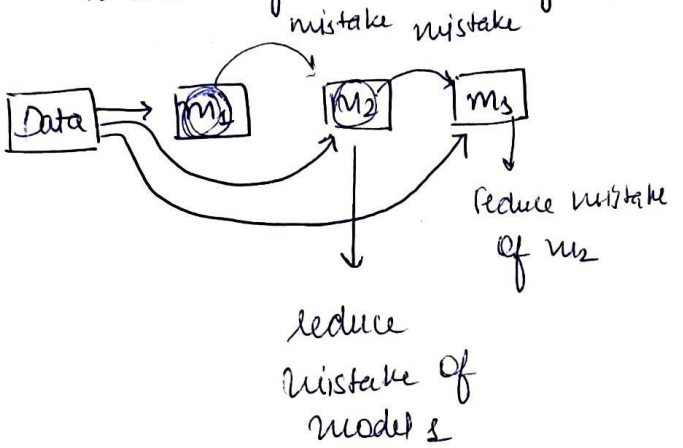
Gradient Boosting

Boosting: Boosting is a general ensemble method in Machine learning that aims to create a strong classifier or regressor by combining the predictions of several weaker models. The idea is to build the strong model incrementally, by sequentially adding weak models that are trained to correct the mistakes made by the existing ensemble.



* Sequential Training
* All the model → same data
* HBLV model

What is Gradient Boosting

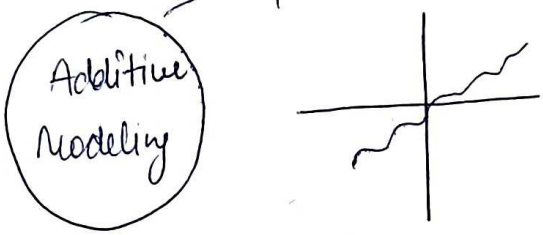
Gradient Boosting is a machine learning ensemble technique that aims to build a strong predictive model by combining the prediction of several weaker models using the concept of Additive Modelling typically decision trees. The method works by iteratively adding models to the ensemble, with each new model trained to correct the mistakes made by the combined ensemble of existing models.

Gradient Boosting is a powerful and flexible method that can be used for both regression and classification tasks.

It is particularly effective when the data has complex, non-linear relationships, and it often performs well even with little hyperparameter tuning.

Its popularity in real-world application and machine learning competitions is testament to its effectiveness and it has implementations in most major machine learning libraries, such as sk-learn, XGBoost, LightGBM, Catboost

→ Gradient boosting work on mathematical pricipal



$y = x + \sin x$

$y = x \quad y = \sin x$

$m_1 \rightarrow m_2 \rightarrow m_3$

$f_1(x) \quad f_2(x) \quad f_3(x)$

$\downarrow$

$f_1(x) + f_2(x) + f_3(x)$

How Gradient Boosting Works? (Regression)

X	Y

$y = f(x)$

find small fun^n → find the relationship bet^n X, Y with the help of gradient boosting.

$\downarrow$

$f_0(x) + f_1(x) + f_2(x) - - - f_n(x)$

Steps: Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$

Step 2: for $m = 1$ to M:

(a) For $i = 1, 2, \ldots N$ compute

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

pseudo residual

(b) Fit a regression tree to the target r_{im} giving terminal regions.

$$R_{jm}, \quad j = 1, 2, \ldots, J_m$$

(c) For $j = 1, 2, 3, \ldots, J_m$ compute

$$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{ij}} L(y_i, f_{m-1}(x_i) + \gamma)$$

(d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{ij})$

Steps: output $\hat{f}(x) = f_M(x)$.

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, f(x))$, number of iterations M.

$\hat{y} = f(x)$

$L(y, \hat{y})$

output, prediction

$\{(x_i, y_i)\}_{i=1}^{n}$ → rows

Squared loss

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

↓

regression

$$L(y, \hat{y}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$L(y, \hat{y}) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Explain Step 1

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

$\frac{\partial L}{\partial \gamma} = 0$

↳ find that type of value of gamma which decreases value of these

no. of rows

$$\frac{\partial f_0(x)}{\partial \gamma} = \frac{\partial}{\partial \gamma} \sum_{i=1}^{n} (y_i - \gamma)^2$$

$$= -2 \sum_{i=1}^{n} (y_i - \gamma) = 0$$

$$y_1 - \gamma + y_2 - \gamma + y_3 - \gamma = 0$$

$192 - \gamma + 144 - \gamma + 91 - \gamma = 0$

$3\gamma = 192 + 144 + 91$

$\gamma = \frac{192 + 144 + 91}{3}$ ↳ γ mean

	Red	Ops	Marketing	Profit
0	165	137	472	192
1	101	92	250	144
2	29	127	201	91

explain Step 2: As many decision tree ⇒ No. of loops 2nd step will run.

m = decision tree

m_1 → first decision tree ($m = 1$)

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

$$r_{i1} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f = f_0} = \left[\frac{\partial L(y_i, f_0(x_i))}{\partial f_0(x_i)} \right]$$

$$L(y, \hat{y}) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad r_{i1} = \frac{1}{2} \frac{\partial}{\partial f_0(x_i)} \sum_{i=1}^{n} (y_i - f_0(x_i))^2$$

$$= -\frac{2}{2}(y_i - f_0(x_i)) \Rightarrow r_{i1} = f_0(x_i) - y_i$$

$$r_{11} = f_0(x_1) - y_1 = 142 - 92$$
$$r_{21} = f_0(x_2) - y_2 = 142 - 144$$
$$r_{31} = f_0(x_3) - y_3 = 144 - 91$$

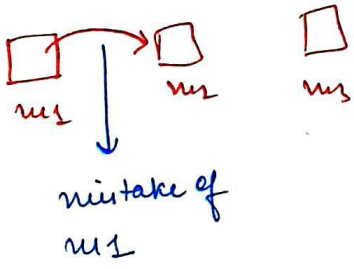
} Value from table

step 1 ↑ step 2 (a) ↑

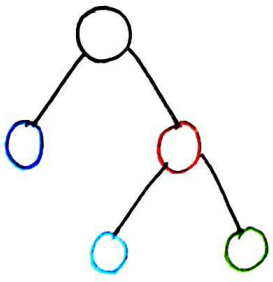
	R&D	Ops	Marketing	Profit	$f_0(x)$	R_1
0	165	137	472	192	142.3	49.6
1	101	92	250	144	142.3	1.6
2	29	127	201	91	142.3	-51.3
.						

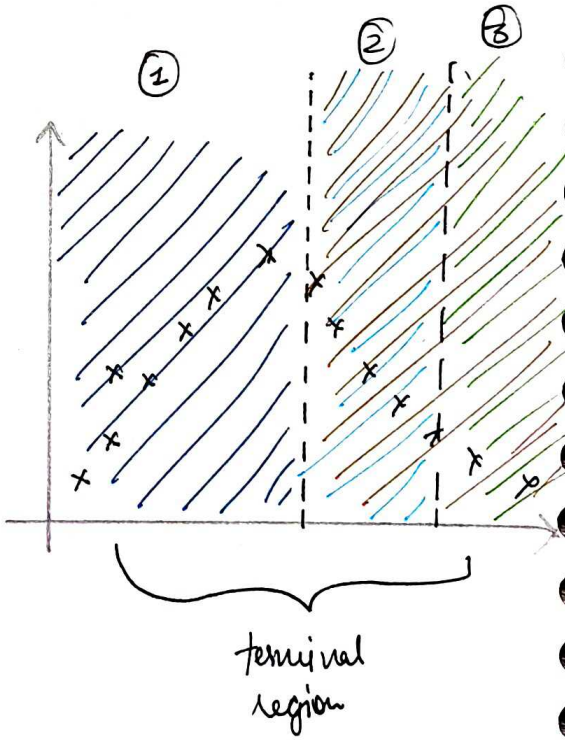
input — output

train decision tree as a input and output → step 2(b)



terminal region





$$\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

$$\gamma_{j1} = \underset{\gamma}{\operatorname{argmin}} \left[\sum_{x_i \in R_{j1}} L(y_i, f_0(x_i) + \gamma) \right]$$

we have to find value of γ which help to reduce the value of

* we accept those x value which fall on specific region.

ex:- $\gamma_{jm} = \gamma_{11}$ we accept those x value which fall on first region of terminal region.

