Introduction

Gradient Boost-(ish)

Regularization

A Unique Regression Tree

Approximate Greedy Algorithm

Weighted Quantile Sketch

Sparsity-Aware Split Finding

Parallel Learning

Cache-Aware Access

Blocks for Out-of-Core Computation

XGBoost is **EXTREME!!!!!** And that means it's a big **Machine Learning** algorithm with lots of parts.

The good news is that each part is pretty simple and easy to understand, and we'll go through them one step at a time.

Gradient Boost-(ish)

Regularization

A Unique Regression Tree

Approximate Greedy Algorithm

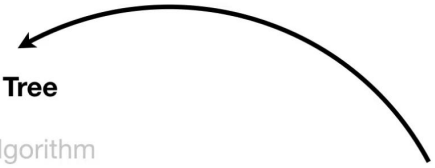
Weighted Quantile Sketch

Sparsity-Aware Split Finding

Parallel Learning

Cache-Aware Access

Blocks for Out-of-Core Computation



…so we'll start by learning about **XGBoost's** unique **Regression Trees**.

NOTE: XGBoost was designed to be used with large, complicated data sets.

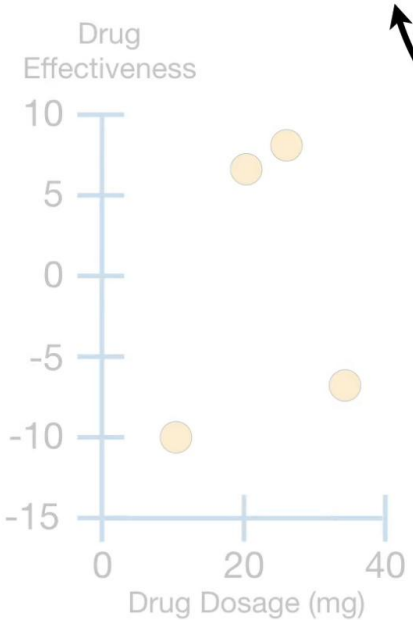
Mass	Age	BP	Color	Movie	Car	Hair	etc.
120	23	102	Brown	T2	Ford	Long	…
150	25	98	Brown	Frozen	Kia	Short	…
165	22	130	Black	Spiderman	Ford	Short	…
123	45	98	Red	T2	Kia	Long	…
156	33`	78	Brown	Frozen 2	Ford	Long	…
…	…	…	…	…	…	…	…

Initial Prediction

Predicted Drug Effectiveness

0.5

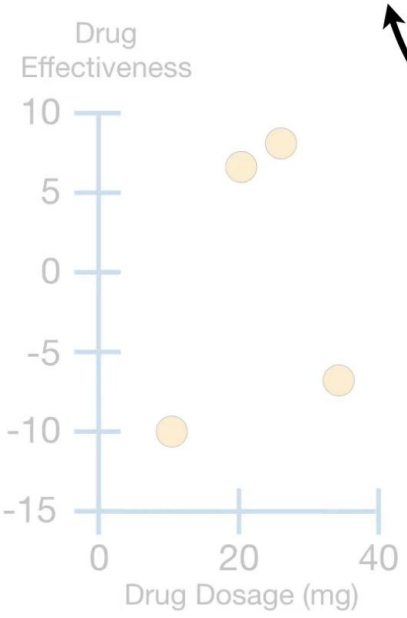
The very first step in fitting **XGBoost** to the **Training Data** is to make an initial prediction.



Predicted Drug Effectiveness

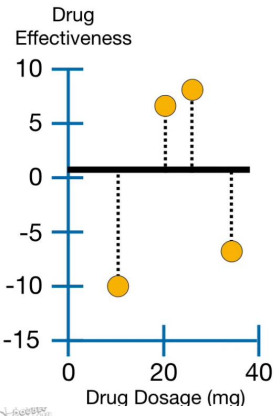
0.5

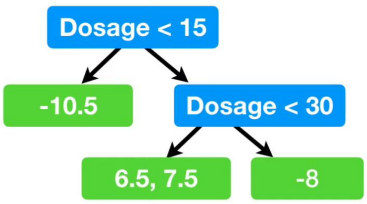
This prediction can be anything, but by default it is **0.5**, regardless of whether you are using **XGBoost** for **Regression** or **Classification**.



Predicted Drug Effectiveness

0.5

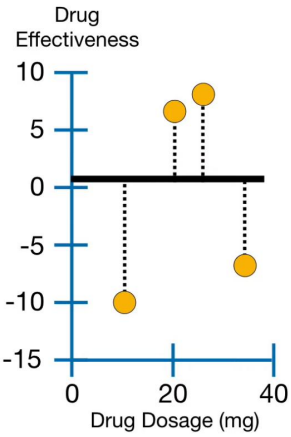


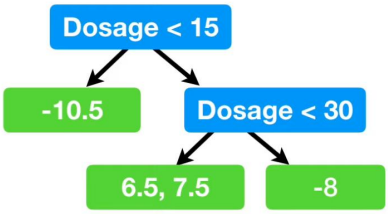


Now, just like unextreme **Gradient Boost**, **XGBoost** fits a **Regression Tree** to the residuals…

Predicted Drug Effectiveness

0.5

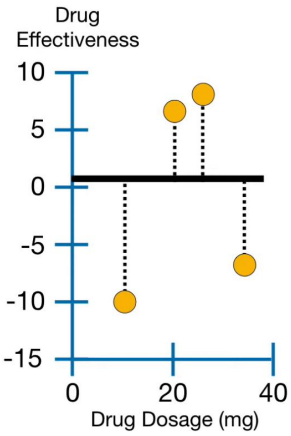


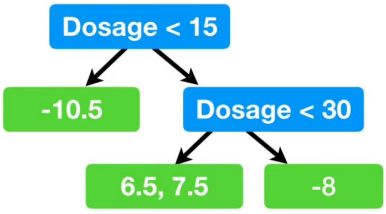


However, unlike unextreme **Gradient Boost**, which typically uses regular, off-the-shelf, **Regression Trees**…

Predicted Drug Effectiveness

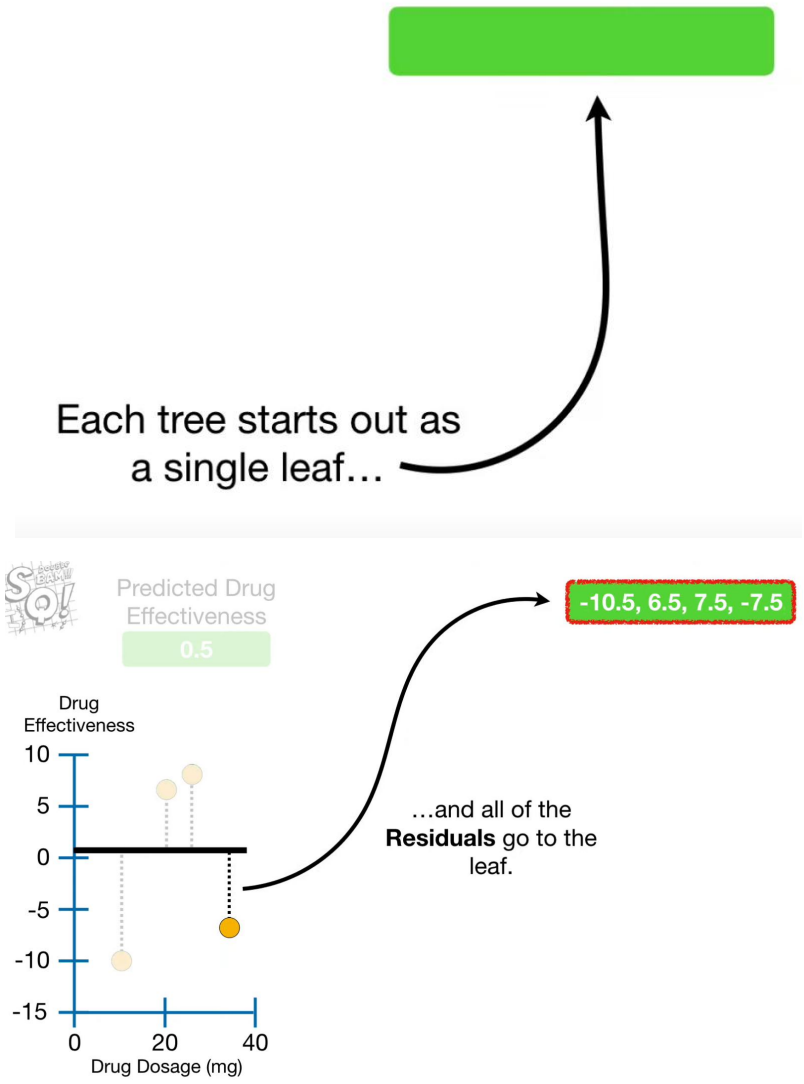
0.5





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

NOTE: There are many ways to build **XGBoost Trees**. This video focuses on the most common way to build them for **Regression**.



Similarity Score and Gain

-10.5, 6.5, 7.5, -7.5

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

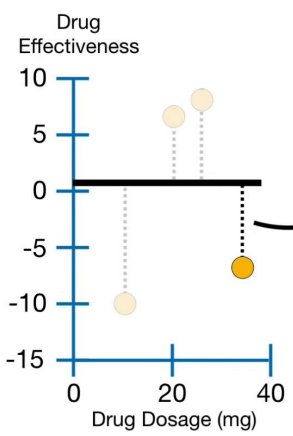
-10.5, 6.5, 7.5, -7.5

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

NOTE: λ (lambda) is a **Regularization** parameter, and we'll talk more about that later.

$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + 0}$$

For now, let $\lambda = 0$.



Similarity Score = $\frac{(-10.5 + 6.5 + 7.5 + -7.5)^2}{\text{Number of Residuals} + 0}$

Now we plug the **4 Residuals** into the numerator…

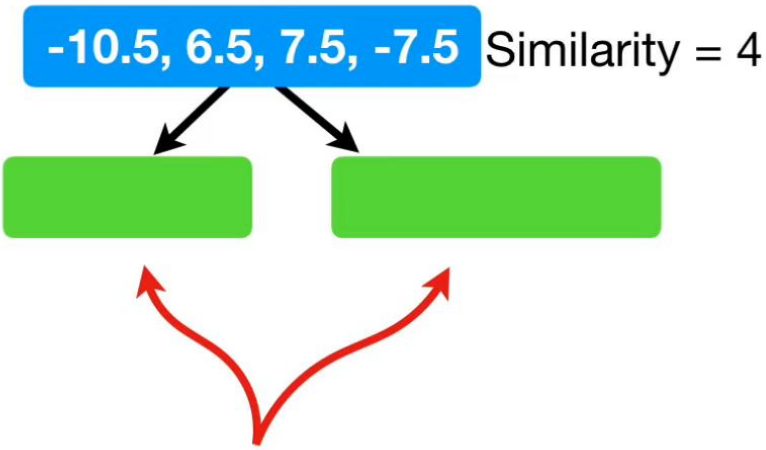
$$\text{Similarity Score} = \frac{(-10.5 + 6.5 + 7.5 + -7.5)^2}{4 + 0}$$

…and since there are **4** **Residuals** in the leaf, we put a **4** in the denominator.

