분류 알고리즘

분류 알고리즘의 종류

- 분류(Classification)는 학습 데이터로 주어진 데이터의 feature와 label 값(결정 값, 클래스 값)을 머신러닝 알고리즘으로 학습해 모델을 생성하고, 생성된 모델에 새로운 데이터 값이 주어졌을 때 미지의 레이블 값을 예측

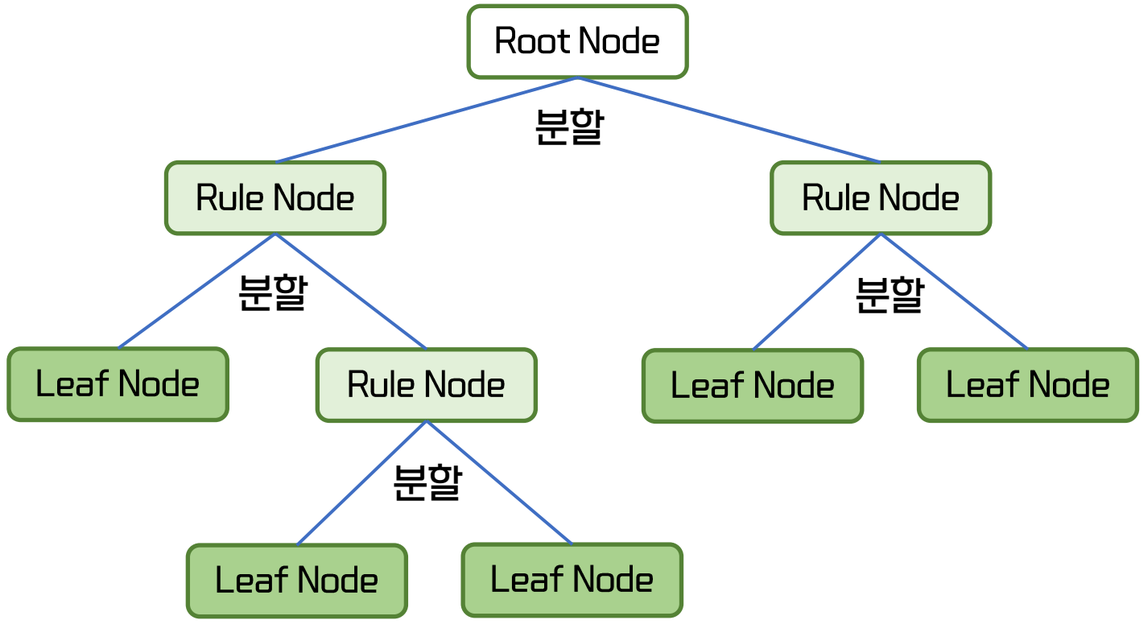
대표적인 분류 알고리즘

- 베이즈 통계와 생성 모델에 기반한 나이브 베이즈(Naive Bayes)
- 독립변수와 종속변수 선형 관계성에 기반한 로지스틱 회귀(Logistic Regression)
- 데이터 균일도에 따른 규칙 기반의 결정 트리(Decision Tree)
- 개별 클래스 간의 최대 분류 마진을 효과적으로 찾아주는 서포트 벡터 머신(Support Vector Machine)
- 근접 거리를 기준으로 하는 최소 근접 알고리즘 (Nearest Neighbor)
- 심층 연결 기반의 신경망
- 서로 다른(또는 같은) 머신러닝 알고리즘을 결합한 앙상블(Ensemble)

결정 트리

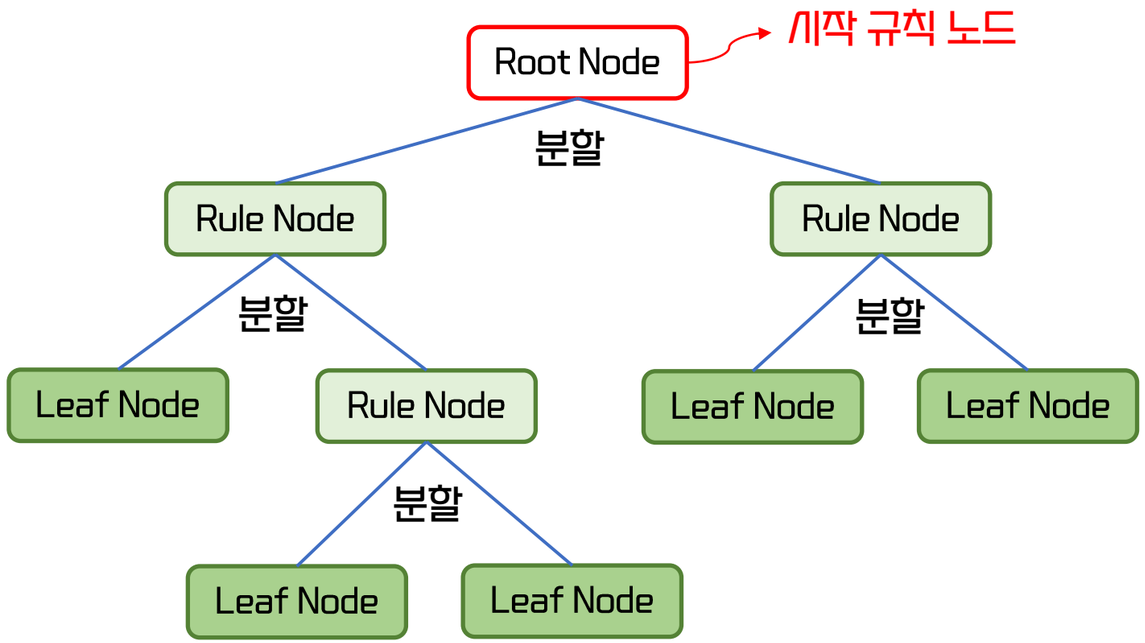
결정 트리

- 결정 트리 알고리즘은 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내는 트리 기반의 분류 규칙을 만든다.
- 따라서 데이터의 어떤 기준을 바탕으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가가 알고리즘의 성능을 크게 좌우한다.



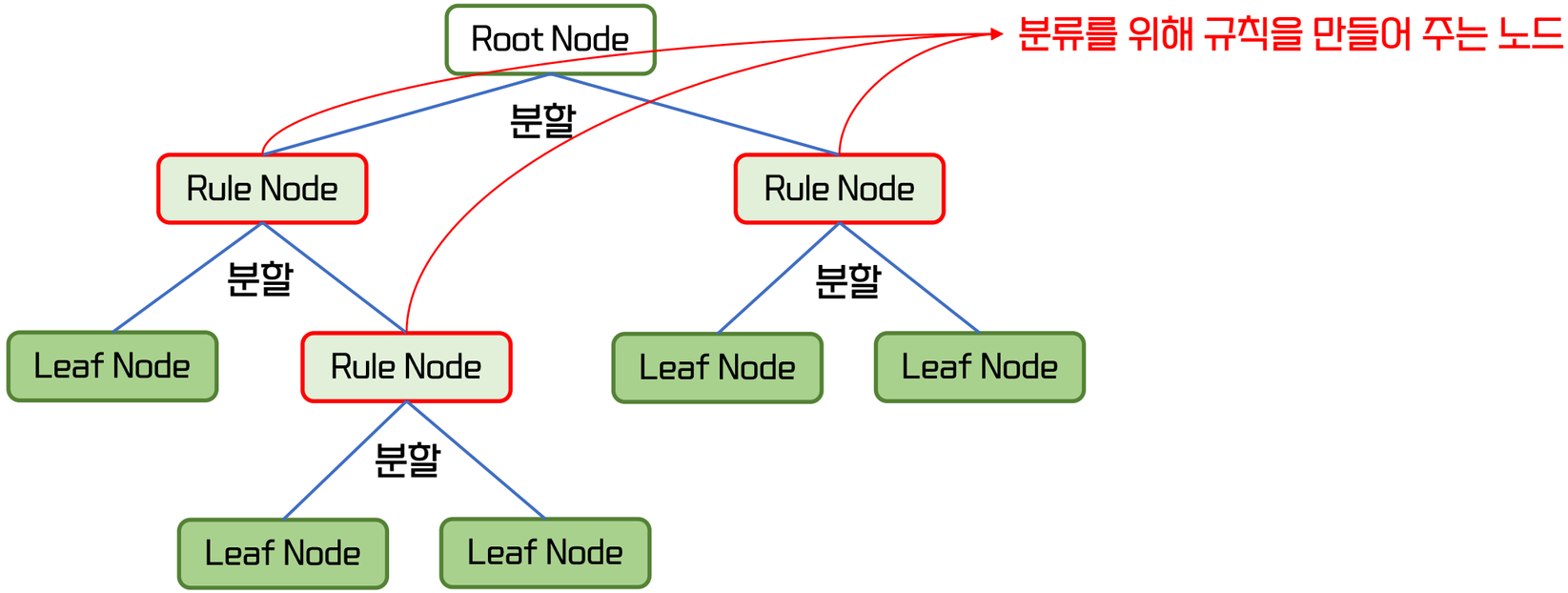
결정 트리

- 결정 트리 알고리즘은 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내는 트리 기반의 분류 규칙을 만든다.
- 따라서 데이터의 어떤 기준을 바탕으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가가 알고리즘의 성능을 크게 좌우한다.



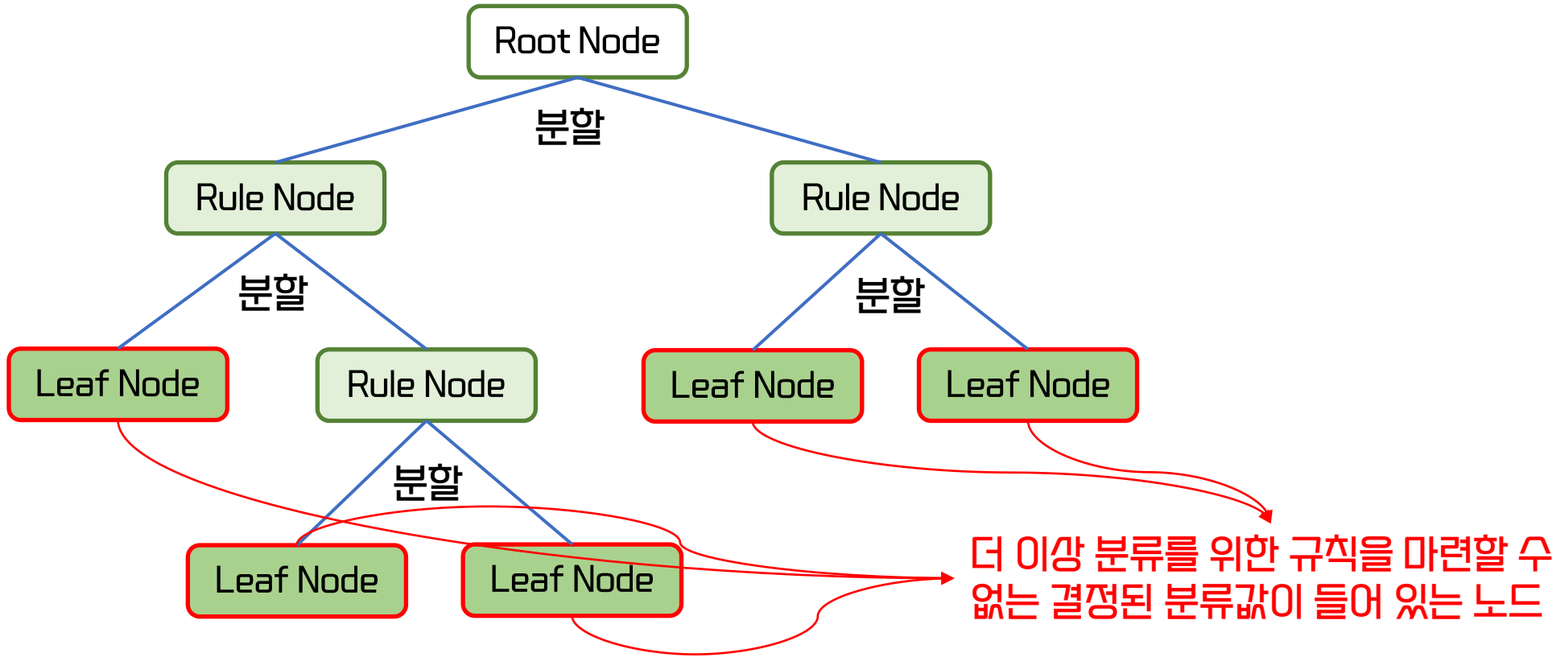
결정 트리

- 결정 트리 알고리즘은 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내는 트리 기반의 분류 규칙을 만든다.
- 따라서 데이터의 어떤 기준을 바탕으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가가 알고리즘의 성능을 크게 좌우한다.



