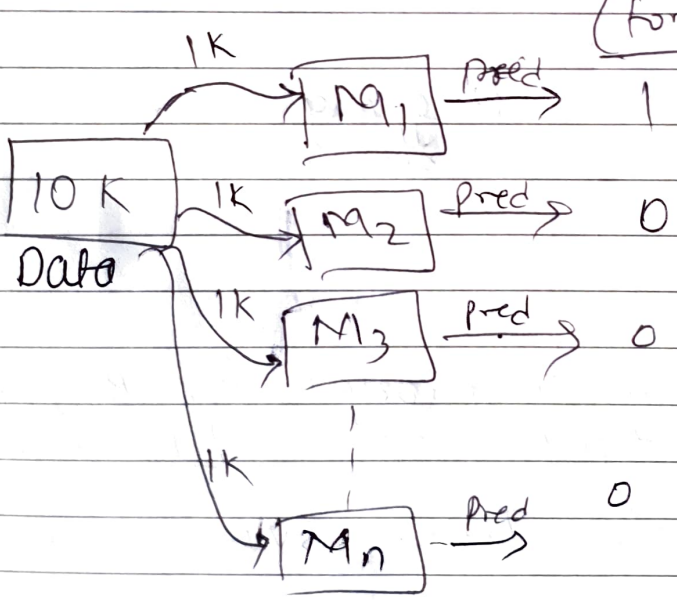
Bagging

Bootstrapping → technique where population is being drawn from random sample.

Aggregation

All the base models ~~is~~ used for Bagging is ~~same~~ same.

for e.g. If I have 10 models then all 10 models should be having similar kind.



∴ By using majority (mode) the output will be predicted as 0

↓

known as aggregation.

all these 1000 rows (Data) given to each model ~~@~~ using sampling will be diff while training

↓

known as Bootstrapping

Why Bagging ?

→ for better result, LB & LV

there is negative correlation bet^n bias & variance.

$$\text{bias} \propto \frac{1}{\text{variance}}$$

→ Ideally getting LB & LV is not feasible all the time due to their relation.

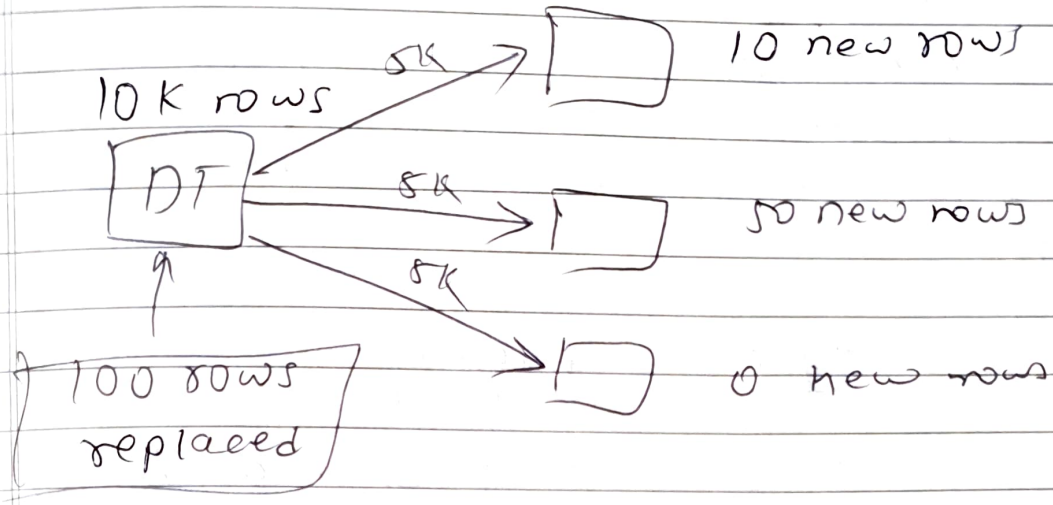
∴ Either we get LB HV
or HB LV

Hence, By using Bagging, LB & LV can be achieved.

Explaination

while selecting a model for Bagging consider models with LB & HV such as Decision Tree., SVM, KNN

Note: LB & HV will have Overfitting issues



the above diag. says if 100 new rows are added / replaced then due to bootstraping it is always the case that all newly added rows will not come under training data of each base model.

Hence biased & variance both will be low in such cases.

Or

consistent results can be obtained.

When to Use Bagging ?

→ whenever you have a project or dealing with LB HV data then in such cases bagging will be useful.

types of Bagging -

① Pasting :- Row sampling same as bagging but without replacement.

② Random subspaces :- column sampling with or without replacement.

③

```
df = pd.read_csv('iris.csv')
df = df.sample(10)
df.sample(2, replace=True, axis=1)
```

(2 → No. of columns; replace=True → with replacement; axis=1 → represents column sampling)

③ Random patches :- Row sampling & column sampling

```
df.sample(8, replace=True).sample(2, replace=True, axis=1)
```

PAGE No.
DATE: / /

Bagging classifier

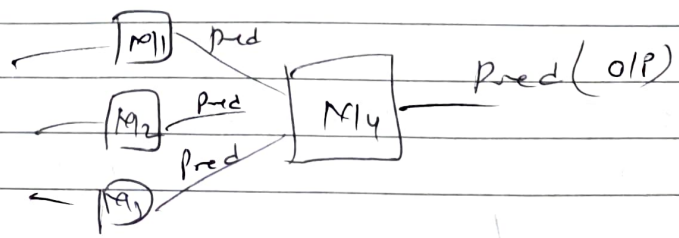
Keypoints Related to bagging

1. Bagging generally gives better results than pasting
2. Good results come around the 25% to 50% row sampling max
3. Random patches and subspaces should be used while dealing with high dimensional data
4. To find the correct hyperparameter values we can do gridsearchcv / randomsearchcv

Stacking

It extends ideas of max voting.

Steps:-

1. Train your base models.
2. Predict the results using base model.
3. based on the prediction of base model, take that o/p and use it to train the meta model.
4. 

Stacking vs Bagging (boosting)

	Stacking	Bagging (boosting)
①	diff base models can be used.	Same base models are used.
②	New training	No new training

Drawback of stacking

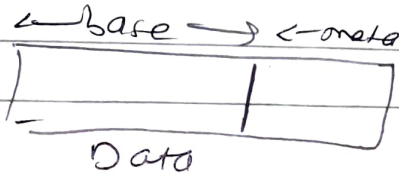
Overfitting.

→ Reason is that we are using same data for the training of base model & then on the basis of same data only we are training the meta model too. Hence if there is overfitting while training the base model (which is always the case) then predicted result from meta model will also have overfitting.

How to overcome?

