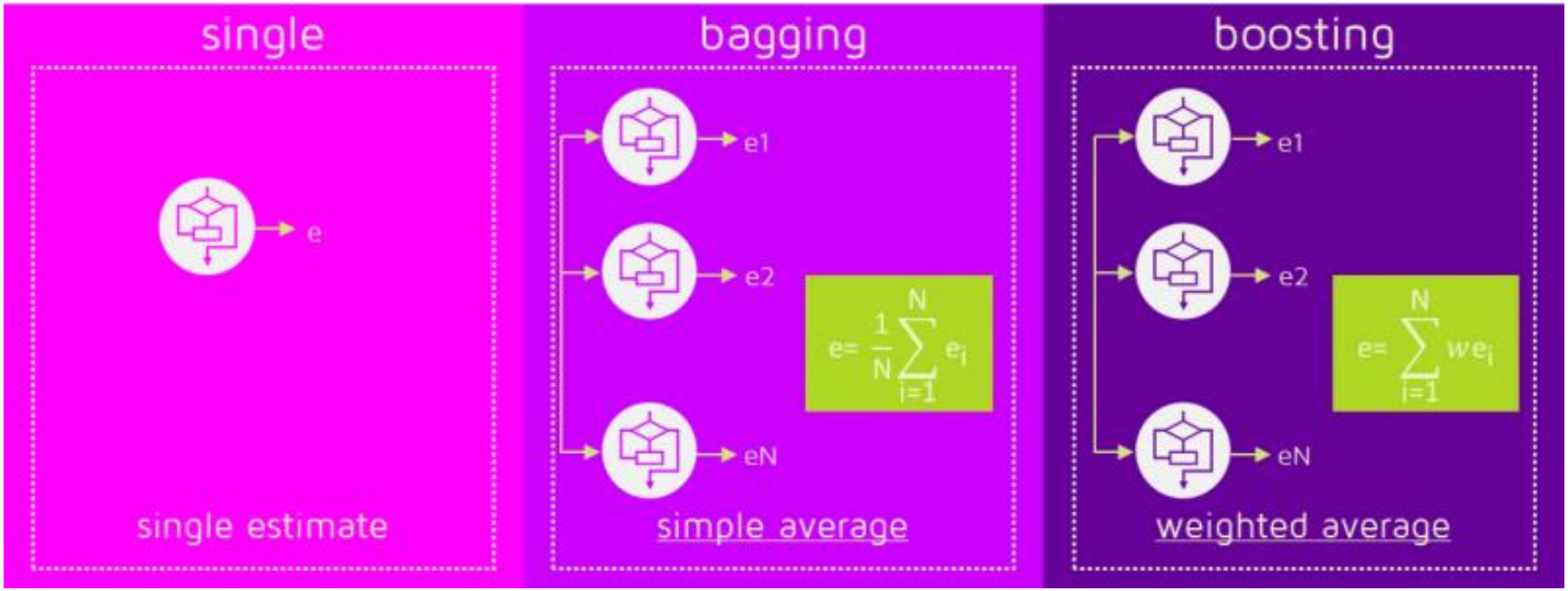
- XGBoost
- Training loss + Complexity of the trees

## Bagging vs Boosting



# 1. Ensemble 개요



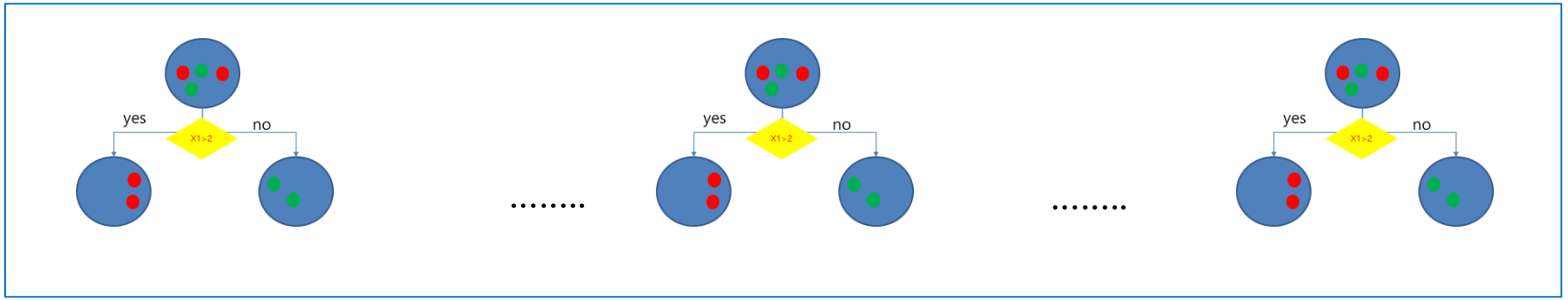
# 2. Bagging - Random Forest

- **Ensemble 기법: 여러 분류 모형의 결과를 결합하는 기법**
- **Random Forest: 앙상블 학습 방법의 일종으로, 훈련 과정에서 구성한 다수의 Decision Tree로부터 Voting을 통해 결과 예측**
- **Bagging: 주어진 데이터에서 랜덤하게 여러 개의 같은 크기의 부분집합을 생성**
- **Out of Bag과 Voting: Out of Bag(OOB)는 Bagging에서 제외되는 데이터들을 의미하며, Voting은 Random Forest내 여러 Decision Tree의 결과 중 다수의 결과를 선택하는 방법**

# 2. Bagging - Random Forest

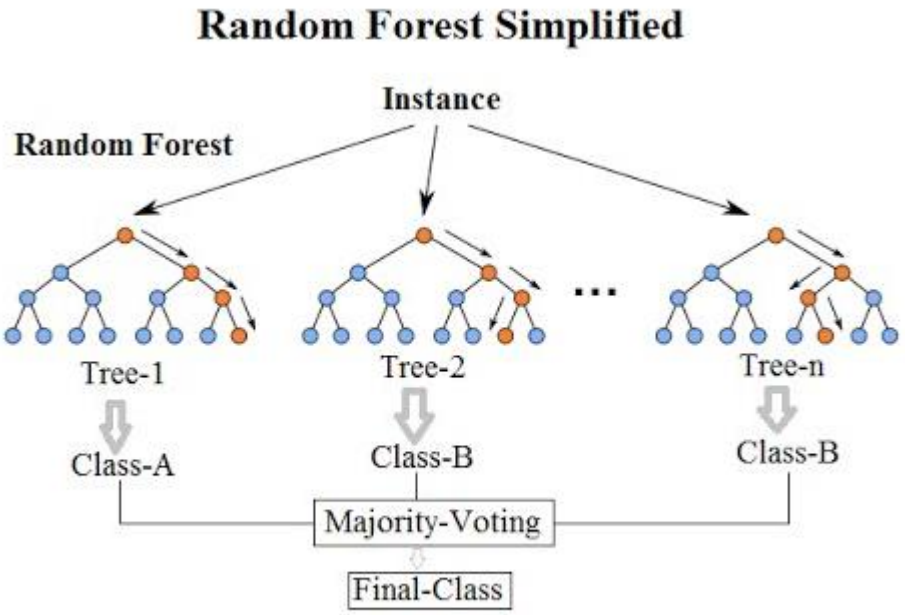
**Random Forest**

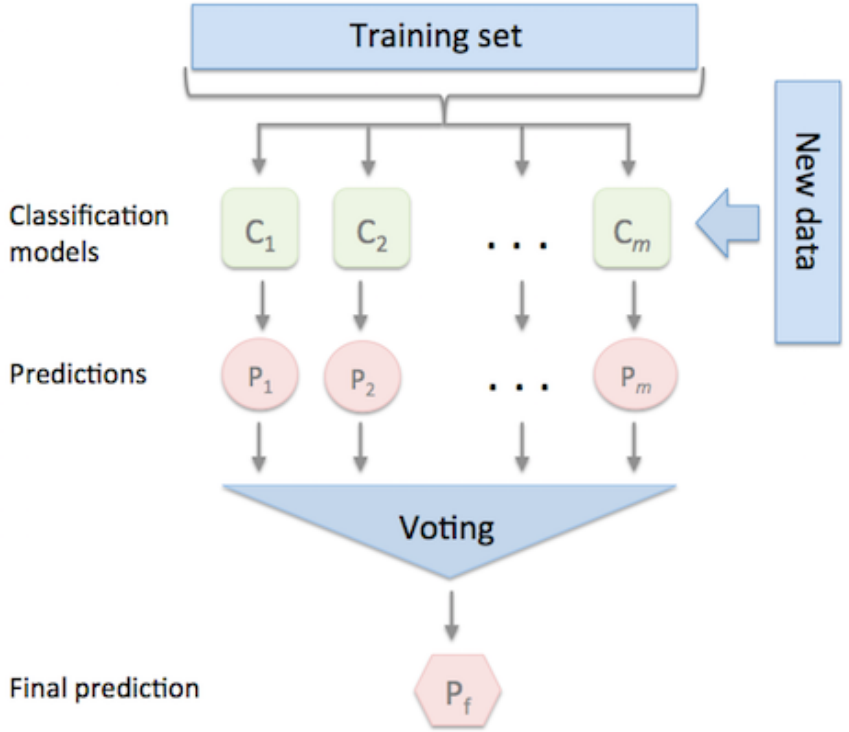
- Breiman의 " bagging " 과 변수 랜덤 선택 아이디어 기반
- 처음에는 random decision forests로 시작하여 발전
- 데이터의 다양한 경우를 반영할 수 있도록 보완
- 다양한 경우에 대한 Decision Tree를 통해 성능과 안정성을 제고



# 2. Bagging - Random Forest

- **Random forest (or random forests)**
  - Ensemble classifier that consists of many decision trees and outputs the class that is the mod e of the class's output by individual trees



# 2. Bagging - Random Forest

- **Algorithm**

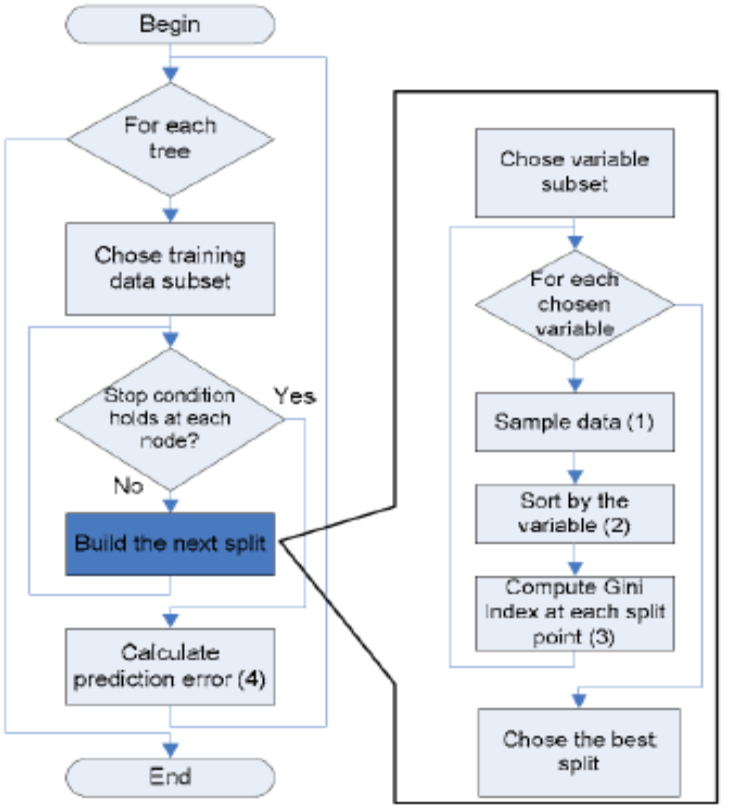
① N: # of training cases / M: 분류기의 변수

② M개 중 m개의 변수가 Tree의 각 노드에서 분류에 사용

③ N개의 training case 중에서 각 tree에 사용되는 n개의 case를 선택 (예: bootstrap sample). 선택되지 않은 Case는 error 추정에 사용

