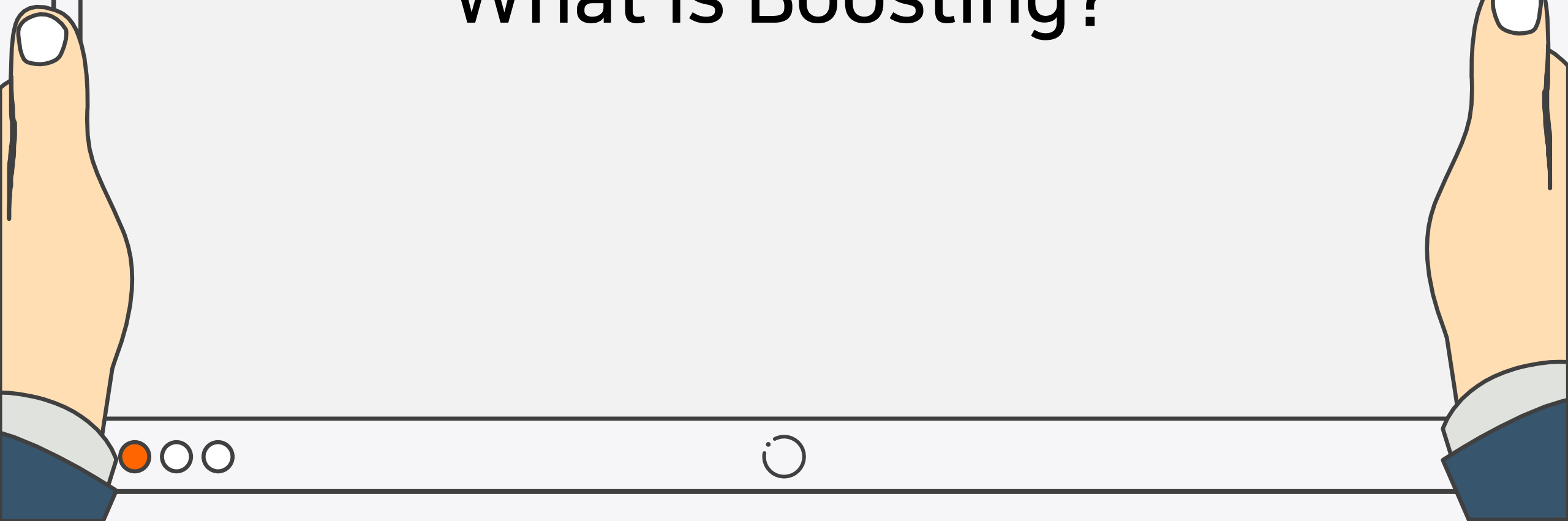


Yoo\_Kye0m

# What is Boosting?



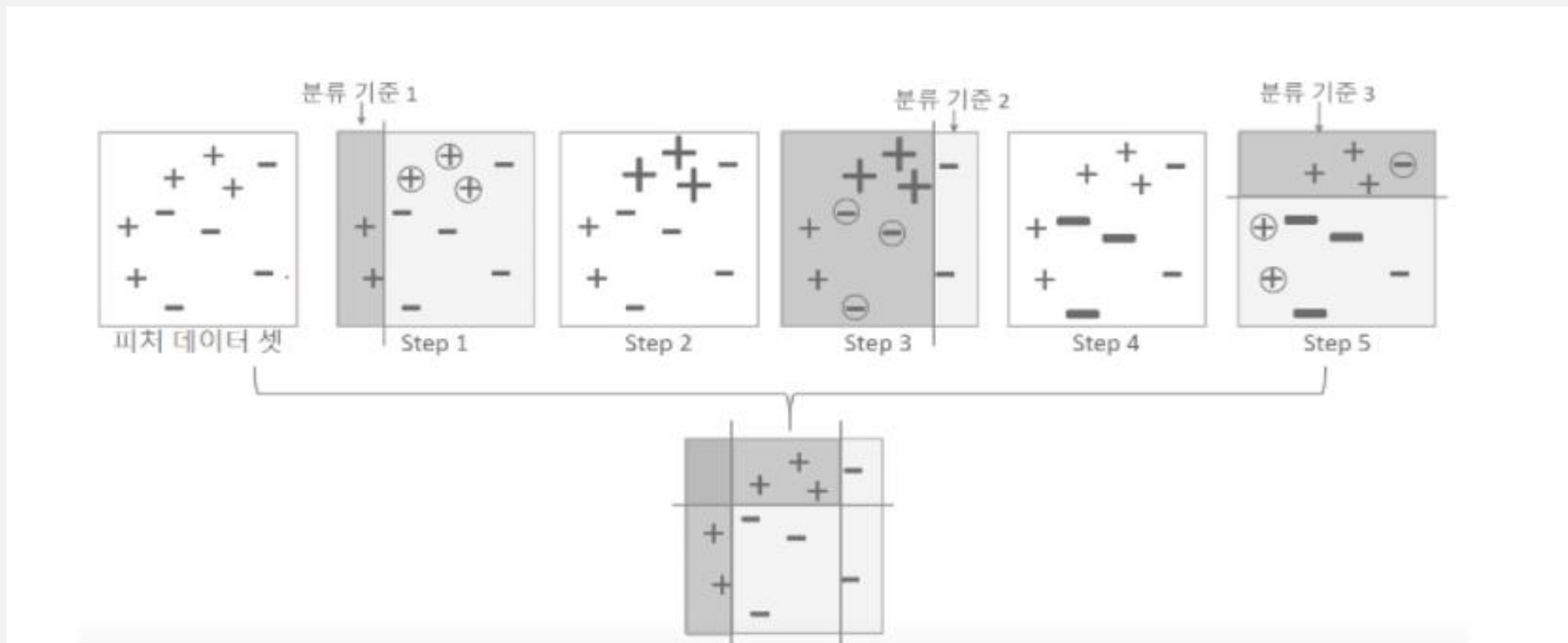
## Contents

First

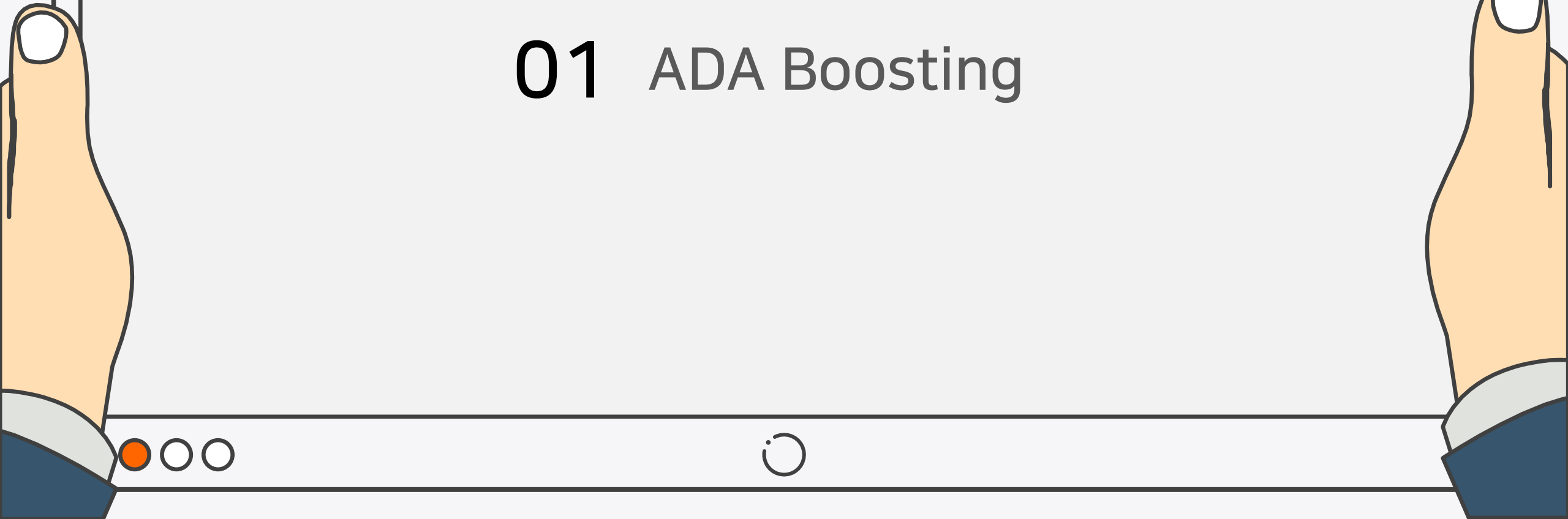
ADA Boosting

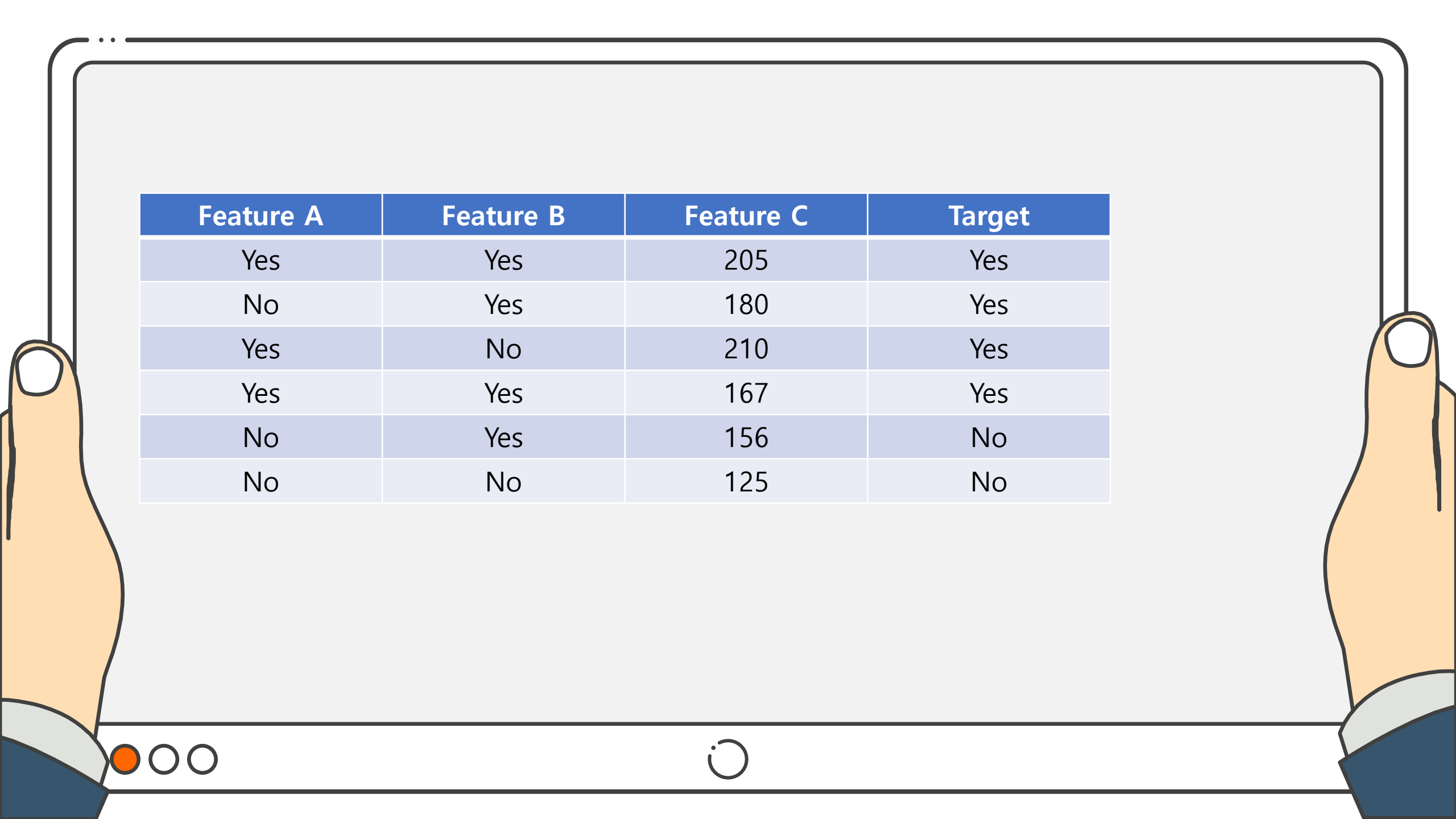
Second

Gradeint Boosting

