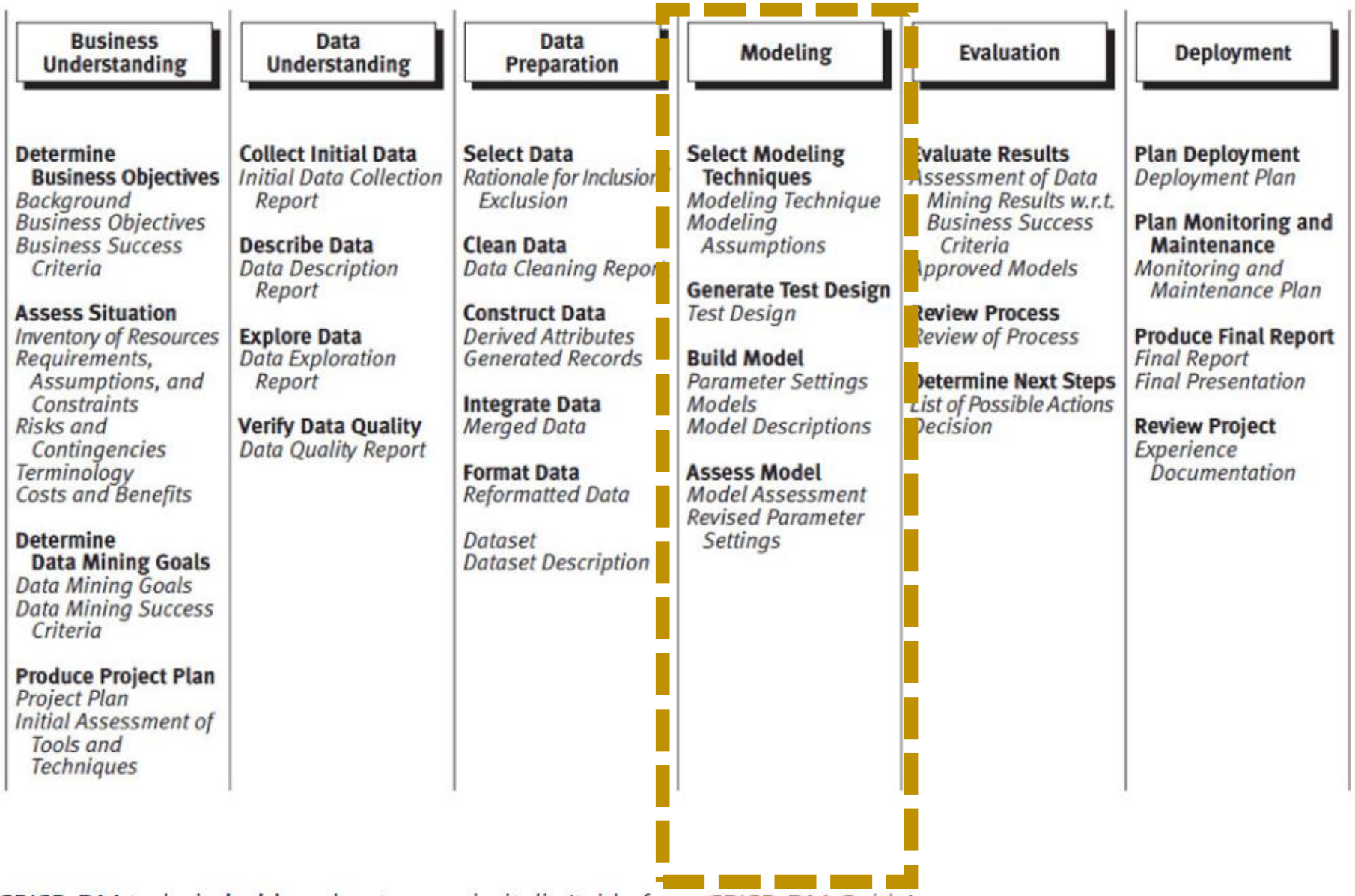
Ensemble

Ensemble



CRISP-DM tasks in **bold**, and outcomes in *italic* (table from CRISP-DM Guide)

Ensemble

What is Ensemble?

◎ 사전적 의미 : 2인 이상에 의한 가창이나 연주 / 주로 실내악을 연주하는 소인원의 합주단·합창단.

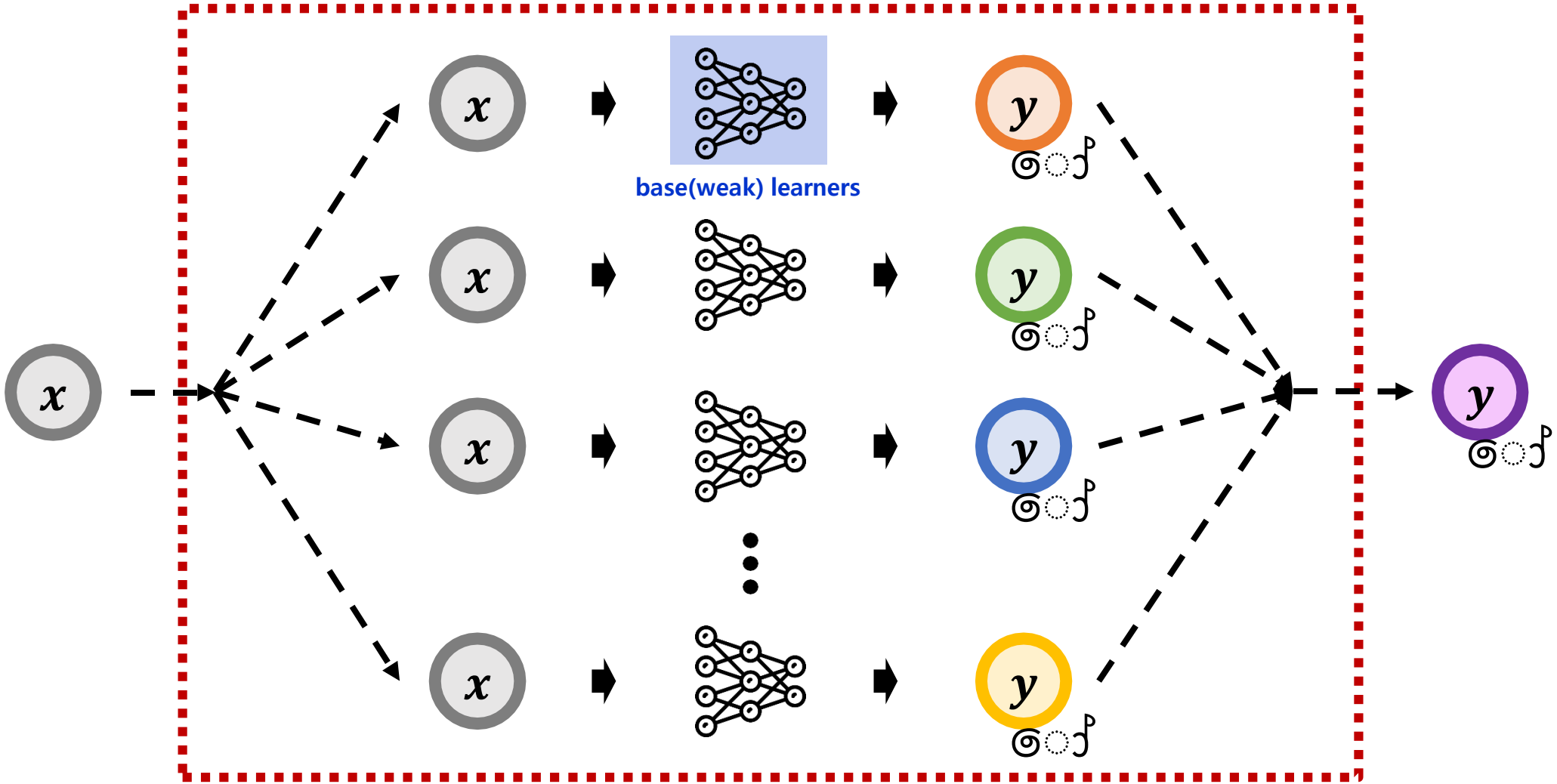
Ensemble

- **앙상블**
  - 여러 개의 기본 학습기(Base Learners =약한 학습기(Weak Learners))라고 불리는,
    다수의 예측 모델을 결합하여 단일 예측 모델보다 더 좋은 예측 성능을 예측을 목표로 하는 모델링 과정
  - 여러 개의 개별 모델을 조합하여 최적의 모델로 일반화하는 방법
  - 복잡하고 다양한 데이터셋에 대해 더 정확한 예측을 제공하기 위해 개발

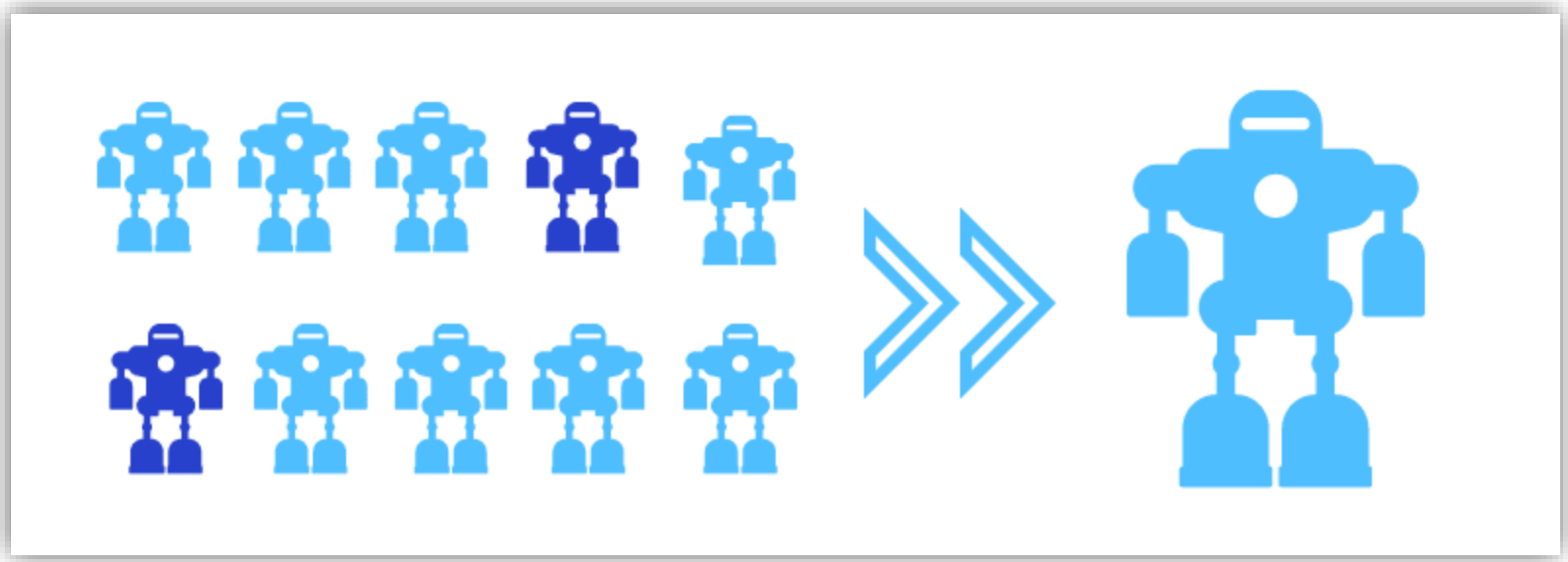
*** 약한 학습기(Weak Learners):**

- 혼자서는 데이터의 패턴을 완벽하게 학습하지 못하지만, 여러 개를 조합함으로써 강한 예측 성능을 낼 수 있는 모델을 뜻함
- 개별 모델이 약할지라도 **여러 모델의 예측을 적절히 결합, 성능 향상**

