2021 머신러닝 여름 스터디

Esemble, boosting

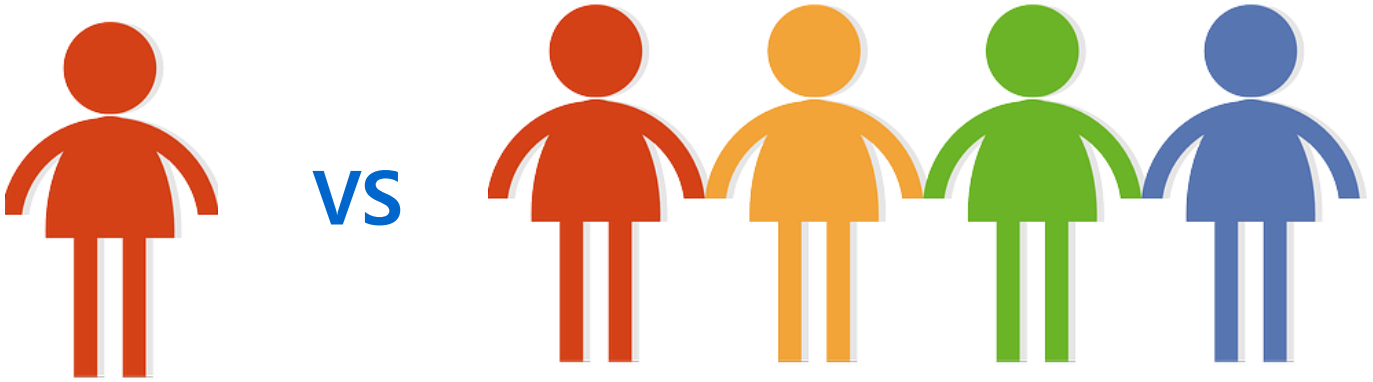
Sejong Oh
Dankook University

1. 앙상블의 개념

- Problem: 내년에 어떤 독감이 유행할지 예측해 보자
  - 한명의 전문가가 예측 vs 여러명의 전문가가 협의하여 예측

어떤 경우가 더 정확한 예측을 하는데 도움이 될까?

⇒ 대체로 여러 명이 협의하여 결정하는 것이 신뢰도가 더 높다.



