앙상블

- 참고문헌: 파이썬 머신러닝 완벽가이드, 권철민, 위키북스, 2022 – **권수태 교수**
  데이터 과학을 위한 파이썬 머신러닝, 최성철, 한빛미디어, 2022
  StatQuest with Josh Starmer, Naïve Bayes

1. 앙상블(보팅,배깅,부스팅)

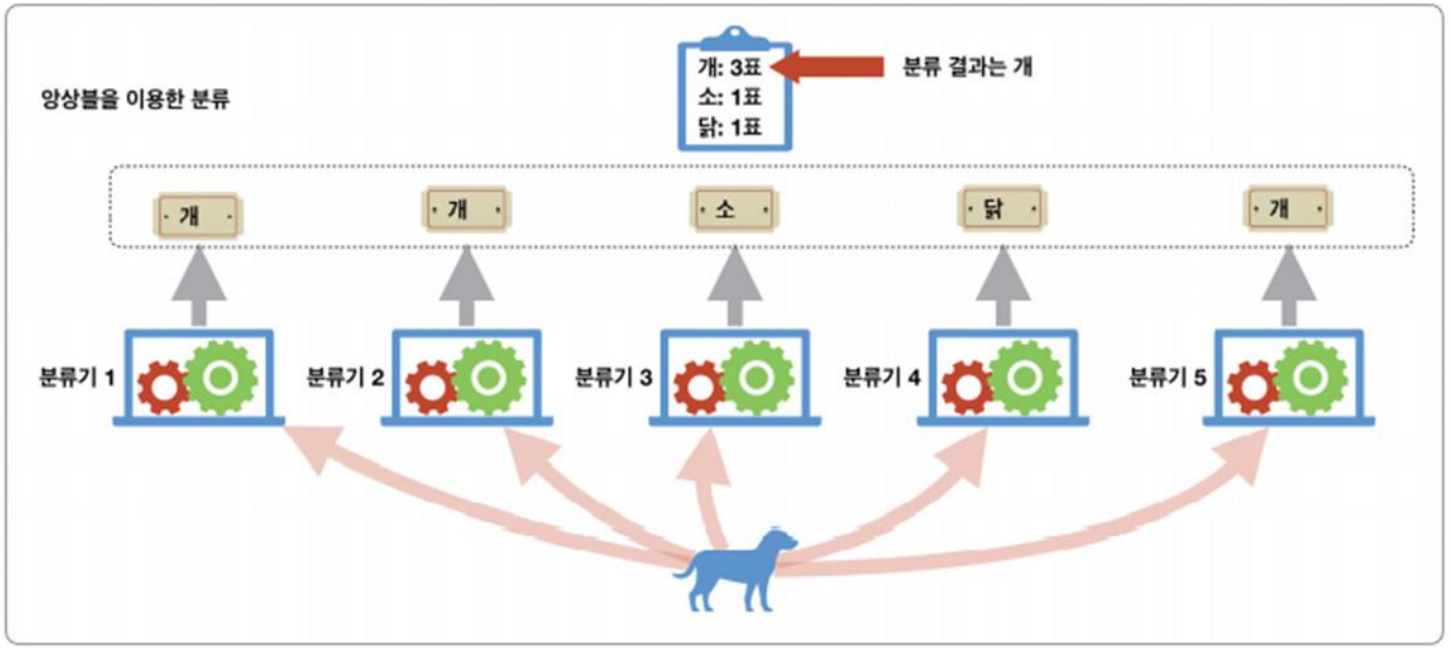
❖ 앙상블(Ensemble) 학습

- 어느 수준의 성능에 도달하면 분류기를 개선하는 일이 매우 어려워 짐
- 이를 해결하기 위한 앙상블Ensemble 기법은 여러 가지 종류의 분류기를 개별적으로 개선하는 데에 한계가 있을 때 이들의 협력을 이용하는 방법
- 앙상블이란 여러 개의 알고리즘을 사용하여, 그 예측을 결합함으로써 보다 정확한 예측을 도출하는 기법
- 집단지성이 힘을 발휘하는 것처럼 단일의 강한 알고리즘보다 복수의 약한 알고리즘이 더 뛰어날 수 있다는 생각에 기반을 둠
- 이미지, 영상, 음성 등의 비정형 데이터의 분류는 딥러닝이 뛰어난 성능을 보이지만, 대부분 정형 데이터의 분류에서는 앙상블이 뛰어난 성능을 보이고 있음

1. 앙상블(보팅,배깅,부스팅)

❖ 앙상블(Ensemble) 학습

➢ 여러 개의 분류기를 생성하고, 그 예측을 결합함으로써 보다 정확한 예측을 도출하는 기법



1. 앙상블(보팅,배깅,부스팅)

❖ 앙상블(Ensemble) 학습

- 앙상블 학습의 유형은 보팅(Voting), 배깅(Bagging), 부스팅(Boosting), 스태킹(Stacking) 등이 있음
- 보팅
  - ✓ 여러 종류의 알고리즘을 사용한 각각의 결과에 대해 투표를 통해 최종 결과를 예측하는 방식
- 배깅
  - ✓ 같은 알고리즘에 대해 데이터 샘플을 다르게 두고 학습을 수행해 보팅을 수행하는 방식
  - ✓ 이 때의 데이터 샘플은 중첩이 허용됨. 즉 10000개의 데이터에 대해 10개의 알고리즘이 배깅을 사용할 때, 각 1000개의 데이터 내에는 중복된 데이터가 존재할 수 있음
  - ✓ 배깅의 대표적인 방식이 Random Forest

1. 앙상블(보팅,배깅,부스팅)

❖ 앙상블(Ensemble) 학습

➢ 부스팅

✓ 여러 개의 알고리즘이 순차적으로 학습을 하되, 앞에 학습한 알고리즘 예측이 틀린 데이터에 대해 올바르게 예측할 수 있도록, 그 다음 번 알고리즘에 가중치를 부여하여 학습과 예측을 진행하는 방식

✓ 대표적인 방식은 그래디언트 부스트, XGBoost, LightGBM

➢ 스태킹

✓ 여러 가지 다른 모델의 예측 결과값을 다시 학습 데이터로 만들어 다른 모델(메타 모델)로 재 학습시켜 결과를 예측하는 방법

1. 앙상블(보팅,배깅,부스팅)

❖ 앙상블(Ensemble) 학습

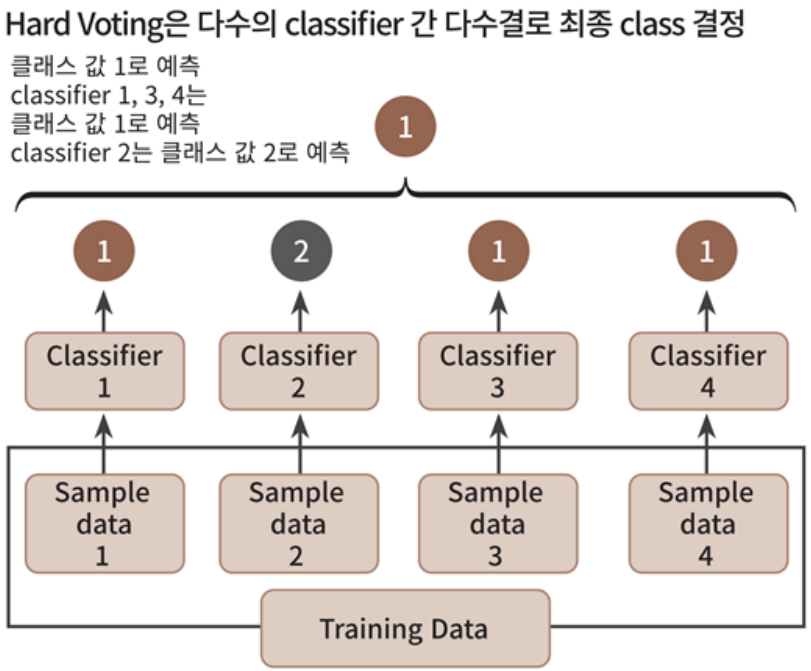
➢ 보팅

- ✓ 하드보팅(Hard Voting)과 소프트보팅(Soft Voting)
- ✓ 하드보팅을 이용한 분류는 다수결 원칙과 비슷
- ✓ 소프트 보팅은 각 알고리즘이 레이블 값 결정 확률을 예측해서, 이것을 평균하여 이들 중 확률이 가장 높은 레이블 값을 최종 값으로 예측
- ✓ 일반적으로는 소프트 보팅이 성능이 더 좋아서 많이 적용

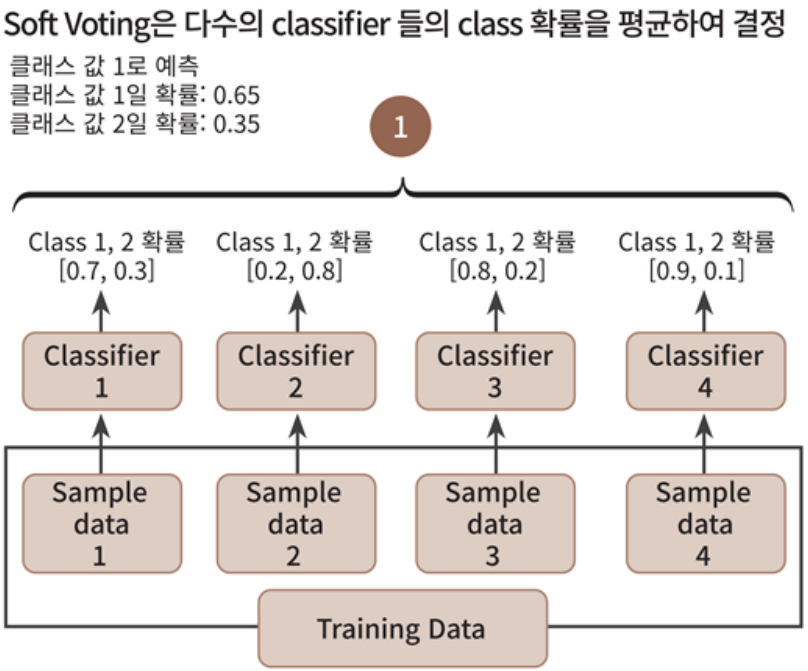
1. 앙상블(보팅,배깅,부스팅)

❖ 앙상블(Ensemble) 학습

➢ 보팅



< 하드 보팅 >



< 소프트 보팅 >

1. 앙상블(보팅,배깅,부스팅)

❖ 앙상블(Ensemble) 학습

➢ 보팅 분류기(Voting Classifier)

- ✓ 사이킷런은 보팅방식의 앙상블을 구현한 VotingClassifier 클래스를 제공
- ✓ 사이킷런에서 제공되는 위스콘신 유방암 데이터 세트를 이용해 보팅방식의 앙상블을 적용

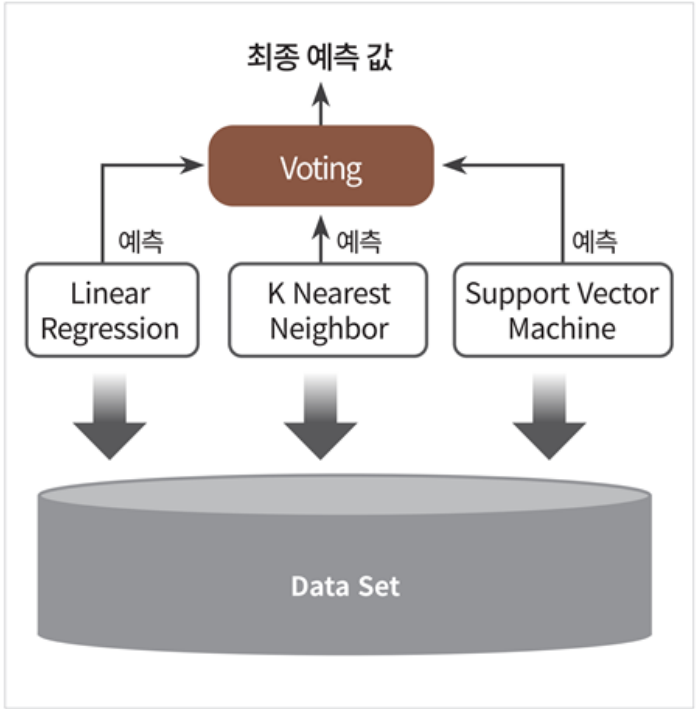
➢ 로지스틱회귀와 KNN을 기반으로 하여 소프트 보팅 방식으로 보팅 분류기를 만들기