Ensemble



Ensemble



Ensemble

Diversity of Base Learners – 앙상블 학습에서의 다양성

① Manipulating the training data (데이터셋 조정)
→ bootstrap sampling, cross-validation, AdaBoost algorithm

② Manipulating the input features (입력 변수 조정)
→ Random forest, Feature Selection

③ Manipulating the output targets (출력 타겟 조정)
→ Label Transformation, Target Smoothing,
Class Imbalance Adjustment

④ Injecting randomness (무작위성 부여)
→ Neural Networks(randomness of the initial weights)

학습 과정에 의도적으로 무작위성을 추가
> 모델이 데이터의 복잡성 더 잘 포착,
> 과적합 방지,
> 일반화 성능 향상

Ensemble

Diversity of Base Learners

① Manipulating the training data (데이터셋 조정)
→ bootstrap sampling, cross-validation, AdaBoost algorithm

② Manipulating the input features (입력 변수 조정)
→ Random forest

③ Manipulating the output targets (출력 타겟 조정)
→ Label Transformation, Target Smoothing,
Class Imbalance Adjustment

④ Injecting randomness (무작위성 부여)
→ Neural Networks(randomness of the initial weights)

Ensemble

Ensemble - Model

- **Bagging (Bootstrap Aggregating)**
  - ✓ Bootstrap(부트스트랩) : 원본 데이터에서 중복을 허용하며 샘플링하는 기법
  - ✓ Aggregating(집계) : 개별 모델들의 예측 결과를 평균(회귀) 또는 다수결(분류) 방식으로 결합

- **Boosting**
  - ✓ Boost : 강화하다(Boost)라는 의미
  - ✓ 약한 모델(Weak Learner)을 여러 개 쌓아가면서 점점 더 강한 모델로 만듦

- **Stacking**
  - ✓ Stack : 쌓다(Stack)
  - ✓ 개별 모델(Base Learners)의 예측 결과를 메타 모델(Meta Model)에 입력, 또 다른 모델을 학습
    -> Stacking (여러 개의 모델을 쌓아서 최종적인 예측 수행)

Ensemble

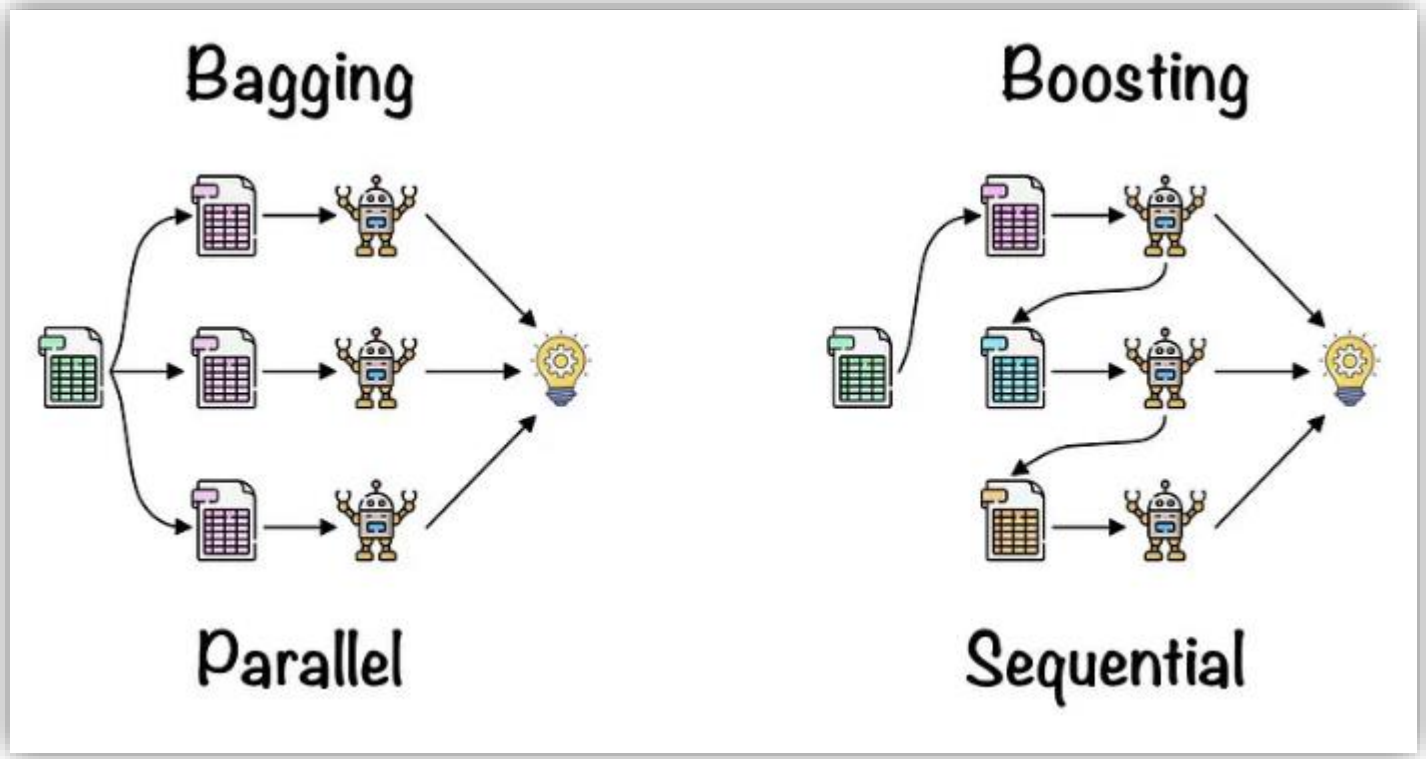
Ensemble - Model

- **배깅(Bagging)**
  - Bootstrap Aggregating
  - 여러 개의 모델 독립적으로 학습 -> 각 모델의 평균 또는 투표(Voting)로 결과 통합
  - 훈련 데이터셋에서 여러 번의 복원 추출(Bootstrap)을 통해 다양한 훈련 데이터셋 생성
  - ex) 랜덤 포레스트(Random Forest)

- **부스팅(Boosting)**
  - 여러 약한 학습기를 순차적으로 학습, 이전 학습기가 잘못 예측한 데이터에 더 많은 가중치를 두어 오류를 줄여 나가는 방식.
  - ex) AdaBoost와 Gradient Boosting

Ensemble

Ensemble - Model



Ensemble

Ensemble - Model

- **스태킹(Stacking)**
  - **서로 다른 모델들의 예측을 새로운 데이터로 사용**하여, 최종적인 예측을 위한 또 다른 모델 (**메타 학습기**)을 학습시키는 방법.
  - **서로 다른 유형의 모델을 결합,** 더 강력한 최종 모델 생성

Ensemble

Ensemble - Model

	배깅(Bagging)	부스팅(Boosting)	스태킹(Stacking)
학습 방식	병렬(Parallel)	순차(Sequential)	메타 모델 사용
목적	분산(Variance) 감소	편향(Bias) 감소	모델 조합 최적화
개별 모델	독립적 학습	이전 모델의 결과 반영	서로 다른 알고리즘 가능
알고리즘	랜덤 포레스트	XGBoost, LightGBM	StackingClassifier

- **배깅(Bagging) : 다양한 훈련 데이터셋 생성**
- **부스팅(Boosting) : 학습기를 순차적으로 학습**
- **스태킹(Stacking) : 서로 다른 모델들의 예측을 새로운 데이터로 사용**

Ensemble

Ensemble - Model

3. 스태킹(Stacking)

- **서로 다른 모델들의 예측을 새로운 데이터로 사용**하여, 최종적인 예측을 위한 또 다른 모델(**메타 학습기**)을 학습시키는 방법.

앙상블 모델의 장점

- ✓ **개별 모델이 가지는 편향과 분산을 줄이고,**
- ✓ **과적합(Overfitting)의 위험을 감소,**
- ✓ **전반적인 예측 성능을 향상**

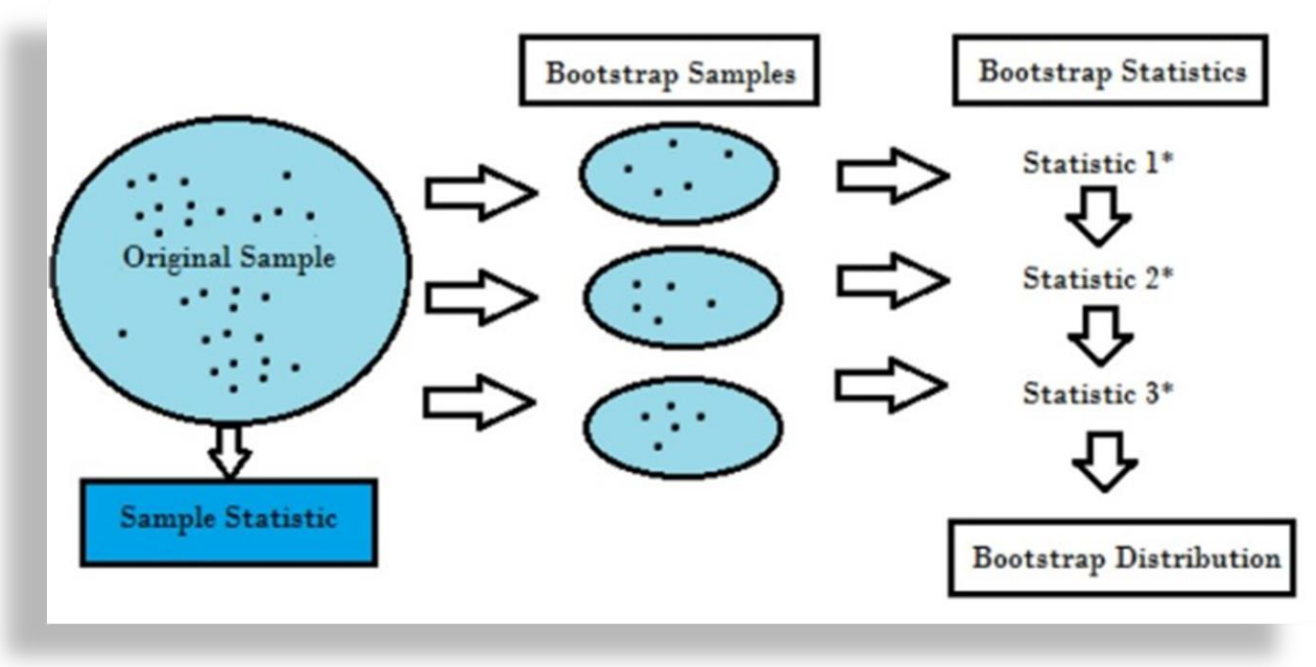
다양한 모델의 예측을 결합함으로써,

데이터의 복잡한 구조와 패턴을 더 잘 포착 하여 더 높은 신뢰도를 지닌 예측 제공

Ensemble

Bagging Bootstrap Aggregating 가죽손잡이

- **부트스트랩 : 무작위 복원 추출**
  통계적 추정을 위한 재표집(resampling) 기법
  원래의 데이터 세트에서 반복하여 샘플을 추출하는 방법



