Lets modify the sample weights so that next stump will take into account the error made by first stump.

To Modify Weight

New Sample weight of Incorrectly classified sample = Sample weight $* e^{\text{amount of say}}$

In our case for Incorrectly classified Sample

New Sample weight $= \frac{1}{8} * e^{0.97} = 0.33$

Note

if Amt of say is less than the stump is not doing good job.

For correctly classified Sample

New Sample weight = Sample weight $* e^{-\text{amount of say}}$

$\therefore$ updated Sample weight $= \frac{1}{8} * e^{-0.97} \Rightarrow 0.05$

Initial sample weight → Stump1 → New Sample weight → Stump 2

Now

Chest Pain	Blocked Arteries	Patient weight	Heart Disease	Sample weight	New Sample weight	Normalized Sample weight
Yes	Yes ✓	205	Yes	1/8	0.05	0.07
No.	Yes ✓	180	Yes	1/8	0.05	0.07
Yes	No	210	Yes	1/8	0.05	0.07
Yes	Yes ✓	167	Yes	1/8	0.33	0.49
No	Yes	156	No	1/8	0.05	0.07
No.	Yes	125	No	1/8	0.05	0.07
Yes	No	168	No	1/8	0.05	0.07
Yes	Yes	172	No	1/8	0.05	0.07

Stump 2 will focus on this More.

Divide Group

Yes	No	168	No	1/8 ↓	0.05	
Yes	Yes	172				

Normalized — Not Normalized

Total Sum = 0.68

Divide Every Entry in New Sample weight by 0.68

We have new Normalized Sample weight ⇒

The incorrectly classified Sample has Higher New Sample weight

→ which means Stump2 will focus more on it.

→ The correctly classified sample has comparatively lower Sample weight so stump2 will have less focus on it

* Similarly we can have many stumps.

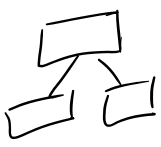
. This is How the Adaboost creates & uses Stumps.

Now lets chk how forest of stumps made by Adaboost does classification.

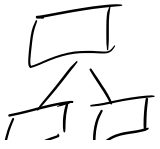
(1) we will pass the test sample through each stump.

(2) Let there be few stumps that classifies the sample as Heart Disease & let there be stumps that classifies the sample as Not Heart Disease

(3) Has Heart Disease



⇒ Amt of Say 0.97

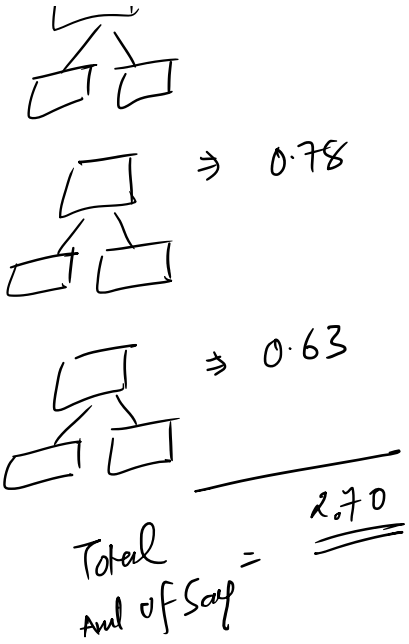


⇒ 0.32

Has Not Heart Disease

⇒ Amt of Say 0.41

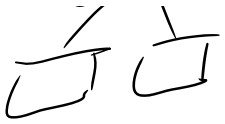
⇒ 0.82.



⇒ 0.78

⇒ 0.63

Total Amt of Say = 2.70



Total Amt of Say = 1.23

Total Amt of Say For Stumps that Classifies as Heart Disease is greater

∴ The test Sample is classified as Heart Disease

Summary

(1) Assign Sample Weights (Initial)

(2) Create Stumps for each Property.

(3) Use Gini Index to Identify first Stump.

(4) For the first Stump:
Calculate Total Error
Calculate Amt of Say.

(5) Update the Sample weights for each sample

For Incorrectly Classified = Updated weight = Sample weight * $e^{\text{amt of say}}$

For Correctly Classified = Updated weight = Sample weight * $e^{-\text{amt of say}}$.

(6) Now Specify Updated Weight for Each Sample & Normalize it

to generate New Sample Weight

(7) Identify New Stump based on New Sample Weight & Repeat step 3 to 7

(8) For Classification

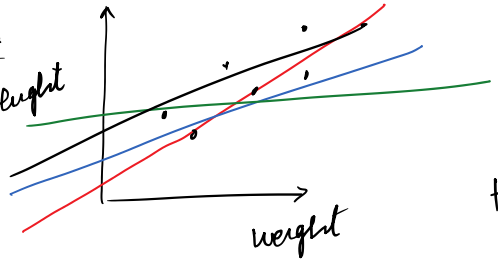
8.1 Run the Test Sample through All the Stumps

