Introduction to Boosting

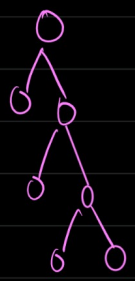
Ensembles

(Bootstrap Aggregation) (Base learner) Bagging → Random forest classifier → Random forest regressor.

Boosting
1. AdaBoost
2. Gradient Boosting
3. XG Boost

{Bagging: base learner (Decision Trees)
Boosting: weak learners (Decision trees)

⇒ for both regression & Classification problem statement.



DT → lead to overfitting (high variance)

⇓

To reduce this

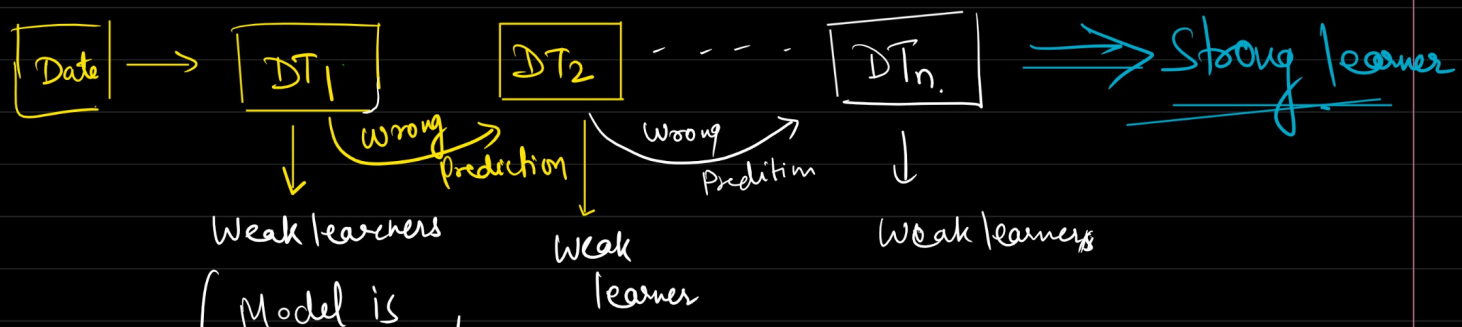
Bagging → Random forest

* Bagging reduces the variance

(multiple overfitted model combined reduces the variance)

* Boosting (Boost ಕನ್ನಡ)

Principles:- Build a first model on training dataset and then build a second model to rectify the errors present in the first model. This procedure is continued until and unless the errors are minimised | data is correctly predicted.



(Model is very simple / Not trained properly — not learned much from data)

Analogy :-

→ Football → pass the ball

→ easy to break or stick but hard for a group.

* Prediction

Bagging

→ classification: Voting

→ Regression: Average.

Boosting

$M_1 \rightarrow M_2 \rightarrow M_3 \cdots M_n$

$\alpha_1(M_1) + \alpha_2(M_2) + \cdots \alpha_n(M_n)$

$\alpha_1, \alpha_2 \cdots \alpha_n \Rightarrow$ weights

Strong learners

Boosting is a machine learning ensemble technique that improves model accuracy by combining multiple weak learners, typically decision trees. It operates sequentially, where each model corrects the errors of its predecessor by giving more weight to misclassified instances. Popular algorithms include AdaBoost, Gradient Boosting, XGBoost, LightGBM, and CatBoost. Boosting's strength lies in its ability to enhance predictive performance, making it effective for various applications. However, it requires careful hyperparameter tuning to avoid overfitting. Boosting is widely used due to its capacity to create robust models from relatively simple learners.

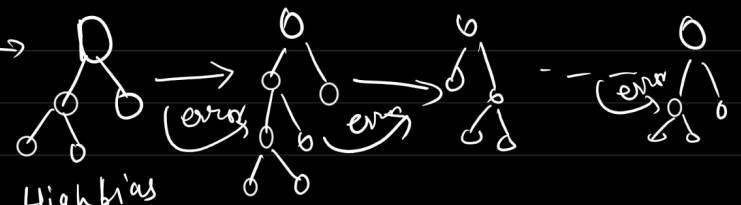
AdaBoost (Adaptive Boosting)

by Yoav Freund and Robert Schapire in 1995.

* Works both for classification & regression problem.

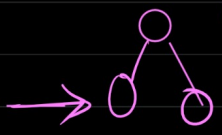
DT → High variance → Bagging → Random Forest → It reduces the high variance of individual decision trees.

