

정 규 세션 4 주차

ToBig's 10기 이준걸

# Ensemble

Bagging, Boosting, XGBOOST, LightGBM, Stacking

# 들어가기 전..

수식이 많고 내용이 많이 어려웠지만 정말 배운게 많은 강의였네요. 개인적으로 너무 준비하시느라 힘드셨을 것 같 은데 좋은 강의 해주셔서 너무 감사드립니다.

내용면에서는 왜 배워야하는지 알려주셔서 더욱 의미 있었던 것 같네요. 또한 강의구성에 대한 큰 그림을 이해할 수 있도록 세세하게 설명해주셔서 더욱 재미있게 들을 수 있었습니다.

질문이 많이 생겼는데 하나하나 대답해주시는 걸 보면서 새삼 내공에 감탄했습니다. 개인적으로 더 공부해보고 싶 은데 참고할만한 강의/사이트 있으시면 공유 탁드릴게요!

2시간이 너무 빠르게 지나갔다. 너무 재밌었음

덕분에 어려운 내용이 재밌게 다가왔습니다. 갓준결님 교수님해주세요ㅠㅠ 학부생으로 들어갈게요

# 들어가기 전..

실습 때 너무 빠른 감이 있었습니다ㅠ 코드 결과 보다 어떤 의미인지 풀기하기 힘들었어요

뒷부분이 살짝 어려웠지만 그래도 강의 정말 좋았습니다.

너무 어렵다. 그래도 코드실습을 통해 어려운 개념이지만 코드는 한 줄인게 그나마 위로가 되었다

어려웠습니다ㅠㅠ 그리고 양이 조금 많아 처음보는 이 많은 내용들을 머릿속에 넣기 버거웠습니다. 그래도 실습 코드를 직접 보면서 코드에 대한 설명을 들으니 어떻게 활용되는지 이해가 쉬웠고 좋았습니다! 실습 코드가 체계적으로 준비된 점이 지금껏 들었던 강의중에 제일 만족하는 부분입니다.

앙상블은 조금 어려웠습니다... 강의가 길어져서 중간에 집중력도 조금 흐려졌던 것 같습니다  
강의가 길어지면 중간에 한 5분정도 쉬는것도 좋을것 같습니다~

어려운 내용을 개념 위주로 최대한 이해할 수 있게 설명해주시고, 실습해볼 수 있어서 좋았습니다! 다만 강의 ppt가  
틀린 부분이 몇개 있었는데 그 부분이 많이 헛갈려요ㅠㅠ

# 들어가기 전..

실습 때 너무 빠른 감이 있었습니다ㅠ 코드 결과 보다 어떤 의미인지 풀기하기 힘들었어요

-> 실습 때 느리게 진행하  
겠습니다.

어려운 내용을 개념 위주로 최대한 이해할 수 있게 설명해주시고, 실습해볼 수 있어서 좋았습니다! 다만 강의 ppt가  
틀린 부분이 몇개 있었는데 그 부분이 많이 헷갈려요ㅠㅠ

-> 오타 정정하였습니다.

뒷부분이 살짝 어려웠지만 그래도 강의 정말 좋았습니다.

너무 어렵다. 그래도 코드실습을 통해 어려운 개념이지만 코드는 한 줄인게 그나마 위로가 되었다

어려웠습니다ㅠㅠ 그리고 양이 조금 많아 처음보는 이 많은 내용들을 머릿속에 넣기 버거웠습니다. 그래도 실습 코드를 직접 보면서 코드에 대한 설명을 들으니 어떻게 활용되는지 이해가 쉬웠고 좋았습니다! 실습 코드가 체계적으로 준비된 점이 지금껏 들었던 강의중에 제일 만족하는 부분입니다.

양상불은 조금 어려웠습니다... 강의가 길어져서 중간에 집중력도 조금 흐려졌던 것 같습니다

강의가 길어지면 중간에 한 5분정도 쉬는것도 좋을것 같습니다~

-> 원래 어려운 내용이나 최대한 쉽게  
강의하겠습니다. 양은 어쩔 수 없을거  
같습니다.

-> 길어지면 Boosting 끝나고 5분 쉬  
고 진행하겠습니다.

# 들어가기 전..

- 강의시간은 2시간 정도 소요 될 것 같습니다.
- 중간중간에 개념이 끝나자마자 실습이 진행됩니다. 따라서 제공한 모든 코드를 틀어놓는 것을 권장드립니다.
- 배우는 내용은 크게 3가지(Bagging, Boosting, Gradient Boosting)이고 구체적인 알고리즘은 5개(Random Forest, Adaboost, Gradient Boosting, XGBoost, LightGBM)입니다.
- 꼭 복습 부탁드립니다!

# Content

---

Unit 01 | Ensemble Model Overview & Voting Classifier

---

Unit 02 | Bagging, Boosting

---

Unit 03 | AdaBoost, Gradient Boosting

---

Unit 04 | XGBoosting, GBM & LightGBM

---

Unit 05 | Stacking

---

## Unit 01 | Ensemble Model OverView

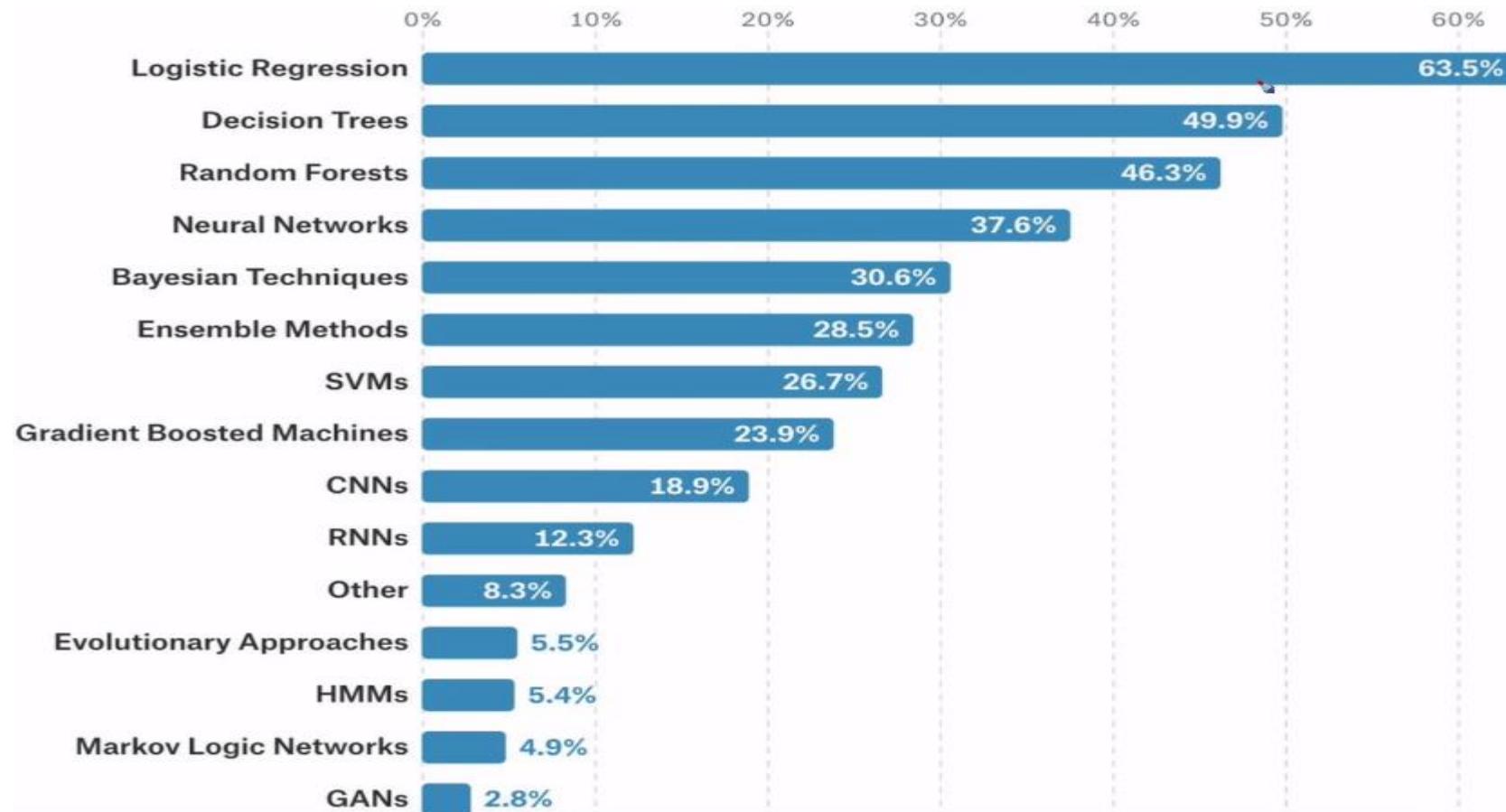
# no free lunch Theorem

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 Theorem')

-> 간단히 말해 특정한 문제에 최적화된 알고리즘은 다른 문제에서 는 그렇지 않다는 것을 수학적으로 증명한 정리이다.

## Unit 01 | Ensemble Model OverView

## Kaggle에서 많이 쓰이는 모델들



## Unit 01 | Ensemble Model OverView

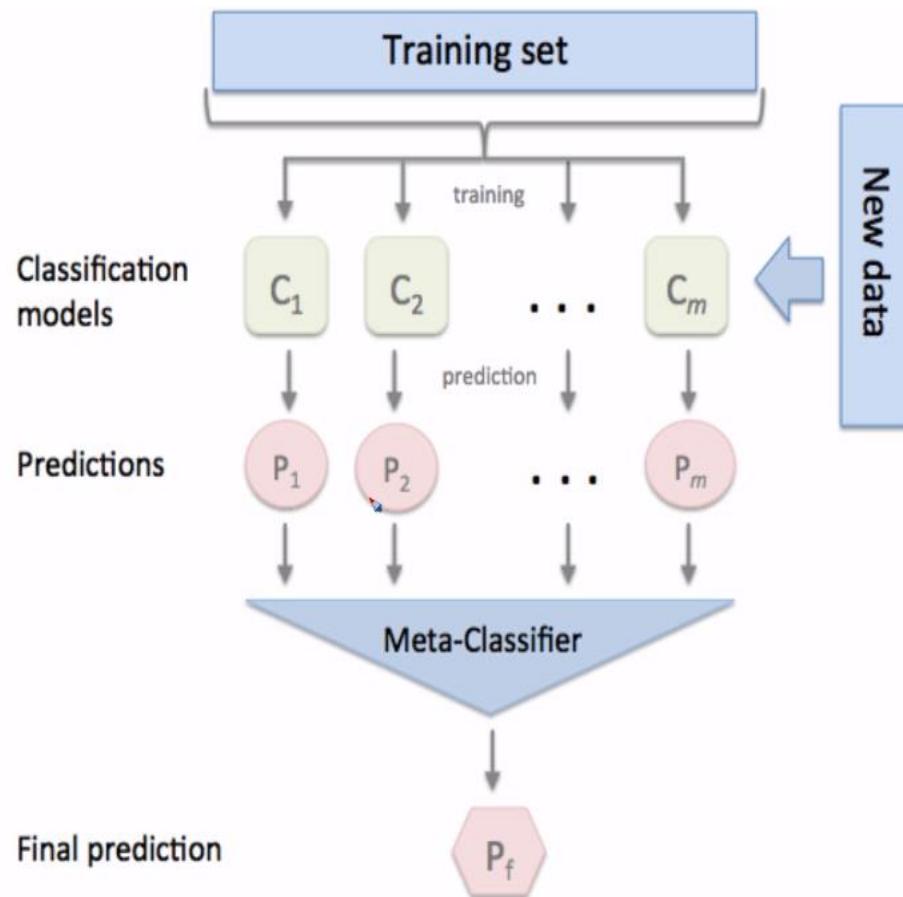
## Ensemble Model OverView

- 하나의 모델이 아니라 여러 개의 모델의 투표로 Y값 예측
- Regression 문제에서는 평균값으로 예측
- Meta – Classifier : 여러 모델의 연계를 통해 문제 해결
- Stacking(Meta – ensemble) : 앙상블을 여러 개를 섞어서 연결해 문제 해결
- 학습은 오래 걸리나 성능이 매우 매우 좋음!
- Kaggle에서 Structured Dataset에서의 대세 기법



# Unit 01 | Ensemble Model OverView

## Meta - Classifier



ex1) 모두 다른 모델

C1 : Logistic Regression

C2 : KNN

C3 : SVM

...

Cm : DT

ex2) 모두 같은 모델

(단 hyperparameter는 다름)

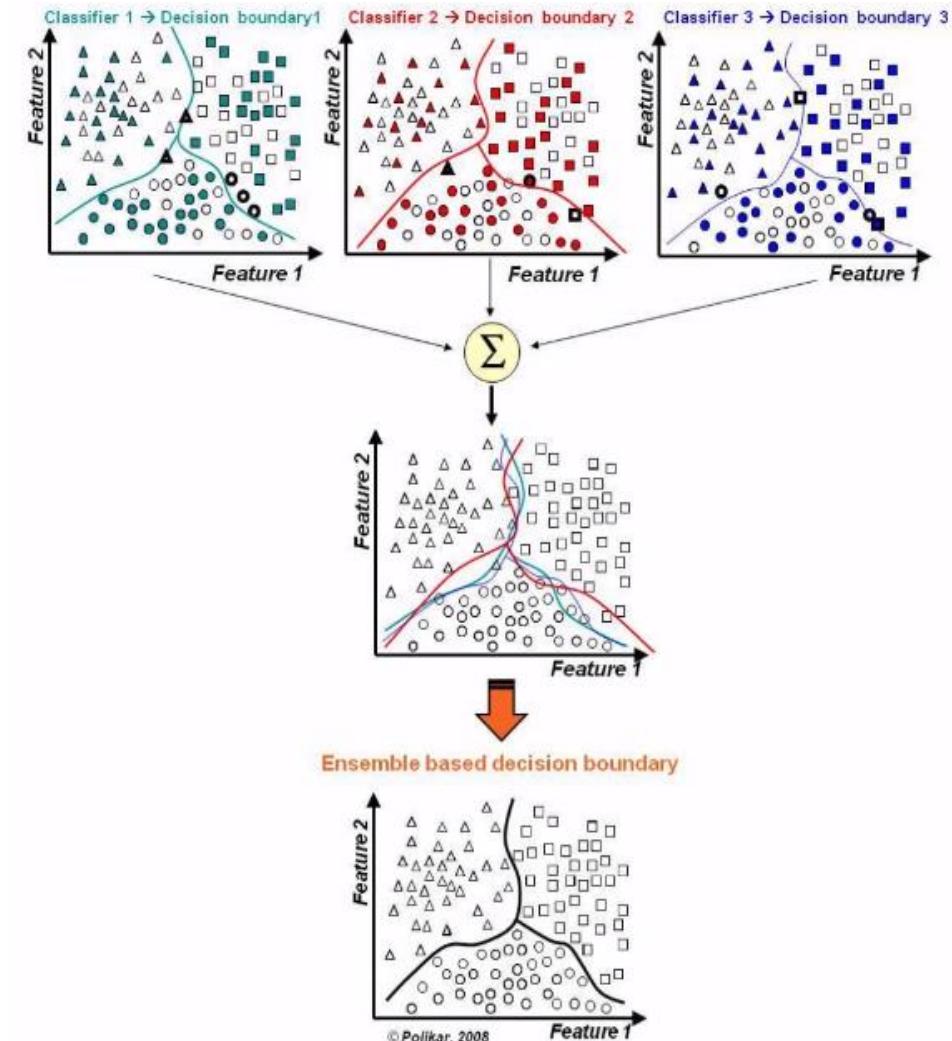
C1 : DT

C2 : DT

C3 : DT

...

