**10 thuật toán machine learning**

**1. Linear Regression – Hồi quy tuyến tính**

Bài toán này đôi khi được gọi là Linear Fitting (trong thống kê) hoặc Linear Least Square. Hồi quy tuyến tính có lẽ là một trong những thuật toán nổi tiếng nhất và hay được sử dụng nhất trong thống kê và machine learning.

Mô hình dự báo (Predictive model) chủ yếu quan tâm đến việc giảm thiểu sai sót của một mô hình hoặc đưa ra các dự đoán chính xác nhất có thể. Chúng ta sẽ mượn, sử dụng lại và lấy các thuật toán từ nhiều lĩnh vực khác nhau, bao gồm số liệu thống kê và sử dụng chúng cho những mục đích này.

Biểu diễn hồi quy tuyến tính là một phương trình mô tả một đường thẳng biểu diễn một cách phù hợp nhất mối quan hệ giữa các biến đầu vào (x) và các biến đầu ra ( y ), bằng cách tìm các trọng số cụ thể cho các biến đầu vào được gọi là các hệ số (B).

Ví dụ: y = B0 + B1 \* x

Chúng ta sẽ dự đoán y với biến x cho trước và mục tiêu của thuật toán hồi quy tuyến tính là tìm các giá trị cho các hệ số B0 và B1.

Các kĩ thuật khác nhau có thể được sử dụng để tìm mô hình hồi quy tuyến tính từ dữ liệu, chẳng hạn như một giải pháp đại số tuyến tính cho Ordinary least square và việc tối ưu hóa Gradient descent.

Hồi quy tuyến tính đã được phát minh khoảng hơn 200 năm và đã được nghiên cứu rộng rãi. Một số quy tắc tốt khi sử dụng kĩ thuật này là loại bỏ các biến tương quan (correlated) và để loại bỏ bớt nhiễu từ dữ liệu của bạn, nếu có thể. Đây là một kĩ thuật đơn giản và nhanh chóng, và là thuật toán tốt đầu tiên để thử.

Tuy nhiên, Linear Regression rất nhạy cảm với nhiễu (sensitive to noise). Vì vậy, trước khi thực hiện Linear Regression, các nhiễu (outlier) cần phải được loại bỏ. Bước này được gọi là tiền xử lý (pre-processing).

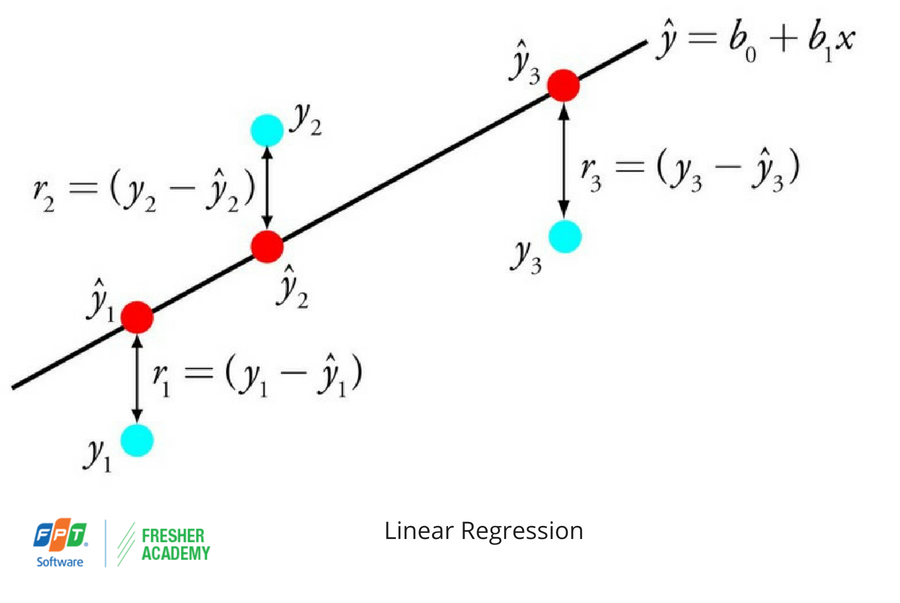
Bạn có thể hiểu thêm về Hồi quy tuyến tính tại:

👉 <https://en.wikipedia.org/wiki/Linear_regression>

[](https://en.wikipedia.org/wiki/Linear_regression)👉 <https://machinelearningmastery.com/simple-linear-regression-tutorial-for-machine-learning/>

[](https://machinelearningmastery.com/simple-linear-regression-tutorial-for-machine-learning/)👉 <http://www.sci.utah.edu/~gerig/CS6640-F2012/Materials/pseudoinverse-cis61009sl10.pdf>

[See Translation](https://www.facebook.com/FPTSoftware.tuyendung/photos/pcb.1636638303132333/1636637829799047/?type=3&theater#)



**2. Logistic Regression – Hồi quy logistic**

Hồi quy logistic là một thuật toán khác mà machine learning mượn từ lĩnh vực thống kê. Đây là phương thức tốt nhất cho các vấn đề phân loại nhị phân (các bài toán với các giá trị 2 lớp).

Hồi quy logistic giống như hồi quy tuyến tính với mục đích là để tìm ra các giá trị cho các hệ số mà trọng số mỗi biến đầu vào. Không giống như hồi quy tuyến tính, dự đoán đầu ra được chuyển đổi bằng cách sử dụng một hàm không tuyến tính (non-linear function) được gọi là hàm logistic.

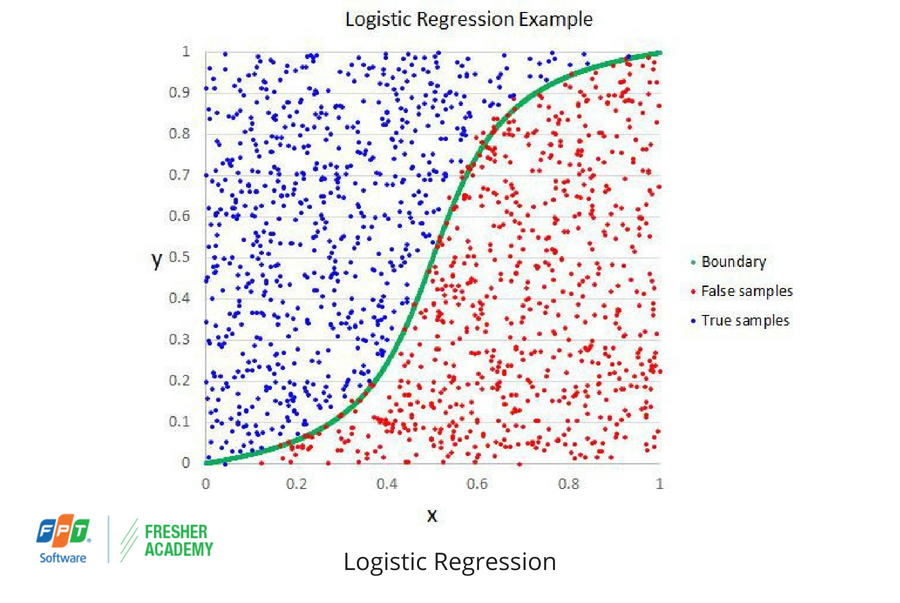
Hàm logistic trông giống như một S lớn và sẽ biến đổi bất kỳ giá trị nào sang khoảng 0-1. Điều này rất hữu ích bởi vì chúng ta có thể áp dụng một quy tắc cho đầu ra của hàm logistic để lấy các giá trị 0 và 1 (ví dụ IF ít hơn 0.5 sau đó đầu ra 1) và dự đoán một lớp giá trị.

