XGBoost Regression: Explain It To Me Like I'm 10

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

Shreya Rao August 24, 2021

towards data science

Getting Started

When I was just starting on my quest to understand Machine Learning algorithms, I would get overwhelmed with all the math-y stuff. I found it difficult to understand the math behind an algorithm without fully grasping the intuition. So I would gravitate towards sources that completely broke down the algorithm into simple steps and made it digestible to someone who never even heard the word *Algorithm* before. Okay, that is a blatant exaggeration, but you know what I mean. So that's what I'm attempting to do now. Explaining the intuition behind the XGBoost Algorithm to a 10-year-old. Here goes!

Let's start with our training dataset that consists of five people. We recorded their ages, whether or not they have a master's degree, and their salary (in thousands). Our goal is to predict *Salary* using the XGBoost Algorithm.

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

Step 1: Make an Initial Prediction and Calculate Residuals

This prediction can be anything. But let's assume our initial prediction is the average value of the variables we want to predict.

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

We can calculate residuals using the following formula:

$$\text{Residuals} = \text{Observed Values} - \text{Predicted Values}$$

Here, our Observed Values are the values in the *Salary* column and all Predicted Values are equal to 70 because that is what we chose our initial prediction to be.

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

Step 2: Build an XGBoost Tree

Each tree starts with a single leaf and all the residuals go into that leaf.

-20, 0, 10, -5, 15

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

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

Regularization Parameter

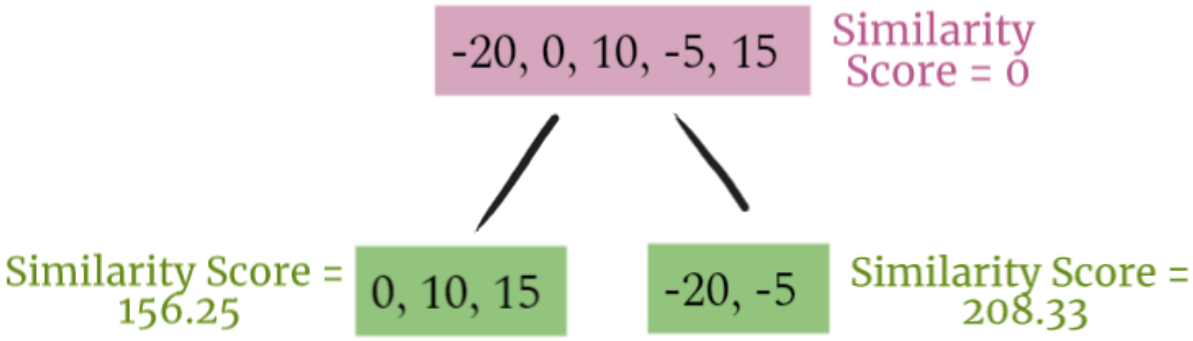
λ (lambda) is a regularization parameter that reduces the prediction's sensitivity to individual observations and prevents the overfitting of data (this is when a model fits exactly against the training dataset). The default value of λ is 1 so we will let λ = 1 in this example.

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

Now we should see if we can do a better job clustering the residuals if we split them into two groups using thresholds based on our predictors — *Age* and *Master's Degree?.* Splitting the *Residuals* basically means that we are adding branches to our tree.

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

And then calculate the **Similarity Scores** for the left and right leaves of the above split:

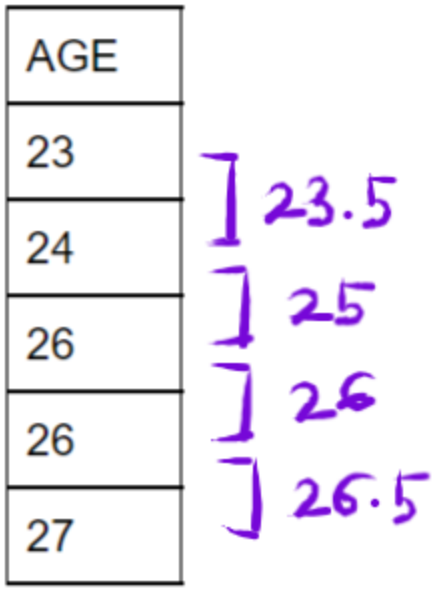


Now we need to quantify how much better the leaves cluster similar *Residuals* than the root does. We can do this by calculating the **Gain** of splitting the *Residuals* into two groups. If the **Gain** is positive, then it's a good idea to split, otherwise, it is not.

