DATA MINING 2
Gradient Boost

Riccardo Guidotti

a.a. 2024/2025

Slides edited from StatQuest

UNIVERSITÀ DI PISA

Gradient Boosting for Regression

Gradient Boost – Main Idea

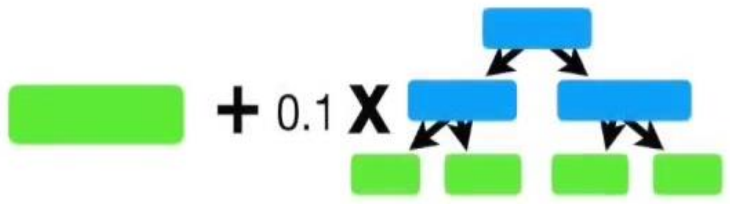
Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

Gradient Boost – Main Idea

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

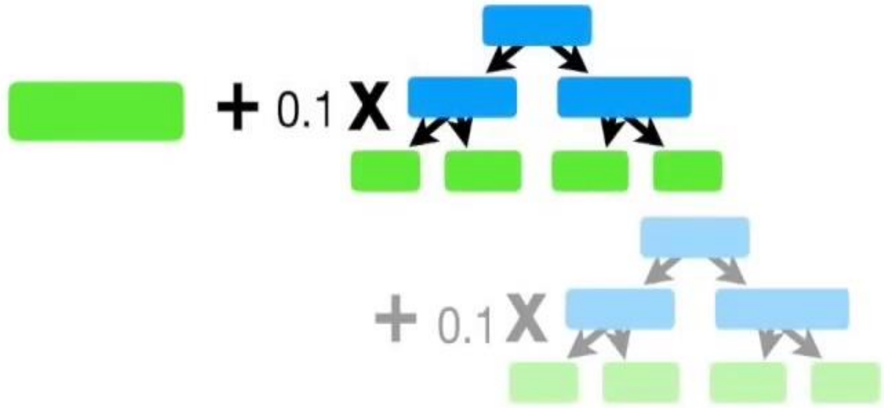
Gradient Boost – Main Idea

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



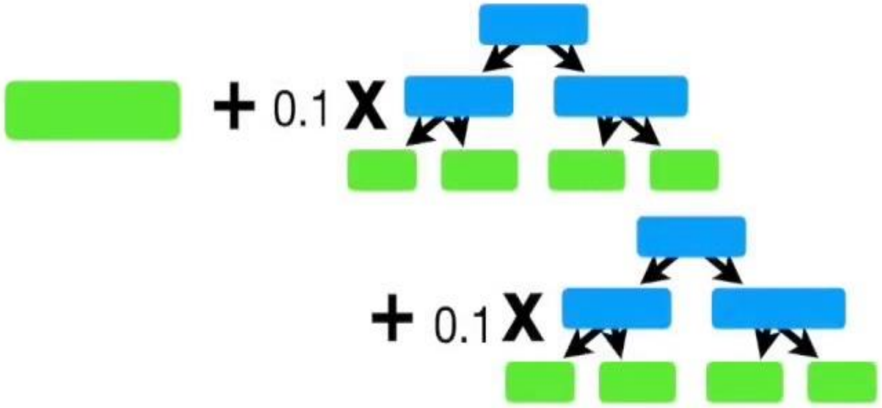
Gradient Boost – Main Idea

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



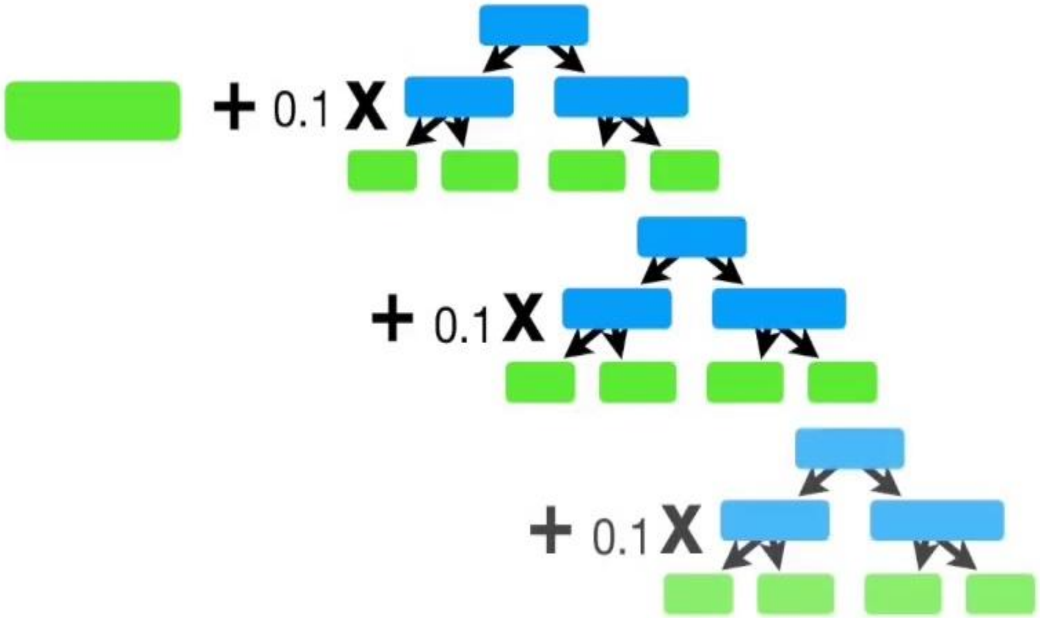
Gradient Boost – Main Idea

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



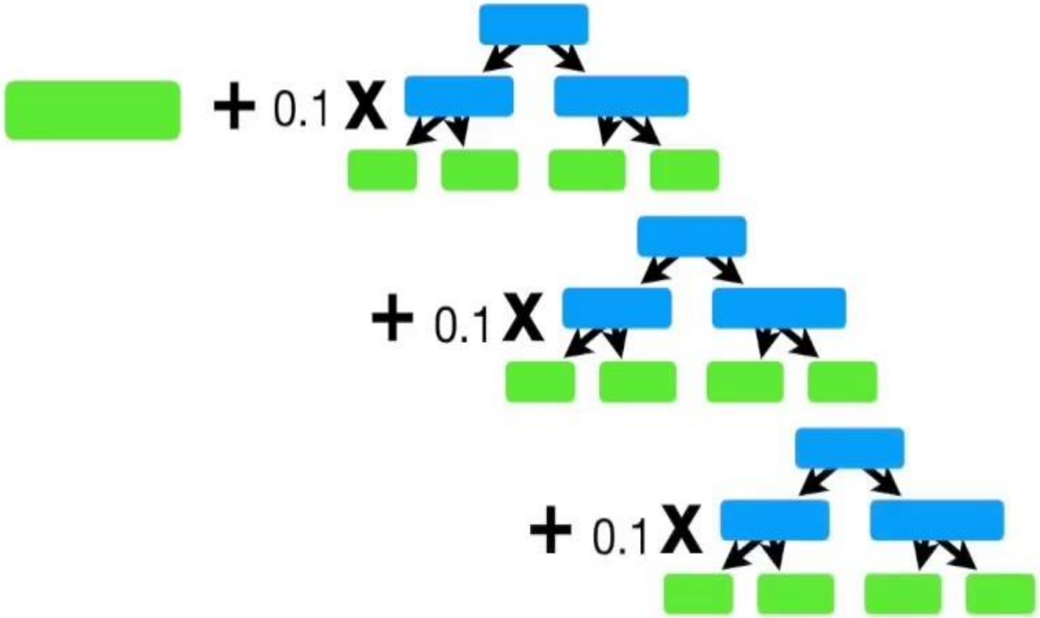
Gradient Boost – Main Idea

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



Gradient Boost – Main Idea

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



Gradient Boost – Example

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

The first thing we do is calculate the average **Weight.**

Example

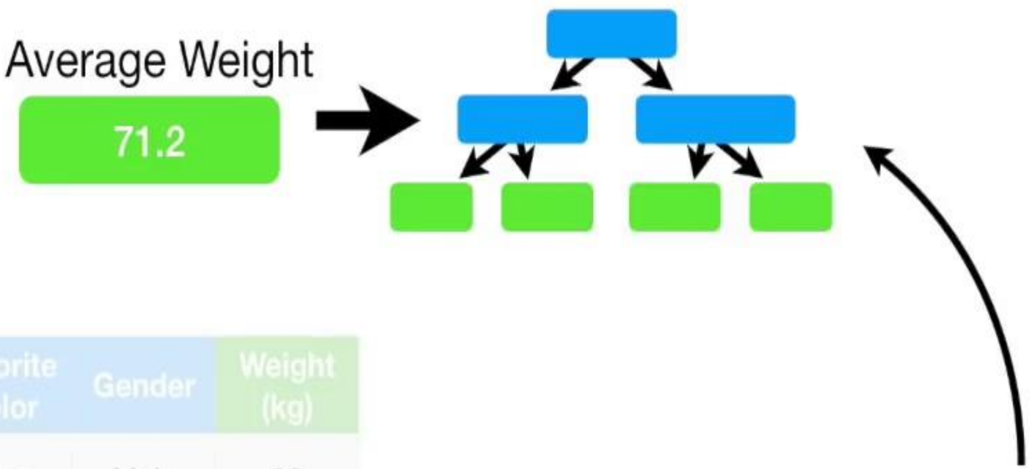
Average Weight

71.2

This is the first attempt at predicting everyone's 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

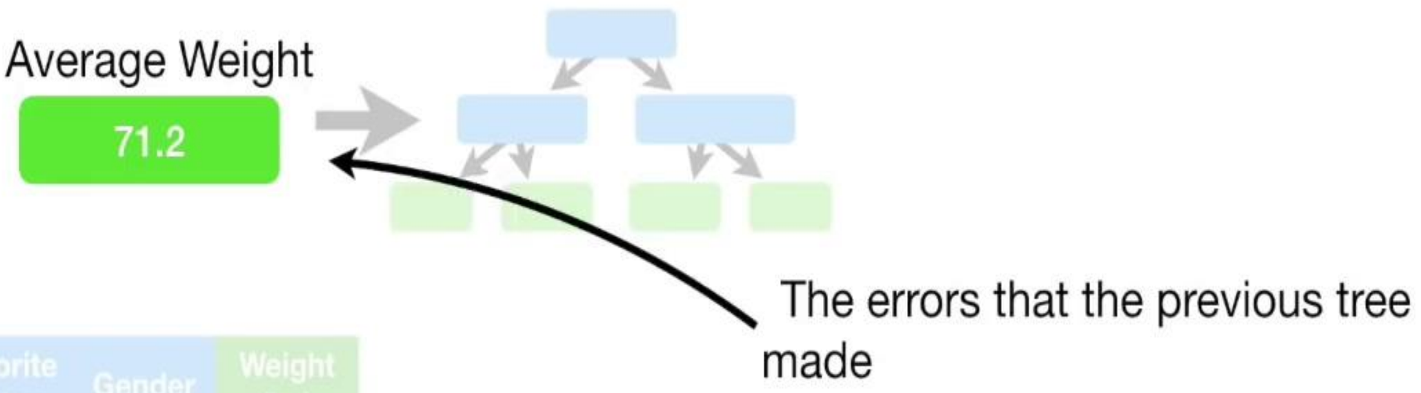
Example



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

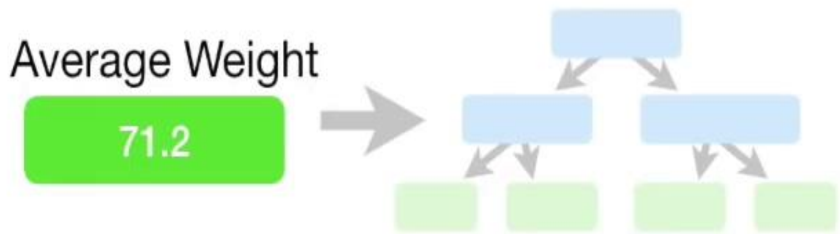
The next thing we do is build a tree based on the errors from the first tree.

Example



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

Example

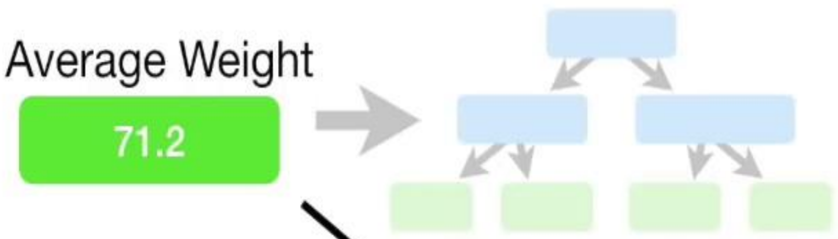


Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

The errors that the previous tree made are the differences between the **Observed Weights**

(**Observed Weight** - Predicted Weight)

Example



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

The errors that the previous tree made are the differences between the **Observed Weights** and the **Predicted Weight, 71.2**.

(**Observed Weight** - **Predicted Weight**)

Example

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.

$(88 - 71.2) = 16.8$

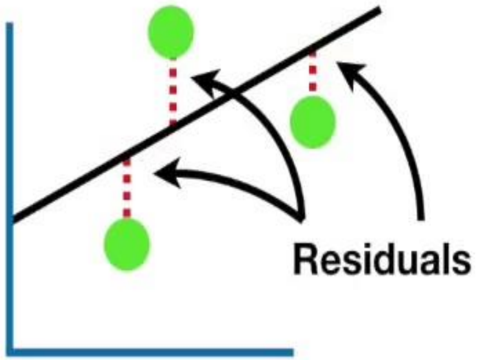
Example

Average Weight

71.2

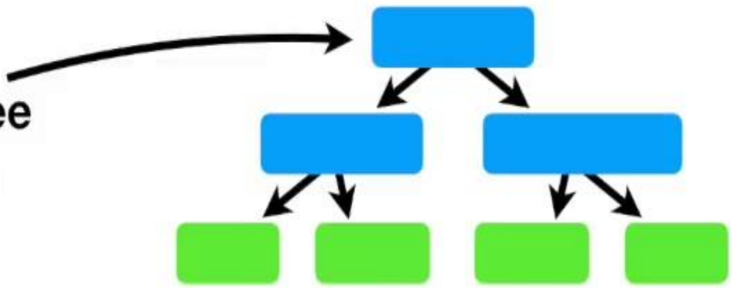
NOTE: The term **Pseudo Residual** is based on **Linear Regression**, where the difference between the **Observed** values and the **Predicted** values results in **Residuals**.

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	



Example

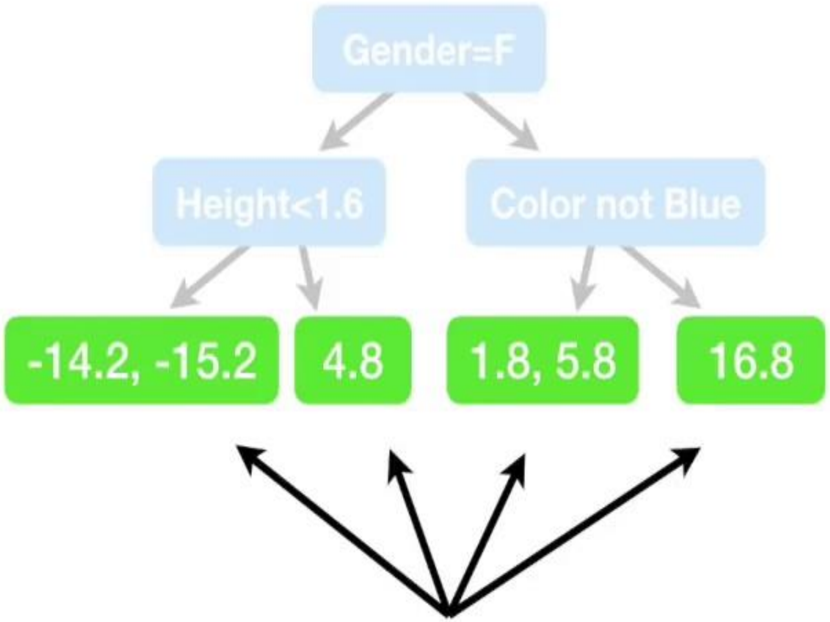
Now we will build a **Tree**



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

Example

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

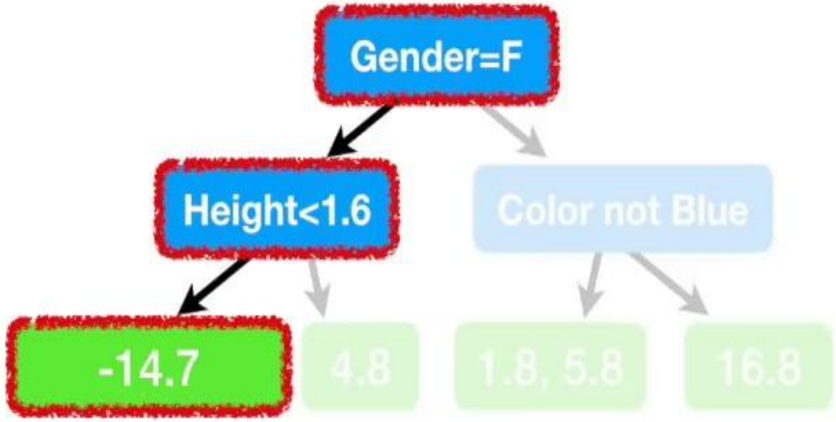


Remember, in this example we are only allowing up to four leaves...

...but when using a larger dataset, it is common to allow anywhere from **8** to **32**.

Example

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

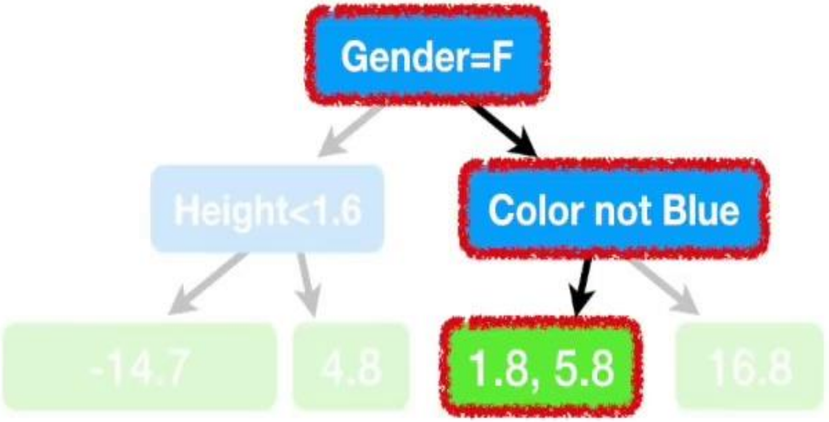


So we replace these residuals with their average.

$$\frac{(-14.2 + -15.2)}{2} = -14.7$$

Example

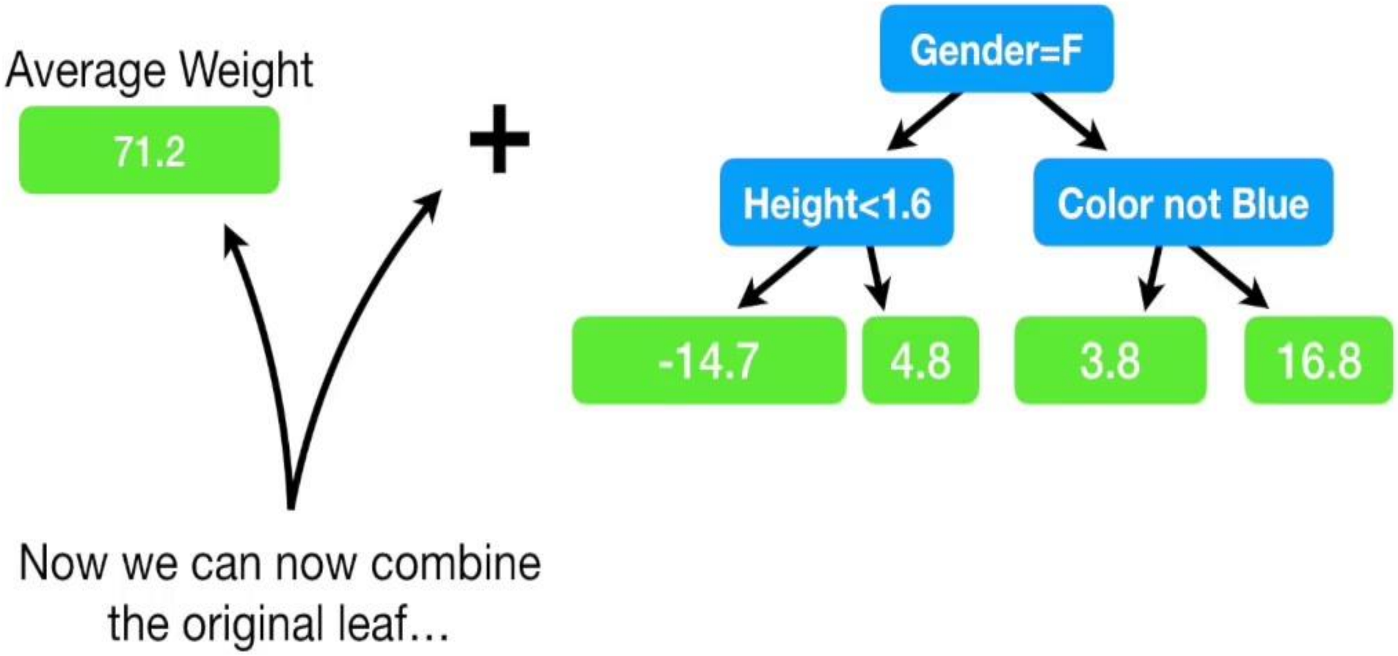
Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2



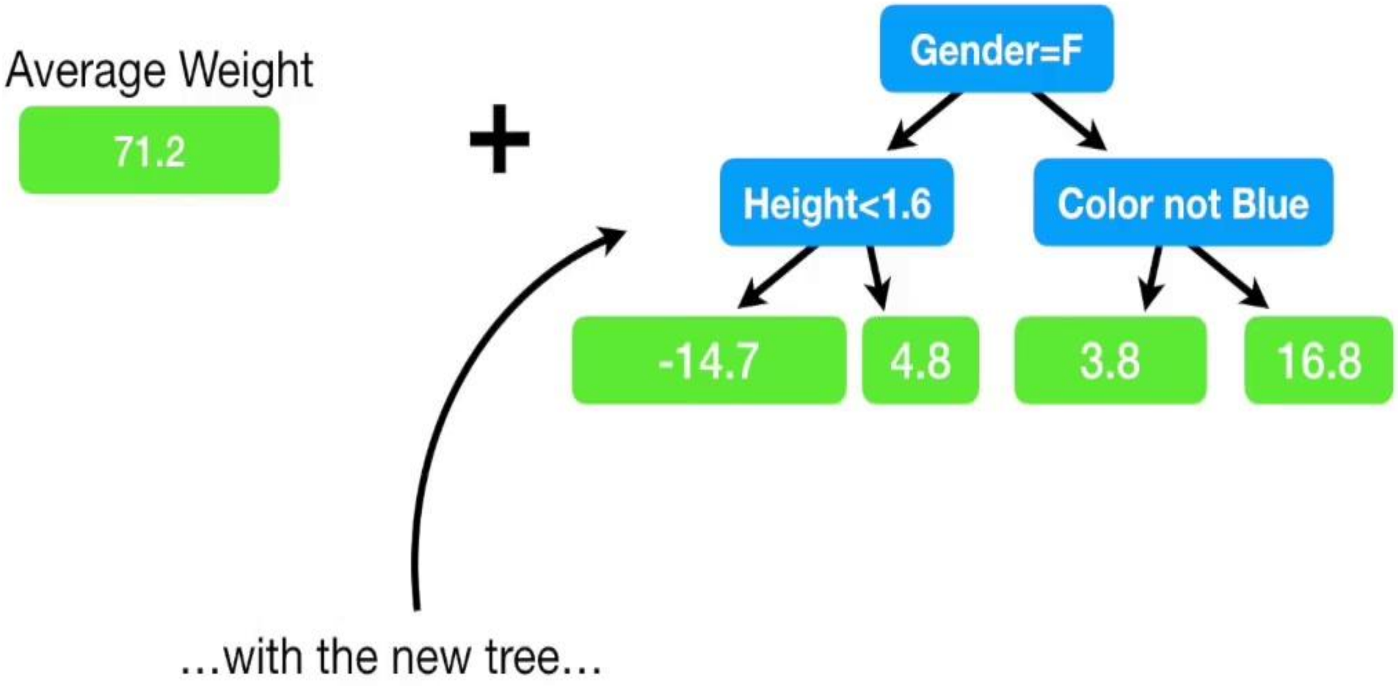
So we replace these residuals with their average.

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

Example

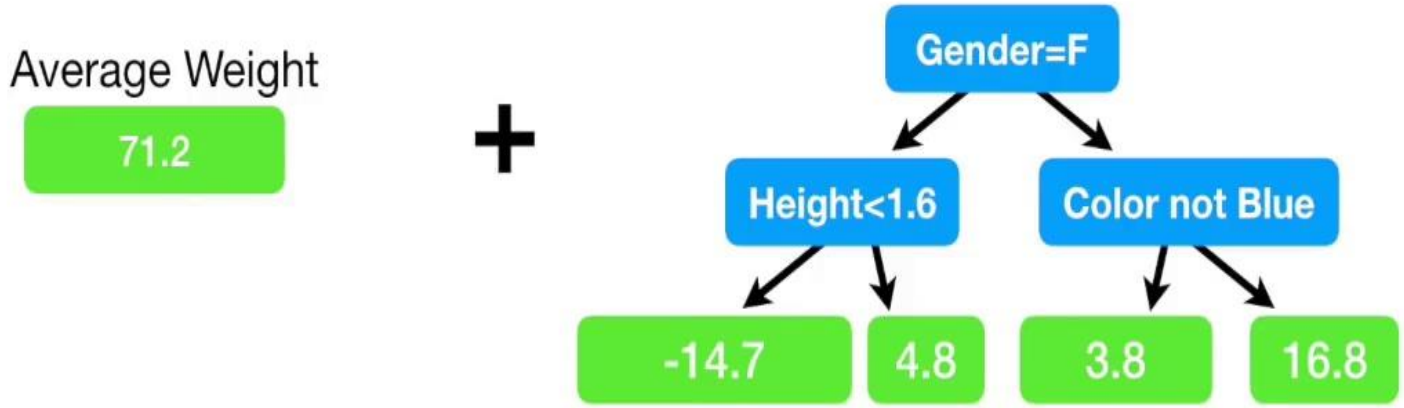

Average Weight
71.2
+
Gender=F
Height<1.6
Color not Blue
-14.7
4.8
3.8
16.8
Now we can now combine
the original leaf…

Example



Average Weight
71.2
+
Gender=F
Height<1.6
Color not Blue
-14.7
4.8
3.8
16.8
...with the new tree...

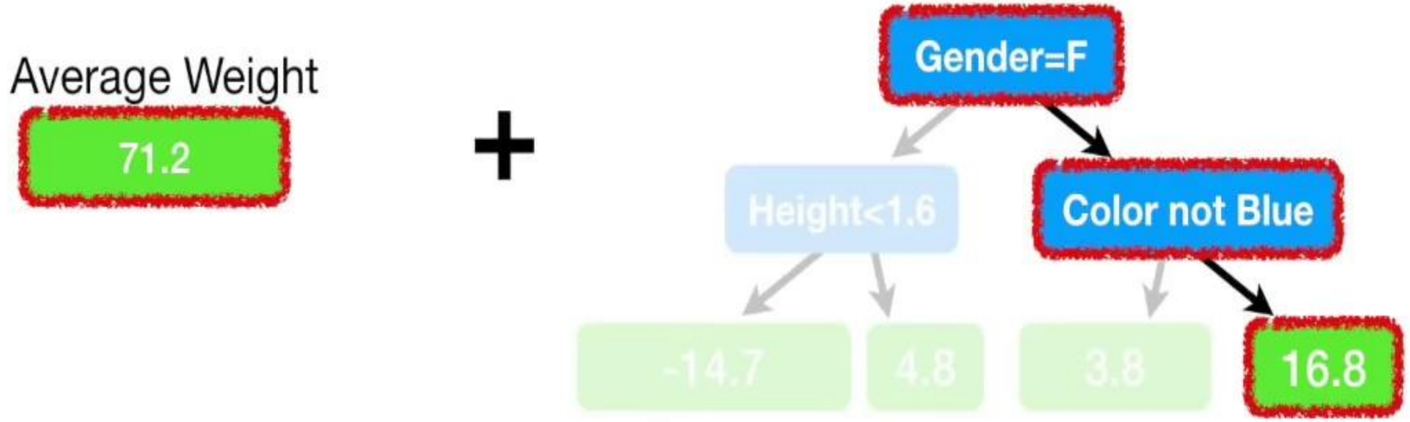
Example



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**.

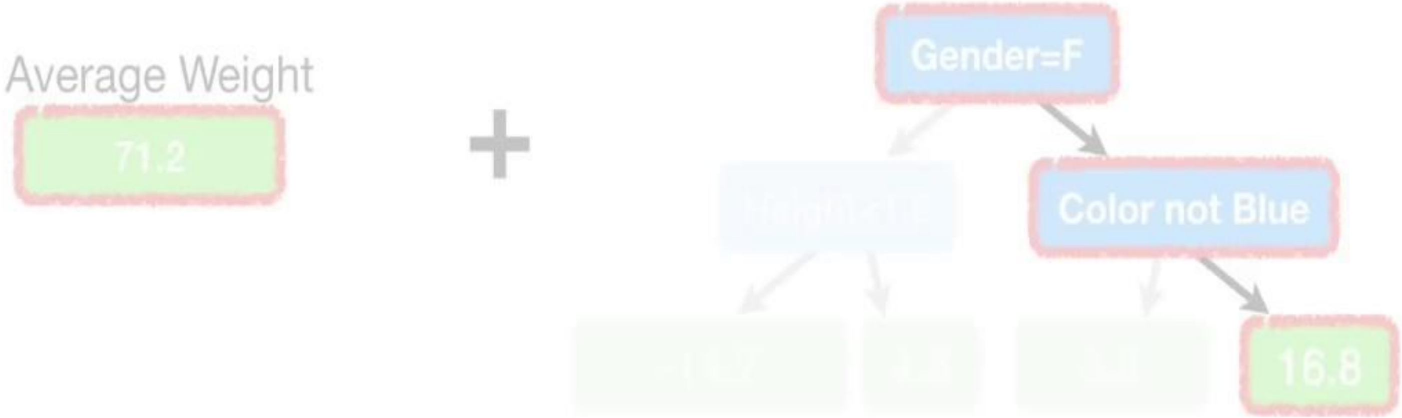
Example



…so the **Predicted Weight** = 71.2 + 16.8 = 88

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

Example

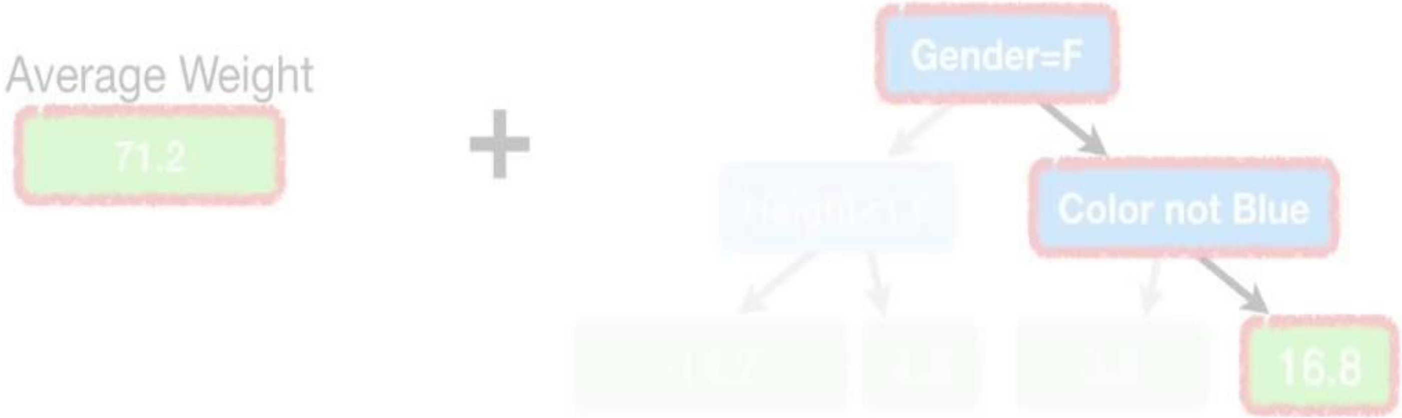


Predicted Weight = 71.2 + 16.8 = 88

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

Is this awesome???

Example

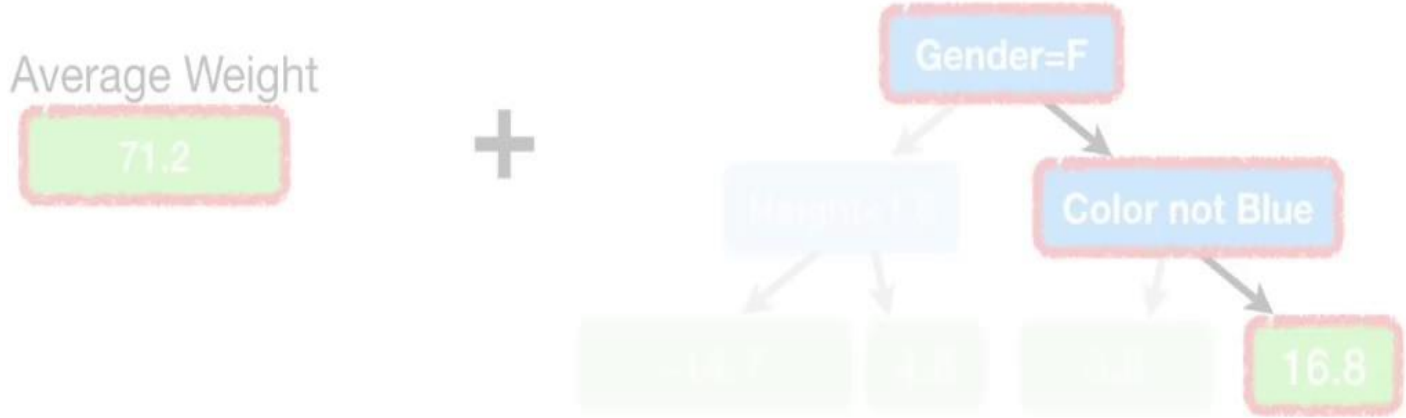


Predicted Weight = 71.2 + 16.8 = 88

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

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

Example

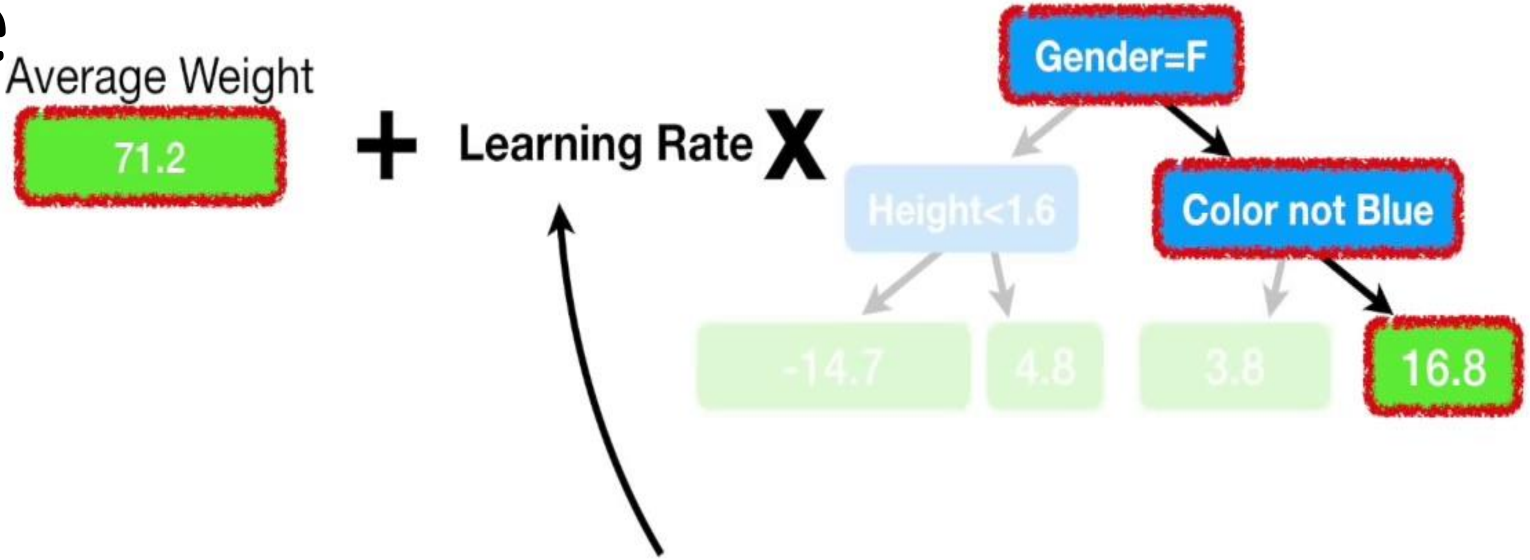


Predicted Weight = 71.2 + 16.8 = 88

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

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

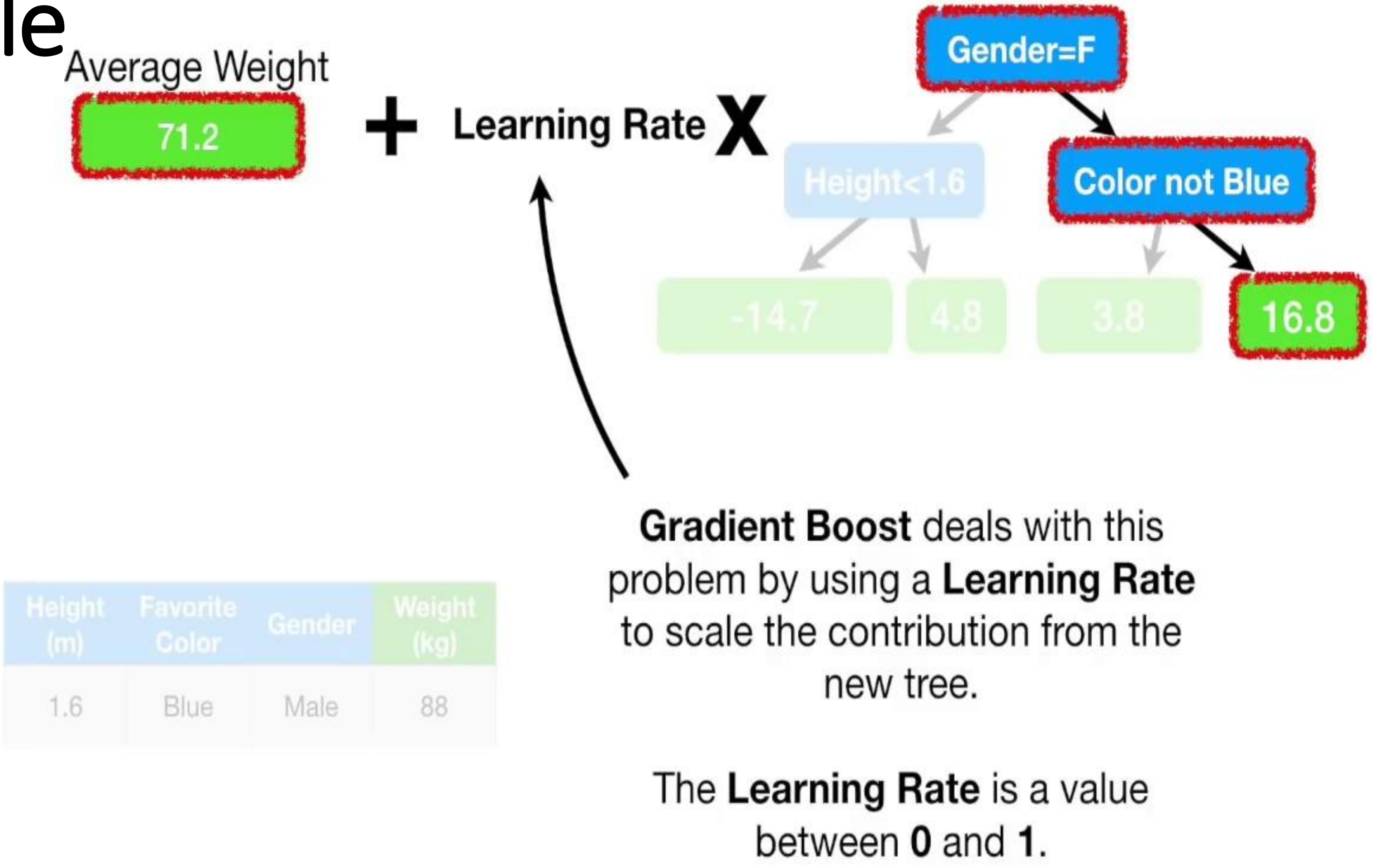
Example



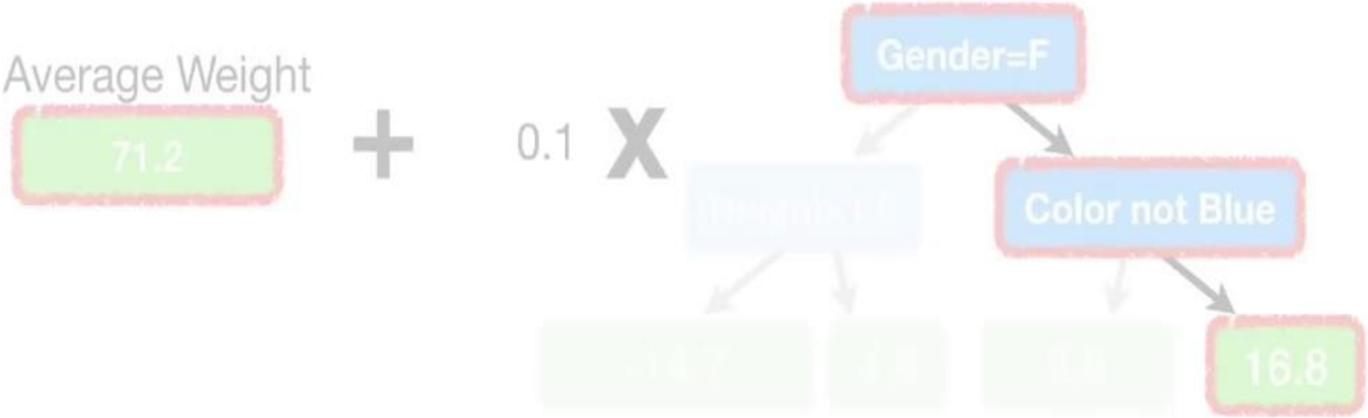
Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

Gradient Boost deals with this problem by using a **Learning Rate** to scale the contribution from the new tree.

Example


Average Weight
71.2
+ Learning Rate X
Gender=F
Height<1.6
Color not Blue
-14.7
4.8
3.8
16.8
Height (m)
Favorite Color
Gender
Weight (kg)
1.6
Blue
Male
88
Gradient Boost deals with this problem by using a **Learning Rate** to scale the contribution from the new tree.
The **Learning Rate** is a value between **0** and **1**.

Example

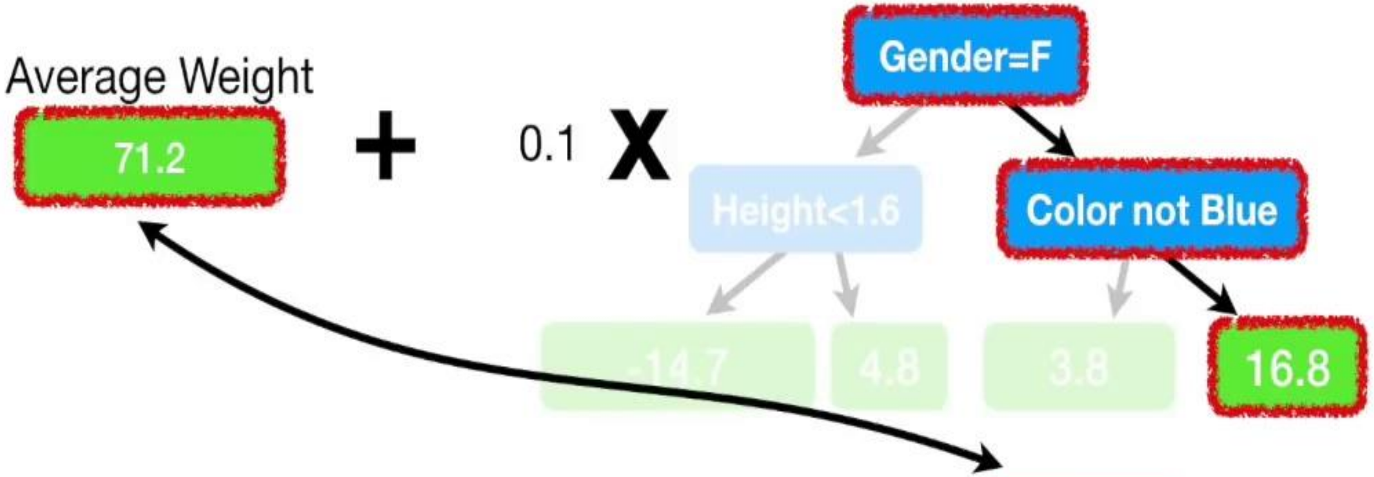


Predicted Weight = 71.2 + (0.1 × 16.8) = 72.9

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

With the **Learning Rate** set to **0.1**, the new **Prediction** isn't as good as as it was before...

Example

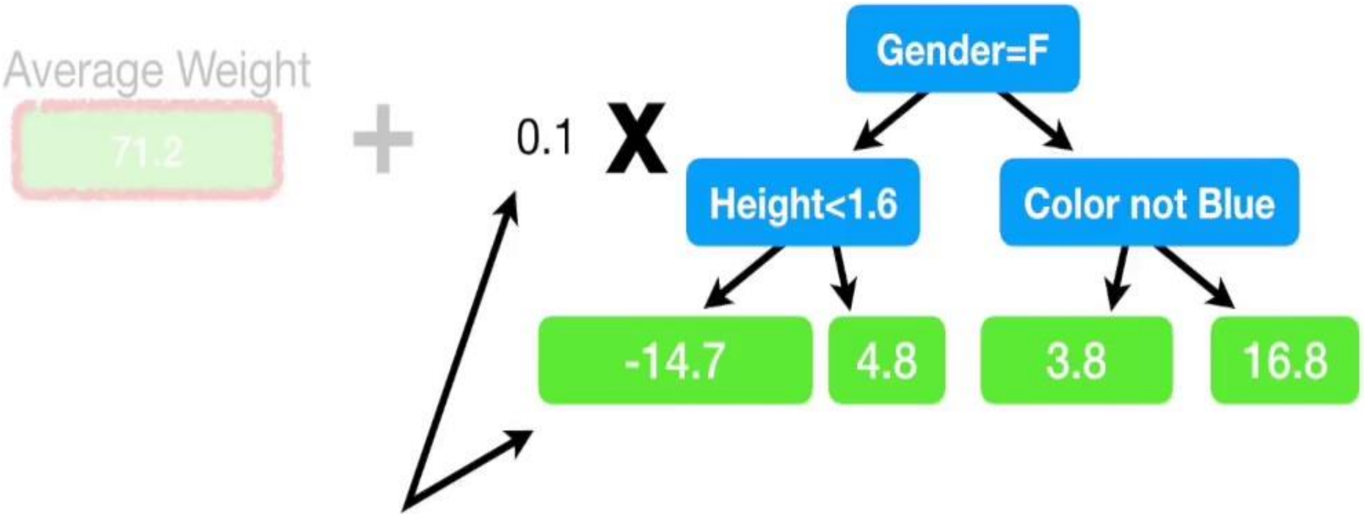


Predicted Weight = 71.2 + (0.1 × 16.8) = 72.9

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

…but it's a little bit better than the **Prediction** made with just the original leaf, which predicted that all samples would weigh **71.2**.

Example



In other words, scaling the tree by the **Learning Rate** results in a small step in the right direction.

Example

So let's build another tree so we can take another small step in the right direction.

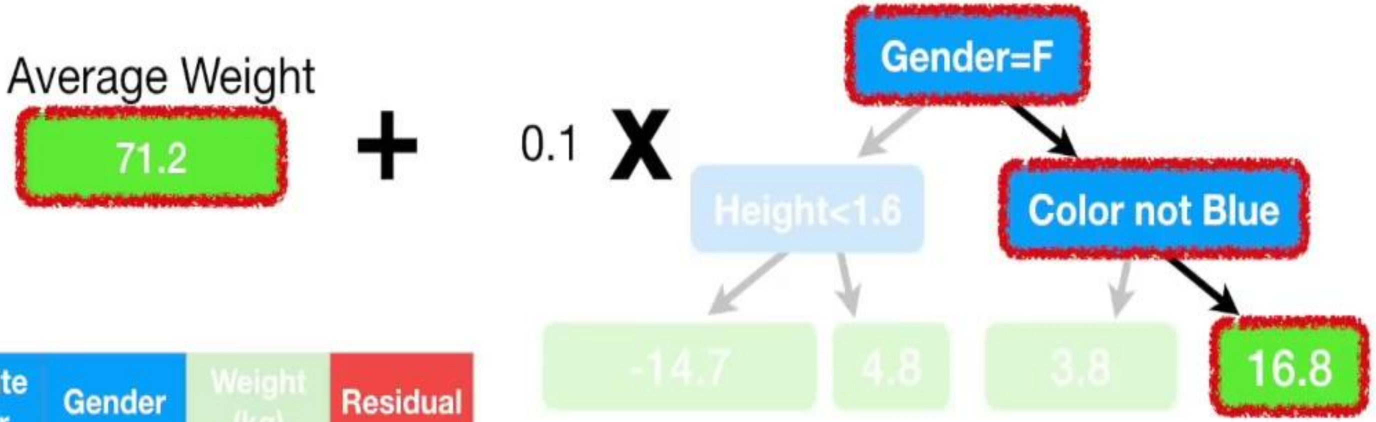
Example

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
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	

Just like before, we calculate the **Pseudo Residuals**, the difference between the **Observed Weights** and our latest **Predictions**.

← **Residual** = (**Observed** - **Predicted**)

Example



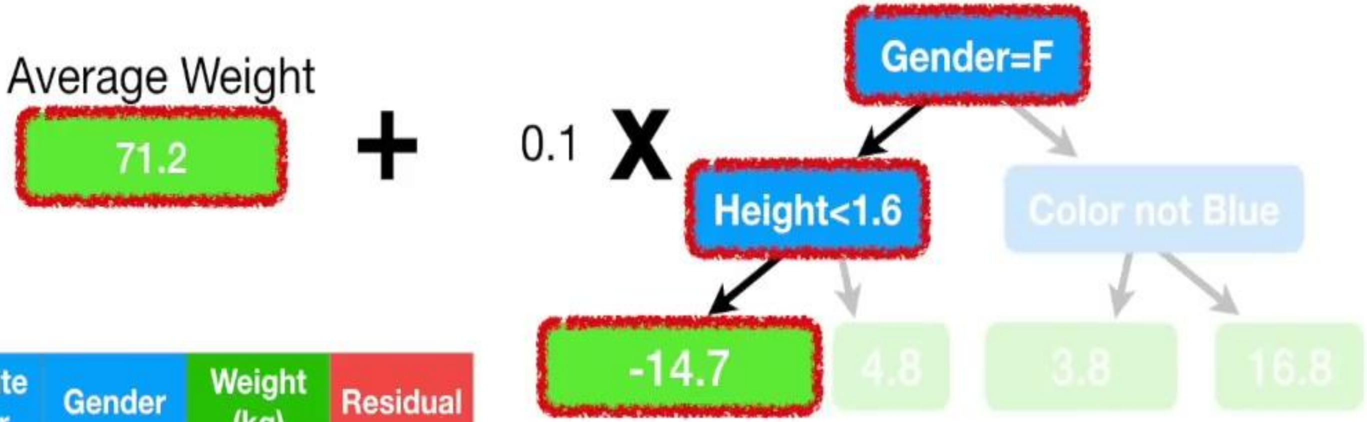
Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	
1.6	Green	Female	76	
1.5	Blue	Female	56	
1.8	Red	Male	73	
1.5	Green	Male	77	
1.4	Blue	Female	57	

Residual = (**88** - (71.2 + 0.1 x 16.8))

= 15.1

...and we save that in the column for **Pseudo Residuals**.

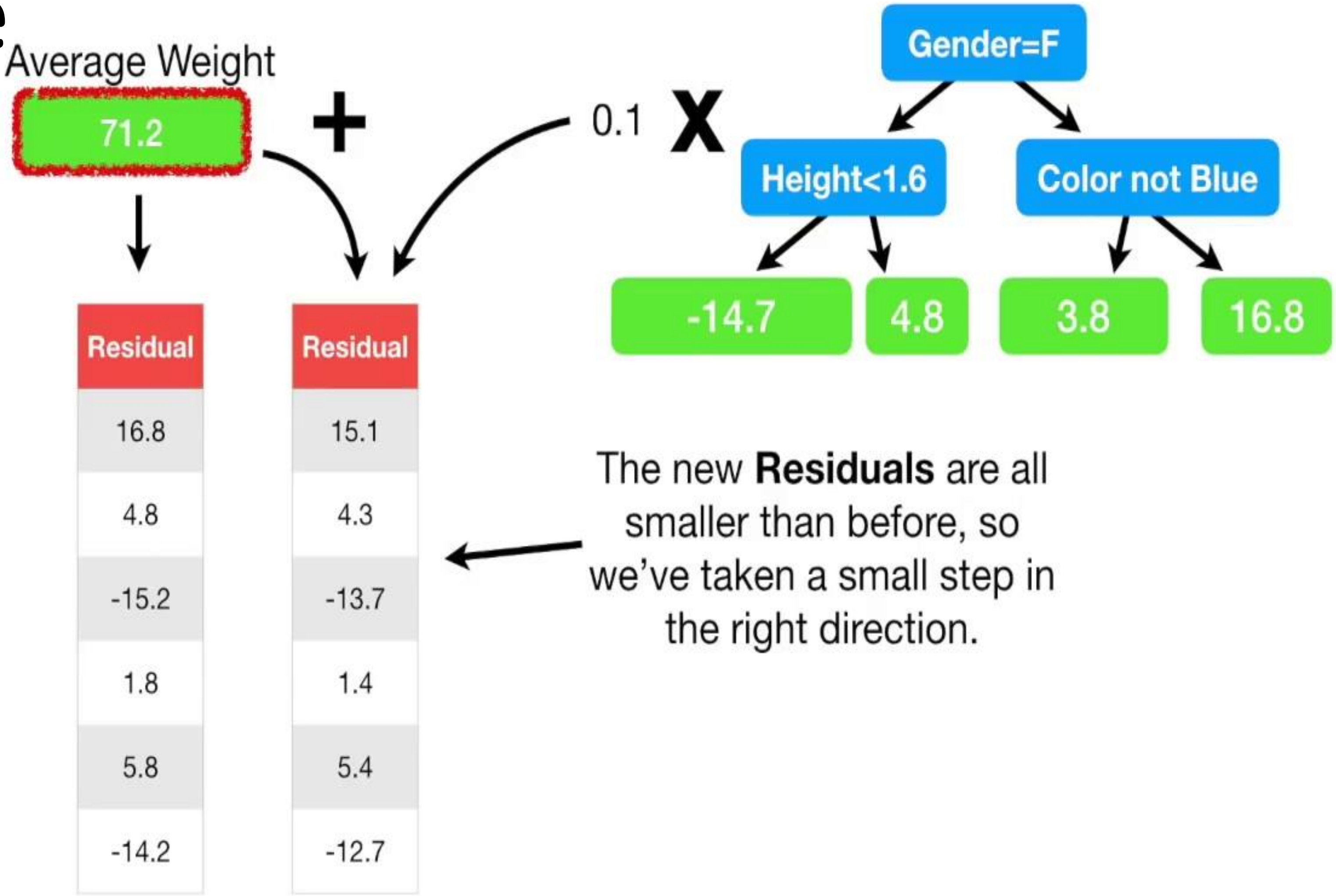
Example



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

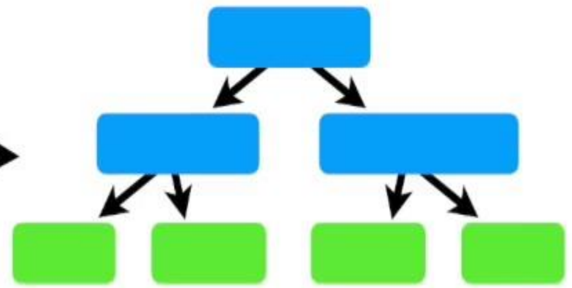
Residual = (Observed - Predicted)

Example


Average Weight
71.2
+
0.1
X
Gender=F
Height<1.6
Color not Blue
-14.7
4.8
3.8
16.8
Residual
16.8
4.8
-15.2
1.8
5.8
-14.2
Residual
15.1
4.3
-13.7
1.4
5.4
-12.7
The new **Residuals** are all smaller than before, so we've taken a small step in the right direction.

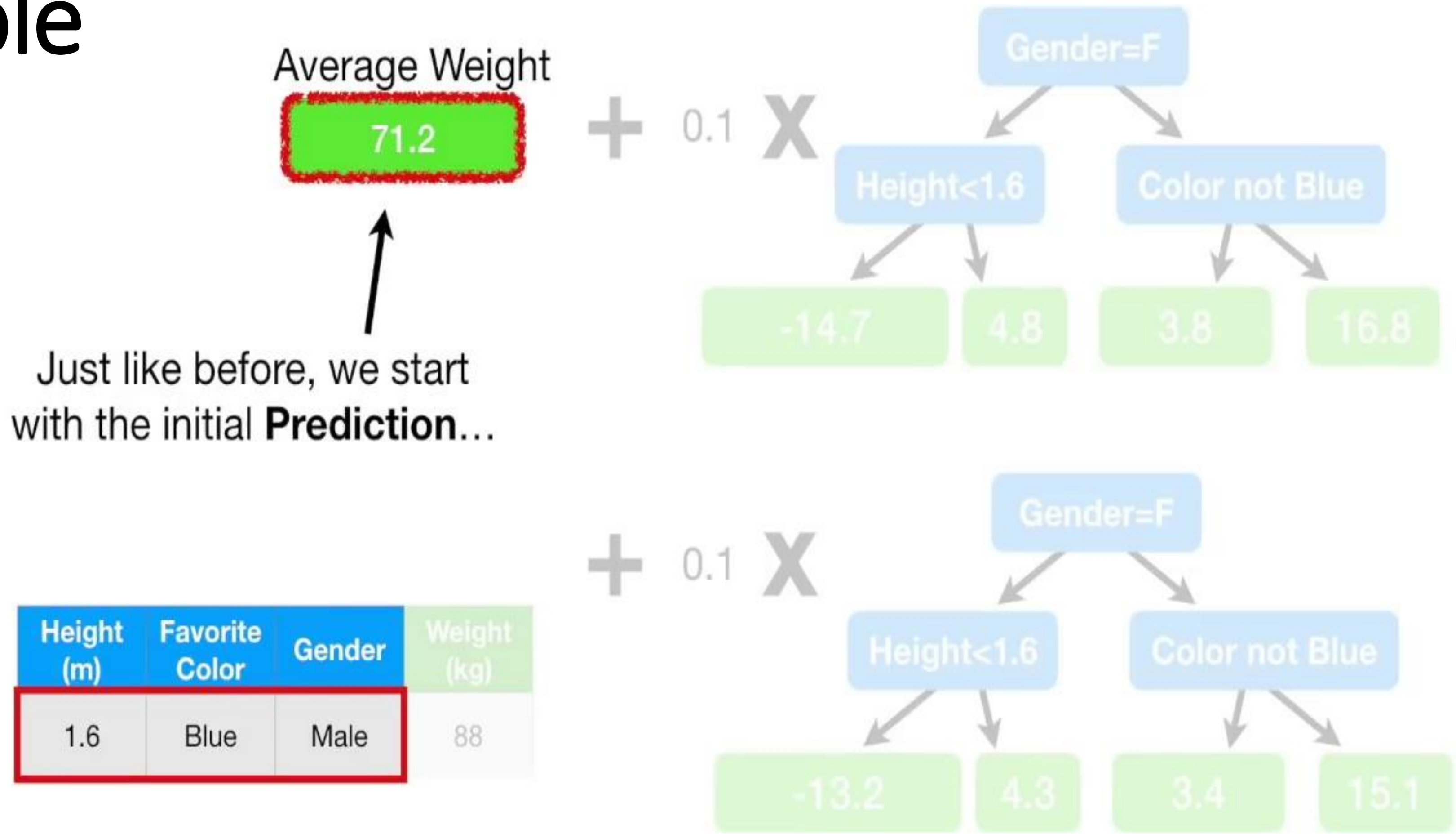
Example

Now let's build a new tree to predict the new **Residuals**.



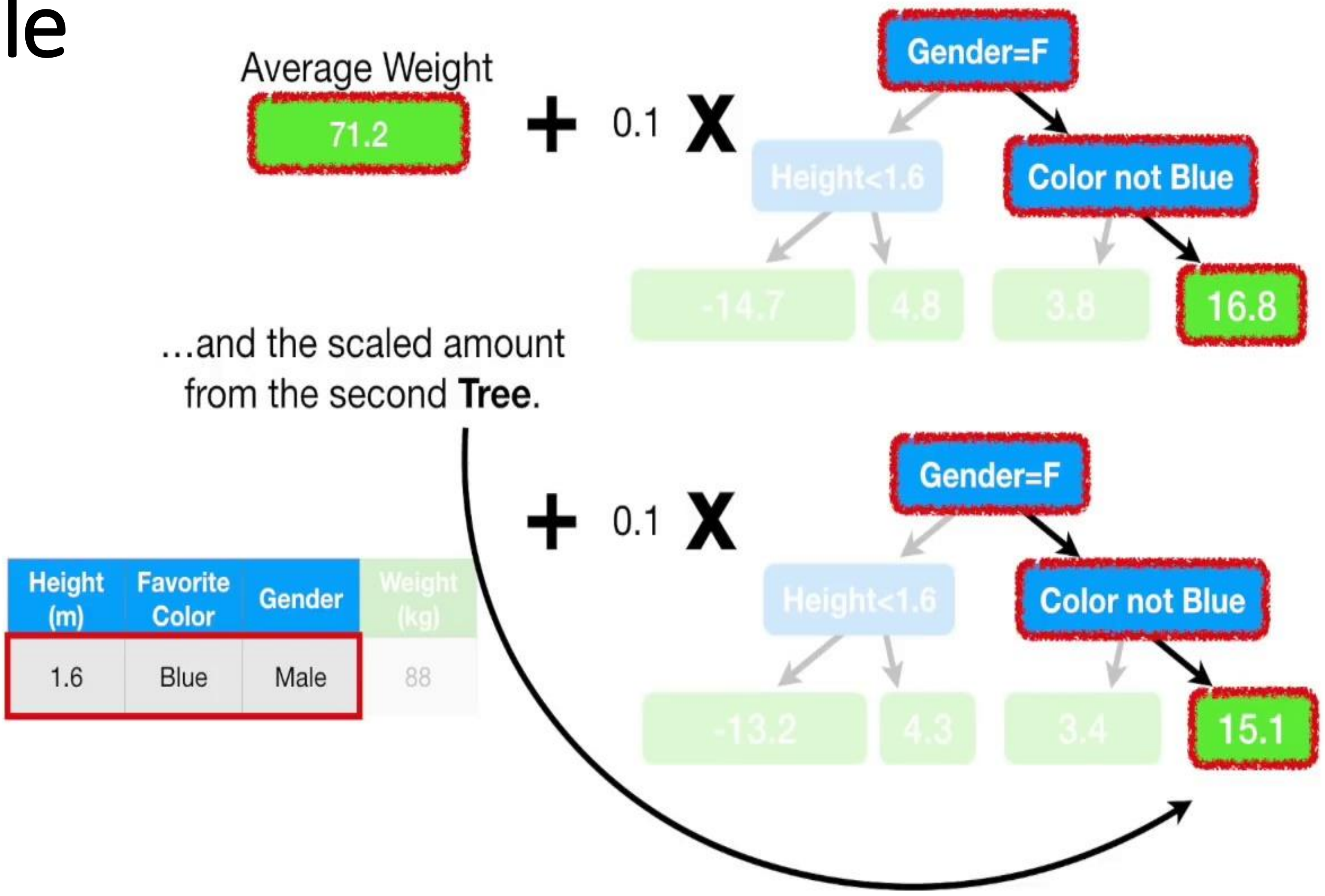
Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	15.1
1.6	Green	Female	76	4.3
1.5	Blue	Female	56	-13.7
1.8	Red	Male	73	1.4
1.5	Green	Male	77	5.4
1.4	Blue	Female	57	-12.7

Example

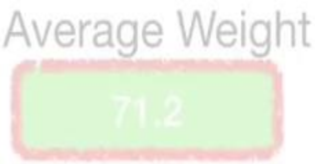


Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

Example


Average Weight
71.2
+ 0.1 X
Gender=F
Height<1.6
Color not Blue
-14.7
4.8
3.8
16.8
...and the scaled amount
from the second Tree.
+ 0.1 X
Gender=F
Height<1.6
Color not Blue
-13.2
4.3
3.4
15.1
Height (m)
Favorite Color
Gender
Weight (kg)
1.6
Blue
Male
88

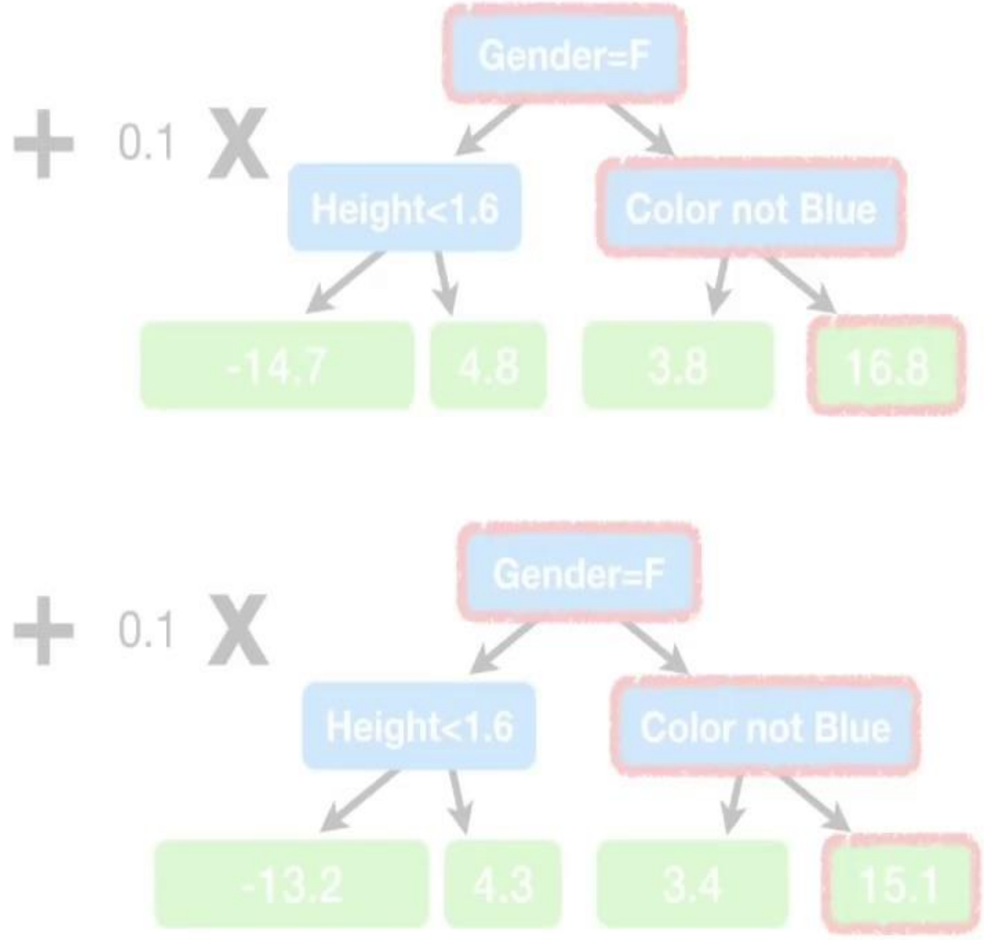
Example



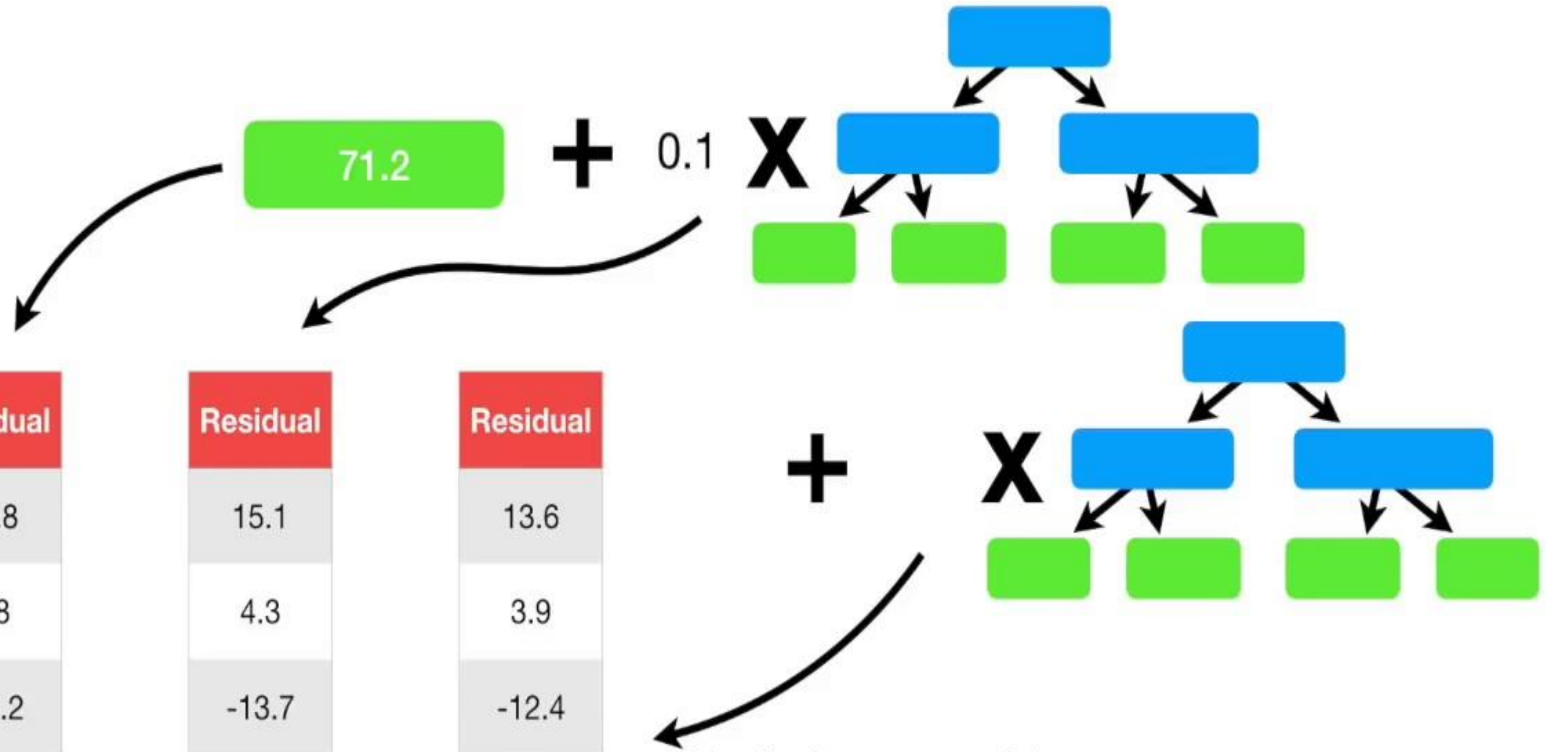
That gives us...

71.2 + (0.1 x 16.8) + (0.1 x 15.1)

= 74.4



Example



Residual	Residual	Residual
16.8	15.1	13.6
4.8	4.3	3.9
-15.2	-13.7	-12.4
1.8	1.4	1.1
5.8	5.4	5.1
-14.2	-12.7	-11.4

Each time we add a tree to the **Prediction**, the **Residuals** get smaller.

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: for m = 1 to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,...,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Step 3: Output $F_M(x)$

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$ and a differentiable **Loss Function** $L(y_i, F(x))$

The **Loss Function** that is most commonly used when doing **Regression** with **Gradient Boost** is…

$$\frac{1}{2}(\textbf{Observed} - \textbf{Predicted})^2$$

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

The reason why people choose this **Loss Function** for **Gradient Boost** is that when we differentiate it with respect to "**Predicted**"...

$$\frac{d}{d\,Predicted}\frac{1}{2}(\textbf{Observed} - \textbf{Predicted})^2$$

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

...then the **2/2** cancels out...

$$\frac{d}{d\,Predicted}\ \frac{1}{2}(\textbf{Observed - Predicted})^2$$

$$= \frac{2}{2}(\textbf{Observed - Predicted}) \times -1$$

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

...and that leaves you with the **Observed** minus the **Predicted** multiplied by **-1**.

$$\frac{d}{d\,Predicted}\frac{1}{2}(\text{Observed} - \text{Predicted})^2$$

$$= \frac{2}{2}(\text{Observed} - \text{Predicted}) \times -1$$

$$= -(\text{Observed} - \text{Predicted})$$

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

In other words, we are left with the *negative* **Residual**, and this makes the math easier since **Gradient Boost** uses the derivative a lot.

$$\frac{d}{d\,Predicted}\frac{1}{2}(\textbf{Observed} - \textbf{Predicted})^2$$

$$= \frac{2}{2}(\textbf{Observed} - \textbf{Predicted}) \times -1$$

$$= -(\textbf{Observed} - \textbf{Predicted})$$

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

The $\boldsymbol{y_i}$'s are the **Observed** values…

$$\frac{1}{2}(\textbf{Observed} - \textbf{Predicted})^2$$

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

...and ***F(x)*** is a function that gives us the **Predicted** values.

$$\frac{1}{2}(\textbf{Observed} - \textbf{Predicted})^2$$

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

We start by initializing the model with a constant value...

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

$$\frac{1}{2}(88 - \textbf{Predicted})^2 + \frac{1}{2}(76 - \textbf{Predicted})^2 + \frac{1}{2}(56 - \textbf{Predicted})^2$$

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56

The summation means that we add up one **Loss Function** for each **Observed** value…

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

$$\frac{1}{2}(88 - \textbf{Predicted})^2 + \frac{1}{2}(76 - \textbf{Predicted})^2 + \frac{1}{2}(56 - \textbf{Predicted})^2$$

...and the "**argmin over gamma**" means we need to find a **Predicted** value that minimizes this sum.

