

HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.



ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Một số mô hình học máy cơ bản cho bài toán phân loại

ONE LOVE. ONE FUTURE.

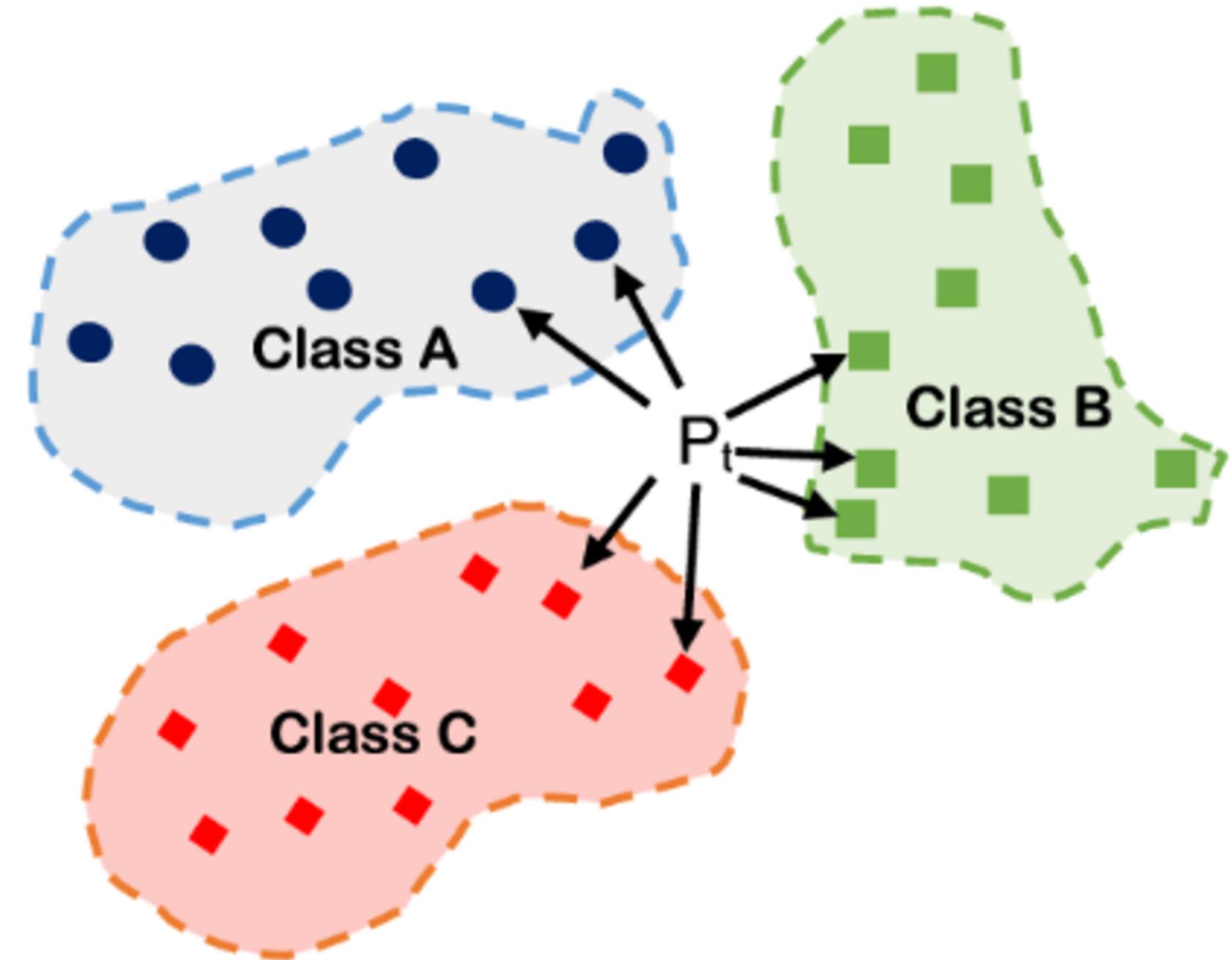
Nội dung chính

- Các mô hình học máy
 - K-Nearest Neighbor
 - Naive Bayes Classifier
 - Perceptron Learning Algorithm, Support Vector Machine
 - Decision Tree, Random Forest
- Ứng dụng trong bài phân loại



K-Nearest Neighbor

- KNN chỉ lưu trữ mà không học gì từ dữ liệu huấn luyện
- Khi cần dự đoán kết quả dữ liệu mới, thuật toán dựa vào K điểm dữ liệu gần nó nhất



K-Nearest Neighbor

- KNN có thể được sử dụng trong các bài toán phân loại, hồi quy, xây dựng hệ thống đề xuất,...
- Ưu điểm của KNN:
 - Đơn giản, dễ hiểu, quá trình tính toán không phức tạp
 - Dễ cập nhật thêm dữ liệu mới
 - Không có yêu cầu về phân phối dữ liệu
- Nhược điểm của KNN:
 - Với K nhỏ thì nhạy cảm với nhiễu
 - K càng lớn thì thời gian tính toán càng tăng
 - Ảnh hưởng đến bộ nhớ lưu trữ dữ liệu lớn



Naive Bayes Classifier

- Với một điểm dữ liệu \mathbf{x} và các nhãn 1, 2, ..., C

$$c = \operatorname{argmax}_c p(c|\mathbf{x}) = \operatorname{argmax}_c \frac{p(\mathbf{x}|c)p(c)}{p(\mathbf{x})} = \operatorname{argmax}_c p(\mathbf{x}|c)p(c)$$

$$p(\mathbf{x}|c) = p(x_1, x_2, \dots, x_d|c) = \prod_{i=1}^d p(x_i|c)$$

$$c = \operatorname{argmax}_{c \in \{1, \dots, C\}} p(c) \prod_{i=1}^d p(x_i|c) = \operatorname{argmax}_{c \in \{1, \dots, C\}} \left(\log(p(c)) + \sum_{i=1}^d \log(p(x_i|c)) \right)$$



Naive Bayes Classifier

- Gaussian naive Bayes:

$$p(x_i|c) = p(x_i|\mu_{ci}, \sigma_{ci}^2) = \frac{1}{\sqrt{2\pi\sigma_{ci}^2}} \exp\left(-\frac{(x_i - \mu_{ci})^2}{2\sigma_{ci}^2}\right)$$

- Multinomial naive Bayes:

$$p(x_i|c) = \hat{\lambda}_{ci} = \frac{N_{ci} + \alpha}{N_c + d\alpha}$$

- Bernoulli naive Bayes:

$$p(x_i|c) = p(i|c)x_i + (1 - p(i|c))(1 - x_i)$$

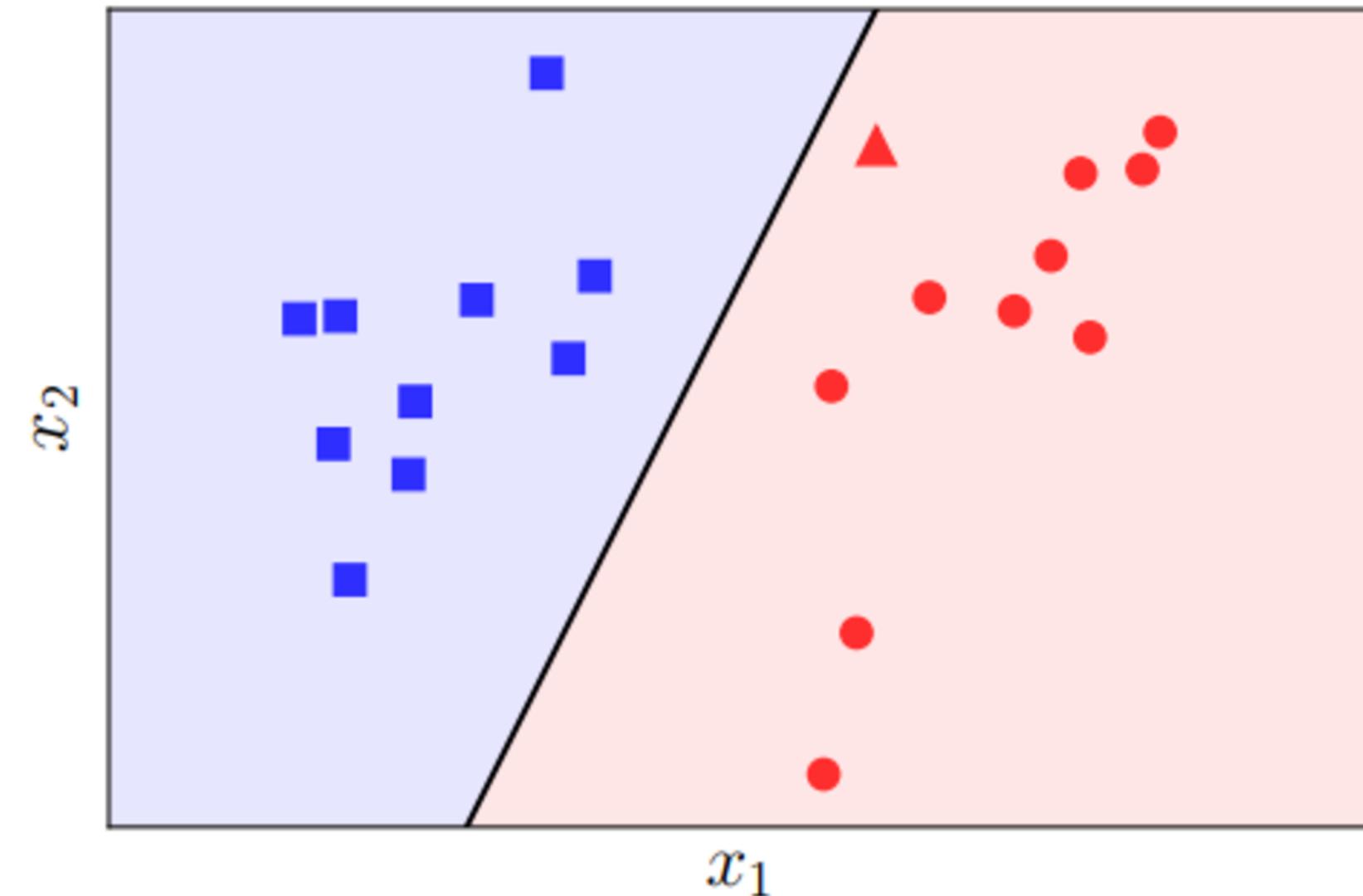
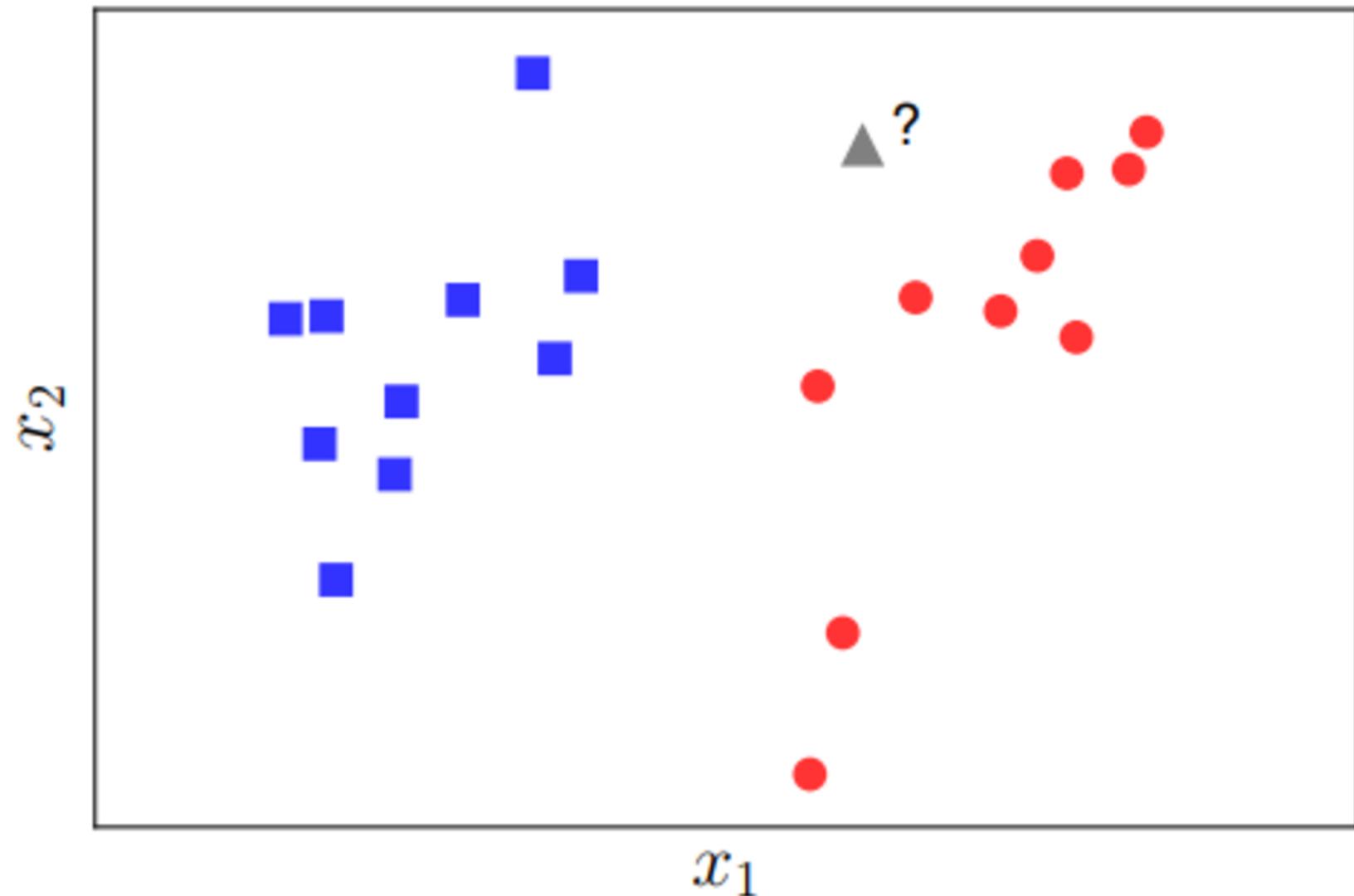


