BAGGING RECAP

- Aggregation of **Base Learners** → **Base Learners: Low Bias & High Variance**
- Reduce variance using **Randomization** → **Row & Column Sampling**

BOOSTING INTRODUCTION

Base Learners → **High Bias & Low Variance**

How to combine base learners ?

Aggregation ? - Reduces Variance

But, variance is already low

What do we use then ? - Additive Combining

POINTS TO REMEMBER

Base Learners

↓

High Bias & Low Variance

Boosting Intuition - How to combine base learners ?

Let's understand it using a **regression** example :

#	Height	Gender	Weight(y)
1	1.6	M	82
2	1.5	F	55
3	1.4	F	61
4	1.4	M	65

Given : **Features**

1. Height
2. Gender

Predict : **Weight**

$$\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$$

On this **training data** , we build our model in step.

STEP 0: M_0

On training data, build a model M_0

M_0

↓

High bias, low variance model

I.e Simplest Model

Simplest Model ?

Mean Model → **Take average of all y values**

mean model
(high bias
low variance)

$\{x^i, y^i\}$ → fit model → M_0

D_{Train}

$\hat{y} \Rightarrow mean(y^{(i)})$

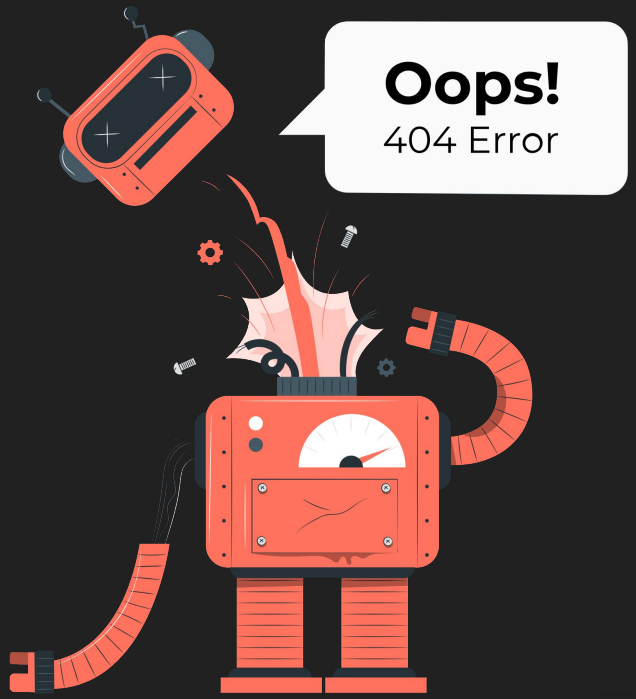
Let's represent

$\hat{y}$

here as

$h_0(x)$

Each prediction will have an **error** associated with it



For every point,

$$err^{(i)} = y^{(i)} - \hat{y}^{(i)}$$

$y^{(i)}$ → Actual

$\hat{y}^{(i)}$ → Predicted = $h_0(x)$

So, $err^{(i)} = y^{(i)} - h_0(x)$

As it is a mean model → **Large Training Data**

Using this,

Actual value, $y^{(i)}$ can be represented as

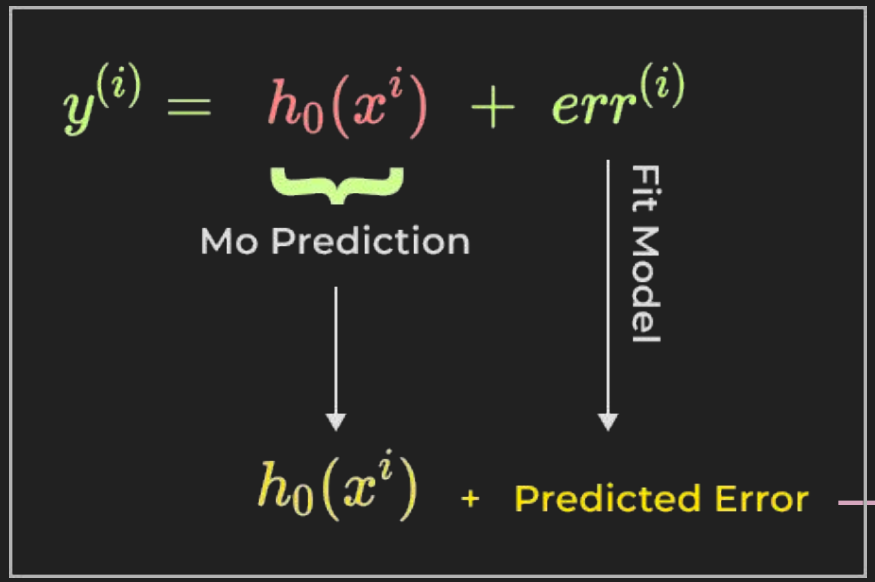
$$y^{(i)} = h_0(x^i) + err^{(i)}$$

M_0 Prediction

I.e Combination of Prediction & errors

GOAL : Reduce the error

$$y^{(i)} = h_0(x) + err^{(i)}$$ ↓ ↓ Reduce



How to reduce error?

