분류

4️⃣ 분류

4.5 부스팅 알고리즘

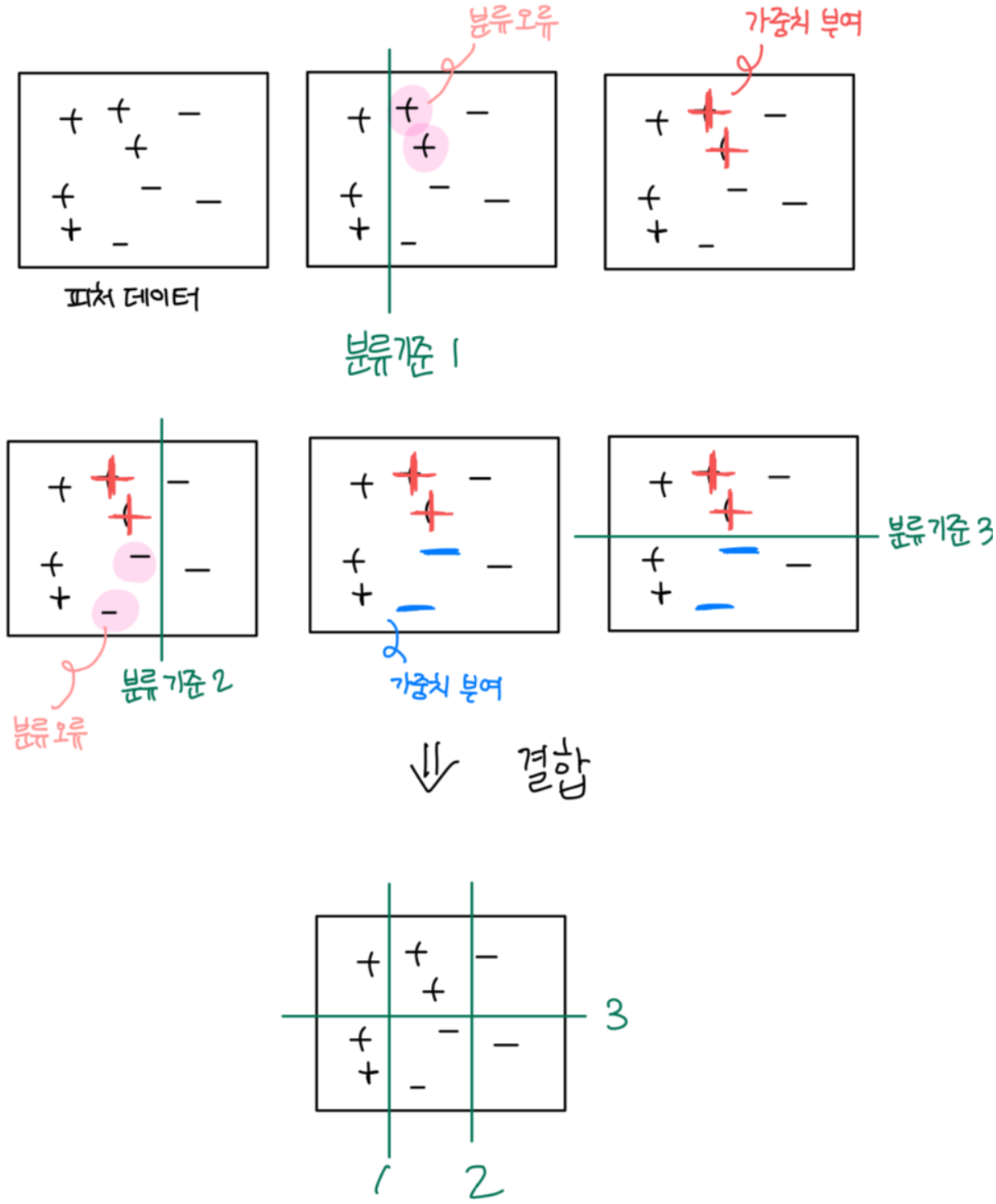
- 개념

여러 개의 약한 학습기 (weak learner) 를 순차적으로 학습, 예측하면서 잘못 예측한 데이터에 가중치 부여를 통해 오류를 개선해 나가는 학습 방식