Cm : DT



## Unit 01 | Ensemble Model OverView

### 오늘 배울 내용은?

- Vanilla Ensemble : 기본적인 Ensemble(분류 – Voting, 회귀 – Mean, Median)
- Bagging : Data Sampling 1
- Boosting : Data Sampling 2
- Adaptive Boosting (AdaBoost)
- XGBoost
- Light GBM



Boosting의 발전된 Model

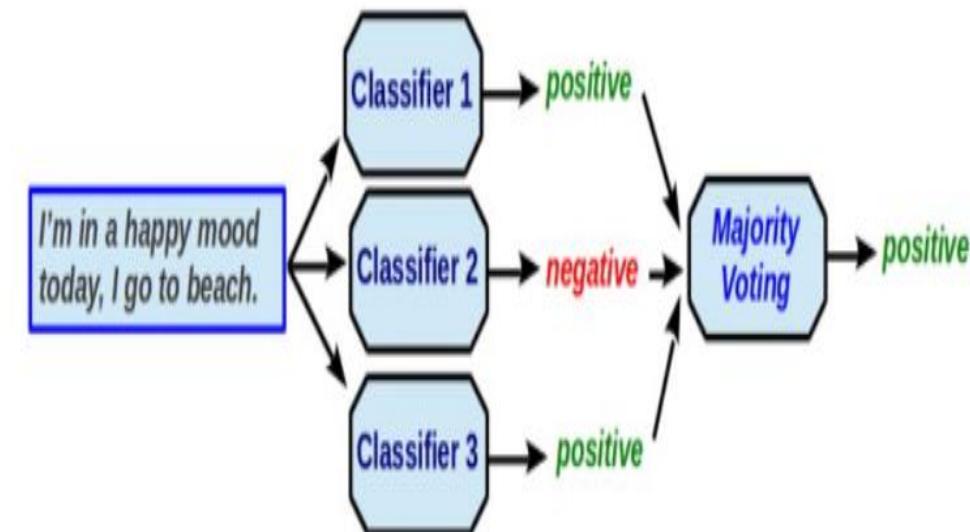
## Unit 01 | Voting Classifier

# Voting Classifier = Majority Voting = Vanllila Ensemble

- 가장 기본적인 Ensemble Classifier
- 여러 개의 Model의 Voting을 통해 예측

Code

```
clf1 = LogisticRegression(random_state = 1)
clf2 = DecisionTreeClassifier(random_state = 1)
clf3 = GaussianNB()
clf = VotingClassifier(
    Estimators = [('lr', clf1), ('dt', clf2), ('gnb', clf3)],
    Voting = 'hard')
```



## Unit 01 | Voting Classifier

[실습1-Voting\_Ensemble] 켜주세요!

---

Unit 01 | Ensemble Model Overview & Voting Classifier

---

Unit 02 | Bagging – Random Forest

---

Unit 03 | Boosting – AdaBoost, Gradient Boosting

---

Unit 04 | XGBoosting, GBM & LightGBM

---

Unit 05 | Stacking

---

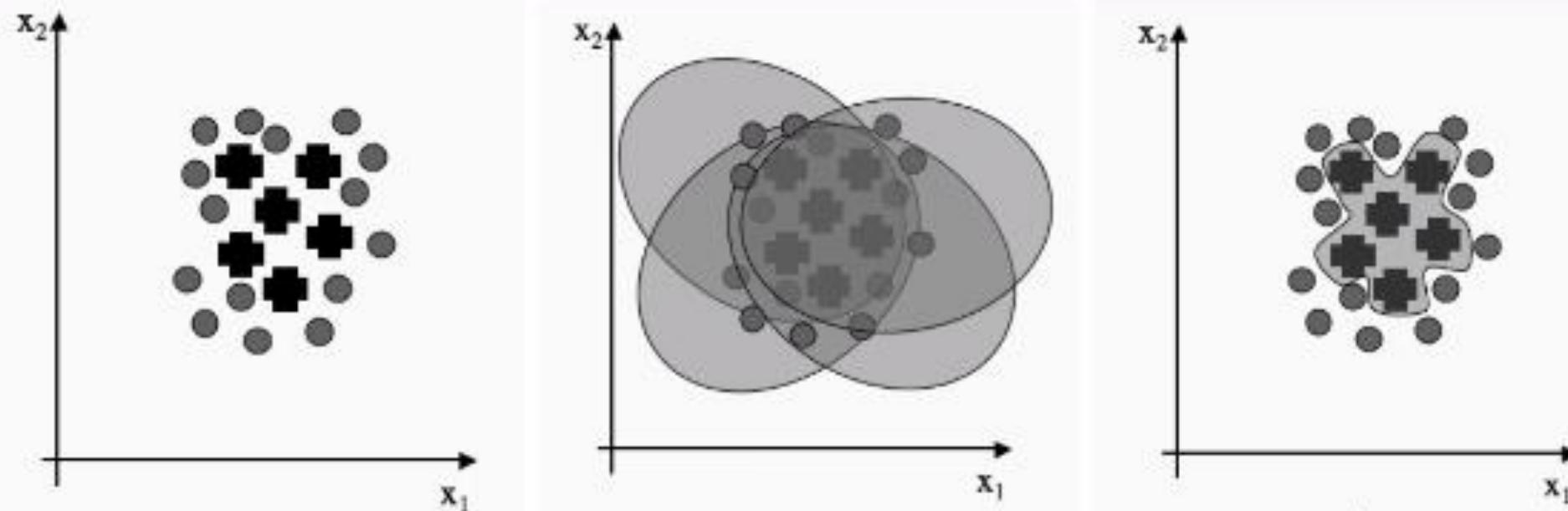
## Unit 02 | Bagging

# What is Bagging? – Sampling

- 단순히 같은 데이터 셋으로 만드는 Classifier는 의미가 없다. Ex) 같은 Dataset으로 만든 여러 개의 Tree를 생각해 보면 매우 비슷하게 만들어질 것이므로 성능에 영향을 미치지 않는다.
- 우리가 보고 있는 데이터는 매우 큰 부분의 일부에 지나지 않는다. 마찬가지로 우리가 갖고 있는 샘플 자체를 하나의 모집단이라고 생각을 하고 이 모집단을 샘플링을 해서 모델을 만든다면 매우 Robust한 모델을 만들 수 있다.
- 즉, 다양한 Sampling Dataset으로 다양한 Classifier를 만들어 보자!!

## Unit 02 | Bagging

## Ensemble Sampling



→ 분류기를 여러 개 모아서 학습을 시키면 강한 분류기를 만들 수 있다.  
이때 사용하는 여러 개의 분류기를 Base Model이라 하고 Decision Tree는 base model로써 활용도가 높다.

## Unit 02 | Bagging

# Why DT?

1. Low computational complexity : 데이터의 크기가 방대한 경우에도 Train 속도가 빠르다.
2. Nonparametric : 데이터 분포에 대한 전제가 필요하지 않음.
3. Variance : Variance의 조절이 매우 편리함( Depth를 깊게 쌓으면 Variance가 높아지고 얕게 쌓으면 낮아진다.)

## Unit 02 | Bagging

# Bootstrapping

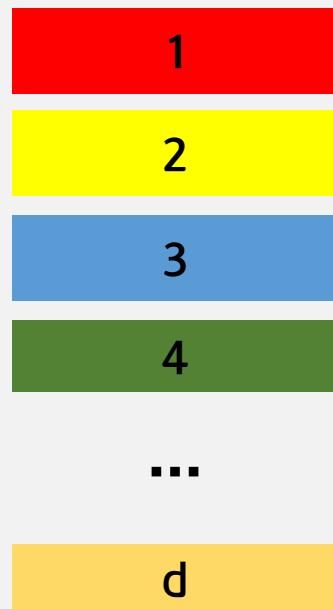
- 데이터를 외부 추가 없이 추출하는 것 -> 외부의 input 없이 구성된 데이터셋으로만 성능을 최대화 시키는 방법.
- 학습데이터 Subset을 구성하여 n개를 추출하자! Ex) 밑에는  $n = 3$



## Unit 02 | Bagging

# .632 Bootstrap

Observed Data



나눠진 subset을 d번 뽑는 시행을 있다고 하자.

한 데이터가 뽑힐 확률 :  $\frac{1}{d}$  / 뽑히지 않을 확률 :  $1 - \frac{1}{d}$

적어도 한번 뽑힐 확률 :

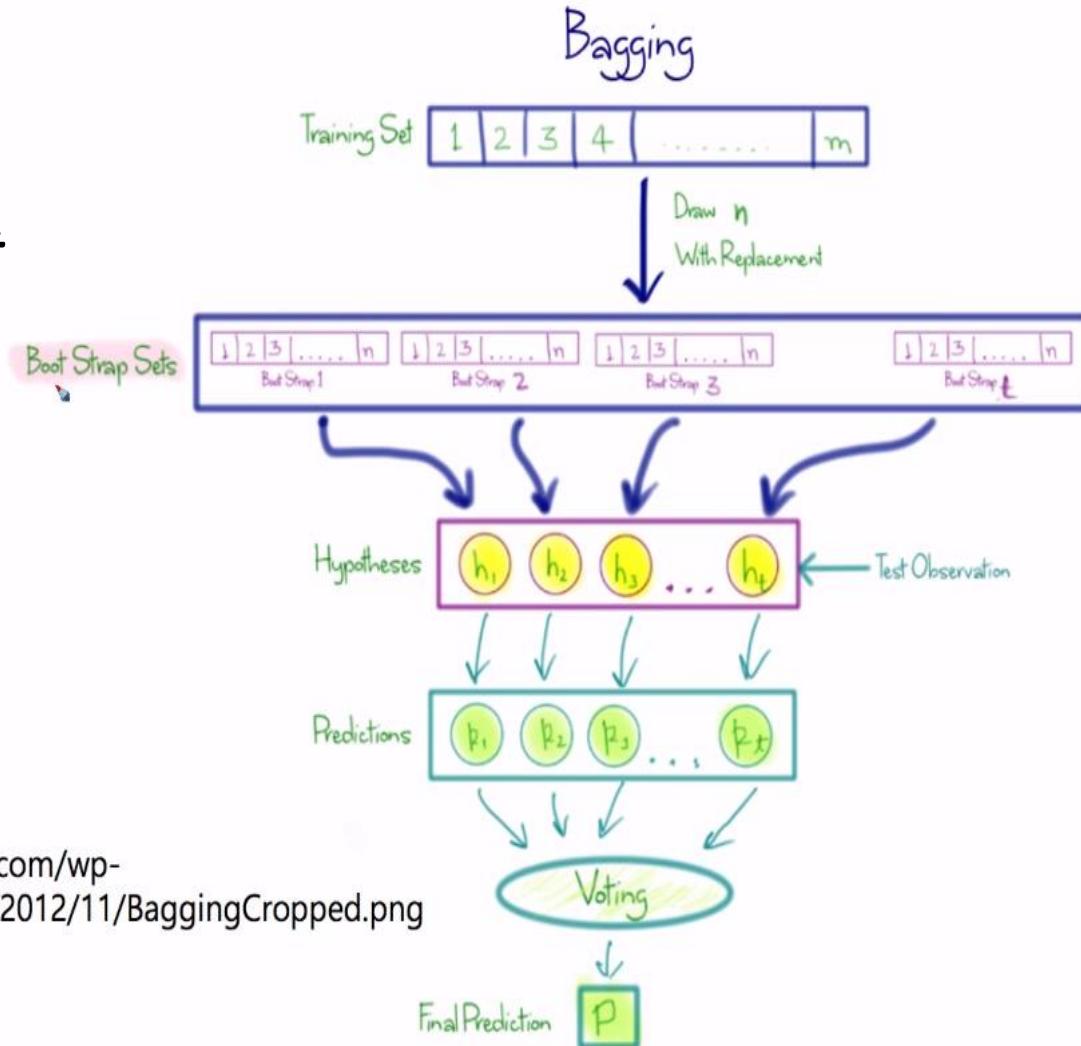
$$1 - \prod\left(1 - \frac{1}{d}\right) = 1 - \left(1 - \frac{1}{d}\right)^d \approx 1 - e^{-1}$$

$$= 0.632$$

## Unit 02 | Bagging

# Bagging

- Bootstrap의 Subset Sample로 **동일한** 모델 n개를 학습  
→ 양상불
- High Variance(Overfitting이 심함) 모델이 적합
- Regressor(평균 Or 중위수), Classifier 모두 존재



<http://manish-m.com/wp-content/uploads/2012/11/BaggingCropped.png>

## Unit 02 | Bagging

# Bagging = Bootstrap Aggregation

- Bootstrap의 Subset Sample로 **동일한** 모델 n개를 학습 → 양상을
- High Variance(Overfitting이 심함) 모델이 적합
- Regressor(평균 Or 중위수), Classifier 모두 존재



## Unit 02 | Bagging

# Out of Bag error = OOB error

- Bagging 실행 시, Bag에 미포함 데이터로 성능 측정
- Validation set, K-fold 와 처리하는 방법과 유사
- Bagging 성능 측정을 위한 굉장히 좋은 지표



## Unit 02 | Bagging

[실습2-Bagging] 켜주세요!