Naive Bayes Classifier

- Naive Bayes dùng trong các bài toán phân loại, đặc biệt là phân loại văn bản, hợp với các bài toán đa nhãn
- Ưu điểm của Naive Bayes:
 - Đơn giản và nhanh chóng
 - Hoạt động với các đặc trưng mà một phần liên tục, một phần rời rạc
- Nhược điểm của Naive Bayes:
 - Giả định các biến độc lập là bất khả thi trong thực tế



Perceptron Learning Algorithm



Perceptron Learning Algorithm

- Xác định nhãn bằng phép toán:

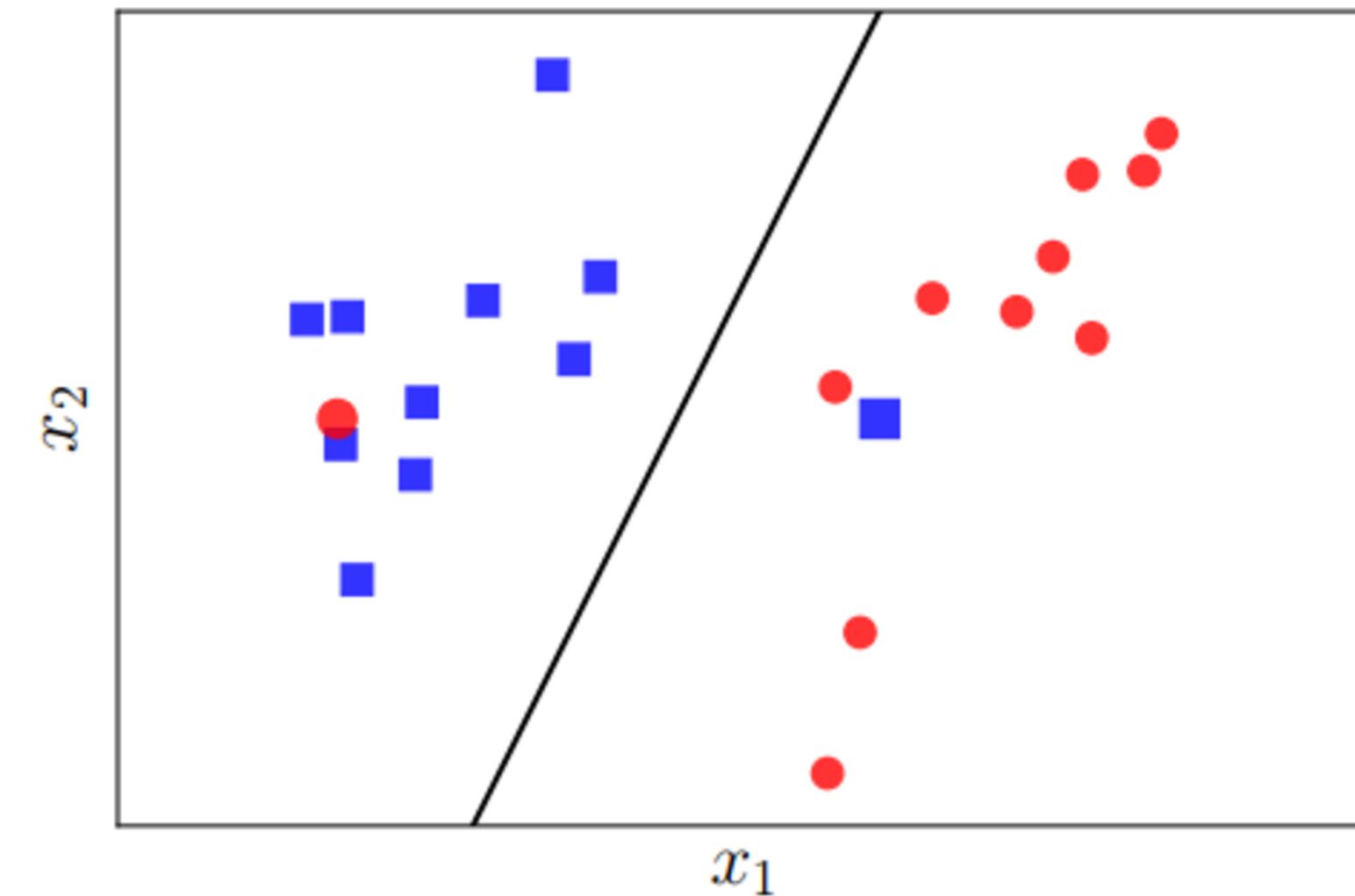
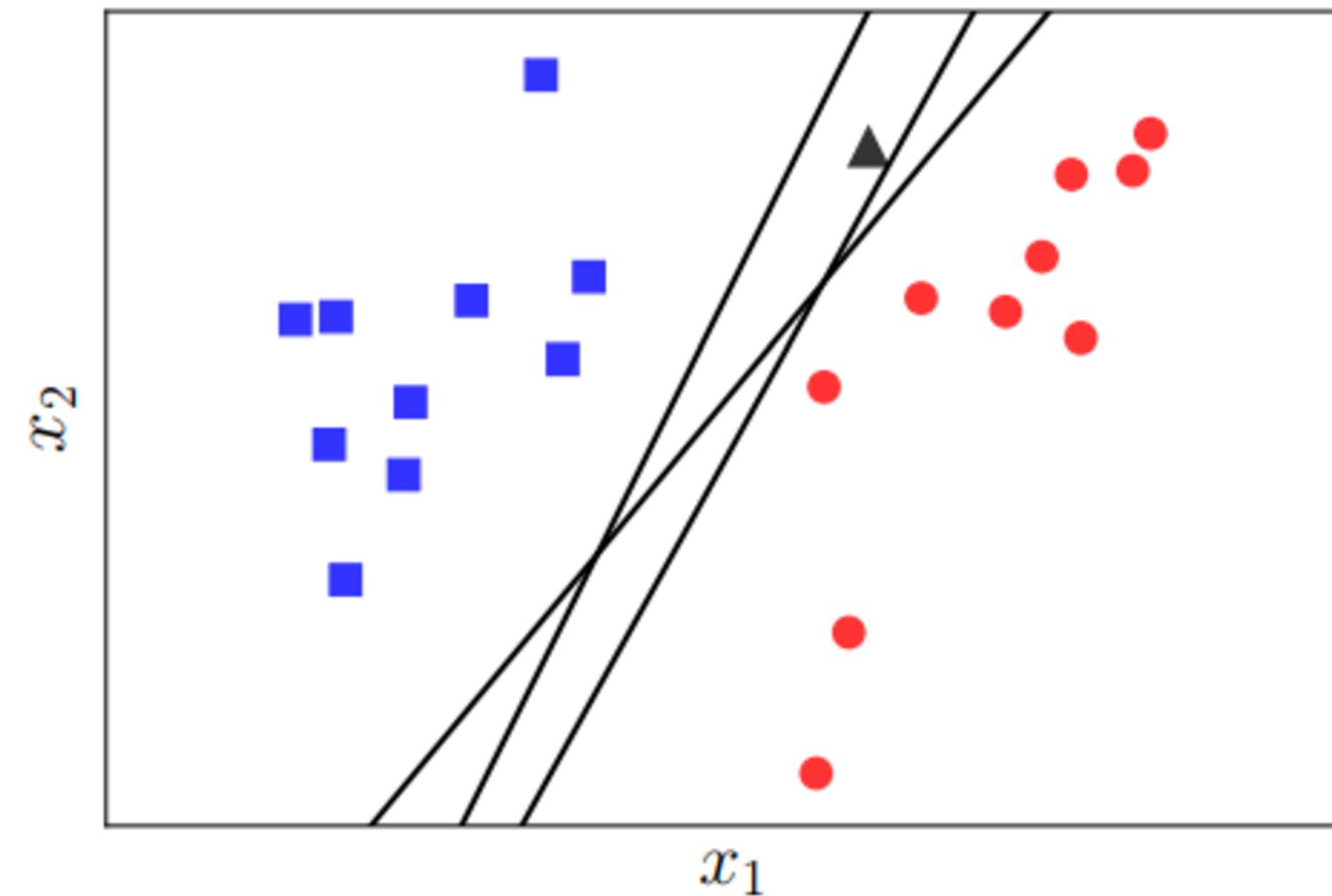
$$\text{label}(\mathbf{x}) = \begin{cases} 1 & \text{nếu } \mathbf{w}^T \mathbf{x} \geq 0 \\ -1 & \text{o.w.} \end{cases} = \text{sgn}(\mathbf{w}^T \mathbf{x}), \text{ giả sử rằng sgn}(0) = 1$$

- Hàm mất mát: $J(\mathbf{w}) = \sum_{\mathbf{x}_i \in \mathcal{M}} (-y_i \mathbf{w}^T \mathbf{x}_i)$
- Quy tắc cập nhật: $\mathbf{w}_{t+1} = \mathbf{w}_t + y_i \mathbf{x}_i$



Perceptron Learning Algorithm

- PLA có thể cho vô số nghiệm
- PLA đòi hỏi hai lớp dữ liệu có khả năng phân tách tuyến tính

