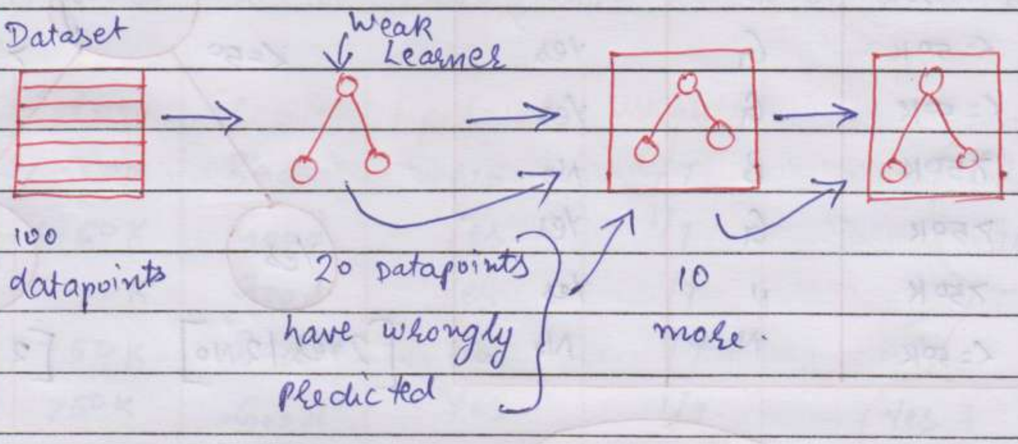
Machine Learning : ADABOOSTING

- It is Sequential
- It Provides computational scalability
- It is used to exploit / use the dependency between models
- Stagewise additive multimodeling using Multiclass Exponential Loss Function. (Performance of Stump)
- Decision Stump : Decision tree created only upto one Depth
- Ada Boosting can handle missing values and outliers.
- Ada Boosting can handle mixed predictors as well (Quantitive and qualitative)

In ada boosting, we combine many weak learners in series and every weak learner gives output by adding some weight and at the end we get strong learner



$$f = \alpha_1(M_1) + \alpha_2(M_2) + \alpha_3(M_3) + \cdots + \alpha_n(M_n)$$

M : Model

α : Weight of every model.

⇒ weight shows how strong the decision tree predict some point

⇒ If (α_1) weight is high, this means the model (M_1) will take the responsibility of making the prediction.

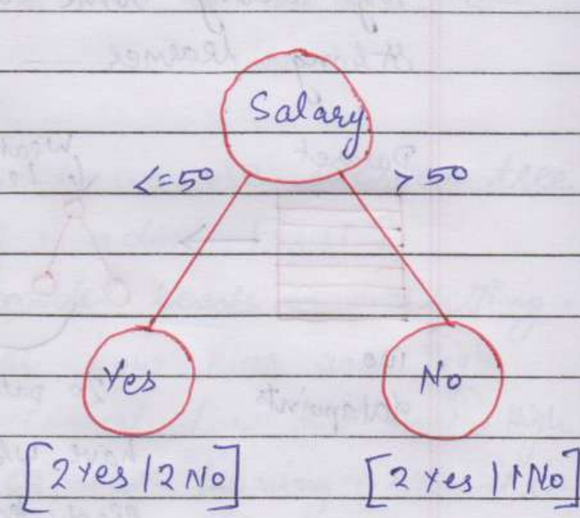
⇒ If (α) weight is -ve, this means the model (m) is doing no work.

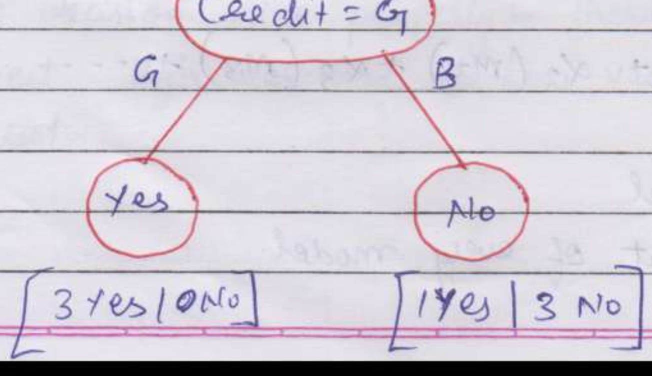
$M_1, M_2, M_3 \cdots M_n \rightarrow$ Weak Learners

⇒ Weak learners are the learners in which decison tree is created only upto 1 Depth. Therefore, it can not give right prediction.

Example DATASET :-

Salary	Credit	Approval
<=50K	B	No
<=50K	G	Yes
<=50K	G	Yes
>50K	B	No
>50K	G	Yes
>50K	N	Yes
<=50K	N	No





Steps for Adaboost algorithm

=> How to assign weights?