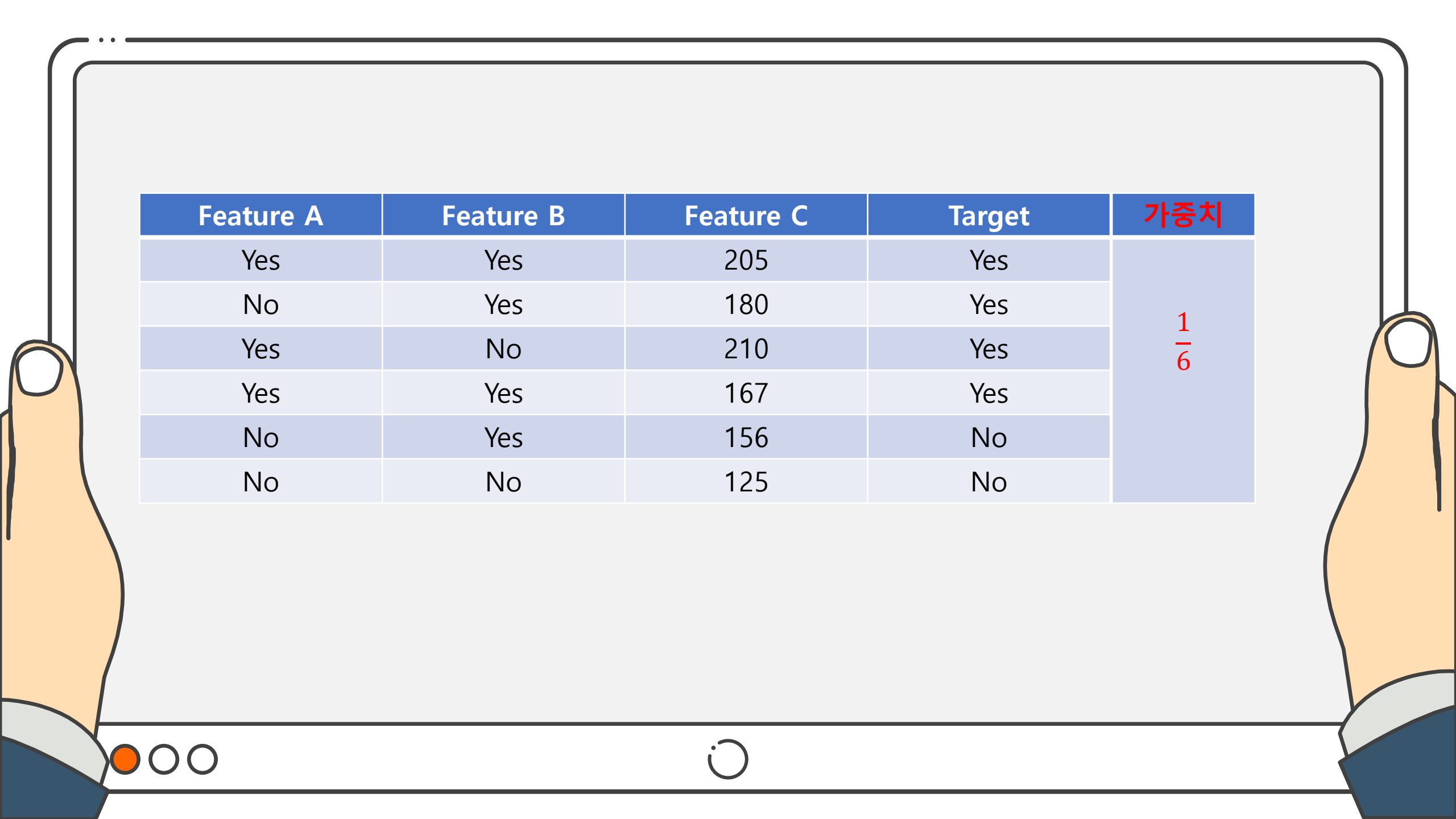


# 01 ADA Boosting





Feature A	Feature B	Feature C	Target
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	No	125	No



Feature A	Feature B	Feature C	Target	가중치
Yes	Yes	205	Yes	$\frac{1}{6}$
No	Yes	180	Yes	
