17기 정규세션

ToBig's 16기 강의자
김건우

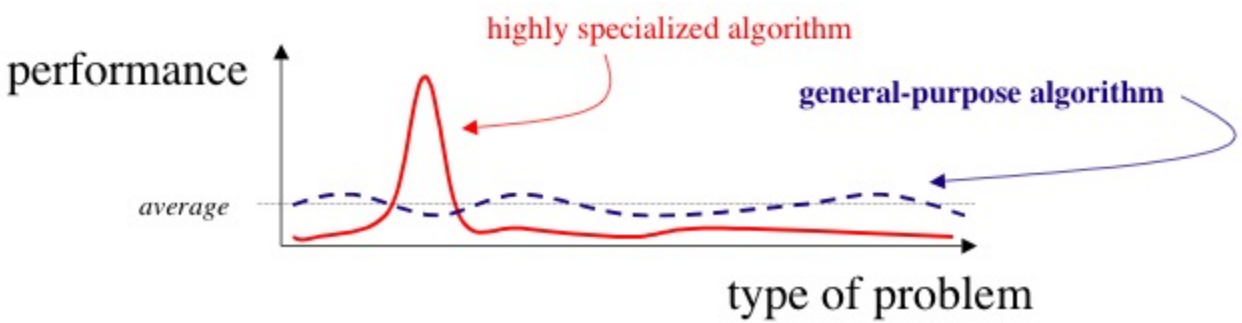
Ensemble

Contents

Unit 01 - Introduction

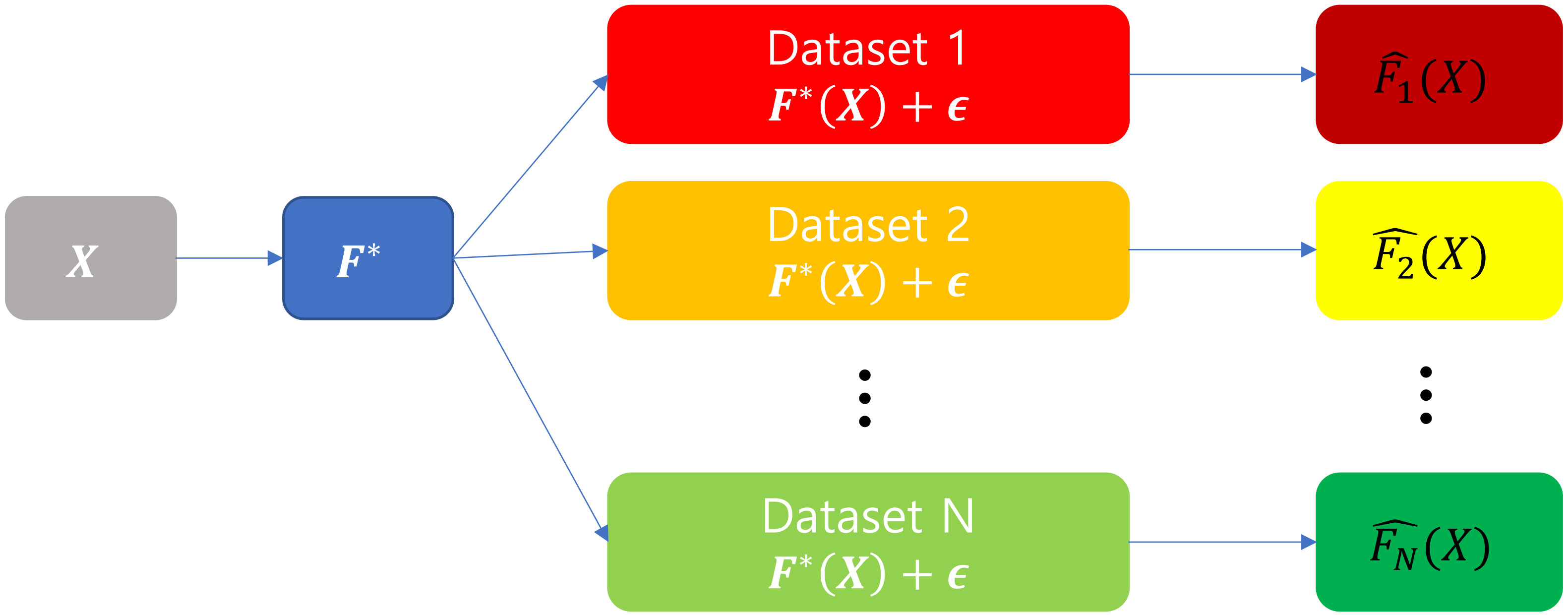
No Free Lunch Theorem (NFLT)

"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)



Bias-Variance Decomposition

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



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

Bias-Variance Decomposition

Calculate Error (MSE)

$$Err(X_0) = E\left[y - \hat{F}(X) \middle| X = X_0\right]^2$$

$$= E\left[F^*(X_0) + \epsilon - \hat{F}(X_0)\right]^2$$

$$= E\left[F^*(X_0) - \hat{F}(X_0)\right]^2 + \sigma^2$$

$$= E\left[F^*(X_0) - \bar{F}(X_0) + \bar{F}(X_0) - \hat{F}(X_0)\right]^2 + \sigma^2$$

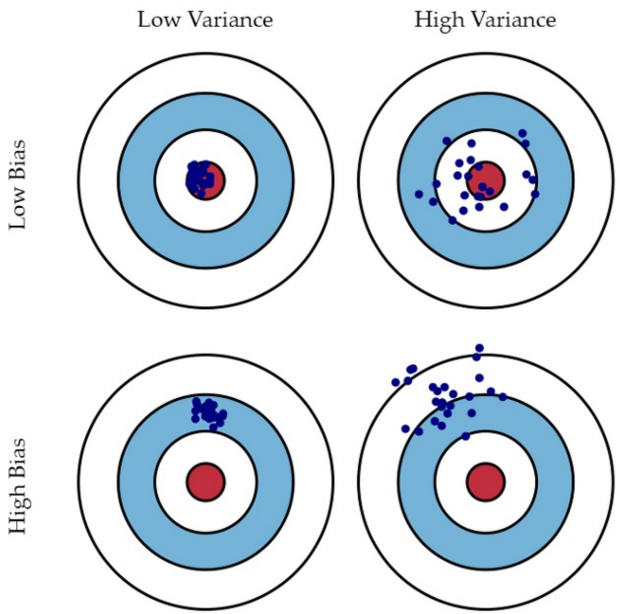
$$= [F^*(X_0) - \bar{F}(X_0)]^2 + \left[\bar{F}(X_0) - \hat{F}(X_0)\right]^2 + \sigma^2$$

$$= bias^2 + varinace + \sigma^2$$

Bias-Variance Decomposition

✓ 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

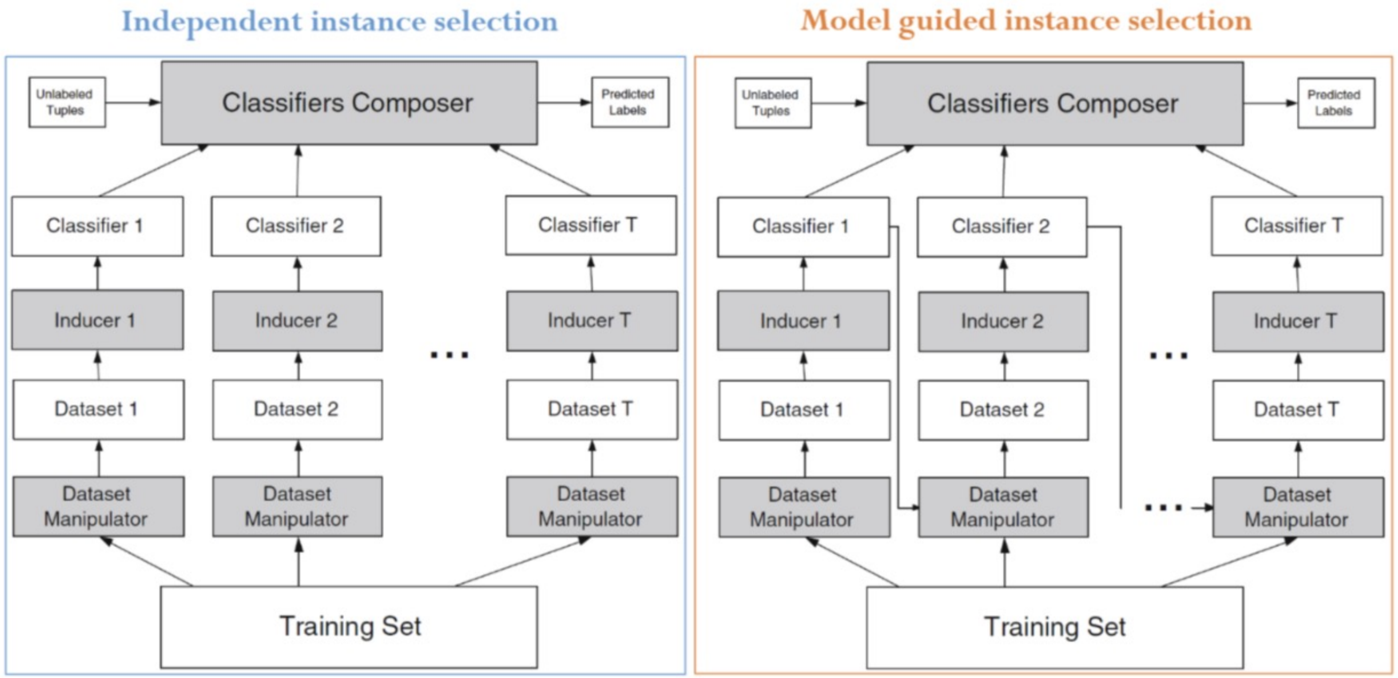


❖ High Bias + Low Variance: Logistic Regression, LDA, KNN
❖ Low Bias + High Variance: ANN, SVM, DT

