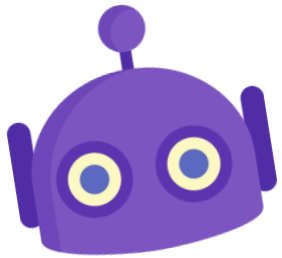
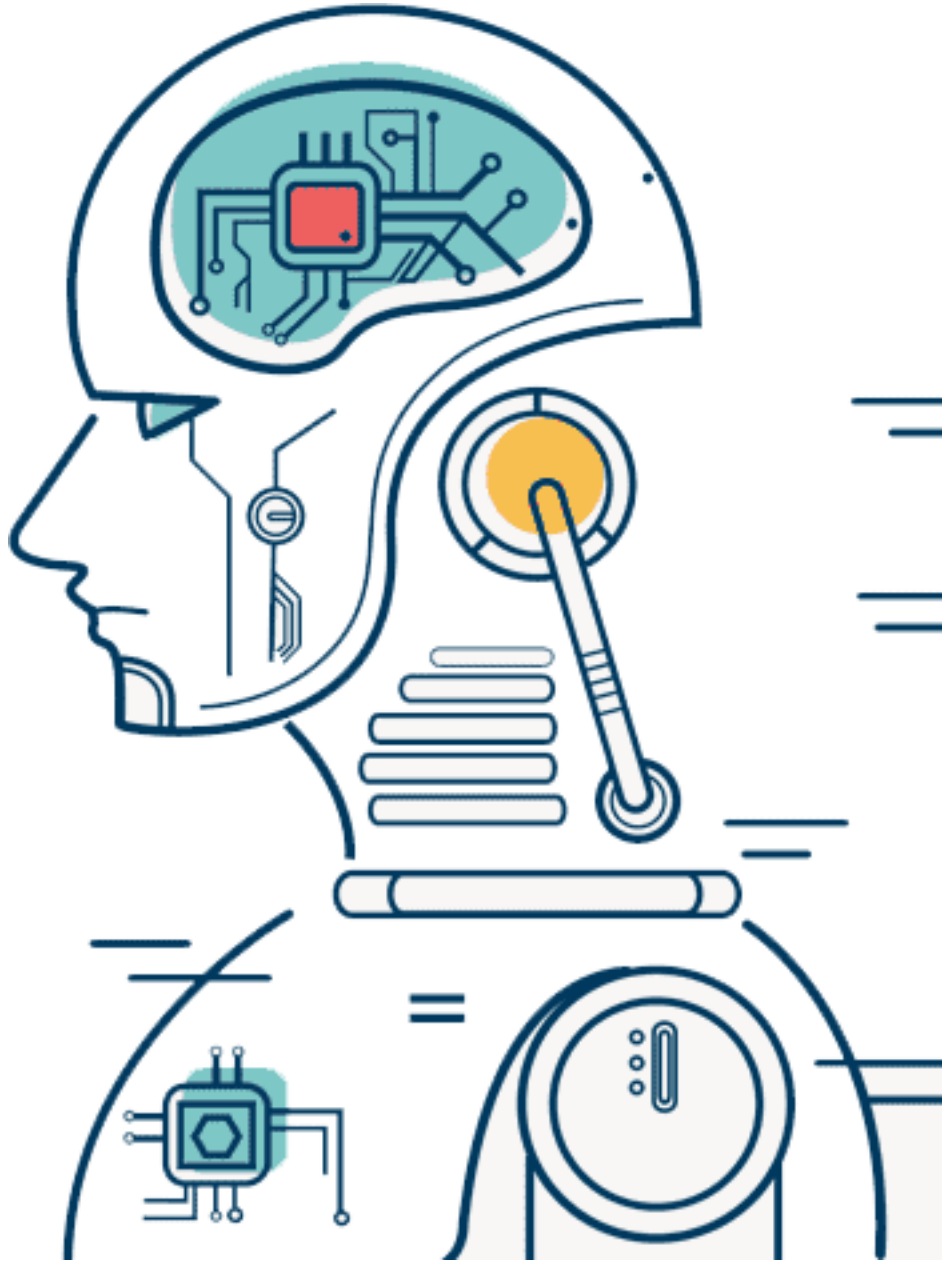
스마트인재개발원
Smart Human Resources Development



Ensemble Model

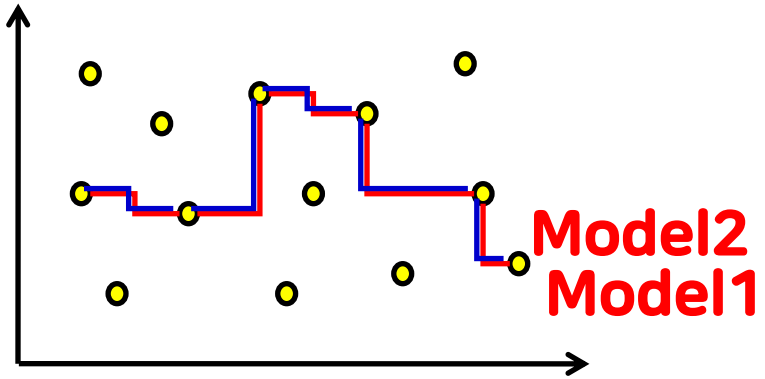
(bagging/boosting,
Random Forest, GBM, Xgboost)



Ensemble Model

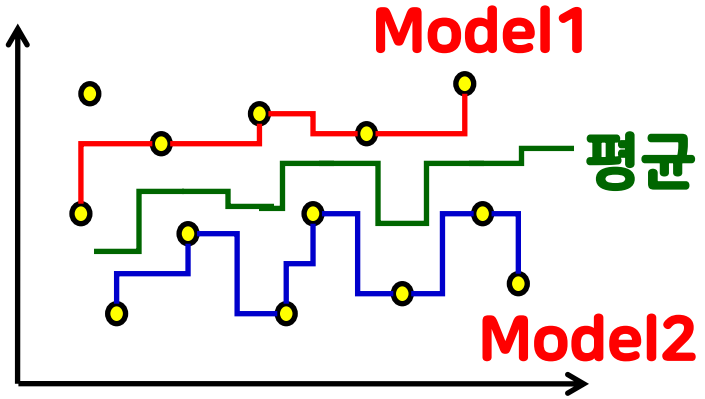
- 모델마다 결과가 다를 때 통합하는 방법의 일종 → 예측 모델들의 평균값을 사용

  → 모델이 많아지면 평균값이 모델의 실제 데이터와 유사해짐



Low bias

모델1과 2가 가까운 거리에 위치



High variance

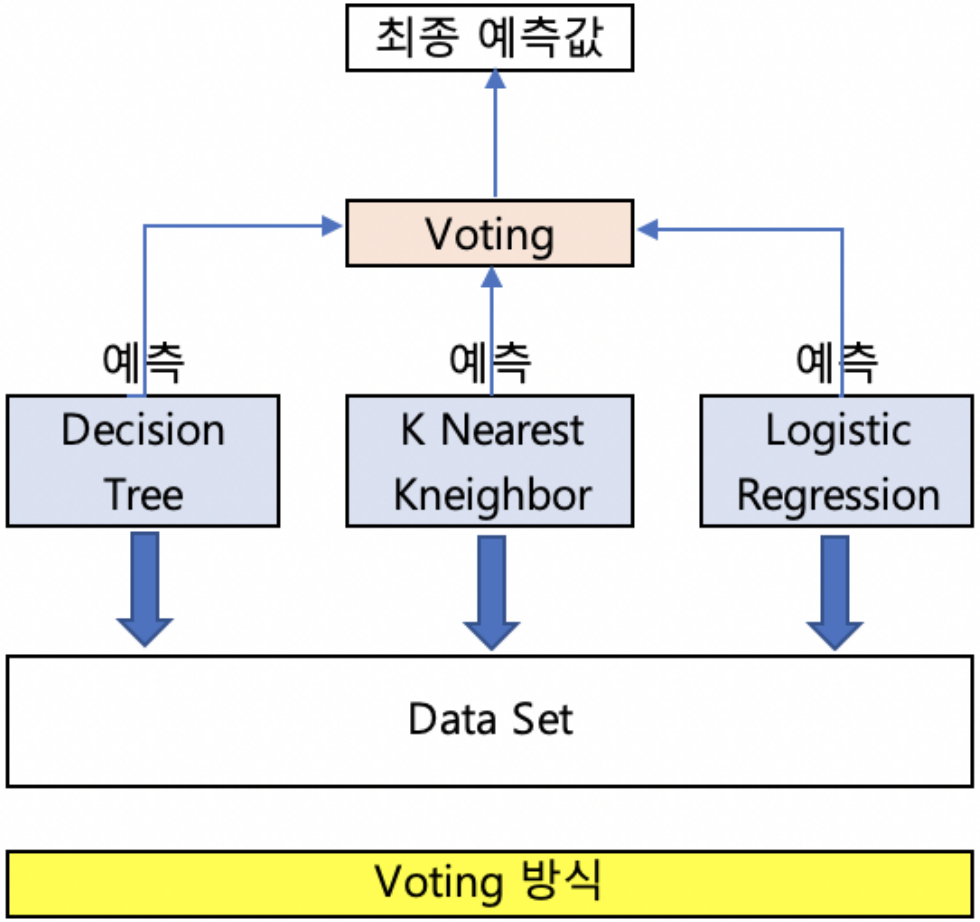
모델1과 2의 예측 값이 차이가 크다

Voting vs Stacking

- **Voting** : 여러 개의 학습 모델을 사용하여 평가 결과를 투표하는 방식
- **Stacking** : 여러 개의 모델을 사용하여 학습하고 학습 결과를 훈련 데이터로 사용하여 다시 학습하는 것