- 종류
  - AdaBoost
  - 그래디언트 부스트

4.5.1 AdaBoost (에이다 부스트)

- 오류 데이터에 가중치를 부여하면서 부스팅을 수행하는 알고리즘
- 개별적인 약한 학습기 각각에 가중치를 부여하여 결합



4.5.2 GBM (Gradient Boosting Machine)

🗂️ 개요

- 반복 수행을 통해 오류를 최소화할 수 있도록 가중치의 업데이트 값을 도출하는 기법
- 가중치 업데이트를 경사 하강법 (Gradient Descent) 이용

$$\text{오류식} : h(x) = y - F(x)$$

  - y: 실제 결과값
  - x1,x2,... : feature
  - F(x): feature에 기반한 예측 함수
- 사이킷에서 `GradientBoostingClassifier` 클래스 제공

🗂️ GBM 하이퍼 파라미터 및 튜닝

파라미터	의미
loss	경사 하강법에서 사용할 비용 함수
learning_rate	weak learner 가 순차적으로 오류 값을 보정해 나가는 데 적용하는 계수
n_estimator	weak learner의 개수
subsample	weak learner가 학습에 사용하는 데이터의 샘플링 비율

4.5.3 XGBoost (eXtra Gradient Boost)

🗂️ 개요

- 트리 기반의 앙상블 학습에 이용
- GBM에 기반하지만 GBM의 단덤인 느린 수행 시간 및 과적합 규제 부재 등의 문제를 해결
- 초기: 사이킷런과 연동되지 않는 **파이썬 래퍼 XGBoost 모듈**
- 후기: 사이킷런과 연동되는 **사이킷런 래퍼 XGBoost 모듈**

🗂️ 파이썬 래퍼 XGBoost 하이퍼 파라미터

- 조기 중간, 과적합 규제 위한 하이퍼 파라미터 추가됨

1. **일반 파라미터**

   : 일반적으로 실행 시 스레드의 개수나 silent 모드 등의 선택

   - `booster` - gbtree (tree based model) 또는 gblinear (linear model) 선택
   - `silent` - 출력 메시지를 나타내고 싶지 않은 경우
   - `nthread` - CPU의 실행 스레드 개수

2. **부스터 파라미터**

   : 트리 최적화, 부스팅, regularization 등과 관련

   - `eta` - GBM의 학습률 (learning rate) 와 같은 역할. 부스팅 스텝을 반복적으로 수행할 때 업데이트 된다.
   - `gamma` - 트리의 리프 노드를 추가적으로 나눌지를 결정할 최소 손실 감소 값. 해당 값보다 큰 손실이 감소된 경우에 리프노드를 분리

3. **학습 태스크 파라미터**

   : 학습 수행 시의 객체 함수, 평가를 위한 지표 등을 설정

   - `objective` - 최솟값을 가져야할 손실 함수
   - `eval_metric` - 검증에 사용되는 함수를 정의 (회귀: rmse , 분류 : error)

📦 사이킷런 래퍼 XGBoost

- 분류 : XGBClassifier
- 회귀: XGBRegressor

eta → learning_rate
sub_sample → subsample

lambda → reg_lambda
alpha → reg_alpha

4.6 LightGBM

🗂️ 개요

- 학습에 걸리는 시간과 메모리 사용량이 적다.
- 리프 중심 트리 분할 (Leaf Wise) 방식 사용
  - 트리의 균형에 맞추지 않고 최대 손실 값을 가지는 리프 노드를 지속적으로 분할하면서 트리의 깊이가 깊어지고 비대칭적인 규칙 트리 생성
  - 학습을 반복할수록 예측 오류 손실 최소화

🗂️ LightGBM 하이퍼 파라미터

1. **주요 파라미터**
   - `num_iterations` - 반복 수행하려는 트리의 개수
   - `learning_rate` - 0에서 1사이의 값을 지정하며 부스팅 스텝을 반복적으로 수행할 때 업데이트되는 학습률

