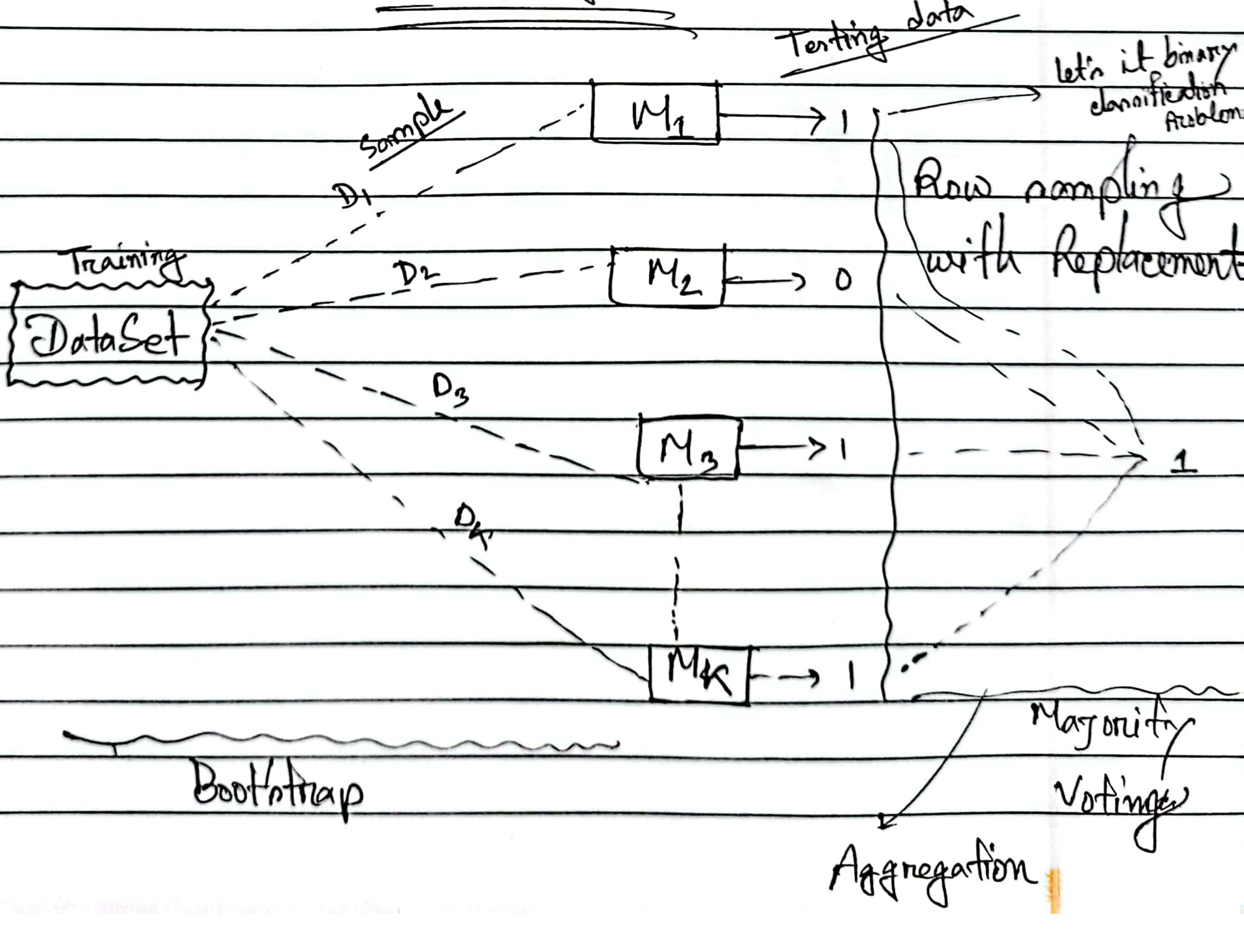
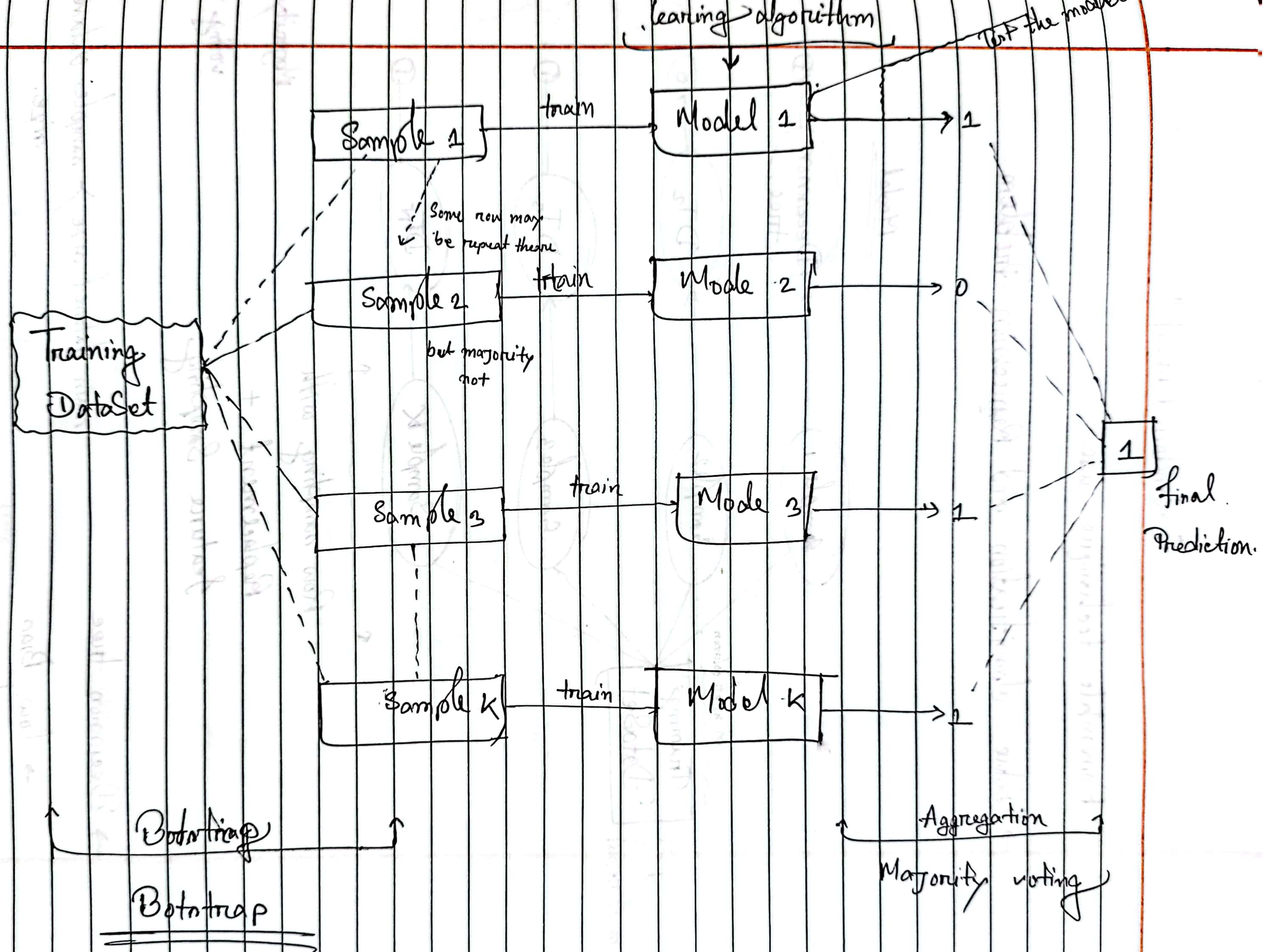
→ "Ensemble Technique".

