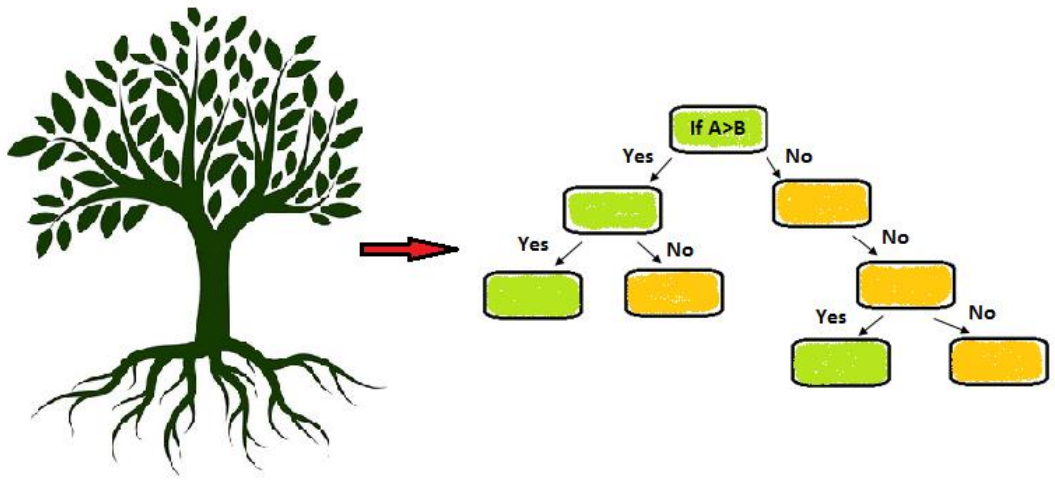
AI VIET NAM
@aivietnam.edu.vn

Random Forest

(Basic, Advanced Concepts and Its Applications)



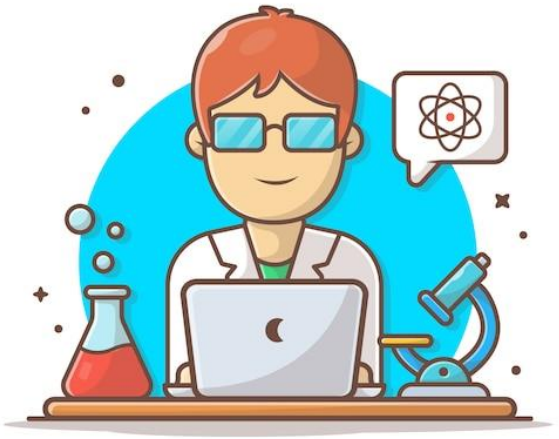
Vinh Dinh Nguyen
PhD in Computer Science

Outline

- Decision Tree: Review
- Random Forest
- Fill in missing data with Random Forest
- Time series Vs. Supervised Learning
- Example
- Summary

AI VIET NAM
@aivietnam.edu.vn

Decision Tree Review



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

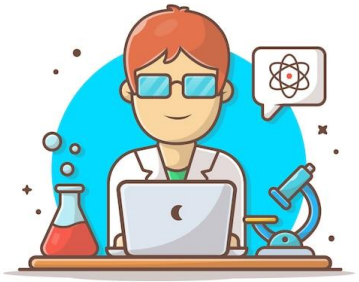
Tiêm 5 đơn vị vaccine

Hiệu quả vaccine: 44%

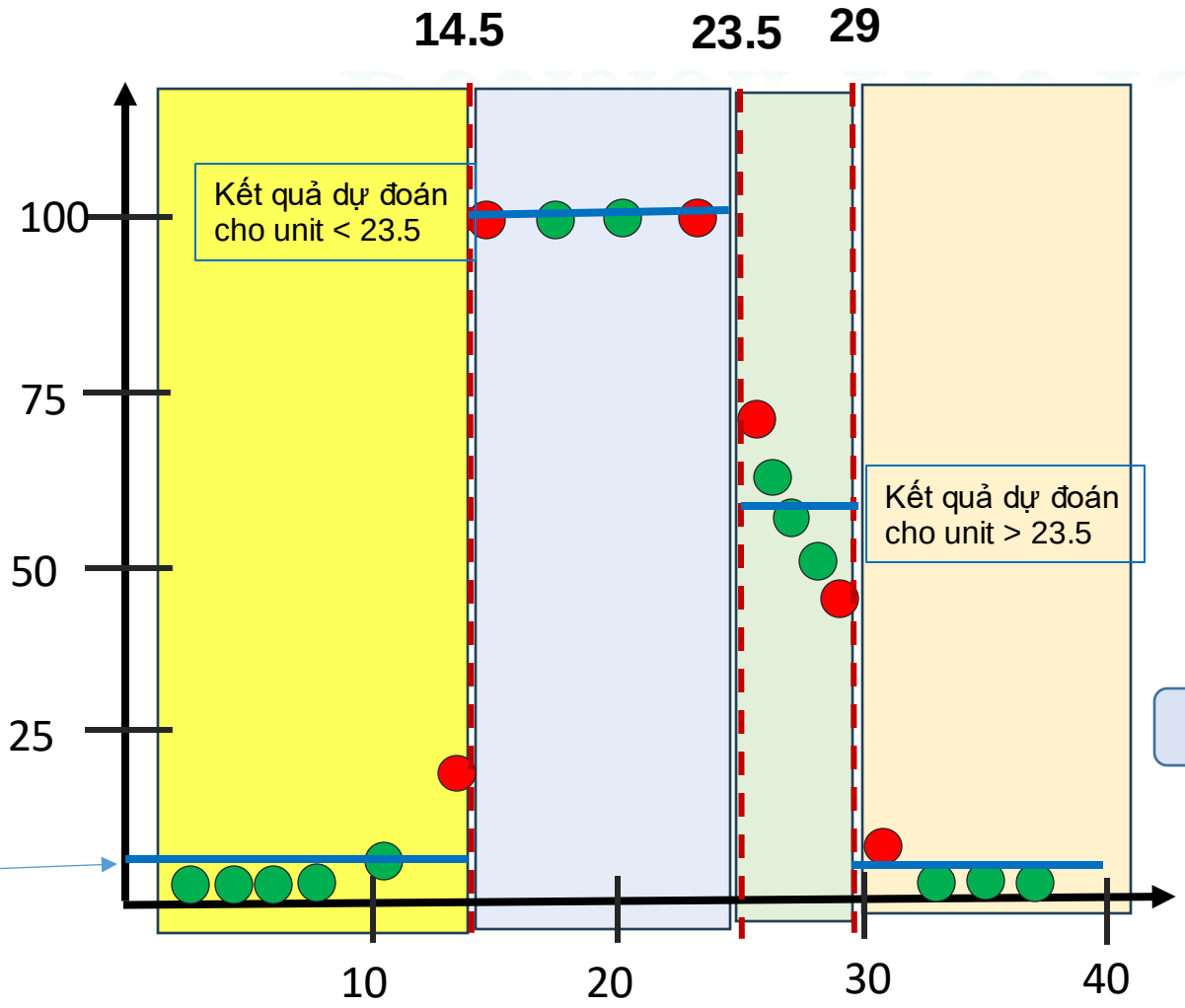
Khi có 1 vaccine ra đời, chúng ta muốn dự đoán xem nó hiệu quả bao nhiêu % ứng với từng liều lượng dùng trên bệnh nhân.

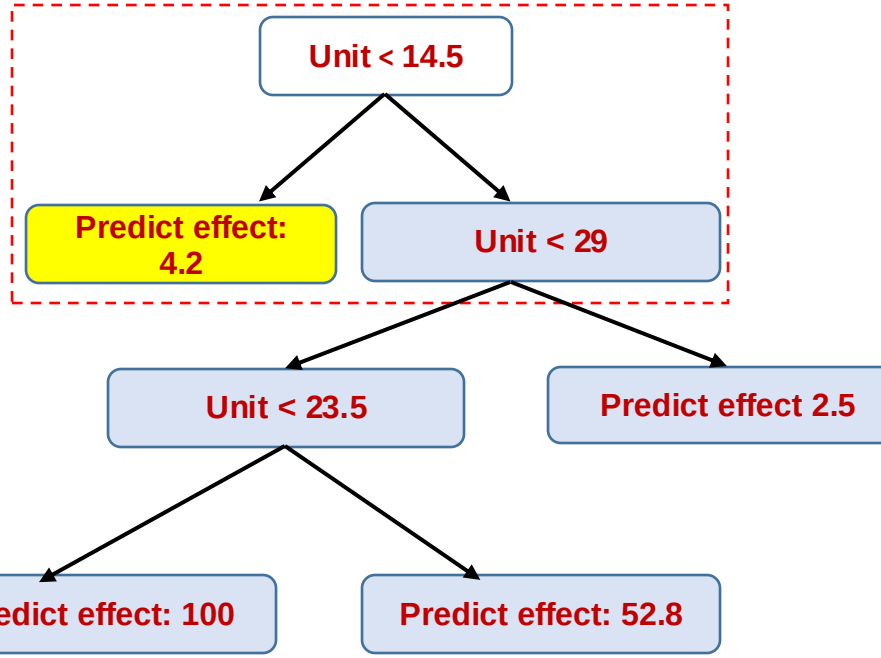
AI VIET NAM
@aivietnam.edu.vn

Decision Tree Review



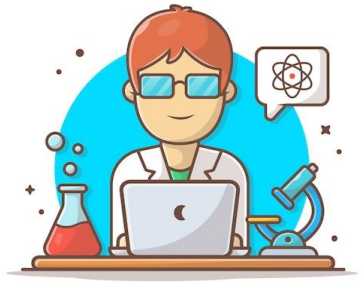
Effectiveness (Hiệu quả) (%)



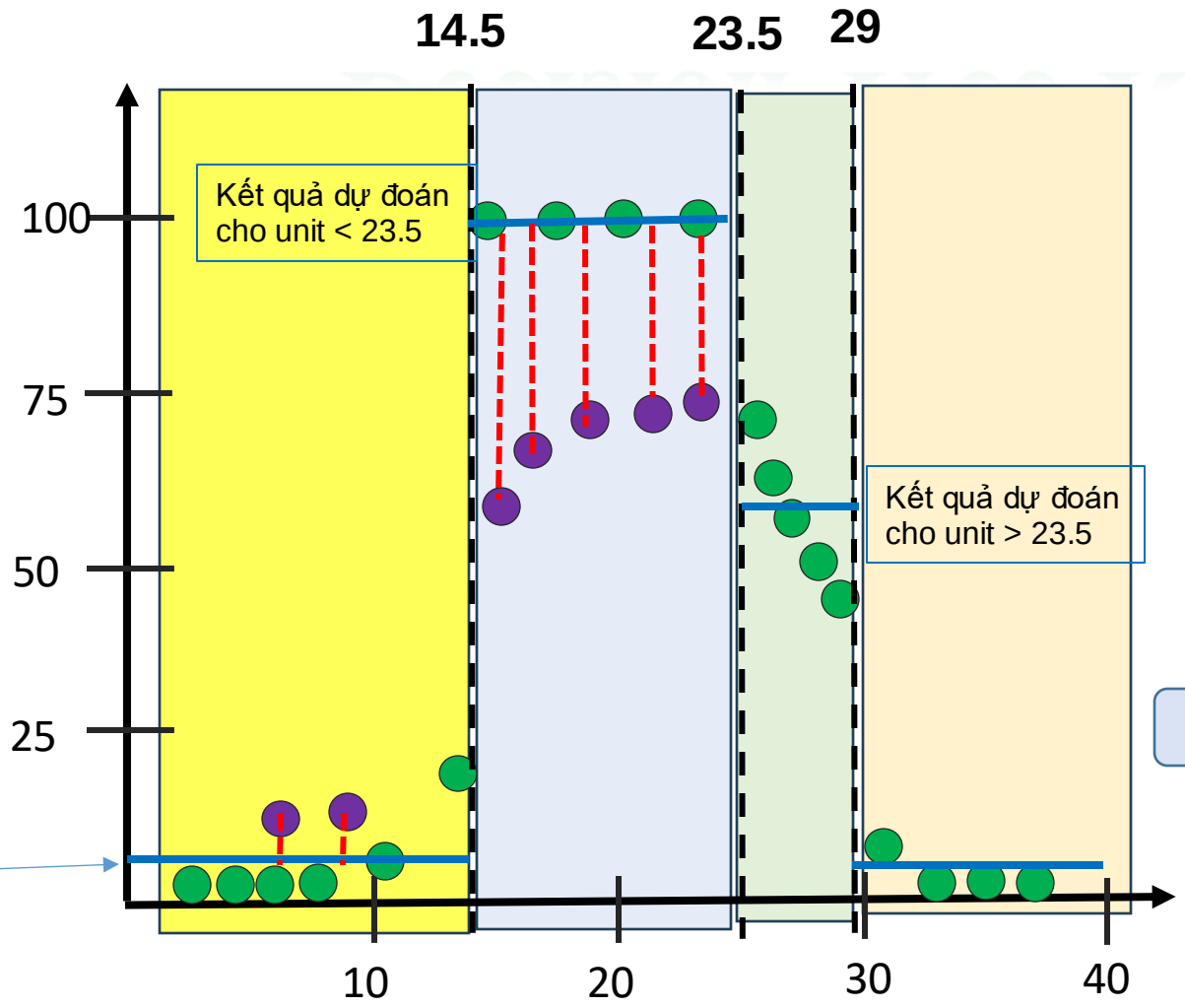


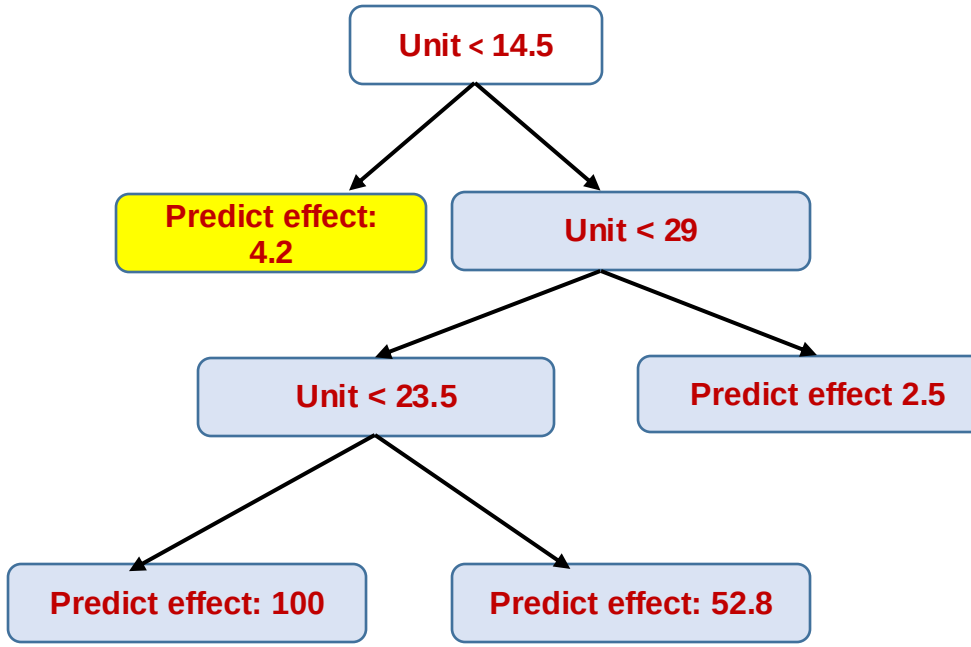
AI VIET NAM
@aivietnam.edu.vn

Decision Tree Review



Effectiveness (Hiệu quả) (%)





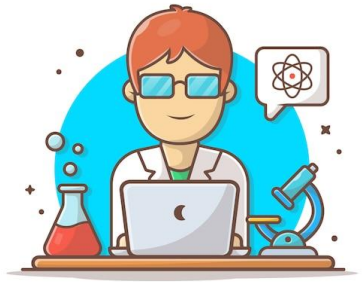
Dữ liệu test

Dữ liệu train

Error

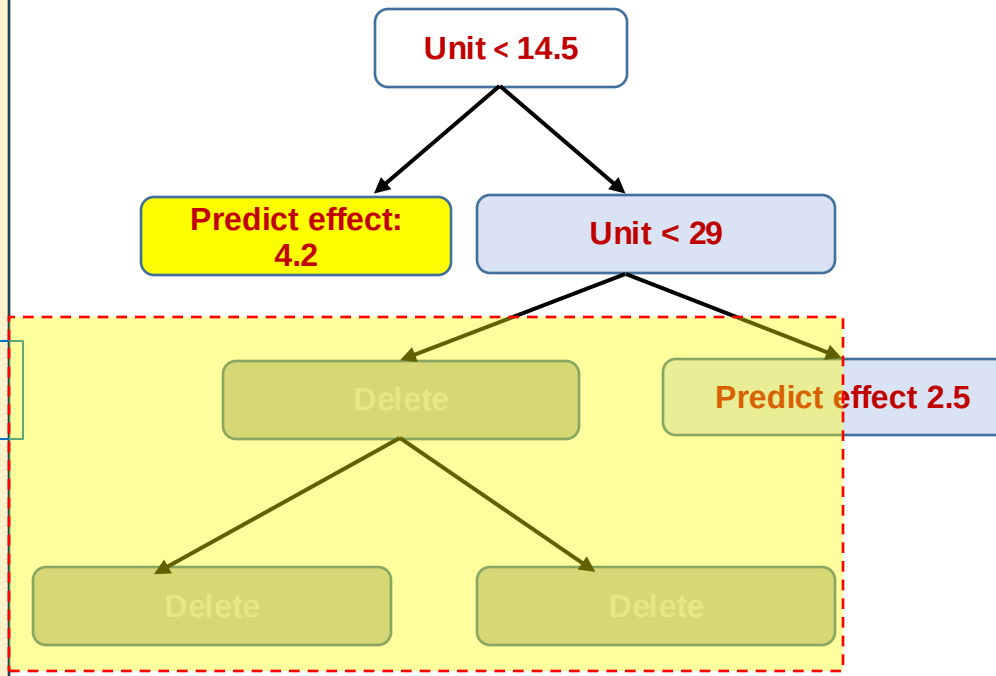
AI VIET NAM
@aivietnam.edu.vn

Decision Tree Review



Effectiveness (Hiệu quả) (%)