8.2 Calculate Total Amt of Soup for Samples classifying Yes & NO

8.3 Classify the test sample based on largest sum of Amt of Soup.

Gradient Descent →

Consider



predicted height = intercept + slope * weight

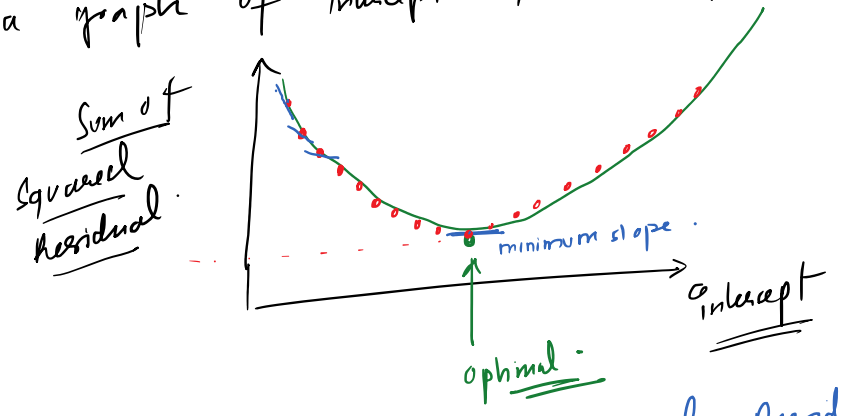
Here we can many line with diff intercept & slope values.

We want line with intercept & slope such that the cost Fn should give Minimum value.

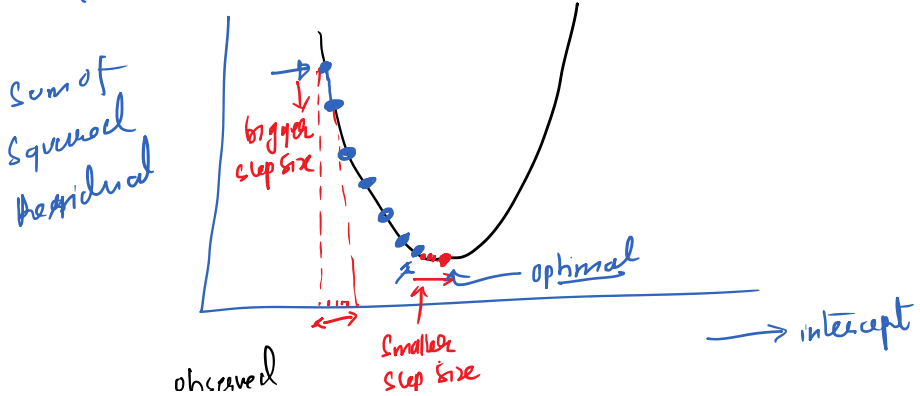
Let the cost Fn be Sum of Squared Error (Residual)

Let us Consider only intercept

Lets plot a graph of Intercept V/s Sum of Squared Residual.



To find Minimum value for Sum of Squared Residual we will start with small value of intercept and will increase the value by very small Amount.



observed

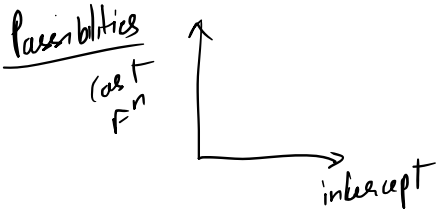
Consider

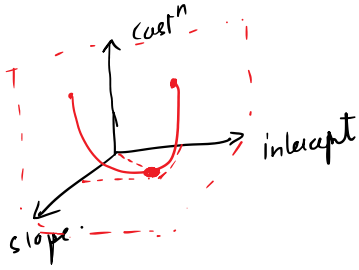
x	y
0.5	1.4
2.3	1.9
2.9	3.2

Let the cost F^n = Sum of Squared Residual

$$\text{Sum of Squared Residual} = \underbrace{(1.4}_{\text{observed}} - \underbrace{(\text{intercept} + \text{slope} * 0.5)}_{\text{predicted}})^2 + (1.9 - (\text{intercept} + \text{slope} * 2.3))^2 + (3.2 - (\text{intercept} + \text{slope} * 2.9))^2$$

Possibilities





Here we want to find value of intercept & slope that gives Minimum Sum of Squared Error.

Step 1

$$\frac{d}{d\,\text{intercept}}(\text{Sum of Squared Residual}) = \& \quad \frac{d}{d\,\text{slope}}(\text{Sum of Squared Residual})$$

$$\text{for } \frac{d}{d\,\text{intercept}}(1.4 - (\text{intercept} + \text{slope} * 0.5))^2 \rightarrow \text{Using chain rule.}$$

$$\Rightarrow 2 * (1.4 - (\text{intercept} + \text{slope} * 0.5)) * \frac{d}{d\,\text{intercept}}(1.4 - \text{intercept} - \text{slope} * 0.5)$$

(1.4: 0; intercept: 1; $\text{slope} * 0.5$: 0; derivative of -1)

$$= -2(1.4 - (\text{intercept} + \text{slope} * 0.5))$$

$$\frac{d}{d\text{intercept}}(\text{Sum of Squared Residual}) = (-2)(1.4 - (\text{intercept} + \text{slope} \times 0.5)) + (-2)(1.9 - (\text{intercept} + \text{slope} \times 2.3)) + (-2)(3.2 - (\text{intercept} + \text{slope} \times 2.9))$$ } Eq 1

$$\frac{d}{d\text{slope}}(\text{Sum of Squared Residual}) = (-2)(0.5)(1.4 - (\text{intercept} + \text{slope} \times 0.5)) + (-2)(2.3)(1.9 - (\text{intercept} + \text{slope} \times 2.3)) + (-2)(2.9)(3.2 - (\text{intercept} + \text{slope} \times 2.9))$$ } Eq 2