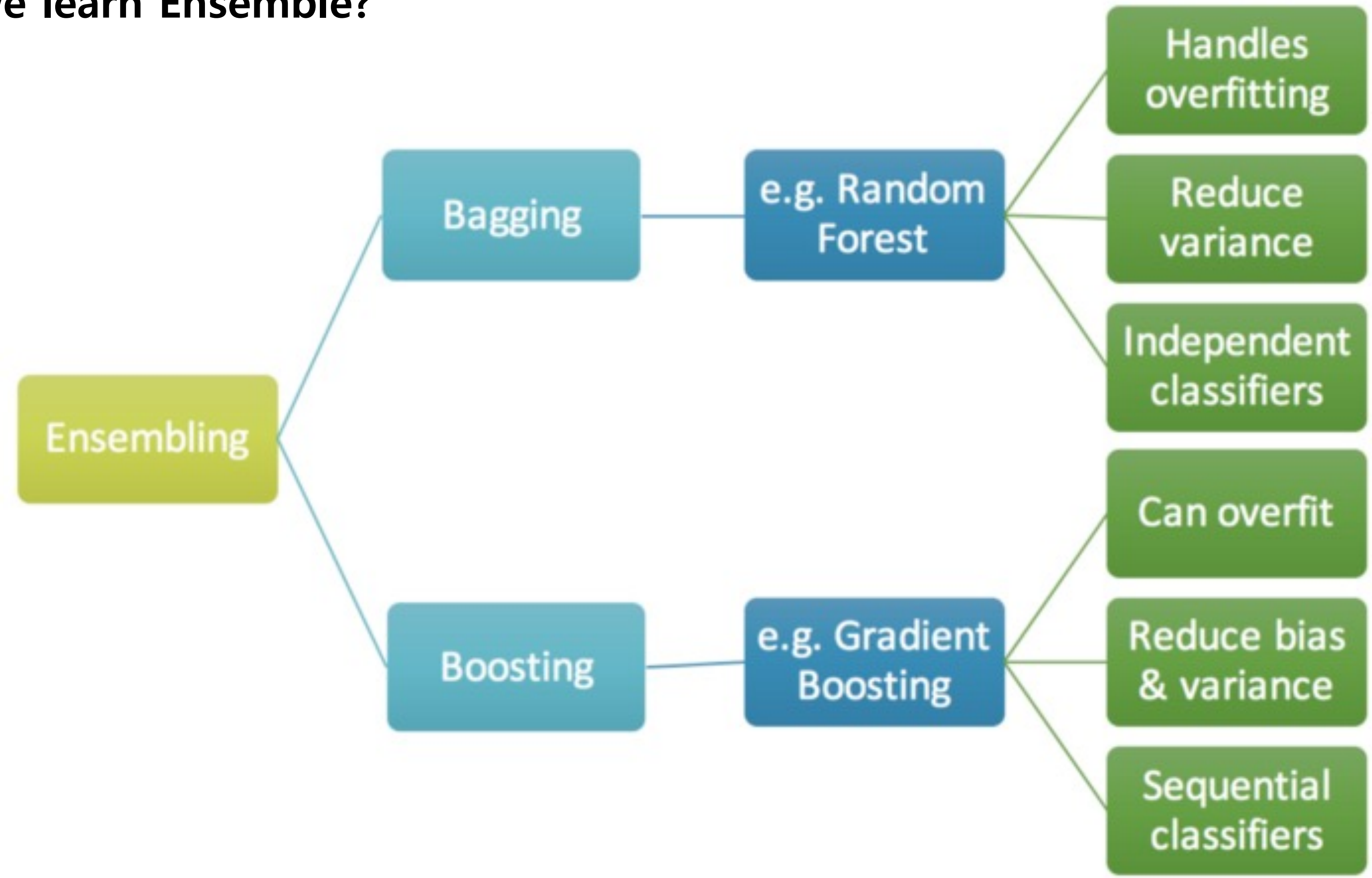
❖ Low Bias + Low Variance: Best Model!!!

Why do we learn Ensemble?

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



Why do we learn Ensemble?



Why do we learn Ensemble?

$$y_m = f(x) + \epsilon_m(x)$$

$$(y_m : Estimated\ Value, f(x) : True\ Function, \epsilon_m : Expected\ Error)$$

Average Error made by M models:

$$E_{Avg} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_x[\{y_m(x) - f(x)\}^2] = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_x[\epsilon_m(x)^2]$$

Expected Error of the Ensemble:

$$\frac{1}{M} * M * f(x)$$

$$E_{Ensemble} = \mathbb{E}_x\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(x) - f(x)\right\}^2\right] = \mathbb{E}_x[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(x)\right\}^2]$$

Why do we learn Ensemble?

$$\textit{if we assume,}\quad \mathbb{E}_x\left[\epsilon_m(x)\right] = 0, \quad \mathbb{E}_x\left[\epsilon_m(x)\epsilon_l(x)\right] = 0\ (m \neq l)$$

$$E_{Ensemble} = \mathbb{E}_x\left[\left\{\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(x)\right\}^2\right] = \frac{1}{M^2}\mathbb{E}_x\left[\left\{\sum_{m=1}^{M}\epsilon_m(x)\right\}^2\right]$$

$$E_{Avg} = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_x\left[\epsilon_m(x)^2\right]$$

$$E_{Ensemble} = \frac{1}{M}E_{Avg}$$

$$\textit{In real, by using Cauchy's Inequaility such that,}\quad E_{Ensemble} \leq E_{Avg}$$

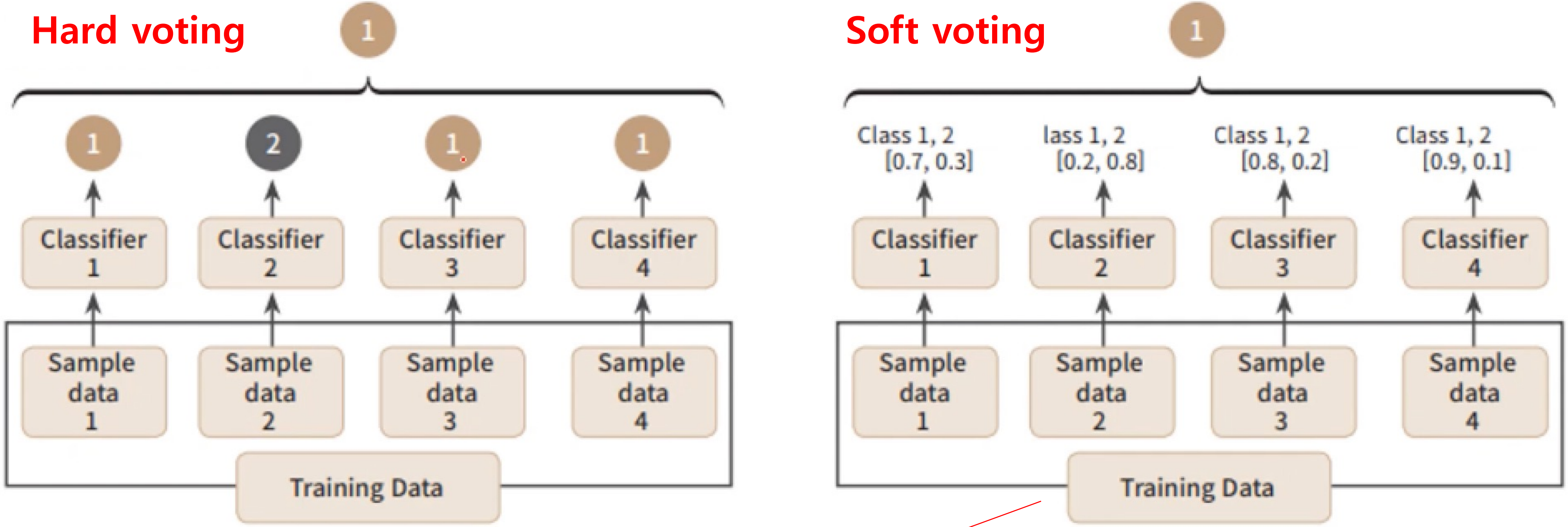
Unit 02 - Voting

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.**

Voting



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

Unit 03 - 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.