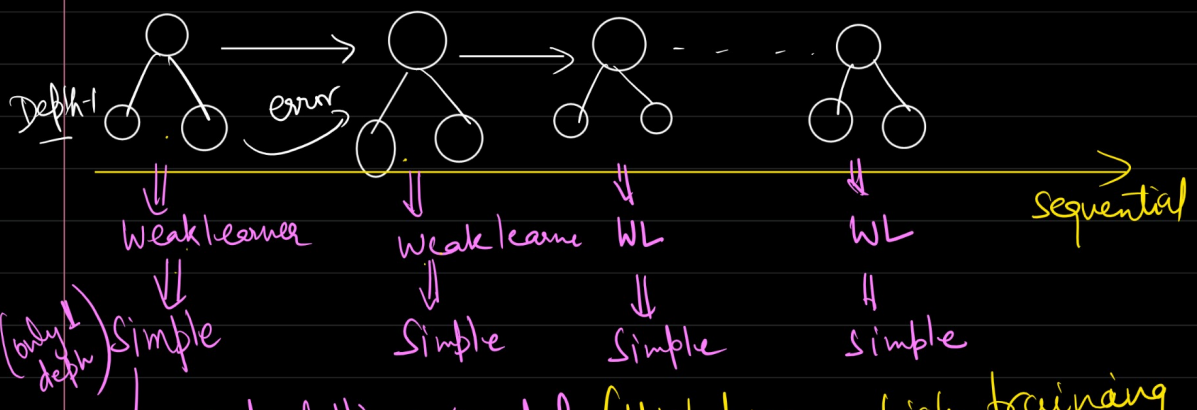
* Boosting →



* AdaBoost follows the concept of boosting.

Decision Stumps :- A decision tree with only one level of depth





↳ Underfitting model (High bias — high training Error)

↳ train acc ↓ → High bias

* Each individual DT is a high bias model but error is transferred to next model leading to low bias at an overall level.

→ AdaBoost solves the problem of high bias.

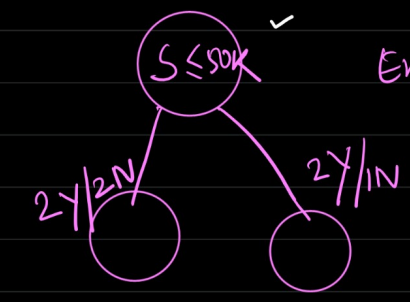
→ AdaBoost is low bias model.

→ we always want low bias & low variance model.

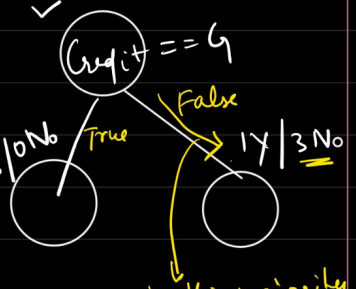
* Mathematical explanation of AdaBoost

Step-1 → Create decision tree stump.

Salary	Credit Score	Approval
→ <=50k	B	No
<=50k	G	Yes
<=50k	G	Yes
>50k	B	No
>50k	G	Yes
>50k	N	Yes
<=50k	N	No



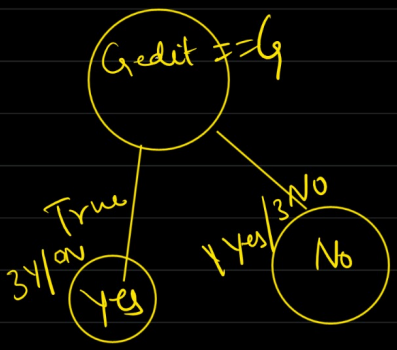
Entropy or Gini impurity



In this majority is No, that's Credit score != G, will be predicted as No but 2nd last row is predicted Yes, this is wrong prediction.

Step-2 → Assign equal weights to all of the datapoints and predict using the higher IG Decision Stump

Salary	Credit Score	Approval	Sample 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



Step-3 → Total Error and performance of stump.

Total Error = 1/7

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

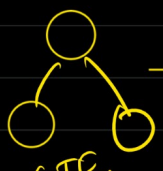
$$= \frac{1}{2} \ln\left[\frac{1-1/7}{1/7}\right]$$

$$= \frac{1}{2} \ln[6] \approx 0.896$$

↳ base to e

$$f = \alpha_1 M_1 + \alpha_2 M_2 + \cdots \alpha_n M_n$$

↳ $\alpha_1 = 0.896$ → weight of first decision stump.



→ next step is to give more priority to wrong prediction.

Salary	Credit Score	Approval	Sample 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

Step 4 → Give more weights to incorrect classified datapoint and lesser weight to correct datapoint

* For correct classified dp's

= weight * $e^{-\text{Performance of stump}}$

$= 1/7 \times e^{-0.896} = 0.058$ $e = 2.73$

* Incorrect classified dp -

= weight $\times e^{+\text{Performance of stump}}$

$= \frac{1}{7} * e^{0.896} = 0.349$

$\ominus 5$ (Correct ones)

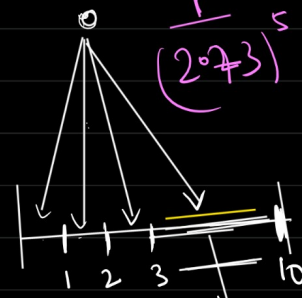
$(2.73)^5$ $(2.73)^1$ $\frac{1}{(2.73)^5}$

Step-5 Normalized wt & Bin assignment.

