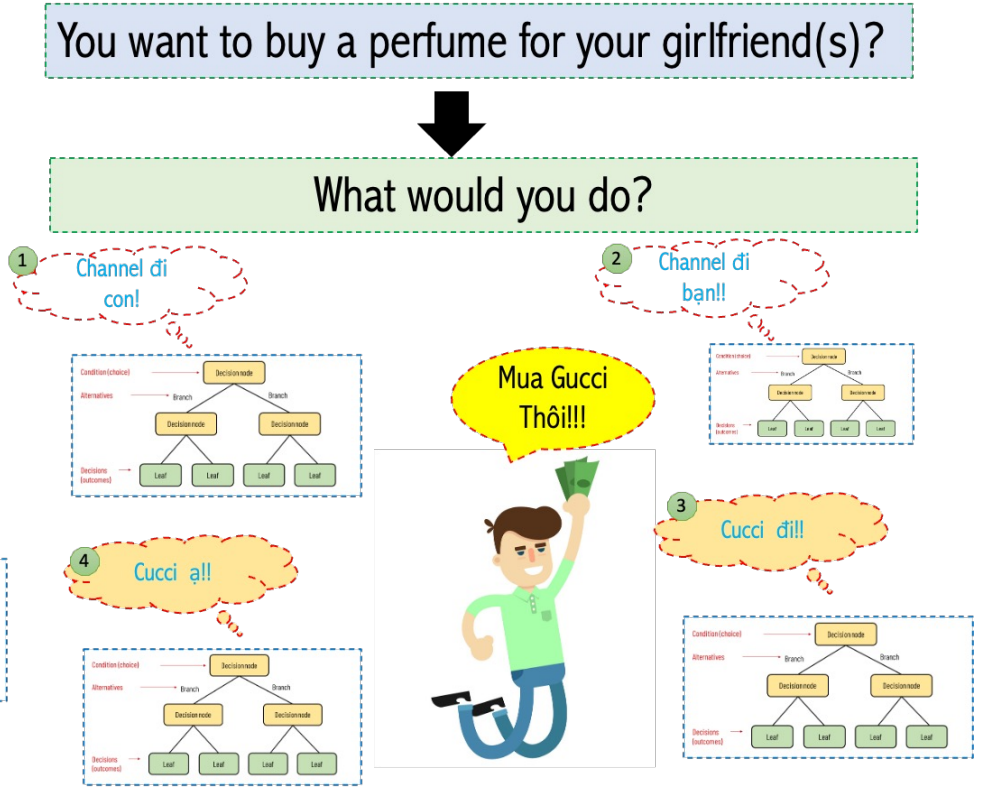
AI VIETNAM
All-in-One Course

Random Forest



Vinh Dinh Nguyen
PhD in Computer Science

Outline

- Decision Tree Review
- Random Forest
- Fill in missing data with Random Forest
- Case study

Outline

- Decision Tree Review
- Random Forest
- Fill in missing data with Random Forest
- Case study

Decision Tree Review

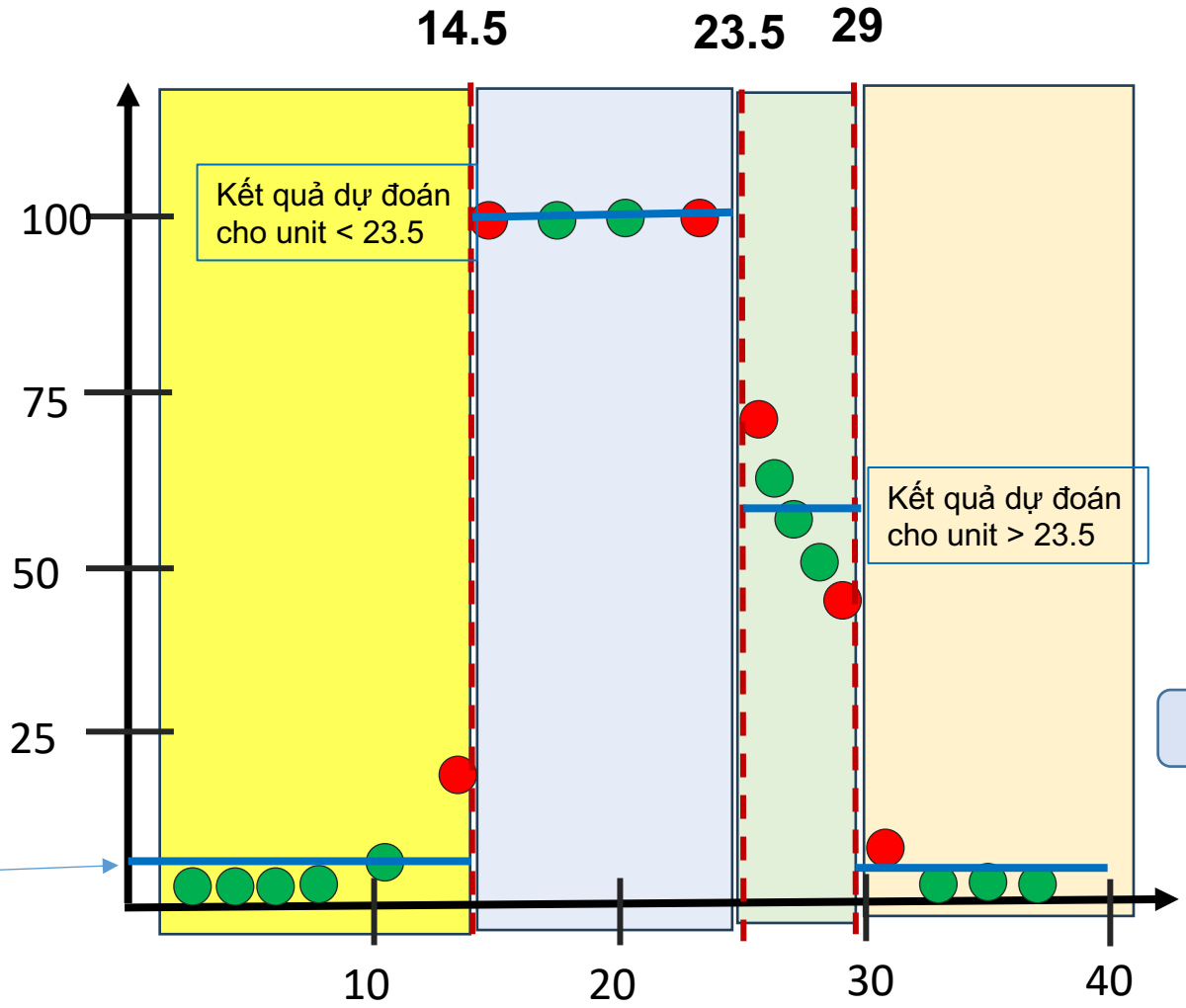
Unit(đơn vị)	Effect (hiệu quả) (%)
10	98
20	0
35	100
5	44
…	…

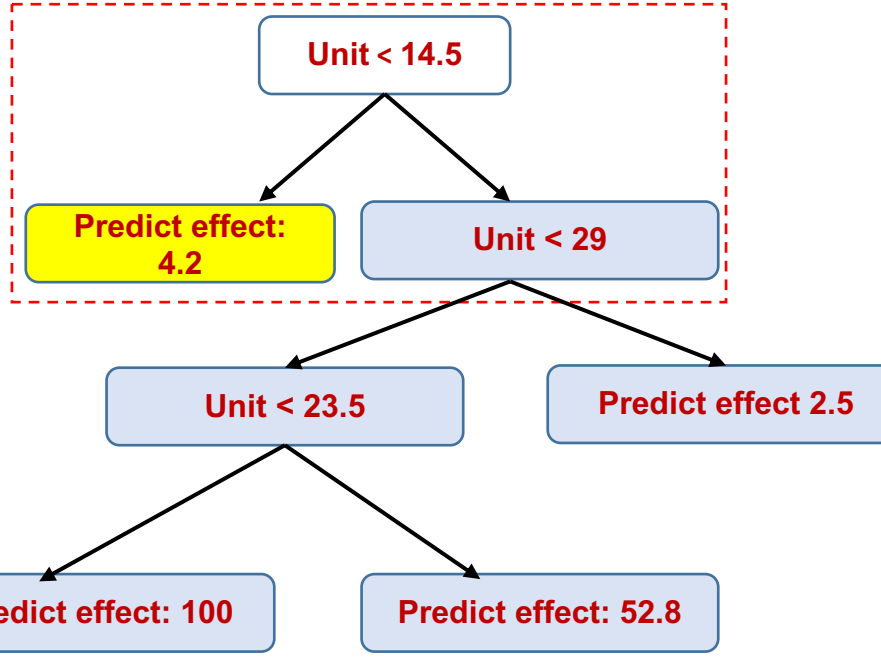
Tiêm 5 đơn vị vaccine

Hiệu quả vaccine: 44%

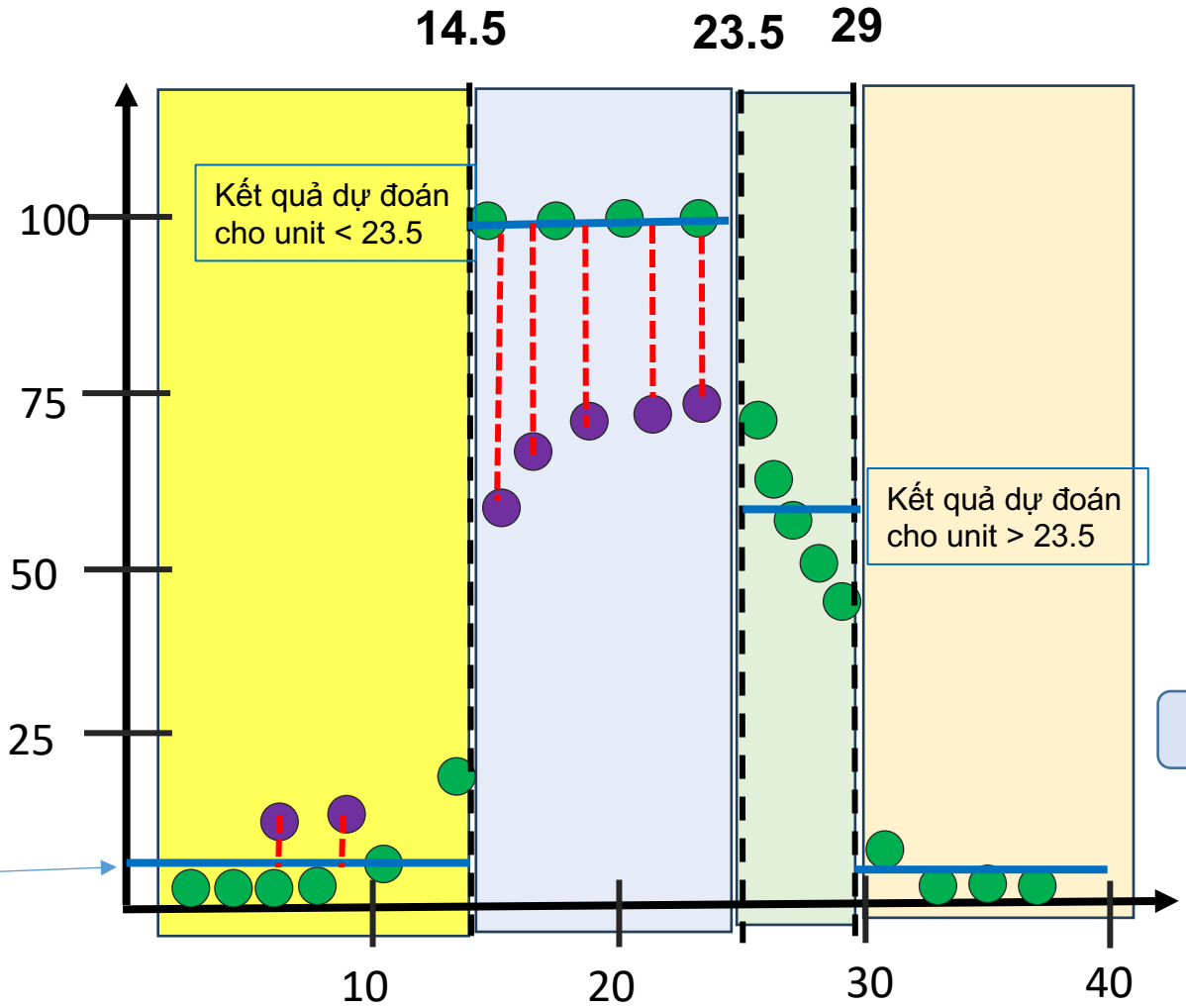
Khi có 1 vaccine ra đời, chúng ta muốn dự đoán xem nó hiệu quả bao nhiêu % ứng với từng liều lượng dùng trên bệnh nhân.

