Lecture - 05 :

→ Ensemble Techniques [Bagging, Boosting]

→ Random forest.

→ AdaBoost

→ XgBoost.

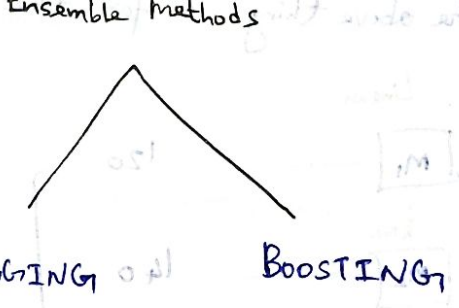
Ensemble Techniques :

→ Classification and Regression Problem.

We are Solved One algorithm for any Classification and Regression Problem.

Can we Solve multiple algorithm to Solve a problem?

Yes, we can use Multiple algorithm to Solve a Problem. Something Called as Ensemble methods.

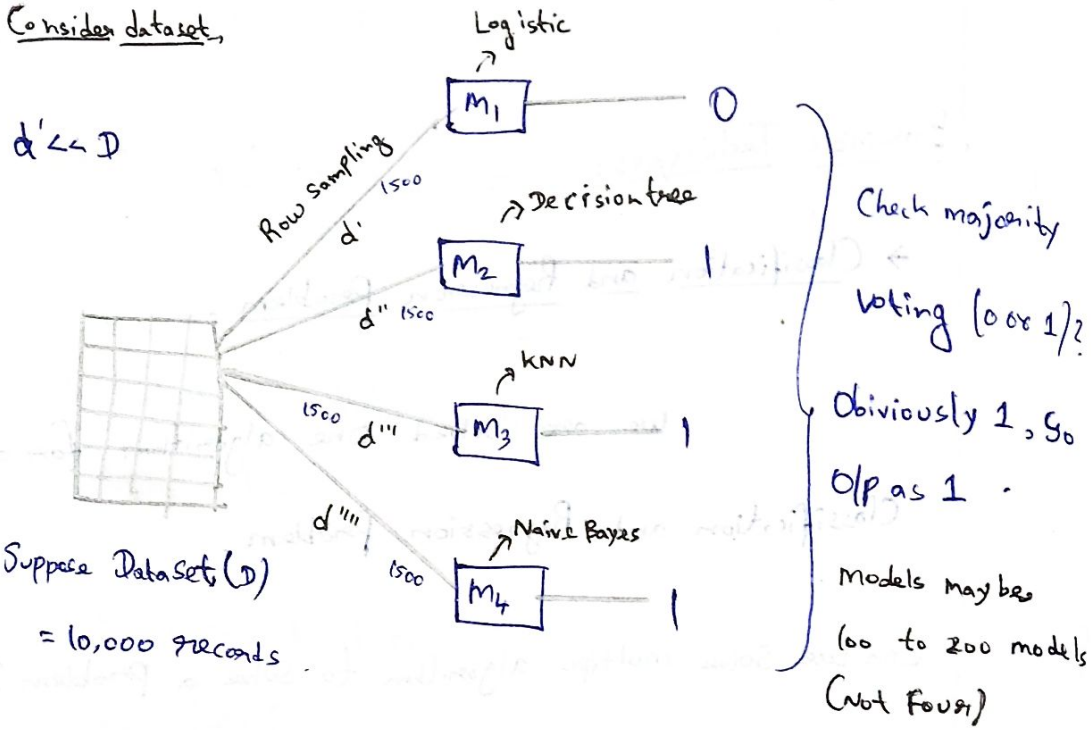


The Two techniques used to Solve problems with Multiple algorithms.

BAGGING:

Bagging is a technique to combine multiple algorithm to solve a problem. [parallely]

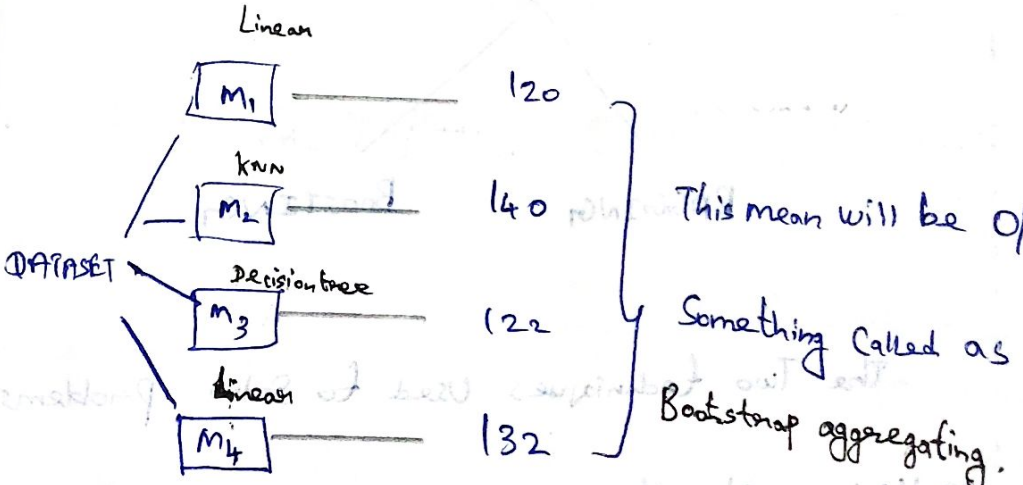
Classification Problem



the records in d' or may be in d" or may be in d"'.

