



# **RANDOM FOREST for CLASSIFICATION AND REGRESSION**

*Giảng Viên Hướng Dẫn*  
TS. Vũ Tiến Dũng

*Học Viên:*  
Tạ Văn Nhân

**VNU - HUS**

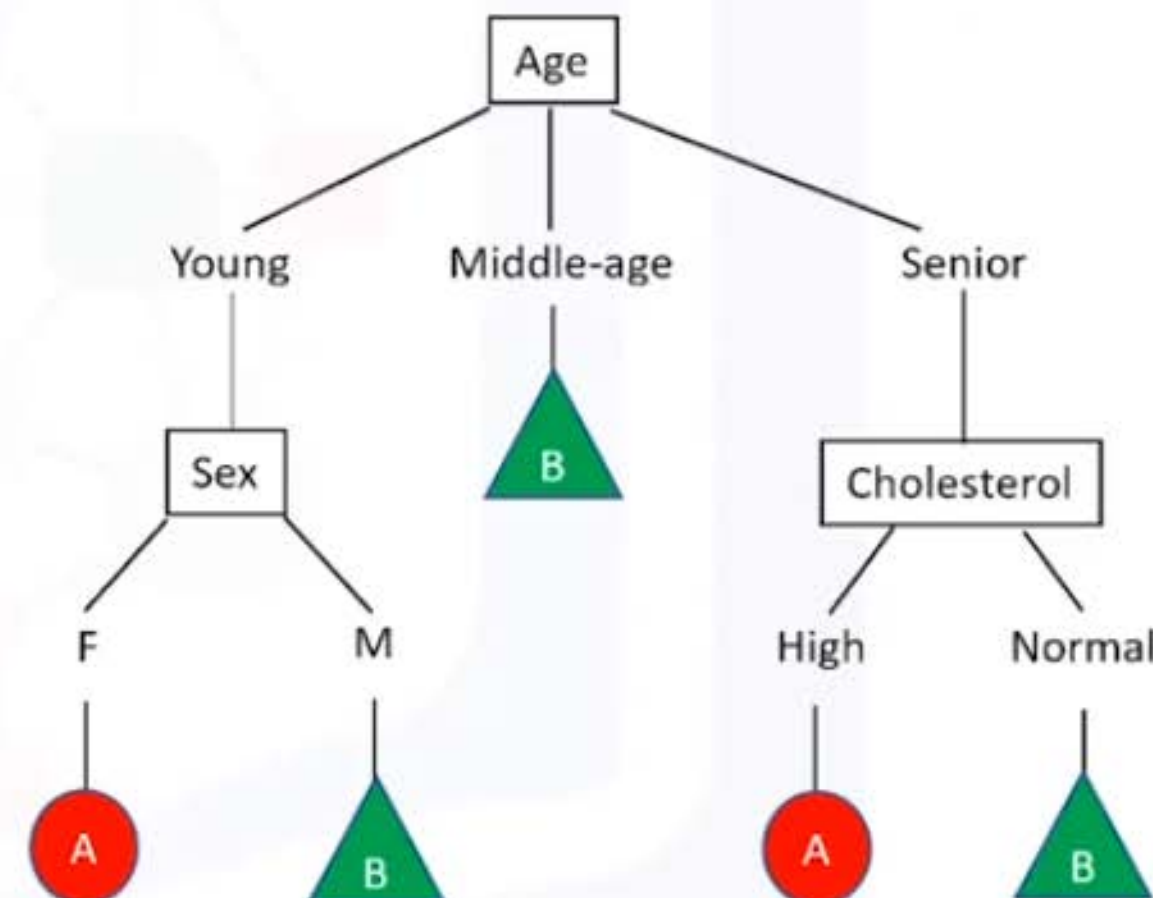
# NỘI DUNG

- CÂY QUYẾT ĐỊNH (DECISION TREE)
  - BÀI TOÁN PHÂN LOẠI
  - BÀI TOÁN HỒI QUY
- RỪNG NGẪU NHIÊN (RANDOM FOREST)

# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

Patient ID	Age	Sex	BP	Cholesterol	Drug
p1	Young	F	High	Normal	Drug A
p2	Young	F	High	High	Drug A
p3	Middle-age	F	High	Normal	Drug B
p4	Senior	F	Normal	Normal	Drug B
p5	Senior	M	Low	Normal	Drug B
p6	Senior	M	Low	High	Drug A
p7	Middle-age	M	Low	High	Drug B
p8	Young	F	Normal	Normal	Drug A
p9	Young	M	Low	Normal	Drug B
p10	Senior	M	Normal	Normal	Drug B
p11	Young	M	Normal	High	Drug B
p12	Middle-age	F	Normal	High	Drug B
p13	Middle-age	M	High	Normal	Drug B
p14	Senior	F	Normal	High	Drug A
p15	Middle-age	F	Low	Normal	?



- Chia tập huấn luyện thành các nút riêng biệt.
- Mỗi nút có thể chứa một loại dữ liệu.

# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

Các bước thực hiện:

**Bước 1:** Chọn một thuộc tính từ tập dữ liệu.

**Bước 2:** Tính toán thuộc tính trội hơn để chia dữ liệu.

**Bước 3:** Chia dữ liệu dựa trên giá trị của thuộc tính tốt nhất.

**Bước 4:** Quay trở lại bước 1.

# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

**Bước 2:** Tính toán thuộc tính trội hơn.

**Entropy** của nút đại diện cho sự hỗn loạn của nút:

$$Entropy = -P(A)\log(P(A)) - P(B)\log(P(B))$$

Entropy càng nhỏ thì:

- Phân phối càng ít đều.
- Nút càng ít hỗn loạn hay càng đồng nhất.




# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

**Bước 2:** Tính toán thuộc tính trội hơn. Ví dụ:

Patient ID	Age	Sex	BP	Cholesterol	Drug
p1	Young	F	High	Normal	Drug A
p2	Young	F	High	High	Drug A
p3	Middle-age	F	High	Normal	Drug B
p4	Senior	F	Normal	Normal	Drug B
p5	Senior	M	Low	Normal	Drug B
p6	Senior	M	Low	High	Drug A
p7	Middle-age	M	Low	High	Drug B
p8	Young	F	Normal	Normal	Drug A
p9	Young	M	Low	Normal	Drug B
p10	Senior	M	Normal	Normal	Drug B
p11	Young	M	Normal	High	Drug B
p12	Middle-age	F	Normal	High	Drug B
p13	Middle-age	M	High	Normal	Drug B
p14	Senior	F	Normal	High	Drug A

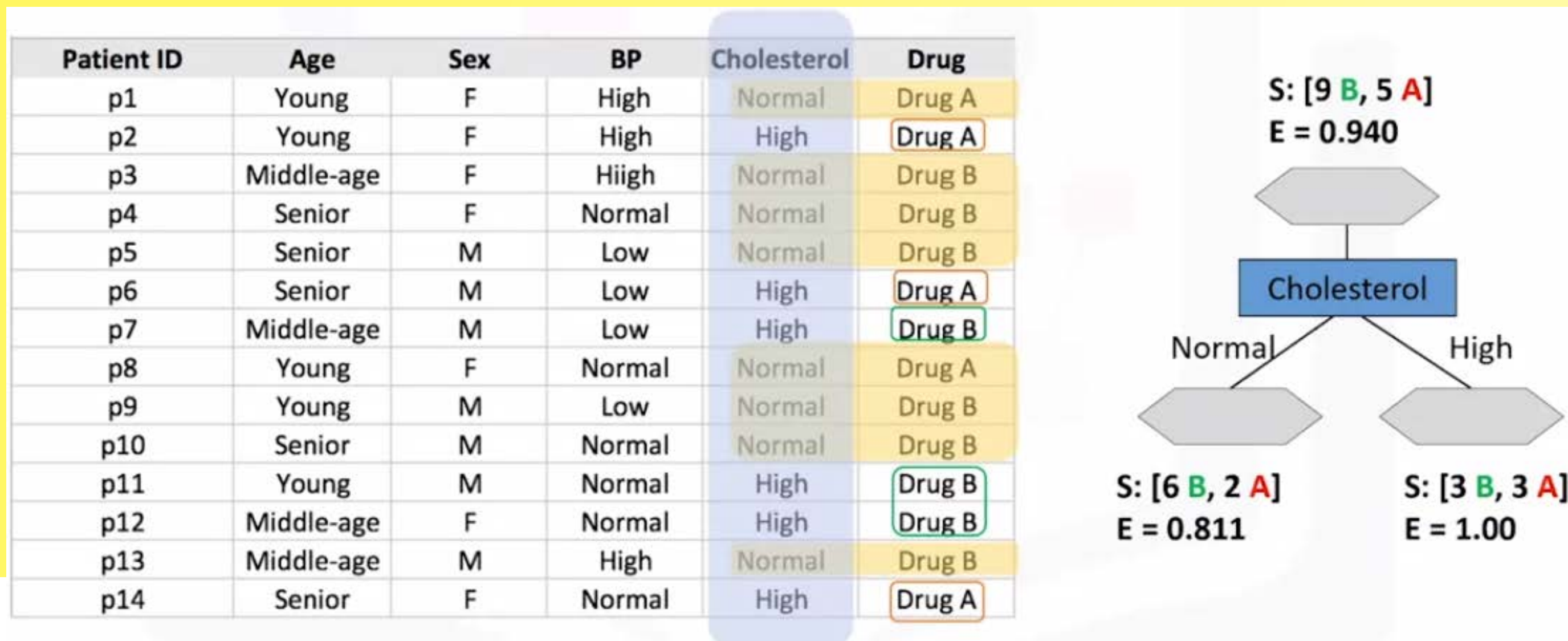
S: [9 B, 5 A]  
 $E = -p(B)\log(p(B)) - p(A)\log(p(A))$   
 $E = -(9/14)\log(9/14) - (5/14)\log(5/14)$   
**E = 0.940**



# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

**Bước 2:** Tính toán thuộc tính trội hơn. Ví dụ:



# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

**Bước 2:** Tính toán thuộc tính trội hơn. Ví dụ:

Patient ID	Age	Sex	BP	Cholesterol	Drug
p1	Young	F	High	Normal	Drug A
p2	Young	F	High	High	Drug A
p3	Middle-age	F	High	Normal	Drug B
p4	Senior	F	Normal	Normal	Drug B
p5	Senior	M	Low	Normal	Drug B
p6	Senior	M	Low	High	Drug A
p7	Middle-age	M	Low	High	Drug B
p8	Young	F	Normal	Normal	Drug A
p9	Young	M	Low	Normal	Drug B
p10	Senior	M	Normal	Normal	Drug B
p11	Young	M	Normal	High	Drug B
p12	Middle-age	F	Normal	High	Drug B
p13	Middle-age	M	High	Normal	Drug B
p14	Senior	F	Normal	High	Drug A

Decision Tree Diagram:

- Root Node: Sex
  - F: S: [3 B, 4 A], E = 0.985
  - M: S: [6 B, 1 A], E = 0.592



# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

**Bước 2:** Tính toán thuộc tính trội hơn.

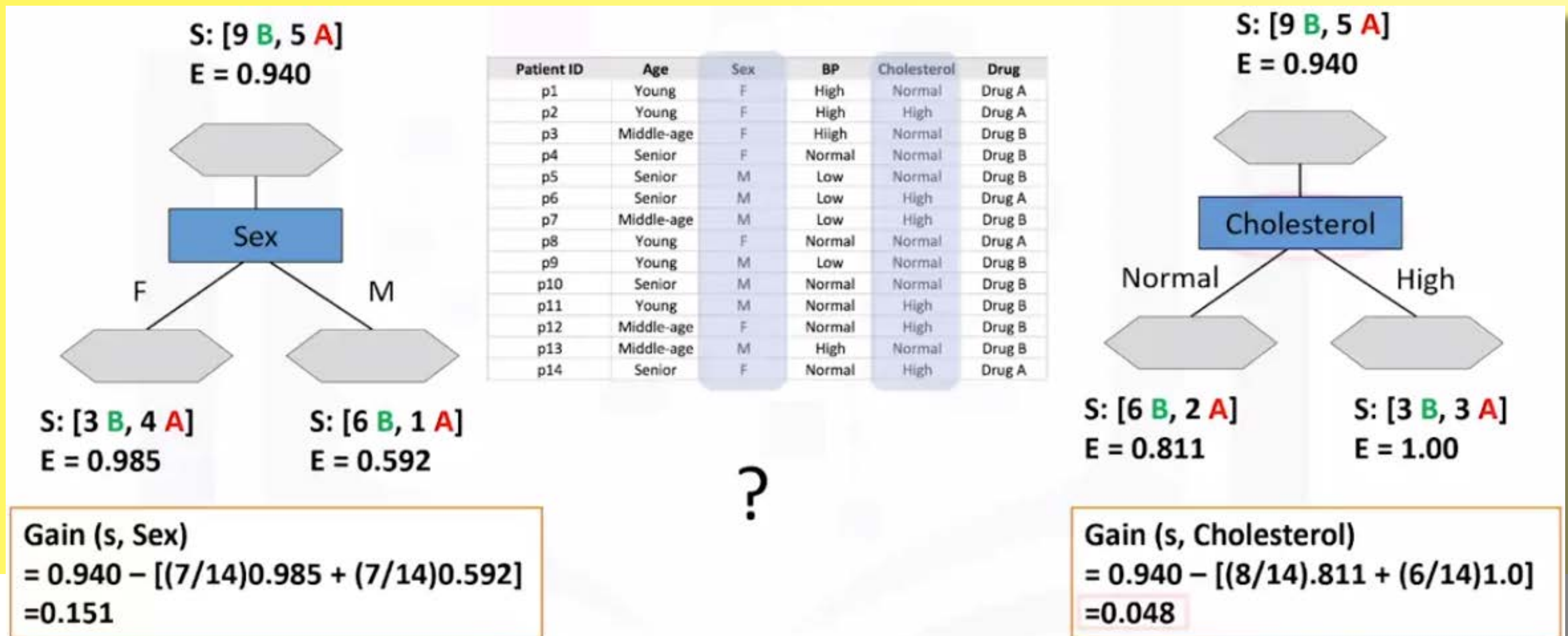
- **Information gain:** là thông tin làm tăng mức độ tin tưởng sau khi chia.
- *Lựa chọn thuộc tính thu nhận được nhiều thông tin hơn sau khi chia.*

**Information Gain** = (Entropy trước khi chia)  
- (Entropy sau khi chia được đánh trọng số)

# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

Bước 2: Tính toán thuộc tính trội hơn. Ví dụ:



# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

Ví dụ: với dữ liệu drug200.csv.

```
1 my_data = pd.read_csv("drug200.csv", delimiter=",")  
2 my_data[0:5]
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY

# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

Cần chuyển các biến phân loại thành các biến chỉ định.

```
1 from sklearn import preprocessing
2 le_sex = preprocessing.LabelEncoder()
3 le_sex.fit(['F', 'M'])
4 X[:,1] = le_sex.transform(X[:,1])
```