④ 각 tree의 각 노드에서, m개의 변수를 무작위 선택하여 분류에 사용. 이후 m개의 변수로 가장 분류를 잘하도록 계산

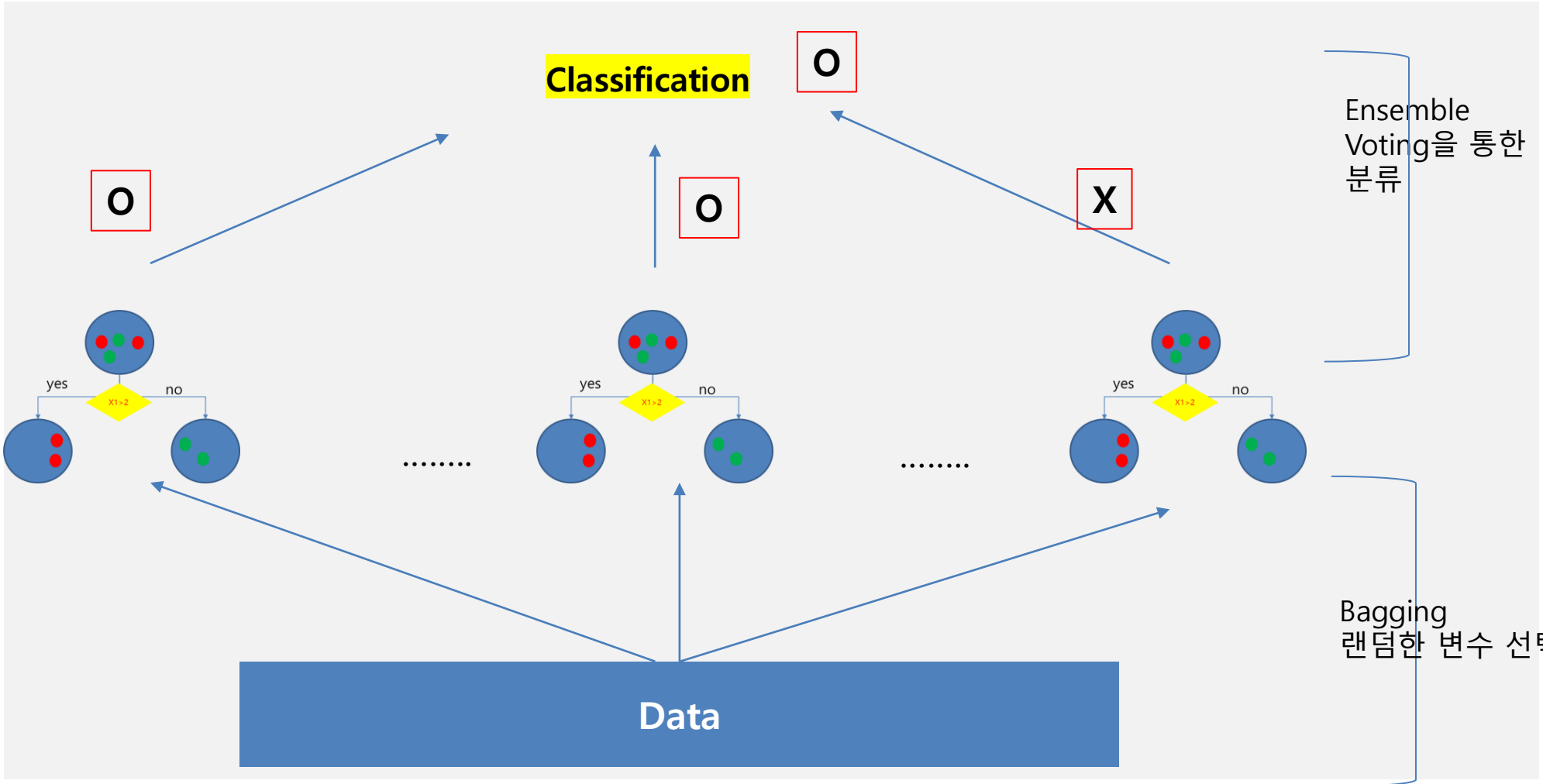
⑤ 각 Tree **fully grown and not pruned**



# 2. Bagging - Random Forest

***Random Forest***

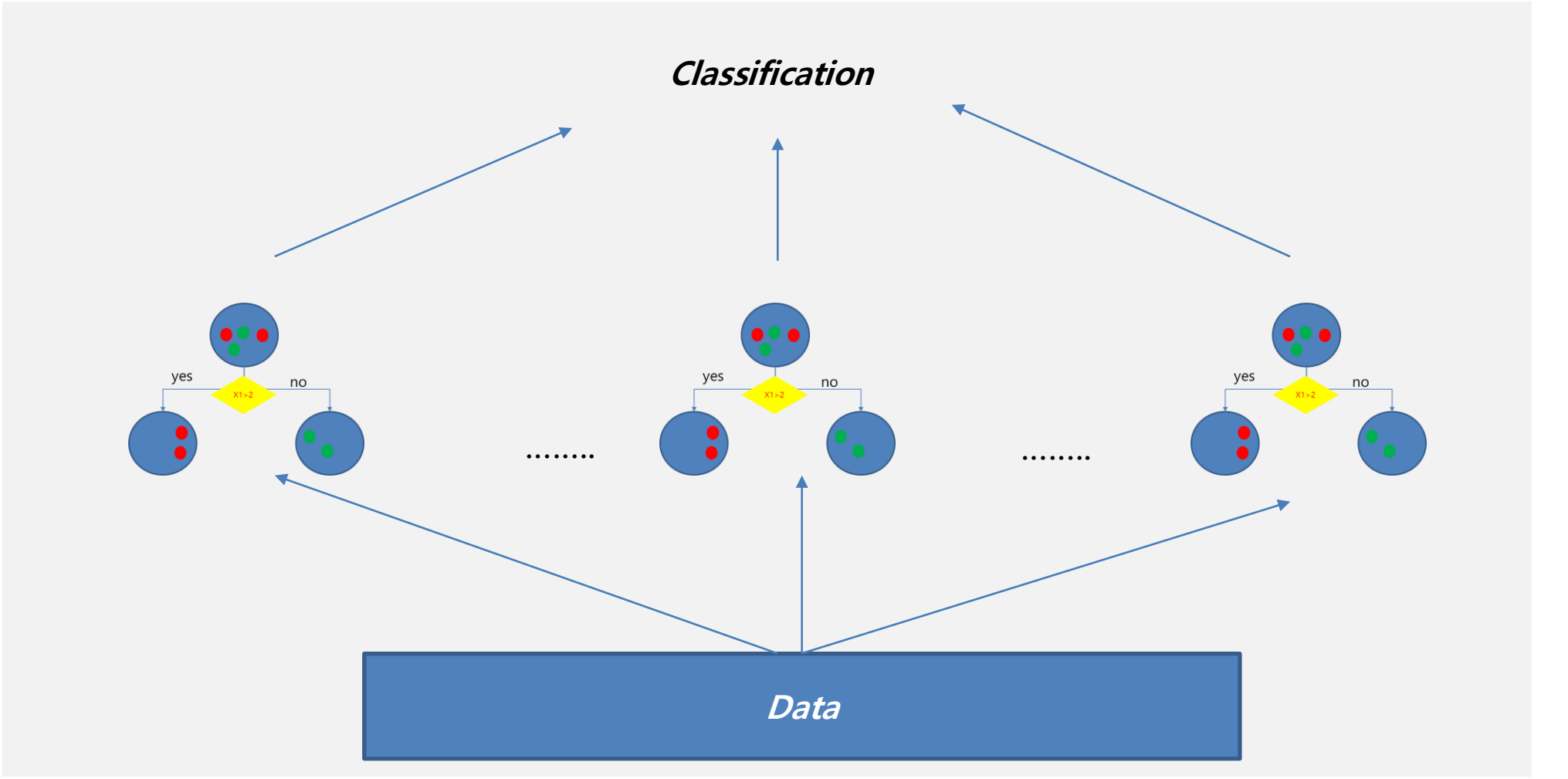
- 데이터의 다양한 경우를 반영할 수 있도록 보완
- 안정성을 제고



# 2. Bagging - Random Forest

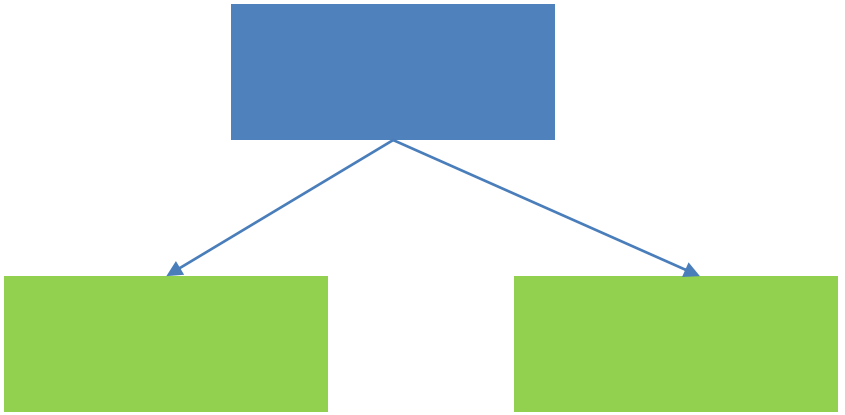
## Random Forest

- 몇 개의 Decision Tree를 만들 것인지?
- 몇 개의 X변수를 Random하게 선택할 것인지?