Fit a new model to predict the error & add that prediction to $h_0(x^i)$

Better than previous prediction

How would predicting the **error** help in **reducing** it ?

Suppose, predicted error = e_{pred}

Total Prediction = $\underbrace{h_0(x^i) + e_{pred}}$

Closer to actual prediction

than $h_0(x^i)$ alone

Conclusion : We are reducing the error by predicting it & adding it to previous prediction

POINTS TO REMEMBER

Base Learners → **High Bias & Low Variance**

Combine learners using **additive combining.**

STEP 0: Mo → **Mean Model**

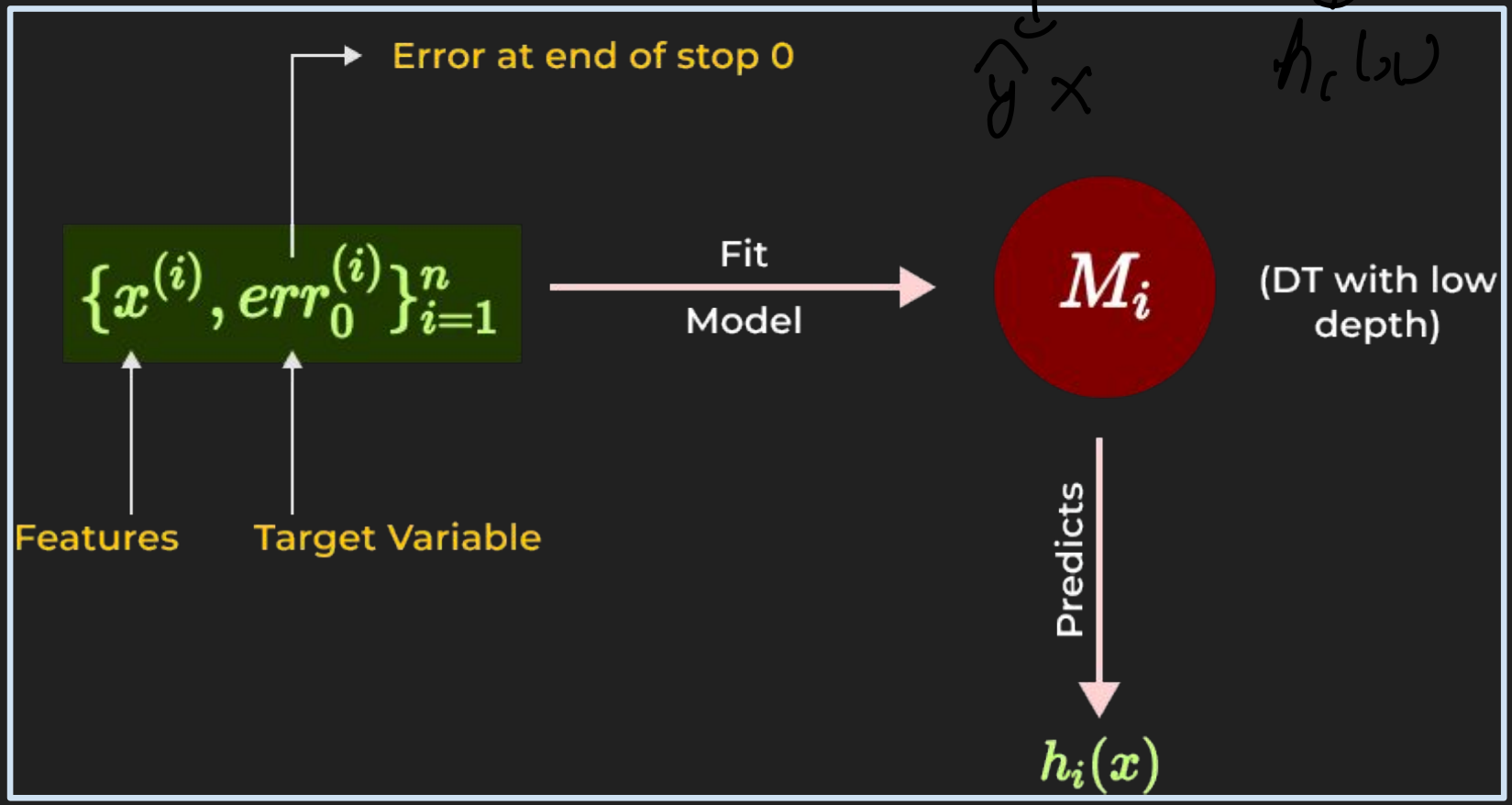
STEP 1: M_1

In this step, we fit a model (M_1) on error of previous step (err_0^i)

What model shall we use as (M_1) ?