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56

GB Algorithm

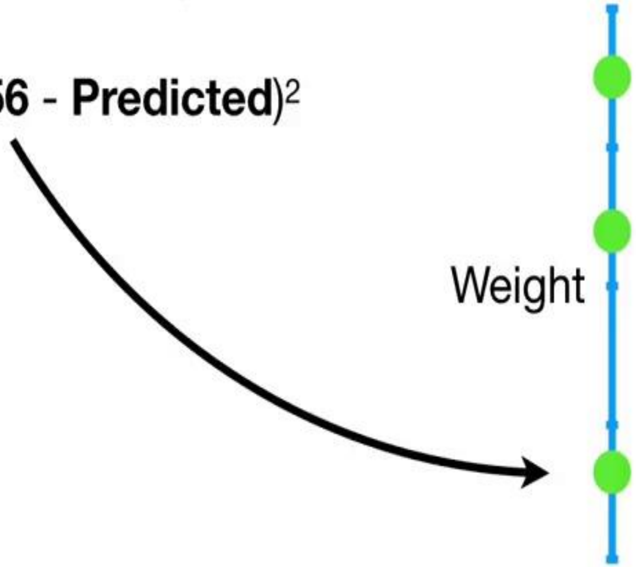
Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

$\frac{1}{2}$(88 - **Predicted**)^2 +

$\frac{1}{2}$(76 - **Predicted**)^2 +

$\frac{1}{2}$(56 - **Predicted**)^2



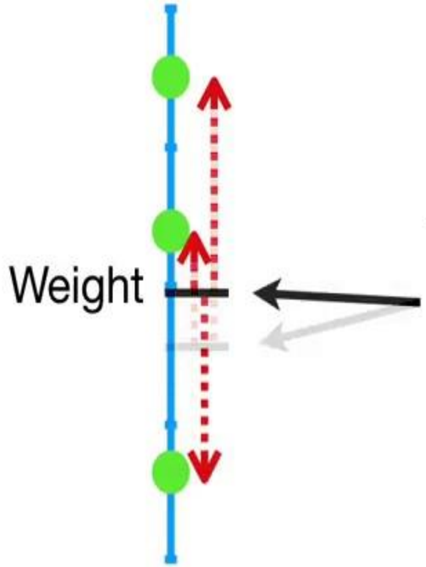
In other words, if we plot the **Observed Weights** on a number line…

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

$$\frac{1}{2}(88 - \textbf{Predicted})^2 + \frac{1}{2}(76 - \textbf{Predicted})^2 + \frac{1}{2}(56 - \textbf{Predicted})^2$$



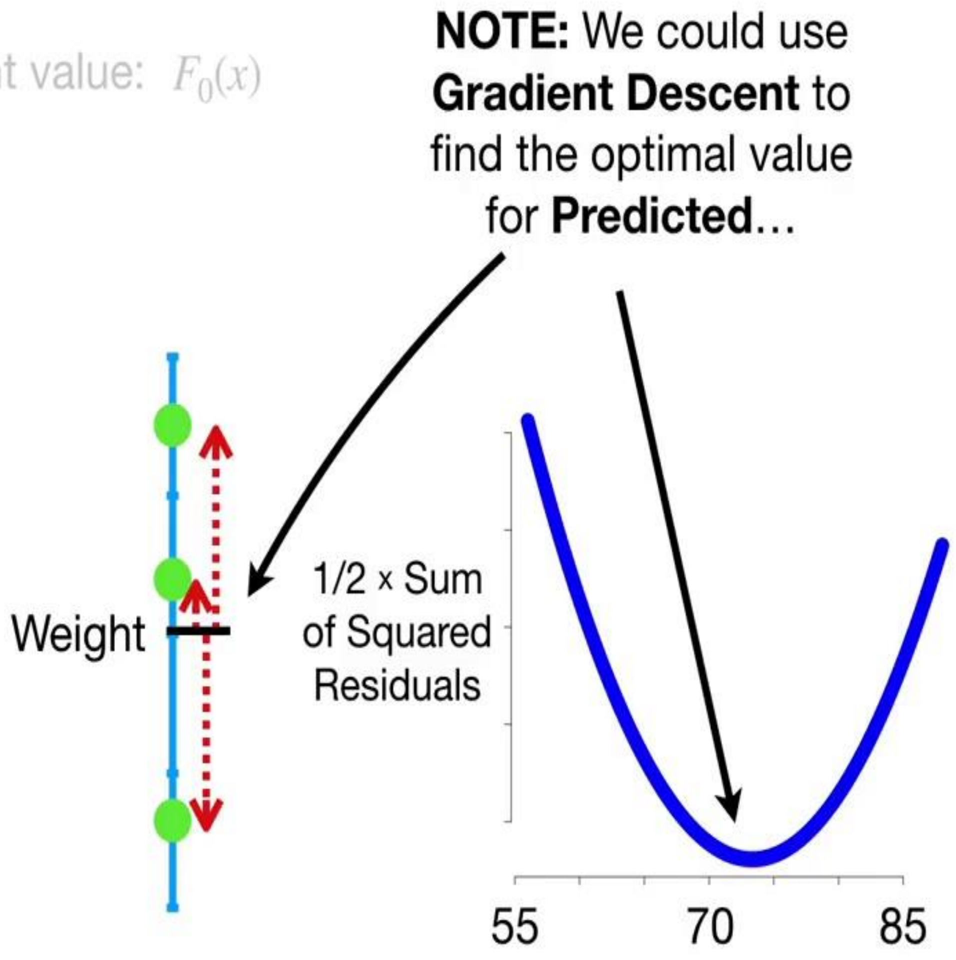
...then we want to find the point on the line that minimizes the sum of the squared residuals...

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Fu**

Step 1: Initialize model with a constant value: $F_0(x)$

$$\frac{1}{2}(88 - \textbf{Predicted})^2 + \frac{1}{2}(76 - \textbf{Predicted})^2 + \frac{1}{2}(56 - \textbf{Predicted})^2$$



GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56

...and we end up with the **Average** of the **Observed Weights.**

$$\textbf{Predicted} = \frac{88 + 76 + 56}{3}$$

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

We have now created the initial predicted value, $\boldsymbol{F_0(x)}$…

$$F_0(x) = \frac{88 + 76 + 56}{3}$$

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

...and it equals **73.3**.

$$F_0(x) = \frac{88 + 76 + 56}{3}$$

$$= 73.3$$

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

That means that the initial predicted value, ***$F_0(x)$***, is just a leaf.

↓

73.3

$$F_0(x) = \frac{88 + 76 + 56}{3}$$

$$= 73.3$$

GB Algorithm

Now we can work on **Step 2**…

Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1\ldots J_m$

(C) For $j = 1\ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_m I(x \in R_{jm})$

GB Algorithm

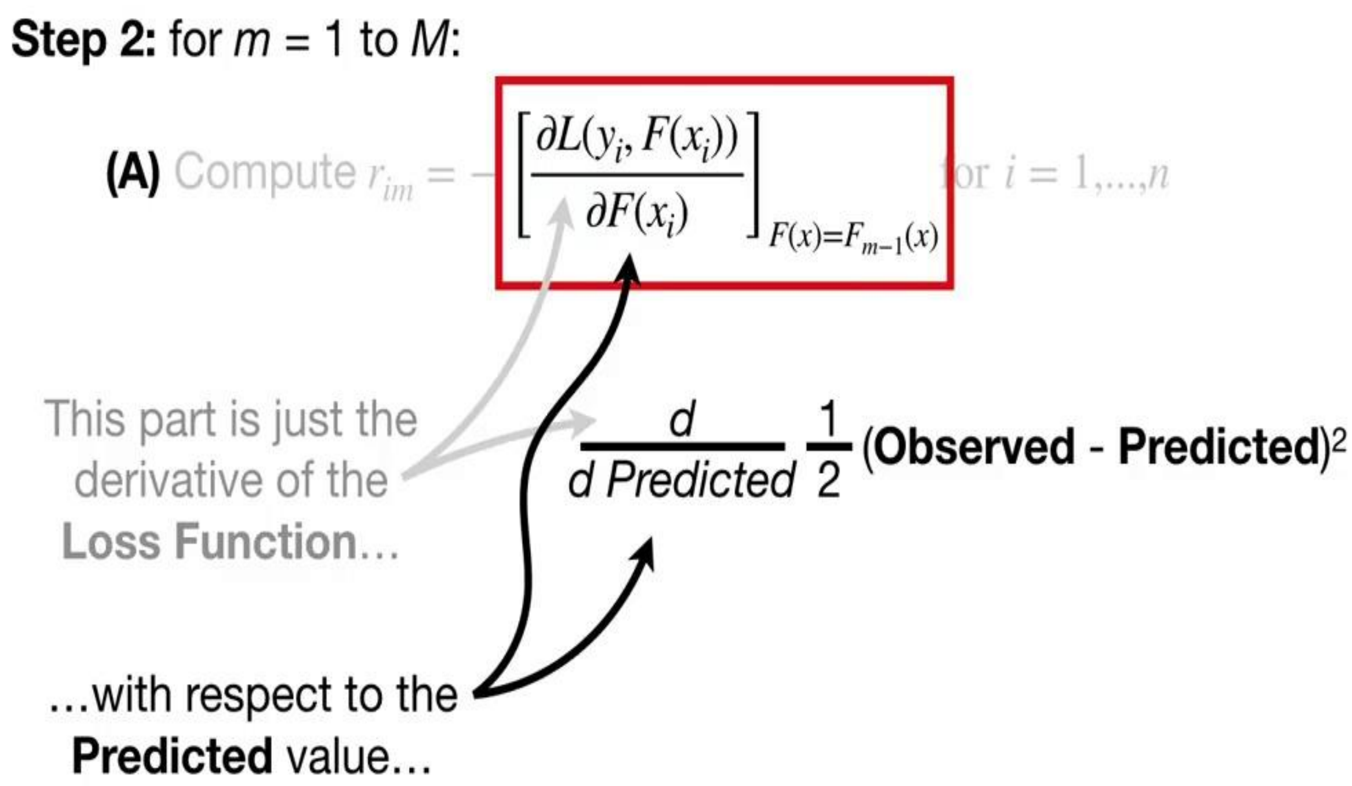
Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

This part is just the derivative of the **Loss Function…**

$$\frac{d}{d\,Predicted}\,\frac{1}{2}\,(\textbf{Observed - Predicted})^2$$

GB Algorithm



GB Algorithm

Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

$$\frac{d}{d\ Predicted}\frac{1}{2}(\textbf{Observed - Predicted})^2$$

…and we've already calculated this.

$$= -(\textbf{Observed - Predicted})$$

GB Algorithm

Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,...,n$

Height (m)	Favorite Color	Gender	Weight (kg)	$r_{i,1}$
1.6	Blue	Male	88	14.7
1.6	Green	Female	76	2.7
1.5	Blue	Female	56	-17.3

We've finished **Part A** of **Step 2** by calculating a **Residual** for each sample.

GB Algorithm

Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,...,n$

NOTE: Before we move on, I just want to point out that this derivative is the **Gradient** that **Gradient Boost** is named after.

GB Algorithm

Step 2: for m = 1 to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,...,n$

I also want to point that the $\boldsymbol{r_{i,m}}$ values are technically called **Pseudo Residuals**.

GB Algorithm

Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1\ldots J_m$

↑

All this is saying is that we will build a regression tree…

GB Algorithm

Now let's do **Part C.**

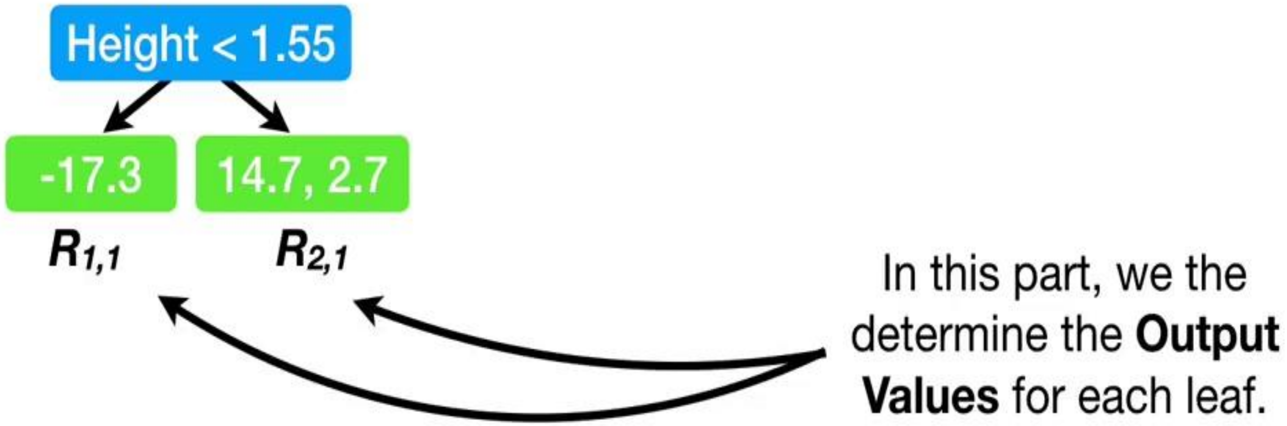
Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1\ldots J_m$

(C) For $j = 1\ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

GB Algorithm



(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

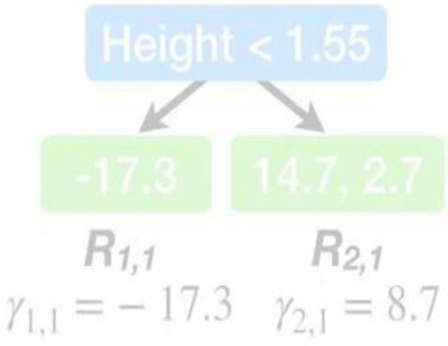
GB Algorithm

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

NOTE: This minimization is like what we did in **Step 1**.

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

GB Algorithm



Given our choice of **Loss Function**, the **Output Values** are *always* the average of the **Residuals** that end up in the same leaf.

$$\frac{1}{2}(\textbf{Observed - Predicted})^2$$

GB Algorithm

Now let's do **Part D**!!!

Step 2: for $m = 1$ to M:

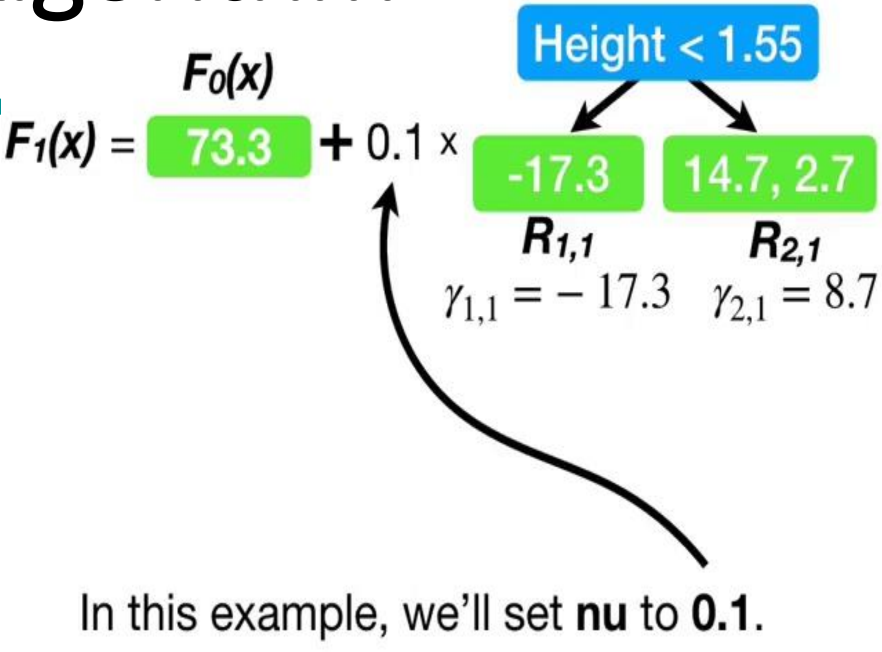
(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1\ldots J_m$

(C) For $j = 1\ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

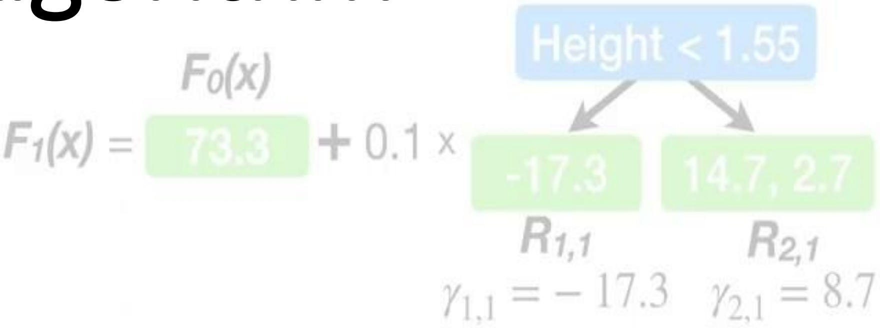
(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

GB Algorithm



(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

GB Algorithm



Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56

Now we will use ***$F_1(x)$*** to make new **Predictions** for each sample.

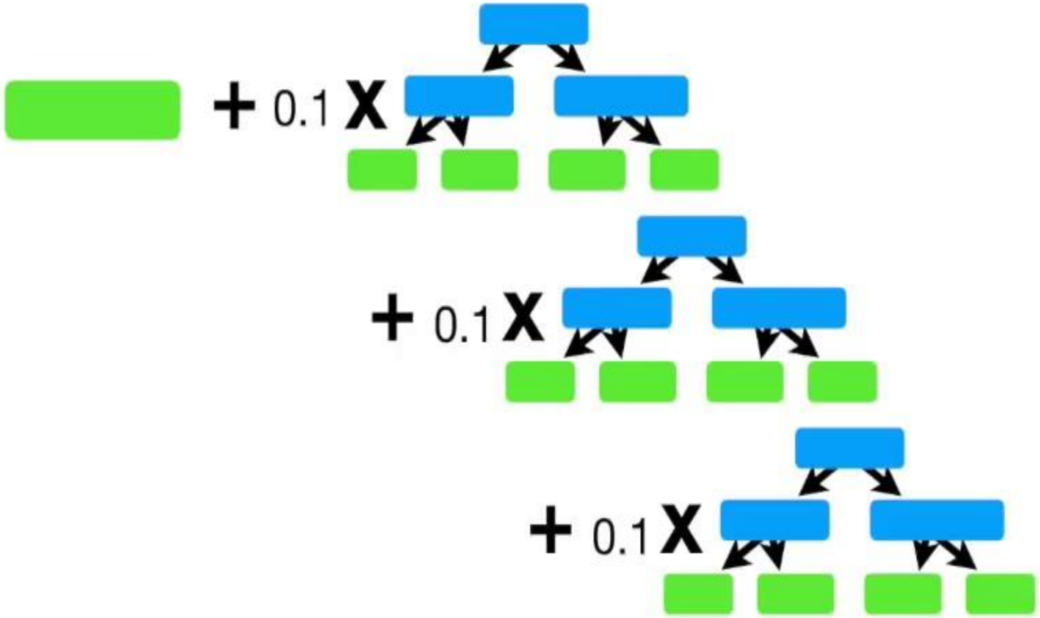
(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Gradient Boosting for Classification

Example

...and walk through, step-by-step, the most common way that **Gradient Boost** fits a model to this **Training Data**.

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes



Example

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

When we use **Gradient Boost for Classification**, the initial **Prediction** for every individual is the **log(odds)**.

Example

$\log(4/2) = 0.7$ ← So this is the **Initial Prediction**.

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

Example

log(4/2) = 0.693

Just like with **Logistic Regression**, the easiest way to use the **log(odds)** for **Classification** is to convert it to a **Probability**…

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

Example

log(4/2) = 0.693

Just like with **Logistic Regression**, the easiest way to use the **log(odds)** for **Classification** is to convert it to a **Probability**…

…and we do that with a **Logistic Function**.

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

Example

log(4/2) = 0.693

So we plug the **log(odds)** into the **Logistic Function…**

$$\text{Probability of Loving Troll 2} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

Example

$\log(4/2) = 0.693$

...and we get **0.7** as the **Probability of Loving Troll 2.**

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

$$\textbf{Probability of Loving Troll 2} = \frac{e^{\log(4/2)}}{1 + e^{\log(4/2)}} = \boxed{0.7}\quad 0.666$$

Example

log(4/2) = 0.693

NOTE this are rounded values

...and we get **0.7** as the **Probability of Loving Troll 2.**

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

$$\text{Probability of Loving Troll 2} = \frac{e^{\log(4/2)}}{1 + e^{\log(4/2)}} = \boxed{0.7}\quad 0.666$$

Example

log(4/2) = 0.7

Probability of Loving Troll 2 = 0.7 0.666

Since the **Probability** of **Loving Troll 2** is greater than **0.5**, we can **Classify** everyone in the **Training Dataset** as someone who **Loves Troll 2**.

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

Example

log(4/2) = 0.693

Probability of **Loving Troll 2** = 0.7 0.666

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

We can measure how bad the initial **Prediction** is by calculating **Pseudo Residuals**, the difference between the **Observed** and the **Predicted** values.

Residual = (**Observed** - **Predicted**)

Example

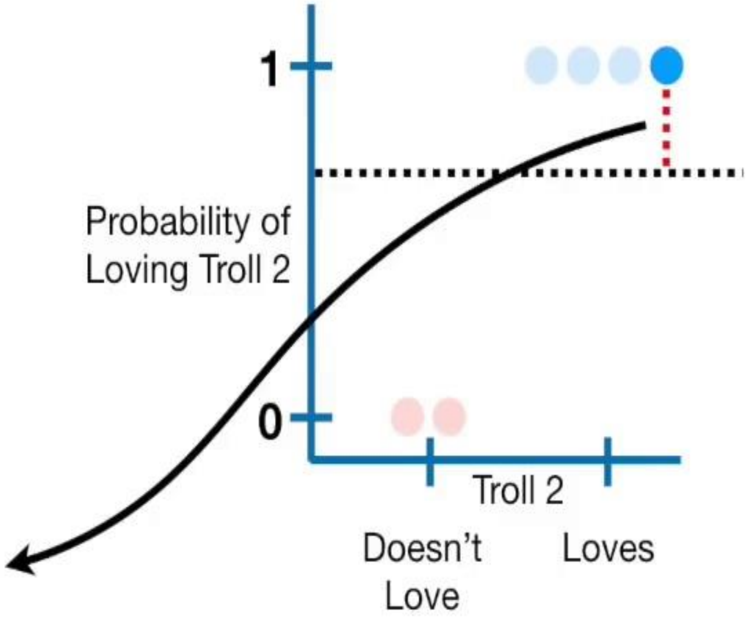
log(4/2) = 0.7

Probability of Loving Troll 2 = 0.7

Then we calculate the rest of the **Residuals**…

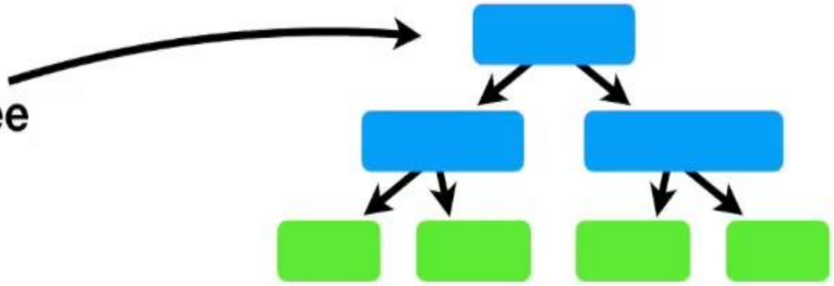
Residual = (**Observed** - **Predicted**)

Likes Popcorn	Age	Favorite Color	Loves Troll 2	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3



Example

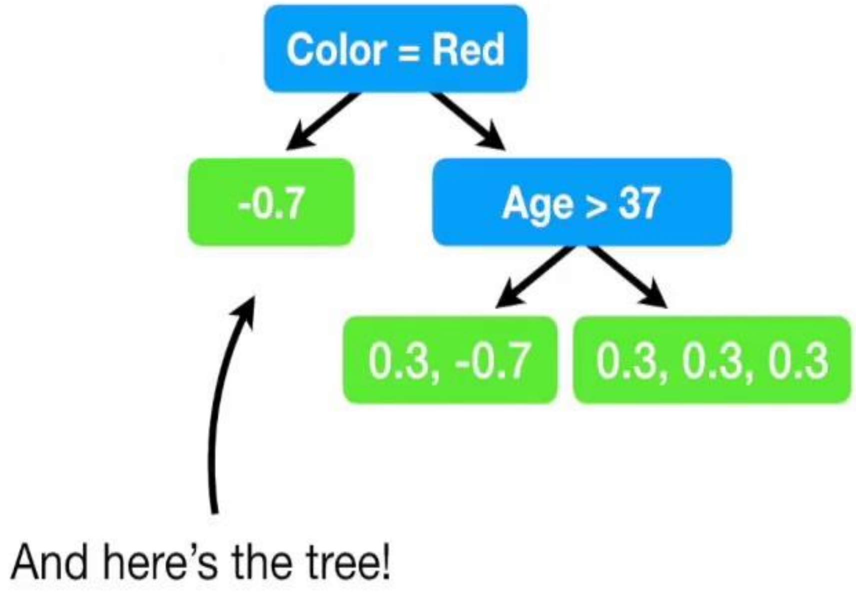
Now we will build a **Tree**



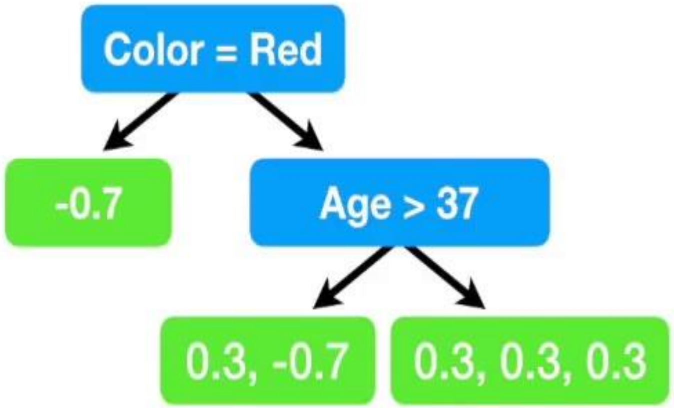
Likes Popcorn	Age	Favorite Color	Loves Troll 2	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3

Example

Likes Popcorn	Age	Favorite Color	Loves Troll 2	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3



Example



Likes Popcorn	Age	Favorite Color	Loves Troll 2	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3

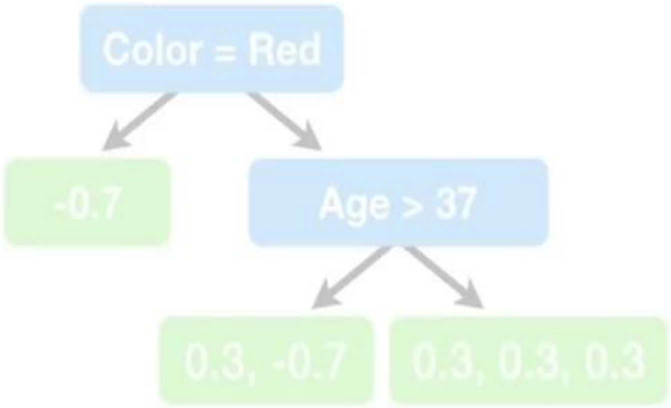
In this simple example, we are limiting the number of leaves to **3**.

In practice people often set the maximum number of leaves to be between **8** and **32**

Example

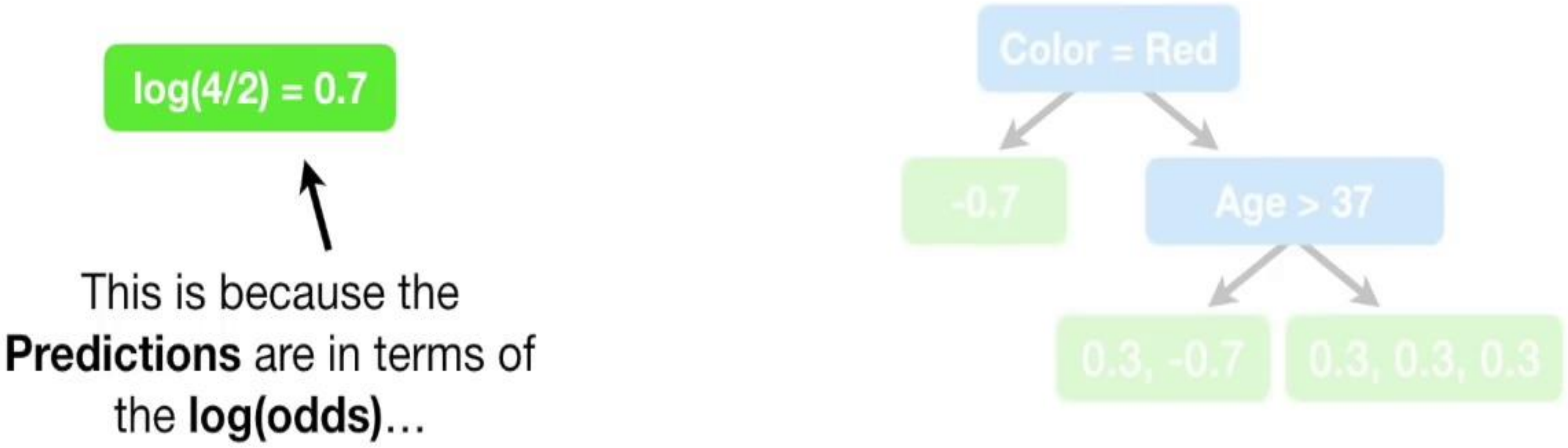
When we used **Gradient Boost** for **Regression**, a leaf with single **Residual** had an **Output Value** equal to that **Residual**.

Example

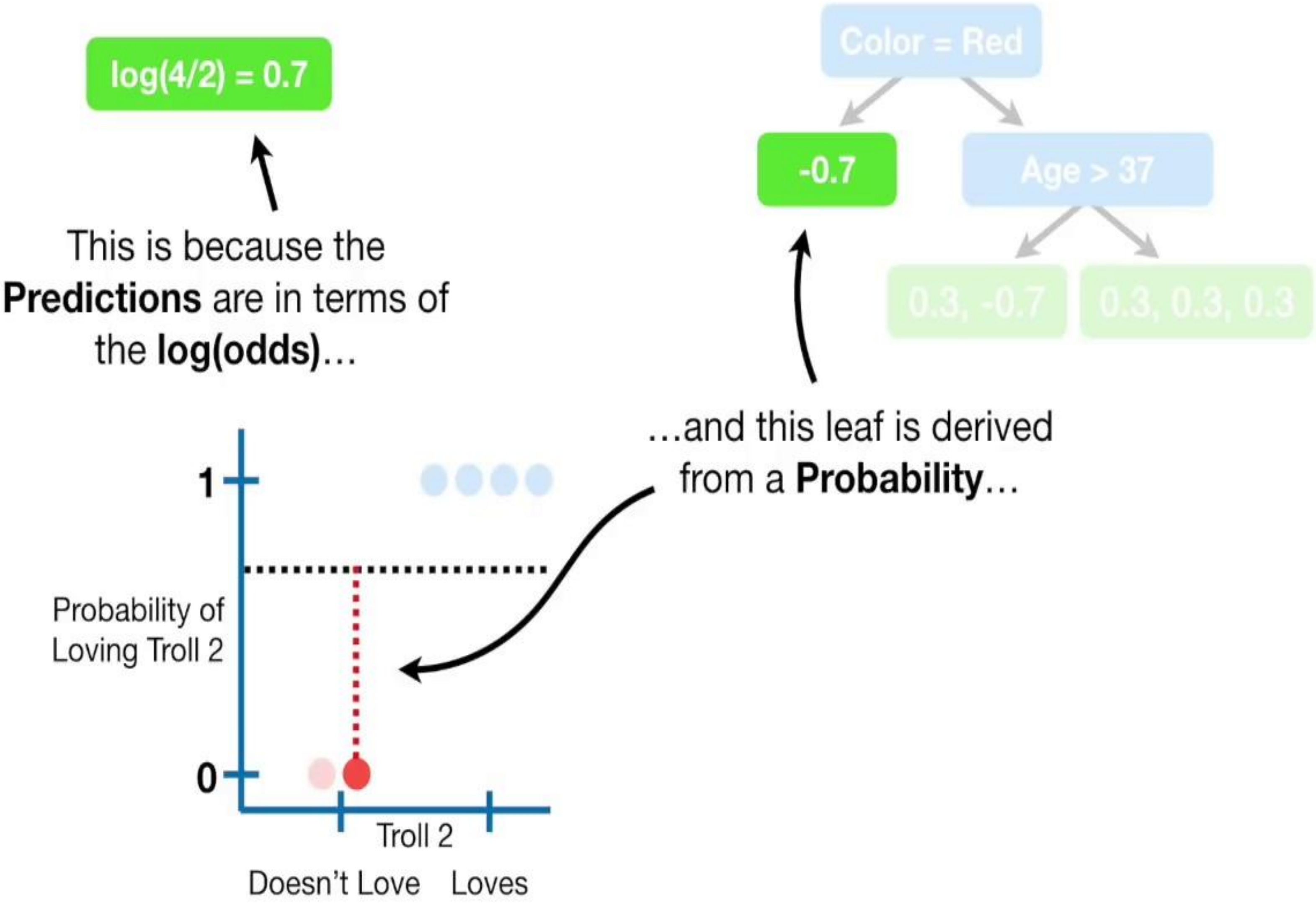


In contrast, when we use **Gradient Boost** for **Classification**, the situation is a little more complex.

Example


log(4/2) = 0.7
This is because the **Predictions** are in terms of the **log(odds)**…
Color = Red
-0.7
Age > 37
0.3, -0.7
0.3, 0.3, 0.3

Example


log(4/2) = 0.7
This is because the **Predictions** are in terms of the **log(odds)**…
Color = Red
-0.7
Age > 37
0.3, -0.7
0.3, 0.3, 0.3
…and this leaf is derived from a **Probability**…
1
0
Probability of Loving Troll 2
Troll 2
Doesn't Love
Loves

Example

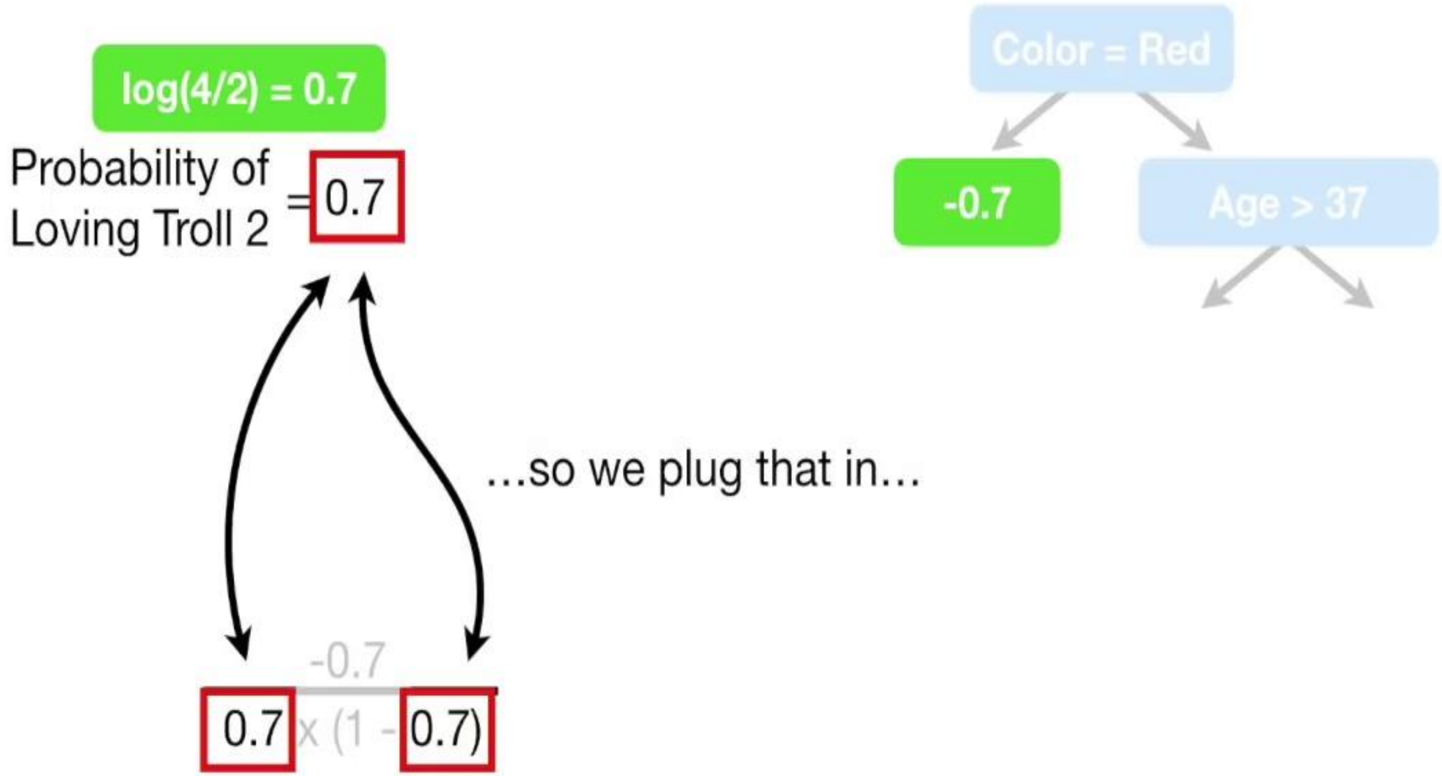
…so we can't just add them together to get a new **log(odds) Prediction** without some sort of transformation.

Example

When we use **Gradient Boost** for **Classification**, the most common transformation is the following formula.

$$\frac{\sum \text{Residual}_i}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right]}$$

Example



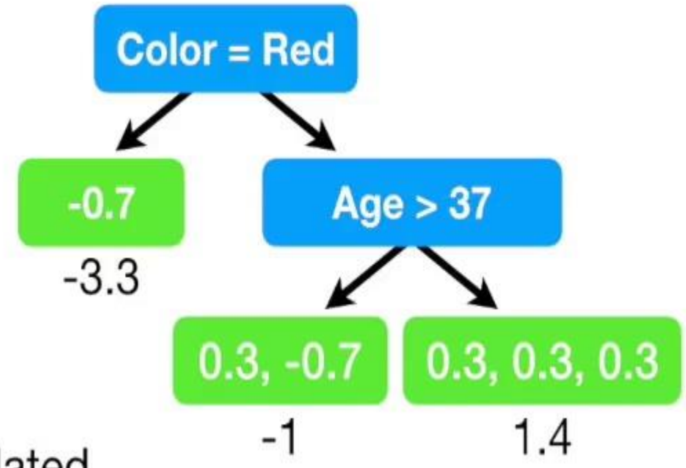
log(4/2) = 0.7
Probability of Loving Troll 2 = 0.7
...so we plug that in...
-0.7
0.7 x (1 - 0.7)
Color = Red
-0.7
Age > 37

Example

...and we end up with -3.3 as the new **Output Value** for this leaf.

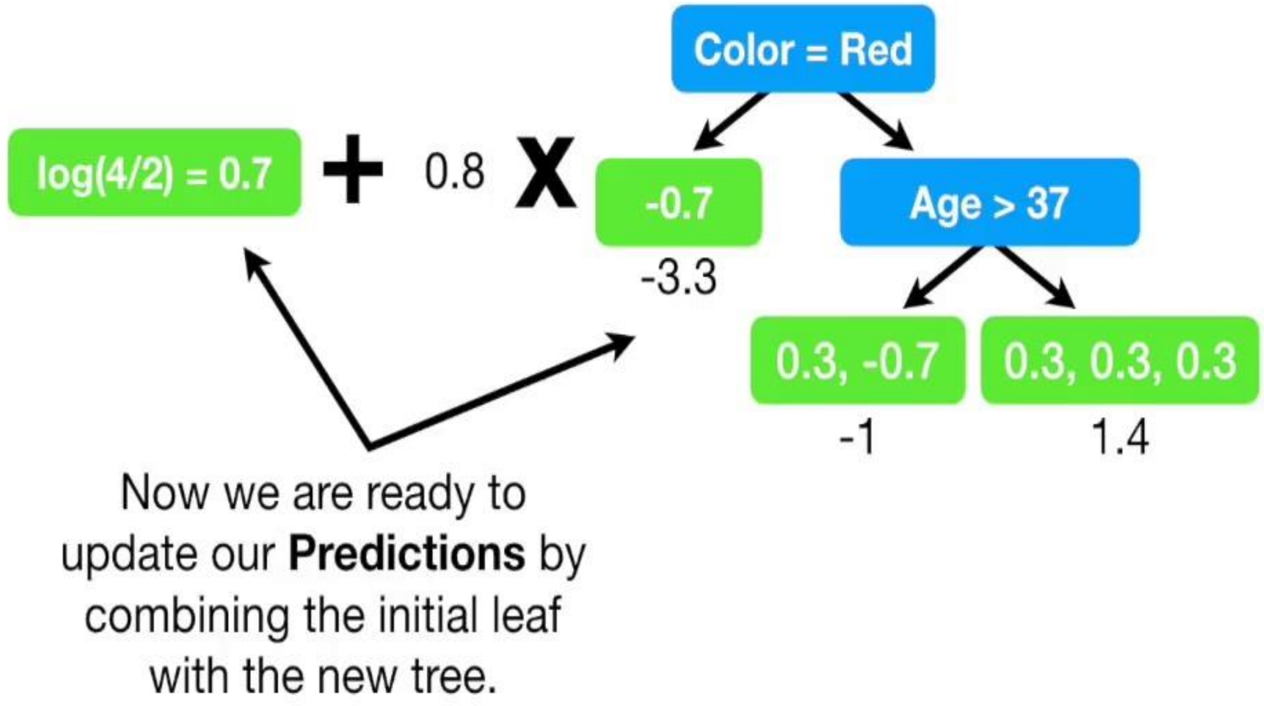
$$\frac{-0.7}{0.7 \times (1 - 0.7)} = \boxed{-3.3}$$

Example



We've calculated **Output Values** for all three leaves in the tree!

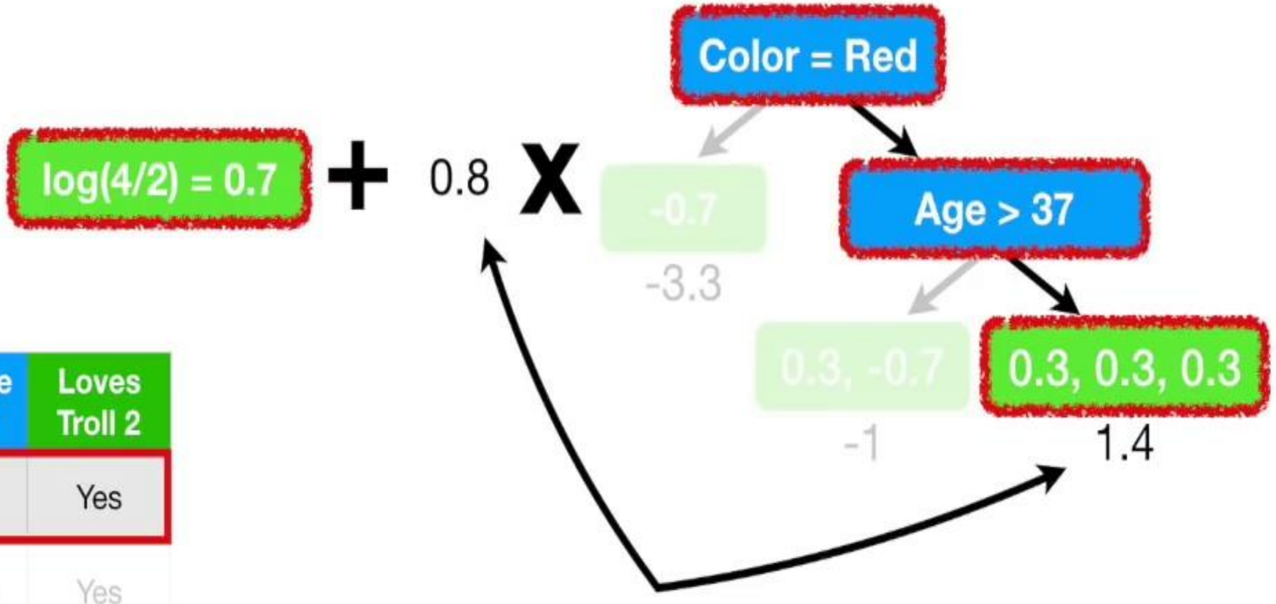
Example



log(4/2) = 0.7
+
0.8
X
Color = Red
-0.7
-3.3
Age > 37
0.3, -0.7
-1
0.3, 0.3, 0.3
1.4
Now we are ready to update our **Predictions** by combining the initial leaf with the new tree.

Example

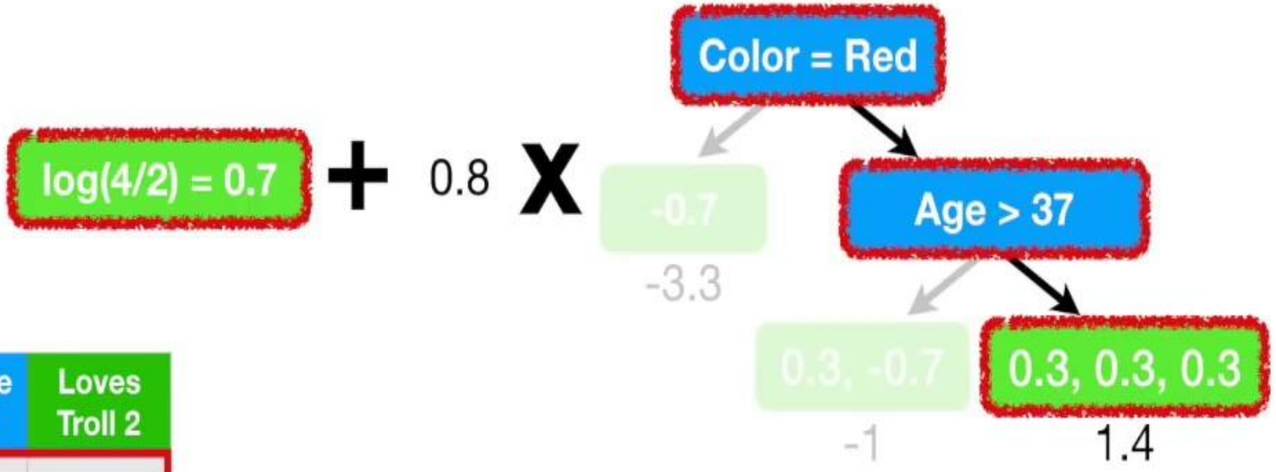
Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes



...plus the **Output Value** from the tree scaled by the **Learning Rate**...

$$\text{log(odds) Prediction} = 0.7 + (0.8 \times 1.4)$$

Example



Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

…and the new **log(odds) Prediction** = **1.8**.

log(odds) Prediction = 0.7 + (0.8 × 1.4) = 1.8

Example

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

Now we convert the new **log(odds) Prediction** into a **Probability**…

$$\text{Probability} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

$$\log(\text{odds})\ \text{Prediction} = 0.7 + (0.8 \times 1.4) = 1.8$$

Example

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

Now we convert the new **log(odds) Prediction** into a **Probability**…

$$\text{Probability} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

$$\log(\text{odds})\ \text{Prediction} = 0.7 + (0.8 \times 1.4) = 1.8$$

Example

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

Now we convert the new **log(odds) Prediction** into a **Probability**…

$$\text{Probability} = \frac{e^{1.8}}{1 + e^{1.8}} = 0.9$$

$$\text{log(odds) Prediction} = 0.7 + (0.8 \times 1.4) = 1.8$$

Example

log(4/2) = 0.7

Initial Probability of **Loving Troll 2** = 0.7

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

…so we are taking a small step in the right direction since this person **Loves Troll 2**.

$$\text{Probability} = \frac{e^{1.8}}{1 + e^{1.8}} = 0.9$$

$$\text{log(odds) Prediction} = 0.7 + (0.8 \times 1.4) = 1.8$$

Example

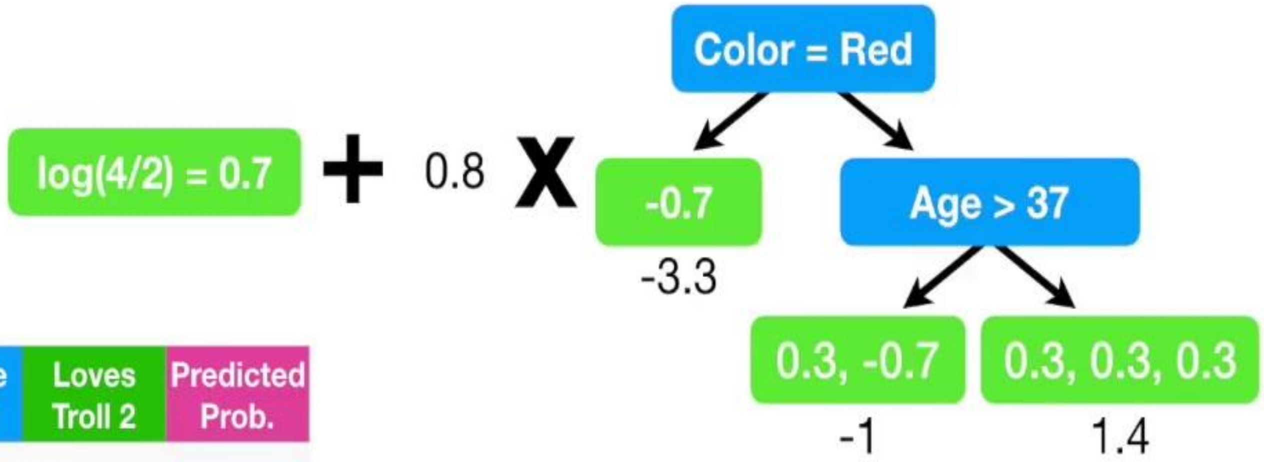
Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.
Yes	12	Blue	Yes	0.9
Yes	87	Green	Yes	
No	44	Blue	No	
Yes	19	Red	No	
No	32	Green	Yes	
No	14	Blue	Yes	

We save the new **Predicted Probability** here.

$$\text{Probability} = \frac{e^{1.8}}{1 + e^{1.8}} = 0.9$$

log(odds) Prediction = 0.7 + (0.8 × 1.4) = 1.8

Example



Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.
Yes	12	Blue	Yes	0.9
Yes	87	Green	Yes	0.5
No	44	Blue	No	0.5
Yes	19	Red	No	0.1
No	32	Green	Yes	0.9
No	14	Blue	Yes	0.9

Then we calculate the **Predicted Probabilities** for the remaining people.

Example

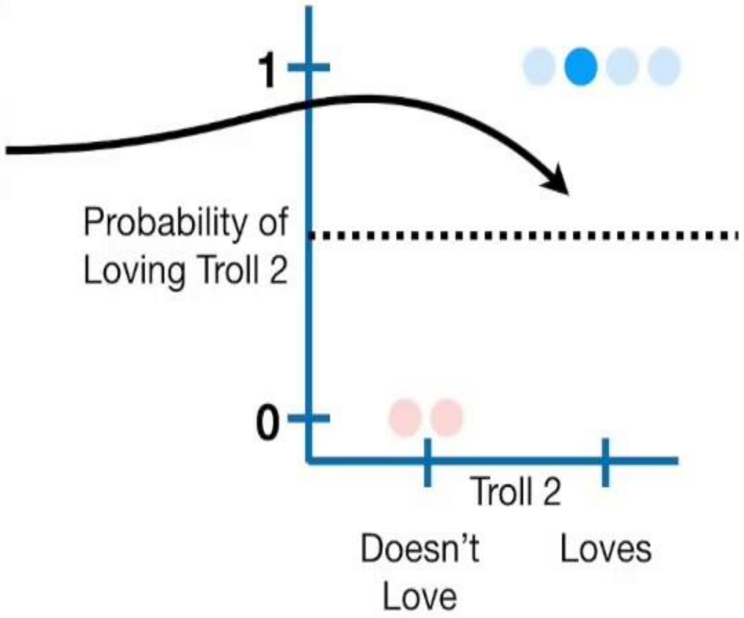
And now, just like before, we calculate the new **Residuals**…

Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.	Residual
Yes	12	Blue	Yes	0.9	
Yes	87	Green	Yes	0.5	
No	44	Blue	No	0.5	
Yes	19	Red	No	0.1	
No	32	Green	Yes	0.9	
No	14	Blue	Yes	0.9	

Example

Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.	Residual
Yes	12	Blue	Yes	0.9	0.1
Yes	87	Green	Yes	0.5	
No	44	Blue	No	0.5	
Yes	19	Red	No	0.1	
No	32	Green	Yes	0.9	
No	14	Blue	Yes	0.9	

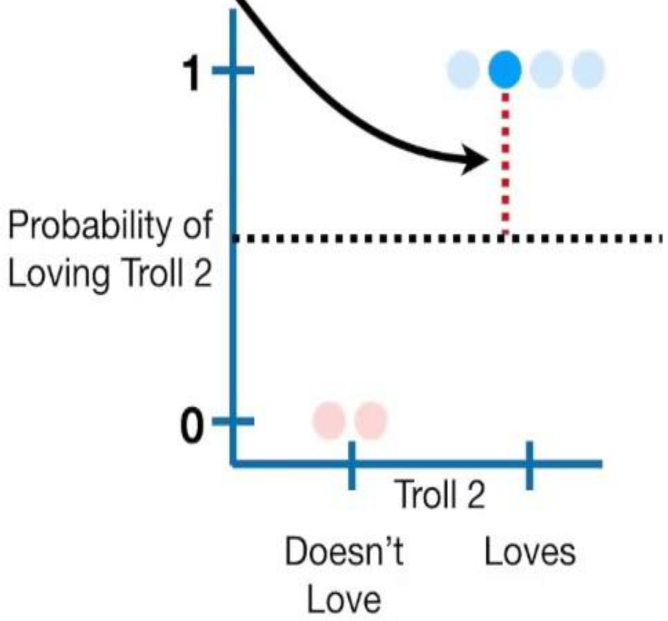
We plot the **Predicted Probability**…



Example

...and the **Residual** is the difference. → **Residual** = (**Observed** - **Predicted**)

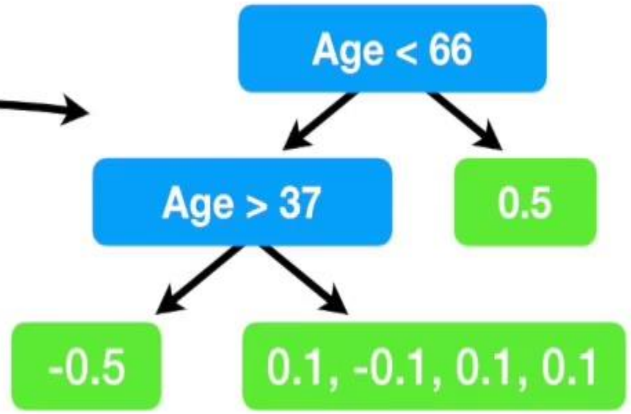
Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.	Residual
Yes	12	Blue	Yes	0.9	0.1
Yes	87	Green	Yes	0.5	
No	44	Blue	No	0.5	
Yes	19	Red	No	0.1	
No	32	Green	Yes	0.9	
No	14	Blue	Yes	0.9	



Example

Now that we have the **Residuals**, we can build a new tree…

Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.	Residual
Yes	12	Blue	Yes	0.9	0.1
Yes	87	Green	Yes	0.5	0.5
No	44	Blue	No	0.5	-0.5
Yes	19	Red	No	0.1	-0.1
No	32	Green	Yes	0.9	0.1
No	14	Blue	Yes	0.9	0.1



GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Step 3: Output $F_M(x)$

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

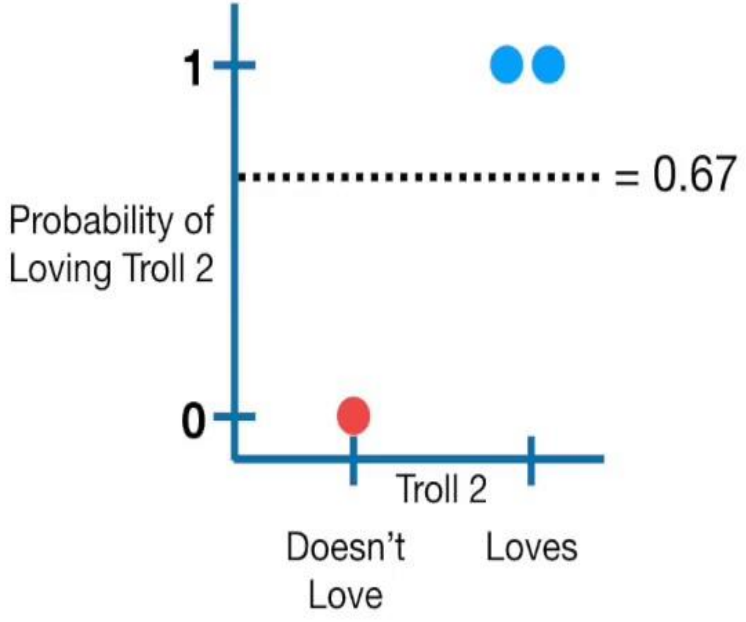
Now we need a differentiable **Loss Function** that will work for **Classification.**

GB Algorithm

Log(Likelihood of the Observed Data given the Prediction) =

$$\sum_{i=1}^{N} y_i \times \log(p) + (1 - y_i) \times \log(1 - p)$$

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
No	87	Green	Yes
No	44	Blue	No



$$-\left[\textbf{Observed} \times \log(\boldsymbol{p}) + (1 - \textbf{Observed}) \times \log(1 - \boldsymbol{p})\right]$$

$$\sum_{i=1}^{N} \boldsymbol{y}_i \times \log(\boldsymbol{p}) + (1 - \boldsymbol{y}_i) \times \log(1 - \boldsymbol{p})$$

1) $-\textbf{Observed} \times \log(\boldsymbol{p}) - (1 - \textbf{Observed}) \times \log(1 - \boldsymbol{p})$

2) $-\textbf{Observed} \times \log(\boldsymbol{p}) - \log(1 - \boldsymbol{p}) + \textbf{Observed} \times \log(1 - \boldsymbol{p})$

3) $-\textbf{Observed} \times [\log(\boldsymbol{p}) - \log(1 - \boldsymbol{p})] - \log(1 - \boldsymbol{p})$

4) $-\textbf{Observed} \times \log(\text{odds}) - \log(1 - \boldsymbol{p})$

5) $-\textbf{Observed} \times \log(\text{odds}) + \log(1 + e^{\log(\text{odds})})$

We converted the **negative log(likelihood)** of the data, which is a function of the predicted probability, ***p***…

…into a function of the predicted **log(odds)**.

This is the Loss function

GB Algorithm

$$\frac{d}{d\,\log(odds)}\ -\mathbf{Observed} \times \log(odds) + \log(1 + e^{\log(odds)}) =$$

So let's take the derivative of the **Loss Function** with respect to the predicted **log(odds)**.

GB Algorithm

$$\frac{d}{d\,log(odds)} \; \textbf{-Observed} \times log(odds) \; + log(1 + e^{log(odds)}) =$$

$$= \textbf{-Observed} + \frac{e^{log(odds)}}{1 + e^{log(odds)}}$$

$$= \textbf{-Observed} + \boldsymbol{p}$$

Indeed, in the previous example we performed

- 1 – 0.7 = 0.3
- 0 – 0.7 = -0.7

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

We have created the initial predicted **log(odds)**, $\boldsymbol{F_0(x)}$...

$$\boldsymbol{F_0(x)} = \log\left(\frac{2}{1}\right)$$

GB Algorithm

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

$$\boldsymbol{F_0(x)} = \log\left(\frac{2}{1}\right)$$

$$= 0.69$$

So this is the initial leaf, $\boldsymbol{F_0(x)}$.

log(2/1) = 0.69

GB Algorithm

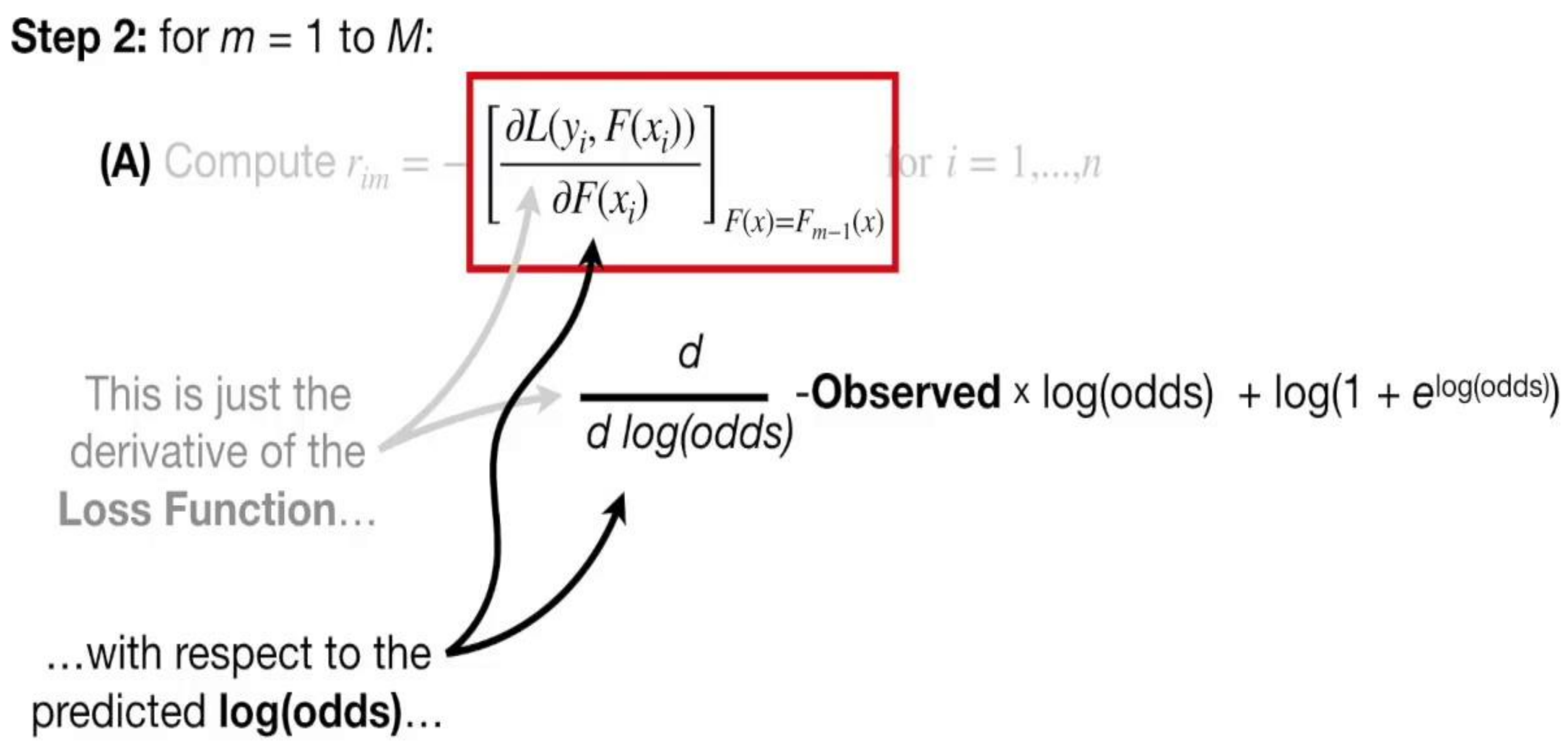
Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

This is just the derivative of the **Loss Function**…

$$\frac{d}{d\,log(odds)}\ -\textbf{Observed} \times \log(odds) + \log(1 + e^{\log(odds)})$$

GB Algorithm



GB Algorithm

Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

$$\frac{d}{d\,log(odds)}\ \textbf{-Observed} \times \log(\text{odds}) + \log(1 + e^{\log(\text{odds})})$$

…and we've already calculated this.

$$= (\textbf{-Observed} + \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}})$$

GB Algorithm

Step 2: for m = 1 to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,...,n$

...and that leaves us with this equation for calculating **Pseudo Residuals.**

$$(\textbf{Observed} - \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}})$$

GB Algorithm

Step 2: for m = 1 to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,...,n$

NOTE: As we have seen before, we can replace this term with the predicted probability, ***p***… → (**Observed** - ***p***)

$$\left(\textbf{Observed} - \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}\right)$$

GB Algorithm

Step 2: for m = 1 to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,...,n$

...so we can think of the **Pseudo Residuals** as the **Observed** probability minus the **Predicted** probability... → (**Observed** - ***p***)

$$(\text{Observed} - \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}})$$

GB Algorithm

Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

Now we are ready for **Part B**, where we will build a regression tree.

GB Algorithm

Now let's do **Part C**.

Step 2: for $m = 1$ to M:

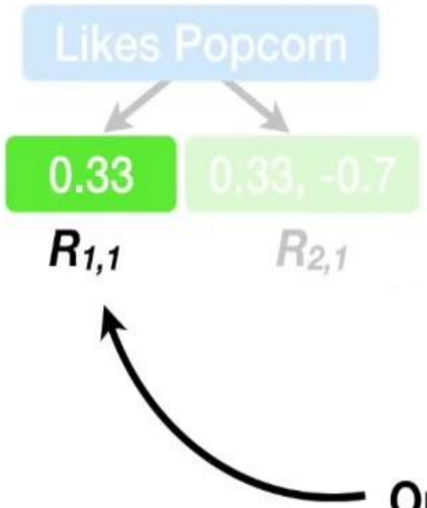
(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

GB Algorithm

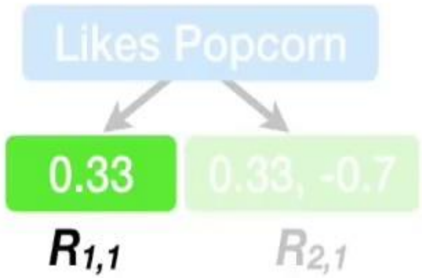
Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
No	87	Green	Yes
No	44	Blue	No



find the
Output Value for this leaf.

$$\gamma_{1,1} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$

GB Algorithm



Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
No	87	Green	Yes
No	44	Blue	No

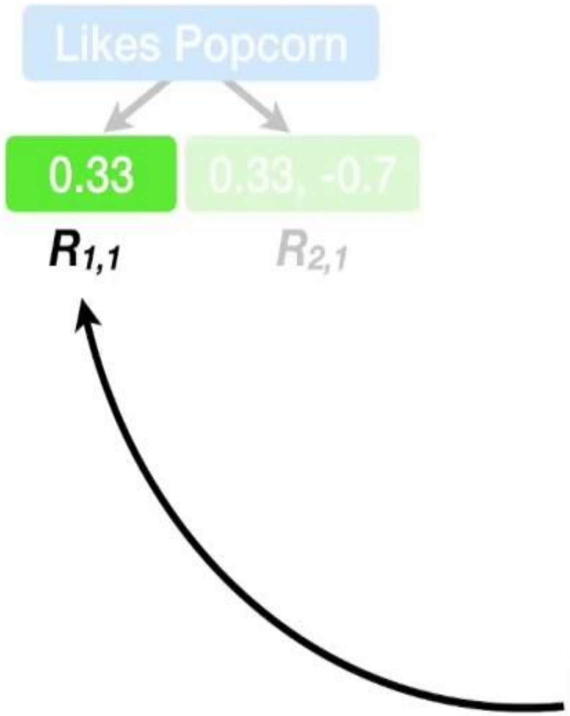
$$\frac{\sum \text{Residual}_i}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right]}$$

$$\gamma_{1,1} = \frac{\textbf{Residual}}{\boldsymbol{p} \times (1 - \boldsymbol{p})}$$

0.33/(2/3 x 1/3) = 1.48

GB Algorithm

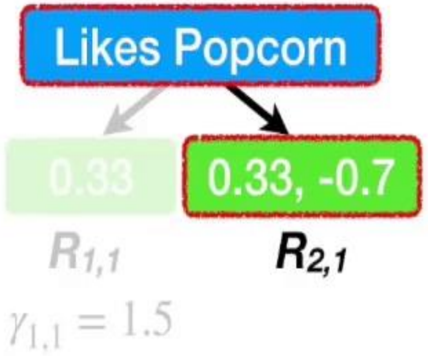
Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
No	87	Green	Yes
No	44	Blue	No



...and the output value for leaf $R_{1,1}$ is **1.5**.

$$\gamma_{1,1} = 1.5$$

GB Algorithm

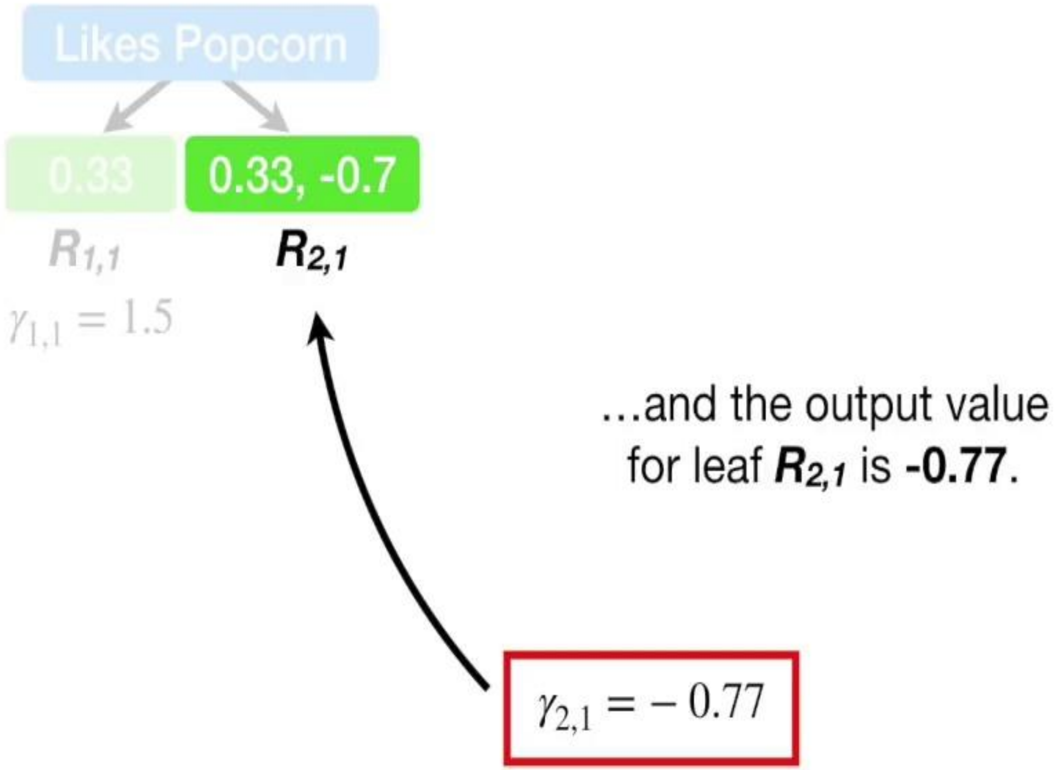


Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
No	87	Green	Yes
No	44	Blue	No

$$\gamma_{2,1} = \frac{\text{Residual}_2 + \text{Residual}_3}{[\, p_2 \times (1 - p_2)\,] + [\, p_3 \times (1 - p_3)\,]}$$

$$\frac{\sum \text{Residual}_i}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)]}$$

GB Algorithm

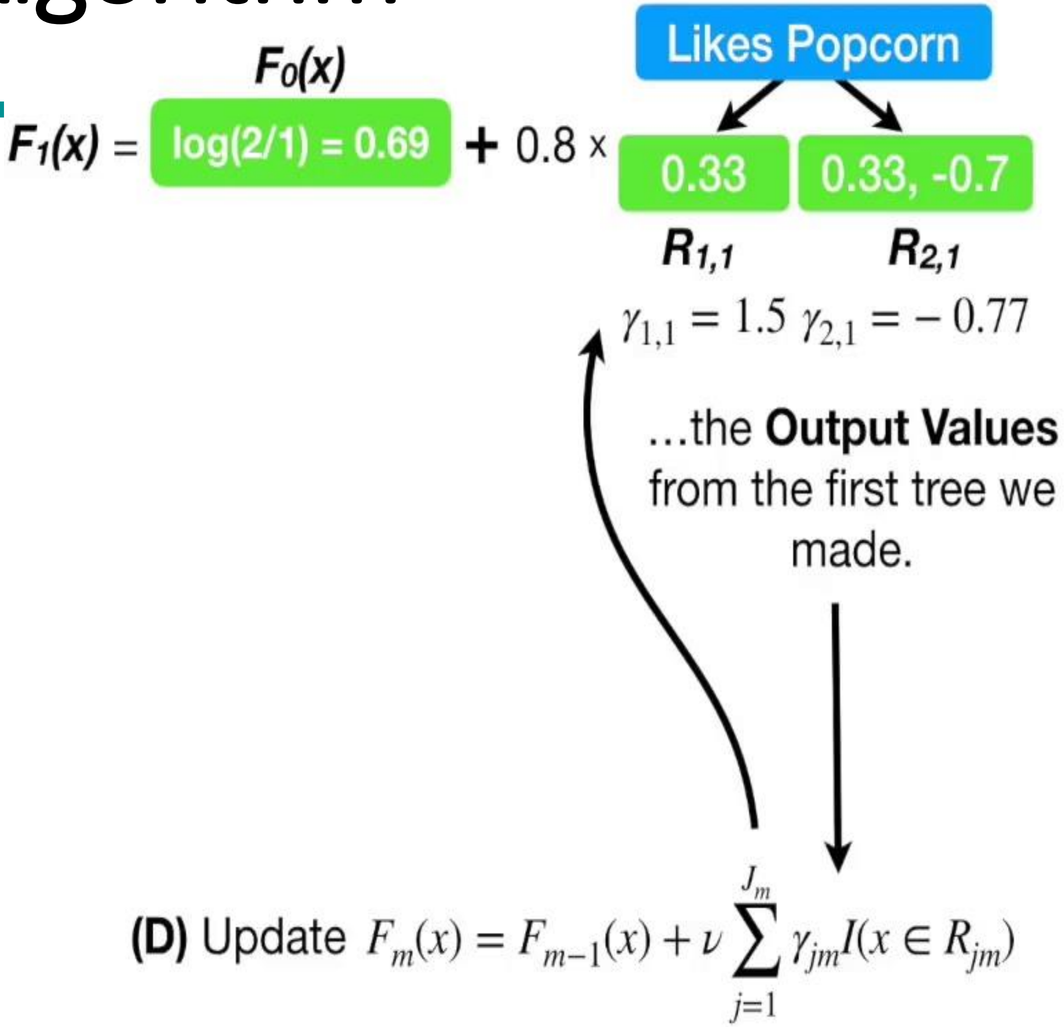


GB Algorithm

In **Part D**, we make a new prediction for each sample.

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

GB Algorithm



Gradient Boost Improved

- XGBoost (Extreme Gradient Boost)
- LightGBM (Light Gradient Boosting Machine)
- Both designed for large complex datasets

Mass	Age	BP	Color	Movie	Car	Hair	etc.
120	23	102	Brown	T2	Ford	Long	...
150	25	98	Brown	Frozen	Kia	Short	...
165	22	130	Black	Spiderman	Ford	Short	...
123	45	98	Red	T2	Kia	Long	...
156	33`	78	Brown	Frozen 2	Ford	Long	...
...	...	...	...	...	...	...	...