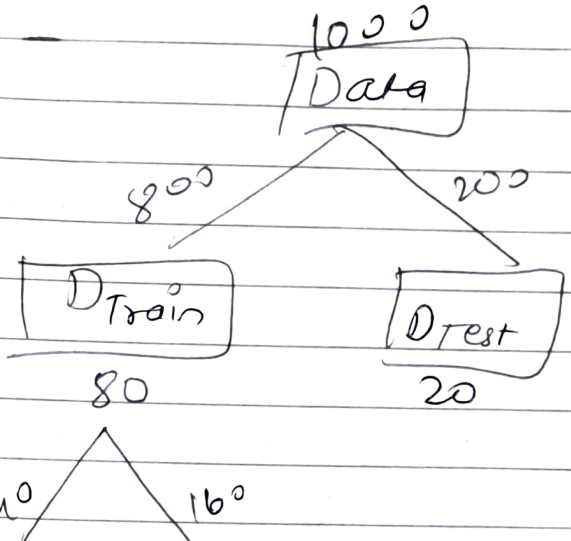
① K-Fold method → known as stacking.

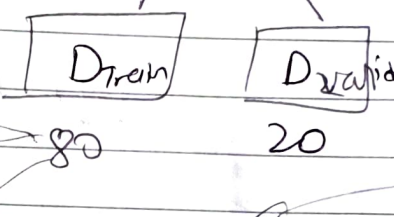
② Hold out method → known as blending



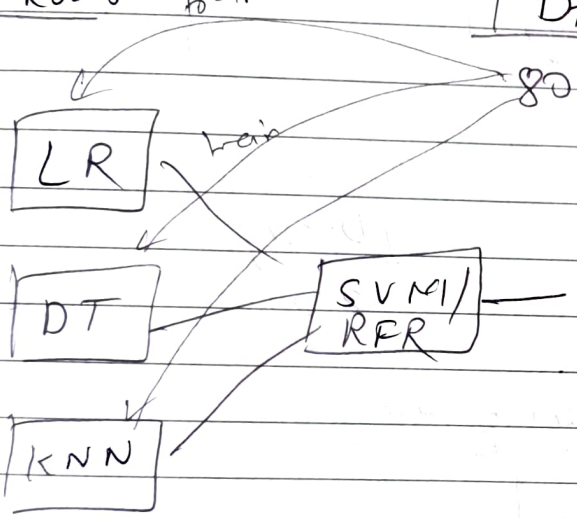
blending (hold out)

cgpa	iq	package
7	70	2.5
9	101	5.1
7.9	120	6.0
-	-	.





1000 Rows



LR will be tested on Dvalidation & OIP will be stored as new dataset.

→ Similarly for all the base models prediction will be done on Dvalidation

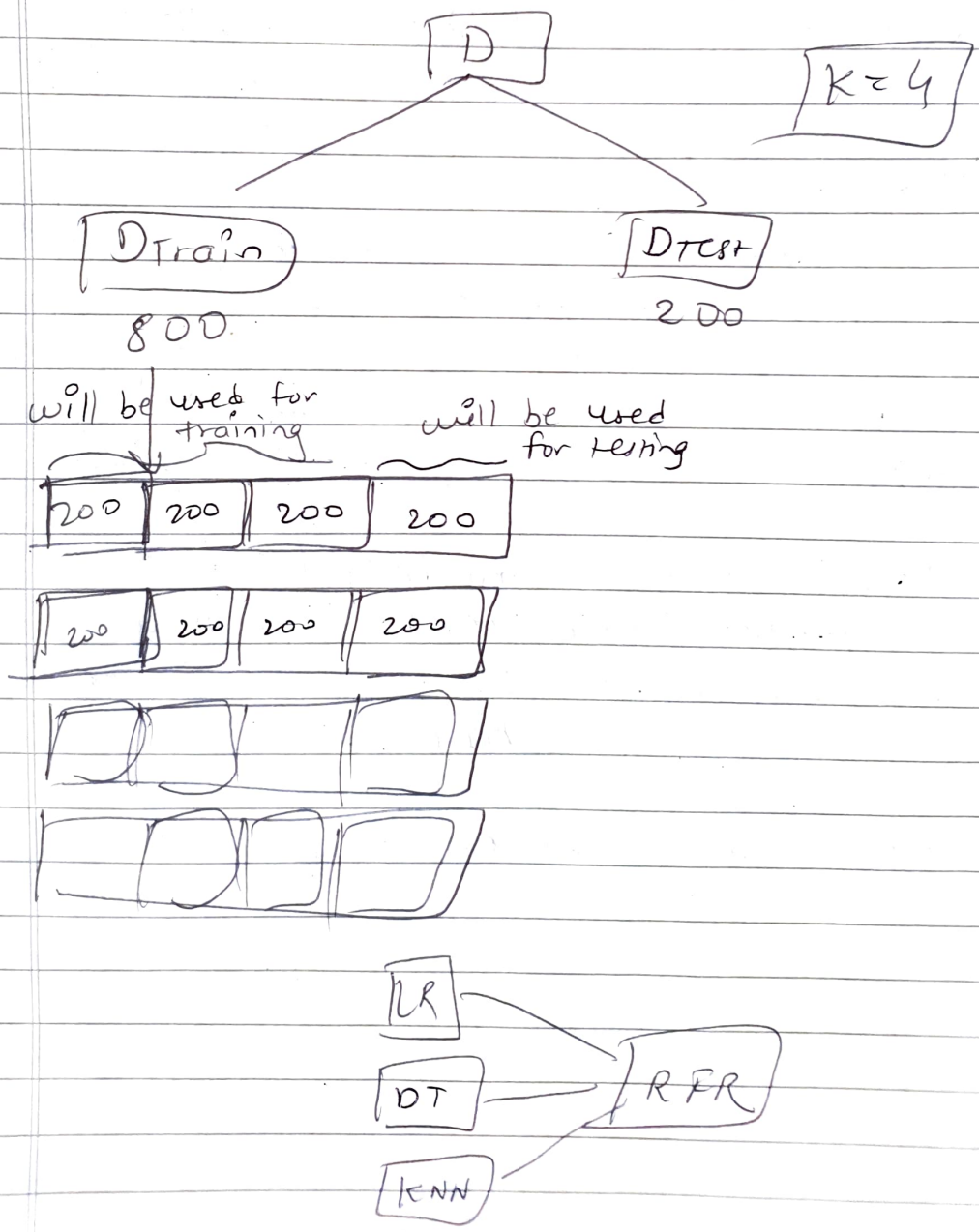
Step:
1. train 3 base models (3 trained on Dtrain)
2. Do the prediction on M1, M2 & M3 on Dvalidation
3. You form new dataset (using prediction of each base model)
4. train the meta model on Dtest (200 data)

② Stacking (K-Fold)

→ K is how many equal parts of training dataset.

Generally the value of K = 5 // K = 10.

→ for K more than 10, computation will be more hence not suggested.



Steps!