Step 2 (c) ↓

$$\gamma_{11} = \underset{\gamma}{\operatorname{argmin}} L(Y_1, f_0(x_3) + \gamma)$$ (∵ not using Σ bcz we using only one row)

$$\gamma_{11} = \underset{\gamma}{\operatorname{argmin}} (Y_3 - f_0(x_3) - \gamma)^2$$ (∵ $L = (y_i - \hat{y}_i)^2$, $L = Y_3 - f_0(x_3) + \gamma)^2$

$$= \frac{\partial}{\partial \gamma} \neq \underset{\gamma}{\operatorname{argmin}} (Y_3 - f_0(x_3) - \gamma)^2$$

$$= -2 (Y_3 - f_0(x_3) - \gamma) = 0$$

$$Y_3 - f_0(x_3) - \gamma = 0 \Rightarrow \gamma = Y_3 - f_0(x_3)$$

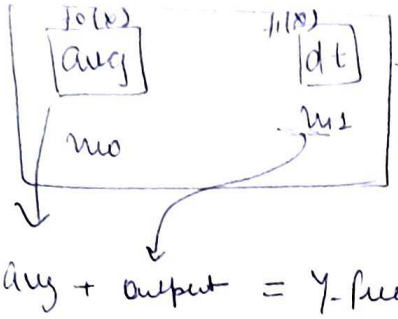
$$= 91 - 142$$

$$\boxed{\gamma = -51}$$

Step 2 (d) ↓

$$f_m(x) = f_{m-1}(x) + \left[\sum_{j=1}^{jm} \gamma_{jm} I(x \in R_{jm}) \right]$$ ← m1 output

$f_1(x) = f_0(x) +$ output of first decision tree



→ Let consider this our boosting model and train this model

→ (test data) → calculate output

$\hat{y} = f1(x)$

Avg + output = Y_pred

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$$

$$f5(x) = f4(x) + dt5$$

$$f4(x) = f3(x) + dt4$$

$$f3(x) = f2(x) + dt3$$

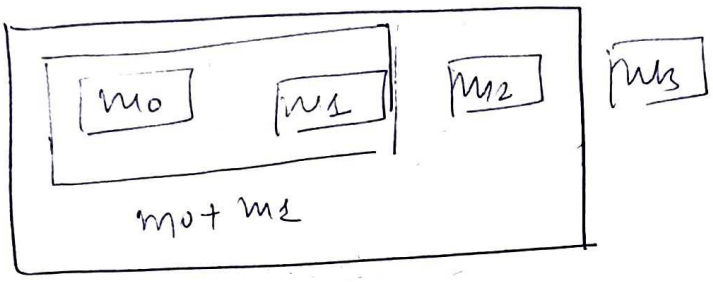
$$f1(x) + dt2$$

$$f0(x) + dt1$$

$$f5(x) = f0(x) + dt_1 + dt_2 + dt_3 = = - dt_r$$

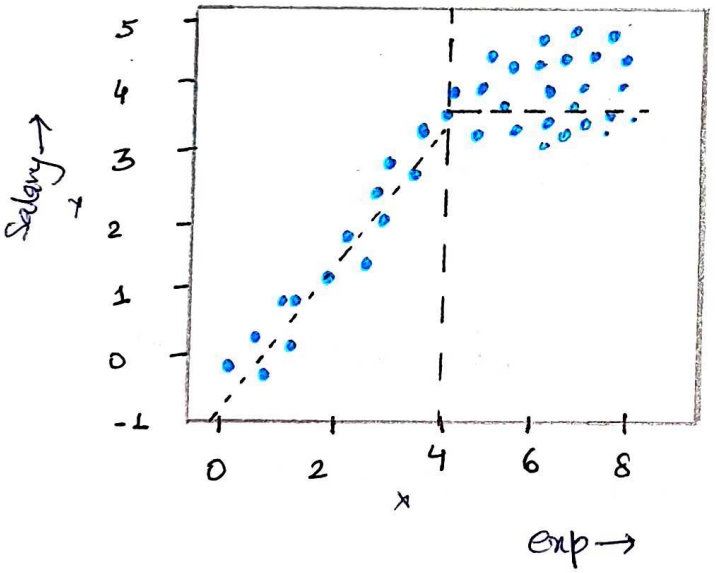
The what?

	R&D	OPS	Marketing	Profit	$f_0(x)$	r_1	$f_1(x_1)$	$y - f_1(x_1)$
0	165	137	472	192	142	–	–	
1	101	92	250	144	142	–	–	
2	29	127	201	91	142	–	–	
				↓ mean	↑			



m0 + m1 + m2 + m3

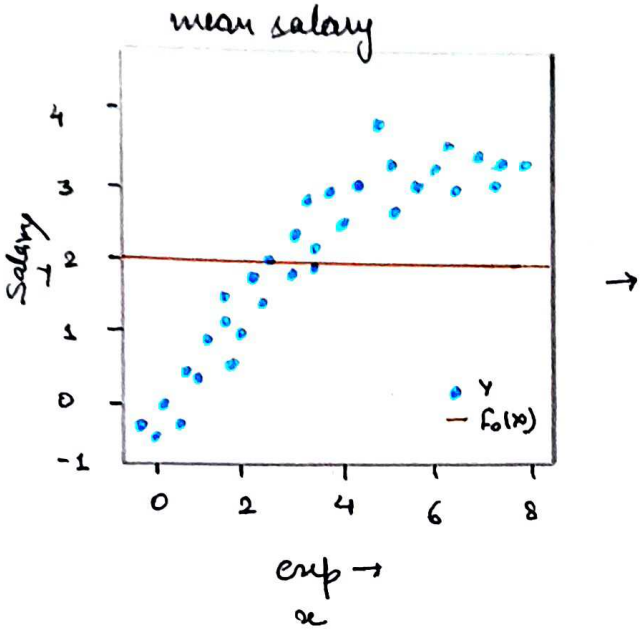
How Gradient Boosting Works for Regression?