- ✓ VotingClassifier 클래스를 이용해 보팅분류기 생성
- ✓ 생성인자로 estimators 와 voting 값 입력
- ✓ estimators는 리스트값으로 보팅에 사용될 여러 개의 Classifier 객체들을 튜플형식으로 입력
- ✓ voting 은 기본값이 hard

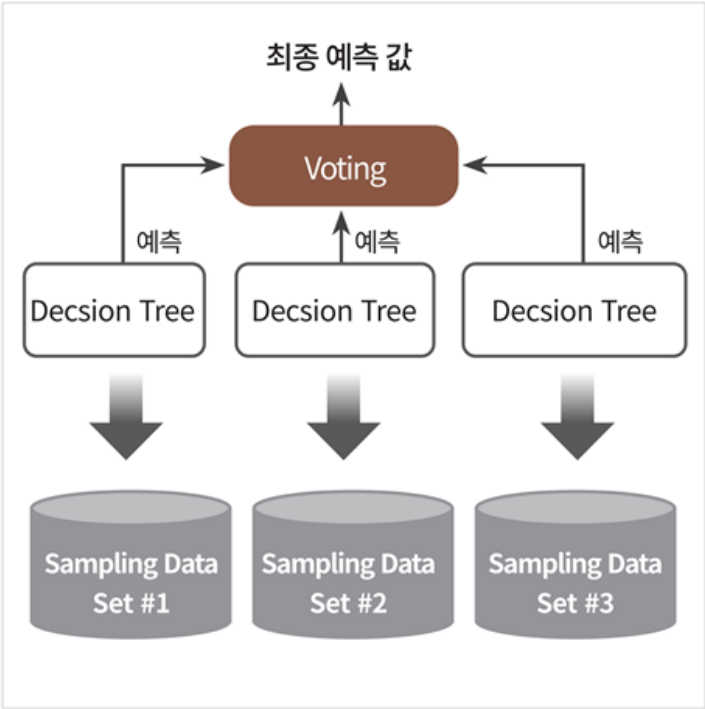
1. 앙상블(보팅,배깅,부스팅)

❖ 앙상블(Ensemble) 학습

➢ 보팅



Voting 방식



Bagging 방식

1. 앙상블(보팅,배깅,부스팅)

❖ 랜덤 포레스트

- 배깅(Bagging)은 Bootstrap Aggregating의 약자로, 보팅(Voting)과는 달리 동일한 알고리즘으로 여러 분류기를 만들어 보팅으로 최종 결정하는 알고리즘
- 배깅은 다음과 같은 방식으로 진행

  (1) 동일한 알고리즘을 사용하는 일정 수의 분류기 생성

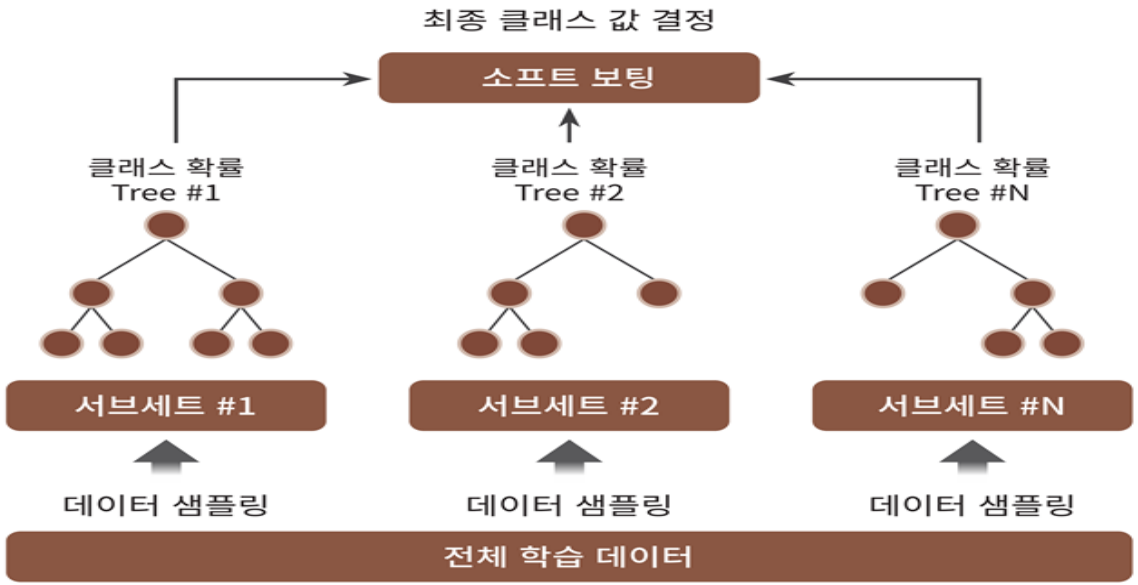
  (2) 각각의 분류기는 부트스트래핑(Bootstrapping)방식으로 생성된 샘플데이터를 학습

  (3) 최종적으로 모든 분류기가 보팅을 통해 예측 결정
- 부트스트래핑 샘플링은 전체 데이터에서 일부 데이터의 중첩을 허용하는 방식

1. 앙상블(보팅,배깅,부스팅)

❖ 랜덤 포레스트

➢ 랜덤 포레스트는 여러 개의 결정 트리 분류기가 전체 데이터에서 배깅 방식으로 각자의 데이터를 샘플링해 개별적으로 학습을 수행한 뒤 최종적으로 모든 분류기가 보팅을 통해 예측 결정



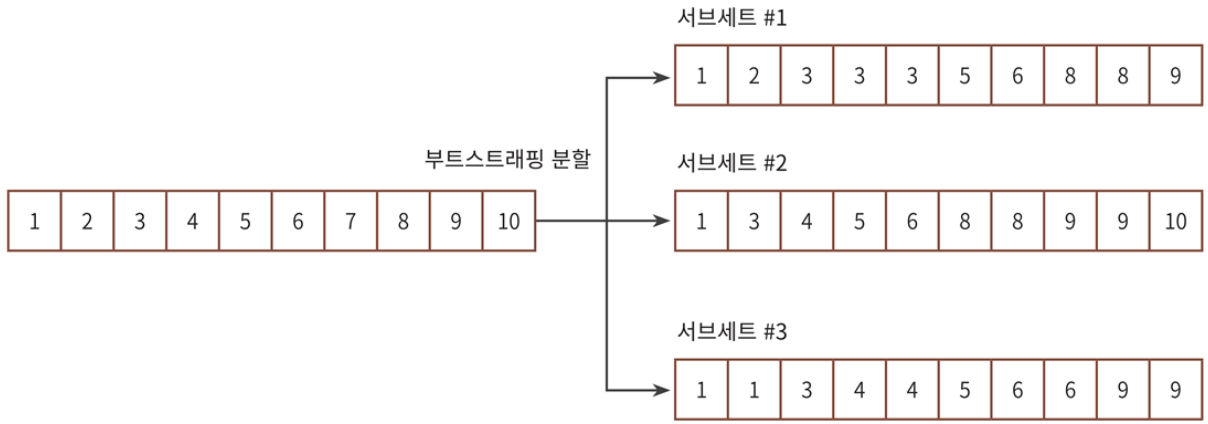
〈 랜덤 포레스트 〉

JEONJU UNIVERSITY

1. 앙상블(보팅,배깅,부스팅)

❖ 랜덤 포레스트

- 부트스트래핑 : 여러개의 데이터 세트를 중첩되게 분리하는 것
- 랜덤 포레스트의 서브세트 데이터는 부트스트래핑으로 데이터가 임의로 만들어짐
- 서브데이터의 데이터 건수는 전체데이터의 건수와 동일하지만 개별데이터가 중첩되어 만들어짐

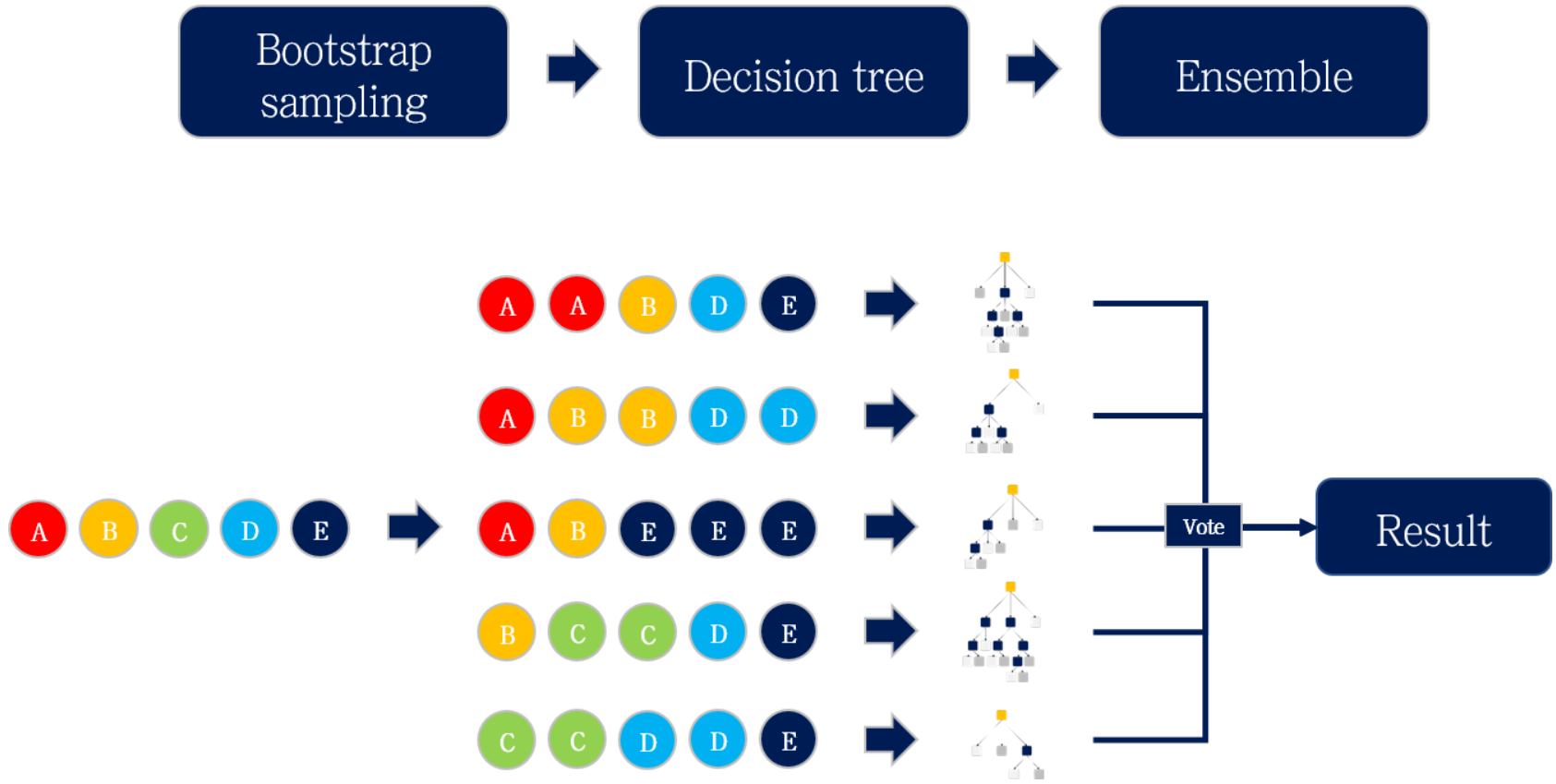


〈 부트스트래핑 샘플링 방식 〉

JEONJU UNIVERSITY

1. 앙상블(보팅,배깅,부스팅)

❖ 랜덤 포레스트



JEONJU UNIVERSITY

1. 앙상블(보팅,배깅,부스팅)

