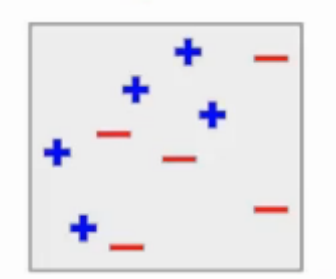
Adaptive boosting

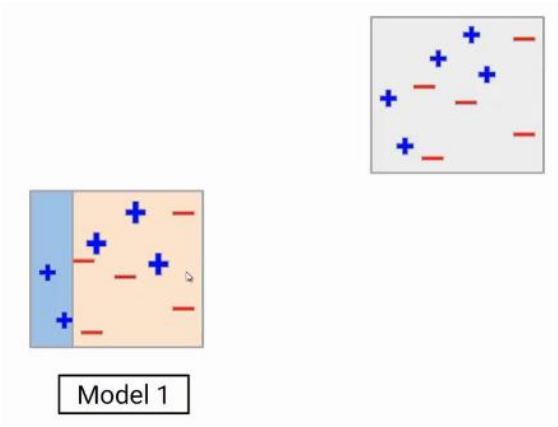
Example.



The first step is to build a model to classify this data.

Model-1

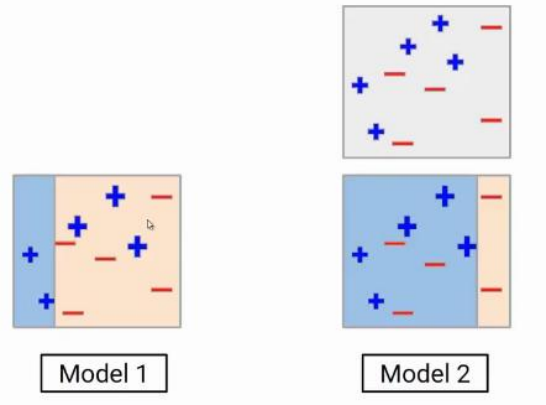
Suppose the first model gives the following result, where it is able to classify two blue points on the left side and all red points correctly. But the model also miss-classify the three blue points here.



Model 2

Now, these **miss- classified data points will be given higher weight**. So these three blue positive points will be given higher weights in the next iteration. For representation, the points with higher weight are bigger than the others in the image.

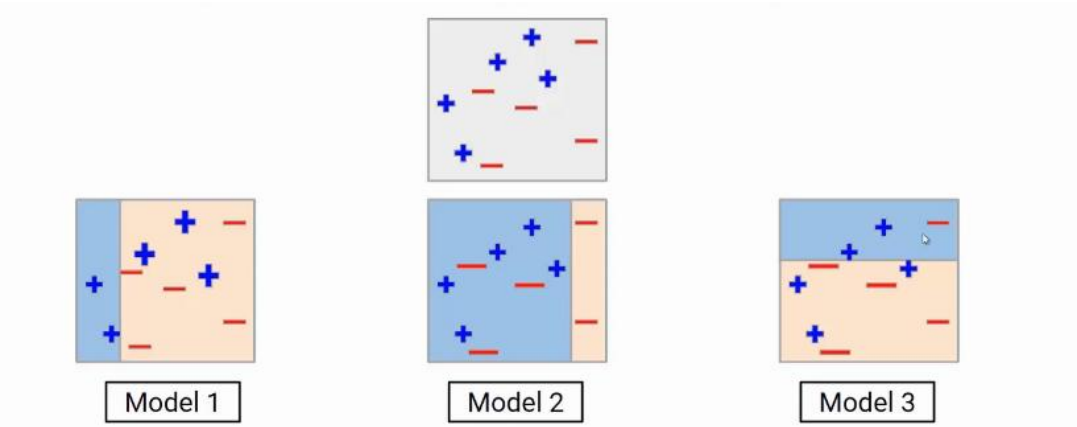
Giving higher weights to these points means model is going to focus more on these values. Now build a new model.