통계량의 변동성 추정

Ensemble

Bagging Bootstrap Aggregating 가중손잡이

- **부트스트랩 : 무작위 복원 추출**

1.샘플 추출:

> 원본 데이터 세트에서 n개의 데이터 포인트를 무작위로 선택

(n 원본 데이터 세트의 크기)

> '복원 추출' 방식 사용

한 데이터 여러 번 선택 or 전혀 선택되지 않을 수도

2. 부트스트랩 표본 생성:

> 1의 샘플 추출 과정을 R번 반복, R개의 부트스트랩 표본 생성

> R은 충분히 큰 수(일반적으로 수 백에서 수천 번 사이 수행)

> 각 반복마다 독립적으로 샘플을 추출

3. 데이터 분석:

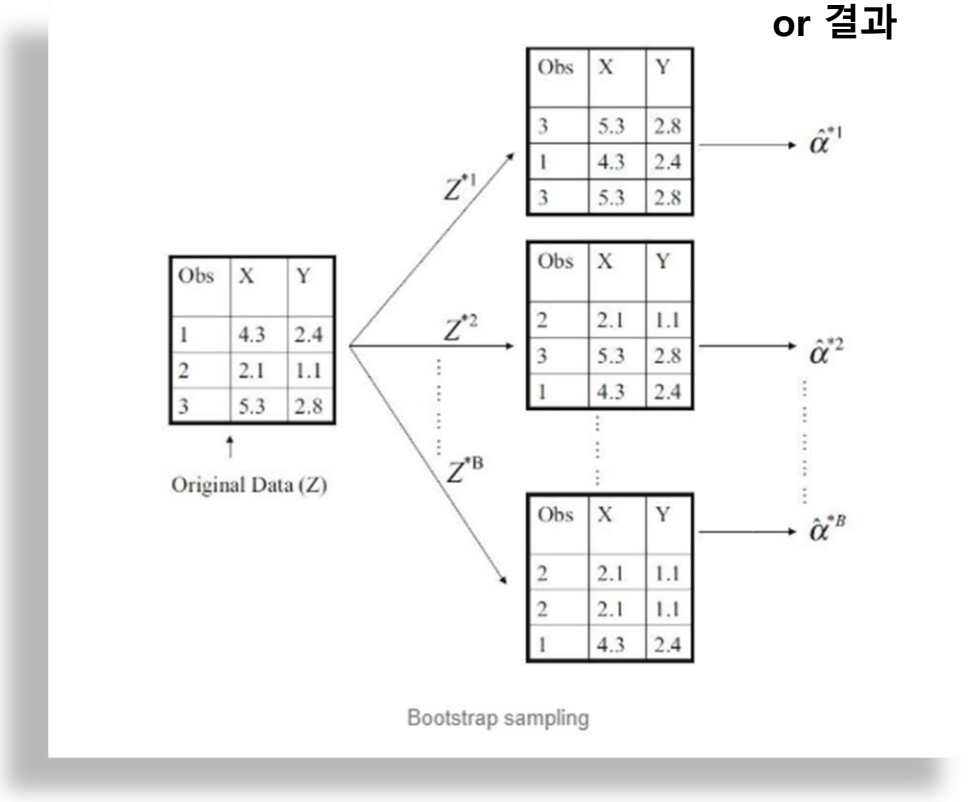
> 생성된 각 부트스트랩 표본에 대해 분석 작업 수행

ex 통계적 추정, 모델 학습, 성능 평가 등

4. 결과 집계 및 해석:

> 모든 부트스트랩 표본에 대한 분석 결과를 집계하고 해석

다양한 통계량 or 결과

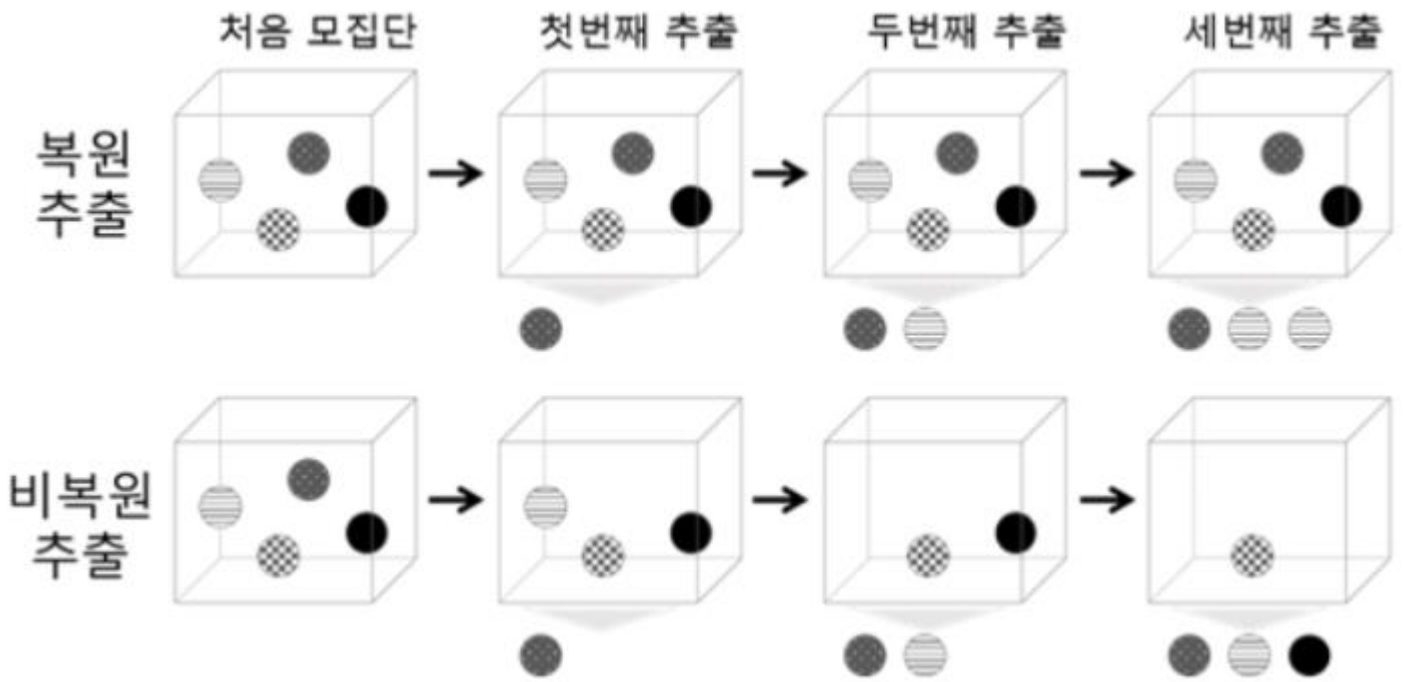


Ensemble

Bagging Bootstrap Aggregating 가중손잡이

- **부트스트랩 : 무작위 복원 추출**

복원추출 *vs* **비복원추출**

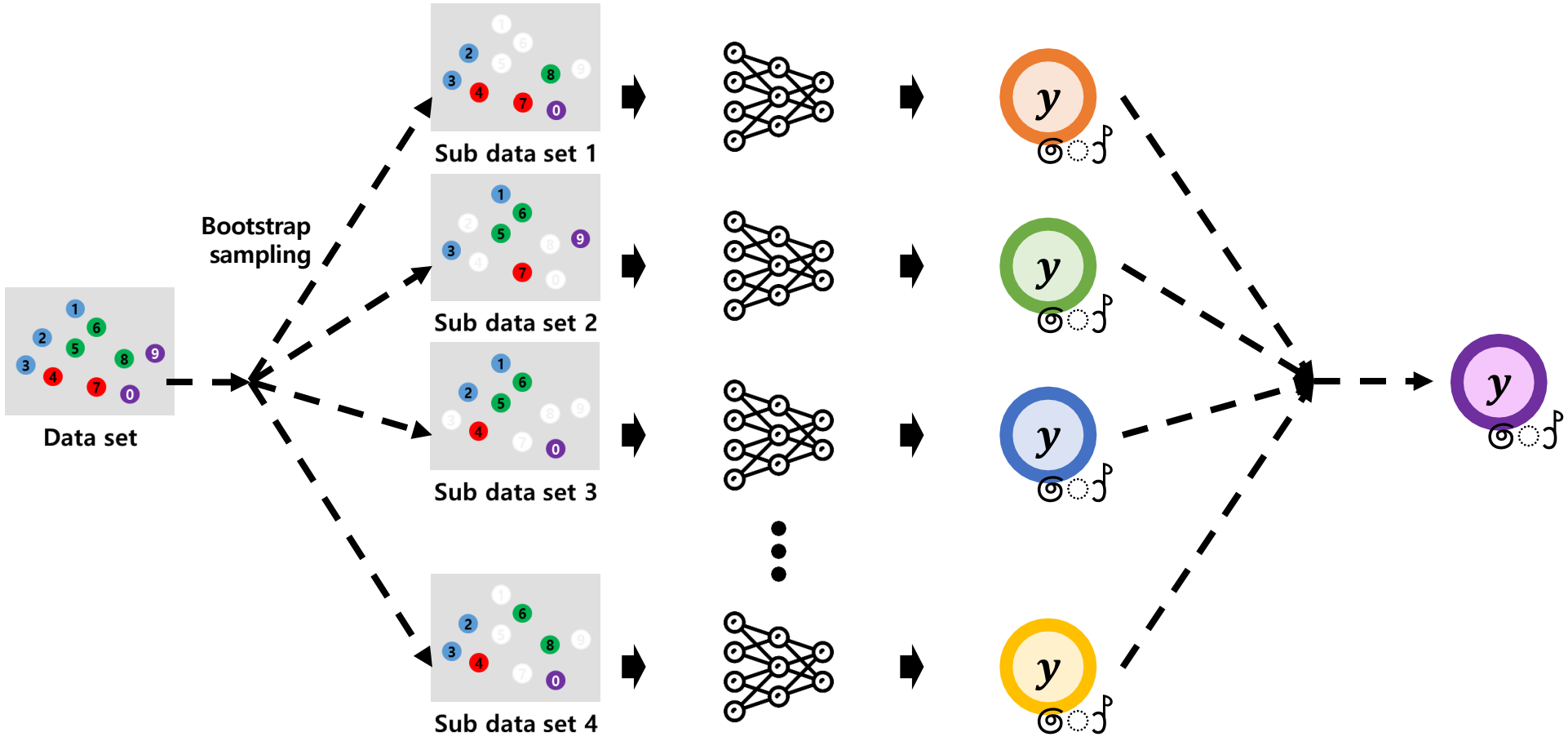


Ensemble

Bagging Bootstrap Aggregating 가중손잡이

- **부트스트랩 : 무작위 복원 추출**

"Bootstrap Aggregating" = **부트스트랩(Bootstrap) 샘플링 + 앙상블 학습의 집계(Aggregating)**

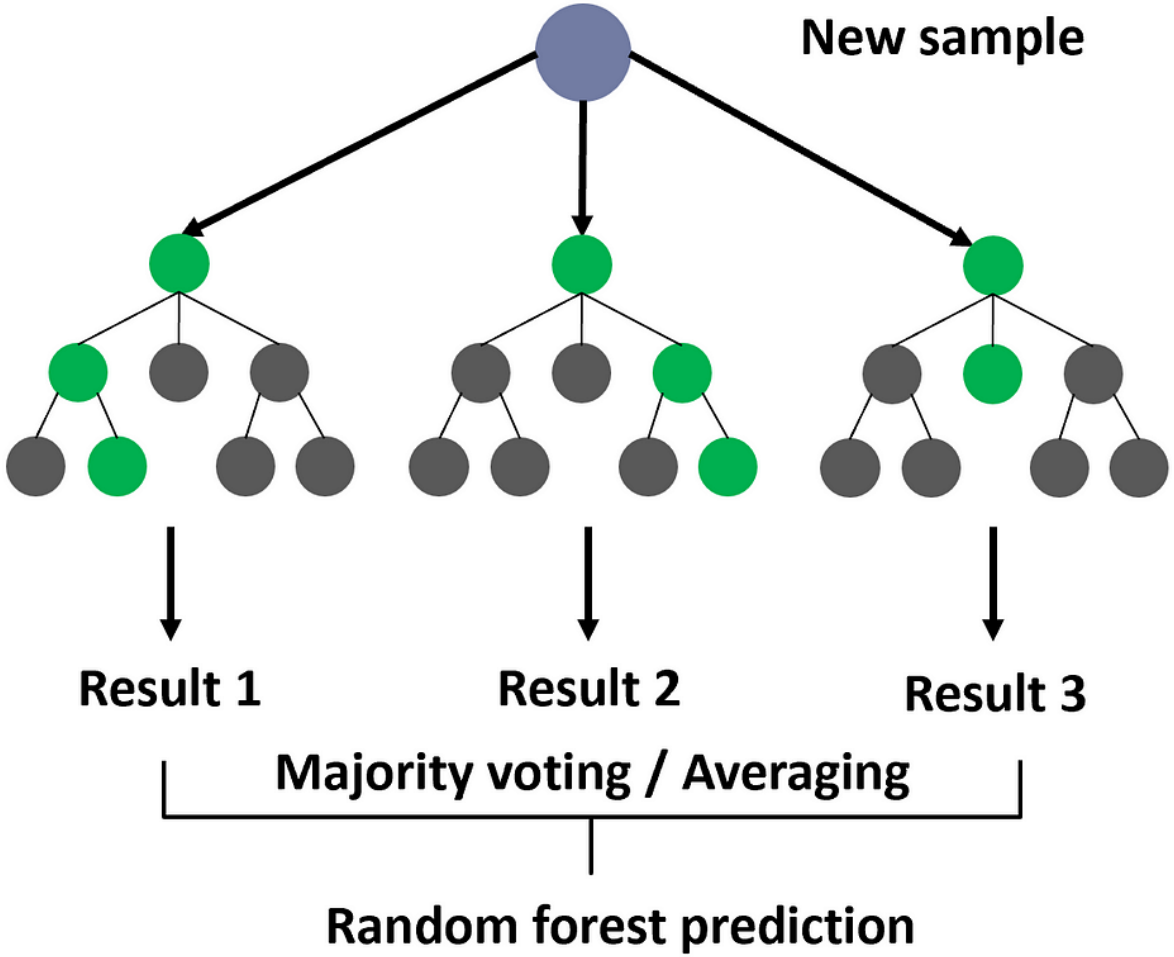


Bootstrap sampling 을 기반으로 서로 다른 sub data set을 확보 → 서로 다른 sub data set으로 각각의 base learner 학습

Ensemble

Bagging Bootstrap Aggregating 가중손잡이

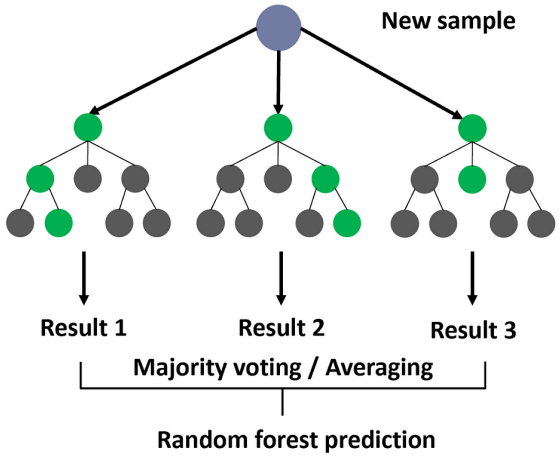
Random Forest



Ensemble

Bagging Bootstrap Aggregating 가중손잡이

Random Forest



- 포레스트 : 숲(Forest)
- 결정 트리 : 나무(Tree)
- 나무가 모여 숲을 이룸 = 결정 트리(Decision Tree)가 모여 랜덤 포레스트(Random Forest)를 구성

