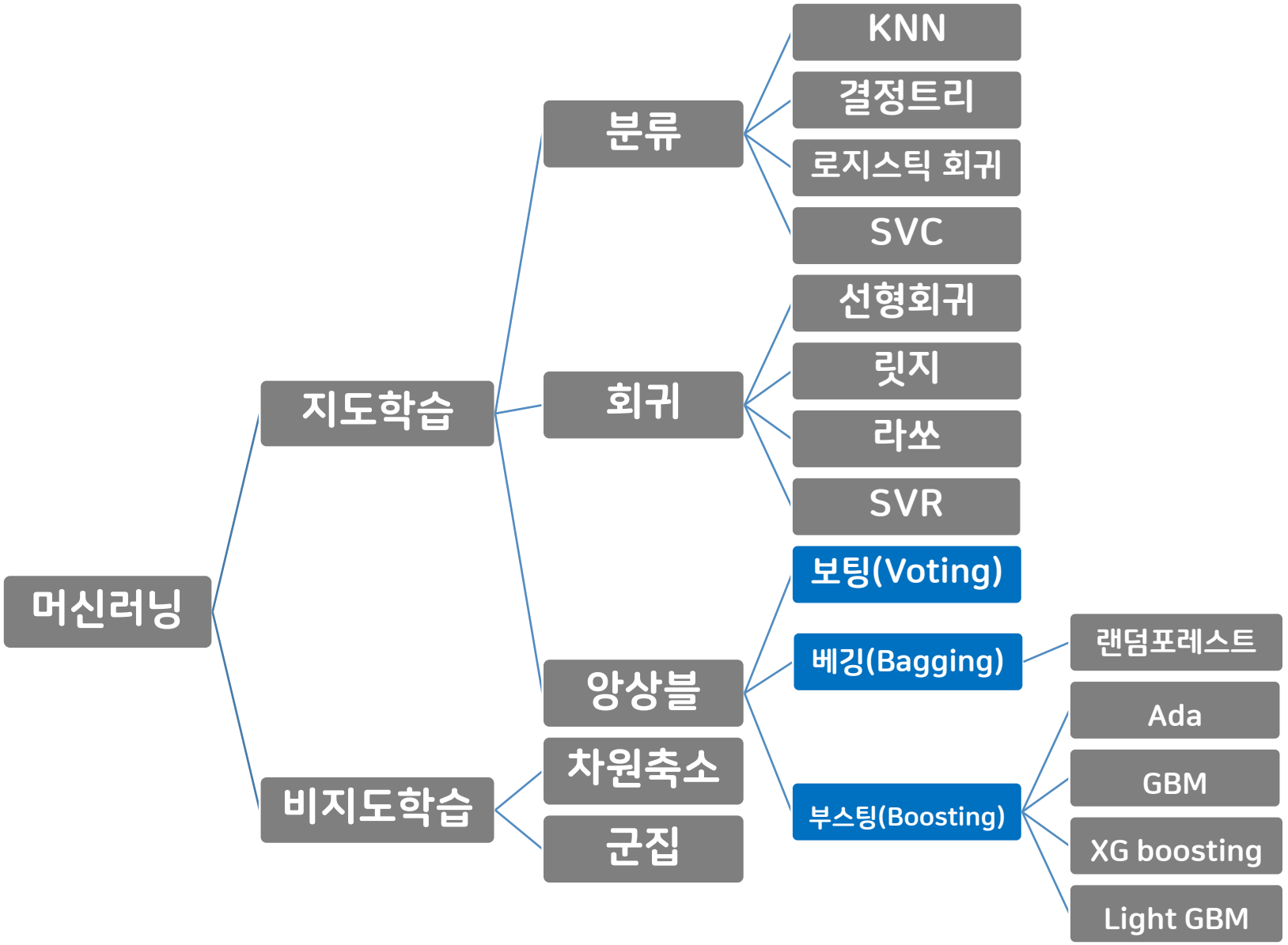
학습목표

- 앙상블(Ensemble)의 개념을 이해할 수 있다.
- 앙상블(Ensemble)의 종류를 이해하고 적용할 수 있다.

머신러닝 모델 개략도


머신러닝
지도학습
분류
KNN
결정트리
로지스틱 회귀
SVC
회귀
선형회귀
릿지
라쏘
SVR
앙상블
보팅(Voting)
베깅(Bagging)
랜덤포레스트
부스팅(Boosting)
Ada
GBM
XG boosting
Light GBM
비지도학습
차원축소
군집

앙상블 수업 흐름도

스마트인재개발원
Smart Human Resources Development

개요

·개념

앙상블 종류

·보팅(Voting)
·베깅(**Bagging**)
·부스팅(Boosting)

Bagging모델

·랜덤 포레스트
(Random forest)

Boosting모델

·Ada boosting
·Gradient
boosting(GBM)
·XG boosting
·LightGBM

코드실습

집단지성

앙상블(Ensemble)

스마트인재개발원
Smart Human Resources Development

여러 개  합하여

정확 기법

앙상블을 사용하는 이유

1. 단일 모델에 비해 높은 성능과 신뢰성을 얻을 수 있음

2. 데이터의 양이 적어도 충분한 학습 효과를 거둘 수 있음

앙상블(Ensemble) 종류

1. 보팅(Voting)

여러 개의 다른 종류의 모델이 예측한 결과를 투표 혹은 평균을 통해 최종 선정

2. 베깅(Bagging)

여러 개의 같은 종류의 모델이 예측한 결과를 투표 혹은 평균을 통해 최종 선정

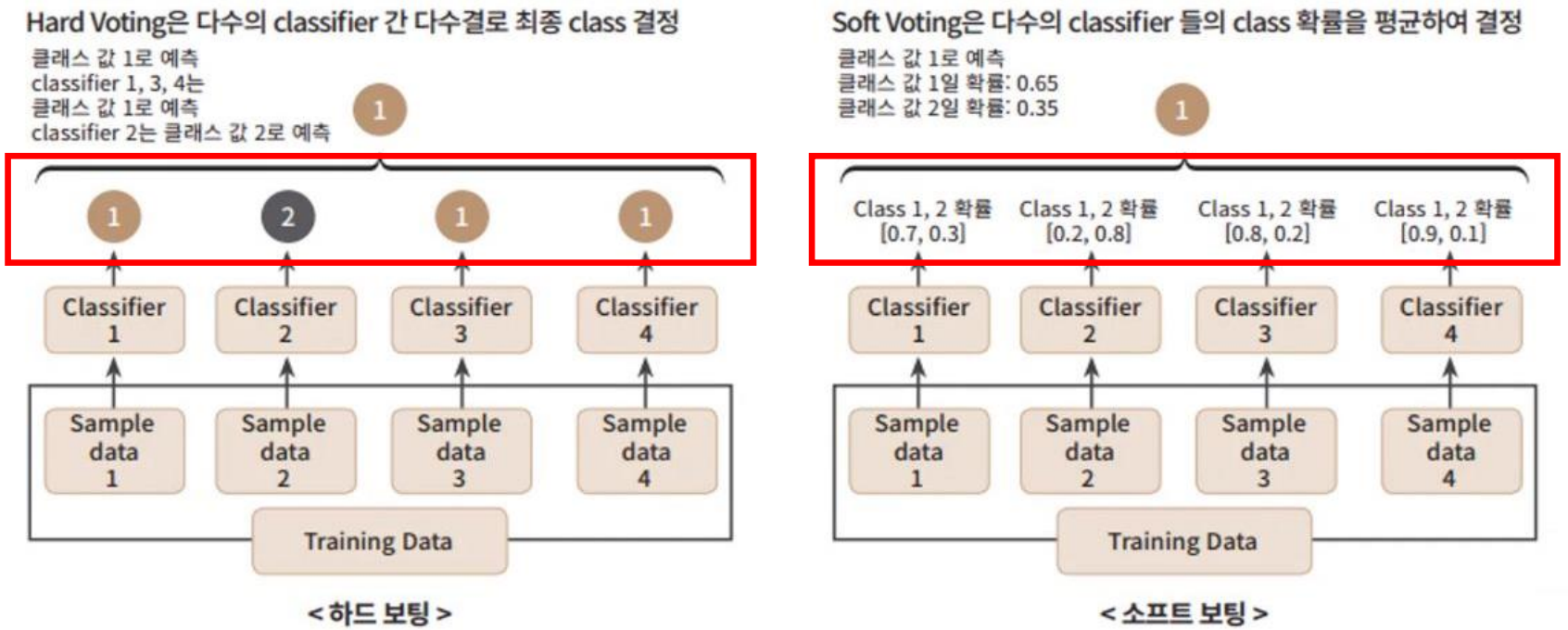
3. 부스팅(Boosting)

