AI PROGRAMMING WITH PYTHON

앙상블(Ensemble)

앙상블(ENSEMBLE)

◆ 앙상블 학습 / 방법

- **여러 모델 또는 동일한 모델 여러 개 결합하여 정확도 높은 모델을 만드는 학습 방법**
  - Estermotor : 학습 완료된 모델, 추정기

- **핵심원리 ➔ 모델(Estermotor 추정기)의 편향과 분산 줄이기**
  - 편향(Bias)
    - 예측값과 정답이 떨어져 있는 정도(오차 Error)
    - 편향이 크면 '과소적합'
  - 분산(Variance)
    - 예측값들의 흩어져있는 분포 정도

앙상블(ENSEMBLE)

◆ 앙상블 학습 / 방법

- **모델 및 샘플 결합 방법에 따른 분류**
  - ➢ 보팅(Voting 투표) 기반 앙상블
  - ➢ 배깅(Bagging, Bootstrap Aggregating) 기반 앙상블
  - ➢ 부스팅(Boosting)기반 앙상블

- **기법에 따른 분류**
  - ➢ 평균화 기법 : 배깅, 랜덤포레드스
  - ➢ 순차적 기법 : 에이다부스트, 그레디언트 부스팅

앙상블(ENSEMBLE)

◆ 앙상블 학습 / 방법

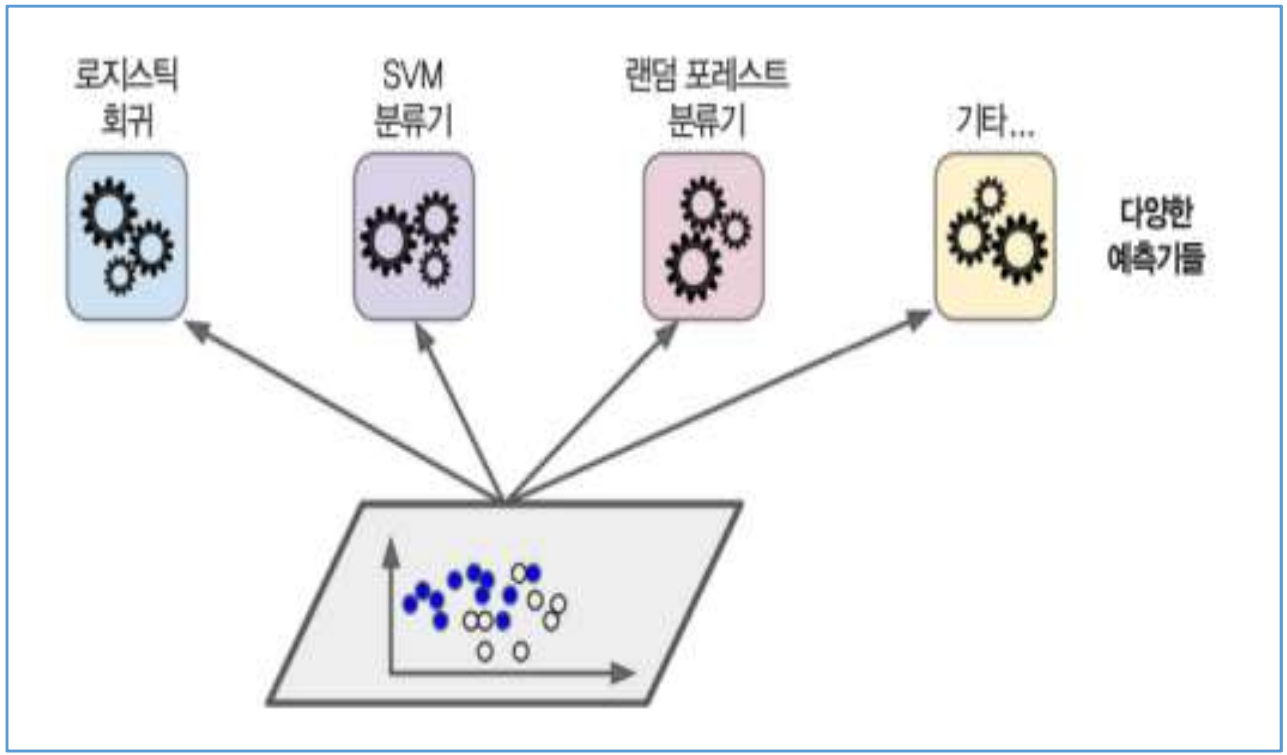
- **DecisionTree를 많이 사용하는 이유**
  - ➢ 가만히 두면 성능이 최대가 됨 즉 과대적합
  - ➢ 여러 개의 Tree를 사용해서 과대적합을 줄이는 방식으로 사용

