Statistical Machine Learning

7주차 : Ensemble

담당: 18기 신인수

1. What is Ensemble?

2. Ensemble Methods

3. Ensemble Models

참고자료

- Hastie, Trevor. "The elements of statistical learning: data mining, inference, and prediction." (2009).
- James, Gareth, et al. *An introduction to statistical learning: With applications in python*. Springer Nature, 2023.
- Géron, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.
- 권철민, 『파이썬 머신러닝 완벽 가이드』, 위키북스, 2020
- StatQuest 유튜브

1. What is Ensemble Learning?

Ensemble

여러 개의 악기가 조화롭게 연주하는 것

Ensemble

Wisdom of the crowd

Ensemble

→ 그룹 과외

Ensemble learning

- 다수의 기본 분류 모델(base classifier, **weak classifier**)의 예측 결과를 종합하여, 정확한 예측 성능을 얻도록 하는 방법론

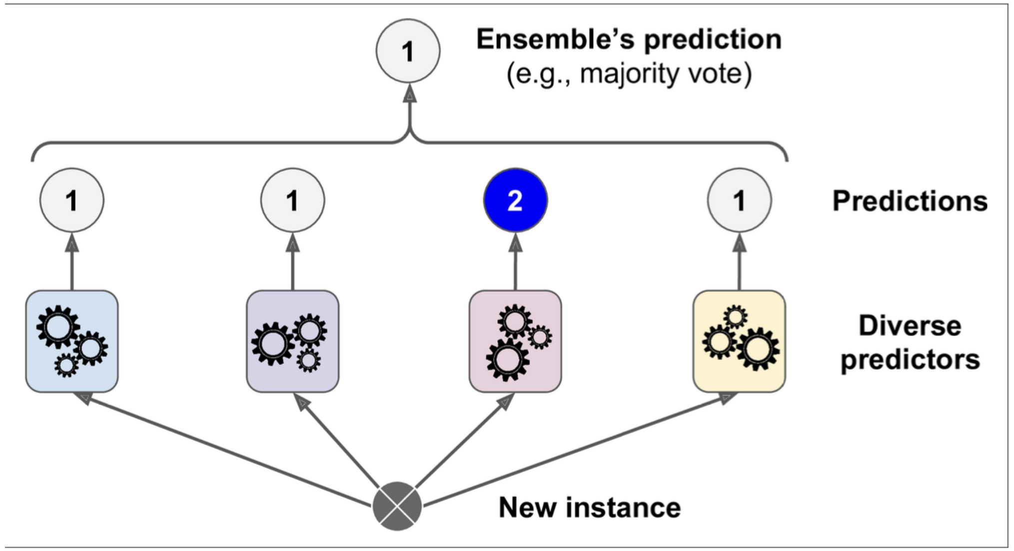
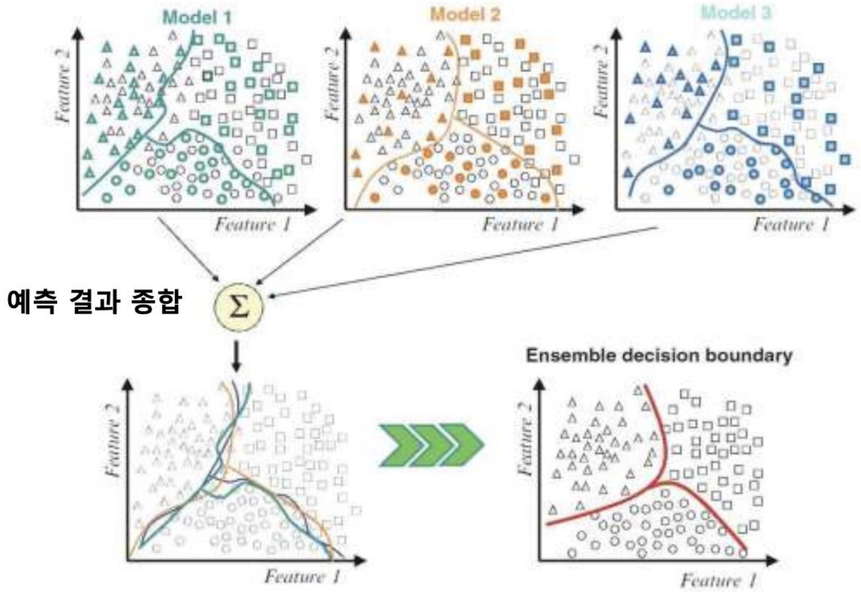


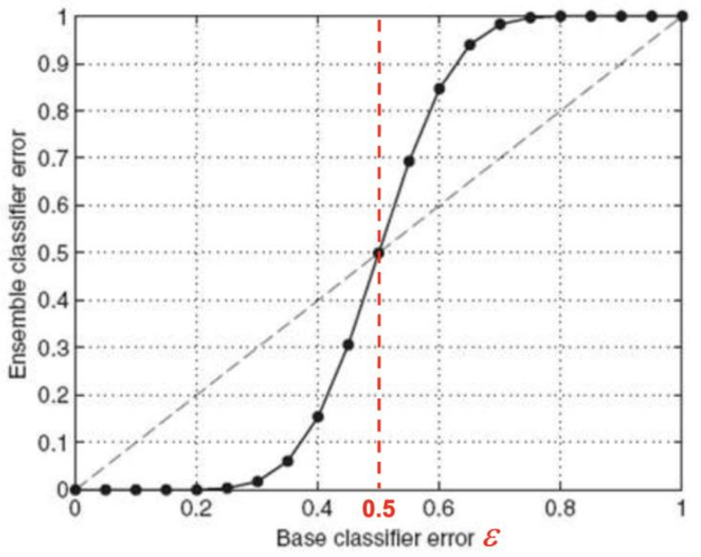
Figure 7-2. Hard voting classifier predictions



Ensemble

Example

- 25 base classifiers
- Error rate $\varepsilon = 0.35$
- Each independent
- Ensemble classifier : Majority vote



$$e_{ensemble} \sim Binomial(25, 0.35)$$

$$P(incorrect\ classifier\ \geq 13) = e_{ensemble} = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Probability that the ensemble classifier makes wrong prediction

왜 Weak Learner를 사용하는가?

Weak Learner: low model complexity, high bias, low variance
Strong learner: high model complexity, low bias, high variance

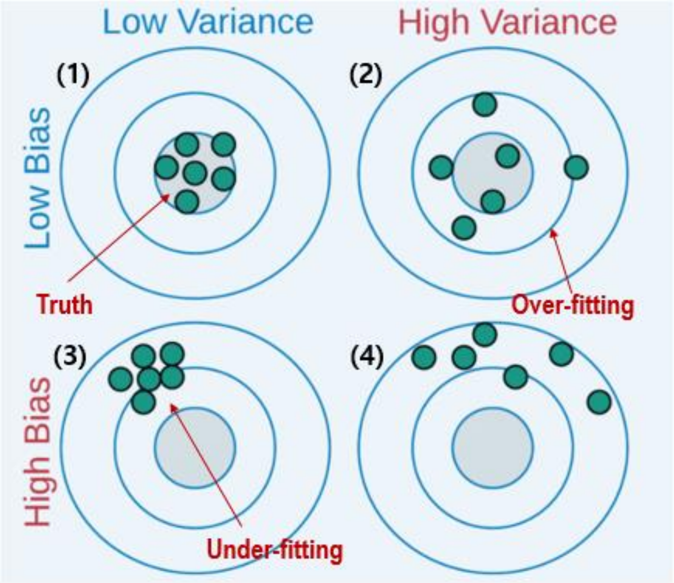
→ Schapire, Robert E. "The strength of weak learnability." *Machine learning* 5 (1990): 197-227.

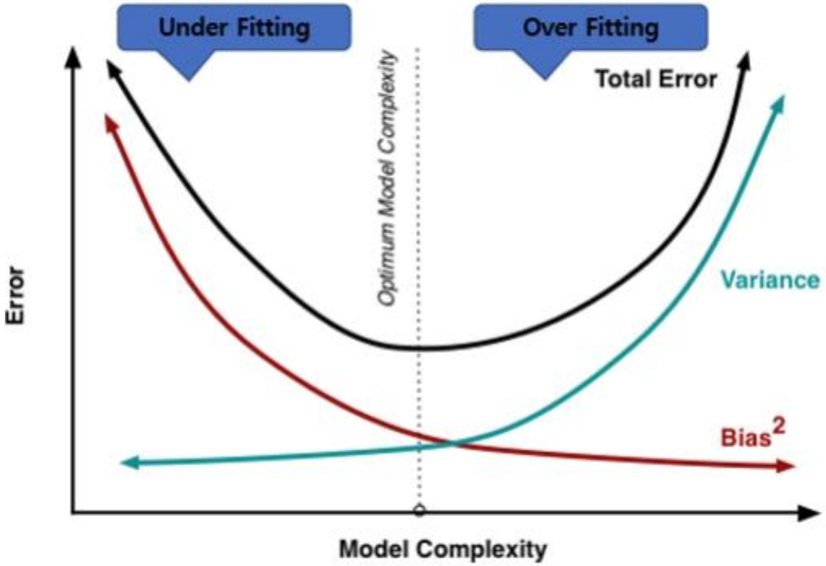
Weak learner와 Strong learner가 동일함!
Weak learner의 예측확률이 50%를 넘어가면, 참값에 도달하게 됨!

