Math Behind AdaBoost Algorithm in 3 steps…

Sampath · Follow
8 min read · May 1, 2020

Listen Share More

Hi guys, whenever we are participating in DataScience Hackthons the first algorithm that comes into our mind is ***Boosting*** which improves the accuracy of our model. But unfortunately, we don't know the math behind these Boosting algorithms.

In this article, We are going to master the math behind the boosting algorithms in a simple manner. They are different types of boosting algorithms:

1. **AdaBoost** (Adaptive Boosting)

2. **Gradient Boosting**

3. **XGBoost**

In this article, we will focus on AdaBoost and will focus on Gradient Boosting and XGboost in an upcoming article.

I am assuming that you have a basic understanding of how a Decision Tree works. If you're not sure of your understanding I would request you to go through Decision Tree Algorithm before you read on

Terminology related to boosting algorithms:

Most of the blogs or books used term is ***"Weak Learner".*** Technically Weak Learners are called as ***stumps.***

A tree with just one node and two leaves is called a ***stump.*** As shown in Fig.1

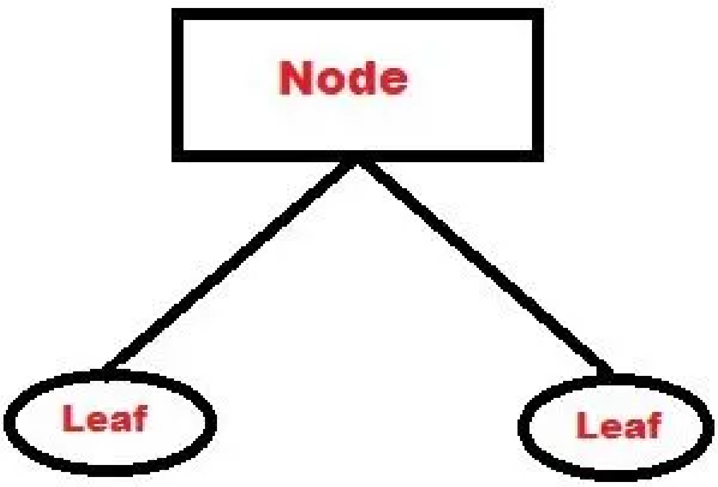


Fig 1

Ada Boost:

AdaBoost is best used to boost the performance of decision trees on binary classification problems. It is best used with weak learners. Let me explain the 3 ideas behind Ada Boost:-

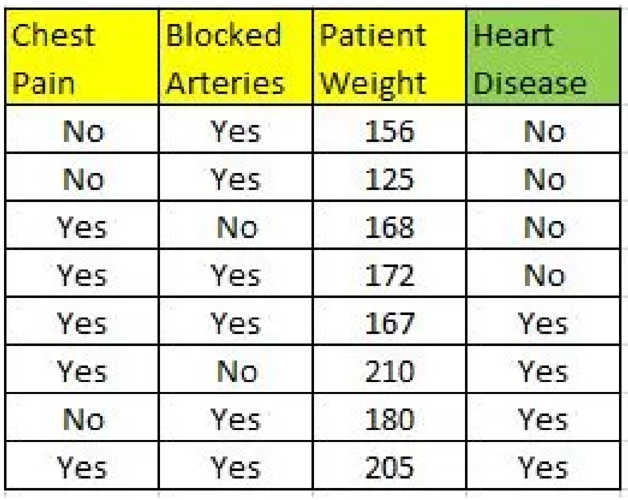
1. AdaBoost combines a lot of "**weak learners**"(stumps) to make classifications.

2. Some stumps get more say (information) in the classification than others.

3. Each stump is made by taking the previous stump's mistakes into account

Now we go through the math behind Ada Boost then you can understand the above 3 points for sure.

We considered a simple dataset to understand the concept clearly. Dataset is shown below Fig.2



Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No
Yes	Yes	167	Yes
Yes	No	210	Yes
No	Yes	180	Yes
Yes	Yes	205	Yes

Fig.2

The first thing we do is assign a weight to each and every sample(data point) that indicates how important it is to be correctly classified. Now we create a new column with the name "**Sample Weight**".

Note:- The "Sample Weight" is different from "Patient Weight".

