

18.19기 정규세션

---

ToBigs 18기 강연자  
표솔빈

# Ensemble

앙상블

# Contents

---

Unit 01 | Introduction

---

Unit 02 | Ensemble

---

Unit 03 | Voting

---

Unit 04 | Bagging

---

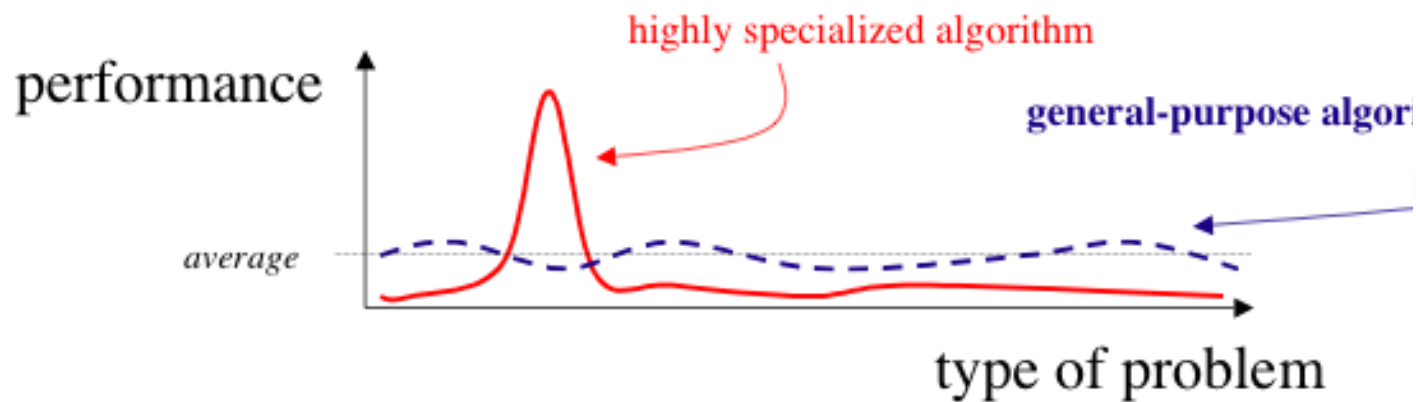
Unit 05 | Boosting

# Unit 01 - Introduction

## Unit 01 | Introduction

### No Free Lunch Theorem (NFLT)

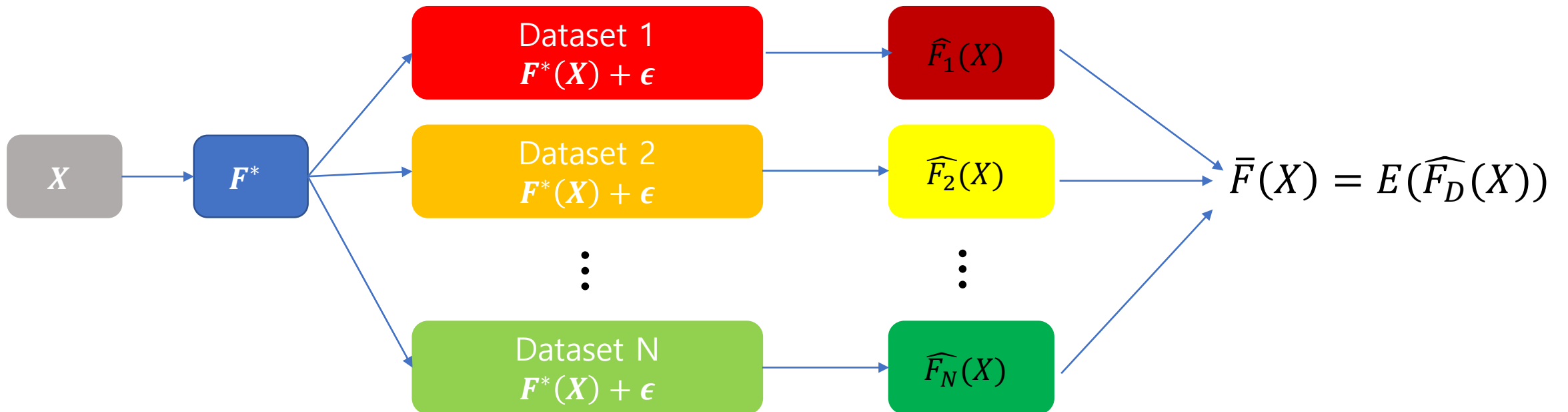
*"We have dubbed the associated results "No Free Lunch" theorems because they demonstrate that if an algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems."* - 「No Free Lunch Theorems for Optimization(1997)」 (William Macready)



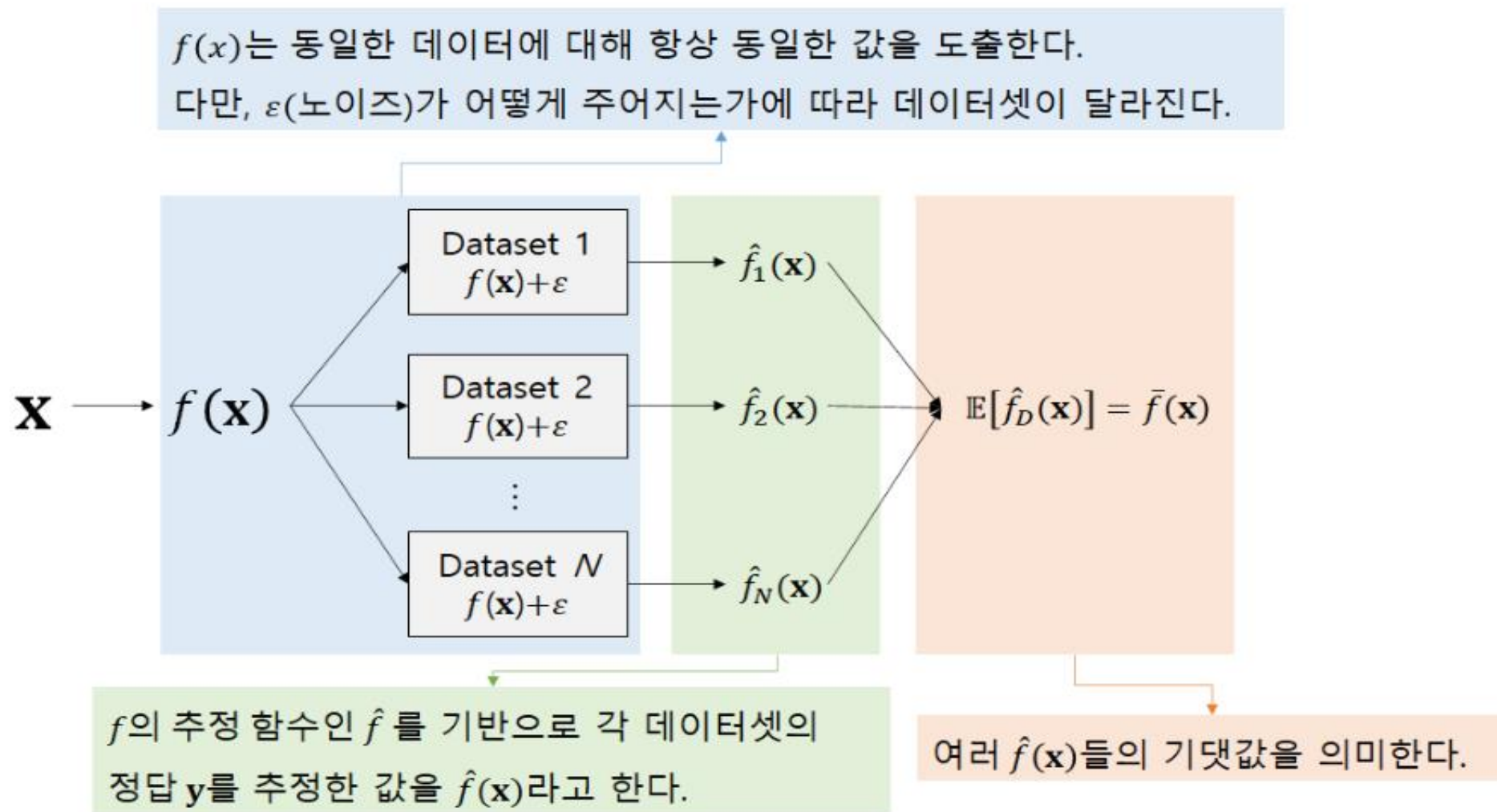
## Unit 01 | Introduction

### Additive Error Model

$$y = F^*(X) + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$



## Unit 01 | Introduction



## Unit 01 | Introduction

### Additive Error Model (참고)

$$y = F^*(X) + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$

<i>Notation</i>	<i>Description</i>
$y = f(\mathbf{x}) + \varepsilon$	데이터
$f$	데이터를 잘 설명하는 정답 함수 (정답 매커니즘)
$\varepsilon$	데이터에 내재된 사람이 컨트롤 불가한 자연 발생적인 변동성
$\varepsilon \sim N(0, \sigma^2)$	변동성이 정규분포(평균=0, 분산= $\sigma^2$ )를 따른다고 가정
$\hat{f}$	정답 함수 $f$ 를 추정하는 함수 (학습 모형)
$\hat{f}(\mathbf{x})$	학습 모형( $\hat{f}$ )을 통해 추정한 값
$\bar{f}(\mathbf{x}) = \mathbb{E}[\hat{f}(\mathbf{x})]$	여러 $\hat{f}(\mathbf{x})$ 들의 기댓값
$f(\mathbf{x}) - \bar{f}(\mathbf{x})$	$\hat{f}(\mathbf{x})$ 의 편향: 정답과 추정치의 기댓값 간의 차이
$\mathbb{E}\left[\left(\hat{f}(\mathbf{x}) - \bar{f}(\mathbf{x})\right)^2\right]$	$\hat{f}(\mathbf{x})$ 의 분산: 추정치의 개별 값들과 추정치의 기댓값 간의 차이를 제공한 것의 기댓값

$$\bar{F}(X) = E(\widehat{F_D}(X))$$

### Bias-Variance Decomposition

#### Calculate Error (MSE)

$$\begin{aligned} Err(X_0) &= E[y - \hat{F}(X)|X = X_0]^2 \\ &= E[F^*(X_0) + \epsilon - \hat{F}(X_0)]^2 \\ &= E[F^*(X_0) - \bar{F}(X_0) + \bar{F}(X_0) - \hat{F}(X_0)]^2 + \sigma^2 \\ &= [F^*(X_0) - \bar{F}(X_0)]^2 + [\bar{F}(X_0) - \hat{F}(X_0)]^2 + \sigma^2 \\ &= \text{bias}^2 + \text{varinace} + \sigma^2 \end{aligned}$$



## Unit 01 | Introduction

### Bias-Variance Decomposition (참고)

### Calculate Error (MSE) 노이즈의 분산 유도과정

$$\begin{aligned}\mathbb{E} \left[ \left( y - \hat{f}(\mathbf{x}) \right)^2 \right] &= \mathbb{E} \left[ \left( f(\mathbf{x}) + \varepsilon - \hat{f}(\mathbf{x}) \right)^2 \right] \\ &= \mathbb{E} \left[ \left( f(\mathbf{x}) - \hat{f}(\mathbf{x}) \right)^2 + \varepsilon^2 + 2 \left( f(\mathbf{x}) - \hat{f}(\mathbf{x}) \right) \varepsilon \right] \\ &= \mathbb{E} \left[ \left( f(\mathbf{x}) - \hat{f}(\mathbf{x}) \right)^2 \right] + \mathbb{E}[\varepsilon^2] + 2 * \mathbb{E} \left[ f(\mathbf{x}) - \hat{f}(\mathbf{x}) \right] * \mathbb{E}[\varepsilon] \\ &= \mathbb{E} \left[ \left( f(\mathbf{x}) - \hat{f}(\mathbf{x}) \right)^2 \right] + \mathbb{E}[\varepsilon^2] \\ &= \mathbb{E} \left[ \left( f(\mathbf{x}) - \hat{f}(\mathbf{x}) \right)^2 \right] + \sigma^2\end{aligned}$$

$\varepsilon \sim N(0, \sigma^2)$  이므로,  $\mathbb{E}[\varepsilon] = 0$ ,  
따라서 해당 부분이 전부 0 이 된다.

분산 공식  
 $Var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$  에 따라,  
 $\mathbb{E}[\varepsilon^2] = Var(\varepsilon) + \mathbb{E}[\varepsilon]^2$  인데,  
 $\mathbb{E}[\varepsilon] = 0$  이므로  
 $\mathbb{E}[\varepsilon^2] = Var(\varepsilon) = \sigma^2$  이 된다.

# Unit 01 | Introduction

## Bias-Variance Decomposition (참고)

Calculate Error (MSE) : 편향의 제곱과 분산 분해 유도

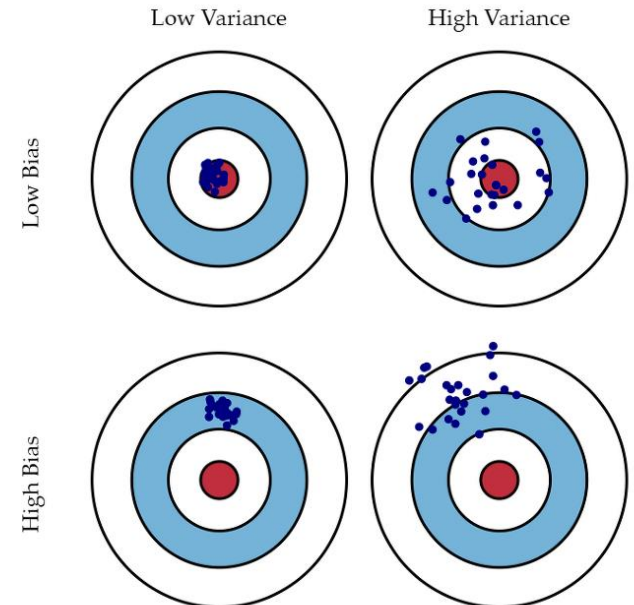
$$\begin{aligned}\mathbb{E} \left[ \left( f(\mathbf{x}) - \hat{f}(\mathbf{x}) \right)^2 \right] &= \mathbb{E} \left[ \left( f(\mathbf{x}) - \bar{f}(\mathbf{x}) + \bar{f}(\mathbf{x}) - \hat{f}(\mathbf{x}) \right)^2 \right] \\&= \mathbb{E} \left[ \left( f(\mathbf{x}) - \bar{f}(\mathbf{x}) \right)^2 - \left( \hat{f}(\mathbf{x}) - \bar{f}(\mathbf{x}) \right)^2 + 2 \left( f(\mathbf{x}) - \bar{f}(\mathbf{x}) \right) \left( \hat{f}(\mathbf{x}) - \bar{f}(\mathbf{x}) \right) \right] \\&= \mathbb{E} \left[ \left( f(\mathbf{x}) - \bar{f}(\mathbf{x}) \right)^2 \right] - \mathbb{E} \left[ \left( \hat{f}(\mathbf{x}) - \bar{f}(\mathbf{x}) \right)^2 \right] + 2 * \mathbb{E} \left[ f(\mathbf{x}) - \bar{f}(\mathbf{x}) \right] * \mathbb{E} \left[ \hat{f}(\mathbf{x}) - \bar{f}(\mathbf{x}) \right] = 0 \\&= \mathbb{E} \left[ \left( f(\mathbf{x}) - \bar{f}(\mathbf{x}) \right)^2 \right] - \mathbb{E} \left[ \left( \hat{f}(\mathbf{x}) - \bar{f}(\mathbf{x}) \right)^2 \right] \\&= \underbrace{\mathbb{E} \left[ \left( f(\mathbf{x}) - \bar{f}(\mathbf{x}) \right)^2 \right]}_{\text{Bias}(\hat{f}(\mathbf{x}))} - \underbrace{\mathbb{E} \left[ \left( \hat{f}(\mathbf{x}) - \bar{f}(\mathbf{x}) \right)^2 \right]}_{\text{Variance}(\hat{f}(\mathbf{x}))}\end{aligned}$$

$$\begin{aligned}\mathbb{E} \left[ \hat{f}(\mathbf{x}) - \bar{f}(\mathbf{x}) \right] &= \mathbb{E} \left[ \hat{f}(\mathbf{x}) \right] - \mathbb{E} \left[ \bar{f}(\mathbf{x}) \right] \\&= \mathbb{E} \left[ \hat{f}(\mathbf{x}) \right] - \mathbb{E} \left[ \mathbb{E} \left[ \hat{f}(\mathbf{x}) \right] \right] \\&= \mathbb{E} \left[ \hat{f}(\mathbf{x}) \right] - \mathbb{E} \left[ \hat{f}(\mathbf{x}) \right] \leftarrow * \mathbb{E} \left[ \hat{f}(\mathbf{x}) \right] \text{는 상수이기 때문에 기댓값을 또 취해도 똑같음} \\&= \mathbb{E} \left[ \hat{f}(\mathbf{x}) \right] - \mathbb{E} \left[ \hat{f}(\mathbf{x}) \right] = 0\end{aligned}$$