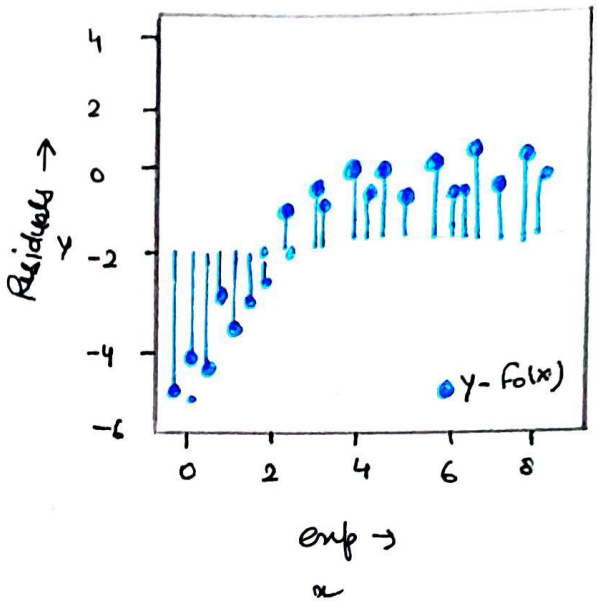


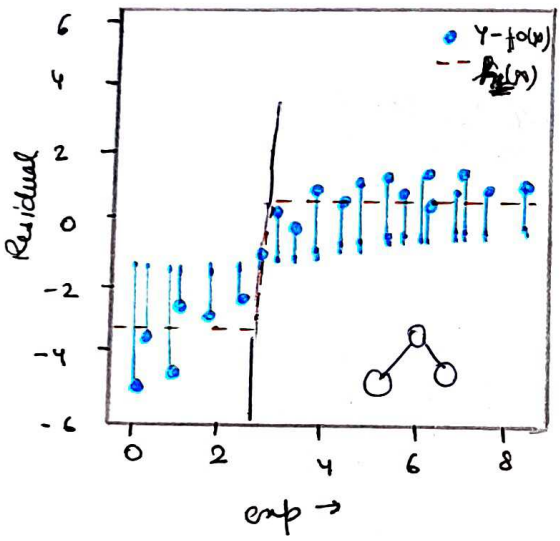
exp	Salary
7	32
3	18
1	5
–	–
–	–
–	–

we have to find

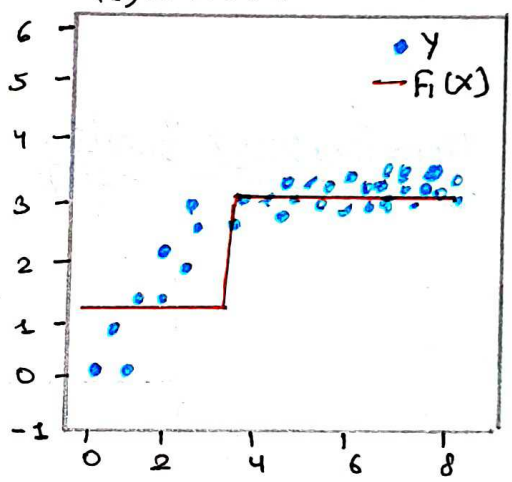
$$y = f x$$

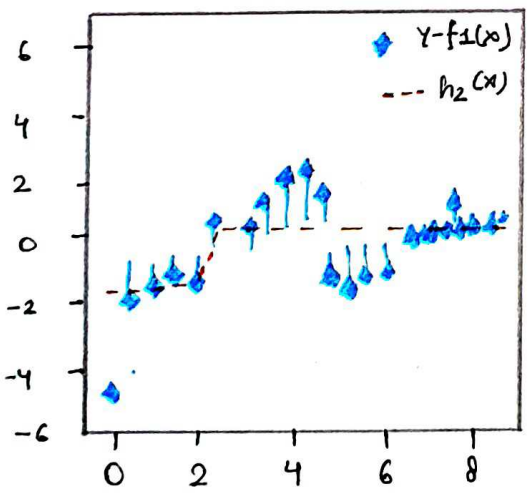




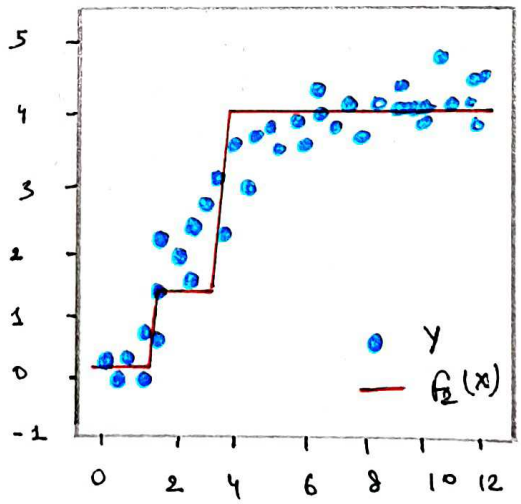


$f_1(x) = f_0(x) + h_1(x)$

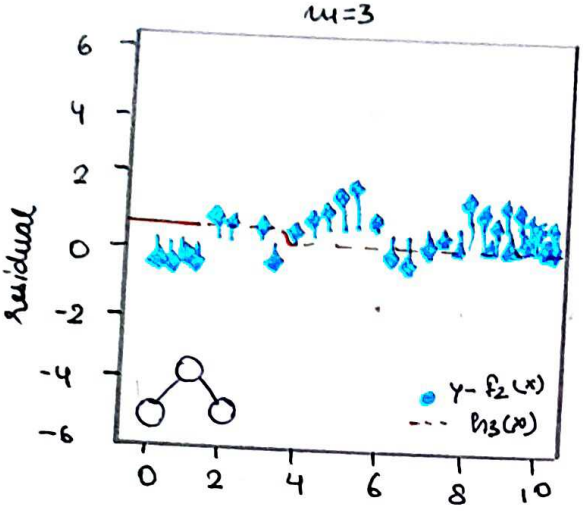




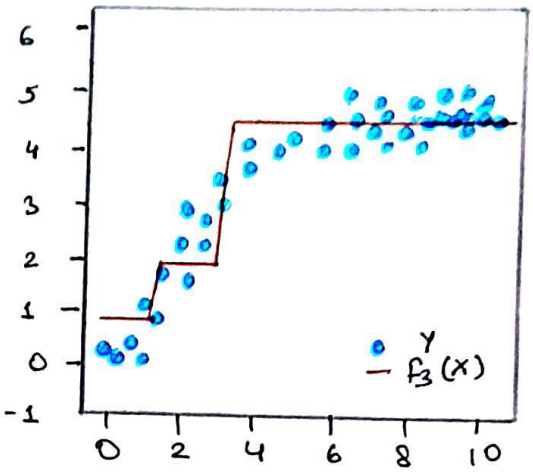
$f_2(x) = f_0(x) + h_1(x) + h_2(x)$



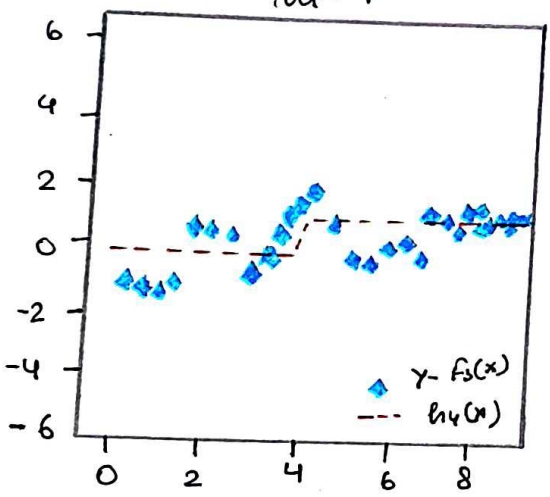
m=3



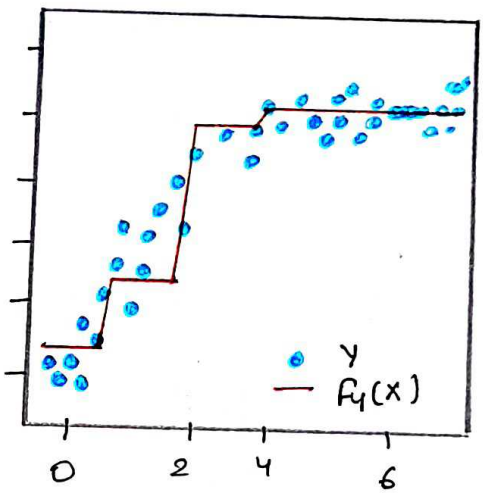
$f_3(x) = f_0(x) + h_1(x) + h_2(x) + h_3(x)$