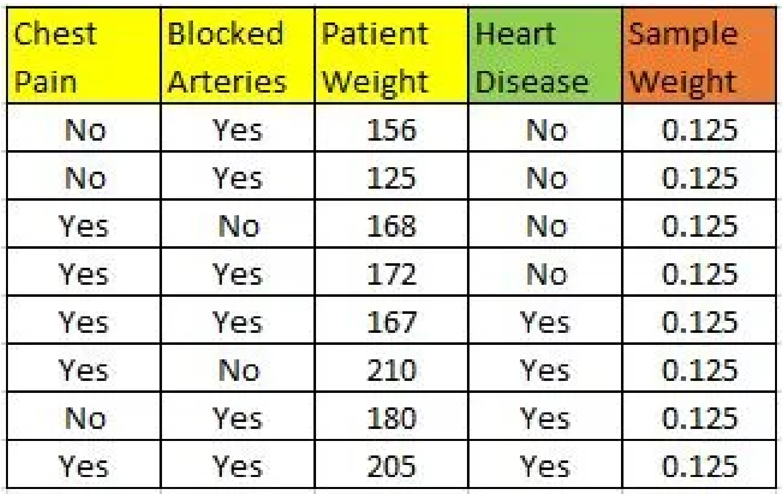
At the start, all samples get the same weight and that makes all samples equally important. The formula to calculate Sample Weight is

$$\frac{1}{Total\ No\ of\ Samples}$$

Here we have 8 observations in our sample dataset, So our sample weight is

Search

After adding the “***Sample Weight***” column, now dataset looks like as shown below:-



Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
No	Yes	156	No	0.125
No	Yes	125	No	0.125
Yes	No	168	No	0.125
Yes	Yes	172	No	0.125
Yes	Yes	167	Yes	0.125
Yes	No	210	Yes	0.125
No	Yes	180	Yes	0.125
Yes	Yes	205	Yes	0.125

Fig.3

However, after we make the first stump, these weights will change in order to guide, how the next stump is created.

Initially, we will ignore the “***Sample Weight***” column because all the weights are the same. Now we need to make the first stump.

We start by seeing how well “***Chest Pain***” classifies the samples and will see how the variables(Blocked Arteries, Patient Weight) classifies the samples

Chest Pain:-

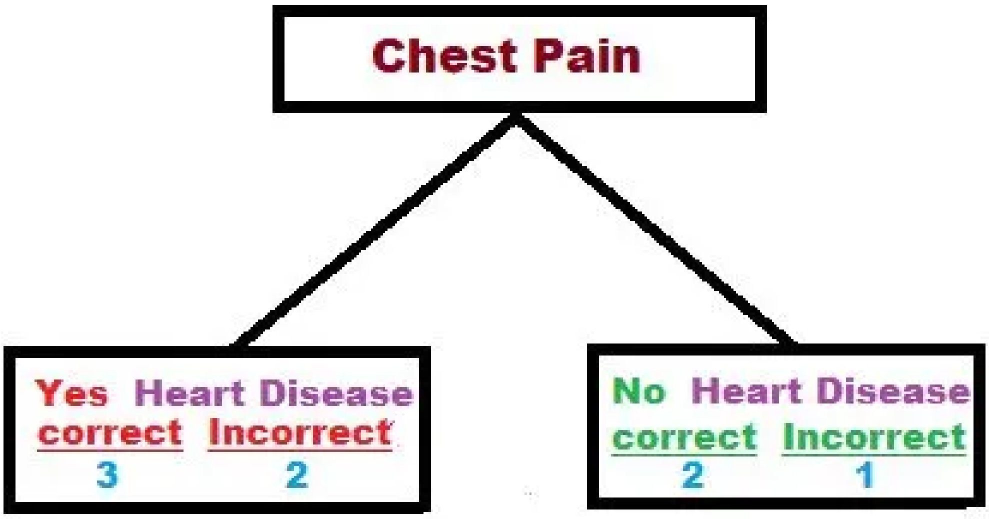


Fig.4

Of the 5 samples with “**Heart Disease**”,3 were correctly classified as having Heart Disease and 2 were incorrectly classified.

Of the 3 samples “**without Heart Disease**”, 2 were correctly classified as not Heart Disease, and 1 was incorrectly classified.

