SVM

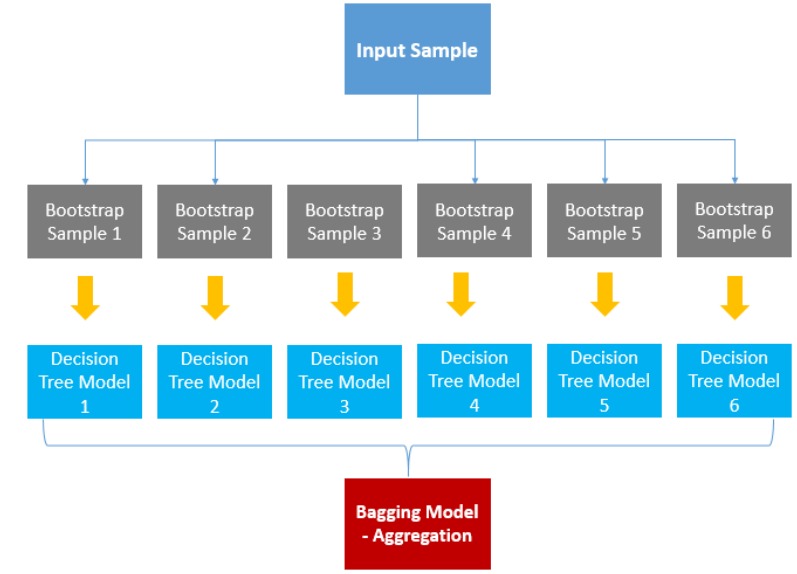
Boosting , Stacking

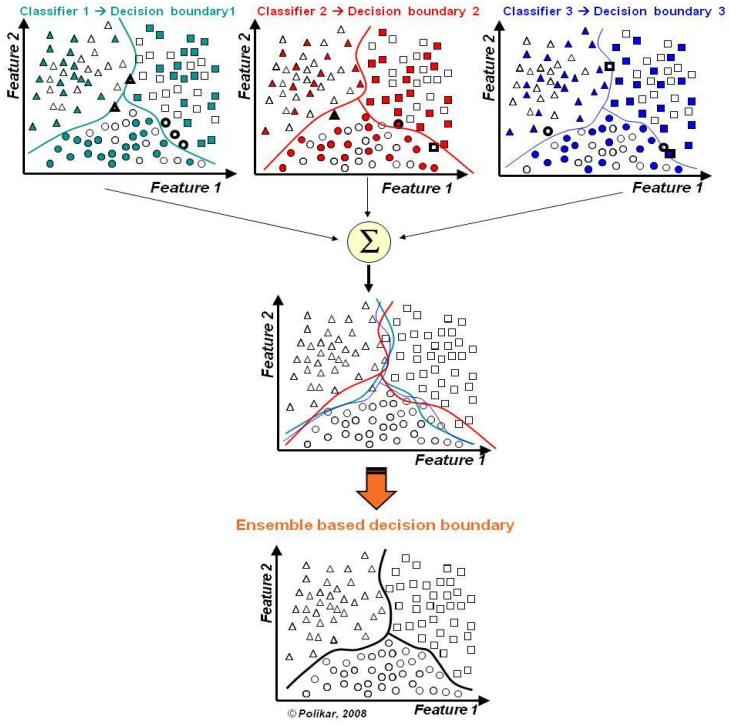
Boosting

- Ensemble
  - 일반적으로 한 개의 모델을 활용하여 결과를 예측
  - Weak Learner를 여러 개 결합한다면 Single Learner보다 더 나은 성능을 얻을 수 있다는 아이디어에서 출발
  - Bagging, Boosting, 그리고 Stacking 모두 이 아이디어에서 출발