# Unit 01 | Introduction

## Bias-Variance Decomposition (참고)

- ✓ **Bias:** difference between predicted value and expected value
  - Low Bias: Accurately estimate the function
  - High Bias: Imply a poor match
- ✓ **Variance:** when the model takes into account the fluctuations in the data
  - Low Variance: Estimated function doesn't change much
  - High Bias: Imply a weak match
- ❖ High Bias + Low Variance: Logistic Regression, LDA, KNN
- ❖ Low Bias + High Variance: ANN, SVM, DT
- ❖ **Low Bias + Low Variance: Best Model!!**, Except 과적합



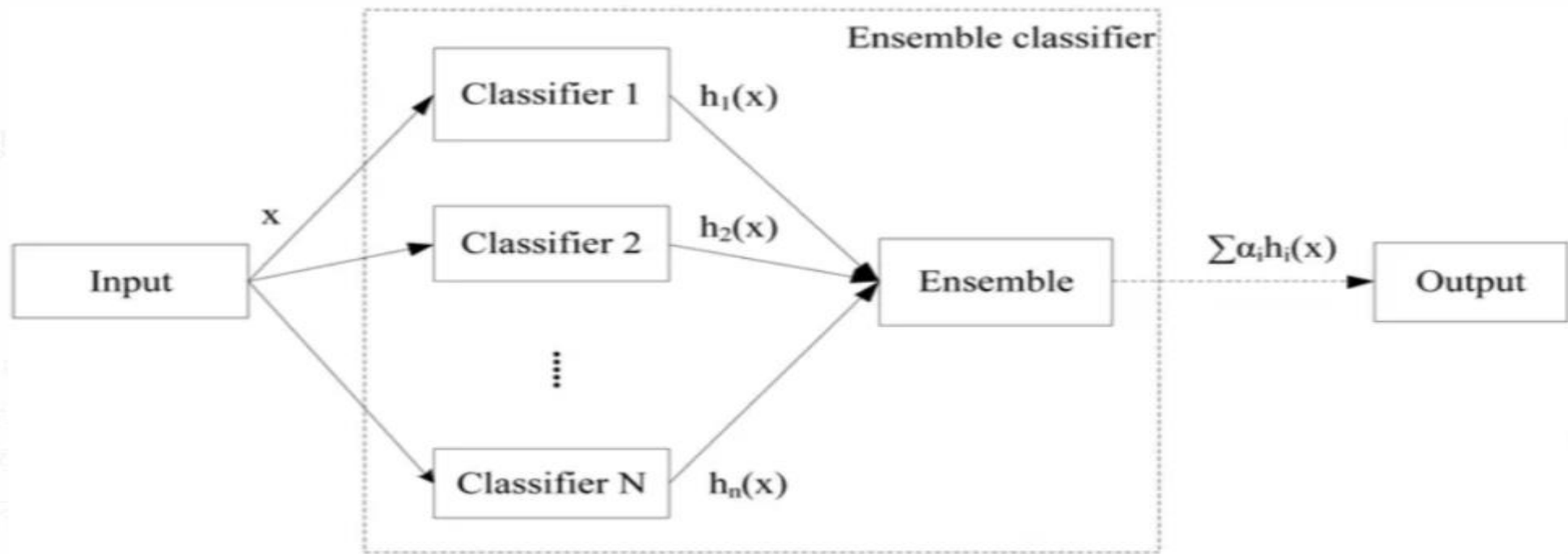
Unit 02 | Ensemble

## Unit 02 - Ensemble

## Unit 02 | Ensemble

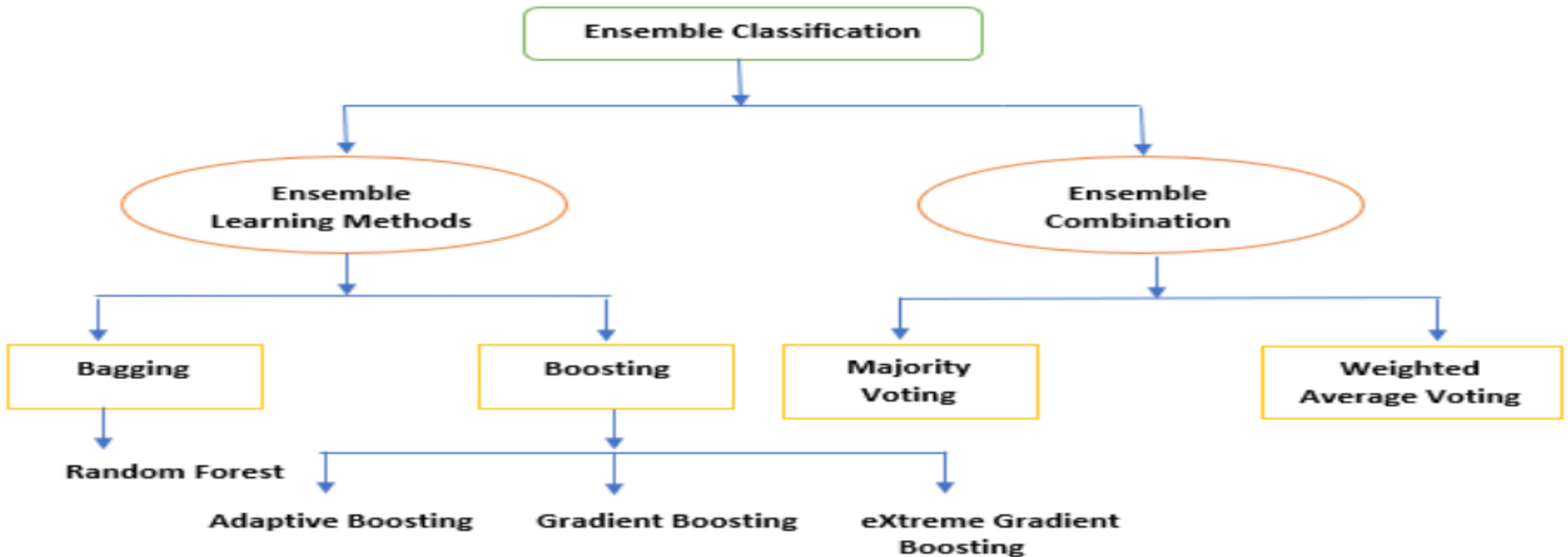
### Ensemble

- Regression -> average
- Classification -> Majority vote



## Unit 02 | Ensemble

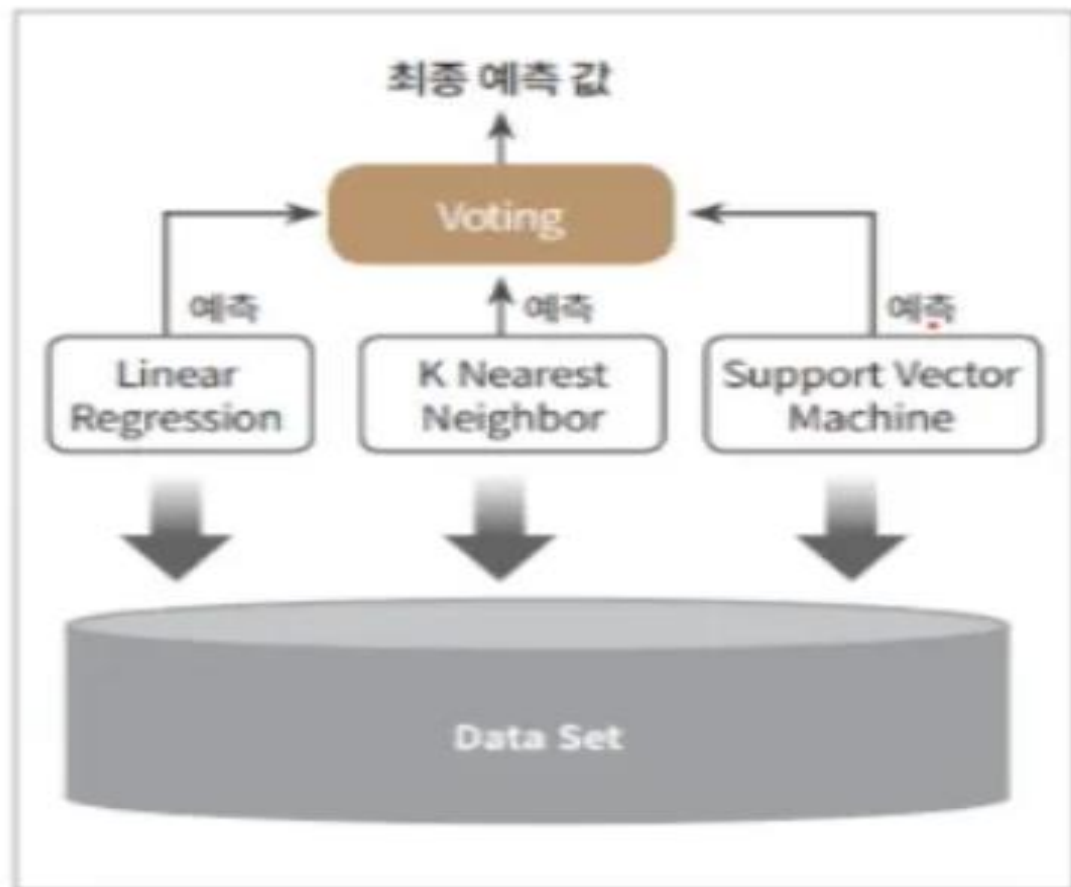
앙상블 학습을 통한 분류는 여러 개의 분류기(Classifier)를 생성하고 그 예측을 결합함으로써 더 정확한 최종예측을 도출 하는 기법이다.



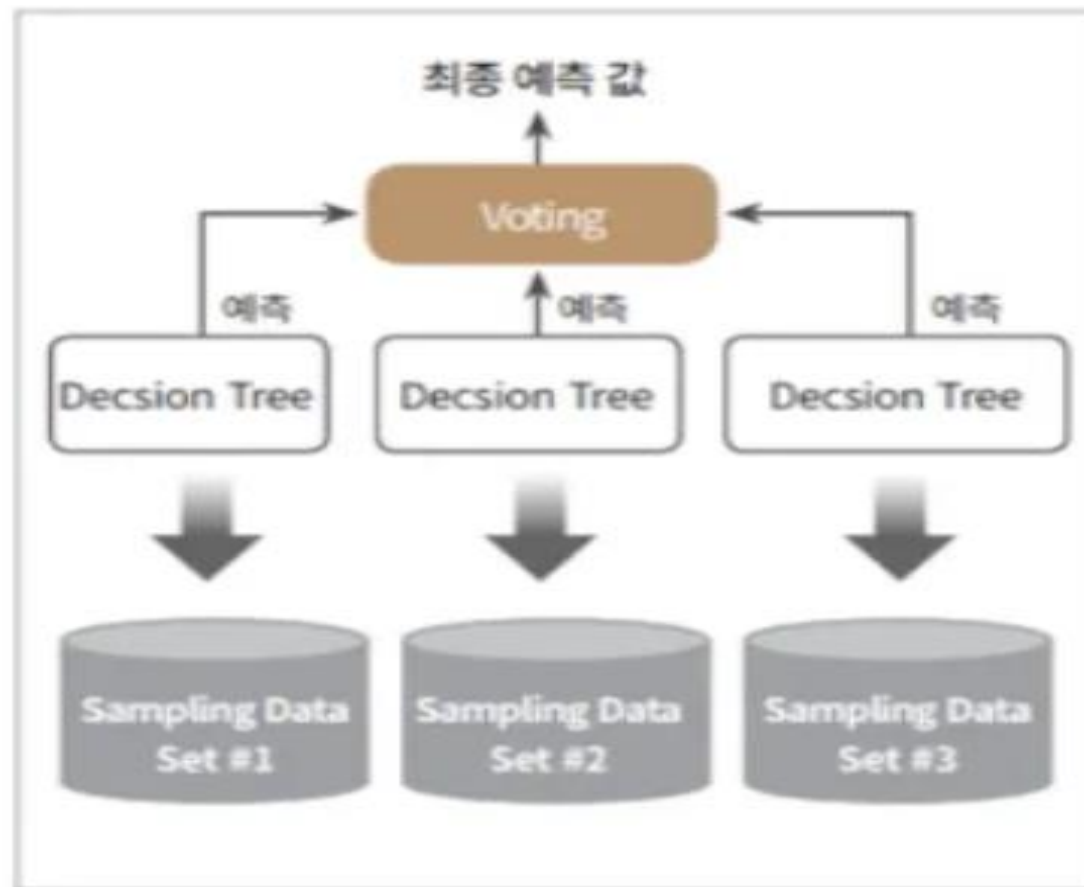
Unit 03 | Voting

## Unit 03 - Voting

## Unit 03 | Voting



Voting 방식



Bagging 방식



## Unit 03 | Voting

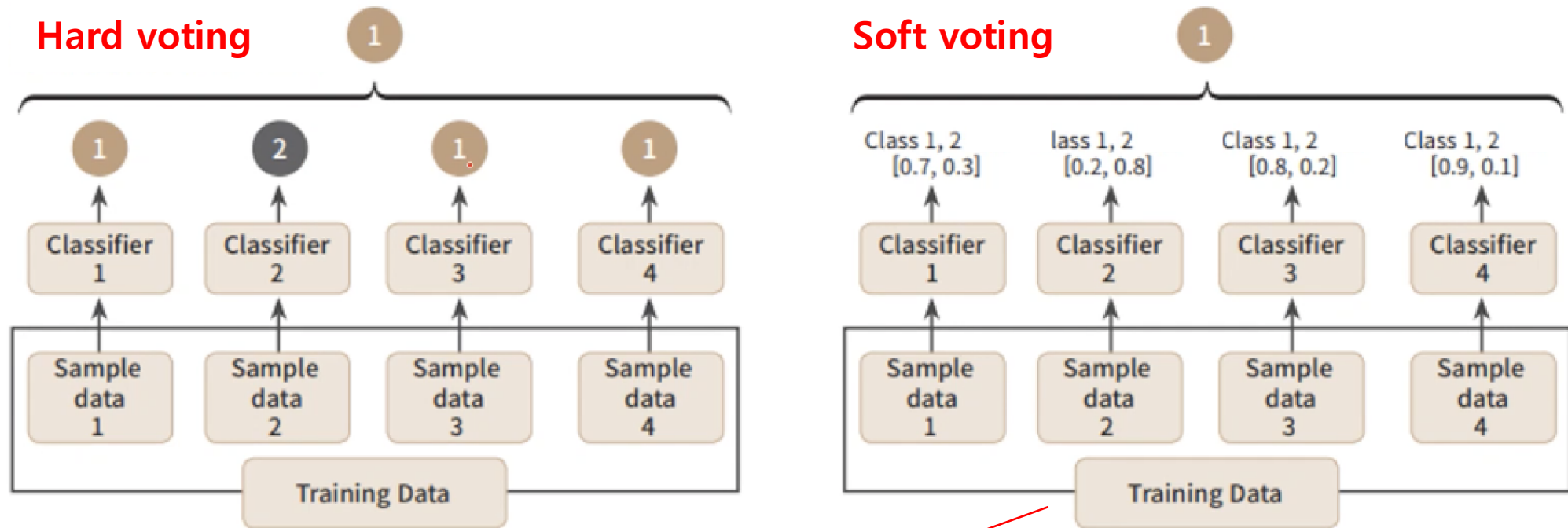
### Voting

- ✓ **Hard voting:** out of multiple outputs produced by the classifiers, the majority output is chosen to be the final result of the model
- ✓ **Soft voting:** Sums the predicted probabilities for class labels and returns the final classification with the largest sum probability.



