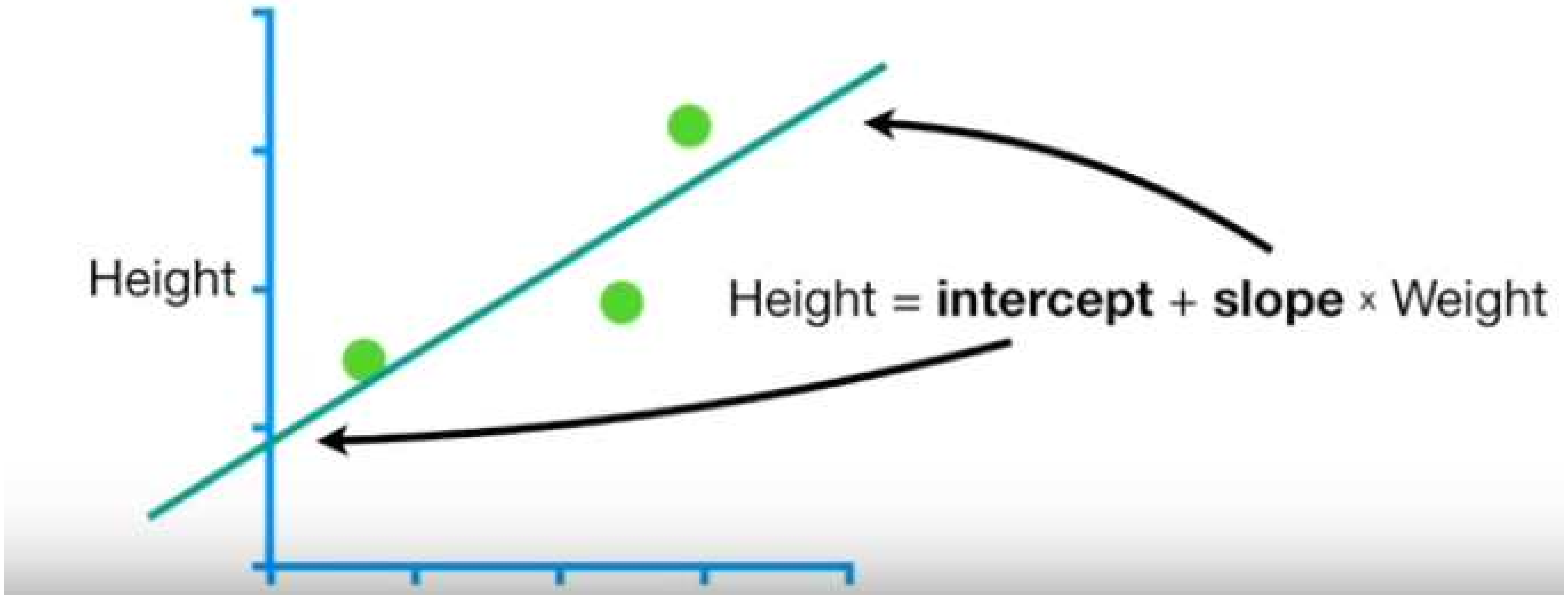
Gradient Descent

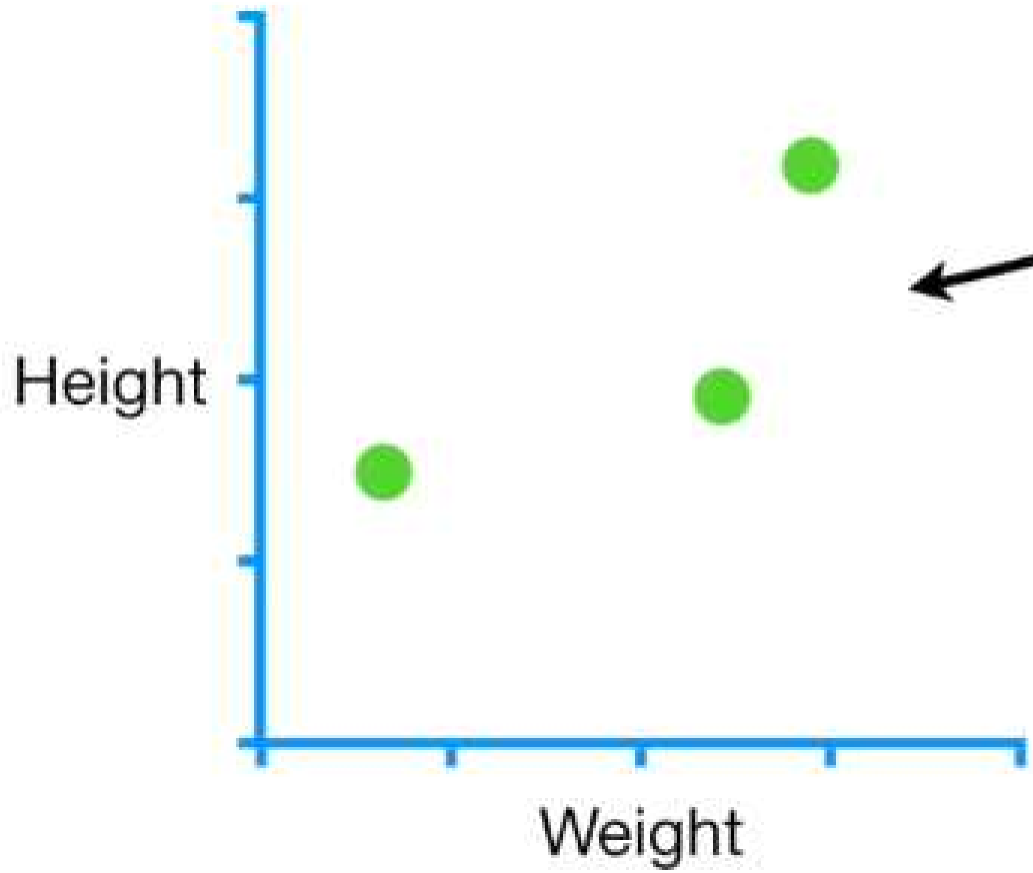
Gradient Descent

- 기본적으로 최적화를 찾는데 사용되는 기본적인 방법론
- 여러가지 알고리즘에서 많이 등장하게 된다.



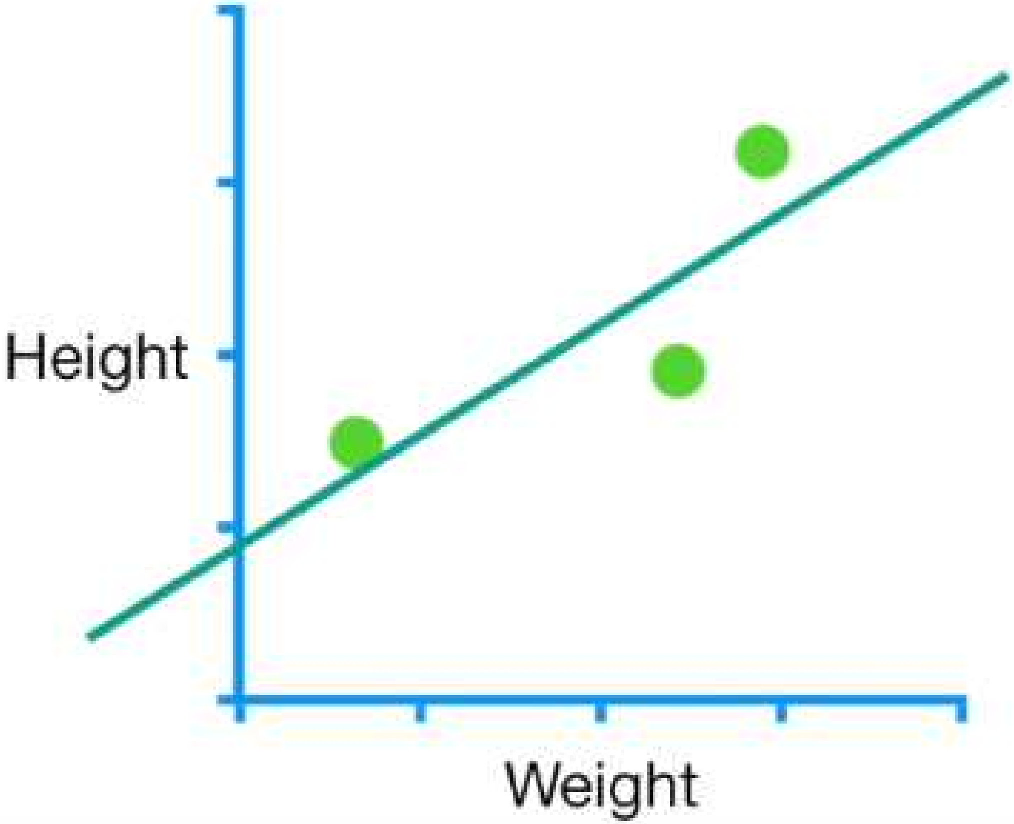
Weight

So let's start with a
simple data set.



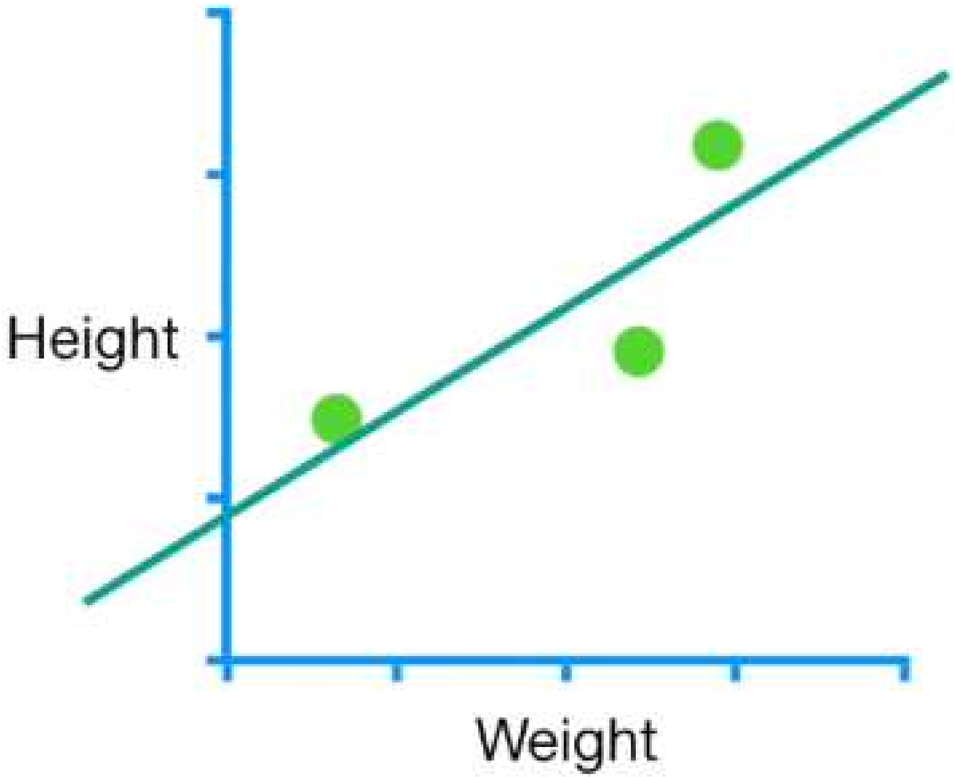
Predicted Height = intercept + slope × **Weight**

So let's learn how **Gradient Descent** can fit a line to data by finding the optimal values for the **Intercept** and the **Slope**.

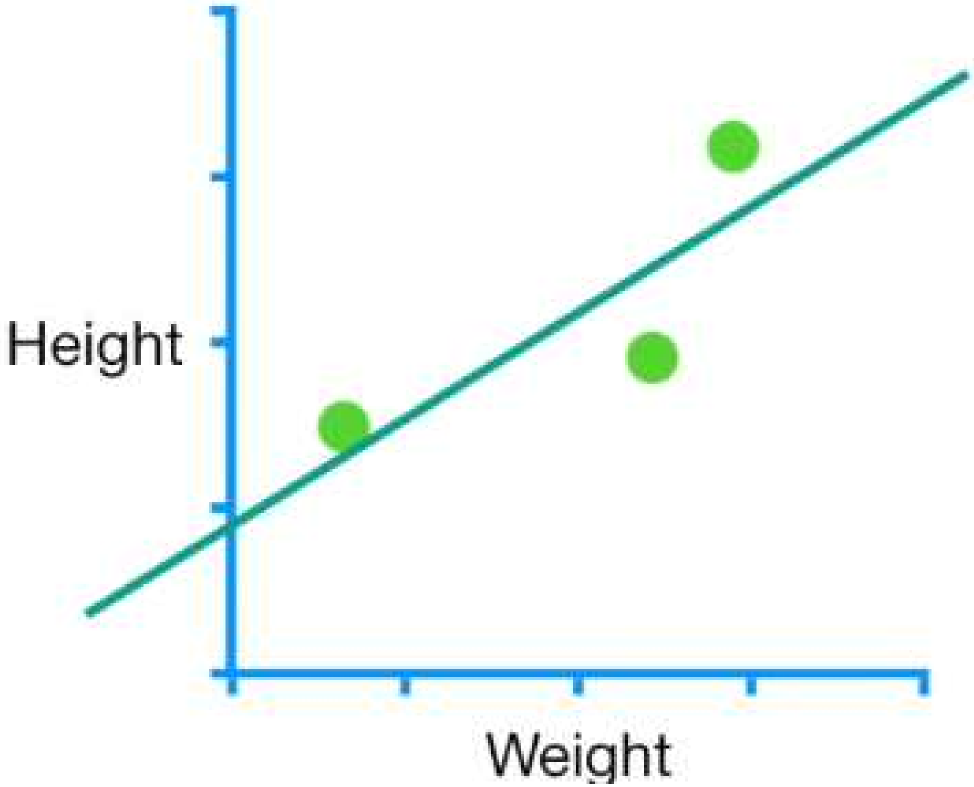


Predicted Height = intercept + slope × **Weight**

Actually, we'll start by using **Gradient Descent** to find the **Intercept**.



Predicted Height = $\boxed{\text{intercept}}$ + $\boxed{\text{slope}}$ × **Weight**

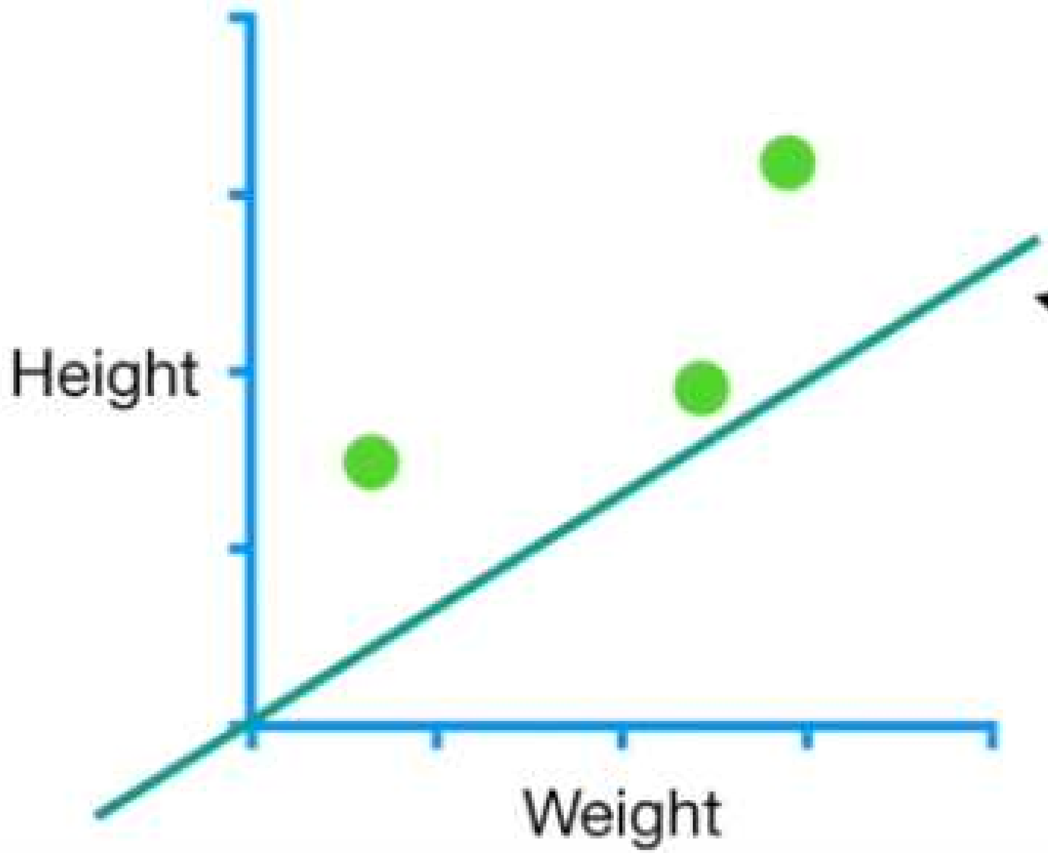


Then, once we understand how **Gradient Descent** works, we'll use it to solve for the **Intercept** *and* the **Slope**.

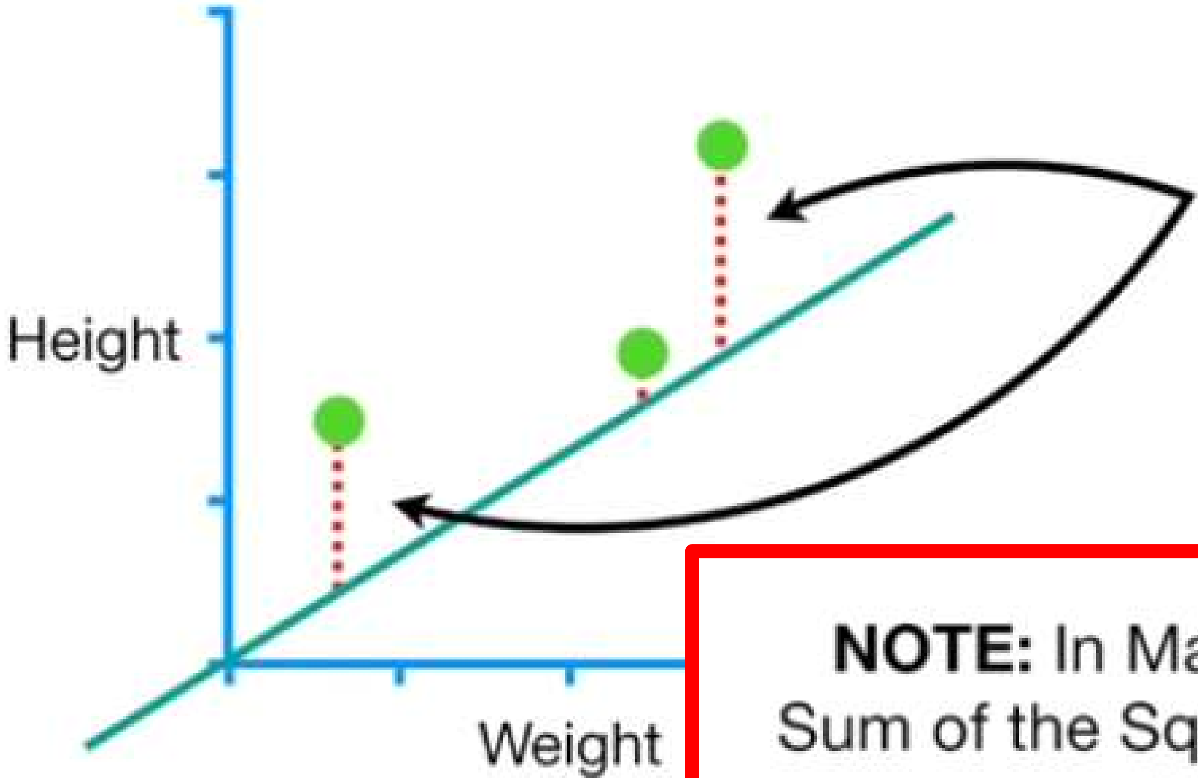
일단 출발은 Slope가 0.64라고 알고 있다고 생각을 하고, 이 때의 최적의 intercept를 찾는 경우를 먼저 알아보자!!!!

Predicted Height = $\boxed{0}$ + 0.64 × **Weight**

일단
Intercept를
0일 때 알아보자



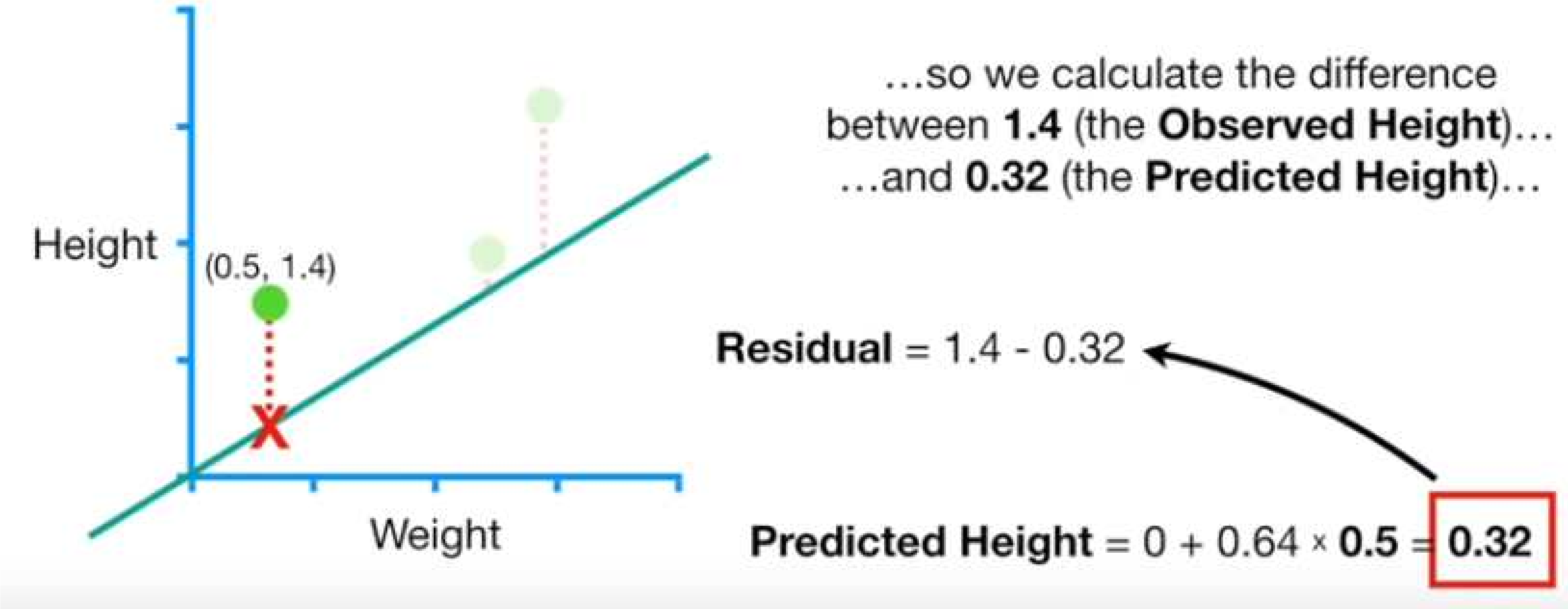
And that gives us the equation for this line.

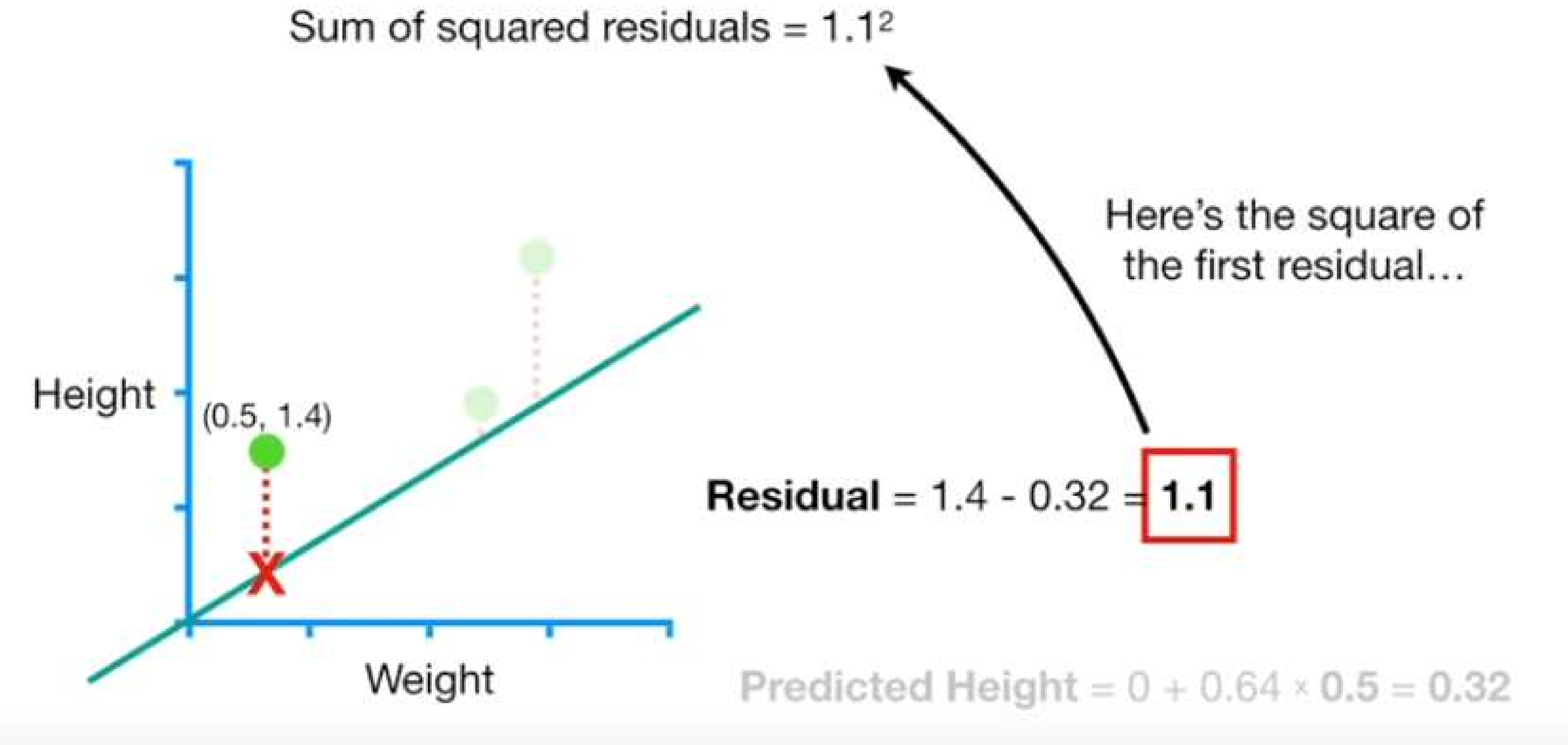


In this example, we will evaluate how well this line fits the data with the **Sum of the Squared Residuals**.

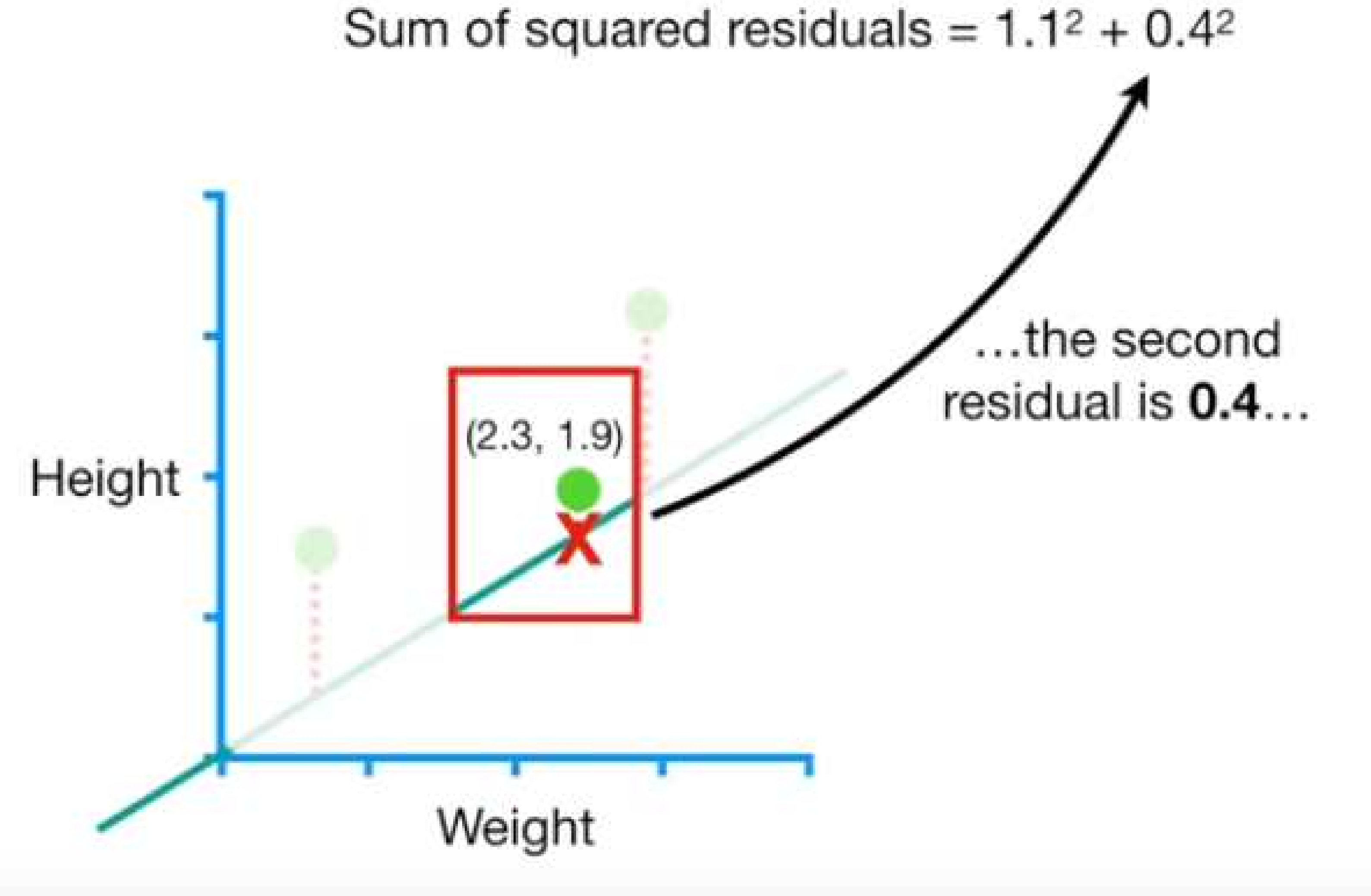
NOTE: In Machine Learning lingo, The Sum of the Squared Residuals is a type of **Loss Function**.