-10.5, 6.5, 7.5, -7.5 Similarity = 4

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

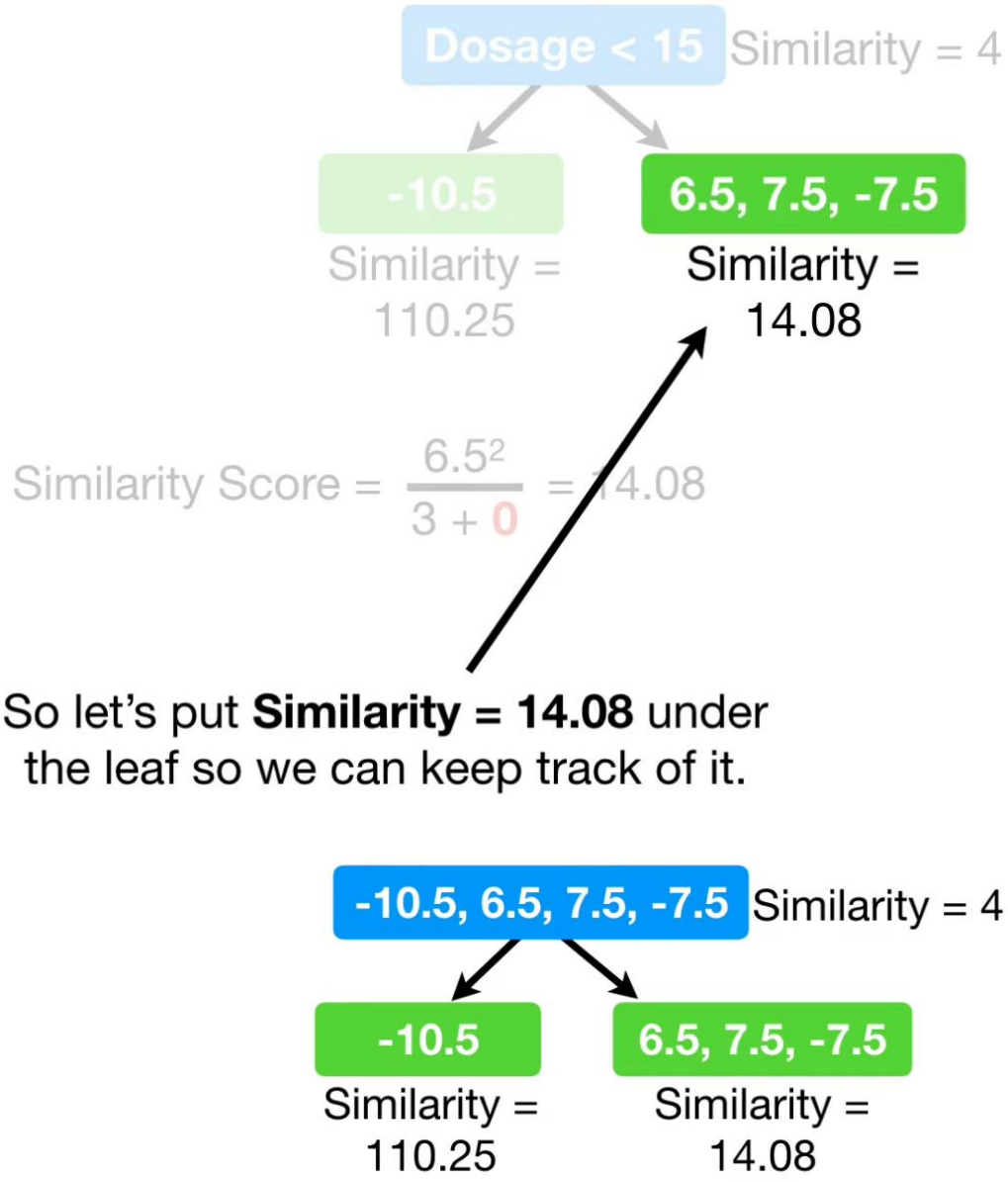
So let's put **Similarity = 4** up here to can keep track of it.



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

$$\text{Similarity Score} = \frac{-10.5^2}{1 + 0} = 110.25$$

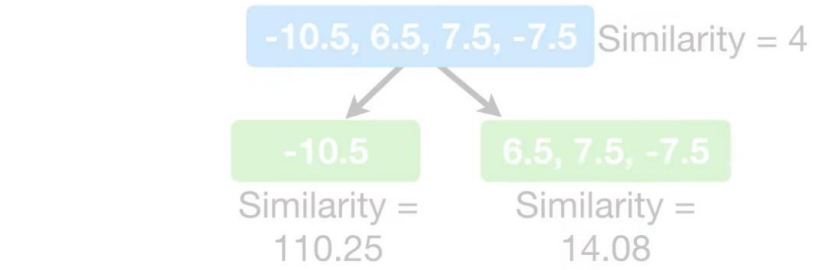
…and the **Similarity Score** for the leaf on the left = **110.25**.



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

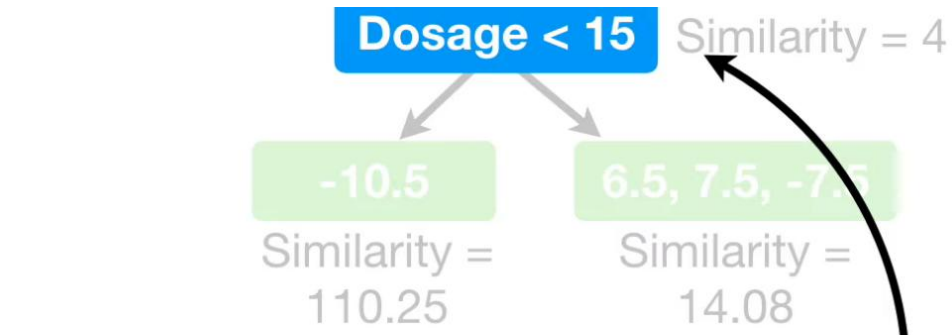
Gain = $\text{Left}_{\text{Similarity}}$ + $\text{Right}_{\text{Similarity}}$ - $\text{Root}_{\text{Similarity}}$

We do this by calculating the **Gain** of splitting the **Residuals** into two groups.



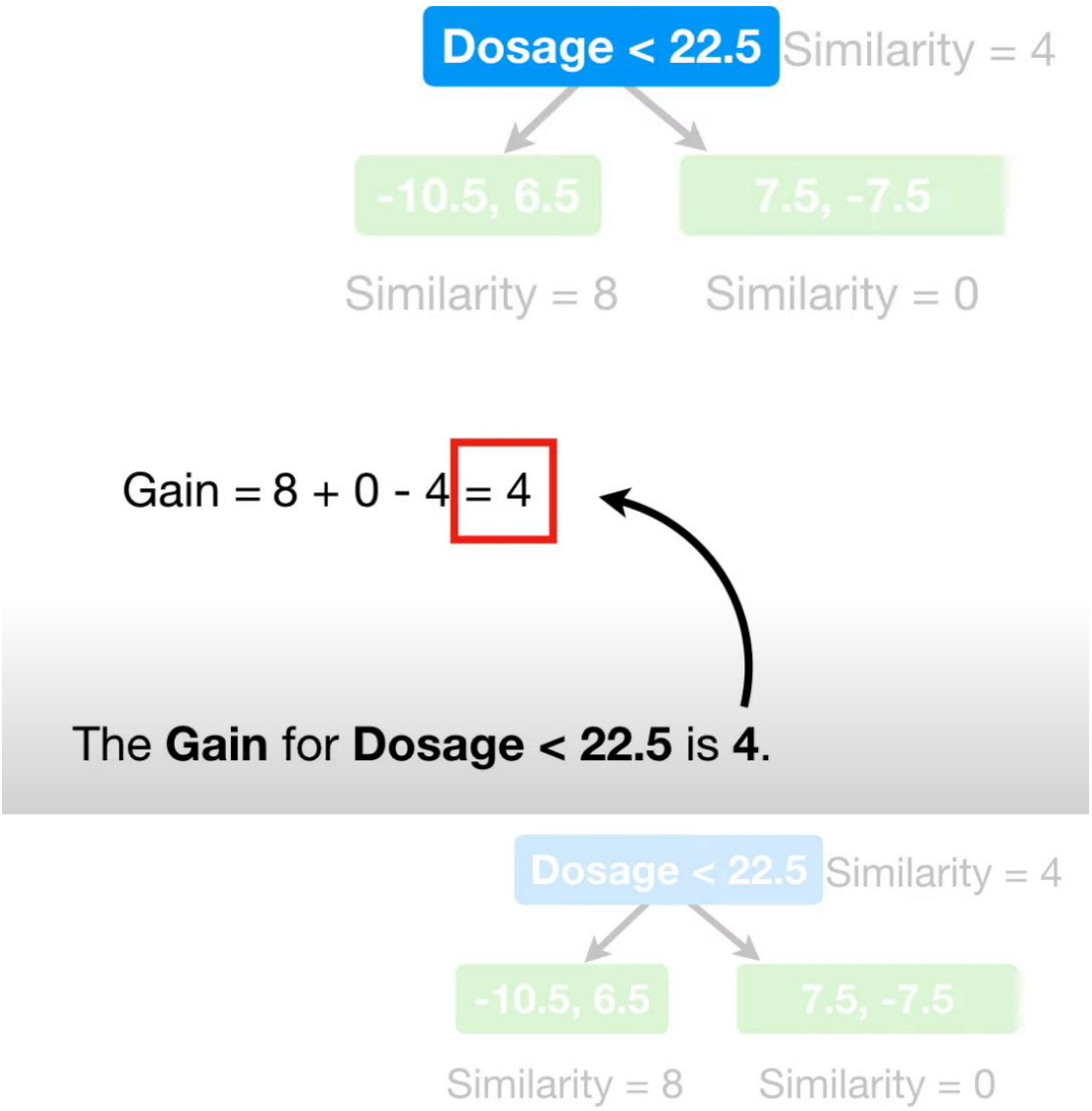
Gain = 110.25 + 14.08 - 4 = 120.33

…gives us **120.33**.



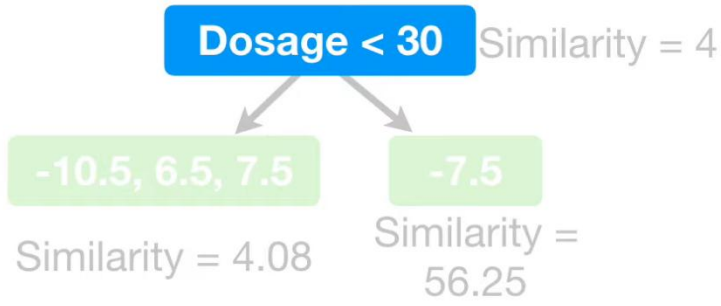
Gain = 110.25 + 14.08 - 4 = 120.33

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



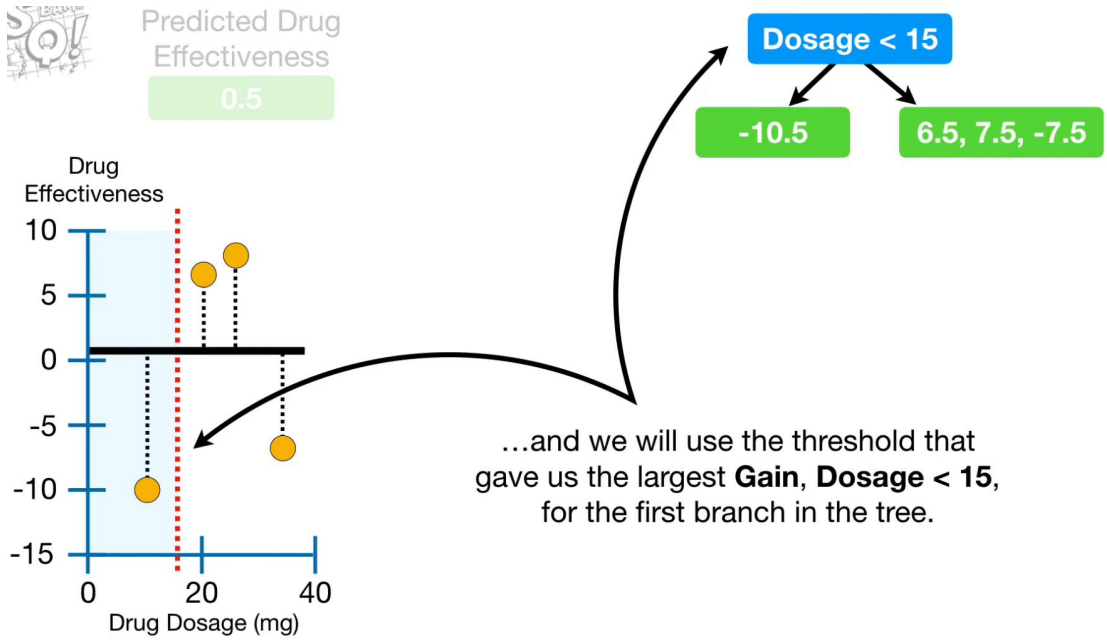
Gain = 8 + 0 - 4 = 4

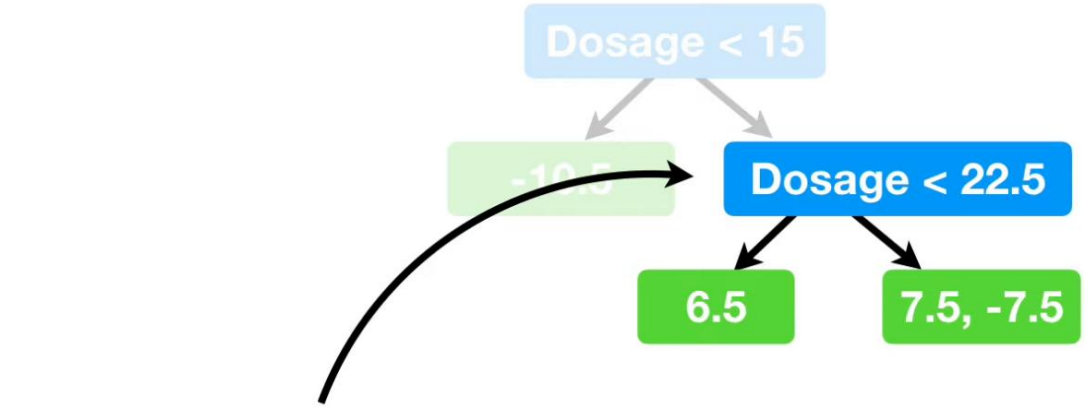
Since the **Gain** for **Dosage < 22.5** (**Gain = 4**) is less than the **Gain** for **Dosage < 15** (**Gain = 120.33**), **Dosage < 15** is better at splitting the **Residuals** into clusters of similar values.



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.

