Constructing an XGBoost Tree: A Blueprint for Regression

A step-by-step deconstruction of the core mechanics that drive XGBoost's performance.

Similarity Score

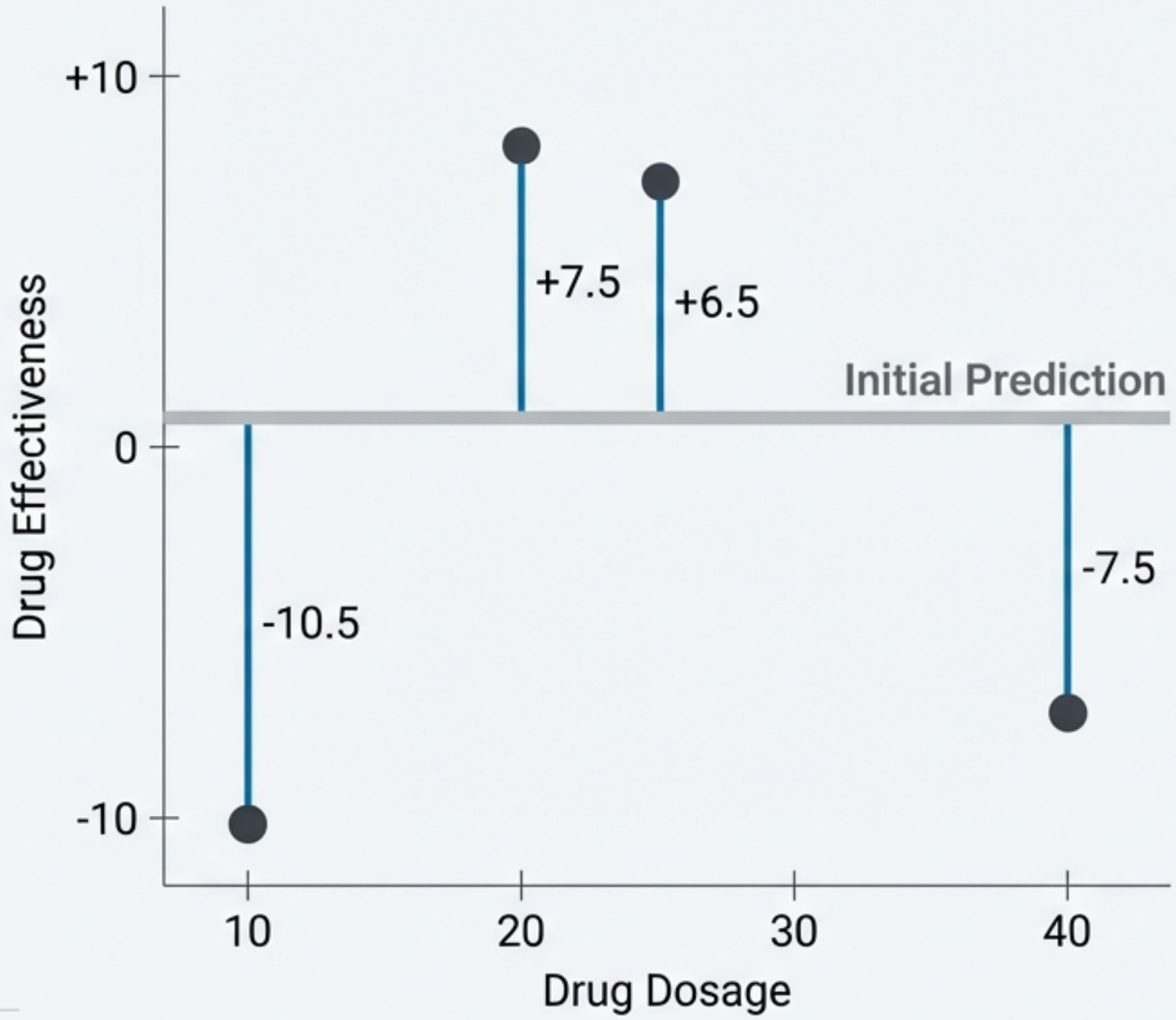
Gain

Pruning (γ)

Regularisation (λ)

NotebookLM

The Foundation: An Initial Prediction and Its Residuals



The Problem

We need to build a model to predict Drug Effectiveness based on Dosage.

The Data

Dosage	Effectiveness
10	-10
20	8
25	7
40	-7

The Starting Point

XGBoost begins with an initial prediction, by default 0.5 for all observations.

