**Machine Learning Algorithms(13) — Ensemble techniques (Boosting — Xgboost Regression)**

[[Kasun Dissanayake](https://kasunprageethdissanayake.medium.com/?source=post_page-----c8225311ab6d--------------------------------)](https://kasunprageethdissanayake.medium.com/?source=post_page-----c8225311ab6d--------------------------------)

[[Towards Dev](https://towardsdev.com/?source=post_page-----c8225311ab6d--------------------------------)](https://towardsdev.com/?source=post_page-----c8225311ab6d--------------------------------)

[Kasun Dissanayake](https://kasunprageethdissanayake.medium.com/?source=post_page-----c8225311ab6d--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F4a3ca0851dfe&operation=register&redirect=https%3A%2F%2Ftowardsdev.com%2Fmachine-learning-algorithms-13-ensemble-techniques-boosting-xgboost-regression-c8225311ab6d&user=Kasun+Dissanayake&userId=4a3ca0851dfe&source=post_page-4a3ca0851dfe----c8225311ab6d---------------------post_header-----------)

Published in

[Towards Dev](https://towardsdev.com/?source=post_page-----c8225311ab6d--------------------------------)

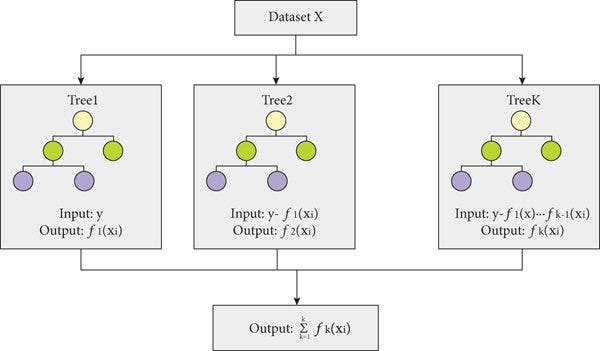
·

7 min read

·

Dec 14

206



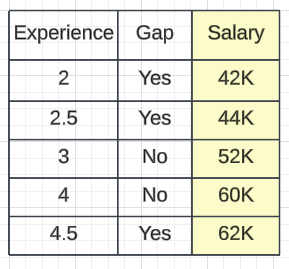
Inthis article, we delved into the **XG Boost Regressor**and explored the creation process of decision trees. We also looked at the mathematical formulas involved in XGBoost, making it an in-depth intuition into XG Boost Regression. If you missed my previous article on the **XGBoost Classifier,** you can read that article as well.

**[Machine Learning Algorithms(12) — Ensemble techniques (Boosting — Xgboost Classification)](https://towardsdev.com/machine-learning-algorithms-12-ensemble-techniques-boosting-xgboost-classification-885c06b221e5?source=post_page-----c8225311ab6d--------------------------------" \t "_blank)**

[This is the 4th article under Ensemble Techniques and if you want to learn more about Ensemble Techniques you can refer…](https://towardsdev.com/machine-learning-algorithms-12-ensemble-techniques-boosting-xgboost-classification-885c06b221e5?source=post_page-----c8225311ab6d--------------------------------" \t "_blank)

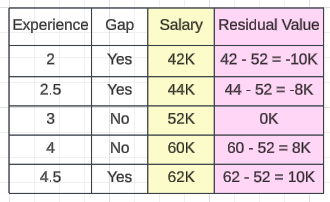
[towardsdev.com](https://towardsdev.com/machine-learning-algorithms-12-ensemble-techniques-boosting-xgboost-classification-885c06b221e5?source=post_page-----c8225311ab6d--------------------------------" \t "_blank)

Let’s start with a simple problem statement with the dataset. This problem statement has three features: **experience**, **gap(**a categorical feature based on experience), and **salary**.



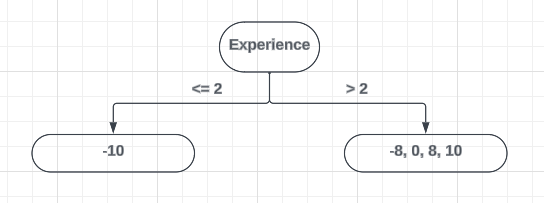
Experience Salary Dataset

To start, we must keep in mind the XGBoost creates **sequential decision trees**. To do this, we’ll begin by creating a**base model** that will provide a certain **output**. For instance, we can take the average of all salaries, which would be 42K, 44K, 52K, 60K, and 62K, resulting in an **average salary of 52K**. This will be our**residual value**, and we’ll use it to train the decision tree. we can subtract our average value from the salary and fill in the residual values for each row.