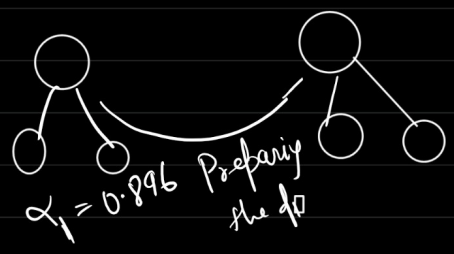
Salary	Credit Score	Approval	SW (initial)	Updated wt	Normalized wt	Bin assignment
<=50K	B	No	1/7	0.058	0.08	0-0.08
<=50K	G	Yes	1/7	0.058	0.08	0.08-0.16
<=50K	G	Yes	1/7	0.058	0.08	0.16-0.24
>50K	B	No	1/7	0.058	0.08	0.24-0.32
>50K	G	Yes	1/7	0.058	0.08	0.32-0.40
>50K	N	Yes	1/7	0.349	0.50	0.40-0.90
<=50K	N	No	1/7	0.058	0.08	0.90-0.98
					≈ 1	

total weight = 0.697

Here total weight is not 1, then we have to normalize



Higher Prob of Stone falling in this region



Step-6 Generate a random no between 0 to 1

0.3
0.4
0.5
0.4
0.6
0.2
0.7
0.8
0.9

Salary	Credit Score	Approval	SW	Updated wt	Normalized wt	Bin assignment
<=50K	B	No	1/7	0.058	0.08	0-0.08
<=50K	G	Yes	1/7	0.058	0.08	0.08-0.16
<=50K	G	Yes	1/7	0.058	0.08	0.16-0.24
>50K	B	No	1/7	0.058	0.08	0.24-0.32
>50K	G	Yes	1/7	0.058	0.08	0.32-0.40
>50K	N	Yes	1/7	0.349	0.50	0.40-0.90
<=50K	N	No	1/7	0.058	0.08	0.90-0.98
					≈ 1	

Salary	Credit score	Approval	Bin assignment
<=50k	B	No	0–0.08
<=50k	G	Yes	0.08–0.16
<=50k	G	Yes	0.16–0.24
>50k	B	No	.24–0.32
>50k	G	Yes	.32–0.40
>50k	N	Yes	0.40–0.90
<=50k	N	No	0.90–0.98