Blocked Arteries:-

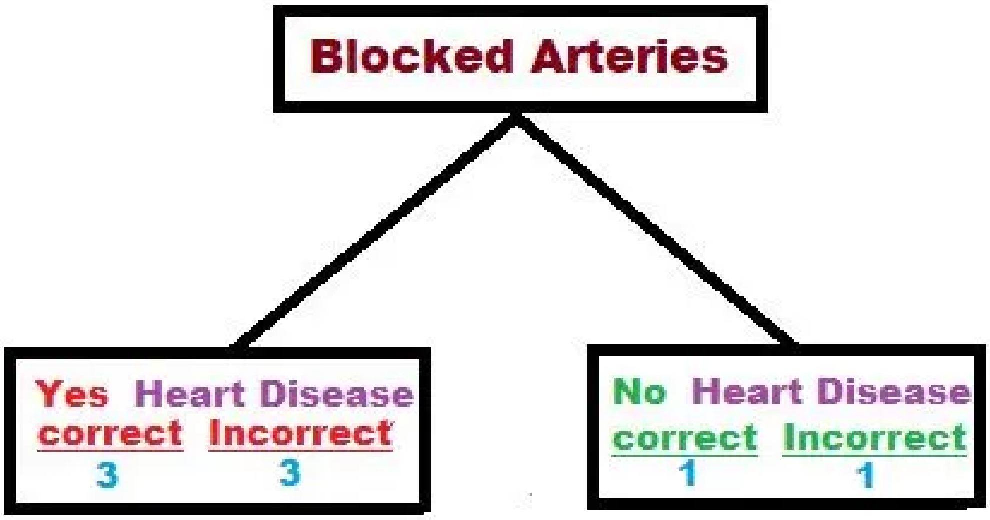


Fig.5

Of the 6 samples with "**Heart Disease**",3 were correctly classified as having Heart Disease and 3 were incorrectly classified.

Of the 2 samples "**without Heart Disease**", 1was correctly classified as not Heart Disease, and 1 was incorrectly classified.

Patient Weight:-

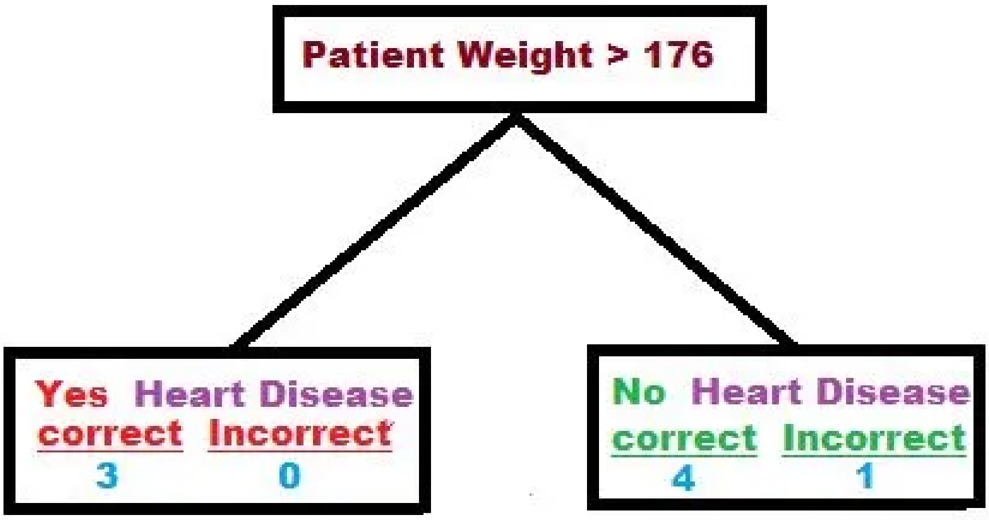


Fig.6

Of the 3 samples with "**Heart Disease**",3 were correctly classified as having Heart Disease, and 0 was incorrectly classified.

Of the 5 samples "**without Heart Disease**", 4 were correctly classified as not Heart Disease, and 1 was incorrectly classified.

Now will calculate the Gini Index for these three stumps

Steps to calculate the Gini Index for a split:-

1. Calculate the Gini Index for *sub-nodes,* using formula sum of squares of probability for success(In our case "*Correct*") and failure(In our case "*Incorrect*") ($p^2 + q^2$)

2. Calculate the Gini Index for a ***split,*** using (1- weighted Gini score of each node of that split).

Now, I will explain with an example for better understanding. We start with "Chest Pain" refer to fig.4

1. Calculate Gini for subnode "***Yes***" = ((3/5)*(3/5))+((2/5)*(2/5))=**0.52.** Gini for subnode "***No***" = ((2/3)*(2/3))+((1/3)*(1/3)) = **0.556**
2. Calculate Weighted Gini Index for split "**Chest Pain**" =1-((5/8)*0.52+(3/8)*0.556) = 0.47

In the same way, we calculated for "***Blood Arteries***" and "***Patient Weight"*** the Gini Index for ***Blood Arteries*** is 0.5 and ***Patient Weight*** is 0.2

Chest Pain ***0.47***

Blocked Arteries ***0.5***

Patient Weight ***0.2***

The Gini Index for "***Patient Weight***" is the lowest, so this would be the first stump.