Step01) 1개의 데이터에 대해서 주어진 조건 하의 예측값과 실제 값의 차이를 계산을 시작

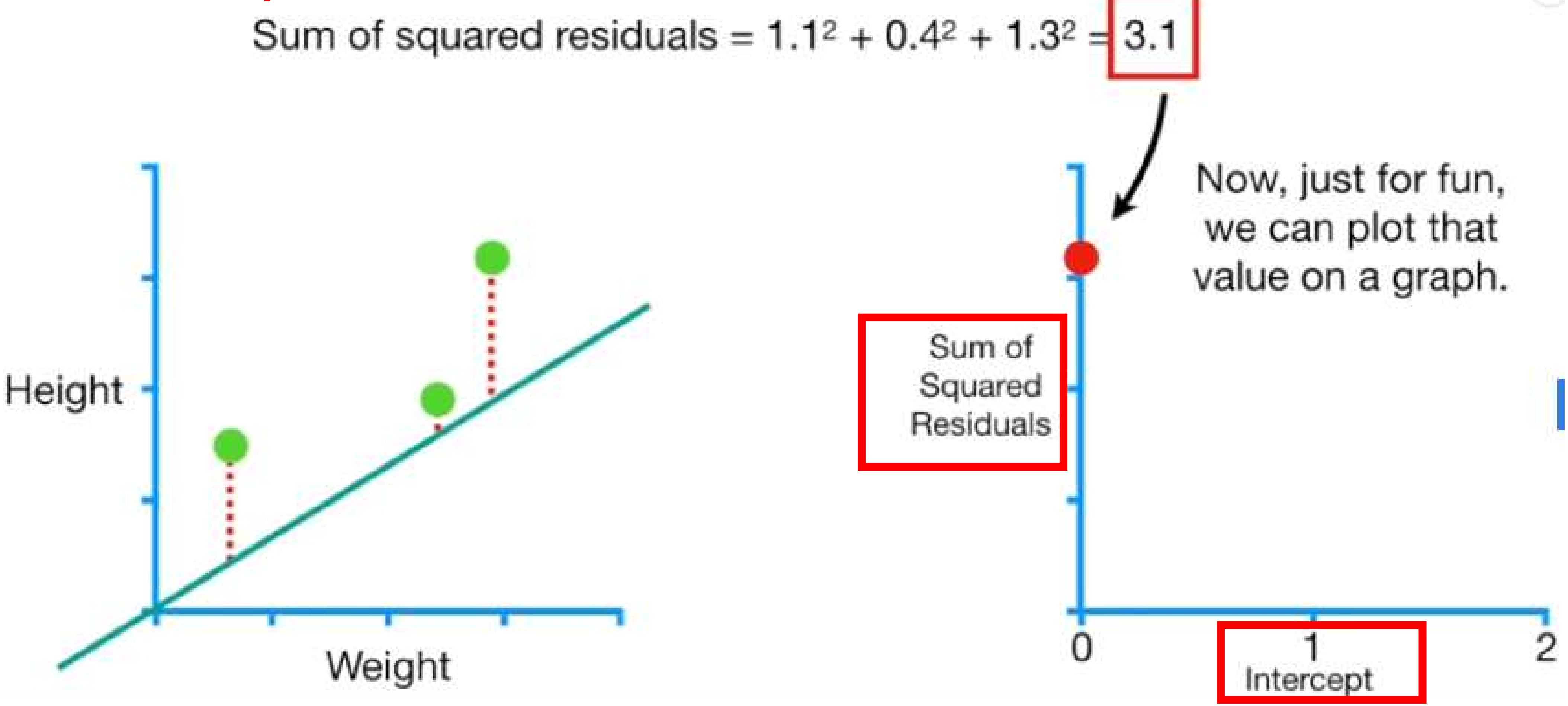



Sum of squared residuals = 1.1²
Here's the square of
the first residual…
Height
(0.5, 1.4)
Weight
Residual = 1.4 - 0.32 = 1.1
Predicted Height = 0 + 0.64 × 0.5 = 0.32

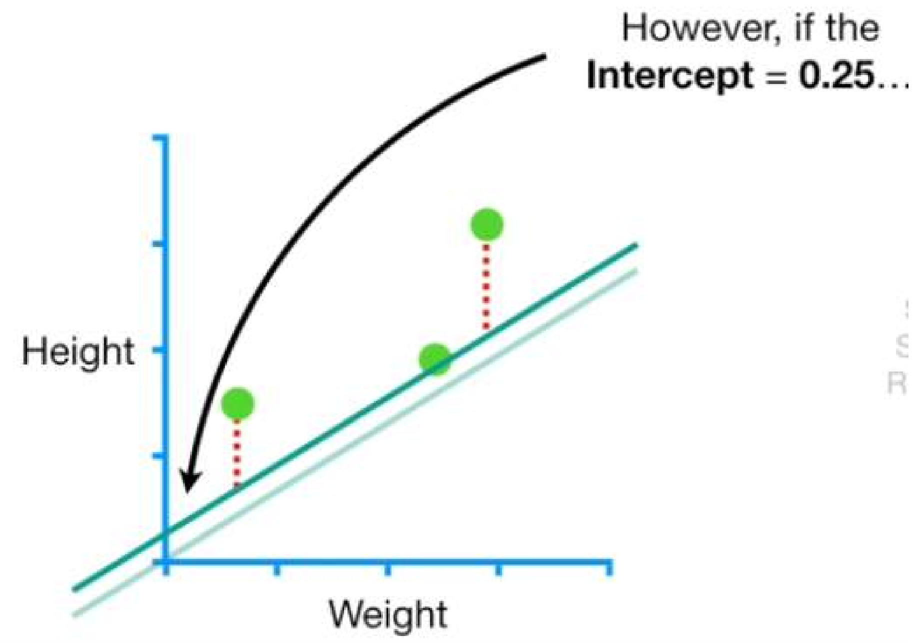
Step02) 모든 데이터들에 대해서 각기 residuals의 제곱의 합을 구해나가기!

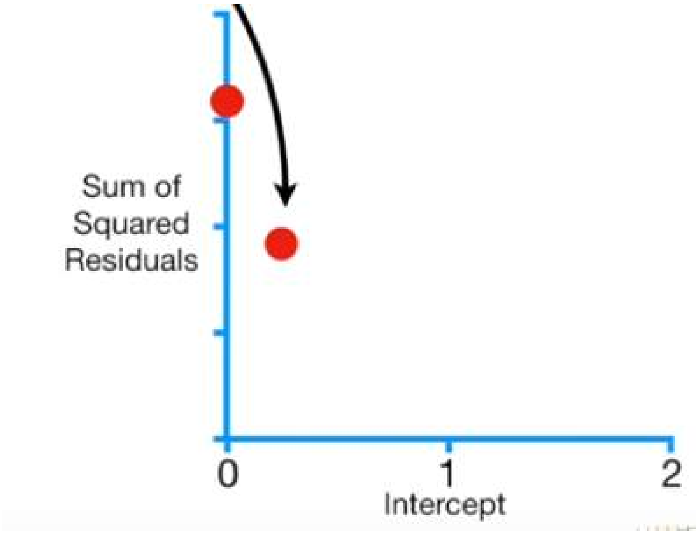


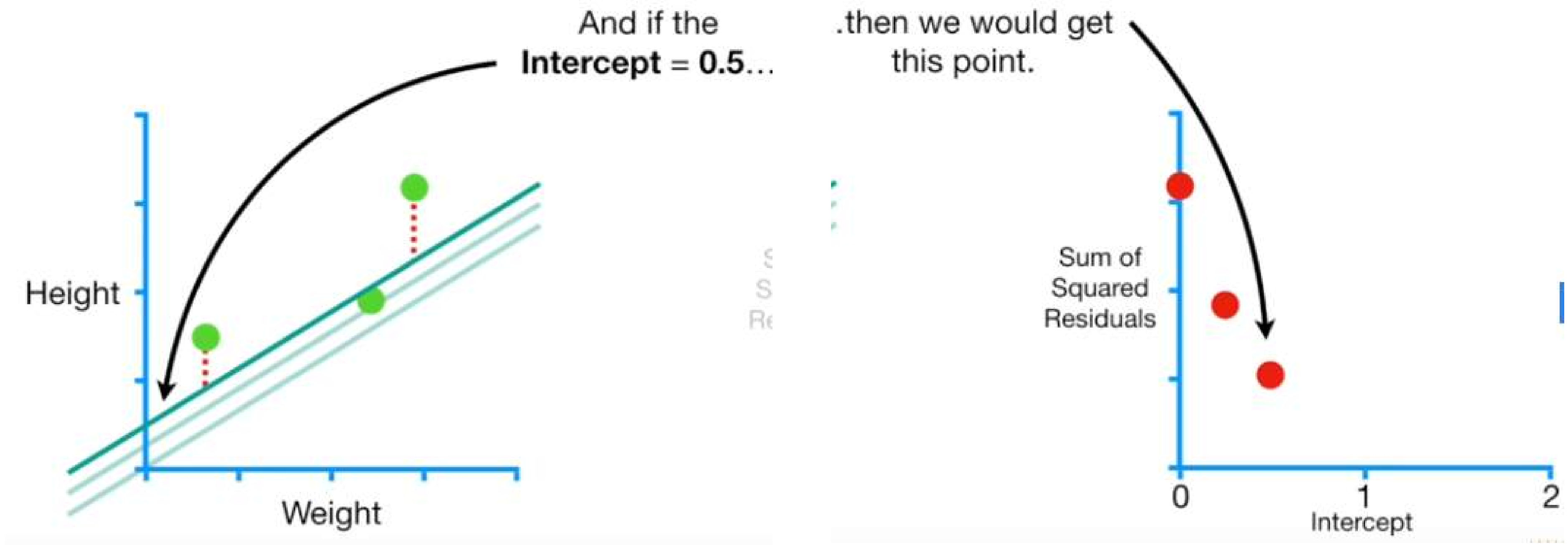
Step03) 모든 데이터들의 residuals의 제곱의 합을 구하고 나서 Intercept의 값과 오차제곱의 합에 대한 것을 그래프로!!!!



Step04) 이제 intercept의 값을 바꾸어 가면서 앞에서 한 step01~03의 과정을 모두 다시 해보기!!!

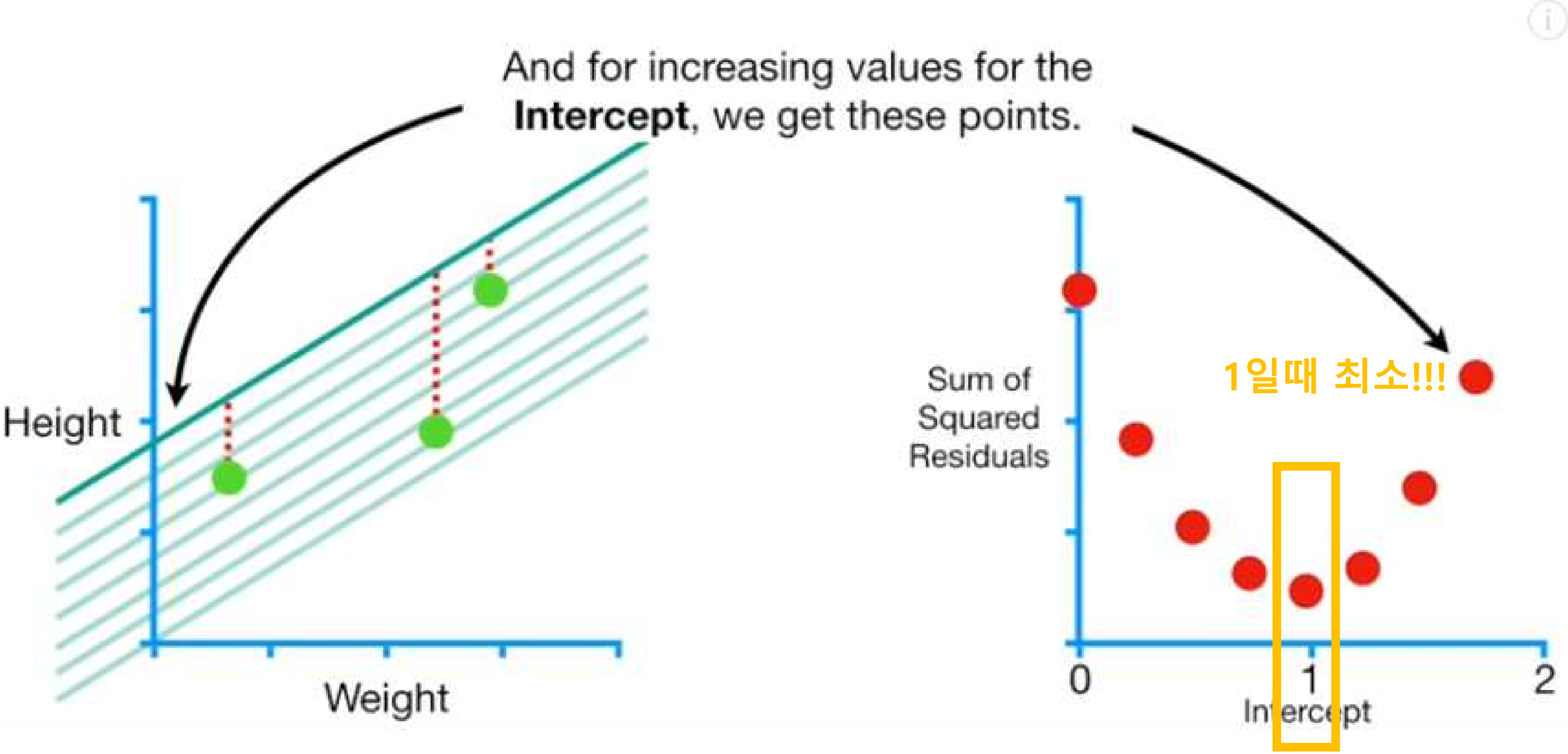






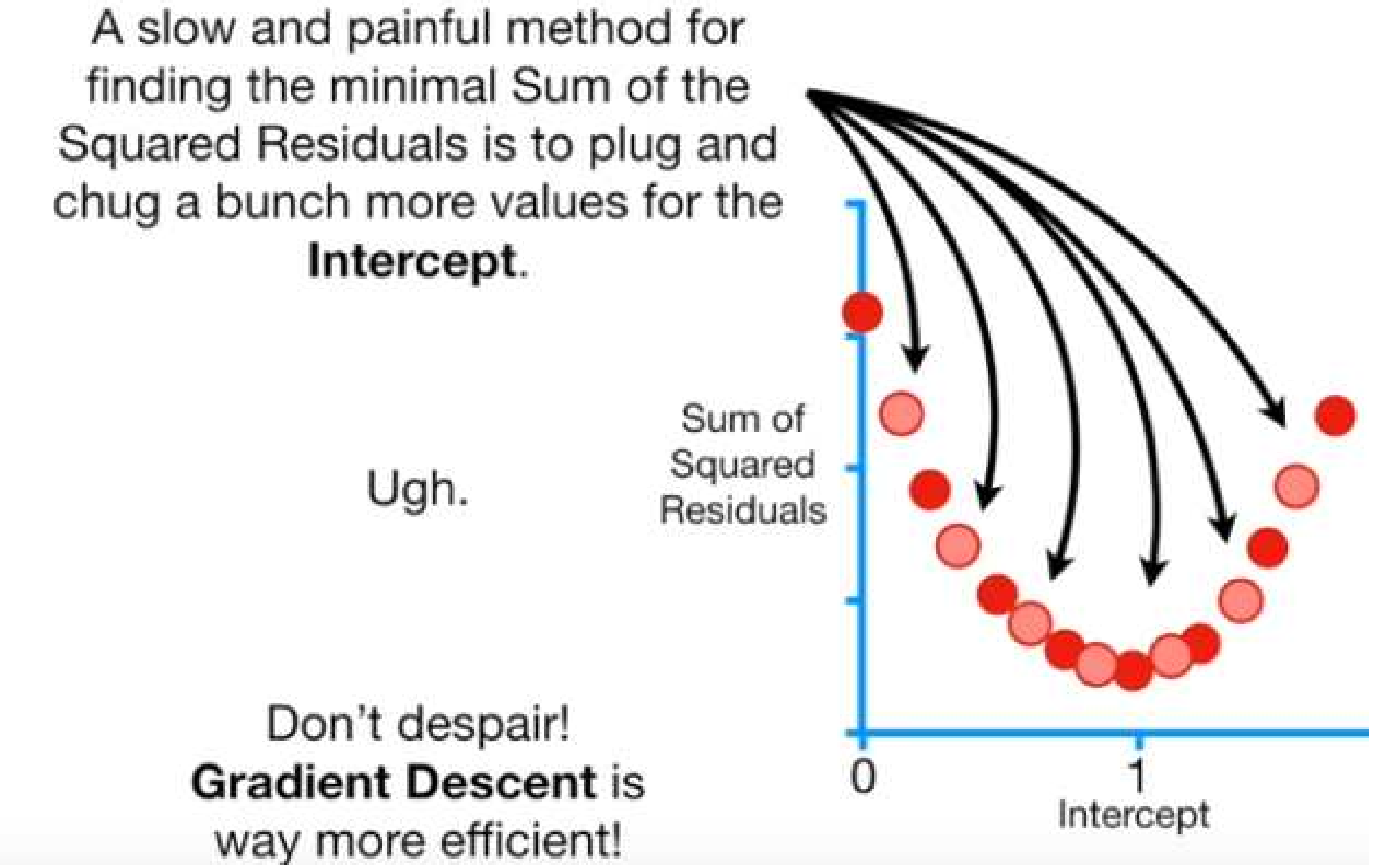
And if the
Intercept = **0.5**…
…then we would get
this point.
Height
Weight
Sum of
Squared
Residuals
0
1
2
Intercept

Step05) 앞의 과정을 여러 intercept에 대해서 수행한 결과 -> 제일 작은 값을 만드는 intercept 의 값을 찾기!!!!!

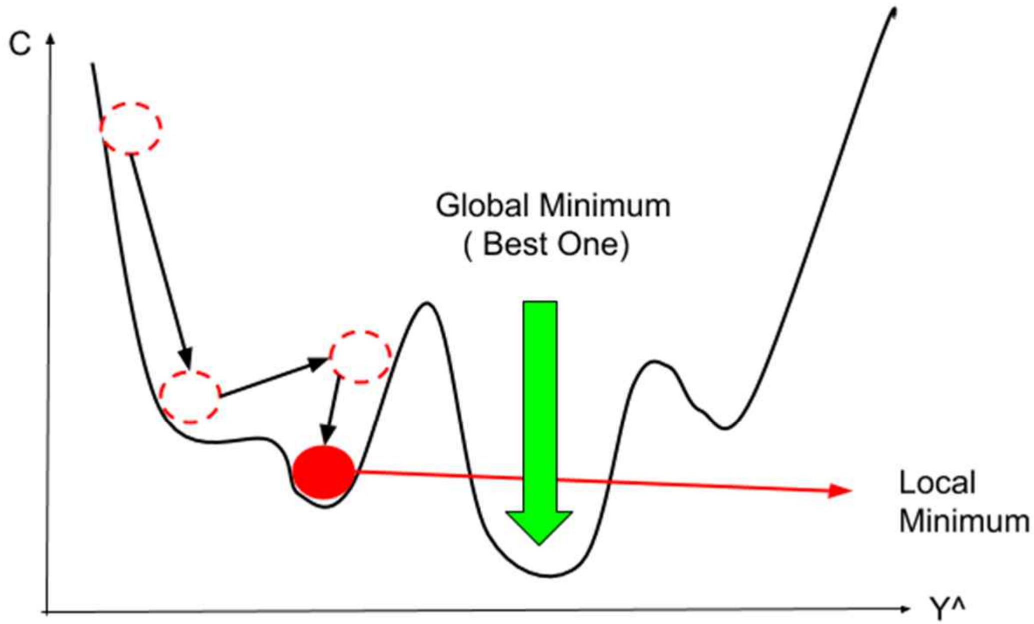


But) 어떻게 이렇게 하나씩 노가다로 해야하는 것인가;;;

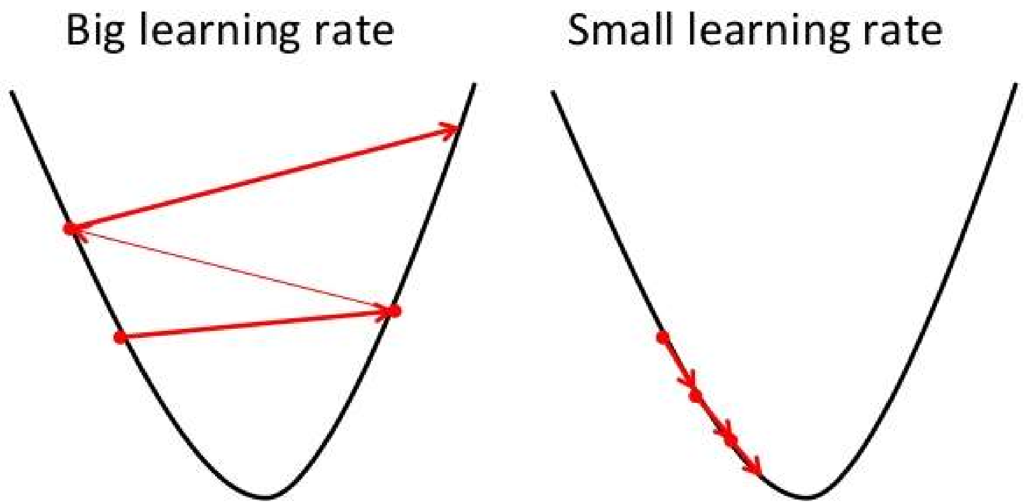
→ 이러한 경우에 나타난 것이 Gradient Descent 가 된다!!!!!!!!!!



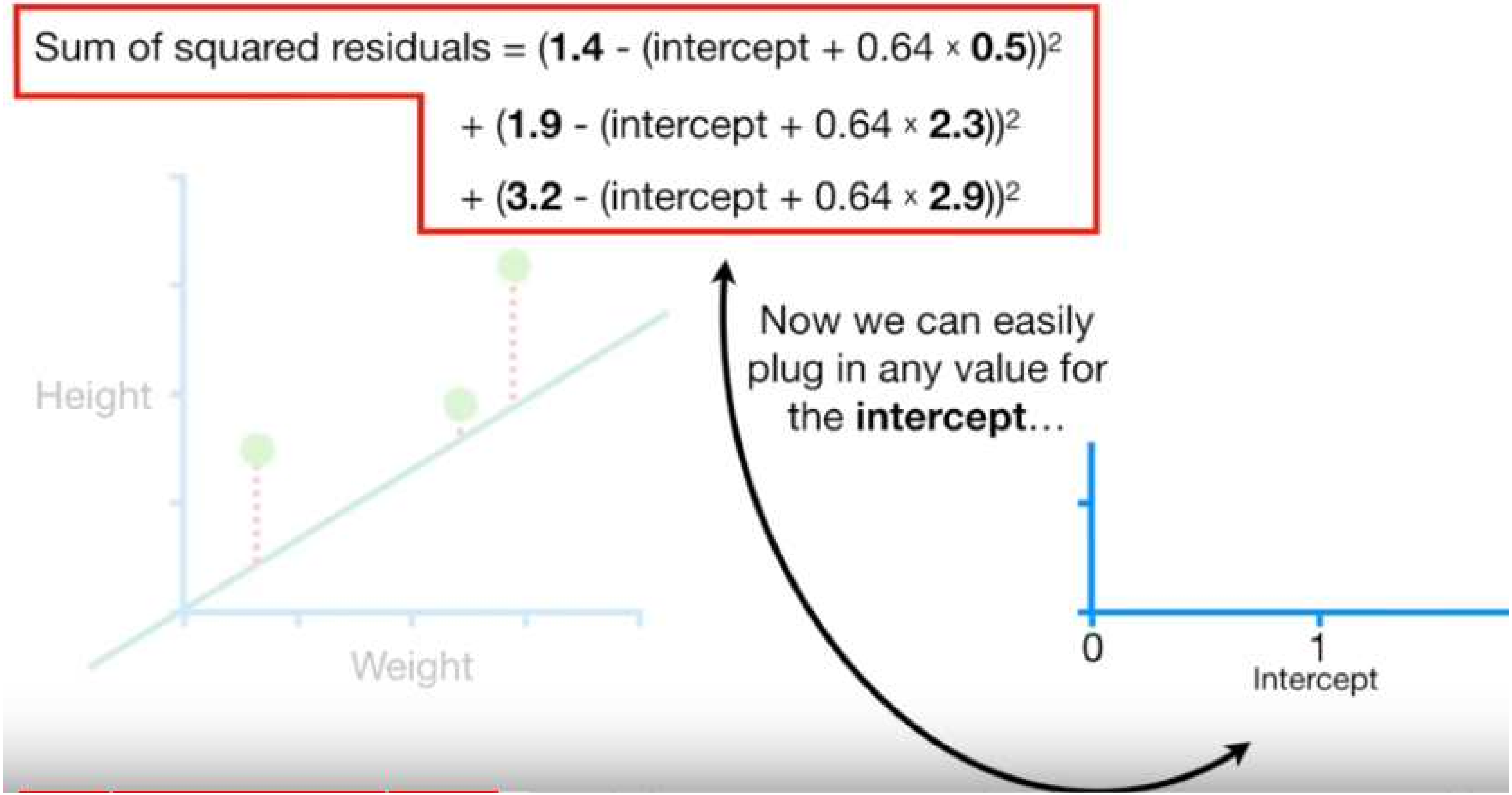

Gradient
Descent



Gradient Descent

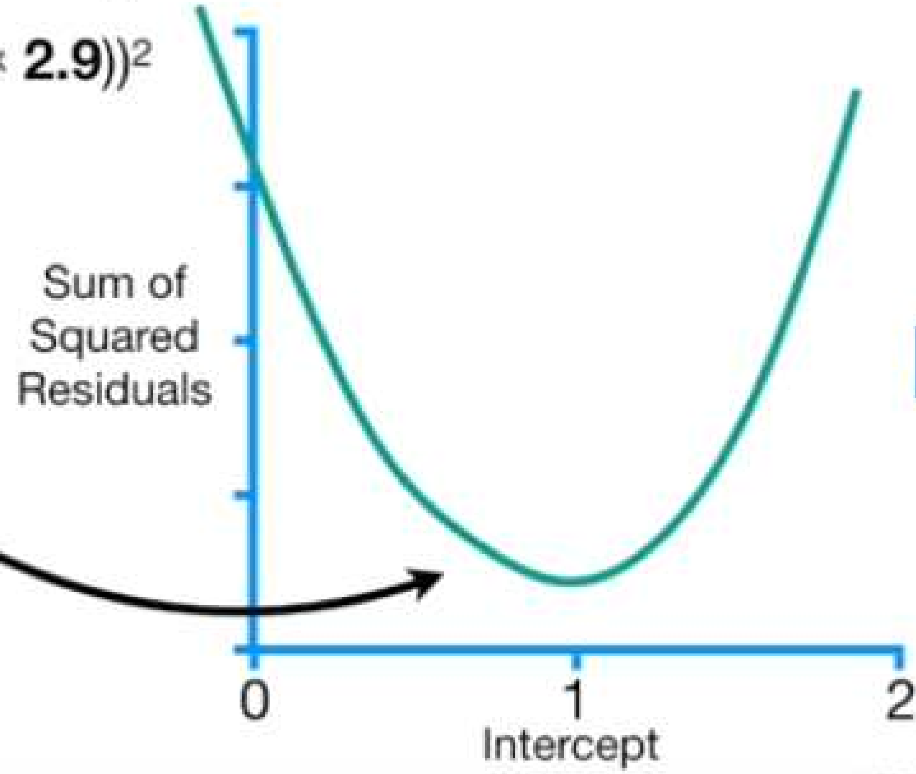


* 앞에서 직접 계산 값을 intercept의 식으로 표현해보자!!!!!

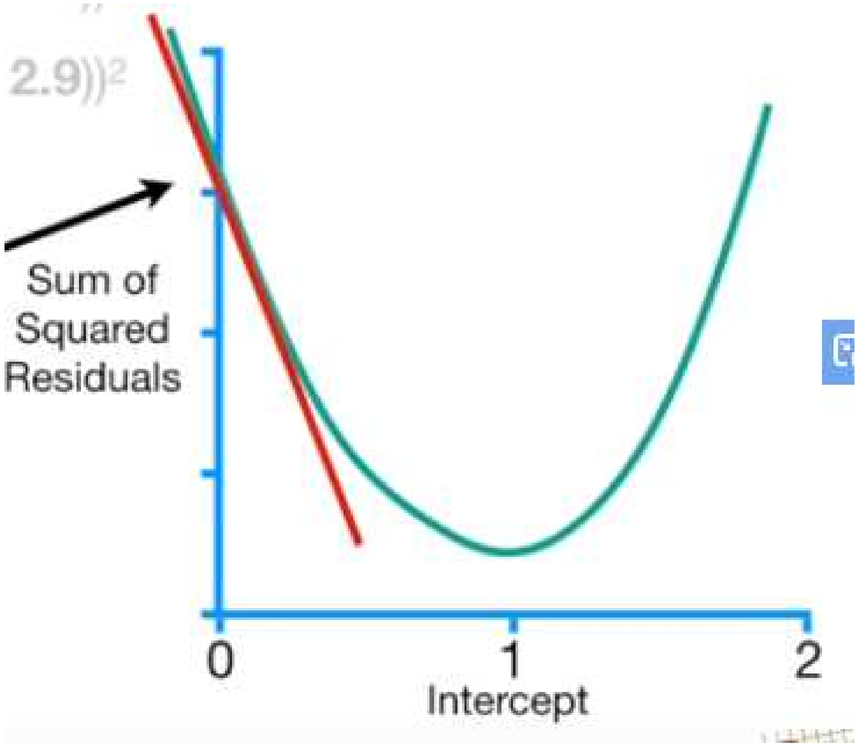


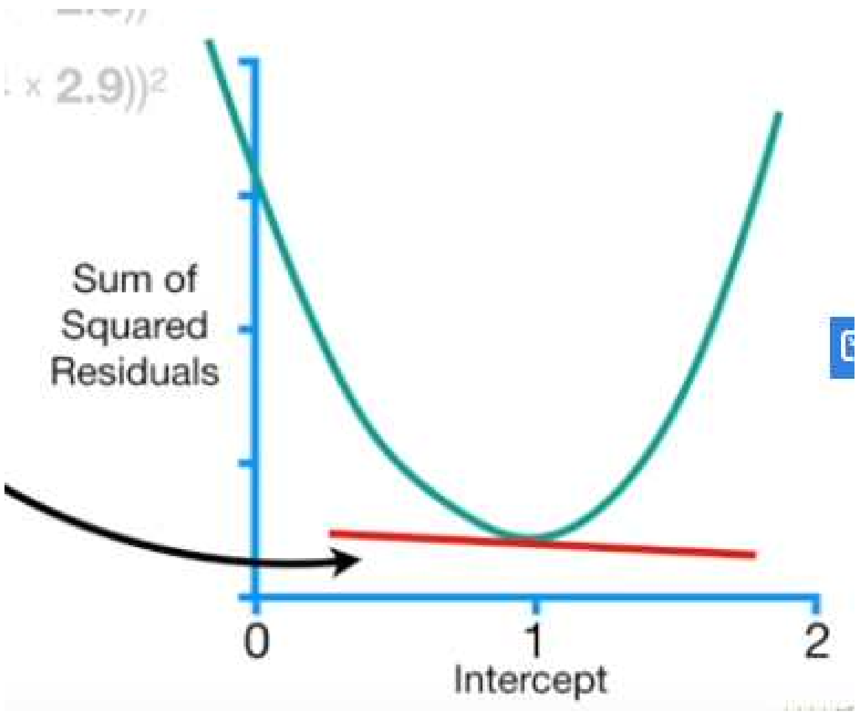
$$\text{Sum of squared residuals} = (\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5}))^2$$
$$+ (\mathbf{1.9} - (\text{intercept} + 0.64 \times \mathbf{2.3}))^2$$
$$+ (\mathbf{3.2} - (\text{intercept} + 0.64 \times \mathbf{2.9}))^2$$

Thus, we now have an equation for this curve…



*** 이제 어디서 최소가 될지에 대한 것은 우리가 고딩때 배운 “미분 " 이 나타날 시기가 됨!!!!**





$$\text{Sum of squared residuals} = (\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5}))^2 + (\mathbf{1.9} - (\text{intercept} + 0.64 \times \mathbf{2.3}))^2 + (\mathbf{3.2} - (\text{intercept} + 0.64 \times \mathbf{2.9}))^2$$

미분

$$\frac{d}{d\ \textit{intercept}} \text{Sum of squared residuals} = \boxed{\frac{d}{d\ \textit{intercept}} (\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5}))^2} + \frac{d}{d\ \textit{intercept}} (\mathbf{1.9} - (\text{intercept} + 0.64 \times \mathbf{2.3}))^2 + \frac{d}{d\ \textit{intercept}} (\mathbf{3.2} - (\text{intercept} + 0.64 \times \mathbf{2.9}))^2$$

1개의 항목에 대해서 좀 자세히 알아보자!!!!

$$\frac{d}{d\ intercept}\boxed{(\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5}))}^2 = 2(\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5}))\boxed{\times -1}$$

$$\frac{d}{d\ intercept}\ \mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5})$$

$$\frac{d}{d\ intercept}\ \mathbf{1.4} + (-1)\text{intercept} - 0.64 \times \mathbf{0.5} = \boxed{-1}$$

$$\frac{d}{d\ intercept}\text{ Sum of squared residuals} = \boxed{\frac{d}{d\ intercept}(\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5}))^2}$$

$$+ \frac{d}{d\ intercept}(\mathbf{1.9} - (\text{intercept} + 0.64 \times \mathbf{2.3}))^2$$

$$+ \frac{d}{d\ intercept}(\mathbf{3.2} - (\text{intercept} + 0.64 \times \mathbf{2.9}))^2$$

$$\frac{d}{d\ intercept}\text{ Sum of squared residuals} = -2(\mathbf{1.4}\ (\text{intercept} + 0.64 \times \mathbf{0.5})$$

$$\boxed{\begin{aligned} &+ \frac{d}{d\ intercept}(\mathbf{1.9} - (\text{intercept} + 0.64 \times \mathbf{2.3}))^2 \\ &+ \frac{d}{d\ intercept}(\mathbf{3.2} - (\text{intercept} + 0.64 \times \mathbf{2.9}))^2 \end{aligned}}$$

$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals} = -2(\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5}))$$

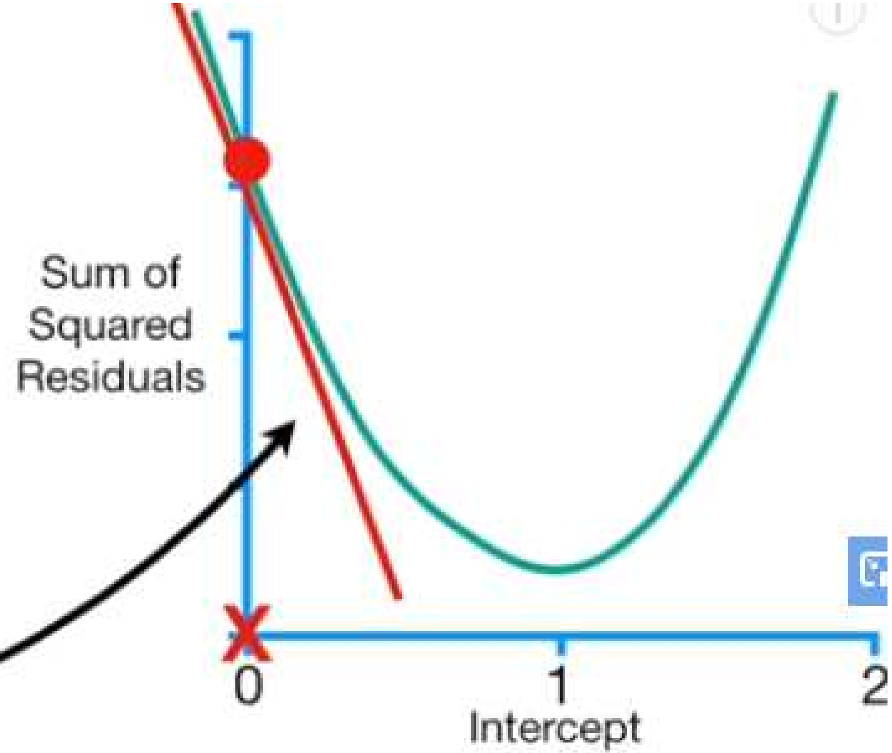
$$+ -2(\mathbf{1.9} - (\text{intercept} + 0.64 \times \mathbf{2.3}))$$

$$+ -2(\mathbf{3.2} - (\text{intercept} + 0.64 \times \mathbf{2.9}))$$

미분식과 그래프의 의미 보자

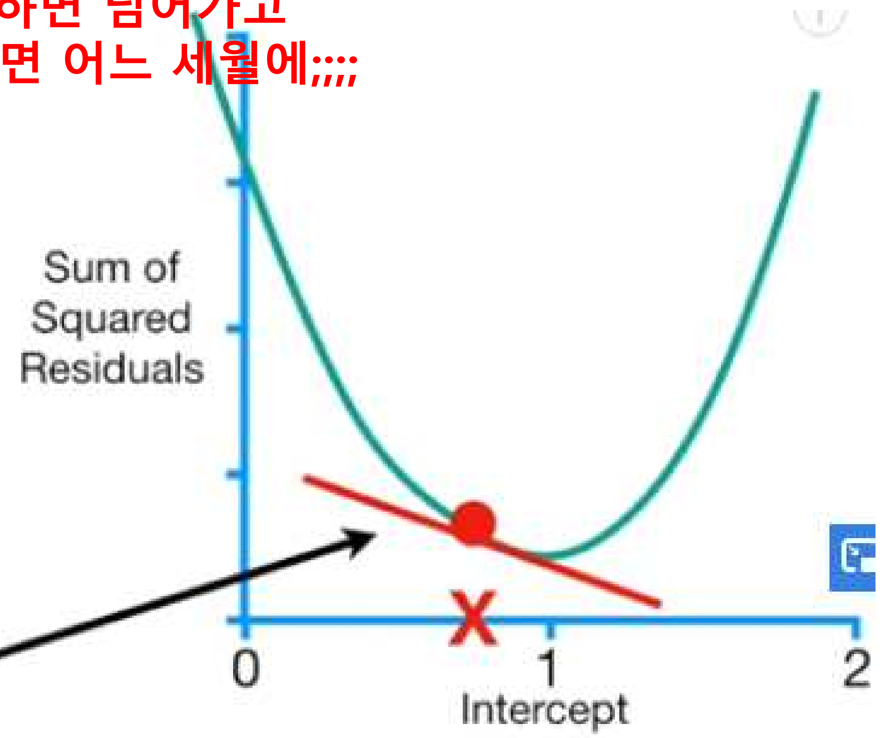
$$\frac{d}{d\ intercept} \text{Sum of squared residuals} =$$
$$-2(\mathbf{1.4} - (0 + 0.64 \times \mathbf{0.5}))$$
$$+ -2(\mathbf{1.9} - (0 + 0.64 \times \mathbf{2.3}))$$
$$+ -2(\mathbf{3.2} - (0 + 0.64 \times \mathbf{2.9}))$$
$$= -5.7$$

So when the **Intercept** = **0**,
the slope of the curve = **-5.7**.