Vì cách mô hình được học, các dự đoán thực hiện bởi hồi quy logistic cũng có thể được sử dụng như là xác suất của một ví dụ dữ liệu nhất định thuộc lớp 0 hoặc lớp 1. Điều này có thể hữu ích cho các vấn đề khi bạn cần đưa ra nhiều lý do cho một dự đoán.

Giống như hồi quy tuyến tính, hồi quy logistic hoạt động tốt hơn khi bạn loại bỏ các thuộc tính không liên quan đến biến đầu ra cũng như các thuộc tính tương quan (correlated). Đây là một mô hình có thể học hỏi nhanh và có hiệu quả với các vấn đề phân loại nhị phân.

👉 Tìm hiểu thêm về Logistic Regression: <https://machinelearningcoban.com/2017/01/27/logisticregression/>

[See Translation](https://www.facebook.com/FPTSoftware.tuyendung/photos/pcb.1636638303132333/1636637853132378/?type=3&theater#)



**3. Linear Discriminant Analysis**

Hồi quy Logistic là một thuật toán phân loại truyền thống giới hạn vào các vấn đề phân loại hai lớp. Nếu bạn có nhiều hơn hai lớp thì thuật toán Linear Discriminant Analysis nên được ưu tiên.

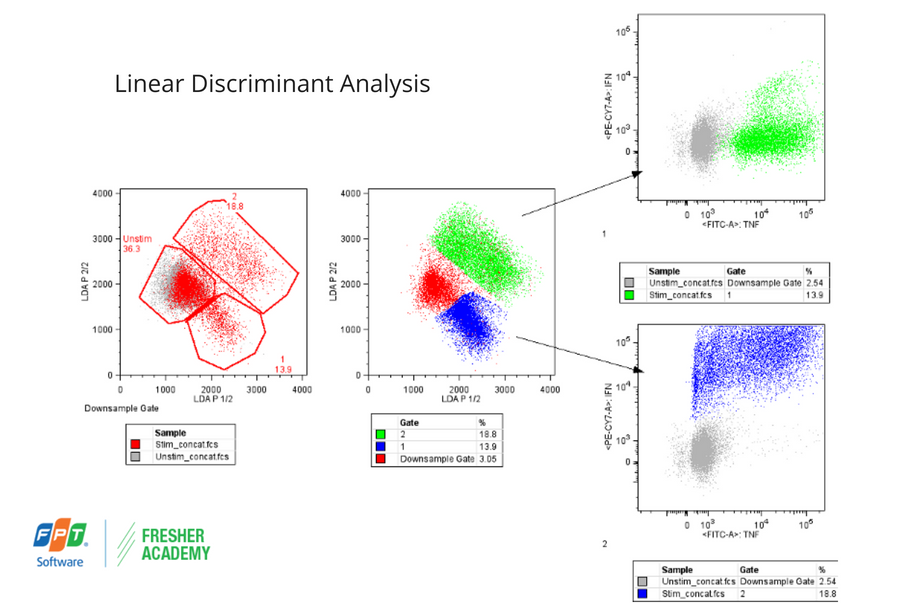
Biểu diễn của LDA khá đơn giản. Nó bao gồm các thuộc tính thống kê của dữ liệu của bạn, được tính cho mỗi lớp. Đối với một biến đầu vào duy nhất, nó bao gồm:

🌳 Giá trị trung bình cho mỗi lớp.

🌳 Phương sai được tính trên tất cả các lớp.

Dự đoán được thực hiện bằng cách tính giá trị phân biệt cho mỗi lớp và dự đoán cho lớp đó có giá trị lớn nhất. Kĩ thuật này giả định rằng dữ liệu có một phân bố Gaussian (bell curve), do đó tốt hơn là bạn nên loại bỏ các giá trị ngoại vi khỏi dữ liệu của bạn trước. Đây là một phương pháp đơn giản và mạnh mẽ để phân loại các vấn đề mô hình dự báo.