Now we need to determine how much "**Amount of Say**"(Information) of this stump will have in the final classification.

Formula to calculate the "**Amount of say**" is

$$\frac{1}{2}\log\frac{1 - Total\ Error}{Total\ Error}$$

First, we need to calculate "***Total Error***" for stumps. The Total Error for a stump is the **sum of the weights associated with the incorrectly classified samples.**

Total Error *for* ***Chest Pain***:- It made 3 errors i.e 1/8+1/8+1/8=3/8.

Total Error *for* ***Blood Arteries***:- It made 4 errors i.e 1/8+1/8+1/8+1/8 = 4/8.

Total Error *for* ***Patient Weight***:- It made 1 error i.e 1/8.

Note:- Because all the sample's weight is added up to 1, Total Error will always be between 0 and 1.

0 indicates perfect stump, 1 indicates horrible stump.

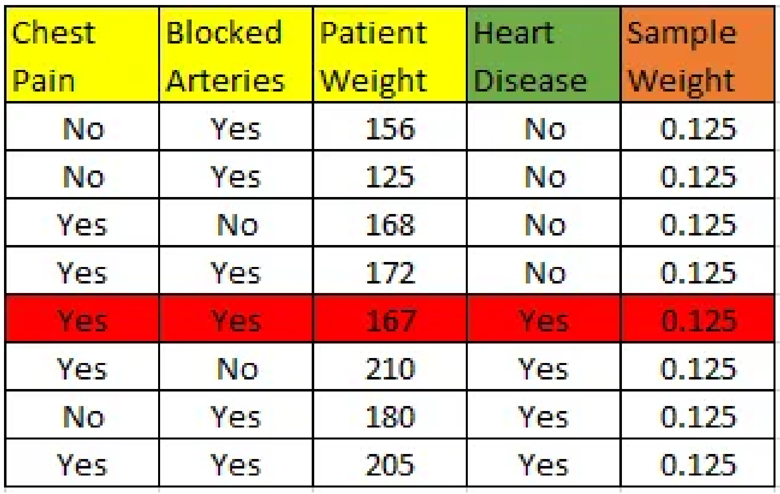
Now, the **Amount of say** for "***Patient Weight***" is **0.97** [1/2 log(7)].

Guys, whenever you attend webinar/lecture on boosting there is one point i.e Boosting will increase weight for misclassified samples in the follow-up tree but, how we don't know. Now it's time to learn how to update the weights for samples.

Basically boosting will increase sample weight for incorrectly classified samples and decrease sample weight for correctly classified samples. The formula is below for ***updating weights of incorrectly classified samples***:-

$$\text{New Sample Weight} = \text{Sample Weight} * e^{-Amount\ of\ Say}$$

There is one misclassified sample, Here ***sample weight*** of that sample is ***0.125*** and the ***amount of say*** of ***Patient Weight*** is ***0.97***



Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
No	Yes	156	No	0.125
No	Yes	125	No	0.125
Yes	No	168	No	0.125
Yes	Yes	172	No	0.125
Yes	Yes	167	Yes	0.125
Yes	No	210	Yes	0.125
No	Yes	180	Yes	0.125
Yes	Yes	205	Yes	0.125

New Sample Weight = 0.125* e^0.97 = 0.125 * 2.67 = 0.33. The New Sample Weight for a misclassified sample is 0.33, which is more than the old weight.

