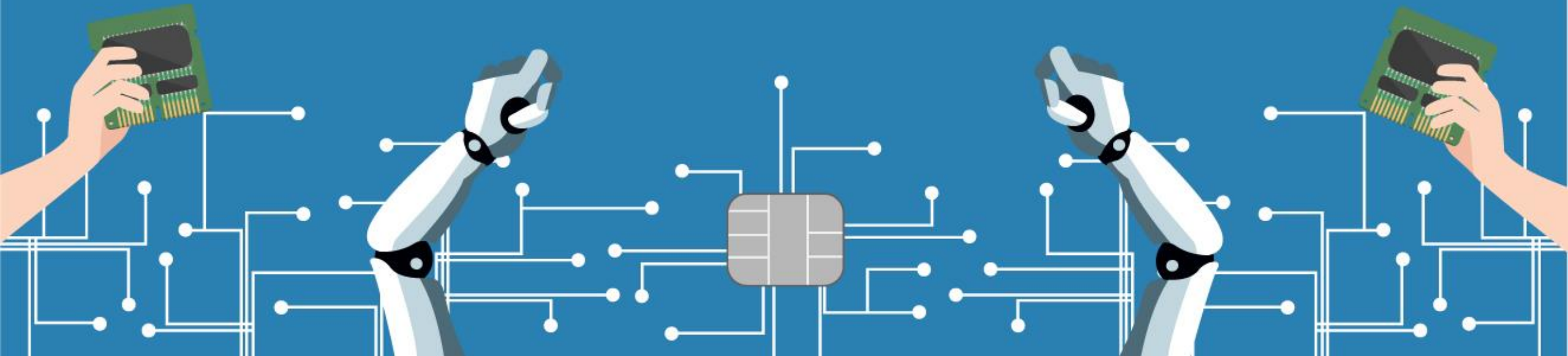
Kuggle

2020_02 Kuggle 정규세션_W5

이 현

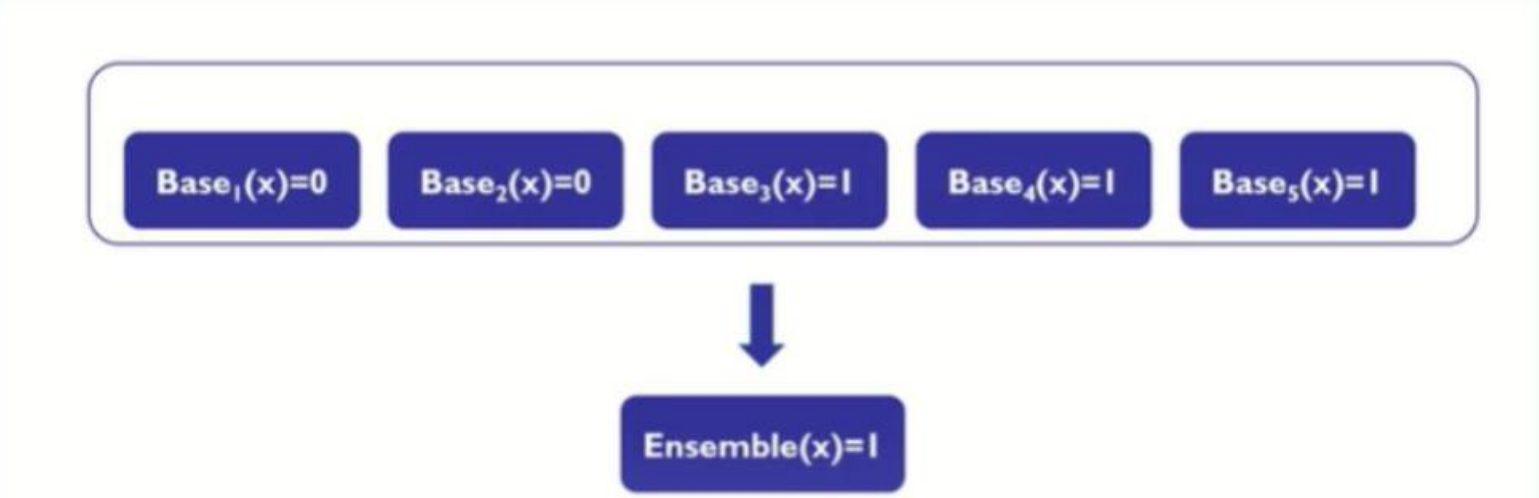


오늘 뚝배기 부술 내용

1. Voting
2. Bagging
3. Boosting
4. Coding

00 Ensemble

앙상블 학습(Ensemble Learning)

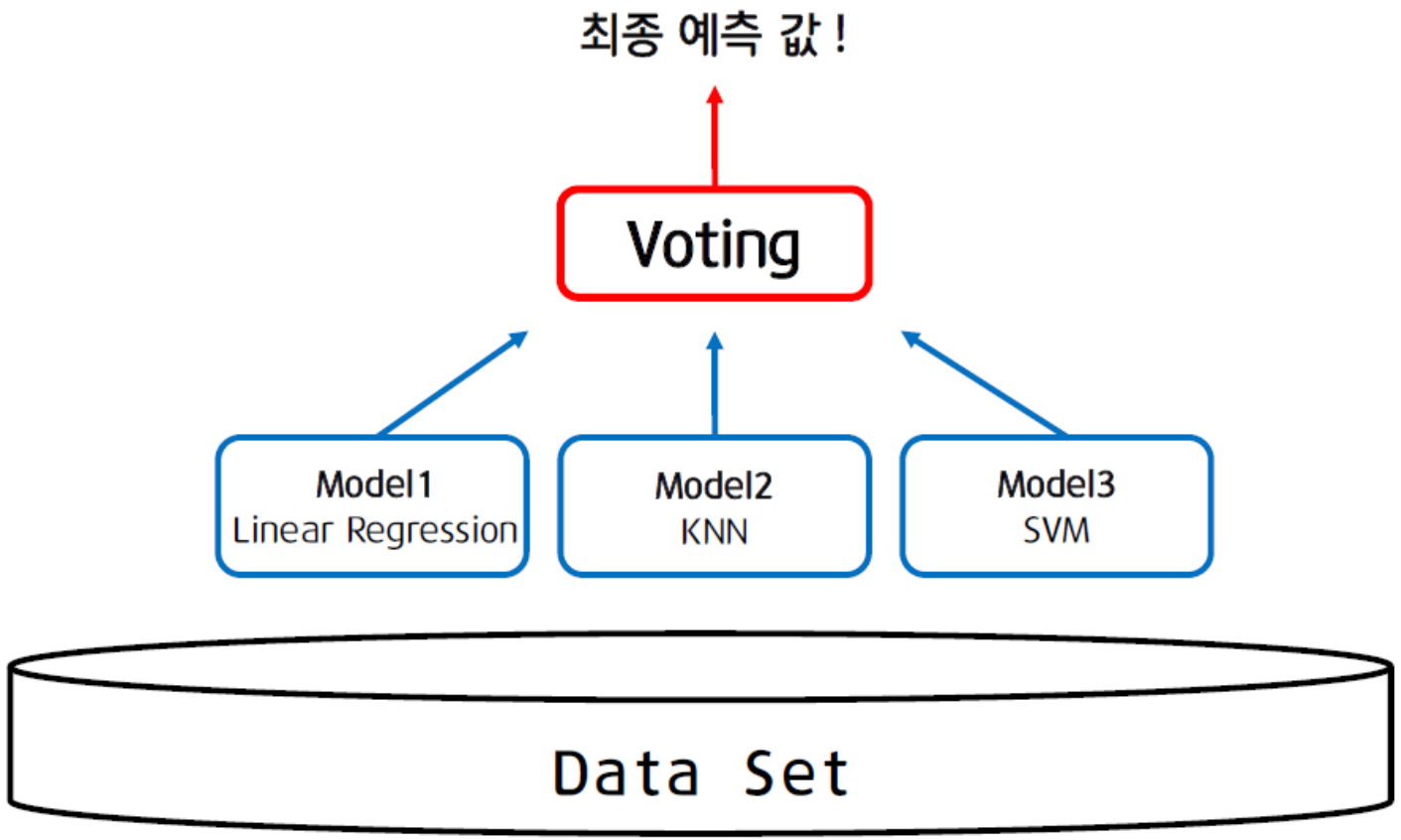


여러 베이스모델을 생성하고 그 예측을 결합함으로써 보다 정확한 최종예측을 도출하는 기법

01. Voting

01 Voting

Voting 보팅 투표 그 자체!