## Unit 03 | Voting

### Voting



$$p(\text{class} = 1|X) = \frac{0.7+0.2+0.8+0.9}{4} = 0.65, \quad p(\text{class} = 2|X) = \frac{0.3+0.8+0.2+0.1}{4} = 0.35$$

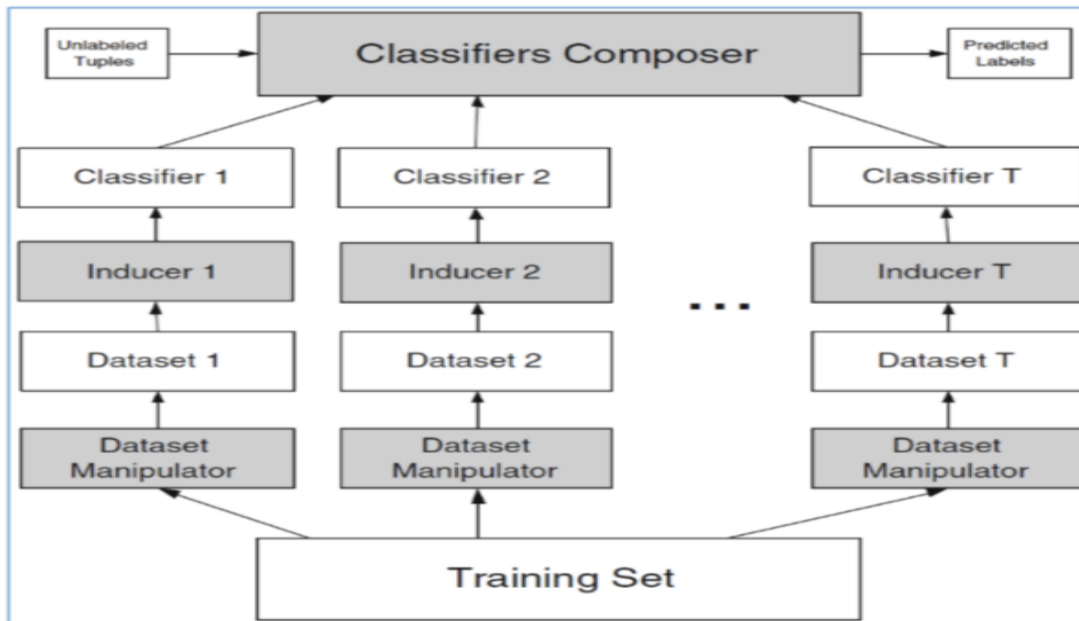
## Unit 04 - Bagging

## Unit 04 | Bagging

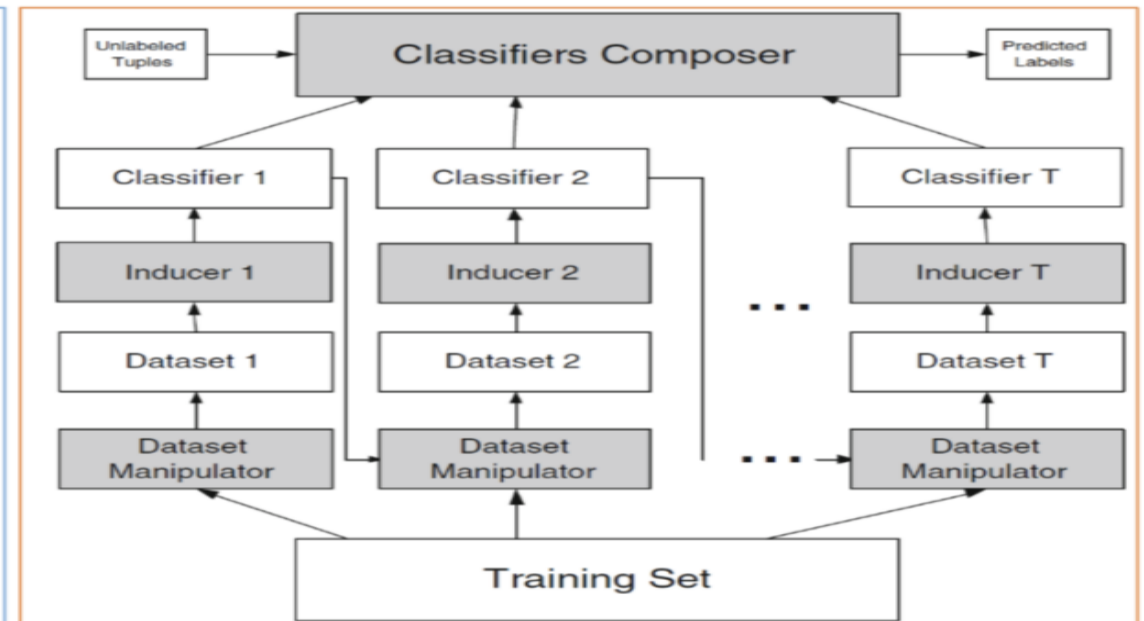
Why do we learn Ensemble?

- In order to reduce variance, we should use Ensemble model based on '**Bagging**' strategy
- In order to reduce bias, we should use Ensemble model based on '**Boosting**' strategy

Independent instance selection

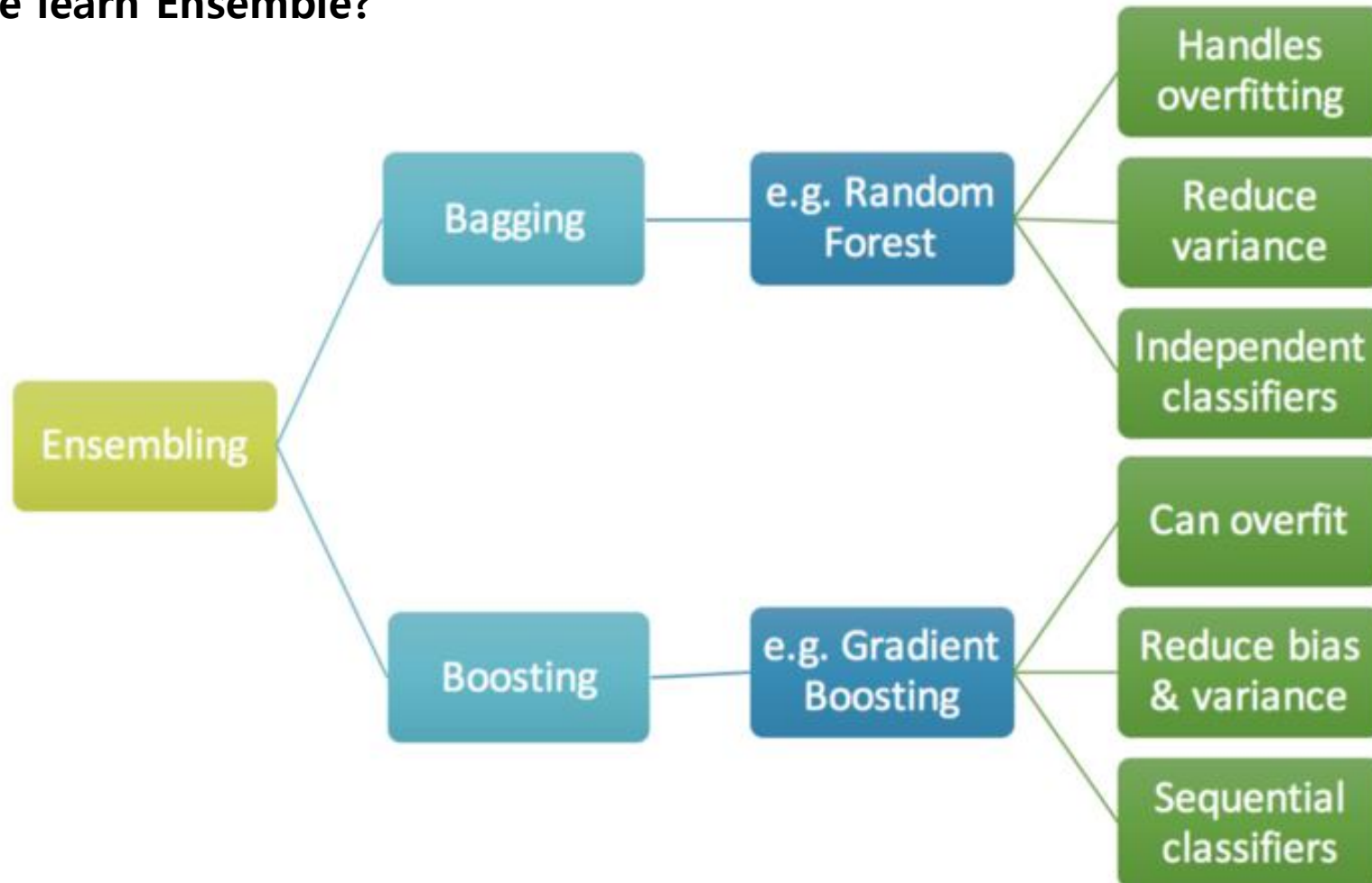


Model guided instance selection



## Unit 04 | Bagging

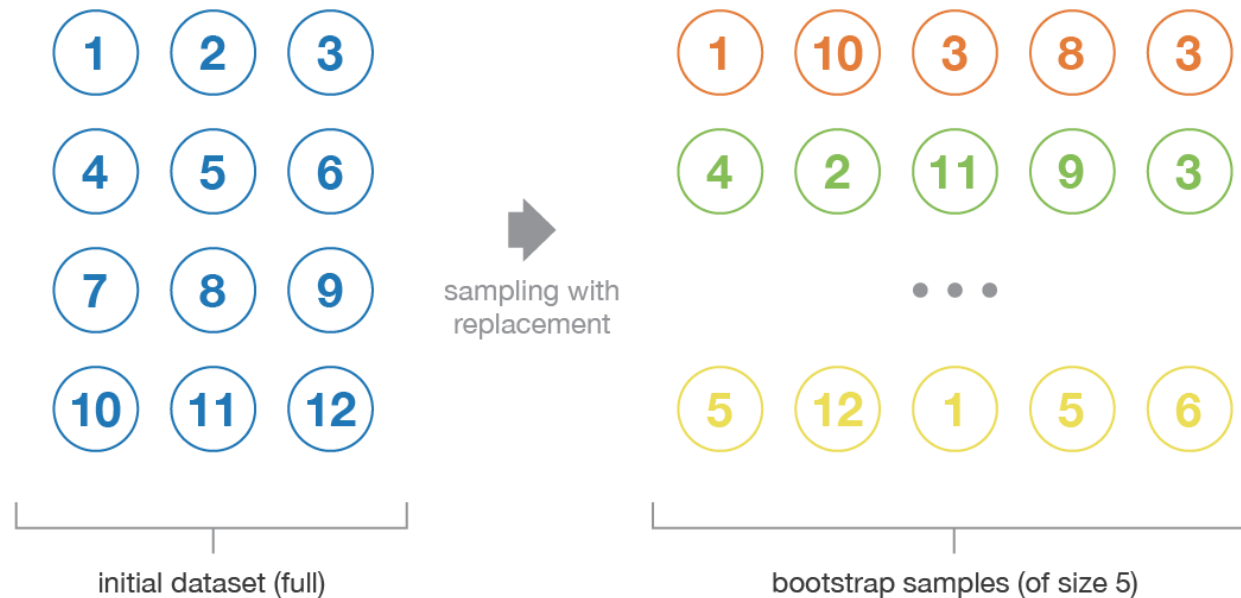
Why do we learn Ensemble?



## Unit 04 | Bagging

### Bagging (Bootstrap Aggregating)

- The objective is to create several subsets of data from training **sample** chosen **randomly** with **replacement** in order to **reduce the variance**.



$$y = f(x) + \epsilon$$

## Unit 04 | Bagging

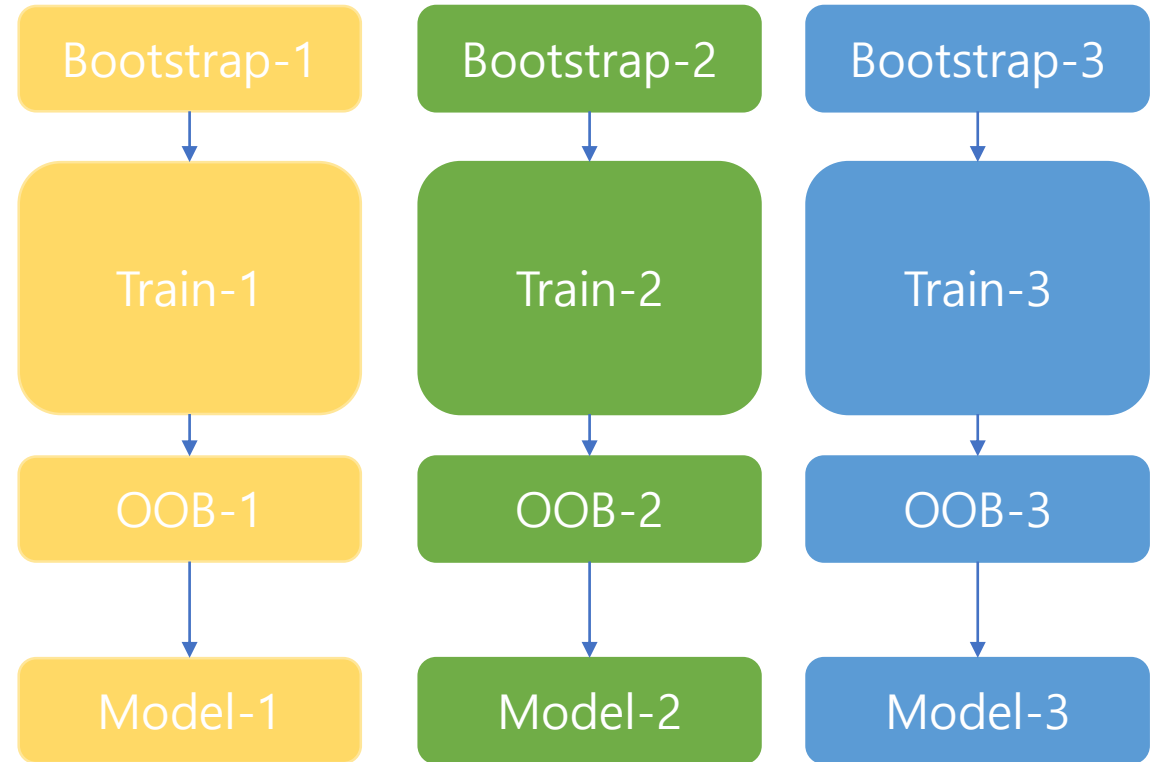
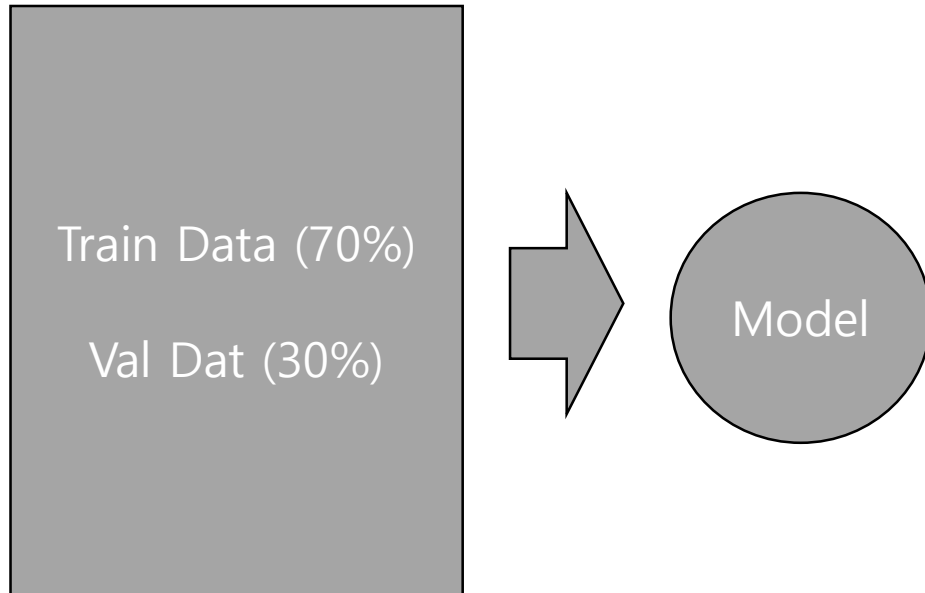
### Bootstrap