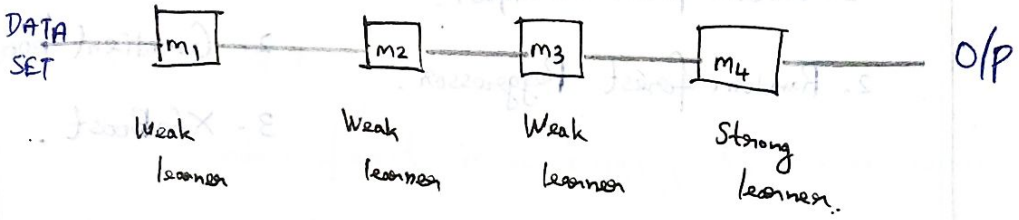
Just row sampling.

The above thing is for the Classification problem.



BOOSTING:

Bagging is Combined model as Parallely, Then the Boosting Combined model Serially and also improve the efficiency of Output.



To Combine these weak learners to get the strong learner and get efficient Output.

Example:

To create a data Science Course, we need

Another Example: One Scientist Solve a particular problem which involves various geography, maths, Physics and Chemis.

Many Experts are help to find the answer and give Correct answer.

Which algorithm is used for working with BAGGING and BOOSTING?

BAGGING

↓

1. Random forest Classifier.
2. Random forest Reggressor.

BOOSTING

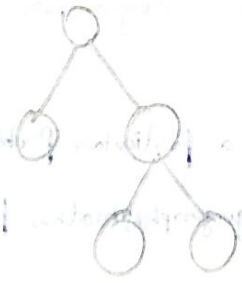
↓

1. Adaboost.
2. Gradient boost.
3. XGBoost.

BAGGING,

Random forest Classifier and Regressor:

Before moving, what is the main problem in the Decision Tree?



Overfitting

So, Generally, the overfitting is occured.

{ Low Bias, High Variance }

In order to avoid overfitting, we use Pruning techniques.

The Post Pruning is used, Sometime the large dataset. We can't Post Pruning.

On otherhand, Pre Pruning is set by hyperparameter. Sometime the model may be not accurate based on this data.

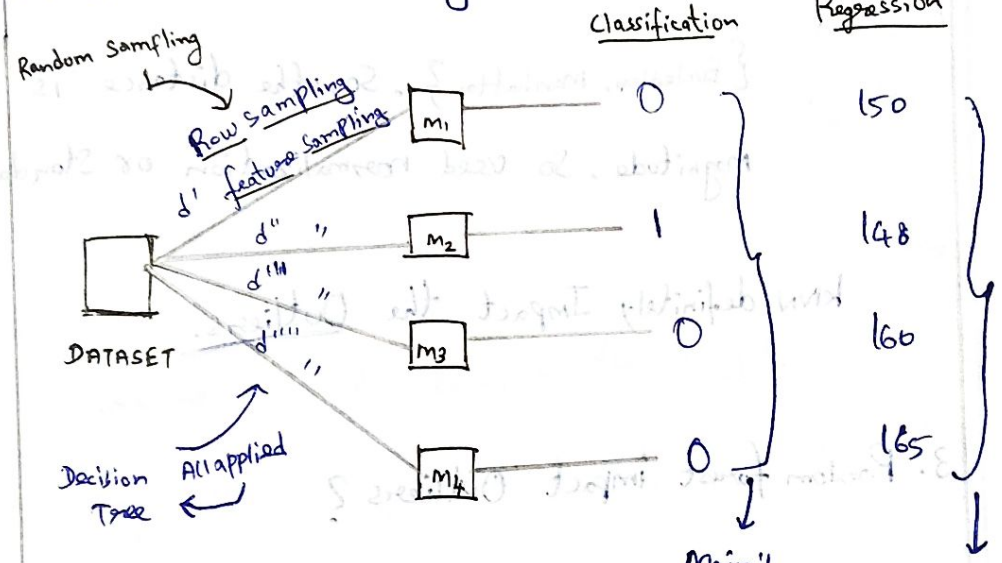
In this case, the random forest helps to reduce the High Variance to Low Variance.

Then we get the Generalised model.

{ Low Bias
Low Variance }

How ~~work~~? Work ?