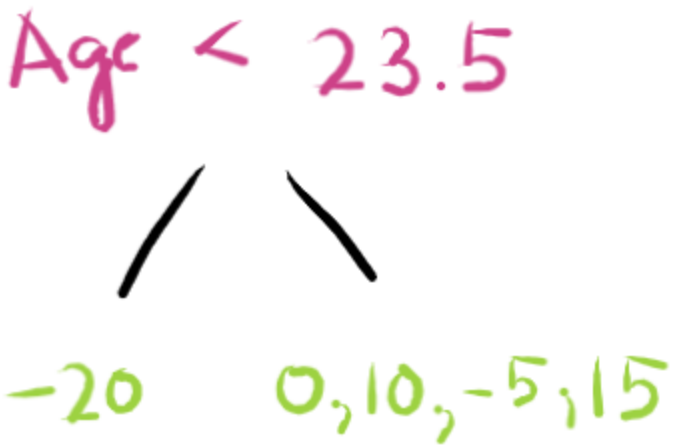
$$\text{Gain} = \text{Left}_{\text{Similarity}} + \text{Right}_{\text{Similarity}} - \text{Root}_{\text{Similarity}}$$

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

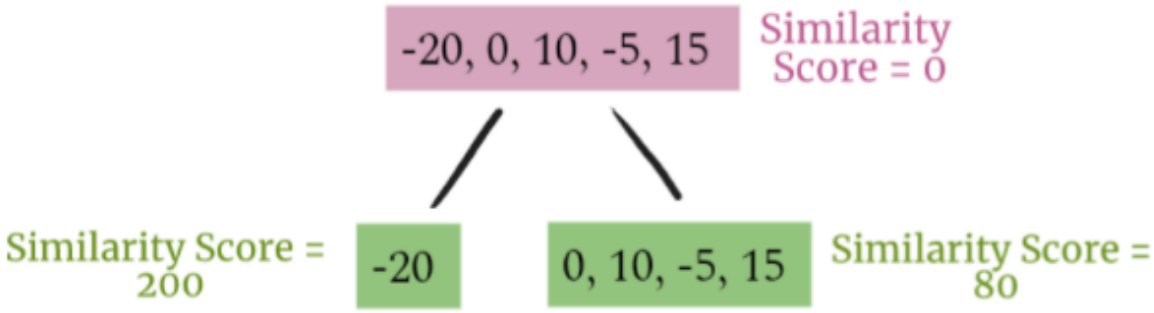
Then I need to compare this **Gain** to those of the splits in *Age*. Since *Age* is a continuous variable, the process to find the different splits is a little more involved. First, need to arrange the rows of our dataset according to the ascending order of*Age*. Then we calculate the average values of the adjacent values in *Age*.



Now we split the *Residuals* using the four averages as thresholds and calculate **Gain** for each of the splits. The first split uses *Age* < *23.5*:

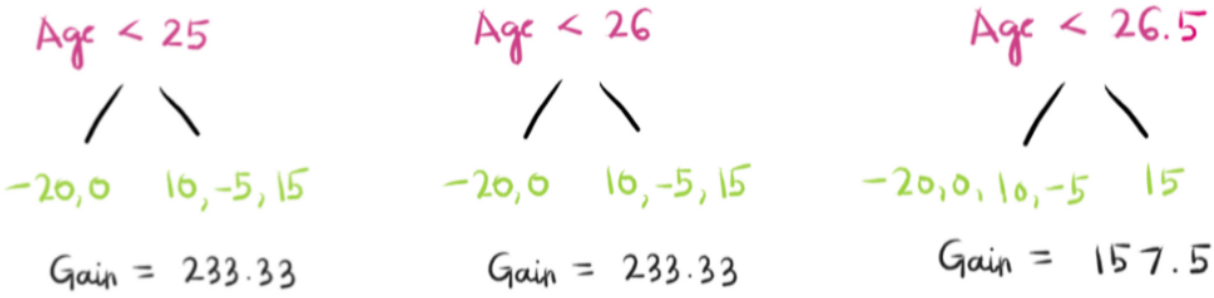


For this split, we find the **Similarity Score** and **Gain** the same way we did for *Master's Degree?*



$$\text{Gain} = 200 + 80 - 0 = 280$$

Do the same thing for the rest of the *Age* splits:



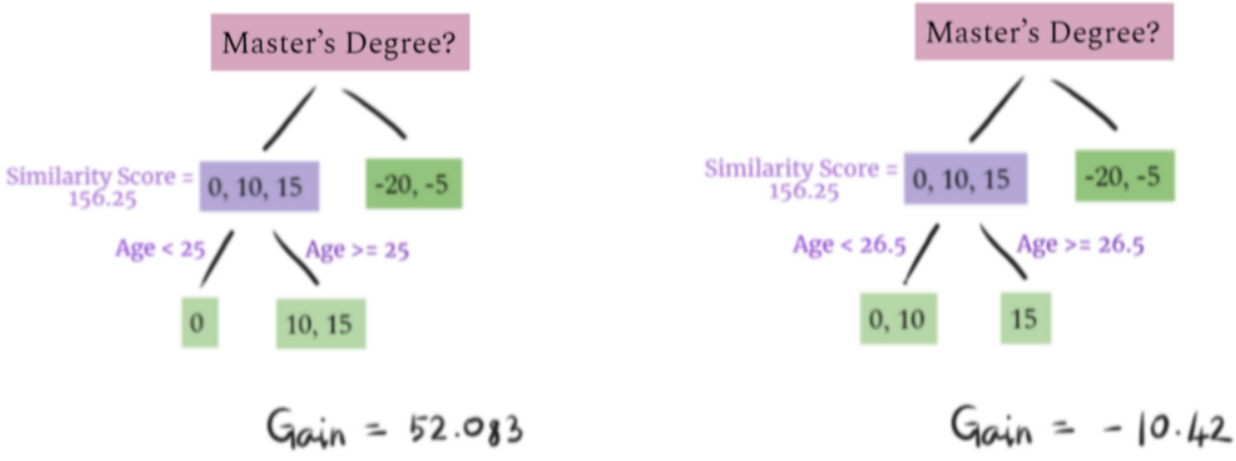
Out of the one *Mater's Degree?* split and four *Age* splits, the *Master's Degree* splithas the greatest **Gain** value, so we'll use that as our initial split. Now we can add more branches to the tree by splitting our *Master's Degree?* leaves again using the same process described

above. But, only this time, we use the initial *Master's Degree?* leaves as our root nodes and try splitting them by getting the greatest **Gain** value that is greater than 0.

Let's start with the left node. For this node, we only consider the observations that have the value 'Yes' in *Master's Degree?* because only those observations land in the left node.

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

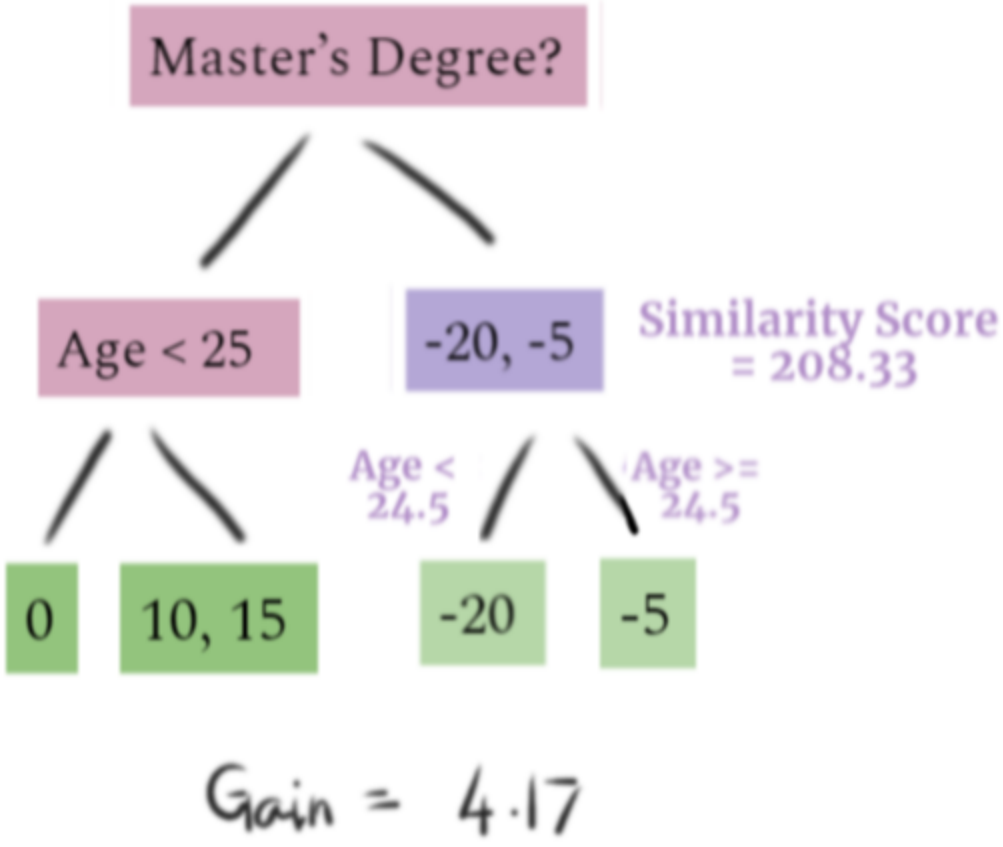
So we calculate the **Gain** of the *Age* splits using the same process as before, but this time using the *Residuals* in the highlighted rows only.



