



# Boosting

**Adaboost, Gradient Boost, XGBoost**

**Seunghan Lee**



# Contents

1

## **What is Boosting?**

1) Ensemble method / 2) Bagging vs Boosting

2

## **1. Adaboost**

## **2. Gradient Boost**

## **3. XGBoost**

3

## **Boosting with Python**



# 1. What is Boosting?

- 1) Ensemble method
- 2) Bagging vs Boosting

# 1. What is Boosting?

## (1) Ensemble

*In statistics and machine learning, **ensemble methods** use **multiple learning algorithms** to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. ( Wikipedia )*

- Use “**multiple models**” to predict the output  
( both regression & classification )
- Several Weak learner > One Strong learner
- ex) Bagging & Boosting

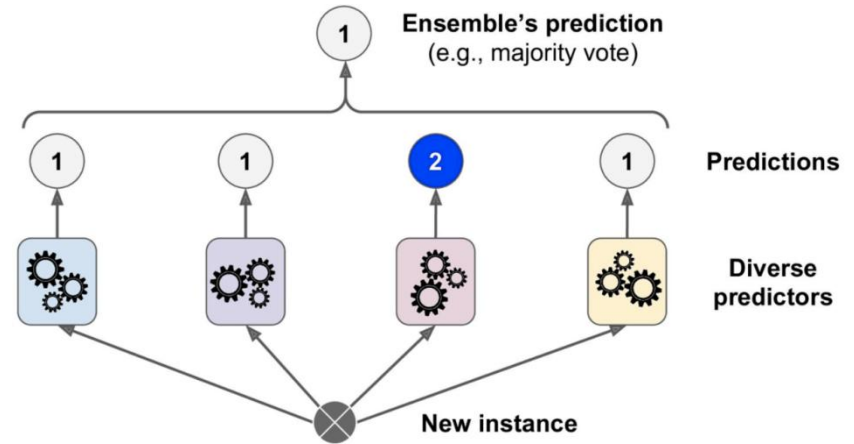


Figure 7-2. Hard voting classifier predictions

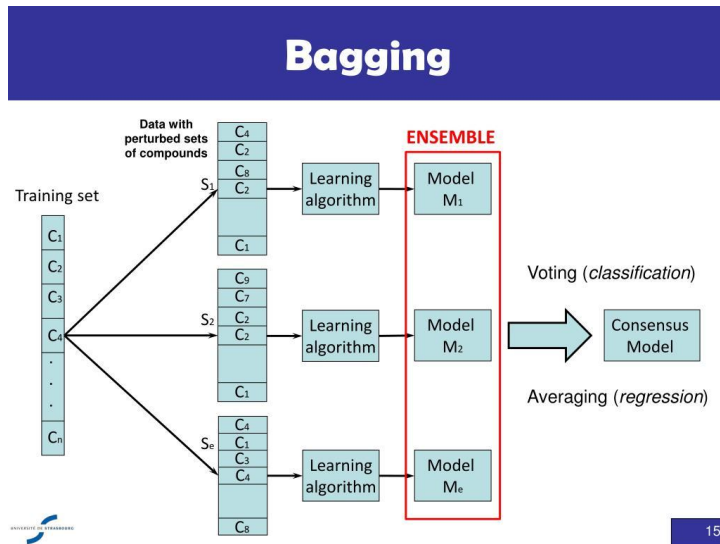
# 1. What is Boosting?

## (2) Bagging vs Boosting

### Steps of Bagging

- 1) Random sampling with replacement ( = Bootstrapping )
- 2) Select subset of features randomly
- 3) Grow the tree to the largest
- 4) Repeat 1)~3) N times ( total of N models )  
-> make prediction based on these N models

<https://www.slideserve.com/tarala/short-overview-of-weka>

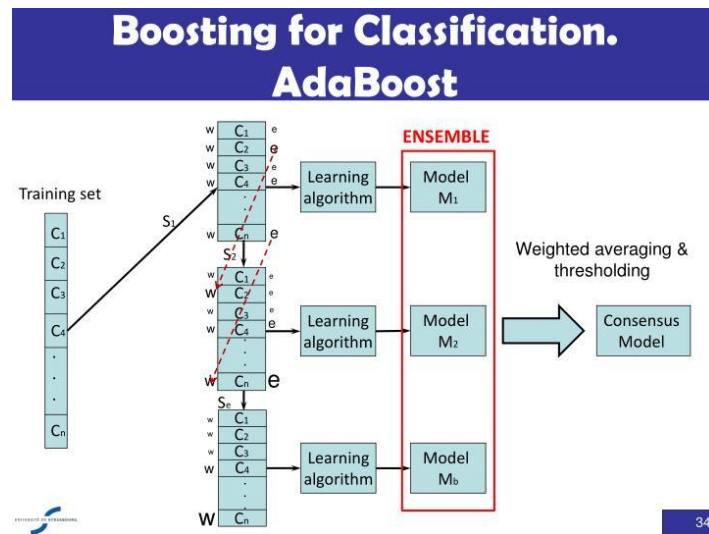


# 1. What is Boosting?

## (2) Bagging vs Boosting

### Steps of Boosting

- 1) First random sample ( $d_1$ ) without replacement  
-> train a weak learner  $C_1$
- 2) Second random sample ( $d_2$ ) without replacement, considering the misclassified sample in learner  $C_1$   
-> train a weak learner  $C_2$
- 3) Third random sample ( $d_3$ ) without replacement, considering the misclassified sample in learner  $C_1, C_2$   
-> train a weak learner  $C_3$
- 4) Combine all the weak learners ( $C_1 \sim C_n$ ) via majority voting.



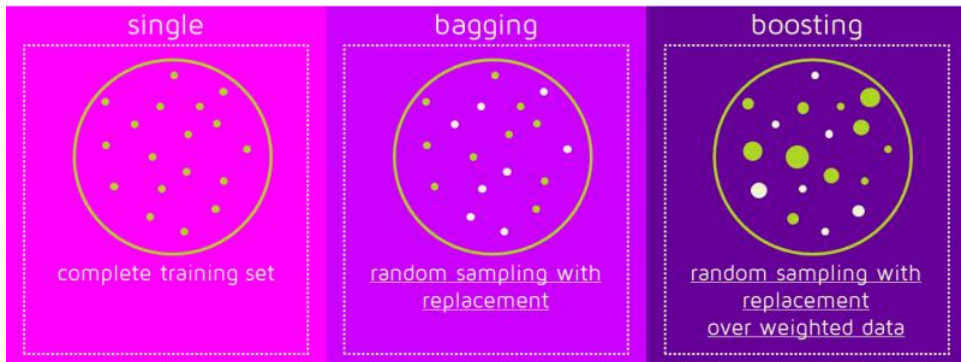
# 1. What is Boosting?

## (2) Bagging vs Boosting

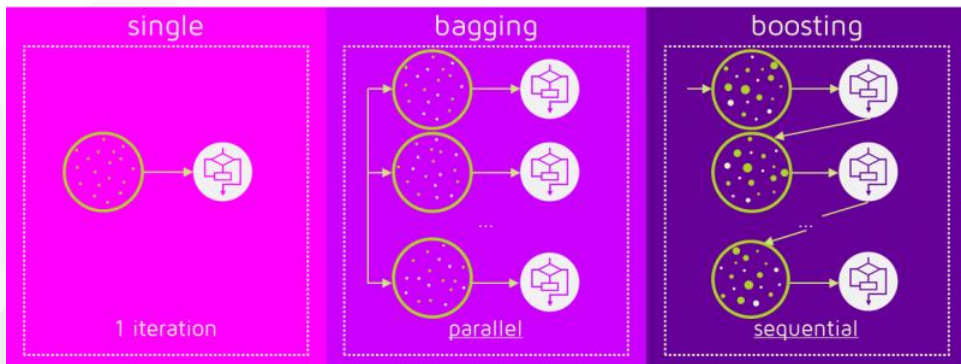
|              | Bagging  | Boosting   |
|--------------|--|--|
|              | Parallel Ensemble<br>( Each model is independent ) | Sequential Ensemble<br>( Next Model is dependent to the previous model ) |
| Sampling     | Random sample ( Bootstrapping )                    | Higher vote to misclassified sample                                      |
| Goal         | Reduce Variance                                    | Reduce Bias  |
| Base Learner | Unstable Learner<br>( ex. Decision Tree )          | Stable Learner<br>( ex. Stump )  |
| Example)     | Random Forest                                      | XGBoost, LightGBM  |

# 1. What is Boosting?

## (2) Bagging vs Boosting



way of sampling



way of making models

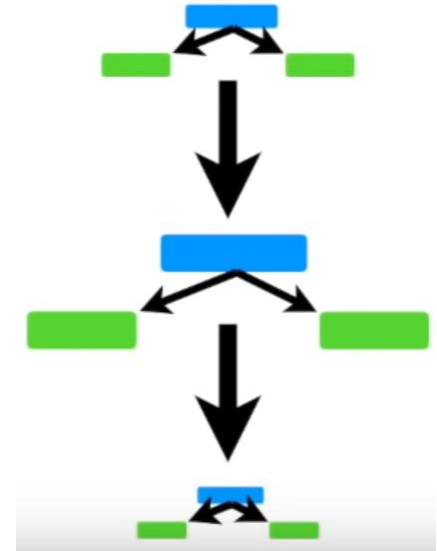
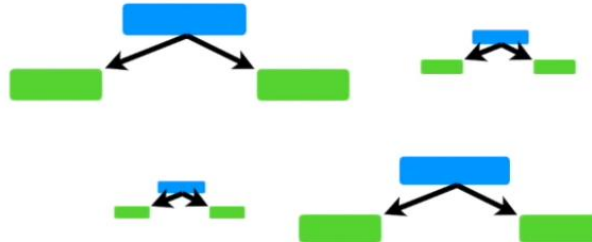




## 2-a. Adaboost

### What is Adaboost?

- Adaboost = **Adaptive Boosting**  
( Adaptive? “more likely to choose the misclassified samples” from the previous learners )
- base learner : **weak learners** ( not very accurate )  
ex) **Stump** ( = a tree with 1 node & 2 leaves )
- **weighted sum** of the output of weak learners  
( = unlike RF, learners get different vote to the final output! )
- weak learners are **dependent** to each other! ( made **sequentially** )



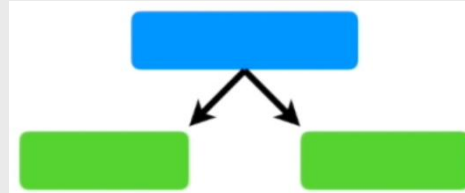
## 2-a. Adaboost

### How does Adaboost work?

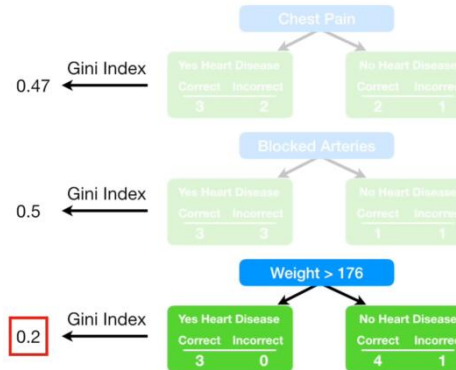
| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |

( How each data is important to be correctly classified ) -> will be updated!

### 1) Make a Stump



with the variable which makes the best classification



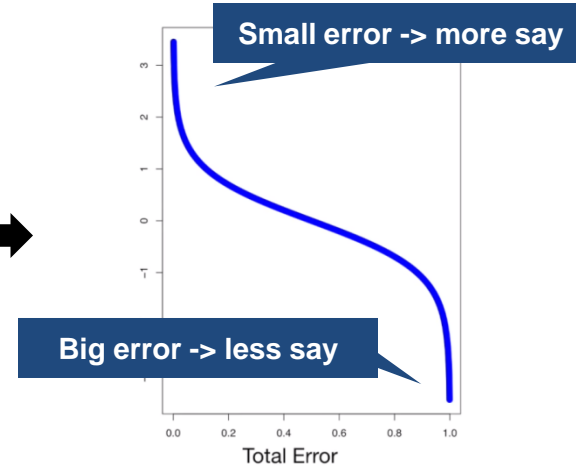
In this example,  
“weight > 176” will be chosen

## 2-a. Adaboost

### How does Adaboost work?

“How much say” does each stump has?

$$\text{Amount of Say} = \frac{1}{2} \log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$



Example)



| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |



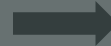
$$\text{Amount of Say} = \frac{1}{2} \log(7) = 0.97$$

## 2-a. Adaboost

### How does Adaboost work?

“How much say” does each stump has?