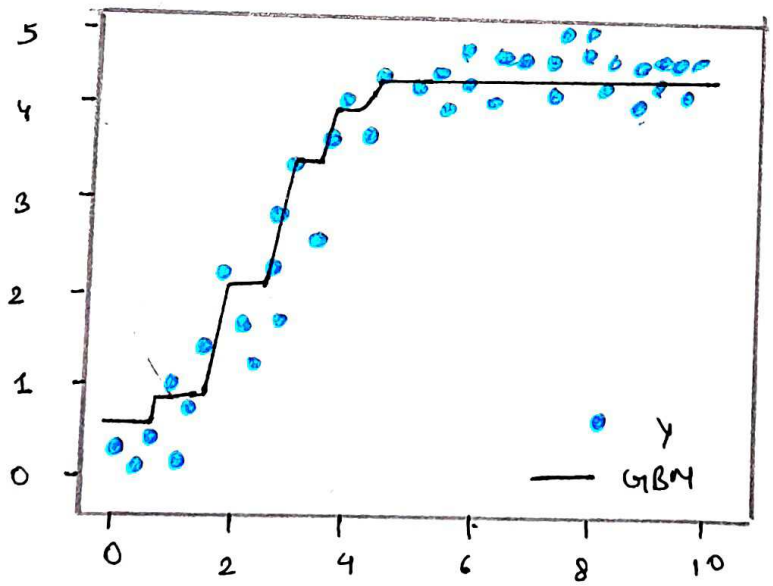
m=4



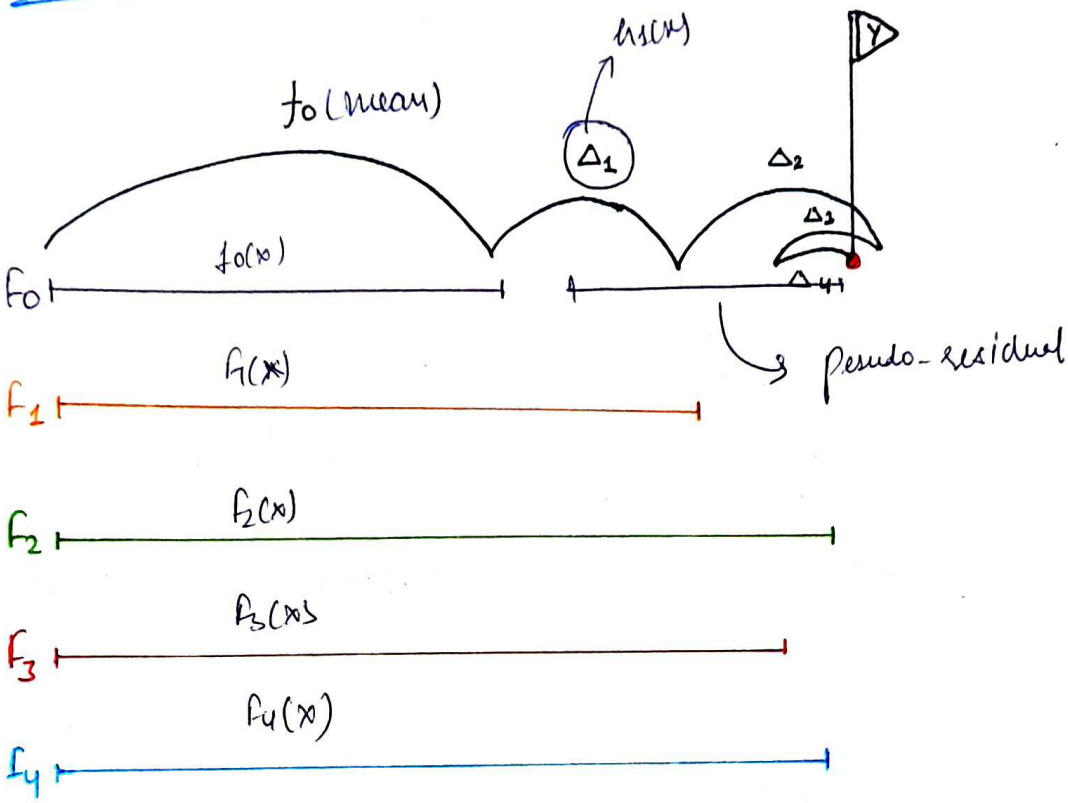
$f_4(x) = f_0(x) + h_1(x) + \cdots + h_4(x)$



10 Decision Tree



Intuition

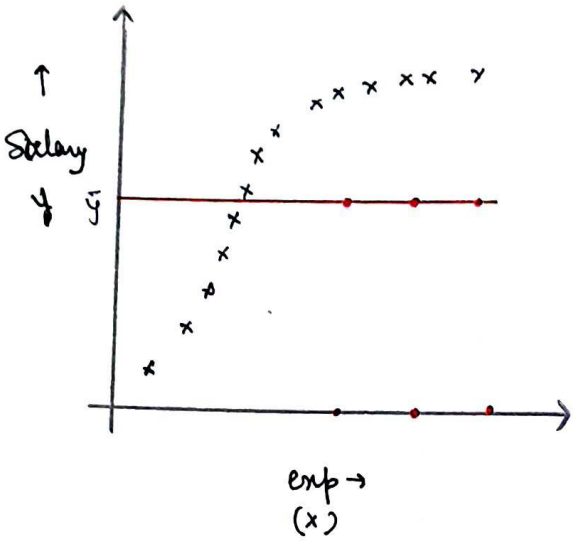


* Gradient Boosting is performing Gradient Descent in function space"

2. Function Space Vs Parameter Space

$y = f(x)$ ← 5000 (constant)

$y = f(x) = \bar{y}$ → y mean
→ improve
→ error
→ How to reduce error



error → Loss function

$L(y, \hat{y})$ → number

- → mse
- → logloss
- → hingeloss

exp	Salary	Pred
10	10	20
20	20	20
30	30	20

bcz mean output is all one same

MSE

$$L(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \frac{1}{3} \left[(10-20)^2 + (20-20)^2 + (30-20)^2 \right]$$

Function Space

$$L(y, \hat{y}) = \left(\frac{1}{n}\right) \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

* with $\frac{1}{n}$ is mean square error

* remove $\frac{1}{n}$ is squared error

$$L(y, \hat{y}) = \sum_{i=1}^{n} (y_i - y_i)^2 = (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \dots + (y_n + \hat{y}_n)^2$$

y_1: Constant (always present in data)

$\hat{y}_2$: Variable (may vary) depend on model function

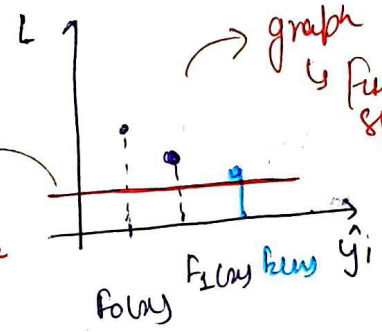
$L(\hat{y}_1, \hat{y}_2, \hat{y}_3 \cdots \hat{y}_n)$ → how many dims → $n+1$

↑ Math func

Loss function $\hat{y}_i$ (no items)

↳ $n+1$ dimension relationship

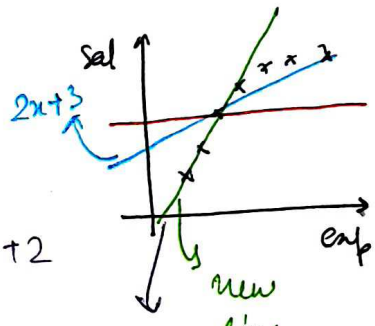
* We cannot draw $n+1$ dimension graphs, so, we assume 2-D graph



$f_1(x_i) = \hat{y}_i = 3x_i + 2$

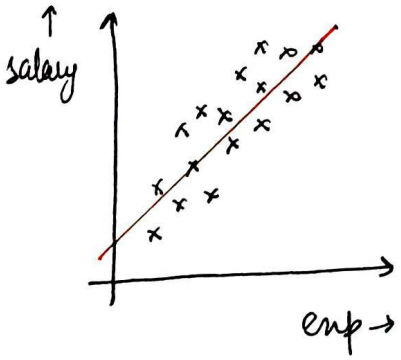
↳ new value

↳ new Loss value



$f_2(x_i) = \hat{y}_i = 2x_i + 3$

↳ new loss value



parametric

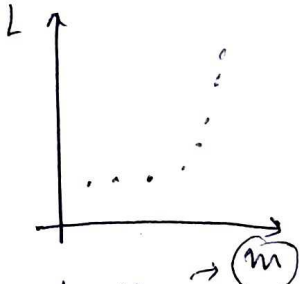
alg → Linear reg

$y = f(x) = \boxed{mx + b} = \text{err}$

↑ m, b

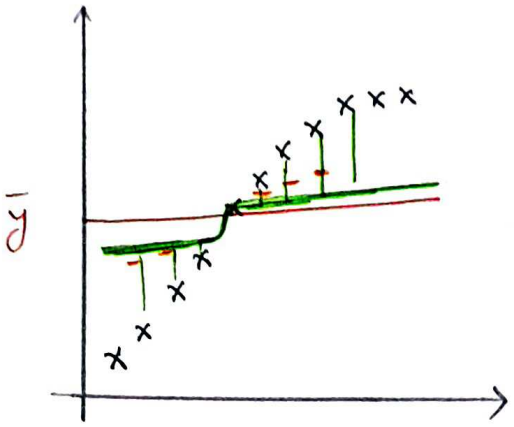
Log x

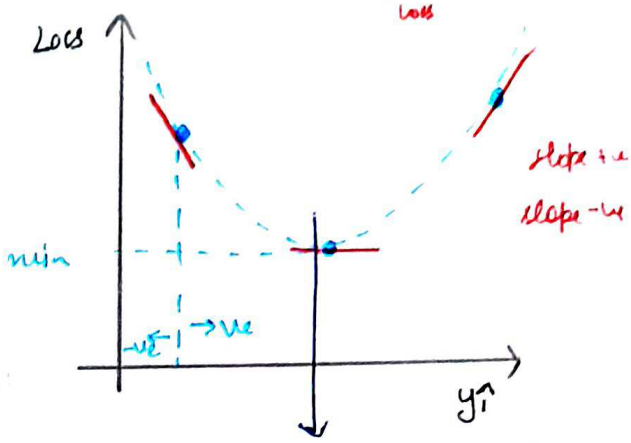
sine



* In parametric algo, we already know which func^n is used. We have to just changed the parameter and compare with loss function.

2. Direction of loss Minimization





This is 2D curve in real so, we have to find slope

$$\left[\frac{\partial L}{\partial \hat{y}_1}, \frac{\partial L}{\partial \hat{y}_2}, \ldots \frac{\partial L}{\partial \hat{y}_n}\right]$$

help to find the direction.