The random forest is a bagging technique is applied on the decision tree algorithms.



The main thing is that,

All M_1, M_2, M_3, M_4 models are Decision Tree.

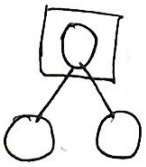
That's why we called it Random forest.

This helps to avoid overfitting in the decision tree and get Generalised model.

Some Interview Questions:

1. Is Normalization is required for Decision Tree?

Ans: No, because the decision tree split the things only. [Not apply any Quantity].



2. Is KNN is required Standardization (or) Normalization?

Ans: Yes, Because we apply the two formula are {Euclidian, Manhattan}. So, the distance is high magnitude. So used normalization or Standardization.

KNN definitely Impact the Outliers.

3. Random forest impact Outliers?

Ans: The Random forest is Generally less Sensitive to outliers Compared to individual decision tree.

But Decison Tree is impacted by outliers due to the nature of making splits.

In Bagging, Custom Bagging is also available.

BOOSTING:

ADABOOST:

Adaboost also working with Decision Tree.

Let's consider,

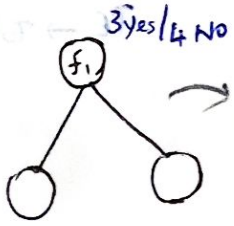
f_1	f_2	f_3	f_4	O/P		Weight
-	-	-	-	Yes	AdaBoost Initializes ⟹ weights	1/7
-	-	-	-	NO		1/7
-	-	-	-	-		1/7
-	-	-	-	-		1/7
-	-	-	-	-		1/7
-	-	-	-	-		1/7
-	-	-	-	-		1/7

Total records = 7 [Based on this weights are assigned]

Overall weight = 1 $\left[\frac{1}{7}+\frac{1}{7}+\frac{1}{7}+\frac{1}{7}+\frac{1}{7}+\frac{1}{7}+\frac{1}{7}=1\right]$

After that taking the feature with the help of information Gain.

In this Scenario, the decision tree take,

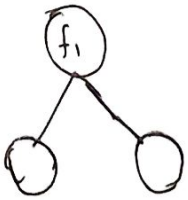


{Weak Learner}

In Adaboost, The decison tree only take one depth of the tree. This is called as STUMPS.

Stump - One level decision Tree.

Why use Stump? Because it is weak learner.



f_1	f_2	f_3	f_4	o/p	weight	New weight
-	-	-	-	yes	1/7	0.05
-	-	-	-	No	1/7	0.05
-	-	-	-	-	1/7	0.05
-	-	-	-	No	1/7	0.349
-	-	-	-	-	1/7	0.05
-	-	-	-	-	1/7	0.05
-	-	-	-	-	1/7	0.05

After train this weak learner Model, Pass all the records Suppose one of the record wrong predict wrong.

The 4th row o/p is No, Suppose predicting "yes" as a wrong.

If wrong, Calculating the Total Error, (1)

1. Calculating Total Error,

$$\text{Total Error} = 1/7 \cdot [\text{weight}]$$

2. Performance of Stump,

$$= 1/2 \log_e \left(\frac{1 - TE}{TE} \right) \quad TE \rightarrow \text{Total Error}$$

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

$$= 0.895.$$

3. Updating the weights,

- Why Updating? Because find out the Correct and wrong records. If wrong record found, ~~if~~ this pass to another weak learner. [STUMPS].

PS → Performance of STUMP.

Correct records:

New sample weight = Weight * e^{-PS} = $1/7 \times e^{-0.895}$ = 0.05

$1/7 = 0.1428$
↓↓ reduced
0.05

Incorrect records:

New sample weight = Weight * e^{PS} = $1/7 \times e^{0.895}$ = 0.349

New weight: See the left Page.

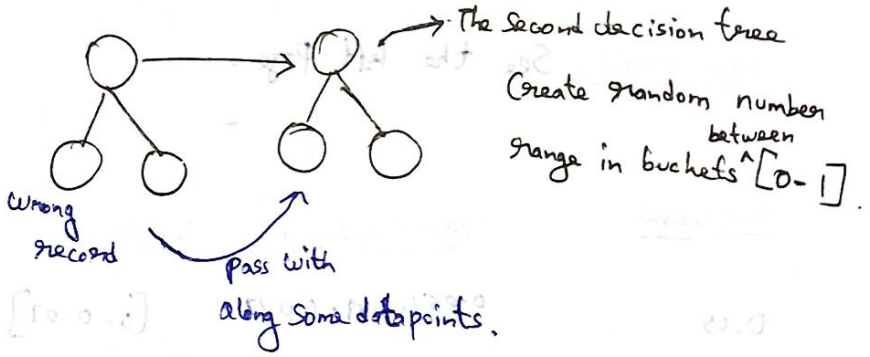
New weight	Normalized weight	Buckets
0.05	0.05/0.649 = 0.077	[0 - 0.07]
0.05	0.05/0.649 = 0.077	[0.07 - 0.14]
0.05	0.077	[0.14 - 0.21]
0.349	0.349/0.649 = 0.537	[0.21 - 0.747] ← High size
0.05	0.077	[0.747 - 0.753]
0.05	0.077	[0.753 - 0.760]
0.05	0.077	[0.760 - 0.767]
0.649	≃ 1	

Why Normalize ?