The Goal

The first tree will not be fit to the drug effectiveness values, but to the **residuals**—the errors from our initial prediction. These are the values our tree must learn to predict.

NotebookLM

Quantifying Purity with the Similarity Score

Root Node

{7.5, 6.5, -10.5, -7.5}

Definition

The Similarity Score measures how similar the residuals are within a leaf. A higher score means the residuals are more clustered and less varied.

Formula

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

Calculation (for Root Node)

For now, we set the regularisation parameter $\lambda = 0$.

Sum of Residuals $= 7.5 + 6.5 - 10.5 - 7.5 = -4$

Number of Residuals $= 4$

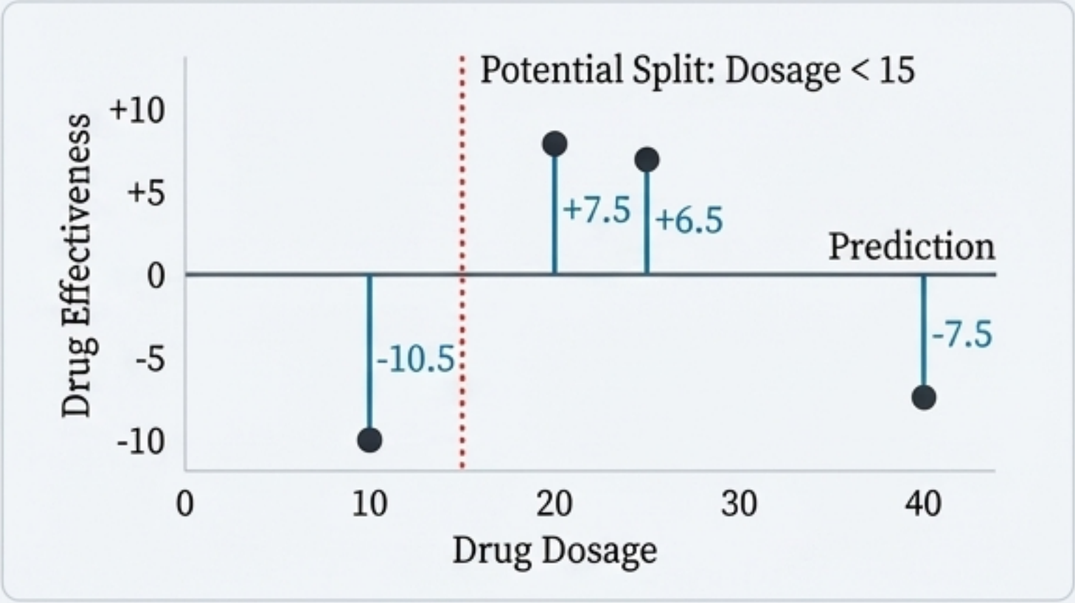
Similarity Score $= \frac{(-4)^2}{4+0} = \frac{16}{4} = \mathbf{4}$

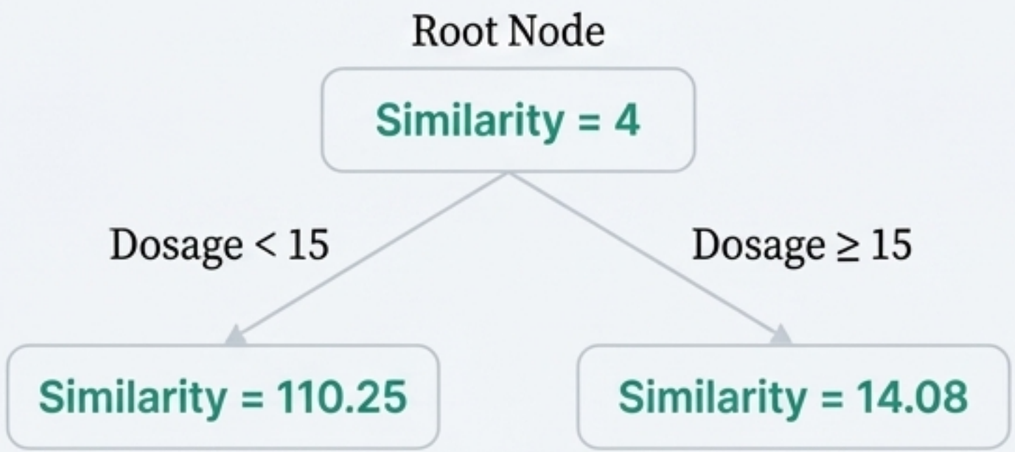
Key Insight

Notice that positive and negative residuals cancel each other out, leading to a small score. Our goal is to create splits that group residuals with the same sign together.