앙상블(ENSEMBLE)

◆ 보팅(Voting)

매우 단순한 개념 및 형태이지만 강력한 성능을 가짐

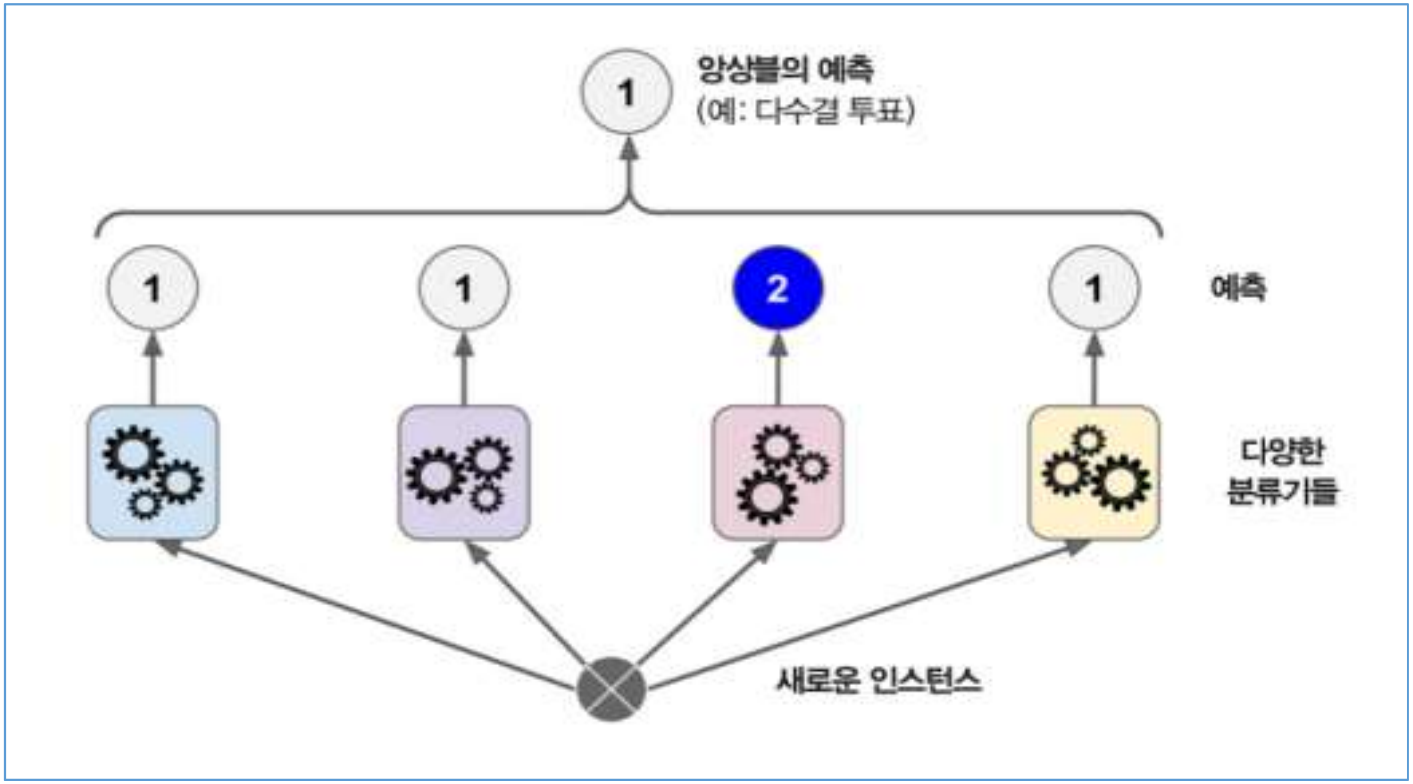
동일 훈련데이터로 여러 알고리즘의 여러 모델(추정기) 병렬 학습 진행



앙상블(ENSEMBLE)

◆ 보팅(Voting)