## Unit 02 | RandomForest

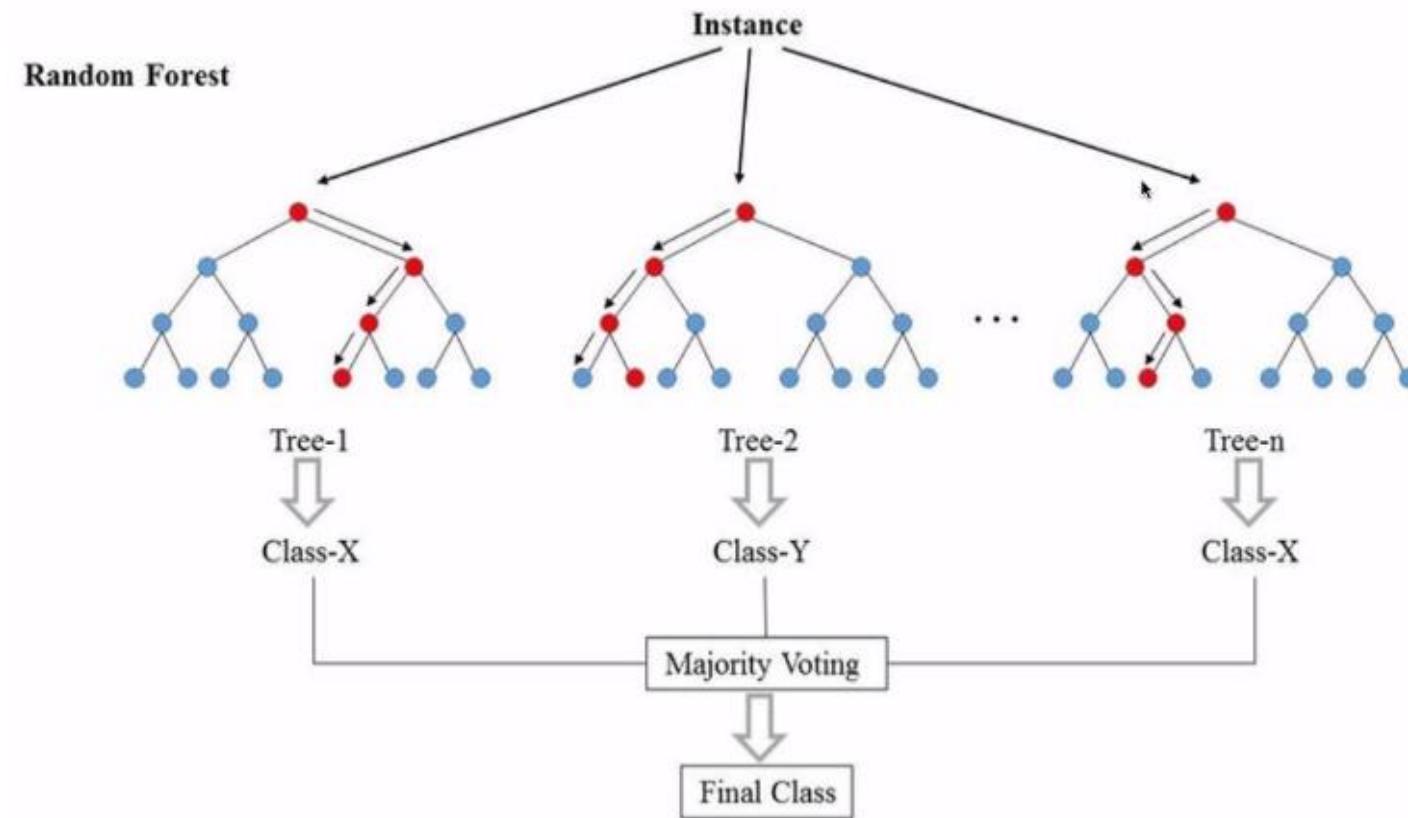
# What is Random Forest?

- Bagging + Randomized Decision Tree
- Variance가 높은 DT들의 Ensemble
- 관측치에 비해 변수의 수가 많은 고차원 데이터에서  
중요 변수 선택 기법으로도 활용됨.



## Unit 02 | Bagging

# Random Forest



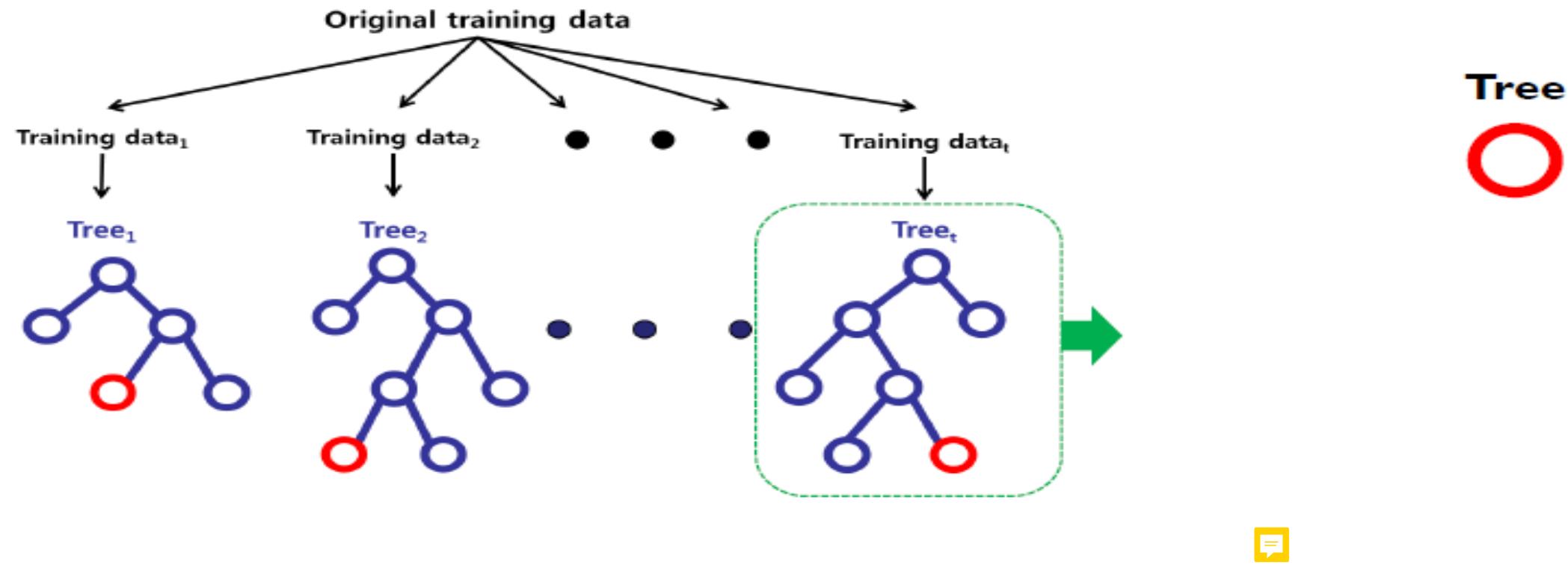
1. Bootstrap기법을 이용하여 다수의 Train data set을 구성
2. 생성된 training data로 DT 모델 구축
3. 예측 종합

## Unit 02 | RandomForest

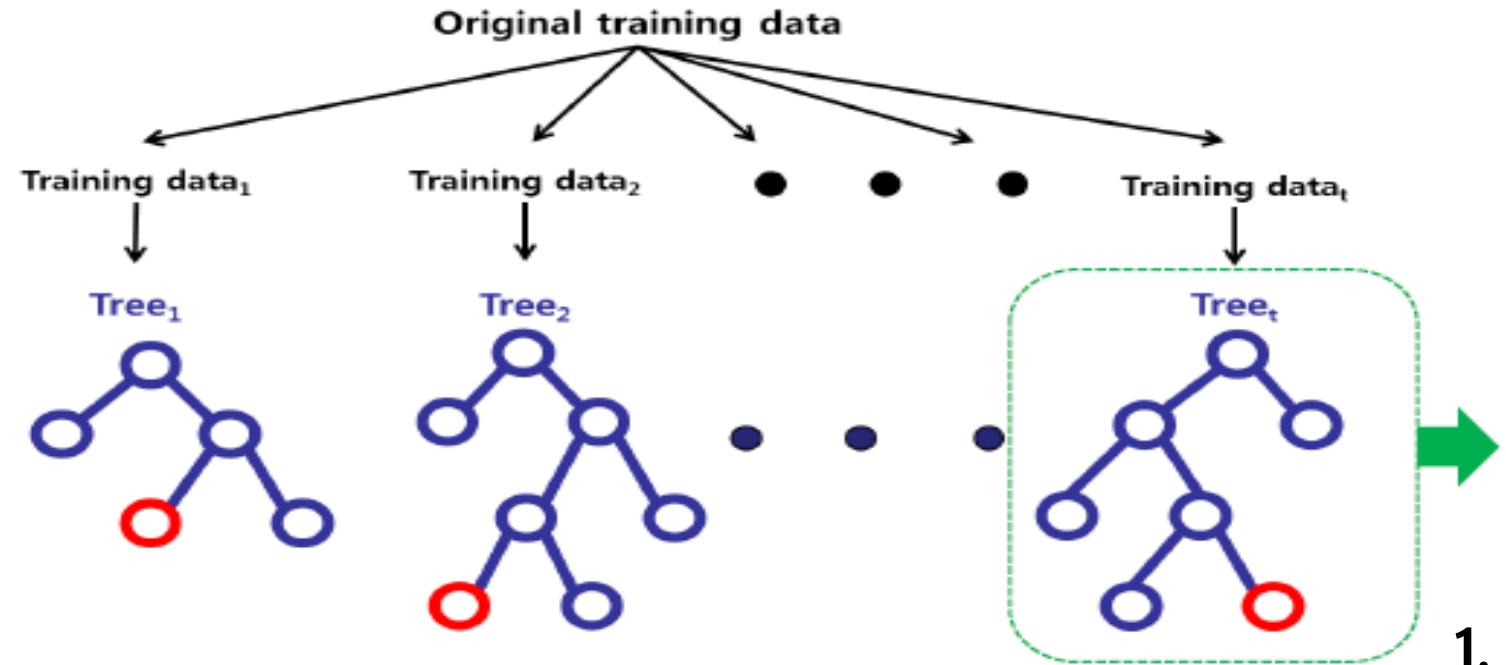
### Main idea : Diversity, Random

1. Diversity : 여러 개의 Training data를 생성하여 각 데이터마다 개별 DT 구축 -> Bagging
2. DT 구축 시 변수 무작위로 선택 -> Feature Random Selection

# Unit 02 | RandomForest



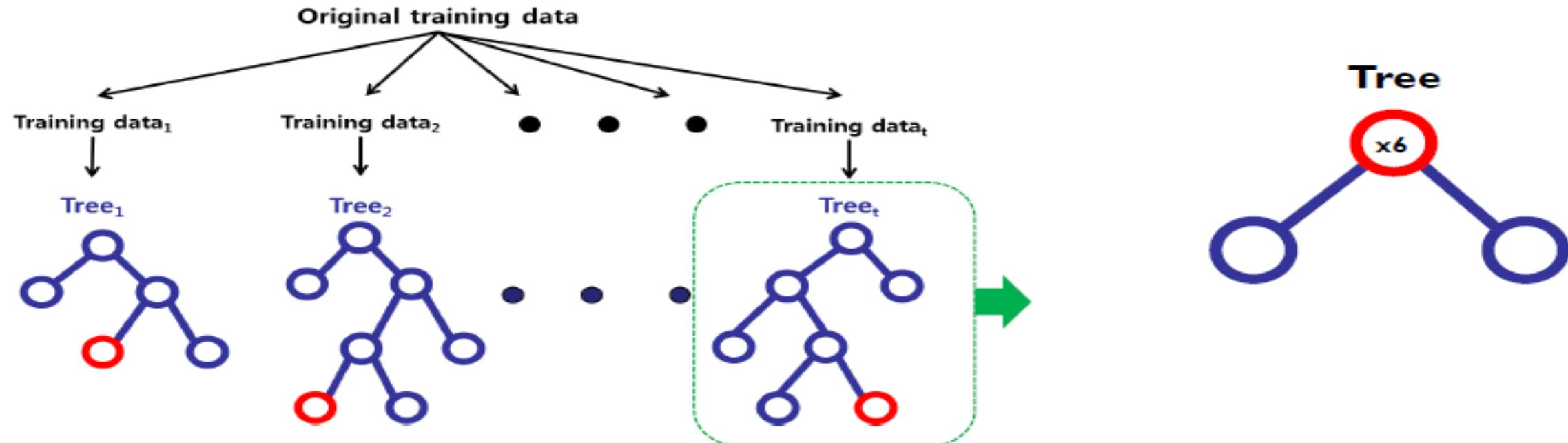
## Unit 02 | RandomForest



1. 원래 변수들 중에서 모델 구축에  
쓰일 입력변수를 무작위로 선택

원래 변수	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16
입력 변수			x3	x5	x6	x7	x8	x9	x10							

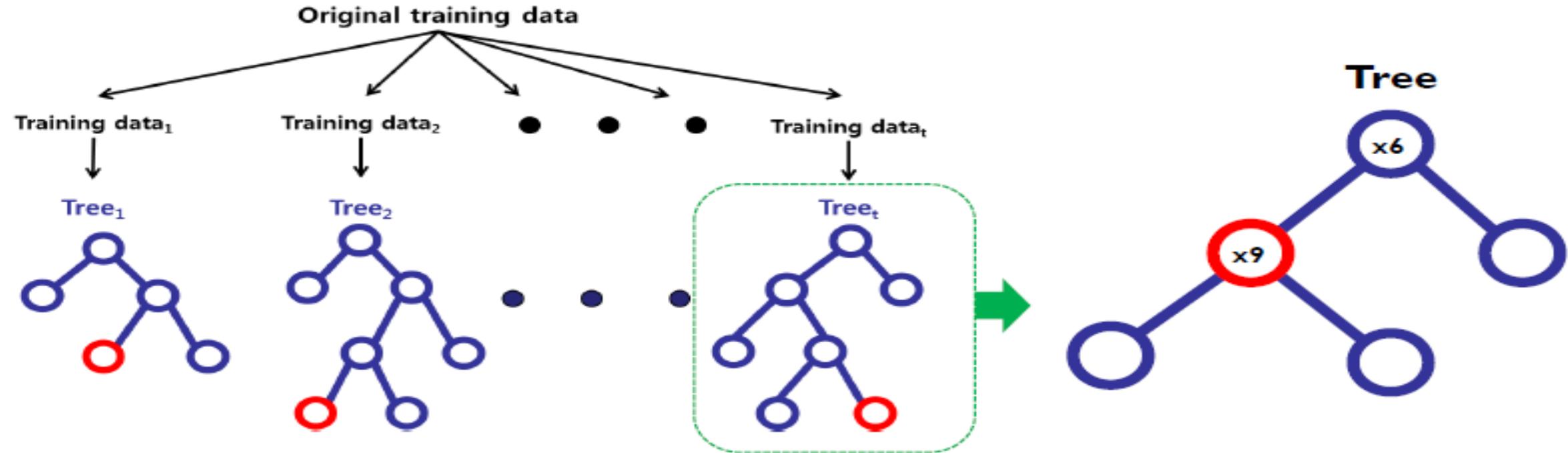
## Unit 02 | RandomForest



2. 선택된 입력 변수 중에서 분할될 변수를 선택

원래 변수	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16
입력 변수			x3		x5	x6				x10						