한명의 결정 여러명의 협의 결정

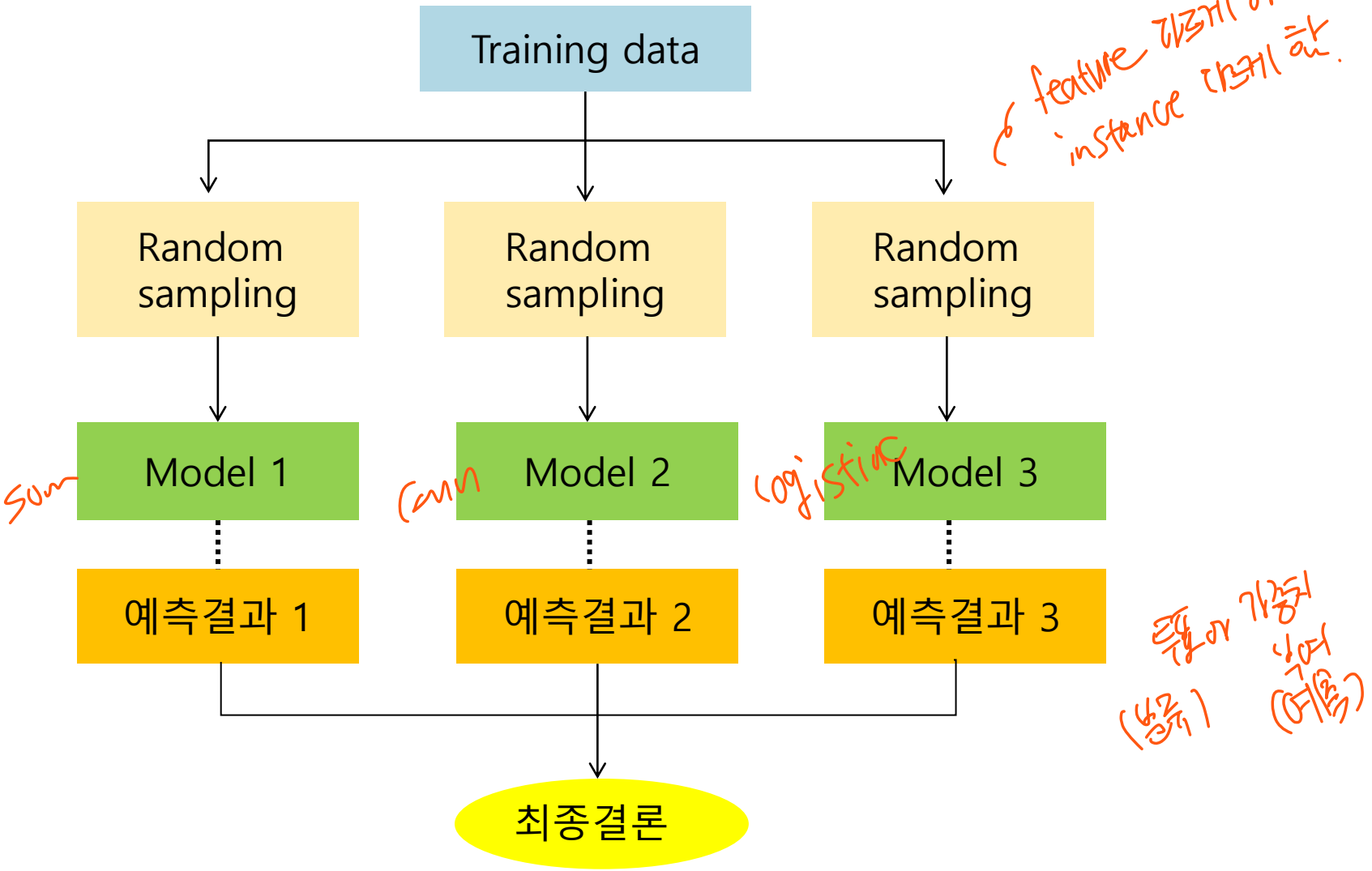
1. 앙상블의 개념

- '다수 협의에 의한 결정' 이라는 원리를 예측 문제 해결에 적용한 것이 앙상블(Ensemble)
  - 일반적으로 예측문제에는 하나의 모델을 사용
  - 여러 개의 모델을 학습시킨 뒤 그 모델들의 예측 결과들을 취합하여 최종 결정을 내린다면 예측 정확도 향상에 도움이 됨
  - 결정 트리 기반 모델들에서 많이 사용이 됨

  - 앙상블에는 크게 두가지 종류가 있음
    - 배깅(bagging)
    - 부스팅(boosting)

1. 앙상블의 개념

- 배깅(bagging)



