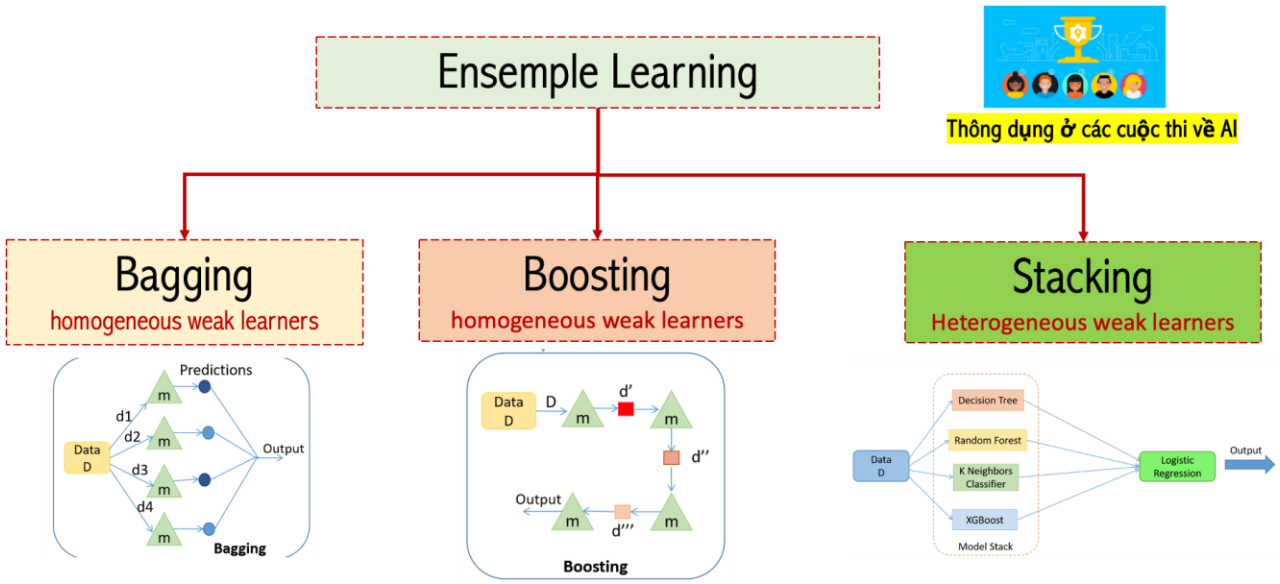
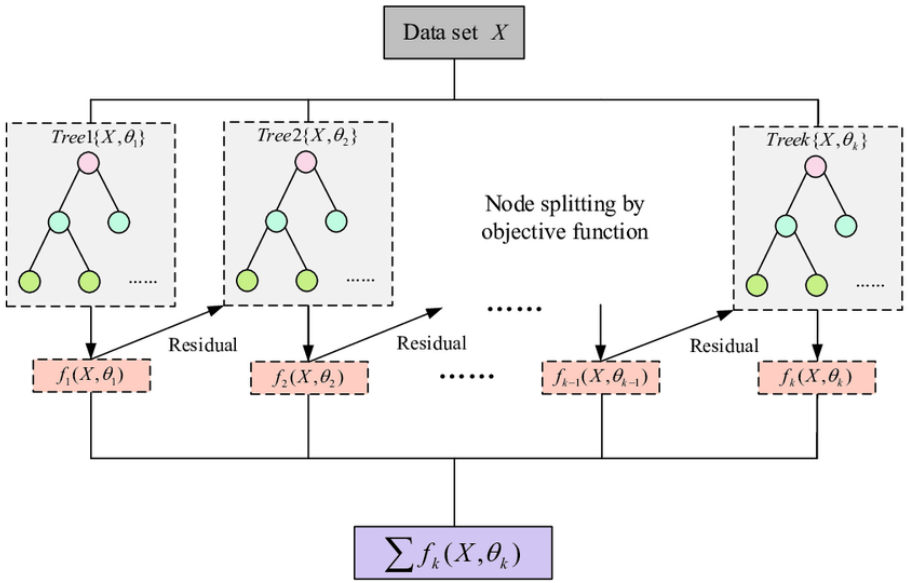
AI VIET NAM
@aivietnam.edu.vn

AdaBoost & Gradient Boost

(Basic, Advanced Concepts and Its Applications)





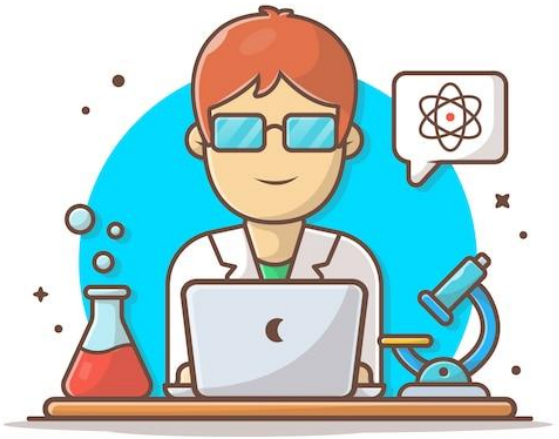
Vinh Dinh Nguyen
PhD in Computer Science

Outline

- Boosting Techniques
- AdaBoost Clearly Explain
- Gradient Boost Clearly Explain
- Time Series Data: Predicting Energy Consumption
- Summary

AI VIET NAM
@aivietnam.edu.vn

Decision Tree and Its Variance



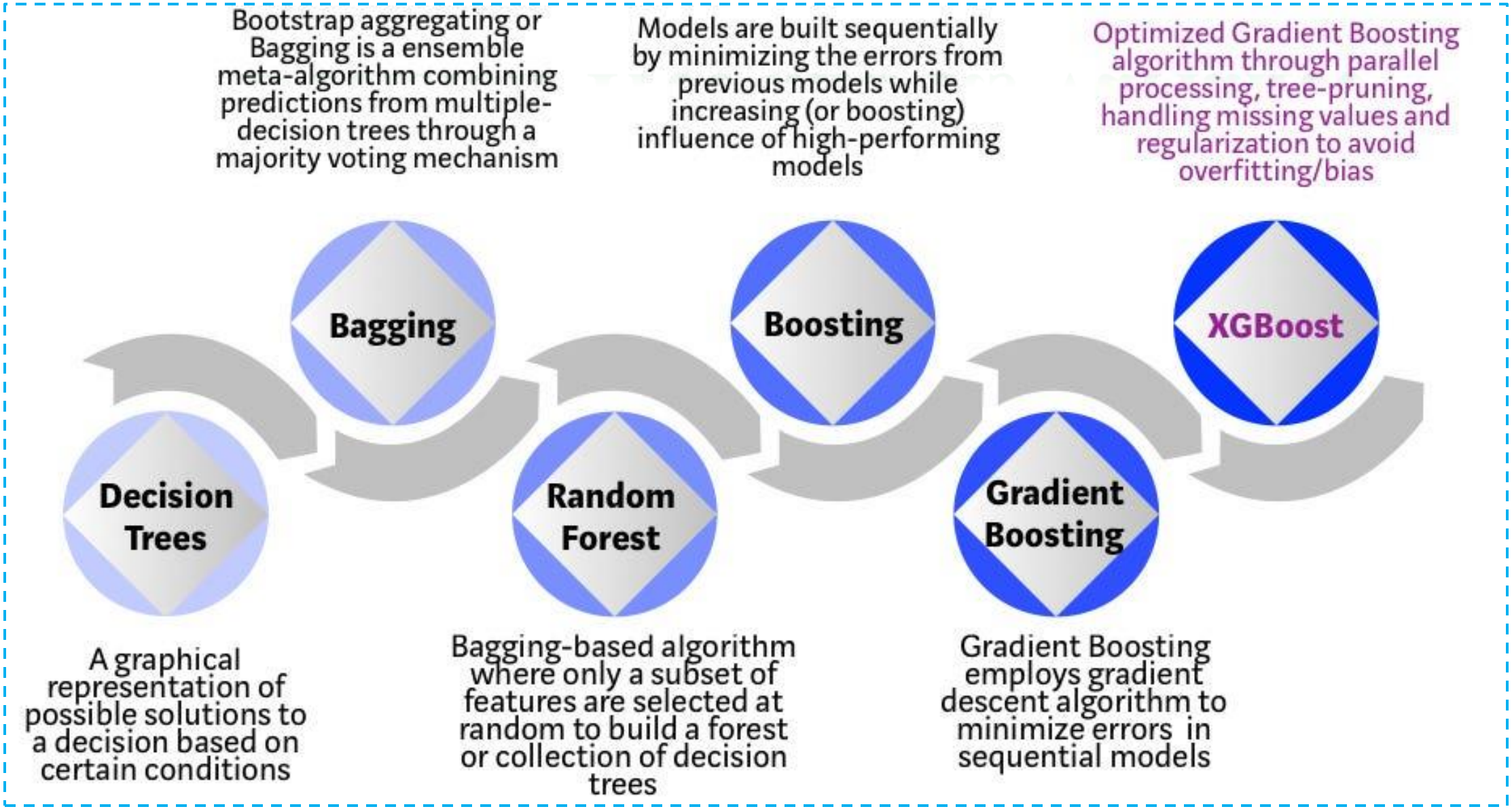
Unit(đơn vị)	Effect (hiệu quả) (%)
10	98
20	0
35	100
5	44
…	…

Tiêm 5 đơn vị vaccine

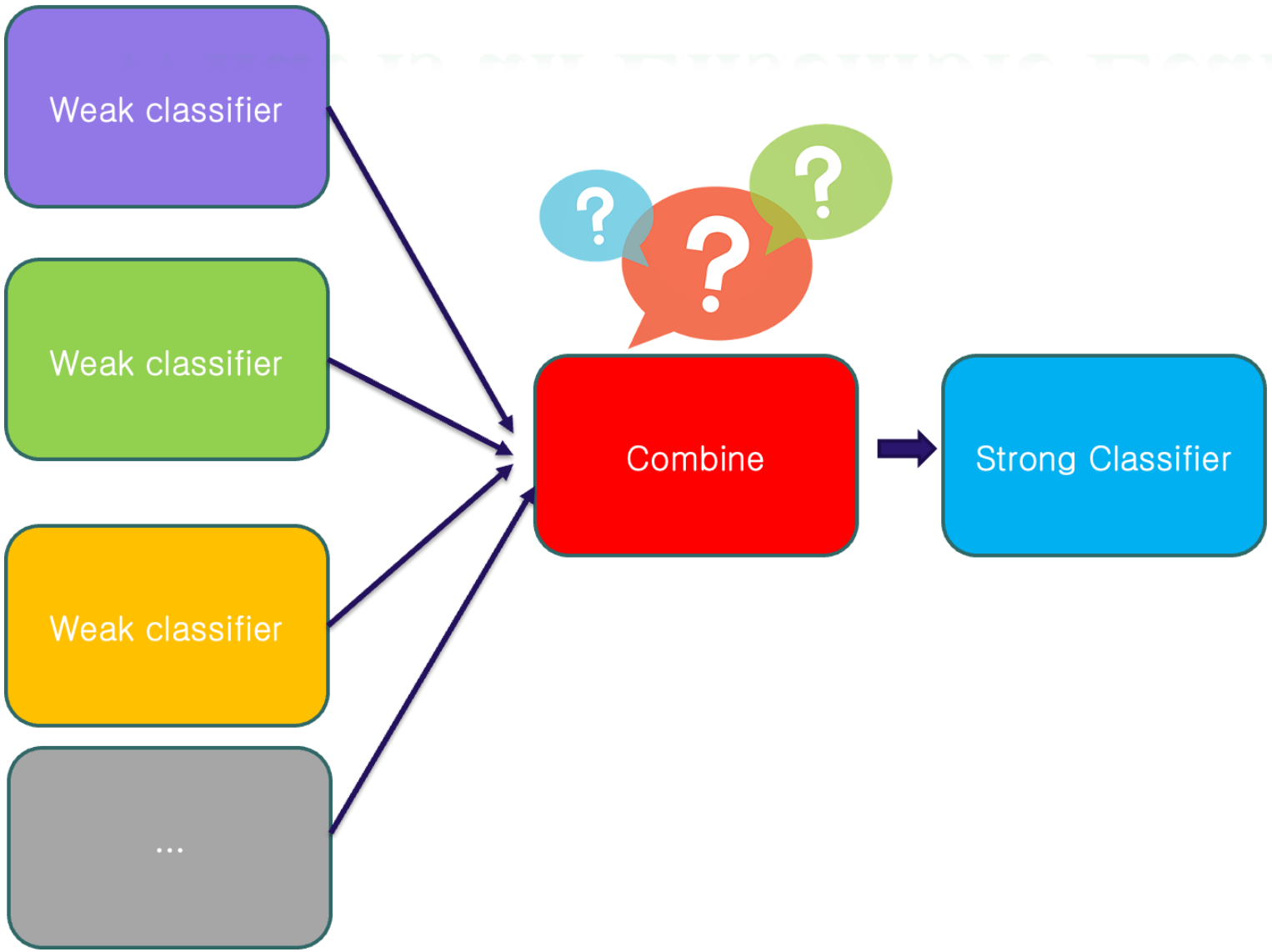
Hiệu quả vaccine: 44%

Khi có 1 vaccine ra đời, chúng ta muốn dự đoán xem nó hiệu quả bao nhiêu % ứng với từng liều lượng dùng trên bệnh nhân.

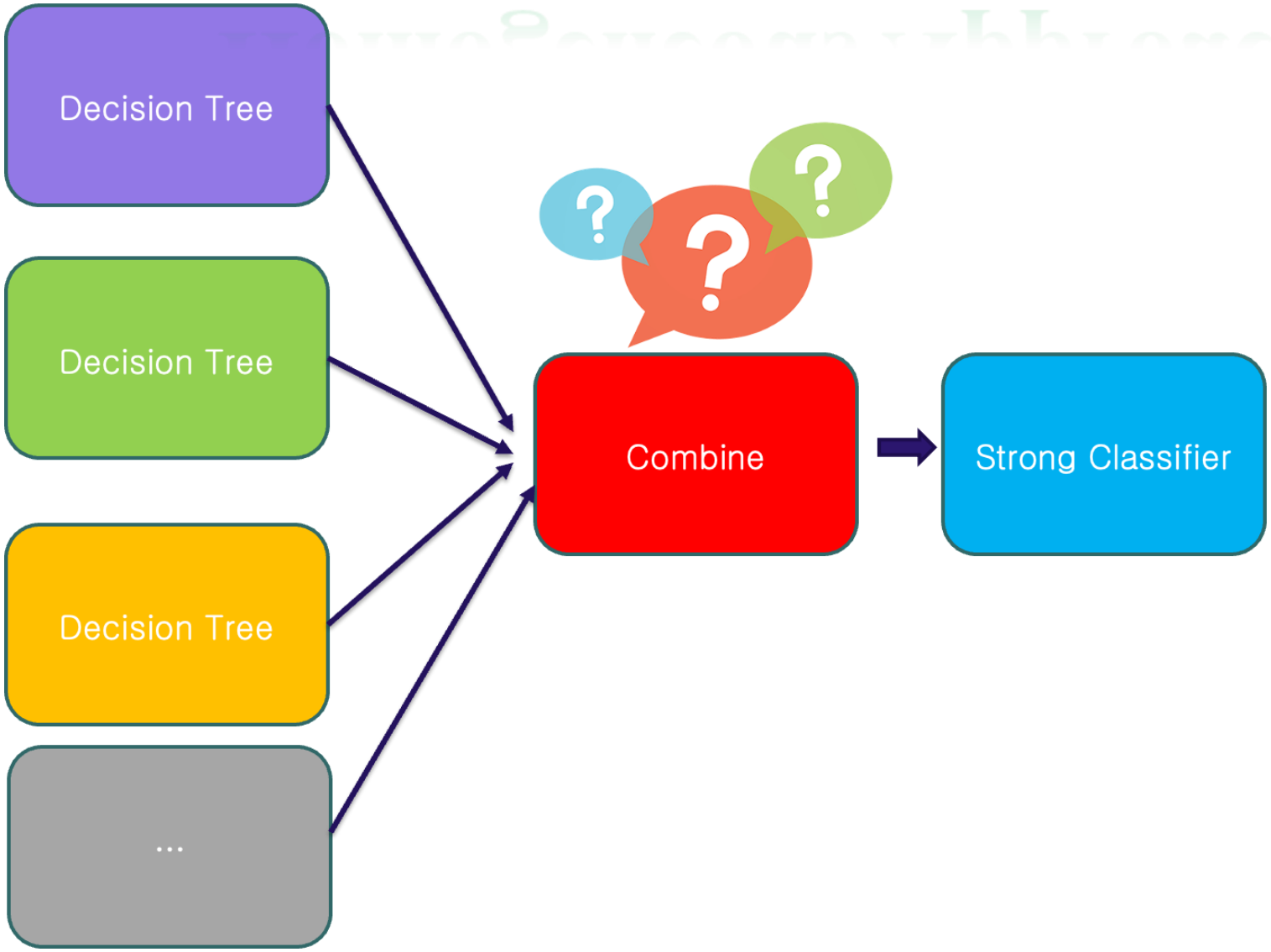
Decision Tree and Its Variance



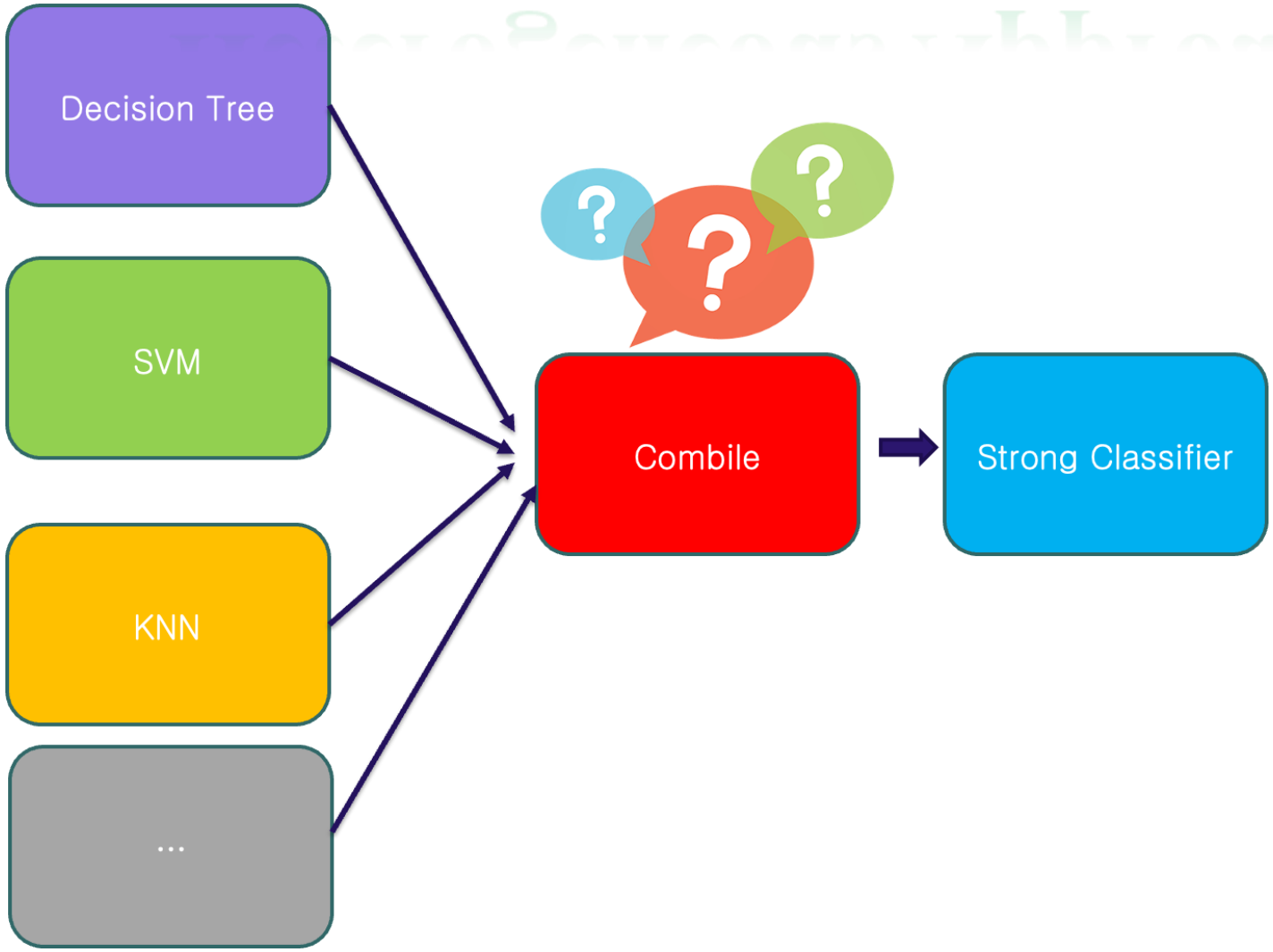
What is an Ensemble Learning?