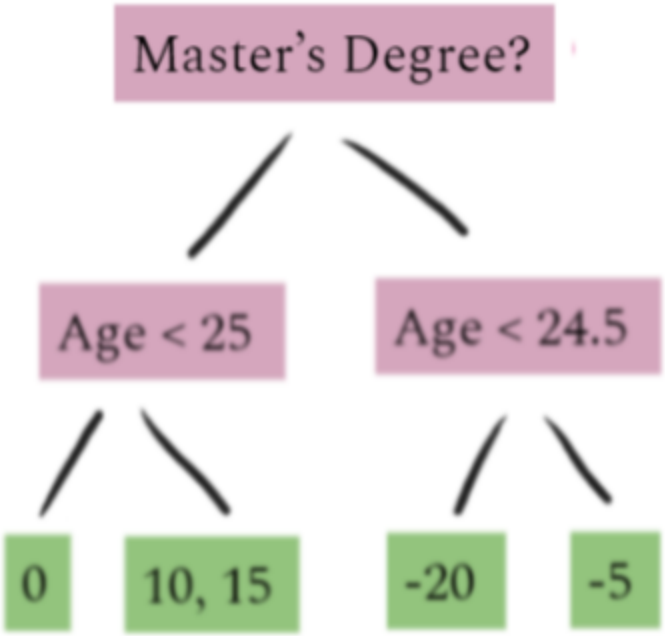
Since only *Age* < *25* gives us a positive **Gain**, we split the left node using this threshold. Moving onto our right node, we only look at values with 'No' values in *Master's Degree?*

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

We only have two observations in our right node, so the only split possible is *Age* < *24.5* because that is the average of the two *Age* values in the highlighted rows.