1. 앙상블의 개념

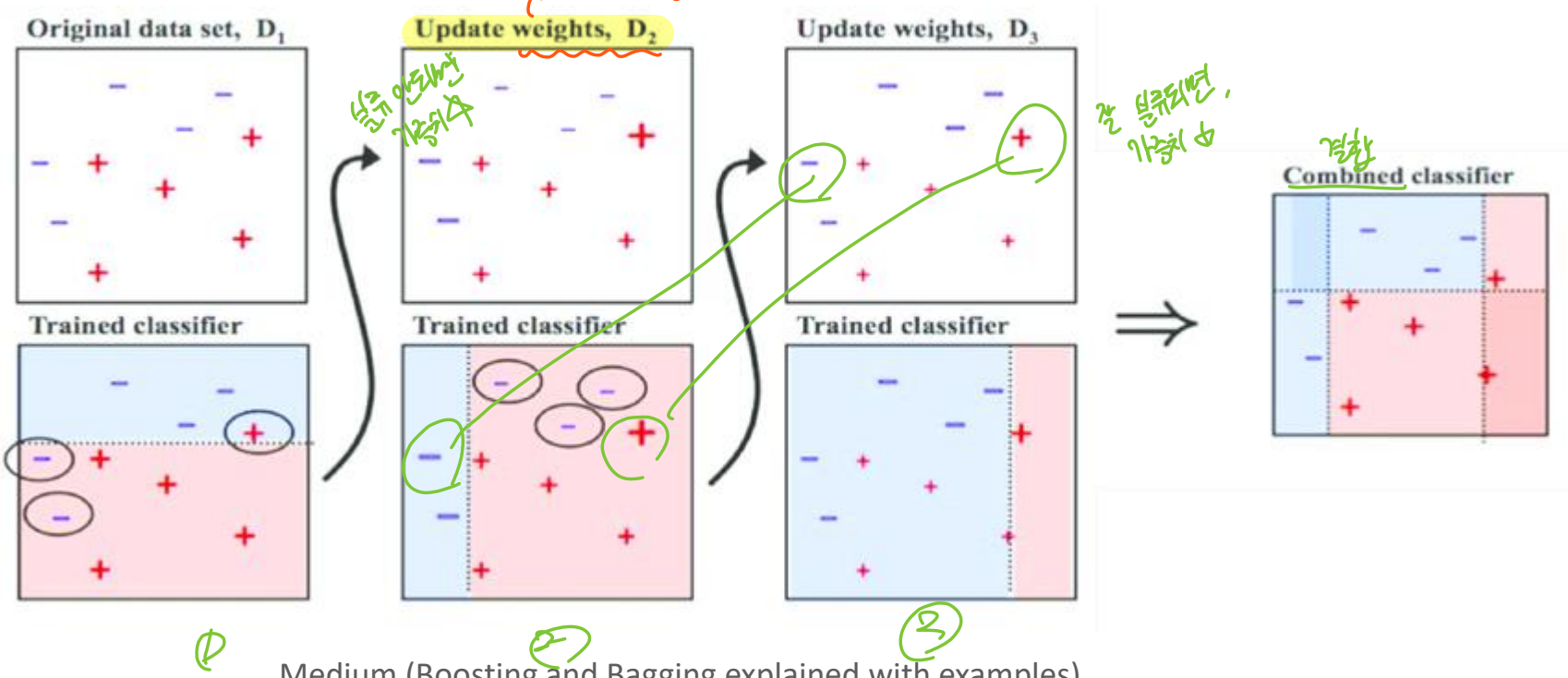
- 분류(classification) 문제에서 예측 결과 합산 방법
  - Hard voting
    - 각 분류모델이 동등하게 한표씩 행사
  - Soft voting
    - 각 분류모델의 예측값에 대한 확률을 합산하여 확률 값이 높은 클래스를 선택하는 방법

- 회귀(regression) 문제에서 예측 결과 합산 방법
  - 평균 계산

※ randomForest
: ntree(모델 개수)

1. 앙상블의 개념

- 부스팅(boosting)



Medium (Boosting and Bagging explained with examples)

1. 앙상블의 개념

- 부스팅(boosting)
  - 예측모델 1 의 결과를 보고 오답에는 가중치를 부여
  - 예측모델 2 는 예측 모델 1에의 오답 부분을 집중적으로 학습, 오답을 낮춤
  - 예측모델 3은 예측 모델 2의 오답 부분을 집중적으로 학습, 오답을 낮춤
  - 이 과정을 반복

