# ADABoost Classifier

## Boosting in ML:

Boosting is an ensemble learning technique where multiple weak learners(model) works together to build a strong Learner/classifier/ model.

Example of Boosting Algorithms → AdaBoost, Gradient Tree boosting, XBoost.

Let is say we are setting some rules to classify an Email as Spam and Not Spam. Rules →

1) Email has promotional Image → Spam
2) Email has link(s) → Spam
3) Email consists of sentences like "You won a prize money of $....." — Spam
4) Email from our official Domain "bracu.com" — Not Spam
5) Email from know source — Not Spam.

If we consider these rules individually while cheeking for spam and not spam email, we

might mark 'spam' emails as 'not spam' and 'not spam' emails as 'spam'.

Rule-2 says, Email has link(s). We may receive links from known sources, which are not spam. But if Rule-2 works individualy it may mark a 'not spam' email as 'spam'.

According to rule-4, if email consists of our official domain "bracu.com", the email is "Not spam". But we sometimes get spam and fake emails from similar domains. So, we should mark those mis-leading emails as spam.

So, instead of using each rule individually, we should combine all rules together while checking for spam emails.

Combining predictions of weak learners:

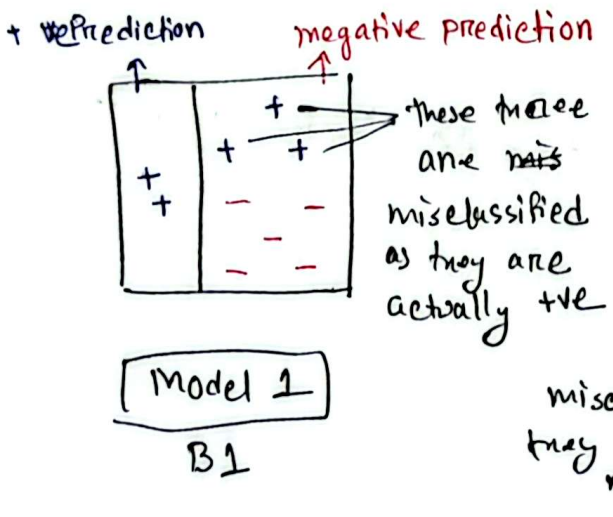
We can combine the results / predictions of multiple weak learners using the following methods →

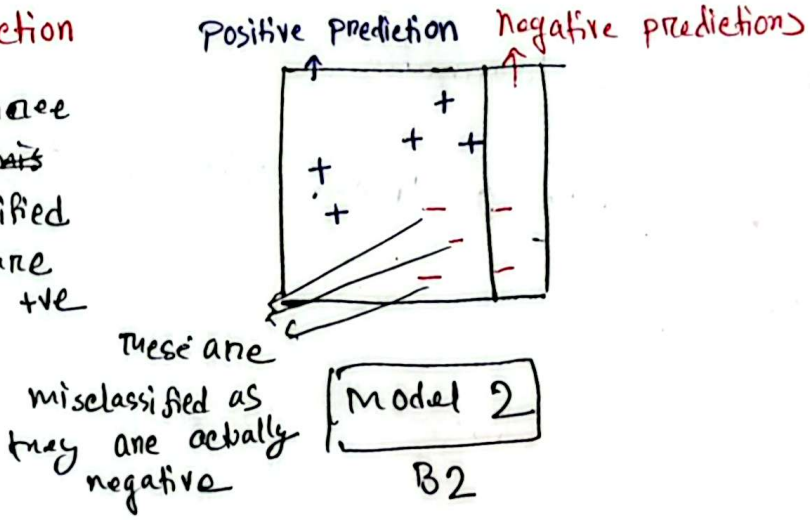
1) ~~Use~~ Considering the prediction that has higher vote
2) using average / weighted average.

# Adaptive Boosting

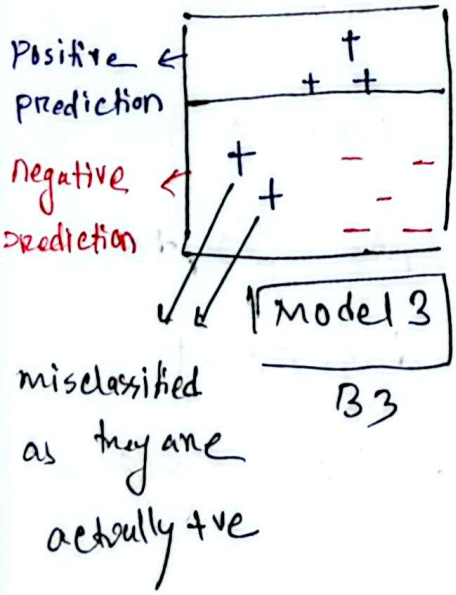
Adaboost is the first successful boosting algorithm developed for the purpose of binary classification.