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

Bagging (Bootstrap Aggregating)

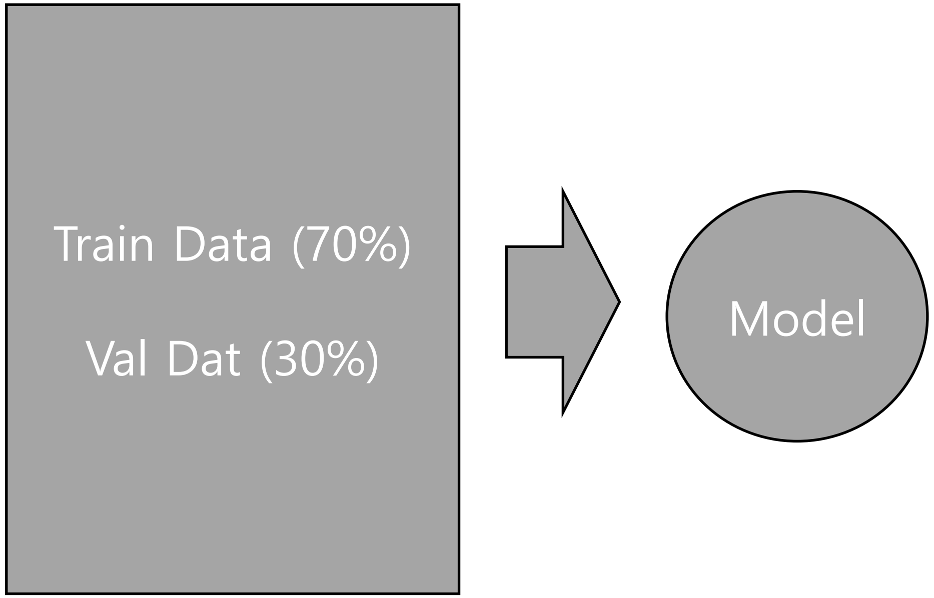
$$p = \left(1 - \frac{1}{N}\right)^{N} \longrightarrow \lim_{N \to \infty} \left(1 - \frac{1}{N}\right)^{N} \longrightarrow e^{-1} = 0.368 = 36.8\%$$

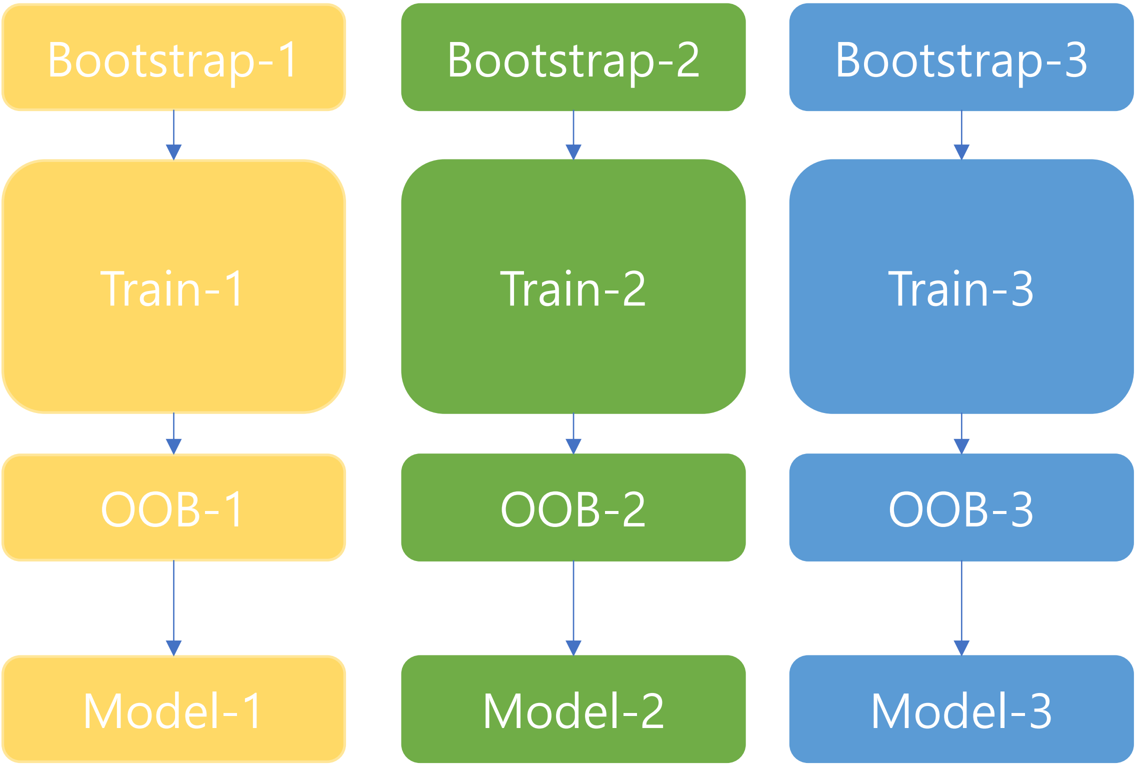
Out Of Bag data (OOB)

$$1 - p = 1 - 0.368 = 73.2\%$$

Sampled more than once in bootstrap

Bagging (Bootstrap Aggregating)



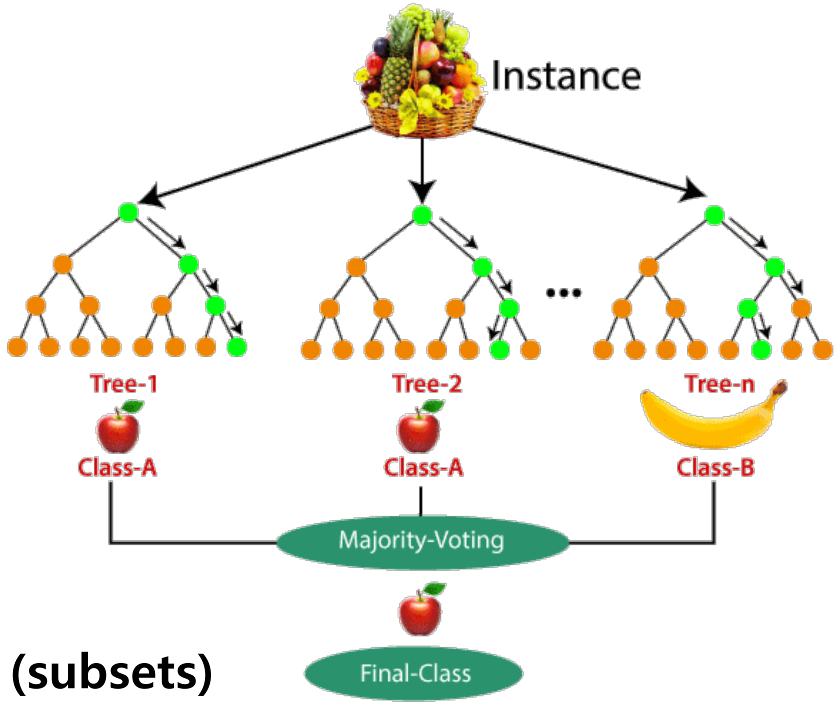


Random Forest

- A specialized bagging for DT (base learner)