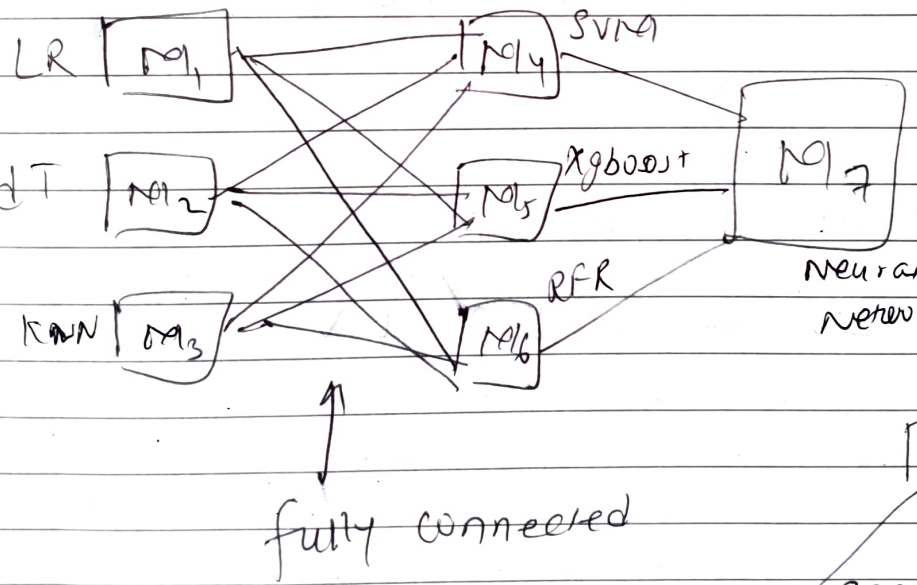
(1) while training for each ~~base model~~ K-fold, new dataset ~~output~~ will be (800, 4)

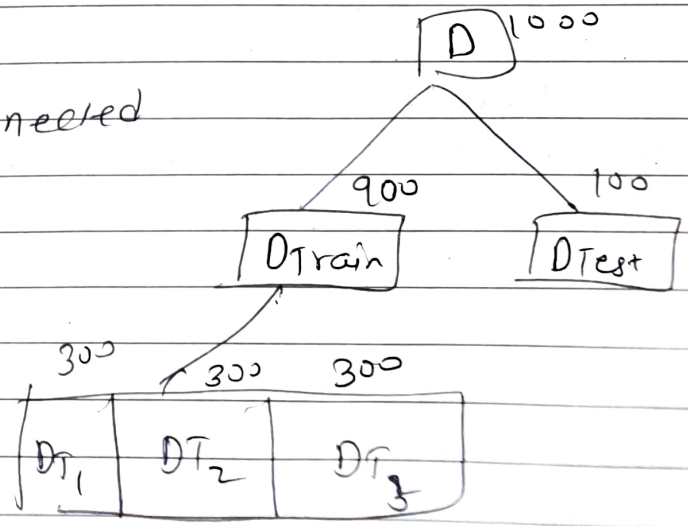
~~4~~th i.e. each base model will be trained K times (4 times in this case)

(2) and then this predicted data will be given as training data (input data) to Meta model.

(3) Retrain on X_train & Y_train to train base model.

Multi-layer stacking





① M_1, M_2 & M_3 trained on data dT_1

② prediction of M_1, M_2 & M_3 (i.e the dataset which we get after prediction) will be given as training data to M_4, M_5 & M_6.

③ prediction of M_4, M_5 & M_6 will be used to train M_7

Boosting

Bagging is nothing but conversion of LB & HV model into LB & LV

Similary,

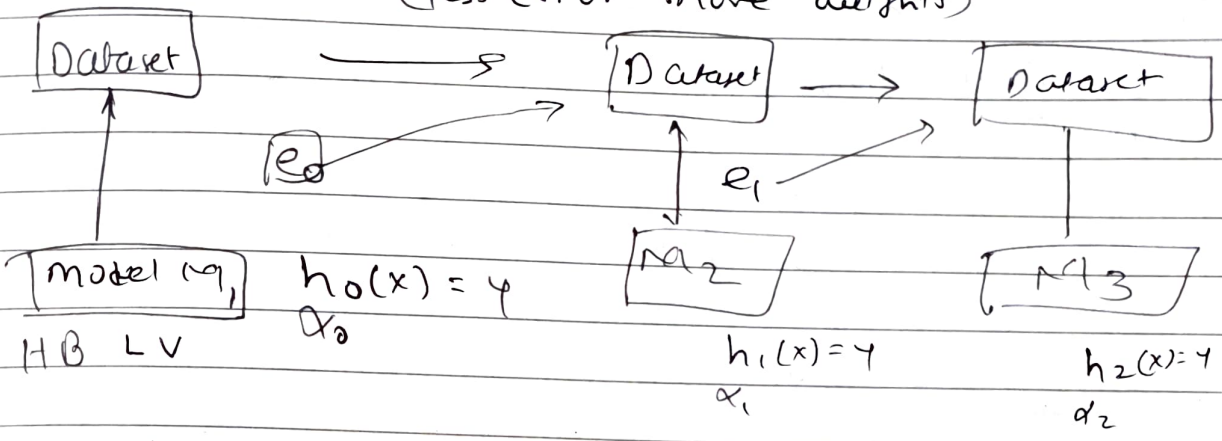
boosting is used to convert

HB LV ⟶ LB LV

↙

Decision tree of less depth.

(x, y) α = weight assign to model based on error. (less error more weights)



The points on which the error is coming will be boosted in the next step.

→ NO sampling is used in boosting

→ Depending on the dataset, boosting stages are defined. (n_estimator = 10/100/1000/---)

$$f = \alpha_0 h_0(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_n h_n(x).$$

Boosting techniques

1. Gradient boosting
2. Xg boost (Xtreme boost)
3. Adaboost (adaptive boosting)

Boosting techniques

Ada boost

Multiple weak learners are added together to form strong powerful model.

① **Weak learners :-**

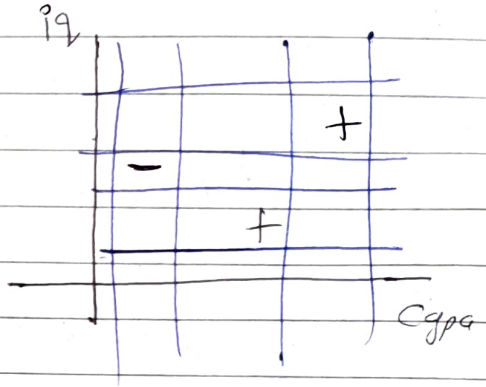
→ Generally their accuracies are less than 50% or just over 50%.

② **Decision stumps :-**

type of weak learner with maximum depth as 1.

for e.g.

x_1	x_2	y
6.4	79	1
4.9	81	0
9.1	102	1



→ Decision stumps gives better result in case of adaboost as weak learner.