0부터 시작을 하니
최적의 값이 아니다.
→ 이 때 큰 스탭으로 이동하면 넘어가고
→ 너무 작은 스탭으로 이동하면 어느 세월에;;;;

$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals} = -2(\mathbf{1.4} - (0 + 0.64 \times \mathbf{0.5}) + -2(\mathbf{1.9} - (0 + 0.64 \times \mathbf{2.3})) + -2(\mathbf{3.2} - (0 + 0.64 \times \mathbf{2.9})) = -5.7$$



NOTE: The closer we get to the optimal value for the **Intercept**, the closer the slope of the curve gets to **0**.

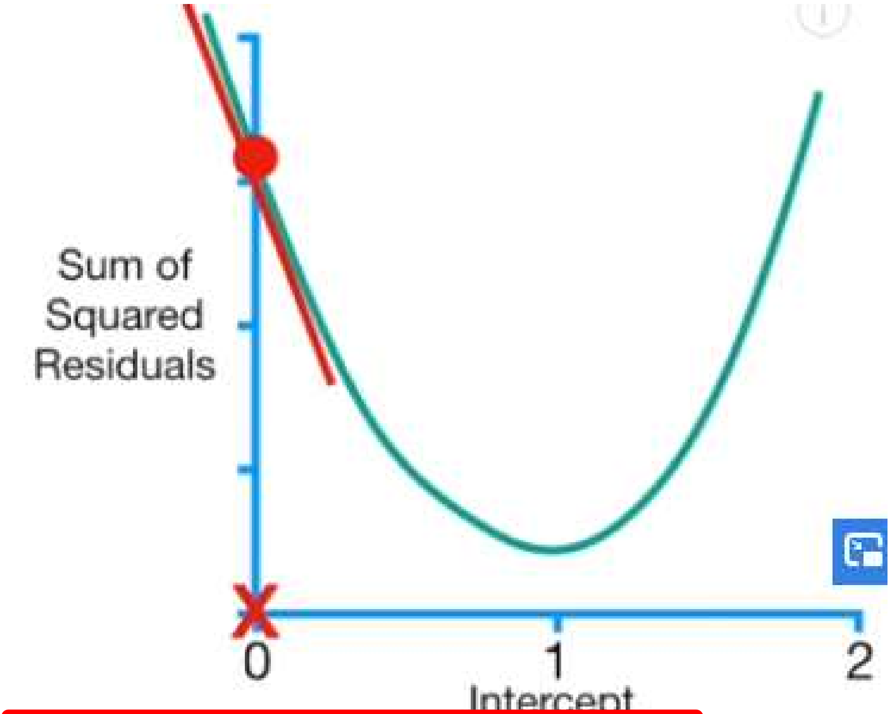
$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals} = -2(\mathbf{1.4} - (0 + 0.64 \times \mathbf{0.5}))$$

$$+ -2(\mathbf{1.9} - (0 + 0.64 \times \mathbf{2.3}))$$

$$+ -2(\mathbf{3.2} - (0 + 0.64 \times \mathbf{2.9}))$$

$$\boxed{= -5.7}$$

Step Size = -5.7



Gradient Descent determines the **Step Size** by multiplying the **slope**…

Step Size = -5.7 × 0.1

…by a small number called **The Learning Rate**.

$$\frac{d}{d\ intercept} \text{ Sum of squared residuals} =$$

$$-2(\mathbf{1.4} - (0 + 0.64 \times \mathbf{0.5})$$

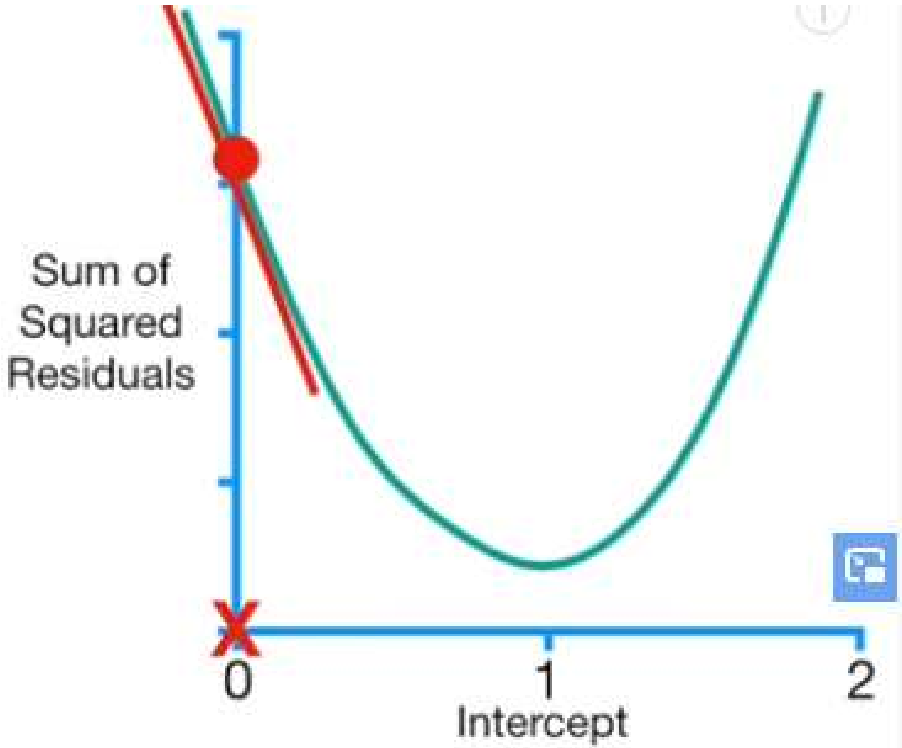
$$+ -2(\mathbf{1.9} - (0 + 0.64 \times \mathbf{2.3}))$$

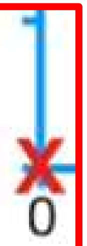
$$+ -2(\mathbf{3.2} - (0 + 0.64 \times \mathbf{2.9}))$$

$$= -5.7$$

Step Size = -5.7 × 0.1 = -0.57

When the **Intercept** = **0**, the **Step Size** = **-0.57**.





Step Size = -5.7 × 0.1 = $\boxed{-0.57}$

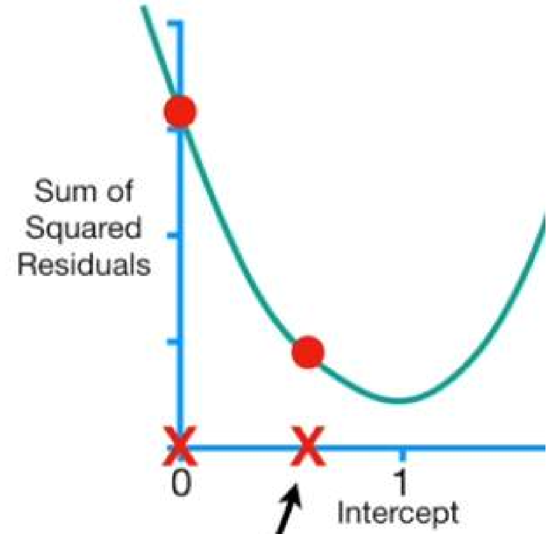
New Intercept = Old Intercept - Step Size

...minus the **Step Size**.

+ -2(**3.2** - (0 + 0.64 × **2.9**))

= -5.7

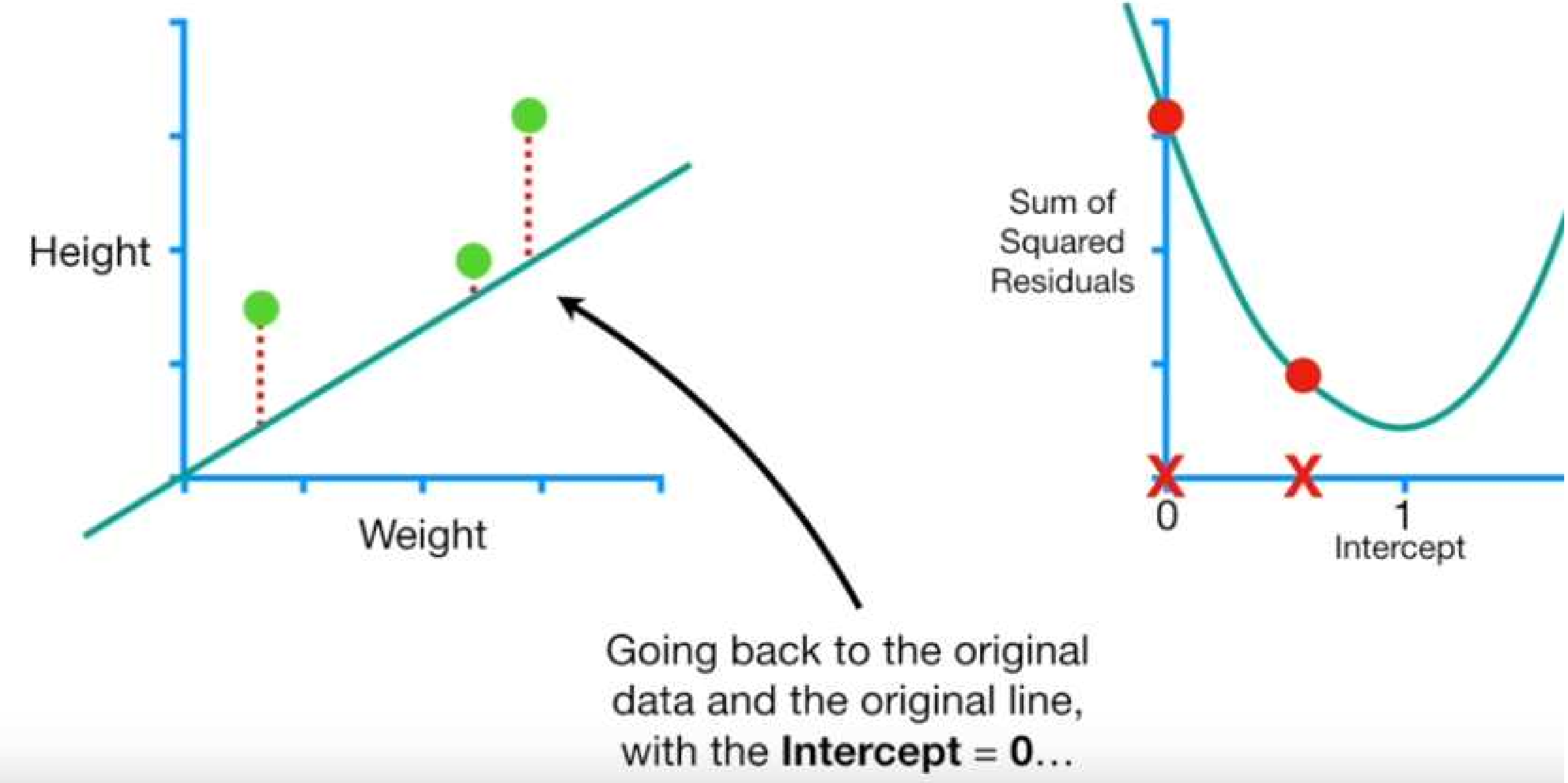
Step Size = -5.7 × 0.1 = -0.57



New Intercept = 0 - (-0.57) = $\boxed{\mathbf{0.57}}$

...and the the **New Intercept** = **0.57**.

이제 다시 0으로 부터 시작해보자!!!!!

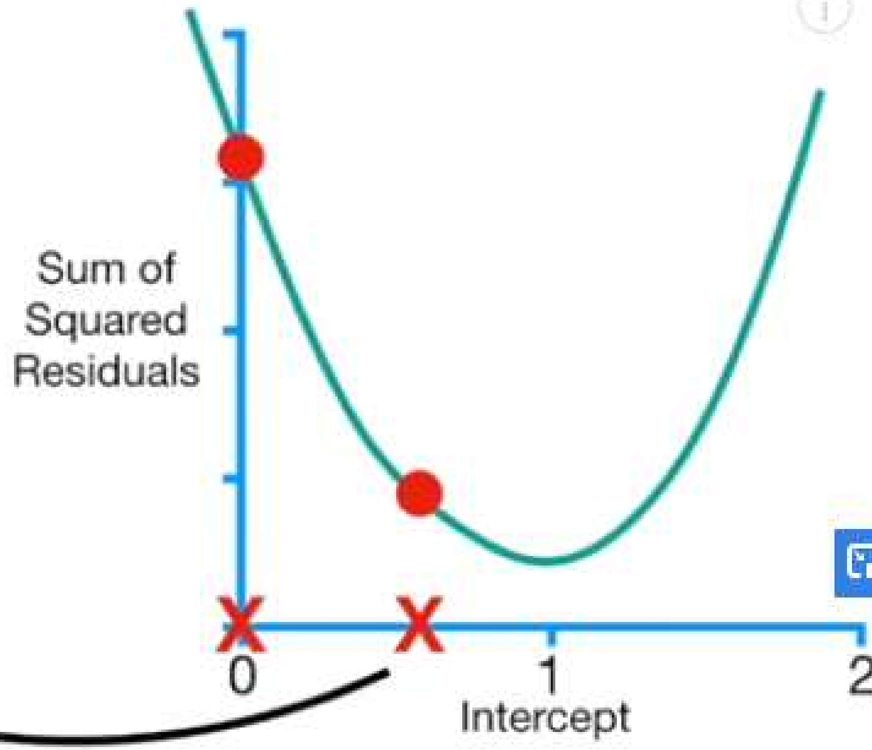


0일 때 값을 계산하고
Learning Rate인 0.1을 고려해서
0 -5.7 * (-0.1) = 0.57에서 다시 기울기 계산을..

$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals} = -2(\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5})) + -2(\mathbf{1.9} - (\text{intercept} + 0.64 \times \mathbf{2.3})) + -2(\mathbf{3.2} - (\text{intercept} + 0.64 \times \mathbf{2.9}))$$

$$\boxed{= -5.7}$$

To take another step, we go back to the derivative and plug in the **New Intercept** (**0.57**)…

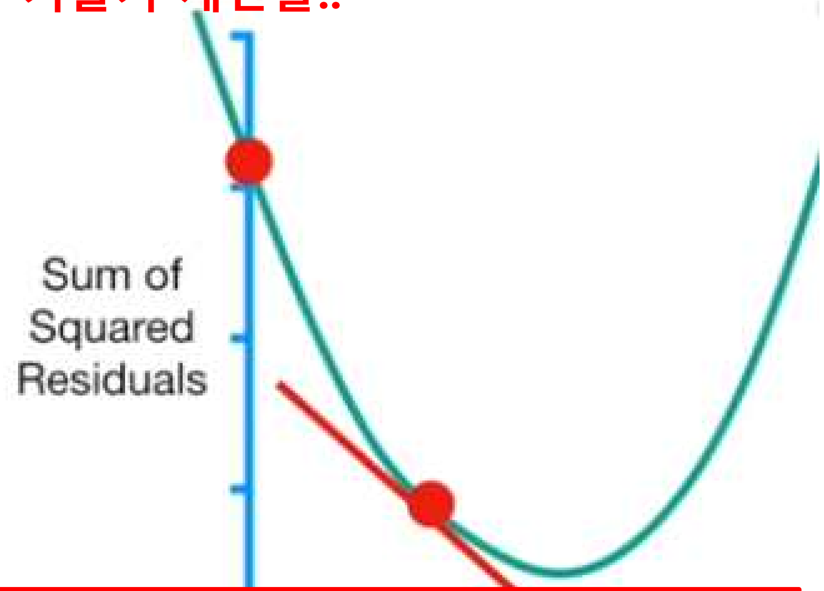


0.57일 때 값을 계산하고
Learning Rate인 0.1을 고려해서
0.57 -2.3 * (-0.1) = 0.80에서 다시 기울기 계산을..

$$\frac{d}{d\ intercept}\text{ Sum of squared residuals} = -2(\mathbf{1.4} - (0.57 + 0.64 \times \mathbf{0.5})) + -2(\mathbf{1.9} - (0.57 + 0.64 \times \mathbf{2.3})) + -2(\mathbf{3.2} - (0.57 + 0.64 \times \mathbf{2.9})) = \boxed{-2.3}$$

Step Size = -2.3 × **Learning Rate**

...by plugging in **-2.3** for the **Slope**...



Step Size = -2.3 × 0.1 = $\boxed{\mathbf{-0.23}}$

New Intercept = 0.57 - **Step Size**

...and the **New Intercept**...

$$\frac{d}{d\ intercept} \text{Sum of squared residuals} =$$

$$-2(\mathbf{1.4} - (0.8 + 0.64 \times \mathbf{0.5})$$

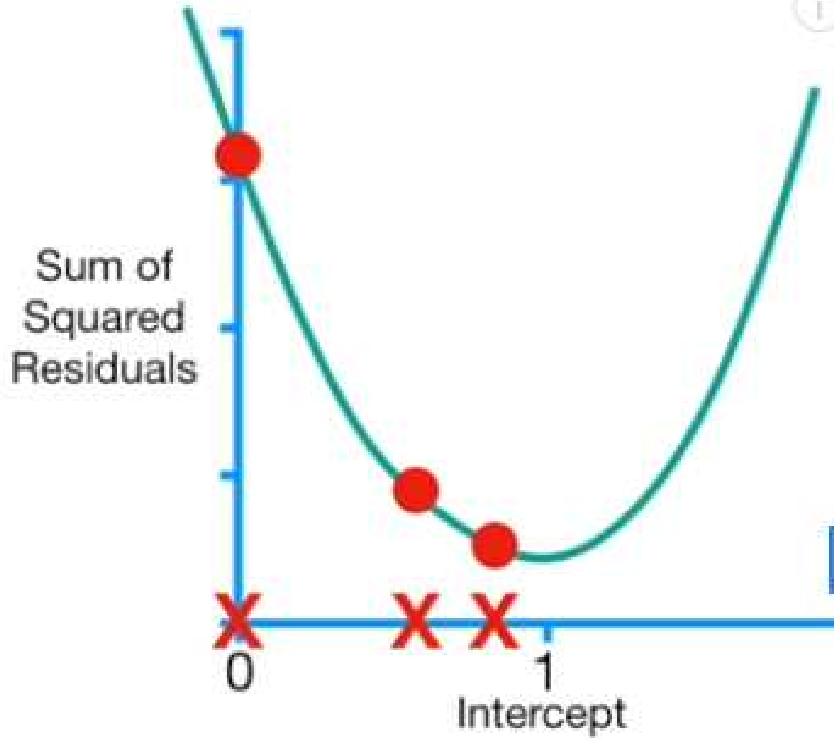
$$+ -2(\mathbf{1.9} - (0.8 + 0.64 \times \mathbf{2.3}))$$

$$+ -2(\mathbf{3.2} - (0.8 + 0.64 \times \mathbf{2.9}))$$

$$= \mathbf{-0.9}$$

Step Size = **Slope** × **Learning Rate**

The **Step Size**…



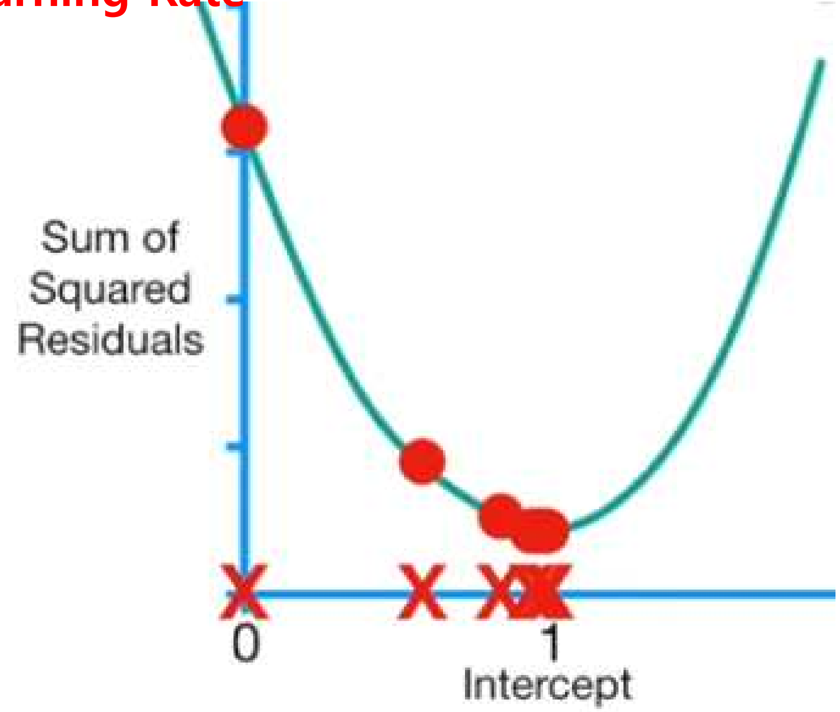
이런 방식으로 계속 진행을...

그러면 언제까지 진행을 해야하는 것일까?+??

→ Step Size가 거의 0 에 가까이 가면 그만하자고 할 수 있다!!!!!!!
→ Step Size = Slope * Learning Rate

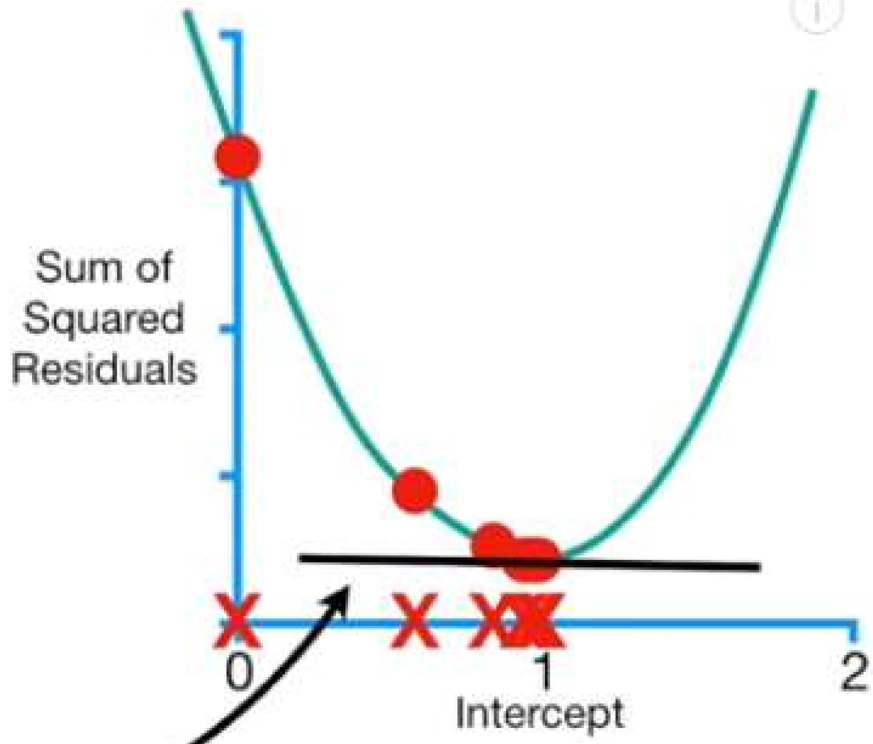
After 6 steps, the **Gradient Descent** estimate for the **Intercept** is **0.95**.

NOTE: The **Least Squares** estimate for the intercept is also **0.95**.



The **Step Size** will be **Very Close to 0** when the **Slope** is very close to **0**.

Step Size = $\boxed{\textbf{Slope}}$ × **Learning Rate**



이제 제대로 전체를 해보자!!

- 앞의 예제가 가지는 한계점
  - lope가 앞에서 수행한 값이 정말로 최적일까?
  - Intercept / Slope 모두 고려해서 최적의 잔차제곱의 합을 구해보자!!

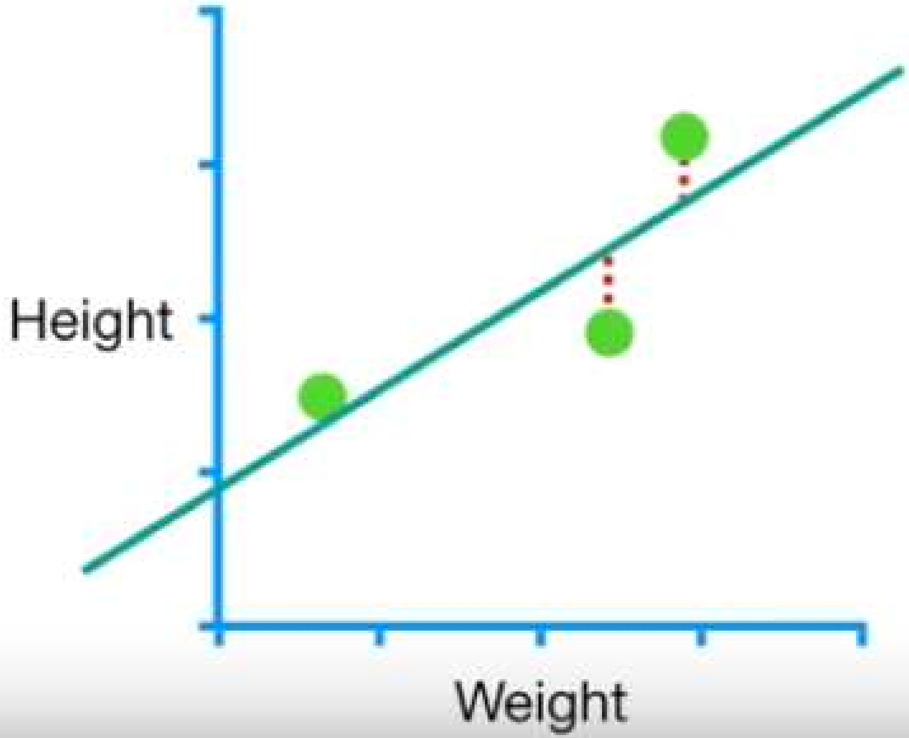
- 위의 경우 Full로 하게 되면 미지의 변수/우리가 찾아야 할 파라미터 2개에 대한 것이므로 3차원에서 그려야 한다!!!