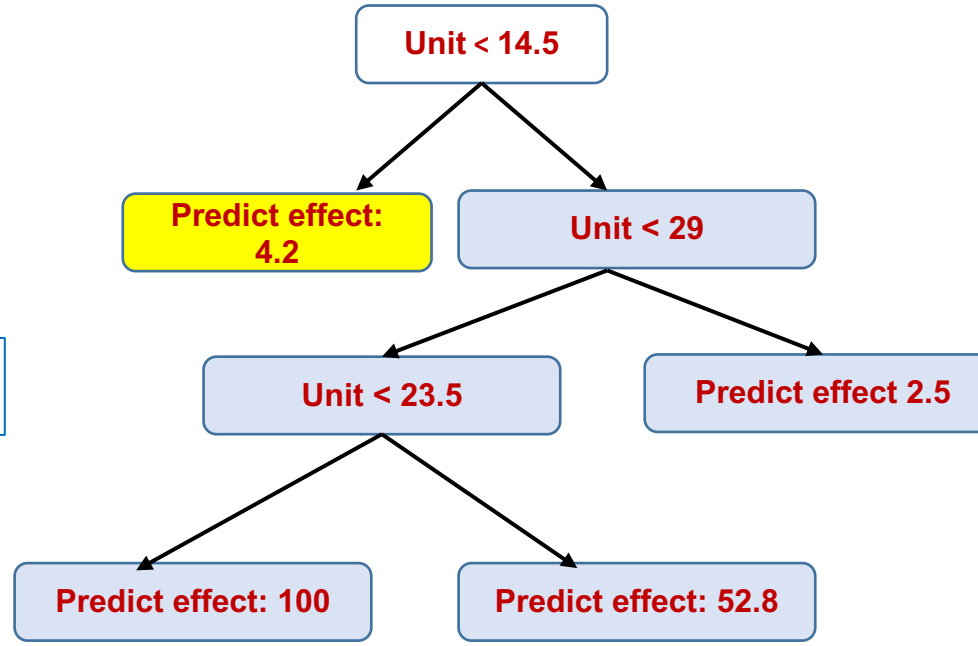
Decision Tree Review





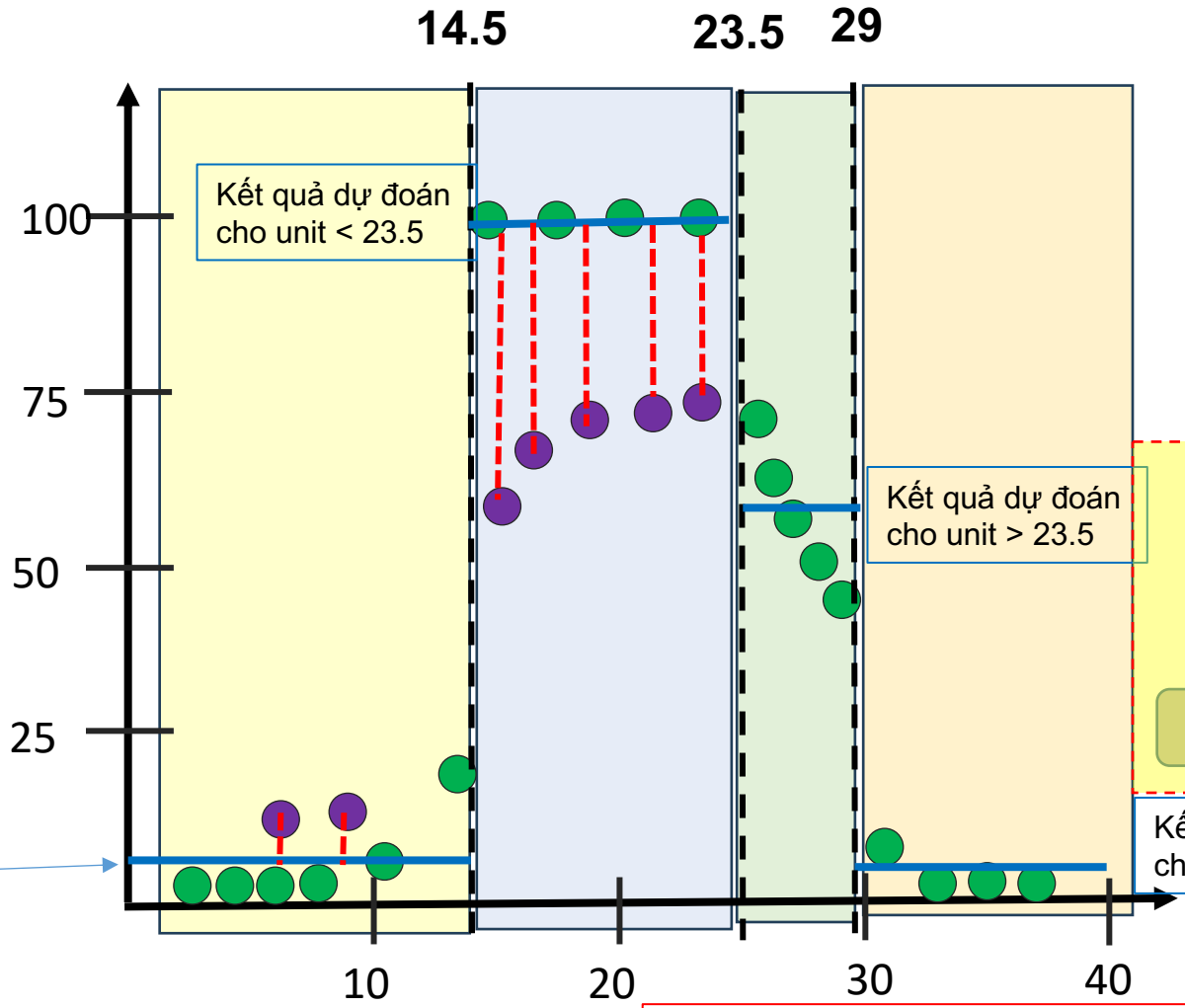
Decision Tree Review

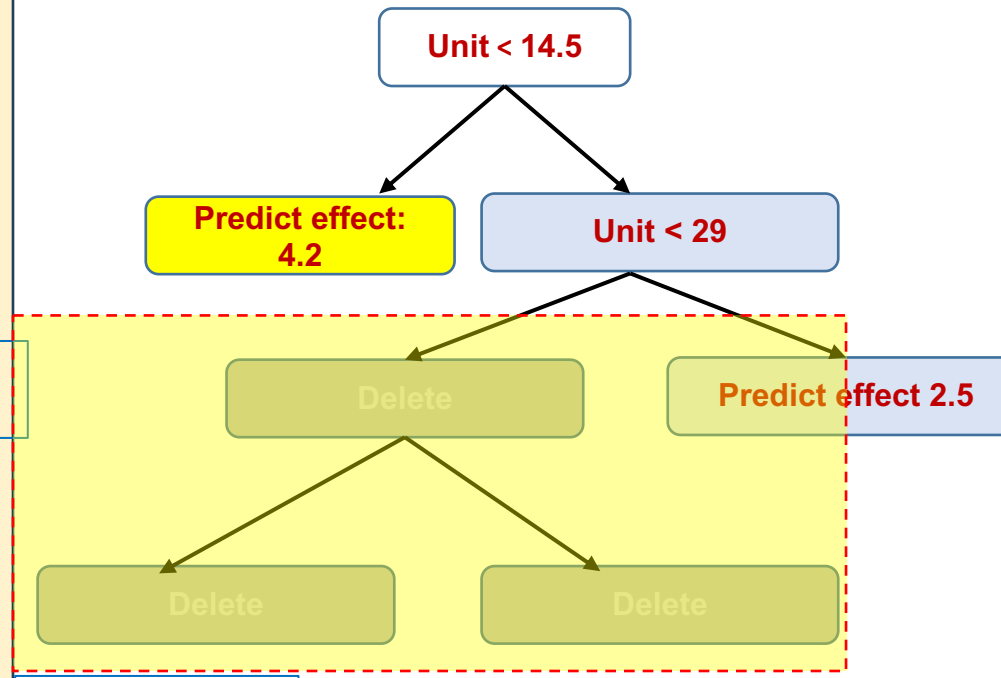




Dữ liệu test
Dữ liệu train
Error

Decision Tree Review





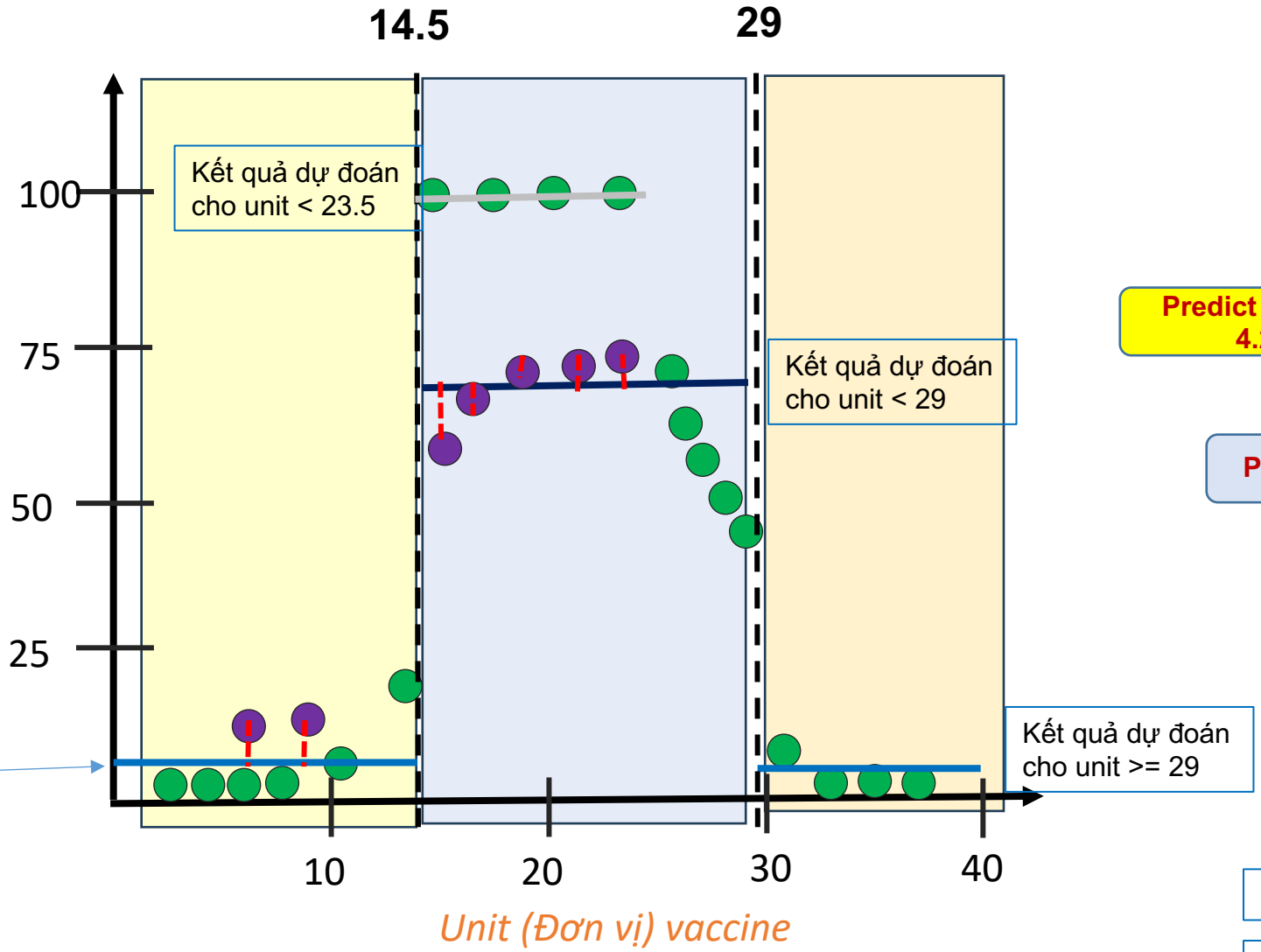
Dữ liệu test

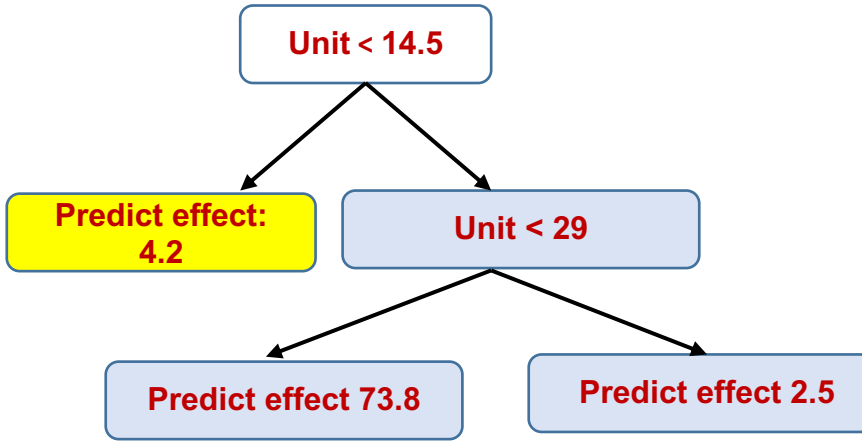
Dữ liệu train

Note : If we want to prune the tree more, we could remove last two leaves and replace the split with a leaf that is the average of all of the observations

Error

Decision Tree Review





Dữ liệu test

Dữ liệu train

Error

Tree complexity penalty

The tree complexity penalty compensates for the difference in the number of leaves.

Tree Score = sum of squared residual + αT

α (alpha) is a tuning parameter that we finding using cross validation.
T is the total number of terminal nodes/the total number of leaves

For now, let's let $\alpha = 10{,}000$ and calculate tree score for each tree.

How to select α

	α = 0	α = 10,000	α = 15000	α =20,000
Split 1	…	…	…	…
Split 2	…	…	…	…
Split 3	…	…	…	…
Split 4	…	…	…	…
Split 5	…	…	…	…
Average	50,000	5000	11,000	30,000

In this case, the optimal trees built with **α = 10,000** had, on average, the lowest sum of square residuals. So **α = 10,000** is our final value.

Decision Tree Review

Advantages

- ✓ Very easy to explain

(Bạn nghĩ có dễ hiểu hơn linear regression không?)

- ✓ More closely mirror human

(Bạn nghĩ sao về điều này?)

- ✓ Can easily handle qualitative predictors without the need of create dummy variables.

(Dummy variable là gì?)

Disadvantages

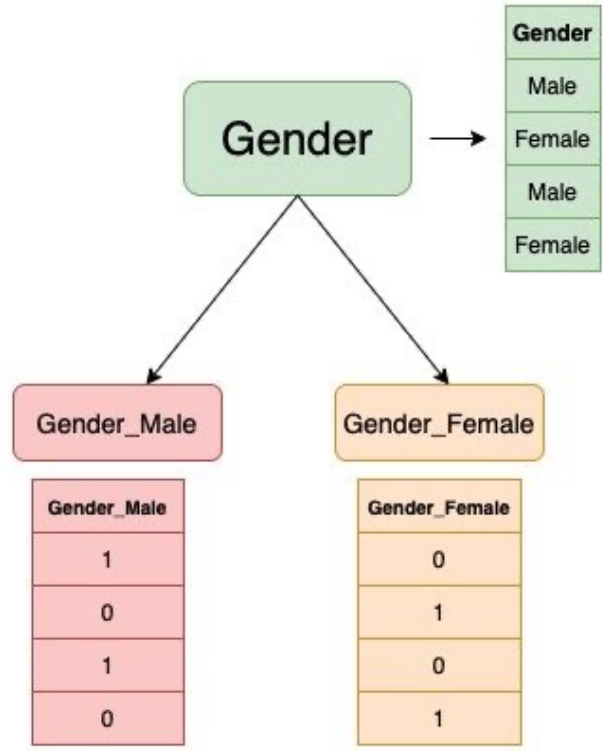
- ✓ Do not have the same level of predicting accuracy as some other regression and classification methods
- ✓ Small changes in the data can cause a large change in the large estimated tree.
- ✓ Are less effective in making predictions when the main goal is to predict the outcome of a continuous variable

Dummy Variable

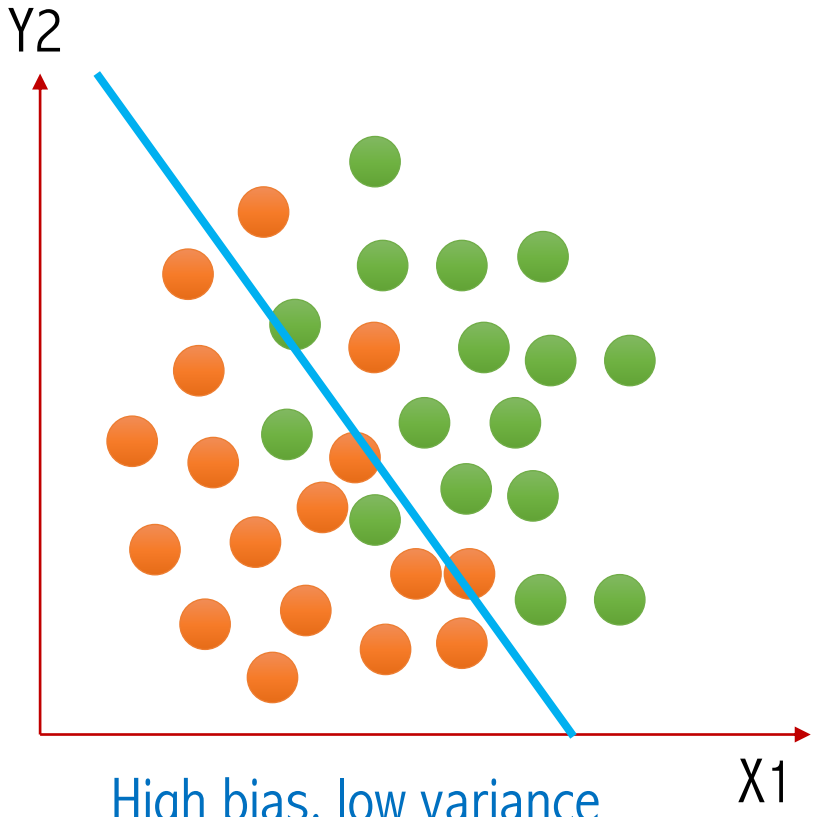
Water	Temperature
A	Hot
B	Cold
C	Warm
D	Cold

Dummy Variables

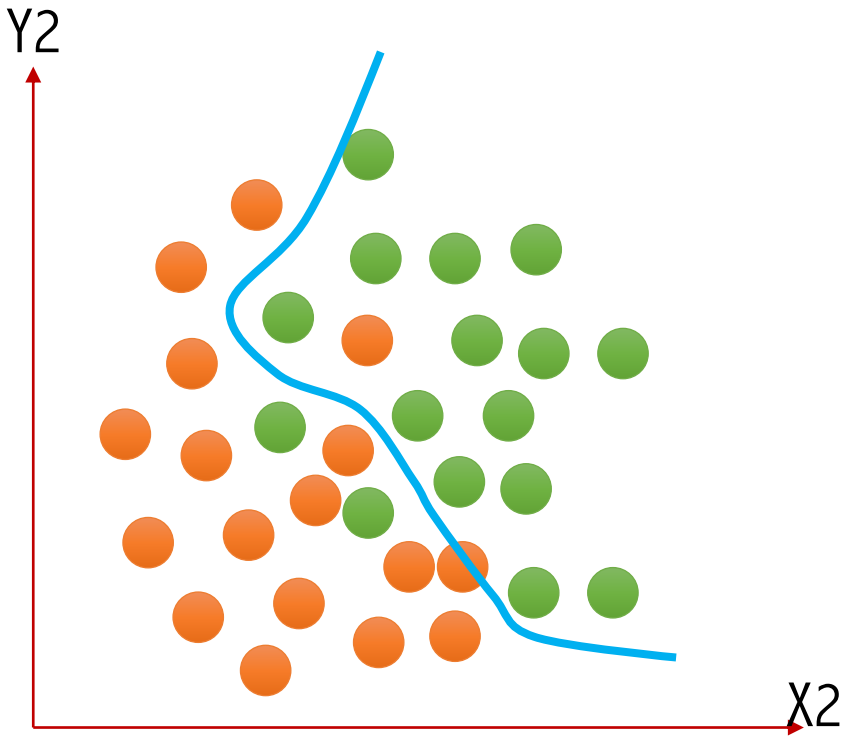
Water	Temperature	var_hot	var_warm	var_cold
A	Hot	1	0	0
B	Cold	0	0	1
C	Warm	0	1	0
D	Cold	1	0	0



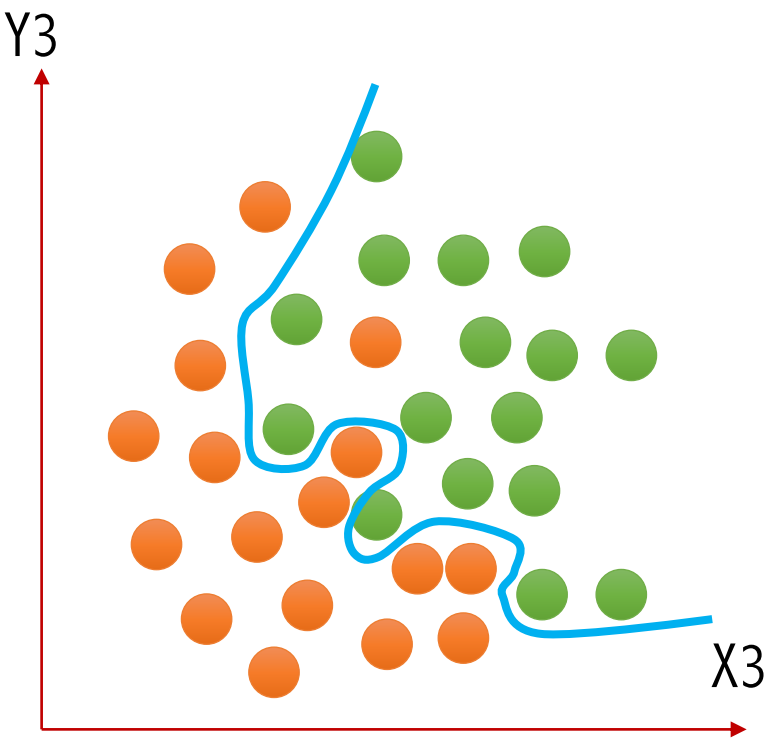
Bias-Variance Trade-off



High bias, low variance
(Underfitting)