NotebookLM

The First Cut: Calculating Gain for a Potential Split





Step 1: Calculate Similarity for the Left Leaf (`Dosage < 15`)

Residuals: {-10.5}

Similarity Score: $\frac{(-10.5)^2}{1+0} = \mathbf{110.25}$

Step 2: Calculate Similarity for the Right Leaf (`Dosage ≥ 15`)

Residuals: {7.5, 6.5, -7.5}

Similarity Score: $\frac{(7.5+6.5-7.5)^2}{3+0} = \frac{(6.5)^2}{3} = \mathbf{14.08}$

Step 3: Calculate Gain

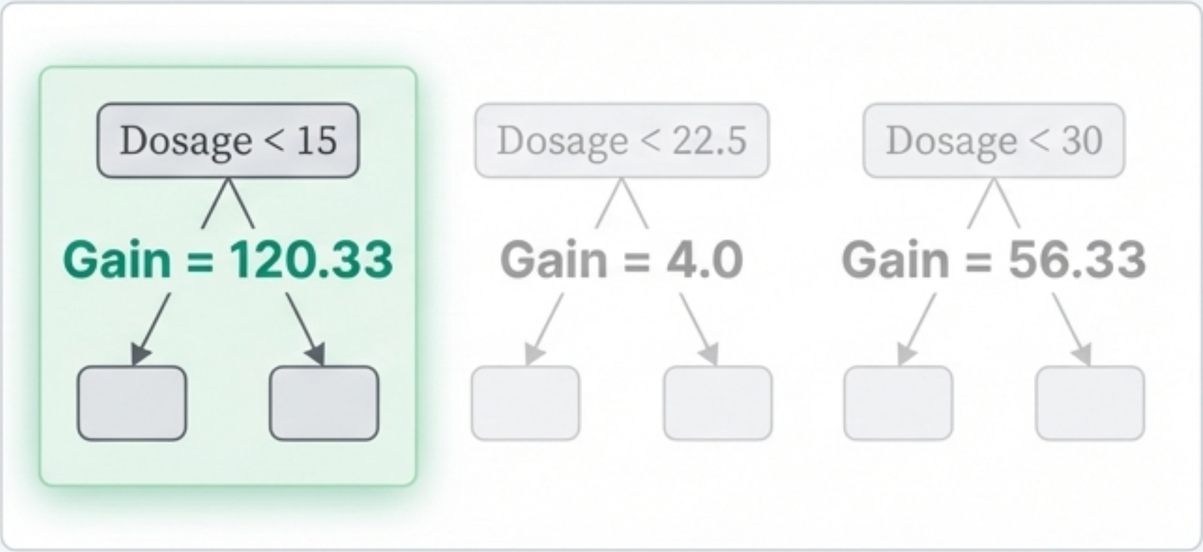
Gain quantifies the improvement in purity from splitting. It's the total Similarity of the children minus the Similarity of the parent.

Gain = Left Similarity + Right Similarity – Parent Similarity

Gain = 110.25 + 14.08 – 4 = **120.33**

NotebookLM

An Exhaustive Search for the Split with Maximum Gain



Principle

XGBoost tests every possible split point (defined as the average between consecutive data points) to find the one that delivers the highest improvement.

Summary Table

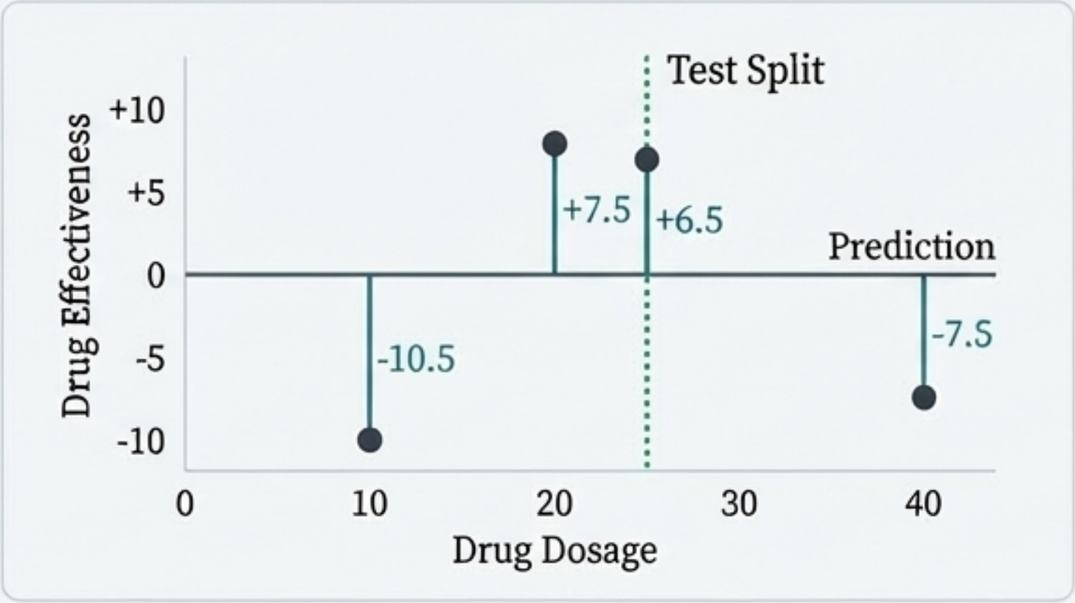
Threshold	Left Similarity	Right Similarity	Root Similarity	Gain
Dosage < 15	**110.25**	**14.08**	**4**	**120.33**
Dosage < 22.5	4.5	4.5	4	4.0
Dosage < 30	64	0.33	4	56.33