2. **Learning Task 파라미터**
   - `objective` - 최솟값을 가져야할 손실 함수

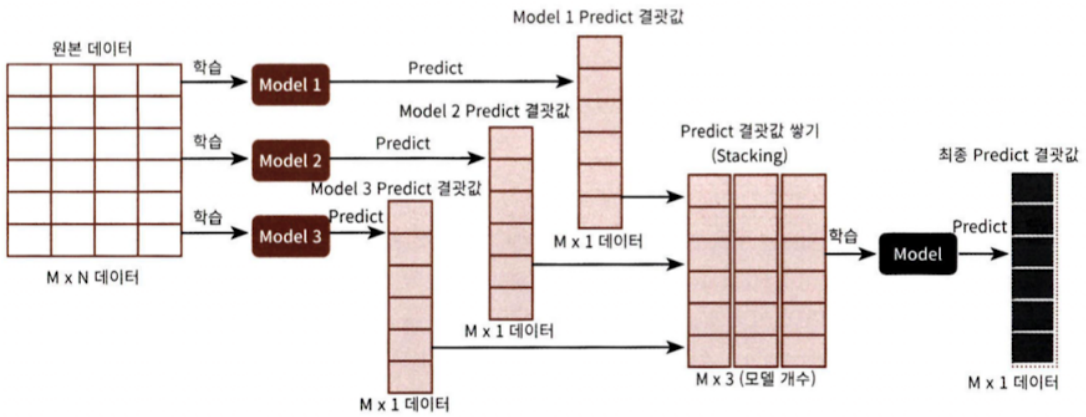
🗂️ 튜닝 방안

- `num_leaves`
  - 개별 트리가 가질 수 있는 최대 리프의 개수

    - 개수를 높이면 정확도가 높아지지만, 반대로 트리의 깊이가 깊어지고 복잡도가 커져 과적합 영향도가 커진다.
- `min_data_in_leaf` → `min_child_samples`
    - 큰 값으로 설정하면 트리가 깊어지는 것을 방지
- `max_depth`
    - 크기의 깊이 제한

4.7 스태킹 앙상블

- 개별 알고리즘의 예측 결과 데이터 세트를 최종적인 메타 데이터 세트로 만들어 별도의 ML 알고리즘으로 최종 학습을 수행하고 테스트 데이터를 기반으로 다시 최종 예측을 수행하는 방식
- 종류
    1. 개별적인 기반 모델
    2. 개별 기반 모델의 예측 데이터를 학습 데이터로 만들어서 학습하는 최종 메타 모델



4.8 하이퍼 파라미터 튜닝

4.8.1 베이지안 최적화

- 목적 함수 식을 알 수 없는 형태의 함수에서 최대 또는 최소 함수 반환 값을 만드는 최적 입력값을 찾는 것

🗂️ 구성

- 대체 모델 (Surrogat Model)
  - 획득 함수로부터 최적 함수를 예측할 수 있는 입력값을 추천 받은 뒤 이를 기반으로 최적 함수 모델 개선
- 획득 함수 (Acquisition Function)
  - 개선도니 대체 모델을 기반으로 최적 입력값 계산

🗂️ 단계

1. 랜덤하게 파라미터들을 샘플링
2. 관측도니 값을 기반으로 대체 모델은 최적 함수를 추정
3. 추정도니 최적 함수를 기반으로 획득함수는 다음으로 관측할 하이퍼 파라미터를 대체 모델에 전달
4. 획득 함수로부터 전달된 하이퍼 파라미터를 수행하여 관측된 값을 기반으로 대체 모델 갱신

3,4 단계를 반복하면 불확실성이 개선되고 정확한 최적 함수 추정이 가능하게 된다.

4.8.2 HyperOpt

- 베이지안 최적화를 머신러닝 모델의 하이퍼 파라미터 튜닝에 적용할 수 있게 제공되는 파이썬 패키지 중 하나

🗂️ 단계

1. 입력 변수명과 입력값의 검색 공간 설정
2. 목적함수 생성
3. 목적 함수의 반환값이 최소가 되는 최적의 입력값을 찾기