- Can be any model as long as it is high bias low variance.
- We will consider **DECISION TREES**
- DT will be of low depth

The process will look like :



Now, we need to add prediction of step 0 & step 1

Additive Combination

The final model $F_1(x)$ (model at the end of Stage 1) will be defined as :

$$F_1(x) = h_0(x) + h_1(x)$$

Prediction at stage 0

Prediction at stage 1

Issue with **simple** addition ?

There is a possibility that

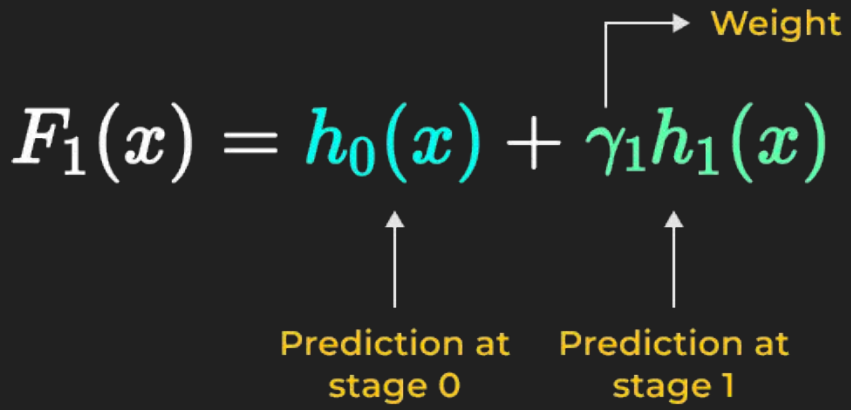
M_1 model has large residual

Not contributing much to final prediction

We need a mechanism to control the model contribution in final prediction.

How to do that ?

→ Multiply the model pred with weight values



Lower Residual ? → Good Model → Give high influence in final prediction → **Large weights**

Higher Residual ? → Give Low influence in final prediction → **assign small weights**