  - 부스팅은 배깅에 비해 성능이 더 좋으나 과적합(overfitting)이 더 쉽게 발생하는 것으로 알려짐

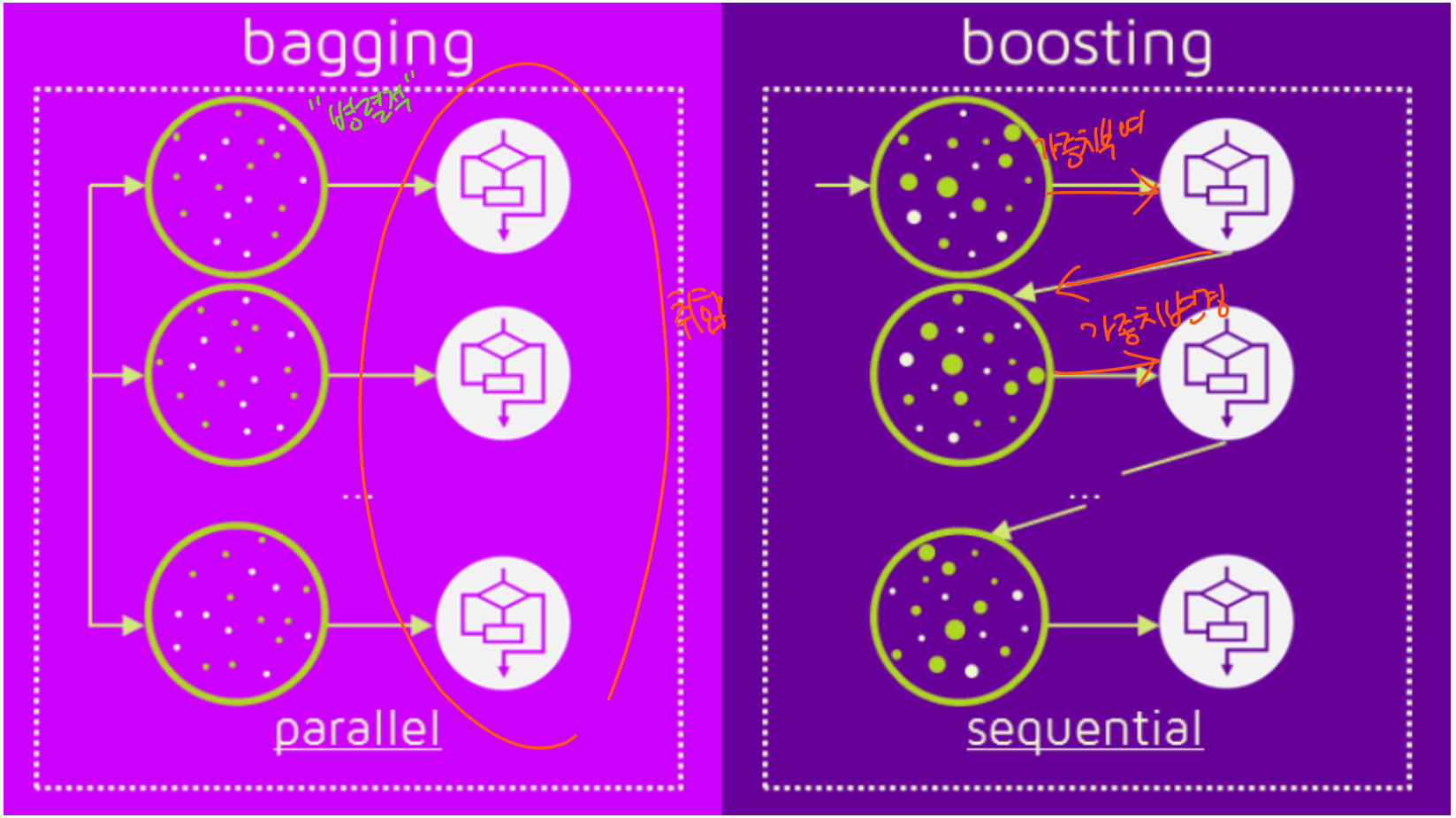
1. 앙상블의 개념

- 부스팅(boosting) in R
  - boosting {adabag} : adaboost
  - gbm {gbm} : gradient boosting
  - xgboost {xgboost}

