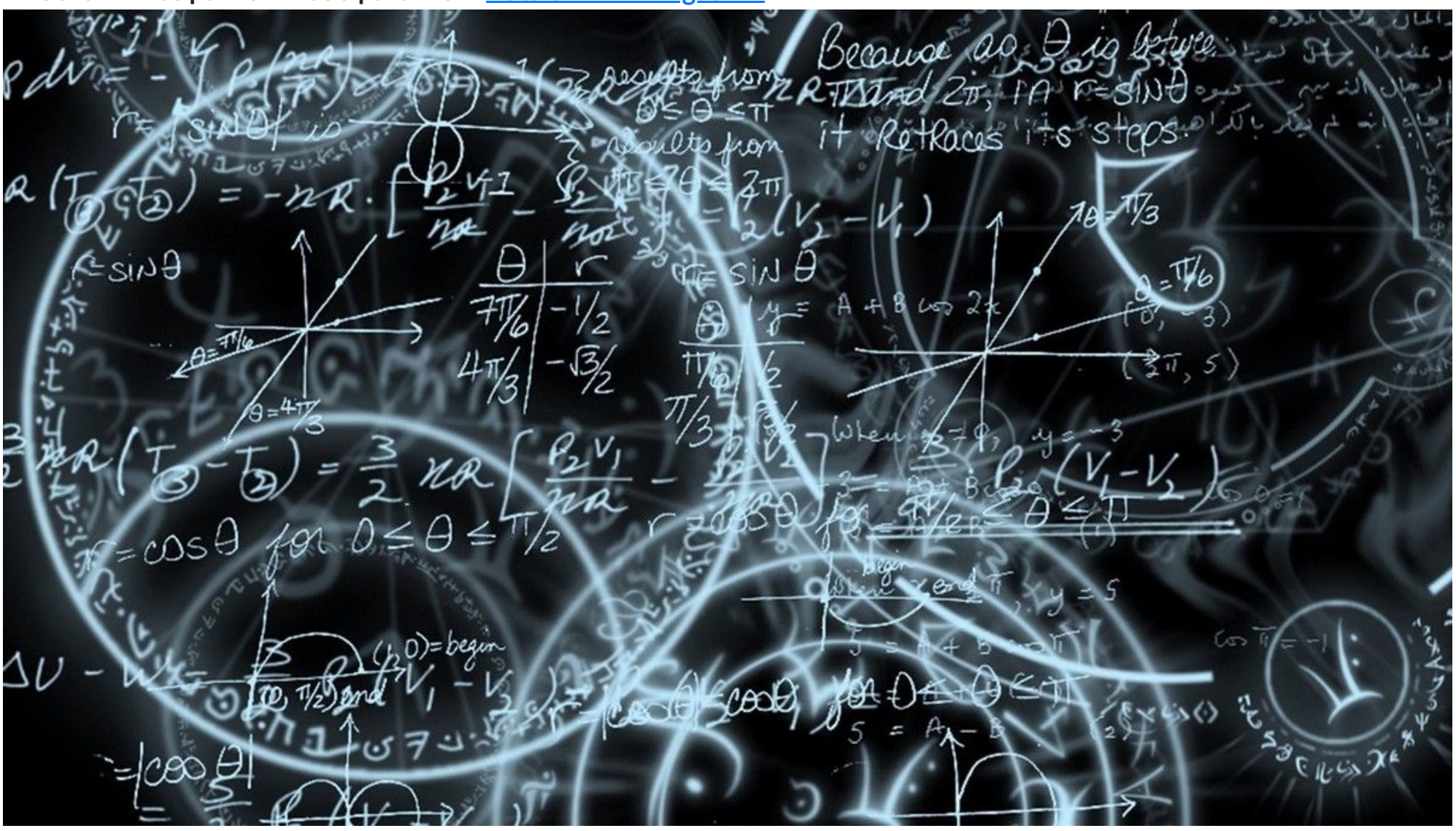
AdaBoost Algorithm – A Complete Guide for Beginners

Anshul Saini — Published On September 15, 2021 and Last Modified On September 15th, 2021

Algorithm Beginner Machine Learning Technique

This article was published as a part of the Data Science Blogathon



Introduction

Boosting is an ensemble modelling technique that was first presented by Freund and Schapire in the year 1997, since then, Boosting has been a prevalent technique for tackling binary classification problems. These algorithms improve the prediction power by **converting a number of weak learners to strong learners.**

The principle behind boosting algorithms is first we built a model on the training dataset, then a second model is built to rectify the errors present in the first model. This procedure is continued until and unless the errors are minimized, and the dataset is predicted correctly.