Homogeneous Approach



Heterogeneous Approach



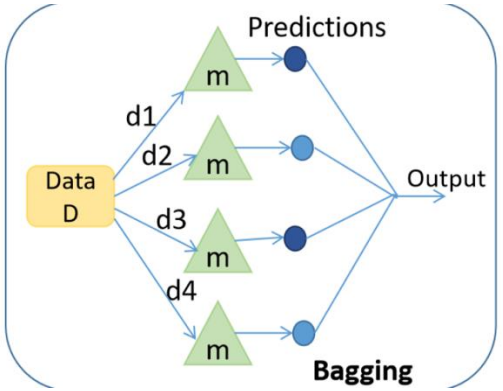
AI VIET NAM
@aivietnam.edu.vn

Ensemple Learning Techniques

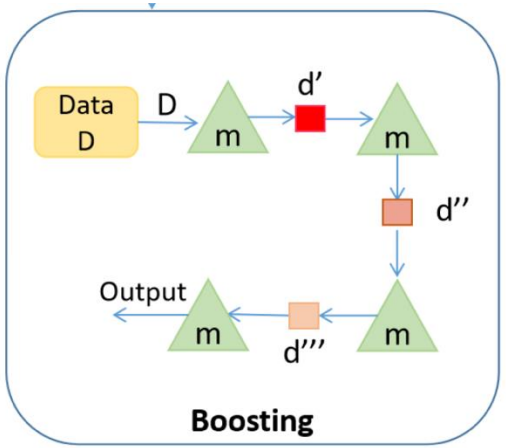
Ensemple Learning

Thông dụng ở các cuộc thi về AI

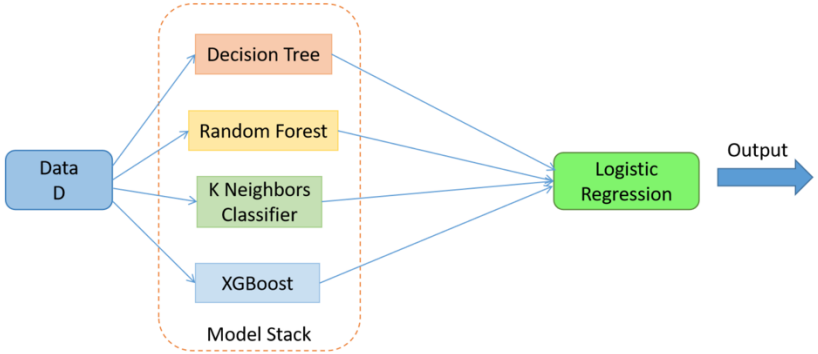
Bagging
homogeneous weak learners



Boosting
homogeneous weak learners



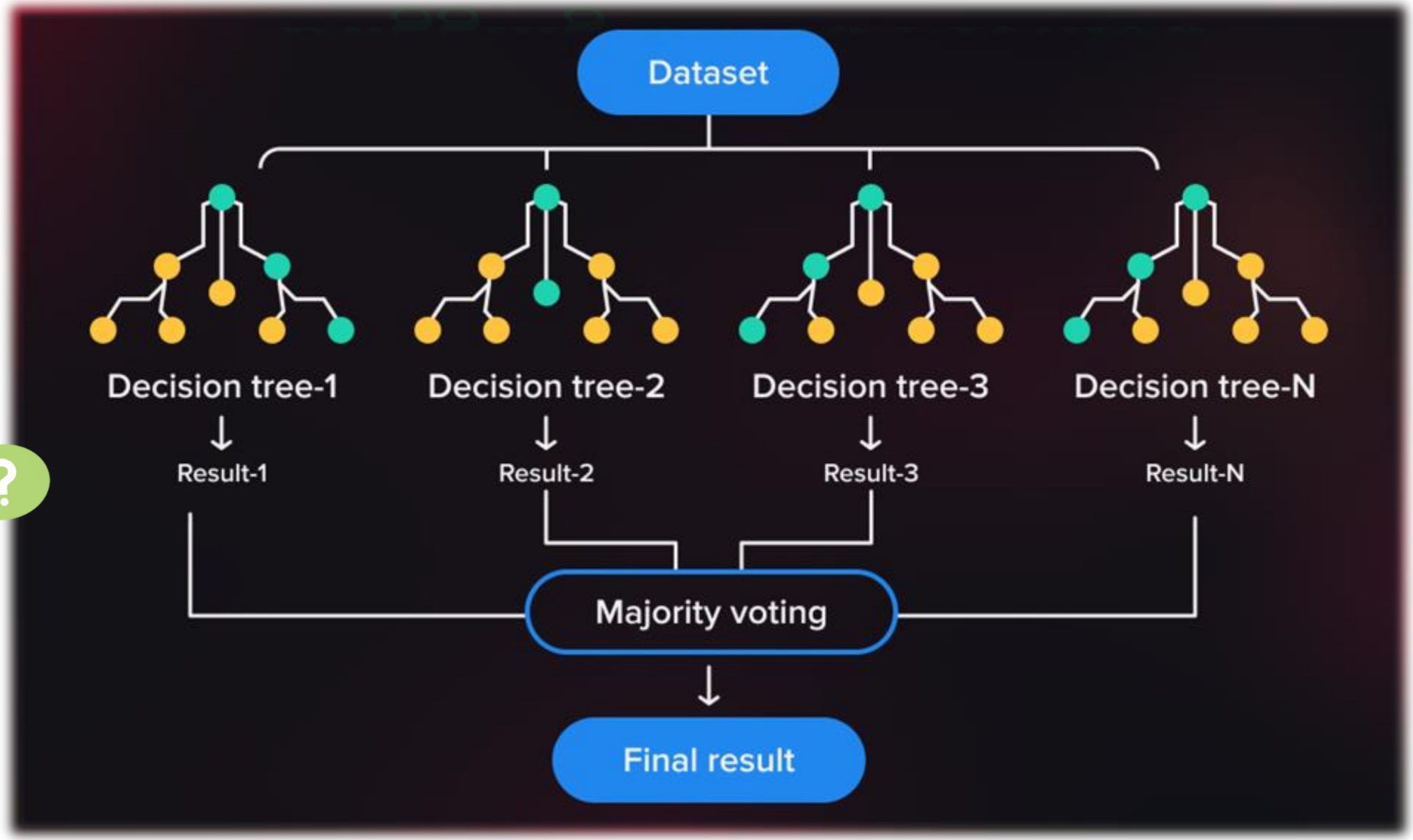
Stacking
Heterogeneous weak learners



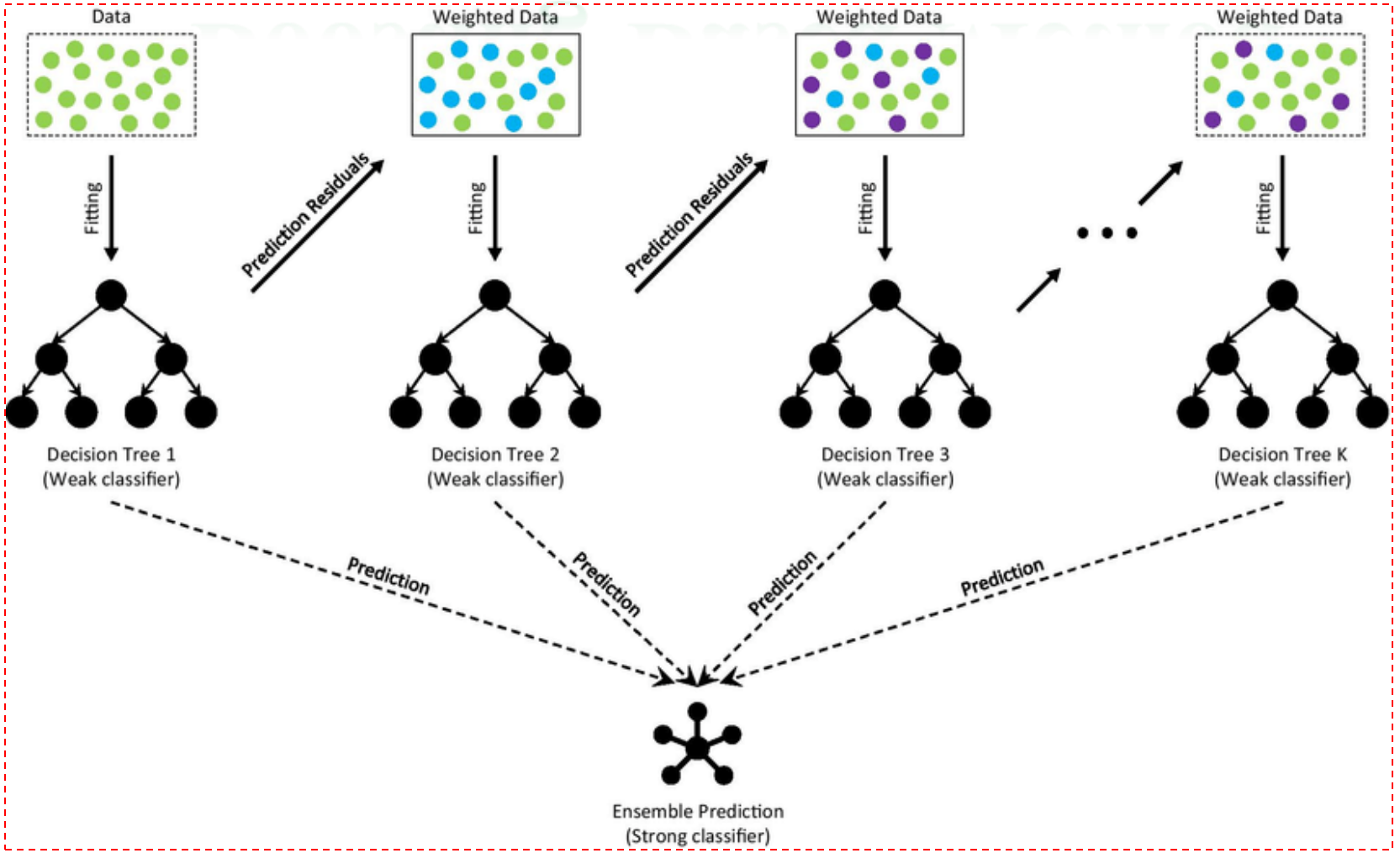
Bagging-based Method

Random Forest

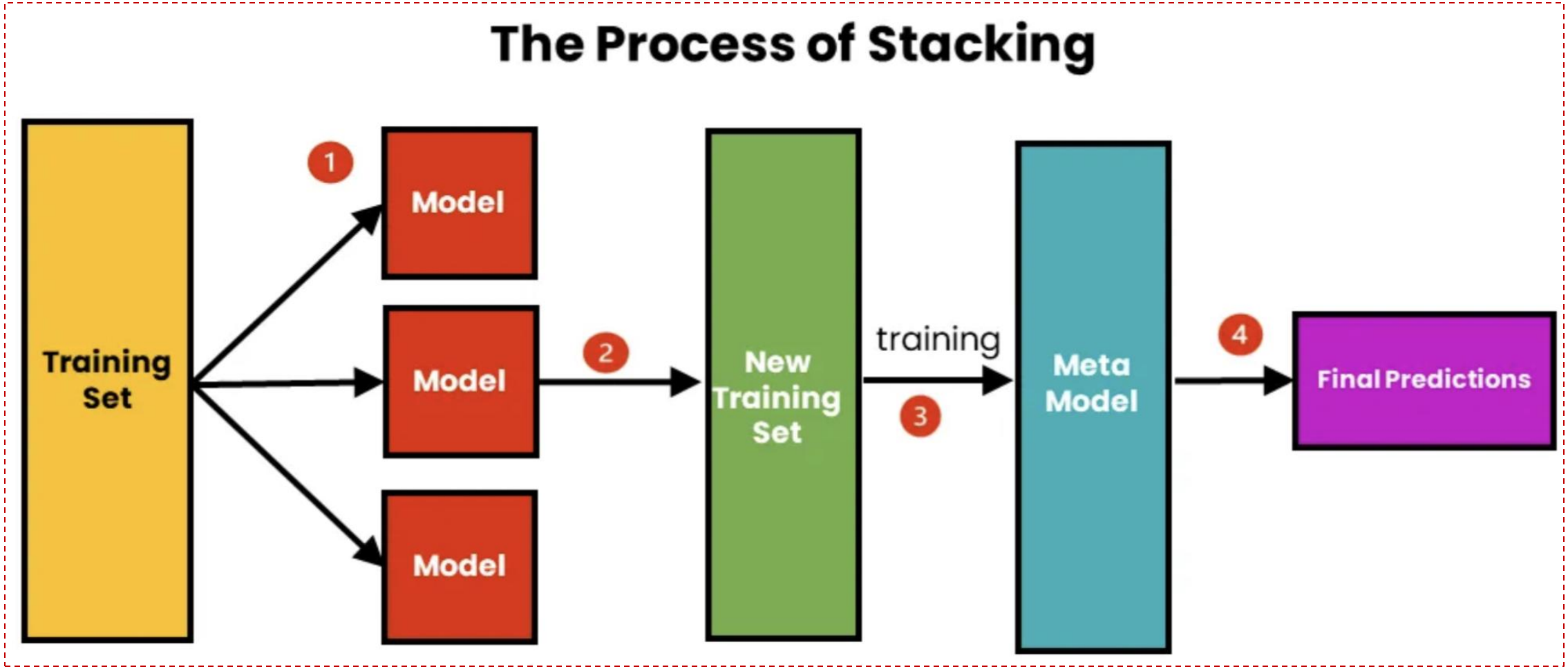
Last Week



Boosting-Based Method

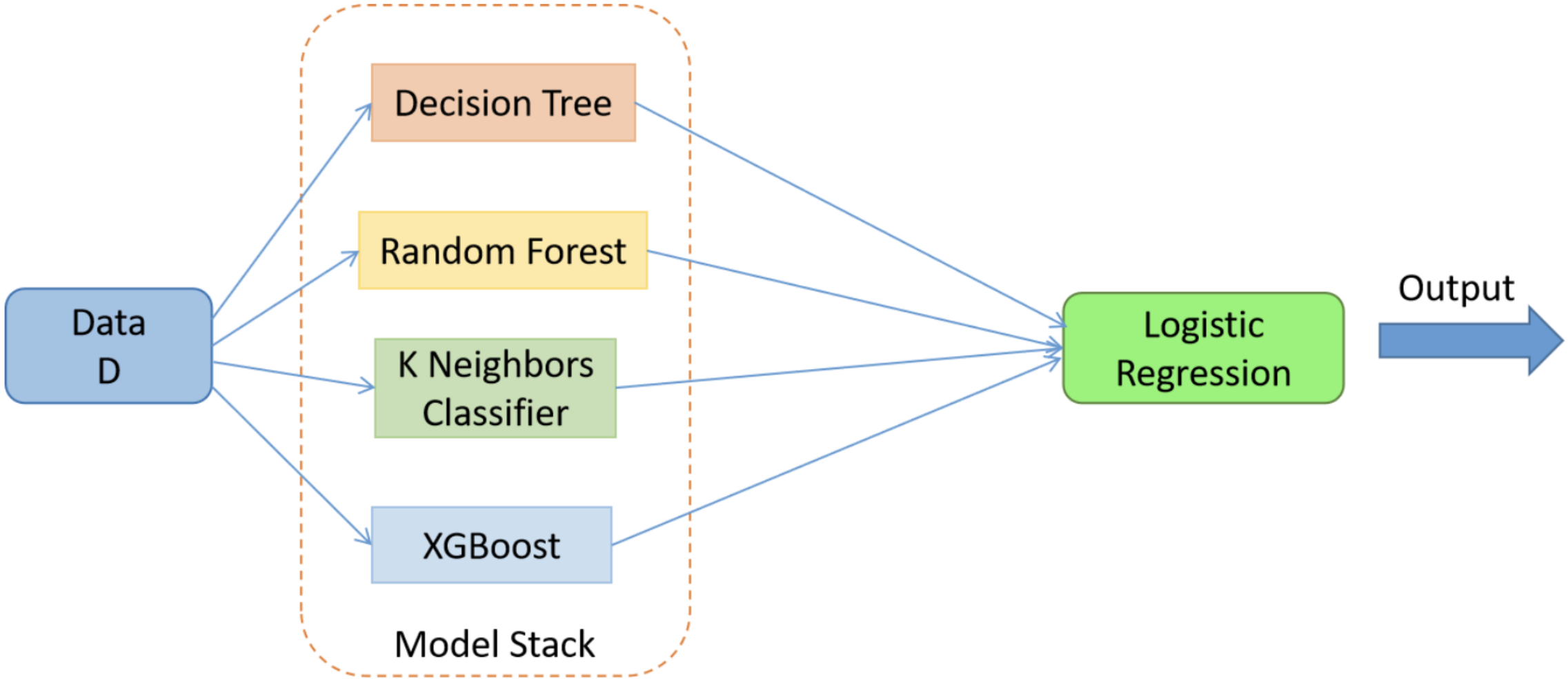


Stacking-Based Method



AI VIET NAM
@aivietnam.edu.vn

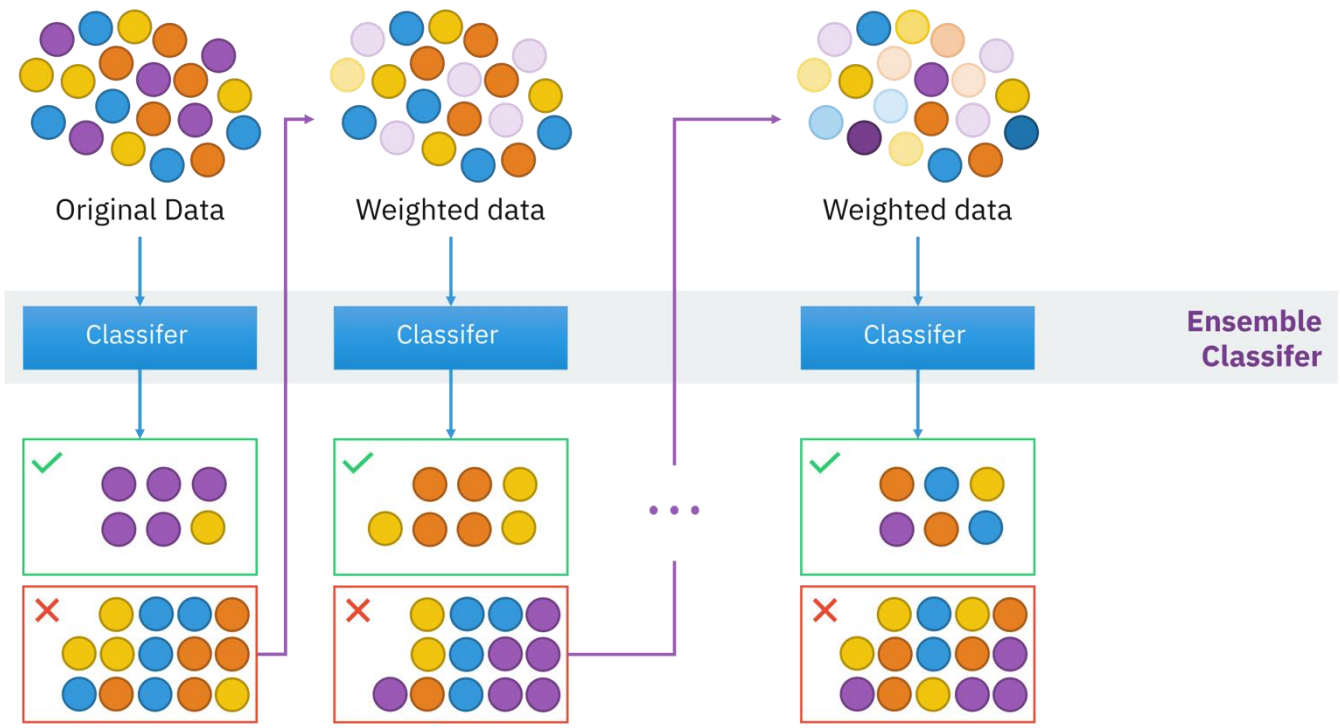
Stacking-Based Method



AI VIET NAM
@aivietnam.edu.vn

Boosting Technique

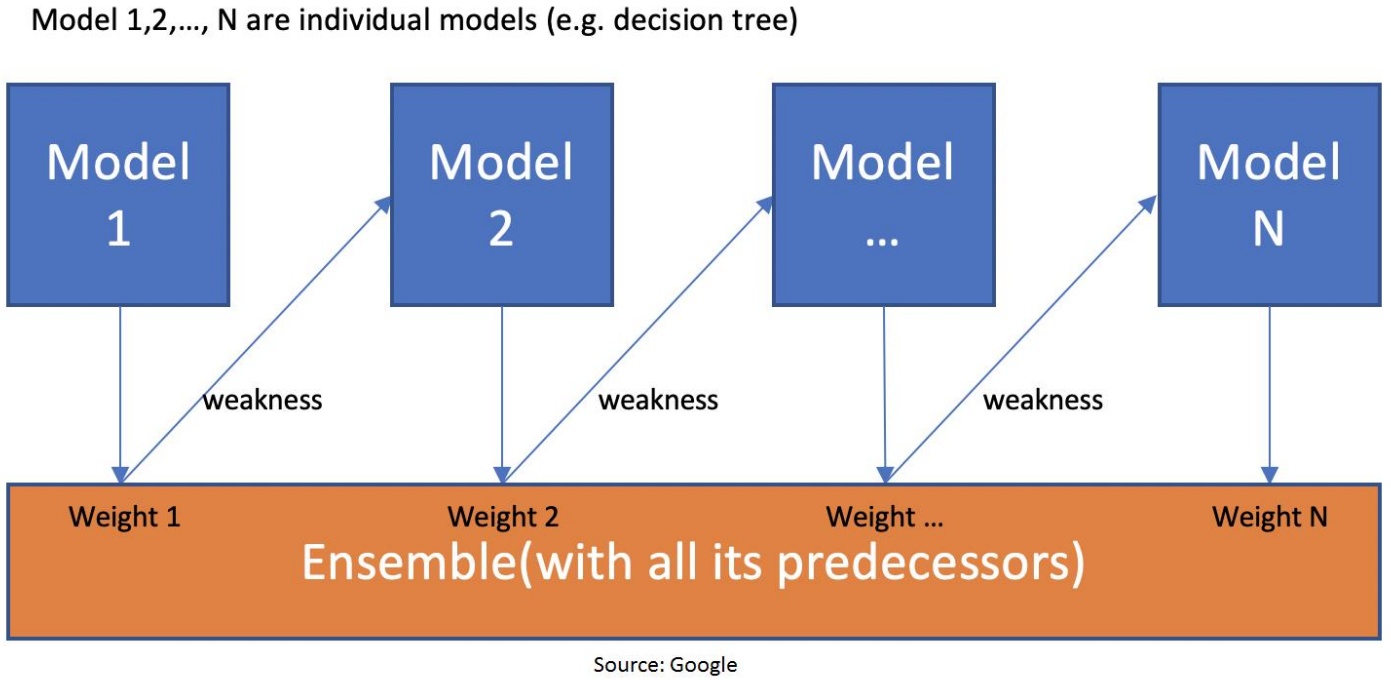
Boosting is an ensemble modelling, technique that attempts to build a strong classifier from the number of weak classifiers



https://en.wikipedia.org/wiki/Boosting_%28machine_learning%29#/media/File:Ensemble_Boosting.svg