# 3. Boosting - Adaboost

- Adaboost는 Ensemble 기법의 Boosting을 DT에 적용
- Stump로 부터 학습을 시작
  - Stump: 단순한 형태의 Tree, Weak learner

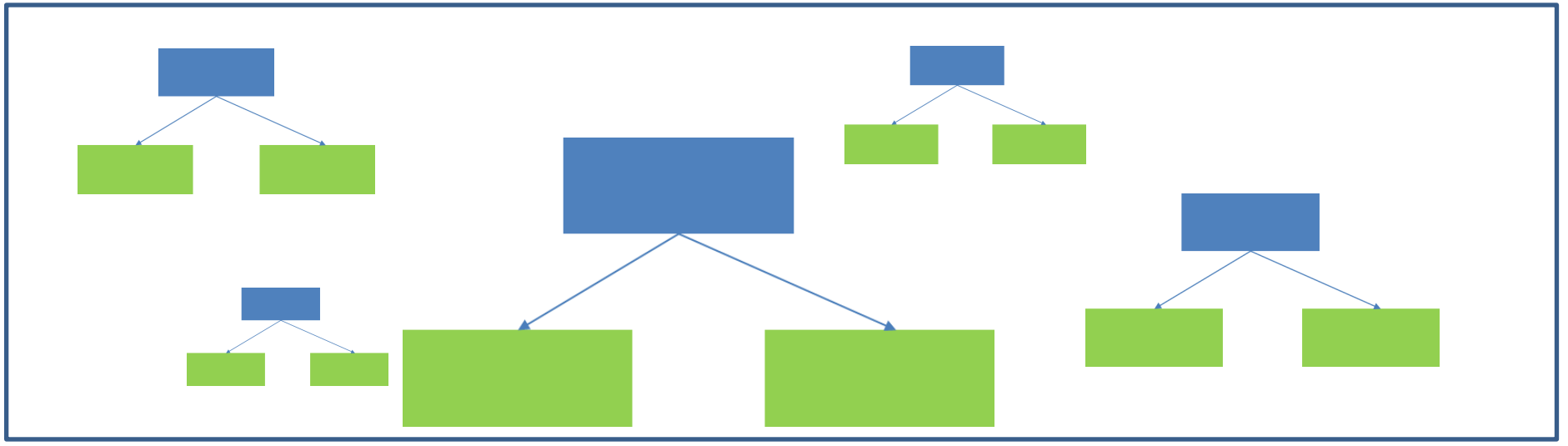


# 3. Boosting - Adaboost

- Forest of stumps를 활용
  - Random Forest: 모든 tree는 같은 weight을 갖음
  - Adaboost: Stump마다 중요도의 차이가 존재

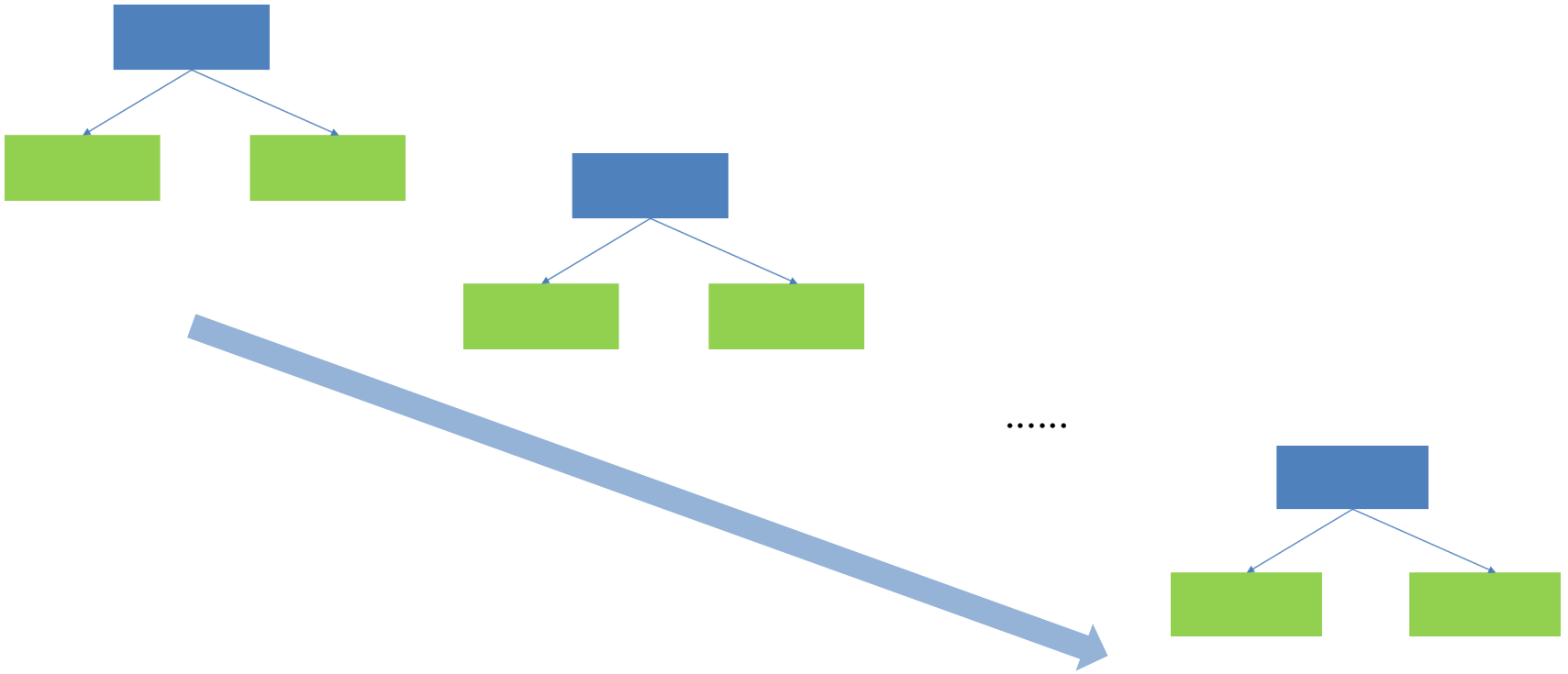
- Random Forest에서는 Tree가 같은 중요도를 지님


- Adaboost에서의 stump의 중요도: Amount of say로 표현, 클 수록 결과에 큰 영향을 미침



# 3. Boosting - Adaboost

- Forest of stumps
  - 첫 stump는 다음 stump에 영향, 순차적으로 다음 stump에 영향을 주는 방식



# 3. Boosting - Adaboost

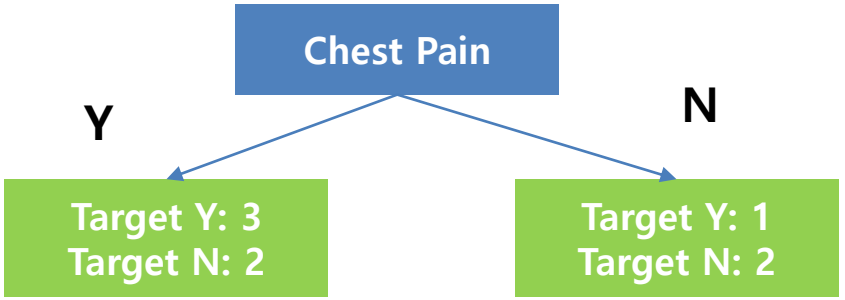
- **Example**

Target　　Sample Weight의 합은 1

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|---|---|---|---|---|
| Y | Y | 205 | Y | 1/8 |
| N | Y | 180 | Y | 1/8 |
| Y | N | 210 | Y | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| N | Y | 156 | N | 1/8 |
| N | Y | 125 | N | 1/8 |
| Y | N | 168 | N | 1/8 |
| Y | Y | 172 | N | 1/8 |

# 3. Boosting - Adaboost

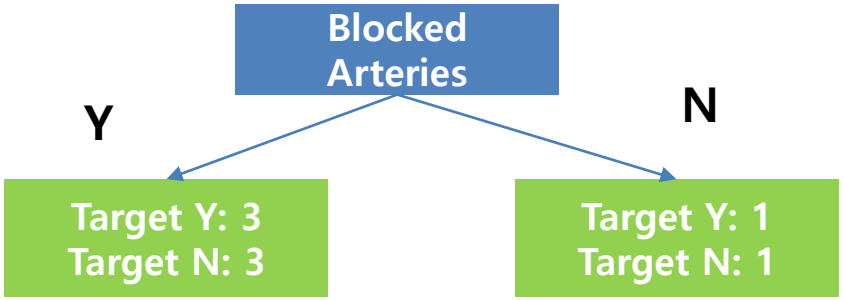
- 각 변수별 Target과의 관계




| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|---|---|---|---|---|
| Y | Y | 205 | Y | 1/8 |
| N | Y | 180 | Y | 1/8 |
| Y | N | 210 | Y | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| N | Y | 156 | N | 1/8 |
| N | Y | 125 | N | 1/8 |
| Y | N | 168 | N | 1/8 |
| Y | Y | 172 | N | 1/8 |

# 3. Boosting - Adaboost

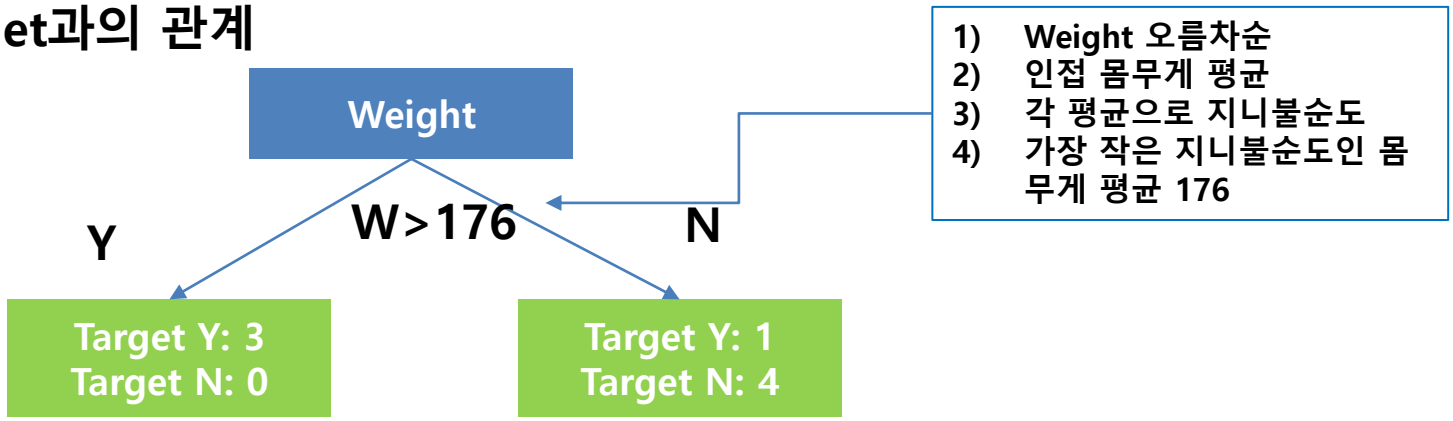
- 각 변수별 Target과의 관계




| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|---|---|---|---|---|
| Y | Y | 205 | Y | 1/8 |
| N | Y | 180 | Y | 1/8 |
| Y | N | 210 | Y | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| N | Y | 156 | N | 1/8 |
| N | Y | 125 | N | 1/8 |
| Y | N | 168 | N | 1/8 |
| Y | Y | 172 | N | 1/8 |