## Unit 02 | RandomForest



3. 이러한 과정을 **full-grown tree**가  
될 때 까서 반복

Original Features	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16
Input Features	x1				x5				x9	x10						

## Unit 02 | RandomForest



3. 이러한 과정을 **full-grown tree**가  
될 때 까서 반복

원래 변수	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$	$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$	$x_{15}$	$x_{16}$
입력 변수	$x_1$							$x_7$					$x_{13}$	$x_{14}$		

## Unit 02 | RandomForest

### 변수의 중요도

- 랜덤 포레스트는 선형 회귀모델/로지스틱 회귀모델과는 달리 개별 변수가 통계적으로 얼마나 유의하지 알 수 없음(확률 분포 가정을 하지 않기 때문에)
- 따라서 개별 변수의 중요도를 알기 위해 전체 Tree에서 각각의 변수가 impurity를 얼마나 감소시키는지를 평균 계산하여 중요도를 계산함.(Sklearn random forest feature importance)



## Unit 02 | RandomForest

# 변수의 중요도에서 주의할점!

- 불순도를 낮추는 변수는 class가 많은 categorical 변수가 뽑힐 확률이 높게 됨.
- 상관관계가 높은 변수들 경우 하나가 Selection이 되면 나머지는 동일한 가능하기 때문에 Selection 될 확률이 줄어든다. -> **상관관계가 높을 변수들끼리 importance의 왜곡이 생기게된다.**



## Unit 02 | RandomForest

# Hyperparameter

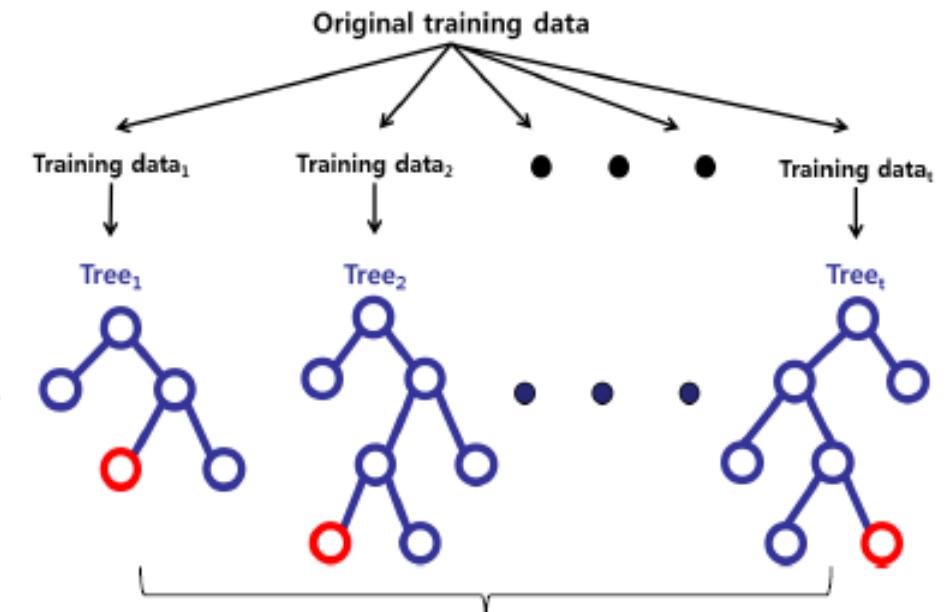
1. Decision Tree의 수 : Strong law od Large Numbers을 만족시키기 위해 2000개 이상의 DT 필요

2. DT에서 노드 분할 시 무작위로 선택되는 변수의 수

- 일반적으로 변수의 수에 따라 다음과 같이 추천됨
- Classification :  $\sqrt{p}$
- Regression :  $\frac{p}{3}$

2. Hyperparameter

$\left\{ X_1, X_5, X_7, X_{12}, X_8 \right\}$   
(무작위 선택 변수 수)



I. Hyperparameter (Base가 되는 Decision tree 수)

## Unit 02 | RandomForest

# Random Forest

```
class sklearn.ensemble. RandomForestRegressor (n_estimators=10, criterion='mse', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,  
n_jobs=1, random_state=None, verbose=0, warm_start=False) ¶ [source]
```

**max\_features** : int, float, string or None, optional (default="auto")

The number of features to consider when looking for the best split:

- If `int`, then consider `max_features` features at each split.
- If `float`, then `max_features` is a percentage and `int(max_features * n_features)` features are considered at each split.
- If "auto", then `max_features=sqrt(n_features)`.
- If "sqrt", then `max_features=sqrt(n_features)` (same as "auto").
- If "log2", then `max_features=log2(n_features)`.
- If `None`, then `max_features=n_features`.

## Unit 02 | RandomForest

# Random Forest –sudo code

1. For  $b = 1$  to  $B$ :
  - (a) Draw a bootstrap sample  $\mathbf{Z}^*$  of size  $N$  from the training data.
  - (b) Grow a random-forest tree  $T_b$  to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size  $n_{min}$  is reached.
    - i. Select  $m$  variables at random from the  $p$  variables.
    - ii. Pick the best variable/split-point among the  $m$ .
    - iii. Split the node into two daughter nodes.
2. Output the ensemble of trees  $\{T_b\}_1^B$ .

To make a prediction at a new point  $x$ :

Regression:  $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$ .

Classification: Let  $\hat{C}_b(x)$  be the class prediction of the  $b$ th random-forest tree. Then  $\hat{C}_{\text{rf}}^B(x) = \text{majority vote } \{\hat{C}_b(x)\}_1^B$ .

## Unit 02 | RandomForest

**[실습3-Randomforest] 켜주세요!**

# Content

---

Unit 01 | Ensemble Model Overview & Voting Classifier

---

Unit 02 | Bagging – Random Forest

---

Unit 03 | Boosting – AdaBoost, Gradient Boosting

---

Unit 04 | XGBoosting, GBM & LightGBM

---

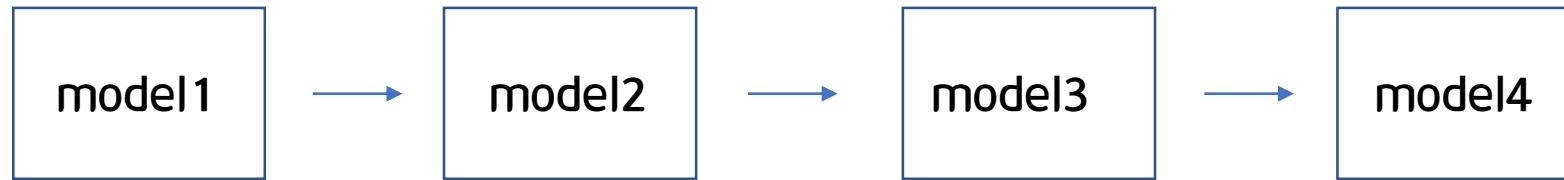
Unit 05 | Stacking

---

## Unit 03 | Boosting

# What is Boosting?

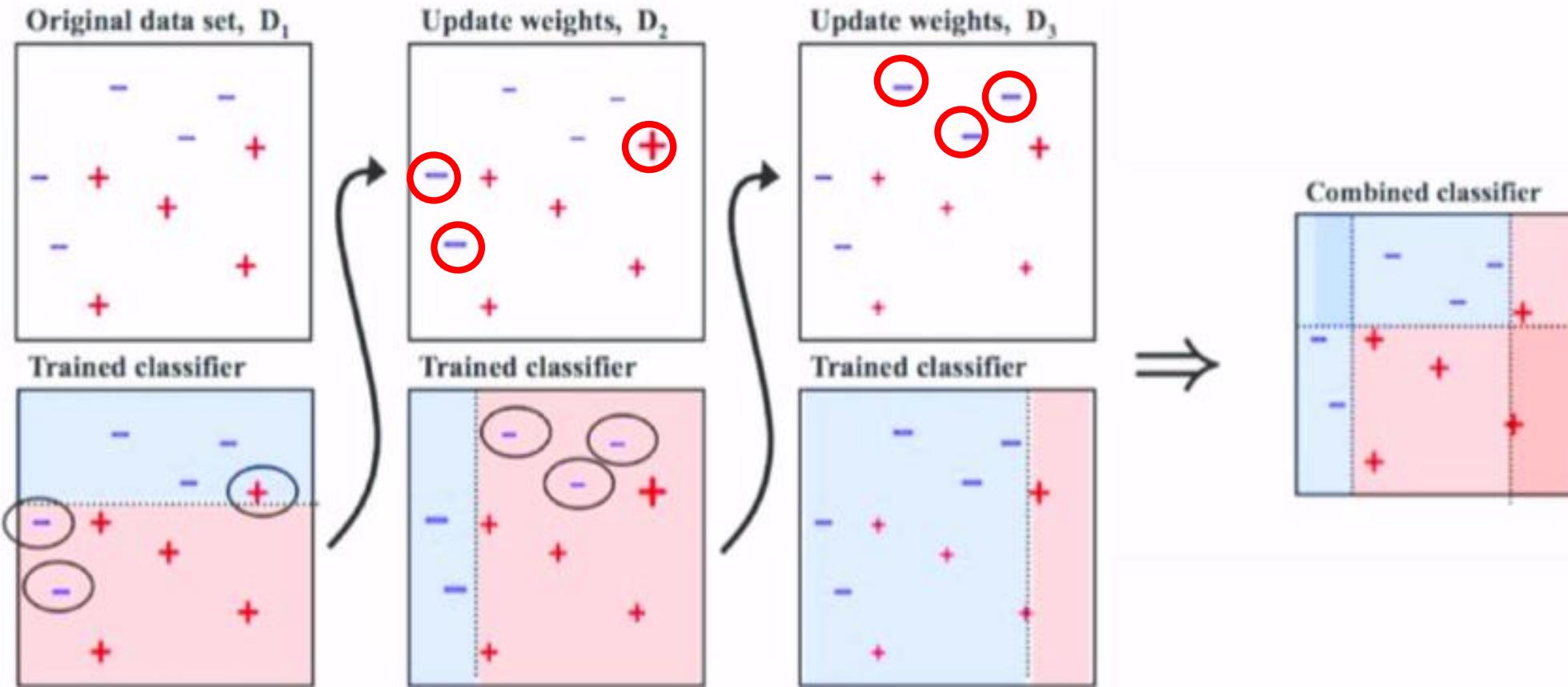
- 학습 Round을 진행하면서 모델을 생성 -> 모델에 의해 각 row의 Weight를 업데이트 함.



- Instance Weight가 높은(오분류된) row를 중심으로 모델을 생성
- 해당 모델들로 양상을 모델을 만듦
- 잘못 분류된 데이터를 더 잘 분류해보는 것이 목적

## Unit 03 | Boosting

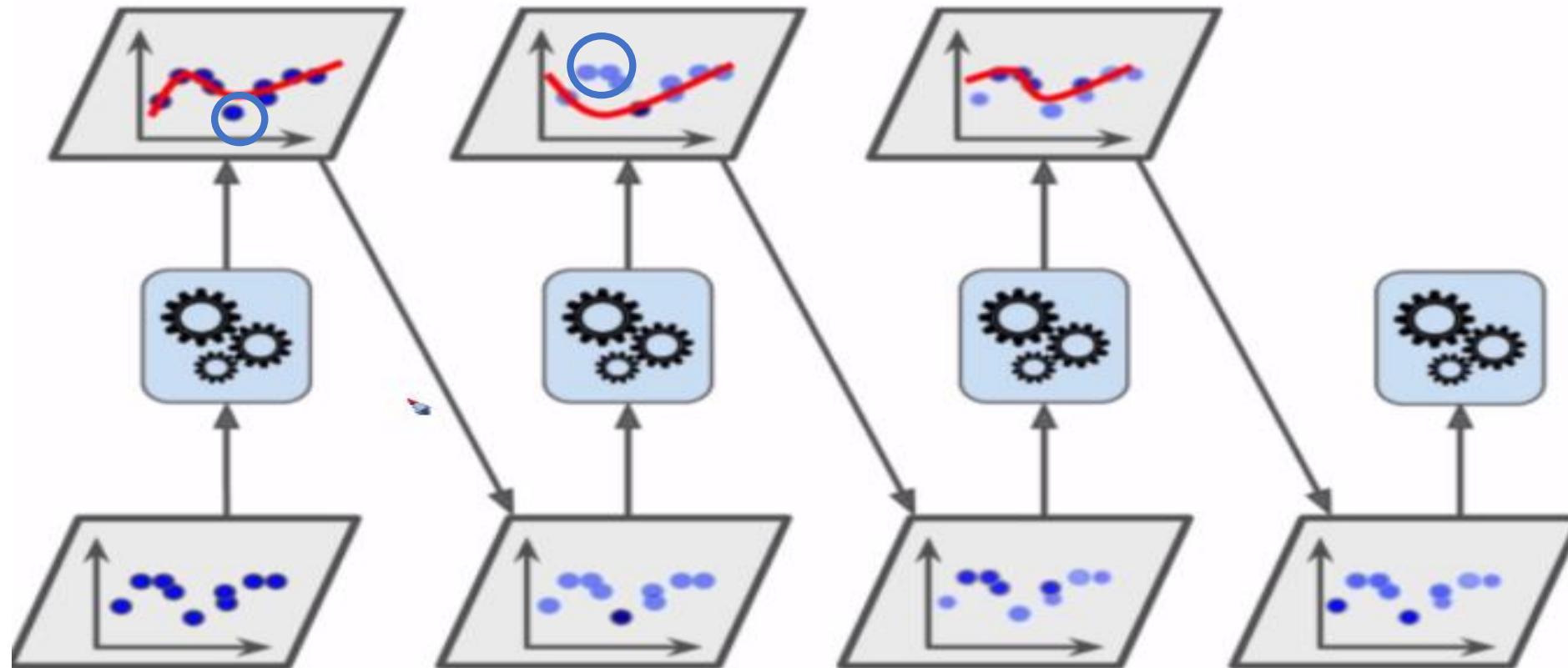
## Boosting - Classifier



오분류된  
Instance에 가중  
치를 줘서 완벽  
해져가는 모델.

## Unit 03 | Boosting

# Boosting – Regression



## Unit 03 | AdaBoost

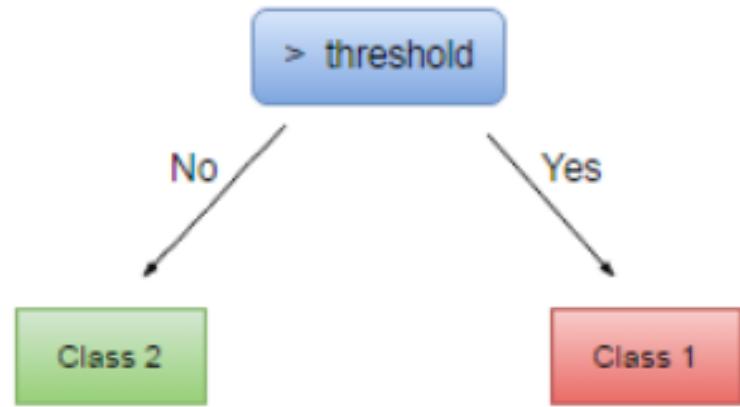
# What is Adaboost?