- 이것을 ML에서는 Loss Function / Cost Function의 입장이 된다!!!

$$\text{Sum of squared residuals} = (\mathbf{1.4} - (\text{intercept} + \text{slope} \times \mathbf{0.5}))^2$$

$$+ (\mathbf{1.9} - (\text{intercept} + \text{slope} \times \mathbf{2.3}))^2$$

$$+ (\mathbf{3.2} - (\text{intercept} + \text{slope} \times \mathbf{2.9}))^2$$



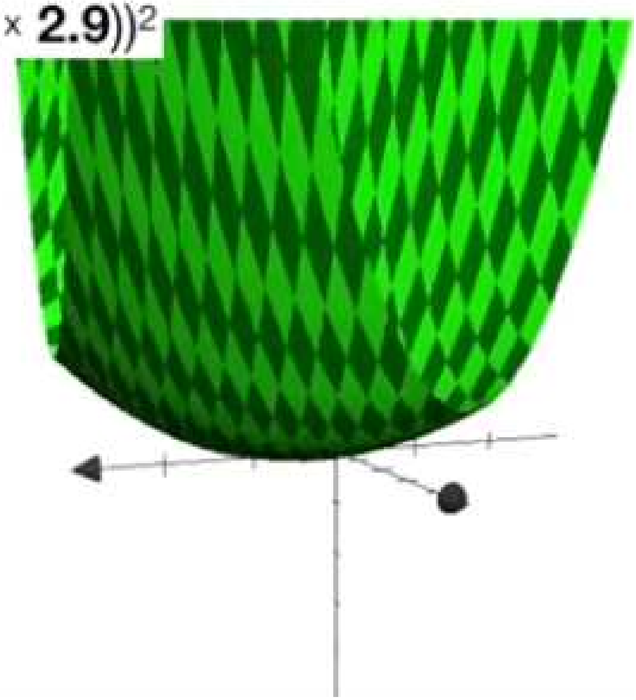
Just like before, we will use the Sum of the Squared Residuals as the **Loss Function**

Sum of squared residuals = $(\mathbf{1.4} - (\text{intercept} + \text{slope} \times \mathbf{0.5}))^2$

$$+ (\mathbf{1.9} - (\text{intercept} + \text{slope} \times \mathbf{2.3}))^2$$

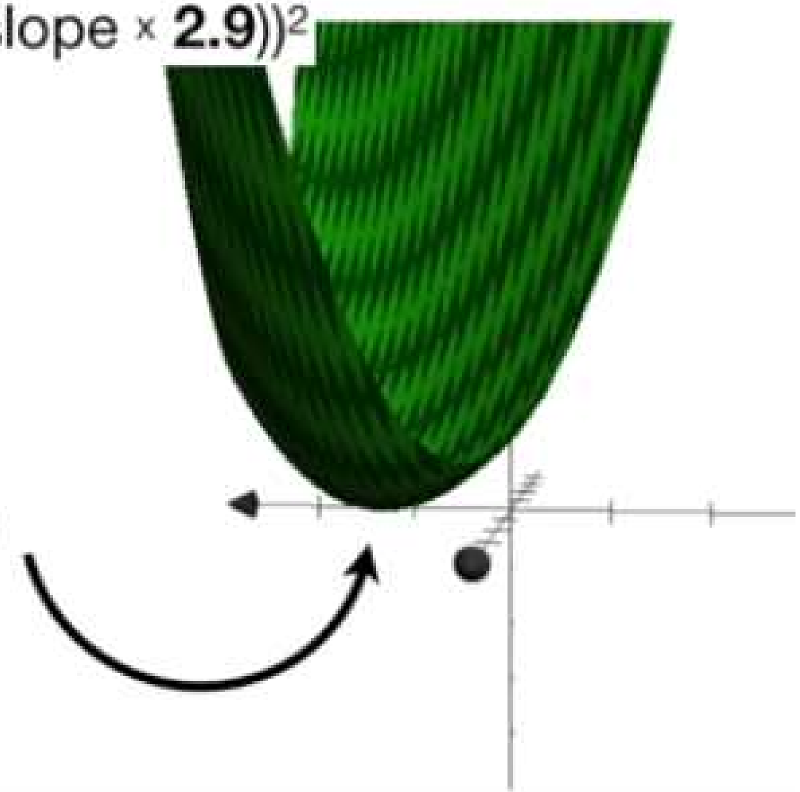
$$+ (\mathbf{3.2} - (\text{intercept} + \text{slope} \times \mathbf{2.9}))^2$$

This is a 3-D graph of the **Loss Function** for different values for the **Intercept** and the **Slope**

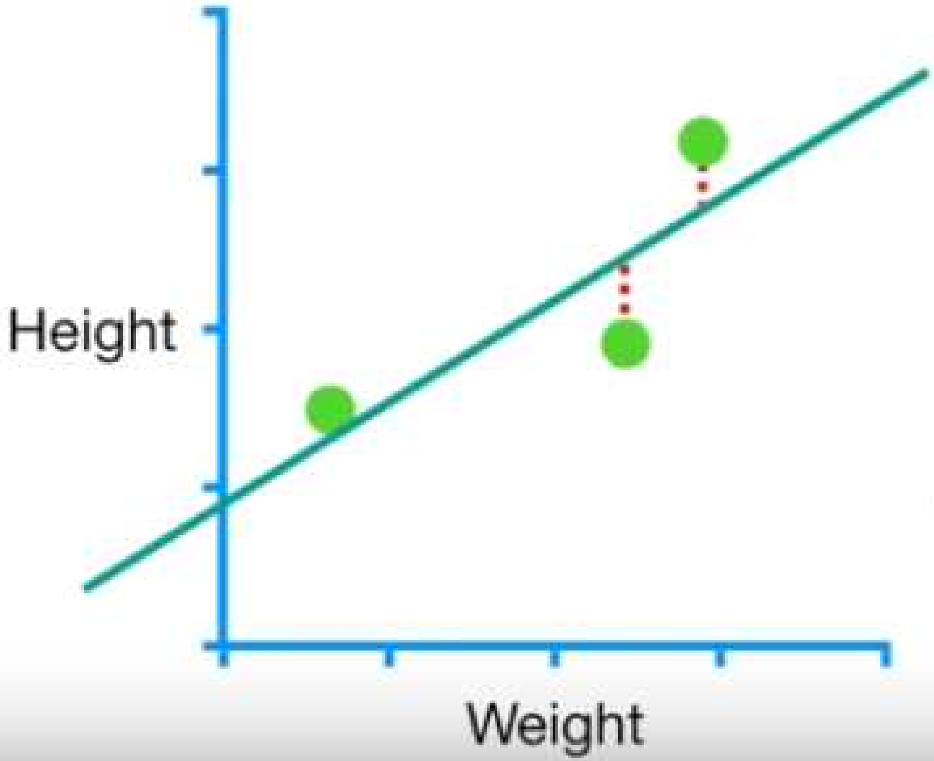


$$\text{Sum of squared residuals} = (\mathbf{1.4} - (\text{intercept} + \text{slope} \times \mathbf{0.5}))^2$$
$$+ (\mathbf{1.9} - (\text{intercept} + \text{slope} \times \mathbf{2.3}))^2$$
$$+ (\mathbf{3.2} - (\text{intercept} + \text{slope} \times \mathbf{2.9}))^2$$

We want to find the values for the **Intercept** and **Slope** that give us the minimum Sum of the Squared Residuals.



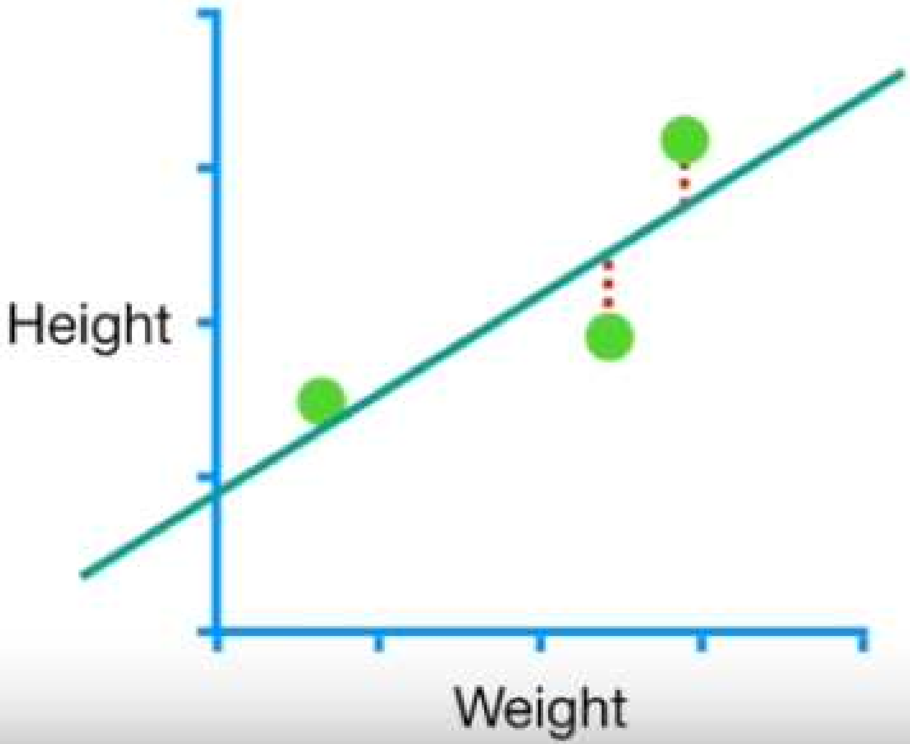
$$\text{Sum of squared residuals} = (\mathbf{1.4} - (\text{intercept} + \text{slope} \times \mathbf{0.5}))^2$$
$$+ (\mathbf{1.9} - (\text{intercept} + \text{slope} \times \mathbf{2.3}))^2$$
$$+ (\mathbf{3.2} - (\text{intercept} + \text{slope} \times \mathbf{2.9}))^2$$



...and just like before, we'll take the derivative with respect to the **Intercept**...

$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals}$$

$$\text{Sum of squared residuals} = (\mathbf{1.4} - (\text{intercept} + \text{slope} \times \mathbf{0.5}))^2$$
$$+ (\mathbf{1.9} - (\text{intercept} + \text{slope} \times \mathbf{2.3}))^2$$
$$+ (\mathbf{3.2} - (\text{intercept} + \text{slope} \times \mathbf{2.9}))^2$$



...but unlike before, we'll also take the derivative with respect to the **Slope**!

$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals}$$

$$\boxed{\frac{d}{d\ slope}\ \text{Sum of squared residuals}}$$

$$\frac{d}{d\ intercept} \text{ Sum of squared residuals} =$$

$$-2(\mathbf{1.4} - (\text{intercept} + \text{slope} \times \mathbf{0.5})$$

$$+ -2(\mathbf{1.9} - (\text{intercept} + \text{slope} \times \mathbf{2.3}))$$

$$+ -2(\mathbf{3.2} - (\text{intercept} + \text{slope} \times \mathbf{2.9}))$$

Here's the derivative of the Sum of the Squared Residuals with respect to the **Intercept**…

…and here's the derivative with respect to the **Slope**.

$$\frac{d}{d\ slope} \text{ Sum of squared residuals} =$$

$$-2 \times \mathbf{0.5}(\mathbf{1.4} - (\text{intercept} + \text{slope} \times \mathbf{0.5}))$$

$$+ -2 \times \mathbf{2.9}(\mathbf{3.2} - (\text{intercept} + \text{slope} \times \mathbf{2.9}))^2$$

$$+ -2 \times \mathbf{2.3}(\mathbf{1.9} - (\text{intercept} + \text{slope} \times \mathbf{2.3}))^2$$

$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals} =$$

$$-2(\mathbf{1.4} - (\text{intercept} + \text{slope} \times \mathbf{0.5}))$$

$$+ -2(\mathbf{1.9} - (\text{intercept} + \text{slope} \times \mathbf{2.3}))$$

$$+ -2(\mathbf{3.2} - (\text{intercept} + \text{slope} \times \mathbf{2.9}))$$

$$\frac{d}{d\ slope}\ \text{Sum of squared residuals} =$$

$$-2 \times \mathbf{0.5}(\mathbf{1.4} - (\text{intercept} + \text{slope} \times \mathbf{0.5}))$$

$$+ -2 \times \mathbf{2.9}(\mathbf{3.2} - (\text{intercept} + \text{slope} \times \mathbf{2.9}))^2$$

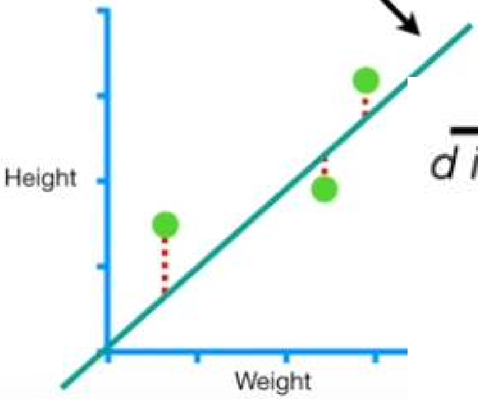
$$+ -2 \times \mathbf{2.3}(\mathbf{1.9} - (\text{intercept} + \text{slope} \times \mathbf{2.3}))^2$$

일단 시작은 2개의 random한 값으로 부터 시작을 하고, 점차 기울기를 활용해서 줄여나가자 임!!!!!!!!

Just like before, we will start by picking a random number for the **Intercept**. In this case we'll set the **Intercept** = **0**…

…and we'll pick a random number for the **Slope**. In this case we'll set the **Slope** = **1**.

Thus, this line, with **Intercept** = **0** and **Slope** = **1**, is where we will start.



$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals} = -2(\mathbf{1.4} - (0 + 1 \times \mathbf{0.5})) + -2(\mathbf{1.9} - (0 + 1 \times \mathbf{2.3})) + -2(\mathbf{3.2} - (0 + 1 \times \mathbf{2.9})) \boxed{= \mathbf{-1.6}}$$

$$\textbf{Step Size}_{\text{Intercept}} = \textbf{Slope} \times \textbf{Learning Rate}$$

…now we plug the **Slopes** into the **Step Size** formulas…

$$\frac{d}{d\ slope}\ \text{Sum of squared residuals} = -2 \times \mathbf{0.5}(\mathbf{1.4} - (0 + 1 \times \mathbf{0.5})) + -2 \times \mathbf{2.9}(\mathbf{3.2} - (0 + 1 \times \mathbf{2.9}))^2 + -2 \times \mathbf{2.3}(\mathbf{1.9} - (0 + 1 \times \mathbf{2.3}))^2 \boxed{= \mathbf{-0.8}}$$

$$\textbf{Step Size}_{\text{Slope}} = \textbf{Slope} \times \textbf{Learning Rate}$$

$$\frac{d}{d\ intercept}\text{ Sum of squared residuals} = -2(\mathbf{1.4} - (0 + 1 \times \mathbf{0.5})) + -2(\mathbf{1.9} - (0 + 1 \times \mathbf{2.3})) + -2(\mathbf{3.2} - (0 + 1 \times \mathbf{2.9})) = \mathbf{-1.6}$$

$$\textbf{Step Size}_{\text{Intercept}} = -1.6 \times \textbf{Learning Rate}$$

...and multiply by the **Learning Rate**, which this time we set to **0.01**...

$$\frac{d}{d\ slope}\text{ Sum of squared residuals} = -2 \times \mathbf{0.5}(\mathbf{1.4} - (0 + 1 \times \mathbf{0.5})) + -2 \times \mathbf{2.9}(\mathbf{3.2} - (0 + 1 \times \mathbf{2.9}))^2 + -2 \times \mathbf{2.3}(\mathbf{1.9} - (0 + 1 \times \mathbf{2.3}))^2 = \mathbf{-0.8}$$

$$\textbf{Step Size}_{\text{Slope}} = -0.8 \times \textbf{Learning Rate}$$

$$\frac{d}{d\ intercept}\text{ Sum of squared residuals} =$$

$$-2(\mathbf{1.4} - (0 + 1 \times \mathbf{0.5}))$$

$$+ -2(\mathbf{1.9} - (0 + 1 \times \mathbf{2.3}))$$

$$+ -2(\mathbf{3.2} - (0 + 1 \times \mathbf{2.9})) = \mathbf{-1.6}$$

Step Size$_{\text{Intercept}}$ $= -1.6 \times 0.01 =$ **-0.016**

Anyway, we do the math and get two **Step Sizes**.

$$\frac{d}{d\ slope}\text{ Sum of squared residuals} =$$

$$-2 \times \mathbf{0.5}(\mathbf{1.4} - (0 + 1 \times \mathbf{0.5}))$$

$$+ -2 \times \mathbf{2.9}(\mathbf{3.2} - (0 + 1 \times \mathbf{2.9}))^2$$

$$+ -2 \times \mathbf{2.3}(\mathbf{1.9} - (0 + 1 \times \mathbf{2.3}))^2 = \mathbf{-0.8}$$

Step Size$_{\text{Slope}}$ $= -0.8 \times 0.01 =$ **-0.008**

$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals} =$$

$$-2(\mathbf{1.4} - (0 + 1 \times \mathbf{0.5}))$$

$$+ -2(\mathbf{1.9} - (0 + 1 \times \mathbf{2.3}))$$

$$+ -2(\mathbf{3.2} - (0 + 1 \times \mathbf{2.9})) = \mathbf{-1.6}$$

$$\textbf{Step Size}_{\text{Intercept}} = -1.6 \times 0.01 = \mathbf{-0.016}$$

New Intercept = 0 - **Step Size**

Now we calculate the **New Intercept** and **New Slope** by plugging in the **Old Intercept** and the **Old Slope**…

$$\frac{d}{d\ slope}\ \text{Sum of squared residuals} =$$

$$-2 \times \mathbf{0.5}(\mathbf{1.4} - (0 + 1 \times \mathbf{0.5}))$$

$$+ -2 \times \mathbf{2.9}(\mathbf{3.2} - (0 + 1 \times \mathbf{2.9}))^2$$

$$+ -2 \times \mathbf{2.3}(\mathbf{1.9} - (0 + 1 \times \mathbf{2.3}))^2 = \mathbf{-0.8}$$

$$\textbf{Step Size}_{\text{Slope}} = -0.8 \times 0.01 = \mathbf{-0.008}$$

New Slope = 1 - **Step Size**

$$\frac{d}{d\ intercept}\text{ Sum of squared residuals} = -2(\mathbf{1.4} - (0 + 1 \times \mathbf{0.5})) + -2(\mathbf{1.9} - (0 + 1 \times \mathbf{2.3})) + -2(\mathbf{3.2} - (0 + 1 \times \mathbf{2.9})) = \mathbf{-1.6}$$

$$\textbf{Step Size}_{\text{Intercept}} = -1.6 \times 0.01 = \mathbf{-0.016}$$

New Intercept = 0 - (-0.016) = 0.016

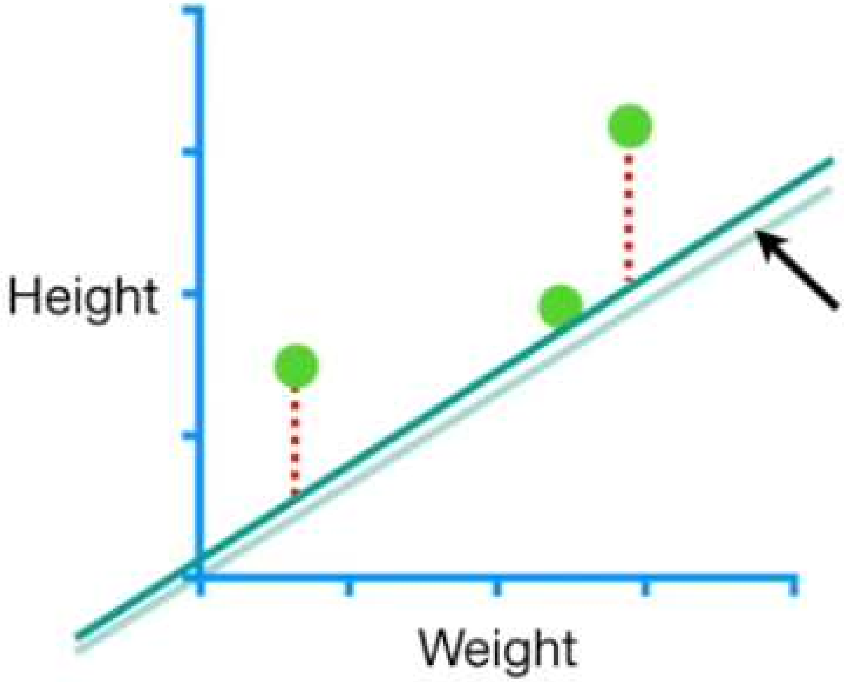
…and we end up with a **New Intercept** and a **New Slope**.

$$\frac{d}{d\ slope}\text{ Sum of squared residuals} = -2 \times \mathbf{0.5}(\mathbf{1.4} - (0 + 1 \times \mathbf{0.5})) + -2 \times \mathbf{2.9}(\mathbf{3.2} - (0 + 1 \times \mathbf{2.9}))^2 + -2 \times \mathbf{2.3}(\mathbf{1.9} - (0 + 1 \times \mathbf{2.3}))^2 = \mathbf{-0.8}$$

$$\textbf{Step Size}_{\text{Slope}} = -0.8 \times 0.01 = \mathbf{-0.008}$$

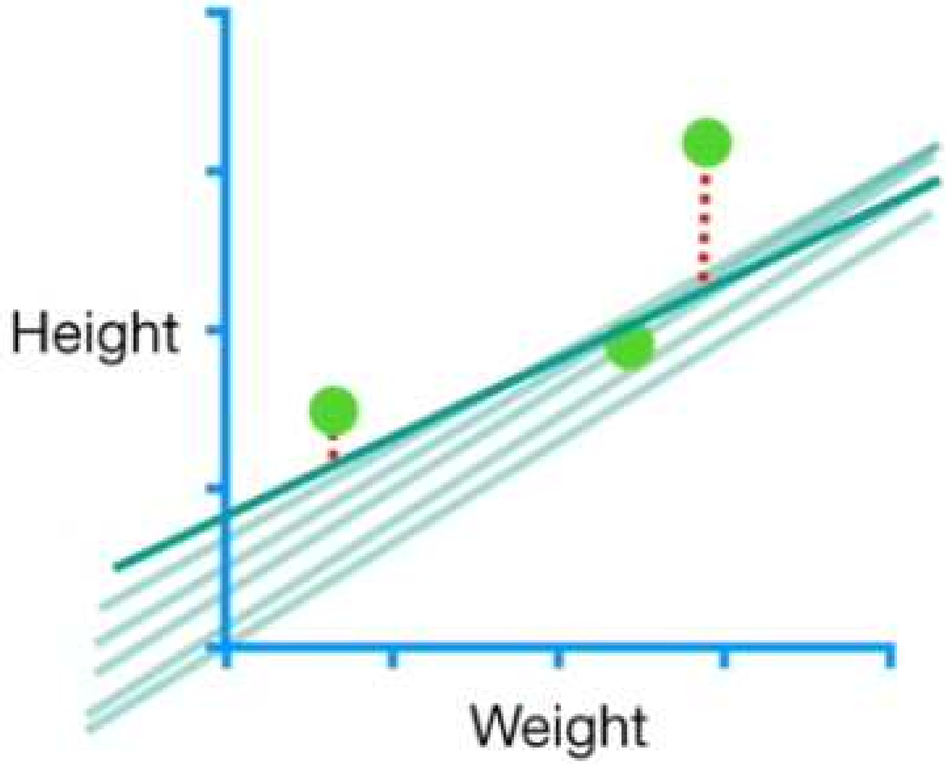
New Slope = 1 - (-0.008) = 1.008

New Intercept = 0 - (-0.016) = 0.016

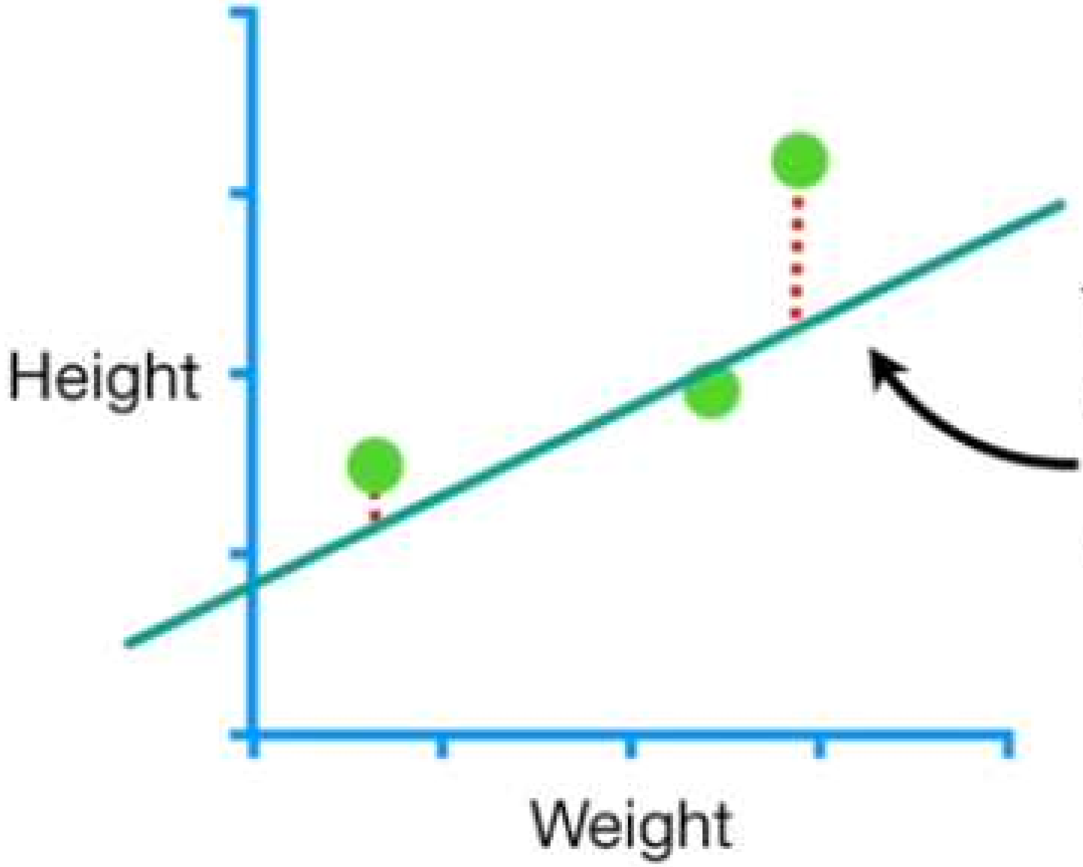


...and this is the new line (with **Slope = 1.008** and **Intercept = 0.016**) after the first step.

New Slope = 1 - (-0.008) = 1.008



Now we just repeat what we did until all of the **Steps Sizes** are very small or we reach the **Maximum Number of Steps**.



This is the best fitting line, with **Intercept** = **0.95** and **Slope** = **0.64**, the same values we get from **Least Squares**.

Summary

Step 1: Take the derivative of the **Loss Function** for each parameter in it. In fancy Machine Learning Lingo, take the **Gradient** of the **Loss Function**.

Step 2: Pick random values for the parameters.

Step 3: Plug the parameter values into the derivatives (ahem, the **Gradient**).

Step 4: Calculate the Step Sizes: **Step Size** = **Slope** × **Learning Rate**

Step 5: Calculate the New Parameters:

New Parameter = Old Parameter - Step Size

Now go back to **Step 3** and repeat until
Step Size is very small, or you reach
the **Maximum Number of Steps**.

Step 3: Plug the parameter values into the derivatives (ahem, the **Gradient**).

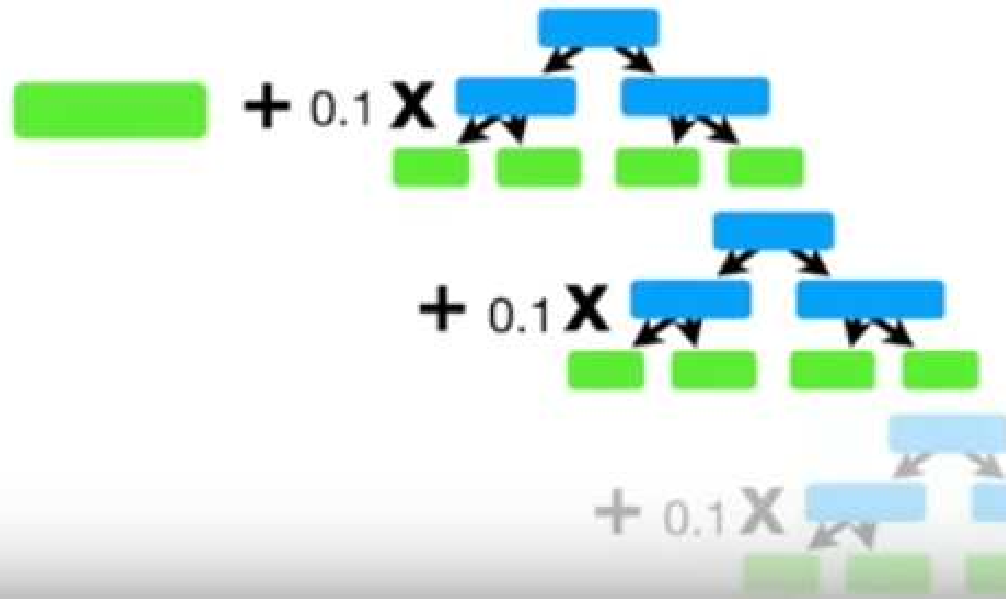
Step 4: Calculate the Step Sizes: **Step Size** = **Slope** × **Learning Rate**

Step 5: Calculate the New Parameters:

New Parameter = **Old Parameter** - **Step Size**

GBM

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57



NOTE: When **Gradient Boost** is used to **Predict** a continuous value, like **Weight**, we say that we are using **Gradient Boost** for **Regression**.

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

Using **Gradient Boost** for **Regression** is different from doing **Linear Regression**, so while the two methods are related, don't get them confused with each other.

Step01) 우선 1개의 leaf 에서 시작을 한다.
Regression : 평균으로 시작을

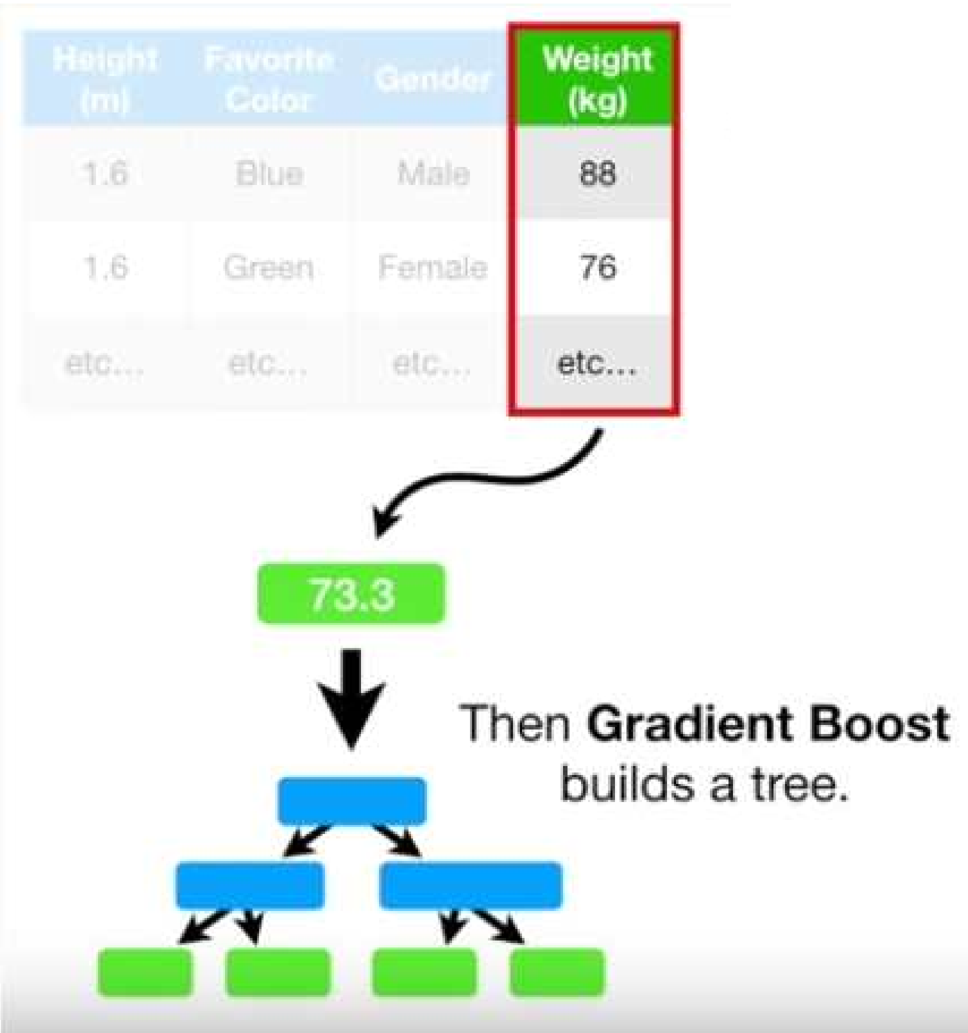
Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
etc...	etc...	etc...	etc...

This leaf represents an initial guess for the **Weights** of all of the samples.

