# CS116 – LẬP TRÌNH PYTHON CHO MÁY HỌC

## Lesson 08

# Phân loại mô hình & các mô hình máy học

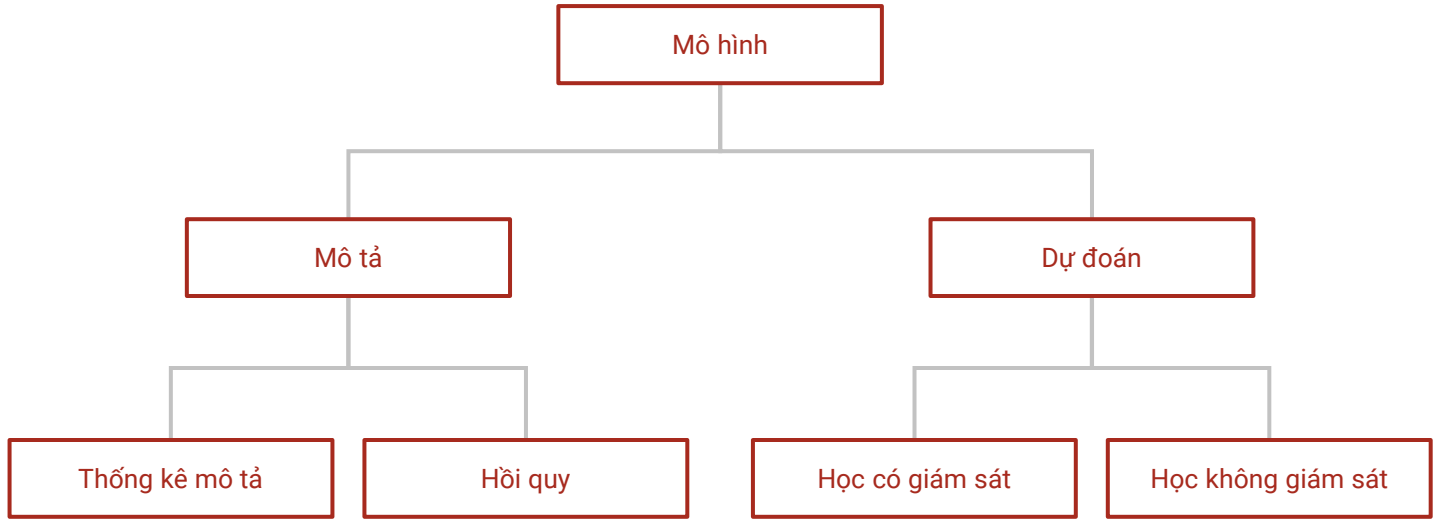
TS. Nguyễn Vinh Tiệp

# NỘI DUNG

1. Phân loại mô hình
2. Các mô hình máy học
    1. Mô hình cơ bản
    2. Bagging và Boosting
    3. Mô hình dựa trên cấu trúc cây
3. Auto ML

# Phân loại mô hình

- Mô hình mô tả và mô hình dự đoán



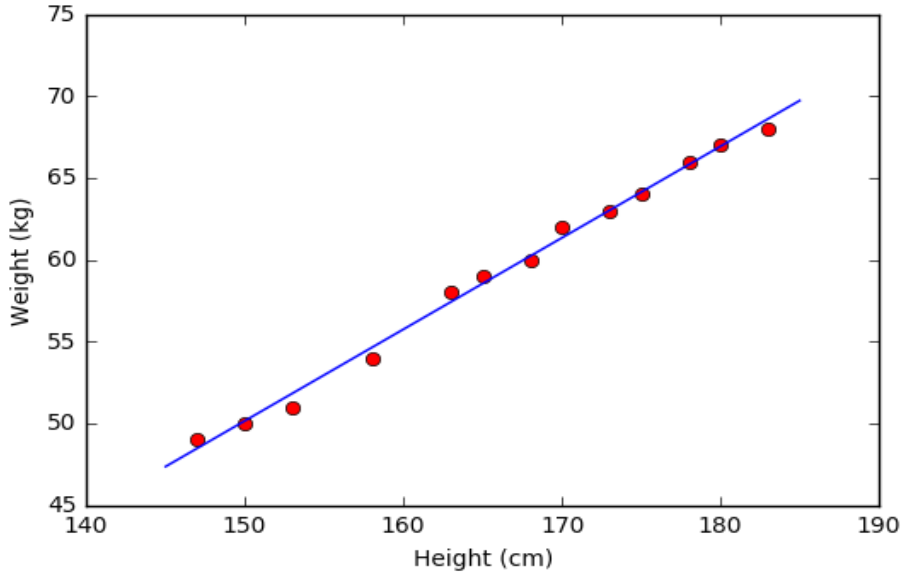
# Mô hình máy học

1. Mô hình cơ bản
    1. Hồi quy tuyến tính (Linear Regression)
    2. Hồi quy luận lý (Logistic Regression)
    3. Cây quyết định (Decision Tree)
2. Bagging và Boosting
3. Mô hình dựa trên cấu trúc cây
    1. Random Forest
    2. XGBoost
    3. LightGBM
    4. CatBoost

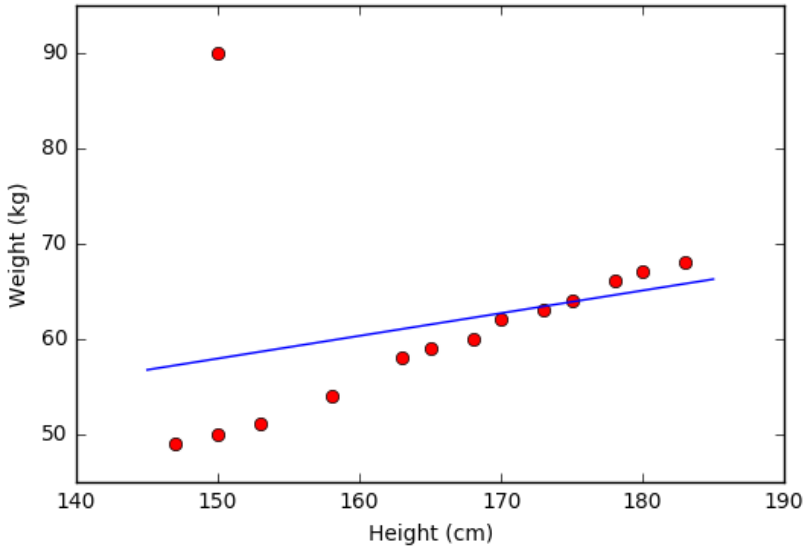

Thực hiện bởi Trường Đại học Công nghệ Thông tin, ĐHQG-HCM

# Hồi quy tuyến tính

- Một cách tiếp cận tuyến tính để mô hình hóa mối quan hệ giữa một biến phản hồi vô hướng (scalar response) và một hoặc nhiều biến giải thích (Explanatory variable) -> Mô hình tham số
- Nhạy cảm với nhiễu/ngoại lệ




https://machinelearningcoban.com/2016/12/28/linearregression/




https://machinelearningcoban.com/2016/12/28/linearregression/


Thực hiện bởi Trường Đại học Công nghệ Thông tin, ĐHQG-HCM

# Hồi quy tuyến tính: Về mặt toán học

- Độ lỗi, ước lượng tham số, tính Bias, Variance

$$Y = \beta * X + \epsilon$$

$$L_{ols}(\hat{\beta}) = \sum_{i=0}^{n} \left\| y_i - x_i * \hat{\beta}_i \right\|^2 = \left\| Y - X * \hat{\beta} \right\|^2$$