The formula is below for ***updating weights of correctly classified samples:-***

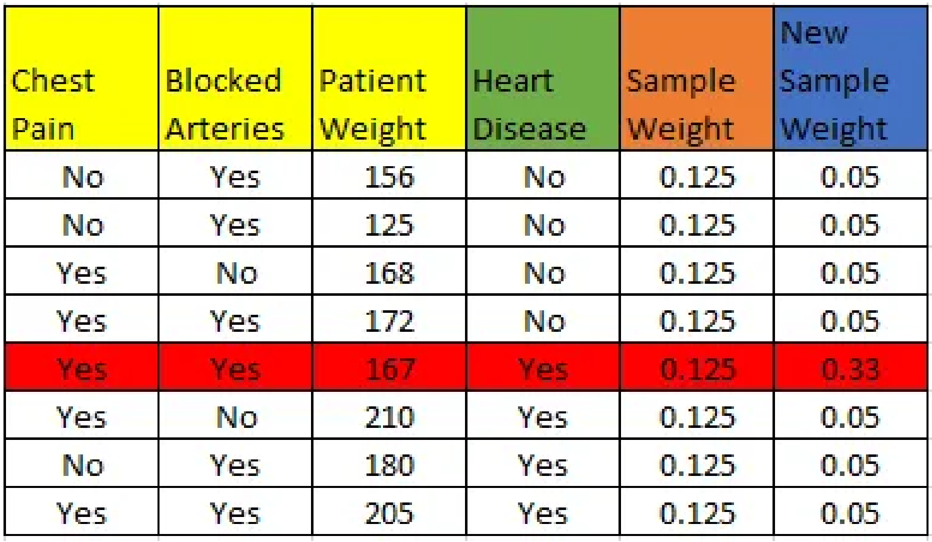
$$\text{New Sample Weight} = \text{Sample Weight} * e^{Amount\ of\ Say}$$

There are seven correctly classified samples, Here ***sample weight*** of that samples is ***1/8*** and the ***amount of say*** of ***Patient Weight*** is ***0.97***

New Sample Weight = 0.125*e^-0.97 = 0.125*0.38 = 0.05. The New Sample Weight for correctly classified samples is 0.05, which is less than the old weight.

Note:- Here in the above scenario the "Sample weight" is the same for all 7 samples, so we are calculating "New Sample Weight" only once. If "Sample Weight" is different then we need to calculate "New Sample Weight" for each and every sample individually.

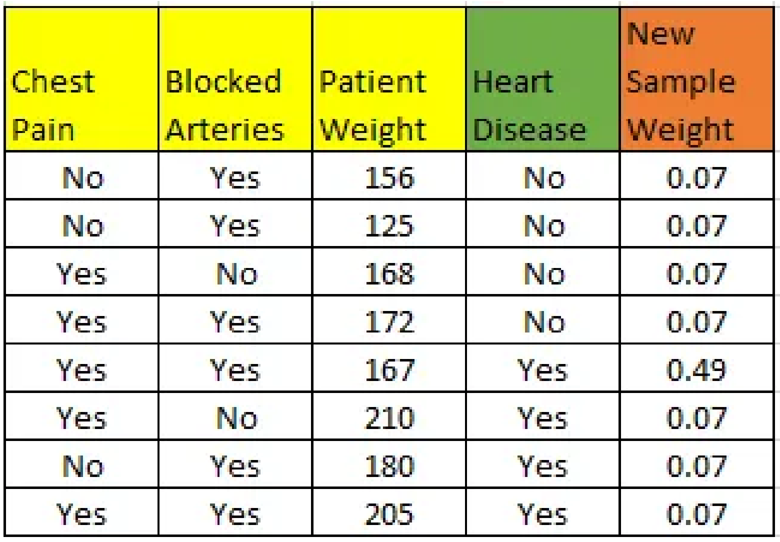
After adding the *New Sample Weight* Column to our Dataset…



Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	New Sample Weight
No	Yes	156	No	0.125	0.05
No	Yes	125	No	0.125	0.05
Yes	No	168	No	0.125	0.05
Yes	Yes	172	No	0.125	0.05
Yes	Yes	167	Yes	0.125	0.33
Yes	No	210	Yes	0.125	0.05
No	Yes	180	Yes	0.125	0.05
Yes	Yes	205	Yes	0.125	0.05

Guys if you observe, the New Sample Weight of a misclassified sample is increased from 0.125 to 0.33 and correctly classified samples weight is decreased from 0.125 to 0.05.

Right now, if you add up the ***New Sample Weights***, you get 0.68. So we divide each sample weight with 0.68 to get the normalized values. Now we consider Normalized Weights as New Sample Weights as shown below.