Let's take an example to understand this, suppose you built a decision tree algorithm on the Titanic dataset and from there you get an accuracy of 80%. After this, you apply a different algorithm and check the accuracy and it comes out to be 75% for KNN and 70% for Linear Regression.

We see the accuracy differs when we built a different model on the same dataset. But what if we use combinations of all these algorithms for making the final prediction? We'll get more accurate results by taking the average of results from these models. We can increase the prediction power in this way.

Boosting algorithms works in a similar way, it combines multiple models (weak learners) to reach the final output (strong learners).

In this article, we will understand the math behind different types of boosting algorithms.

There are mainly 3 types of boosting algorithms:

- AdaBoost algorithm
- Gradient descent algorithm
- Xtreme gradient descent algorithm

Here I will be focusing on the AdaBoost algorithm. Gradient Descent and XGboost will be covered in upcoming articles.

What is the AdaBoost Algorithm?

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called **Decision Stumps.**

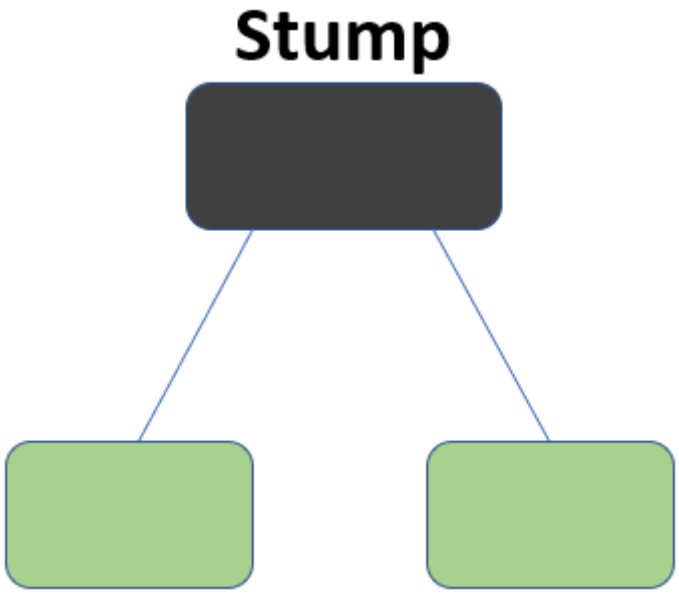


Image Source: Author

What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lowe error is received.