1st 2nd 3rd 4th

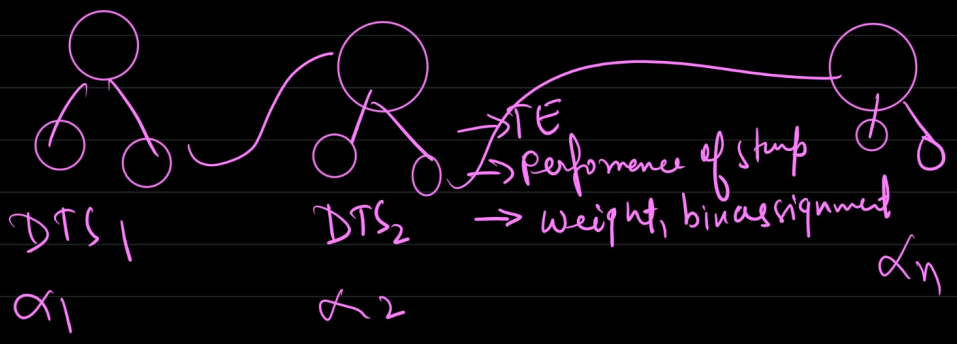
rn between

0.4
0.5
0.4
0.6
0.2
0.7
0.8
0.9

* These records will be sent to next Decision Stump

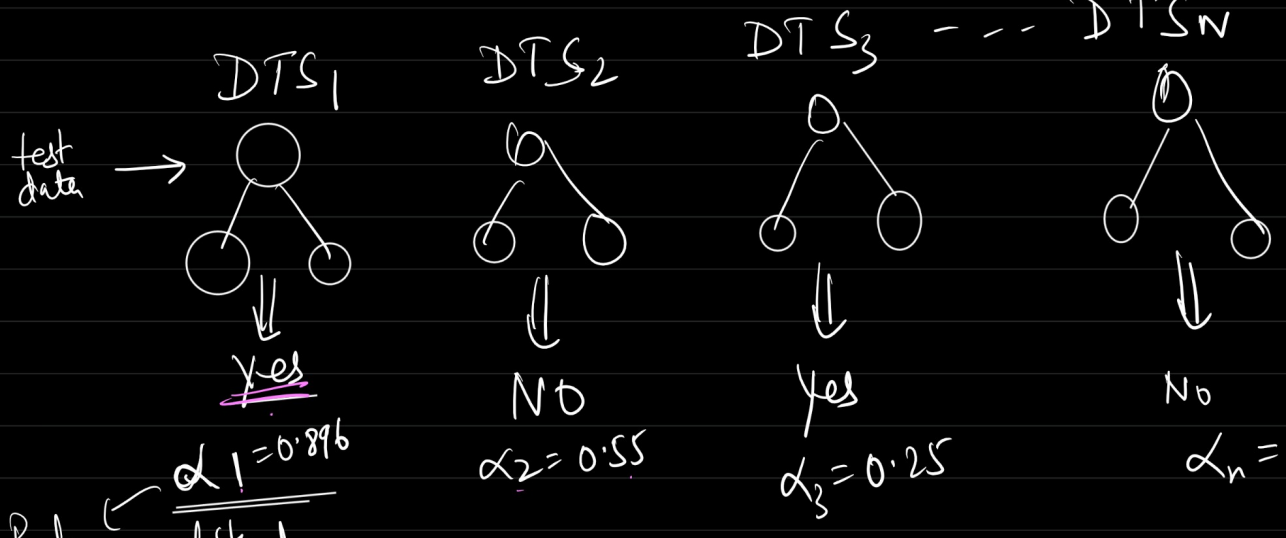
>50k	N	Yes
>50k	N	Yes
<50k	G	Yes
<50	G	Yes
>50k	N	Yes

* Note: → The row with higher bin range due to misclassification can be repeat ⇒ more priority to misclassified dp.



* Final Prediction

Test data (<=50k, G)



$$f = \alpha_1(M_1) + \alpha_2(M_2) + \alpha_3(M_3) \cdots \alpha_4(M_4)$$

$$= 0.88(\text{Yes}) + 0.55(\text{No}) + 0.25(\text{Yes}) \cdots 0.30(\text{No})$$

$$= 1.15(\text{Yes}) + 0.25(\text{No}) \rightarrow \text{Performance of No} = 0.25$$

$\Downarrow$

Performance of Yes = 1.15

final output $\Rightarrow$ Yes.

* multiclass $\rightarrow$ Same method $\rightarrow$ there will three Aggregated alphas.

* AdaBoost Regressor

↳ All the steps will be same except instead of Information gain, you will use variance reduction to select the decision tree stumps.

↳ during prediction $f = \alpha_1 M_1 + \alpha_2 M_2 + \cdots \alpha_n M_n$

$M_1, M_2 \cdots M_n \Rightarrow$ will be a Continuous Value.

AdaBoost (Adaptive Boosting) is an ensemble learning algorithm that combines multiple weak classifiers to form a strong classifier. It iteratively adjusts the weights of training data points, emphasizing those misclassified by previous classifiers. Each weak classifier is trained to correct errors from the previous ones, and their contributions are weighted based on accuracy. The final model is a weighted vote of these classifiers. AdaBoost is known for improving classification accuracy and reducing overfitting, but it can be sensitive to noisy data and requires more computational resources due to multiple training iterations.

Steps of AdaBoost Algorithm:
Initialize weights: Assign equal weights to all training examples.
Train weak classifier: Train a weak classifier on the weighted training data.
Evaluate classifier: Calculate the classifier's error rate.
Update weights: Increase the weights of misclassified examples and decrease the weights of correctly classified ones.
Calculate classifier weight: Determine the weight of the classifier based on its accuracy.
Combine classifiers: Add the weak classifier to the final model, weighted by its accuracy.
Repeat: Repeat the process for a specified number of iterations or until the model achieves a desired level of accuracy.

Gradient Boosting Algorithms

Gradient boosting is a machine learning technique used for regression and classification. It builds models iteratively by combining multiple weak learners, typically decision trees, to form a strong predictive model. Each new model is trained to correct the errors of the combined previous models by minimizing a loss function, often using gradient descent. The process involves adding the new model's predictions to the existing model with a controlled learning rate. Gradient boosting is known for its high accuracy and ability to handle a variety of data types but can be computationally intensive and prone to overfitting without proper regularization.