❖ 랜덤 포레스트

➢ 장점

- ✓결정 트리의 쉽고 직관적인 장점을 그대로 가지고 있음
- ✓앙상블 알고리즘 중 비교적 빠른 수행 속도를 가지고 있음
- ✓다양한 분야에서 좋은 성능을 나타냄

➢ 단점

- ✓하이퍼 파라미터가 많아 튜닝을 위한 시간이 많이 소요됨

1. 앙상블(보팅,배깅,부스팅)

❖ 랜덤 포레스트

➢ 랜덤포레스트 하이퍼 파라미터 튜닝

- ✓ 랜덤포레스트는 트리기반의 하이퍼 파라미터에 배깅, 부스팅, 학습, 정규화 등을 위한 하이퍼 파라미터까지 추가됨
- ✓ n_estimators: 결정트리의 갯수를 지정. Default = 10. 무작정 트리 갯수를 늘리면 성능 좋아지는 것 대비 시간이 걸릴 수 있음
- ✓ min_samples_split: 노드를 분할하기 위한 최소한의 샘플 데이터수. 과적합을 제어하는데 사용. Default = 2(작게 설정할 수록 분할 노드가 많아져 과적합 가능성 증가)
- ✓ min_samples_leaf: 리프노드가 되기 위해 필요한 최소한의 샘플 데이터수. min_samples_split과 함께 과적합 제어 용도. 불균형 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 작게 설정 필요

1. 앙상블(보팅,배깅,부스팅)

❖ 랜덤 포레스트

➢ 랜덤포레스트 하이퍼 파라미터 튜닝

✓ max_features: 최적의 분할을 위해 고려할 최대 feature 개수. Default = 'auto' (결정트리에서는 default가 none이었음)

- int형으로 지정 →피처 갯수 / float형으로 지정 →비중
- sqrt 또는 auto : 전체 피처 중 √(피처개수) 만큼 선정
- log : 전체 피처 중 log2(전체 피처 개수) 만큼 선정

✓ max_depth: 트리의 최대 깊이. default = None (완벽하게 클래스 값이 결정될 때 까지 분할 또는 데이터 개수가 min_samples_split보다 작아질 때까지 분할)

- 깊이가 깊어지면 과적합될 수 있으므로 적절히 제어 필요

✓ max_leaf_nodes: 리프노드의 최대 개수

1. 앙상블(보팅,배깅,부스팅)

➢부스팅(Boosting)

✓여러 개의 약한 학습기(weak learner)를 순차적으로 학습-예측하면서 잘못 예측한 데이터에 가중치를 부여해 오류를 개선해 나가며 학습하는 방식

✓부스팅 알고리즘 종류

- AdaBoost
- Gradient Booting Machine(GBM)
- XGBoost
- LightGBM
- CatBoost

1. 앙상블(보팅,배깅,부스팅)

❖ 부스팅(Boosting)

➢ 알고리즘

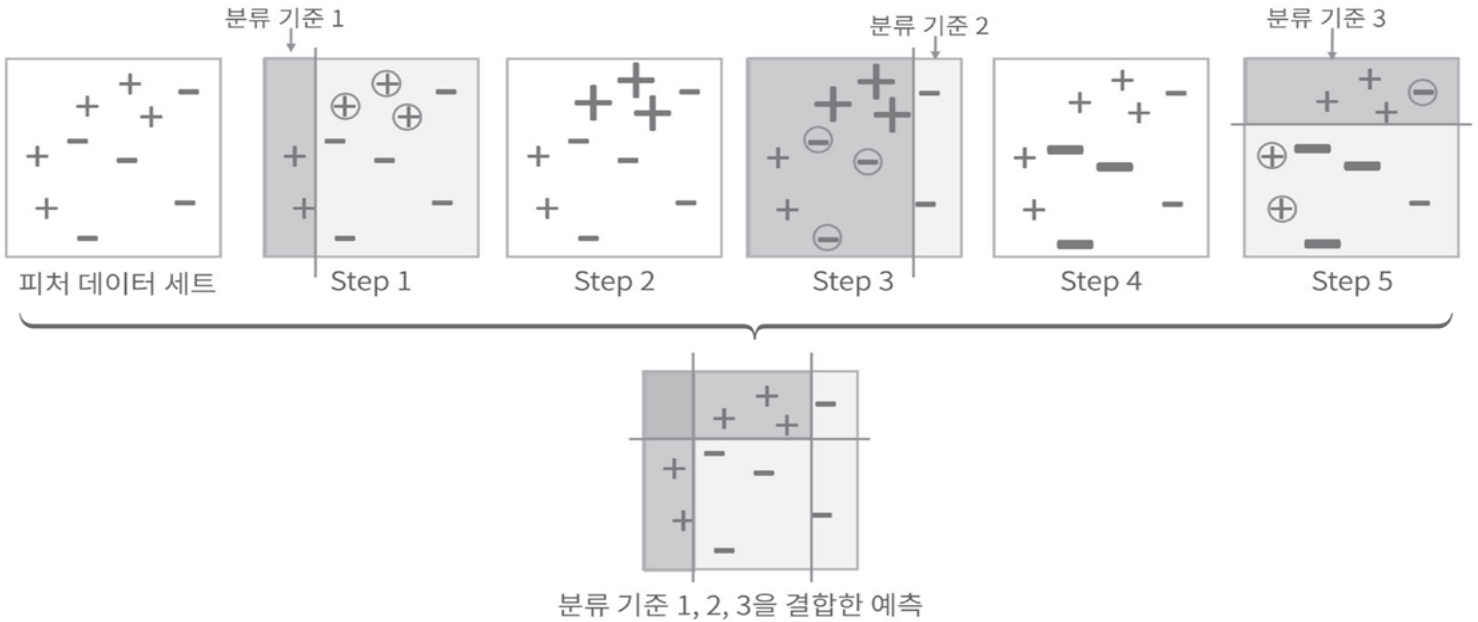
<Boosting Algorithm>

1. 앙상블(보팅,배깅,부스팅)

➢AdaBoost

✓Adaptive Boost의 줄임말로서 약한 학습기(weak learner)의 오류 데이터에 가중치를 부여하면서 부스팅을 수행하는 대표적인 알고리즘

✓속도나 성능적인 측면에서 decision tree를 약한 학습기로 사용함



JEONJU UNIVERSITY

1. 앙상블(보팅,배깅,부스팅)

❖AdaBoost

➢ 절차

Step 1) 첫 번째 약한 학습기가 첫번째 분류기준(D1)으로 + 와 - 를 분류

Step 2) 잘못 분류된 데이터에 대해 가중치를 부여(두 번째 그림에서 커진 + 표시)

Step 3) 두 번째 약한 학습기가 두번째 분류기준(D2)으로 +와 - 를 다시 분류

Step 4) 잘못 분류된 데이터에 대해 가중치를 부여(세 번째 그림에서 커진 - 표시)

Step 5) 세 번째 약한 학습기가 세번째 분류기준으로(D3) +와 -를 다시 분류해서 오류 데이터를 찾음

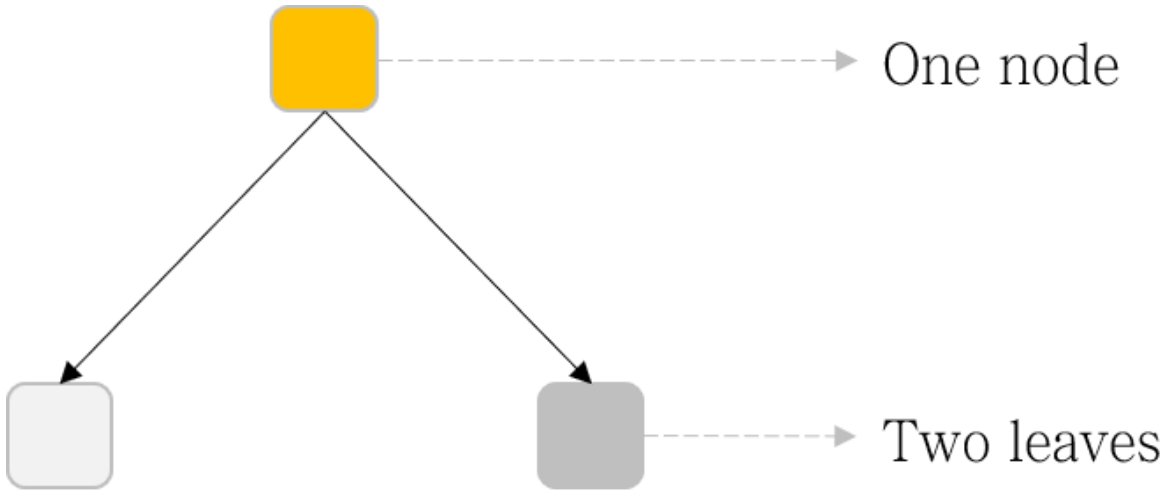
Step 6) 마지막으로 분류기들을 결합하여 최종 예측 수행

→ 약한 학습기를 순차적으로 학습시켜, 개별 학습기에 가중치를 부여하여 모두 결합함으로써 개별 약한 학습기보다 높은 정확도의 예측 결과를 만듦

1. 앙상블(보팅,배깅,부스팅)

❖ AdaBoost

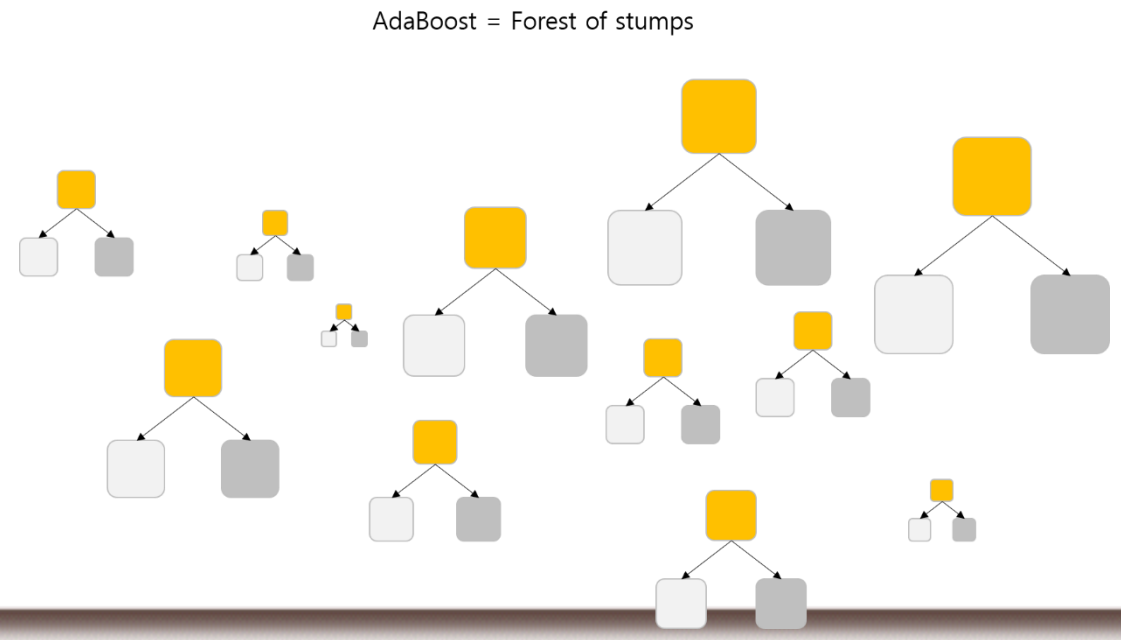
- ➢ Random forest에서는 개별 모델로 decision tree를 사용한 반면, AdaBoost에서는 개별 모델로 stump를 사용
- ➢ Stump란 한 노드와 두 개의 가지를 갖는 decision tree



1. 앙상블(보팅,배깅,부스팅)

❖ AdaBoost

- ➢ Random forest에서 모든 tree는 동일한 weight를 가지는 반면 AdaBoost에서 각 stump는 서로 다른 weight (amount of say)를 가짐
- ➢ 즉, 최종 앙상블 시, 개별 모델마다 예측 결과값에 가중치를 달리 부여