👉 Tìm hiểu thêm về LDA: [https://machinelearningcoban.com/2017/06/30/lda/](https://l.facebook.com/l.php?u=https%3A%2F%2Fmachinelearningcoban.com%2F2017%2F06%2F30%2Flda%2F&h=AT3U0EXoMehOi6Aq5bwWUvu1d_QxBeyGodlrn8FFdcr6LTv00Jpuc3gBDJpph1zw3g78cHGmDiEJG6MDdnSigHVlYLy2ukaRdV5JFXxCSz-fokcZQhMcCda2LVrfKtIB-Q)



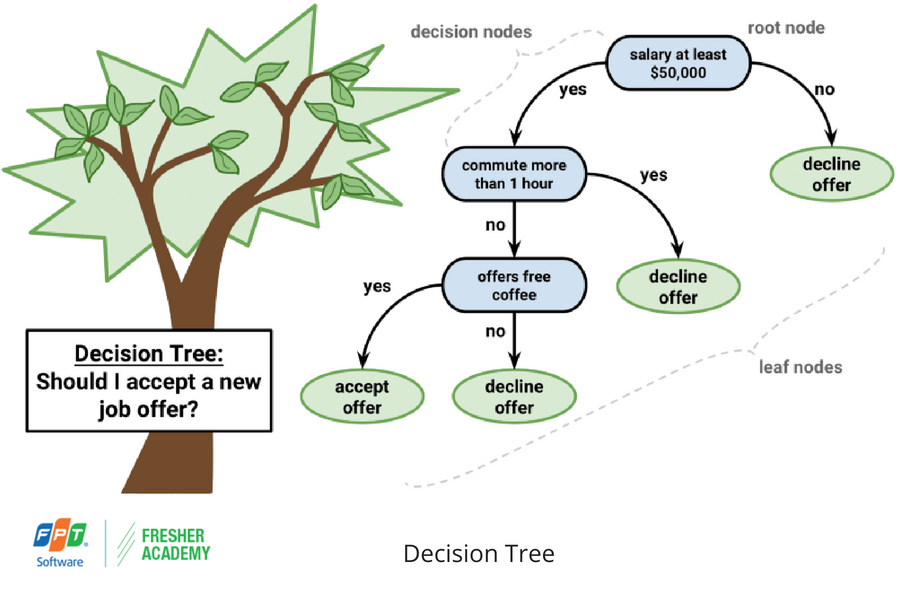
**4. Classification and regression trees**

Decision trees là một loại thuật toán quan trọng cho mô hình tiên đoán machine learning.

Biểu diễn của mô hình decision tree là một cây nhị phân. Mỗi node đại diện cho một biến đầu vào duy nhất (x) và một điểm phân chia trên biến đó (giả sử biến đó là số).

Các node lá của cây chứa một biến đầu ra (y) được sử dụng để dự báo. Các tiên đoán được thực hiện bằng cách đi trên các nhánh của cây cho đến khi đến một node lá và đưa ra lớp giá trị tại node lá đó.

Cây có thể học rất nhanh và có thể dùng để dự đoán rất nhanh. Chúng thường chính xác cho nhiều loại vấn đề và dữ liệu của bạn và không cần bất kỳ sự chuẩn bị đặc biệt nào.



**5. Naive Bayes**

Naive Bayes là một thuật toán đơn giản nhưng mạnh mẽ về mô hình tiên đoán.