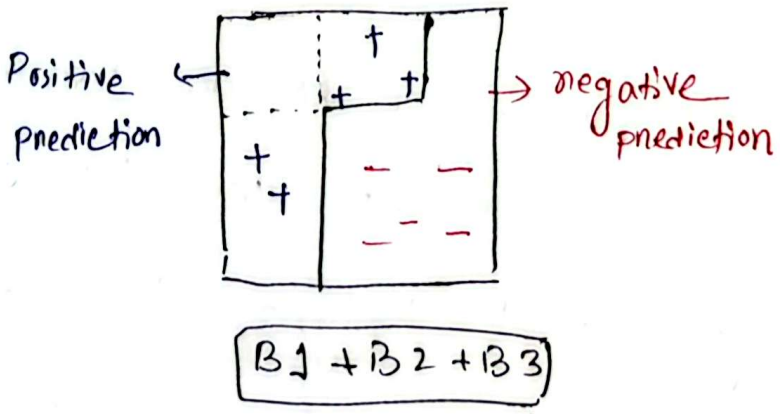
Suppose there are three weak learners. We can see the classification done by each learners in a pictorial view as below →




Now if we combine $B_1 + B_2 + B_3$




This combined model is successfully combining all data points correctly.

# ADABoost with Tree (CT and RT):

In ~~Regressi~~ Random Forest we constructed multiple weak trees and combined those as Random Forest model.

In Adaboost we will construct stumps. The Adaboost will consist of multiple stumps. Stumps will work as weak learners.

**stump**: Its a tree with just one Node and its leaves.

## AdaBoost Procedure:

**STEP-1**: Initialize the dataset and assign eaual weights to each of the datapoint.

Let us assume we are running Adaboost for the following dataset with **14 datapoints** so, to assign eaual weight to each datapoint we will divide 1 by 14. So, the weight for each data point is $\frac{1}{14}$.

Dataset →

| Day | Outlook | Temperature | Humidity | wind | Play Tennis | Sample Weight |
|---|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | weak | N | 1/14 |
| D2 | Sunny | Hot | High | Strong | N | 1/14 |
| D3 | Overcast | Hot | High | weak | Y | 1/14 |
| D4 | Rain | Mild | High | Weak | Y | 1/14 |
| D5 | Rain | cool | Normal | Weak | Y | 1/14 |
| D6 | Rain | cool | Normal | Strong | N | 1/14 |
| D7 | Overcast | cool | Normal | Strong | Y | 1/14 |
| D8 | Sunny | Mild | High | Weak | NO | 1/14 |
| D9 | Sunny | Cool | Normal | Weak | Y | 1/14 |
| D10 | Rain | mild | Normal | Weak | Y | 1/14 |
| D11 | sunny | Mid | Normal | Strong | Y | 1/14 |
| D12 | Overcast | Mild | High | strong | Y | 1/14 |
| D13 | Overcast | Hot | Normal | Weak | Y | 1/14 |
| D14 | Rain | Mild | High | Strong | N | 1/14 |

Features

target
NO/Yes

Assigned weights

Now we will find the first stump for this dataset.

## Step-2 : Building the Stump :

To construct the stump we will use Gini Index as this ~~dataset is~~ is classification problem.

The stump will consist of one node. we need to find the ~~node~~ value of that node using Gini Index. Unlike Random forest we have find Gini of all features.

### ~~Outlo~~ Outlook feature :


D1, D2, D8
D9, D11

D3, D7, D12
D13

D4, D5, D6, D10
D14

Gini(outlook=sunny)

$= 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2$

$= 0.48$

Gini(outlook=overcast)

$= 1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{4}\right)^2$

$= 0$

Gini(outlook=Rain)

$= 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2$

$= 0.48$

$\therefore$ Gini(outlook) $= \frac{5}{14} \times 0.48 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.48$

$= 0.348$

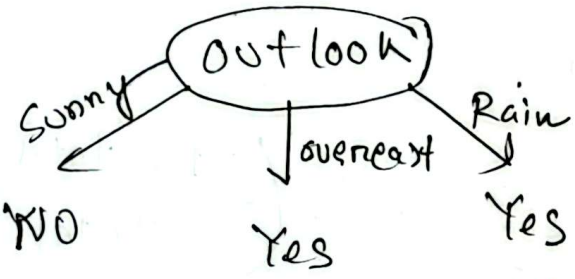
Similarly if we find Gini Impurity for all other features - we get

Gini(Temparature) = 0.441

Gini(Humidity) = 0.367

Gini(Wind) = 0.429

Among these 4 Gini impurities, Gini(outlook) = 0.34 is lowest. So, outlook will be the first stump.