Residual Values

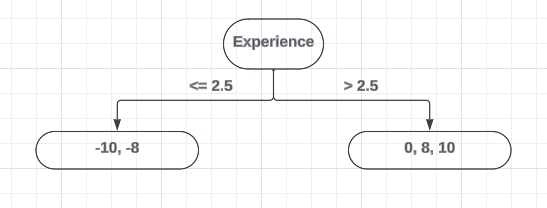
Assume the **base model as output 51K** and we have obtained the residuals which means errors. Currently, we are working on the first decision tree and we are selecting the independent feature as **experience and gap**and the output as residual values. We aim to identify the root node and determine how it can be divided based on the continuous feature and output values. As XGBoost creates binary trees, we will construct a binary tree with the condition of**≥ 2 and <2**in the experience feature. In the first record which is 2, we can put the output value for the left side of the tree and all other values to the right side of the tree.



Now we can calculate the Similarity Weight. We use the similarity weight to check whether the split is perfect split or not based on the Information Gain.

==============================================================  
Similarity Weight = Σ (Residuals)^2 / No of Residuals + λ   
λ = Hypoparameter  
==============================================================  
  
Similarity Weight of Left Node  
---------------------------------  
Similarity Weight = -10^2 / 1 + 1 = 100 /2 λ = 1  
Similarity Weight = 50  
  
  
Similarity Weight of Right Node  
---------------------------------  
 Similarity Weight = (-8 + 0 + 8 + 10)^2 / 4 + 1 λ = 1  
 = 100 / 4 + 1 = 20  
  
  
// Compute the Similarity Weight of the Root Node,  
Similarity Weight = (-10 + -8 + 0 + 8 + 10)^2 / 5 + 1 = 0  
  
// Calculate the Information Gain  
Similarity Weight of Left Node + Similarity Weight of Right Node - Similarity Weight of Root Node  
 = 50 + 20 - 0 = 70  
  
// So the total gain we will get from this split is 70

Now we will go with the next split, we will construct a binary tree with the condition of**≥ 2.5 and <2.5**in the experience feature according to the second record.

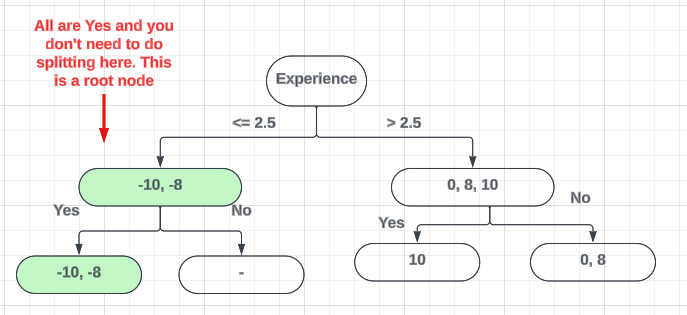


Now we can calculate the Similarity Weight.

Similarity Weight of Left Node  
---------------------------------  
Similarity Weight = (-10 + -8 )^2 / 2 + 1 = 324 /3 λ = 1  
Similarity Weight = 108  
  
Similarity Weight of Right Node  
---------------------------------  
 Similarity Weight = (0 + 8 + 10)^2 / 3 + 1 λ = 1  
 = 324 / 3 + 1 = 81  
  
// Compute the Similarity Weight of the Root Node,  
Similarity Weight = (-10 + -8 + 0 + 8 + 10)^2 / 5 + 1 = 0  
  
// Calculate the Information Gain  
Similarity Weight of Left Node + Similarity Weight of Right Node - Similarity Weight of Root Node  
 = 108 + 81 - 0 = 189  
  
// So the total gain we will get from this split is 189

Now you can see this Information Gain is better than the previous Information Gain. Similarly, you can do all the splits for other records (3, 4, and 4.5) and select the split that is given the highest Information Gain value. You can select that as the perfect split.

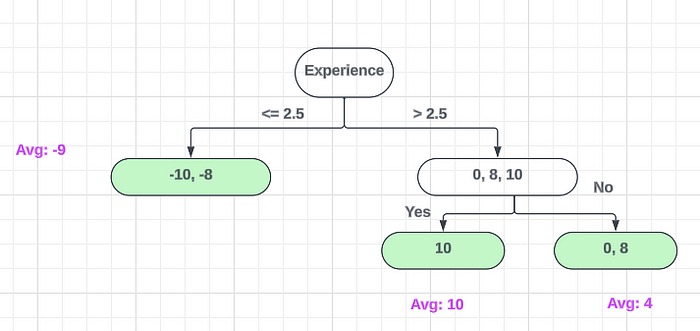
Suppose we selected 2.5 as the perfect split. Now you can go with the next category feature Gap. You will get Yes and No as the values for the category feature. You can do the splitting process.