  1) Based on Bagging Ensemble
  2) Randomly choose variables

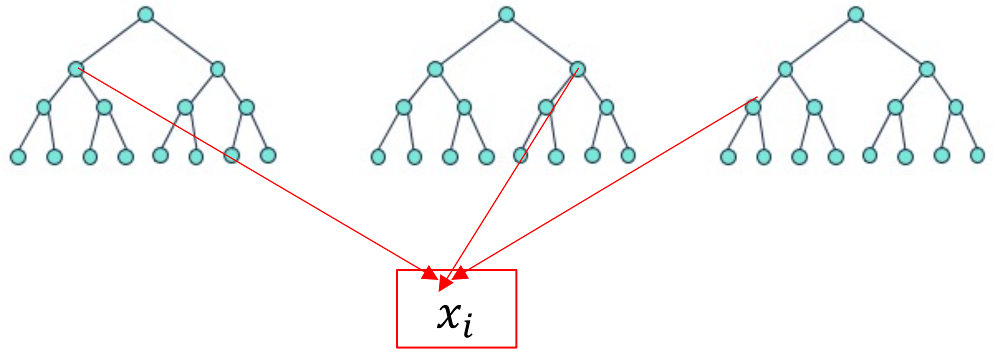
Randomly select 'm' variables



- 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

Random Forest

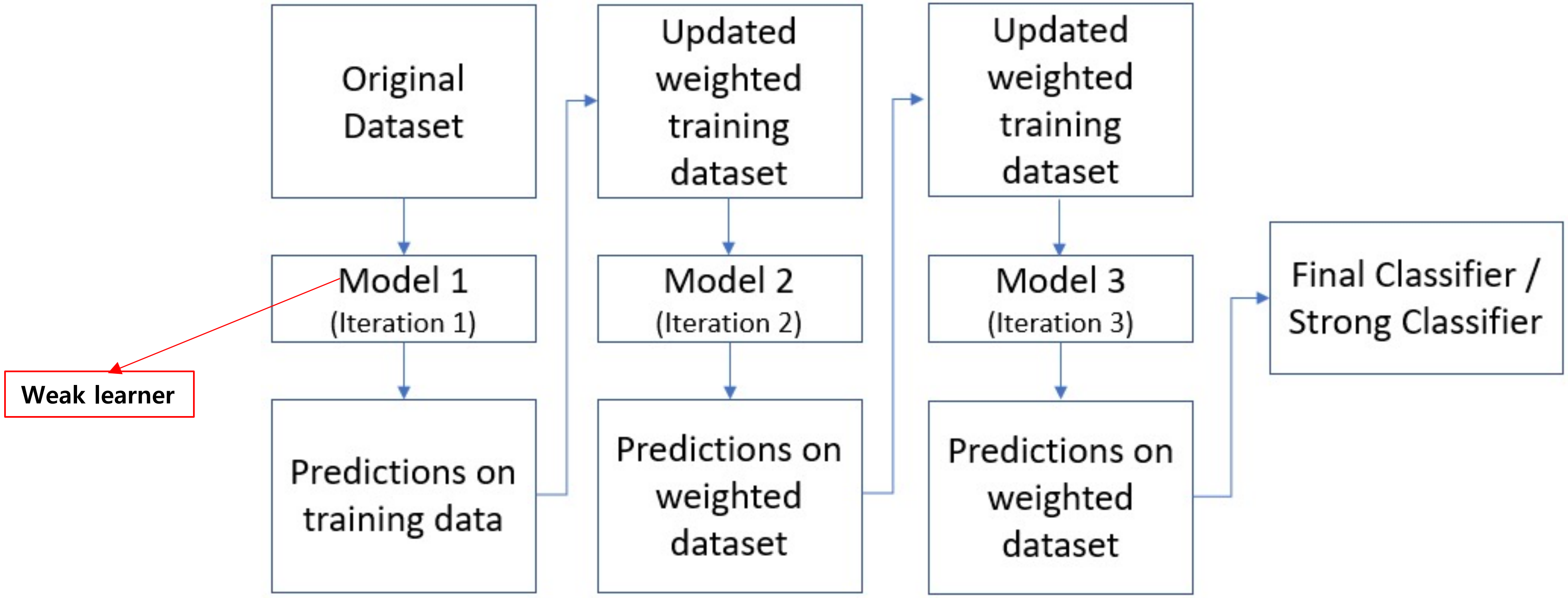
- Able to use on both classification and regression tasks
- Handle ‘Missing Values’ well
- Each tree (base learner) in random forest may overfit the data because ‘pruning‘ is not conducted
- Prevents overfitting problem
- Variable Importance --→ Not suggest which variables to select
- Difficult to interpret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



Unit 04 - Boosting

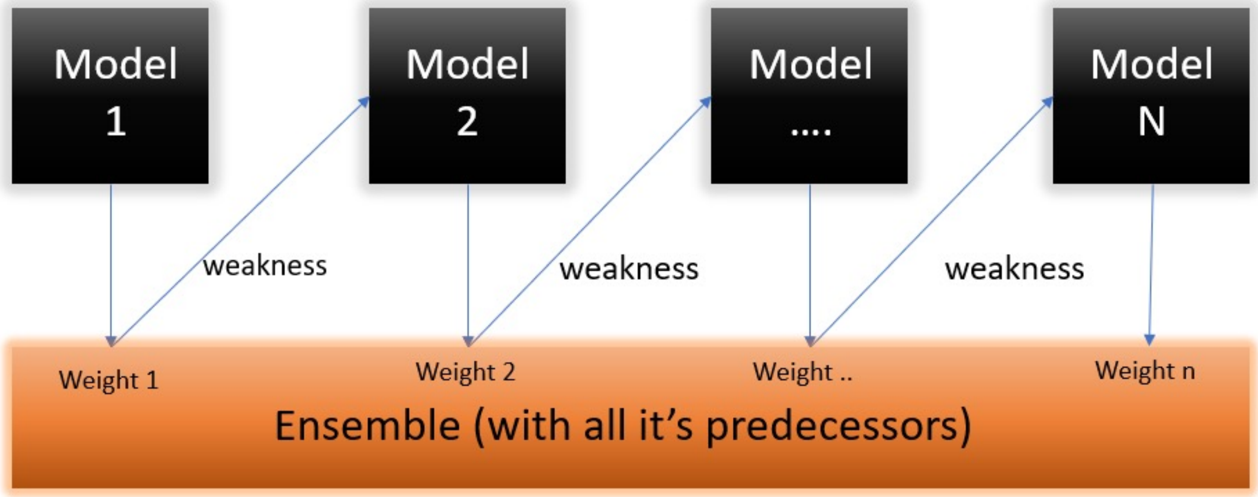
Boosting: An iterative procedure to adaptively change distribution of training data by focusing more on previously mis-classified records.

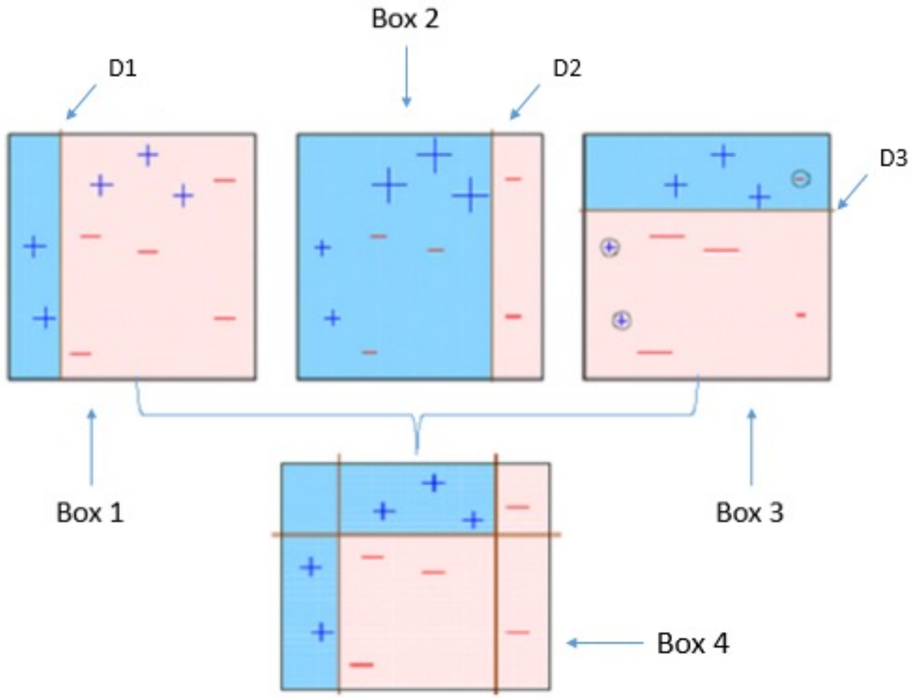
- **Sequential learning process unlike 'Bagging' (parallel learning)**



AdaBoost (Adaptive Boosting)

- **Weak learner, performing only slightly better than random guessing, could be boosted in to arbitrarily accurate strong learner**





AdaBoost (Adaptive Boosting)

