

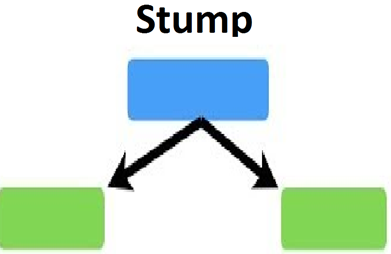
**ADA Boost(Adaptive Boost):**

**Consider the classification algorithm**

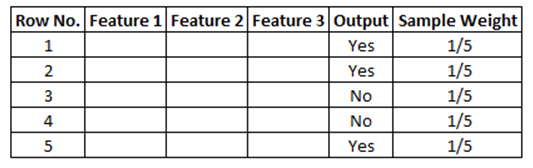
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **outlook** | **temperature** | **humidity** | **windy** | **O/P** | **Sample Weight** |
| 1 | sunny | hot | high | FALSE | Yes | 1/10 |
| 2 | sunny | hot | high | TRUE | No | 1/10 |
| 3 | overcast | hot | high | FALSE | No | 1/10 |
| 4 | rainy | mild | high | FALSE | No | 1/10 |
| 5 | rainy | cool | normal | FALSE | Yes | 1/10 |
| 6 | rainy | cool | normal | TRUE | No | 1/10 |
| 7 | overcast | cool | normal | TRUE | Yes | 1/10 |
| 8 | sunny | mild | high | FALSE | Yes | 1/10 |
| 9 | sunny | cool | normal | FALSE | Yes | 1/10 |
| 10 | rainy | mild | normal | FALSE | No | 1/10 |

Based on the features **entropy or GINI**, which should be the **Stump** as a first base learner.

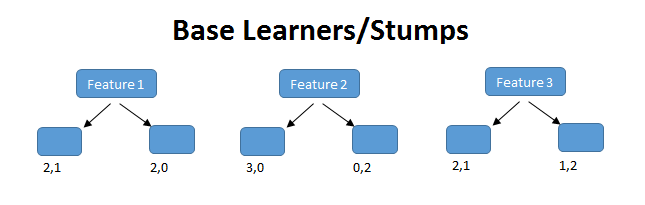
**Step 1**: Decide the which is your first base learner



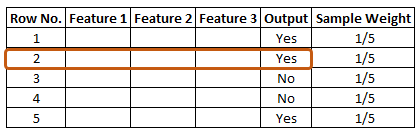
**Step 2**: Assign the sample weights for all the records



**Step 3**: Create the First Base Learner

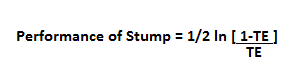


**Step 4:** Based on the first decision tree, calculate the **Total Error(TE)**

****

**The Total Error(TE)**is the sum of all the errors in the classified record for sample weights. In our case, there is only 1 error, so **Total Error (TE) = 1/5**

**Step 5:** Calculate the **Performance of Stump:**



**The performance of Stump as 0.693.**

**Step 5:** Updating Weights

For incorrectly classified records the formula is:

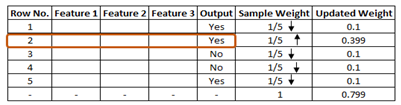
**New Sample Weight = Sample Weight \* e^(Performance)**

In our case Sample weight = 1/5 so, **1/5 \* e^ (0.693) = 0.399**

And for correctly classified records, we use the same formula with a negative sign with performance, so that the weight for correctly classified records will reduce compared to the incorrect classified ones. The formula is:

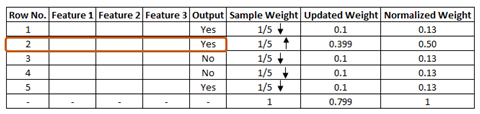
**New Sample Weight = Sample Weight \* e^- (Performance)**

Putting the values, **1/5 \* e^-(0.693) = 0.100**



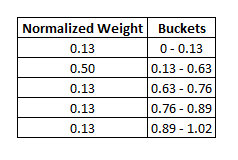
The total sum of all the weights should be 1. But in this case, one can see that the total updated weight of all the records is not 1, it’s 0.799. To make the total sum 1, one must divide every updated weight by the total sum of updated weight. For example, if our updated weight is 0.399 and we divide this by 0.799, i.e. **0.399/0.799=0.50**.