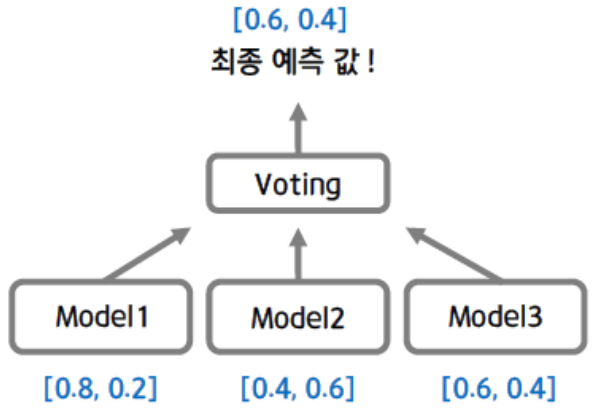


1. 하드 보팅 Hard Voting

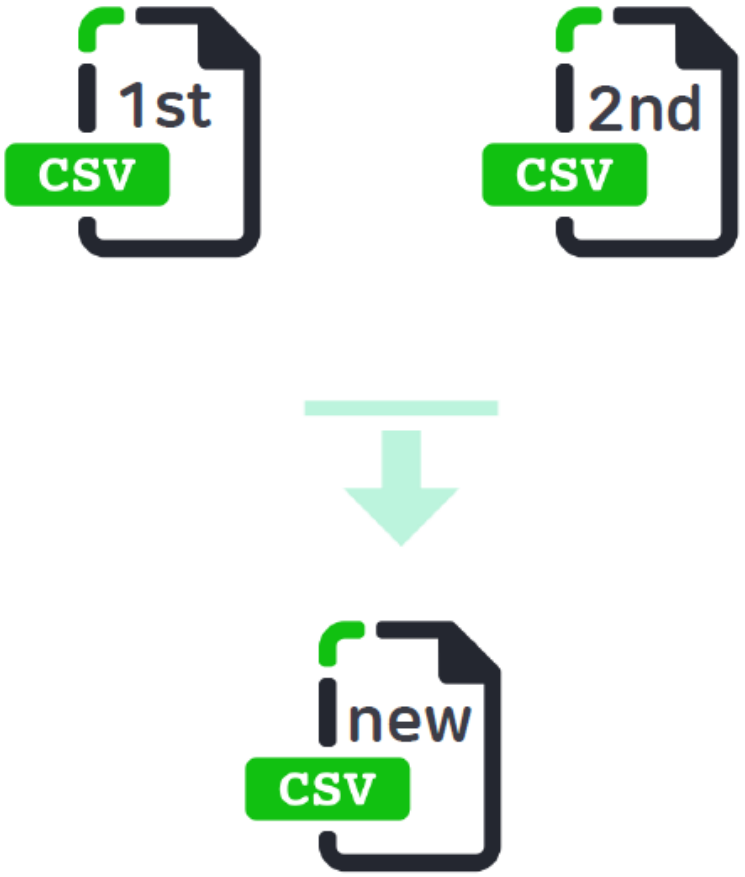
-> 다수결의 원칙과 비슷, 예측한 결과값 중 다수의 분류기가 결정한 값을 최종 예측 값으로 선정

2. 소프트 보팅 Soft Voting

-> 가중치 투표, 클래스의 결정 확률을 평균하여 결정



01 Voting



Submission Ensemble

굳이 scikit-learn의 voting을 쓰지 않고 직접 각각의 예측 값들에 대해 Ensemble해주는 경우가 많음.
이때 유용한 계산법을 골라보자면

1. 산술평균
2. 기하평균
3. 가중평균

Appendix. 멱평균 (Power mean)

02. Bagging

02 Bagging

Bagging 배깅 Bootstrap Aggregation !!

Bootstrap 부트스트랩

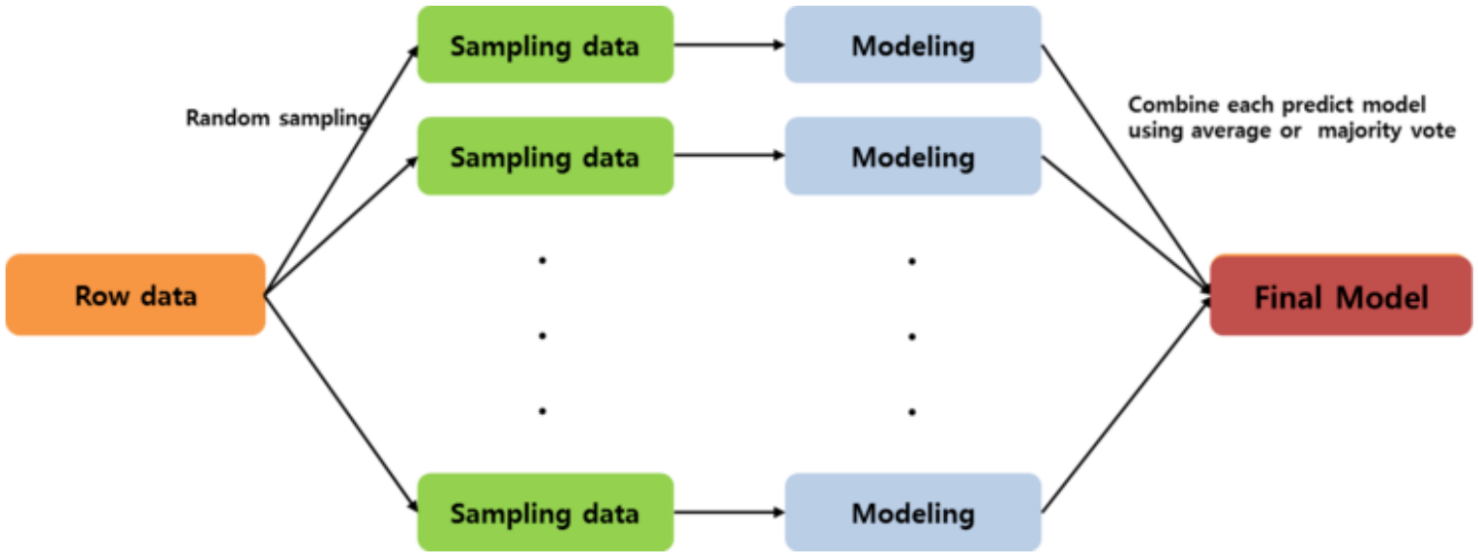
- 부트스트래핑은 샘플 카피 기법
- 카피된 샘플들을 부트스트랩 샘플이라고 함
- n 개의 train sample 에서 m 번 복원 추출 (with replacement)

  (보통 m=n 일 때 평균적으로 36.8% 의 sample 이 생략됨!)
- 분산을 줄이는 효과
- 10000개의 데이터에서 10000번 복원 추출해서 뽑는다면 동일하게 10000개지만 처음 데이터와는 다름!

02 Bagging

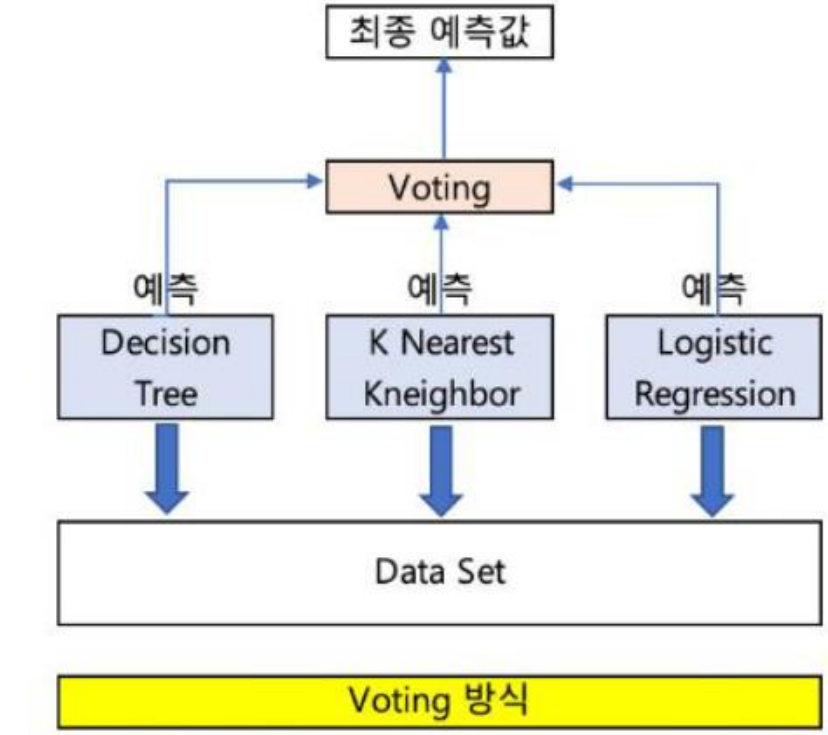
단순히 같은 데이터셋으로 만든 여러가지 classifier 는 의미가 없다!! -> 부트스트랩 샘플 사용

- 배깅 (1996 ~), 최초의 앙상블 방법
- 부트스트랩 샘플을 이용해서 Voting 보팅!
  -> 범주형 변수는 다수결
  -> 연속형 변수는 평균으로 집계