D9, D11 → Yes : 2

D1, D2, D8 → NO : 3

↓

This leaf predicts all sunny examples as 'NO'. But there are two Yes Example in sunny category D9 and D11. These two are being misclassified

D3, D7, D12, D13 → Yes : 4

NO : 0

↓

No datapoint is being misclassified here

Yes : 3 → D4, D5, D10

NO : 2 → D6, D14

↓

2 'NO' datapoint D6 and D14 are being misclassified

So, this stump is not perfect as few datapoints, D9, D11, D6 and D14 are being misclassified. So, we need to find how well this stump is working/ classifying.

## Step-3: Determine Amount of say ($\alpha$)

Amount of say is the ~~determination~~ measurement of how well it classified the samples in the classification.

$$\text{Amount of say } \alpha = \frac{1}{2} \times \ln\left(\frac{1 - \text{total error}}{\text{total error}}\right)$$

Total error is the sum of the weights associated with incorrectly classified samples.

### Total Error:

We have seen,

Sunny branch misclassified two data points D9, D11

~~overcast~~ Outlook branch did not misclassified any datapoint

Rain branch misclassified two data points D6, D14

The datapoints are errors. Each datapoint is weighted $\frac{1}{14}$.

$\therefore$ Total Error $= 2 \times \frac{1}{14} + 0 \times \frac{1}{14} + 2 \times \frac{1}{14} = 0.29$

2 misclassified data in sunny branch

No misclassified data in outlook branch.

2 misclassified data in Rain branch

$\therefore$ Amount of say of first stump

$$\alpha = \frac{1}{2} \times \ln \frac{1 - 0.29}{0.29}$$

$$= 0.45$$

Now as 4 datapoints have been misclassified, we need to increase the attention towards these datapoints, so that the next stump focuses on these datapoints more. Also, we can decrease the attention towards the classified samples as they are already correctly classified by the first stump. We do that by updating the sample weights.

Now we will proceed to create new dataset for the next stump.

## Step-4 : Update the sample weight

New sample weight for **incorrectly classified** samples = sample weight (previous) $\times e^{+\alpha}$ (+ve for increasing weight/attention)

$= \frac{1}{14} \times e^{0.45}$

$= 0.11$

New sample weight for **correctly classified** samples = sample weight (previous) $\times e^{-\alpha}$ (negative for decreasing weight/attention)

$= \frac{1}{14} \times e^{-0.45}$

$= 0.046$

## Step-5: Normalized and cumulative Normalized sample weight.

After assigning new weights for 14 data points we can see the summation of all new weight do not give us 1. So we need to normalized each weights.

| Day | Features | Play tennis | Updated sample weight | Normalized sample weight | Cumulative normalized |
|---|---|---|---|---|---|
| D1 | | N | 0.046 | 0.051 | 0.051 |
| D2 | | N | 0.046 | 0.051 | 0.102 |
| D3 | | Y | 0.046 | 0.051 | 0.153 |
| D4 | | Y | 0.046 | 0.051 | 0.204 |
| D5 | | Y | 0.046 | 0.051 | 0.255 |
| D6 | | No | 0.11 | 0.122 | 0.377 |
| D7 | | Y | 0.046 | 0.051 | 0.428 |
| D8 | | N | 0.046 | 0.051 | 0.479 |
| D9 | | Y | 0.11 | 0.122 | 0.601 |
| D10 | | Y | 0.046 | 0.051 | 0.652 |
| D11 | | Y | ~~0.046~~ 0.11 | 0.122 | 0.774 |
| D12 | | Y | 0.046 | 0.051 | 0.825 |
| D13 | | Y | 0.046 | 0.051 | 0.876 |
| D14 | | N | 0.11 | 0.122 | 0.998 |

Summation of Updated sample weight = $10 \times 0.046 + 4 \times 0.11$

$= 0.9$

Normalized weight for correctly classified samples $= \frac{0.046}{0.9} = 0.051$

Normalized weight for incorrectly classified samples $= \frac{0.11}{0.9} = 0.122$

So, we write all Normalized weights. Now we will calculate the cumulative Normalized weight [adding immediate weight in increasing order]

Step-6: Generate random number between 0-1.

For each datapoint, we will generate random numbers.

| Day | Features | Play Tennis | Cumulative Normalized weight | Generated Random Numbers |
|---|---|---|---|---|
| D1 | | N | 0.051 | 0.090 |
| D2 | | N | 0.102 | 0.100 |
| D3 | | Y | 0.153 | 0.151 |
| D4 | | Y | 0.204 | 0.200 |
| D5 | | Y | 0.255 | 0.250 |
| D6 | | N | 0.377 | 0.267 |
| D7 | | Y | 0.428 | 0.500 |
| D8 | | NO | 0.479 | 0.700 |
| D9 | | Y | 0.601 | 0.990 |
| D10 | | Y | 0.652 | 0.370 |
| D11 | | Y | 0.774 | 0.600 |
| D12 | | Y | 0.825 | 0.682 |
| D13 | | Y | 0.876 | 0.886 |
| D14 | | N | 0.998 | 0.980 |

## Step 7 : Create New dataset for 2nd Stump :

Now we will check for Each random numbers. We will ~~check~~ consider the cumulative Normalized weights as Range.