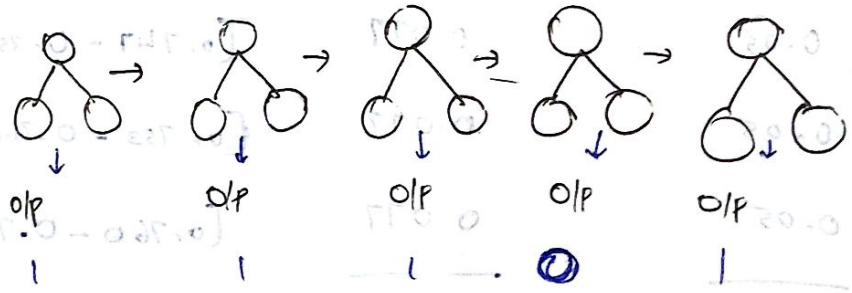
While update the weights, the magnitude are changed. So, the new weight does not come to 1. That's why normalize them.

Buckets are used to identify the wrong records, if conclude the bucket range is high.

4th row record size bucket is high. So, identified 4th record as wrong and error.



Then the second decision tree train with wrong records and also some datapoints. After some amount of decision tree, the output as,



Majority votes => 1

So, 1 as the output for Classification problem.

In regression,

o/p	o/p	o/p	o/p	o/p
64-3	69	68-3	67	59

Aggregate Mean

⇓

o/p for regression problem.

So, the number of decision tree, the time Complexity is Worst.

NOTE: BAGGING AND BOOSTING not only support random forest and decision tree and also support any algorithm for base model.

BLACK BOX MODEL VS WHITE BOX MODEL: [Interview Q/A]

BLACK BOX - We Cannot see how algorithm work deeply.

WHITE BOX - We Can Understand how algorithm work deeply.

Linear Regression → white Box [θ value change] | ANN → Black Box [Unable to see how many neurons present]

Random forest → Black box [No of decision Tree]

Decision Tree → White Box [we see decision tree]

BOOSTING:

GRADIENT BOOSTING:

Consider the dataset,

Exp	Degree	Salary (y)	Base model prediction ($\hat{y}$)	R_1	R_2	R_3	R_4
2	BE	50k	75	-25	-23	-	-
3	Masters	70k	75	-5	-3	-	-
4	Masters	80k	75	3	3	-	-
5	Phd	100k	75	25	20	-	-

Residual decreasing

Step 1:

Base model O/P.

$$\text{Average mean} = \frac{50+70+80+100}{4} \approx 75 \Rightarrow \hat{y}$$

Step 2:

Compute residuals Errors, Pseudo residuals.

Base model Residual $\rightarrow R_1$

Step 3:

Construct decision tree $\{x_i, R_1\}$

$x_i \rightarrow$ independent feature.

$R_1 \rightarrow$ Residuals of base model.

Both are passed decision tree.

Then the decision tree work with inputs and gives the residuals as output (R_2).

Suppose R_2 have some value.

Suppose take the first record,

~~residuals = $\hat{y}$ - R_1~~

~~= 1.75~~

We pass the first record in base model,

$= \hat{y} - R_2$

$= 75 - 23 \Rightarrow 52$ [It is overfitting because it is closely to y]

To prevent Overfitting,

To add Learning rate (α)

$= 75 + \alpha(R_2)$

$= 75 + 0.1(-23)$

$= 75 - 2.3 = 73.7$ [But this value is huge diff. from y]

So, we add another decision tree,

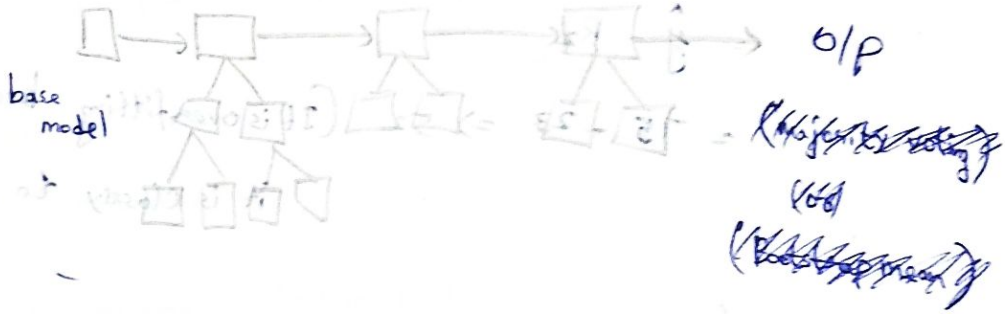
Therefore, this decision tree Computes next residuals. Then give the prediction.