We get the above 2 Eqn lets start with Random values of

intercept = 0 & slope = 1

Substitute the values ⇒ intercept = 0 slope = 1

$\frac{d}{d\text{intercept}} = -1.6$ } slope of line wrt intercept

$\frac{d}{d\text{slope}} = -0.8$ } slope of line wrt slope

For step size

New step size = Slope of line at point × Learning Rate

Note Learning Rate ⇒ takes small value 0.1 → 0.3

Let us take Learning Rate = 0.01

$\therefore$ $\text{step size}_{\text{intercept}} = \frac{d}{d\text{intercept}} * 0.01 = -1.6 * 0.01 = -0.016$

$\text{step size}_{\text{slope}} = \frac{d}{d\text{slope}} * 0.01 = -0.8 * 0.01 \Rightarrow -0.008$

$$\text{step size}_{\text{slope}} = \frac{\alpha}{d\,\text{slope}} * 0.01$$

$$\text{New Intercept} = \text{Old Intercept} - \text{step size}_{\text{intercept}} = 0 - (-0.016) = 0.016$$

$$\text{New slope} = \text{old slope} - \text{step size}_{\text{slope}} = 1 - (-0.008) = 1.008$$

[Substitute this New Intercept & New slope in eqn $\frac{d}{d\,\text{intercept}}$, $\frac{d}{d\,\text{slope}}$
& Calculate New step size for intercept & slope
And again calculate New Intercept & New slope]

We will repeat until step size becomes very small = 0.001
or some Maximum Number of steps is reached Ex: 1000

For above Dataset Best fitting line will have intercept = 0.95
& slope = 0.64

This is How the Gradient Descent Optimizes Parameter.

Summary

1. Get eqn of line
2. Want Best intercept & slope
3. Identify loss fn (Sum of Squared Residual)
4. Find eqn for $\frac{d}{d\,\text{intercept}}$ & $\frac{d}{d\,\text{slope}}$
5. Start with initial value of slope & intercept
6. Plug & $\frac{d}{}$

(5) Start with ....

(6) find $\frac{d}{d\,intercept}$ & $\frac{d}{d\,slope}$ value

(7) Calculate $stepSize_{intercept} = \frac{d}{d\,intercept} * Learning\ Rate$

$stepSize_{slope} = \frac{d}{d\,slope} * Learning\ Rate$

(8) Updated Intercept = old Intercept - $stepSize_{intercept}$

Updated slope = old slope - $stepSize_{slope}$

(9) Repeat step 6 ⟶ step 8 untill stepSize is very small or No of iterations ≥ 1000

Gradient Boost for Regression →

* Gradient Boost starts with **Single leaf** instead of Tree/stump.
* Leaf represents initial guess for the weights of the samples.

Ex Consider

observation

Height	favorite color	Gender	Weight
1.6	Blue	male	88
1.6	Green	female	76
1.5	Blue	Female	56
1.8	Red	male	73
1.5	Green	male	77
1.4	Blue	female	57

Initial Prediction

Step 1) First lets make Initial Guess = Weight (Average) = **71.2**

Step 2 Like Adaboost, Even Gradient Boost builds tree based on previous Tree

But Unlike Adaboost, the Gradient Boost tree are larger than stump

However the Height is still Restricted [No of leaf 8 → 32] in Each Tree

[Gradient Boost will Build Another Tree based on Error of previous Tree]

* Gradient Boost Continous to Build Tree in this fashion Untill it has made numbers of Trees you asked for

or the Additional Tree fails to improve the fit

or the Additional Tree tends to improve ...

Here Initial Guess = Weight (Average) = 71.2

Predicted Value = 71.2 + ____ + ____

71.2 ← Leaf

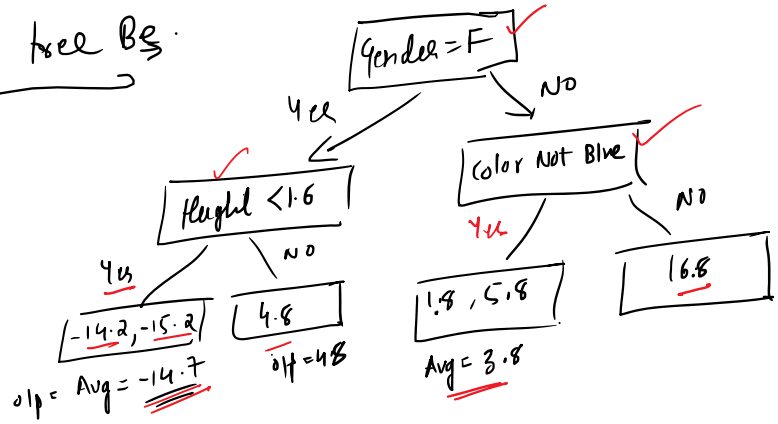
Boosting →

Now we Need to find Residual for All the Samples.