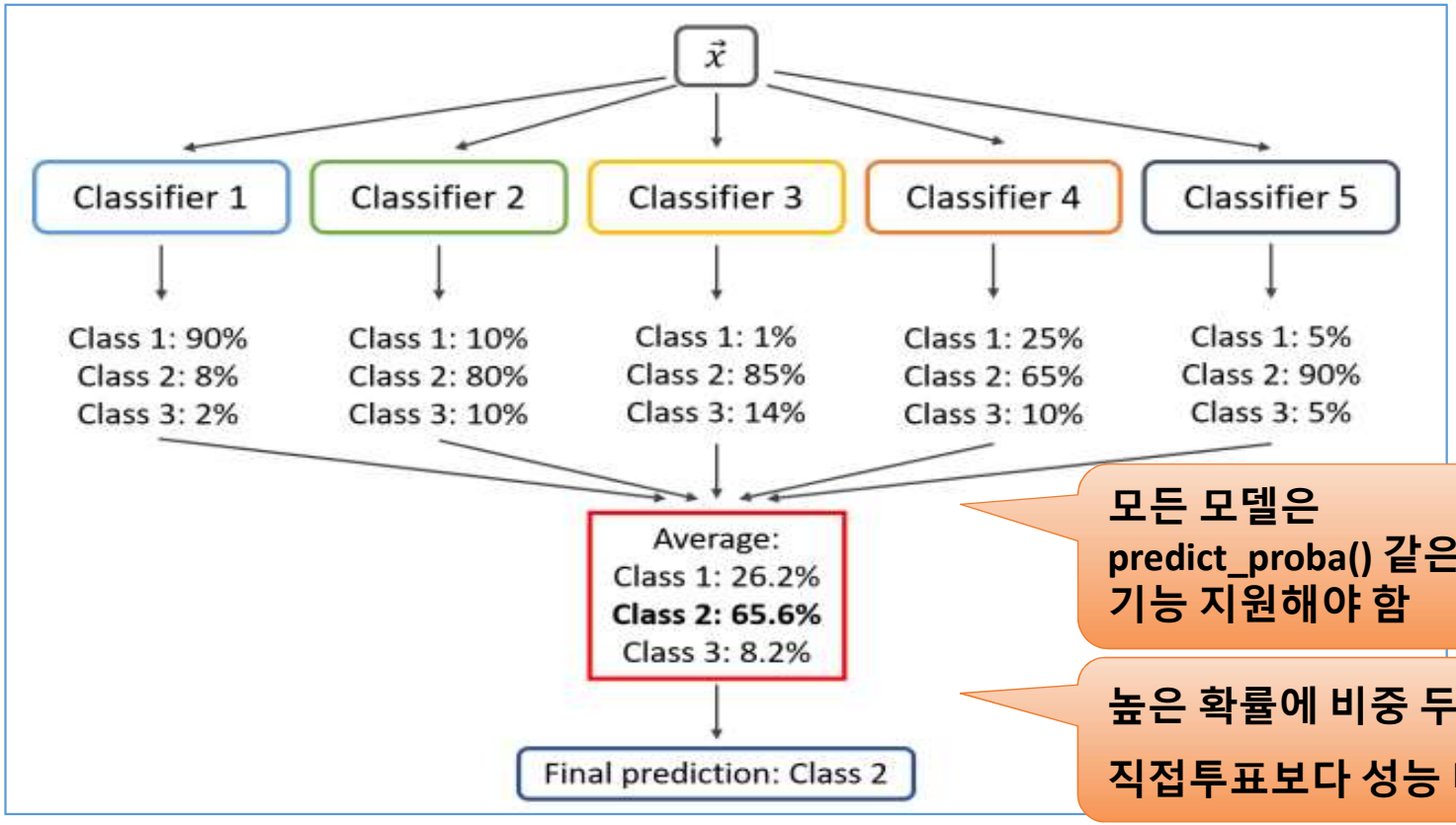
직접 투표 방식 ➔ 여러 모델의 예측값들의 다수로 결정



앙상블(ENSEMBLE)

◆ 보팅(Voting)