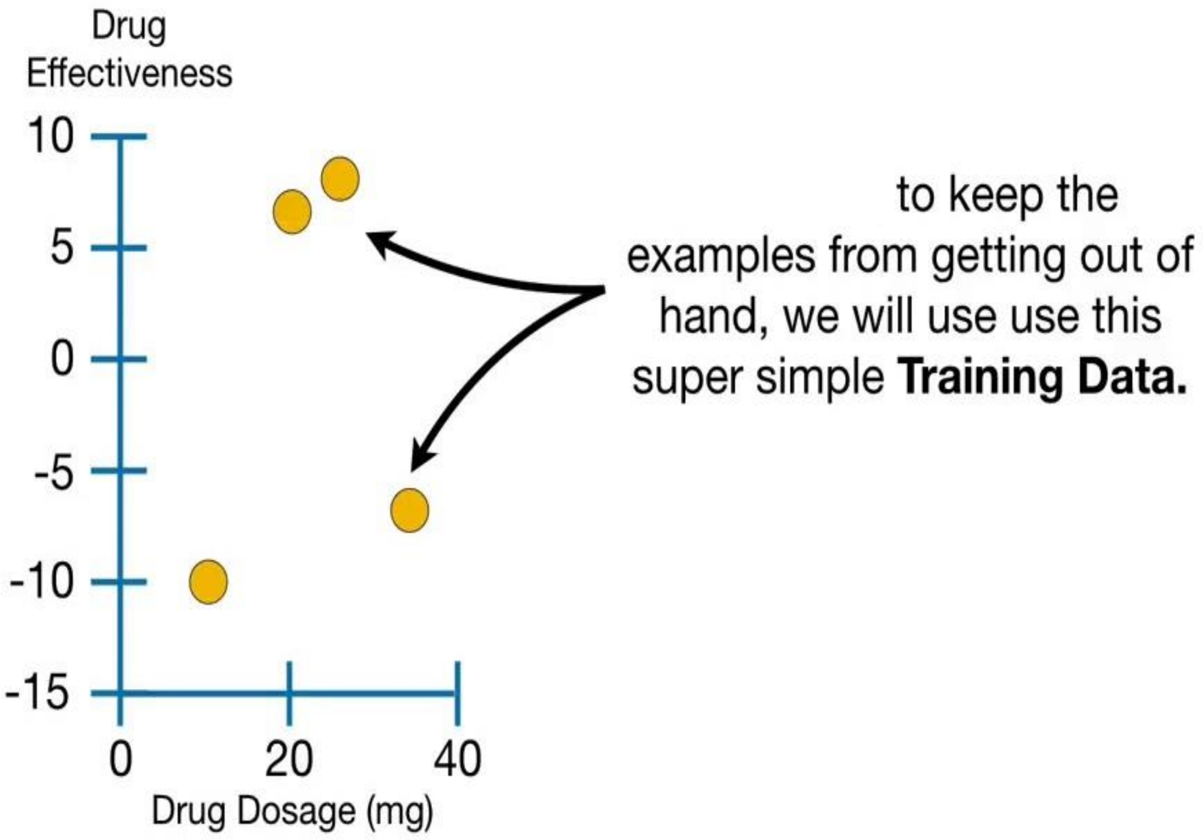
XGBoost Characteristics

- Gradient Boost
- Regularization
- Approximate Greedy Algorithm
- Weighted Quantile Sketch
- Sparsity-Aware Split Finding
- Parallel Learning
- Cache-Aware Access
- Blocks for Out-of-Core Computation

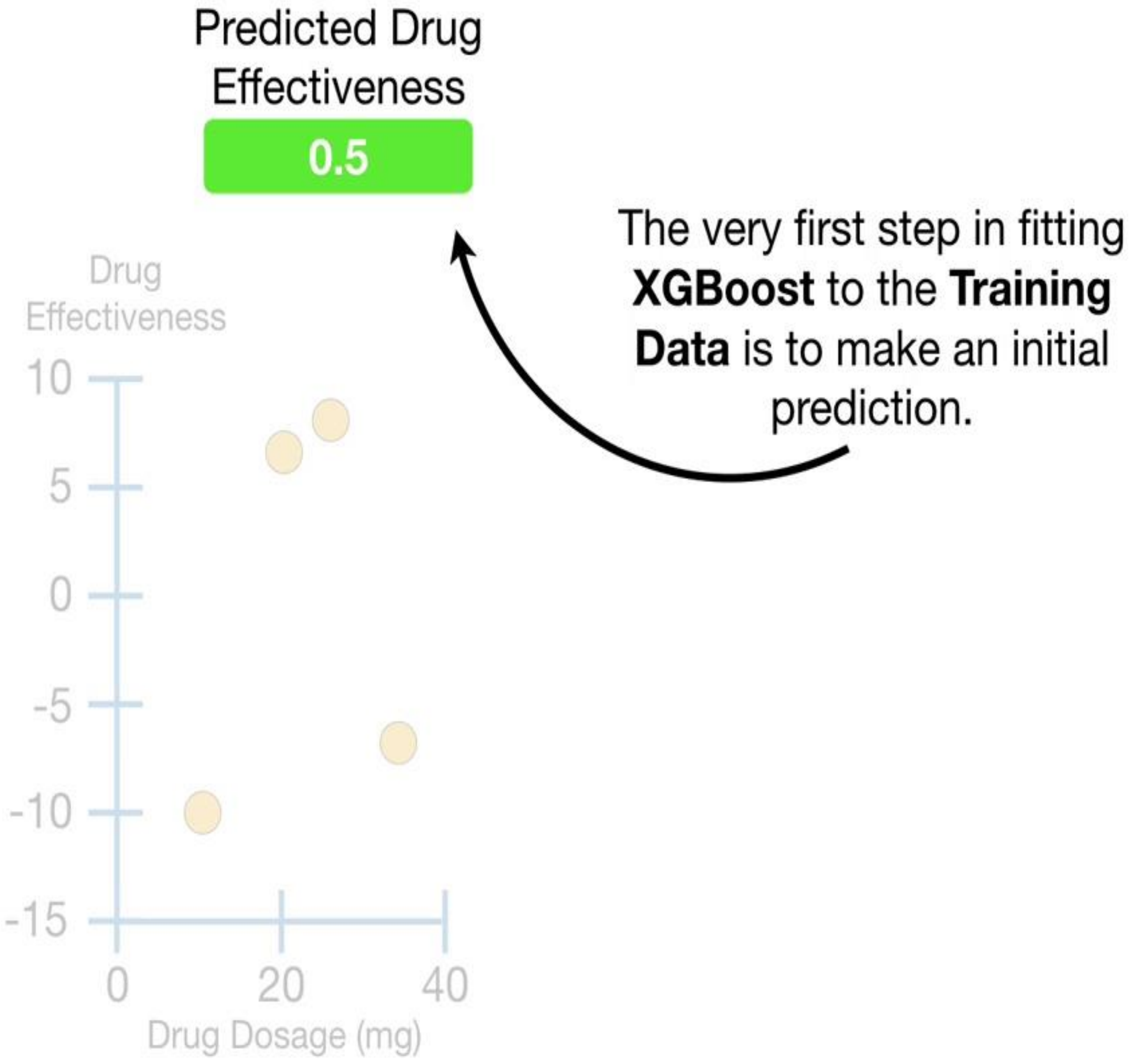
XGBoost Characteristics

- Gradient Boost
- Regularization
- Approximate Greedy Algorithm
- Weighted Quantile Sketch
- Sparsity-Aware Split Finding
- Parallel Learning
- Cache-Aware Access
- Blocks for Out-of-Core Computation

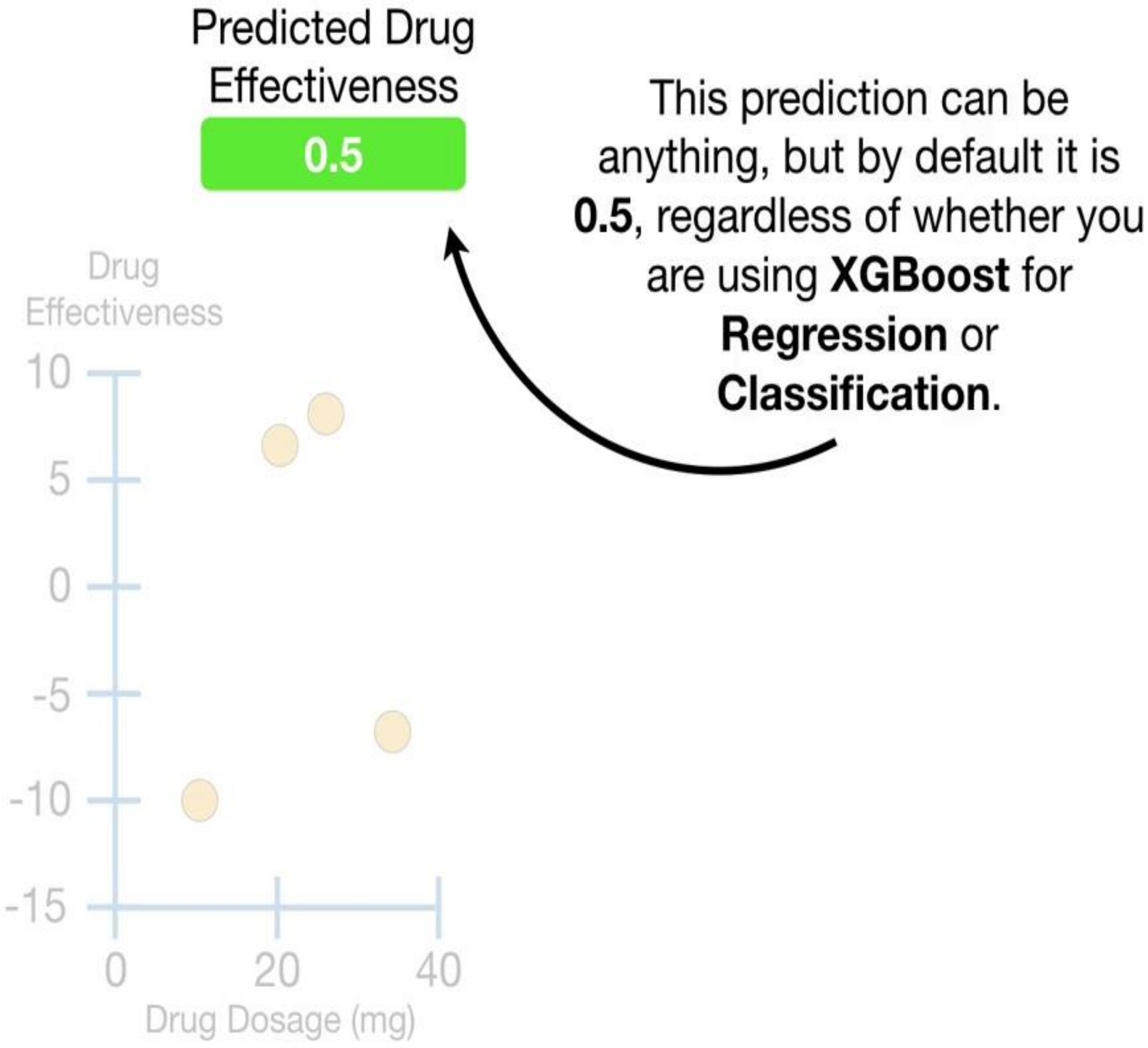
XGBoost Tree



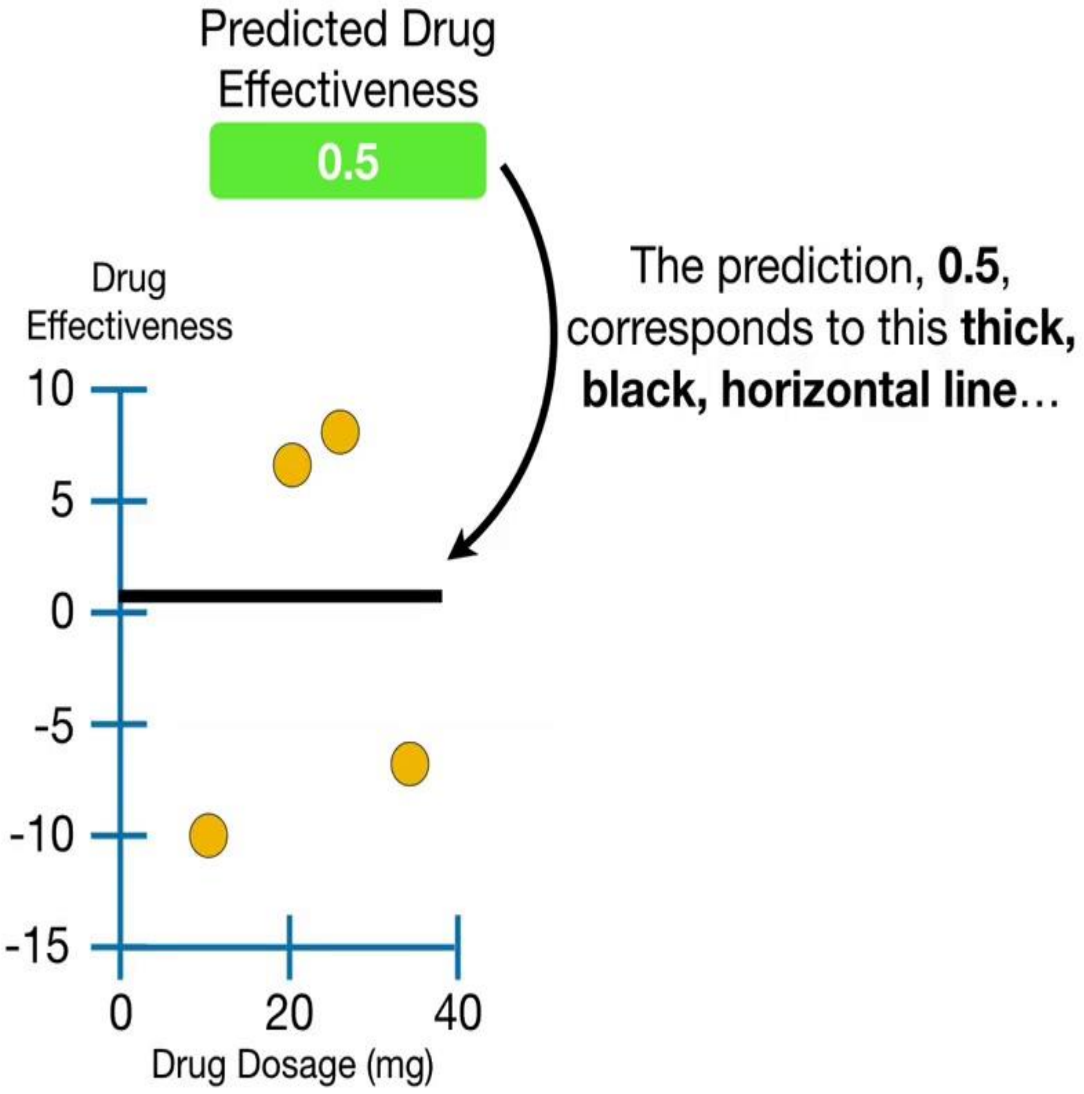
XGBoost Tree



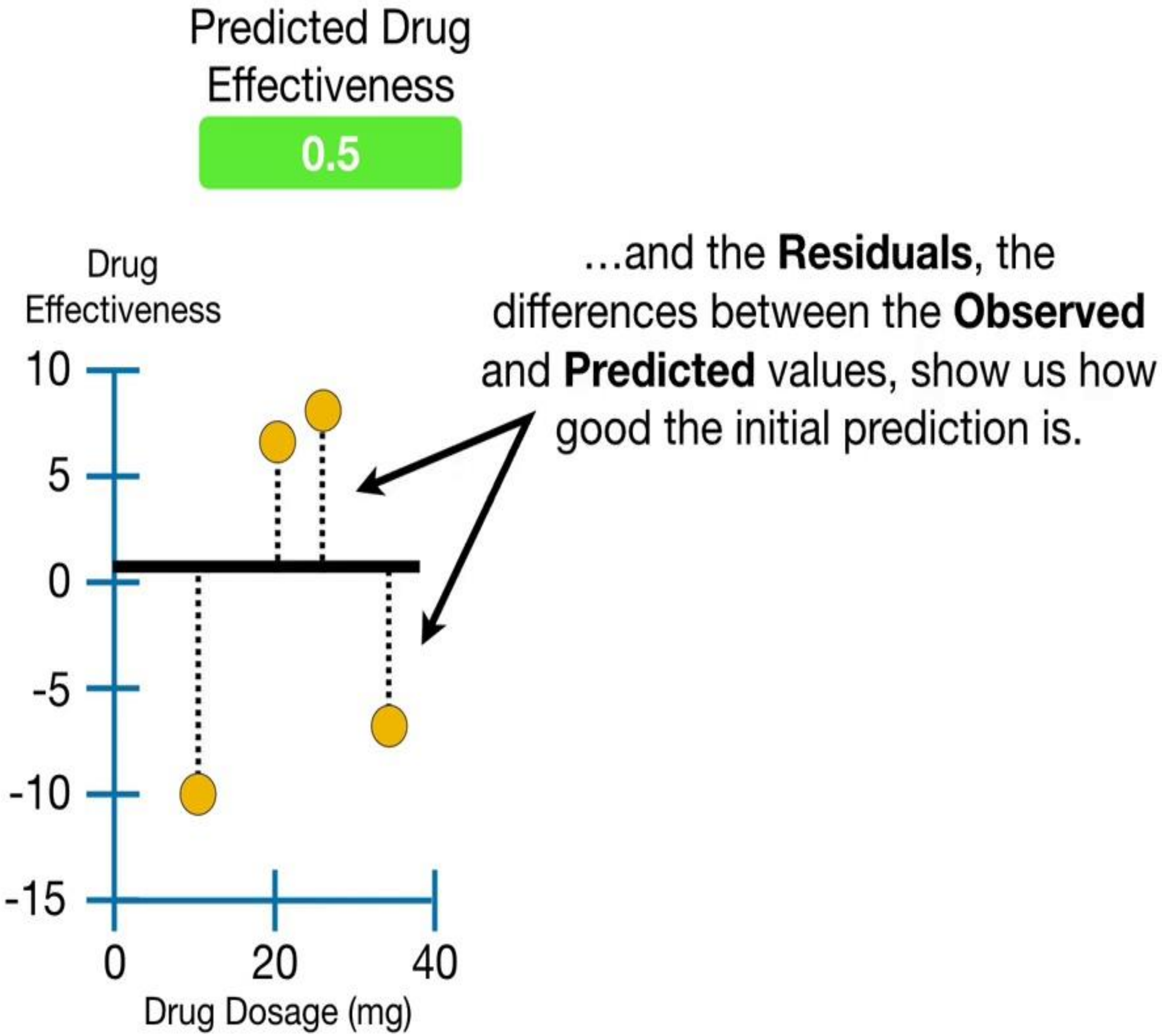
XGBoost Tree



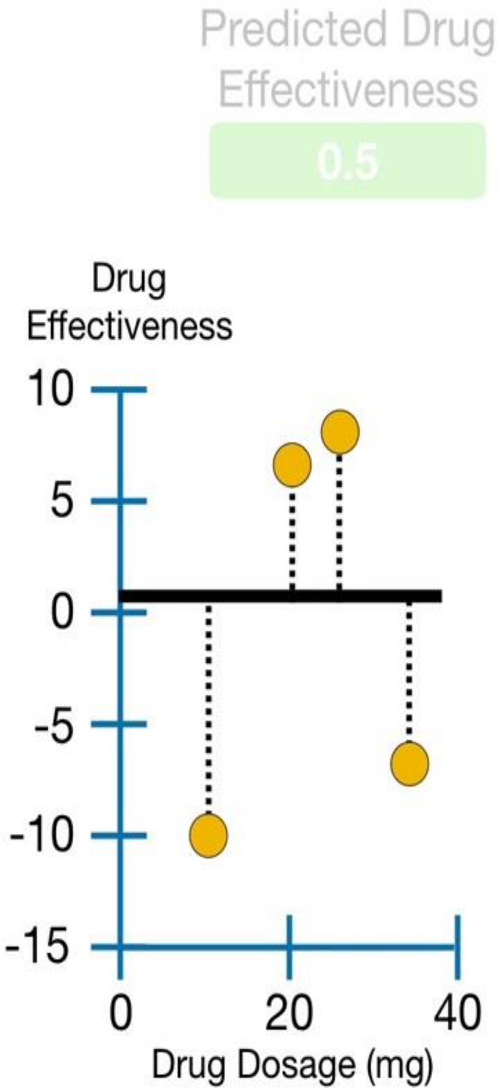
XGBoost Tree

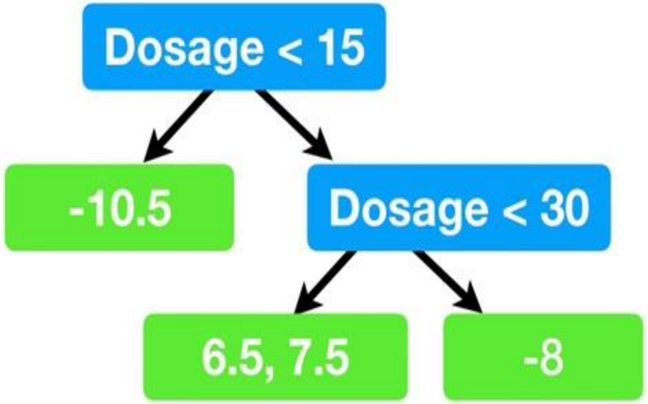


XGBoost Tree



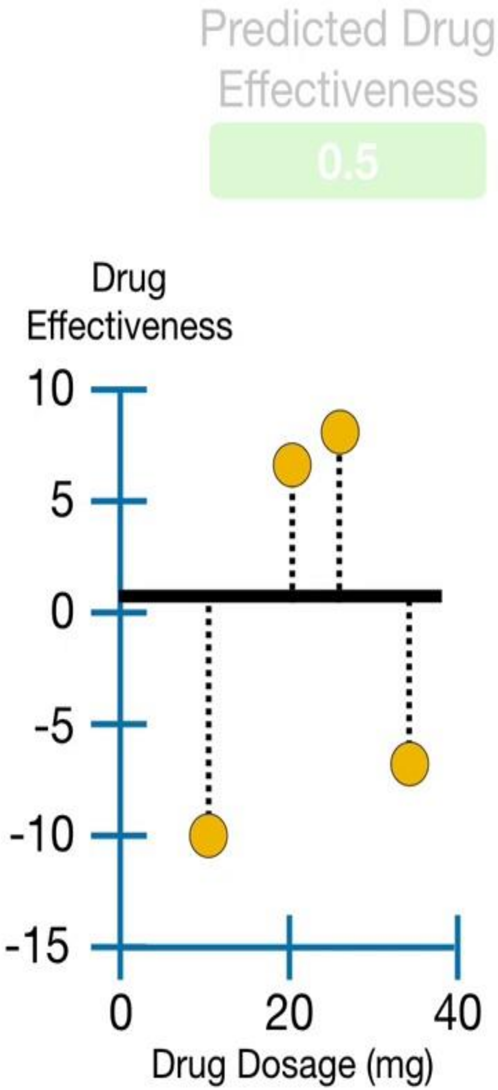
XGBoost Tree

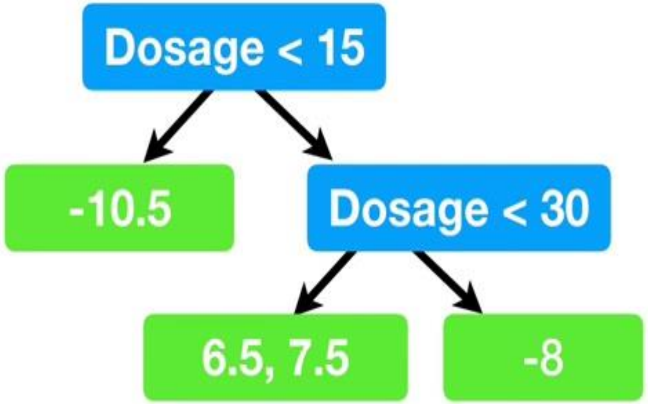




However, unlike unextreme **Gradient Boost**, which typically uses regular, off-the-shelf, **Regression Trees**…

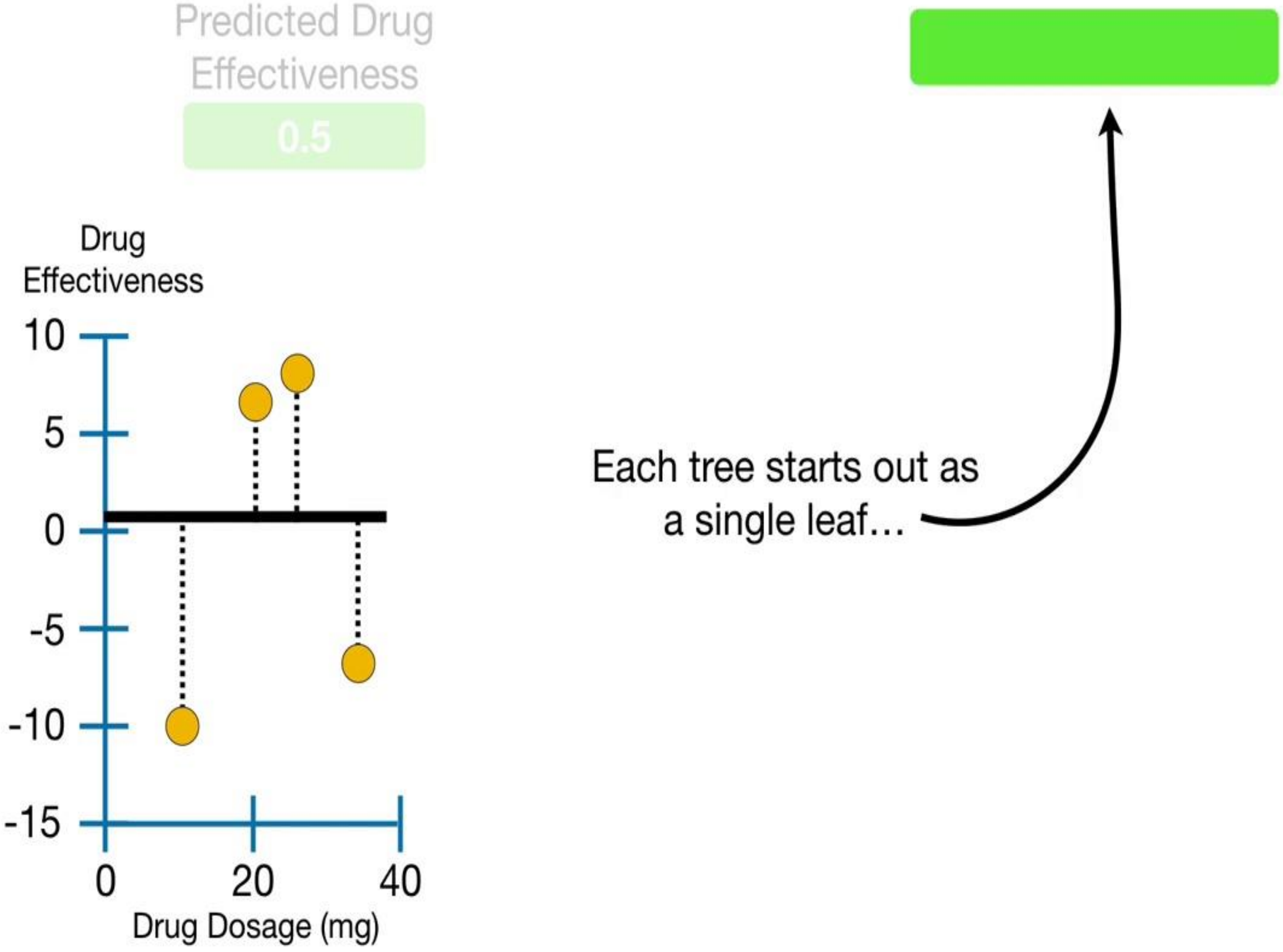
XGBoost Tree



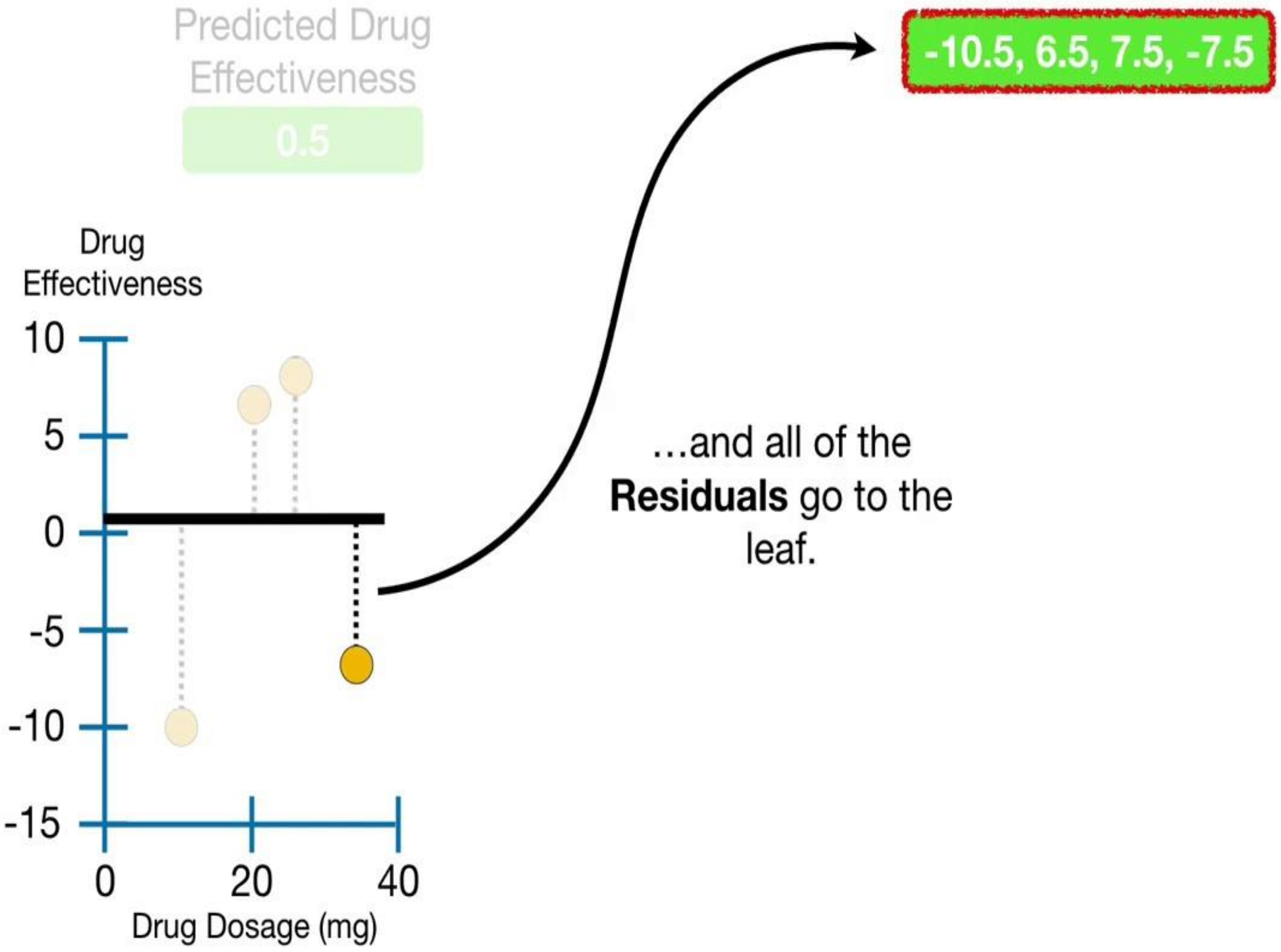


...**XGBoost** uses a unique **Regression Tree** that I call an **XGBoost Tree**.

XGBoost Tree



XGBoost Tree

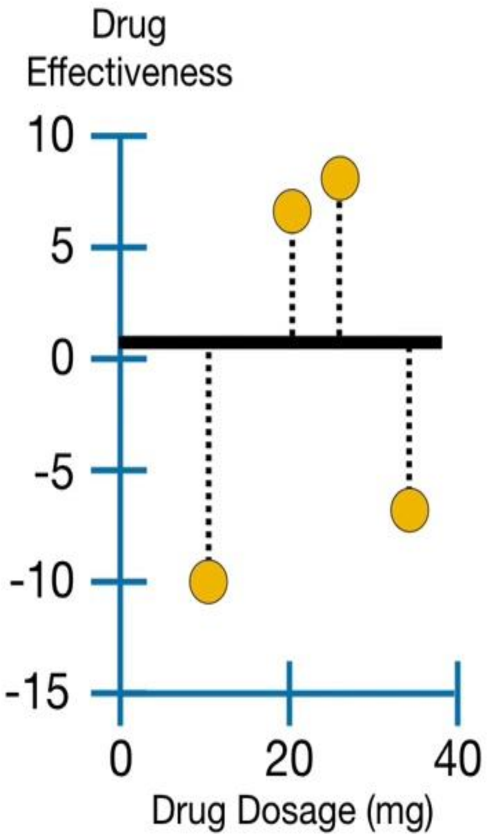


XGBoost Tree

Predicted Drug Effectiveness

0.5

-10.5, 6.5, 7.5, -7.5



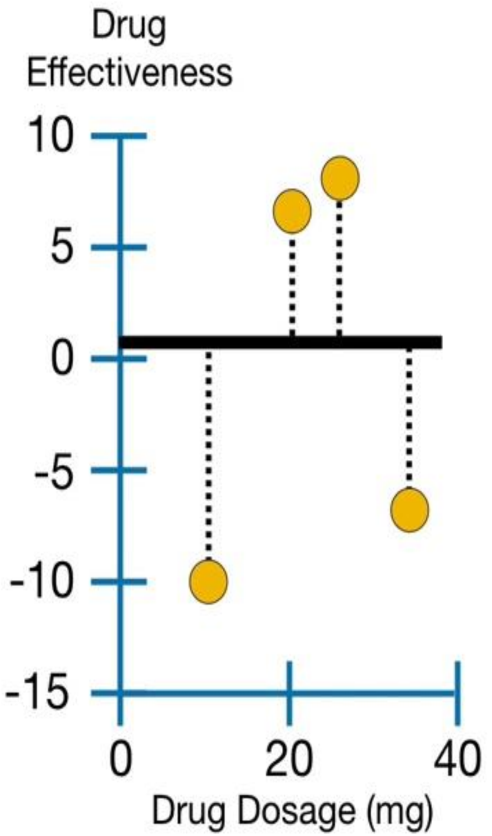
Now we calculate a **Quality Score**, or **Similarity Score**, for the **Residuals**.

XGBoost Tree

Predicted Drug Effectiveness

0.5

-10.5, 6.5, 7.5, -7.5



$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

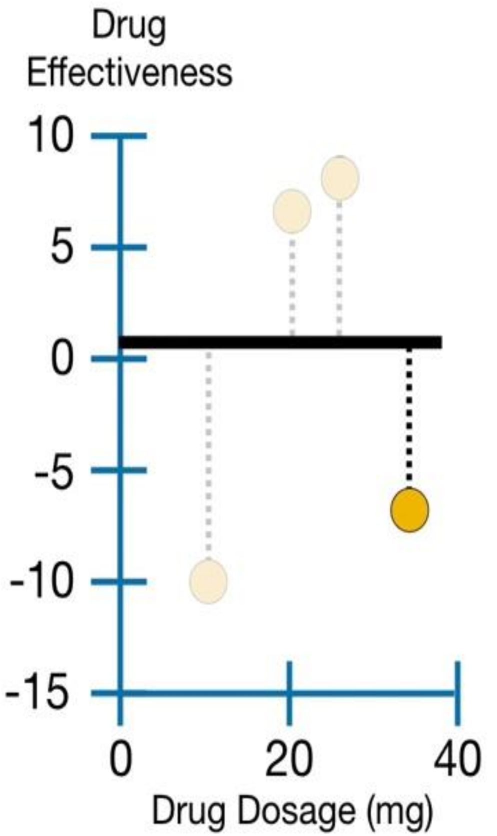
NOTE: **λ (lambda)** is a **Regularization** parameter,

XGBoost Tree

Predicted Drug Effectiveness

0.5

-10.5, 6.5, 7.5, -7.5



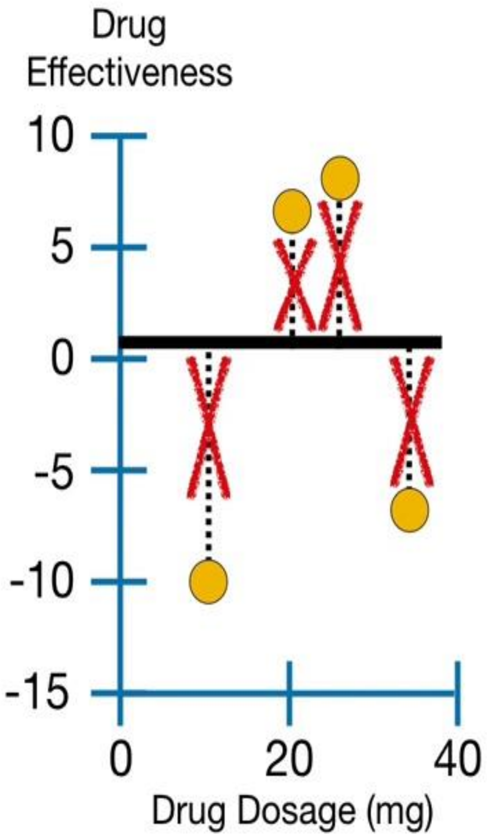
$$\text{Similarity Score} = \frac{(-10.5 + 6.5 + 7.5 + -7.5)^2}{4 + \mathbf{0}}$$

XGBoost Tree

Predicted Drug Effectiveness

0.5

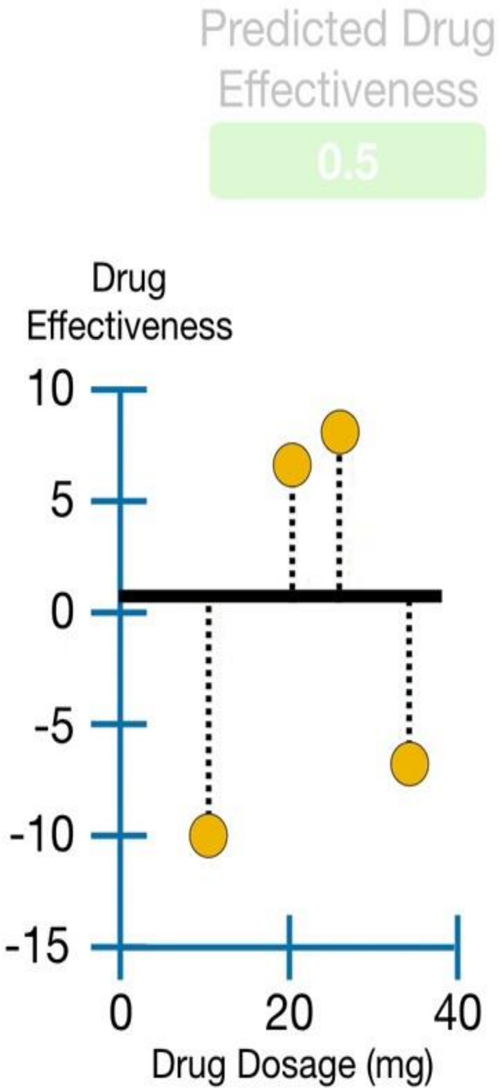
-10.5, 6.5, 7.5, -7.5

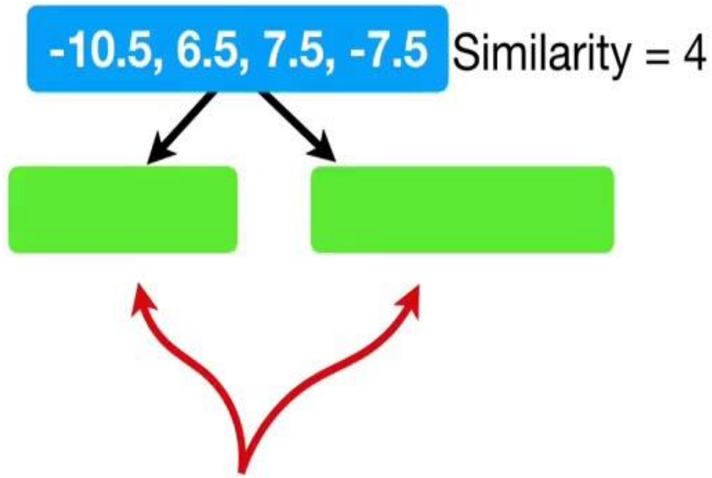


$$\text{Similarity Score} = \frac{(-4)^2}{4 + \mathbf{0}} = 4$$

Thus, the **Similarity Score** for the **Residuals** in the root = **4**.

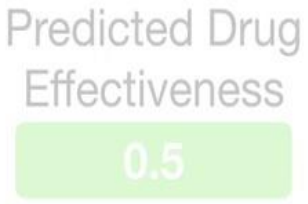
XGBoost Tree

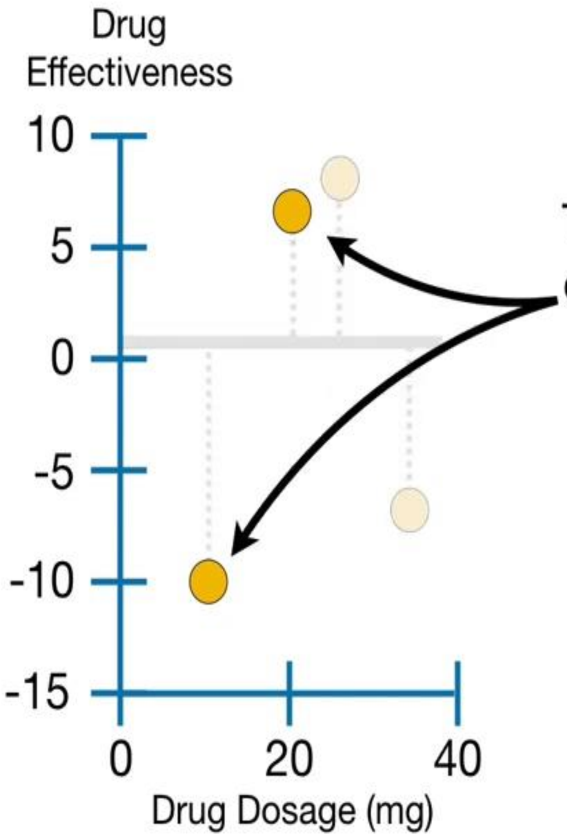




Now the question is whether or not we can do a better job clustering similar **Residuals** if we split them into two groups.

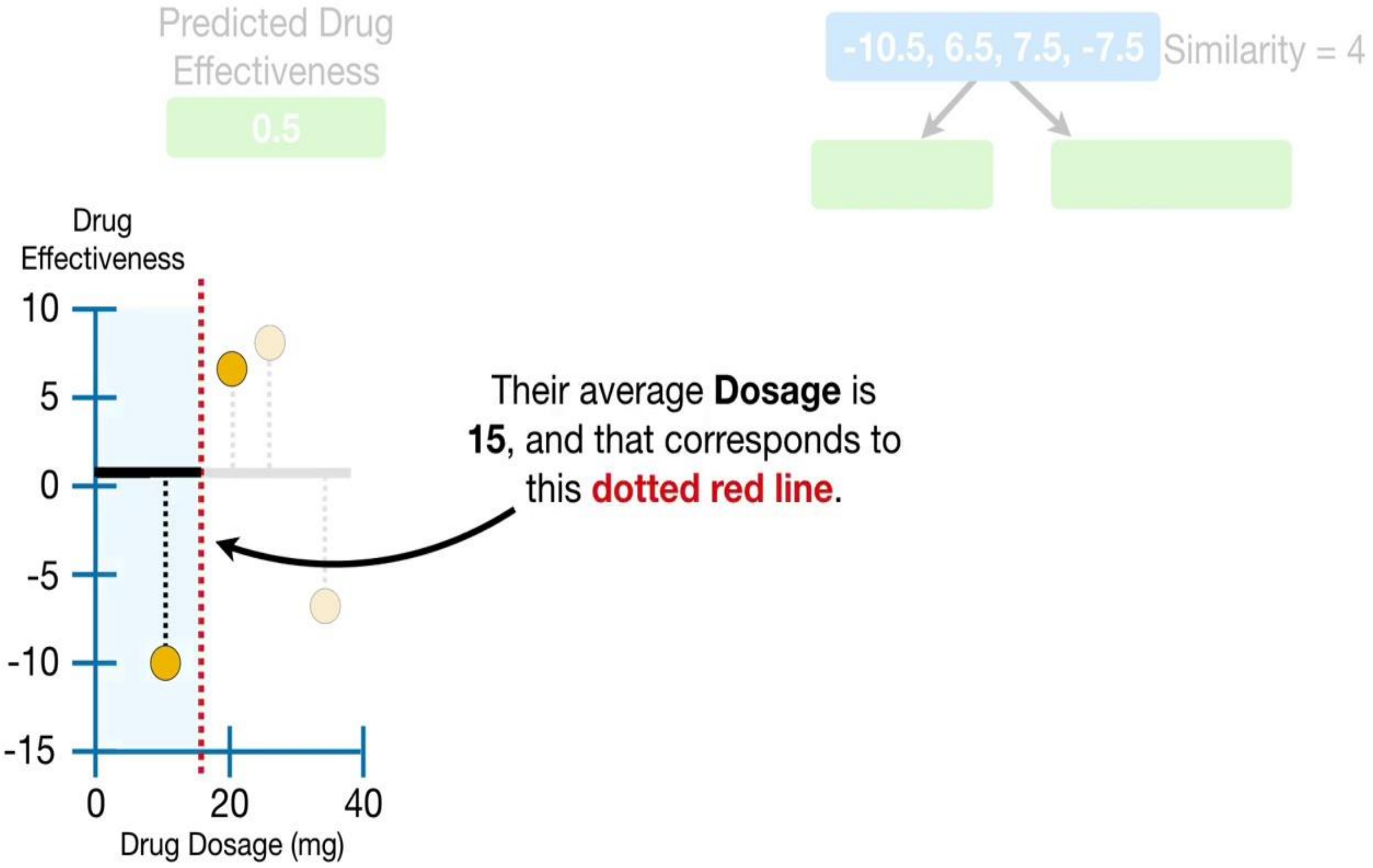
XGBoost Tree



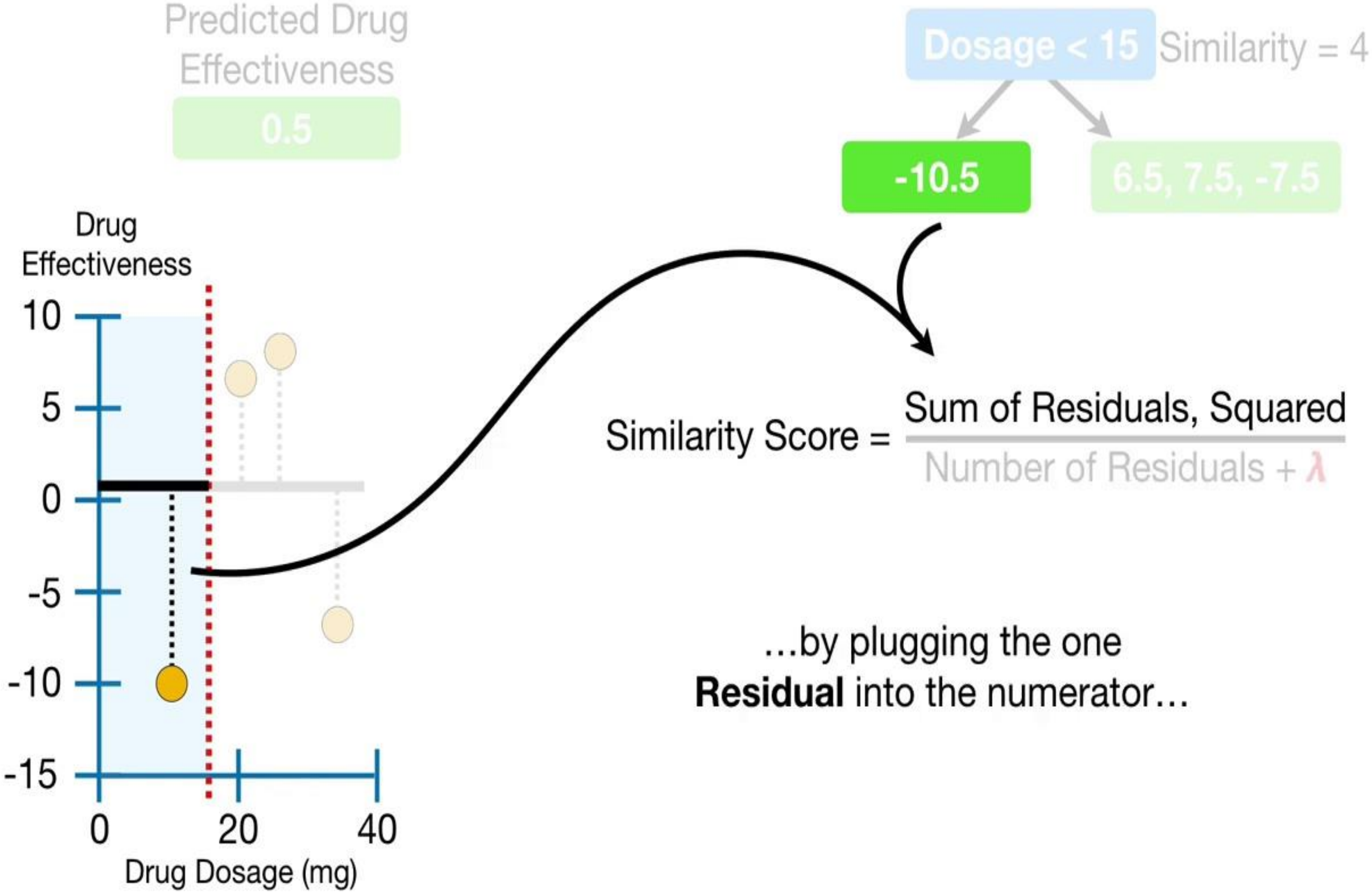


To answer this, we first focus on the two observations with the lowest **Dosages**.

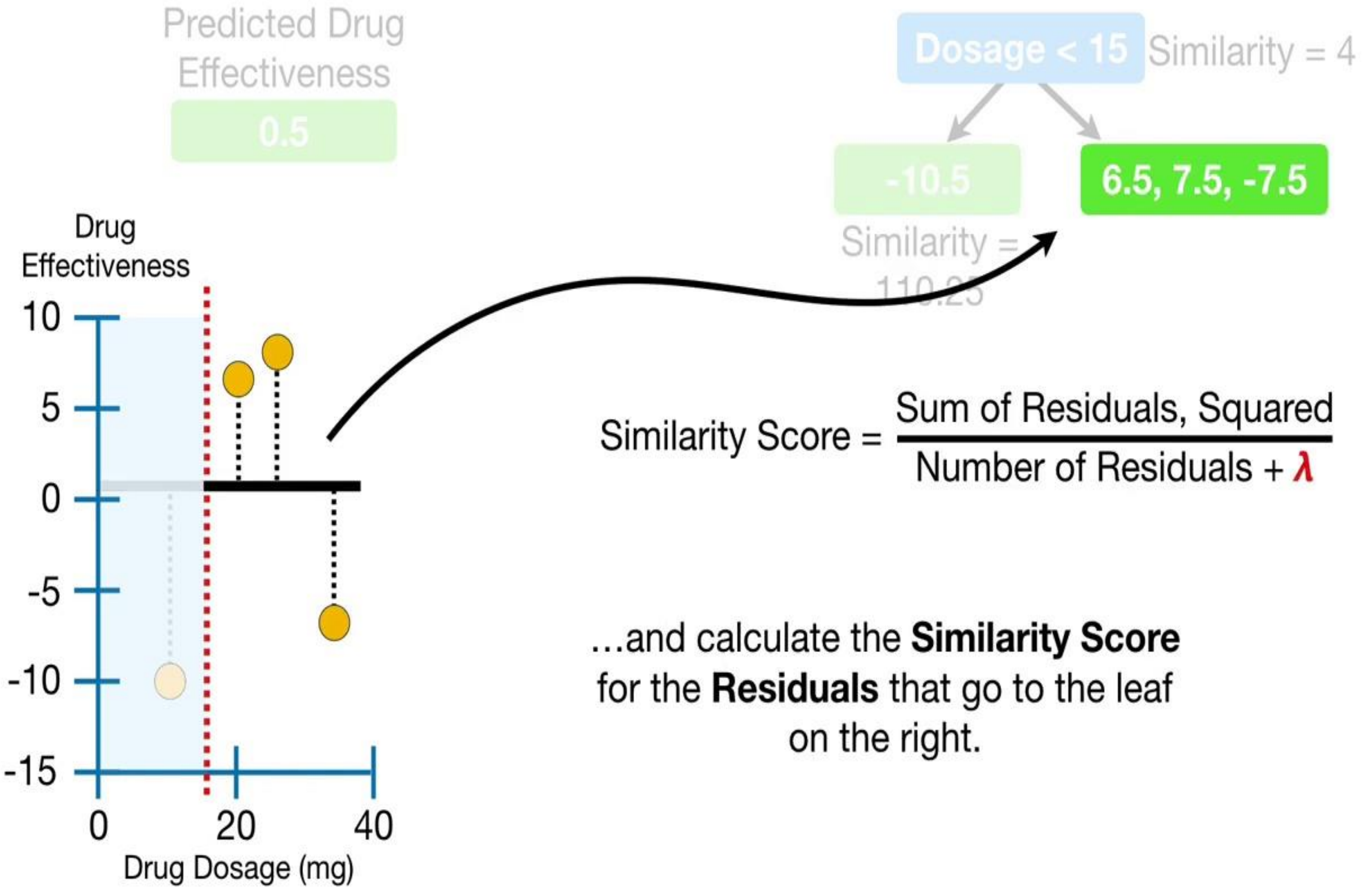
XGBoost Tree



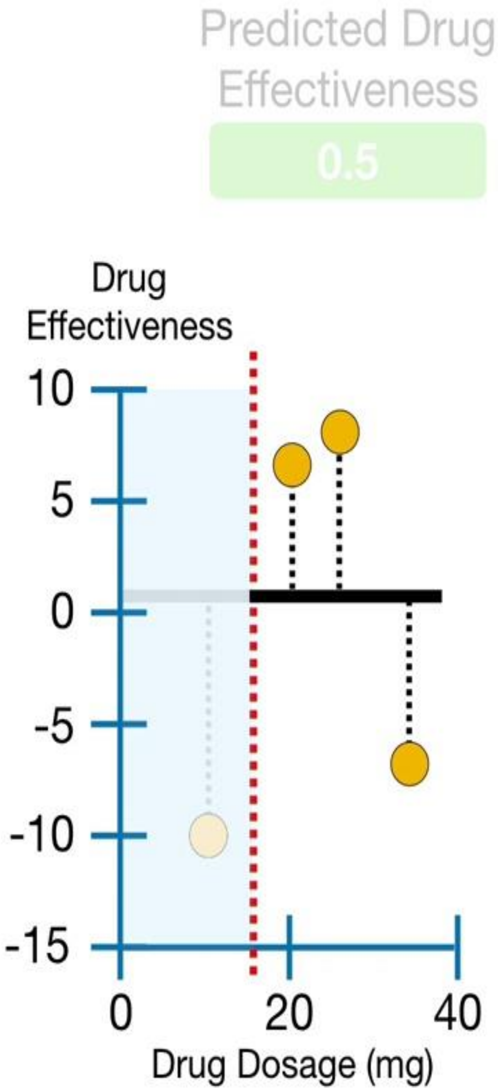
XGBoost Tree

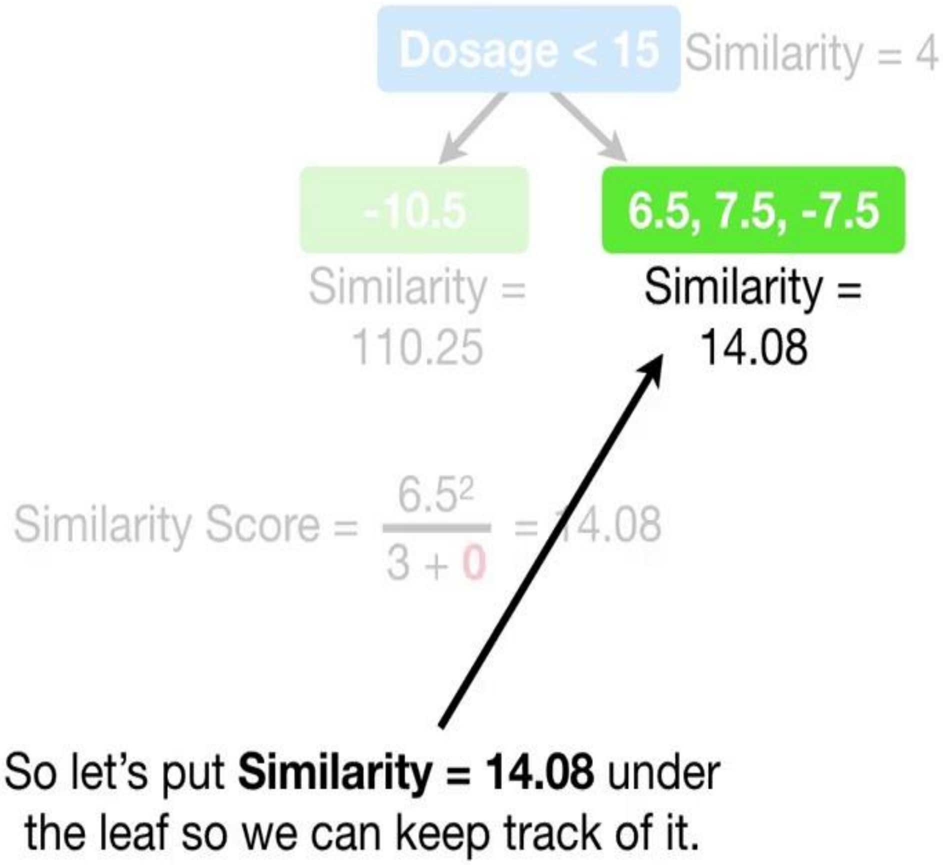


XGBoost Tree

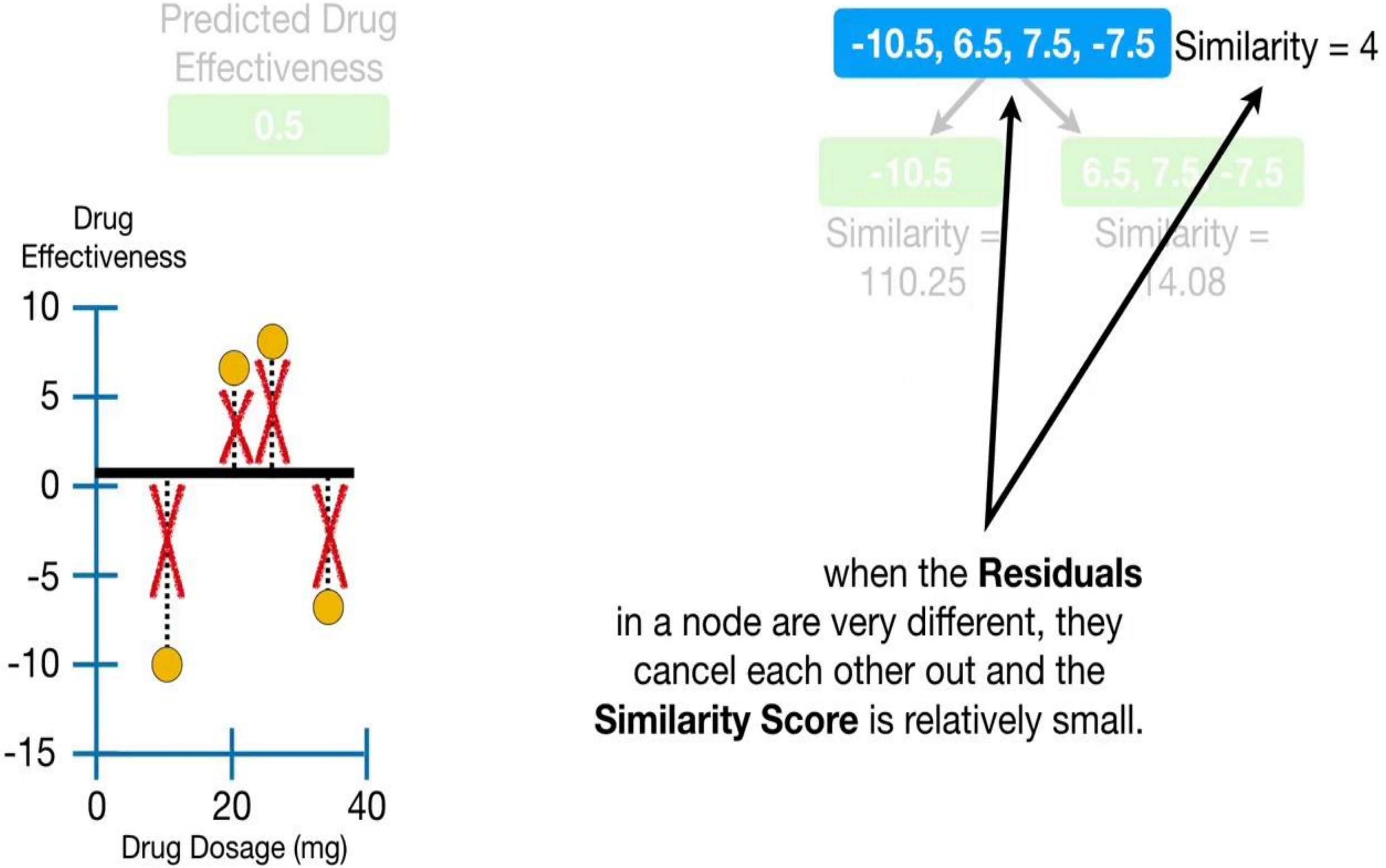


XGBoost Tree

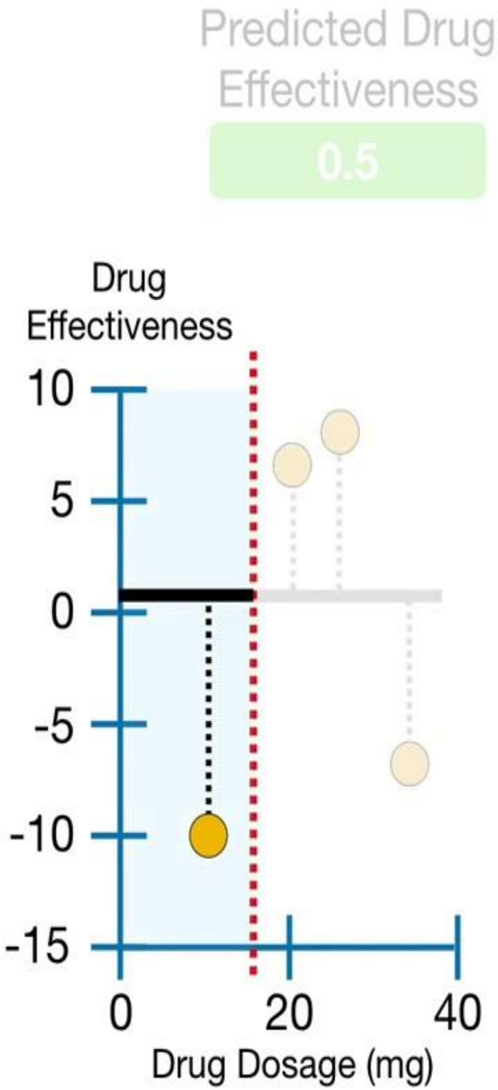


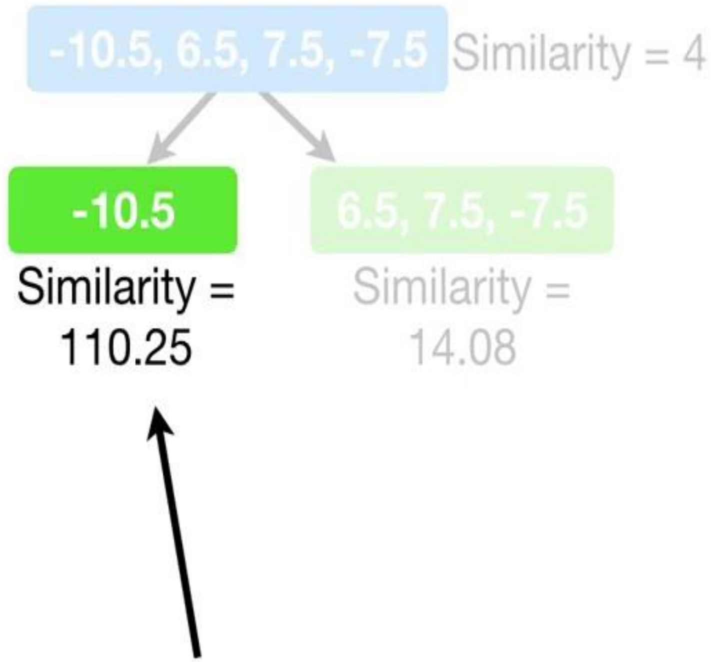


XGBoost Tree



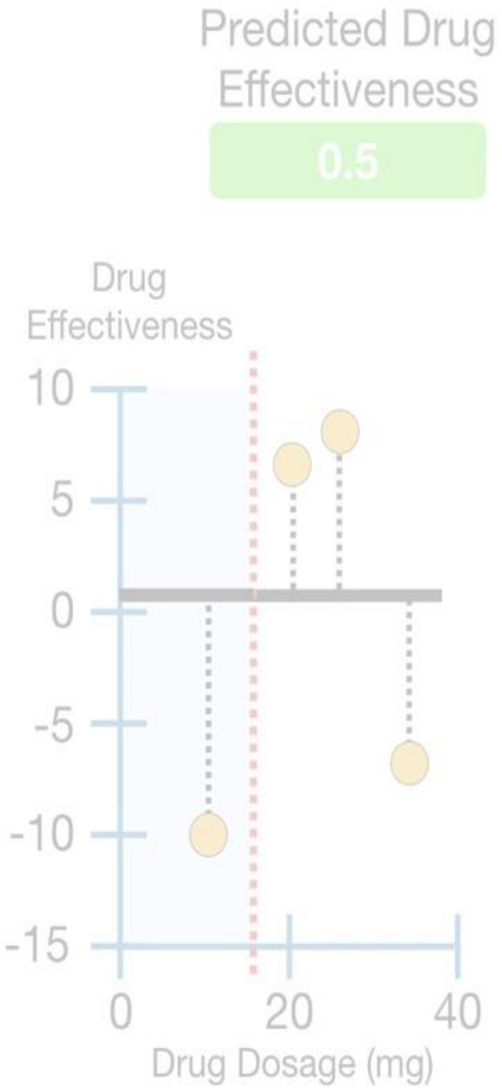
XGBoost Tree

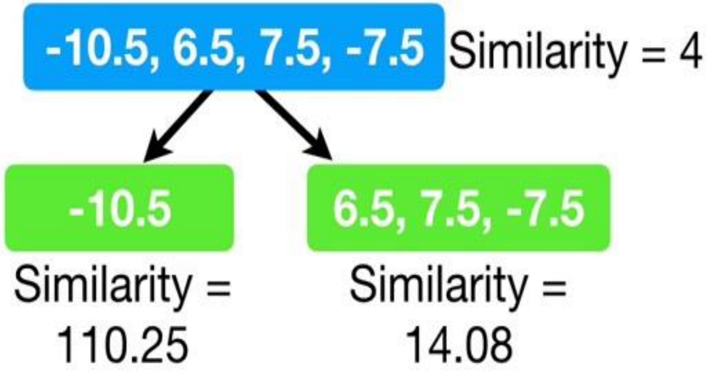




In contrast, when the **Residuals** are similar, or there is just one of them, they do not cancel out and the **Similarity Score** is relatively large.

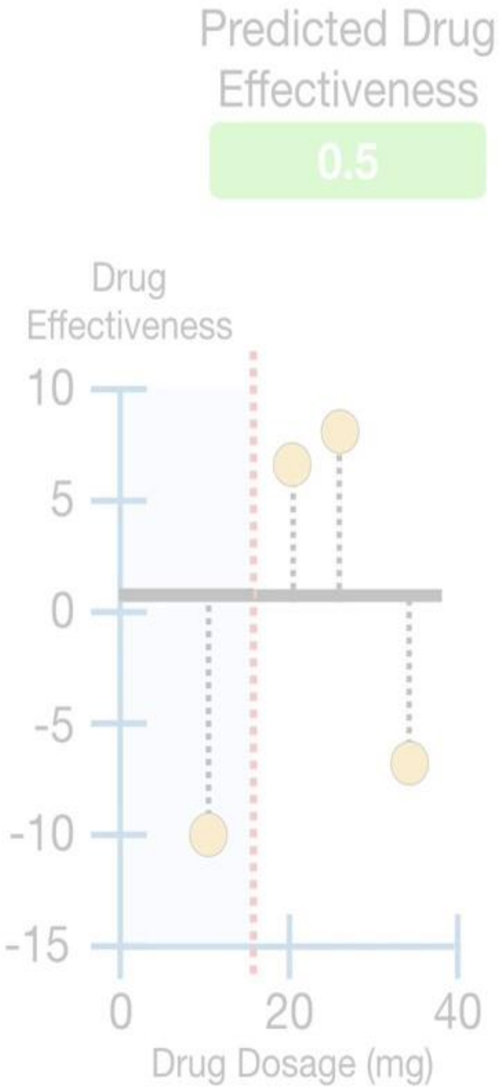
XGBoost Tree

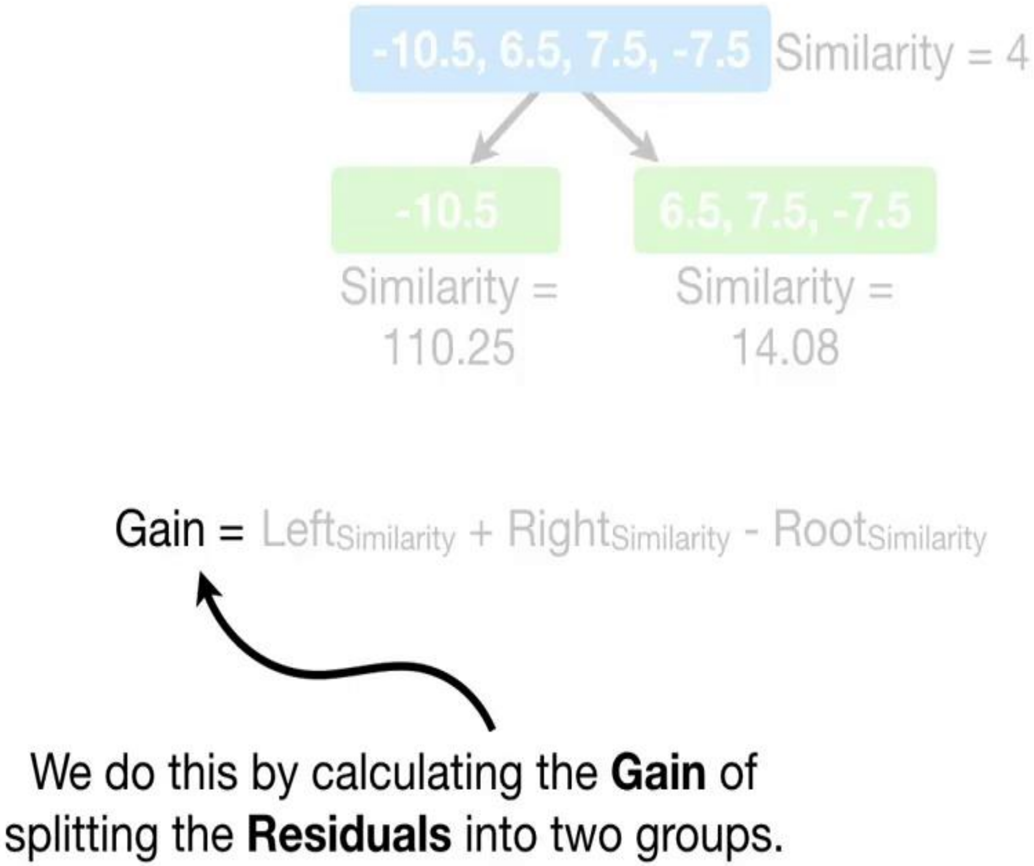




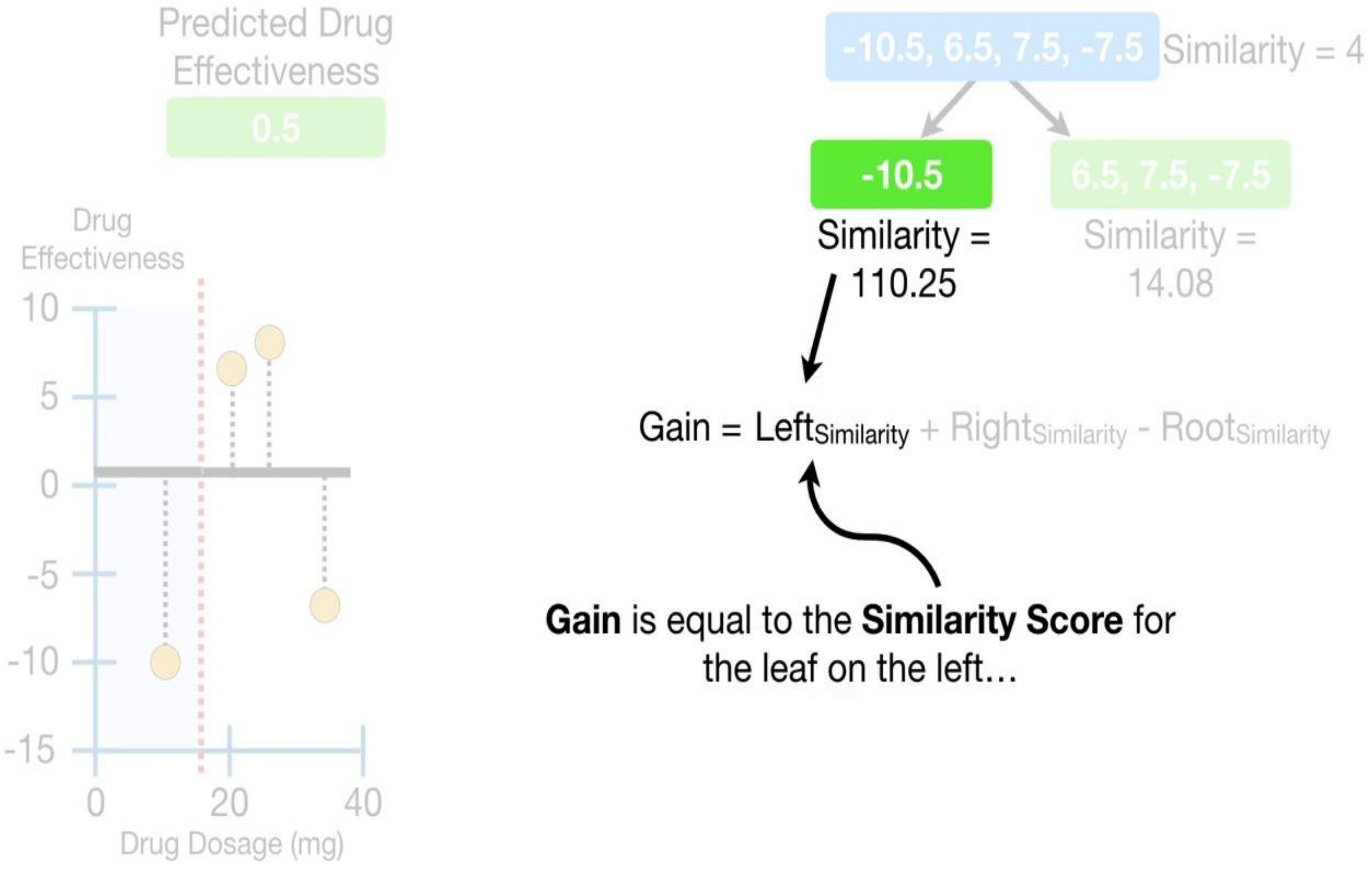
Now we need to quantify how much better the leaves cluster similar **Residuals** than the root.