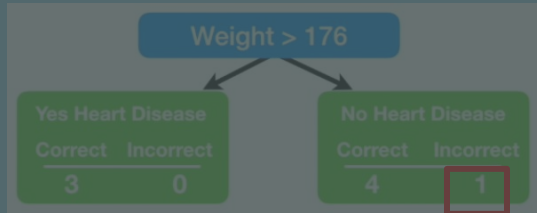
| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |



Need to update!  
(to a SMALLER value)



Need to update!  
(to a LARGER value)



| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |



$$\text{Amount of Say} = \frac{1}{2} \log(7) = 0.97$$

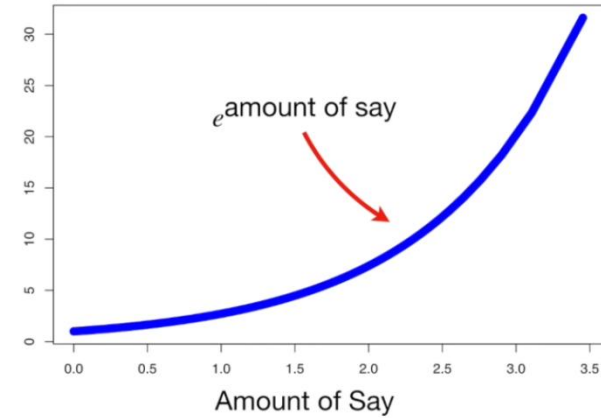
## 2-a. Adaboost

### How does Adaboost work?

< for the “Misclassified” sample >

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

( increase the weight for the “incorrectly classified” sample! )



Example)

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |

$$\text{Amount of Say} = \frac{1}{2} \log(7) = 0.97$$

$$\frac{1}{8} e^{0.97} = \frac{1}{8} \times 2.64 = 0.33$$

Before) 0.125

After ) **0.33**

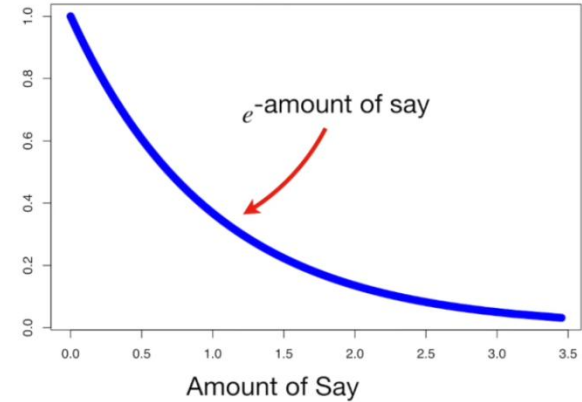
## 2-a. Adaboost

### How does Adaboost work?

< for the “Correctly classified” sample >

$$\text{New Sample Weight} = \text{sample weight} \times e^{-\text{amount of say}}$$

( decrease the weight for the “correctly classified” sample! )



Example)

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |

$$\text{Amount of Say} = \frac{1}{2} \log(7) = 0.97$$

$$\frac{1}{8} e^{-0.97} = \frac{1}{8} \times 0.38 = 0.05$$

Before) 0.125

After ) **0.05**

## 2-a. Adaboost

### How does Adaboost work?

After updating the weight  
( + normalization )

| New Weight | Norm. Weight |
|------------|--------------|
| 0.05       | 0.07         |
| 0.05       | 0.07         |
| 0.05       | 0.07         |
| 0.33       | 0.49         |
| 0.05       | 0.07         |
| 0.05       | 0.07         |
| 0.05       | 0.07         |
| 0.05       | 0.07         |



This becomes the new sample weight!

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 0.07          |
| No         | Yes              | 180            | Yes           | 0.07          |
| Yes        | No               | 210            | Yes           | 0.07          |
| Yes        | Yes              | 167            | Yes           | 0.49          |
| No         | Yes              | 156            | No            | 0.07          |
| No         | Yes              | 125            | No            | 0.07          |
| Yes        | No               | 168            | No            | 0.07          |
| Yes        | Yes              | 172            | No            | 0.07          |



## 2-a. Adaboost

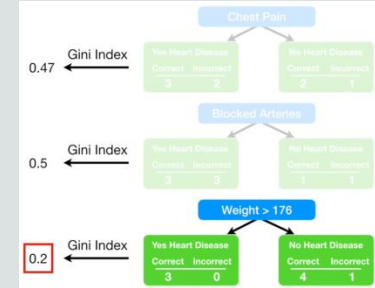
### How does Adaboost work?

Make the next stump,  
with the “new sample weight”

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 0.07          |
| No         | Yes              | 180            | Yes           | 0.07          |
| Yes        | No               | 210            | Yes           | 0.07          |
| Yes        | Yes              | 167            | Yes           | 0.49          |
| No         | Yes              | 156            | No            | 0.07          |
| No         | Yes              | 125            | No            | 0.07          |
| Yes        | No               | 168            | No            | 0.07          |
| Yes        | Yes              | 172            | No            | 0.07          |

Option 1)

Use “weighted Gini index” to select  
the feature to make new stump



Option 2)

Duplicate copies of the samples  
( bigger ‘sample weight’ ->  
more likely to be selected)

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease |
|------------|------------------|----------------|---------------|
| No         | Yes              | 156            | No            |
| Yes        | Yes              | 167            | Yes           |
| No         | Yes              | 125            | No            |
| Yes        | Yes              | 167            | Yes           |
| Yes        | Yes              | 167            | Yes           |
| Yes        | Yes              | 172            | No            |
| Yes        | Yes              | 205            | Yes           |
| Yes        | Yes              | 167            | Yes           |

## 2-a. Adaboost

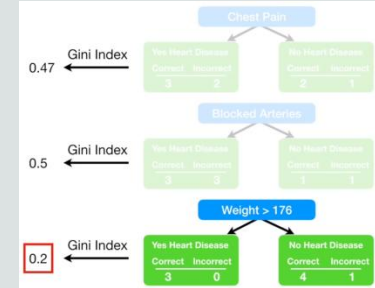
### How does Adaboost work?

Make the next stump,  
with the “new sample weight”

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 0.07          |
| No         | Yes              | 180            | Yes           | 0.07          |
| Yes        | No               | 210            | Yes           | 0.07          |
| Yes        | Yes              | 167            | Yes           | 0.49          |
| No         | Yes              | 156            | No            | 0.07          |
| No         | Yes              | 125            | No            | 0.07          |
| Yes        | No               | 168            | No            | 0.07          |
| Yes        | Yes              | 172            | No            | 0.07          |

Option 1)

Use “weighted Gini index” to select  
the feature to make new stump



Option 2)

Duplicate copies of the samples  
( bigger ‘sample weight’ ->  
more likely to be selected)

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease |
|------------|------------------|----------------|---------------|
| No         | Yes              | 156            | No            |
| Yes        | Yes              | 167            | Yes           |
| No         | Yes              | 125            | No            |
| Yes        | Yes              | 167            | Yes           |
| Yes        | Yes              | 167            | Yes           |
| Yes        | Yes              | 172            | No            |
| Yes        | Yes              | 205            | Yes           |
| Yes        | Yes              | 167            | Yes           |

## 2-a. Adaboost

### How does Adaboost work?

| Chest Pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |
| Yes        | Yes              | 205            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |

Give equal weight to all the samples!

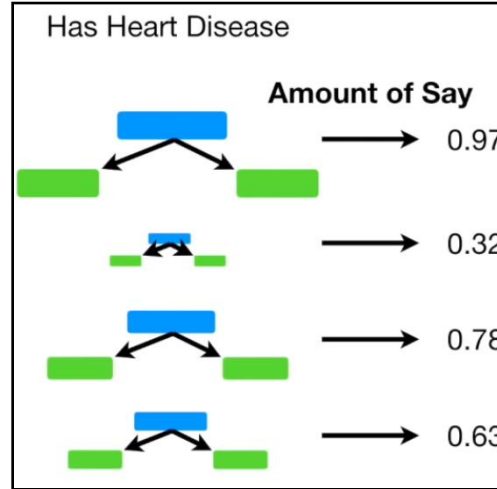
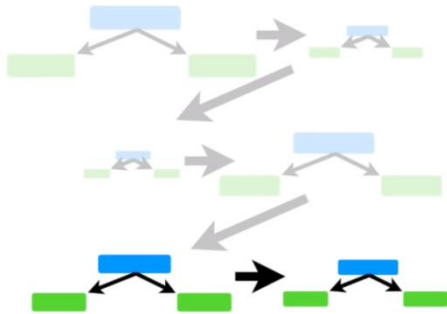
Same sample weights doesn't mean same importance to every sample!

( important samples are duplicated! )

## 2-a. Adaboost

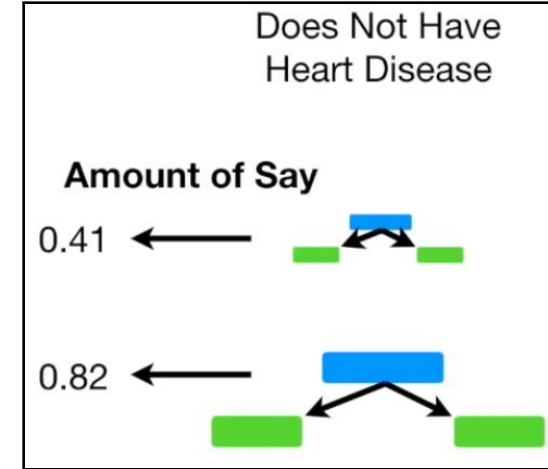
### How does Adaboost work?

Make models(stumps) sequentially



Total Sum : 2.7

>



Total Sum : 1.23

How to predict  
the final output?

Result : "Has Heart Disease"

## How does Adaboost work?

### Binary AdaBoost

Input:

**binary**

- Data:  $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ , where  $x^{(i)} \in \mathcal{X}$  and  $y^{(i)} \in \mathcal{Y} = \{-1, 1\}$  for  $i = 1, \dots, N$
- Total number of rounds:  $T$

**Initialize:** Define the initial probability distribution  $D_1(i) = \frac{1}{N}$  for  $i = 1, \dots, N$ .

**Do for**  $t = 1, \dots, T$ :

- Train using the probability distribution  $D_t$  on  $\{1, \dots, N\}$ .
- Get a hypothesis (classifier)  $h_t : \mathcal{X} \rightarrow \mathcal{Y}$  that has a training error rate  $\varepsilon_t$  (with respect to  $D_t$ ) given by

$$\varepsilon_t = \sum_{i: h_t(x^{(i)}) \neq y^{(i)}} D_t(i).$$

- If  $\varepsilon_t > \frac{1}{2}$ , then set  $T = t - 1$  and abort the loop.
- Set  $\alpha_t = \frac{1}{2} \log \left( \frac{1 - \varepsilon_t}{\varepsilon_t} \right)$ .
- Define the new probability distribution  $D_{t+1}$  by setting

where

$$Z_t = \sum_{i=1}^N D_t(i) \exp(-\alpha_t y^{(i)} h_t(x^{(i)})).$$

**Output:** Define  $g(x) = \sum_{t=1}^T \alpha_t h_t(x)$ . The final hypothesis (classifier)  $H(x)$  is

$$H(x) = \text{sgn}(g(x)).$$



Extend to  
multi-class

### AdaBoost.M1

Input:

**Multi-class**

- Data:  $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ , where  $x^{(i)} \in \mathcal{X}$  and  $y^{(i)} \in \mathcal{Y}$  for  $i = 1, \dots, N$ . Here  $\mathcal{Y}$  is a set of  $K$  elements, which, as usual, is identified with  $\{1, \dots, K\}$
- Total number of rounds:  $T$

**Initialize:** Define the initial probability  $D_1(i) = \frac{1}{N}$  for  $i = 1, \dots, N$ .

**Do for**  $t = 1, 2, \dots, T$ :

- Train using the probability distribution  $D_t$  on  $\{1, \dots, N\}$ .
- Get a hypothesis (classifier)  $h_t : \mathcal{X} \rightarrow \mathcal{Y}$  that has a training error rate  $\varepsilon_t$  (with respect to  $D_t$ ) given by

$$\varepsilon_t = \sum_{i: h_t(x^{(i)}) \neq y^{(i)}} D_t(i).$$

- If  $\varepsilon_t > \frac{1}{2}$ , then set  $T = t - 1$  and abort the loop.
- Set  $\alpha_t = \frac{1}{2} \log \left( \frac{1 - \varepsilon_t}{\varepsilon_t} \right)$ .
- Define the new probability distribution  $D_{t+1}$  by setting

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y^{(i)} = h_t(x^{(i)}) \\ e^{\alpha_t} & \text{if } y^{(i)} \neq h_t(x^{(i)}) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \exp(-\alpha_t S(h_t(x^{(i)}), y^{(i)})), \end{aligned}$$

where  $Z_t = \sum_{i=1}^N D_t(i) \exp(-\alpha_t S(h_t(x^{(i)}), y^{(i)}))$ , and  $S(a, b) = \mathbb{I}(a = b) - \mathbb{I}(a \neq b)$  so that  $S(a, b) = 1$  if  $a = b$  and  $S(a, b) = -1$  if  $a \neq b$ .