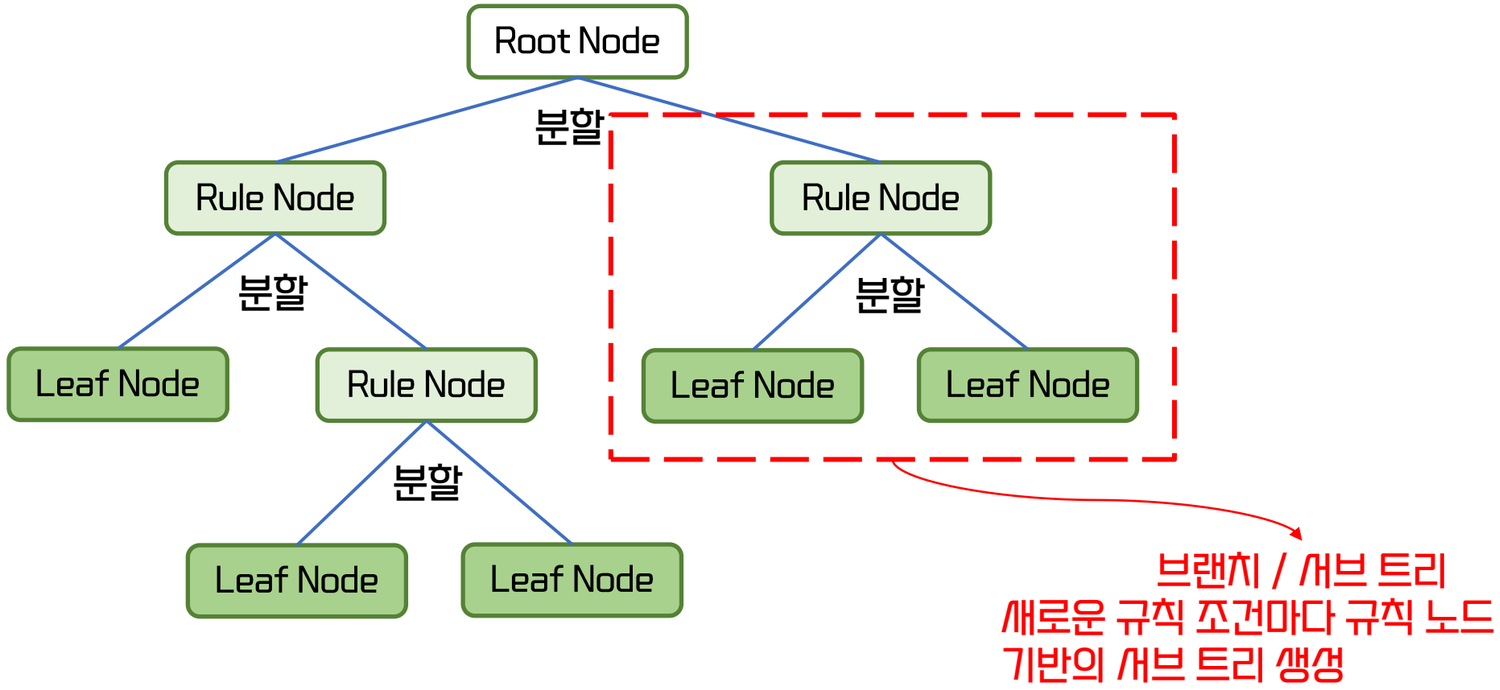
결정 트리

- 결정 트리 알고리즘은 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내는 트리 기반의 분류 규칙을 만든다.
- 따라서 데이터의 어떤 기준을 바탕으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가가 알고리즘의 성능을 크게 좌우한다.



결정 트리

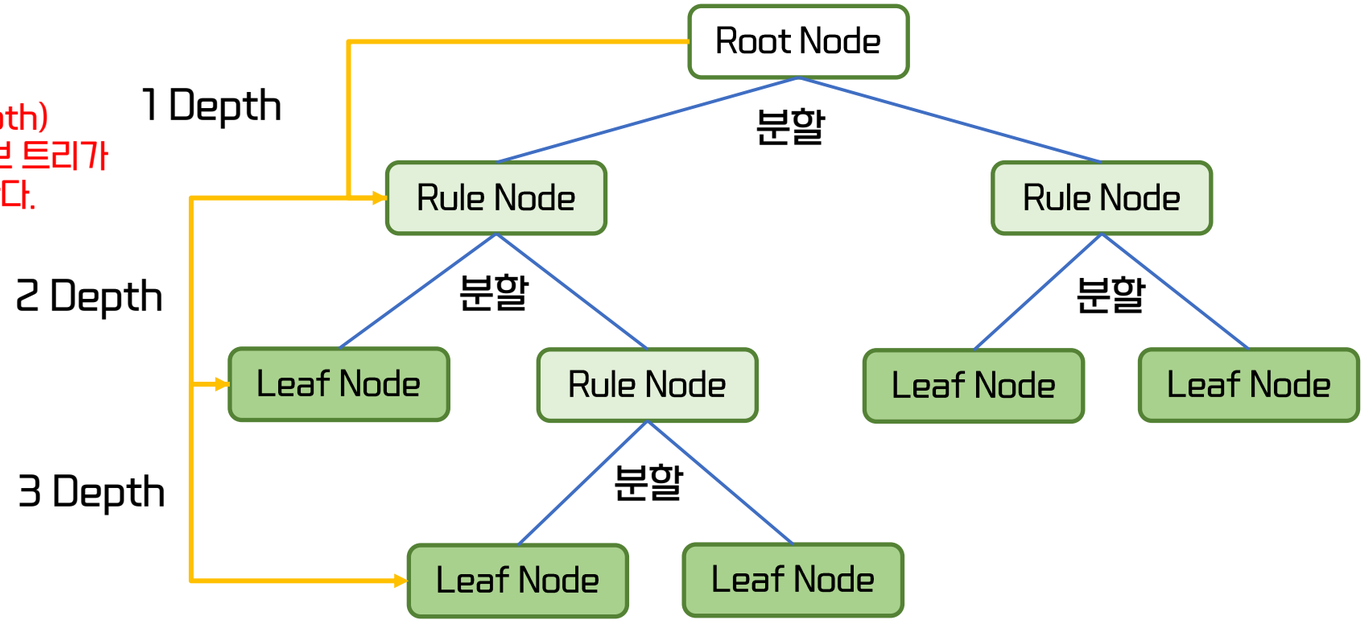
- 결정 트리 알고리즘은 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내는 트리 기반의 분류 규칙을 만든다.
- 따라서 데이터의 어떤 기준을 바탕으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가가 알고리즘의 성능을 크게 좌우한다.



결정 트리

- 결정 트리 알고리즘은 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내는 트리 기반의 분류 규칙을 만든다.
- 따라서 데이터의 어떤 기준을 바탕으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가가 알고리즘의 성능을 크게 좌우한다.

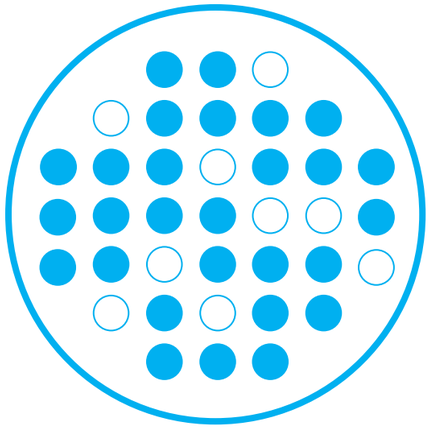
나무의 깊이(Depth)
Rule Node에 의해 서브 트리가 생성되면 깊이가 증가한다.



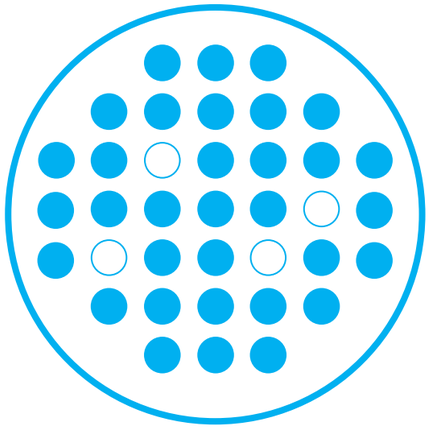
트리 분할을 위한 데이터의 균일도

다음 중 가장 균일한 데이터 세트는 무엇인가?

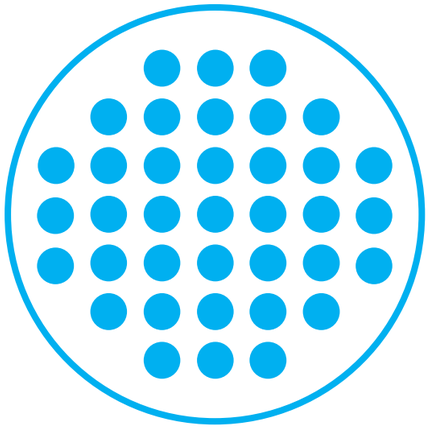
A



B



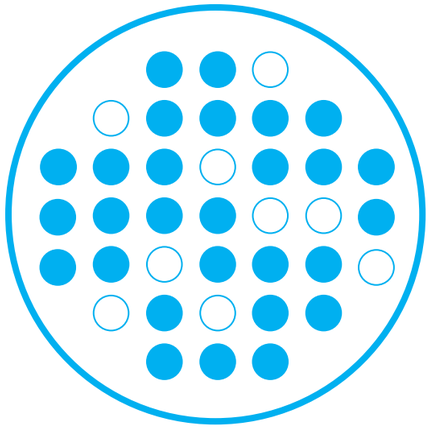
C



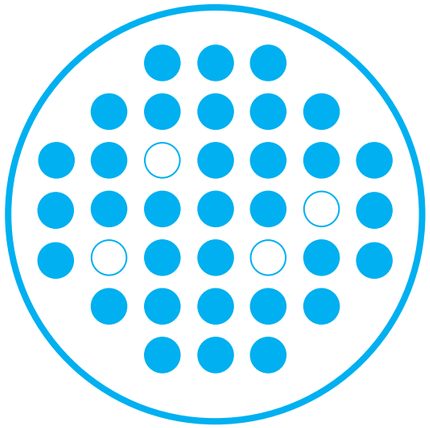
트리 분할을 위한 데이터의 균일도

다음 중 가장 균일한 데이터 세트는 무엇인가?