- Ensemble means → Combining Multiple Model.

→ Bagging	→ Boosting
(Bootstrap Aggregation)	
① Random forest	① AdaBoost.
↓	② Gradient Boosting
use multiple decision tree.	③ XgBoost.

Bagging



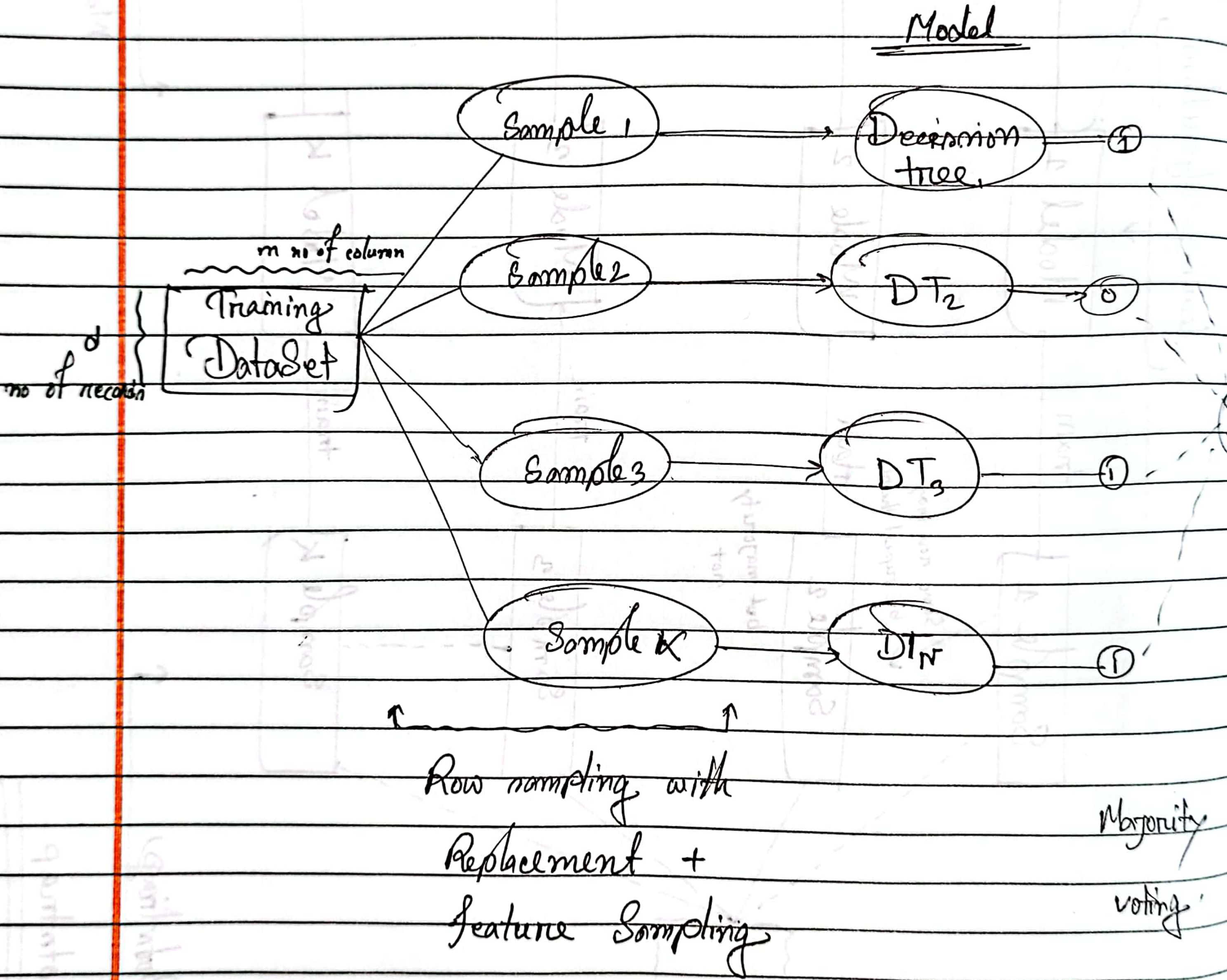


learning algorithm
Test the model.
Training DataSet
Sample 1
Some row may be repeat there
Sample 2
but majority not
Sample 3
Sample K
train
train
train
train
Model 1
Mode 2
Mode 3
Model K
1
0
1
1
1
final Prediction.
Aggregation
Majority voting
Bootstrap
Bootstrap

"Random forrest"

→ Ensemble technique use.