Ranges are 0 – 0.051, 0.051 – 0.102 ....
0.876 – 0.998.

We well check in which random numbers falls into.

The first random number 0.040 Falls into range 0–0.051, or 0.040 < 0.051. So we will select the data point corresponding to 0.051, which is **Day 1**. So Day 1 is selected for the ~~dat~~ new dataset.

Lets check another random number 0.700, which falls into 0.652 – 0.774 range, or 0.700 < 0.774. So we select the **Day 11** that is corresponding to 0.774 weight

we do this for all 14 random numbers, and will get 14 ~~rows~~ datapoints for the new dataset as below →

New dataset

| Day | Outlook | Temparature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | | | | | |
| D2 | | | | | |
| D3 | | | | | |
| D4 | | | | | |
| D5 | | | | | |
| D6 | | | | | |
| D9 | | | | | |
| D11 | | | | | |
| D14 | | | | | |
| D6 | | | | | |
| D9 | | | | | |
| D11 | | | | | |
| D14 | | | | | |
| D14 | | | | | |

Here we can see, the misclassified examples D6, D9, D11 and D14 has been repeated multiple times. So, the next stump will forces more on this datapoints.

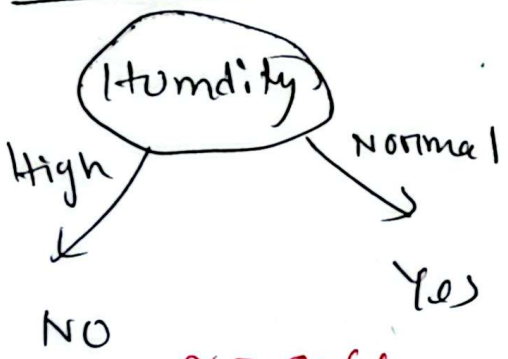
Etc.

Now Repeat step-1 – Step 7 for this new dataset and create the second stump.

If we continue this process two times more, we will get two more stumps.

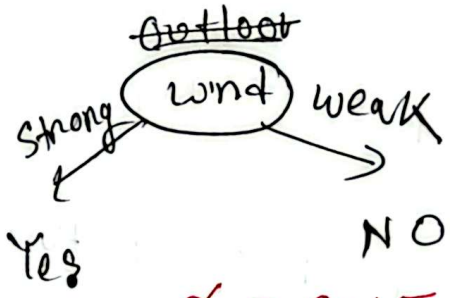
Let's assume, we got the following two stumps,

Stump-2



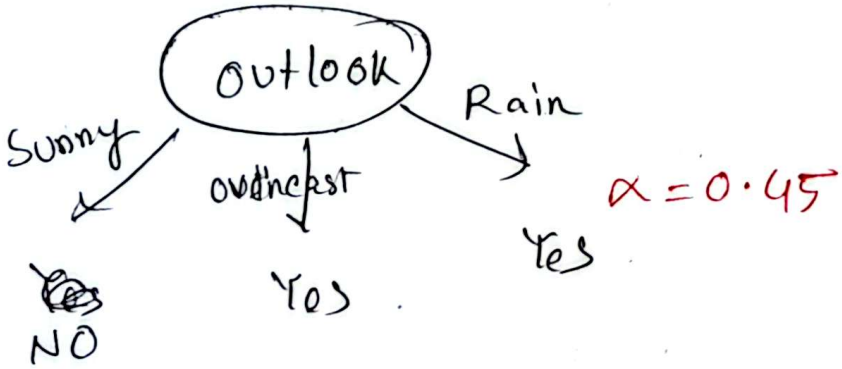

$\alpha = 0.66$

Stump-3




$\alpha = 0.15$

The stump 1 we got




$\alpha = 0.45$

Now we will do the Bagging process.

# Bagging :

Let's we will ~~assume~~ predict the class

For, Outlook = Sunny, temparature = Mild,

Humidity = Normal and Wind = Weak.

If we run this input all three stumps →

we get →

Stump-1 : NO $\alpha = 0.45$

Stump-2 : Yes $\alpha = 0.66$

stump-3 : NO, $\alpha = 0.05$

So, The final prediction,

Weighted voted of Yes = 0.66

weighted voted of NO = 0.45 + 0.05 = 0.50

As weight of Yes is largeon, the final

Result will be Yes.

We can use adaboost Algoritum for Regression problem as well.