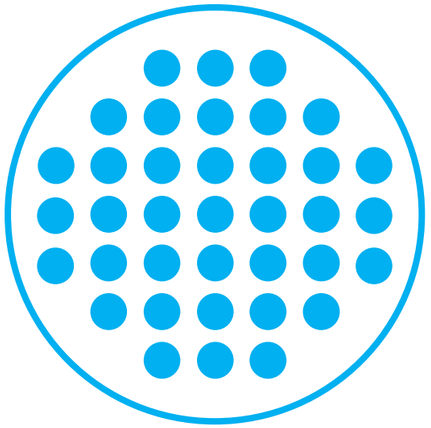
A



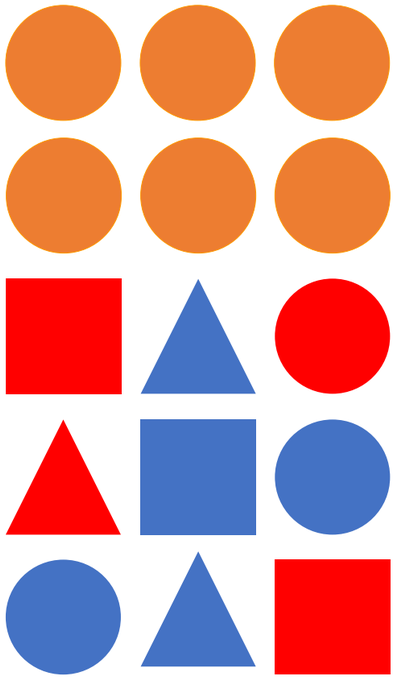
B



C

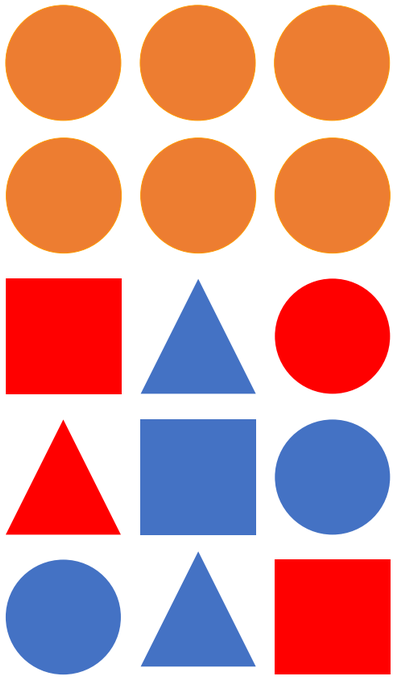


균일도 기반 규칙 조건



주황색 블록은 모두 동그라미로, 빨강과 파랑 블록의 경우는 동그라미, 세모, 네모가 골고루 섞여 있다고 한다면 각 레고 블록을 분류하고자 할 때 가장 첫 번째로 만들어져야 하는 규칙 조건은?

균일도 기반 규칙 조건



주황색 블록은 모두 동그라미로, 빨강과 파랑 블록의 경우는 동그라미, 세모, 네모가 골고루 섞여 있다고 한다면 각 레고 블록을 분류하고자 할 때 가장 첫 번째로 만들어져야 하는 규칙 조건은?

if 색깔 == 주황색

정보 균일도 측정 방법

- 정보 이득(Information Gain)
  - 정보 이득은 엔트로피라는 개념을 기반으로 주어진 데이터의 혼잡도를 사용
  - 서로 다른 값이 섞여 있으면 엔트로피가 높고, 같은 값이 섞여 있으면 엔트로피가 낮다.
  - 정보 이득 지수는 1에서 엔트로피 지수를 뺀 값.
    - 정보 이득 지수 = 1 – 엔트로피 지수
  - 결정 트리는 이 정보 이득 지수로 분할 기준을 정한다. 즉 정보 이득이 높은 속성을 기준으로 분할한다.

정보 균일도 측정 방법

- 지니 계수
  - 지니 계수는 원래 경제학에서 불평등 지수를 나타낼 때 사용하는 지수
  - 0이 가장 평등하고, 1로 갈수록 불평등해 진다.
  - 머신러닝에 적용될 때는 지니 계수가 낮을 수록 데이터 균일도가 높은 것으로 해석되어 계수가 낮은 속성을 기준으로 분할 한다.

결정 트리의 규칙 노드 생성 프로세스

1 데이터 집합의 모든 아이템이 같은 분류에 속하는지 확인

결정 트리의 규칙 노드 생성 프로세스

1. 데이터 집합의 모든 아이템이 같은 분류에 속하는지 확인

2.1 데이터의 집합의 모든 아이템이 한 종류라면 리프 노드로 만들어서 분류 결정

결정 트리의 규칙 노드 생성 프로세스

1. 데이터 집합의 모든 아이템이 같은 분류에 속하는지 확인

2.1 데이터의 집합의 모든 아이템이 한 종류라면 리프 노드로 만들어서 분류 결정

2.2 데이터의 집합의 아이템이 여러 종류라면 데이터를 분할하는데 가장 좋은 속성과 분할 기준을 찾음(정보 이득 or 지니 계수)

결정 트리의 규칙 노드 생성 프로세스

1. 데이터 집합의 모든 아이템이 같은 분류에 속하는지 확인

2.1 데이터의 집합의 모든 아이템이 한 종류라면 리프 노드로 만들어서 분류 결정

2.2 데이터의 집합의 아이템이 여러 종류라면 데이터를 분할하는데 가장 좋은 속성과 분할 기준을 찾음(정보 이득 or 지니 계수)

3. 해당 속성과 분할 기준으로 데이터를 분할하여 규칙 노드 생성

결정 트리의 규칙 노드 생성 프로세스

1. 데이터 집합의 모든 아이템이 같은 분류에 속하는지 확인

2.1 데이터의 집합의 모든 아이템이 한 종류라면 리프 노드로 만들어서 분류 결정

2.2 데이터의 집합의 아이템이 여러 종류라면 데이터를 분할하는데 가장 좋은 속성과 분할 기준을 찾음(정보 이득 or 지니 계수)

3. 해당 속성과 분할 기준으로 데이터를 분할하여 규칙 노드 생성

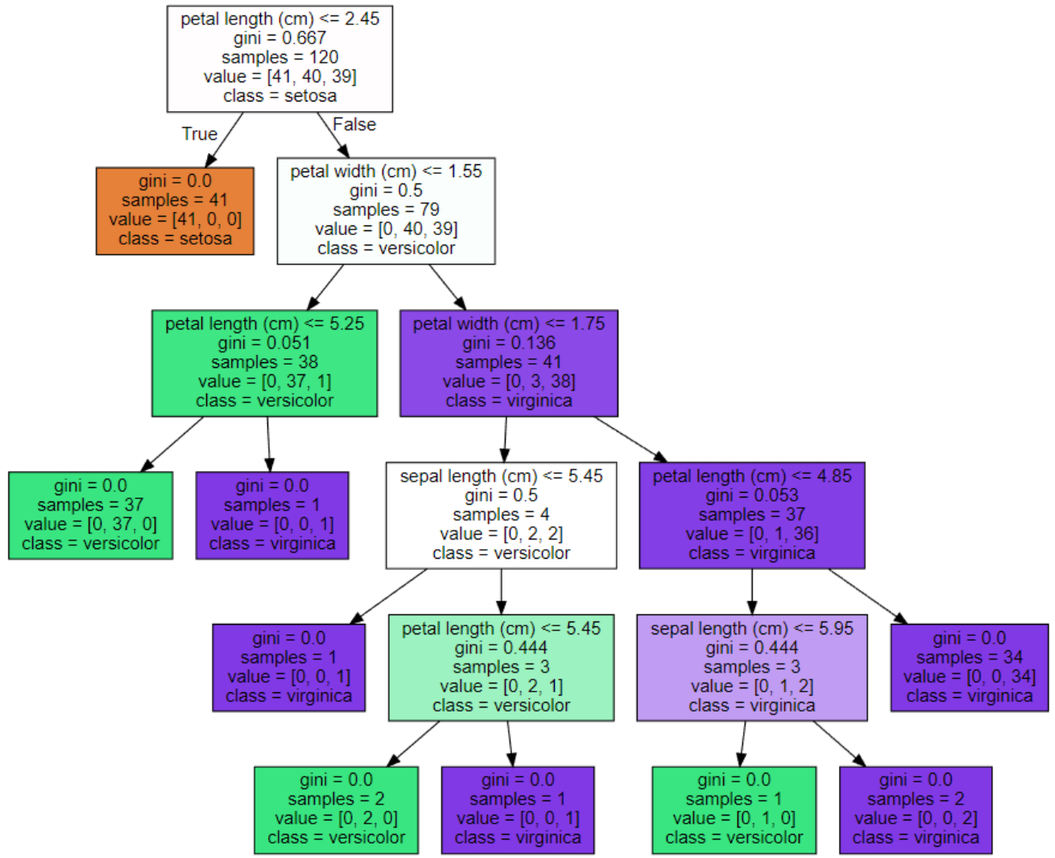
모든 데이터 집합의 분류가 결정 될 때 까지 1 – 2.2 – 3 단계가 반복된다.

결정 트리의 장단점

- 장점
  - 쉽고 직관적이다. 시각화를 통해 모델이 어떻게 학습했는지 보기 쉽다.
  - feature의 스케일링이나 정규화 등의 사전 가공 영향도가 크지 않다.

- 단점
  - 과적합(Overfitting)으로 알고리즘 성능이 떨어진다.
  - 이를 극복하기 위해 트리의 크기를 사전에 제한하는 튜닝이 필요하다.

Graphviz를 이용한 Tree 시각화



결정 트리 주요 하이퍼 파라미터

- max_depth
  - 트리의 최대 깊이를 규정
  - Default는 None으로 설정하면 완벽하게 클래스 결정 값이 될 때까지 깊이를 계속 키우며 분할하거나 노드가 가지는 데이터 개수가 min_samples_split보다 작아질 때 까지 계속 깊이를 증가시킴
  - 깊이가 깊어지면 min_samples_split 설정대로 최대 분할하여 과적합할 수 있으므로 적절한 값으로 제어 필요

결정 트리 주요 하이퍼 파라미터

- max_features
  - 최적의 분할을 위해 고려할 최대 feature의 개수
  - Default는 None으로 모든 feature를 사용
  - max_features=정수 설정하면 개수, max_features=실수 설정하면 비율
  - max_features='sqrt' : feature 개수의 제곱근
  - max_features='auto': sqrt와 같음
  - max_features='log' : log(feature 개수)
- min_samples_split
  - 노드를 분할하기 위한 최소한의 샘플 데이터 개수
  - Default는 2이고, 작게 설정할 수록 분할되는 노드가 많아져 과적합 가능성 증가

결정 트리 주요 하이퍼 파라미터

- min_samples_leaf
  - 규칙 노드에서 분할된 왼쪽 / 오른쪽 노드를 만들기 위해 가지고 있어야 할 최소한의 샘플 데이터 개수
  - 큰 값으로 설정 될 수록 분할된 왼쪽 / 오른쪽 노드에서 가져야 할 최소한의 샘플 데이터 수 조건을 만족시키기가 어려우므로 상대적으로 노드 수행을 덜 수행한다.
- max_leaf_nodes
  - 말단 노드의 최대 개수

실습

사용자 행동 인식 예측 모델 만들기

앙상블 학습

앙상블 학습 개요

- 앙상블 학습을 통한 분류는 여러 개의 분류기(Classifier)를 생성하고 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법을 일컫는다.

- 복합적이고 어려운 문제의 결론을 내기 위해 각 분야 별 전문가들의 다양한 의견을 수렴하고 결정하듯이 앙상블 학습의 목표는 다양한 분류기의 예측 결과를 결합함으로써 단일 분류기보다 신뢰성이 높은 예측값을 얻는 것이다.

앙상블의 유형

- 앙상블의 유형은 보팅(Voting), 배깅(Bagging), 부스팅(Boosting)으로 구분할 수 있으며 이외에 스태킹(Stacking) 등의 기법이 있다.

- 대표적인 배깅 방식은 랜덤 포레스트 알고리즘이 있으며, 부스팅은 에이다 부스팅, 그라디언트 부스팅, XGBoost, LightGBM 등이 존재한다. 정형 데이터의 분류나 회귀에서는 GBM 부스팅 계열의 앙상블이 전반적으로 높은 예측 성능을 나타낸다.

- 넓은 의미로는 서로 다른 모델을 결합한 것들을 앙상블로 지칭하기도 한다.