→ Solve classification and Regression Problem.



→ Decision tree

→ low Bias

→ High variance

main dataset size > sample dataset size.

when we create decision tree to it's complete depth it's leads to something called Overfitting.

→ But in "Random forest"

High variance

↓ Converted

low variance

$\frac{\text{Data } 1000}{\text{change } 200}$ is any impact ? No.

In Regression,

Prediction ↓

How many no of DT - hyperparameter Tuning : n-estimator.

DT$_1$ ------ Continuous value

DT$_2$ ------ Continuous value

DT$_3$ ------ Continuous value

⋮

DT$_K$ ------ Continuous value

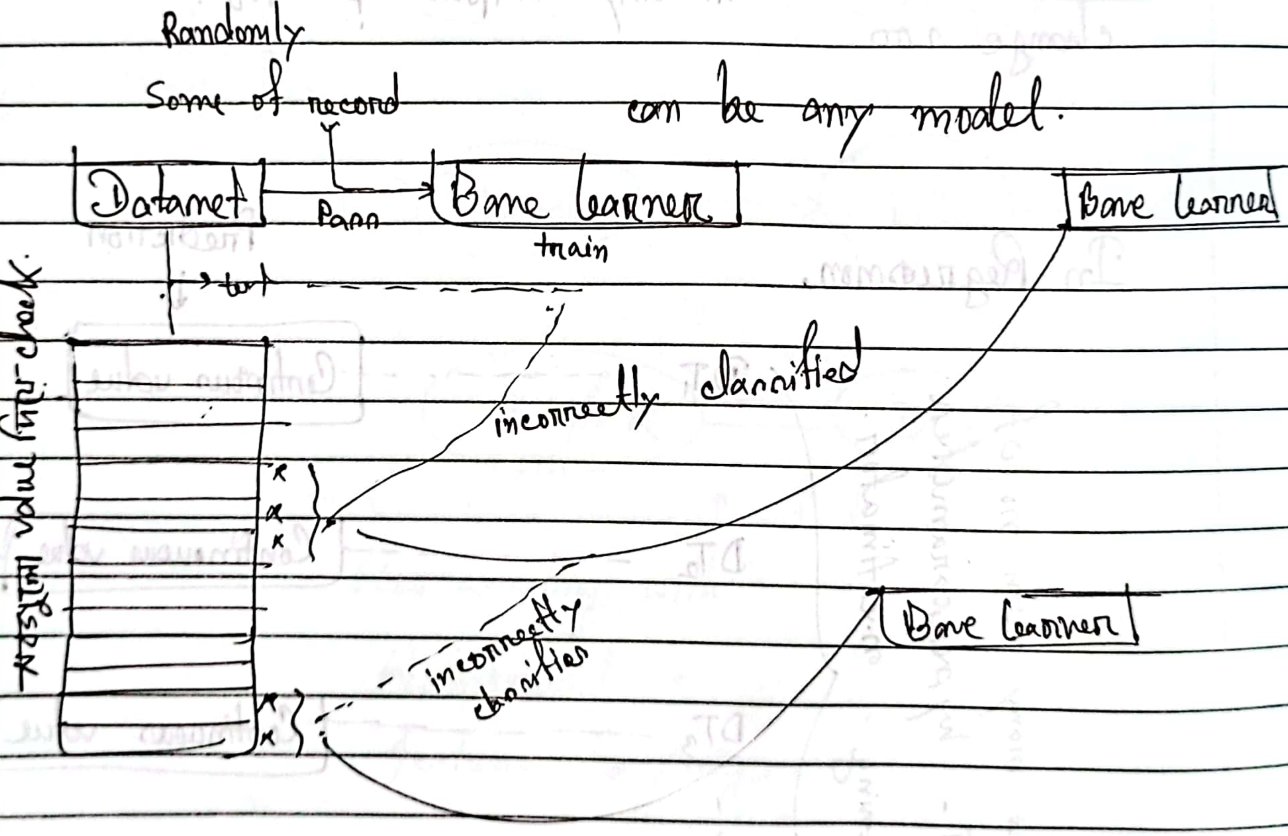
$\boxed{\text{final Prediction}}$ use Mean / Median

Based on Distribution

Boosting

Boosting is an ensemble learning technique where model attempts to correct the errors of the Previous model.

Combine weak learners → Strong learner.



Check Notebook article.

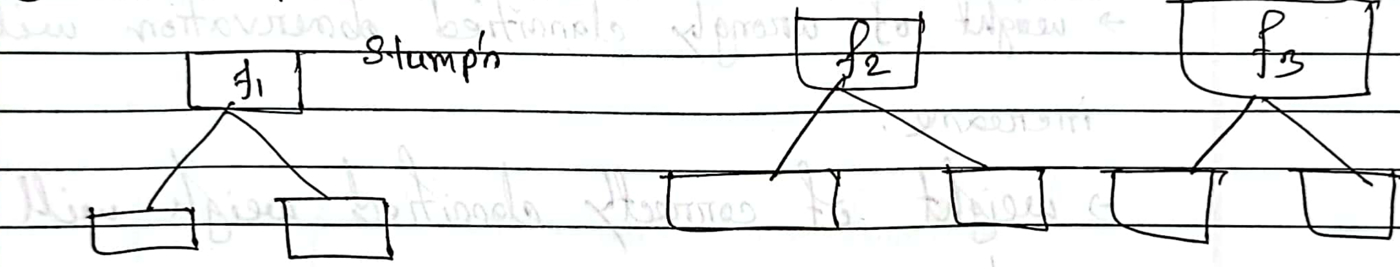
AdaBoost [all base learner decision tree]

f_1	f_2	f_3	o/p	Sample weight	Updated weight	Normalized weight	bucket
				① $1/7$	0.05		[ ]
				$1/7$	0.05		[ ]
				$1/7$	0.349		[ ]
				$1/7$	0.05		[ ]
				$1/7$	0.05		[ ]
				$1/7$	0.05		[ ]
				$1/7$	0.05		[ ]
				1	1.1		