So, the formula would be,

$$F(x) = h_\theta(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_n (h_n x)$$

↓ base model ↓ model-1 (decision Tree) ↓ model-2 (decision tree) - - - model-n

$$\boxed{F(x) = \sum_{i=1}^{n} \alpha_i h_i(x)}$$



Algorithm:

1. Initialize model with a constant value

$$F_0(x) = \arg\min \sum_{i=1}^{n} L(y_i, \nu)$$

2. For $m = 1$ to M:

i) Compute so-called pseudo-residuals:

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x) = f_{m-1}(x)}$$

for $i = 1, 2 \dots n$

2. $\{(x_i, \gamma_{im})\}_{i=1}^{n}$ (Fit the base learner)

3. Compute multiplier γ_m

$$\gamma_m = \text{argmin} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

4. Update the model,

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

3. Output $F_m(x)$

Inputs:

1. $\{x_i, y_i\}$

$x_i \rightarrow$ independent feature.

$y \rightarrow$ dependent feature.

2. $L(y, F(x))$

↳ Loss function

Regression - mse, mae etc

Classification - Log loss etc.

3. Number of trees in gradient boosting.

These three inputs are required for gradient boosting.

Consider,

Exp	Degree	Salary	$\hat{y}$ (base model)
2	BE	50k	60
3	Phd	70k	60
4	Master	60k	60

$\gamma \rightarrow$ predicted value $(\hat{y})$

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \left(\sum_{i=1}^{n} L(y, \gamma) \right)$$

↳ reduce the error

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

(1) Initialize model with constant value using above formula,

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

↑ All records Value.

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

Which value is assigned for constant? Ans: → Just derivative

Simple derivative,

$$\frac{\partial}{\partial x} (x^n) = n x^{n-1}$$

↓ first order

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

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

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

$$3\hat{y} = 180 \Rightarrow \hat{y} = 180/3$$

$$\boxed{\hat{y} = 60}$$

This value assigned for Constant base model. ($\hat{y}$).

② Iterate the tree Count M = 1 to M.

③ ↳ Compute pseudo residuals,

$$r_{im} = -\left[\frac{\partial L(y, F(x_i))}{\partial F(x_i)}\right] \text{ for } 1 \text{ to } n$$

It represents the

Loss $\frac{1}{2}(y-\hat{y})^2$

$$= \frac{2}{2}(y-\hat{y})(-1)$$

Looks like

$$= -y + \hat{y}$$

$$\frac{\partial L}{\partial \hat{y}} = -(y-\hat{y})$$

$$\boxed{-\frac{\partial L^{(y,\hat{y})}}{\partial \hat{y}} = y-\hat{y}}$$

Basically, r_{im} $(y-\hat{y})$

r_{11} → first record, model(1) (base model)

$r_{11} = -10$ → 1st record

$r_{21} = 10$ → 2nd record

$r_{31} = 0$

Once get the residual free, then fit base learner $h_m(x)$,

↳ ⑧ Fit a Base learner $h_m(x)$

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

③ $\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \gamma)$ ↳ Previous model

↓

Loss $\frac{1}{2}(y-\hat{y})^2$

$$\sum_{i=1}^{n} \frac{1}{2}(y_i - (60 + \hat{y}))^2$$

↓ previous model output.

⇓

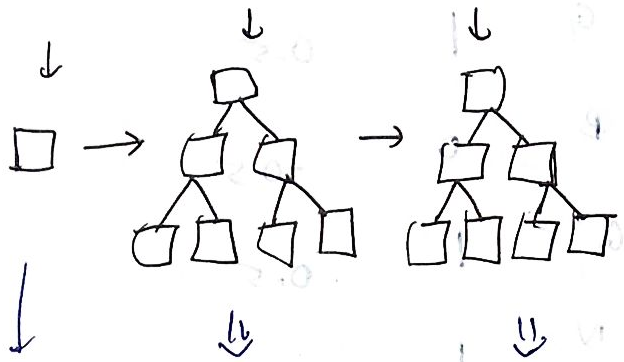
Then again, Calculate residues

argmin → It means minimum residuals.

④ Update the model,

$$F_m(x) = F_{m-1}(x) + \alpha\,(h(x))$$

Without α, It may be causes overfitting.



.. until get the ~~less~~ minimum residuals.

60 ① 60 + (-10)

rec

Learn rate = 0.1

60 - 1.0

= 59.0

② rec:

60 + (0.1)(10)

= 61

These all are about the Gradient Boosting.