여러 개의 같은 종류의 모델이 순차적으로 학습-예측하고 가중치를 달리하여
오류를 개선하는 방식

보팅(Voting)

여러 개의 다른 모델이 예측한 결과를 투표 혹은 평균을 통해 최종 예측결과를 선정

1. 하드 보팅(Hard voting) : 다수결
2. 소프트 보팅(Soft voting) : 각 확률의 평균

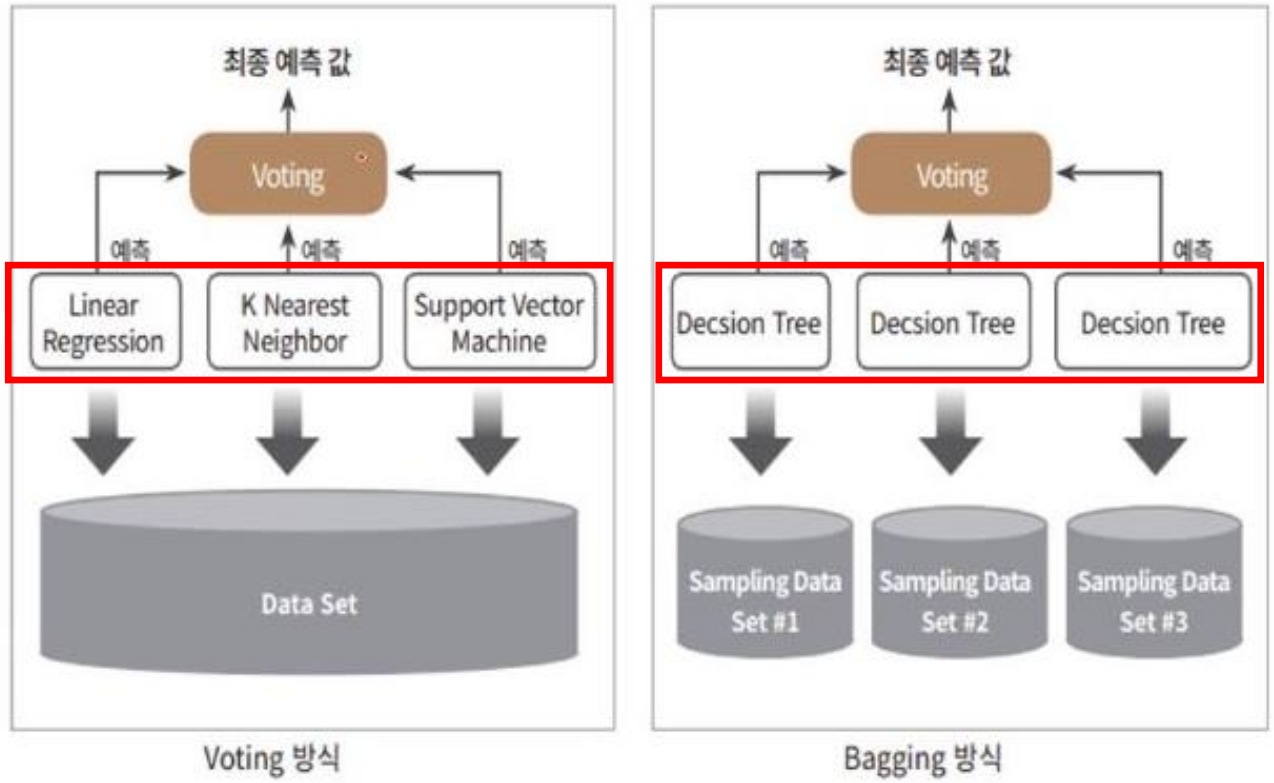


<하드 보팅>　　<소프트 보팅>

- 일반적으로 하드 보팅보다는 소프트 보팅이 예측 성능이 상대적으로 우수하여 주로 사용됨.
- 사이킷런은 VotingClassifier 클래스를 통해 보팅(Voting)을 지원

출처 : 인프런 머신러닝 강의

보팅(Voting), 베깅(Bagging)