③ **+1 & −1**

Binary classes of Adaboost.

+1 → +ve

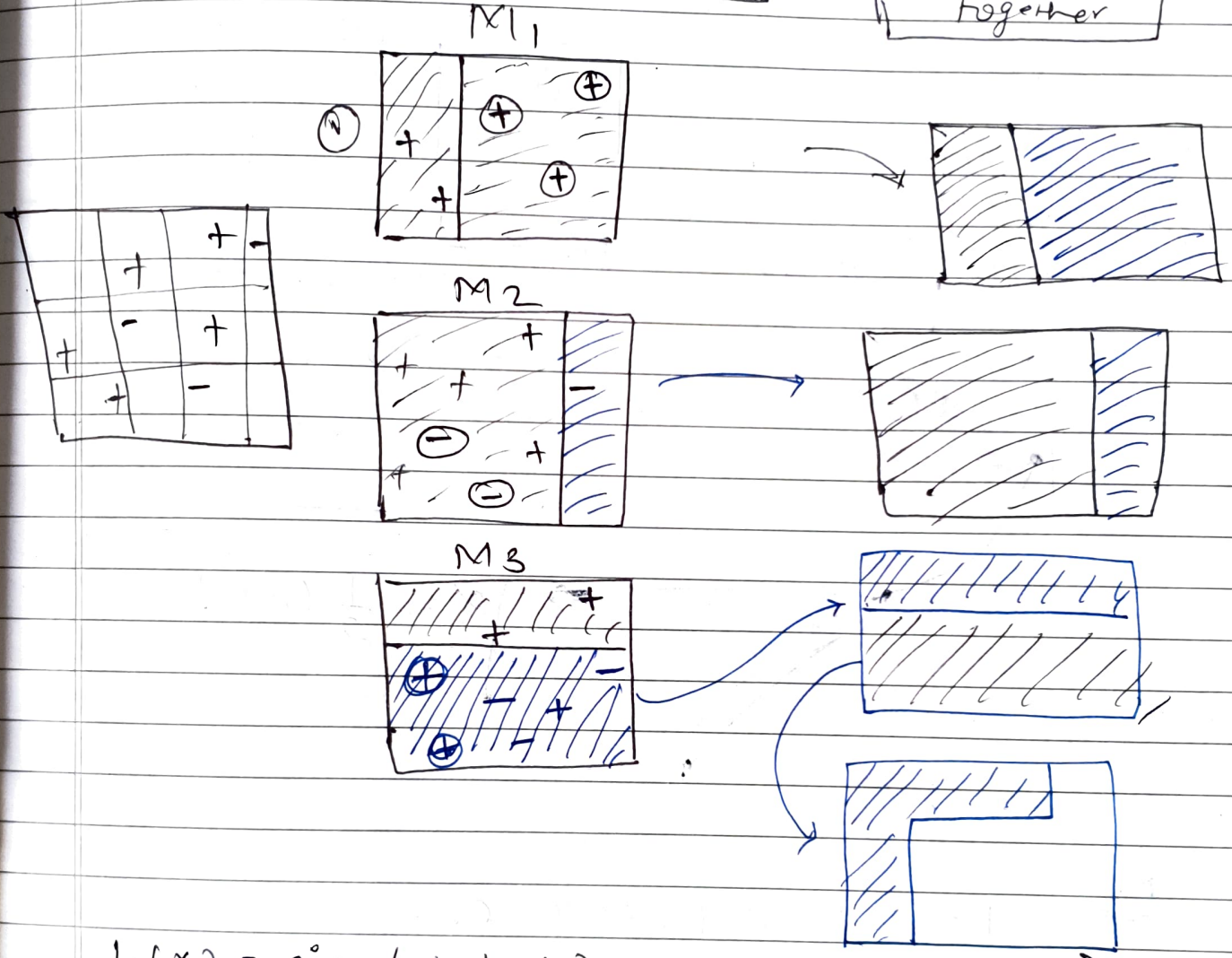
−1 → −ve

There are no zero's (0) is used in classification.

Geometric Intuition

PAGE No.
DATE : / /

* Adaboost is a stagewise additive method
  - stagewise → weak learners are added one by one
  - additive → Adding weak learners all together



$$h(x) = \text{sign}\left(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)\right)$$

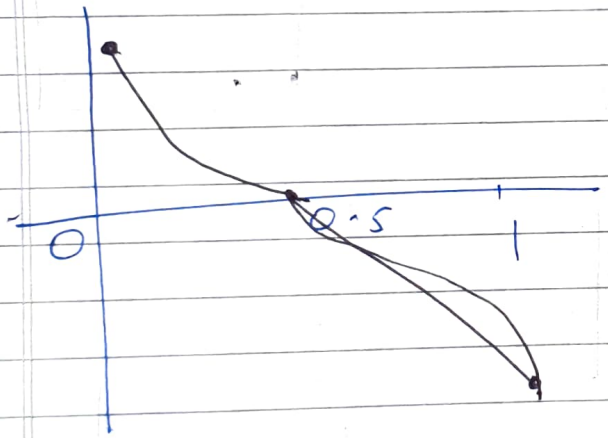
Adaboost – step by step

X_1	X_2	Y	Y-pred	weight	Updated wt	normalized wt
3	7	1	1	0.2	0.16	0.166
2	9	0	1	0.2	0.24	0.25
1	4	1	0	0.2	0.24	0.25
9	8	0	0	0.2	0.16	0.166
3	7	0	0	0.2	0.16	0.166

Stage 1

Initial weight assign $= \frac{1}{n}$ → no. of rows.

$= \frac{1}{5} = 0.2$



$\Rightarrow \boxed{\alpha = \frac{1}{2} \ln\left(\frac{1-\text{error}}{\text{error}}\right)}$

Now, how to calculate error of a model.

∴ error = sum of weight of all misclassified points.

∴ for M1

error = 0.2 + 0.2

= 0.4

$$\therefore \alpha_1 = \frac{1}{2} \ln\left(\frac{1-0.4}{0.4}\right)$$

$$= \frac{1}{2} \ln\left(\frac{0.6}{0.2}\right)$$

$$\boxed{\alpha_1 = 0.20}$$

stage 2:-

Now, M_1 has to inform M_2 that some mistake has been done so be careful.

∴ the weights of misclassified points will be [increased] → (boost)

And

the weights of correctly classified points will be reduced.