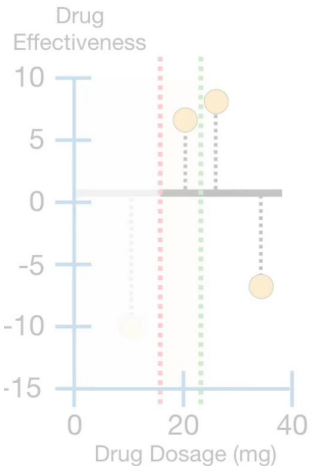


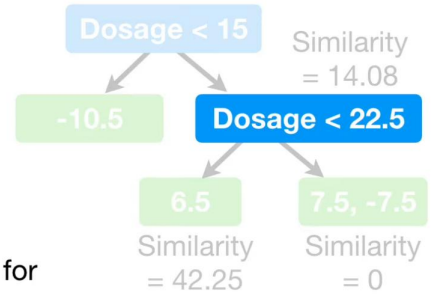


So the first threshold that we try is **Dosage < 22.5**.

Predicted Drug Effectiveness

0.5



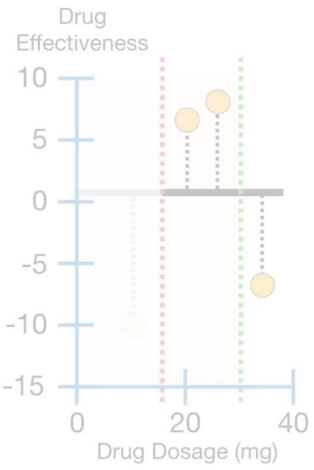


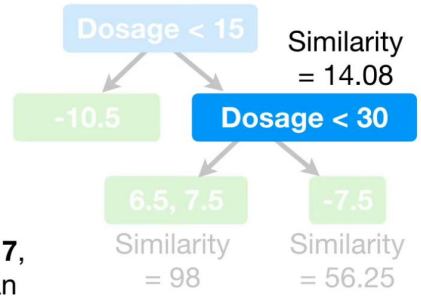
And we get **Gain = 28.17** for when the threshold is **Dosage < 22.5**.

Gain = 42.25 + 0 - 14.08 = 28.17

Predicted Drug Effectiveness

0.5



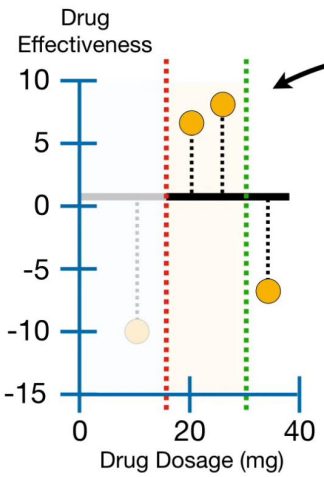


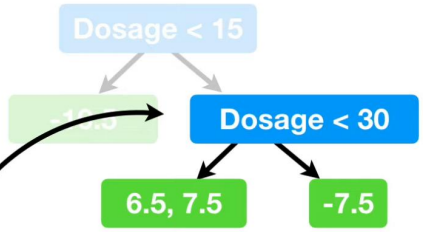
And we get **Gain = 140.17**, which is much larger than **28.17**, when the threshold was **Dosage < 22.5**.

Gain = 98 + 56.25 - 14.08 = 140.17

Predicted Drug Effectiveness

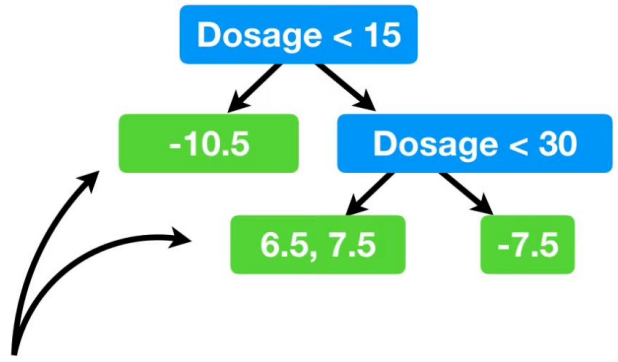
0.5





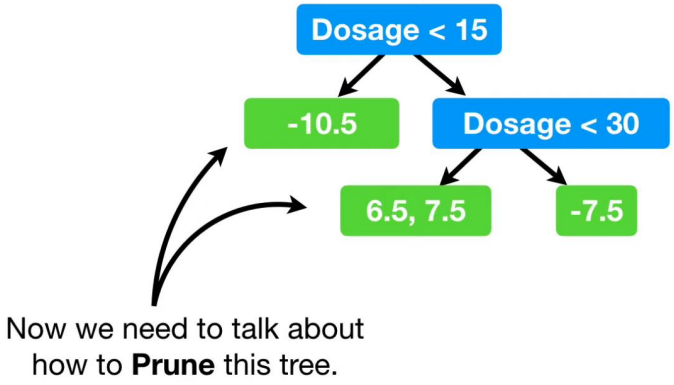
So we will use **Dosage < 30** as the threshold for this branch.

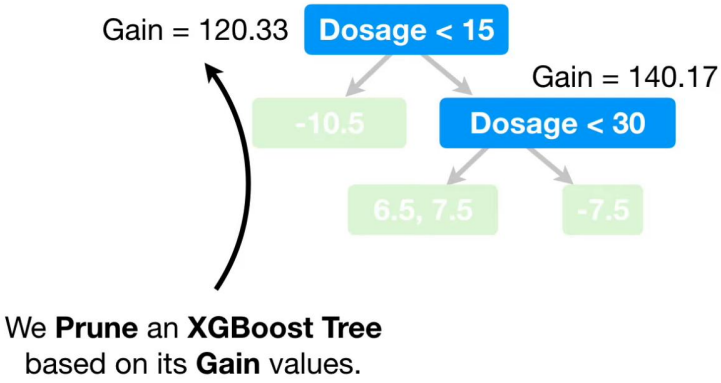
Gain = 98 + 56.25 - 14.08 = 140.17

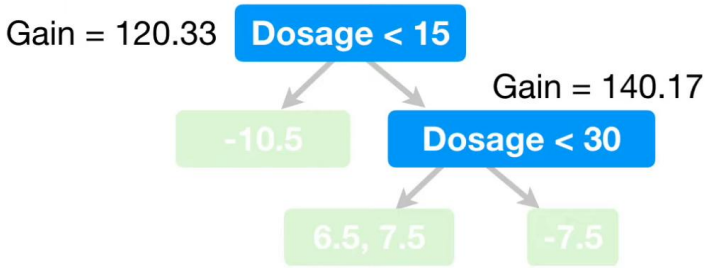


NOTE: To keep this example from getting out of hand, I've limited the tree depth to two levels…

Tree Pruning

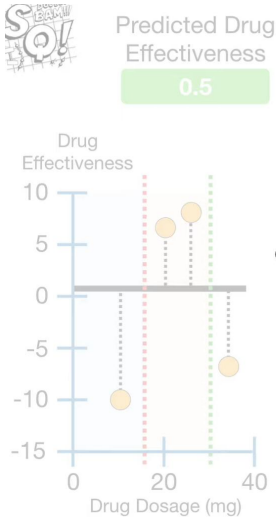


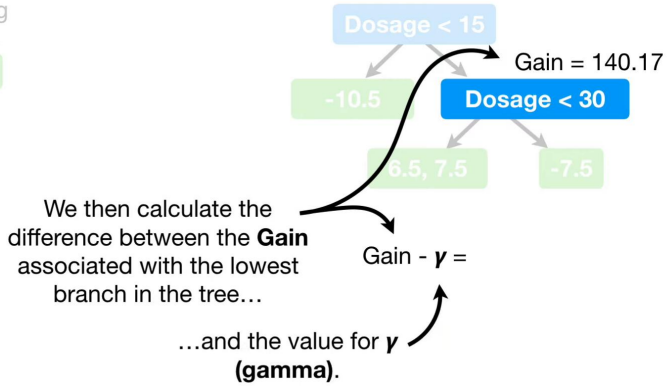


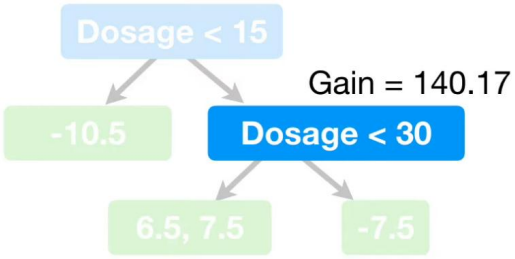


We start by picking a number, for example, **130**.

TERMINOLOGY ALERT!!!
XGBoost calls this number ***γ* (gamma)**.

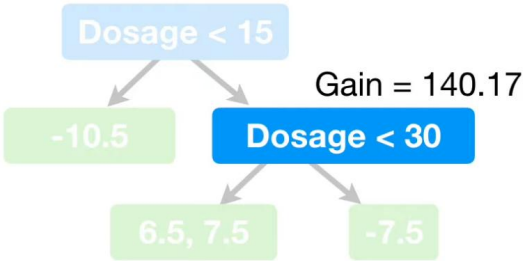






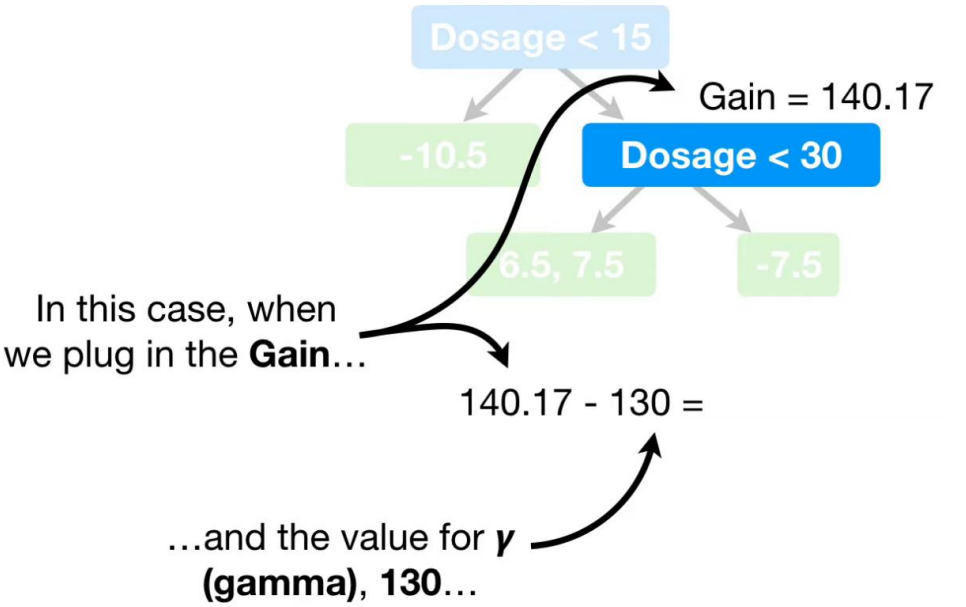
Gain - **γ** =

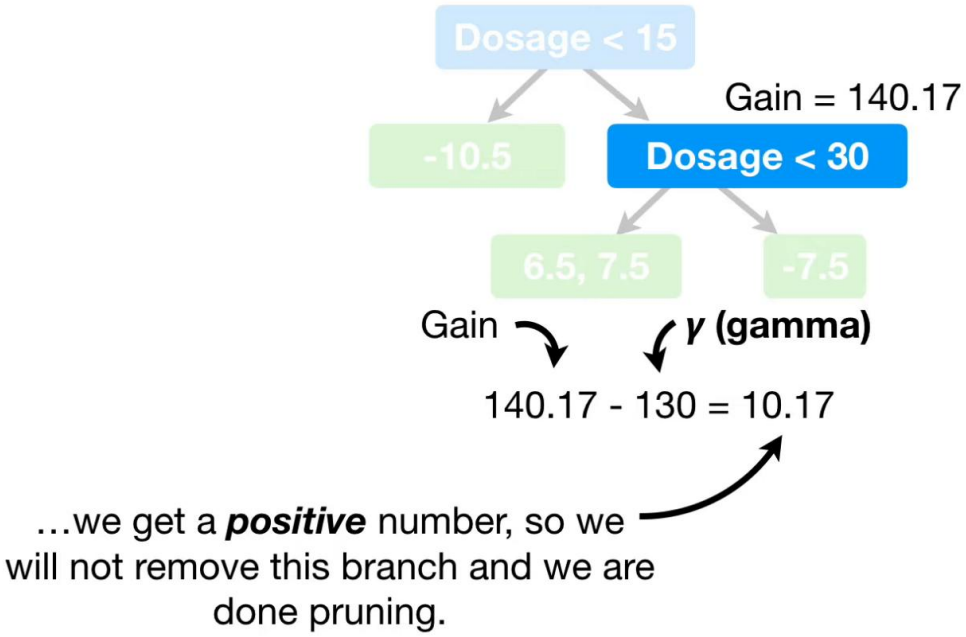
If the difference between the **Gain** and **γ (gamma)** is ***negative*** we will remove the branch…