여러 개의 모델이 투표 혹은 평균을 통해 최종 예측결과를 선정

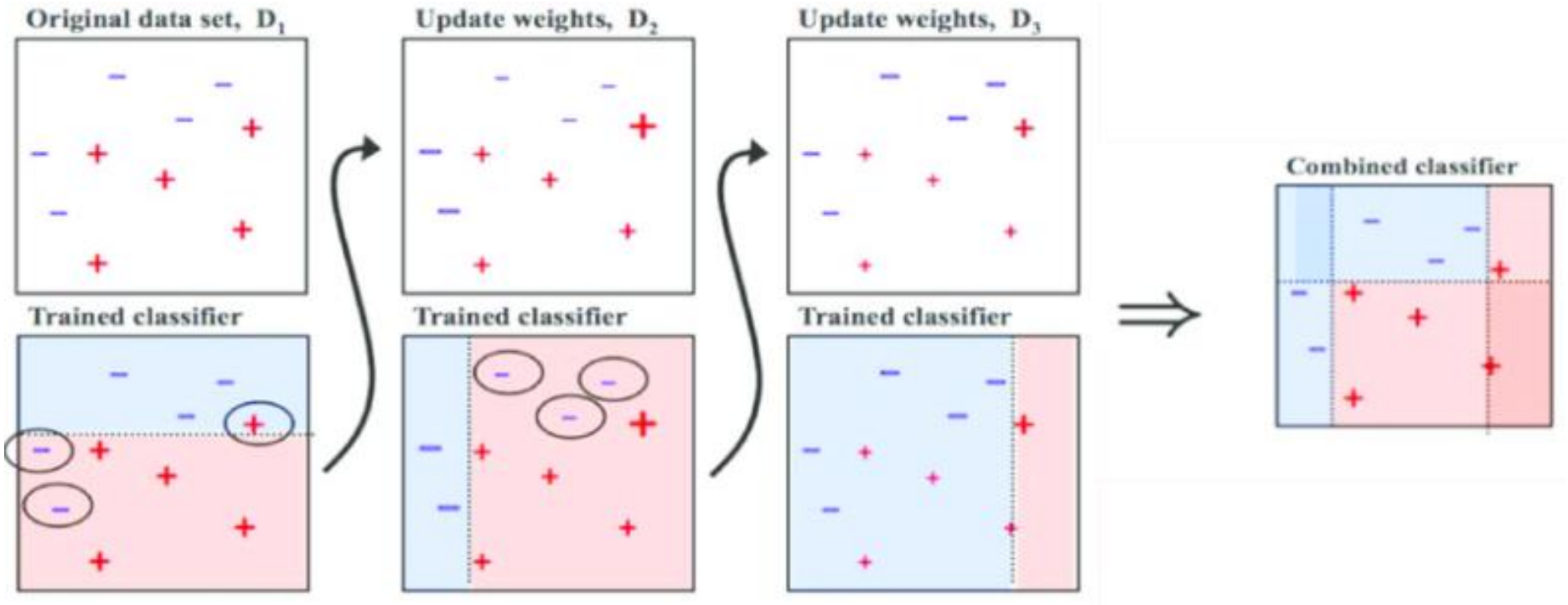
1.Voting : 서로 다른 모델을 결합

2.Bagging : 같은 종류의 모델을 결합(데이터 샘플링을 다르게, 중첩 허용)

출처 : 인프런 머신러닝 강의

부스팅(Boosting)

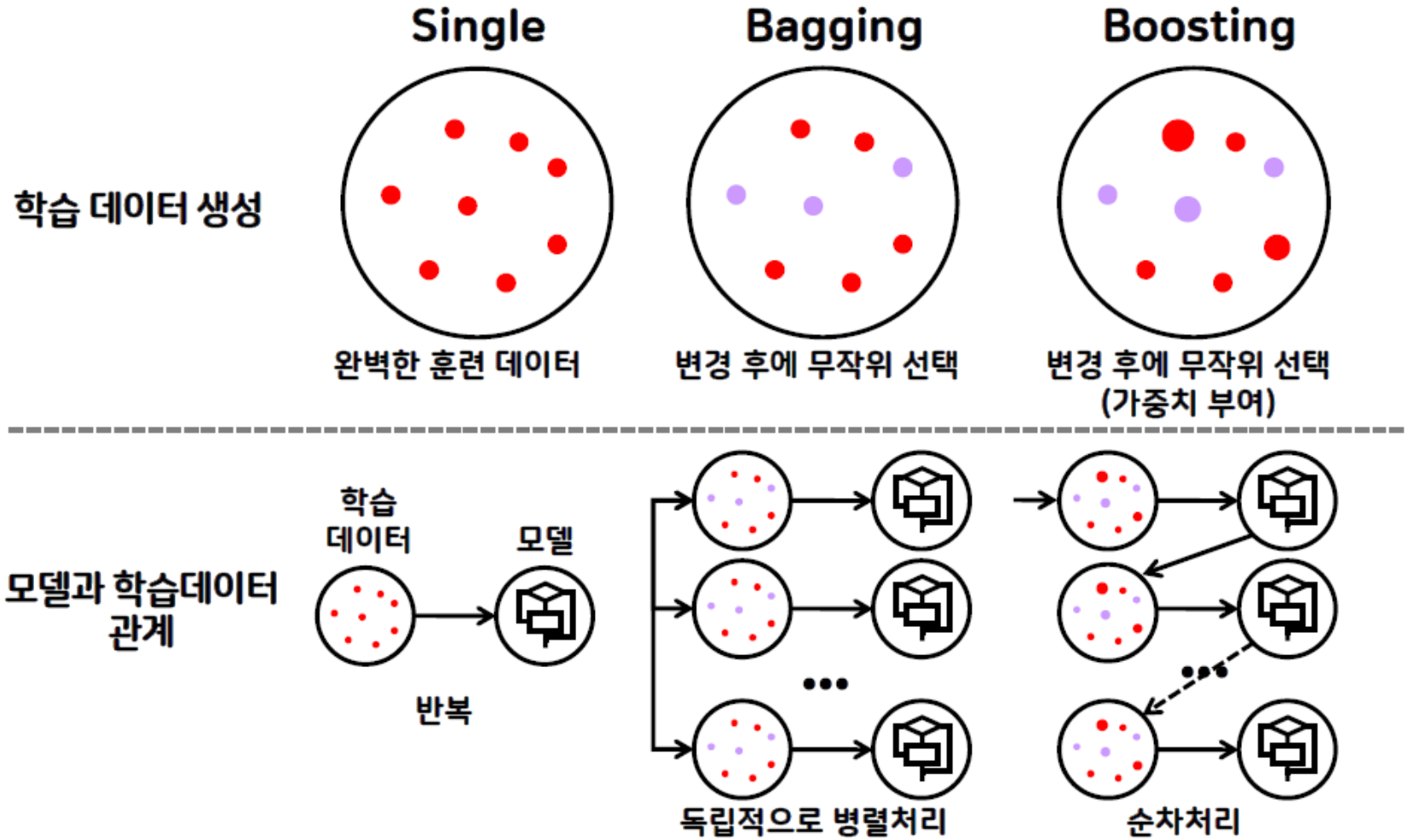
여러 개의 모델이 순차적으로 학습-예측하며 잘못 예측한 데이터에 가중치를 부여해 오류를 개선해 나가면서 학습하는 방식(결정 트리 모형을 베이스로 사용)



출처: Medium (Boosting and Bagging explained with examples)

베깅(Bagging), 부스팅(Boosting)