Boosting Technique

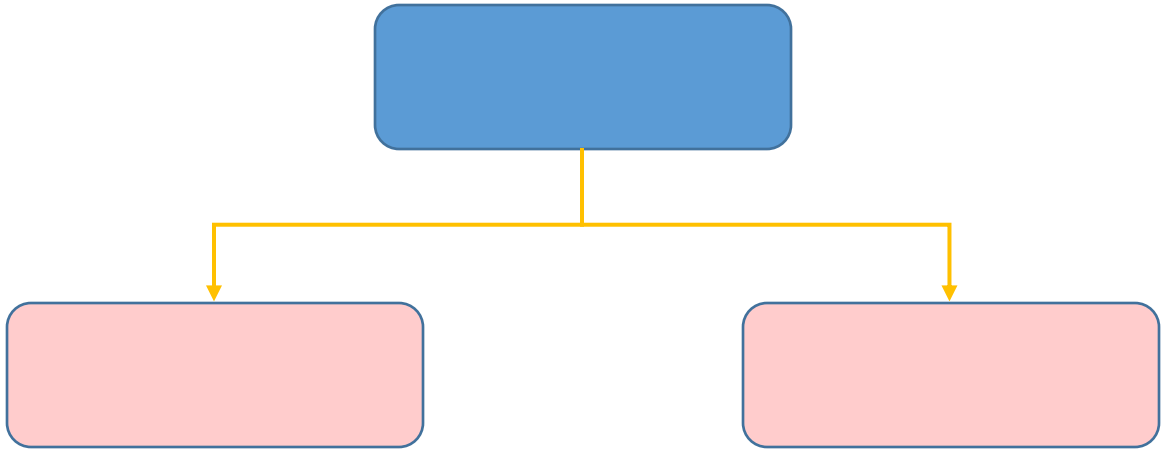
Boosting is an ensemble modelling, technique that attempts to build a strong classifier from the number of weak classifiers



Source: Google

https://en.wikipedia.org/wiki/Boosting_%28machine_learning%29#/media/File:Ensemble_Boosting.svg

Stump Definition

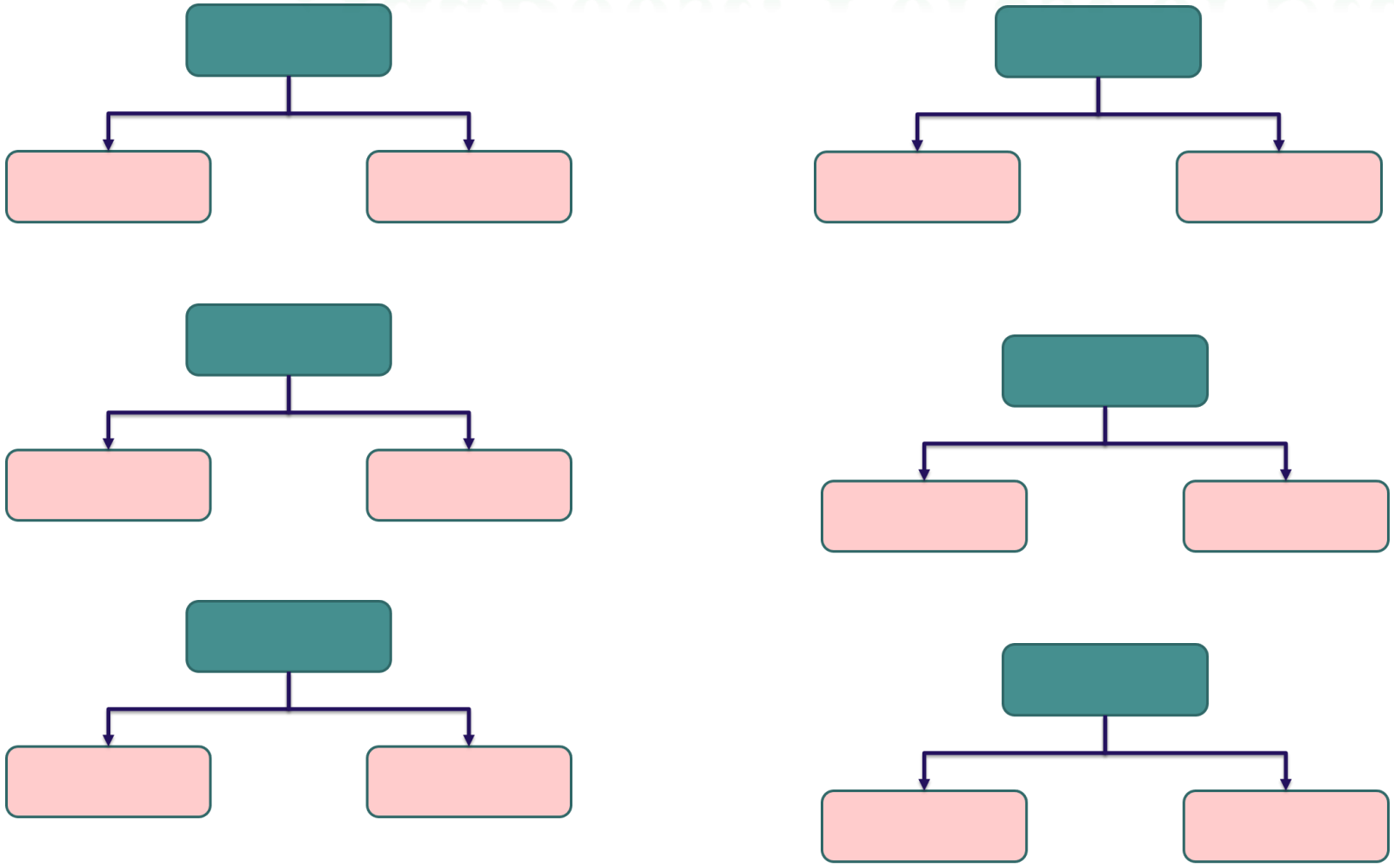


a node with two leaves and this is known as Stump

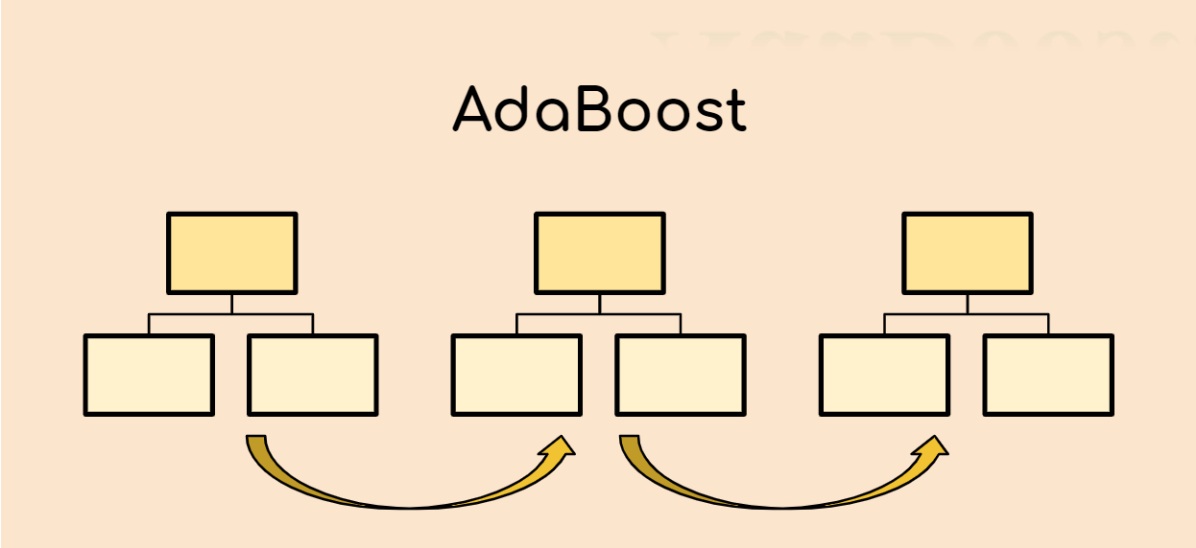
Outline

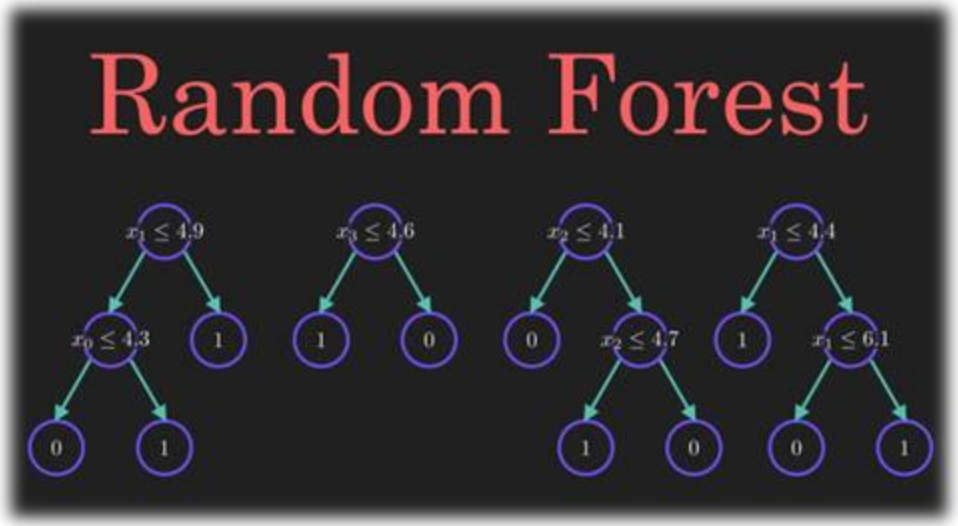
- Boosting Techniques
- AdaBoost Clearly Explain
- Gradient Boost Clearly Explain
- Time Series Data: Predicting Energy Consumption
- Summary

AdaBoost: Forest of Stump



AdaBoost: Forest of Stump



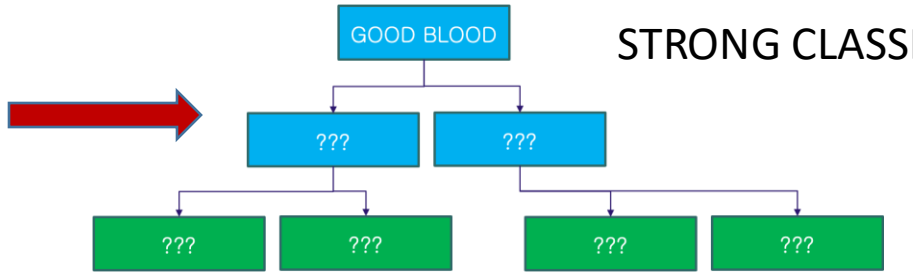


Sample Dataset

CHEST PAIN	GOOD BLOOD CIRCULATION	BLOCKED ARTERIES	WEIGHT	HEART DISEASE
NO	NO	NO	125	NO
YES	YES	YES	180	YES
YES	YES	NO	210	NO
YES	NO	YES	167	YES

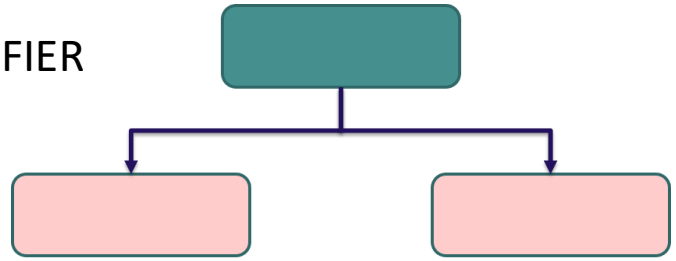
Sample Dataset

CHEST PAIN	GOOD BLOOD CIRCULATION	BLOCKED ARTERIES	WEIGHT	HEART DISEASE
NO	NO	NO	125	NO
YES	YES	YES	180	YES
YES	YES	NO	210	NO
YES	NO	YES	167	YES



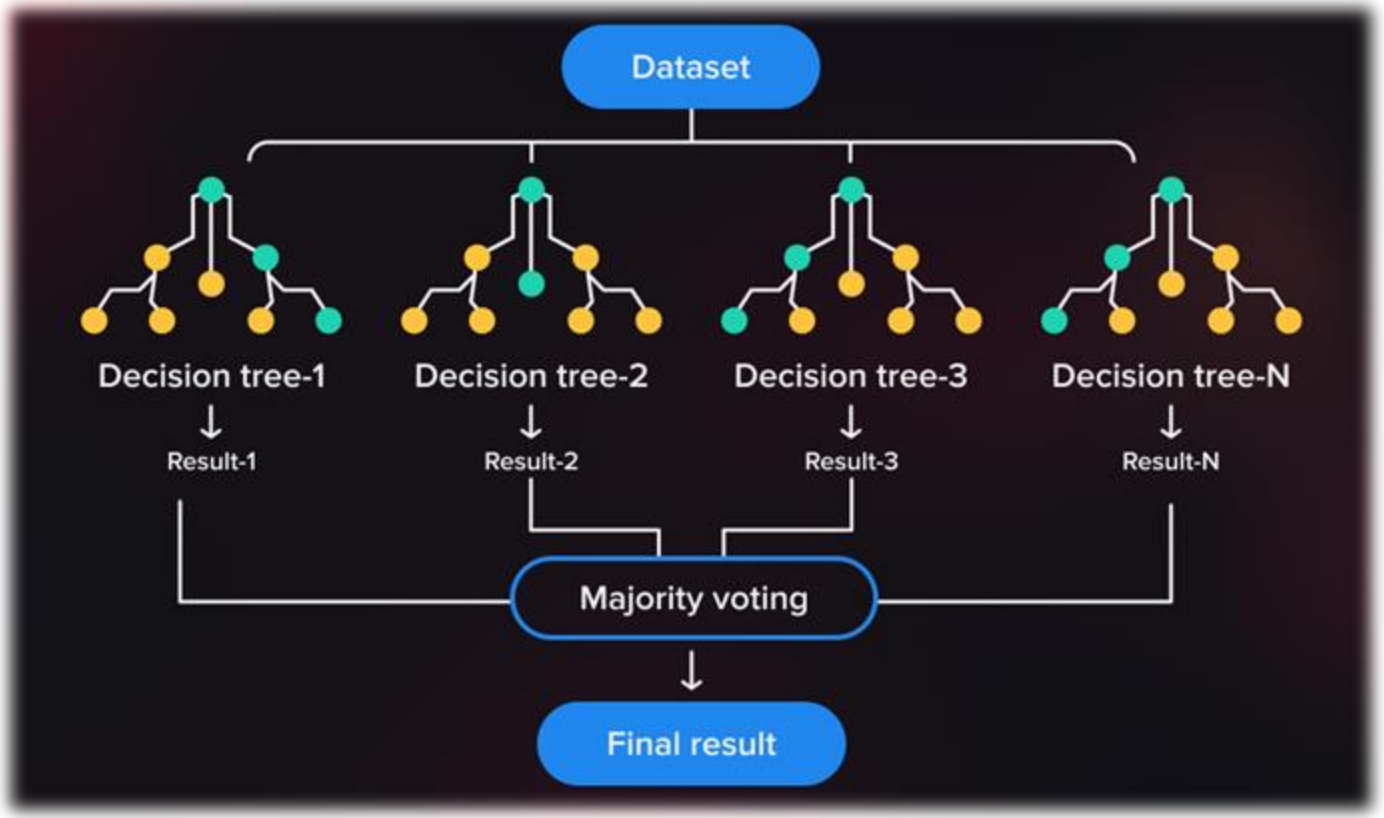
STRONG CLASSIFIER

WEAK CLASSIFIER



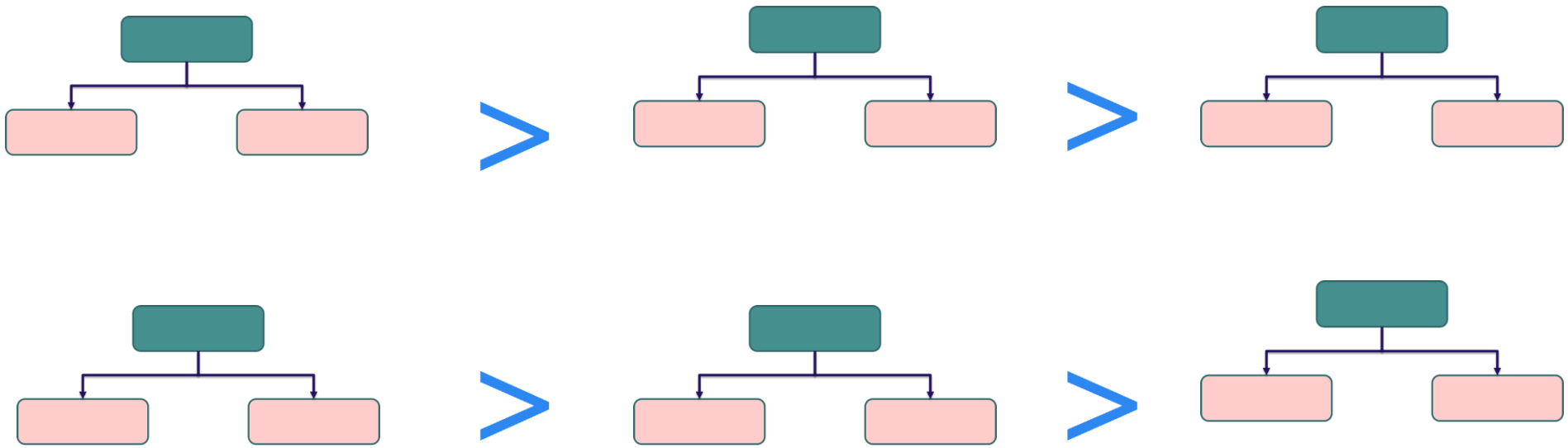
Random Forest

- Each tree in the random forest has equal votes(weights) on the final decision



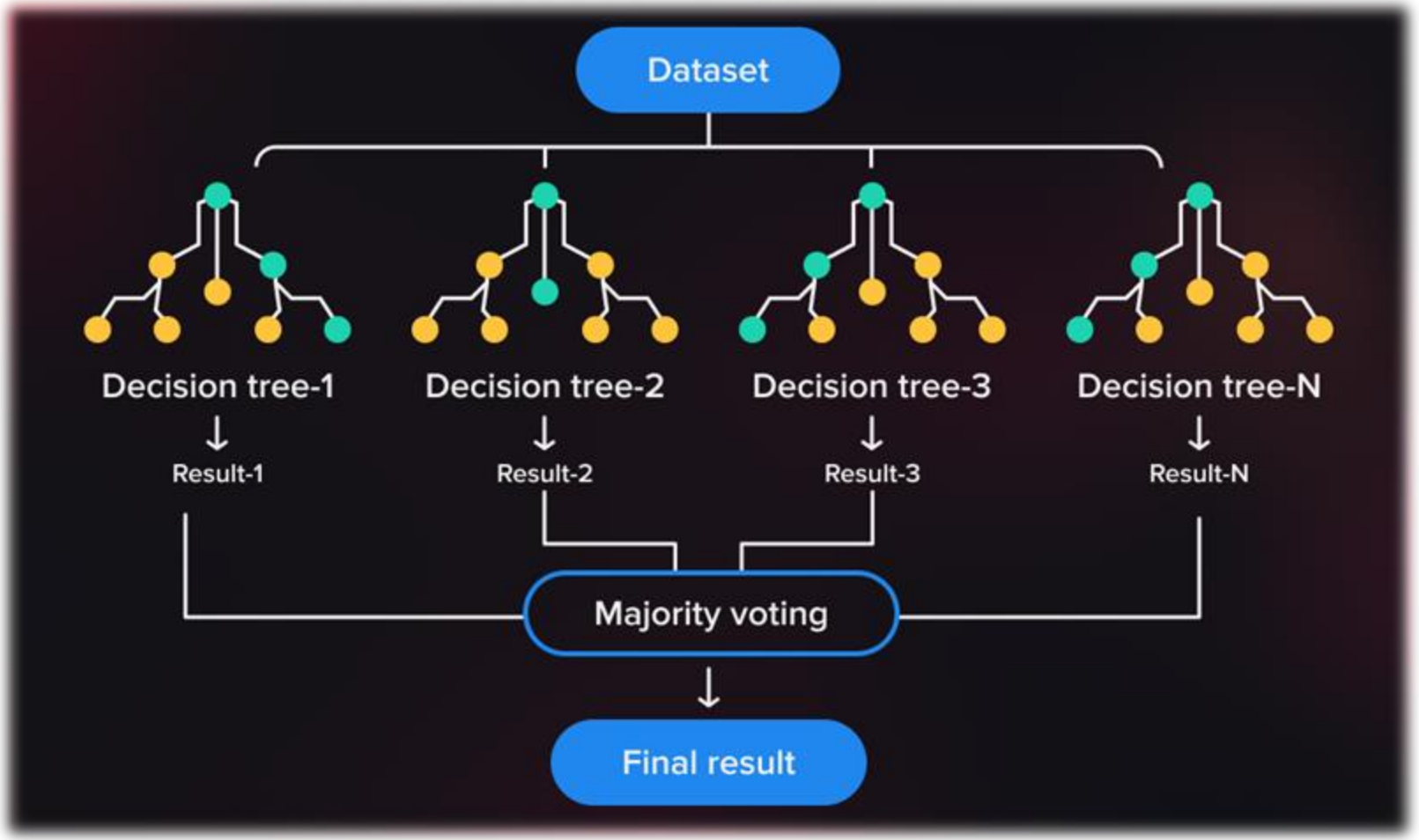
AdaBoost: FOREST OF STUMP

- Stump are not equally weighted in the final decision.
- Stump that create more error will have less contribution in the final decision