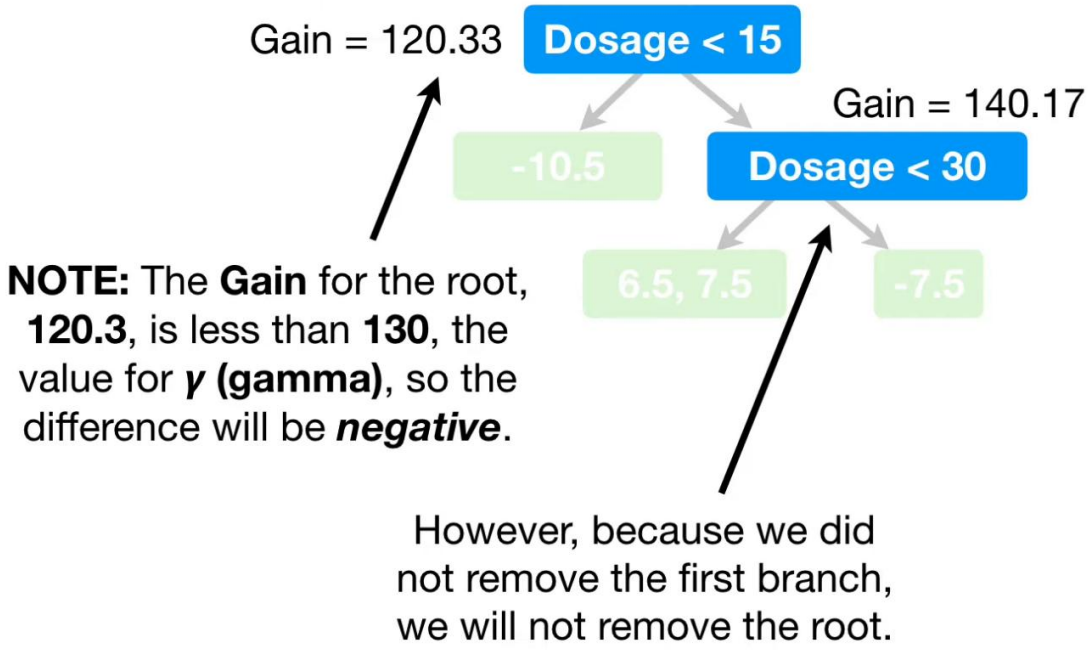


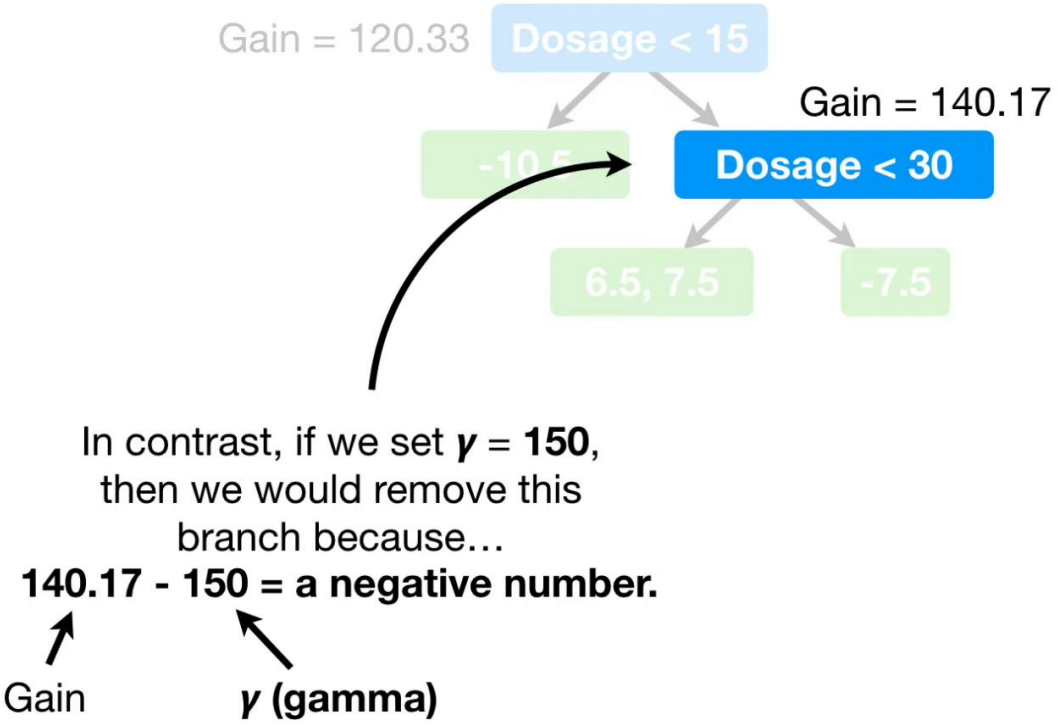
Gain - **γ** =

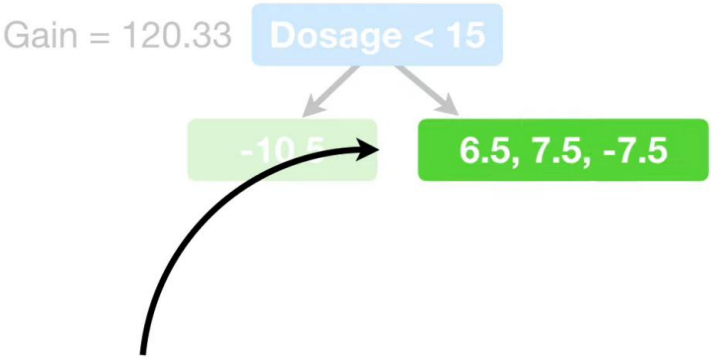
…and if the difference between the **Gain** and **γ (gamma)** is ***positive*** we will not remove the branch.


Dosage < 15
Gain = 140.17
-10.5
Dosage < 30
6.5, 7.5
-7.5
In this case, when we plug in the **Gain**…
140.17 - 130 =
…and the value for **γ (gamma)**, **130**…

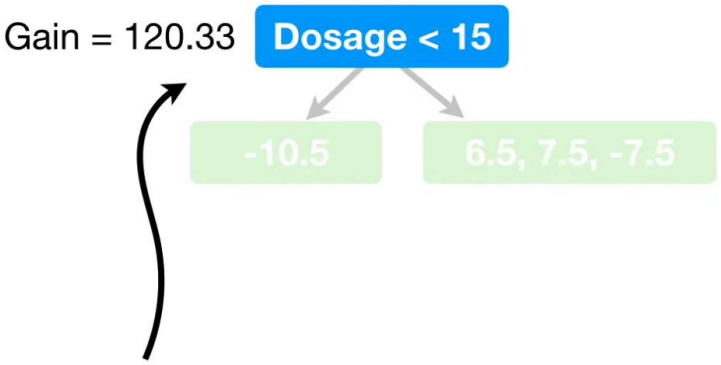

Dosage < 15
Gain = 140.17
-10.5
Dosage < 30
6.5, 7.5
-7.5
Gain
γ (gamma)
140.17 - 130 = 10.17
…we get a ***positive*** number, so we will not remove this branch and we are done pruning.


Gain = 120.33
Dosage < 15
Gain = 140.17
-10.5
Dosage < 30
6.5, 7.5
-7.5
NOTE: The Gain for the root, 120.3, is less than 130, the value for γ (gamma), so the difference will be negative.
However, because we did not remove the first branch, we will not remove the root.

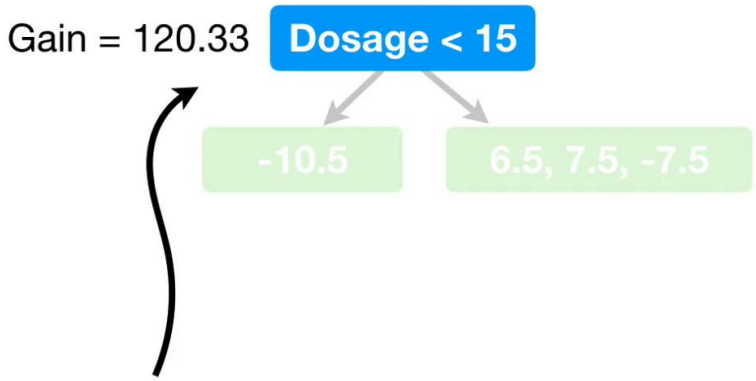

Gain = 120.33
Dosage < 15
Gain = 140.17
Dosage < 30
6.5, 7.5
-7.5
In contrast, if we set γ = 150, then we would remove this branch because…
140.17 - 150 = a negative number.
Gain
γ (gamma)



So let's remove this branch.

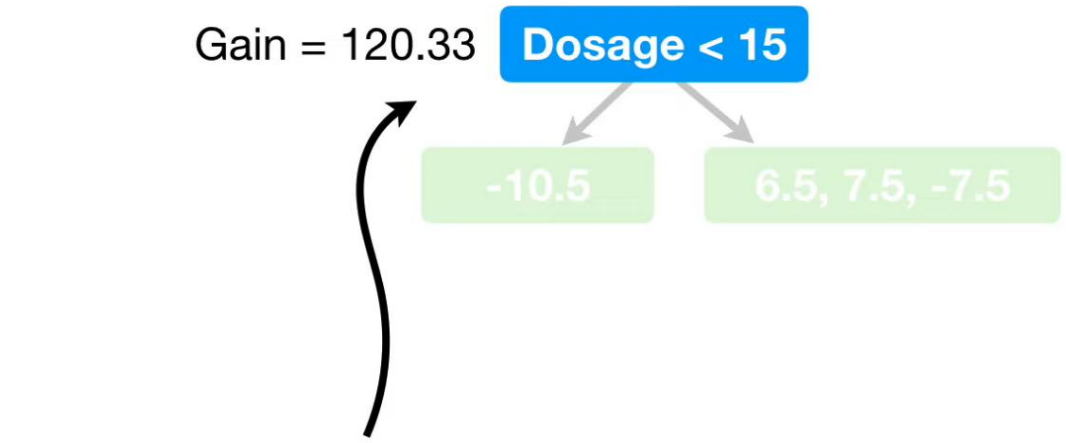


Now we will subtract ***γ* (gamma)**
from the **Gain** for the **Root**.



Since
120.33 - 150 = a negative number…

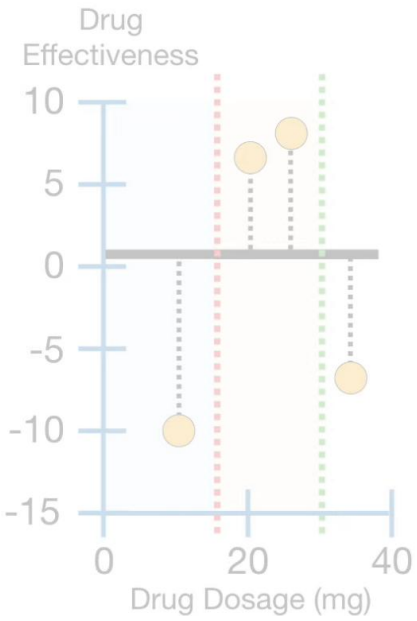
Gain ***γ* (gamma)**



…we will remove the root…

Predicted Drug Effectiveness

0.5



…and all we would be left with is the original prediction, which is pretty extreme pruning.