Boosting

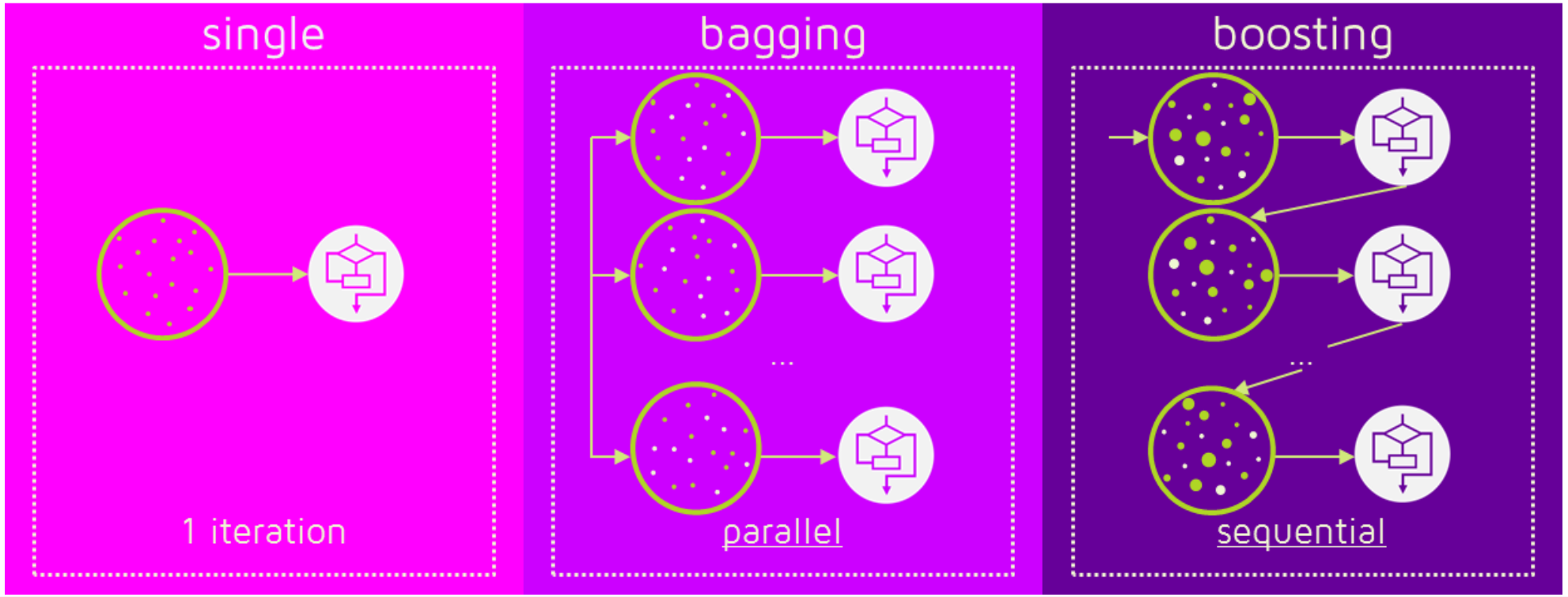
- Bagging이란?
  - 샘플을 여러 번 뽑아 각 모델을 학습시켜 결과물을 집계하는 방법
  - 복원 추출(Bootstrap) + 집계(Aggregation) -> Bagging
  - Categorical Data : Voting
  - Continuous Data : Average





