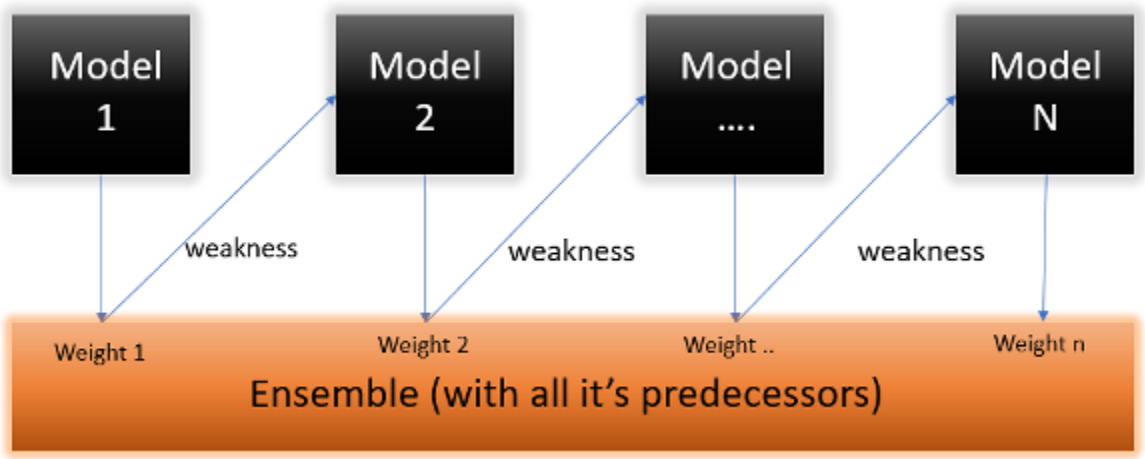
AdaBoost - How This Algorithm works

Introduction

If you ask random 10 data scientist about which ML Algorithm has been their goto algorithm for any classification problem, 9 out of 10 will choose Boosting algorithm, and the remaining one data scientist is probably a stoner.
Thats why as a data scientist, it is really important to understand the mechanics of boosting algorithm and how boosting actually works.
There is a famous indian brand called “Boost”, which had Sachin Tendulkar as their brand ambassador, so the first thing which comes to every indian mind when they probably hear the term Boosting is something energetic, something which makes your current life more productive and energetic.
Boosting Algorithm actually comes with the similar concept, it actually made the DecisionTrees more intuitive and made them superior.
Adaptive Boost is one of the boosting technique which gained popularity in recent times, and If you are planning to learn how exactly this algorithm works internally, then this article might help you in so many ways, Infact, i will share the jupyter notebook of exact working of how this algorithm is working and you can then try it out with your own dataset to understand it better

What is AdaBoost

AdaBoost short for Adaptive Boosting , is basically an ensemble method, where we combine weak learners into a strong learner



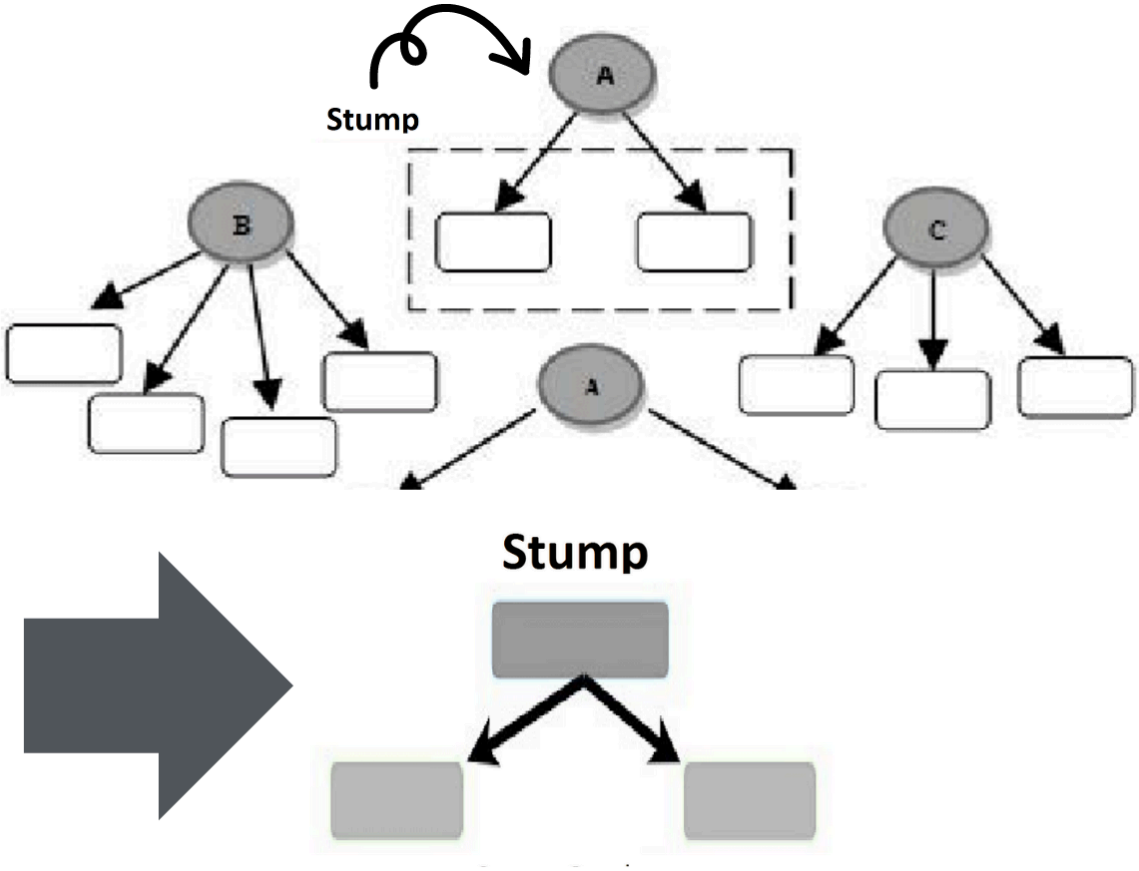
So This approach builds the model by assigning the equal weights to all data points initially and reevaluating those weights based on the model, and builds the model subsequently by combining the weak learners,
Here are the Step by Step APproach of by this algorithm actually works