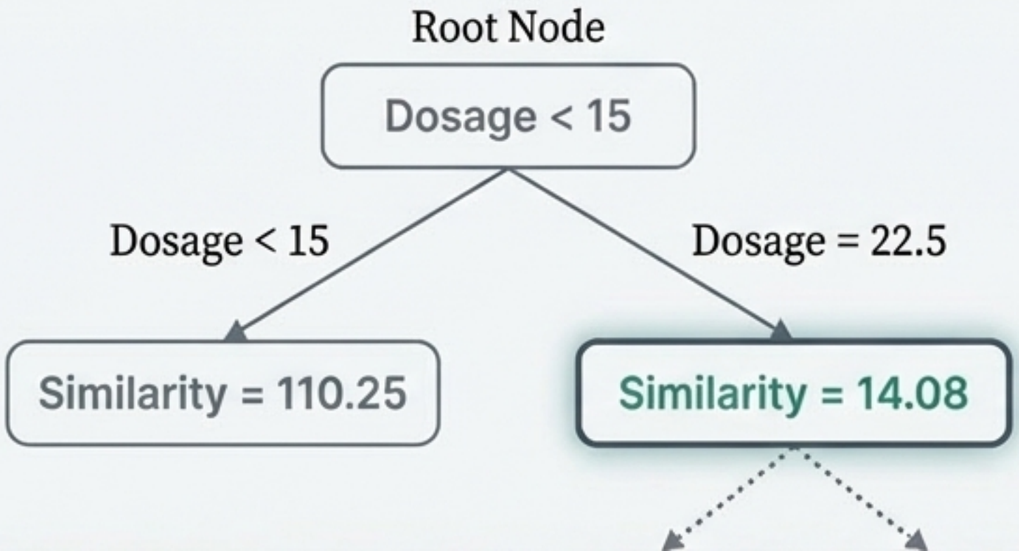
Conclusion

The split at **Dosage < 15** provides the largest gain. This becomes the first branch of our tree. The gain value of **120.33** is repeated here and coloured green.

NotebookLM

Growing the Next Level: The Process Repeats





Objective

Find the best split for the three residuals {7.5, 6.5, -7.5} in the right-hand node.

Parent Node Similarity

We already know this is **14.08**.

Test 1: Dosage < 22.5

Left Leaf Sim: 28.125, Right Leaf Sim: 14.125.
Gain = 28.125 + 14.125 – 14.08 = **28.17**.

Test 2: Dosage < 30

Left Leaf Sim: 140.167, Right Leaf Sim: 14.083.
Gain = **140.17**.