In the second model you will see, the model boundary has been shifted to the right side in order to correctly classify the higher weighted points. Still, it's not a perfect model. You will notice three red negatives are miss-classified by model 2.

Model 3

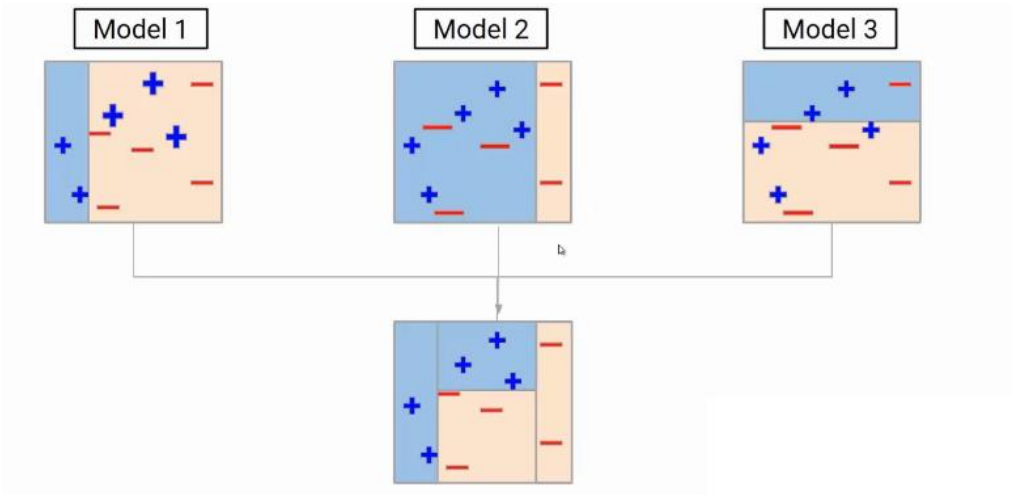
Now, these miss-classified red points will get a higher weight. Again we will build another model and do the predictions. The task of the third model is two focus on these three red negative points. So the decision boundary will be something as shown here.



This new model again incorrectly predicted some data points. At this point, we can say all these individual models are not strong enough to classify the points correctly and are often called weak learners.

Ensemble

In the next step, we have to aggregate these models. One of the ways could be taking the weighted average of the individual weak learners. So our final model will be the weighted mean of individual models.



After multiple iterations, we will be able to create the right decision boundary with the help of all the previous weak learners. As you can see the final model is able to classify all the points correctly. This final model is known as a strong learner.

Let's once again see all the steps taken in AdaBoost.

Adaptive boosting

1. Build a model and make predictions.
2. Assign higher weights to miss-classified points.
3. Build next model.
4. Repeat steps 3 and 4.
5. Make a final model using the weighted average of individual models.

Example

Row No.	Feature 1	Feature 2	Feature 3	Output	Sample Weight
1				Yes	1/5
2				Yes	1/5
3				No	1/5
4				No	1/5
5				Yes	1/5

Consider sample dataset consisting of only three features where the output is in categorical form. As the output is in binary/categorical form, it becomes a classification problem. In real life, the dataset can have any number of records and features in it. Let us consider 5 datasets for the purposes.

The output is in categorical form, here in the form of *Yes* or *No*. All these records will be assigned a sample weight.

The formula used for this is '**W=1/N'** where **N** is the number of records. In this dataset, there are only 5 records, so the sample weight becomes **1/5** initially.

Every record gets the same weight. In this case, it's 1/5.

Step 1 – Creating the First Base Learner

To create the first learner, the algorithm takes the first feature, i.e., **feature 1** and creates the first stump, **f1**.