```
array([[23, 0, 0, 0, 25.355],
       [47, 1, 1, 0, 13.093],
       [47, 1, 1, 0, 10.113999999999999],
       [28, 0, 2, 0, 7.797999999999999]])
```



# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại

**max\_depth:** chiều sâu lớn nhất của cây. Nếu None thì các nút được mở rộng cho đến khi tất cả các lá có entropy = 0 hoặc các lá chứa ít hơn min\_samples\_split mẫu.

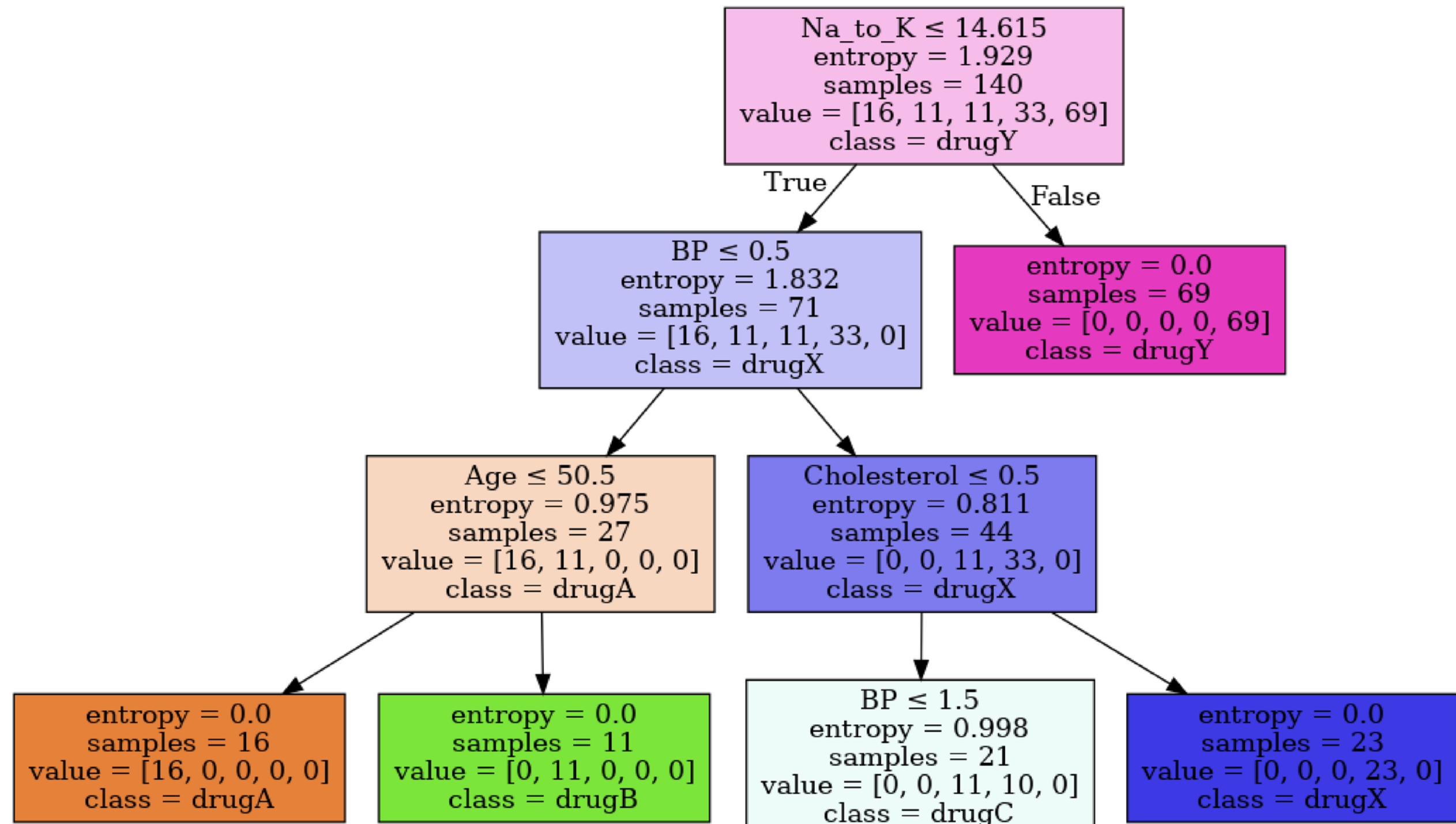
**min\_impurity\_split:** kết thúc sớm sự phát triển của cây.