Step 1: Assign an equal sample weight to each datapoints in the dataset

As I have already told you, we assign an equal sample weight to each datapoints in the dataset. This step ensures that initially, all data points carry the same importance in the training process. By assigning equal weights, AdaBoost treats each data point as equally significant at the outset, allowing the algorithm to learn from the entire dataset without bias towards any particular data point.
This uniform weighting sets the foundation for subsequent iterations of the algorithm, where the weights are adjusted based on the performance of the weak learners, ultimately leading to the creation of a strong learner that effectively combines multiple weak learners to make accurate predictions.

Step 2: Create M decision stumps, for M number of features

Step 2 of the AdaBoost algorithm involves the creation of M decision stumps, where M represents the number of features in the dataset
A **decision stump** is a simple, weak learner consisting of a one-level decision tree, essentially a tree with only a single split. These decision stumps are designed to make binary decisions based on a single feature, essentially acting as rudimentary classifiers. While decision stumps are individually weak learners with limited predictive power, they serve as the building blocks for the AdaBoost algorithm, contributing to the creation of a strong ensemble classifier through their collective performance.



AdaBoost aims to capture diverse aspects of the data, thereby improving the overall predictive accuracy of the model.
Through iterative training rounds, AdaBoost assigns varying weights to the decision stumps based on their performance, with misclassified data points receiving higher weights, allowing subsequent stumps to focus more on these challenging instances.
As a result, AdaBoost effectively leverages the simplicity and diversity of decision stumps to construct a robust ensemble classifier capable of making accurate predictions on complex datasets.

Step 3: Select one best decision tree based on Entropy or Gini Index

Step 3 of the AdaBoost algorithm involves selecting the best decision stump from the pool of M decision stumps created in the previous step.
This selection is based on evaluating the performance of each decision stump using a criterion such as Entropy or Gini coefficient. These metrics quantify the impurity or disorder of the data and help in identifying the decision stump that provides the best split.

Entropy, a measure of impurity in a dataset, is calculated by examining the distribution of class labels within the subsets created by a split. A lower entropy value indicates a more homogenous distribution of class labels, implying a better split.
Similarly, the Gini coefficient measures the impurity of a dataset by assessing the probability of misclassifying a randomly chosen element if it were randomly labeled according to the class distribution in the subset. Like entropy, a lower Gini coefficient signifies a more effective split.
During this step, AdaBoost compares the entropy or Gini coefficient produced by each decision stump and selects the one that yields the lowest value, indicating the most orderly and effective split.
By prioritizing decision stumps with lower impurity measures, AdaBoost ensures that the selected model can accurately classify a significant portion of the data, thereby enhancing the overall performance of the ensemble classifier.
This meticulous selection process forms a crucial aspect of AdaBoost's iterative training approach, allowing it to iteratively improve the ensemble's predictive capabilities with each subsequent round.

Step 4: Find the Total Error

Now, the focus shifts to evaluating the performance of the selected decision stump. This evaluation is centered on calculating the total error, which quantifies the number of misclassified observations resulting from the application of the decision stump to the training dataset.
The total error serves as a critical metric for evaluating the decision stump's effectiveness in classifying the data. A lower total error indicates successful classification of a substantial portion of the dataset, while a higher total error signifies suboptimal performance, necessitating further adjustments in subsequent iterations.