$\Sigma = .68$

② One Depth Decision tree

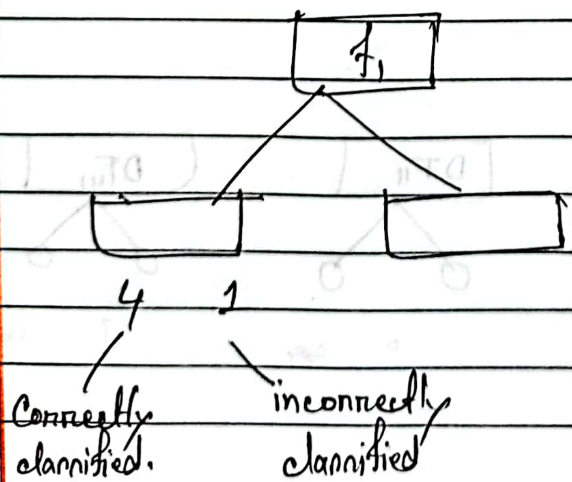
Stumps



~~check less Entropy or high Information Gain.~~

~~then Select it as a base learner~~

Let's f_1 Selected.



③ Total Error $= \frac{T}{N}$

T number of observation missclassified.

$= 1/7$

check notebook details.

(4) Performance stump $= \frac{1}{2} \log_e \left(1 - TE / TE\right)$

$= 1/2 \log_e \left(\left(1 - 1/7\right) / 1/7\right)$

$= 0.896$

(5) Update the weights,

$\Rightarrow$ old weight $* \, e^{+P.S}$ [incorrectly classify]

$= 0.349$

→ weight of wrongly classified observation will increase.

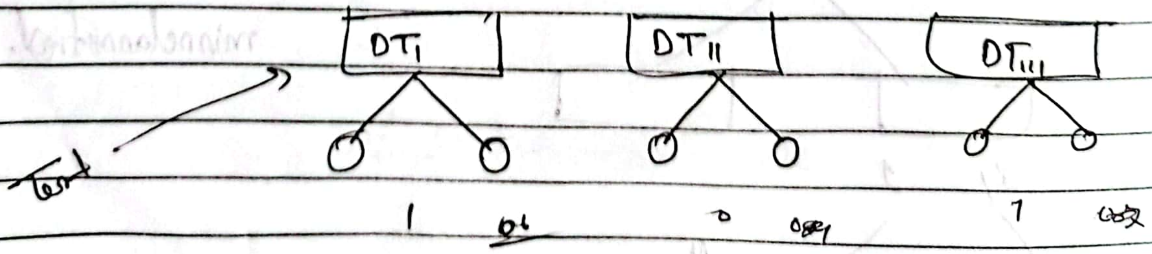
→ weight of correctly classified weigh will be reduce

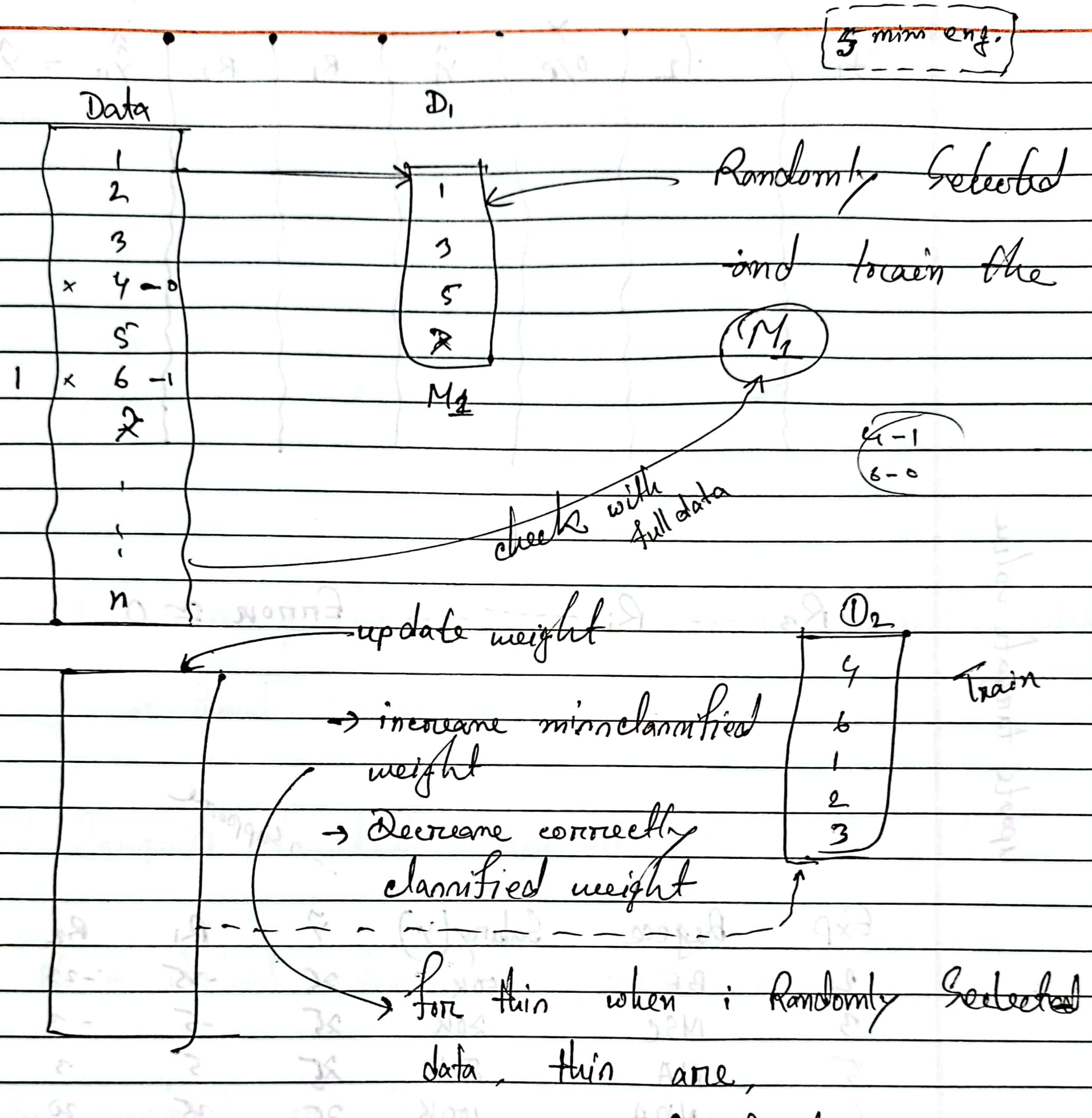
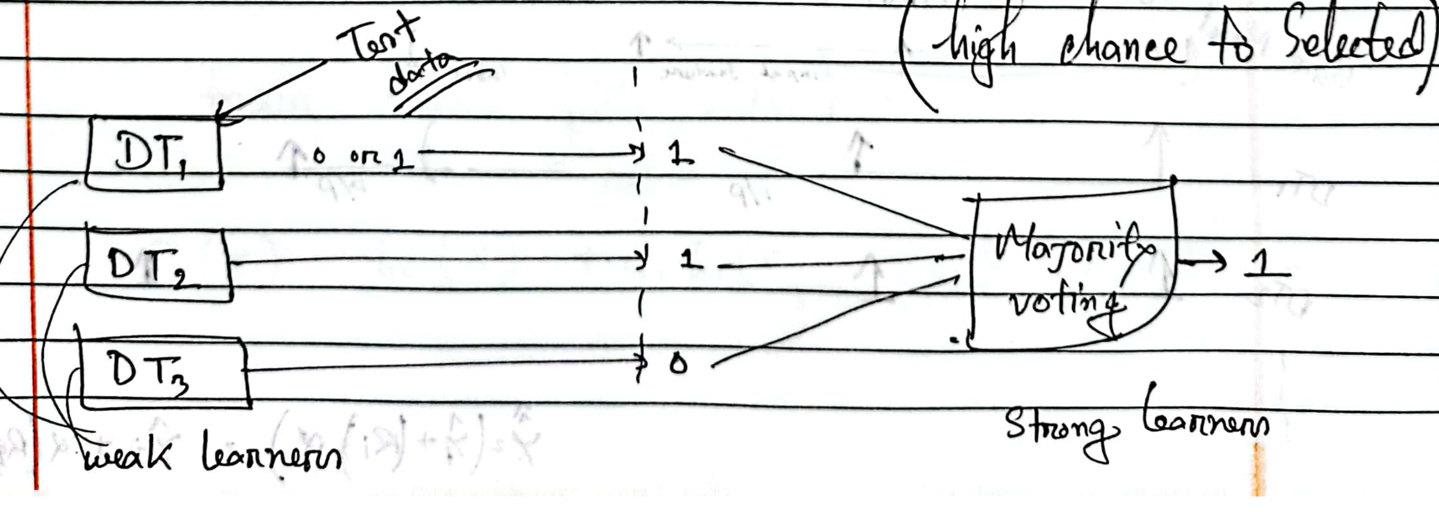
→ update the weight which are correctly classified.

