

# Báo Cáo Kỹ Thuật

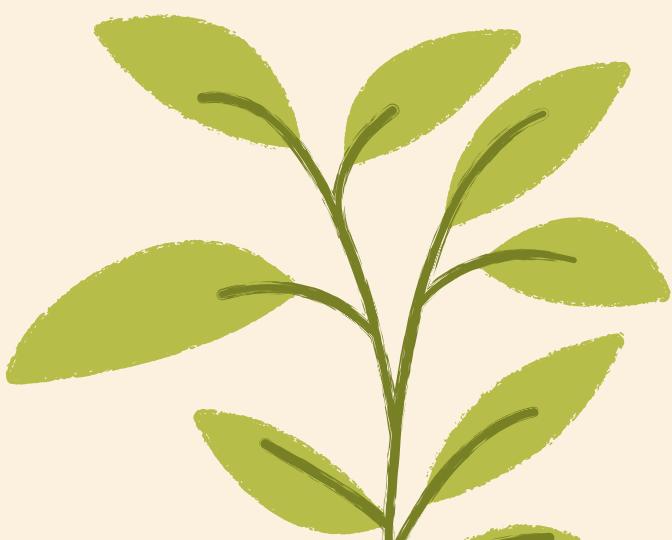
Nhóm 17 - 64KTPM3

DEF  
KLM M  
RST  
XYZ

## Hàm Kích Hoạt ReLU trong Mô Hình Neural Network

# NỘI DUNG

- Giới thiệu
- Định nghĩa ReLU
- Ưu điểm của ReLU
- Nhược điểm của ReLU
- Các biến thể của ReLU
- Ứng dụng của ReLU
- Tổng kết

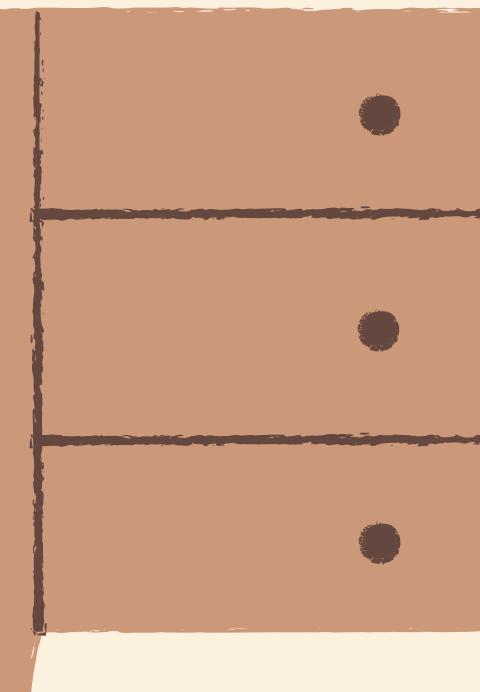
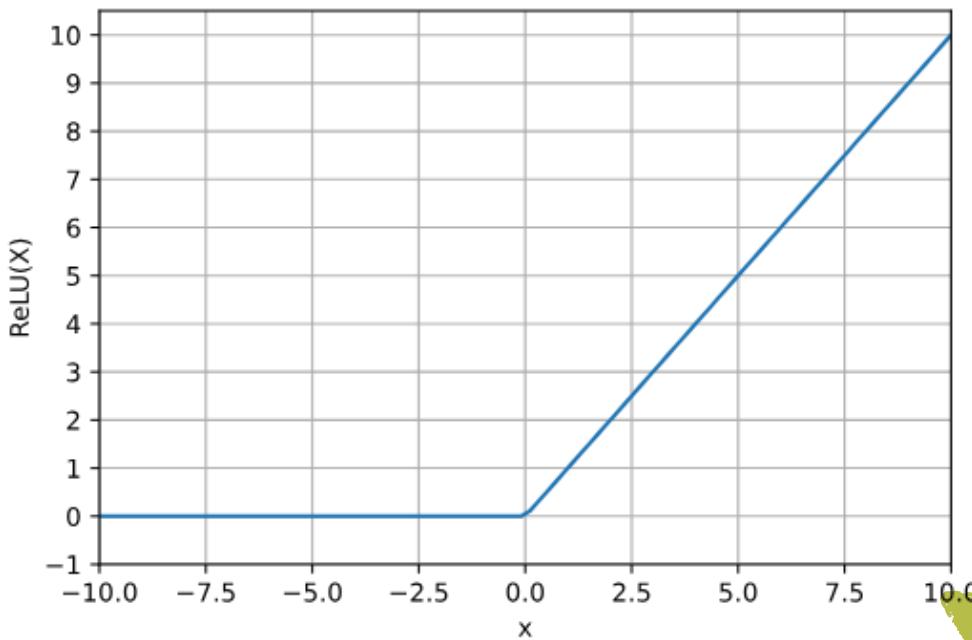