- Adaptive Boosting = 에이다 부스트
- 매 Round마다 Instance의 Weight값을 계산 
- 틀리는 Instance의 Weight up -> weight 기준 Resampling
- Instance의 Weight합이 클수록, Model의 Weight를 줄임
- 약한 분류기에 입력값을 변화시켜가며 강한 분류기를 만듦(Weak learner)
- High-Depth DT, NN (High Variance Model)에는 적합하지 않음

## Unit 03 | AdaBoost

# Adaboost with Stump

Adaboost의 사용되는 Classifier를 1Depth Tree를 사용하는 방법



Why? Boosting은 약한 모델(High Bias)을 점점 성장 시켜 좋은 분류를 하게 하는 것이기 때문이다. 물론, 다양한 분류 모델을 사용해도 된다! 하지만, DT의 경우 bias, Variance를 조정하기 쉬운 모델이기 때문에 대부분 Tree모델을 사용한다고 보면 된다.

## Unit 03 | AdaBoost

# Adaboost

## Algorithm 10.1 AdaBoost.M1.

값 샘플의 weight를  $1/N$ 로 초기화

1. Initialize the observation weights  $w_i = 1/N$ ,  $i = 1, 2, \dots, N$ .

$N$ 은 데이터의 갯수

2. For  $m = 1$  to  $M$ :  $M$ 은 모델의 갯수

Weight값을 기준으로 분류기 생성(resampling)

(a) Fit a classifier  $G_m(x)$  to the training data using weights  $w_i$ .

(b) Compute 해당 분류기의 에러 계산

$$I(y_i, G_m(x_i)) = \begin{cases} 0 & \text{if } y_i = G_m(x_i) \\ 1 & \text{if } y_i \neq G_m(x_i) \end{cases}$$

$$\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$

(c) Compute  $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$ . 해당 분류기의 분류기 가중치 생성

(d) Set  $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$ ,  $i = 1, 2, \dots, N$ .

Instance의 weight 업데이트

3. Output  $G(x) = \text{sign} \left[ \sum_{m=1}^M \alpha_m G_m(x) \right]$ .

## Unit 03 | AdaBoost

# Adaboost

값 샘플의 weight를  $1/N$ 로 초기화

1. Initialize the observation weights  $w_i = 1/N, i = 1, 2, \dots, N.$   
N은 데이터의 갯수

모든 Instance의 초기값은 다 동일하게 시작을 함.

Ex) Instance가 100개 이면 모든  $W_i = 1/100$

## Unit 03 | AdaBoost

# Adaboost

2. For  $m = 1$  to  $M$ : **M은 모델의 갯수**

Weight값을 기준으로 분류기 생성(resampling)

(a) Fit a classifier  $G_m(x)$  to the training data using weights  $w_i$ .

Instance의 Weight값을 기준으로 Sampling을 진행

최초 샘플링 : 모두 뽑힐 확률이 동일함.

n round 샘플링 : 틀린 값에 가중치가 부여되므로 뽑힐 확률이 높아짐

**!지금부터 for문이 돌아간다고 생각하면 됩니다!!!**

## Unit 03 | AdaBoost

# Adaboost

(b) Compute 해당 분류기의 에러 계산

$$\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$
$$I(y_i, G_m(x_i)) = \begin{cases} 0 & \text{if } y_i = G_m(x_i) \\ 1 & \text{if } y_i \neq G_m(x_i) \end{cases}$$

예측값과 실제값이 같으면 0이 되고, 다르면 1이 되게 함.

분자 : 틀린 예측 값의 instance의 weigh만 남게 되어 모두 더함

분모 : 모든 가중치를 더 함(Normalizer 역할)

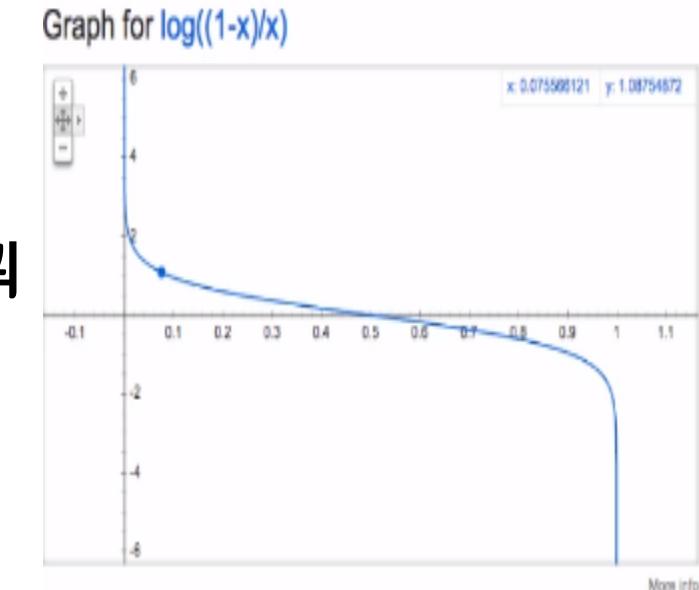
따라서 err의 범위는  $0 < \text{err} < 1$ 이다.

## Unit 03 | AdaBoost

# Adaboost

(c) Compute  $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$ . 해당 분류기의 분류기 가중치 생성

- 에러가 높으면 m번째 모델은 좋지 않는 모델
- 에러가 낮으면 m번째 모델은 좋은 모델
- 따라서 반비례 관계이므로  $(1-\text{err})/\text{err}$ 를 통해 반대방향으로 바꿔 모델의 가중치를 계산한다.
- 전체 양상블에 얼마만큼의 역할을 반영할지 계산하는거임!!



## Unit 03 | AdaBoost

# Adaboost

$$I(y_i, G_m(x_i)) = \begin{cases} 0 & \text{if } y_i = G_m(x_i) \\ 1 & \text{if } y_i \neq G_m(x_i) \end{cases}$$

- (d) Set  $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$ ,  $i = 1, 2, \dots, N.$   
- - - Instance의 weight 업데이트

모델의 가중치의 exp를 하여 틀린 놈들만 가중치를 업데이트 시켜버림.

이 짓을 for문이 Classifier 개수만큼 돌아가게 합니다.

## Unit 03 | AdaBoost

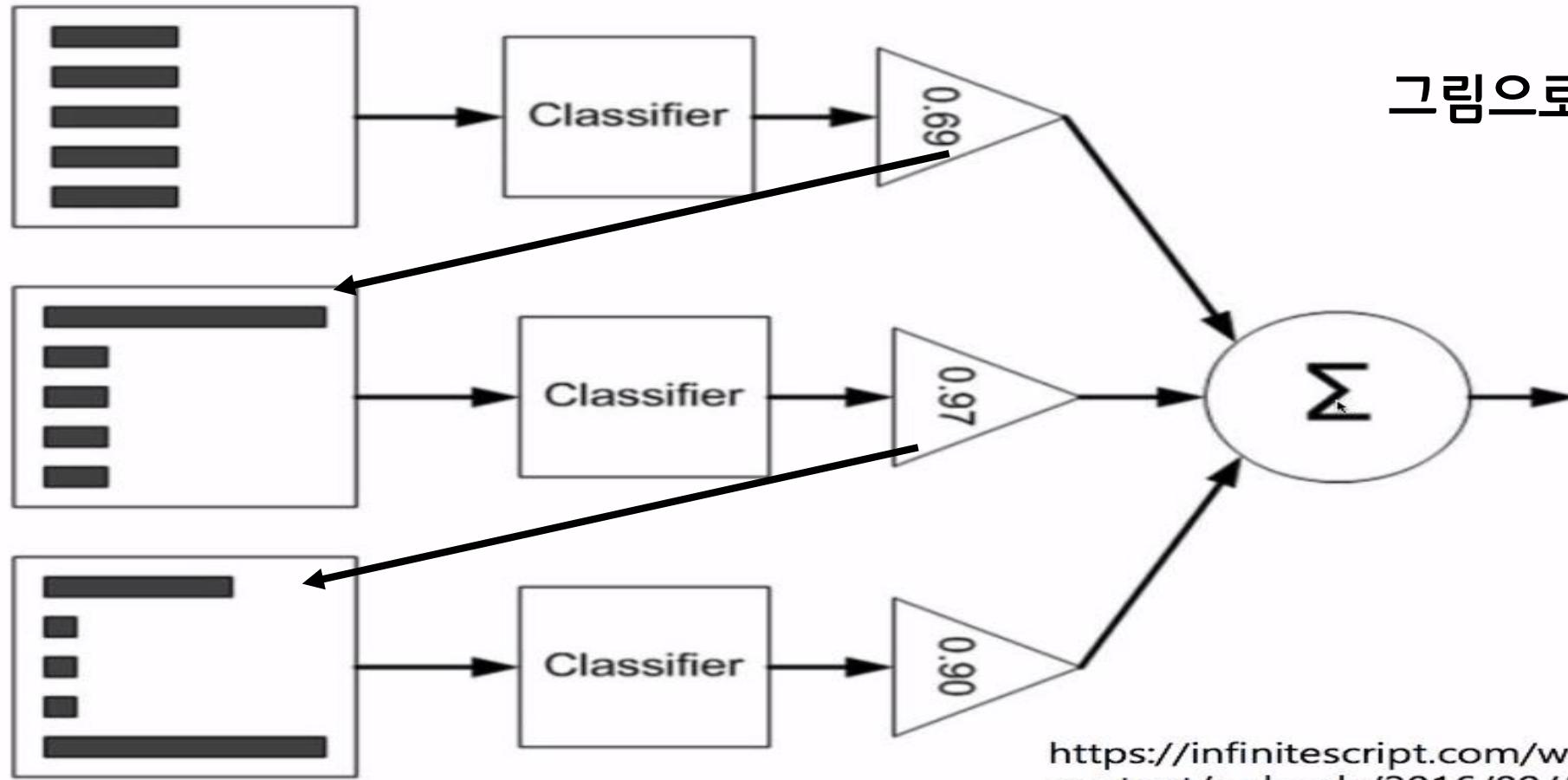
# Adaboost

$$3. \text{ Output } G(x) = \text{sign} \left[ \sum_{m=1}^M \alpha_m G_m(x) \right].$$

그러면 Classifier M개를 돌렸겠죠? 그리고 각각의 모델의 가중치가 있을 겁니다. 그 가중치를 곱하여 각각의 역할 비중을 줘서 Sumation을 통해 예측을 하는겁니다.

## Unit 03 | AdaBoost

# Adaboost



그림으로 보면 이렇습니다.

<https://infinitescript.com/wordpress/wp-content/uploads/2016/09/AdaBoost.jpg>

## Unit 03 | AdaBoost

# Adaboost

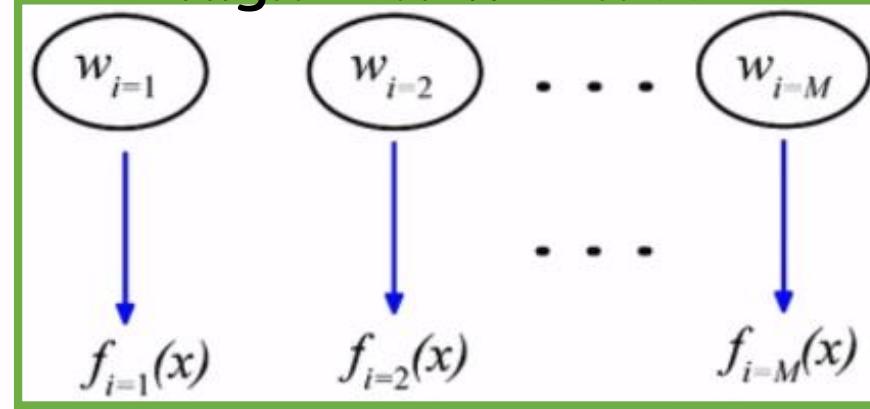


- 가장 인기있는 Boosting 알고리즘. 높은 성능을 보임.
- 계속 틀리는 Instance만 조져서 Model을 향상시킴
- 물론, Hyperparameter 조정을 통해 성능을 향상시켜야 한다.

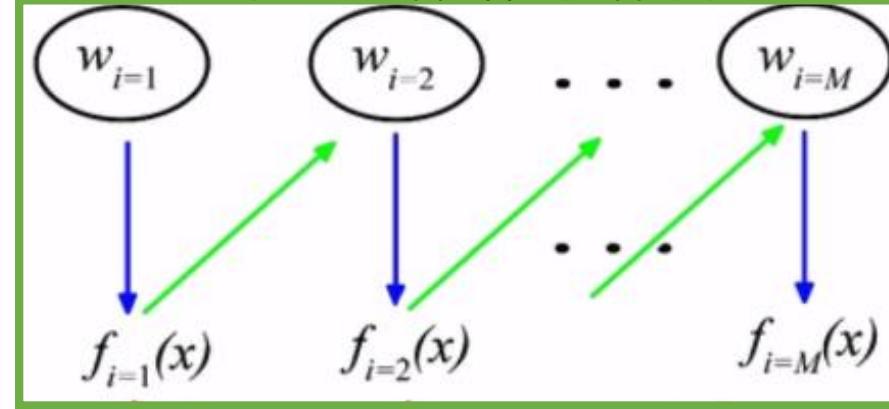
## Unit 03 | Bagging VS Boosting

# Bagging VS Boosting

High Variance Tree



Low Variance Tree



$$F_M(x) = \text{sign}(\sum_{i=1}^M f_i(x))$$

b. Bagging

$$F_M(x) = \text{sign}(\sum_{i=1}^M f_i(x))$$

a. Boosting

## Unit 03 | Bagging VS Boosting

# Bagging VS Boosting

특성	Bagging	Boosting
병렬화	O	X
Variance	High Variance Tree	Low Variance Tree (High Bias)
Sampling	All Same	Not Same ->Bagging보다 좋은 성능

## Unit 03 | Adaboost

[실습4-Adaboost] 켜주세요!

## Unit 03 | Gradient Boosting

※주의 지금부터 수식적인 내용이 많아서 개념 위주로 가겠습니다.