Algorithm 2 Adaboost

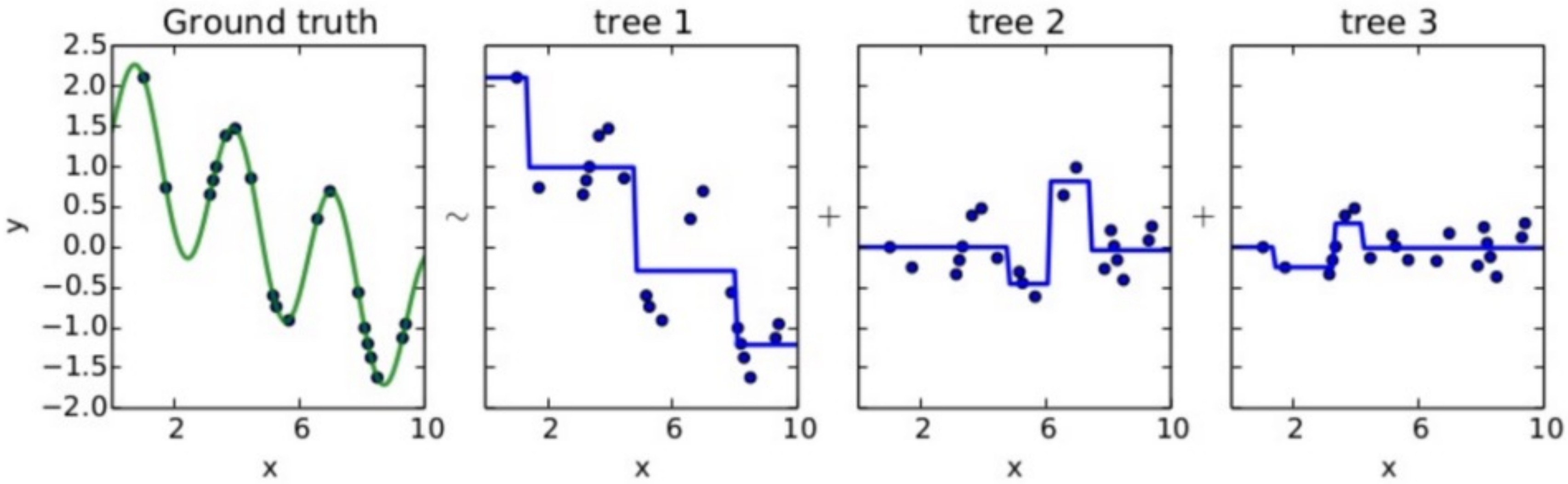
Input: Required ensemble size T
Input: Training set $S = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where $y_i \in \{-1, +1\}$
Define a uniform distribution $D_1(i)$ over elements of S.
for $t = 1$ to T **do**
  Train a model h_t using distribution D_t.
  Calculate $\epsilon_t = P_{D_t}(h_t(x) \neq y)$
  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'),
$H(x') = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x') \right)$

GBM (Gradient Boosting Machine)

✓ **AdaBoost: update weights on Dataset by sampling**

✓ **GBM: update ‘y’ not Dataset**

- **Understand the concept of ‘Residual Fitting’**



GBM (Gradient Boosting Machine)

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

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

$$Residual = y_i - f(x_i)$$

$$Residual = Negative\ Gradient$$

$$GBM = Gradient\ Descent + Boosting$$

GBM (Gradient Boosting Machine)

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:

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

2. For m = 1 to M:

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

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

   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 γ_m by solving the following one-dimensional optimization problem:

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

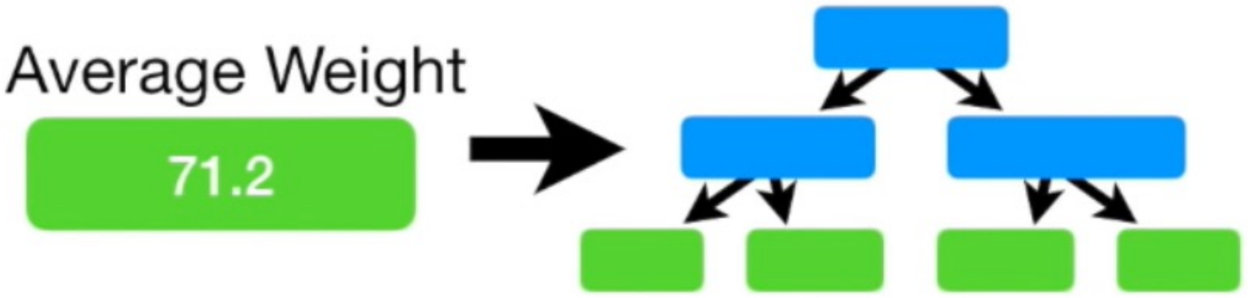
   4. Update the model:

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

3. Output $F_M(x)$.

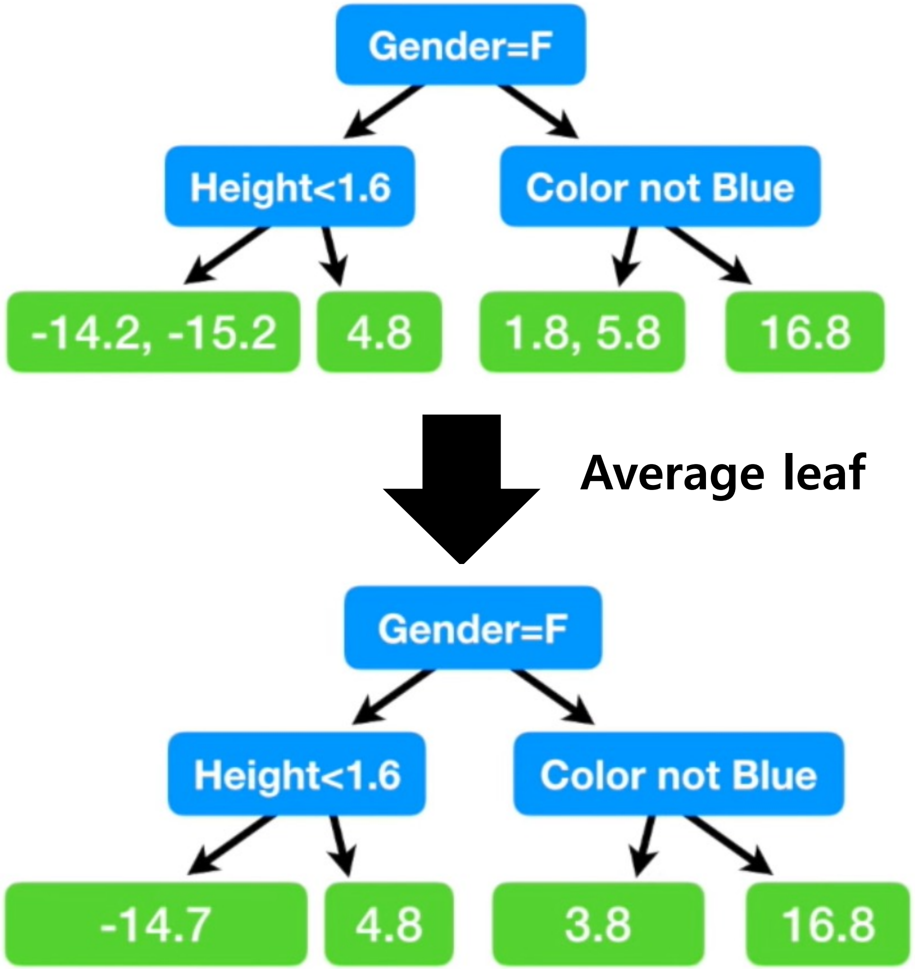
GBM (Gradient Boosting Machine)

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



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



GBM (Gradient Boosting Machine)

…so the **Predicted Weight** = 71.2 + 16.8 = 88

Learning Rate = 0.1

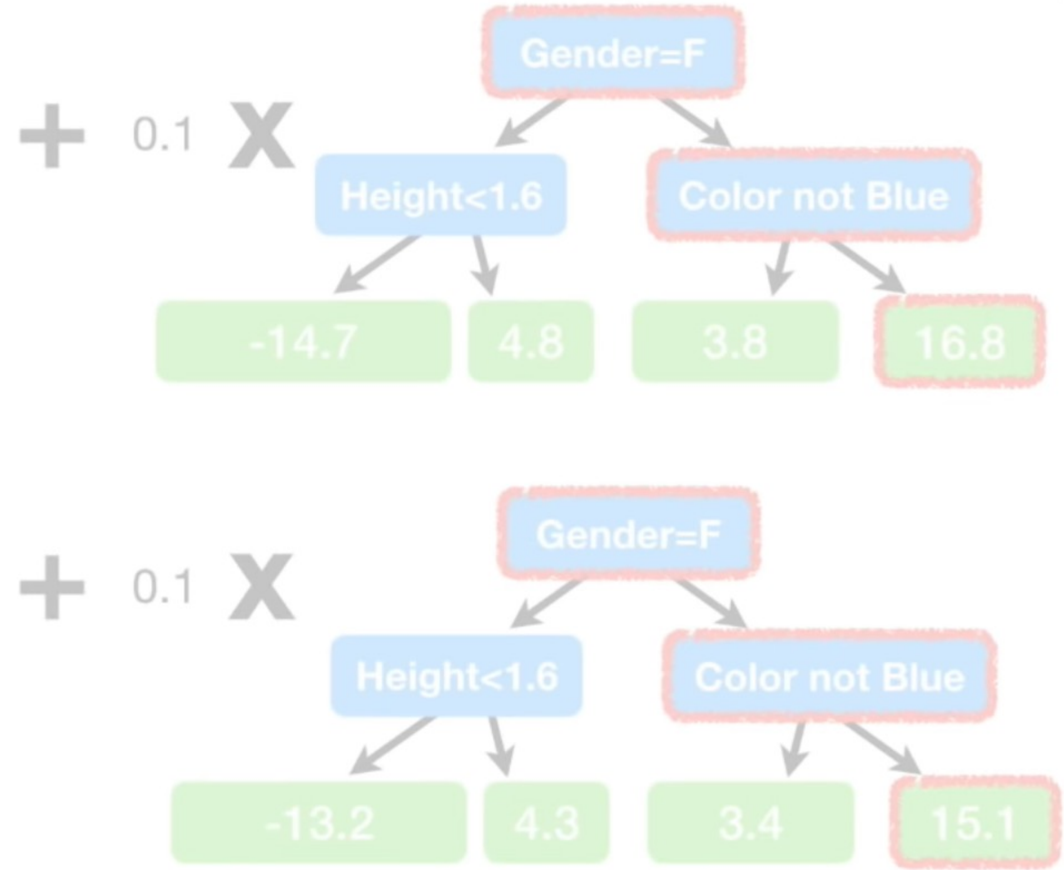
GBM (Gradient Boosting Machine)