OOB ERROR : Bootstrap에 포함되지 않은 데이터를 예측값과 비교해서 나온 값의 차이

## Unit 04 | Bagging

### Bagging (Bootstrap Aggregating)





## Unit 04 | Bagging

### Bagging (Bootstrap Aggregating)

- Bootstrap set의 크기가  $N$ 이라고 할 때 한 번의 추출과정에서 어떤 한 샘플이 추출되지 않을 확률 =  $\frac{N-1}{N}$
- $N$ 회 복원추출 진행했을 때 그 샘플이 추출되지 않았을 확률 =  $\left(1 - \frac{1}{N}\right)^N$

$$p = \left(1 - \frac{1}{N}\right)^N \rightarrow \lim_{N \rightarrow \infty} \left(1 - \frac{1}{N}\right)^N \rightarrow e^{-1} = 0.368 = 36.8\%$$

Out Of Bag data (OOB)

$$1 - p = 1 - 0.368 = 63.2\%$$

Sampled more than once in bootstrap

## Unit 04 | Bagging

### Random Forest

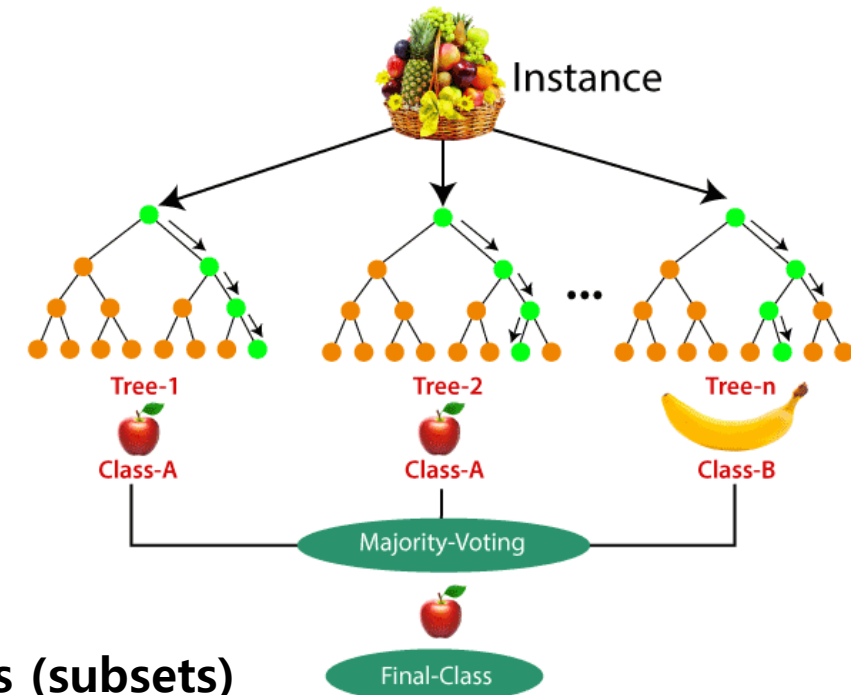
- A specialized bagging for DT (base learner)

- 1) Based on Bagging Ensemble
- 2) Randomly choose variables

Randomly select 'm' variables

- Random Forest Algorithm Procedures

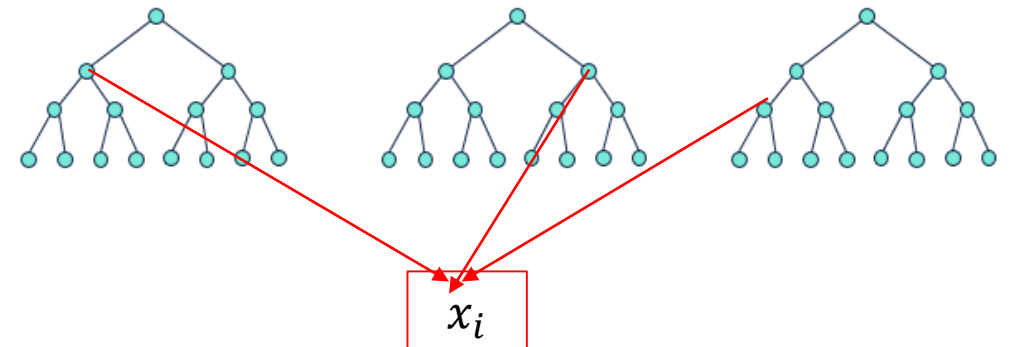
1. Select random 'K' data points from the training set
2. Build the Decision Trees associated with the selected data points (subsets)
3. Choose the number of 'N' for Decision Trees that you want to build
4. Repeat Step1 and Step2
5. For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes



## Unit 04 | Bagging

### Random Forest 특징

- Able to use on both classification and regression tasks
- Handle 'Missing Values' well
- Each tree (base learner) in random forest may overfit the data because 'pruning' is not conducted
- Prevents overfitting problem
- Variable Importance --→ **Not suggest which variables to select**
- Difficult to interpret the results (Black-Box Model)
- Too slow when the number of data size is large



## Unit 04 | Bagging

### Random Forest의 중요 Parameters

- Decision Tree의 개수
  - Forest가 작으면 테스트 시간 감소 but 일반화 능력 하락
  - Forest가 크다면 테스트 시간 증가 but 일반화 능력 상승

### Tree의 최대 깊이

- Root Node부터 Terminal Node까지 최대 몇 개의 Node를 거칠 것인지 결정

### - Node 분할 함수 (split function)

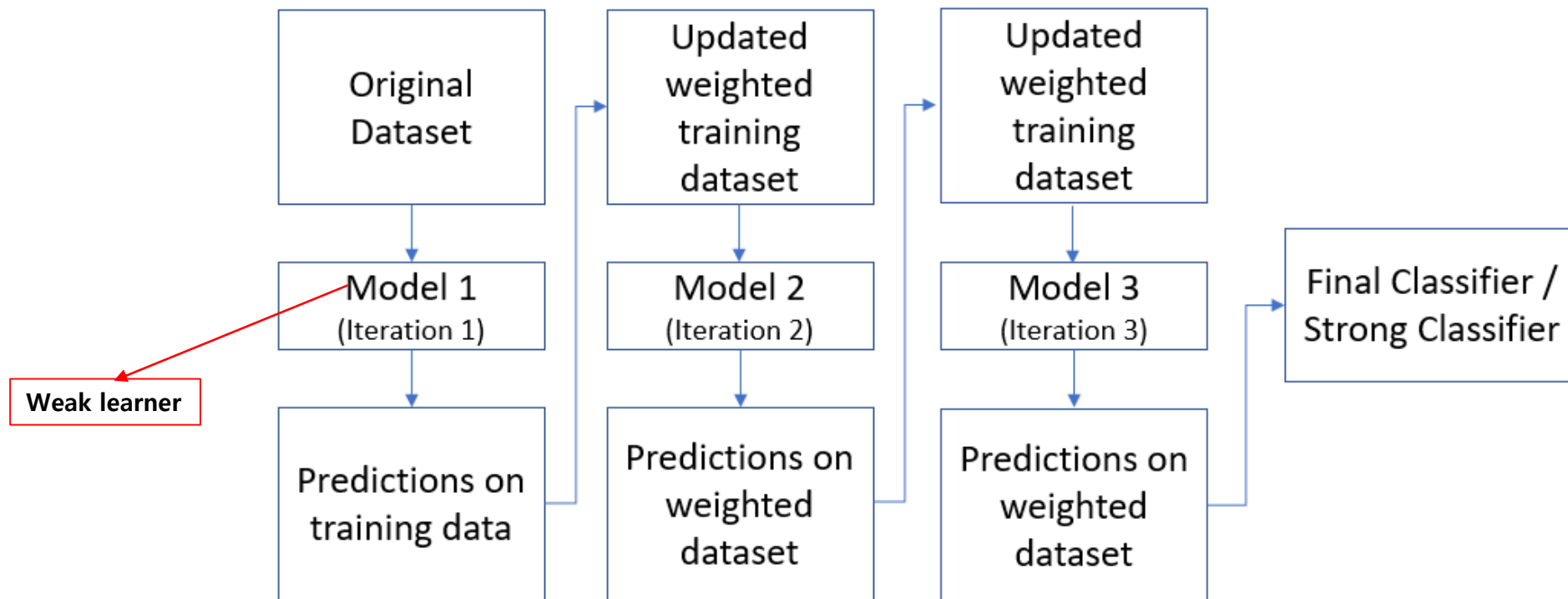
- 전체 데이터 셋에서 몇 개의 특징(feature)선택
- 큰 중요도 점수를 가지는 변수를 높은 순위로 올리는 방식을 선택하는 것이 일반적

## Unit 05 - Boosting

## Unit 05 | Boosting

**Boosting:** An iterative procedure to adaptively change distribution of training data by focusing more on previously mis-classified records.

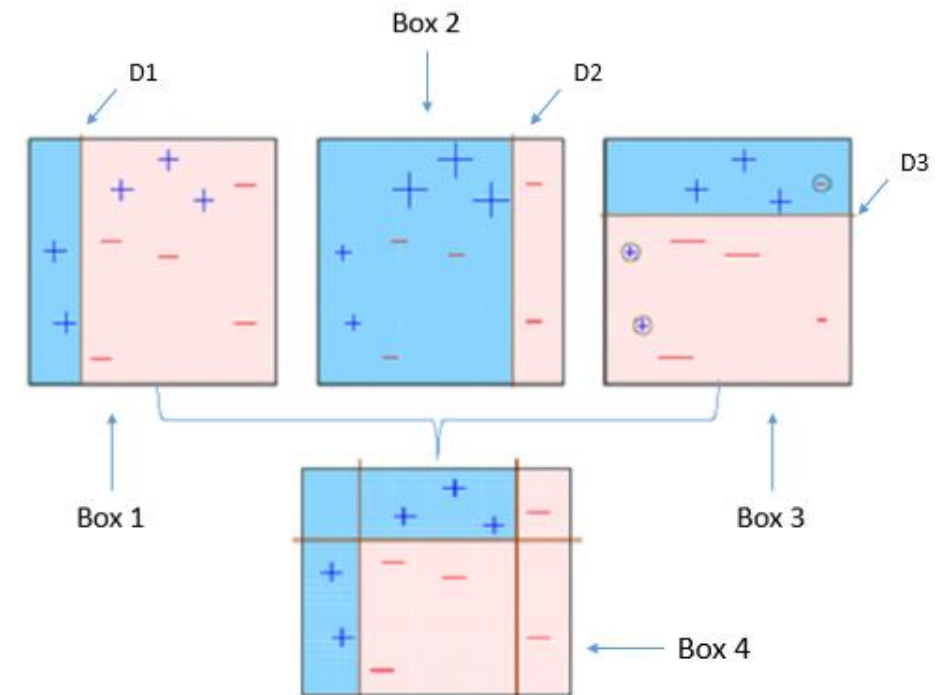
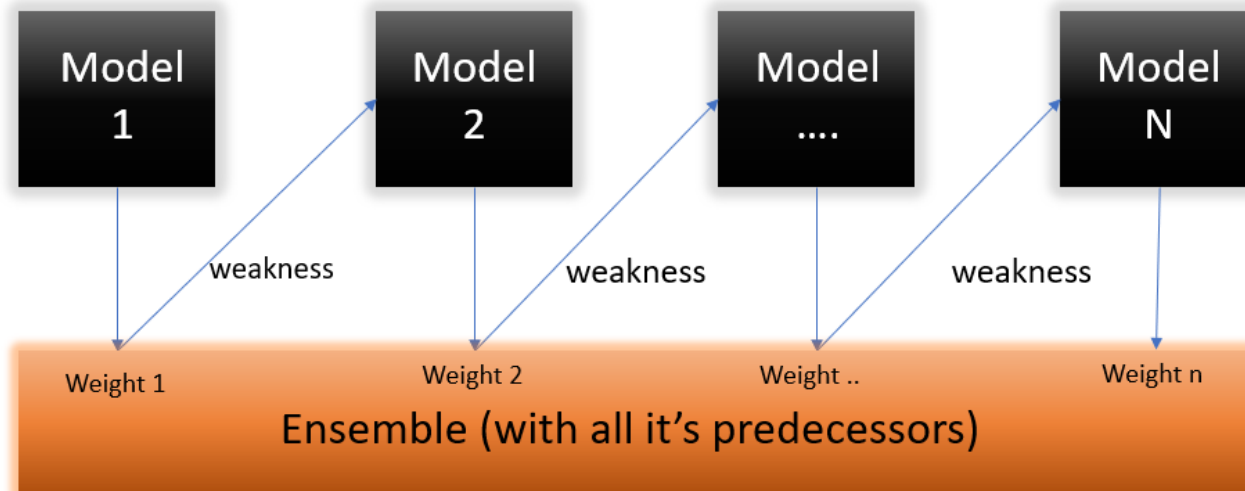
- Sequential learning process unlike 'Bagging' (parallel learning)