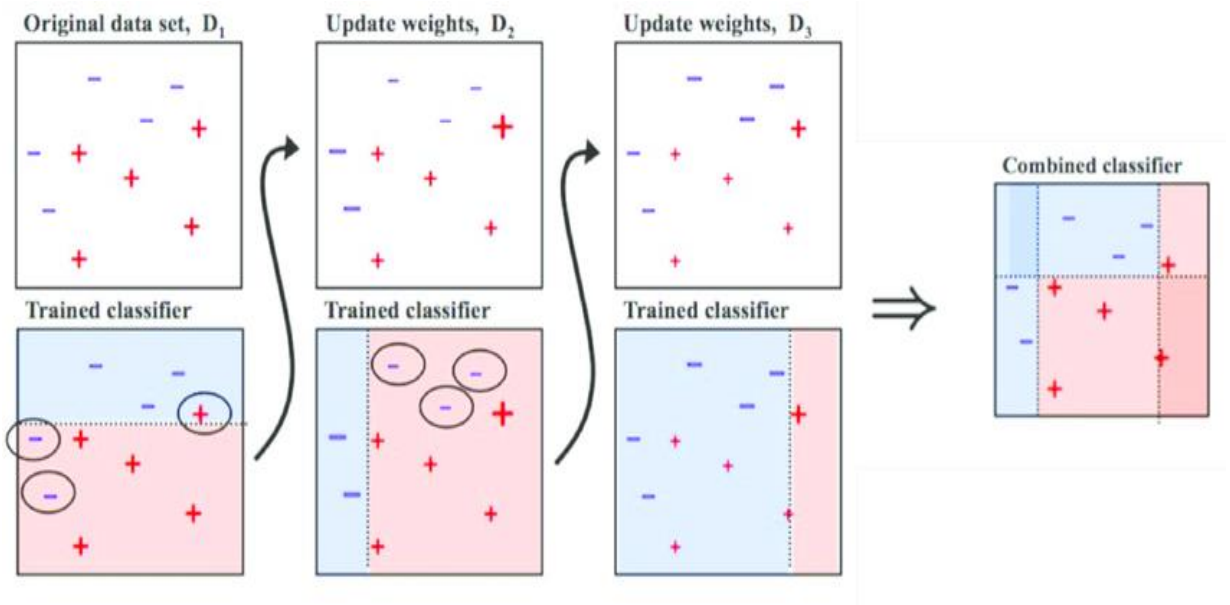
Boosting

- Boosting이란?
  - Bagging과는 다르게 순차적 방법을 통해 이전 학습의 결과가 다음 학습의 결과에 영향



Boosting

- Sample을 Bagging과 마찬가지로 복원 추출
- 첫번째 Sample의 예측 결과에 따라 오답에 대해 가중치를 적용하여 다음 Sample을 순차적으로 학습



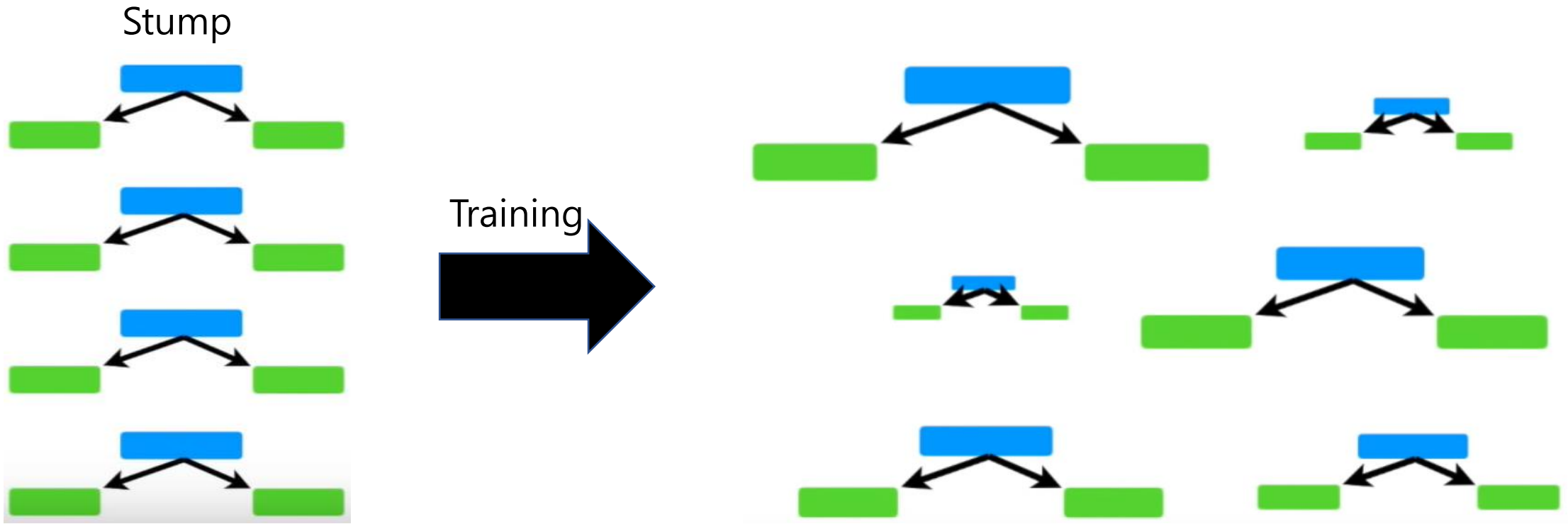
Boosting

- 장점
  - Error에 대해 가중치를 주어 학습하기 때문에 작은 Error
- 단점
  - 순차적으로 진행되기 때문에 느린 속도
  - 높은 확률의 Overfitting