**Output:** the final hypothesis (classifier)

$$H(x) = \operatorname{argmax}_{y \in \mathcal{Y}} \sum_{t: h_t(x) = y} \alpha_t.$$



### Adaboost Summary

- 1) Combine **weak learners** to make classifications
- 2) Some learners **have more say** than over learners
- 3) Each learner is made by considering the result of the **previous learner's mistake**



## 2-b. Gradient Boost

## 2-b. Gradient Boost

### What is Gradient Boost?

- not so much different with Adaboost

- difference?

- 1) Combine **weak learners** to make classifications
- 2) Some learners **have more say** than over learners
- 3) Each learner is made by considering the result of the **previous learner's mistake**

Can be expressed in more general terms! ( by using “**gradient**” )

+ use trees larger than stumps as base learners

1. Gradient Boosting Regression

2. Gradient Boosting Classification

$$\begin{aligned} y &= f(x) + \underline{e1} \\ e1 &= g(x) + \underline{e2} \\ e2 &= h(x) + e3 \end{aligned}$$

$$Y = f(x) + g(x) + h(x) + \dots$$



### 1. Gradient Boost Regression

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 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          |

1

First, make a “**leaf** ( = guess for the y value )”

( initial leaf : average of all the y values )

Average Weight  
71.2

( = first predicted y )

| Height (m) | Favorite Color | Gender | Weight (kg) | Residual |
|------------|----------------|--------|-------------|----------|
| 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    |

2

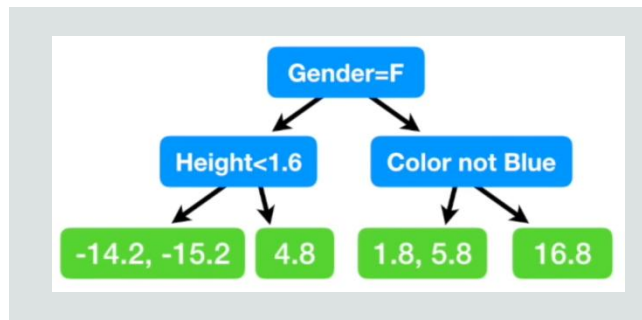
Then, predict the “**residual**”

( build the next model (tree) predicting the residuals )

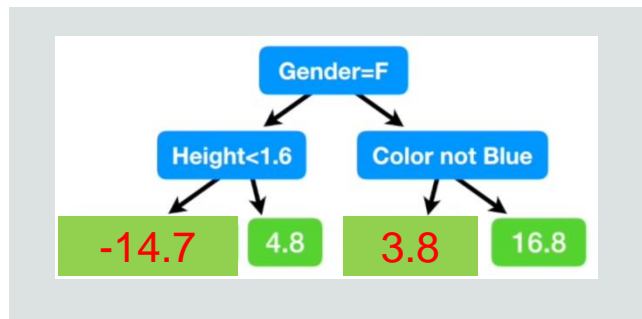
### 1. Gradient Boost Regression

( usually 8~32 leaves )

| Height (m) | Favorite Color | Gender | Weight (kg) | Residual |
|------------|----------------|--------|-------------|----------|
| 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    |



Replace the residuals into their average



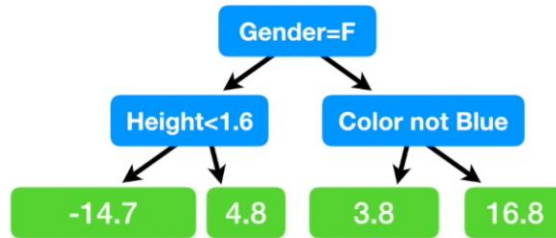
## 2-b. Gradient Boost

### 1. Gradient Boost Regression

Average Weight

71.2

+



| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |

100% correct ( no error )

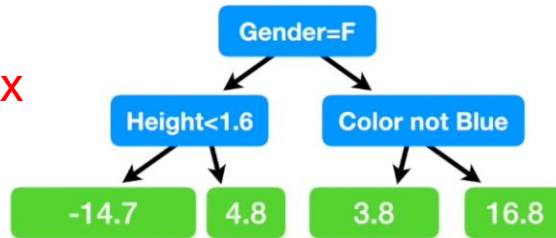
**Overfitting!**

Average Weight

71.2

+

$lr \times$



| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 72.9        |

( If learning rate = 0.1 )

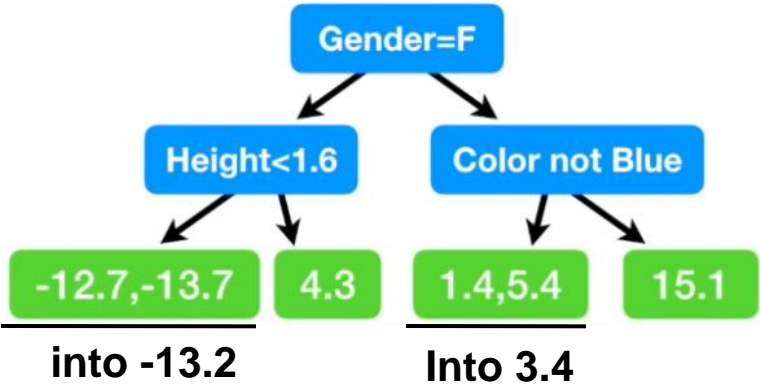
$$71.2 + (0.1 \times 16.8) = 72.9$$



1. Gradient Boost Regression

new residuals  
( becomes smaller )

| Height (m) | Favorite Color                    | Gender | Weight (kg) | Residual |
|------------|-----------------------------------|--------|-------------|----------|
| 1.6        | $88 - (71.2 + 0.1 \times 16.8) =$ |        |             | 15.1     |
| 1.6        | Green                             | Female | 76          | 4.3      |
| 1.5        | Blue                              | Female | 56          | -13.7    |
| 1.8        | Red                               | Male   | 73          | 1.4      |
| 1.5        | Green                             | Male   | 77          | 5.4      |
| 1.4        | Blue                              | Female | 57          | -12.7    |



3

Then, predict the “residual’s residual”  
( build the next model (tree) predicting the residuals )

### 1. Gradient Boost Regression

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 74.4        |

$$71.2 + (0.1 \times 16.8) + (0.1 \times 15.1) = 74.4$$

$$y = f(x) + e1$$

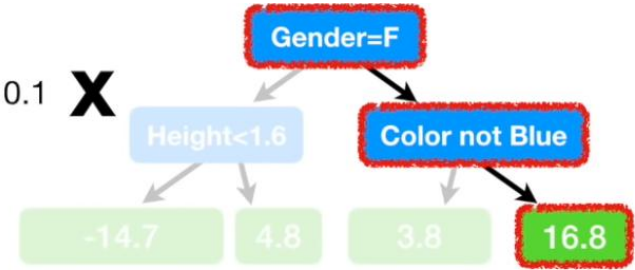
$$e1 = g(x) + e2$$

$$e2 = h(x) + e3$$

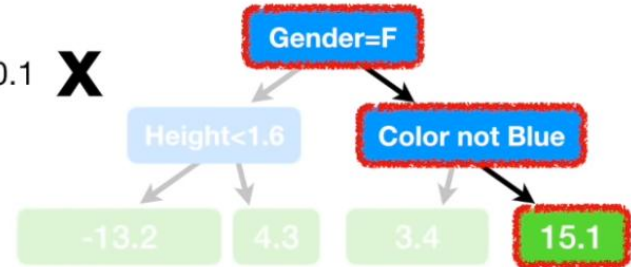
Average Weight

71.2

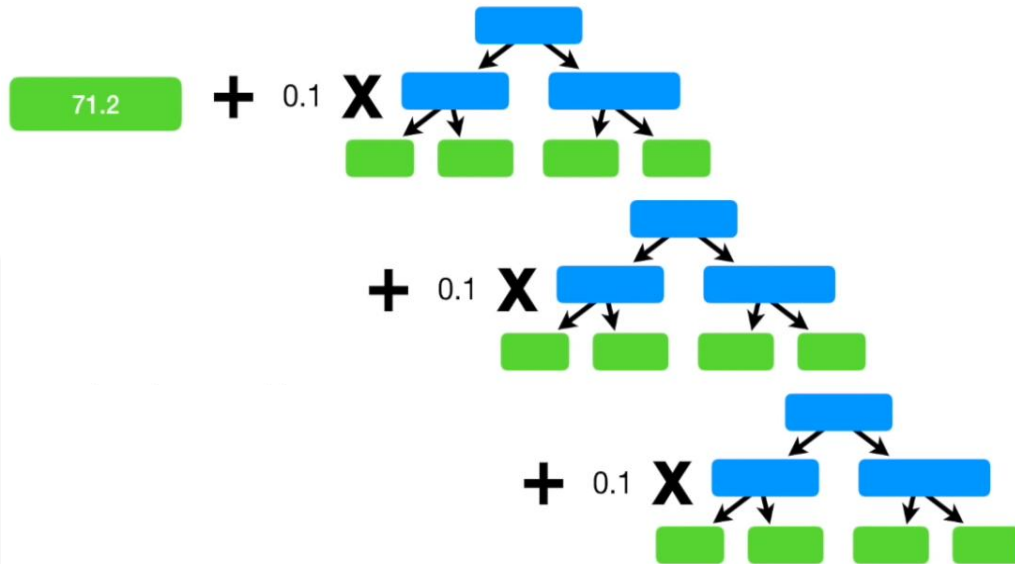
+ 0.1 X



+ 0.1 X



### 1. Gradient Boost Regression



Keep making the sequential trees!

( until specified level, or until the residuals  
doesn't significantly drops )

# 1. Gradient Boost Regression

## Mathematical Expression

$$\text{Loss Function} = \frac{1}{2} (\text{Observed} - \text{Predicted})^2$$



To minimize this loss function

$$\frac{d}{d \text{ Predicted}} \frac{1}{2} (\text{Observed} - \text{Predicted})^2 = -(\text{Observed} - \text{Predicted}) = 0$$

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 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          |

$$\begin{aligned} & \frac{1}{2} (88 - \text{Predicted})^2 + \rightarrow -(88 - \text{Predicted}) + \\ & \frac{1}{2} (76 - \text{Predicted})^2 + \rightarrow -(76 - \text{Predicted}) + \\ & \frac{1}{2} (56 - \text{Predicted})^2 \rightarrow -(56 - \text{Predicted}) \\ & \frac{d}{d \text{ Predicted}} \end{aligned} = 0$$

### 1. Gradient Boost Regression

#### Mathematical Expression

$$\text{Loss Function} = \frac{1}{2} (\text{Observed} - \text{Predicted})^2$$



To minimize this loss function

$$\frac{d}{d \text{ Predicted}} \frac{1}{2} (\text{Observed} - \text{Predicted})^2 = -(\text{Observed} - \text{Predicted}) = 0$$

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |
| 1.6        | Green          | Female | 76          |
| 1.5        | Blue           | Female | 56          |
| 1.8        | Red            | Male   | 75          |
| 1.5        | Green          | Male   | 77          |
| 1.4        | Blue           | Female | 57          |

That is why we kept making the predicted value as the “AVERAGE” value!