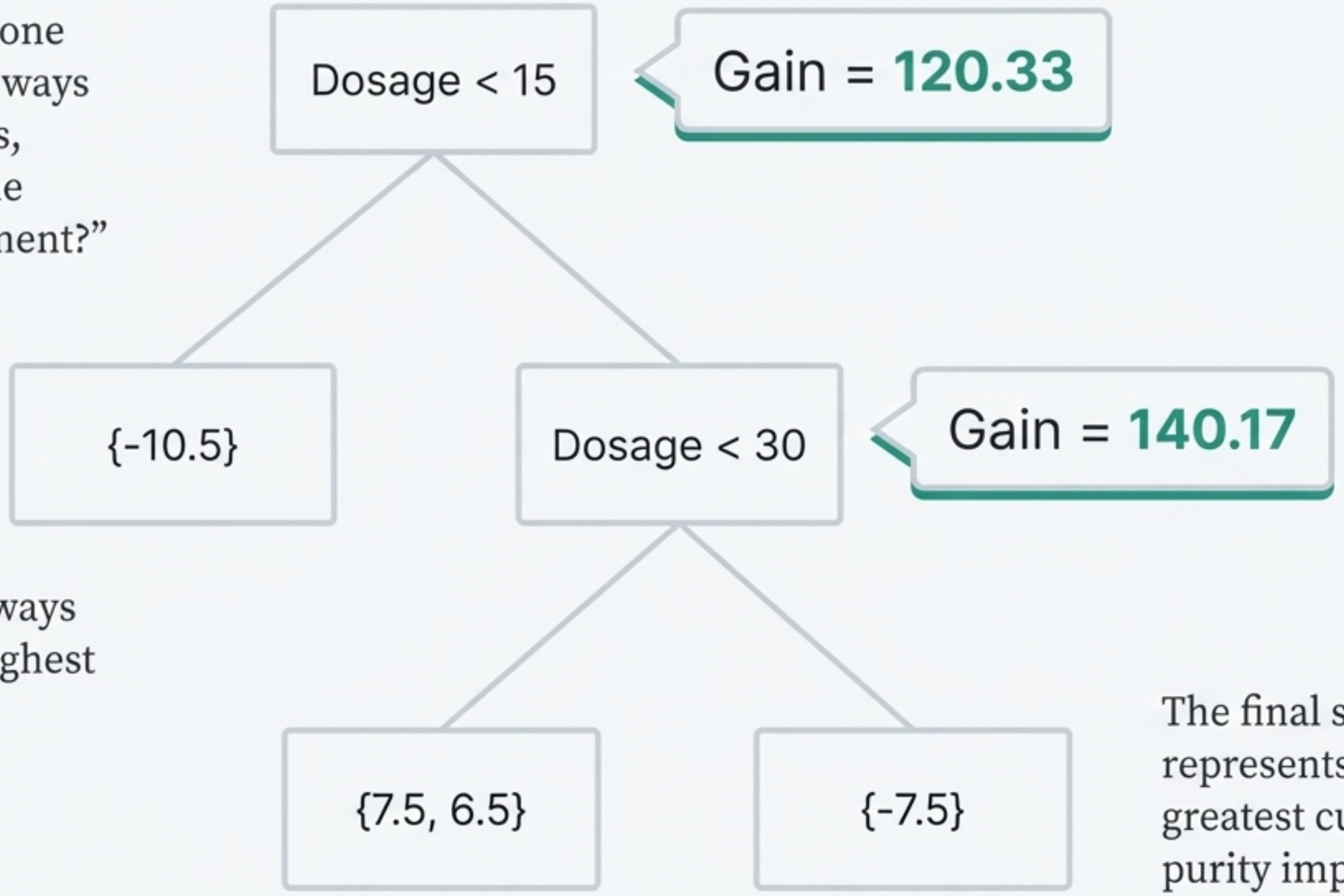
Conclusion

The split **Dosage < 30** yields a much larger gain (**140.17** vs 28.17). We add this branch to our tree.

NotebookLM

The Guiding Principle: The Tree is Built on Maximum Gain

At every stage, XGBoost asks one question: “Of all the possible ways to split this group of residuals, which single split provides the maximum possible improvement?”