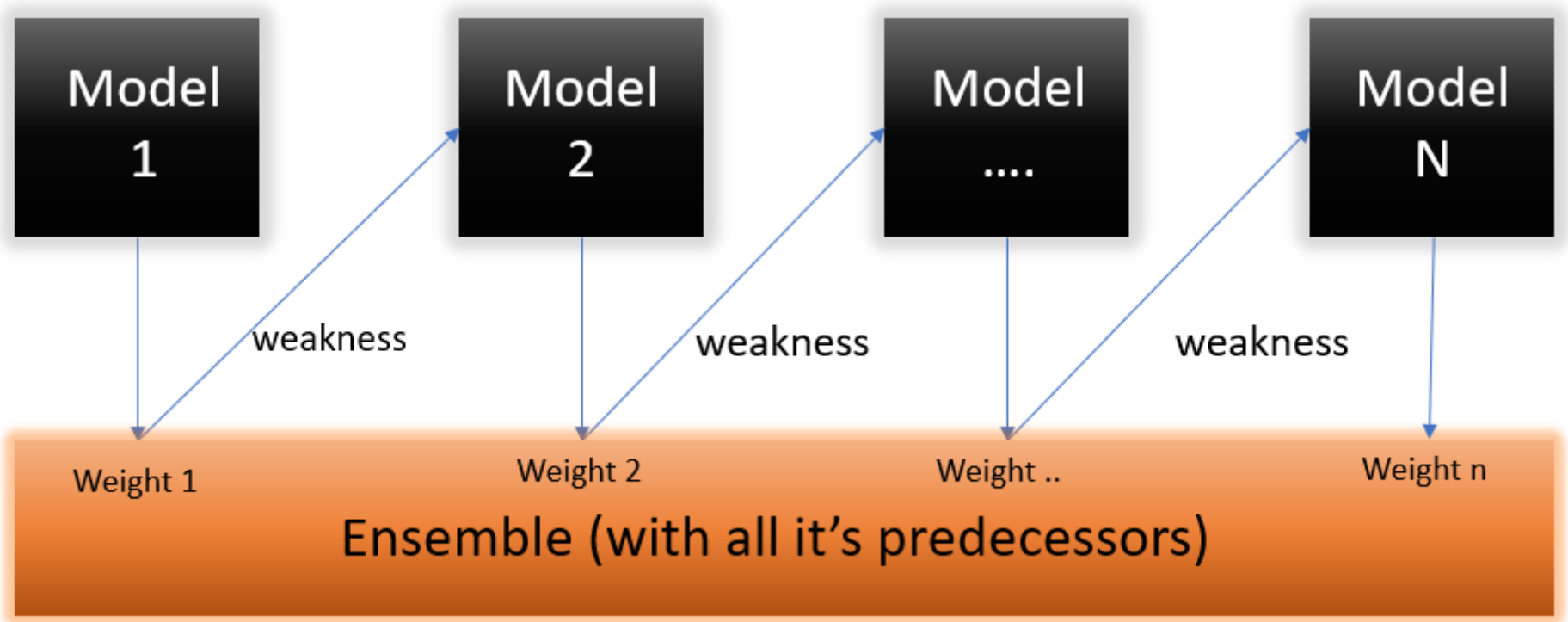


Image Source: Author

If you want to understand this same paragraph visually, I strongly recommend you go through this article.

Here we will be more focused on mathematics intuition.

Understanding the working of AdaBoost Algorithm

Let's start understanding what and how does this algorithm work under the hood.

Step 1 – The Image is shown below is the actual representation of our dataset. Since the target column is binary it is a classification problem. First of all these data points will be assigned some weights. Initially, all the weights will be equal.

Row No.	Gender	Age	Income	Illness	Sample Weights
1	Male	41	40000	Yes	1/5
2	Male	54	30000	No	1/5
3	Female	42	25000	No	1/5
4	Female	40	60000	Yes	1/5
5	Male	46	50000	Yes	1/5

Image Source: Author

The formula to calculate the sample weights is:

$$w(x_i, y_i) = \frac{1}{N}, \; i = 1, 2, \ldots . n$$

Where N is the total number of datapoints

Here since we have 5 data points so the sample weights assigned will be 1/5.

Step 2 – We start by seeing how well "*Gender*" classifies the samples and will see how the variables (Age, Income) classifies the samples.

We'll create a decision stump for each of the features and then calculate the ***Gini Index*** of each tree. The tree with the lowest Gini Index will be our first stump.

Here in our dataset let's say ***Gender*** has the lowest gini index so it will be our first stump.

Step 3 – We'll now calculate the **"Amount of Say"** or **"Importance"** or **"Influence"** for this classifier in classifying the datapoints using this formula:

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

The total error is nothing, but the summation of all the sample weights of misclassified data points.

Here in our dataset let's assume there is 1 wrong output, so our total error will be 1/5, and alpha(performance of the stump) will be:

$$Performance\ of\ the\ stump\ =\ \frac{1}{2}\log_e(\frac{1-Total\ Error}{Total\ Error})$$

$$\alpha\ =\ \frac{1}{2}\log_e\left(\frac{1-\frac{1}{5}}{\frac{1}{5}}\right)$$

$$\alpha\ =\ \frac{1}{2}\log_e\left(\frac{0.8}{0.2}\right)$$

$$\alpha\ =\ \frac{1}{2}\log_e(4)\ =\ \frac{1}{2}*(1.38)$$

$$\alpha\ =\ 0.69$$

Note: Total error will always be between 0 and 1.