Height	favorite color	Gender	Weight (observed)	Residual (Observed - Average 71.2)
1.6	Blue	male	88	16.8
1.6	Green	female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	male	73	1.8
1.5	Green	male	77	5.8
1.4	Blue	Female	57	-14.2

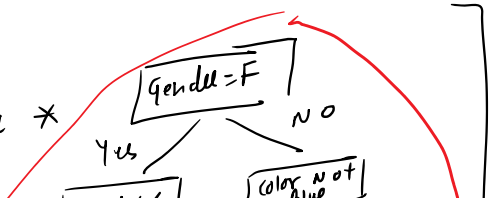
Lets Build a Tree using the Residual & Not weight

Let the tree Be:



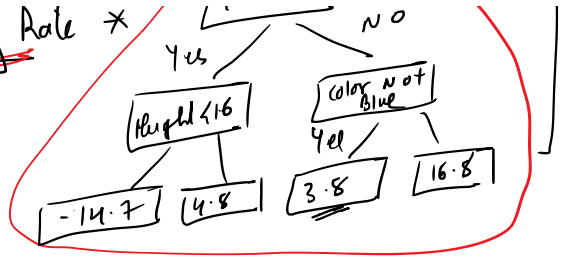
Predi... If there will be only 1 Tree Assume.

Then

Predicted = Inital Guess + Learning Rate * 

Then

Predicted weight = Inihal Guess + Learning Rate * 0.1



Ex for first sample

1.6	Blue	male	2

weight

71.21

Predicted weight = 71.2 + 0.1 * 3.8 = 71.58

Apply this on All the Samples to get New Prediction.

Height	favorite color	Gender	Weight	Residual (Inihal Guess = 71.2)	New Prediction	New Residual
1.6	Blue	male	88	16.8	71.58	R_1
1.6	Green	female	76	4.8	P_1	R_2
1.5	Blue	Female	56	-15.2	P_2	R_3
1.8	Red	male	73	1.8	P_3	R_4
1.5	Green	male	77	5.8	P_4	R_5
1.4	Blue	Female	57	-14.2	P_5	R_6

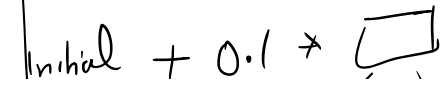
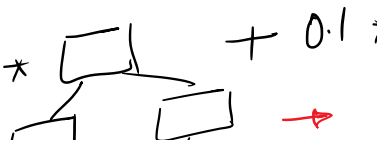
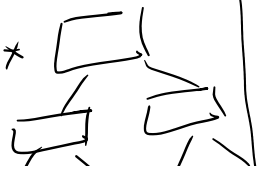
Residual 1 — Residual 2

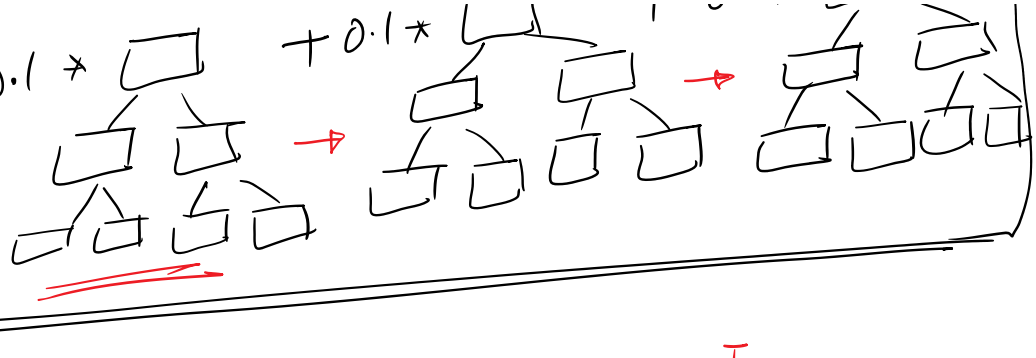
* Now Again find New Residual for New Prediction for all the Samples.

* Using this New Residual Construct New Tree and so on.

Untill the New Tree does not make Significant Contribution to new Prediction.

Let say we have 3 Tree for above DataSet

Final = Inihal + 0.1 *  + 0.1 *  + 0.1 * 

∴ final Prediction = Initial guess + 0.1 *  + 0.1 *

For our Ex

= 71.2 + 0.1 * T_1 + 0.1 * T_2 + 0.1 * T_3

Boosting

Gradient Boost Principle →

[Taking lot of small steps in right direction will result into better predictions]

* Gradient Boost for Classification →

* Consider

Observation (target)

Likes Popcorn	Age	Favorate Color	Loves Troll 2 (movie)
Yes	12	Blue	Yes 1
Yes	87	Green	Yes 1
No	44	Blue	No 0
Yes	19	Red	No 0
No	32	Green	Yes 1
No	14	Blue	Yes 1

(Classification.

Step 1 → Start with leaf that represent Initial prediction.

* When we use Gradient Boost for classification the initial prediction for every individual sample = log(odd) [In regression, its average of the observation]

log(odd) → That someone loves Troll 2 movie

Here 4 ppl loves Troll 2 & 2 do not love Troll 2

$$\log_e(\text{odd}) = \log\left(\frac{\text{No of Yes}}{\text{No of No}}\right) = \log_e\left(\frac{4}{2}\right) = \ln(2) = \boxed{0.7}$$

To use log(odd) we have to convert it into Probability.