Salary	Credit	Approval	weights
<=50K	B	No	1/7
<=50K	G	Yes	1/7
<=50K	G	Yes	1/7
>50K	B	No	1/7
>50K	G	Yes	1/7
>50K	N	Yes	1/7
<=50K	N	No	1/7

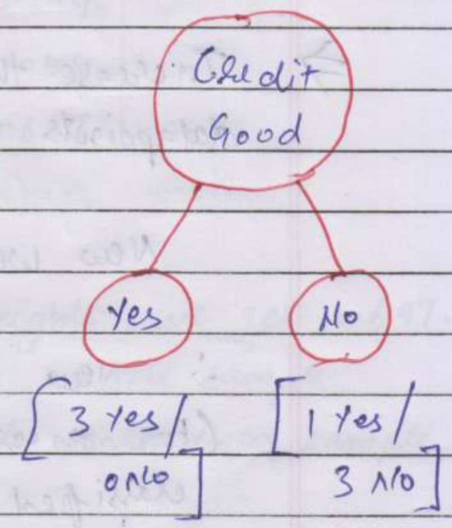
G = Good
B = Bad
N = Normal

STEP 1:- Initiallize the weights as 1/n to every n observations.

STEP 2:- Select the feature acording to Lowest Gini / Highest Information Gain and calculate the Total Error.

In our dataset lowest Gini Impurity is of "Credit" Feature. So the table will be become like below:

Salary	Credit	Approval	Weights
<=50K	Bad	No	1/7
<=50K	Good	Yes	1/7
<=50K	Good	Yes	1/7
>50K	Bad	No	1/7
>50K	Good	Yes	1/7
>50K	Normal	Yes	1/7
<=50K	Normal	No	1/7



This stump classified only one data incorrectly i.e.
>50K Normal Yes.

Because If the Credit is not good, then Approval should be No

So our Total error will be $\left(\frac{1}{7}\right)$

STEP 3 Calculate the Performance of the stump.

$$\text{Performance of the stump} = \frac{1}{2}\left(\log\left(\frac{1-\text{total error}}{\text{total error}}\right)\right)$$

$$\therefore \text{Performance of the stump } (\alpha_1) = \frac{1}{2}\log\left(\frac{1-1/7}{1/7}\right)$$

$$= \frac{1}{2}\log(6) = 1.79 \times \frac{1}{2}$$

$$\boxed{\alpha_1 = 0.895}$$

Step 4 Calculate the new weights for each classified datapoint.

⇒ Increase the sample weight for incorrectly ~~co~~ classified datapoints

$$\text{New weight} = \text{old weight} * e^{(+\alpha)}$$

$$\therefore \text{New weight} = \frac{1}{7} * e^{(0.895)}$$

(For incorrectly classified datapoint)

$$\therefore \text{New weight} = 0.349$$

=> Decrease the sample weight for correctly classified datapoints.

New weight = old weight * $e^{(-\alpha)}$

∴ New weight = $\frac{1}{7} * e^{(-0.875)}$
(For correctly classified datapoints)

∴ New weight = 0.058

So the updated weight will be.

Salary	Credit	Approval	weight	Updated weight
<=50K	Bad	No	1/7	0.058
<=50K	Good	Yes	1/7	0.058
<=50K	Good	Yes	1/7	0.058
>50K	Bad	No	1/7	0.058
>50K	Good	Yes	1/7	0.058
>50K	Normal	Yes	1/7	0.349
<=50K	Normal	No	1/7	0.058
			Total	0.697

Step 5 Normalize the sample weight:

If we add all the updated weights, we get 0.697. Hence, for normalization we divide all the sample weights by 0.697 and then create normalized sample weights as shown below.

Salary	Credit	Approval	Updated weight	Normalized weight
<=50K	Bad	No	0.058	0.083
<=50K	Good	Yes	0.058	0.083
<=50K	Good	Yes	0.058	0.083
>50K	Bad	No	0.058	0.083
>50K	Good	Yes	0.058	0.083
>50K	Normal	Yes	0.349	0.501
<=50K	Normal	No	0.058	0.083
		Total	0.697	1

These new normalized weight will act as the sample weight for the next iteration.

Step 6 Now Repeat from step2 and so on till the configured number of estimators reached or the accuracy achieved.

⇒ Now this will be the output of 1 decision tree, similarly we will receive output of many decision trees.

⇒ Suppose, m trees (stumps) are classifying a person get approval "yes" and n trees (stumps) are classifying a person get approval "No", then the performance of the stumps (m trees and n trees) are added seperately and whichever has the highest value, the person gets approval as that.

For example:-

If the performance of stump is 1.2 and the performance of n trees stump is 0.5 then the final result will go in the favour of m trees and the ~~perfo~~ person will get the approval "Yes"