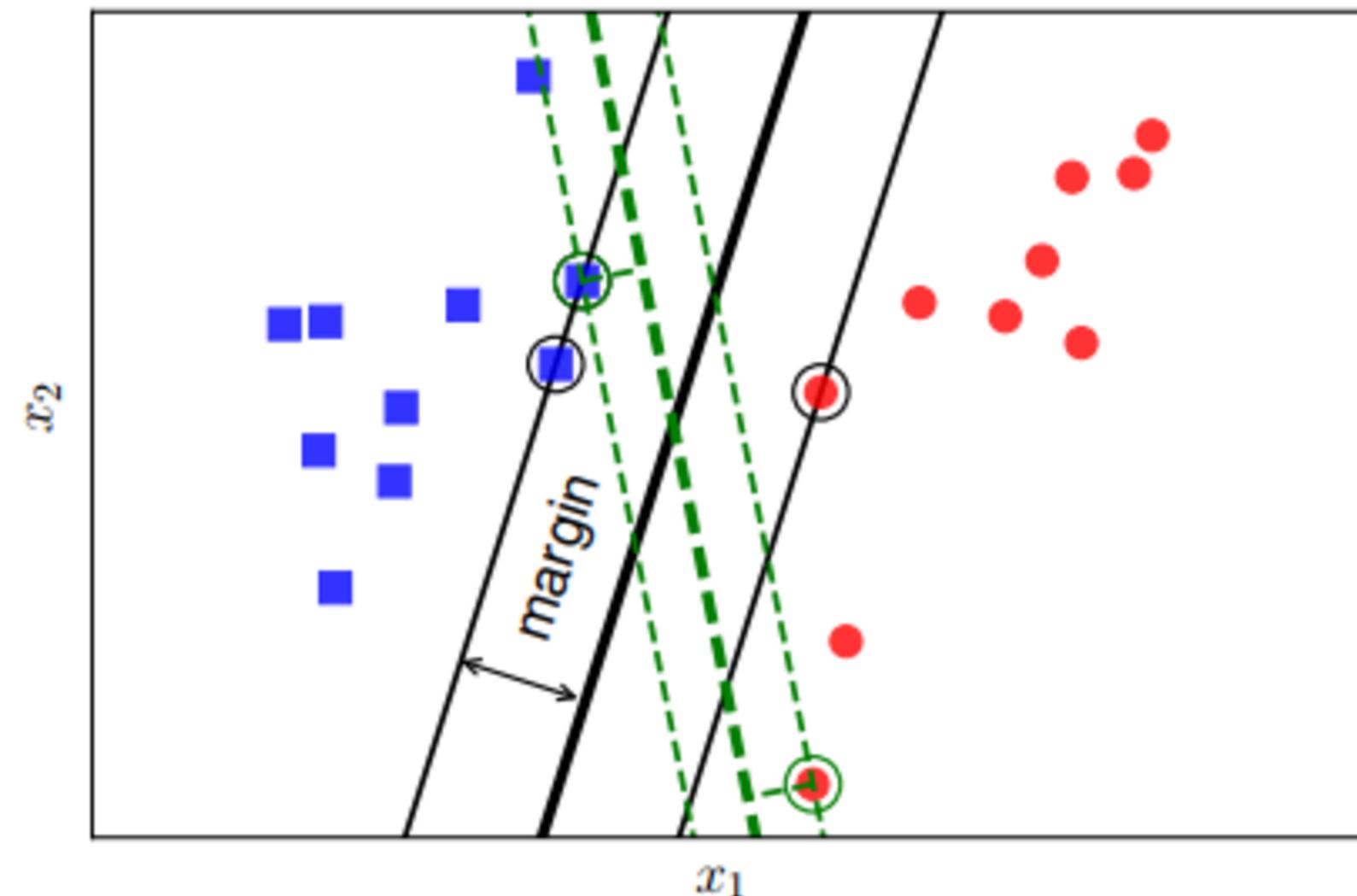
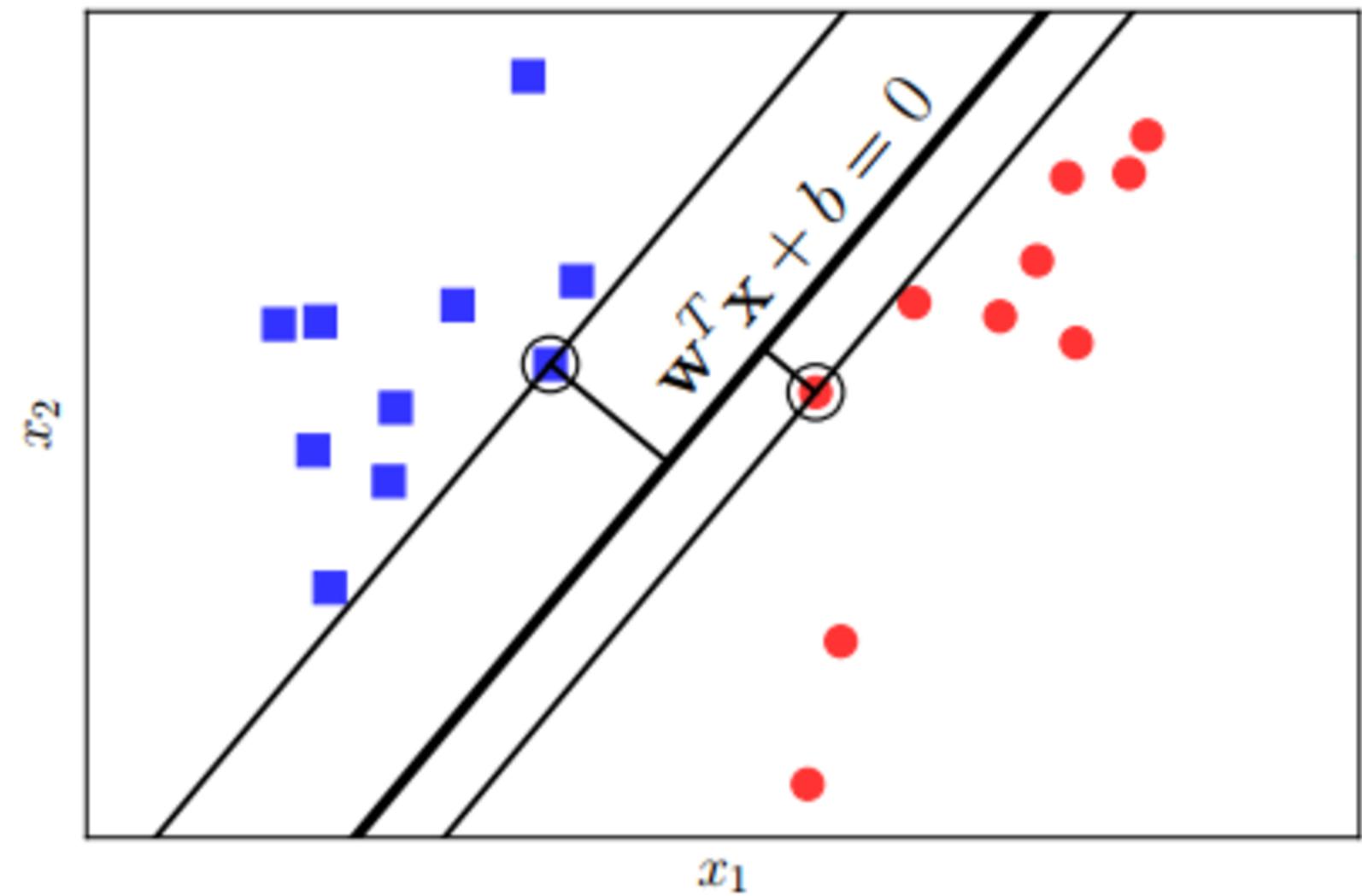


Perceptron Learning Algorithm

- Ưu điểm của PLA:
 - Đơn giản, dễ hiểu, dễ triển khai
- Nhược điểm của PLA:
 - Nhạy cảm với nhiễu
 - Không giải được bài toán không tách tuyến tính
 - Không thể biểu diễn các ranh giới quyết định phức tạp



Support Vector Machine



Support Vector Machine

- Khoảng cách gần nhất từ một điểm đến đường phân cách

$$\text{margin} = \min_n \frac{y_n(\mathbf{w}^T \mathbf{x}_n + b)}{\|\mathbf{w}\|_2}$$

- Bài toán tối ưu của SVM:

$$(\mathbf{w}, b) = \arg \max_{\mathbf{w}, b} \left\{ \min_n \frac{y_n(\mathbf{w}^T \mathbf{x}_n + b)}{\|\mathbf{w}\|_2} \right\} = \arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|_2} \min_n y_n(\mathbf{w}^T \mathbf{x}_n + b) \right\}$$

$$(\mathbf{w}, b) = \arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|_2^2$$

$$\text{thoả mãn: } 1 - y_n(\mathbf{w}^T \mathbf{x}_n + b) \leq 0, \forall n = 1, 2, \dots, N$$



Support Vector Machine

- Hàm đối ngẫu Lagrange

$$g(\boldsymbol{\lambda}) = \min_{\mathbf{w}, b} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda}) = \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|_2^2 + \sum_{n=1}^N \lambda_n (1 - y_n (\mathbf{w}^T \mathbf{x}_n + b))$$

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda}) = \mathbf{w} - \sum_{n=1}^N \lambda_n y_n \mathbf{x}_n = \mathbf{0} \Rightarrow \mathbf{w} = \sum_{n=1}^N \lambda_n y_n \mathbf{x}_n$$

$$\nabla_b \mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda}) = \sum_{n=1}^N \lambda_n y_n = 0$$

$$g(\boldsymbol{\lambda}) = \sum_{n=1}^N \lambda_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \lambda_n \lambda_m y_n y_m \mathbf{x}_n^T \mathbf{x}_m$$



Support Vector Machine

- Bài toán đối ngẫu Lagrange

$$\boldsymbol{\lambda} = \arg \max_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda})$$