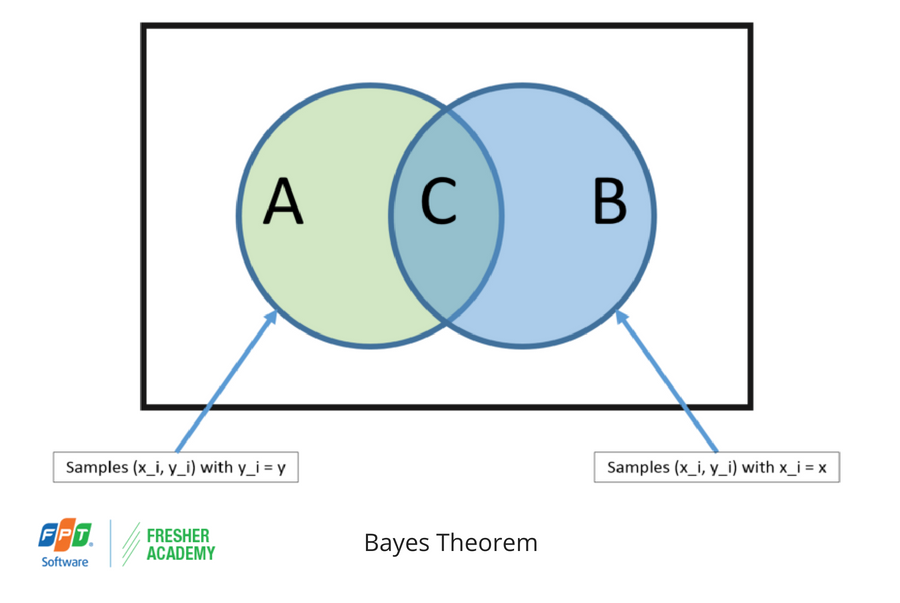
Mô hình bao gồm hai loại xác suất có thể được tính trực tiếp từ dữ liệu của bạn:

1) Xác suất của mỗi lớp;

2) Xác suất có điều kiện cho mỗi lớp với mỗi giá trị x.

Sau khi tính, mô hình xác suất có thể được sử dụng để đưa ra dự đoán cho dữ liệu mới bằng Định lý Bayes. Khi dữ liệu của bạn có giá trị thực, giả sử một phân phối Gaussian (bell curve) khá phổ biến nên bạn có thể dễ dàng ước tính được các xác suất này.

Naive Bayes được coi là naive vì nó giả định rằng mỗi biến đầu vào là độc lập. Đây là một giả định mạnh mẽ nhưng không thực tế đối với dữ liệu thực, tuy nhiên, kĩ thuật này rất hiệu quả trên một phạm vi rộng lớn với các vấn đề phức tạp.



**6. K – Nearest Neighbors – KNN**

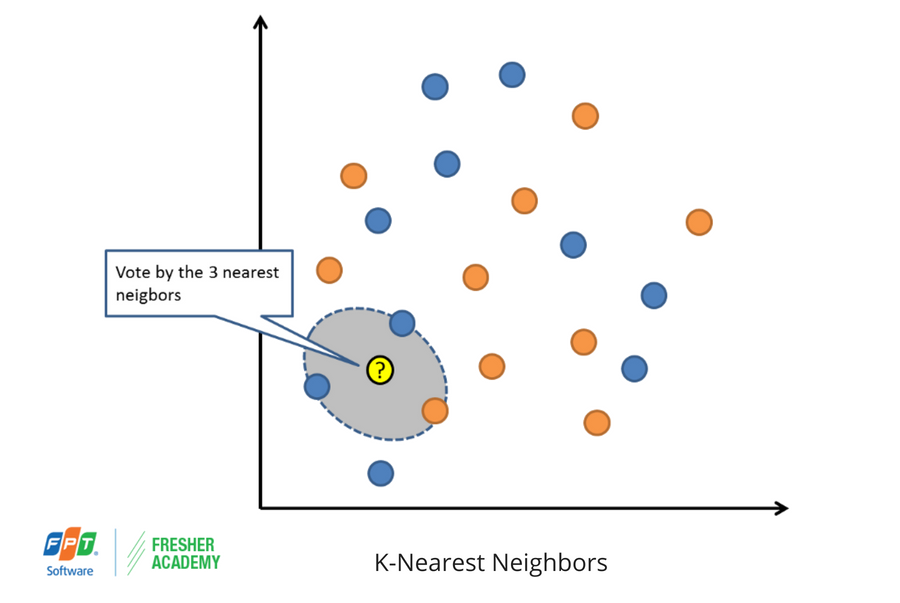
Thuật toán KNN rất đơn giản và rất hiệu quả. Mô hình biểu diễn cho KNN là toàn bộ dữ liệu training.