Ensemble Model

Voting

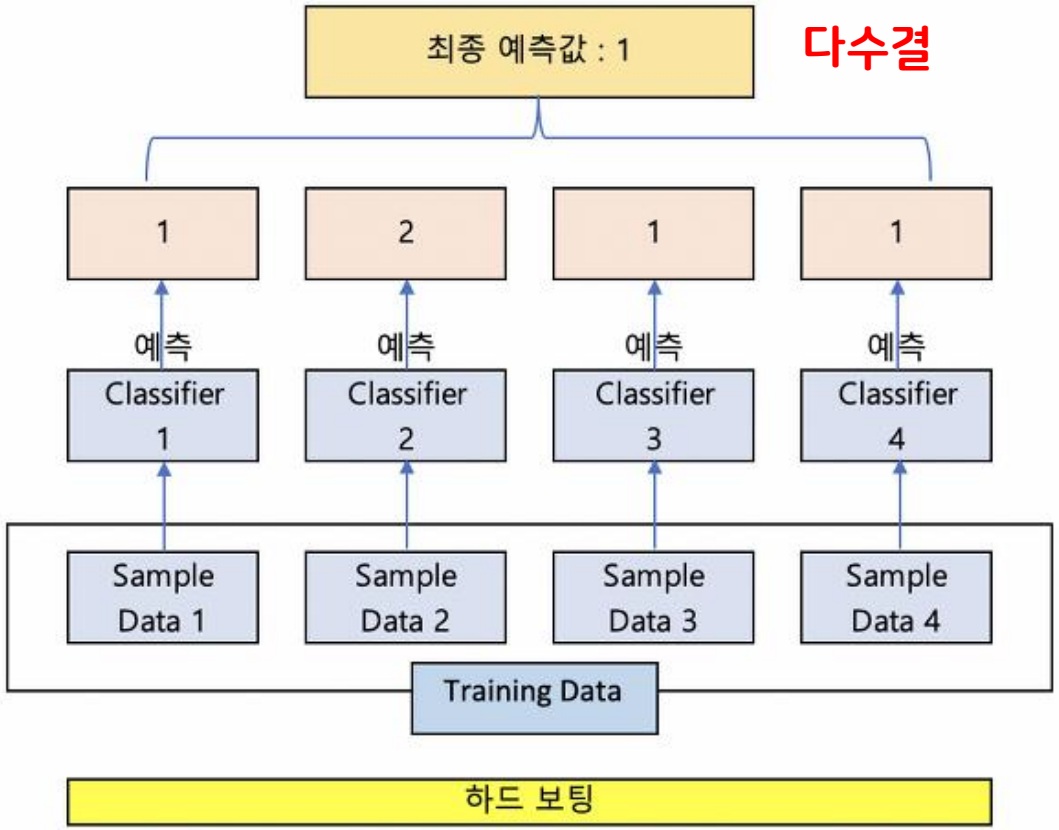


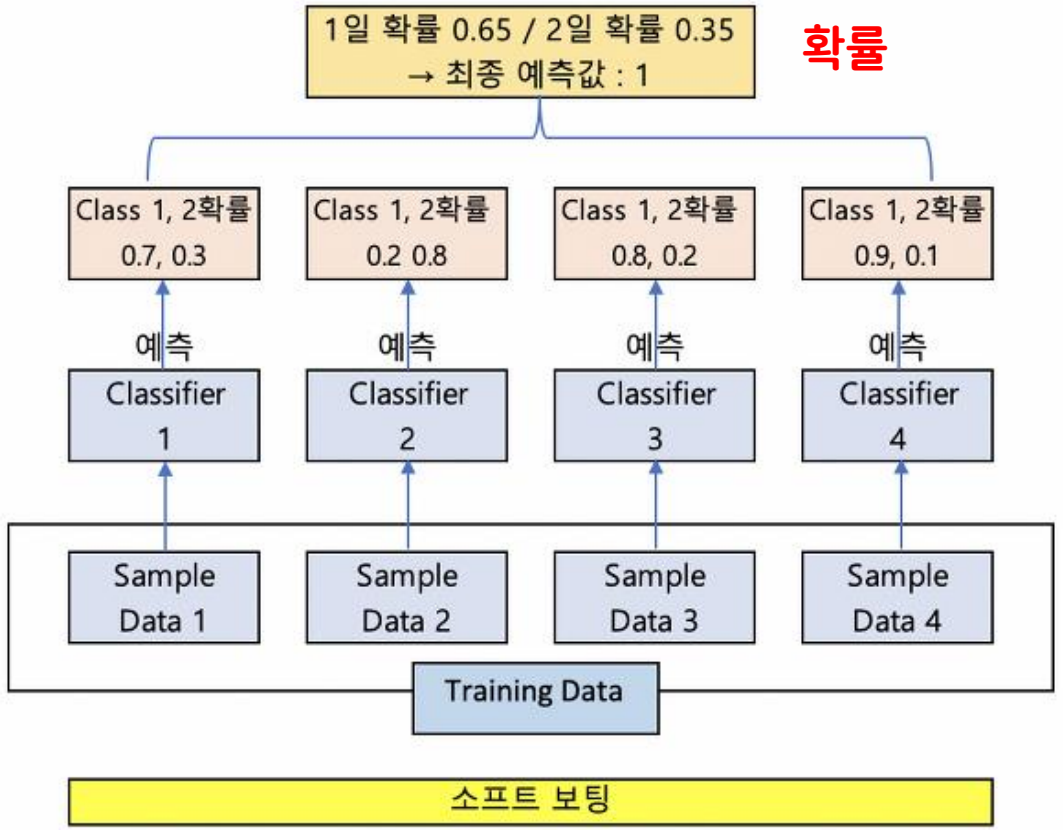
https://injo.tistory.com/22

Ensemble Model

스마트인재개발원
Smart Human Resources Development

Hard Voting vs Soft Voting



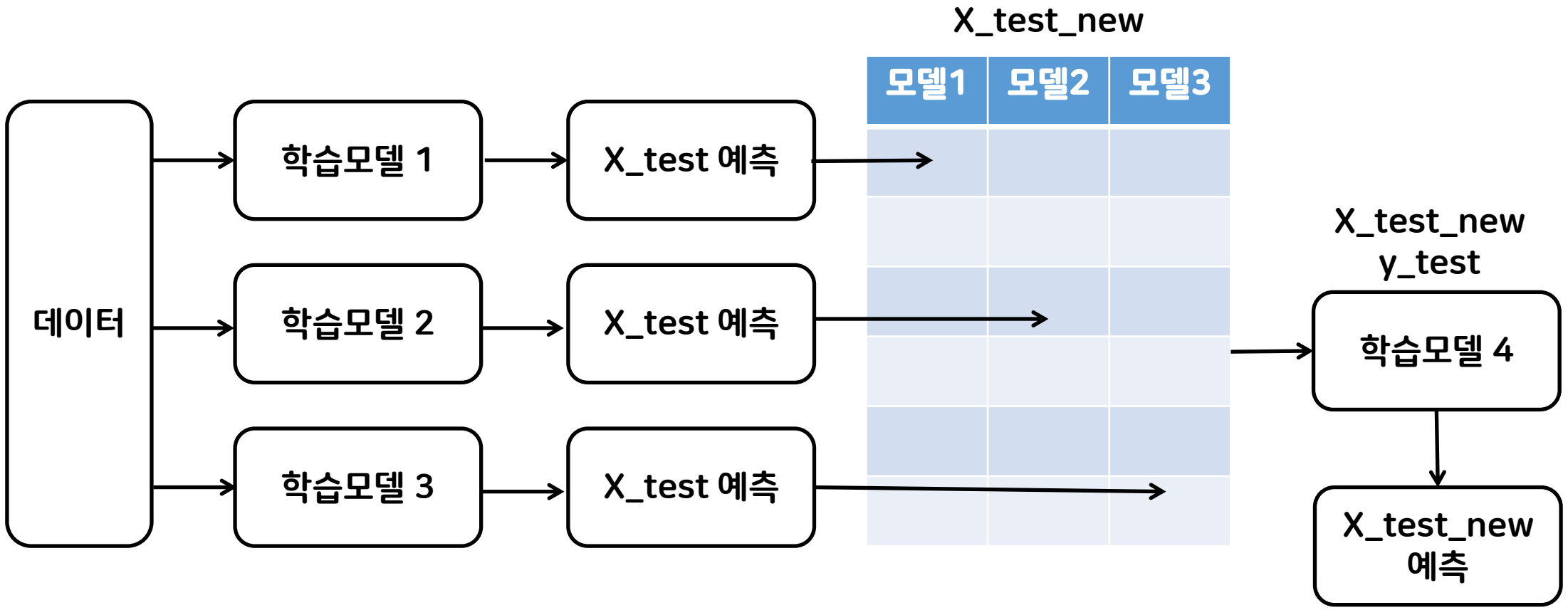


https://injo.tistory.com/22

Ensemble Model

스마트인재개발원
Smart Human Resources Development

Stacking



https://lsjsj92.tistory.com/558

Bagging vs Boosting

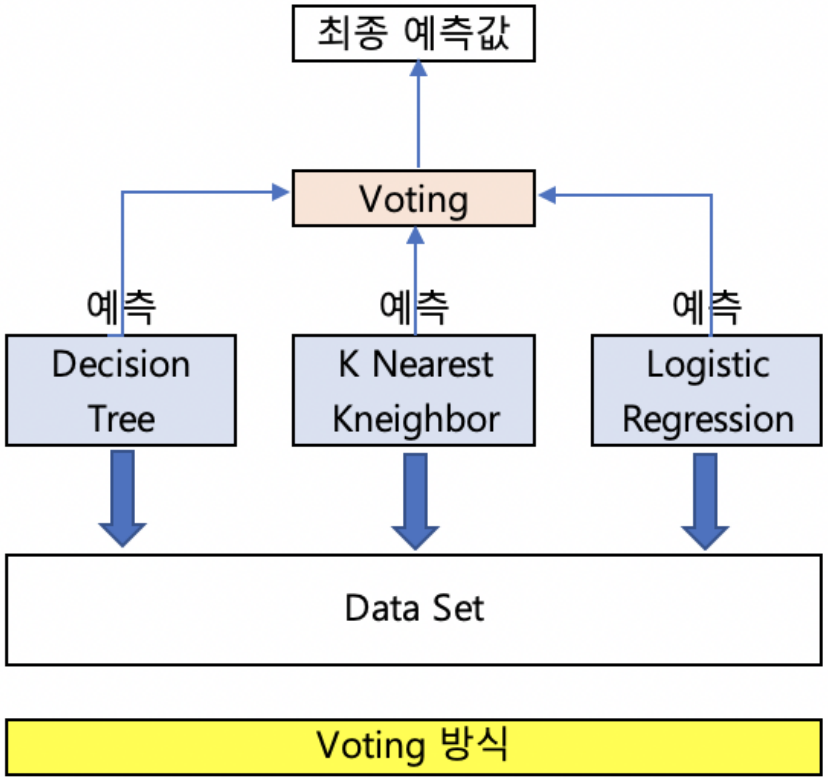
- **Boosting** : 일반적으로 학습 데이터의 개수가 작을 때 다수의 학습 데이터 대신에 이전 데이터의 오차를 고려하여 주어진 **학습 데이터를 가중치를 변경하면서 여러 번 반복 추출**하여 활용 (AdaBoost, GradientBoost, XGBoost 등)

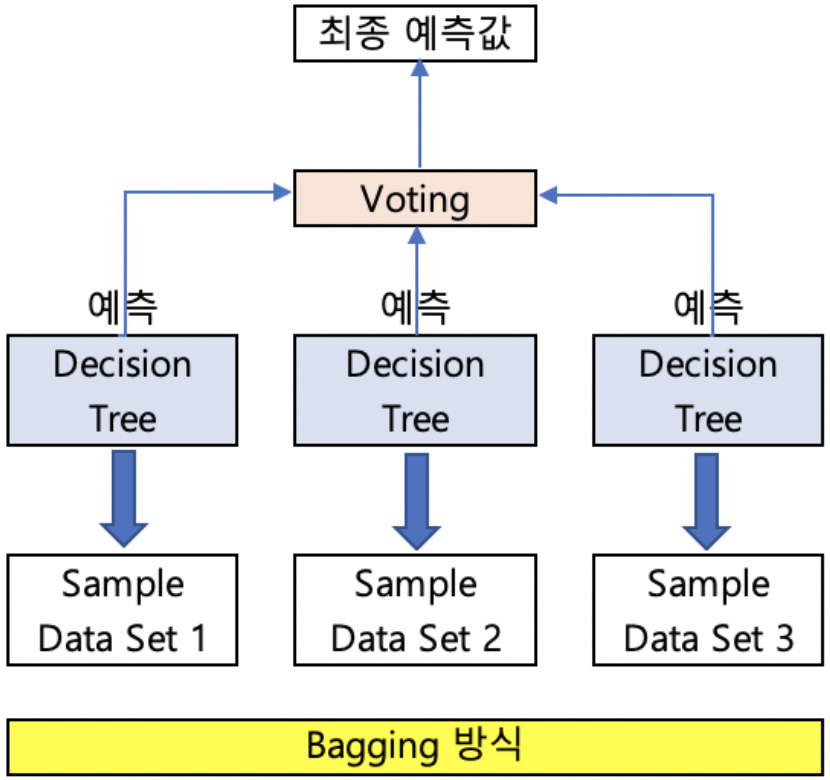
- **Bagging** (**B**oostrap **Agg**regation) : **약간씩 다른 독립된 훈련 데이터 세트를 여러 개 생성**하는 것으로 통계적 학습 (머신러닝)의 결과 분산을 줄이는 알고리즘
  → Overfitting 감소 → 이산 데이터인 경우는 투표(voting) 방식, 연속 데이터인 경우는 평균으로 집계 (RamdomForest 등)