thoả mãn: $\boldsymbol{\lambda} \succeq 0$

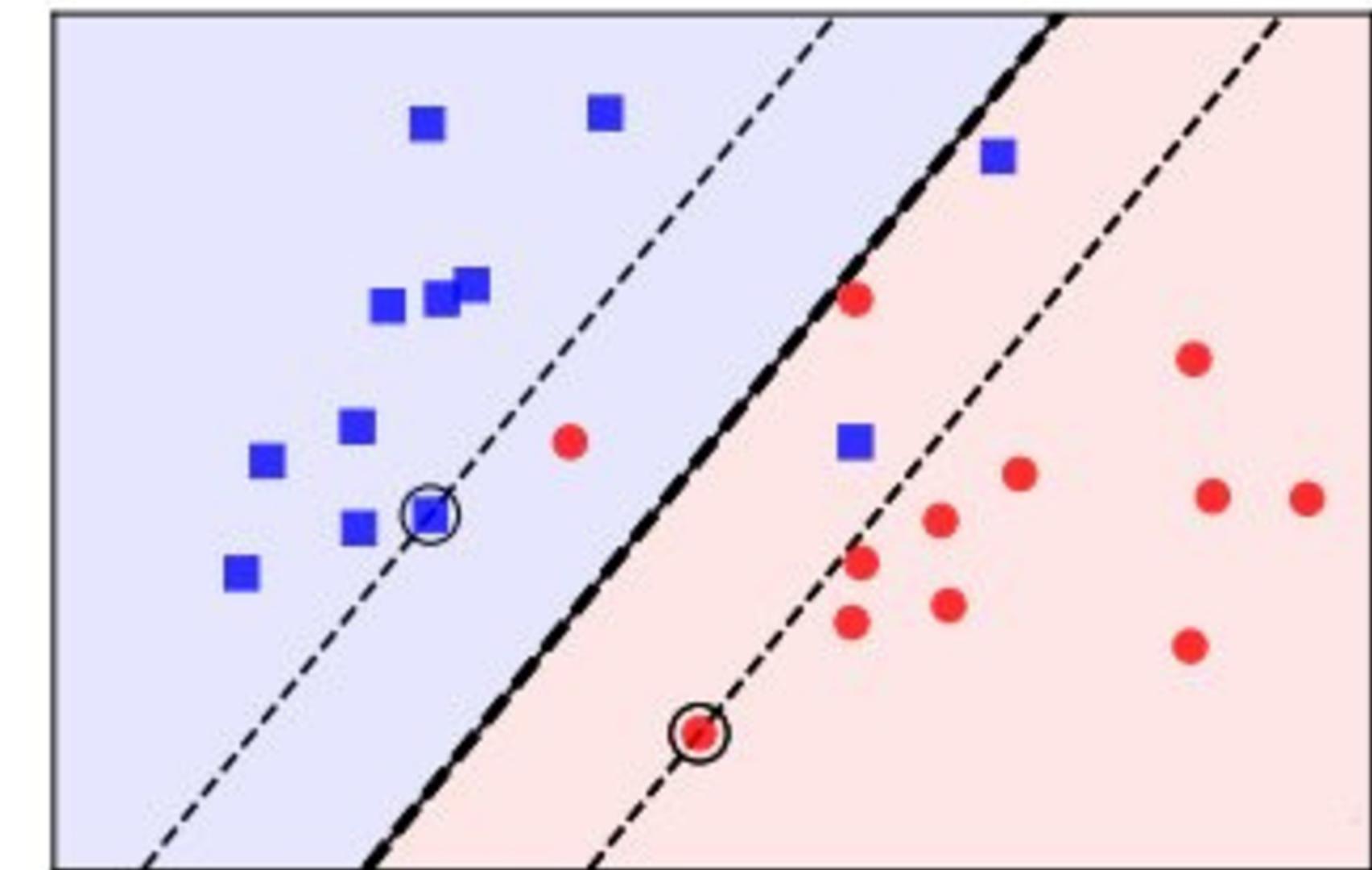
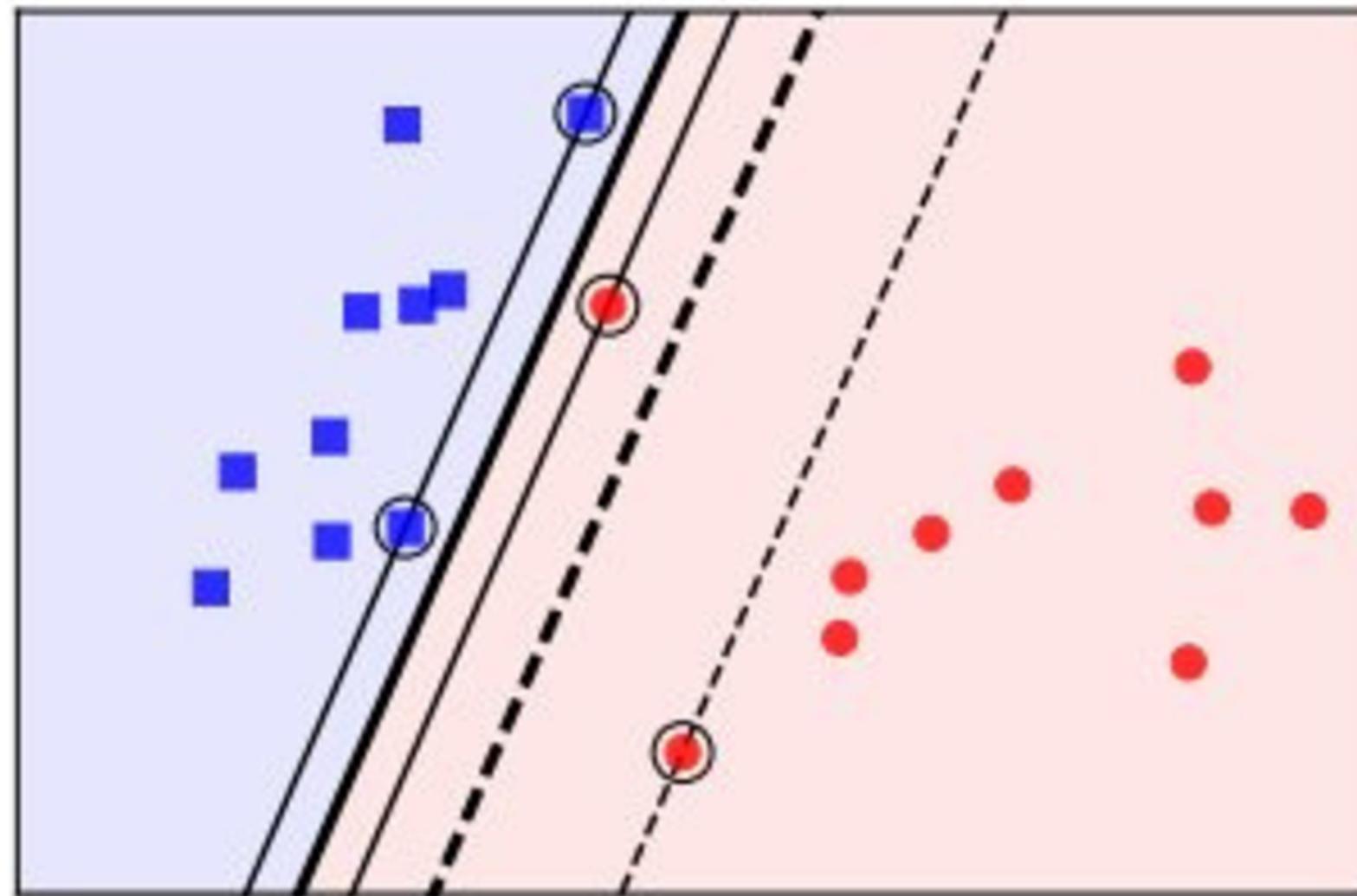
$$\sum_{n=1}^N \lambda_n y_n = 0$$

- Nghiệm của bài toán đối ngẫu thoả mãn một điều kiện quan trọng:

$$\lambda_n(1 - y_n(\mathbf{w}^T \mathbf{x}_n + b)) = 0, \quad \forall n = 1, 2, \dots, N$$