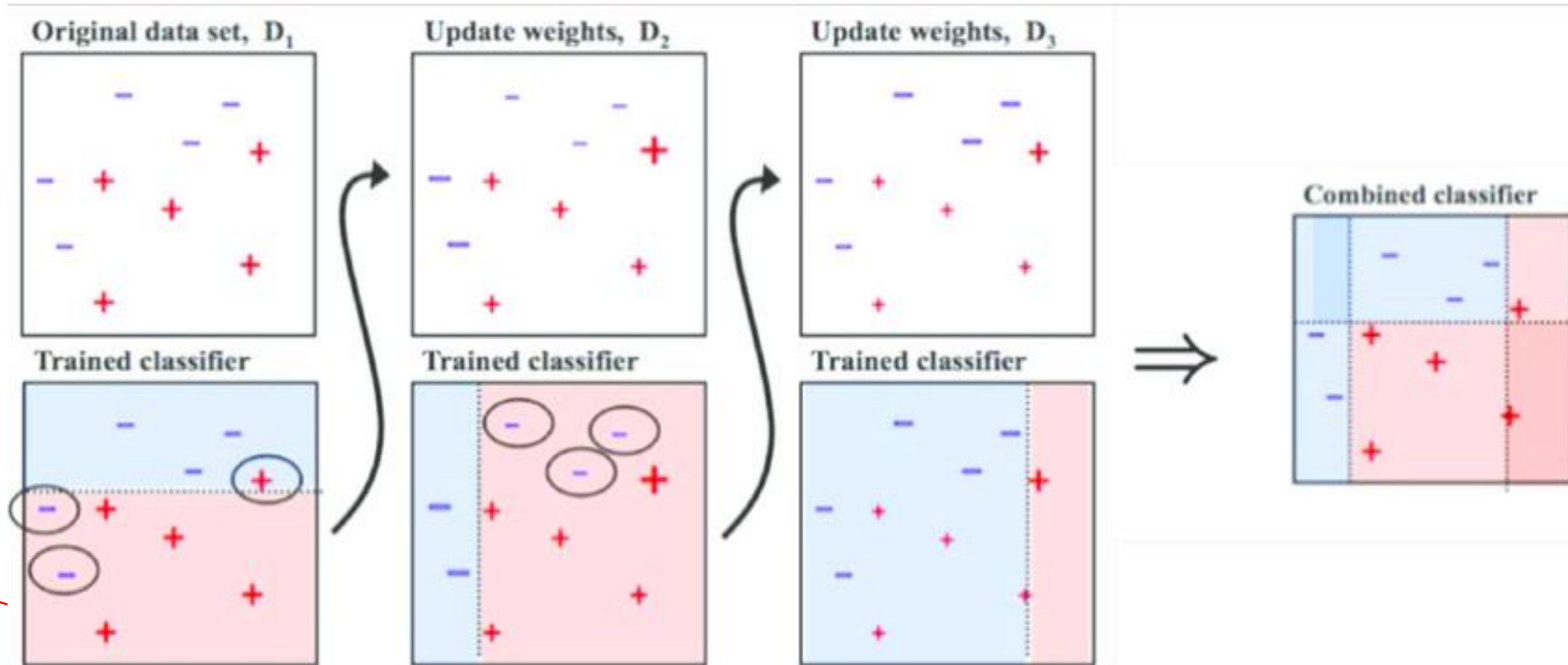
## Unit 05 | Boosting

### AdaBoost (Adaptive Boosting)

- Weak learner, performing only slightly better than random guessing, could be **boosted** in to arbitrarily accurate **strong learner**
- [https://www.youtube.com/watch?v=HZg8\\_wZPZGU](https://www.youtube.com/watch?v=HZg8_wZPZGU)



## Unit 05 | Boosting





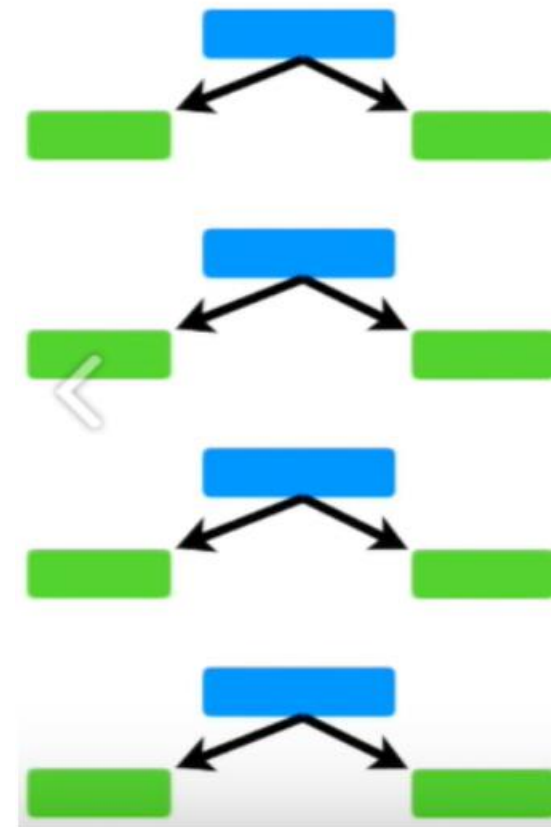
## Unit 05 | Boosting

- AdaBoost
- Gradient Boosting Machine (GBM)
- XGBoost
- LightGBM

Stump (약분류기)

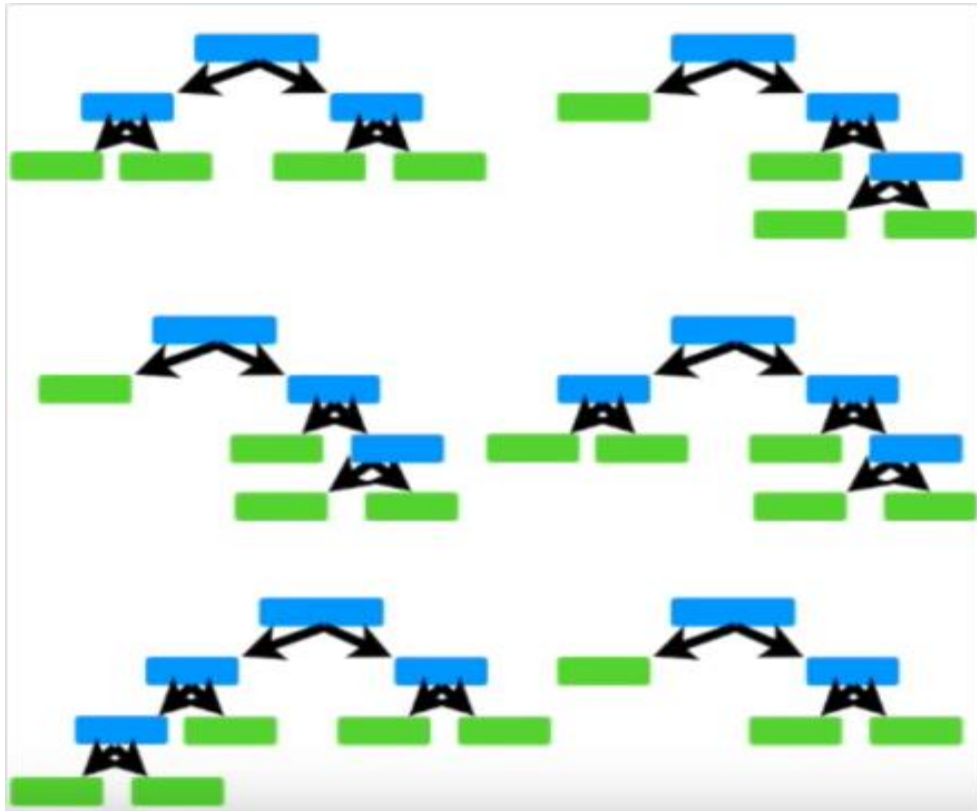


Forest of Stumps

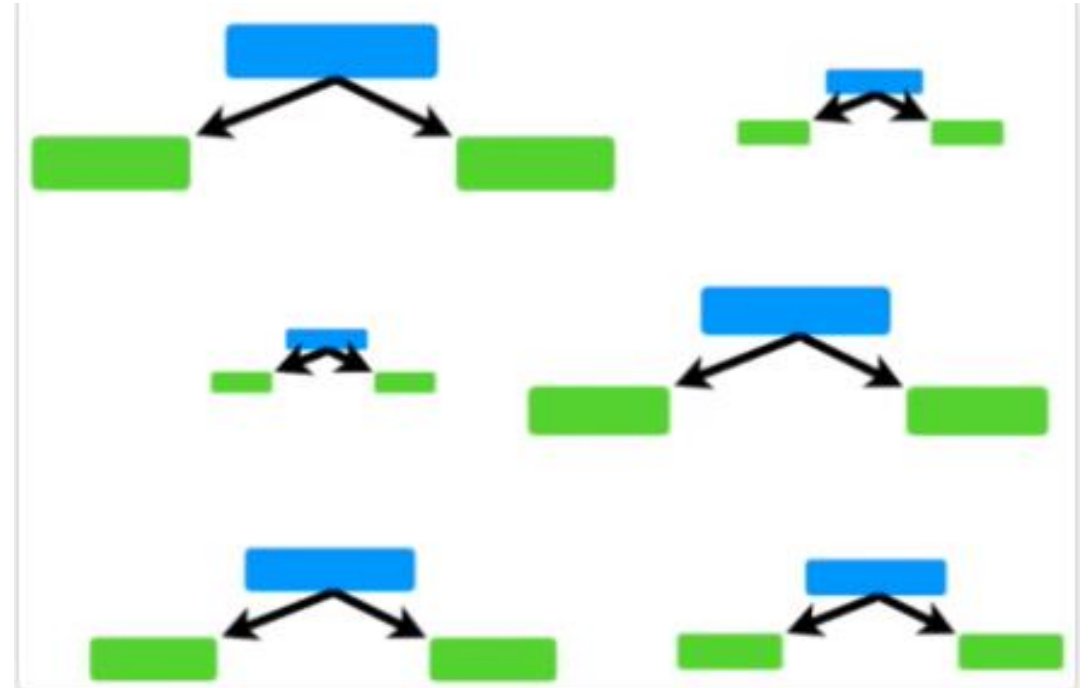


## Unit 05 | Boosting

Random Forest



AdaBoost



## Unit 05 | Boosting

### AdaBoost (Adaptive Boosting)

- ✓ 약한 학습기, stump 형태
- ✓ 오답에 대해서 가중치를 높게 부여
- ✓ 각 stump의 error는 다음 stump의 결과에 영향을 줌

---

#### Algorithm 2 Adaboost

---

**Input:** Required ensemble size  $T$

**Input:** Training set  $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ , where  $y_i \in \{-1, +1\}$

Define a uniform distribution  $D_1(i)$  over elements of  $S$ .

**for**  $t = 1$  to  $T$  **do**

    Train a model  $h_t$  using distribution  $D_t$ .

    Calculate  $\epsilon_t = P_{D_t}(h_t(x) \neq y)$

    If  $\epsilon_t \geq 0.5$  break

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

    Update  $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

    where  $Z_t$  is a normalization factor so that  $D_{t+1}$  is a valid distribution.

**end for**

For a new testing point  $(x', y')$ ,

$$H(x') = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x') \right)$$

---

## Unit 05 | Boosting

### AdaBoost (Adaptive Boosting)

---

#### Algorithm 2 Adaboost

---

**Input:** Required ensemble size  $T$

**Input:** Training set  $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ , where  $y_i \in \{-1, +1\}$

Define a uniform distribution  $D_1(i)$  over elements of  $S$ .


**for**  $t = 1$  to  $T$  **do**

Train a model  $h_t$  using distribution  $D_t$ .  비복원 추출을 기본 하지만 복원도 가능

Calculate  $\epsilon_t = P_{D_t}(h_t(x) \neq y)$

If  $\epsilon_t \geq 0.5$  break  랜덤 추출은 오차율이 0.5

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

Update  $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$  

where  $Z_t$  is a normalization factor so that  $D_{t+1}$  is a valid distribution.

**end for**

For a new testing point  $(x', y')$ ,

$$H(x') = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x') \right)$$

---

## Unit 05 | Boosting

### AdaBoost (Adaptive Boosting)

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

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$y_i h_t(x_i) = \begin{cases} 1 & \text{if } y_i = h_t(x_i) \\ -1 & \text{if } y_i \neq h_t(x_i) \end{cases}$$

---

#### Algorithm 2 Adaboost

---

**Input:** Required ensemble size  $T$

**Input:** Training set  $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ , where  $y_i \in \{-1, +1\}$   
Define a uniform distribution  $D_1(i)$  over elements of  $S$ .

**for**  $t = 1$  to  $T$  **do**

    Train a model  $h_t$  using distribution  $D_t$ .

    Calculate  $\epsilon_t = P_{D_t}(h_t(x) \neq y)$

    If  $\epsilon_t \geq 0.5$  break

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

    Update  $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

    where  $Z_t$  is a normalization factor so that  $D_{t+1}$  is a valid distribution.

**end for**

For a new testing point  $(x', y')$ ,

$$H(x') = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x') \right)$$

---

## Unit 05 | Boosting

### AdaBoost (Adaptive Boosting)

$\exp(-\alpha_t y_i h_t(x_i))$	성능 좋은 분류기 ( $\alpha_t \gg 0$ )	그저 그런(?) 분류기 ( $\alpha_t \approx 0$ )
$y_i = h_t(x_i)$ [ $y_i h_t(x_i) = 1$ ]	$\approx 0$	$< 1$
$y_i \neq h_t(x_i)$ [ $y_i h_t(x_i) = -1$ ]	$\gg 1$	$> 1$

$$H(x') = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x')\right)$$

---

#### Algorithm 2 Adaboost

---

**Input:** Required ensemble size  $T$

**Input:** Training set  $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ , where  $y_i \in \{-1, +1\}$   
Define a uniform distribution  $D_1(i)$  over elements of  $S$ .

**for**  $t = 1$  to  $T$  **do**

    Train a model  $h_t$  using distribution  $D_t$ .

    Calculate  $\epsilon_t = P_{D_t}(h_t(x) \neq y)$

    If  $\epsilon_t \geq 0.5$  break

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

    Update  $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

    where  $Z_t$  is a normalization factor so that  $D_{t+1}$  is a valid distribution.