Unit (Đơn vị) vaccine



Dữ liệu test

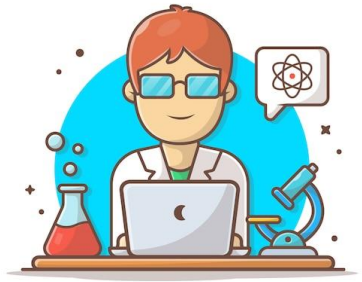
Dữ liệu train

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

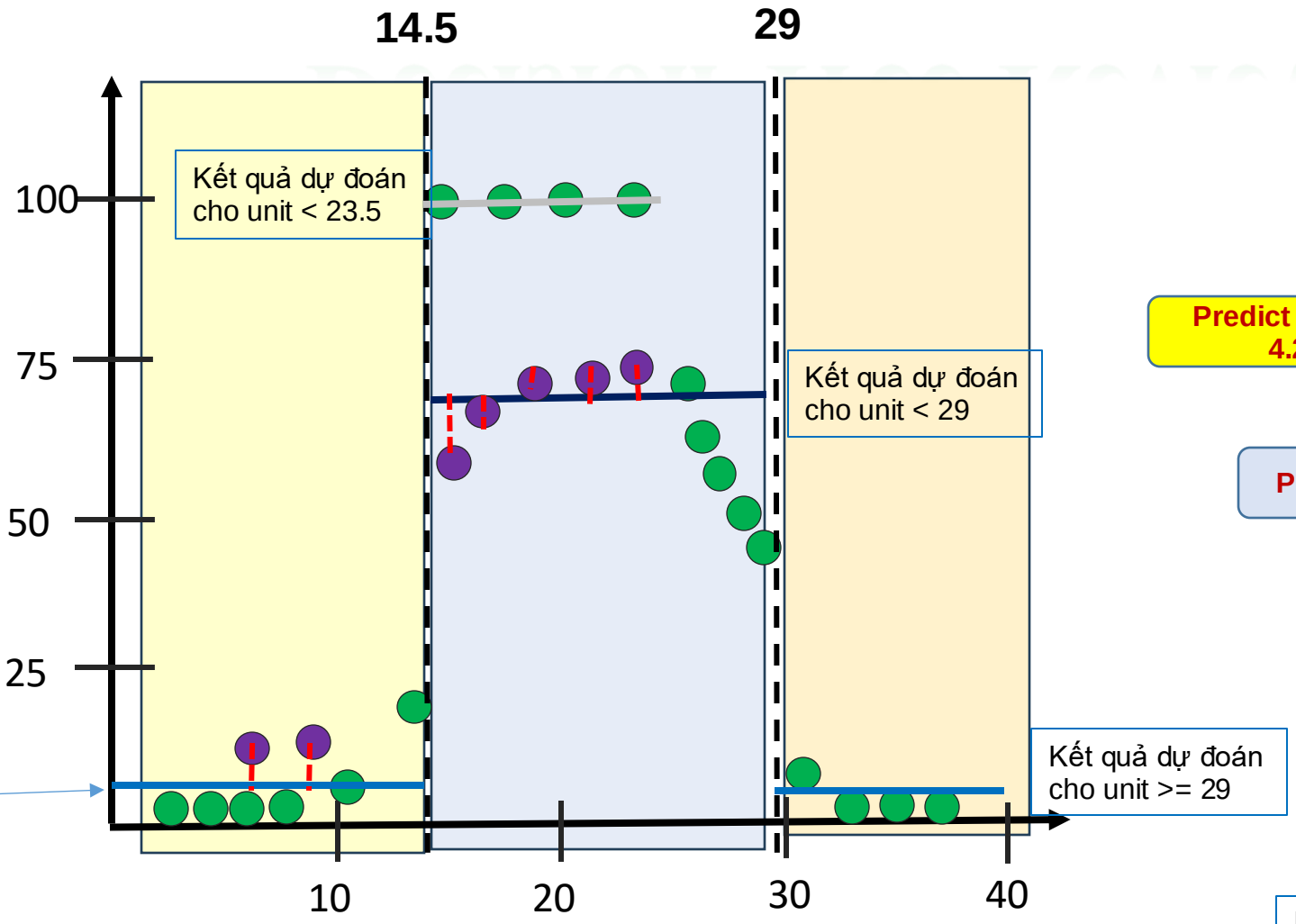
Error

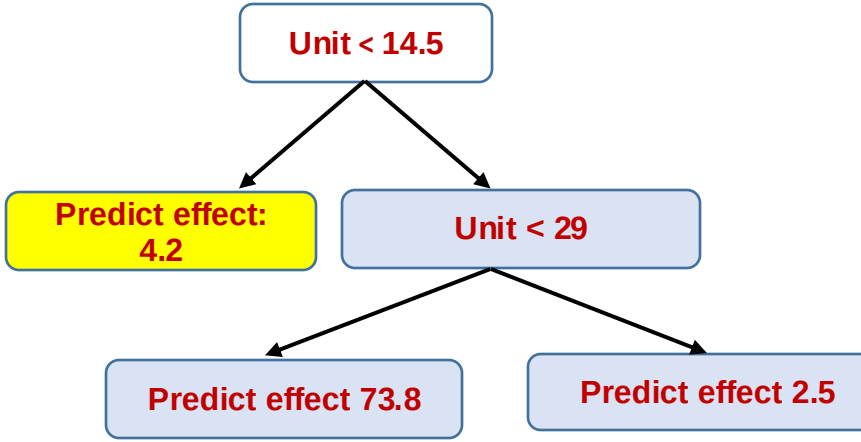
AI VIET NAM
@aivietnam.edu.vn

Decision Tree Review



Effectiveness (Hiệu quả) (%)





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

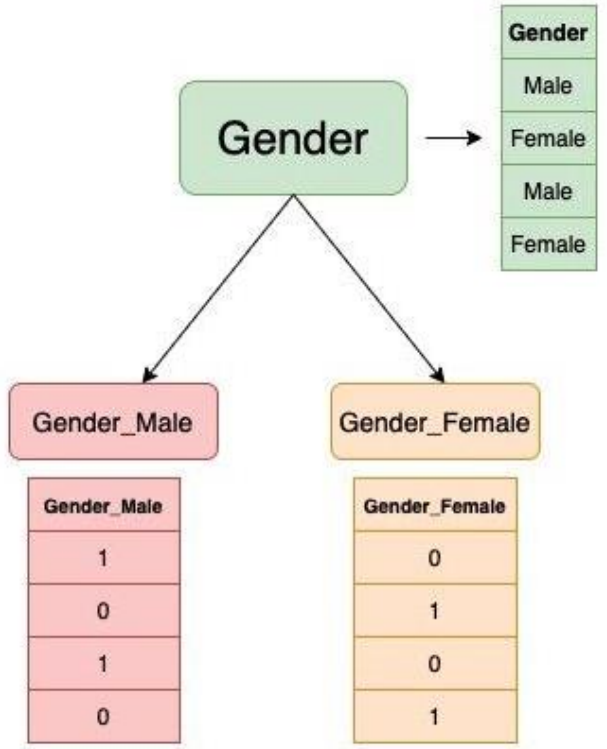
AI VIET NAM
@aivietnam.edu.vn

Dummy Variable

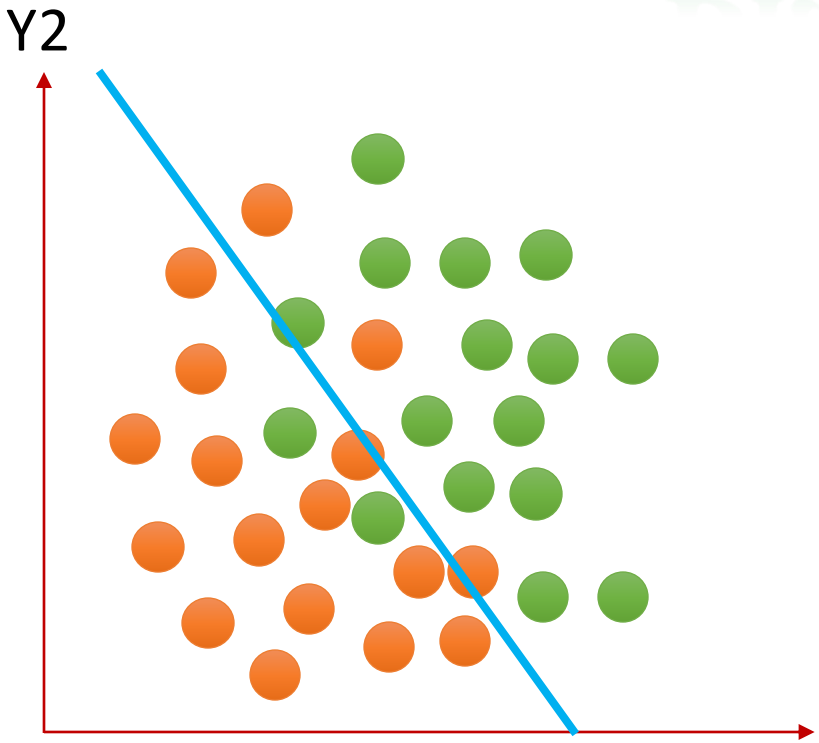
Water	Temperature
A	Hot
B	Cold
C	Warm
D	Cold

Dummy Variables

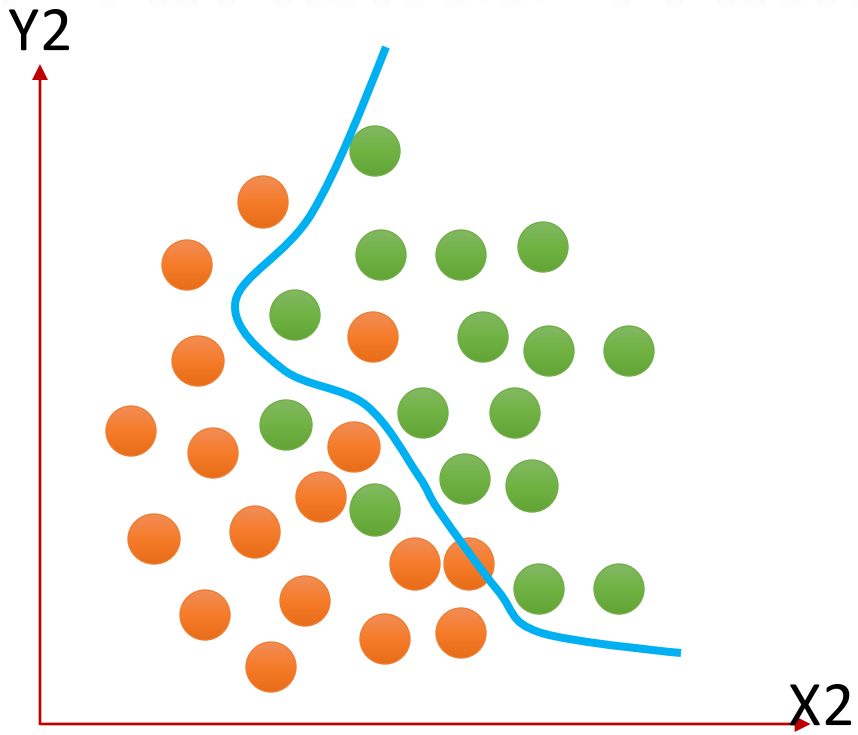
Water	Temperature	var_hot	var_warm	var_cold
A	Hot	1	0	0
B	Cold	0	0	1
C	Warm	0	1	0
D	Cold	1	0	0



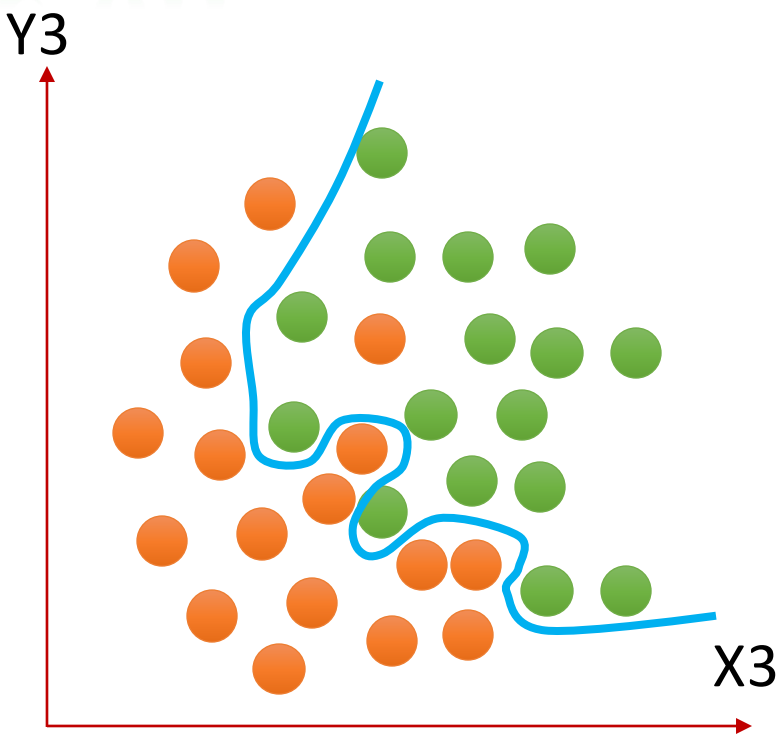
Bias-Variance Trade-off



High bias, low variance
(Underfitting)



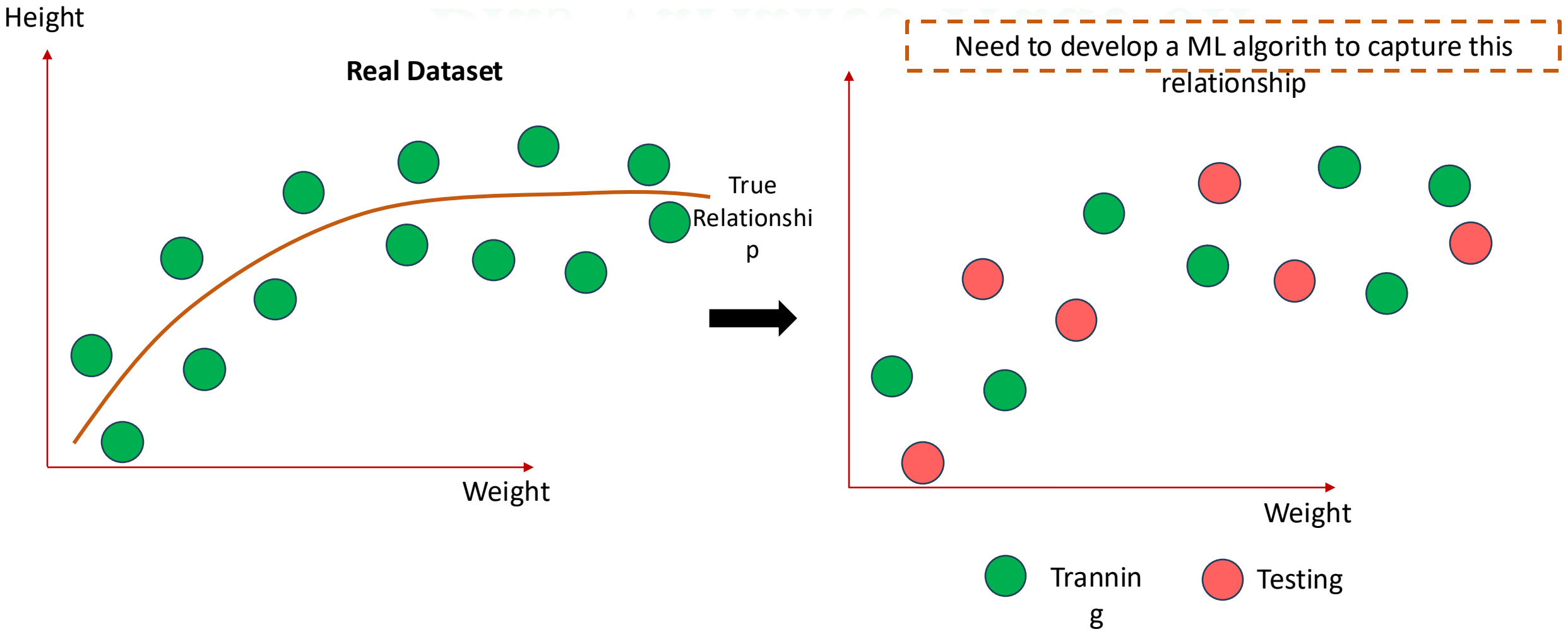
Low variance, low bias
(just right)