Yes	No	210	Yes	
Yes	Yes	167	Yes	
No	Yes	156	No	
No	No	125	No	

Stump(1)

Feature A

Correct 3 Incorrect 2

Correct 1 Incorrect 0

Stump(2)

Feature B

Correct 2 Incorrect 3

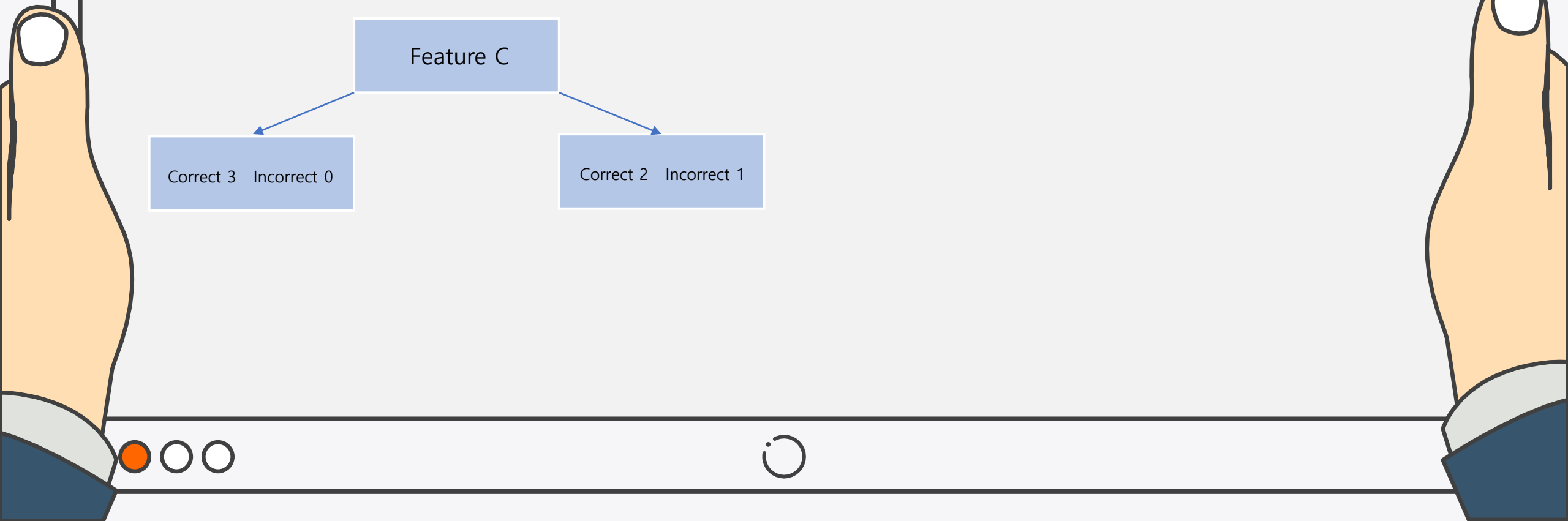
Correct 1 Incorrect 0

Stump(3)