0 Indicates perfect stump and 1 indicates horrible stump.

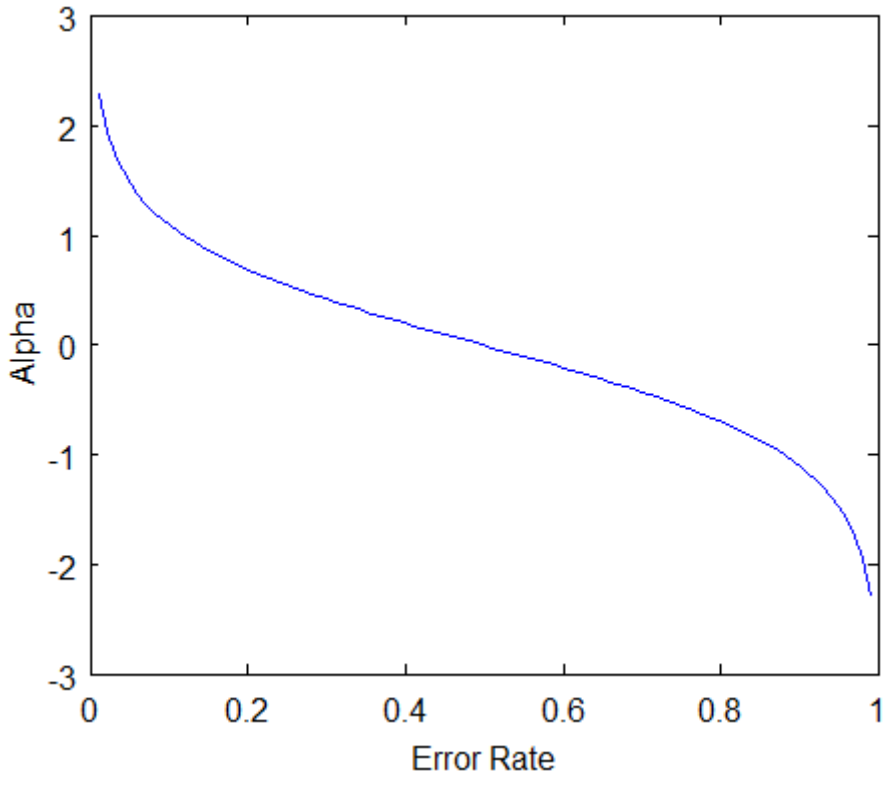


Image1

From the graph above we can see that when there is no misclassification then we have no error (Total Error = 0), so the "amount of say (alpha)" will be a large number.

When the classifier predicts half right and half wrong then the Total Error = 0.5 and the importance (amount of say) of the classifier will be 0.

If all the samples have been incorrectly classified then the error will be very high (approx. to 1) and hence our alpha value will be a negative integer.

Step 4 – You must be wondering why is it necessary to calculate the TE and performance of a stump? Well, the answer is very simple, we need to update the weights because if the same weights are applied to the next model, then the output received will be the same as what was received in the first model.

The wrong predictions will be given more weight whereas the correct predictions weights will be decreased. Now when we build our next model after updating the weights, more preference will be given to the points with higher weights.

After finding the importance of the classifier and total error we need to finally update the weights and for this, we use the following formula:

$$New\ sample\ weight\ =\ old\ weight\ *\ e^{\pm Amount\ of\ say\ (\alpha)}$$

The amount of say (alpha) will be ***negative*** when the sample is **correctly classified**.

The amount of say (alpha) will be ***positive*** when the sample is **miss-classified.**

There are four correctly classified samples and 1 wrong, here the ***sample weight*** of that datapoint is *1/5* and the ***amount of say/performance of the stump*** of ***Gender*** is *0.69*.

New weights for *correctly classified* samples are:

$$New\ sample\ weight\ =\ \frac{1}{5}\ *\ \exp(-0.69)$$

$$New\ sample\ weight\ =\ 0.2\ *\ 0.502\ =\ 0.1004$$

For *wrongly classified* samples the updated weights will be:

$$New\ sample\ weight\ =\ \frac{1}{5}\ *\ \exp(0.69)$$

$$New\ sample\ weight\ =\ 0.2\ *\ 1.994\ =\ 0.3988$$

Note: See the sign of alpha when I am putting the values, the **alpha is negative** when the data point is correctly classified, and this *decreases the sample weight* from 0.2 to 0.1004. It is **positive** when there is **misclassification**, and this will *increase the sample weight* from 0.2 to 0.3988

Row No.	Gender	Age	Income	Illness	Sample Weights	New Sample Weights
1	Male	41	40000	Yes	1/5	0.1004
2	Male	54	30000	No	1/5	0.1004
3	Female	42	25000	No	1/5	0.1004
4	Female	40	60000	Yes	1/5	0.3988
5	Male	46	50000	Yes	1/5	0.1004

Image Source: Author

We know that the total sum of the sample weights must be equal to 1 but here if we sum up all the new sample weights, we will get 0.8004. To bring this sum equal to 1 we will normalize these weights by dividing all the weights by the total sum of updated weights that is 0.8004. So, after normalizing the sample weights we get this dataset and now the sum is equal to 1.

