Bagging & Boosting
실습자료

삼성 SDS

KAIST SGVR lab.

윤성의 교수 (sungeui@kaist.edu)

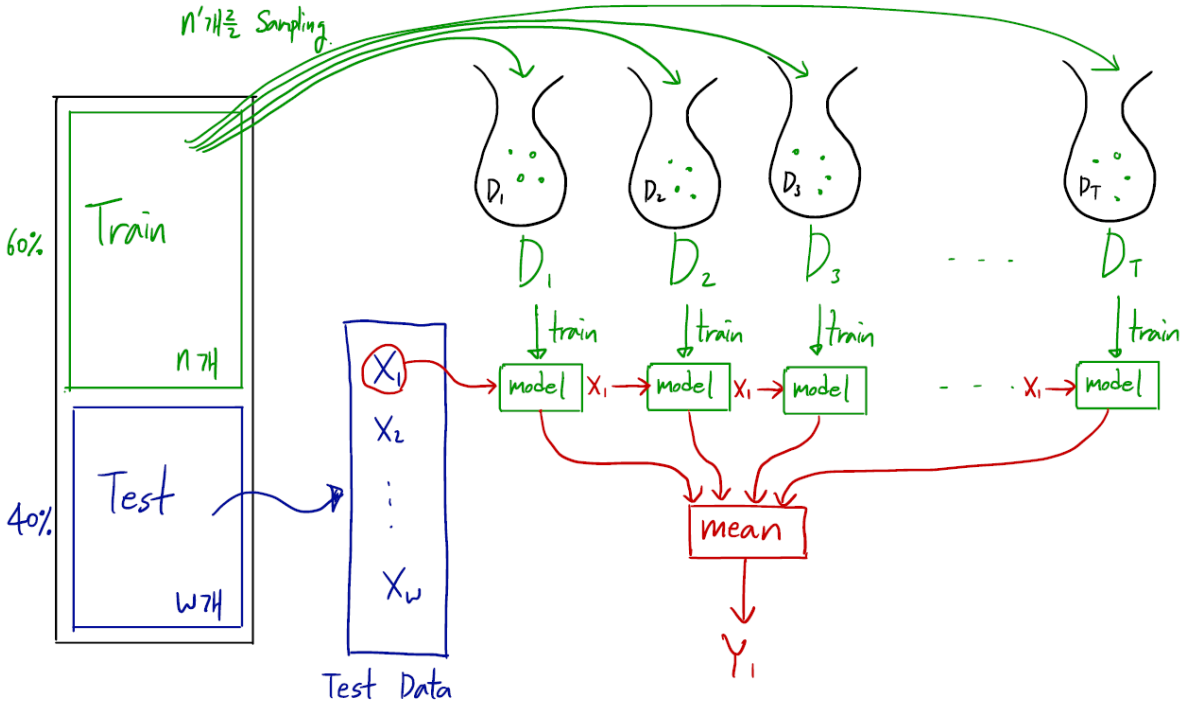
안인규 (inkyu.an@kaist.ac.kr),
신희찬 (shn4438@gmail.com)

KAIST

Bagging |

실습자료:
sgvr.kaist.ac.kr/~ikan/sds

1. 모(母) dataset을 여러 개의 sub-dataset으로 sampling한 후
2. 각각의 sub-dataset으로 여러 개의 model을 학습시켜
3. 그 평균값으로 결과를 예상하는 방법



https://www.youtube.com/watch?v=2Mg8QD0F1dQ

Bagging 실습 I

심장병 유무,
0이면 건강
1이상이면 심장병

1. 데이터 로드
2. 샘플링
3. 모델 학습
4. 예측 평균

<모(母) dataset 구조 (HeartDisease.csv 파일)>

나이, 흡연, 음주, …

1	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
2	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
3	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
4	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
5	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
6	56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
7	62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
8	57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
9	63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
10	53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
11	57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
12	56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
13	56	1	3	130	256	1	2	142	1	0.6	2	1	6	2
14	44	1	2	120	263	0	0	173	0	0	1	0	7	0
15	52	1	3	172	199	1	0	162	0	0.5	1	0	7	0
16	57	1	3	150	168	0	0	174	0	1.6	1	0	3	0
17	48	1	2	110	229	0	0	168	0	1	3	0	7	1
18	54	1	4	140	239	0	0	160	0	1.2	1	0	3	0
19	48	0	3	130	275	0	0	139	0	0.2	1	0	3	0
20	49	1	2	130	266	0	0	171	0	0.6	1	0	3	0
21	64	1	1	110	211	0	2	144	1	1.8	2	0	3	0
22	58	0	1	150	283	1	2	162	0	1	1	0	3	0
23	58	1	2	120	284	0	2	160	0	1.8	2	0	3	1
24	58	1	3	132	224	0	2	173	0	3.2	1	2	7	3
25	60	1	4	130	206	0	2	132	1	2.4	2	2	7	4
26	50	0	3	120	219	0	0	158	0	1.6	2	0	3	0
27	58	0	3	120	340	0	0	172	0	0	1	0	3	0
28	66	0	1	150	226	0	0	114	0	2.6	3	0	3	0
29	43	1	4	150	247	0	0	171	0	1.5	1	0	3	0
30	40	1	4	110	167	0	2	114	1	2	2	0	7	3
31	69	0	1	140	239	0	0	151	0	1.8	1	2	3	0
32	60	1	4	117	230	1	0	160	1	1.4	1	2	7	2
33	64	1	3	140	335	0	0	158	0	0	1	0	3	1
34	59	1	4	135	234	0	0	161	0	0.5	2	0	7	0