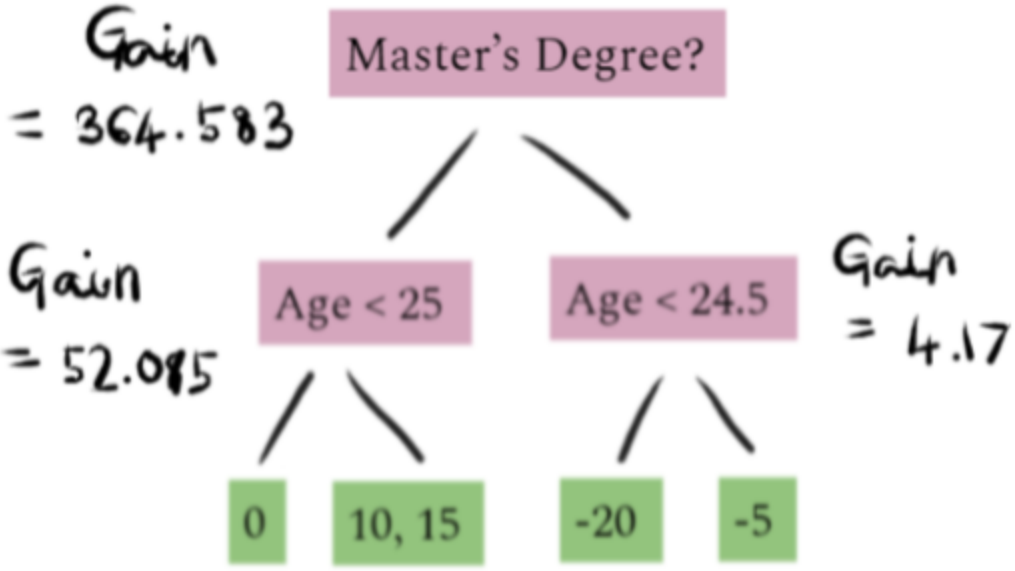


The **Gain** of this split is positive, so our final tree is:

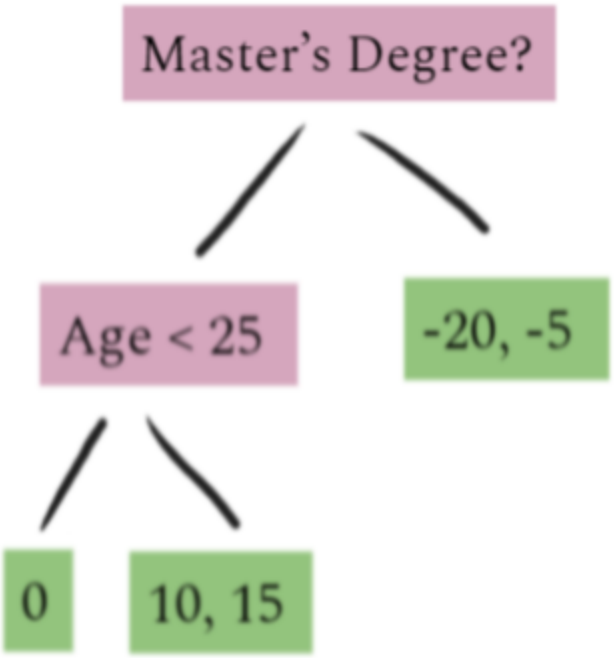


Step 3: Prune the Tree

Pruning is another way we can avoid overfitting the data. To do this we start from the bottom of our tree and work our way up to see if a split is valid or not. To establish validity, we use γ (gamma). If **Gain —** γ is positive then we keep the split, otherwise, we remove it. The default value of γ is 0, but for illustrative purposes, let's set our γ to 50. From previous calculations we know the **Gain** values:



Since **Gain —** γ is positive for all splits except that of *Age* < *24.5*, we can remove that branch. So the resulting tree is:

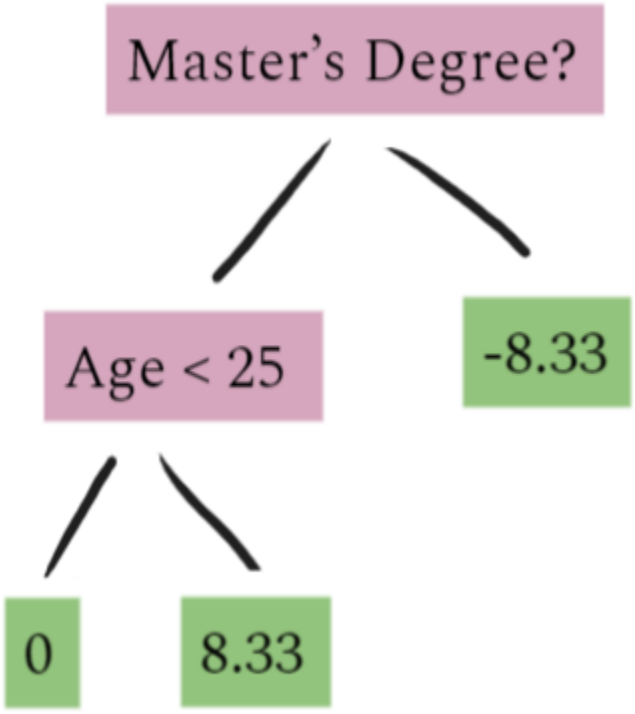


Step 4: Calculate the Output Values of Leaves

We are almost there! All we have to do now is calculate a single value in our leaf nodes because we can not have a leaf node giving us multiple outputs.