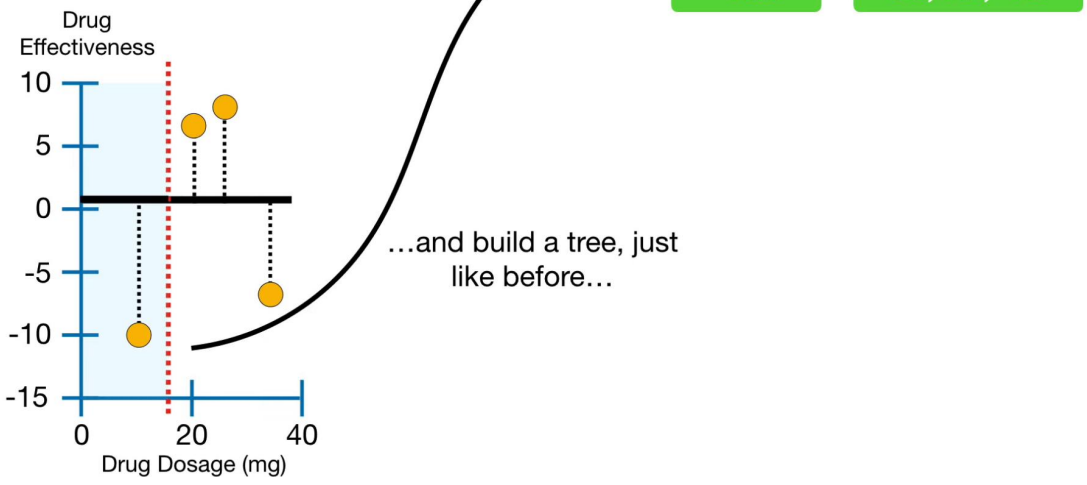
Regularization

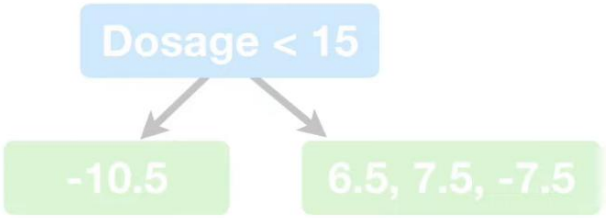
Predicted Drug Effectiveness

0.5

Predicted Drug Effectiveness

0.5



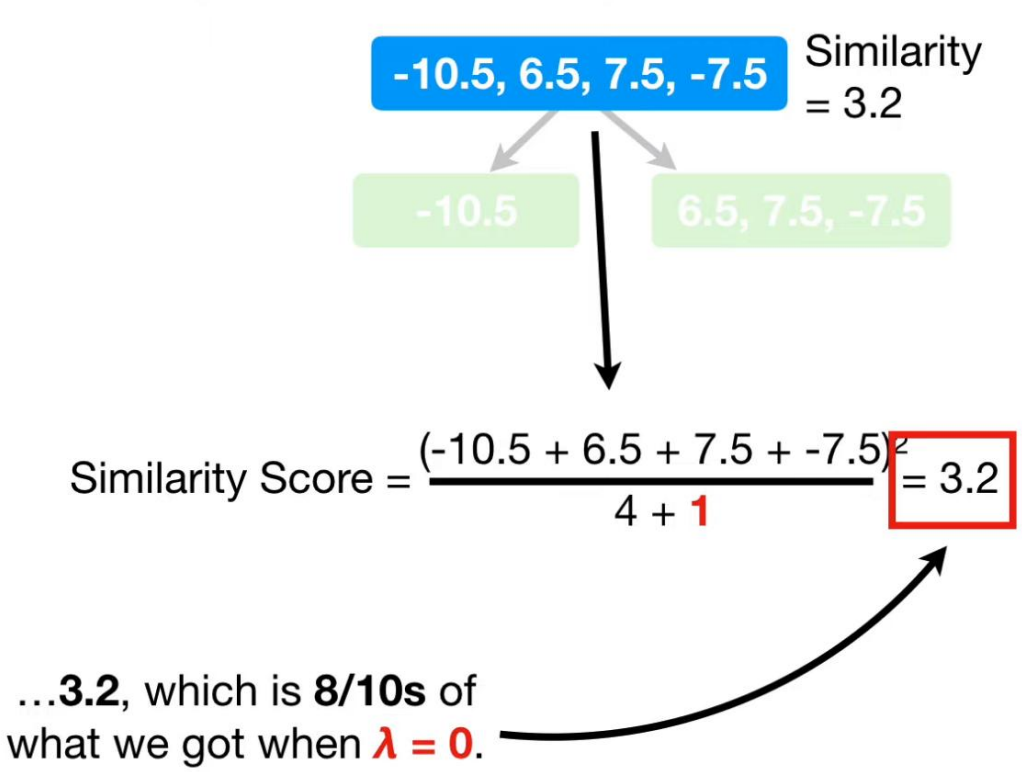


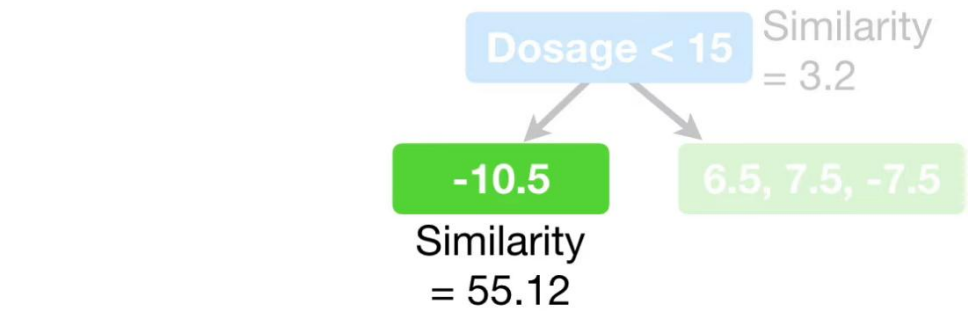
$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

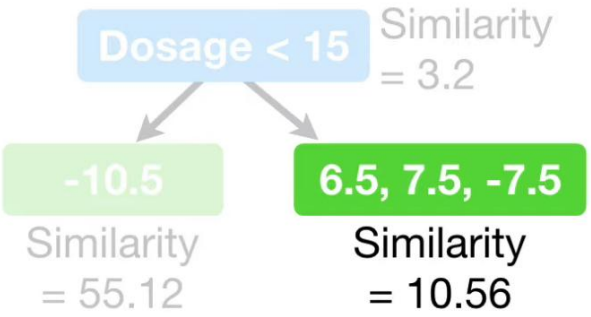
…only this time, when we calculate **Similarity Scores**, we will set **λ (lambda) = 1**.

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

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

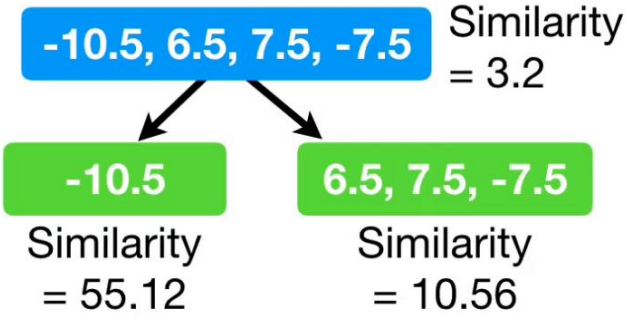

-10.5, 6.5, 7.5, -7.5
Similarity
= 3.2
-10.5
6.5, 7.5, -7.5
Similarity Score = (-10.5 + 6.5 + 7.5 + -7.5)² / 4 + 1 = 3.2
…3.2, which is 8/10s of
what we got when λ = 0.


Dosage < 15
Similarity
= 3.2
-10.5
Similarity
= 55.12
6.5, 7.5, -7.5
Similarity Score = -10.5² / 1 + 1 = 55.12
…we get 55.12, which is half
of what we got when λ = 0.

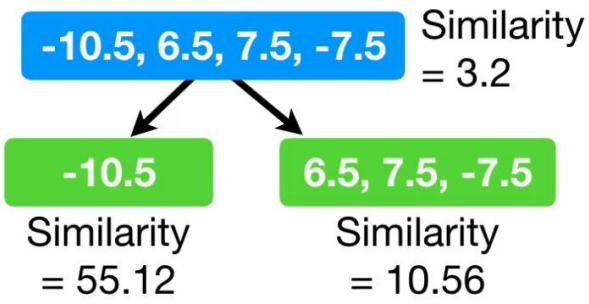


$$\text{Similarity Score} = \frac{(6.5 + 7.5 + -7.5)^2}{3 + \mathbf{1}} = 10.56$$

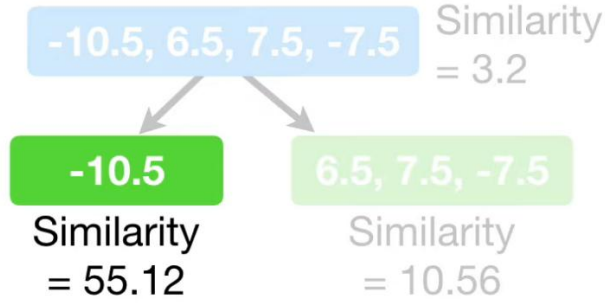
…we get **10.56**, which is **3/4s** of what we got when **λ = 0**.



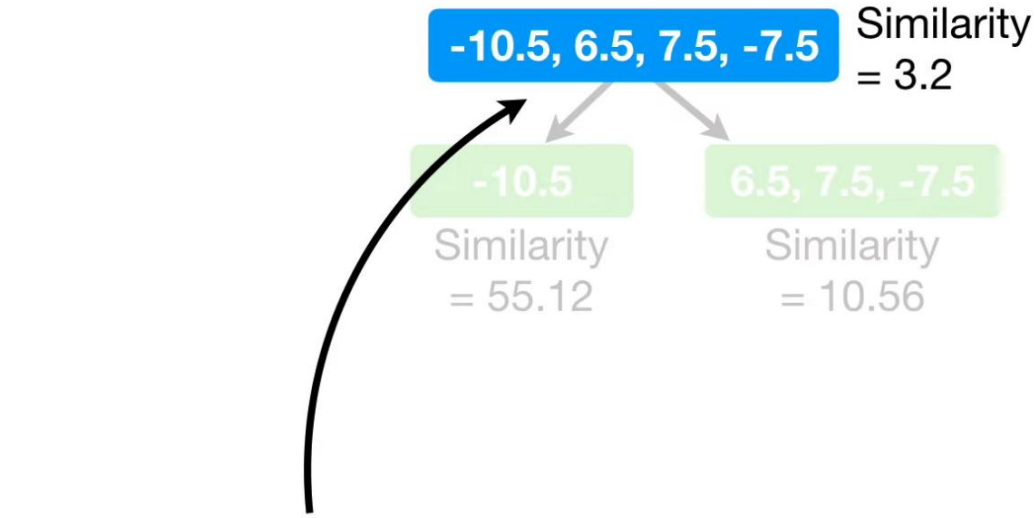
So, one thing we see is that when **λ > 0**, the **Similarity Scores** are smaller…



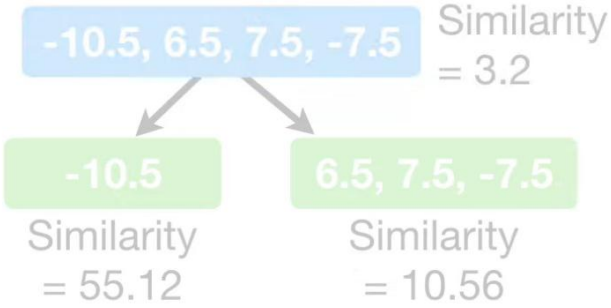
…and the amount of decrease is **inversely proportional** to the number of **Residuals** in the node.



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

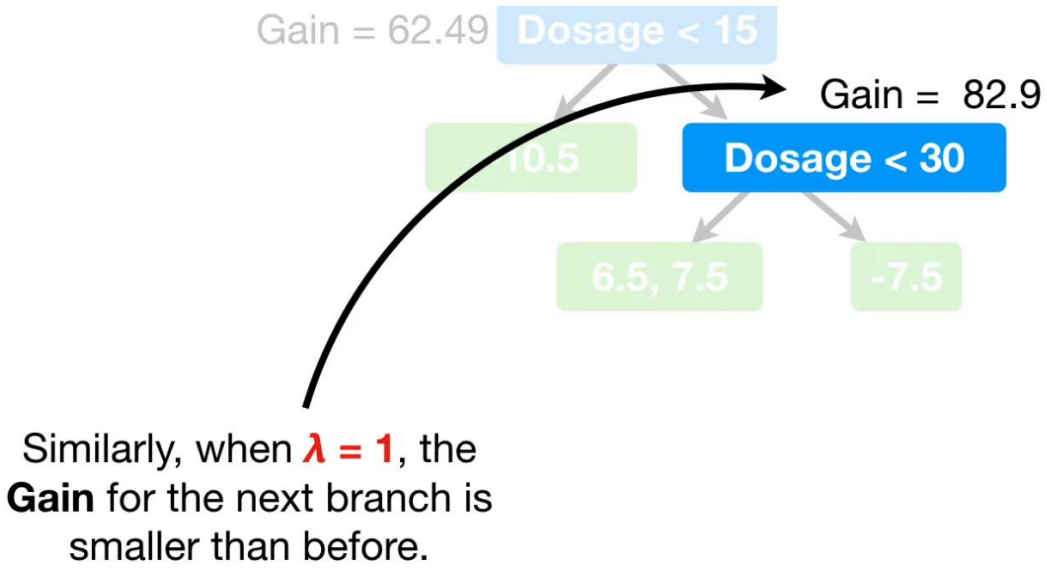


In contrast, the root had all **4** **Residuals** and the smallest decrease, **20%**.

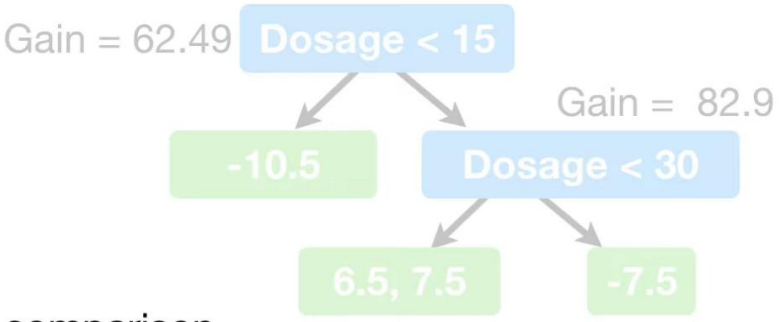


Gain = 55.12 + 10.56 - 3.2 = 62.48

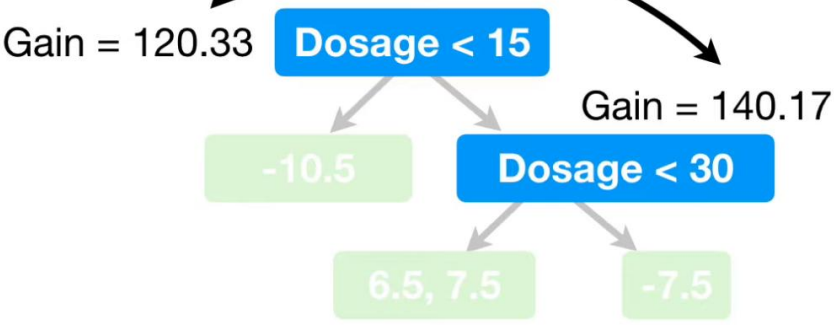
…we get **66**, which is a lot less than **120.33**, the value we got when $\lambda = 0$.