XGBoost Tree

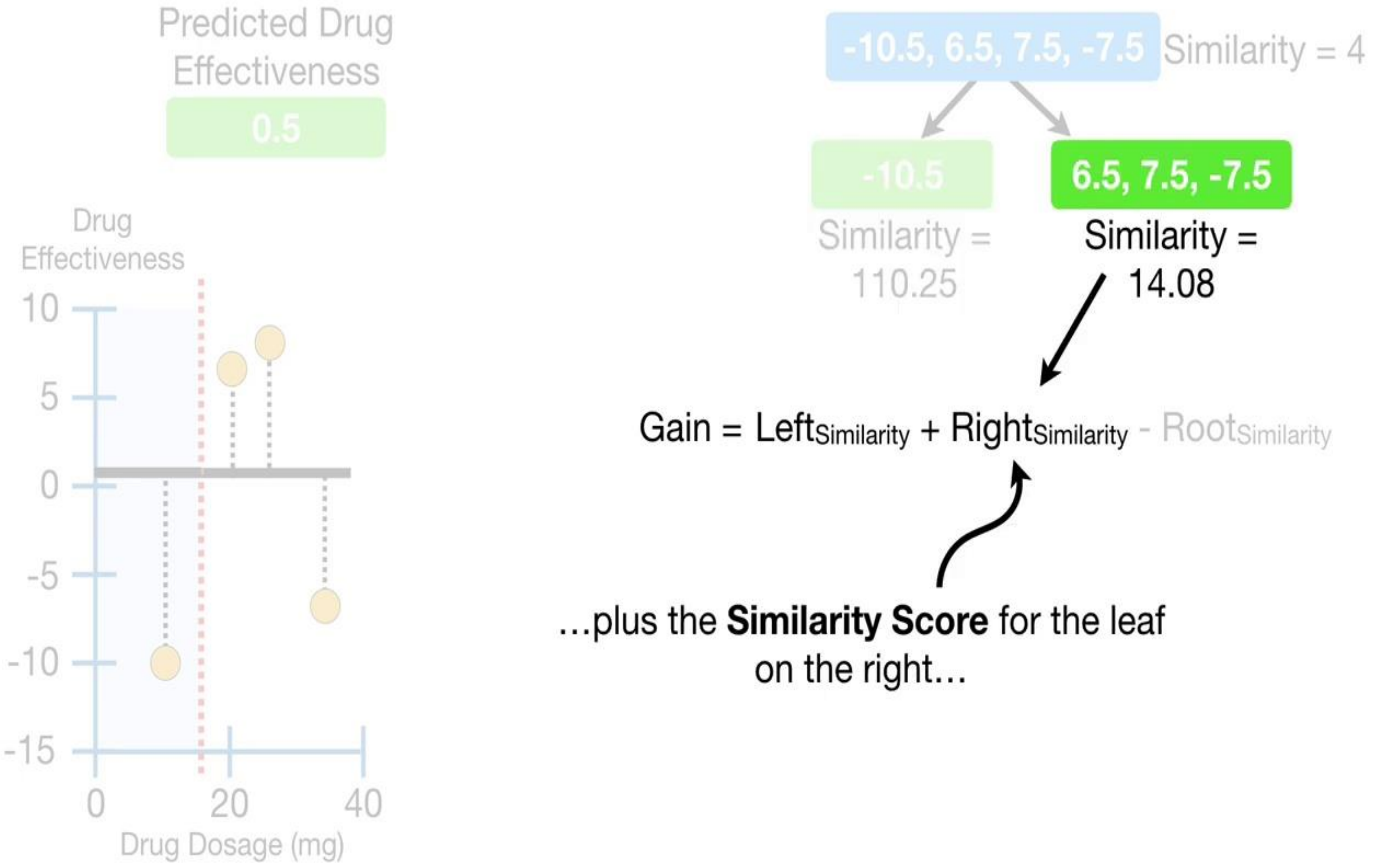




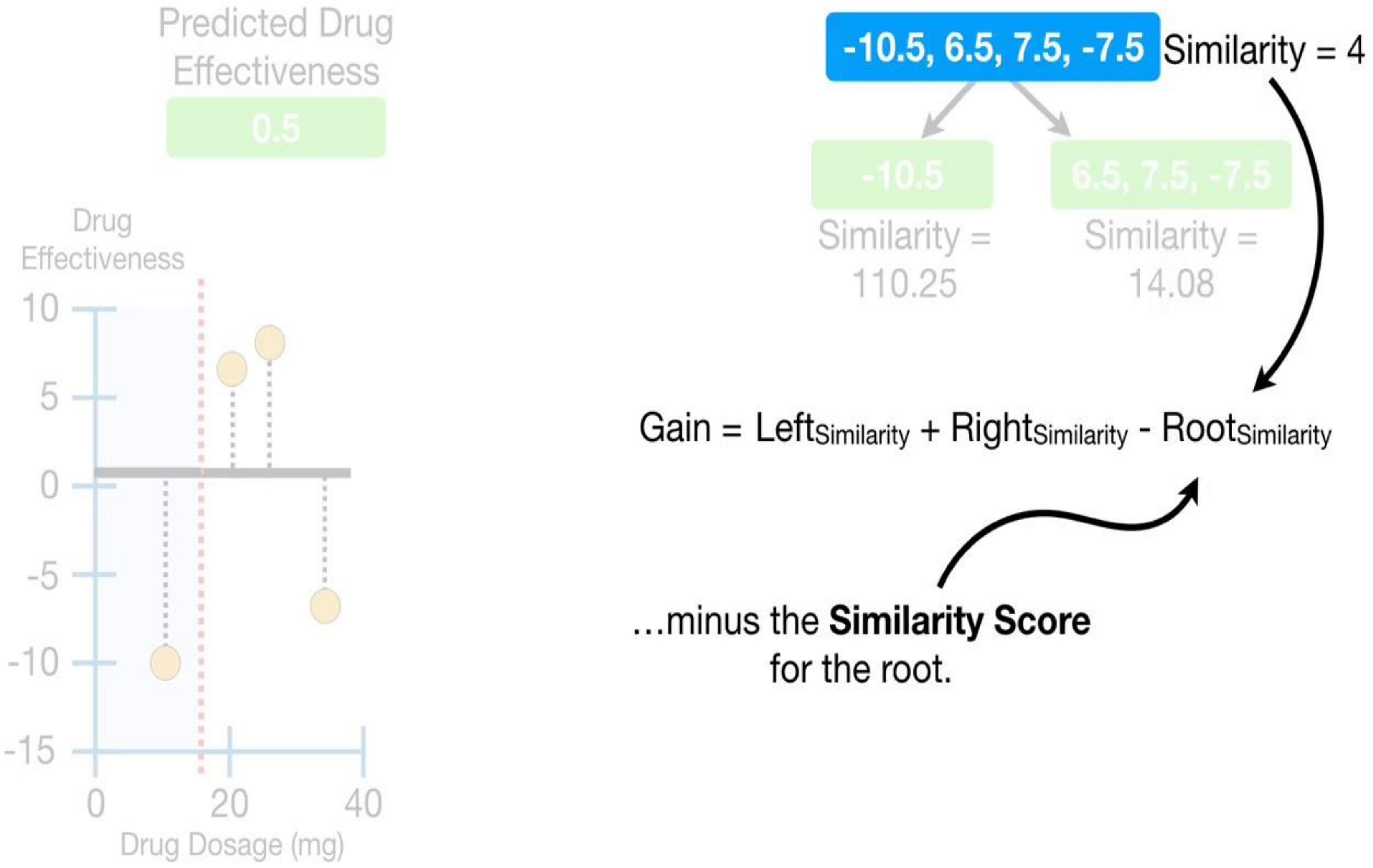
XGBoost Tree



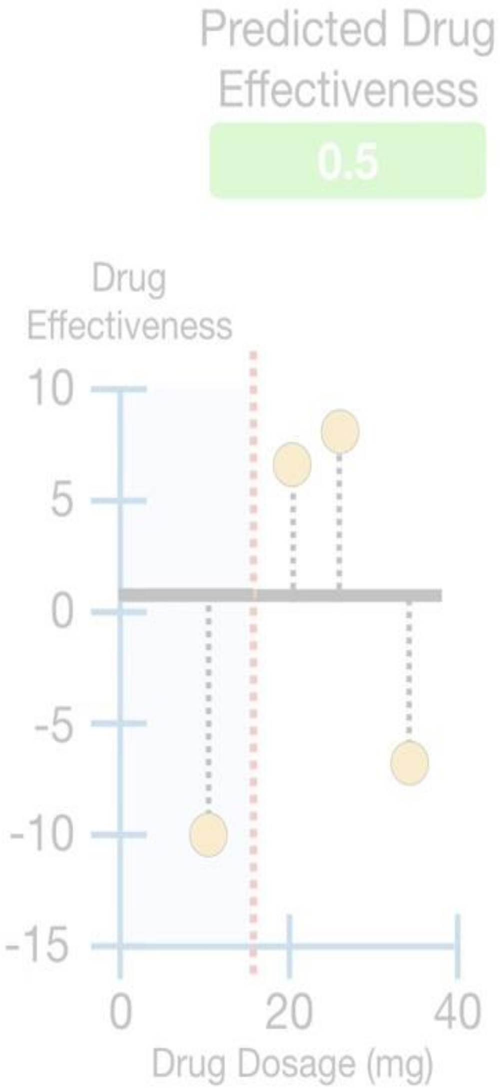
XGBoost Tree

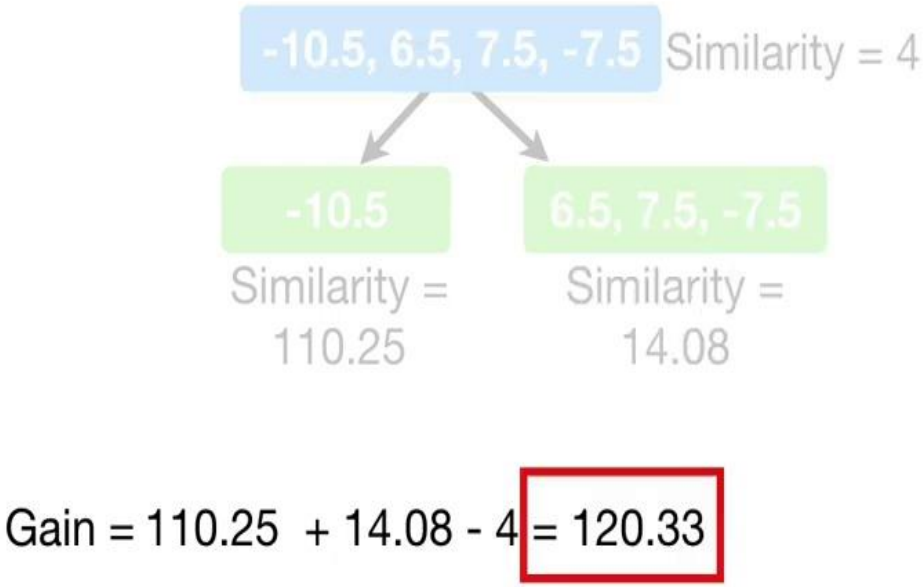


XGBoost Tree

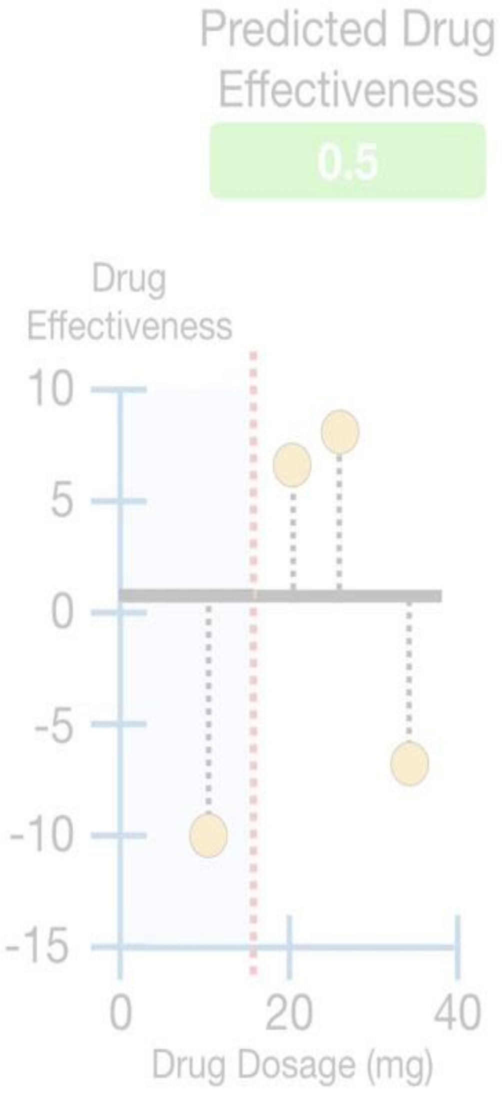


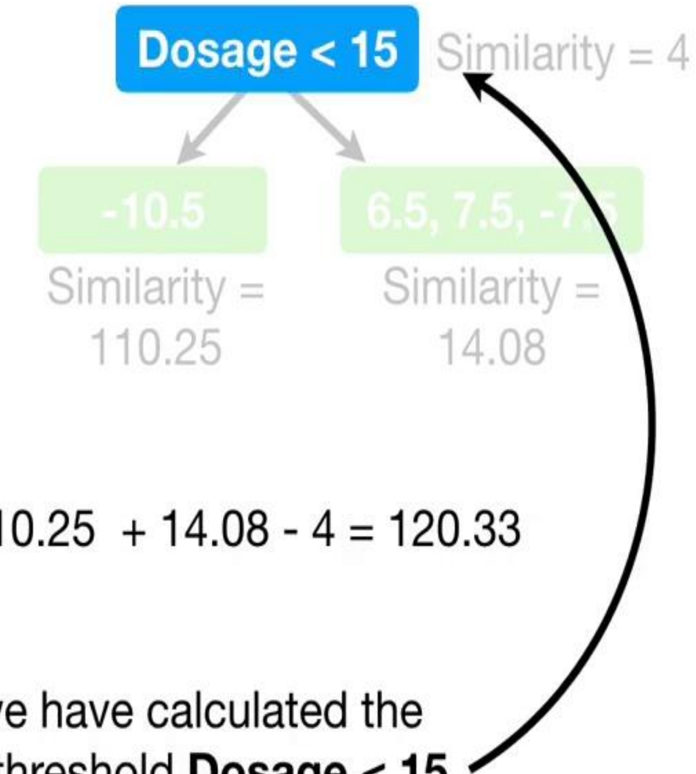
XGBoost Tree





XGBoost Tree

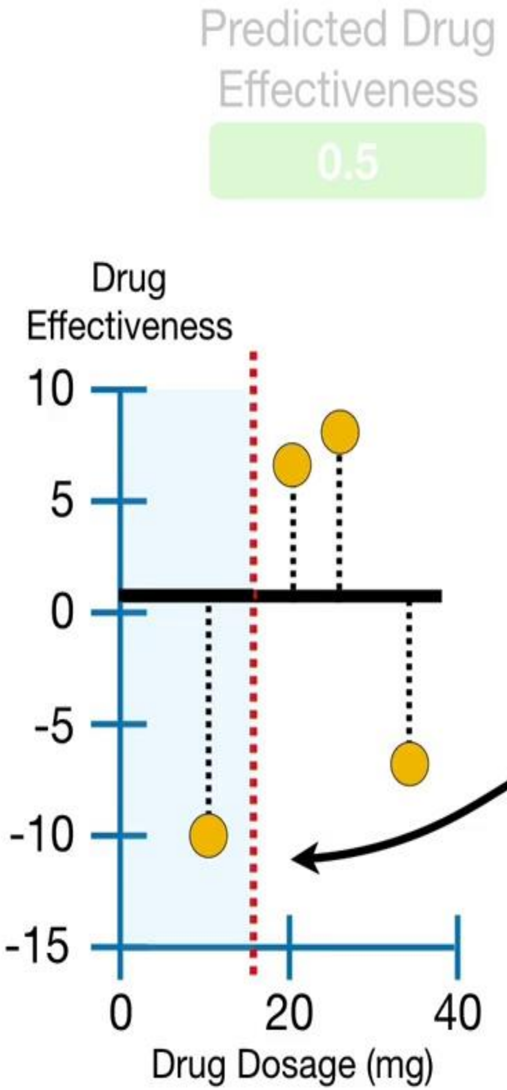




Gain = 110.25 + 14.08 - 4 = 120.33

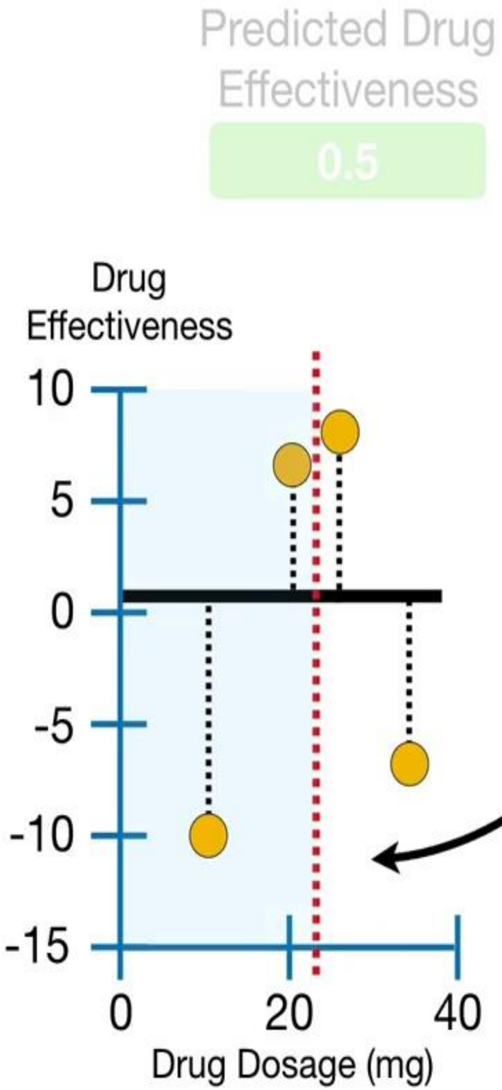
Now that we have calculated the **Gain** for the threshold **Dosage < 15**, we can compare it to the **Gain** calculated for other thresholds.

XGBoost Tree



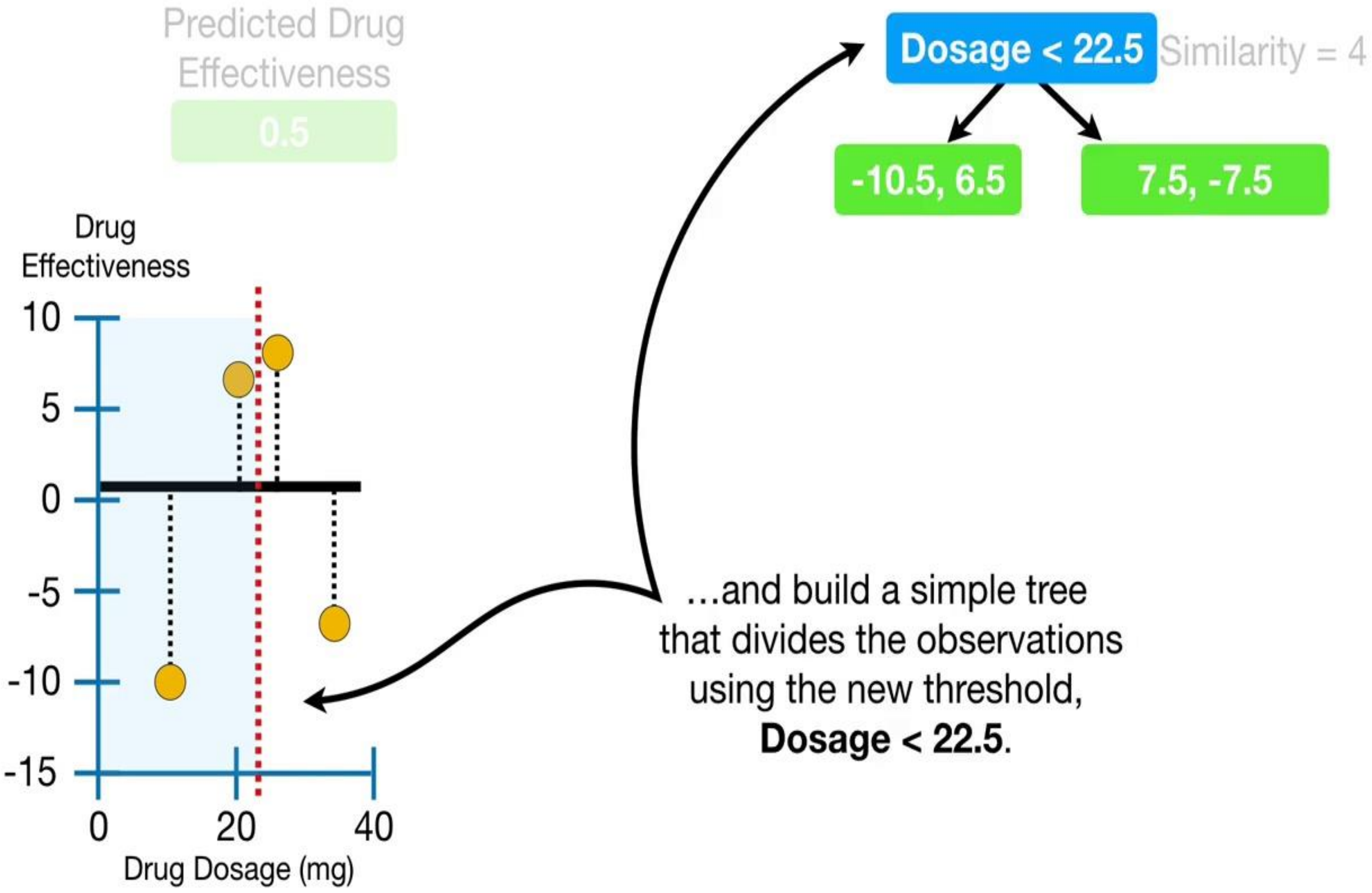
So we shift the threshold over so that it is the average of the next two observations...

XGBoost Tree

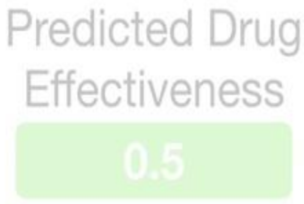


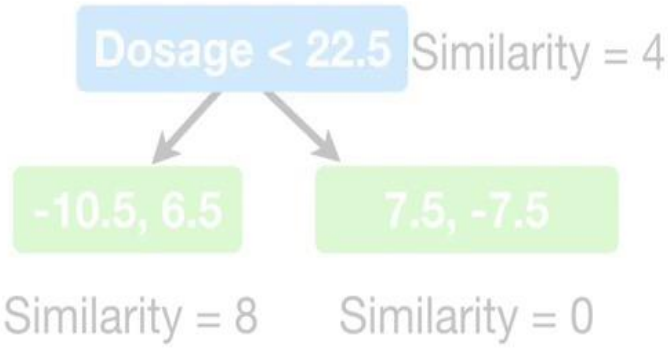
So we shift the threshold over so that it is the average of the next two observations...

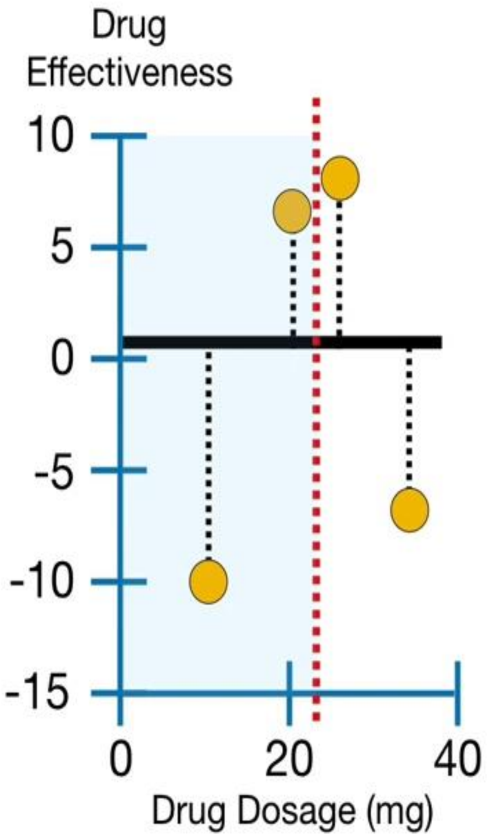
XGBoost Tree



XGBoost Tree



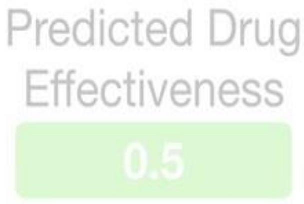


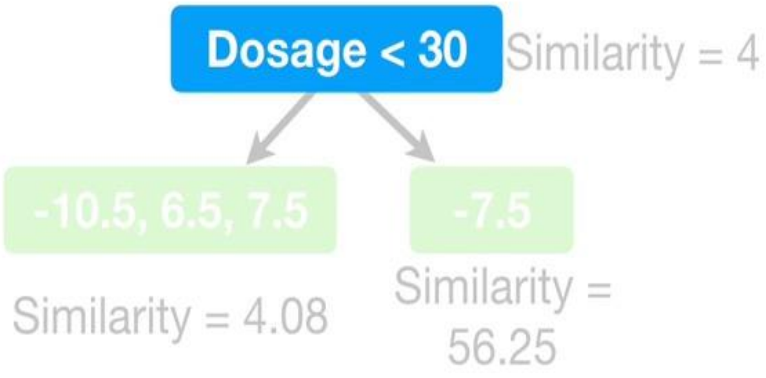


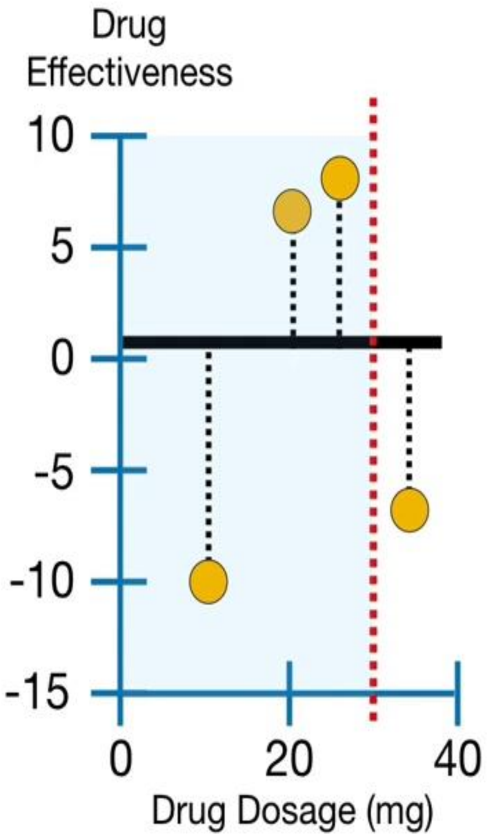
Gain = 8 + 0 - 4 = 4

Since the **Gain** for **Dosage < 22.5** (**Gain = 4**) is less than the **Gain** for **Dosage < 15** (**Gain = 120.33**), **Dosage < 15** is better at splitting the **Residuals** into clusters of similar values.

XGBoost Tree



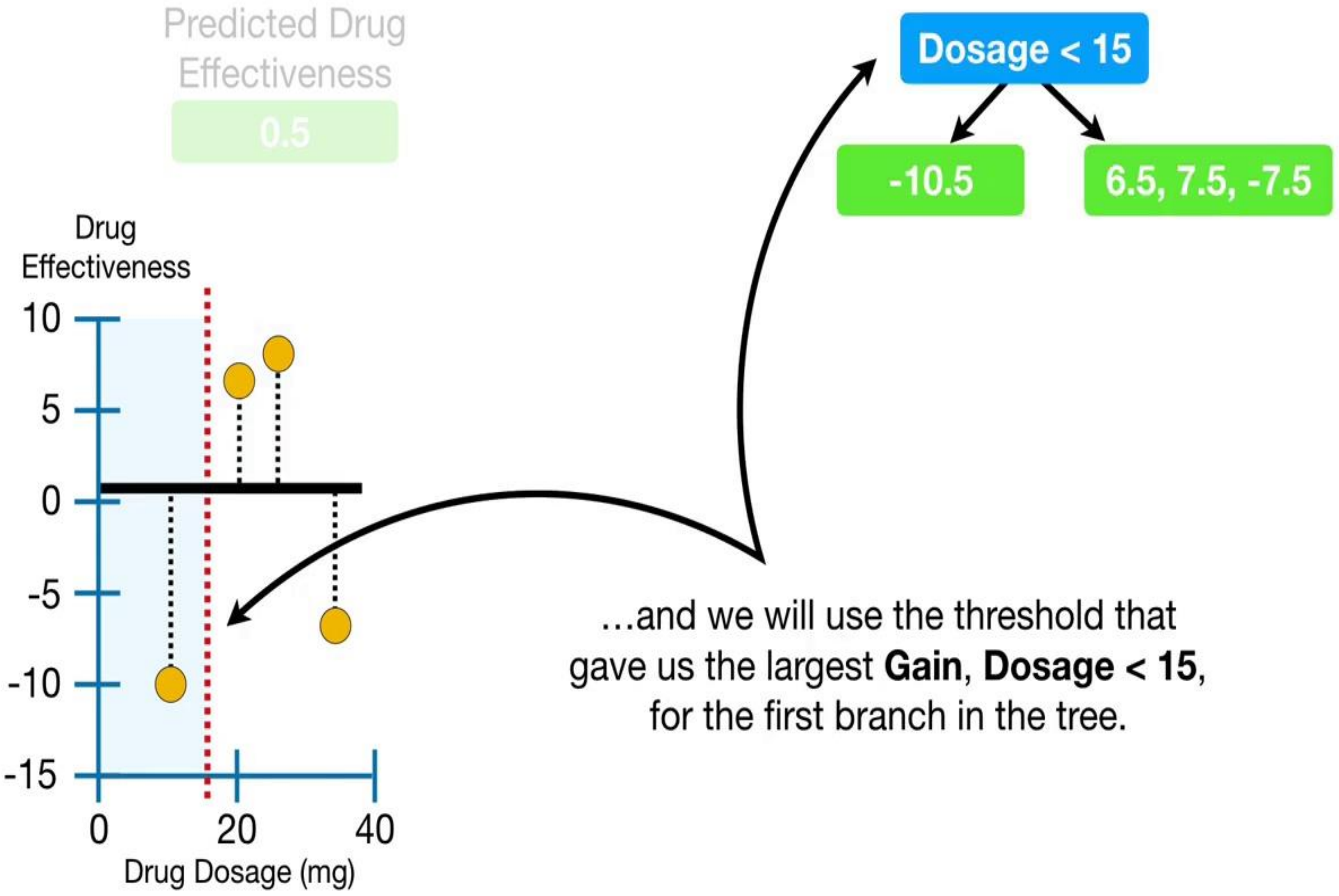




Gain = 4.08 + 56.25 - 4 = 56.33

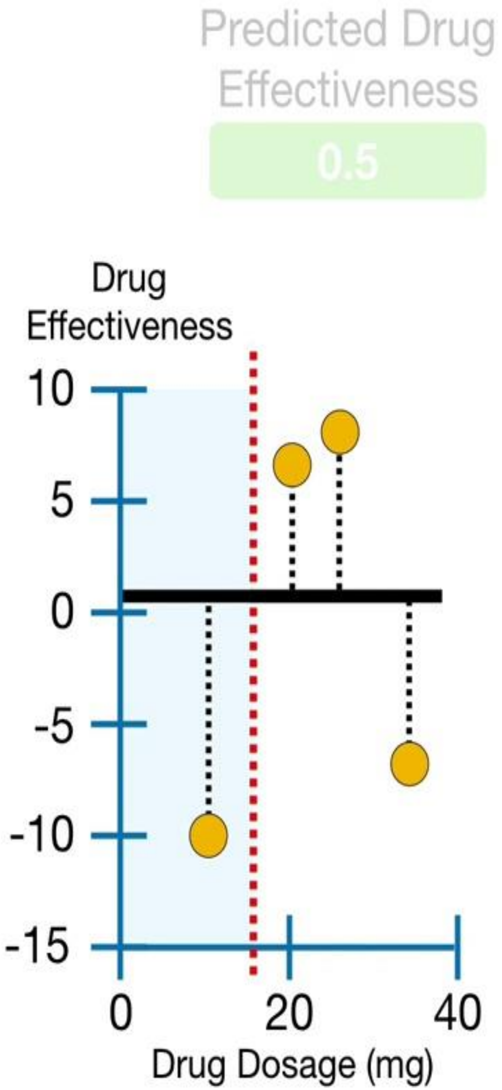
Again, since the **Gain** for **Dosage < 30** (**Gain = 56.33**) is less than the **Gain** for **Dosage < 15** (**Gain = 120.33**), **Dosage < 15** is better at splitting the observations.

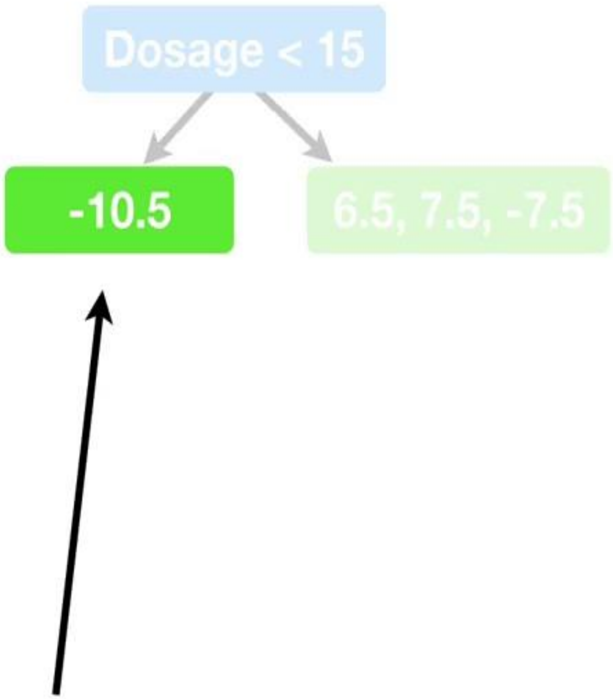
XGBoost Tree



...and we will use the threshold that gave us the largest **Gain**, **Dosage < 15**, for the first branch in the tree.

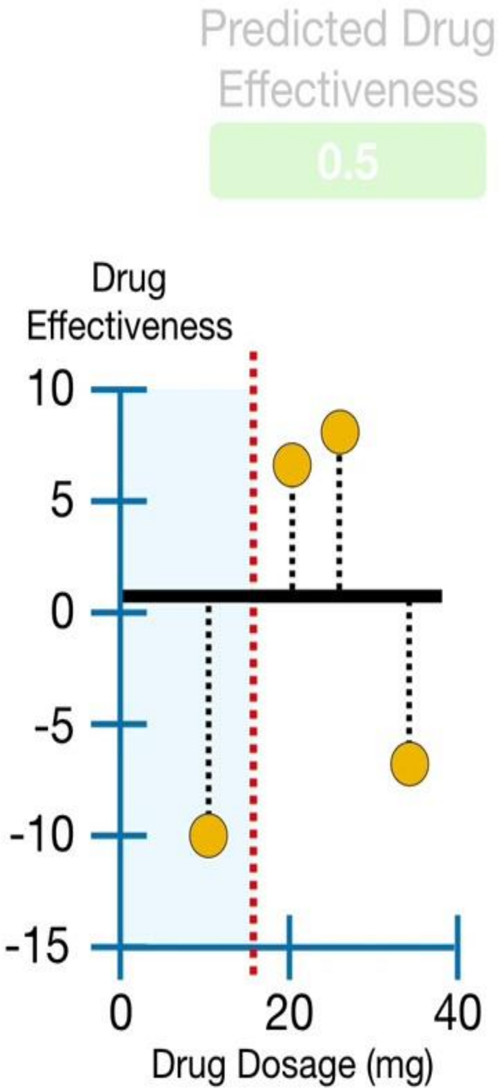
XGBoost Tree





Now, since there is only one **Residual** in the leaf on the left, we can't split it any further.

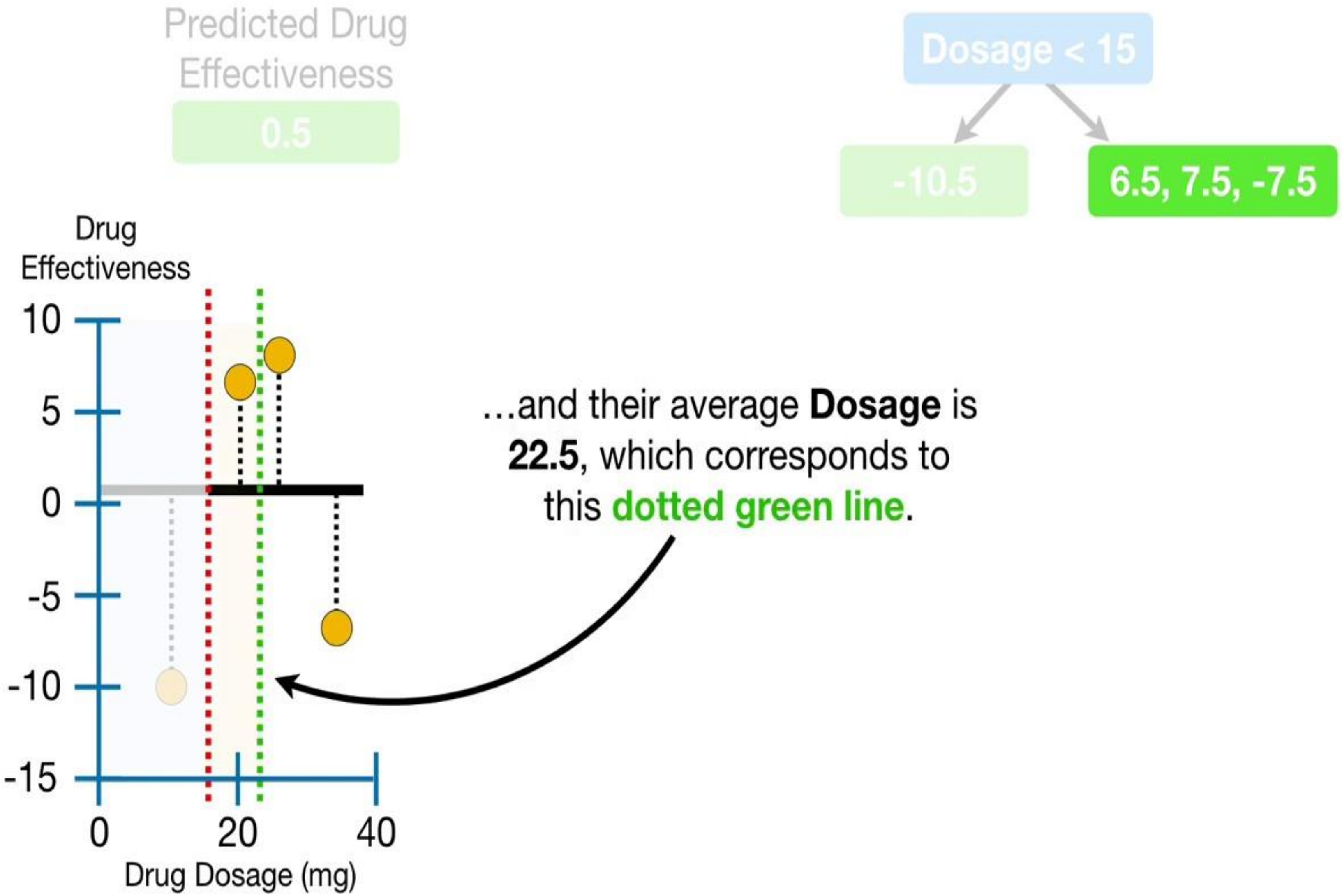
XGBoost Tree



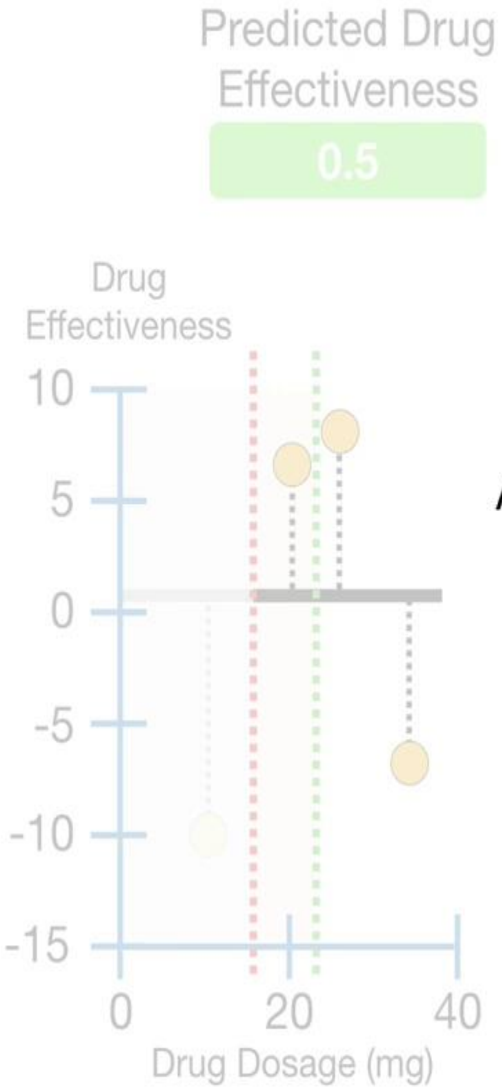
However, we can split the **3 Residuals** in the leaf on the right.

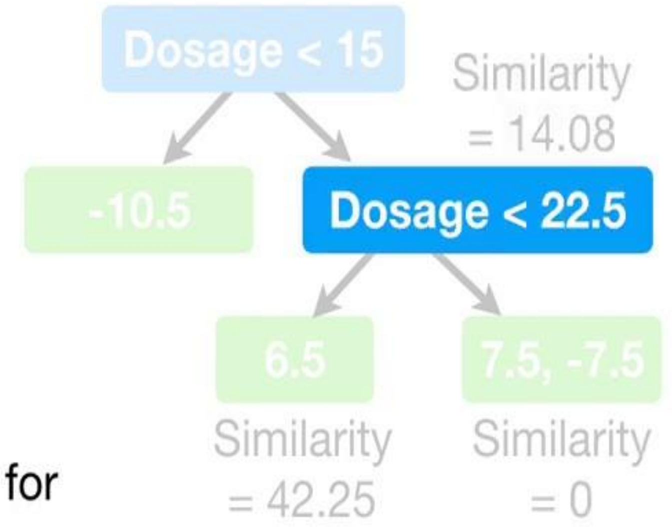
XGBoost Tree

XGBoost Tree



XGBoost Tree

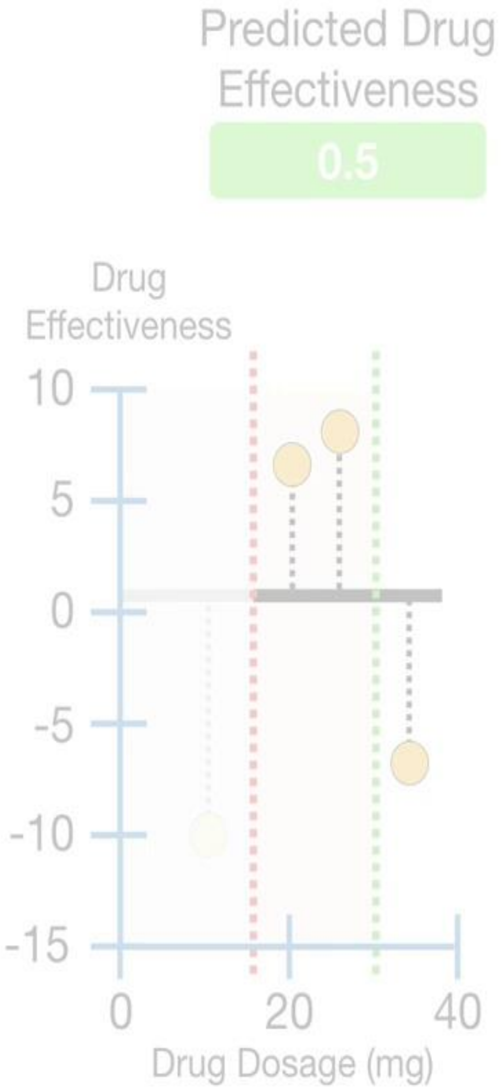


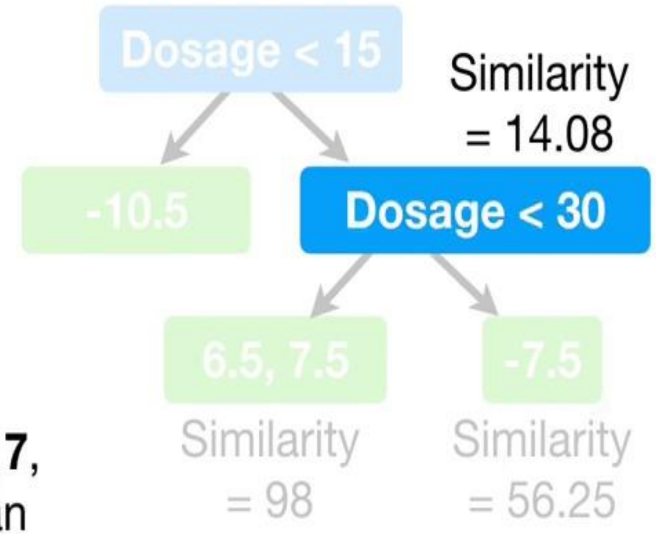


And we get **Gain = 28.17** for when the threshold is **Dosage < 22.5**.

Gain = 42.25 + 0 - 14.08 = 28.17

XGBoost Tree

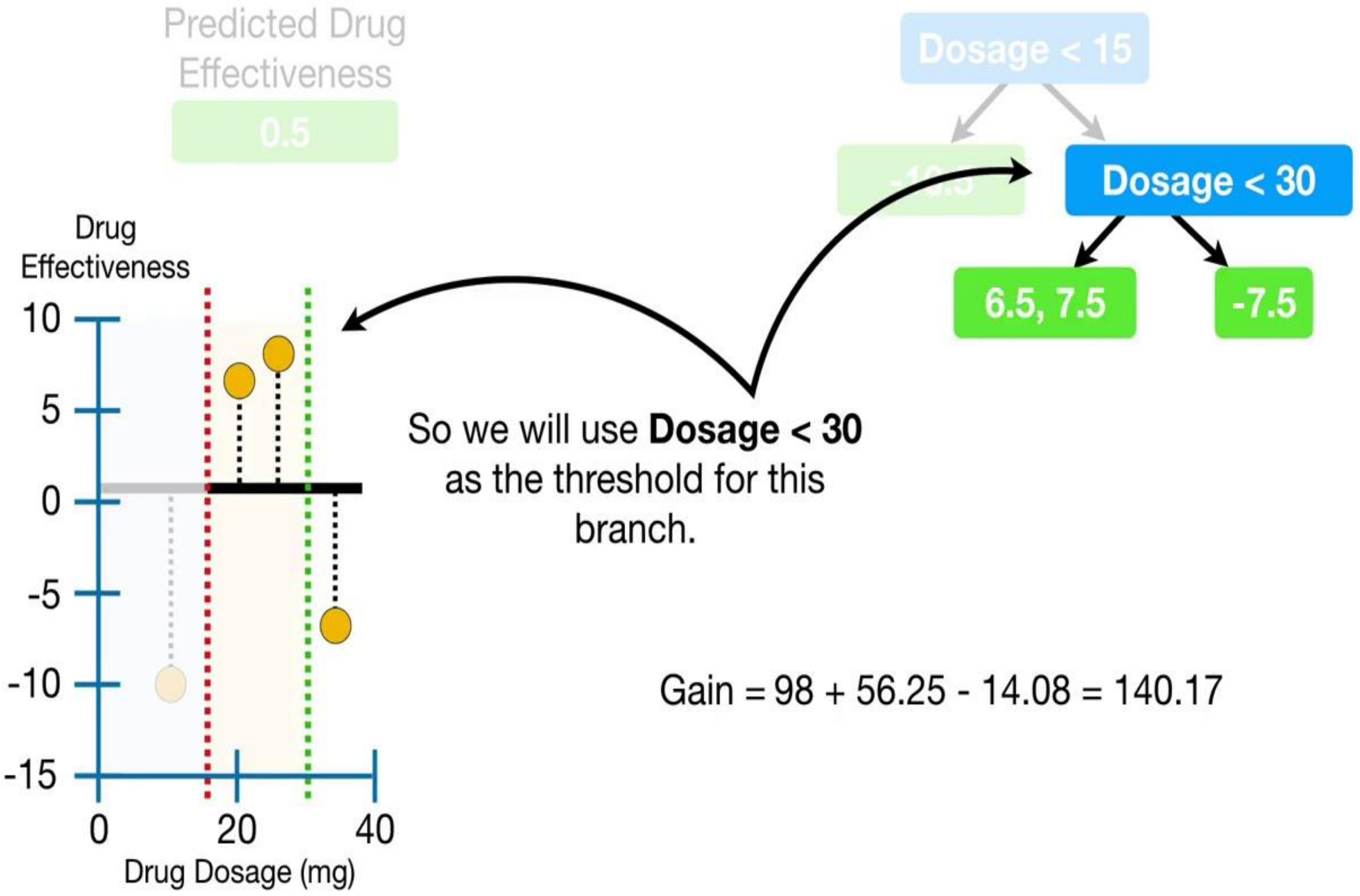




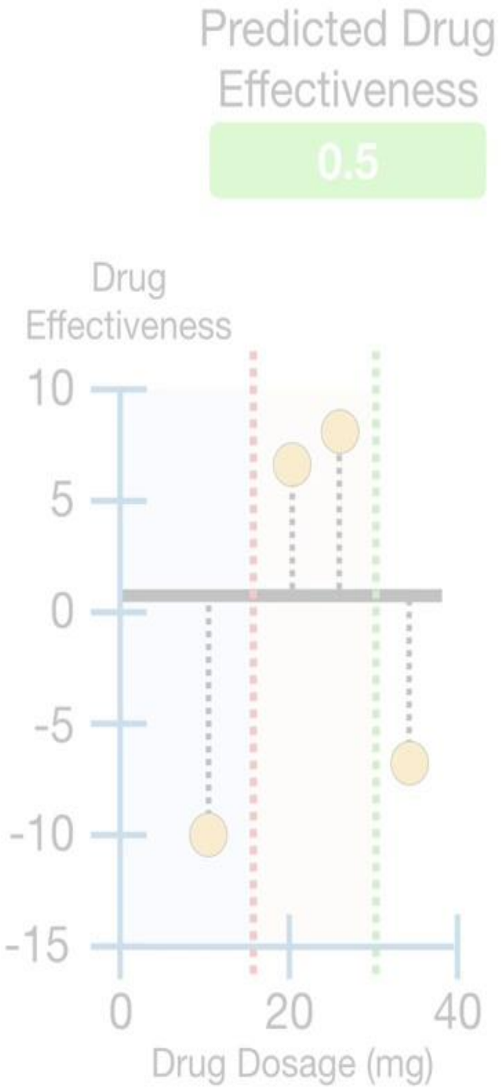
And we get **Gain = 140.17**, which is much larger than **28.17**, when the threshold was **Dosage < 22.5**.

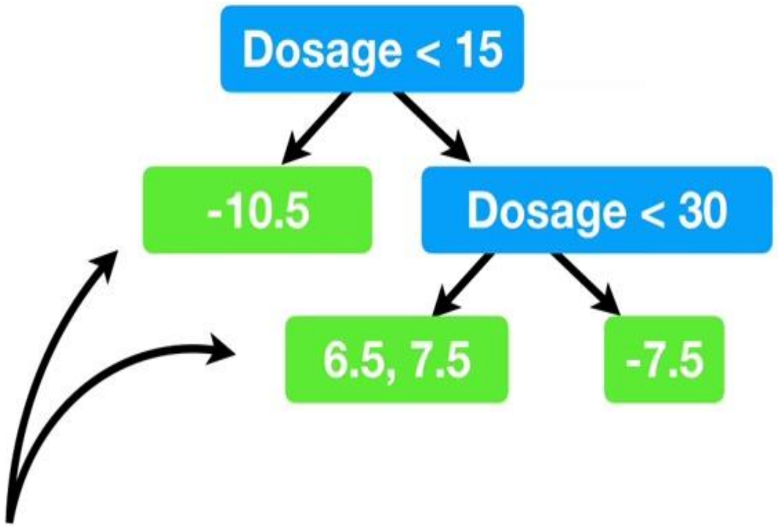
Gain = 98 + 56.25 - 14.08 = 140.17

XGBoost Tree



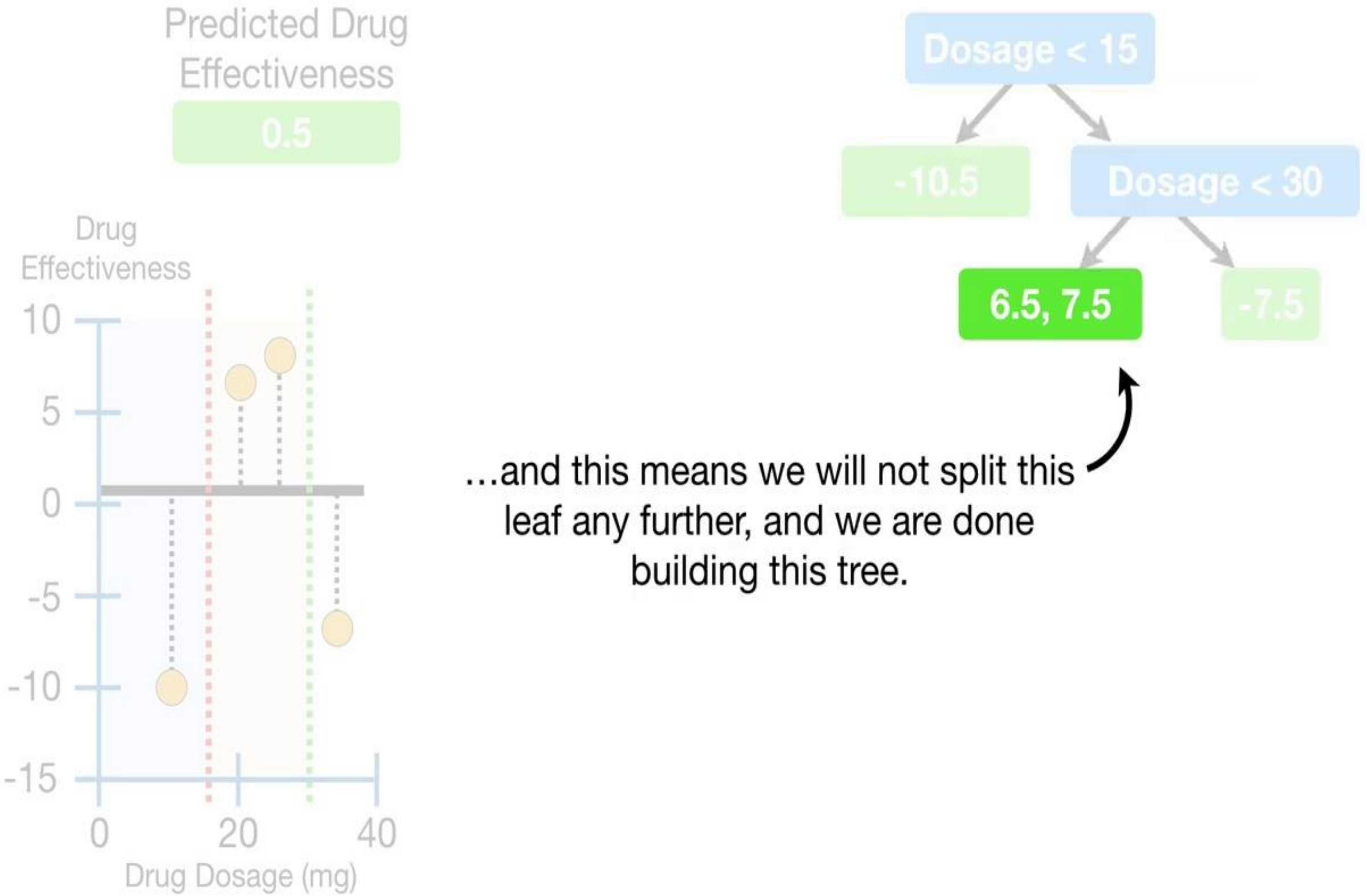
XGBoost Tree





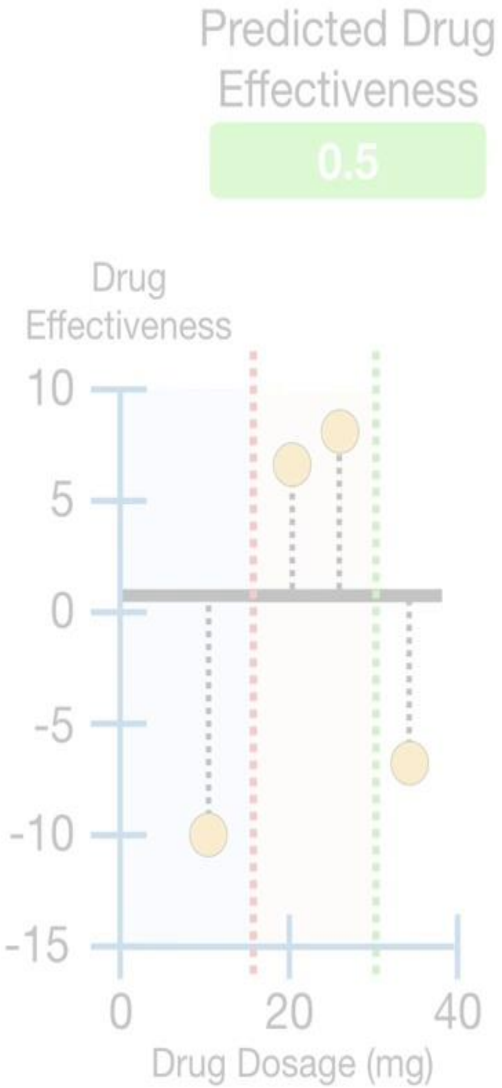
NOTE: To keep this example from getting out of hand, I've limited the tree depth to two levels...

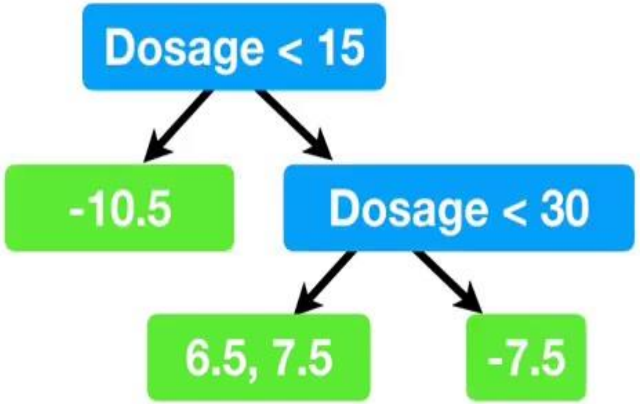
XGBoost Tree



...and this means we will not split this leaf any further, and we are done building this tree.

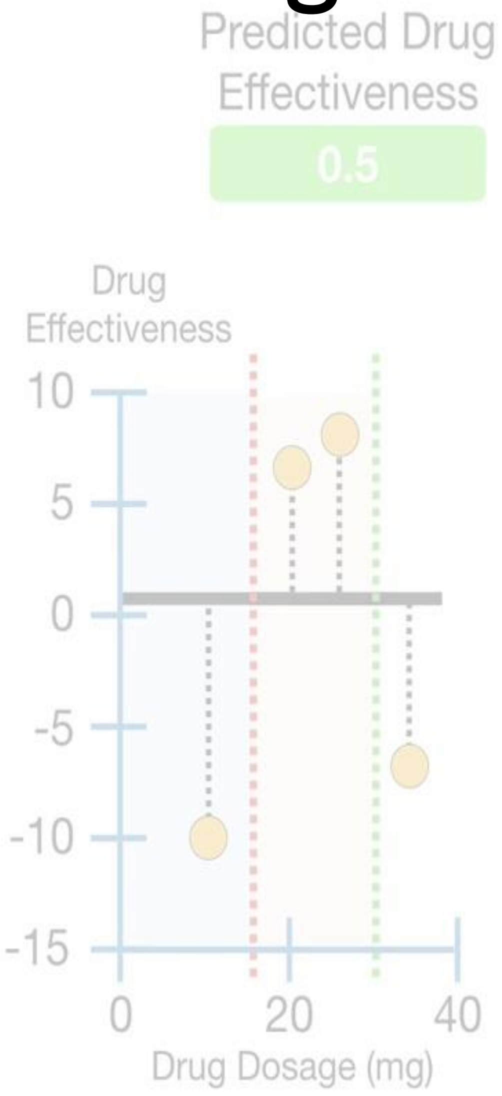
XGBoost Tree

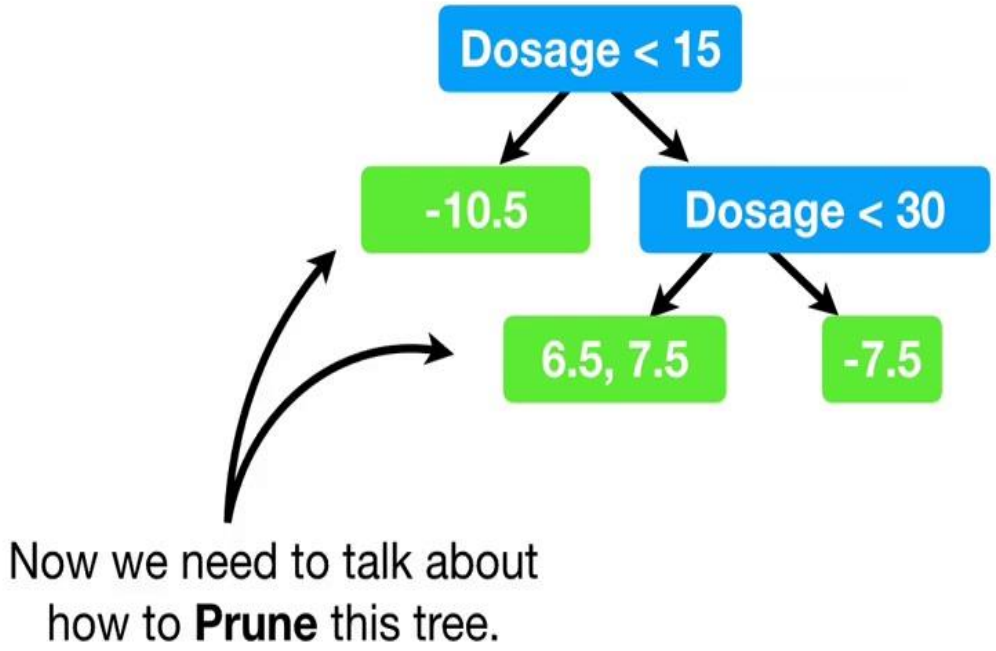




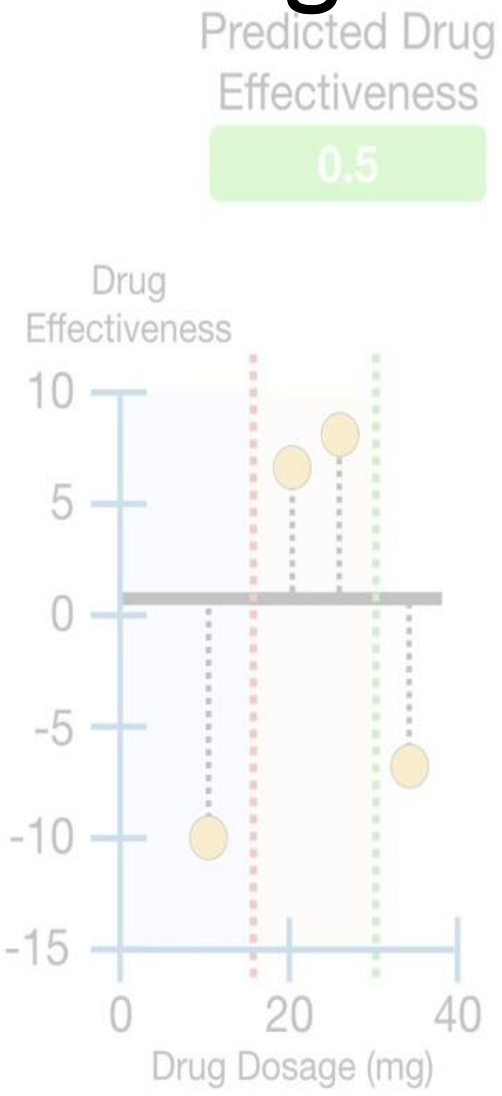
However, the default is to allow up to **6** levels.

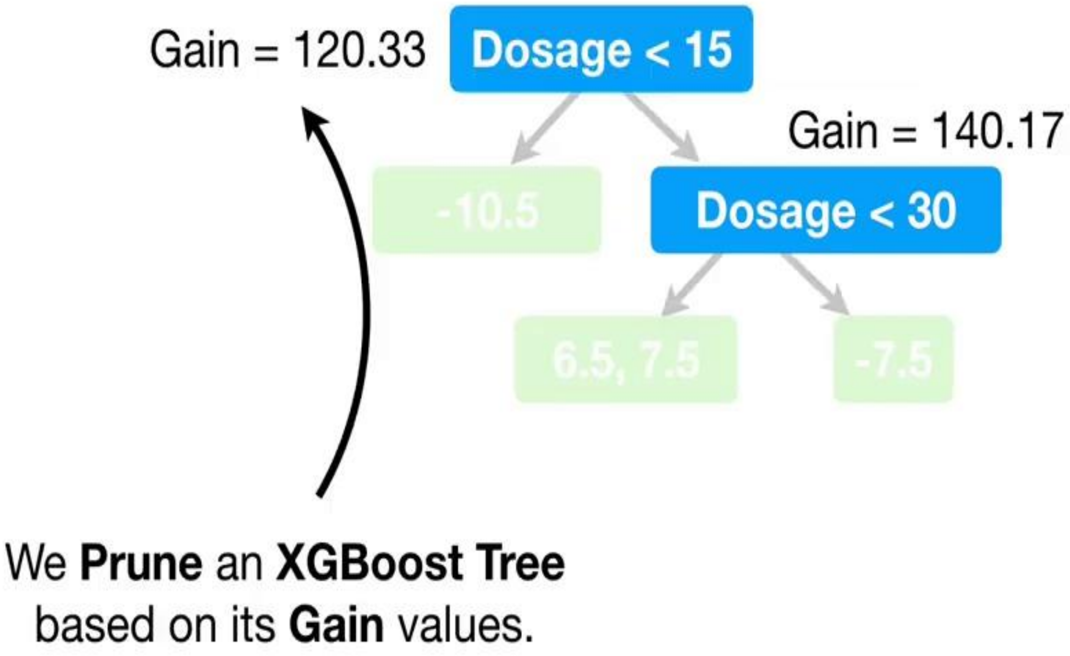
Tree Pruning





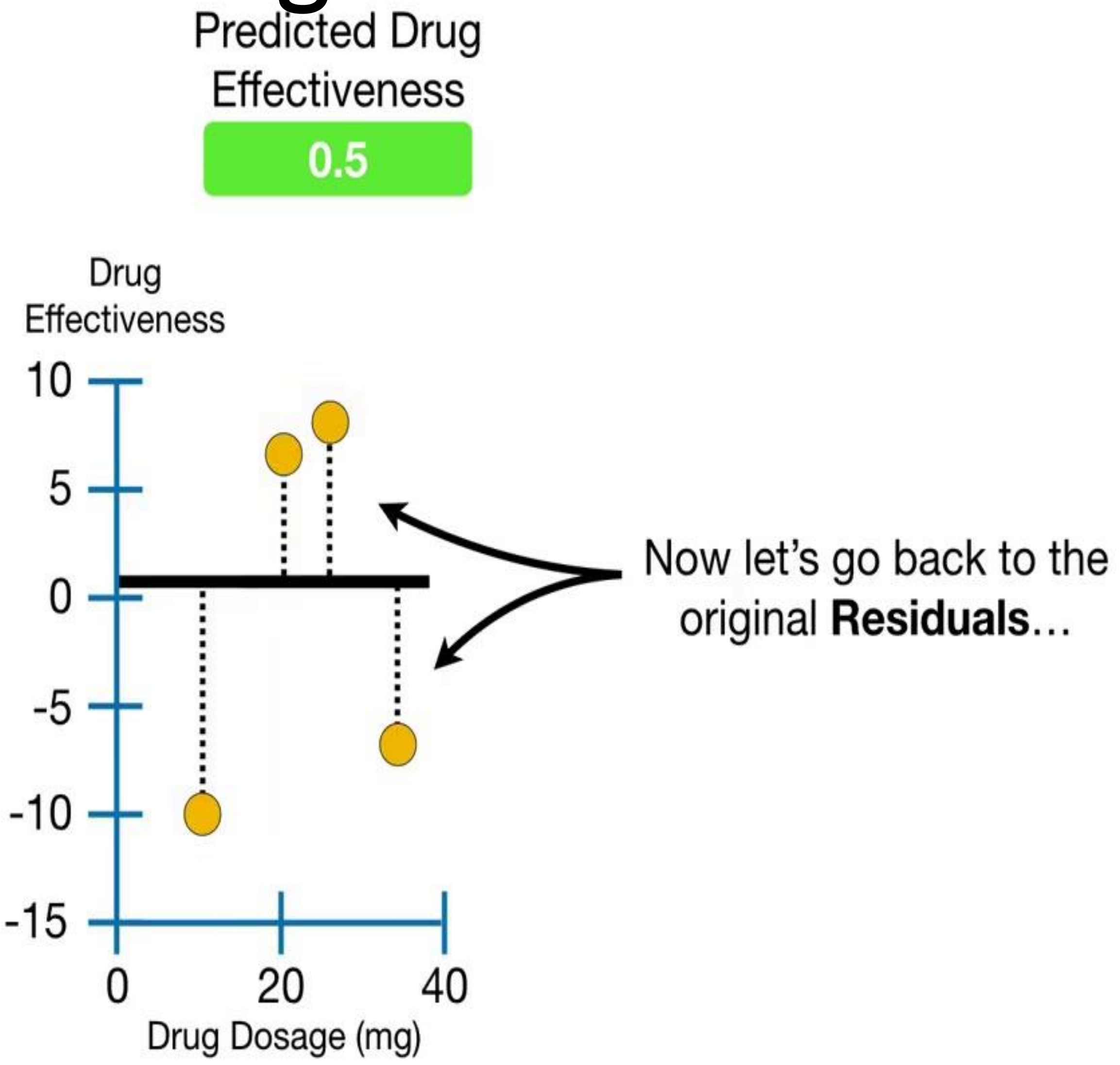
Tree Pruning



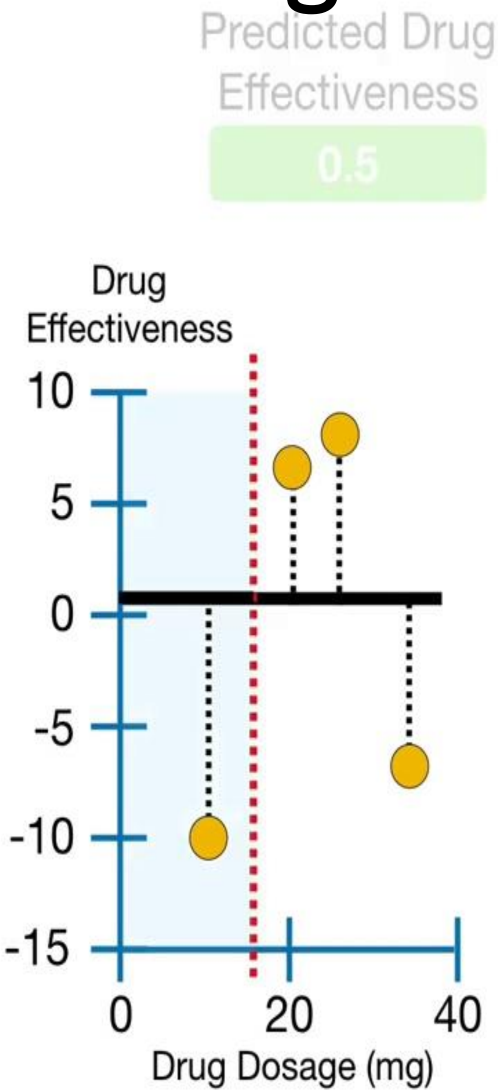


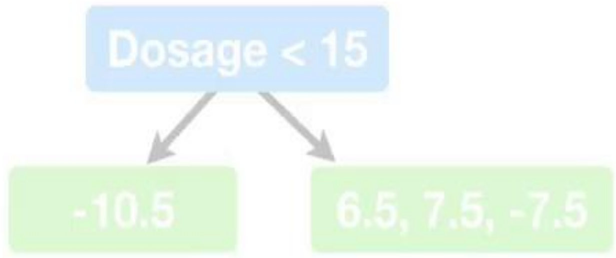
We set a threshold parameter **gamma**.
Then *we cut leaves with a gain less than* **gamma**.

Tree Pruning



Tree Pruning

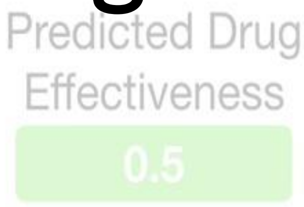


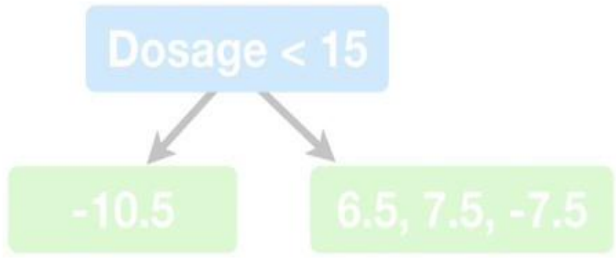


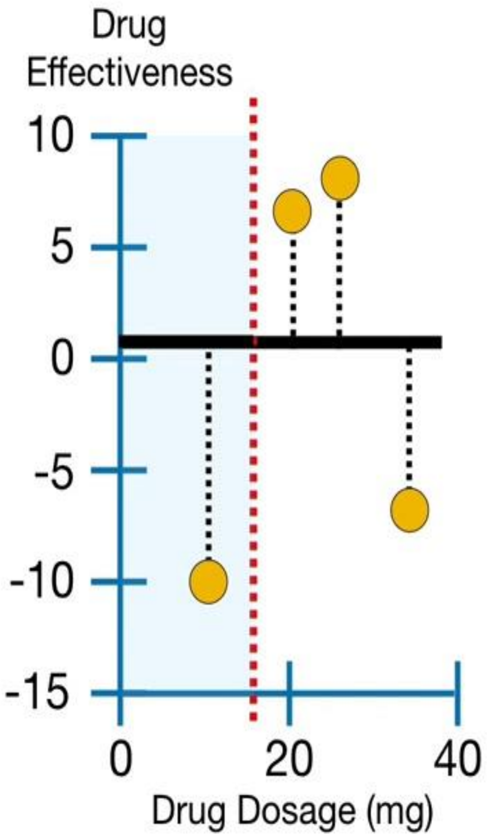
$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

...only this time, when we calculate **Similarity Scores**, we will set **λ (lambda) = 1**.

Tree Pruning



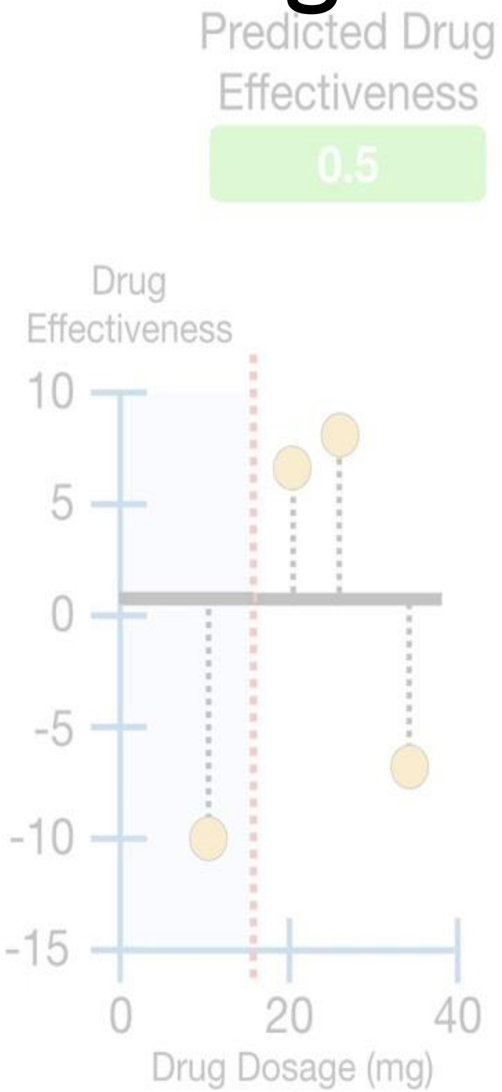


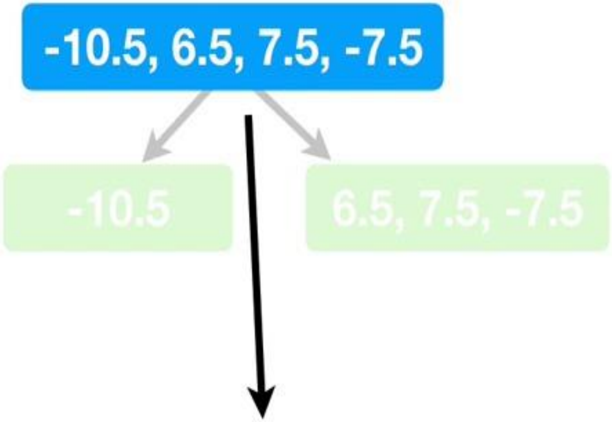


$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + 1}$$

Remember **λ (lambda)** is a **Regularization Parameter**, which means that it is intended to reduce the prediction's sensitivity to individual observations.