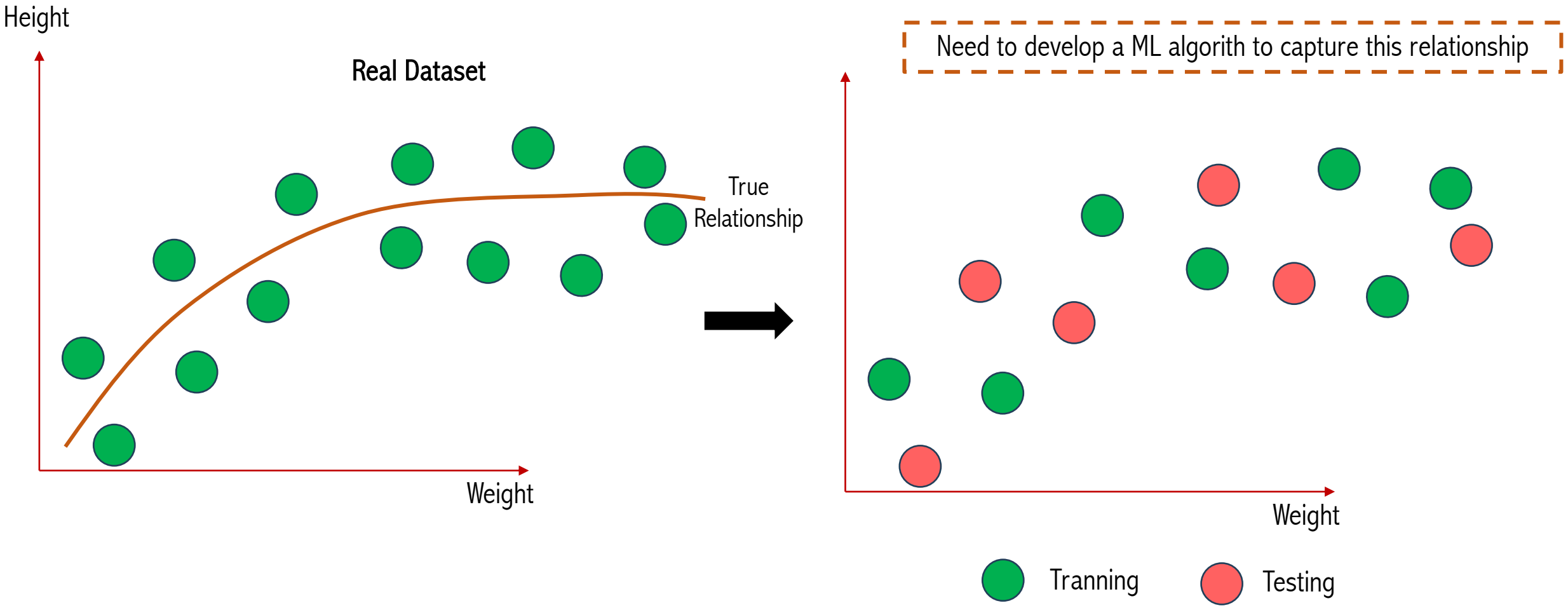
Low variance, low bias
(just right)



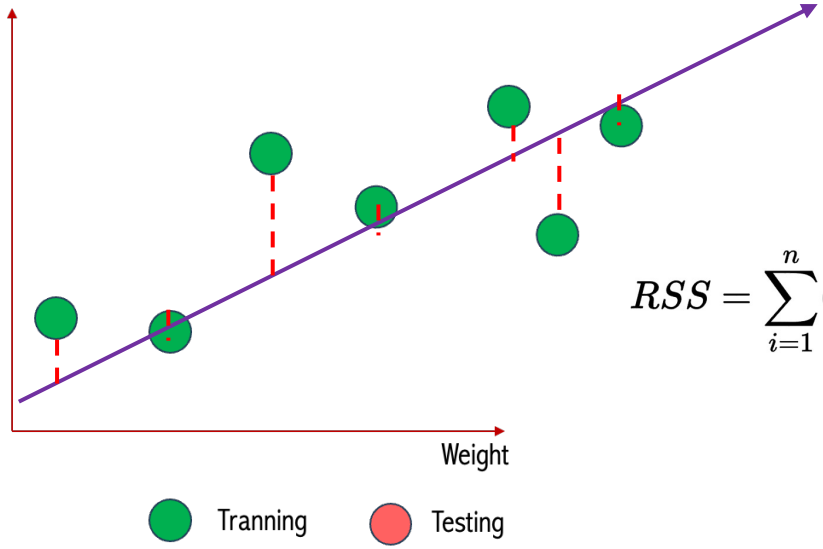
High variance, low bias
(overfitting)

Weak Learner

Bias-Variance Trade-off



Bias-Variance Trade-off



Linear Regression

$$RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2 \quad >> 0$$

Linear Regression will never capture the true relationship between weight and height

↓

The inability of machine learning to capture the true relationship is call **bias**

↓

Linear Regression has a high bias

Polynomial Regression

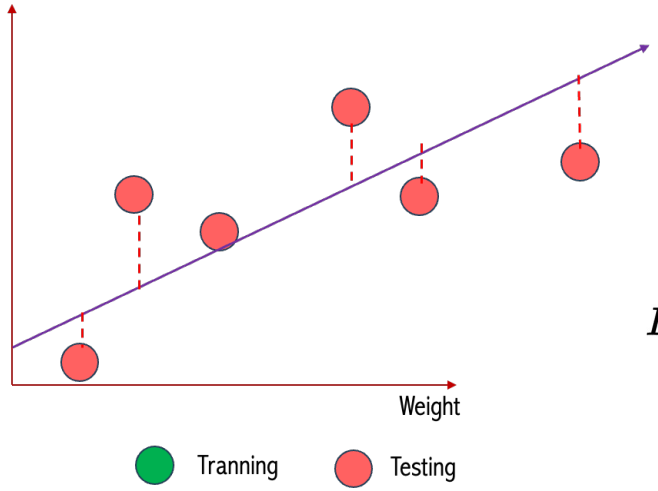
$$RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2 \quad \approx 0$$

Polynomial Regression can capture the true relationship between weight and height

↓

Polynomial Regression has a low bias

Bias-Variance Trade-off



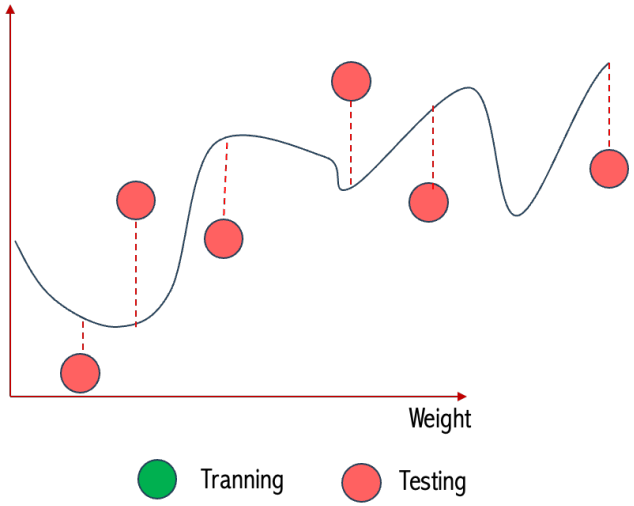
Linear Regression

$$RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2 >> 0$$

Linear Regression has a high bias because ...

Linear Regression has low variance because its SSR are very similar for diference dataesets

The different in fits between datatasets is call variance



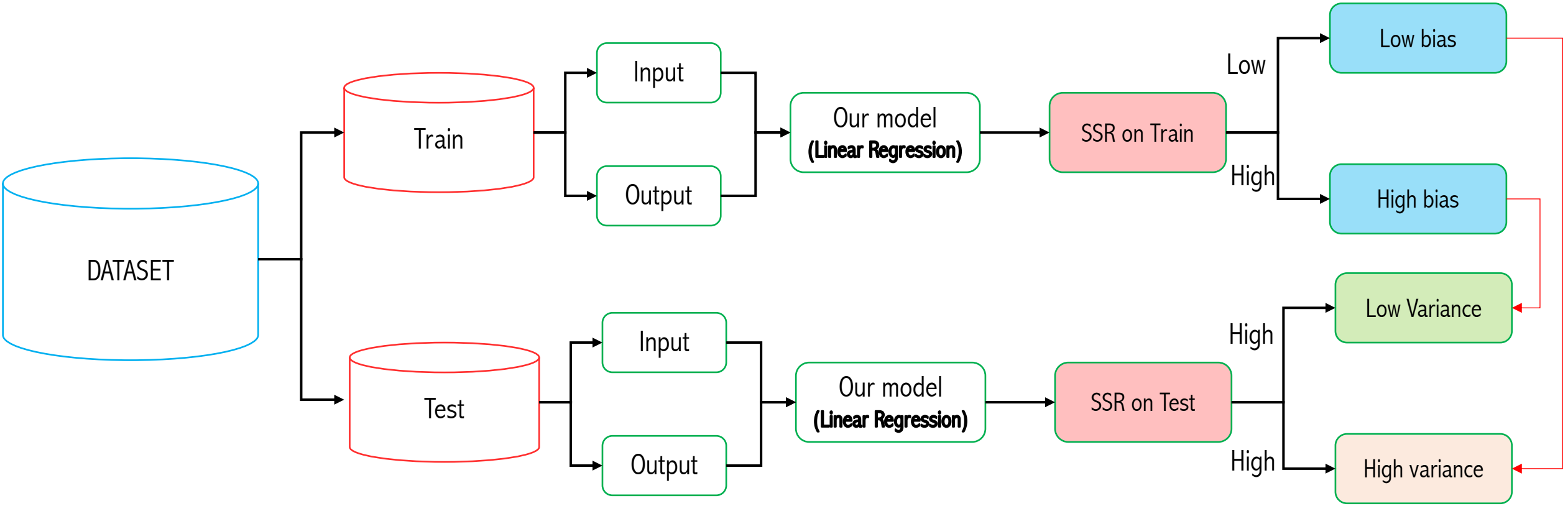
Polynomial Regression

$$RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2 >> 0$$

Polynomial Regression has a low bias because ...

Polynomial Regression has high variance because it returns in huge different in SSR between train and test dataset

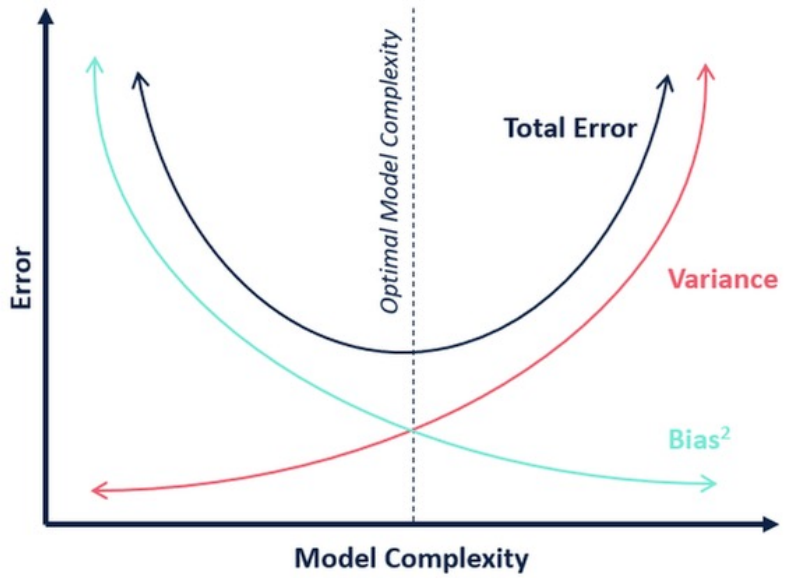
Bias-Variance Trade-off

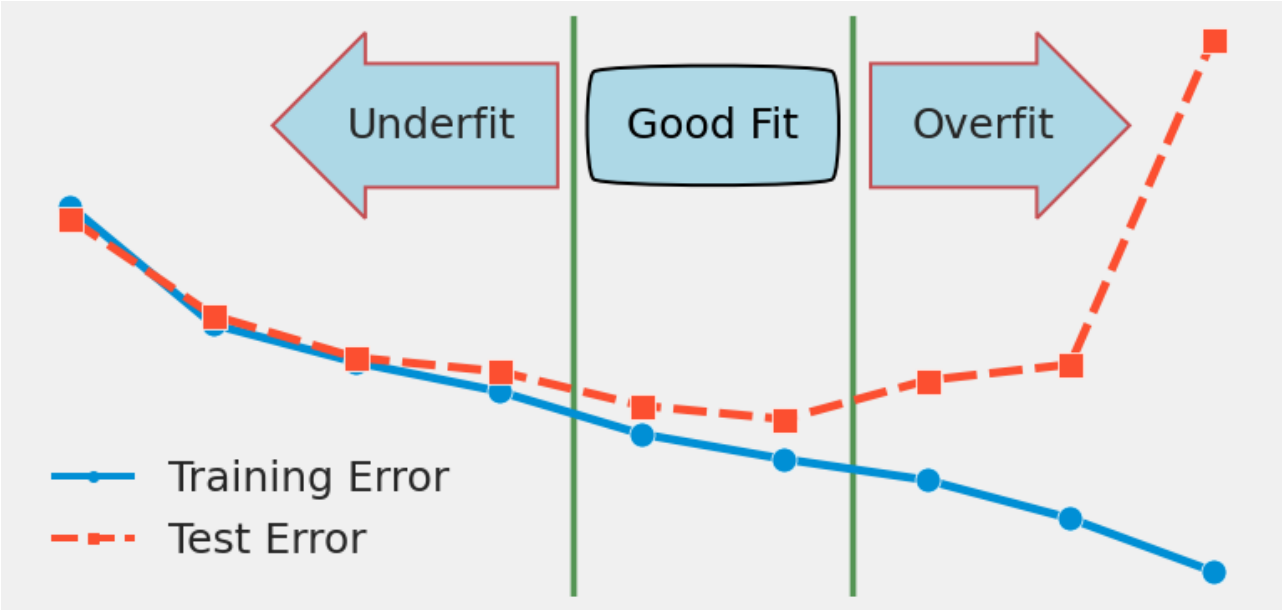


Bias as the error rate of the training data.

The different in fits between datatasets is call variance

Prediction errors (bias and variance)





Bias as the error rate of the training data.

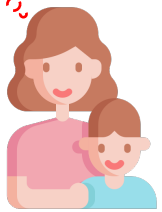
The different in fits between datatasets is call variance

Random Forest Motivation

You want to buy a perfume for your girlfriend(s)?

What would you do?

1 Channel đi con!



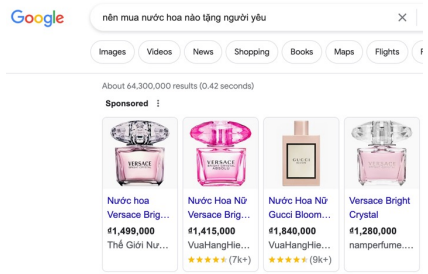
2 Channel đi bạn!!

Ask for Idea!

3 Cucci đi!!

4 Cucci ạ!!

5 Search: Cucci

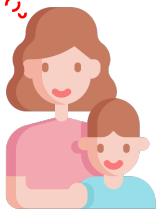


Random Forest Motivation

You want to buy a perfume for your girlfriend(s)?

What would you do?

1 Channel đi con!



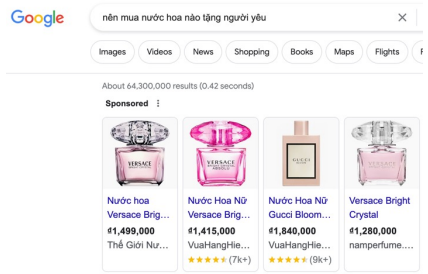
2 Channel đi bạn!!

Mua Gucci Thôi!!!

3 Cucci đi!!

4 Cucci ạ!!

5 Search: Cucci

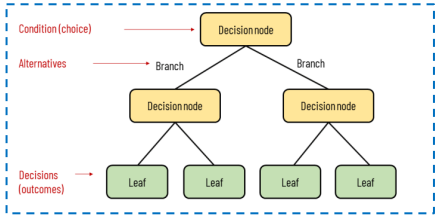


Random Forest Motivation

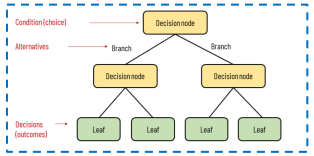
You want to buy a perfume for your girlfriend(s)?

What would you do?

1 Channel đi con!

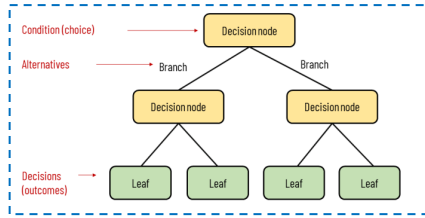


2 Channel đi bạn!!

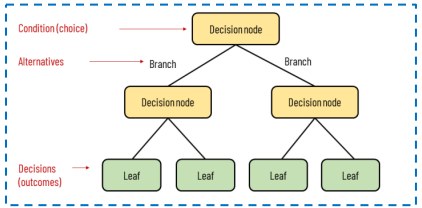


Mua Gucci Thôi!!!

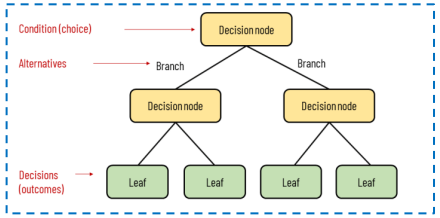
3 Cucci đi!!