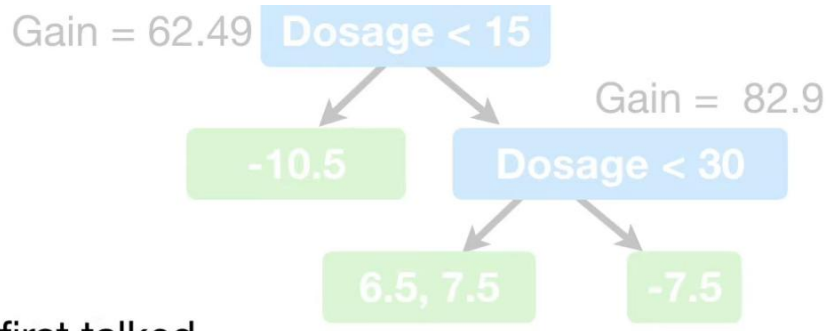


Similarly, when $\lambda = 1$, the **Gain** for the next branch is smaller than before.

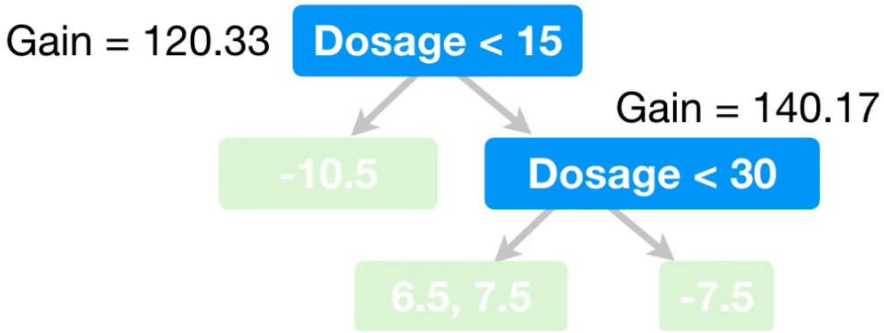


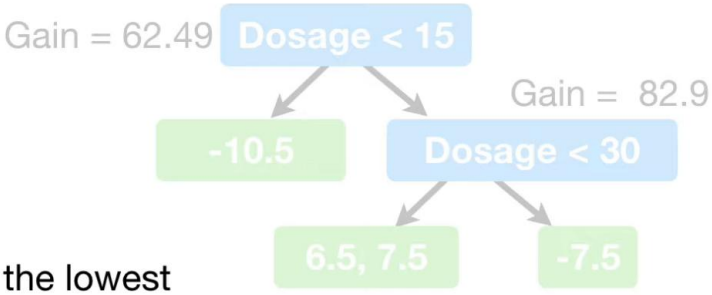
Now, just for comparison, these were the **Gain** values when $\lambda = 0$.



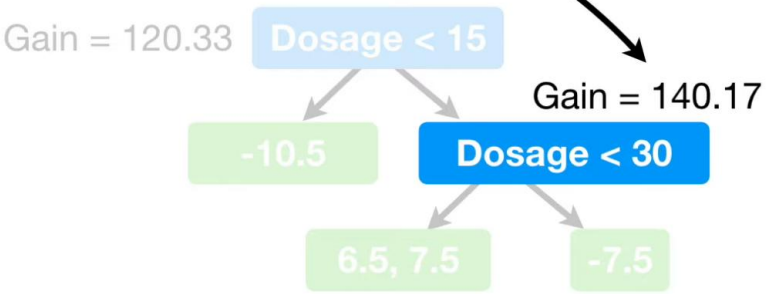


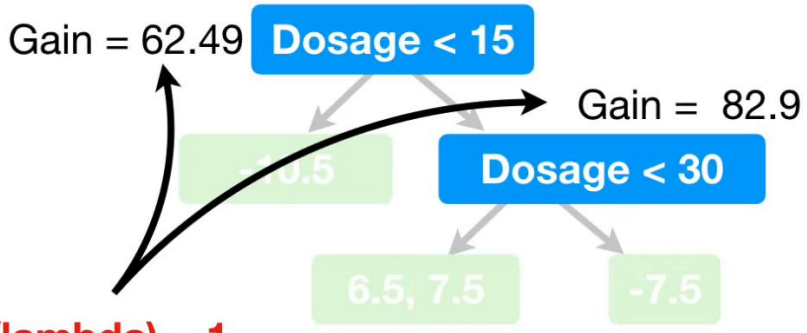
When we first talked about pruning trees, we set ***γ* (gamma) = 130**…





…and because, for the lowest branch in the first tree, **Gain - *γ* = a positive number**, we did not prune at all.



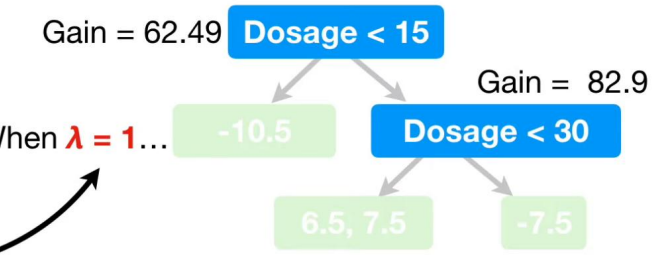


Now, with **λ (lambda) = 1**, the values for **Gain** are both **< 130**...

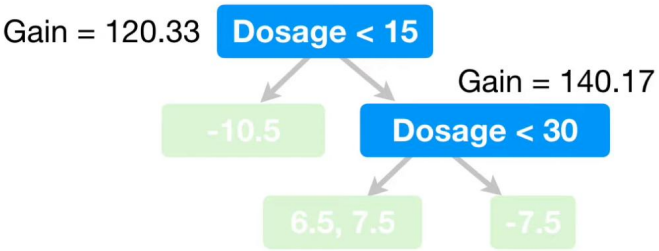
...so we would prune the whole tree away.

So when **λ > 0**, it is easier to prune leaves because the values for **Gain** are smaller.

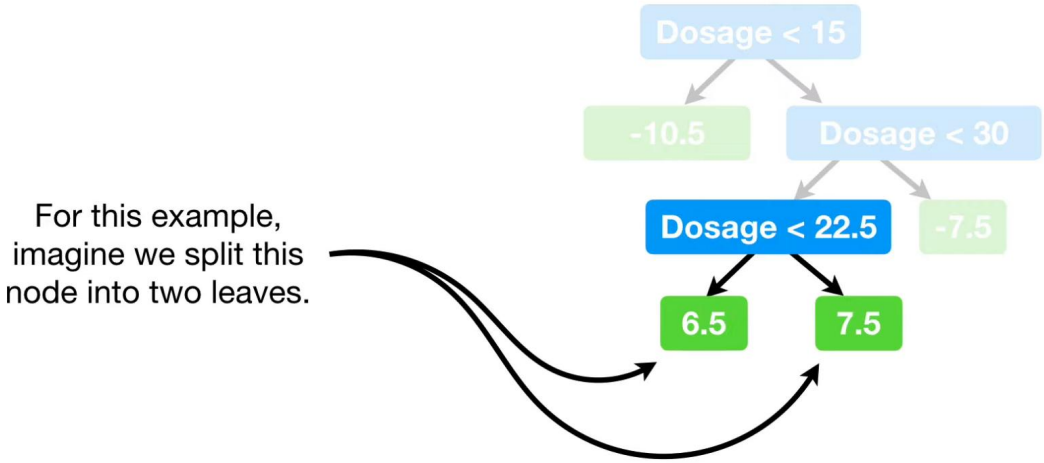
When **λ = 1**...

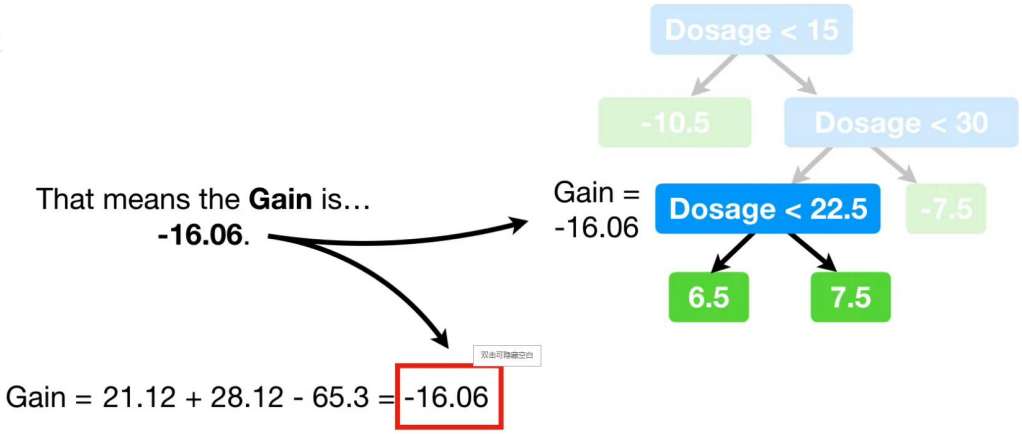


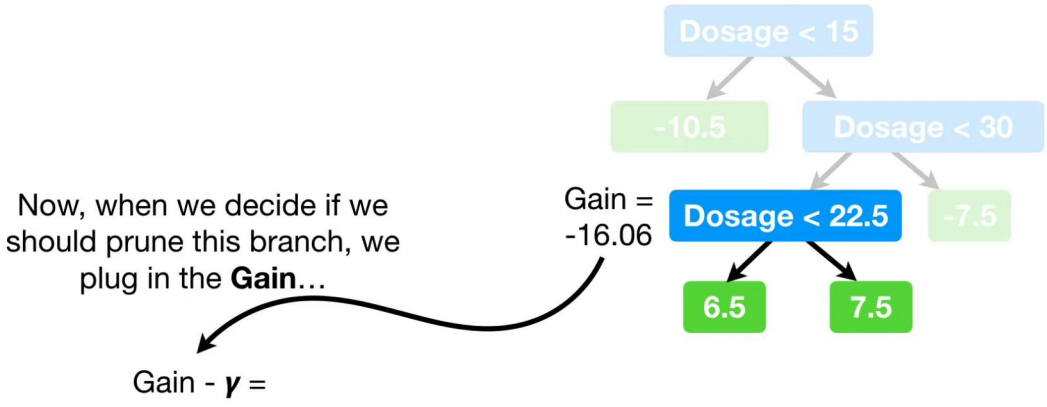
When **λ = 0**...

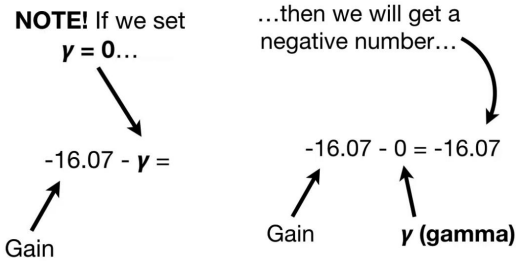


NOTE: Before we move on, I want illustrate one last feature of **λ (lambda)**.

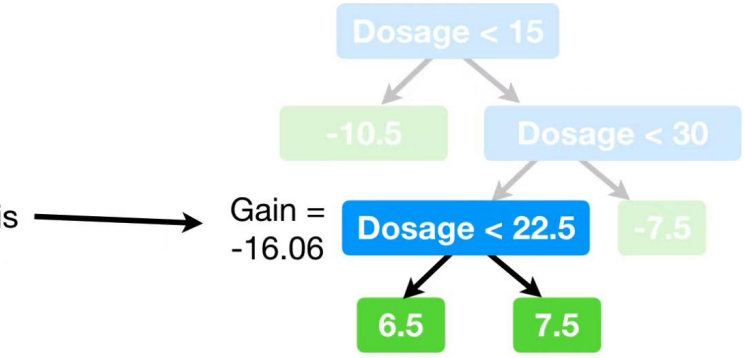

Dosage < 15
-10.5
Dosage < 30
Dosage < 22.5
-7.5
6.5
7.5
For this example, imagine we split this node into two leaves.