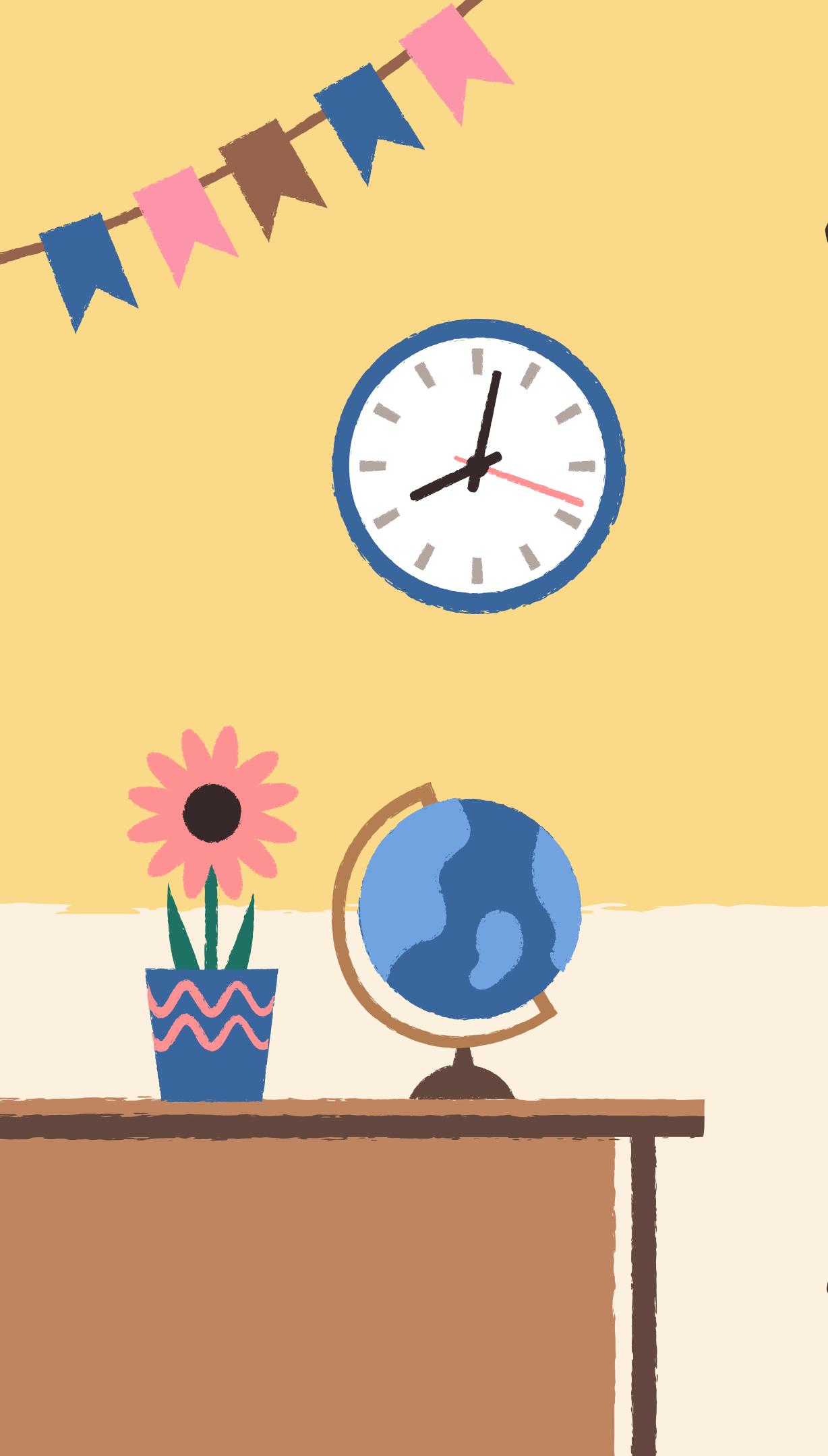
# Giới thiệu

- **Mạng nơ-ron:** Là một mô hình tính toán lấy cảm hứng từ hệ thống thần kinh của con người, được sử dụng để nhận dạng các mẫu phức tạp trong dữ liệu.
- **Hàm kích hoạt:** Giới thiệu phi tuyến tính vào mạng nơ-ron, cho phép chúng học các hàm phức tạp.
- **ReLU:** Một hàm kích hoạt phổ biến, đơn giản và hiệu quả, được sử dụng rộng rãi trong các mạng nơ-ron hiện đại.

# Định nghĩa

- Công thức:  $\text{ReLU}(x) = \max(0, x)$
- Giải thích: Đầu ra của ReLU bằng 0 nếu đầu vào âm, và bằng chính đầu vào nếu đầu vào dương.
- Đặc điểm: Đơn giản, dễ tính toán.
- Không bão hòa cho các giá trị dương.





## Ưu điểm của ReLU



- Hội tụ nhanh: ReLU giúp mô hình học nhanh hơn đáng kể so với các hàm sigmoid và tanh.
- Khắc phục vanishing gradient: ReLU ít gặp vấn đề vanishing gradient, giúp huấn luyện các mạng nơ-ron sâu hiệu quả hơn.
- Tính toán hiệu quả: ReLU đơn giản, dễ tính toán, giúp giảm thiểu tài nguyên tính toán.
- Sparsity: ReLU tạo ra sự thưa thớt trong mạng nơ-ron, giúp mô hình tổng quát hóa tốt hơn và tránh overfitting.