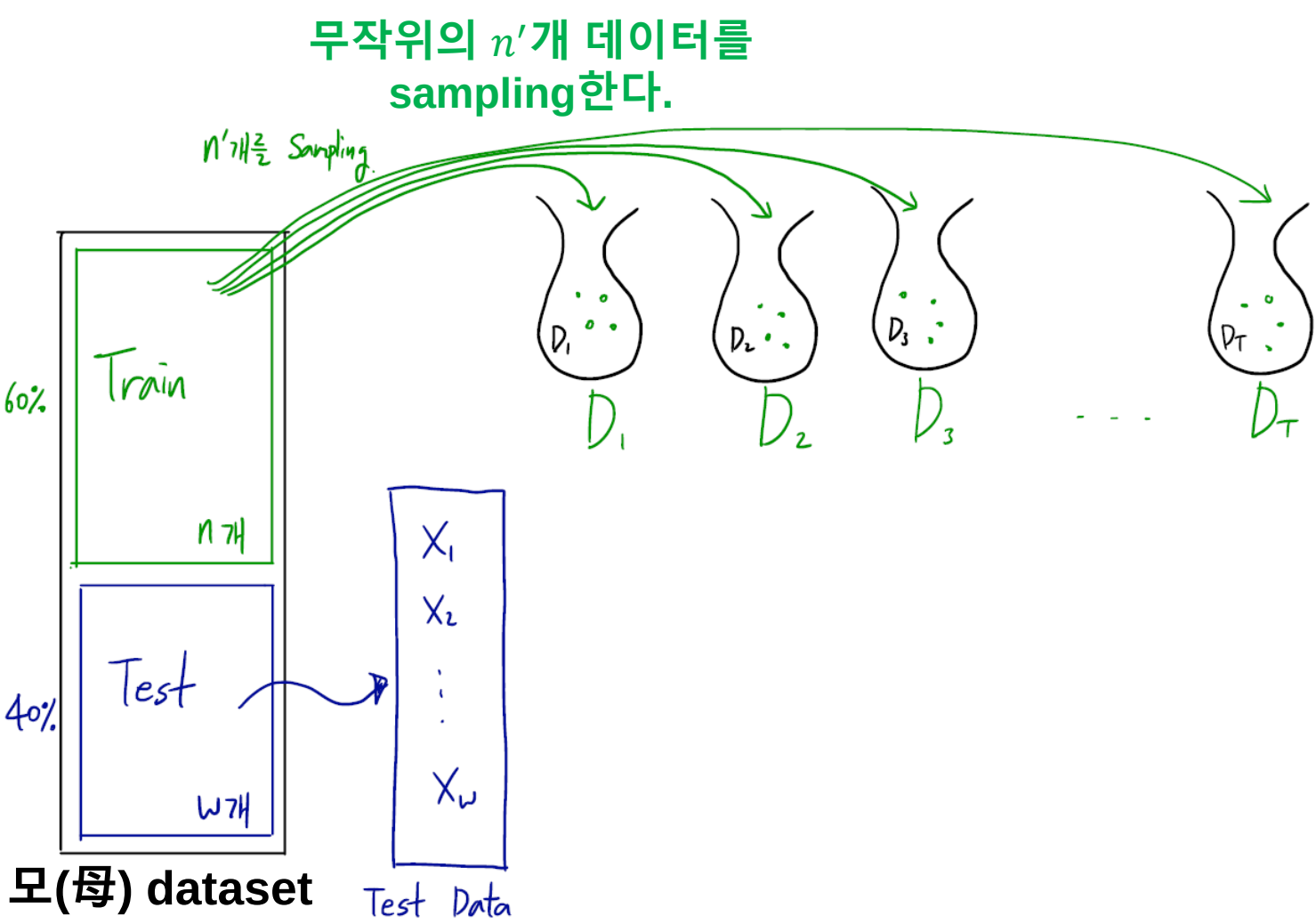
Bagging 실습 I

1. 데이터 로드
2. 샘플링
3. 모델 학습
4. 예측 평균

```
def load_csv(filename):
    dataset = []
    with open(filename, 'r') as file:
        reader = csv.reader(file, quoting=csv.QUOTE_NONNUMERIC)
        for data in reader:
            if not data:
                continue
            else:
                dataset.append(data)

    return np.asarray(dataset)
```