→ 시간, 용량이 많이 드는 strong learner를 굳이 안 써도 된다!

THEOREM 2. For $0 < \epsilon < 1/2$ and for $0 < \delta \leq 1$, the hypothesis returned by calling Learn(ϵ, δ, EX) is ϵ-close to the target concept with probability at least $1 - \delta$.

Bias and Variance





$Y = \mathrm{f(x) + e}$ (Y: predictions, x: covariates)

$$\mathrm{Error(x)} = \mathrm{E}\left[(Y - f'(x))^2\right] = \mathrm{E}\left[((f(x) - f'(x)) + e)^2\right]$$

$$= (\underbrace{E[f'(x) - f(x)]}_{\text{Bias}})^2 + \underbrace{\mathrm{E}\left[(f'(x) - E[f'(x)])^2\right]}_{\text{Variance}} + \sigma_e^2$$

$$\mathrm{Error(x)} = Bias^2 + Variance + Irreducible\ Error$$

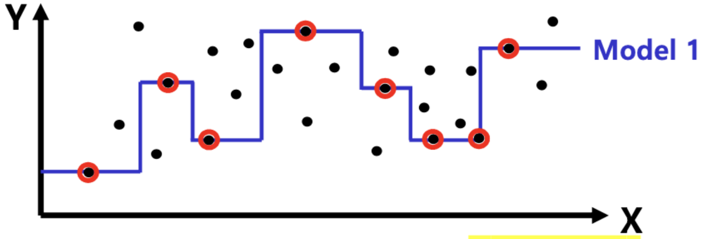
Ensemble

Ensemble Learning

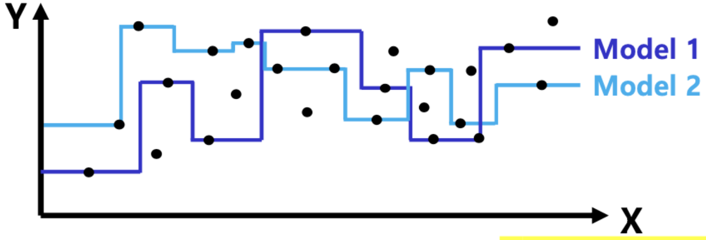
- Reduce Learning error
- Reduce Bias
- Reduce Variance

→ Logistic regression: high bias low variance
→ Decision tree: low bias high variance

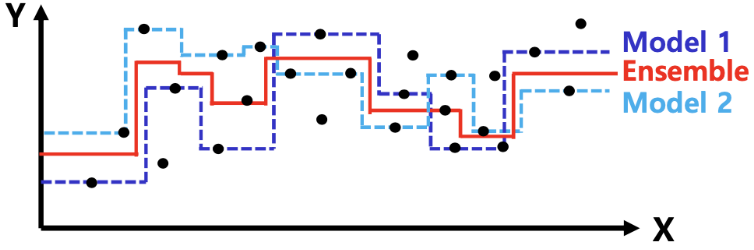
→ Ensemble 사용시 단점을 상쇄



Sample 데이터에 대해 잘 예측 “ Low bias ”



Model 1과 Model 2 예측 값이 서로 다름 “ High variance ”



Model 1과 Model 2 예측 값의 평균 사용: Ensemble

“Low variance”

Ensemble

→ 그룹과외를 하는데...