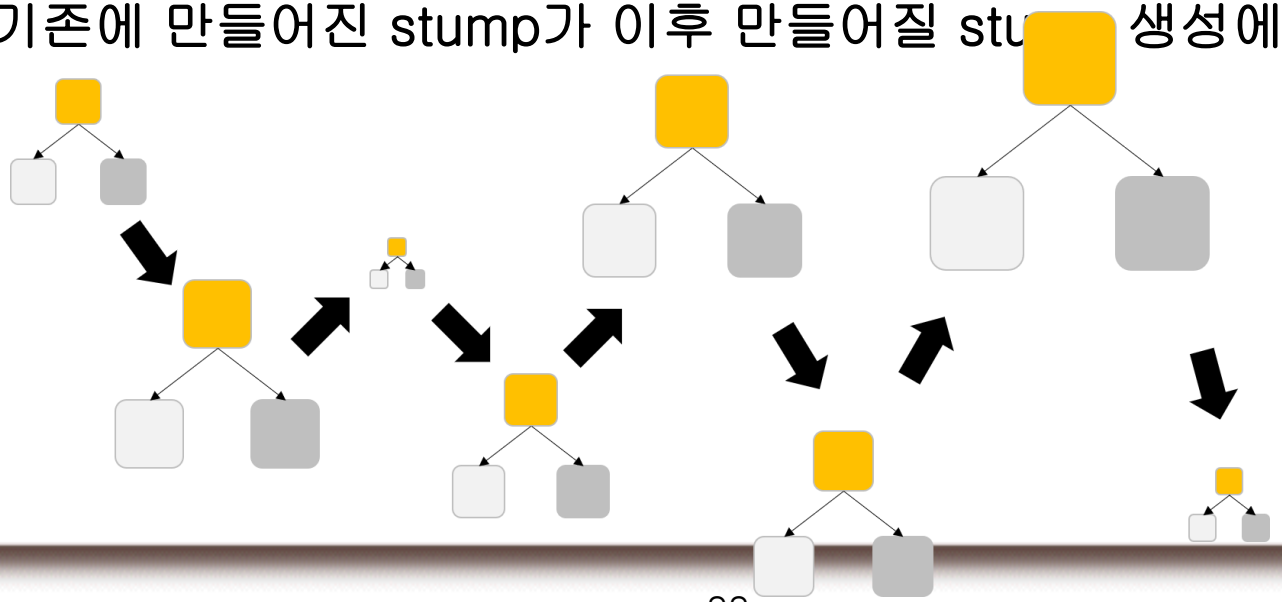


JEONJU UNIVERSITY

1. 앙상블(보팅,배깅,부스팅)

❖ AdaBoost

- Random forest에서 tree들은 parallel하게 만들어 짐(즉, decision tree 간 서로 영향이 없음)
- 반면, AdaBoost에서 stump들은 sequential하게 만들어진다는 특징을 가짐
- 즉, 기존에 만들어진 stump가 이후 만들어질 stu 생성에 영향을 줌



1. 앙상블(보팅,배깅,부스팅)

- Gradient Boost Machine(GBM)
  - GBM의 학습 방식
    - ✓ AdaBoost와 유사하지만, 가중치 업데이트를 경사하강법(Gradient Descent)를 이용하여 최적화된 결과를 얻는 알고리즘
    - ✓ GBM은 예측 성능이 높지만 Greedy Algorithm으로 과적합이 빠르게 되고, 시간이 오래 걸린다는 단점이 있음
  - Greedy Algorithm(탐욕 알고리즘)
    - ✓ 미래를 생각하지 않고 각 단계에서 가장 최선의 선택을 하는 기법으로,
    - ✓ 각 단계에서 최선의 선택이 전체적으로도 최선이길 바라는 알고리즘
    - ✓ 예) 지금 선택하면 1개 마시멜로, 1분 기다렸다 선택하면 2개 마시멜로를 받는 문제에서는 1개의 마시멜로 선택
  - GBM 은 CART 기반의 다른 알고리즘과 마찬가지로 분류는 물론이고 회귀도 가능
  - 사이킷런은 GBM 기반의 분류를 위해 GradientBoostingClassifier 클래스를 제공

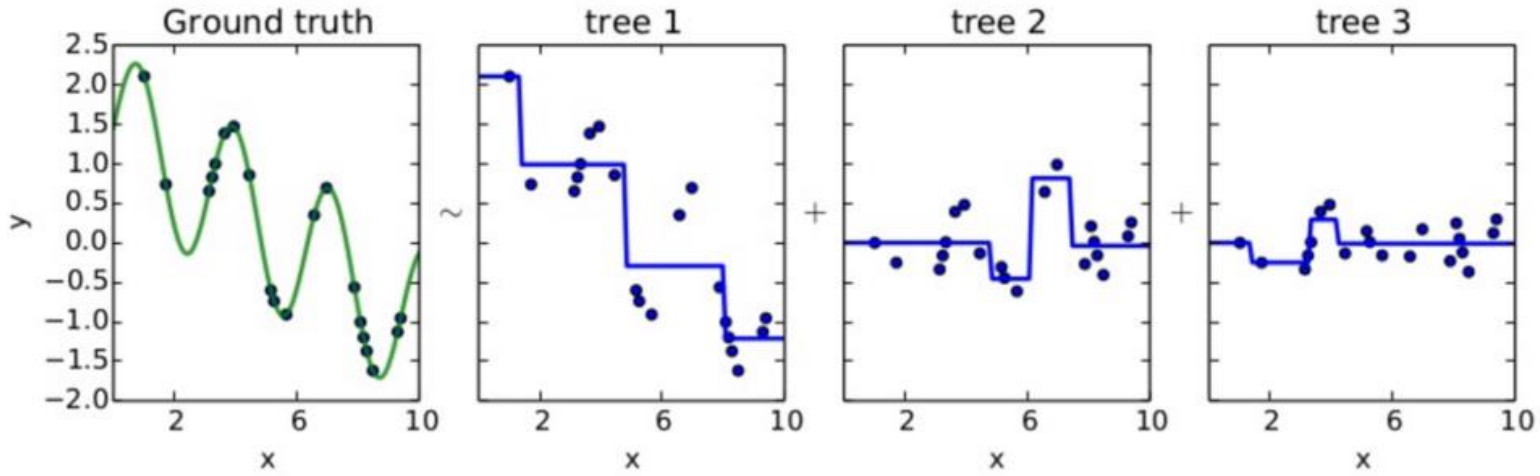
1. 앙상블(보팅,배깅,부스팅)

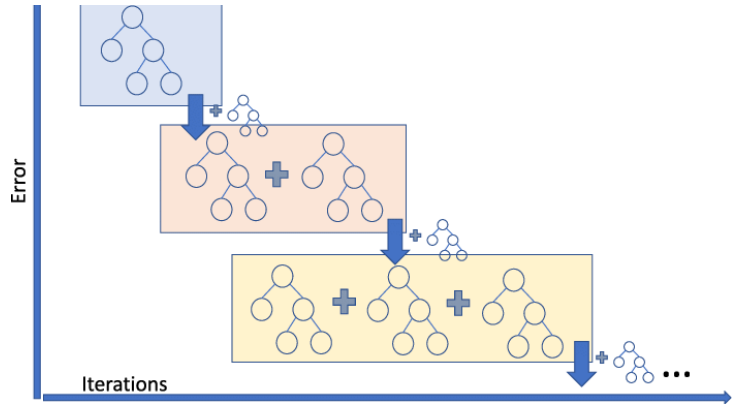
❖ Gradient Boost Machine(GBM)

- ➢ 부스팅 앙상블의 대표적인 특징은 모델 학습이 sequential. 즉, 먼저 생성된 모델의 예측값이 다음 모델 생성에 영향을 줌
- ➢ Single leaf 로 시작하여, 이후 각 단계에서 이전 tree 의 error 를 반영한 새로운 tree 구축
  - ✓ Tree 는 이전 단계에서 발생한 residual 을 예측하는 방식으로 학습
- ➢ Residual fitting 으로 이해
  - ✓ 간단한 모델 A 를 통해 y 를 예측하고, 남은 잔차를 다시 B라는 모델을 통해 예측하고 A+B 모델을 통해 y 를 예측
  - ✓ tree 1 을 통해 예측하고 남은 잔차를 tree 2를 통해 예측하고, 이를 반복하여 점점 잔차를 줄여 나감

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)





JEONJU UNIVERSITY

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ AdaBoost 와의 차이

- ✓ Weak learner: Stumps VS A leaf & Restricted trees
- ✓ Predicted value: Output VS Pseudo-residual
- ✓ Model weight: Different model weights (amount of say) VS Equal model weight (learning rate)

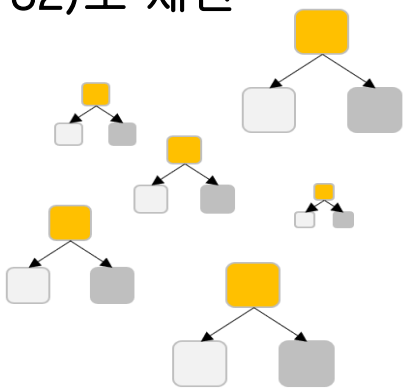
1. 앙상블(보팅,배깅,부스팅)

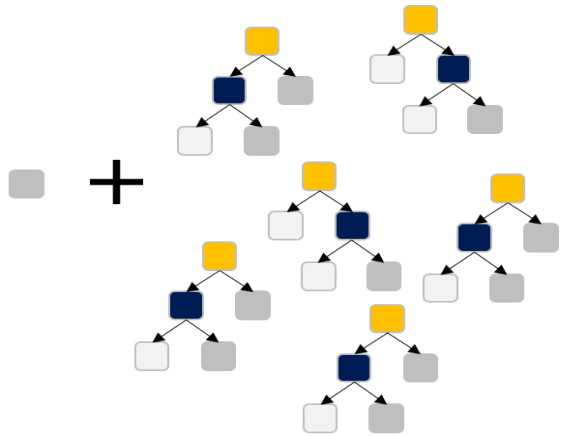
❖ Gradient Boost Machine(GBM)

➢ AdaBoost 와의 차이

✓ Weak learner: Stumps VS A leaf & Restricted trees

- Stump: 한 개 노드와 두 개의 가지를 갖는 매우 작은 decision tree
- A leaf & Restricted trees: 첫 번째 weak learner는 모든 샘플의 output 평균값을 갖는 하나의 leaf과 maximum number of leaves (8-32)로 제한