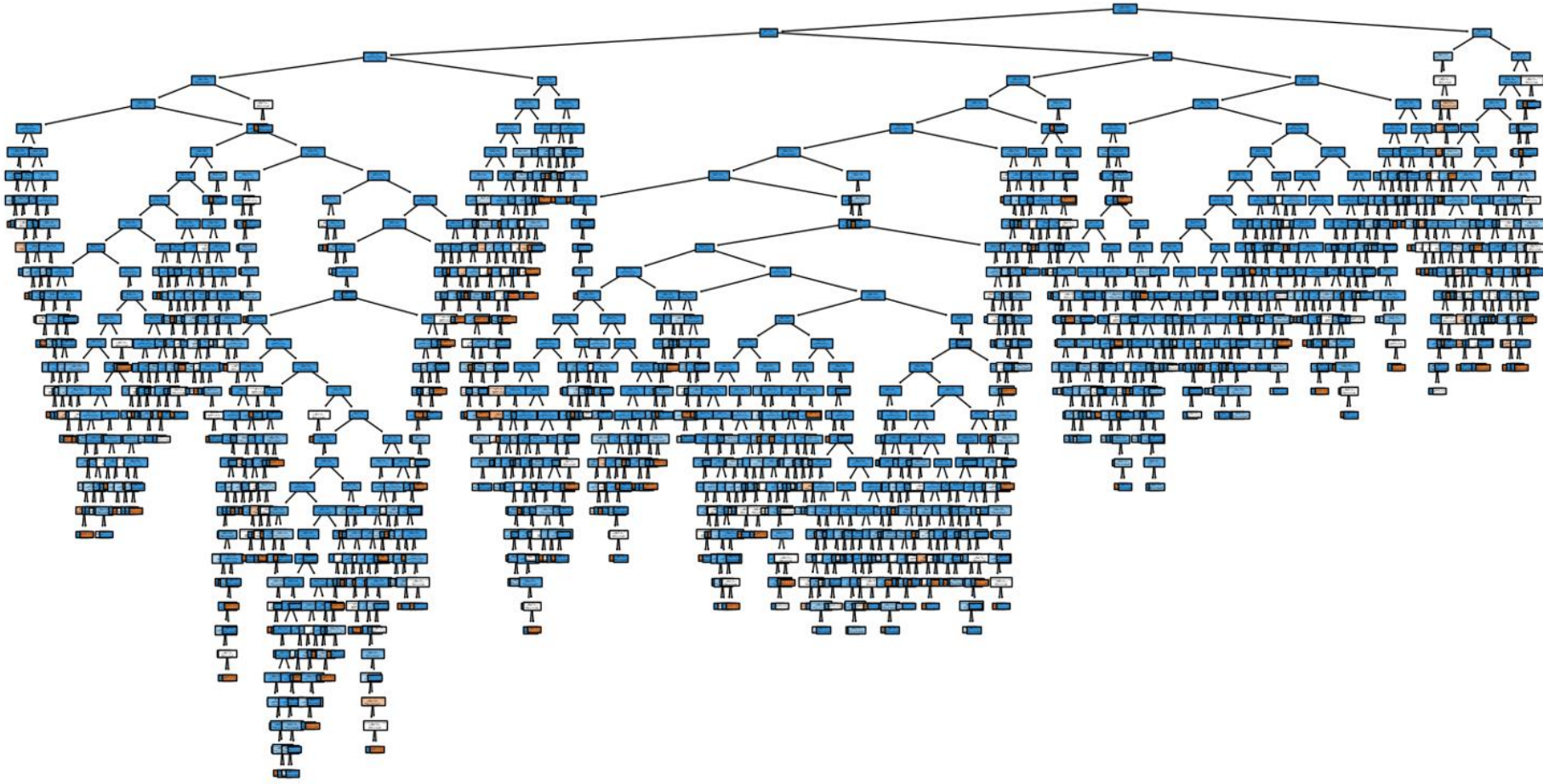
- 결정 트리의 단점 : 훈련 데이터에 오버피팅이 되는 경향

  -> 여러 개의 결정 트리를 통해 랜덤 포레스트 구성

Ensemble

Bagging Bootstrap Aggregating 가중손잡이

Random Forest



Ensemble

Bagging Bootstrap Aggregating 가중손잡이

Random Forest

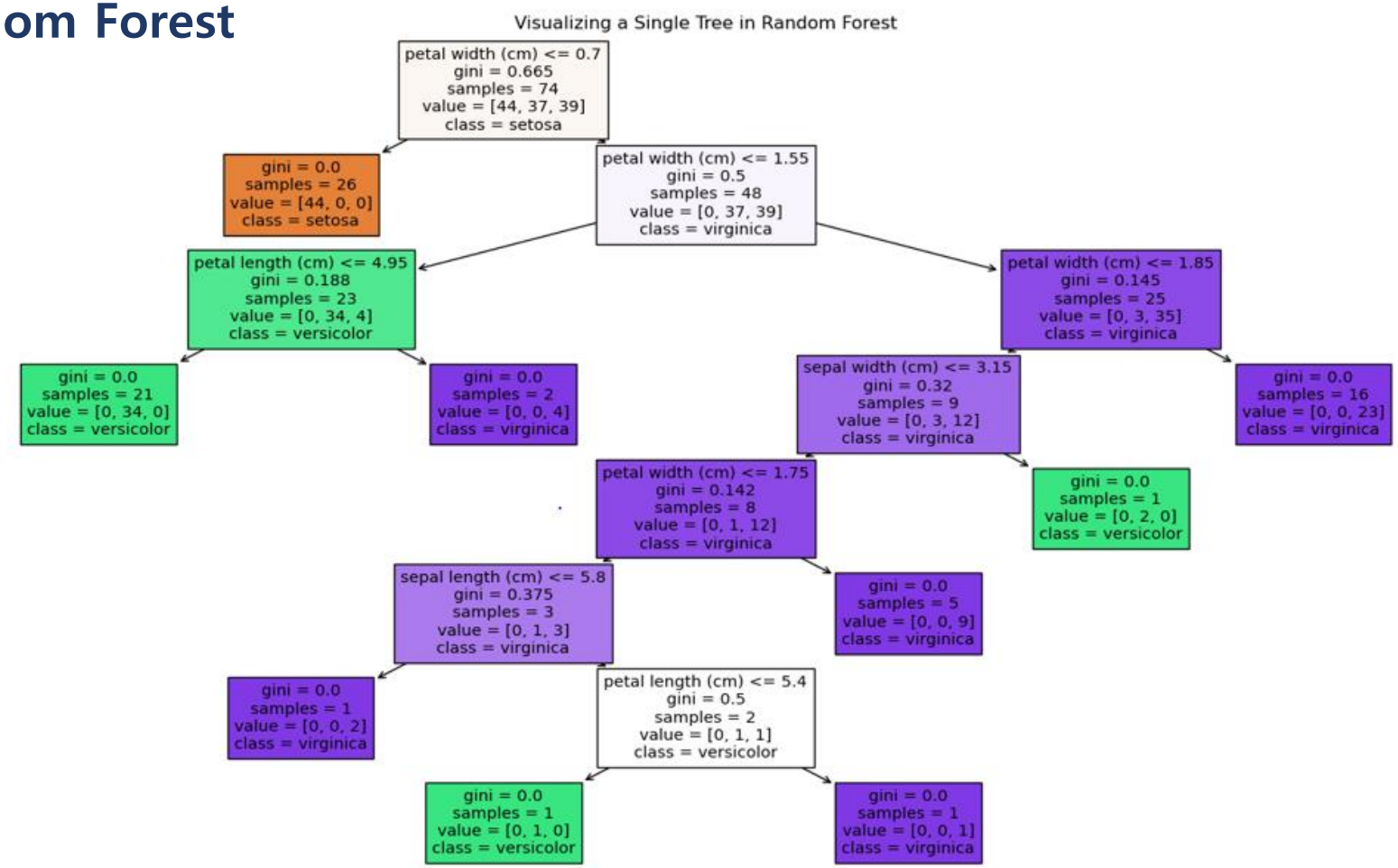
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns; sns.set() # For a nicer plot style

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42) # 100 trees in the forest

# Fit the model with the training data
rf_clf.fit(X_train, y_train)

# Predict the test dataset
predictions = rf_clf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy*100:.2f}%")

# Feature Importance
feature_importances = rf_clf.feature_importances_
features = iris.feature_names
plt.figure(figsize=(10, 6))
plt.barh(range(len(features)), feature_importances, align='center')
plt.yticks(range(len(features)), features)
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Feature Importance in Random Forest Classifier')
plt.show()
exit()
```

n_estimators: 랜덤 포레스트 안의 결정 트리 개수
max_features: 무작위로 선택할 Feature의 개수

Ensemble

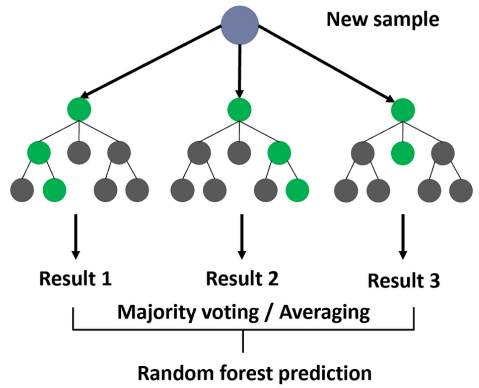
Bagging Bootstrap Aggregating 가죽손잡이

Random Forest



Ensemble

Bagging Bootstrap Aggregating 가죽손잡이



Random Forest

- If 30개의 Feature를 기반으로 하나의 결정 트리 생성
  많은 가지 -> 오버피팅
- **30개의 Feature 중 랜덤으로 5개의 Feature만 선택**, 하나의 결정 트리 구성
- 또 30개 중 랜덤으로 5개의 Feature를 선택해서 또 다른 결정 트리를 구성
- 반복... -> 여러 개의 결정 트리 구성.
- 결정 트리 하나마다 예측 값 산출
- 여러 결정 트리들이 내린 예측 값들 중 가장 많이 나온 값을 최종 예측값 선정
- 다수결의 원칙 -> **앙상블(Ensemble) (의견을 통합하거나 여러 가지 결과를 합치는 방식)**

요약 :

- 하나의 거대한 깊이가 결정 트리를 만드는 것이 아니라 여러 개의 작은 결정 트리를 만듦
- 여러 개의 작은 결정 트리가 예측한 값들 중 가장 많은 값(분류일 경우)
  혹은 평균값(회귀일 경우)을 최종 예측 값으로 결정

Ensemble

- 실습

  3.01.RF.ipynb

  3.01.bagging.KNN.ipynb

  3.01.bagging.DT.ipynb

  (과제) 3.01.bagging.hr.p.ipynb

  3.01.bagging.hr.ipynb

Ensemble

- **실습**

  3.01.bagging.hr.ipynb

```
predictions = knn_bagging_clf.predict(X_test)
accuracy = accuracy_score(y_test, predictions)