Ensemble Model

스마트인재개발원
Smart Human Resources Development

Voting vs Bagging





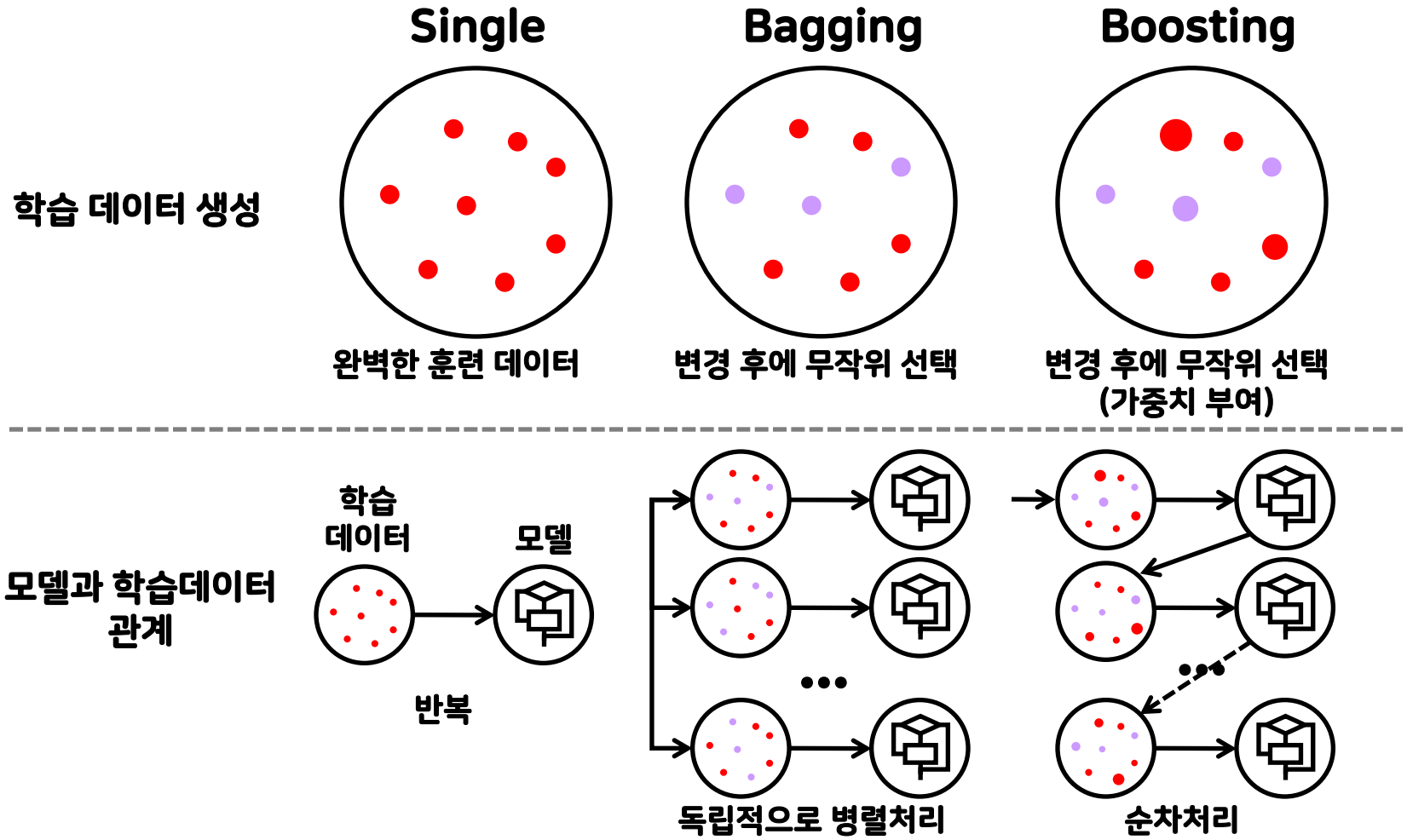
https://injo.tistory.com/22

Bagging vs Boosting

비교	Bagging	Boosting
특징	병렬 앙상블 모델 (각 모델이 서로 독립적)	연속 앙상블 (이전 모델의 오차를 고려)
목적	유사한 데이터 생성 → Variance 감소	오차를 감소시킨 데이터 생성 → Bias 감소
적합한 상황	복잡한 모델 (High variance, Low bias)	Low variance, High bias 모델
대표 알고리즘	Random Forest	AdaBoost, Gradient Boosting, Xgboost
데이터 선택	무작위 선택	무작위 선택 (오류 데이터에 가중치 적용)

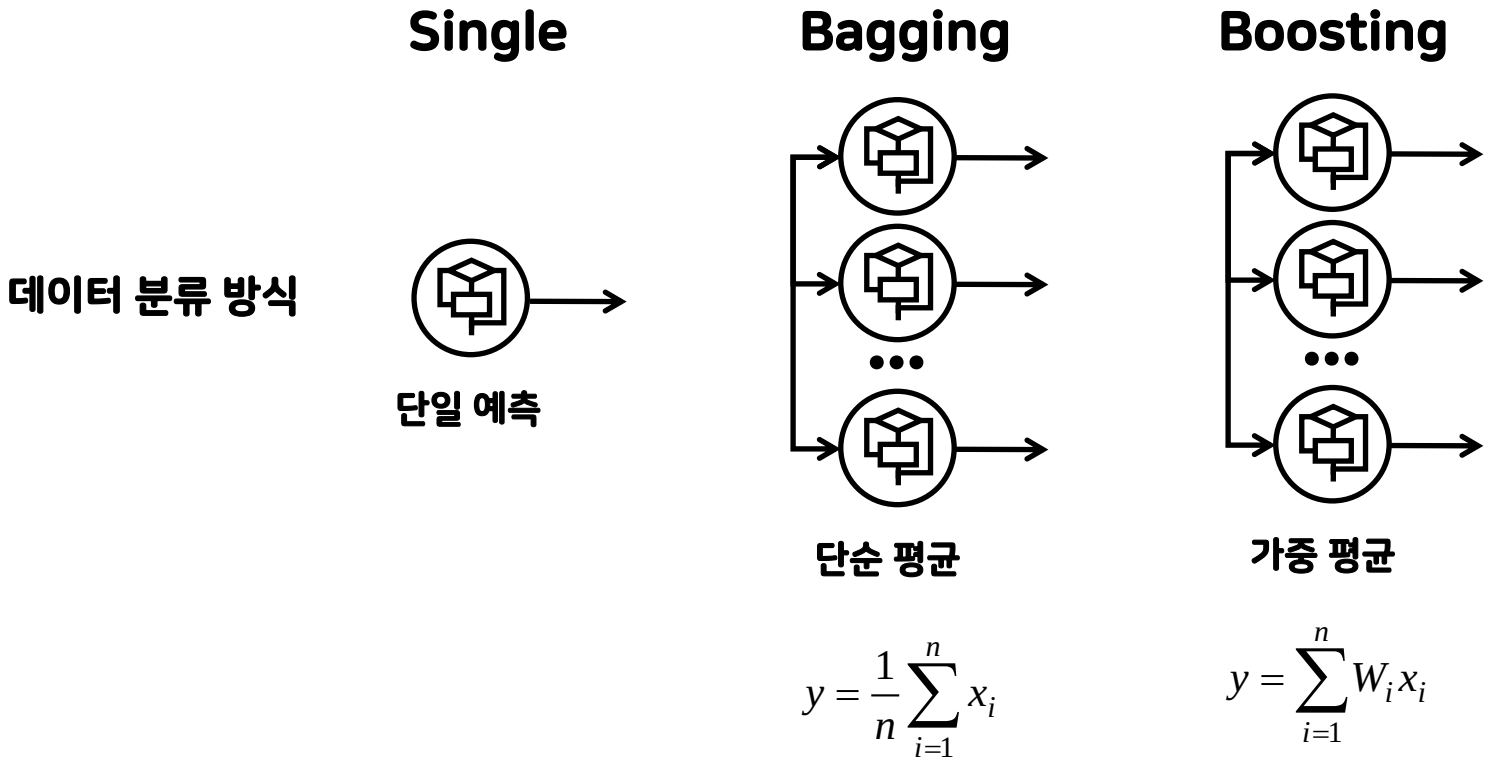
https://www.slideshare.net/freepsw/boosting-bagging-vs-boosting