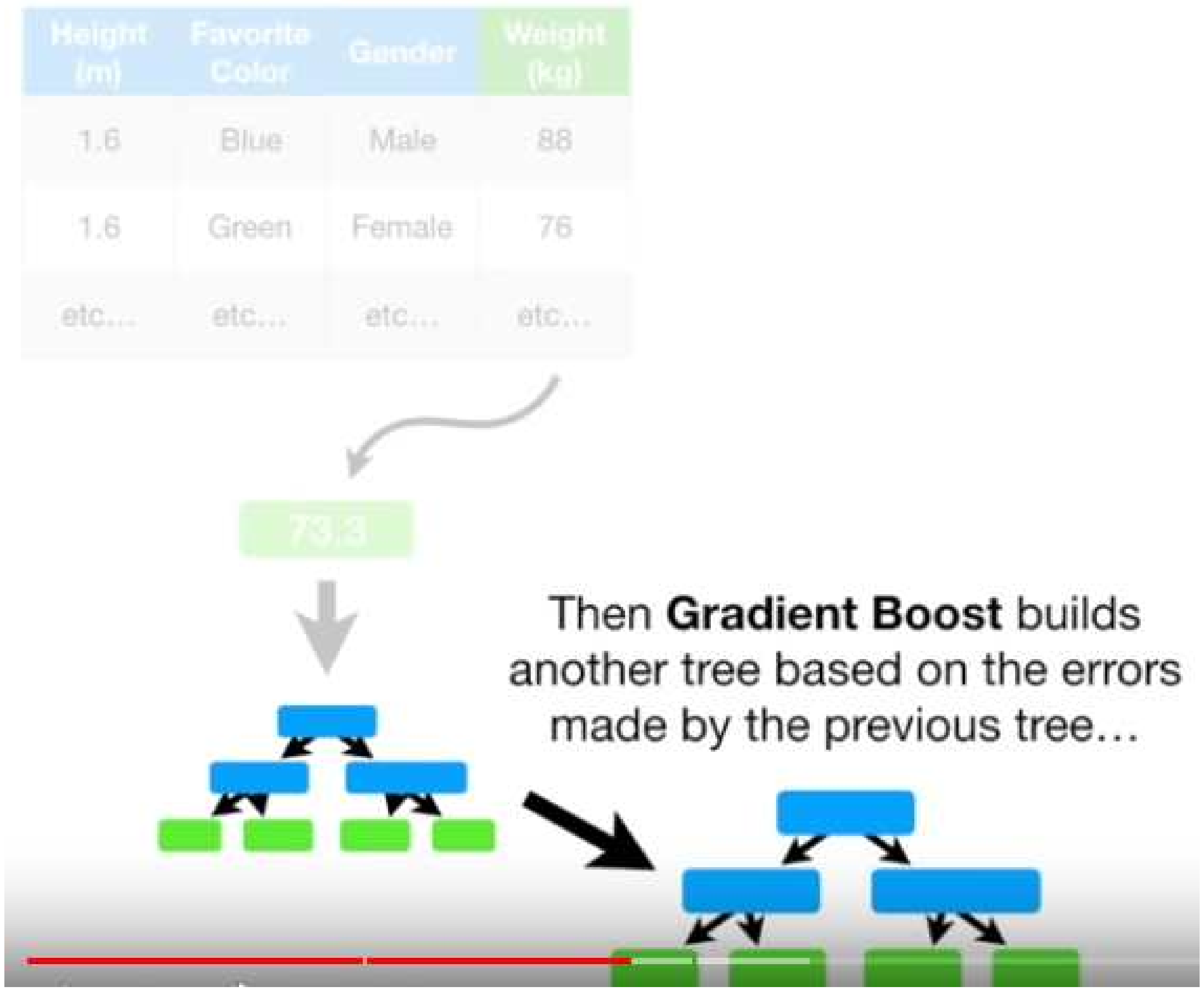
주어진 Label의 평균값 계산.

When trying to **Predict** a continuous value like **Weight**, the first guess is the the average value.

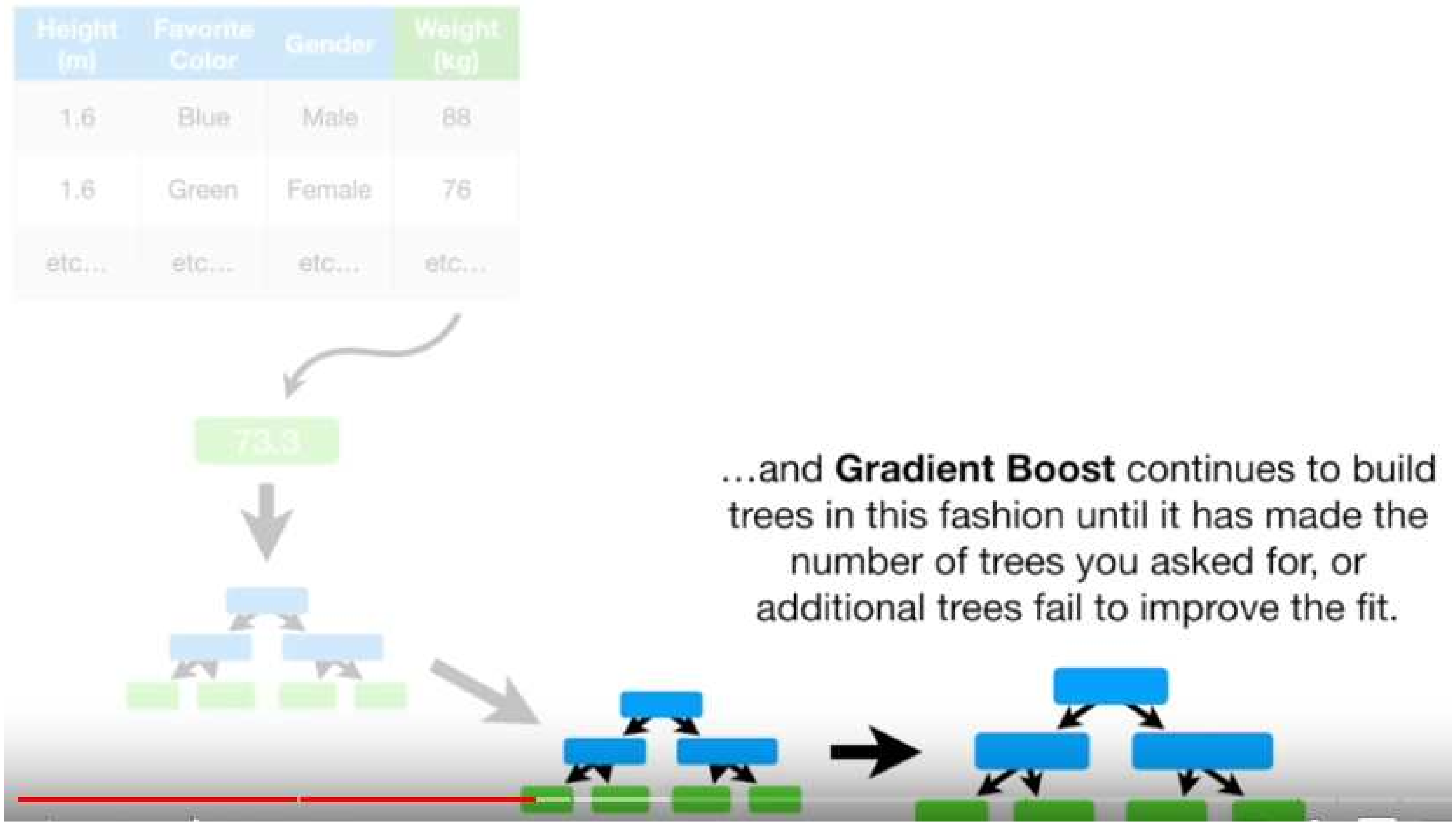
Step02) 이제 그 평균으로 부터 tree을 생성을 한다.



Step03) 전 tree의 error 기반의 예측 tree 만들기



Step04) 이것을 지속적으로 이어서 수행을 함.



Let’s Start

...let's see how the most common **Gradient Boost** configuration would use this **Training Data** to **Predict Weight**.

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

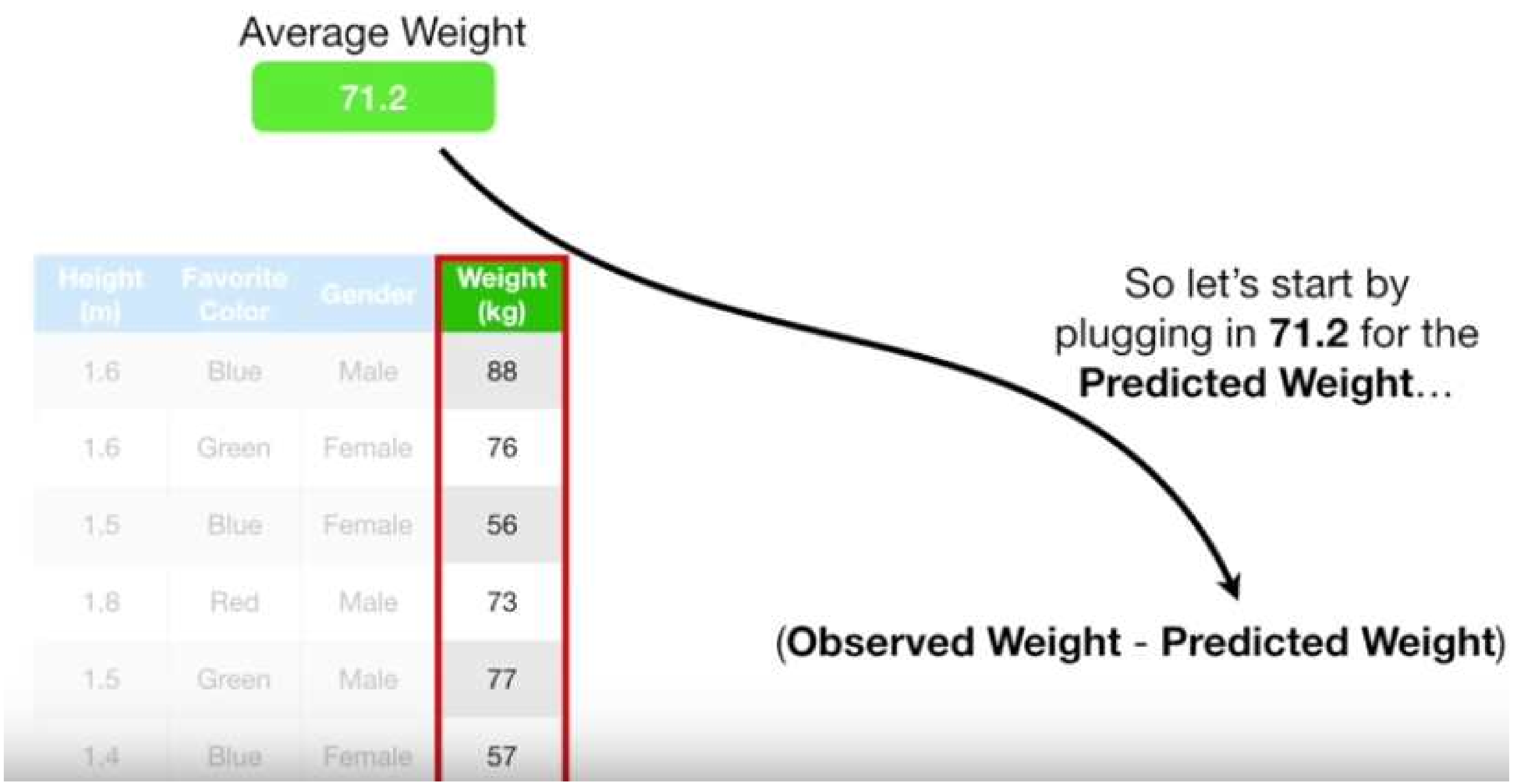
평균 구하기!

Average Weight

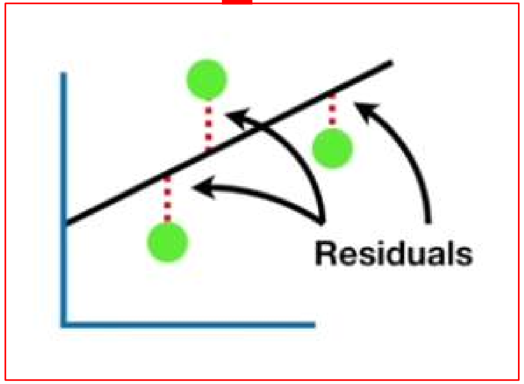
71.2

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

오차들 구하기


Average Weight
71.2
Height (m)
Favorite Color
Gender
Weight (kg)
1.6 Blue Male 88
1.6 Green Female 76
1.5 Blue Female 56
1.8 Red Male 73
1.5 Green Male 77
1.4 Blue Female 57
So let's start by plugging in 71.2 for the Predicted Weight...
(Observed Weight - Predicted Weight)

참고



Average Weight

71.2

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	
1.5	Blue	Female	56	
1.8	Red	Male	73	
1.5	Green	Male	77	
1.4	Blue	Female	57	

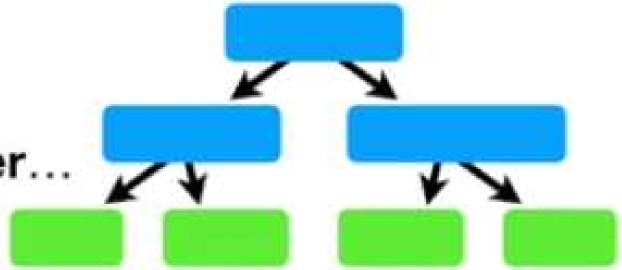
…and save the difference, which is called a **Pseudo Residual**, in a new column.

$$(\mathbf{88} - \mathbf{71.2}) = \boxed{16.8}$$

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

오차를 예측하는 tree 만들기!!!!!

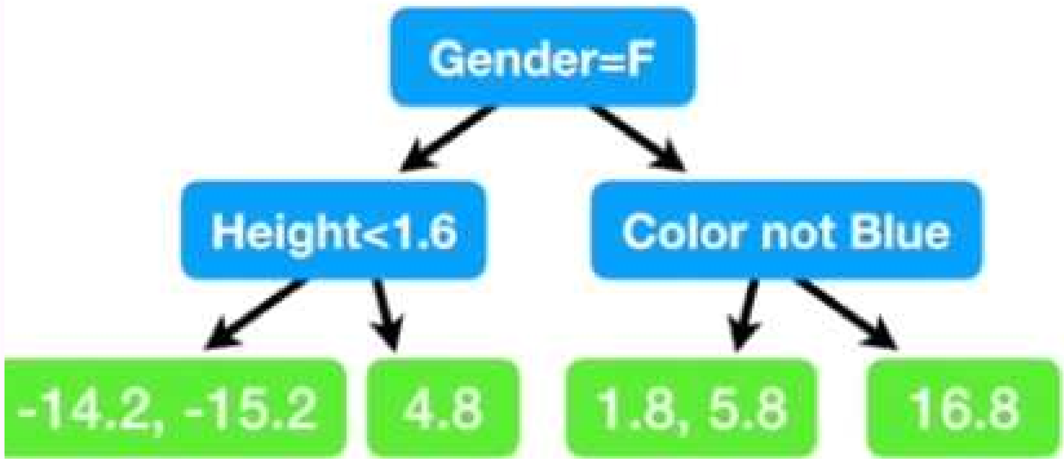
Now we will build a **Tree**, using
Height, **Favorite Color** and **Gender**…



Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

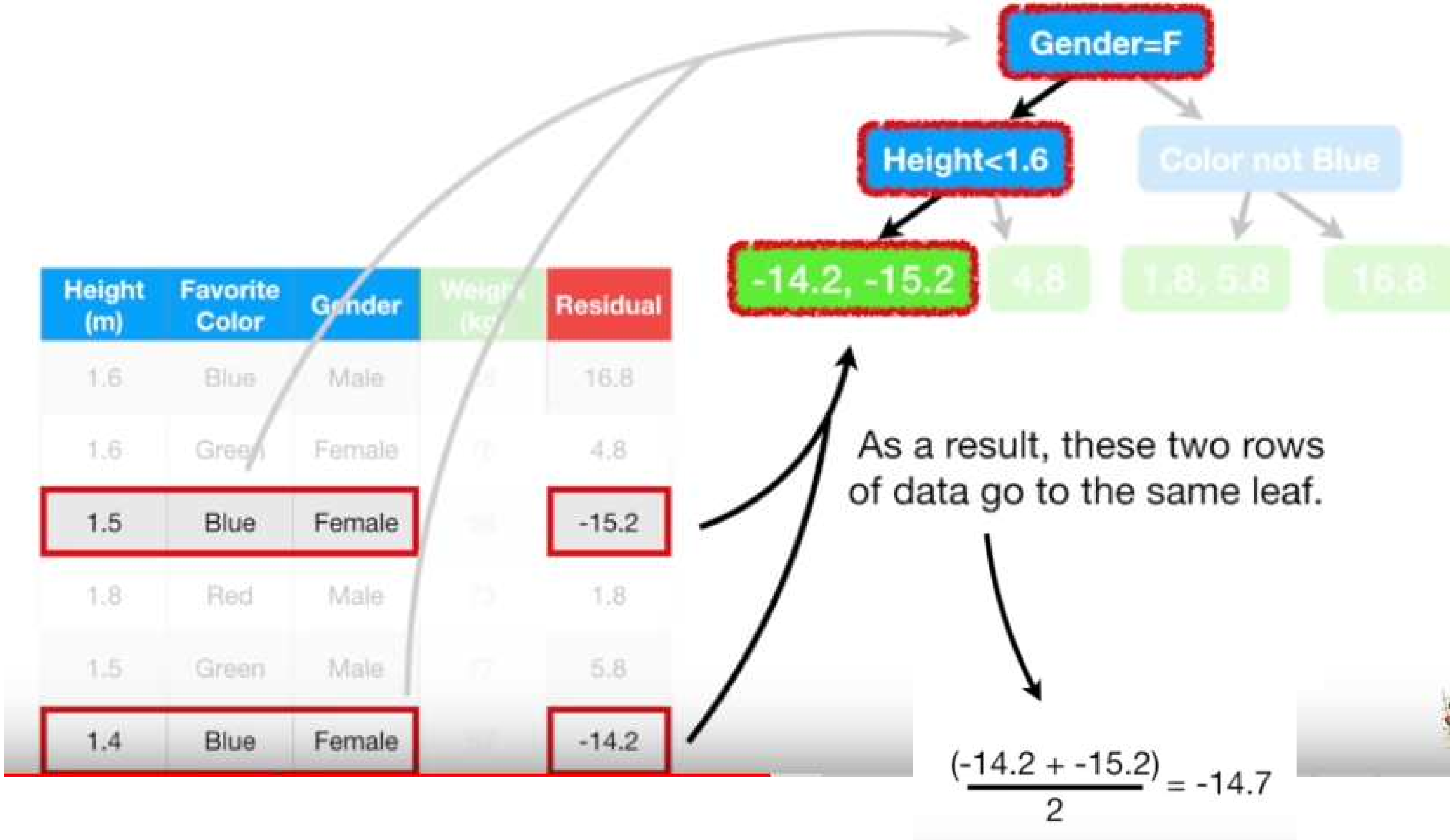
…to **Predict** the **Residuals**.

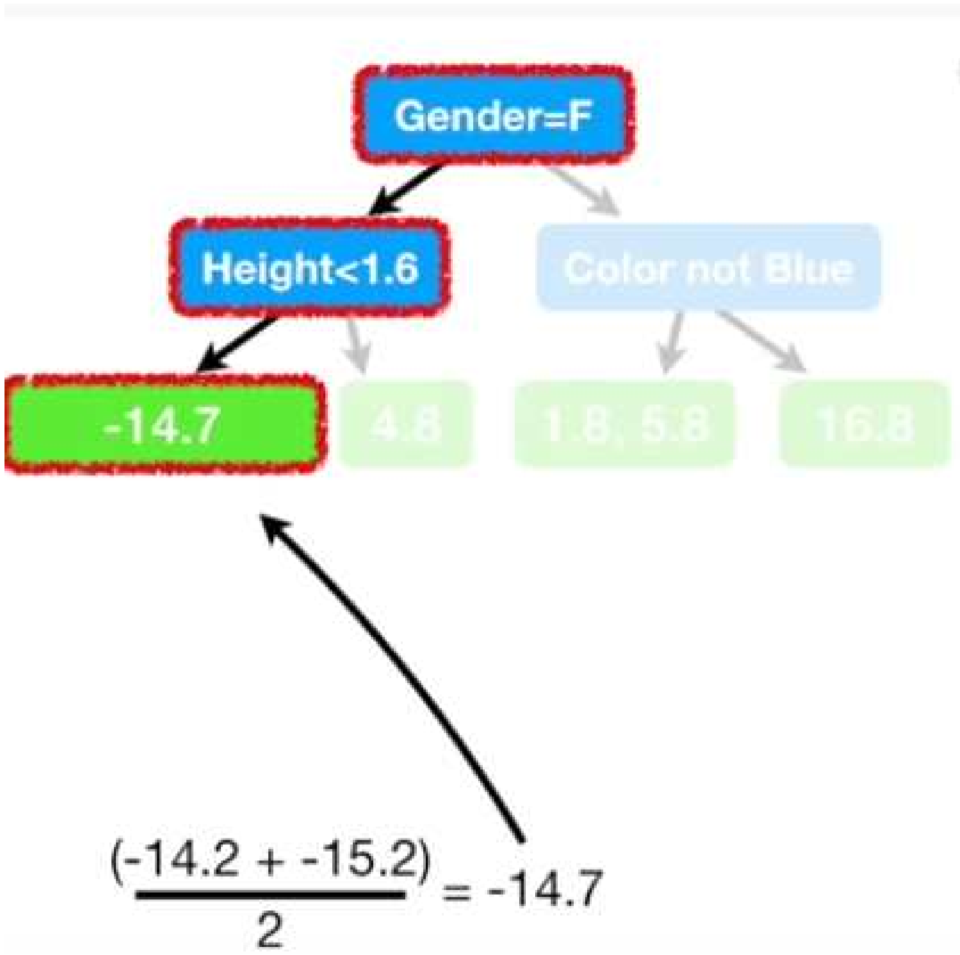
Height (m)	Favorite Color	Gender	[illegible]	Residual
1.6	Blue	Male	[illegible]	16.8
1.6	Green	Female	[illegible]	4.8
1.5	Blue	Female	[illegible]	-15.2
1.8	Red	Male	[illegible]	1.8
1.5	Green	Male	[illegible]	5.8
1.4	Blue	Female	[illegible]	-14.2




Gender=F
Height<1.6
Color not Blue
-14.2, -15.2
4.8
1.8, 5.8
16.8
Height (m)
Favorite Color
Gender
Residual
1.6 Blue Male 16.8
1.6 Green Female 4.8
1.5 Blue Female -15.2
1.8 Red Male 1.8
1.5 Green Male 5.8
1.4 Blue Female -14.2
By restricting the total number of leaves, we get fewer leaves than **Residuals**.

1개의 leaf에 여러 개가 존재하면 “평균 " 으로 구하기!



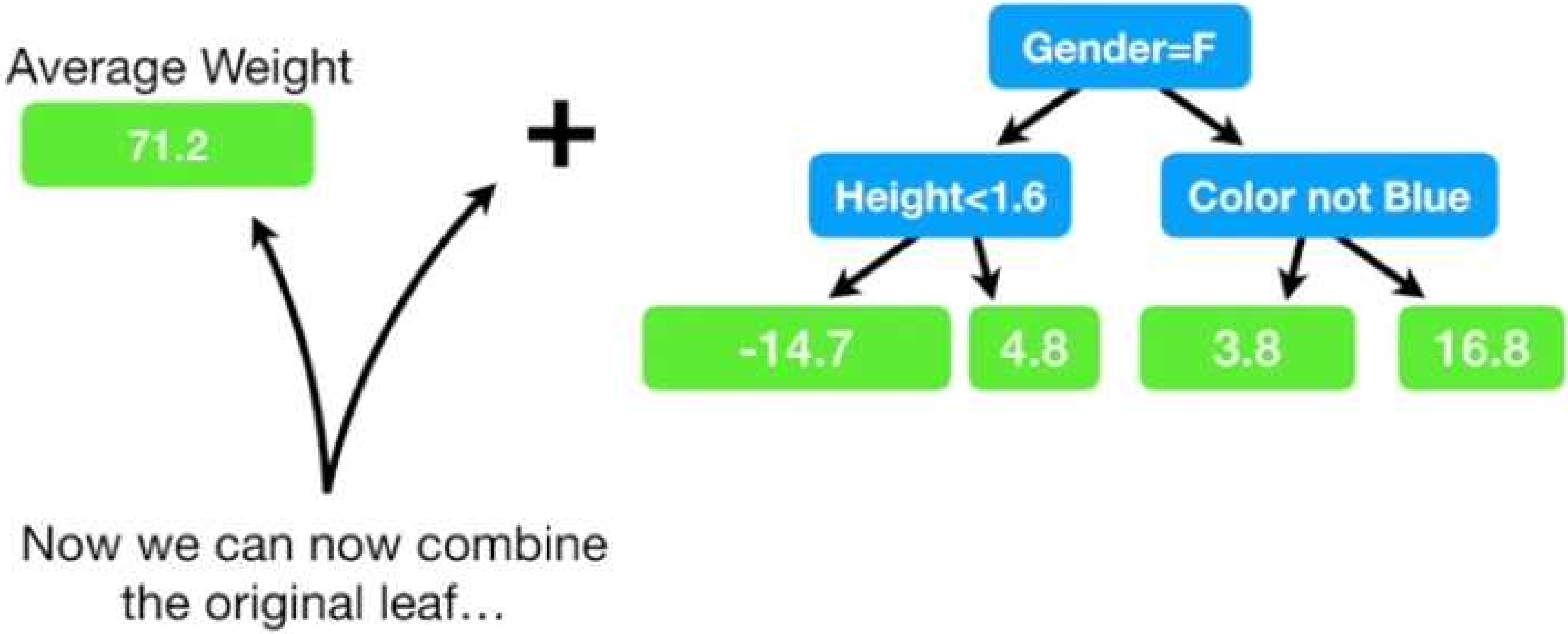

Gender=F
Height<1.6
Color not Blue
-14.7
4.8
1.8, 5.8
16.8
(-14.2 + -15.2) / 2 = -14.7

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	[illegible]	16.8
1.6	Green	Female	[illegible]	4.8
1.5	Blue	Female	[illegible]	-15.2
1.8	Red	Male	[illegible]	1.8
1.5	Green	Male	[illegible]	5.8
1.4	Blue	Female	[illegible]	-14.2

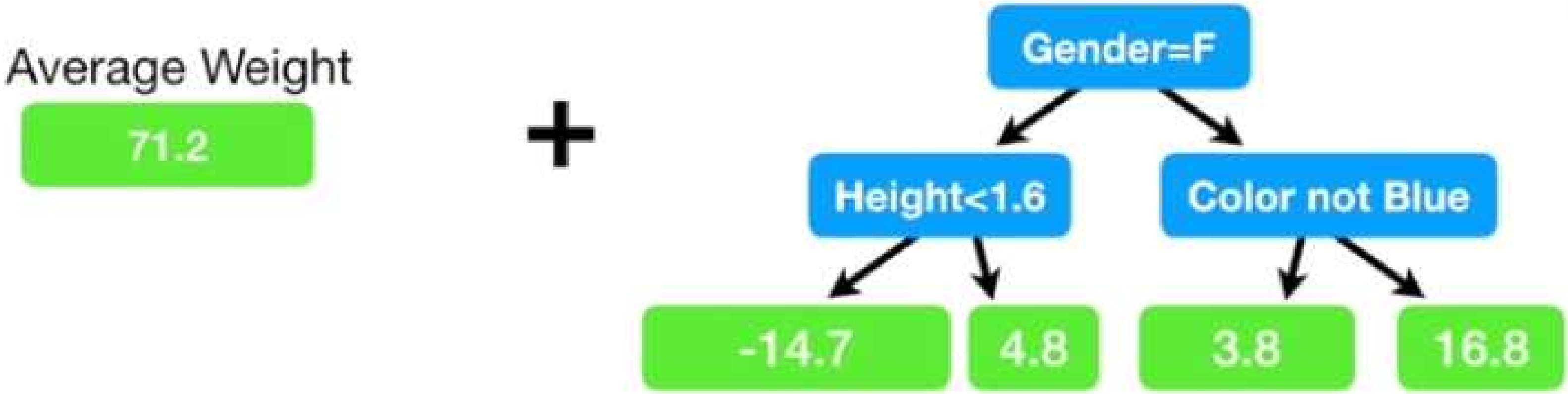
So we replace these residuals with their average.

$$\frac{(1.8 + 5.8)}{2}$$

이제 앞에서 한 것들을 합해보자!!!!

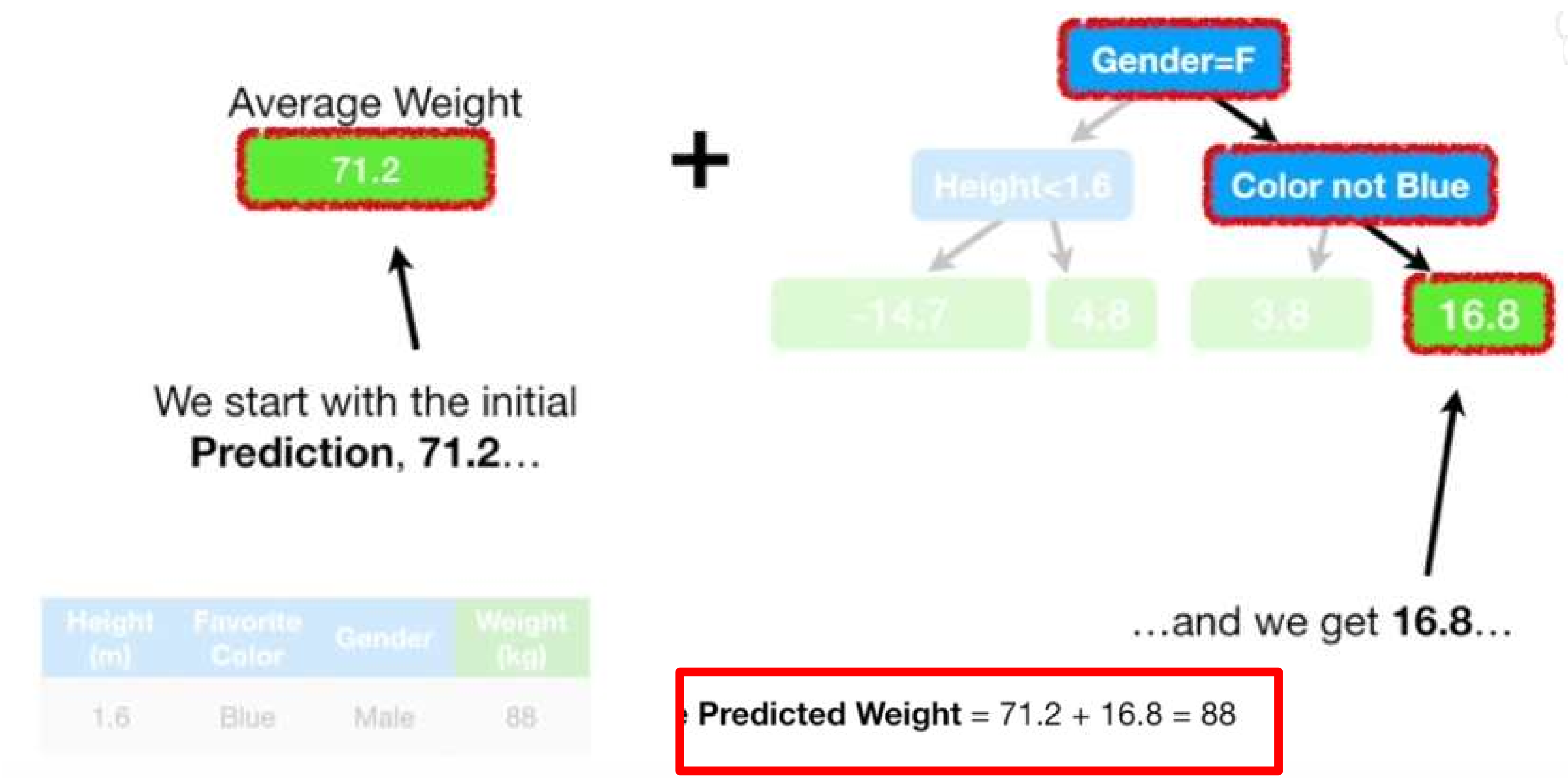


1번 sampl에 대해서 알아보자!!!
-> 아래의 2개의 부분을 1번 sample 입장으로 보자!!



Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

…to make a new **Prediction** of an individual's **Weight** from the **Training Data**.