print(accuracy)
```

```
0.8396739130434783
```

```
Bagging with Decision Tree Accuracy: 1.0
```

```
Bagging with SVM Accuracy: 0.8695652173913043
```

<u>최적의 모델 선택 ???</u>

Ensemble

부스팅(Boosting)

- 약한 학습 알고리즘을 강한 학습 알고리즘으로 변환하는 프레임워크.
- 여러 개의 약한 학습기를 **순차적으로** 학습시키면서 이전 학습기가 **잘못 분류한 샘플에 더 많은 가중치를 부여**하여 학습 진행 <- 전체 모델의 성능을 점진적으로 향상시키는 데 기여

▪ **아다부스트(AdaBoost, Adaptive Boosting)**

- 각 반복에서 약한 학습기를 학습시키고, 학습된 모델의 성능에 따라 학습 데이터의 **가중치를 조정**
- 잘못 분류된 샘플은 더 높은 가중치를 받아, 다음 학습기가 이 샘플들을 더 정확하게 분류하도록 유도
- **각 학습기의 훈련 데이터는 이전 학습기의 성능에 기반하여 결정**, 새로운 데이터 세트는 무작위 생성 x
  이전 학습기들이 **잘못 분류한 샘플들을 포함** <- 이전 학습기들의 오류를 보완, 전체 모델의 성능 향상
- 데이터의 복잡한 패턴을 효과적으로 학습할 수 있도록 돕고 과적합에 비교적 강함
- 다양한 유형의 기반 학습기와 결합될 수 있어, 분류 및 회귀 문제 모두에 적용 가능(유연성)

Ensemble

부스팅(Boosting)

약한 학습 알고리즘들을 순차적으로 학습시켜, 각각의 약점을 보완하여 강한 학습 알고리즘을 만드는 앙상블 기법

- **Process**

1.초기화: 모든 학습 데이터에 동일한 가중치를 부여

2. 반복적 학습:

1) 약한 학습기를 순차적으로 학습

각 학습기는 **<u>이전 학습기가 잘못 분류한 샘플에 더 많은 주의</u>**를 기울여 학습 진행

2) 학습된 각 모델은 그 성능에 따라 가중치가 부여

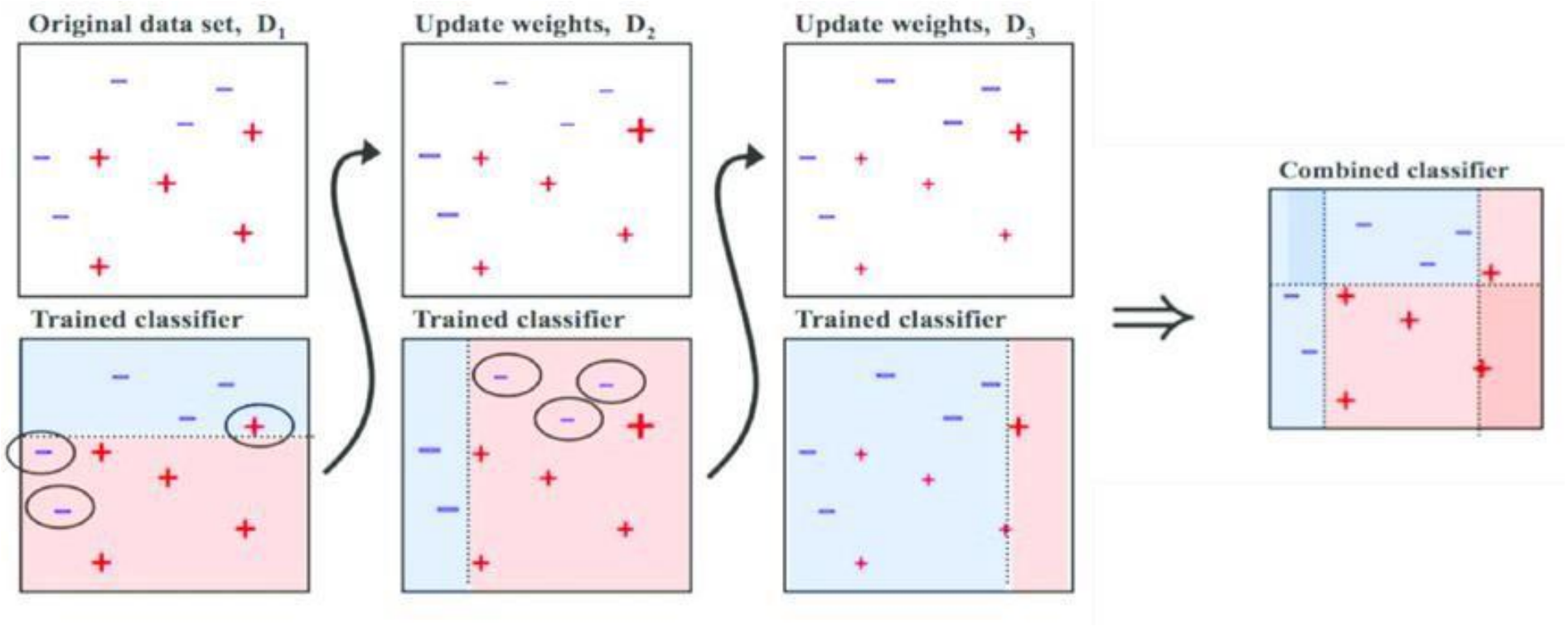
잘못 분류된 샘플은 다음 모델이 학습할 때 더 높은 가중치를 받게 되어, 모델이 이 샘플을 올바르게 분류하도록 유도

3) 지정된 수의 모델이 학습될 때까지 또는 더 이상 성능 개선이 없을 때까지 반복

3. 결합: 최종 모델은 여러 개의 약한 학습기에 부여된 가중치를 바탕으로 결합, 강한 예측 성능을 가진 모델 형성.

Ensemble

부스팅(Boosting)

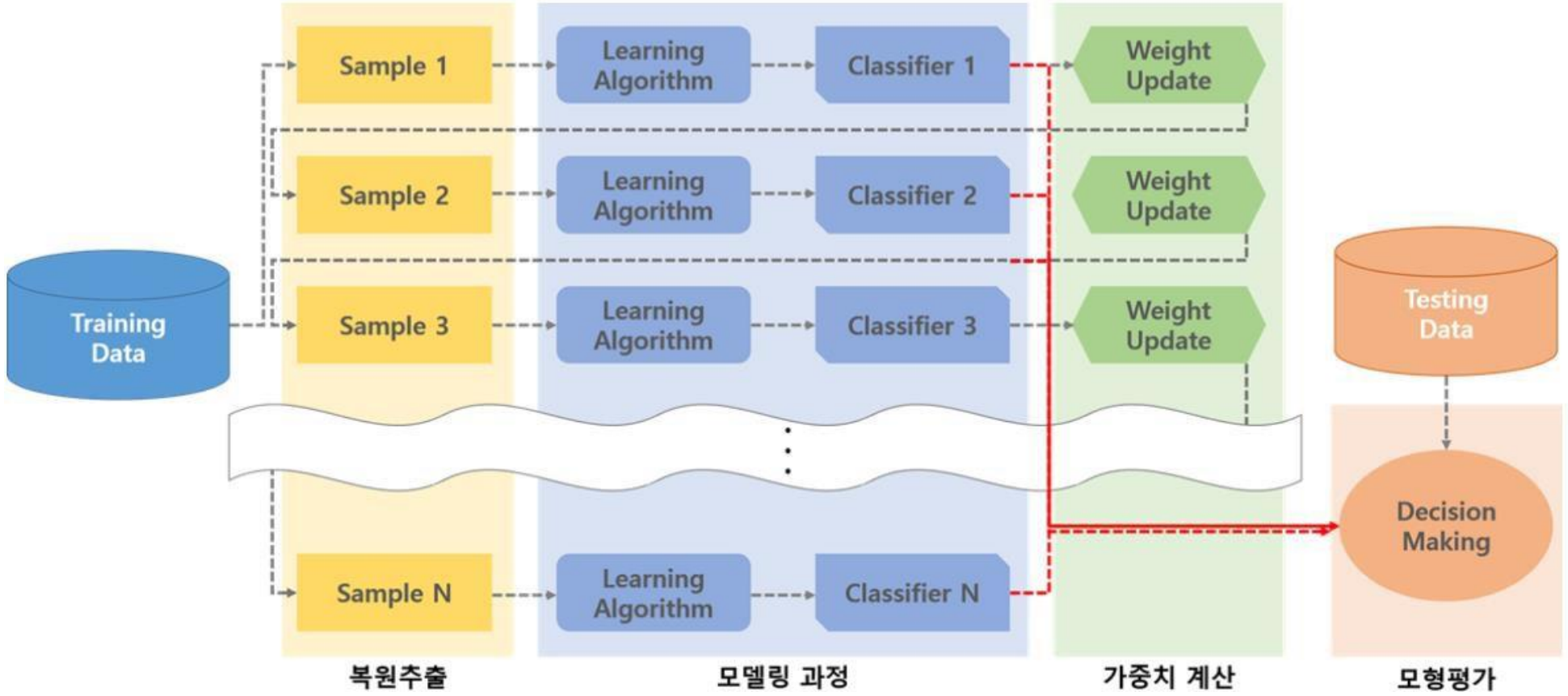


① 첫번째 base learner의 오분류 데이터에 가중치를 부여하여 Sampling 후, 두번째 base learner 학습
② 두번재 base learner의 오분류 데이터에 가중치를 부여하여 Sampling 후, 세번째 base learner 학습
③ n-1번째 base learner의 오분류 데이터에 가중치를 부여하여 Sampling 후, n번째 base learner 학습

Ensemble

부스팅(Boosting)

가중치 업데이트(Weight Update):
각 학습 단계 후에 샘플의 가중치 업데이트
잘못 분류된 샘플들의 가중치는 증가하고, 올바르게 분류된 샘플들의 가중치는 감소(혹은 유지)
잘못 분류된 샘플들에 초점 -> 이전 학습기 오류 보정



Ensemble

부스팅(Boosting)

- **부스팅 알고리즘**

  • **AdaBoost (Adaptive Boosting)**:

  - 가장 널리 사용

  - 각 학습 단계에서 잘못 분류된 샘플에 더 많은 가중치를 부여하여 순차적으로 학습

  • **Gradient Boosting**:

  - 손실 함수의 그래디언트를 사용하여 약한 학습기를 학습

  - 연속적인 개선을 통해 모델의 성능을 향상

  • **XGBoost (eXtreme Gradient Boosting)**:

  - Gradient Boosting을 확장 알고리즘, 성능과 속도 면에서 개선

  • **LightGBM**:

  - XGBoost와 유사, 대규모 데이터셋 처리에 특화되어 있어 학습 속도가 매우 빠름

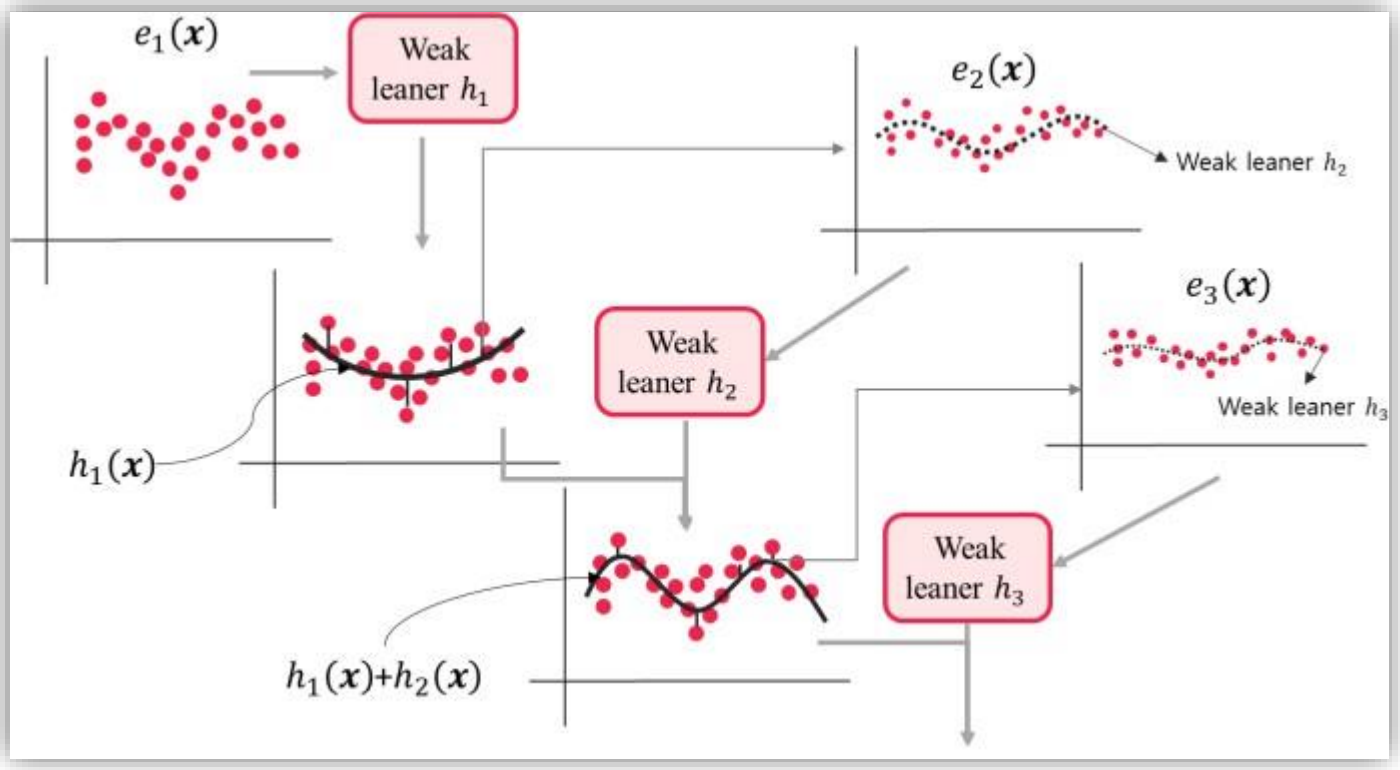
  • **CatBoost**:

  -범주형 데이터 처리에 특화, 데이터 전처리 단계를 간소화하면서 높은 성능 제공

Ensemble

부스팅(Boosting) - XGBoost

repetitive learning scheme



- 약한 학습기
- 잔여 오차 학습
- 점진적인 학습
- 강력한 예측 성능

Ensemble

부스팅(Boosting) - XGBoost

Algorithm XGBoost

Input: training set $X_{train} = \{\boldsymbol{x}_i, y_i\}, i = 1, \ldots, N$; T: number of iterations;

$\mathcal{L}$: loss function; γ: regularization term for the number of leaves.