1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ AdaBoost 와의 차이

✓ Predicted value: Output VS Pseudo-residual

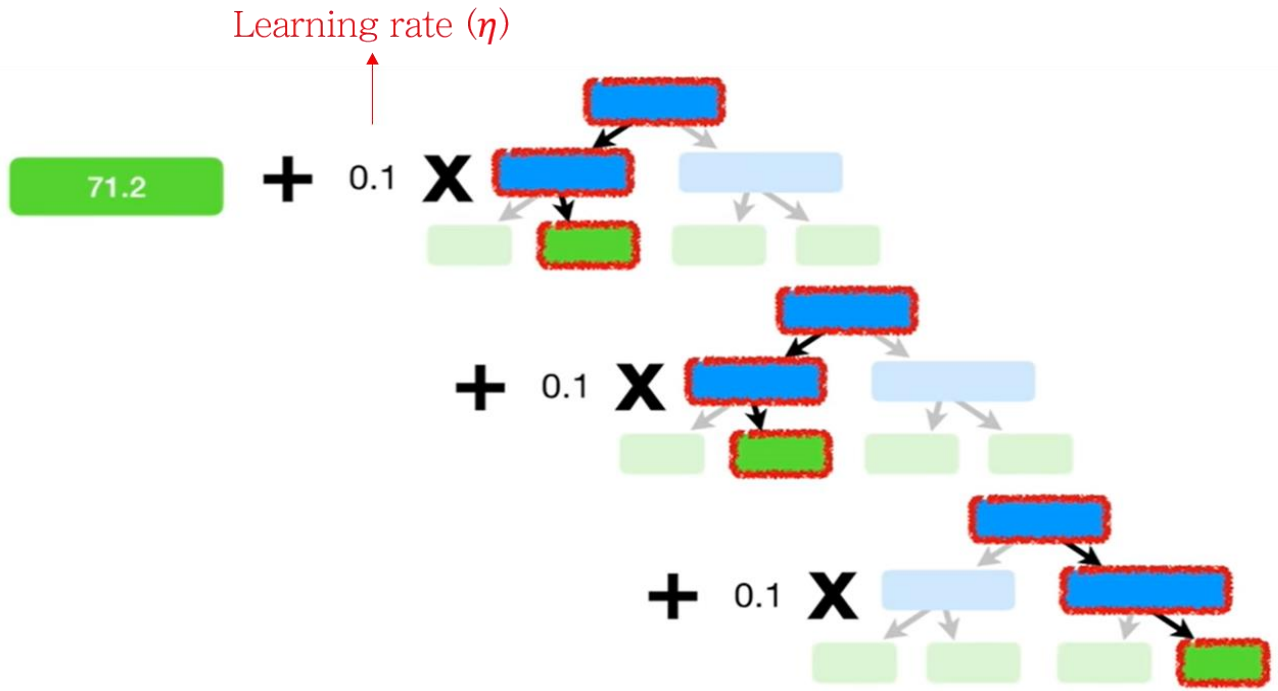
- AdaBoost: 각 stump들은 모두 실제 output 값을 예측하는 모델
- Gradient Boosting: 각 restricted tree들이 예측하는 값은 실제 output 과 이전 모델의 예측치 사이의 오차 (pseudo-residual)
  - 손실함수: $\frac{1}{2}$ * (실제값-예측값)^2
- 최종 예측 시에는 각 모델의 오차를 scaling 후, 합하는 과정을 통해 실제 값에 가까운 예측값을 만들어냄

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ AdaBoost 와의 차이

✓ Predicted value: Output VS Pseudo-residual



1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ AdaBoost 와의 차이

✓ Model weight: Different model weights (amount of say) VS Equal model weight (learning rate)

$$F_t(x) = \sum_{t=1}^{M} \alpha_t h_t(x)$$

$$F_t(x) = F_0(x) + \eta \sum_{t=1}^{M} h_t(x)$$

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ 알고리즘

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: for $m = 1$ to M:

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

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

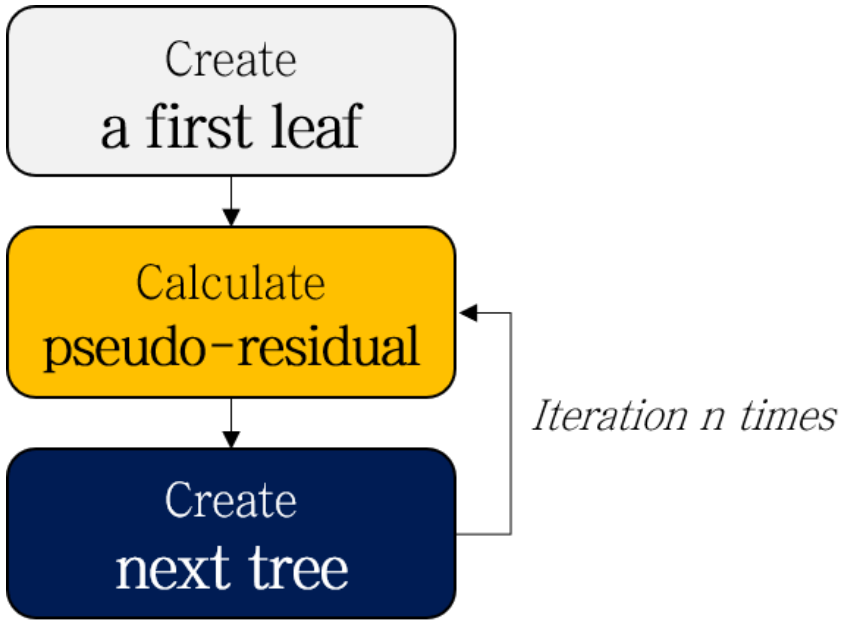
(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ 알고리즘



1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

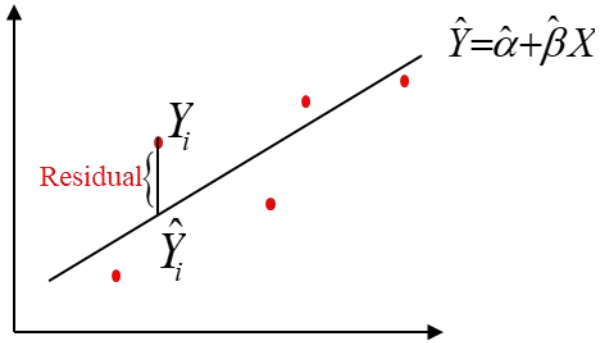
➢ Gradiend Boosting for Regression

✓ **Input:** Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

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

x_i y_i

$$\text{손실함수} = \frac{1}{2}(\text{관측치} - \text{예측치})^2$$



JEONJU UNIVERSITY

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ **Step 1:** Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

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

평균

$$\text{손실함수} = \frac{1}{2}(\text{관측치} - \text{예측치})^2$$

$$L = \sum_{i=1}^{8} \frac{1}{2}(y_i - \gamma)^2$$

$$= \frac{1}{2}(y_1 - \gamma)^2 + \cdots + \frac{1}{2}(y_8 - \gamma)^2$$

$$\frac{\partial L}{\partial \gamma} = -(y_1 - \gamma) - (y_2 - \gamma) \cdots -(y_8 - \gamma) = 0$$

$$\gamma = \frac{\sum_{i=1}^{8} y_i}{8} \qquad F_0(x) = 71.2$$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ **Step 2:** for $m = 1$ to M:

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

손실함수 $\frac{1}{2}(\text{관측치} - \text{예측치})^2$

손실함수 미분 $-(\text{관측치} - \text{예측치})$

$r_{im} = (\text{관측치} - \text{예측치})$ residual

$r_{i1} = (\text{관측치} - F_0(x))$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ **Step 2:** for $m = 1$ to M:

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

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

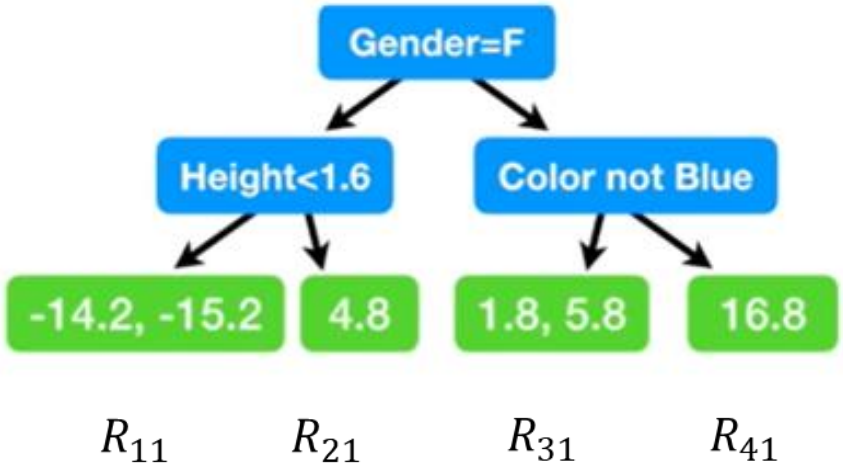
r_{i1} = (관측치 - $F_0(x)$)

$F_0(x) = 71.2$

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

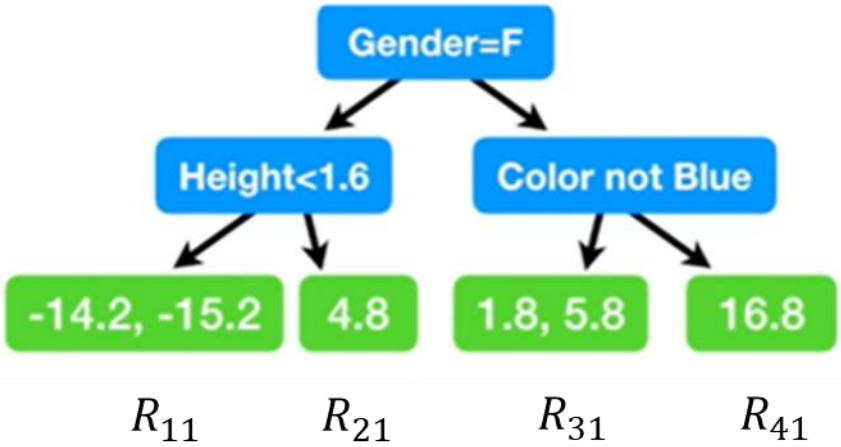
✔ **(B)** Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$



❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ **(C)** For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ii}} L(y_i, F_{m-1}(x_i) + \gamma)$



$$r_{12} = \underset{\gamma}{argmin} \frac{1}{2} (y_2 - (F_0(x) + \gamma))^2$$

$$= \underset{\gamma}{argmin} \frac{1}{2} (76 - 71.2 - \gamma))^2$$

$$= 4.8$$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ **(C)** For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ii}} L(y_i, F_{m-1}(x_i) + \gamma)$

$$r_{11} = \underset{\gamma}{argmin} \frac{1}{2}(y_6 - (F_0(x) + \gamma))^2 + \frac{1}{2}(y_3 - (F_0(x) + \gamma))^2$$

$$= \underset{\gamma}{argmin} \frac{1}{2}(57 - 71.2 - \gamma))^2 + \frac{1}{2}(56 - 71.2 - \gamma))^2$$

$$= \underset{\gamma}{argmin} \frac{1}{2}(-14.2 - \gamma))^2 + \frac{1}{2}(-15.2 - \gamma))^2$$

$$= -14.7$$

$$\frac{\partial L}{\partial \gamma} = 14.2 + \gamma + 15.2 + \gamma = 0$$

$$\gamma = (-14.2 - 15.2)/2$$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ **(C)** For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ii}} L(y_i, F_{m-1}(x_i) + \gamma)$

R_{11} R_{21} R_{31} R_{41}

-14.7 4.8

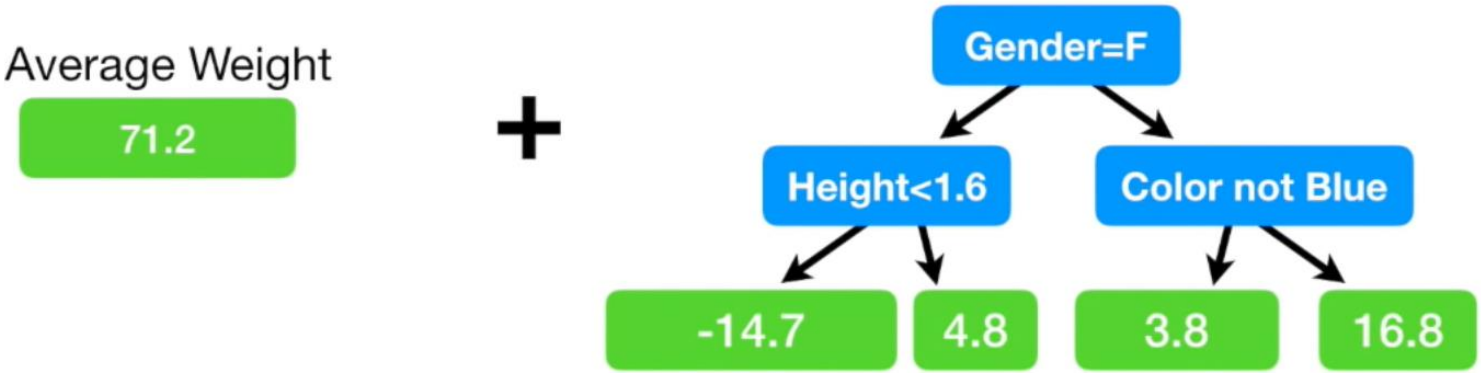
Leaf 값들의 평균값

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✔ **(D)** Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