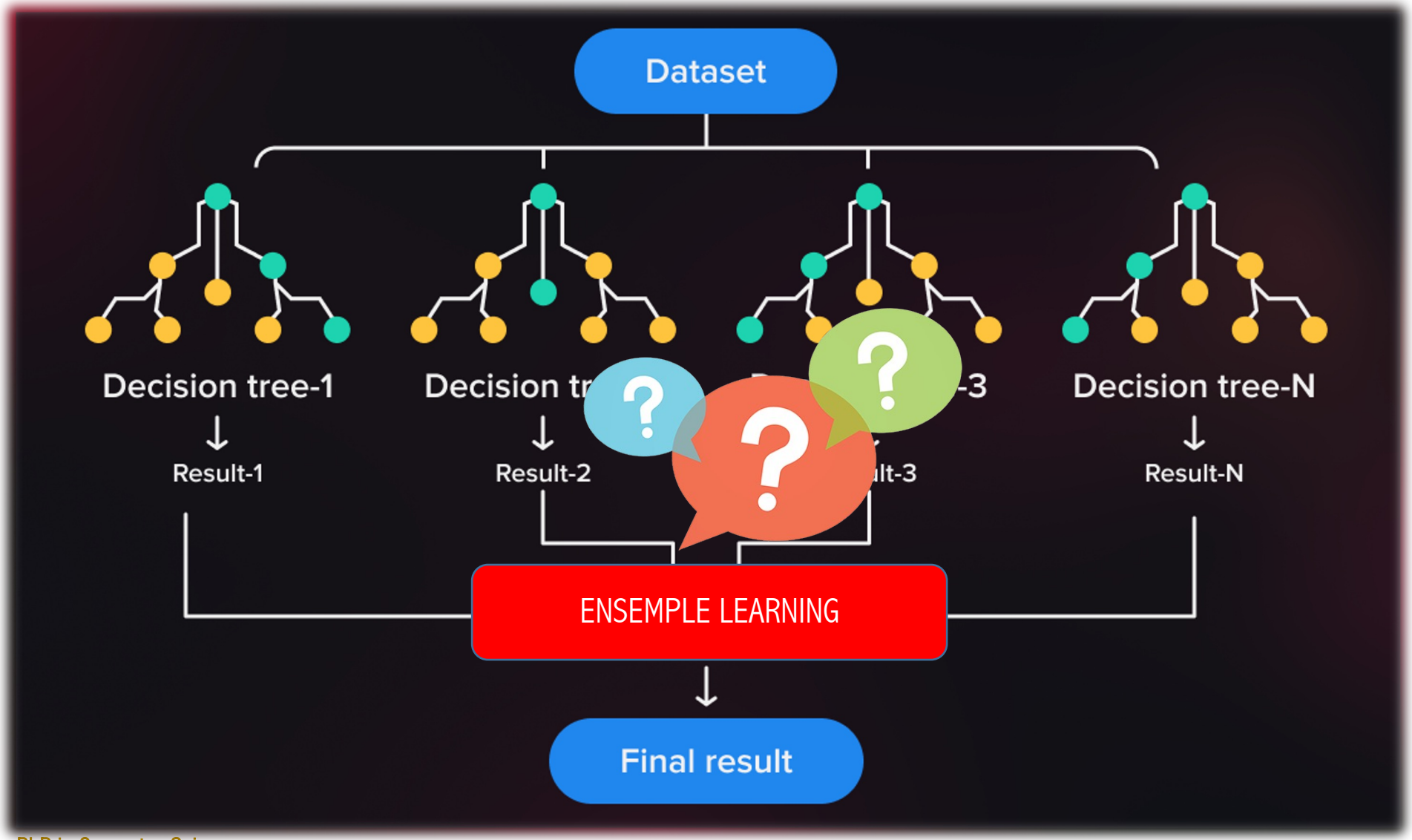
4 Cucci ạ!!



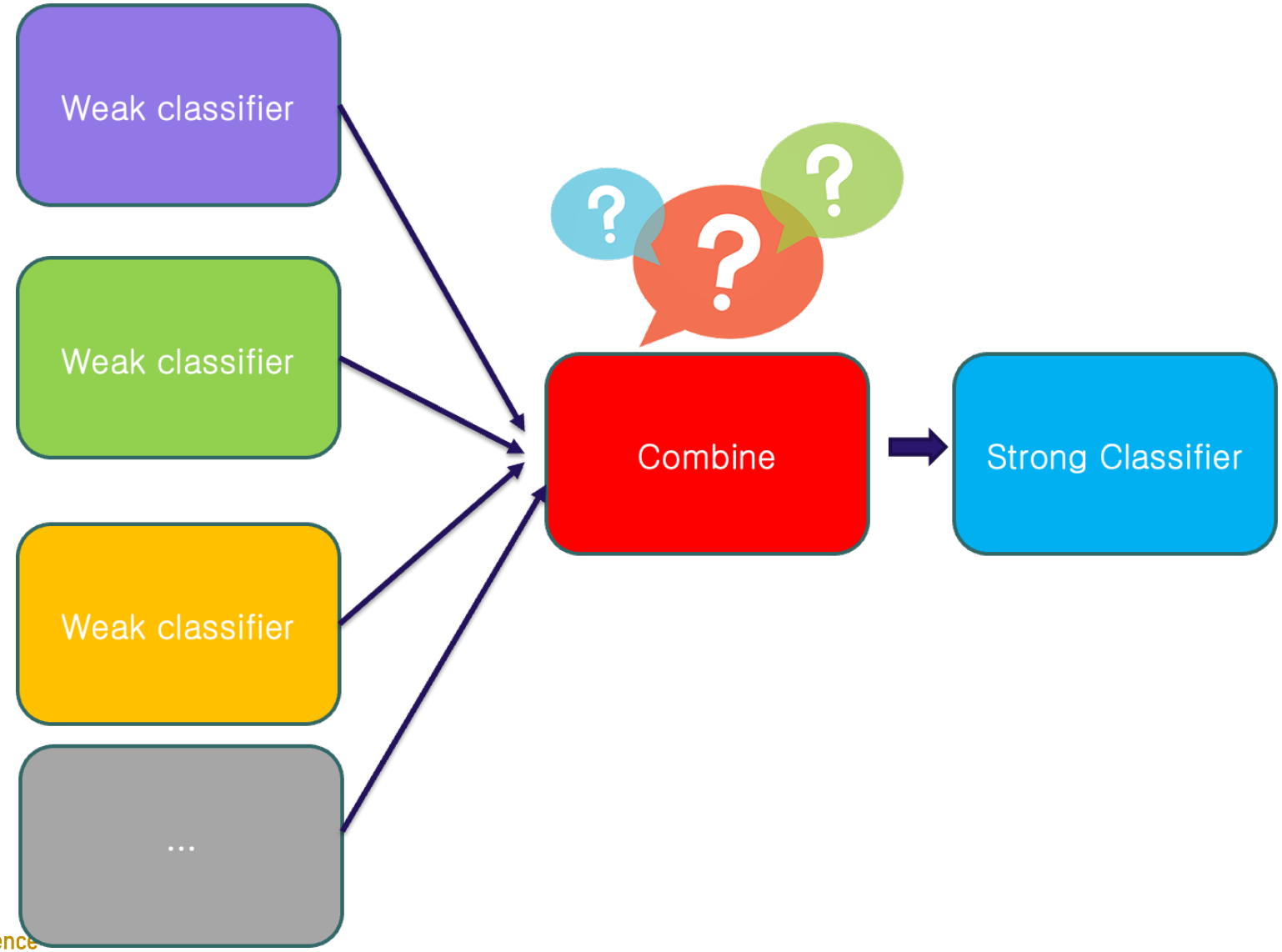
5 Search: Cucci



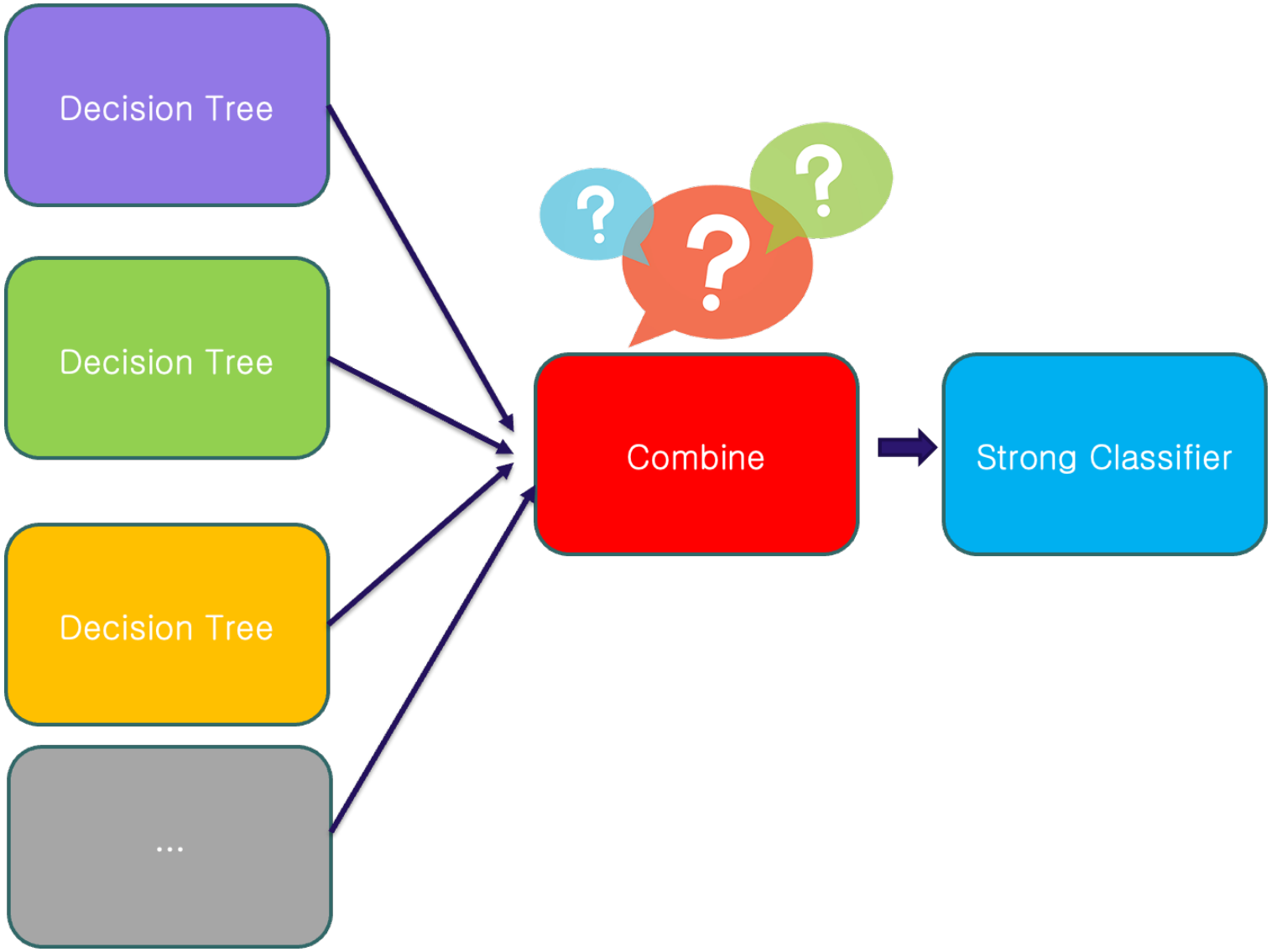
Random Forest Motivation



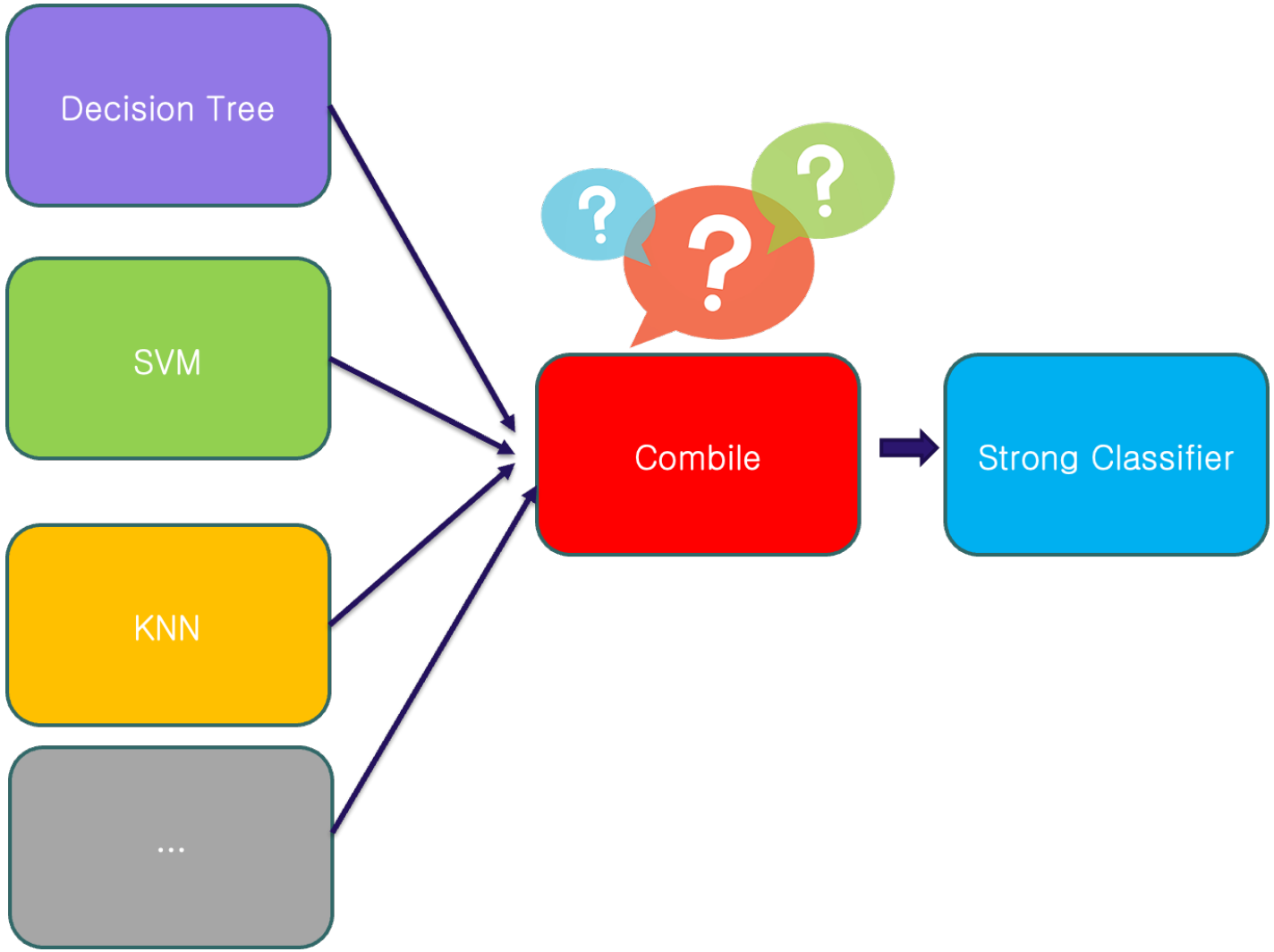
What is an Ensemple Learning?