앙상블의 특징

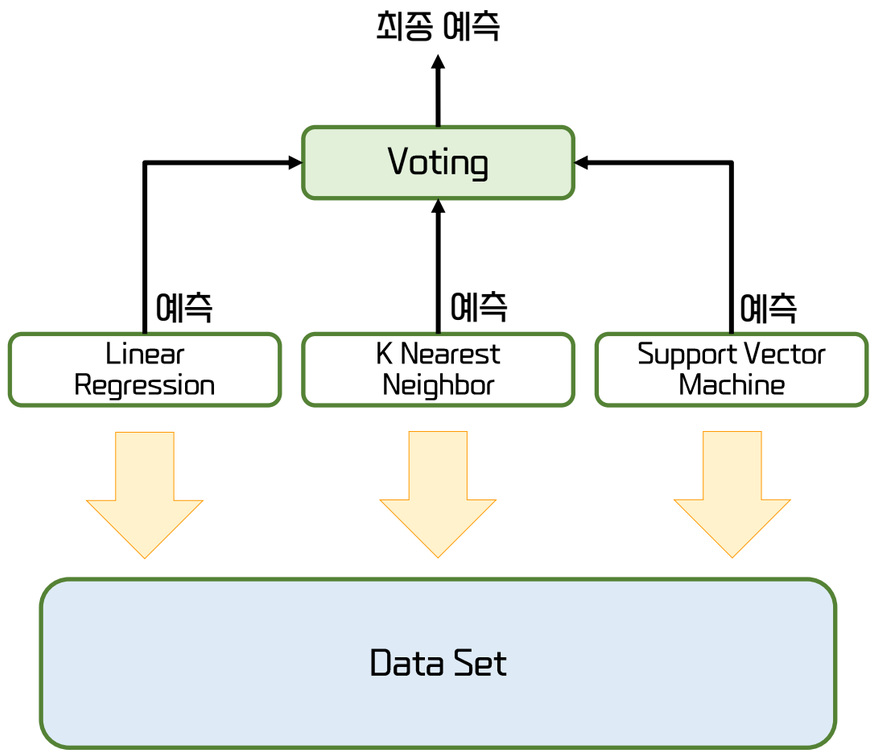
- 단일 모델의 약점을 다수의 모델들을 결합하여 보완
- 뛰어난 성능을 가진 모델들로만 구성하는 것보다 성능이 떨어지더라도 서로 다른 유형의 모델을 섞는 것이 오히려 전체 성능에 도움이 됨
- 랜덤 포레스트 및 뛰어난 부스팅 알고리즘들은 모두 결정 트리 알고리즘을 기반 알고리즘으로 적용함
- 결정 트리의 단점이 과적합(오버피팅)을 수십~수천개의 많은 분류기를 결합해 보완하고 장점인 직관적인 분류 기준은 강화됨

보팅(Voting), 배깅(Bagging)

보팅(Voting)과 배깅(Bagging) 개요

- 보팅과 배깅은 여러 개의 분류기가 투표를 통해 최종 예측 결과를 결정하는 방식

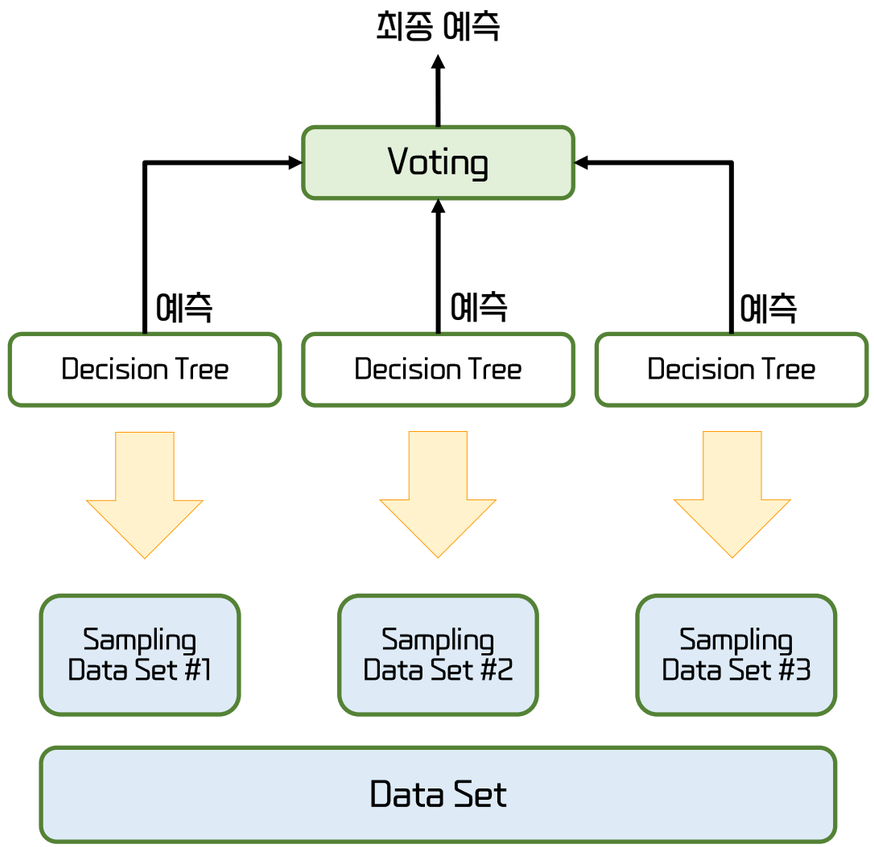
- 보팅과 배깅의 차이
  - 보팅은 서로 다른 알고리즘을 가진 분류기를 결합



보팅(Voting)과 배깅(Bagging) 개요

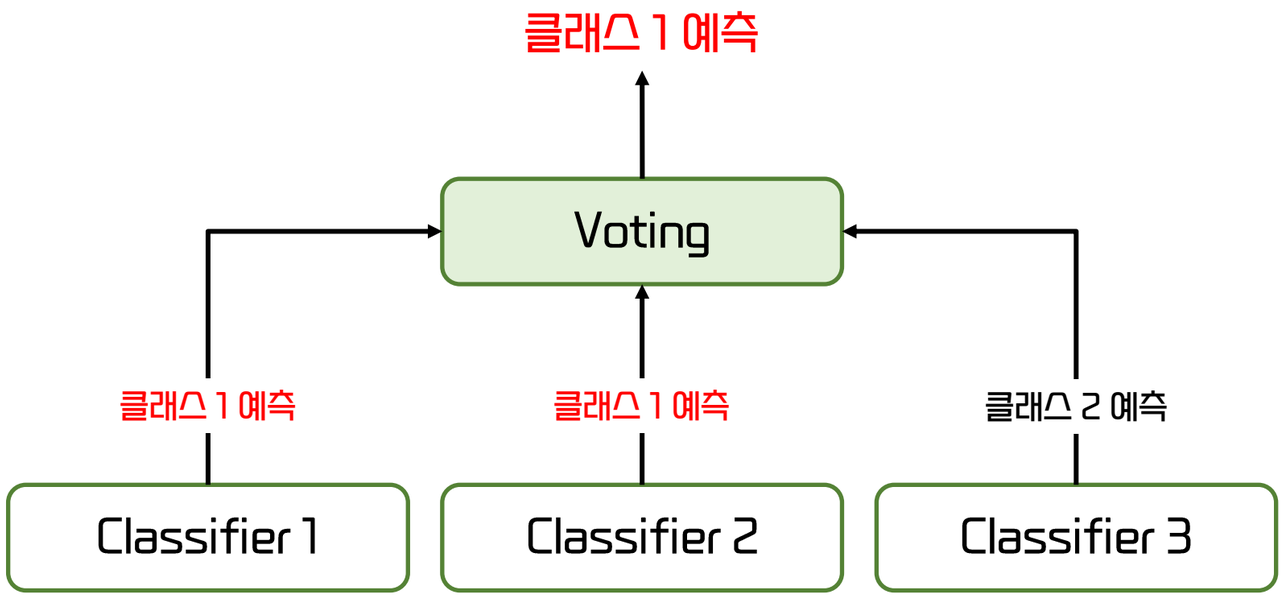
- 보팅과 배깅은 여러 개의 분류기가 투표를 통해 최종 예측 결과를 결정하는 방식

- 보팅과 배깅의 차이
  - 보팅은 서로 다른 알고리즘을 가진 분류기를 결합
  - 배깅은 같은 알고리즘을 가진 분류기를 결합하지만 데이터 샘플링을 서로 다르게 수행하면서 학습을 수행해 보팅을 수행



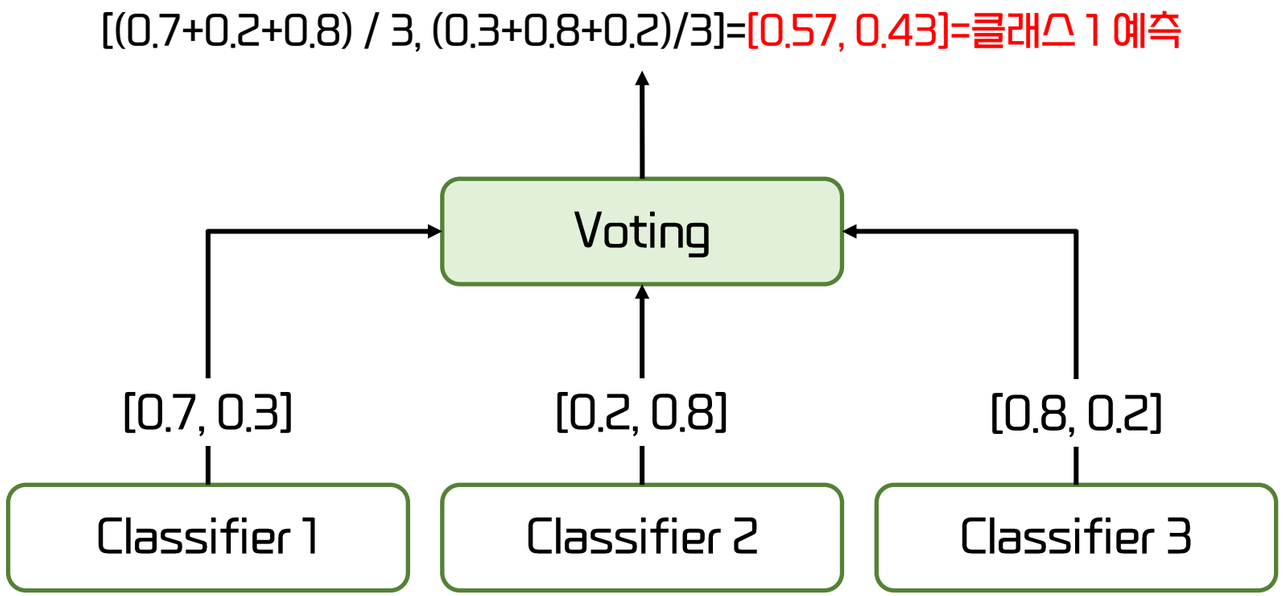
보팅의 종류 – Hard Voting, Soft Voting

- Hard Voting은 다수의 Classifier 간 다수결로 최종 class 결정



보팅의 종류 – Hard Voting, Soft Voting

- Soft Voting은 다수의 Classifier 들의 class 확률을 평균하여 결정. 일반적으로 많이 사용된다.



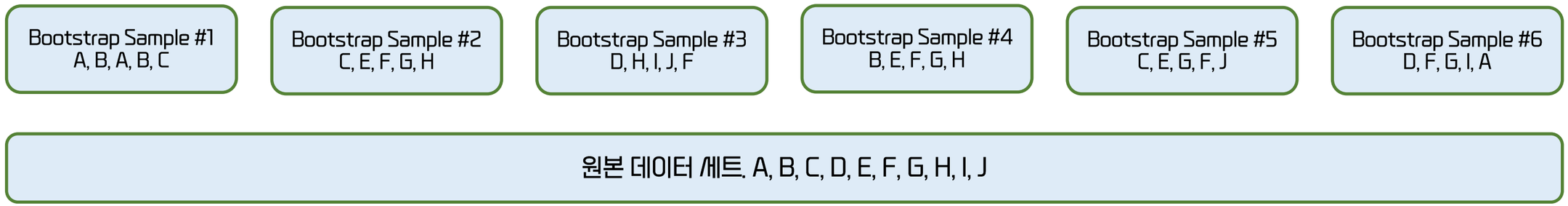
실습

VotingClassifier를 이용한 유방암 예측 분류기 생성

배깅(Bagging)

- Bagging은 Bootstrap Sampling의 줄임말

- Bootstrap이란 기존 학습 데이터 세트로 부터 랜덤하게 복원추출 하여 동일한 사이즈의 데이터 세트를 여러 개 만드는 것을 의미한다.