We will use logit function

$$\therefore \text{Probability of loving Troll 2} = \frac{e^{\log(\text{odd})}}{1 + e^{\log(\text{odd})}} = \frac{e^{0.7}}{1 + e^{0.7}} = \frac{1}{1 + e^{-0.7}} = \boxed{0.7}$$

Probability of loving Troll 2 $= \frac{1}{1+e^{-\log(odd)}} = \frac{1}{1+e^{-0.7}} = \frac{1}{1+e^{-}} =$

[Since probability of loving Troll 2 is greater than 0.5, so we can classify everyone in Training Dataset as someone who loves Troll 2]

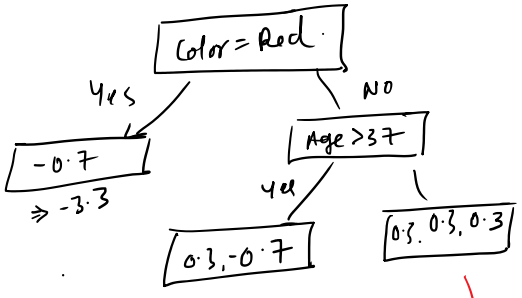
Initial = 0.7

Likes Popcorn	Age	Favorate color	loves Troll 2 (movie)	Residual
Yes	12	Blue	Yes 1	0.3
Yes	87	Green	Yes 1	0.3
No	44	Blue	No 0	-0.7
Yes	19	Red	No 0	-0.7
No	32	Green	Yes 1	0.3
No	14	Blue	Yes 1	0.3

Initial Residual

Now Build Tree Using Color and Age.

Let the Tree be →



$o/p = \frac{-0.7}{0.7*(1-0.7)}$

$= -3.3$

$\Rightarrow -3.3$

$o/p = \frac{0.3+(-0.7)}{(0.7*(1-0.7))+(0.7*(1-0.7))}$

$= -1$

$o/p = \frac{0.3+0.3+0.3}{(0.7*0.3)+(0.7*0.3)+(0.7*0.3)}$

$= 1.4$

Note →

When gradient boost is used for Regression the leaf with Single Residual has o/p equal to that Residual & for leaf with more than one residual the o/p is Average of residual

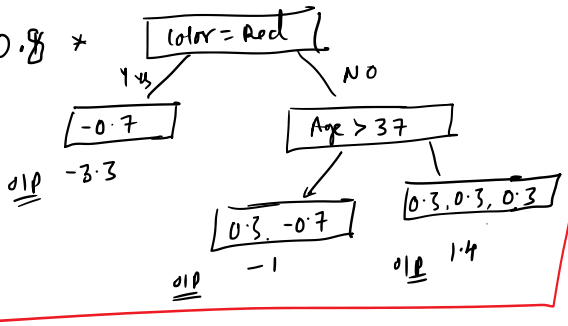
For Classification

o/p value at each leaf

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

For each residual.

New Prediction = $0.7 + 0.8 *$



Consider first sample → Likes Popcorn: Yes | Age: 12 | Favorite Color: Blue

New Prediction = $0.7 + 0.8 \times 1.4 = 1.8$

Convert into Probability = $\frac{1}{1+e^{-1.8}} = 0.9$

Now Find New Prediction for each sample using above model.

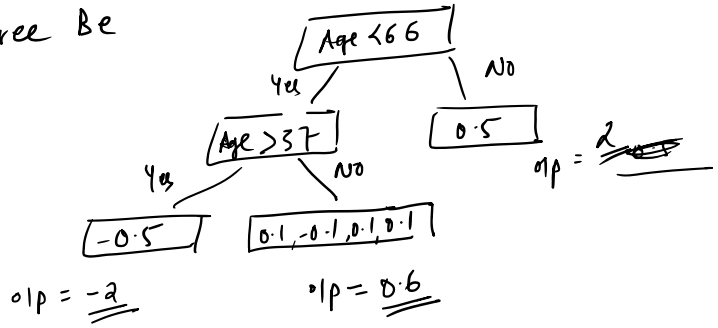
Observation ✓ | Initial Prediction = 0.7 ✓ | Let

Likes Popcorn	Age	Favorite Color	Loves Troll 2 (Target)	Residual	New Prediction	New Residual
Yes ✓	12	Blue	Yes 1 ✓	0.3	0.9 ✓	0.1
Yes	87	Green	Yes 1	0.3	0.5	0.5
NO	44	Blue	NO 0	-0.7	0.5	-0.5
Yes	19	Red	NO 0	-0.7	0.1	-0.1
NO	32	Green	Yes 1	0.3	0.9	0.1
NO	14	Blue	Yes 1	0.3	0.9	0.1

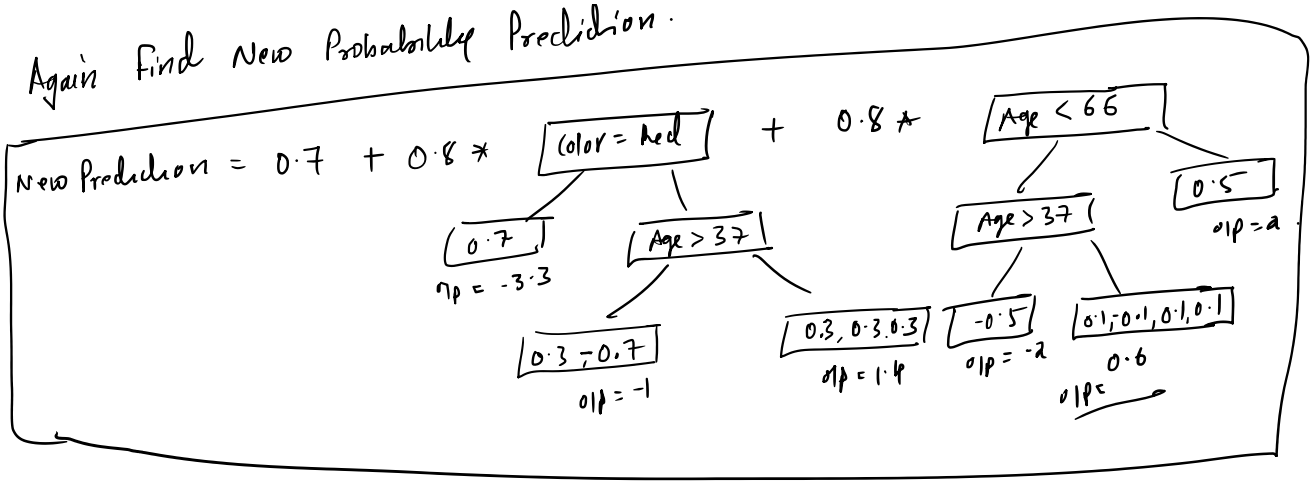
Here New Prediction is not 0.7 for all sample, here each sample has its own new Prediction & we calculated New Residual for each sample.