Dosage < 15
-10.5
Dosage < 30
Dosage < 22.5
-7.5
6.5
7.5
That means the **Gain** is… **-16.06**.
Gain = -16.06
Gain = 21.12 + 28.12 - 65.3 = -16.06


Dosage < 15
-10.5
Dosage < 30
Dosage < 22.5
-7.5
6.5
7.5
Now, when we decide if we should prune this branch, we plug in the **Gain**…
Gain = -16.06
Gain - **γ** =

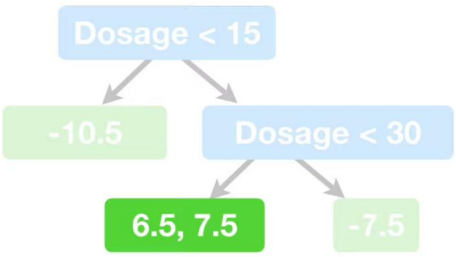

NOTE! If we set **γ = 0**…
-16.07 - **γ** =
Gain
…then we will get a negative number…
-16.07 - 0 = -16.07
Gain
γ (gamma)

…and we will prune this branch, even though $\gamma = 0$.



-16.07 - 0 = -16.07

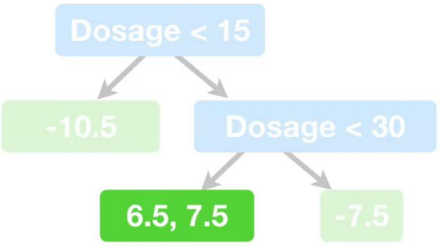
Gain — γ (gamma)



In other words, setting $\gamma = 0$ does not turn off pruning.

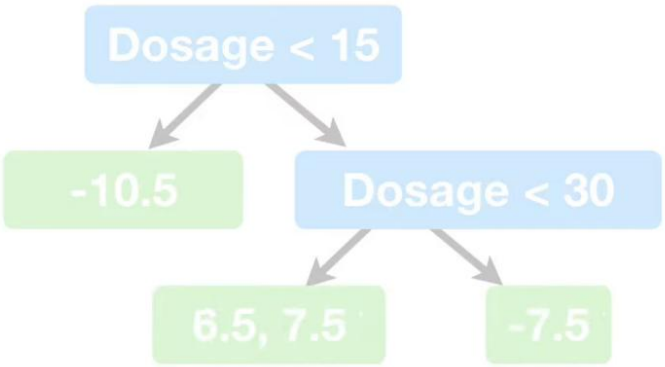
-16.07 - 0 = -16.07

Gain — γ (gamma)



On the other hand, by setting λ (lambda) = 1, λ did what it was supposed to do; it prevented over fitting the **Training Data**.

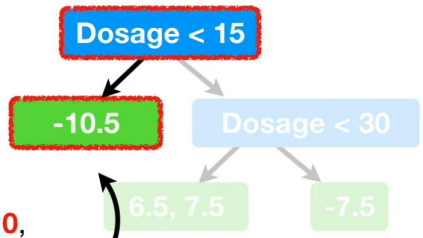
Calculate the Output Value

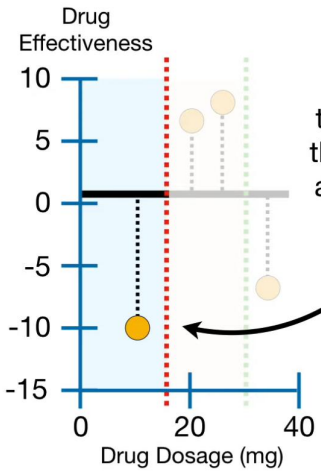


$$\text{Output Value} = \frac{\text{Sum of Residuals}}{\text{Number of Residuals} + \lambda}$$

Predicted Drug Effectiveness

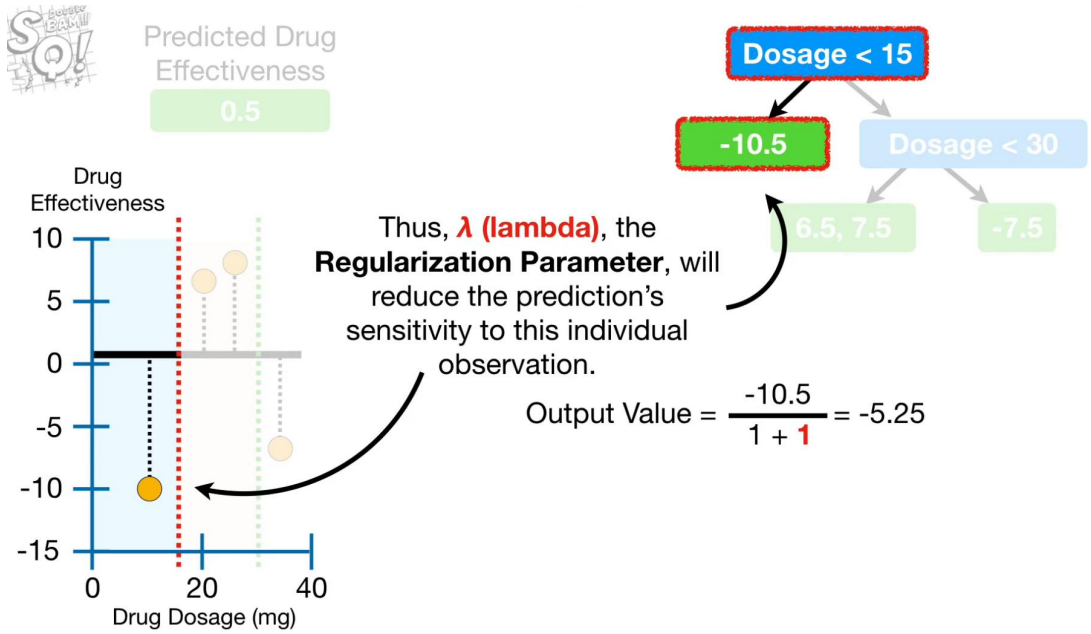
0.5

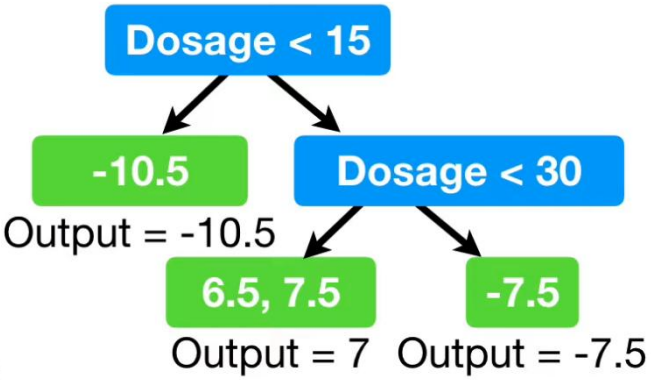




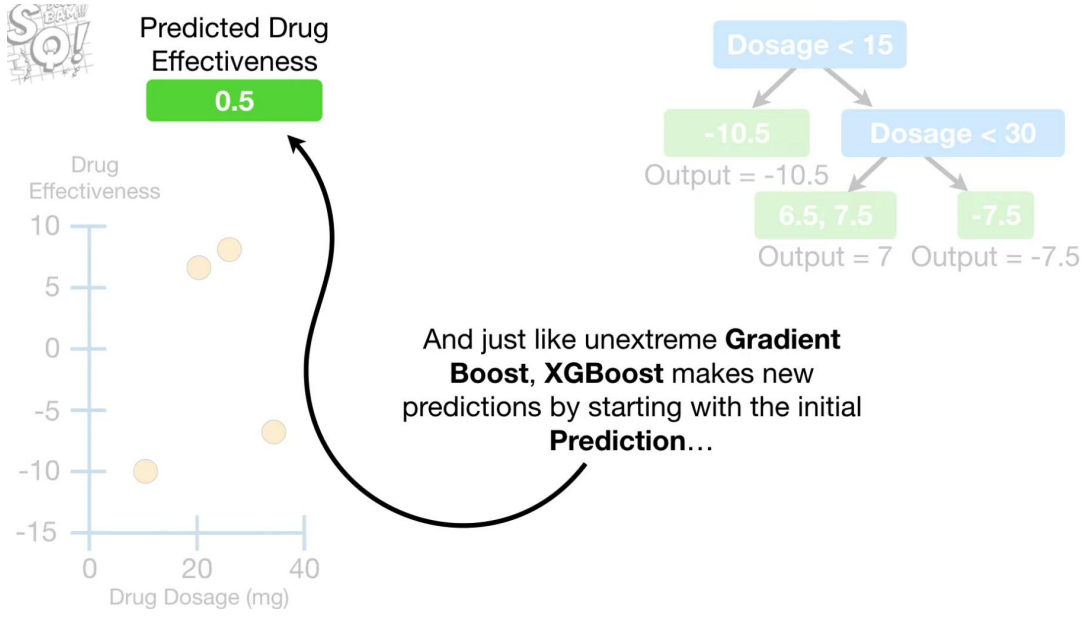
In other words, when $\lambda > 0$, then it will reduce the amount that this individual observation adds to the overall prediction.

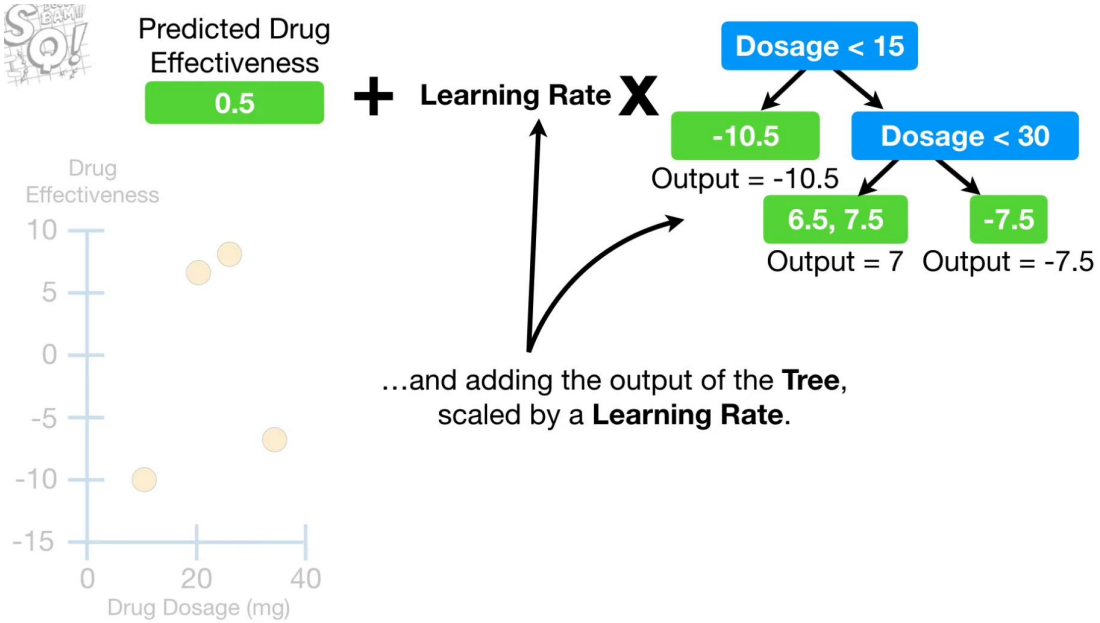
$$\text{Output Value} = \frac{-10.5}{1 + 1} = -5.25$$



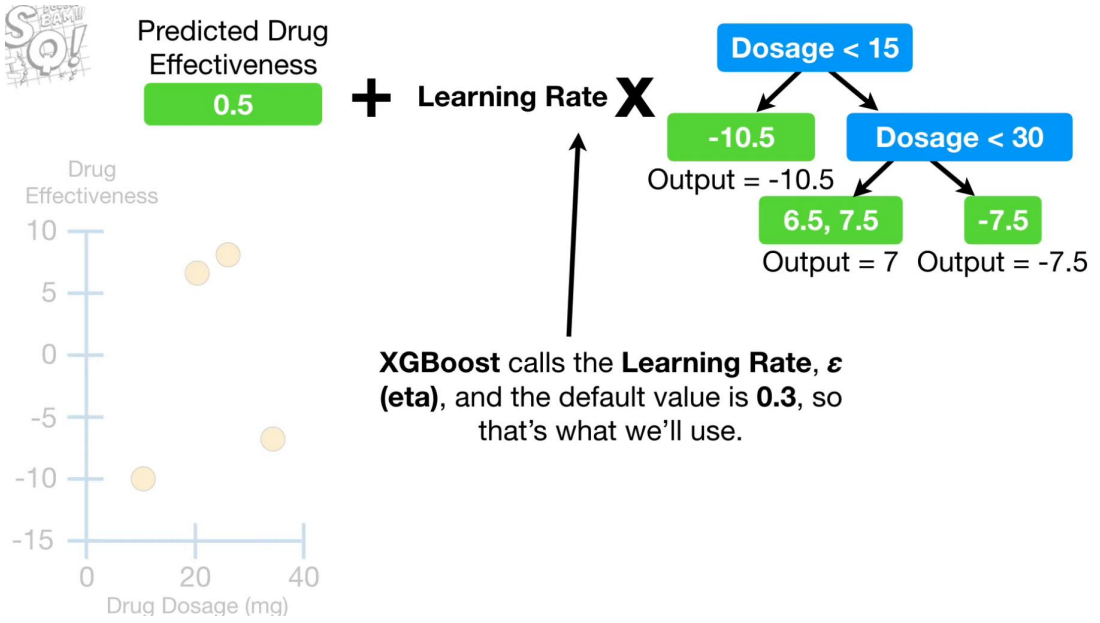


Now, at long last, the first tree is complete!