Bagging vs Boosting

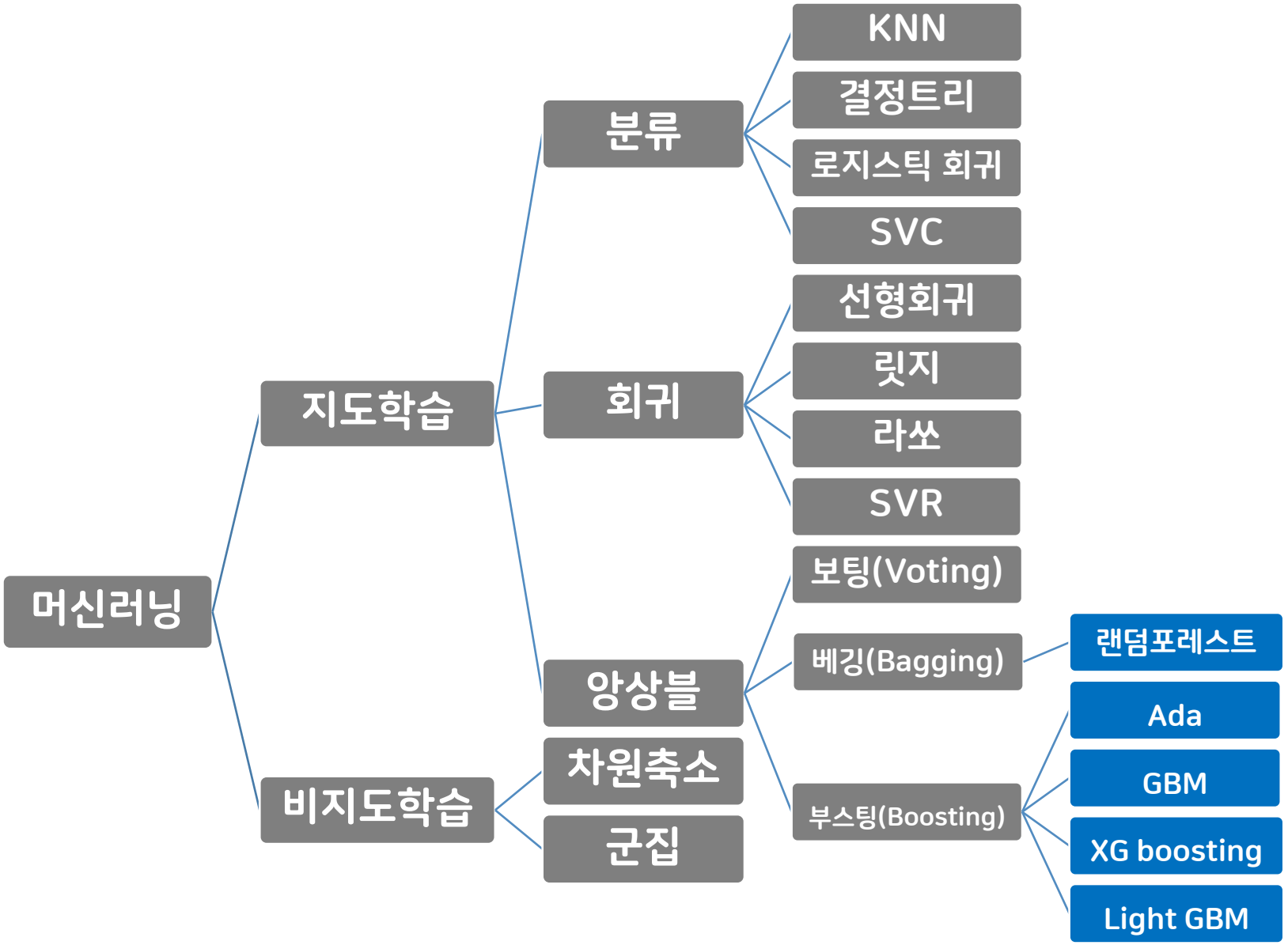


https://www.slideshare.net/freepsw/boosting-bagging-vs-boosting

베깅(Bagging), 부스팅(Boosting)

구 분	베깅(Bagging)	부스팅(Boosting)
특징	· 같은 종류의 모델이 투표를 통해 최종 예측 결과를 도출 (데이터 샘플을 다르게 가져감)	· 연속 학습+예측 (이전 모델의 오류를 고려)
목적	· 과대적합 방지 · 일반적으로 좋은 모델을 만들기 위해	· 과소적합 방지 · 맞추기 어려운 문제를 풀기 위해
적합한 상황	· 분산 및 표준편차가 큰 모델 (High variance, Low bias)	· 데이터의 수가 적은 모델 · 오차가 큰 모델 (Low variance, High bias)
대표 모델	· Random Forest	· Ada Boosting, Gradient Boosting, XG Boosting, Light GBM
데이터 선택	· 무작위 선택	· 무작위 선택 (오류 데이터에 가중치 적용)

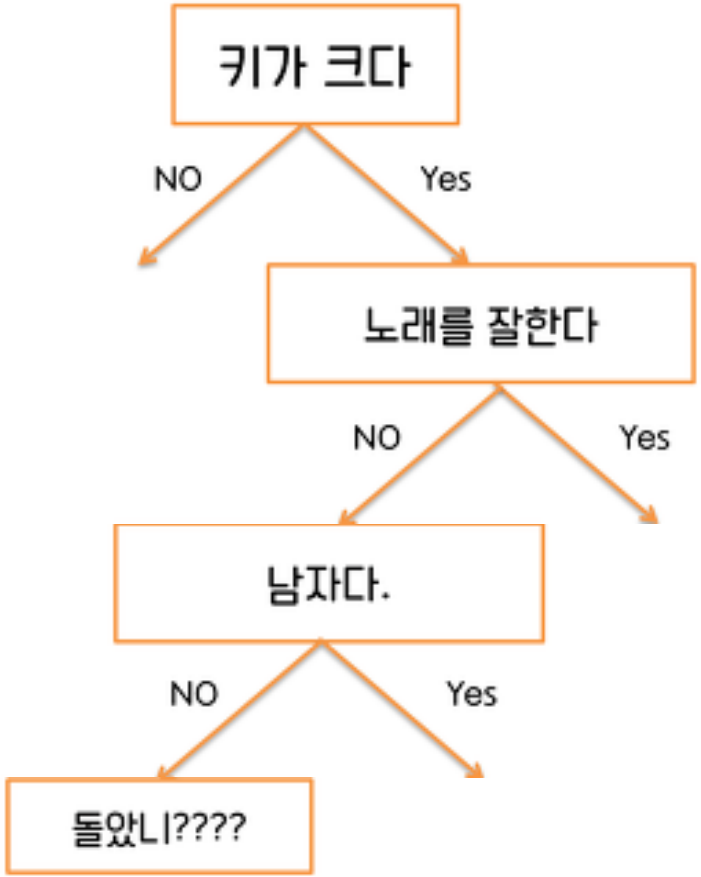
머신러닝 모델 개략도


머신러닝
지도학습
분류
KNN
결정트리
로지스틱 회귀
SVC
회귀
선형회귀
릿지
라쏘
SVR
앙상블
보팅(Voting)
베깅(Bagging)
랜덤포레스트
부스팅(Boosting)
Ada
GBM
XG boosting
Light GBM
비지도학습
차원축소
군집

랜덤 포레스트(Random forest)

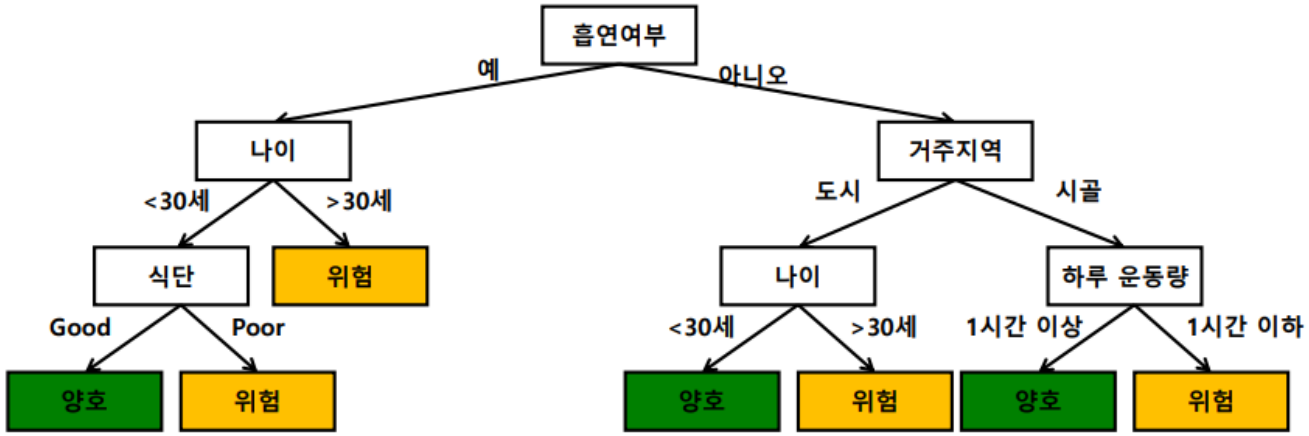
여러 개의 결정 트리 모델로 예측한 값을 투표를 통해서 최종 선택하는 베깅의 대표적 모델