* This whole step are doing in 2nd step (a) (How gradient boosting works)

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

↓ Decision Tree

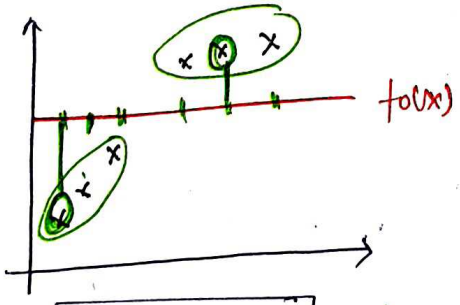
↓ solve

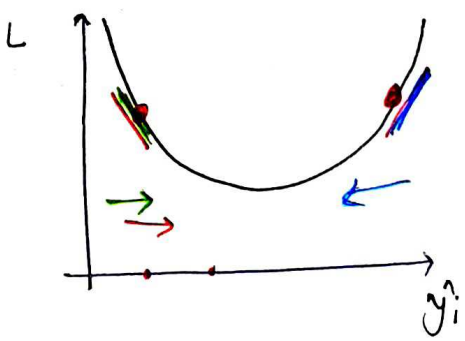
$$\boxed{y - f_0(x_i)} \rightarrow \text{slop}$$

→ $y_1 - f_0(x_1)$

→ $y_2 - f_0(x_2)$

→ $y_n - f_0(x_n)$

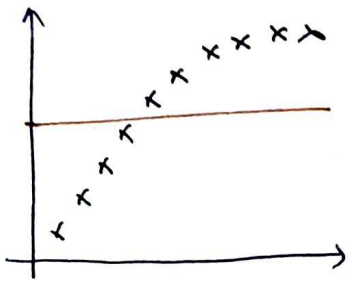


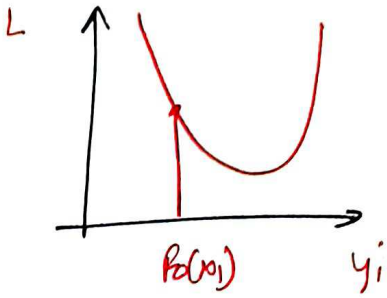


$\boxed{y_i - f_0(x_i)}$ → increase (+ve)

→ decrease (-ve)

3. Update the function





$$= -\left[\frac{dL}{d\hat{y}_i}\right] \rightarrow \text{direction}$$

$$\rightarrow \boxed{y_i - f_0(x_i)}$$

$f_1(x_i) = f_0(x_i) - \text{slope}$

$f_1(x_i) = f_0(x_i) + y_i - f_0(x_i)$

$\boxed{m_{new} = m_{old} - \text{slop}}$

↳ acc. to Gradient descent to update function.

$f_{0(x)} = m_{old}$

$f_1(x) = m_{new}$

$$f_1(x_i) = f_0(x_i) + y_i - f_0(x_i)$$

$\boxed{f_1(x_i) = y_i}$ → output of the data (Same data output assign in new function)

~~Massive overfitting~~

* That's why we not use Gradient descent method.

$$\text{Slope} = y_i - f_0(x_i) = a$$

$$h_1(x_i) \simeq a$$

↳ approx equal to because if equal to a then again cancel the slop with old function.

X | Residuall → weak ↘

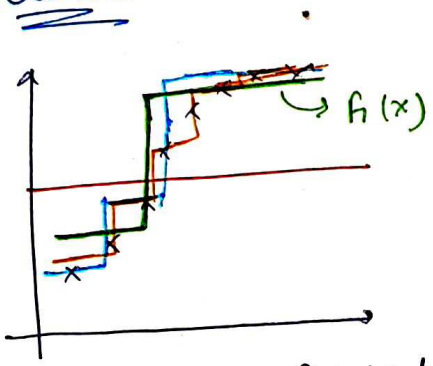
$h_1(x)$ → predict weak or not perfect

X

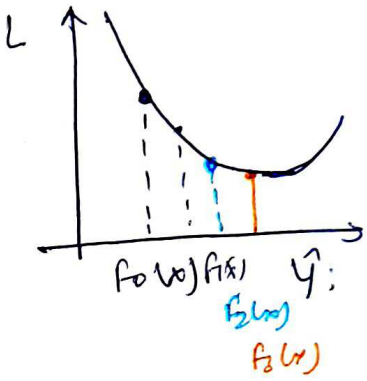
* If predict = not perfect than not cancel with Old function.
  ↳ not perfect but in right direction.

$$f_1(x) = f_0(x) + h_1(x)$$

4. Iterate



$f_1(x) = f_0(x) + h_1(x)$



X → residual → $h_2(x)$

$f_2(x) = f_0(x) + h_1(x) + h_2(x)$

X → residual → $h_3(x)$

$f_3(x) = f_0(x) + h_1(x) + h_2(x) + h_3(x)$

Difference Between Gradient Boosting and Gradient Descent

2) Gradient Descent → Parameter Space
   Gradient Boosting → function Space

$m_n = m_0 - \eta \text{ slope}$

$$f_m(x) = f_{m-1}(x) + \eta h_m(x)$$

Advantages

* Use any loss function, it give same result.
  mean squared error, mean absolute error, log loss. . .

* we can use any weak model itead of $h_m(x)$
  decision tree.

Gradient Boosting for Classification

We use same steps of Regression Gradient Boosting in classification. One difference is → log loss instead of use
↳ for classification

cgpa	iq	is_placed
6.82	118	0
6.36	125	1
5.39	99	1
5.50	106	1
6.39	148	0
9.13	148	1
7.17	147	1
7.72	72	0

$f_0(x)$ ↳ simple model

↳ log(odds) → $\log\left(\frac{\text{no. of 1}}{\text{no. of 0}}\right)$

$\log\left(\frac{5}{3}\right) = f_0(x)$ → 1st model

$\log_e$ ↳ 0.51

cgpa	iq	is_placed	pre1(log-odds)
6.82	118	0	0.51
6.36	125	1	0.51
5.39	99	1	0.51
5.50	106	1	0.51
6.39	148	0	0.51
9.13	148	1	0.51
7.17	147	1	0.51
7.72	72	0	0.51

* is_placed is kind of probability. So, we cannot combine is_placed with pre1(log-odds). We have to find probability of column pre1(log-odds).

We use the formula

$$P = \frac{1}{1+e^{-\text{log(odds)}}} = \frac{1}{1+e^{-0.51}} = 0.62$$

Cgpa	iq	is placed	pre1 (log-odd)	pre1 (probability)
6.82	118	0	0.51	0.62 → 1
6.36	125	1	0.51	0.62 → 1
5.39	99	1	0.51	0.62 → 1
5.50	106	1	0.51	0.62 → 1
6.39	148	0	0.51	0.62 → 1
9.13	148	1	0.51	0.62 → 1
7.17	147	1	0.51	0.62 → 1
7.22	72	0	0.51	0.62 → 1

0.62

↳ In classification prob output is 0 or 1

Let assume threshold is 0.5 (1 / 0 — 0.5)

* Calculate pseudo residual

res1 = output − prediction (pre1 probility)

X: Cgpa, iq — Y: res1

Cgpa	iq	is_placed	pre1 (log-odd)	pre1 (probability)	res1
6.82	118	0	0.51	0.62	-0.62
6.36	125	1	0.51	0.62	0.37
5.39	99	1	0.51	0.62	0.37
5.50	106	1	0.51	0.62	0.37
6.39	148	0	0.51	0.62	-0.62
9.13	148	1	0.51	0.62	0.37
7.14	147	1	0.51	0.62	0.37
7.22	72	0	0.51	0.62	-0.62

→ Now, we can apply decision Tree on X and Y

where X is Cgpa and iq and Y is res1

→ Use decision tree on X and Y.

↳ regression tree bcz Y is number

using regression tree in classification

weak learner → leaf node = 3

node = #0
cgpa <= 6.375
Squarred_error = 0.234
Samples = 8
value = 0.0

node#1
Squarred error = 0.0
Samples = 3
value = 0.375

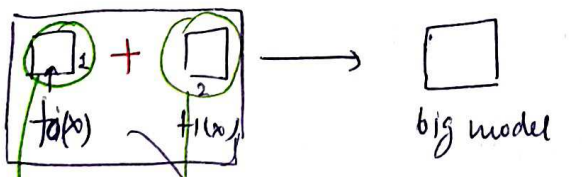
node #2
iq <= 132.5
Squarred error = 0.24
Samples = 5
value = -0.225

node #3
Squarred error = 0.0
Samples = 2
value = 0.625

node #4
Squarred_error = 0.222
Samples = 3
value = 0.042

In regression these value are output but in classification these values are prob. bcz we train our model on differnce of prob. ↴

→ And we cannot add these prob with log odds. So, we have to convert prob into log odds.