- 종류
  - Boosting 계열 방법론에는 대표적으로 AdaBoost, Gradient Boost 등이 존재

AdaBoost(=Adaptive Boosting)

- AdaBoost는 Stump를 이용하여 학습을 진행



➢ 각각의 Stump에 가중치를 두는 방향으로 학습을 진행

AdaBoost(=Adaptive Boosting)

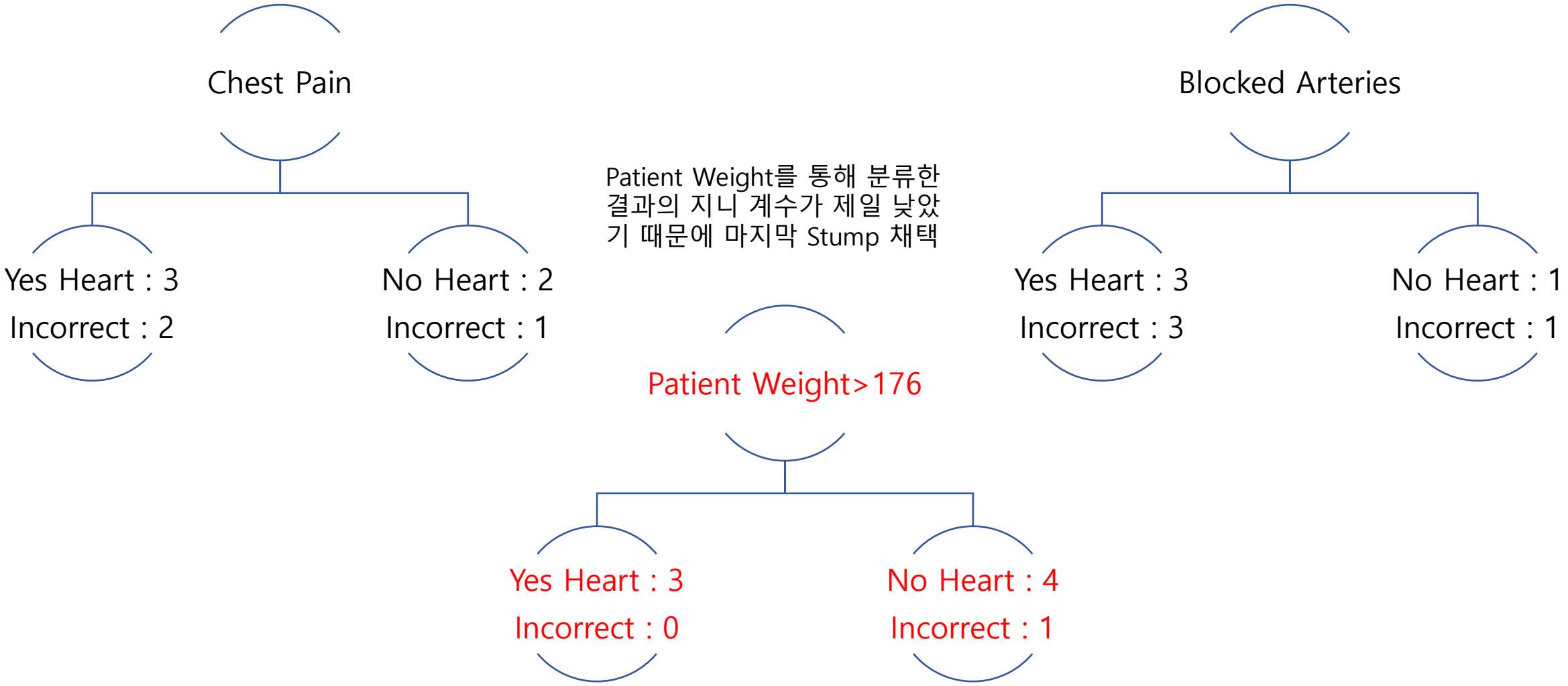
첫 샘플링할 때 각각의 데이터가
뽑힐 확률은 $\frac{1}{8}$로 모두 같다.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