**end for**

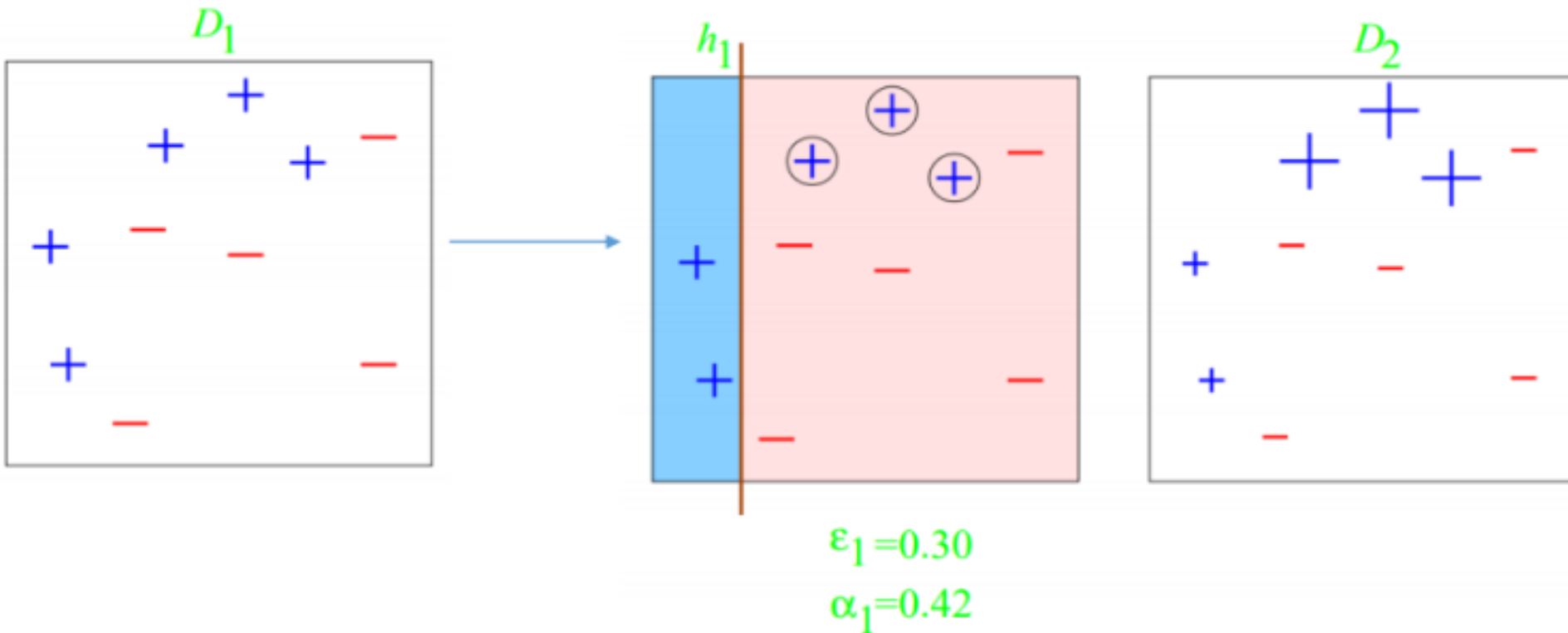
For a new testing point  $(x', y')$ ,

$$H(x') = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x')\right)$$

---

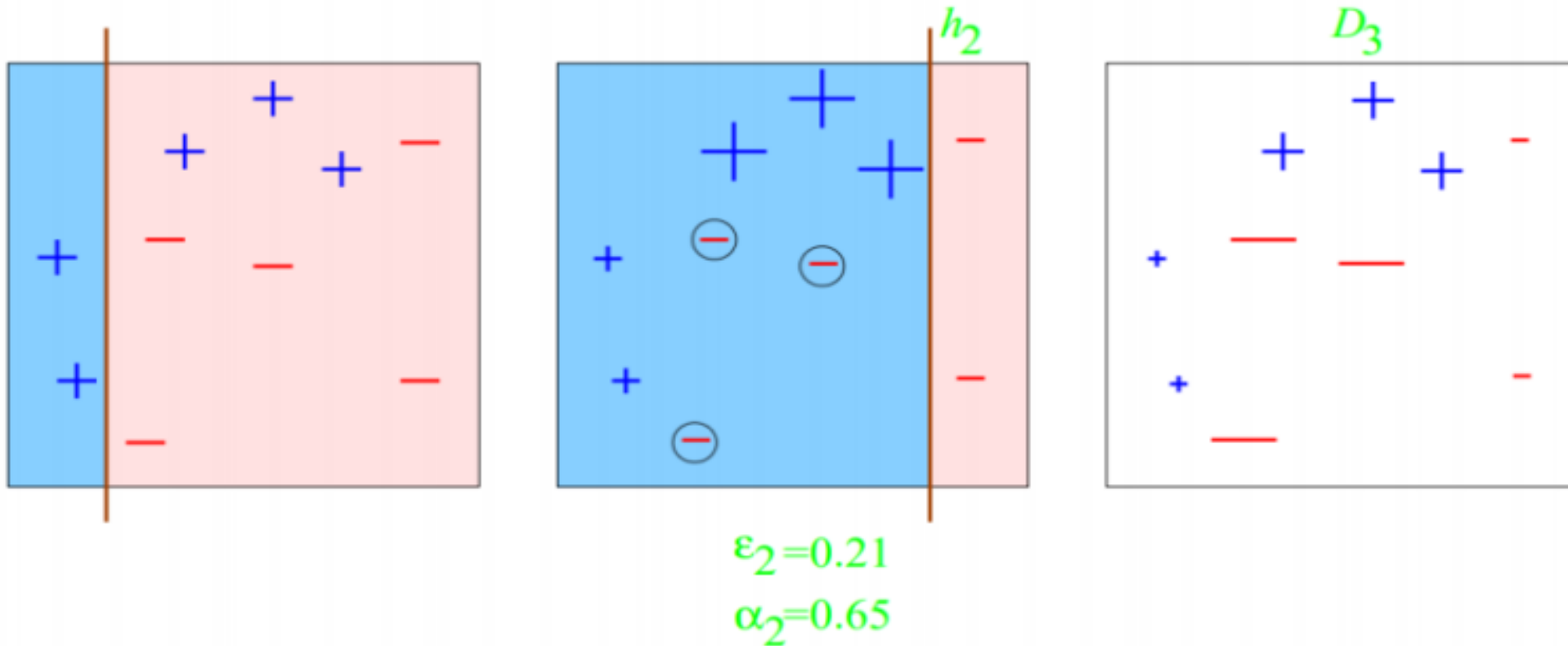
## Unit 05 | Boosting

### AdaBoost (Adaptive Boosting)



## Unit 05 | Boosting

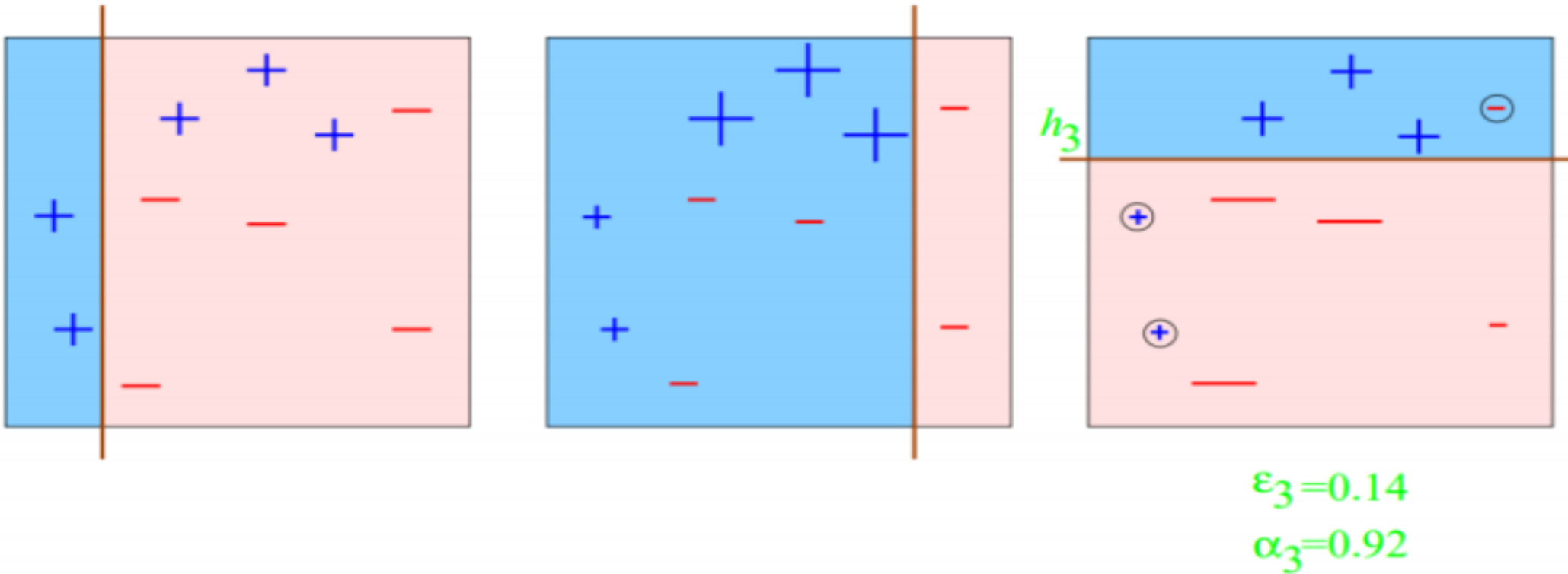
### AdaBoost (Adaptive Boosting)





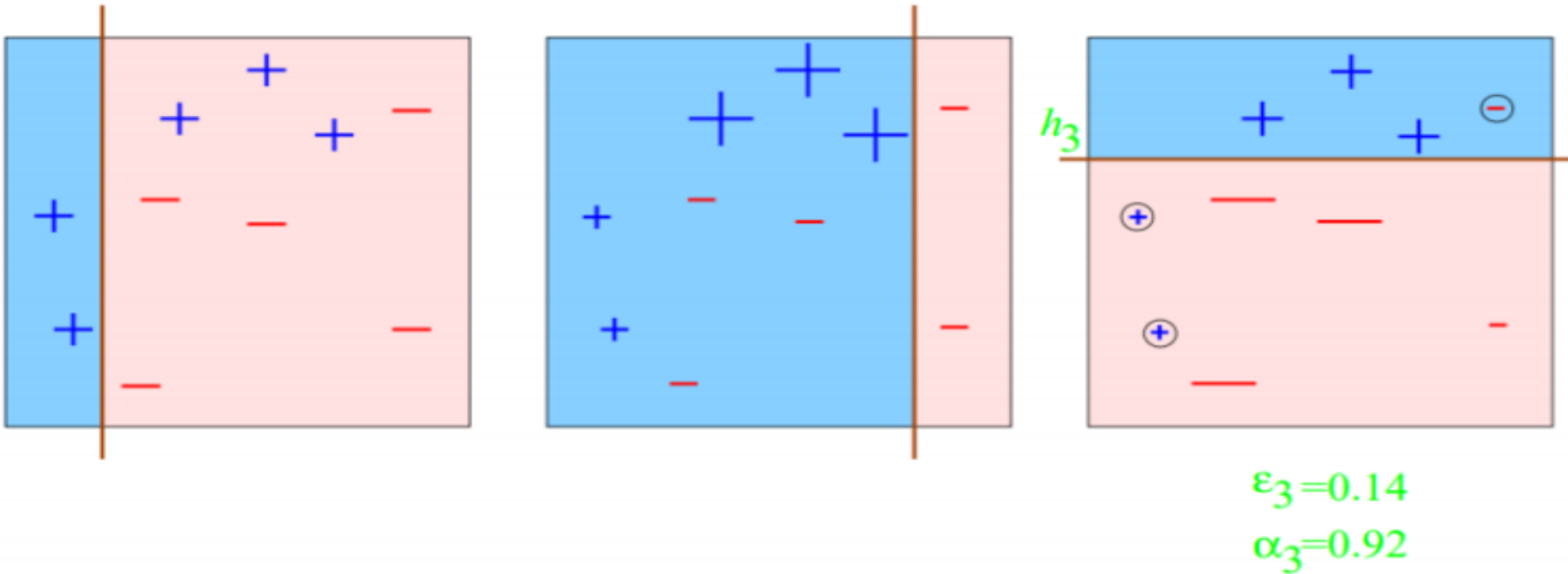
## Unit 05 | Boosting

### AdaBoost (Adaptive Boosting)



## Unit 05 | Boosting

### AdaBoost (Adaptive Boosting)

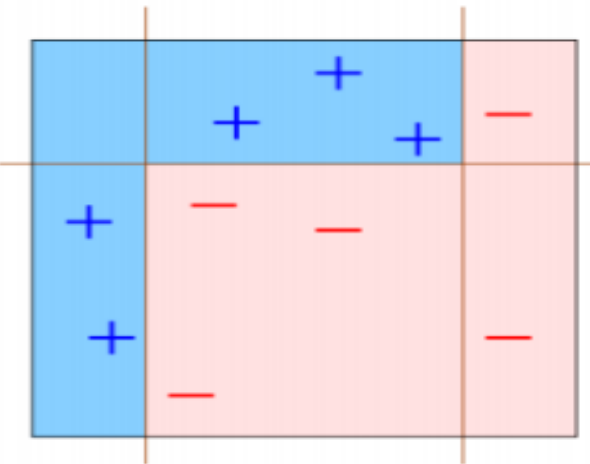


## Unit 05 | Boosting

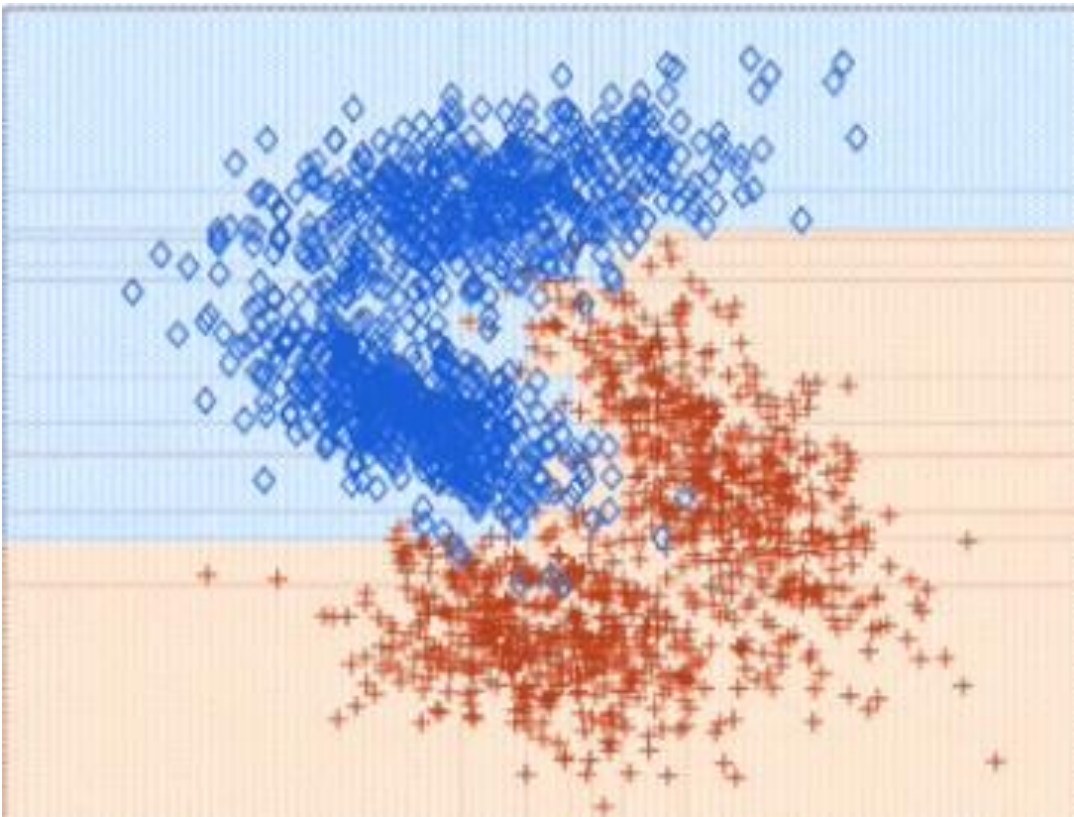
### AdaBoost (Adaptive Boosting)

$$H_{\text{final}} = \text{sign} \left( 0.42 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} \right)$$