Homogeneous Approach



Heterogeneous Approach



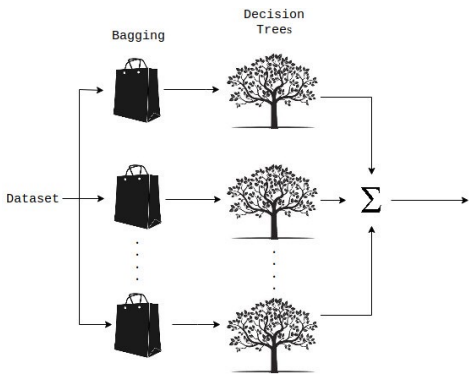
Ensemple Learning Techniques

Ensemple Learning

Thông dụng ở các cuộc thi về AI

Bagging
homogeneous weak learners

Random Forest



Boosting
homogeneous weak learners

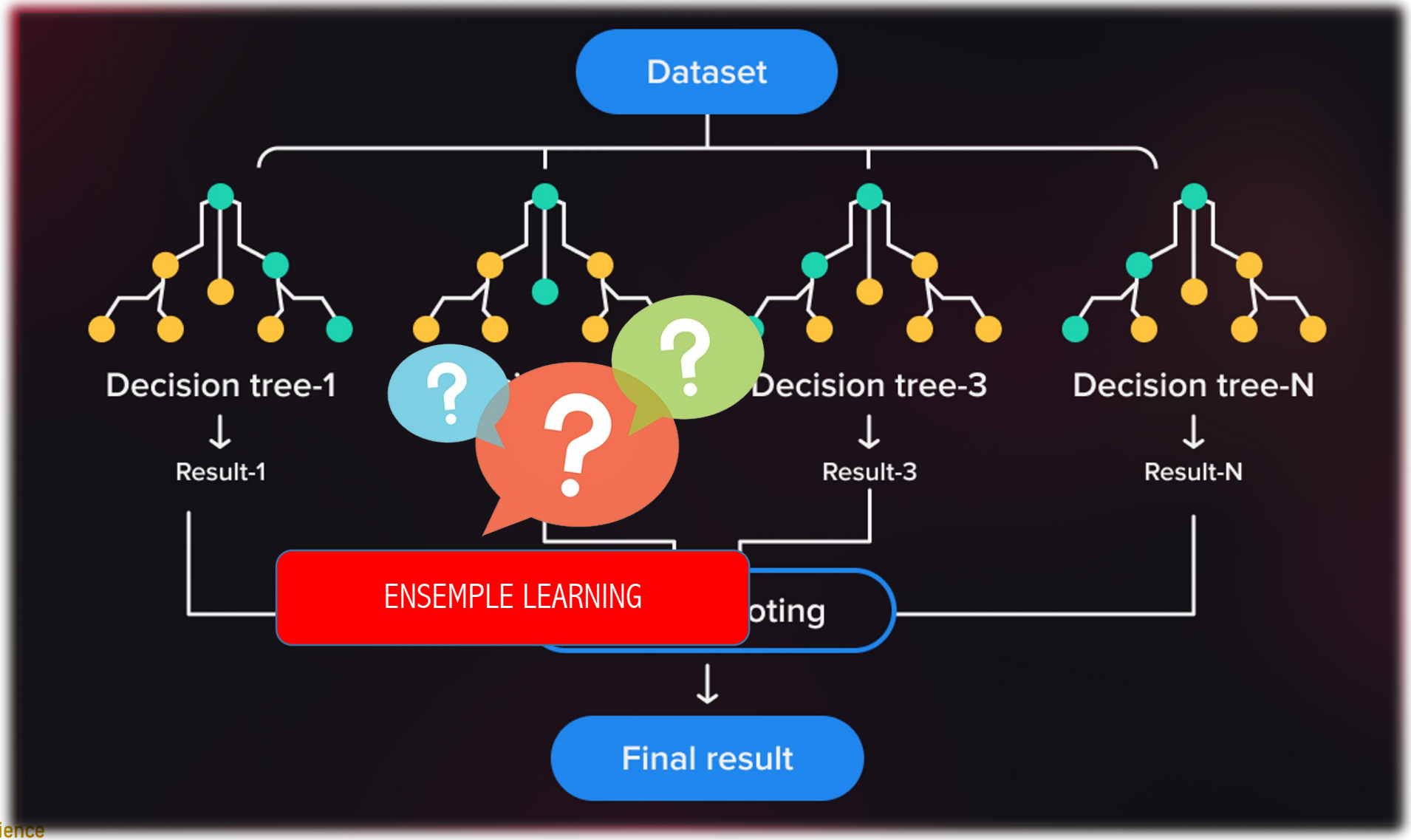
NEXT WEEK

Stacking
Heterogeneous weak learners

OUT OF SCOPE

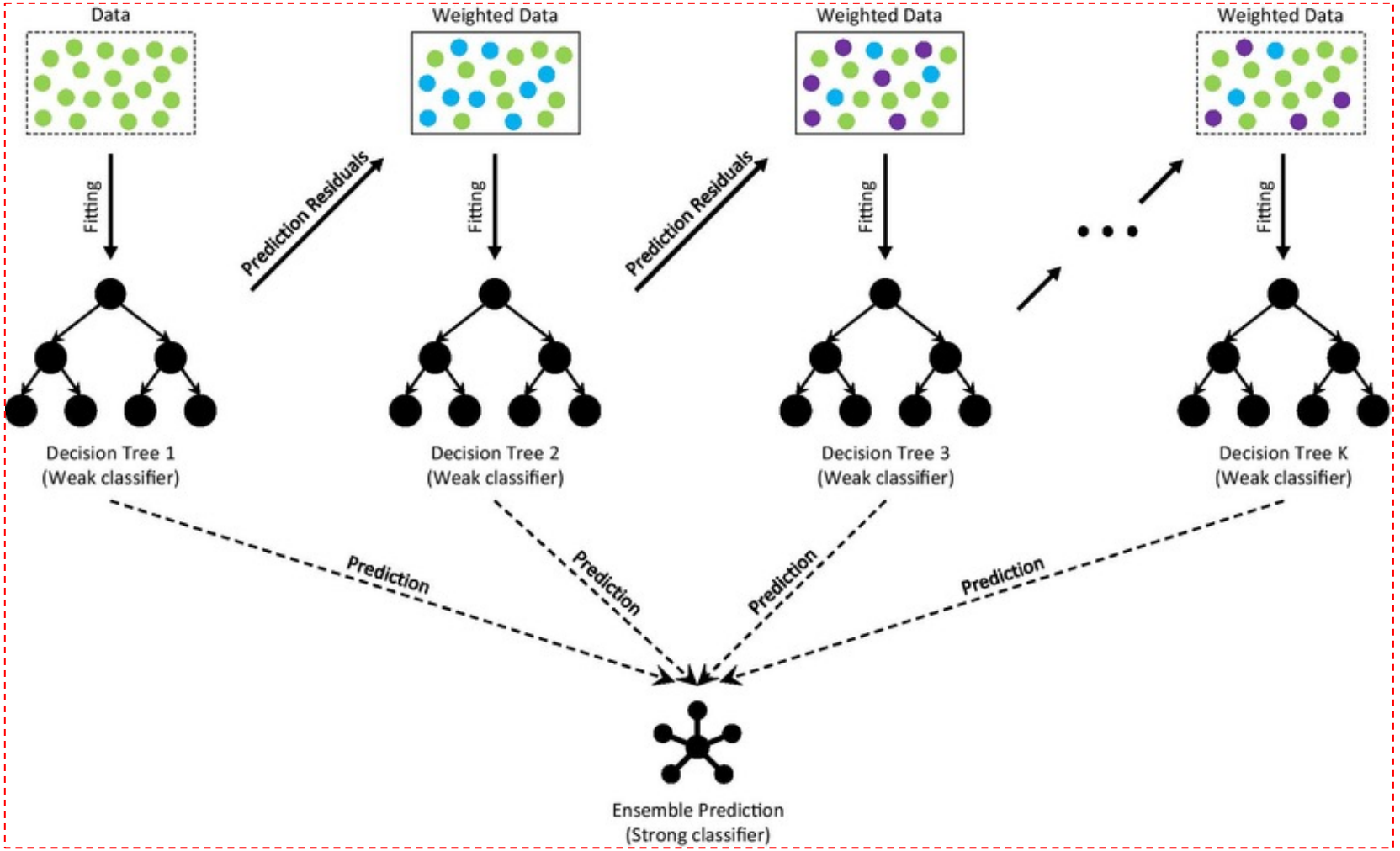
Bagging-based Method

Random Forest

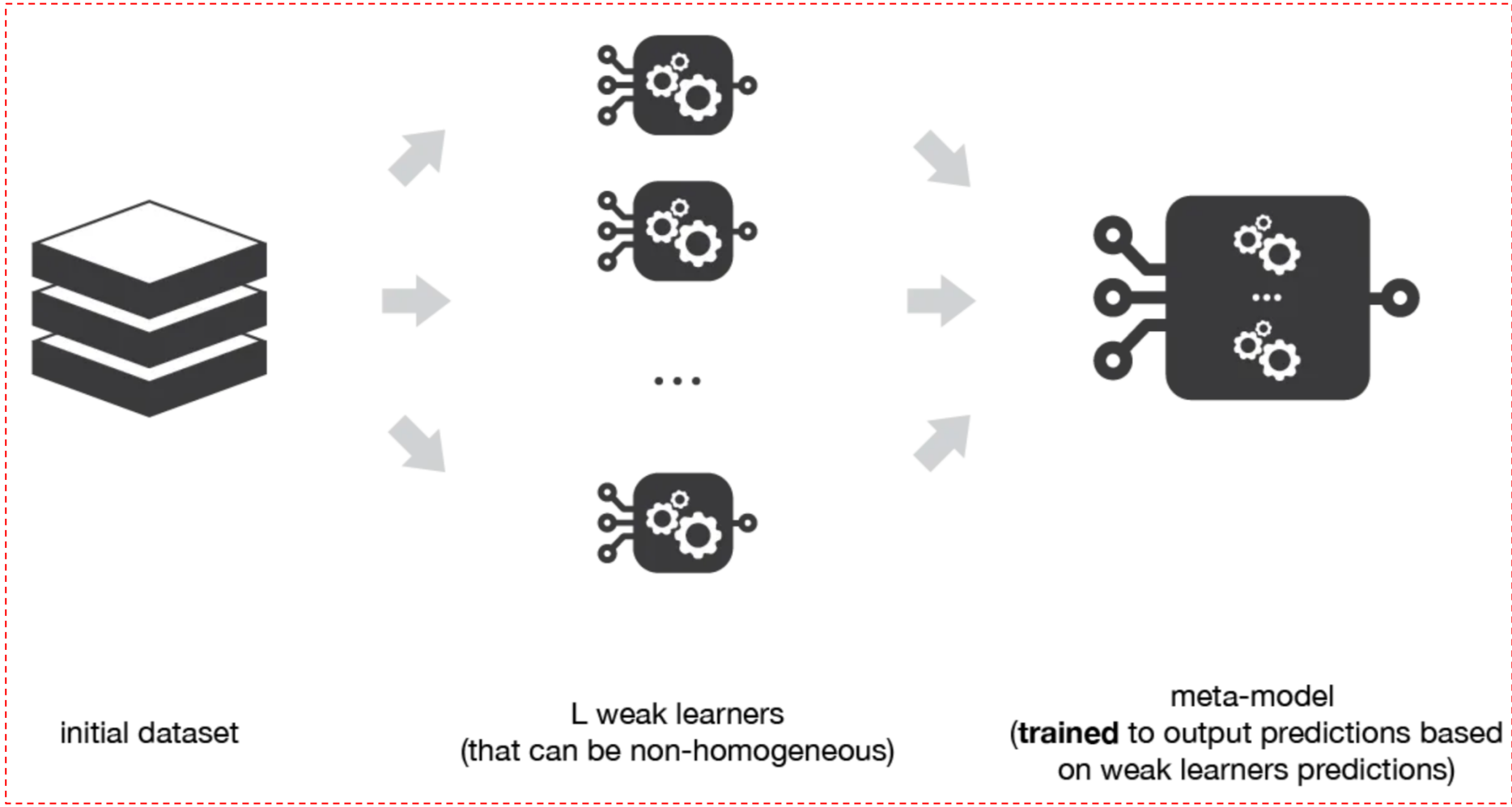


Boosting-Based Method

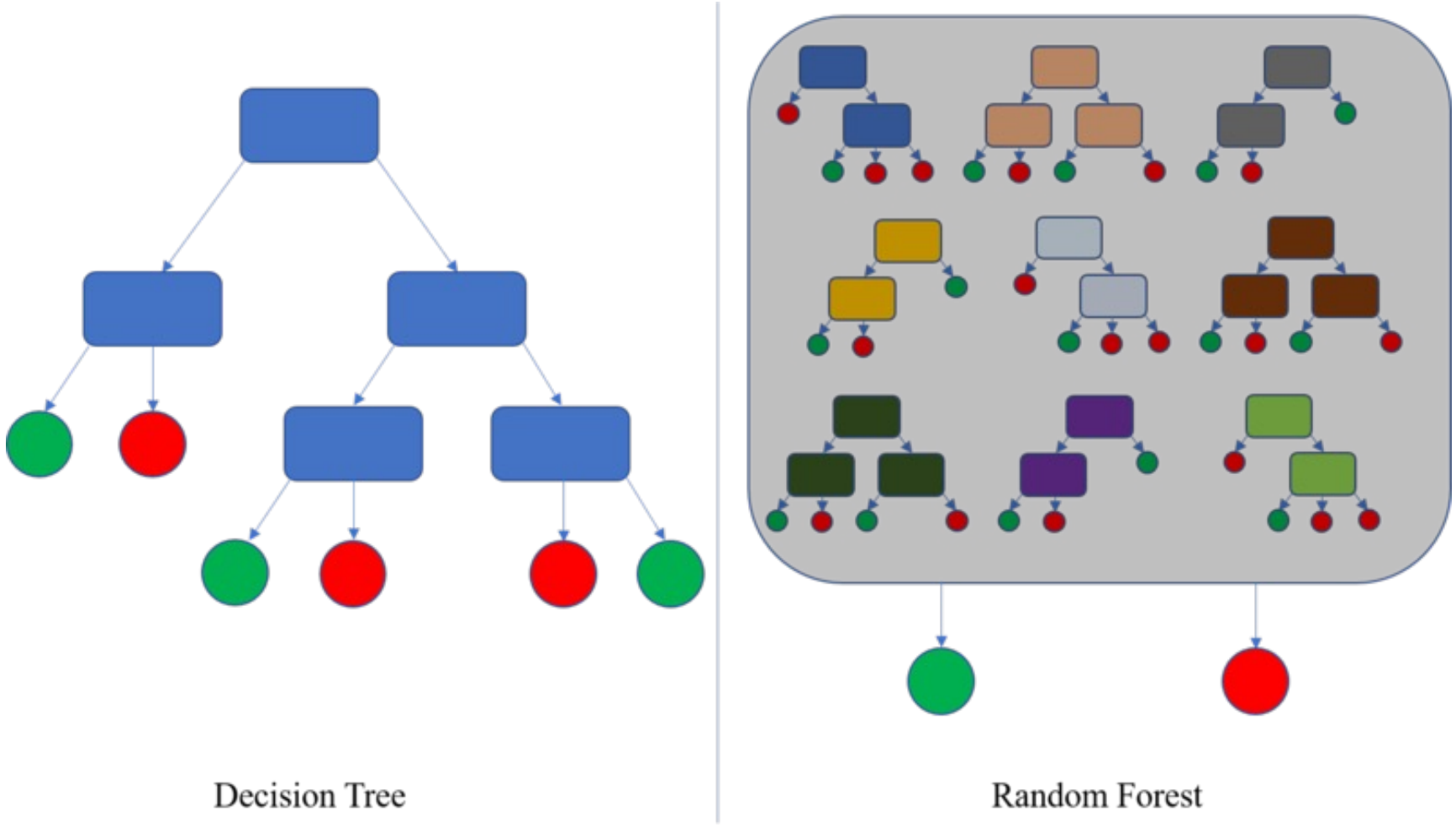
NEXT WEEK



Stacking-Based Method



Decision Tree vs Random Forest

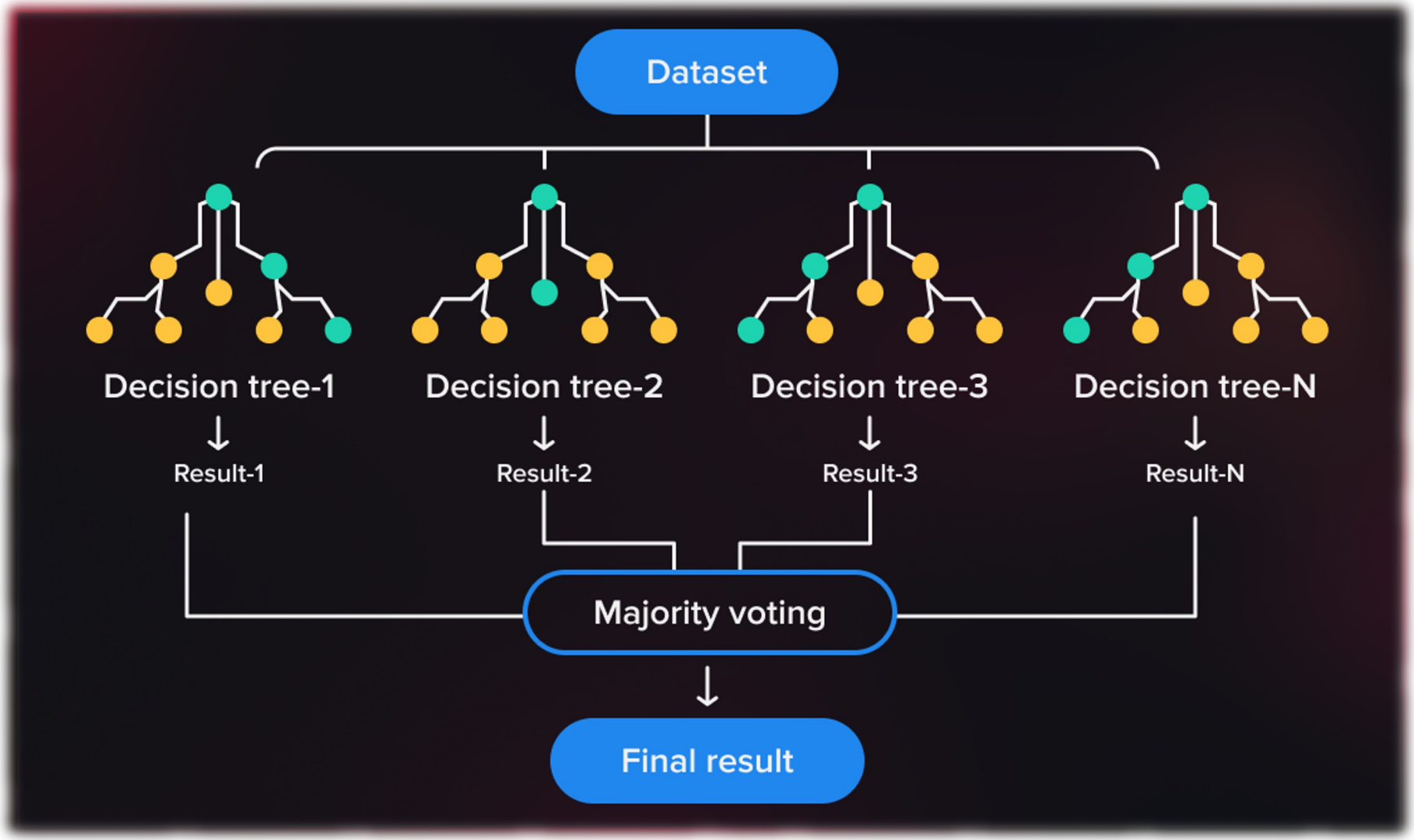


https://commons.wikimedia.org/wiki/File:Decision_Tree_vs._Random_Forest.png

Outline

- Decision Tree Review
- Random Forest
- Fill in missing data with Random Forest
- Case study

RANDOM FOREST IS A SOLUTION



Step to Random Rorest

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

1st Step: Create a New 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

Original DATA

Chọn lựa ngẫu nhiên từ dataset ban đầu

New DATA

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

1st Step: Create a New 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

Original DATA

Chọn lựa ngẫu nhiên từ dataset ban đầu

Bootrapped Dataset

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

2nd Step: Decision Tree from Boostrapped Dataset

GENERATE DECISION TREES FROM THE BOOTSTRAPPED DATASET USING **PREDEFINED CONDITIONS**

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

A RANDOM SUBSET OF 2 ATTRIBUTES (OR 2 COLUMNS).