$$\hat{\beta_{ols}} = \left(X^T X\right)^{-1} \left(X^T Y\right)$$

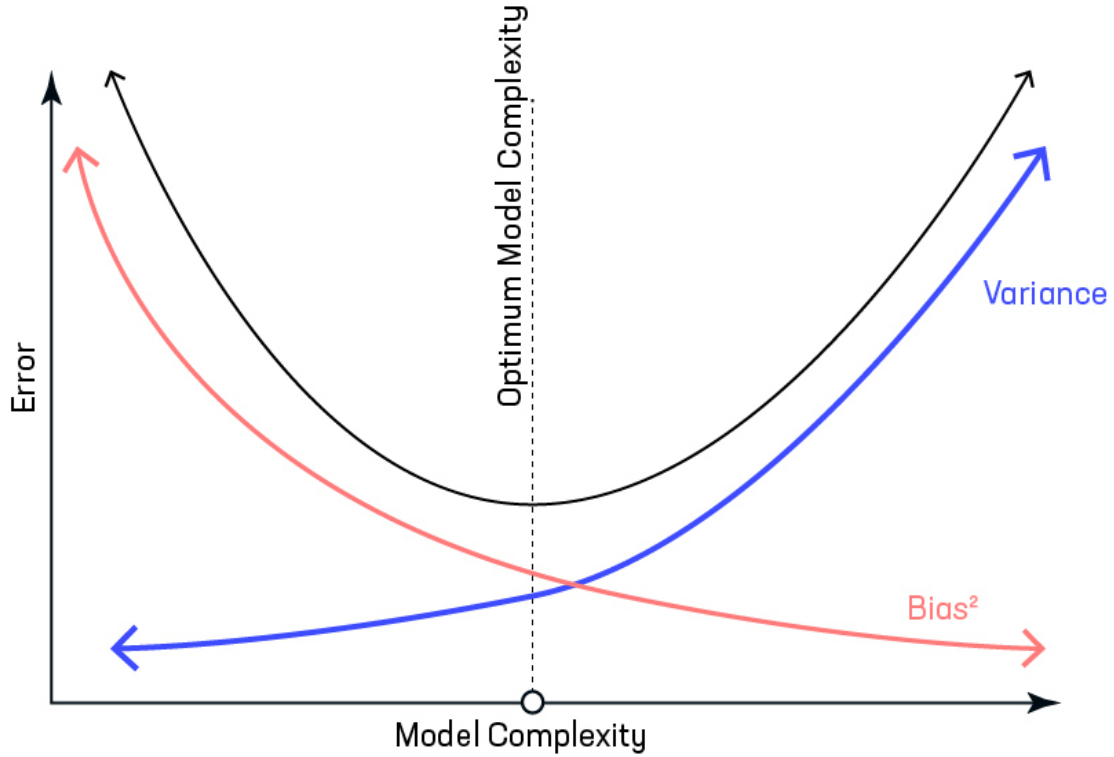
$$Bias(hat\beta) = E\left(\hat{\beta}\right) - \beta$$

$$Variance\left(\hat{\beta}\right) = \sigma^2 \left(X'X\right)^{-1}$$

# Hồi quy tuyến tính: Độ lỗi

$$Error - term = \left(E\left(X\hat{\beta}\right) - X\beta\right)^2 + E\left(X\hat{\beta} - E\left(X\hat{\beta}\right)\right)^2 + \sigma^2$$

**Câu hỏi đặt ra:** Làm thế nào để có được độ phức tạp mô hình tối ưu




https://www.geeksforgeeks.org/lasso-vs-ridge-vs-elastic-net-ml/

# Hồi quy tuyến tính: Ví dụ

- Sử dụng thư viện sklearn & công thức toán (Slide 5)

```
# Step 1: add x0 =1 to dataset
X_train_0 = np.c_[np.ones((X_train.shape[0],1)),X_train]
X_test_0 = np.c_[np.ones((X_test.shape[0],1)),X_test]

# Step2: build model
theta = np.matmul(np.linalg.inv( np.matmul(X_train_0.T,X_train_0)
), np.matmul(X_train_0.T,y_train))
```

```
# Scikit Learn module
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_train,y_train) # Note: x_0 =1 is no need to add, skl
earn will take care of it.

#Parameter
sk_theta = [lin_reg.intercept_]+list(lin_reg.coef_)
parameter_df = parameter_df.join(pd.Series(sk_theta, name='Sklear
n_theta'))
parameter_df
```

| | Parameter | Columns | theta | Sklearn_theta |
|---|---|---|---|---|
| 0 | theta_0 | intersect:x_0=1 | 7.059171 | 7.059171 |
| 1 | theta_1 | age | 0.033134 | 0.033134 |
| 2 | theta_2 | bmi | 0.013517 | 0.013517 |
| 3 | theta_3 | OHE_male | -0.067767 | -0.067767 |
| 4 | theta_4 | OHE_1 | 0.149457 | 0.149457 |
| 5 | theta_5 | OHE_2 | 0.272919 | 0.272919 |
| 6 | theta_6 | OHE_3 | 0.244095 | 0.244095 |

https://www.kaggle.com/code/sudhirnl7/linear-regression-tutorial

# LASSO

- Hồi quy tuyến tính + L1 Regularization → LASSO
- Có thể sử dụng để chọn lọc đặc trưng (Bài 6)

$$L_{lasso} = argmin_{\hat{\beta}} \left( \|Y - \beta * X\|^2 + \lambda * \|\beta\|_1 \right)$$

# Ridge Regression

- Hồi quy tuyến tính + L2 Regularization → Ridge

$$L_{ridge} = argmin_{\hat{\beta}} \left( \|Y - \beta * X\|^2 + \lambda * \|\beta\|_2^2 \right)$$

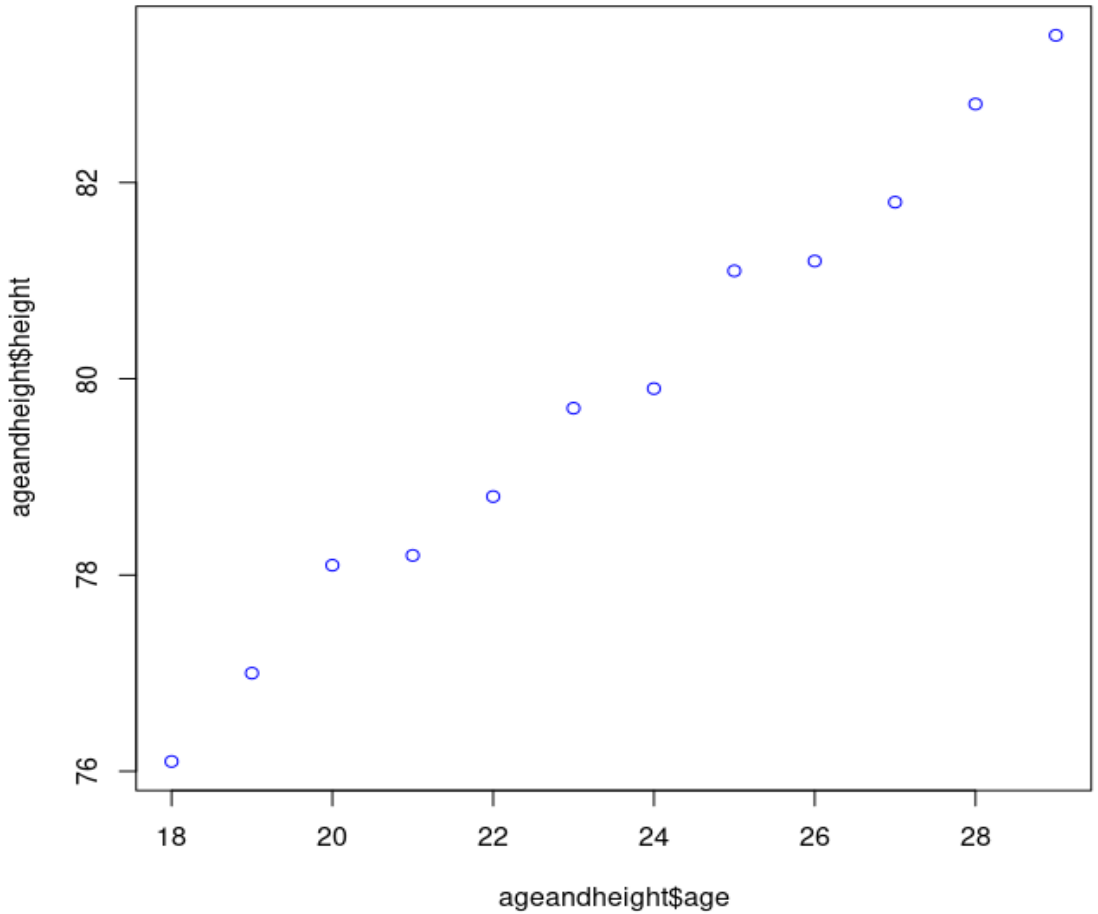
where $\lambda$ is regularization penalty.