Tree Pruning

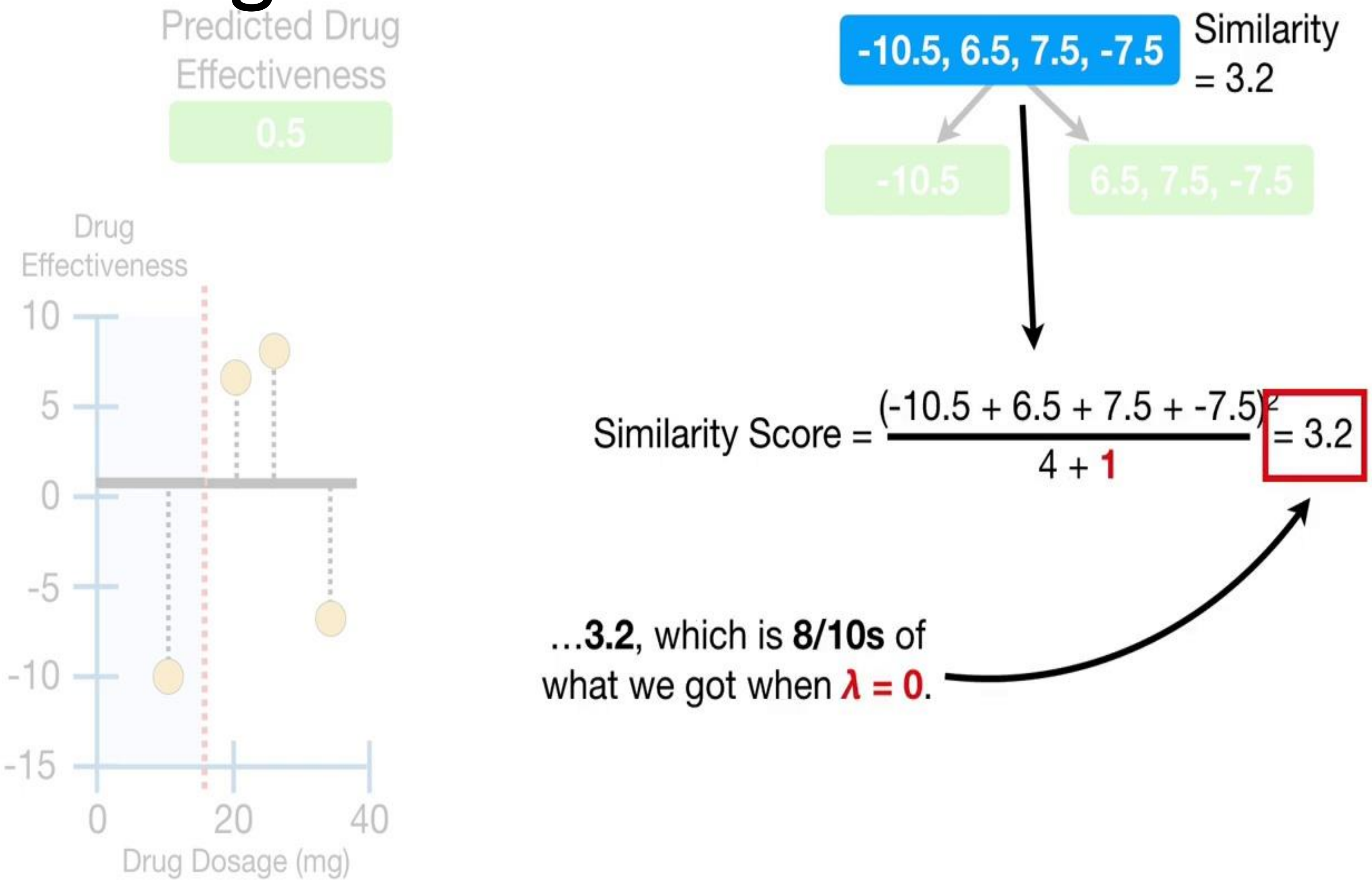




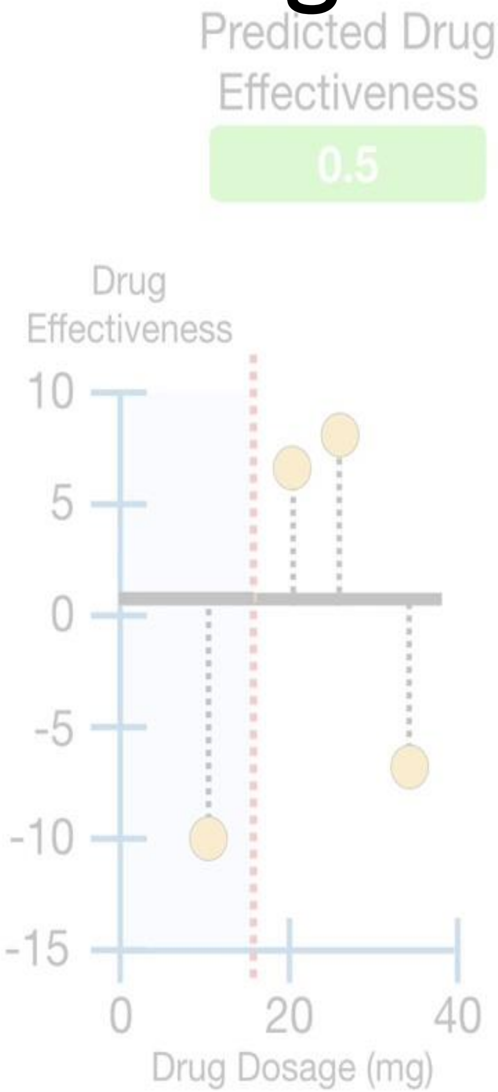
$$\text{Similarity Score} = \frac{(-10.5 + 6.5 + 7.5 + -7.5)^2}{4 + 1}$$

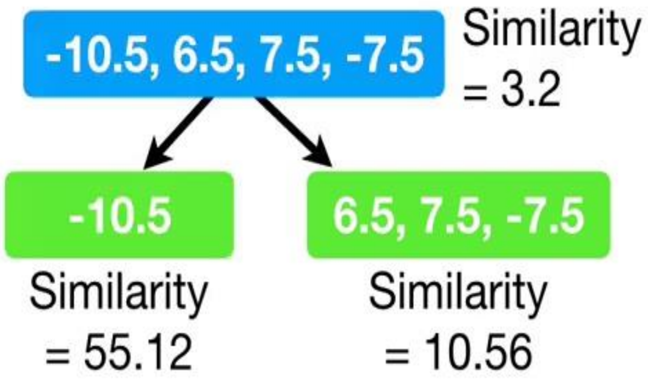
Now the **Similarity Score** for the root is…

Tree Pruning



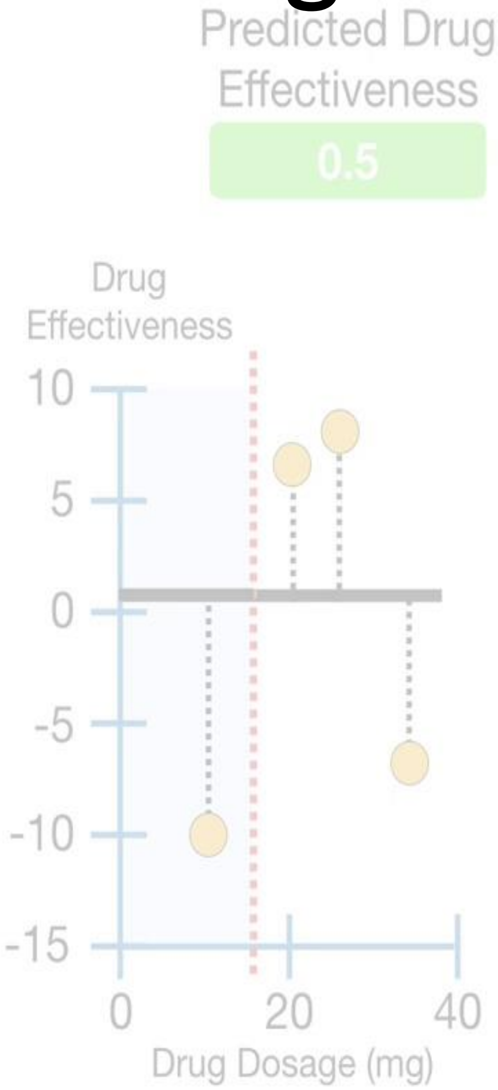
Tree Pruning

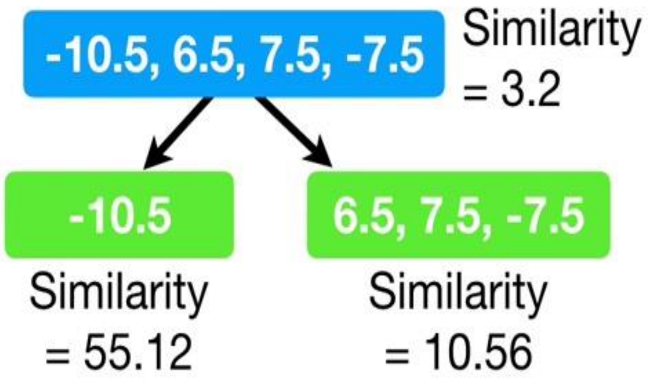




So, one thing we see is that when **λ > 0**, the **Similarity Scores** are smaller…

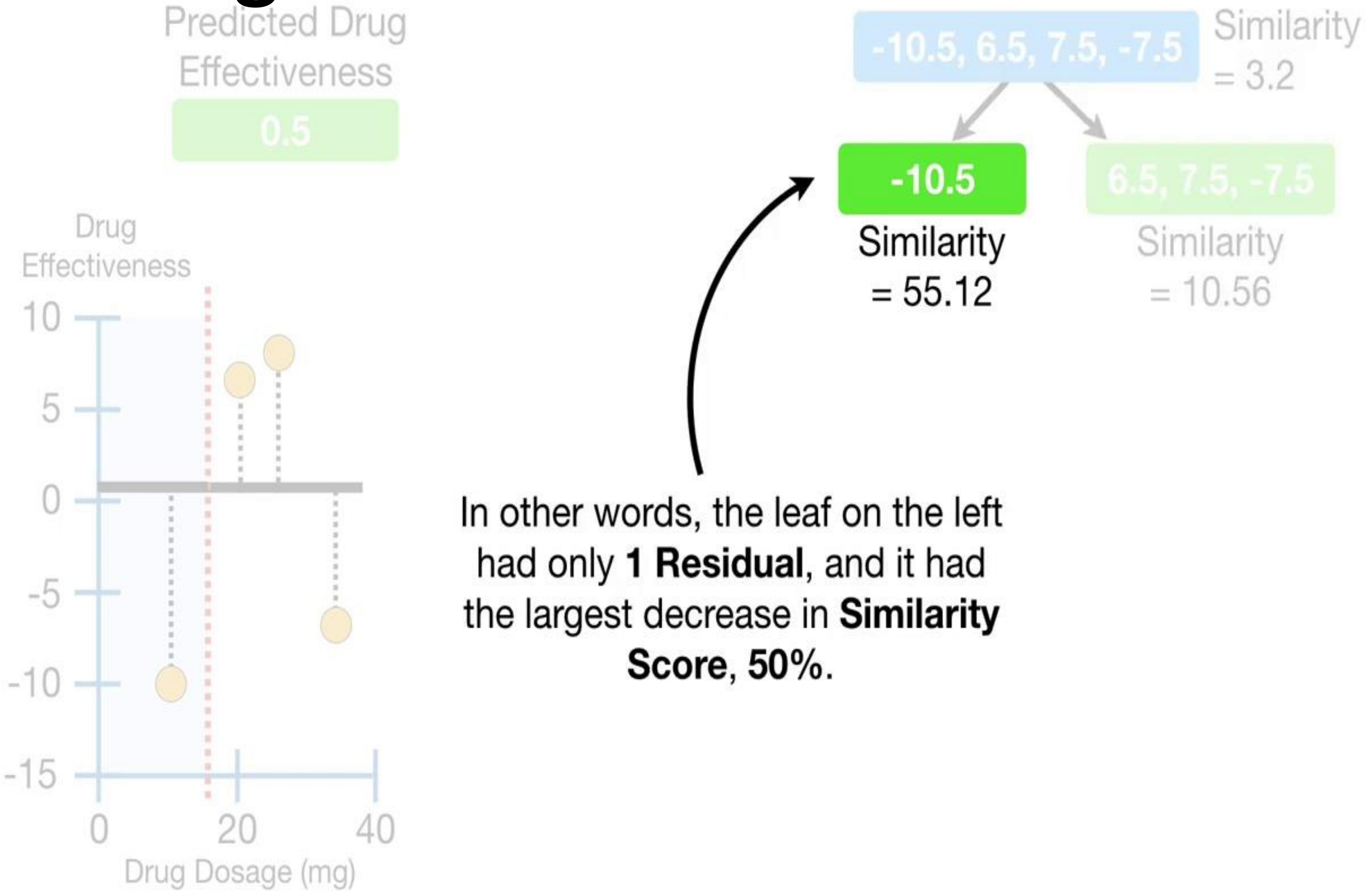
Tree Pruning





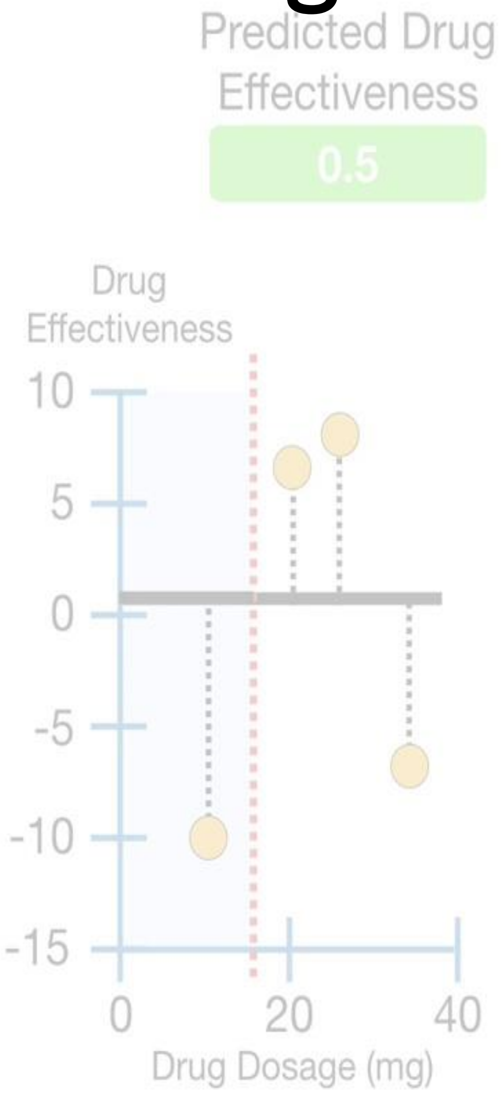
…and the amount of decrease is **inversely proportional** to the number of **Residuals** in the node.

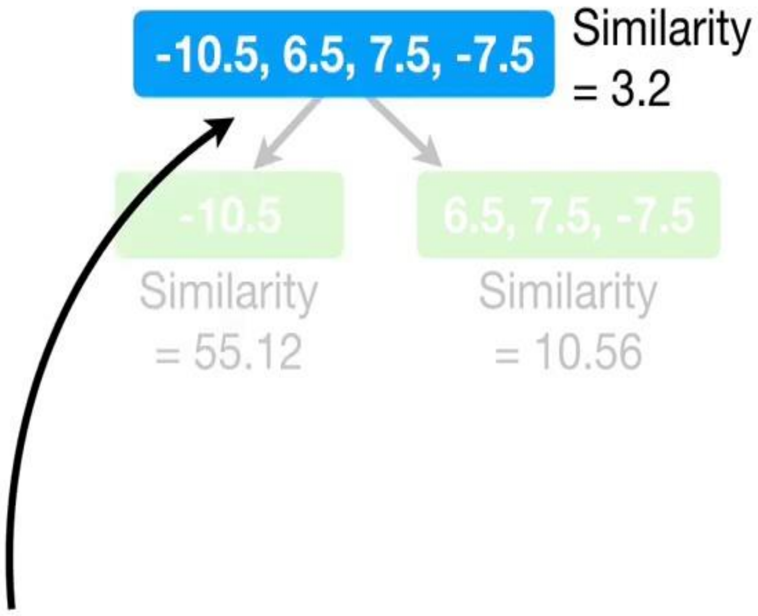
Tree Pruning



In other words, the leaf on the left had only **1 Residual**, and it had the largest decrease in **Similarity Score, 50%**.

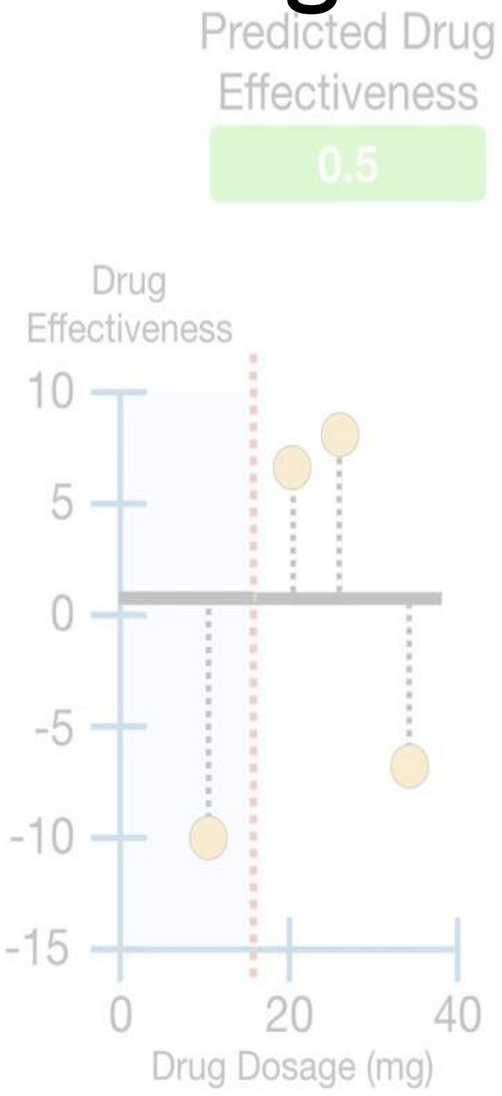
Tree Pruning

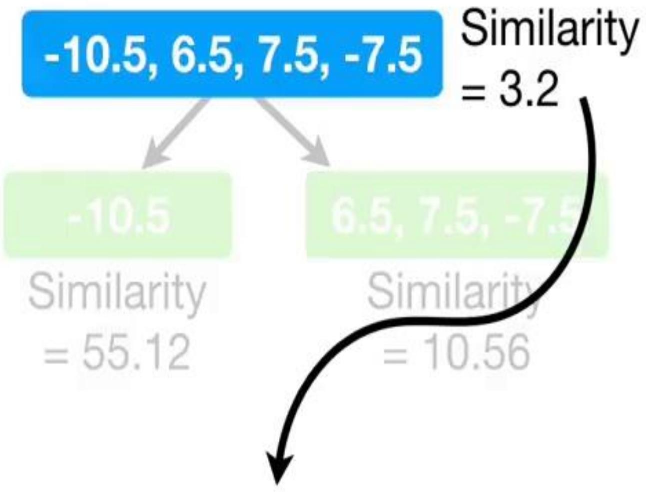




In contrast, the root had all **4 Residuals** and the smallest decrease, **20%**.

Tree Pruning

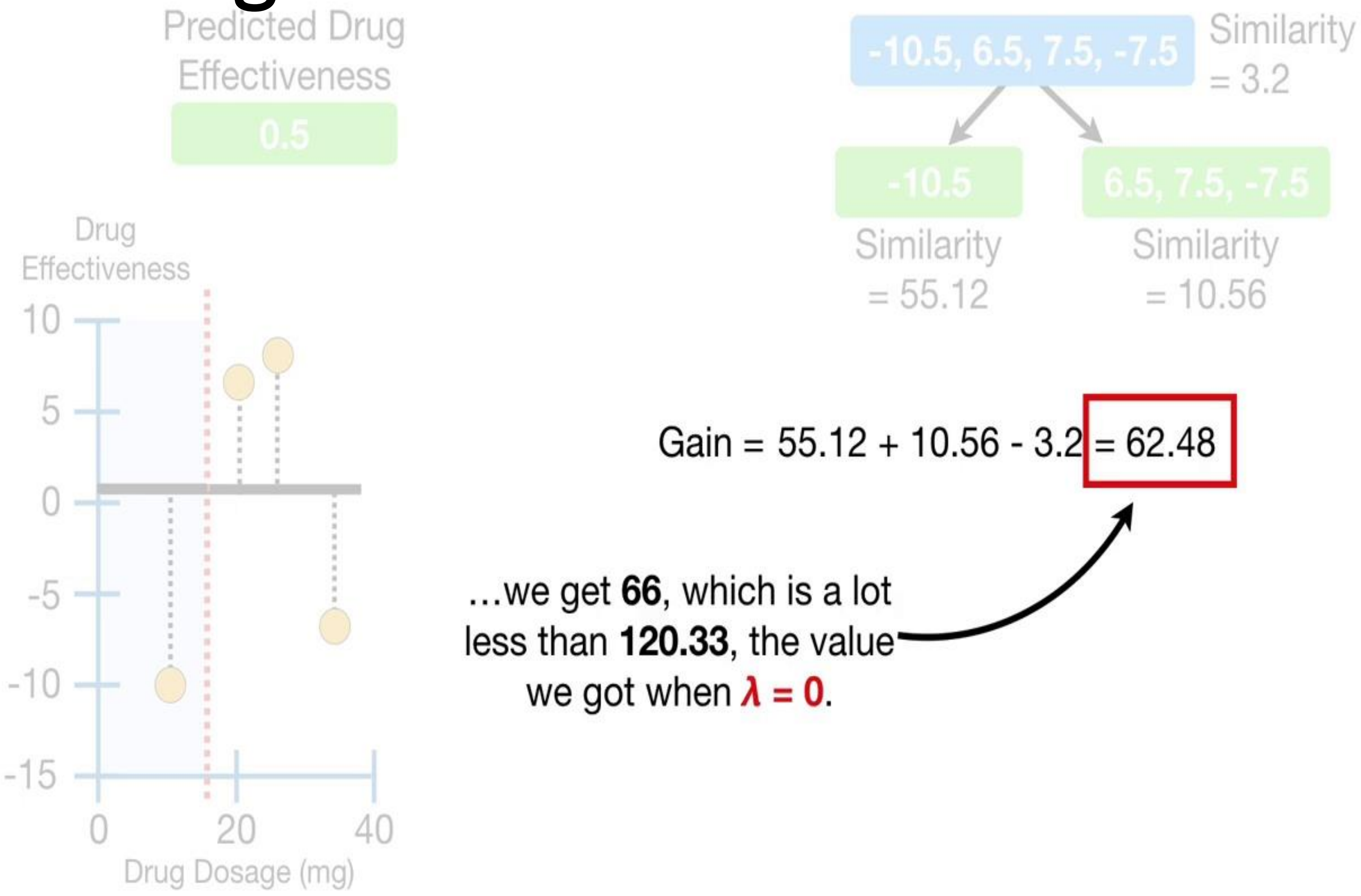




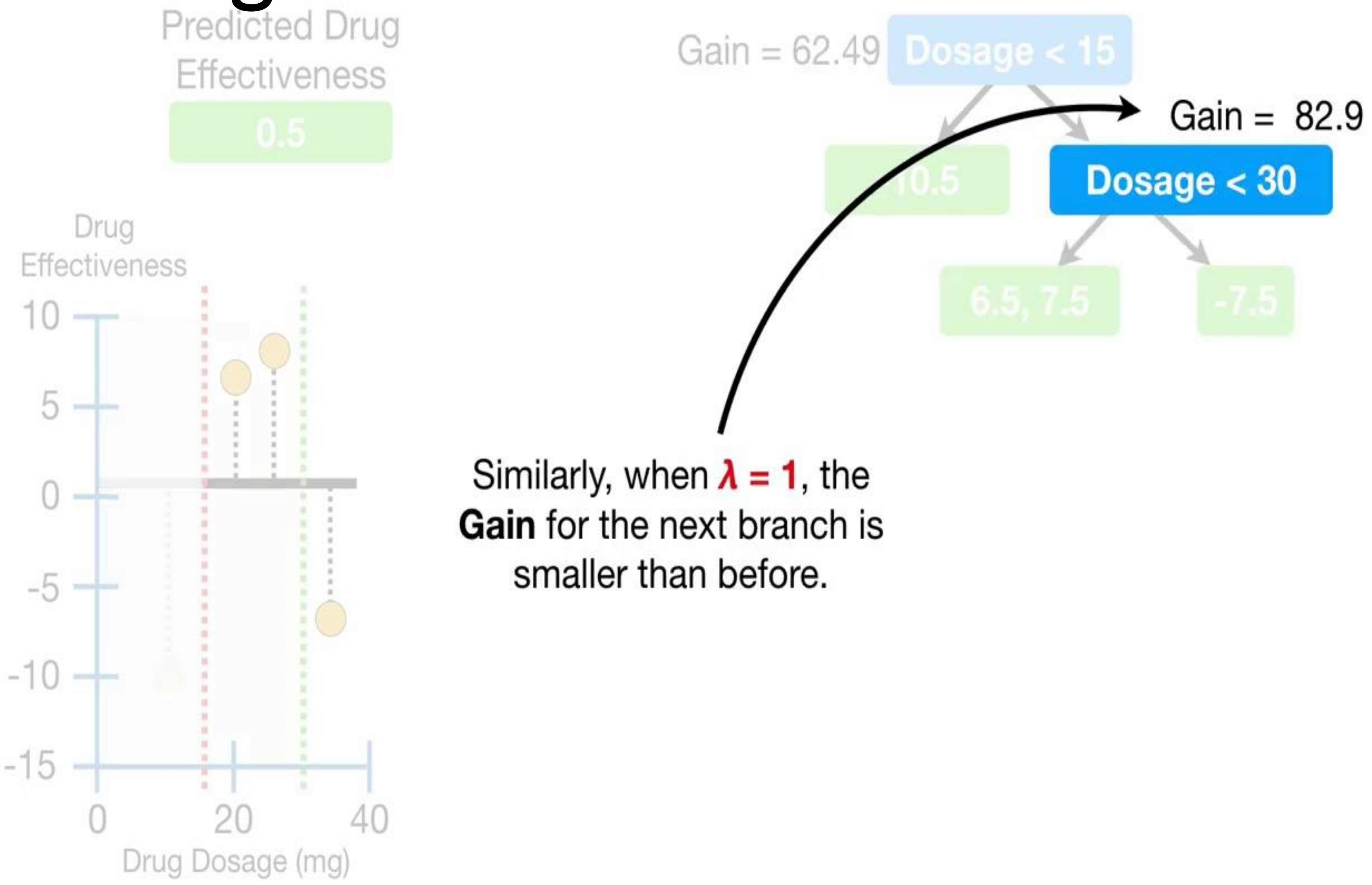
$$\text{Gain} = 55.12 + 10.56 - \text{Root}_{\text{Similarity}}$$

Now when we calculate the **Gain**…

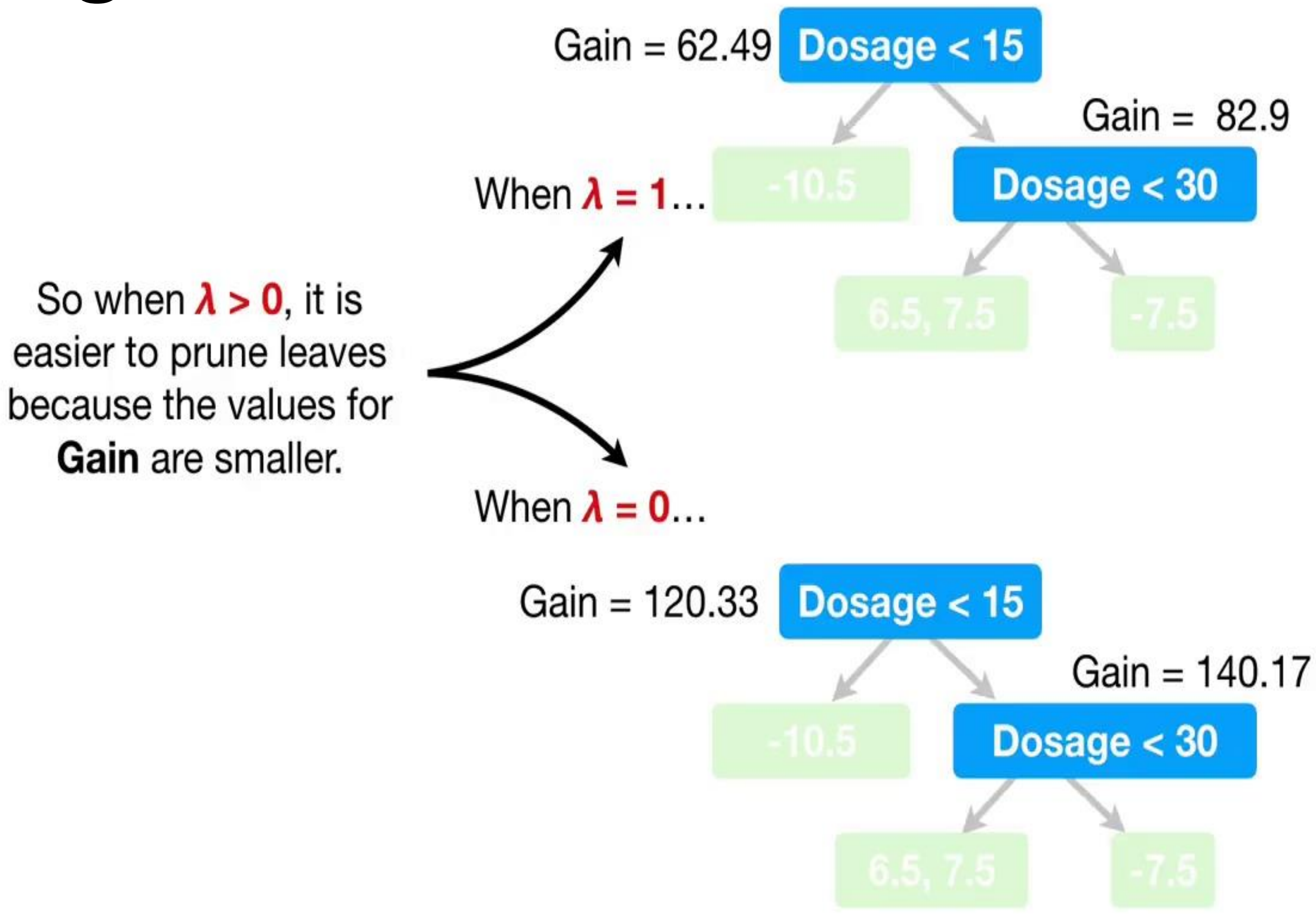
Tree Pruning



Tree Pruning

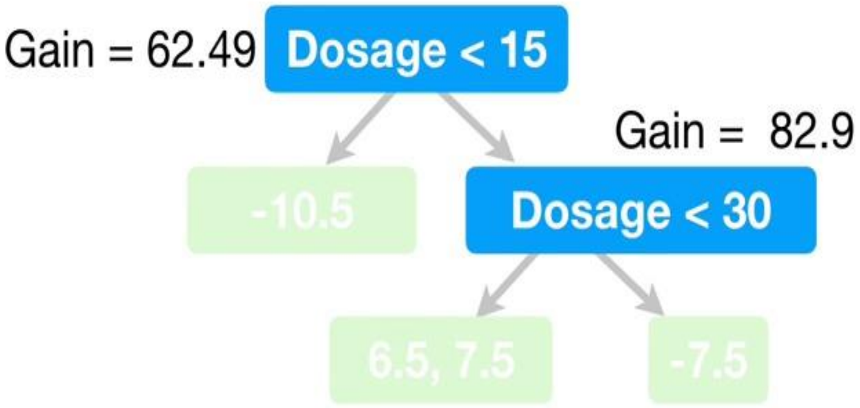


Tree Pruning

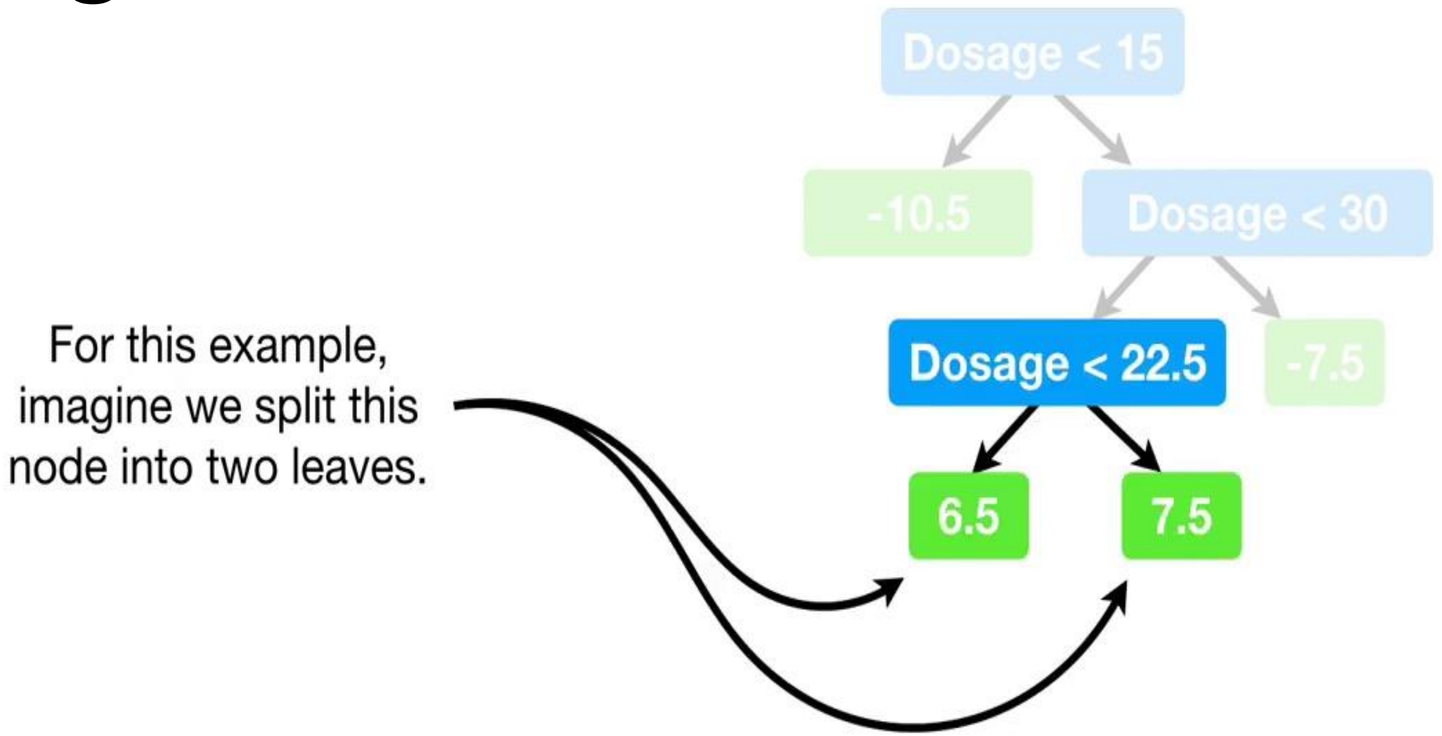


Tree Pruning

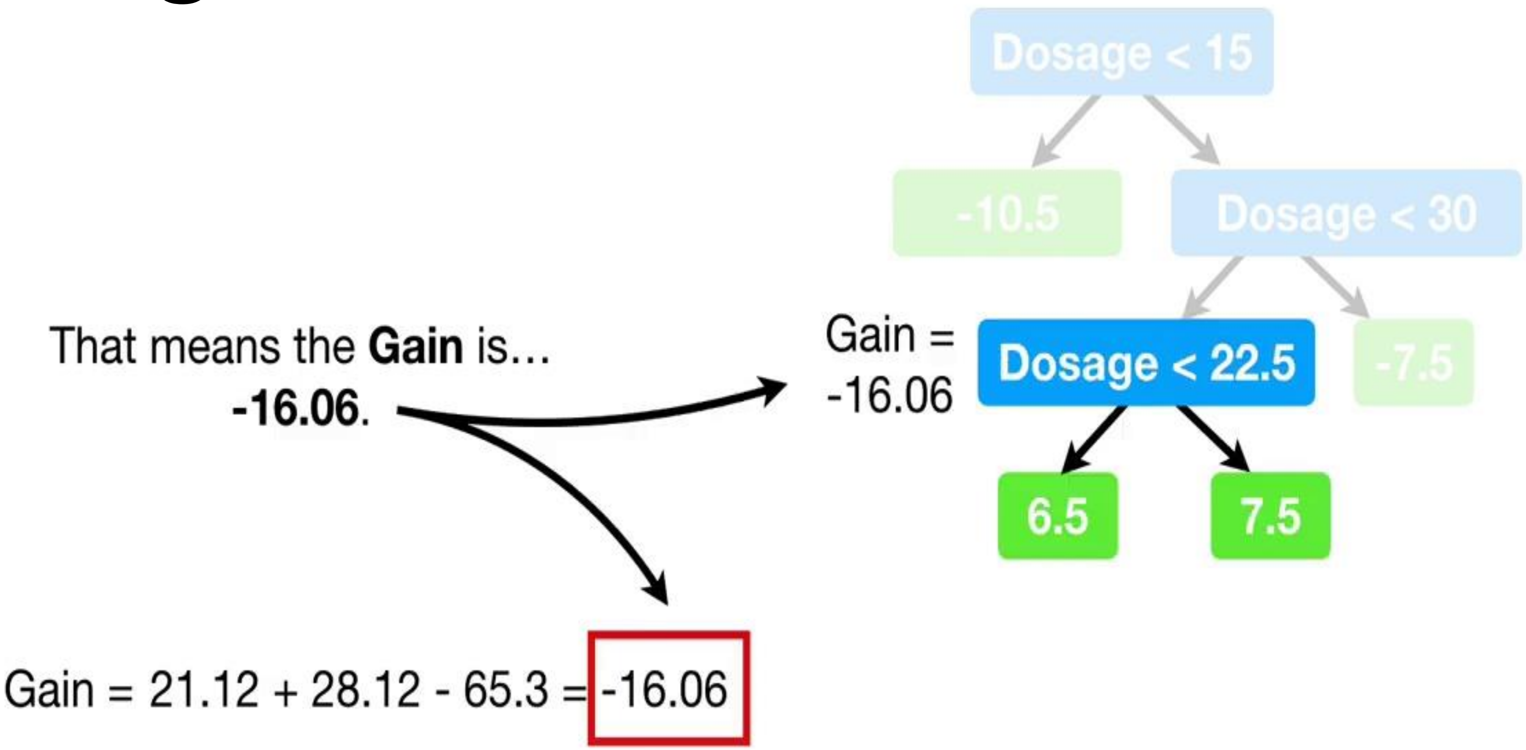
NOTE: Before we move on, I want illustrate one last feature of **λ (lambda)**.



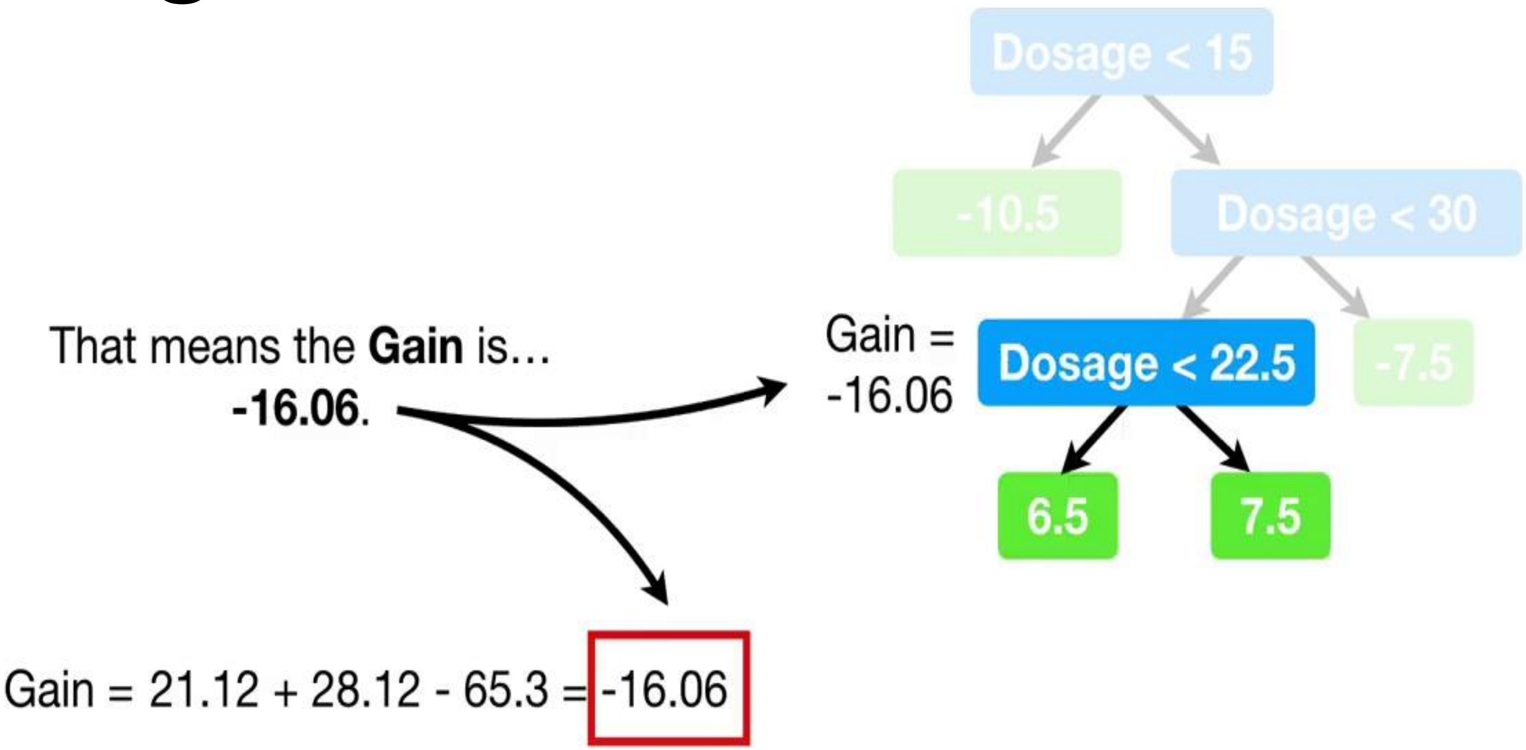
Tree Pruning



Tree Pruning

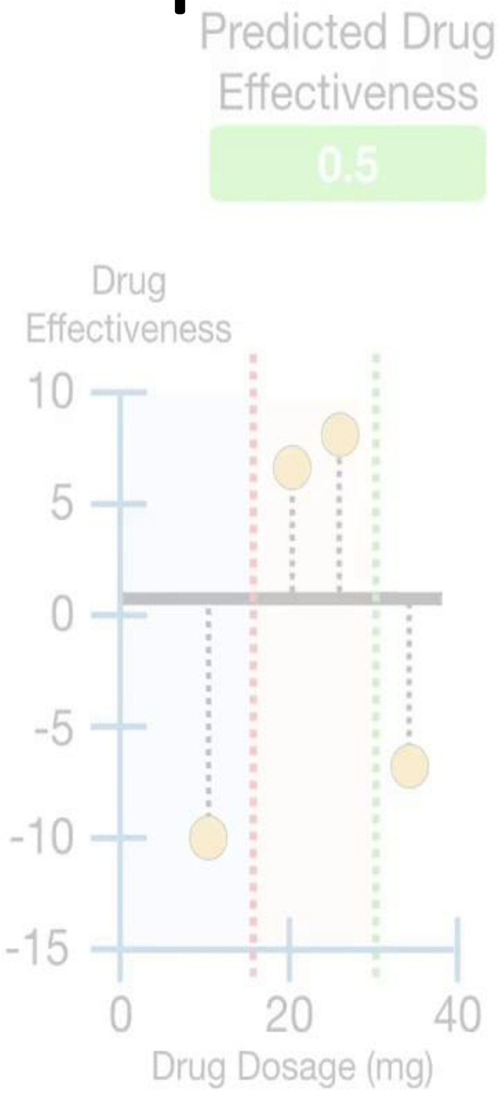


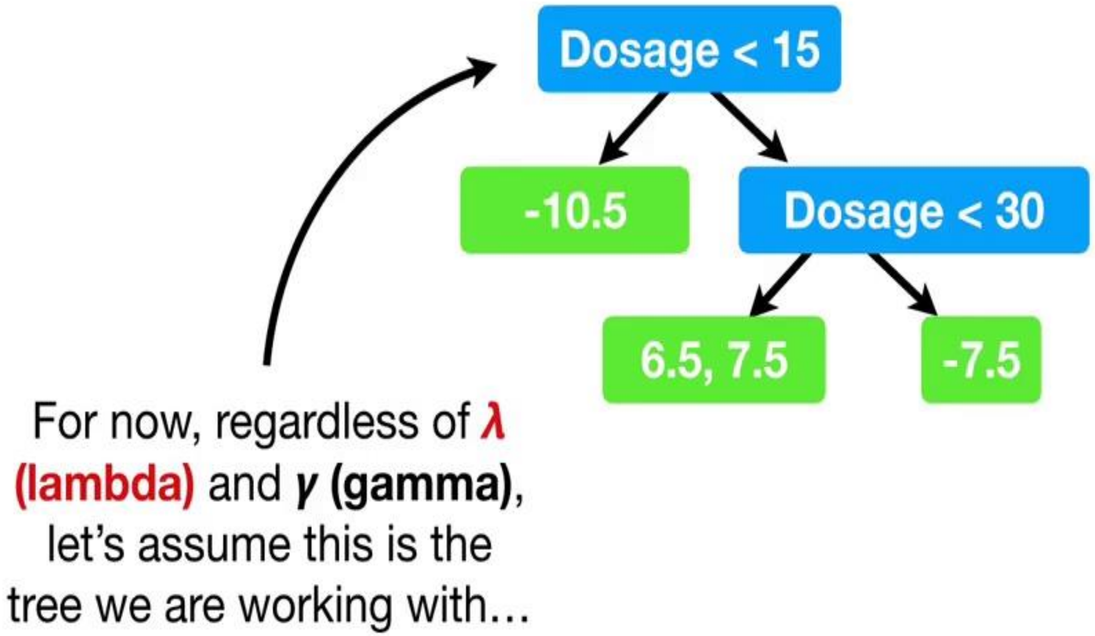
Tree Pruning



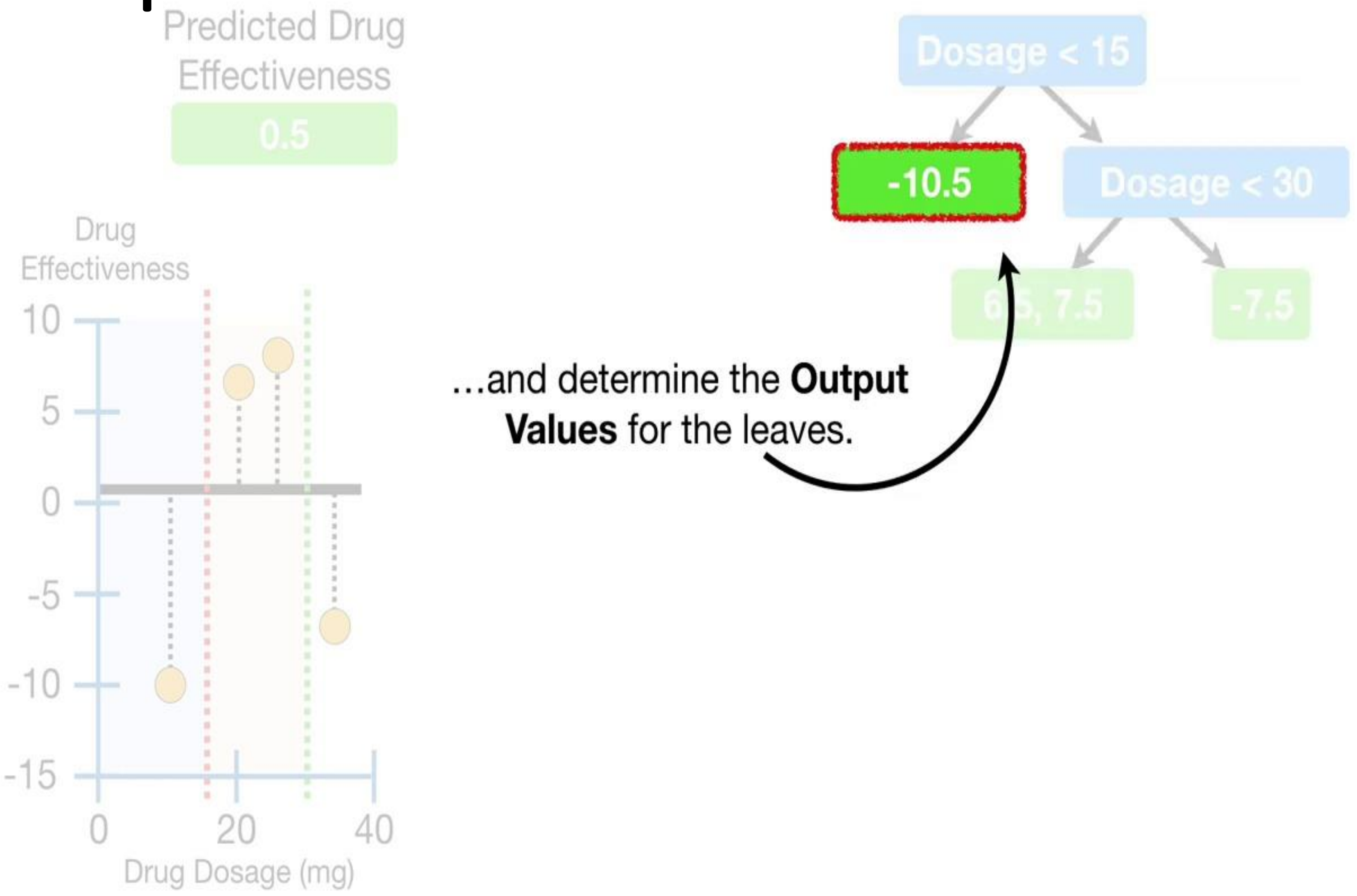
Also if we set **gamma** equals to 0 we prune this branch

Leaves Update

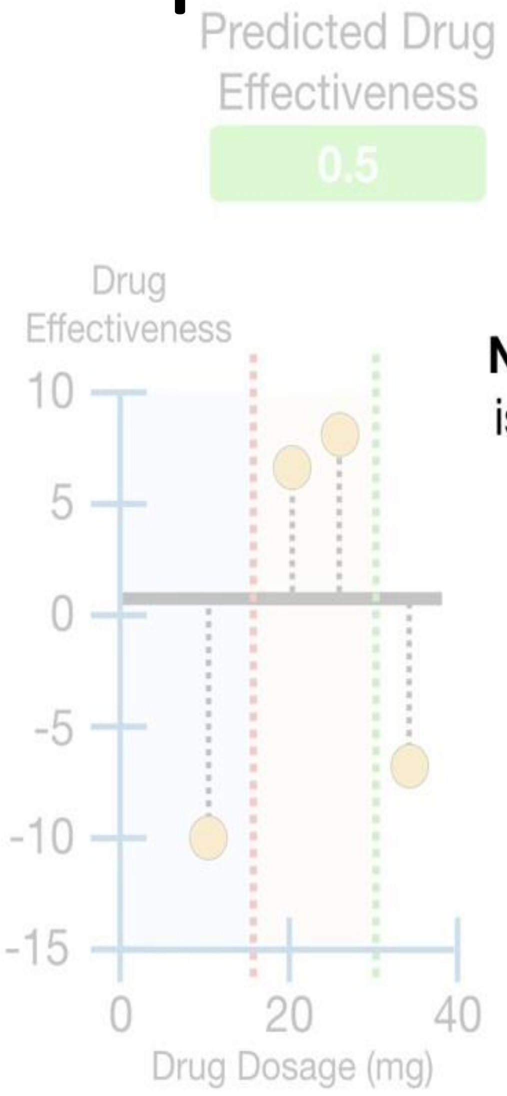


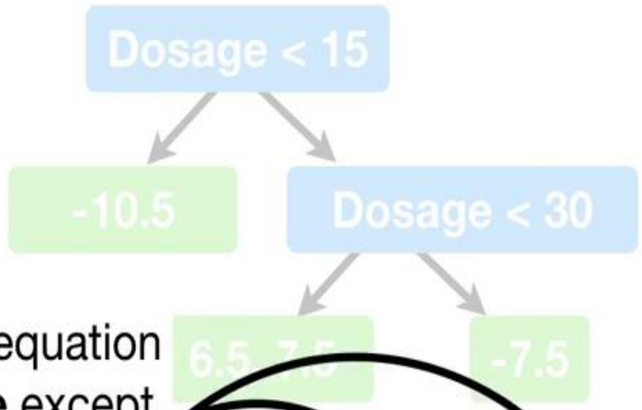


Leaves Update



Leaves Update



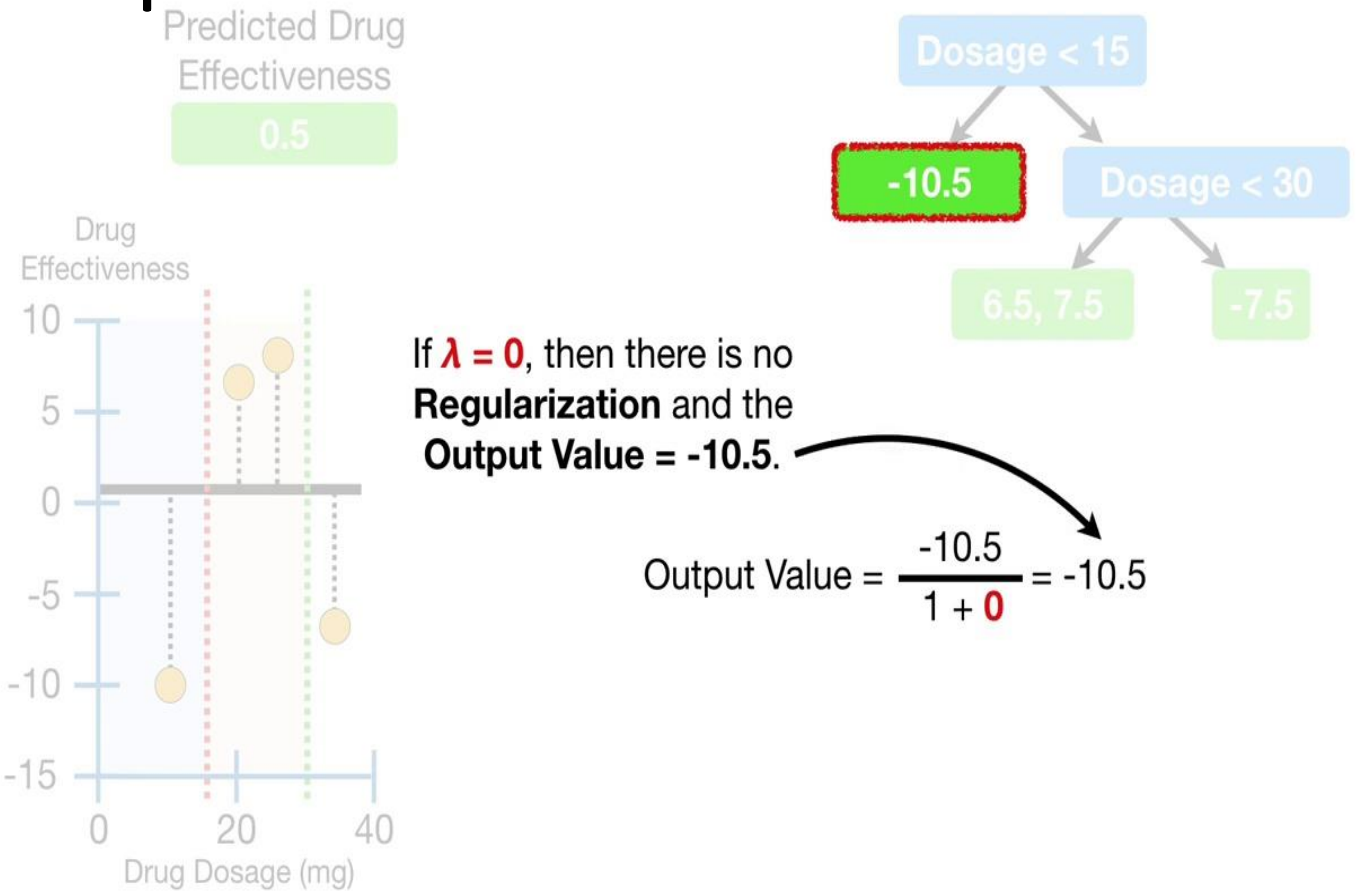


NOTE: The **Output Value** equation is like the **Similarity Score** except we do not square the sum of the residuals.

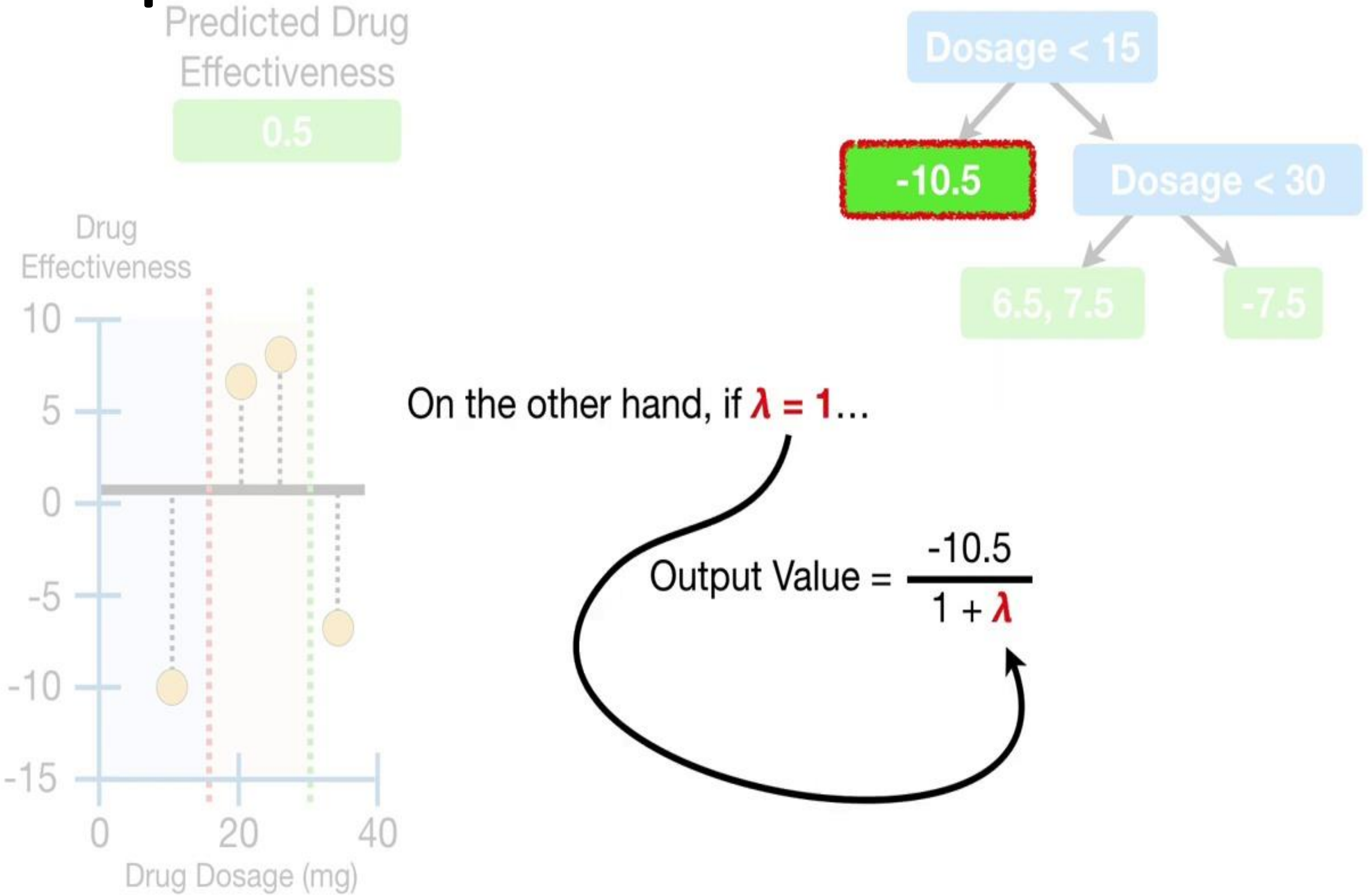
$$\text{Output Value} = \frac{\text{Sum of Residuals}}{\text{Number of Residuals} + \lambda}$$

$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + 1}$$

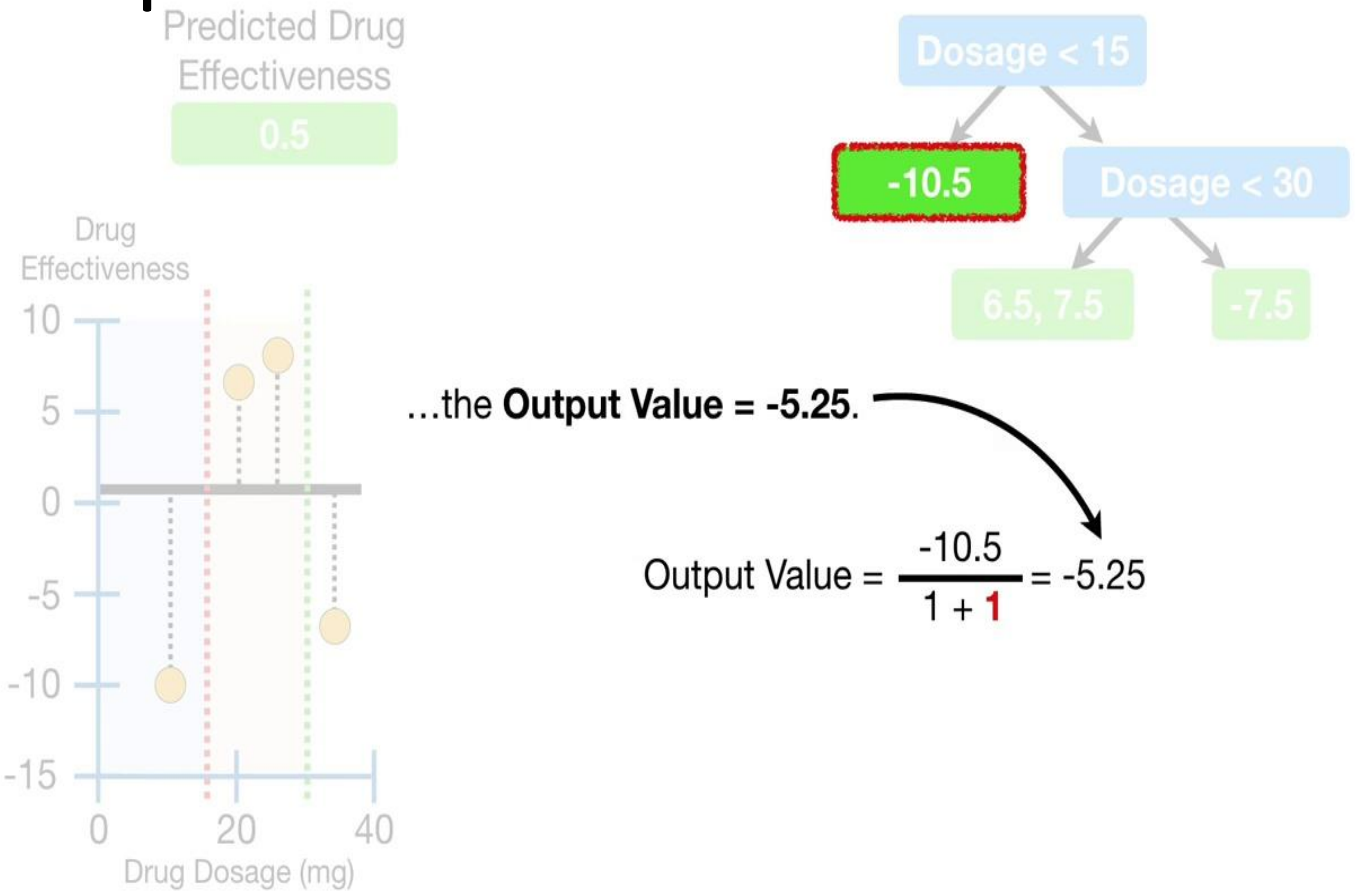
Leaves Update



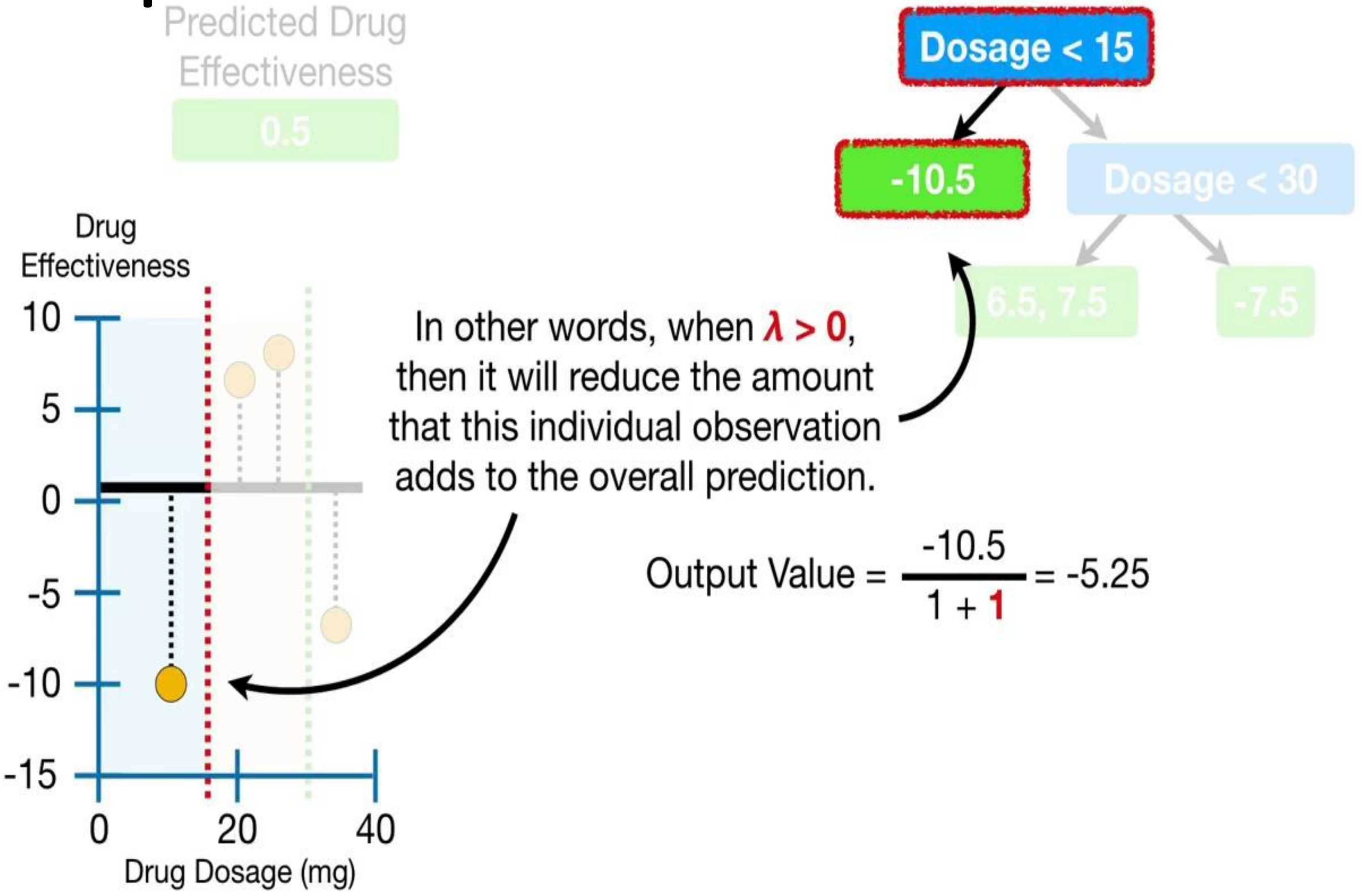
Leaves Update



Leaves Update

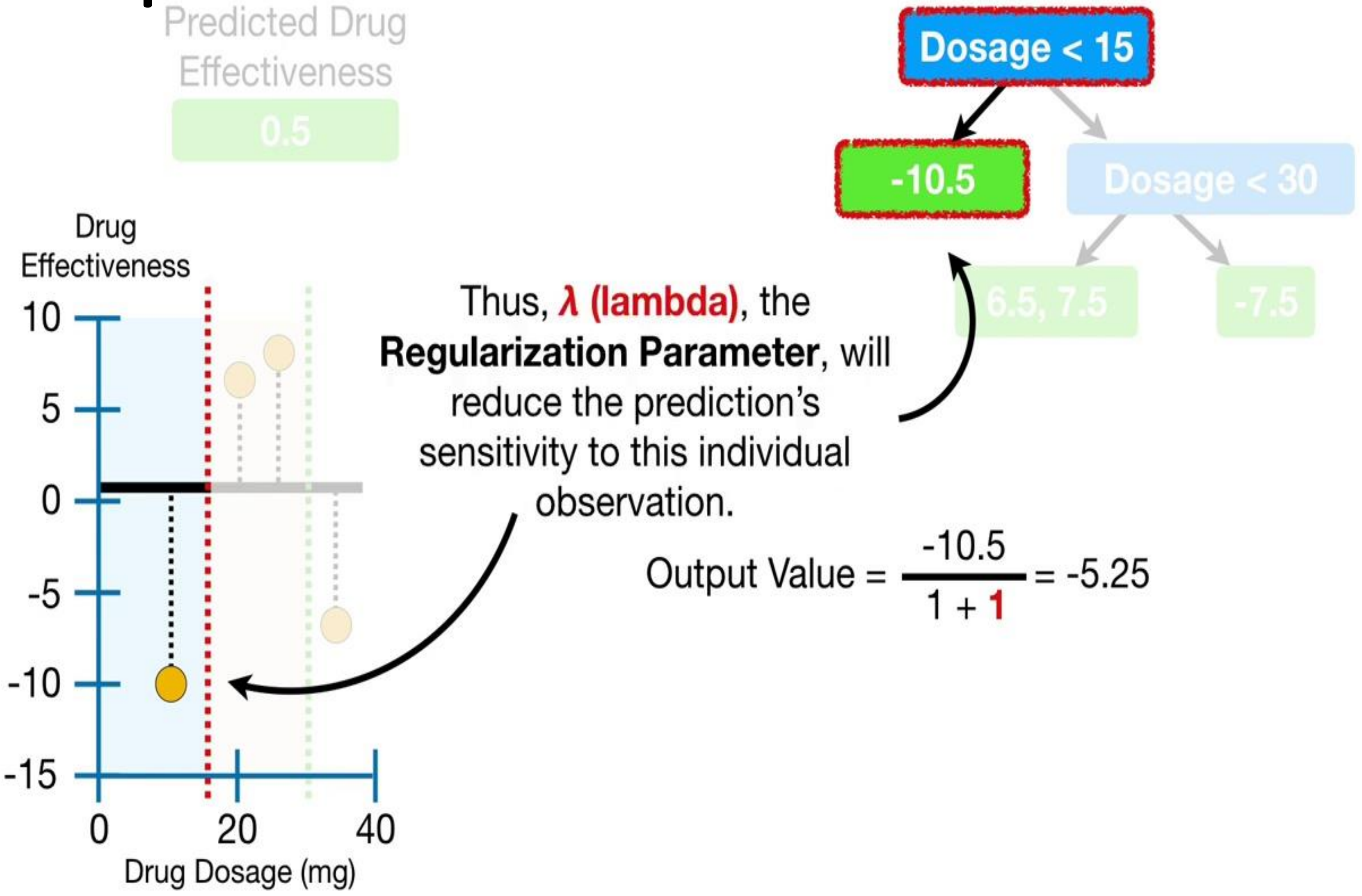


Leaves Update

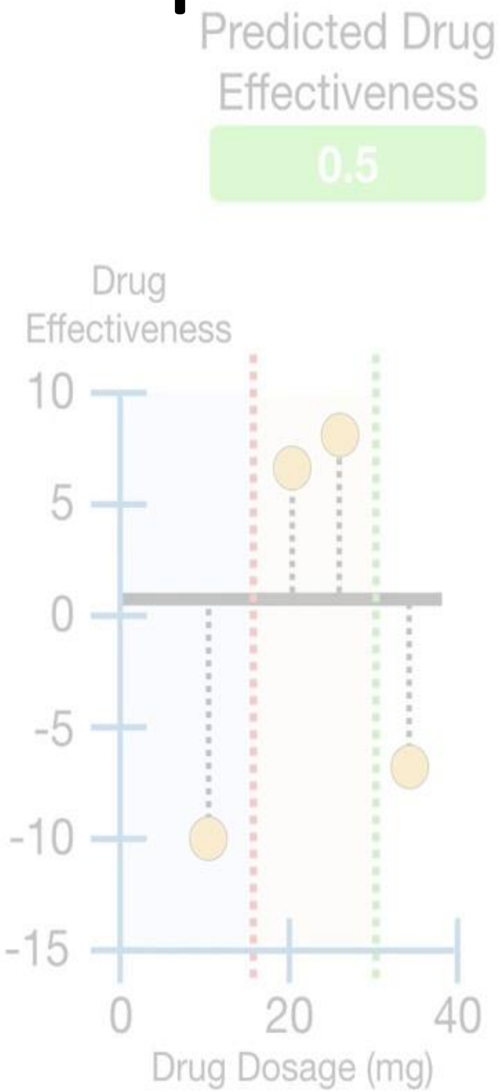


$$\text{Output Value} = \frac{-10.5}{1 + 1} = -5.25$$

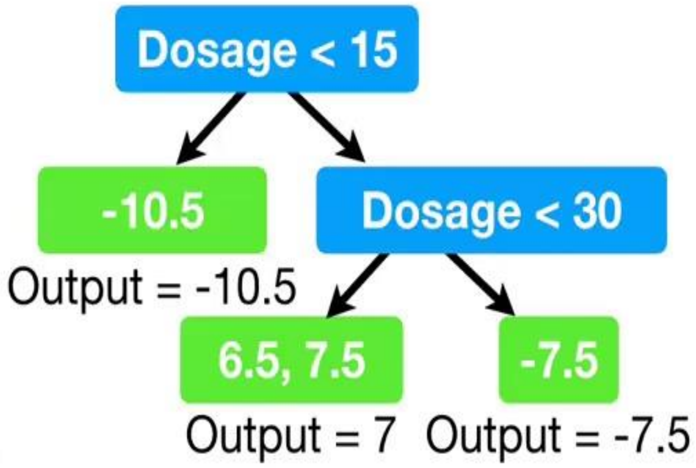
Leaves Update



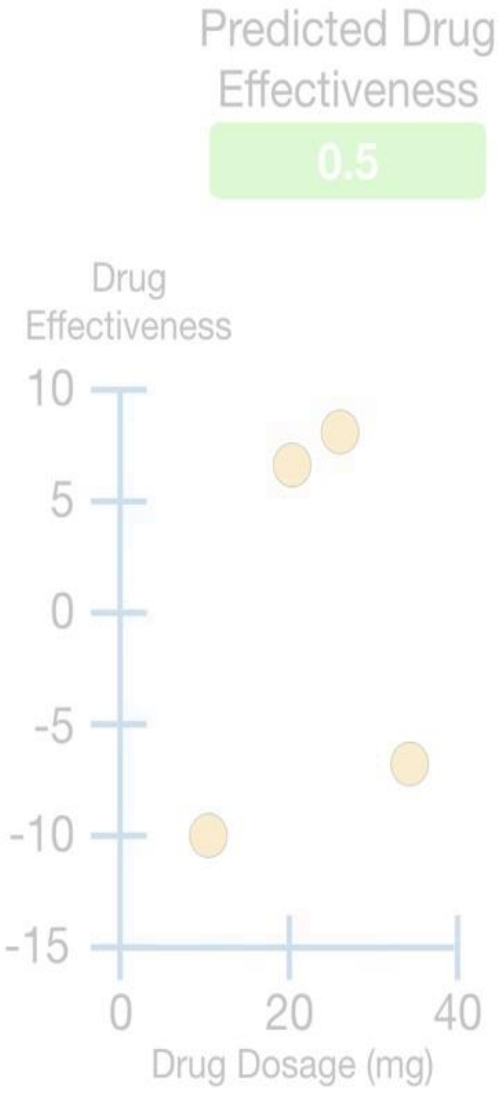
Leaves Update

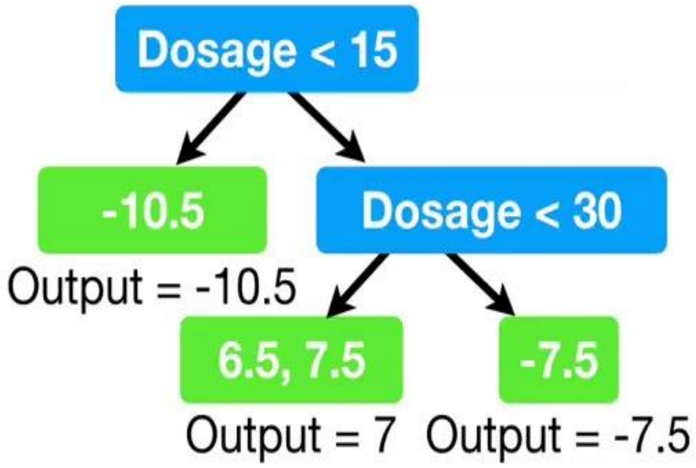


Now, at long last, the first tree is complete!