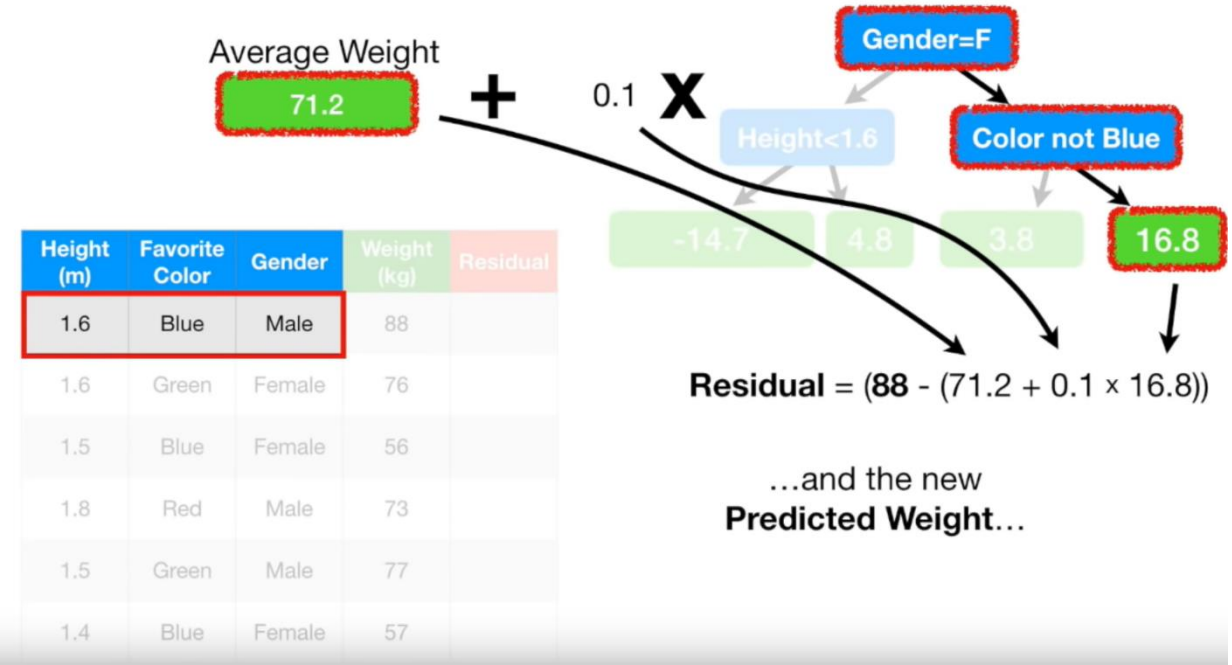


1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✔ **(D)** Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$



Height (m)	Favorite Color	Gender	Weight (kg)	Residual
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	

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✔ **(D)** Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

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

Less Residual

Residual
15.1
4.3
-13.7
1.4
5.4
-12.7

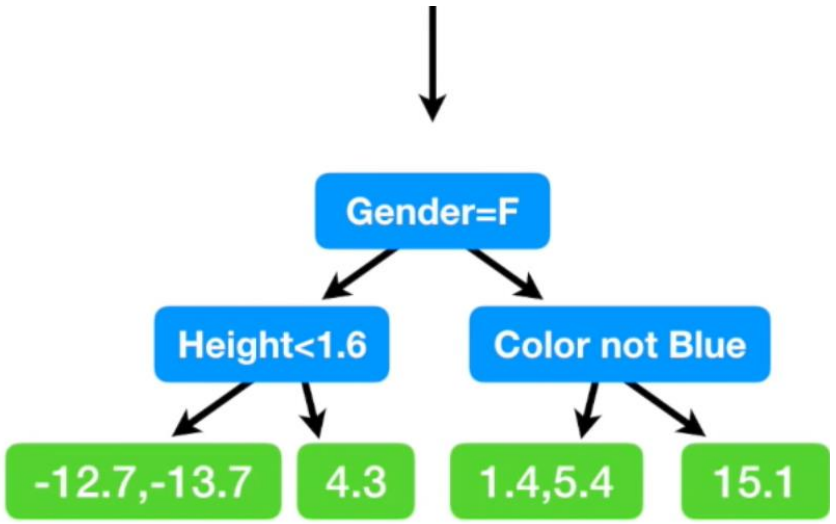
1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ 반복