# Hồi quy tuyến tính + Regularizations

- Hồi quy tuyến tính + (L1 + L2) Regularization → Elastic Net
- Có thể kết hợp ưu điểm của cả Lasso và Ridge

$$L_{elasticNet} = argmin_{\hat{\beta}}\left(\hat{\beta}\right)\left(\sum\left(y - x_i^J\hat{\beta}\right)^2\right)/2n + \lambda\left((1-\alpha)/2 * \sum_{j=1}^{m}\hat{\beta}_j^2 + \alpha * \sum_{j=1}^{m}\left\|\hat{\beta}_j\right\|\right)$$

# Ví dụ về hồi quy tuyến tính với R



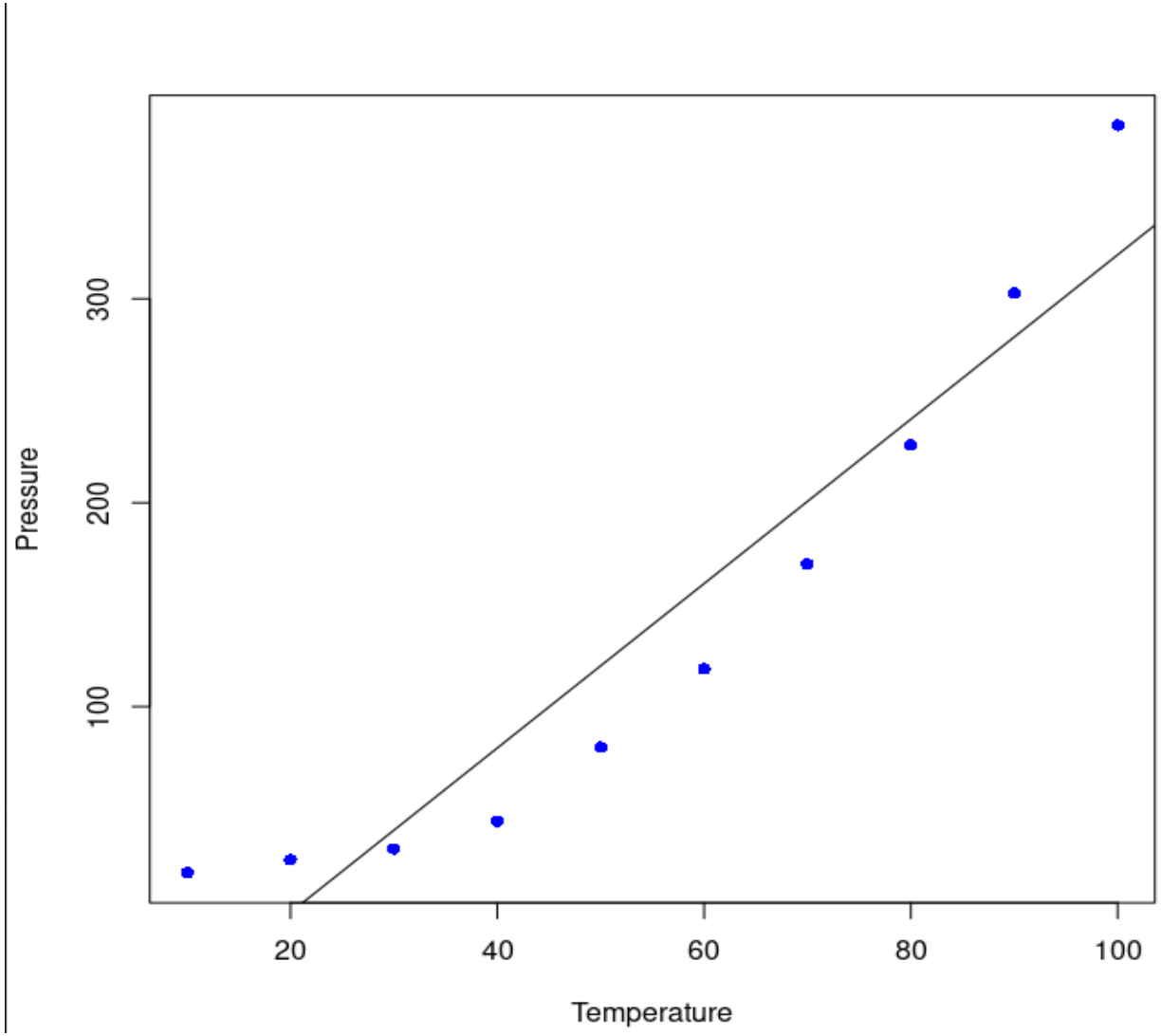
# Ví dụ về hồi quy tuyến tính với R

```
Call:
lm(formula = height ~ age, data = ageandheight)

Residuals:
     Min       1Q   Median       3Q      Max
-0.27238 -0.24248 -0.02762  0.16014  0.47238

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  64.9283     0.5084  127.71  < 2e-16 ***
age           0.6350     0.0214   29.66 4.43e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.256 on 10 degrees of freedom
Multiple R-squared:  0.9888,    Adjusted R-squared:  0.9876
F-statistic:   880 on 1 and 10 DF,  p-value: 4.428e-11
```

# Ví dụ về hồi quy tuyến tính với R

```
Call:
lm(formula = height ~ age + no_siblings, data = ageandheight)

Residuals:
     Min       1Q   Median       3Q      Max
-0.32065 -0.13587 -0.03329  0.17380  0.36860

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 64.65632    0.50961 126.875 5.96e-16 ***
age          0.64007    0.02038  31.407 1.65e-10 ***
no_siblings  0.09123    0.05970   1.528    0.161
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2404 on 9 degrees of freedom
Multiple R-squared:  0.9911,    Adjusted R-squared:  0.9891
F-statistic: 499.9 on 2 and 9 DF,  p-value: 5.982e-10
```