## Unit 03 | Gradient Boosting

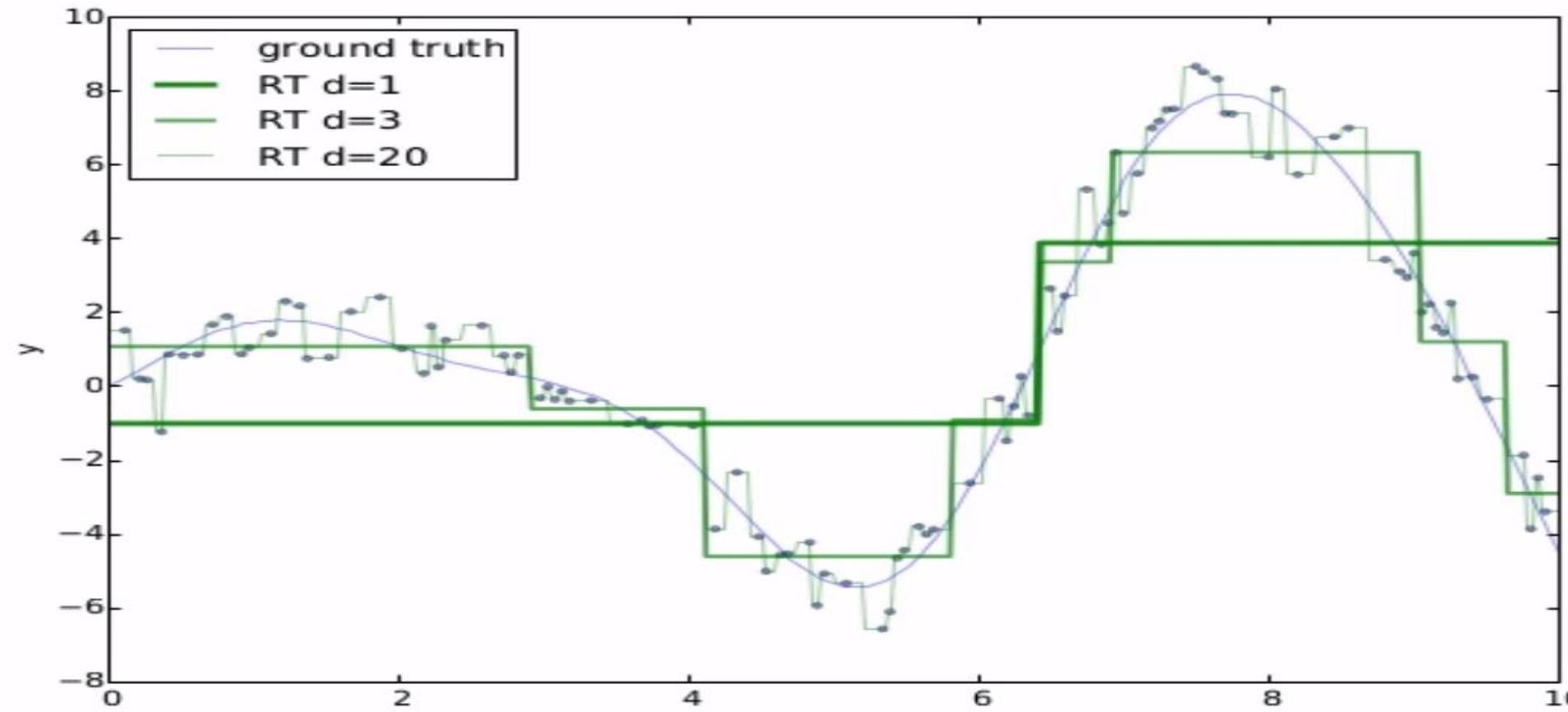
# What is Gradient Boosting

- Adaboost와 같은 Boosting 기법의 일종
- Regression, Classification 등 모두 사용
- Sequential + Additive Model
- 이전 모델의 Residual을 가지고 Model를 강화시킴
- Residual을 예측하여 발전하는 Weak learner

## Unit 03 | Gradient Boosting

# Boosting with regression

Function approximation with Regression Trees



## Unit 03 | Gradient Boosting

# Boosting with regression

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
etc...	etc...	etc...	etc...



- Gradient Boost는 하나의 leaf로 시작함.
- 이 leaf는 sample에 대한 label의 첫번째 예측값임.

## Unit 03 | Gradient Boosting

# Boosting with regression

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
etc...	etc...	etc...	etc...



73.3

- 보통 이 값은 Train label의 평균으로 지정함.

## Unit 03 | Gradient Boosting

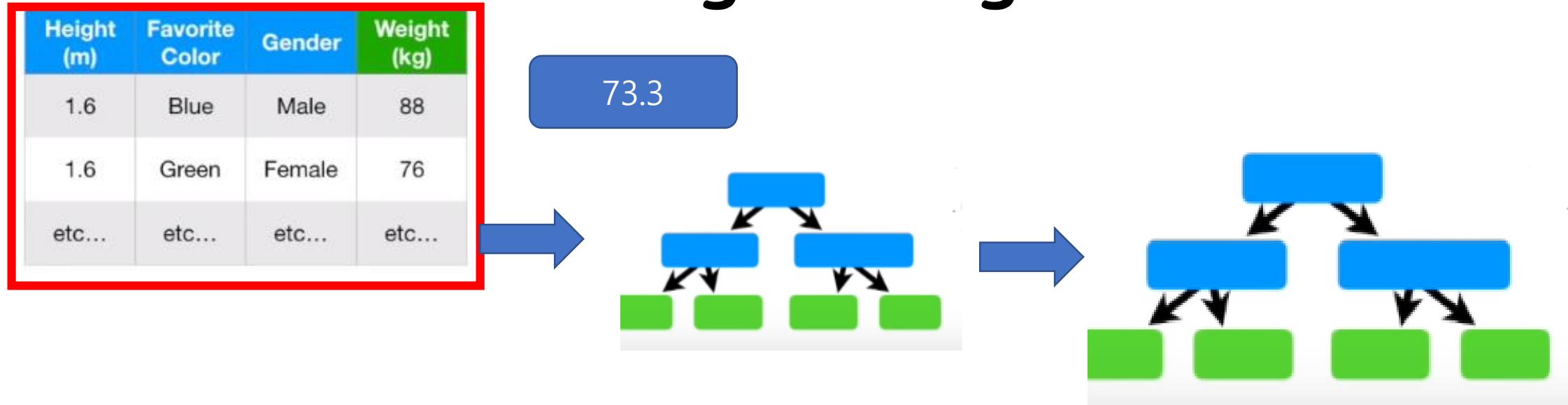
## Boosting with regression



- 다음으로 Tree를 만들게 되고, 이때 Tree는 Stump는 아님.
- 보통 Leaf의 개수를 데이터 size에 따라 4,8,32개로 제한함.
- Gradient Boost는 모든 Tree를 같은 정도로 scale을 진행함. 이 때의 정도를 Learning Rate라 함.

## Unit 03 | Gradient Boosting

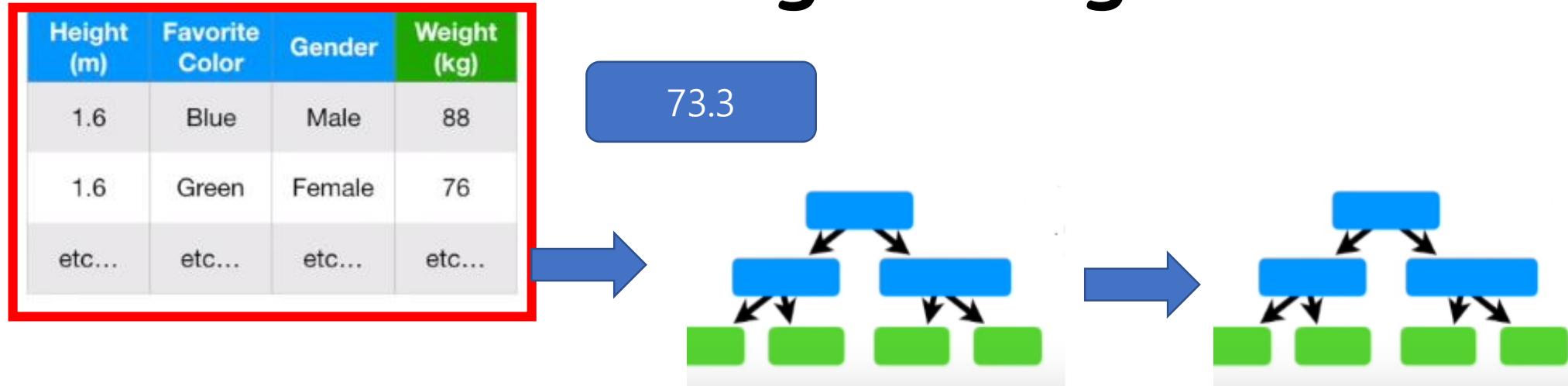
## Boosting with regression



- 다음 Round에서 전 Tree의 error를 기반으로 새로운 Tree를 만든다.

## Unit 03 | Gradient Boosting

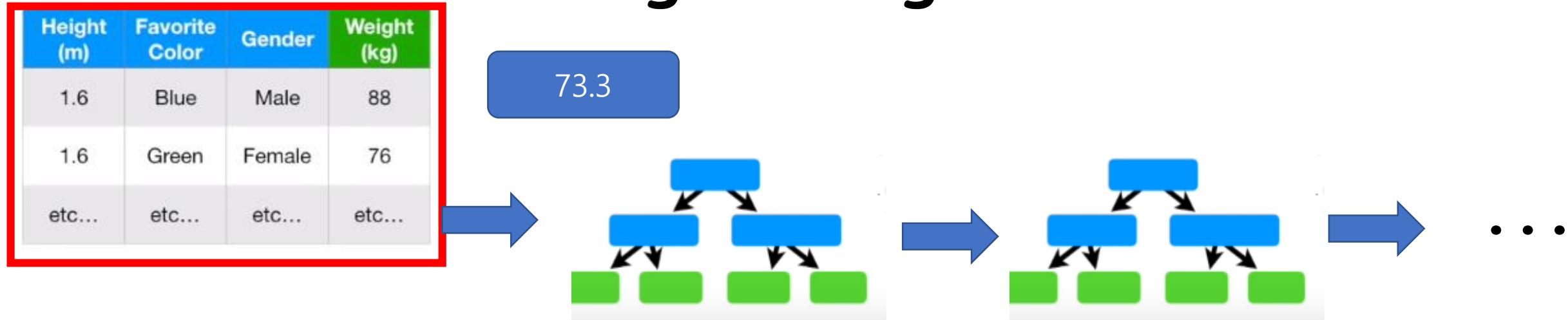
## Boosting with regression



- Learning Rate에 따라 다시 Scale을 제한함.

## Unit 03 | Gradient Boosting

## Boosting with regression



- 반복 Step

## Unit 03 | Gradient Boosting

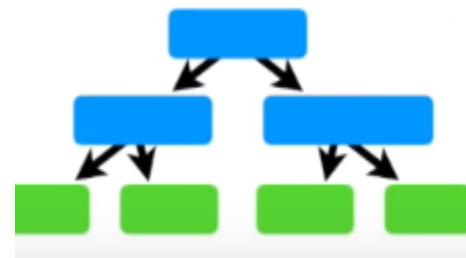
## Boosting Example

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

첫번째 예측값



첫번째 예측값을 기준으로  
잔차를 구하고 이 잔차를 예  
측하는 Tree를 만든다.



## Unit 03 | Gradient Boosting

## Boosting Example

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



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

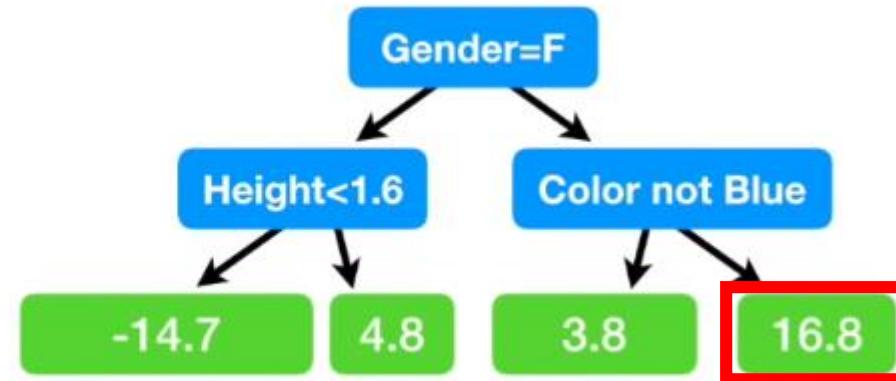


## Unit 03 | Gradient Boosting

## Boosting Example

Average Weight  
71.2

+



Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

Predicted =  $71.2 + 16.8 = 88$   
→ 완벽하게 맞쳤으나 이는 Over fitting 됐다  
고 볼 수 있다.(Low Bias High Variance)

## Unit 03 | Gradient Boosting

## Boosting Example

Average Weight  
71.2

+ Learning Rate X



따라서 Gradient Boosting은 이런 문제를 해결하기 위해 Learning Rate를 곱하여 Tree의 영향력을 Scale하게 된다.

## Unit 03 | Gradient Boosting

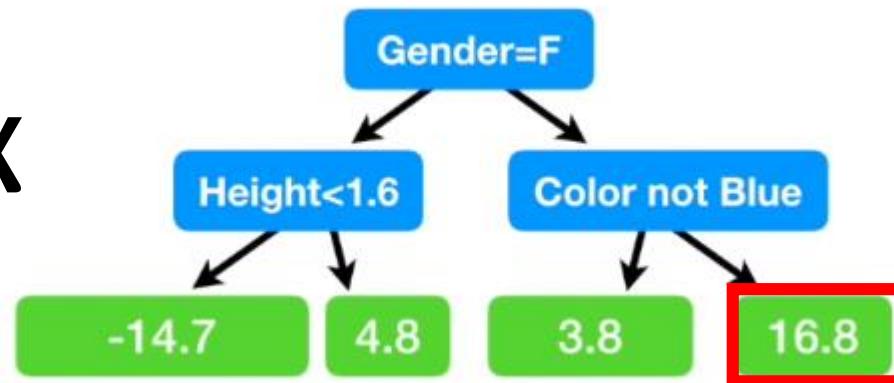
## Boosting Example

Average Weight

71.2

+

0.1 X



$$\text{Prediction} = 71.2 + (0.1 \times 16.8) = 72.9$$

Jerome Friedman(GB 만드신 분)은 경험적으로 이러한 scale을 통한 많은 Round를 진행하는 것이 Test 성능이 좋다는 것을 증명하였다.