# 3. Boosting - Adaboost

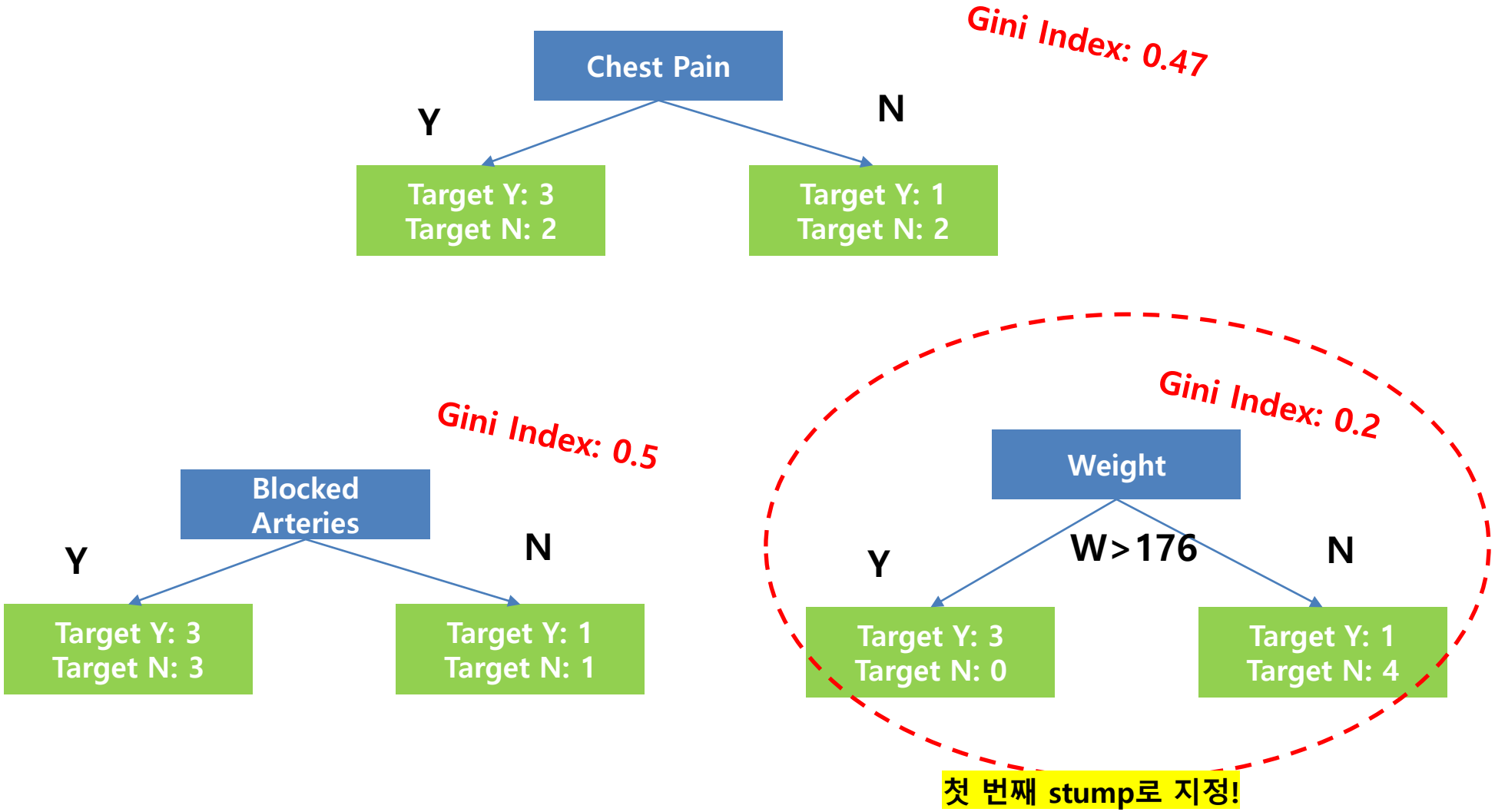
- 각 변수별 Target과의 관계




| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|---|---|---|---|---|
| Y | Y | 205 | Y | 1/8 |
| N | Y | 180 | Y | 1/8 |
| Y | N | 210 | Y | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| N | Y | 156 | N | 1/8 |
| N | Y | 125 | N | 1/8 |
| Y | N | 168 | N | 1/8 |
| Y | Y | 172 | N | 1/8 |

# 3. Boosting - Adaboost

- 각 stump별 지니 계수



# 3. Boosting - Adaboost

- 첫 stump의 Total Error


| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|---|---|---|---|---|
| Y | Y | 205 | Y | 1/8 |
| N | Y | 180 | Y | 1/8 |
| Y | N | 210 | Y | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| N | Y | 156 | N | 1/8 |
| N | Y | 125 | N | 1/8 |
| Y | N | 168 | N | 1/8 |
| Y | Y | 172 | N | 1/8 |

Amount of Say = 0.97

Amount of Say = $\frac{1}{2} log(\frac{1 - Total\ Error}{Total\ Error})$

Amount of Say 계산

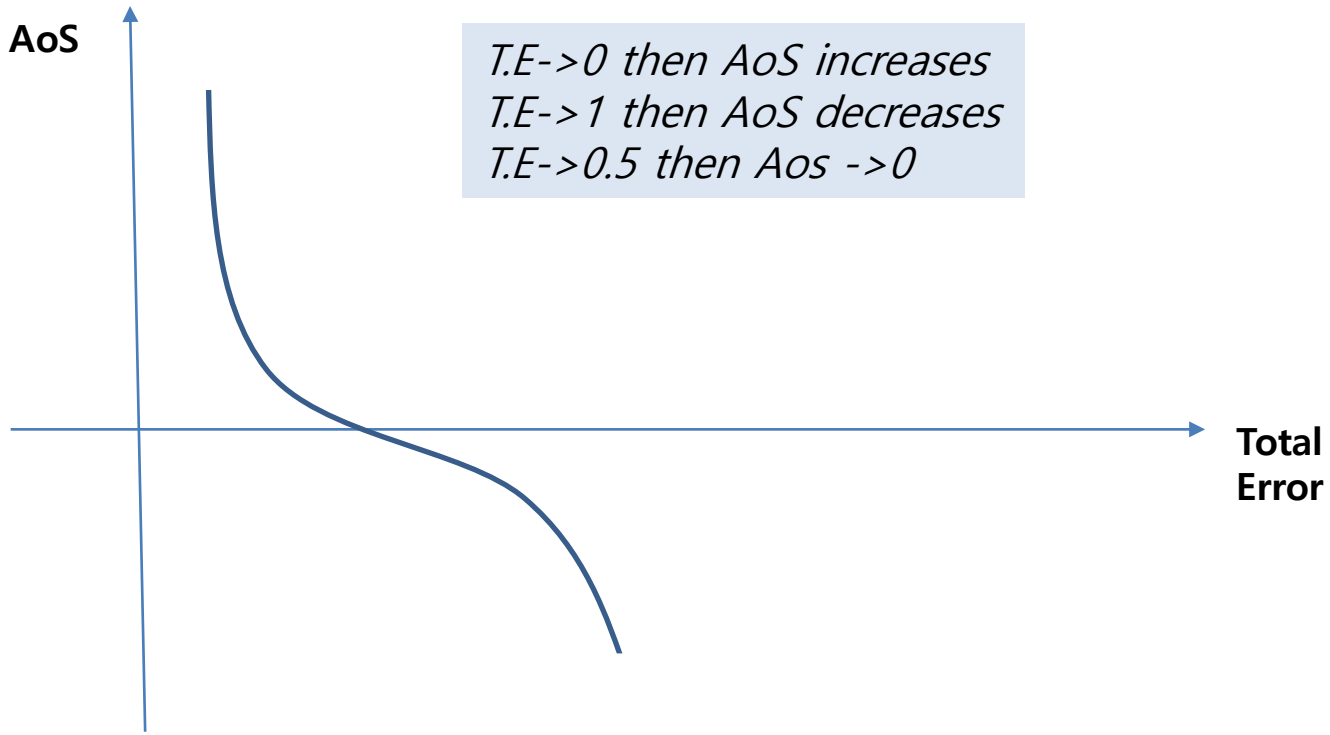
Total Error = 1/8
(0~1사이 값)

*첫 stump에서 틀린 Case*

# 3. Boosting - Adaboost

- 첫 stump의 Total Error

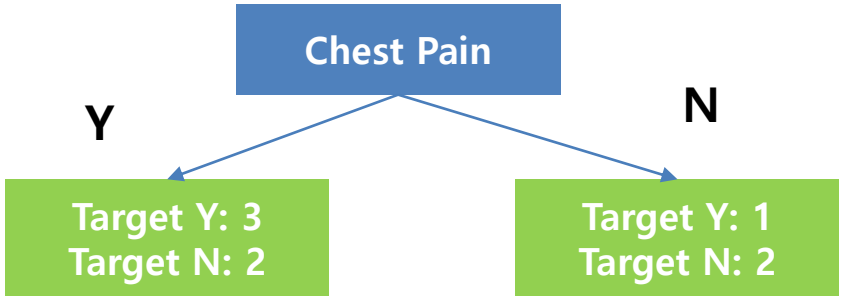
$$\text{Amount of Say} = \frac{1}{2} log\left(\frac{1-Total\ Error}{Total\ Error}\right)$$




*T.E->0 then AoS increases*
*T.E->1 then AoS decreases*
*T.E->0.5 then Aos ->0*

# 3. Boosting - Adaboost

- AoS 계산 예: Chest Pain (실제 stump는 아님)




| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|---|---|---|---|---|
| Y | Y | 205 | Y | 1/8 |
| N | Y | 180 | Y | 1/8 |
| Y | N | 210 | Y | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| N | Y | 156 | N | 1/8 |
| N | Y | 125 | N | 1/8 |
| Y | N | 168 | N | 1/8 |
| Y | Y | 172 | N | 1/8 |

Amount of Say = 0.42

Amount of Say = $\frac{1}{2} log(\frac{1 - Total\ Error}{Total\ Error})$

Amount of Say 계산

Total Error = 3/8
(0~1사이 값)

*Chest Pain stump에서 틀린 Case*

# 3. Boosting - Adaboost

- **두 번째 Stump 계산**
  - 첫 Stump가 잘못 분류한 Sample의 Weight를 높여줌
    - 이후 원래 데이터에서 샘플링을 통해 새롭게 데이터 구성
    - Weight를 활용한 샘플링
  - 첫 Stump에서 오분류된 Sample이 높은 Weight으로 더 많이 Sampling됨
    - 다음 Stump에서 이전 단계에 오분류된 Obs.에 집중

$$\boldsymbol{New\ Sample\ Weight} = \boldsymbol{Sample\ Weight} \times e^{amount\ of\ say}$$