Again Build a Tree using let say Age only

Let the Tree Be



Again find New Probability Prediction.



Again apply each Sample dataset on the above model and calculate New prediction.

→ Again Calculate new residual and so on.

This process repeate until we have made the maximum specified no of Trees or the residual gets very very small

Summary Gradient Boost for Classification

1. Consider the dataset
2. Find log (odd).
3. find Initial Probability using $\log(\text{odd}) = \frac{1}{1+e^{-\log(\text{odd})}}$
4. Calculate Residual based on Initial Probability
5. Construct Tree using Initial Residual.
6. new Prediction = Initial Prediction + Learning Rate * Tree1.
7. Calculate New Prediction for all Sample.
8. Again find New Residual.
9. Construct New Tree and so on.
10. Repeat until max no of Tree constructed or the Residual is Insignificant.

Summary Gradient Boost for Regression.

1. Consider the dataset
2. Find Initial Prediction = Avg of the Observed value.
3. Calculate Residual based on Initial Probability
4. Construct Tree using Initial Residual.
5. new Prediction = Initial Prediction + Learning Rate * Tree1.
6. Calculate New Prediction for all Sample.
7. Again find New Residual.
8. Construct New Tree and so on.
9. Repeat until max no of Tree constructed or the Residual is Insignificant.

XGBoost (Xtreme Gradient Boost)

→ Implementation of Gradient Boosted Decision Tree

Here Decision Tree is created in Sequential form.

→ Weights plays important Role in XGBoost

→ Weights are assigned to all independent variables which are then fed into Decision Tree that predict result.

→ Weights of variable predicted wrong by tree is increased and that variable are fed in Second decision tree.

→ These individual predictors/classifiers can ensemble to give Strong and more precise model.

→ XG Boost is faster than Gradient Boost

→ There is stopping criterion for tree splitting in XG Boost.

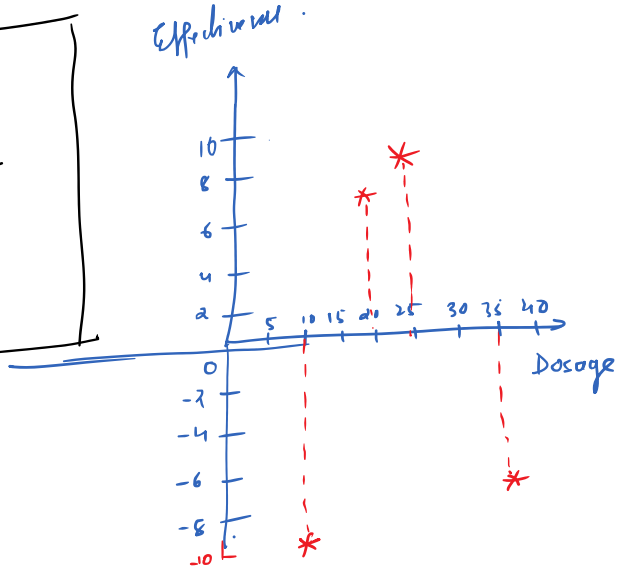
→ XG Boost uses max_depth parameter and it starts pruning the Tree backward.

→ This pruning improves computational performance and also helps to overcome problem of overfitting

XGBoost for Regression →

Consider

x (Dosage of Drug)	y (Drug effectiveness)
10	-10
20	7
25	8
35	-7



Build Prediction model using XGBoost.

Step1 → Assume initial threshold for effectiveness = 0.5 (Initial Prediction)

Step2: Calculate Residual.

observation x	y	Initial = 0.5 Residual
10	-10	-10.5
20	7	6.5
25	8	7.5
35	-7	-7.5

Initial Residual

Step3 Consider all the Residuals in Root, let $\lambda = 0$

[-10.5, 6.5, 7.5, -7.5]

$$\text{Similarity} = \frac{(-10.5+6.5)^2}{4+0} = 4$$

Now let us see if we can cluster the residual better on Similarity by split

Note

$$\text{Similarity Score} = \frac{(\text{Sum of Residual})^2}{\text{No of Residual} + \lambda}$$

λ = Regularization Parameter

Gain = Similarity (left) + Similarity (right) − Similarity (root)

To Determine Dosage value for Split ⇒

Case 1> Consider Avg of first two X

Average Dosage = 15

Dosage < 15 (Similarity = 4)

Yes — NO

Case 2>

Consider Avg of Next two X

Avg = $\frac{20+25}{2}$ = 22.5

Dosage < 22.5 (Similarity = 4)