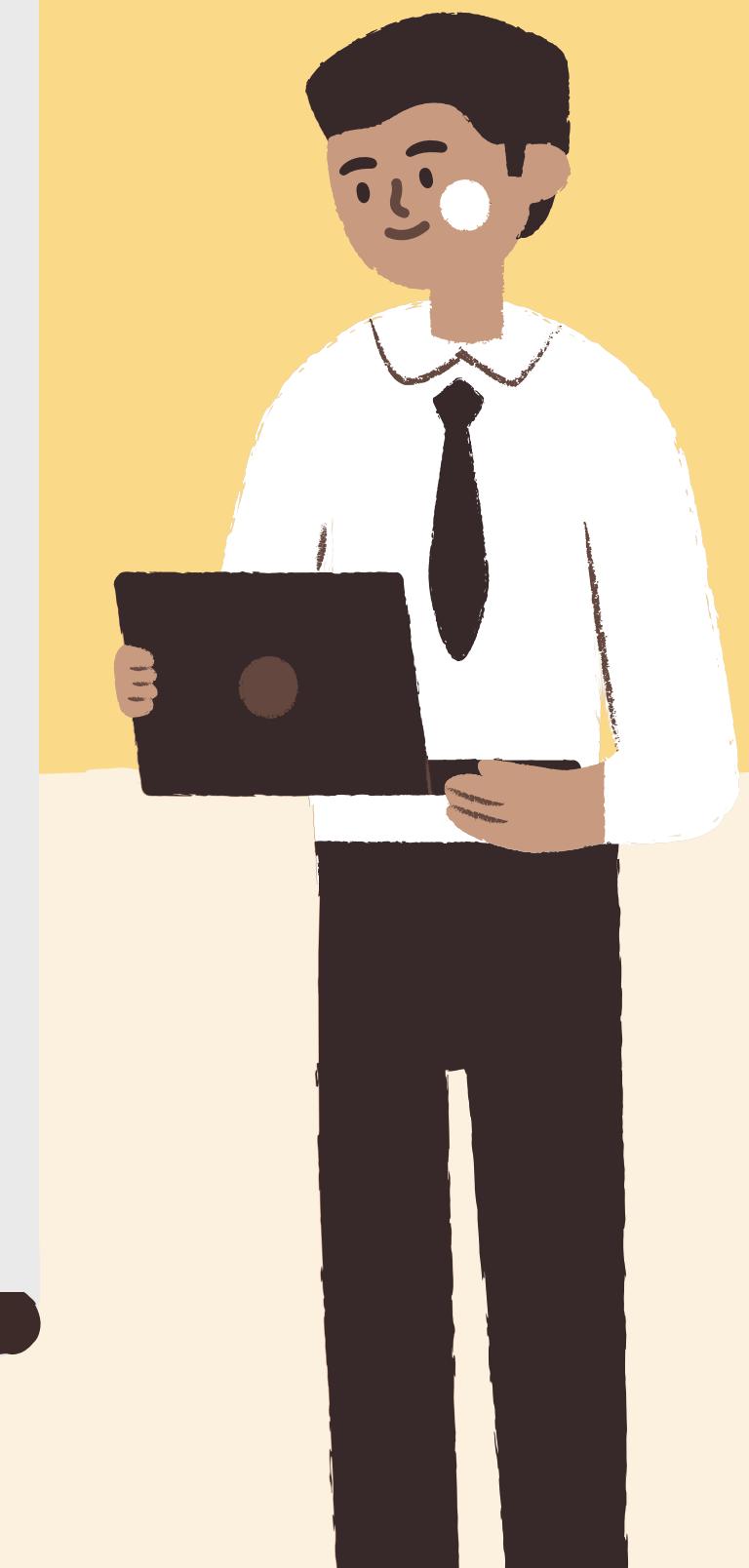
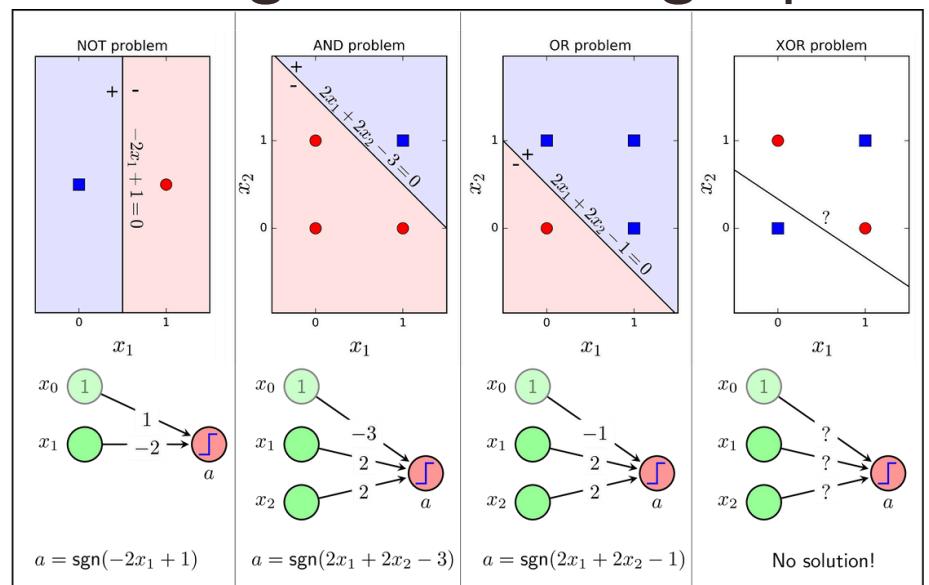
## Nhược điểm của ReLU

- Dying ReLU: Các neuron có thể "chết" khi nhận đầu vào âm, dẫn đến không học được gì thêm. Giải thích: Khi một neuron luôn nhận đầu vào âm, đạo hàm của ReLU tại điểm đó bằng 0, khiến neuron không cập nhật trọng số và trở nên "chết".
- Không khả vi tại 0: Mặc dù không ảnh hưởng nhiều trong thực tế, ReLU không khả vi tại điểm 0.



# Các biến thể của ReLU

- Leaky ReLU:  $\text{ReLU}(x) = \max(0.01x, x)$  - Cho phép một độ dốc nhỏ cho các giá trị âm, giúp tránh Dying ReLU.
- Parametric ReLU (PReLU):  $\text{ReLU}(x) = \max(ax, x)$  - Tương tự Leaky ReLU, nhưng độ dốc  $a$  là một tham số có thể học được.
- Exponential Linear Unit (ELU): Hàm mượt hơn ReLU, có thể mang lại hiệu suất tốt hơn trong một số trường hợp.



## Ứng dụng của ReLU

- Thị giác máy tính: Nhận dạng đối tượng, phân loại ảnh, phát hiện đối tượng,...
- Xử lý ngôn ngữ tự nhiên: Phân tích cảm xúc, dịch máy, tạo văn bản,...
- Nhận dạng giọng nói: Chuyển đổi giọng nói thành văn bản.
- Các mô hình sử dụng ReLU: CNN, RNN, Transformer,...
- ReLU là một hàm kích hoạt quan trọng trong học sâu, đóng góp vào sự phát triển của nhiều ứng dụng AI hiện đại.



# PHÂN TÍCH THAM SỐ

- test\_size=0.3
- hidden\_layer\_sizes=(100, 50)
- alpha=0.1

# test\_size=0.3

## Tại sao lại chọn

Đây là một tỉ lệ phân chia phổ biến trong học máy vì nó thường mang lại sự cân bằng tốt giữa hai yếu tố:

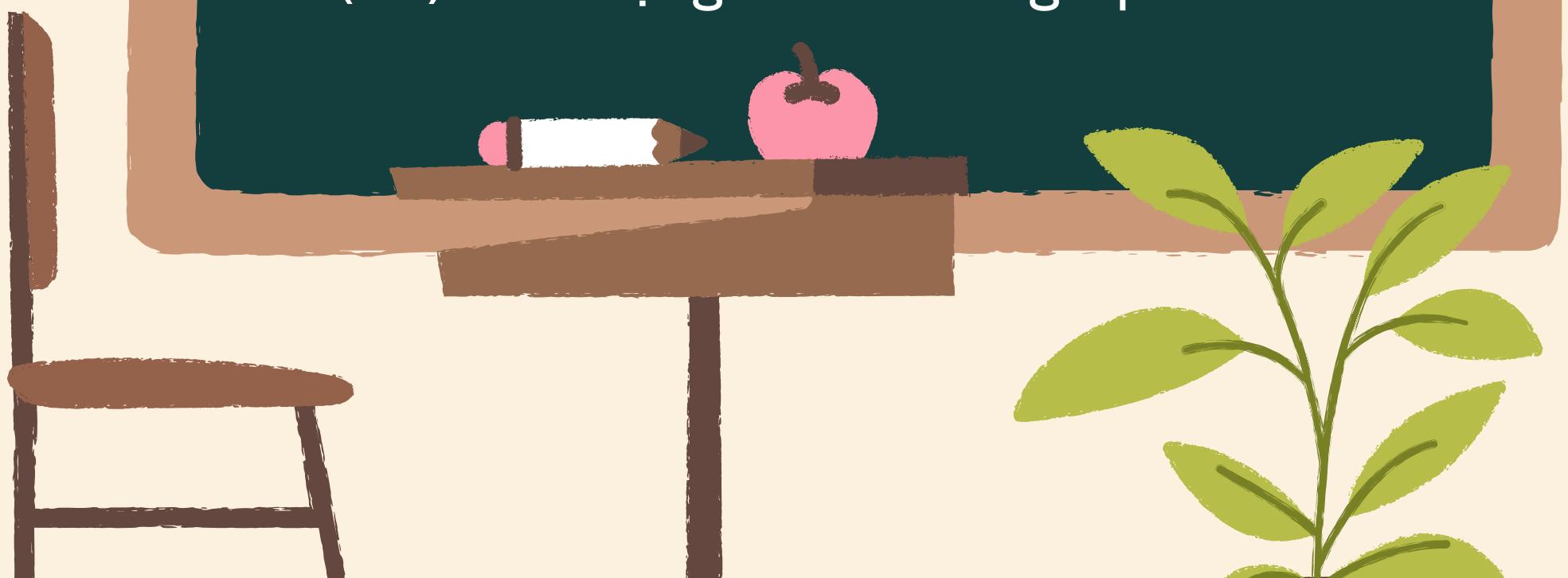
- Đủ dữ liệu huấn luyện: 70% dữ liệu là đủ lớn để mô hình học được các đặc trưng quan trọng từ dữ liệu.
- Đủ dữ liệu kiểm tra: 30% dữ liệu đủ để đánh giá hiệu suất của mô hình một cách đáng tin cậy.