### 1. Gradient Boost Regression

<https://ascelibrary.org/cms/asset/>

---

#### Gradient Boosted Regression Tree

---

1. **Input:**  $D = \{(x_1, y_1), \dots, (x_N, y_N)\}, \theta, \gamma$
  2. **Output:**  $F(x) = \sum_{i=0}^M F_i(x)$
  3. Initialize  $F_0(x) = \arg \min_{\beta} \sum_{i=0}^N L(y_i, \beta)$
  4. **While** ( $m < M$ )
    5.  $d_i = -[\partial L(y_i, F(x_i))/\partial F(x_i)]_{F(x_i)=F_{m-1}(x_i)}$
    6.  $\vartheta = \{(x_i, d_i)\}, i = 1, N$
    7.  $g(x) = \text{FITREGRETREE}(\vartheta, \theta)$
    8.  $\rho_m = \arg \min_{\rho} \sum_{i=1}^N L(y_i, F_{m-1}(x) + \rho g(x))$
    9.  $F_m(x) = F_{m-1}(x) + \gamma \rho_m g(x)$
  13. **End while**
- 

**Done!**

## 2-b. Gradient Boost

### 1. Gradient Boost Regression

#### Gradient Boosted Regression Tree

1. **Input:**  $D = \{(x_1, y_1), \dots, (x_N, y_N)\}, \theta, \gamma$
2. **Output:**  $F(x) = \sum_{i=0}^M F_i(x)$
3. Initialize  $F_0(x) = \arg \min_{\beta} \sum_{i=0}^N L(y_i, \beta)$
4. **While** ( $m < M$ )
  - Loss Function : MSE
  - $d_i = -[\partial L(y_i, F(x_i)) / \partial F(x_i)]_{F(x_i)=F_{m-1}(x_i)}$
  - $\vartheta = \{(x_i, d_i)\}, i = 1, N$
  - $g(x) = \text{FITREGRETREE}(\vartheta, \theta)$
  - $\rho_m = \arg \min_{\rho} \sum_{i=1}^N L(y_i, F_{m-1}(x) + \rho g(x))$
  - $F_m(x) = F_{m-1}(x) + \gamma \rho_m g(x)$
13. **End while**

$$y = f(x) + e_1$$
$$e_1 = g(x) + e_2$$
$$e_2 = h(x) + e_3$$

$$Y = f(x) + g(x) + h(x) + \dots$$

First prediction = “average” of all y values

| Height (m) | Color | Gender | Weight (kg) | Residual |
|------------|-------|--------|-------------|----------|
| 1.6        | Blue  | Male   | 80          | 16.8     |
| 1.6        | Green | Female | 76          | 4.8      |
| 1.5        | Blue  | Female | 66          | -15.2    |
| 1.8        | Red   | Male   | 73          | 1.8      |
| 1.5        | Green | Male   | 77          | 5.8      |
| 1.4        | Blue  | Female | 57          | -14.2    |



$$Y = f(x) + g(x) + e_2$$



## 2. Gradient Boost Classification

| Likes Popcorn | Age | Favorite Color | Loves Troll 2 |
|---------------|-----|----------------|---------------|
| Yes           | 12  | Blue           | Yes           |
| Yes           | 87  | Green          | Yes           |
| No            | 44  | Blue           | No            |
| Yes           | 19  | Red            | No            |
| No            | 32  | Green          | Yes           |
| No            | 14  | Blue           | Yes           |



4 Yes, 2 No

Odds ratio (of yes):  $4/2 = 2$

Odds ratio (of no) :  $2/4 = 0.5$

$\text{Log}(2) = 0.7$

$\text{Log}(0.5) = -0.7$

Log Odds Ratio?

1

First, make a “leaf ( = guess for the y value )”

( initial leaf : log odds ratio of the highest value )

$\text{log}(4/2) = 0.7$

( = first predicted y )

## 2. Gradient Boost Classification

### 1) Log Odds Ratio ( = Logit )

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

If  $p$  is a probability, then  $p/(1-p)$  is the corresponding odds

Example)

Chance of winning?

|     |        |
|-----|--------|
| DSL | YBIGTA |
| 5   | 2      |



$$\frac{5}{2}$$

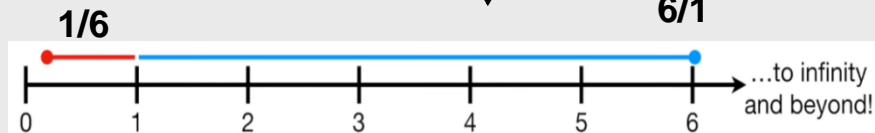
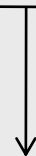
$$(5/7) / (2/7) = 5/2$$

**Odds are 2.5**  
**that DSL will win!**  
( probability  $(p) = 5/7$  )



$$\log(5/2) = 0.916$$

Why log?





### 2. Gradient Boost Classification

#### 2) Logistic Function ( = Sigmoid )

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

As a method to turn 'log odds ratio' back to 'probability'

Example)

( The probability was 5/7 )

$$\log(5/2) = 0.916 \quad \longrightarrow \quad f(\log(5/2)) = 5/7$$

```
>>> 5/7
0.7142857142857143
>>>
>>> 1/(1+np.exp(-np.log(5/2)))
0.7142857142857143
```



2. Gradient Boost Classification

| Likes Popcorn | Age | Favorite Color | Loves Troll 2 |
|---------------|-----|----------------|---------------|
| Yes           | 12  | Blue           | Yes           |
| Yes           | 87  | Green          | Yes           |
| No            | 44  | Blue           | No            |
| Yes           | 19  | Red            | No            |
| No            | 32  | Green          | Yes           |
| No            | 14  | Blue           | Yes           |



4 Yes, 2 No  
Odds ratio (of yes):  $4/2 = 2$   
Odds ratio (of no) :  $2/4 = 0.5$   
  
 $\text{Log}(2) = 0.7$   
 $\text{Log}(0.5) = -0.7$

Therefore, every sample will be classified ‘Yes’ for the initial guess!

1

First, make a “leaf ( = guess for the y value )”  
( initial leaf : log odds ratio of the highest value )  
log(4/2) = 0.7 ( = first predicted y )



(threshold : 0.5)  
Bigger than 0.5 -> classify into ‘Yes’  
Smaller than 0.5 -> classify into ‘No’

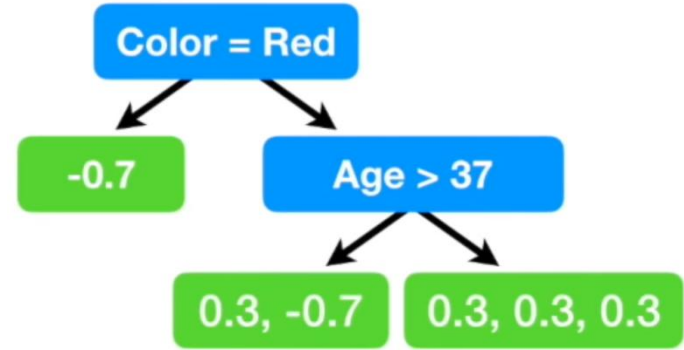
### 2. Gradient Boost Classification

| Likes Popcorn | Age | Favorite Color | Loves Troll 2 | Residual |
|---------------|-----|----------------|---------------|----------|
| Yes           | 12  | Blue           | Yes           | 0.3      |
| Yes           | 87  | Green          | Yes           | 0.3      |
| No            | 44  | Blue           | No            | -0.7     |
| Yes           | 19  | Red            | No            | -0.7     |
| No            | 32  | Green          | Yes           | 0.3      |
| No            | 14  | Blue           | Yes           | 0.3      |

( Yes = 1, No = 0 )

( = 1(actual) – 0.7(predicted) )

( = 0(actual) – 0.7(predicted) )



2

Calculate **Error** based on the first prediction

3

Make a tree, **predicting the “residuals”**

### 2. Gradient Boost Classification

( in Regression )



A bit different in Classification



( in Classification )



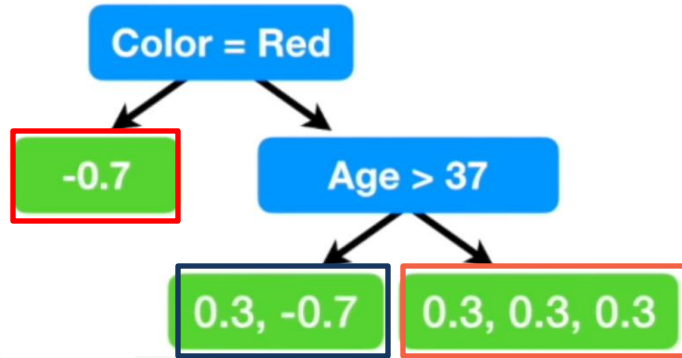
Log odds ratio

Probability

$$\frac{\sum \text{Residual}_i}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)]}$$



### 2. Gradient Boost Classification

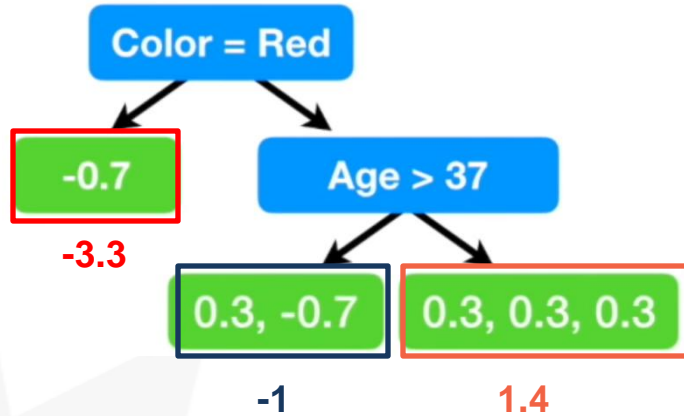


$$\frac{\sum \text{Residual}_i}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)]}$$

$$\frac{-0.7}{0.7 \times (1 - 0.7)} \quad (= -3.3)$$

$$\frac{0.3 + -0.7}{(0.7 \times (1 - 0.7)) + (0.7 \times (1 - 0.7))} \quad (= -1)$$

### 2. Gradient Boost Classification



Replace the value!  
( to be compatible with 'log odds ratio' )

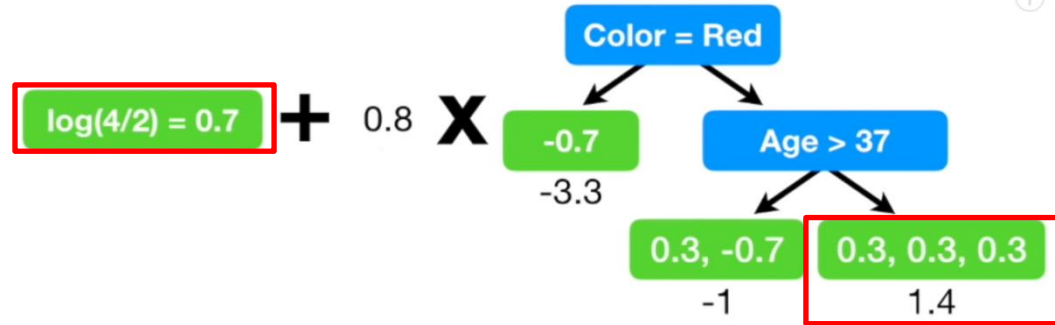
$$\frac{\sum \text{Residual}_i}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)]}$$

$$\frac{-0.7}{0.7 \times (1 - 0.7)} \quad (= -3.3)$$

$$\frac{0.3 + -0.7}{(0.7 \times (1 - 0.7)) + (0.7 \times (1 - 0.7))} \quad (= -1)$$

### 2. Gradient Boost Classification

| Likes Popcorn | Age | Favorite Color | Loves Troll 2 |
|---------------|-----|----------------|---------------|
| Yes           | 12  | Blue           | Yes           |
| Yes           | 87  | Green          | Yes           |
| No            | 44  | Blue           | No            |
| Yes           | 19  | Red            | No            |



Prediction (before) : 0.7

Prediction (after) :  $0.7 + (0.8 \times 1.4) = 1.82$

convert 1.82 (log odds ratio) into 0.9 (probability)

Make a new prediction! ( considering the tree newly made )

## 2-b. Gradient Boost

### 2. Gradient Boost Classification

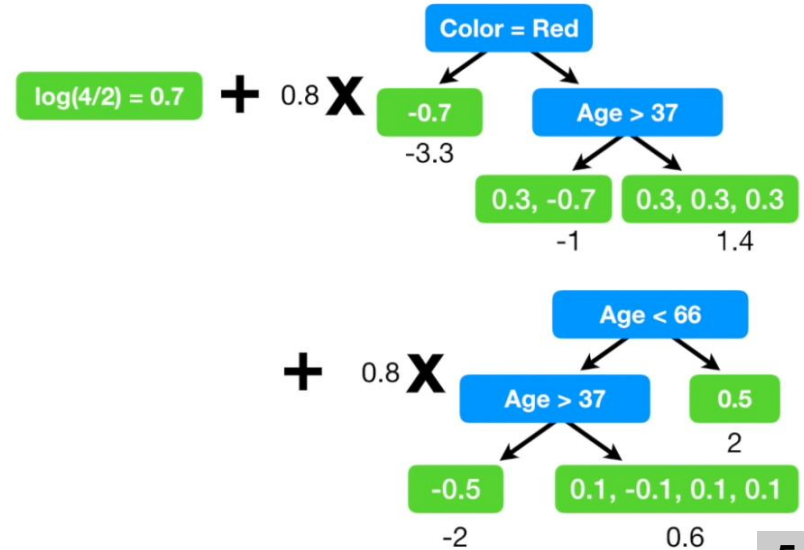
| Likes Popcorn | Age | Favorite Color | Loves Troll 2 | Predicted Prob. | Residual |
|---------------|-----|----------------|---------------|-----------------|----------|
| Yes           | 12  | Blue           | Yes           | 0.9             | 0.1      |
| Yes           | 87  | Green          | Yes           | 0.5             | 0.5      |
| No            | 44  | Blue           | No            | 0.5             | -0.5     |
| Yes           | 19  | Red            | No            | 0.1             | -0.1     |
| No            | 32  | Green          | Yes           | 0.9             | 0.1      |
| No            | 14  | Blue           | Yes           | 0.9             | 0.1      |

( = 1(actual) – 0.9(predicted) )

( = 1(actual) – 0.5(predicted) )

4

**Calculate error**, based on the second prediction!



5

Keep making **sequential trees**!