Output: final classifier: $H(\boldsymbol{x}) = \sum_{t=1}^{T} h_t(\boldsymbol{x})$ where h_t is the induced weak classifiers.

1: $\hat{y}_i^{(0)} \leftarrow 0$ for $i = 1, \ldots, N$

2: **for** $t = 1$ to T do

3: $h_t \leftarrow \text{decision\_tree}\left(X_{train}, \hat{\boldsymbol{y}}^{(t-1)}, \mathcal{L}, \gamma\right)$

4: $\hat{y}_i^{(t)} \leftarrow \hat{y}_i^{(t-1)} + h_t(\boldsymbol{x}_i)$ for $i = 1, \ldots, N$

5: **end for**

Ensemble

부스팅(Boosting) - LightGBM

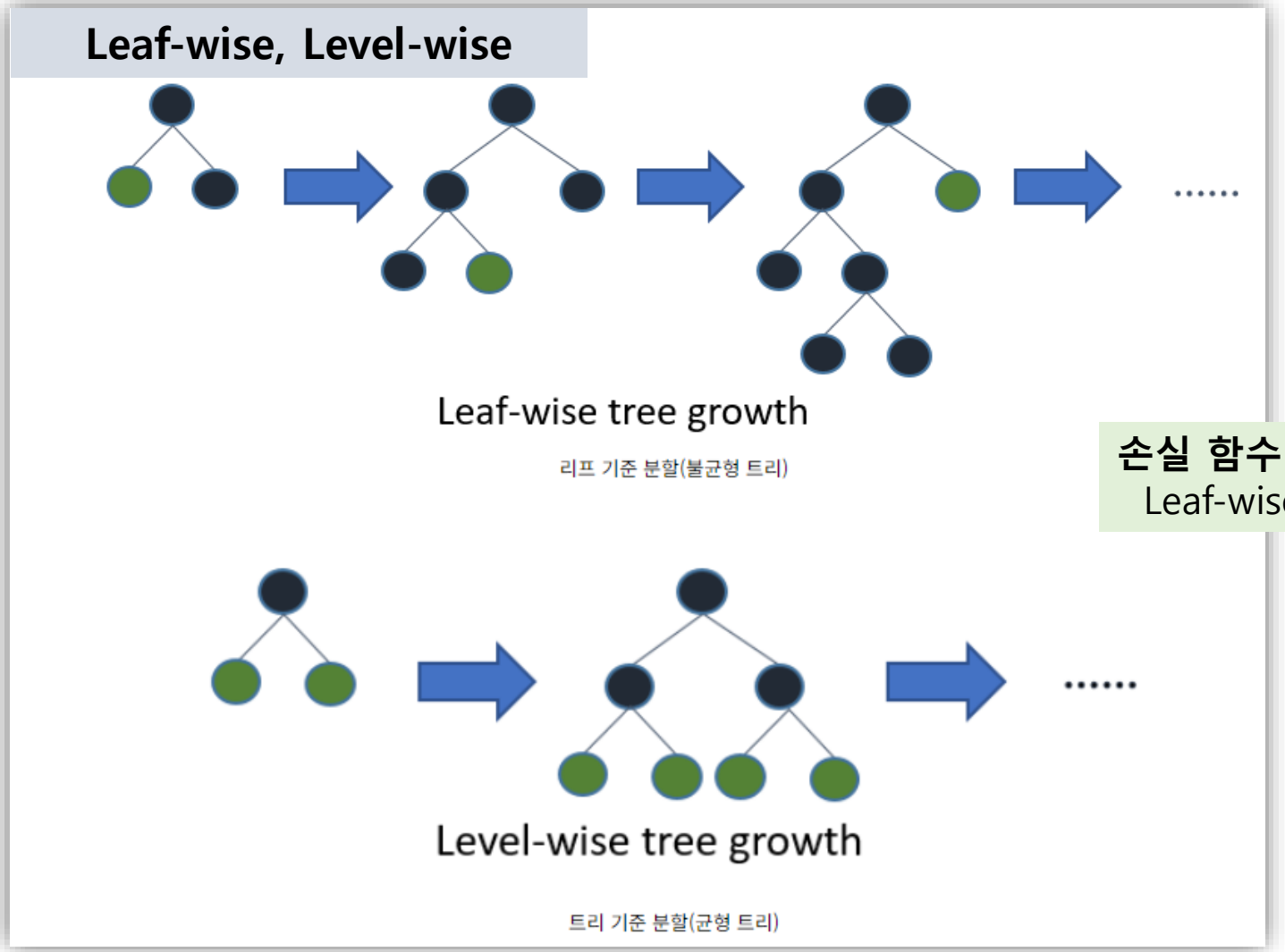
Light Gradient Boosting Machine

- 마이크로소프트에서 개발한 고성능 그래디언트 부스팅 프레임워크
- 약한 예측 모델(ex 결정 트리)을 순차적으로 학습시켜 최종적으로 강한 예측 모델을 만드는 앙상블 기법 중 하나
- 기존의 그래디언트 부스팅 시스템을 개선하여 대규모 데이터셋에서도 빠른 학습 속도와 높은 효율성 제공

- **리프 중심 트리 분할(Leaf-wise Tree Growth)**:
  - 손실을 가장 많이 줄일 수 있는 리프에서 트리를 계속 성장시키는 방식 사용
  - 모델의 정확도 향상 but 과적합(overfitting) 주의 필요

Ensemble

부스팅(Boosting) - LightGBM



손실 함수 기반 분할 *vs* **불순도(Impurity) 기반**
Leaf-wise growth Information Gain, Gini Index

Ensemble

부스팅(Boosting) - LightGBM

LightGBM 파라미터

파라미터	default	설명
num_iterations	100	반복 수행하려는 트리의 개수 (너무 크면 오버피팅 발생)
objective	regression	수치예측이면 regression, 이진분류이면 binary
learning_rate	0.1	부스팅 스탭 반복할 때 학습률, 0~1 사이의 값
max_depth	1	트리의 깊이
min_data_in_leaf	20	한 리프의 최소 데이터 수 (decision tree의 min_sample_leaf와 동일, 오버피팅 제어)
num_leaves	31	하나의 트리가 가질 수 있는 최대 리프 개수
boosting	gbdt	부스팅 방법 (gbdt: Gradient Boosting DecisionTree / rf: RandomForest)
bagging_fraction	1.0	데이터 샘플링 비율, 오버피팅 제어
feature_fraction	1.0	개별 트리 학습 시 무작위로 선택하는 feature의 비율
lambda_l1	0.0	L1 regulation 제어
lambda_l2	0.0	L2 regulation 제어
metric	""	성능평가를 어떤 것으로 할 것인지 (auc, l1, l2 등)

Ensemble

부스팅(Boosting)

- 실습

  3.01.AdaBoost.ipynb

  [TASK] 3.01.AdaBoost.hr.p.ipynb

  3.01.AdaBoost.hr.ipynb

  3.01. ensemble.boosting.GB.ipynb

  3.01. ensemble.boosting.XGB.ipynb

  3.01. ensemble.boosting.LightGBM.ipynb

Ensemble

스태킹(Stacking)

✓ 개별적인 여러 알고리즘을 서로 결합해 예측 결과를 도출

= 배깅(Bagging) 및 부스팅(Boosting)

✓ **개별 알고리즘으로 예측한 데이터를 기반으로 다시 예측을 수행**

▪ **핵심 요소**

• **기본 모델(Base Learners)**:

✓ **서로 다른 알고리즘을 사용**하는 여러 모델 사용

✓ 각 모델들은 각각 다른 관점에서 데이터 해석, 예측 수행

• **메타 모델(Meta-model)**:

✓ 기본 모델들의 예측을 기반으로 **최종 예측을 수행하는 모델**

✓ 어떤 모델의 예측을 더 중시할지 결정하는 방법을 학습

Ensemble

스태킹(Stacking)

- **스태킹 과정**

1. **훈련 데이터 분할**: 원본 훈련 데이터를 두 개 이상의 서브셋으로 분할
   일반적으로 k-겹 교차 검증 방식 사용
2. **기본 모델 훈련**: 첫 번째 서브셋을 사용하여 여러 기본 모델을 각각 독립적으로 훈련
3. **기본 모델 예측**: 두 번째 서브셋을 사용하여 훈련된 각 기본 모델로부터 예측을 수행
   이 예측들은 메타 모델의 훈련 데이터로 사용
4. **메타 모델 훈련**: **기본 모델들의 예측을 입력**으로, 원래의 타깃 레이블을 출력으로 사용하여