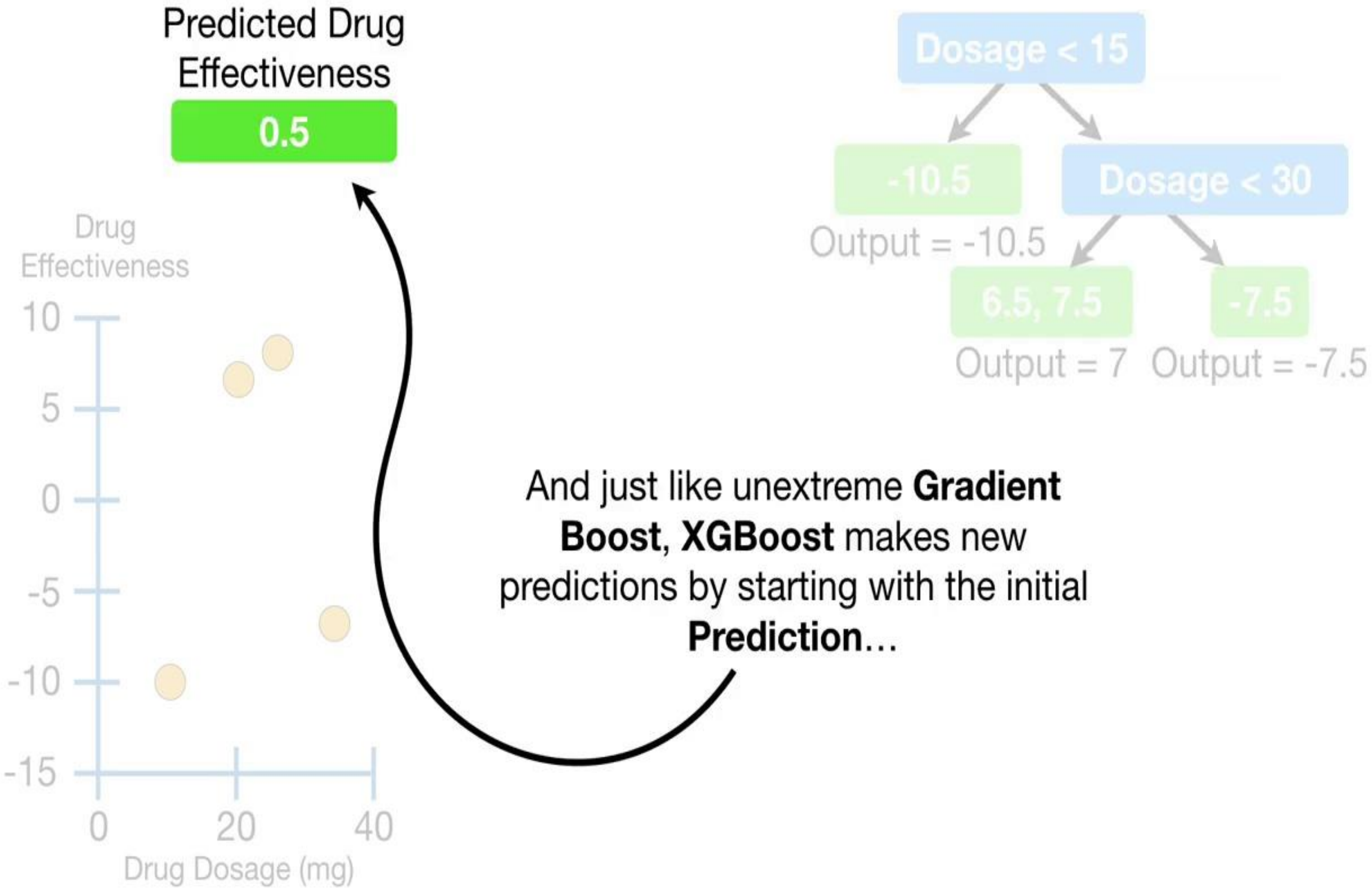
Tree Prediction



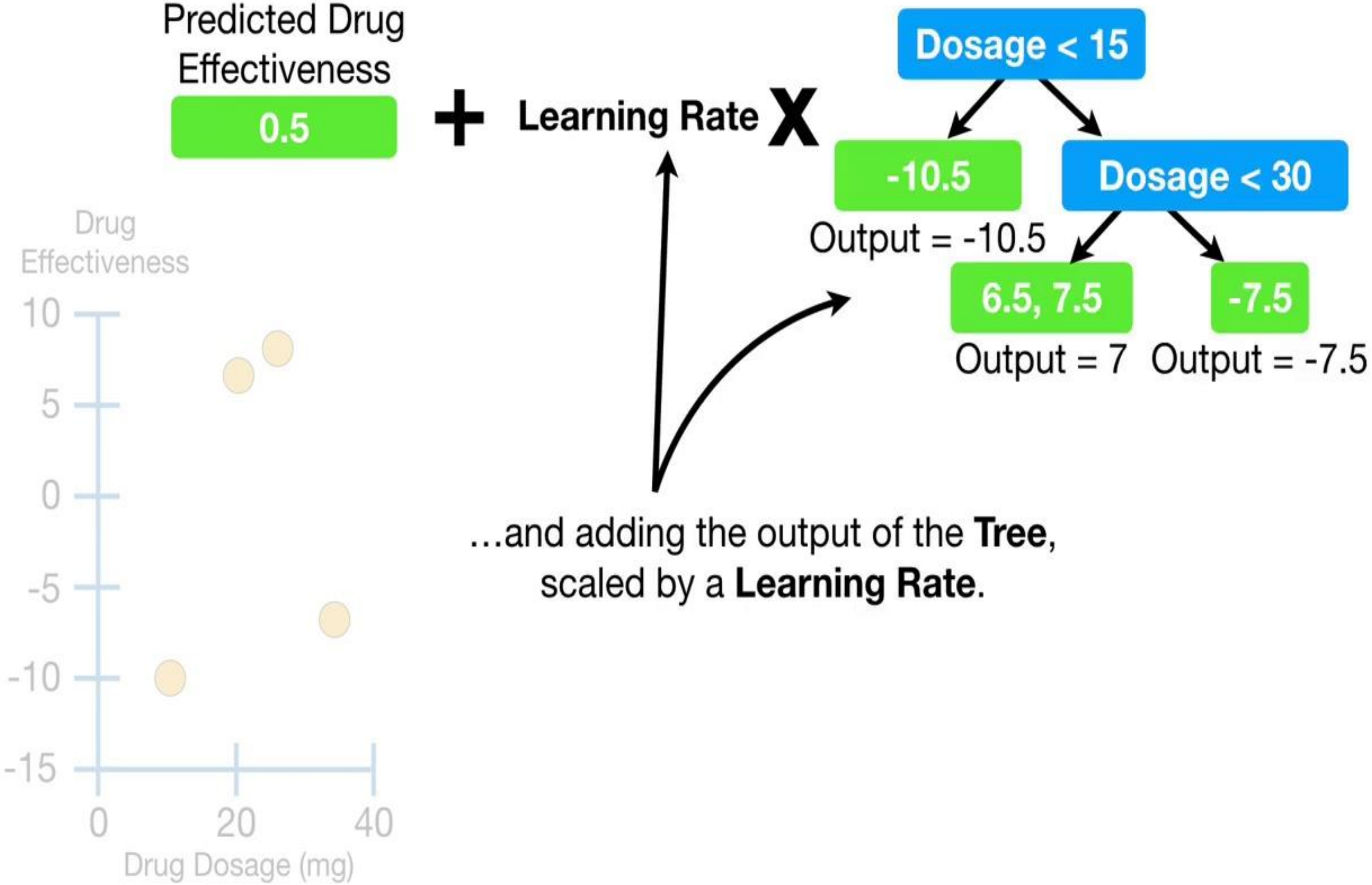


Since we have built our first tree,
we can make new **Predictions**.

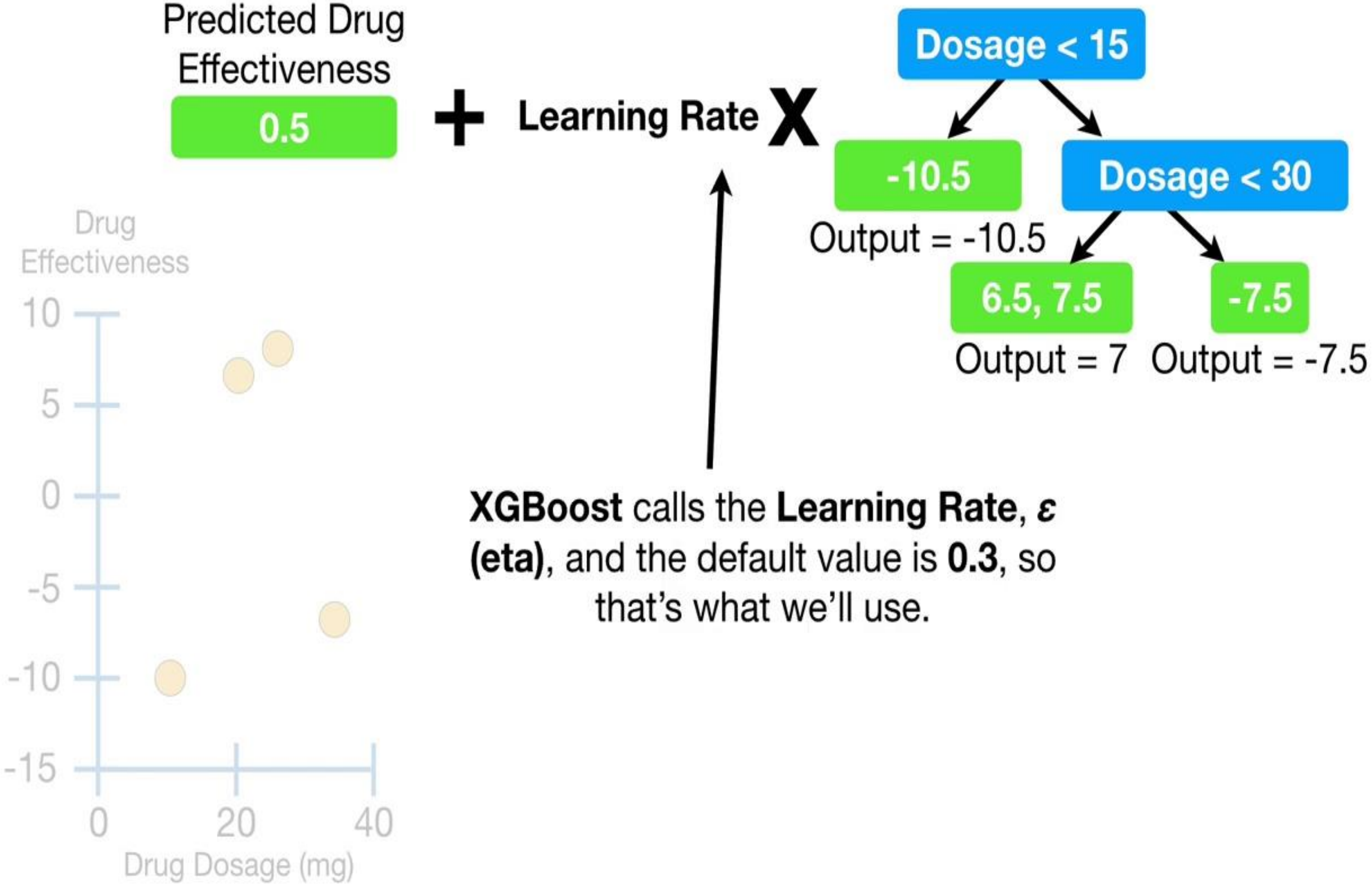
Tree Prediction



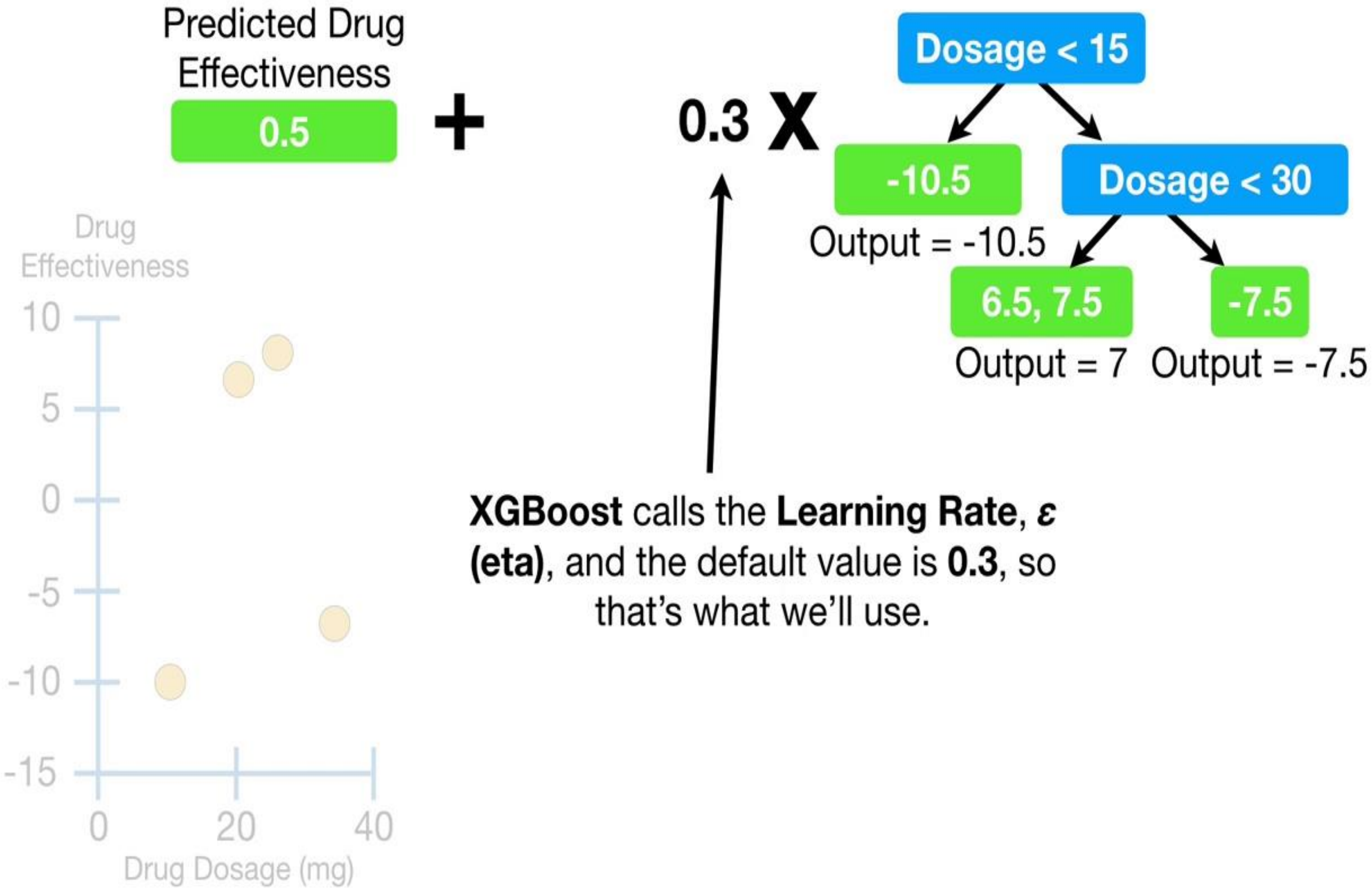
Tree Prediction



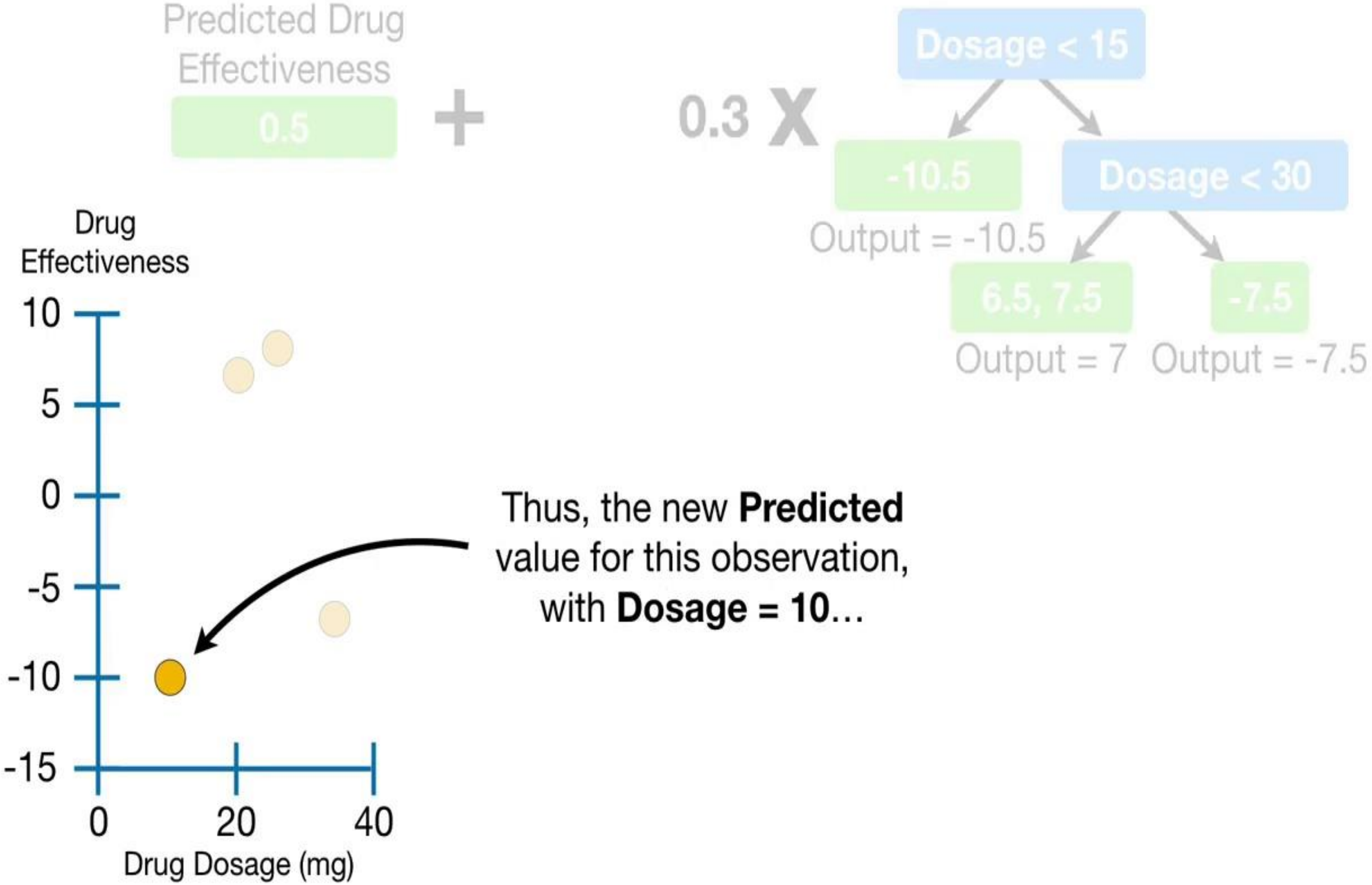
Tree Prediction



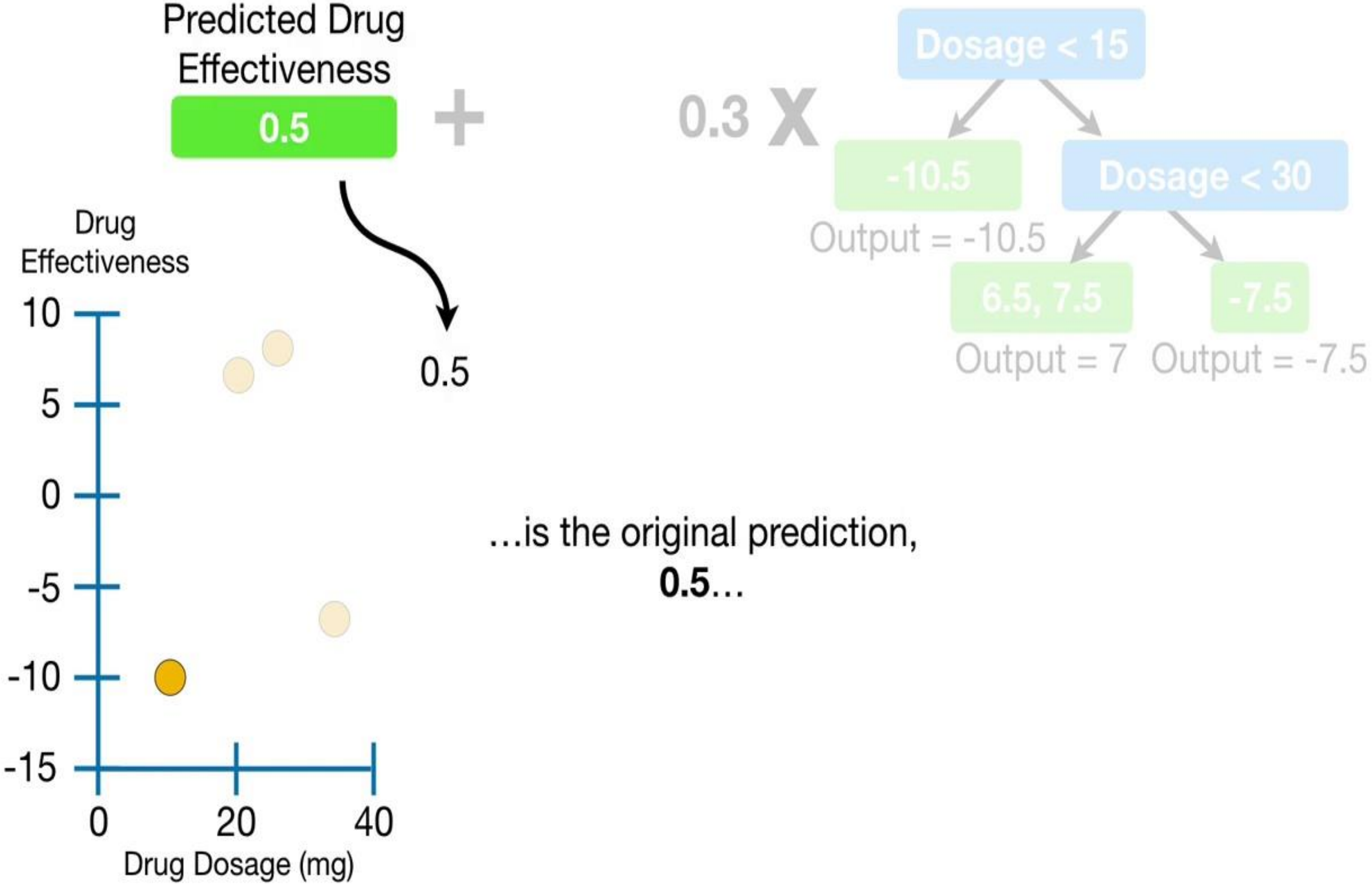
Tree Prediction



Tree Prediction

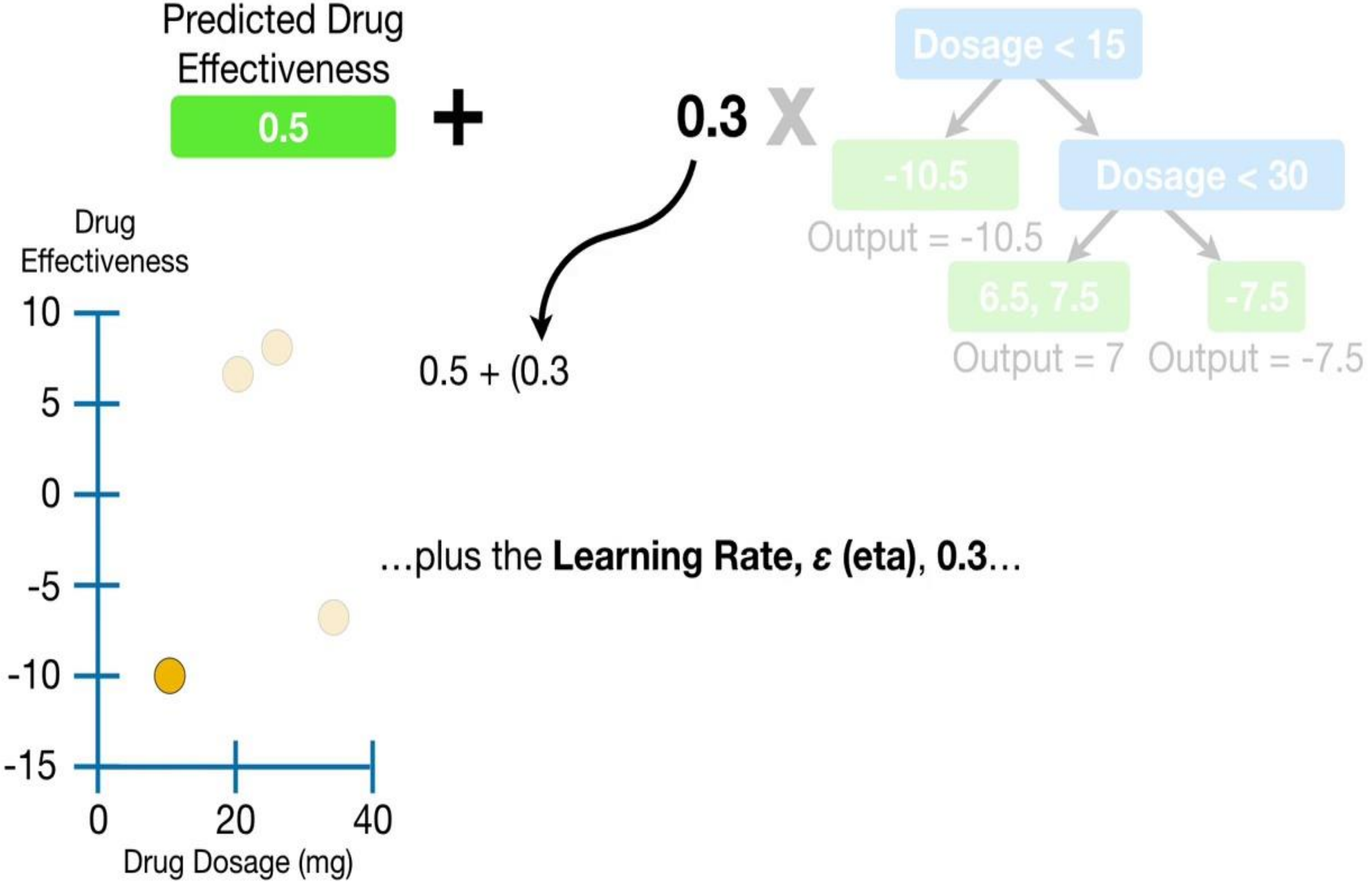


Tree Prediction

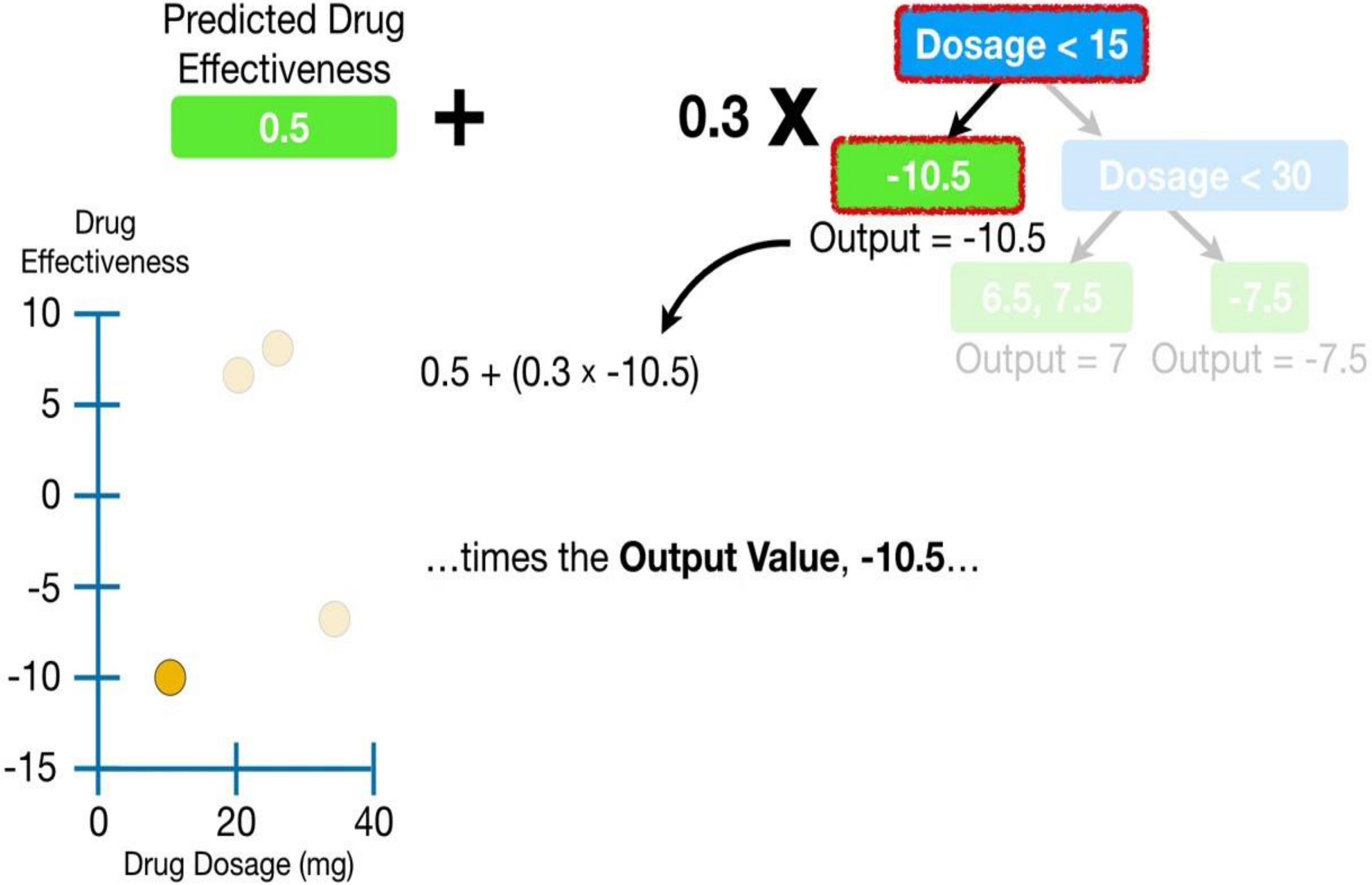


...is the original prediction, **0.5**...

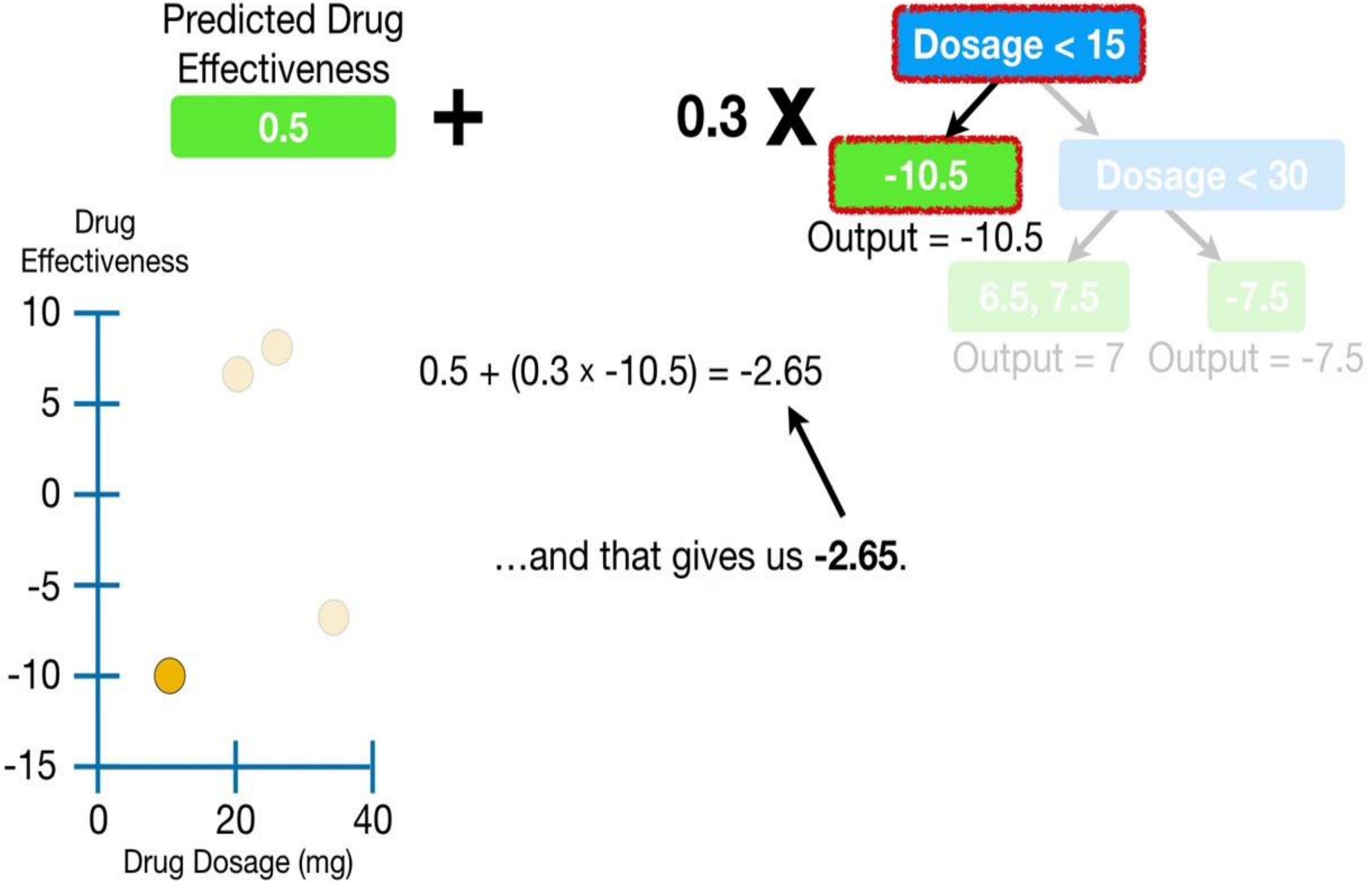
Tree Prediction



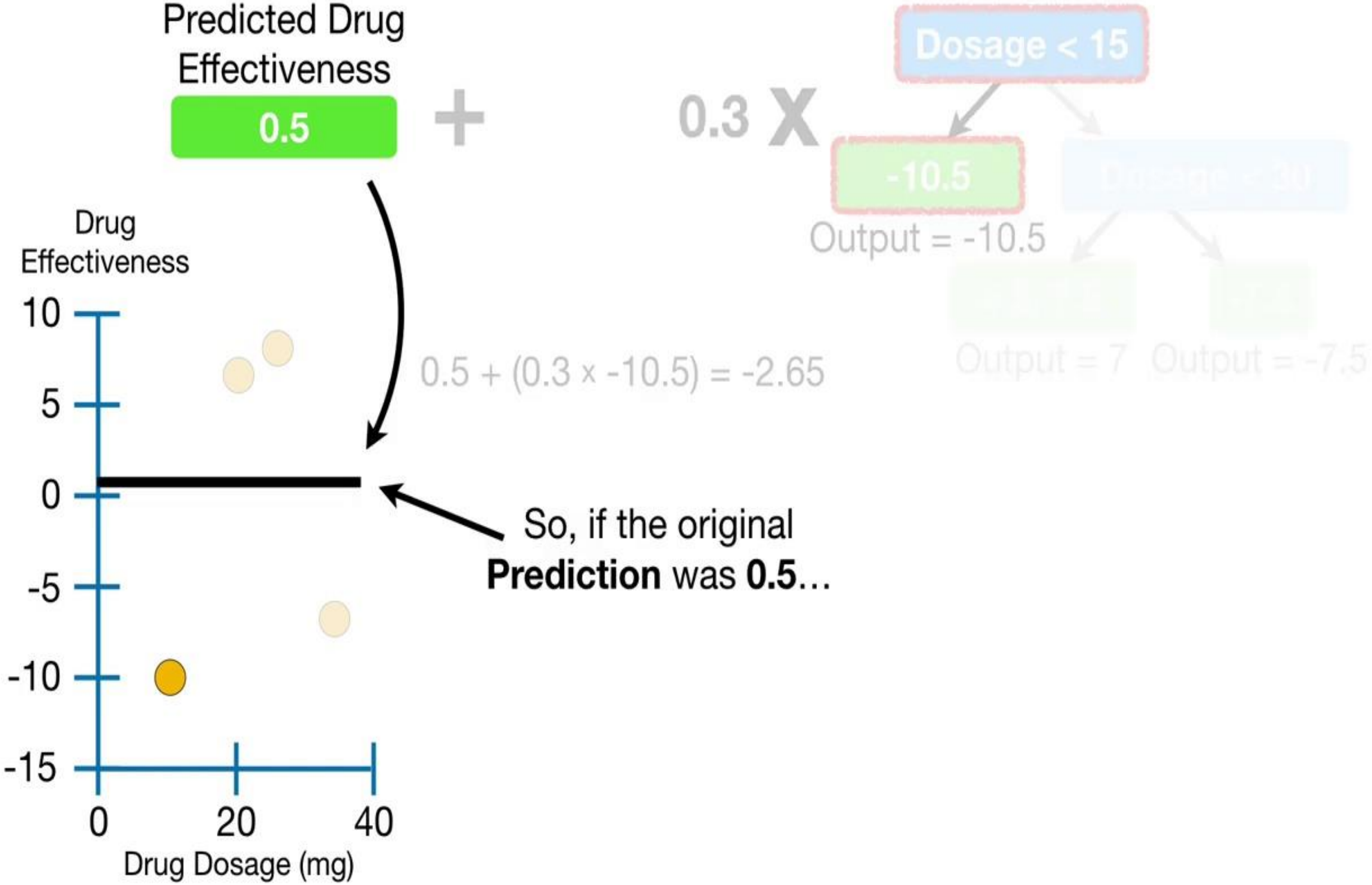
Tree Prediction



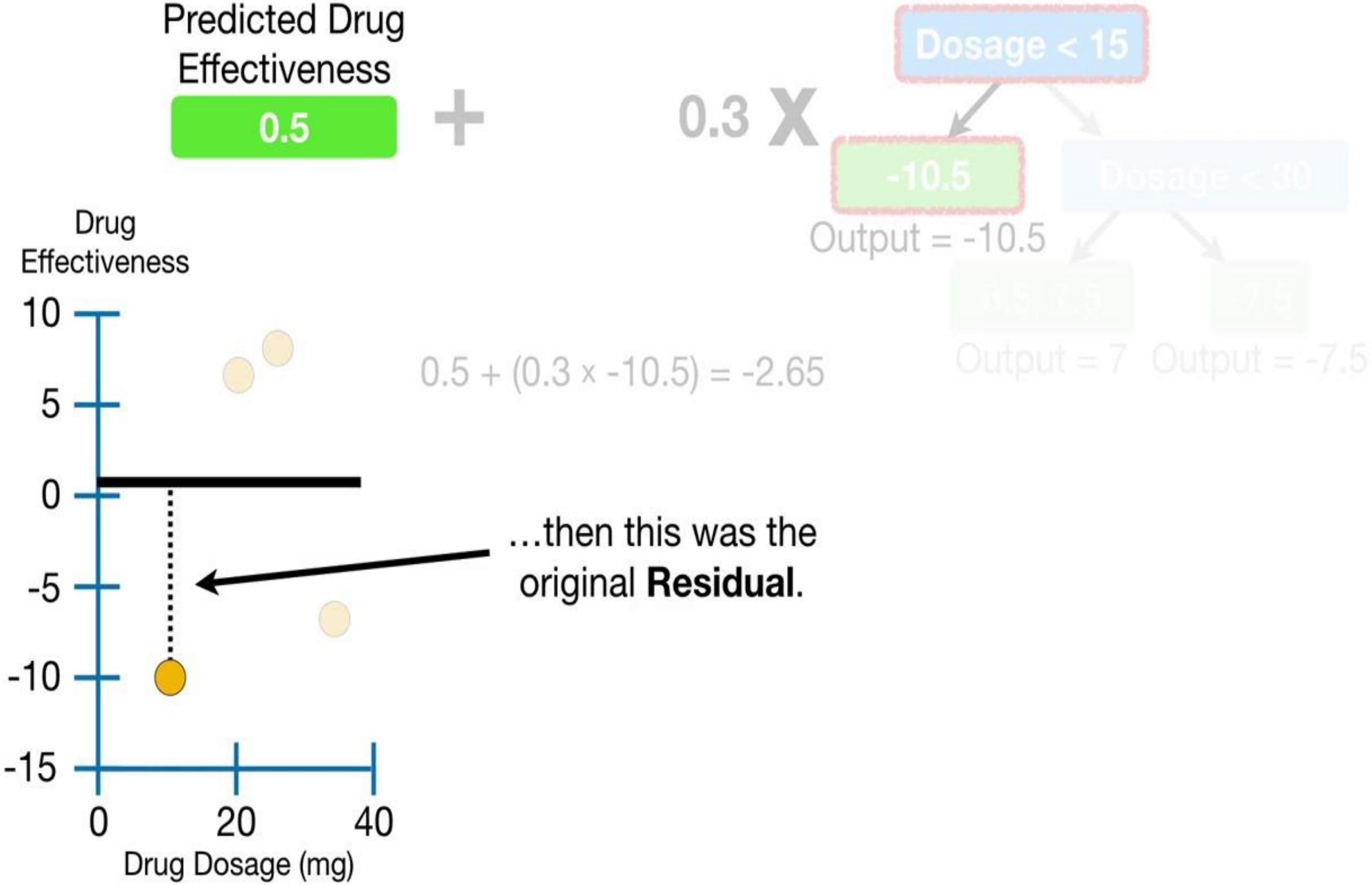
Tree Prediction



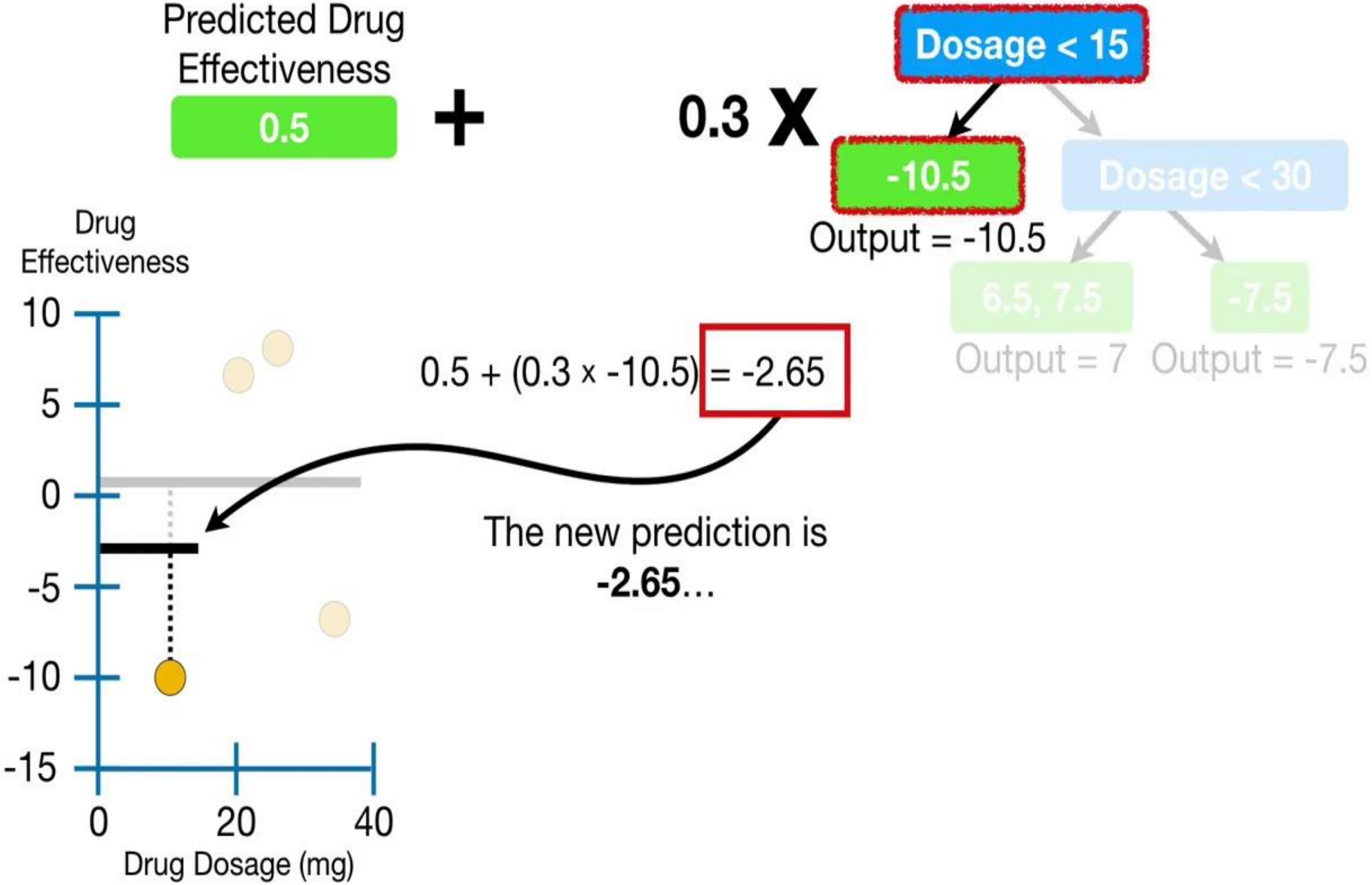
Tree Prediction



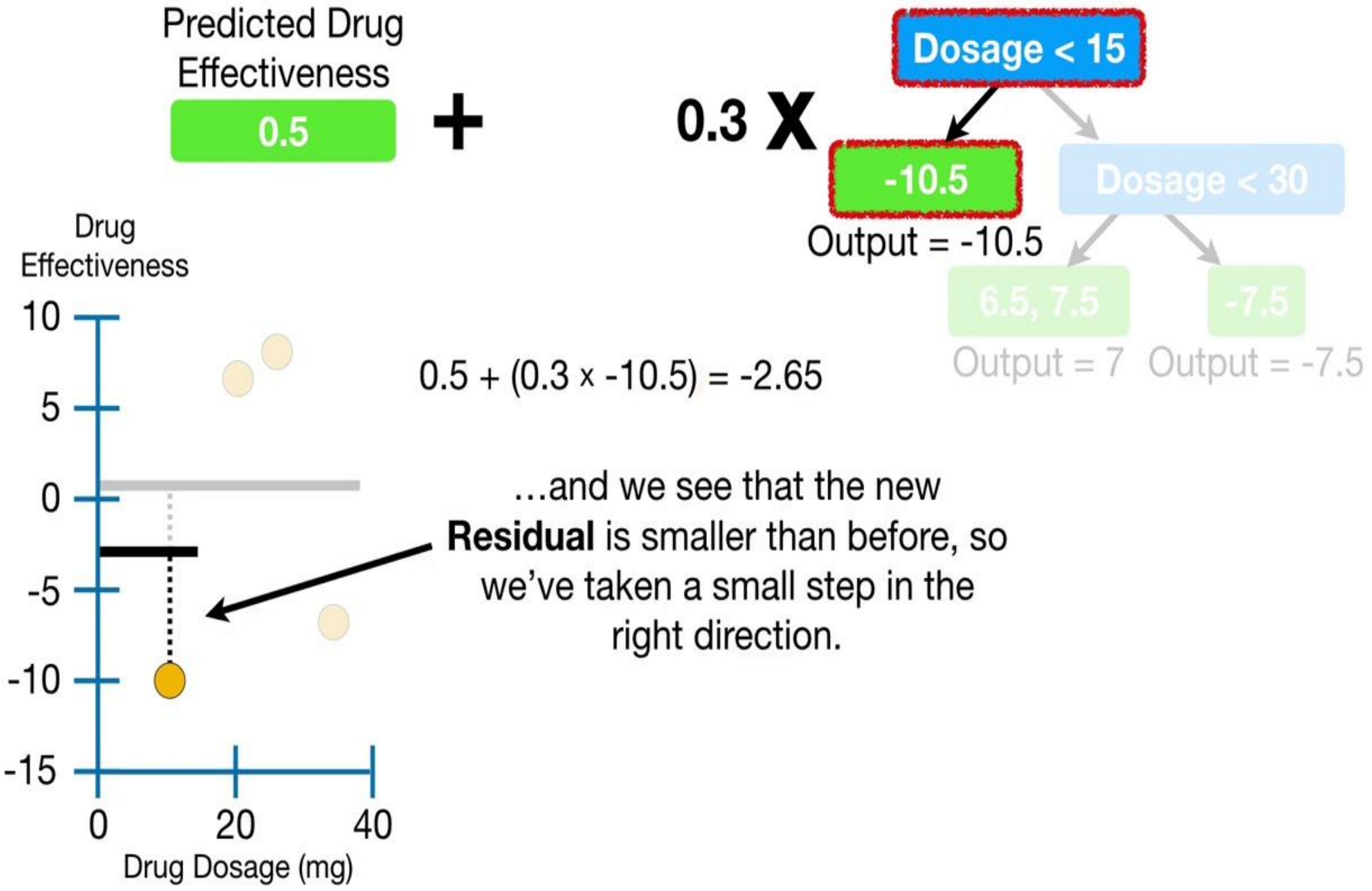
Tree Prediction



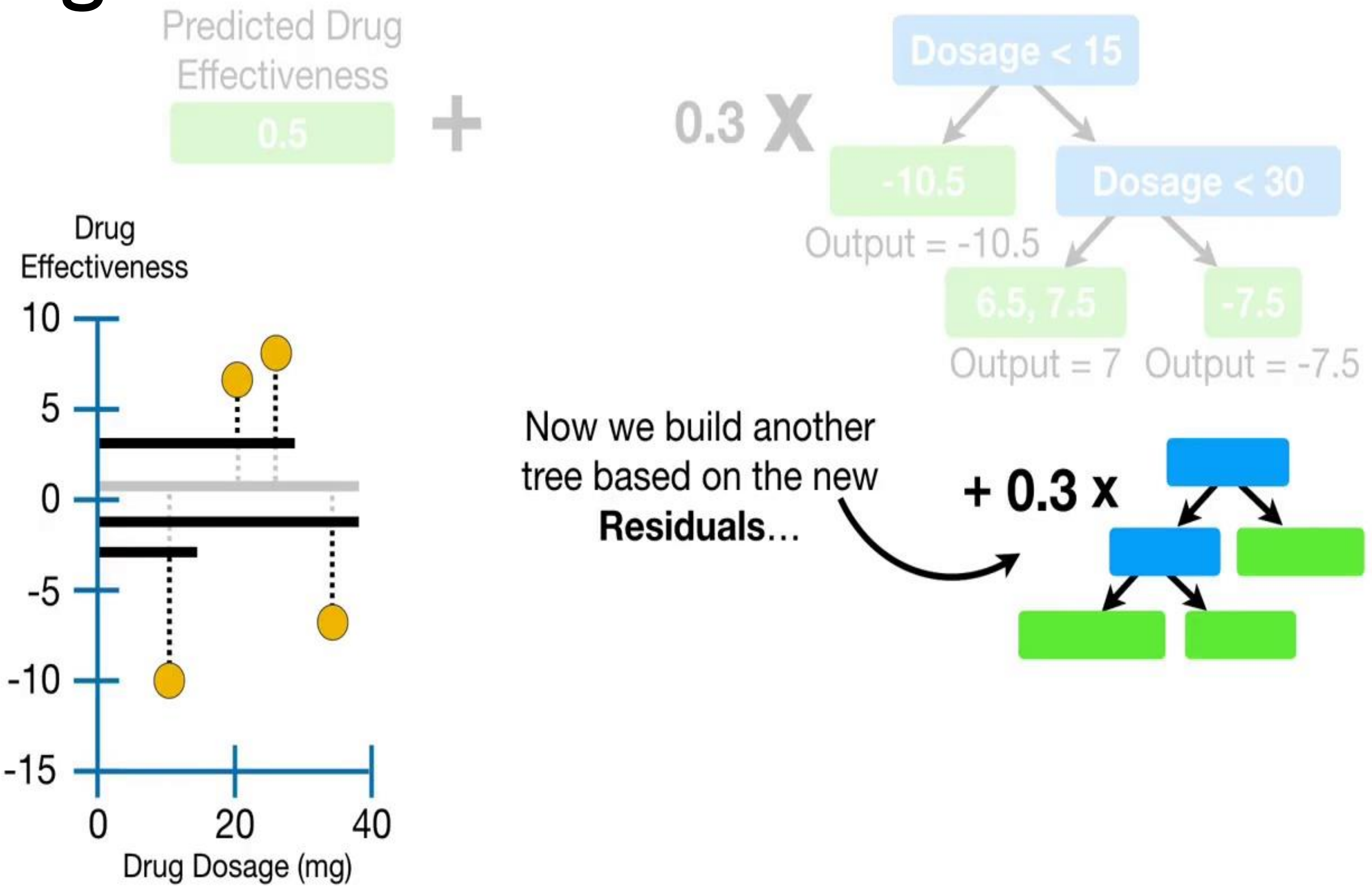
Tree Prediction



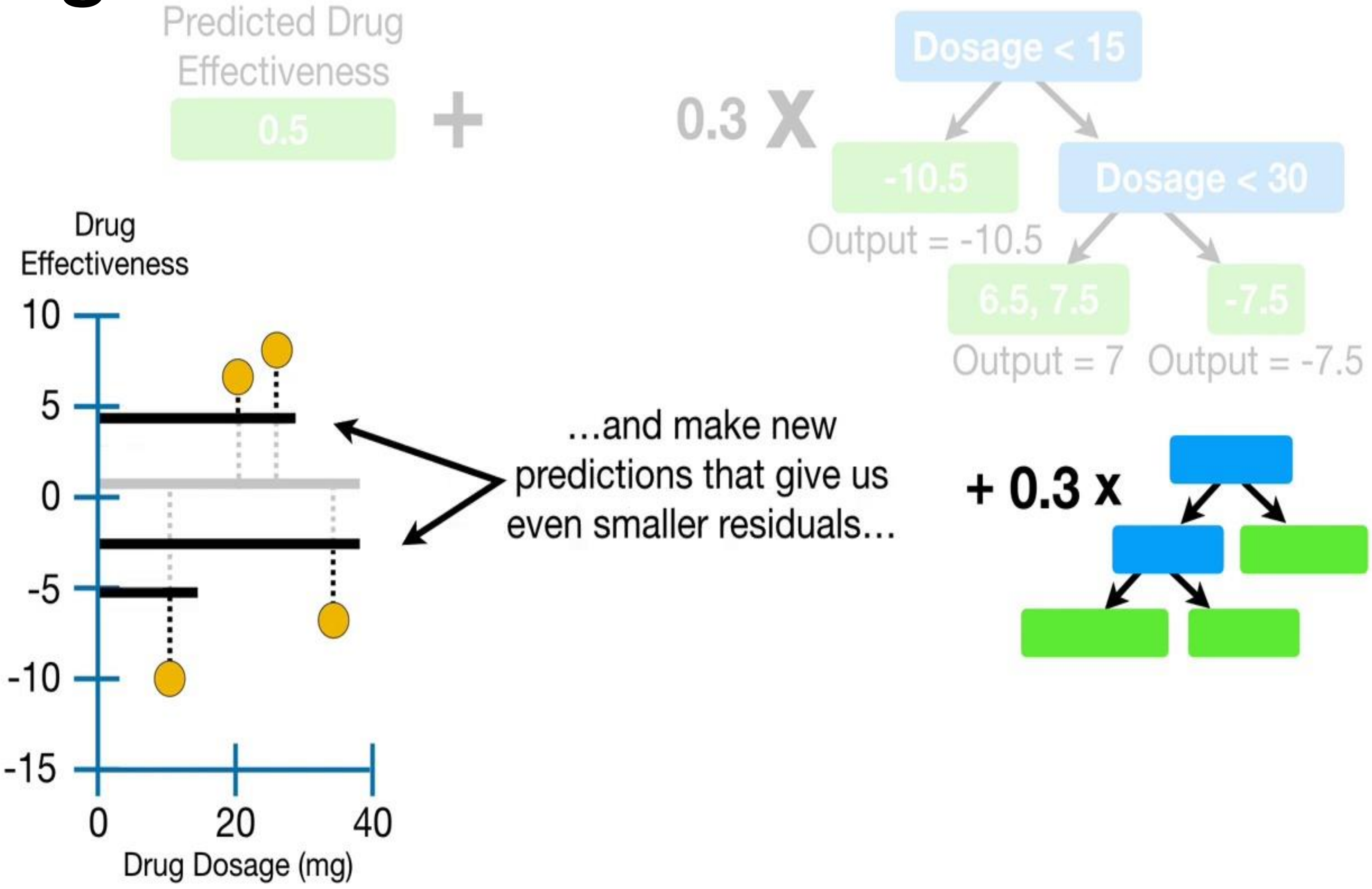
Tree Prediction



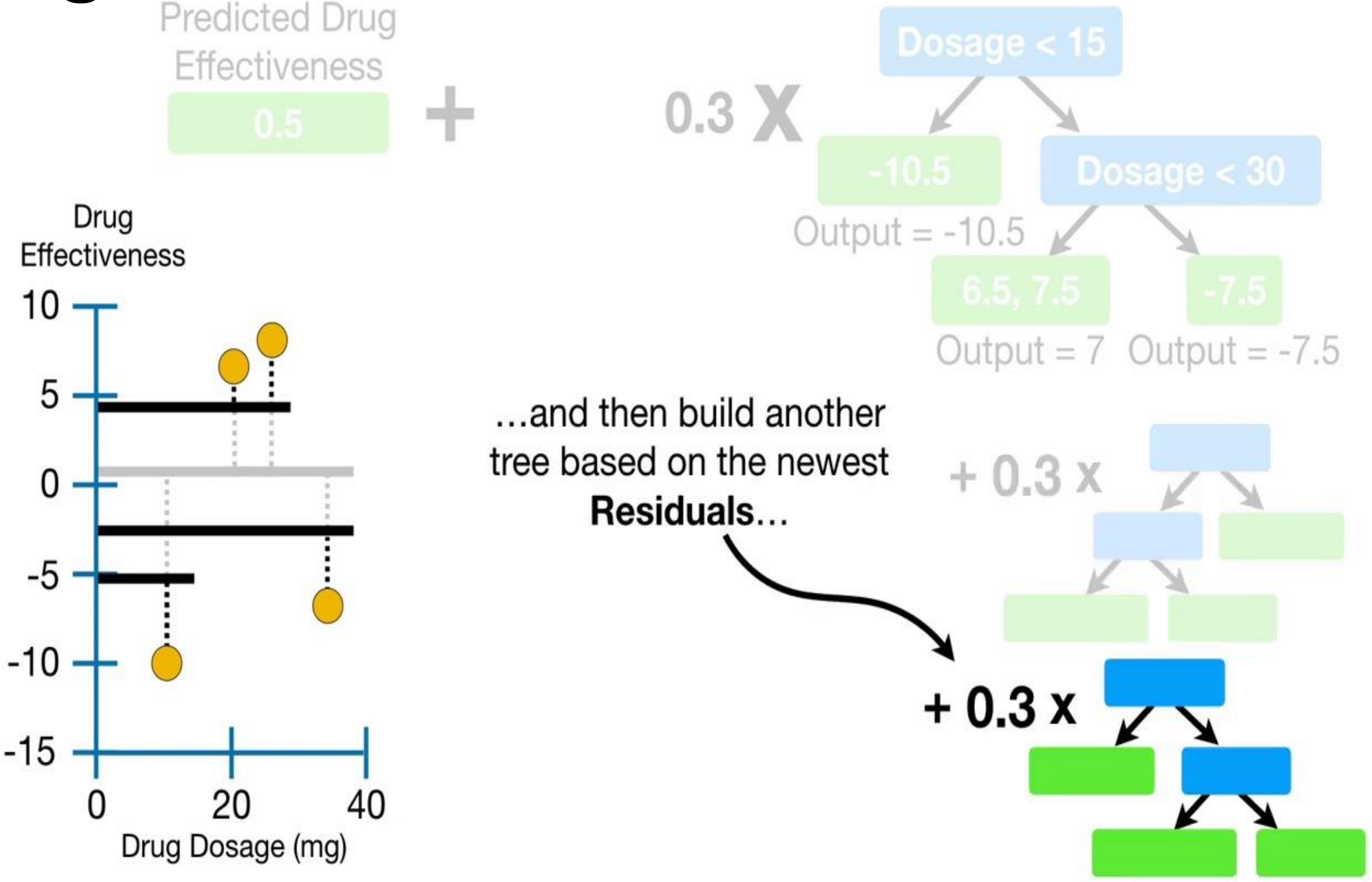
Building More Trees



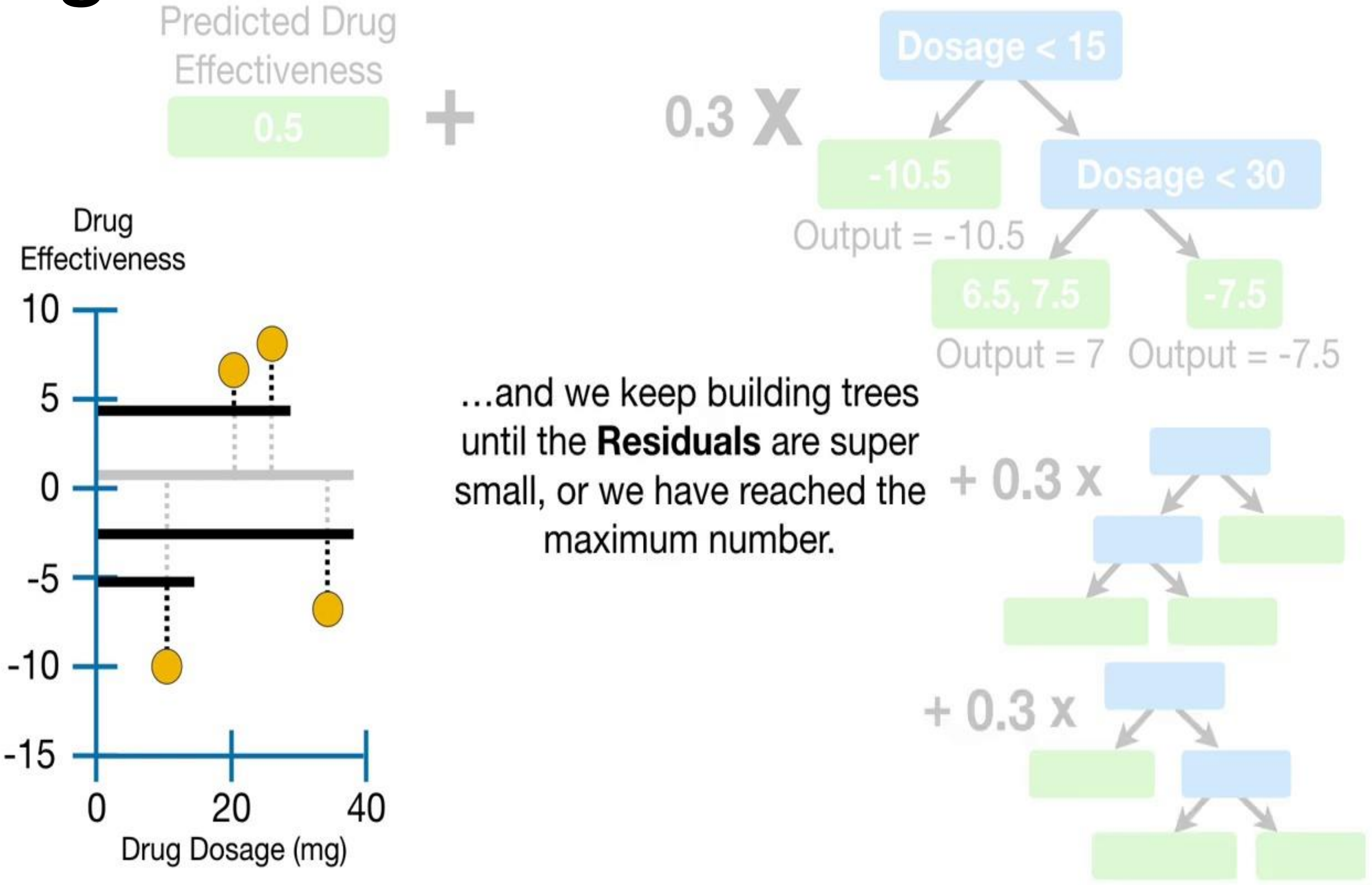
Building More Trees



Building More Trees

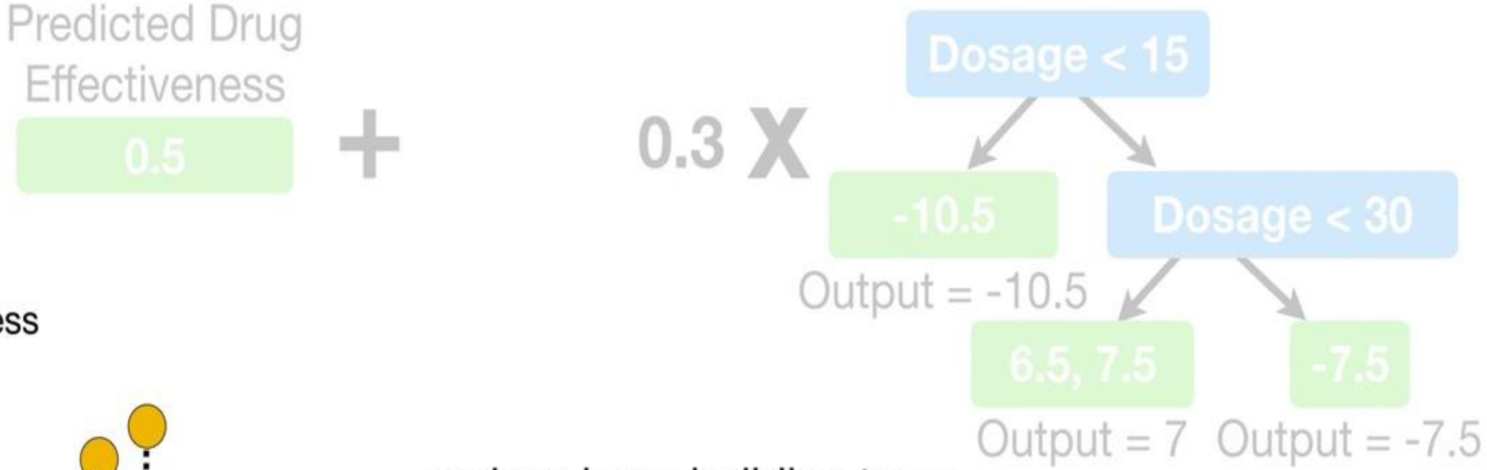


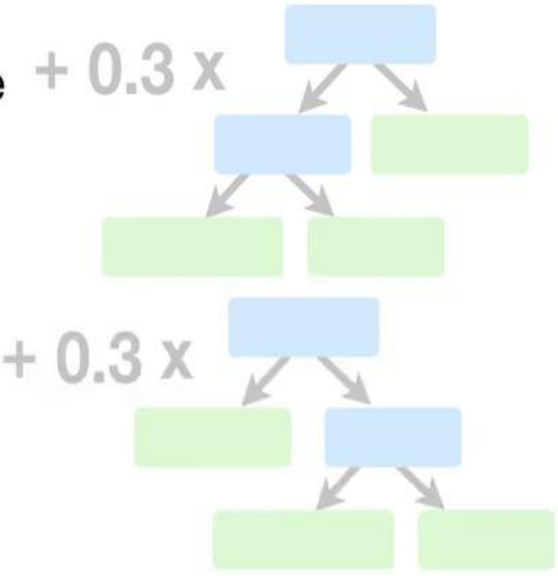
Building More Trees

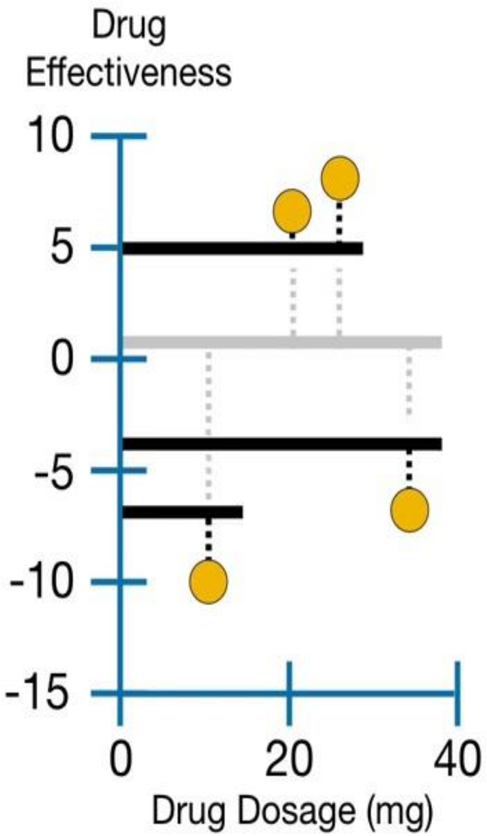


...and we keep building trees until the **Residuals** are super small, or we have reached the maximum number.

Building More Trees

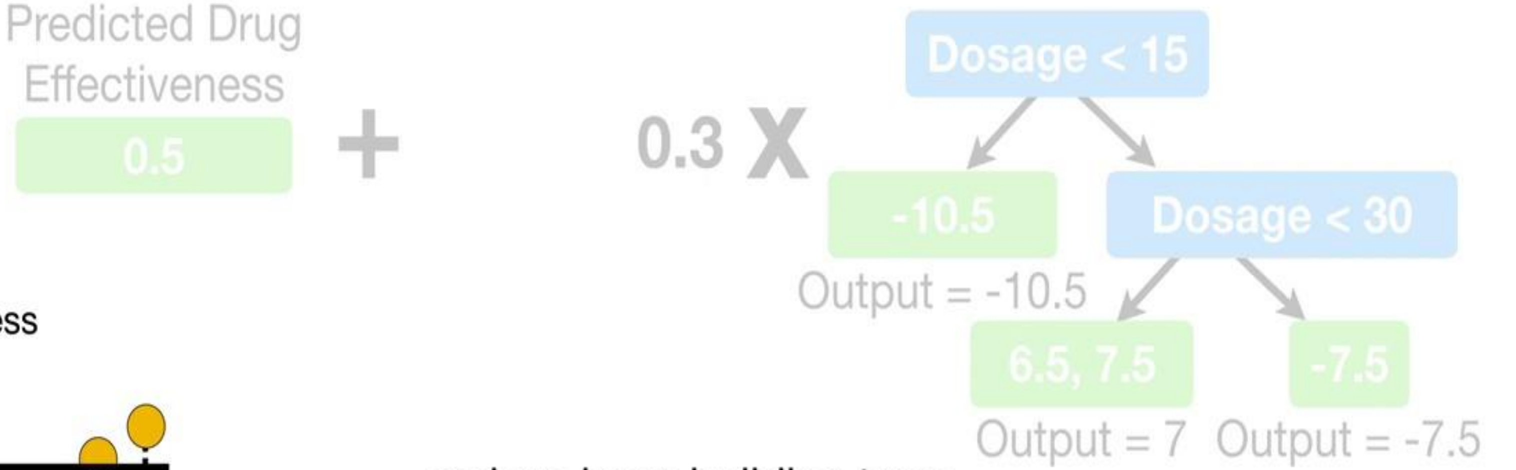


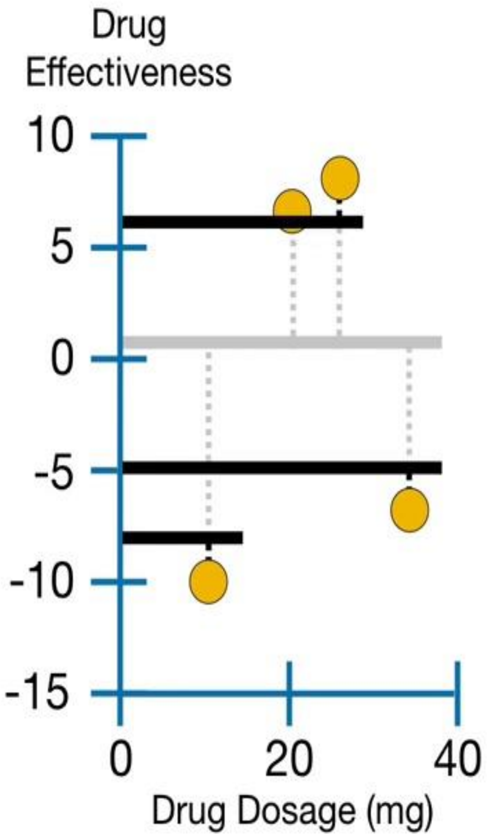




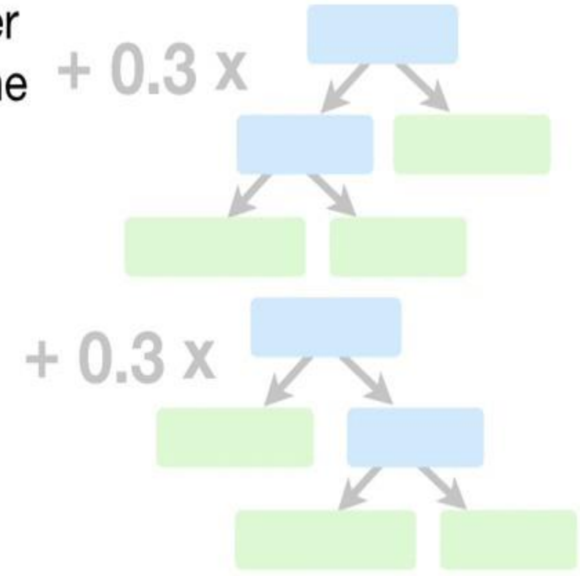
...and we keep building trees until the **Residuals** are super small, or we have reached the maximum number.