STEP 2: M2

$h_0(x) + \gamma_1 h_1(x) + \varepsilon_1^{(x_i)}$

$\perp$

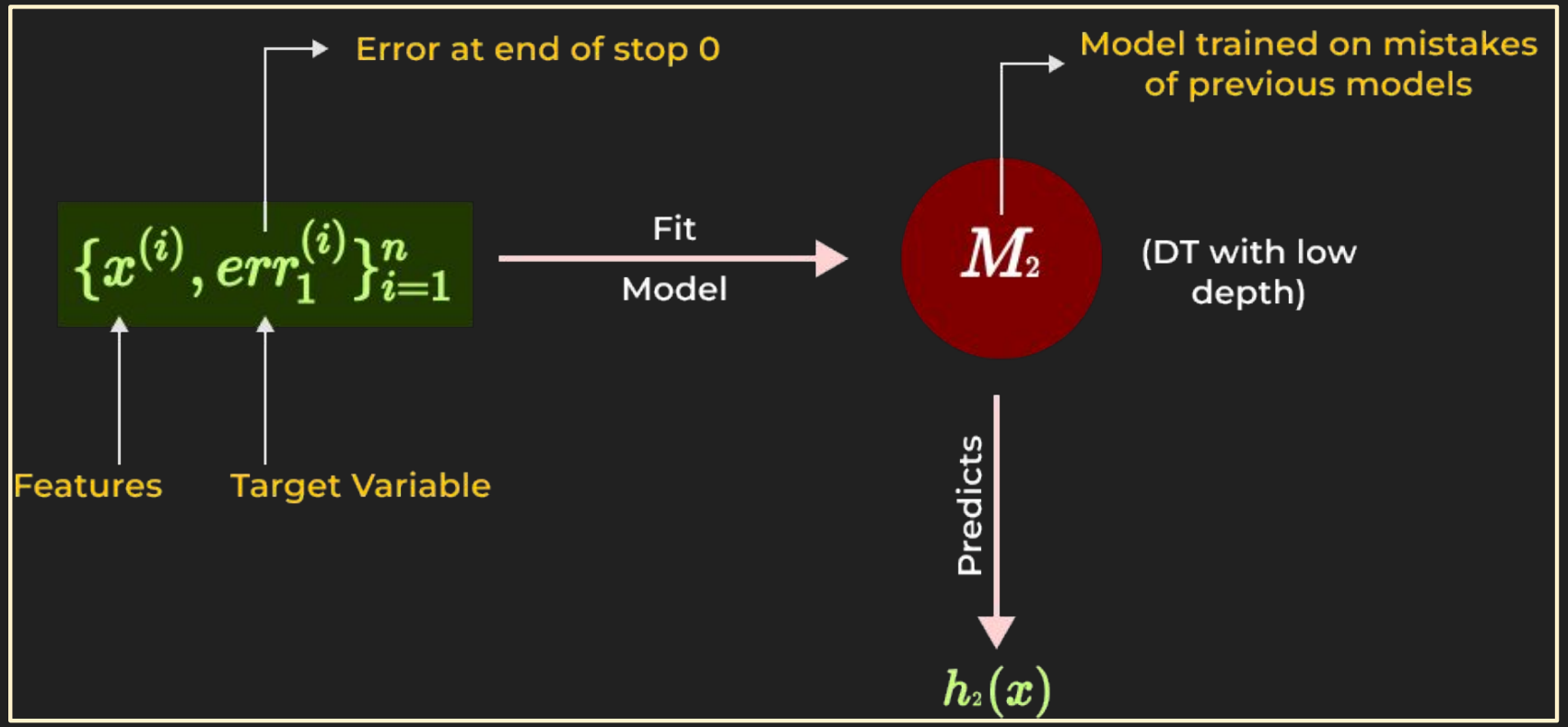
$h_0(x) + e_0^{(x_i)}$

→ We calculate the error after Stage 1 i.e err_1^i

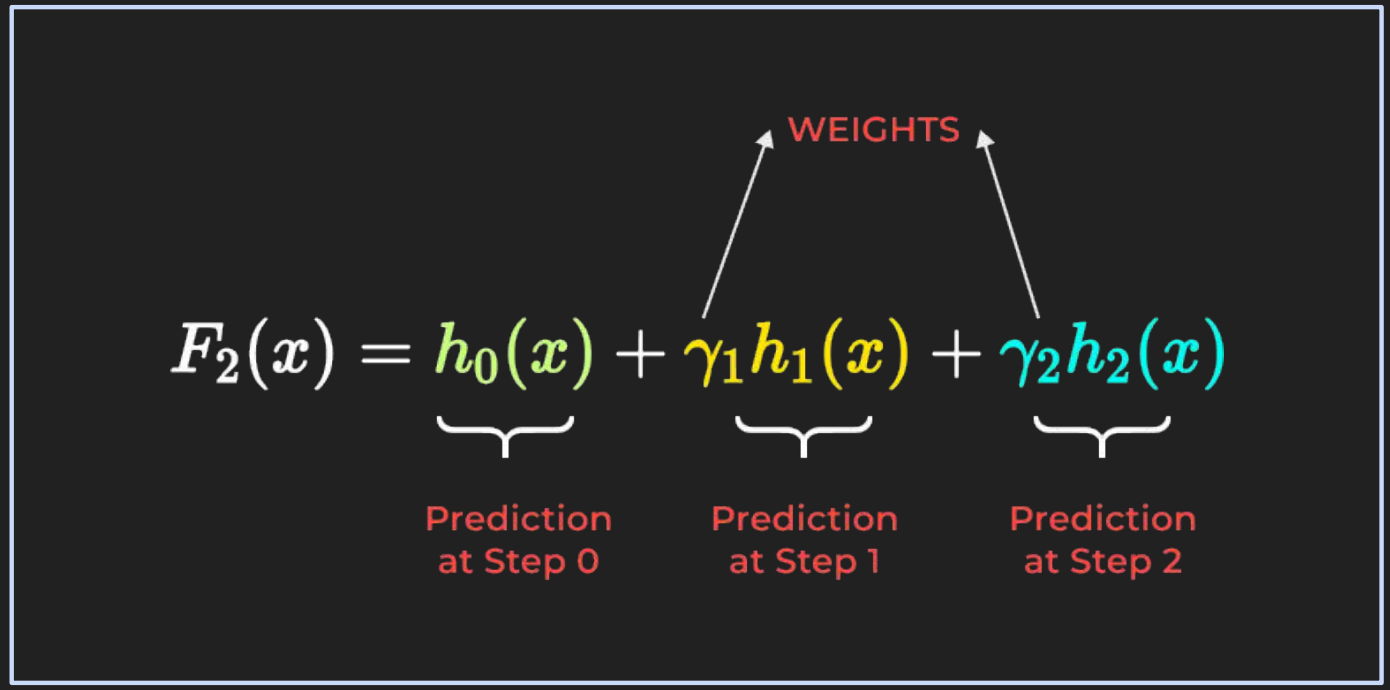
$\{x^{(i)}, err_1^{(i)}\}$

→ $y^{(i)} - \underbrace{F_1(x)}_{h_0(x) + \gamma_1 h_1(x)}$

Using $\{x^i, err_1^i\}$ we again fit a model



The final prediction at the end of Stage 2 :



We keep doing this till **M stages**

$$F_m(x) = h_0(x) + \underbrace{\gamma_1 h_1(x) + \gamma_2 h_2(x) + \ldots \gamma_m h_m(x)}$$

We can also write it as :

$$F_m(x) = h_0(x) + \sum_{i=1}^{m} \gamma_i h_i(x)$$

Note:

M is a hyper parameter

POINTS TO REMEMBER

1. **Base Learners** → **High Bias & Low Variance**

2. Combine learners using **additive combining.**

3. **Step 0 : M_0** → **Mean Model**

4. Iteratively build models on the error of previous model **(Sequential process)**

5. Use weights to adjust the influence of model in final prediction.

Summarizing Process using an Example