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

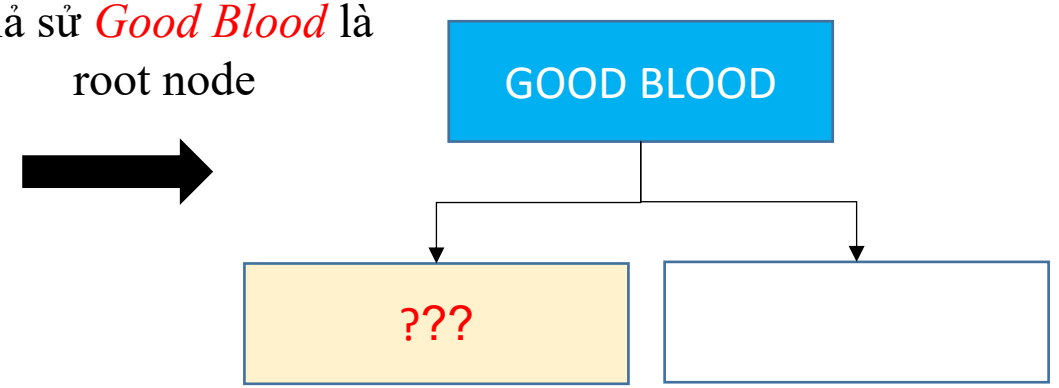
Traditional Tree

Tree with Predefined Conditions

Chọn lựa ngẫu nhiên 2 features (columns)

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

Giả sử *Good Blood* là root node



Loại bỏ Good Blood ra khỏi dataset

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

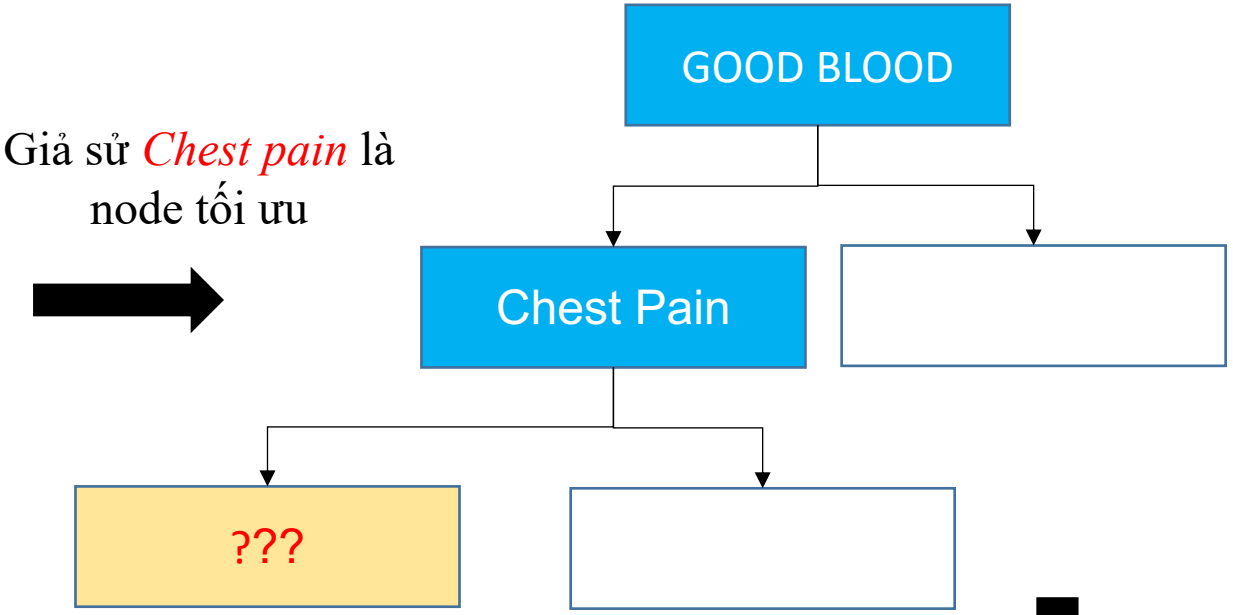
Chọn lựa ngẫu nhiên 2 features (columns)

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

Chọn lựa ngẫu nhiên 2 features (columns)

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

Giả sử *Chest pain* là node tối ưu



Chọn lựa ngẫu nhiên 2 features (columns)

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

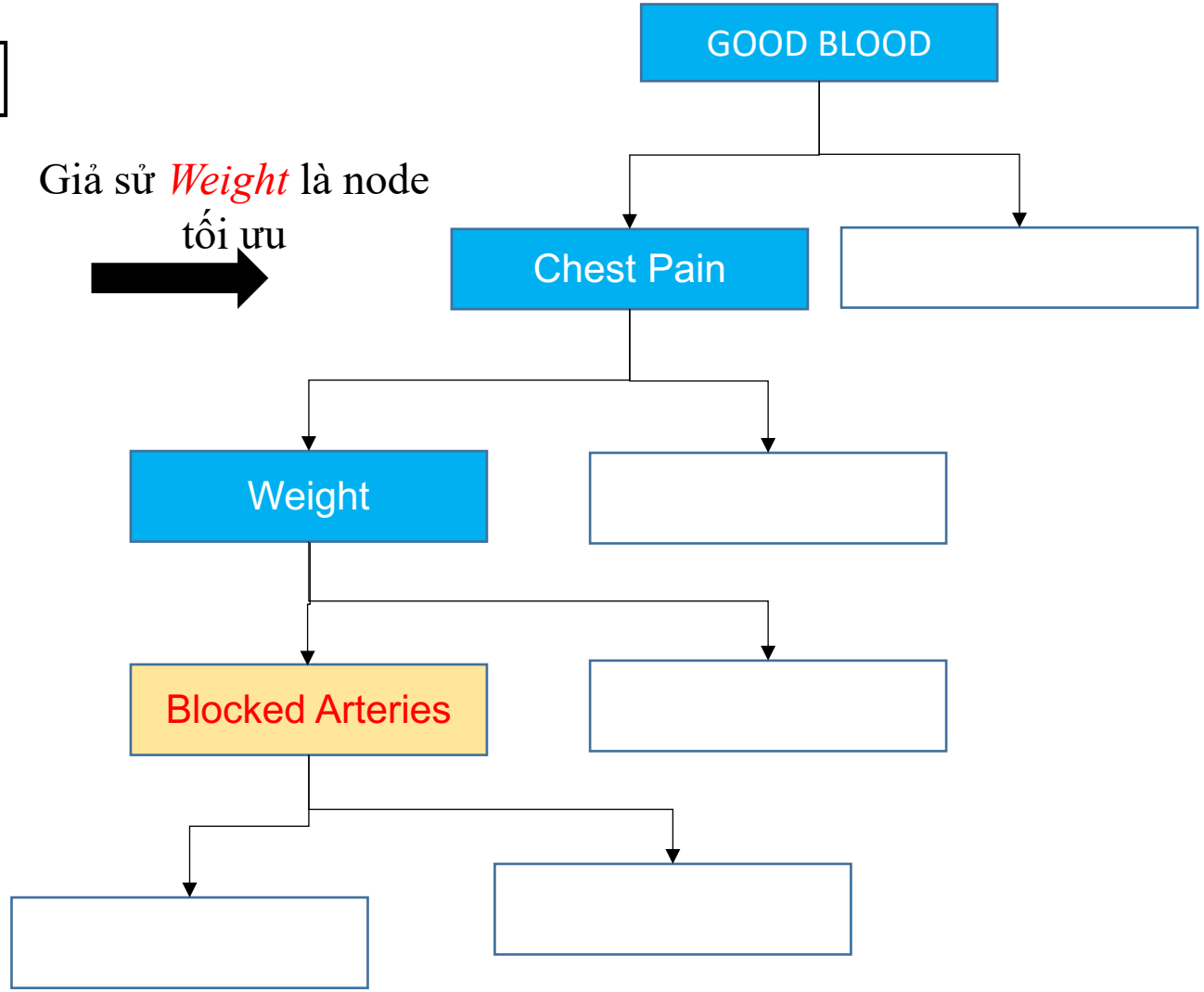
Loại bỏ *chest pain* ra khỏi dataset

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

Chọn lựa ngẫu nhiên 2 features (columns)

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

Giả sử *Weight* là node tối ưu

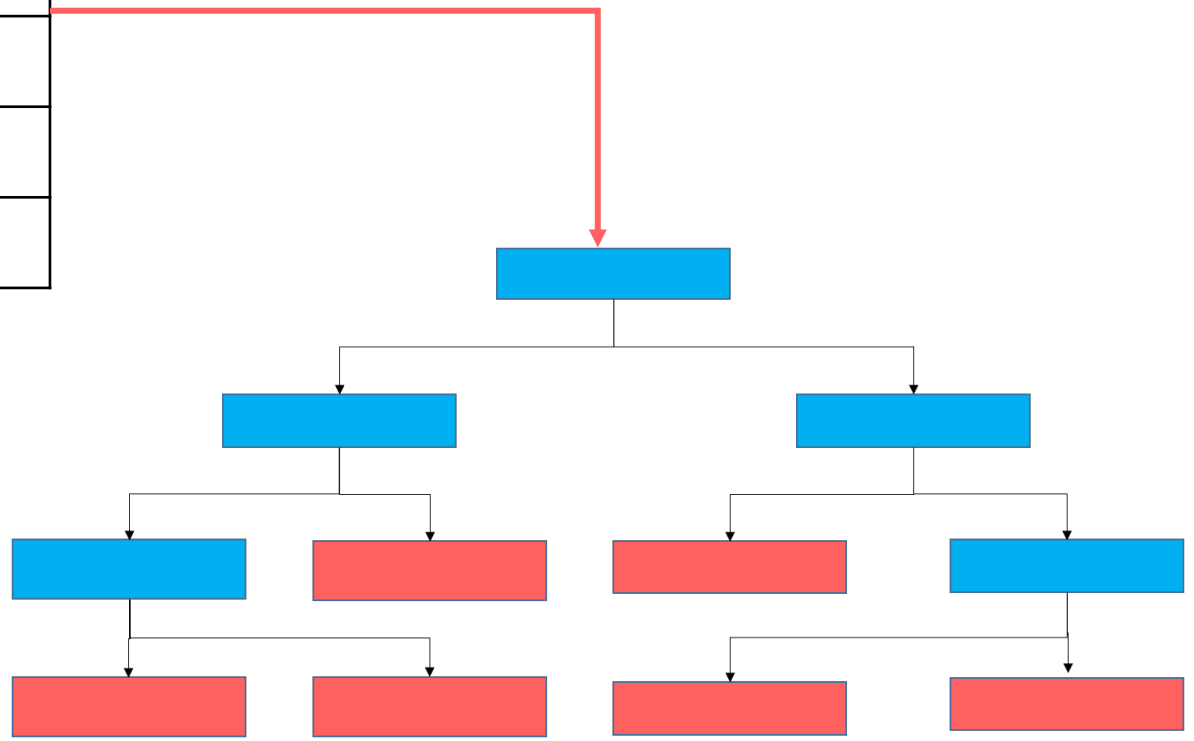


Loại bỏ *Weight* ra khỏi dataset

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

1st Decision Tree

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



Create N Tree

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

Generate

1st bootstrapped dataset

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

2nd bootstrapped dataset

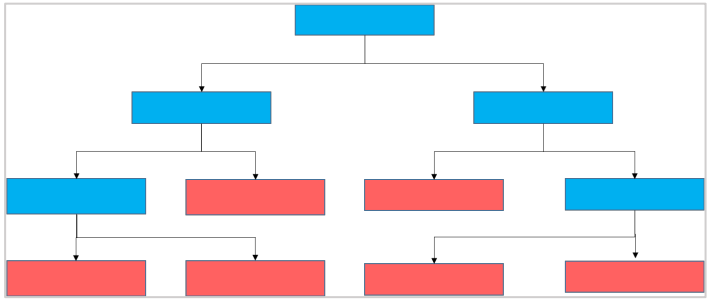
CHEST PAIN	GOOD BLOOD CIRCULATION	BLOCKED ARTERIES	WEIGHT	HEART DISEASE
YES	YES	YES	180	YES
NO	NO	NO	125	NO
YES	NO	YES	167	YES
YES	YES	NO	210	NO

...

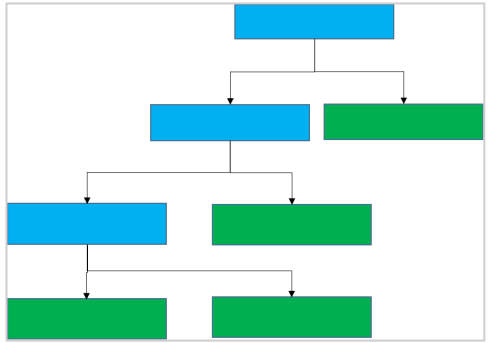
nth bootstrapped dataset

CHEST PAIN	GOOD BLOOD CIRCULATION	BLOCKED ARTERIES	WEIGHT	HEART DISEASE
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

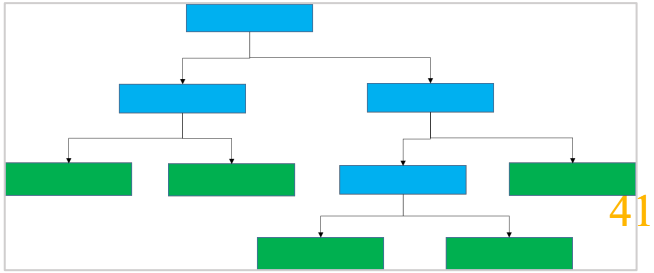
1st tree



2nd tree



nth tree



Create N Tree

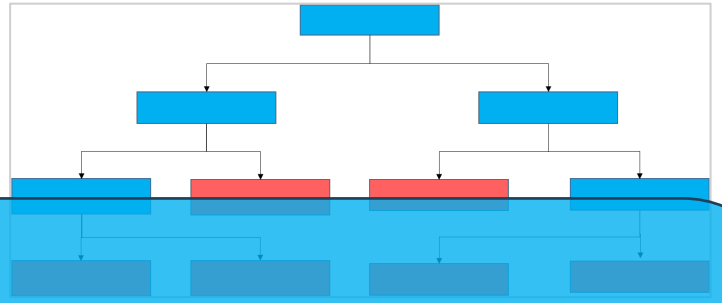
CHEST PAIN	GOOD BLOOD CIRCULATION	BLOCKED ARTERIES	WEIGHT	HEART DISEASE
NO	NO	NO	125	NO
YES	YES	YES	180	YES
YES	YES	NO	210	NO
YES	NO	YES	167	YES

Generate

1st bootstrapped dataset

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

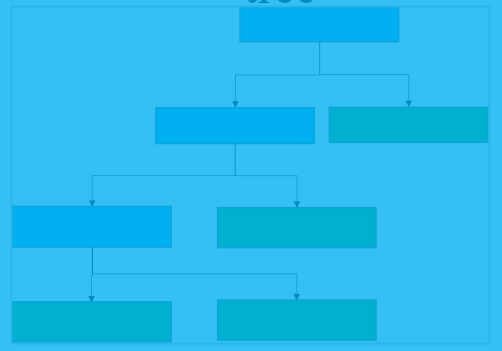
1st tree



2nd bootstrapped dataset

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

2nd tree

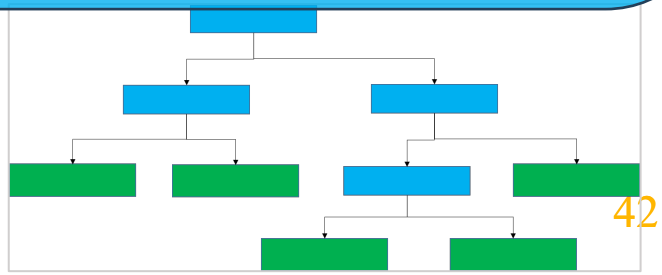


...

nth bootstrapped dataset

CHEST PAIN	GOOD BLOOD CIRCULATION	BLOCKED ARTERIES	WEIGHT	HEART DISEASE
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