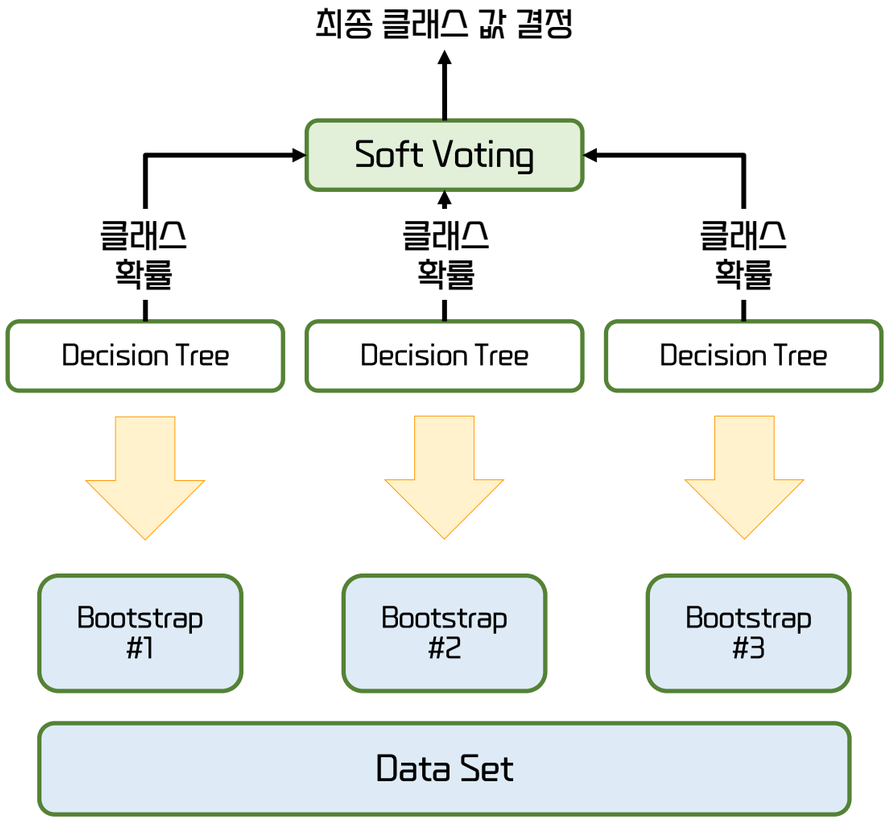


배깅(Bagging) - 랜덤 포레스트(Random Forest)

- 배깅의 대표적인 알고리즘은 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 결정을 한다.

- Bootstrap을 통해 데이터를 Random하게, Tree를 모아 놓은 구조이기 때문에 숲(Forest)이 붙어 RandomForest가 되었다.



랜덤 포레스트 주요 하이퍼 파라미터

- n_estimators : 랜덤 포레스트에서 결정 트리의 개수를 지정. 기본은 100개
  - 많이 설정할수록 좋은 성능을 기대할 수 있지만 무조건 향상 되는 것은 아미
  - 트리의 개수가 많아질수록 학습 수행 시간이 오래 걸리는 것도 감안해야 한다.

- max_features : 결정 트리에 사용된 max_features 파라미터와 같다.
  - 기본 max_features는 'auto' 로서 'sqrt'를 사용한다.
  - 랜덤 포레스트의 트리를 분할하는 피처를 참조할 때 전체 피처가 아니라 sqrt(전체 피처 개수)만큼 참조한다.

- max_depth나 min_samples_leaf와 같이 결정 트리에서 과적합을 개선하기 위해 사용되는 파라미터가 랜덤 포레스트에도 똑같이 적용될 수 있다.

실습

랜덤 포레스트 기반의 사용자 행동 인식 예측 모델 만들기

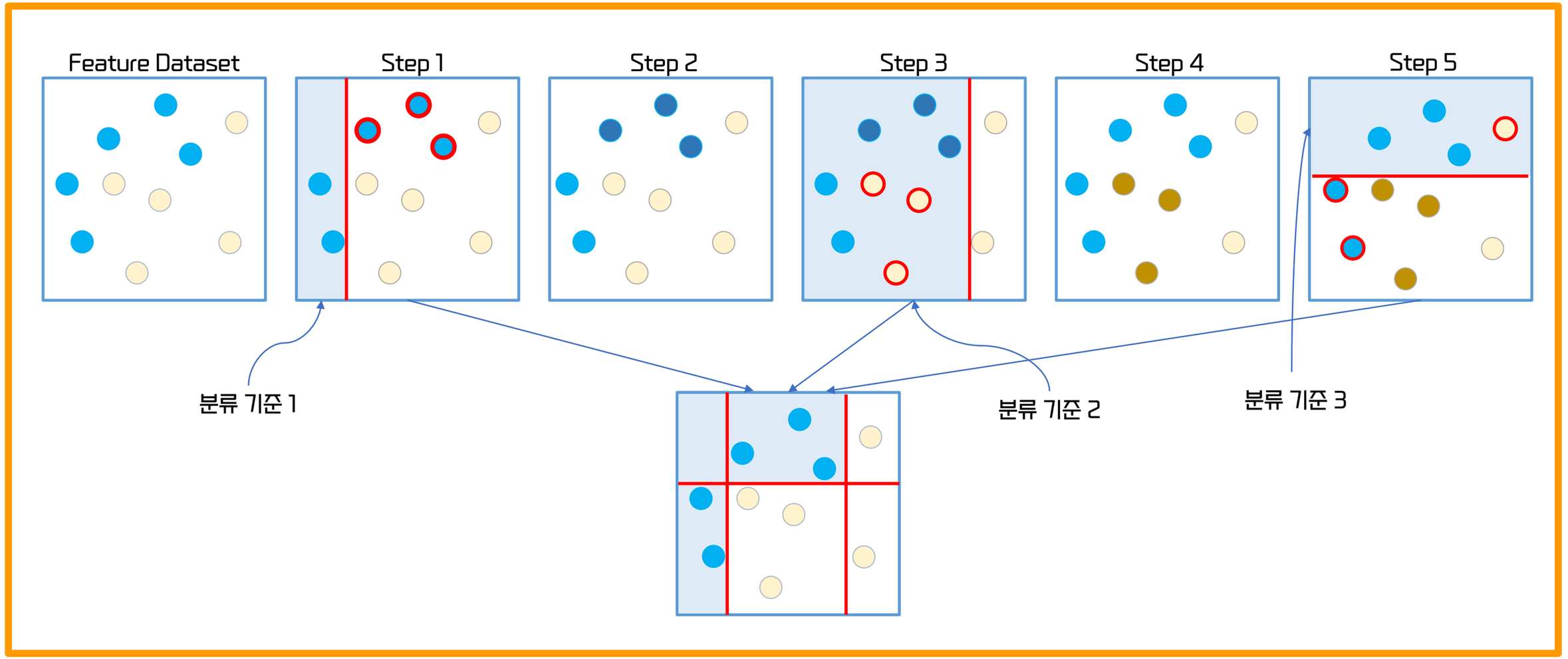
부스팅(Boosting)

부스팅(Boosting)이란?

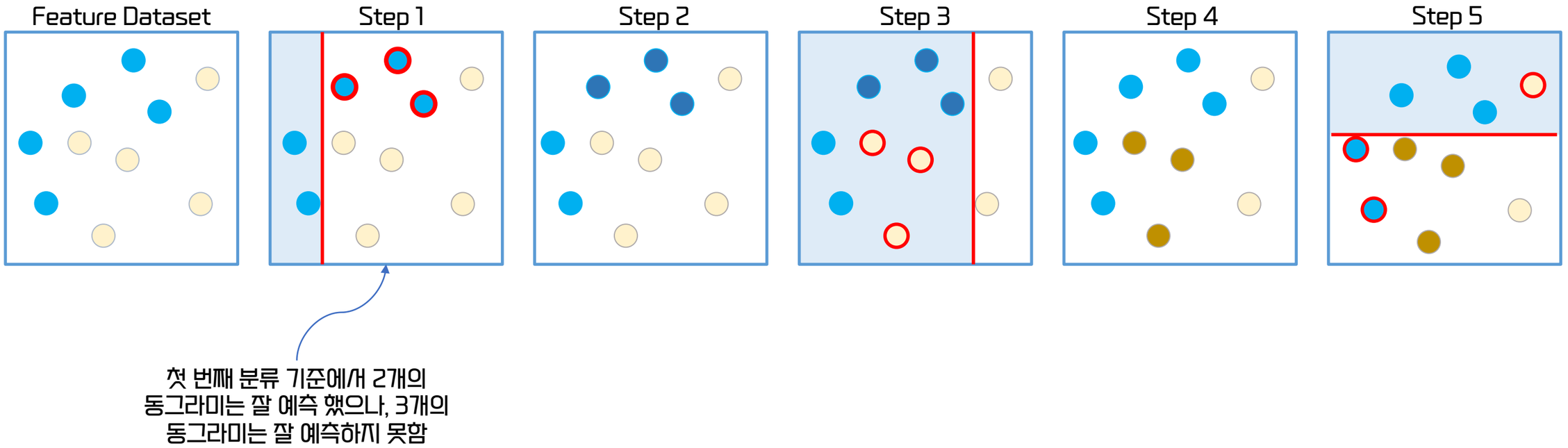
- 부스팅 알고리즘은 여러 개의 약한 학습기(Weak Learner)를 순차적으로 학습-예측한 데이터나 학습 트리에 가중치 부여를 통해 오류를 개선해 나가면서 학습하는 방식