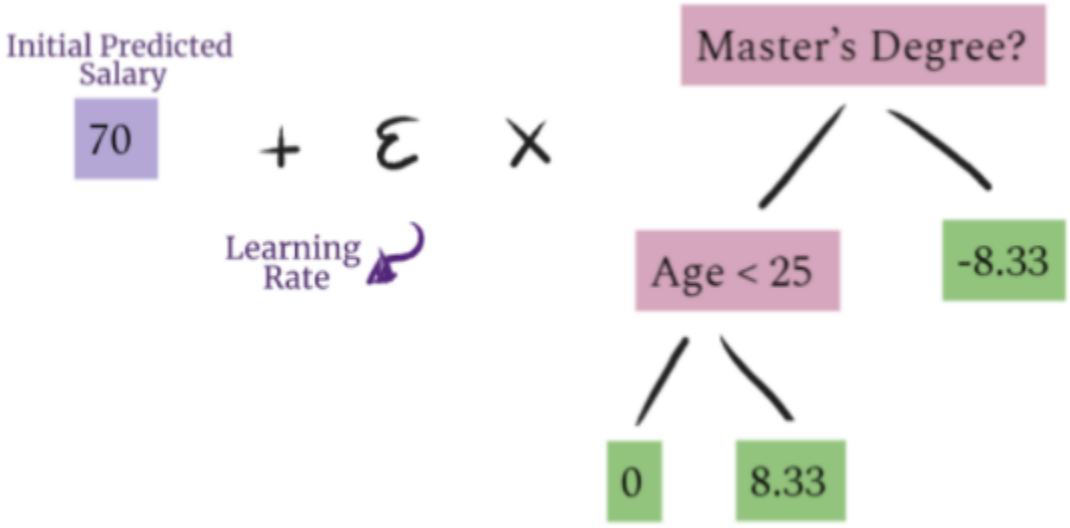
$$Output\ Value\ =\ \frac{Sum\ of\ Residuals}{Number\ of\ Residuals\ +\ \lambda}$$

This is similar to the formula to calculate **Similarity Score** except we are not squaring the *Residuals*. Using the formula and λ = 1, **drum roll** our final tree is:

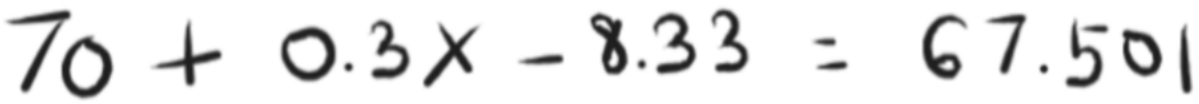


Step 5: Make New Predictions

Now that all that hard model building is behind us, we come to the exciting part and see how much our predictions improve using our new model. We can make predictions using this formula:



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



Similarly, we can calculate the rest of the predicted values:

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

Step 6: Calculate Residuals Using the New Predictions

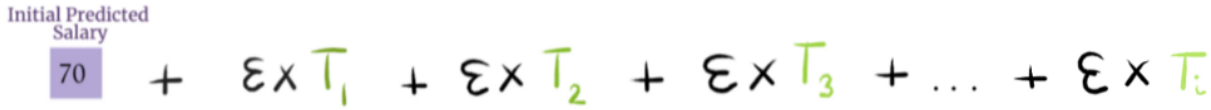
AGE	MASTER'S DEGREE	SALARY	RESIDUALS
23	No	50	-17.501
24	Yes	70	0
26	Yes	80	7.501
26	No	65	-2.501
27	Yes	85	12.501

We see that the new *Residuals* are smaller than the ones before, this indicates that we've taken a small step in the right direction. As we repeat this process, our *Residuals* will get smaller and smaller indicating that our predicted values are getting closer to the observed values.

Step 7: Repeat Steps 2–6

Now we just repeat the same process over and over again, building a new tree, making predictions, and calculating *Residuals* at each iteration. We do this until the *Residuals* are super small or we reached the maximum number of iterations we set for our algorithm. If the

tree we built at each iteration is indicated by T_i, where *i* is the current iteration, then the formula to calculate predictions is:



And that's it. Thanks for reading and good luck with the rest of your algorithmic journey!

My learning was greatly enhanced by Josh Starmer from StatQuest and this article was hugely inspired by his videos on XGBoost. For more on Machine Learning and Statistics, check out StatQuest!