∴ For misclassified

$$\text{new-wt} = \text{curr-wt} \times e^{\alpha_1}$$

For correctly classified

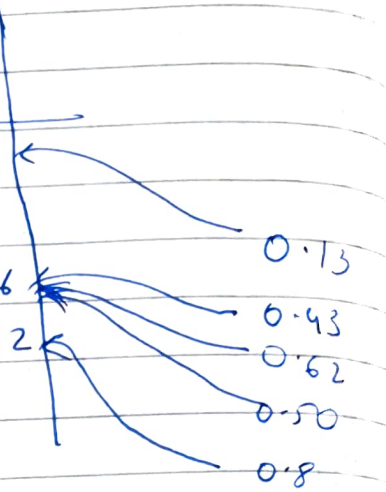
$$\text{new wt.} = \text{curr wt} \times e^{-\alpha_1}$$

∴ for previous table

$$\text{new-wt} = 0.2 \times e^{+0.2} = 0.24 \quad \text{(for misclassified)}$$

$$\text{new-wt} = 0.2 \times e^{-0.2} = 0.16 \quad \text{(for correctly classified)}$$

x_1	x_2	y	new. wt	range
3	7	1	0.166	0 - 0.166
2	9	0	0.25	0.166 - 0.416
1	4	1	~~0.166~~ 0.25	0.416 - 0.666
9	8	0	0.~~166~~ 666	0.666 - 0.832
3	7	0	0.166	0.832 - 1.0



for upsampling

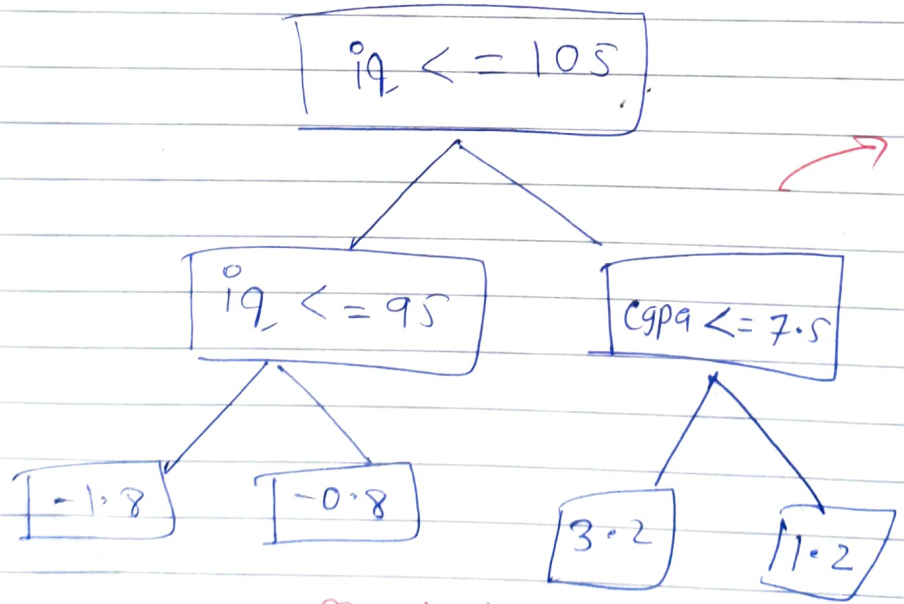
model will generate 5 random nos. (coz n=5) based on weightage and find ~~where~~ the rows for that each no. ~~con~~ depending on range.

After that new dataset will be prepared where row no. 1, row. no. 3 (3 times) and row no. 4 will be there.

Gradient Boosting (Regression)

x_1	x_2	y	M_1 Pred-1	Pseudo-residual = actual - pred = y - Pred-1	Pred-2	res-2	Pred 3	res-3
90	8	3	4.8	-1.8	-1.8	-1.62		
100	7	4	4.8	-0.8	-0.8	-0.72		
110	6	8	4.8	3.2	3.2	2.88		
120	9	6	4.8	1.2	1.2	1.08		
80	5	3	4.8	-1.8	-1.8	-1.62		
		24						

Note:- In Gradient boosting, for 1st model in case of regression is calculating simply the mean of output.



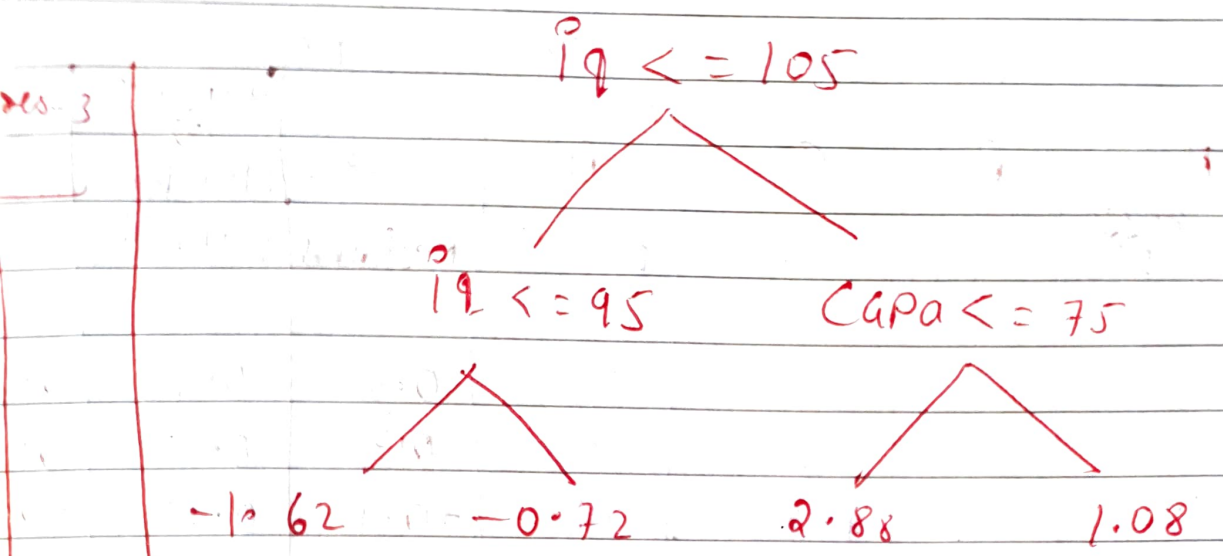
for M2 (M_2 is a Decision Tree).

y-pred = $m_1 + LR \times m_2$

$= m_1 + 0.1 \times m_2$

Res2 = Actual − (~~res~~ pred1 + 0.1 × Pred 2)
3

for M3



at this point,
prediction will be given as,

$$y\text{-pred} = \frac{m_1 + 0.1 \times M_2 + 0.1 \times m_3}{2}$$

Gradient boosting (classification)

→ It is similar as logistic Regression. (E)

→

	A	B	C	D		using eqn (2)	
	likes ~~ML~~ engg.	Age	fav. sub	likes offline lec?	(actual − Pred) Residual	New Pred − Probability	New Residual
①	Yes	12	ML	Yes	0.3	0.9	0.1
②	Yes	87	AI	Yes	0.3	0.5	0.5
③	NO	44	ML	No	−0.7	0.5	−0.5
④	Yes	19	DL	NO	−0.7	0.1	−0.1
⑤	NO	32	AI	Yes	0.3	0.9	0.1
⑥	NO	14	ML	Yes	0.3	0.9	0.1

① Initial prediction for every individual is the log(odds).

$$= \log\left(\frac{\text{total yes}}{\text{total NO}}\right)_0$$

$$= \log\left(\frac{4}{2}\right)$$

$$= 0.7 \text{ (approx)}$$

just like logistic regression, to use the log(odds) for classification is to convert it to a probability.