Average Weight
71.2
We start with the initial
Prediction, 71.2…
Gender=F
Height<1.6
Color not Blue
-14.7
4.8
3.8
16.8
…and we get 16.8…
Height (m)
Favorite Color
Gender
Weight (kg)
1.6
Blue
Male
88
Predicted Weight = 71.2 + 16.8 = 88

Predicted Weight = 71.2 + 16.8 $\boxed{= 88}$

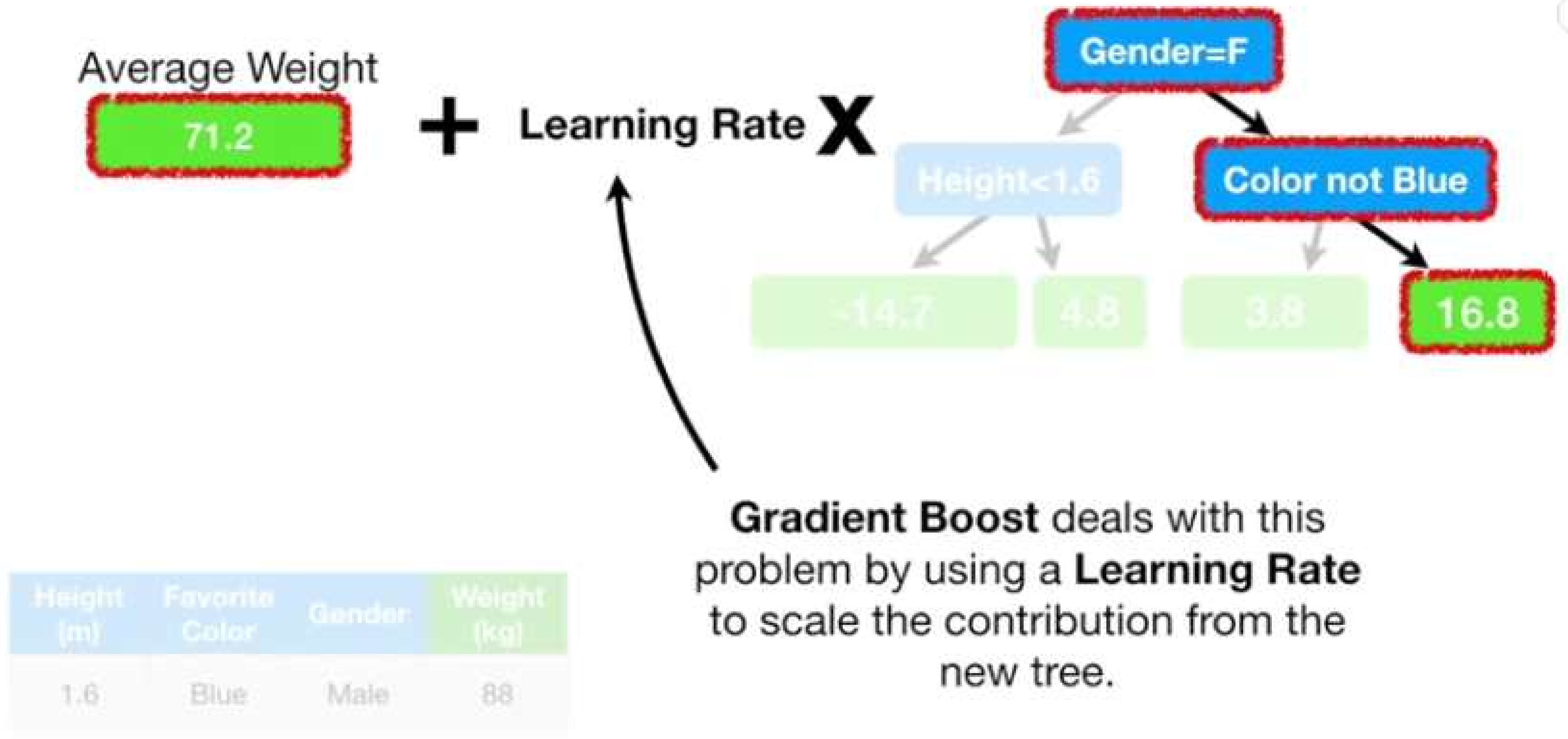
Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

Is this awesome???

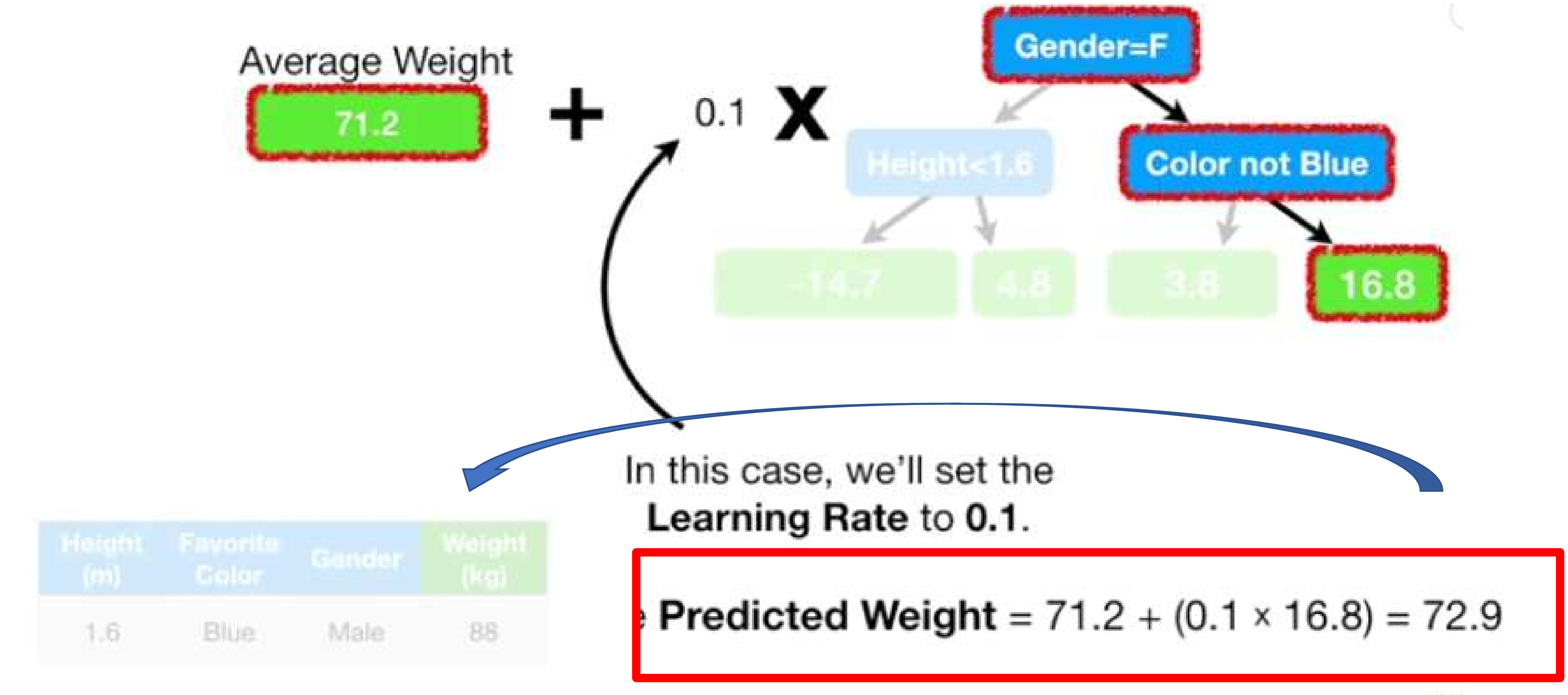
No. The model fits the **Training Data** too well.

In other words, we have low **Bias**, but probably very high **Variance**.

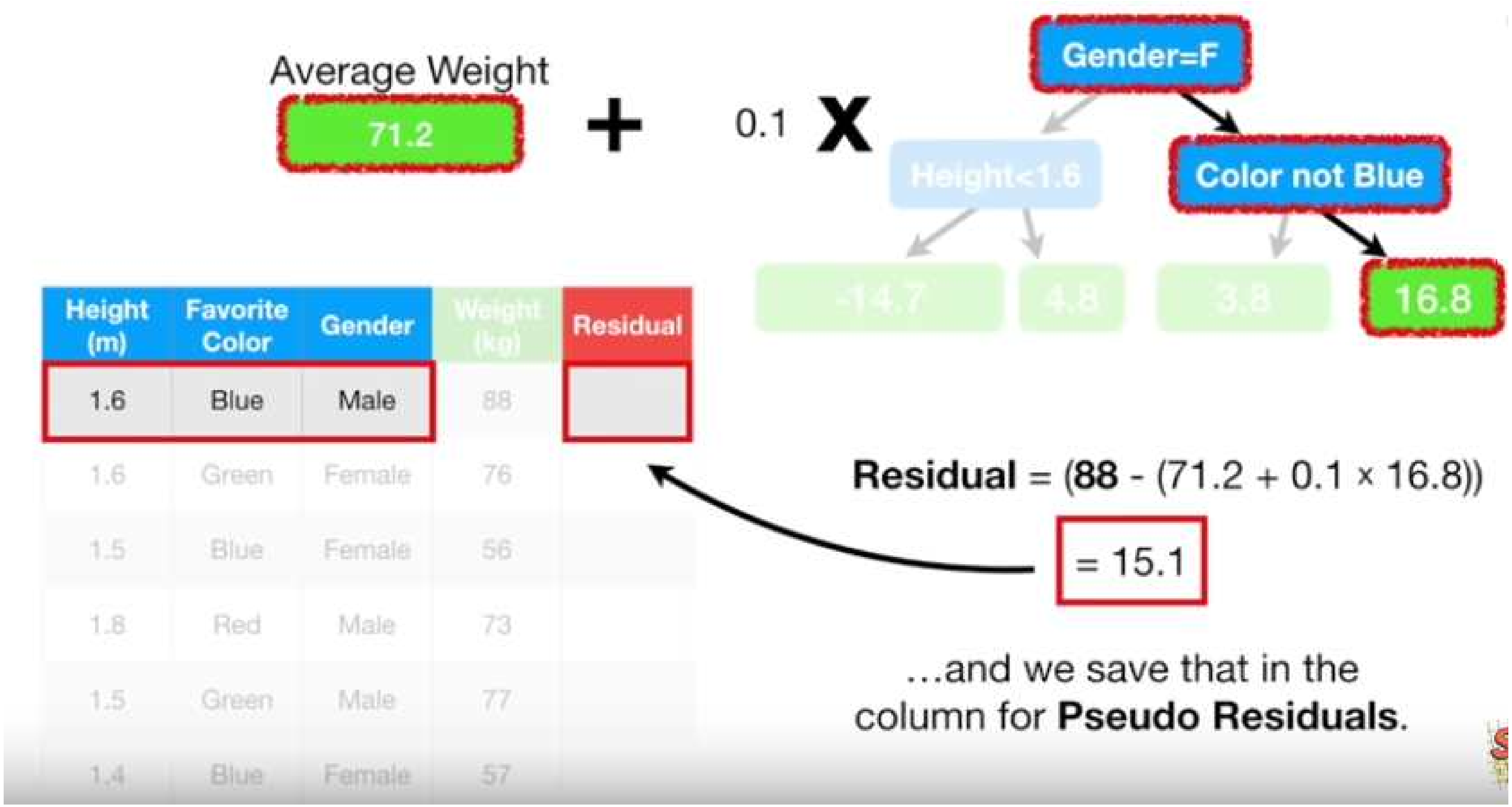
Learning rate를 활용해서 이제 점차 에러의 에러를
더 줄여나가보자!!!!!



Learning rate를 0.1로 예를 들어보자!

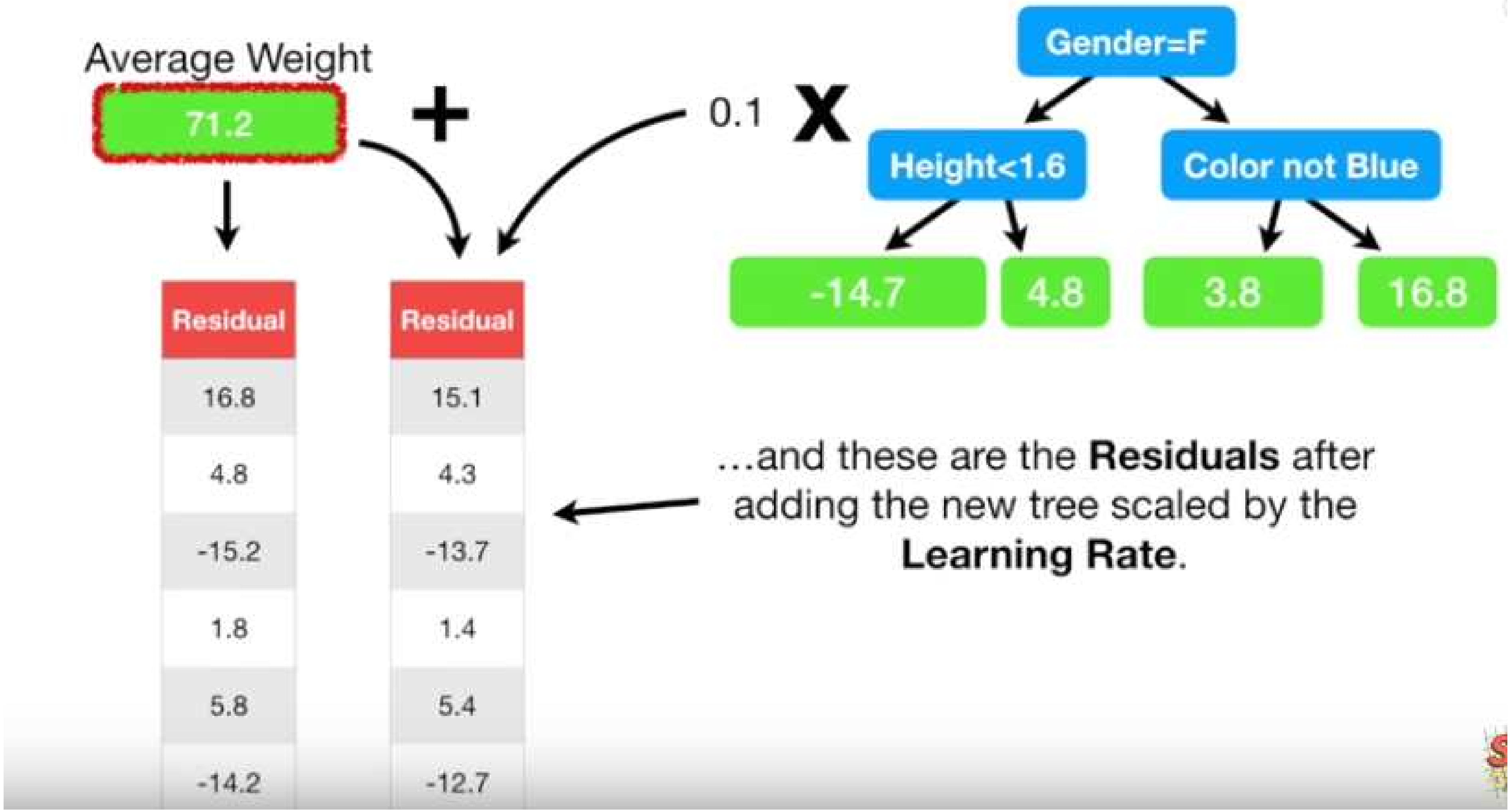


앞의 전체평균 & 1번 tree로 예측된 값으로 다시
에러의 값을 업데이트를 해보자!!!!!!



Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	15.1
1.6	Green	Female	76	4.3
1.5	Blue	Female	56	-13.7
1.8	Red	Male	73	1.4
1.5	Green	Male	77	5.4
1.4	Blue	Female	57	-12.7

에러들이 지금 수행을 하면서 어떻게 변화 되었는지 보자!!!!!!



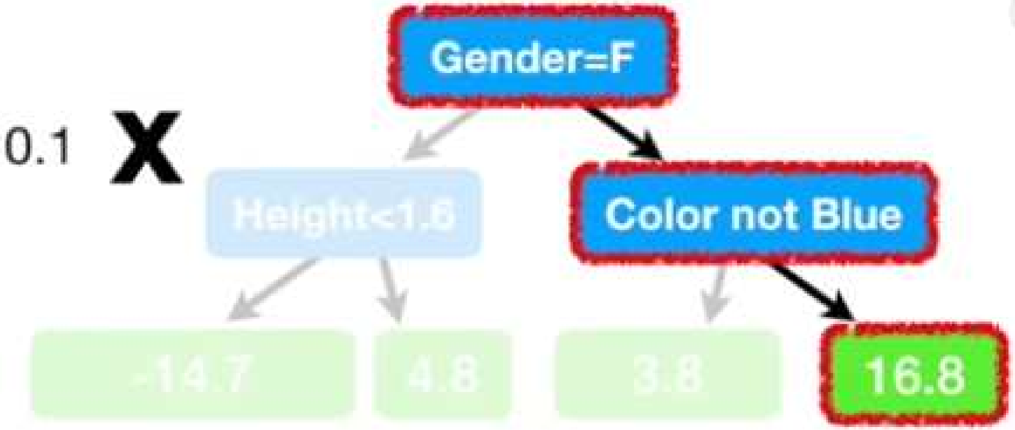
이 2번째 에러를 예측하는 tree를 다시 만들자!!
그리고 여러 값이 되는 것은 그 값들의 “평균”으로 다시 처리!!!!


Gender=F
Height<1.6
Color not Blue
-12.7,-13.7
4.3
1.4,5.4
15.1
평균 처리 수행!!!!
Gender=F
Height<1.6
Color not Blue
-13.2
4.3
3.4
15.1

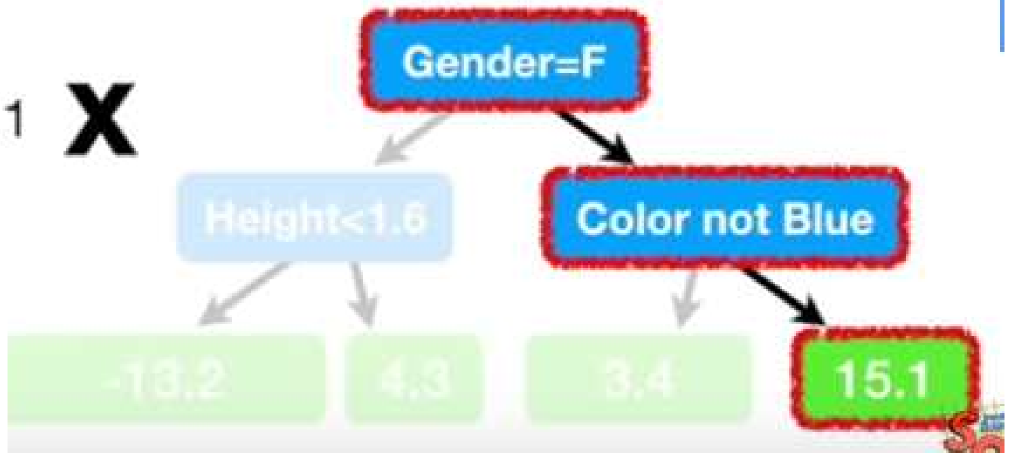
1번 sampl에 대해서 다시 알아보자!!!!

Average Weight

71.2 $+ \ 0.1 \times$



$+ \ 0.1 \times$

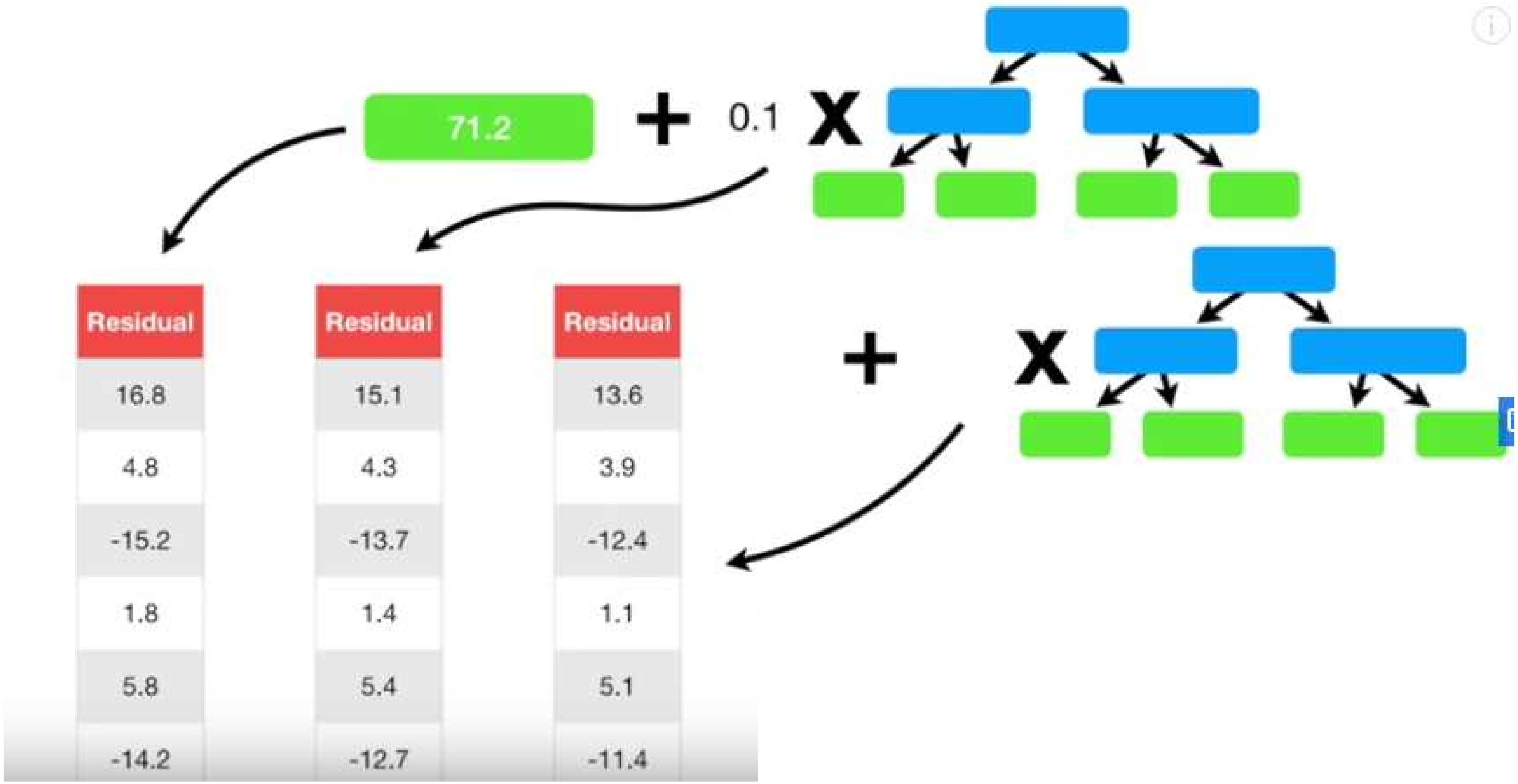


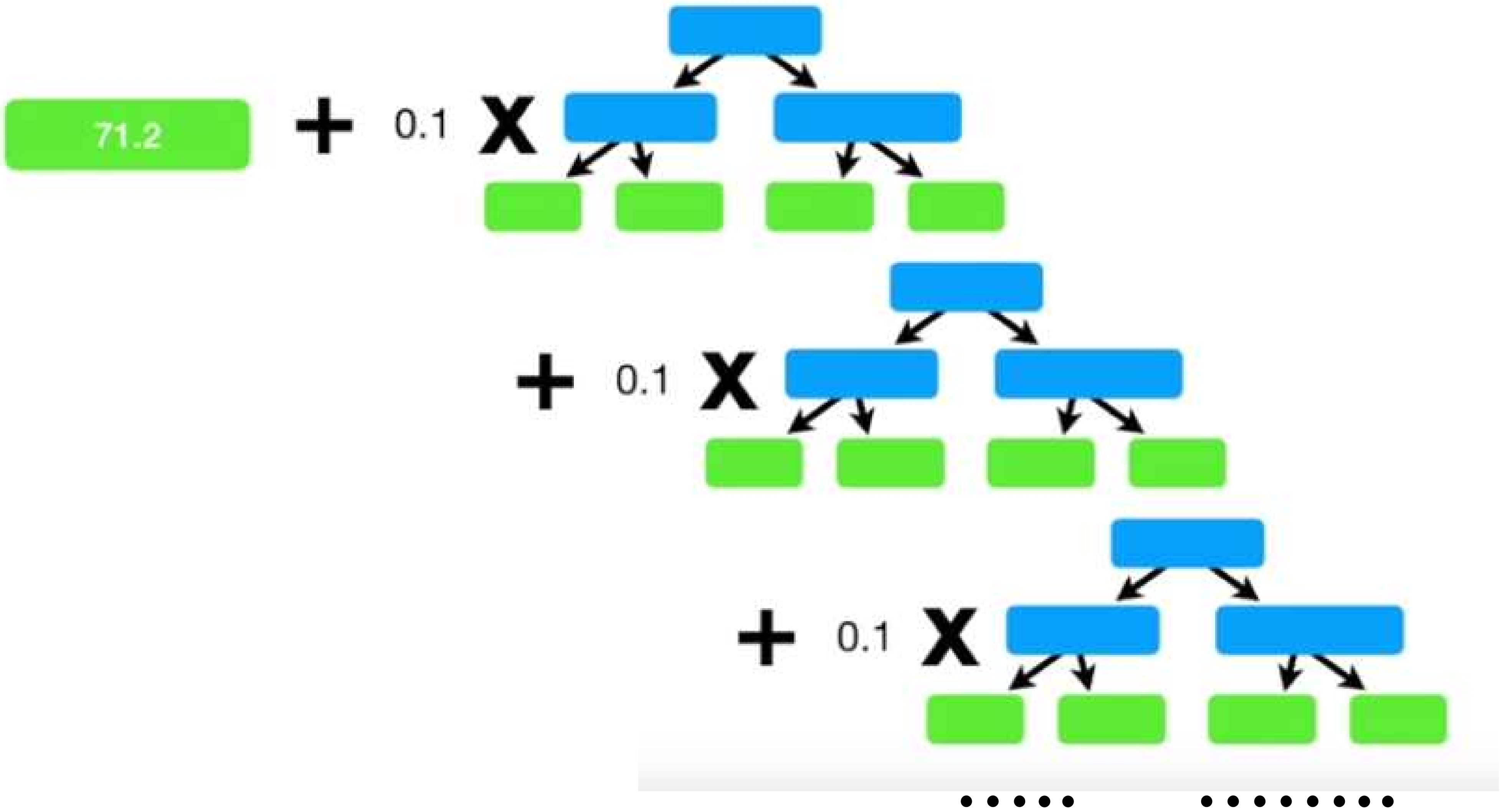
$$71.2 + (0.1 \times 16.8) + (0.1 \times 15.1)$$

$$\boxed{= 74.4}$$

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

에러가 여러 단계를 거치면서 줄어들고 있다!!!
이것을 계속 늘리면!!!!!



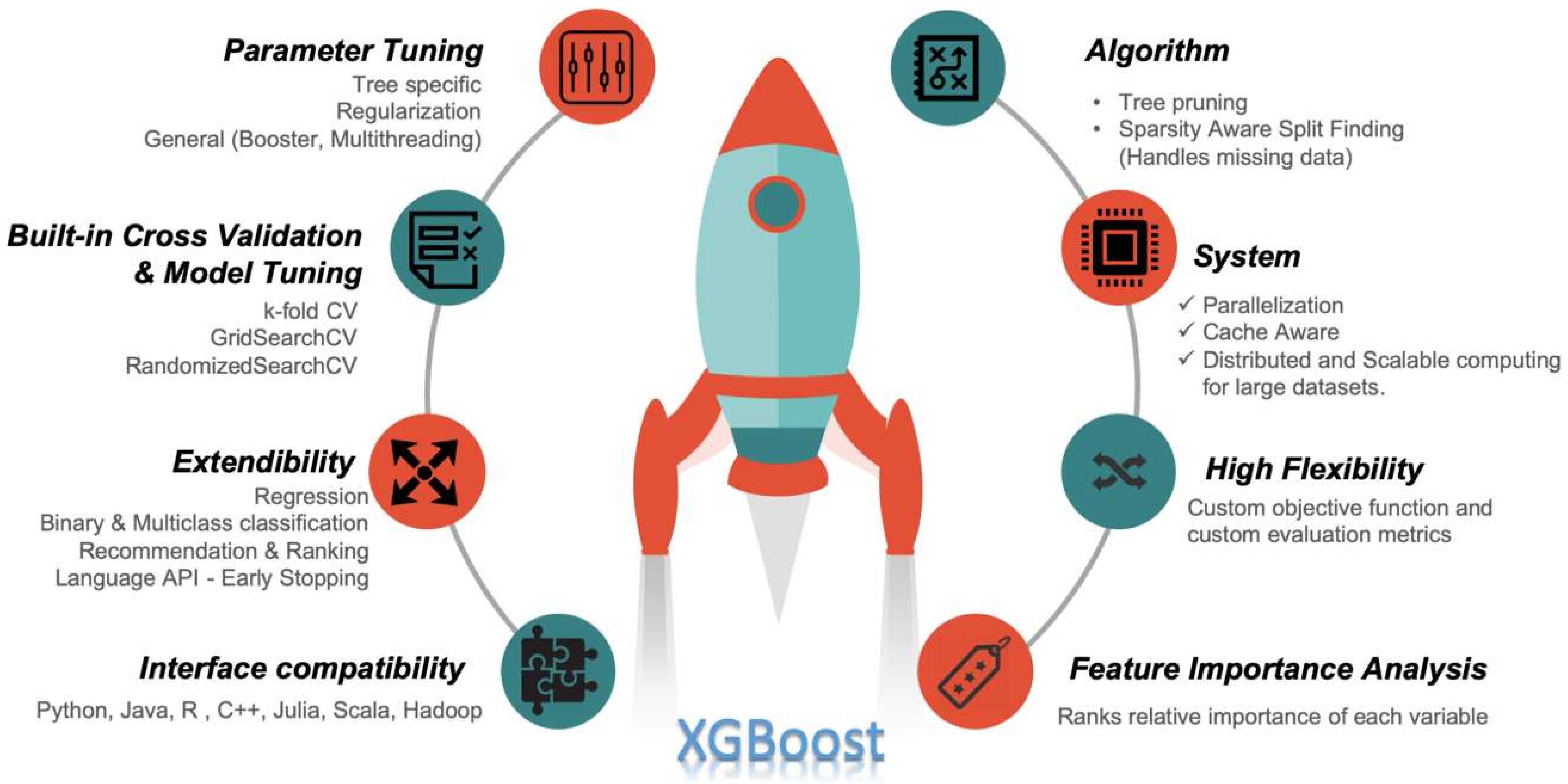


71.2
+ 0.1 X
+ 0.1 X
+ 0.1 X

Xgboost

xgboost

- Gbm의 기본적인 방법을 따르면서 빠르게 대용량의 데이터에 적용할지에 대한 고민을 한 방법임.
- 그래서 이 부분에 대해서는 알고리즘적인 개선 & HW적인 구현에 따라는 개선으로 나눠지게 된다. → 결론은 엄청나게 고민을 하고, 주어진 환경에서 쥐어짜내서 최대한을 하겠다는 모토가 보임.
- Ref) XGBoost: A Scalable Tree Boosting System , Tianqi Chen/Carlos Guestrin


Parameter Tuning
Tree specific
Regularization
General (Booster, Multithreading)
Built-in Cross Validation & Model Tuning
k-fold CV
GridSearchCV
RandomizedSearchCV
Extendibility
Regression
Binary & Multiclass classification
Recommendation & Ranking
Language API - Early Stopping
Interface compatibility
Python, Java, R , C++, Julia, Scala, Hadoop
Algorithm
Tree pruning
Sparsity Aware Split Finding (Handles missing data)
System
Parallelization
Cache Aware
Distributed and Scalable computing for large datasets.
High Flexibility
Custom objective function and custom evaluation metrics
Feature Importance Analysis
Ranks relative importance of each variable
XGBoost

Idea_1) 빠르게 하기 위해서 Approximation을 사용함.

- 앞에서 본 DT의 경우에서 보면, 모든 경우에 대해서 다 해보고 그 모든 경우 중에서 제일 좋은 기준들을 찾는 것들을 수행을 한다!!! → 모든 경우에서 제일 좋은 최적의 기준을 찾으려고 하는것이 DT가 가지는 기본적인 방향임.
- But
  - Data가 너무나 커서 메모리에 다 들어가지 않으면 계산이 안 되는데;;;
- Solutions
  - Approximation을 사용하고자 한다.
    - Data를 특징별로 k개로 나눠서두고(percentile을 사용) → HPT : eps
    - 이것을 Tree 단위(Global) or Split의 분할(Local) 에서 모두 사용을 한다.

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ *to* m **do**

  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature k.