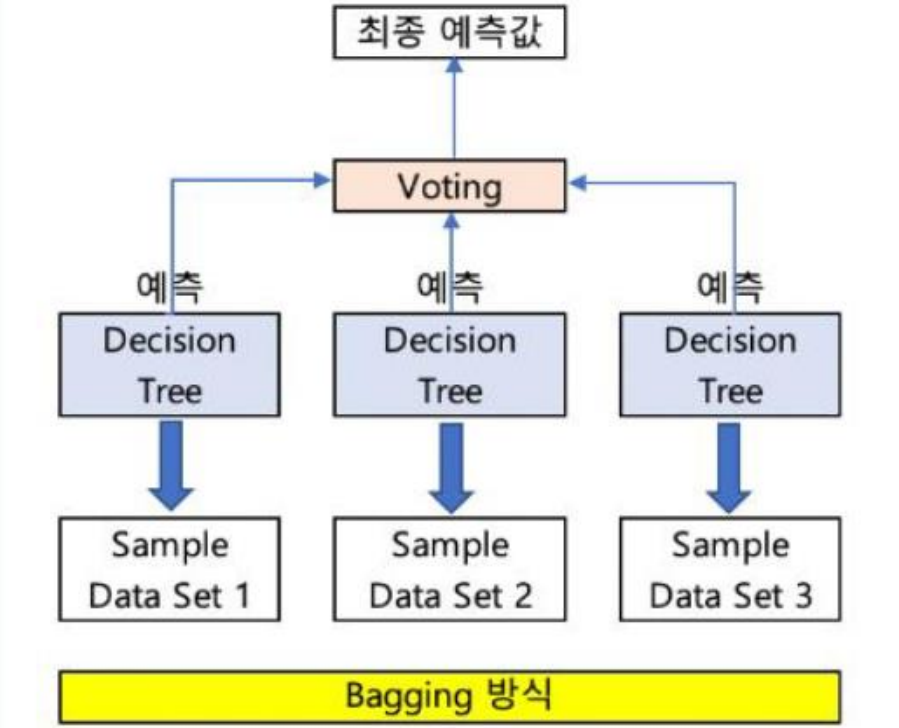


02 Bagging

보팅 (Voting)



배깅 (Bagging)



02 Bagging

장점

- 분산을 줄이는 효과

  -> 원래 추정 모델이 불안정하면 급격한 분산 감소 있을 수 있다!

Bagging 단점

1. 해석력 상실 : 약분류기를 tree 사용한다면, tree 는 더 이상 tree가 아니다… tree의 명확한 해석능력 상실
2. 계산 복잡도 : 본질적으로 한 모델 만드는 데에 B개의 부트스트랩 샘플의 사이즈 만큼 곱하는 중…

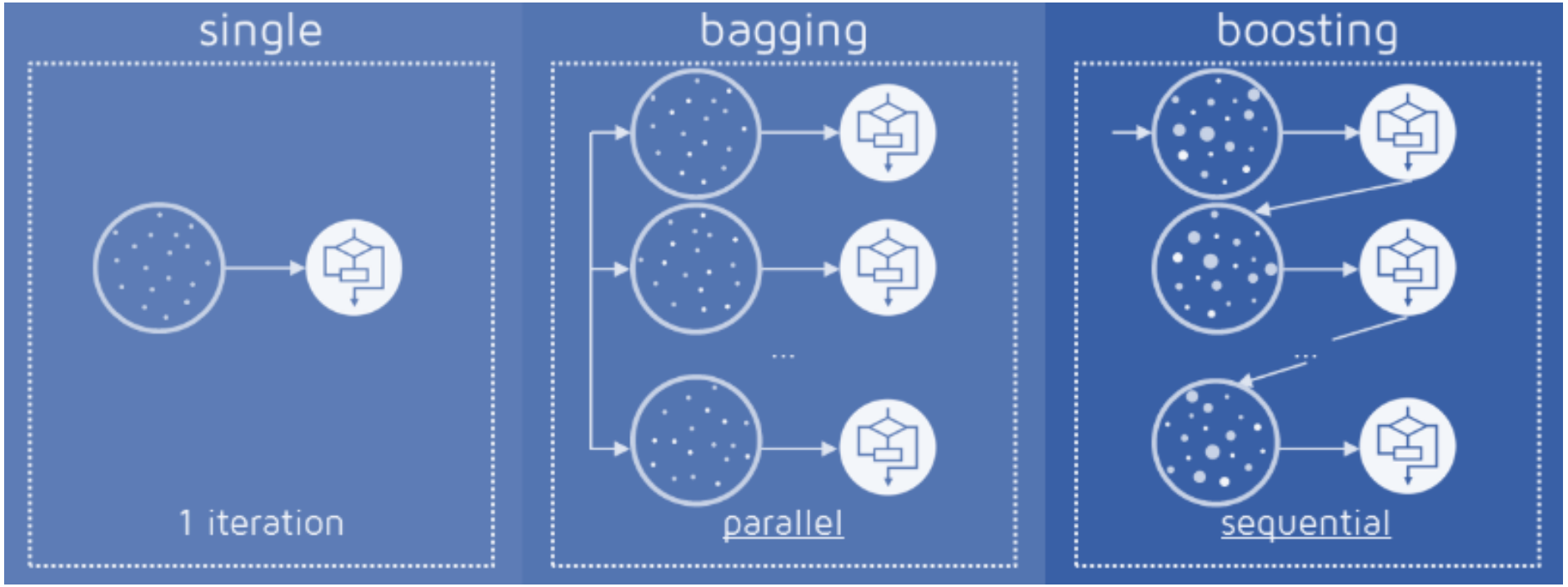
03. Boosting

1) AdaBoost
2) Gradient Boost
3) Extreme Gradient Boost
4) Light Gradient Boost

03 Boosting

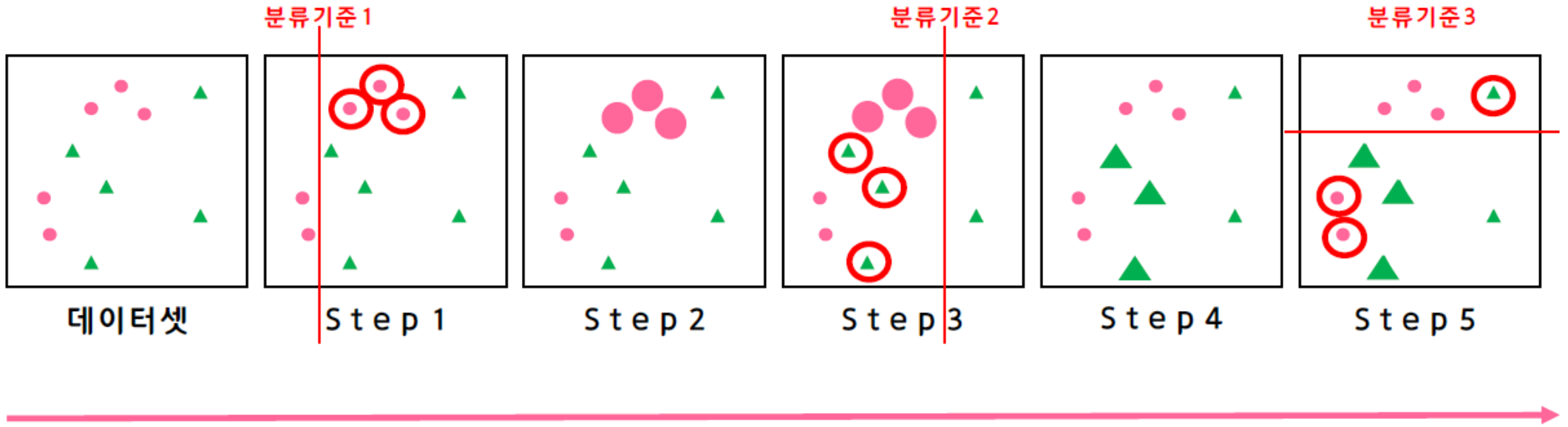
"약한 학습기 (weak learner) 를 순차적으로 학습 – 예측을 반복하면서
잘못 예측한 데이터에 가중치를 부여해서 오류를 개선하겠다!"