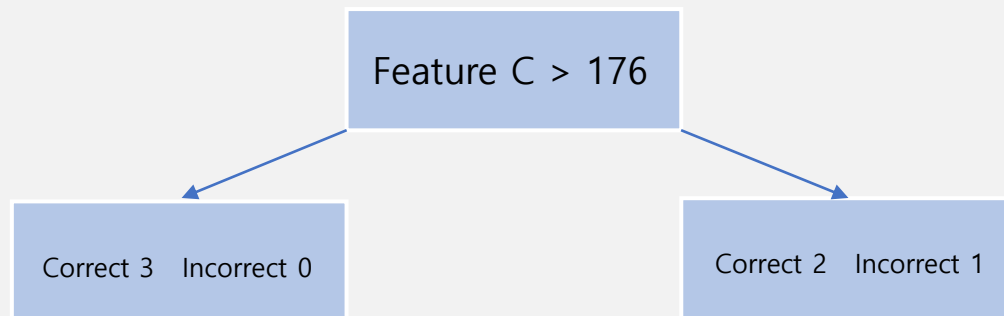
Feature C

Correct 3 Incorrect 0

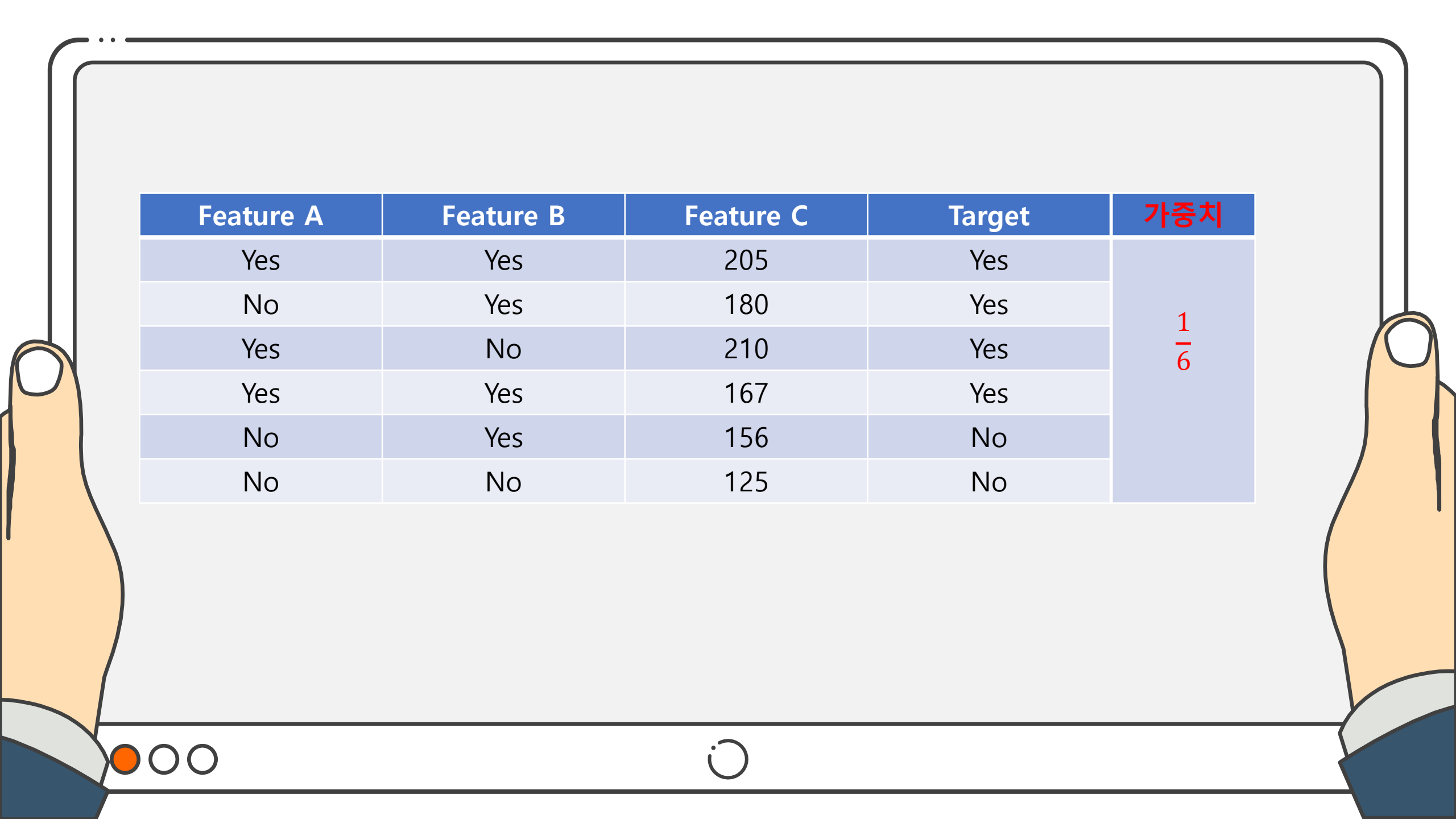
Correct 2 Incorrect 1



Stump(3)







Feature A	Feature B	Feature C	Target	가중치
Yes	Yes	205	Yes	$\frac{1}{6}$
No	Yes	180	Yes	
Yes	No	210	Yes	
Yes	Yes	167	Yes	
No	Yes	156	No	
No	No	125	No	

Stump(3)