Voting → 다른 알고리즘 사용 가능

Hard Voting

Soft Voting

Weighted Voting

→ 다른 학생

Stacking

Meta level Learning

Blending

Bagging → 같은 알고리즘

Bootstrap + Aggregating

→ 다 같은 학생

Boosting

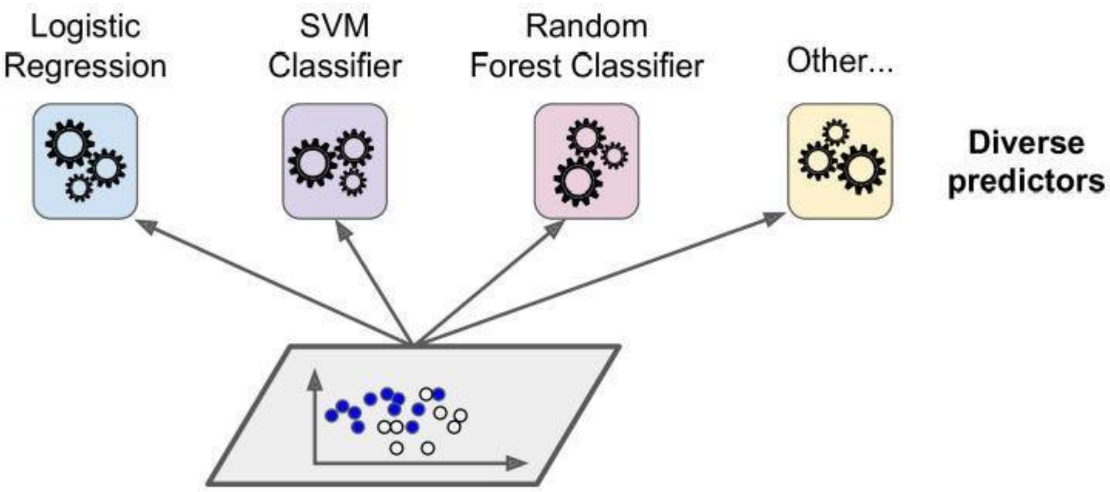
Error learner

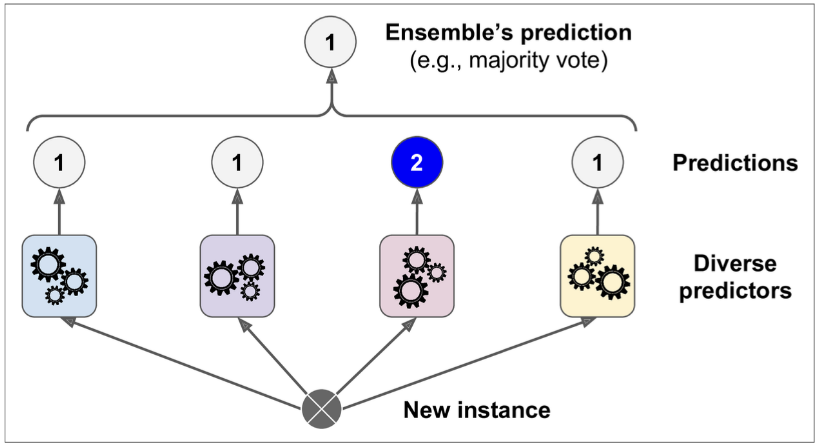
2. Ensemble Methods

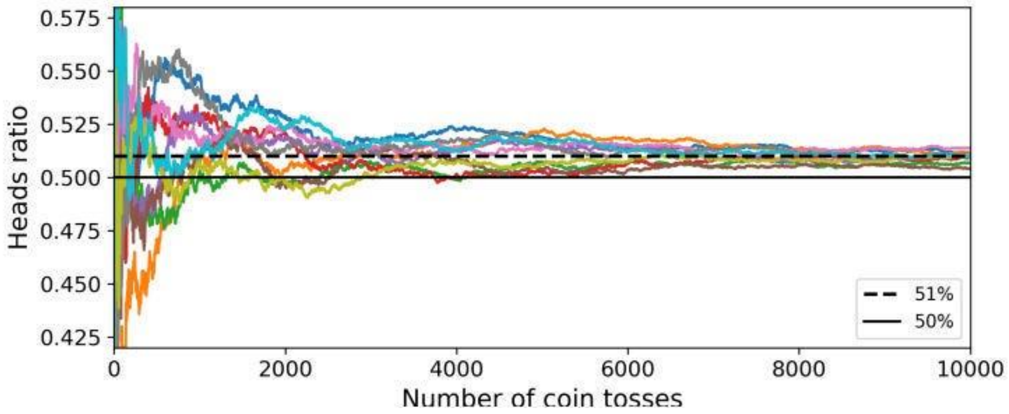
Voting

→ 동점인 경우 빠른 클래스를 따름

Hard Voting : Majority Voting

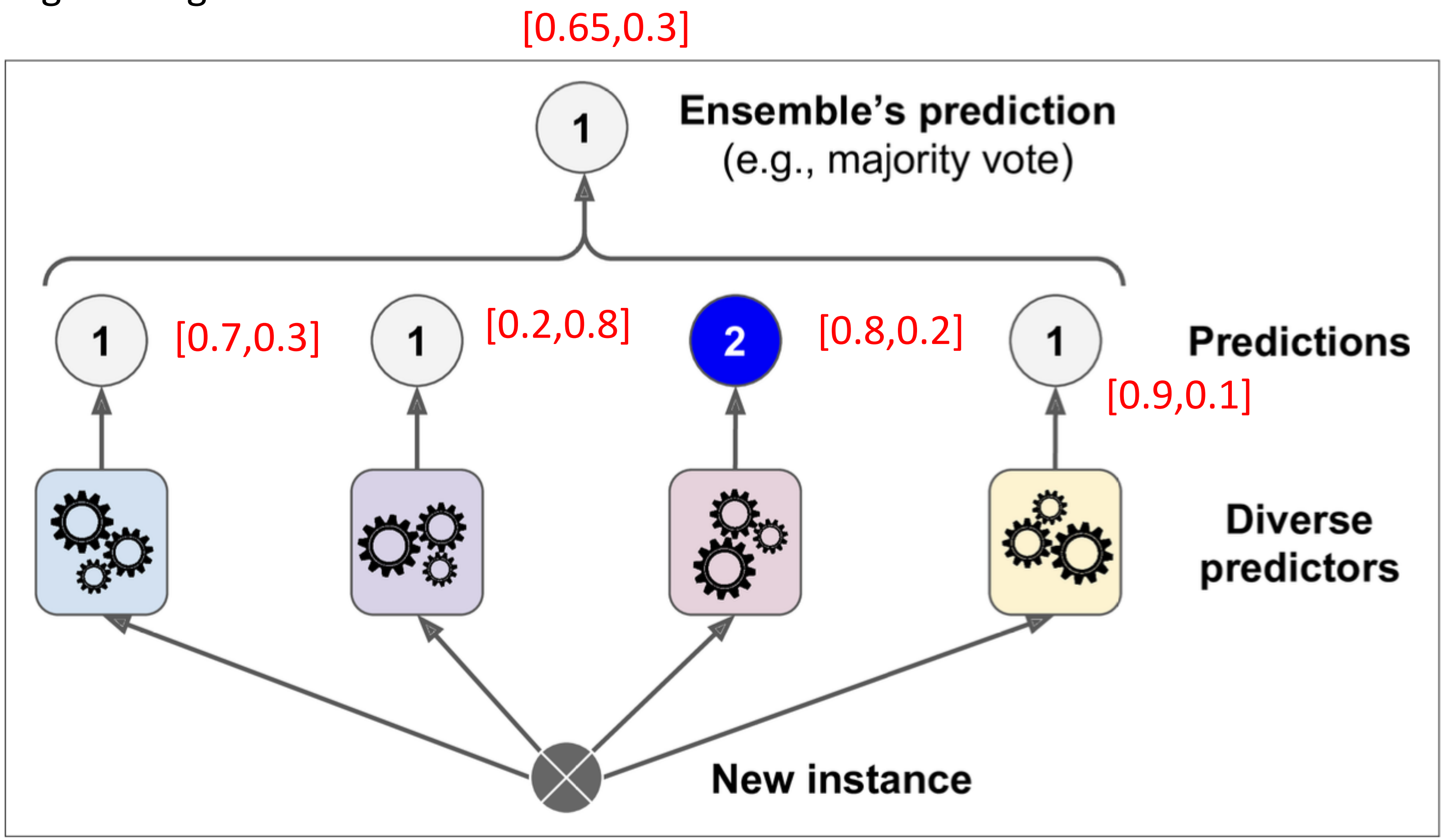




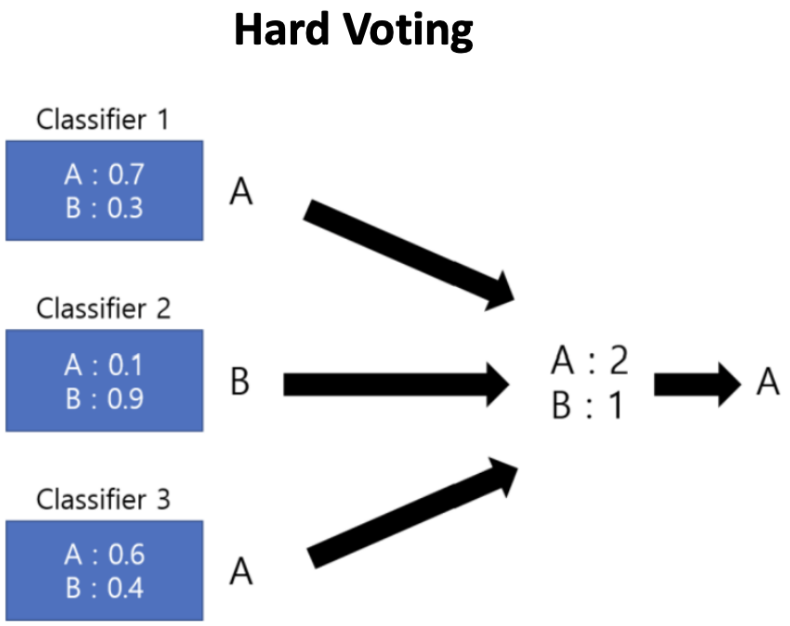


Voting

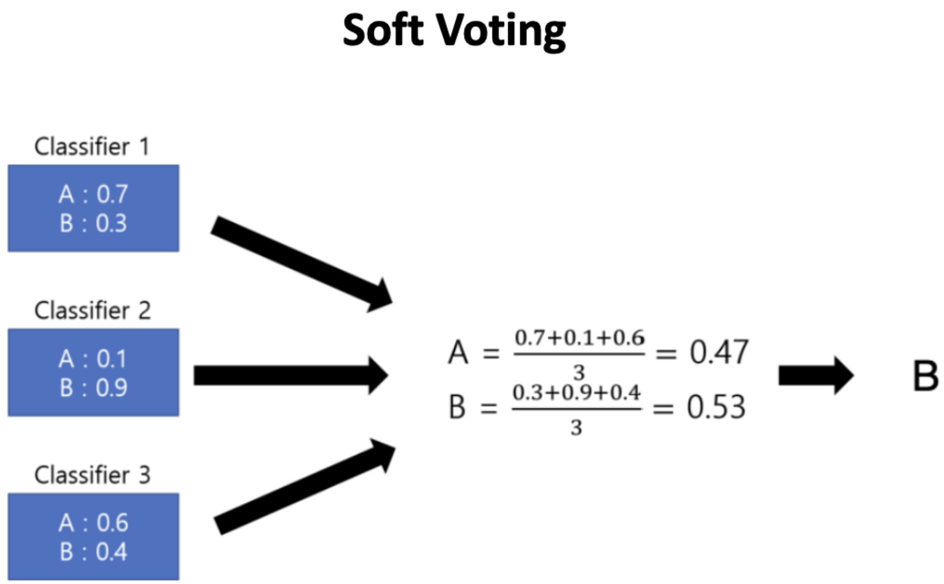
Soft Voting : Average Voting



Voting



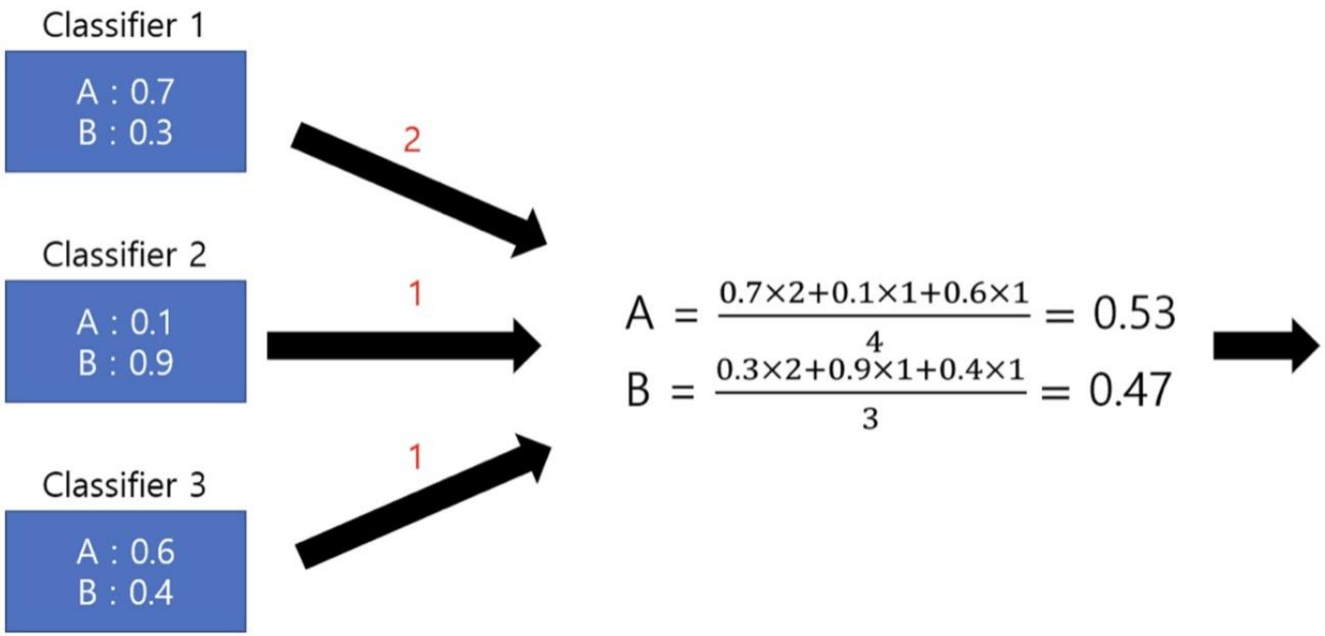
Hard Voting
Classifier 1
A : 0.7
B : 0.3
A
Classifier 2
A : 0.1
B : 0.9
B
Classifier 3
A : 0.6
B : 0.4
A
A : 2
B : 1
A