  Proposal can be done per tree (global), or per split(local).

end

for $k = 1$ *to* m **do**

  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$

  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$

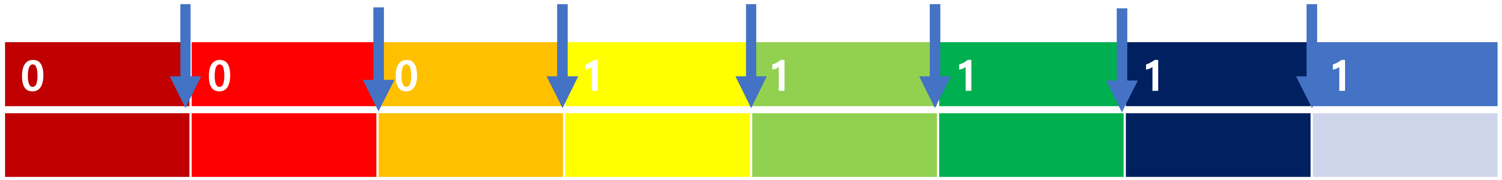
end

Follow same step as in previous section to f

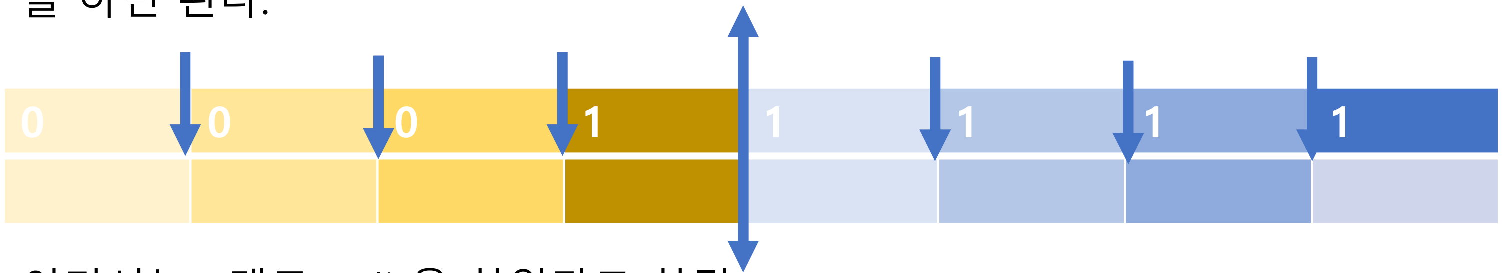
score only among proposed splits.

Solutions

- Approximation을 사용하고자 한다.
  - Data를 특징별로 k개로 나눠서두고(percentile을 사용)
  - → HPT : eps
  - 이것을 Tree 단위(Global) or Split의 분할(Local) 에서 모두 사용을 한다.

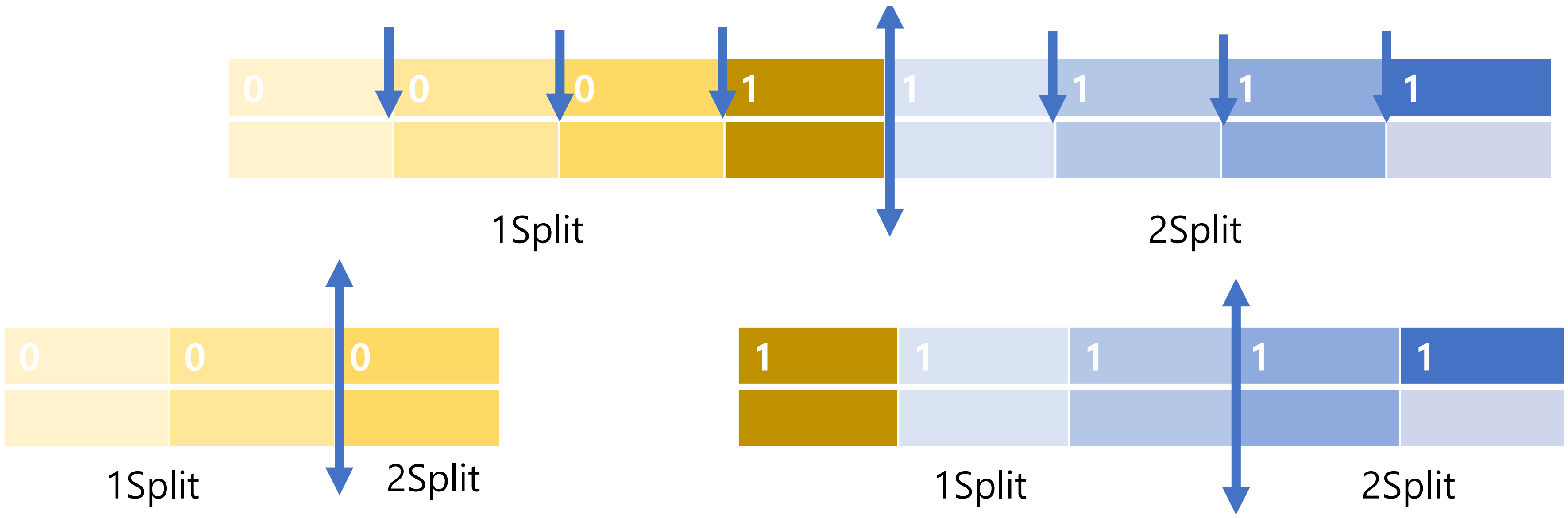


일반적인 경우에는 모든 경우들에 대해서 수행을 하게 된다.
그래서 위의 경우에는 7번의 기준들에 대해서 어디가 잘 분류가 되는지 수행을 하면 된다.



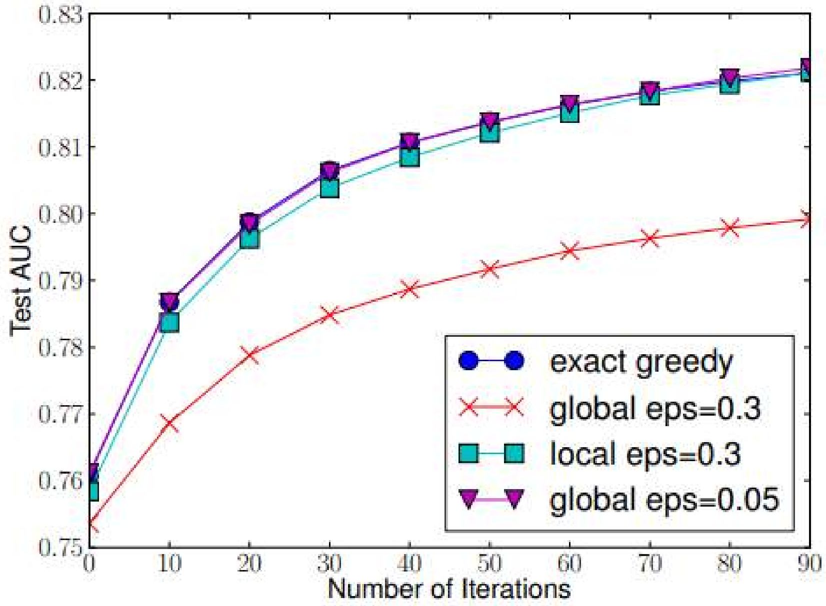
여기서는 2개로 Split을 하였다고 하면
각기에서 1개의 split당 3번만 하면 되기에
총 6번을 수행을 하면 된다
+ 여기서는 각기 split 당 하는 일이 동일하기에 Parallel 병렬 처리까지 가능함!!!!!

물론 이건 극단적인 케이스가 되므로
전체로 하나 동일하게 되지만,
이렇게 대략적으로 근사를 하면서 하게 되므로
경계가 최적이 아닐 수 있는 경우는 늘 존재를 하게 된다!! !But 빠르다..
뒤에 이에 대한 보상의 방식이 나옴.



앞에서 수행을 한 것을 기준으로
Child Node를 수행을 하였다고 하면 위의 그림과 같이 분리가 될 것인데,
Child Node단에서도 위에서 한 것 처럼 2개로 분리를 하고자 하면
각기 child node로 2개씩 분리를 하게 되고
각 split에 개수가 조금 다를 수 있는 경우도 있음(데이터의 percentile 등 고려)

→ **<u>이렇게 Depth가 깊게 아래로 진행을 하여도 Split의 크기는 일정하게 (지금 이 경우에서는 2개로 유지) 유지하면서 진행</u>**
→ 그래서 병렬처리가 계속 가능 & 수도 depth가 진행되면 될 수록 1개 split의 데이터가 줄어드니 속도도 향상



논문의 결과 그래프 인용
→ # of Split = 1 / eps 로 split의 수를 정의하는 HPT

→ 파란색의 exact greedy가 전체를 다 하는 것이며
→ Global eps를 작게하면→ split을 크게 할 수록 전체하는 것과 거의 유사하다.
→ Global eps를 크게하면 → split을 작게하면 성능이 떨어진다.
→ Local과 Global에 대한 비교는 어렵지만, Global에서는 eps를 작게 잡아야 한다는 점!!!!!

Idea_2) Missing Value Handling

- Real_1)실제 데이터는 모든 Feature의 값들이 다 있는 경우가 잘 없음!!!!→ 빵구난 데이터가 상당히 많이 존재를 함
- Real_2) 카테고리 변수에 대한 인코딩에 의해서 0의 값이 너무나도 많이 나타나게 된다!!!! → 특히나 One Hot Encoding의 경우에는 Label Encoding 보다 훨씬 더 많이 존재하게 된다!
- Solution) Sparsity – Aware Split Finding

Algorithm 3: Sparsity-aware Split Finding

Input: I, instance set of current node
Input: $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$
Input: d, feature dimension
Also applies to the approximate setting, only collect statistics of non-missing entries into buckets
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I}, g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  // enumerate missing value goto right
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted(*I_k*, ascent order by* $\mathbf{x}_{jk}$*)* **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
  // enumerate missing value goto left
  $G_R \leftarrow 0,\ H_R \leftarrow 0$
  for j *in sorted(*I_k*, descent order by* $\mathbf{x}_{jk}$*)* **do**
    $G_R \leftarrow G_R + g_j,\ H_R \leftarrow H_R + h_j$
    $G_L \leftarrow G - G_R,\ H_L \leftarrow H - H_R$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split and default directions with max gain

Value	**1.5**	**NaN**	**1.3**	**0.6**	**NaN**	**1.8**	**1.9**
Target	1	0	0	0	0	1	1

Missing value goto right

Value	**1.3**	**0.6**	**1.5**	**1.8**	**1.9**	**NaN**	**NaN**
Target	0	0	1	1	1	0	0

Missing value goto left

Value	**NaN**	**NaN**	**1.3**	**0.6**	**1.5**	**1.8**	**1.9**
Target	0	0	0	0	1	1	1

주어진 데이터들에 대해서 일단 NaN을 오른쪽에 보내두고, 기준을 찾아보고 또 왼쪽으로 보내보고 기준을 찾아본다!

이렇게 했을 때 위의 2가지 경우 중에서는 아래의 Left로 보내는 것이 더 좋음 → 그러면 Missing Value가 오면 Left로 보내는 것을 취하게 된다!!!!

Data

Example	Age	Gender
X1	?	male
X2	15	?
X3	25	female

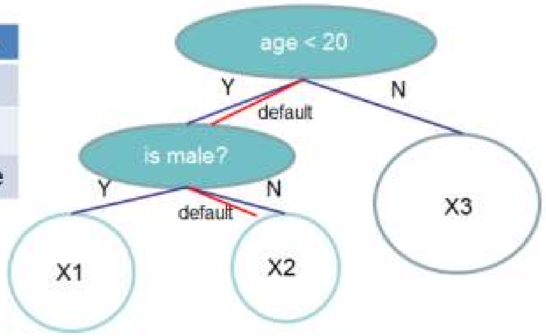


Figure 4: Tree structure with default directions. An example will be classified into the default direction when the feature needed for the split is missing.

논문에 있는 그림인데
앞의 예제처럼 왼쪽/오른쪽으로 NaN을 보내고
각기 어느 방향이 최적인지 했을 때
Age는 왼쪽, Gender는 오른쪽이라고 하자

→ 위의 표의 1번 샘플과 같은 Age가 Missing Data 가 들어오면 Default 방향인 왼쪽으로 보내고
→ 위의 표의 2번 샘플은 Age는 있어서 그 정보를 따라가고, Gender의 정보가 없으니 Default 방향인 오른쪽으로 보낸다.

아래의 y축을 보면 엄청나게 시간의 차이가 발생을 한다!!!!!

빠름;;;

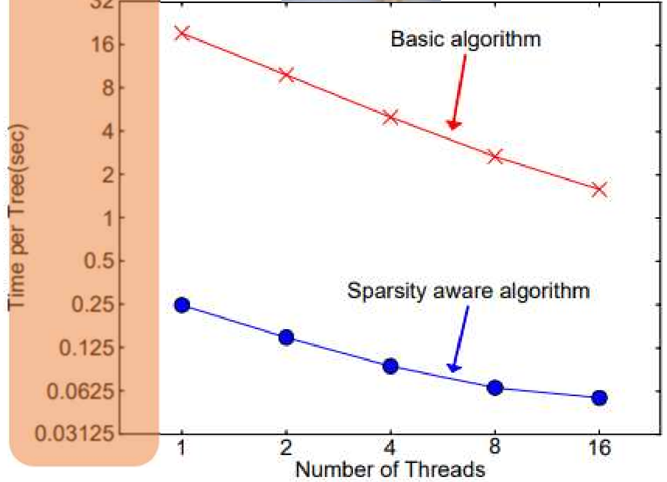


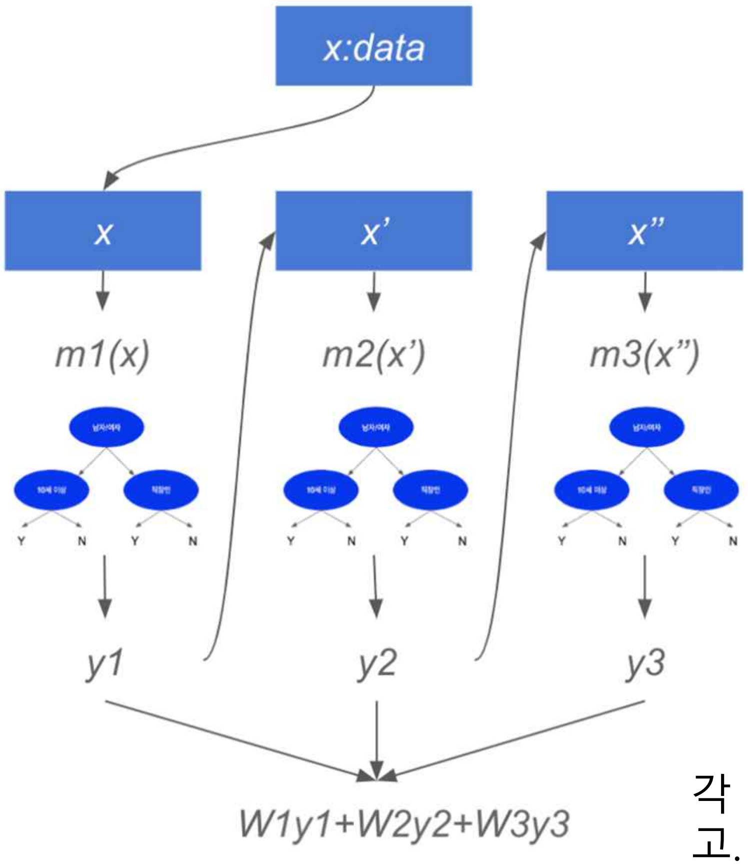
Figure 5: Impact of the sparsity aware algorithm on Allstate-10K. The dataset is sparse mainly due to one-hot encoding. The sparsity aware algorithm is more than 50 times faster than the naive version that does not take sparsity into consideration.

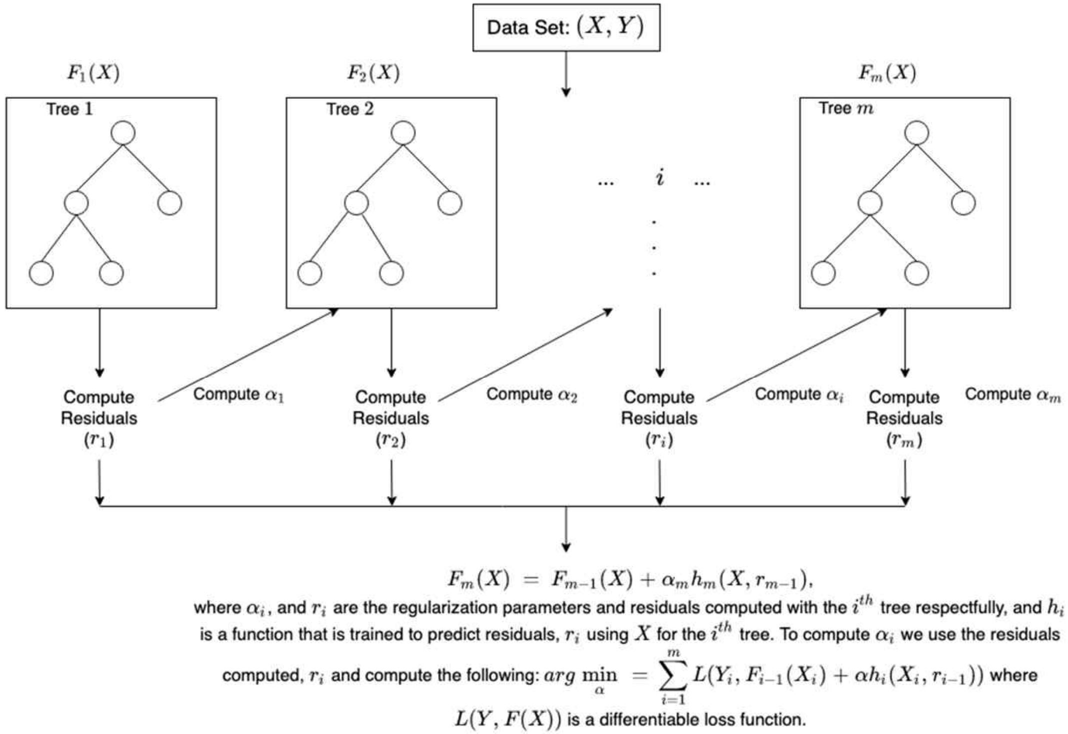
Idea_3) For Speed

- 컬럼들에 대한 사전 정렬을 통한 속도 향상
- Cash 메모리를 사용한 속도 향상
- Out of Core Computing
- Block Compression

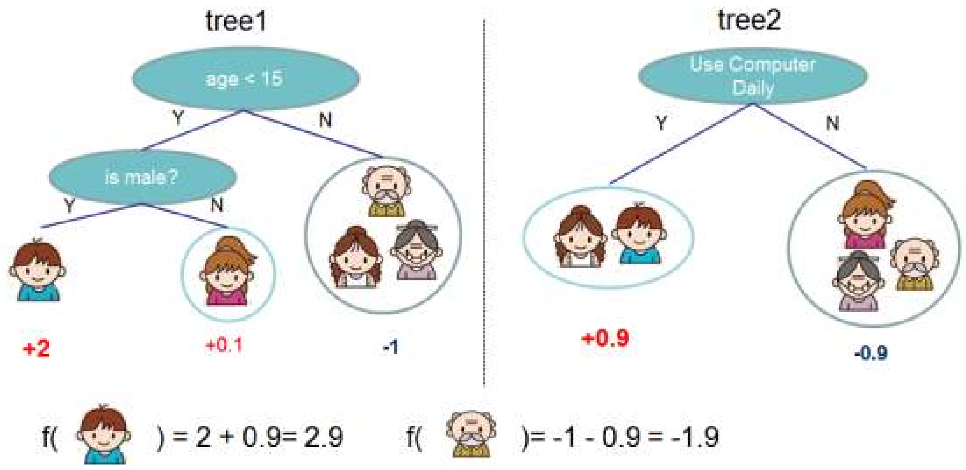
→ 결론은 HW에 기반하여 빠르게 하겠다는 것은데,,저도 잘;;;;ㅜㅜ

Detail_01)





각 단계별 나온 것들에 대한 weight를 가지고. 이에 대한 Regularization 을~



$F = \{f(x) = w_{q(x)}\}$ Function f에 input x를 넣었을 때 나오는 결과는, Tree $q(x)$의 Node w

$q: R^m \Rightarrow T$ q: Structure of each tree
T: Number of Leaves

$w \in R^T$ w: Leaf Weights

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F},$$

$= f_1(Xi) + f_2(Xi) + f_3(Xi) + \dots + f_k(xi)$

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F},$$

$= f_1(X_i) + f_2(X_i) + f_3(X_i) + \ldots + f_k(x_i)$

여기서 $f_k(x_i)$라는 새로운 Tree/ 새로운 함수는 어떻게 선정할 것인가?

→ 기존의 함수에 $f_k(x_i)$가 추가가 되었을 때 Loss Function이 최소가 되는 함수를 찾는다!!!

$$\mathcal{L}(\phi) = \underbrace{\sum_i l(\hat{y}_i, y_i)}_{\text{Training Loss}} + \underbrace{\sum_k \Omega(f_k)}_{\text{규제}}$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

$\hat{y}_i = \hat{y}^{t-1} + f_t(x_i)$

t-th iteration으로 추정!

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

Talyor 근사 식 사용

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

$$= f(a) + f^{(1)}(a)(x-a) + \frac{f^{(2)}(a)}{2!}(x-a)^2 + \cdots$$

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

const

$g_i = \frac{\partial}{\partial \hat{y}^{t-1}} l(y_i, \hat{y}^{t-1})$, $h_i = \frac{\partial^2}{\partial \hat{y}^{t-1}} l(y_i, \hat{y}^{t-1})$

앞에서 근사를하고, 상수에 대한 것을 생략을 하고 하면 다음과 같이 근사

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n}[l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

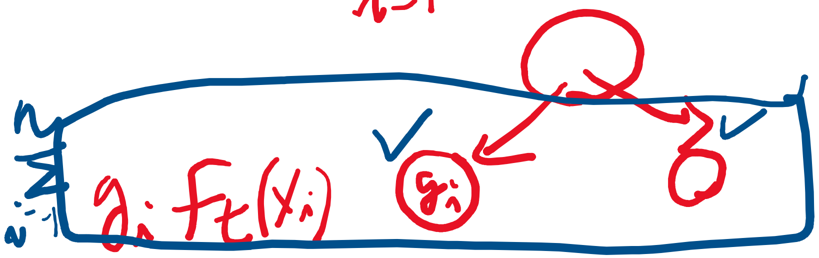
$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

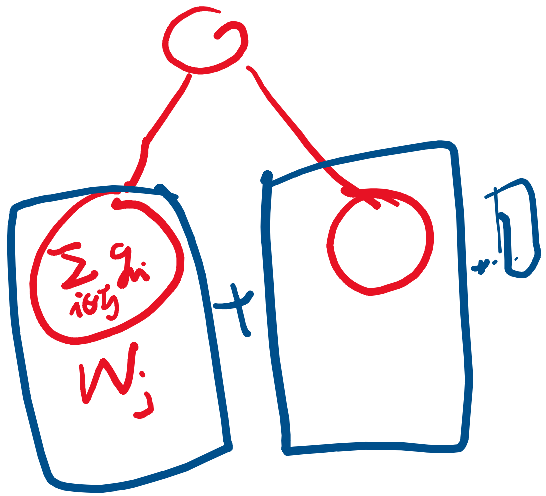
$$= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T$$

n:샘플수instance
T : leaf수

Sigma가 2개가 존재를 하는데,
우리는 Tree를 기준으로 하면서 하기에
Tree인 T를 기준으로 수식을 정리한다.

$$\sum_{i=1}^{n} g_i f_t(x_i) = \sum_{j=1}^{T} (\sum_{i\in I_j} g_i) w_j$$





Tree
leaf기준
sum

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}[(\underbrace{\sum_{i\in I_j} g_i}_{G_j})w_j + \frac{1}{2}(\underbrace{\sum_{i\in I_j} h_i}_{H_j} + \lambda)w_j^2] + \gamma T$$

$$= \sum_{j=1}^{T}\left(\underbrace{G_j w_j + \frac{1}{2}(H_j+\lambda)w_j^2}_{\text{의 최소화}}\right) + \gamma T$$

w_j로 미분 $\rightarrow G_j + (H_j+\lambda)w_j = 0$

$$w_j = -\frac{G_j}{H_j+\lambda}$$

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda},$$

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T}\left[G_j \times \left(-\frac{G_j}{H_j+\lambda}\right) + \frac{1}{2}(H_j+\lambda)\left(-\frac{G_j}{H_j+\lambda}\right)^2\right] + \gamma T$$

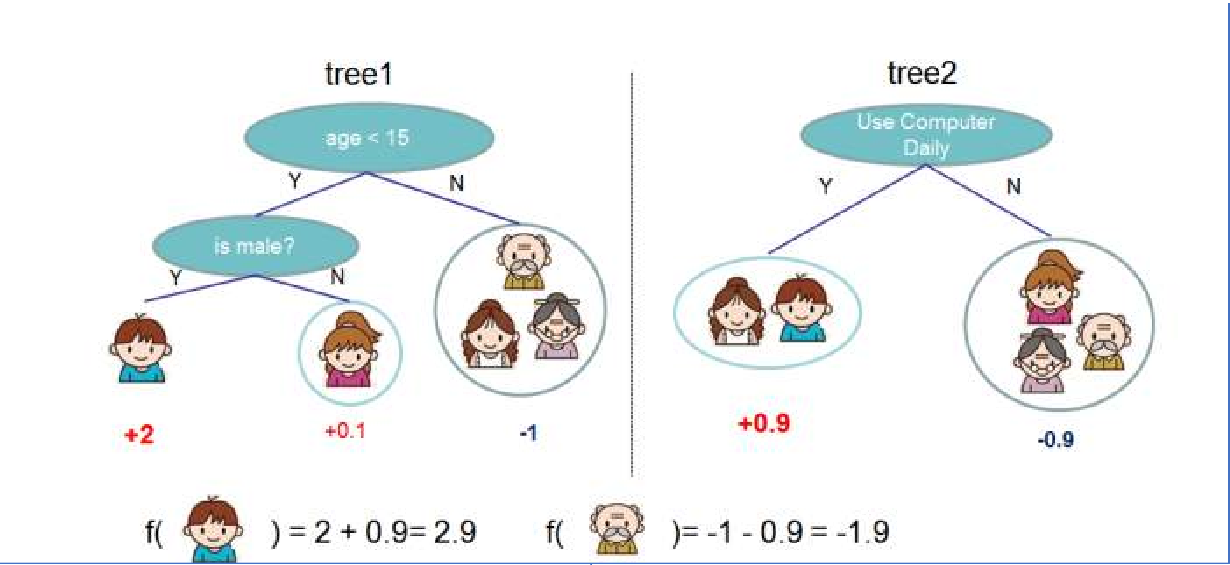
$$= -\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T$$

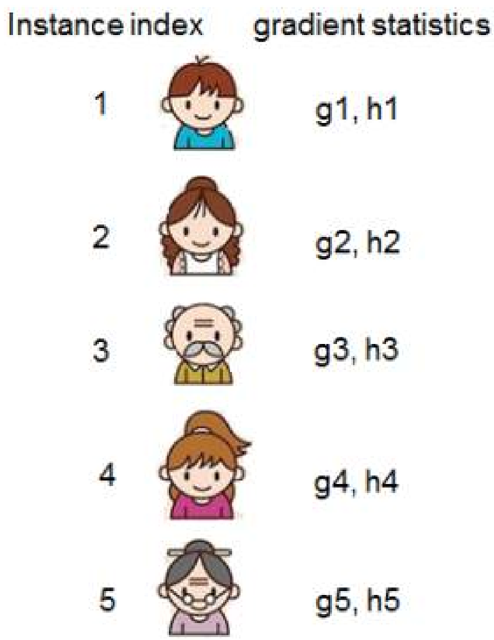
$$\tilde{L}^{(t)} = \sum_{j=1}^{T} \left(G_j \times \left(- \frac{\quad}{H_j+\lambda} \right) + \frac{1}{2}(H_j+\lambda)\left(-\frac{G_j}{H_j+\lambda}\right)^2 \right] + \gamma T$$

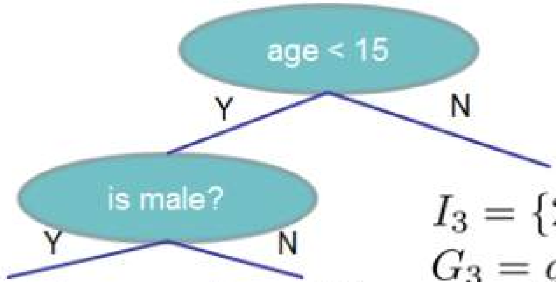
$$= -\sum_{j=1}^{T} \frac{G_j^2}{H_j+\lambda} + \gamma T$$

Tree의 품질 $\frac{G_j^2}{H_j+\lambda}$ ↑ ⇒ $-\sum$ 낮아짐 loss function ⇓

평가 가능함?!