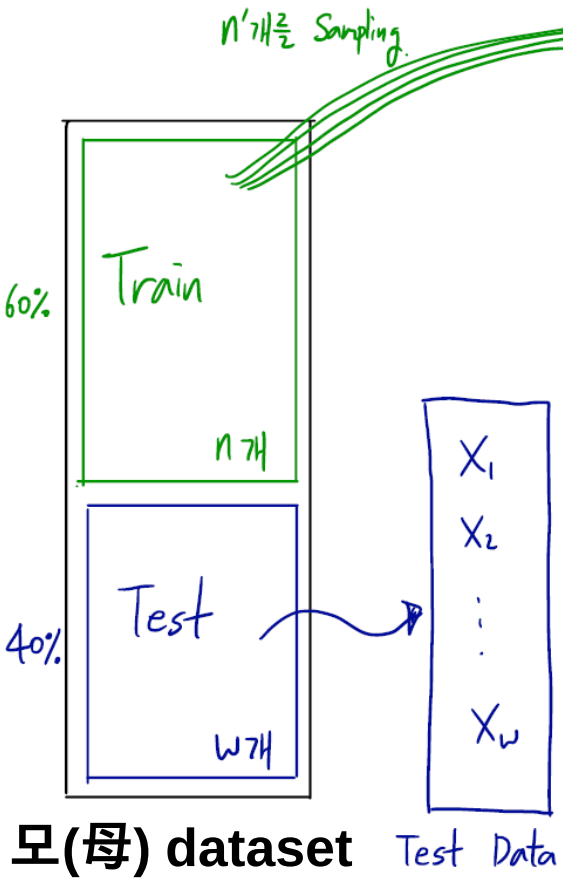
Bagging 실습 I

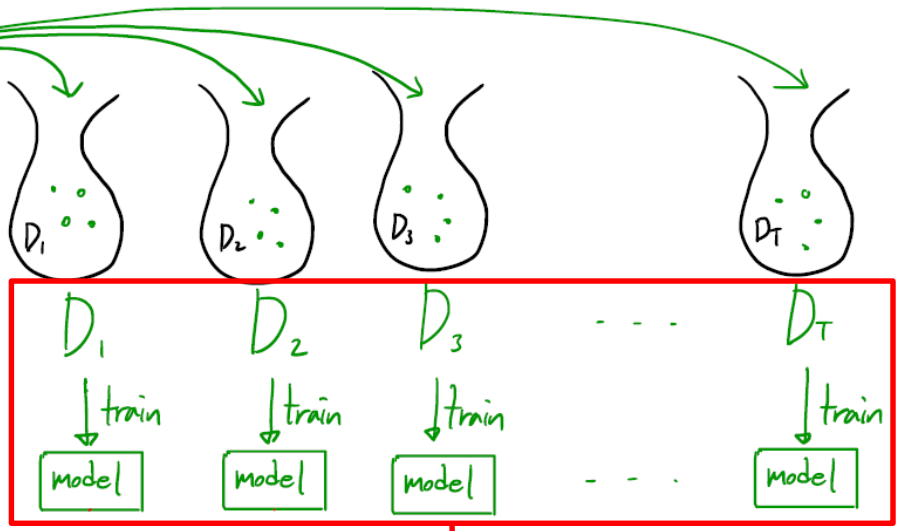
1. 데이터 로드
2. **샘플링**
3. 모델 학습
4. 예측 평균



Bagging 실습 I

1. 데이터 로드
2. 샘플링
3. **모델 학습**
4. 예측 평균

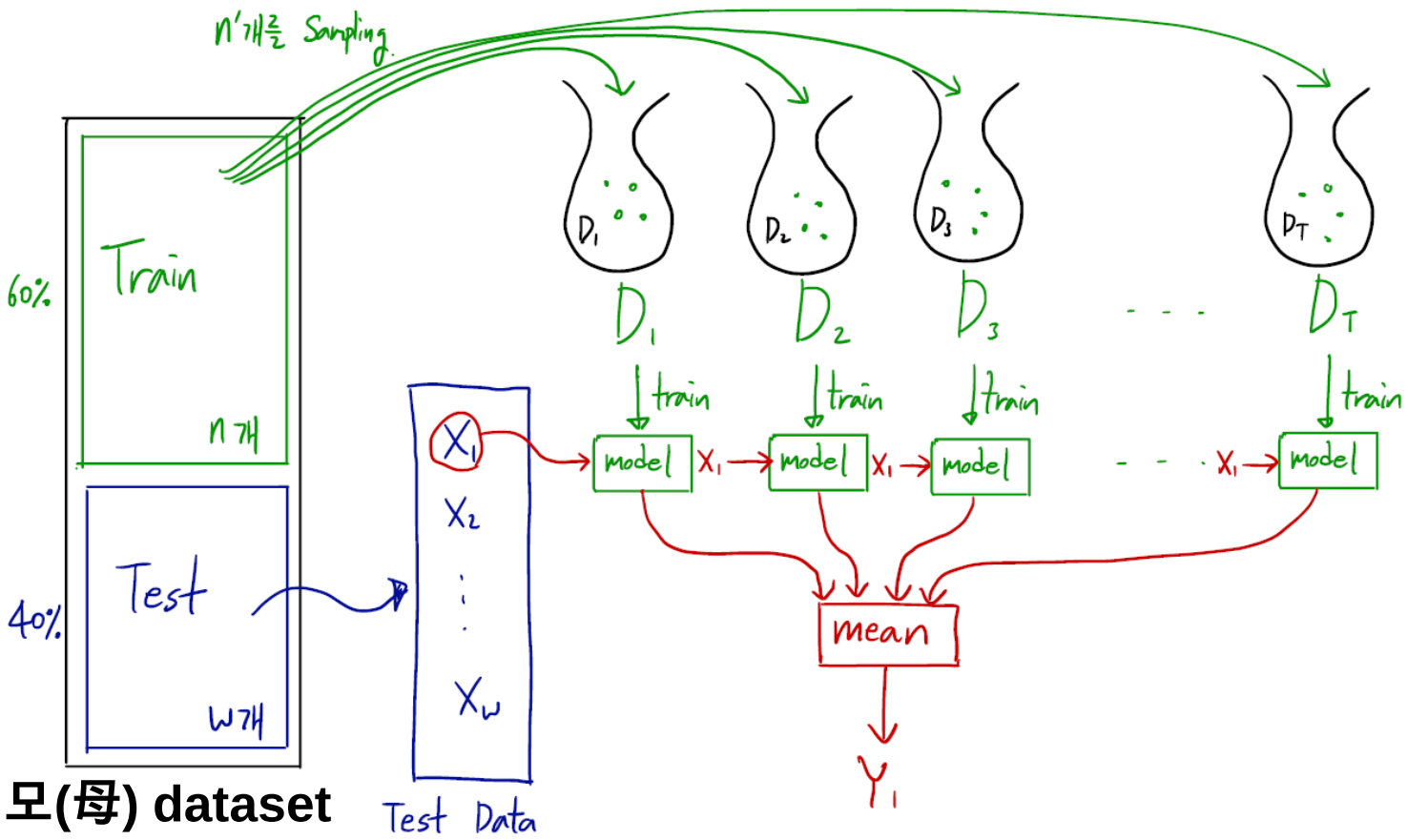





```
model = DecisionTreeClassifier(max_depth=1)
model.fit(sub_train_dataset, sub_train_label)
models.append(model)
```