간접 투표 방식 ➔ 여러 모델 예측한 확률값들의 평균값으로 예측값 결정



모든 모델은 predict_proba() 같은 확률 예측 기능 지원해야 함

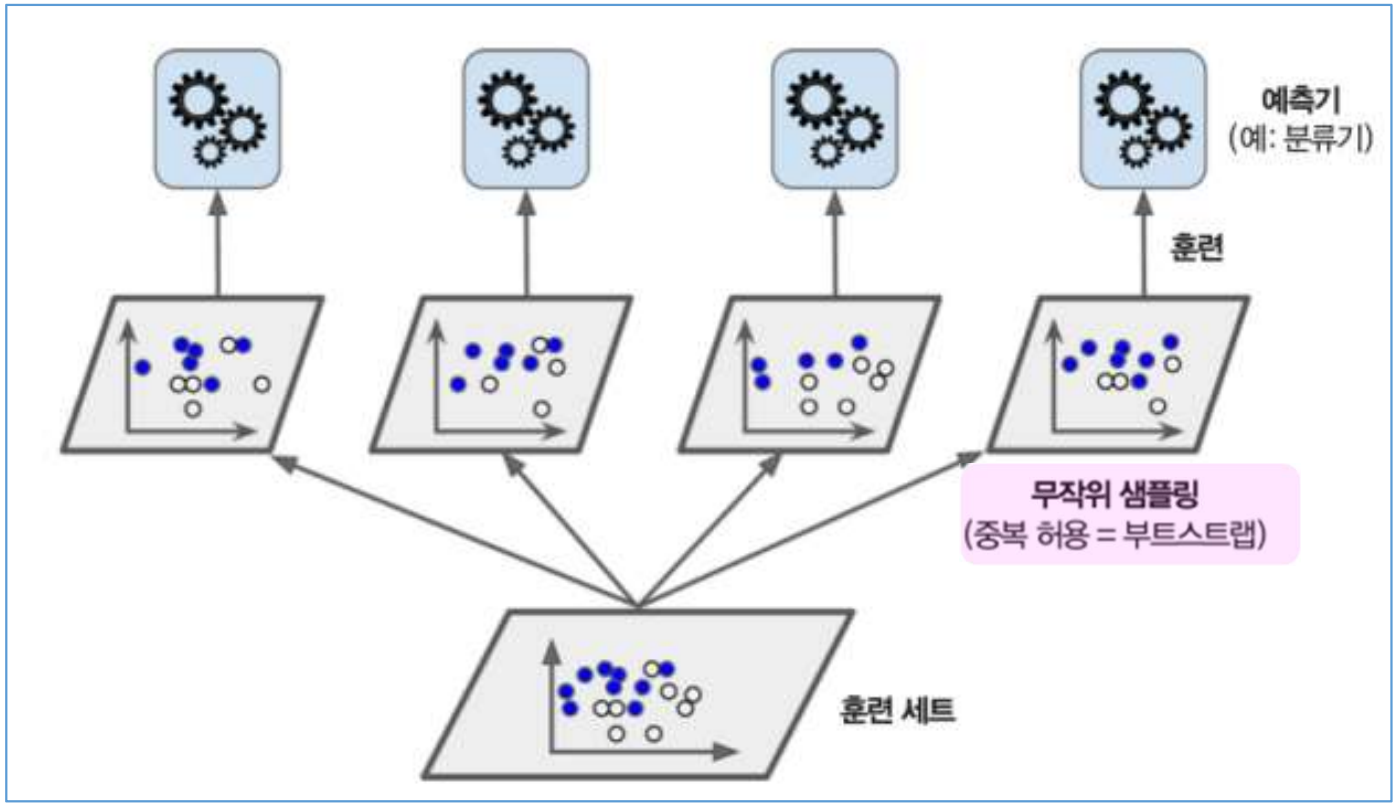
높은 확률에 비중 두어 직접투표보다 성능 더 좋음!

앙상블(ENSEMBLE)

◆ 배깅(Bagging, Bootstrap Aggregation)

중복 허용 샘플링하는 방식 의미하며 학습 데이터 부족 해결

중복 샘플링 다른 데이터와 동일 알고리즘의 여러 모델 병렬 학습 진행



앙상블(ENSEMBLE)

◆ 배깅(Bagging, Bootstrap Aggregation)

동작원리

- **중복 허용 랜덤** 샘플링 통한 **학습 샘플 데이터 생성**
- **동일 모델 여러 개 병렬 학습** 진행
- 학습 결과 결합
  - 분류 : 다중 투표
  - 회귀 : 평균