   메타 모델을 훈련
5. **최종 예측 수행**: 테스트 데이터에 대해 기본 모델들의 예측을 수집하고, 이를 메타 모델에
   입력하여 최종 예측 생성

Ensemble

스태킹(Stacking)

장점:

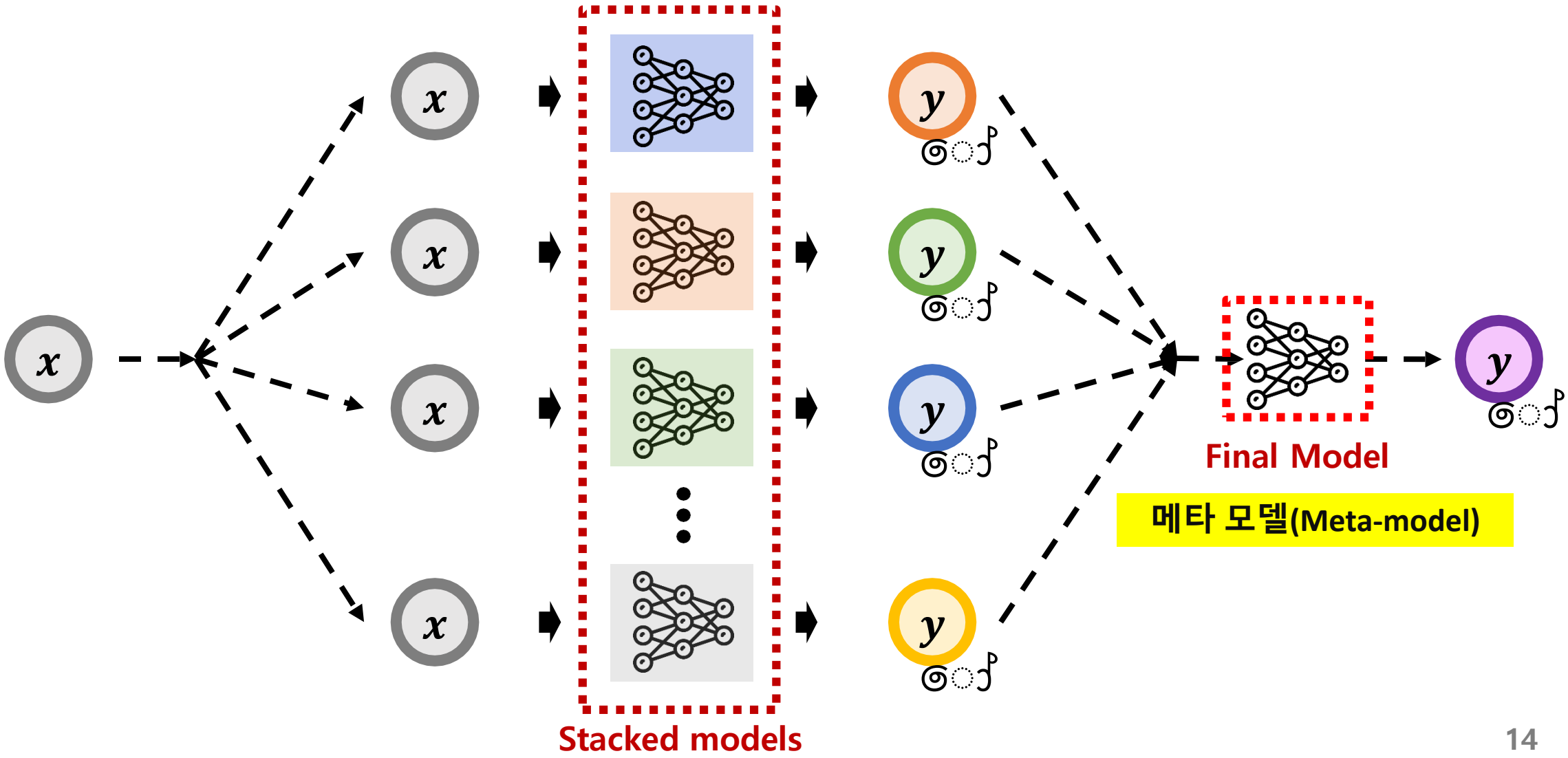
- 다양한 모델의 강점을 결합하여 보다 정확한 예측 수행
- 기본 모델들 사이의 상관관계를 줄이고, 모델의 다양성을 증가
- 복잡한 문제와 데이터 세트에 대해 뛰어난 일반화 성능 제공

•**단점**:

- 구현과 튜닝이 상대적으로 복잡하고 시간이 많이 소요
- 과적합의 위험이 있으며, 특히 메타 모델이 복잡할 경우 이 위험 증가
- 모델 학습과 예측 시간이 길어질 수 있음

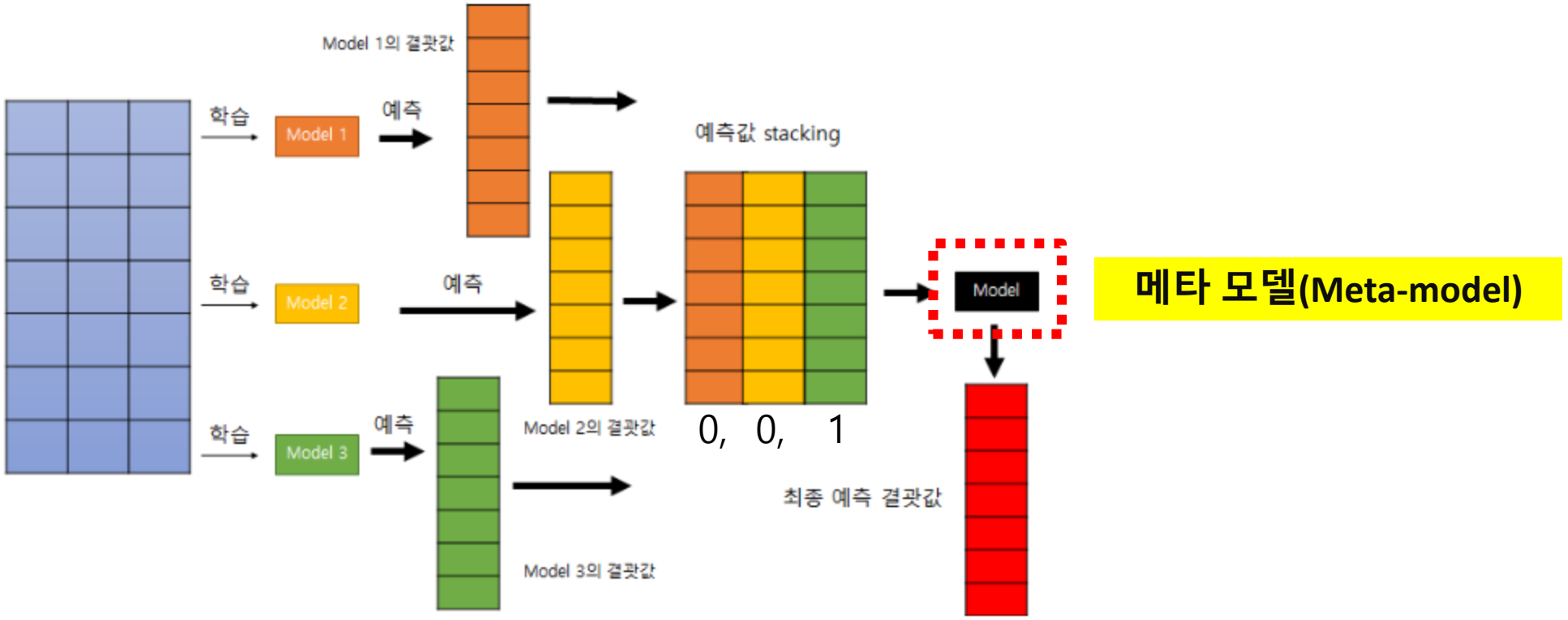
Ensemble

스태킹(Stacking)



Ensemble

스태킹(Stacking)



Ensemble

스태킹(Stacking)

- **실습**

  3.01.ensemble.stacking.ipynb

  (과제) 3.01.stacking.hr.p.ipynb

ID	모델1-예측	모델2-예측	모델3-예측
1	0.8	0.5	0.8
2	0.2	0.3	0.7
3	0.7	0.2	0.7
4	0.3	0.7	0.7
5	0.9	0.3	0.7

Logistic Regression 예측 | Decision Tree 예측

0 | 0

1 | 1

2 | 1

Ensemble

Ensemble - Sum-up

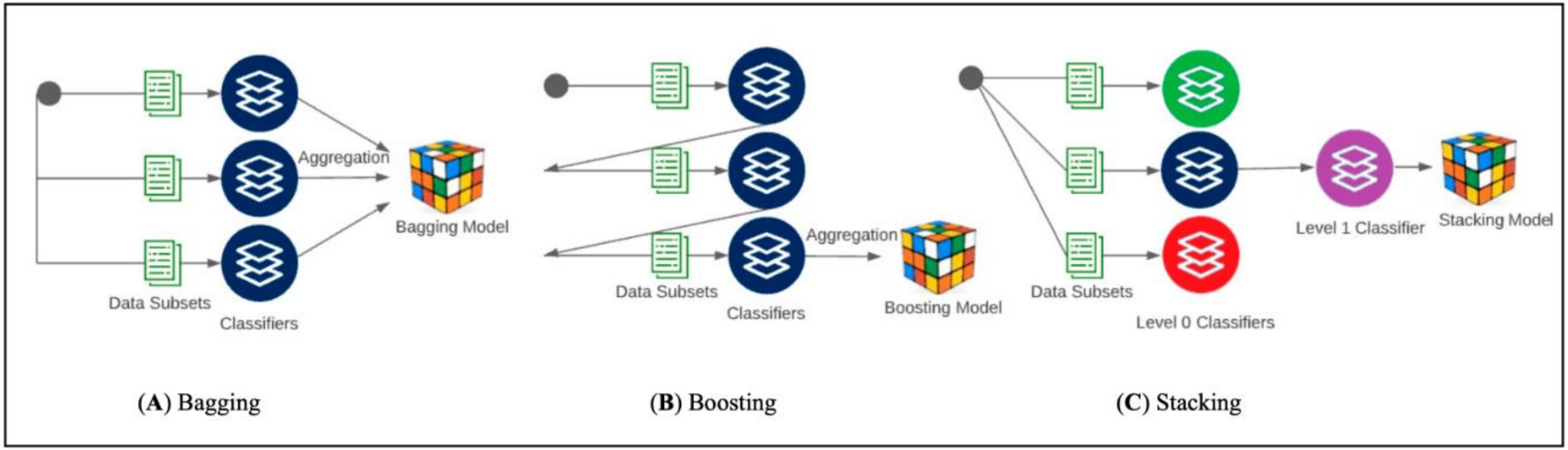
	Bagging	Boosting	Stacking
Purpose	Reduce Variance	Reduce Bias	Improve Accuracy
Base Learner Types	Homogeneous	Homogeneous	Heterogeneous
Base Learner Training	Parallel	Sequential	Meta Model
Aggregation	Max Voting, Averaging	Weighted Averaging	Weighted Averaging

https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/

Ensemble

Ensemble - Sum-up

https://velog.io/@jk01019/ensemble



- **동일한 알고리즘의 여러 인스턴스 사용**: 훈련 데이터 세트에서 무작위로 여러 서브세트 생성 각 서브세트에 대해 동일한 유형의 분류기를 독립적으로 훈련
- **병렬 처리**: 각 분류기는 서로 독립적으로 훈련, 분류기들의 결과를 평균 or 다수결로 결합, 최종 예측 수행

- **단계적 학습**: 약한 학습기(Weak Learner)를 순차적으로 훈련, 각 단계에서 이전 학습기의 오차를 줄이는 방향으로 학습 진행
- **동일한 유형의 모델 사용**: 각 단계에서 데이터에 가중치를 적용하여 모델을 순차적으로 개선

- **다양한 모델 조합**: 여러 다른 유형의 모델 (ex: DT, SVM 등)을 조합하여 사용
- **메타 모델 사용**: 기본 모델들의 예측 결과를 새로운 입력 데이터로 사용.메타 모델은 기본 모델들의 예측을 최적으로 결합하는 방법 학습

Ensemble

Diversity of Base Learners

③ Manipulating the output targets (출력 타겟 조정)
→ Label Transformation, Target Smoothing,
Class Imbalance Adjustment

1. Label Transformation (레이블 변환)
- 분류 문제에서 **출력 레이블**을 변형 -> 각 모델이 다른 패턴 학습
  ex) 다중 클래스 분류, **One-vs-Rest** 또는 **One-vs-One** 방식 사용
- 다양한 학습기가 동일한 데이터를 다르게 학습하도록 유도 -> 앙상블 성능 향상

2. Target Smoothing (타겟 스무딩)
- 회귀 문제에서 **출력 값**을 **부드럽게(Smooth)** 만드는 방법
  -> 모델이 과도하게 특정 값에만 집중하는 것 방지 -> 더 안정적인 예측 수행
  ex, **타겟 값**에 **노이즈** 추가, 전체 타겟 분포의 경계를 부드럽게 처리

3. Class Imbalance Adjustment (클래스 불균형 조정)
- **클래스 불균형**이 있는 경우, **각 클래스의 중요도를 조정**
- **클래스 가중치** 조정 -> 불균형 문제를 해결 + 다양한 모델 학습

Ensemble

Diversity of Base Learners

③ Manipulating the output targets (출력 타겟 조정)
→ Label Transformation, Target Smoothing,
Class Imbalance Adjustment

- 실습

  3.01.Ensemble.ManipulatingOutput.ipynb

THANK YOU