```
1 drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
2 drugTree # it shows the default parameters
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                        splitter='best')
```

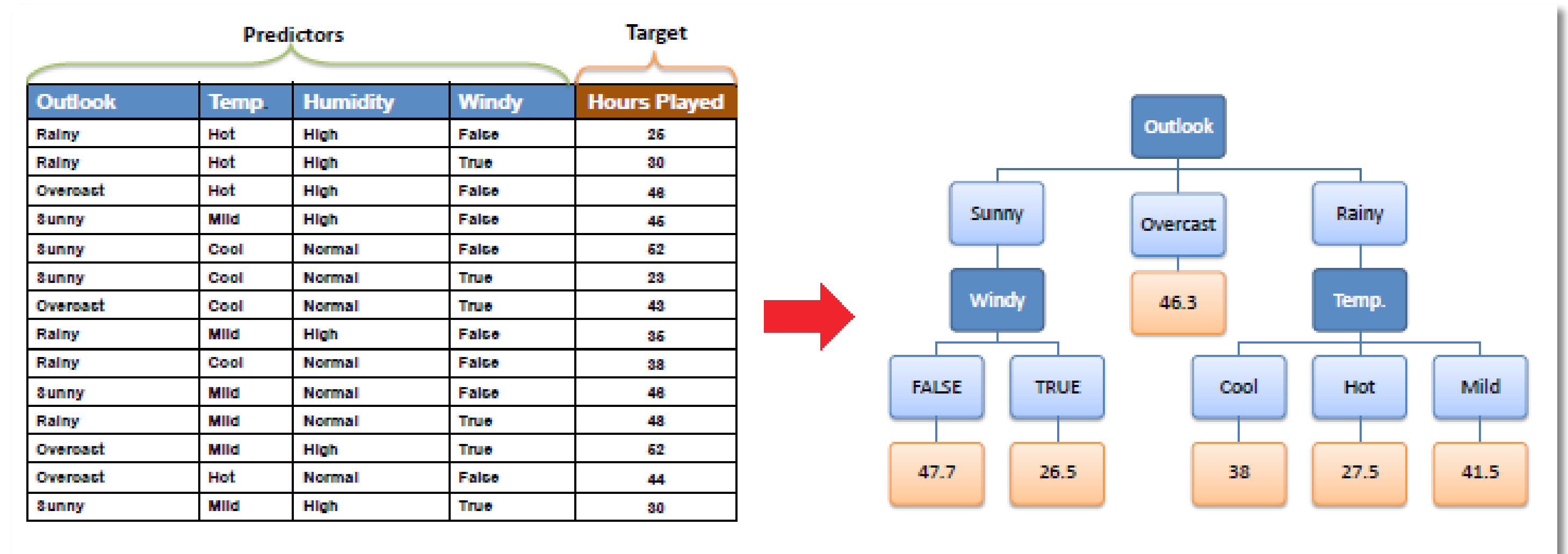
# DECISION TREE

## Cây Quyết Định Cho Bài Toán Phân Loại



# DECISION TREE

## Cây Quyết Định Cho Bài Toán Hồi Quy



- Chia tập huấn luyện thành các nút riêng biệt.
- Mỗi nút có thể chứa một loại dữ liệu.

# DECISION TREE

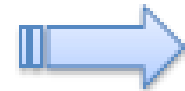
## Cây Quyết Định Cho Bài Toán Hồi Quy

Dựa trên thuật toán ID3 của J. R. Quinlan. Thay vì sử dụng Entropy để tính sự hỗn loạn của mẫu, chúng ta sẽ sử dụng độ lệch chuẩn để tính sự đồng nhất của mẫu.

Hours Played
25
30
46
45
52
23
43
35
38
46
48
52
44
30

$$\text{Count} = n = 14$$

$$\text{Average} = \bar{x} = \frac{\sum x}{n} = 39.8$$


$$\text{Standard Deviation} = S = \sqrt{\frac{\sum (x - \bar{x})^2}{n}} = 9.32$$

$$\text{Coefficient of Variation} = CV = \frac{S}{\bar{x}} * 100\% = 23\%$$



# DECISION TREE

## Cây Quyết Định Cho Bài Toán Hồi Quy

**Bước 2:** Tính toán thuộc tính trội hơn.

- **Standard Deviation Reduction:** là độ giảm độ lệch chuẩn sau khi chia dữ liệu với một thuộc tính nào đó.
- *Lựa chọn thuộc tính với standard deviation reduction lớn nhất sau khi chia.*

**Standard Deviation Reduction** = (STD trước khi chia)  
- (STD sau khi chia được đánh trọng số)

# DECISION TREE

## Cây Quyết Định Cho Bài Toán Hồi Quy

**Bước 2:** Tính toán thuộc tính trội hơn. Ví dụ:

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