대표 알고리즘

- RandomForest => RandomForestClassifier
  RandomForestRegressor

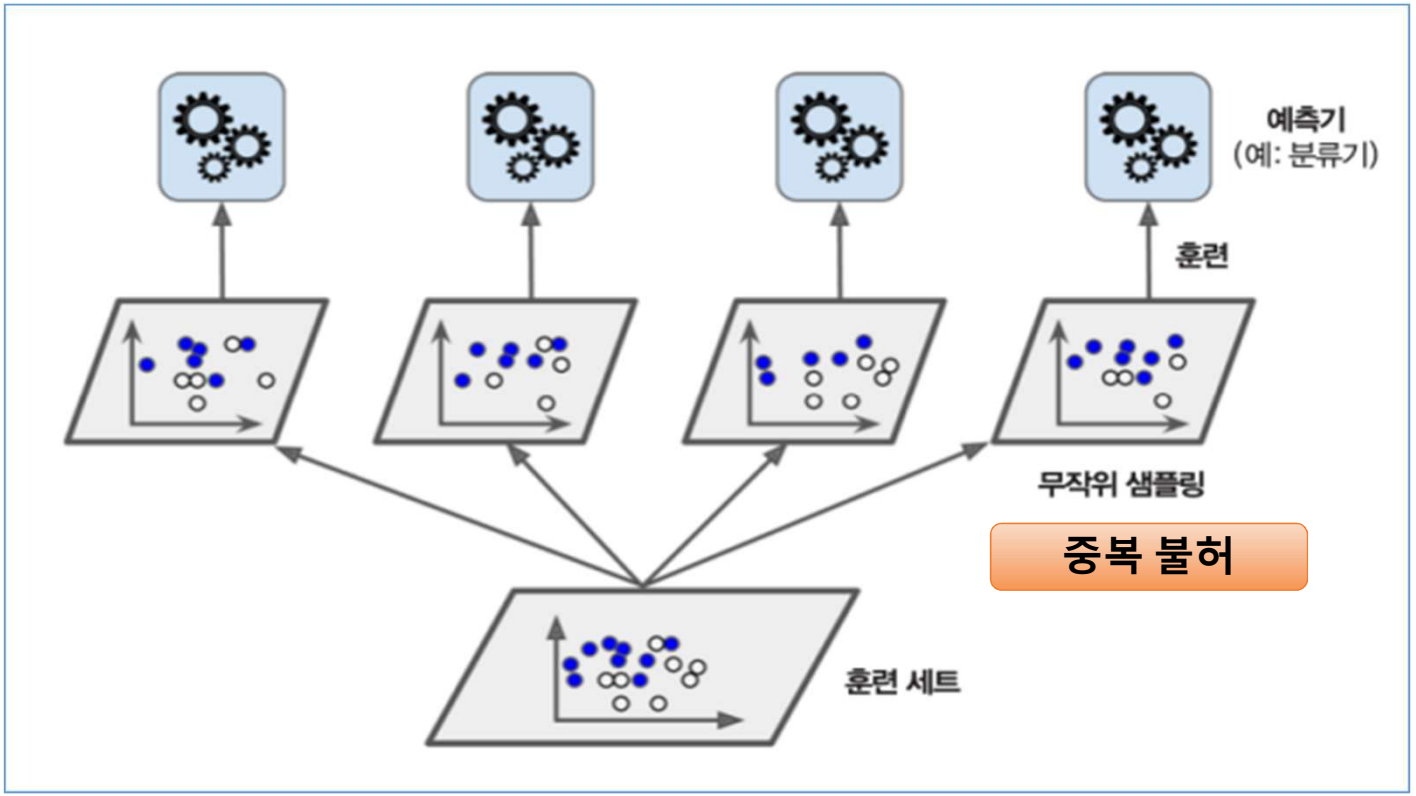
앙상블(ENSEMBLE)

◆ 페이스팅(Pasting)

중복 불허 샘플링하는 방식 의미하며 병렬 학습 진행

중복 불허 샘플링 다른 데이터와 동일 알고리즘의 여러 모델 병렬 학습

대표 알고리즘 : ExtraTrees



앙상블(ENSEMBLE)

◆ 랜덤포레스트(RandomForest)

- **앙상블 학습의 대표적인 방법 중 하나**
- 안정적인 성능으로 널리 사용
- 결정트리를 여러 개 랜덤하게 만들어 모델들의 예측값들을 사용
- 앙상블 알고리즘 중 비교적 빠른 수행 속도
- 다양한 영역에서 높은 예측 성능

- 분류, 회귀 모두 사용
  - ➢ 분류 : RandomForestClassifier
  - ➢ 회귀 : RandomForestRegressor

앙상블(ENSEMBLE)

◆ 랜덤포레스트(RandomForest)

『**동작 방식**』

① **중복 허용**한 훈련 데이터 **랜덤**하게 추출 ➔ 랜덤 샘플링

➔ **중복 허용** 샘플링, **부트스트랩 샘플**

➔ **매번 훈련 데이터 셋 다름**

➔ **중복 허용되어 성능 나쁨 : 결정트리 과적합 회피**

② 전체 특성에서 **무작위 특성 추출**

➔ **불순도 최소가 되는 특성 선택 ➔ 최적 노드 분할 적용**

➔ 분류 : 선택 특성 수 = 전체 특성 개수의 **제곱근 (특성 줄임)**

➔ 회귀 : 선택 특성 수 = 전체 특성

앙상블(ENSEMBLE)

◆ 랜덤포레스트(RandomForest)