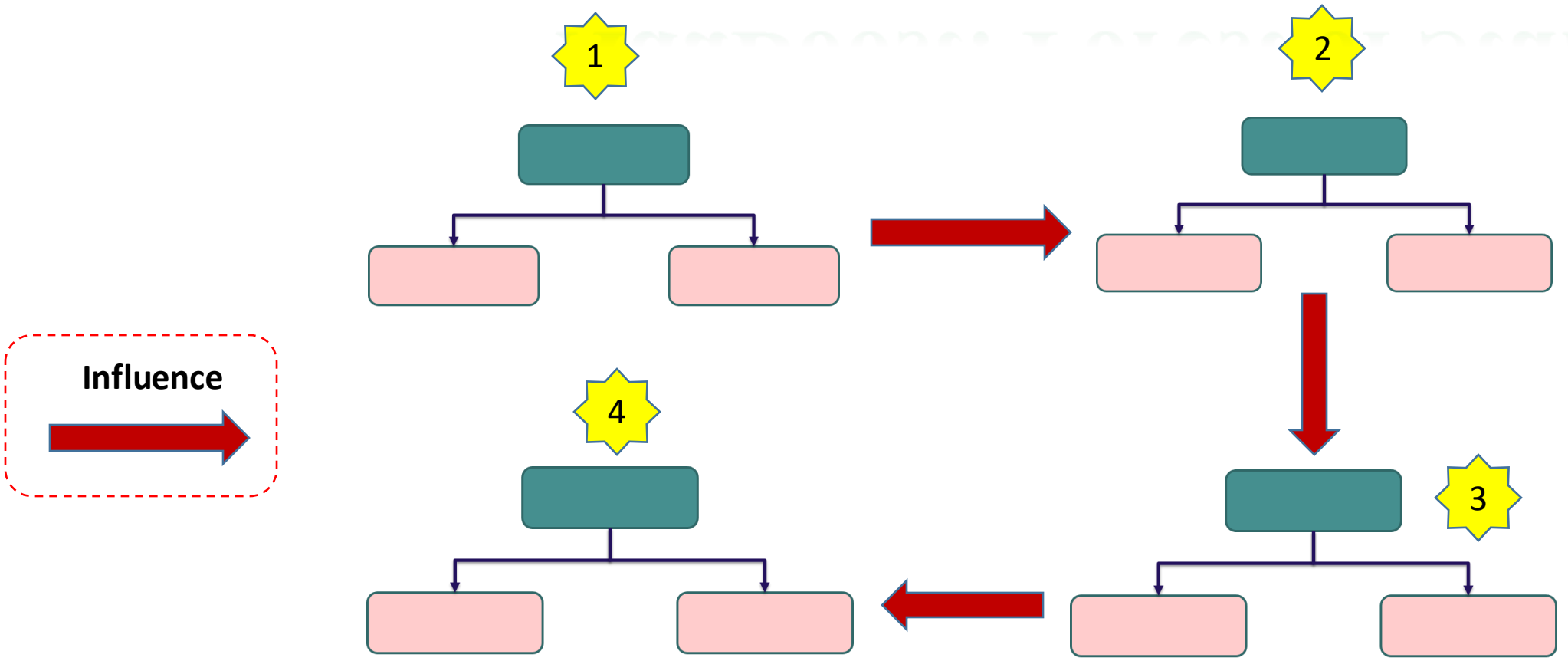


Random Forest

Tree are indepently created



AdaBoost: Forest of Stump



Differences Between RF and AdaBoost

1 ONE

Weak Learners is a ___
AdaBoost combines a lot of___

2

Stumps have various ___ to the final result

Each stump is created by considering ___

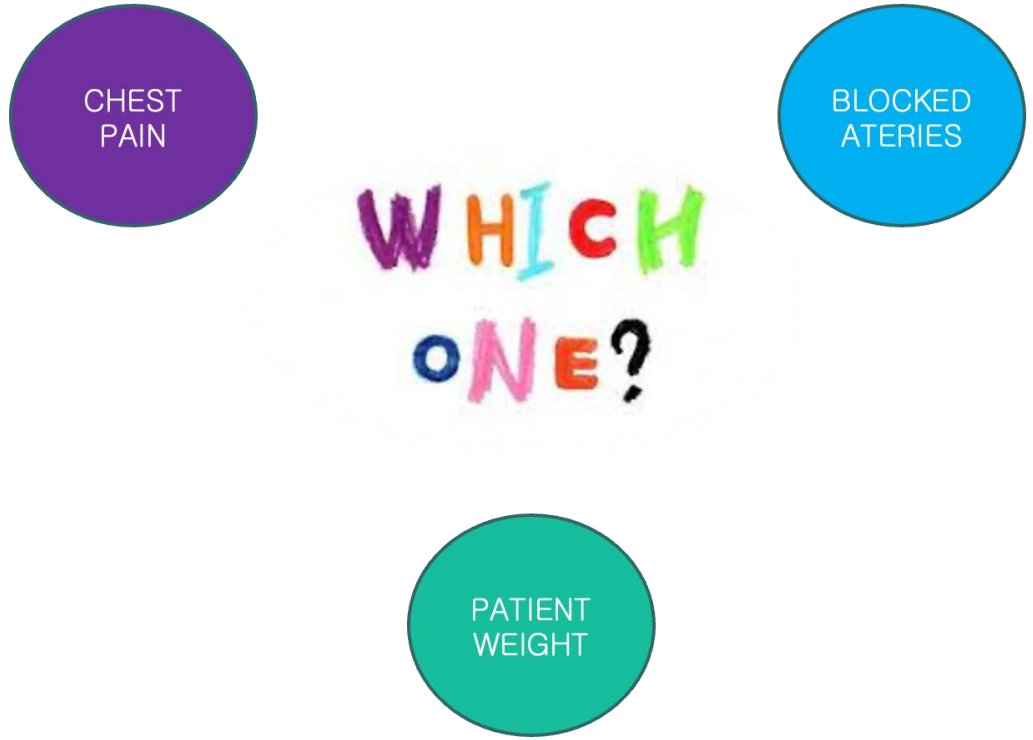
Heart Disease Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

Important of sample = Sample weight = 1 / number of samples = 1/8

1st Stump in the Tree

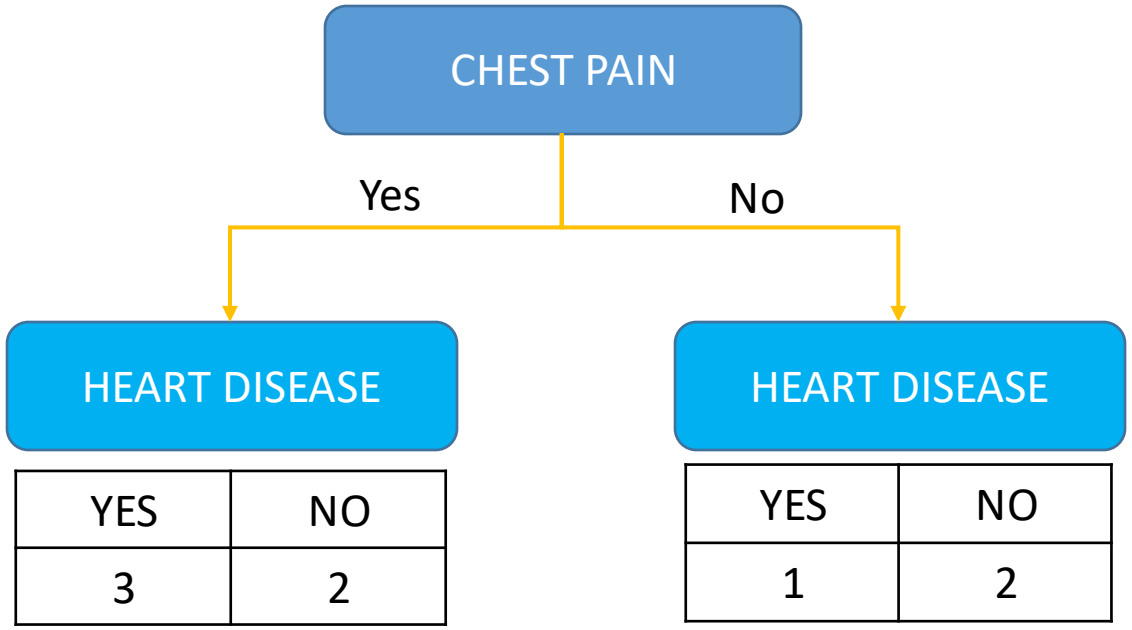
Chest Pain	Blocked Arteries	Patient Weight	Heart Diease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No



Compute Gini Index For Chest Pain

Chest Pain	Heart Diease	Sample Weight
Yes	Yes	1/8
No	Yes	1/8
Yes	Yes	1/8
Yes	Yes	1/8
No	No	1/8
No	No	1/8
Yes	No	1/8
Yes	No	1/8

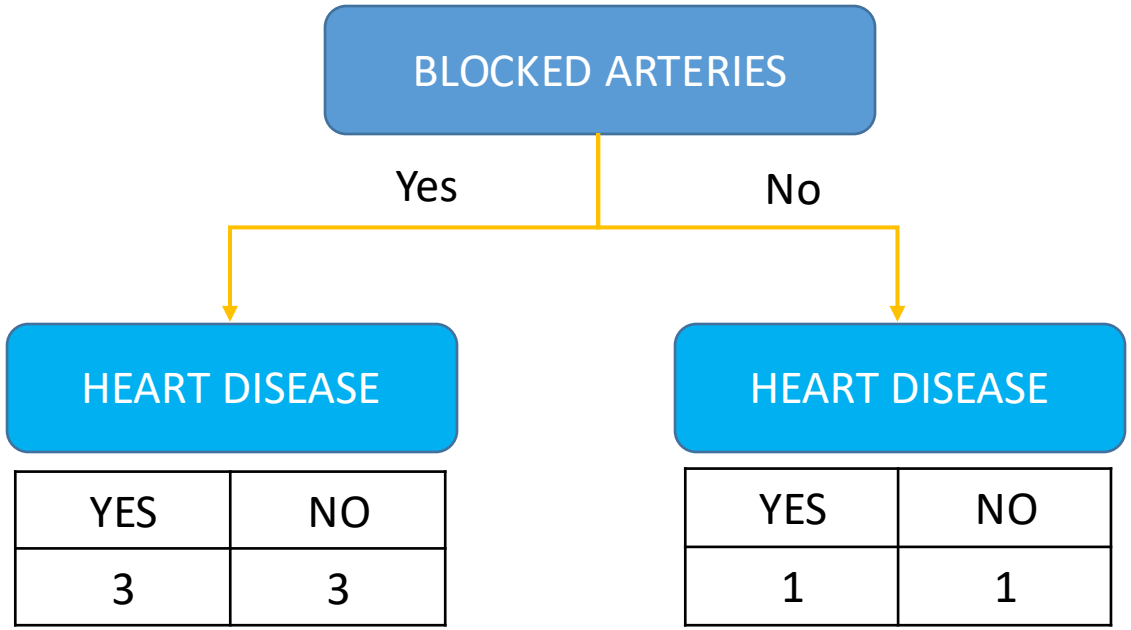
Gini index = 5/8 * (1 – (3/5)^2 – (2/5)^2) + 3/8* (1 - (1/3)^2 - (2/3)^2) =
0.57



Gini Index for Blocked Arteries

Blocked Arteries	Heart Diease	Sample Weight
Yes	Yes	1/8
Yes	Yes	1/8
No	Yes	1/8
Yes	Yes	1/8
Yes	No	1/8
Yes	No	1/8
No	No	1/8
Yes	No	1/8

Gini index = 6/8 * (1 – (3/6)^2 – (3/6)^2) + 2/8* (1 - (1/2)^2 - (1/2)^2) = **0.5**



YES	NO
3	3

YES	NO
1	1

Gini Index for Heart Disease

Patient Weight	Heart Diease	Sample Weight
205	Yes	1/8
180	Yes	1/8
210	Yes	1/8
167	Yes	1/8
156	No	1/8
125	No	1/8
168	No	1/8
172	No	1/8

Gini index = =4/8 * (1-(1/4)^2 - (3/4)^2) + 4/8* (1-(1/4)^2 -(3/4)^2) = 0.375

192.5

195

188.5

161.5

140.5

146.5

170

PATIENT WEIGHT > 170

HEART DISEASE

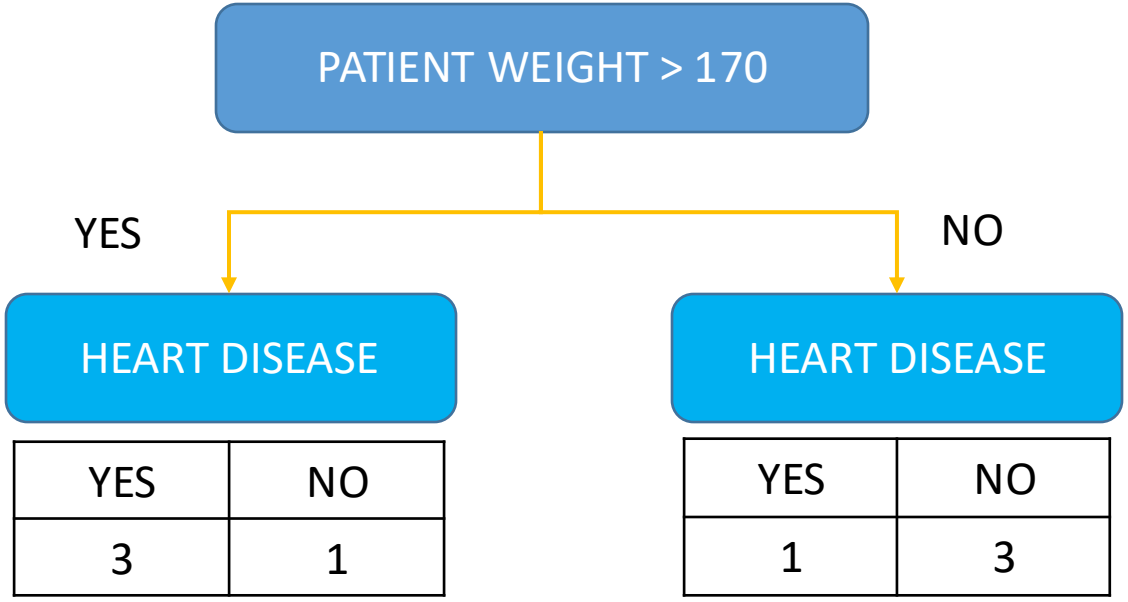
YES	NO
3	1

HEART DISEASE

YES	NO
1	3

Amount of Say

How was this stump contribute to the final decision (classification)?



YES	NO
3	1

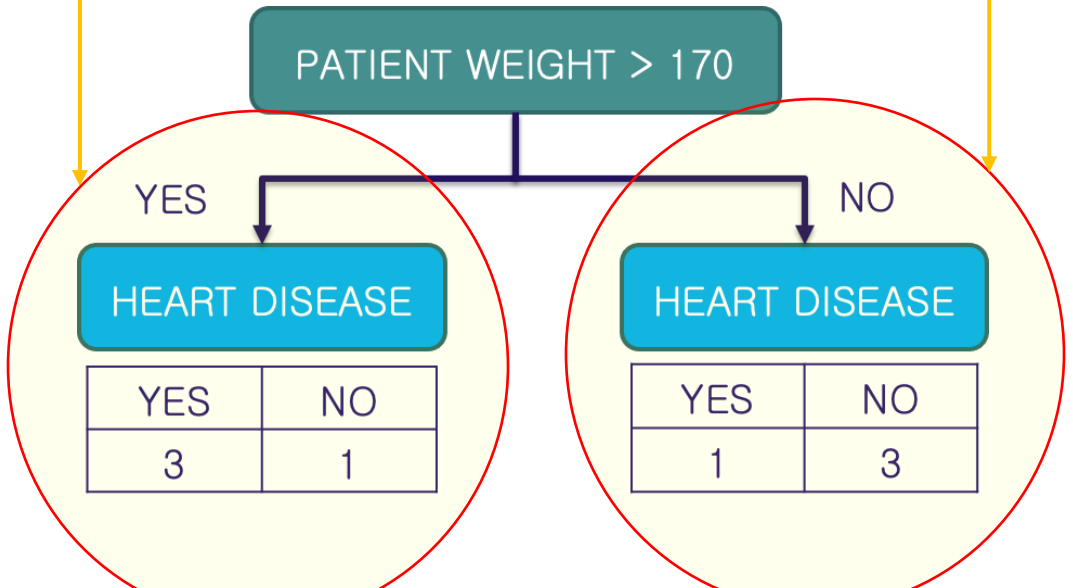
YES	NO
1	3

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

Amount of Say: Patient Weight

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	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

- Total error is equal to the sum of the weights of the incorrect classified
- Amount of say = 1/2*log((1-2/8) / (2/8)) = 0.55



Amount of Say: Weight of The Tree

Probability Vs. Odds

Your friend went fishing 10 times a month

- Caught a fish 4 times
- Failed to catch 6 times

What is the *probability* and *odds* of getting a Fish for lunch?

$$\text{Probability} = \frac{\text{Chance for catching fish}}{\text{Total chances}} = \frac{4}{10} = 0.4$$

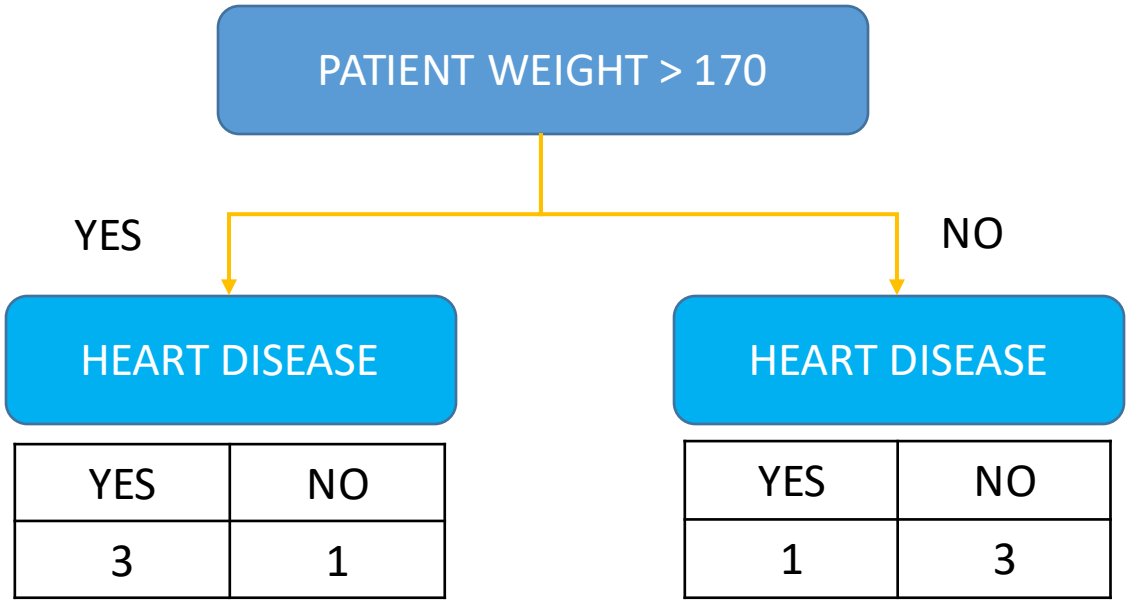
$$\text{Odds} = \frac{\text{Chance for catching fish}}{\text{Chance for not catching fish}} = \frac{4}{6} = 0.67$$

$$\text{Odds} = \frac{\text{Probability of catching fish}}{\text{Probability of not catching fish}} = \frac{4/10}{6/10} = 0.67$$

$$\text{Odds} = \frac{1 - \text{Probability of not catching fish}}{\text{Probability of not catching fish}} = \frac{4/10}{6/10} = 0.67$$

AI VIET NAM
@aivietnam.edu.vn

Amount of Say: Weight of The Tree



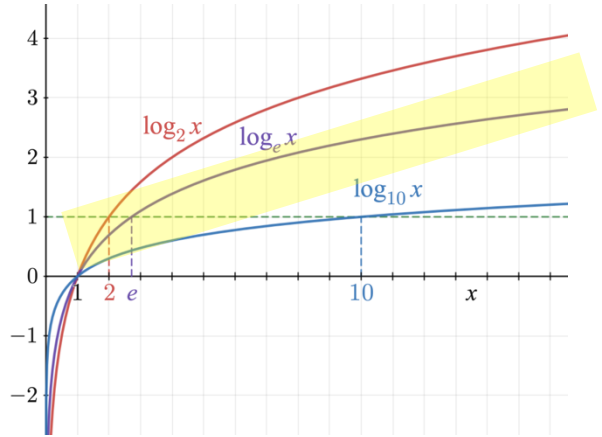
YES	NO
3	1

YES	NO
1	3

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

$$\text{Odds} = \frac{1 - \text{Probability of not catching fish}}{\text{Probability of not catching fish}} = \frac{4/10}{6/10} = 0.67$$

$$\text{Odds} = \frac{1 - \text{Probability of incorrect prediction}}{\text{Probability of incorrect prediction}}$$

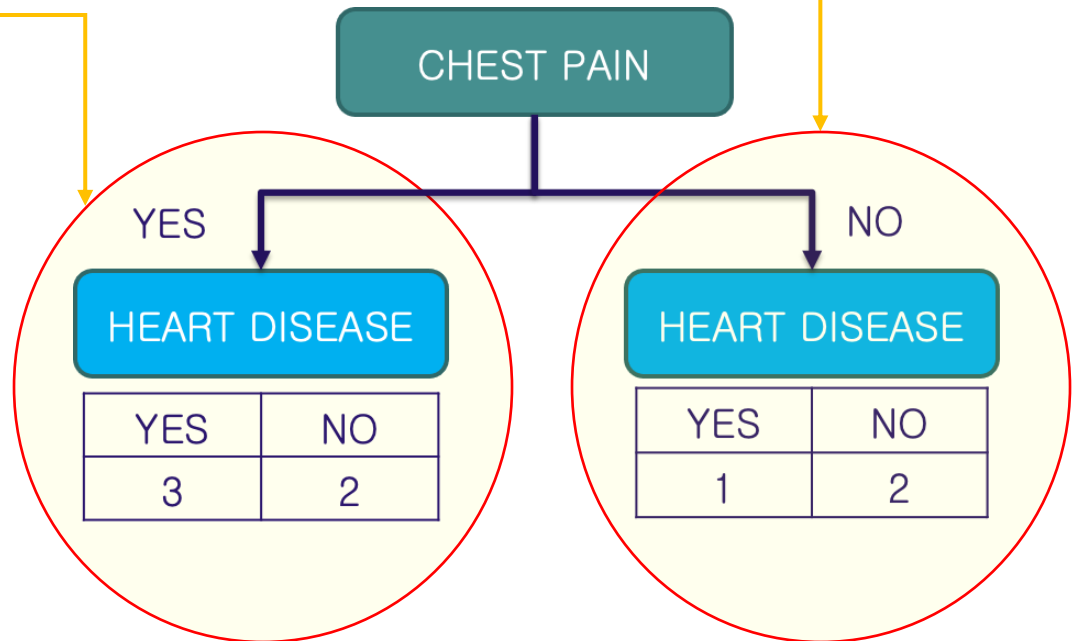


Amount of says = Odds or Amount of says = ½ x log(Odds)

Why?