**AoS가 크면 Weight도 증가**

- 첫 stump에서의 4번째 행

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|---|---|---|---|---|
| Y | Y | 205 | Y | 1/8 |
| N | Y | 180 | Y | 1/8 |
| Y | N | 210 | Y | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| N | Y | 156 | N | 1/8 |
| N | Y | 125 | N | 1/8 |
| Y | N | 168 | N | 1/8 |
| Y | Y | 172 | N | 1/8 |

$$\boldsymbol{New\ Sample\ Weight} = \frac{1}{8} \times e^{0.97} = 0.125 \times 2.64 = 0.33$$

# 3. Boosting - Adaboost

- **두 번째 Stump 계산**
  - 첫 Stump에서 정분류된 Sample은 낮은 Weight으로 덜 Sampling
    - 다음 Stump에서 이전 단계에 정분류된 Obs.는 덜 고려함

$$\boldsymbol{New\ Sample\ Weight = Sample\ Weight \times e^{-amount\ of\ say}}$$

**AoS가 크면 Weight는 감소**

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|---|---|---|---|---|
| Y | Y | 205 | Y | 1/8 |
| N | Y | 180 | Y | 1/8 |
| Y | N | 210 | Y | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| N | Y | 156 | N | 1/8 |
| N | Y | 125 | N | 1/8 |
| Y | N | 168 | N | 1/8 |
| Y | Y | 172 | N | 1/8 |

$$\boldsymbol{New\ Sample\ Weight = \frac{1}{8} \times e^{-0.97} = 0.125 \times 0.38 = 0.05}$$

- **Weight의 합이 1이 되도록 정규화**

# 3. Boosting - Adaboost

- 두 번째 Stump를 위한 Sampling

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight | New Weight | Sampling 을 위한 값 |
|---|---|---|---|---|---|---|
| Y | Y | 205 | Y | 1/8 | 0.07 | 0~0.07 |
| N | Y | 180 | Y | 1/8 | 0.07 | 0.07~0.14 |
| Y | N | 210 | Y | 1/8 | 0.07 | 0.14~0.21 |
| Y | Y | 167 | Y | 1/8 | 0.49 | 0.21~0.7 |
| N | Y | 156 | N | 1/8 | 0.07 | 0.7~0.77 |
| N | Y | 125 | N | 1/8 | 0.07 | 0.77~0.84 |
| Y | N | 168 | N | 1/8 | 0.07 | 0.84~0.91 |
| Y | Y | 172 | N | 1/8 | 0.07 | 0.91~1 |

- 0~1사이 난수 생성
- 해당 난수 값이 속하는 행을 선택
- 중복해서 선택 가능
- Weight가 높은 행이 Sampling될 확률이 높음

# 3. Boosting - Adaboost

- 다음 Stump를 위한 Sampling 및 학습 반복

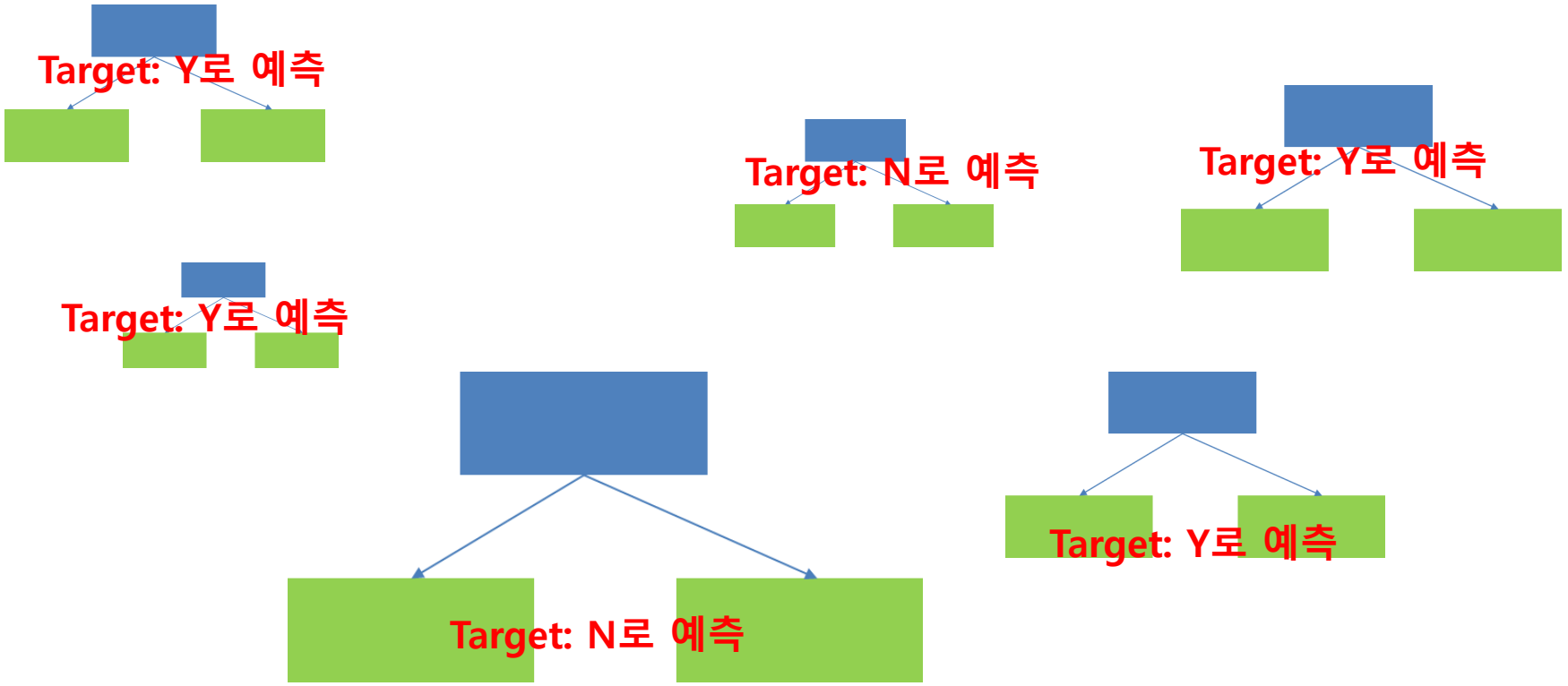
| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight | New Weight | Sampling 을 위한 값 |
|---|---|---|---|---|---|---|
| Y | Y | 205 | Y | 1/8 | 0.07 | 0~0.07 |
| N | Y | 180 | Y | 1/8 | 0.07 | 0.07~0.14 |
| Y | N | 210 | Y | 1/8 | 0.07 | 0.14~0.21 |
| Y | Y | 167 | Y | 1/8 | 0.49 | 0.21~0.7 |
| N | Y | 156 | N | 1/8 | 0.07 | 0.7~0.77 |
| N | Y | 125 | N | 1/8 | | |
| Y | N | 168 | N | 1/8 | | |
| Y | Y | 172 | N | 1/8 | | |

- 다음 계산을 위해 가중치 초기화

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|---|---|---|---|---|
| N | Y | 156 | N | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| N | Y | 125 | N | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| Y | Y | 167 | Y | 1/8 |
| Y | Y | 172 | N | 1/8 |
| Y | Y | 205 | Y | 1/8 |
| Y | Y | 167 | Y | 1/8 |

# 3. Boosting - Adaboost

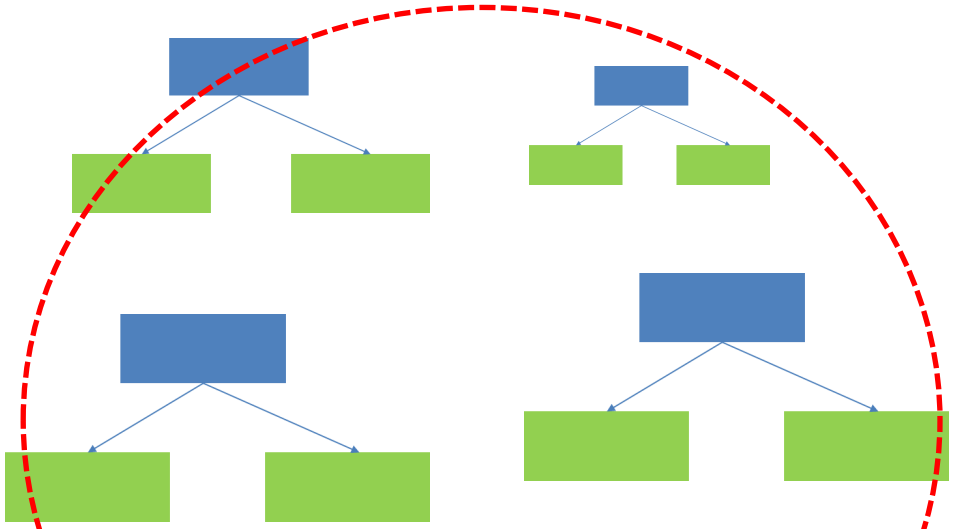
- **Adaboost의 예측 방식**
  - 주어진 X값에 대해 Forest of Stump 내의 Stump에 적용



# 3. Boosting - Adaboost

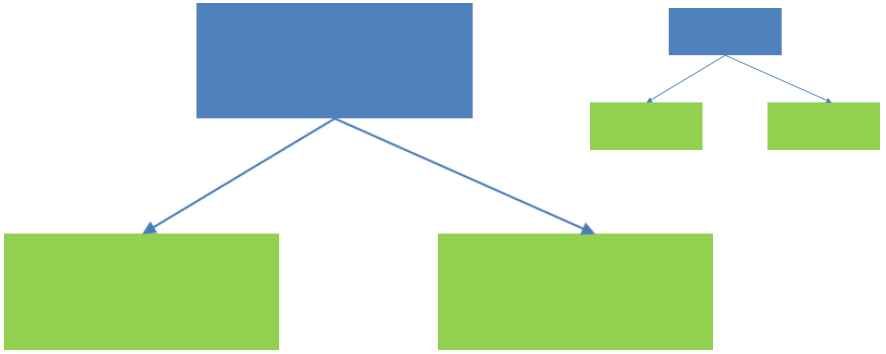
- **Adaboost의 예측 방식**
  - 주어진 X값에 대해 Forest of Stump 내의 Stump에 적용

**Target: Y로 예측**



- Amount of Say: 0.97, 0.32, 0.78, 0.63
- Total AoS: 2.7

**Target: N로 예측**



- Amount of Say: 0.41, 0.82
- Total AoS: 1.23

# 4. Adaboost in detail

Given a training set

$D = \{(x_i , y_i): x_i \in R^d , y_i \in \{-1, +1\}, i = 1, \ldots , m\}.$

Define a distribution over the dataset D such that $\sum_i D(i) = 1$.

Initialize the distribution $D_1$ to be uniform: $D_1(i) = 1/m$.