Feature C > 176

Correct 3   Incorrect 0

Correct 2   **Incorrect 1**

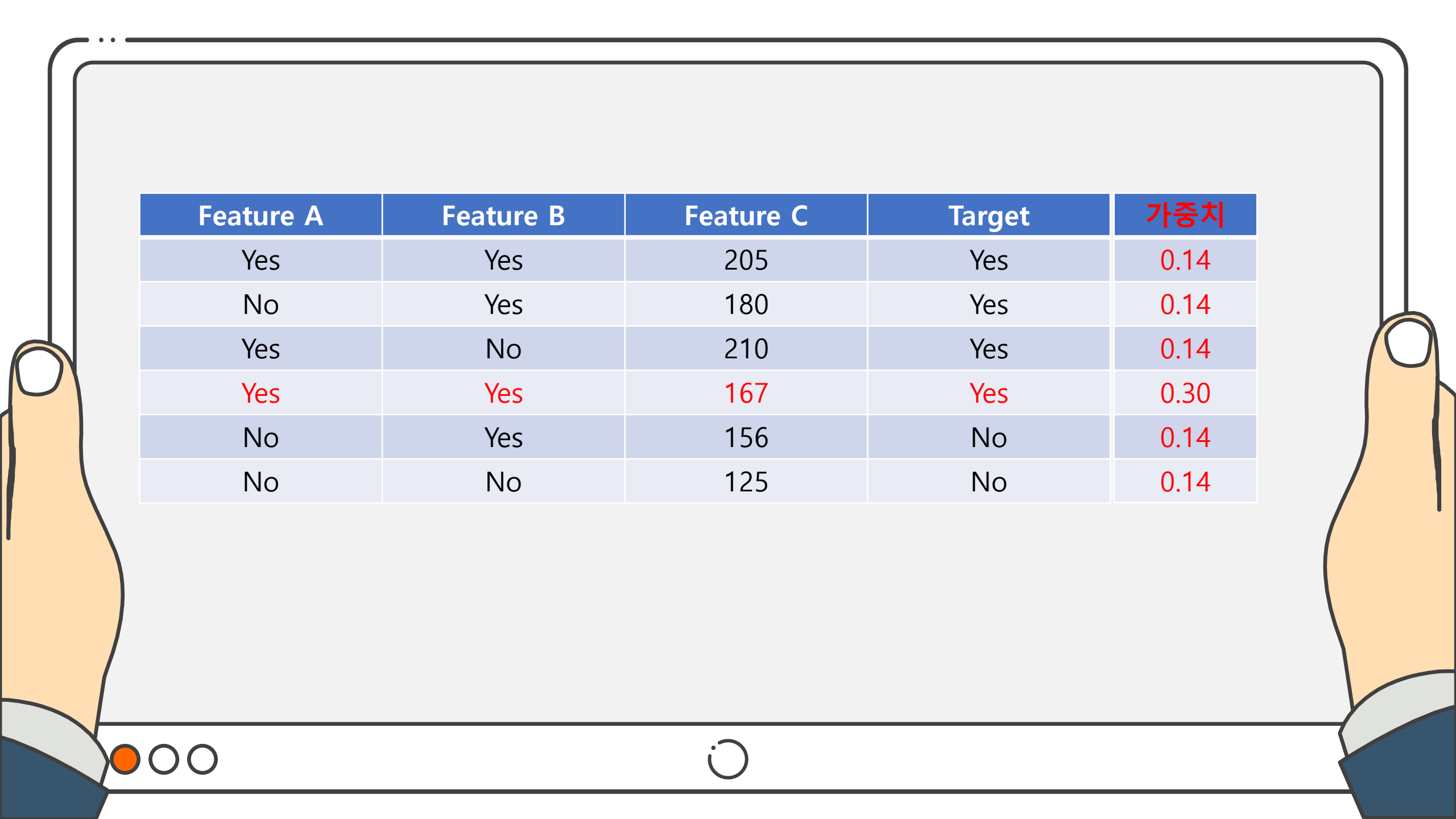
Total Error = 오류가 난 데이터 / 전체 데이터

$$\text{Amount of say} = \frac{1}{2} \log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

$$\text{New Sample Weight} = \text{Sample Weight} * e^{\text{Amount of say}}$$

$$\text{New Sample Weight} = \text{Sample Weight} * e^{-\text{Amount of say}}$$

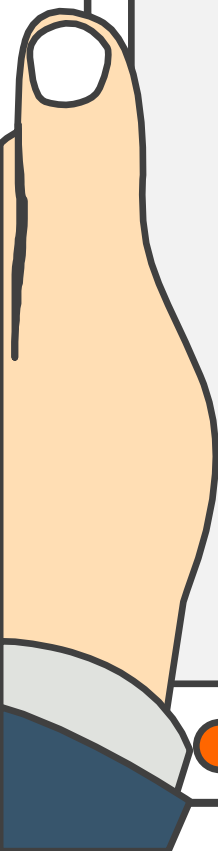
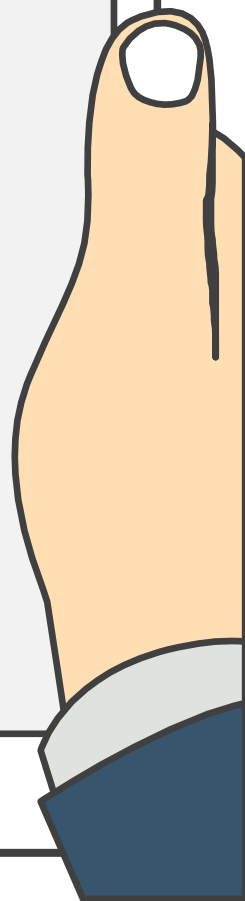
대충 가중치를 부여한다는 뜻



Feature A	Feature B	Feature C	Target	가중치
Yes	Yes	205	Yes	0.14
No	Yes	180	Yes	0.14
Yes	No	210	Yes	0.14
Yes	Yes	167	Yes	0.30
No	Yes	156	No	0.14
No	No	125	No	0.14

### 가중치

0.14	0.00 ~ 0.14
0.14	0.14 ~ 0.28
0.14	0.28 ~ 0.42
0.30	0.42 ~ 0.72
0.14	0.72 ~ 0.86
0.14	0.86 ~ 1.00

- 
- 
1. 초기에는 같은 값의 가중치를 부여한다
  2. Stump중 가장 지니계수가 낮은(=잘 분류된) Stump를 선택
  3. 가중치를 업데이트 후 새로운 데이터 셋을 만든다,
  4. 새로운 만들어진 데이터 셋으로 Stump를 다시 만든다.
  5. 이 과정을 일정 Loop 동안 반복한다.



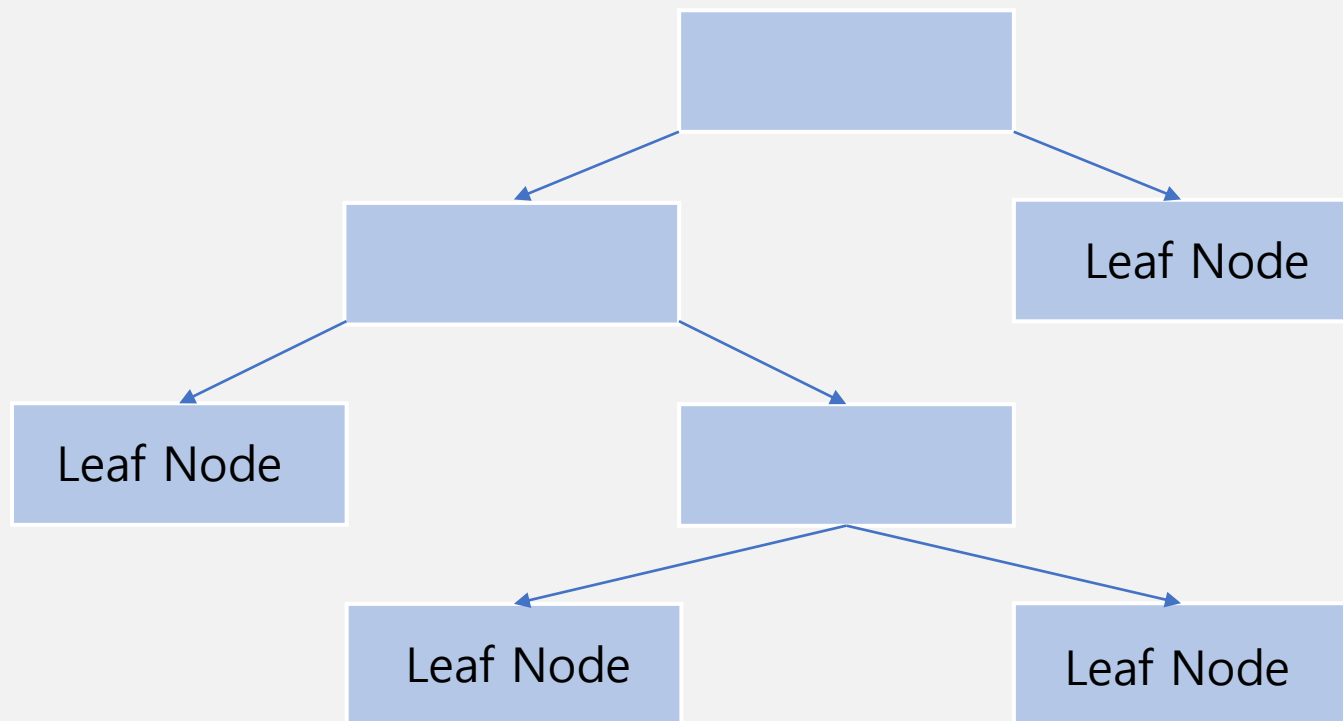
장점 : 단순한 알고리즘으로 구현하기 쉬움

단점 : 노이즈 데이터의 영향을 많이 받음

## 02 Gradient Boosting









키	좋아하는 색깔	성별	몸무게
1.6	파랑	남성	88
1.6	초록	여성	76
1.5	파랑	여성	56
1.8	빨강	남성	73
1.5	초록	남성	77
1.4	파랑	여성	57