Amount of Say: Chest Pain

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	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

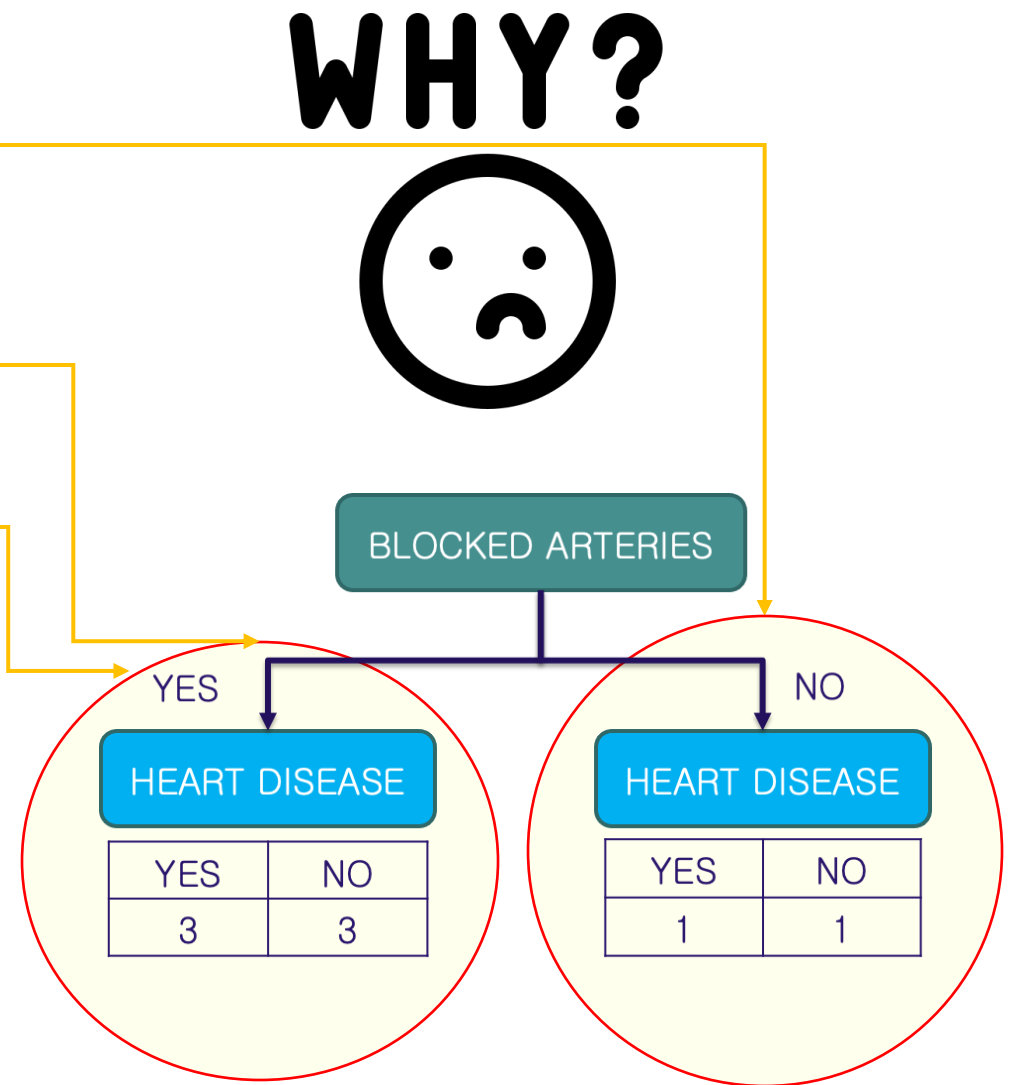


- Total error is equal to the sum of the weights of the incorrect classified
- ~~Amount of say = 1/2*log((1-3/8) / (3/8)) = 0.25~~

$$\log(\text{Odds}) = \log\left(\frac{1 - \text{Probability of incorrect prediction}}{\text{Probability of incorrect prediction}}\right)$$

Amount of Say: Blocked Arteries

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	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



- Total error is equal to the sum of the weights of the incorrect classified
- Amount of say = 1/2*log((1-4/8) / (4/8)) = 0

Assumptions

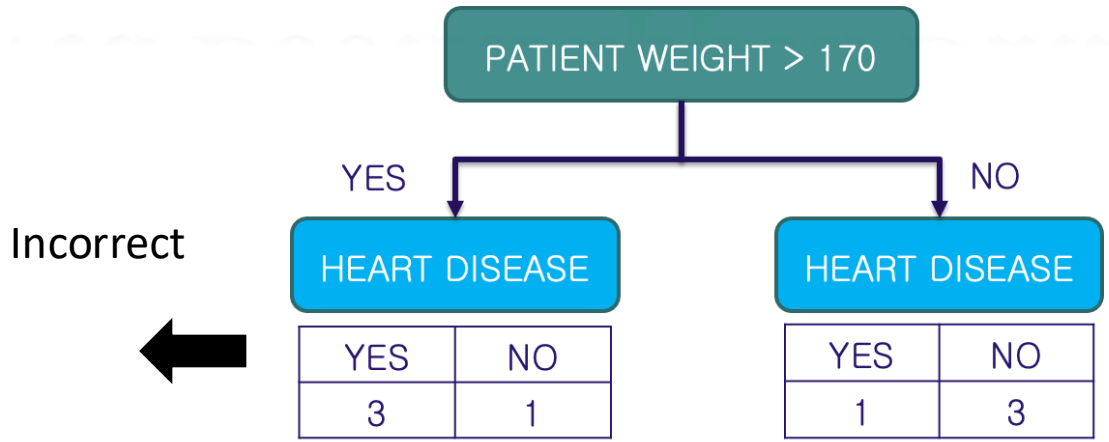
Known: weight cho các sample dự đoán sai được sử dụng để tính "Amount of Say" cho từng stump hiện tại.

Unknown: Tiếp theo, chúng ta cần làm thế nào để sử dụng thông tin các weight của sample dự đoán sai này để xây dựng stump tiếp và khắc phục các dự đoán sai này

AI VIET NAM
@aivietnam.edu.vn

Idea: Improved Bootstrapped Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

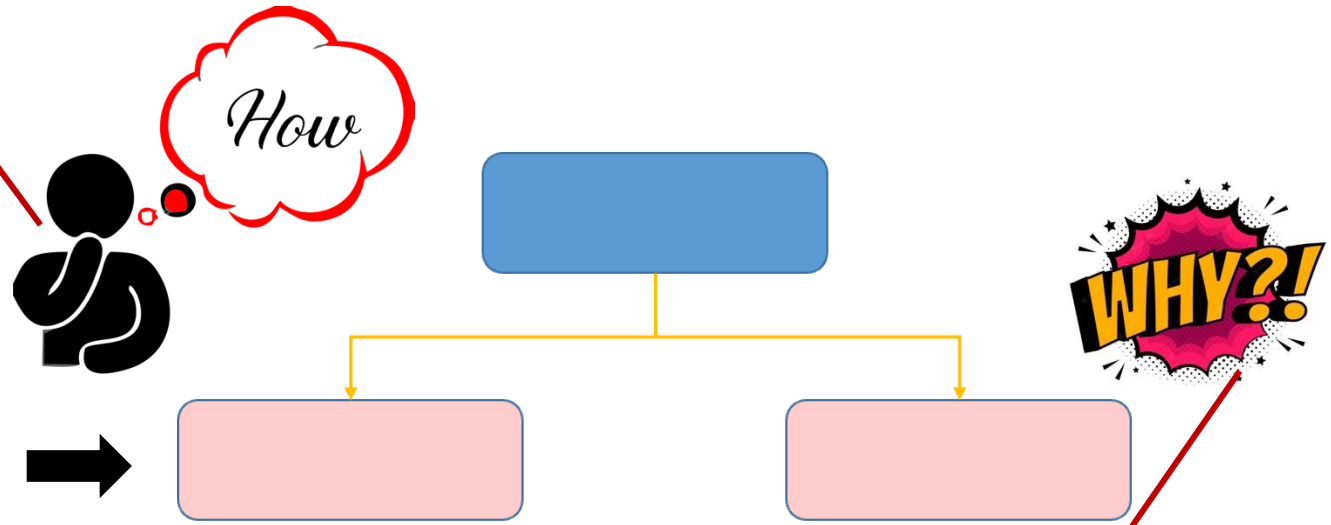


Incorrect

Incorrect

Create new dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
Yes	Yes	167	Yes
Yes	Yes	167	Yes
Yes	Yes	172	No
Yes	Yes	172	No



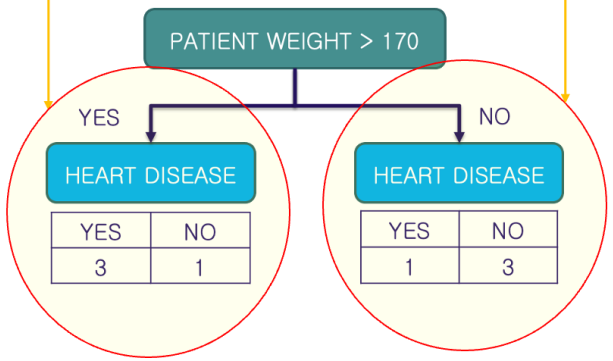
This new stump can handel incorrect classification

How to Build Next Stump

Increase the sample weights of samples that were incorrectly classified and decrease sample weights of samples that were correctly classified. Label {-1, 1}

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	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

- Total error is equal to the sum of the weights of the incorrect classified
- Amount of say = 1/2*log((1-2/8) / (2/8)) = 0.55



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

New sample weight = 1/8 * e^{0.55} = 0.22

```
def update_weights_formular1(w_i, alpha, y, y_pred):
    result = w_i * np.exp(-alpha * y * y_pred)
    w_norm = result / np.sum(result)
    return w_norm
```

How to Build Next Stump

Increase the sample weights of samples that were incorrectly classified and decrease the sample weights of samples that were correctly classified. Label {-1, 1}

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

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

New sample weight = 1/8 * e^{-0.55} = 0.07

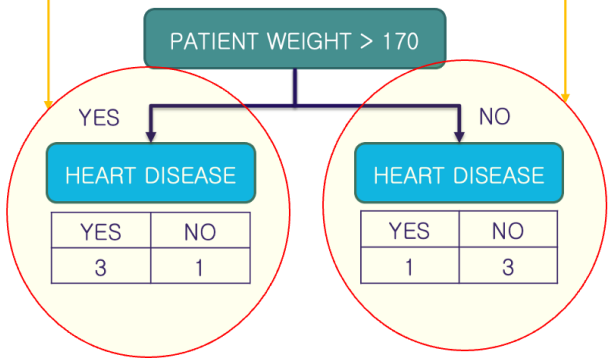
```
def update_weights_formular1(w_i, alpha, y, y_pred):
    result = w_i * np.exp(-alpha * y * y_pred)
    w_norm = result / np.sum(result)
    return w_norm
```

How to Build Next Stump

Increase the sample weights of samples that were incorrectly classified and keep the sample weights of samples that were correctly classified. Label {0, 1}

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	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

- Total error is equal to the sum of the weights of the incorrect classified
- Amount of say = 1/2*log((1-2/8) / (2/8)) = 0.55



$$w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],$$

New sample weight = 1/8 * e^{0.55*1} = 0.22

```
def update_weights_formular2(w_i, alpha, y, y_pred):
    result = w_i * np.exp(alpha * (
        np.not_equal(y, y_pred)).astype(int))
    w_norm = result / np.sum(result)
    return w_norm
```

How to Build Next Stump

Increase the sample weights of samples that were incorrectly classified and keep the sample weights of samples that were correctly classified. Label {0, 1}

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

$$w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],$$

New sample weight = 1/8 * e^{0.55*0} = 0.125

```
def update_weights_formular2(w_i, alpha, y, y_pred):
    result = w_i * np.exp(alpha * (
        np.not_equal(y, y_pred)).astype(int))
    w_norm = result / np.sum(result)
    return w_norm
```

New Sample Weight

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Sample Weight	New Weight	Normal Weight
Yes	Yes	205	Yes	1/8	0.07	0.08
No	Yes	180	Yes	1/8	0.07	0.08
Yes	No	210	Yes	1/8	0.07	0.08
Yes	Yes	167	Yes	1/8	0.22	0.25
No	Yes	156	No	1/8	0.07	0.08
No	Yes	125	No	1/8	0.07	0.08
Yes	No	168	No	1/8	0.07	0.08
Yes	Yes	172	No	1/8	0.22	0.25
Sum				~1.0	0.86	~1.0

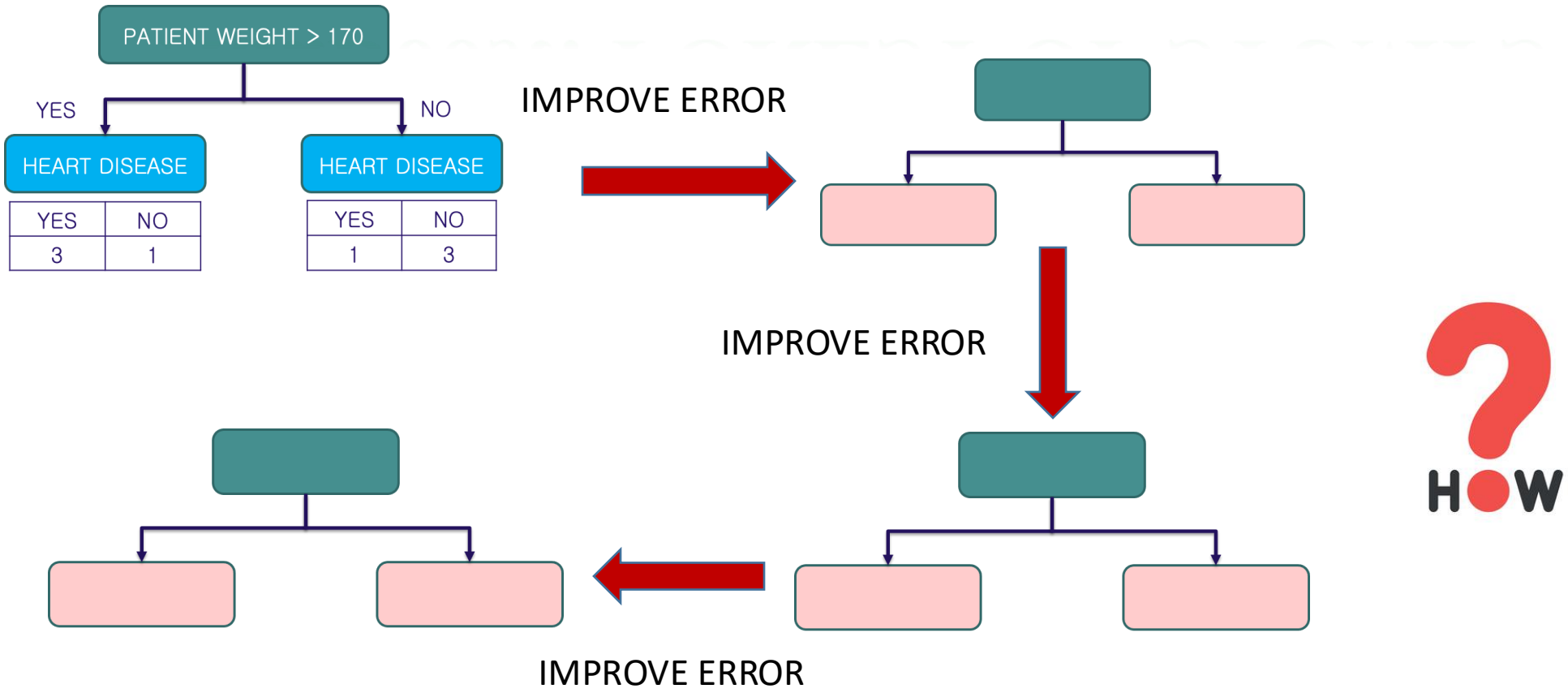
$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

Update

New Sample Weight

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	New Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

AdaBoost: FOREST OF STUMPS



New Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

New Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

Random [0,1] to select samples

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

New Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight	Range
Yes	Yes	205	Yes	0.08	[0-0.08]
No	Yes	180	Yes	0.08	(0.08-0.16]
Yes	No	210	Yes	0.08	(0.16-0.24]
Yes	Yes	167	Yes	0.25	(0.24-0.495]
No	Yes	156	No	0.08	(0.495-0.575]
No	Yes	125	No	0.08	(0.575-0.655]
Yes	No	168	No	0.08	(0.655-0.735]
Yes	Yes	172	No	0.25	(0.735-1.0]
Sum				~1.0	

New Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

Old dataset

Random [0,1] to select samples

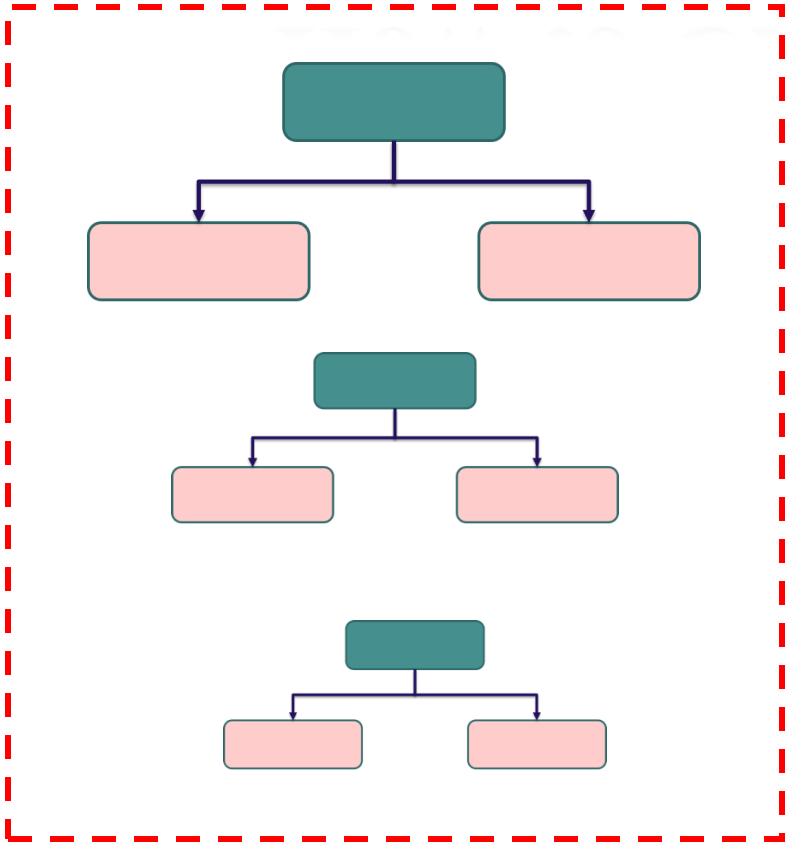
CONTINUE TO BUILD THE NEXT STUMP

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease
Yes	Yes	167	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	Yes	172	No
Yes	Yes	172	No

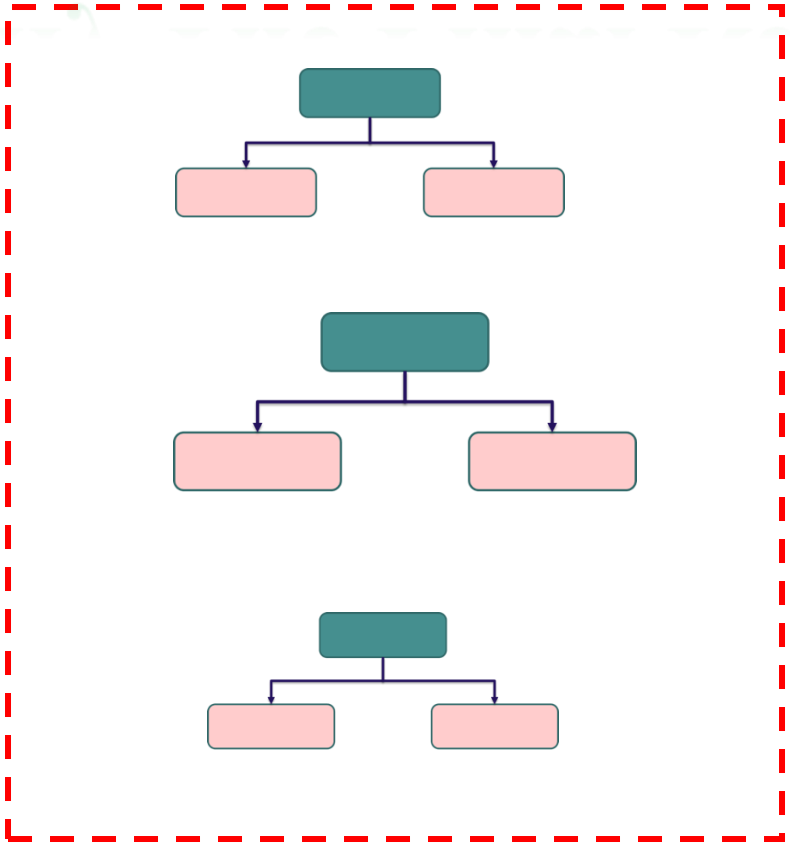
New dataset

Ý tưởng: các sample bị phân loại sai, sẽ được được nhiều hơn vào dataset mới

How to Classify The Final Result



These stumps for predicting heart disease



These stumps for predicting no heart disease

AI VIET NAM
@aivietnam.edu.vn

How to Classify The Final Result

Algorithm 1 *AdaBoost (Freund & Schapire 1997)*

1. *Initialize the observation weights* $w_i = 1/n,\ i = 1, 2, \ldots, n.$
2. *For* $m = 1$ *to* M*:*
   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i*.*
   (b) *Compute*
   $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$
   (c) *Compute*
   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}}.$$
   (d) *Set*
   $$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, 2, \ldots, n.$$
   (e) *Re-normalize* w_i*.*
3. *Output*
   $$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

https://hastie.su.domains/Papers/samme.pdf

AI VIET NAM
@aivietnam.edu.vn

How to Classify The Final Result

SAMME — Stagewise Additive Modeling using a Multi-class Exponential loss function