Let's recap using the regression example i.e predict weight using height & gender.

Height	Gender	Weight(y)
1.6	M	82
1.5	F	55
1.4	F	61
1.4	M	65

STEP 0: Mean Model

$$h_0(x) = \bar{y}$$

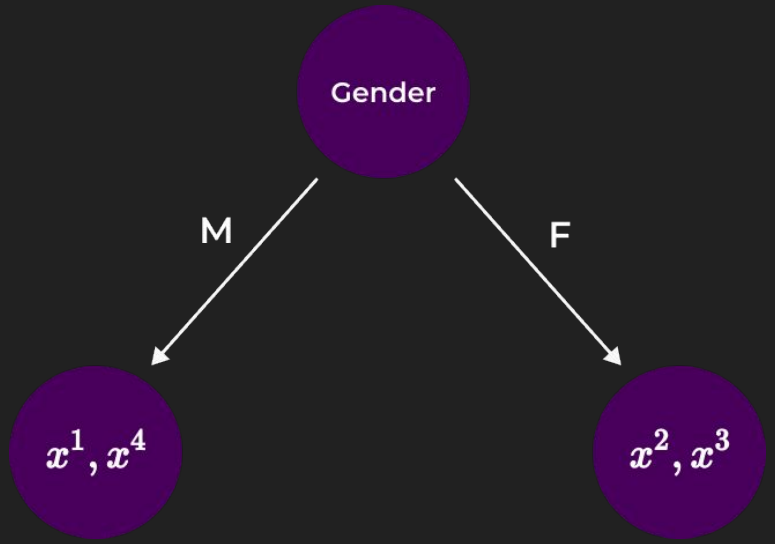
$$= \frac{82 + 55 + 61 + 65}{4}$$

$$h_0(x) = 65.75$$

Using the prediction of Stage 0 we calculate residual / error.

Height	Gender	Weight(y)	err_0
1.6	M	82	16.25
1.5	F	55	-10.75
1.4	F	61	-4.75
1.4	M	65	-0.75

Stage 1 : Using the residual (err_0^i) & features build a low depth DT



	Height	Gender	err_0	h1(x)
x1	1.6	M	16.25	7.75
x2	1.5	F	-10.75	-7.75
x3	1.4	F	-4.75	-7.75
x4	1.4	M	-0.75	7.75

$$\hat{y}_{left} = \frac{16.25 - 0.75}{2}$$

= 7.75

$$\hat{y}_{right} = \frac{-10.75 - 4.75}{2}$$

= - 7.75

Finally, combine the prediction of Stage 1 with Stage 0

Mo Model

$\gamma_1 = 1$ (Assume)

	Height	Gender	err	h1(x)	h0(x)	$f_1(x) = h_0(x) + \gamma_1 h_1$
x1	1.6	M	16.25	7.75	65.75	73.50
x2	1.5	F	-10.75	-7.75	65.75	58
x3	1.4	F	-4.75	-7.75	65.75	58
x4	1.4	M	-0.75	7.75	65.75	73.5