formula → old weight $* \, e^{-P.S}$

$= 0.05$

Selected, Sequential Stump




5 mins eng.
Data
1
2
3
4 - 0
5
6 - 1
~~7~~
n
D1
1
3
5
~~7~~
M1
Randomly Selected
and train the
M1
4 - 1
6 - 0
check with full data
update weight
update whole
D2
4
6
1
2
3
Train
→ increase missclassified weight
→ Decrease correctly classified weight
for this when i Randomly Selected data, this are,
(high chance to Selected)

Test data
DT1
0 or 1
1
DT2
1
DT3
0
Majority voting
1
weak learners
Strong learners

$\to Y - \hat{Y}$

$\to \hat{Y} + R_2$

f_1	f_2	$\overset{Y}{O/P}$	$\hat{Y}_1$	R_1	R_2	$\hat{Y}_{11} = \hat{Y} + \alpha(R_2)$

update target value

R_3 --- R_i --------- Error $\simeq 0$

suppose

Exp	Degree	Salary(Y)	$\hat{Y}$	R_1	R_2
2	BE	50K	25	-25	-23
3	MSC	20K	25	-5	-3
5	NA	80K	25	5	3
6	MBA	100K	25	25	20

Base Model — input feature — label

DT_1 — i/p — o/p

DT_2

$$\hat{Y}_2(\hat{Y}_i + (R_i)\cdot\alpha) = \hat{Y}_1 + \alpha(R_1)$$

"Gradient Boosting"

Combines the prediction from multiple decision tree to Generate the final Prediction.