결정 트리(Decision Tree)

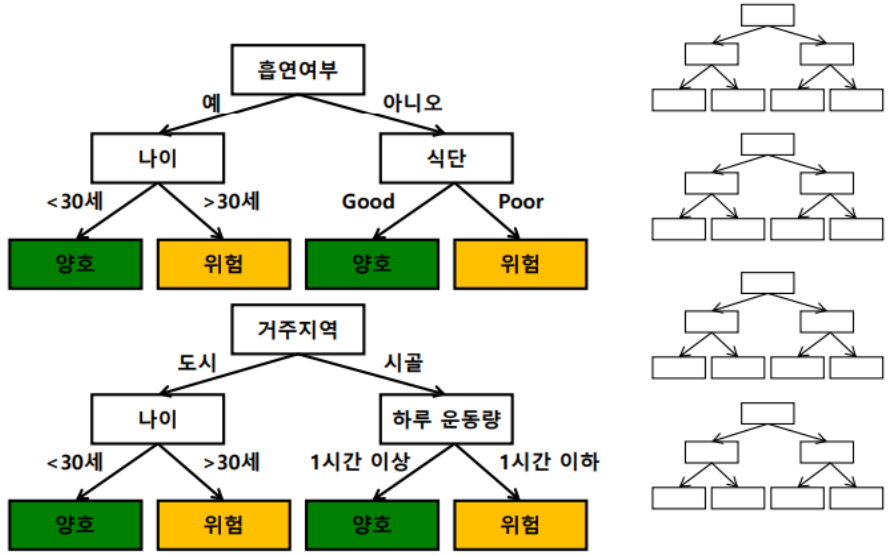


- 직관적이어서 결과를 쉽게 이해할 수 있음
- 과대적합이 되기 쉬움

랜덤 포레스트(Random forest)



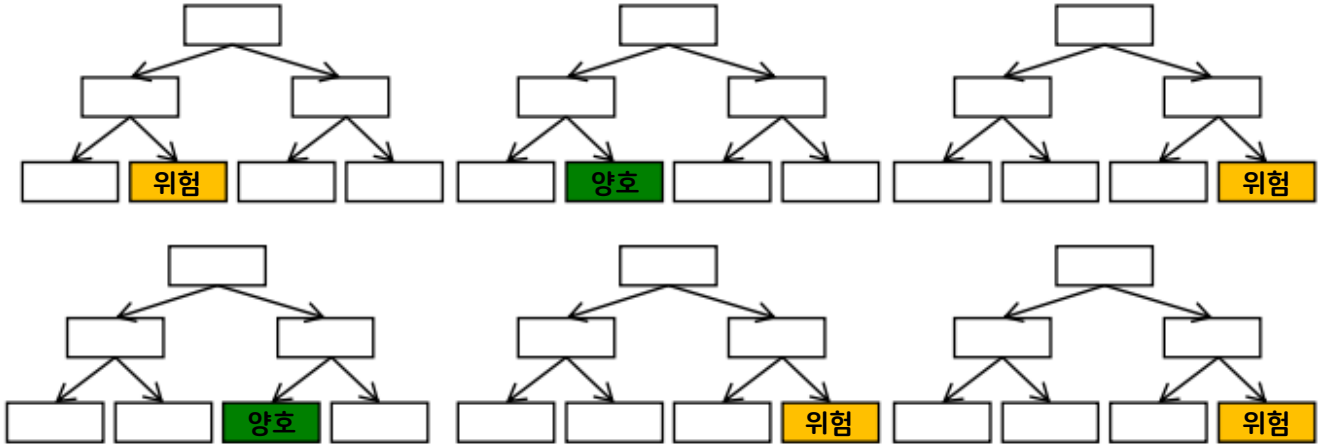
건강 위험도를 예측하기 위한 결정 트리



건강 위험도를 예측하기 위한 랜덤 포레스트

랜덤 포레스트(Random forest)

- 다수의 의사결정트리의 **의견이 통합되지 않는다면** → 투표에 의한 **다수결의 원칙**을 따름 → **앙상블 방법** (Ensemble Methods)
- 장점 : 실제값에 대한 추정값 오차 평균화, 분산 감소, 과적합 감소



위험으로 판정 (양호 2, 위험 4)

랜덤 포레스트(Random forest)

주요 매개변수(Hyperparameter)

scikit-learn의 경우

```
RandomForestClassifier(n_estimators,
max_features, random_state)
```

- 트리의 개수 : n_estimators
- 선택할 특징의 최대 수 : max_features
  (1로 하면 특성을 고려하지 않으며 큰 값이면 DT와 비슷해짐)
- 선택할 데이터의 시드 : random_state

랜덤 포레스트(Random forest)

결정 트리 매개변수(Hyperparameter)

- 트리의 최대 깊이 : max_depth
- 말단 노드 최대 개수 : max_leaf_nodes
- 말단 노드가 되기 위한 최소 샘플 수 : min_samples_leaf

랜덤 포레스트(Random forest)

특징

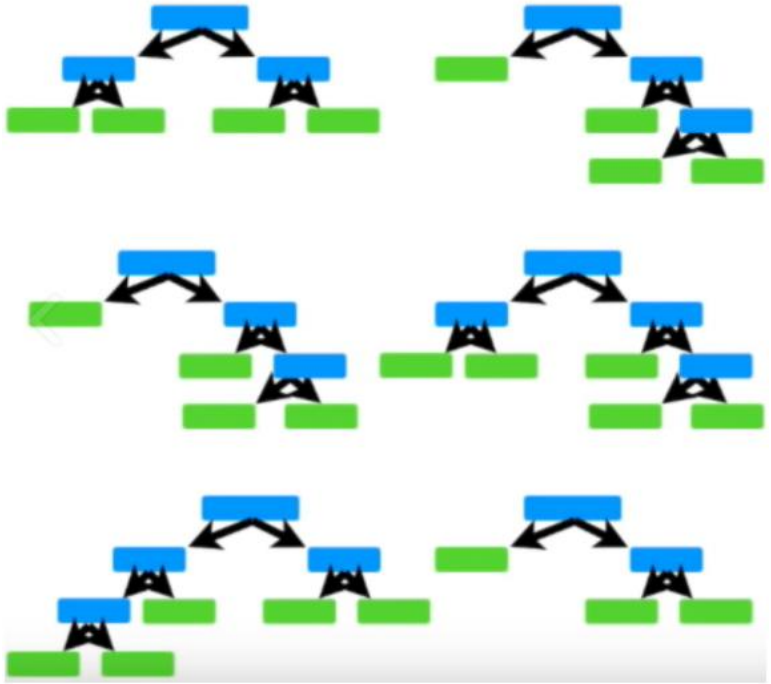
- 결정 트리 모델의 과대적합을 통계적 방법으로 해소
- 결정 트리 모델처럼 쉽고 직관적임
- 앙상블 모델 중 비교적 빠른 수행 속도
- 모델 튜닝을 위한 시간이 많이 필요(하이퍼 파라미터의 종류가 많음)
- 큰 데이터 세트에도 잘 동작하지만 트리 개수가 많아질수록 시간이 오래 걸림