몸무게

88

76

56

73

77

57

평균 값 = 71.2

오차

88 - 71.2

76 - 71.2

56 - 71.2

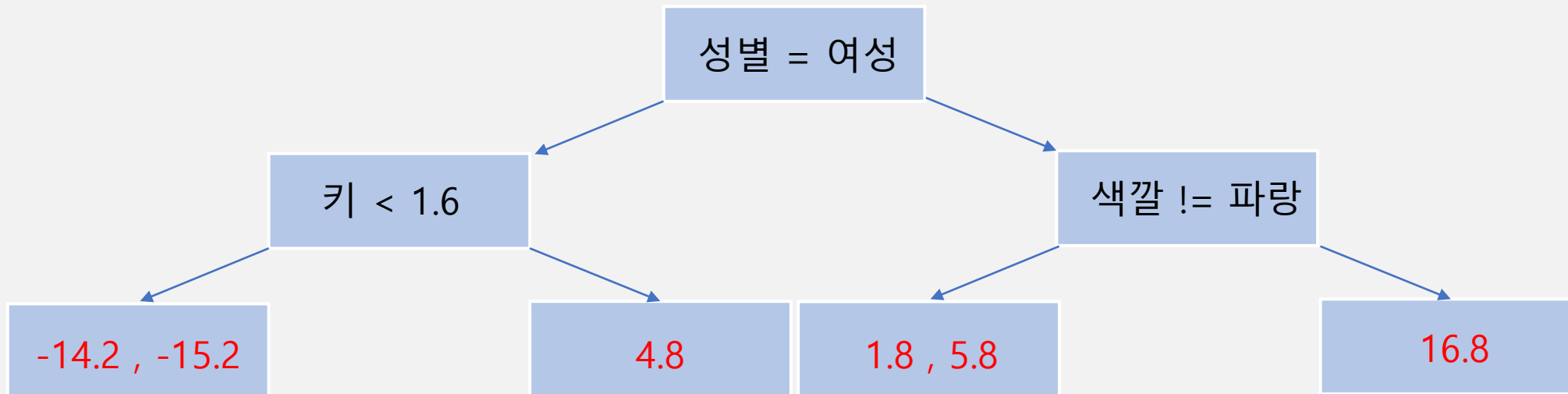
73 - 71.2

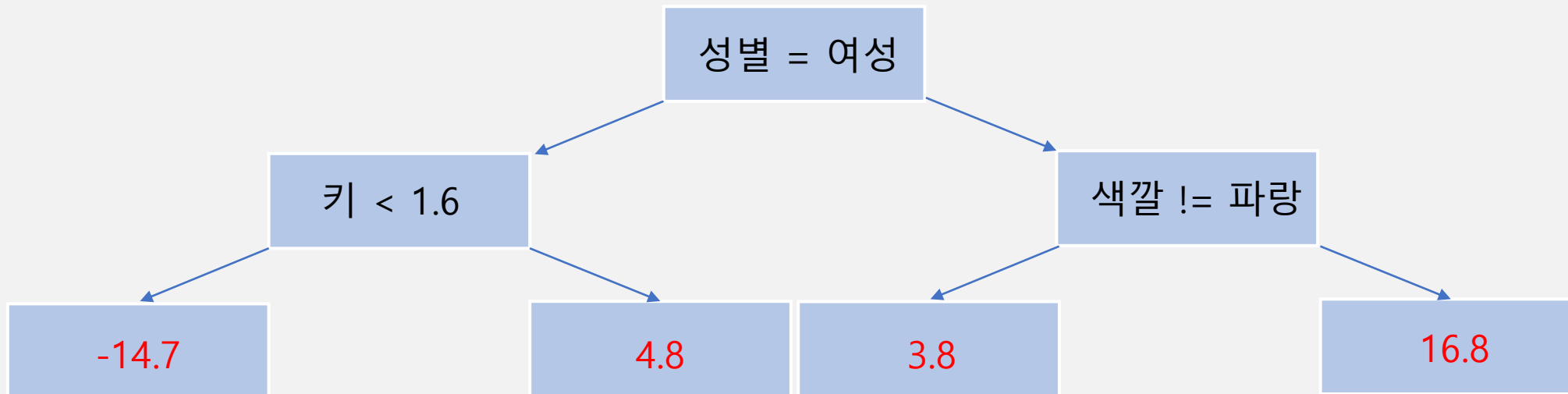
77 - 71.2

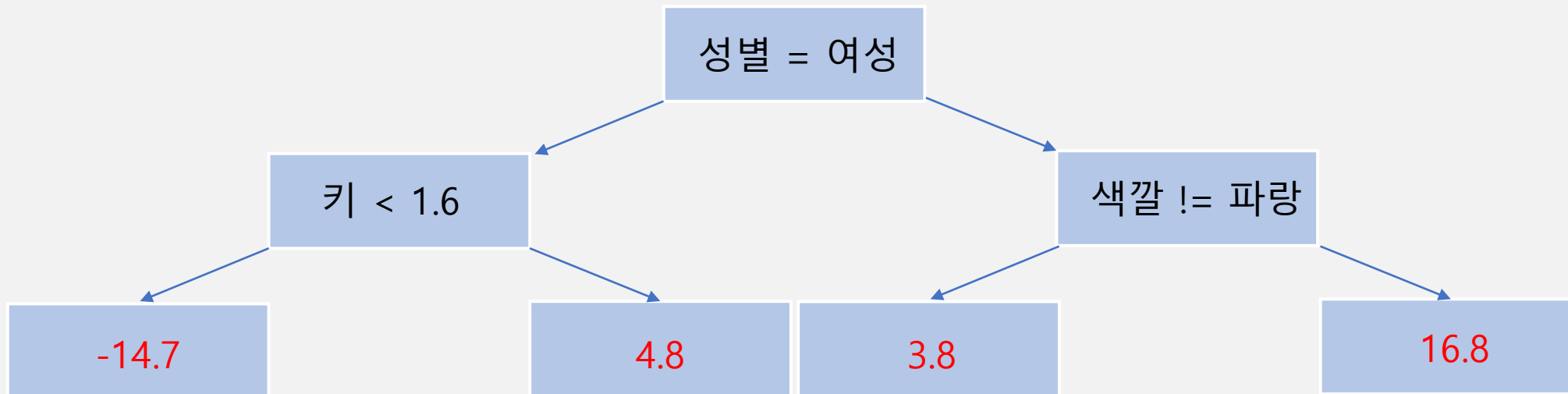
57 - 71.2



키	좋아하는 색깔	성별	몸무게	오차
1.6	파랑	남성	88	16.8
1.6	초록	여성	76	4.8
1.5	파랑	여성	56	-15.2
1.8	빨강	남성	73	1.8
1.5	초록	남성	77	5.8
1.4	파랑	여성	57	-14.2