BOOSTING,

XGBOOST:

Xgboost Classifier:

Consider the dataset,

Base model → O/P = 0.5

$(y - \hat{y})$ ↓ Pr = 0.5

Salary	Credit	Approval	Residual (R_1)	R_2	R_3	R_n
<=50k	B	0	-0.5			
<=50k	G	1	0.5			
<=50	G	1	0.5			
>50	B	0	-0.5			
>50	G	1	0.5			
>50	N	1	0.5			
<=80k	N	0	-0.5			

Step1: Create base model and given O/P = 0.5 (weak learner)

After Creating base model, the following steps to be Construct,

1. Create a Binary Decision Tree Using the feature.

2. Calculate Similarity weight,

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

3. Information Gain.

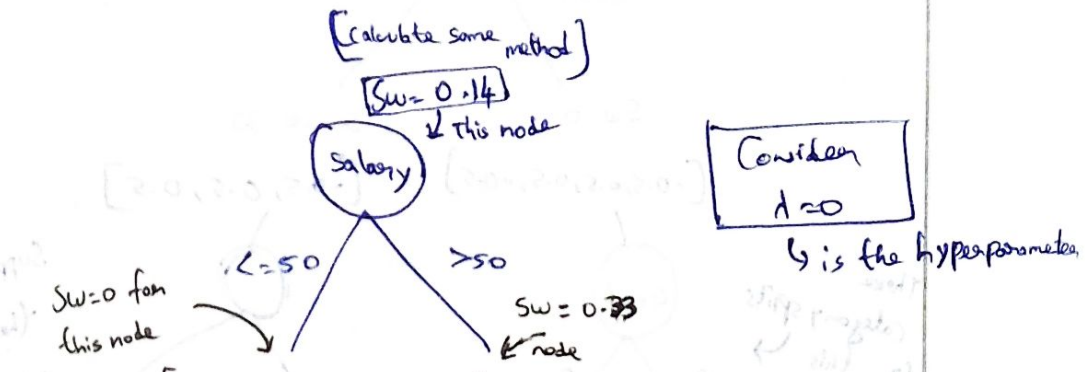
1. Base model - O/P = 0.5

$$Pr = 0.5$$

Ensure base model trained with the residuals before Creating binary decision tree.

2. Create a Binary decision tree.

Consider Salary has a high information Gain.



Consider $\lambda = 0$

↳ is the hyperparameter

$P_0 = 0.5$ from base model

$$Sw = \frac{(-0.5+0.5+0.5-0.5)^2}{0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)}$$

$Sw = 0$

$$Sw = \frac{(-0.5+0.5+0.5)^2}{0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)}$$

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

$Sw = 0.33$

Calculate information Gain,

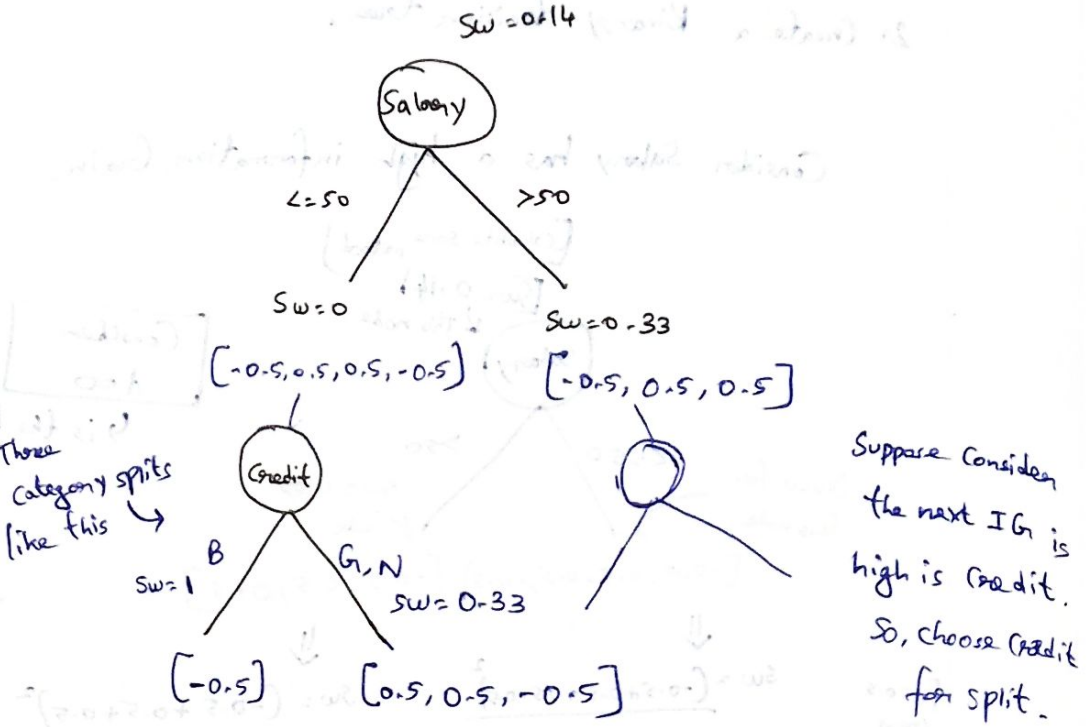
This information finding is little different, we find information finding through Similarity weight,

$$IG = \underset{Sw}{\text{Child node}} - \underset{Sw}{\text{root node}}$$

$$IG = (0 + 0.33) - 0.14$$

$IG = 0.19$ → It means this is the information Gain for root node. [Salary]

Then Again further Split will be happend and Commute.