① Regression

② classification

Dataset

assume dt1 predicted and then we have calculated residual of it

Exp (x_1)	Degree (x_2)	Salary (y)	$R_1 = (y - \hat{y})$	Predicted	R_2	$\hat{y}$	R_3
→ 2	B.E	50K	-25K	-23K		74.77	-24.77
→ 3	Masters	70K	-5K	-3K		74.97	-4.97
5	Masters	80K	5K	3		-	-
6	PHD	100K	25K	20K		-	-
		75K					

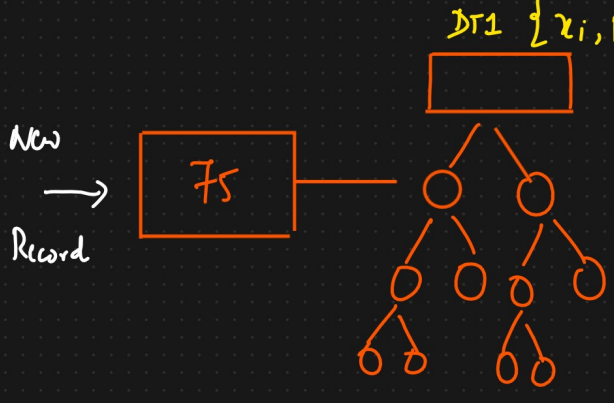
Steps

① Create a Base Model

$$\text{Average} = \frac{[50K + 70K + 80K + 100K]}{4} = 75K$$

② Compute Residuals, Error

③ Construct a Decision Tree consider inputs x_i and o/p R_1



xi is experience and degree

Base Model

Predicted o/p : 75 + (-23) = 75 − 23 = 52 {Overfitting}.

(DT1)

Predicted o/p : $75 + \alpha(-23)$

α = Learning Rate

$\alpha = 0.01 \rightarrow \{0 \text{ to } 1\}$

$= 75 + (0.01)(-23)$

$= 75 - 0.23$

$= 74.77$

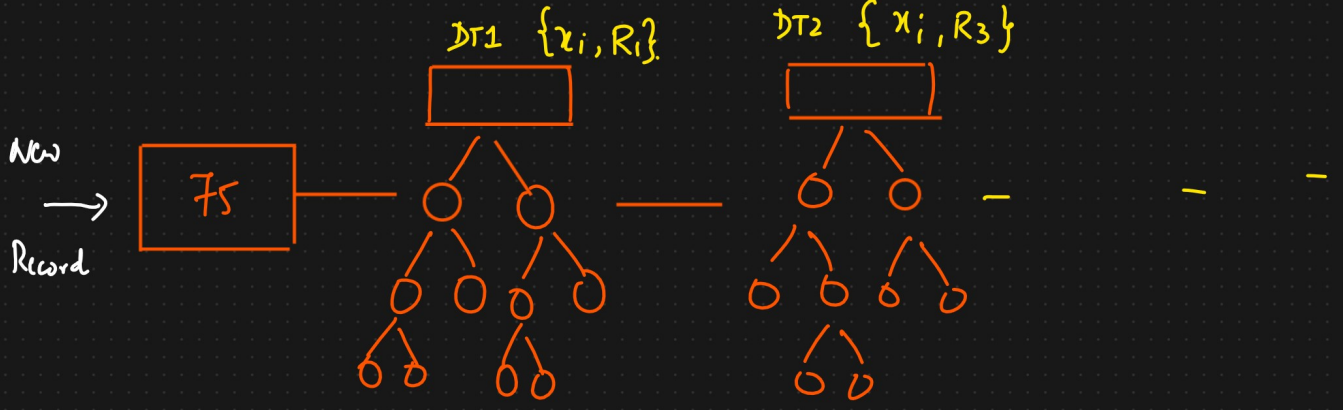
Learning Rate: A scaling factor to control the contribution of each weak learner.

② Predicted o/p = $75 + \alpha(-3k)$

$= 75 + (0.01)(-3)$

$= 75 - 0.03$

$= 74.97$ //



Base Model

o/p = $75 + \alpha_1(DT_1) + \alpha_2(DT_2) + \cdots \alpha_n(DT_n)$.

Mathematical

$\alpha_0 = 1$

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

$\{\alpha_0, \alpha_1, \alpha_2, \cdots \alpha_n\} \rightarrow$ Learning Rate $\{0 \text{ to } 1\}$

$$F(x) = \sum_{i=0}^{n} \alpha_i h_i(x)$$

Initialization: Start with a base model, usually a simple model like the mean of the target variable for regression tasks.

Iterative Improvement: In each iteration, a new weak learner (typically a decision tree) is trained to predict the residuals (errors) of the combined model from the previous iteration.

Update Model: The predictions of the new learner are added to the model with a certain weight (learning rate) to update the combined model.