## 2. Gradient Boost Classification

Ex) How would the person like below be classified?

| Likes Popcorn | Age | Favorite Color | Loves Troll 2 |
|---------------|-----|----------------|---------------|
| Yes           | 25  | Green          | ???           |



$$0.7 + (0.8 \times 1.4) + (0.8 \times 0.6) = 2.3$$

( into Logistic function )

$$f(2.3) = \exp(2.3) / (1 + \exp(2.3)) = 0.9$$

Therefore, it will be classified as 'Yes'



### 2. Gradient Boost Classification

#### Mathematical Expression

$$\text{Loss Function : } \text{LogLoss} = -\frac{1}{n} \sum_{i=0}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

( also called binary-cross entropy )

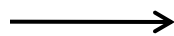
$$- \left[ \text{Observed} \times \log(p) + (1 - \text{Observed}) \times \log(1 - p) \right]$$

Case 1) Actual(=Observed) = 1, Predicted = 1



Log loss : 0

Case 2) Actual(=Observed) = 1, Predicted = 0.5



Log loss : 0.69

Case 3) Actual(=Observed) = 1, Predicted = 0



Log loss : infinity

## 2. Gradient Boost Classification

---

### Gradient Boosted Regression Tree

---

1. **Input:**  $D = \{(x_1, y_1), \dots, (x_N, y_N)\}, \theta, \gamma$
  2. **Output:**  $F(x) = \sum_{i=0}^M F_i(x)$
  3. Initialize  $F_0(x) = \arg \min_{\beta} \sum_{i=0}^N L(y_i, \beta)$
  4. **While** ( $m < M$ )
    5.  $d_i = -[\partial L(y_i, F(x_i)) / \partial F(x_i)]_{F(x_i)=F_{m-1}(x_i)}$
    6.  $\vartheta = \{(x_i, d_i)\}, i = 1, N$
    7.  $g(x) = \text{FITREGRETREE}(\vartheta, \theta)$
    8.  $\rho_m = \arg \min_{\rho} \sum_{i=1}^N L(y_i, F_{m-1}(x) + \rho g(x))$
    9.  $F_m(x) = F_{m-1}(x) + \gamma \rho_m g(x)$
  13. **End while**
-

2. Gradient Boost Classification

| Gradient Boosted | Classification   | Tree |
|------------------|--|------|
| 1.               | <b>Input:</b> $D = \{(x_1, y_1), \dots, (x_N, y_N)\}, \theta, \gamma$        |      |
| 2.               | <b>Output:</b> $F(x) = \sum_{i=0}^M F_i(x)$                                  |      |
| 3.               | Initialize $F_0(x) = \arg \min_{\beta} \sum_{i=0}^N L(y_i, \beta)$           |      |
| 4.               | <b>While</b> ( $m < M$ )   |      |
| 5.               | $d_i = -[\partial L(y_i, F(x_i)) / \partial F(x_i)]_{F(x_i) = F_{m-1}(x_i)}$ |      |
| 6.               | $\vartheta = \{(x_i, d_i)\}, i = 1, N$                                       |      |
| 7.               | $g(x) = \text{FITREGRETREE}(\vartheta, \theta)$                              |      |
| 8.               | $\rho_m = \arg \min_{\rho} \sum_{i=1}^N L(y_i, F_{m-1}(x) + \rho g(x))$      |      |
| 9.               | $F_m(x) = F_{m-1}(x) + \gamma \rho_m g(x)$                                   |      |
| 13.              | <b>End while</b>   |      |

Almost the same as the Regression Model!



## 2-b. Gradient Boost

### 2. Gradient Boost Classification

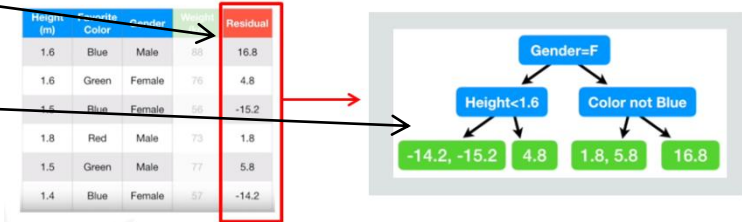
#### Gradient Boosted Regression Tree

1. **Input:**  $D = \{(x_1, y_1), \dots, (x_N, y_N)\}, \theta, \gamma$
2. **Output:**  $F(x) = \sum_{i=0}^M F_i(x)$
3. Initialize  $F_0(x) = \arg \min_{\beta} \sum_{i=0}^N L(y_i, \beta)$
4. **While** ( $m < M$ )
  - Loss Function : **MSE**
  - $d_i = -[\partial L(y_i, F(x_i)) / \partial F(x_i)]_{F(x_i)=F_{m-1}(x_i)}$
  - $\vartheta = \{(x_i, d_i)\}, i = 1, N$
  - $g(x) = \text{FITREGTREETREE}(\vartheta, \theta)$
  - $\rho_m = \arg \min_{\rho} \sum_{i=1}^N L(y_i, F_{m-1}(x) + \rho g(x))$
  - $F_m(x) = F_{m-1}(x) + \gamma \rho_m g(x)$
13. **End while**

$$y = f(x) + e_1$$
$$e_1 = g(x) + e_2$$
$$e_2 = h(x) + e_3$$

$$Y = f(x) + g(x) + h(x) + \dots$$

First prediction = “**average**” of all y values



$$Y = f(x) + g(x) + e_2$$

## 2. Gradient Boost Classification

### Gradient Boosted Classification Tree

1. **Input:**  $D = \{(x_1, y_1), \dots, (x_N, y_N)\}, \theta, \gamma$
2. **Output:**  $F(x) = \sum_{i=0}^M F_i(x)$
3. Initialize  $F_0(x) = \arg \min_{\beta} \sum_{i=0}^N L(y_i, \beta)$
4. **While** ( $m < M$ )
 

Loss Function : **Log Loss**
5.  $d_i = -[\partial L(y_i, F(x_i)) / \partial F(x_i)]_{F(x_i)=F_{m-1}(x_i)}$
6.  $\vartheta = \{(x_i, d_i)\}, i = 1, N$
7.  $g(x) = \text{FITREGRETREE}(\vartheta, \theta)$
8.  $\rho_m = \arg \min_{\rho} \sum_{i=1}^N L(y_i, F_{m-1}(x) + \rho g(x))$
9.  $F_m(x) = F_{m-1}(x) + \gamma \rho_m g(x)$
13. **End while**

$$y = f(x) + e1$$

$$e1 = g(x) + e2$$

$$e2 = h(x) + e3$$

$$Y = f(x) + g(x) + h(x) + \dots$$

First prediction = “log odds ratio” of the mode

| Height (m) | Color | Gender | Weight (kg) | Residual |
|------------|-------|--------|-------------|----------|
| 1.6        | Blue  | Male   | 80          | 16.8     |
| 1.6        | Green | Female | 76          | 4.8      |
| 1.5        | Blue  | Female | 66          | -15.2    |
| 1.8        | Red   | Male   | 73          | 1.8      |
| 1.5        | Green | Male   | 77          | 5.8      |
| 1.4        | Blue  | Female | 57          | -14.2    |



$$Y = f(x) + g(x) + e2$$



## 2-c. XGBoost



### 1. Characteristics of XGBoost

XGBoost ( eXtreme Gradient Boost )

Consists of lots of parts!

- 1) Gradient Boost
- 2) Regularization
- 3) Unique Regression Tree
- 4) Approximate Greedy Algorithm
- 5) Weighted Quantile Sketch
- 6) Sparsity-Aware Split Finding
- 7) Parallel Learning
- 8) Cache-Aware Access
- 9) Blocks for Out-of-Core Computation

### 1. XGBoost Regression

### 2. XGBoost Classification



## **2. XGBoost Regression**



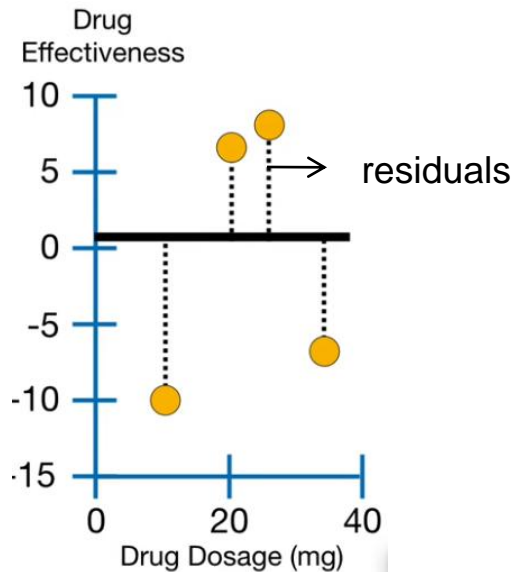
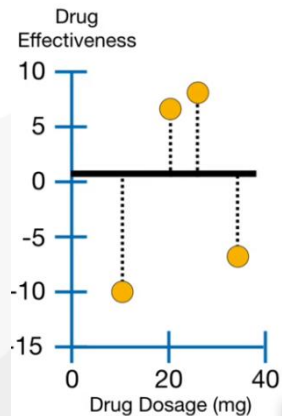
### 2. XGBoost Regression

#### Unique Regression Tree!

Example with simple data)

X : drug dosage

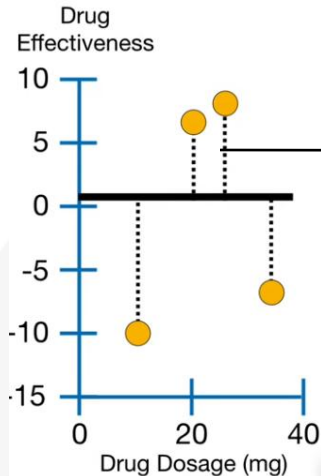
Y : drug Effectiveness



1

Initial Prediction : **0.5 (default)**  
( both in Regression & Classification )

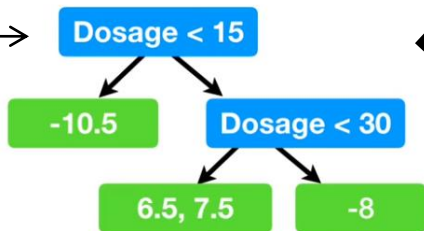
### 2. XGBoost Regression



( same as non-extreme GB ) **2**

Then, predict the “**residual**”

( build the next model (tree) predicting the residuals )



### HOW?

#### Step 1.

Put all the residuals to the same leaf

**-10.5, 6.5, 7.5, -7.5**

And calculate a “**similarity score**”

$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

( lambda : regularization term )

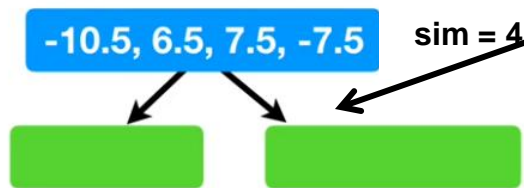
$$\frac{(-10.5 + 6.5 + 7.5 + -7.5)^2}{4 + \mathbf{0}} = 4$$

## 2. XGBoost Regression

### HOW?

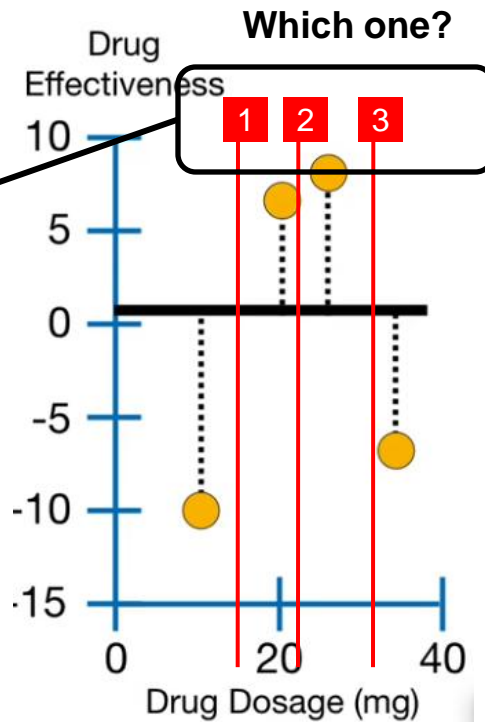
#### Step 2.