Dự đoán được thực hiện cho một điểm dữ liệu mới bằng cách tìm kiếm toàn training set cho hầu hết các ví dụ K giống nhau nhất (hàng xóm) và tóm tắt biến đầu ra cho các ví dụ K này. Đối với các vấn đề hồi quy, đây có thể là biến đầu ra trung bình, đối với các vấn đề phân loại, đây có thể là mode (hoặc phổ biến nhất) của lớp.

Bí quyết là làm thế nào để xác định sự giống nhau giữa các trường hợp dữ liệu. Kĩ thuật đơn giản nhất nếu các thuộc tính của bạn có cùng kích cỡ (ví dụ tất cả đều là inch) là sử dụng khoảng cách Euclide, một con số bạn có thể tính toán trực tiếp dựa trên sự khác biệt giữa mỗi biến đầu vào.

KNN có thể yêu cầu rất nhiều bộ nhớ hoặc không gian để lưu trữ tất cả dữ liệu, nhưng chỉ thực hiện tính toán (hoặc học) khi một dự báo là cần thiết, chỉ vừa đúng lúc. Bạn cũng có thể cập nhật và tổ chức các bài tập đào tạo theo thời gian để giữ các dự đoán chính xác.

Ý tưởng về khoảng cách hoặc độ gần có thể bị phá vỡ với các quy mô rất cao (rất nhiều biến đầu vào) có thể ảnh hưởng tiêu cực đến hiệu suất của thuật toán đối với vấn đề của bạn. Đây được gọi là lời nguyền của quy mô. Nó cho thấy bạn chỉ sử dụng những biến đầu vào có liên quan nhất đến dự đoán biến đầu ra.



**7. Learning Vector Quantization**

Nhược điểm của K- Nearest Neighbors là bạn cần phải giữ nguyên bộ dữ liệu đào tạo của mình.

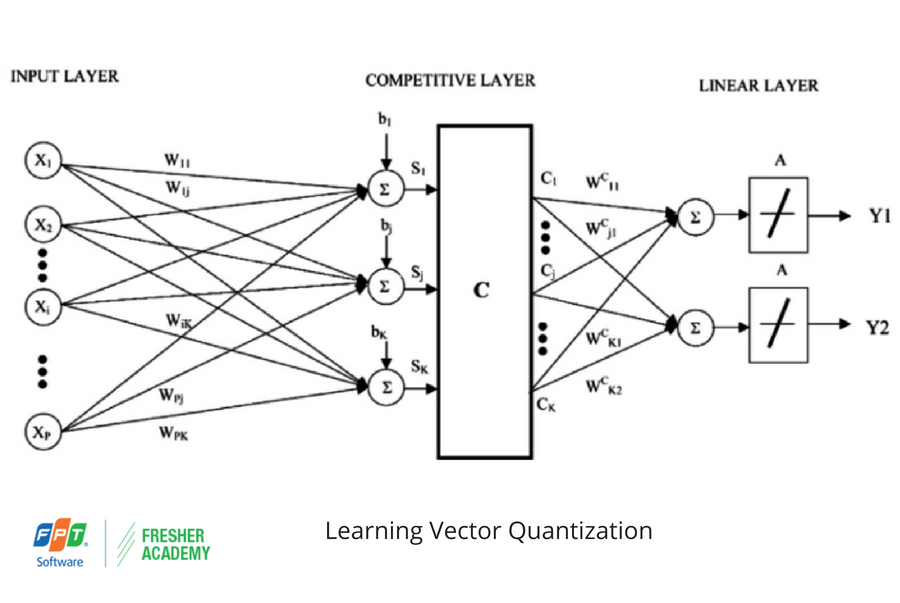
Thuật toán Quantization về Vector (hay LVQ) là một thuật toán mạng thần kinh nhân tạo cho phép bạn chọn có bao nhiêu trường hợp đào tạo để treo lên và tìm hiểu chính xác những trường hợp này sẽ như thế nào.