## Unit 03 | Gradient Boosting

## Boosting Example

Average Weight

**71.2**

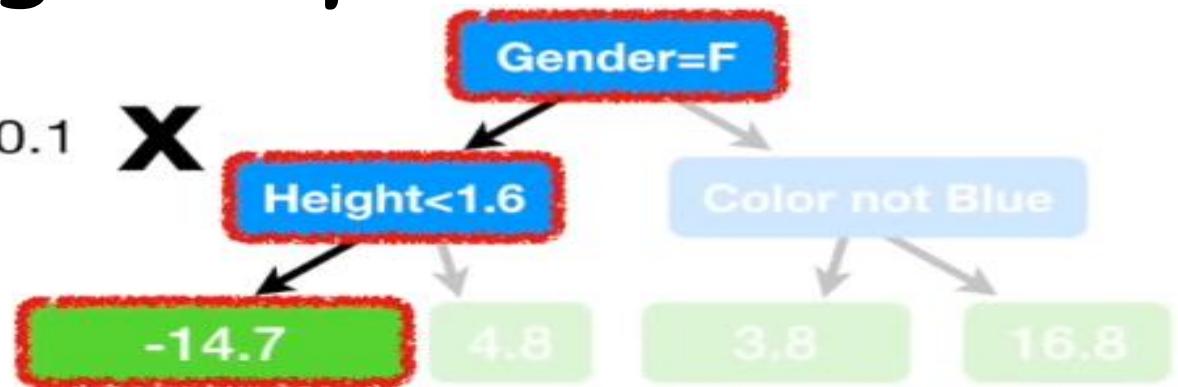
+

0.1

X

Gender=F

Height&lt;1.6



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

**Residual = (Observed - Predicted)**

$$57 - (71.2 + 0.1 \times -14.7) = -12.7$$

## Unit 03 | Gradient Boosting

## Boosting Example



## Unit 03 | Gradient Boosting

## Boosting Example

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 03 | Gradient Boosting

## Boosting Example

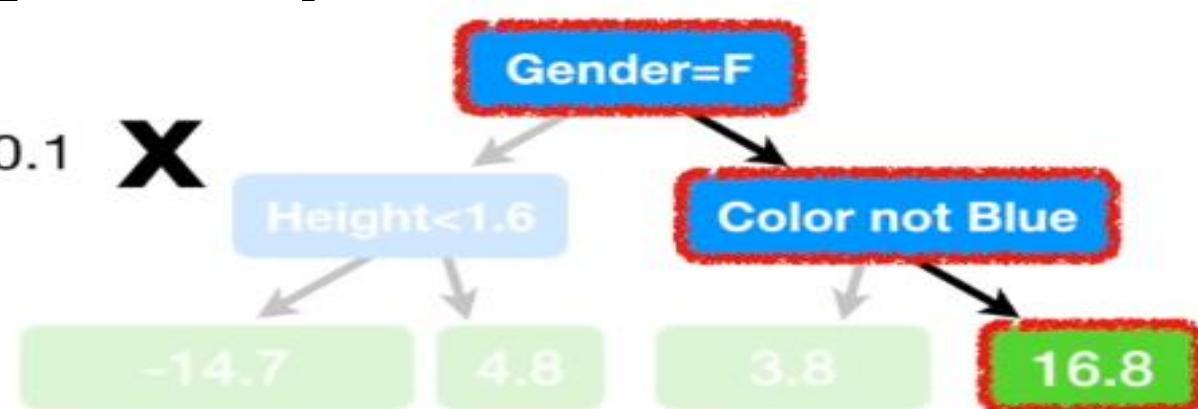
Average Weight

71.2

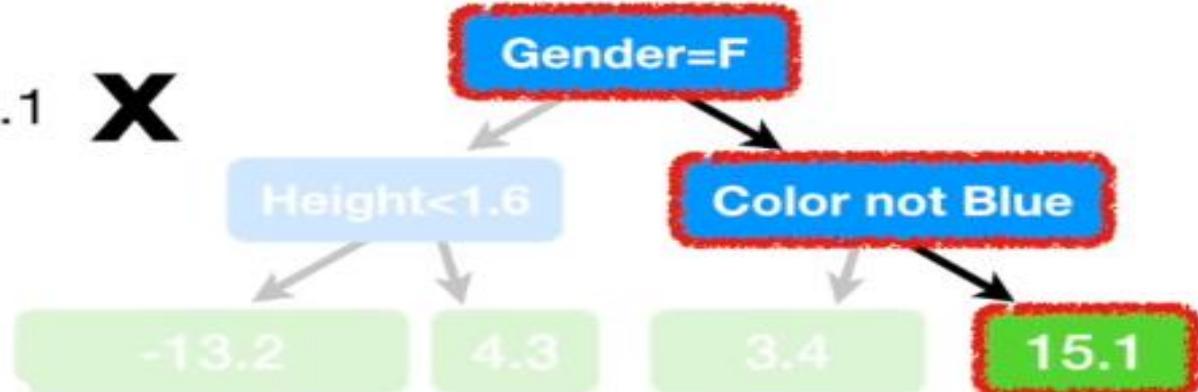
+ 0.1 ×

That gives us...

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



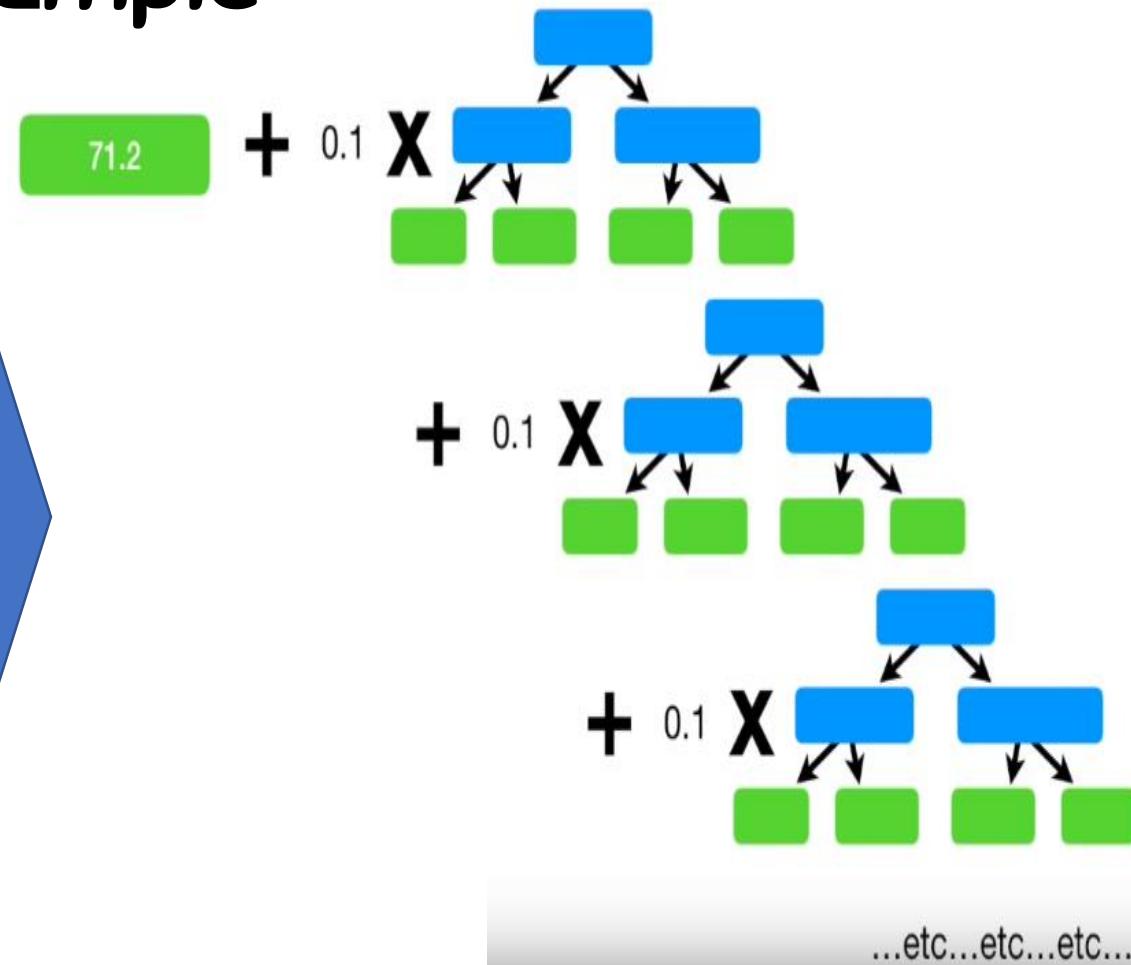
+ 0.1 ×



## Unit 03 | Gradient Boosting

## Boosting Example

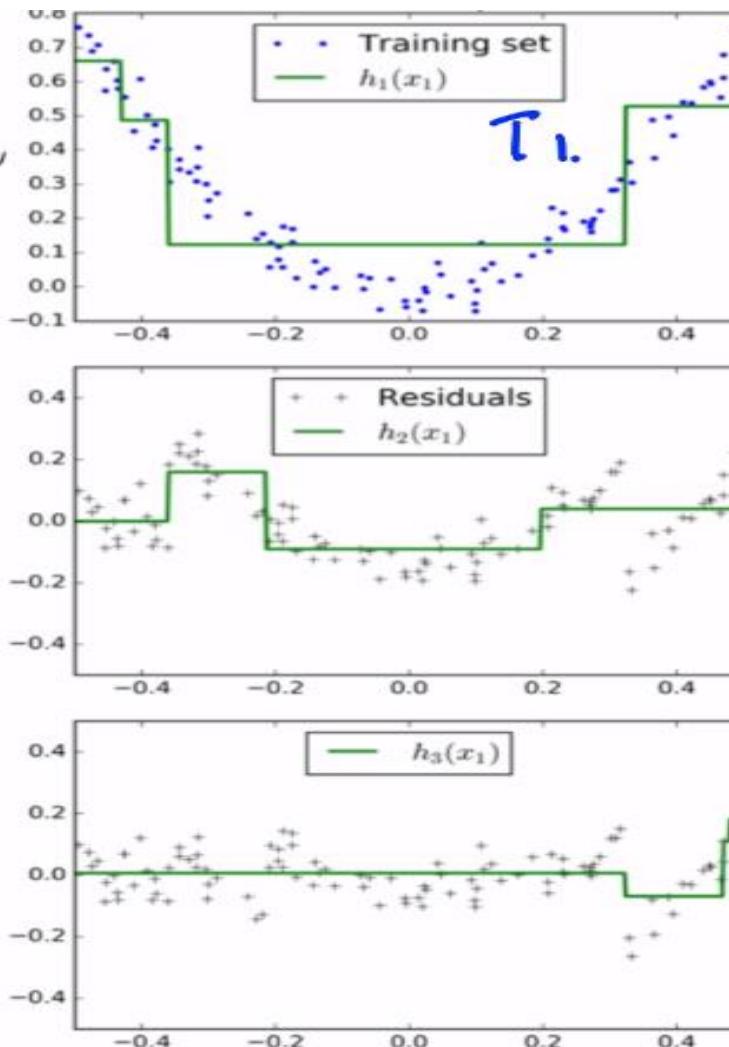
Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	13.6
1.6	Green	Female	76	3.9
1.5	Blue	Female	56	-12.4
1.8	Red	Male	73	1.1
1.5	Green	Male	77	5.1
1.4	Blue	Female	57	-11.4



## Unit 03 | Gradient Boosting

근데 왜 Gradient라고 할까????

## Unit 03 | Gradient Boosting



**Gradient Boosting은 잔차를 없애는  
방향으로 학습을 하게 된다.  
→ 이때 학습 방법을 Gradient  
Method를 쓰기 때문이다.**

## Unit 03 | Gradient Boosting

## Tuning parameters

- Number of Tree : 얼마나 많은 Sequential한 Tree를 만들 것인가?
- Depth of Tree : 트리의 복잡성 -> 2,3,4 정도로 짧을 수록 좋다.
- Subsampling : Random Forest의 Sampling과 같음
- Shrinkage Parameter  $\lambda$  : 보통 람다는 작고 estimator는 큰 것이 좋음
- Fitting to low variance

## Unit 03 | Gradient Boosting

# Sklearn with Gradient Boosting

## 3.2.4.3.6. `sklearn.ensemble.GradientBoostingRegressor`

```
class sklearn.ensemble.GradientBoostingRegressor (loss='ls', learning_rate=0.1, n_estimators=100,  
subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,  
max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None,  
max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, presort='auto') [source]
```

**loss** : {'ls', 'lad', 'huber', 'quantile'}, optional (default='ls')

**learning\_rate** : float, optional (default=0.1)

**n\_estimators** : int (default=100)

**max\_depth** : integer, optional (default=3)

**subsample** : float, optional (default=1.0)

## Unit 03 | Gradient Boosting

**[실습6-Gradient Boosting] 켜주세요!**

---

Unit 01 | Ensemble Model Overview & Voting Classifier

---

Unit 02 | Bagging – Random Forest

---

Unit 03 | Boosting – AdaBoost, Gradient Boosting

---

Unit 04 | XGBoosting, GBM & LightGBM

---

Unit 05 | Stacking

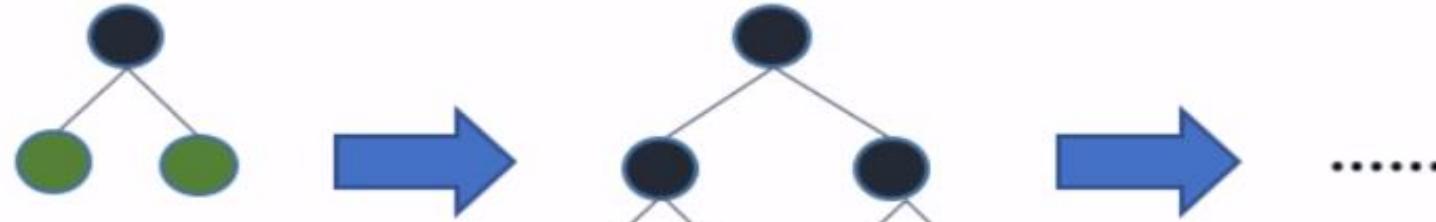
---

## Unit 04 | XGBoost and LightGBM

## Various boosting method

- GBM 연상량과 병렬처리를 위한 패키지가 존재
- 대표적인 패키지는 XGBoost와 LightGBM
- XGBoost – eXtreme Gradient Boosting
- LightGBM – Light Gradient Boosting Machine
- 구현 알고리즘은 일부 상이하나 목표는 비슷함
- 둘다 Tree기반은 Gradient Boosting임!!