XG Boost (Extreme Gradient Boosting)

→ XG Boost classifier
→ XG Boost Regressor

* XG Boost classifier:

Salary	Credit Score	Approval	Base model ($y_{base}P$)	R_1 ($y_{act} - y_{base}P$)
→ <=50k	B	0	0.5	−0.5
<=50k	G	1	0.5	0.5
<=50k	G	1	0.5	0.5
>50k	B	0	0.5	−0.5
>50k	G	1	0.5	0.5
>50k	N	1	0.5	0.5
<=50k	N	0	0.5	−0.5

(y_{act}) ($y_{base}P$)

Step-1 → Create a base model

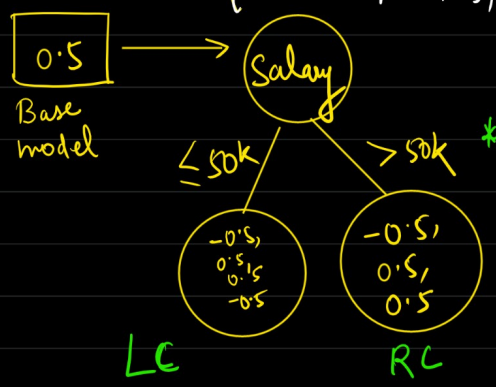
*Since base model should be unbiased, take 0.5 here

Step-2 - Construct a decision tree with root

Step-3 Calculate similarity weights

$$\text{Similarity weight} = \frac{\sum (\text{Residual})^2}{\sum Pr(1-Pr) + \lambda = 0}$$

[−0.5, 0.5, 0.5, −0.5, 0.5, 0.5, −0.5]



* $SW_{root} = \frac{[-0.5+0.5+0.5-0.5+0.5+0.5-0.5]^2}{0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)}$

Step4: Highest Similarity weight gain

$$= \frac{(0.5)^2}{0.25+0.25+0.25+0.25+0.25+0.25+0.25} = \frac{0.25}{1.75} = 0.14$$

* SW Left child $= \frac{(-0.5+0.5+0.5-0.5)}{0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)} = 0$

* SW Right child $= \frac{(-0.5+0.5+0.5)^2}{0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)}$

$$= \frac{0.25}{0.75} = \frac{1}{3} = 0.33.$$

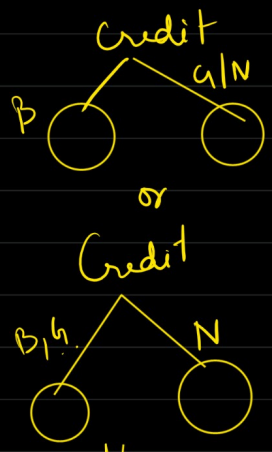
$$\text{SW gain} = (\text{Left child} + \text{Right child}) - \text{Root}$$

$$= 0 + 0.33 - 0.14 = 0.19$$

* Whichever feature has the highest Similarity weight gain, the split will happen on that feature!

* Here assuming Salary column is giving similarity weight gain

Two possibilities



or

based on similarity gain Score

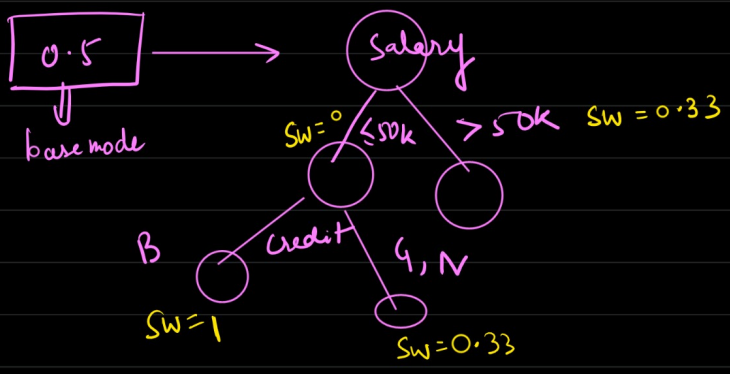
$$SW(L.C) = \frac{(-0.5)^2}{0.5(1-0.5)} = \frac{0.25}{0.25} = 1 \text{ (Pure node)}$$

$$SW(RC) = \frac{-0.5+0.5+0.5}{0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)} = 0.25/0.75 = 0.33$$

$$SW(Root) = \frac{[-0.5+0.5+0.5-0.5)}{(0.25+0.25+0.25+0.25)} = 0$$

$$SW_{gain} = 1+0.33-0) = 1.33 \Rightarrow$$

* How to make prediction ?



XGB classifier

$$O/P = \sigma\left(\text{Base mode} + \alpha(SW)\right)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$

** In Gradient Boosting

$f(x) = \text{base model} + \alpha_1 M_1 + \alpha_2 M_2 + \alpha_3 M_3 + \cdots \alpha_n M_n$

since its a classification problem, we need to Sigmoid fn.