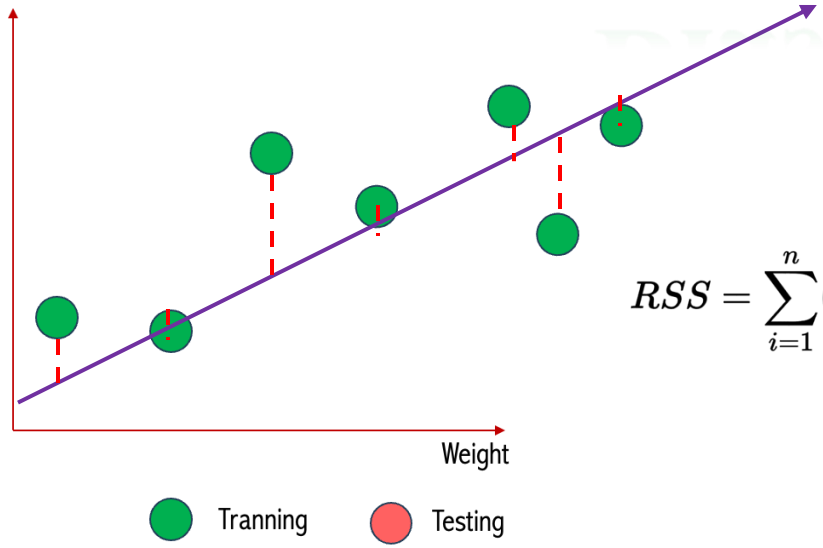
High variance, low bias
(overfitting)

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

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

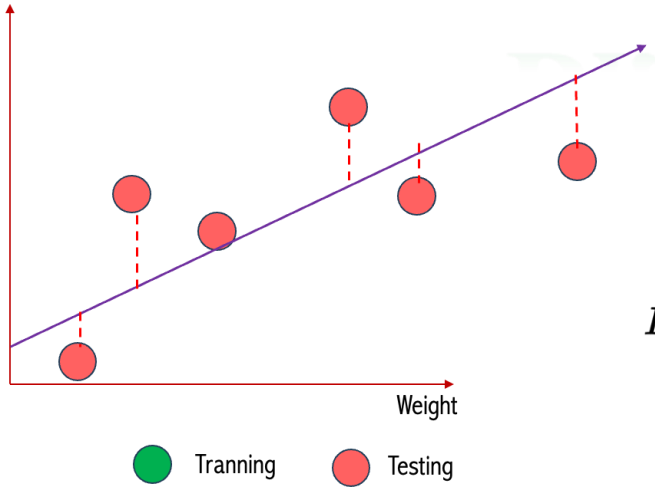
Polynomial Regression can capture the true relationship between weight and height

↓

Polynomial Regression has a low bias

AI VIET NAM
@aivietnam.edu.vn

Bias-Variance Trade-off



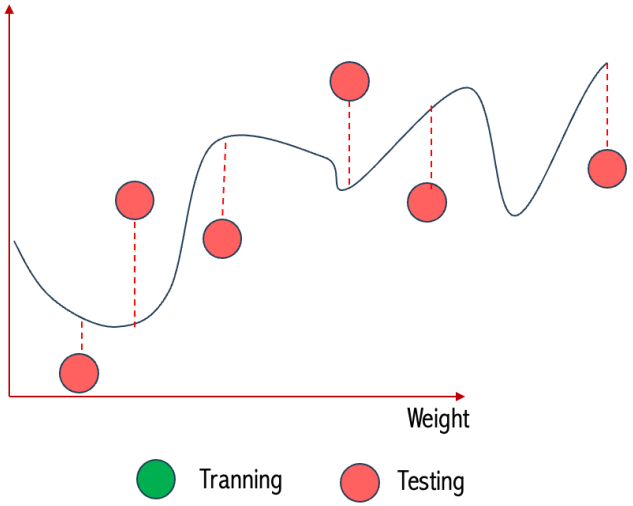
Linear Regression

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

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



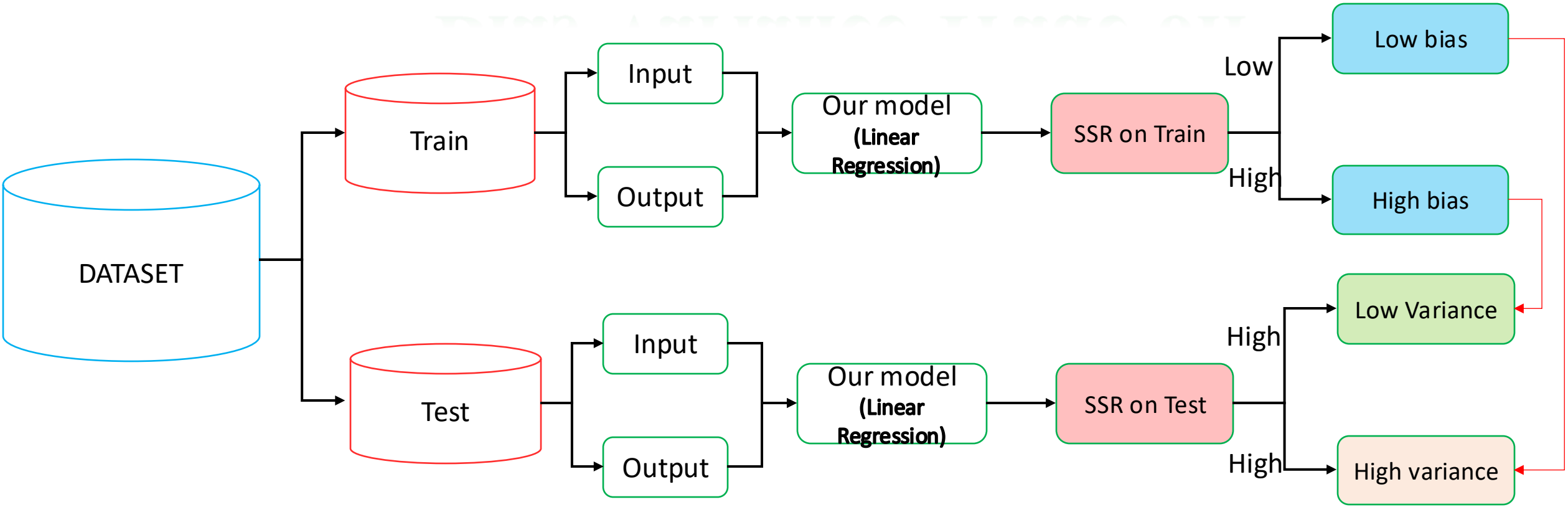
Polynomial Regression

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

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

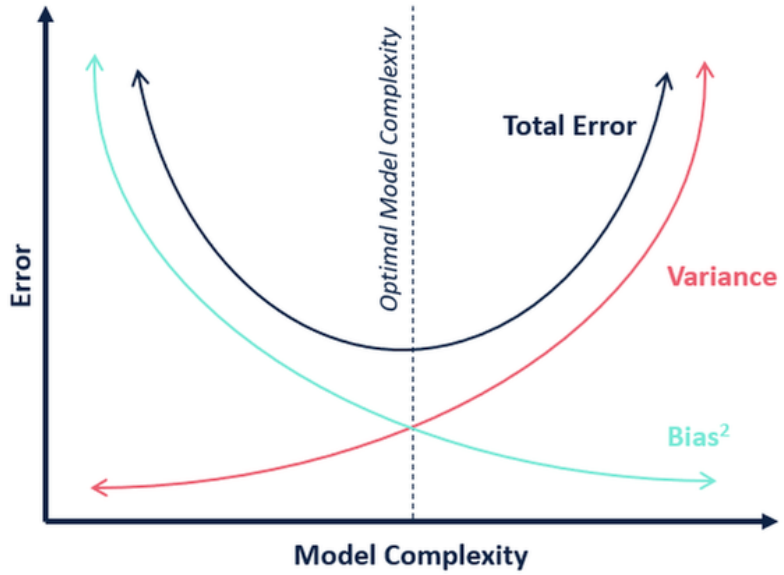
Bias-Variance Trade-off

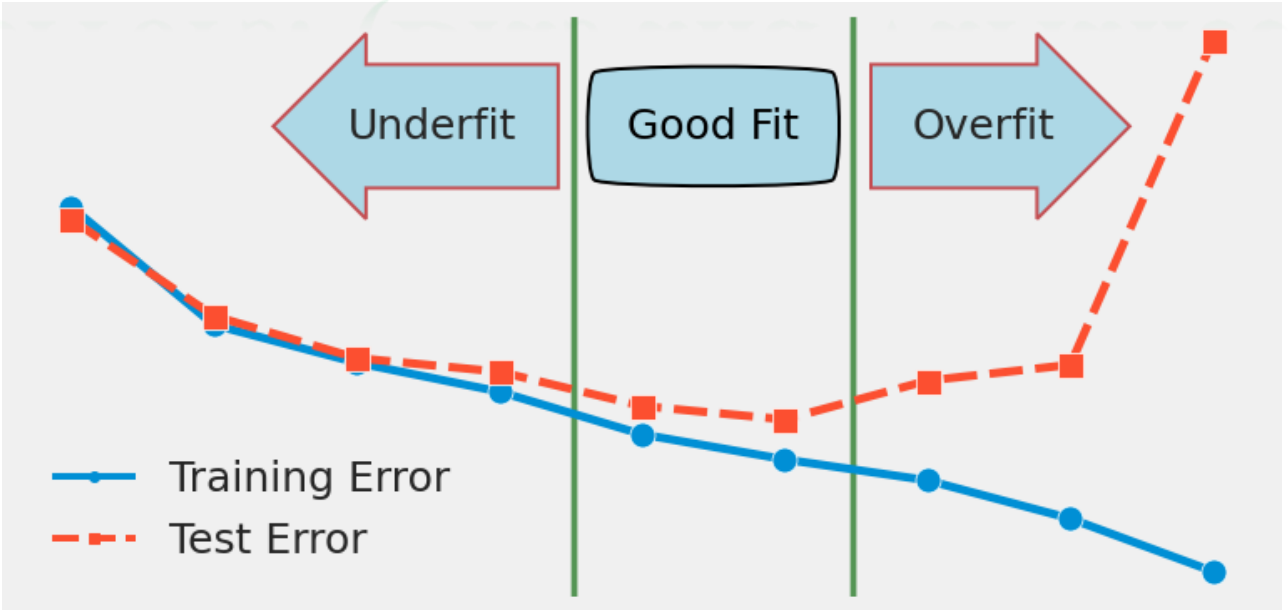


Bias as the error rate of the training data.

The different in fits between datatasets is call variance

Prediction errors: (Bias and Variance)



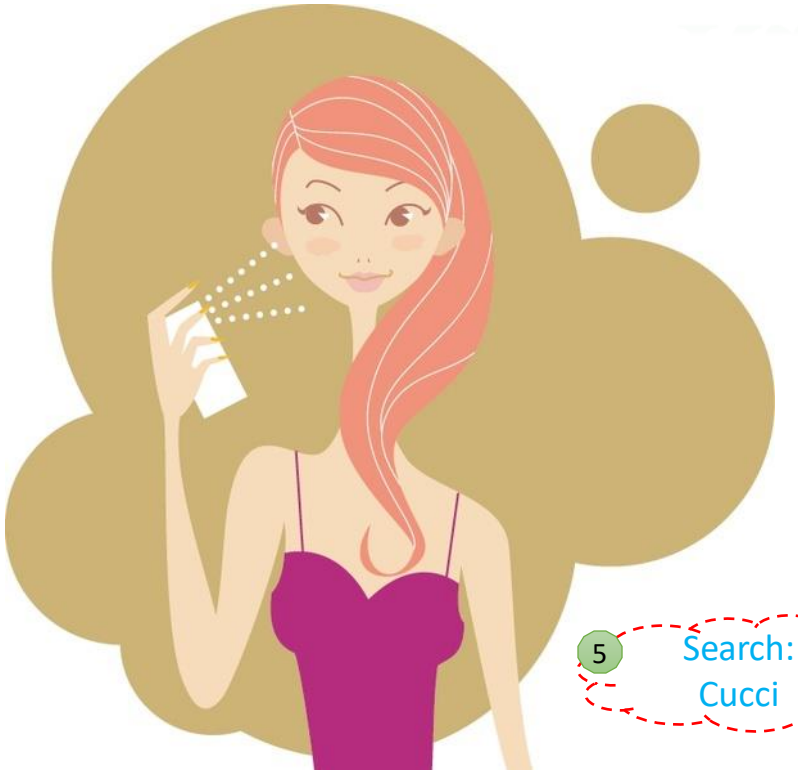


Bias as the error rate of the training data.

The different in fits between datatasets is call variance

AI VIET NAM
@aivietnam.edu.vn

Random Forest: Motivation



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

What would you do?

1 Channel đi con!

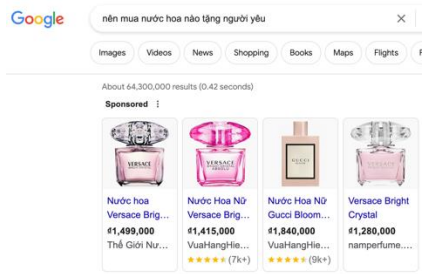
2 Channel đi bạn!!

Ask for Idea!

3 Cucci đi!!

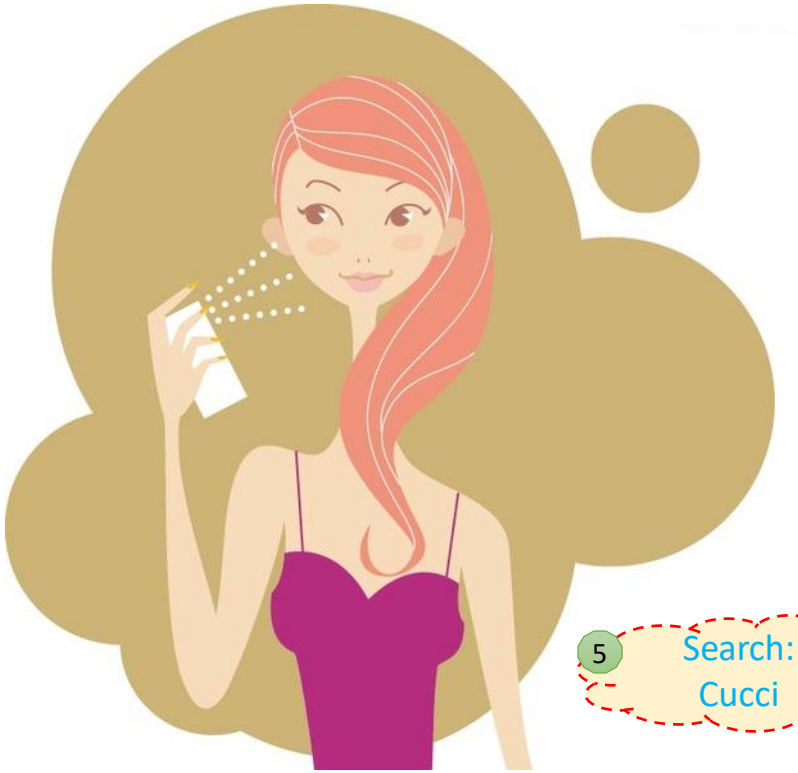
4 Cucci ạ!!

5 Search: Cucci



AI VIET NAM
@aivietnam.edu.vn

Random Forest: Motivation



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

What would you do?

1 Channel đi con!

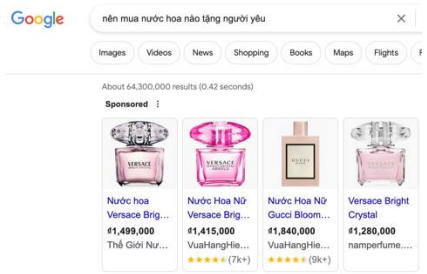
2 Channel đi bạn!!

Mua Gucci Thôi!!!

3 Cucci đi!!

4 Cucci ạ!!

5 Search: Cucci



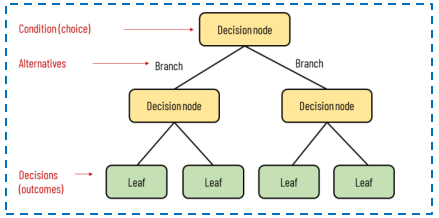
AI VIET NAM
@aivietnam.edu.vn

Random Forest: Motivation

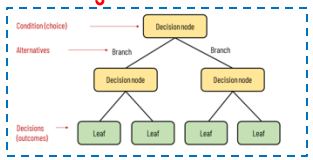
You want to buy a perfume for your girlfriend(s)?

What would you do?

1 Channel đi con!

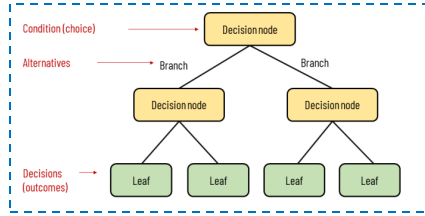


2 Channel đi bạn!!

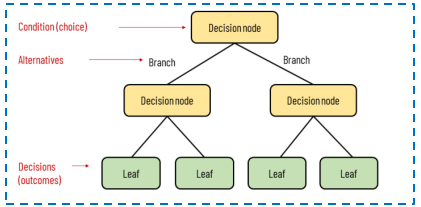


Mua Gucci Thôi!!!

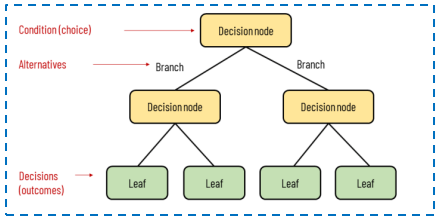
3 Cucci đi!!