**0.50** can be known as the normalized weight. In the below figure, we can see all the normalized weight and their sum is approximately 1.



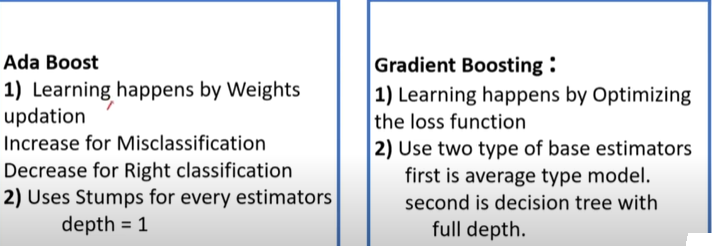
## ****Step 6:** Creating New Dataset**

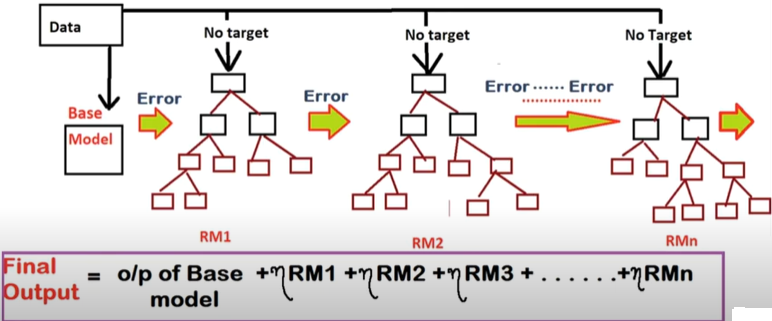
## create a new dataset from our previous one. In the new dataset, the frequency of incorrectly classified records will be more than the correct ones. While considering these normalized weights, we have to create a new dataset and that dataset is based on normalized weights. It will probably select the wrong records for training purposes. That will be the second decision tree/stump. To make a new dataset based on normalized weight, the algorithm will divide it into buckets.



So, our first bucket is from **0 – 0.13,** second will be from **0.13 – 0.63(0.13+0.50),** third will be from **0.63 – 0.76(0.63+0.13),** and so on. After this the algorithm will run 5 iterations to select different-different records from the older dataset. Suppose, in 1st iteration, the algorithm will take a random value **0.46,** then it will go and see in which bucket that value falls and selects that records in the new dataset, then again it will select a random value and see in which bucket it is and select that record for the new dataset and the same process is repeated for 5 times.

**Gradient Boosting:**





**Steps in Gradient Boosting:**

1. Create a base model, Average model or most frequent category.
2. Calculate the residuals from average prediction and actual values.
3. Create another model RM1(residual Model 1) which will take residuals as target.
4. New predicted residual value, now we will calculate new predicted target value.
5. Now we have residuals again (actual – predicted) and new model RM2 will fit again on the residuals as target and will predict new residuals.

**Example Gradient Boosting:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Age** | **Sq.ft** | **Location** | **Price** |
| **5** | **1500** | **5** | **480** |
| **11** | **2030** | **4** | **1090** |
| **14** | **1442** | **7** | **350** |
| **8** | **2501** | **9** | **1310** |
| **12** | **1300** | **2** | **400** |
| **10** | **1789** | **11** | **500** |

**Step 1: Calculate the Average of the target label**

**Average = (480+1090+350+1310+400+500) / 6 = 688**

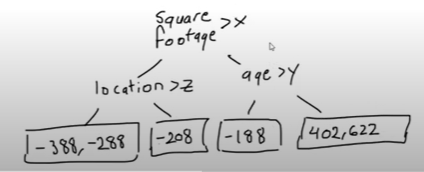
**Step 2: Calculate the Residuals (Error)**