**Repeat for t = 1, . . . , T:**

1. Learn weak classifier $h_t$ ($h_t : R^d \rightarrow \{-1, +1\}$) using distribution $D_t$.

   1) Compute the weighted error for each weak classifier.

$$\epsilon_t(h) = \sum_{i=1}^{m} D_t(i)\delta(h(\mathbf{x}_i) \neq y_i), \quad \forall h$$

   2) Select the weak classifier with minimum error.

$$h_t = \text{argmin}_h \, \epsilon_t(h), \quad \epsilon_t(h_t) < \frac{1}{2}$$

2. Set weight $\alpha_t$ based on the error:

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} \right)$$

# 4. Adaboost in detail

3. Update the distribution based on the performance so far:

$$D_{t+1}(i) = \frac{1}{Z_t} D_t(i) \exp\left[-\alpha_t y_i h_t(x_i)\right] \rightarrow \exp\left[-y_i \alpha_t h_t(\mathbf{x}_i)\right] = \begin{cases} \exp\left[-\alpha_t\right] < 1 & \text{if } h_t(\mathbf{x}_i) = y_i \\ \exp\left[\alpha_t\right] > 1 & \text{if } h_t(\mathbf{x}_i) \neq y_i \end{cases}$$

where $Z_t$ is a normalization factor to keep $D_{t+1}$ a distribution.

- At each iteration, the weights on the data points are normalized by

$$\begin{aligned} Z_t &= \sum_{\mathbf{x}_i} D_t(\mathbf{x}_i) \exp\left[-y_i \alpha_i h_t(\mathbf{x}_i)\right] \\ &= \sum_{\mathbf{x}_i \in \mathcal{A}} D_t(\mathbf{x}_i) \exp\left[-\alpha_t\right] + \sum_{\mathbf{x}_i \in \overline{\mathcal{A}}} D_t(\mathbf{x}_i) \exp\left[\alpha_t\right] \end{aligned}$$

where $\mathcal{A}$ is the set of correctly classified points: $\{\mathbf{x}_i : y_i = h_t(\mathbf{x}_i)\}$.

4. The strong classifier has the following form: $H_{final}(x) = \text{sign}(\sum_t \alpha_t h_t(x))$

# 5. Gradient Boost

- **Adaboost VS Gradient Boost**
  - Adaboost: 여러 Stump의 순차적 계산
  - GB: leaf로 부터 시작
    - Leaf: Target에 대한 초기 추정값(예: 평균, log(odds ratio) 등)
    - Stump가 아닌 Tree를 생성: 각 tree는 leaf가 8~32개 크기 수준으로 생성

Target

| Height | Color | Gender | Weight |
|---|---|---|---|
| 1.6 | B | M | 88 |
| 1.6 | G | F | 76 |
| 1.5 | B | F | 56 |
| 1.8 | R | M | 73 |
| 1.5 | G | M | 77 |
| 1.4 | B | F | 57 |

# 5. Gradient Boost

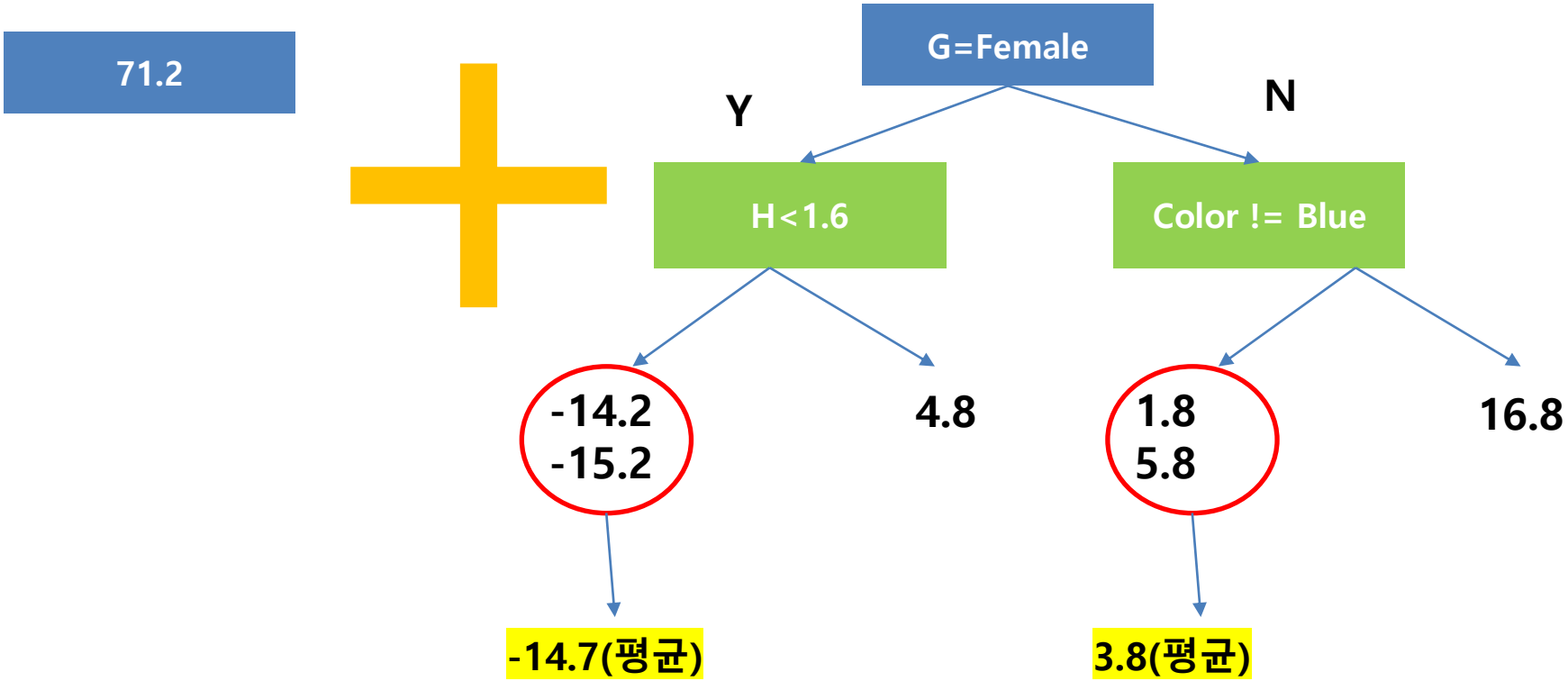
- **Gradient Boost, Step 1**
  - Leaf의 계산
  - Target인 Weight의 평균: 71.2
  - Residual을 계산: 실제값과 예측값의 차이(error)

**같은 X변수들로**
**Residual에 대한 Tree**

| Height | Color | Gender | Weight | Residual |
|---|---|---|---|---|
| 1.6 | B | M | 88 | 16.8 |
| 1.6 | G | F | 76 | 4.8 |
| 1.5 | B | F | 56 | -15.2 |
| 1.8 | R | M | 73 | 1.8 |
| 1.5 | G | M | 77 | 5.8 |
| 1.4 | B | F | 57 | -14.2 |

# 5. Gradient Boost

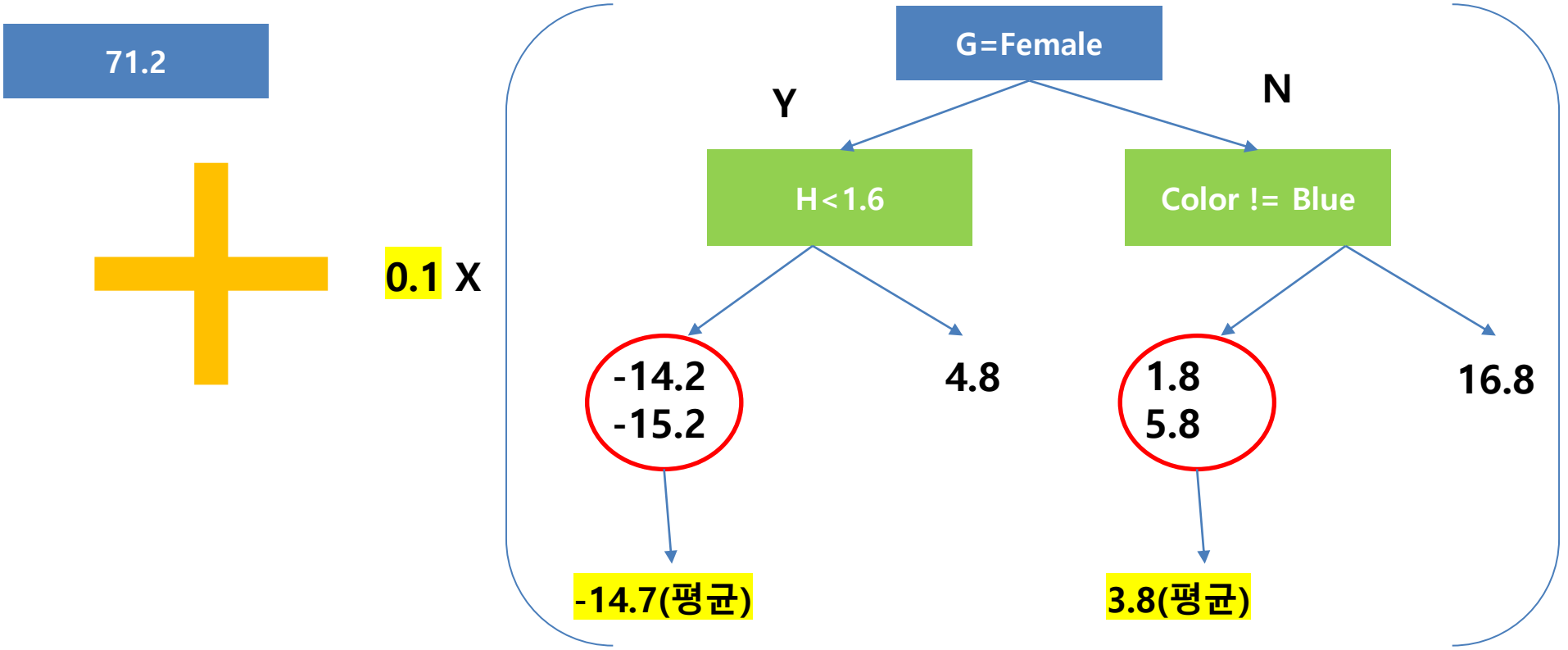
- **Gradient Boost, Step 1**
  - **Leaf + 1st Tree**




- **Male, Blue인 경우 예측 예시:**
  - 71.2 + 16.8 = 88 (관측치와 동일하지만 과적합)
  - Bias는 작지만 Variance 큰 상태