How to predict value using 2 model?

using y = odd log

using y = diff of prob

log odd

→ We have to add the pred of model 1 and model 2

↳ diff of probability

$$\text{Log(odds)} = \frac{\sum \text{Residuals}}{\sum(\text{Previous Prob} * (1 - \text{Previous Prob}))}$$

↳ Use this formula to every node to change value from ~~log~~ Probability into Logodds.

Cgpa	iQ	Is-placed	Pre(log-odds)	Pre(Prob)	Resi	leaf-entry
6.82	-	-	-	0.625	-0.625	3
-	-	-	-	-	-	1
-	-	-	-	-	-	1
-	-	-	-	-	-	1
-	-	-	-	-	-	4
-	-	-	-	-	-	4
-	-	-	-	-	-	4
-	-	-	-	0.625	-0.625	3

We find which row fallen on which leaf node using Sklearn

ex:- 1 row → 3 leaf node
2 row → 1 leaf node
5 row → 4 leaf node

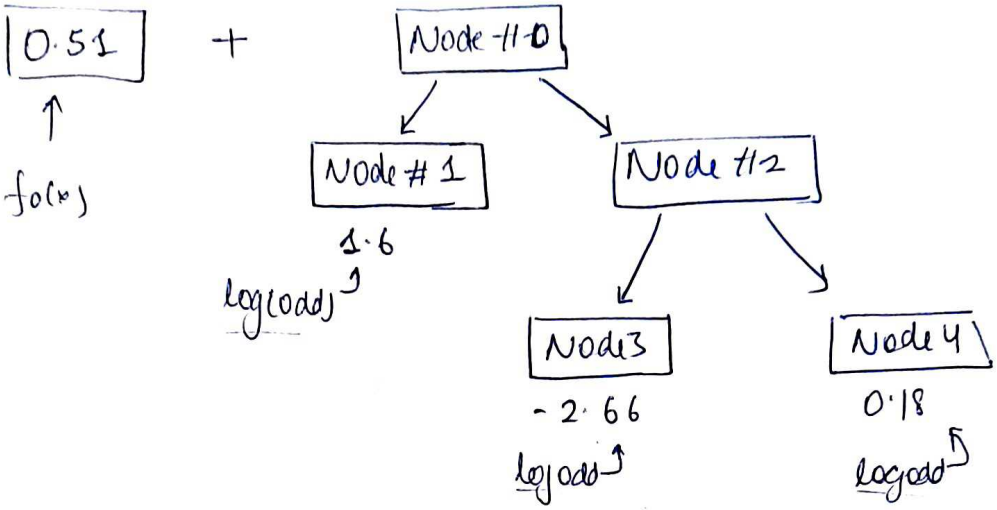
Let's calculate the log(odds) of node 3.

$$\log(\text{odd}) = \frac{-0.625 - 0.625}{0.625(1-0.625) + 0.625(1-0.625)}$$

$$= \frac{-2 \times 0.625}{(0.625 \times 0.375) + (0.625 \times 0.375)}$$

$$= \frac{-2 \times 0.625}{2 \times 0.625 \times 0.375} = -\frac{1}{0.375} = -2.66$$

log odd of leaf node 3



Cgpa	id	Is-placed	pres1(log-odd)	pres(prob)	ress	Leaf-entry 1	Pre2(log-odd
5.82	118	0	0.51	0.625	-0625	3	-2.15
-					-	-	2.11
-					-	-	2.11
-					-	-	2.11
-					-	-	0.69
-					-	-	0.69
-					-	-	0.69
7.72	72	0	0.51	0.625	-0625	3	-2.15

row 1 = 0.51 + (-2.66) → 3 leaf node = -2.15

fo(x0) ↓ dt(log odd) ↓

* Now again convert Pre2(log-odd) into prob to find out residual (Res)

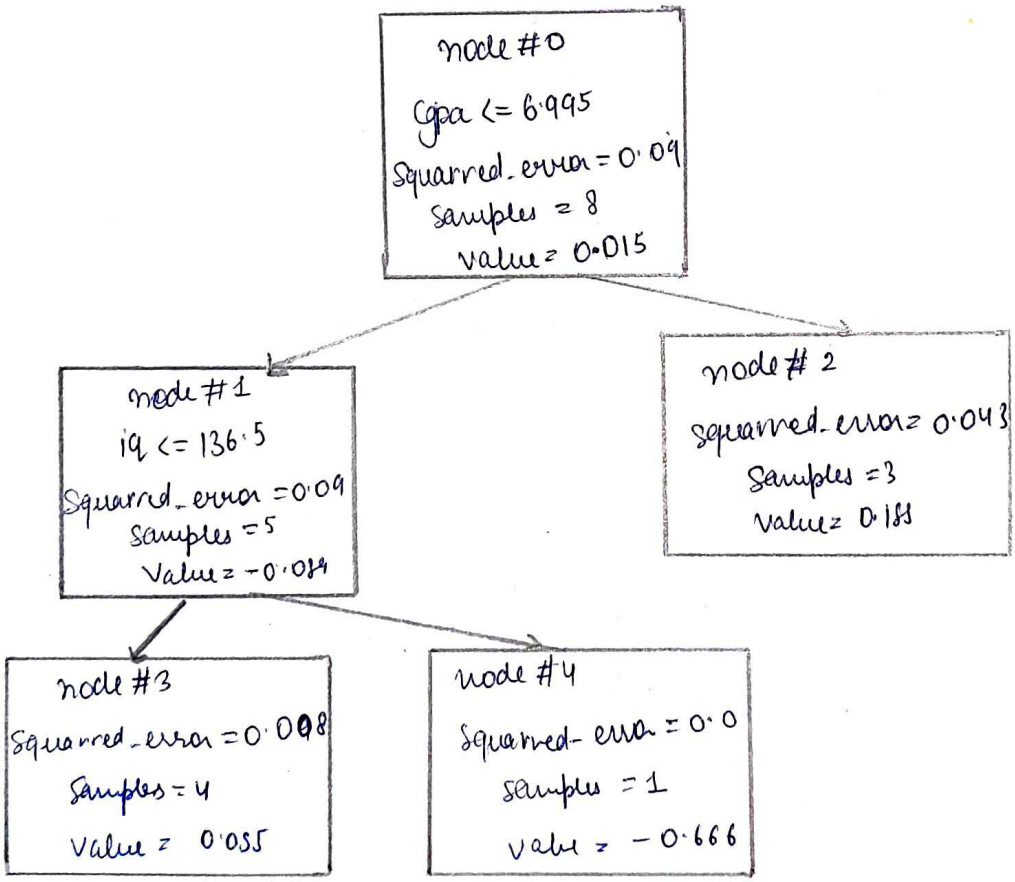
Probab (Pre2(log-odd) Prob = $\frac{1}{1+e^{-\text{log odd}}}$

cgpa	iq	is_placed	pre1(log-odd)	pres(prob)	res1	leaf_entry1	pre2(log_odd)	pre2(prob)	res2 → Y
X								0.103472	-0.103472
								0.891951	0.108049
								0.891951	0.108049
								0.891951	0.108049
								0.666151	-0.666151
								0.666151	0.333849
								0.666151	0.333849
								0.103472	-0.103472

res2 = is_placed − pre2(prob)

↳ residual (error)

Now again train second decision Tree with X (cgpa, iq) and Y (res2)



* If big difference between res1 and res2 then we can multiply with learning rate (η).

ex:- res1 = -0.625 res2 = -0.103

first row ↑

while finding pre2 (log odds) we can add

pre2 (log odd) = 0.51 + η (1.6)

(0.1)

→ Again we have to find the log odd of second decision tree using this formula:

$$\frac{\sum \text{Residual}}{\sum (\text{Previous Prob} * (1 - \text{Previous Prob}))}$$