| Base Learner / Weak Learner | ← Decision tree |

Step 1 : Base Model --> ($\hat{y}$) Prediction value

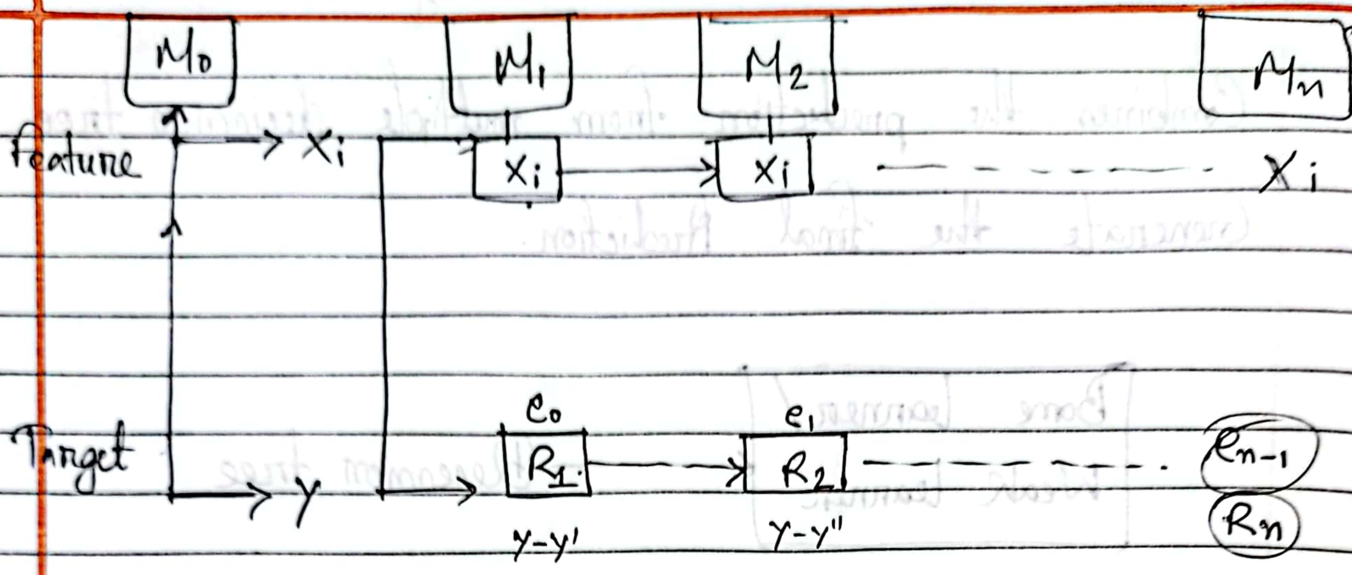
or Provide average value ($\hat{y}$)

② Compute Residual Error or Pseudo Residual.

R_1 = Actual value − Predicted value.

③ Construct a Decision trees model and train this: $\{\underbrace{x_i}_{\text{feature}}, \underbrace{R_1}_{\text{label}}\}$

$$F(x) = h_0(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x_1) \text{ ------ } + \sum_{i=1}^{n} \alpha_i h_i(x)$$



$H_0(X,Y)$ $\quad$ $H_1(X,e_0)$ $\quad$ $H_2(X,e_1)$ ------

$F_0(X)$ $\quad$ $F_1(X)$ $\quad$ $F_2(X)$ $\quad$ $F_n(X) = H_n(X, e_{n-1})$

Build first Model $\rightarrow H_0$

↳ which gave me some Prediction and generated some errors.

let's Combine it $\rightarrow f_0(X)$

$$f_0(X) = H_0(X,Y) + e_0' \qquad M_0$$

$$f_1(X) = f_0(X) + H_1(X, e_0) + e_1 \qquad M_1$$

$$f_2(X) = f_1(X) + H_2(X, e_1) + e_2 \qquad M_3$$

$\vdots$

$$f_n(X) = f_{n-1}(X) + H_n(X, e_{n-1}) + e_n$$

weight or co-efficient.

$$f_{n+1}(x) = f_n(x) + \gamma_n H(x, e_n)$$

$$f_0(x) = \gamma_0 H_0(x, y) + e_0$$

$$f_1(x) = f_0(x) + \gamma_1 H(x, e_0) + e_1$$

$$f_2(x) = f_1(x) + \gamma_2 H_2(x, e_1) + e_2$$

$\vdots$

$$f_n(x) = f_{n-1}(x) + \gamma_n H_n(x, e_{n-1}) + e_n$$

$$f_{n+1}(x) = f_n(x) + \gamma_n H(x, e_n)$$

$$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - f_n(x))^2$$

$$\frac{d\mathcal{L}}{df_n(x)} = -(y - f_n(x))$$

$$-\frac{d\mathcal{L}}{df_n(x)} = (y - f_n(x))$$

$$f_{n+1}(x) = f_n(x) + \gamma_n * H\left(x, -\frac{d\mathcal{L}}{df_n(x)}\right)$$

Pseudo Algorithm.

① I/P → $\{x_i, y_i\}$ dependent, independent.

② loss function,

Classification
Regression } different loss func.

differentiable

③ No. of tree

1. Initialize model with constant value

$$f_0(x) = \arg\min_{\gamma} \left(\sum_{i=1}^{n} L(y, \gamma) \right)$$

$\gamma = \hat{y}$

50
60
70

$$Loss = \sum_{i=1}^{n} \frac{1}{2} (y - \hat{y})^2 \downarrow$$

Residual.

$$\frac{1}{2}(50-\hat{y})^2 + \frac{1}{2}(70-\hat{y})^2 + \frac{1}{2}(60-\hat{y})^2$$

First Order derivative. $\hat{y}$

$$= 50 - \hat{y}(-1) + 70 - \hat{y}(-1) + 60 - \hat{y}(-1)$$

$$= 3\hat{y} - 180$$

$$\hat{y} = 60 \quad \text{(base model)}$$

⇒ Iterate $m = 1 - M$

↳ no of tree

→ Compute Pseudo Residual.

$$R_{im} = -\left[\frac{dL(Y, F(x_i))}{dF(x_i)} \right] \text{ for } i = 1 - n$$

$$L = \frac{1}{2}(Y - \hat{Y})^2$$

$$\frac{dL(Y,\hat{Y})}{d\hat{Y}} = -1(Y - \hat{Y})$$

$$-\frac{dL}{d\hat{Y}} = (Y - \hat{Y})$$

$R_{11} \rightarrow -10$
$R_{21} \rightarrow 10$
$R_{31} \rightarrow 0$

III ⇒ fit a base learner $h_m(x)$

i/p $\{(x_i, R_{im})\}$

$$\Rightarrow \gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} \left(L(Y_i, F_{m-1}(x_i) + \gamma) \right)$$

$Y - \hat{Y}$

⇒ Updating the model $F_m(x) = F_{m-1}(x) + \alpha_m (h_m(x))$

learning rate

Also some confusion.

Check → Gradient Boosting Machine learing for Data Scientist. Analyticvidhya

XGBoost especially concerned speed and Concern.

XGBoost

→ 10 time faster

→ Hackathon Problem solve.

EXtreme Gradient Boosting.

classification.