# 5. Gradient Boost

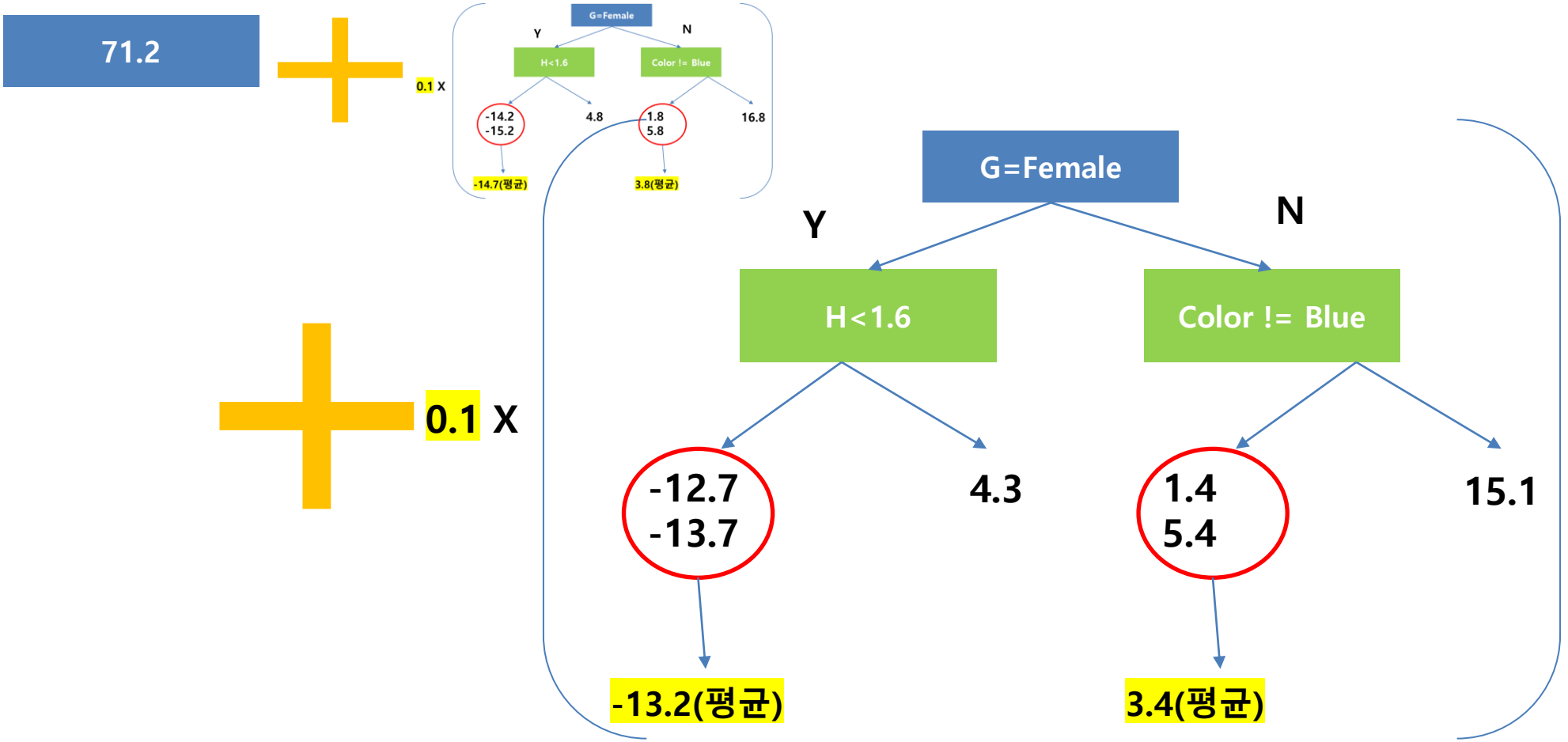
- **Gradient Boost, Step 2**
  - 과적합 방지, 학습속도 조절을 위한 학습율 도입
  - Learning Rate: 0~1사이, 이 예에서는 0.1 사용




- **Male, Blue인 경우 예측 예시:** 71.2 + 0.1 X 16.8 = 72.9
  - 실제값에 가까워지지만, 그 정도가 조절됨 (Gradient의 개념)
  - Variance를 낮게 유지할 수 있음

# 5. Gradient Boost

- **Gradient Boost, Step 3**
  - Learning Rate: 0~1사이, 이 예에서는 0.1 사용




- **H=1.6, Male, Blue인 경우 예측 예시:** 71.2 + 0.1 X 16.8 + 0.1 X 15.1 = 74.4

# 5. Gradient Boost

- **Gradient Boost, Step 3**
  - 학습율 반영 예측값을 통한 두 번째 Residual 계산

같은 X변수들로 New Residual에 대한 Tree

| Height | Color | Gender | Weight | Residual | Residual(new) |
|---|---|---|---|---|---|
| 1.6 | B | M | 88 | 16.8 | 15.1 |
| 1.6 | G | F | 76 | 4.8 | 4.3 |
| 1.5 | B | F | 56 | -15.2 | -13.7 |
| 1.8 | R | M | 73 | 1.8 | 1.4 |
| 1.5 | G | M | 77 | 5.8 | 5.4 |
| 1.4 | B | F | 57 | -14.2 | -12.7 |

Residual 크기 감소

# 5. Gradient Boost

- **Gradient Boost**
  - 위의 과정을 계속 반복
    - 정해진 iteration한도 까지 반복
    - 또는 이전 단계와 이후 단계의 Residual 차이가 없을 때까지 반복

  - 매 iteration에서의 Tree의 leaf는 8~32개 사이에서 생성
  - 매 iteration마다 다르게 생성
    - $1^{st}$ tree: leaf 8개
    - $2^{nd}$ tree: leaf 32개
    - $3^{rd}$ tree: leaf 16개
    - …

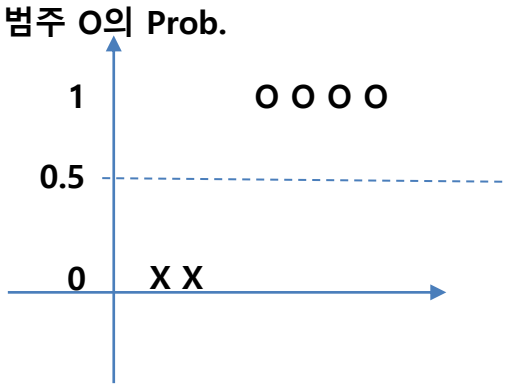
# 5. Gradient Boost

- **Gradient Boost for Classification, Step 1**
  - Leaf의 계산
    - X 범주 2개 대비 O범주는 4개, Odds = 4/2,  leaf는 log(odds) = 0.7

| X1 | X2 | X3 | Target |
|---|---|---|---|
| Y | 12 | Blue | O |
| Y | 87 | Green | O |
| N | 44 | Blue | X |
| Y | 19 | Red | X |
| N | 32 | Green | O |
| N | 14 | Blue | O |

# 5. Gradient Boost

- **Gradient Boost for Classification, Step 1**
  - Leaf의 계산: X 범주 2개 대비 O범주는 4개, Odds = 4/2, leaf는 log(odds) = 0.7
  - Leaf를 통한 O 범주의 확률?
    - Exponential(log(odds)) / (1+exponential(log(odds))) = 0.7
    - 이 값이 기준인 0.5와 비교하여 O, X 분류
  - Residual을 계산: 예를 들어 O는 확률 1이고, leaf 는 0.7이어서 Residual은 0.3

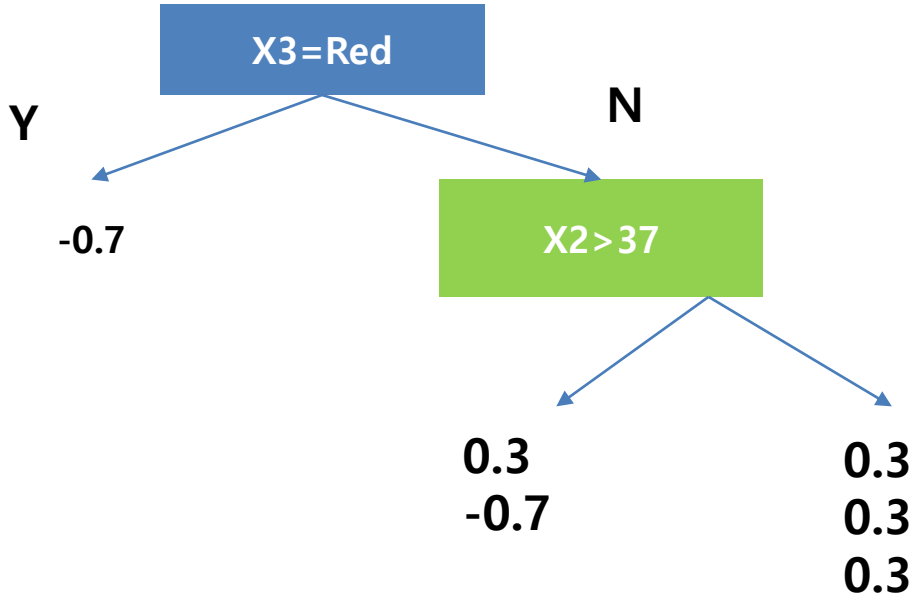



**같은 X변수들로 Residual에 대한 Tree**

| X1 | X2 | X3 | Target | Residual |
|---|---|---|---|---|
| Y | 12 | Blue | O | 0.3 |
| Y | 87 | Green | O | 0.3 |
| N | 44 | Blue | X | -0.7 |
| Y | 19 | Red | X | -0.7 |
| N | 32 | Green | O | 0.3 |
| N | 14 | Blue | O | 0.3 |

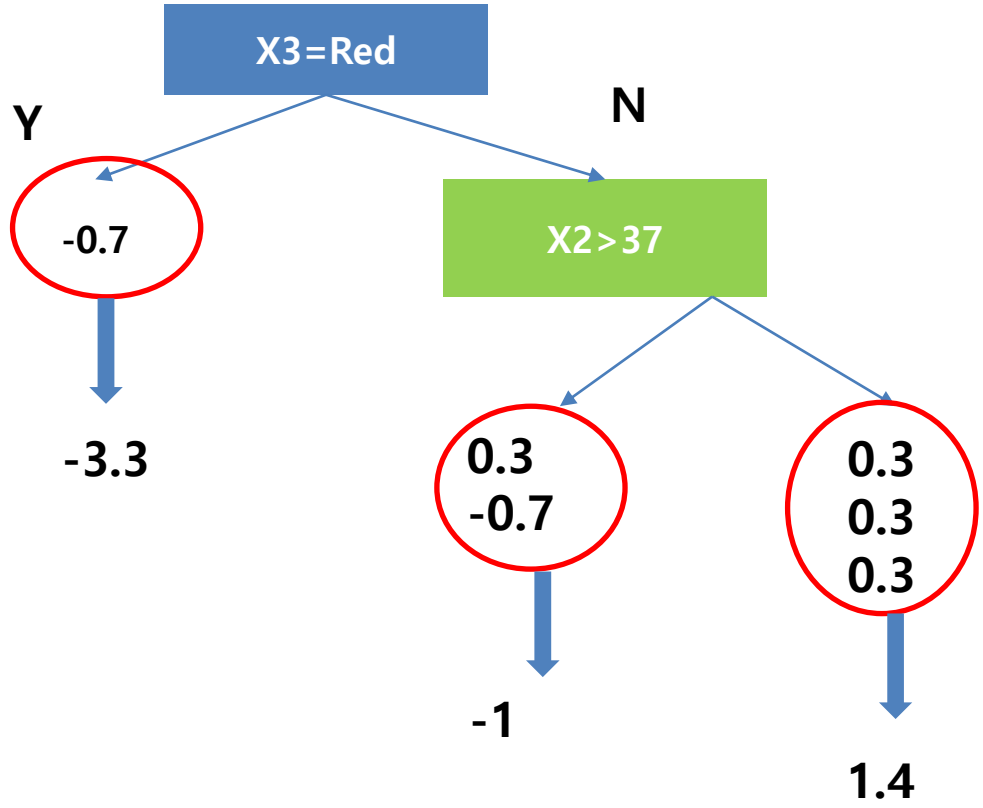
# 5. Gradient Boost

- **Gradient Boost, Step 1**
  - **1st Tree**
    - leaf의 수를 8~32로 제한하며 그 범위내에서 tree 생성