Yes — NO

Case 3>

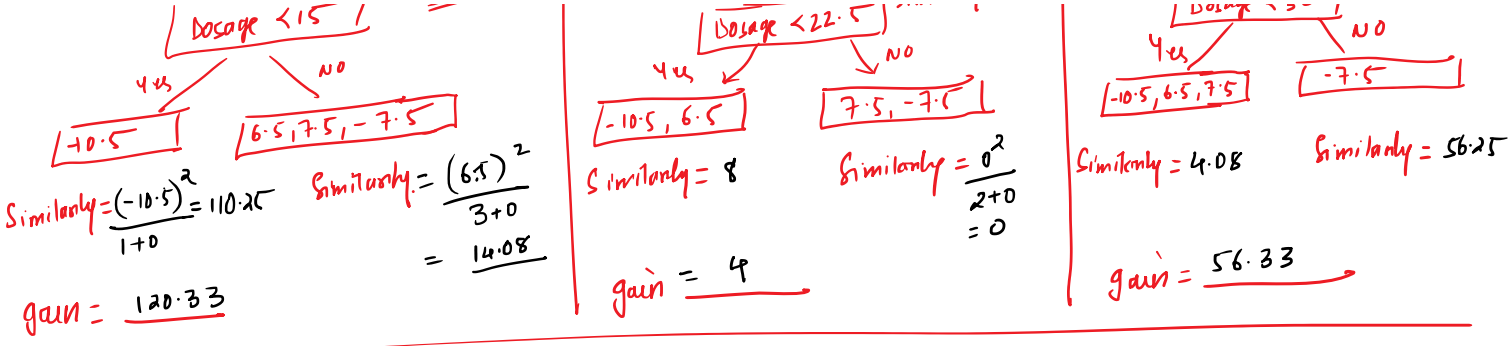
Consider Avg of last two X

Avg = $\frac{25+35}{2}$ = 30

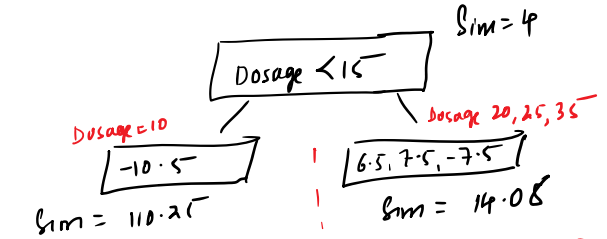
Dosage < 30 (Similarity = 4)

Yes — NO

[-10.5, 6.5, 7.5] — -7.5



Since Dosage < 15 gives Highest gain so it will be split factor.

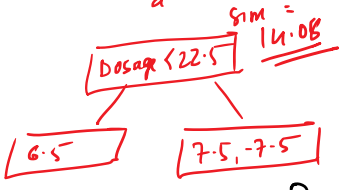


→ cannot be split further.

Can be split further.

(Case 1) Consider first 2.

Avg = (20 + 25)/2 = 22.5

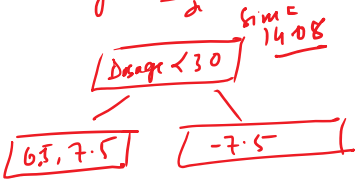


Sim = 42.25 Sim = 0

gain = 28.17

(Case 2) Consider last 2.

Avg = (25 + 35)/2 = 30

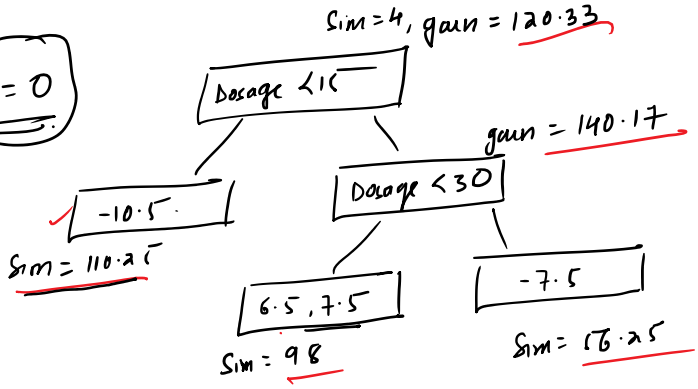


Sim = 98 Sim = 56.25

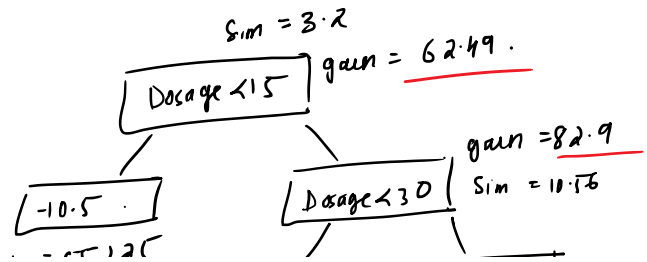
gain = 140.17

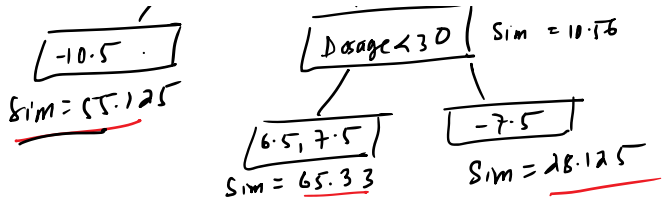
Highest gain is for Dosage Avg = 30. So Dosage = 30 will be split condition.