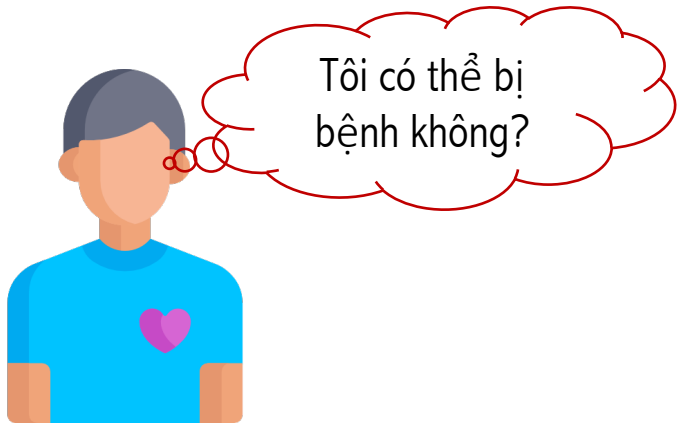
nth tree

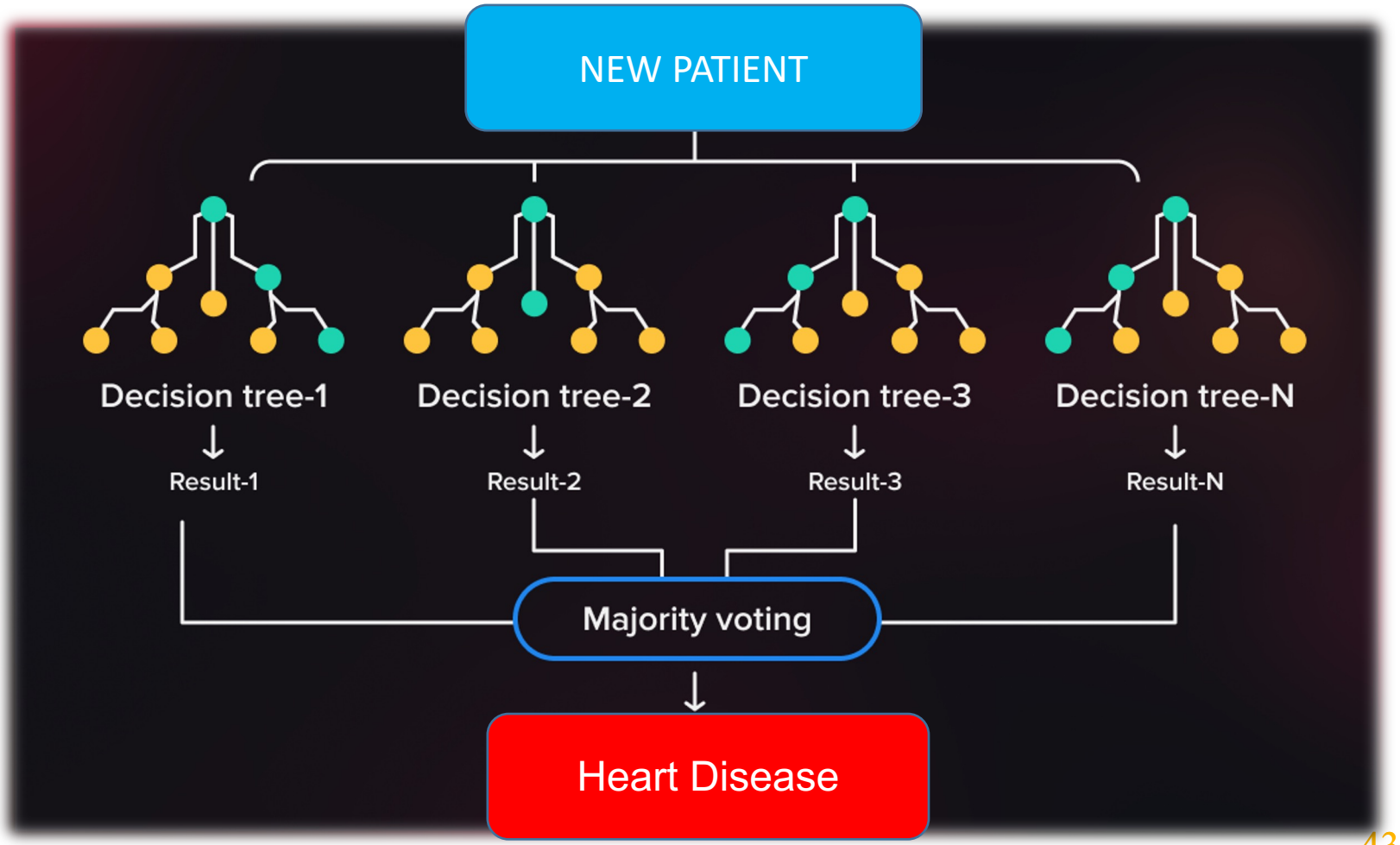


Random Forest

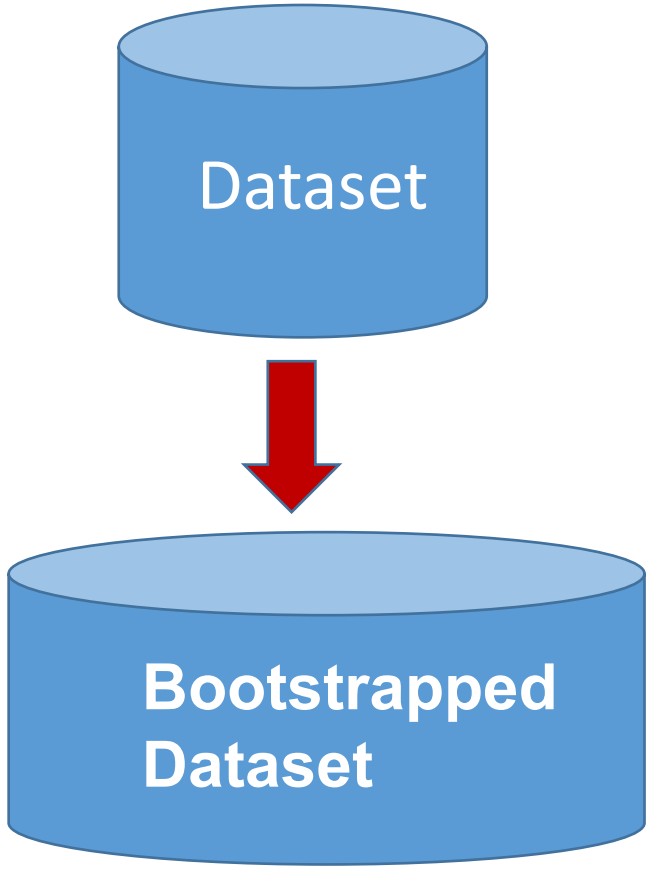
How to Predict New Sample

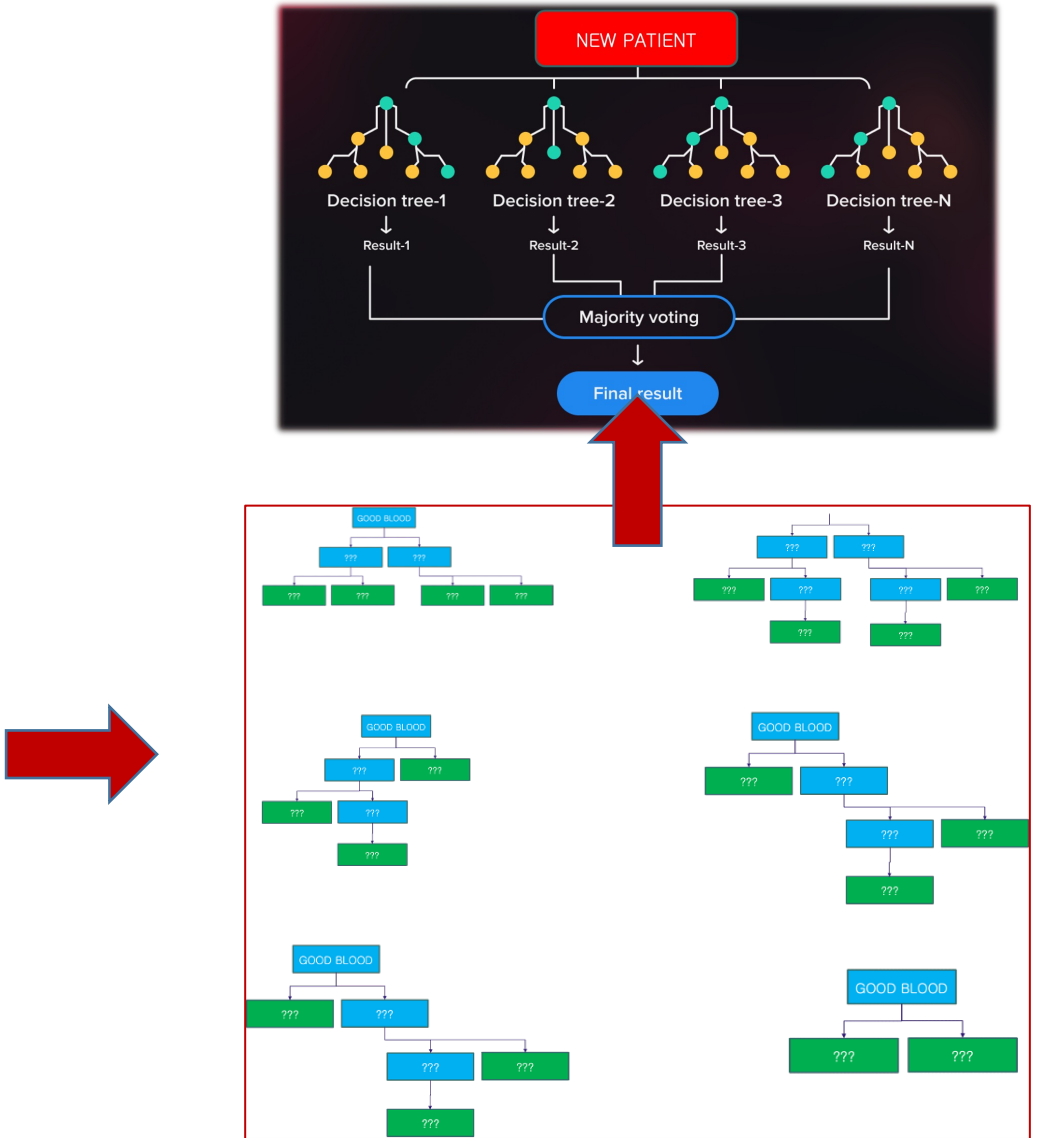
Chest Pain	No
GOOD BLOOD CIRCULATION	No
BLOCKED ARTERIES	No
Weight	125





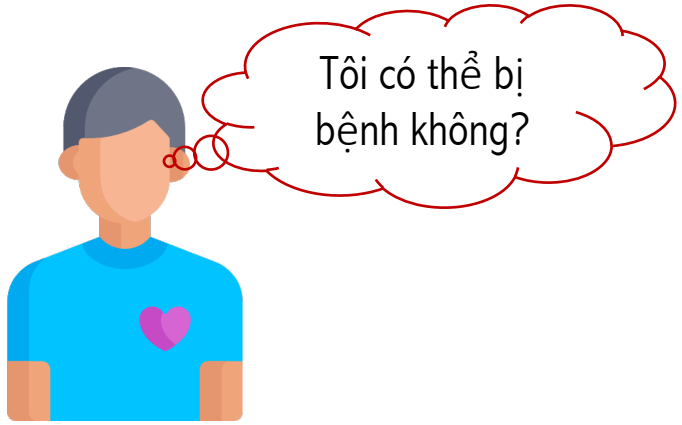
Bagging Technique

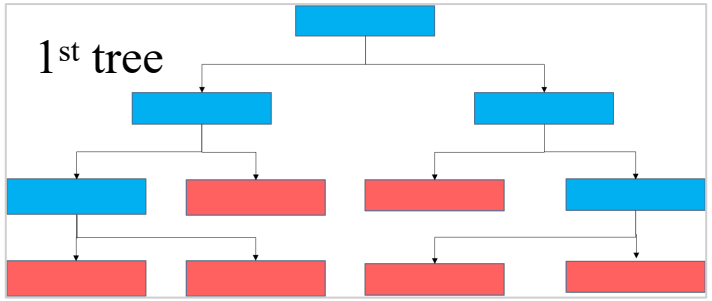




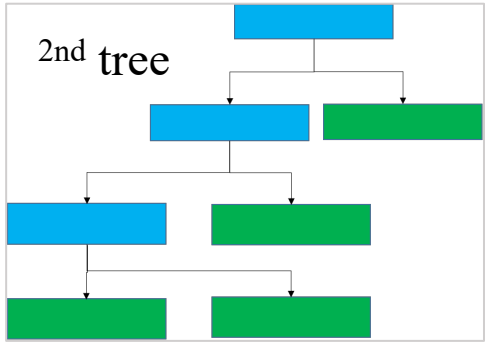
How to Predict New Sample

Chest Pain	No
GOOD BLOOD CIRCULATION	No
BLOCKED ARTERIES	No
Weight	125

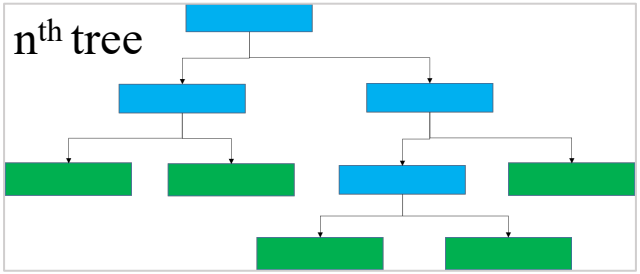




Predict



Predict



Predict

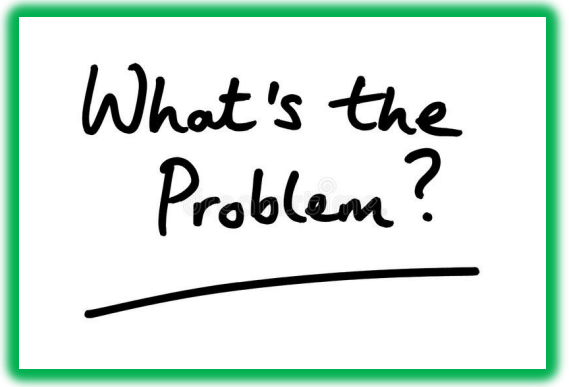
Heart Disease

Yes	No
7	2

Review

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

ORIGINAL DATA



RANDOMLY SELECT DATA

BOOTSTRAPPED DATASET

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

ALLOW DUPLICATED VALUES

Review

Original 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

Bootstrapped Dataset

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

Mộp phần của dataset ban đầu có thể không có mặt ở Bootstrapped dataset

Out-of-bag 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

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

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

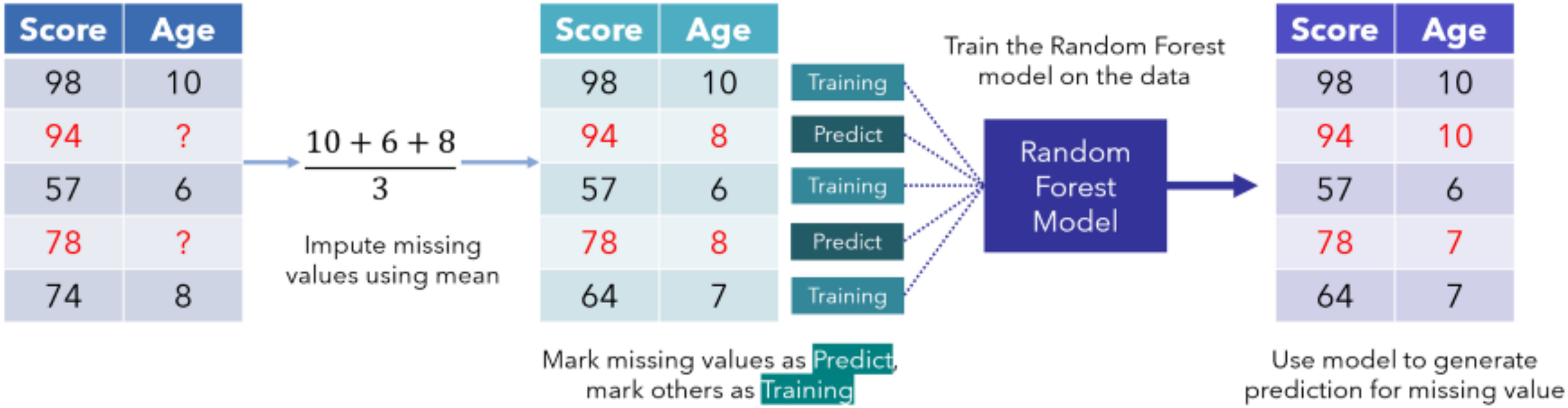
OUT-OF-BAG ERROR

Chúng ta có thể sử dụng out-of-bag dataset để đo lường độ chính xác của Random Forest

Outline

- Decision Tree Review
- Random Forest
- Fill in missing data with Random Forest
- Case study