Soft Voting
Classifier 1
A : 0.7
B : 0.3
Classifier 2
A : 0.1
B : 0.9
Classifier 3
A : 0.6
B : 0.4
$A = \frac{0.7+0.1+0.6}{3} = 0.47$
$B = \frac{0.3+0.9+0.4}{3} = 0.53$
B

Voting

Soft + Weighted



Bagging

→ 문제은행

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$

Bagging = Bootstrap + Aggregating(Average)

Bootstrap : sampling with Replacement → Variance 개선

Probability that one sample is not chosen by bootstrap
(N records, N sample size)

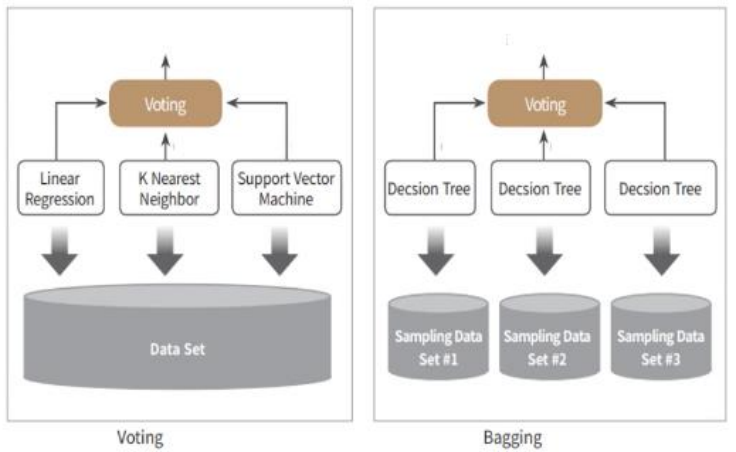
$= (1 - \frac{1}{N})^N$

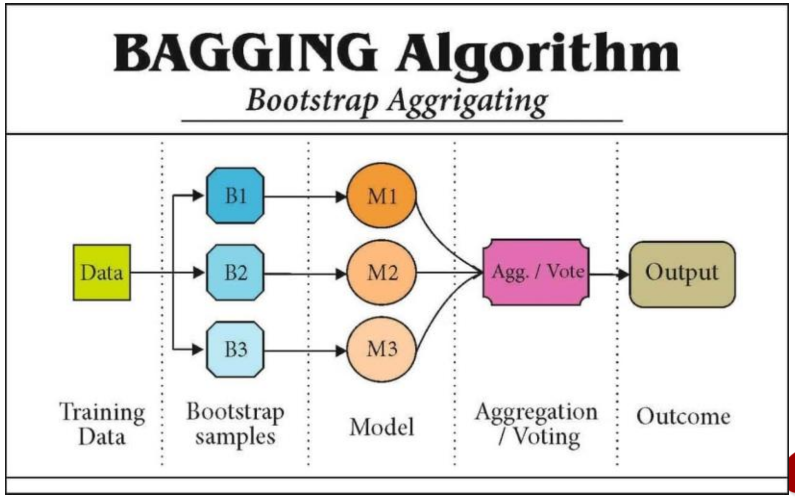
If N is large enough, then $\lim_{N\to\infty}\left(1-\frac{1}{N}\right)^N = e^{-1} = 0.3678$

36.7% of original train dataset

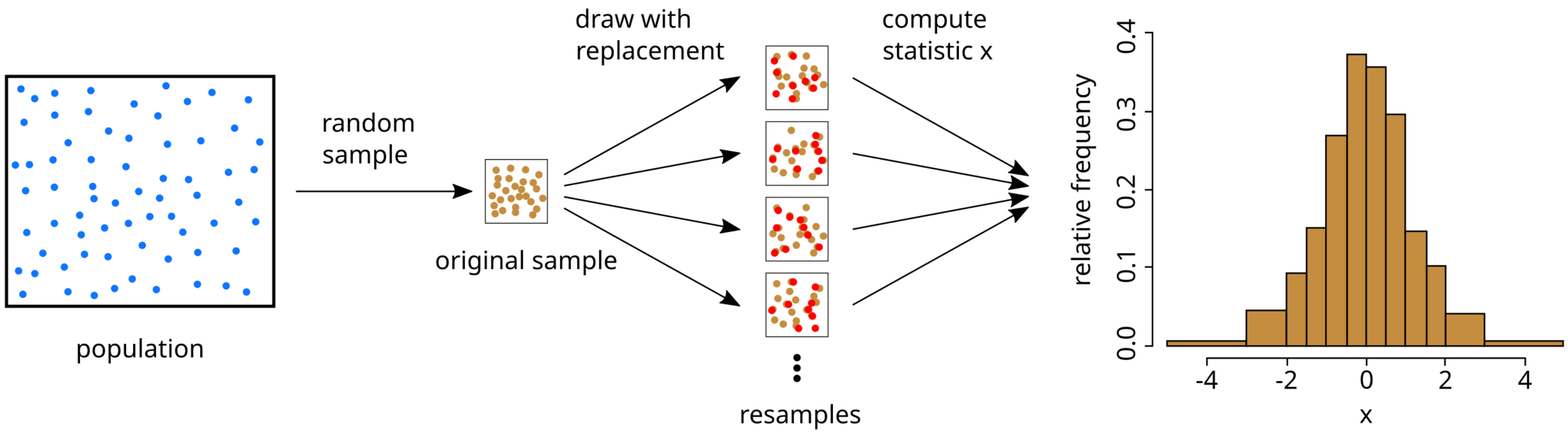
→ 이게 마음에 안 들면 Jacknife, Balanced boostrap

Aggregating : Majority Voting, Weighting, Soft voting





Bootstrapping (4주차 Recap)



Sample size가 N, replication 수를 B 라고 하자.

- sample에서 N개를 복원 추출 (중복된 값 허용)
- B번 반복
- B개의 sample을 통해 통계량 계산 → bootstrap mean, bootstrap variance, bootstrap sd 등등

Boosting

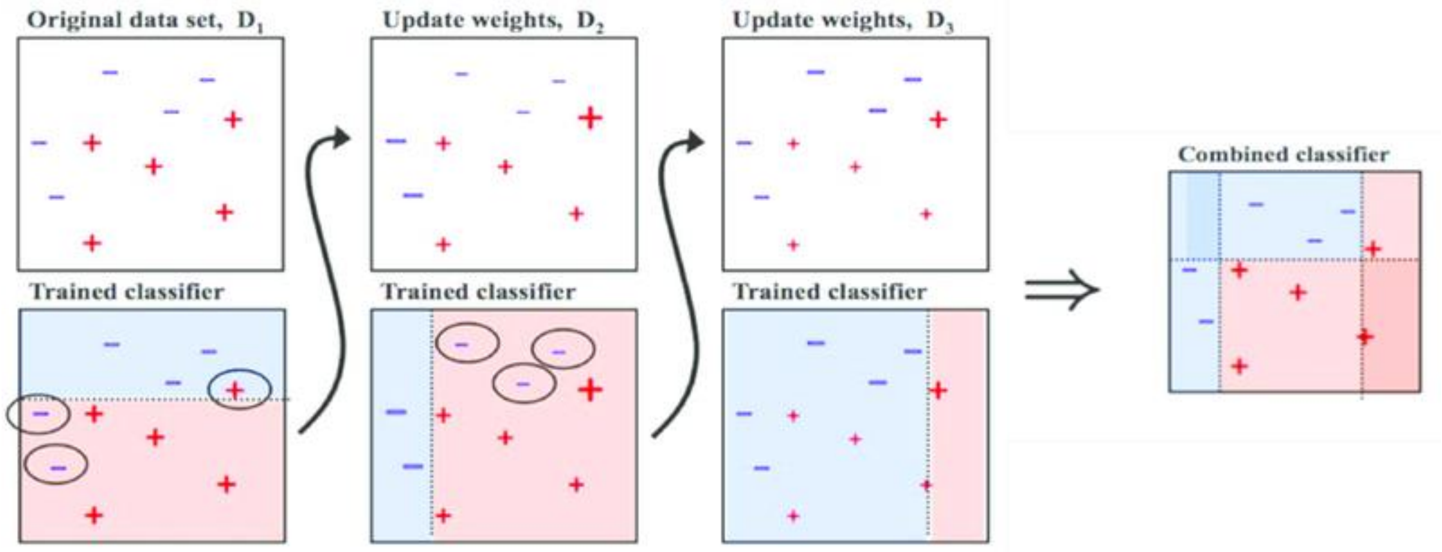
→ 오답노트

Boosting

- 오분류된 샘플에 더 많은 가중치 부여 -> 오답을 다시 학습
- 예측이 틀린 데이터가 다시 뽑힐 가중치가 높아진다.
- 이전 모델이 잘못 예측한 부분을 집중적으로 학습