Salary	Credit	Approval	P_r 0.5	Res₁	New (P_r) consider	Res₂
<=50K	B	0		-.5	0.6	-0.4
<=50K	G	1		0.5	0.4	-0.6
<=50K	G	1		0.5	0.3	-0.7
>=50K	B	0		-0.5	0.7	-0.3
>=50K	G	1		0.5	0.2	0.8
>=50K	N	1		0.5	0.1	0.9
<=50K	N	0		-0.5	0.4	-0.6

GD

DT₁

Bad
Good
Normal.

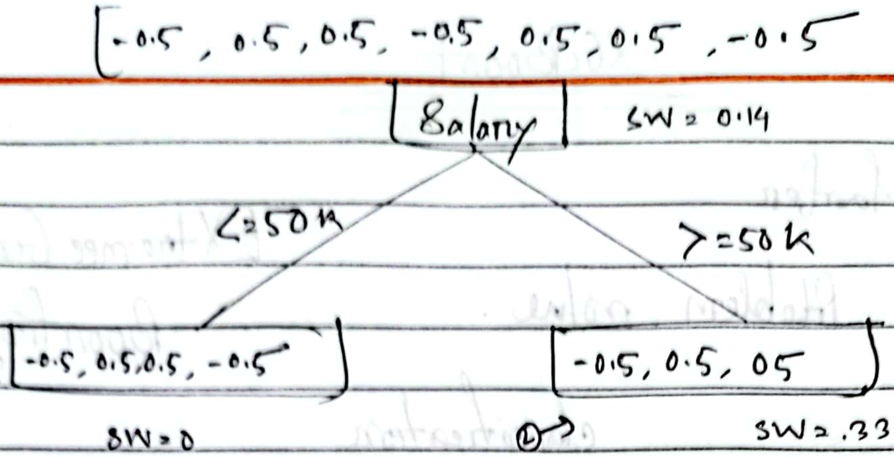
① Constructing a Base Model.

DT₂

$$\hat{y} = \frac{0+1}{2} = 0.5$$

$$P_r = 0.5$$

→ Constructing the tree.



Binary tree leave nodes 2 every time

② Compute Similarity weight $= \dfrac{\sum (\text{Residual})^2}{\sum (\text{Pr} * (1-\text{Pr}))}$

$$① \rightarrow = \frac{\cancel{-0.5} + \cancel{0.5} + \cancel{0.5} - \cancel{0.5}}{0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5)}$$

$$= \cancel{0/1}$$

$$= 0$$

② - Similarity weight $= \dfrac{(-0.5 + 0.5 + 0.5)^2}{0.75}$

$$= \frac{0.25}{0.75}$$

$$= 1/3$$

- Similarity weight $= \dfrac{0.25}{0.25 \times 7}$

main net

Root

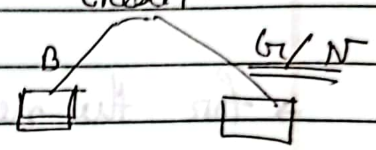
$$= \frac{0.25}{1.75}$$

$$= 1/7$$

$$Gain = 0 + 0.33 - 0.14$$
$$= 0.21$$

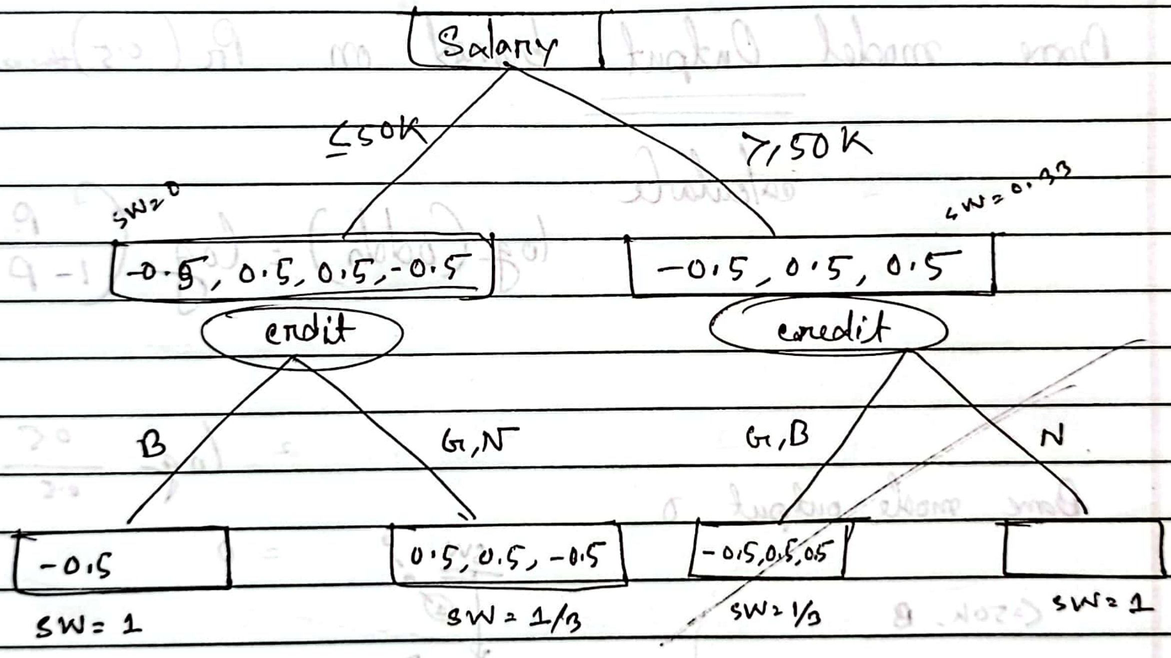
let's,
Consider Salary had better Gain.

then Split for Credit.



then Compare Gain which is higher & that is selected.

DT1



Gain = 1 + 1/3 − 0
= 1.33

Gain = 1.33

Can be select anyone if same.

after this we should splitting or not?

Post Pruning → Cutting a branch

Cover value → Pr(1−Pr)
0.25

if- Gain < 0.25 cut the branch.

Once I calculate the residual, i may construct any number of decision tree.