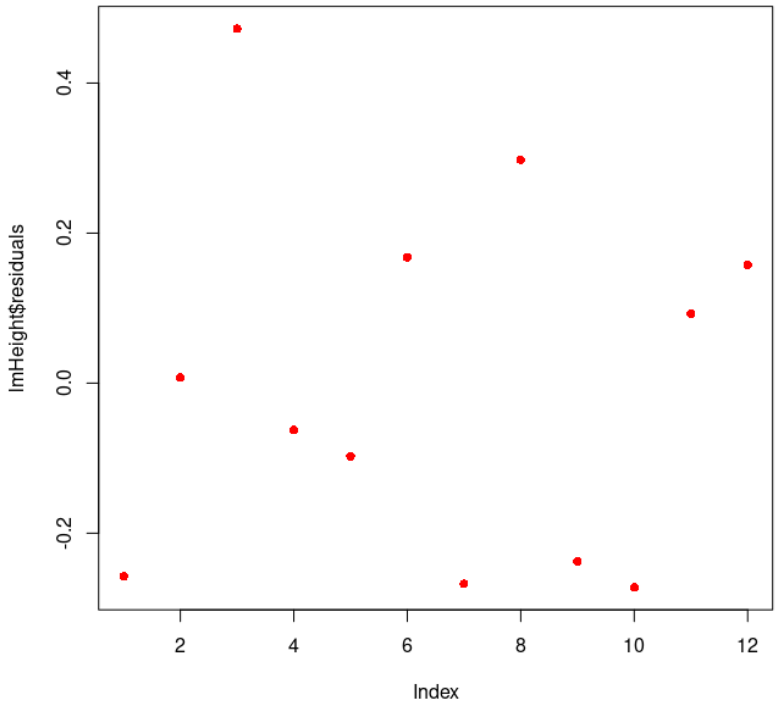
# Ví dụ về hồi quy tuyến tính với R

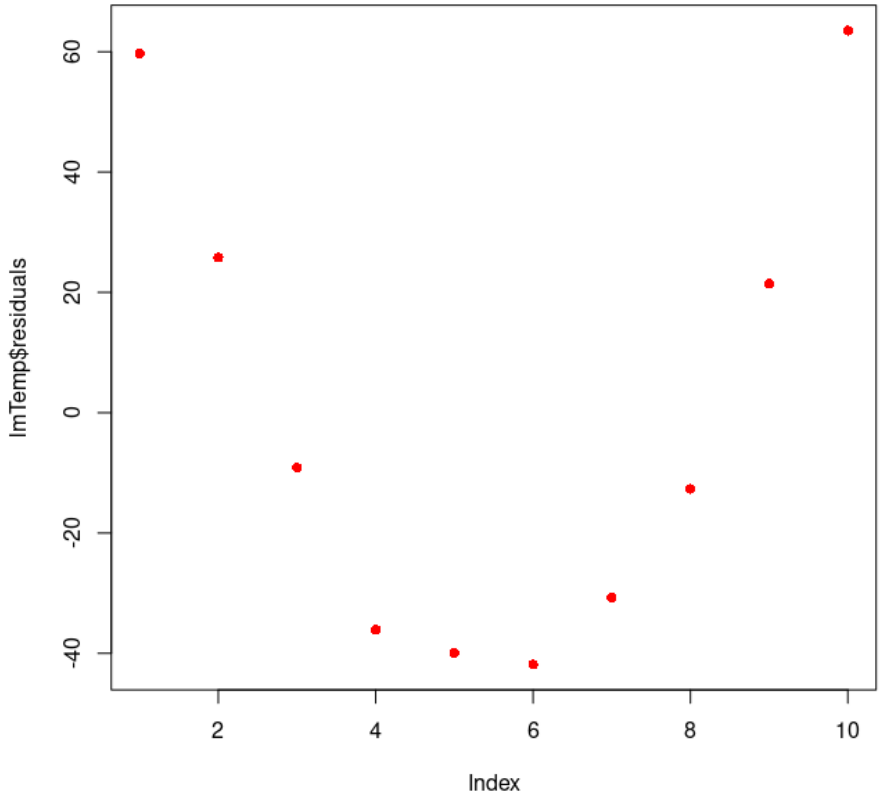


# Ví dụ về hồi quy tuyến tính với R

```
Call:
lm(formula = Pressure ~ Temperature, data = pressure)

Residuals:
   Min     1Q Median     3Q    Max
-41.85 -34.72 -10.90  24.69  63.51

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -81.5000    29.1395  -2.797   0.0233 *
Temperature   4.0309     0.4696   8.583 2.62e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 42.66 on 8 degrees of freedom
Multiple R-squared:  0.902,     Adjusted R-squared:  0.8898
F-statistic: 73.67 on 1 and 8 DF,  p-value: 2.622e-05
```

# Ví dụ về hồi quy tuyến tính với R

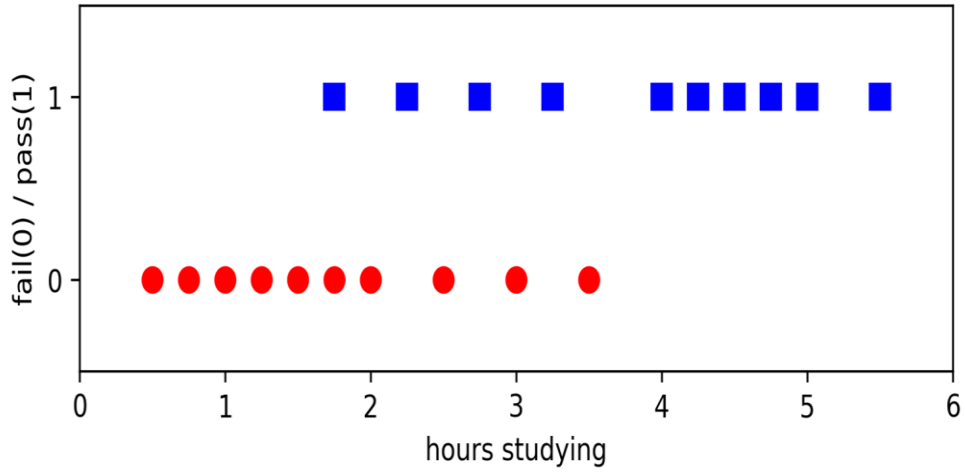


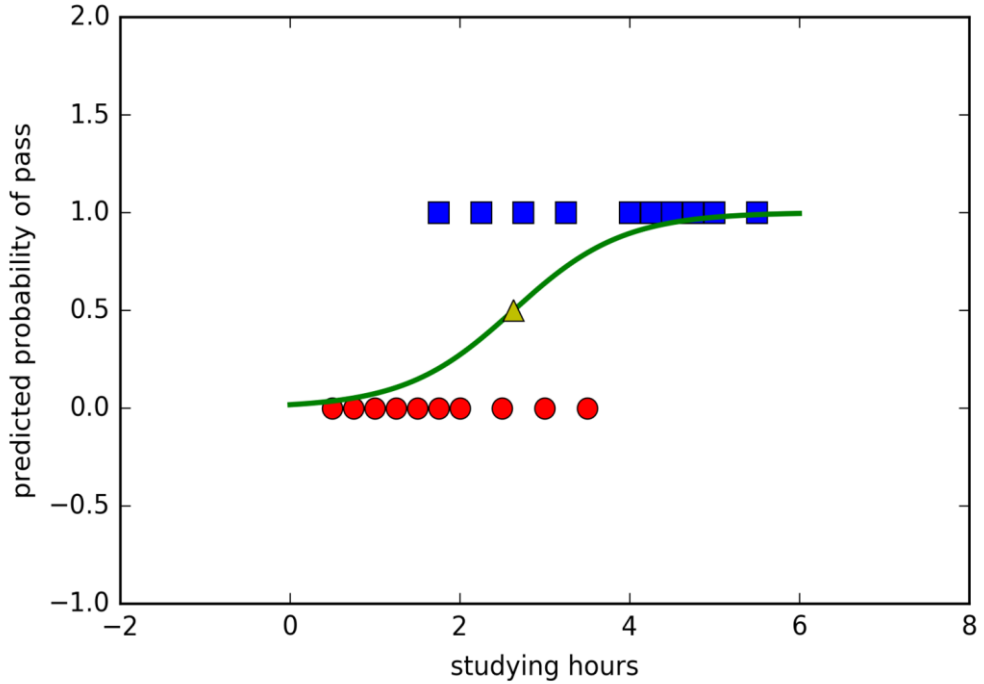



# Hồi quy luận lý

- Estimate the parameters of logistic model (the coefficients in the linear combination)

Why name is Logistic Regression but can use for Binary Classification?



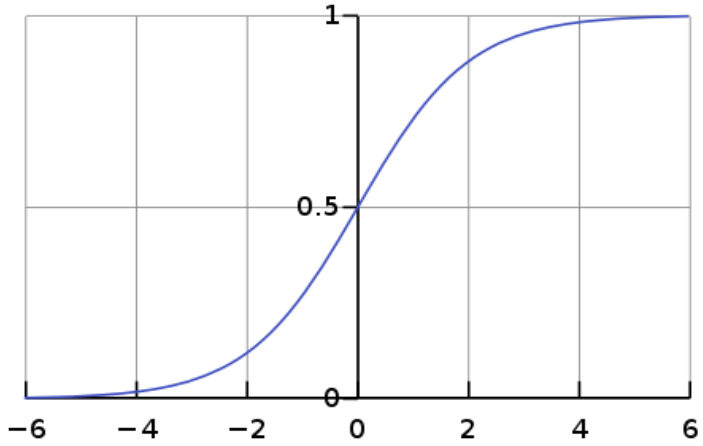




https://machinelearningcoban.com/2017/01/27/logisticregression/

# Hồi quy luận lý: Về mặt toán học

- Hàm Sigmoid

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$




https://en.wikipedia.org/wiki/Logistic_regression

$$t = \beta_0 + \beta_1 x$$

And the general logistic function $p : \mathbb{R} \to (0, 1)$ can now be written as:

$$p(x) = \sigma(t) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

$$\ell = \sum_{k=1}^{K} y_k \log_b(p(\boldsymbol{x_k})) + \sum_{k=1}^{K} (1 - y_k) \log_b(1 - p(\boldsymbol{x_k}))$$

Hàm mất mát

# Hồi quy luận lý: Ví dụ

## Notebook example

```
# train a logistic regression model on the training set
from sklearn.linear_model import LogisticRegression


# instantiate the model
logreg = LogisticRegression(solver='liblinear', random_state=0)


# fit the model
logreg.fit(X_train, y_train)
```

```
# probability of getting output as 1 - rain

logreg.predict_proba(X_test)[:,1]
```