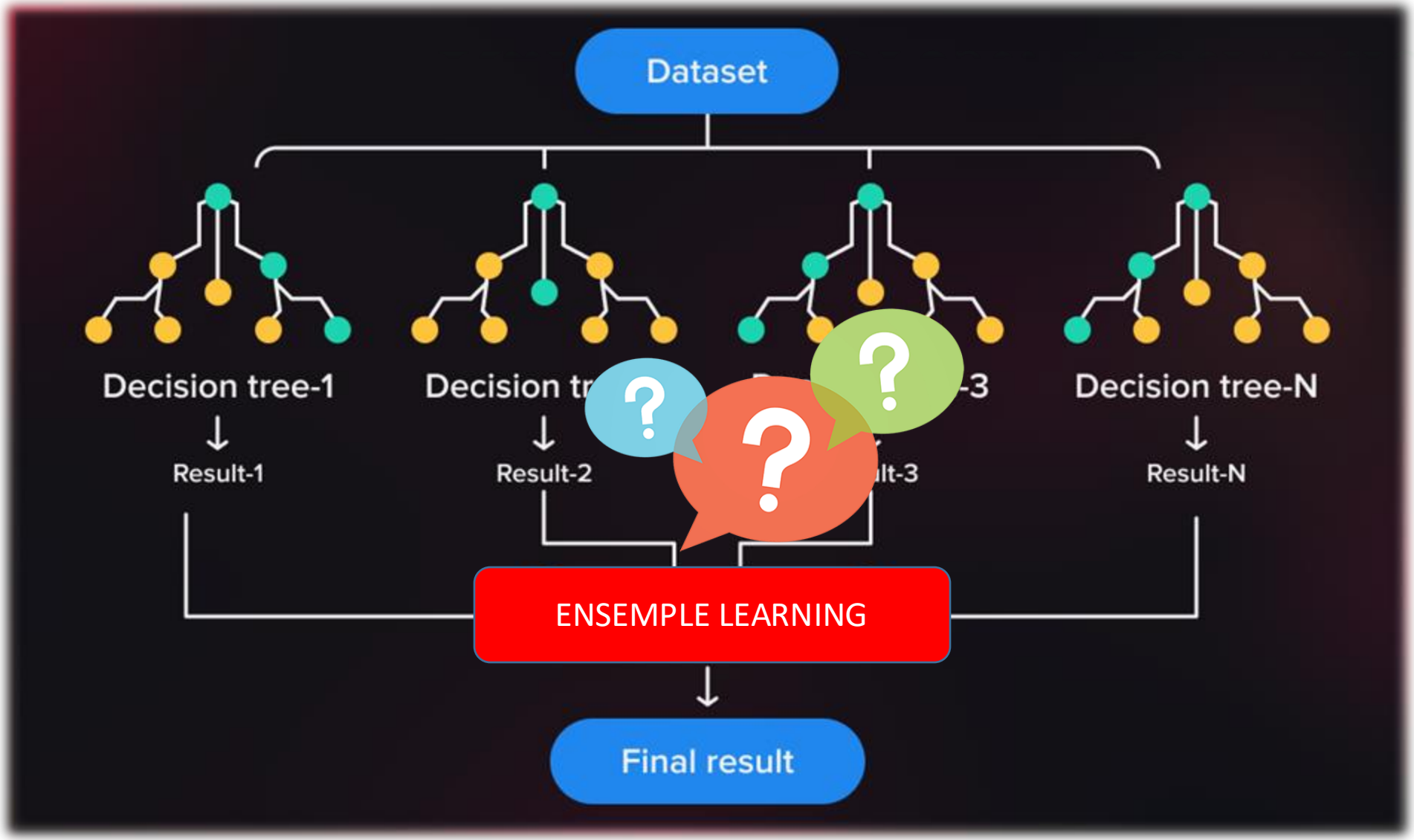
4 Cucci ạ!!