Building More Trees

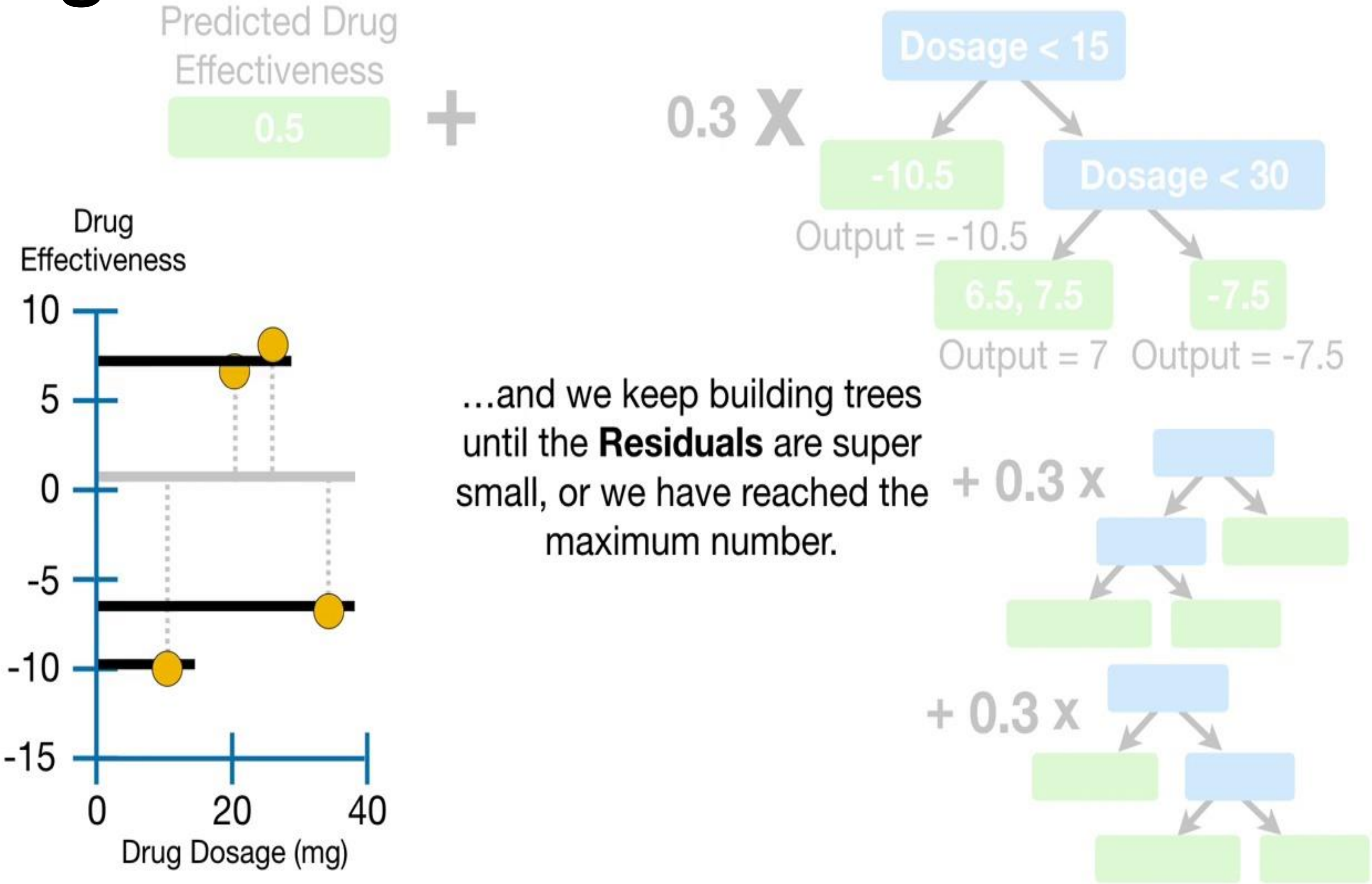




...and we keep building trees until the **Residuals** are super small, or we have reached the maximum number.

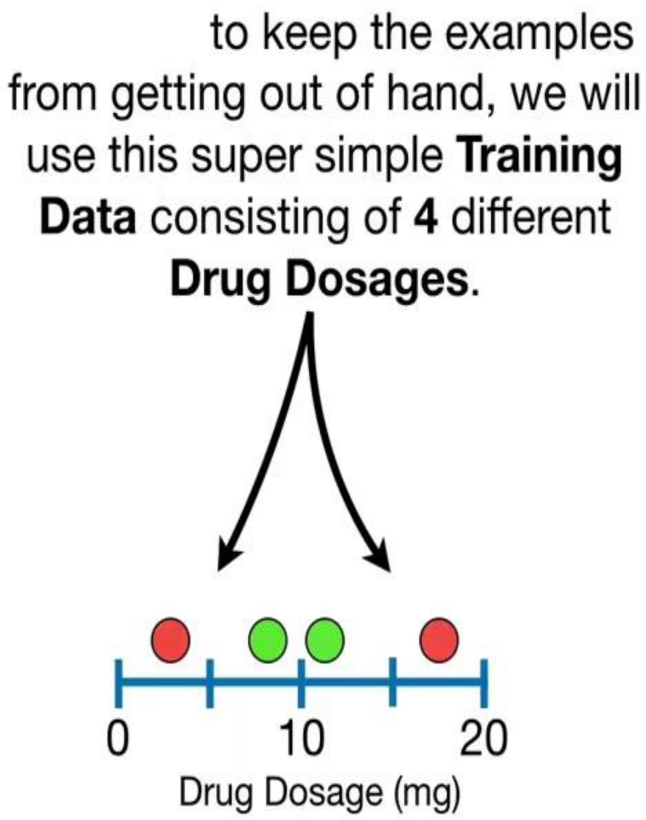


Building More Trees



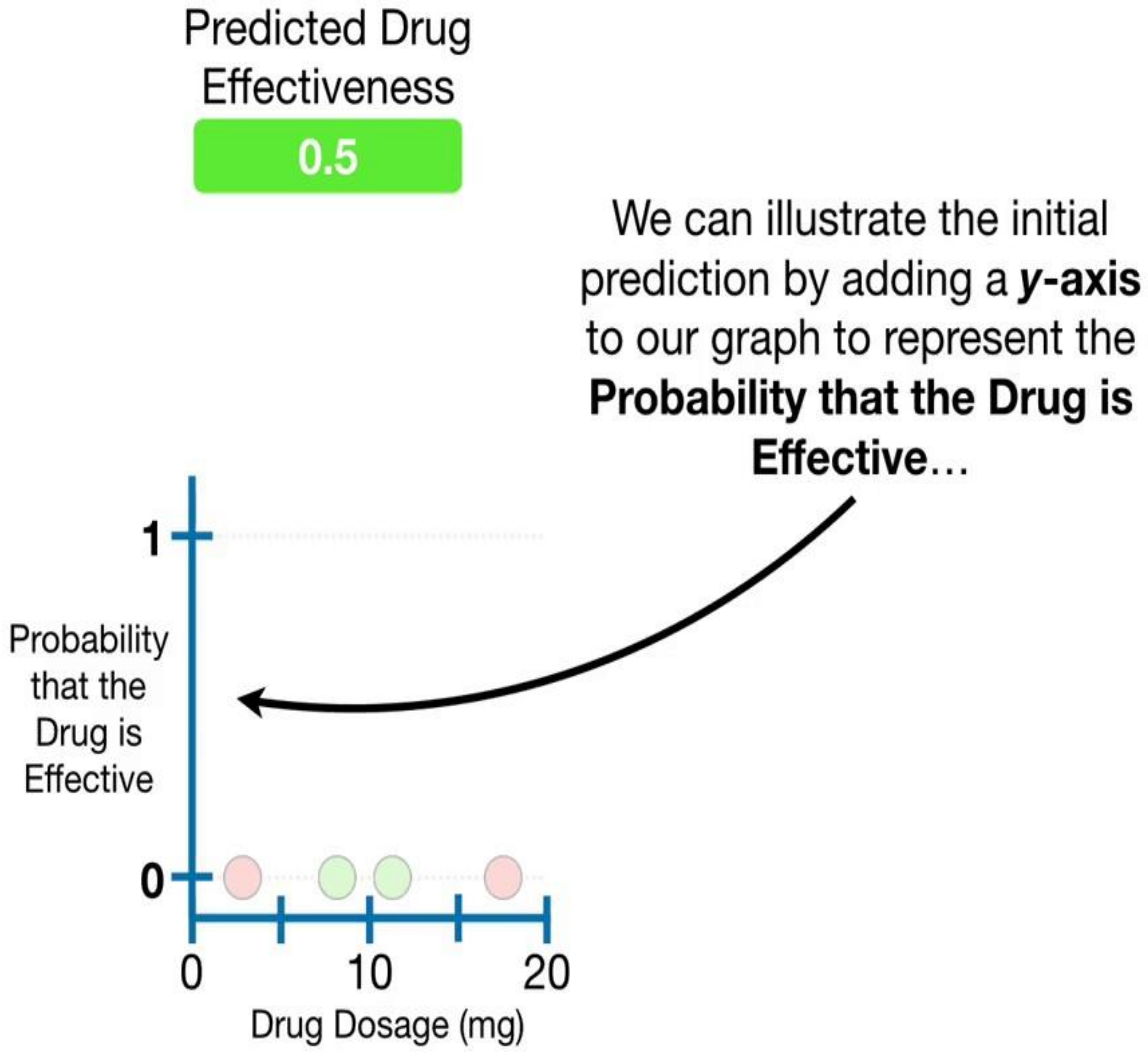
...and we keep building trees until the **Residuals** are super small, or we have reached the maximum number.

XGBoost for Classification

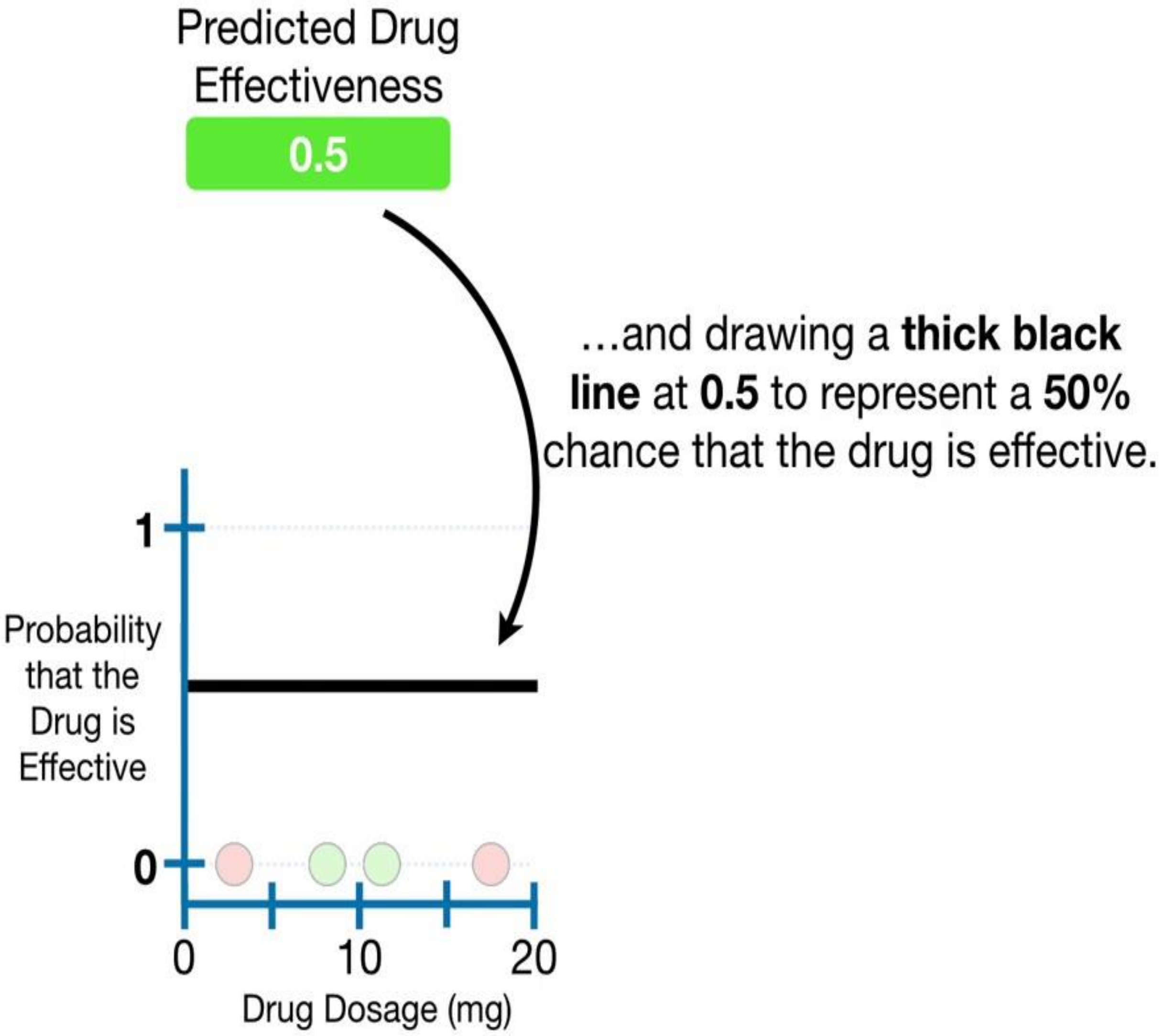


XGBoost for Classification

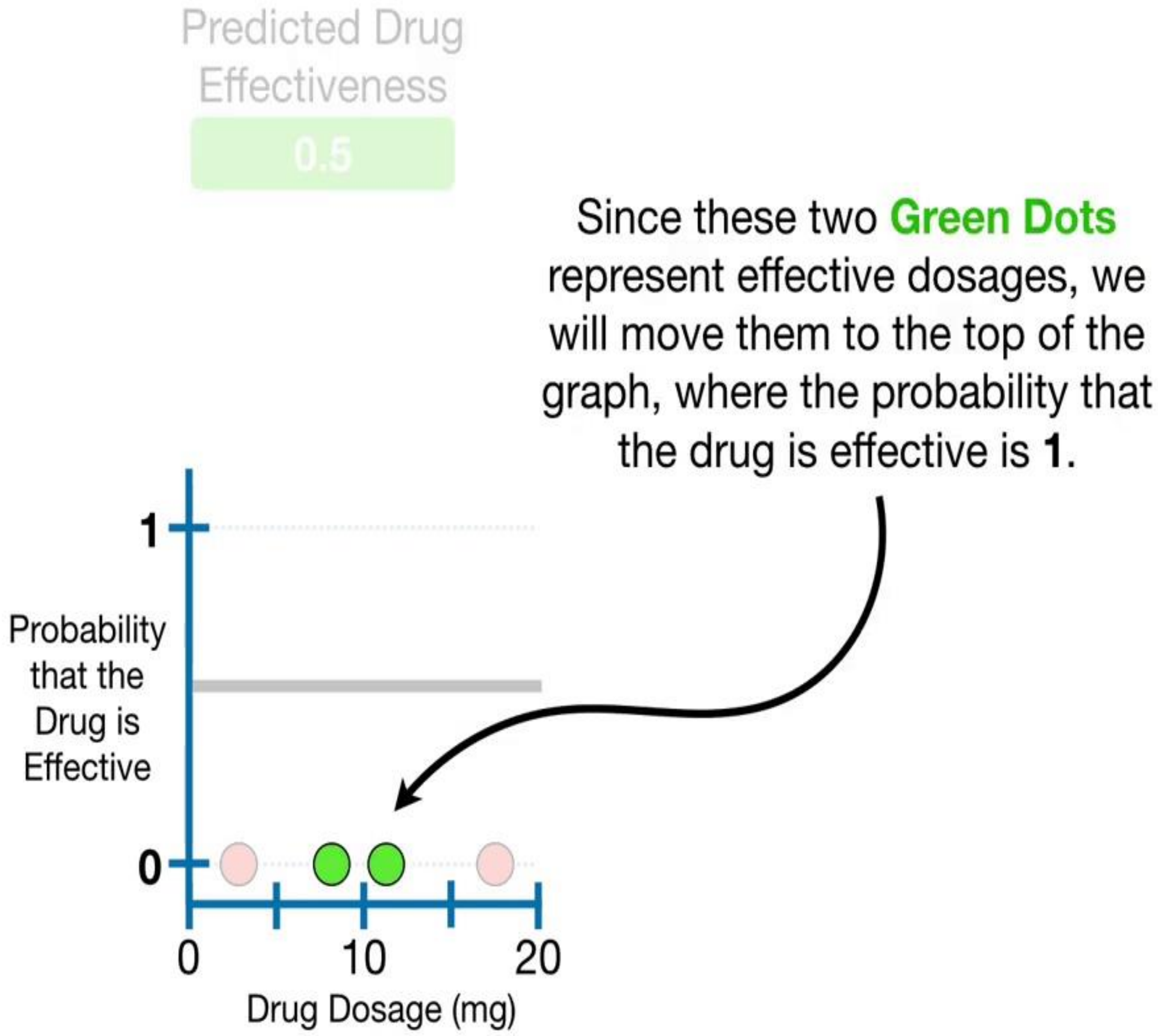
XGBoost for Classification



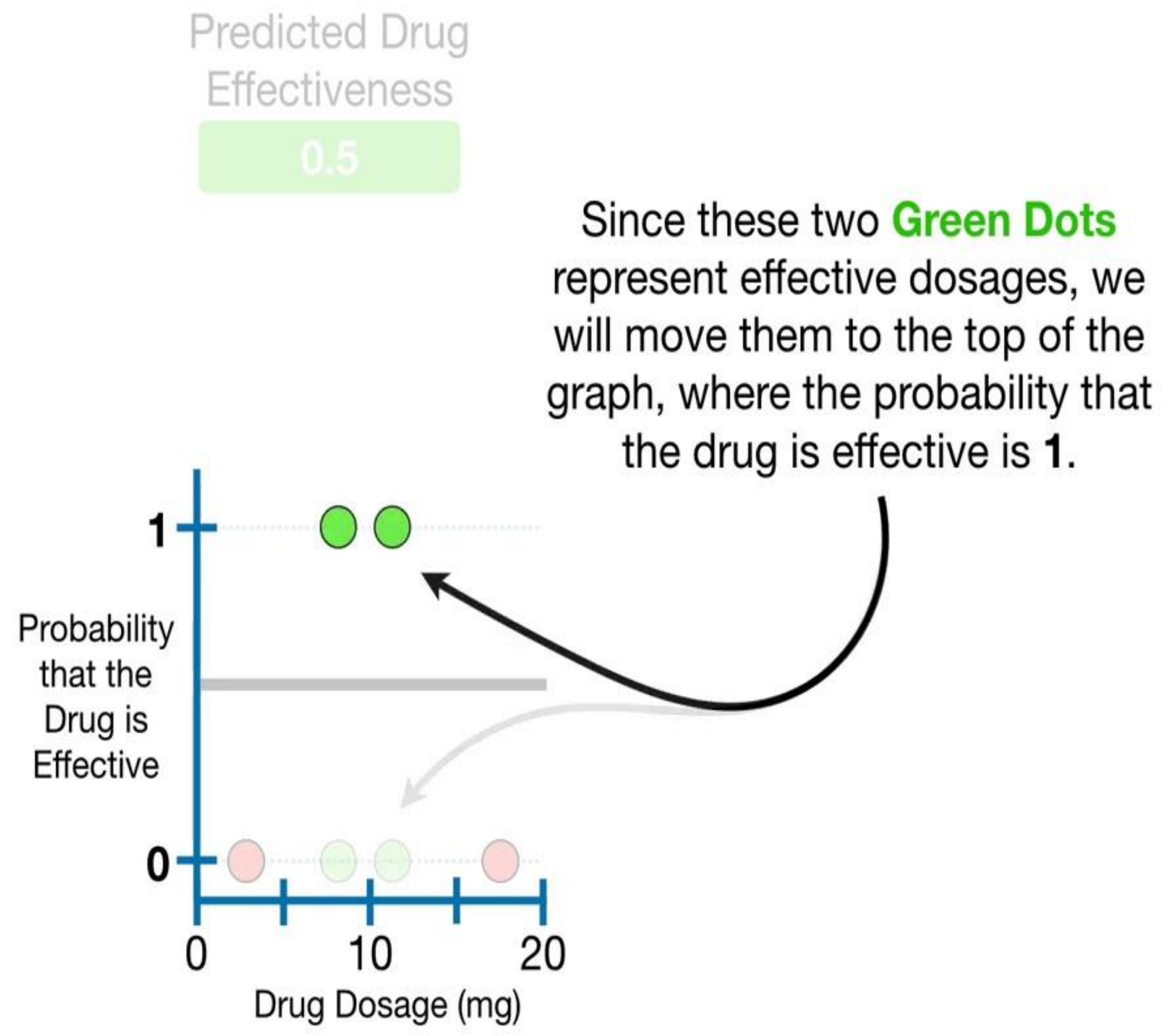
XGBoost for Classification



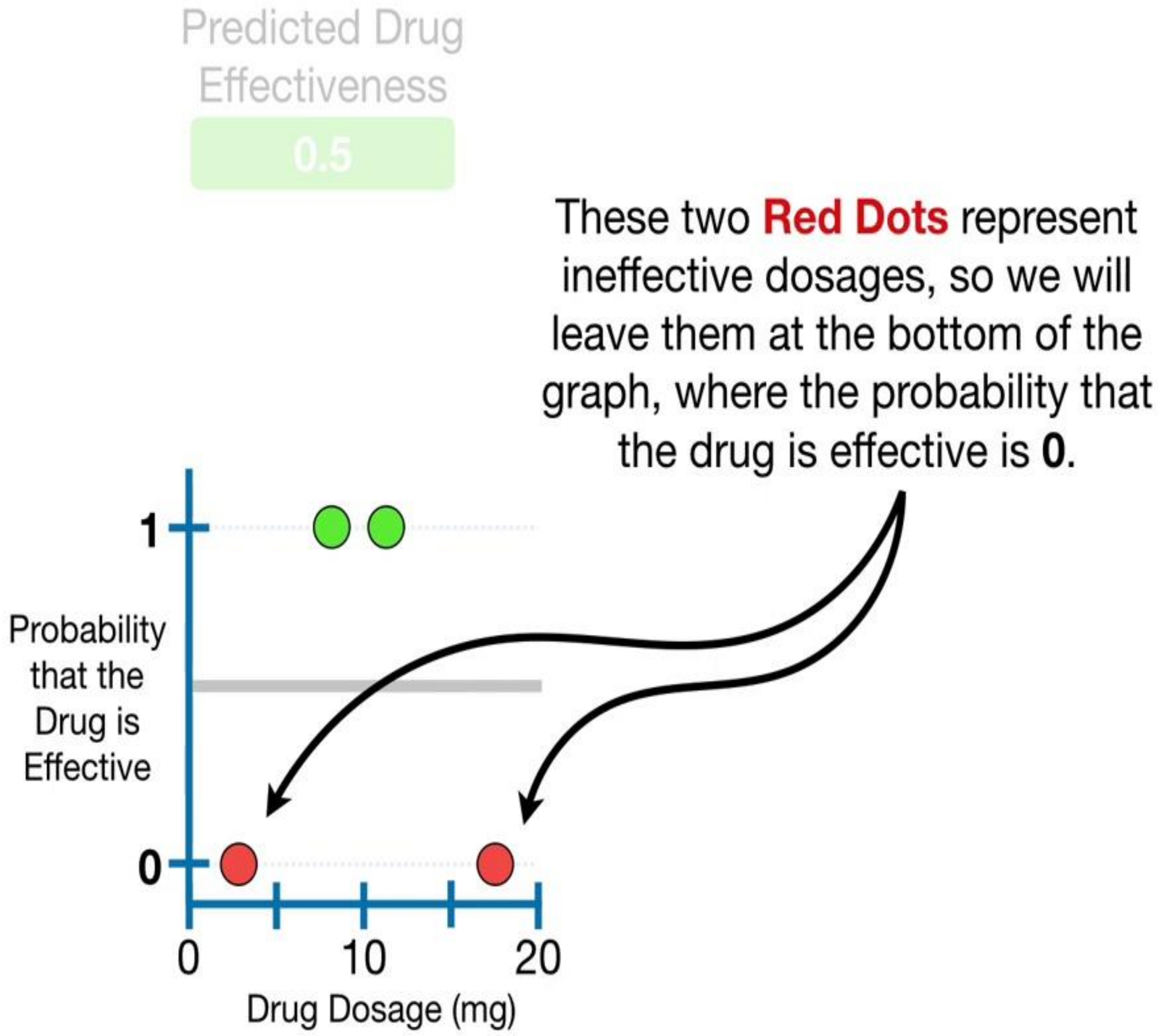
XGBoost for Classification



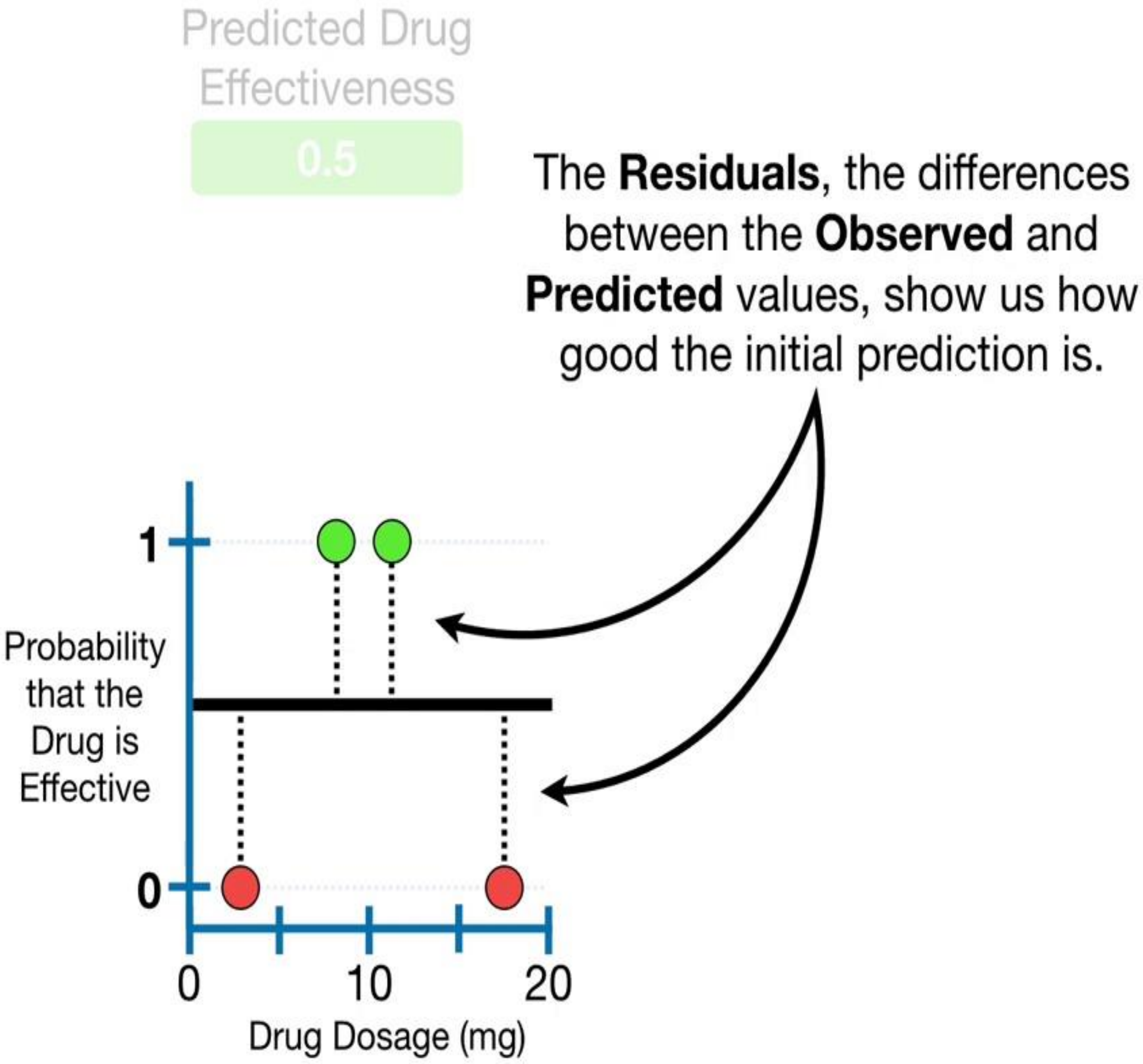
XGBoost for Classification



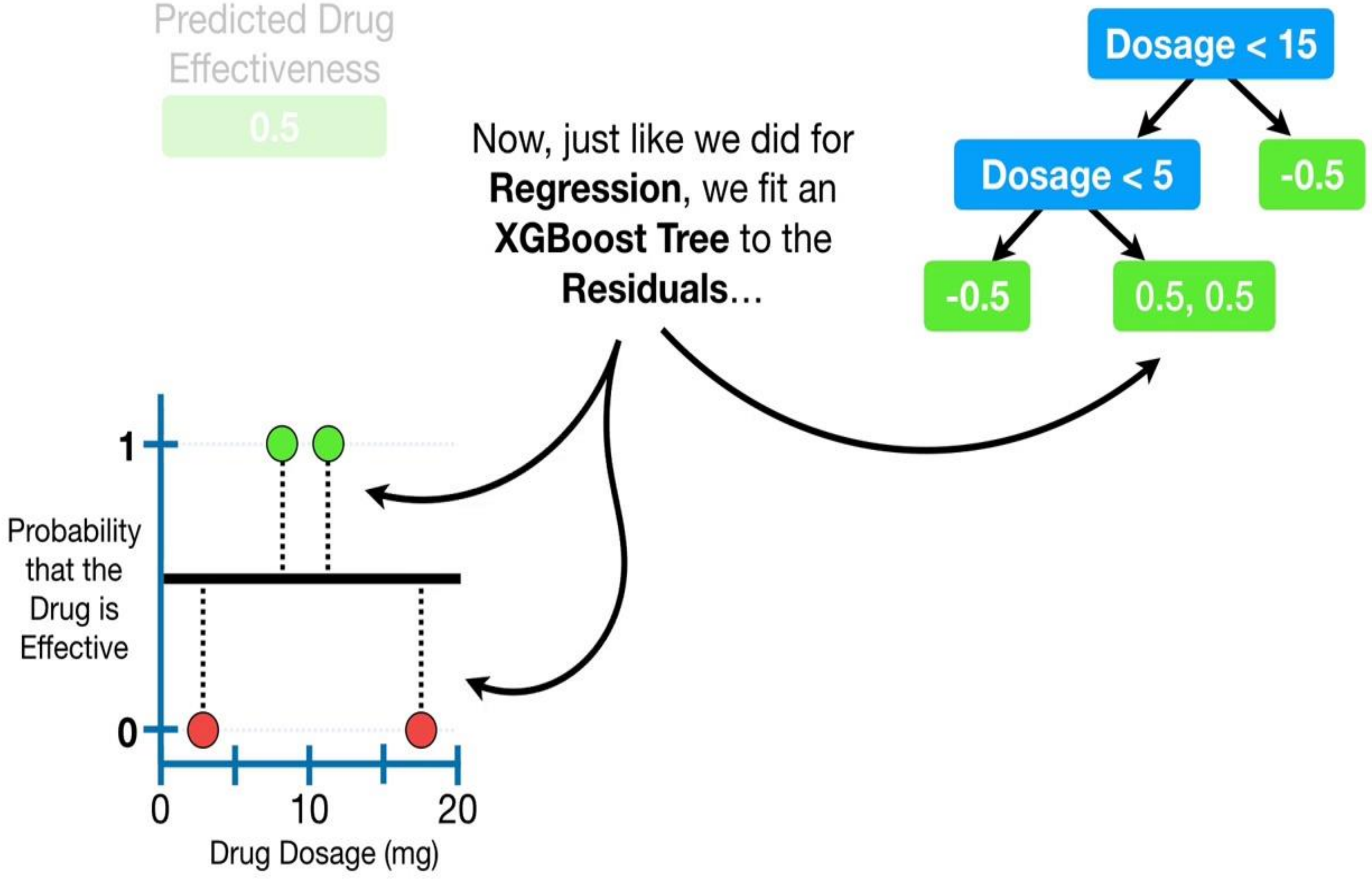
XGBoost for Classification



XGBoost for Classification

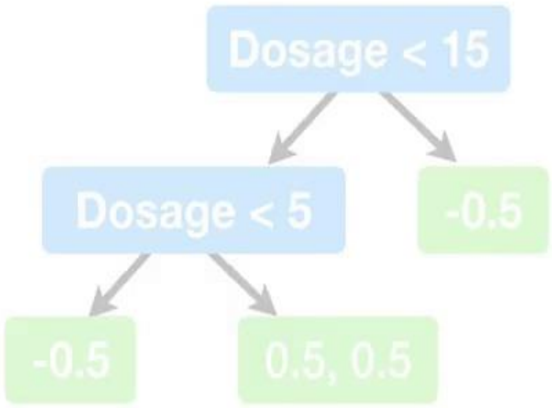


XGBoost for Classification

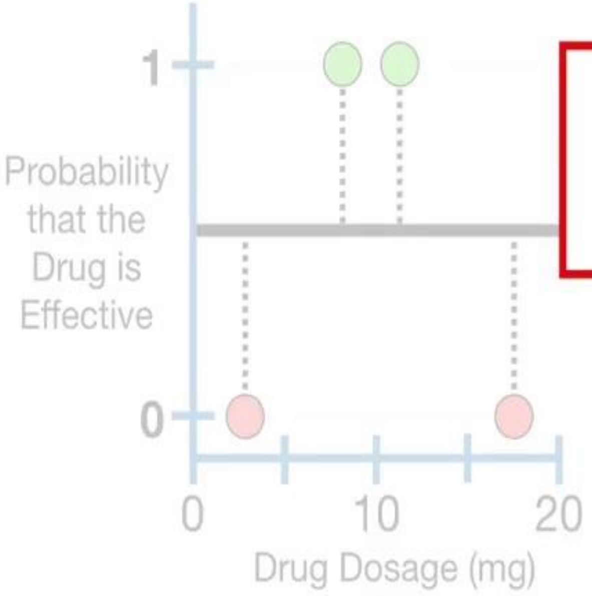


XGBoost for Classification

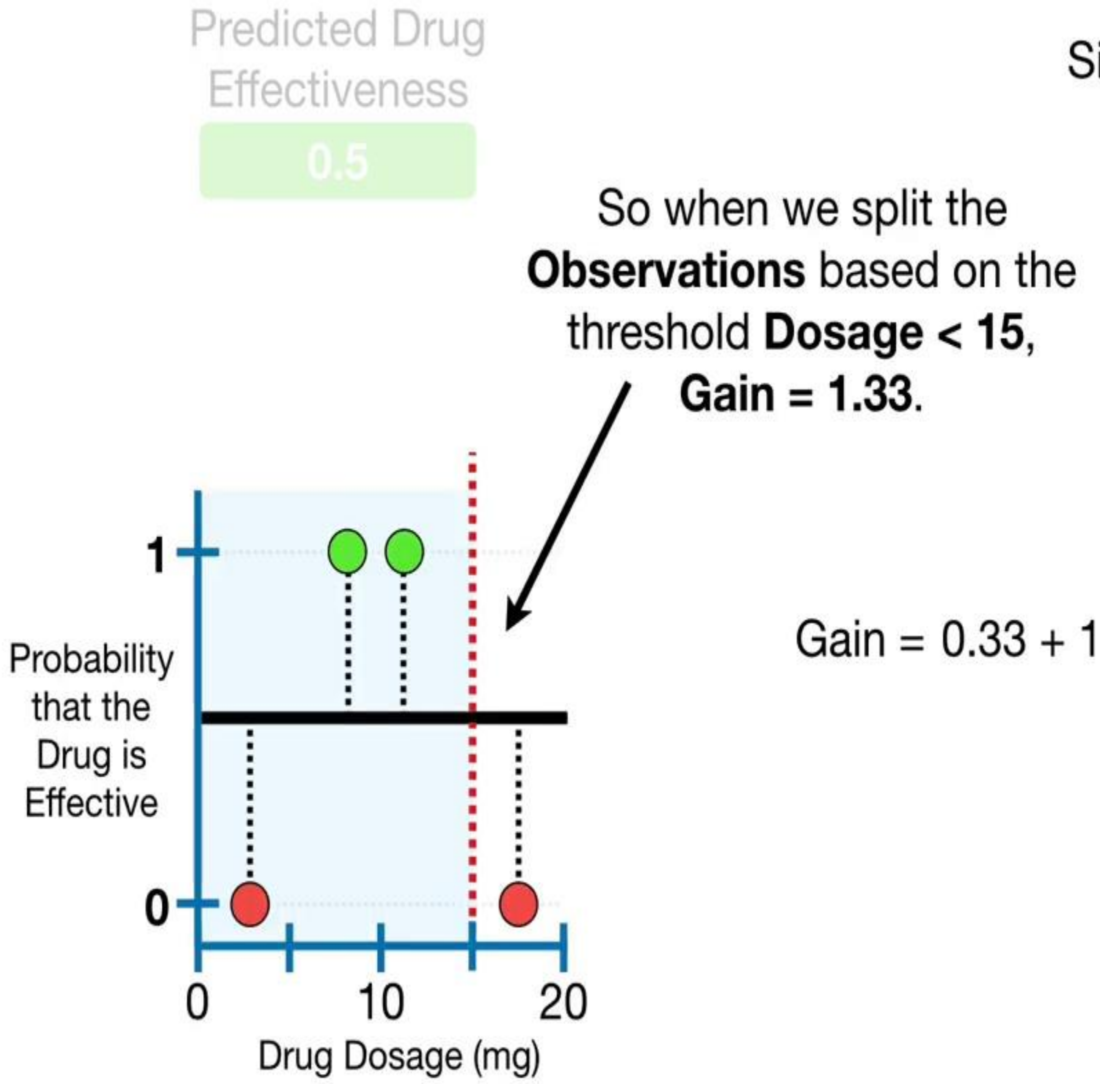
...however, since we are using **XGBoost** for **Classification**, we have a new formula for the **Similarity Scores**.

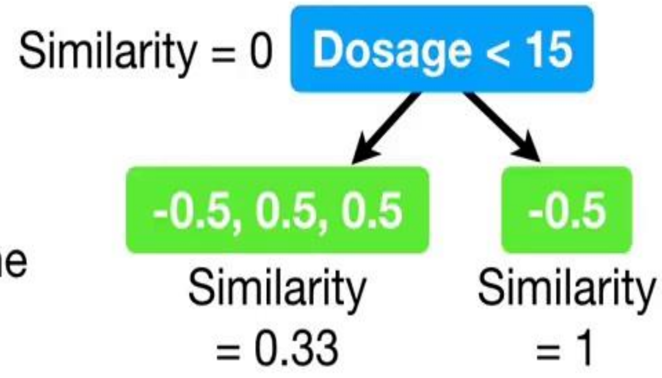


$$\frac{\left(\sum \text{Residual}_i\right)^2}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right] + \lambda}$$



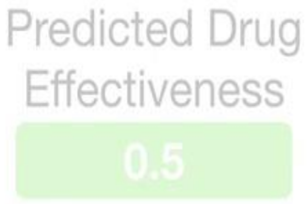
XGBoost for Classification

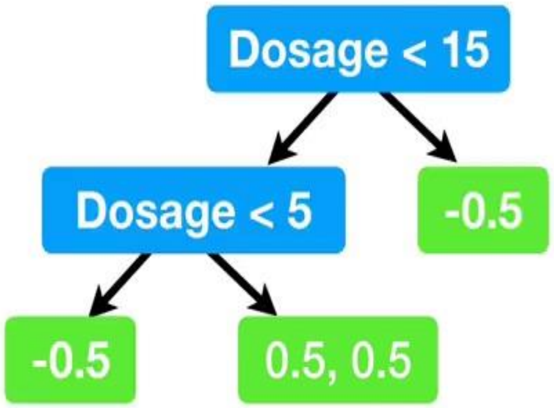




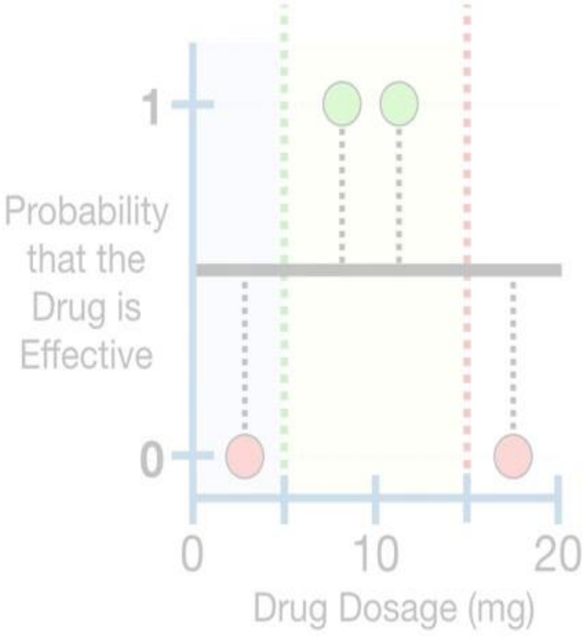
Gain = 0.33 + 1 - 0 = 1.33

XGBoost for Classification

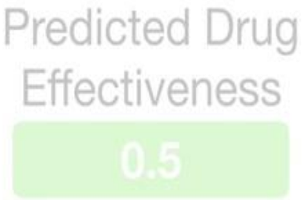


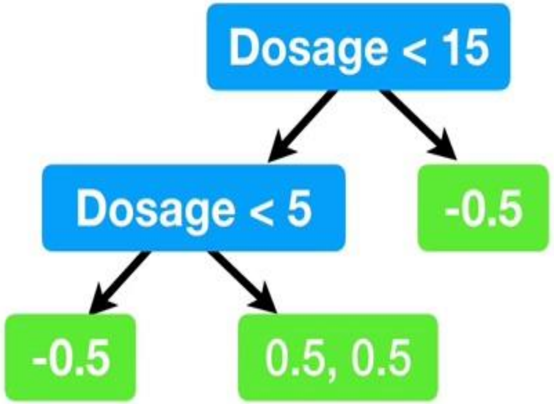


Now, since I'm limiting trees to **2** levels, we will not split this leaf any further, and we are done building this tree.

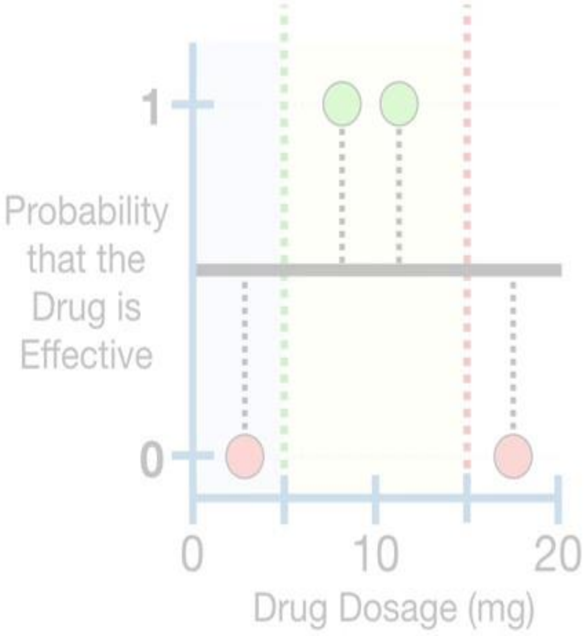


XGBoost for Classification

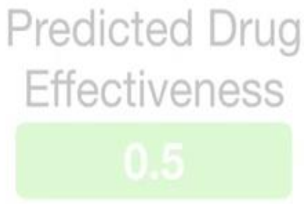


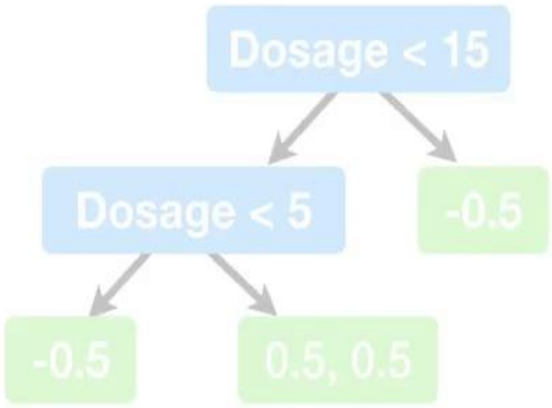


The minimum number of **Residuals** in each leaf is determined by calculating something called **Cover**.

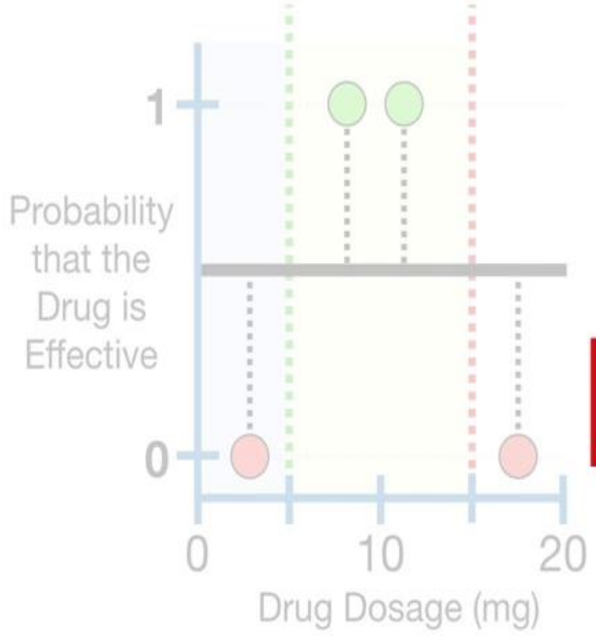


XGBoost for Classification



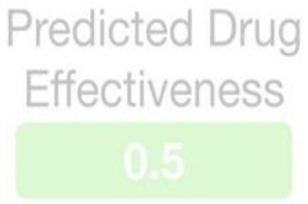


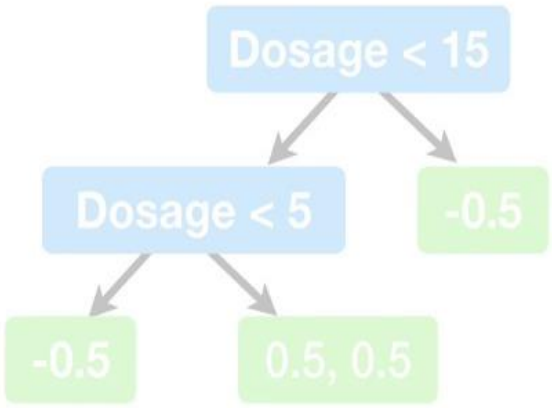
Cover is defined as the denominator of the **Similarity Score** minus **λ (lambda)**.



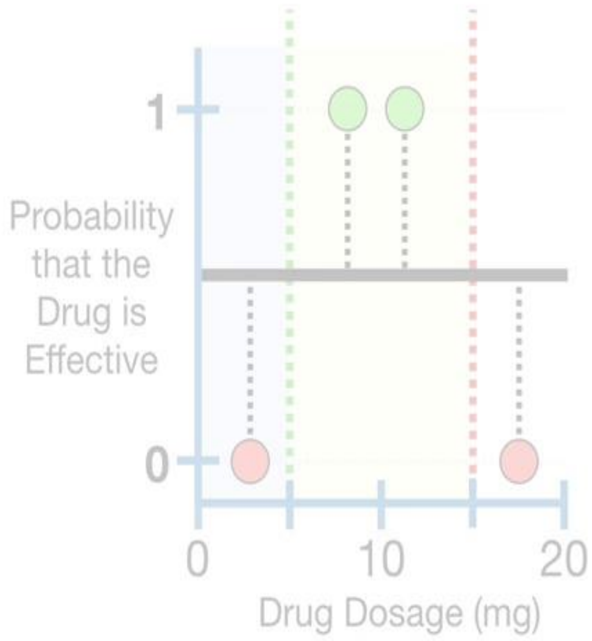
$$\text{Similarity} = \frac{\left(\sum \text{Residual}_i\right)^2}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right] + \lambda}$$

XGBoost for Classification



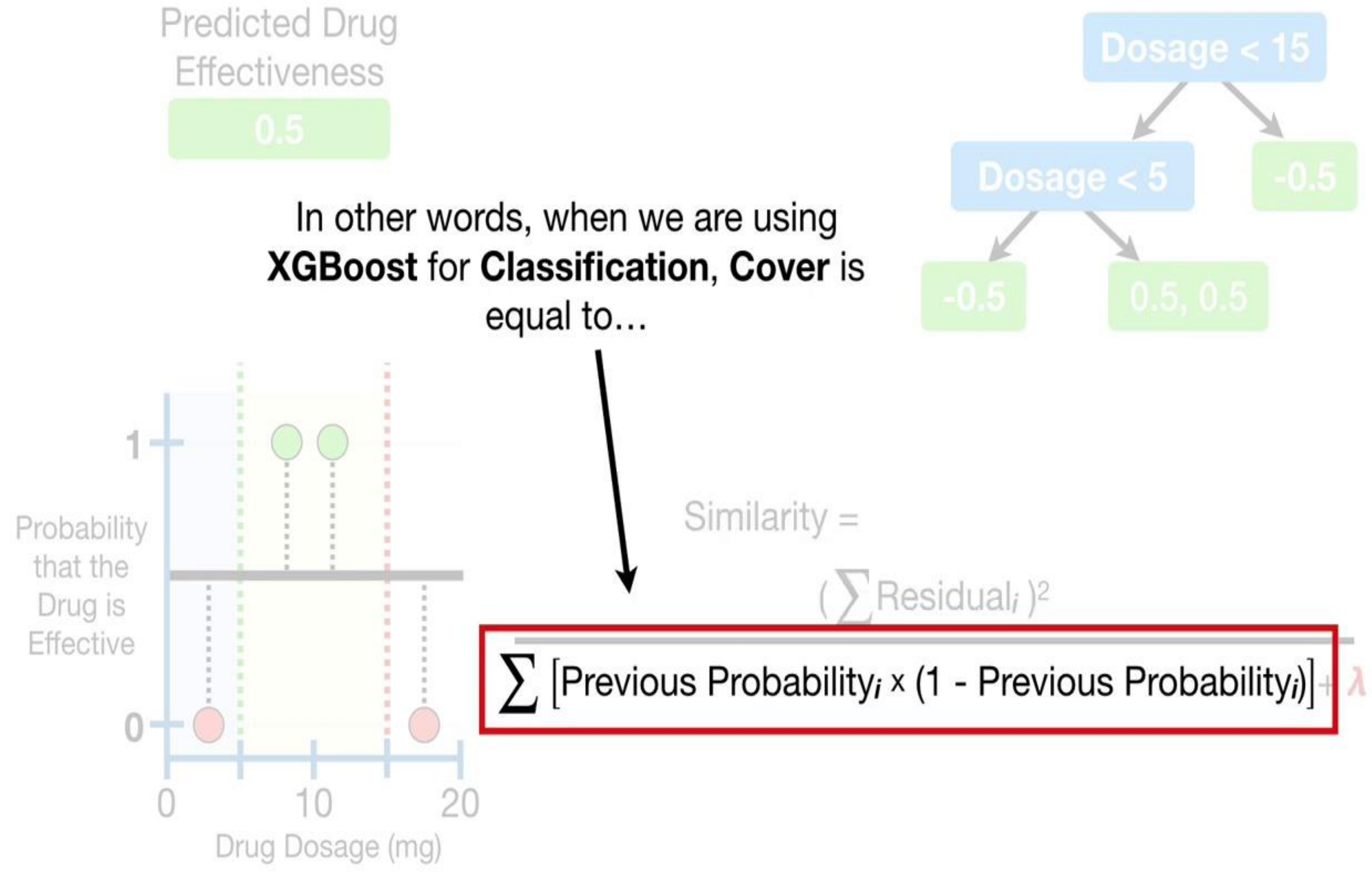


In other words, when we are using **XGBoost** for **Classification**, **Cover** is equal to…

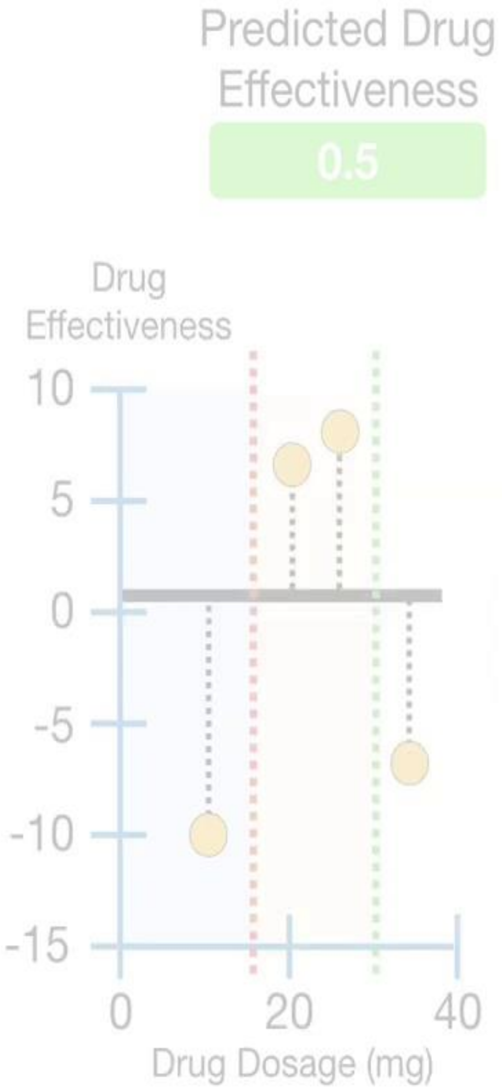


$$\text{Similarity} = \frac{\left(\sum \text{Residual}_i\right)^2}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right] + \lambda}$$

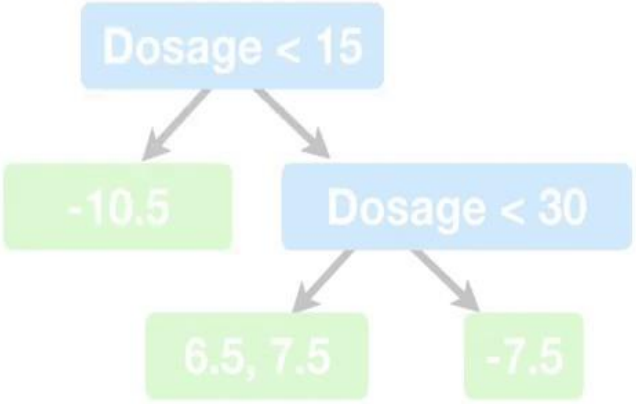
XGBoost for Classification



XGBoost for Classification

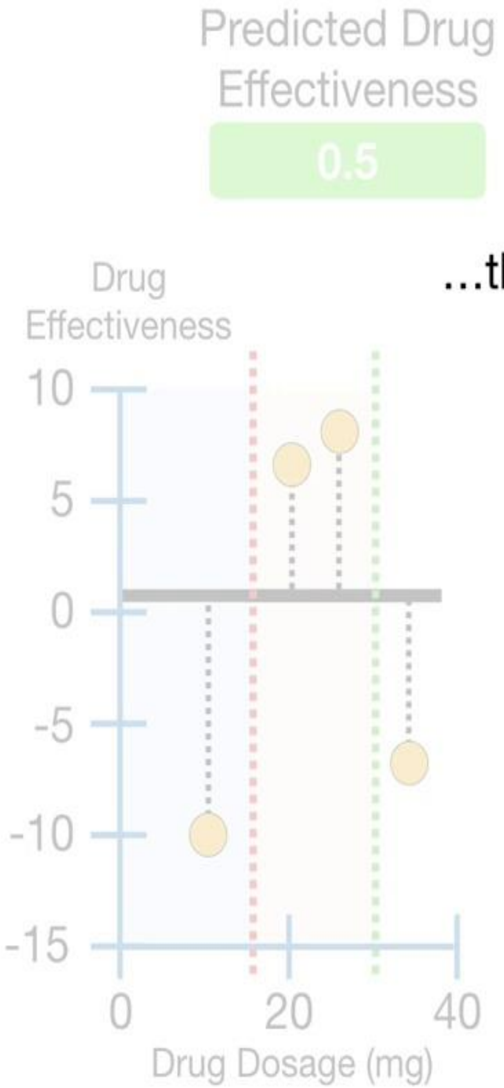


In contrast, when **XGBoost** is used for **Regression** and we are using this formula for the **Similarity Score**...

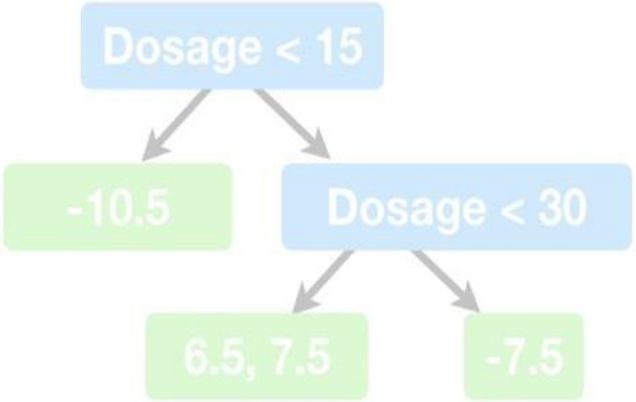


$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

XGBoost for Classification

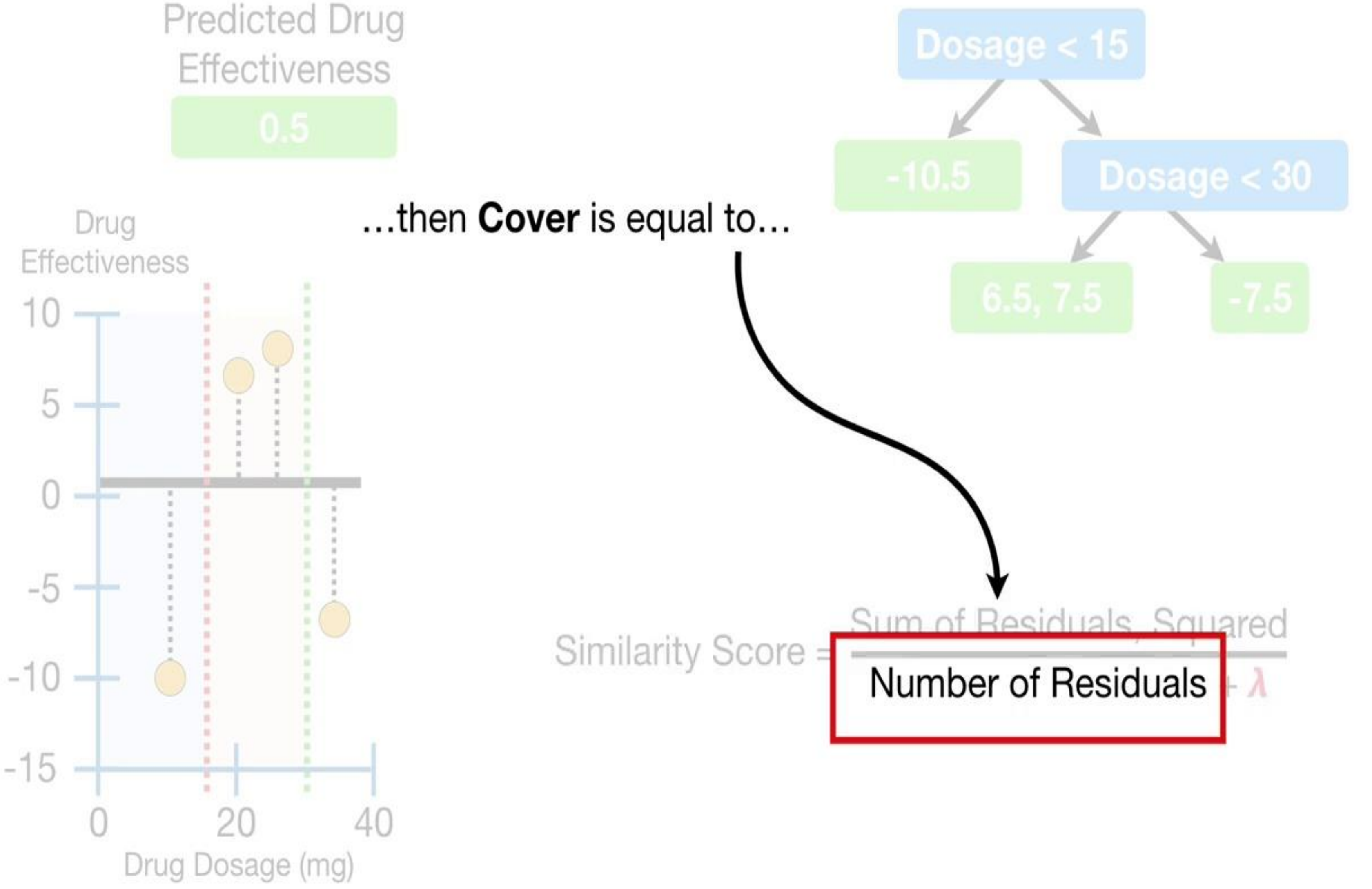


...then **Cover** is equal to...

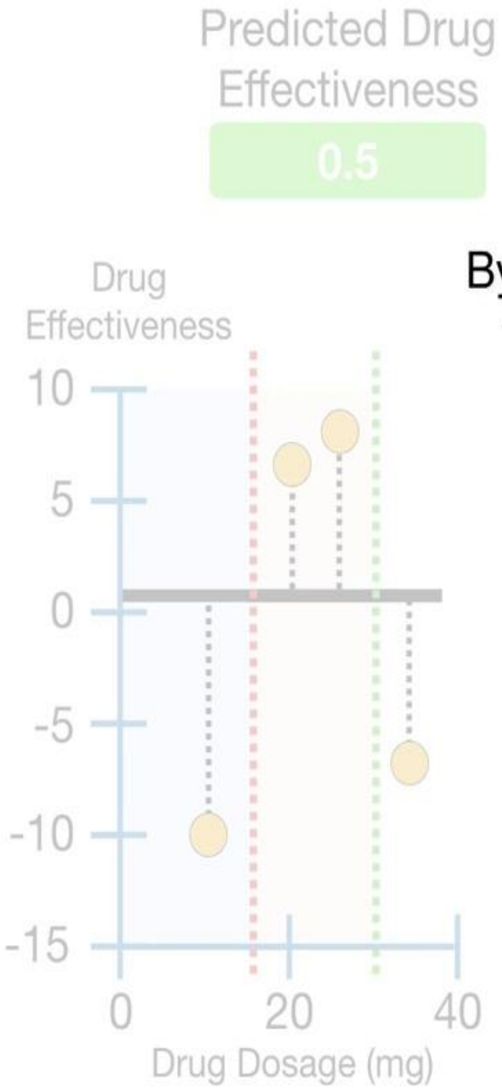


$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

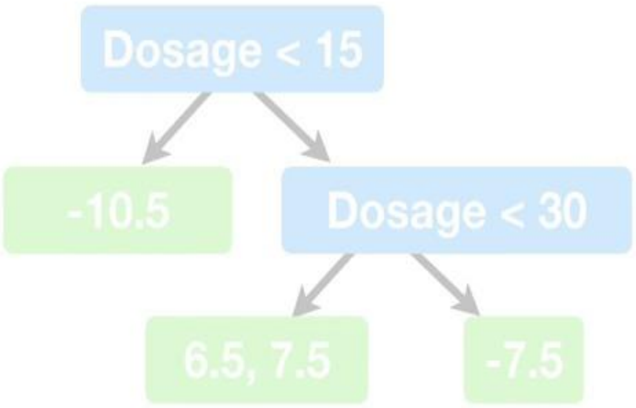
XGBoost for Classification



XGBoost for Classification

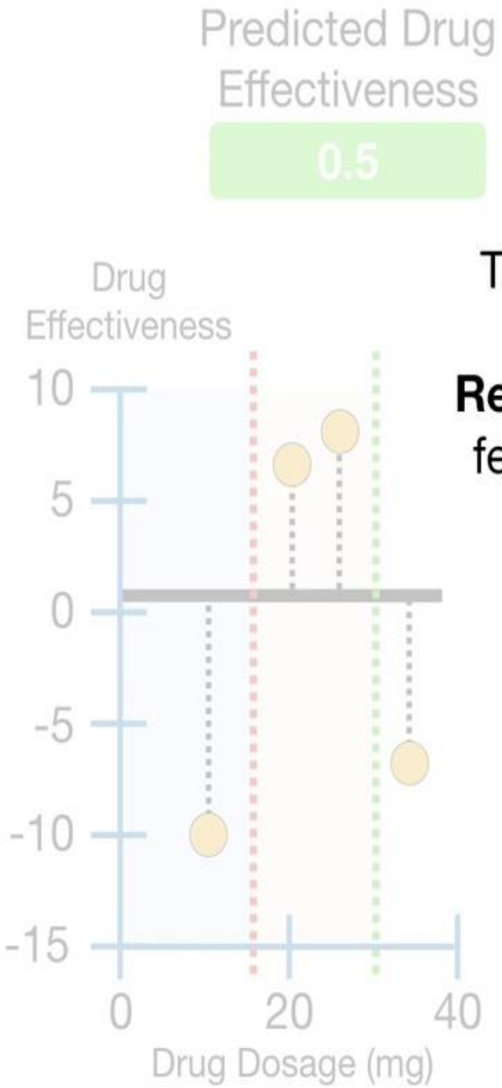


By default, the minimum value for **Cover** is **1**.

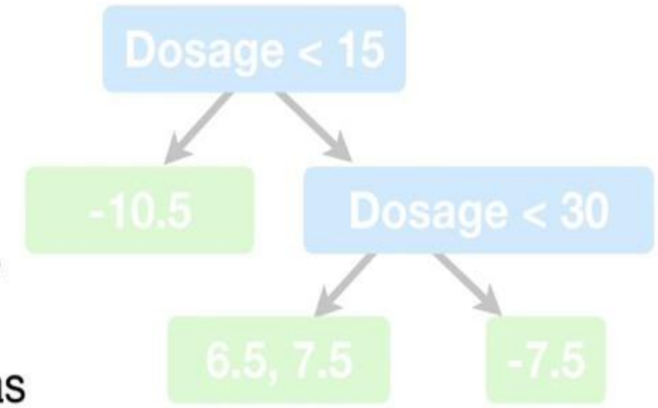


$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

XGBoost for Classification

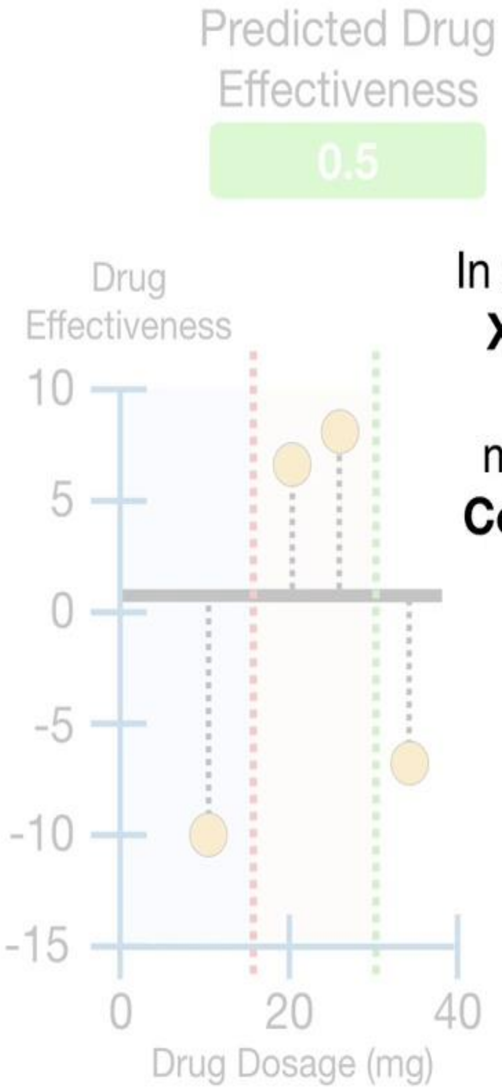


Thus, by default, when we use **XGBoost** for **Regression**, we can have as few as **1 Residual** per leaf.

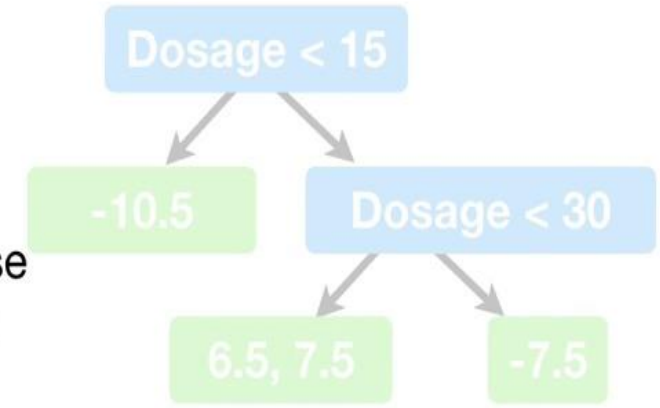


$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

XGBoost for Classification

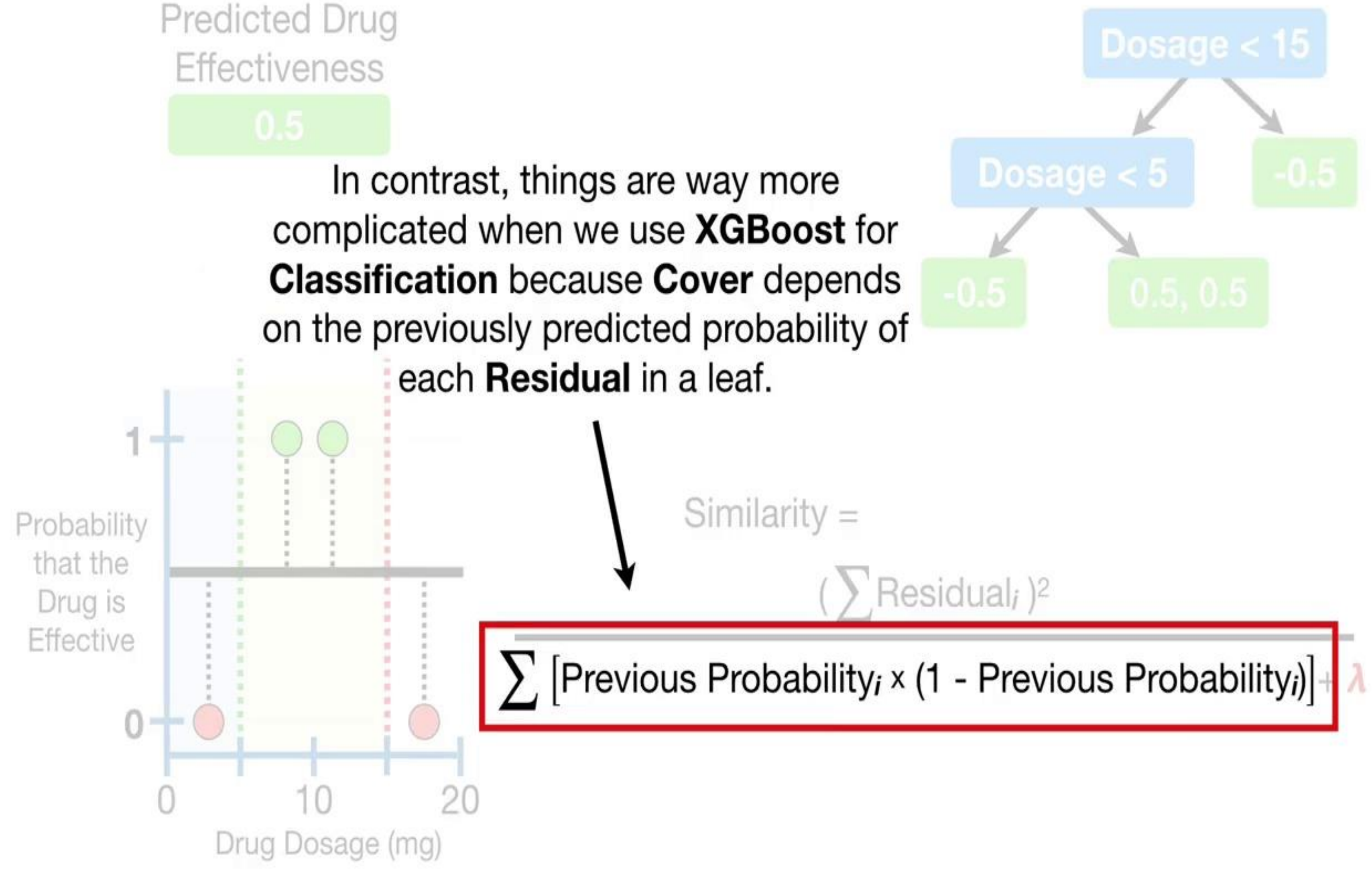


In other words, when we use **XGBoost** for **Regression** and use the default minimum value for **Cover**, **Cover** has no effect on how we grow the tree.