**SDR**(Hours , Outlook)

= **S**(Hours ) – **S**(Hours, Outlook)

= 9.32 – 7.66 = 1.66

**S**(Hours, Outlook) = **P**(Sunny)\***S**(Sunny) + **P**(Overcast)\***S**(Overcast) + **P**(Rainy)\***S**(Rainy)

= (4/14)\*3.49 + (5/14)\*7.78 + (5/14)\*10.87

= 7.66

# DECISION TREE

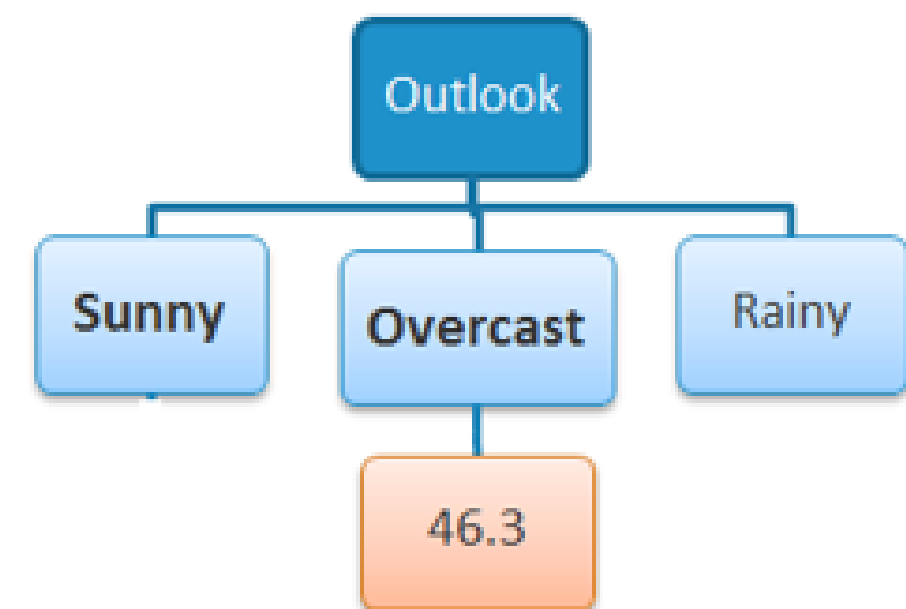
## Cây Quyết Định Cho Bài Toán Hồi Quy

**Coefficient of Variation (CV):** là threshold để quyết định dừng sự phát triển của cây.

**Average (Avg):** là giá trị của lá.

Outlook - Overcast

		Hours Played (StDev)	Hours Played (AVG)	Hours Played (CV)	Count	
Outlook	Overcast	3.49	46.3	8%	4	46
	Rainy	7.78	35.2	22%	5	43
	Sunny	10.87	39.2	28%	5	52
						44

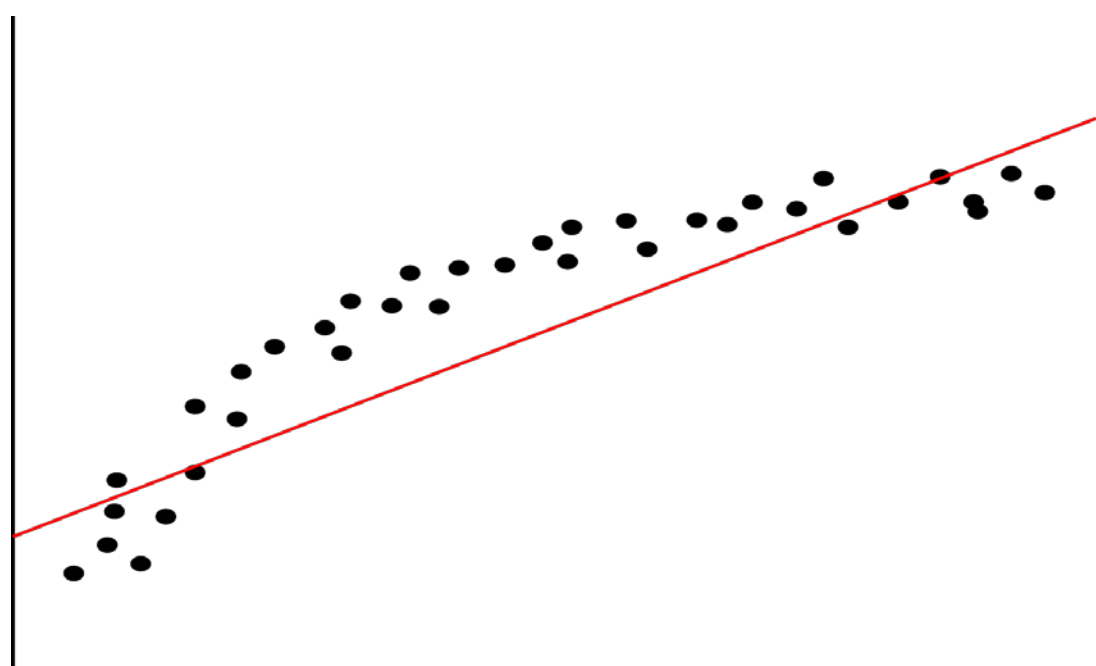


# RANDOM FOREST

**Nhóm phương pháp "ensemble learning":** kết hợp nhiều mô hình được huấn luyện cho việc giải quyết cùng một bài toán để đạt được kết quả tốt hơn.

**Có 3 loại sai số dự đoán:** bias, variance, irreducible.

**1. Bias error:** sự khác nhau giữa giá trị dự đoán và giá trị thực. Ví dụ ở dưới cho thấy mô hình high bias, underfitting.

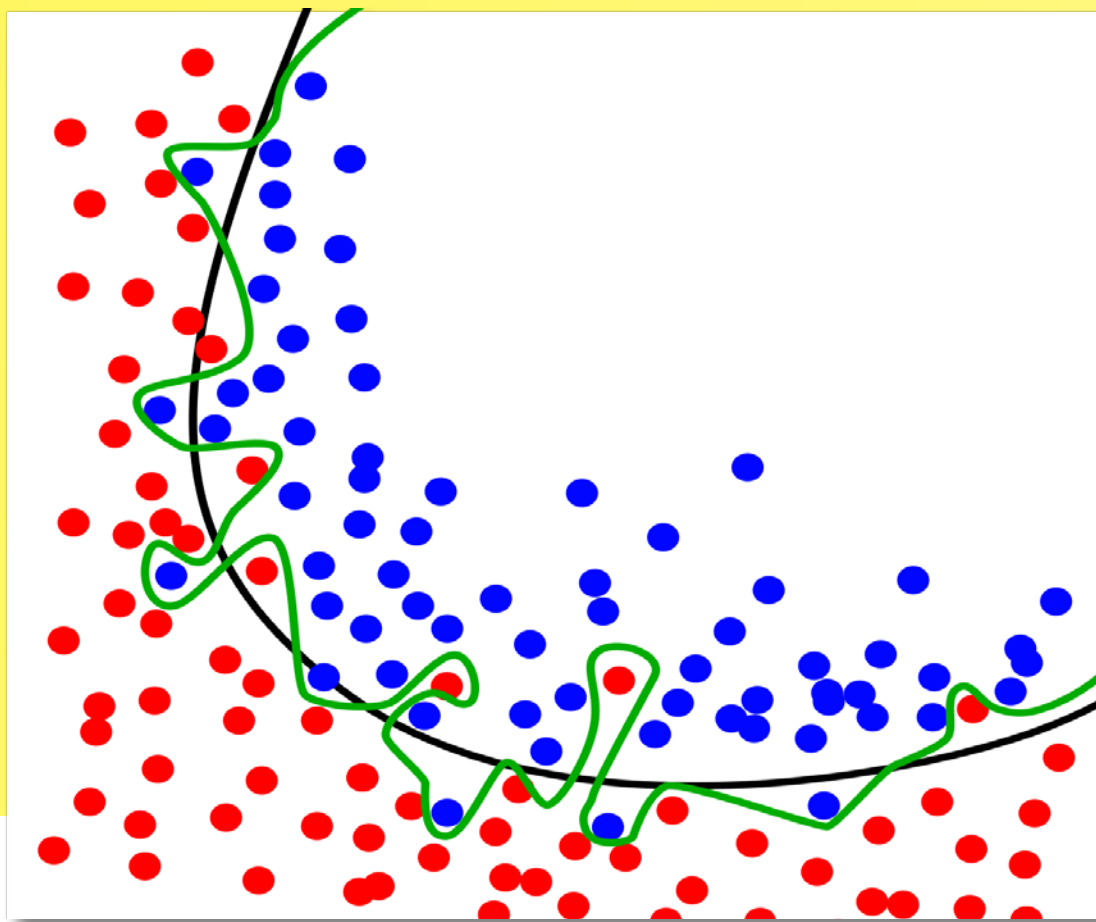




# RANDOM FOREST

**Có 3 loại sai số dự đoán:** bias, variance, irreducible.

**2. Variance error:** sự thay đổi của dự đoán mô hình cho một điểm dữ liệu nhất định. Hình dưới là mô hình high variance, overfitting.

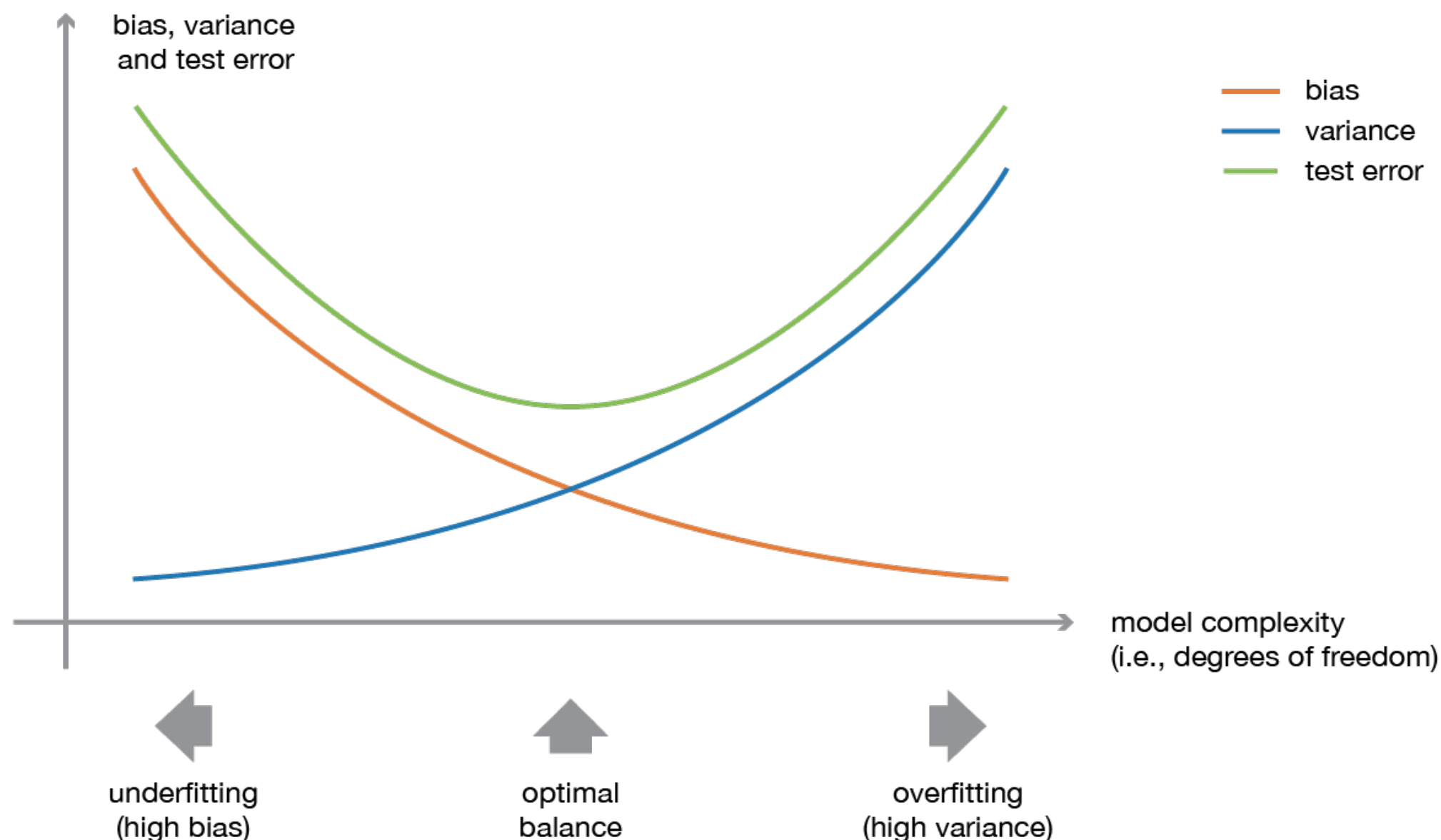


**3. The irreducible error:** là nhiễu không thể giảm được bởi bất kỳ mô hình nào.

*Bias và Variance tương đương với underfitting và overfitting.*