Bagging 실습 I

1. 데이터 로드
2. 샘플링
3. 모델 학습
4. **예측 평균**



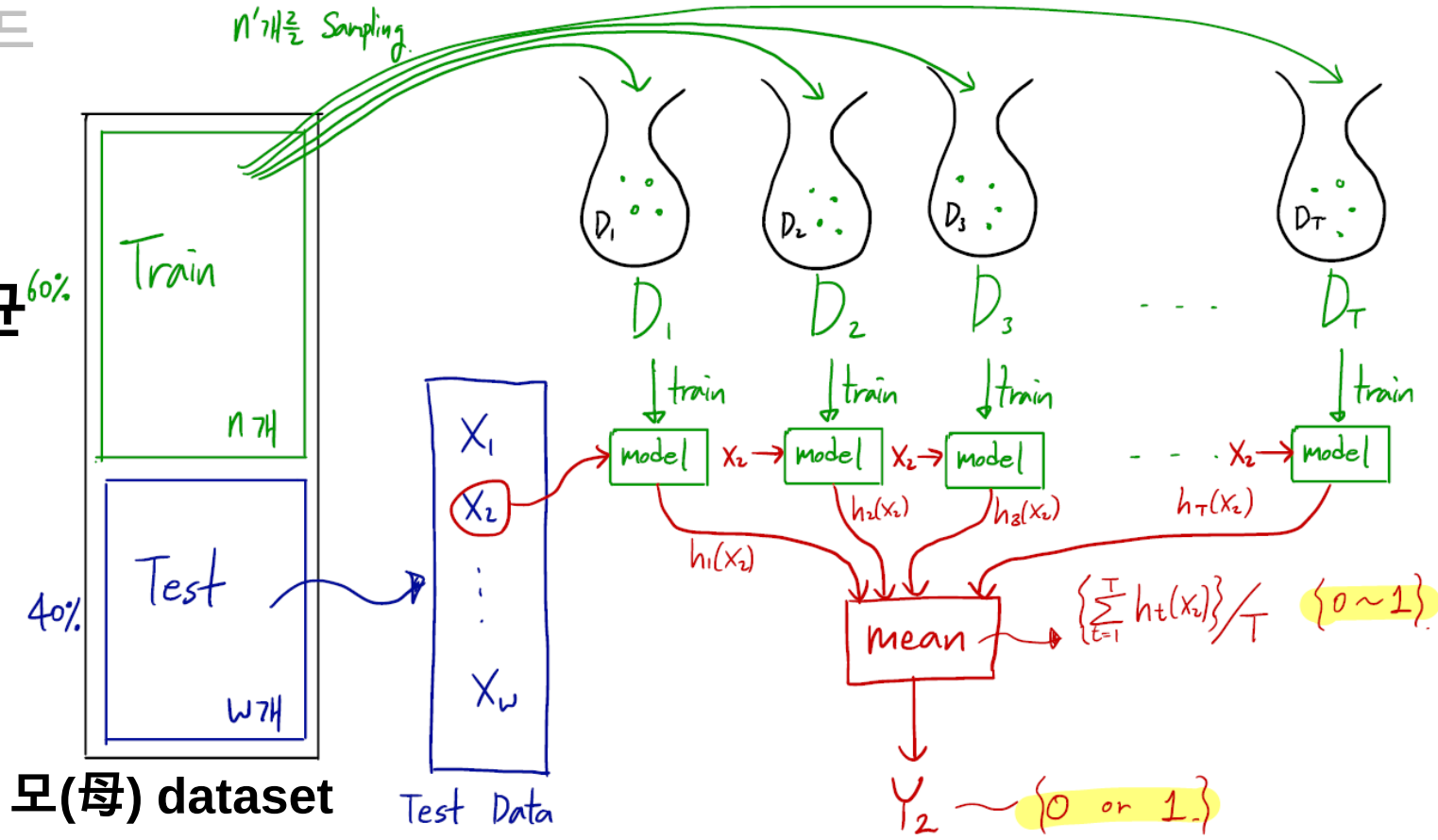
Test Dataset에서 X_1의 대한 결과 예측

(Y_1: 0 *or* 1)

KAIST

Bagging 실습 I

1. 데이터 로드
2. 샘플링
3. 모델 학습
4. **예측 평균**



$[Y_1, Y_2, \ldots, Y_w]$:
"final_prediction"

⇦ **Test Dataset에서 X_2의 대한 결과 예측**
(Y_2: 0 *or* 1)