『동작 방식』

③ **예측결과 결정**

➔ 분류 : 각 트리의 클래스별 확률 평균 **후 가장 높은 확률 가진 클래스** 선정

➔ 회귀 : 예측값들의 **평균값을 예측값**

랜덤한 샘플과 특성 추출 후 모델들 생성

훈련 세트에 **과대적합 방지**

검증 세트, 테스트 세트의 **안정적인 성능 보장**

앙상블(ENSEMBLE)

◆ 랜덤포레스트(RandomForest)

OOB(Out Of Bag)

➔ 중복 샘플링으로 남겨진 데이터 발생

➔ 해당 데이터를 **검증 세트로 사용** ➔ **OOB Sample**

특성중요도

➔ 랜덤포레스의 장점

➔ **특성의 상대적 중요도 측정** 가능

앙상블(ENSEMBLE)

◆ 랜덤포레스트(RandomForest)

- Scikit-Learn LIB

```
sklearn.ensemble.RandomForestClassifier
( n_estimators=100          # DecisionTree 100개
  criterion='gini'
  max_depth=None
  min_samples_split=2
  min_samples_leaf=1
  min_weight_fraction_leaf=0.0
  max_features='sqrt'
  max_leaf_nodes=None
  min_impurity_decrease=0.0
```

앙상블(ENSEMBLE)

◆ 랜덤포레스트(RandomForest)

➢ Scikit-Learn LIB

```
sklearn.ensemble.RandomForestClassifier
(  bootstrap=True              # 중복 허용 랜덤 샘플링
   oob_score=False,            # OOB 샘플로 성능 평가, 검증 데이터로 사용
   n_jobs=None
   random_state=None,
   verbose=0
   warm_start=False
   class_weight=None
   ccp_alpha=0.0
   max_samples=None
 )
```

앙상블(ENSEMBLE)

◆ 엑스트라 트리(ExtraTrees)

- RandomForest와 매우 비슷
- **가장 극단적인 형태의 랜덤 포레스트**
- 분산 감소 - 편향 증가

- 차이점
  - **샘플데이터 ➔** 부트스트랩 샘플 사용하지 않음. 즉 **중복불허!**
    ➔ 전체 입력 샘플 사용
  - **노드 분할 ➔ 무작위 분할** (가장 좋은 분할 찾지 X)
    ➔ 무작위 분할 후 그 중 좋은 것 선택 후 분할
    ➔ **RandomForest에 비해 속도 빨라짐**
    ➔ **RandomForest 보다 Tree 수 늘려야 함**

앙상블(ENSEMBLE)

◆ 엑스트라 트리(ExtraTrees)

- Scikit-Learn LIB

```
sklearn.ensemble.ExtraTreesClassifier
( n_estimators=100        # DecisionTree 100개
  criterion='gini'
  max_depth=None
  min_samples_split=2
  min_samples_leaf=1
  min_weight_fraction_leaf=0.0
  max_features='sqrt'
  max_leaf_nodes=None
  min_impurity_decrease=0.0
```

앙상블(ENSEMBLE)

◆ 엑스트라 트리(ExtraTrees)

➢ Scikit-Learn LIB

```
sklearn.ensemble.ExtraTreesClassifier
(  bootstrap=False          # 중복 불허 랜덤 샘플링
   oob_score=False,         # OOB 샘플로 성능 평가, 검증 데이터로 사용
   n_jobs=None
   random_state=None,
   verbose=0
   warm_start=False
   class_weight=None
   ccp_alpha=0.0
   max_samples=None
 )
```

앙상블(ENSEMBLE)

◆ 부스팅(Boosting)