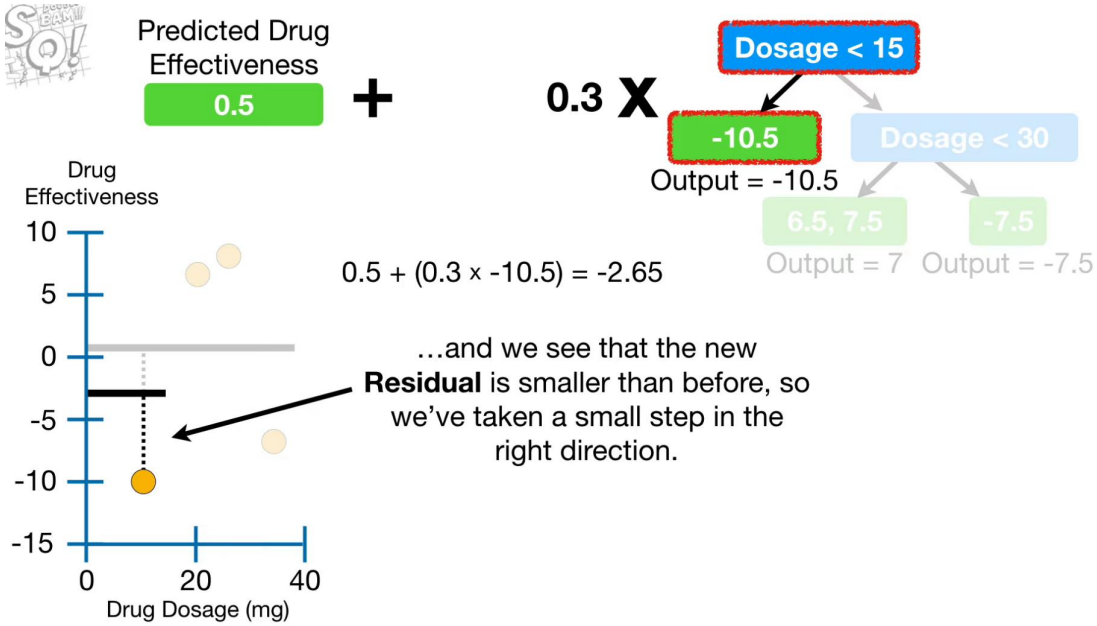


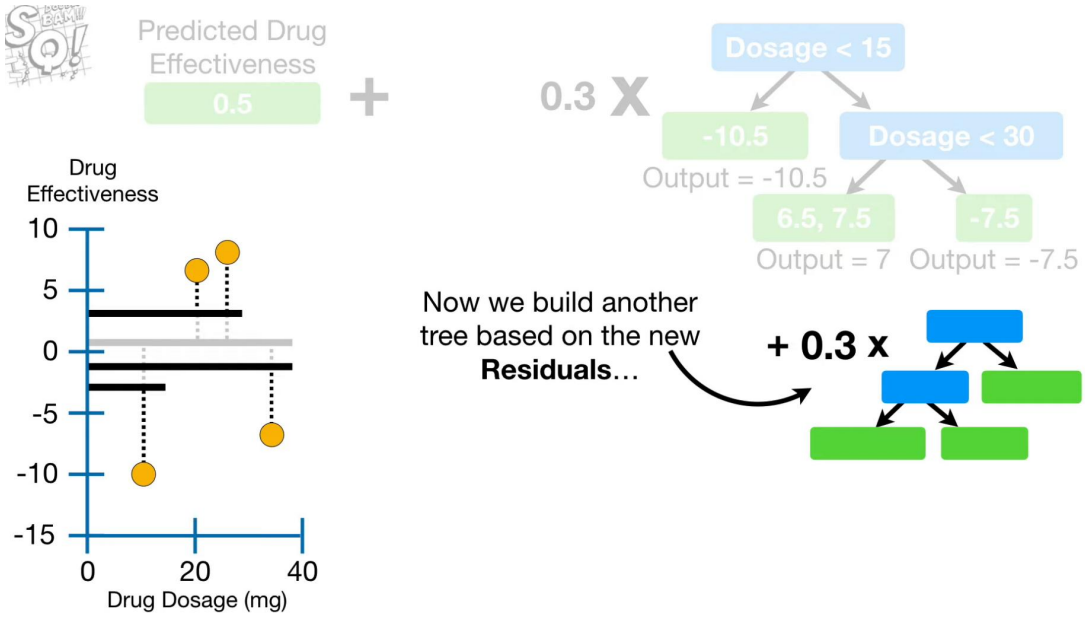


...and adding the output of the **Tree**, scaled by a **Learning Rate**.



XGBoost calls the **Learning Rate**, ***ε*** **(eta)**, and the default value is **0.3**, so that's what we'll use.


Predicted Drug Effectiveness
0.5
+
0.3 X
Dosage < 15
-10.5
Dosage < 30
Output = -10.5
6.5, 7.5
-7.5
Output = 7
Output = -7.5
Drug Effectiveness
10
5
0
-5
-10
-15
0
20
40
Drug Dosage (mg)
0.5 + (0.3 x -10.5) = -2.65
…and we see that the new **Residual** is smaller than before, so we've taken a small step in the right direction.

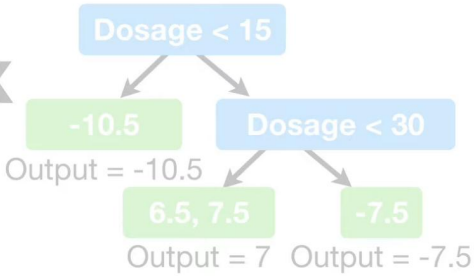

Predicted Drug Effectiveness
0.5
+
0.3 X
Dosage < 15
-10.5
Dosage < 30
Output = -10.5
6.5, 7.5
-7.5
Output = 7
Output = -7.5
Drug Effectiveness
10
5
0
-5
-10
-15
0
20
40
Drug Dosage (mg)
Now we build another tree based on the new **Residuals**…
+ 0.3 x

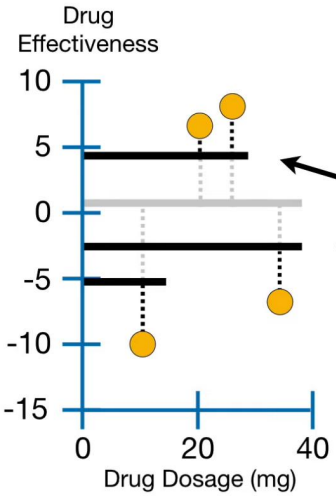
Predicted Drug Effectiveness

0.5

+

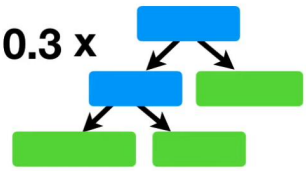
0.3 X





…and make new predictions that give us even smaller residuals…

+ 0.3 x

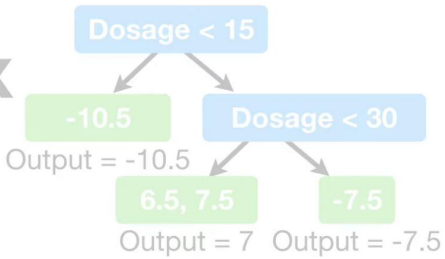


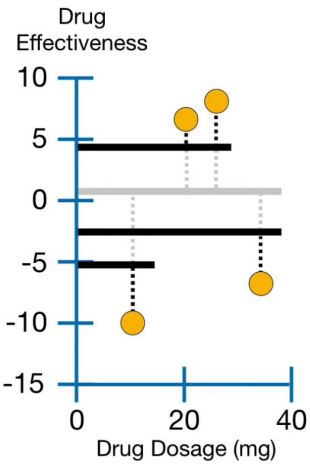
Predicted Drug Effectiveness

0.5

+

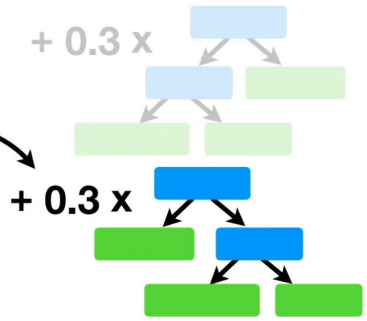
0.3 X





…and then build another tree based on the newest **Residuals**…

+ 0.3 x



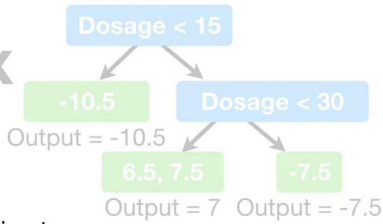
+ 0.3 x

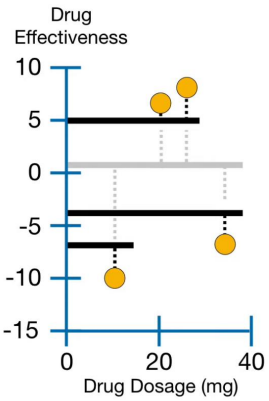
Predicted Drug Effectiveness

0.5

+

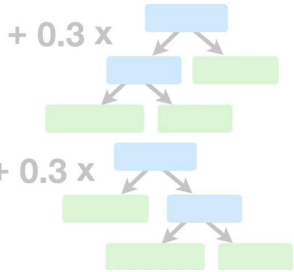
0.3 X





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

+ 0.3 x

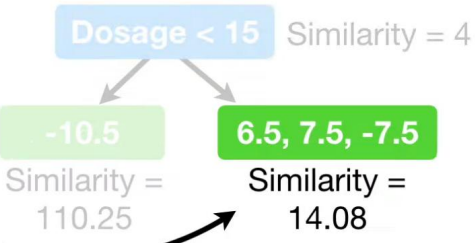


+ 0.3 x

Summary

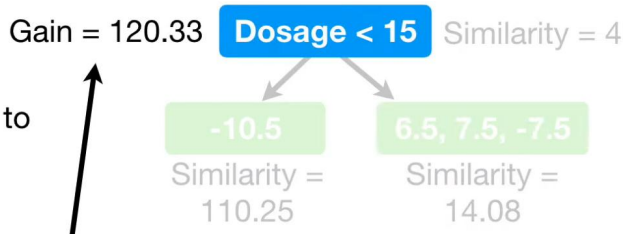
In summary, when building **XGBoost Trees** for **Regression**…

…we calculate **Similarity Scores**…



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

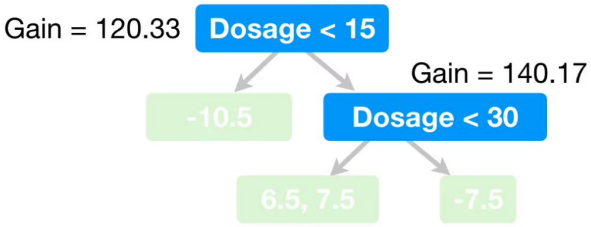
… and **Gain** to determine how to split the data…

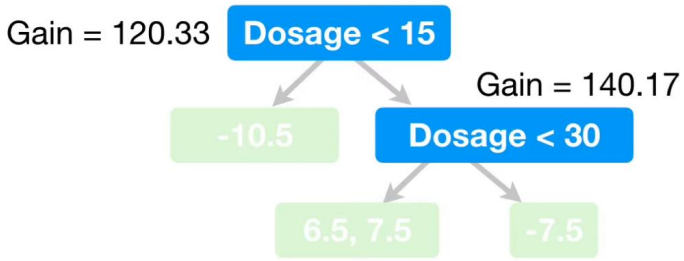


$$\text{Gain} = \text{Left}_{\text{Similarity}} + \text{Right}_{\text{Similarity}} - \text{Root}_{\text{Similarity}}$$

…and we prune the tree by calculating the differences between **Gain** values and a user defined **Tree Complexity Parameter**, ***γ*** **(gamma)**.

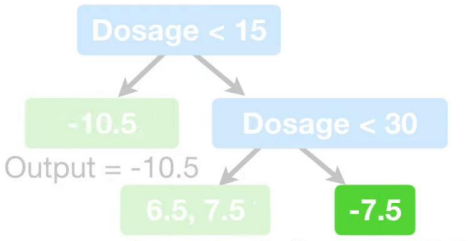
Gain - ***γ*** =





$$\text{Gain - } \gamma = \begin{cases} \text{If positive, then do not prune.} \\ \text{If negative, then prune.} \end{cases}$$

Then we calculate the **Output Values** for the remaining leaves…



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

…and lastly, **λ (lambda)** is a **Regularization Parameter** and when **λ > 0**, it results in more pruning, by shrinking the **Similarity Scores**, and it results in smaller **Output Values** for the leaves.

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

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