Bagging vs Boosting



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

Bagging vs Boosting

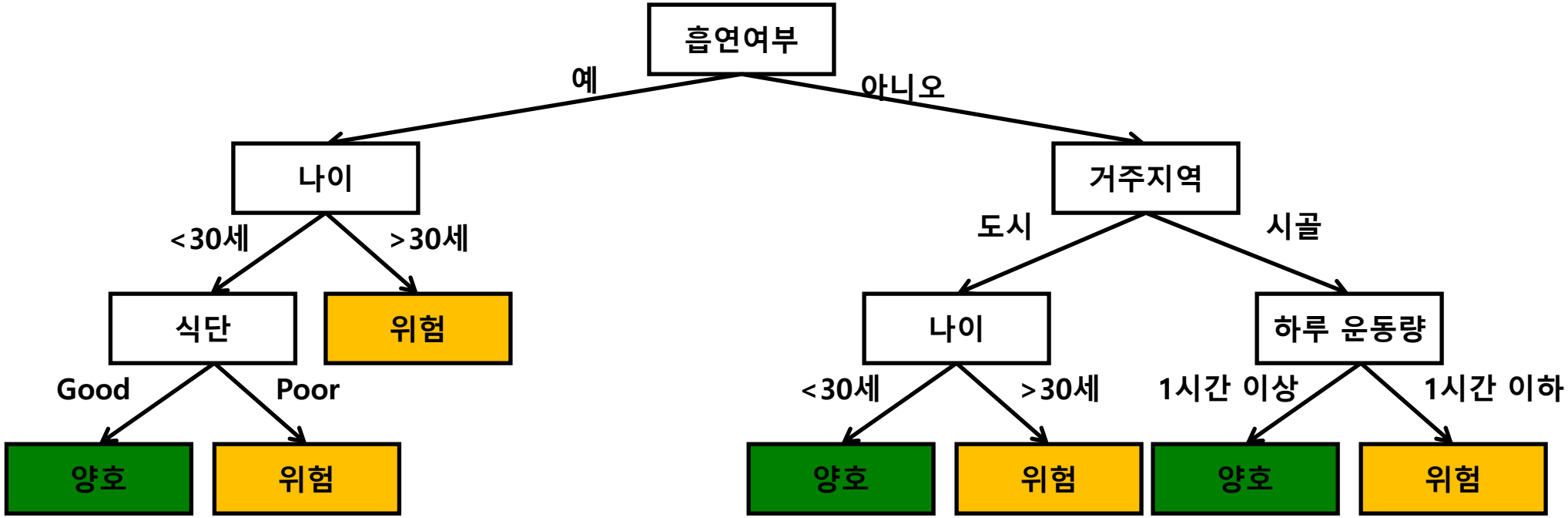


https://www.slideshare.net/freepsw/boosting-bagging-vs-boosting

Ensemble Model - RandomForest

스마트인재개발원
Smart Human Resources Development

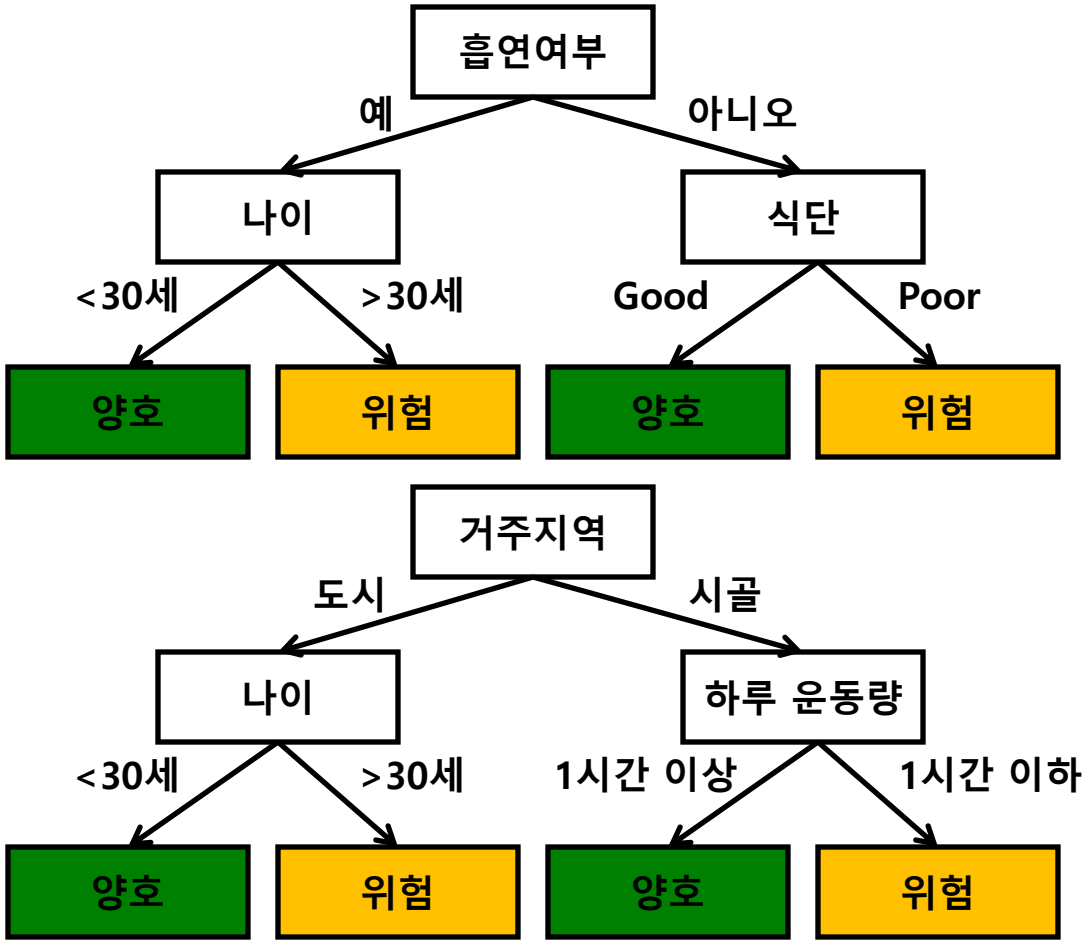
건강위험도를 예측하기 위한 의사결정트리

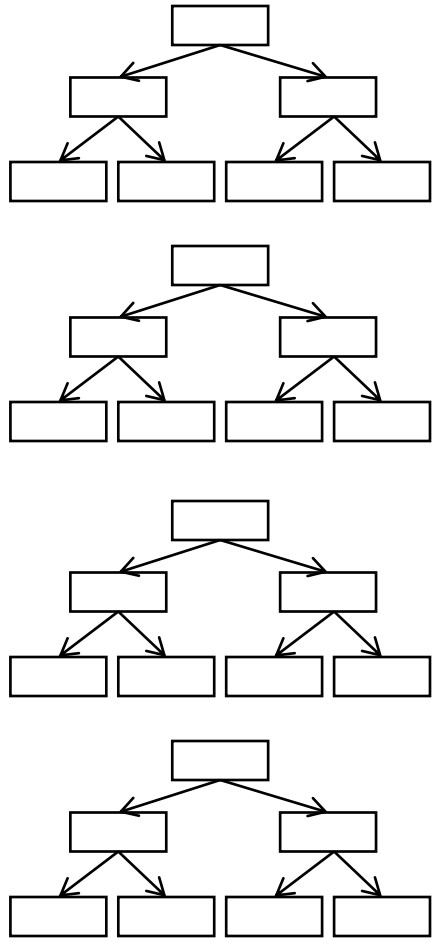


Ensemble Model - RandomForest

스마트인재개발원
Smart Human Resources Development

건강위험도를 예측하기 위한 랜덤포레스트

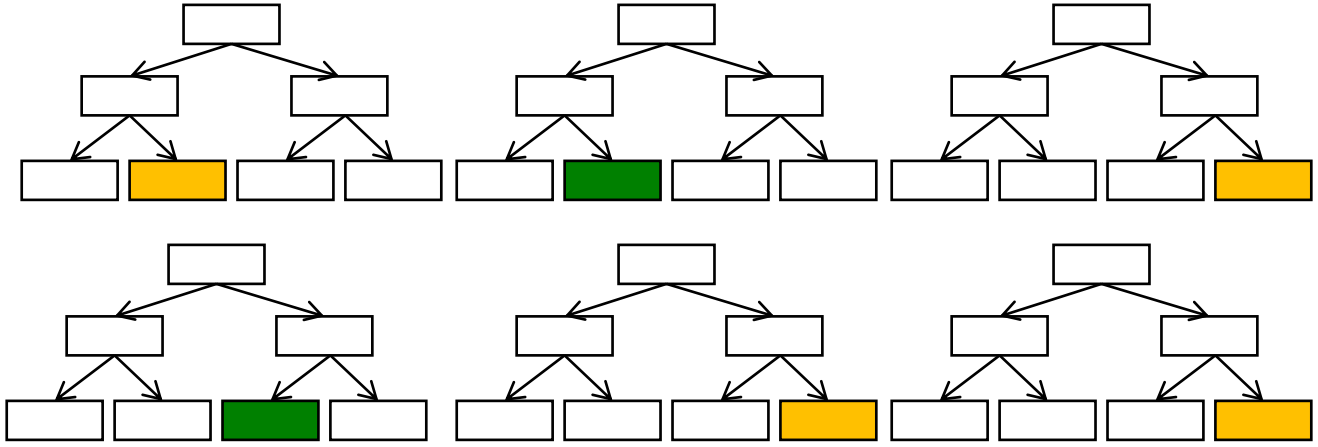




참고 : refactorthis.net의 포스트

Ensemble Model - RandomForest

- 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)

Ensemble Model - RandomForest

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

장단점

- 텍스트 데이터와 같은 희소한 데이터에는 잘 동작하지 않는다.
- 큰 데이터 세트에도 잘 동작하지만 훈련과 예측이 상대적으로 느리다.
- 트리 개수가 많아질 수록 시간이 더 오래 걸린다.

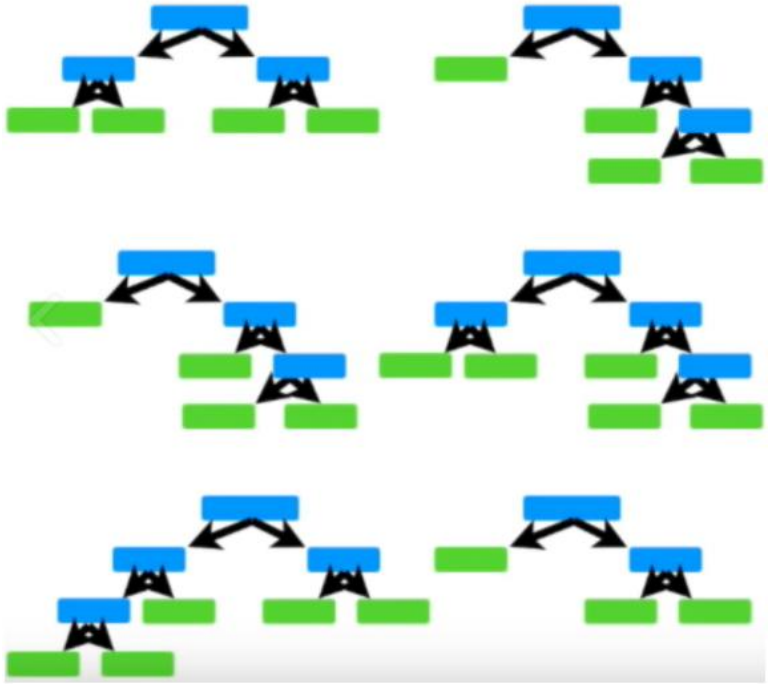
Ensemble Model - RandomForest

유방암 데이터를 100개의 트리로 만들어
RandomForest 모델로 학습해보자

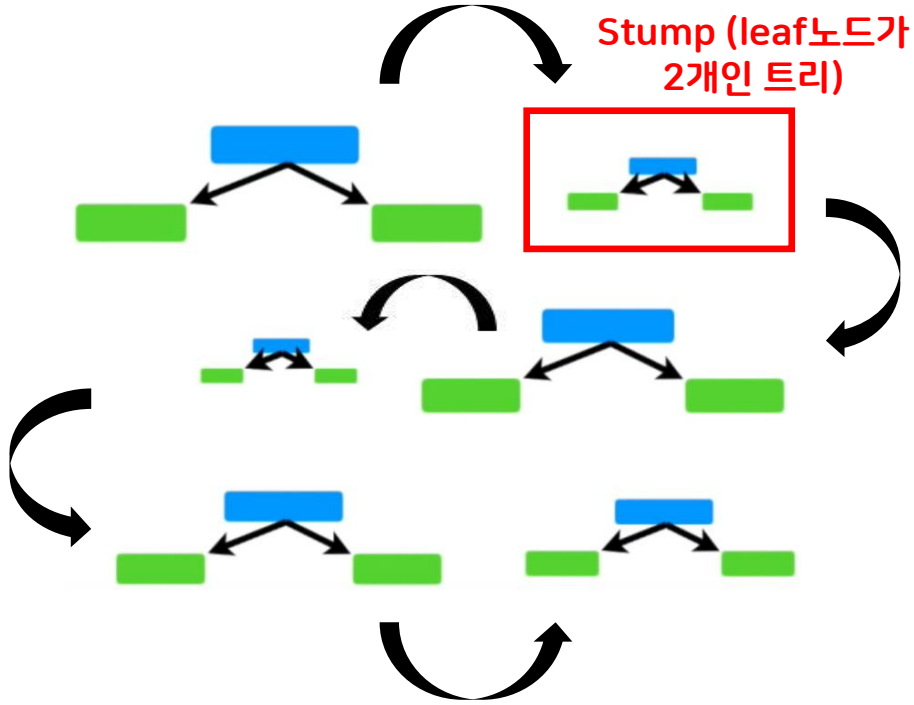
AdaBoost (Adaptive Boosting)