Bagging 실습 I

1. 데이터 로드
2. 샘플링
3. 모델 학습
4. **예측 평균**

```
[[ FOLD 1 ]]
[1] bagging_accuracy    : 0.767, sk_bagging_accuracy    : 0.683
[5] bagging_accuracy    : 0.800, sk_bagging_accuracy    : 0.867
[10] bagging_accuracy   : 0.867, sk_bagging_accuracy    : 0.850
[50] bagging_accuracy   : 0.883, sk_bagging_accuracy    : 0.883
[100] bagging_accuracy  : 0.883, sk_bagging_accuracy    : 0.883
[500] bagging_accuracy  : 0.883, sk_bagging_accuracy    : 0.900

[[ FOLD 2 ]]
[1] bagging_accuracy    : 0.763, sk_bagging_accuracy    : 0.712
[5] bagging_accuracy    : 0.831, sk_bagging_accuracy    : 0.729
[10] bagging_accuracy   : 0.780, sk_bagging_accuracy    : 0.746
[50] bagging_accuracy   : 0.831, sk_bagging_accuracy    : 0.847
[100] bagging_accuracy  : 0.831, sk_bagging_accuracy    : 0.831
[500] bagging_accuracy  : 0.831, sk_bagging_accuracy    : 0.847

[[ FOLD 3 ]]
[1] bagging_accuracy    : 0.729, sk_bagging_accuracy    : 0.695
[5] bagging_accuracy    : 0.695, sk_bagging_accuracy    : 0.814
[10] bagging_accuracy   : 0.831, sk_bagging_accuracy    : 0.746
[50] bagging_accuracy   : 0.797, sk_bagging_accuracy    : 0.780
[100] bagging_accuracy  : 0.712, sk_bagging_accuracy    : 0.814
[500] bagging_accuracy  : 0.780, sk_bagging_accuracy    : 0.780

[[ FOLD 4 ]]
[1] bagging_accuracy    : 0.627, sk_bagging_accuracy    : 0.712
[5] bagging_accuracy    : 0.712, sk_bagging_accuracy    : 0.780
[10] bagging_accuracy   : 0.763, sk_bagging_accuracy    : 0.763
[50] bagging_accuracy   : 0.763, sk_bagging_accuracy    : 0.780
[100] bagging_accuracy  : 0.797, sk_bagging_accuracy    : 0.797
[500] bagging_accuracy  : 0.763, sk_bagging_accuracy    : 0.797

[[ MEAN ]]
[1] bagging_accuracy    : 0.730, sk_bagging_accuracy    : 0.717
[5] bagging_accuracy    : 0.784, sk_bagging_accuracy    : 0.804
[10] bagging_accuracy   : 0.805, sk_bagging_accuracy    : 0.788
[50] bagging_accuracy   : 0.828, sk_bagging_accuracy    : 0.825
[100] bagging_accuracy  : 0.814, sk_bagging_accuracy    : 0.835
[500] bagging_accuracy  : 0.821, sk_bagging_accuracy    : 0.831
```

Bagging 실습 I

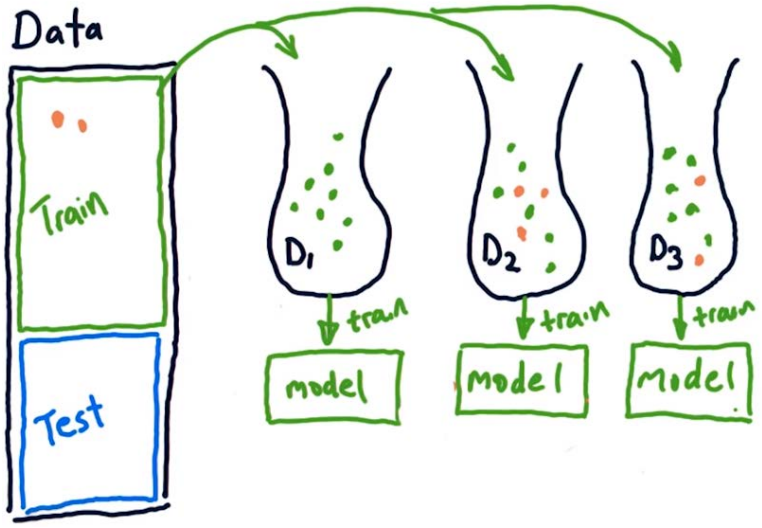
정답:

1. 데이터 로드
2. 샘플링
3. 모델 학습
4. **예측 평균**

```
[[ FOLD 1 ]]
[1] bagging_accuracy    : 0.767, sk_bagging_accuracy    : 0.683
[5] bagging_accuracy    : 0.800, sk_bagging_accuracy    : 0.867
[10] bagging_accuracy   : 0.867, sk_bagging_accuracy    : 0.850
[50] bagging_accuracy   : 0.883, sk_bagging_accuracy    : 0.883
[100] bagging_accuracy  : 0.883, sk_bagging_accuracy    : 0.883
[500] bagging_accuracy  : 0.883, sk_bagging_accuracy    : 0.900

[[ FOLD 2 ]]
[1] bagging_accuracy    : 0.763, sk_bagging_accuracy    : 0.712
[5] bagging_accuracy    : 0.831, sk_bagging_accuracy    : 0.729
[10] bagging_accuracy   : 0.780, sk_bagging_accuracy    : 0.746
[50] bagging_accuracy   : 0.831, sk_bagging_accuracy    : 0.847
[100] bagging_accuracy  : 0.831, sk_bagging_accuracy    : 0.831
[500] bagging_accuracy  : 0.831, sk_bagging_accuracy    : 0.847

[[ FOLD 3 ]]
[1] bagging_accuracy    : 0.729, sk_bagging_accuracy    : 0.695
[5] bagging_accuracy    : 0.695, sk_bagging_accuracy    : 0.814
[10] bagging_accuracy   : 0.831, sk_bagging_accuracy    : 0.746
[50] bagging_accuracy   : 0.797, sk_bagging_accuracy    : 0.780
[100] bagging_accuracy  : 0.712, sk_bagging_accuracy    : 0.814
[500] bagging_accuracy  : 0.780, sk_bagging_accuracy    : 0.780

[[ FOLD 4 ]]
[1] bagging_accuracy    : 0.627, sk_bagging_accuracy    : 0.712
[5] bagging_accuracy    : 0.712, sk_bagging_accuracy    : 0.780
[10] bagging_accuracy   : 0.763, sk_bagging_accuracy    : 0.763
[50] bagging_accuracy   : 0.763, sk_bagging_accuracy    : 0.780
[100] bagging_accuracy  : 0.797, sk_bagging_accuracy    : 0.797
[500] bagging_accuracy  : 0.763, sk_bagging_accuracy    : 0.797

[[ MEAN ]]
[1] bagging_accuracy    : 0.730, sk_bagging_accuracy    : 0.717
[5] bagging_accuracy    : 0.784, sk_bagging_accuracy    : 0.804
[10] bagging_accuracy   : 0.805, sk_bagging_accuracy    : 0.788
[50] bagging_accuracy   : 0.828, sk_bagging_accuracy    : 0.825
[100] bagging_accuracy  : 0.814, sk_bagging_accuracy    : 0.835
[500] bagging_accuracy  : 0.821, sk_bagging_accuracy    : 0.831
```