Soft-Margin Support Vector Machine

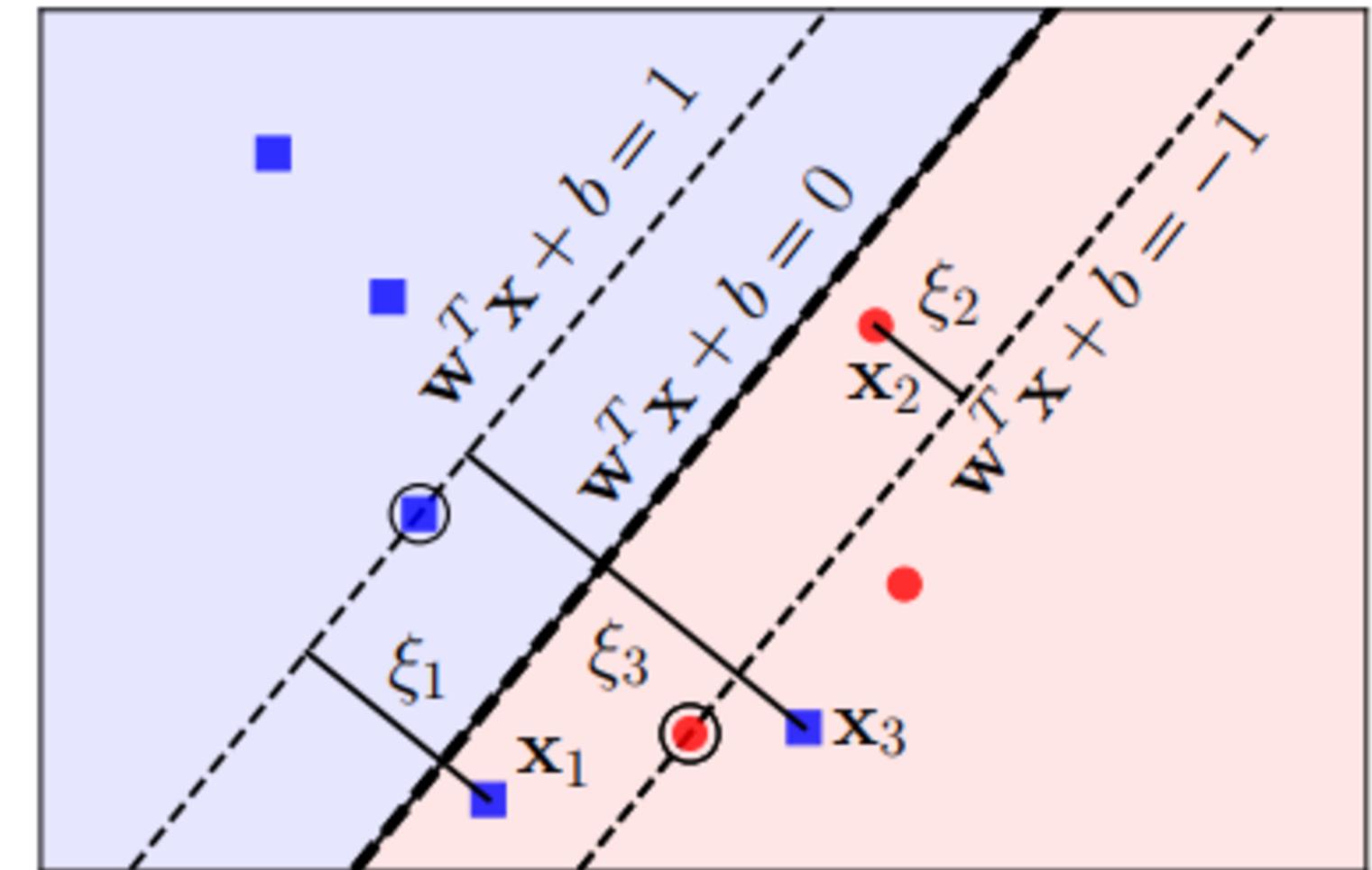


Soft-Margin Support Vector Machine

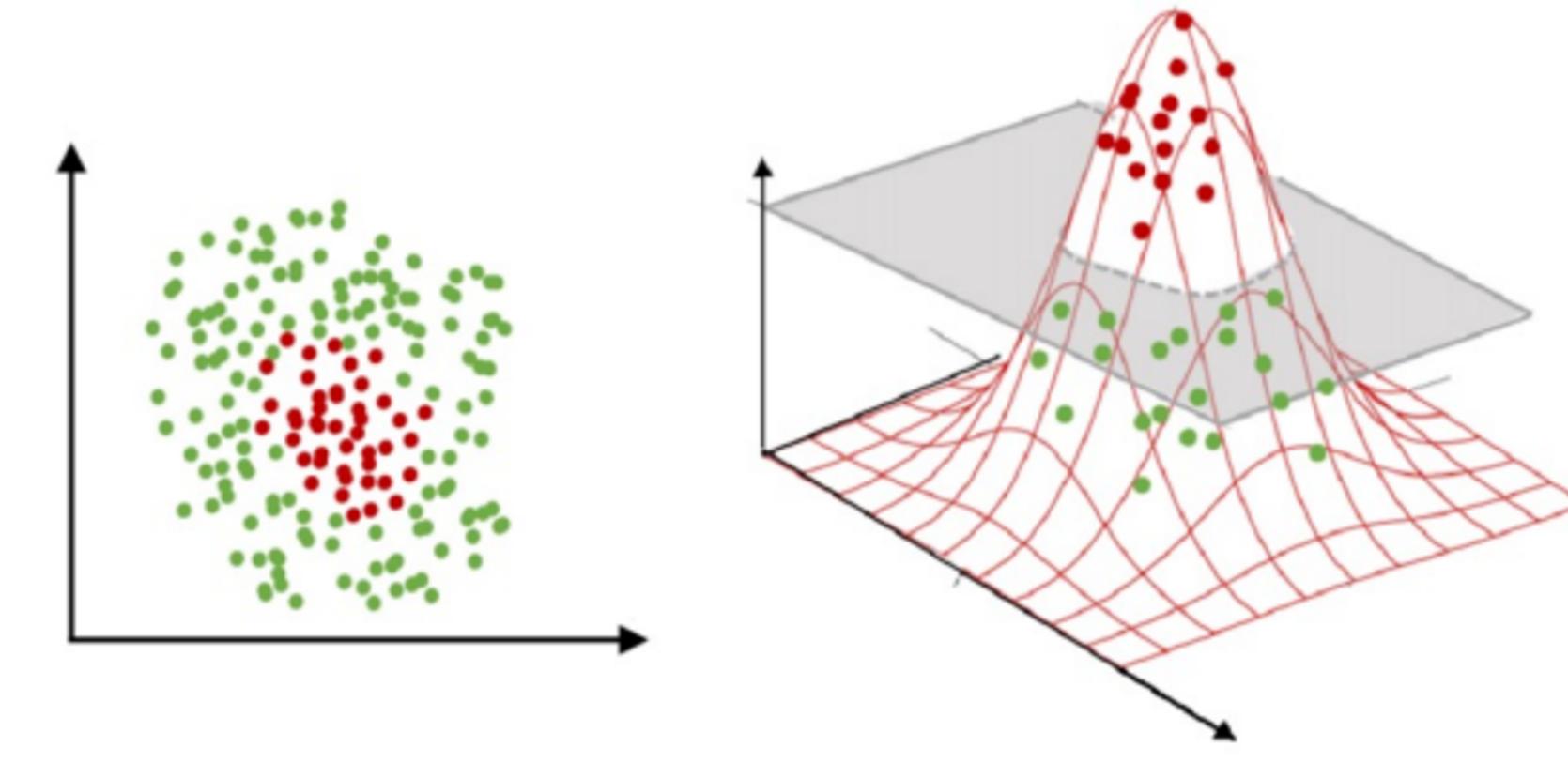
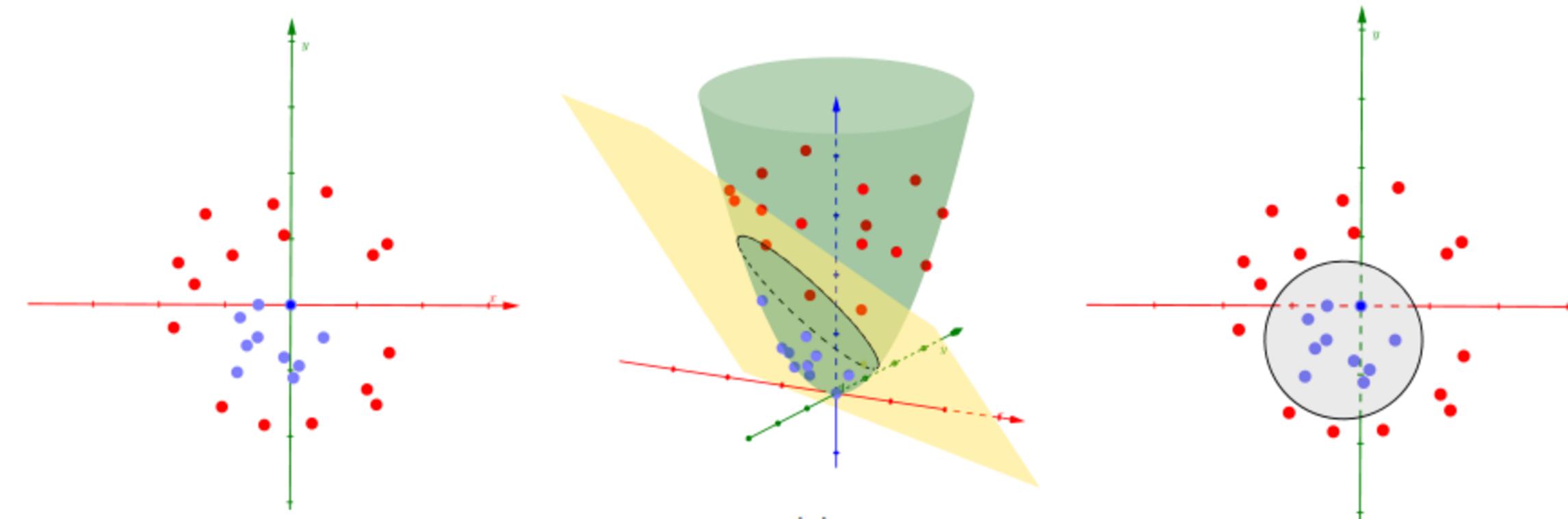
- Bài toán tối ưu mới cho soft-margin SVM:

$$(\mathbf{w}, b, \xi) = \arg \min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{n=1}^N \xi_n$$

thoả mãn: $1 - \xi_n - y_n(\mathbf{w}^T \mathbf{x}_n + b) \leq 0, \forall n = 1, 2, \dots, N$
 $-\xi_n \leq 0, \forall n = 1, 2, \dots, N$



Kernel Support Vector Machine



Kernel Support Vector Machine

- Kernel SVM là phương pháp đi tìm một hàm số biến đổi dữ liệu x từ không gian đặc trưng ban đầu thành dữ liệu trong không gian mới mà có thể dùng bộ phân lớp tuyến tính thông thường để giải quyết
- Các hàm biến đổi này thường tạo ra dữ liệu mới có số chiều lớn hơn số chiều ban đầu
- Một cách tiếp cận là dùng các hàm kernel mô tả quan hệ giữa hai điểm bất kỳ trong không gian mới



Kernel Support Vector Machine

- Bài toán tối ưu của SVM trong không gian mới:

$$\boldsymbol{\lambda} = \arg \max_{\boldsymbol{\lambda}} \sum_{n=1}^N \lambda_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \lambda_n \lambda_m y_n y_m \Phi(\mathbf{x}_n)^T \Phi(\mathbf{x}_m)$$

thoả mãn: $\sum_{n=1}^N \lambda_n y_n = 0$

$$0 \leq \lambda_n \leq C, \forall n = 1, 2, \dots, N$$



Kernel Support Vector Machine

- Bài toán tối ưu của SVM trong không gian mới:

$$\boldsymbol{\lambda} = \arg \max_{\boldsymbol{\lambda}} \sum_{n=1}^N \lambda_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \lambda_n \lambda_m y_n y_m k(\mathbf{x}_n, \mathbf{x}_m)$$

thoả mãn: $\sum_{n=1}^N \lambda_n y_n = 0$

$$0 \leq \lambda_n \leq C, \forall n = 1, 2, \dots, N$$



Kernel Support Vector Machine

- Linear Kernel: $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
- Polynomial Kernel: $k(\mathbf{x}, \mathbf{z}) = (r + \gamma \mathbf{x}^T \mathbf{z})^d$
- RBF Kernel: $k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \|\mathbf{x} - \mathbf{z}\|_2^2), \quad \gamma > 0$
- Sigmoid Kernel: $k(\mathbf{x}, \mathbf{z}) = \tanh(\gamma \mathbf{x}^T \mathbf{z} + r)$

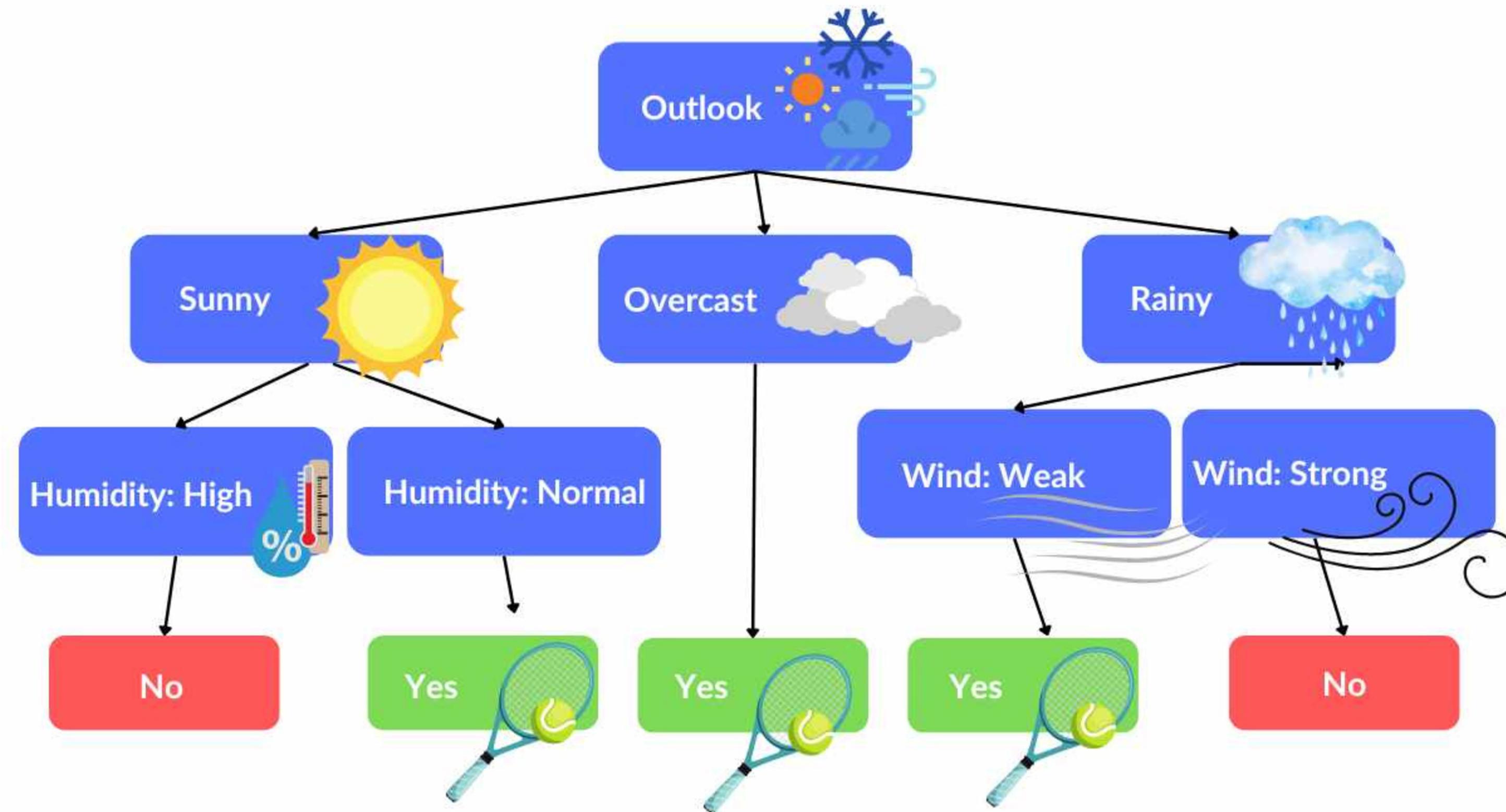


Support Vector Machine

- Ưu điểm của SVM:
 - Nghiệm SVM là duy nhất và toàn cục
 - Hiệu quả trong việc xử lý dữ liệu đa chiều
 - Có thể mô hình hóa ranh giới quyết định phi tuyến tính bằng cách sử dụng kernel
 - Dùng được cho cả bài toán phân loại và hồi quy
 - Bền trước nhiễu nhỏ
- Nhược điểm của SVM:
 - Khó diễn giải mô hình trong trường hợp kernel phi tuyến
 - Nhạy với các tham số, khó xác định tham số tối ưu
 - Tốn thời gian tính toán với bộ dữ liệu lớn