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

약한 학습기(Weak learner)



Random Forest



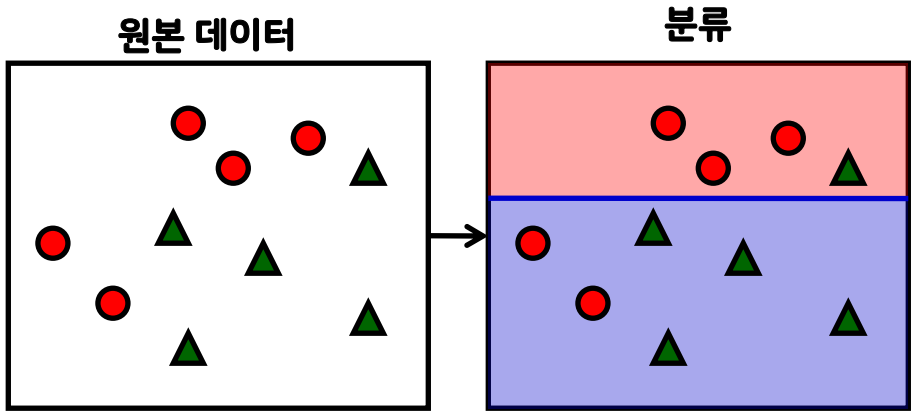
Ada Boosting

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

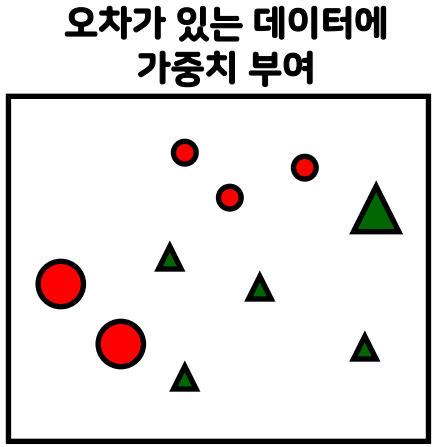


https://www.slideshare.net/freepsw/boosting-bagging-vs-boosting

AdaBoost (Adaptive Boosting)

- 동작 순서

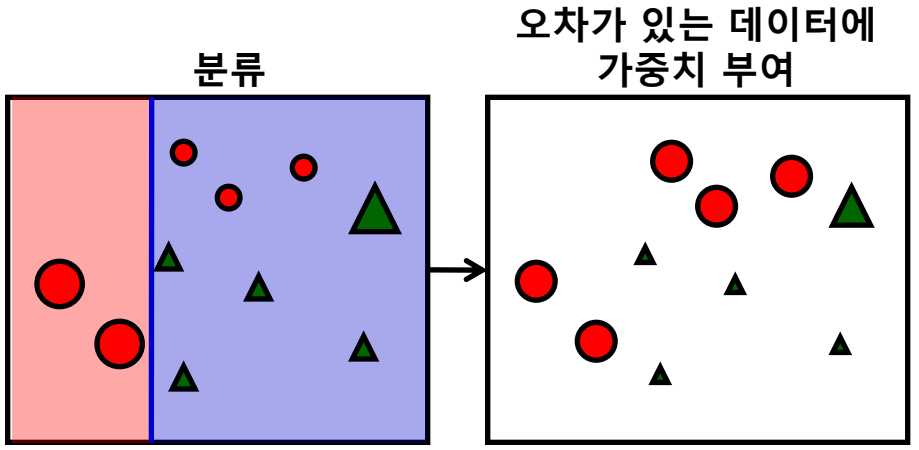
(2) 잘못된 2개의 빨간 원과 1개의 녹색 세모에 높은 가중치를 부여하고 맞은 것에는 빨간 원 3개와 녹색 세모 4개는 낮은 가중치 부여



AdaBoost (Adaptive Boosting)

- 동작 순서

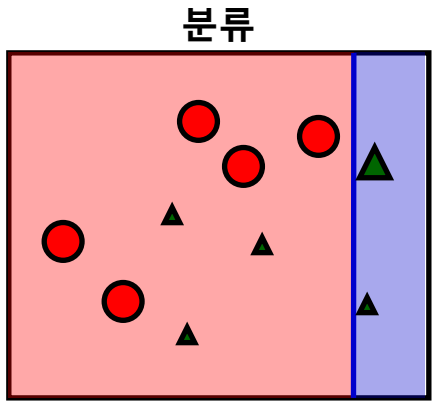
(3) 가중치를 부여한 상태에서 다시 분류 시킴 → 잘못된 3개의 빨간 원에 높은 가중치를 부여하고 맞은 5개의 녹색 세모는 낮은 가중치를 부여



AdaBoost (Adaptive Boosting)

- 동작 순서

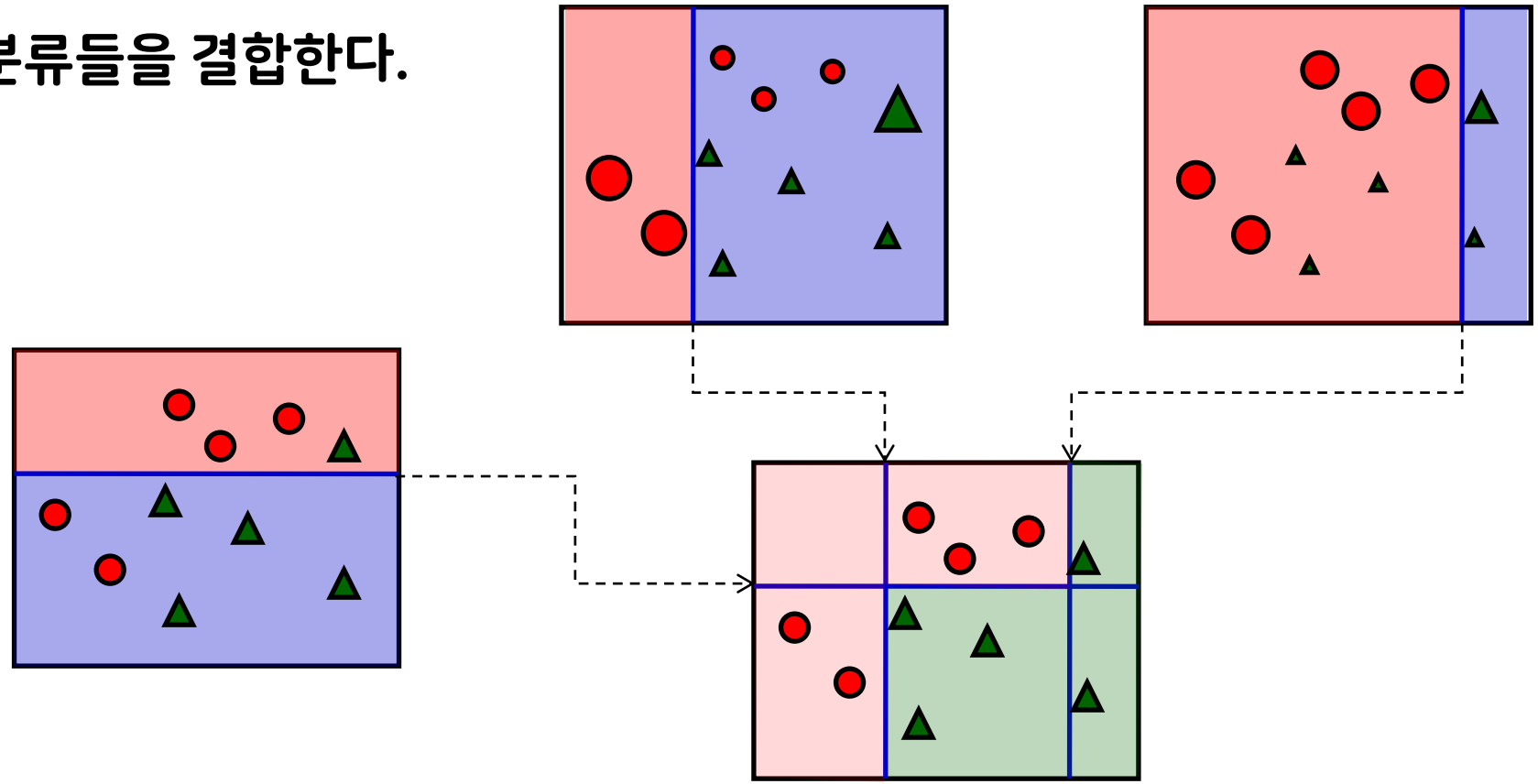
(4) 가중치를 부여한 상태에서 다시 분류 시킴



AdaBoost (Adaptive Boosting)

- 동작 순서

(5) 진행한 분류들을 결합한다.



주요 매개변수(Hyperparameter)

scikit-learn의 경우

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

- 트리의 개수 : n_estimators
- 선택할 데이터의 시드 : random_state

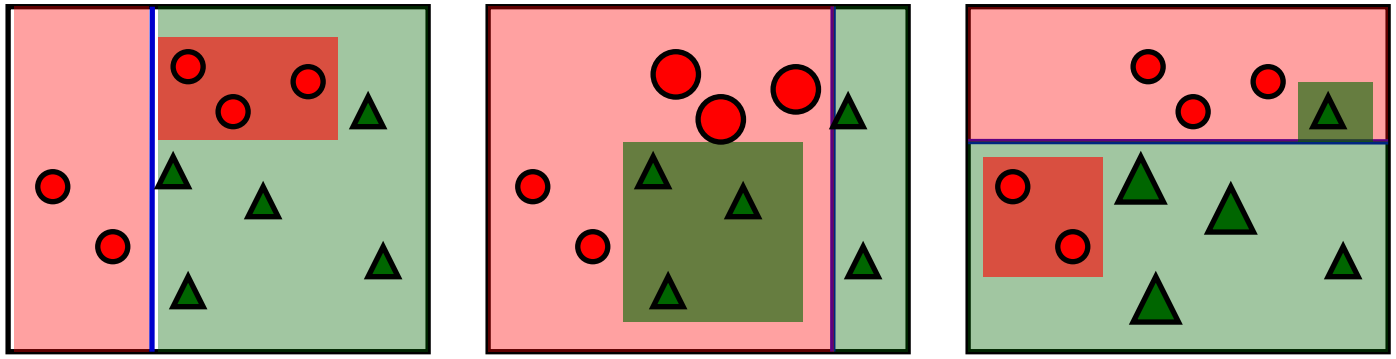
Ensemble Model - AdaBoost

유방암 데이터를
Adaboost 모델로 학습해 보고
특징의 중요도를 feature_importance_를
활용하여 bar차트로 표시해보자

Ensemble Model - GBM (Gradient Booting Machine)