Algorithm 2 *SAMME*

1. *Initialize the observation weights* $w_i = 1/n,\ i = 1, 2, \ldots, n.$
2. *For* $m = 1$ *to* M*:*
   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i.
   (b) *Compute*

$$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$

   (c) *Compute*

$$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K - 1). \tag{1}$$

   (d) *Set*

$$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, \ldots, n.$$

   (e) *Re-normalize* w_i.
3. *Output*

$$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

https://hastie.su.domains/Papers/samme.pdf

Behind The Scene

The hypothesis function f(x) is defined as:

$$f(x) = \sum_{m=1}^{M} \alpha_m * G_m(x)$$

M trees Label: {-1,1} N Samples

Exponential loss function:

$$Err(f) = \sum_{i=1}^{N} e^{-y_i * f(x_i)}$$

Supposing that:

$$f_m(x) = f_{m-1}(x) + \alpha_m * G_m(x)$$

Expand the error function:

$$Err(f) = \sum_{i=1}^{N} e^{-y_i f_{m-1}(x_i)} * e^{-y_i \alpha_m G_m(x_i)}$$

$$=> Err(\alpha_m, G_m) = \sum_{i=1}^{N} e^{-y_i f_{m-1}(x_i)} * e^{-y_i \alpha_m G_m(x_i)}$$

$$w_i = e^{-y_i f_{m-1}(x_i)}$$

not dependent on α_m and Gm

Behind The Scene

Expand the error function:

$$Err(f) = \sum_{i=1}^{N} e^{-y_i f_{m-1}(x_i)} * e^{-y_i \alpha_m G_m(x_i)}$$

$$=> Err(\alpha_m, G_m) = \sum_{i=1}^{N} e^{-y_i f_{m-1}(x_i)} * e^{-y_i \alpha_m G_m(x_i)}$$

M trees

N Samples

Label: {-1,1}

$$w_i = e^{-y_i f_{m-1}(x_i)}$$

$$Err = e^{-\alpha_m} \sum_{y_i = G_m(x_i)} w_i + e^{\alpha_m} \sum_{y_i \neq G_m(x_i)} w_i$$

It's easy to show that the expression $-y_i\alpha_m G_m(x_i)$ is $-\alpha_m$ if $y_i = G_m(x_i)$ and is α_m if $y_i != G_m(x_i)$.

$$Err = e^{-\alpha_m} \sum_{y_i = G_m(x_i)} w_i + e^{\alpha_m} \sum_{y_i \neq G_m(x_i)} w_i$$

Total weight T_w as:

$$T_w = \sum_{i=1}^{N} w_i$$

Error weight E_w :

$$E_w = \sum_{y_i \neq G_m(x_i)} w_i$$

$$Err = e^{-\alpha_m}(T_w - E_w) + e^{\alpha_m} E_w$$

$$\frac{dErr}{d\alpha_m} = -\alpha_m e^{-\alpha_m}(T_w - E_w) + \alpha_m e^{\alpha_m} E_w$$

Behind The Scene

$$Err = e^{-\alpha_m}(T_w - E_w) + e^{\alpha_m} E_w$$

$$\frac{dErr}{d\alpha_m} = -\alpha_m e^{-\alpha_m}(T_w - E_w) + \alpha_m e^{\alpha_m} E_w$$

Taking log on both sides we have

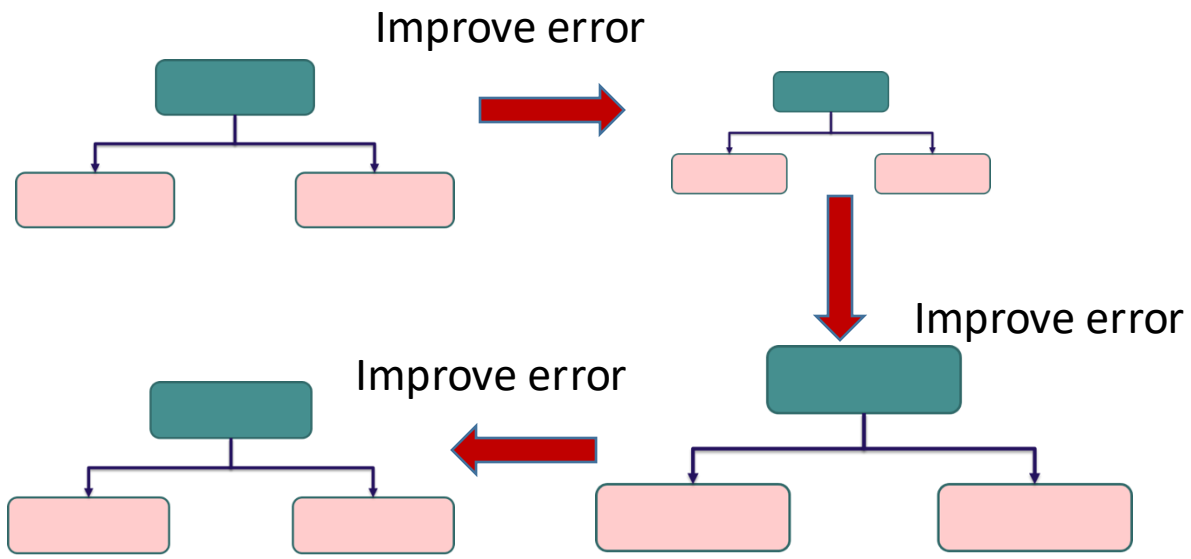
$$\begin{aligned} -\alpha_m e^{-\alpha_m}(T_w - E_w) + \alpha_m e^{\alpha_m} E_w &= 0 \\ => e^{\alpha_m} E_w - e^{-\alpha_m}(T_w - E_w) &= 0 \\ => e^{\alpha_m} E_w &= e^{-\alpha_m}(T_w - E_w) \\ => e^{2\alpha_m} &= (Tw - Ew)/Ew \\ => e^{2\alpha_m} &= \frac{1 - (E_w/T_w)}{(E_w/T_w)} \end{aligned}$$

$$\alpha_m = \frac{1}{2} log \frac{1 - err_m}{err_m}$$

$$err_m = \frac{E_w}{T_w} = \frac{\sum_{y_i \neq G_m(x_i)} w_i}{\sum w_i}$$

Summary

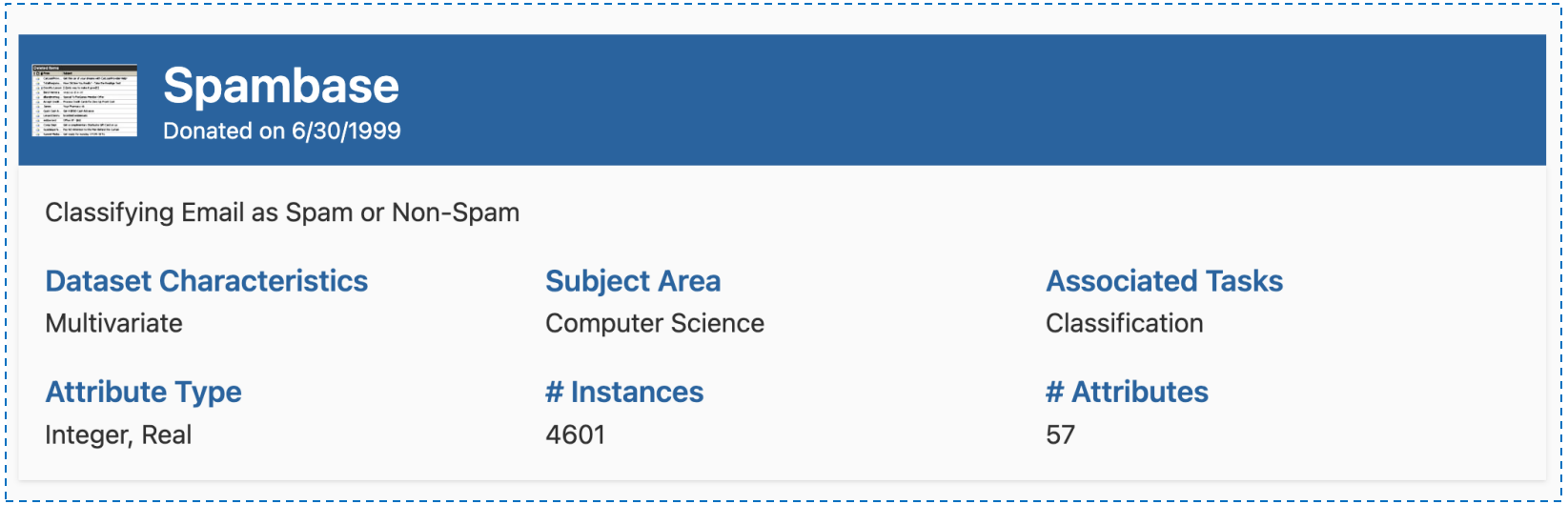
Height	Favorite Color	Gender	Weight
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57



Adaboost builds a stump based on the the error made by previous stumps

Example: Spam Classification

- Classifying Email as Spam or Non-Spam



Spambase
Donated on 6/30/1999

Classifying Email as Spam or Non-Spam

Dataset Characteristics	Subject Area	Associated Tasks
Multivariate	Computer Science	Classification
Attribute Type	**# Instances**	**# Attributes**
Integer, Real	4601	57

http://archive.ics.uci.edu/dataset/94/spambase

Example: Spam Classification

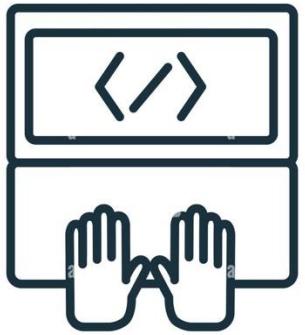
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Example: Spam Classification

Our implementation

```
# Fit model
ab = AIVNAdaBoost()
ab.fit(X_train, y_train, M = 50)

# Predict on test set
y_pred = ab.predict(X_test)
print('The accuracy_score of the model is:', round(accuracy_score(y_test, y_pred), 4))
```

```
The accuracy_score of the model is: 0.9392
```



CUSTOM CODE

```
from sklearn.ensemble import AdaBoostClassifier

ab_sk = AdaBoostClassifier(n_estimators = 50)
ab_sk.fit(X_train, y_train)
y_pred_sk = ab_sk.predict(X_test)
print('The accuracy_score of the model is:', round(accuracy_score(y_test, y_pred_sk), 4))
```

```
The accuracy_score of the model is: 0.9435
```

Using Sklearn library

Outline

- Boosting Techniques
- AdaBoost Clearly Explain
- Gradient Boost Clearly Explain
- Time Series Data: Predicting Energy Consumption
- Summary

Gradient Boost For Regression

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

Input **Output**

Tree-based Gradient Boost

- Step 1: Build 1st tree
  - Calculate the average of weights

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

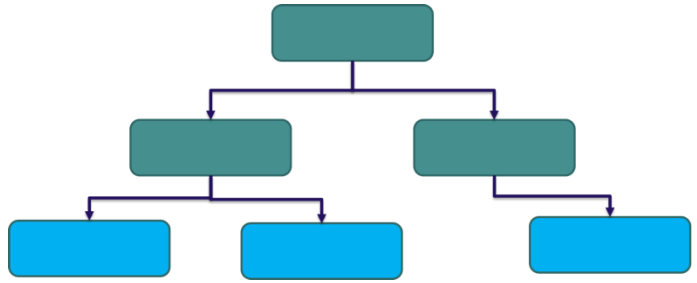
Node of 1st Tree

Average of Weights: 71.17

Tree-based Gradient Boost

- Step 2:
  - Build 2nd tree

Average of weights: 71.17

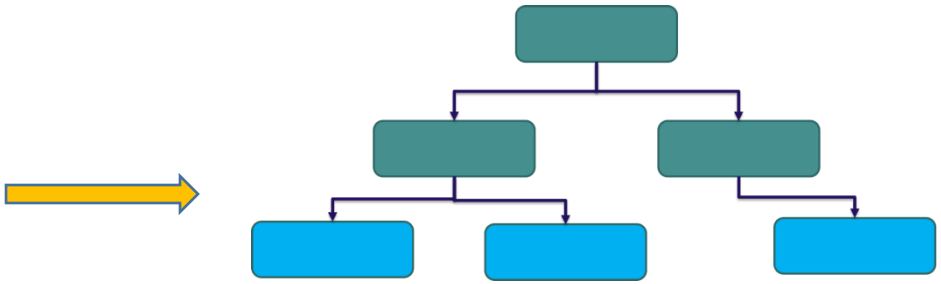


Height	Favorite Color	Gender	Weight
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

Tree-based Gradient Boost

- Step 2:
  - ➢Build 2nd tree

Average of weights: 71.2

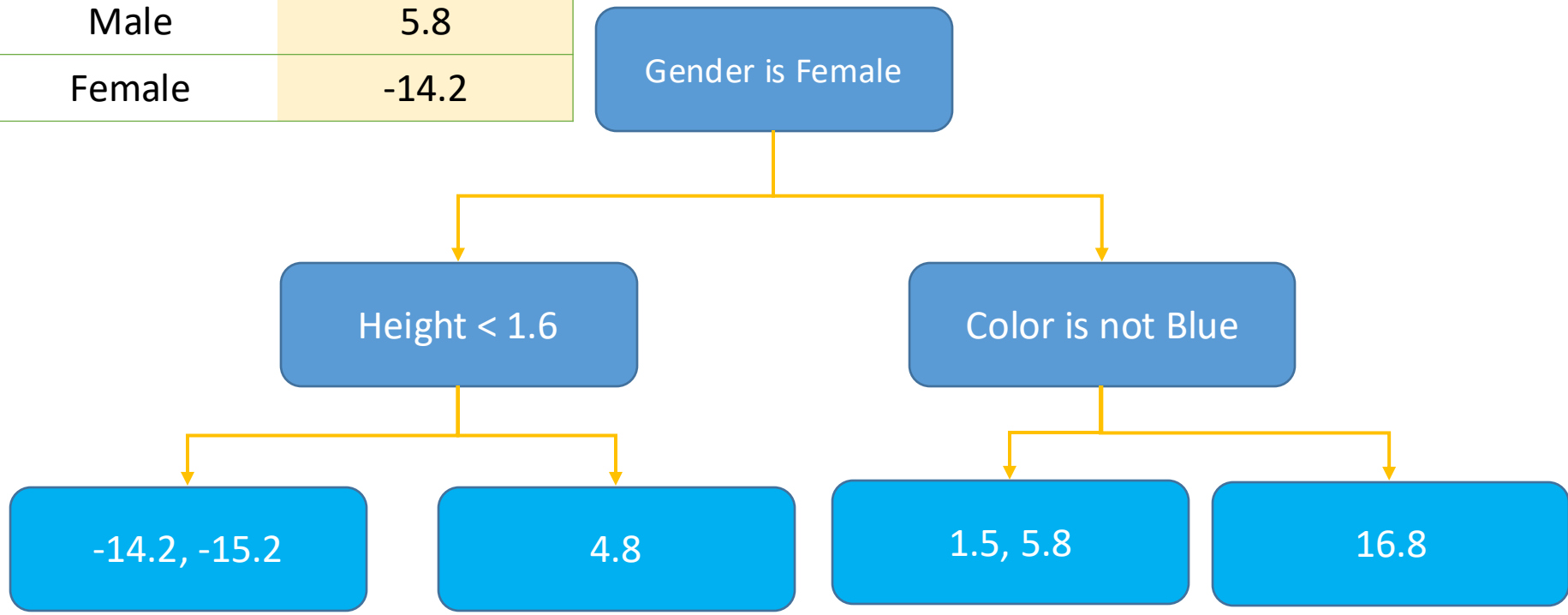


Height	Favorite Color	Gender	Weight	Residual Error
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	1.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

Tree-based Gradient Boost

Height	Favorite Color	Gender	Residual Error
1.6	Blue	Male	16.8
1.6	Green	Female	1.8
1.5	Blue	Female	-15.2
1.8	Red	Male	1.8
1.5	Green	Male	5.8
1.4	Blue	Female	-14.2

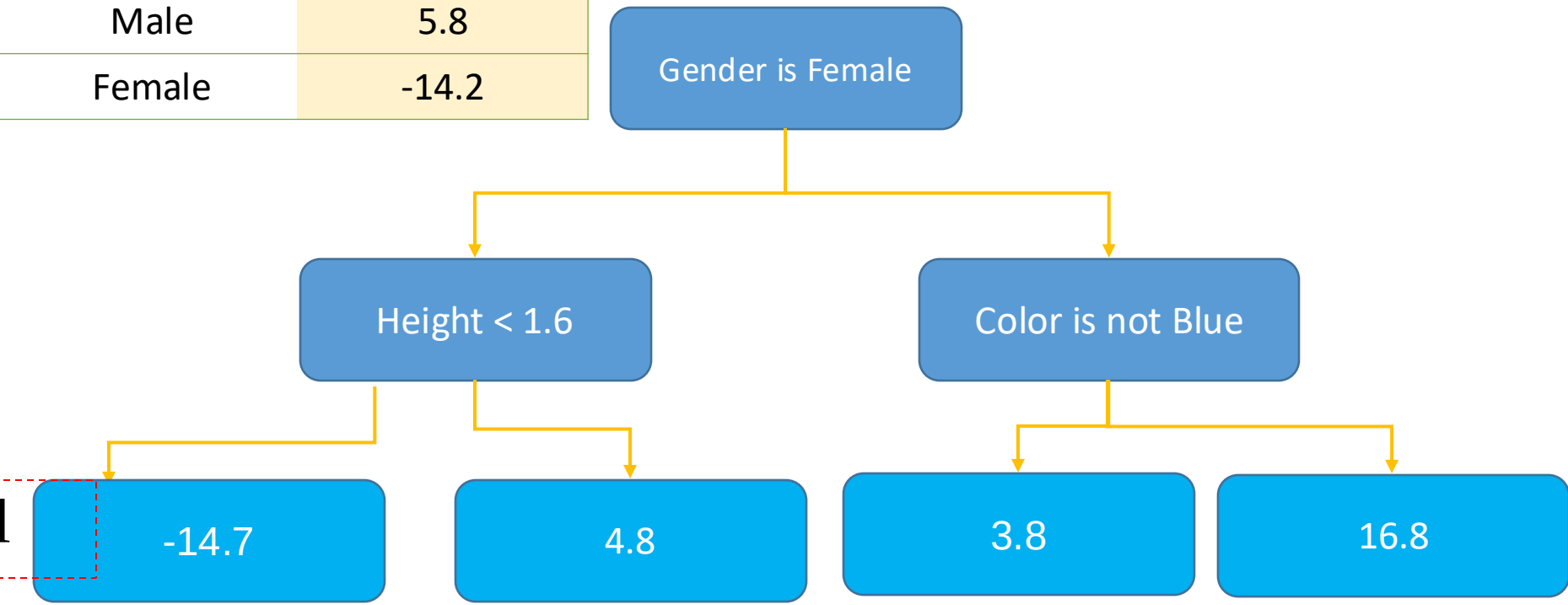
Tại sao lại xây dựng cây dự đoán Residual Error



Tree-based Gradient Boost

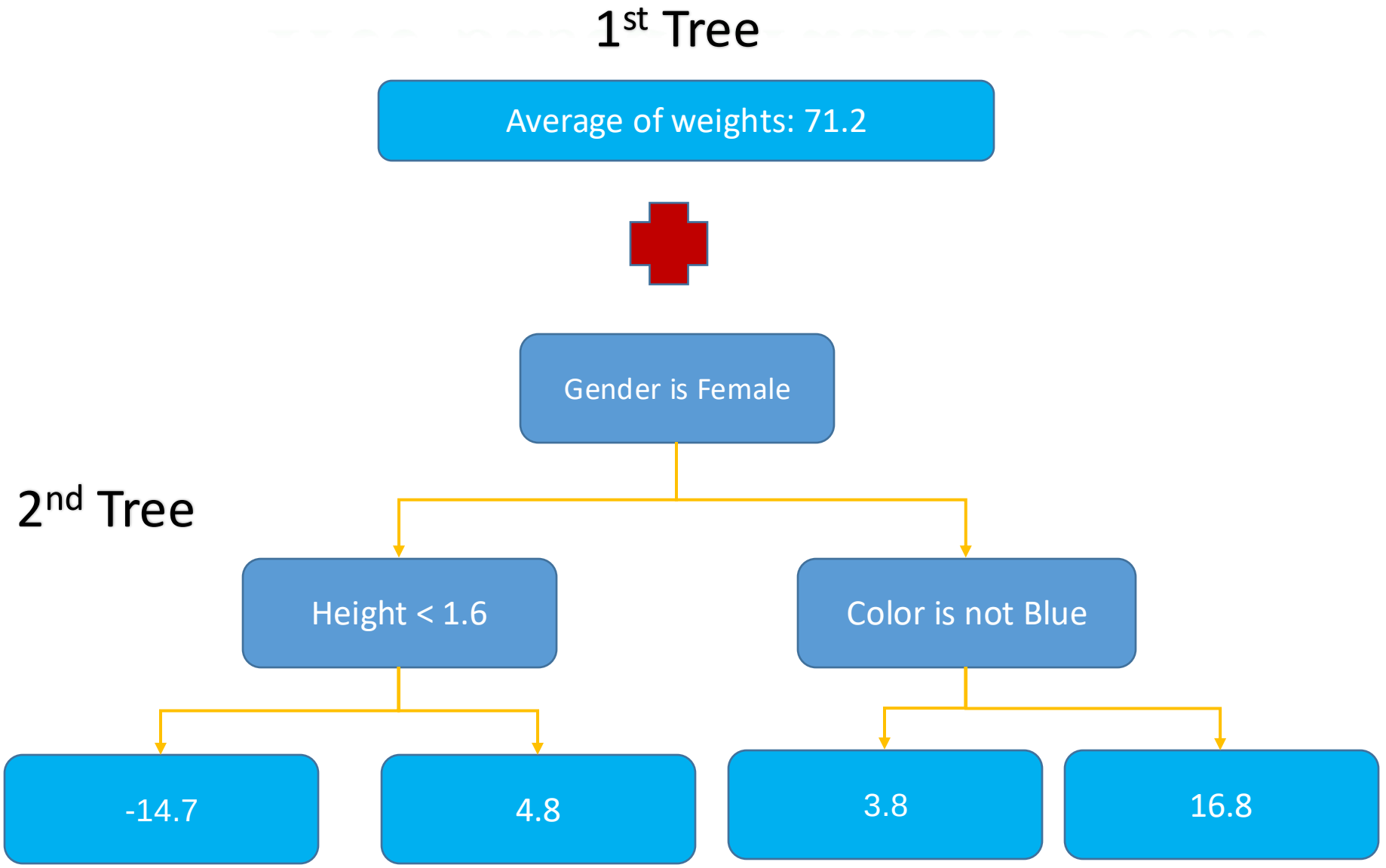
Height	Favorite Color	Gender	Residual Error
1.6	Blue	Male	16.8
1.6	Green	Female	1.8
1.5	Blue	Female	-15.2
1.8	Red	Male	1.8
1.5	Green	Male	5.8
1.4	Blue	Female	-14.2