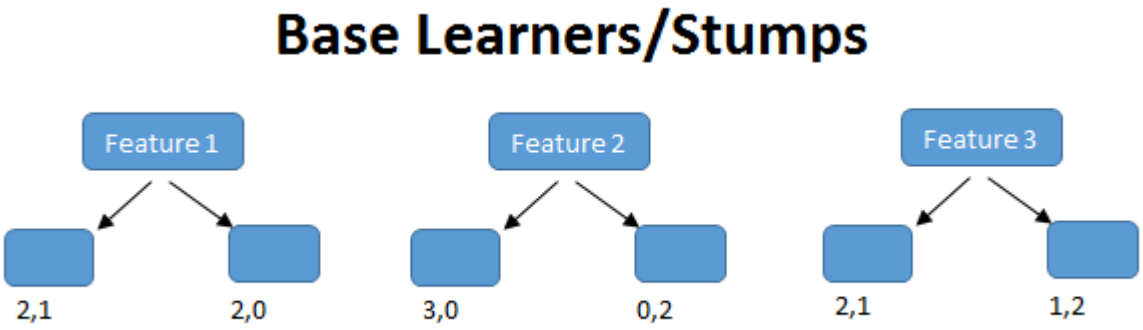
It will create the same number of stumps as the number of features. In the case below, it will create 3 stumps as there are only 3 features in this dataset. From these stumps, it will create three decision trees. This process can be called the **stumps-base learner model.**

Out of these 3 models, the algorithm selects only one. **Two properties are considered while selecting a base learner – Gini and Entropy.** We must calculate Gini or Entropy the same way it is calculated for decision trees.

The stump with the least value will be the first base learner. In the figure below, all the 3 stumps can be made with 3 features. The number below the leaves represents the correctly and incorrectly classified records.

By using these records, the Gini or Entropy index is calculated. **The stump that has the least Entropy or Gini will be selected as the base learner.**

Let's **assume that the entropy index is the least for stump 1.** So, let's take stump 1, i.e., feature 1 as our first base learner.



Row No.	Feature 1	Feature 2	Feature 3	Output	Sample Weight
1				Yes	1/5
2				Yes	1/5
3				No	1/5
4				No	1/5
5				Yes	1/5

Here, **feature (f1) has classified 2 records correctly and 1 incorrectly.** The row in the figure that is marked red is incorrectly classified. For this, we will be calculating the total error.

Step 2 – Calculating the Total Error (TE)

The total error 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 3 – Calculating Performance of the Stump

Formula for calculating Performance of the Stump is: –

$$\text{Performance of Stump} = \frac{1}{2} \ln \left[\frac{1-TE}{TE} \right]$$

where, **ln** is natural log and **TE** is Total Error.

In our case, TE is 1/5. By substituting the value of total error in the above formula and solving it, we get the value for **the performance of the stump as 0.693.**

Why is it necessary to calculate the TE and performance of a stump?

- The answer is, we must update the sample weight before proceeding to the next model or stage because if the same weight is applied, the output received will be from the first model. In boosting, only the wrong records/incorrectly classified records would get more preference than the correctly classified records. Thus, **only the wrong records from the decision tree/stump are passed on to another stump**. Whereas, **in AdaBoost, both records were allowed to pass and the wrong records are repeated more than the correct ones.**

We must **increase the weight for the wrongly classified records** and decrease the weight for the correctly classified records. In the next step, we will be updating the weights based on the performance of the stump.

Step 4 – Updating Weights

For incorrectly classified records, the formula for updating weights is:

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

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

For correctly classified records, we use the same formula with the performance value being negative. This leads the weight for correctly classified records to be reduced as compared to the incorrectly classified ones. The formula is:

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

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

Row No.	Feature 1	Feature 2	Feature 3	Output	Sample Weight	Updated Weight
1				Yes	1/5 ↓	0.1
2				Yes	1/5 ↑	0.399
3				No	1/5 ↓	0.1
4				No	1/5 ↓	0.1
5				Yes	1/5 ↓	0.1
-	-	-	-	-	1	0.799

The updated weight for all the records can be seen in the figure.

As is known, **the total sum of all the weights should be 1**. In this case, it is seen that the total updated weight of all the records is not 1, it's 0.799. To bring the sum to 1, **every updated weight must be divided 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.