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

GBM (Gradient Boosting Machine)

Average Weight

71.2

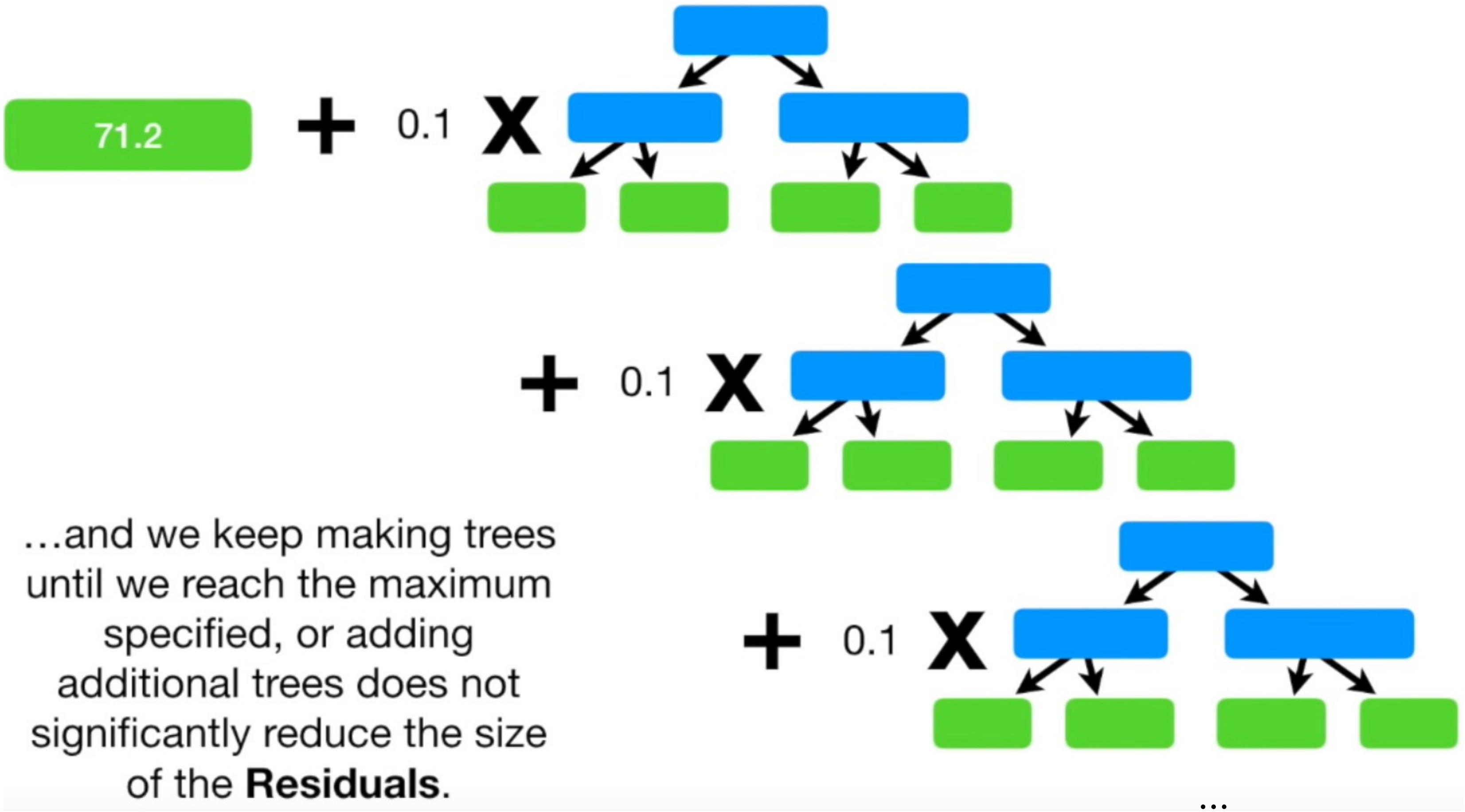


That gives us…

$$71.2 + (0.1 \times 16.8) + (0.1 \times 15.1)$$

$$\boxed{= 74.4}$$

GBM (Gradient Boosting Machine)



…

…

Boosting

kaggle

dmlc XGBoost

LightGBM

CatBoost

…

…

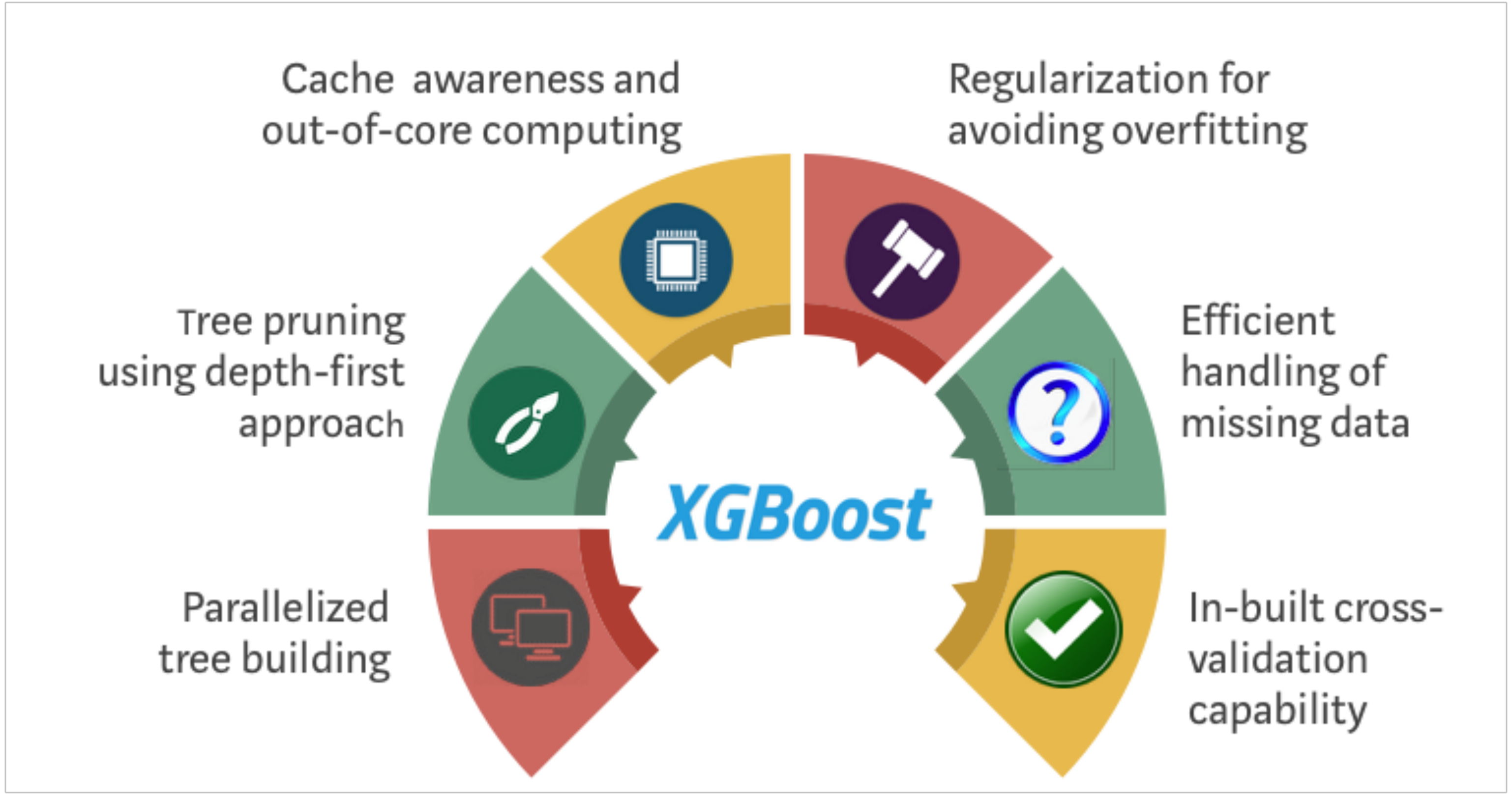
XGBoost (eXtreme Gradient Boosting)