Decision Tree



- Hàm entropy đánh giá mức độ tinh khiết của phân phối xác suất của một sự kiện:

$$\mathbf{H}(\mathbf{p}) = - \sum_{i=1}^C p_i \log p_i$$

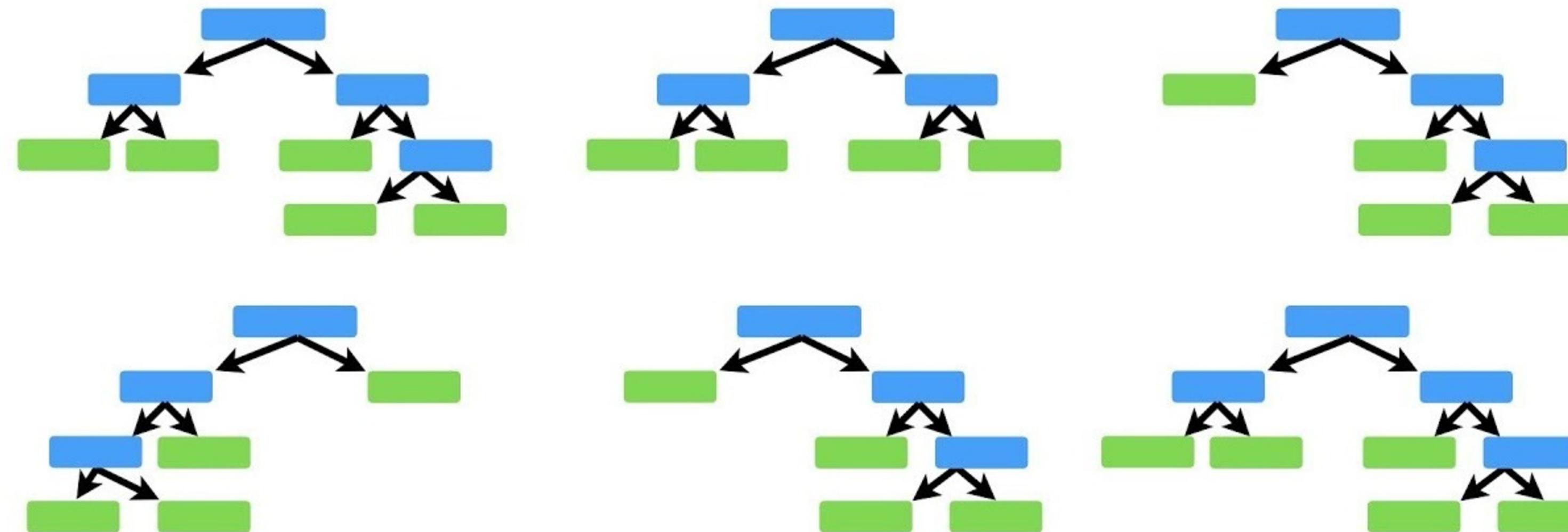
- Hàm số định nghĩa lượng thông tin thu được hay giá trị entropy giảm sau khi phân chia:

$$\begin{aligned}\mathbf{G}(x^{(j)}, t; \mathcal{S}) &= \mathbf{H}(\mathcal{S}) - \mathbf{H}(x^{(j)}, t; \mathcal{S}) \\ &= \mathbf{H}(\mathcal{S}) - \frac{N_0}{N} \mathbf{H}(\mathcal{S}_0) - \frac{N_1}{N} \mathbf{H}(\mathcal{S}_1)\end{aligned}$$

- Decision Tree được sử dụng trong bài toán phân loại, hồi quy, bài toán trích xuất luật,...
- Ưu điểm của Decision Tree:
 - Dễ hiểu, dễ giải thích, có thể trực quan hóa
 - Xử lý được cả dữ liệu số và dữ liệu phân loại
 - Ít bị ảnh hưởng bởi nhiễu
- Nhược điểm của Decision Tree:
 - Dễ bị overfit, đặc biệt khi cây phát triển quá sâu
 - Khó nắm bắt quan hệ tuyến tính
 - Chỉ tạo ra một kịch bản dự báo duy nhất cho mỗi một quan sát

Random Forest

- Thuật toán Random Forest tạo ra nhiều Decision Tree mà mỗi cây quyết định được huấn luyện dựa trên nhiều mẫu con khác nhau và kết quả dự báo là bầu cử từ toàn bộ những cây đó



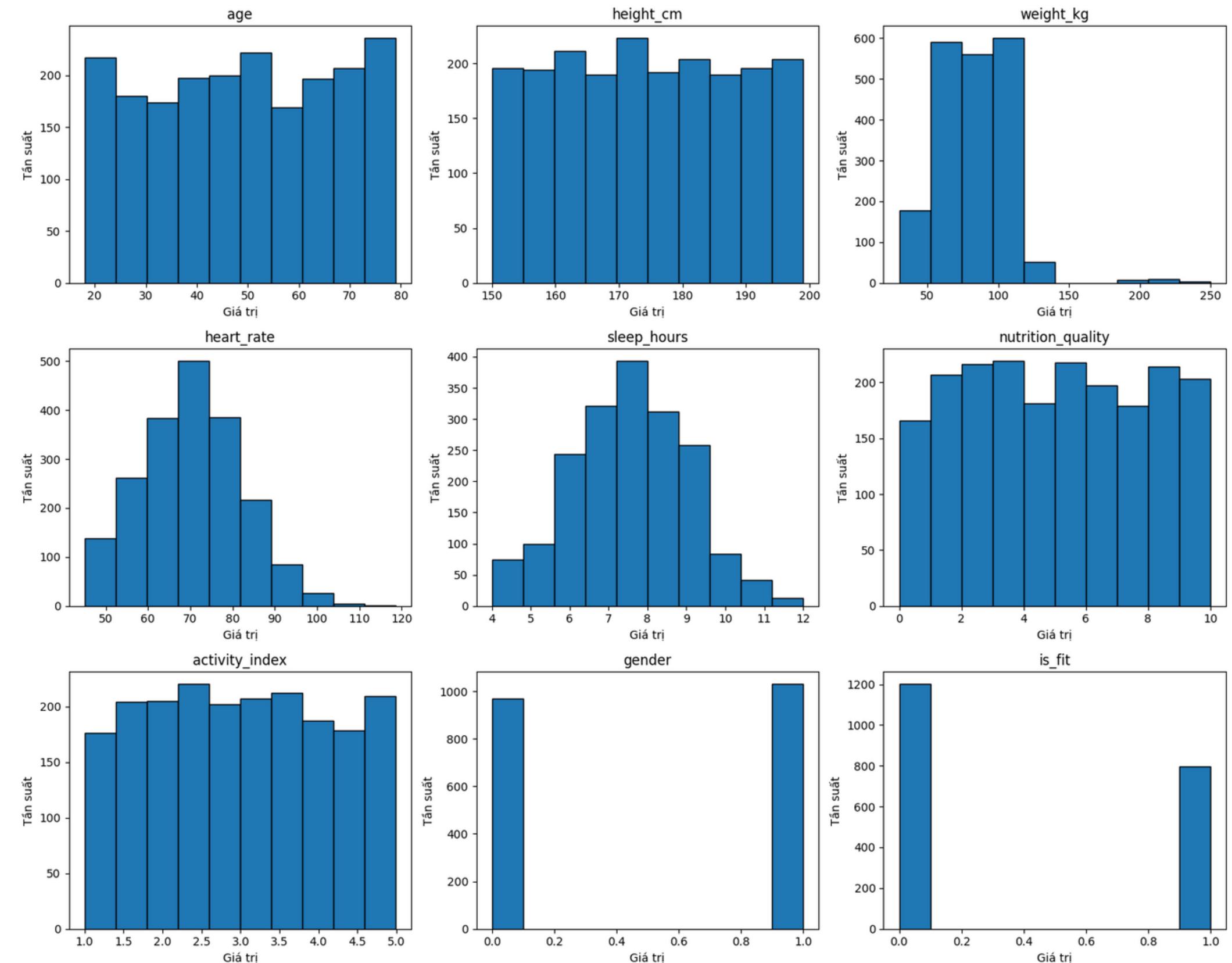
Random Forest

- Random Forest phổ biến trong các bài toán phân loại, hồi quy, trích xuất tầm quan trọng của đặc trưng
- Ưu điểm của Random Forest:
 - Giảm overfit so với Decision Tree
 - Nắm bắt được cả quan hệ tuyến tính và phi tuyến
 - Ổn định và ít nhạy cảm với nhiễu
 - Đánh giá được độ quan trọng của đặc trưng
- Nhược điểm của Random Forest:
 - Thời gian tính toán chậm hơn so với Decision Tree
 - Khó diễn giải
 - Vẫn có thể overfit nếu không kiểm soát



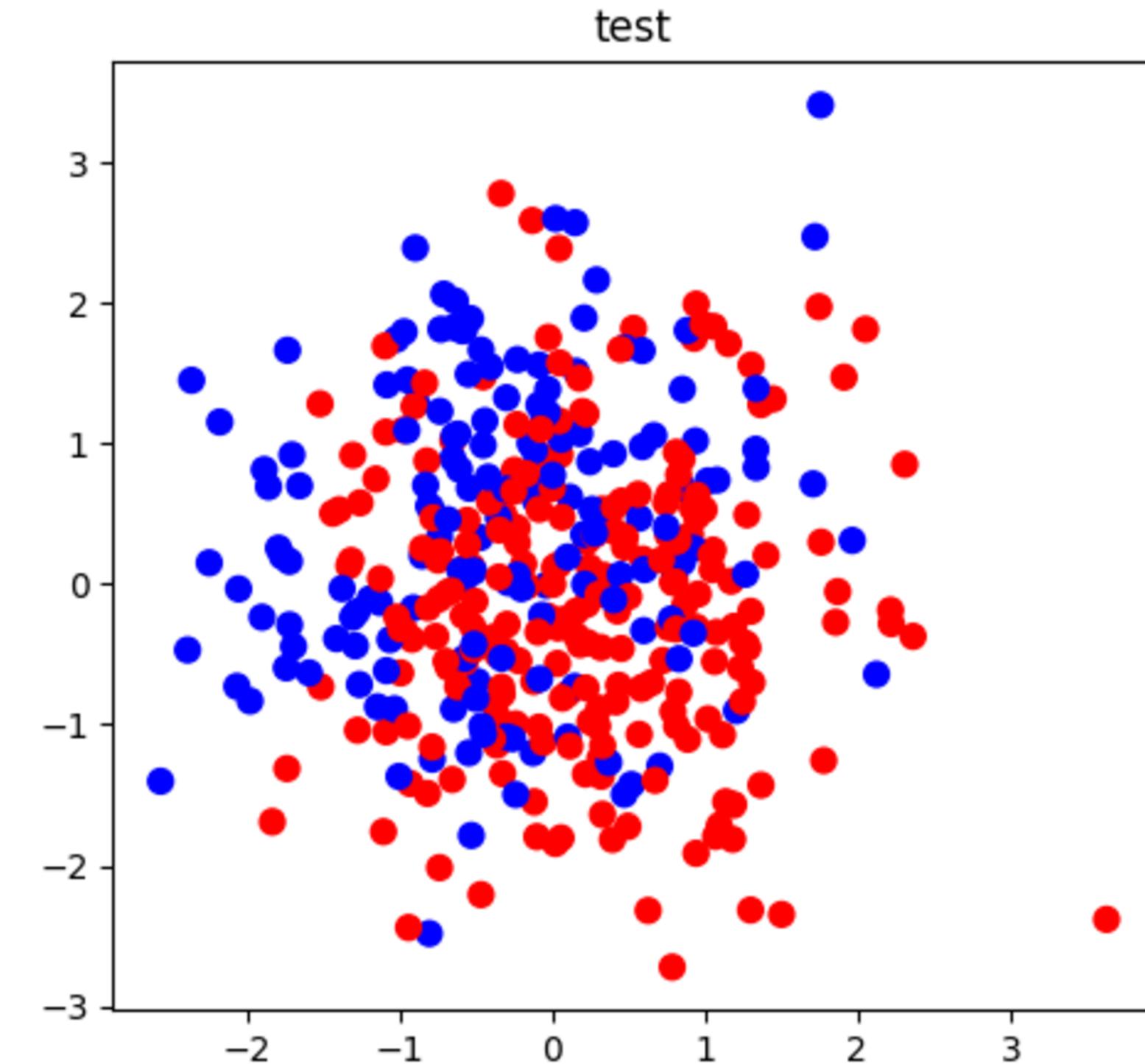
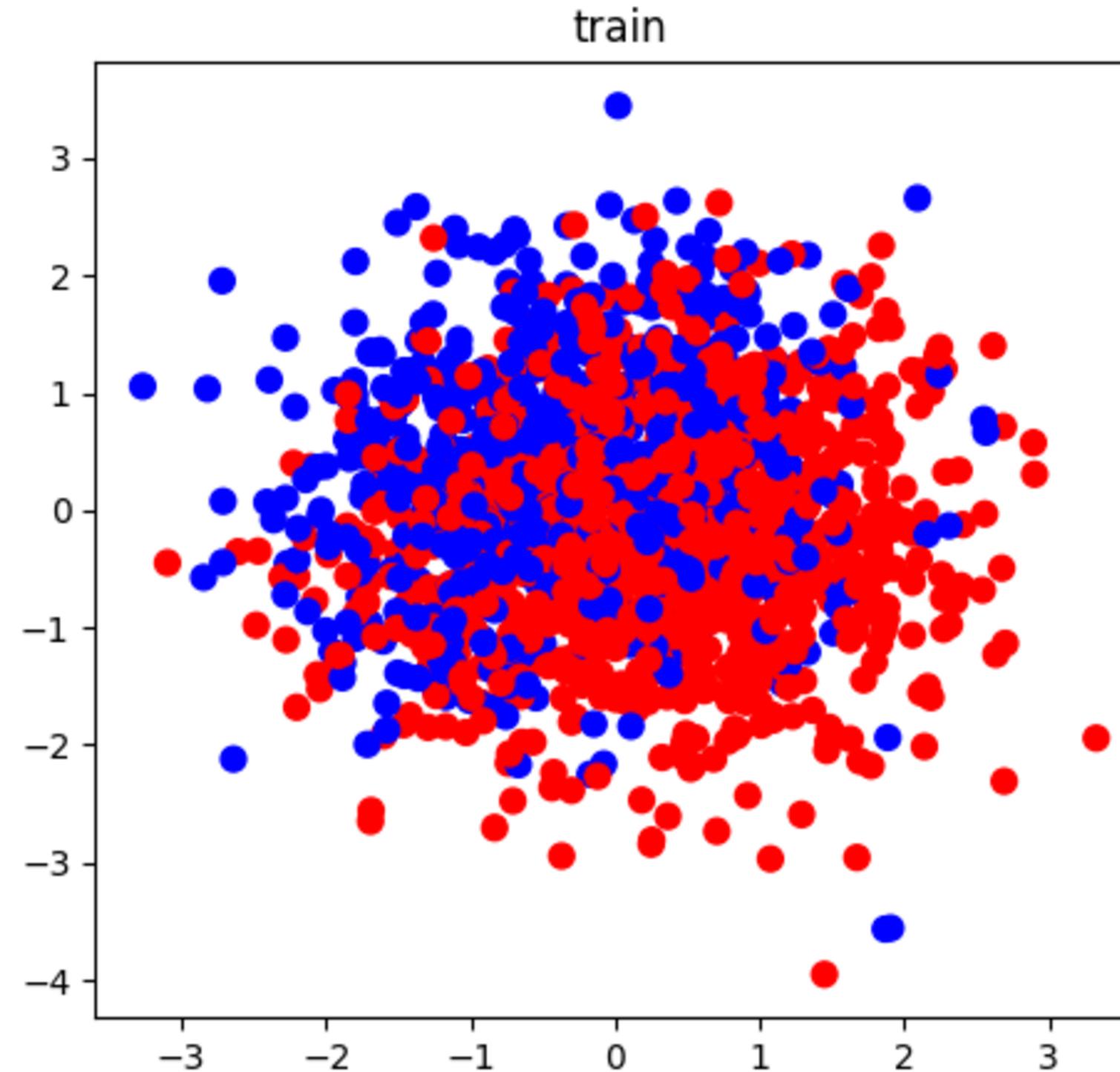
Ứng Dụng Trong Bài Toán Phân Loại