# RANDOM FOREST

Khi bias giảm thì variance tăng và ngược lại. Do đó cần có sự cân đối giữa bias và variance trong một mô hình phức tạp.



# RANDOM FOREST

**Bagging:** xem xét các *mô hình cơ bản* (học yếu) *đồng nhất, độc lập, học song song* với nhau, và kết hợp chúng theo một số quy trình tính trung bình nhất định.

*Bagging tập trung vào việc tạo ra một mô hình kết hợp với sai số phương sai nhỏ.*

**Cây quyết định:** là một mô hình cơ bản.

- Cây nông: variance thấp và bias cao .
- Cây sâu: bias thấp nhưng variance cao.

# RANDOM FOREST

**Bootstrapping:** tạo ra **các mẫu** có kích thước  $B$  (mẫu bootstrap) từ bộ dữ liệu ban đầu có kích thước  $N$  bằng cách lấy ngẫu nhiên có hoàn lại.

**$N$  phải đủ lớn để:**

- Lấy mẫu từ  $N$  xấp xỉ tốt như lấy mẫu từ phân phối thực.
- Các mẫu  $B$  không quá tương quan (gần như độc lập).

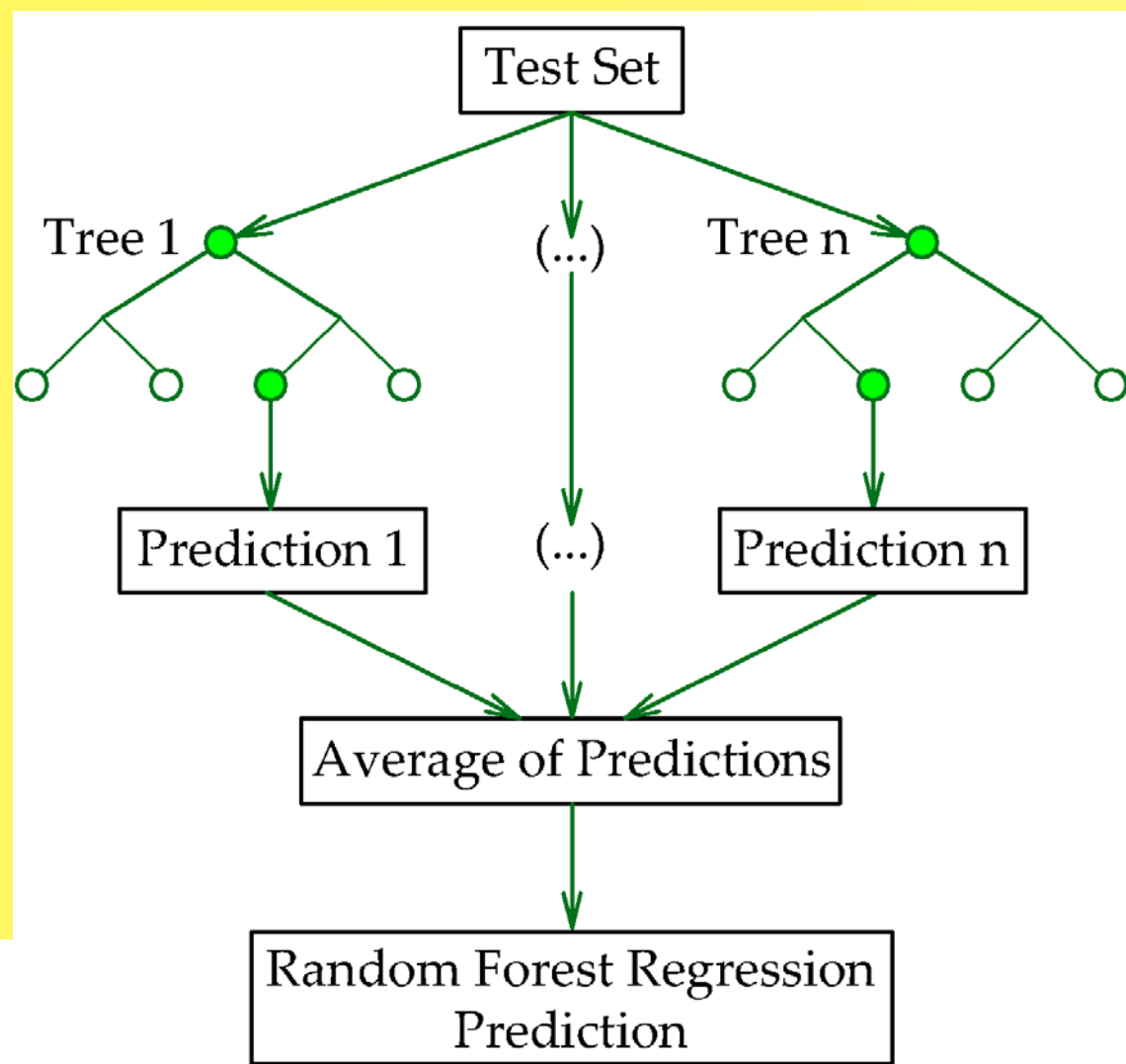
**Random Forest:** sử dụng phương pháp **bagging** mô hình hóa các **cây sâu** mà dữ liệu đầu vào là các mẫu **bootstrap**, và tổng hợp chúng theo **quy trình lấy trung bình** nào đó để nhận được một **mô hình kết hợp** với **phương sai thấp**.



# RANDOM FOREST

## Các Quy Trình Lấy Trung Bình

**Với bài toán hồi quy:** dự đoán của mô hình kết hợp bằng trung bình của các dự đoán của các mô hình cơ bản.



**Với bài toán phân loại:** chọn lớp phổ biến của các mô hình (hard-vote), hoặc chọn lớp có xác suất trung bình lớn nhất tính từ xác suất từng lớp của tất cả các mô hình (soft-vote).

# RANDOM FOREST

## Random Forest và Grid Search Cross Validation