# Các yếu tố ảnh hưởng đến việc lựa chọn test\_size

- Kích thước tập dữ liệu: Với tập dữ liệu nhỏ, có thể cần tăng test\_size (ví dụ 40% hoặc 50%) để đảm bảo tập kiểm tra đủ lớn. Ngược lại, với tập dữ liệu rất lớn, có thể giảm test\_size xuống 10% hoặc thậm chí nhỏ hơn.
- Độ phức tạp của mô hình: Mô hình phức tạp hơn thường cần nhiều dữ liệu huấn luyện hơn, do đó có thể cần giảm test\_size.
- Mục tiêu của mô hình: Nếu mục tiêu là đạt độ chính xác cao nhất có thể, có thể cần tăng test\_size để đánh giá mô hình kỹ lưỡng hơn.

# hidden\_layer\_sizes=(100, 50)

- Hai lớp ẩn: Số lượng số trong tuple tương ứng với số lượng lớp ẩn.
- Lớp ẩn thứ nhất có 100 neuron: Giá trị đầu tiên (100) là số lượng neuron trong lớp ẩn đầu tiên.
- Lớp ẩn thứ hai có 50 neuron: Giá trị thứ hai (50) là số lượng neuron trong lớp ẩn thứ hai.



Ý nghĩa đối với  
mô hình



## Tại sao lại chọn cấu trúc này?

Việc lựa chọn số lượng lớp ẩn và số lượng neuron trong mỗi lớp là một phần của quá trình điều chỉnh siêu tham số (hyperparameter tuning).

Không có một công thức chung nào để xác định cấu trúc tối ưu, nó phụ thuộc vào nhiều yếu tố



## CÁC YẾU TỐ PHỤ THUỘC

- Độ phức tạp của vấn đề: Vấn đề càng phức tạp thì càng cần nhiều lớp ẩn và/hoặc nhiều neuron hơn.
- Kích thước tập dữ liệu: Tập dữ liệu lớn hơn có thể hỗ trợ mạng nơ-ron lớn hơn.
- Thử nghiệm và sai: Thường phải thử nghiệm nhiều cấu trúc khác nhau để tìm ra cấu trúc tốt nhất cho từng bài toán cụ thể.

hidden\_layer\_sizes=(100, 50)

# ALPHA = 0.1

## Tại sao lại chọn alpha=0.1 ?

- Cân bằng: alpha=0.1 thường tạo ra sự cân bằng tốt giữa việc đơn giản hóa mô hình (ngăn chặn overfitting) và duy trì độ chính xác dự đoán. Nó cho phép mô hình giữ lại các đặc trưng quan trọng trong khi loại bỏ bớt các đặc trưng không cần thiết hoặc nhiễu.
- Thực nghiệm: Trong thực tế, alpha=0.1 thường cho kết quả khá tốt trên nhiều tập dữ liệu khác nhau.
- Điểm bắt đầu: Nó là một điểm khởi đầu hợp lý để từ đó điều chỉnh alpha lên hoặc xuống dựa trên kết quả đánh giá mô hình.