    https://xgboost.readthedocs.io/en/latest/R-package/xgboostPresentation.html

1. 앙상블의 개념

- 배깅 vs 부스팅



출처: swallow.github.io

2. xgboost

```
library(xgboost)

# Load the data and remove NAs
data("PimaIndiansDiabetes2", package = "mlbench")
Pima <- na.omit(PimaIndiansDiabetes2)
head(Pima)

# convert factor class into 0,1
Pima$diabetes <- as.integer(Pima$diabetes)-1
head(Pima)

# Split the data into training and test set
set.seed(123)
idx <- sample(nrow(Pima), nrow(Pima)*.2)
train.data  <- xgb.DMatrix(data = as.matrix(Pima[-idx, 1:8]),
                           label = Pima$diabetes[-idx])
test.data <- xgb.DMatrix(data = as.matrix(Pima[idx, 1:8]),
                         label = Pima$diabetes[idx])
```

2. xgboost

```
# Training ##############################

model <- xgboost(data = train.data,
                 max.depth = 2,      # maximum depth of a tree
                 eta = 1,            # learning rate: (0,1]
                 nthread = 2,        # number of threads
                 nrounds = 2,        # max boosting iterations
                 objective="binary:logistic")

# Testing ###############################
pred <- predict(model, test.data)
pred <- as.numeric(pred > 0.5)

acc <- mean(pred == Pima$diabetes[idx])
acc
```

```
> head(Pima)
   pregnant glucose pressure triceps insulin mass pedigree age diabetes
4         1      89       66      23      94 28.1    0.167  21      neg
5         0     137       40      35     168 43.1    2.288  33      pos
7         3      78       50      32      88 31.0    0.248  26      pos
9         2     197       70      45     543 30.5    0.158  53      pos
14        1     189       60      23     846 30.1    0.398  59      pos
15        5     166       72      19     175 25.8    0.587  51      pos
```

```
> acc
[1] 0.7051282
```

2. xgboost

- XGBoost parameters can be divided into three categories (as suggested by its authors):
  - General Parameters: Controls the booster type in the model which eventually drives overall functioning
  - Booster Parameters: Controls the performance of the selected booster
  - Learning Task Parameters: Sets and evaluates the learning process of the booster from the given data

https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial/

2. xgboost

- Objective parameter
  - reg:linear - for linear regression [default]
  - binary:logistic - logistic regression for binary classification. It returns class probabilities
  - multi:softmax - multiclassification using softmax objective. It returns predicted class labels. It requires setting num_class parameter denoting number of unique prediction classes.
  - multi:softprob - multiclassification using softmax objective. It returns predicted class probabilities.

2. xgboost

- Other parameters
  - eta: learning rate (0,1]
    - Big values of eta result in a faster convergence and more over-fitting problems. Small values may need to many trees to converge.
  - max_depth: maximum depth of the trees (default:6)
  - sub_sample: It controls the number of samples (observations) supplied to a tree
    - Typically, its values lie between (0.5-0.8)
  - gamma: minimum reduction in the loss function
  - min_child_weight: minimum number samples (instances) in a terminal node
  - nrounds: max number of boosting iterations
  - nthread: the number of CPU threads we are going to use

[과제]

- mlbench 패키지에 포함된 Sonar 데이터셋에 대해 xgboost 로 예측 모델을 만들고 테스트 하시오.
  - Train:test = 8:2

- mlbench 패키지에 포함된 Vowel 데이터셋에 대해 xgboost 로 예측 모델을 만들고 테스트 하시오.
  - 10-fold cross validation 을 적용하여 평균 accuracy 를 구한다.

[과제]

- mlbench 패키지에 포함된 Sonar 데이터셋에 대해 xgboost 로 예측 모델을 만들고 테스트 하시오.
  - Train:test = 8:2

- mlbench 패키지에 포함된 Vowel 데이터셋에 대해 xgboost 로 예측 모델을 만들고 테스트 하시오.
  - 10-fold cross validation 을 적용하여 평균 accuracy 를 구한다.