Sự biểu diễn cho LVQ là một tập hợp các codebook vector. Chúng được lựa chọn ngẫu nhiên từ đầu và thích nghi để tóm tắt tốt nhất tập dữ liệu đào tạo qua một số lần lặp của thuật toán.

Sau khi học được, các vector mã có thể được sử dụng để tạo ra các dự đoán giống như K- Nearest Neighbors. Các hàng xóm tương tự nhất ( phù hợp với codebook vector nhất ) được tìm thấy bằng cách tính toán khoảng cách giữa mỗi vector và ví dụ dữ liệu mới.

Lớp giá trị hoặc (giá trị thực trong trường hợp hồi quy) cho đơn vị kết hợp tốt nhất sau đó sẽ được trả về như dự đoán. Kết quả tốt nhất đạt được khi bạn thay đổi dữ liệu của mình để có cùng phạm vi, chẳng hạn như giữa 0 và 1.

Nếu bạn phát hiện ra rằng KNN mang lại kết quả tốt trên tập dữ liệu của bạn, hãy thử sử dụng LVQ để giảm yêu cầu về bộ nhớ để lưu trữ toàn bộ tập dữ liệu đào tạo.



**8. Support Vector machines**

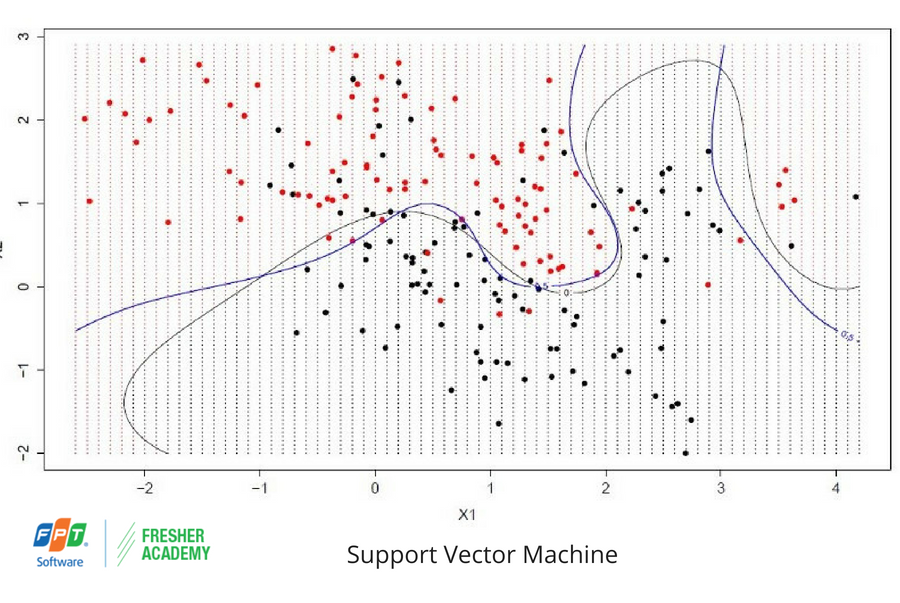
Support vector machines có lẽ là một trong những thuật toán machine learning phổ biến nhất và được nói về nhiều nhất.

Một hyperplane là một đường phân chia không gian biến đầu vào. Trong SVM, một hyperplance được chọn để phân tách tốt nhất các điểm trong không gian các biến đầu vào theo lớp của chúng, hoặc là lớp 0 hoặc lớp 1.