Split the leaf, which gives the most **“Gain”**



- 1 Drug Dosage < 15
- 2 Drug Dosage < 22.5
- 3 Drug Dosage < 15

Which split criterion  
should we choose?





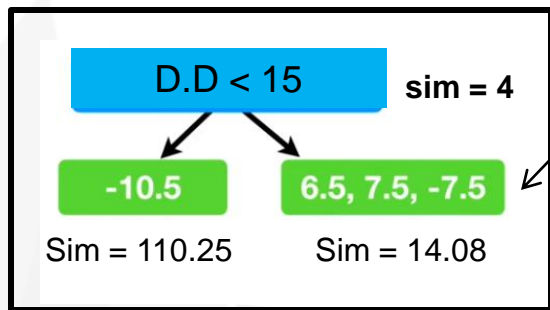
## 2. XGBoost Regression

- 1 Drug Dosage < 15
- 2 Drug Dosage < 22.5
- 3 Drug Dosage < 30

Which split criterion  
should we choose?

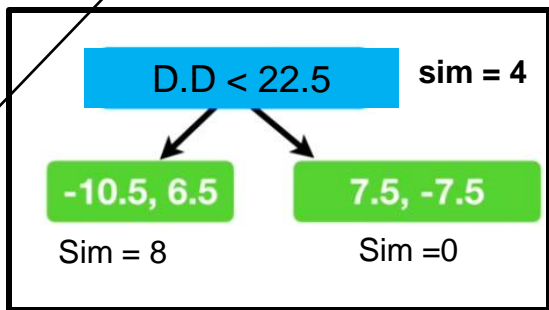
Example)

$$14.08 = \frac{(6.5 + 7.5 + -7.5)^2}{3 + 0}$$



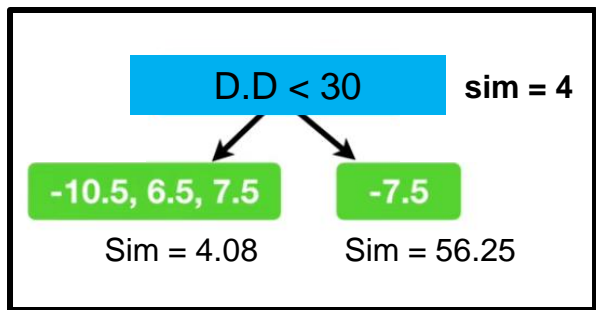
Gain

$$(110.25 + 14.08) - 4 = 120.33$$



Gain

$$(8 + 0) - 4 = 4$$



Gain

$$(4.08 + 56.25) - 4 = 56.33$$

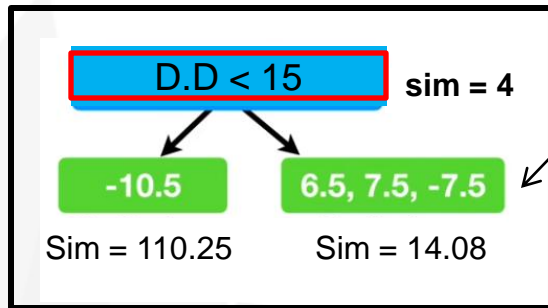
## 2. XGBoost Regression

- 1 Drug Dosage < 15
- 2 Drug Dosage < 22.5
- 3 Drug Dosage < 30

Which split criterion  
should we choose?

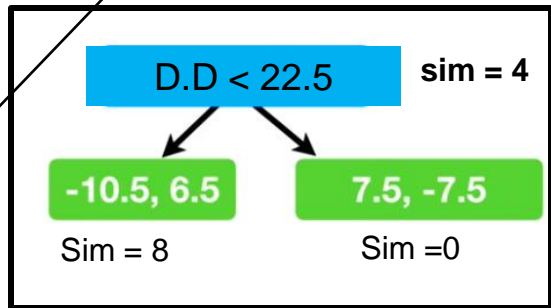
Example)

$$14.08 = \frac{(6.5 + 7.5 + -7.5)^2}{3 + 0}$$



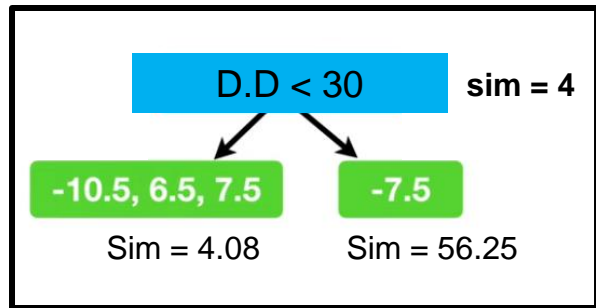
Gain

$$(110.25 + 14.08) - 4 = 120.33$$



Gain

$$(8 + 0) - 4 = 4$$

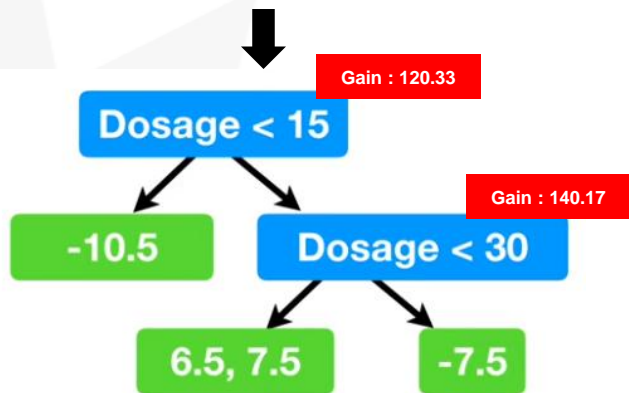
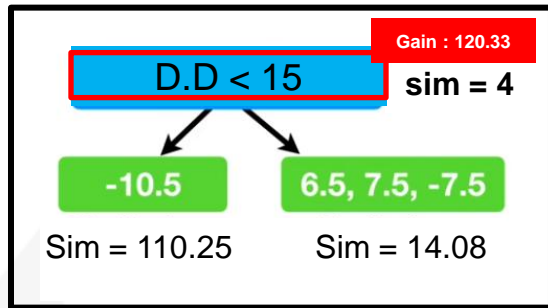


Gain

$$(4.08 + 56.25) - 4 = 56.33$$

### 2. XGBoost Regression

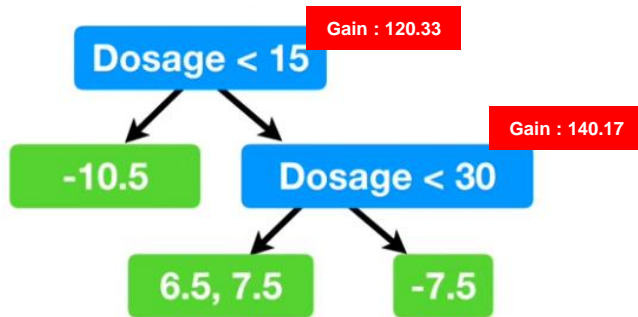
choose "D.D < 15" as the criterion



### HOW?

#### Step 3.

Prune the tree ( if necessary )



New parameter : 'gamma'

If gamma = 130 :

( 140.17 > 130 ) -> Prune X

If gamma = 150 :

( 140.17 < 150 ) -> Prune O -> ( 120.33 < 150 ) -> Prune O

**To Prevent  
Overfitting!**

## 2. XGBoost Regression

Another method for preventing overfitting ( regularization )

$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

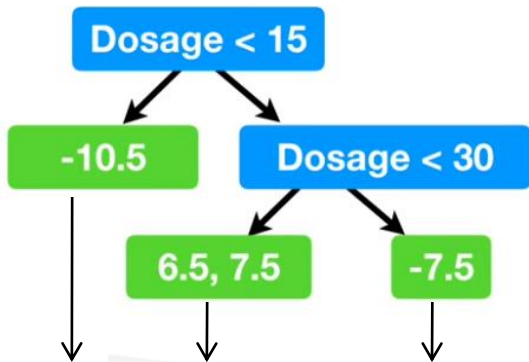
What happens if **lambda** gets larger?

lambda  $\uparrow$   $\rightarrow$  similarity score  $\downarrow$   $\rightarrow$  More chance of getting pruned

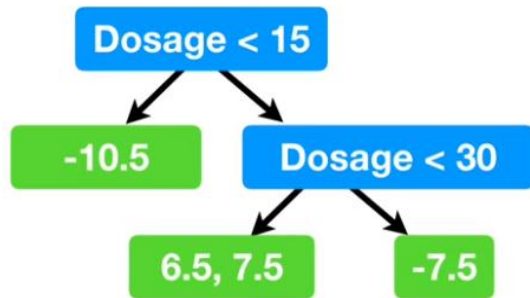
### 2. XGBoost Regression

#### Step 4.

Predict output values of each leaf !



$$\text{Output Value} = \frac{\text{Sum of Residuals}}{\text{Number of Residuals} + \lambda}$$

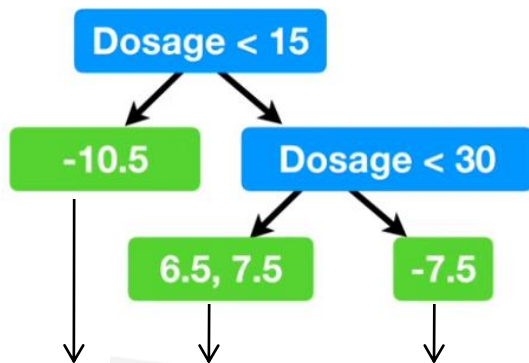


How will these values (of leaves) change?

## 2. XGBoost Regression

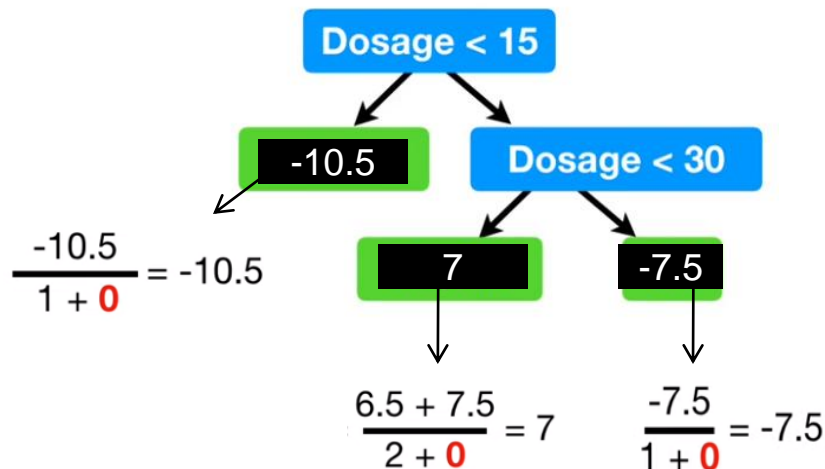
### Step 4.

Predict output values of each leaf !



these are all RESIDUALS, not output values!

$$\text{Output Value} = \frac{\text{Sum of Residuals}}{\text{Number of Residuals} + \lambda}$$



## 2. XGBoost Regression

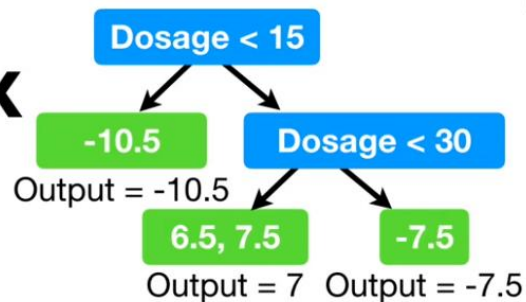
### Step 5.

Predict the final outcome!