IG = 1 + 0.33 − 0 = 1.33 [IG for Credit]

Why base model have P(G) = 0.5 ?

We Something apply, $\log\left(\frac{p}{1-p}\right) = \log\left(\frac{0.5}{0.5}\right) = 0$.

So, In first Case whenever the record goes to the base model, The model gives 0 with 0.5 probability.

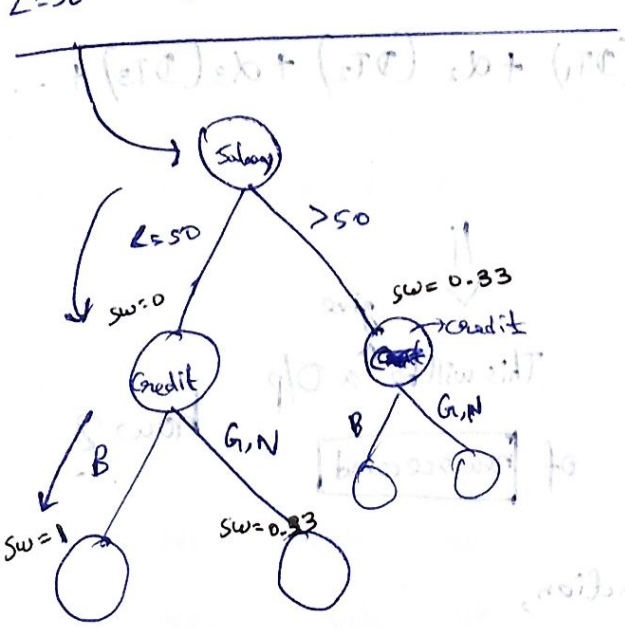
So, The boosting look like,

~~Suppose the new record~~

For understanding,

Take the first record in the dataset.

Salary	Credit	Approval	Residual
<=50	B	0	-0.5



So, $[0 + \alpha(sw)]$ (Learning rate)

$[0 + \alpha(1)]$

base model — First model of decision tree

Then apply Sigmoid function, $\sigma[0 + \alpha(1)]$

⇓

The output comes between 0 to 1.

↓

This output as the probability. Is passing next residual (R_2). Then second model compute.

The decision trees are more number. When stop?

The answer is Xgboost evaluate the metrics and Performance based on that, The decision tree Making is stop.

Then the model 2 decision tree as Same,

$$\sigma[0 + \alpha(\text{DT}) + \alpha(\text{DT}_2)]$$

Likewise,

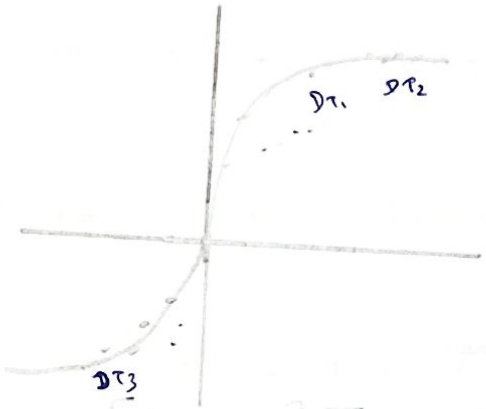
$$\left[\sigma\left[0 + \alpha_1(DT_1) + \alpha_2(DT_2) + \alpha_3(DT_3) + \cdots \alpha_n(DT_n)\right]\right]$$

$\Downarrow$

This will be give a O/p of new record

how?

The sigmoid function,



$\Rightarrow$ In this way, The O/p will be classified.

This is how the Xgboost classifier work.

It is the black box model, Because we didn't See the working of decision tree.

Pre pruning → Applied some hyperparameter tuning.

Note:

The each decision tree is based on independent feature not take dependent one.

XGBOOST REGRESSOR:

Base model o/p as average of all salary is 51.