Boosting (Ada-Boost) |

1. 모(母) dataset에서 sub-dataset을 sampling한 후
2. Sub-dataset으로 하나의 model을 학습시키고
3. 에러가 큰 표본에 대하여 weight를 증가하고, 에러가 작은 표본에 대하여 weight를 감소시킴
4. 1~3을 반복할 때 증/감된 weight를 사용하여 에러가 큰 표본이 더 잘 sampling 되도록 반영시키는 방법



https://www.youtube.com/watch?v=GM3CDQfQ4sw

KAIST

Ada-Boost 실습 I

1. 데이터 로드
2. 약분류기 학습
3. 새로운 weight 계산
4. 강분류기 계산

<모(母) dataset 구조 (HeartDisease.csv 파일)>

```
def load_csv(filename):
    dataset = []
    with open(filename, 'r') as file:
        reader = csv.reader(file, quoting=csv.QUOTE_NONNUMERIC)
        for data in reader:
            if not data:
                continue
            else:
                dataset.append(data)

    return np.asarray(dataset)
```


KAIST

Ada-Boost 실습 I

1. 데이터 로드
2. **약분류기 학습**
3. 새로운 weight 계산
4. 강분류기 계산

```
model = DecisionTreeClassifier(max_depth=1, random_state=1)
model.fit(train_dataset, train_label, sample_weight=weights)
models.append(model)
```

Ada-Boost 실습 |

1. 데이터 로드
2. 약분류기 학습
3. **새로운 weight 계산**
4. 강분류기 계산

<Hint!>

(1) Training dataset

$$Given: (x_1, y_1), \dots, (x_m, y_m); x_i \in \chi, \quad y_i \in \{0(False), 1(True)\}$$

$$Initial\ weight\ W_0 = \frac{1}{m}$$

(2)

$h(x): weak\ classifier$

```
model = DecisionTreeClassifier(max_depth=1, random_state=1)
model.fit(train_dataset, train_label, sample_weight=weights)
models.append(model)
```

$For\ t = 0, \dots, T:$

$$\epsilon_t = \Sigma_{i=1}^{m} W_t(i) \mathbb{I}\left[y_i \neq h_j(x_i)\right]$$

(3) ϵ_t should have a 0~1 value (만약 모두 틀리면, ϵ_t은 1)

$$Set\ \alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

$$W_{t+1} = \begin{cases} W_t * e^{\alpha_t} & if\ y_i \neq h(x_i) \\ W_t * e^{-\alpha_t} & if\ y_i = h(x_i) \end{cases}$$

$Nomalize\ W_{t+1}$

<Notice>

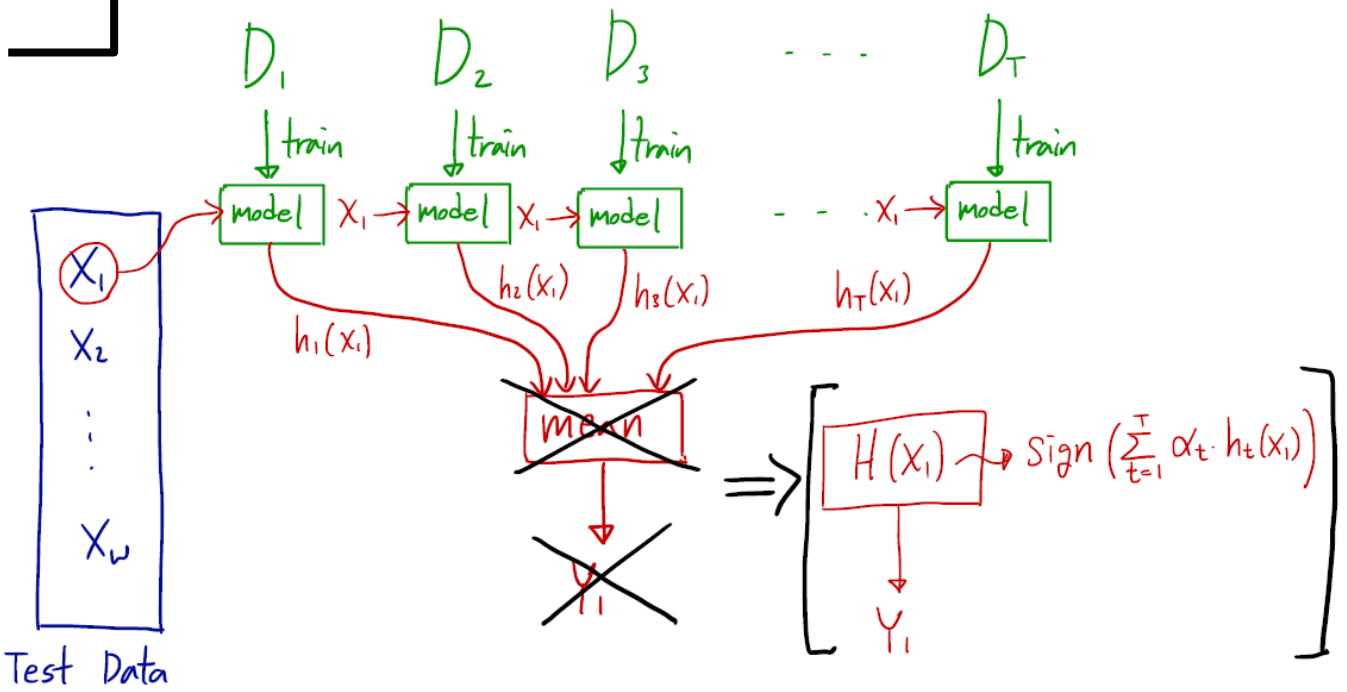
Ada-Boost 실습 I

$$sign(\cdot) = \begin{cases} 1, & if\ (\cdot) > 0 \\ 0 & if\ (\cdot) = 0 \\ -1 & if\ (\cdot) < 0 \end{cases}$$