→ for the new data →

→ <=50k B

DT₁ ↓ sw → 1 output,

Base model Output based on P_R (0.5) this ex.

calculate

$$\log(\text{odds}) = \log\left(\frac{P}{1-P}\right) \qquad P = P_R$$

$$= \log \frac{0.5}{0.5}$$

Base model output 0

$$= 0$$

<=50k, B

SW ↓ 1

$$\sigma(0 + \alpha(1))$$

α → Learning Rate

$$= \sigma(0.1)$$

$$= \frac{1}{1+e^{-0.1}} = 0.6$$

0.6 in the new probability for this record.

<=50 . B

$Pr = 0.6$

$$\sigma\left[0 + \alpha_1(DT_1) + \alpha_2(DT_2) + \alpha_3(DT_3 \;---\; \alpha_n(DT_n)\right]$$

Base Model input = Exp, Gap, Salary ←

D Tree₁ input = Exp, Gap, Res 1

D Tree₂ input = Exp, Gap, Res 2

Final Output = Base model + $\alpha_1 (T_1) + \alpha_2 (T_2)$ - - - -

$+ \alpha_n (T_n)$

Gama, γ = 150.5

← Pruning

Gain (—) of a split of tree.

140

Gain − Gama(γ) = −ve → Pruning this tree ~~that mean cut~~ this split of sub tree.

If true we should not Prun it.

XGboost Regression

Exp	Gap	Salary	Res 1	DT, O/P	Res 2
2	Yes	40K	-11K	46	
2.5	Yes	42K	-9K	46	
3	NO	52K	+1K	53.5	
4	NO	60K	9K	'	
4.5	Yes	62K	12K	'	

→ [Based Model]

Provide Avarage value

let's, Avarage Salary = 51K

→ Res,

→ [DT,] ← Exp, Gap, Res 1

[-11, -9, 1, 9, 11]

$SW = \frac{1}{5+1} = 0.16$

[Exp]

≤2 | >2

[-11] | [-9, 1, 11, 9]

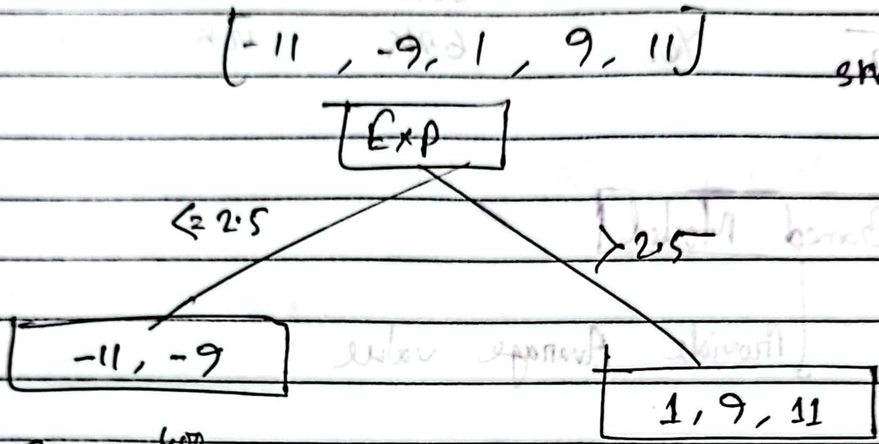
$SW = \frac{121}{1+1} = 65.5$ (λ=1)

$SW = \frac{144}{4+1} = 28.8$

Gain = (65.5 + 28.8) − (0.16) = 98.84

$$\text{Similarity Weight} = \frac{\Sigma(\text{Residual})^2}{\text{No. of Residual} + \lambda}$$

① Gain → 98.34



SW = ~~133.33~~ 0.16

$SW = \frac{400}{3} = 133.3$ $SW = \frac{441}{4} = 110.25$

$Gain_{(2.5)} = 133.3 + 110.25 - 0.16$

$= 143.42$

Gain (3)

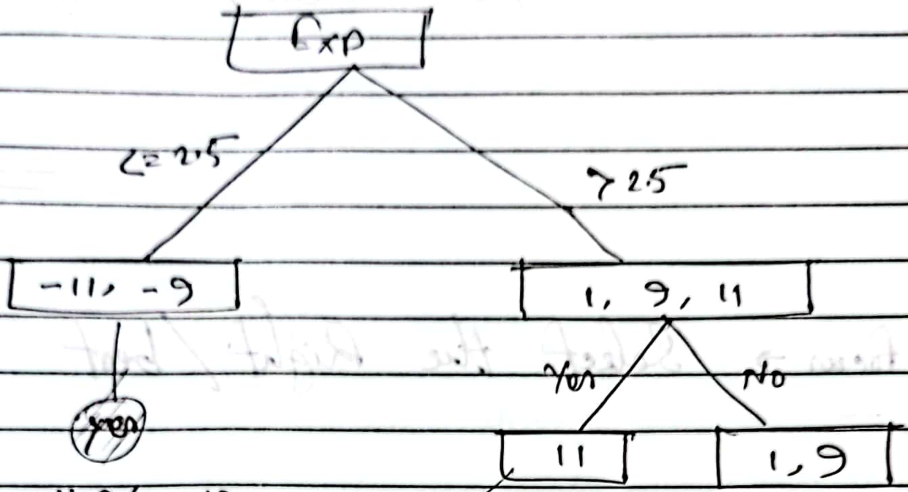
Gain (4)

Gain (4.5)

Better Info. Gain will be selected for split.

Lets, Consider G(2.5) is highest.

[-11, -9, 1, 9, 11]



output = $-11-9/2 = -10$

an = -10

Calculate Similarity score

Calculate Gain.

output = 11 output = 5

→ Pass first record.

2, Yes, -4

Base model output 51k.

output of this tree

DT1

α

$51 + 0.5 \boxed{-10}$

$= 51 - 5$

$= 46$

Hyperparameter Optimization for XGBoost Algorithm.

Here some Problem.

focus → Select the Right / best Parameter.

Hyperparameter Tuning ↓

GBoosting →

XGBoosting →

SVM
⋮

Bagging used

low bias
high variance

occurs

Convert

low bias
low variance

Boosting used

high bias
low variance

observed

low bias
high variance

Bagging can be used to avoid overfitting.

high bias
high variance

Boosting can be used to avoid underfitting.