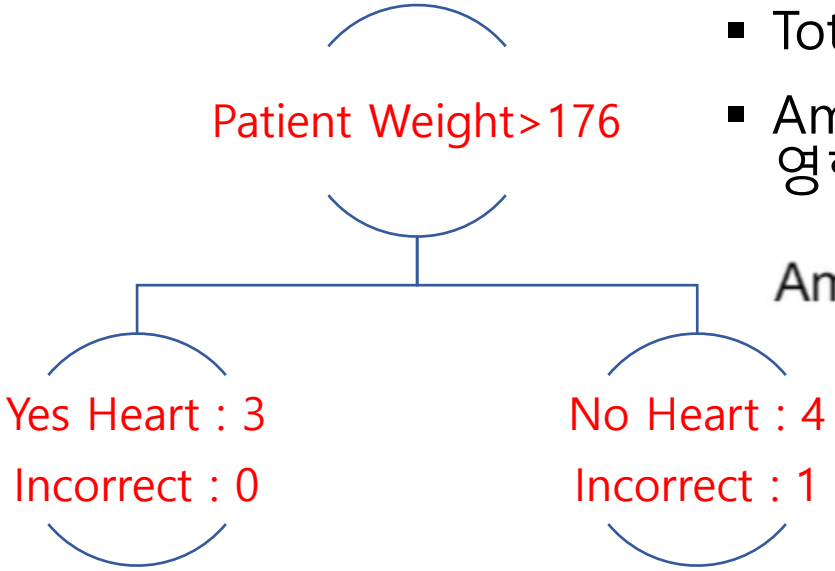
Chest Pain, Blocked Arteries, Patient Weight를 통해 Heart Disease의 발병 유무를 분류하는 Task

Stump를 활용하기 때문에 Heart Disease를 분류하기 위해 하나의 Column을 활용

AdaBoost(=Adaptive Boosting)



AdaBoost(=Adaptive Boosting)



- Total Error = 1/8
- Amount of Say란 최종 분류에 있어 해당 Stump가 얼마만큼의 영향을 주는지에 대한 지표

$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

AdaBoost(=Adaptive Boosting)

- Total Error = 1/8
- Amount of Say란 최종 분류에 있어 해당 Stump가 얼마만큼의 영향을 주는지에 대한 지표

$$New\ Sample\ Weight = Sample\ Weight * e^{Amount\ of\ Say}$$

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	**Yes**	**167**	**Yes**	**1/8**
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

오분류된 데이터 -> 더 높은 가중치

$$New\ Sample\ Weight = 0.125 * e^{0.97} = 0.33$$

AdaBoost(=Adaptive Boosting)

- Total Error = 1/8
- Amount of Say란 최종 분류에 있어 해당 Stump가 얼마만큼의 영향을 주는지에 대한 지표

$$New\ Sample\ Weight = Sample\ Weight * e^{Amount\ of\ Say}$$

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

제대로 분류된 데이터 -> 더 낮은 가중치

$$New\ Sample\ Weight = 0.125 * e^{-0.97} = 0.05$$

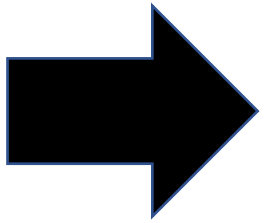
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

Gradient Boosting

- Gradient Boosting은 말 그대로 **Gradient**를 이용해서 Boosting하는 방법론
- Target을 예측하는 것이 아닌 **Residual**을 줄여 나가는 방식으로 학습

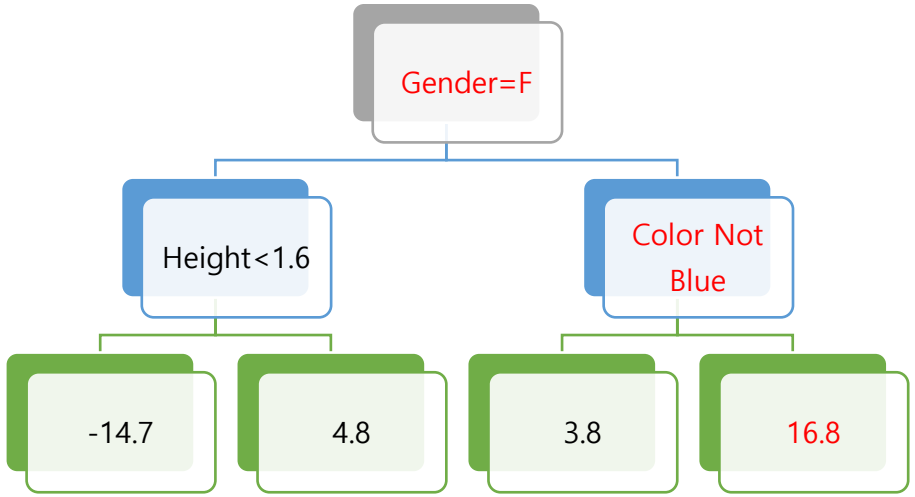
Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