Huấn luyện và dự đoán cho mô hình hồi quy luận lý

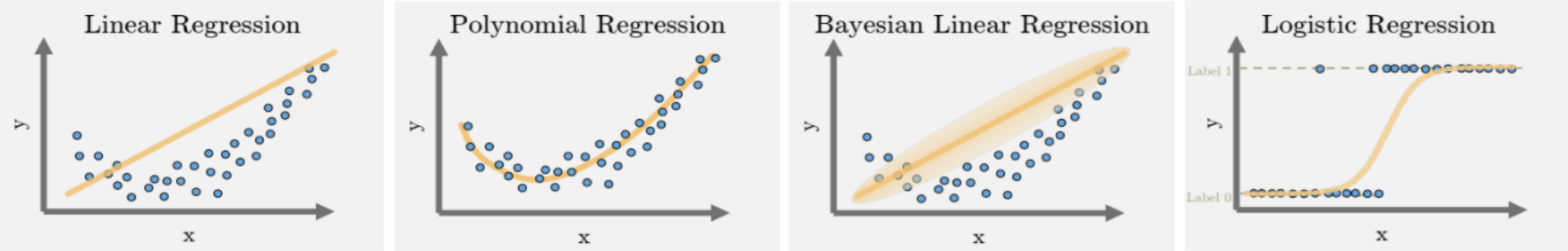
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_i
ntercept=True,
                   intercept_scaling=1, l1_ratio=None, max_ite
r=100,
                   multi_class='warn', n_jobs=None, penalty='l
2',
                   random_state=0, solver='liblinear', tol=0.0
001, verbose=0,
                   warm_start=False)
```

Tham số của mô hình hồi quy luận lý

# Tổng kết

## Cheat Sheet

**Visual Representation:**

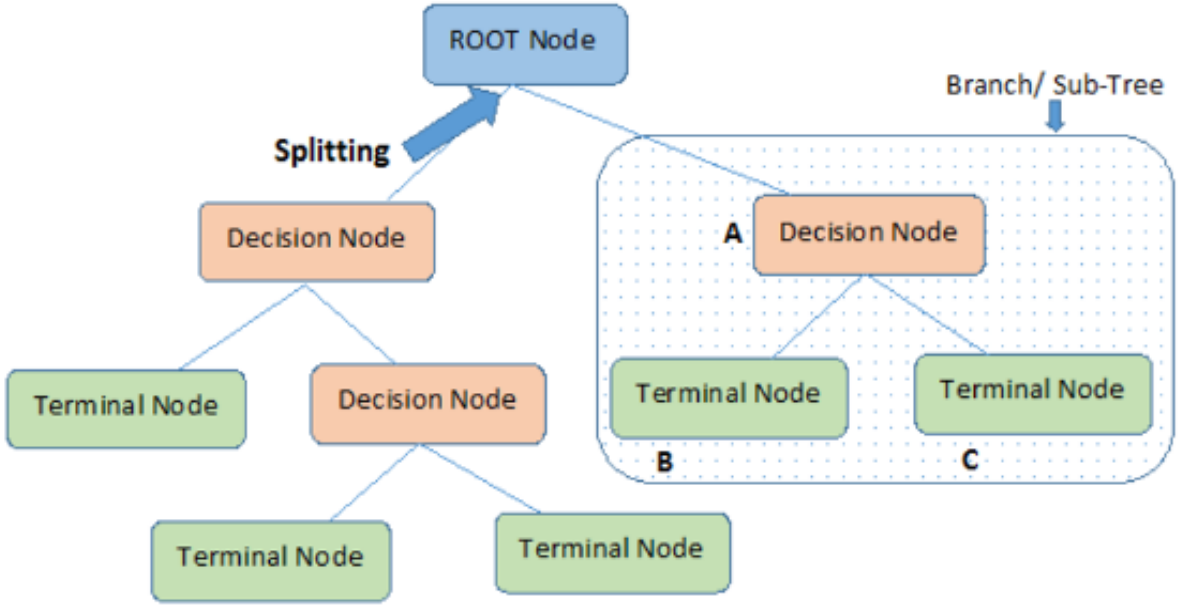



**Summary:**

| | What does it fit? | Estimated function | Error Function |
|---|---|---|---|
| Linear | A line in n dimensions | $f_\beta^{linear}(x_i) = \beta_0 + \beta_1 x_i$ | $\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2$ |
| Polynomial | A polynomial of order k | $f_\beta^{poly}(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots$ | $\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2$ |
| Bayesian Linear | Gaussian distribution for each point | $\mathcal{N}(f_\beta(x_i), \sigma^2)$ | $\sum_i \|y_i - \mathcal{N}(f_\beta(x_i), \sigma^2)\|^2$ |
| Ridge | Linear/polynomial | $f_\beta^{poly}(x_i)$ or $f_\beta^{linear}(x_i)$ | $\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2 + \sum_{j=0}^{n} \beta_j^2$ |
| LASSO | Linear/polynomial | $f_\beta^{poly}(x_i)$ or $f_\beta^{linear}(x_i)$ | $\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2 + \sum_{j=0}^{n} \|\beta_j\|$ |
| Logistic | Linear/polynomial with sigmoid | $\sigma(f_\beta(x_i))$ | $min_\beta \sum_i -y_i log(\sigma(f_\beta(x_i))) - (1-y_i) log(1-\sigma(f_\beta(x_i)))$ |

# Cây quyết định

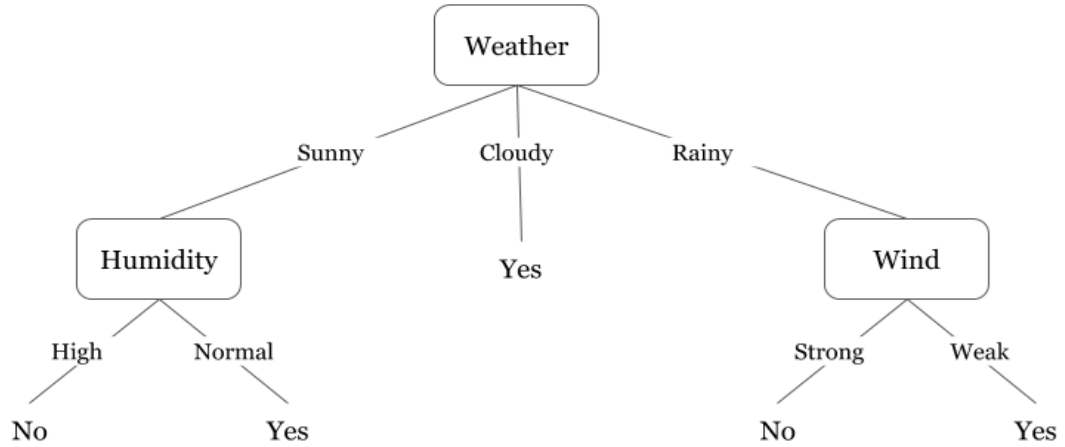
- Là một công cụ giúp bạn đưa ra quyết định bằng cách xem xét tất cả các lựa chọn có sẵn và kết quả tiềm năng của mỗi lựa chọn. Nó cũng tính đến các yếu tố như rủi ro, chi phí và lợi ích.
- Mô hình **không tham số**




https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/

# Cây quyết định: Ví dụ

- Hôm nay trời nắng, độ ẩm cao, gió yếu. Chúng ta có nên chơi cầu lông không?
- Chúng ta có nên quyết định dựa trên thời tiết không?




https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/

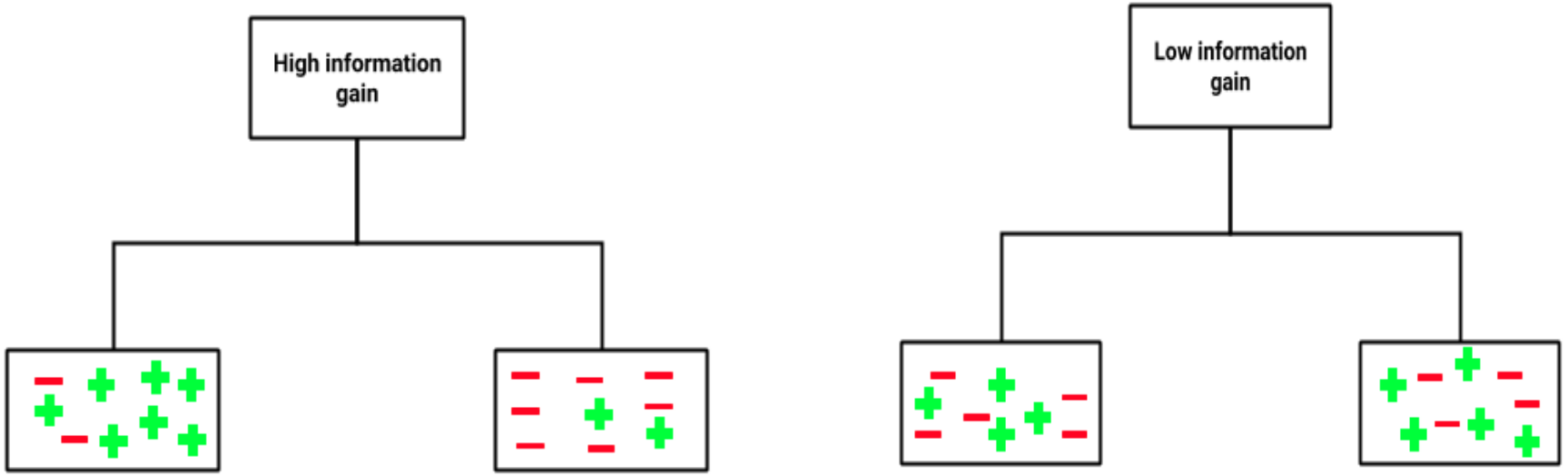
# Xây dựng cây quyết định

Cho một bộ dữ liệu, làm sao để xây dựng cây quyết định?

| Day | Weather | Temperature | Humidity | Wind | Play? |
|---|---|---|---|---|---|
| 1 | Sunny | Hot | High | Weak | No |
| 2 | Cloudy | Hot | High | Weak | Yes |
| 3 | Sunny | Mild | Normal | Strong | Yes |
| 4 | Cloudy | Mild | High | Strong | Yes |
| 5 | Rainy | Mild | High | Strong | No |
| 6 | Rainy | Cool | Normal | Strong | No |
| 7 | Rainy | Mild | High | Weak | Yes |