→ Bias 개선

→ 시간이 오래 걸림
→ Bootstrapping 사용 X
→ Sequential 하게 학습



Example: Boosting for regression tree

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
   $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

→ B, λ, d는 사용자가 정의하는 hyperparameter에 해당
→ B: tree 개수
→ λ : learning rate
→ d: split의 개수

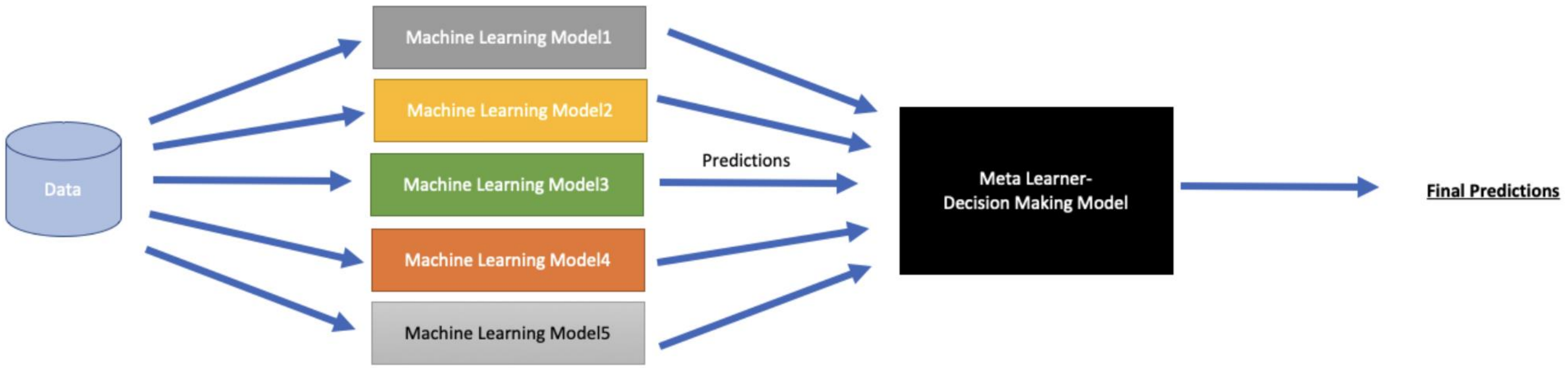
→ Sequential하게 update 되는 중

Stacking

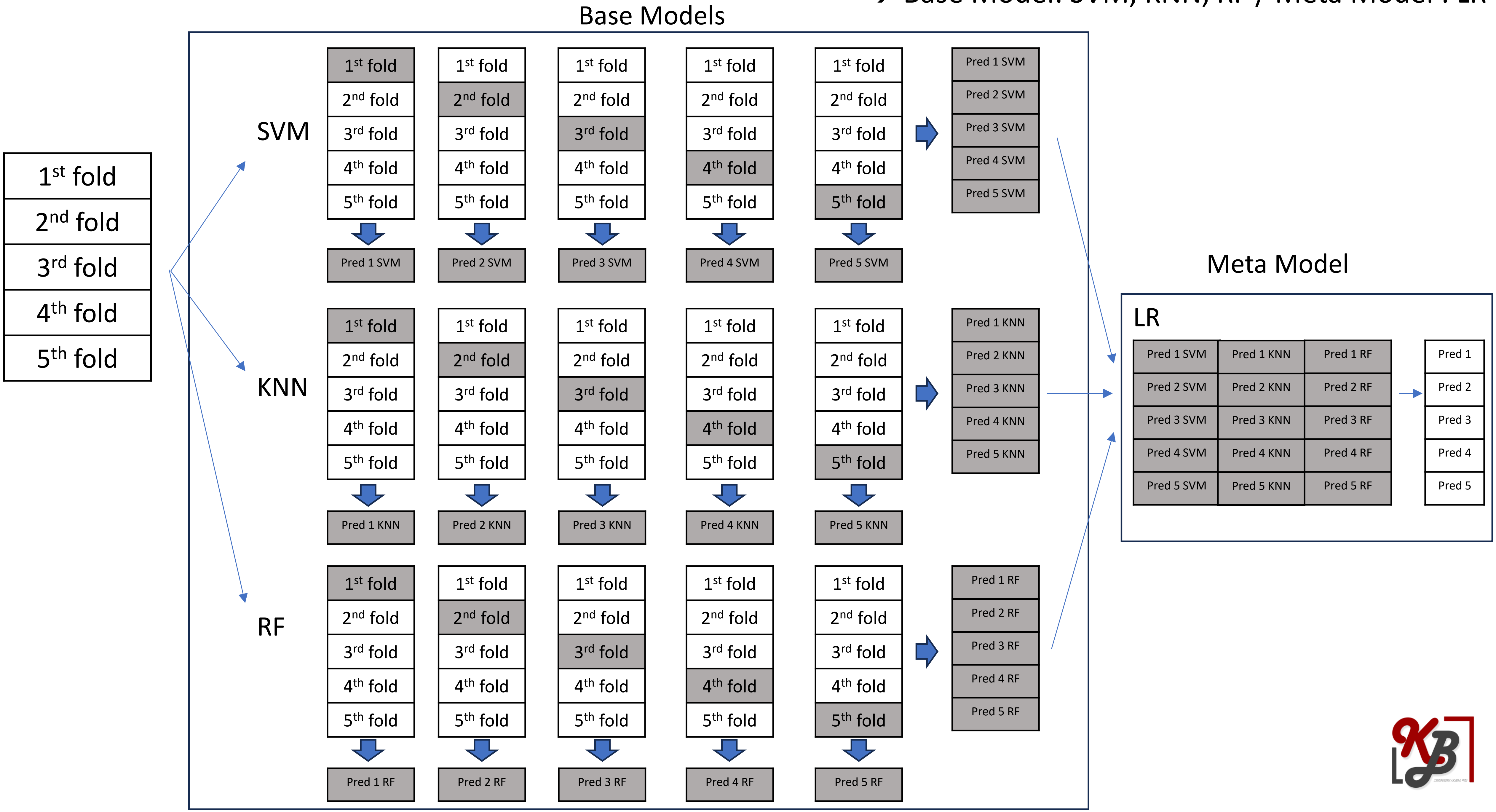
→성능 끌어올리기

Stacking Generalization

- Meta-learning model
- **여러 가지 모델들의 예측값을 최종 모델의 학습 데이터로 사용**
- K-fold cv
- Step 0 : 각 weak model에 k-fold cv를 적용하여 예측 데이터를 형성
- Step 1 : step 0에서 만든 예측 데이터를 stack.하여 meta-model을 train 및 예측



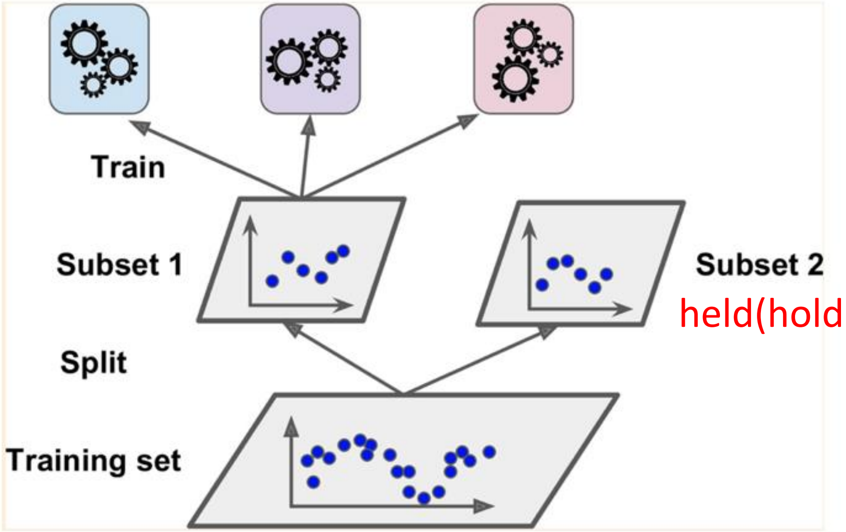
→ Base Model: SVM, KNN, RF / Meta Model : LR