- 부스팅의 대표적인 구현은 AdaBoost(Adaptive Boosting)과 그라디언트 부스트가 있음

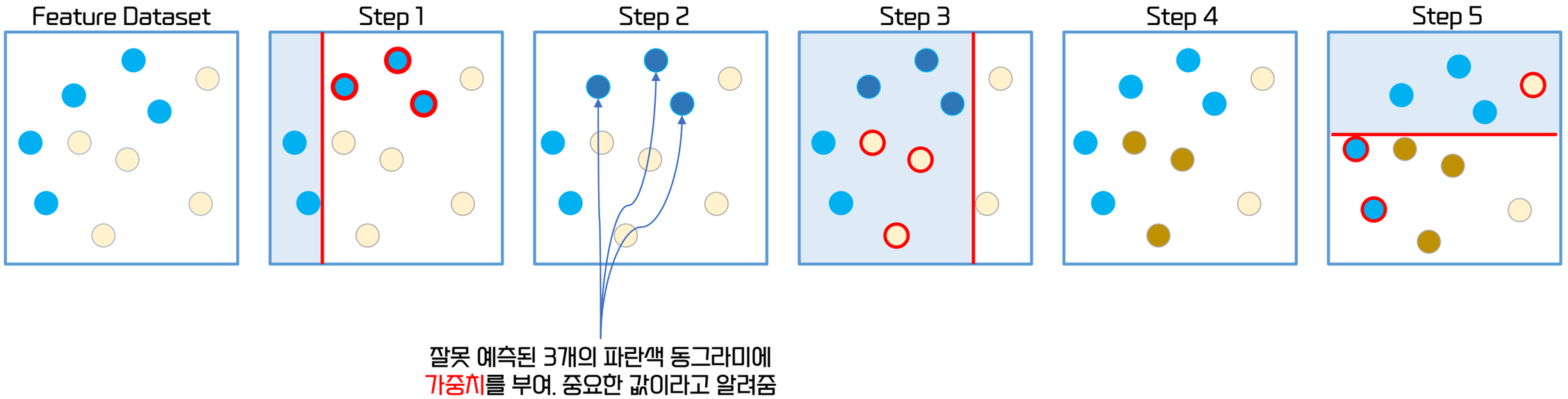
에이다 부스팅의 학습/예측 프로세스



에이다 부스팅의 학습/예측 프로세스

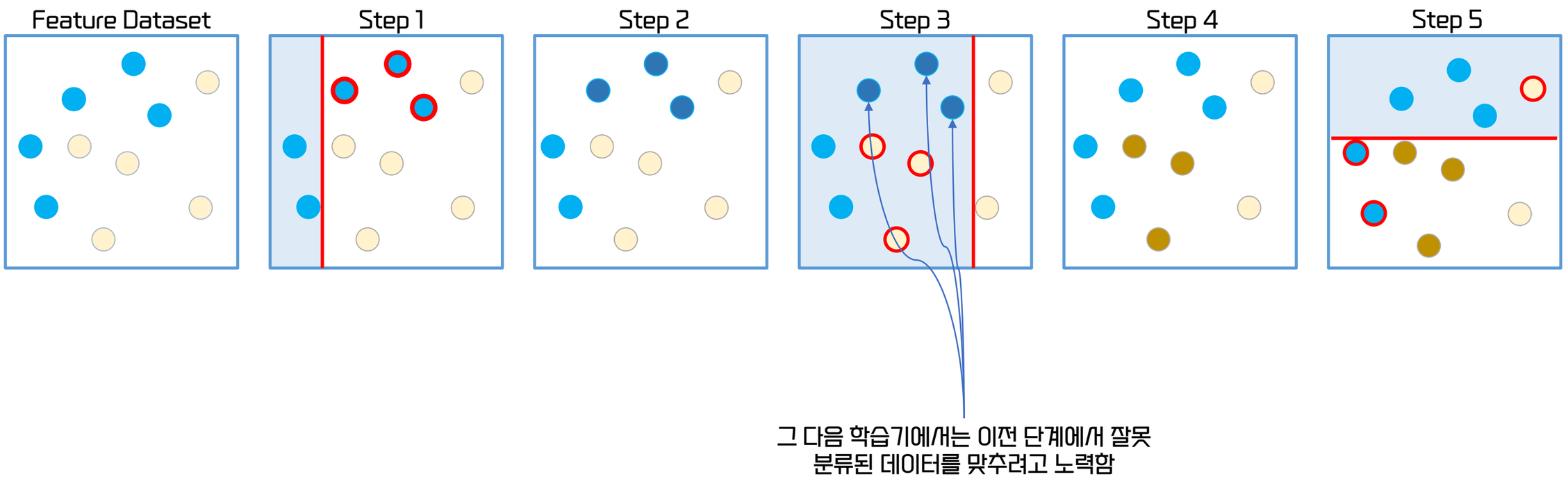


에이다 부스팅의 학습/예측 프로세스

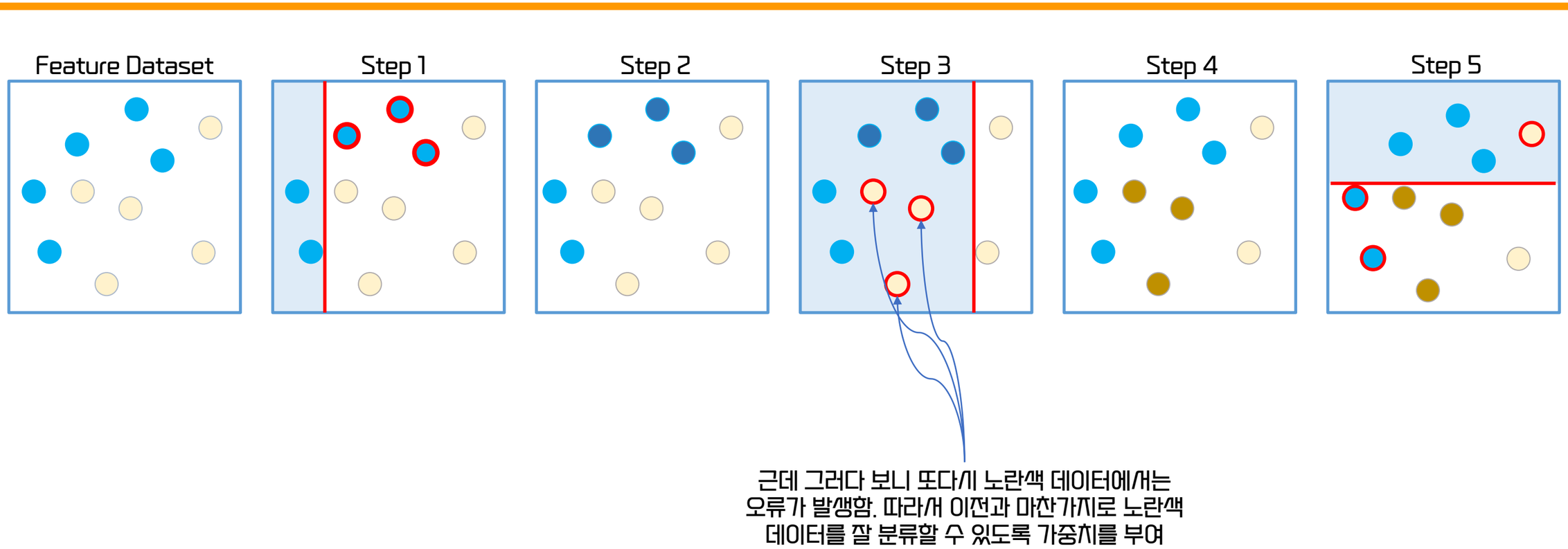


잘못 예측된 3개의 파란색 동그라미에
가중치를 부여. 중요한 값이라고 알려줌

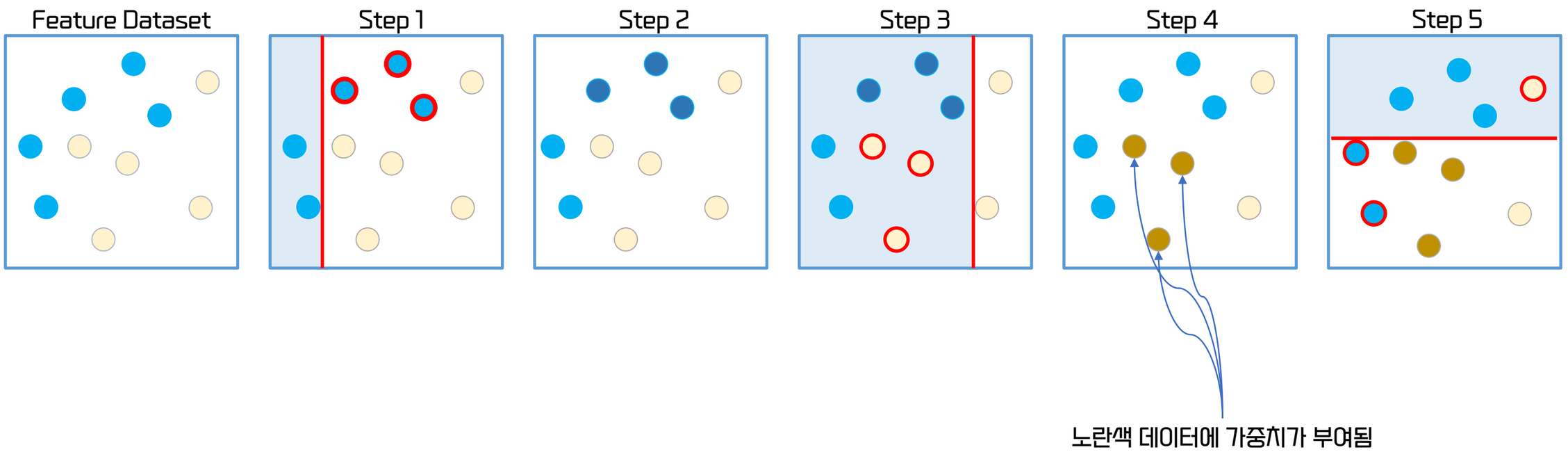
에이다 부스팅의 학습/예측 프로세스



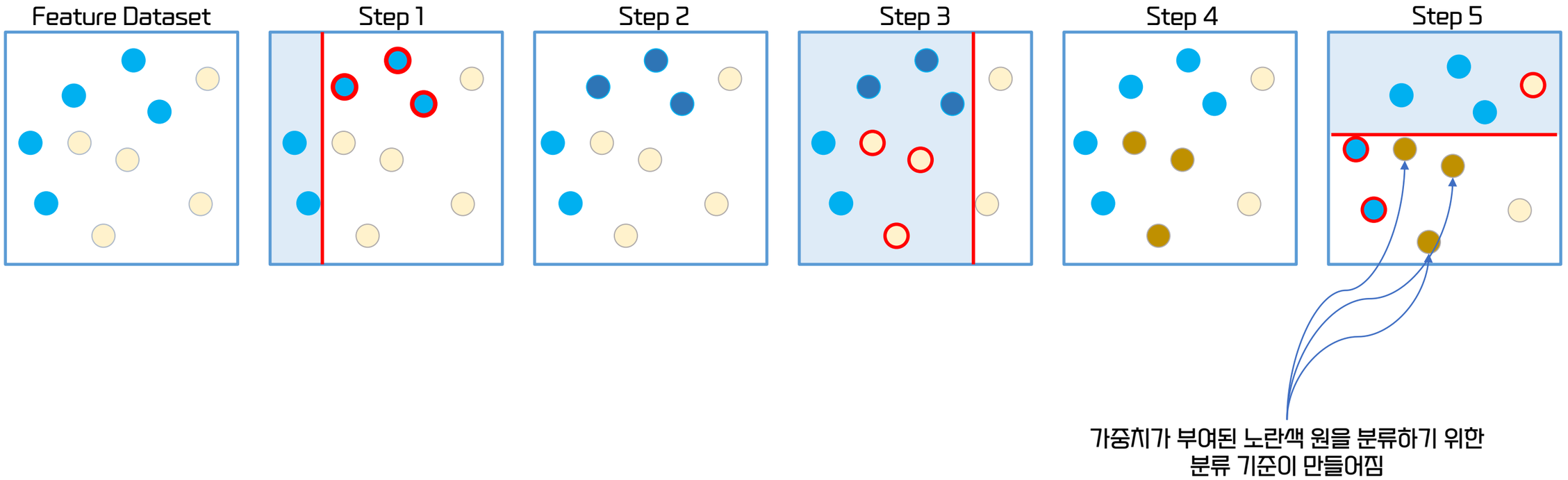
에이다 부스팅의 학습/예측 프로세스



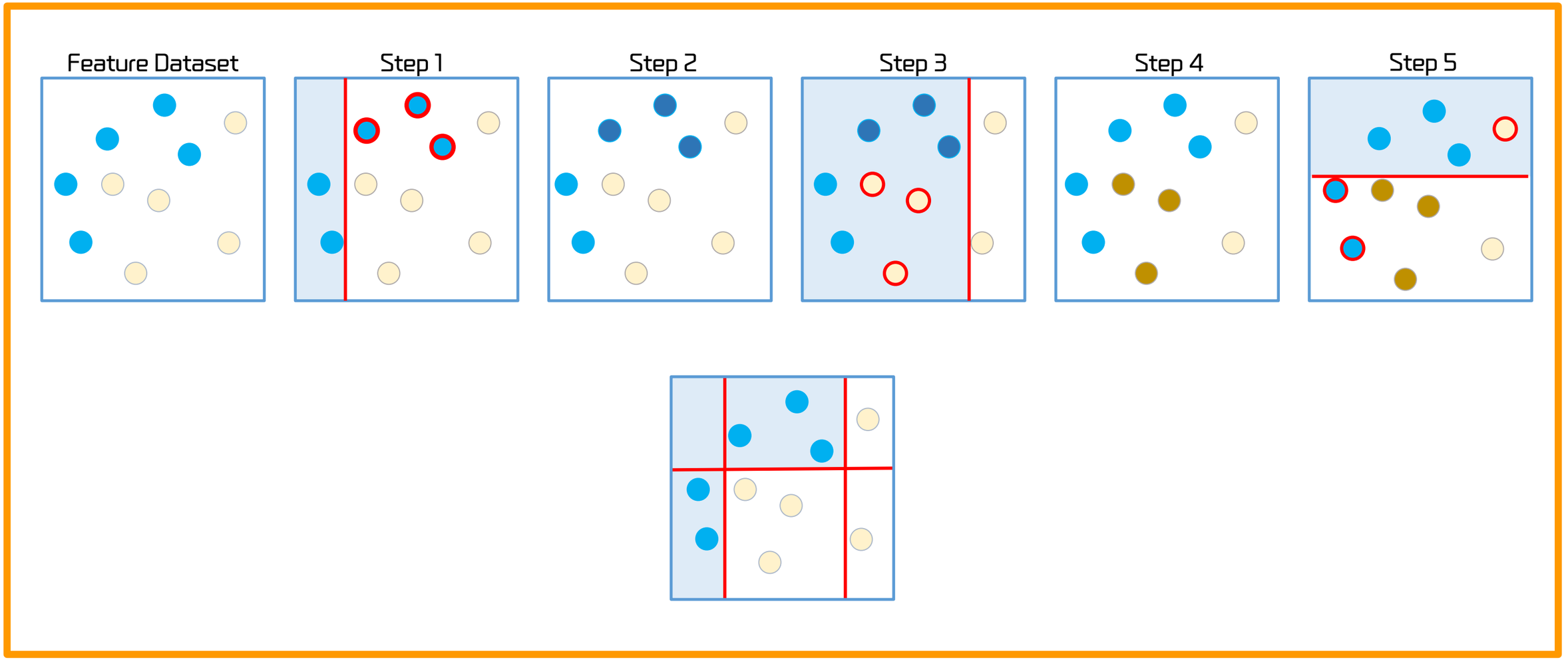
에이다 부스팅의 학습/예측 프로세스



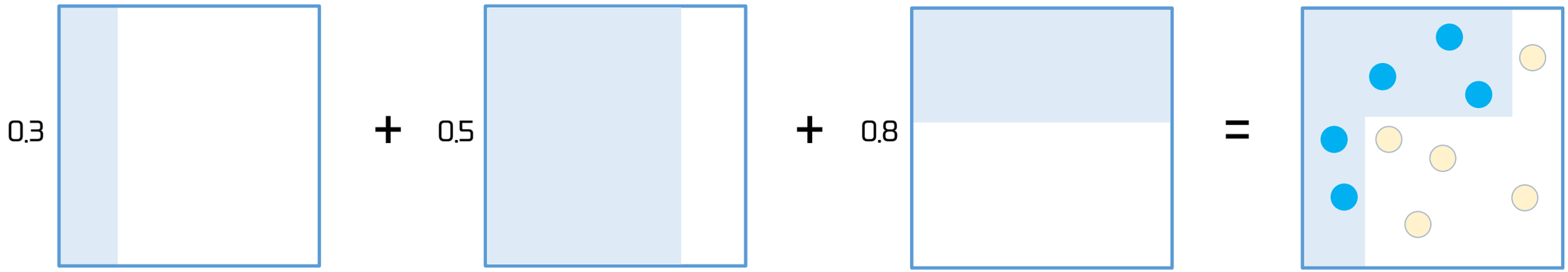
에이다 부스팅의 학습/예측 프로세스



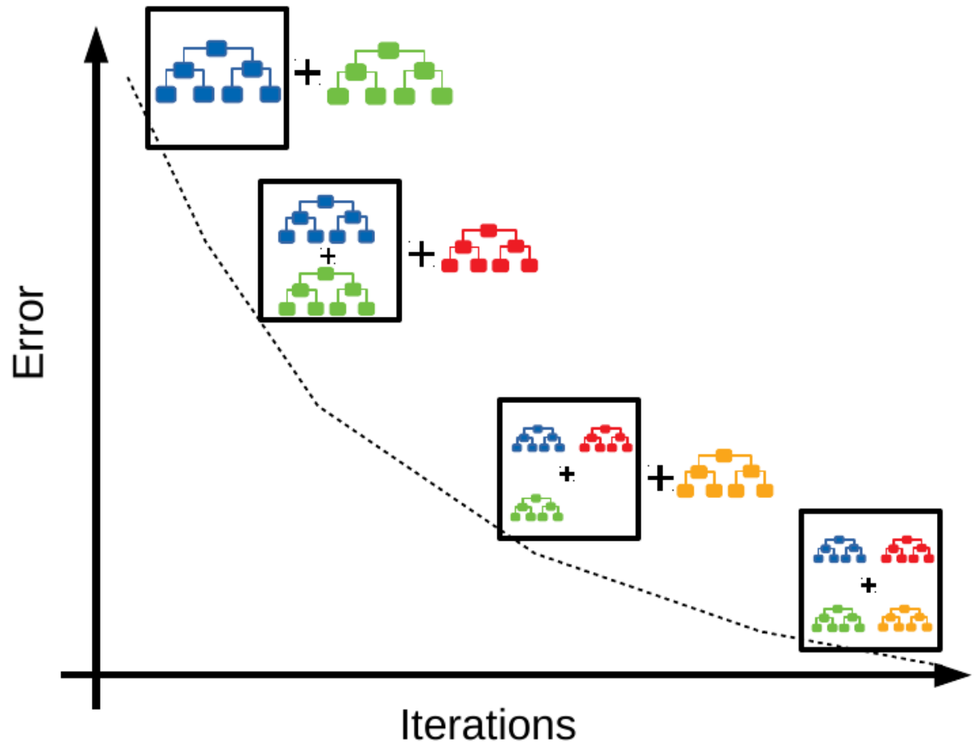
에이다 부스팅의 학습/예측 프로세스



에이다 부스팅의 학습/예측 프로세스



GBM(Gradient Boost Machine) 개요



- GBM도 AdaBoost와 유사하나, 가중치 업데이터를 경사 하강법을 이용함
- feature(x)를 입력했을 때 모델의 예측 함수를 F(x), 실제 타깃값을 y라고 한다면 오류식 h(x) = y – F(x)가 된다. 이 h(x)를 최소화 하는 방향성을 가지고 반복적으로 가중치 값을 갱신하는 것이 경사하강법이다.
- 경사 하강법은 반복 수행을 통해 오류를 최소화 할 수 있도록 가중치의 업데이트 값을 도출하는 기법으로서 머신러닝에서 매우 중요한 기법 중 하나이다.

사이킷런 GBM 주요 하이퍼 파라미터 및 튜닝

- loss : 경사 하강법에서 사용할 손실 함수 지정. 기본은 deviance

- learning_rate : GBM이 학습을 진행할 때 마다 적용하는 학습률. Weak Learner가 순차적으로 오류 값을 보정해 나가는데 적용하는 계수로서, 0 ~ 1 사이의 값을 지정한다. 너무 작은 값을 적용하면 업데이트 되는 값이 작아져서 최소 오류 값을 찾아 예측 성능이 높아질 가능성이 높으나, 많은 Weak Learner는 순차적인 반복이 필요해서 수행 시간이 오래 걸리고, 또 너무 작게 설정하면 모든 Weak Learner의 반복이 완료되어도 최소 오류 값을 찾지 못할 가능성이 생긴다. 반대로 너무 큰 값을 적용하면 최소 오류 값을 찾지 못하고 그냥 지나쳐 버려 예측 성능이 떨어질 가능성이 높아지지만 빠른 수행이 가능하다.

사이킷런 GBM 주요 하이퍼 파라미터 및 튜닝

- n_estimators : Weak Learner의 개수. Weak Learner가 순차적으로 오류를 보정하므로 개수가 많을 때는 예측 성능이 일정 수준까지는 좋아질 수는 있으나 개수가 많을수록 수행 시간이 오래 걸림. 기본값은 100

- subsample : Weak Learner가 학습에 사용하는 데이터의 샘플링 비율로서 기본값은 1이다. 기본적으로 전체 학습 데이터를 기반으로 학습하는데, 과적합이 염려되는 경우 subsample을 1보다 작은 값으로 설정하면 된다.

실습

GBM 기반의 사용자 행동 인식 예측 모델 만들기

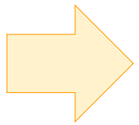
XGBoost 개요

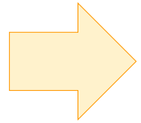
- XGBoost(eXtra Gradient Boost)
- 주요 장점
  - 뛰어난 예측 성능
  - GBM 대비 빠른 수행 시간. CPU 병렬 처리, GPU 지원
  - 다양한 성능 향상 기능
    - 규제(Regularization) 기능 탑재
    - Tree Pruning (가지치기)
  - 다양한 편의 기능
    - 조기 중단(Early Stopping)
    - 자체 내장된 교차 검증
    - 결손값 자체 처리

XGBoost 조기 중단 기능(Early Stopping)

- XGBoost는 특정 반복 횟수 만큼 더 이상 손실함수가 감소하지 않으면 지정된 반복횟수를 다 완료하지 않고 수행을 종료할 수 있다.
- 학습을 위한 시간을 단축시킨다. 특히 최적화 튜닝 단계에서 적절하게 사용이 가능하다.