# Ảnh hưởng của giá trị alpha

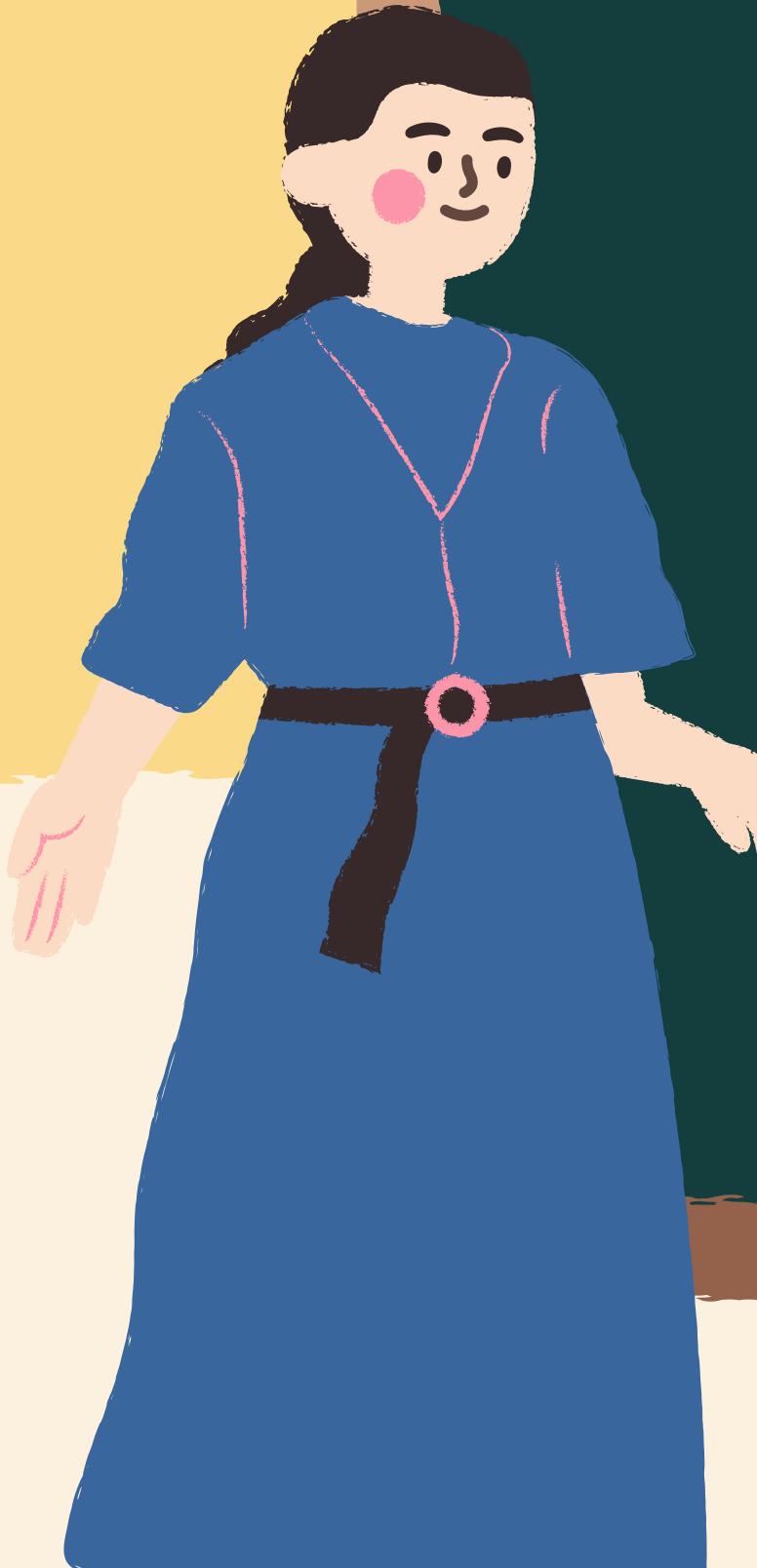
Tham số alpha  
trong Lasso  
Regression kiểm  
soát mức độ phạt  
áp dụng lên các hệ  
số.

- alpha = 0: Không có điều chuẩn, mô hình trở thành hồi quy tuyến tính thông thường. Mô hình có thể phức tạp và dễ bị overfitting.
- alpha nhỏ (ví dụ: 0.01): Mức độ phạt yếu, một số hệ số sẽ bị thu nhỏ lại nhưng vẫn khác 0. Mô hình vẫn giữ được độ phức tạp nhất định.
- alpha lớn (ví dụ: 1, 10): Mức độ phạt mạnh, nhiều hệ số bị ép về 0. Mô hình trở nên rất đơn giản, có thể dẫn đến underfitting (thiếu khớp).



# PHÂN TÍCH DỮ LIỆU

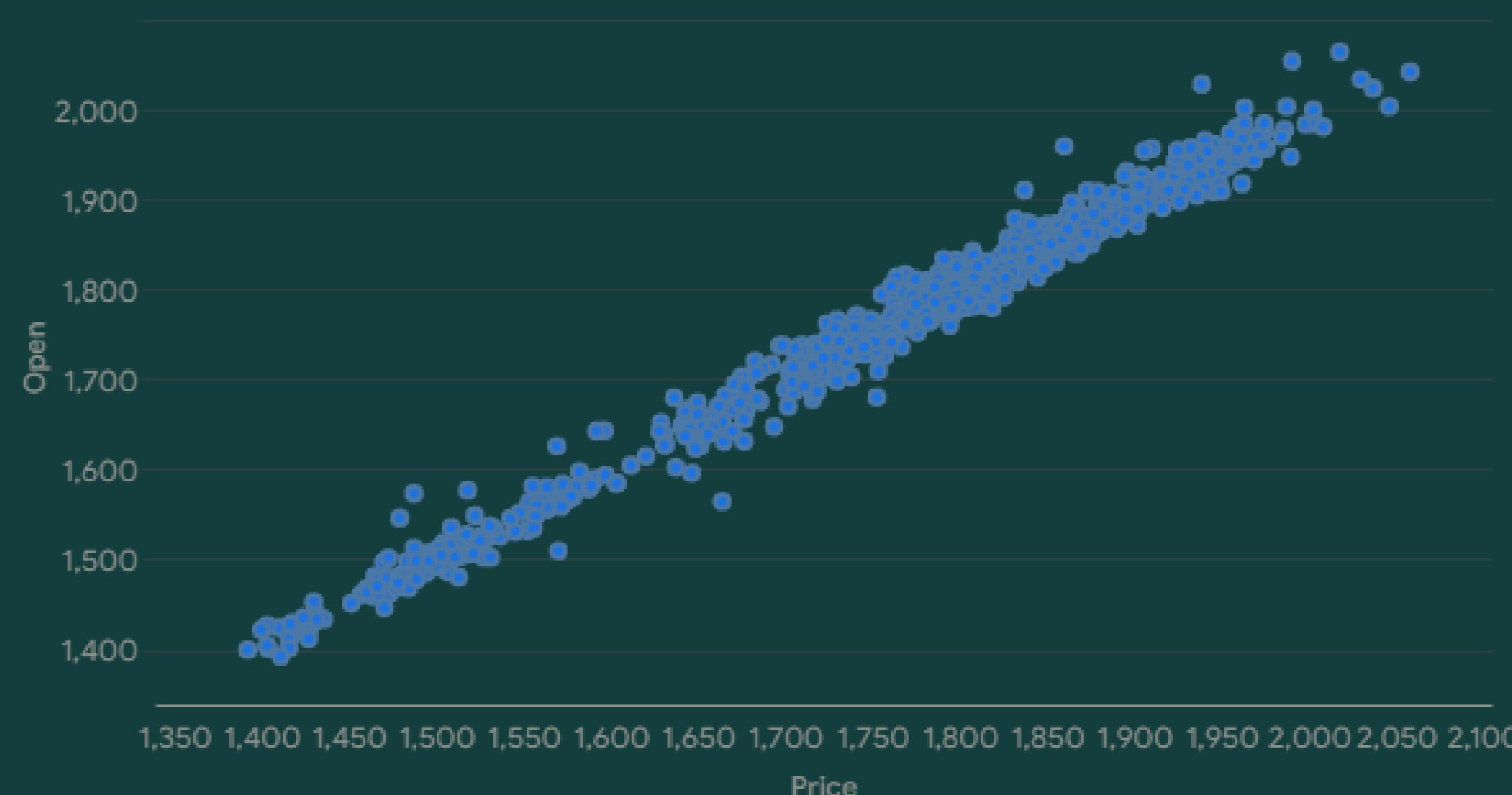
- Giá đóng cửa dao động từ 1620 đến 1950.
- Có một số ngày mà khối lượng giao dịch bằng 0.
- Giá mở cửa có mối tương quan tuyến tính mạnh với giá đóng cửa, với một vài ngoại lệ.
- Không có mối tương quan rõ ràng nào giữa giá đóng cửa và khối lượng giao dịch.