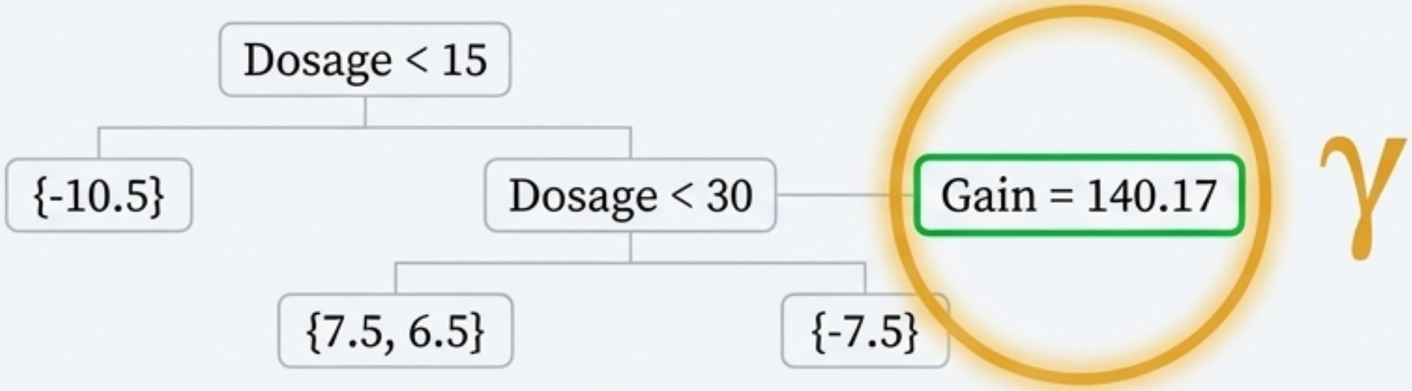


This “greedy” approach of always choosing the split with the highest Gain ensures that the tree is constructed as efficiently as possible, adding the most informative branches first.

The final structure represents the path of greatest cumulative purity improvement.

NotebookLM

Refining the Blueprint: Pruning with Gamma (γ)



The Problem

A fully grown tree might overfit the training data. Pruning simplifies the tree by removing branches that provide low value.

The Tool

Gamma (γ) is a user-defined parameter that sets the minimum Gain required for a split to be kept.

The Rule

Pruning is evaluated from the bottom up. A branch is removed if (Gain – γ) is negative.

$\boxed{(\text{Gain} - \gamma)}$

γ = 130 (Low Penalty)

140.17 (Gain) – 130 (γ) = +10.17

Positive. The branch is kept. ✅
The algorithm stops, as it won't prune a parent if its child is kept.

γ = 150 (High Penalty)

140.17 (Gain) – 150 (γ) = –9.83

Negative. The branch is removed. ❌
The algorithm would then proceed to check the root node.

NotebookLM

A Deeper Control: Regularisation with Lambda (λ)

Definition

Lambda (λ) is a regularisation parameter that reduces the model's sensitivity to individual observations. It is applied during the calculation of Similarity Scores and leaf output values.

Purpose

To prevent overfitting by penalising complexity in a different way to Gamma.

Let's rebuild the tree with $\lambda = 1$.

The Formula Revisited

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

Impact

Adding λ to the denominator consistently reduces all Similarity Scores. This effect is strongest on leaves with fewer residuals.

Comparison Table

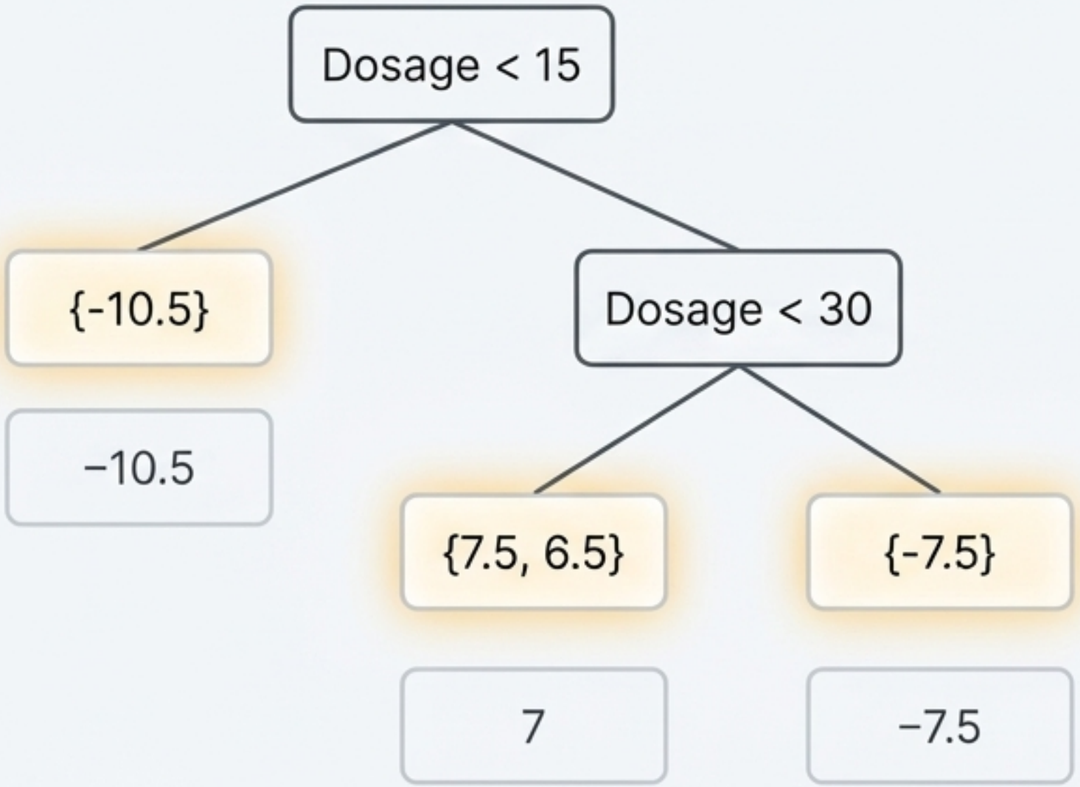
Split	Gain (λ = 0)	Gain (λ = 1)	% Decrease
Root	120.33	66.0	45.1%
2nd Branch	140.17	65.3	53.4%