03 Boosting



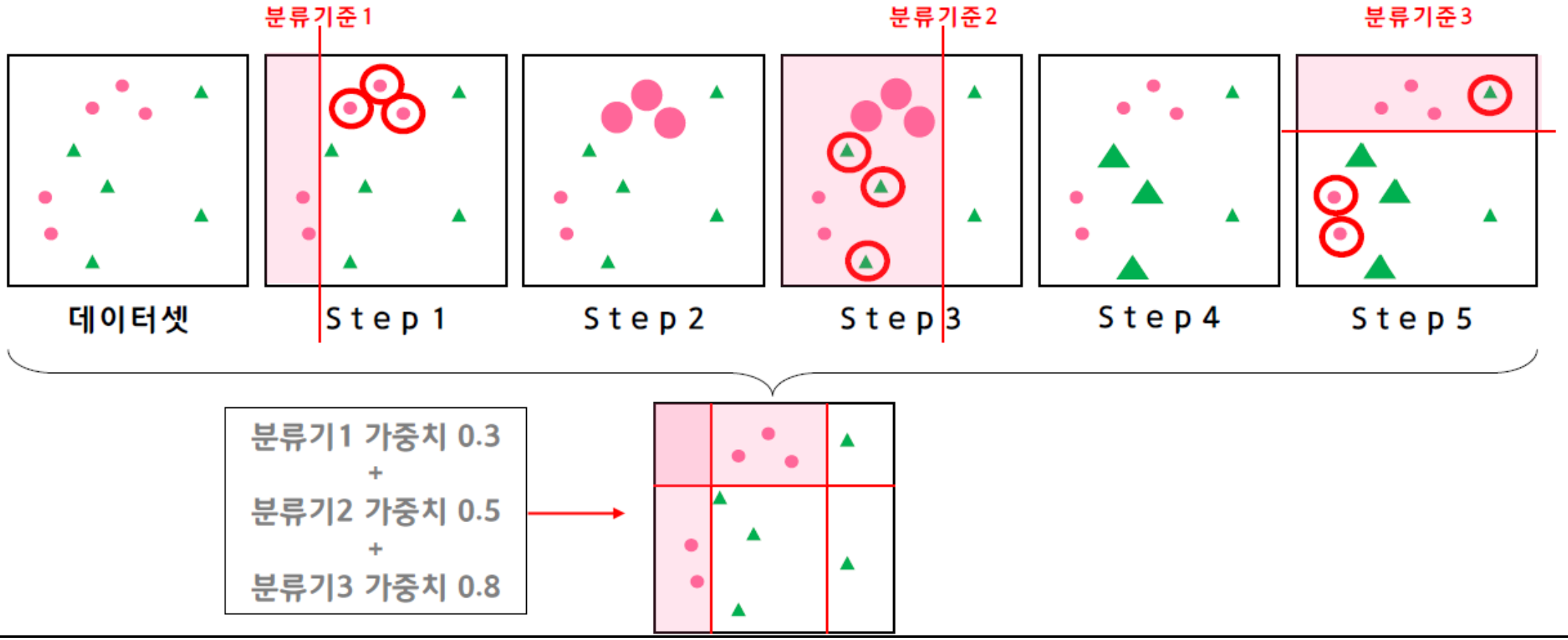
03 Boosting - AdaBoost

AdaBoost 에이다 부스트 Adaptive Boosting - 가중치 부여하면서 부스팅을 수행하는 대표적인 알고리즘



03 Boosting - AdaBoost

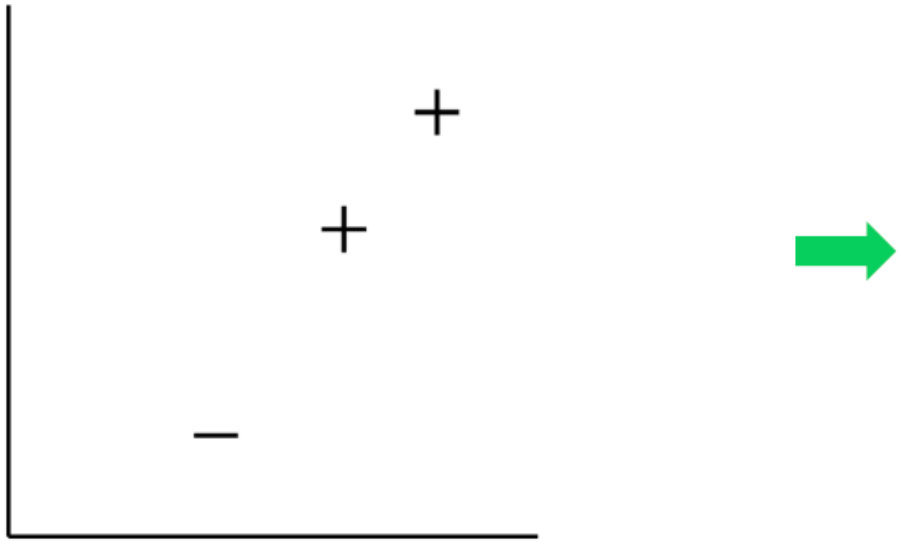
AdaBoost 에이다 부스트 Adaptive Boosting

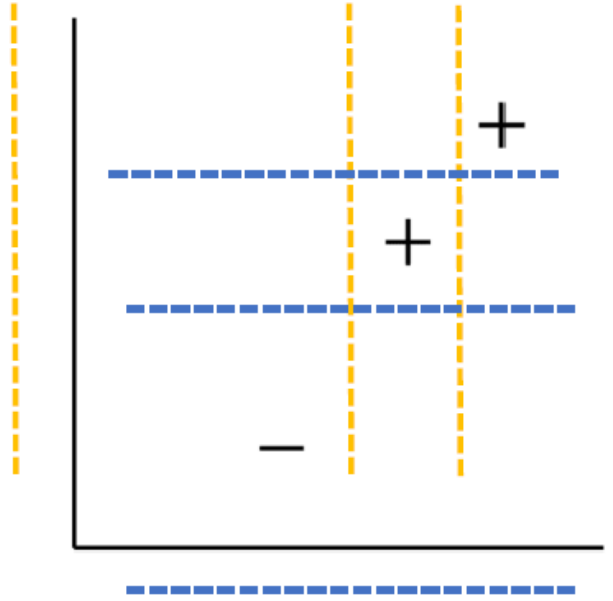


03 Boosting - AdaBoost

- 약한 분류기로 *Stump를 사용 (* Stump : 바로 leaf node만을 가지는 tree, 한번의 test만 가능)
- 이 stump들을 순차적으로 사용
- 처음에는 gini계수가 가장 낮은 feature의 stumps로 start!

**Stumps로 아래 데이터를 test할 수 있는 경우의 수





03 Boosting - AdaBoost

- Feature 수대로, weak learner(stumps)를 만들고, Gini Impurity 가 낮은 Stump 를 먼저 사용한다.
- 샘플 데이터의 가중치 초기화를 한다. $w_i = 1/n, \; i = 1, \ldots, n$

- 첫번째 weak learner 로 부터 예측한 값에서 error 계산
- 기존 Dataset의 가중치(C_m) 를 위에서 구한 err로 업데이트한다.
- Weak learner의 가중치 계산

$$err_m = \sum_{i=1}^{n} w_i \cdot \mathbb{I}(y_i \neq \hat{f}_m(\mathbf{x}_i))$$

$$c_m = \tfrac{1}{2} \log((1 - err_m)/err_m)$$

$$w_i = w_i \cdot \exp\left(- c_m \cdot y_i \cdot \hat{f}_m(\mathbf{x}_i)\right)$$

- 업데이트 된 데이터 가중치(C_m)를 고려하여 새로운 Dataset 을 만든다.
- 이 Dataset 으로 두번째 weak learner 학습

N번 반복!

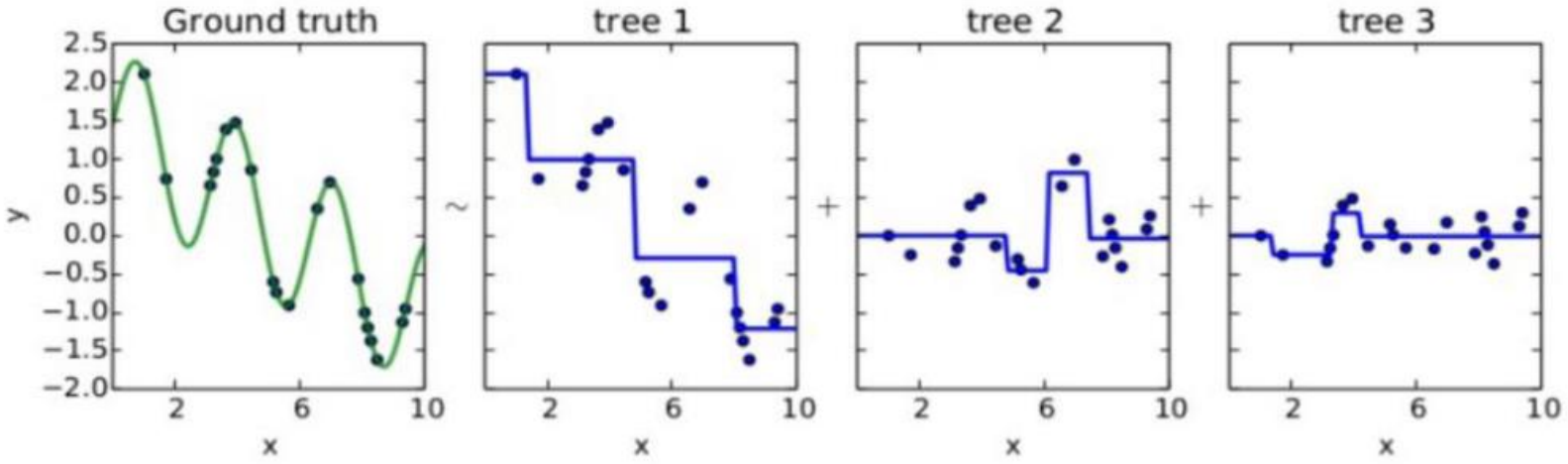
- N번의 iteration 후, N개의 weak learner들에 모델 가중치(w_i)를 주어 최종 모델을 만든다

03 Boosting – Gradient Boost

GBM => Gradient Boosting Machine

- residual을 줄이는 방향으로 weak learner들을 결합
- residual이 negative gradient와 같은 의미를 지니게 되므로 gradient boosting
- Weak learner가 tree (stump X)

Loss : $j(y_i, f(x_i)) = \frac{1}{2}(y_i - f(x_i))^2$ -> Residual : $y_i - f(x_i)$



03 Boosting – Gradient Boost

GBM 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

03 Boosting – Gradient Boost

GBM

Step1. 초기에는 평균값으로 모든 예측 값을 예측한다.