## Biểu đồ đường của Price theo thời gian



## Biểu đồ phân tán của Price và Open

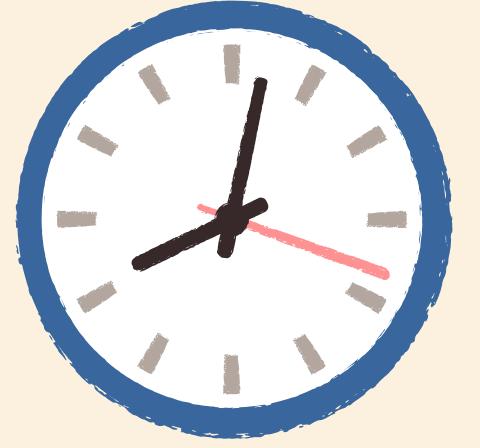


# PHÂN TÍCH

- Biểu đồ histogram của Vol. cho thấy rằng phần lớn các giá trị tập trung ở khoảng 0 đến 100. Vol. có một số giá trị ngoại lai, với giá trị lớn nhất là 680,82.
- Các giá trị ngoại lai của Price nằm trong khoảng 1392,4 đến 1418, trong khi các giá trị ngoại lai của Open nằm trong khoảng 1392,2 đến 1411,6.
- Mức chênh lệch trung bình giữa Price và Open là -0,0117009, với độ lệch chuẩn là 1,00033. Điều này cho thấy rằng giá đóng cửa trung bình thấp hơn một chút so với giá mở cửa. Mức chênh lệch nhỏ nhất là -5,46518 và mức chênh lệch lớn nhất là 6,35306.

# BAGGING





# Giới thiệu Bagging

Khái niệm Ensemble Learning và vai trò của nó trong Machine Learning.

Các loại phương pháp Ensemble phổ biến (Bagging, Boosting, Stacking).

Giới thiệu sơ lược về Bagging và ứng dụng của nó.



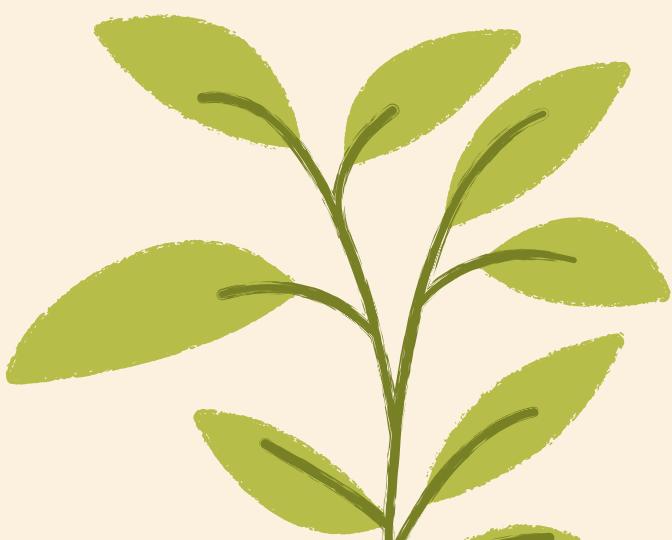
# Ensemble Learning

- Là kỹ thuật kết hợp nhiều mô hình học máy riêng lẻ (base learners) để tạo ra một mô hình mạnh mẽ hơn.
- Giúp cải thiện độ chính xác, tính ổn định và khả năng tổng quát hóa của mô hình.



# Các phương pháp Ensemble phổ biến

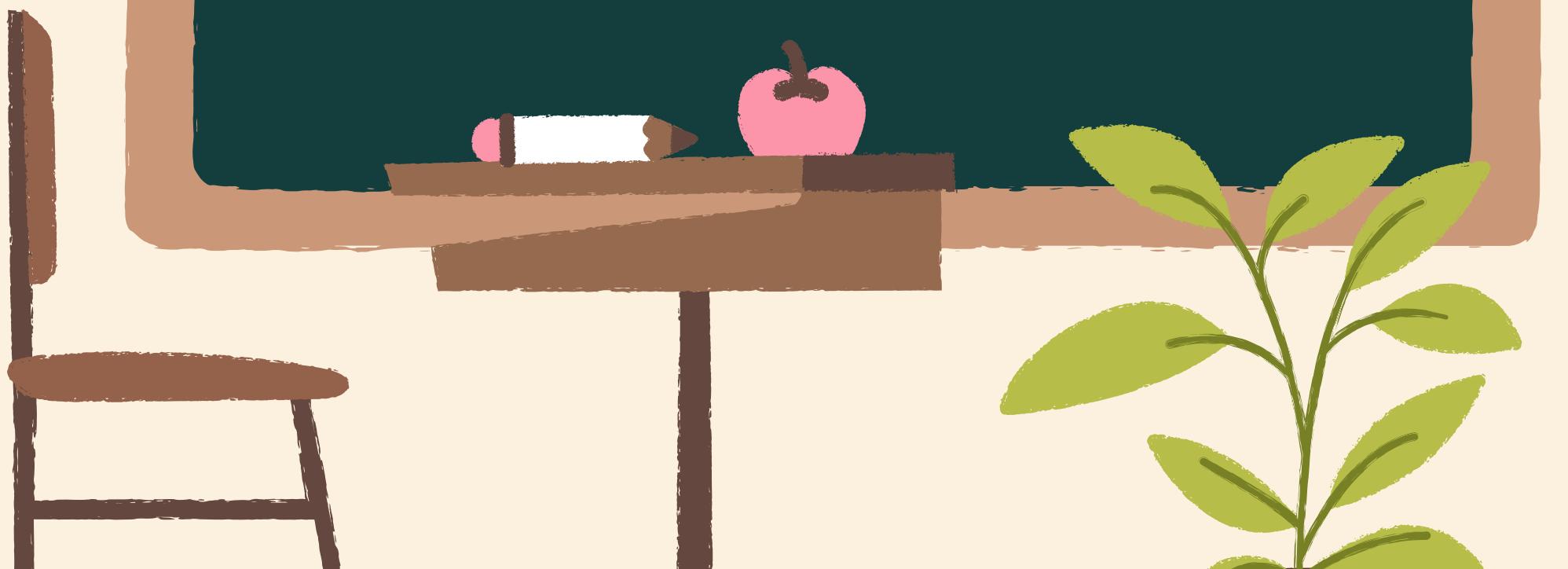
- Bagging: Huấn luyện các mô hình cơ sở trên các tập dữ liệu con được tạo bằng phương pháp bootstrap.
- Boosting: Huấn luyện tuần tự các mô hình cơ sở, mỗi mô hình tập trung vào việc sửa lỗi của mô hình trước đó.
- Stacking: Kết hợp dự đoán từ các mô hình cơ sở bằng một mô hình meta-learner.