Now let’s calculate the Similarity Weights for the next level of the tree,

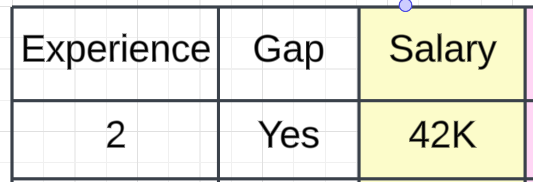
Similarity Weight of Right tree Left Node  
---------------------------------  
Similarity Weight = (10 )^2 /1 + 1 = 100 /2 λ = 1  
Similarity Weight = 50  
  
Similarity Weight of Right tree Right Node  
---------------------------------  
 Similarity Weight = (0 + 8)^2 / 2 + 1 λ = 1  
 = 64 / 3 = 21.3  
  
// Compute the Similarity Weight of the Root Node,  
Similarity Weight = (0 + 8 + 10)^2 / 3 + 1 λ = 1  
 = 324 / 3 + 1 = 81  
  
// Calculate the Information Gain  
Similarity Weight of Left Node + Similarity Weight of Right Node - Similarity Weight of Root Node  
 = 50 + 21.3 - 81 = -9.7  
  
// So the total gain we will get from this split is -9.7

Suppose this is the overall tree structure that we have created for the first tree. How will we calculate the output? It will be the average of leaf node values.



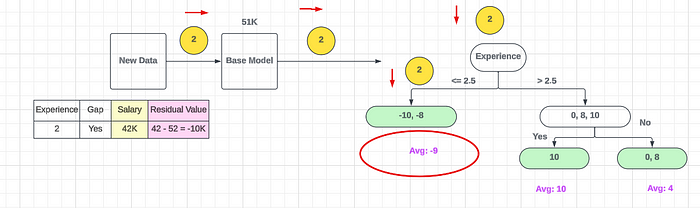
Output of the Leaf Nodes

Suppose now we will take the first record of the dataset and check what will be the output. First of all, we need to take the Base Model value. Then we can concatenate the Decision Tree output values concerning the selected row feature values.



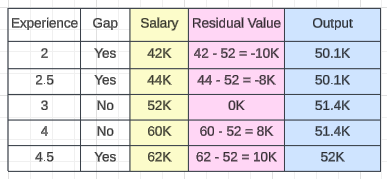
Expected Output Value = 51K + α (Output of Decision Tree 1)  
  
α = Learning Rate Parameter

So selected row Experience value is 2 and it will pass through the Decision Tree and it will go through the left side of the tree since Experience 2 is less than **2.5**. The output will be **-9.**



Expected Output Value = 51K + α (Output of Decision Tree 1)  
 = 51K + 0.1 (-9)  
 = 51K + - 0.9 = 50.1 K

This means that once we pass Experience 2 as our independent feature, our real output now is **50.1K**. You can do this for all the datasets and our output will be like this.

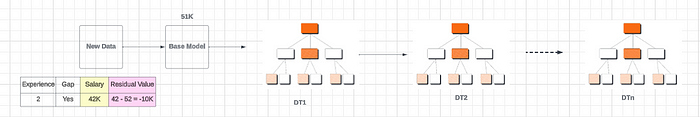


Output Values

Now we can calculate the new Residual value using new output values and Average values.

Residual Value 2 = Output - Average  
   
Average = (50.1 + 50.1 + 51.4 + 51.4 + 52) / 5 = 51K

Now in the next step, we will be creating another **Decision Tree** using **Experience and Gap as the input features and Residual Value 2 as the output feature**. This will be our second decision tree. Likewise, we can create multiple decision trees.



Now our complete XgBoost output will be like this,

Output = Base Model Value + α1 (Decision Tree 1 output) + α2 (Decision Tree 2 output) + ....... + αn (Decision Tree n output)

Also, we can use **γ** value for **post pruning** purposes. If we subtract the tree's Information Gain value from this is **γ** value and the result is a negative value we can cut the tree from that **root node and stop splitting** only when the tree is trying to get **overfitting**. In most of the scenarios, this **γ** value is set as a **default value**. You can check and see them in some libraries. **XgBoost is a kind of black box model**, you can not visualize all the processes inside this algorithm. The models are not quite as easy to interpret as a decision tree, or a set of manually coded rules. But XgBoost is not entirely a black box. There is pretty good interpretability in terms of how the learning rate/shrinkage works. The difficulty in interpretation comes along with the use of trees.

This is all about the **XgBoost Regressor**and I hope you get a perfect understanding of this algorithm**. See**you in another Machine Learning algorithm tutorial.