And using this final pred

Calculate the residual at end of Stage 1

Final pred at stage 1

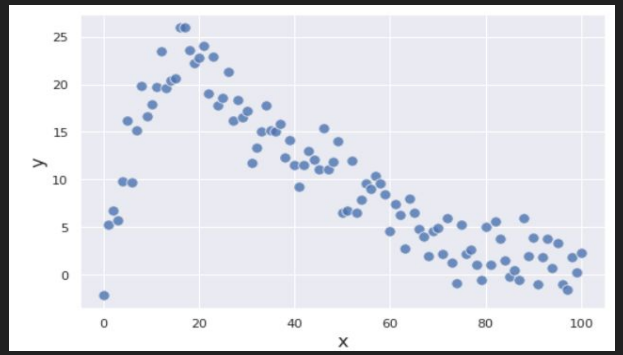
	Height	Gender	Weight(y)	F1(x)	err_1
x1	1.6	M	82	73.1	8.5
x2	1.5	F	55	58	-3
x3	1.4	F	61	58	3
x4	1.4	M	73.5	73.5	-8.5

Model error after stage 1

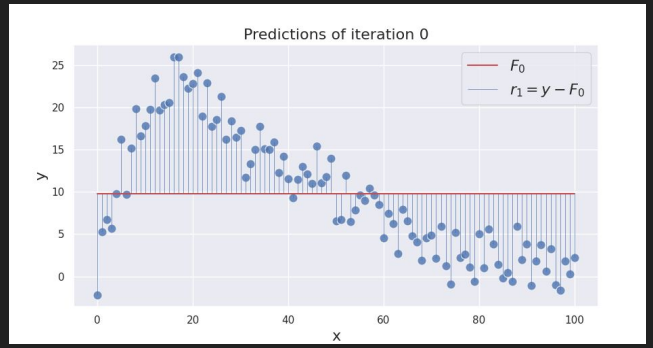
- Using the err_1^i & x^i ,fit a new model 1 (i.e stage 2)
- This goes on till M steps.

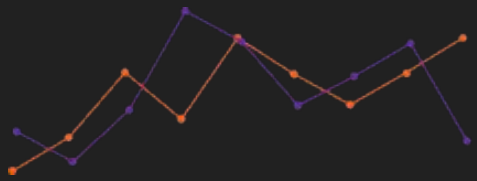
How does the model fitting looks geometrically ?

Imagine, we have 2D regression data like :



The regression line at each stage will look like :





Notice:

- Starts from straight line
- Adds complexity as stages increases

What happens at Train & Test time ?

At train time

- Fits all base learners (DT)
- Find the value of weights (γ_m)

NOTE:

- Training is bit slow.
- As it is a sequential process.

At test time

- We have already found hyper parameter M at train time .

Say , M = 3

For a query point

- Just pass it through DTs (base learners) & get predictions
- Multiply predictions with weight values (γ_m)

POINTS TO REMEMBER

- **Base Learners** → **High Bias & Low Variance**
- Combine learners using **additive combining.**
- **Step 0 : M_0** → **Mean Model**
- Iteratively build models on the error of previous model **(Sequential process)**
- Use weights to adjust the influence of model in final prediction.

Break : 8:10am

- **At train time :** Fit DTs & find weights
- **At test time :** Simply pass query point through base learner & multiply prediction with weight value to get final prediction

WHY BOOSTING ? (GBDT Intuition)

Since it is a sequential process → **Can't parallelize it.**

Hence, boosting is slow.

However, Random Forest can be easily parallelize.

Important: Real Magic of boosting is →

We can **MINIMIZE** any loss using gradient boosting.

What's Gradient Boosting ?