| 8 | Sunny | Hot | High | Strong | No |
| 9 | Cloudy | Hot | Normal | Weak | Yes |
| 10 | Rainy | Mild | High | Strong | No |

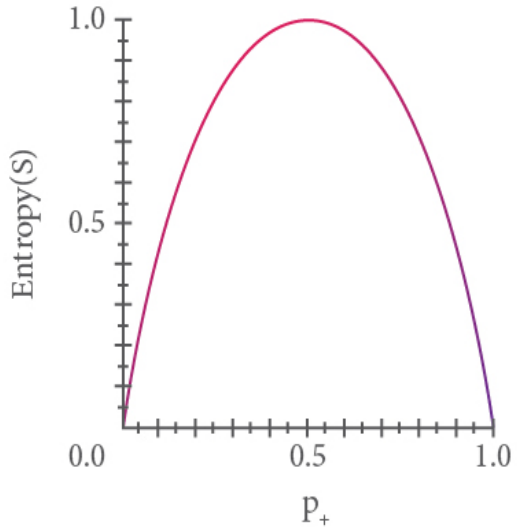
# Entropy & thông tin thu thập

Làm thế nào chúng ta có thể định lượng được thông tin thu thập?

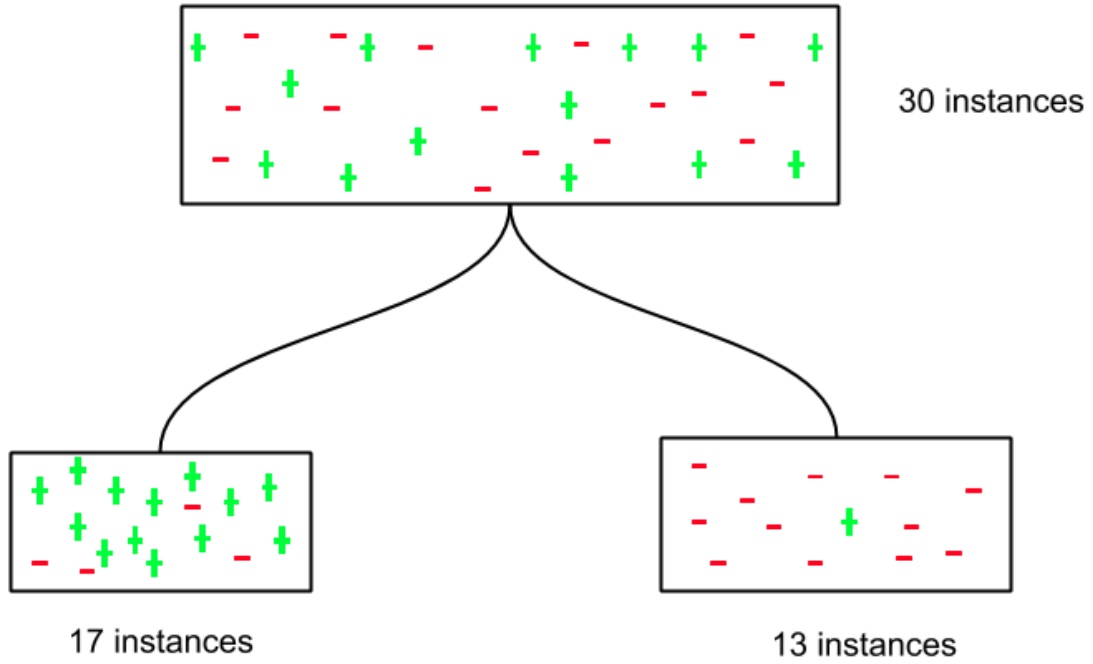



https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/

# Entropy & thông tin thu thập




$$Entropy(S) = -p_+ log_2 p_+ - p_- log_2 p_-$$




$$InformationGain = Entropy(parentnode) - [AverageEntropy(children)]$$

https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/

# Khi nào thì dừng phân tách?

# Tránh Overfitting: Cắt tỉa (Pruning)

- Một phương pháp khác có thể giúp chúng ta tránh overfitting
  - Cắt các nút hoặc nút phụ không đáng kể
  - Loại bỏ các nhánh có tầm quan trọng rất thấp

**Pre-pruning**

Chúng ta có thể ngừng xây dựng cây sớm hơn, điều đó có nghĩa là chúng ta có thể tỉa/loại bỏ/cắt một nút nếu nó có tầm quan trọng thấp trong quá trình xây dựng cây.

**Post-pruning**

Khi cây đã được xây dựng theo chiều sâu, chúng ta có thể bắt đầu cắt tỉa các nút dựa trên tầm quan trọng của chúng.

# Ưu điểm & Nhược điểm

| Advantages | Disadvantages |
| --- | --- |
| **Interpretability**: use visualization | **Overfitting**: easily **overfit** the data if not properly **pruned** or if they are allowed to **grow** too **deep** (model is **too complex**) |
| Handle **both** numerical and categorical features | **Instability**: can be **unstable** because **small** changes in the data can lead to a completely **different tree** → **mitigated** using **ensemble methods**, like Random Forests |

# Ưu điểm & Nhược điểm

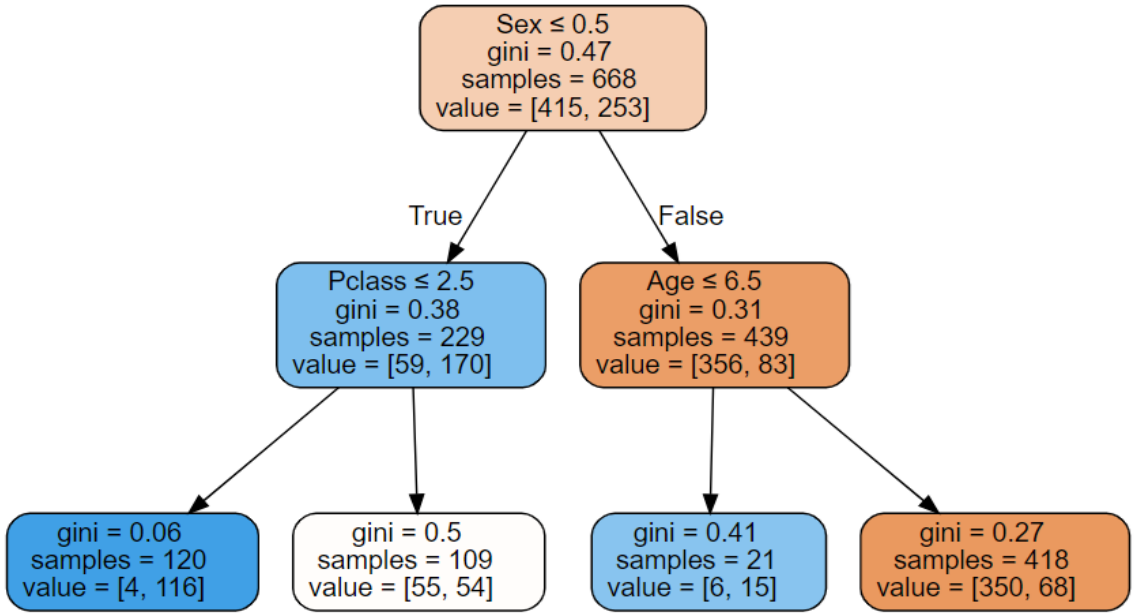
| Ưu điểm | Nhược điểm |
| --- | --- |
| **Phi tham số: không đưa ra** giả định về **sự phân bố** của các biến và **mối quan hệ** giữa các đặc trưng và đầu ra | **Optimization Difficulty**: Finding the **optimal** decision tree for a set of data is computationally **expensive**. Heuristics such as **greedy** algorithms aren't **guaranteed** to find the optimal tree.<br><br>**Khó để tối ưu:** Tìm kiếm cây quyết định **tối ưu** cho tập dữ liệu rất **tốn kém** về mặt tính toán. Các phương pháp phỏng đoán như thuật toán **tham lam** không **đảm bảo** sẽ tìm được cây tối ưu. |
| **Lựa chọn đặc trưng**: thực hiện lựa chọn đặc trưng tiềm ẩn bằng **cách ưu tiên** chọn các đặc trưng **nhiều thông tin** để **phân tách**. | **Bias:** Chúng có xu hướng bị **ảnh hưởng** bởi các đặc trưng có **nhiều cấp độ hơn** (trong trường hợp biến phân loại) hoặc **phạm vi lớn hơn** (trong trường hợp biến số), vì chúng cung cấp **nhiều tùy chọn** hơn để **phân tách** dữ liệu. |

# Cây quyết định: Ví dụ (Titanic)

## Notebook example