- Bộ dữ liệu: Fitness Classification Dataset
 - 2000 mẫu
 - Tỉ lệ nhãn 0, 1 là 3:2
- Đánh giá mô hình với Accuracy:
 - Trung bình: ~60%
 - Tốt: 65-75%
 - Xuất sắc: >80%



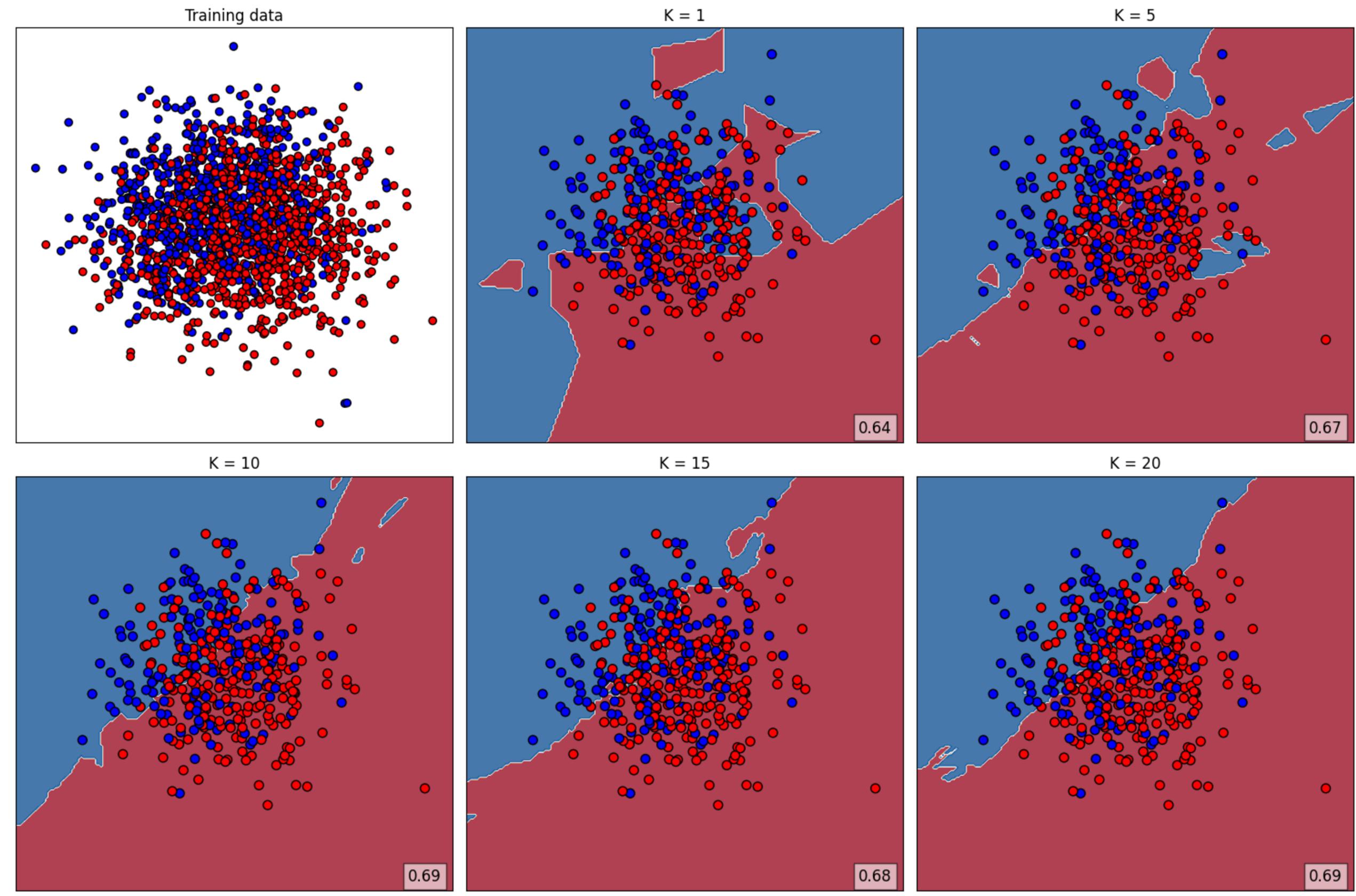
Ứng Dụng Trong Bài Toán Phân Loại

- Chia dữ liệu thành 2 phần train và test với tỉ lệ 8:2



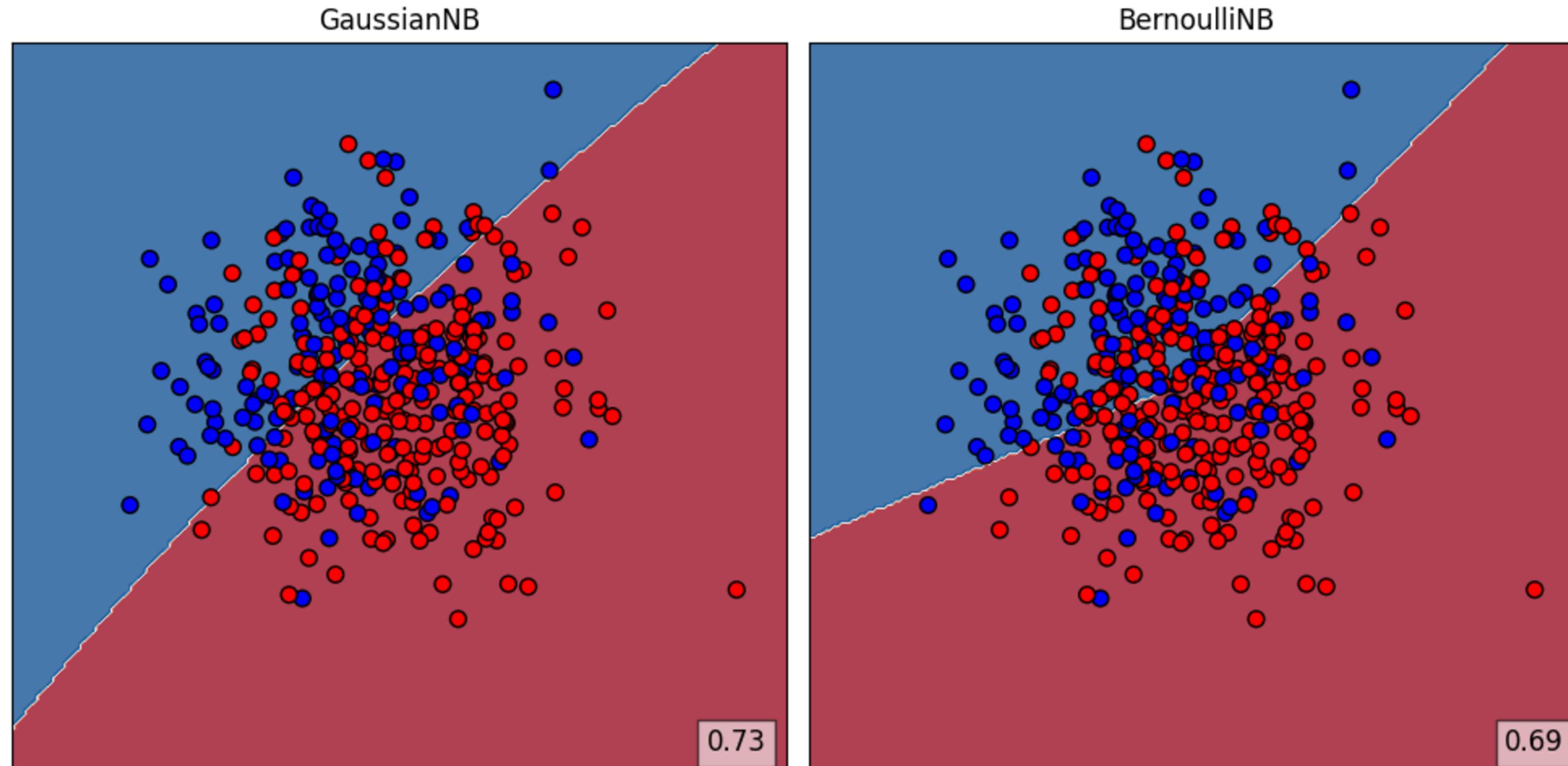
Ứng Dụng Trong Bài Toán Phân Loại

- KNN: Thử tham số K lần lượt các giá trị 1, 5, 10, 15, 20



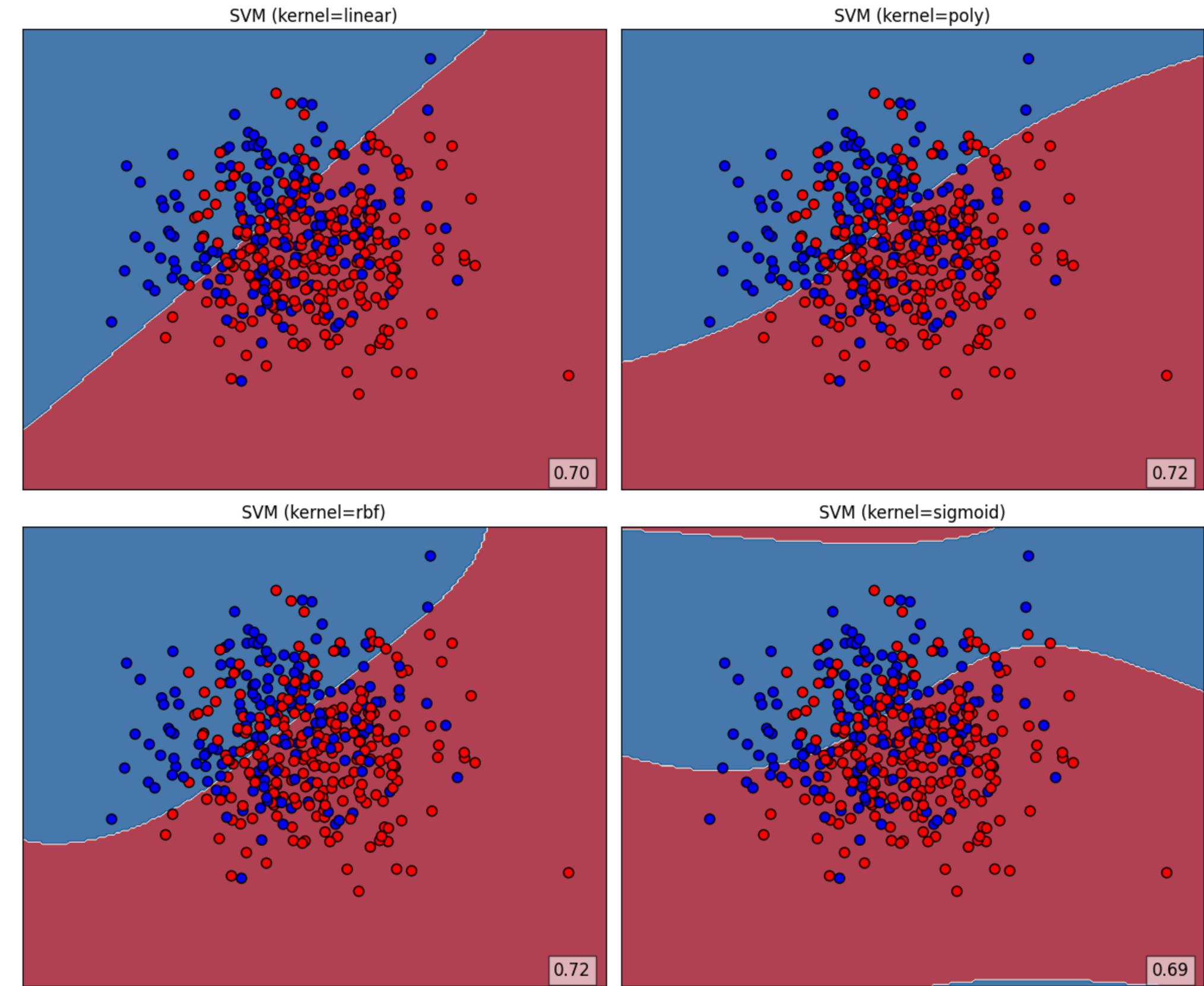
Ứng Dụng Trong Bài Toán Phân Loại