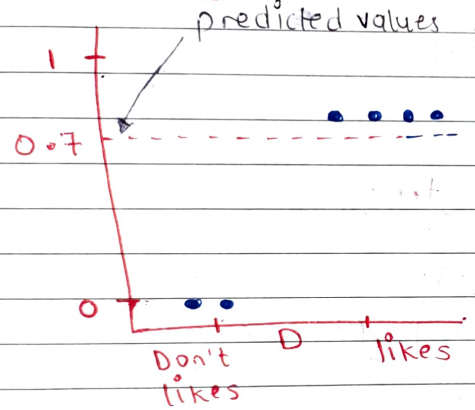
$$\therefore \text{probability of like offline lec} = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}}$$

$$= 0.7 \text{ (approx)}$$

Since probability of log(odds) is greater than 0.5, we can classify everyone (every row) in the training dataset as someone who loves (likes) offline lectures.

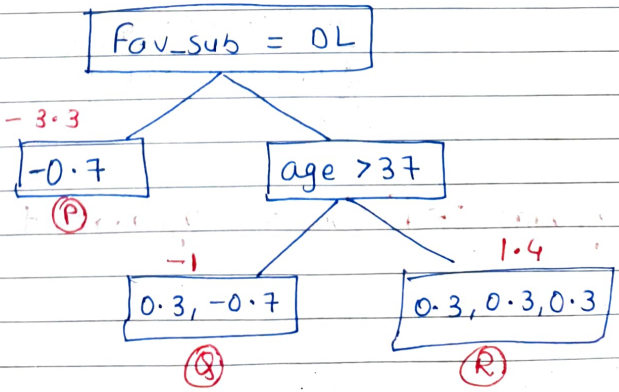
* Based on above conclusion (prediction) it is observed that ~~predi~~ initial prediction is not correct.

② Calculating residual = (Observed - Predicted)



(blue dotts are observed (actual) values).

③ Now, build a tree using column A, B, C to predict residuals.



Note:- while creating decision tree, maximum depth can vary from 8 to 32.

Now,
for Gradient boosting for classification, most common transformation formula is.

$$\text{O/P Value} = \frac{\sum \text{residual}_i}{\sum [\text{Previous probability}_i \times (1 - \text{Prev-probability})_i]} \quad - (1)$$

∴ for Row 4 (for (P)) [from Decision tree]

$$\text{o/p value} = \frac{-0.7}{0.7 \times (1-0.7)} = ?$$

$$= -3.3$$

for Q (from Decision tree)

$$= \frac{0.3 + (-0.7)}{[0.7 \times (1-0.7)][0.7 \times (1-0.7)]}$$

$$= -1$$

for (R) [from Decision tree]

$$= \frac{0.3 + 0.3 + 0.3}{3[0.7 \times (1-0.7)]}$$

$$= 1.4$$

(4) $\log(\text{odds})\ \text{prediction} = \log(\text{odds}) + \lambda \times (\text{o/p value from tree formed (1)})$ — (2)

for e.g.:
for Row 1:-

$\log(\text{odds})\ \text{prediction} = 0.7 + 0.8 \times 1.4$

$= 1.8$

[$\lambda = 0.8$ for illustration purpose. Generally λ is 0.1]

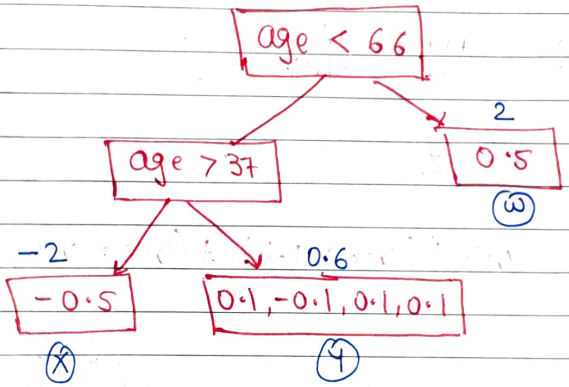
PAGE No:
DATE: / /

Now,

converting new log(odds) prediction into probability

$$\text{probability log(odds)} = \frac{e^{1.8}}{1+e^{1.8}} = 0.9$$

Now, based on column A, B, C and E(new-residual) we create a new tree.



Next,

Calculating o/p value for each leaf node (W, X, Y)

① for W:-

$$= \frac{0.5}{0.5(1-0.5)} = \boxed{2}$$

② for X :-

$$= \frac{-0.5}{0.5 \times (1-0.5)} = \boxed{-2}$$

③ for Y :-

$$= \frac{-0.1 + 0.1 + 0.1 + 0.1}{[0.9(1-0.9)][0.1 \times (1-0.1)][2 \times 0.9 \times (1-0.9)]}$$

$$= \boxed{0.6}$$

Now, again,

log(odds) prediction = log(odds) + λ × (O/P values from tree 2)

* Prediction *.

→ let's for an input,

likes engg.	age	fav. sub.	likes offline lec?
yes	25	AI	???

→ log(odds) prediction = $0.7 + (0.8 \times 1.4) + (0.8 \times 0.6) = 2.3$

- 0.7 → log(odds)
- $\lambda_1 \times$ O/P value of ⑧ from tree 1 → (0.8×1.4)
- $\lambda_2 \times$ O/P value of ④ from tree 2 → (0.8×0.6)

$\therefore \log(odds)_{prediction} = 2.3$

Now, we convert this log(odds) into a probability.

$\therefore$ probability $= \frac{e^{2.3}}{1 + e^{2.3}}$

$= 0.9$ (approx)

Since, threshold value is 0.5 ~~which~~

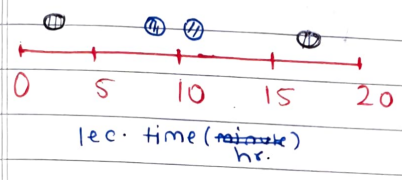
$\therefore$ Class of the given i/p data will be Yes. $(0.9 > 0.5)$

Xgboost [Extreme Boosting] ⇒ classification

PAGE No.
DATE : / /

→ Xgboost is mostly suitable for large, complicated datasets.