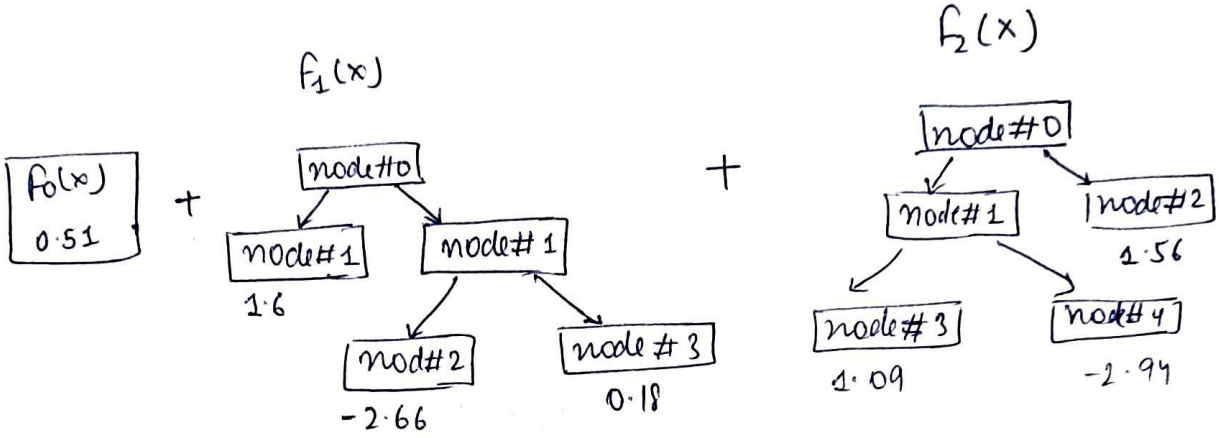
→ Again we have to find which row fallen from which node leaf.

CgPg	Pre2 (prob)	res2	leaf - entry
-	0.10	-0.1034	3
-	0.89	0.1080	3
-	0.89	0.1089	3
-	0.89	0.1089	3
-	0.66	-0.6661	4
-	0.66	0.3338	2
-	0.66	0.3338	2
-	0.103	-0.1034	2

Let's calculate the node 4 log odd value

$$\text{log odd} = \frac{-0.66}{0.66(1-0.66)} = \frac{-1}{0.34} = -2.94$$

→ We can find all leaf node value using same step



→ Add log odd of Fo(x) and f1(x)

Pre2 (log-odd) + loggodds of row (leafentry)

Ex:- Row1 ⇒ $-2.15 + \frac{(3)\text{ Node}}{1.09} = -2.15 + 1.09$

$= -1.06$

Row2 ⇒ $2.11 + 3(\text{Node}) = 2.11 + 1.09 = 3.20$

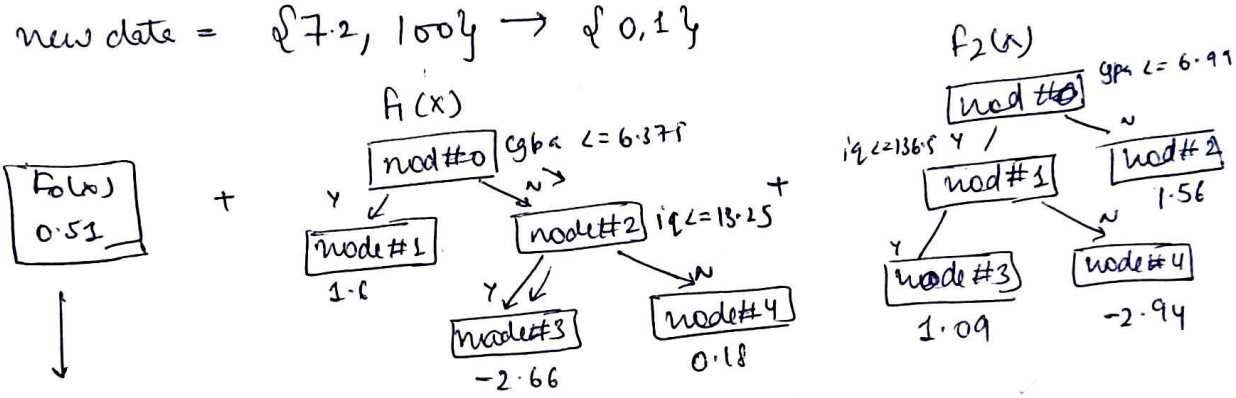
→ We have to find probability of log odd value.

so we use = $P = \frac{1}{1+e^{-\text{log odds}}}$

cgpa	leaf-entry2	pre3(log odds)	pre3(prob)
-	3	-1.068	0.255
-	3	3.201	0.960
-	3	3.201	0.960
-	3	3.201	0.960
-	4	-1.798	0.1420
-	2	2.516	0.904
-	2	2.516	0.904
-	2	-0.598 3	0.354

Prediction

new data = {7.2, 100} → {0, 1}



0.51 + (-2.66) + 1.56 = -0.59 → log odds

$$\text{Prob} = \frac{1}{1+e^{-\text{log odd}}} = \frac{1}{1+e^{0.59}} = 0.35$$

If threshold is 0.5 then output is 0.

because Prob < 0.5

Maths Formulation

Step 0 → The loss function

$$\text{Log Loss} = L = -\frac{1}{n}\sum_{i=1}^{n} y_i \log p_i + (1-y_i)\log(1-p_i)$$

cgpa	iq	placement	$\hat{y}$ (p_i)
8	90	1	0.62
7	70	0	0.52
6	60	1	0.51

$$L = -\frac{1}{n}\sum_{i=1}^{n} y_i \log p_i + (1-y_i)\log(1-p_i)$$

$$L = -\frac{1}{n}\sum_{i=1}^{n} y_i \log p_i + \log(1-p_i) - y_i \log(1-p_i)$$

$$= -\frac{1}{n}\left[\sum_{i=1}^{n} y_i\left[\log p_i - \log(1-p_i)\right] + \log(1-p_i)\right]$$

$$= -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log(\text{odd}_i) + \log(1-p_i)\right]$$

↗ log odd

$$\log\left(\frac{p_i}{1-p_i}\right)$$

$$= -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log(\text{odd}_i) + \log\left(1 - \frac{e^{\log(\text{odd}_i)}}{1+e^{\log(\text{odd}_i)}}\right)\right]$$

$$p_i = \frac{e^{\log(\text{odd}_i)}}{1+e^{\log(\text{odd}_i)}}$$

$$= -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log(\text{odd}_i) + \log\left(\frac{1}{1+e^{\log(\text{odd}_i)}}\right)\right]$$

$$= -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log(\text{odd}_i) + \log 1 - \log\left(1 + e^{\log(\text{odd}_i)}\right)\right]$$

$$L = -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log(\text{odd}_i) - \log\left(1+e^{\log(\text{odd}_i)}\right)\right]$$

↳ for entire data

for single data ↴

$$L = -y \log(\text{odds}) + \log\left[1 + e^{\log(\text{odds})}\right]$$

Step 1 → Finding $F_0(x)$

1. Initialize $f_0(x) = \text{argmin}_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$ ← Loss function.

$$L = -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log(\text{odds}_i) - \log\left(1+e^{\log(\text{odds}_i)}\right)\right]$$

$L(y_i, \log(\text{odd}_i))$ → $L(y_i, \gamma)$

$$L = \text{argmin}_\gamma -\frac{1}{n}\left[\sum_{i=1}^{n} y_i\gamma - \log(1-e^{\gamma})\right]$$

$$\frac{\partial L}{\partial \gamma} = -\frac{1}{n}\left[\sum_{i=1}^{n} y_i - \frac{e^{\gamma}}{1+e^{\gamma}}\right] = 0$$

$$= \underbrace{-\frac{1}{n}\sum_{i=1}^{n} y_i}_{\downarrow} + \underbrace{\frac{1}{n}\sum_{i=1}^{n} \frac{e^{\gamma}}{1+e^{\gamma}}}_{\downarrow} = 0$$

$$= -p_{avg} + \frac{n}{n}\frac{e^{\gamma}}{1+e^{\gamma}} = 0$$

$$\frac{e^{\gamma}}{1+e^{\gamma}} = p_{avg}$$

$e^r = P_{avg} + Pe^r \Rightarrow e^r - P_{avg}e^r = P_{avg}$

$e^r(1-p_{avg}) = P_{avg}$

$e^r = \frac{P_{avg}}{1-p_{avg}}$

$$\boxed{r = \log\left(\frac{p_{avg}}{1-p_{avg}}\right)} \rightarrow f_0(x) = r = \log\left(\frac{p_{avg}}{1-p_{avg}}\right)$$

Cgpa	iq	is_placed
6.82	118	0
6.36	125	1
5.39	99	1
5.50	106	1
6.39	148	0
[illegible]	[illegible]	1
9.13	148	1
7.17	147	1
7.72	72	0