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

Average Weight

71.2

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

03 Boosting – Gradient Boost

GBM

Step2. 실제값과 예측값의 차이(잔차) 구하기

Average Weight

71.2

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

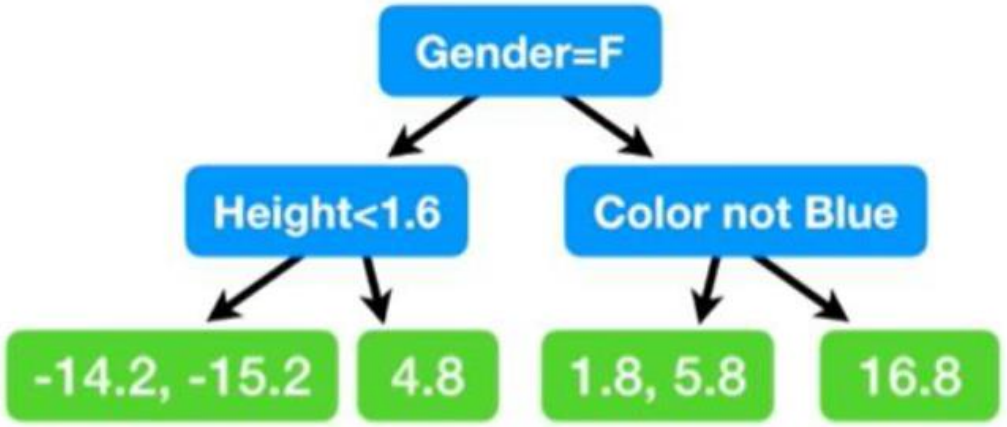
Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