→

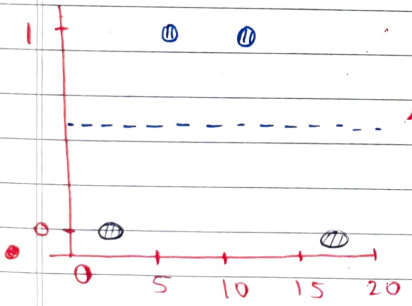


→ blue dots denotes that ML lecture was effective.

→ black dots represents that ML lecture was not effective.

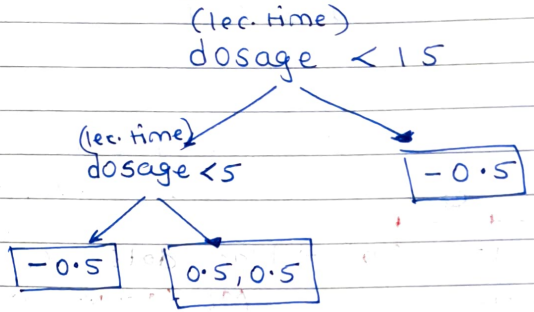
Step 1: Making initial prediction. [This value can be anything but by default it is 0.5 for classification & Regression both]

Now,
illustrating the initial prediction by adding a y-axis to our graph to represent the ~~proj~~ probability that ML lecture is effective.



Now,
Calculate residual = actual - predicted just to check ~~whether~~ how good the initial prediction is.

Now fit an xgboost tree to the residuals.



Next step is to calculate the similarity scores.

$$\text{similarity scores} = \frac{\left[\Sigma(\text{Residual})_i\right]^2}{\Sigma\{[\text{previous probability}]_i \times (1-(\text{prev-probability})_i)\} + \lambda}$$

$$= \frac{(\text{sum of residuals})^2}{[\text{Number of residuals}] + \lambda}$$

for creating a tree

→ Each tree starts out as an single leaf. and all of the residual go to the leaf.

-0.5, 0.5, 0.5, -0.5

→ Next, calculate the similarity score for the leaf.

$$= \frac{(-0.5 + 0.5 + 0.5 - 0.5)^2}{\text{Denominator}}$$

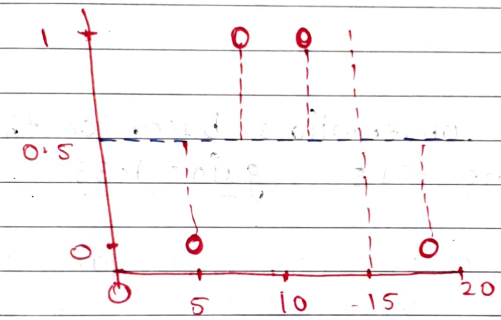
$$= 0$$

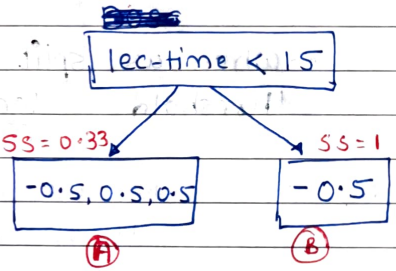
$-0.5, 0.5, 0.5, -0.5$ similarity = 0

Now,
we need to decide if we can do a better job - clustering similar residuals if we split them into two groups.

for e.g.: let's start with this threshold,
lec-time < 15.

the reason we chose 15 because it is the avg. value betn last 2 observations.





Now, Calculate the similarity score of the leaf on the left and then on the right respectively.

* Similarity of left leaf: (A) :-

$$= \frac{(-0.5 + 0.5 + 0.5)^2}{[0.5 \times (1-0.5)][0.5 \times (1-0.5)][0.5 \times (1-0.5)] + \lambda}$$

↳ [Note: Initial prediction was 0.5] ($\lambda = 0$)

$$= 0.33$$

* similarly, similarity score for right leaf (B):-

$$= \frac{(0.5)^2}{0.5 \times (1-0.5) + \lambda} = 1 \text{ (when } \lambda = 0)$$

Now, for simplicity we'll put $\lambda = 0$ but it is known that λ reduces the similarity score which makes leaves easier to prune.

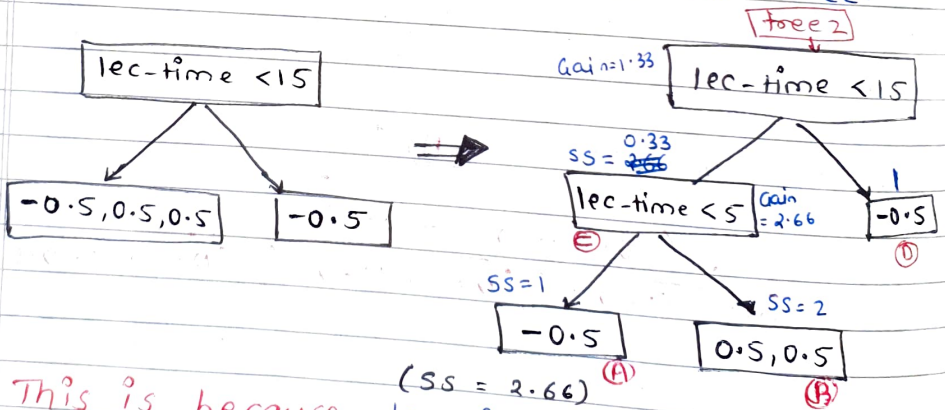
After calculating the similarity score, we need to calculate the ~~Gini~~ Information gain

> Gain = Left similarity + Right similarity − Root similarity

∴ Gain of tree formed = 0.33 + 1 − 0
= 1.33

∴ when we split the observations based on the threshold lec-time < 15, gain = 1.33

★★★ Since, No other threshold will give gain greater than lec-time < 15, Hence lec-time < 15 will be the first branch of the tree.



(SS = 2.66)

This is because lec-time < 5 has a higher gain than the lec-time < 10.
(SS = 0.66)

PAGE No.
DATE / /

Note:- Just for simplicity we are not splitting any leaf further.
i.e. limited the tree as max level = 2

** The minimum number of residuals in each leaf is determined by cover.

Cover:- Cover is defined as diff. bet^n denominator of similarity score and λ.

$$\therefore \text{cover} = \sum [\text{Prev\_probability}_i \times (1 - \text{prev\_probability}_i)] - \lambda.$$

By default, the minimum value for cover is +1.

Now,

for xgboost classification, cover depends on the previously predicted probability of each residual in a leaf.

for e.g. in tree 2, for leaf node A,

$$\text{cover} = 0.5 \times (1-0.5) = 0.25$$

Similary, in tree 2, cover for leaf node B is,

$$\text{cover} = 0.5 \times (1-0.5) + 0.5 \times (1-0.5) = 0.5$$

since, default value for cover is 1 (min.), xgboost would not allow the leaf (A) & (B).

Hence it will be pruned.

Ultimately, if we use the default minimum value for cover as 1, then we would be left with the root, and xgboost requires trees to be larger than just the root.