$p_{avg} = \frac{5}{8}$

$\log\left(\frac{5/8}{1-5/8}\right) = \log(3/8)$

In Gradient Boosting Classification we used $\log(3/8)$ $f_0(x)$

Step2 → Pseudo Residuals

2. For $m = 1$ to M:

(a) For $i = 1, 2, \ldots N$ Compute

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

$$L = -y \log(odds) + \log\left(1 + e^{\log(odds)}\right)$$

$f(x) \rightarrow f(x) + \log odds$

$$\frac{\partial L}{\partial \log odds} = -y + \boxed{\frac{e^{\log(odds)}}{1 + e^{\log(odds)}}} \rightarrow \text{prob output}$$

$$= -y + p = y - p = \boxed{y_i - p_i} \rightarrow \text{log odds}$$

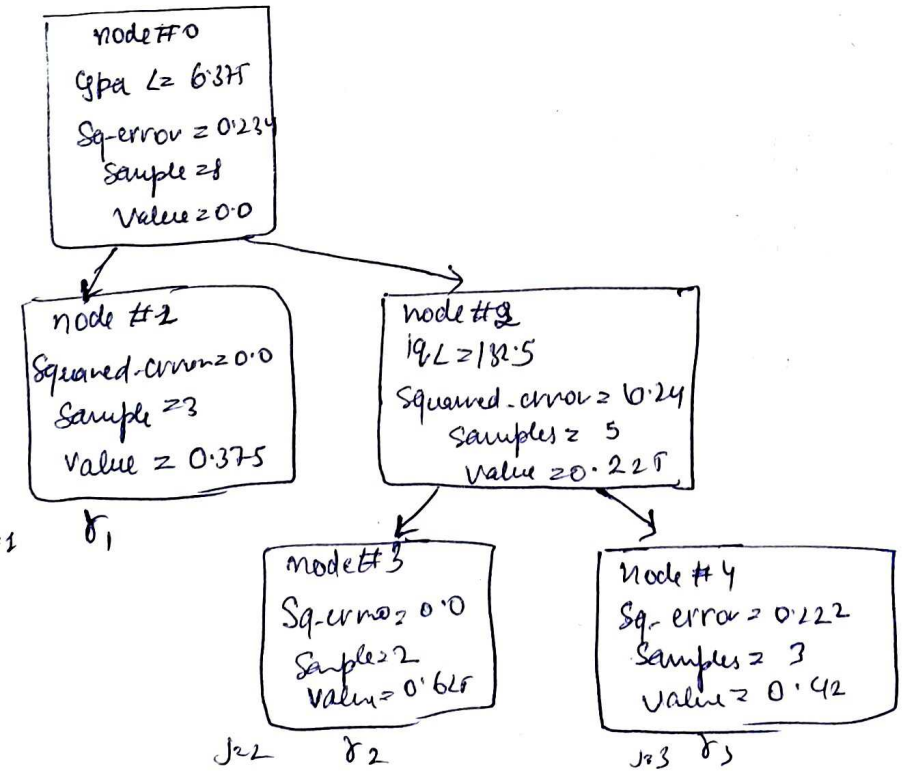
Step 2.b → Train Regression Tree

There's no mathematical formula we have to just train on Regression Tree.

Step 2.c → Compute lambda for all leaf nodes

(C) → For $j = 1, 2, \ldots J_m$ Compute

$$\gamma_{jm} = \arg\min \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$



gfg	iq	is placed	pu1 (log-odds)	pu1 (prob)	xess	leaf-entys
					-0.62	3
					0.375	1
					0.375	1
					0.375	1
					-0.625	4
					0.375	4
					0.375	4
					-0.625	3

→ Find γ for node leaf 2.

$$\gamma_2 = \underset{\gamma}{\operatorname{argmin}} \left[\underbrace{L(y_1, f_0(x_1) + \gamma)}_{1} + \underbrace{L(y_2, f_0(x_2) + \gamma)}_{2} \right] \gamma'$$

$\frac{\partial L'}{\partial \gamma} = 0$ → minimum value

using Taylor series

$X \to f_0(x_1) + \gamma$

$a \to f_0(x_1)$

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots$$

$$\Rightarrow L(y_1, f_0(x_1)) + \frac{\partial L}{\partial f_0(x_1)} \gamma + \frac{1}{2} \frac{\partial^2 L}{\partial f_0(x_1)^2} \gamma^2$$

$$\Rightarrow \frac{\partial L}{\partial y_1} = \frac{\partial L}{\partial f_0(x_1)} + \frac{1}{2} \gamma \frac{\partial^2 L}{\partial f_0(x_1)^2} = 0$$

$$\Rightarrow \frac{\gamma \partial^2 L}{\partial f_0(x_1)^2} = -\frac{\partial L}{\partial f_0(x_1)}$$

$$r = \frac{\frac{-\partial L}{\partial fo(x_1)}}{\frac{\partial^2 L}{\partial fo(x_1)^2}}$$

→ Numinator ↓

$$\frac{\partial L}{\partial fo(x_1)} = -y_1 + \frac{e^{\log(odd)}}{1+e^{\log(odd_1)}} \rightarrow p$$

* Same step for 2.

$$= y_1 - p$$

$$r = \frac{y_1 - p}{\frac{\partial^2 p}{\partial fo(x_1)^2}}$$

denominator

$$\frac{\partial}{\partial fo(x_1)} \frac{\partial L}{\partial fo(x_1)} = \frac{\partial}{\partial \log(odds_1)} y_1 - \frac{e^{\log(odds_1)}}{1+e^{\log(odds_1)}}$$

$$= \frac{\partial}{\partial \log(odds_1)} \left[y_1 - e^{\log(odds_1)} \left(1+e^{\log(odds_1)}\right)^{-1} \right]$$

$x \quad y \quad$ → $x'y + xy'$

$$= -\left[e^{\log(odds_1)} \left(1+e^{\log(odds_1)}\right)^{-1} - e^{\log(odds_1)} \left(1+e^{\log(odds_1)}\right)^{-2} \right]$$

$$= \frac{e^{\log(odds_1)}}{1+e^{\log(odds_1)}} - \frac{e^{\log(odds_1)}}{\left(1+e^{\log(odds_1)}\right)^2}$$

$$= \frac{e^{\log(odds_1)}}{1+e^{\log(odds_1)}} \left[\frac{1}{1+e^{\log(odds_1)}} - 1 \right] \rightarrow \frac{1}{1+e^{\log(odds_1)}}$$

$$= \frac{e^{\log(odds_1)}}{1+e^{\log(odds_1)}} \cdot \frac{1}{\left(1+e^{\log(odds_1)}\right)}$$

p ↓ $1-p$

$$\gamma = \frac{\text{Residual}_1}{P_1(1-P_1)}$$

$$\gamma_2 = \frac{\sum \text{residual}_i}{\sum \text{Prob}_i(1-\text{Prob}_i)}$$ Proceed

↳ Leaf node 2

Step 2 D – Update the model

$$f_1(x) = f_0(x) + \text{output from dt} \begin{cases} \text{same} \\ \text{leaf} \\ \text{node} \end{cases}$$

$$f_2(x) = f_0(x) + f_1(x) + \text{Outp from dt}$$

Step 3 :- Final Model

$$f_M(x) = f_{m-1}(x) + \text{last decision output}$$

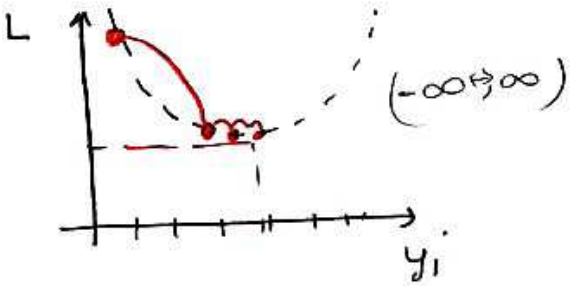
Boosting Model

Logg (odds) Vs Probability

Unconstrained Prediction Space : Log odds can span the entire real line (−∞ to ∞), while probabilities are constrained between 0 and 1, Algorithms like gradient boosting involve adding correction (via the weak learner) to the prediction iteratively. If you're working in the log odds space, there's no need to worry about your predictions going out of bounds.

Better Gradients : When computing gradients (which guide the addition of new trees in boosting), the gradients can be more informative and have better magnitudes in the log odds space, than in the probability space, especially when probabilities are near 0 or 1.

Logg (odds) → Big jumps



prob → small jumps