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

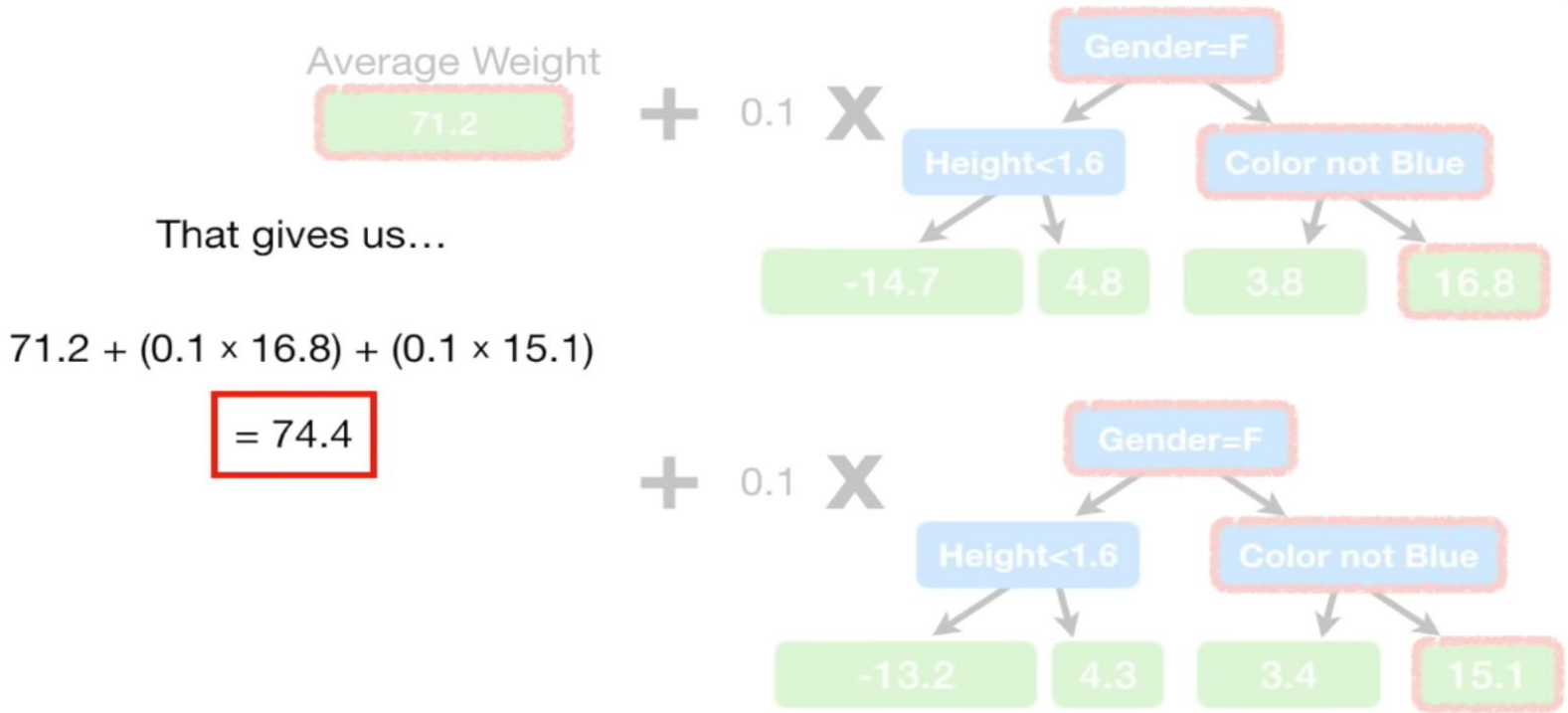


1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ 반복

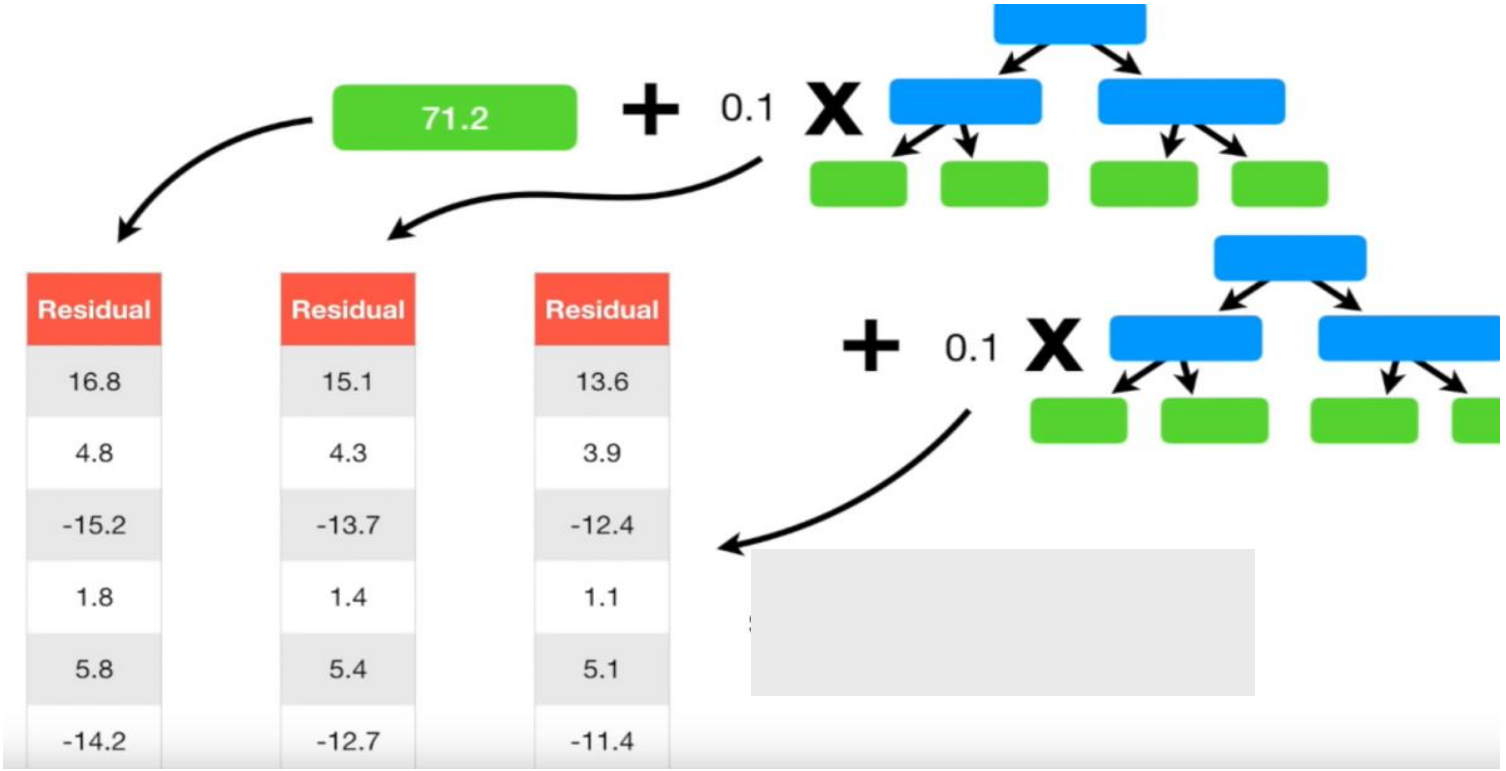


1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ 반복



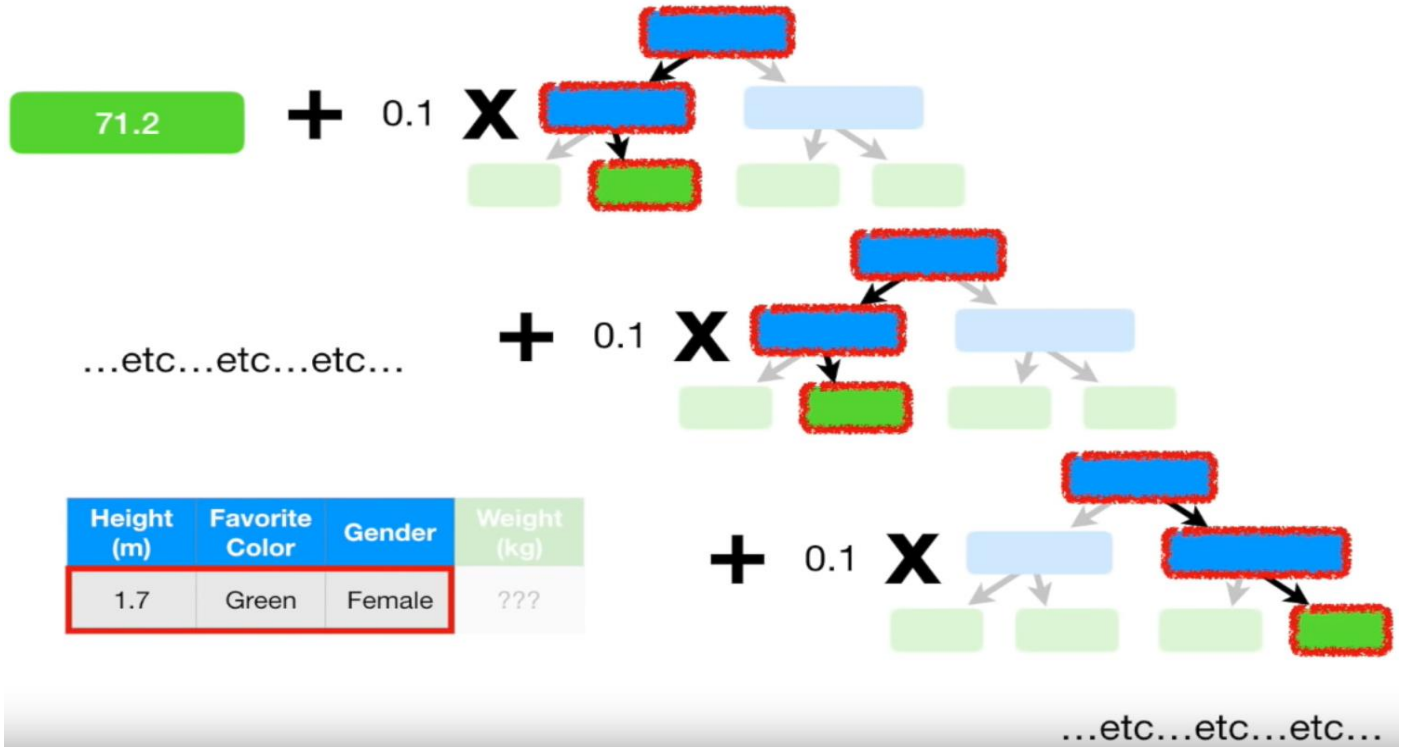
Residual	Residual	Residual
16.8	15.1	13.6
4.8	4.3	3.9
-15.2	-13.7	-12.4
1.8	1.4	1.1
5.8	5.4	5.1
-14.2	-12.7	-11.4

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Regression

✓ 반복횟수(m)까지 혹은 더 이상 residual이 작아지지 않을 때까지 반복



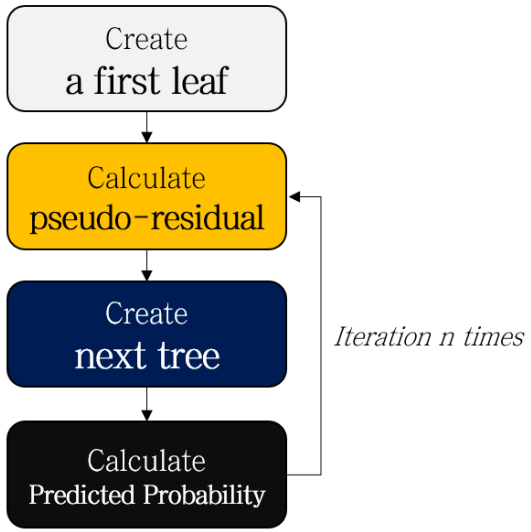
1. 앙상블(보팅,배깅,부스팅)

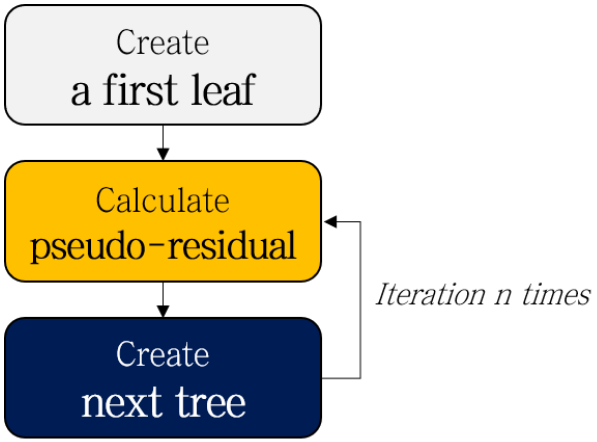
❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ Gradient Boosting for Regression과 전체적인 흐름 (pseudo-residual을 계산하고 이를 예측하는 decision tree를 만들어 나가는 과정)은 비슷하지만, 확률-log(Odds) 변환 같이 세부적인 내용에서 차이가 있음

✓ 알고리즘





JEONJU UNIVERSITY

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 알고리즘

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: for $m = 1$ to M:

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

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 알고리즘

- Create a first leaf
- Calculate pseudo-residuals of probability
- Create a next tree to predict pseudo-residuals
- Calculate predicted probability
- Repeat 2-4

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 알고리즘

- 데이터세트

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

$$\log(odds) = \log(\frac{4}{2}) = log2 = 0.6931$$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

Step 2: for $m = 1$ to M:

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

$$(Observed - p) = Pseudo\ Residual$$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

Step 2: for $m = 1$ to M:

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

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1\ldots J_m$

(C) For $j = 1\ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

$$\gamma = \frac{Residual}{p(1-p)} \qquad \gamma = \frac{Residual_i + Residual_j}{p_i(1-p_i) + p_j(1-p_j)}$$

2차 Taylor Polynomial approximation

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

Step 2: for $m = 1$ to M:

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

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1...J_m$

(C) For $j = 1...J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 알고리즘

- Create a first leaf
- Calculate pseudo-residuals of probability
- Create a next tree to predict pseudo-residuals
- Calculate predicted probability
- Repeat 2-4

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 1. Create a first leaf

- first leaf의 초기 prediction 값은 log(odds)log(odds)를 사용

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

$$\log(odds) = \log(\frac{4}{2}) = log2 = \mathbf{0.6931}$$

Log(odds) 값을 사용하는 가장 쉬운 방법은 확률값으로 변환하는 것

$$P(loves\ troll2 = yes) = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}} = \frac{2}{1+2} = 0.6667$$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 2. Calculate Pseudo-residuals of probability

- Observed Probability
- 실제값 **Observed probability**는 **Output**의 **Yes/No** 값에 따라 **1** 또는 **0**

JEONJU UNIVERSITY

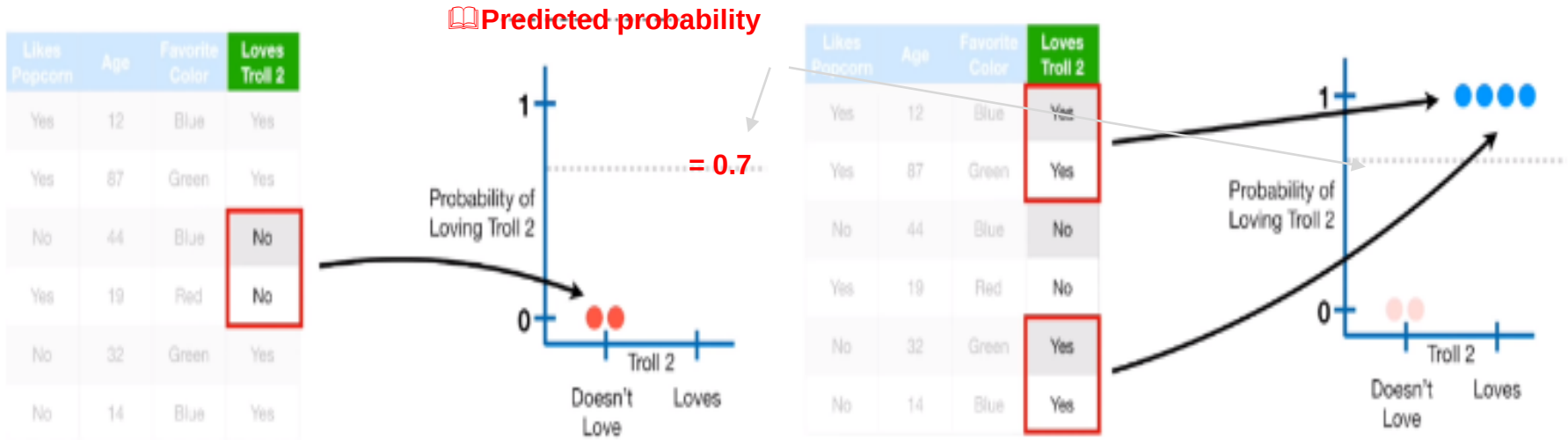
1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 2. Calculate Pseudo-residuals of probability

- Predicted probability
- first leaf의 predicted probability를 사용



JEONJU UNIVERSITY

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

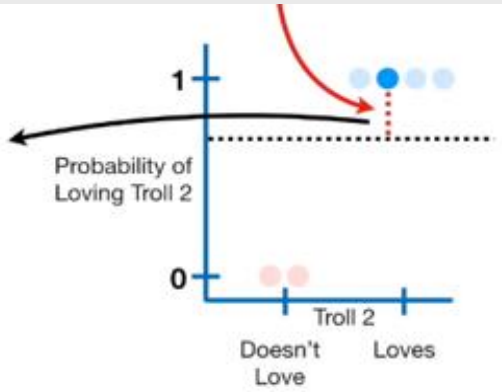
➢ Gradiend Boosting for Classification

✓ 2. Calculate Pseudo-residuals of probability

$$(Observed - p) = Pseudo\ Residual$$

Residual=(1-0.7)

Likes Popcorn	Age	Favorite Color	Loves Troll 2	Residual
Yes	87	Green	Yes	0.3



JEONJU UNIVERSITY

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 2. Calculate Pseudo-residuals of probability

Likes Popcorn	Age	Favorite Color	Loves Troll 2	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3

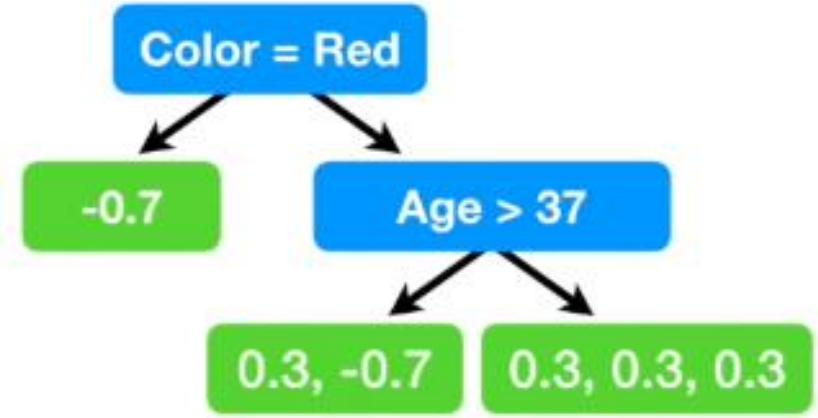
1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 3. Create a next tree to predict pseudo-residual