Conclusion

With $\lambda > 0$, the Gain values are significantly smaller, making the tree far more susceptible to being pruned by Gamma (γ).

NotebookLM

Determining the Output of Each Leaf



The Formula

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

Key Insight

This is similar to the Similarity Score, but the sum of residuals is not squared. With $\lambda = 0$, this is simply the average of the residuals in the leaf.

Example Calculations (Using our final tree)

Leftmost Leaf ($\lambda = 0$): Output = −10.5 / (1 + 0) = −10.5

Middle Leaf ($\lambda = 0$): Output = (7.5 + 6.5) / (2 + 0) = 7

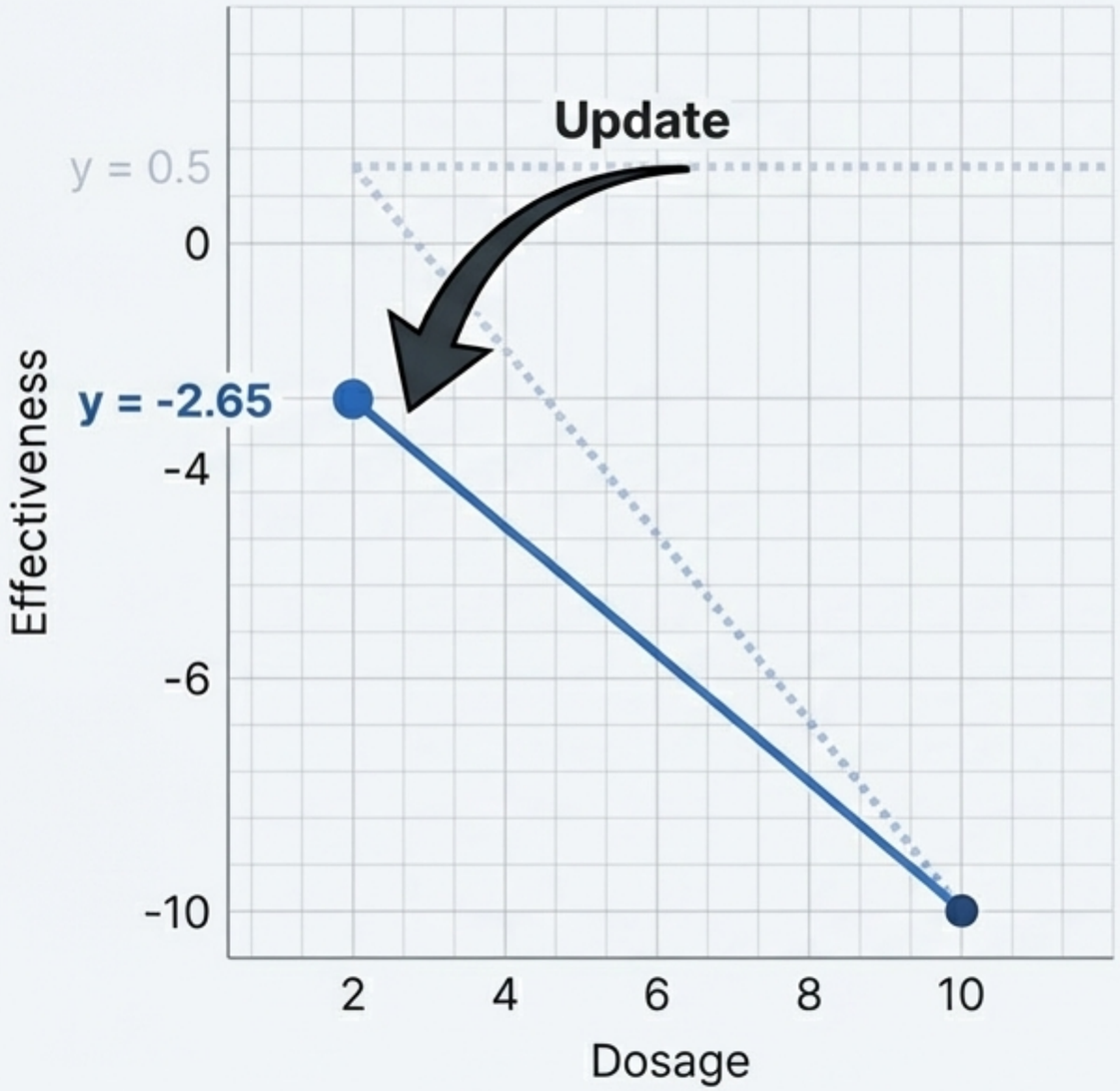
Rightmost Leaf ($\lambda = 0$): Output = -7.5 / (1 + 0) = −7.5