Row No.	Feature 1	Feature 2	Feature 3	Output	Sample Weight	Updated Weight	Normalized Weight
1				Yes	1/5 ↓	0.1	0.13
2				Yes	1/5 ↑	0.399	0.50
3				No	1/5 ↓	0.1	0.13
4				No	1/5 ↓	0.1	0.13
5				Yes	1/5 ↓	0.1	0.13
-	-	-	-	-	1	0.799	1

Step 5 – Creating a New Dataset

Now, 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.

The new dataset has to be created using and considering the **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.

Normalized Weight	Buckets
0.13	0 - 0.13
0.50	0.13 - 0.63
0.13	0.63 - 0.76
0.13	0.76 - 0.89
0.13	0.89 - 1.02

After this the algorithm will run 5 iterations to select different records from the older dataset.

Suppose in the 1st iteration, the algorithm will take a random value **0.46** to see which bucket that value falls into and select that record in the new dataset.

It will again select a random value, see which bucket it is in and select that record for the new dataset. The **same process is repeated 5 times**.

There is a high probability for wrong records to get selected several times. This will form the new dataset. It can be seen in the image below that row number 2 has been selected multiple times from the older dataset as that row is incorrectly classified in the previous one.

Row No.	Feature 1	Feature 2	Feature 3	Output
2				Yes
3				No
2				Yes
5				Yes
2				Yes

Based on this new dataset, the algorithm will create a new decision tree/stump and it will repeat the same process from step 1 till it sequentially passes through all stumps and finds that there is less error as compared to normalized weight that we had in the initial stage.

How Does the Algorithm Decide Output for Test Data?

Suppose with the above dataset, the algorithm constructed 3 decision trees or stumps.

- The test dataset will pass through all the stumps which have been constructed by the algorithm. While passing through the 1st stump, the output it produces is 1.
- Passing through the 2nd stump, the output generated once again is 1. While passing through the 3rd stump it gives the output as 0.
- In the AdaBoost algorithm too, the majority of votes take place between the stumps, in the same way as in random trees.
- In this case, the final output will be 1. This is how the output with test data is decided.

Example: Understanding the working of AdaBoost Algorithm

Step 1 – The Image is shown below is the 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

Adaptive boosting

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:

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

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

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

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

$$\alpha = 0.69$$

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

0 Indicates perfect stump and 1 indicates horrible stump.

Adaptive boosting

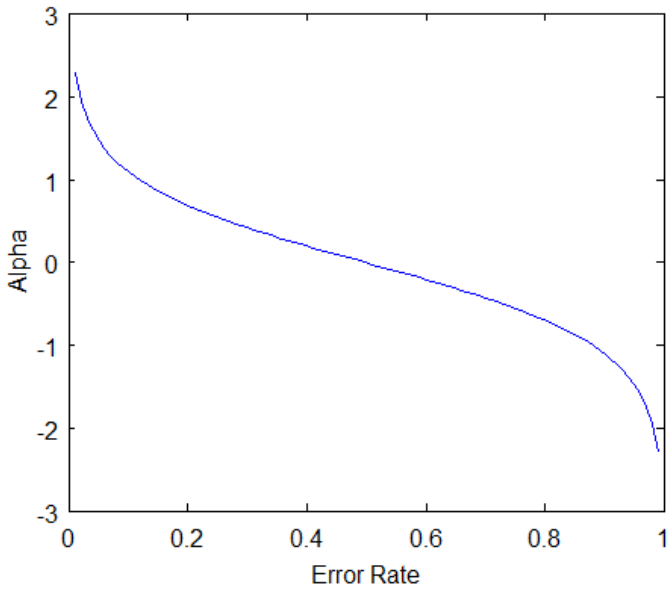


Image1

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 –It necessary to calculate the TE and performance of a stump. 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 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

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.

Adaptive boosting

Row No.	Gender	Age	Income	Illness
1	Female	40	60000	Yes
2	Male	46	50000	No
3	Female	42	25000	No
4	Female	40	60000	Yes
5	Female	40	60000	Yes

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.