스마트인재개발원
Smart Human Resources Development

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



3개 오류
→ 가중치 부여

3개 오류
→ 가중치 부여

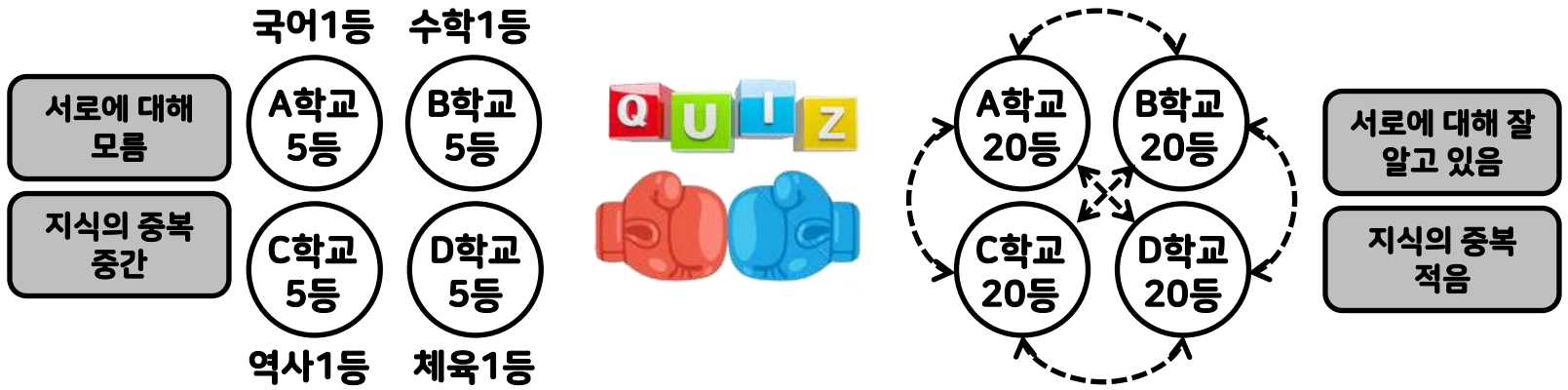
https://www.slideshare.net/freepsw/boosting-bagging-vs-boosting

Ensemble Model – 모델 비교

스마트인재개발원
Smart Human Resources Development

Decision Trees vs Random Forest

Random Forest vs AdaBoost

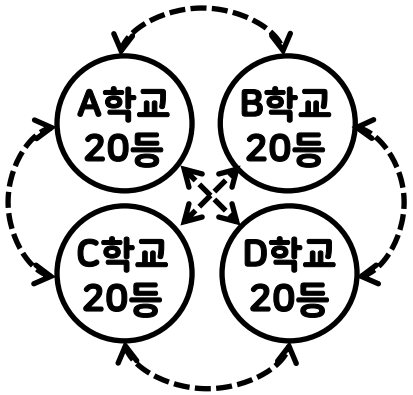


http://medium.com/@deepvalidation/군중은-똑똑하다2-adaboost-ba0122500034 정리

Ensemble Model – 모델 비교

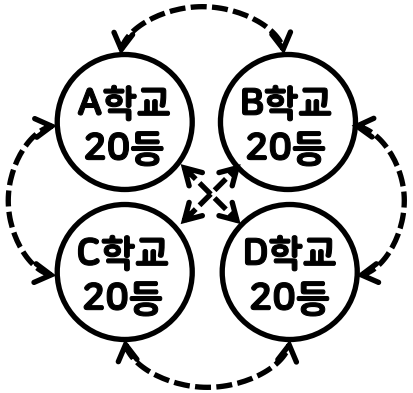
스마트인재개발원
Smart Human Resources Development

AdaBoost vs Gradient Boosting Machine



서로에 대해 잘 알고 있음

지식의 중복 적음



서로에 대해 잘 알고 있음

지식의 중복 적음

전문멘토

Ensemble Model - RandomForest

장단점

- 보통 트리의 깊이를 깊게하지 않기 때문에 예측 속도는 비교적 빠르다.
- 이전 트리의 오차를 반영해서 새로운 트리를 만 들기 때문에 학습속도가 느리다.
- 특성의 스케일을 조정하지 않아도 된다.
- 희소한 고차원 데이터에는 잘 동작하지 않는다
- 머신러닝의 성능을 마지막까지 쥐어짜야 할 때 활용

Ensemble Model - GBM (Gradient Booting Machine)

스마트인재개발원
Smart Human Resources Development

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

Ensemble Model - GBM (Gradient Booting Machine)

유방암 데이터를
GBM 모델로 학습해 보고
특징의 중요도를 feature_importance_를
활용하여 bar차트로 표시해보자

Ensemble Model - Xgboost

- GBM의 단점 : 느림, 과대적합 문제
- 대규모 머신러닝 문제에 그래디언트 부스팅을 적용하려면 xgboost 패키지를 사용 → 분산환경을 고려
- GBM보다 빠름 → Early Stopping 제공
- 과대적합 방지를 위한 규제 포함
- CART (Classification And Regression Tree)을 기반으로 함 → 분류와 회귀가 모두 가능

주요 매개변수(Hyperparameter)

scikit-learn의 경우

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

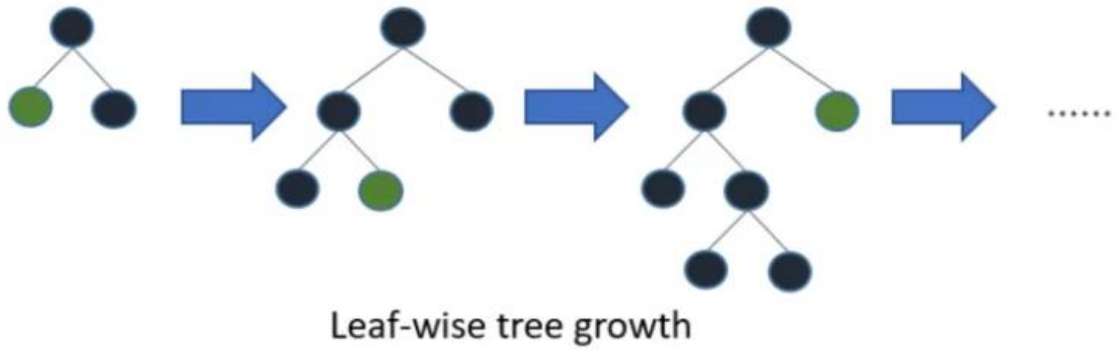
- 트리의 개수 : n_estimators
- 학습률 : learning_rate (높을수록 오차를 많이 보정)
- 트리의 깊이 : max_depth
- 선택할 데이터의 시드 : random_state

Ensemble Model - Xgboost

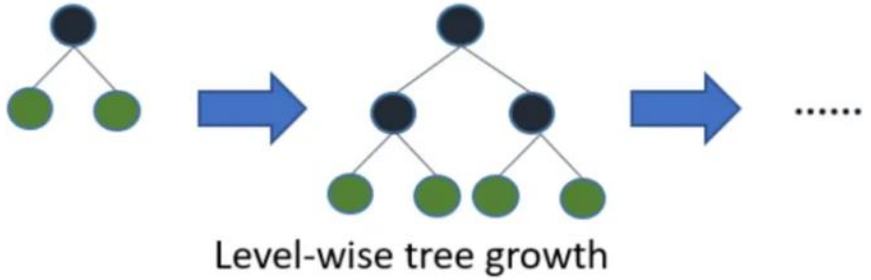
유방암 데이터를
Xgboost 모델로 학습해 보고
특징의 중요도를 feature_importance_를
활용하여 bar차트로 표시해보자

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 → 10000건 이상 데이터에 활용

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)

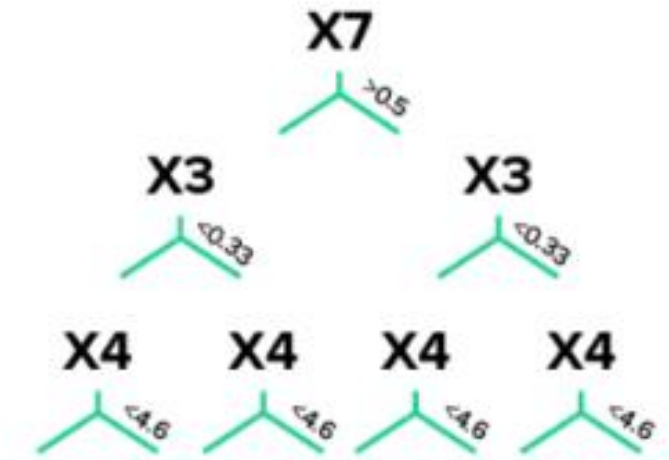
Cat Boosting(Categorical Boosting)

특징

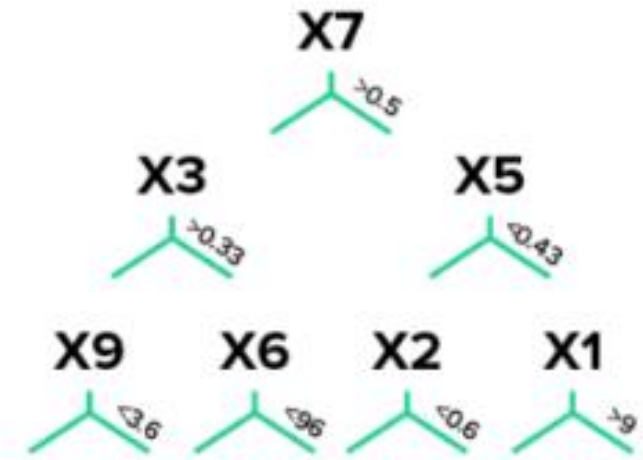
- 범주형 변수를 처리하는데 유용함
- Level_wise 방식
- 과대적합을 방지하기 위한 기법들이 다수 포함
  - → Ordered Boosting(일부 데이터로 모델링하며 점차 늘려나감)
  - → 데이터를 랜덤하게 셔플링하여 사용
  - → Ordered Target Encoding(범주형 변수일 경우 효과 ↑)
  - → 정보획득이 동일한 두 특성을 하나의 특성으로 묶음(전 처리 부담 ↓)
  - → 다른 부스팅에 비해 비교적 하이퍼파라미터 최적화가 잘 되어 있음(튜닝 소모 시간 ↓)

Cat Boosting(Categorical Boosting)

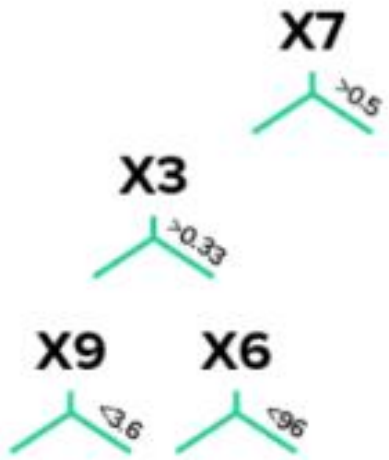
스마트인재개발원
Smart Human Resources Development