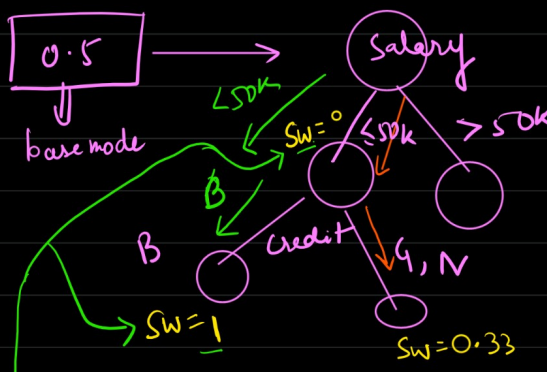
⇓

gives the value b/w 0 & 1

* first row prediction (≤50k, B)

range of sigmoid is 0 to 1

* Base model = $\log\left(\frac{P}{1-P}\right)$ (log odds)



XGBoost (eXtreme Gradient Boosting) is a high-performance, scalable machine learning library for gradient boosting, which is used for regression, classification, and ranking tasks. It implements decision tree-based ensemble learning with a focus on speed and efficiency. XGBoost includes features like regularization to prevent overfitting, parallel processing, tree pruning, and handling of missing values. It also supports cross-validation and early stopping to improve model performance. Known for its high accuracy and flexibility, XGBoost is widely used in both academic research and industry applications, particularly for structured/tabular data.

$$O/P = \sigma\left(\text{Basemodel} + \alpha\,(SW)\right)$$

$$= \sigma\left(\log\left(\frac{0.5}{1-0.5}\right) + \alpha\,(0+1)\right)$$

$$= \sigma\left(0 + 0.1 \times 1\right) \quad \text{say } -0.1$$

$$= \frac{1}{1+e^{-0.1}} = 0.52$$

2nd row (≤50k, G)

$$O/P = \sigma(0 + \alpha(0.33))$$

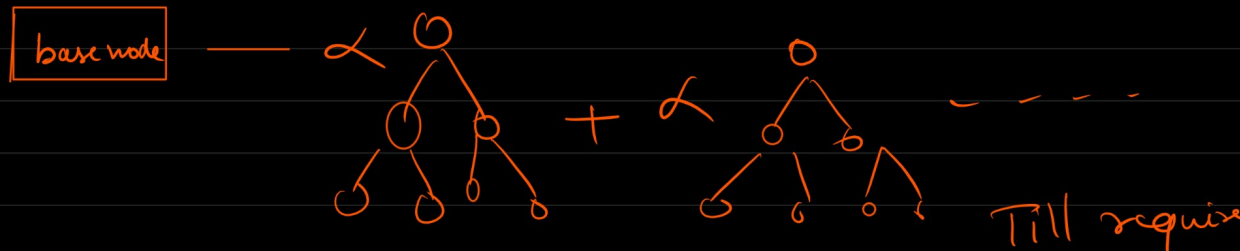
$\alpha = 0.1$

$$= \frac{1}{1+e^{-0.33}} = 0.58$$

Step 4 * Now do this for all rows calculate $\hat{y}$

	Credit Score	Approv	Base model	(R_1) $(y_{act} - y_{base\,P})$	$\hat{y}$
✓	<=50k	B	0	0.5	−0.5	0.52
✓	<=50k	G	1	0.5	0.5	0.508
✓	<=50k	G	1	0.5	0.5	=
✓	>50k	B	0	0.5	−0.5	=
✓	>50k	G	1	0.5	0.5	=
✓	>50k	N	1	0.5	0.5	=
✓	<=50k	N	0	0.5	−0.5	=

Step 5 * Again Construct DT with all and features and $\hat{y}$ as target variable



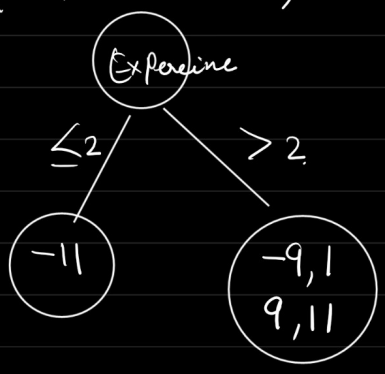
$O/P = \sigma(\text{Base learner} + \alpha_1(DT_1) + \alpha_2(DT_2) \dots \alpha_n(DT_n))$ of tree is achieved.