# 5. Gradient Boost

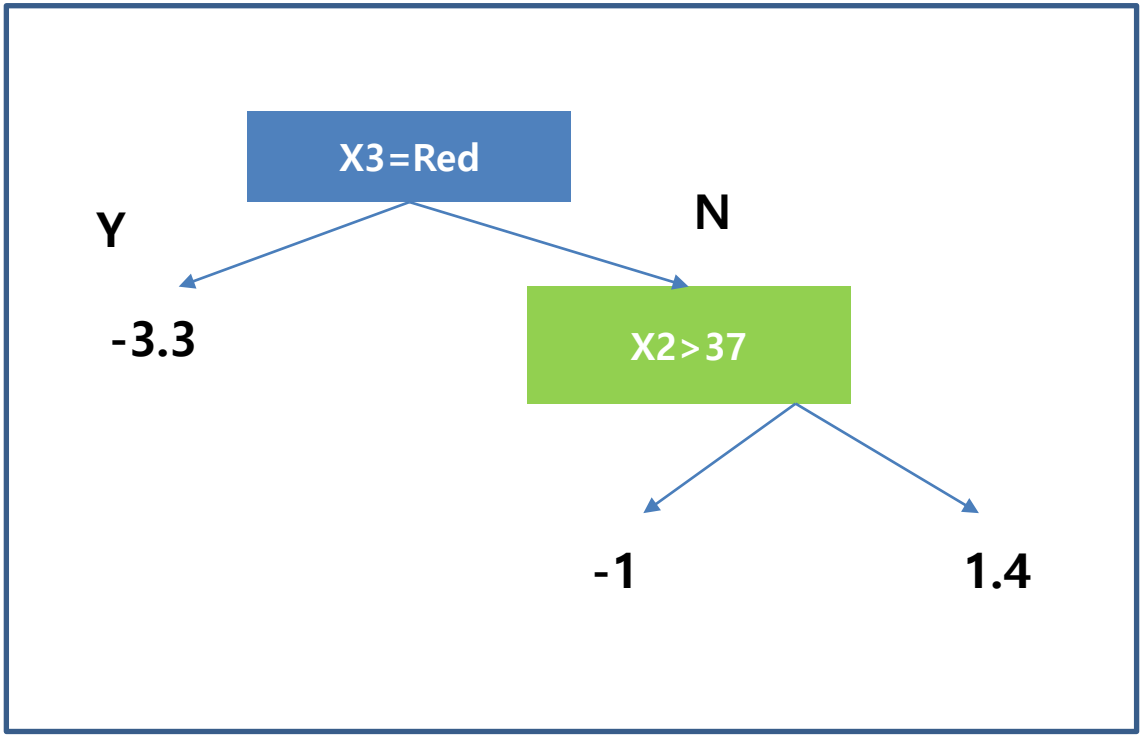
- **Gradient Boost, Step 1**
  - **1st Tree**




$$\frac{\sum Residuals}{\sum(Previous\ Prob\ \times (1 - Previous\ Prob))}$$

# 5. Gradient Boost

- **Gradient Boost, Step 2**
  - Leaf 의 initial prediction에 tree에 학습율 반영하여 계산
  - **Leaf + 1st Tree**

# 5. Gradient Boost

- **Gradient Boost for Classification, Step 3**
  - 각 범주에 대한 발생 확률 계산
    - 1st Obs의 업데이트된 log(odds)는 1.8
      - Leaf 0.7 + 1.4(from tree) X 0.8 = 1.8
      - 1st Obs의 확률: $\frac{e^{1.8}}{1+e^{1.8}}$

| X1 | X2 | X3 | Target | Residual | Prob. |
|---|---|---|---|---|---|
| Y | 12 | Blue | O | 0.3 | 0.9 |
| Y | 87 | Green | O | 0.3 | 0.5 |
| N | 44 | Blue | X | -0.7 | 0.5 |
| Y | 19 | Red | X | -0.7 | 0.1 |
| N | 32 | Green | O | 0.3 | 0.9 |
| N | 14 | Blue | O | 0.3 | 0.9 |

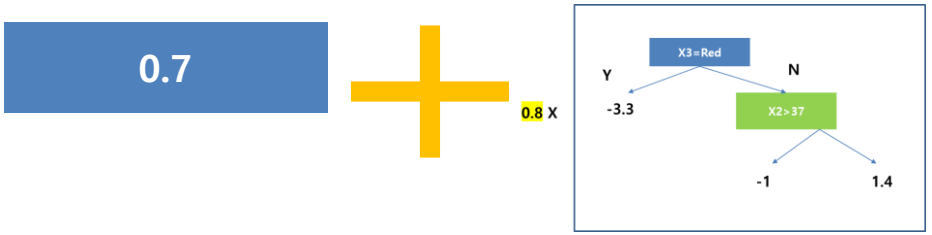
# 5. Gradient Boost

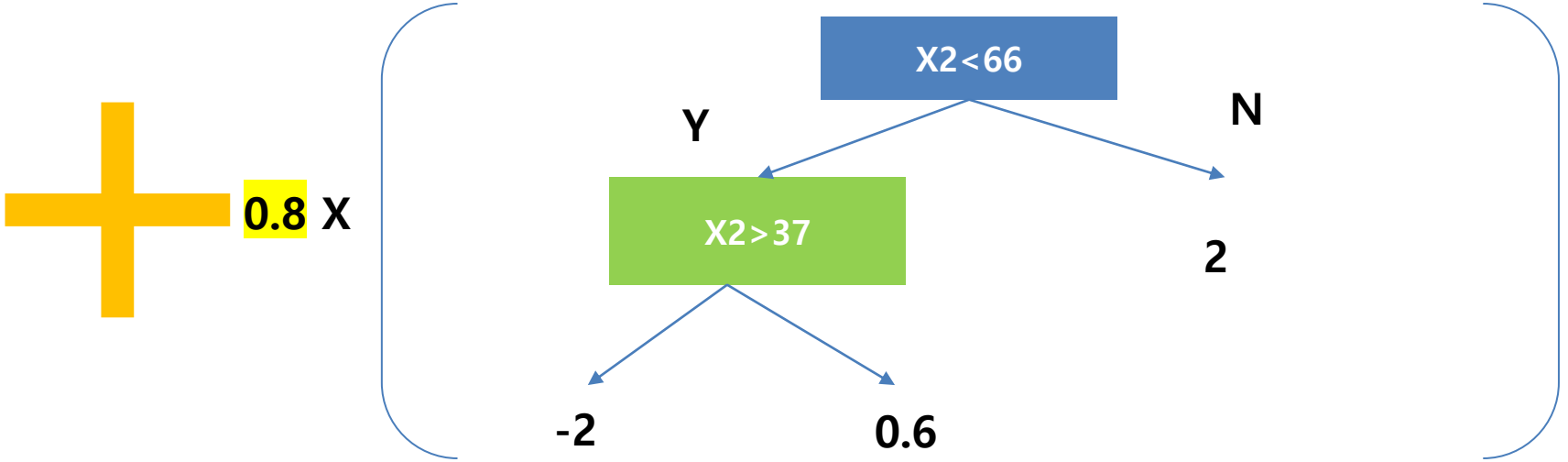
- **Gradient Boost for Classification, Step 3**
  - Residual 다시 계산, 다음 tree 생성

| X1 | X2 | X3 | Target | Residual | Prob. | New Residual |
|---|---|---|---|---|---|---|
| Y | 12 | Blue | O | 0.3 | 0.9 | 1-0.9 |
| Y | 87 | Green | O | 0.3 | 0.5 | 1-0.5 |
| N | 44 | Blue | X | -0.7 | 0.5 | 0-0.5 |
| Y | 19 | Red | X | -0.7 | 0.1 | 0-0.1 |
| N | 32 | Green | O | 0.3 | 0.9 | 1-0.9 |
| N | 14 | Blue | O | 0.3 | 0.9 | 1-0.9 |

# 5. Gradient Boost

- **Gradient Boost, Step 3**
  - Learning Rate: 0~1사이, 이 예에서는 0.1 사용



## Gradient Boosting with Example

### Problem

Recognize the given hand written capital letter.

- Multi-class classification
- 26 classes. A,B,C,...,Z

### Data Set

- http://archive.ics.uci.edu/ml/datasets/Letter+Recognition
- 20000 data points, 16 features

### Feature Extraction

| | | | |
|---|---|---|---|
| 1 | horizontal position of box | 9 | mean y variance |
| 2 | vertical position of box | 10 | mean x y correlation |
| 3 | width of box | 11 | mean of x * x * y |
| 4 | height of box | 12 | mean of x * y * y |
| 5 | total number on pixels | 13 | mean edge count left to right |
| 6 | mean x of on pixels in box | 14 | correlation of x-ege with y |
| 7 | mean y of on pixels in box | 15 | mean edge count bottom to top |
| 8 | mean x variance | 16 | correlation of y-ege with x |

Feature Vector= (2, 1, 3, 1, 1, 8, 6, 6, 6, 6, 5, 9, 1, 7, 5, 10)

Label = G

# 6. Gradient Boosting in detail

## Model

- 26 score functions : $F_A$, $F_B$, $F_C$, ...,$F_Z$.
- $F_A(x)$ assigns a score for class A
- scores are used to calculate probabilities

$$P_A(x) = \frac{e^{F_A(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

$$P_B(x) = \frac{e^{F_B(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

...

$$P_Z(x) = \frac{e^{F_Z(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

- predicted label = class that has the highest probability

# 6. Gradient Boosting in detail

Loss Function for each data point

Step 1 turn the label $y_i$ into a (true) probability distribution $Y_c(x_i)$

For example: $y_5$=G,

$Y_A(x_5) = 0$, $Y_B(x_5) = 0$, ...., $Y_G(x_5) = 1$, ....., $Y_Z(x_5) = 0$

Step 2 calculate the predicted probability distribution $P_c(x_i)$ based on the current model $F_A$, $F_B$, ..., $F_Z$.

$P_A(x_5) = 0.03$, $P_B(x_5) = 0.05$, ..., $P_G(x_5) = 0.3$, ..., $P_Z(x_5) = 0.05$

Step 3 calculate the difference between the true probability distribution and

the predicted probability distribution.

Here we use KL-divergence

# Q&A