# Bagging (Bootstrap Aggregating)

- Là kỹ thuật lấy mẫu có hoàn lại từ tập dữ liệu gốc để tạo ra các tập dữ liệu con.
- Mỗi tập dữ liệu con có cùng kích thước với tập dữ liệu gốc nhưng có thể chứa các bản sao của một số mẫu.

- Tập dữ liệu gốc: {A, B, C, D}
- Các tập dữ liệu con có thể tạo ra:  
{A, A, C, D}, {B, C, C, B},  
{A, B, D, D}, ...



VÍ DỤ



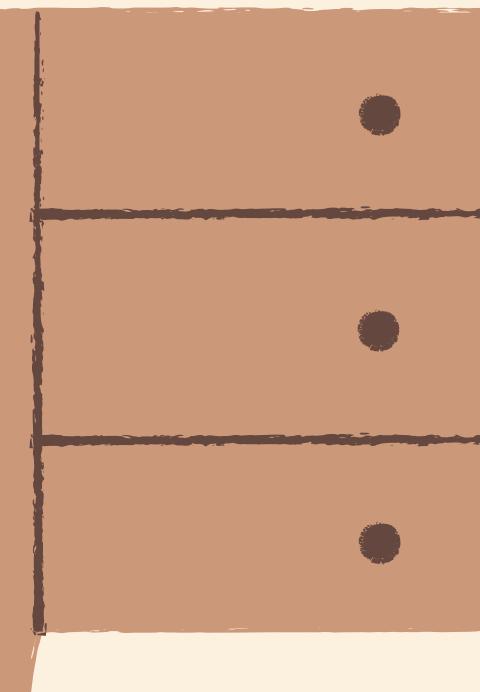


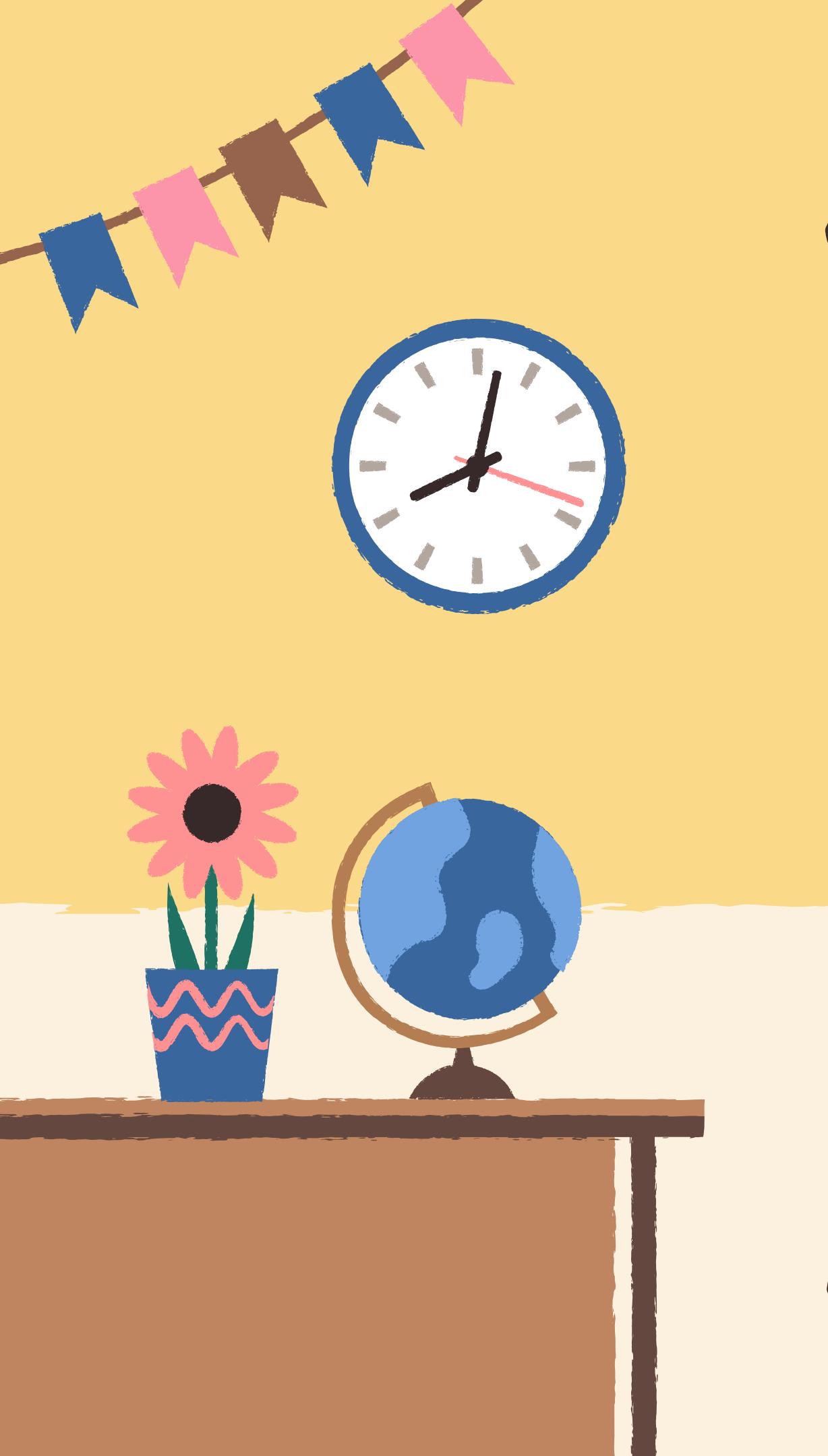
# Thuật toán Bagging

- Tạo các tập dữ liệu con: Sử dụng bootstrapping để tạo ra  $N$  tập dữ liệu con từ tập dữ liệu gốc.
- Huấn luyện các mô hình cơ sở: Huấn luyện  $N$  mô hình cơ sở (ví dụ: cây quyết định) trên  $N$  tập dữ liệu con.
- Tổng hợp kết quả: Kết hợp dự đoán từ  $N$  mô hình cơ sở bằng cách:
  - Trung bình cộng (bài toán hồi quy): Lấy trung bình cộng các giá trị dự đoán.
  - Bỏ phiếu (bài toán phân loại): Chọn lớp được dự đoán nhiều nhất.