```
# Evaluate methods
scoring = ['explained_variance', 'neg_mean_absolute_error', 'neg_mean_squared_error', 'r2']

"""Function to run random forest with grid search and k-fold cross-validation."""
def get_rf_model(X, y):
    # Hyperparameters search grid
    rf_param_grid = {'bootstrap': [False, True],
                     'n_estimators': [60, 70, 80, 90, 100],
                     'max_features': [0.6, 0.65, 0.7, 0.75, 0.8],
                     'min_samples_leaf': [1],
                     'min_samples_split': [2]
                    }

    # Instantiate random forest regressor
    rf_estimator = RandomForestRegressor(random_state=None)

    # Create the GridSearchCV object
    rf_model = GridSearchCV(estimator=rf_estimator, param_grid=rf_param_grid, cv=10, scoring = scoring,
                             refit = 'neg_mean_squared_error', n_jobs=-1)

    # Fine-tune the hyperparameters
    rf_model.fit(X, y)
    print("Best Parameters:\n", rf_model.best_params_)
    print("Best Score:\n", -rf_model.best_score_)
    return(-rf_model.best_score_)
```

# RANDOM FOREST

## Các siêu tham số (hyper parameters) quan trọng

- **n\_estimators**: số lượng cây, càng lớn thì kết quả càng đáng tin nhưng thời gian chạy càng lâu.
- **max\_features**: số lượng lớn nhất các đặc trưng mà random forest xem xét để chia một nốt.
- **min\_sample\_leaf**: số lượng lá nhỏ nhất của một nốt.
- **n\_jobs**: số processor cho phép sử dụng.

```
rf_grw_mse_2 = get_rf_model(X_grw_2_pls, y_grw)
rf_grw_mse.append(rf_grw_mse_2)
```

Best Parameters:

```
{'max_features': 0.6, 'min_samples_split': 2, 'min_samples_leaf': 1, 'n_estimators': 60, 'bootstrap': False}
```

Best Score:

```
0.06673007252838713
```

# RANDOM FOREST

## Ưu, nhược điểm của Random Forest

- *Có thể áp dụng cho cả bài toán phân loại và bài toán hồi quy.*
- *Có thể điều chỉnh mô hình với nhiều siêu tham số để đạt được độ chính xác cao.*
- *Giải được quyết vấn đề lớn của học máy là overfitting, với càng nhiều cây thì overfitting càng giảm, tuy nhiên tốc độ tính toán sẽ chậm hơn.*



# HỎI ĐÁP



CẢM ƠN THẦY, CÔ GIÁO VÀ CÁC BẠN ĐÃ THEO DÕI!