∴ Tree = $\lambda = 0$



Solve it for $\lambda = 1$ Tree





Note
When $\lambda > 0$ the Similarity Score is smaller so gain is smaller.

[Decrease in Similarity Score is Inversely proportional to Number of Residuals on the Leaf].

Pruning → Kyon chahiye.

1. Height Restriction for Tree
2. Ignoring Unnecessary Branches
3. Decrease Computation
4. less Time
5. Avoid Overfitting

Pruning of Tree in XGBoost is purely done on gain

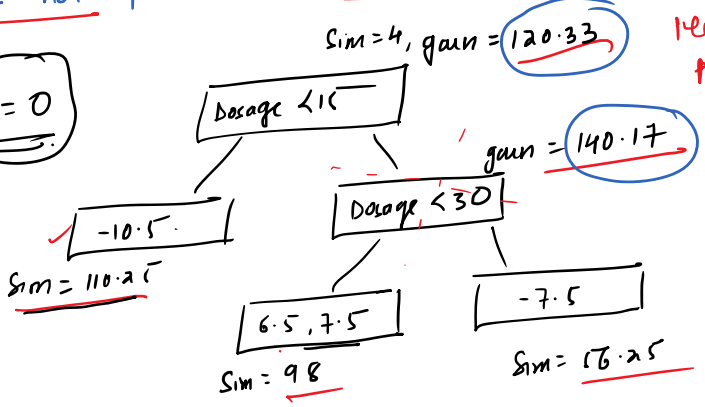
Let us Assume a Treshold gain = γ (Gamma).

Rule for Pruning If at Branch gain - γ < 0 Then Prune and go above
If gain - $\gamma \geq 0$ then do not Prune

* If child is not pruned then parent also will not be pruned

∴ Tree = $\lambda = 0$

Let $\gamma = 130$



Here 120.33 - 130 < 0 Case for pruning But Since child is not pruned so Given root is not.
Here 140.17 - 130 = 10.17 > 0
No pruning

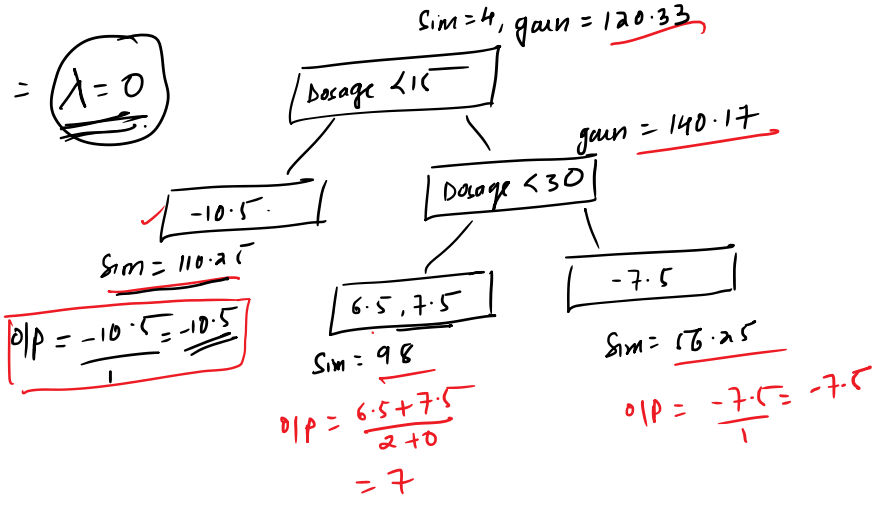
Here if γ is Sufficiently large then it may result into pruning of

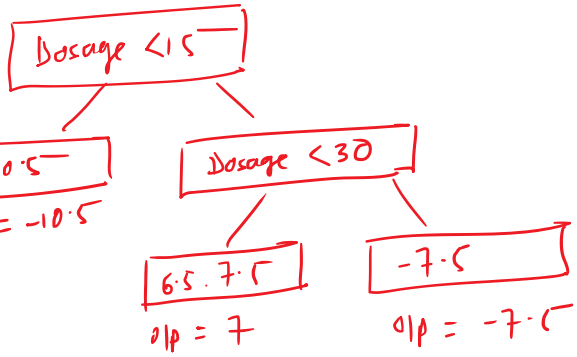
Complete Tree till root → known as Extreme Pruning.

→ Regardless of value of λ & γ let us assume the Tree as →

Find output value at each leaf = $\frac{\Sigma \text{ Residual}}{\text{No of Residual} + \lambda}$

∴ Tree = $\lambda = 0$



New Prediction = 0.5 + 0.3 * 

In XGBoost the Learning rate is known as Eta(ϵ) and default value = 0.3

New Prediction for Dosage = 10

$\Rightarrow 0.5 + 0.3 * (-10.5) \Rightarrow -2.65$

Original Prediction = 0.5 ✓

New Prediction = -2.65

* Find New Prediction for all the other 3 Dosages

* This will give New Residual.

* Using the New Residual Build a New Tree and make New Prediction that will give Smaller Residual.

* Keep Building the Tree until the max No of Tree constructed or residual is significantly small.

Let For above case there are 4 Trees T_1, T_2, T_3, T_4

Final Prediction = $0.5 + 0.3 * T_1 + 0.3 * T_2 + 0.3 * T_3 + 0.3 * T_4$.