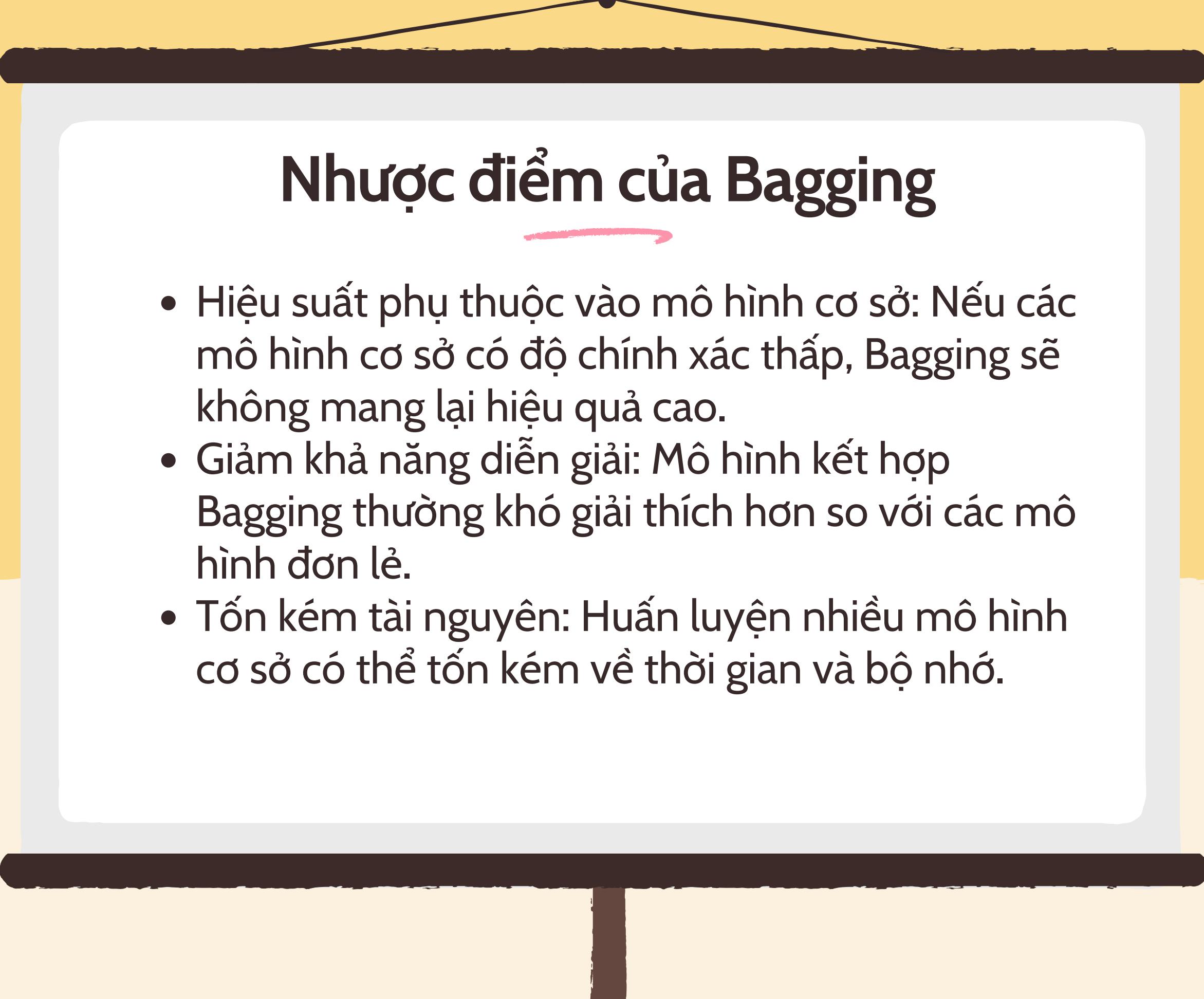
## Ưu điểm của Bagging

- Giảm phương sai: Kết hợp nhiều mô hình giúp giảm sự biến động của kết quả dự đoán, tăng tính ổn định.
- Giảm overfitting: Đặc biệt hiệu quả với các mô hình cơ sở dễ bị overfitting như cây quyết định.
- Dễ dàng triển khai: Thuật toán đơn giản, dễ hiểu và dễ cài đặt.
- Song song hóa: Việc huấn luyện các mô hình cơ sở có thể được thực hiện song song, giúp tiết kiệm thời gian.





## Nhược điểm của Bagging



- Hiệu suất phụ thuộc vào mô hình cơ sở: Nếu các mô hình cơ sở có độ chính xác thấp, Bagging sẽ không mang lại hiệu quả cao.
- Giảm khả năng diễn giải: Mô hình kết hợp Bagging thường khó giải thích hơn so với các mô hình đơn lẻ.
- Tốn kém tài nguyên: Huấn luyện nhiều mô hình cơ sở có thể tốn kém về thời gian và bộ nhớ.

# TÓM TẮT

- Bagging là một kỹ thuật Ensemble Learning đơn giản, hiệu quả, giúp cải thiện độ chính xác và tính ổn định của mô hình.
- Random Forest là một ứng dụng phổ biến của Bagging, mang lại hiệu quả cao trong nhiều bài toán thực tế.
- Hạn chế: Bagging có thể tốn kém tài nguyên và khó giải thích mô hình.
- Hướng phát triển: Nghiên cứu các phương pháp Bagging cải tiến, kết hợp với các kỹ thuật khác để nâng cao hiệu quả.



A classroom-themed illustration featuring a chalkboard, bookshelves, and school supplies. The chalkboard in the center has a white speech bubble containing the text "THANK YOU!". The background includes shelves with books, a globe, and various classroom items like a pencil and an apple.

THANK YOU!