1. 데이터 로드
2. 약분류기 학습
3. 새로운 weight 계산
4. **강분류기 계산**

$$Strong\ classifier\ H(x) = sign\left(\Sigma_{t=1}^{T}\alpha_t h_t(x)\right)$$

$$\| \quad np.sign(\cdot)$$



Ada-Boost 실습 I

1. 데이터 로드
2. 약분류기 학습
3. 새로운 weight 계산
4. 강분류기 계산

```
[[ FOLD 1 ]]
[1] bagging_accuracy    : 0.750, sk_bagging_accuracy    : 0.767, boosting_accuracy  : 0.767, sk_boosting_accuracy   : 0.767
[5] bagging_accuracy    : 0.833, sk_bagging_accuracy    : 0.867, boosting_accuracy  : 0.900, sk_boosting_accuracy   : 0.900
[10] bagging_accuracy   : 0.883, sk_bagging_accuracy    : 0.800, boosting_accuracy  : 0.900, sk_boosting_accuracy   : 0.850
[50] bagging_accuracy   : 0.917, sk_bagging_accuracy    : 0.917, boosting_accuracy  : 0.850, sk_boosting_accuracy   : 0.883
[100] bagging_accuracy  : 0.917, sk_bagging_accuracy    : 0.917, boosting_accuracy  : 0.883, sk_boosting_accuracy   : 0.933
[500] bagging_accuracy  : 0.883, sk_bagging_accuracy    : 0.917, boosting_accuracy  : 0.883, sk_boosting_accuracy   : 0.817

[[ FOLD 2 ]]
[1] bagging_accuracy    : 0.678, sk_bagging_accuracy    : 0.695, boosting_accuracy  : 0.729, sk_boosting_accuracy   : 0.729
[5] bagging_accuracy    : 0.729, sk_bagging_accuracy    : 0.695, boosting_accuracy  : 0.831, sk_boosting_accuracy   : 0.831
[10] bagging_accuracy   : 0.780, sk_bagging_accuracy    : 0.831, boosting_accuracy  : 0.797, sk_boosting_accuracy   : 0.831
[50] bagging_accuracy   : 0.797, sk_bagging_accuracy    : 0.831, boosting_accuracy  : 0.814, sk_boosting_accuracy   : 0.797
[100] bagging_accuracy  : 0.847, sk_bagging_accuracy    : 0.797, boosting_accuracy  : 0.814, sk_boosting_accuracy   : 0.712
[500] bagging_accuracy  : 0.831, sk_bagging_accuracy    : 0.831, boosting_accuracy  : 0.780, sk_boosting_accuracy   : 0.661

[[ FOLD 3 ]]
[1] bagging_accuracy    : 0.763, sk_bagging_accuracy    : 0.763, boosting_accuracy  : 0.695, sk_boosting_accuracy   : 0.695
[5] bagging_accuracy    : 0.695, sk_bagging_accuracy    : 0.729, boosting_accuracy  : 0.814, sk_boosting_accuracy   : 0.814
[10] bagging_accuracy   : 0.695, sk_bagging_accuracy    : 0.678, boosting_accuracy  : 0.831, sk_boosting_accuracy   : 0.814
[50] bagging_accuracy   : 0.746, sk_bagging_accuracy    : 0.847, boosting_accuracy  : 0.847, sk_boosting_accuracy   : 0.763
[100] bagging_accuracy  : 0.729, sk_bagging_accuracy    : 0.746, boosting_accuracy  : 0.831, sk_boosting_accuracy   : 0.746
[500] bagging_accuracy  : 0.763, sk_bagging_accuracy    : 0.763, boosting_accuracy  : 0.797, sk_boosting_accuracy   : 0.729

[[ FOLD 4 ]]
[1] bagging_accuracy    : 0.763, sk_bagging_accuracy    : 0.610, boosting_accuracy  : 0.712, sk_boosting_accuracy   : 0.712
[5] bagging_accuracy    : 0.746, sk_bagging_accuracy    : 0.729, boosting_accuracy  : 0.780, sk_boosting_accuracy   : 0.780
[10] bagging_accuracy   : 0.746, sk_bagging_accuracy    : 0.763, boosting_accuracy  : 0.780, sk_boosting_accuracy   : 0.814
[50] bagging_accuracy   : 0.780, sk_bagging_accuracy    : 0.780, boosting_accuracy  : 0.780, sk_boosting_accuracy   : 0.695
[100] bagging_accuracy  : 0.763, sk_bagging_accuracy    : 0.831, boosting_accuracy  : 0.780, sk_boosting_accuracy   : 0.712
[500] bagging_accuracy  : 0.797, sk_bagging_accuracy    : 0.763, boosting_accuracy  : 0.746, sk_boosting_accuracy   : 0.644

[[ MEAN ]]
[1] bagging_accuracy    : 0.741, sk_bagging_accuracy    : 0.724, boosting_accuracy  : 0.737, sk_boosting_accuracy   : 0.737
[5] bagging_accuracy    : 0.764, sk_bagging_accuracy    : 0.761, boosting_accuracy  : 0.838, sk_boosting_accuracy   : 0.838
[10] bagging_accuracy   : 0.781, sk_bagging_accuracy    : 0.771, boosting_accuracy  : 0.828, sk_boosting_accuracy   : 0.838
[50] bagging_accuracy   : 0.818, sk_bagging_accuracy    : 0.842, boosting_accuracy  : 0.828, sk_boosting_accuracy   : 0.798
[100] bagging_accuracy  : 0.818, sk_bagging_accuracy    : 0.828, boosting_accuracy  : 0.828, sk_boosting_accuracy   : 0.791
[500] bagging_accuracy  : 0.828, sk_bagging_accuracy    : 0.825, boosting_accuracy  : 0.811, sk_boosting_accuracy   : 0.737
```