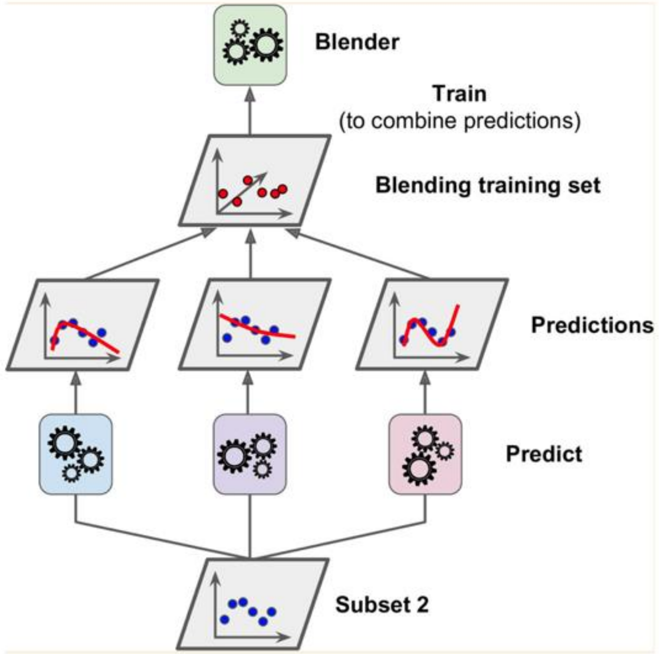
Blending

Blending Generalization

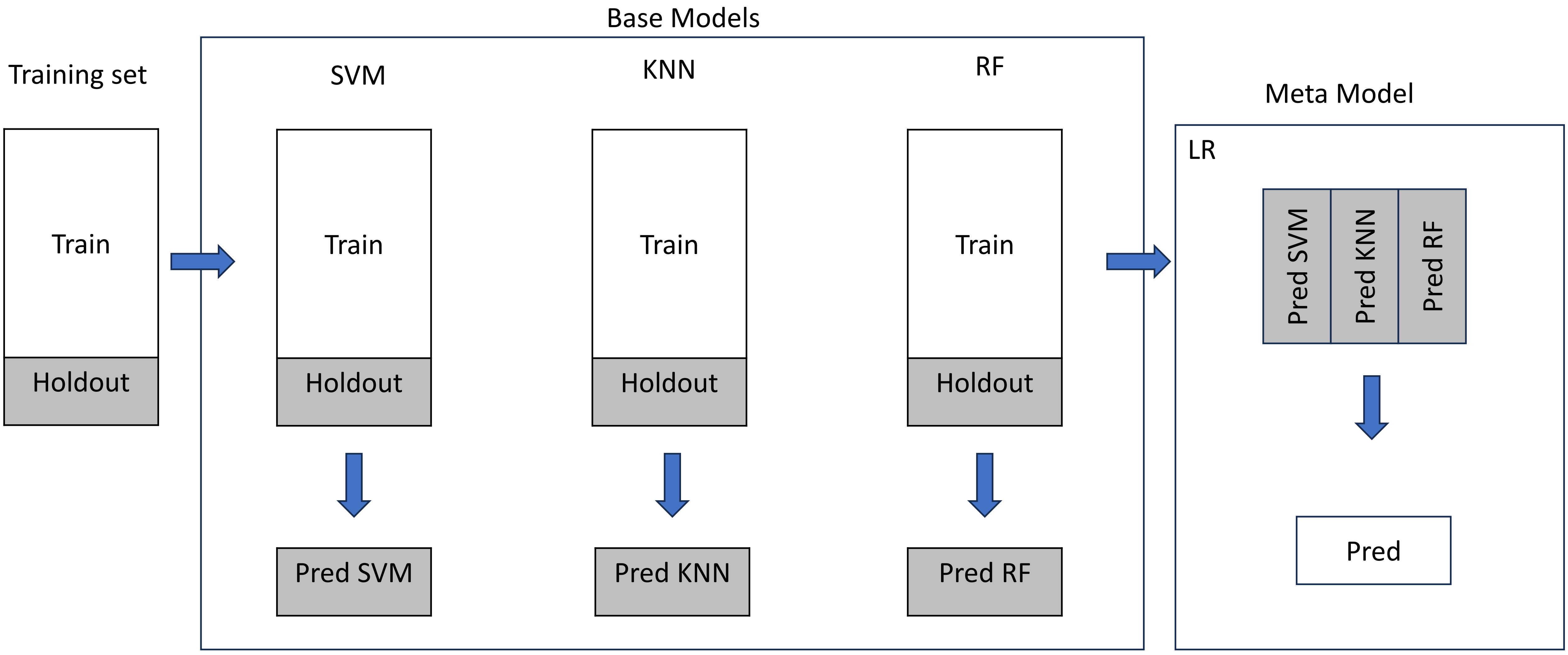
- Meta-learning model
- 개별 모델의 예측값을 다시 input으로 사용
- Use hold-out set



held(hold) out set



→ Base Model: SVM, KNN, RF / Meta Model : LR



3. Ensemble Models

RandomForest

Breiman, Leo. "Random forests." *Machine learning* 45 (2001): 5-32.

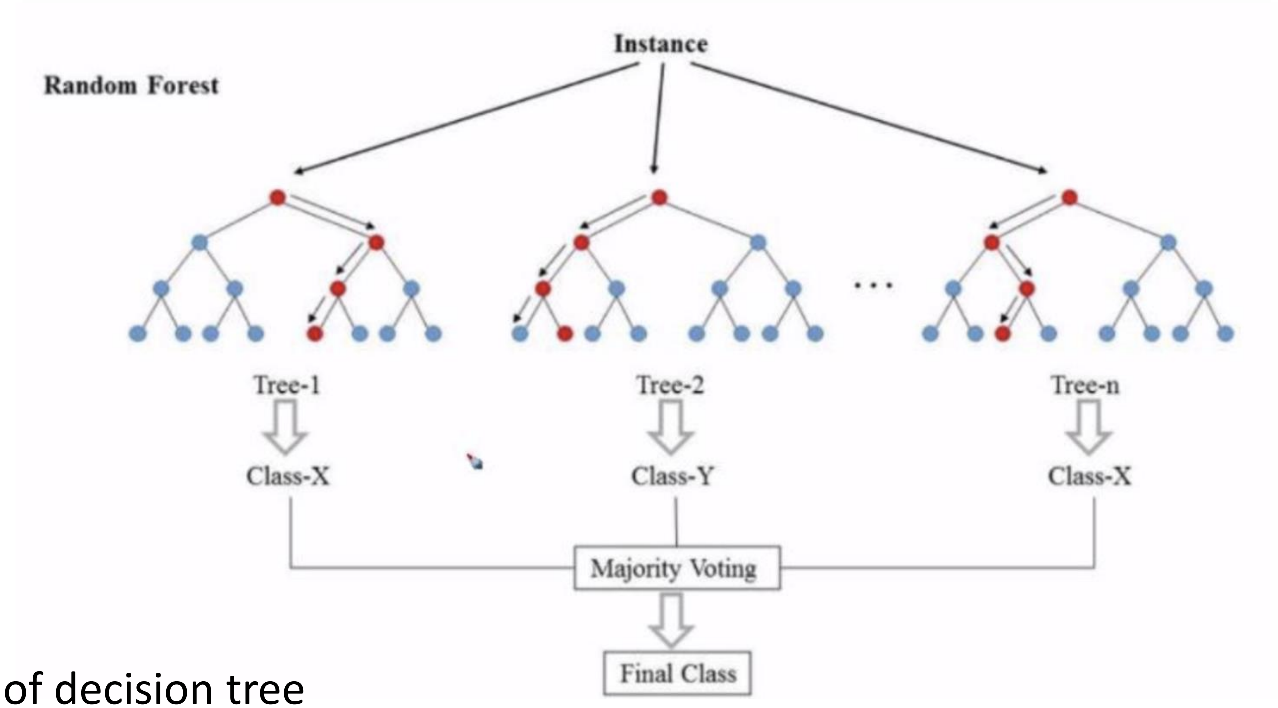
Feature Bagging → RandomForest

RandomForest Decision Tree Generation

- **Forest-RI(random input)**
  randomly select F features
  to split each node of decision tree

- **Forest-RC(randomly combined)**
  F randomly combined new features
  (F linear combination)

- **Randomly select**
  one of the F best splits at each node of decision tree



RandomForest

- 독립변수를 $X_1 \dots X_p$, 종속변수를 Y라고 하자.
- X_k와 Y의 correlation이 높으면, tree를 만들 때 Xk 기준으로 만들려고 한다

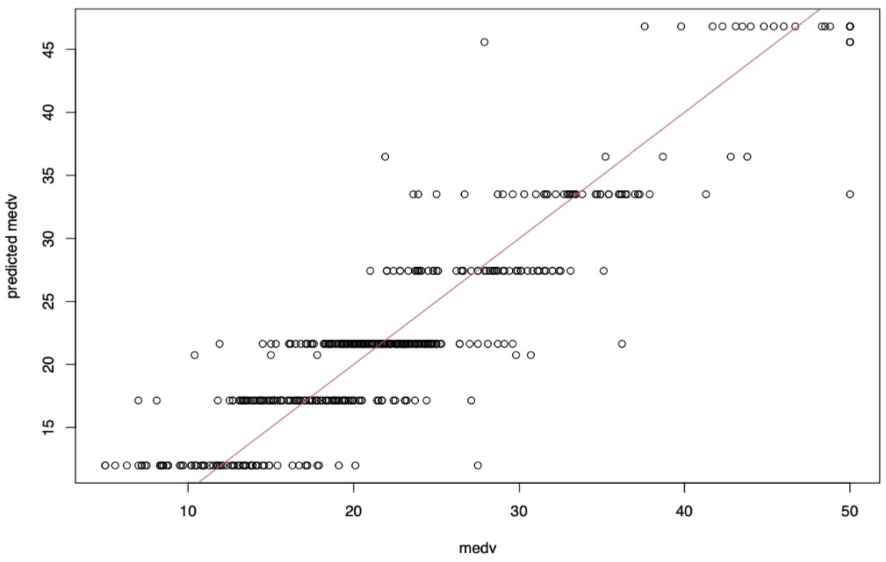
→ Tree들이 correlation이 높게 됨 (BAD)