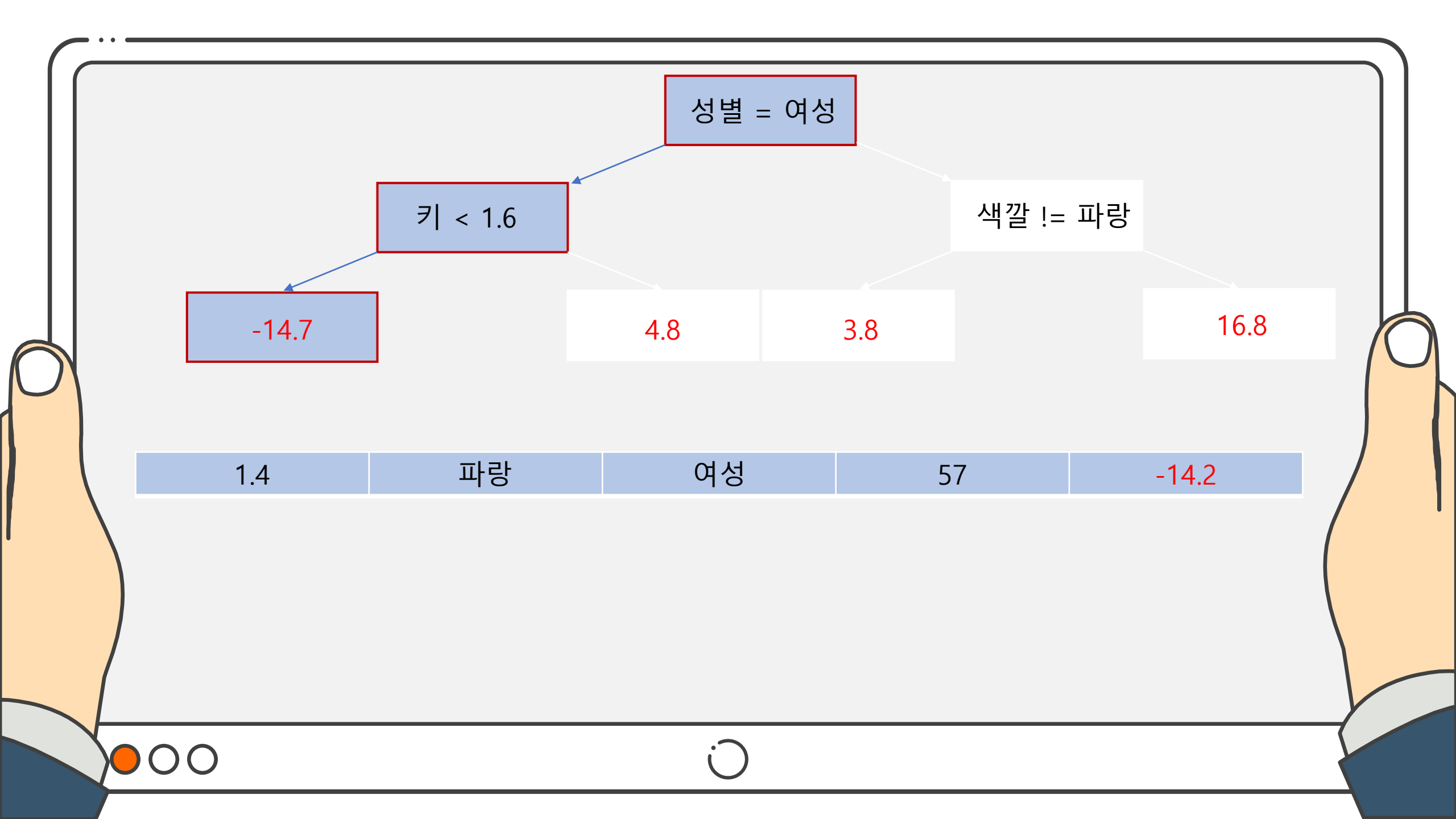






1.4	파랑	여성	57	14.2
-----	----	----	----	------





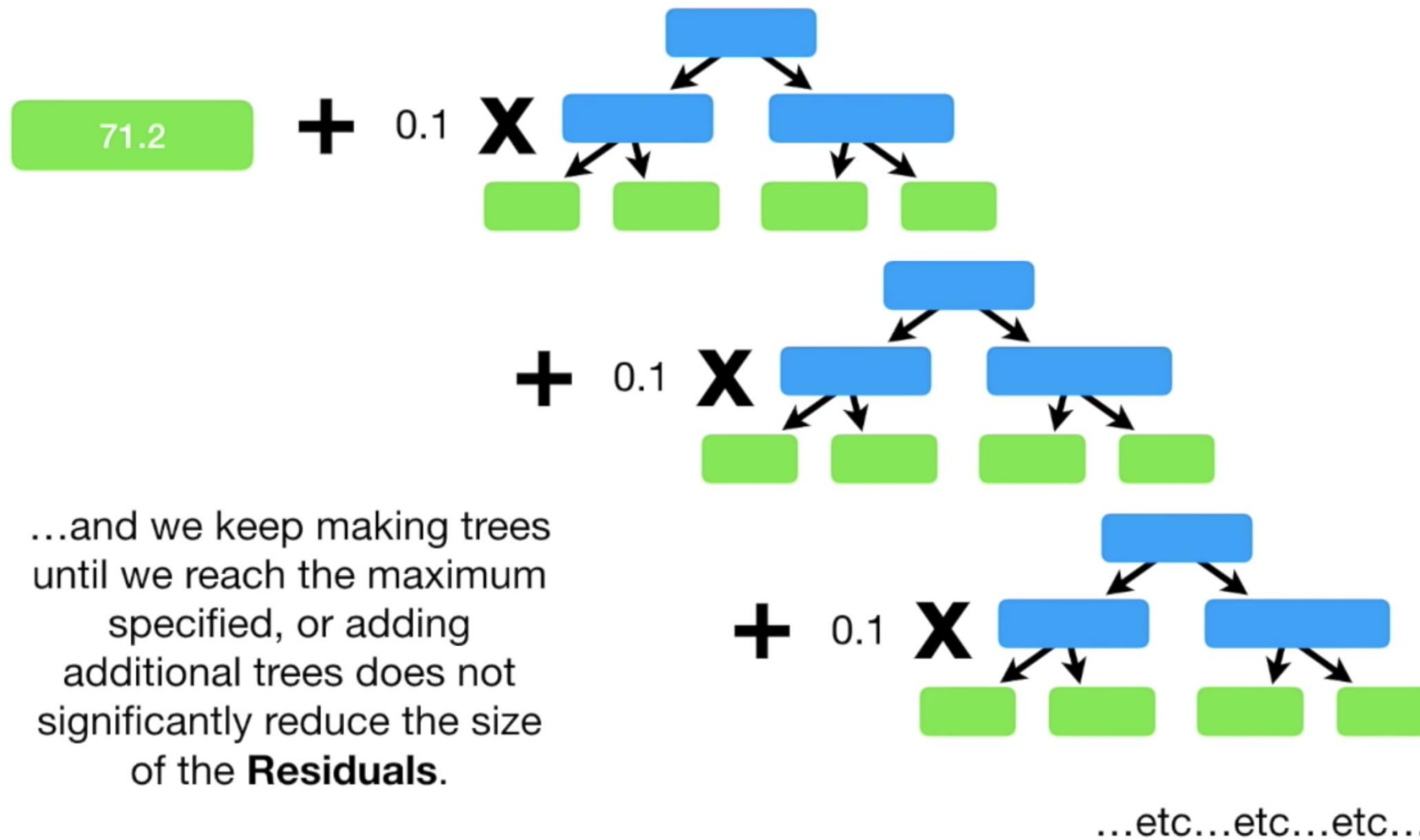


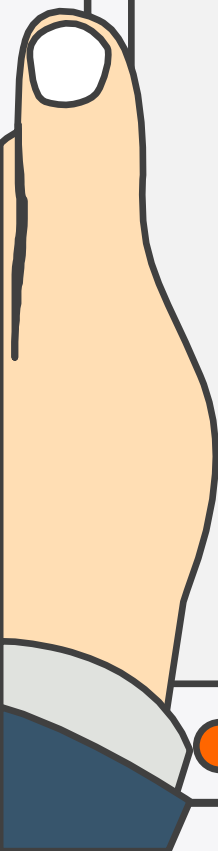
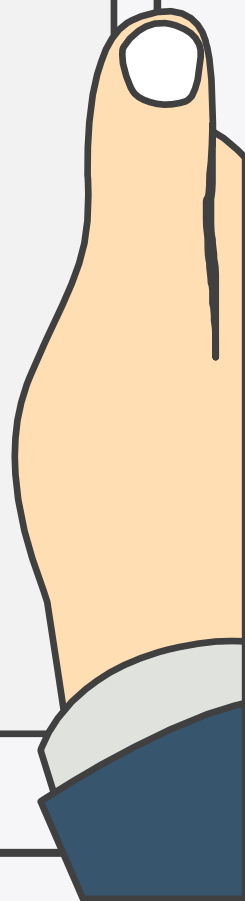
$$-14.2 - (\text{Running rate}(0.1) * (-14.7)) = -12.73$$



The image shows a stylized illustration of a laptop screen. On the screen is a table with five columns: 키 (Height), 좋아하는 색깔 (Favorite Color), 성별 (Gender), 몸무게 (Weight), and 오차 (Error). The table contains six rows of data. The error values are highlighted in red. Two cartoon hands are shown on the sides of the screen, with fingers pointing towards