=



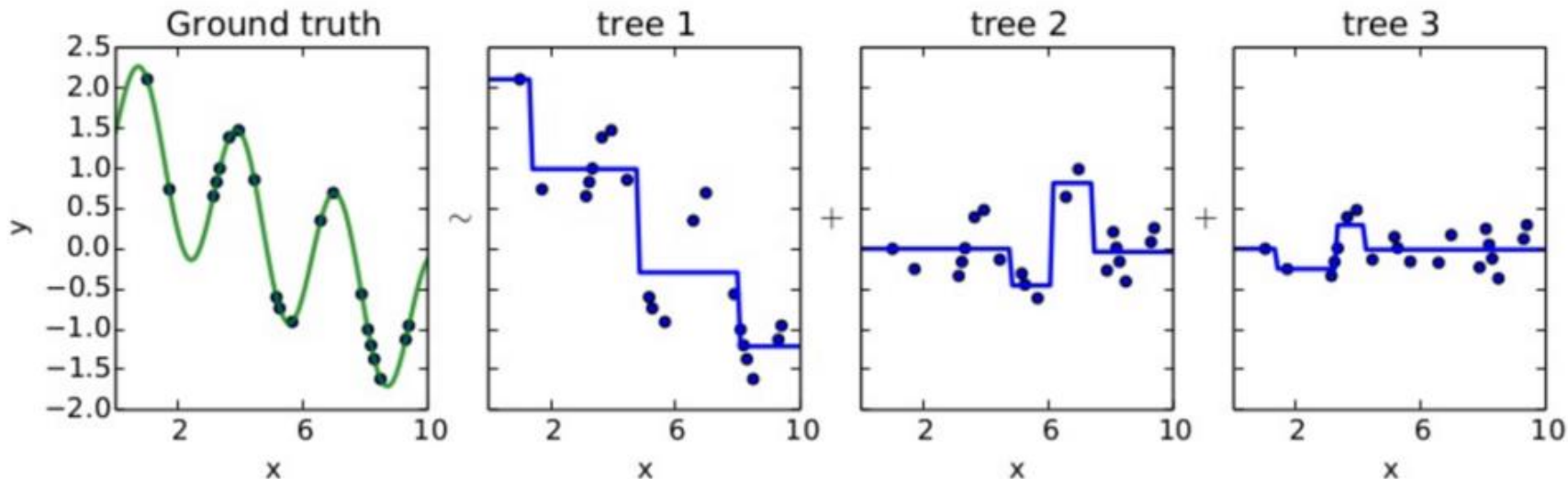
### AdaBoost (Adaptive Boosting)



## Unit 05 | Boosting

### GBM (Gradient Boosting Machine)

- ✓ AdaBoost: update weights on Dataset by sampling
- ✓ GBM: update 'y' not Dataset
- Understand the concept of 'Residual Fitting'



## Unit 05 | Boosting

GBM (Gradient Boosting Machine) (참고)

$$L = MSE = \frac{1}{2} (y_i - f(x_i))^2$$

$$\textit{Gradient} = \frac{\partial L}{\partial f(x_i)} = f(x_i) - y_i$$

$$\textit{Residual} = y_i - f(x_i)$$

$$\textit{Residual} = \textit{Negative Gradient}$$

$$\textit{GBM} = \textit{Gradient Descent} + \textit{Boosting}$$

# Unit 05 | Boosting

## GBM (Gradient Boosting Machine) (참고)

Input: training set  $\{(x_i, y_i)\}_{i=1}^n$ , a differentiable loss function  $L(y, F(x))$ , number of iterations  $M$ .

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma).$$

**New model fitting -> Loss function down!!**

2. For  $m = 1$  to  $M$ :

1. Compute so-called *pseudo-residuals*:

$$r_{im} = - \left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \dots, n.$$

2. Fit a base learner (or weak learner, e.g. tree) closed under scaling  $h_m(x)$  to pseudo-residuals, i.e. train it using the training set  $\{(x_i, r_{im})\}_{i=1}^n$ .

3. Compute multiplier  $\gamma_m$  by solving the following *one-dimensional optimization* problem:

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$$

4. Update the model:

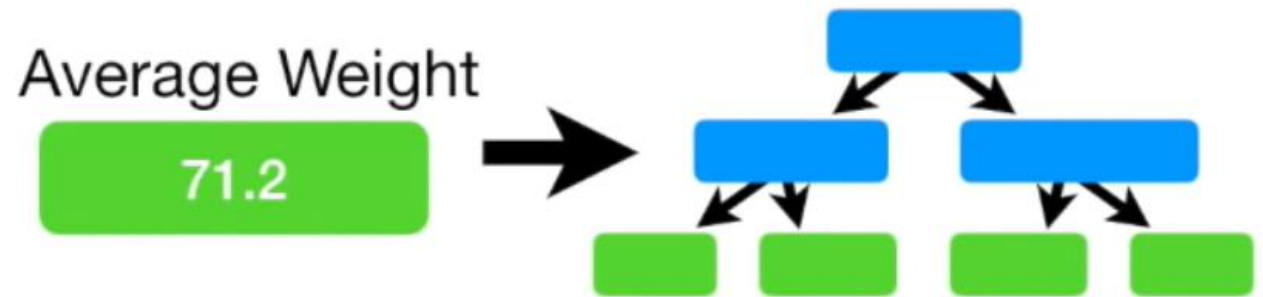
$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output  $F_M(x)$ .

## Unit 05 | Boosting

### GBM (Gradient Boosting Machine)

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

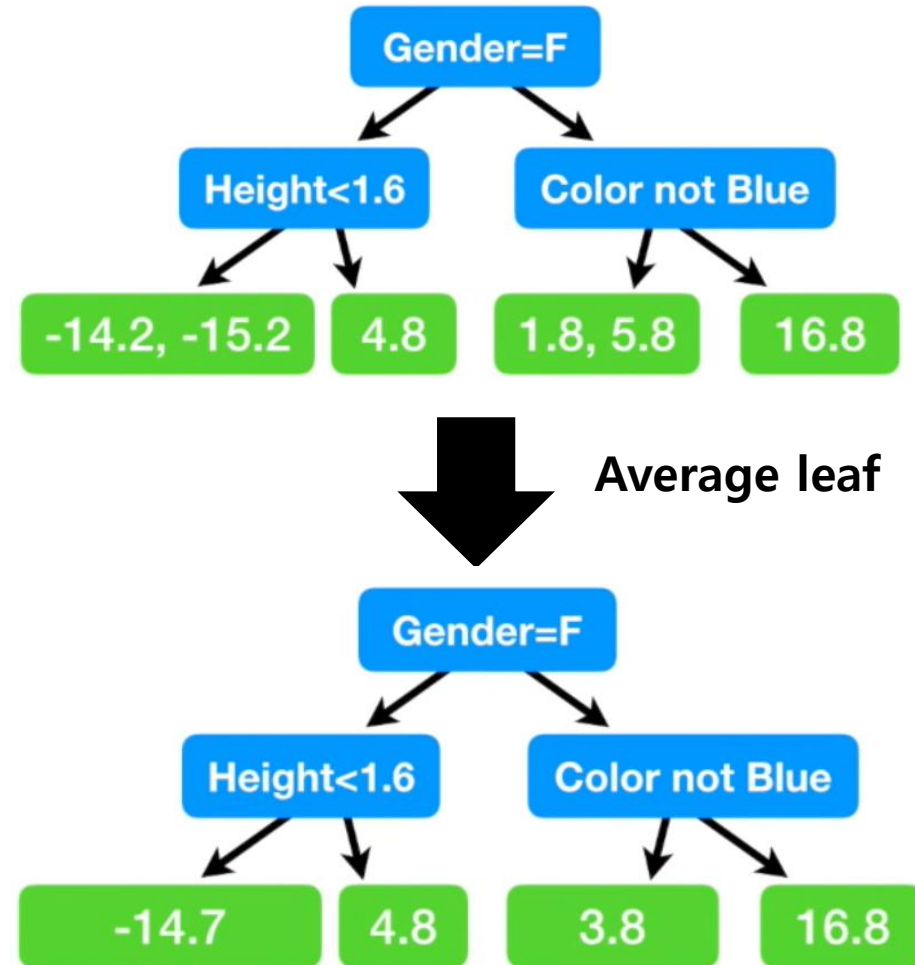




## Unit 05 | Boosting

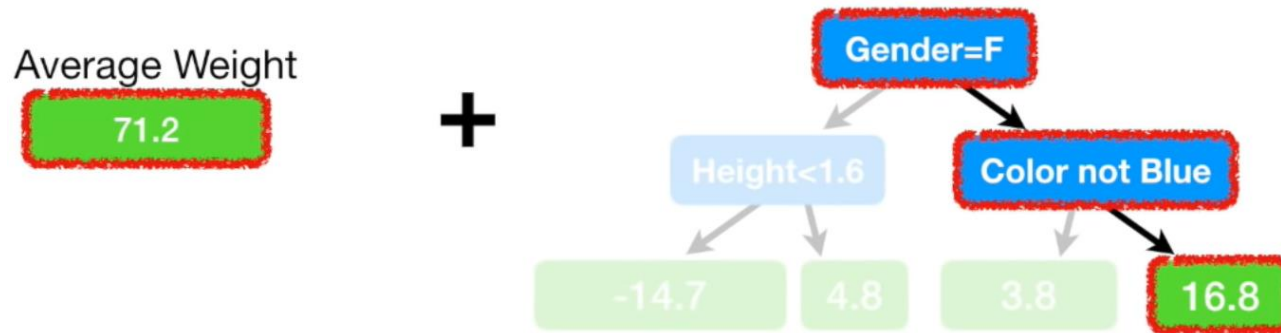
### GBM (Gradient Boosting Machine)

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2



## Unit 05 | Boosting

### GBM (Gradient Boosting Machine)



...so the **Predicted Weight** =  $71.2 + 16.8 = 88$

Learning Rate = 0.1

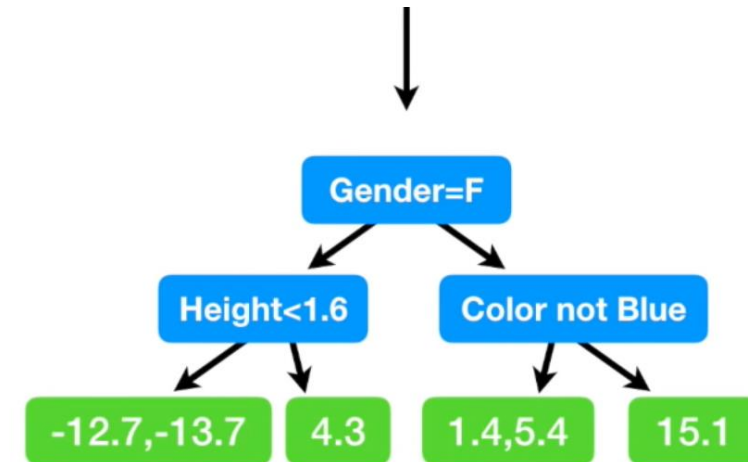


## Unit 05 | Boosting

### GBM (Gradient Boosting Machine)

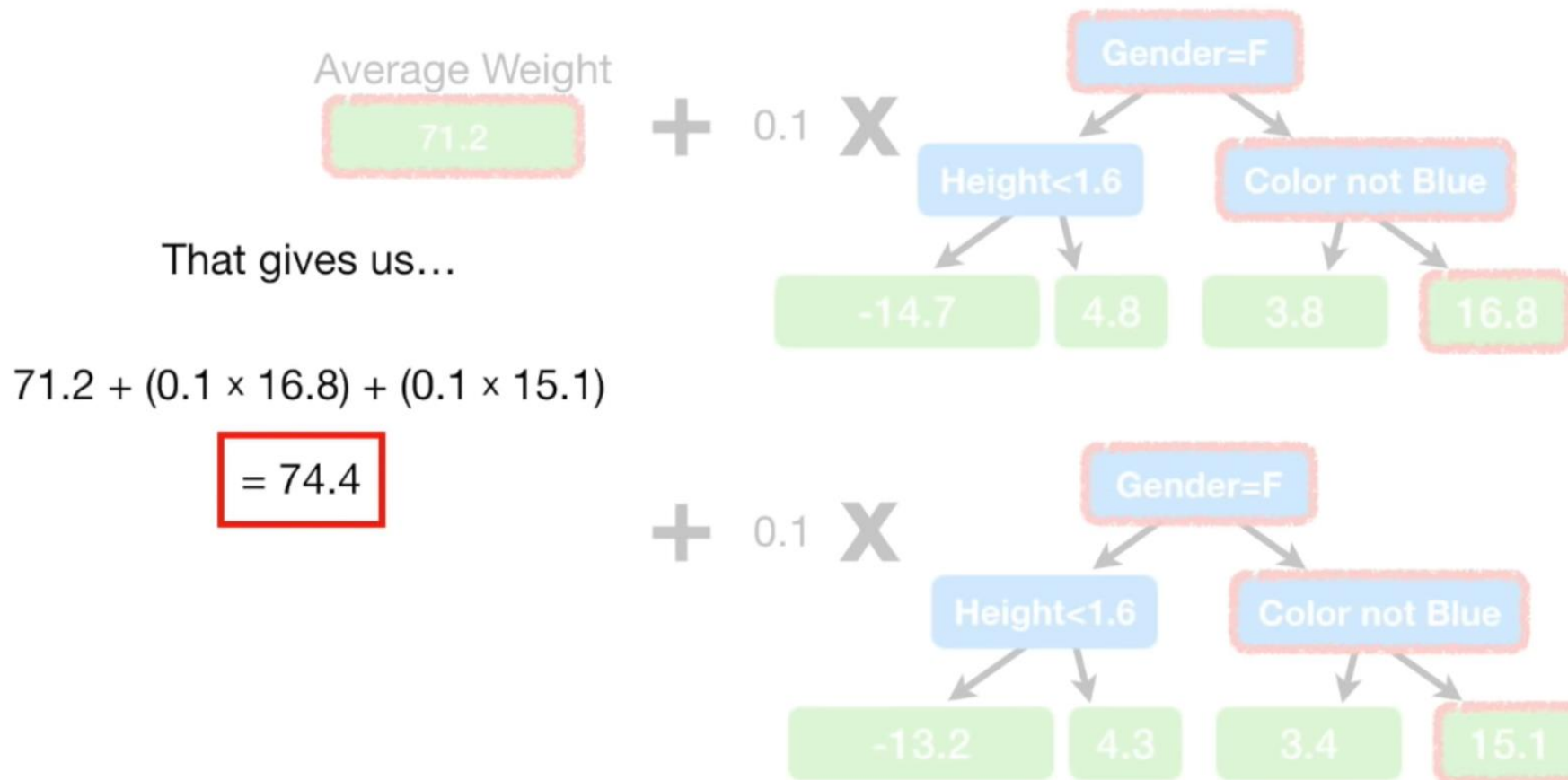
Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	15.1
1.6	Green	Female	76	4.3
1.5	Blue	Female	56	-13.7
1.8	Red	Male	73	1.4
1.5	Green	Male	77	5.4
1.4	Blue	Female	57	-12.7

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	15.1
1.6	Green	Female	76	4.3
1.5	Blue	Female	56	-13.7
1.8	Red	Male	73	1.4
1.5	Green	Male	77	5.4
1.4	Blue	Female	57	-12.7