CatBoost

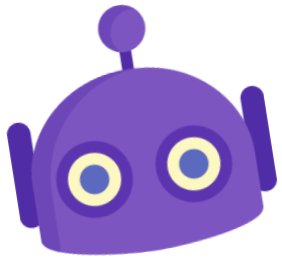


XGBoost

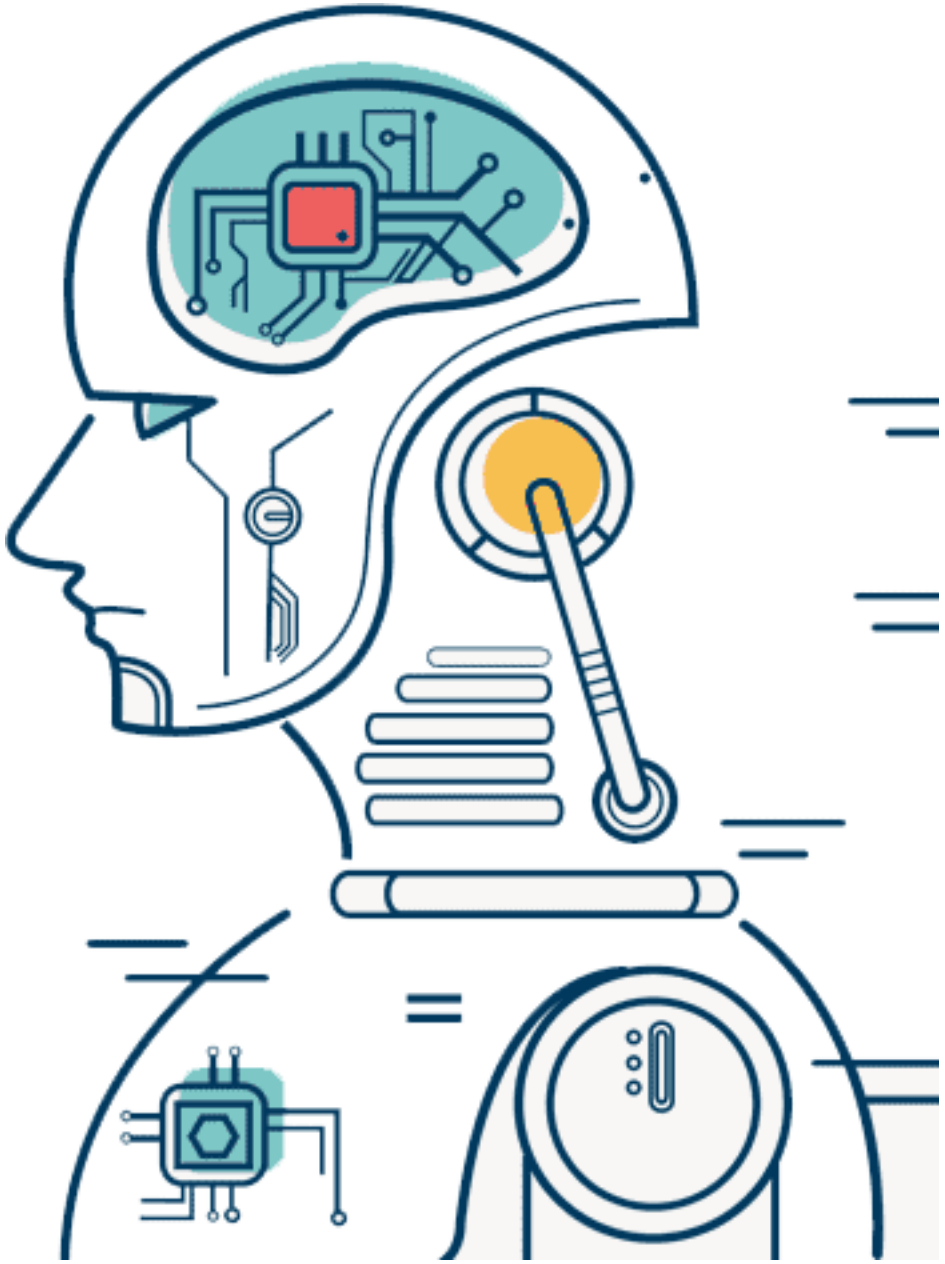


LightGBM

스마트인재개발원
Smart Human Resources Development



Hyperparameter Tuning



Hyperparamter Tuning - GridSearch

- 간단하고 광범위하게 사용되는 알고리즘
- 해당 범위 및 step의 모든 경우의 수를 탐색
- 범위를 넓게 가져갈수록 step을 작게 설정할 수록 최적해를 찾는 가능성은 높아지지만 시간이 오래 걸림
- 일반적으로 넓은 범위와 큰 step으로 설정한 후 범위를 좁혀 나가는 방식을 사용하여 시간을 단축
- scikit-learn의 GridSearchCV 모듈을 사용하여 구현

주요 매개변수(Hyperparameter)

scikit-learn의 경우

GridSearchCV(모델, 모델의 파라미터목록, scoring, cv)

- scoring : 성능 측정 도구 (accuracy, roc_auc, r2, neg_mean_squared_error 등)
  - 참고 : https://scikit-learn.org/stable/modules/model_evaluation.html
- cv : 교차검증 시 나눌 fold 수

Hyperparamter Tuning - GridSearch

- best_params_ : GridSearch 후에 찾는 최고의 파라미터 값
- best_score_ : 최고의 파라미터를 사용한 교차 검증 점수
- best_estimator_ : 전체 파라미터 값

Hyperparamter Tuning - GridSearch

유방암 데이터를 사용하여
RandomForest 모델의 최적 파라미터 찾기
(n_estimators, max_features)

Hyperparamter Tuning - RandomSearch

- 사전에 정해진 범위에서 Random하게 선택
- 기본적으로 더 빠르고 효율적이기 때문에 GridSearch보다 권장되는 방법
- GridSearch보다 속도가 빠르지만 최적값이 아닐 수 있음
- Sample 수가 많다면 최적값을 찾을 가능성이 높아짐
- scikit-learn의 RandomSearchCV 모듈을 사용하여 구현

Hyperparamter Tuning - RandomSearch

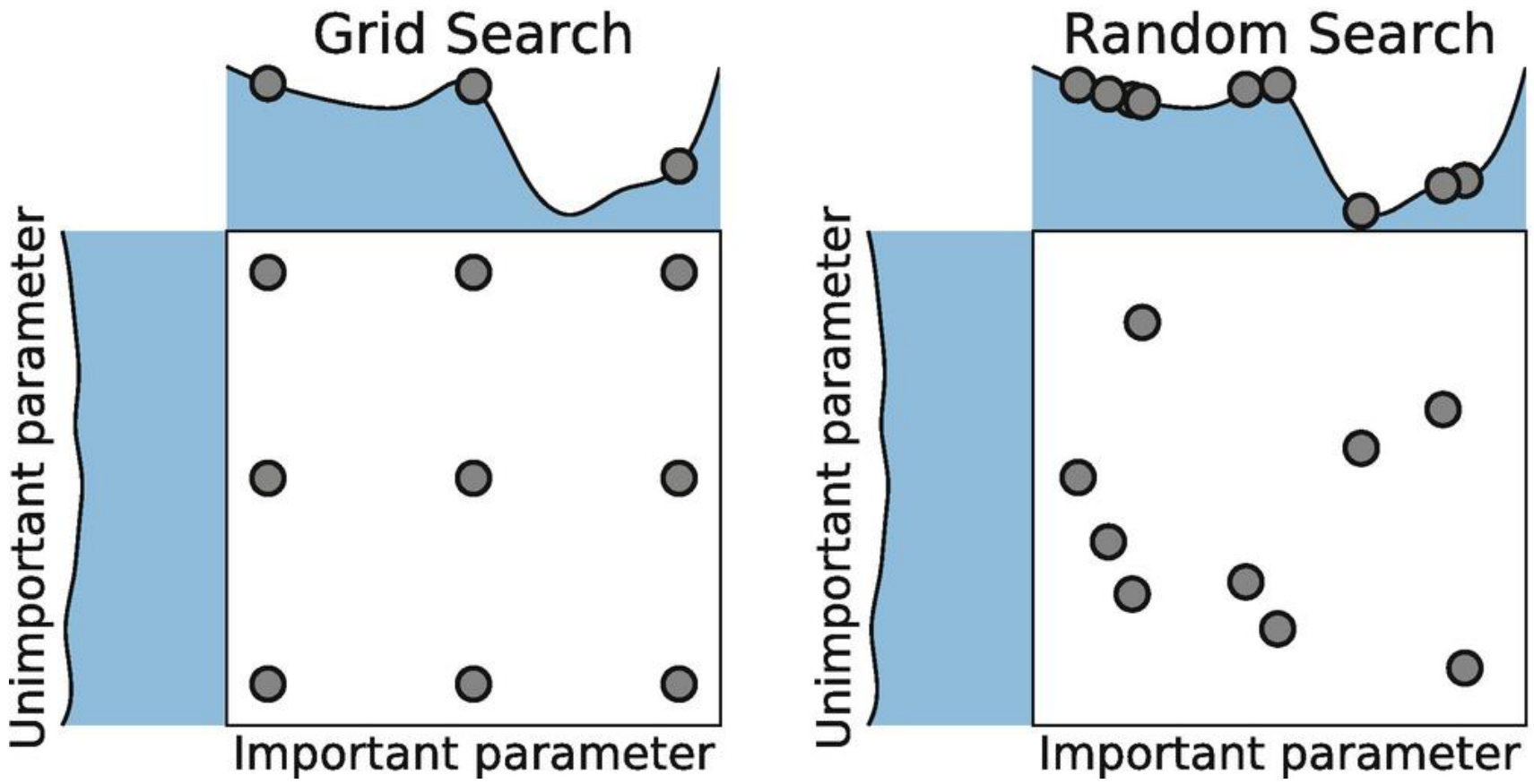
주요 매개변수(Hyperparameter)

scikit-learn의 경우

RandomizedSearchCV(모델, 모델의 파라미터목록, n_iter, scoring, random_state, cv)

- n_iter : random하게 골라지는 파라미터 조합의 수를 설정
- scoring : 성능 측정 도구 (accuracy, roc_auc, r2, neg_mean_squared_error 등)
- random_state : random 변수 생성 시 seed 값
- cv : 교차검증 시 나눌 fold 수