Mean = 71.2



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

Gradient Boosting

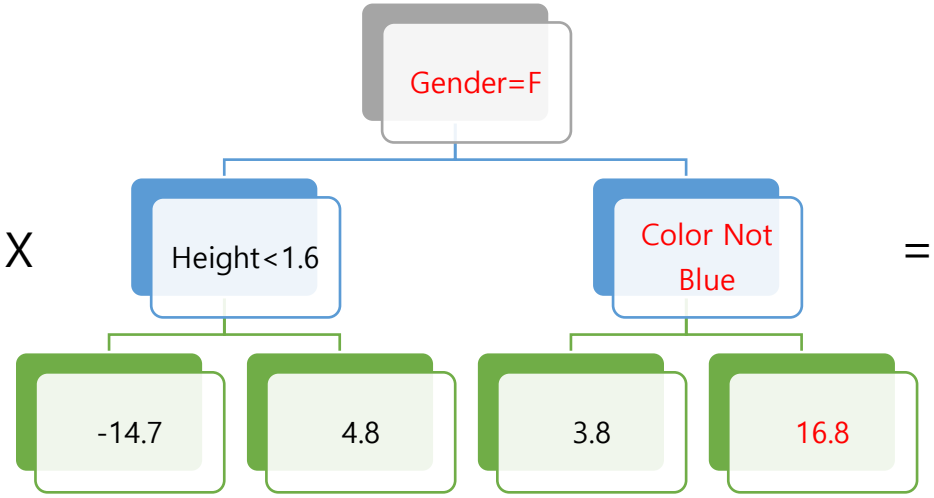


- Tree를 진행시키며 Leaf Node에 Residual를 기록
- 이 때, Gender=M, Color=Blue를 만족하는 데이터의 Residual를 구해보면 16.8이 나오는데 이 때 Residual와 처음 예측값을 더하면 원래 예측해야 되는 값인 88이 나오게 되는 것을 확인
- Train data에 과적합

Gradient Boosting

- Learning Rate를 이용해 Residual를 줄여 나가는 방향으로 학습을 진행

71.2 + Learning Rate X 0.1



=

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

- 업데이트된 Residual가 처음 존재하던 Residual와 비교했을 때 더 작아졌음을 확인

Gradient Boosting

- 순차적인 학습으로 인해 Bagging에 비해 많은 시간 소요
- Residual를 줄여 나가는 방식으로 학습이 진행되기 때문에 Overfitting이 일어나기 쉽고, 이런 Overfitting을 막기 위해 정밀한 Hyperparameter 조정이 필요
- Gradient Boosting을 활용한 모델로는 XGBoost, LGBM와 같은 방법론이 존재

XGBoost(=eXtreme Gradient Boost)

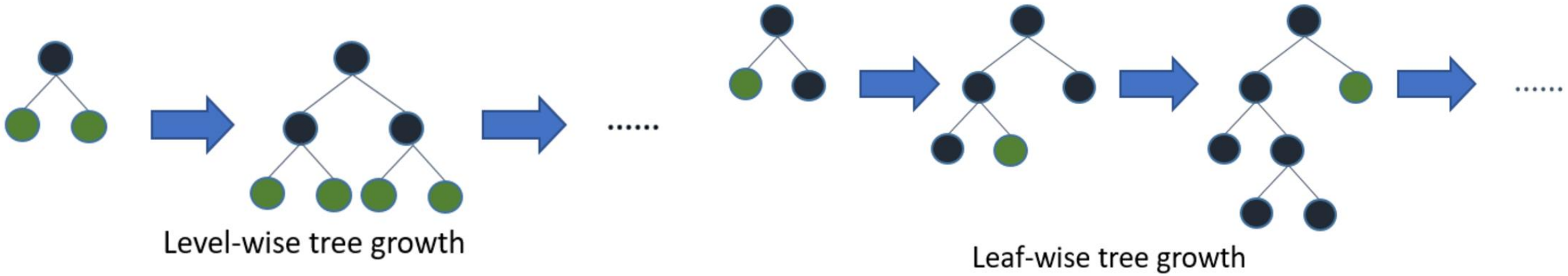
- 장점
  - 기존의 Gradient Boosting Model보다 빠른 속도
    - Tree Pruning
    - 내부적으로 병렬 처리 가능-> 속도 증가
  - Overfitting을 막기 위한 Hyperparameter들이 존재
    - GBM에는 Regularize 할 수 있는 장치가 존재 X
  - Early Stopping
  - CART(Classification And Regression Tree) 기반
    - 분류, 회귀 모두 사용 가능
  - 결측치에 민감 X