Step 5: Calculate the Performance/ or Amount of Say" of the decision stump

In Step 5 of the AdaBoost algorithm, the "Amount of Say" or the performance of the first decision stump is calculated. This metric reflects the contribution of the selected decision stump to the overall ensemble classifier's predictive power.

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

To calculate the "Amount of Say" for the first decision stump, AdaBoost considers the total error and the accuracy of the stump's predictions.

Step 6 :Update the Weight using Peformance/ Amount of Say

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

- When the sample is successfully identified, the amount of, say, (alpha) will be negative.
- When the sample is misclassified, the amount of (alpha) will be positive.

This step is crucial for adjusting the importance of each data point in subsequent iterations of the algorithm. The "Amount of Say" calculated for the first decision stump influences the weight updates, ensuring that misclassified data points receive higher weights while correctly classified points receive lower weights.
The calculation of the "Amount of Say" serves as a guiding factor in updating the weights, as decision stumps with higher performance are given more influence over the training process. Through this iterative weight adjustment process,
AdaBoost aims to construct a robust ensemble classifier capable of effectively capturing the complexities of the dataset and making accurate predictions.

Step 7: Create Buckets

In Step 7 of the AdaBoost algorithm, the creation of buckets or class intervals for the normalized weights is initiated to facilitate the generation of the second decision stump. This process involves organizing the weights of the data points into distinct intervals, each representing a range of normalized weights. By partitioning the weights into buckets,
For Example:

9]:

	f1	f2	f3	f4	output	pred_output	weights	adj_updated_weight	Buckets
0	-0.37	1.26	0.40	-0.59	0	0	0.05	0.02	0 - 0.021950897232253797
1	1.33	-0.21	1.17	-0.98	1	1	0.05	0.02	0.021950897232253797 - 0.04390179446450759
2	0.56	0.95	1.12	-1.16	0	1	0.05	0.02	0.04390179446450759 - 0.06585269169676139
3	0.89	-0.71	0.44	-0.25	1	1	0.05	0.16	0.06585269169676139 - 0.22804910276774626
4	0.47	1.16	1.16	-1.24	0	1	0.05	0.02	0.22804910276774626 - 0.25000000000000006
5	0.91	1.45	1.77	-1.82	1	1	0.05	0.02	0.25000000000000006 - 0.27195089723225385
6	-0.11	-0.87	-0.64	0.72	1	1	0.05	0.16	0.27195089723225385 - 0.4341473083032387
7	-0.86	1.47	0.05	-0.32	0	0	0.05	0.02	0.4341473083032387 - 0.4560982055354925
8	0.66	2.08	1.90	-2.05	1	1	0.05	0.02	0.4560982055354925 - 0.4780491027677463
9	1.40	0.04	1.39	-1.22	1	1	0.05	0.02	0.4780491027677463 - 0.5000000000000001
10	-0.05	-1.46	-0.93	1.09	1	1	0.05	0.16	0.5000000000000001 - 0.662196411070985
11	-1.41	-0.28	-1.54	1.40	0	0	0.05	0.02	0.662196411070985 - 0.6841473083032388
12	0.77	2.00	1.96	-2.09	1	1	0.05	0.02	0.6841473083032388 - 0.7060982055354925

Step 8 :Creating new dataset for subsequent model based on the updated weight

After that, we want to make a second weak model. But to do that, we need a sample dataset on which the second weak model can be run. For making it, we will run N number of iterations. On each iteration, it will calculate a random number ranging between 0-1 and this random will be compared with class intervals(Buckets) we created and on which class interval it lies, that row will be selected for a sample data set. So the new sample data set would also be of N observation.

Step 9: Repeat the above steps till convergence

In Step 9 of the AdaBoost algorithm, the process is repeated iteratively until convergence is achieved. This involves sequentially executing Steps 3 through 8, wherein each iteration entails training a new weak learner, evaluating its performance, updating weights based on misclassifications, creating new class intervals for sampling, and generating a new sample dataset.

This iterative refinement process continues until convergence criteria are met, typically defined by a predetermined number of iterations or when the performance of the ensemble classifier stabilizes.

Through these iterative adjustments and enhancements, AdaBoost progressively constructs a robust ensemble model capable of accurately predicting the target variable.

Conclusion

In conclusion, AdaBoost stands as a powerful ensemble learning algorithm that iteratively combines multiple weak learners to create a strong predictive model. Through its iterative training process, AdaBoost strategically adjusts weights, selects features, and constructs decision stumps to focus on challenging instances and improve overall predictive accuracy.

By harnessing the collective wisdom of its constituent weak learners, AdaBoost achieves convergence, producing a robust ensemble classifier capable of effectively handling complex datasets and making accurate predictions.

Its versatility, simplicity, and ability to adapt to various machine learning tasks make AdaBoost a valuable tool in the data scientist's toolkit, offering a reliable approach to solving classification problems across diverse domains.