Ada-Boost 실습 I

정답:

1. 데이터 로드
2. 약분류기 학습
3. 새로운 weight 계산
4. 강분류기 계산

```
[[ FOLD 1 ]]
[1] bagging_accuracy    : 0.750, sk_bagging_accuracy    : 0.767, boosting_accuracy  : 0.767, sk_boosting_accuracy   : 0.767
[5] bagging_accuracy    : 0.833, sk_bagging_accuracy    : 0.867, boosting_accuracy  : 0.900, sk_boosting_accuracy   : 0.900
[10] bagging_accuracy   : 0.883, sk_bagging_accuracy    : 0.800, boosting_accuracy  : 0.900, sk_boosting_accuracy   : 0.850
[50] bagging_accuracy   : 0.917, sk_bagging_accuracy    : 0.917, boosting_accuracy  : 0.850, sk_boosting_accuracy   : 0.883
[100] bagging_accuracy  : 0.917, sk_bagging_accuracy    : 0.917, boosting_accuracy  : 0.883, sk_boosting_accuracy   : 0.933
[500] bagging_accuracy  : 0.883, sk_bagging_accuracy    : 0.917, boosting_accuracy  : 0.883, sk_boosting_accuracy   : 0.817

[[ FOLD 2 ]]
[1] bagging_accuracy    : 0.678, sk_bagging_accuracy    : 0.695, boosting_accuracy  : 0.729, sk_boosting_accuracy   : 0.729
[5] bagging_accuracy    : 0.729, sk_bagging_accuracy    : 0.695, boosting_accuracy  : 0.831, sk_boosting_accuracy   : 0.831
[10] bagging_accuracy   : 0.780, sk_bagging_accuracy    : 0.831, boosting_accuracy  : 0.797, sk_boosting_accuracy   : 0.831
[50] bagging_accuracy   : 0.797, sk_bagging_accuracy    : 0.831, boosting_accuracy  : 0.814, sk_boosting_accuracy   : 0.797
[100] bagging_accuracy  : 0.847, sk_bagging_accuracy    : 0.797, boosting_accuracy  : 0.814, sk_boosting_accuracy   : 0.712
[500] bagging_accuracy  : 0.831, sk_bagging_accuracy    : 0.831, boosting_accuracy  : 0.780, sk_boosting_accuracy   : 0.661

[[ FOLD 3 ]]
[1] bagging_accuracy    : 0.763, sk_bagging_accuracy    : 0.763, boosting_accuracy  : 0.695, sk_boosting_accuracy   : 0.695
[5] bagging_accuracy    : 0.695, sk_bagging_accuracy    : 0.729, boosting_accuracy  : 0.814, sk_boosting_accuracy   : 0.814
[10] bagging_accuracy   : 0.695, sk_bagging_accuracy    : 0.678, boosting_accuracy  : 0.831, sk_boosting_accuracy   : 0.814
[50] bagging_accuracy   : 0.746, sk_bagging_accuracy    : 0.847, boosting_accuracy  : 0.847, sk_boosting_accuracy   : 0.763
[100] bagging_accuracy  : 0.729, sk_bagging_accuracy    : 0.746, boosting_accuracy  : 0.831, sk_boosting_accuracy   : 0.746
[500] bagging_accuracy  : 0.763, sk_bagging_accuracy    : 0.763, boosting_accuracy  : 0.797, sk_boosting_accuracy   : 0.729

[[ FOLD 4 ]]
[1] bagging_accuracy    : 0.763, sk_bagging_accuracy    : 0.610, boosting_accuracy  : 0.712, sk_boosting_accuracy   : 0.712
[5] bagging_accuracy    : 0.746, sk_bagging_accuracy    : 0.729, boosting_accuracy  : 0.780, sk_boosting_accuracy   : 0.780
[10] bagging_accuracy   : 0.746, sk_bagging_accuracy    : 0.763, boosting_accuracy  : 0.780, sk_boosting_accuracy   : 0.814
[50] bagging_accuracy   : 0.780, sk_bagging_accuracy    : 0.780, boosting_accuracy  : 0.780, sk_boosting_accuracy   : 0.695
[100] bagging_accuracy  : 0.763, sk_bagging_accuracy    : 0.831, boosting_accuracy  : 0.780, sk_boosting_accuracy   : 0.712
[500] bagging_accuracy  : 0.797, sk_bagging_accuracy    : 0.763, boosting_accuracy  : 0.746, sk_boosting_accuracy   : 0.644

[[ MEAN ]]
[1] bagging_accuracy    : 0.741, sk_bagging_accuracy    : 0.724, boosting_accuracy  : 0.737, sk_boosting_accuracy   : 0.737
[5] bagging_accuracy    : 0.764, sk_bagging_accuracy    : 0.761, boosting_accuracy  : 0.838, sk_boosting_accuracy   : 0.838
[10] bagging_accuracy   : 0.781, sk_bagging_accuracy    : 0.771, boosting_accuracy  : 0.828, sk_boosting_accuracy   : 0.838
[50] bagging_accuracy   : 0.818, sk_bagging_accuracy    : 0.842, boosting_accuracy  : 0.828, sk_boosting_accuracy   : 0.798
[100] bagging_accuracy  : 0.818, sk_bagging_accuracy    : 0.828, boosting_accuracy  : 0.828, sk_boosting_accuracy   : 0.791
[500] bagging_accuracy  : 0.828, sk_bagging_accuracy    : 0.825, boosting_accuracy  : 0.811, sk_boosting_accuracy   : 0.737
```