에이다 부스팅(Ada boosting)

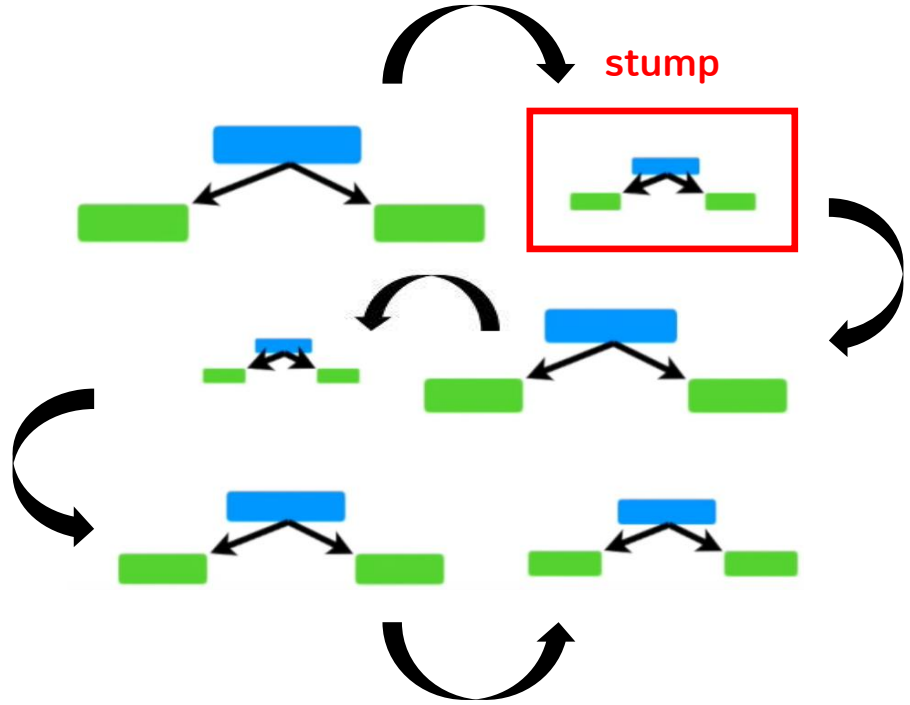
AdaBoost (Adaptive Boosting)

- RF처럼 의사결정 트리 기반의 모델 → 각각의 트리들이 독립적으로 존재하지 않음

약한 학습기(Weak learner)



Random Forest



Ada Boosting

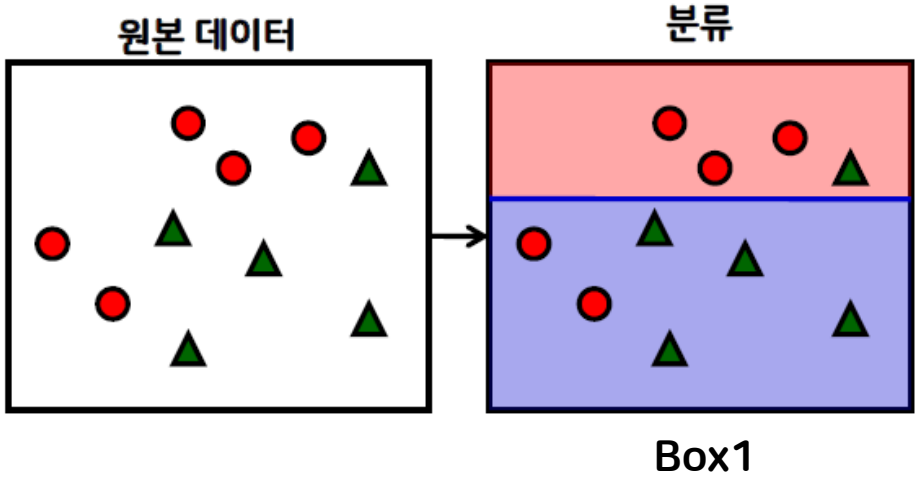
에이다 부스팅(Ada boosting)

AdaBoost (Adaptive Boosting)

- 동작 순서

(1) 첫 번째 의사결정 트리를 생성 → 위쪽 빨간 원이 3개 있는 곳을 대충 분류 시킴

→ 2개의 빨간 원과 1개의 녹색 세모가 잘못 구분됨

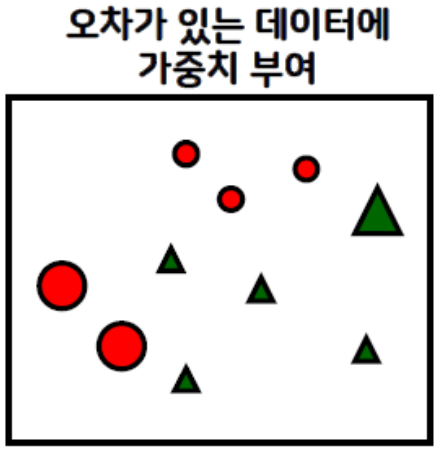


에이다 부스팅(Ada boosting)

AdaBoost (Adaptive Boosting)

- 동작 순서

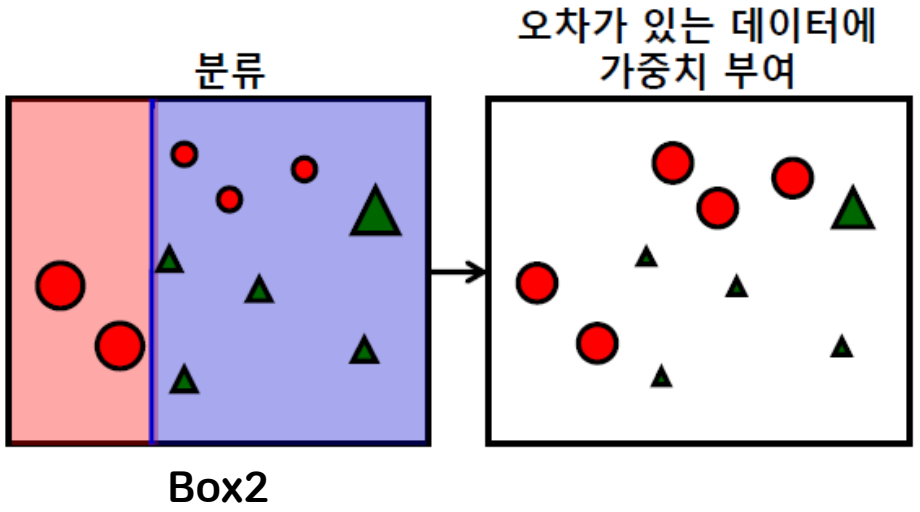
(2) 잘못된 2개의 빨간 원과 1개의 녹색 세모에 높은 가중치를 부여하고 맞은 것에는 빨간 원 3개와 녹색 세모 4개는 낮은 가중치 부여