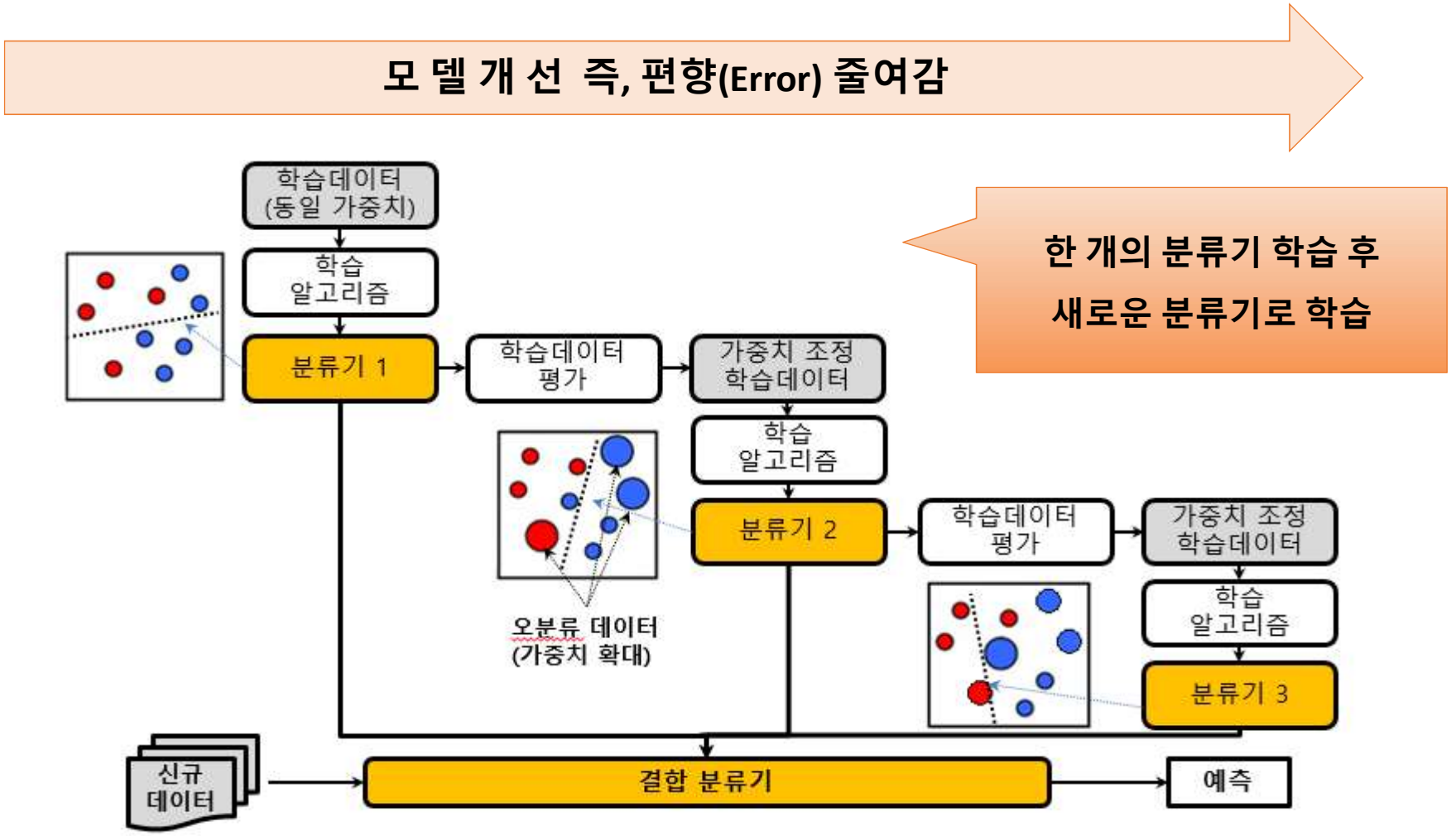
성능 약한 학습기 여러 개 순차적으로 모델(추정기) 추가 및 학습 진행

이전 모델(추정기)의 학습 결과 토대로 다음 모델(추정기)의 학습 데이터

샘플 가중치 조정해서 학습 진행

앙상블(ENSEMBLE)

◆ 부스팅(Boosting)



앙상블(ENSEMBLE)

◆ 부스팅(Boosting)