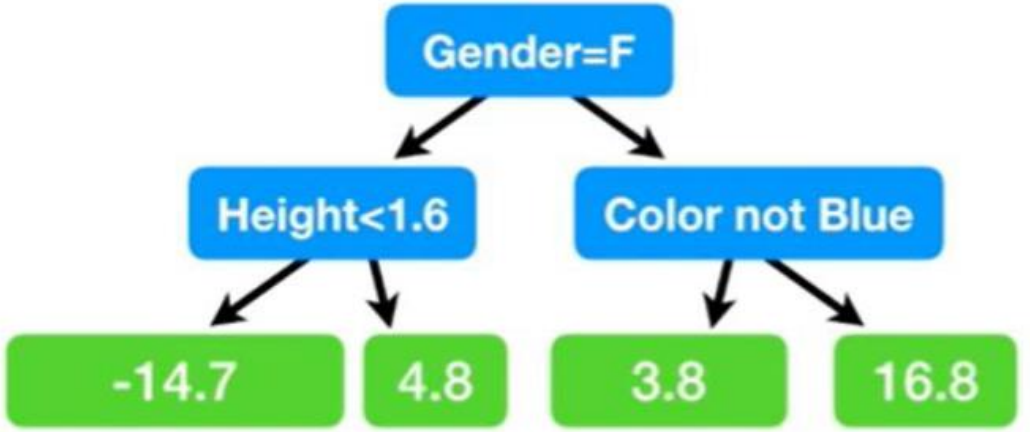
03 Boosting – Gradient Boost

GBM

Step3. 잔차를 예측하는 Tree를 만든다

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



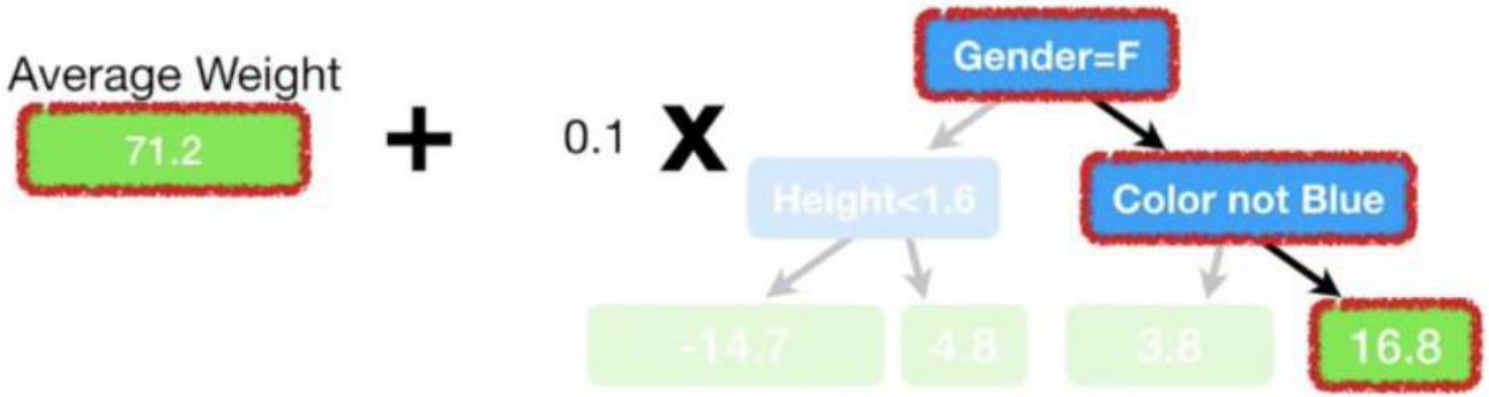


03 Boosting – Gradient Boost

GBM

Step4. 기존 예측값 + (잔차*learning rate) 로 새롭게 예측값 업데이트

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8

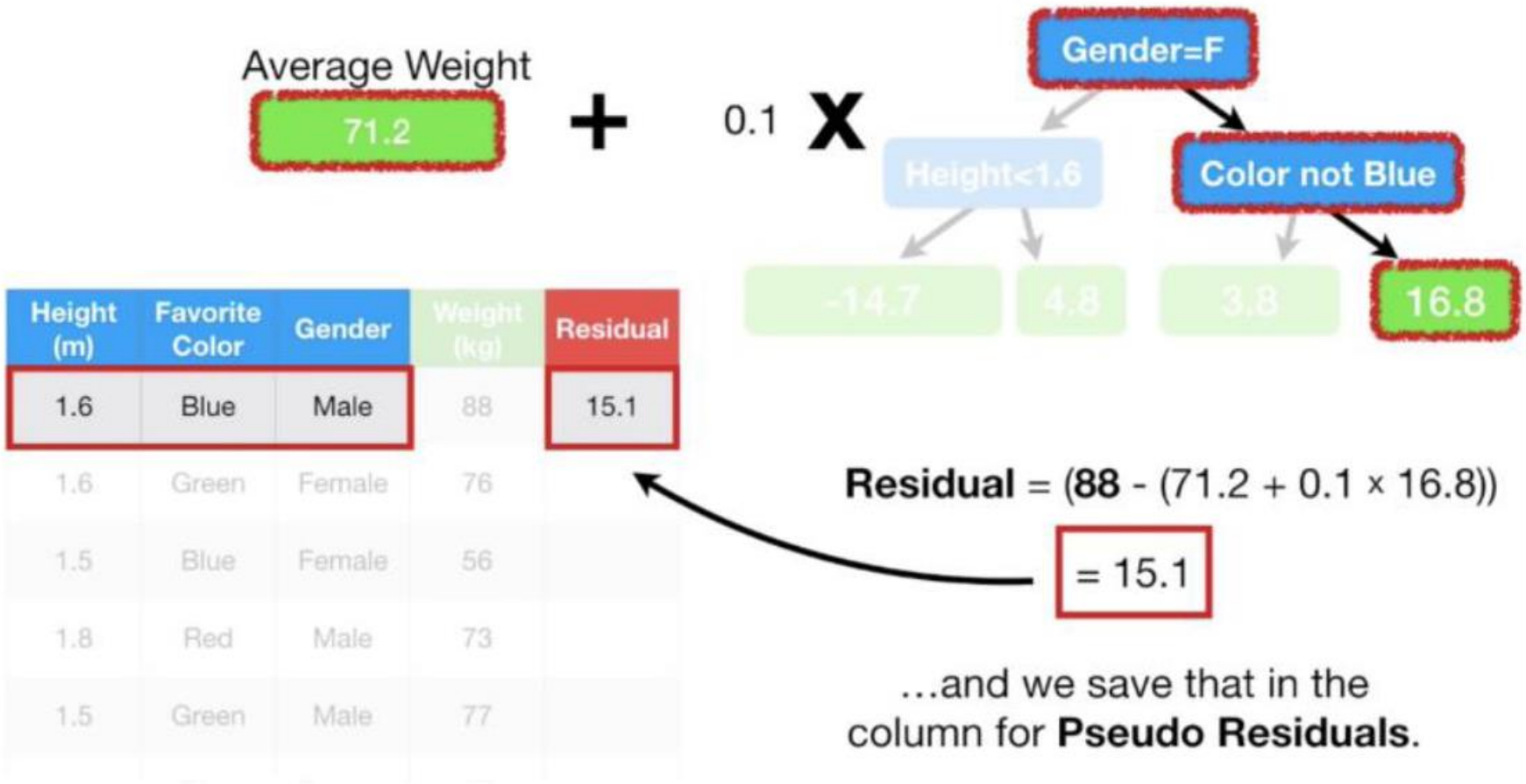


Now the **Predicted Weight** = 71.2 + (0.1 × 16.8) = 72.9

03 Boosting – Gradient Boost

GBM

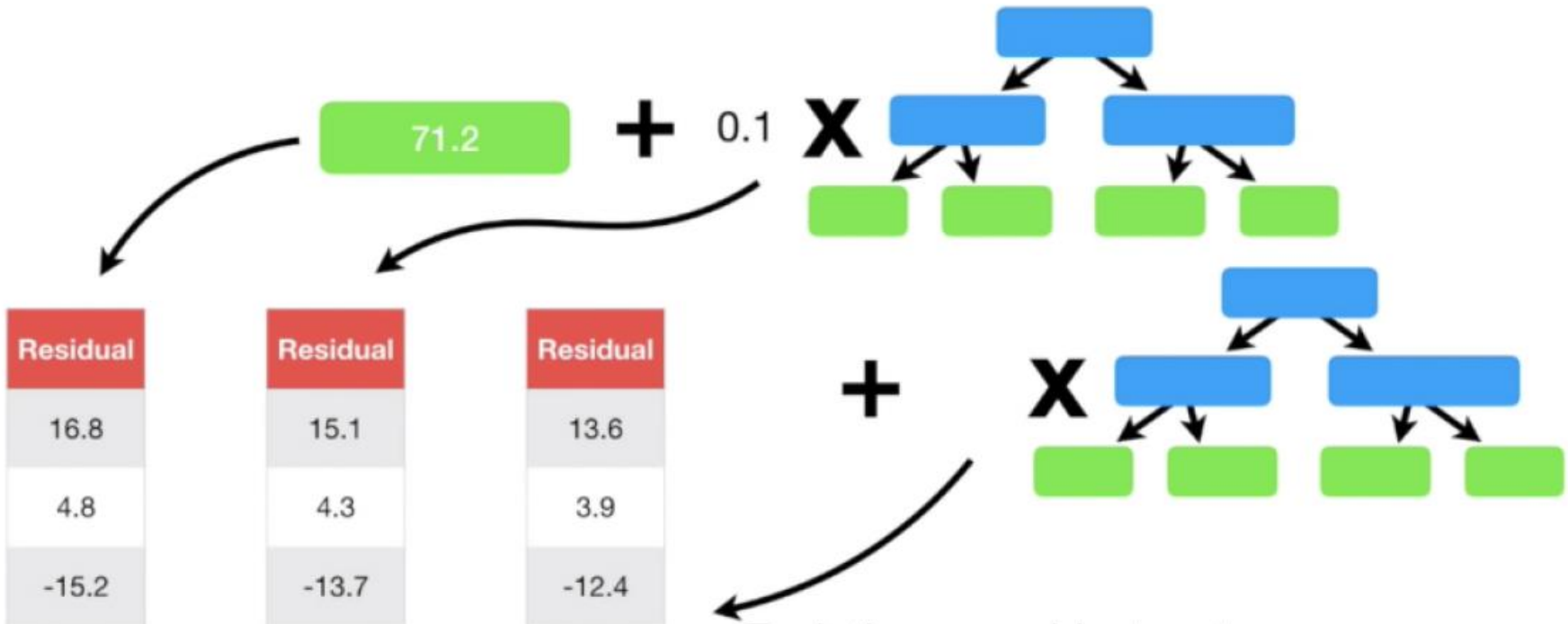
Step4. 기존 예측값 + (잔차*learning rate) 로 새롭게 예측값 업데이트



03 Boosting – Gradient Boost

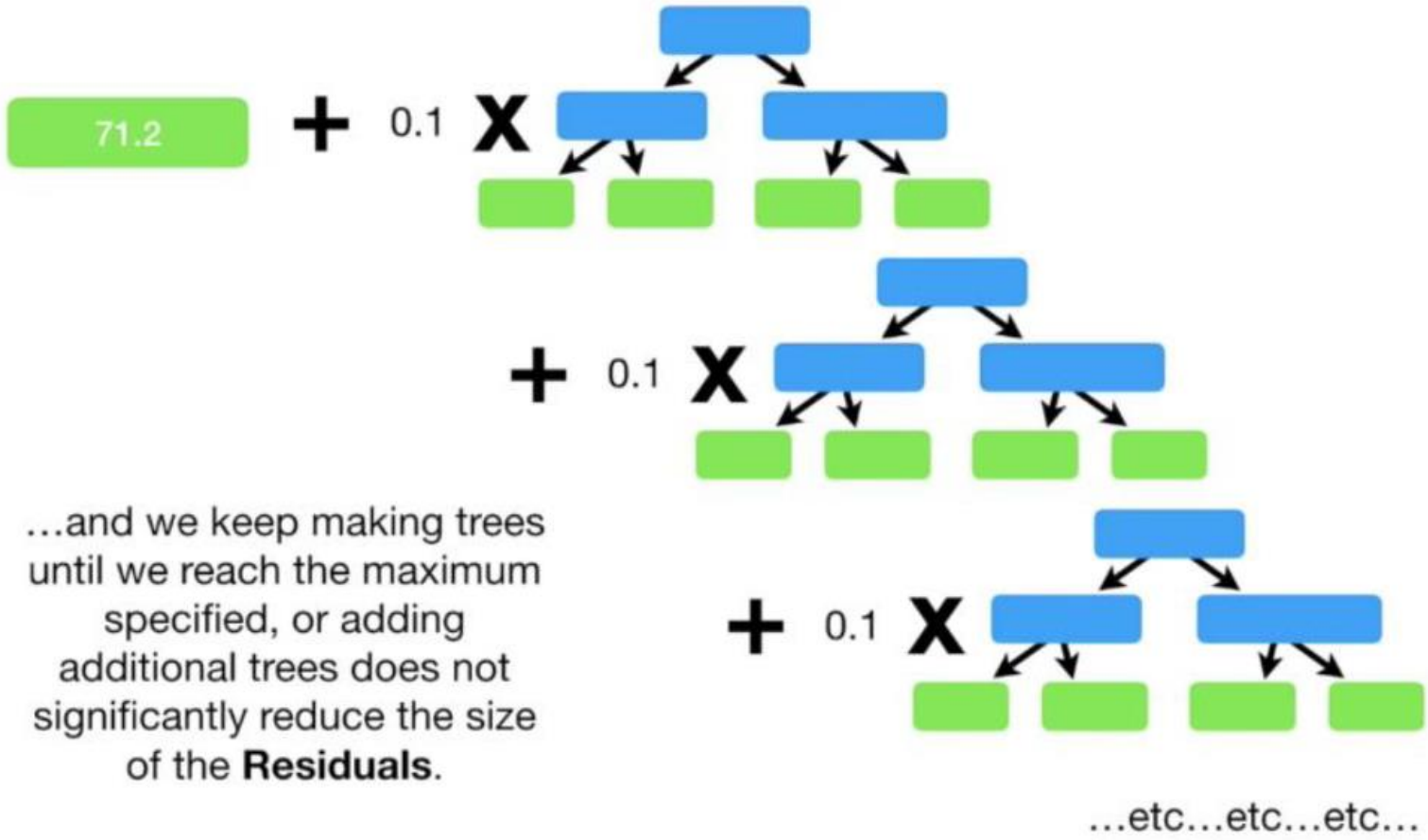
GBM

Step5. 위 step의 과정을 반복한다.