Say , we want to minimise Squared Loss

$$Squared\ Loss : L(y^{(i)}, \hat{y}^{(i)}) = (y^{(i)}, \hat{y}^{(i)})^2$$

$$Here, \hat{y} = F_k(x)$$

Prediction at end of Stage K

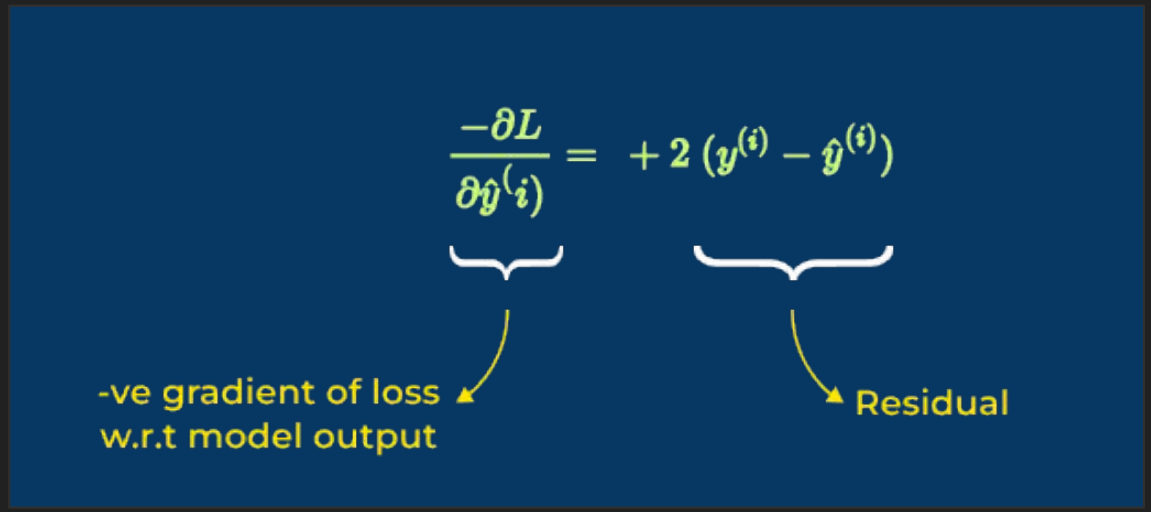
Let's differentiate it w.r.t $\hat{y}$

$$L(y^{(i)} - \hat{y}^{(i)}) = (y^{(i)} - \hat{y}^{(i)})^2$$

$$\frac{\partial L}{\partial \hat{y}^{(i)}} = \frac{(y^i - \hat{y}^{(i)})^2}{\partial \hat{y}^{(i)}}$$

$$\frac{\partial L}{\partial \hat{y}^{(i)}} = -2(y^{(i)} - \hat{y}^{(i)})$$

Let's multiply both sides with - 1



If we were to ignore 2 on R.H.S

- Residual is proportional to -ve gradient of loss function w.r.t model prediction

We know,

$$\hat{y} = F_k(x^i)$$

Replacing $\hat{y}$ with $F_k(x^i)$ in last equation

$$\underbrace{\frac{-\partial L}{\partial F_k(x^i)}}_{\text{-ve gradient of loss w.r.t model output}} = \underbrace{+2\,(y^{(i)} - \hat{y}^{(i)})}_{\text{Residual}}$$

Note:

If it were classification problems, simply replace square loss with log loss.

These -ve gradients are called Pseudo Residual.

How do we use Pseudo Residual ?

Say we are building a model & currently at step j i.e M_j

$$We\ need\ \{x^{(i)}, err^{(i)}\}\ to\ train\ m_j$$

$$M_j \leftarrow \{x^{(i)}, err^{(i)}\}$$

$$h_j(x)$$

$$err^{(i)} = y^{(i)} - F_{j-1}(x^i)$$

$$err^{(i)} = residual$$

Instead of calculating residual, simply calculate pseudo residual

Currently , we are using

$$err^{(i)} = y^{(i)} - F_{j-1}(x^i)$$

$$err^{(i)} = residual$$

Instead use,

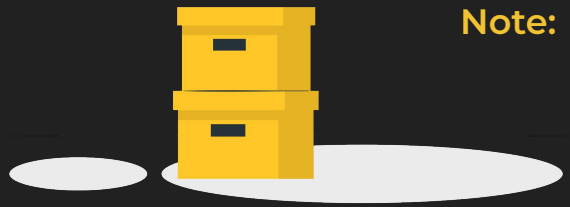
$$err^{(i)} \approx pseudo\ residual$$

$$= \frac{-\partial L}{\partial F_{j-1}}(x^i)$$

How does pseudo residual help us ?

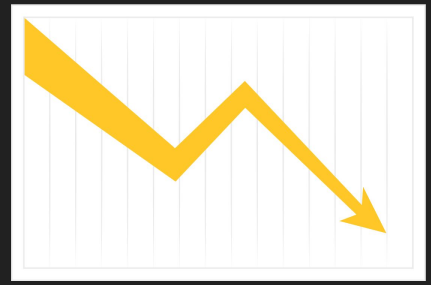
We are building model at each step → To minimize the residual

If we were to use pseudo residuals instead of residuals, → Indirectly minimizing the loss functions



Note:

- Can minimize any custom loss function
- As long as it is differentiable

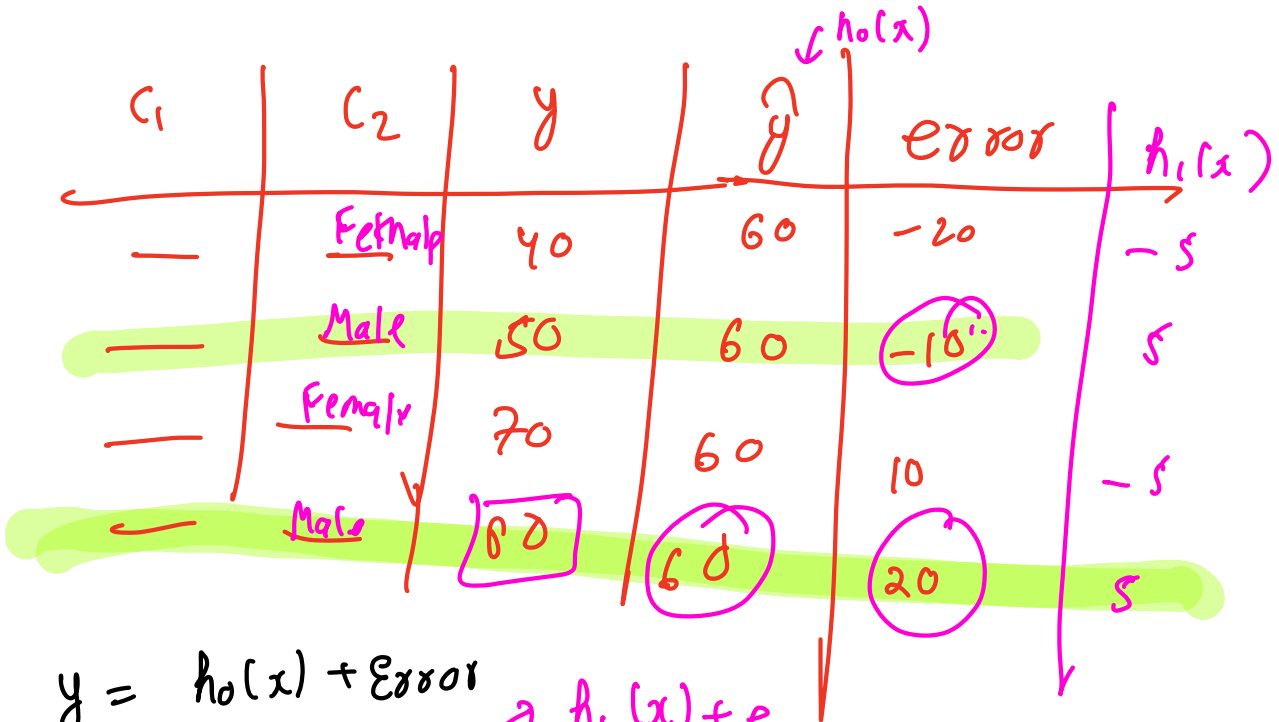


POINTS TO REMEMBER

- **Base Learners** → **High Bias & Low Variance**
- Combine learners using **additive combining.**
- **Step 0 : M_0** → **Mean Model**
- Iteratively build models on the error of previous model **(Sequential process)**
- Use weights to adjust the influence of model in final prediction.

- **At train time :** Fit DTs & find weights
- **At test time :** Simply pass query point through base learner &

  multiply prediction with weight value to get final prediction
- **Why Boosting ? -** Can minimise any loss function
- **How does it minimize any loss function ?** - Using pseudo residuals

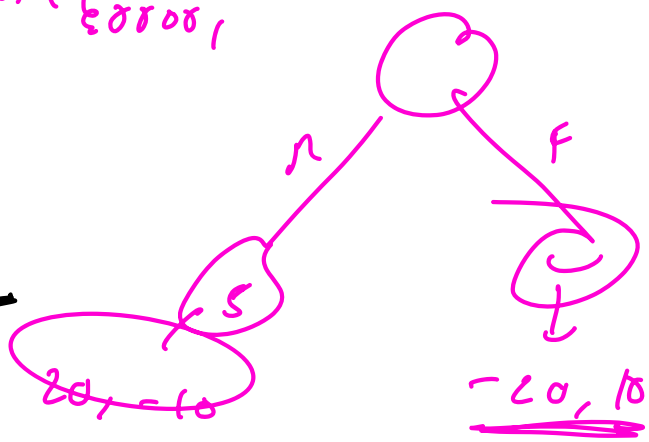


$y = h_0(x) + \text{Error}$

$80 = 60 + 20$ ← $h_1(x) + \text{Error}_1$, $h_1(x) = 5$

60 = $h_0(x)$, 20 = Error

$h_0(x) \rightarrow 60 + h_1(x) \rightarrow 5 + 15$ ← Error



$$h_0(x) + h_1(x) + \cdots + h_m(x) + \text{Error}$$