- Naive Bayes: Hai phân phối Gaussian và Bernoulli



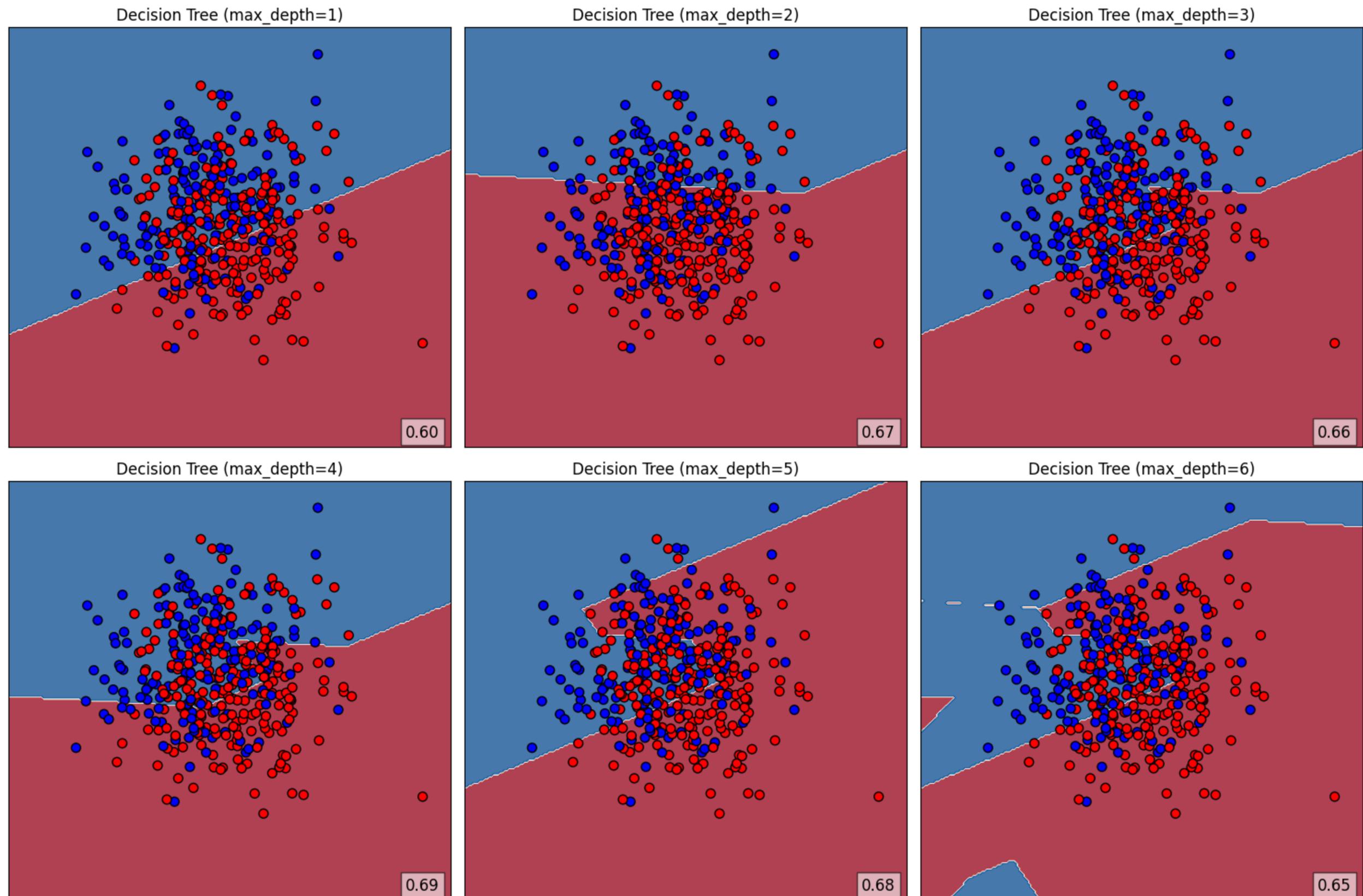
Ứng Dụng Trong Bài Toán Phân Loại

- SVM: 4 hàm kernel
 - linear
 - poly
 - rbf
 - sigmoid



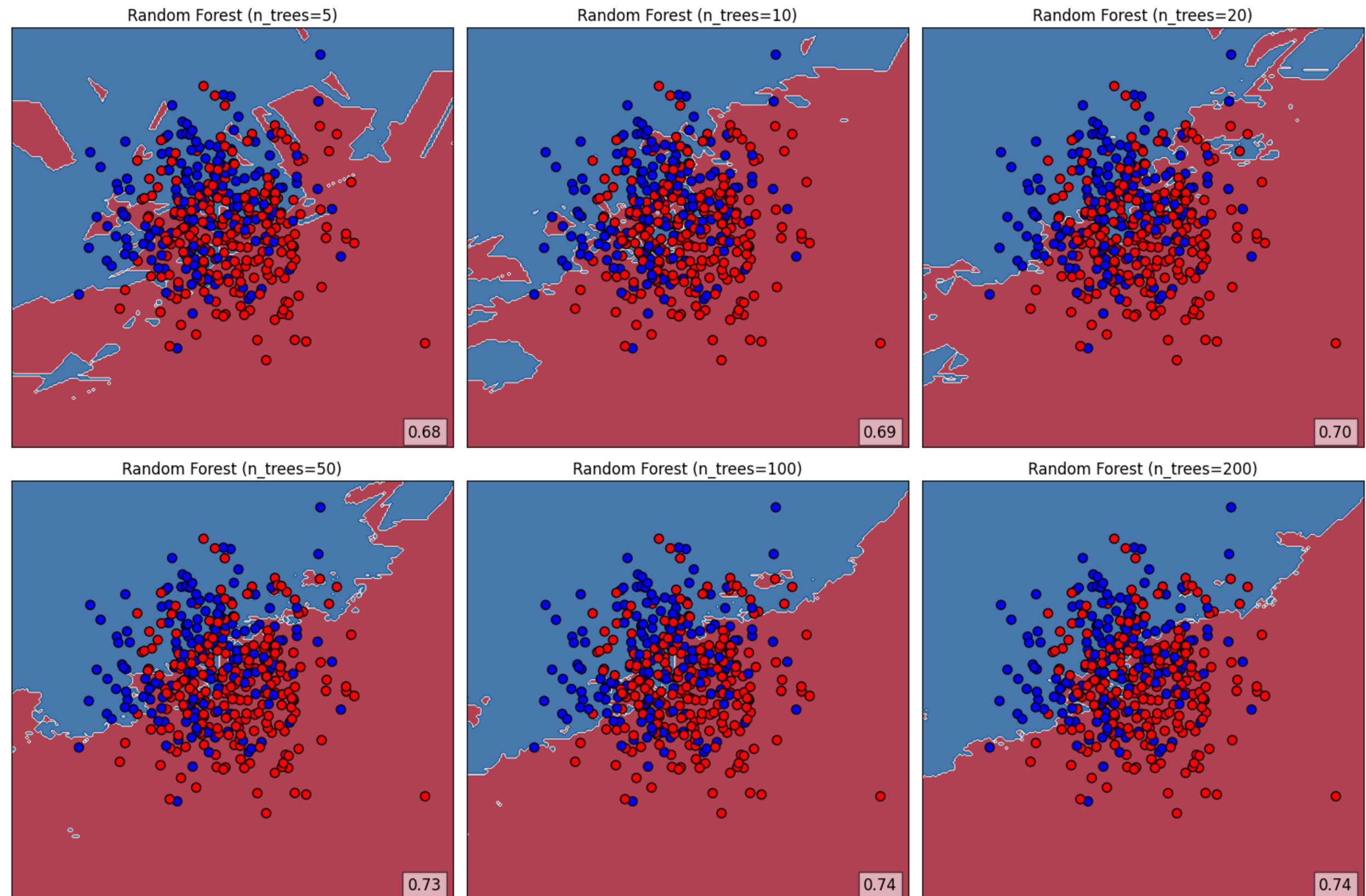
Ứng Dụng Trong Bài Toán Phân Loại

- Decision Tree:



Ứng Dụng Trong Bài Toán Phân Loại

- Random Forest:



Ứng Dụng Trong Bài Toán Phân Loại

- Tổng kết:
 - KNN ($K = 10$): 69%
 - Gaussian Naive Bayes: 73%
 - SVM (kernel = rbf): 72%
 - Decision Tree: 69%
 - Random Forest ($n_{tree} = 100$): 74%





HUST

THANK YOU !