LGBM(=Light GBM)

- XGBoost 또한 여전히 시간이 오래 걸리는 문제가 발생
  - 또한 Grandient Boosting 계열 특성 상 Overfitting에 취약해 여전히 정밀한 Hyperparameter 조정이 필요
  - GridSearch와 같이 Hyperparameter를 여러 개 넣어 매번 XGBoost를 돌리게 되면 학습시간이 늘어나는 문제가 발생

LGBM(=Light GBM)

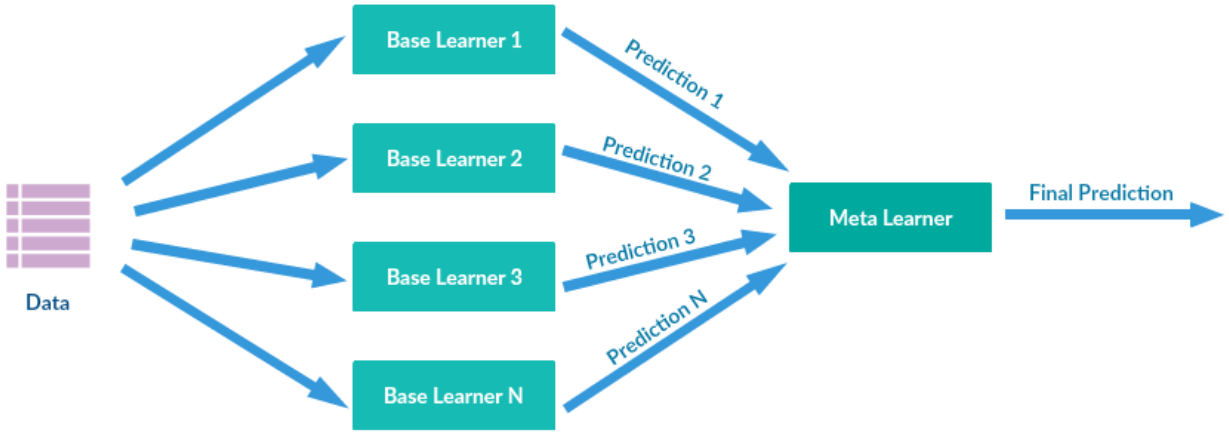
- 장점
  - XGBoost에 비해 훨씬 빠른 속도
    - 기존의 Gradient Boosting Model은 Level-wise 방식 활용
    - 트리의 균형을 맞추지 않고 최대 손실값(max delta loss)를 가지는 리프 노드를 지속적으로 분할
    - 트리의 깊이가 깊어지고 비대칭적 규칙 트리 생성
    - 학습을 반복할 수록 Level-wise보다 예측 오류 손실을 더 줄일 수 있다.
    - 빠르게 학습하기 때문에 대용량 데이터에 적합