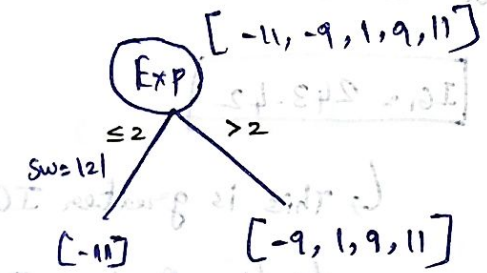
Consider the dataset,

Exp	Gap	Salary (o/p)	Residual (base model o/p)
2	Yes	40k	-11k (40 - 51)
2.5	Yes	42k	-9k (42 - 51)
3	No	52k	1k
4	No	60k	9k
4.5	Yes	62k	11k

The Similarity weight Change in regression, Consider, $\lambda = 0$

$$\text{Similarity weight (sw)} = \frac{\leq (\text{Residuals})^2}{\text{No of residuals} + \lambda}$$

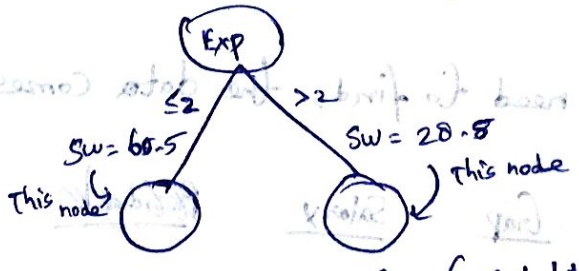
Tree will be,



$Sw = \frac{121}{1+0}$

$Sw = 121$

Suppose take $\lambda = 1$, $sw = \frac{(-11-9+1+9+11)^2}{5+1} = \frac{1}{6} = 0.16$



$Sw = \frac{121}{1+1}$

$Sw = 60.5$

$Sw = \frac{(-9+1+9+11)^2}{4+1} = \frac{144}{5}$

$Sw = 28.8$

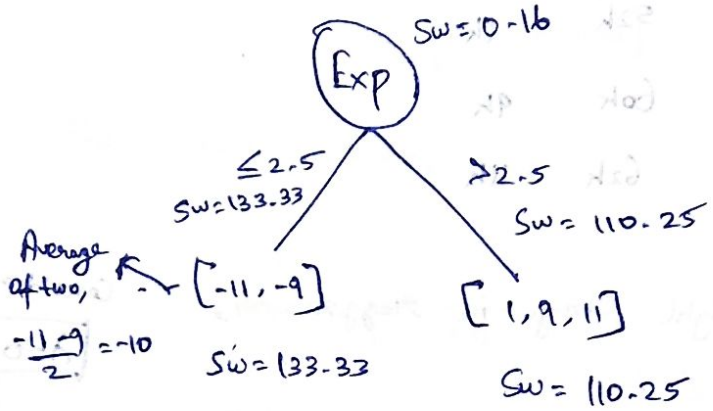
Then find the information gain,

$$= 60.5 + 28.8 - 0.16 = 89.13$$

↓

The information gain for Experience.

Next split,



The information gain,

$$IG = 133.33 + 110.25 - 0.16$$

$$\boxed{IG = 243.42}$$

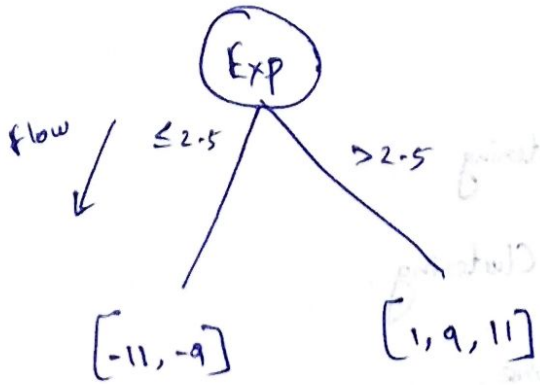
↳ This is greater IG compare to the previous IG.

For doing all the splits and possibilities whichever have a high information gain to take the split.

Suppose we need to find, the data comes

Exp	Gap	Salary
2.2	yes	?

Suppose ~~Comes~~ The above data Comes the Category,



Average

$\frac{-11-9}{2} = -10$

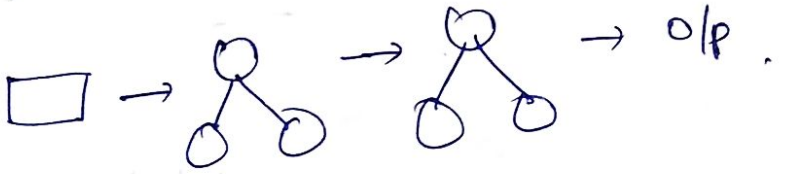
$\alpha_1 = 0.01$

$51 + \alpha_1 (-10) \Rightarrow$ O/p [But this is for one decision tree]

51 → base model

$\alpha_1(-10)$ → First decision Tree

Like wise Create more decision tree,



$O/p = 51 + \alpha_1(DT_1) + \alpha_2(DT_2) + \alpha_3(DT_3) + \cdots \alpha_n(DT_n)$

This is about the XGBoost Regressor.

It is also Comes Under the blackbox model.