Predicted Drug  
Effectiveness  
**0.5**

+

Learning Rate **X**



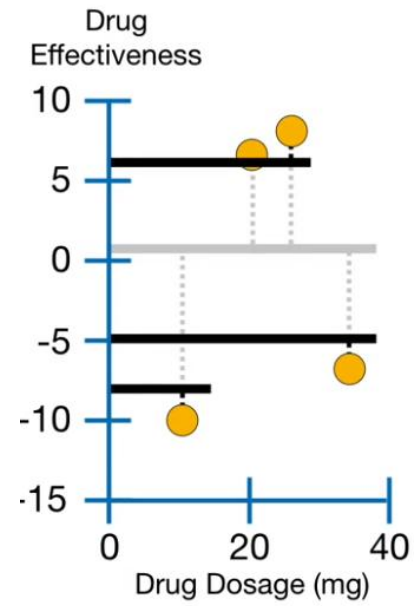
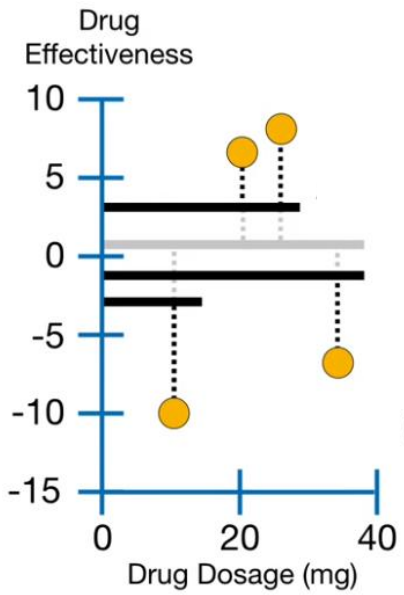
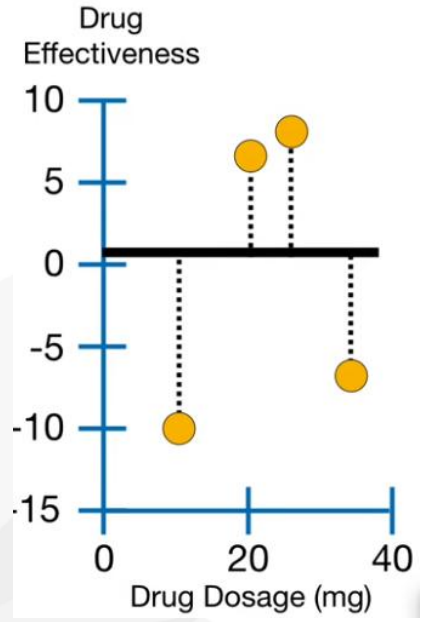
### 2. XGBoost Regression

#### Step 5.

Predict the final outcome!

Getting a more precise prediction as the model is made!

( But be careful of overfitting! )







## **3. XGBoost Classification**

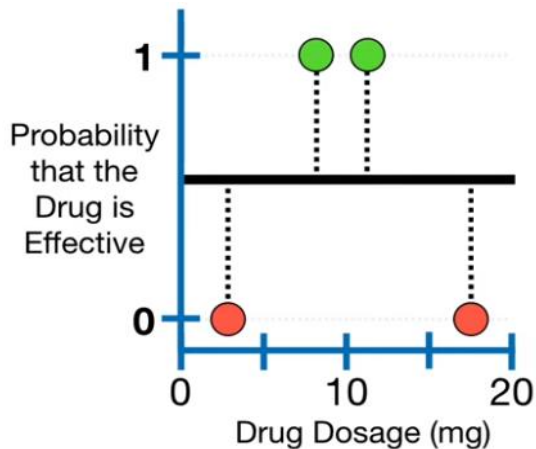
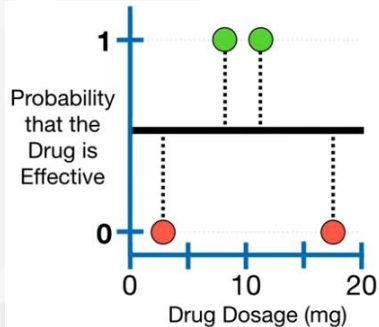
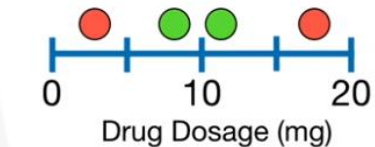


### 3. XGBoost Classification

Example with simple data)

X : drug dosage

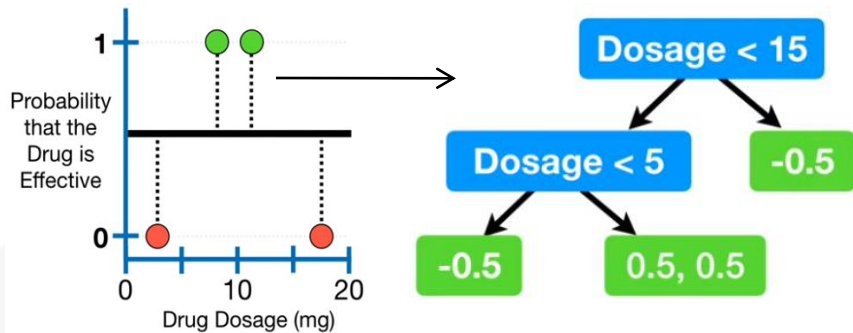
Y : Effective / Not Effective



1

Initial Prediction : 0.5 (default)  
( both in Regression & Classification )

### 3. XGBoost Classification



( same as non-extreme GB ) **2**

Then, predict the “**residual**”

( build the next model (tree) predicting the residuals )

### HOW?

#### Step 1.

Put all the residuals to the same leaf

**-0.5, 0.5, 0.5, -0.5**

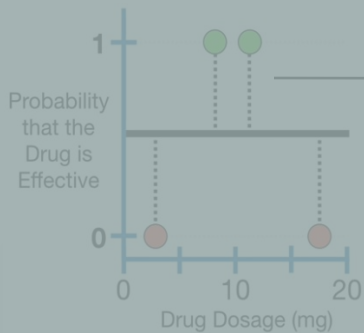
And calculate a “**similarity score**”

$$\frac{(\sum \text{Residual}_i)^2}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda}$$

( lambda : regularization term )

$$\frac{(-0.5 + 0.5 + 0.5 + -0.5)^2}{4 \times (0.5 \times (1 - 0.5))} = 0$$

### 3. XGBoost Classification



Nothing is different from XGBoost Regression, except the “similarity score”

( + similarity is just the same as what we saw in Gradient Boost Classification )

Then, predict the “residual”

( build the next model (tree) predicting the residuals )

HOW?

Step 1.

Put all the residuals to the same leaf

-0.5, 0.5, 0.5, -0.5

Architected a “similarity score”

$$\frac{(\sum \text{Residual}_i)^2}{\sum [\text{Previous Probability} \times (1 - \text{Previous Probability})]}$$

( lambda : regularization term )

$$\frac{(-0.5 + 0.5 + 0.5 + -0.5)^2}{4 \times (0.5 \times (1 - 0.5))} = 0$$

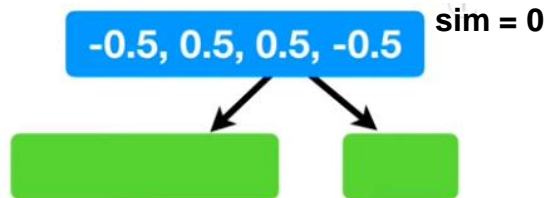
## 2-c. XGBoost

### 2. XGBoost Regression

HOW?

**Step 2.**

Split the leaf, which gives the most **“Gain”**

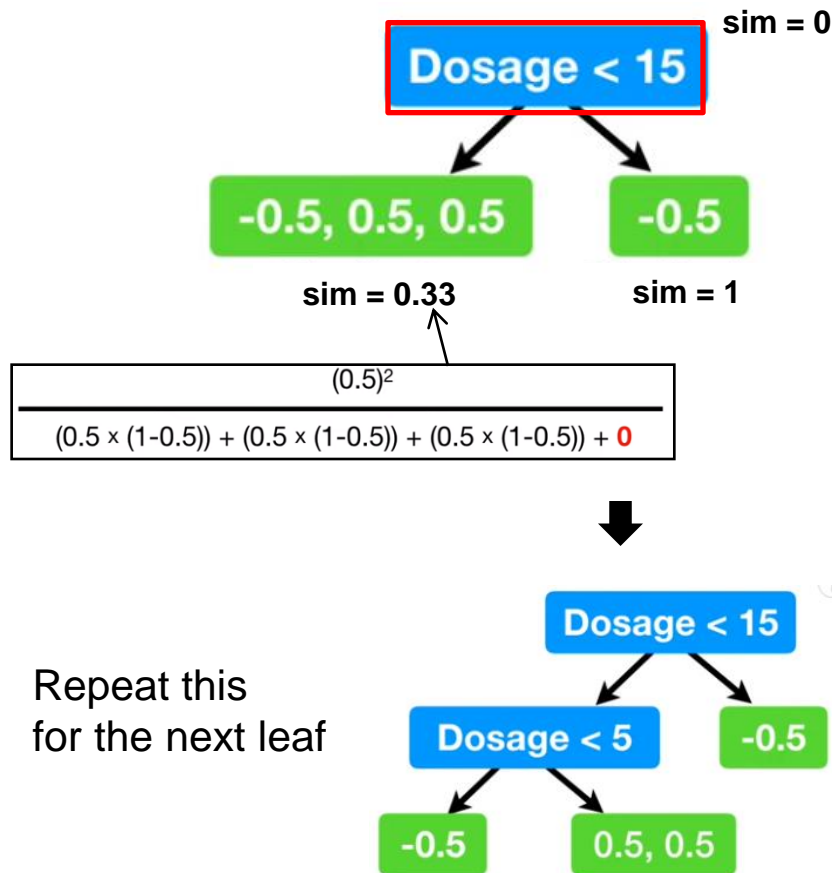


**1** Drug Dosage < 15

**2** Drug Dosage < 22.5

**3** Drug Dosage < 15

Which split criterion  
should we choose?





### 3. XGBoost Classification

#### Step 3.

Pruning the tree

- parameter '**gamma**'  
( minimum gain to make a split! )
- parameter '**lambda**'  
( regularization term )



### 3. XGBoost Classification

#### Step 3.

Pruning the tree

- parameter '**gamma**'  
( minimum gain to make a split! )
- parameter '**lambda**'  
( regularization term )
- **minimum number of residuals in each leaf**  
( new term, '**cover**' )

### 3. XGBoost Classification

#### Step 3.

#### Pruning the tree

- parameter '**gamma**'  
( minimum gain to make a split! )
- parameter '**lambda**'  
( regularization term )
- **minimum number of residuals in each leaf**  
( new term, '**cover**' )

$$(\sum \text{Residual}_i)^2$$

$$\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda$$

( in regression )

$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

- cover can be defined as 'number of residuals'
- default value : 1 -> means that it has no affect on the growth of tree



### 3. XGBoost Classification

#### Step 3.

#### Pruning the tree

- parameter '**gamma**'  
( minimum gain to make a split! )
- parameter '**lambda**'  
( regularization term )
- minimum number of residuals in each leaf  
( new term, '**cover**' )

$$(\sum \text{Residual}_i)^2$$

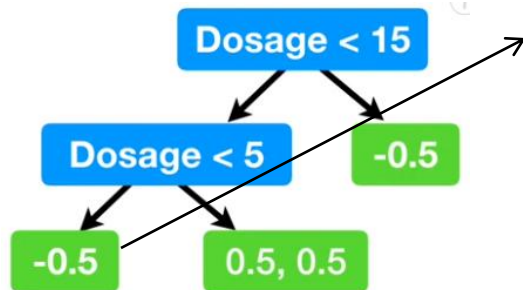
$$\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda$$

( in regression )

$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

- cover can be defined as 'number of residuals'
- default value : 1 -> means that it has no affect on the growth of tree

Example)