## Unit 04 | XGBoost and LightGBM



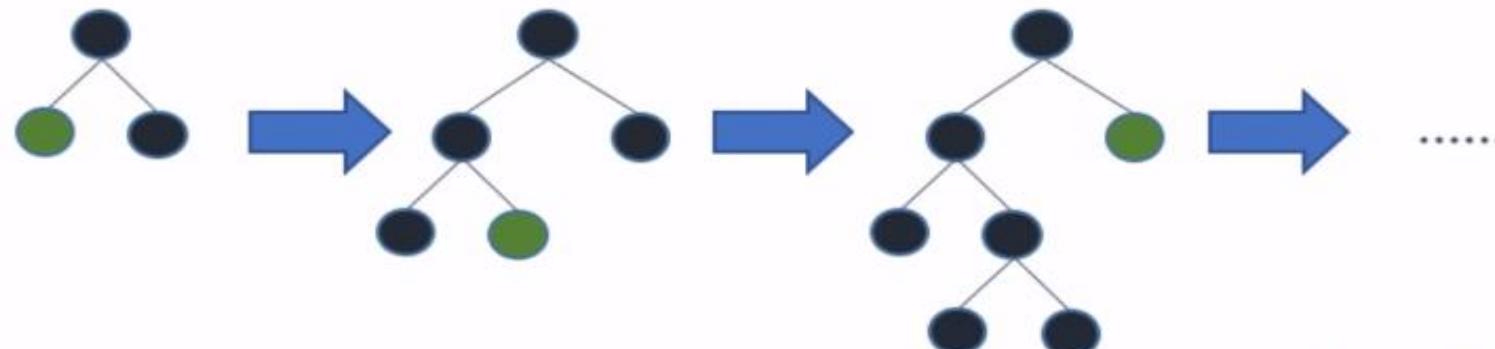
Level-wise tree growth

GBRT

XGBOOST

XGBOOST – 균형적으로 성장

Light GBM – 한쪽으로만 성장



Leaf-wise tree growth

Light GBM

## Unit 04 | XGBoost and LightGBM

		XGBoost	Light GBM	
Parameters Used	max_depth: 50 learning_rate: 0.16 min_child_weight: 1 n_estimators: 200		max_depth: 50 learning_rate: 0.1 num_leaves: 900 n_estimators: 300	
Training AUC Score	0.999	Without passing indices of categorical features	Passing indices of categorical features	0.999
Test AUC Score	0.789	0.785		0.772
Training Time	970 secs	153 secs		326 secs
Prediction Time	184 secs	40 secs		156 secs
Parameter Tuning Time (for 81 fits, 200 iteration)	500 minutes		200 minutes	

# Content

---

Unit 01 | Ensemble Model Overview & Voting Classifier

---

Unit 02 | Bagging – Random Forest

---

Unit 03 | Boosting – AdaBoost, Gradient Boosting

---

Unit 04 | XGBoosting, GBM & LightGBM

---

Unit 05 | Stacking

## Unit 05 | Stacking

# What is Stacking

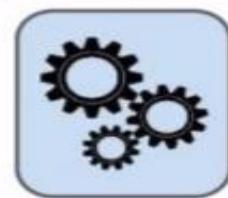
- Meta Ensemble -> 여러 모델의 결과를 뮤어서 예측
- 컴퓨터 성능향상과 함께 최근 대중화 되고 있다.
- 키워드 : Stacking Kaggle Stacknet
- 복잡성으로 인해 완벽한 파이썬 구현체는 아직 없음
- 구현 알고리즘은 일부 상이하나 목표는 비슷함



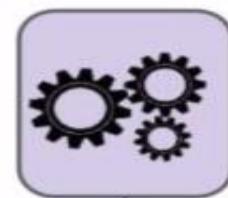
## Unit 05 | Stacking

# Basic Stacking

여러개의 모델을 생성

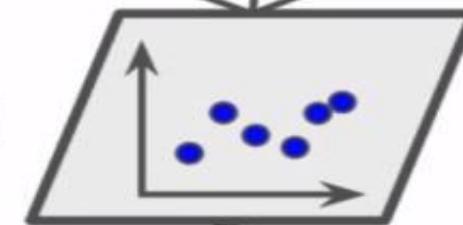


Train

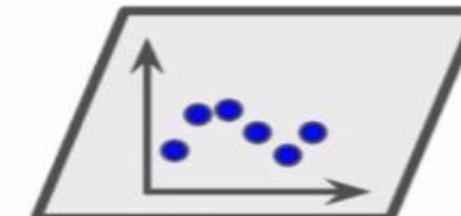


Subset 1로만 학습을 함

Subset 1

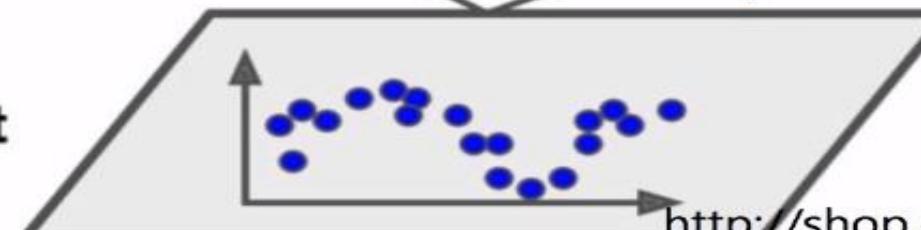


Split



Subset 2  
데이터 셋을 나눔

Training set



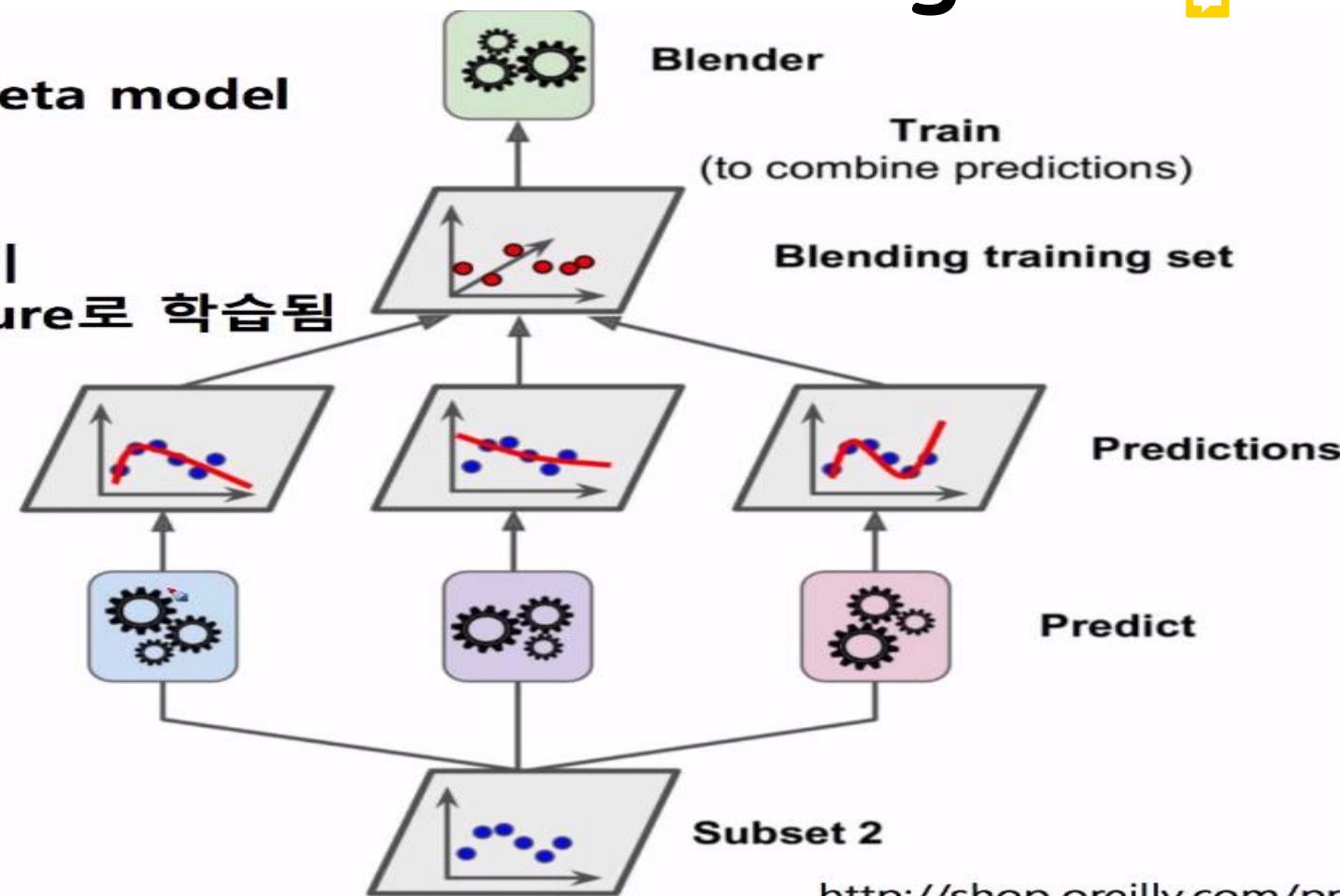
<http://shop.oreilly.com/product/0636920052289.do>

## Unit 05 | Stacking

## Basic Stacking

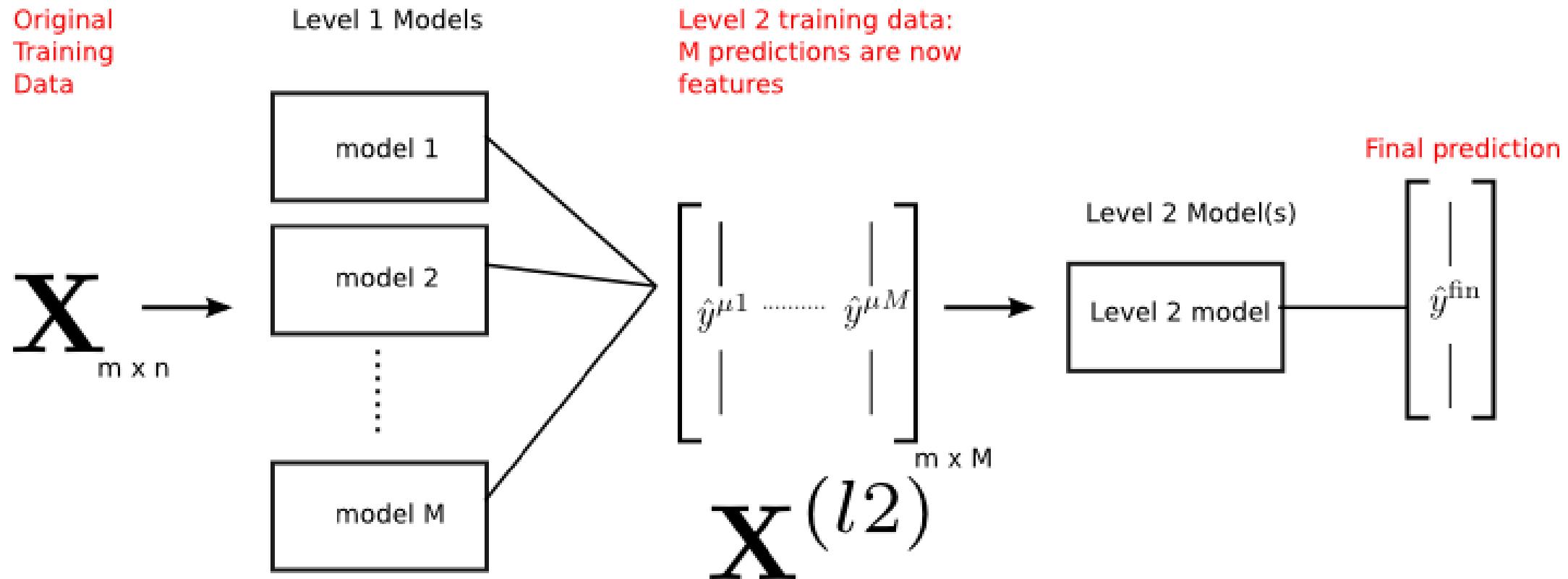
**Meta model**

각 모델의 결과값이  
Meta 모델의 feature로 학습됨

**Blender****Train**  
(to combine predictions)**Blending training set****Predictions****Predict****Subset 2**

## Unit 05 | Stacking

## Basic Stacking



## Unit 05 | Stacking

**[실습7-Stacking] 켜주세요!**

# Q & A

들어주셔서 감사합니다.

이제 과제 설명드리겠습니다.

## Assginment

# 원래 갖고 있던 무기



## Assginment

### 오늘 갖게된 무기



Dual MP7



SG-870\_v3



AK47 Aim



M4A1 Aim



MP5 Aim



Dragunov



KSG-12



KRISS



Tiger Stripe

## Assignment

## 과제 :Dacon

**Mission 2: Predicting the opening/closing of hospitals!**

Assignment : 지금까지 배운 아래 모델 모두를 활용하여 가장 좋은 방법을 찾아 accuracy 를 구해보고 Test도 예측해서 Dacon2에 제출 해보세요! 가장 성능이 좋은 모델로 리더보드 를 찍어보고 캡처해서 올려주세요. 전처리는 제가 해서 드립니다.

지금까지 배운 모델

## 2주차)Linear Regression

## 3주차) KNN, Naive Bayesian

## 4주차) DT(단일 모델), Random Forest, AdaBoost, XGBoost, LightGBM, **STACKING**

**## STACKING방법은 다양할 수 있으나, 생각이 안나시면 오늘 배운 정석적인 Stacking을 하시면 됩니다.**

데이터 설명 : <https://remotedesktop.google.com/access/session/68b1e527-47e4-dc01-1a4e-0e61d1203f27>

## Assignment

# 과제 : Predicting the opening/closing of hospitals!

<https://remotedesktop.google.com/access/session/68b1e527-47e4-dc01-1a4e-0e61d1203f27>

데이터는 전처리 과정없이 그냥 온니 모델링으로만 승부하셔야 합니다. 제가 한 데이터 그대로 모델링을 하면 됩니다. 단일모델로 하이퍼 파리미터 서치를 빽세게 하셔도 좋고 Stacking을 하셔도 좋습니다. Test Accuracy만 높이시면 됩니다. 직접 Submission을 하셔서 코드랑 리더보드 캡처를 올려주시면 됩니다. 그리고 오늘 배운 RF를 이용하여 Feature importance를 뽑아주시기 바랍니다.

꼭 들어가야 하는 내용 : 1. 모델링 및 리더보드 캡처 / 2. Feature importance(RF등 다른 모델도 가능)

12기 분 중 가장 높은 리더보드를 찍으시는 분은 스타벅스 기프티콘을 드리겠습니다.

질문이 있으시면 언제든 편하게 메일 / 카톡 주시면 됩니다.  
메일을 수시로 확인하고 있으니 답장이 늦지 않을겁니다.

이준걸 : 010 8940 6910 / i2019010491@korea.ac.kr

## Assginment

# Reference

1. 존경하는 고려대학교 산업경영공학과 김성범 교수님 K-랜덤포레스트 강의자료
2. 가천대학교 산업경영공학과 최성철 교수님 Machine Learning from scratch 강의자료
3. youtube : <https://www.youtube.com/watch?v=3CC4N4z3GJc> (Gradient boosting)
4. 내 머리