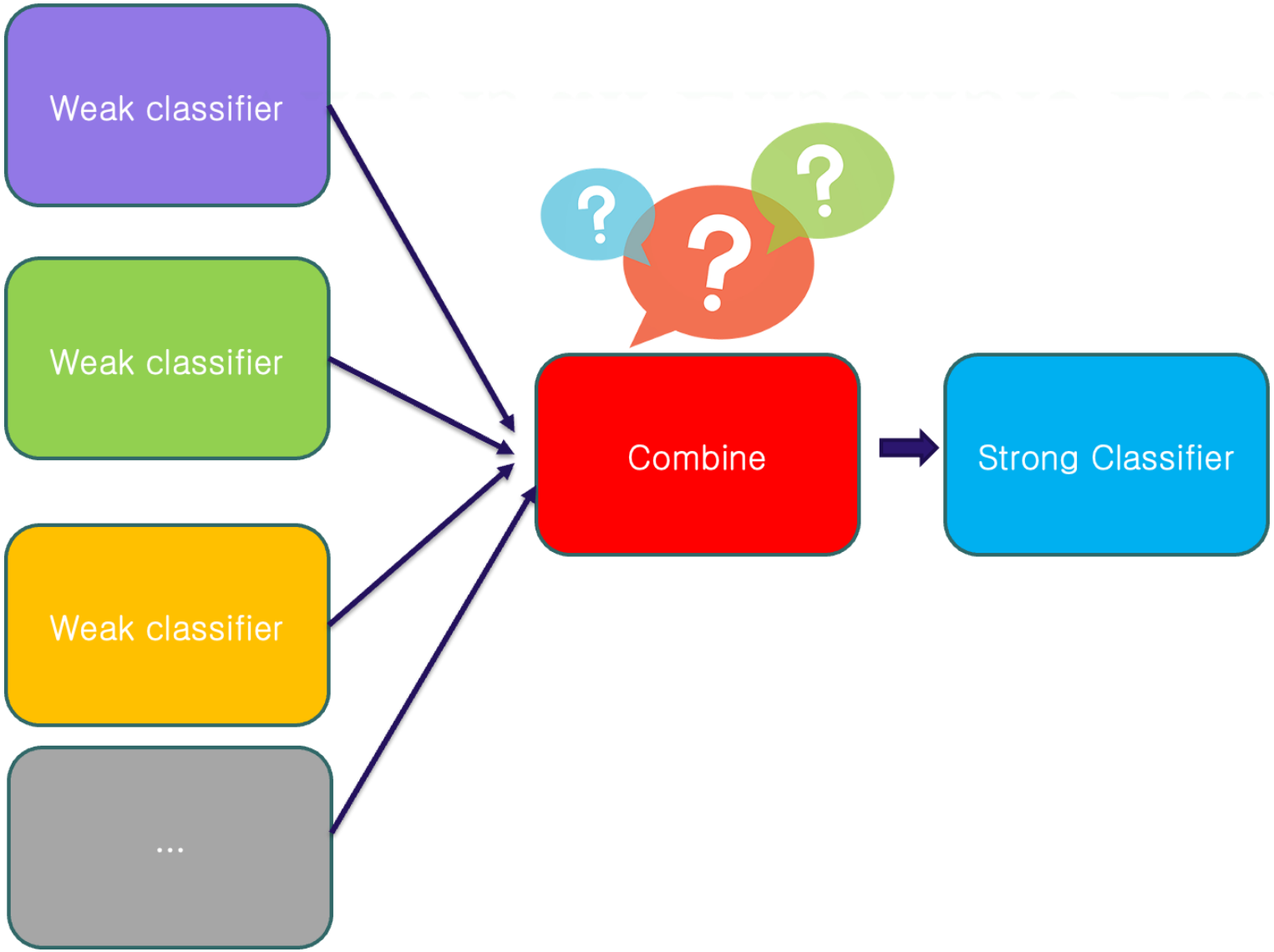
5 Search: Cucci



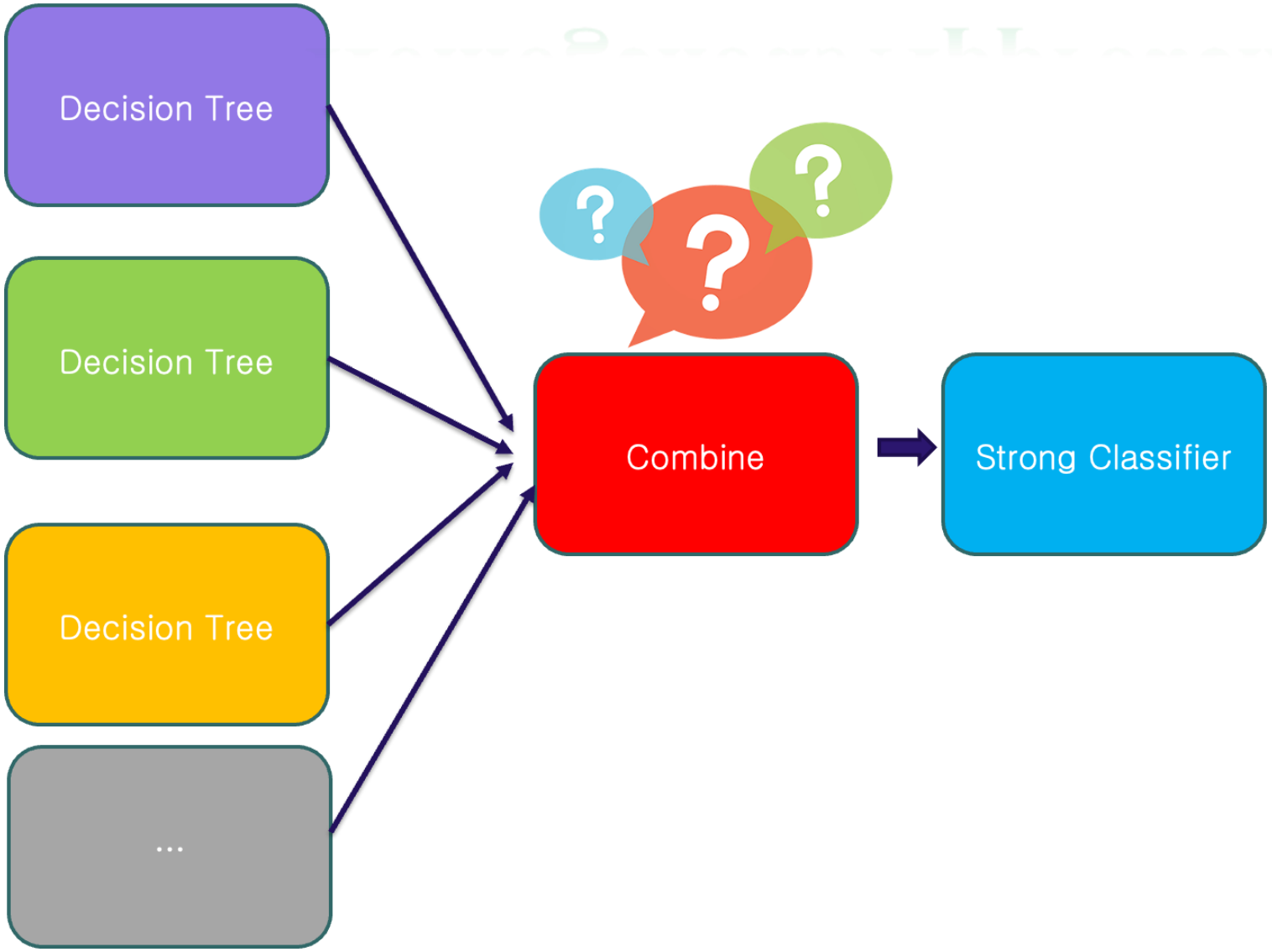
Random Forest: Motivation



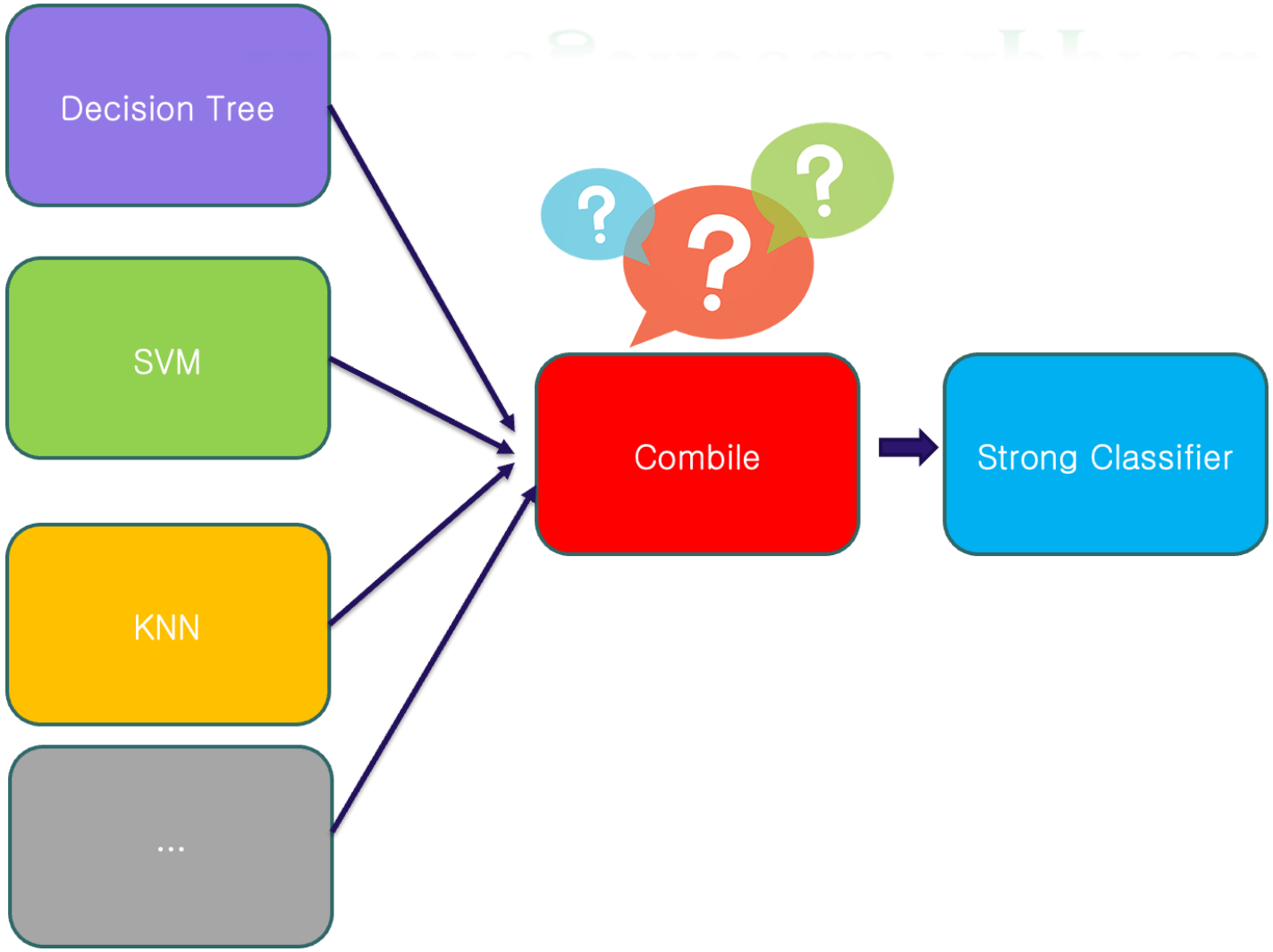
What is an Ensemble Learning



Homogenous Approach



Heterogeneous Approach



Ensemble Learning Techniques

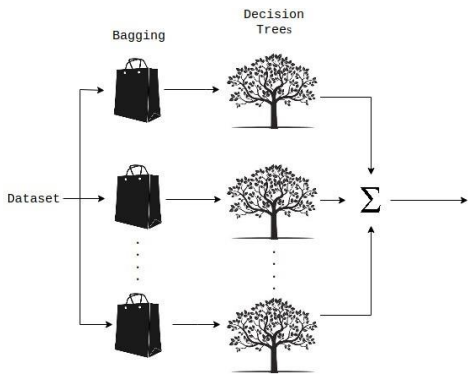
Ensemple Learning

Thông dụng ở các cuộc thi về AI

Bagging

homogeneous weak learners

Random Forest



Boosting

homogeneous weak learners

NEXT WEEK

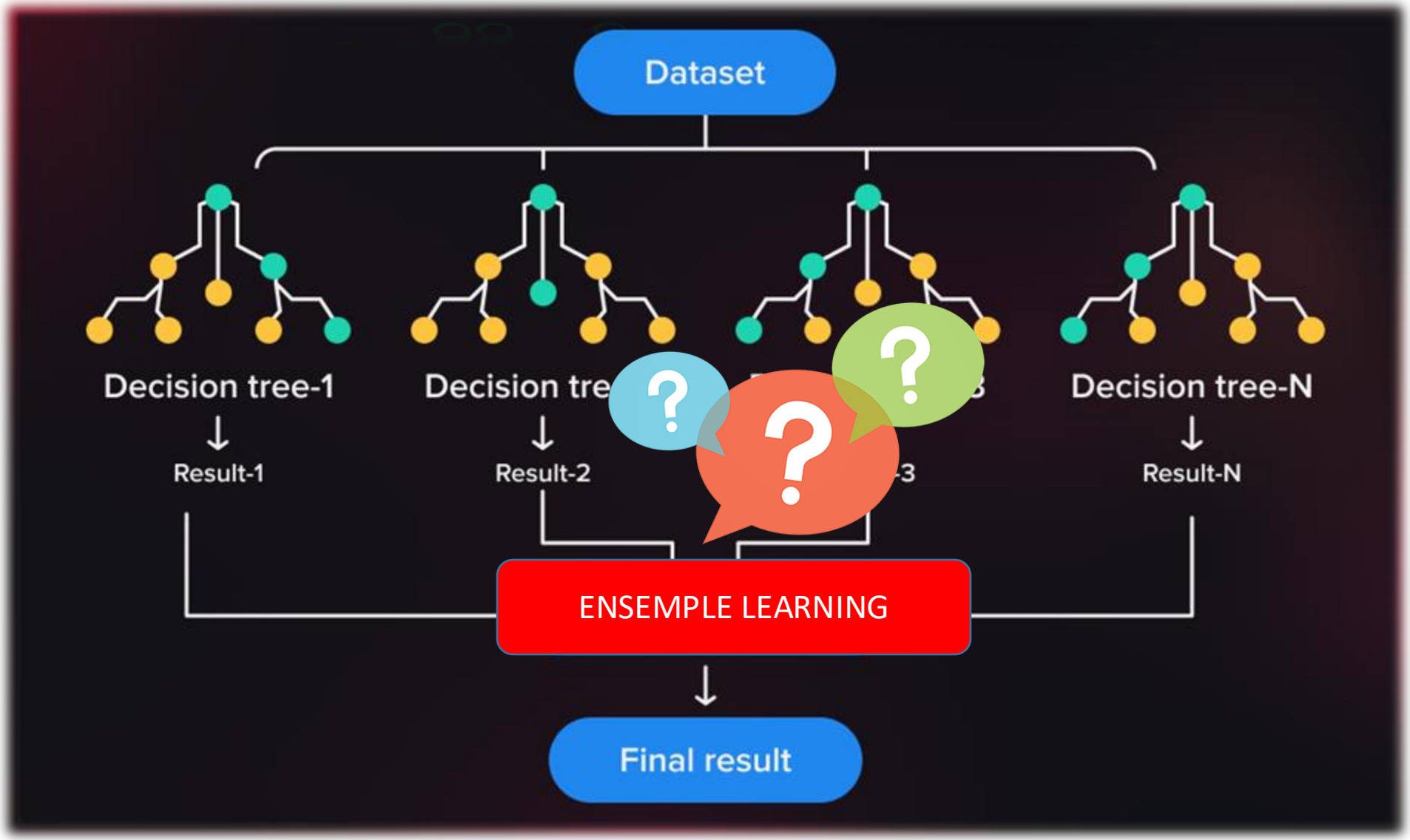
Stacking

Heterogeneous weak learners

OUT OF SCOPF

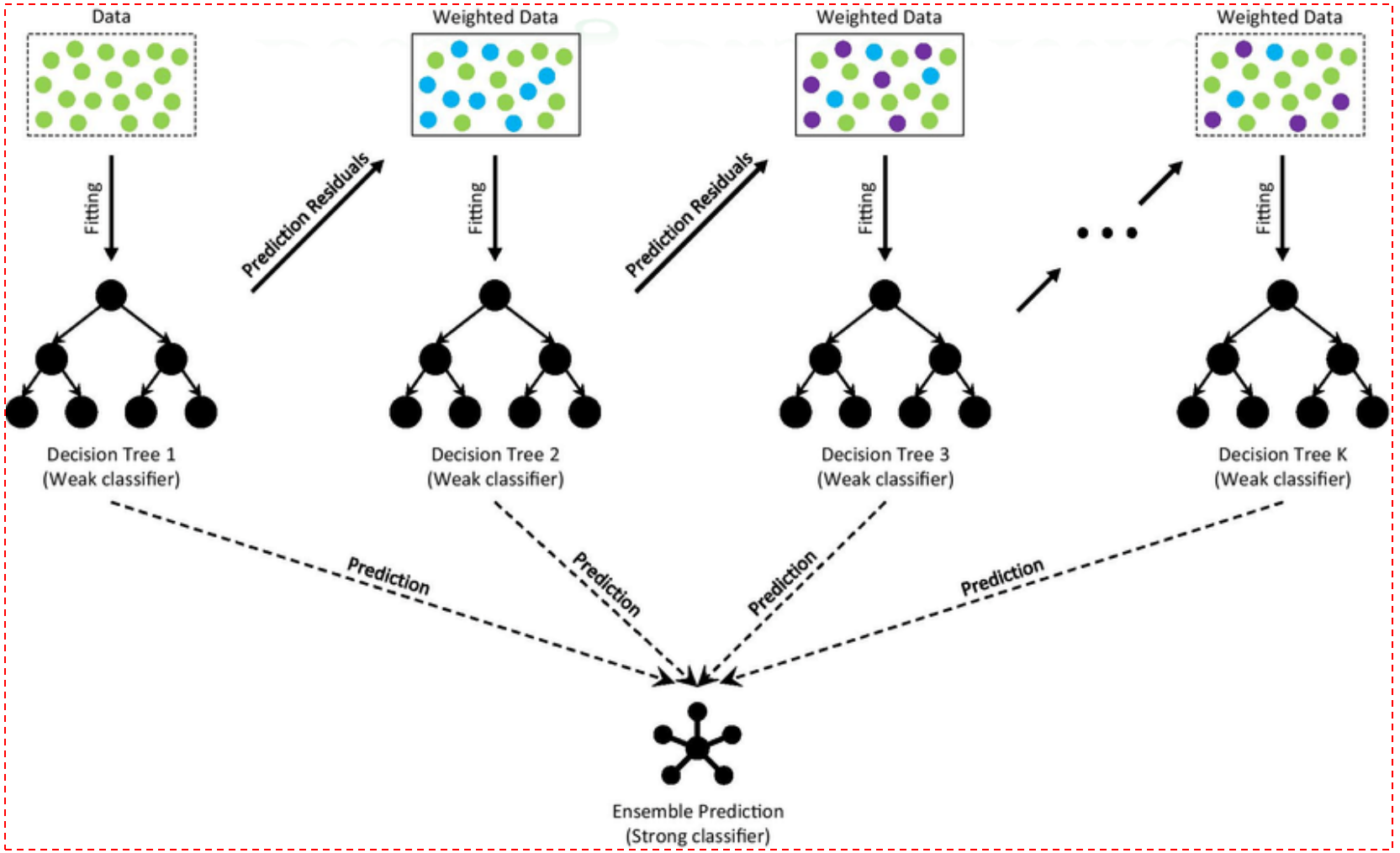
Bagging-based Method

Random Forest

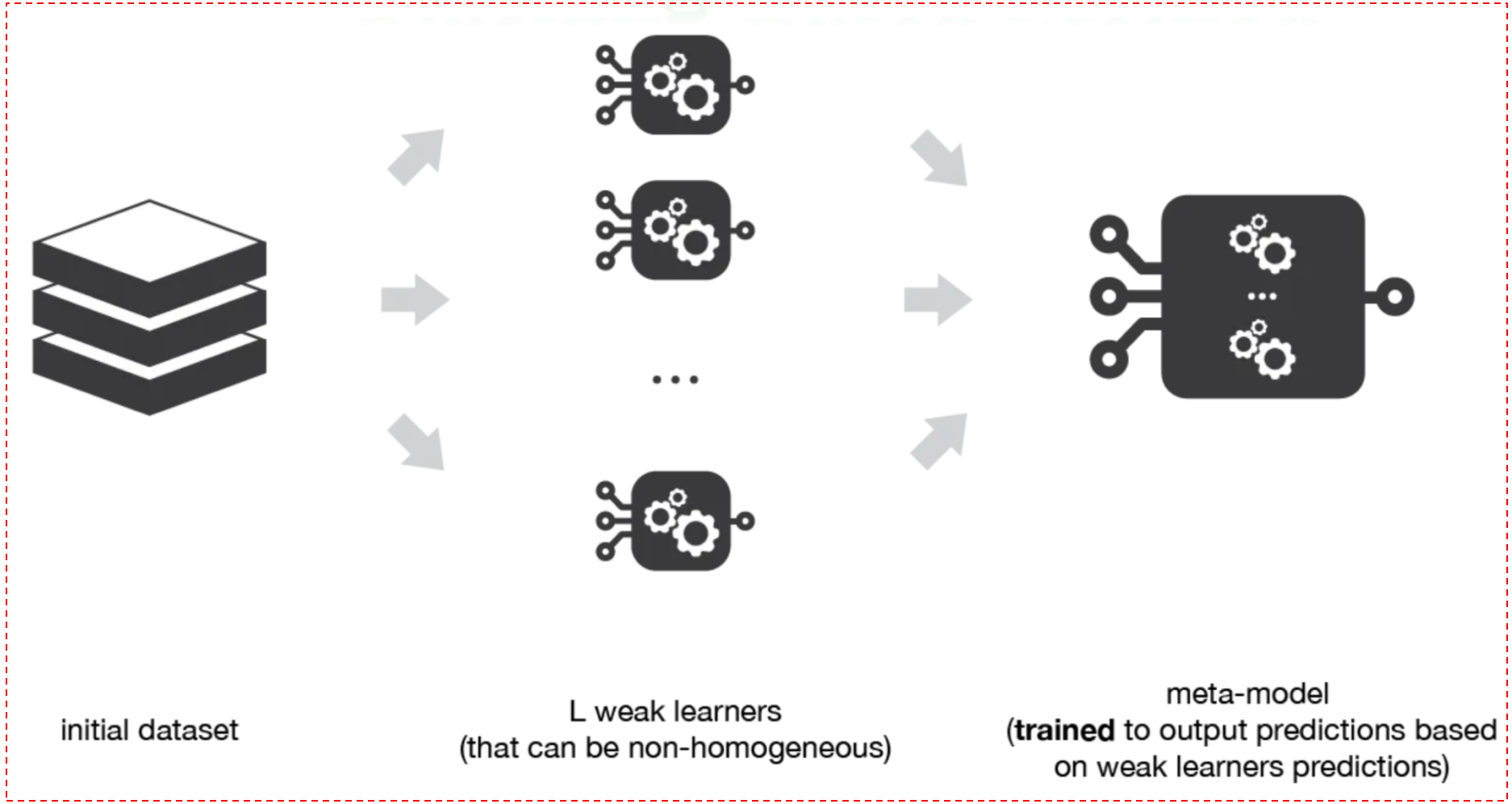


Boosting-Based Method

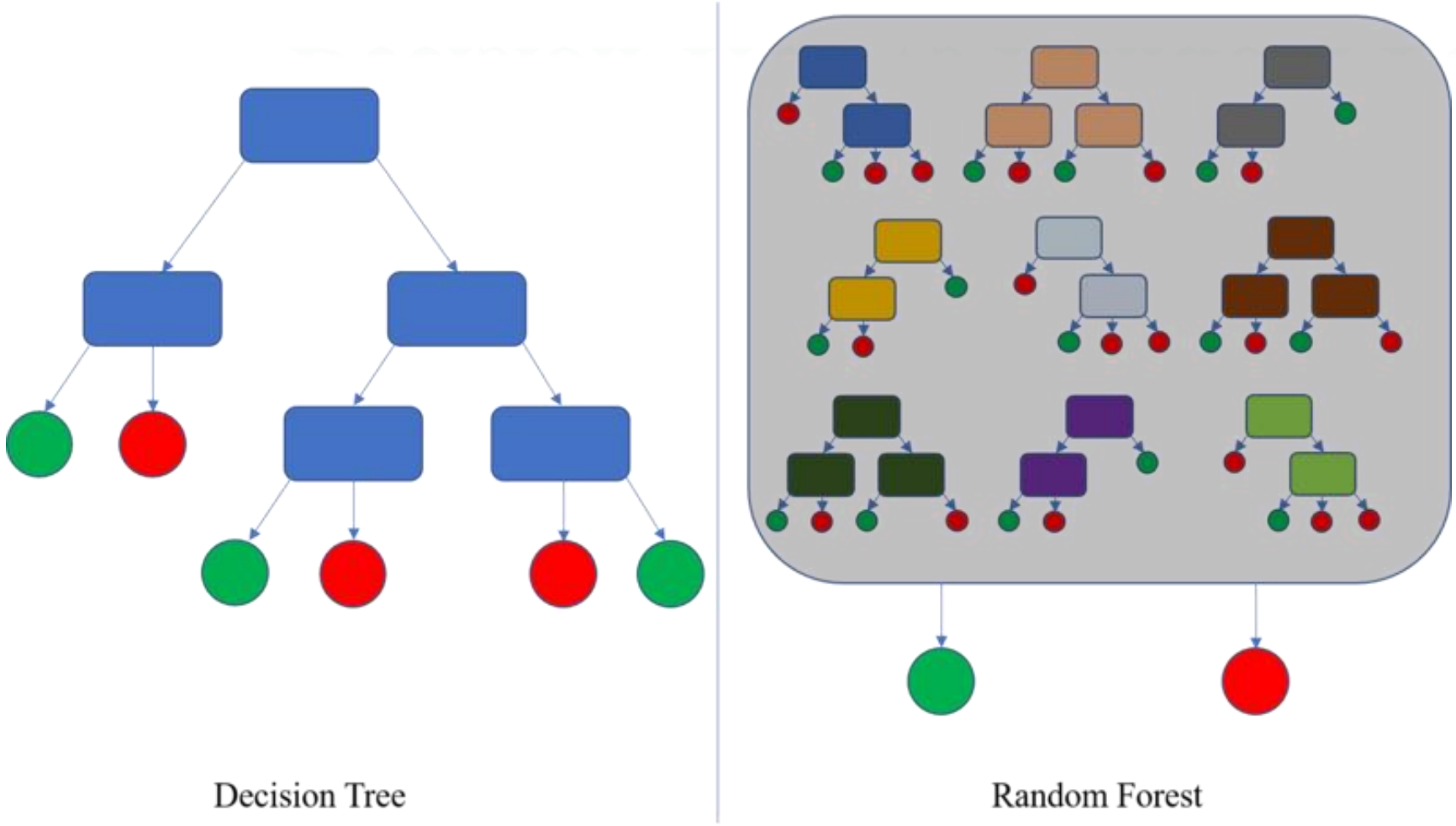
NEXT WEEK



Stacking-Based Method



Decision Tree vs Random Forest

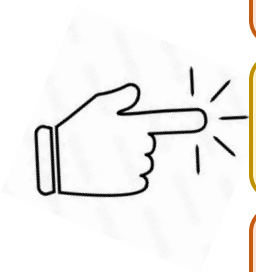


Decision Tree

Random Forest

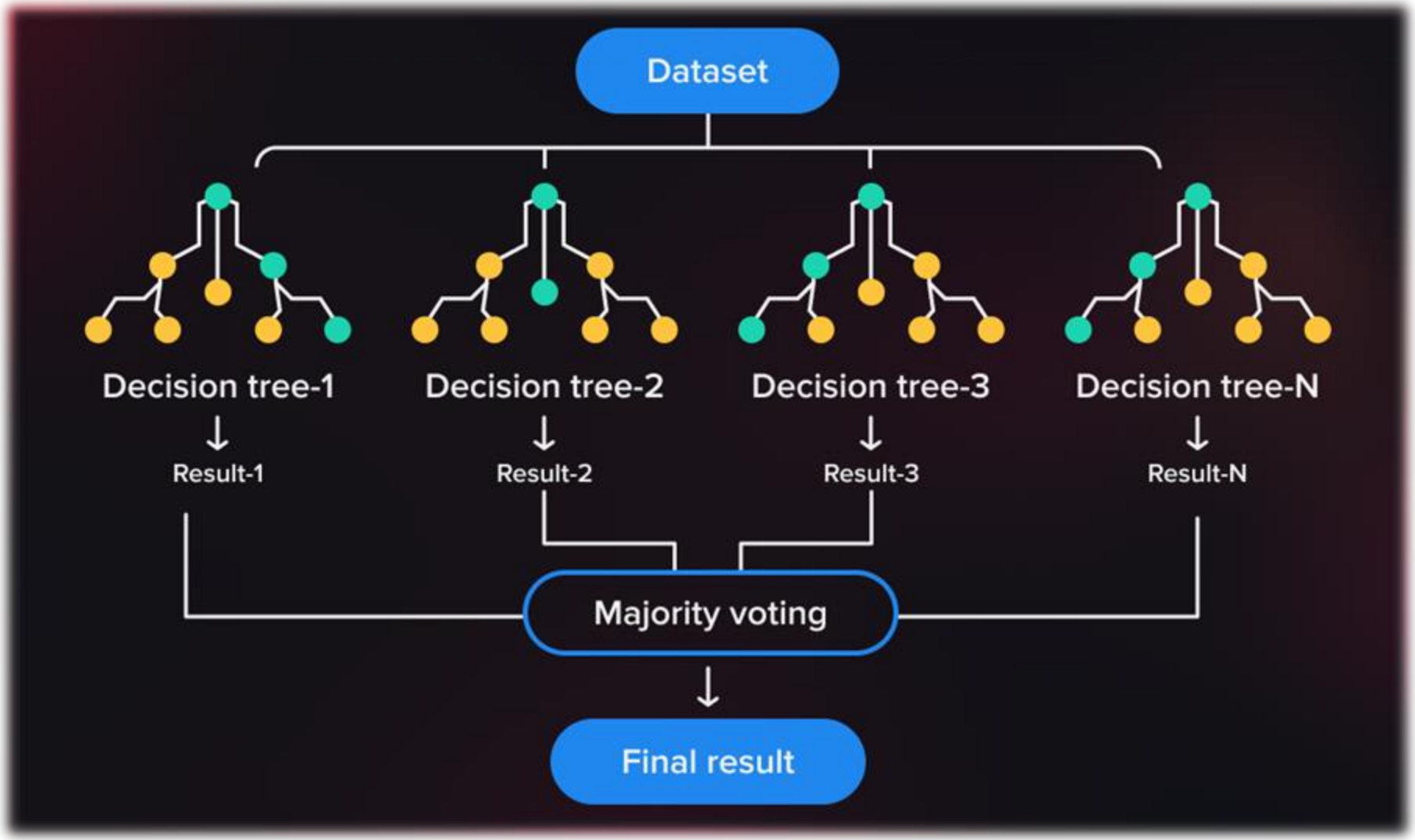
https://commons.wikimedia.org/wiki/File:Decision_Tree_vs._Random_Forest.png

Outline



- Decision Tree: Review
- Random Forest
- Fill in missing data with Random Forest
- Time series Data
- Example
- Summary