- Tree들의 correlation을 떨어뜨리고 싶다

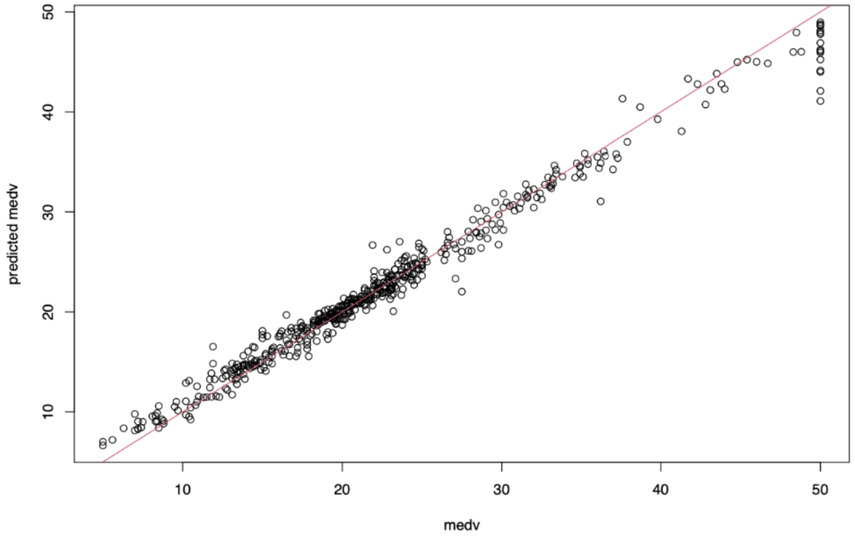
→ 변수들을 적게 사용하자 (무작위로 선택)

RandomForest

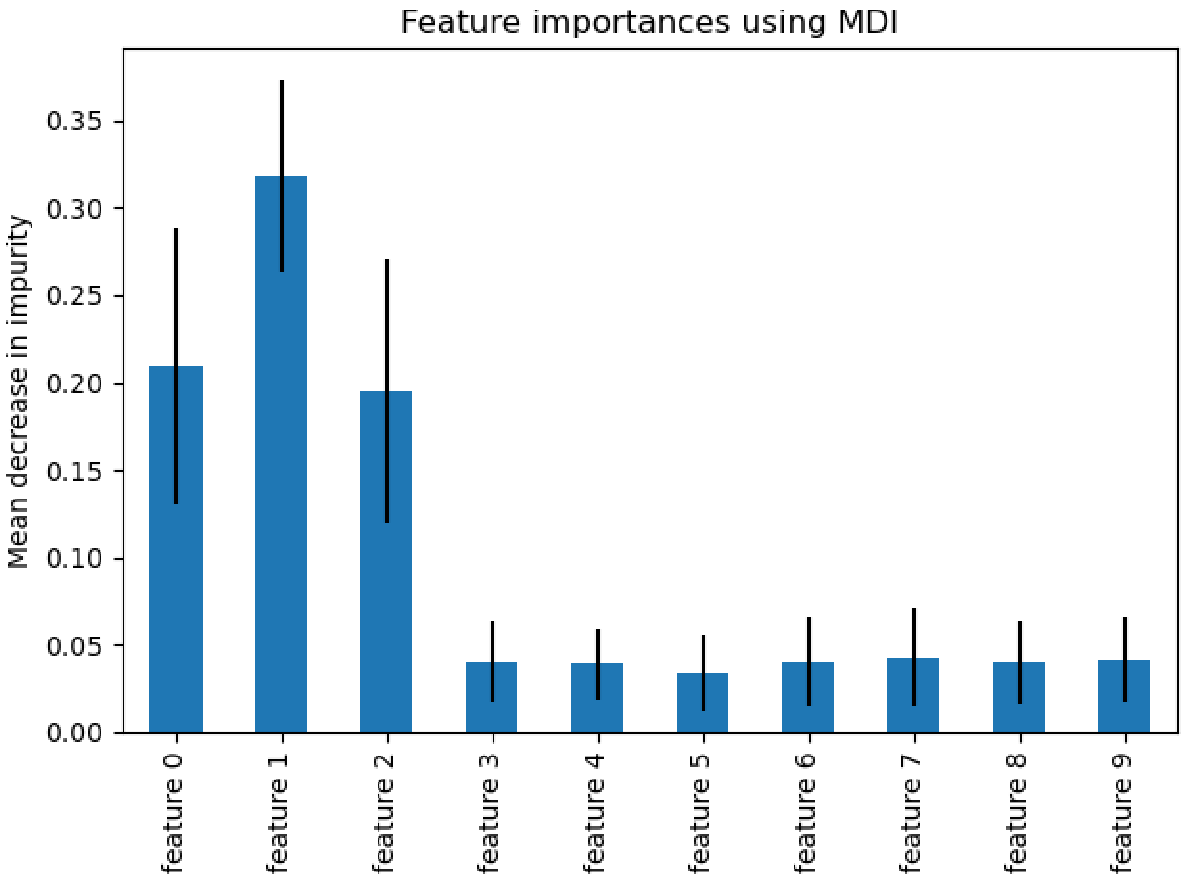
Single Tree



Random Forest



RandomForest

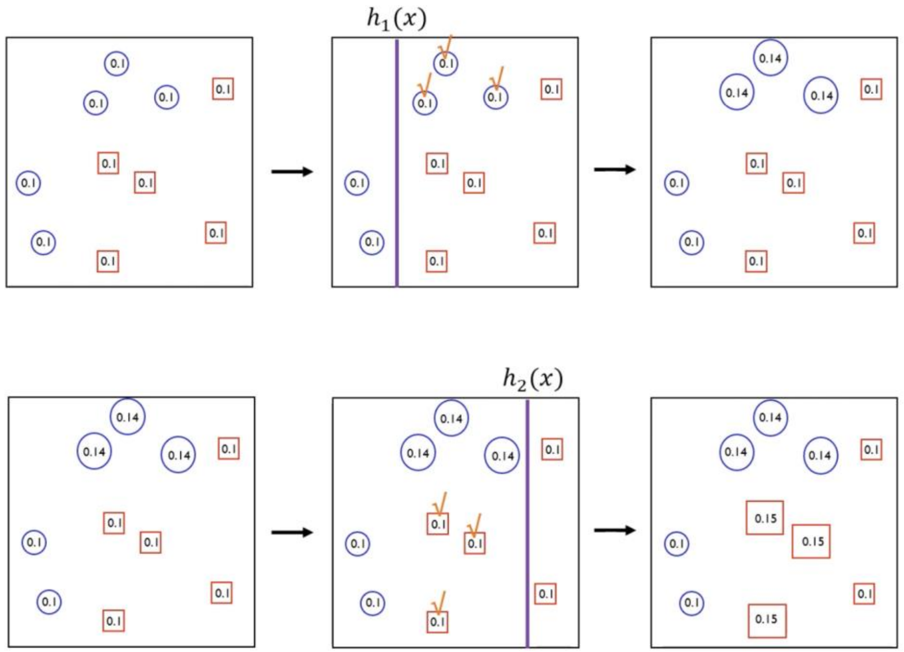


→ Random Forest의 가장 큰 특징은
feature importance를 구할 수 있다는 것!

Adaboost

Adaboost : Adaptive + Boosting

- Adaptive : 이전 모델이 잘못 분류한 데이터의 가중치를 adaptive하게 변경
- Boosting : 이전 모델이 잘못 분류한 데이터들을 중심으로 학습