```
mean_absolute_error(val_y, m.predict(val_xs))
```

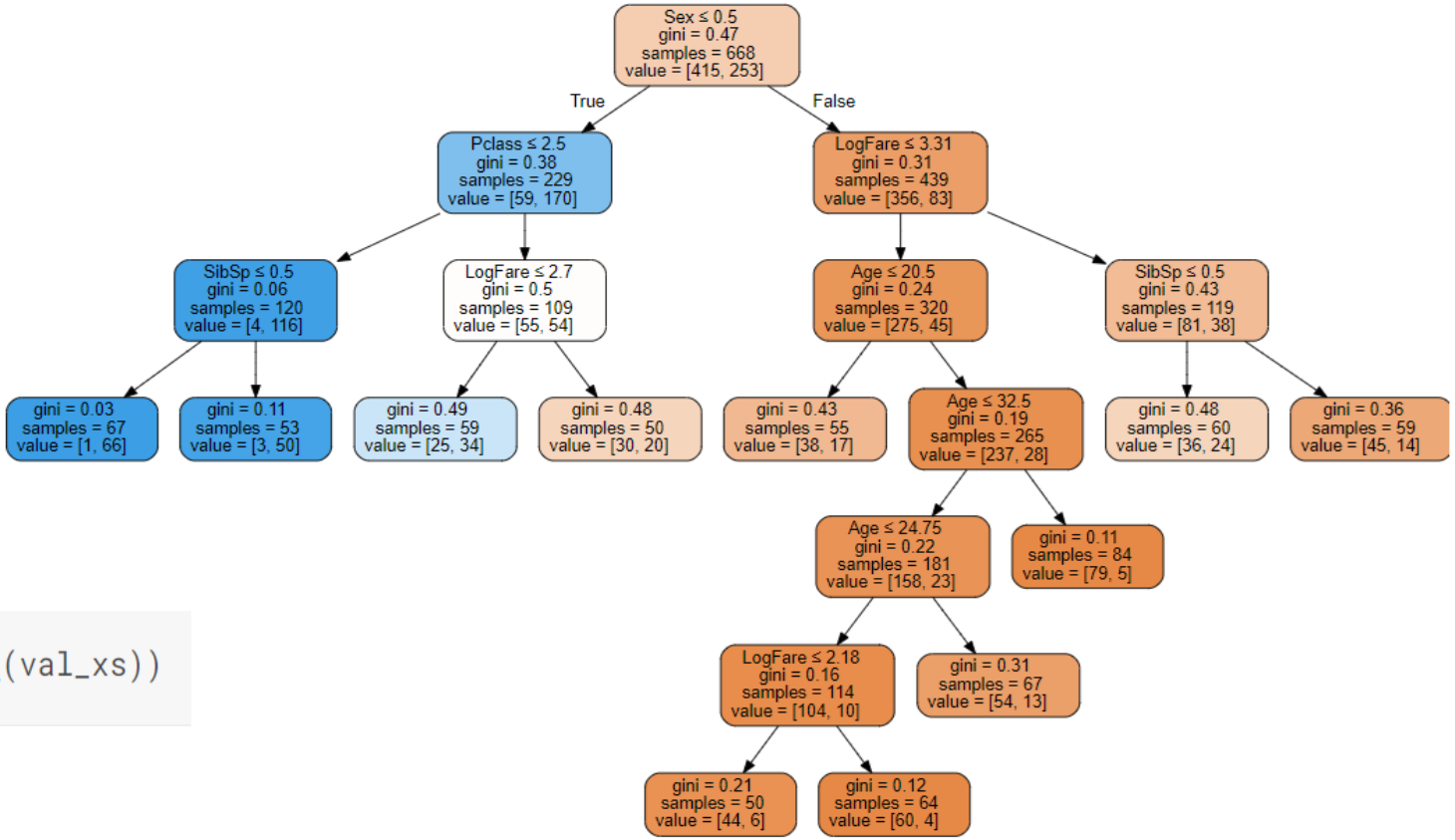
```
0.2242152466367713
```




Cây quyết định đơn giản (max_leaf_node = 4)

# Cây quyết định: Ví dụ (Titanic)

Bigger Tree (min_samples_leaf = 50)
→ get **better result**

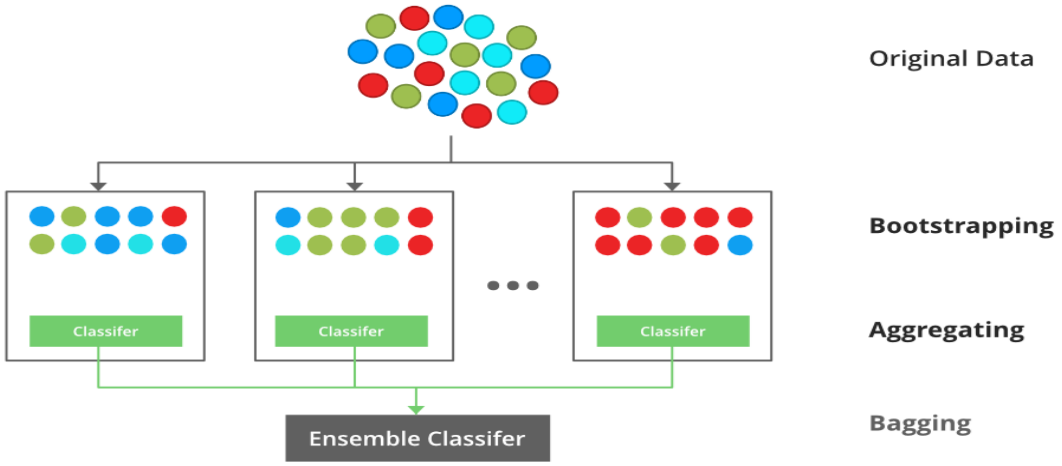