'cover' for this leaf is  
 $1 \times ( 0.5 \times (1-0.5) ) = 0.25$

### 3. XGBoost Classification

#### Step 3.

#### Pruning the tree

- parameter '**gamma**'  
( minimum gain to make a split! )
- parameter '**lambda**'  
( regularization term )
- minimum number of residuals in each leaf  
( new term, '**cover**' )

we call this parametr 'cover', "**min\_child\_weight**"

$$(\sum \text{Residual}_i)^2$$

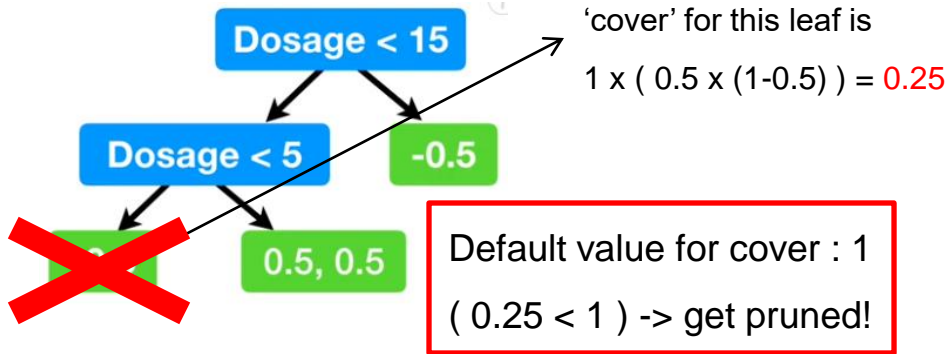
$$\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda$$

( in regression )

$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

- cover can be defined as 'number of residuals'
- default value : 1 -> means that it has no affect on the growth of tree

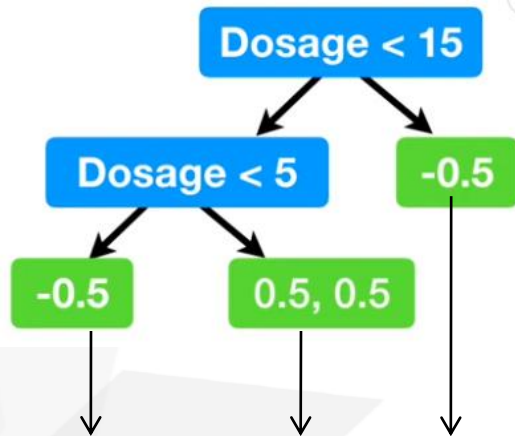
Example)



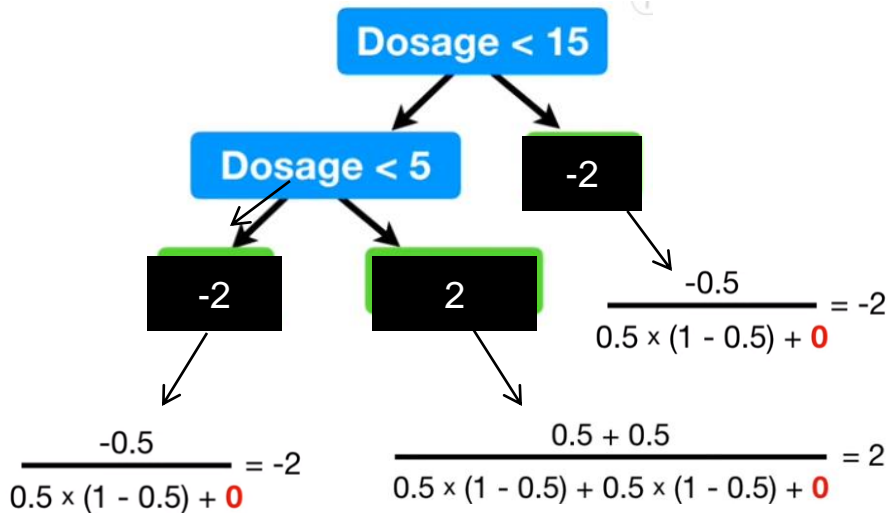
### 3. XGBoost Classification

#### Step 4.

Predict the output values of each leaf



$$\text{Output value} = \frac{(\sum \text{Residual}_i)}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda}$$



### 3. XGBoost Classification

#### Step 5.

Predict the final outcome

Predicted Drug  
Effectiveness

0.5



learning rate



Dosage < 15

Dosage < 5

-0.5

Output  
= -2

-0.5

Output  
= -2

0.5, 0.5

Output  
= 2

...

...

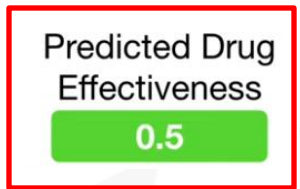
...

?

### 3. XGBoost Classification

**Step 5.**

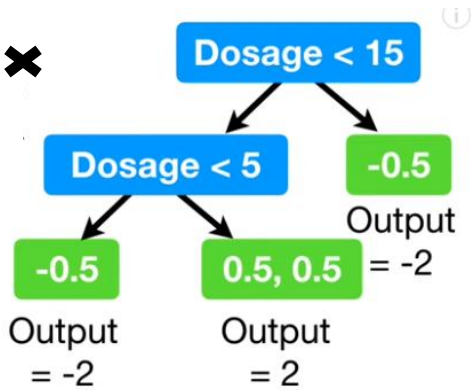
Predict the final outcome



+

learning rate

×



Since it is classification problem,  
Need to change this to 'log(odds)'

...

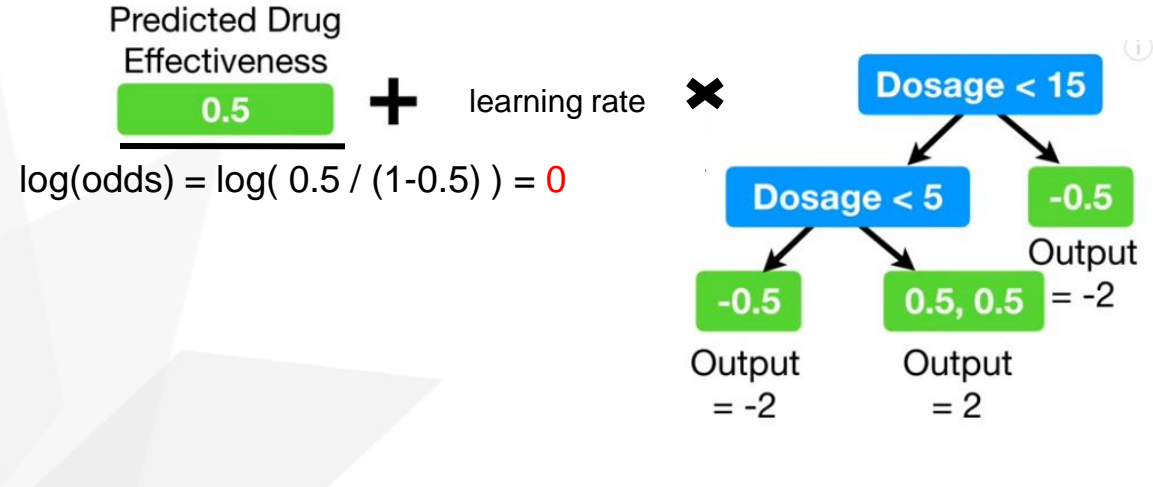
...

...

## 3. XGBoost Classification

### Step 5.

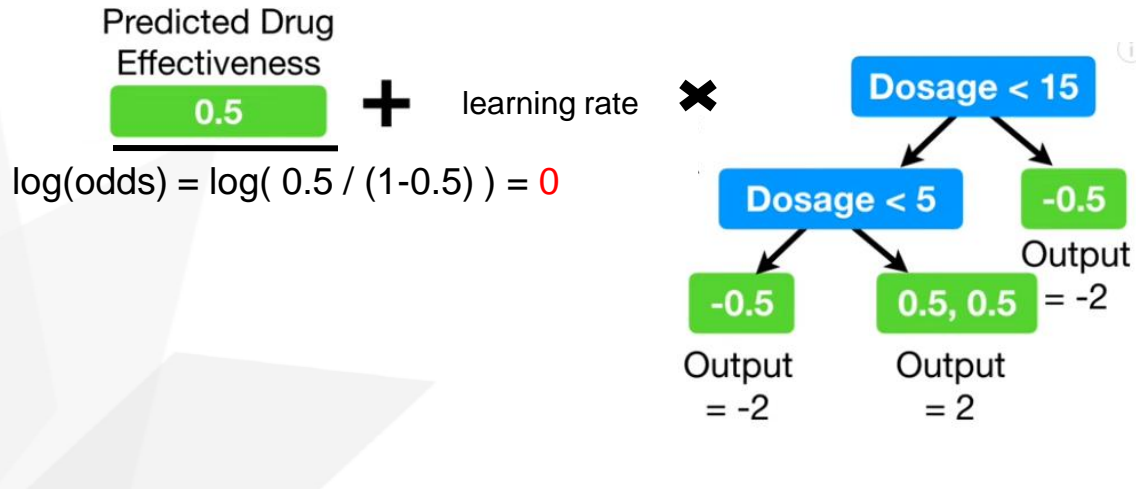
Predict the final outcome



## 3. XGBoost Classification

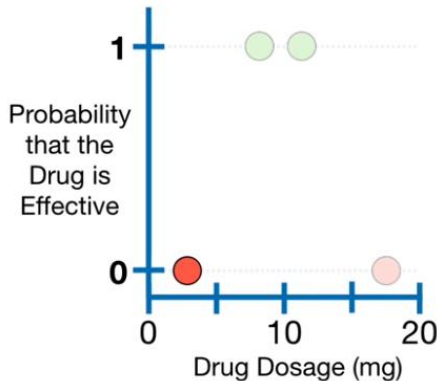
### Step 5.

Predict the final outcome



### 3. XGBoost Classification

Ex) What is the predicted class for the sample below, with the red color? ( let learning rate = 0.3 )



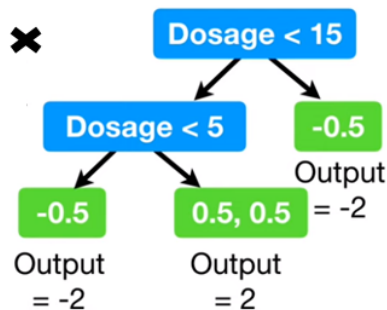
Predicted Drug Effectiveness

$$\frac{0.5}{\log(\text{odds}) = \log(0.5 / (1-0.5)) = 0}$$

+

learning rate

×



Answer :

$$1) 0 + (0.3 \times (-2)) = -0.6$$

( this is in terms of 'log(odds)' ! Have to convert into probability )

$$2) \text{Probability} = \frac{e^{-0.6}}{1 + e^{-0.6}} = 0.35 \rightarrow \text{Predicted as 'NO EFFECT'}$$



### 3. XGBoost Classification

#### Parameters of XGBoost


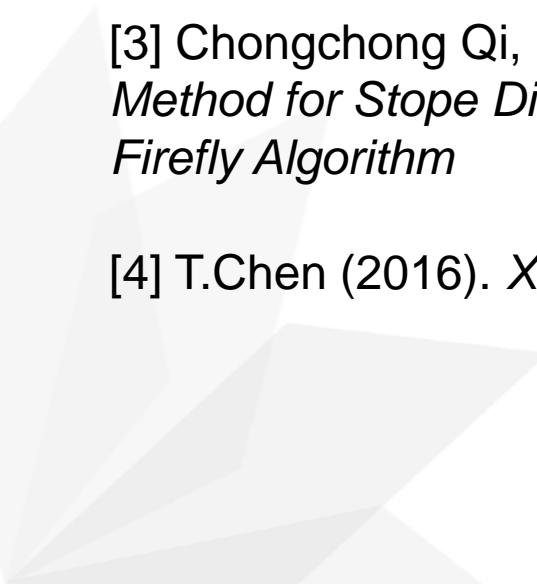
| Parameter        | Description   | Default Value |
|------------------|---|---------------|
| max_depth        | controls the maximum depth of each tree (used to control over-fitting)  | 6             |
| subsample        | specifies the fraction of observations to be randomly sampled at each tree (adds randomness)                            | 1             |
| eta              | the learning rate   | 0.3           |
| nrounds          | the number of trees to be produced (similar to ntree)   | 100–1000      |
| gamma            | controls the minimum loss reduction required to make a node split (used to control over-fitting)                        | 0             |
| min_child_weight | Specifies the minimum sum of instance weight of all the observations required in a child (used to control over-fitting) | 1             |
| colsample_bytree | Specifies the number of features to consider when searching for the best node split (adds randomness)                   | 1             |



# 3. Boosting with Python

- 1) Adaboost
- 2) Gradient Boosting
- 3) XGBoost

## Reference

- 
- [1] Josh Starmer(2019). *StatQuest with Josh Starmer*
  - [2] Seoul National University, Hyeong In Choi (2017). *Lecture 11: Boosting (I) AdaBoost (Draft: version 0.9.2)*
  - [3] Chongchong Qi, S.M.ASCE, Andy Fourie, Xu Zhao (2018). *Back-Analysis Method for Slope Displacements Using Gradient-Boosted Regression Tree and Firefly Algorithm*
  - [4] T.Chen (2016). *XGBoost : A Scalable Tree Boosting System*
- 



**Thank you**