Hence, in order to have a tree, we'll assume the minimum value for cover as zero (0).

in program:- `cover(min_child_weight) = 0`

Condition for pruning:-

$$\text{Gain} - \gamma = \begin{cases} \text{if positive, then do not prune} \\ \text{if negative, then prune.} \end{cases}$$

(γ = tree complexity parameter)

more the no. of γ, less will be the difference (negative result). Hence causes to prune the branches.

Hence value of γ and λ will be selected as zero in this example for simplification.

For now,

regardless of λ and γ, let's assume that this is the tree (tree ②) ~~is the~~ we are working with.

PAGE No.
DATE : / /

* Determining O/P values for the ~~leaf~~ leaves. (Tree ②)

$$\text{O/P values} = \frac{\sum \text{residuals}_i}{\sum [\text{prev-probability}_i \times (1 - \text{prev-probability}_i)] + \lambda}$$

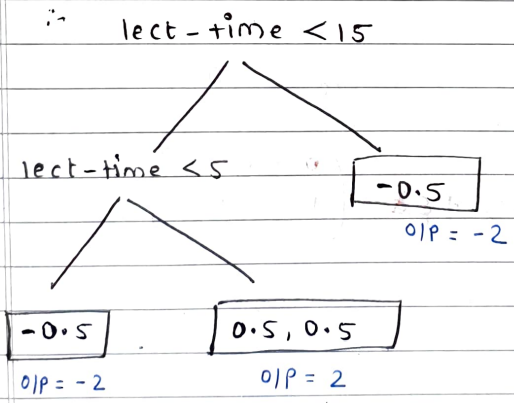
λ = Regularization parameter

∴ O/P value for leaf A (in tree ②)

$$= \frac{-0.5}{0.5 \times (1 - 0.5) + 0} = -2$$

(considering $\lambda = 0$ in this example)

∴



∴ 1st tree is completed after calculating the O/P value of each leaf.

Now, based on the tree formed, new pred will be done.

→ Just like other boosting methods, Xgboost classification makes new predictions by starting with the initial prediction.

Now, calculating log(odds).

$\therefore \log(\text{odds}) = \log\left(\frac{p}{1-p}\right)$

$= \log\left(\frac{0.5}{1-0.5}\right)$ - [∵ initial prediction was 0.5].

$= \log(1)$

$= 0$

→ for Xgboost classification, learning rate (ε (Eta)) is used.

The default value for ε is 0.3.

$\therefore$ prediction = log(odds) + ε × (O/P values from the tree formed)

let's calculate for leaf-time = 2

= log(odds) prediction

= 0 + 0.3 × (−2)

= −0.6

→ To convert log(odds) value into probability, we use logistic function (sigmoid function).

$$\therefore \log(\text{odds}) \text{ probability} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

$$= \frac{e^{-0.6}}{1 + e^{-0.6}}$$

$$= \boxed{0.35}$$

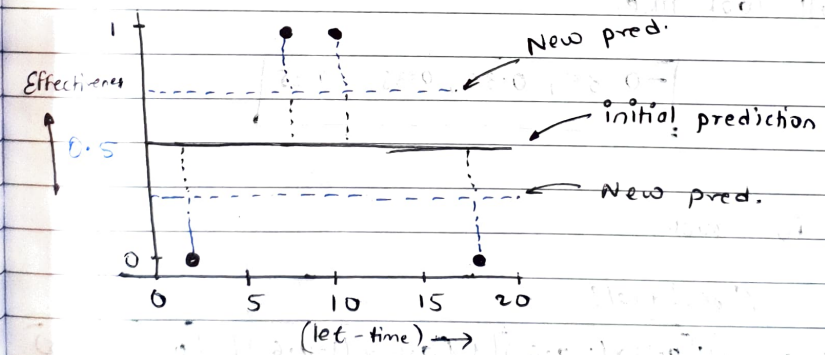
similarly, we'll calculate the log(odds) probability for each observation.

Calculating log(odd) prediction For lect-time = 8

$$= 0 + 0.3 \times 2$$

$$= 0.6$$

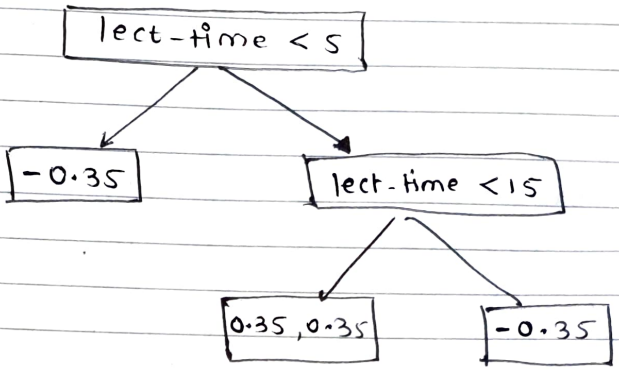
$$\therefore \log(\text{odds}) \text{ probability} = \frac{e^{0.6}}{1 + e^{0.6}} = \boxed{0.65}$$



conclusion:- By looking at above graph, it is observed that newly calculated residual is smaller for each observation than the initially calculated value of residual.

Since, less the residual more is the accuracy.
∴ Xgboost is enhancing step by step.

Now, based on the new residual we can fit a new tree.



Note:-

while building the new tree, calculating the similarity score is more interesting because previous probabilities are no longer same for

∴ root will look like,

-0.35, 0.35, 0.35, -0.35

O/P value for root,

$$= \frac{\Sigma \text{residual}_i}{[0.35 \times (1-0.35)] + [0.65 \times (1-0.65)] + [0.65 \times (1-0.65)] + [0.35 \times (1-0.35)]}$$

Random Forest.

PAGE No:
DATE: / /

Random forest with gradient boosting or xgboosting is one of the most widely used algorithm in ML.

→ Applicable for classification & Regression.

→ **Prerequisites**

1. Decision tree
2. Bagging Technique
3.

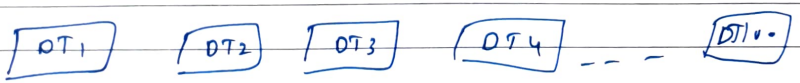
Q. Why Random forest ?

↳ multiple ~~trees~~ ~~one~~ Decision trees are trained together

It is a bagging technique only. But when decision tree is used as a base model in bagging then it becomes Random forest.

∴ Random forest is specialization of bagging technique.

Dataset = 1000 Rows



→ Row sampling
(Feature sampling) → Col^n sampling
→ Combination

} (with / without Replacements)

Bagging vs Random Forest

	Bagging	RF
①	Diff. algorithms can be used. ex. KNN, DT, SVM	Only DT is used as a base model.
②	Tree level col^{n} sampling	Node level sampling

∴ RF is having better performance than Bagging.