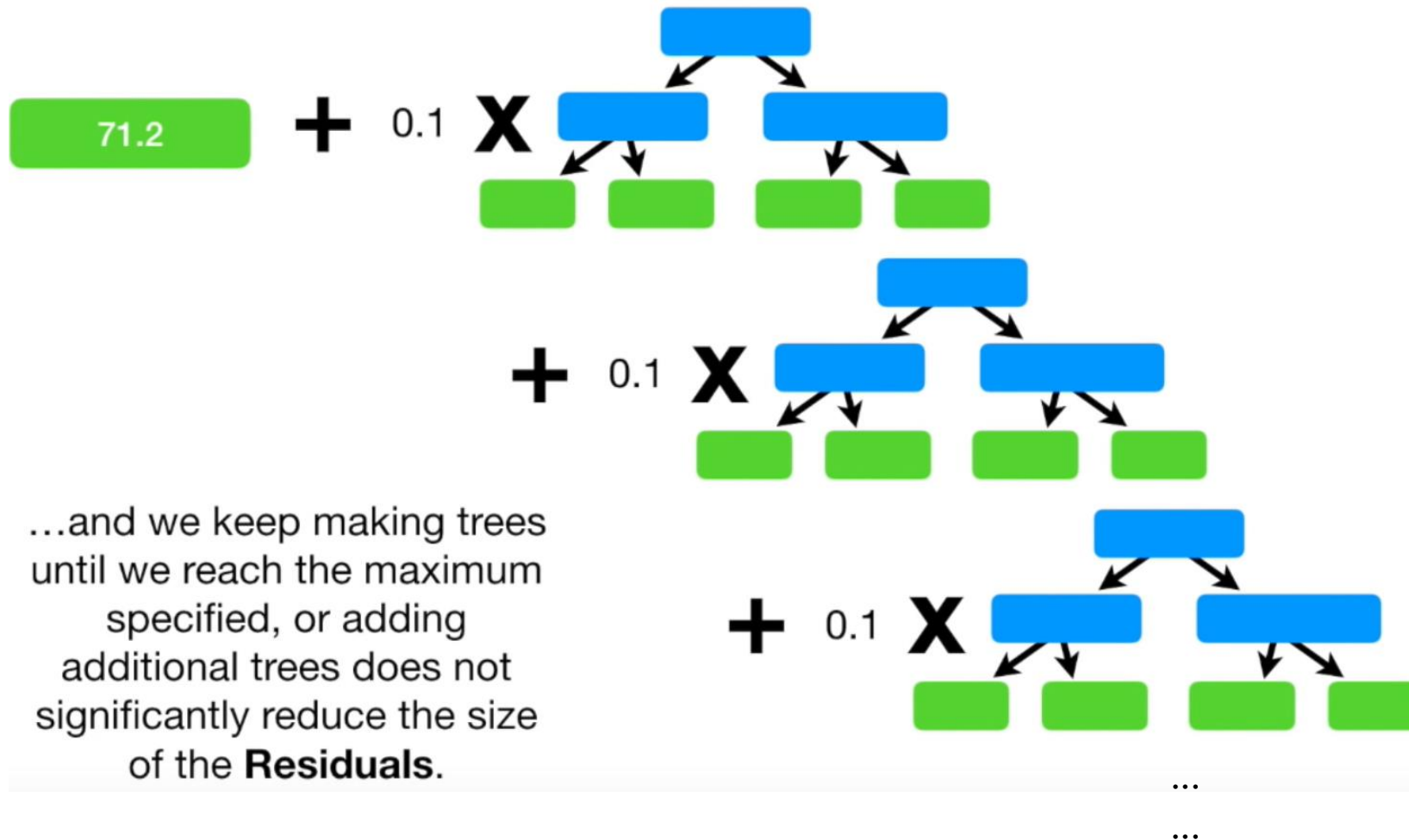
## Unit 05 | Boosting

### GBM (Gradient Boosting Machine)



## Unit 05 | Boosting

### GBM (Gradient Boosting Machine)



### Boosting

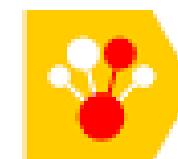
kaggle™



*dmlc*  
***XGBoost***



LightGBM



CatBoost

...

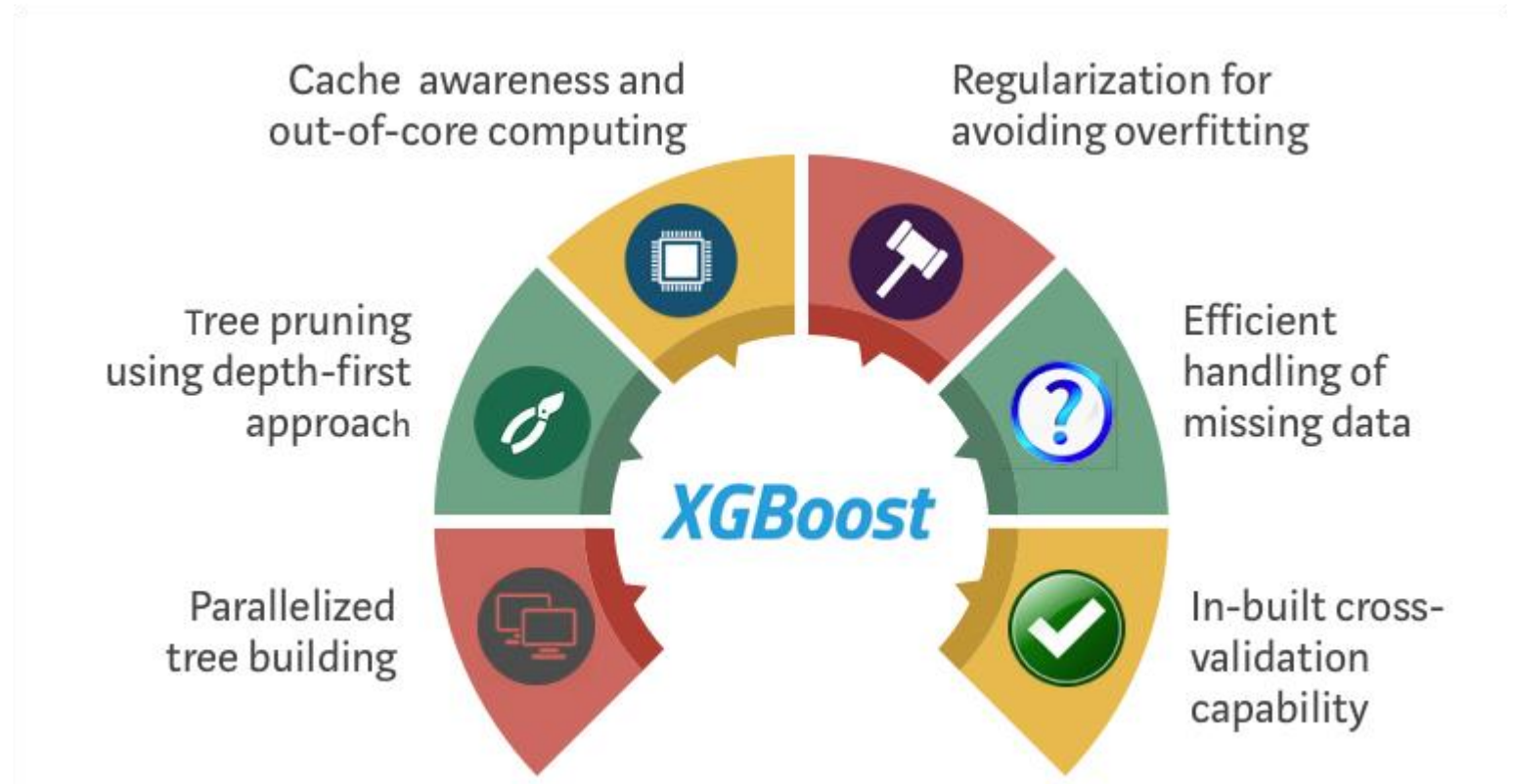
...

## Unit 05 | Boosting

### XGBoost (eXtreme Gradient Boosting)

→ An optimized distributed gradient boosting model designed to be highly efficient, flexible and portable.

- Optimization
  1. Parallelization
  2. Tree Pruning
- Algorithm
  1. Regularization
  2. Built-in Cross Validation
  3. Sparsity Awareness
- Hardware Optimization

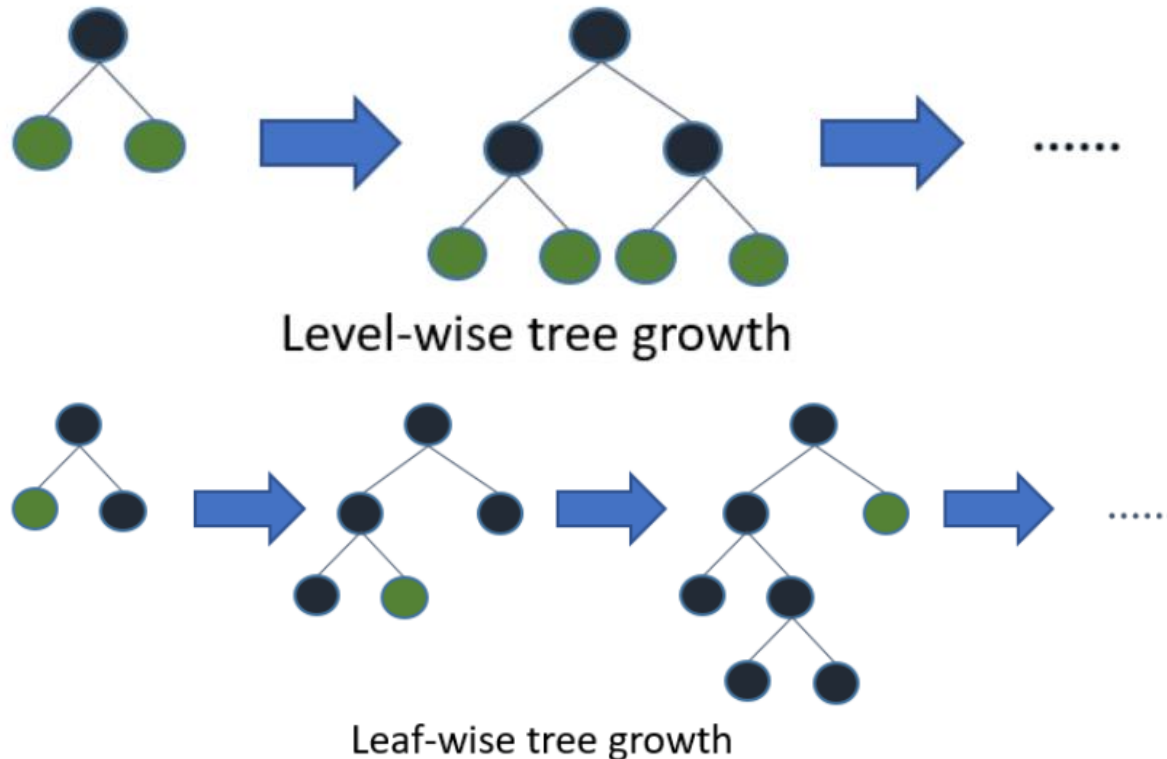




## Unit 05 | Boosting

### Light GBM (Light Gradient Boosting Machine)

- Gradient-based One-Side Sampling (GOSS)



**Input:**  $I$ : training data,  $d$ : iterations

**Input:**  $a$ : sampling ratio of large gradient data

**Input:**  $b$ : sampling ratio of small gradient data

**Input:**  $loss$ : loss function,  $L$ : weak learner

$models \leftarrow \{ \}$ ,  $fact \leftarrow \frac{1-a}{b}$

$topN \leftarrow a \times \text{len}(I)$ ,  $randN \leftarrow b \times \text{len}(I)$

**for**  $i = 1$  **to**  $d$  **do**

$preds \leftarrow models.predict(I)$

$g \leftarrow loss(I, preds)$ ,  $w \leftarrow \{1, 1, \dots\}$

$sorted \leftarrow \text{GetSortedIndices}(\text{abs}(g))$

$topSet \leftarrow sorted[1:topN]$

$randSet \leftarrow \text{RandomPick}(sorted[topN:\text{len}(I)],$

$randN)$

$usedSet \leftarrow topSet + randSet$

$w[randSet] \times = fact$   $\triangleright$  Assign weight  $fact$  to the small gradient data.

$newModel \leftarrow L(I[usedSet], -g[usedSet],$

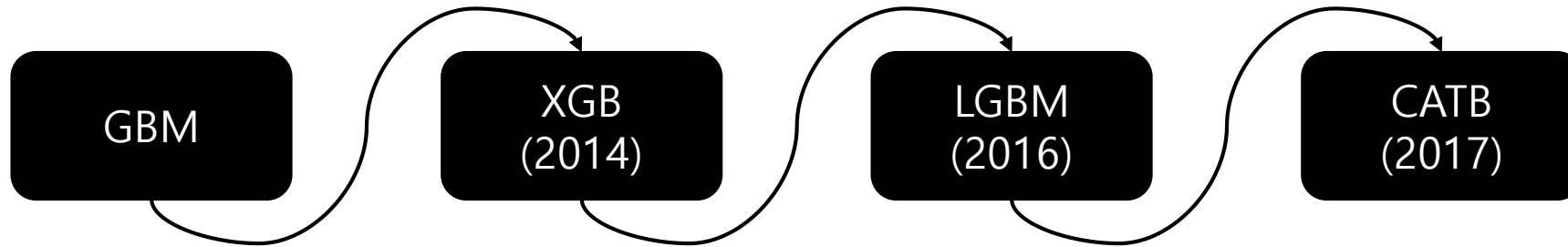
$w[usedSet])$

$models.append(newModel)$



## Unit 05 | Boosting

### Boosting



- Parallel processing
- Pruning types
- Regularization
- Sparsity awareness
- Built-in CV
- **Complex hyperparameters**

- Leaf wise tree growth
- GOSS
- **Overfitting (low # of data)**

- Handle categorical variables well
- **Slower than LGBM (training)**
- **Low performance (most variables are numeric, not categorical)**

## Conclusion

- ✓ Ensemble = “Diversity”
- ✓ Voting (Hard Voting vs Soft Voting)
- ✓ Bagging (Random Forest)
- ✓ Boosting (AdaBoost, GBM, XGB, LightGBM, CatBoost)

## Assignment

- Dacon,kaggle 코드 리뷰!
  - 본인이 직접 선택해서 앙상블 기법을 이용한 코드를 리뷰해주시면 됩니다!
  - Ex : Random forest, Xgboost 그외 모든 ensemble model
  - 채점 기준은 다음과 같습니다.
    - 상세한 코드 리뷰 (앙상블 기법, 파라미터 튜닝 중점적으로)
    - 코드 자체의 완성도 및 대회의 난이도
  - 앞서 배운 강의의 다양한 모델과 기법들을 활용해서 리뷰해주세요!
  - 코드 가져온 링크도 반드시 첨부해주세요! 채점할 때 사용합니다.
  - **Baseline Model은 안됩니다!**
- ✓ 코드리뷰는 꼭 git이 아니라도 Notion, tistory, 기술블로그 모든 형태 가능합니다!

## Assignment

- ✓ (예시)
- ✓ <https://dacon.io/competitions/official/235930/codeshare>
- ✓ <https://dacon.io/competitions/official/236035/codeshare>
  
- ✓ (코드리뷰 예시)
- ✓ <https://tensorflow.blog/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D/2-3-6-%EA%B2%B0%EC%A0%95-%ED%8A%B8%EB%A6%AC%EC%9D%98-%EC%95%99%EC%83%81%EB%B8%94/>
- ✓ <https://lsjsj92.tistory.com/558>

# Reference

<https://velog.io/@jiselectric/Ensemble-Learning-Voting-and-Bagging-at6219ae>  
<https://www.shutterstock.com/ko/image-vector/voting-ballot-box-icon-election-vote-1342666166>  
<https://blog.naver.com/PostView.nhn?blogId=ckdgus1433&logNo=221588139765>  
<https://huidea.tistory.com/35>  
<https://lsjsj92.tistory.com/559>  
<https://www.javatpoint.com/machine-learning-random-forest-algorithm>  
<https://community.alteryx.com/t5/image/serverpage/image-id/33046iB8743F7094DB9C87/image-size/large?v=1.0&px=999>  
<https://michael-fuchs-python.netlify.app/2020/03/26/ensemble-modeling-boosting/>  
<https://injo.tistory.com/31>  
<https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/>  
<https://3months.tistory.com/368>  
<https://koalaverse.github.io/machine-learning-in-R/gradient-boosting-machines.html>  
<https://www.youtube.com/channel/UCPq01cgCcEwhXl7BvcwlQyg>  
<https://www.youtube.com/watch?v=3CC4N4z3GJc>  
<https://bkshin.tistory.com/entry/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-15-Gradient-Boost>  
<https://statinknu.tistory.com/33>  
<https://wyatt37.tistory.com/5>

Q & A

들어주셔서 감사합니다.