03 Boosting – Gradient Boost

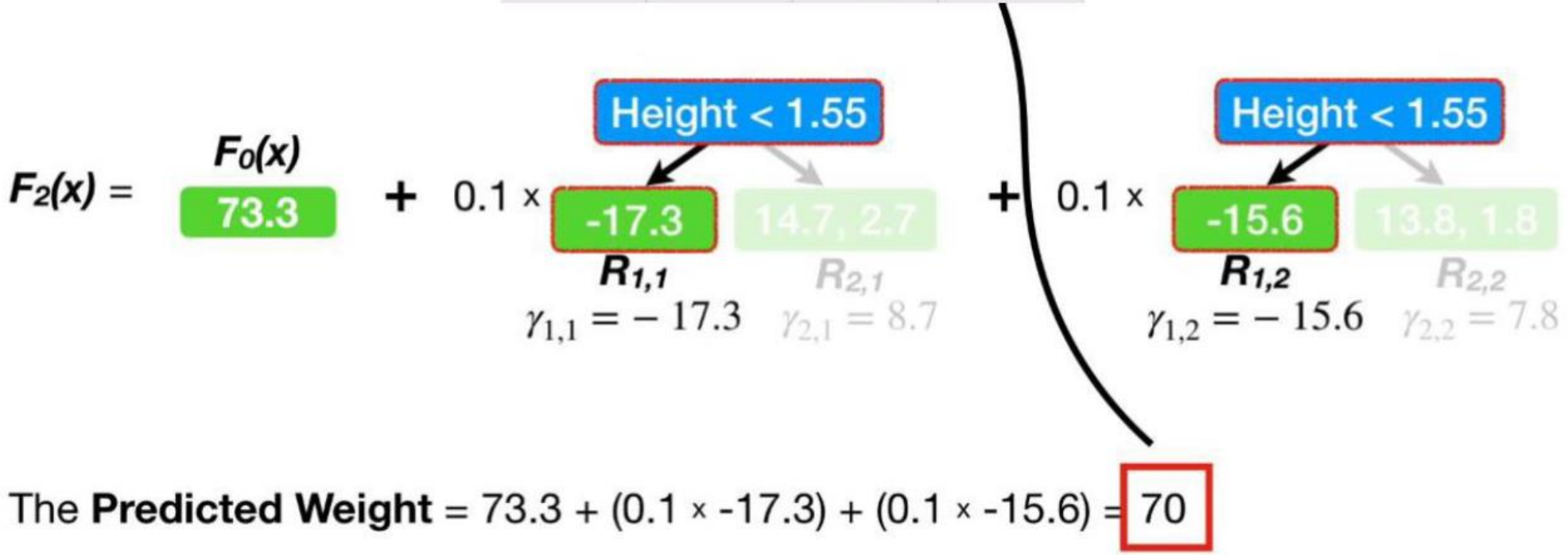
GBM 전체적인 구조



03 Boosting – Gradient Boost

GBM 예측

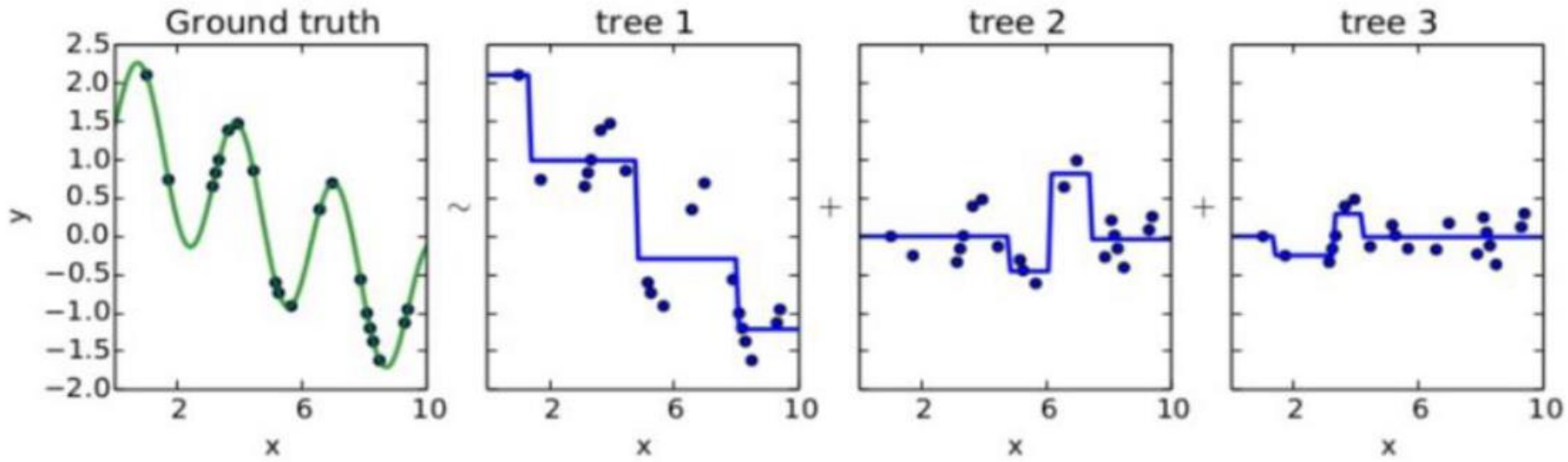
Height (m)	Favorite Color	Gender	Weight (kg)
1.4	Green	Male	???



The **Predicted Weight** = 73.3 + (0.1 × -17.3) + (0.1 × -15.6) = 70

03 Boosting – Gradient Boost

GBM



Data objective function : F(x), weak learner : A(x), B(x), C(x) … , residual : E 라고 하면,

$$F(x) = A(x) + E \text{ , } E = B(x) + E'$$

$$F(x) = A(x) + B(x) + E'$$

반복시, $F(x) = A(x) + B(x) + C(x) + \cdots$

03 Boosting – Gradient Boost

GBM => View of linear algebra

기저(Basis) : 어떤 벡터공간 V의 벡터들이 선형독립이면서 벡터공간 V 전체를 생성하는 벡터들의 집합

벡터공간 R^m 의 임의의 원소인 각 벡터 $\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$ 에 대해서

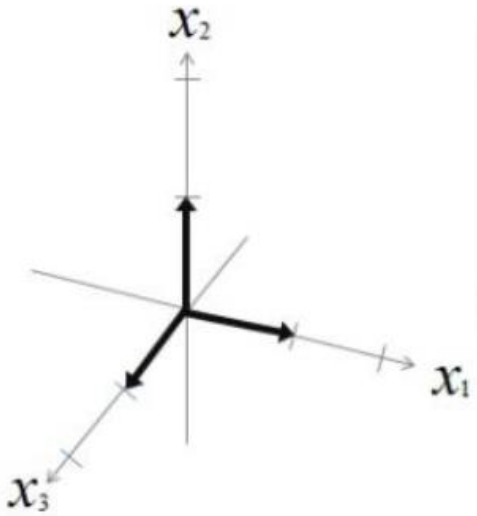
$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = c_1 \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{pmatrix} + c_2 \begin{pmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{pmatrix} + \cdots + c_n \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{pmatrix}$$ 의 해가

1조만 존재하는 경우, 집합 $\left\{ \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{pmatrix}, \begin{pmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{pmatrix}, \cdots, \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{pmatrix} \right\}$ 을

R^m 의 기저(basis)라고 함

R^3의 세벡터 $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

$$\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = c_1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + c_2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + c_3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$ 의 해는



$$\begin{cases} c_1 = 1 \\ c_2 = 2 \\ c_3 = 1 \end{cases}$$ 단 1개의 조만 존재하므로

집합 $\left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\}$ 은 R^3의 기저(basis) 임

03 Boosting – Gradient Boost

GBM => View of linear algebra

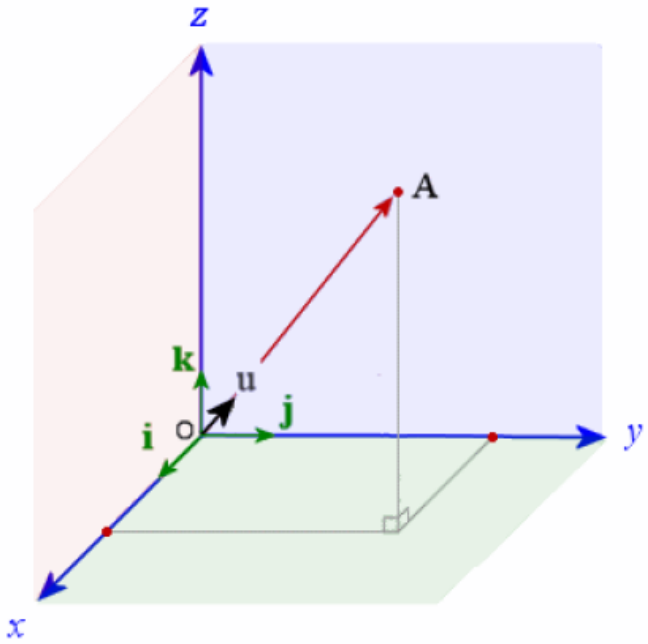
유클리디안 공간 > basis는 실수벡터, vector space > basis는 함수

$F(x) = A(x) + E$, $E = B(x) + E'$

$F(x) = A(x) + B(x) + E'$

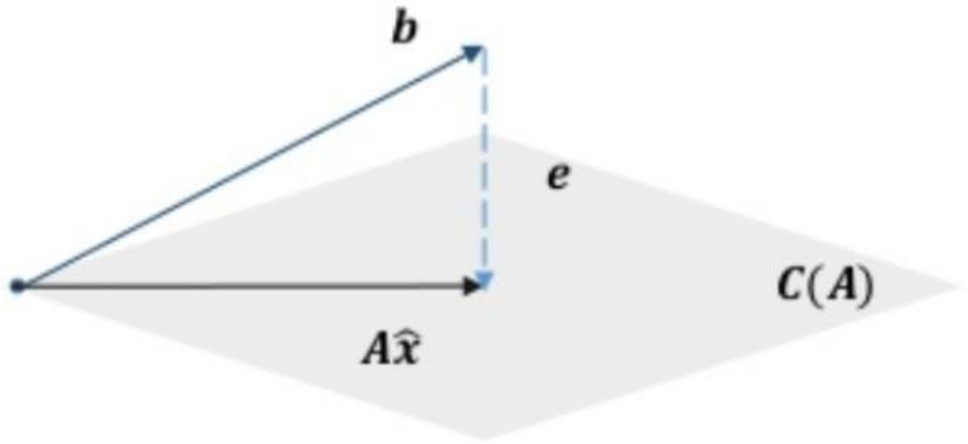
$$F(x) = A(x) + B(x) + C(x) + \cdots$$

weak learner들이 basis역할을 하고 있음



03 Boosting – Gradient Boost

b = A*x = A1*X1 + A2*X2 + … + An*Xn + E



03 Boosting – Extream Gradient Boost(XGBoost)

- GBM 좋은데, residual을 계속 줄이는 방향으로 학습하니까
- Overfitting 되기 쉽다

→ XGBoost = GBM + Regularization

03 Boosting – Extream Gradient Boost(XGBoost)

Xgboost => eXtreme Gradient Boosting, 2014

- GBM에 **규제(Regularization)**를 추가한 모델 > 과적합(overfitting) 방지
- 다양한 loss function 제공
- **병렬처리**를 통한 빠른 연산(노드 단위)

$$\text{XGBoost loss function} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