The Effect of Lambda

If $\lambda > 0$, it shrinks the output values, reducing the influence of any single tree on the final prediction. For $\lambda=1$, the output for the leftmost leaf would be −10.5 / (1+1) = −5.25.

NotebookLM

The Final Assembly: Updating Predictions with a Learning Rate



The Update Formula

New Prediction = Previous Prediction + η * (Tree's Output Value)

The Learning Rate (Eta, η)

XGBoost doesn't add the full output of the tree at once. Instead, it scales it by a learning rate, η (eta). This prevents overfitting and allows for subsequent trees to correct remaining errors. A typical default is η = 0.3.

Example Calculation (for Dosage = 10)

This observation falls into the leftmost leaf, with output -10.5.

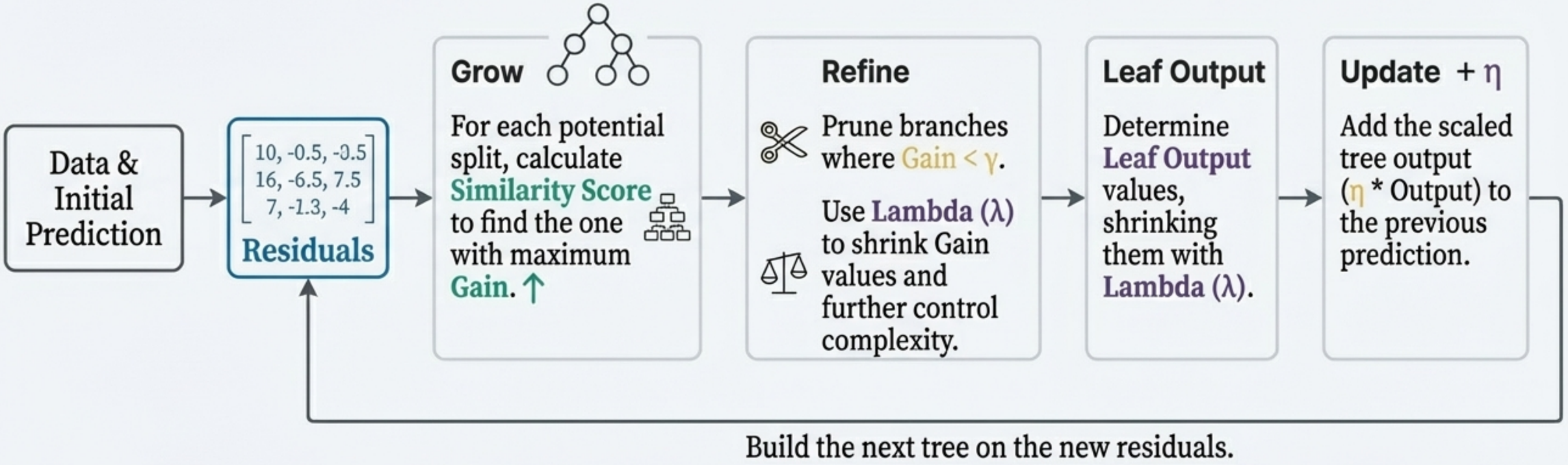
New Prediction = 0.5 + 0.3 * (-10.5) = 0.5 - 3.15 = -2.65

The Result

The new prediction is closer to the actual value, and the residual is smaller. The model has taken one small step in the right direction. This process is repeated, with new trees being built on the new residuals.

NotebookLM

The XGBoost Blueprint: A Summary of the Construction Process



XGBoost's performance stems from this principled and highly-parameterised process. Each tree is not just a decision tool, but a carefully constructed component, **optimised** for Gain and regularised **against overfitting** at every step of its creation.

NotebookLM