정분류 sample : 그대로
오분류 sample : 가중치 ↑

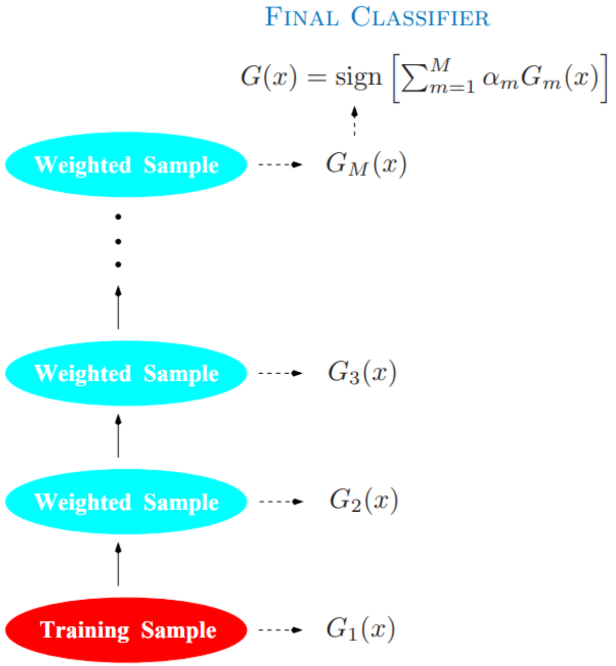


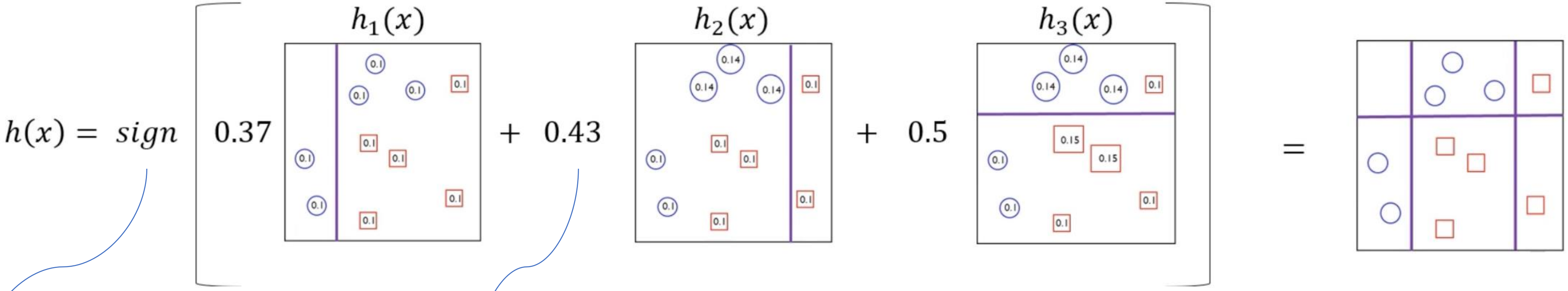
FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Adaboost

→ Weight update 과정이 빠져 있다.

$$W_i = \frac{1}{n} \quad L_j = \frac{\sum_{i=1}^{n} W_i\, I(y_i \neq h_i(x))}{\sum_{i=1}^{n} W_i}. \alpha_j = \log\left(\frac{1-L_j}{L_j}\right)$$

$$h(x) = sign\left[\sum_{i=1}^{m=3} \alpha_j h_j(x)\right]$$



aggregating

Model importance based on error rate

Adaboost

Algorithm 10.1 *AdaBoost.M1.*

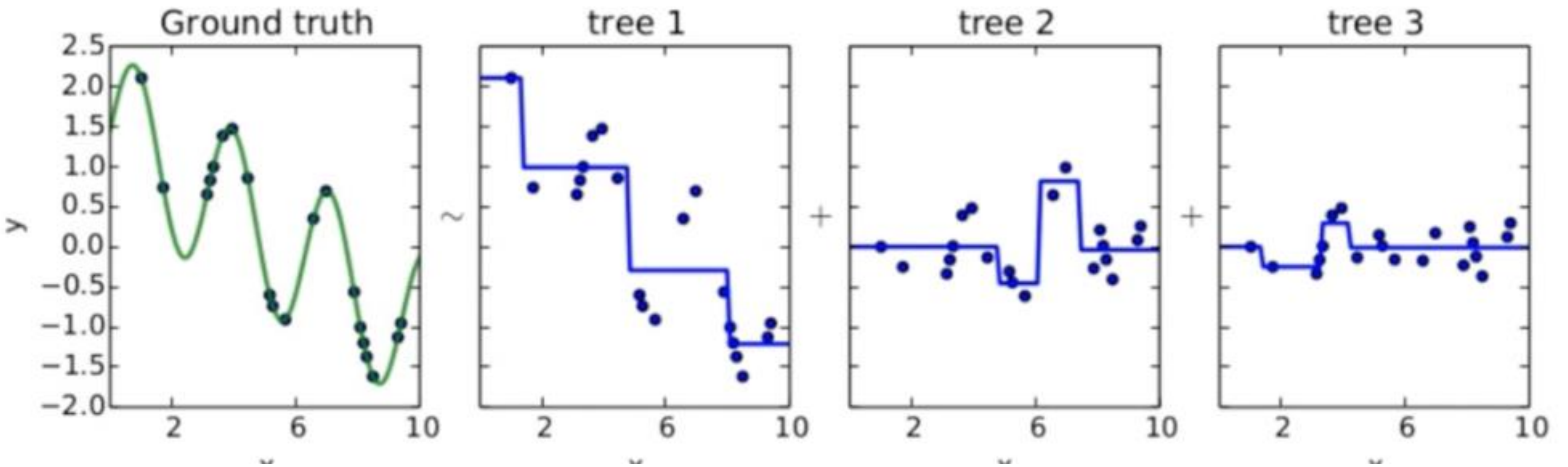
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Gradient Boosting(GBM)

- Gradient boosting = Boosting with gradient decent
- Tree 1을 통해 Y를 예측하고 residual로 tree2 다시 학습
- 점차 residual(실제값과 예측값의 차이 작아짐
- Gradient boosting model = tree1 + tree2 + tree3

$$loss\ function : (y, f(x)) = \frac{1}{2}(y - f(x))^2$$

$$negative\ gradient : \frac{\partial(y, f(x))}{\partial f(x)} = -\frac{\partial\left[\frac{1}{2}(y - f(x))^2\right]}{\partial f(x)} = -(f(x) - y) = y - f(x)$$

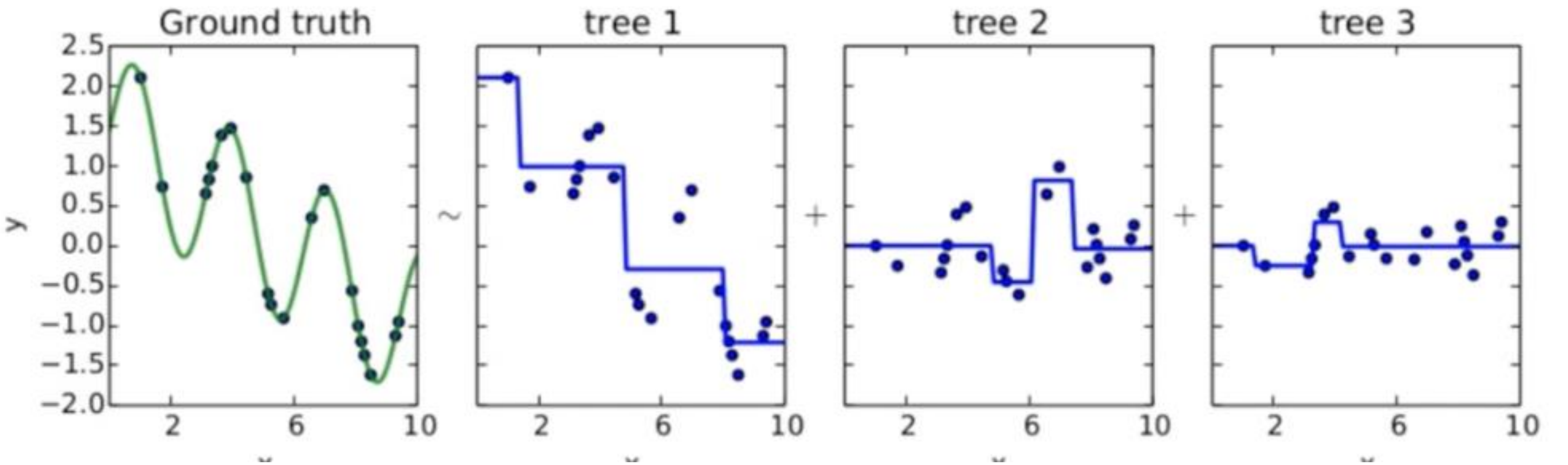


Gradient Boosting(GBM)

- Gradient boosting = Boosting with gradient decent
- Tree 1을 통해 Y를 예측하고 residual로 tree2 다시 학습
- 점차 residual(실제값과 예측값의 차이 작아짐
- Gradient boosting model = tree1 + tree2 + tree3

$$loss\ function : \left(y, f(x)\right) = \frac{1}{2}\left(y - f(x)\right)^2$$

$$negative\ gradient : \frac{\partial\left(y, f(x)\right)}{\partial f(x)} = -\frac{\partial\left[\frac{1}{2}\left(y - f(x)\right)^2\right]}{\partial f(x)} = -\left(f(x) - y\right) = y - f(x)$$



Gradient Boosting(GBM)

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:

   (a) For $i = 1, 2, \ldots, N$ compute

   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$.

   (c) For $j = 1, 2, \ldots, J_m$ compute

   $$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.

3. Output $\hat{f}(x) = f_M(x)$.

→ r_{im}: pseudo-residual
→ 미분 가능한 Loss 함수 사용

Ensemble models

- RandomForest
- ExtraTrees
- Adaboost
- GradientBoost
  - XGBoost: 성능이 가장 우수하나 시간이 오래 걸림
  - LightGBM: 시간이 적게 걸림
  - CatBoost: categorical variable에 강점

수고하셨습니다!

해당 세션자료는 KUBIG Github에서 보실 수 있습니다!