- 너무 반복 횟수를 단축할 경우 예측 성능 최적화가 안된 상태에서 학습이 종료될 수 있으므로 유의해야 한다.
- 조기중단 설정 파라미터
  - early_stopping_rounds : 더 이상 손실 평가 지표가 감소하지 않는 최대 반복 횟수
  - eval_metric : 반복 수행 시 사용하는 비용 평가 지표
  - eval_set : 평가를 수행하는 별도의 검증 데이터 세트 설정. 일반적으로 검증 데이터 세트에서 반복적으로 손실 감소 성능을 평가

XGBoost의 파이썬 구현





C/C++ Native Module

- XGBoost는 최초 C / C++로 작성됨

Python Wrapper

- C/C++ 모듈을 호출하는 파이썬 Wrapper

Scikit Learn Wrapper

- 사이킷런 프레임과 통합 될 수 있는 파이썬 Wrapper Class 지원
  - ✓ XGBClassifier
  - ✓ XGBRegressor
- 학습과 예측을 다른 사이킷런 API와 동일하게 fit()과 predict()로 수행
- GridSearchCV와 같은 다른 사이킷런 모듈과 같이 사용 가능

XGBoost Python Wrapper와 Scikit Learn Wrapper API 비교

항목	파이썬 Wrapper	사이킷런 Wrapper
사용 모듈	from xgboost as xgb	from xgboost import XGBClassifier
학습용과 테스트용 데이터 세트	DMatrix 객체를 별도 생성 train = xgb.DMatrix(data=X_train, label=y_train) DMatrix 생성자로 피처 데이터 세트와 레이블 데이터 세트를 입력	Numpy 또는 Pandas 사용
학습 API	xgb_model = xgb.train() xgb_model은 학습된 객체를 반환 받음	XGBClassifier.fit()
예측 API	xgb.train()으로 학습된 객체에서 predict() 호출. 즉 xgb_model.predict() 이때 반환 결과는 예측 결과가 아닌 예측 결과를 추정하는 확률값	XGBClassifier.predict() 예측 결과값 반환
피처 중요도 시각화	plot_importance() 함수 이용	plot_importance() 함수 이용

XGBoost Python Wrapper와 Scikit Learn Wrapper 하이퍼 파라미터 비교

파이썬 Wrapper	사이킷런 Wrapper	하이퍼 파라미터 설명
eta	learning_rate	GBM의 학습률과 같은 파라미터. 파이썬 래퍼 기반의 xgboost를 사용할 때의 기본값은 0.3, 사이킷런 래퍼 클래스를 사용할 때는 0.1
num_boost_rounds	n_estimators	사이킷런 앙상블의 n_estimators와 동일. 약한 학습기의 개수(반복 수행 횟수)
min_child_weight	min_child_weight	결정트리의 min_child_leaf와 유사.
max_depth	max_depth	결정트리의 max_depth와 동일. 트리의 최대 깊이
sub_sample	subsample	GBM의 subsample과 동일. 트리가 커져서 과적합 되는 것을 방지하기 위해 데이터를 샘플링할 비율을 지정. 일반적으로 0.5 ~ 1 사이의 값을 사용

XGBoost Python Wrapper와 Scikit Learn Wrapper 하이퍼 파라미터 비교

파이썬 Wrapper	사이킷런 Wrapper	하이퍼 파라미터 설명
lambda	reg_lambda	L2 규제(regularization) 적용 값. 기본값은 1로서 값이 클 수록 규제 값이 커진다. 과적합 제어
alpha	reg_alpha	L1 규제(regularization) 적용 값. 기본값은 0으로서 값이 클 수록 규제 값이 커진다. 과적합 제어
colsample_bytree	colsample_bytree	GBM의 max_features와 유사함. 트리 생성에 필요한 피처를 임의로 샘플링 하는 데 사용된다. 매우 많은 피처가 있는 경우 과적합을 조정하는 데 적용한다.
scale_pos_weight	scale_pos_weight	특정 값으로 치우친 비대칭한 클래스로 구성된 데이터 세트의 균형을 유지하기 위한 파라미터. 기본값은 1
gamma	gamma	트리의 리프 노드를 추가적으로 나눌지를 결정하는 최소 손실 감소 값. 해당 값보다 큰 손실이 감소된 경우에 리프 노드를 분리. 값이 커질수록 과적합 감소 효과가 있음.