```
mean_absolute_error(val_y, m.predict(val_xs))
```

```
0.18385650224215247
```

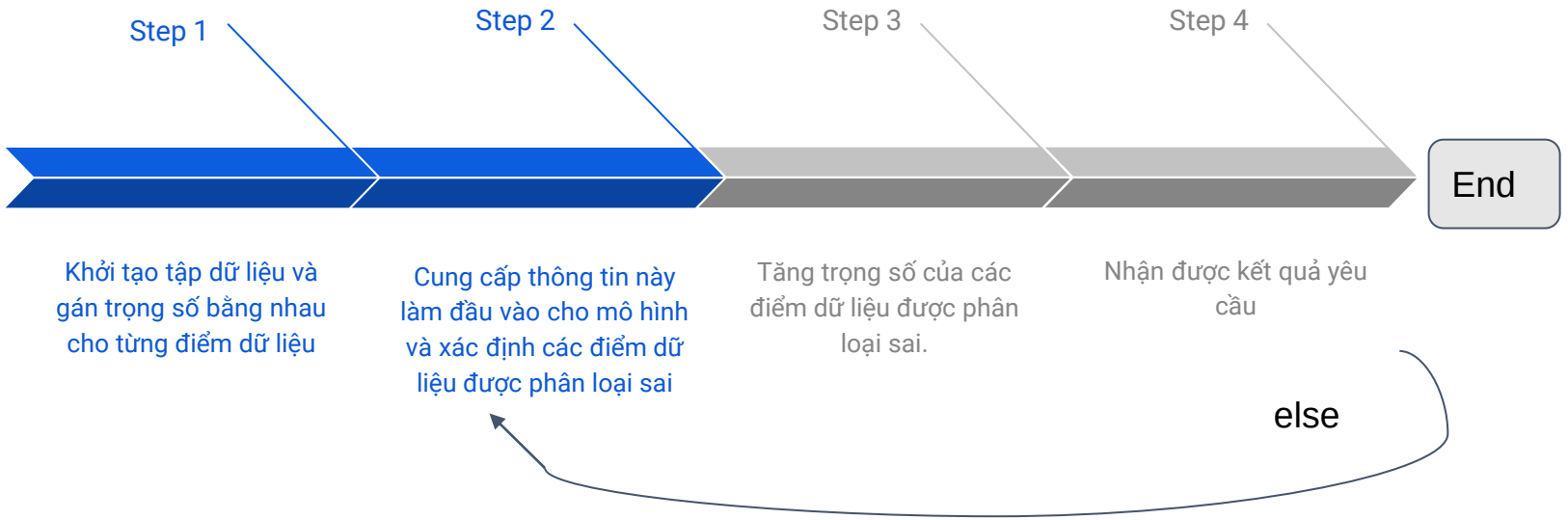
# Bagging

- Các loại Ensemble Learning
- Mô hình **Weak learners'** học độc lập trong quá trình song song và kết hợp chúng để xác định trung bình của mô hình.
- **Random Forest**



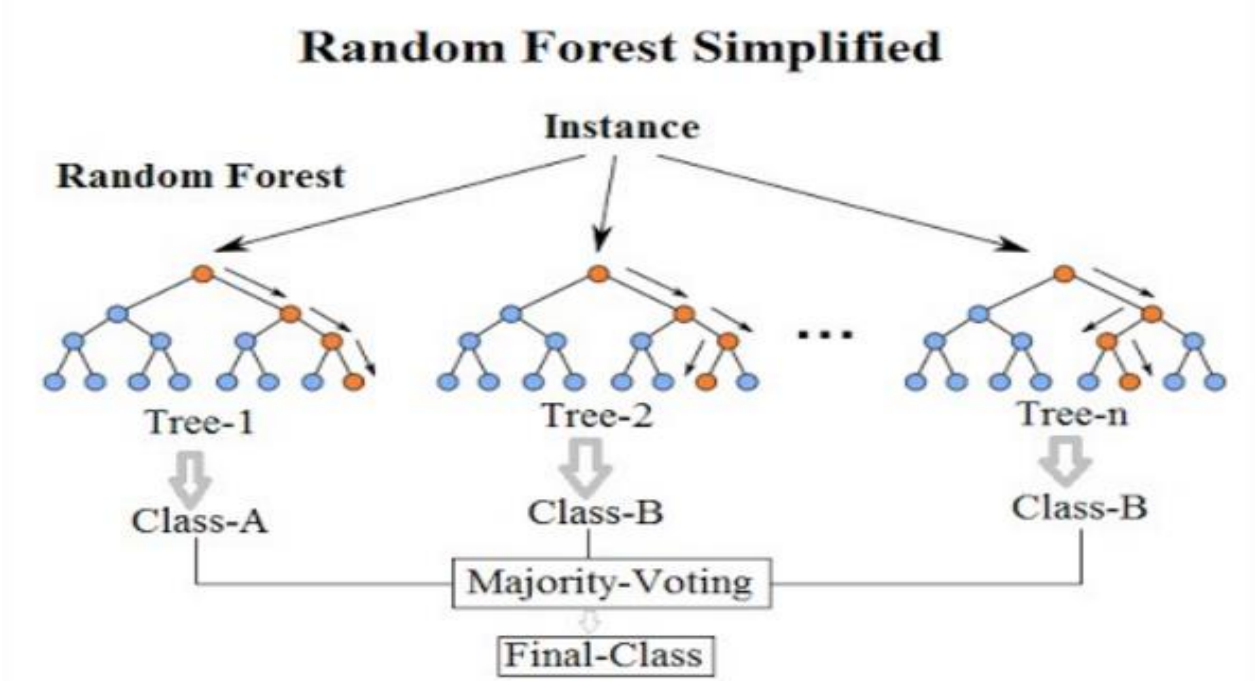

https://www.geeksforgeeks.org/bagging-vs-boosting-in-machine-learning/

# Hiện thực các bước của Bagging



# Random Forest

- Sử dụng bagging (bootstrap aggregating) để huấn luyện từng Cây quyết định trên một tập hợp con dữ liệu ngẫu nhiên khác nhau
- Giảm sự tương quan giữa các cây và làm cho quần thể trở nên mạnh mẽ hơn




https://en.wikipedia.org/wiki/Random_forest

# Những siêu tham số quan trọng

- Siêu tham số có thể tăng độ chính xác dự đoán
- Random Forest sklearn

**n_estimators**
Số cây trong rừng

**max_features**
Số lượng đặc trưng cần xem xét khi tìm kiếm sự phân chia tốt nhất

**min_samples_leaf**
Số lượng mẫu tối thiểu cần có ở một lá node

**criterion**
Hàm đo lường chất lượng của sự phân chia

**max_depth**
Độ sâu tối đa của cây. Nếu None, thì các nút sẽ được mở rộng cho đến khi tất cả các lá đều rỗng hoặc cho đến khi tất cả các lá chứa ít hơn min_samples_split samples

**max_leaf_nodes**
Trồng cây với max_leaf_nodes theo kiểu ưu tiên

# Ưu điểm

**Hiệu suất**

Được coi là mô hình rất chính xác, mạnh mẽ và hiệu quả trên tập dữ liệu lớn

**Xử lý dữ liệu có số chiều lớn**

Có thể xử lý các tập dữ liệu có số lượng đặc trưng cao và không cần phân tích đặc trưng

**Tính linh hoạt**

Có thể được sử dụng cho cả nhiệm vụ hồi quy và phân loại, đồng thời chúng tương đối dễ điều chỉnh vì không yêu cầu điều chỉnh siêu tham số nhiều

# Ưu điểm

**Mạnh mẽ với dữ liệu chứ ngoại lệ và phi tuyến**

Ít có xu hướng trang bị overfitting và mạnh mẽ khi xử lý ngoại lệ, cũng như có khả năng mô hình hóa mối quan hệ phi tuyến tính, phức tạp

**Song song hóa**

Việc huấn luyện các cây khác nhau có thể được thực hiện song song vì mỗi cây được xây dựng độc lập

**Lựa chọn đặc trưng quan trọng**

Đưa ra ước tính tốt về những đặc trưng quan trọng trong dữ liệu → lựa chọn tính năng

# Nhược điểm

**Khả năng giải thích**

Khó diễn giải so với cây quyết định

**Độ phức tạp tính toán**

Có thể tạo số lượng cây quyết định khá lớn và tốn nhiều bộ nhớ

**Bias với dữ liệu không cân bằng**

Bias có thể cao khi xử lý các tập dữ liệu mất cân bằng

# Ví dụ

Notebook example

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(criterion='gini',
                            n_estimators=700,
                            min_samples_split=10,
                            min_samples_leaf=1,
                            max_features='auto',
                            oob_score=True,
                            random_state=1,
                            n_jobs=-1)
rf.fit(train.iloc[:, 1:], train.iloc[:, 0])
print("%.4f" % rf.oob_score_)
```

```
0.8294
```

# Reference

- https://web.stanford.edu/class/stats202/intro.html
- https://towardsdatascience.com/what-makes-lightgbm-lightning-fast-a27cf0d9785e
- http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html
- Cheat Sheet: https://drive.google.com/drive/folders/1IGiethDMHashf9Gsfx3sPTsJOnl15Zvx

# QUIZ & QUESTIONS