Random Forest is a Solution



Step to Random Forest

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	

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	

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: Create a New 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

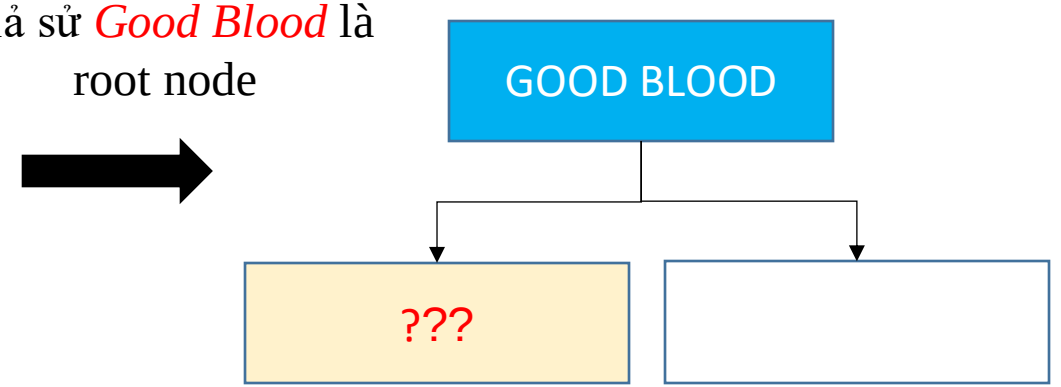
Traditional Tree

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

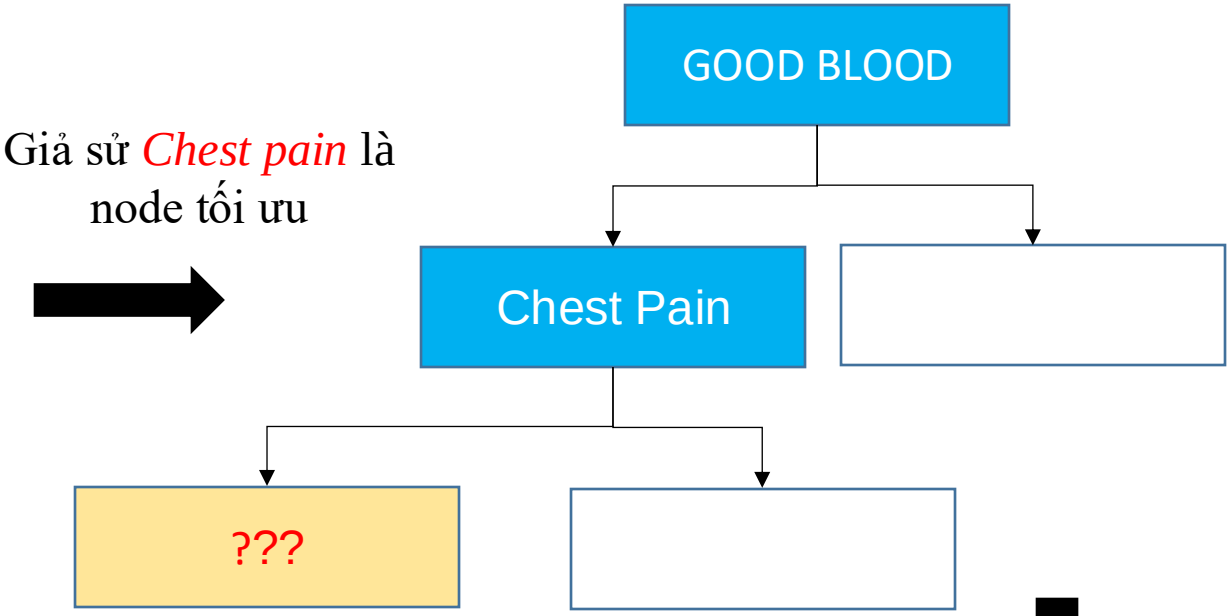
Chọn lựa ngẫu nhiên 2 features (columns)

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

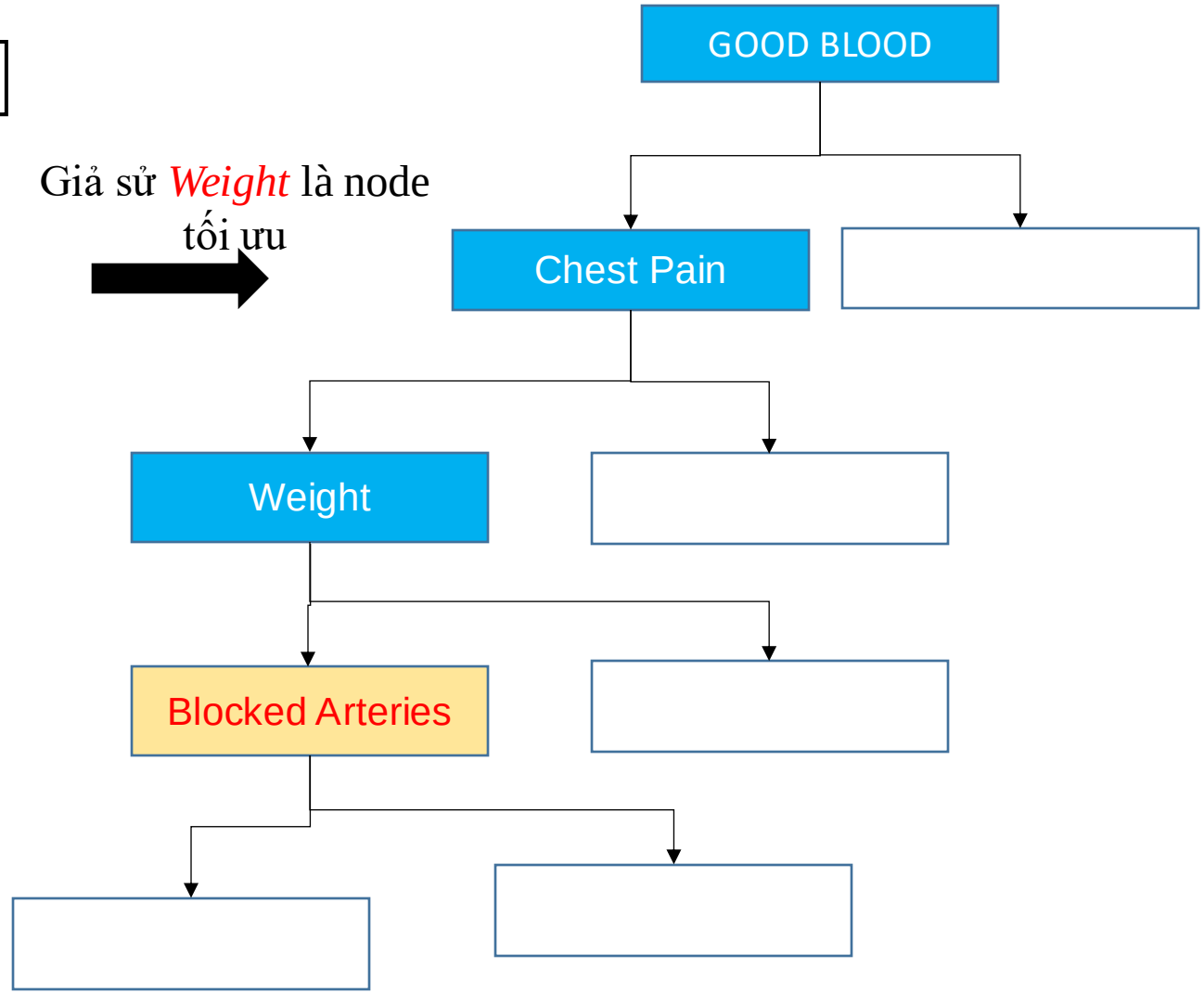
Loại bỏ *chest pain* ra khỏi dataset

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

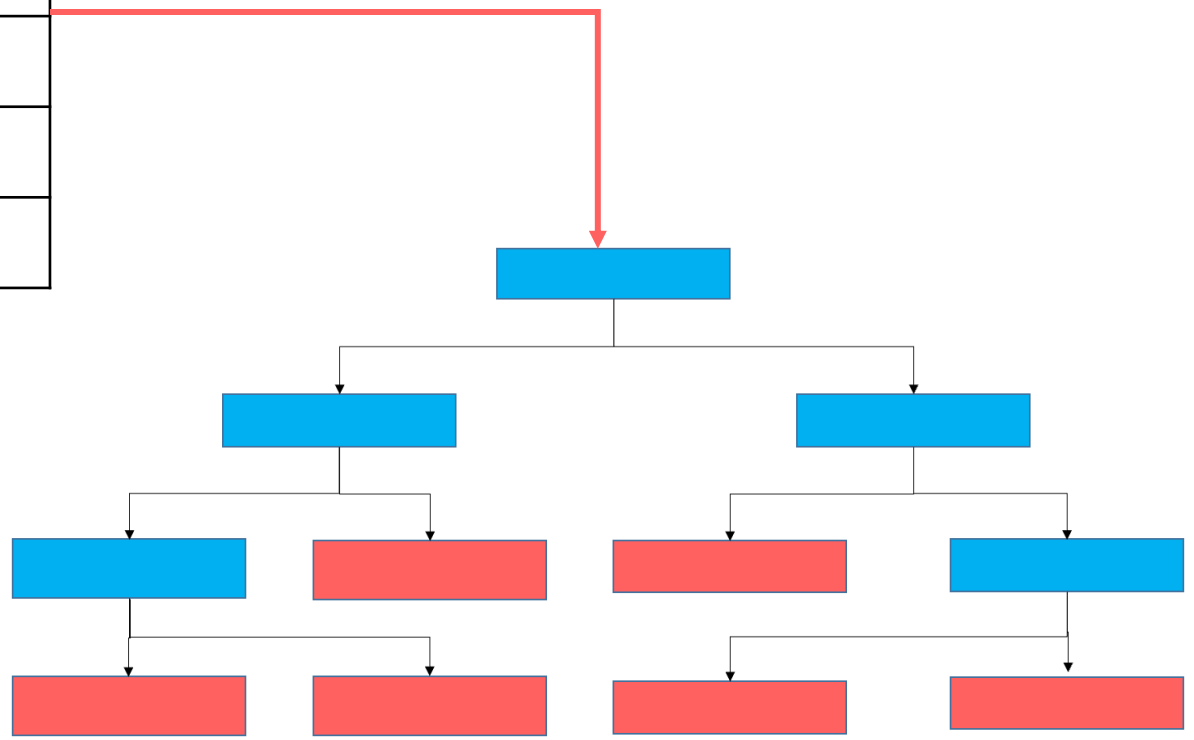


Loại bỏ *Weight* ra khỏi dataset

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

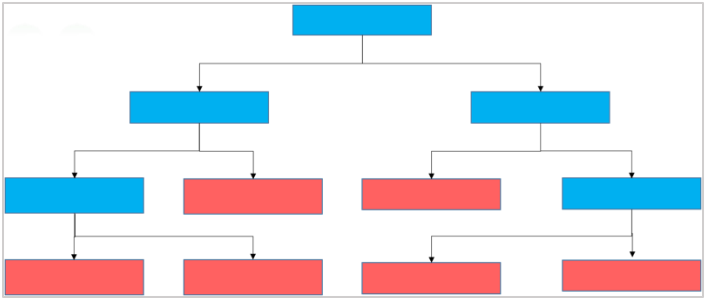
CHEST PAIN	GOOD BLOOD CIRCULATION	BLOCKED ARTERIES	WEIGHT	HEART DISEASE
YES	YES	YES	180	YES
NO	NO	NO	125	NO
YES	NO	YES	167	YES
YES	YES	NO	210	NO

...

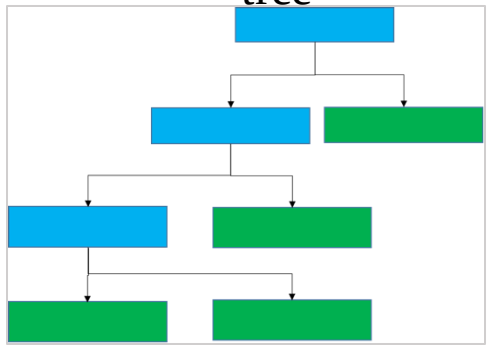
nth bootstrapped dataset

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

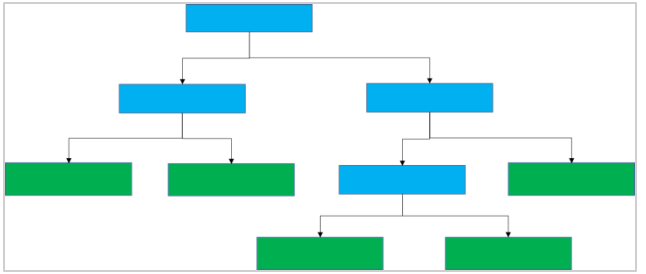
1st tree



2nd tree



nth tree



Create N Tree

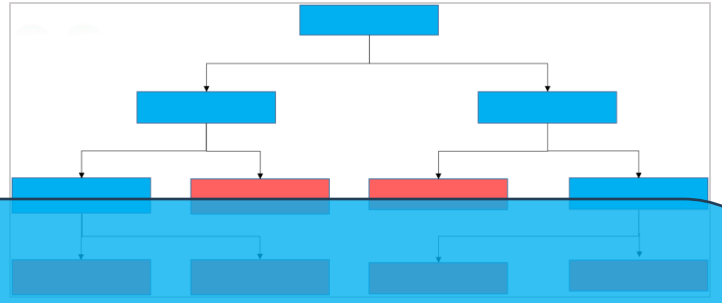
CHEST PAIN	GOOD BLOOD CIRCULATION	BLOCKED ARTERIES	WEIGHT	HEART DISEASE
NO	NO	NO	125	NO
YES	YES	YES	180	YES
YES	YES	NO	210	NO
YES	NO	YES	167	YES

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

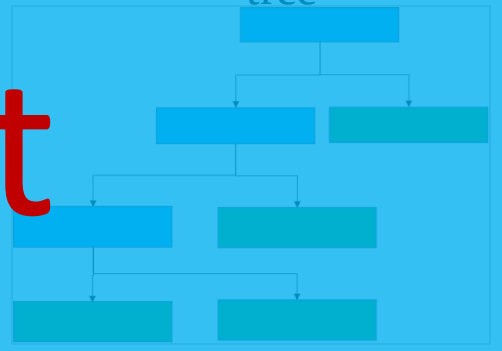
1st tree



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

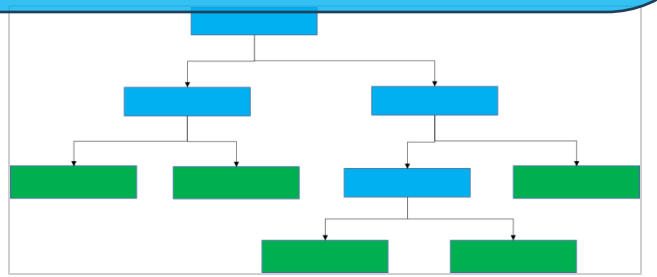


...