- Create a tree
- Pseudo-residual을 예측하는 decision tree를 만듬

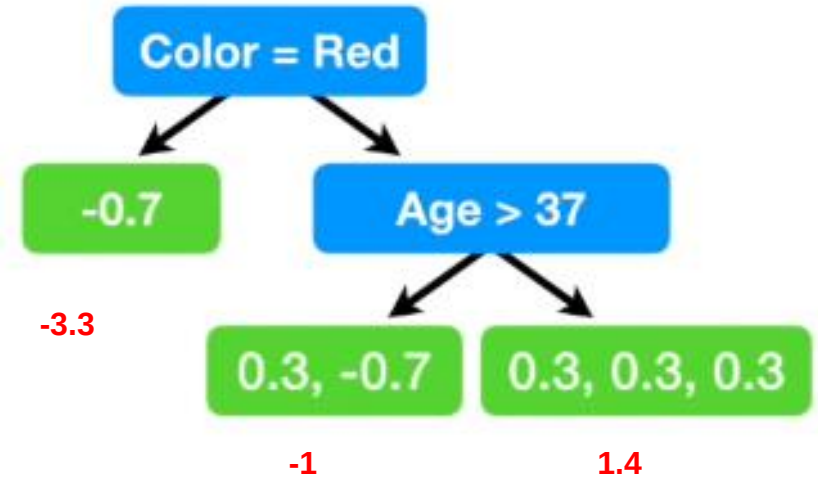


❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 3. Create a next tree to predict pseudo-residual

- Calculate representative value by leaves



$$\gamma = \frac{Residual}{p(1-p)}$$

$$\gamma = \frac{Residual_i + Residual_j}{p_i(1-p_i) + p_j(1-p_j)}$$

$$\gamma = \frac{\sum Residual_i}{\sum p_i(1-p_i)}$$

$$\gamma = \frac{-0.7}{0.7(1-0.7)}$$

P=0.7

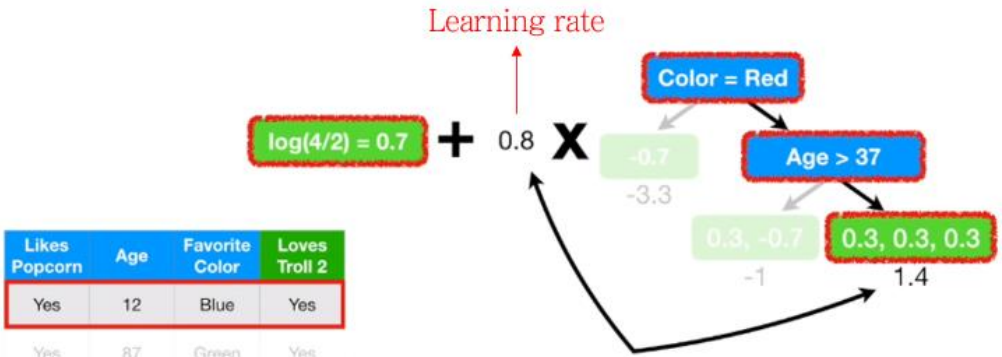
1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 4. Calculate predicted probability

- Pseudo-residual 계산에 사용될 샘플별 예측값을 계산
- 먼저 log(odds) 를 계산: first leaf 예측값+ tree 예측값



Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes

$$\log(odds)\ prediction = \text{0.7+0.8*1.4 = 1.8}$$

$$probability = \frac{e^{\log(odds)}}{1+e^{\log(odds)}} = \frac{e^{1.8}}{1+e^{1.8}} = 0.9$$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 4. Calculate predicted probability

- Pseudo-residual 계산에 사용될 샘플별 예측값을 계산

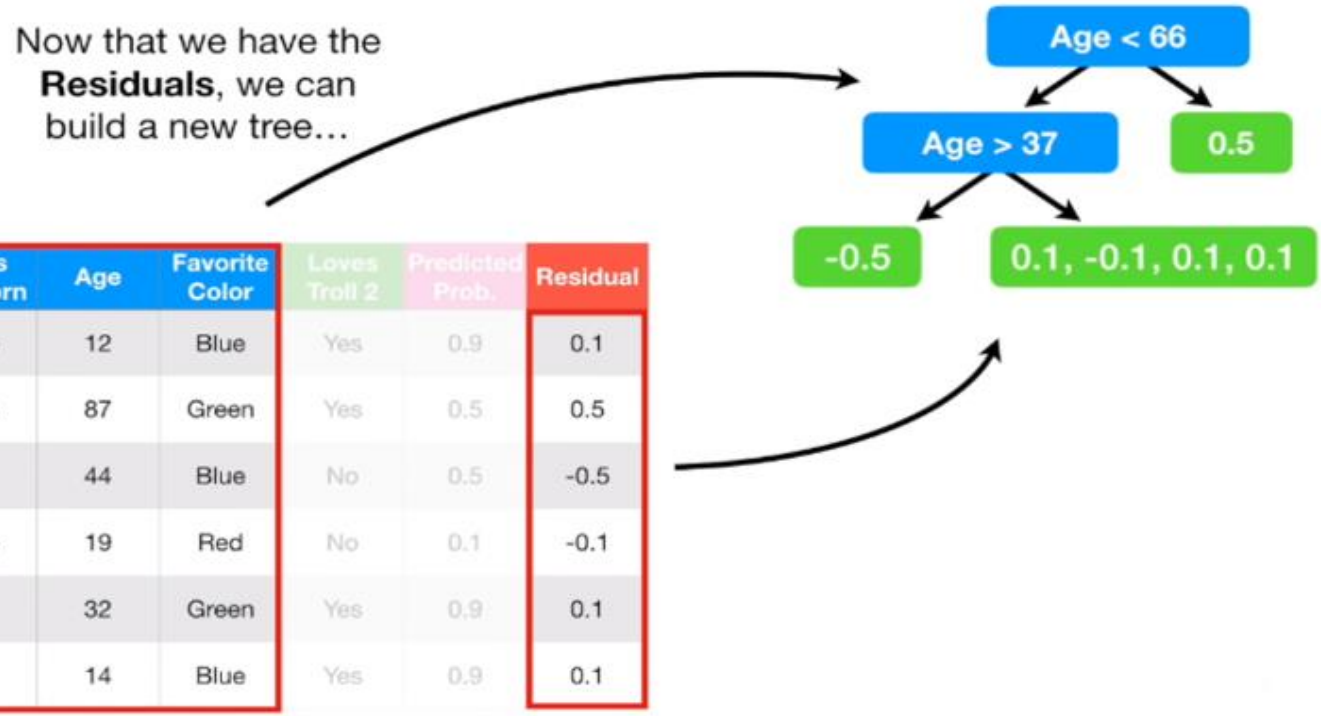
Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.
Yes	12	Blue	Yes	0.9
Yes	87	Green	Yes	0.5
No	44	Blue	No	0.5
Yes	19	Red	No	0.1
No	32	Green	Yes	0.9
No	14	Blue	Yes	0.9

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 5. Repeat 2-4



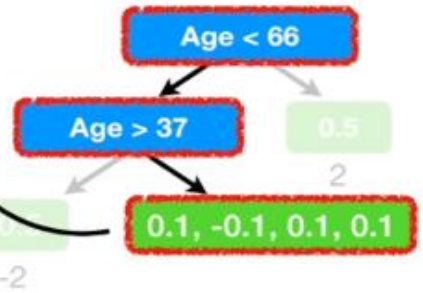
Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.	Residual
Yes	12	Blue	Yes	0.9	0.1
Yes	87	Green	Yes	0.5	0.5
No	44	Blue	No	0.5	-0.5
Yes	19	Red	No	0.1	-0.1
No	32	Green	Yes	0.9	0.1
No	14	Blue	Yes	0.9	0.1

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 5. Repeat 2-4

$$\frac{0.1 + -0.1 + 0.1 + 0.1}{\sum \text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)}$$



Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.	Residual
Yes	12	Blue	Yes	0.9	0.1
Yes	87	Green	Yes	0.5	0.5
No	44	Blue	No	0.5	-0.5
Yes	19	Red	No	0.1	-0.1
No	32	Green	Yes	0.9	0.1
No	14	Blue	Yes	0.9	0.1

So we plug the **Residuals** into the formula for the **Output Values**...

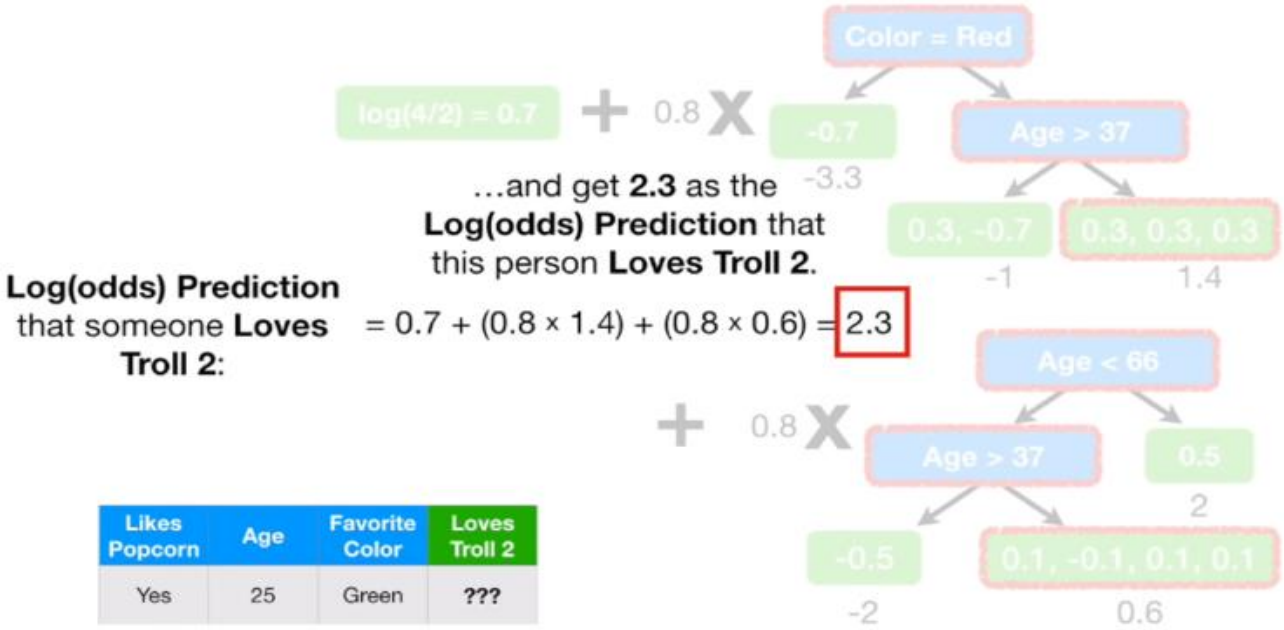
$$\frac{0.1 - 0.1 + 0.1 + 0.1}{(0.9 \times (1 - 0.9)) + (0.1 \times (1 - 0.1)) + (0.9 \times (1 - 0.9)) + (0.9 \times (1 - 0.9))} = 0.6$$

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢ Gradiend Boosting for Classification

✓ 5. Repeat 2-4



$$Probability = \frac{e^{log(odds)}}{1 + e^{log(odds)}} = \frac{e^{2.3}}{1 + e^{2.3}} = 0.9$$

JEONJU UNIVERSITY

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢GBM의 하이퍼 파라미터 및 튜닝

✓Tree에 관한 하이퍼 파라미터

- max_depth(default = 3),
- min_samples_split(Default = 2),
- min_samples_leaf(default = 1),
- max_features(Default = 'none') ,
- max_leaf_nodes(default = None)

1. 앙상블(보팅,배깅,부스팅)

❖ Gradient Boost Machine(GBM)

➢GBM의 하이퍼 파라미터 및 튜닝

✓ Boosting에 관한 하이퍼파라미터

- loss: 경사하강법에서 사용할 cost function 지정. 특별한 이유가 없으면 default 값인 deviance 적용
- n_estimators: 생성할 트리의 갯수를 지정. Default = 100. 많을수록 성능은 좋아지지만 시간이 오래 걸림
- learning_rate: 학습을 진행할 때마다 적용하는 학습률(0-1). Weak learner가 순차적으로 오류 값을 보정해나갈 때 적용하는 계수. default = 0.1. 낮은 만큼 최소 오류 값을 찾아 예측성능이 높아질 수 있음. 하지만 많은 수의 트리가 필요하고 시간이 많이 소요
- subsample: 개별 트리가 학습에 사용하는 데이터 샘플링 비율(0~1). default=1 (전체 데이터 학습). 이 값을 조절하여 트리 간의 상관도를 줄일 수 있음

Thank You !

JEONJU UNIVERSITY
The Place for Superstars