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

- **Optimization**
    1. **Parallelziation**
    2. **Tree Pruning**

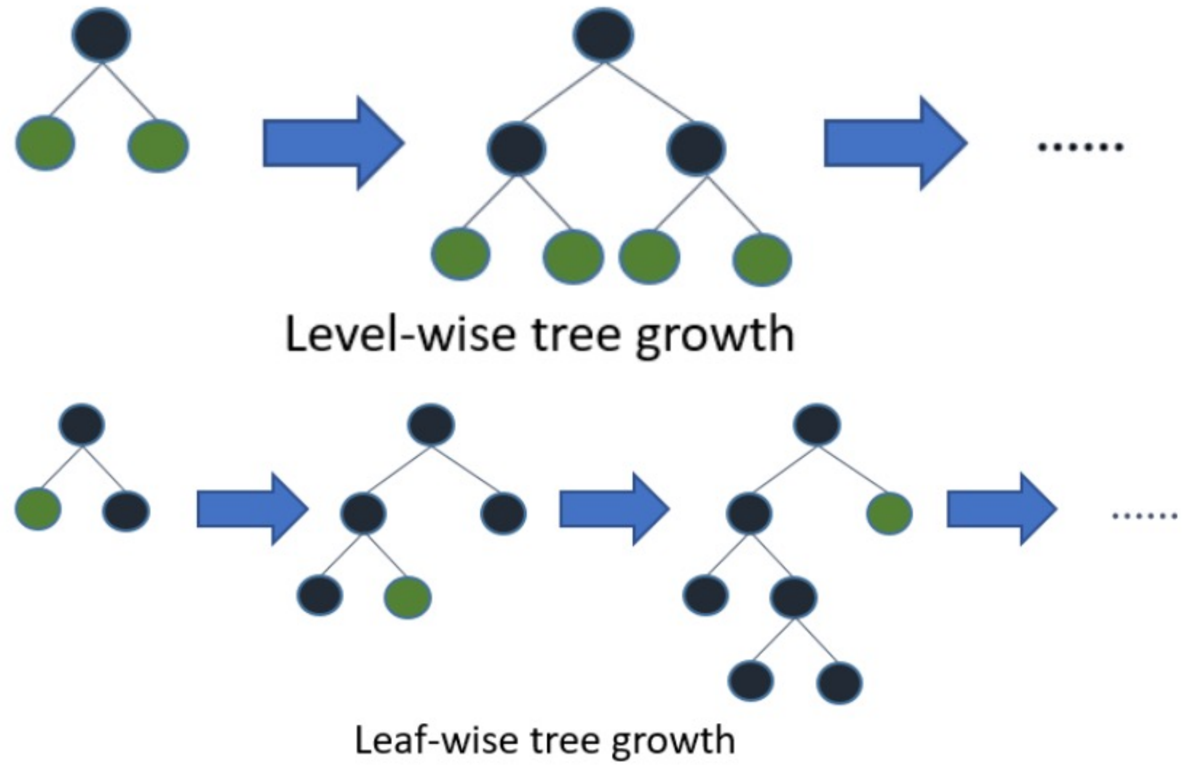
- **Algorithm**
    1. **Regularization**
    2. **Built-in Cross Validation**
    3. **Sparsity Awareness**

- **Hardware Optimization**



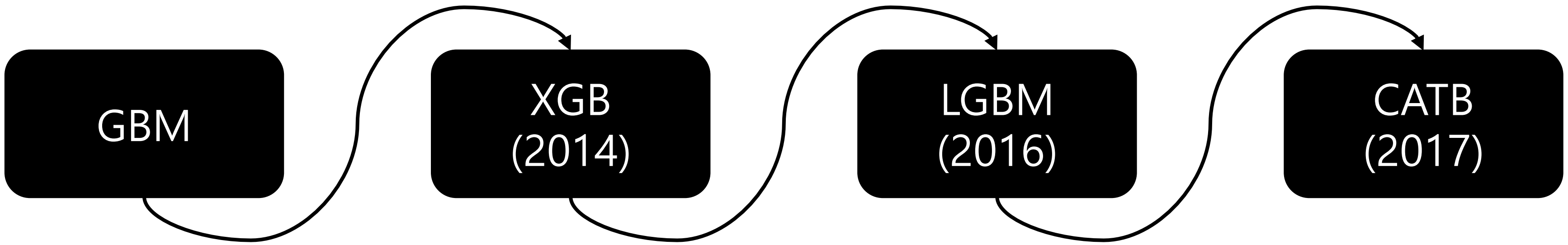
Light GBM (Light Gradient Boosting Machine)

- **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 $\times$ len(I) , randN $\leftarrow$ b $\times$ len(I)
for $i = 1$ ***to*** d **do**
  preds $\leftarrow$ models.predict(I)
  g $\leftarrow$ $loss$(I, preds), w $\leftarrow$ {1,1,...}
  sorted $\leftarrow$ GetSortedIndices(abs(g))
  topSet $\leftarrow$ sorted[1:topN]
  randSet $\leftarrow$ RandomPick(sorted[topN: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)

Boosting



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

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

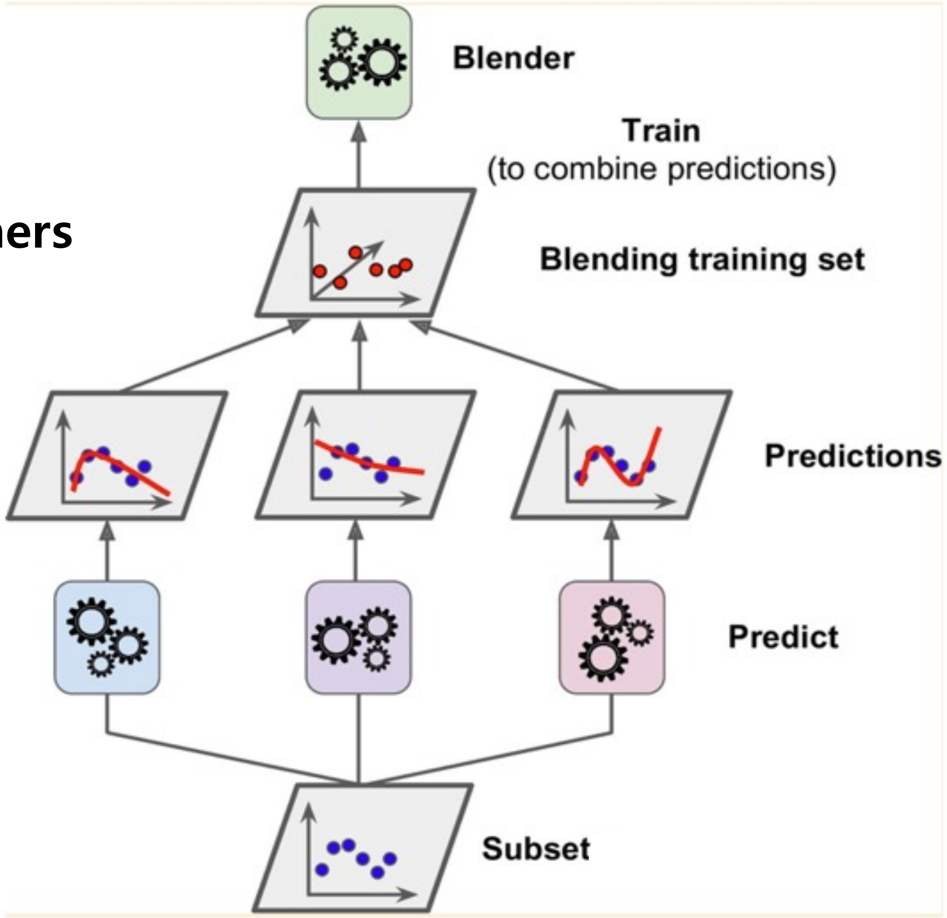
- Handle categorical variables well
- Slower than LGBM (training)
- Low performance (most variables are numeric, not categoric)

Unit 05 - Stacking

Stacking

✓ **Use Meta-Learner to aggregate the results**

- **Meta-Learner's Input: predicted values from base-learners**
- **Meta-Learner's Output: actual true labels**