**Formula = Actual-Predicted**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Age** | **Sq.ft** | **Location** | **Price** | **Predicted** | **Residual** |
| **5** | **1500** | **5** | **480** | **688** | **-208** |
| **11** | **2030** | **4** | **1090** | **688** | **402** |
| **14** | **1442** | **7** | **350** | **688** | **-338** |
| **8** | **2501** | **9** | **1310** | **688** | **622** |
| **12** | **1300** | **2** | **400** | **688** | **-288** |
| **10** | **1789** | **11** | **500** | **688** | **-188** |

**Step 3: Construct a Decision Tree**

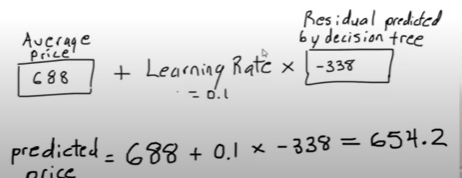
Build a tree with the goal of predicting residuals. In other words, every leaf will contain a prediction as to the value of the residual.



In the event there are more residuals than leaves, some residuals will end up inside the same leaf, when this happens, we compute their average and place that inside the leaf.

**Step 4: Predict the target label using all of the trees within the ensemble**

Each sample passes through the decision nodes of the newly formed tree until it reaches a given lead. The residuals in said leaf is used to predict the house price.



**Step 5: Compute the new residuals**

We calculate a new set of residuals by subtracting the actual house prices from the predictions made in the previous step. The residuals will then be used for the leaves of the next decision tree as described in step 3.

Residual = Actual Price – Predicted Price

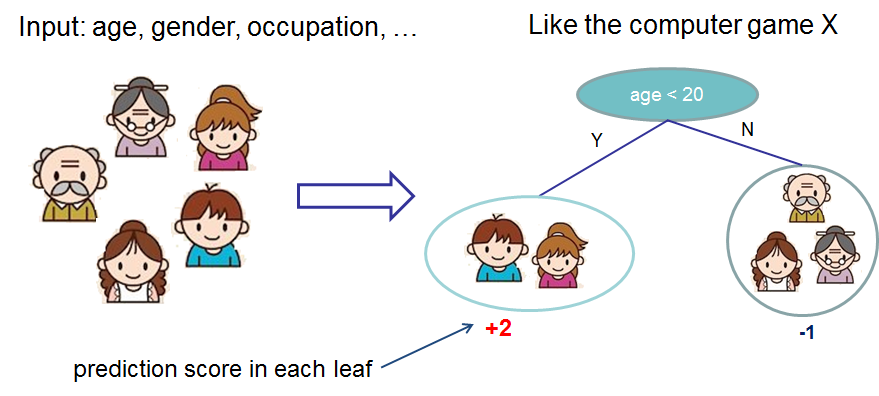
= 350-654.2=-304.2

|  |  |
| --- | --- |
| residual | New Residuals |
| -208 | -187.2 |
| 402 | 350.8 |
| -338 | -304.2 |
| 622 | 570.8 |
| -288 | -259.2 |
| -188 | -169.2 |

**Step 6:** Repeat step 3 to step 6 until the number of iterations matches the number specified by the hyperparameter (i.e. number of estimators)

**XG Boost:**

Now that we have introduced the elements of supervised learning, let us get started with real trees. To begin with, let us first learn about the model choice of XGBoost: **decision tree ensembles**. The tree ensemble model consists of a set of classification and regression trees (CART). Here’s a simple example of a CART that classifies whether someone will like a hypothetical computer game X.



We classify the members of a family into different leaves, and assign them the score on the corresponding leaf. A CART is a bit different from decision trees, in which the leaf only contains decision values. In CART, a real score is associated with each of the leavess, which gives us richer interpretations that go beyond classification. This also allows for a principled, unified approach to optimization, as we will see in a later part of this tutorial.

Usually, a single tree is not strong enough to be used in practice. What is actually used is the ensemble model, which sums the prediction of multiple trees together.