실습

XGBoost를 이용한 유방암 데이터 세트 예측

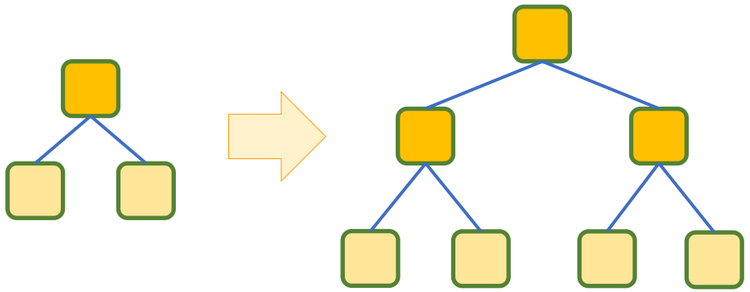
LightGBM 개요

- XGBoost 대비 장점

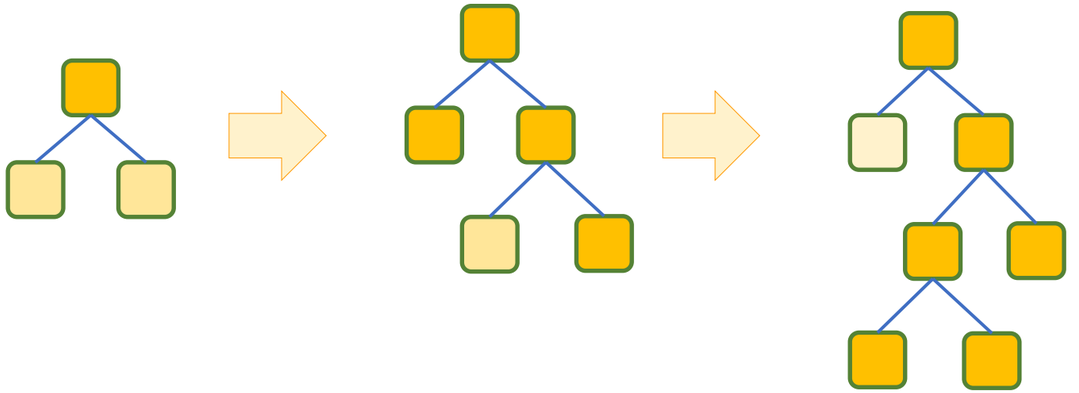
✓ 더 빠른 학습과 예측 수행 시간
✓ 더 작은 메모리 사용량
✓ 카테고리 피처의 자동 변환과 최적 분할(One Hot Encoding을 사용하지 않고도 카테고리형 피처를 최적으로 변환하고 이에 따른 분할 노드 수행)

LightGBM 작동 방식

균형 트리 분할(Level Wise)

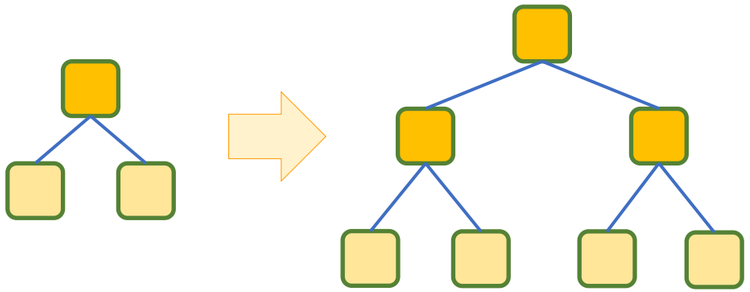


리프 중심 트리 분할(Leaf Wise)



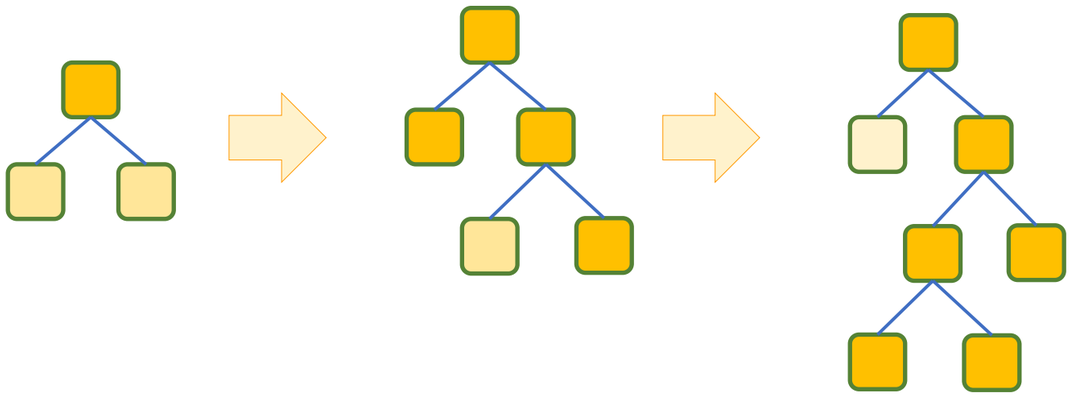
LightGBM 작동 방식

균형 트리 분할(Level Wise)



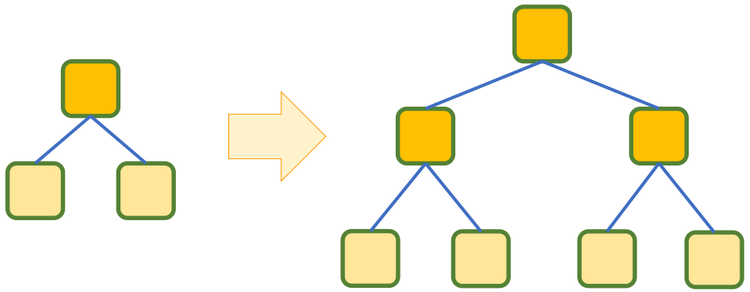
XGBoost를 포함한 일반적인 GBM 방식. 균형 잡힌 트리를 만들어 Depth를 최소화 할 수 있다. 한쪽으로만 트리가 뻗어 나가게 되면 과대적합 될 수 있다! 라는 것이 이론적 근거가 있다.

리프 중심 트리 분할(Leaf Wise)

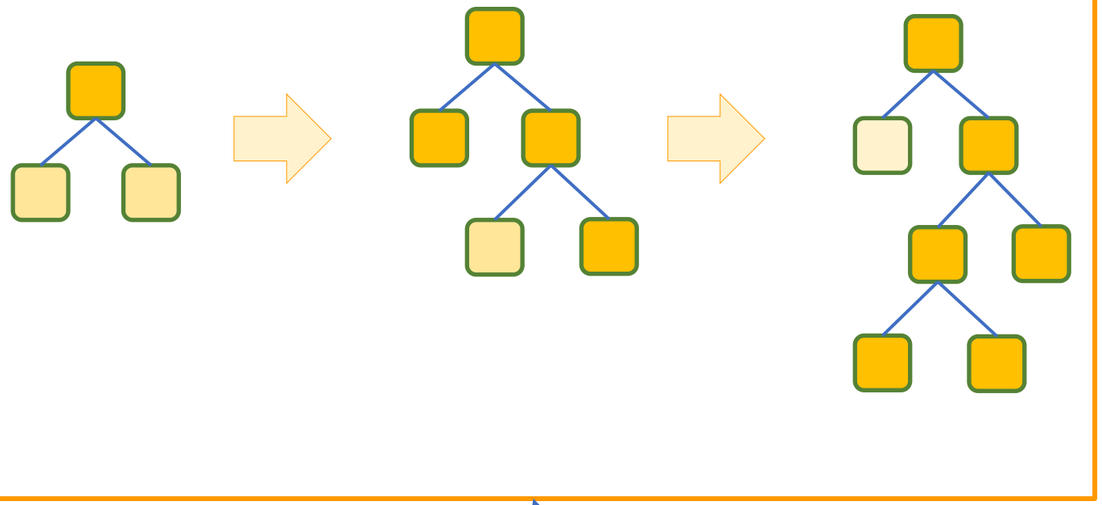


LightGBM 작동 방식

균형 트리 분할(Level Wise)



리프 중심 트리 분할(Leaf Wise)



한쪽 방향으로 예측을 했을 때 예측 오류를 줄여줄 수 있으면 그 노드 쪽으로 계속 리프 노드를 생성하면 조금 더 정확하겠다고 판단

LightGBM의 파이썬 구현

C/C++ Native Module

- 원래는 Windows 기반의 C/C++
- 현재는 Linux 등 여러 OS 지원

Python Wrapper

- C/C++ 모듈을 호출하는 파이썬 Wrapper

Scikit Learn Wrapper

- 사이킷런 프레임과 통합 될 수 있는 파이썬 Wrapper Class 지원
  - ✓ LGBMClassifier
  - ✓ LGBMRegressor
- 학습과 예측을 다른 사이킷런 API와 동일하게 fit()과 predict()로 수행
- GridSearchCV와 같은 다른 사이킷런 모듈과 같이 사용 가능

LightGBM Python Wrapper와 Scikit Learn Wrapper 하이퍼 파라미터 비교

파이썬 Wrapper	사이킷런 Wrapper	하이퍼 파라미터 설명
num_iterations	num_eatimators	약한 학습기의 개수(반복 수행 회수)
learning_rate	learning_rate	학습률(learning rate). 0 에서 1 사이의 값을 지정하며 부스팅 스텝을 반복적으로 수행할 때 업데이트 되는 학습률 값
max_depth	max_depth	결정트리의 max_depth와 동일. 트리의 최대 깊이
min_data_in_leaf	min_child_samples	리프 노드가 될 수 있는 최소 데이터 건수(Sample 수)
bagging_fraction	subsample	트리가 커져서 과적합되는 것을 제엉가 l 위해 데이터를 샘플링 하는 비율을 지정. sub_sample=0.5로 지정하면 전체 데이터의 절반을 트리를 생성하는데 사용합니다.
feature_fraction	colsample_bytree	GBM의 max_features와 유사하다. 트리 생성에 필요한 피처를 임의로 샘플링하는데 사용된다. 매우 많은 피처가 있는 경우 과적합을 조정하는데 적용한다.

LightGBM Python Wrapper와 Scikit Learn Wrapper 하이퍼 파라미터 비교

파이썬 Wrapper	사이킷런 Wrapper	하이퍼 파라미터 설명
lambda	reg_lambda	L2 규제(regularization) 적용 값. 기본값은 1로서 값이 클 수록 규제 값이 커진다. 과적합 제어
alpha	reg_alpha	L1 규제(regularization) 적용 값. 기본값은 0으로서 값이 클 수록 규제 값이 커진다. 과적합 제어
early_stopping_round	early_stopping_rounds	학습 조기 종료를 위한 early stopping interval 값
num_leaves	num_leaves	최대 리프 노드 개수
min_sum_hessian_in_leaf	min_child_weight	결정트리의 min_child_leaf와 유사. 과적합 조절용

num_leaves의 개수를 중심으로 min_child_samples(min_data_in_leaf), max_depth를 함께 조정하면서 모델의 복잡도를 줄이는 것이 기본 튜닝 방안

최종 실습

캐글 산탄데르 은행 고객 만족 예측

최종 실습

신용카드 사기 예측 실습

스태킹(Stacking) 모델

내용을 입력하세요

- 스태킹이란?

실습

스태킹 모델 실습하기

Feature Selection

내용을 입력하세요

- 스태킹이란?

Hyperparameter Tuning

Grid Search의 문제점 이해하기

- 스태킹이란?

베이지안 최적화 개요

- 스태킹이란?

HyperOpt 패키지 사용하기

- 스태킹이란?