AdaBoost (Adaptive Boosting)

- 동작 순서

(3) 가중치를 부여한 상태에서 다시 분류 시킴 → 잘못된 3개의 빨간 원에 높은 가중치를 부여하고 맞은 5개의 녹색 세모는 낮은 가중치를 부여

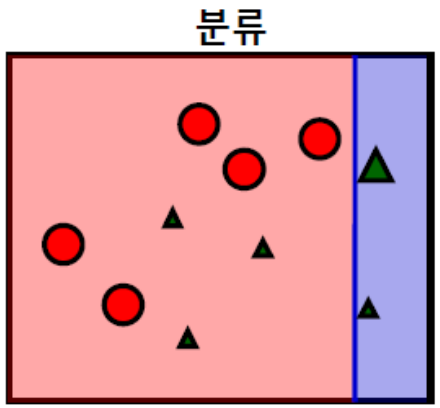


에이다 부스팅(Ada boosting)

AdaBoost (Adaptive Boosting)

- 동작 순서

(4) 가중치를 부여한 상태에서 다시 분류 시킴



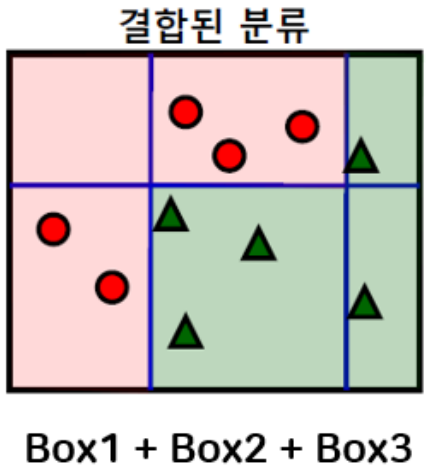
Box3

에이다 부스팅(Ada boosting)

AdaBoost (Adaptive Boosting)

- 동작 순서

(5) 진행한 분류들을 결합한다.



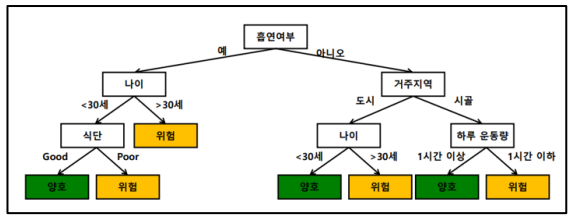
- 에이다 부스팅은 학습과 예측을 진행할수록 데이터들의 가중치가 달라짐
  - 잘못 분류된 데이터는 가중치↑, 잘 분류된 데이터는 가중치 ↓

- 각 모델들의 가중치 또한 다르게 설정됨
  - 예측률이 높은 모델은 가중치 ↑, 낮은 모델은 가중치↓

  예시) 0.4*Box1 + 0.7*Box2 + 0.5*Box3

에이다 부스팅(Ada boosting)

Decision Trees vs Random Forest



국, 영, 수

A학교 내신 1등

국어1등 수학1등

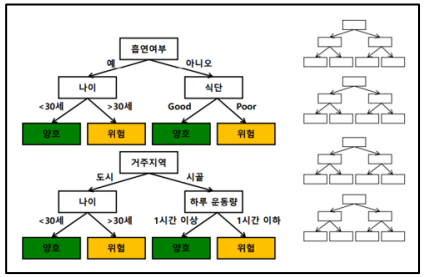
A학교 5등 | B학교 5등

C학교 5등 | D학교 5등

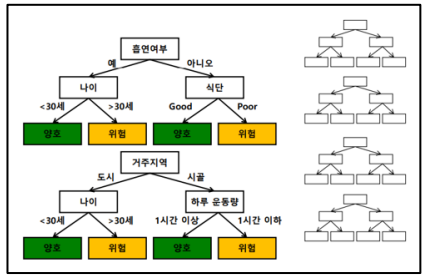
역사1등 체육1등

서로에 대해 모름

지식의 중복 중간



Random Forest vs AdaBoost



서로에 대해 모름

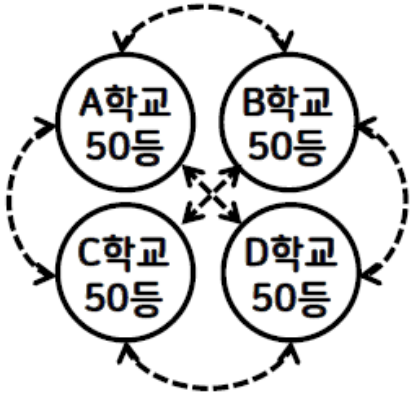
지식의 중복 중간

국어1등 수학1등

A학교 5등 | B학교 5등

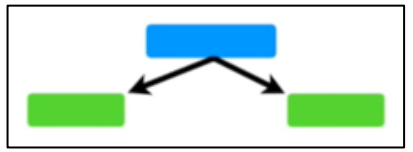
C학교 5등 | D학교 5등

역사1등 체육1등



서로에 대해 잘 알고 있음

지식의 중복 적음



http://medium.com/@deepvalidation/군중은-똑똑하다2-adaboost-ba0122500034 정리

에이다 부스팅(Ada boosting)

주요 매개변수(Hyperparameter)

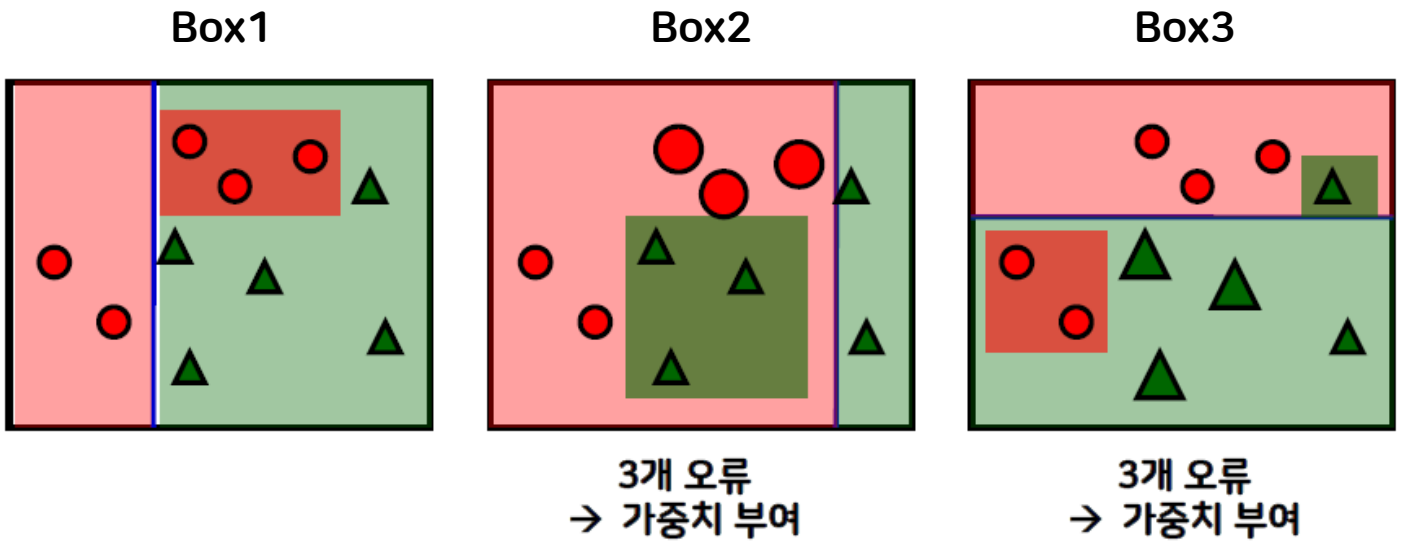
scikit-learn의 경우

```
AdaBoostClassifier(n_estimators,
random_state)
```