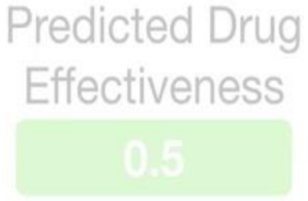


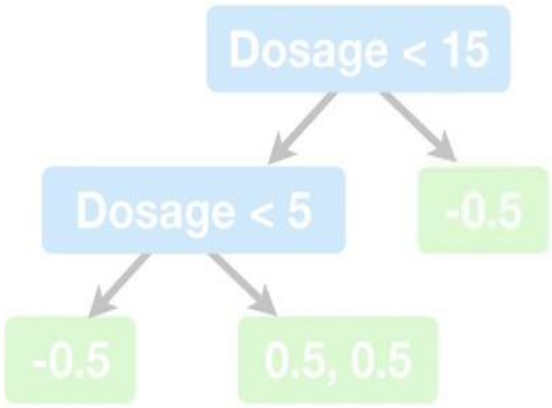
$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

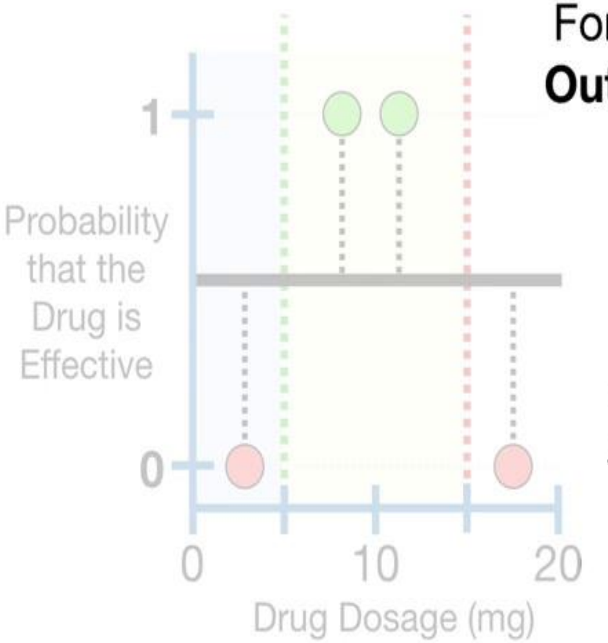
XGBoost for Classification



XGBoost for Classification



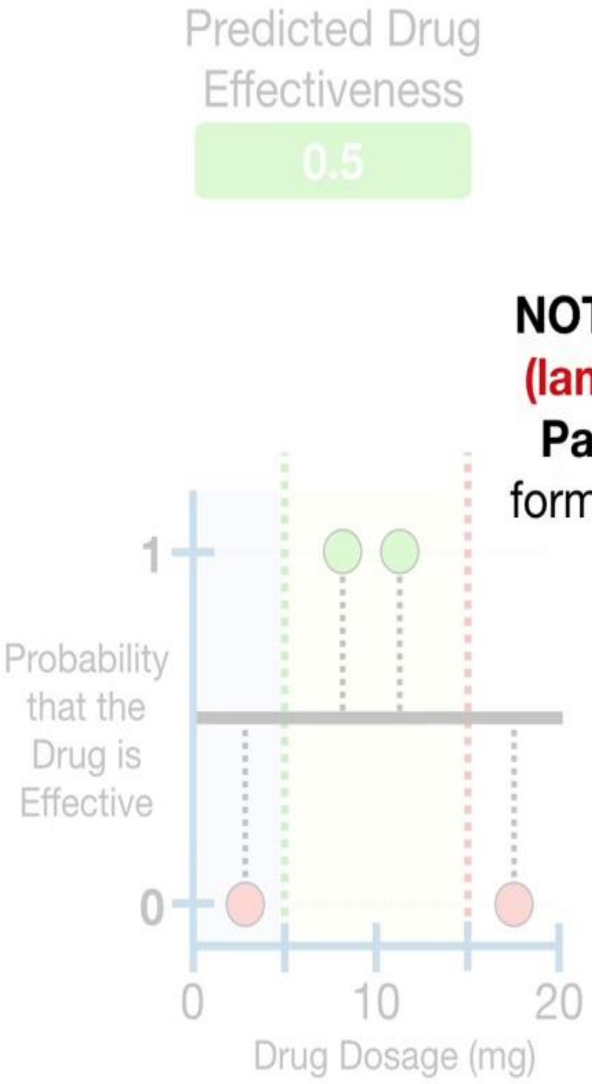


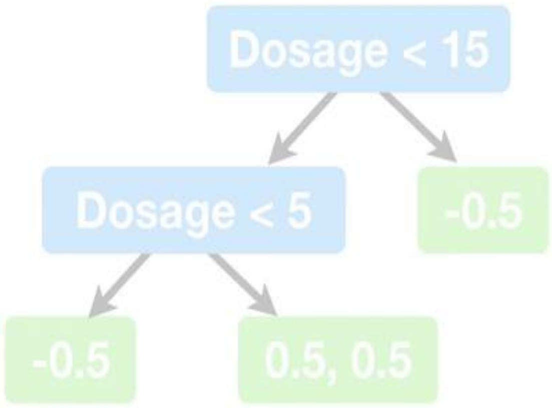


For **Classification**, the **Output Value** for a leaf is…

$$\frac{\left(\sum \text{Residual}_i\right)}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right] + \lambda}$$

XGBoost for Classification

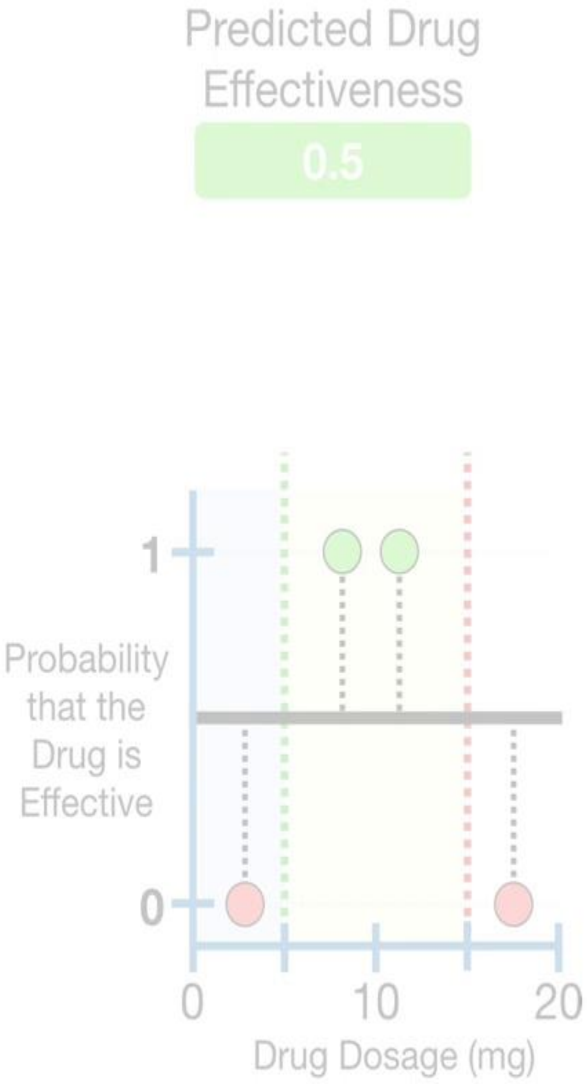




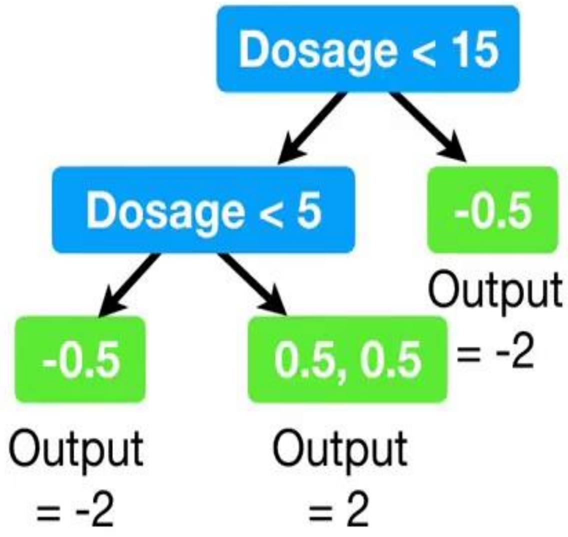
NOTE: With the exception of λ (lambda), the **Regularization Parameter**, this is the same formula we used for unextreme **Gradient Boost**.

$$\frac{\left(\sum \text{Residual}_i\right)}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right] + \lambda}$$

XGBoost for Classification



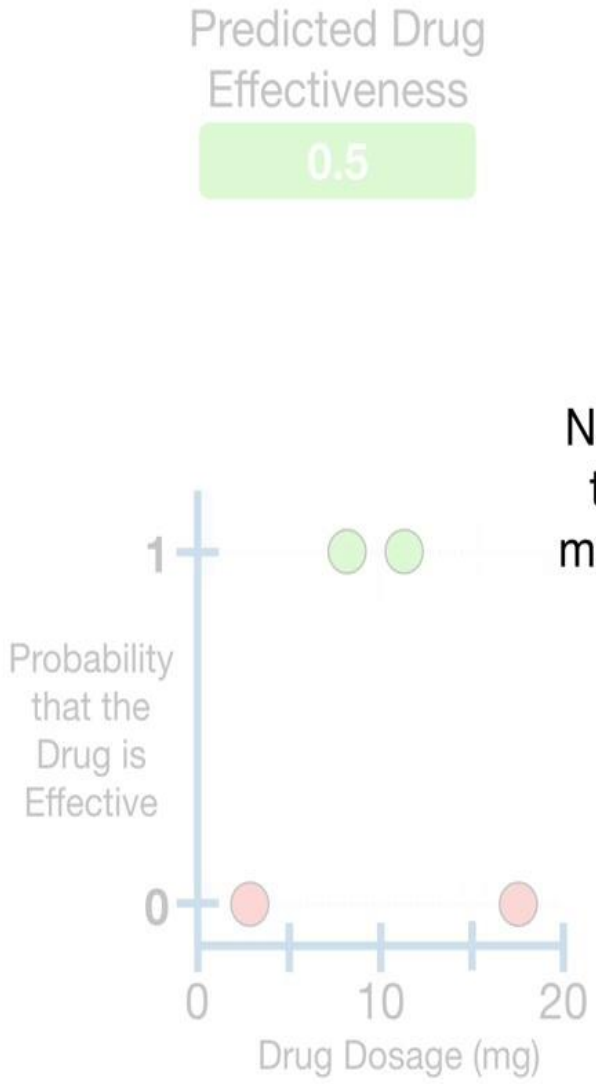
The first tree is complete!



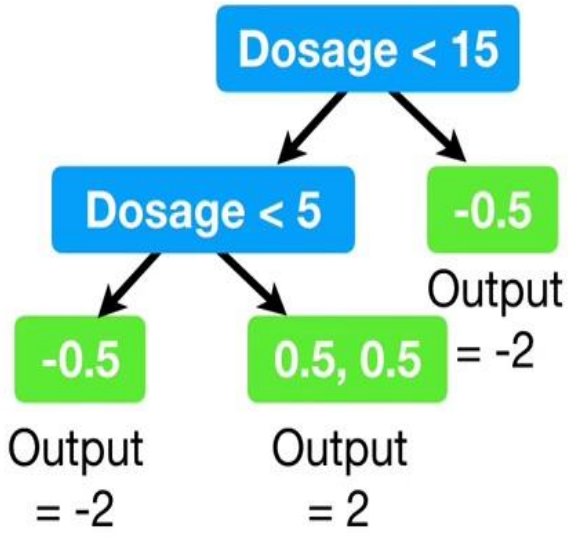
-0.5/(0.5 x (1-0.5) + 0) = -2

$$\frac{\left(\sum \text{Residual}_i\right)}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right] + \lambda}$$

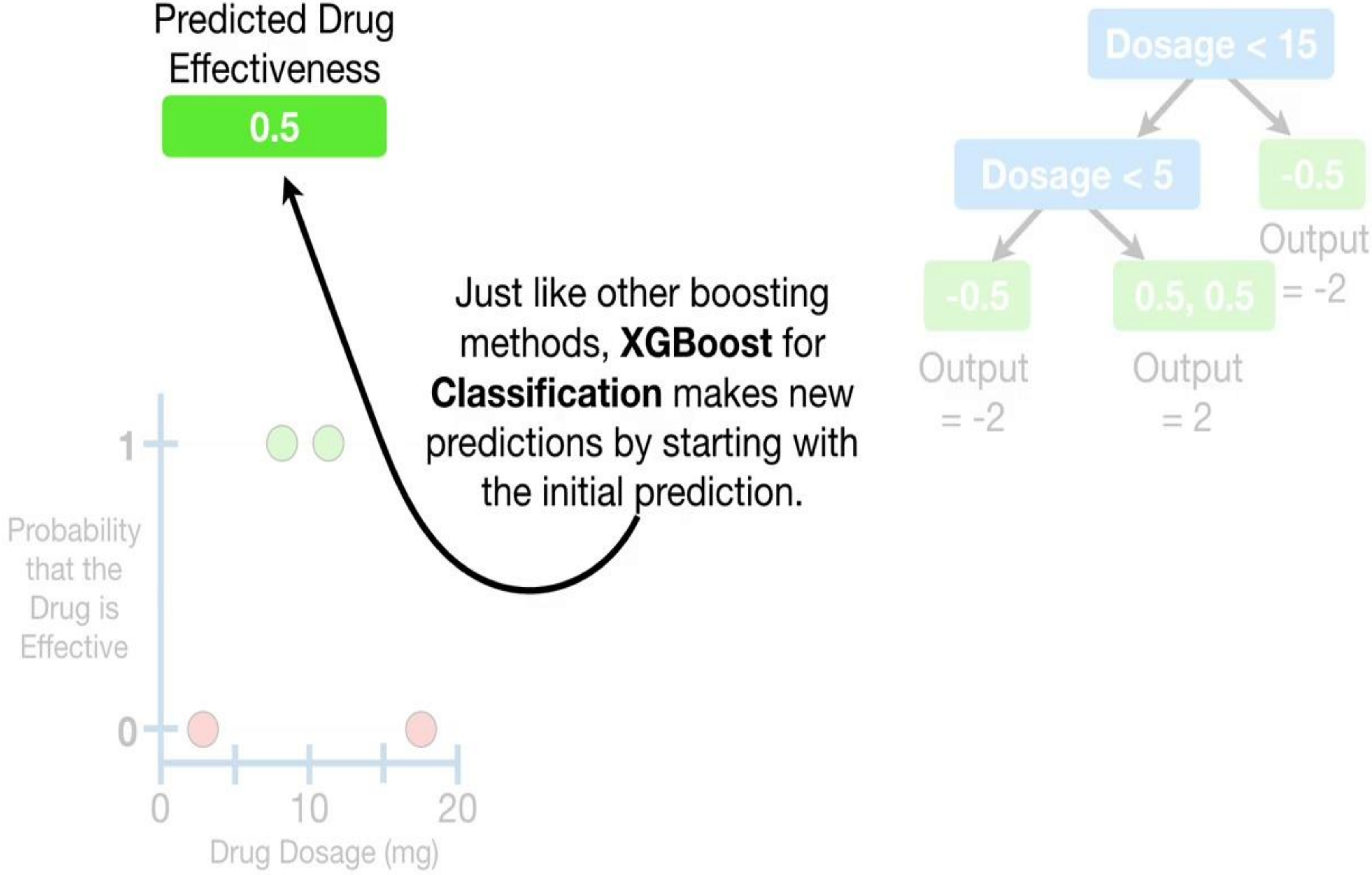
XGBoost for Classification



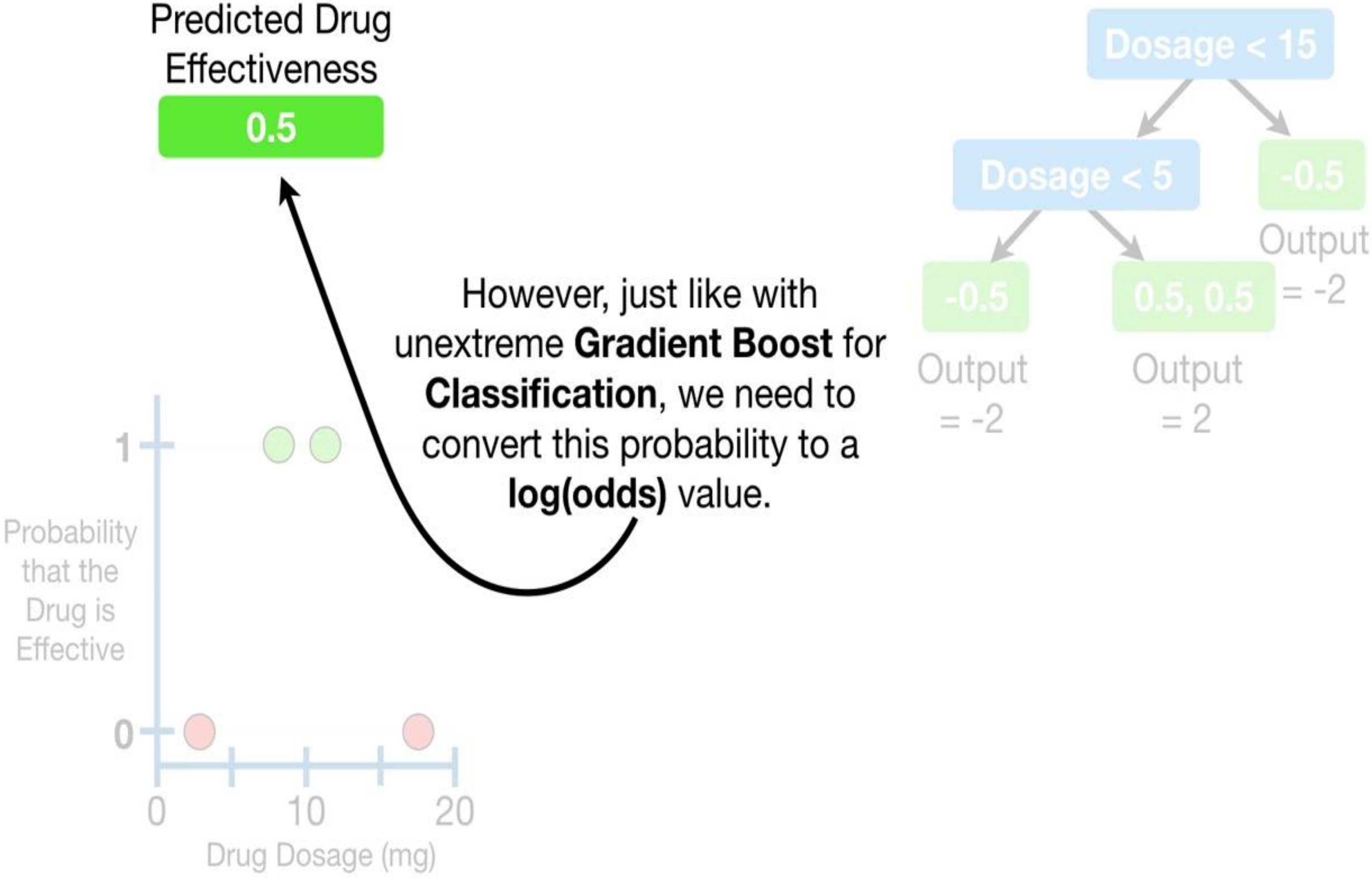
Now that we have built the first tree, we can make new **Predictions**.



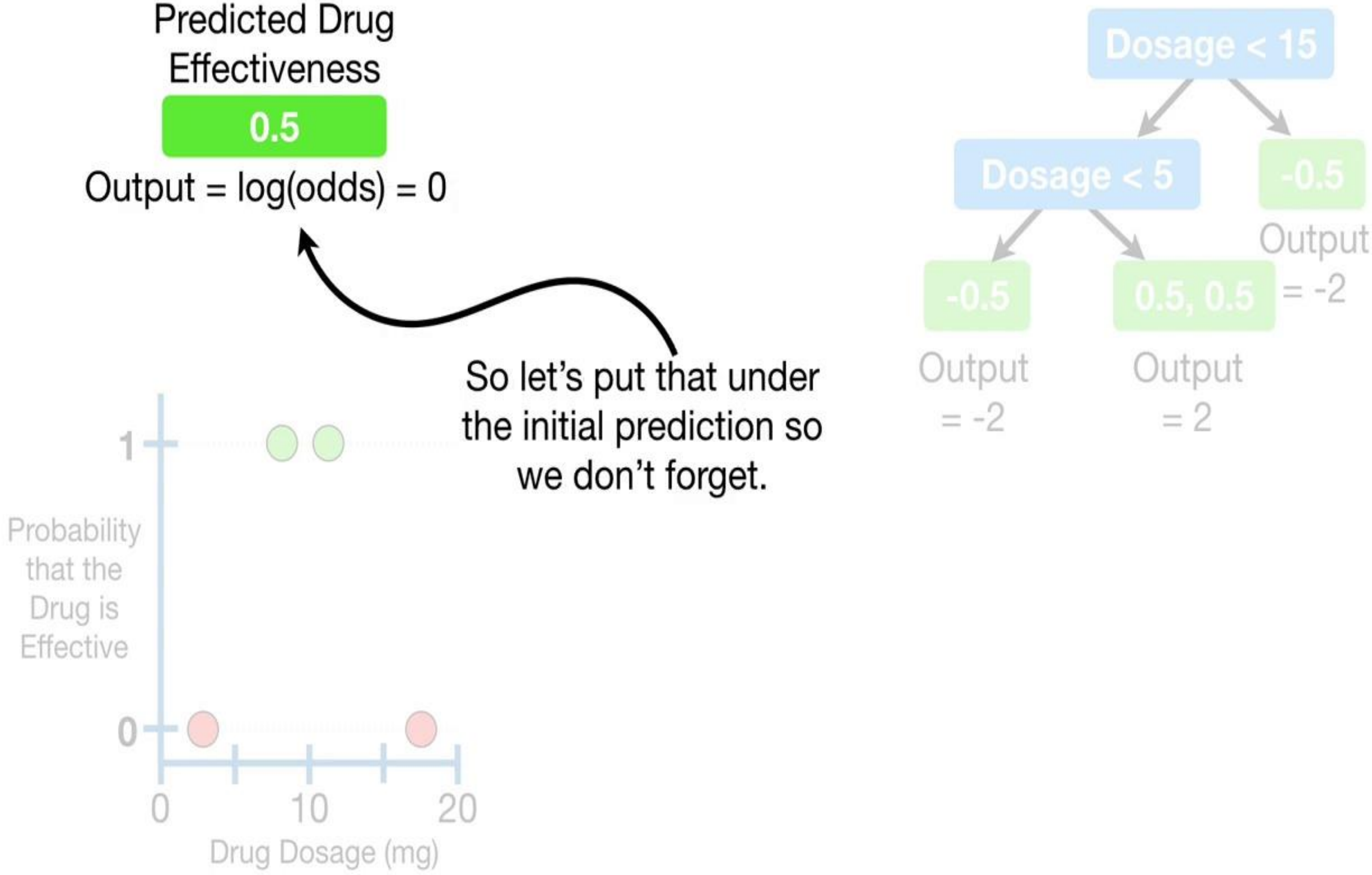
XGBoost for Classification



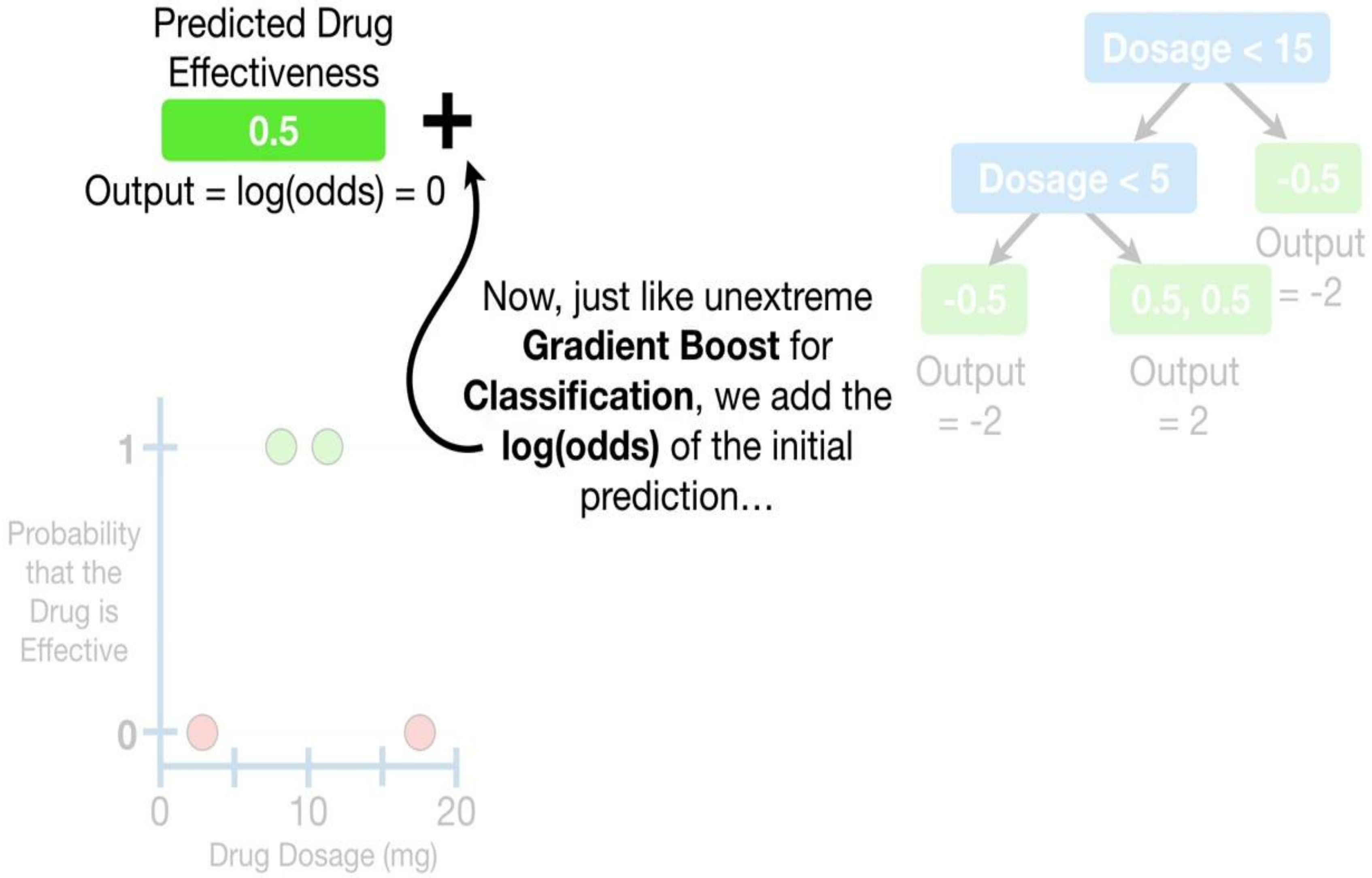
XGBoost for Classification



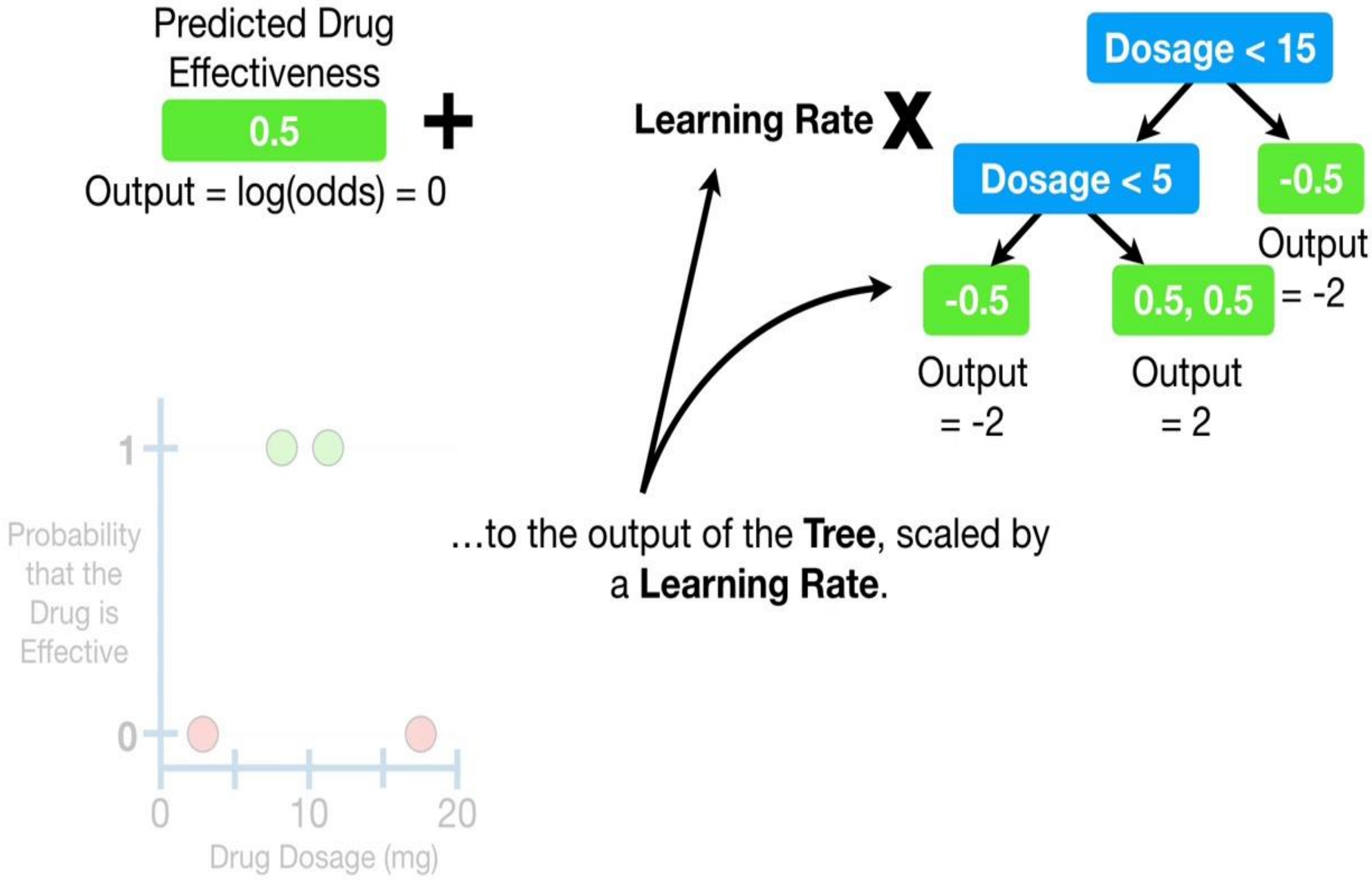
XGBoost for Classification



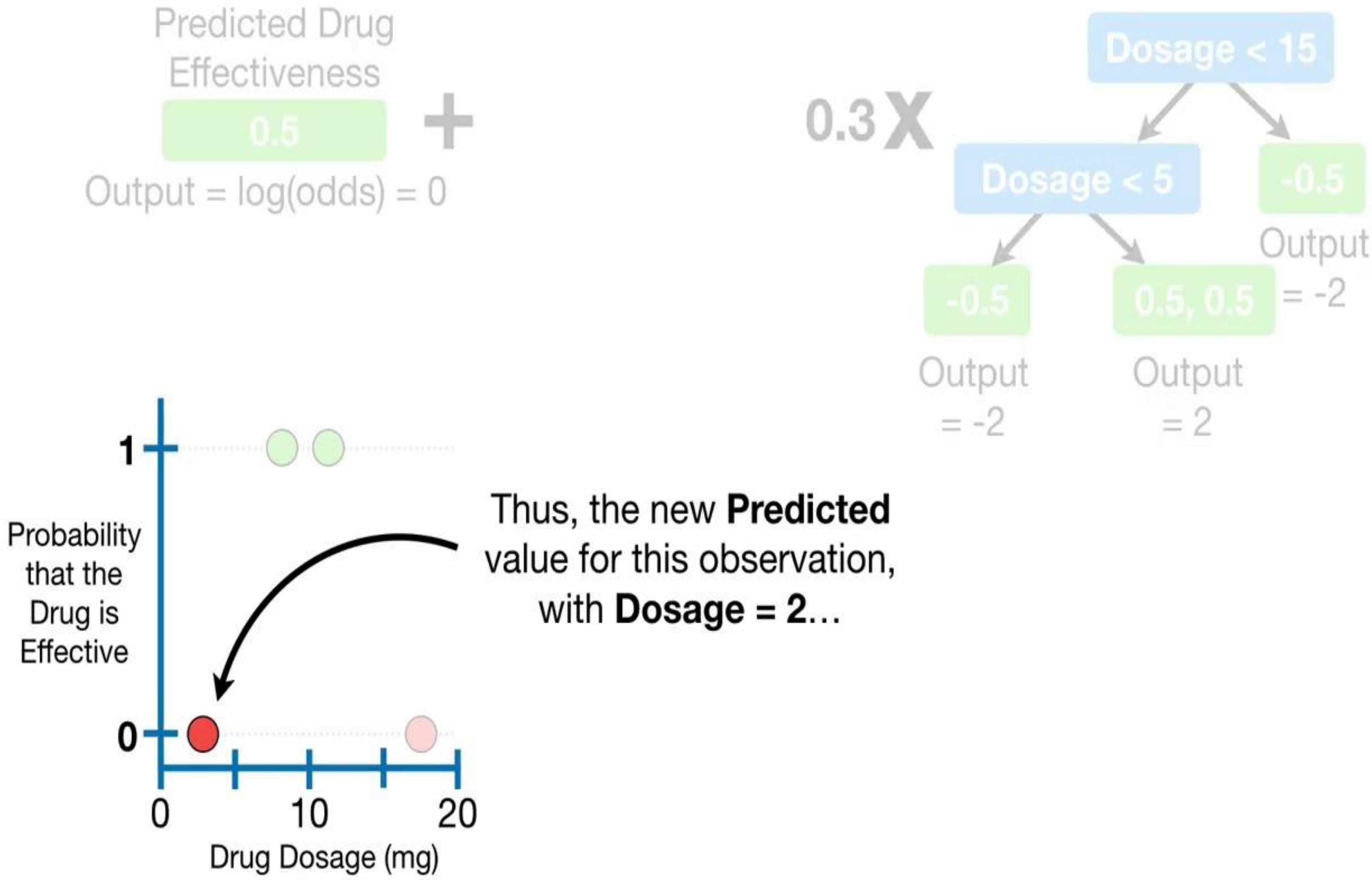
XGBoost for Classification



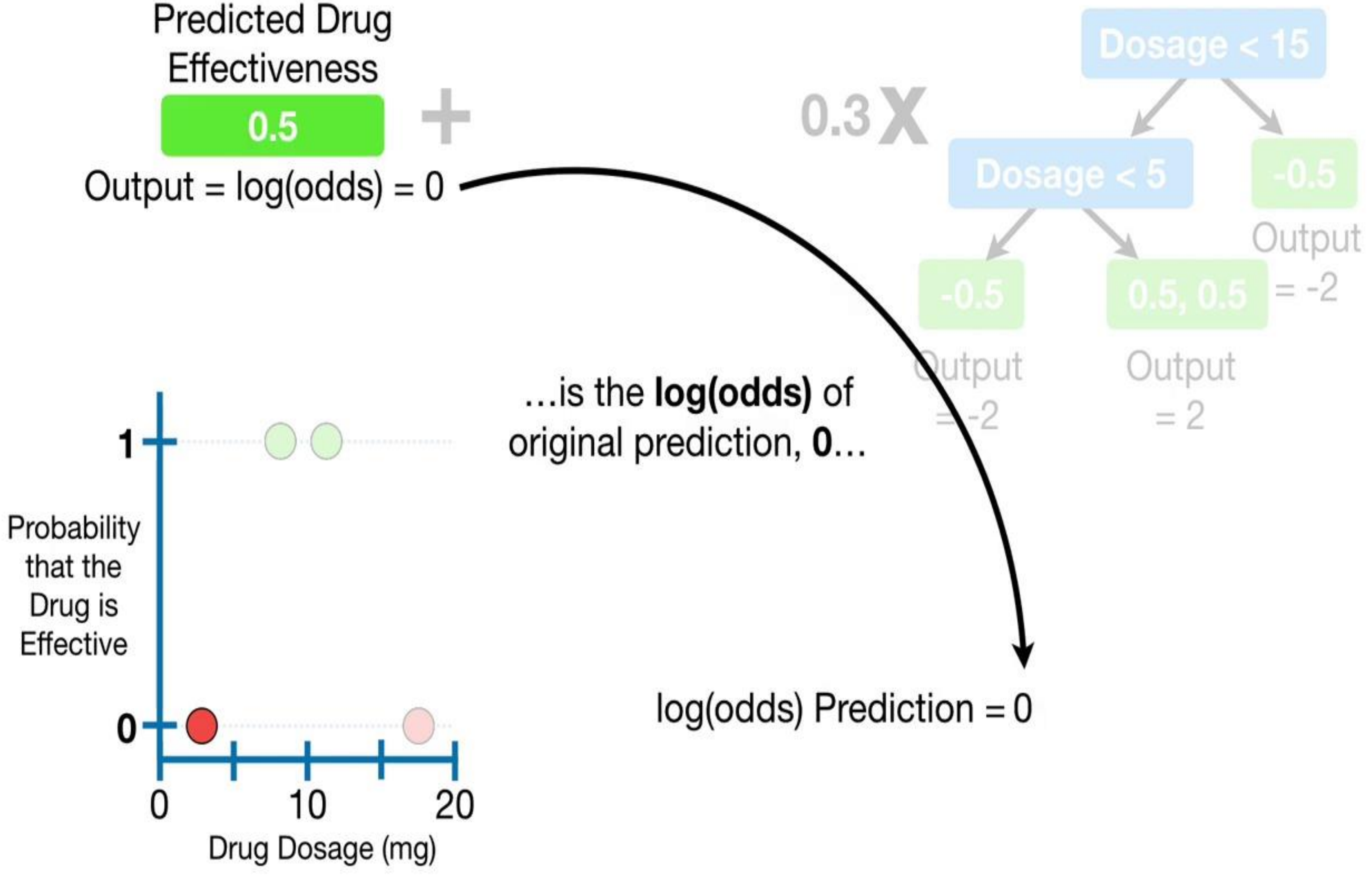
XGBoost for Classification



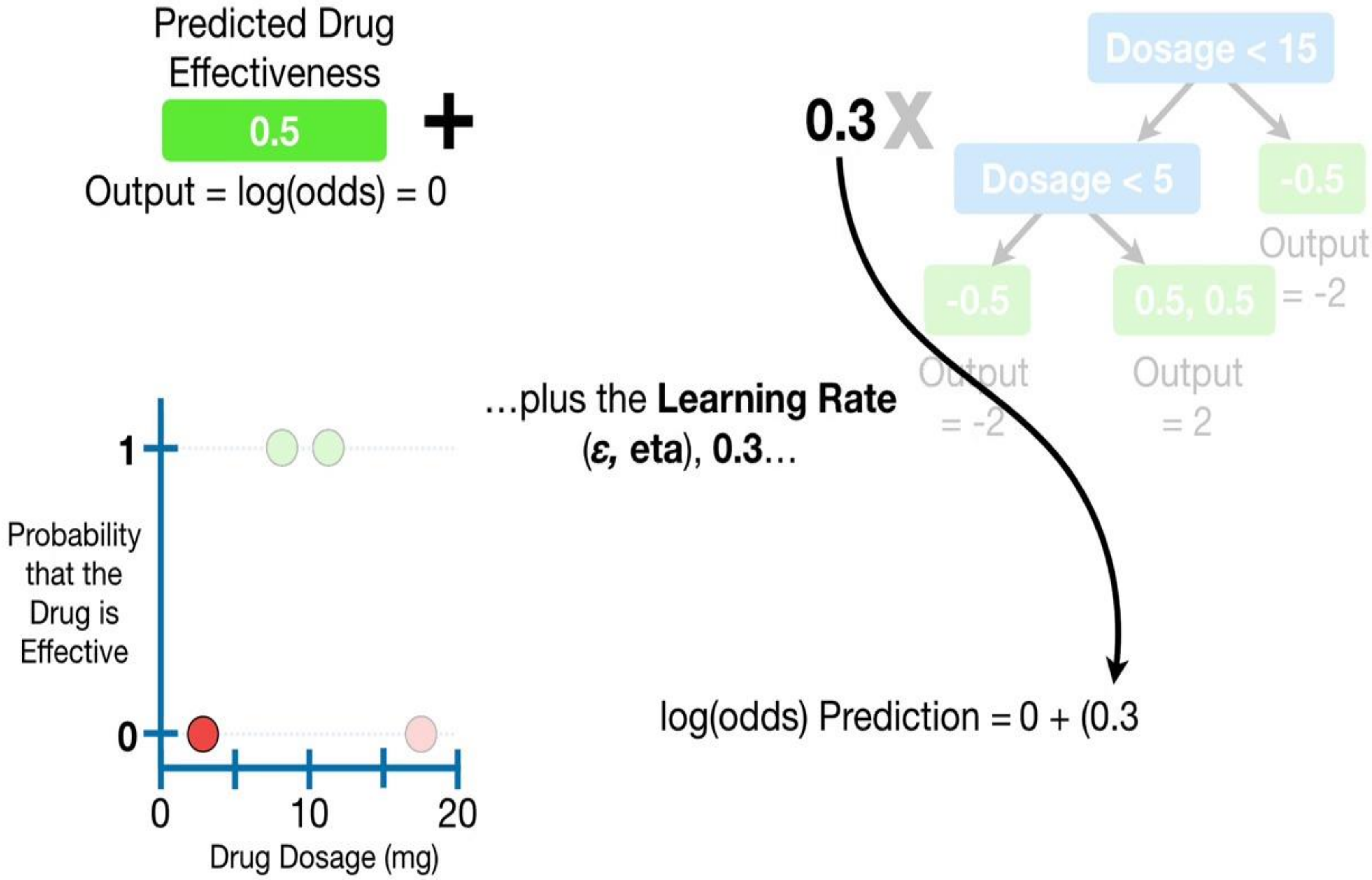
XGBoost for Classification



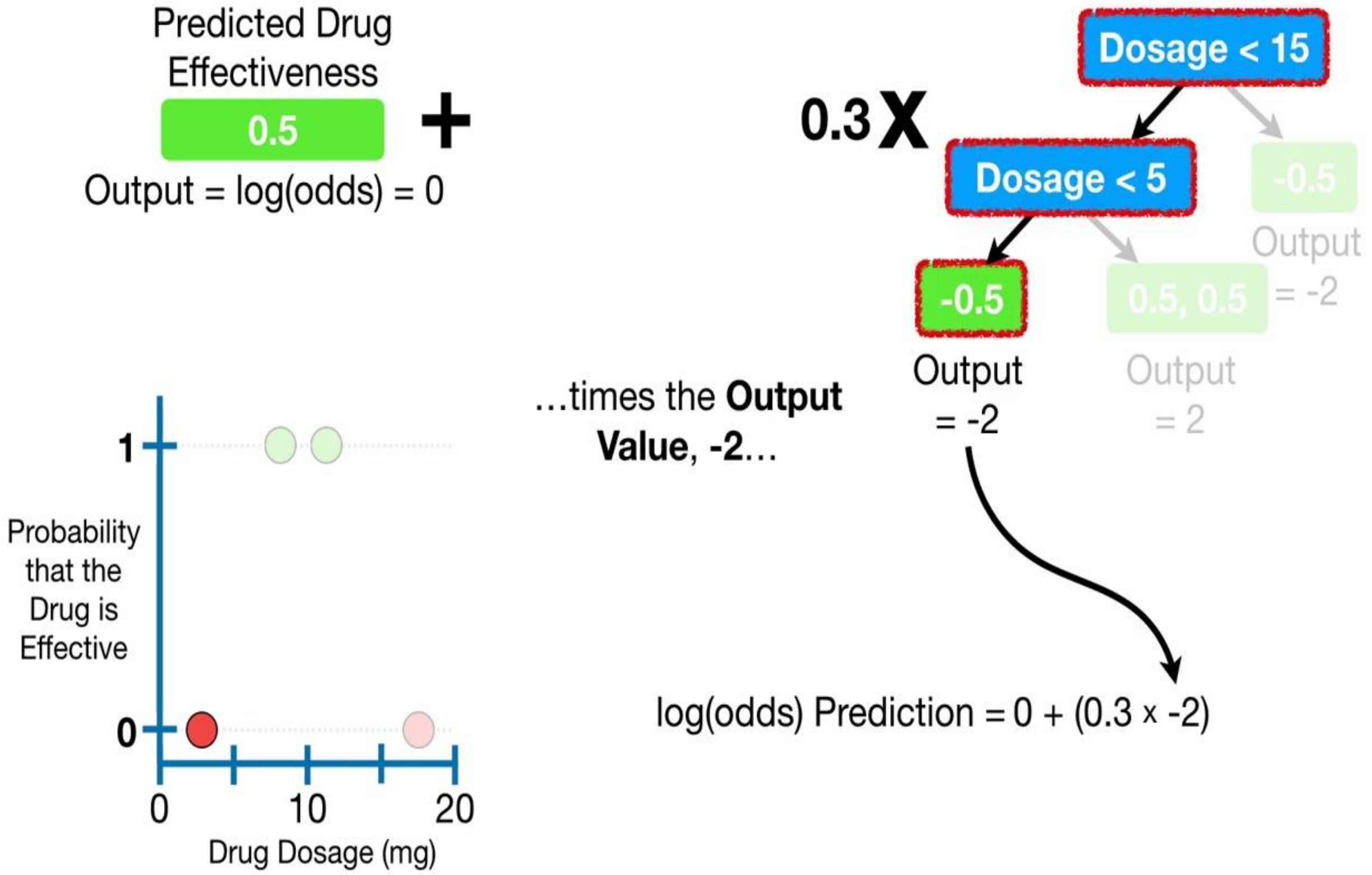
XGBoost for Classification



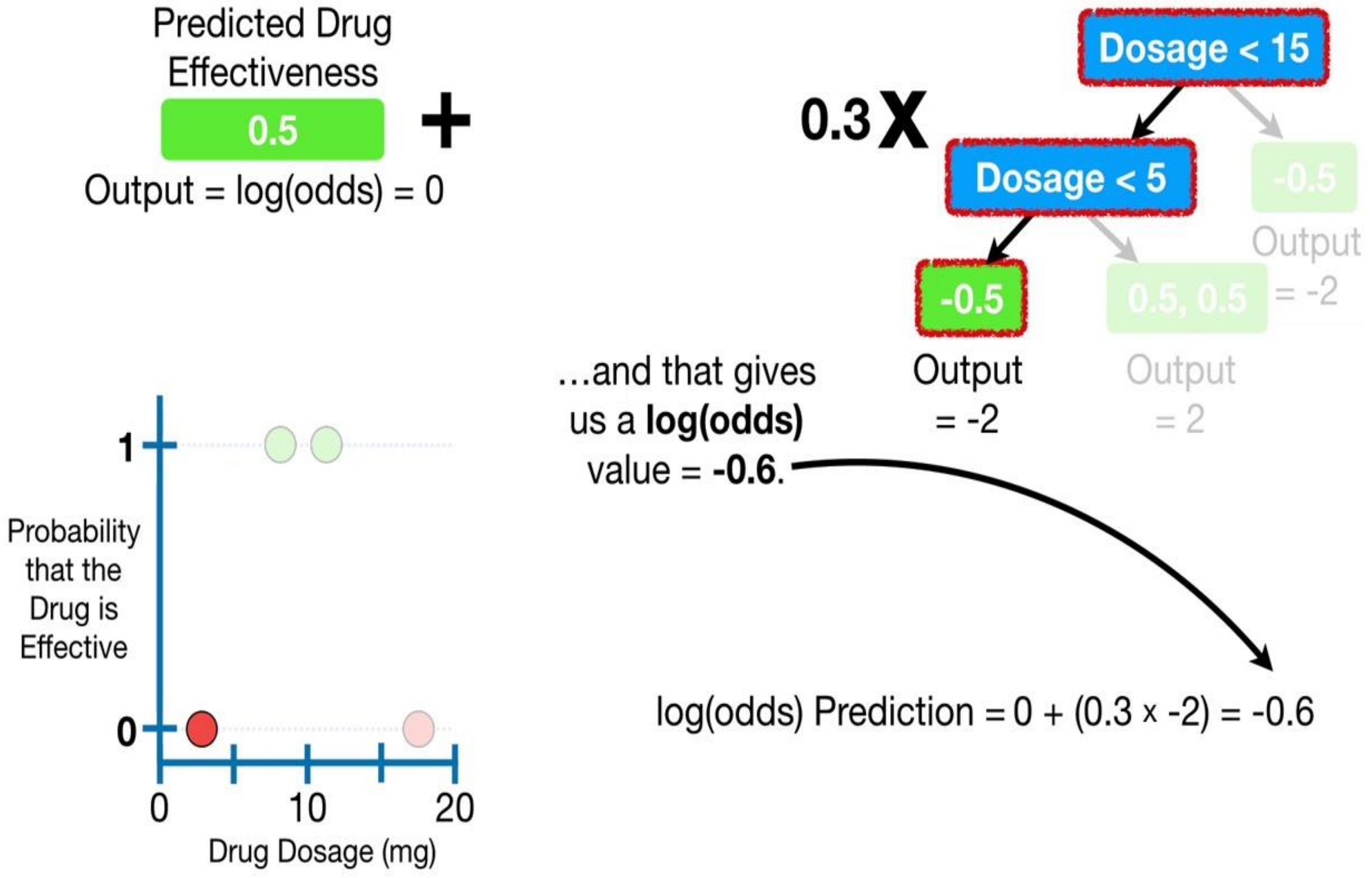
XGBoost for Classification



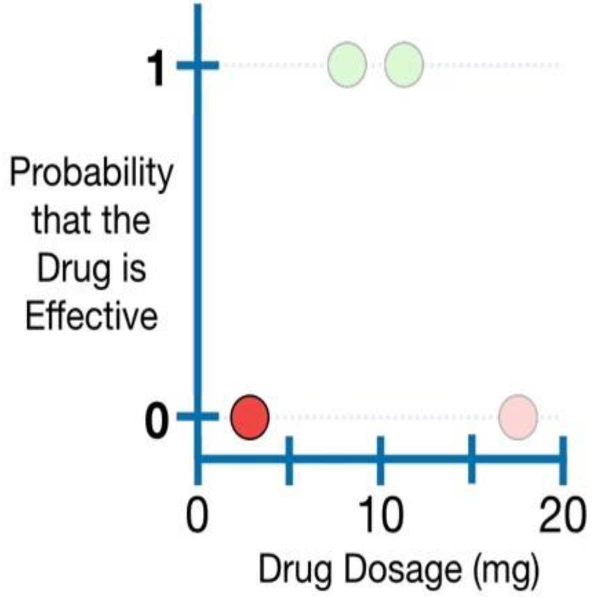
XGBoost for Classification



XGBoost for Classification



XGBoost for Classification

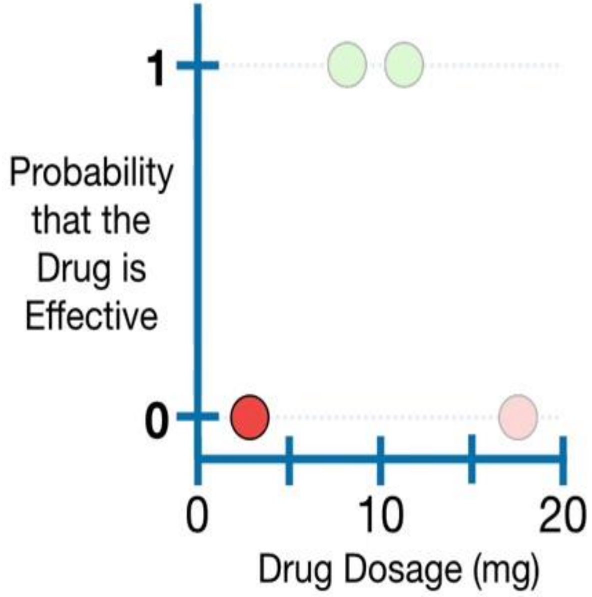


To convert a **log(odds)** value into a probability, we plug it into the **Logistic Function.**

$$\text{Probability} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

$$\log(\text{odds})\ \text{Prediction} = 0 + (0.3 \times -2) = -0.6$$

XGBoost for Classification

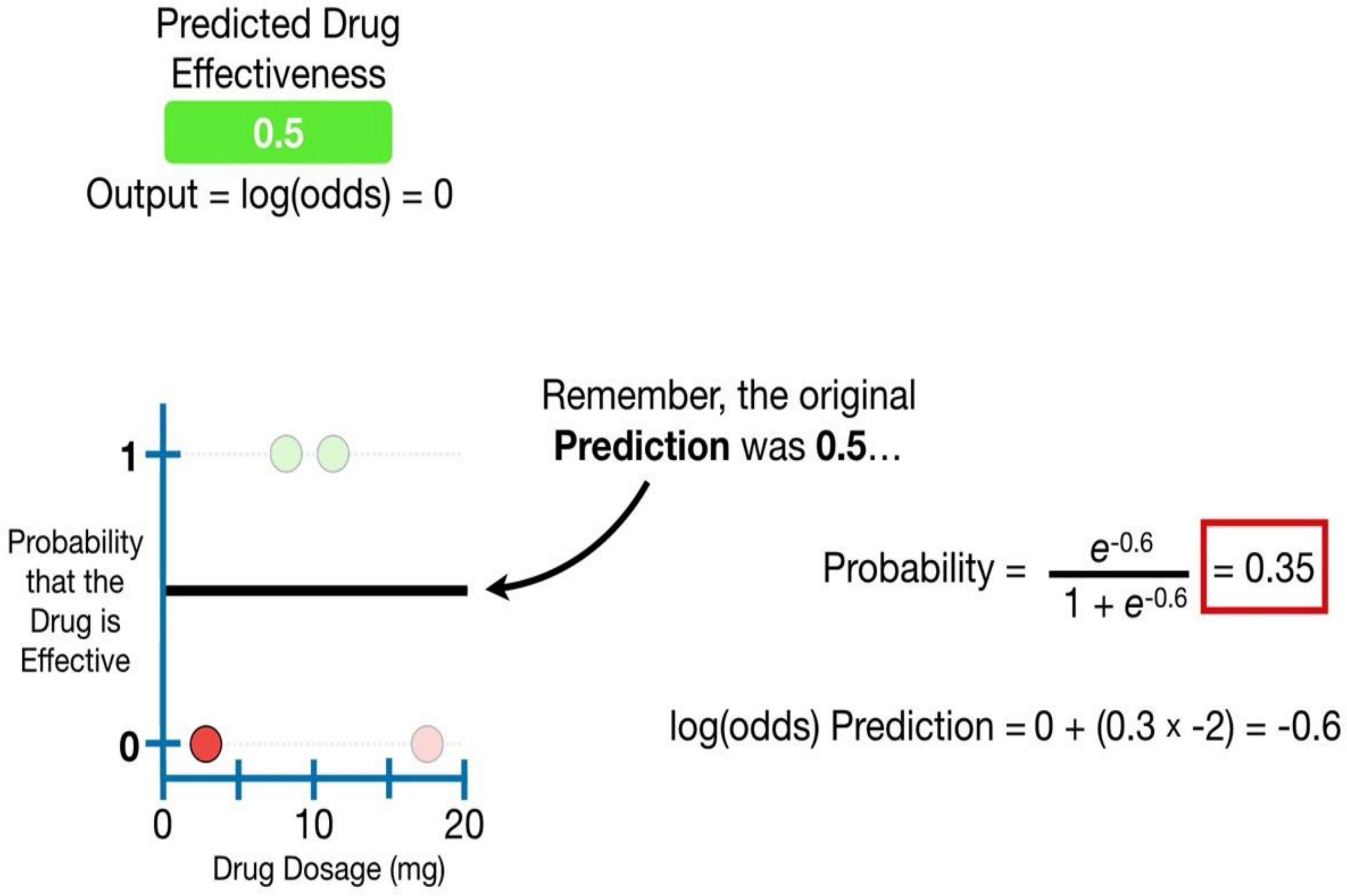


...and the the new predicted probability is **0.35**.

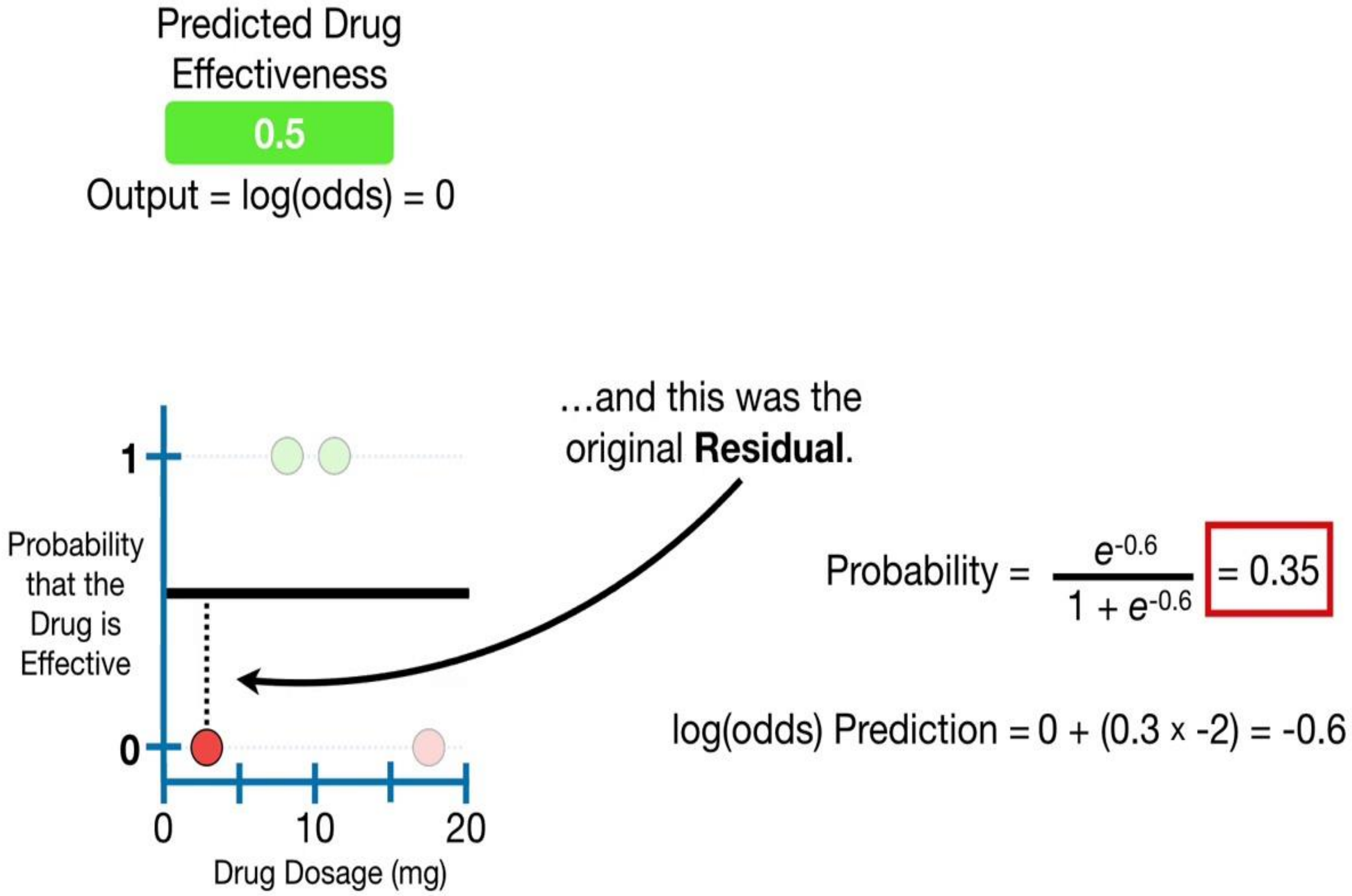
$$\text{Probability} = \frac{e^{-0.6}}{1 + e^{-0.6}} = 0.35$$

$$\text{log(odds) Prediction} = 0 + (0.3 \times -2) = -0.6$$

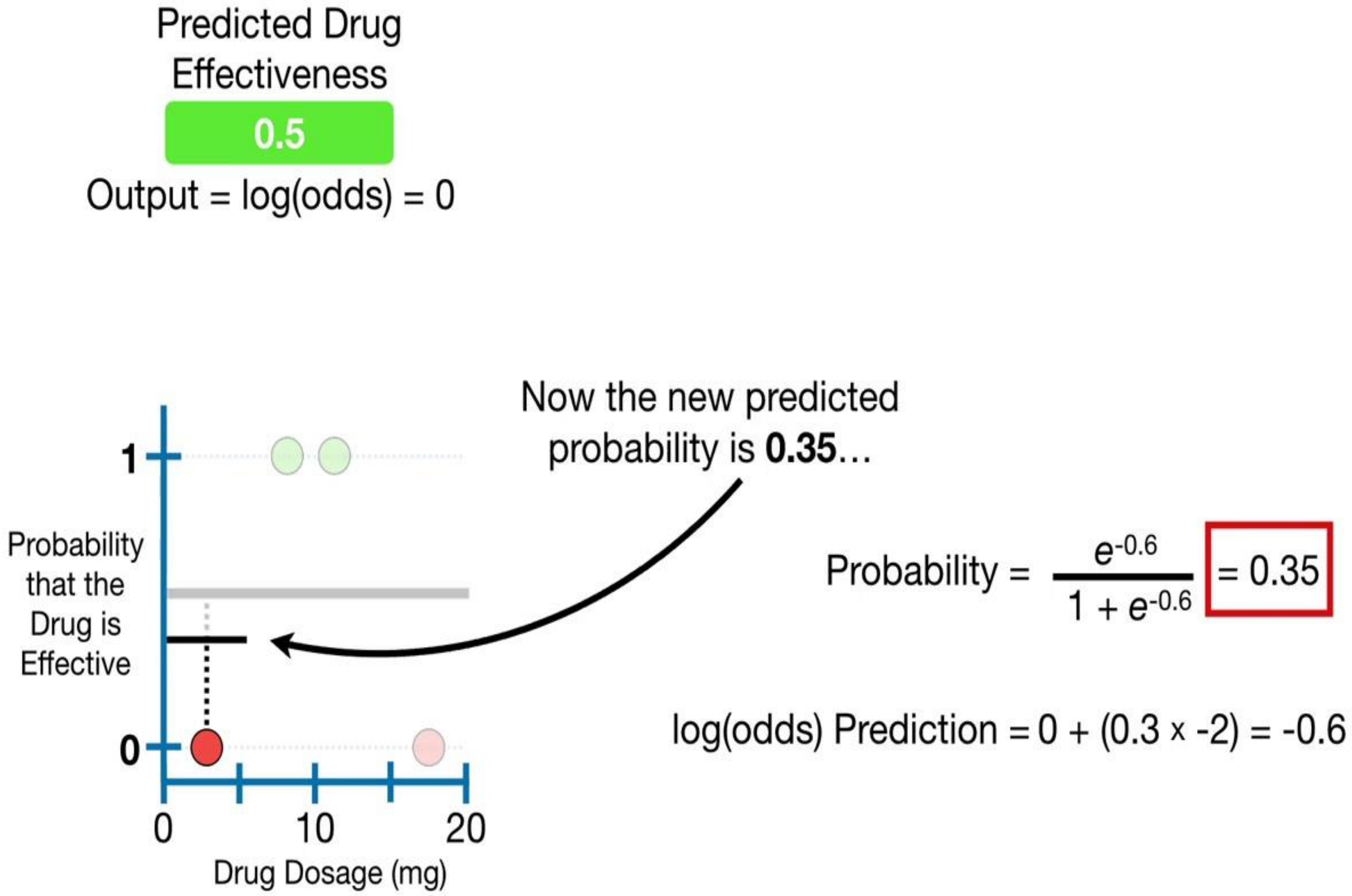
XGBoost for Classification



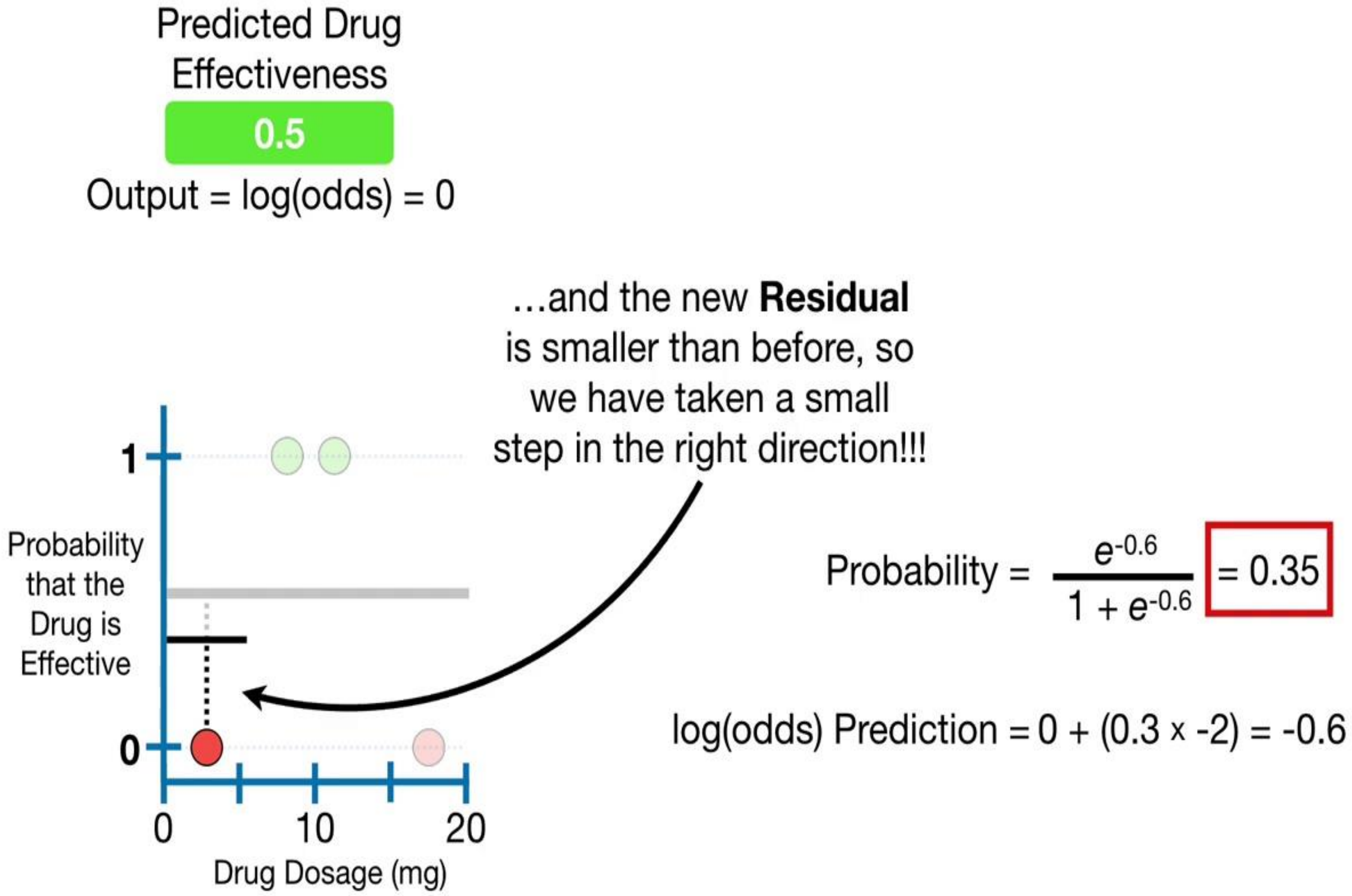
XGBoost for Classification



XGBoost for Classification



XGBoost for Classification

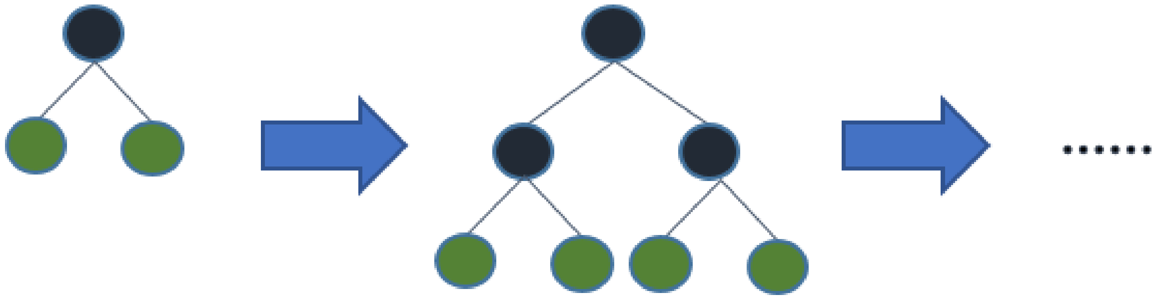


LightGBM

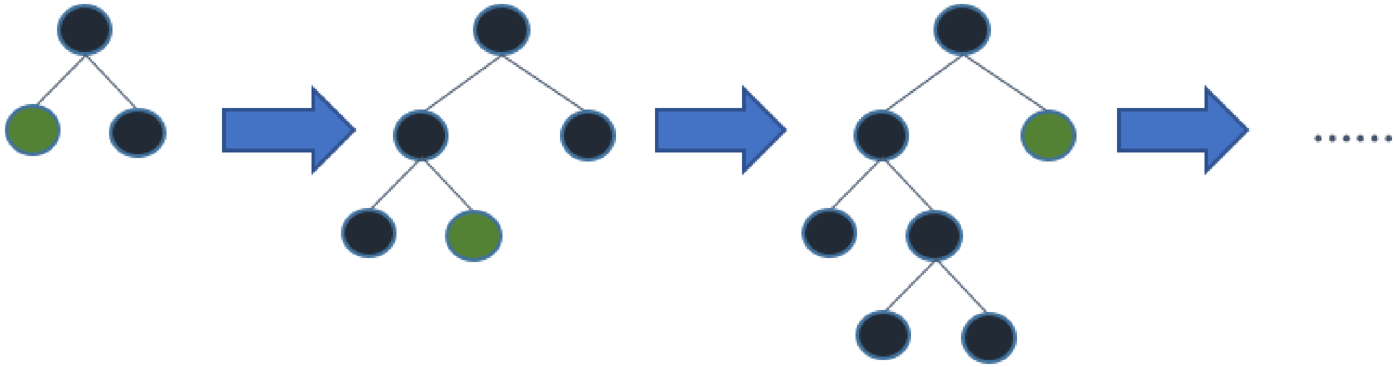
- LightGBM (by Microsoft) is a distributed high-performance framework that uses decision trees for classification, and regression tasks.
- Advantage w.r.t. XGBoost
  - Faster training speed and accuracy resulting from LightGBM being a histogram-based algorithm that performs bucketing of values (also requires lesser memory)
  - Compatible with large and complex datasets but is much faster during training
  - Support for both parallel learning and GPU learning

LightGBM vs XGBoost

- XGBoost: level-wise (horizontal) growth



- LightGBM: out leaf-wise (vertical) growth



- LightGBM is significantly faster than XGBoost but delivers almost equivalent performance

Gradient-Based One-Side Sampling (GOSS)

- In Gradient Boosted Decision Trees, the data instances have no native weight which is leveraged by GOSS.
- Data instances with larger gradients contribute more towards information gain.
- To maintain the accuracy of the information, GOSS retains instances with larger gradients and performs random sampling on instances with smaller gradients.

Exclusive Feature Bundling (EFB)

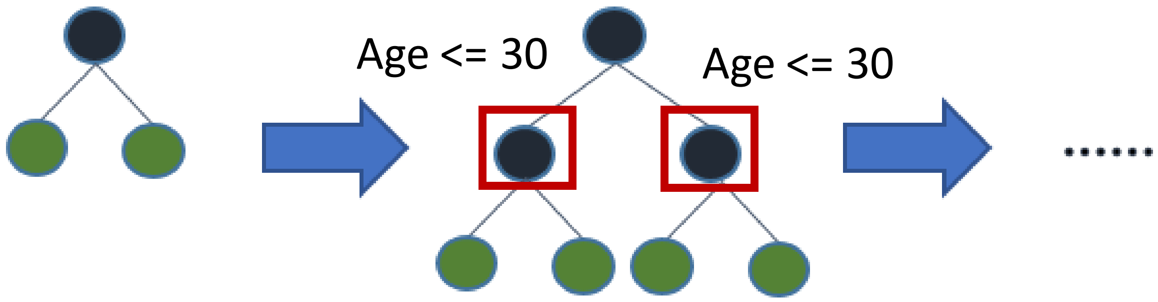
- Exclusive Feature Bundling is a near lossless method to reduce the number of effective features.
- Just like One-Hot encoded features, in the sparse space, many features rarely take non-zero values simultaneously.

LightGBM and XGBoost

- Handling Categorical Features
- Handling Missing Values

CatBoost

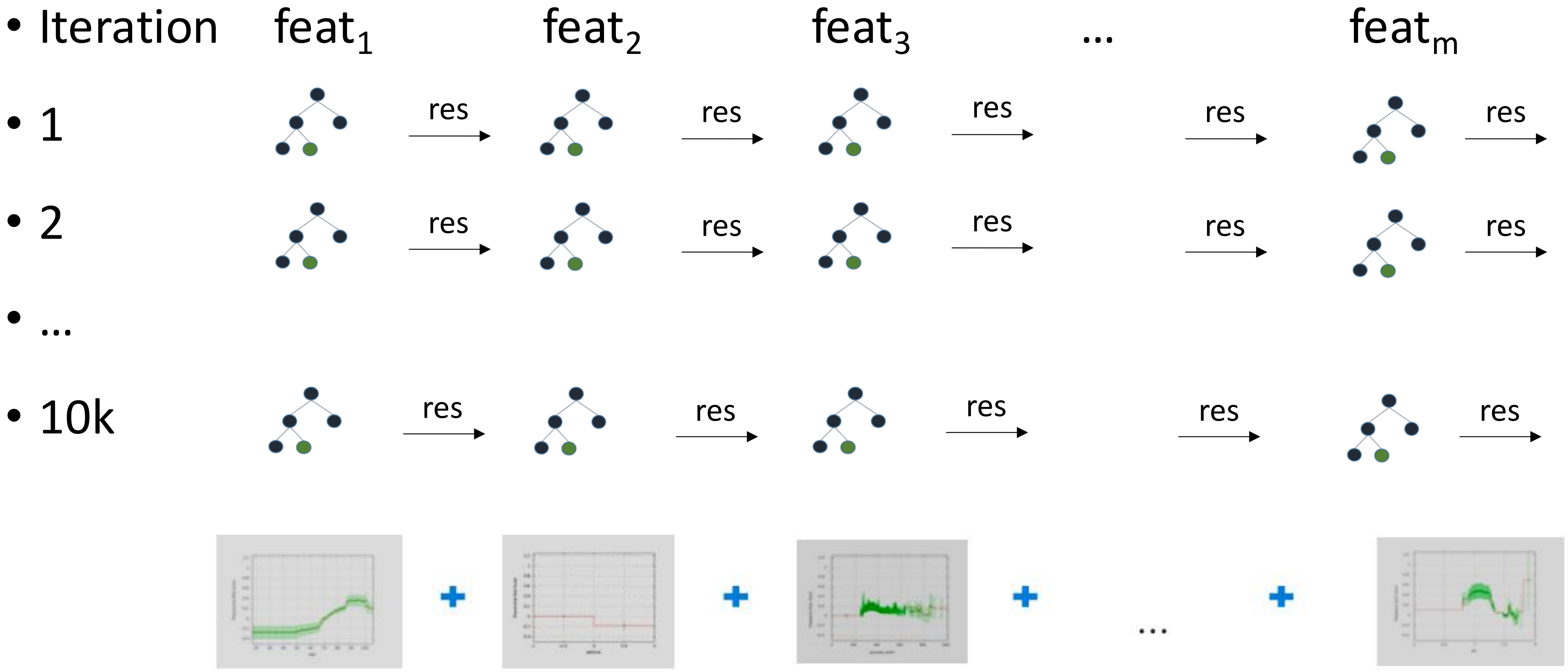
- Categorial Boosting
- Gradient Boosting Machine
- Adopts Ordered Target Encoding procedure for categorical attributes
- Rely on Symmetric Decision Trees, i.e., dedicion trees with the same splitting conditions for nodes at the same lavel



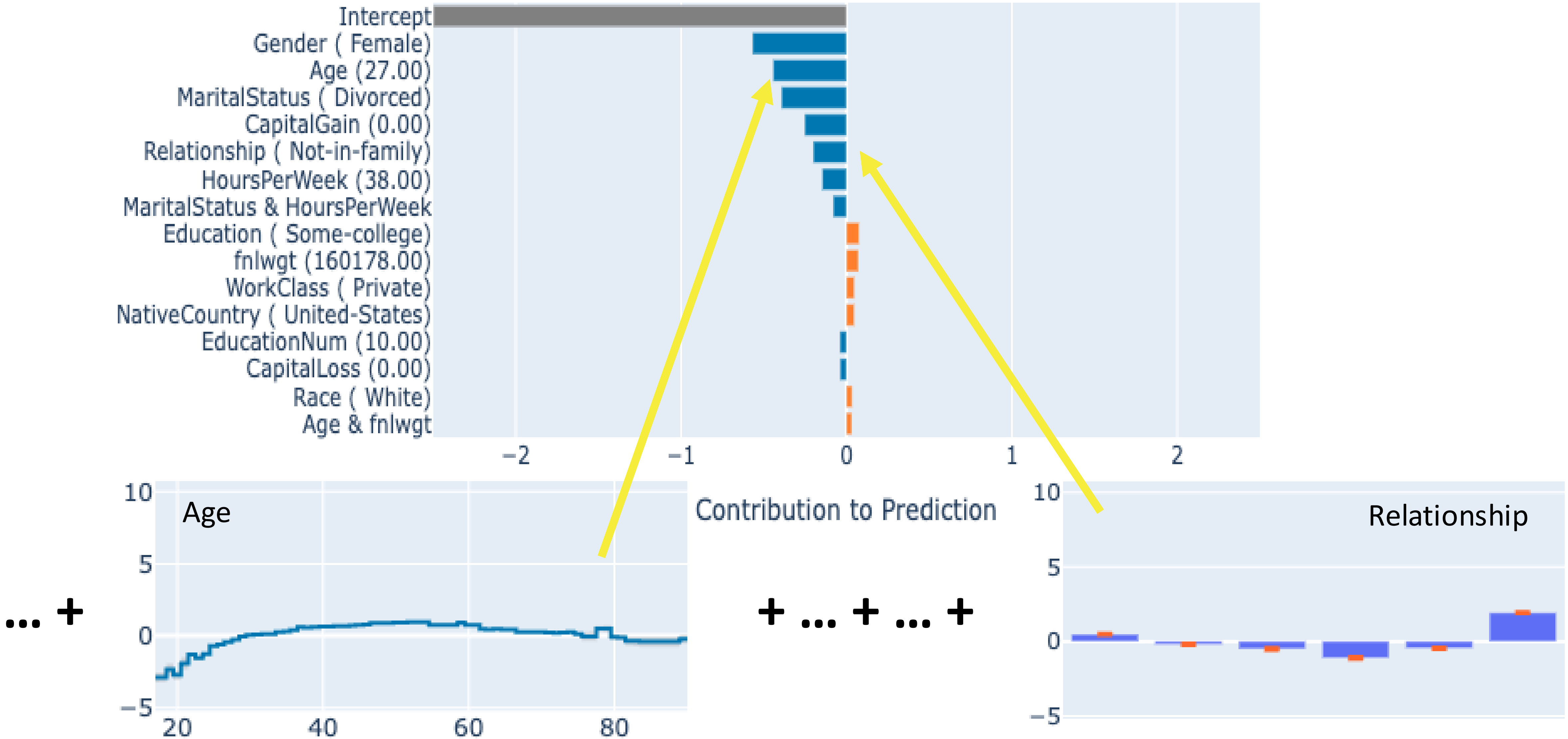
Explainable Boosting Machines (EMB)

- Type of Generalized Additive Models (GAMS)
- Additive Model: $y = f_1(x_1) + f_2(x_2) + \ldots + f_m(x_m)$
- Linear Model: $y = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m$
- Complex Model: $y = f(x_1, x_2, \ldots, x_m)$

EBM Algorithm Sketch



EBM Prediction Time



References

- Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). Boosting algorithms as gradient descent. *Advances in neural information processing systems*, *12*.
- Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, *38*(4), 367-378.
- Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*.