Random Forest with Missing Data



Types of Missing Data

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	N/A	N/A	NO

Text or Numbering

AI VIETNAM
All-in-One Course

How to fill in missing data

DATA WITH MISSING VALUES → GUESSING THE DATA → REFINE THE GUESSES

Score	Age
98	10
94	?
57	6
78	?
74	8

$$\frac{10+6+8}{3}$$

Impute missing values using mean

Score	Age
98	10
94	8
57	6
78	8
64	7

Mark missing values as Predict, mark others as Training

Training
Predict
Training
Predict
Training

Train the Random Forest model on the data

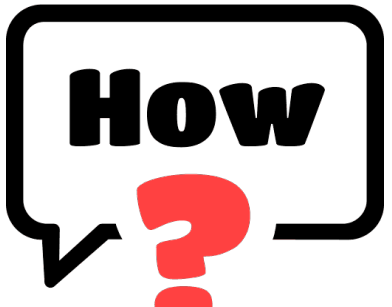
Random Forest Model

Score	Age
98	10
94	10
57	6
78	7
64	7

Use model to generate prediction for missing value

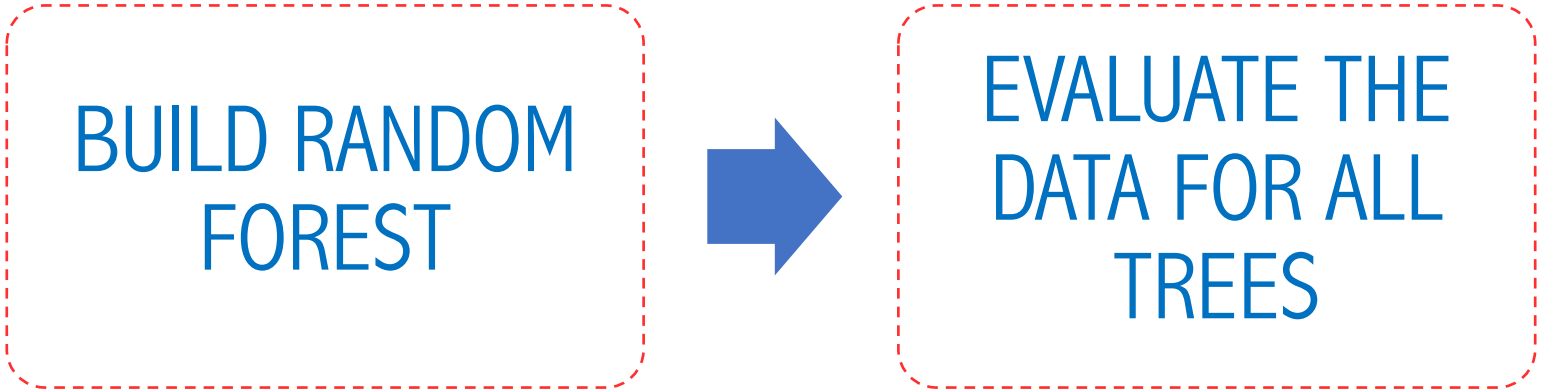
Guessing the Data

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	No	167.5	NO



Ý tưởng: Điền giá ban đầu, sau đó hiệu chỉnh dần cho nó tốt hơn

Refine the Guesses



Proximity Matrix

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	NO	167.5	NO

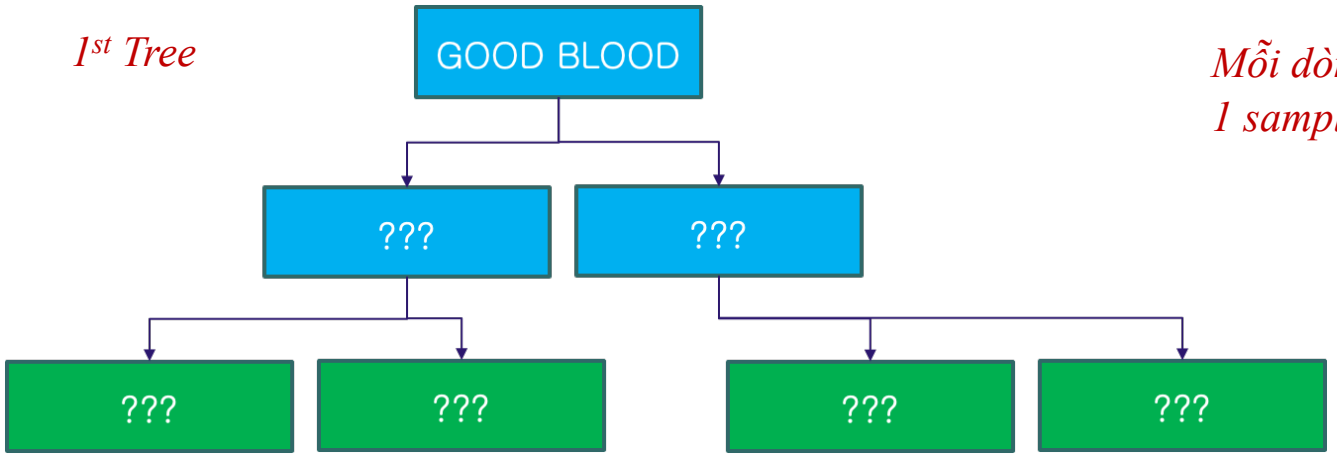
Sample 3 and sample 4 reaches to the same decision

Mỗi cột thể hiện 1 sample

	1	2	3	4
1				
2				
3				1
4			1	

Mỗi dòng thể hiện 1 sample

1st Tree

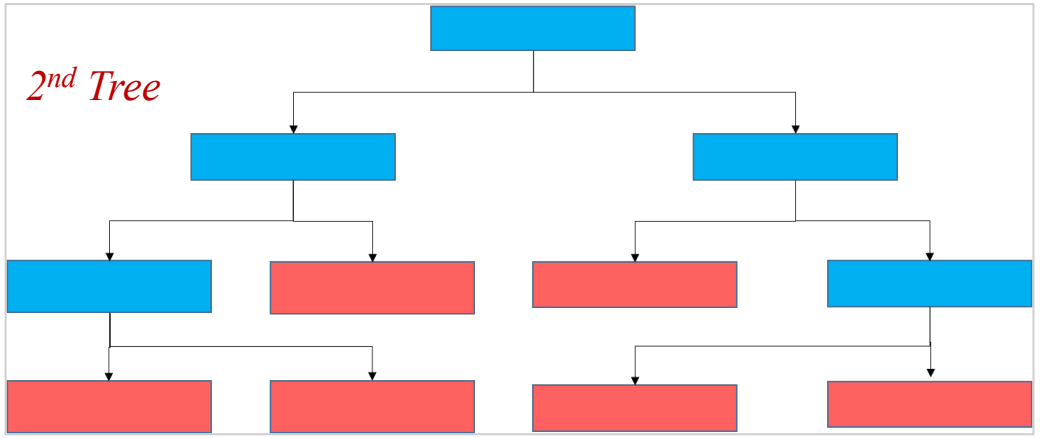


Dòng 3 và 4 cùng trả về kết quả là No

Proximity Matrix

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	NO	167.5	NO

Sample 3 and sample 4 reaches to the same decision



Mỗi dòng thể hiện 1 sample

Mỗi cột thể hiện 1 sample

	1	2	3	4
1				
2			1	1
3		1		2
4		1	2	

Proximity Matrix Of N Trees

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		8
4	1	1	8	

Proximity Matrix Of N Trees

Normalization:
Assume we have 10 trees.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Fill in Missing Values

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	???	???	NO

Frequency of Yes: 1/3

The weight frequency of Yes = Frequency of Yes * Weight for Yes
The weight frequency of Yes = 1/3 * 0.1 = 0.03

Weight for Yes = Proximity of Yes/All proximities

Proximity of Yes: 0.1

All proximities: 0.1 + 0.1 + 0.8 = 1.0

HOW

Proximity Matrix

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Fill in Missing Values

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	???	???	NO

Frequency of No: 2/3

The weight frequency of No = Frequency of No * Weight for No
The weight frequency of No = 2/3 * 0.9 = 0.6

Weight for No = Proximity of No/All proximities

Proximity of No: 0.1 + 0.8 = 0.9

All proximities: 0.1 + 0.1 + 0.8 = 1.0

HOW

Proximity Matrix

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Fill in Missing Values

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	???	???	NO

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	NO	???	NO

Predict

The weight frequency of No = 2/3 * 0.9 = 0.6

The weight frequency of Yes = 1/3 * 0.1 = 0.03

Fill in Missing Values

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	NO	???	NO

s1's weight = 125

Weight s1 = s1's weight * Weighted average weight of s1

Weight s1 = 125 * 0.1 = 12.5

Weighted average weight of s1 = 0.1 / (0.1 + 0.8 + 0.1) = 0.1

Proximity Matrix

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Fill in Missing Values

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	NO	???	NO

Proximity Matrix

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

s2's weight = 180

Weight s2 = s2's weight * Weighted average weight of s2
Weight s2 = 180 * 0.1= 18.0

Weighted average weight of s2 = 0.1 / (0.1 + 0.8 + 0.1) = 0.1

Fill in Missing Values

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	NO	???	NO

s3's weight = 210

Weight s3 = s3's weight * Weighted average weight of s3

Weight s3 = 210 * 0.8= 168.0

Weighted average weight of s3 = 0.8 / (0.1 + 0.8 + 0.1) = 0.8

Proximity Matrix

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Fill in Missing Values

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	NO	???	NO

Summation

Weight s1 = 125 * 0.1 = 12.5

Weight s2 = 180 * 0.1= 18.0

Weight s3 = 210 * 0.8= 168.0

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	NO	198.5	NO

Outline

- Decision Tree Review
- Random Forest
- Fill in missing data with Random Forest
- Case study

Decision Tree Implementation

- **Root node** - node at the top of the tree, contains a feature that best splits the data (a single feature that alone classifies the target variable most accurately)

- **Decision nodes** - nodes where the variables are evaluated. These nodes have arrows pointing to them and away from them

- **Leaf nodes** - final node at which the prediction is made

Decision Tree Implementation

- How to determine the root node:

- Check how every input feature classifies the target variable independently
- If neither is 100% correct, we can consider them as *impure*
- The **Entropy** metric can be used to calculate impurity
  - Values range from 0 (best) to 1 (worst)
- The variable with the lowest entropy (impurity) is used as a root node

Decision Tree Implementation

- **Training process:**

- Determine the root node
- Calculate the **Information gain** for a single split
  - The higher the gain the better the split
- Do a greedy search
  - Go over all input feature and their unique values (thresholds)
  - Calculate information gain for every feature/threshold combination
  - Save the best split feature and best split threshold for every node
  - Build the tree recursively
  - Some stopping criteria should be applied when doing so
    - Think of it as an exit condition of a recursive function
    - This could be maximum depth, minimum samples at node...
  - If at the leaf node, return the prediction (most common value)
    - You'll know you're at a leaf node if a stopping criteria has been met or if the split is pure

Decision Tree Implementation

- Prediction process:

  - Recursively traverse the tree
  - At each node check if the direction of the traversal (left or right), based on the input data
  - When the leaf node is reached, the most common value is returned

Decision Tree Implementation

- **Entropy:**
  - Measures the purity of the split
  - Calculated at the node level
  - Ranges between 0 (pure) and 1 (impure)

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i$$

$$s = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1]$$

$$n_0 = 7$$

$$n_1 = 3$$

$$E(S) = -\frac{7}{10} log_2(\frac{7}{10}) - \frac{3}{10} log_2(\frac{3}{10})$$

$$E(S) = -0.7 log_2(0.7) - 0.3 log_2(0.3)$$

$$E(S) = -0.7 \times -0.51457 - 0.3 \times -1.73697$$

$$E(S) = 0.88129$$

Decision Tree Implementation

- **Entropy:**
  - Measures the purity of the split
  - Calculated at the node level
  - Ranges between 0 (pure) and 1 (impure)

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i$$

```
[38] import numpy as np
     from collections import Counter
```

```
[40] # Tính entropy
     def entropy(s):
         counts = np.bincount(s)
         percentages = counts / len(s)

         entropy = 0
         for pct in percentages:
             if pct > 0:
                 entropy += pct * np.log2(pct)
         return -entropy
```

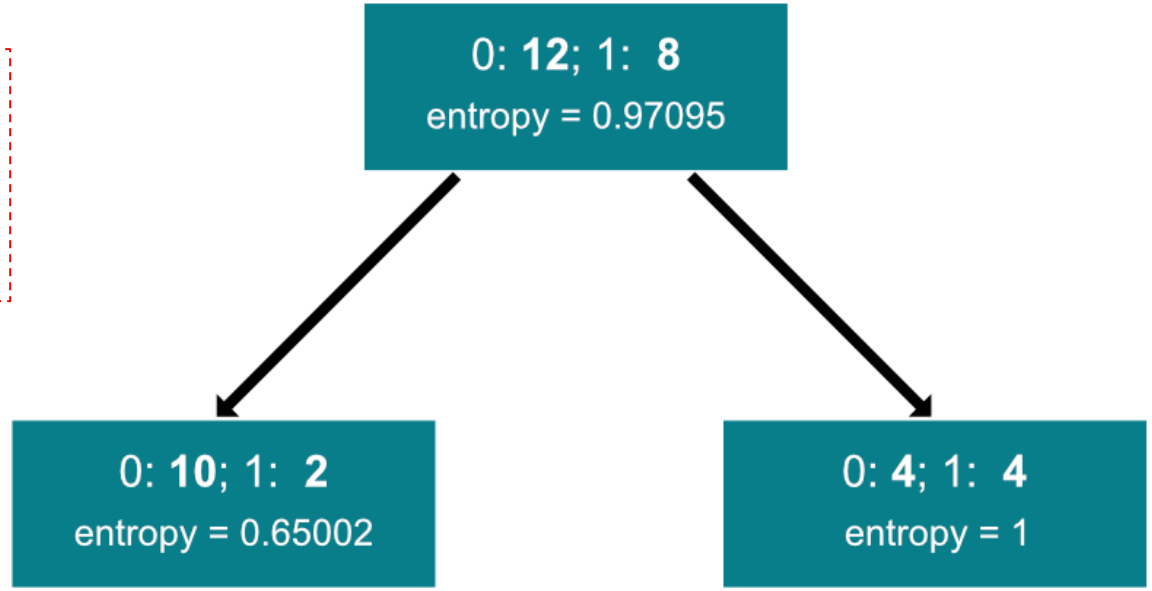
```
[42] # Kiểm tra
     s = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1]
     print(f'Entropy: {np.round(entropy(s), 2)}')

     Entropy: 0.88
```

Decision Tree Implementation

- Information Gain:

- Simply an average of all entropy based on a specific split
- The higher the information gain, the better the decision split is:

$$Gain(S, A) = E(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} E(S_v)$$



Bạn có thể giải thích thông tin từng node không?

Decision Tree Implementation

- Information Gain:

- Simply an average of all entropy based on a specific split
- The higher the information gain, the better the decision split is:

$$Gain(S, A) = E(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} E(S_v)$$

$$Gain(S, A) = E(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} E(S_v)$$

$$Gain(S, A) = 0.97095 - \frac{12}{20} \times 0.65002 - \frac{8}{20} \times 1$$

$$Gain(S, A) = 0.18094$$

Decision Tree Implementation

- Information Gain:

$$Gain(S, A) = E(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} E(S_v)$$

```
[44] #Tính information gain
     def information_gain(parent, left_child, right_child):
         num_left = len(left_child) / len(parent)
         num_right = len(right_child) / len(parent)

         gain = entropy(parent) - (num_left * entropy(left_child) + num_right * entropy(right_child))
         return gain
```

```
[45] # Kiểm thử
     parent = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]
     left_child = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]
     right_child = [0, 0, 0, 0, 1, 1, 1, 1]

     print(f'Information gain: {np.round(information_gain(parent, left_child, right_child), 5)}')
```

```
     Information gain: 0.18094
```

Rercursion Review

```
def factorial(x):
    # Exit condition
    if x == 1:
        return 1
    return x * factorial(x - 1)

print(f'Factorial of 5 is {factorial(5)}')
```

```
Factorial of 5 is 120
```

Define AIONode class

```
class AIONode:
    '''
    Helper class which implements a single tree node.
    '''
    def __init__(self, feature=None, threshold=None,
                 data_left=None, data_right=None, gain=None, value=None):
        self.feature = feature
        self.threshold = threshold
        self.data_left = data_left
        self.data_right = data_right
        self.gain = gain
        self.value = value
```

Define AIODecisionTreee

```
class AIODecisionTree:
    '''
    Class which implements a decision tree classifier algorithm.
    '''
    def __init__(self, min_samples_split=2, max_depth=5):
        self.min_samples_split = min_samples_split
        self.max_depth = max_depth
        self.root = None

    @staticmethod
    def _entropy(s):
        '''
        Helper function, calculates entropy from an array of integer values.

        :param s: list
        :return: float, entropy value
        '''
        # Convert to integers to avoid runtime errors
        counts = np.bincount(np.array(s, dtype=np.int64))
        # Probabilities of each class label
        percentages = counts / len(s)

        # Caclulate entropy
        entropy = 0
        for pct in percentages:
```

Load Iris Dataset


```
[27] # Sử dụng Iris dataset từ Sklearn
     from sklearn.datasets import load_iris
     from sklearn.model_selection import train_test_split

     iris = load_iris()
     X = iris['data']
     y = iris['target']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Your progam vs Sklearn Lib

```
[28] # Đánh giá độ chính xác
     model = AIODecisionTree()
     model.fit(X_train, y_train)
     preds = model.predict(X_test)
     accuracy_score(y_test, preds)
```

```
1.0
```

```
[29] from sklearn.tree import DecisionTreeClassifier

     # Đánh giá độ chính xác sử dụng thư viện Sklearn
     sk_model = DecisionTreeClassifier()
     sk_model.fit(X_train, y_train)
     sk_preds = sk_model.predict(X_test)
     accuracy_score(y_test, sk_preds)
```

```
1.0
```

Random Forest Implementation

- Make N data subsets from the original set (training)
- Build N decision trees (training)
- Make predictions with every trained decision tree, and return a final prediction as a majority vote (prediction)

- Three classes
  - **AIONode** - implements a single node of a decision tree
  - **AIODecisionTree** - implements a single decision tree
  - **AIORandomForest** - implements our ensemble algorithm
- The Node class is here to store the data about the feature, threshold, data going left and right, information gain, and the leaf node value
  - All are initially set to None
  - The leaf node value is available only for leaf nodes

Random Forest Implementation

```python
class AIORandomForest:
    '''
    A class that implements Random Forest algorithm from scratch.
    '''
    def __init__(self, num_trees=25, min_samples_split=2, max_depth=5):
        self.num_trees = num_trees
        self.min_samples_split = min_samples_split
        self.max_depth = max_depth
        # Will store individually trained decision trees
        self.decision_trees = []

    @staticmethod
    def _sample(X, y):
        '''
        Helper function used for boostrap sampling.

        :param X: np.array, features
        :param y: np.array, target
        :return: tuple (sample of features, sample of target)
        '''
        n_rows, n_cols = X.shape
        # Sample with replacement
        samples = np.random.choice(a=n_rows, size=n_rows, replace=True)
        return X[samples], y[samples]
```

AI VIETNAM
All-in-One Course

Random Forest Implementation

```
# Load data từ Sklearn
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

iris = load_iris()
X = iris['data']
y = iris['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Đánh giá độ chính xác giải thuật random forest không dùng lib
model = AIORandomForest()
model.fit(X_train, y_train)
random_forest_preds = model.predict(X_test)
accuracy_score(y_test, random_forest_preds)
```

```
1.0
```

```
# Đánh giá độ chính xác giải thuật random forest
# sử dụng RandomForestClassifier class từ Sklearn
from sklearn.ensemble import RandomForestClassifier

sk_model = RandomForestClassifier()
sk_model.fit(X_train, y_train)
sk_preds = sk_model.predict(X_test)
accuracy_score(y_test, sk_preds)
```

```
1.0
```