nth bootstrapped dataset

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

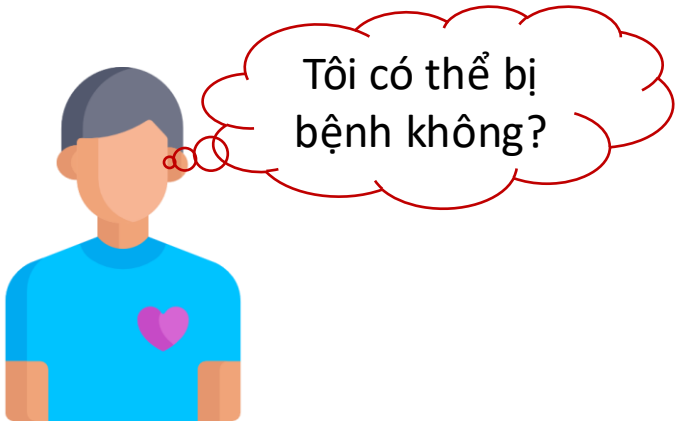
nth tree

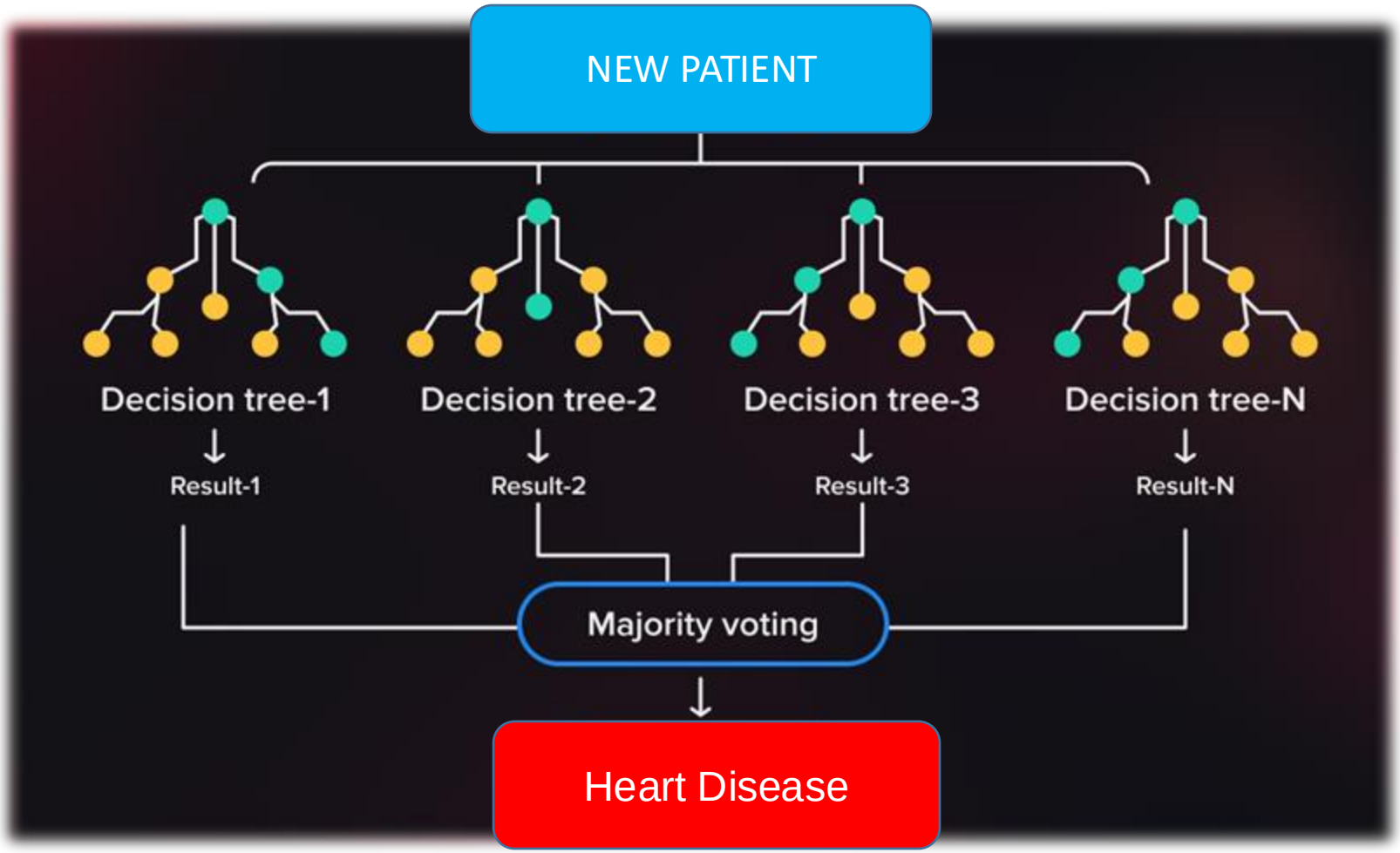


Random Forest

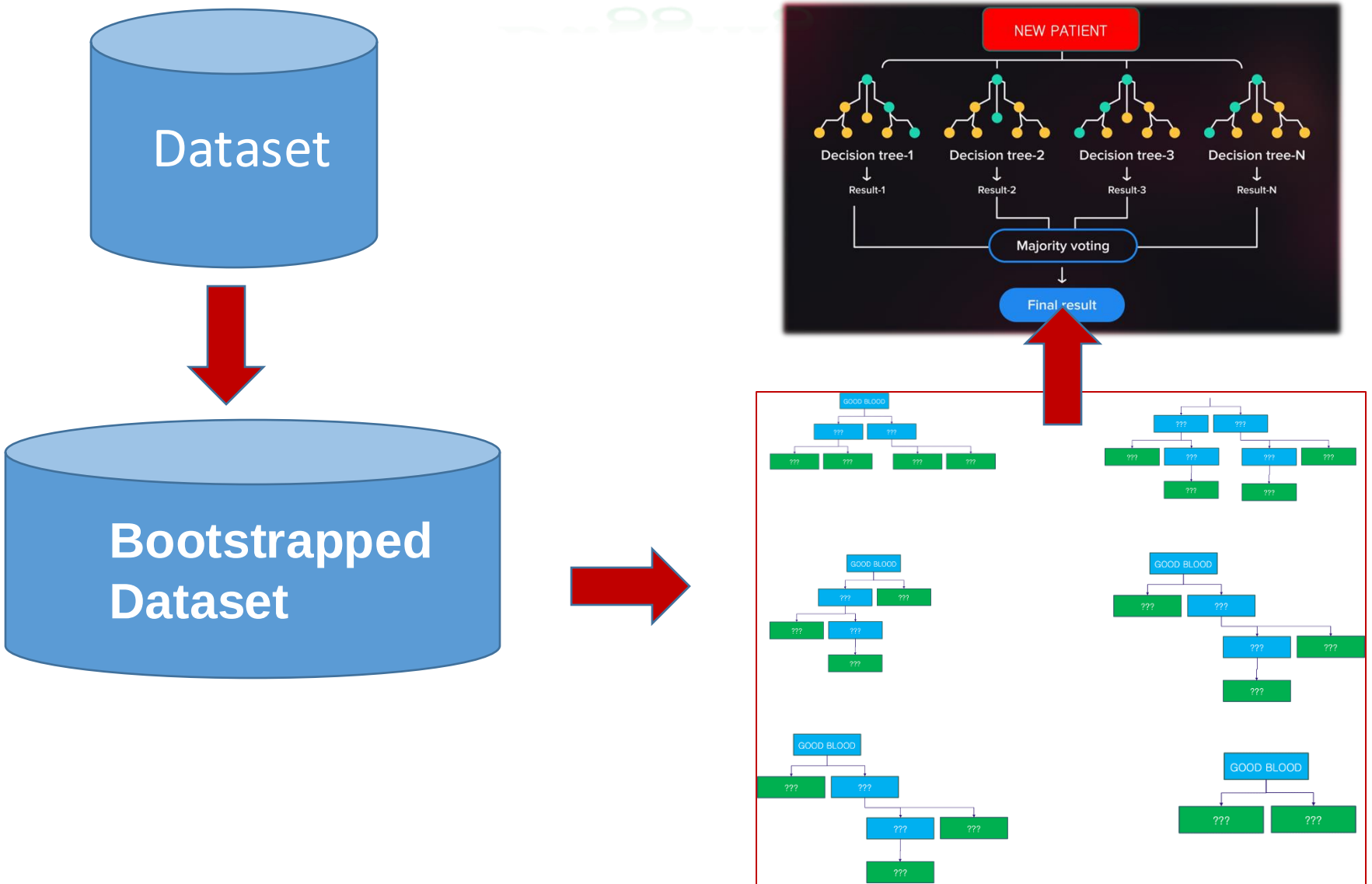
How to Predict New Sample

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



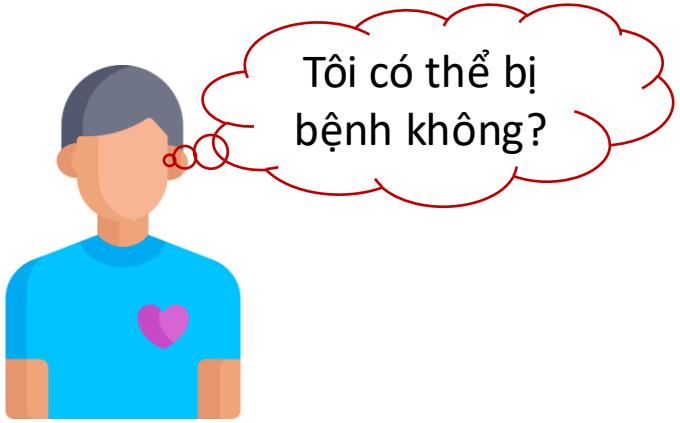


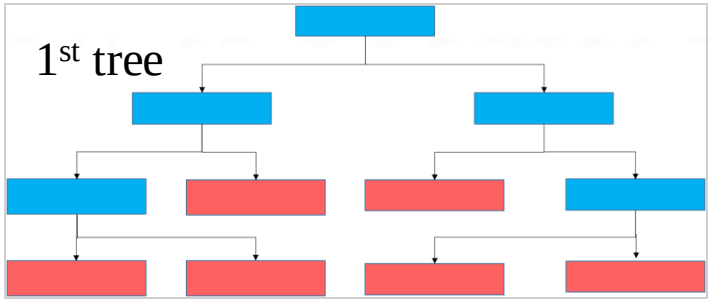
Bagging Technique



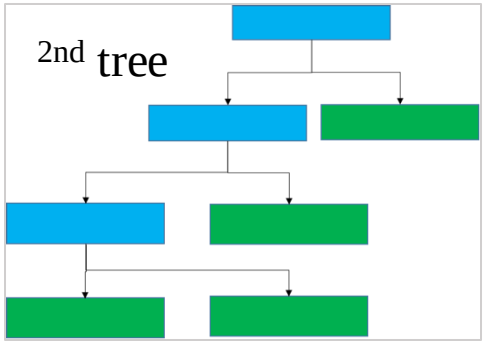
How to Predict New Sample

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

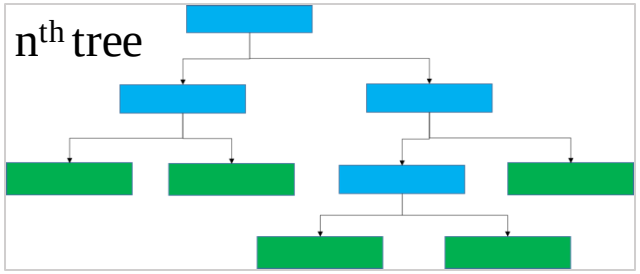




Predict



Predict



Predict

Heart Disease

Yes	No
7	2

Review

ORIGINAL 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	YES	167	YES

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
- Time series Data
- Example
- Summary

Random Forest with Missing Data

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

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

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

Training
Predict
Training
Predict
Training

Mark missing values as Predict, mark others as Training

Train the Random Forest model on the data

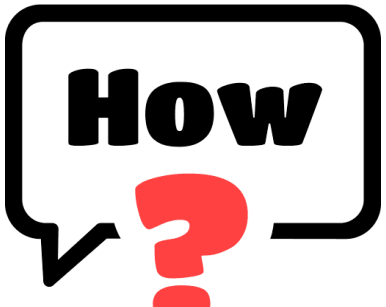
Random Forest Model

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

BUILD RANDOM FOREST

EVALUATE THE DATA FOR ALL TREES

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

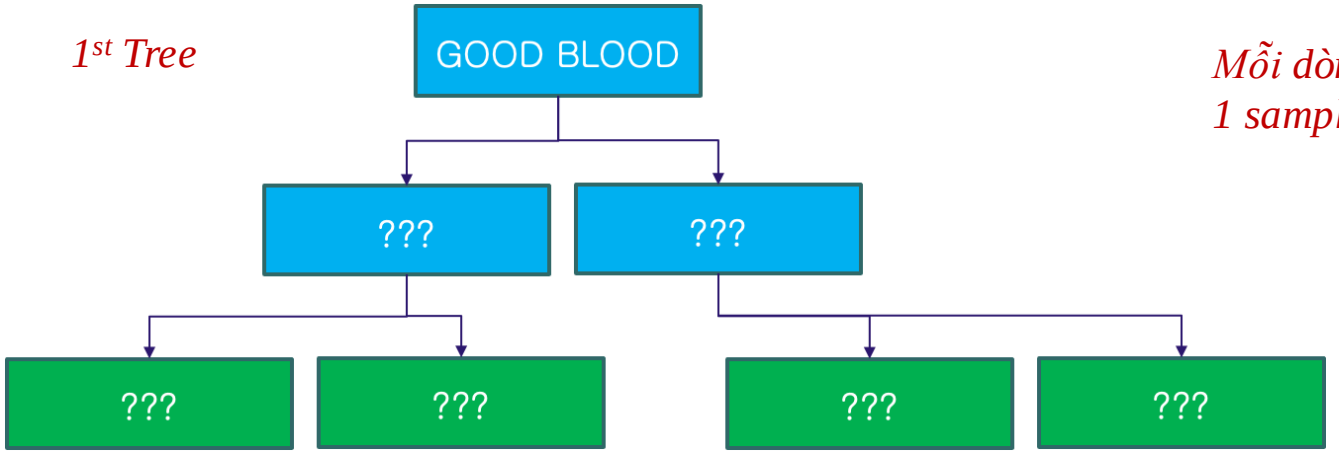
Sample 3 and sample 4 reaches to the same decision

Mỗi cột thể hiện 1 sample

	1	2	3	4
1				
2				
3				1
4			1	

Mỗi dòng thể hiện 1 sample

1st Tree

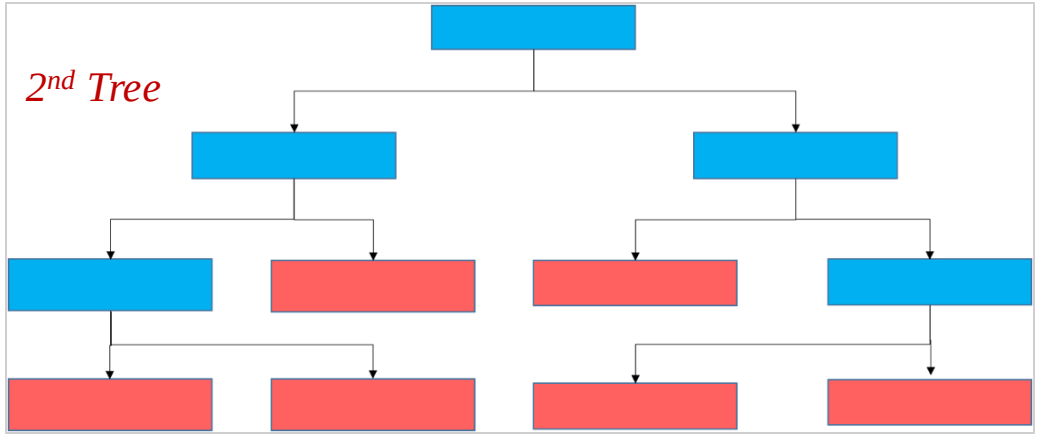


Dòng 3 và 4 cùng trả về kết quả là No

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 the 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

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	

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

Fill in the 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 the 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 the 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	

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

Fill in the 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

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

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 the 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

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 the 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

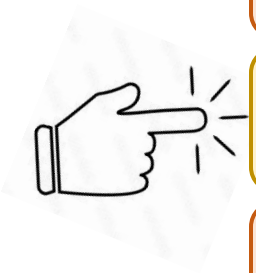
Weight s2 = 180 * 0.1= 18.0

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

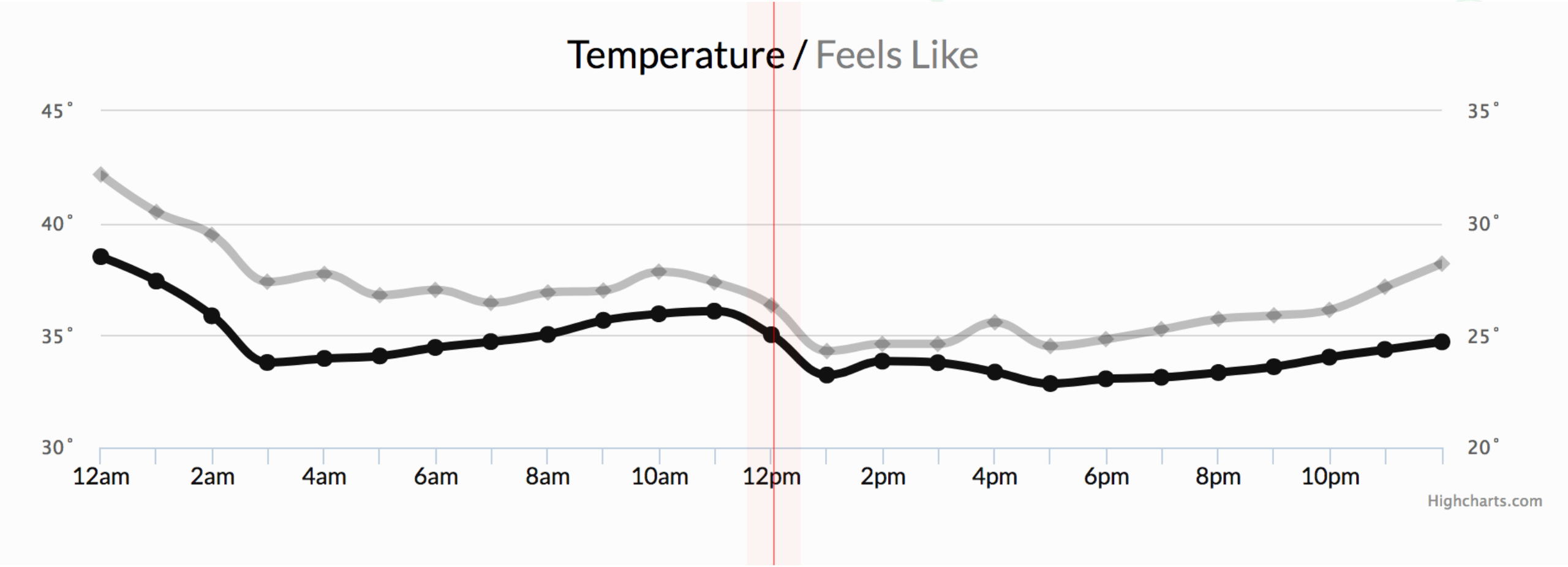
Outline

- Decision Tree: Review
- Random Forest
- Fill in missing data with Random Forest
- Time series Vs. Supervised Learning
- Example
- Summary



Time Series vs Supervised Learning

Time series data is a collection of data points over time.



AI VIET NAM
@aivietnam.edu.vn

Time Series vs Supervised Learning

A time series is a sequence of numbers that are ordered by a time index. This can be thought of as a list or column of ordered values.

1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9

A supervised learning problem is comprised of input patterns (*X*) and output patterns (*y*), such that an algorithm can learn how to predict the output patterns from the input patterns.