Tại sao lại xây dựng cây dự đoán Residual Error



Trung bình residual

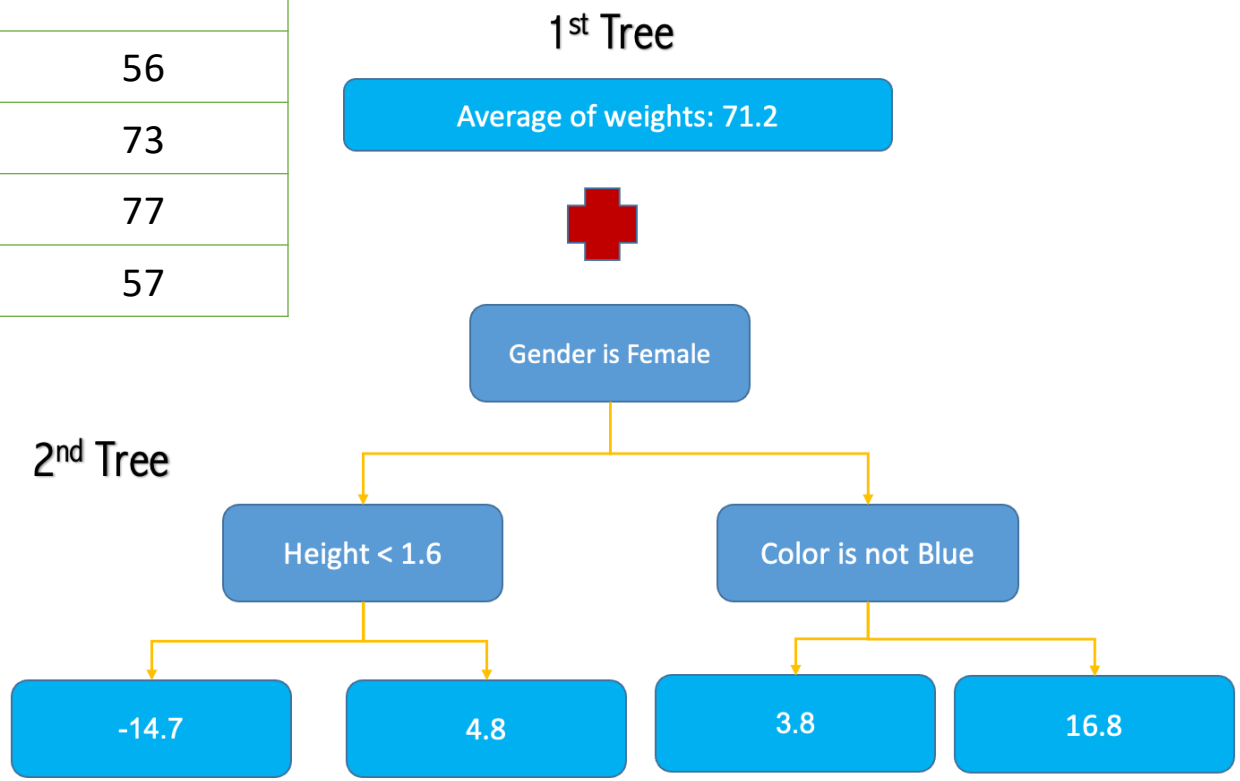
Tree-based Gradient Boost



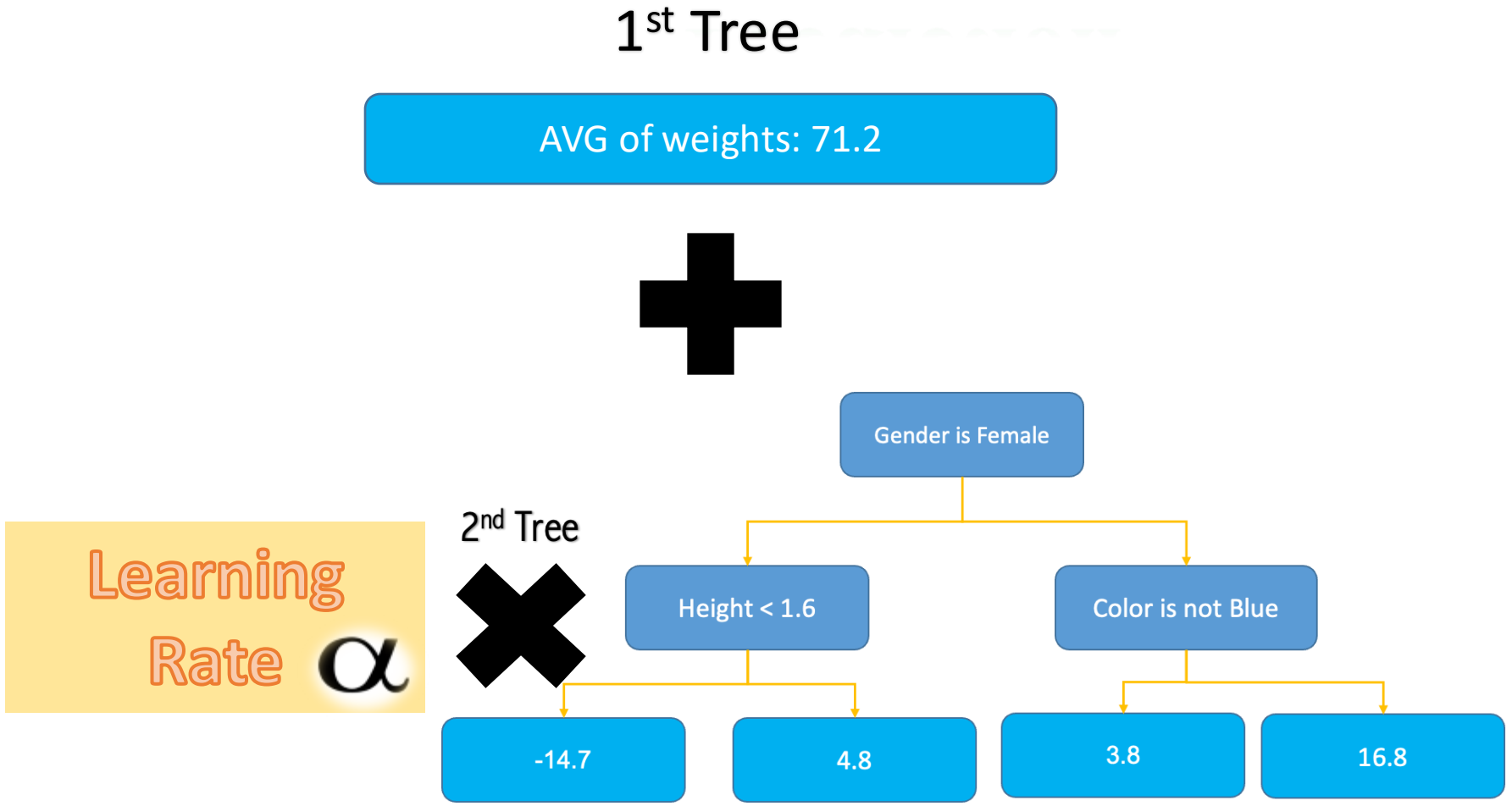
Prediction

Height	Favorite Color	Gender	Weight	Prediction
1.6	Blue	Male	88	88
1.6	Green	Female	76	76
1.5	Blue	Female	56	56
1.8	Red	Male	73	73
1.5	Green	Male	77	77
1.4	Blue	Female	57	57

OVER FITTING



Prediction

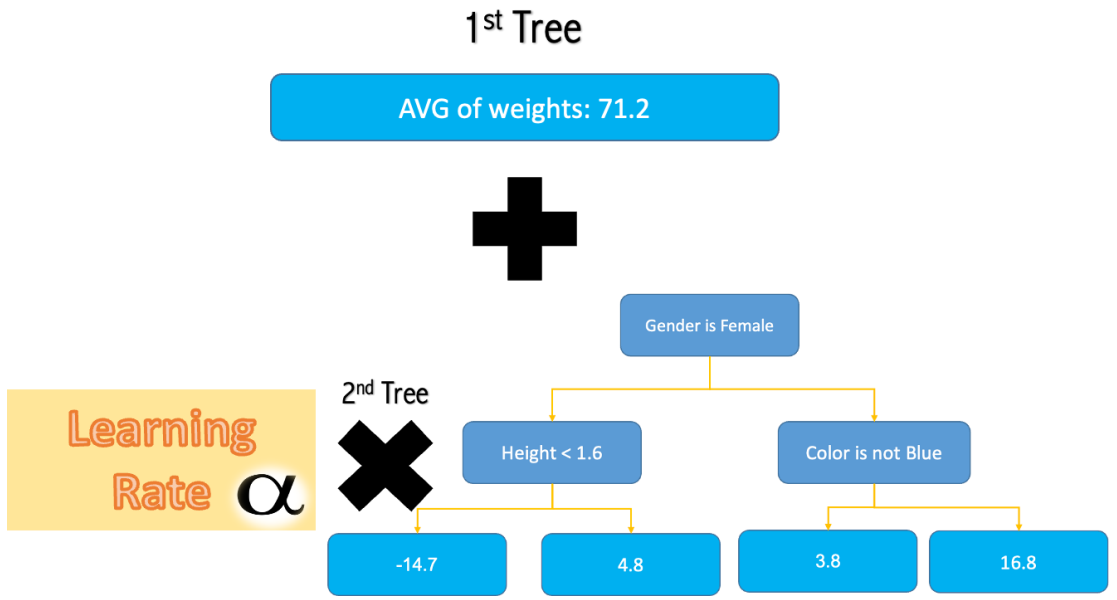


Prediction

Height	Favorite Color	Gender	Weight	Prediction
1.6	Blue	Male	88	74.56

Prediction = 71.2 + 0.2 * 16.8 = 74.56

$\alpha = 0.2$

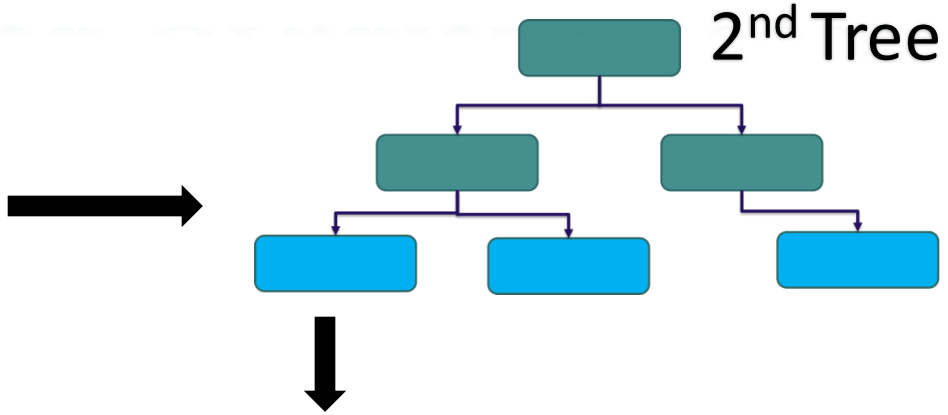


Tree-based Gradient Boost

- Step 3:
  - ➢Build 3rd tree

1st Tree

Average of weights: 71.2



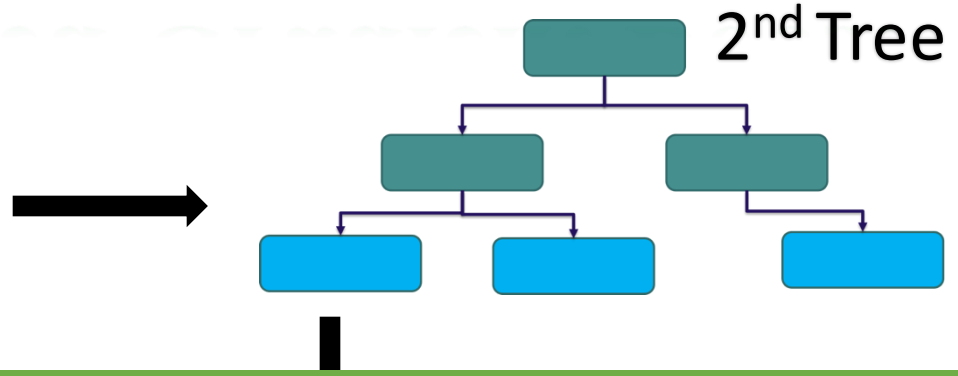
Height	Favorite Color	Gender	Weight	Predicted Weight	Residual Error
1.6	Blue	Male	88	74.56	12.44
1.6	Green	Female	76	…	…
1.5	Blue	Female	56	…	…
1.8	Red	Male	73	…	…
1.5	Green	Male	77	…	…
1.4	Blue	Female	57	…	…

Tree-based Gradient Boost

- Step 3:
  - ➢Build 3rd tree

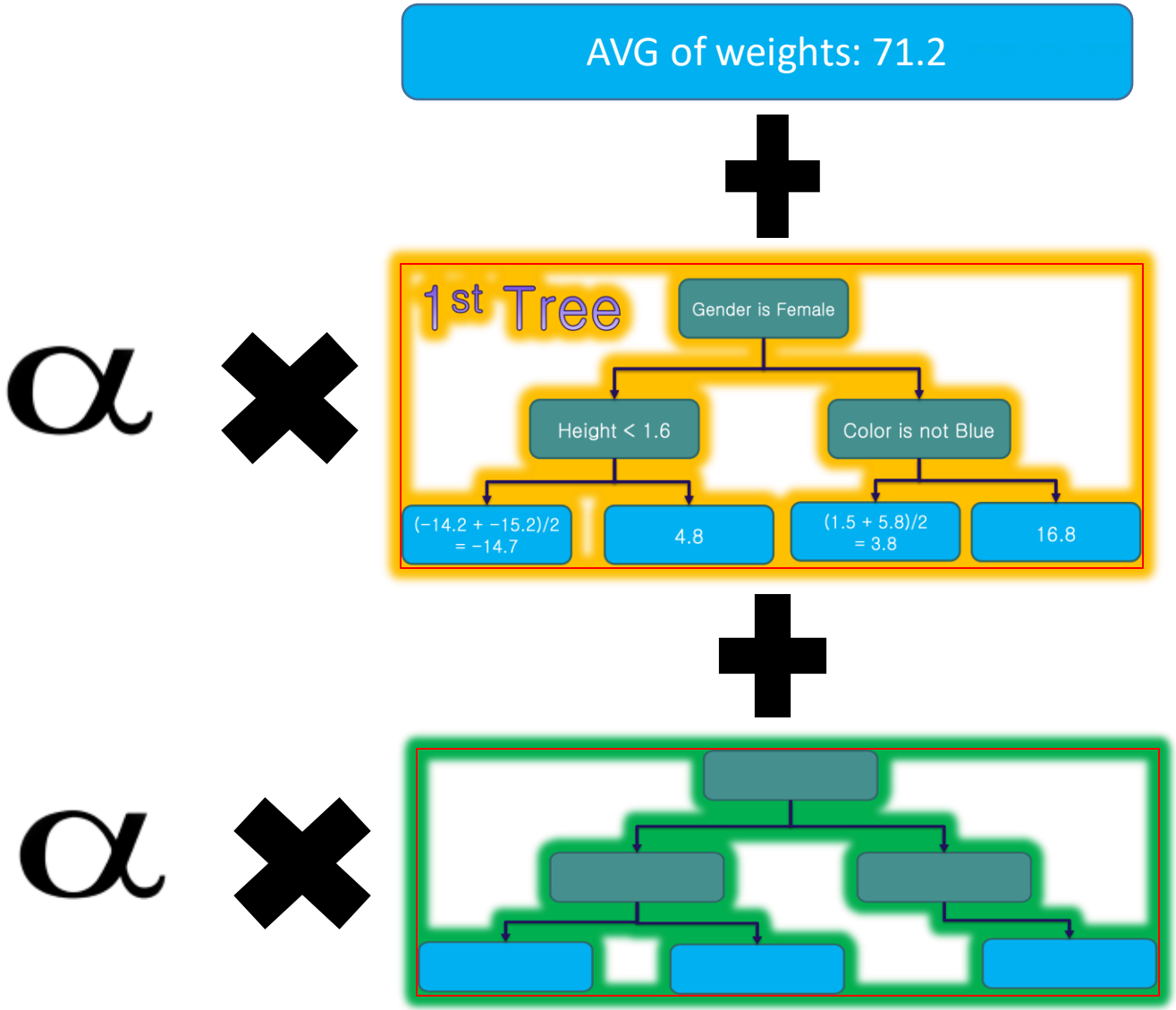
1st Tree

Average of weights: 71.2



Height	Favorite Color	Gender	First Tree Residual	Second Tree Residual	Third Tree Residual
1.6	Blue	Male	16.8	12.44	
1.6	Green	Female	1.8	…	
1.5	Blue	Female	-15.2	…	
1.8	Red	Male	1.8	…	
1.5	Green	Male	5.8	…	
1.4	Blue	Female	-14.2	…	

Prediction



AI VIET NAM
@aivietnam.edu.vn

Gradient Boost: Behind The Scenes

Loss Function:

n

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

- $\{(x_i, y_i)\}^n_1$

Loss function = $L(y_i, F(x))$ = 1/2* (Output - Predicted)^2

$$J = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y} - y)^2$$

$$\frac{dL}{dPredicted}$$ = 2/2 (Output – Predicted) * -1 = - (Output – Predicted)

Tricky implementation here

Gradient Boost: Behind The Scenes

- Step 1:
  - Initialize a model with a constant value:
  - $\boldsymbol{F_{0(x)} = \underset{\delta}{argmin} \sum_{i=1}^{n} L(y, \delta)}$

			y
Height	Favorite Color	Gender	Weight
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56

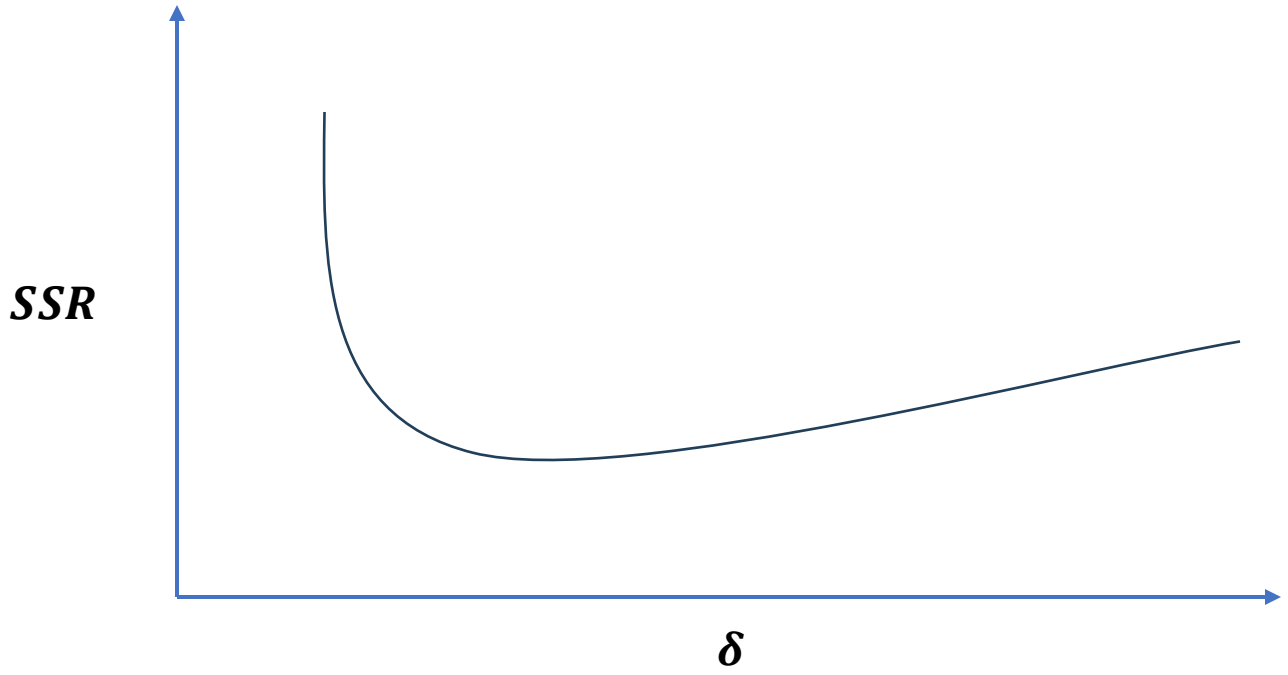
SSR = 1/2 {(88 – $\boldsymbol{\delta}$)^2 + (76 – $\boldsymbol{\delta}$)^2 + (56 - $\boldsymbol{\delta}$)^2}