Row No.	Gender	Age	Income	Illness	Sample Weights	New Sample Weights
1	Male	41	40000	Yes	1/5	0.1004/0.8004 =0.1254
2	Male	54	30000	No	1/5	0.1004/0.8004 =0.1254
3	Female	42	25000	No	1/5	0.1004/0.8004 =0.1254
4	Female	40	60000	Yes	1/5	0.3988/0.8004 =0.4982
5	Male	46	50000	Yes	1/5	0.1004/0.8004 =0.1254

Image Source: Author

Step 5 – Now we need to make a new dataset to see if the errors decreased or not. For this we will remove the “sample weights” and “new sample weights” column and then based on the “new sample weights” we will divide our data points into buckets.

Row No.	Gender	Age	Income	Illness	New Sample Weights	Buckets
1	Male	41	40000	Yes	0.1004/0.8004= 0.1254	0 to 0.1254
2	Male	54	30000	No	0.1004/0.8004= 0.1254	0.1254 to 0.2508
3	Female	42	25000	No	0.1004/0.8004= 0.1254	0.2508 to 0.3762
4	Female	40	60000	Yes	0.3988/0.8004= 0.4982	0.3762 to 0.8744
5	Male	46	50000	Yes	0.1004/0.8004= 0.1254	0.8744 to 0.9998

Image Source: Author

Step 6 – We are almost done, now what the algorithm does is selects random numbers from 0-1. Since incorrectly classified records have higher sample weights, the probability to select those records is very high.

Suppose the 5 random numbers our algorithm take is 0.38,0.26,0.98,0.40,0.55.

Now we will see where these random numbers fall in the bucket and according to it, we'll make our new dataset shown below.

Row No.	Gender	Age	Income	Illness
1	Female	40	60000	Yes
2	Male	54	30000	No
3	Female	42	25000	No
4	Female	40	60000	Yes
5	Female	40	60000	Yes

Image Source: Author

This comes out to be our new dataset and we see the datapoint which was wrongly classified has been selected 3 times because it has a higher weight.

Step 9 – Now this act as our new dataset and we need to repeat all the above steps i.e.

1. Assign ***equal weights*** to all the datapoints
2. Find the stump that does the ***best job classifying*** the new collection of samples by finding their Gini Index and selecting the one with the lowest Gini index
3. Calculate the ***"Amount of Say"*** and ***"Total error"*** to update the previous sample weights.
4. Normalize the new sample weights.

Iterate through these steps until and unless a low training error is achieved.

Suppose with respect to our dataset we have constructed 3 decision trees (DT1, DT2, DT3) in a ***sequential manner.*** If we send our **test data** now it will pass through all the decision trees and finally, we will see which class has the majority, and based on that we will do predictions
for our test dataset.

Conclusion

You have finally mastered this algorithm if you understood each and every line of this article.

We started by introducing you to what is Boosting and what are its various types to make sure that you understand where AdaBoost falls exactly. We have then applied straightforward math to see how every part of the formula works.

In the next article, I will explain about Gradient Descent and Xtreme Gradient Descent algorithm, which are few more important Boosting techniques to enhance the prediction power.

Did you find this article helpful? Please share your opinions/thoughts in the comments section below.

About the Author

I am an undergraduate student currently in my last year majoring in Statistics (Bachelors of Statistics) and have a strong interest in the field of data science, machine learning, and artificial intelligence. I enjoy diving into data to discover trends and other valuable insights about the data. I am constantly learning and motivated to try new things.

I am open to collaboration and work.

For any doubt and queries, feel free to contact me on Email

Connect with me on LinkedIn and Twitter

References:

Image 1 : https://blog.paperspace.com/adaboost-optimizer/

The media shown in this article on Interactive Dashboard using Bokeh are not owned by Analytics Vidhya and are used at the Author's discretion.

AdaBoost blogathon Ensemble Model machine learning