동작원리

- 이전 학습 결과의 오답에 대한 가중치 부여
- 다음 학습에서 오답에 대한 학습 진행
- 모델 성능 개선

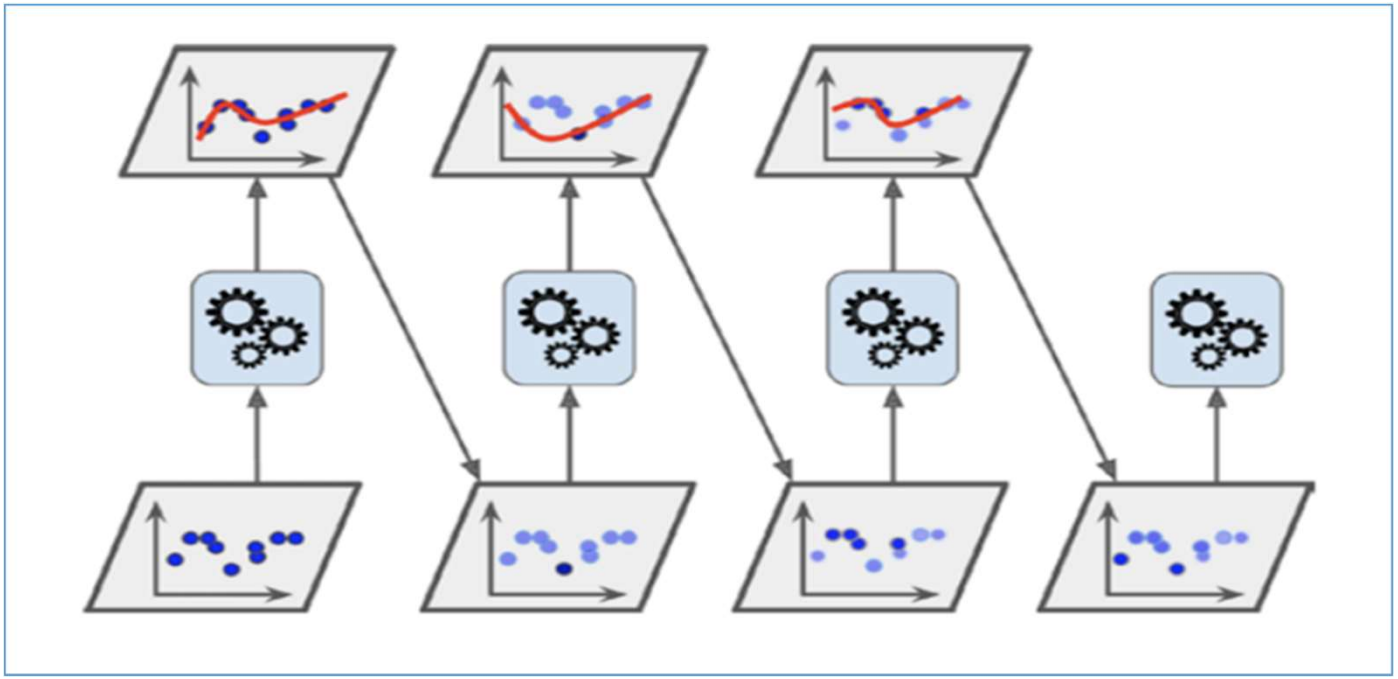
대표 알고리즘

- AdaBoost
- GradiantBoost
- XGBoost, LightGBM

앙상블(ENSEMBLE)

◆ 에이다부스트(AdaBoost)

- **좀 더 나은 모델 생성을 위해** 잘못 적용된 가중치 조정하여 새로운 모델을 추가하는 기법
- 과소적합했던 샘플들에 대한 가중치 더 높이는 방식



앙상블(ENSEMBLE)

◆ 그래디언트 부스팅(Gradient Boosting)

- AdaBoost와 동일한 방식
- 샘플의 가중치를 수정하는 **대신 이전 모델이 만든 오차(Error)에 대해 새로운 모델을 학습** 시키는 방식
- 모델 적용
  - 분류 : GradientBoostingClassifier
  - 회귀 : GradientBoostoingRegressor

앙상블(ENSEMBLE)

◆ 히스토그램기반부스팅(Histogram-based GB)

- **그레이디언트 부스팅의 속도를 개선**한 버전
- 안정적인 결과와 **높은 성능**으로 인기가 좋음
- 정형 데이터를 다루는 **머신러닝 알고리즘 중 가장 인기가 높음**

앙상블(ENSEMBLE)

◆ 히스토그램기반부스팅(Histogram-based GB)

- **동작원리**
  - ✓ **입력 특성 256개 구간**으로 나눔
  - ✓ 그중 하나 누락된 값을 위해 사용 ➔ 누락 특성 전처리 필요 x

- 대표 알고리즘
  - HistGradientBoostingClassifier
  - HistGradientBoostingRegressor
  - **XGBoost , LightGBM**

앙상블(ENSEMBLE)

◆ XGBoost(Extreme Gradient Boosting)

- Gradient Boosting을 분산환경에서도 실행할 수 있도록 구현
- **성능과 자원 효율 좋아 인기 있는 알고리즘**
- 분류, 회귀 모두 지원
- **여러개의 Decision Tree를 조합**해서 사용

- 설치 : !pip install xgboost

앙상블(ENSEMBLE)

◆ XGBoost(Extreme Gradient Boosting)

『 장 점 』

- GBM 대비 **빠른 수행시간**
  - 병렬 처리로 학습, 분류 속도 빠름
- 과적합 규제(Regularization)
  - 자체에 **과적합 규제 기능**으로 강한 내구성
- 분류와 회귀영역에서 **뛰어난 예측 성능** 발휘
- **Early Stopping(조기 종료)** 기능이 있음
- 다양한 옵션을 제공하며 **Customizing 용이**