$\frac{dSSR}{d\boldsymbol{\delta}}$ = -(88 - $\boldsymbol{\delta}$) – (76 - $\boldsymbol{\delta}$) – (56 - $\boldsymbol{\delta}$) = 0

$\boldsymbol{\delta} = \frac{88+76+56}{3}$ = 73.3 = average of all sample' weights

Gradient Boost: Behind The Scenes

- **Step 1:**
- Initialize a model with a constant value:
  - $F_{0(x)} = argmin \sum_{i=1}^{n} L(y, \delta)$ = 73.3



Gradient Boost: Behind The Scenes

- M is the number of trees
- n is the number of samples

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

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,...,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

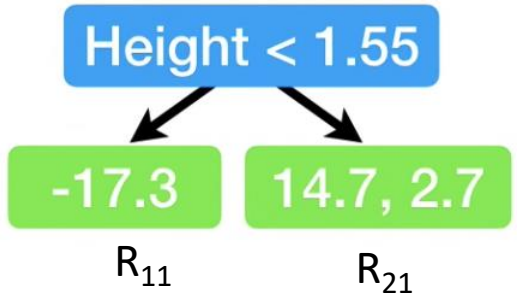
(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Gradient Boost: Behind The Scenes

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

R is the residual error; m is the m-th tree; j is the j-th leaf

Height	Favorite Color	Gender	Weight	r_{i1}
1.6	Blue	Male	88	14.7
1.6	Green	Female	76	2.7
1.5	Blue	Female	56	-17.3



Step 1

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

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$

$$F_{m\_1}(x) = F_0(x) = 73.3$$

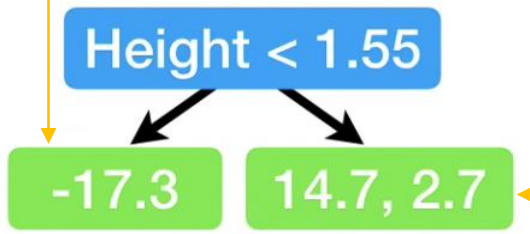
R is the residual error; m is the m-th tree; j is the i-th leaf

Height	Favorite Color	Gender	Weight	r_{i1}
1.6	Blue	Male	88	14.7
1.6	Green	Female	76	2.7
1.5	Blue	Female	56	-17.3

$$\gamma_{11} = \underset{\gamma}{\operatorname{argmin}} \left[\frac{1}{2} \{y3 - (F0(x3) + \gamma)\}^2\right]$$
$$\gamma_{11} = \underset{\gamma}{\operatorname{argmin}} \left[\frac{1}{2} \{56 - (73.3 + \gamma)\}^2\right]$$
$$\gamma_{11} = \underset{\gamma}{\operatorname{argmin}} \left[\frac{1}{2} \{-17.3 - \gamma\}^2\right]$$

γ_{11} = -17.3

γ_{21} = 8.7



xi ∈ R_{11} xi ∈ R_{21}

Step 1

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

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$

$$F_{m\_1}(x) = F_0(x) = 73.3$$

R is the residual error; m is the m-th tree; j is the i-th leaf

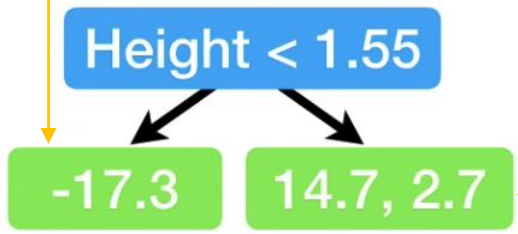
Height	Favorite Color	Gender	Weight	r_{i1}
1.6	Blue	Male	88	14.7
1.6	Green	Female	76	2.7
1.5	Blue	Female	56	-17.3

$$\gamma_{21} = \underset{\gamma}{\operatorname{argmin}} \left[\frac{1}{2}\{y1 - (F0(x1) + \gamma)\}^2 + \frac{1}{2}\{y2 - (F0(x2) + \gamma)\}^2\right]$$

$$\gamma_{11} = \underset{\gamma}{\operatorname{argmin}} \left[\frac{1}{2}\{88 - (73.3 + \gamma)\}^2 + \frac{1}{2}\{76 - (73.33 + \gamma)\}^2\right]$$

$$\gamma_{11} = \underset{\gamma}{\operatorname{argmin}} \left[\frac{1}{2}\{14{,}7 - \gamma\}^2 + \frac{1}{2}\{2.7 - \gamma)\}^2\right]$$

$\gamma_{21} = 8.7$



xi ∈ R_{11} xi ∈ R_{21}

$\gamma_{11} = -17.3$ $\gamma_{21} = 8.7$

Giá trị trung bình của 2 samples

Step 2

Step 1

$$xi \in R_{11}$$

What? How? Why?

$$F_{m\_1}(x) = F_0(x) = 73.3$$

Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

R is the residual error; m is the m-th tree; j is the i-th leaf

Height	Favorite Color	Gender	Weight	r_{i1}	F1(X)
1.6	Blue	Male	88	14.7	74.2
1.6	Green	Female	76	2.7	74.2
1.5	Blue	Female	56	-17.3	71.6

$xi \in R_{11}$ $xi \in R_{21}$

$\gamma_{11} = -17.3$ $\gamma_{21} = 8.7$

Repeat for the next m + 1 iteration

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

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1\ldots J_m$

(C) For $j = 1\ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Summary

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

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^n$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

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

3. Output $F_M(x)$.

Outline

- Boosting Techniques
- AdaBoost Clearly Explain
- Gradient Boost Clearly Explain
- Time Series Data: Predicting Energy Consumption
- Summary

AI VIET NAM
@aivietnam.edu.vn

Time Series Forecasting

We will focus on the energy consumption problem, where given a sufficiently large dataset of the daily energy consumption of different households in a city, we are tasked to predict as accurately as possible the future energy demands.

london_energy

LCLid	Date	KWH
MAC000002	2012-10-12	7.098
MAC000002	2012-10-13	11.087
MAC000002	2012-10-14	13.223
MAC000002	2012-10-15	10.257
MAC000002	2012-10-16	9.769
MAC000002	2012-10-17	10.885
MAC000002	2012-10-18	10.751
MAC000002	2012-10-19	8.431
MAC000002	2012-10-20	17.578
MAC000002	2012-10-21	24.49
MAC000002	2012-10-22	18.885
MAC000002	2012-10-23	10.485
MAC000002	2012-10-24	15.537

Preprocessing

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("/content/drive/MyDrive/AIO2024/london_energy.csv")
print(df.isna().sum())
df.head()
```

```
LCLid    0
Date     0
KWH      0
dtype: int64
```

	LCLid	Date	KWH
0	MAC000002	2012-10-12	7.098
1	MAC000002	2012-10-13	11.087
2	MAC000002	2012-10-14	13.223
3	MAC000002	2012-10-15	10.257
4	MAC000002	2012-10-16	9.769

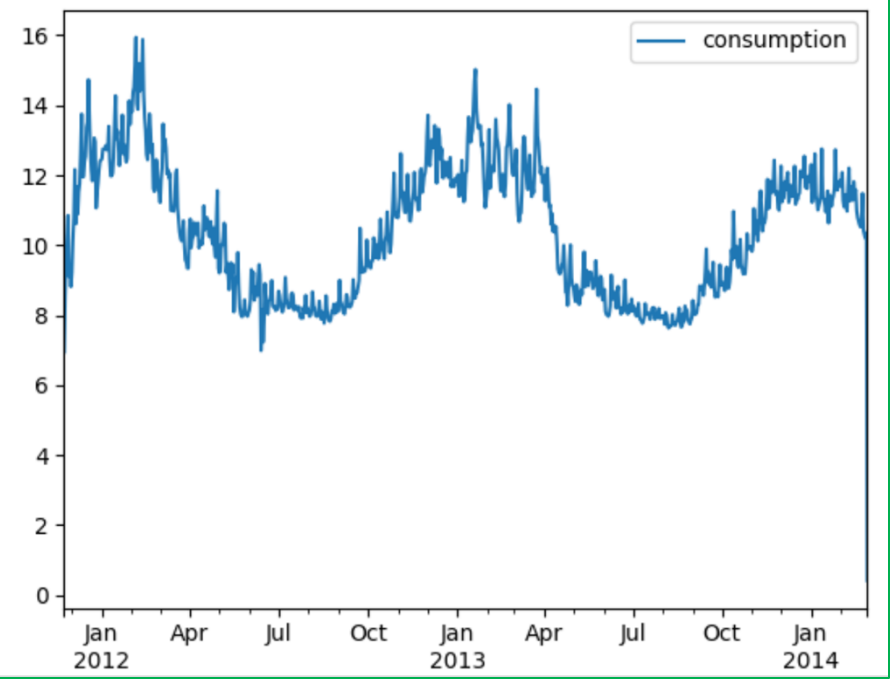
Time Series Forecasting

We will focus on the energy consumption problem, where given a sufficiently large dataset of the daily energy consumption of different households in a city, we are tasked to predict as accurately as possible the future energy demands.

Consumption changes through the years

```
df_avg_consumption.plot(x="date", y="consumption")
```

```
<Axes: xlabel='date'>
```



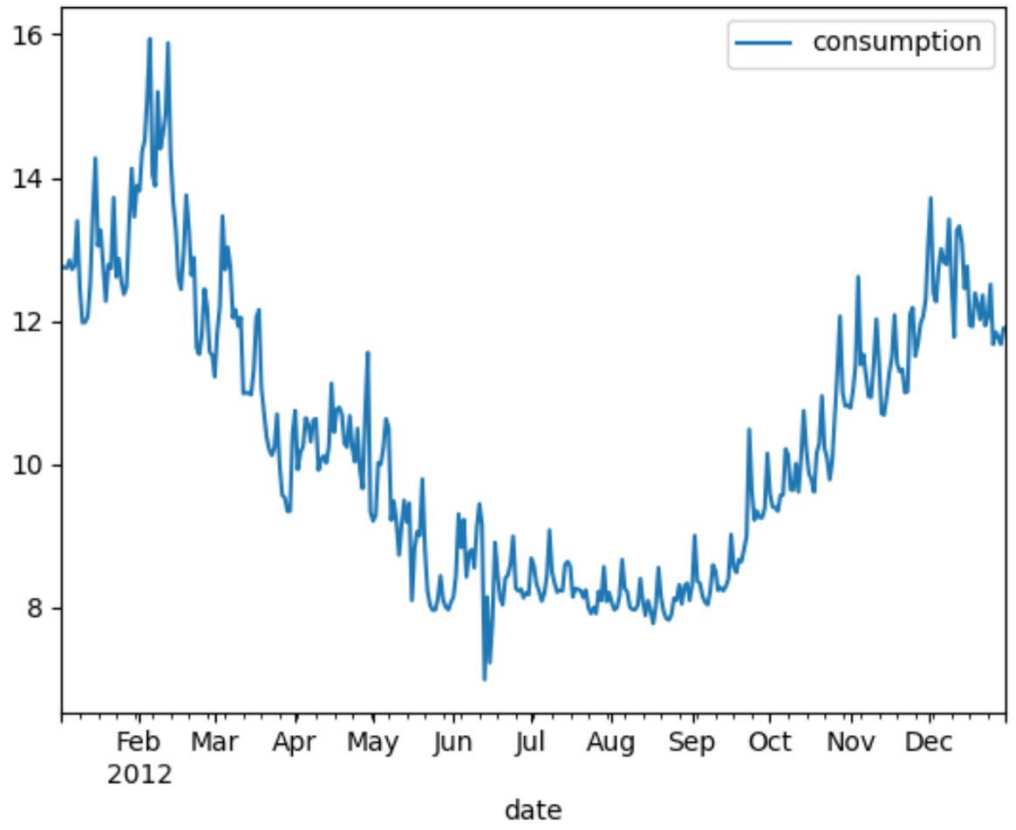
During the winter months we observe high demands in energy, while throughout the summer the consumption is at the lowest levels.

Time Series Forecasting

Visualize the fluctuation in the span of a year we can do

```
df_avg_consumption.query("date > '2012-01-01' & date < '2013-01-01'").plot(x="date", y="consumption")
```

```
<Axes: xlabel='date'>
```



Time Series Forecasting

We have only one feature: the full date. We can extract different features based on the full date such as the day of the week, the day of the year, the month and others

```
df_avg_consumption["day_of_week"] = df_avg_consumption["date"].dt.dayofweek
df_avg_consumption["day_of_year"] = df_avg_consumption["date"].dt.dayofyear
df_avg_consumption["month"] = df_avg_consumption["date"].dt.month
df_avg_consumption["quarter"] = df_avg_consumption["date"].dt.quarter
df_avg_consumption["year"] = df_avg_consumption["date"].dt.year

df_avg_consumption.head()
```

	date	consumption	day_of_week	day_of_year	month	quarter	year
0	2011-11-23	6.952692	2	327	11	4	2011
1	2011-11-24	8.536480	3	328	11	4	2011
2	2011-11-25	9.499781	4	329	11	4	2011
3	2011-11-26	10.267707	5	330	11	4	2011
4	2011-11-27	10.850805	6	331	11	4	2011

AI VIET NAM
@aivietnam.edu.vn

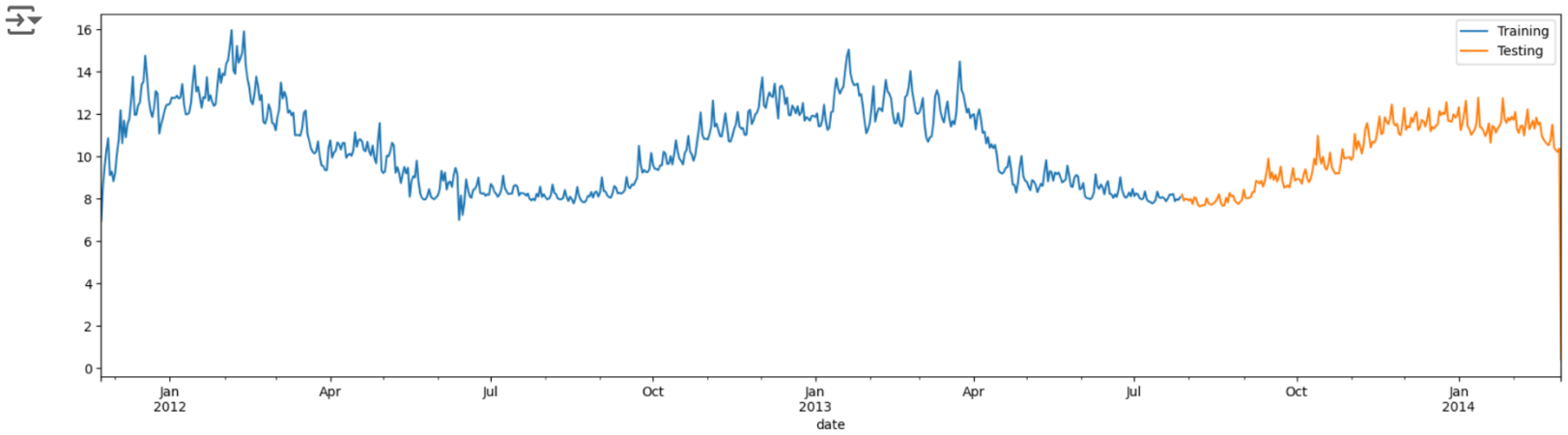
Time Series Forecasting

The dataset contains almost 2.5 years of data, so for the testing set we will use only the last 6 months

```
[6]  training_mask = df_avg_consumption["date"] < "2013-07-28"
     training_data = df_avg_consumption.loc[training_mask]
     print(training_data.shape)

     testing_mask = df_avg_consumption["date"] >= "2013-07-28"
     testing_data = df_avg_consumption.loc[testing_mask]
     print(testing_data.shape)
```

```
(613, 7)
(216, 7)
```

```
figure, ax = plt.subplots(figsize=(20, 5))
training_data.plot(ax=ax, label="Training", x="date", y="consumption")
testing_data.plot(ax=ax, label="Testing", x="date", y="consumption")
plt.show()
```



AI VIET NAM
@aivietnam.edu.vn

Time Series Forecasting

Prepare dataset for Training and Testing

```
[8]  # Dropping unnecessary `date` column
     training_data = training_data.drop(columns=["date"])
     testing_dates = testing_data["date"]
     testing_data = testing_data.drop(columns=["date"])

     X_train = training_data[["day_of_week", "day_of_year", "month", "quarter", "year"]]
     y_train = training_data["consumption"]

     X_test = testing_data[["day_of_week", "day_of_year", "month", "quarter", "year"]]
     y_test = testing_data["consumption"]
```

Performance Evaluation and Visualization

```
[12] from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,\
       mean_squared_error

     def evaluate_model(y_test, prediction):
       print(f"MAE: {mean_absolute_error(y_test, prediction)}")
       print(f"MSE: {mean_squared_error(y_test, prediction)}")
       print(f"MAPE: {mean_absolute_percentage_error(y_test, prediction)}")

     def plot_predictions(testing_dates, y_test, prediction):
       df_test = pd.DataFrame({"date": testing_dates, "actual": y_test, "prediction": prediction })
       figure, ax = plt.subplots(figsize=(10, 5))
       df_test.plot(ax=ax, label="Actual", x="date", y="actual")
       df_test.plot(ax=ax, label="Prediction", x="date", y="prediction")
       plt.legend(["Actual", "Prediction"])
       plt.show()
```

Time Series Forecasting

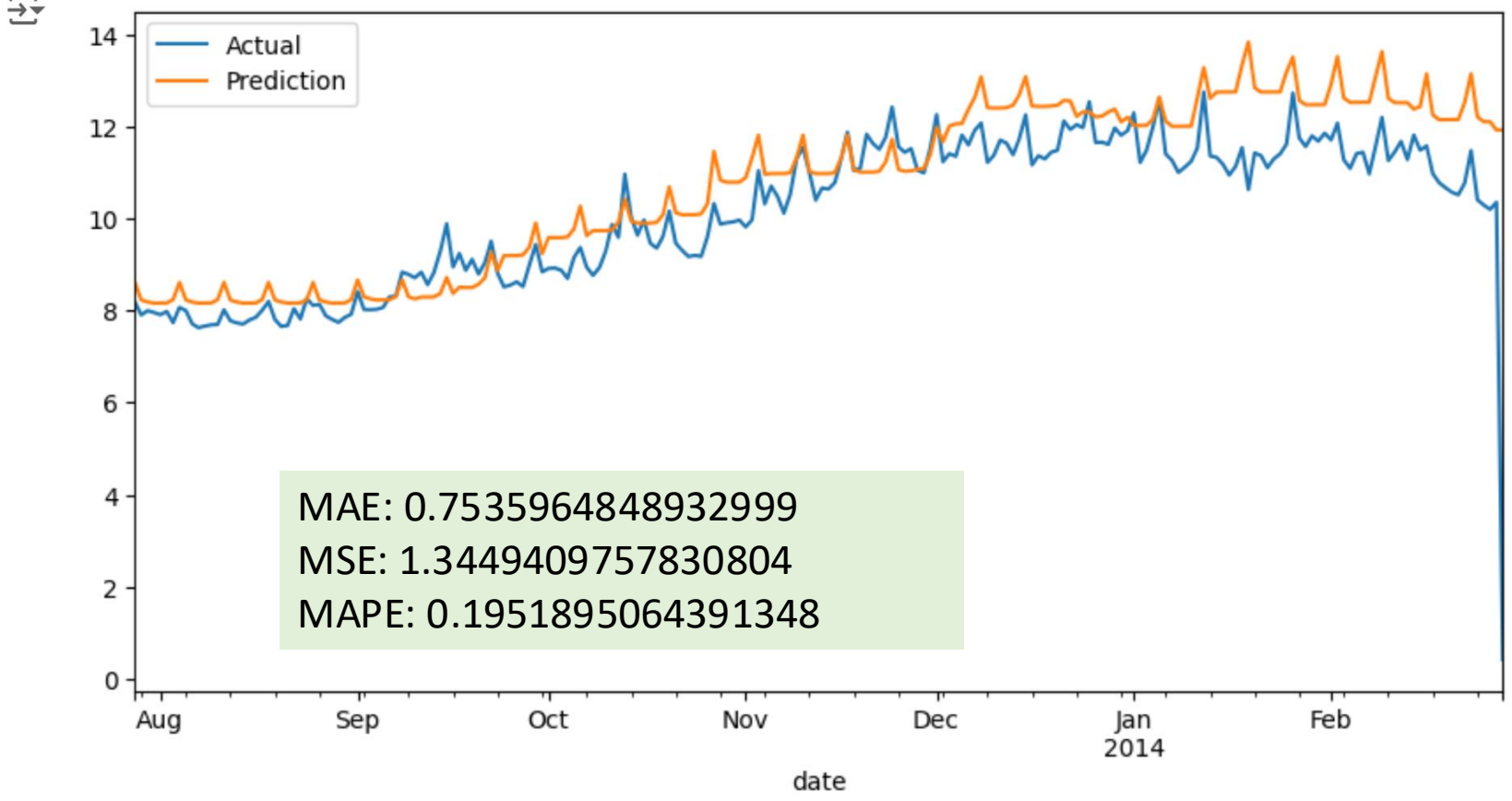
Gradient Boosting For Training

```
[14] from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
     from sklearn.ensemble import GradientBoostingRegressor

     # XGBoost
     cv_split = TimeSeriesSplit(n_splits=4, test_size=100)
     model = GradientBoostingRegressor()
     parameters = {
         "max_features": [3, 4, 5],
         "learning_rate": [0.01, 0.05],
         "n_estimators": [100, 300]
     }


     grid_search = GridSearchCV(estimator=model, cv=cv_split, param_grid=parameters)
     grid_search.fit(X_train, y_train)
```

Time Series Forecasting

Gradient Boosting For Predicting

```
# Evaluating GridSearch results
prediction = grid_search.predict(X_test)
plot_predictions(testing_dates, y_test, prediction)
evaluate_model(y_test, prediction)
```



MAE: 0.7535964848932999
MSE: 1.3449409757830804
MAPE: 0.1951895064391348


Thank
you