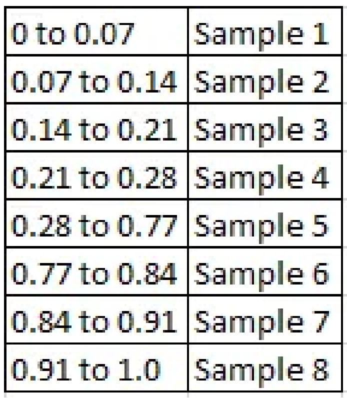
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	New Sample Weight
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07
Yes	Yes	167	Yes	0.49
Yes	No	210	Yes	0.07
No	Yes	180	Yes	0.07
Yes	Yes	205	Yes	0.07

Before building the next stump, we need to create a new dataset.

So we start by making a new but empty dataset that is the same as the original dataset size, Here we pick a random number between 0 and

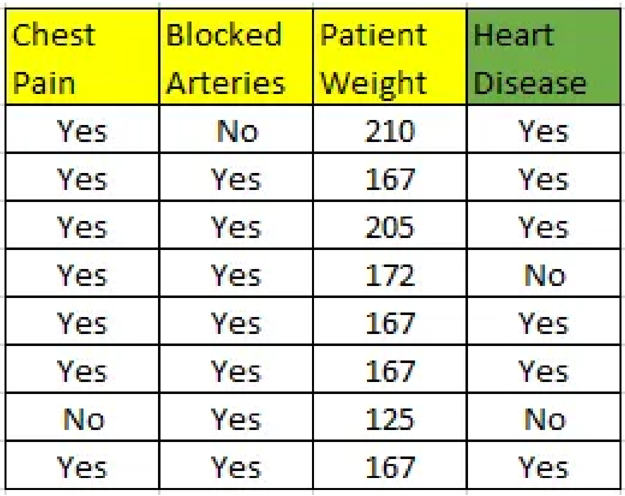
1 and we see where the number falls, here we use sample weights like a distribution. Guys, hold on let me explain with an example for better understanding.

Now if we get a random number lies between 0 to 0.07 then we would put the first sample(data point), 0.071 to 0.14 then we would put the second sample,….



0 to 0.07	Sample 1
0.07 to 0.14	Sample 2
0.14 to 0.21	Sample 3
0.21 to 0.28	Sample 4
0.28 to 0.77	Sample 5
0.77 to 0.84	Sample 6
0.84 to 0.91	Sample 7
0.91 to 1.0	Sample 8

When I pick a random number eight times between 0 and 1 then I got the random numbers are 0.78,0.56,0.94,0.24,0.68,0.32,0.13,0.73. Now our new dataset is created based on these numbers. The new dataset is shown below:



Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	No	210	Yes
Yes	Yes	167	Yes
Yes	Yes	205	Yes
Yes	Yes	172	No
Yes	Yes	167	Yes
Yes	Yes	167	Yes
No	Yes	125	No
Yes	Yes	167	Yes

New Dataset

Ultimately the wrongly classified sample was added to the new collection of samples(dataset) 4 times, reflecting its larger sample weight.

From now will use the new collection of samples as a dataset and repeat the same procedure as above i.e

1. Assign equal weights to all samples
2. Find the stump that does the best job classifying the new collection of samples.
3. Calculate "**Total Error** and **Amount of say**" to calculate ***New Sample Weight***
4. Normalize the New Sample Weights
5. Repeat above 4 steps until all the samples correctly classified.

So that is how the errors that the first tree makes influence how the second tree is made… and the errors that the second tree makes influence how the third tree is made, … and so on

Finally, Now we need to talk about how a forest of stumps created by AdaBoost makes classification.

Imagine there are 6 stumps are created by the AdaBoost algorithm. Out of 6 stumps, 4 stumps are classified patient ***has Heart Disease,*** and the other 2 stumps classified patient ***does not have Heart Disease.***

These are the ***Amount of Say*** for these stumps are 0.97+0.32+0.78+0.63 = ***2.7,*** and the ***Amount of Say*** of the other 2 stumps are 0.41+0.82=***1.23***.

Ultimately, the patient is classified as ***Has Heart Disease*** because of the larger **Amount of Say(2.7)**.

In this article, we looked at AdaBoost, one of the methods of ensemble modeling to enhance the prediction power. Here, we have discussed the math behind AdaBoost.

Guys, if you have any queries don't hesitate to comment…

Adaboost Machine Learning Data Science Classification Boosting

Follow

Written by Sampath

5 Followers