the table. The laptop's bezel at the bottom features three status lights on the left (one orange, two white) and a circular power button in the center.

키	좋아하는 색깔	성별	몸무게	오차
1.6	파랑	남성	88	15.1
1.6	초록	여성	76	4.3
1.5	파랑	여성	56	-13.7
1.8	빨강	남성	73	1.4
1.5	초록	남성	77	5.4
1.4	파랑	여성	57	-12.7



- 
- 
1. 초기에는 평균값으로 모든 예측 값을 예측한다.
  2. 실제 값과 오차를 구해, 해당 오차를 예측하는 Tree 를 만든다.
  3. 기존 예측 값에 오차 \* learning rate 를 더해서,
  4. 새로운 예측 값으로 업데이트 한다.
  5. 이 과정을 일정 Loop 동안 반복한다.



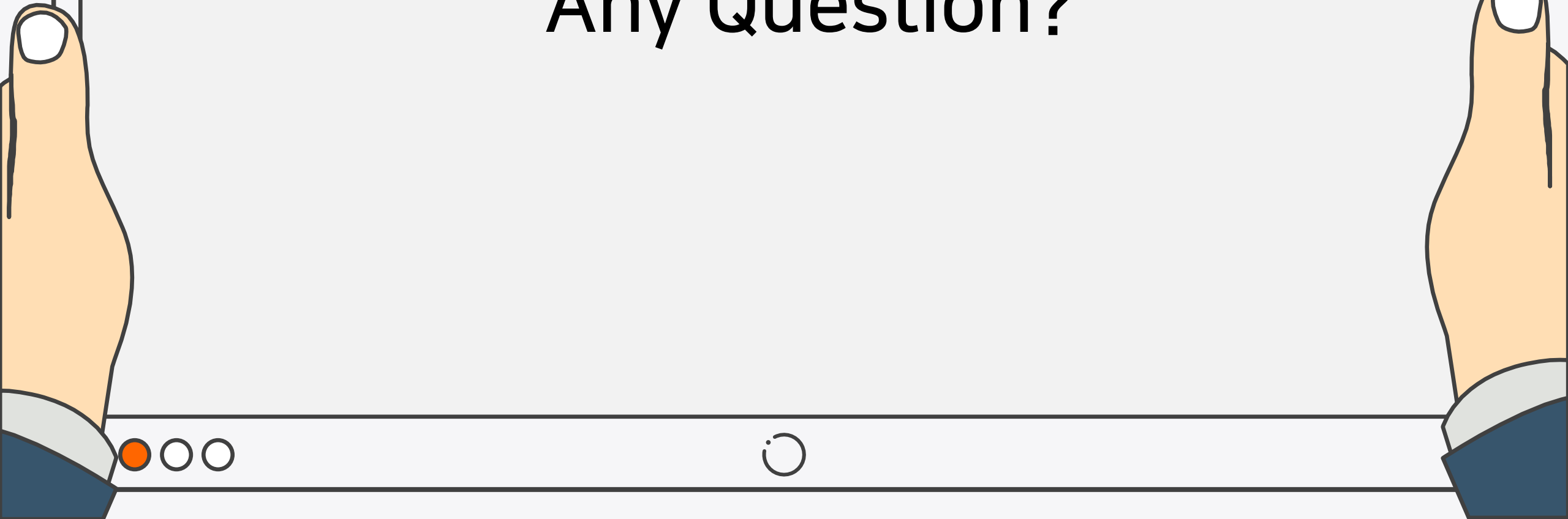
장점 : Random Foest 보다 예측 성능이 뛰어남

단점 : 1. 하이퍼 파라미터 튜닝 노력이 필요

2. 병렬처리를 지원하지 않기 때문에  
학습에 매우 많은 시간이 필요



Any Question?



감사합니다 :)