X, y
1 2
2, 3
3, 4
4, 5
5, 6
6, 7
7, 8
8, 9

Time Series vs Supervised Learning

Time series data can be phrased as supervised learning

Time Series data

time, measure
1, 100
2, 110
3, 108
4, 115
5, 120

Sliding Window For Time Series Data

Supervised learning

X, y
?, 100
100, 110
110, 108
108, 115
115, 120
120, ?

time, measure1, measure2
1, 0.2, 88
2, 0.5, 89
3, 0.7, 87
4, 0.4, 88
5, 1.0, 90

Sliding Window With Multivariate
Time Series Data

X1, X2, X3, y
?, ?, 0.2 , 88
0.2, 88, 0.5, 89
0.5, 89, 0.7, 87
0.7, 87, 0.4, 88
0.4, 88, 1.0, 90
1.0, 90, ?, ?

AI VIET NAM
@aivietnam.edu.vn

Time Series vs Supervised Learning

A key function to help transform time series data into a supervised learning problem is the **Pandas shift()** function.

Time Series data

time, measure
1, 100
2, 110
3, 108
4, 115
5, 120

Sliding Window For Time Series Data

Supervised learning

X, y
?, 100
100, 110
110, 108
108, 115
115, 120
120, ?

time, measure1, measure2
1, 0.2, 88
2, 0.5, 89
3, 0.7, 87
4, 0.4, 88
5, 1.0, 90

Sliding Window With Multivariate
Time Series Data

X1, X2, X3, y
?, ?, 0.2 , 88
0.2, 88, 0.5, 89
0.5, 89, 0.7, 87
0.7, 87, 0.4, 88
0.4, 88, 1.0, 90
1.0, 90, ?, ?

AI VIET NAM
@aivietnam.edu.vn

Time Series vs Supervised Learning

A key function to help transform time series data into a supervised learning problem is the **Pandas shift()** function.

```
from pandas import DataFrame
df = DataFrame()
df['t'] = [x for x in range(10)]
print(df)
```

```
   t
0  0
1  1
2  2
3  3
4  4
5  5
6  6
7  7
8  8
9  9
```

```
df['t-1'] = df['t'].shift(1)
print(df)
```

```
   t  t-1
0  0  NaN
1  1  0.0
2  2  1.0
3  3  2.0
4  4  3.0
5  5  4.0
6  6  5.0
7  7  6.0
8  8  7.0
9  9  8.0
```

AI VIET NAM
@aivietnam.edu.vn

Time Series vs Supervised Learning

A key function to help transform time series data into a supervised learning problem is the **Pandas shift()** function.

```
from pandas import DataFrame
df = DataFrame()
df['t'] = [x for x in range(10)]
print(df)
```

```
   t
0  0
1  1
2  2
3  3
4  4
5  5
6  6
7  7
8  8
9  9
```

```
df['t+1'] = df['t'].shift(-1)
print(df)
```

```
   t  t+1
0  0  1.0
1  1  2.0
2  2  3.0
3  3  4.0
4  4  5.0
5  5  6.0
6  6  7.0
7  7  8.0
8  8  9.0
9  9  NaN
```

AI VIET NAM
@aivietnam.edu.vn

Time Series vs Supervised Learning

A key function to help transform time series data into a supervised learning problem is the **Pandas shift()** function.

```python
from pandas import DataFrame
from pandas import concat

def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
  n_vars = 1 if type(data) is list else data.shape[1]
  df = DataFrame(data)
  cols, names = list(), list()
  # input sequence (t-n, ... t-1)
  for i in range(n_in, 0, -1):
    cols.append(df.shift(i))
    names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
  # forecast sequence (t, t+1, ... t+n)
  for i in range(0, n_out):
    cols.append(df.shift(-i))
    if i == 0:
      names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
    else:
      names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
  # put it all together
  agg = concat(cols, axis=1)
  agg.columns = names
  # drop rows with NaN values
  if dropnan:
    agg.dropna(inplace=True)
  return agg
```

```python
values = [x for x in range(10)]
data = series_to_supervised(values)
print(data)
```

```
   var1(t-1)  var1(t)
1        0.0        1
2        1.0        2
3        2.0        3
4        3.0        4
5        4.0        5
6        5.0        6
7        6.0        7
8        7.0        8
9        8.0        9
```

Time Series vs Supervised Learning

One-Step Univariate Forecasting

```python
from pandas import DataFrame
from pandas import concat

def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
  n_vars = 1 if type(data) is list else data.shape[1]
  df = DataFrame(data)
  cols, names = list(), list()
  # input sequence (t-n, ... t-1)
  for i in range(n_in, 0, -1):
    cols.append(df.shift(i))
    names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
  # forecast sequence (t, t+1, ... t+n)
  for i in range(0, n_out):
    cols.append(df.shift(-i))
    if i == 0:
      names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
    else:
      names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
  # put it all together
  agg = concat(cols, axis=1)
  agg.columns = names
  # drop rows with NaN values
  if dropnan:
    agg.dropna(inplace=True)
  return agg
```

```python
values = [x for x in range(10)]
data = series_to_supervised(values)
print(data)
```

```
   var1(t-1)  var1(t)
1        0.0        1
2        1.0        2
3        2.0        3
4        3.0        4
5        4.0        5
6        5.0        6
7        6.0        7
8        7.0        8
9        8.0        9
```

AI VIET NAM
@aivietnam.edu.vn

Time Series vs Supervised Learning

Multi-Step or Sequence Forecasting

```python
from pandas import DataFrame
from pandas import concat

def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
  n_vars = 1 if type(data) is list else data.shape[1]
  df = DataFrame(data)
  cols, names = list(), list()
  # input sequence (t-n, ... t-1)
  for i in range(n_in, 0, -1):
    cols.append(df.shift(i))
    names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
  # forecast sequence (t, t+1, ... t+n)
  for i in range(0, n_out):
    cols.append(df.shift(-i))
    if i == 0:
      names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
    else:
      names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
  # put it all together
  agg = concat(cols, axis=1)
  agg.columns = names
  # drop rows with NaN values
  if dropnan:
    agg.dropna(inplace=True)
  return agg
```

```python
values = [x for x in range(10)]
data = series_to_supervised(values, 2, 2)
print(data)
```

	var1(t-2)	var1(t-1)	var1(t)	var1(t+1)
2	0.0	1.0	2	3.0
3	1.0	2.0	3	4.0
4	2.0	3.0	4	5.0
5	3.0	4.0	5	6.0
6	4.0	5.0	6	7.0
7	5.0	6.0	7	8.0
8	6.0	7.0	8	9.0

AI VIET NAM
@aivietnam.edu.vn

Time Series vs Supervised Learning

Multivariate Forecasting

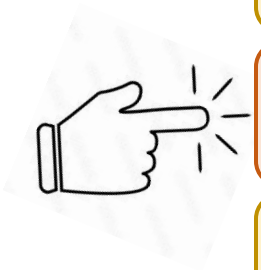
```python
from pandas import DataFrame
from pandas import concat

def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
  n_vars = 1 if type(data) is list else data.shape[1]
  df = DataFrame(data)
  cols, names = list(), list()
  # input sequence (t-n, ... t-1)
  for i in range(n_in, 0, -1):
    cols.append(df.shift(i))
    names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
  # forecast sequence (t, t+1, ... t+n)
  for i in range(0, n_out):
    cols.append(df.shift(-i))
    if i == 0:
      names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
    else:
      names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
  # put it all together
  agg = concat(cols, axis=1)
  agg.columns = names
  # drop rows with NaN values
  if dropnan:
    agg.dropna(inplace=True)
  return agg
```

```python
raw = DataFrame()
raw['ob1'] = [x for x in range(10)]
raw['ob2'] = [x for x in range(50, 60)]
values = raw.values
data = series_to_supervised(values)
print(data)
```

	var1(t-1)	var2(t-1)	var1(t)	var2(t)
1	0.0	50.0	1	51
2	1.0	51.0	2	52
3	2.0	52.0	3	53
4	3.0	53.0	4	54
5	4.0	54.0	5	55
6	5.0	55.0	6	56
7	6.0	56.0	7	57
8	7.0	57.0	8	58
9	8.0	58.0	9	59

Outline

- Decision Tree: Review
- Random Forest
- Fill in missing data with Random Forest
- Time series Vs. Supervised Learning
- Example
- Summary



AI VIET NAM
@aivietnam.edu.vn

Example

The daily female births dataset, that is the monthly births across three years

daily-total-female-

Date	Births
1959-01-01	35
1959-01-02	32
1959-01-03	30
1959-01-04	31
1959-01-05	44
1959-01-06	29
1959-01-07	45
1959-01-08	43
1959-01-09	38
1959-01-10	27
1959-01-11	38
1959-01-12	33
1959-01-13	55
1959-01-14	47
1959-01-15	45
1959-01-16	37

We will use only the previous six time steps as input to the model

```
# load the dataset
series = read_csv('daily-total-female-births.csv', header=0, index_col=0)
```

```
[42] values = series.values
     # transform the time series data into supervised learning
     train = series_to_supervised(values, n_in=6)
```

```
print(train)
```

```
[[35. 32. 30. ... 44. 29. 45.]
 [32. 30. 31. ... 29. 45. 43.]
 [30. 31. 44. ... 45. 43. 38.]
 ...
 [40. 38. 44. ... 37. 52. 48.]
 [38. 44. 34. ... 52. 48. 55.]
 [44. 34. 37. ... 48. 55. 50.]]
```

Example

The daily female births dataset, that is the monthly births across three years

daily-total-female-

Date	Births
1959-01-01	35
1959-01-02	32
1959-01-03	30
1959-01-04	31
1959-01-05	44
1959-01-06	29
1959-01-07	45
1959-01-08	43
1959-01-09	38
1959-01-10	27
1959-01-11	38
1959-01-12	33
1959-01-13	55
1959-01-14	47
1959-01-15	45
1959-01-16	37

We will use only the previous six time steps as input to the model

```
# split into input and output columns
trainX, trainy = train[:, :-1], train[:, -1]
# fit model
model = RandomForestRegressor(n_estimators=1000)
model.fit(trainX, trainy)
# construct an input for a new prediction
row = values[-6:].flatten()
# make a one-step prediction
yhat = model.predict(asarray([row]))
print('Input: %s, Predicted: %.3f' % (row, yhat[0]))
```

```
Input: [34 37 52 48 55 50], Predicted: 42.541
```

k-Fold cross-validation

1. Split the dataset into k equal (if possible) parts (they are called folds)

2.Choose k – 1 folds as the training set. The remaining fold will be the test set

3.Train the model on the training set. On each iteration of cross-validation, you must train a new model independently of the model trained on the previous iteration

4.Validate on the test set

5.Save the result of the validation

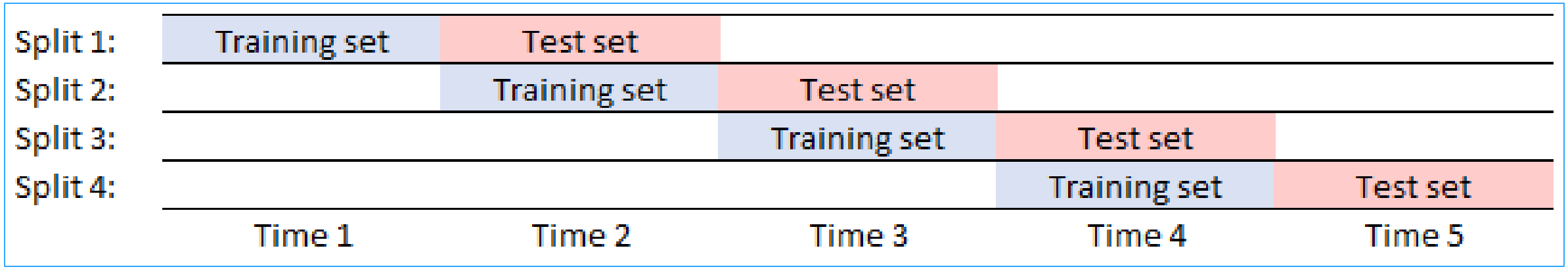
6.Repeat steps 3 – 6 k times. Each time use the remaining fold as the test set. In the end, you should have validated the model on every fold that you have.

7.To get the final score average the results that you got on step 6.

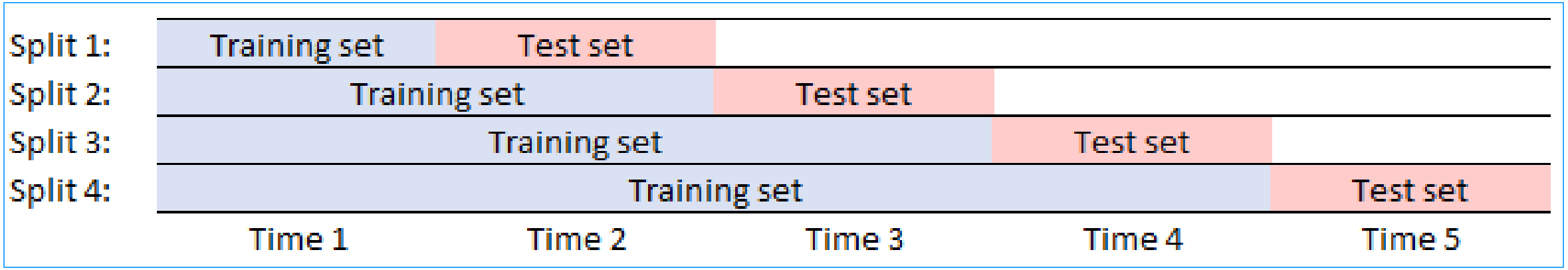
Time Series Cross Validation

With time series data, we cannot shuffle the data!

Rolling Window



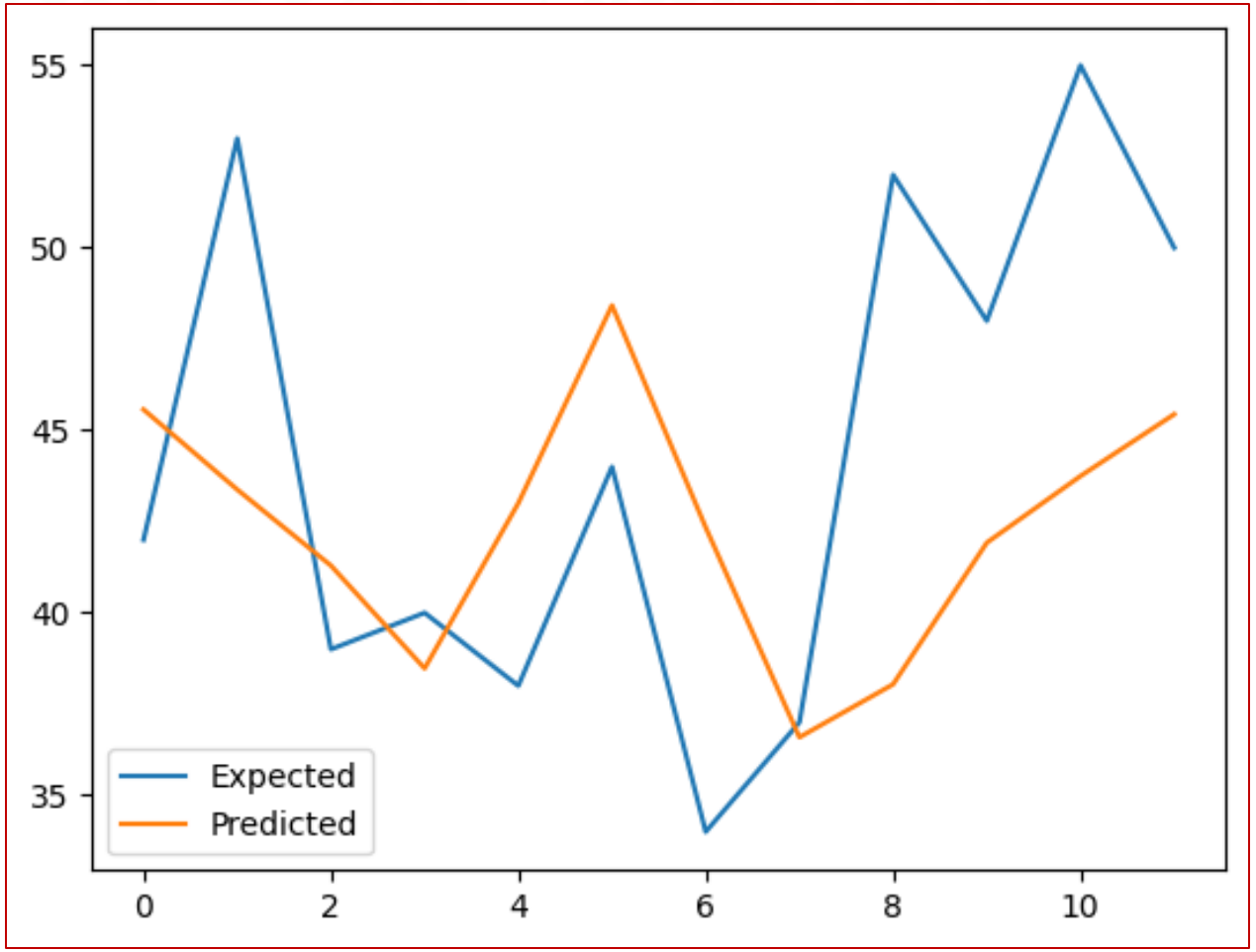
Expanding Window



Time Series Cross Validation

With time series data, we cannot shuffle the data!

```
# evaluate
mae, y, yhat = walk_forward_validation(data, 12)
print('MAE: %.3f' % mae)
# plot expected vs predicted
pyplot.plot(y, label='Expected')
pyplot.plot(yhat, label='Predicted')
pyplot.legend()
pyplot.show()
```



Time Series Cross Validation

Outline

- Decision Tree: Review
- Random Forest
- Fill in missing data with Random Forest
- Time series Vs. Supervised Learning
- Example
- Summary