- 트리의 개수 : n_estimators
- 선택할 데이터의 시드 : random_state

그레디언트 부스팅(Gradient Boosting Machine)

- AdaBoost와 기본 개념이 동일하고 가중치를 계산하는 방식에서 **경사하강법**를 이용하여 최적의 가중치(파라미터)를 찾아냄



https://www.slideshare.net/freepsw/boosting-bagging-vs-boosting

그레디언트 부스팅(Gradient Boosting Machine)

주요 매개변수(Hyperparameter)

scikit-learn의 경우

```
GradientBoostingClassifier(n_estimators,
learning_rate, max_depth, random_state)
```

- 트리의 개수 : n_estimators
- 학습률 : learning_rate (높을수록 오차를 많이 보정)
- 트리의 깊이 : max_depth
- 선택할 데이터의 시드 : random_state
- 학습에 사용하는 데이터 샘플링 비율 : subsample

그레디언트 부스팅(Gradient Boosting Machine)

장단점

- 학습속도가 느림(부스팅의 일반적인 단점)
- 특성의 스케일을 조정할 필요가 없음(트리 기반 모델의 특성)
- 머신 러닝의 성능을 마지막까지 쥐어짜 극대화 시켜야 할 때 사용

XG Boosting(eXtreme Gradient Boosting)

- GBM의 단점 : 느림, 과대적합 문제
- GBM보다 빠름 → Early Stopping 제공
- 과대적합 방지를 위한 규제 포함
- CART (Classification And Regression Tree)을 기반으로 함 → 분류와 회귀가 모두 가능

XG Boosting(eXtreme Gradient Boosting)

주요 매개변수(Hyperparameter)

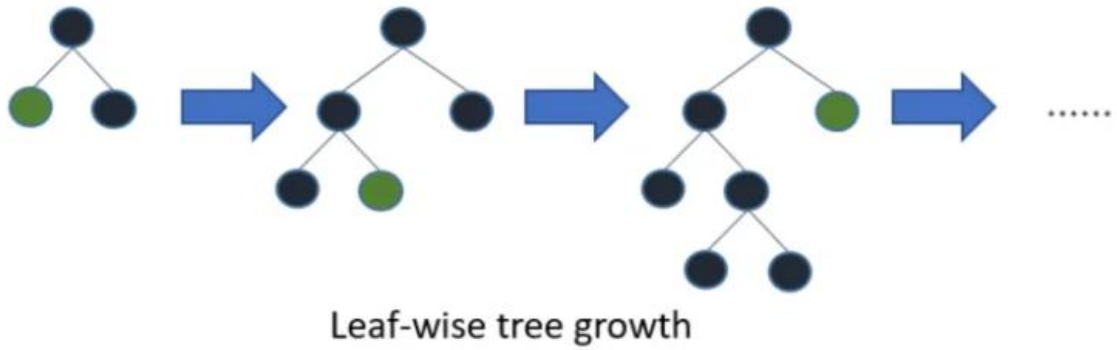
scikit-learn의 경우

```
XGBClassifier(n_estimators, learning_rate,
max_depth, random_state)
```

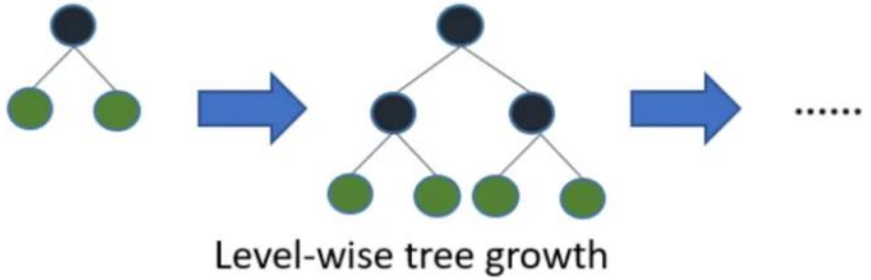
- 트리의 개수 : n_estimators
- 학습률 : learning_rate (높을수록 오차를 많이 보정)
- 트리의 깊이 : max_depth
- 선택할 데이터의 시드 : random_state

XG Boosting에 비해 가볍고(Low memory) 빠르며 정확도가 높은 모델

- Leaf-wise(수직방향, 비대칭)로 트리를 성장시킴(속도↑)
  - Level-wise(수평방향, 깊이↓, 대칭)보다 오류가 더 적음(정확도↑)



Light GBM



Random Forest, XG Boosting

장단점

- 대량(1만개 이상)의 데이터를 병렬로 빠르게 학습가능(Low Memory, GPU활용 가능)

  → XG Boosting 대비 2~10배의 속도(동일 파라미터 설정 시)

  → 소량의 데이터에서는 제대로 동작하지 않음(과대적합 위험)

- 예측 속도가 빠름 (Leaf-wise 트리의 장점)

  → 그러나 Level-wise에 비해 과적합에 민감

주요 매개변수(Hyperparameter)

- 100개 이상

- max_depth : 트리의 최대 깊이
- early_stopping_round : validation 데이터 중 하나의 지표가 정해진 반복 수 만큼 향상되지 않았다면 학습을 중단
- lambda : lambda 값은 regularization 정규화를 합니다. 일반적인 값의 범위는 0 에서 1 사이
- Min_data_in_leaf : Leaf노드가 가지고 있는 최소한의 레코드 수(디폴트 값 : 20, 과적합을 해결할 때 사용되는 파라미터)
- feature_fraction : 0.8 의 의미는 Light GBM이 Tree를 만들 때 매번 각각의 반복 학습 시 파라미터 중에서 80%를 랜덤하게 선택하는 것을 의미
- bagging_fraction : 매번 iteration을 돌 때 사용되는 데이터의 일부를 선택하는데 트레이닝 속도를 높이고 과적합을 방지할 때 주로 사용
- num_boost_round : boosting 반복 학습 수로 일반적으로 100 이상
- min_gain_to_split : 이 파라미터는 분기하기 위해 필요한 최소한의 gain을 의미, Tree에서 유용한 분기의 수를 컨트롤하는데 사용
- max_cat_group : 카테고리 수가 클 때, 과적합을 방지하는 분기 포인트를 찾음(디폴트 값 : 64)
- Task : 데이터에 대해서 수행하고자 하는 임무를 구체화, train일수도 있고 predict 예측일 수도 있음
- application : 문제 타입 설정, 디폴트는 회귀(regression: 회귀분석, binary: 이진 분류, multiclass: 다중 분류)
- learning_rate : 학습률(일반적인 값은 0.1, 0.001, 0.003 등)
- num_leaves : 전체 Tree의 leave 수 이고, 디폴트값은 31
- device : 디폴트 값은 CPU(GPU로 변경가능)