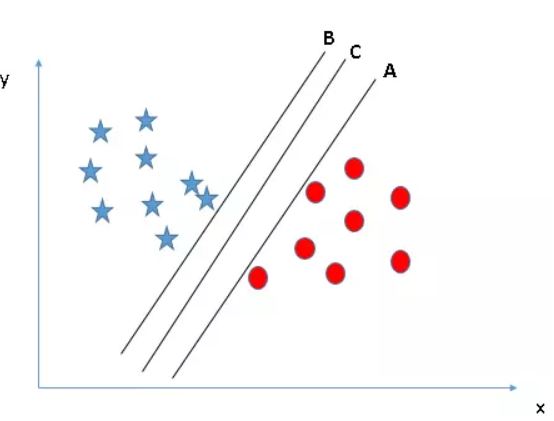
**I Thuật toán SVM**

* K/n
* SVM thuật toán có giám sát
* Hoạt động dựa trên nguyên lý tìm kiếm một *hyperplane (siêu phẳng)* tối ưu để phân chia các điểm dữ liệu thuộc các lớp khác nhau trong không gian
* Hyperplane này được chọn sao cho khoảng cách từ các điểm dữ liệu gần nhất của hai lớp đến hyperplane là lớn nhất
* Các điểm dữ liệu này được gọi là "support vectors".
* Các thành phần chính

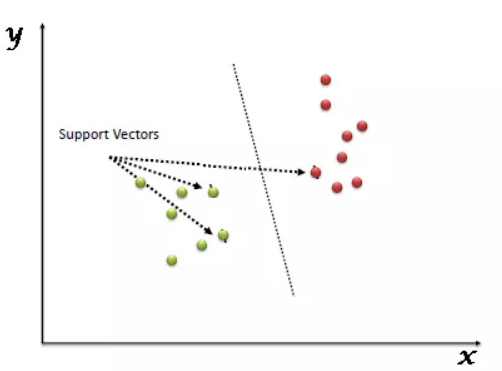
-) Hyperplane: Trong không gian 2 chiều, hyperplane là một đường thẳng; trong không gian 3 chiều, nó là một mặt phẳng.

Quy tắc:



+Xác định khoảng cách lớn nhất từ điểu gần nhất của một lớp nào đó đến đường hyper-plane. Khoảng cách này được gọi là "Margin"

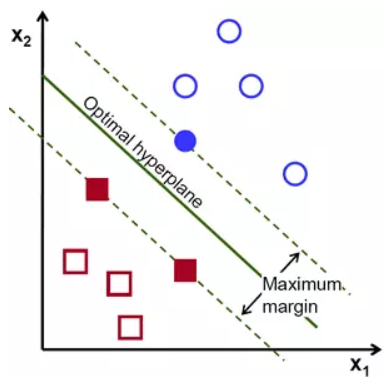
-) Support Vector: Là các điểm dữ liệu nằm gần hyperplane nhất



-) Margin trong SVM

+Là khoảng cách giữa siêu phẳng đến 2 điểm dữ liệu gần nhất tương ứng với các phân lớp

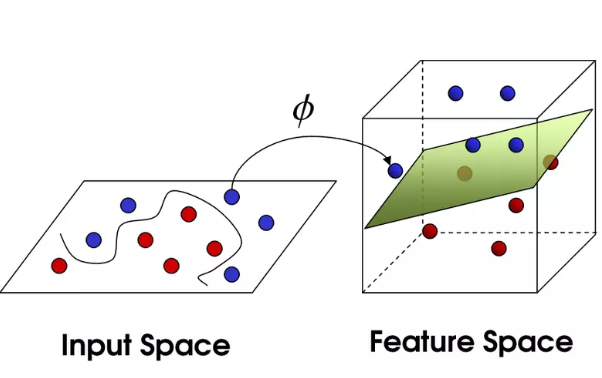
+Phương pháp SVM luôn cố gắng cực đại hóa margin này, từ đó thu được một siêu phẳng tạo khoảng cách xa nhất



* K/luận:

-SVM là một phương pháp hiệu quả cho bài toán phân lớp dữ liệu

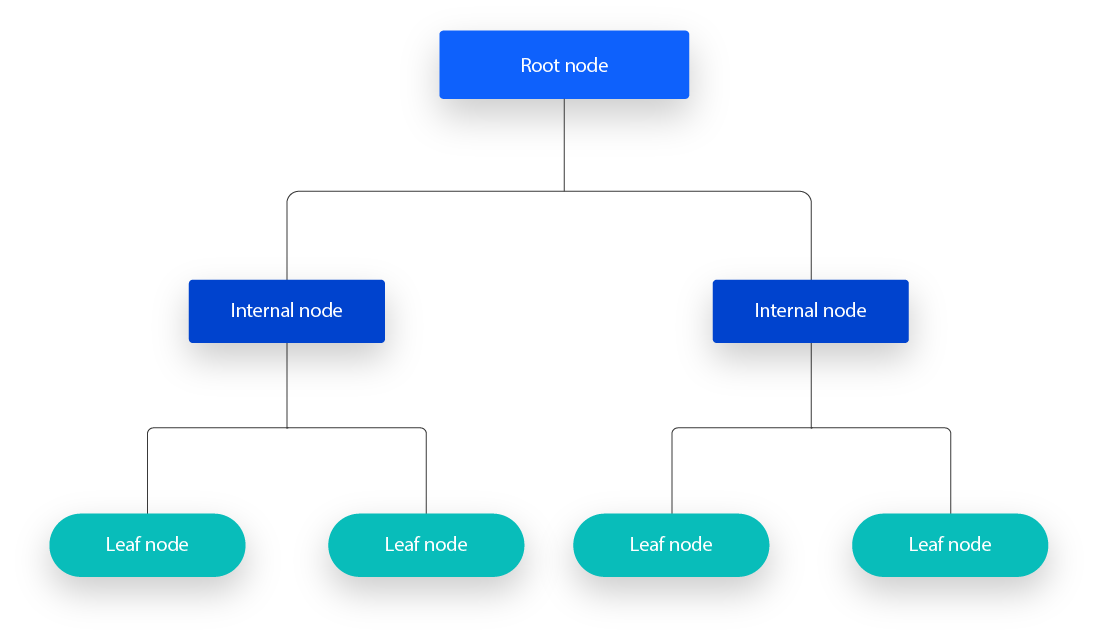
-Một yếu tố làm nên hiệu quả của SVM đó là việc sử dụng ***Kernel function*** khiến cho các phương pháp chuyển không gian trở nên linh hoạt hơn.



**II Thuật toán Decion tree**

* Là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật.
* Các thuộc tính của đối tượngncó thể thuộc các kiểu dữ liệu khác nhau như
* Nhị phân (Binary) ,
* Định danh (Nominal),
* Thứ tự (Ordinal),
* Số lượng (Quantitative)
* Trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal.

Mô hình



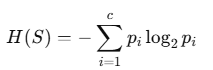
* **Root Node**: Đây là nút gốc của cây, nơi đầu tiên bắt đầu việc phân chia dữ liệu dựa trên một thuộc tính.
* **Internal Node**: Các nút nội bộ, nơi dữ liệu tiếp tục được phân chia dựa trên các thuộc tính khác.
* **Leaf Node**: Các nút lá là các nút cuối cùng, nơi kết quả được đưa ra. Đối với bài toán phân loại, các nút lá thường đại diện cho các nhãn phân loại, còn đối với bài toán hồi quy, các nút lá chứa giá trị dự đoán.
* Thuật toán:

**-ID3:**

-ID3 sử dụng phương pháp tham lam tìm kiếm từ trên xuống thông qua không gian của các nhánh có thể không có backtracking.

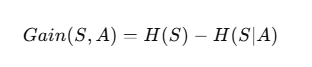
- ID3 sử dụng Entropy và Information Gain để xây dựng một cây quyết định.

+)Entropy: Để đo độ không chắc chắn trong một tập hợp dữ liệu



* c là số lượng lớp trong tập hợp.
* pi​ là xác suất của lớp thứ iii trong tập hợp SSS.

+)Gain: Để đo sự giảm độ không chắc chắn sau khi chia tập dữ liệu.



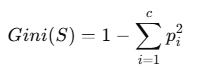
* H(S) là Entropy của tập hợp ban đầu
* H(S∣A) là Entropy của tập hợp sau khi chia theo thuộc tính A.
* Nguyên lý hoạt động
* ID3 sử dụng độ đo thông tin để chọn thuộc tính phân chia tốt nhất tại mỗi nút. Độ đo này thường được tính bằng **Entropy** (độ hỗn loạn) và **Information Gain** (tăng thông tin).
* Tại mỗi nút, thuật toán sẽ tính toán thông tin gain cho tất cả các thuộc tính. Thuộc tính có thông tin gain cao nhất sẽ được chọn làm thuộc tính phân chia cho nút đó.
* Dựa trên thuộc tính phân chia đã chọn, thuật toán sẽ tạo ra các nhánh cho các giá trị của thuộc tính này.
* Các bước trên sẽ được lặp lại cho các nhánh con cho đến khi đạt một điều kiện dừng, chẳng hạn như không còn thuộc tính nào để phân chia, hoặc tất cả các mẫu trong một nhánh đều thuộc cùng một lớp.

**-CART:** Khá giống ID3

* **Cấu Trúc Cây**: Cây CART được xây dựng theo dạng cây nhị phân, nơi mỗi nút trong cây là một câu hỏi (hay điều kiện) về một thuộc tính của dữ liệu, và các nhánh đại diện cho các kết quả của câu hỏi đó.
* **Chia Tách**: Tại mỗi nút, thuật toán tìm cách chia tập dữ liệu thành các nhánh con sao cho giảm thiểu độ không chắc chắn (Entropy) hoặc tối ưu hóa độ chính xác của dự đoán.

-entropy: tương tự id3

-gini index



* Gini(S): Gini index của tập dữ liệu S
* pi​: Xác suất của lớp i

->Thuật toán CART rất phổ biến trong phân tích dữ liệu và học máy do tính đơn giản và hiệu quả của nó. Nó không chỉ giúp tạo ra mô hình dự đoán mà còn dễ dàng giải thích nhờ vào cấu trúc cây của nó.