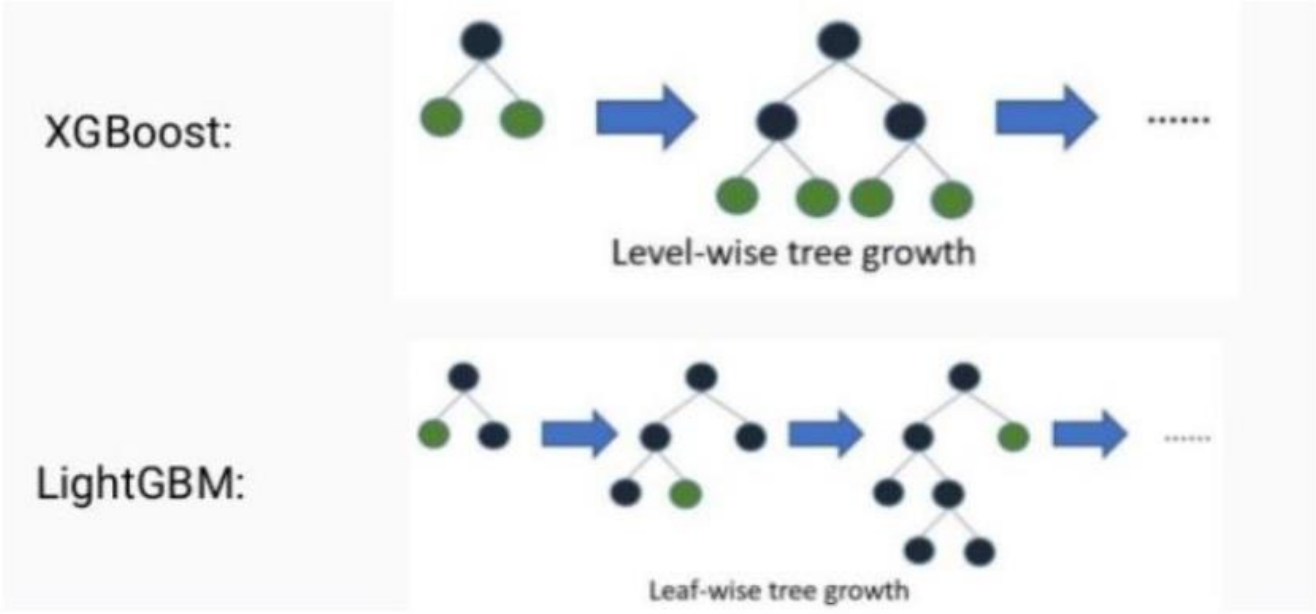
$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

- T : 트리의 최종node 개수, w: 최종node의 score

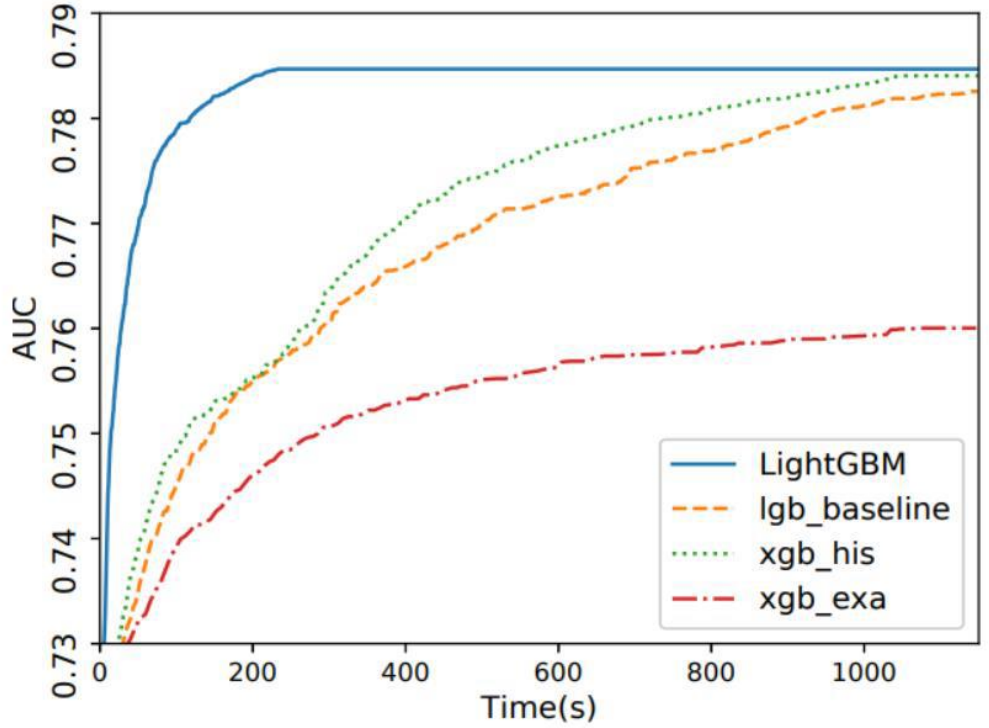
03 Boosting – Light Gradient Boost(LGBM)

Lightboost => Light GBM, 2016

- XGBoost 대비 성능향상 및 자원소모 최소화 (one side sampling 등)
- Level wise(균형 트리) 대신에 leaf wise 사용한다는 것이 특징
- XGBoost와 달리 리프 노드가 계속 분할되면서 트리의 깊이가 깊어지므로 , 이러한 트리에 맞는 하이퍼 파라미터 설정이 필요



03 Boosting – Light Gradient Boost(LGBM)

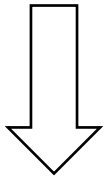


기능상의 다양성은 LightGBM 이 더 많고, 예측력은 LightGBM과 XGBoost의 성능이 별다른 차이가 없다!

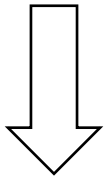
단점은 데이터의 수가 적을 때 과적합이 발생하기 쉽다는 것. (공식문서상 데이터 10000개 이하)

03 Boosting

Gradient Boost(GBM)



Extreme Gradient Boost(XGBoost)



Light Gradient Boost(LGBM)

04. Coding

01. XGBoost

중요도

'eta' : 학습률 (learning_rate), 보통 0.01~0.1 정도로 사용 ★★★

'max_depth' : 자식 노드를 파는 깊이를 정함. -1이면 끝까지 내려가지만 그만큼 Overfitting이 발생하므로 보통 5~12 정도로 많이 설정함. ★★★

'objective' - binary:logistic : 이항 분류 ★★★
- multi:softmax : 다항 분류
- reg:linear : 회귀 문제

'subsample' : bagging처럼 row 일부를 가져온다. Task마다 최적의 파라미터가 너무 다르기 때문에 tuning이 필요 ★★

01. XGBoost

	중요도
‘alpha’ : L1 Regularation	★
‘lambda’ : L2 Regularation	★
‘min_samples_split’ : 노드(잎)에 들어갈 샘플 데이터 수 설정.	☆
‘metric’	★★★

02. LightGBM

	중요도
 ‘feature_fraction’ : XGB의 colsample_bytree와 같음	★★
 ‘num_leaves’ : XGB의 num_leaves와 같음	★☆
 ‘n_estimators’ : 학습을 몇 번 반복할 건지 정함. 너무 과하면 당연히 overfitting. early stopping을 활용한다면 이런 과적을 방지할 수 있음.	★★★
 ‘early_stopping_rounds’ : 정한 valid loss에 대해, 더 이상의 진전이 없다면 학스은 머츠 n_estimators가 노다면 이거 피스저	★★

02. LightGBM

	중요도
 'lambda_l1' : XGB의 alpha와 같음	★
 'lambda_l2' : XGB의 lambda와 같음	★
 'min_data_in_leaf' : min_sample_split와 같음.	☆
 'metric'	★★★

Reference

- 건국대학교 응용통계학과 권성훈 교수님 행렬대수학, 데이터 마이닝
- 파이썬 머신러닝
- 파이썬 머신러닝 완벽가이드
- Freund, Y. and Schapire, R. E. (1997). A desicion-theoretic generalization of on-line learning and an application to boosting.
- https://datascienceschool.net/view-notebook/766fe73c5c46424ca65329a9557d0918/
- https://brunch.co.kr/@chris-song/98
- GBM https://dailyheumsi.tistory.com/116
- XGBoost https://www.slideshare.net/freepsw/boosting-bagging-vs-boosting
- https://brunch.co.kr/@snobberys/137
- https://statkclee.github.io/model/model-python-xgboost-hyper.html
- 부스팅 3모델 비교
- https://medium.com/@beverly.wang0005/gradient-tree-boosting-xgboost-vs-lightgbm-vs-catboost-part-2-275525458968
- Stacking python https://inspiringpeople.github.io/data%20analysis/Ensemble_Stacking/
- Stacking https://lsjsj92.tistory.com/559

감사합니다