$I_1 = \{1\}$
$G_1 = g_1$
$H_1 = h_1$

$I_2 = \{4\}$
$G_2 = g_4$
$H_4 = h_4$

$I_3 = \{2, 3, 5\}$
$G_3 = g_2 + g_3 + g_5$
$H_3 = h_2 + h_3 + h_5$

$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is

그러면 위의 개념들을 가지고 단순화 시킨 예제를 보자…

- https://towardsdatascience.com/xgboost-regression-explain-it-to-me-like-im-10-2cf324b0bbdb

AGE	MASTER'S DEGREE?	SALARY
23	No	50
24	Yes	70
26	Yes	80
26	No	65
27	Yes	85

AGE	MASTER'S DEGREE?	SALARY
23	No	50
24	Yes	70
26	Yes	80
26	No	65
27	Yes	85

Step 1: Make an Initial Prediction and Calculate Residuals

Residuals = Observed values - Predicted Values

$$\frac{50+70+80+65+85}{5} = 70$$

AGE	MASTER'S DEGREE?	SALARY	Residuals
23	No	50	-20
24	Yes	70	0
26	Yes	80	10
26	No	65	-5
27	Yes	85	15

Step 2: Build an XGBoost Tree

AGE	MASTER'S DEGREE?	SALARY	Residuals
23	No	50	-20
24	Yes	70	0
26	Yes	80	10
26	No	65	-5
27	Yes	85	15

-20, 0, 10, -5, 15

Now we need to calculate something called a **Similarity Score** of this leaf.

$$\text{Similarity Score} = \frac{(Sum\ of\ Residuals)^2}{Number\ of\ Residuals\ +\ \lambda}$$

Regularization Parameter

Now we need to calculate something called a **Similarity Score** of this leaf.

$$\text{Similarity Score} = \frac{(Sum\ of\ Residuals)^2}{Number\ of\ Residuals + \lambda}$$

Regularization Parameter

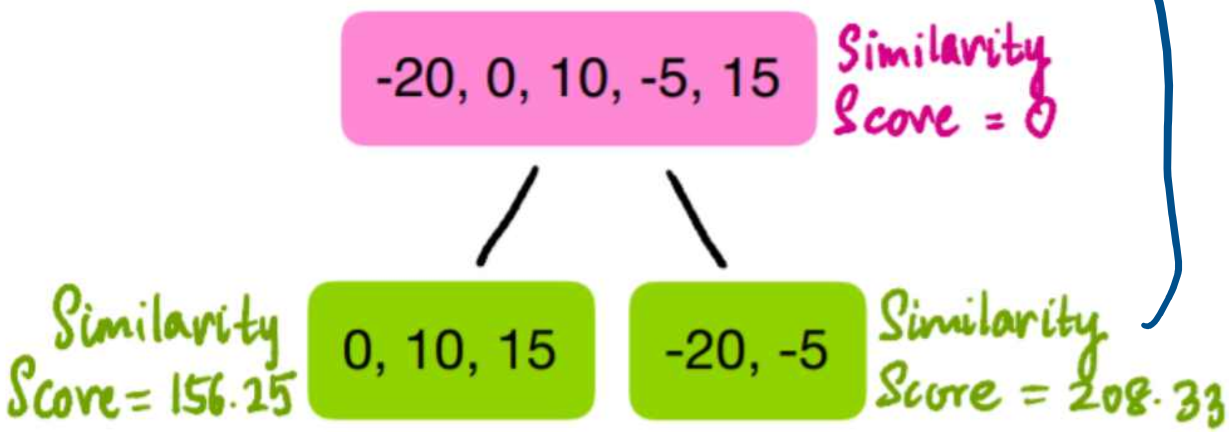
λ=1로 가정

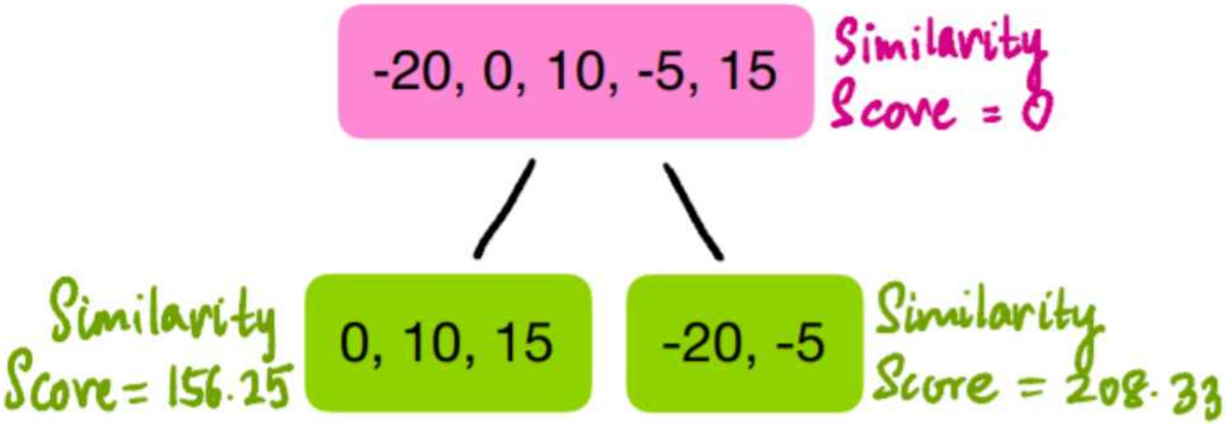
$$\frac{(-20 + 0 + 10 - 5 + 15)^2}{5 + 1} = 0$$

First, let’s try splitting the leaf using *Master’s Degree*?

$$\frac{(-20-5)^2}{2+\underset{\lambda}{1}} = 208.33$$

$$\frac{(0+10+15)^2}{3+\underset{\lambda}{1}} = \frac{625}{4}$$





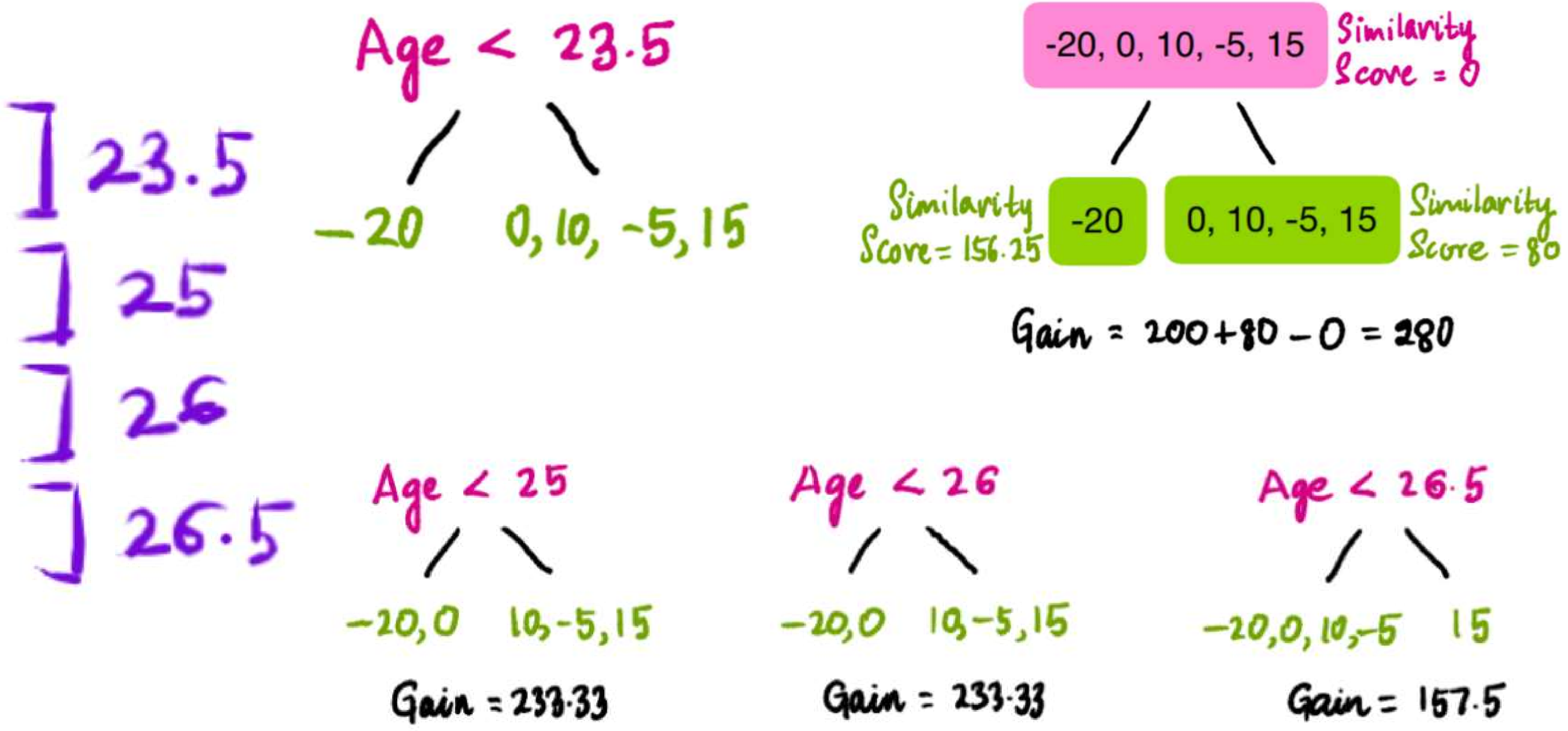
$$\text{Gain} = \text{Left}_{\text{Similarity}} + \text{Right}_{\text{Similarity}} - \text{Root}_{\text{Similarity}}$$

$$= 156.25 + 208.33 - 0 = 364.5833$$

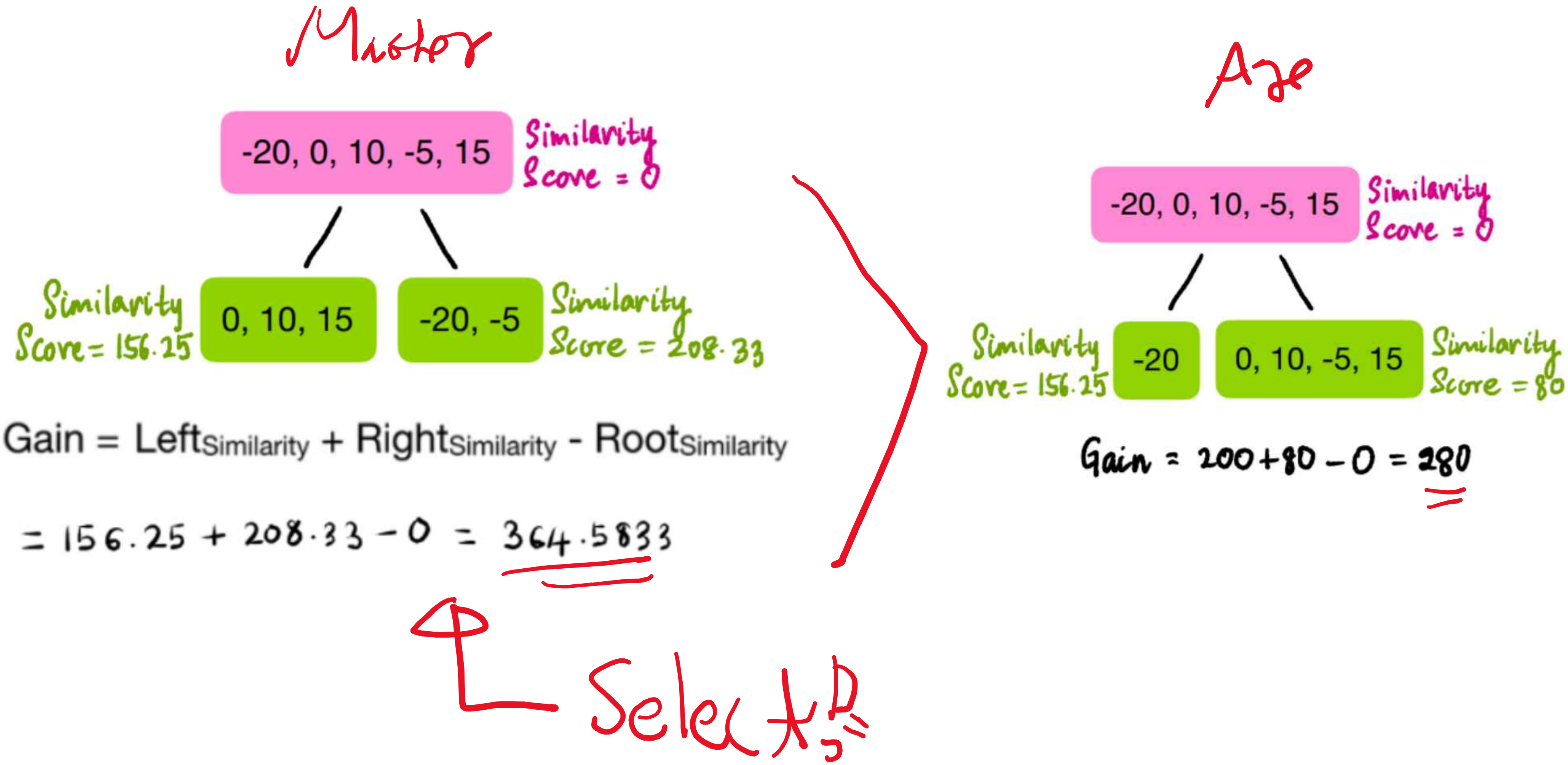
이번에는 Age로 했을 때 앞의 Master Degree로 한 것하고 비교를 해야한다!!!

연속된 값에서는 아래와같이
23.5, 25, 26. 26.5와 여러 개의 기준이 있으니, 이 기준별로
Tree를 해보고 계산을 한다!!!

AGE
23
24
26
26
27

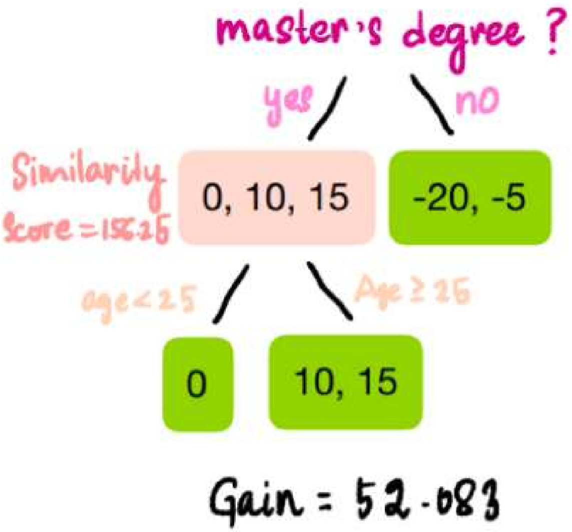


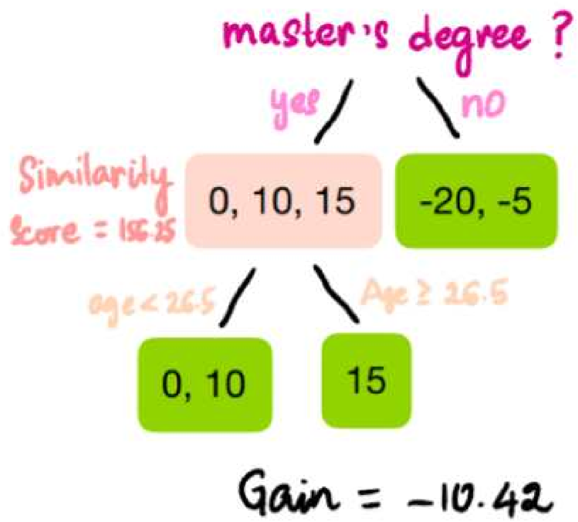
이제는 각기 변수로 해봤으니 Age VS Master Degree



AGE	MASTER'S DEGREE?	SALARY	Residuals
23	No	50	-20
24	Yes	70	0
26	Yes	80	10
26	No	65	-5
27	Yes	85	15

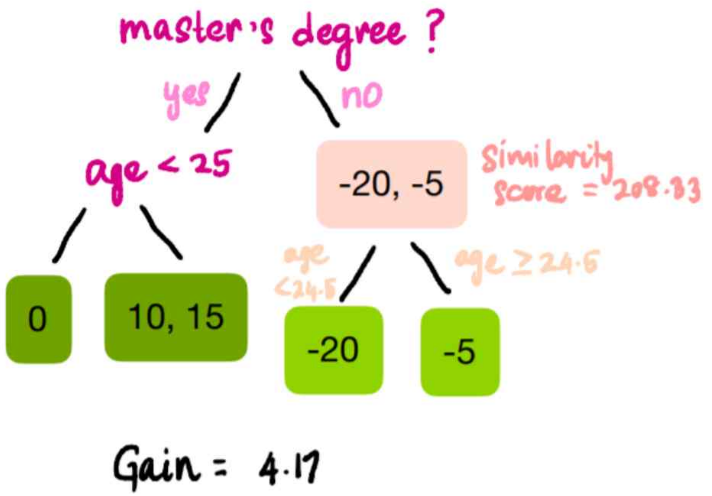
1번 기준은 Master Degree 였으므로, 그 다음 기준에서 Master Degree == Yes인 경우에서 앞에서와 같이 모든 Age에서 하면, 아래와 같이 제일 큰 Gain, 제일 작은 Gain → 제일 큰 값을 취한다!!!!



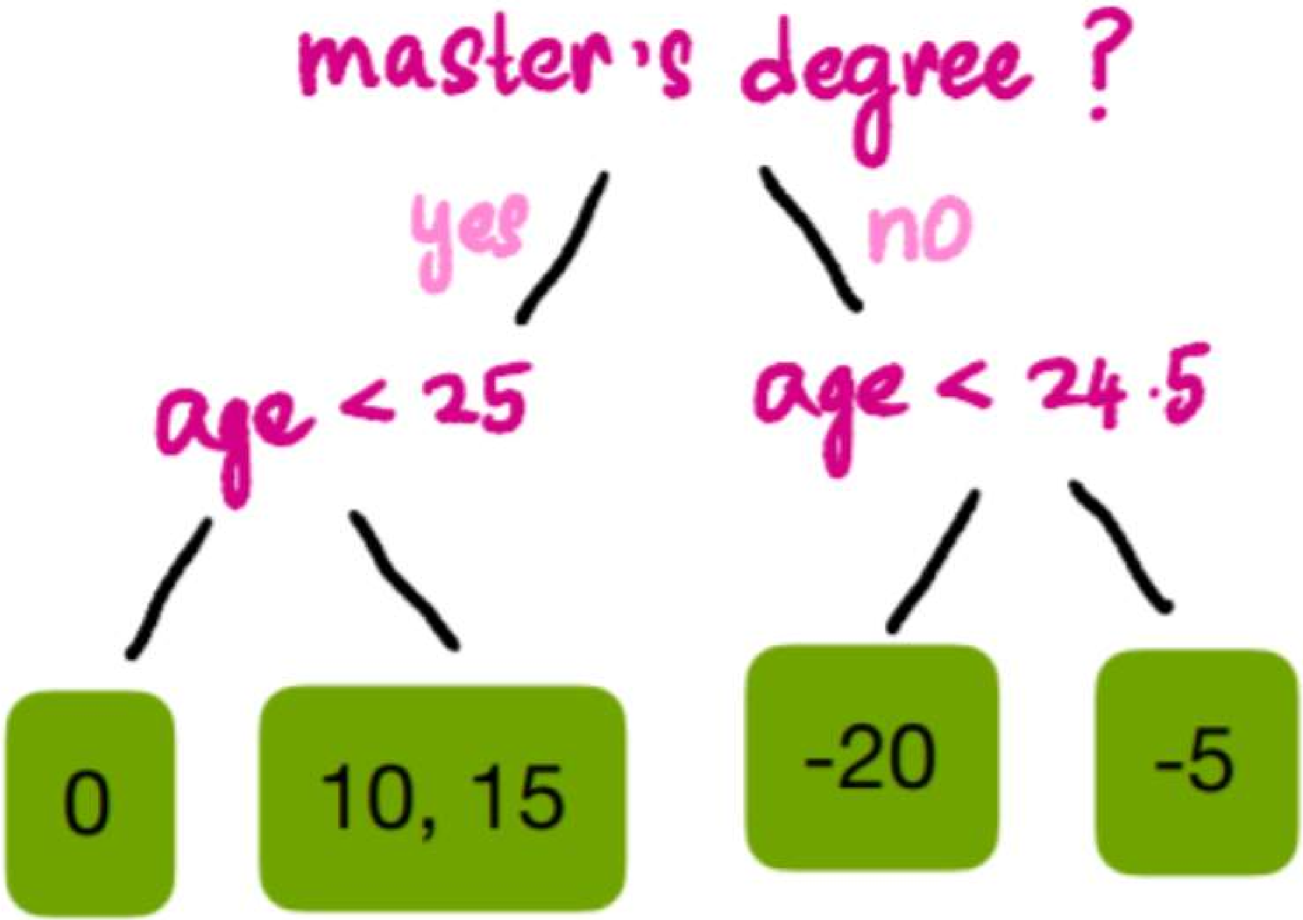


AGE	MASTER'S DEGREE?	SALARY	Residuals
23	No	50	-20
24	Yes	70	0
26	Yes	80	10
26	No	65	-5
27	Yes	85	15

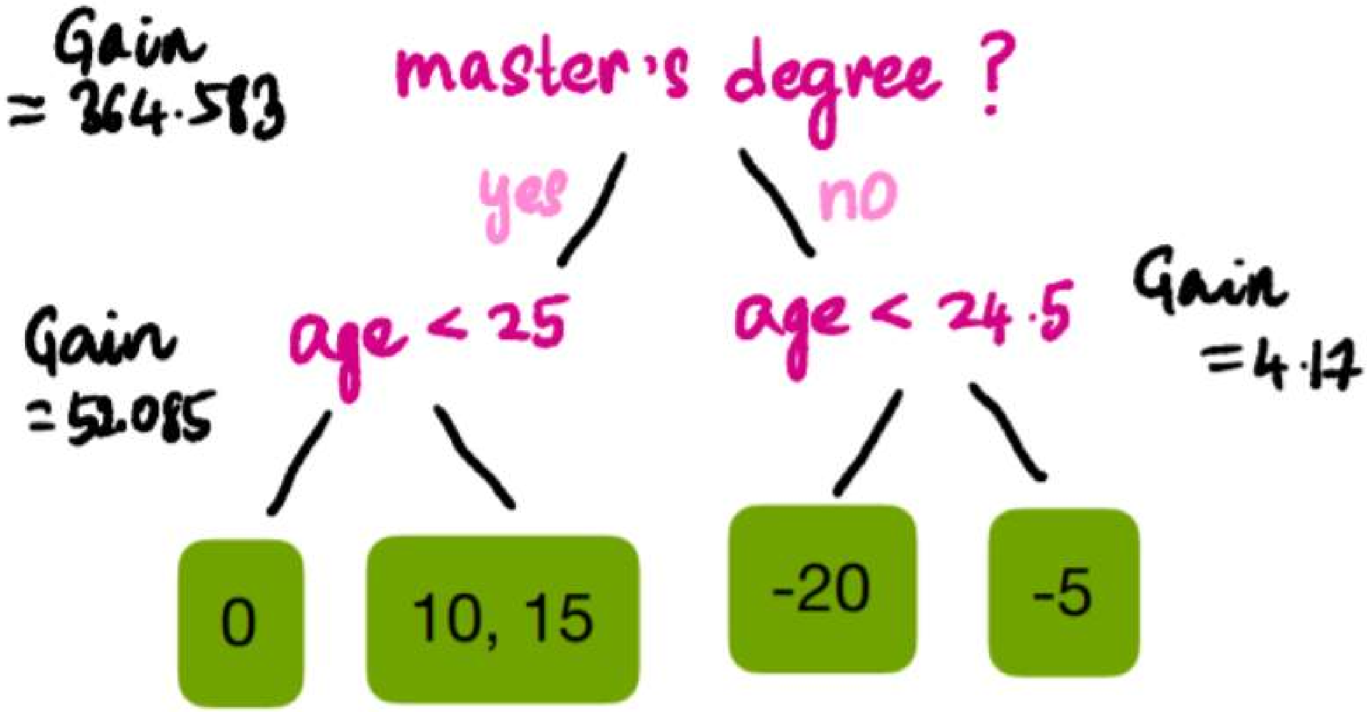
1번 기준은 Master Degree 였으므로, 그 다음 기준에서 Master Degree == Yes인 경우는 앞에서 보면 gain 이 제일 큰 25세를 기준으로 하고, 아래는 Master Degree == No인 경우에서 각기 또 Age 별로 수행한 것임!!!



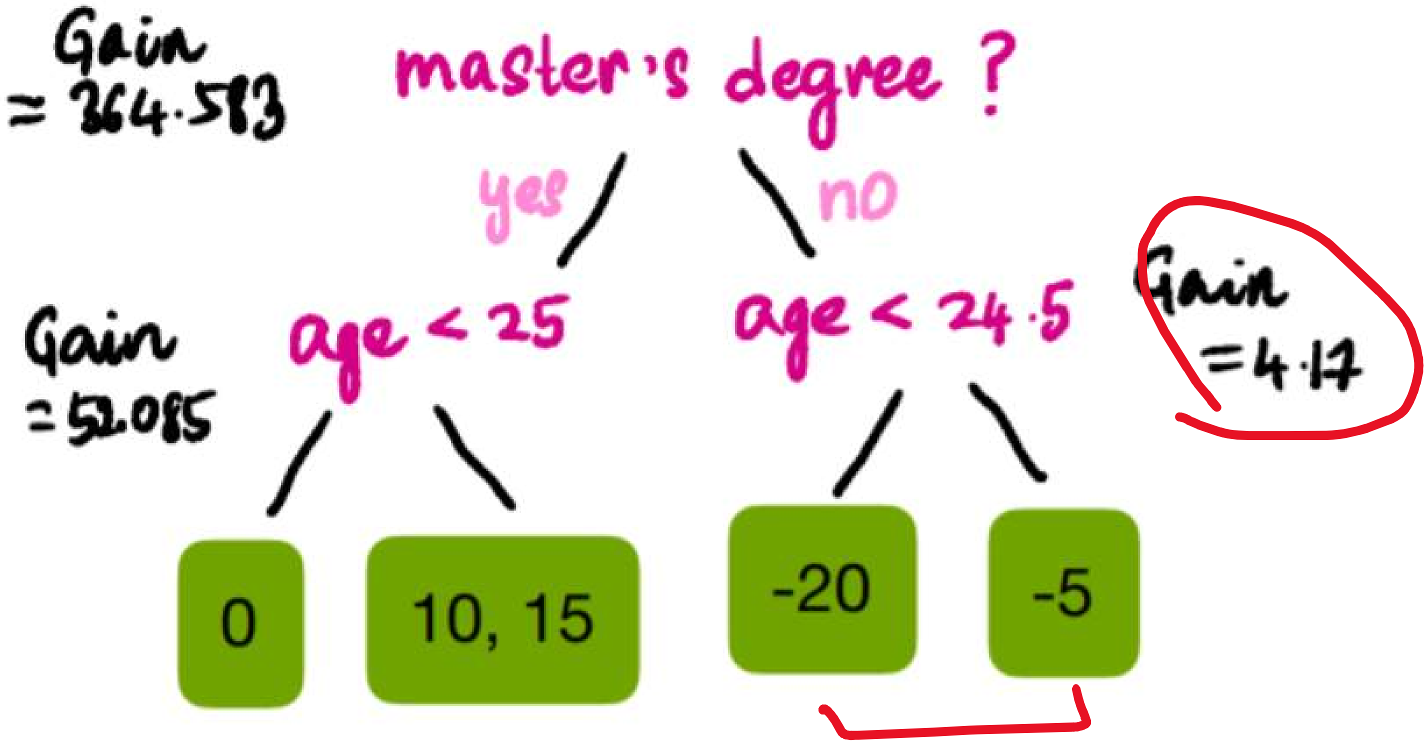
다음과 같은 1단계의 Tree를 완성하게 된다.

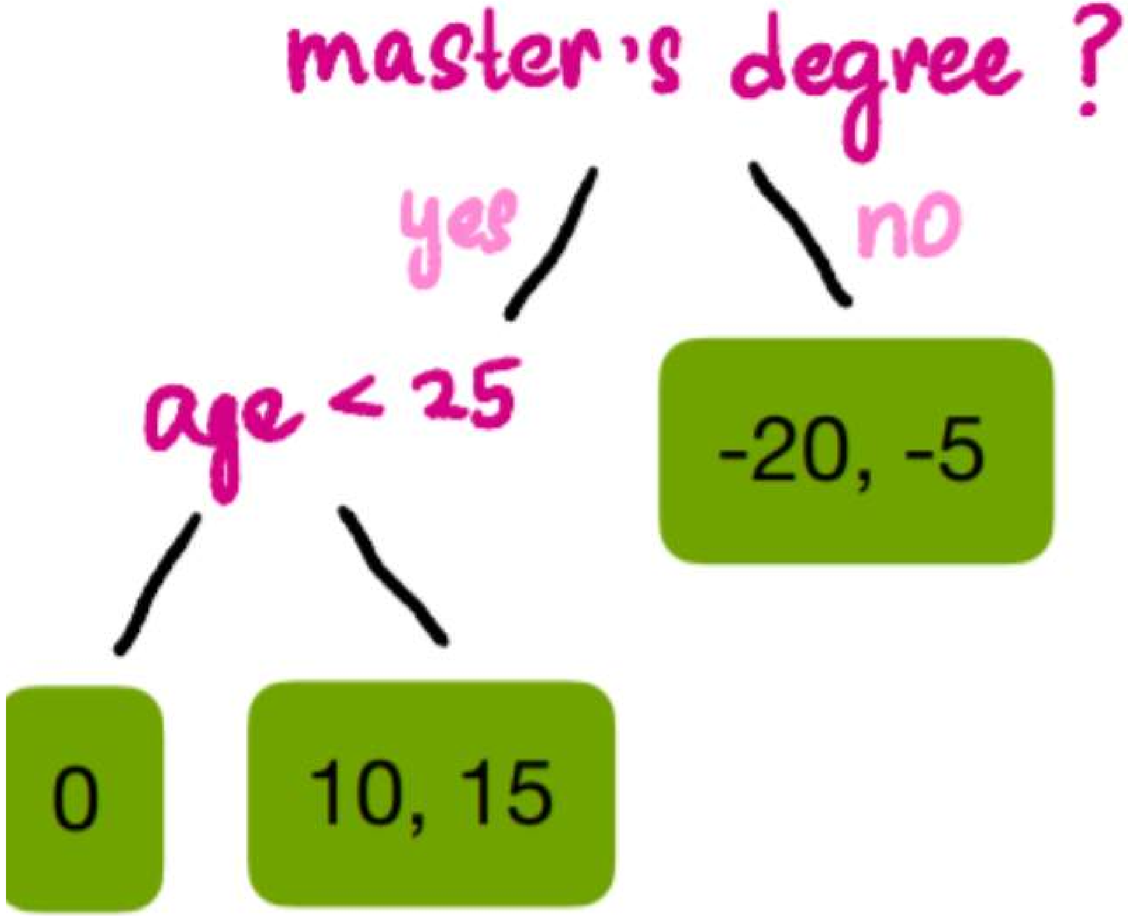


Step 3: Prune the Tree



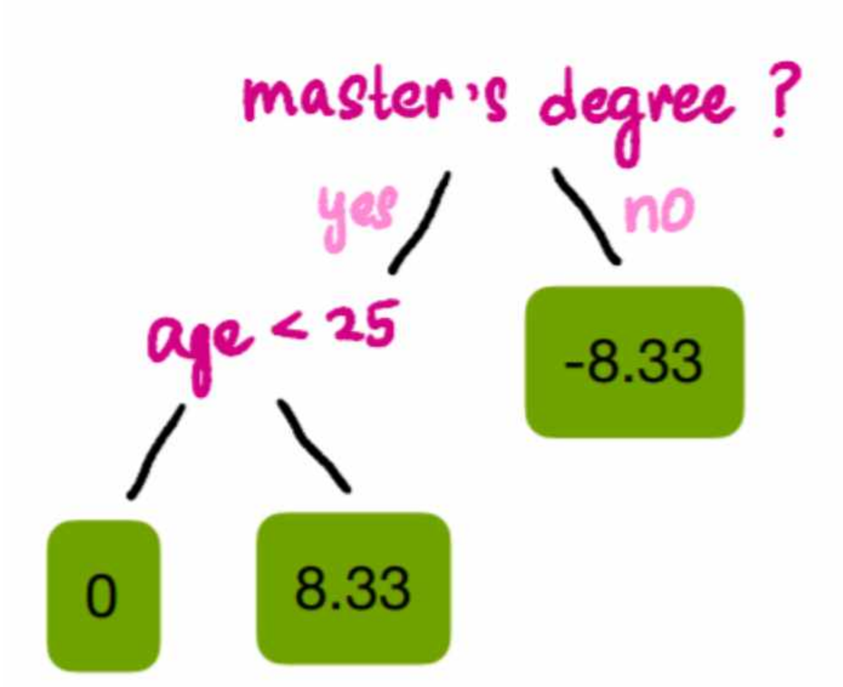
Gamma = 50으로 세팅을 한다면, gamma보다 크지 않은 경우에 대해서는 Split을 하지 않으려고 한다!!!
그래서 아래 그림의 age < 24.5 (Master Degree No Case)를 하나로 병합!!!



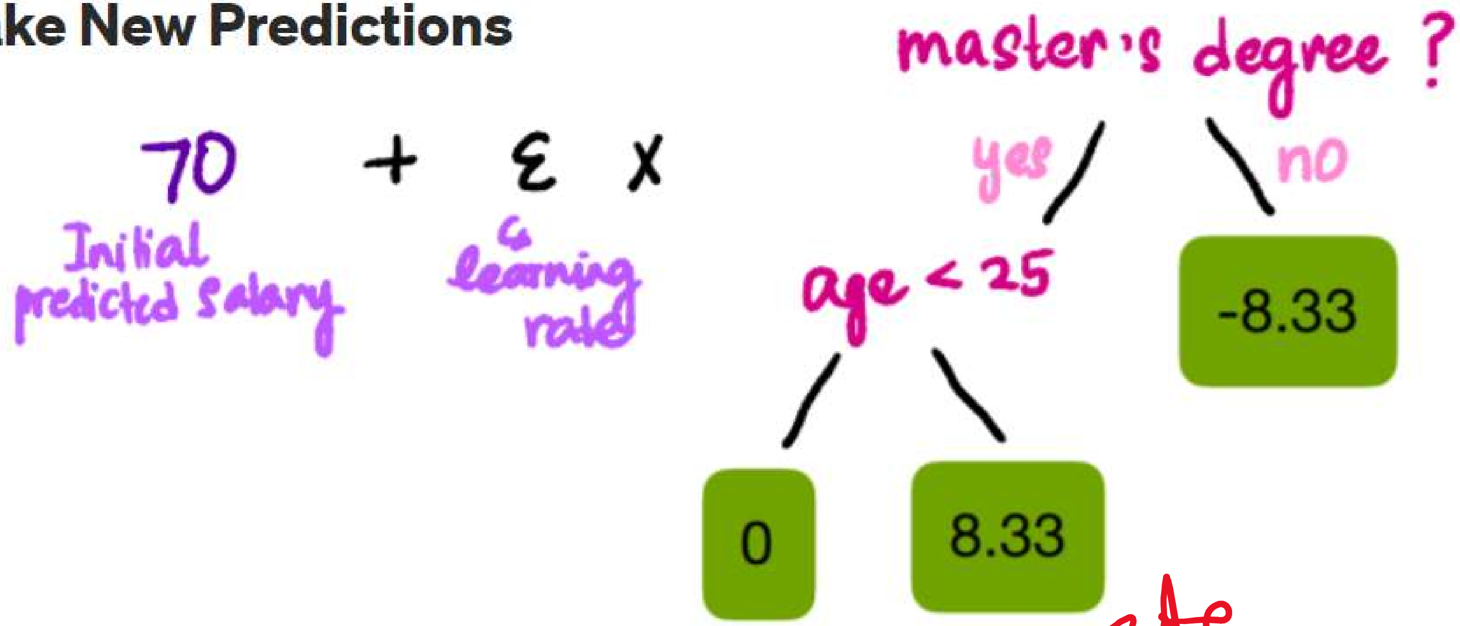

master's degree ?
yes
no
age < 25
-20, -5
0
10, 15

Step 4: Calculate the Output Values of Leaves

$$Output\ Value = \frac{Sum\ of\ Residuals}{Number\ of\ Residuals + \lambda}$$



Step 5: Make New Predictions



learning rate

The XGBoost Learning Rate is ε (eta) and the default value is 0.3. So the predicted value of our first observation will be:

$$70 + 0.3 \times -8.33 = 67.501$$

AGE	MASTER'S DEGREE?	SALARY	Predicted Values
23	No	50	67.501
24	Yes	70	70
26	Yes	80	72.499
26	No	65	67.501
27	Yes	85	72.499

Step 6: Calculate Residuals Using the New Predictions

AGE	MASTER'S DEGREE?	SALARY	Residuals
23	No	50	-17.501
24	Yes	70	0
26	Yes	80	7.501
26	No	65	-2.501
27	Yes	85	12.501

Step 7: Repeat Steps 2-6

$$70 + \varepsilon \times T_1 + \varepsilon \times T_2 + \varepsilon \times T_3 + \ldots + \varepsilon \times T_i$$