Stacking Example

\# of data: 569
\# of features: 30

Train data shape: (455,30) - 80% of data
Test data shape: (114,30) – 20% of data

Weak Learners: XGBoost(XGB), Light GBM(LGBM), SVM, ANN (Use models with high complexity)

Concatenate prediction values from weak learners: XGB_pred, LGBM_pred, SVM_pred, ANN_pred
-> shape (114,4)

Meta Learner: Logistic Regression (Use a model with low complexity)

Meta Learner fitting: Use concatenated values (114,4) as X, use (114,) as Y
Final output: (114,)

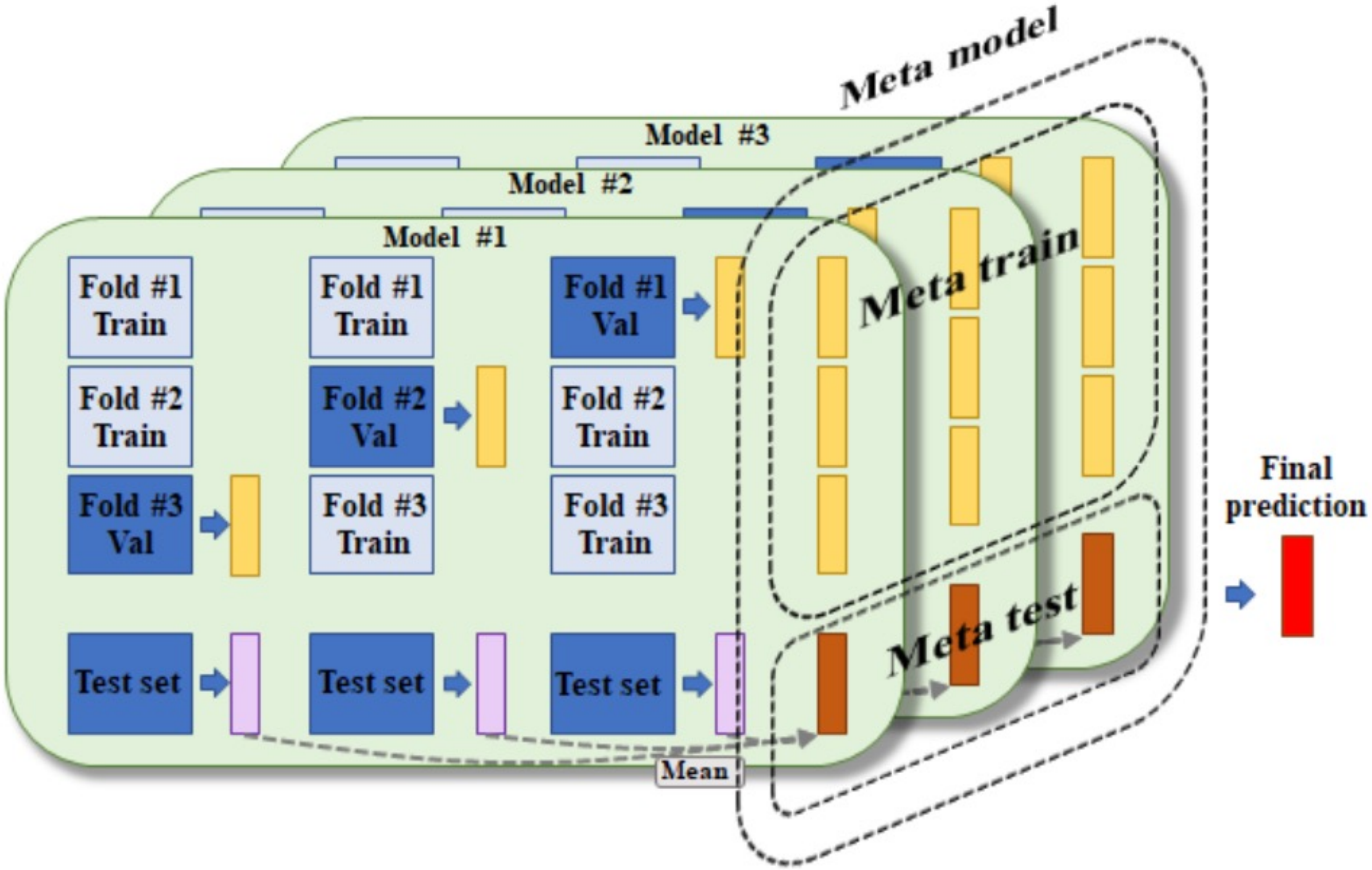
Stacking

- **Pros: Improve performance**
- **Cons: Overfitting**

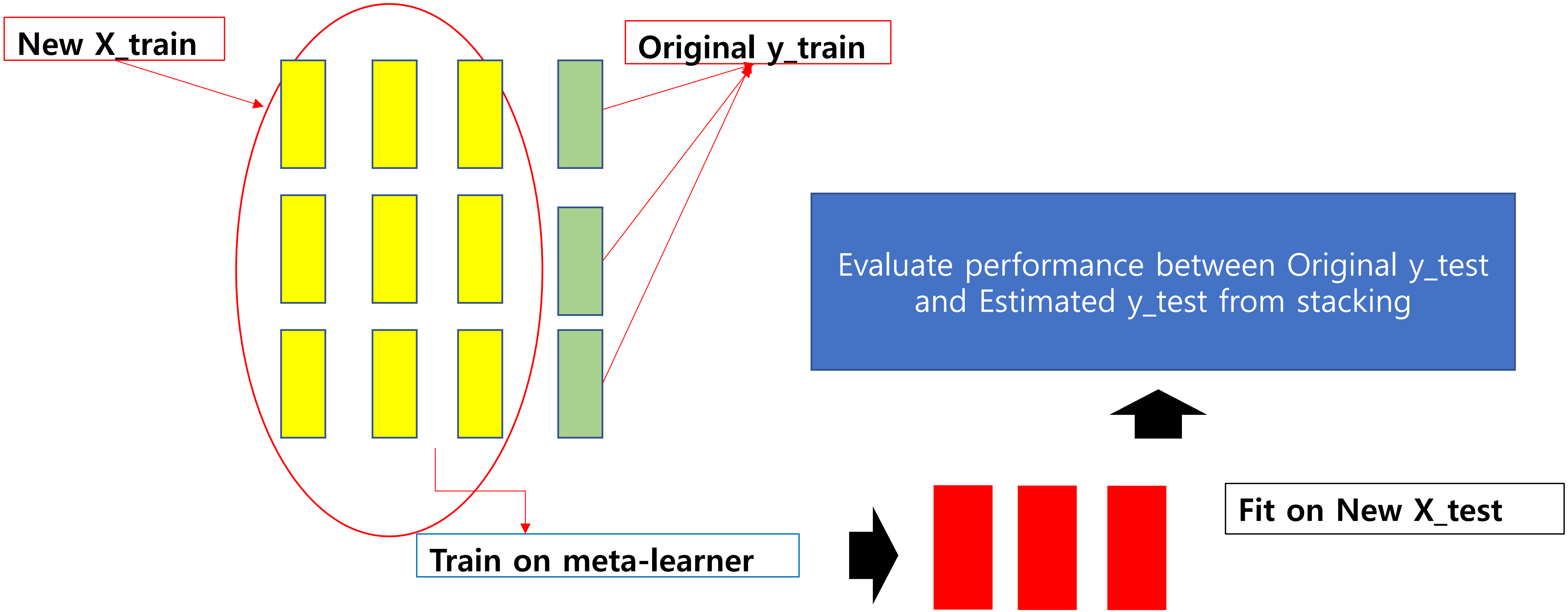
How to solve Overfitting problem?

✓ **Stacking based on Cross Validation**

Stacking based on Cross Validation



Stacking based on Cross Validation



Conclusion

- ✓ **Ensemble = "Diversity"**
- ✓ **Voting (Hard Voting vs Soft Voting)**
- ✓ **Bagging (Random Forest)**
- ✓ **Boosting (AdaBoost, GBM, XGB, LightGBM, CatBoost)**
- ✓ **Stacking (Stacking based on CV)**

Assignment

- Kaggle Competition에 참여하여 가장 좋은 Model 을 만들어 보세요!

- 채점 기준은 다음과 같습니다.
  1. Leaderboard score
  2. EDA
  3. 모델의 결과에 대한 설명

- 2주차에 배운 Hyperparameter Tuning과 1~5주차에 배운 다양한 모델과 기법들을 활용해보세요!

- **Baseline Model의 performance를 모두 넘어야 합니다!!!**

✓ (Kaggle Link)
https://www.kaggle.com/t/933534784e1b4c71abc4918ad97d5271

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/UCPq01cgCcEwhXl7BvcwIQyg
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.