* XG Boost Regressor

Exp	Gap	Salary	Base Prediction	Res !
→ 2	Yes	40	51	−11
→ 2.5	Yes	42	51	−9
3	No	52	51	1
4	No	60	51	9
4.5	Yes	62	51	11

$$\text{Base Prediction} = \frac{(40+42+52+60+62)}{5} = 51.2 \approx 51$$

[−11, −9, 1, 9, 11)



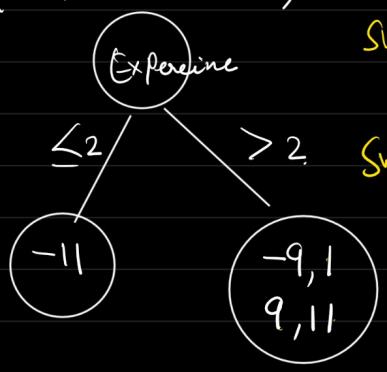
* SW in case of classification

$$= \frac{(\sum \text{Residual})^2}{\sum Pr(1-Pr) + \lambda}$$

* SW in case of regression

$$= \frac{(\sum \text{Residual})^2}{\text{No of residuals} + \lambda}$$

[−11, −9, 1, 9, 11)



$$SW_{Root} = \frac{(-1+-9+1+9+11)^2}{5+1} = \frac{1}{6}$$

$$SW_{LC} = \frac{(-11)^2}{1+1} = \frac{121}{2} = 60.5$$

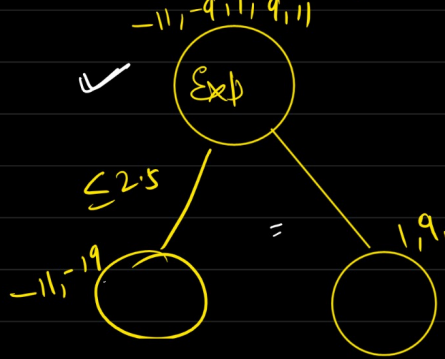
$$SW_{RC} = \frac{(-9+1+9+11)^2}{4+1} \quad (\text{Here } \lambda = 1 \text{ taking})$$

$$= \frac{144}{5} = 28.8$$

$$SW\ Gain = (60.5 + 28.8) - 0.166$$

$$= 89.134$$

* Split will be based on highest SW gain



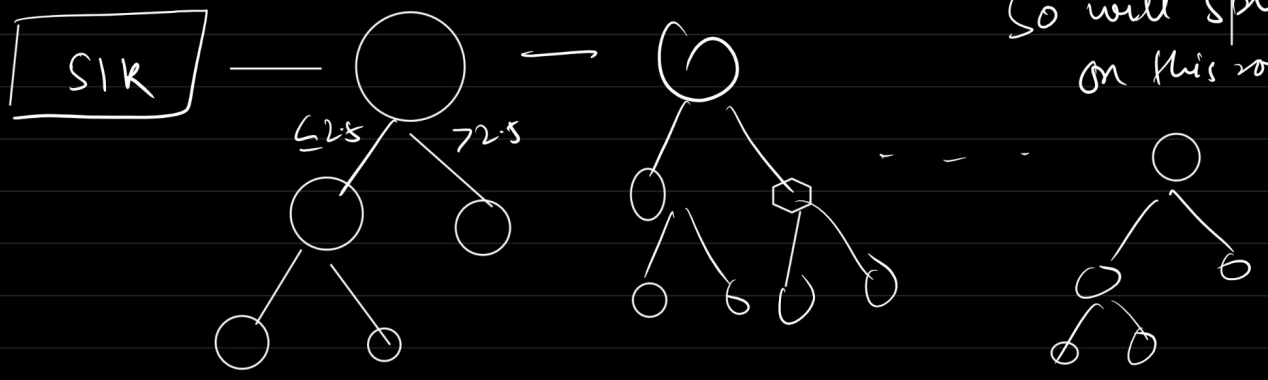
$$SW_{LC} = \frac{(-11-9)^2}{2+1} = \frac{400}{3} = 133.3$$

$$SW_{RC} = \frac{(1+9+11)^2}{3+1} = \frac{(21)^2}{4} = 110.23$$

$$SW_{root} = 0.166$$

$Sw_{gain} = 133.3 + 110.23 - 0166$

$\simeq 244 \Rightarrow$ Highest gain, So will split on this node

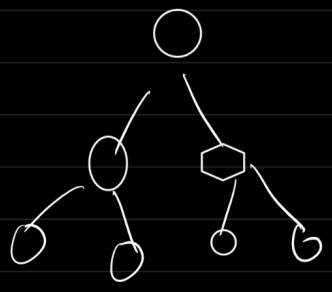


until the required no of dt is made.

Prediction = $51 + \alpha_1(DT_1) + \alpha_2(\ ) + \cdots \alpha_n(DT_n)$

* $Sw = \frac{(\sum \text{Residual})^2}{\sum Pr(1-Pr) + \lambda}$ → Hyperparameter.

One use of λ →



$\lambda \uparrow Sw \downarrow$

Chances are there Similarity weight gain can be negative → Stop splitting

* $\sum Pr(1-Pr)$ → lower value.

Any Sw less than $\sum Pr(1-pr)$ we stop splitting.

Say $0.5(1-0.5) = 0.25$ if whole Sw is less 0.25, stop splitting.