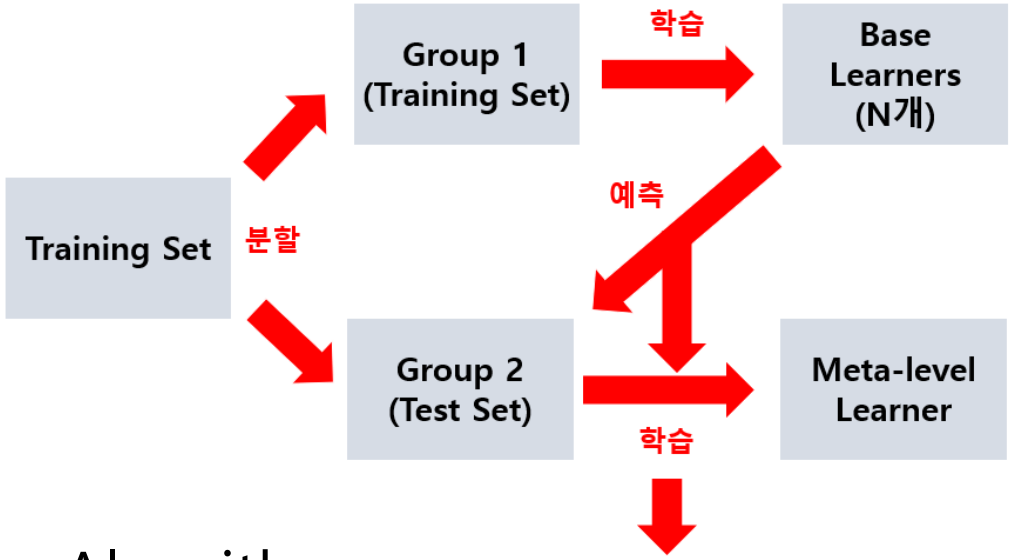
- 단점
  - 적은 데이터셋에 대해서 과적합이 발생하기 쉽다.

Stacking

- Tree 기반의 Bagging이나 Boosting 계열 모델과 달리 서로 다른 모델을 적용 가능
- 각각의 Learner를 통해 나온 결과를 다음 Meta Learner의 Input으로 적용



Stacking

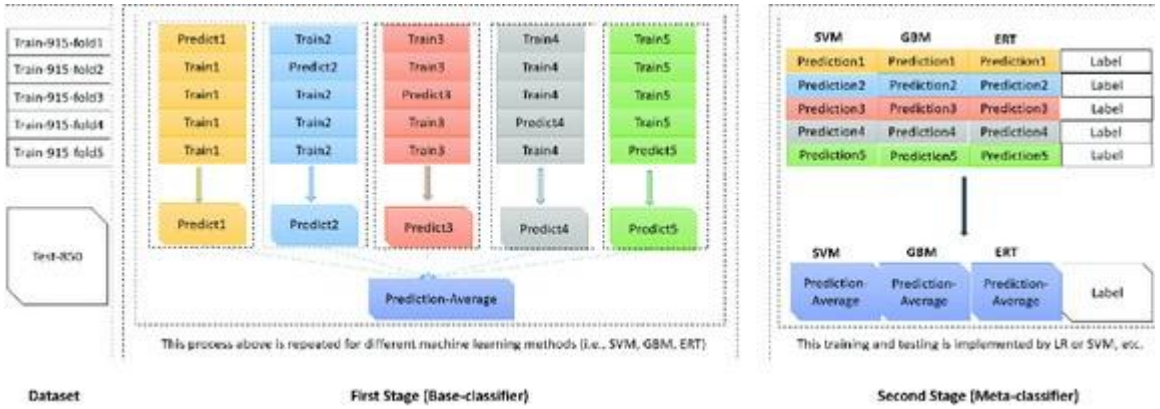


과적합의 문제가 발생해 K-Fold CV를 활용

- Algorithm
  - Training Data를 서로 겹치지 않게 2개의 집단으로 분리
  - 첫번째 그룹을 N개의 Base Learner를 통해 학습
    - 이 때 Bagging이나 Boosting에서는 Tree 기반 모델을 활용했지만 Stacking에서의 Base Learner들은 서로 다른 모델들을 이용할 수 있습니다.(Ex. SVM, 선형 회귀, 신경망 등)
  - 첫번째 그룹을 통해 나온 Output을 두번째 그룹의 Input으로 넣어 두번째 그룹의 실제 Output을 예측하도록 Meta Learner를 학습

Stacking

K-Fold Cross Validation



- K-Fold CV를 통해 K개의 결과를 평균내 사용
- Train, Test 2개로만 분리하는 Hold-out 방식에 비해 과적합 방지하는데 효과적

Boosting, Stacking 참고 자료

Boosting	https://www.youtube.com/watch?v=LsK-xG1cLYA (Adaboost) https://www.youtube.com/watch?v=3CC4N4z3GJc (Gradient Boosting) https://www.youtube.com/watch?v=OtD8wVaFm6E (XGBoost)
Stacking	https://lsjsj92.tistory.com/559 (Stacking Ensemble CV)