Hyperparamter Tuning - RandomSearch



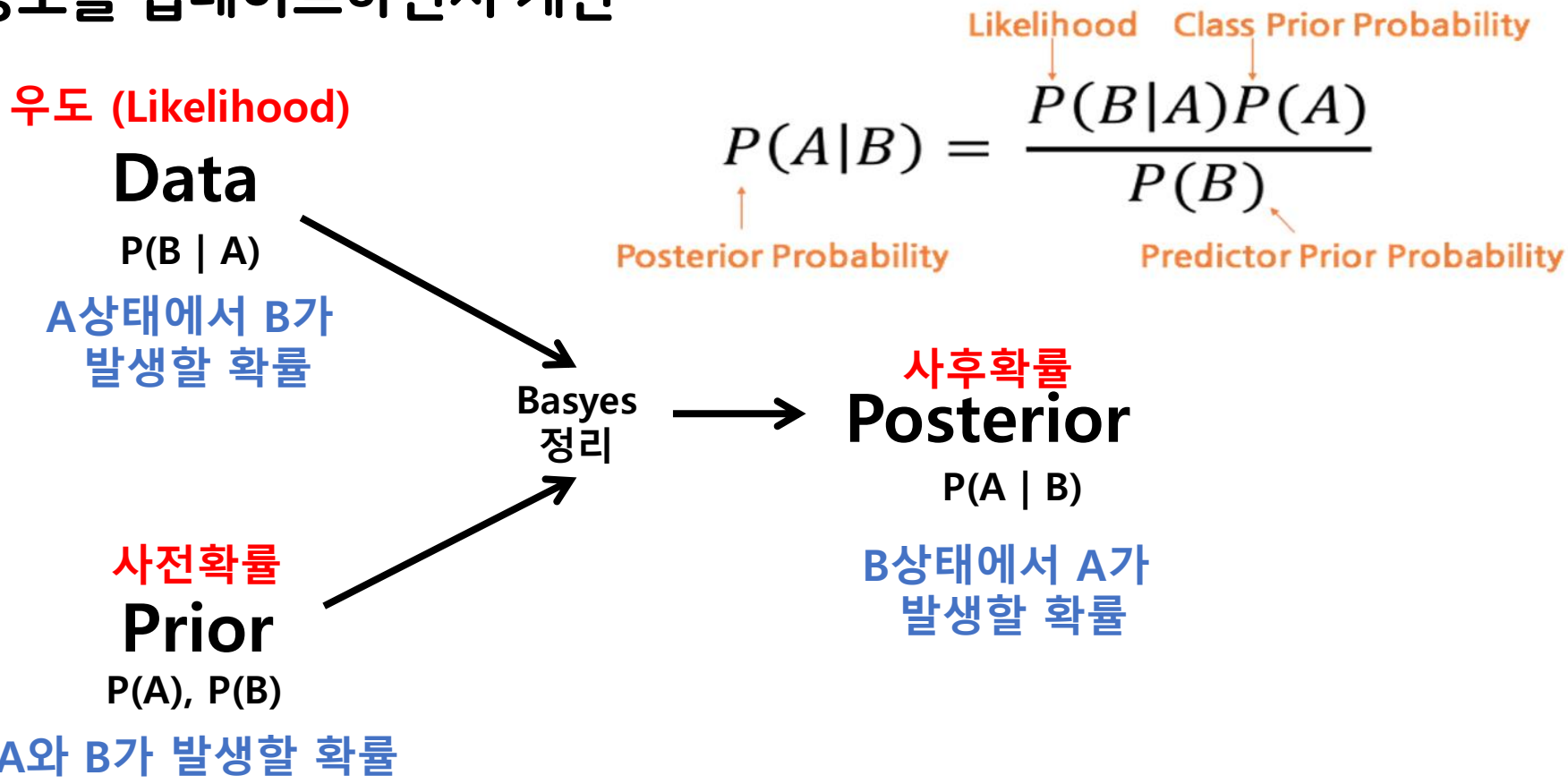
Hyperparamter Tuning – Bayesian Optimization

- 빈도 확률과 베이지안 확률
  - 빈도 확률 : 동전을 던졌을 때 앞면이 나올 확률과 같이 여러 번의 시도가 가능할 때 사용하는 확률
  - 베이지안 확률 : 화산폭발 확률과 같이 여러 번의 시도가 불가능할 때 사용하는 확률 – 사건과 관련있는 데이터를 이용해서 새롭게 일어날 사건을 추정하는 것

Hyperparamter Tuning – Bayesian Optimization

스마트인재개발원 Smart Human Resources Development

- **Bayes 이론** : 사전확률 P(A)와 우도확률 P(B|A)를 안다면 사후확률 P(A|B)를 알 수 있음 – 정보를 업데이트하면서 계산



Hyperparamter Tuning – Bayesian Optimization

- 안경을 쓴 사람이 여성일 확률 예측하기
  - 여성일 확률 : 50% → P(여성)
  - 여성이 안경을 쓸 확률 : 20% → P(안경 | 여성)
  - 안경 쓸 확률 : 40% → P(안경)

$$P(\text{여성}|\text{안경}) = \frac{P(\text{안경}|\text{여성}) \times P(\text{여성})}{P(\text{안경})} = \frac{0.2 \times 0.5}{0.4} = 0.25$$

Hyperparamter Tuning – Bayesian Optimization

- 사전에 정해진 범위에서 Random하게 R번 탐색한 후 B번 만큼 최적값을 찾음
- 사전 정보를 결정(Surrogate Model)하는 방법으로 Gausain Process를 여러 개의 하이퍼파라미터들에 대해 최적값 탐색을 위해 Aqusition Function을 적용
- 최적값을 찾을 수 있고 시간이 적게 걸림
- Random하게 찍힌 값에 따라 시간이 오래걸리거나 최적값 탐색이 불가능하는 경우가 발생
- 수동적으로 하이퍼파라미터 튜닝을 하는데 좋은 결과를 가져옴

Hyperparamter Tuning – Bayesian Optimization

- Surrogate Model
  - 현재까지 조사된 입력값을 바탕으로 미지의 목적함수의 형태에 대한 확률적인 추정을 수행하는 모델
- Acquisition Function
  - 최적의 입력값을 찾는데 있어 가장 유용할 만한 다음 입력값 후보를 추천해 주는 함수

Hyperparamter Tuning – Bayesian Optimization

스마트인재개발원
Smart Human Resources Development

(1) 파라미터와 파라미터의 탐색 구간 (a, b)을 설정

(2) 설정한 탐색 구간 내에서 처음 n개의 입력값들을 랜덤으로 샘플링하여 선택

(3) 샘플링한 값으로 모델을 학습(딥러닝)하여 성능결과를 수치로 계산

(4) Surrogate Model로 입력값과 성능결과에 대해 확률적 추정을 수행

(5) 확률적 추정결과를 바탕으로 탐색 구간에서 Acquisition function를 계산하고 가장 큰 점을 후보값으로 선정 → 후보값이 N개가 될때까지 반복

(6) 후보값들의 평균이 최대로 만드는 최적해를 최종 선택

Hyperparamter Tuning - Bayesian Optimization

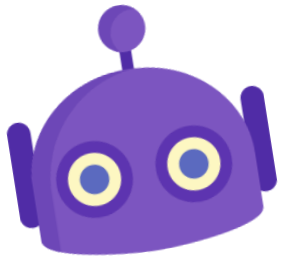
주요 매개변수(Hyperparameter)

scikit-learn의 경우

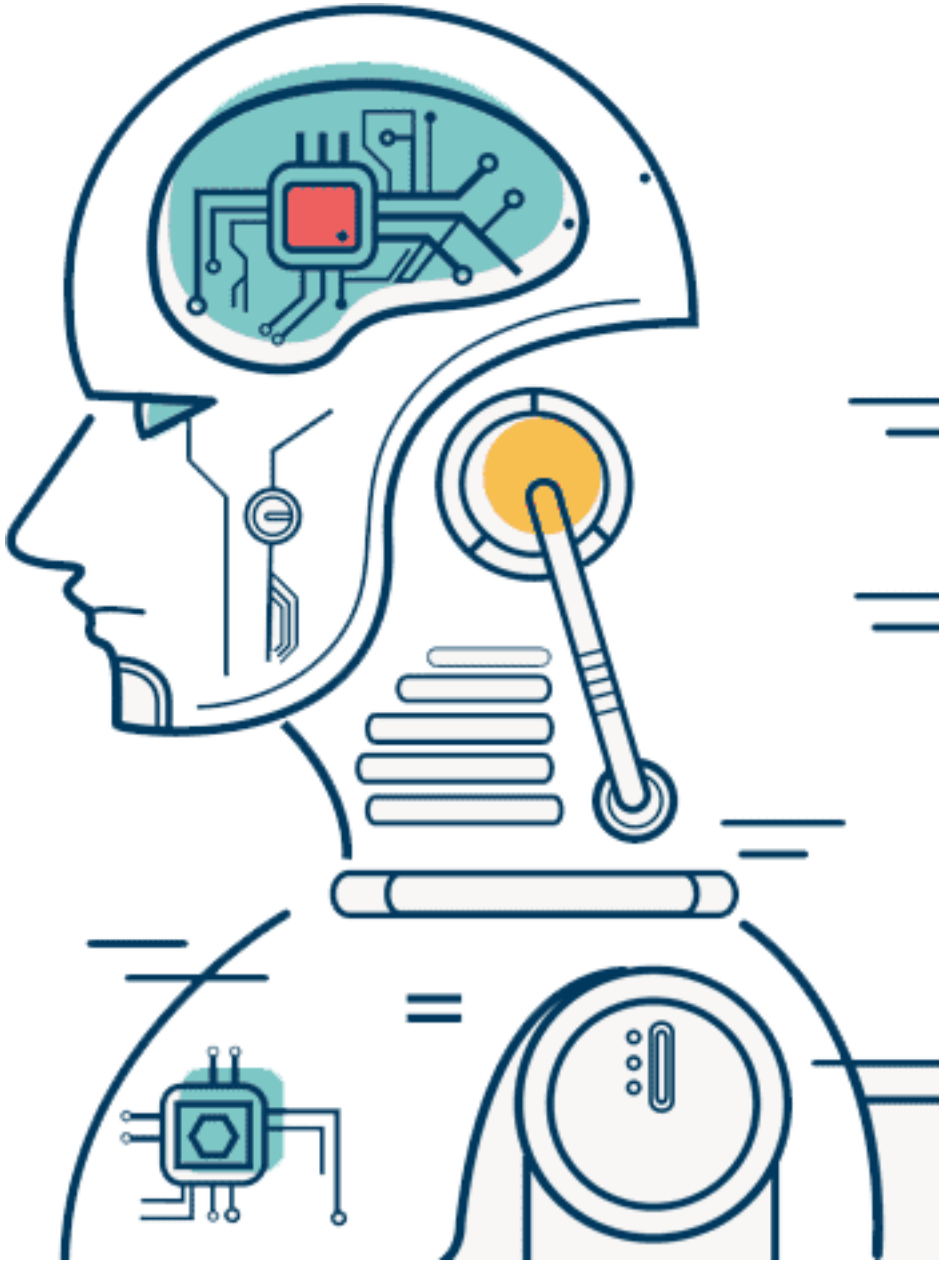
BayesSearchCV(모델, 모델의 파라미터목록, n_iter, scoring, random_state, cv)

- n_iter : random하게 골라지는 파라미터 조합의 수를 설정
- scoring : 성능 측정 도구 (accuracy, roc_auc, r2, neg_mean_squared_error 등)
- random_state : random 변수 생성 시 seed 값
- cv : 교차검증 시 나눌 fold 수

Smart Media
스마트미디어인재개발원



특성 선택
(Feature Selection)



Feature Selection

종류

- 일변량 통계
  - SelectKBest : 고정된 k개의 특징을 선택
  - SelectPercentile : 지정된 비율만큼 특징을 선택

- 모델 기반 선택
  - SelectFromModel : 학습 모델에서 계산된 중요도가 지정한 임계치 보다 큰 모든 특성을 선택

- 반복적 선택
  - RFE : 모든 특성을 시작해서 반복적으로 중요도가 낮은 특성을 제거

주요 매개변수(Hyperparameter)

scikit-learn의 경우

```
SelectKBest(score_func=f_classif, k=10)
```

- 특성성택 함수 : score_func
- 선택할 특성 수 : k

주요 매개변수(Hyperparameter)

scikit-learn의 경우

```
SelectPercentile(score_func=f_classif, percentile=50)
```

- 특성성택 함수 : score_func
- 선택할 특성의 비율 : percentile

주요 매개변수(Hyperparameter)

scikit-learn의 경우

```
SelectFromModel(모델(), threshold="median")
```

- 임계값 : threshold

주요 매개변수(Hyperparameter)

scikit-learn의 경우

```
RFE(모델(), n_features_to_select)
```

- 선택할 특성의 수 : n_features_to_select