Trong hai chiều, bạn có thể hình dung nó như một đường thẳng và giả sử rằng tất cả các biến đầu vào của chúng ta có thể được tách hoàn toàn bằng dòng này. Thuật toán SVM tìm ra các hệ số dẫn đến sự phân tách tốt nhất của các lớp theo hyperplance.

Khoảng cách giữa hyperplane và điểm dữ liệu gần nhất được gọi là biên. Hyperplane tốt nhất hoặc tối ưu có thể tách riêng hai lớp là dòng có biên lớn nhất. Chỉ những điểm này có liên quan đến việc xác định hyperplane và trong việc xây dựng các điểm phân loại. Những điểm này được gọi là các vector hỗ trợ. Chúng hỗ trợ hoặc xác định hyperplane. Trong thực tế, một thuật toán tối ưu được sử dụng để tìm các giá trị cho các hệ số tối đa hóa biên.

SVM có thể là một trong những phương pháp phân loại hàng đầu mạnh mẽ nhất và đáng thử trên tập dữ liệu của bạn.

****

**9. Bagging and Random Forest**

Random Forest là một trong những thuật toán machine learning phổ biến nhất và mạnh nhất. Nó là một loại thuật toán machine learning được gọi là Bootstrap Aggregation hoặc Bagging.

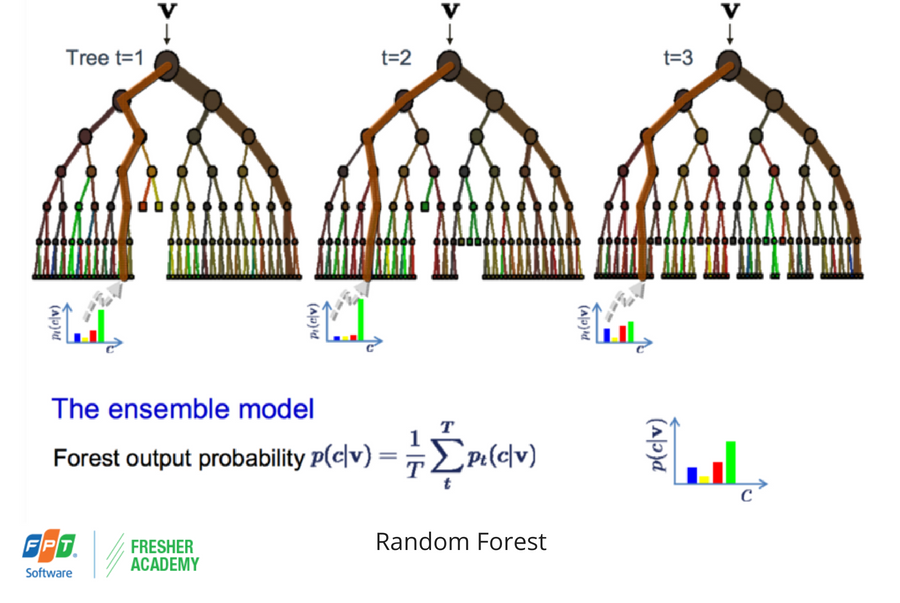
Bootstrap là một phương pháp thống kê mạnh mẽ để ước lượng số lượng từ một mẫu dữ liệu. Chẳng hạn như một giá trị trung bình. Bạn lấy rất nhiều mẫu dữ liệu của bạn, tính giá trị trung bình, sau đó trung bình tất cả các giá trị trung bình của bạn để bạn ước lượng tốt hơn giá trị trung bình thật sự.

Trong bagging, cách tiếp cận tương tự được sử dụng, nhưng thay vì để ước lượng toàn bộ mô hình thống kê, thường là decision trees. Nhiều mẫu dữ liệu đào tạo của bạn được lấy sau đó các mô hình được xây dựng cho mỗi mẫu dữ liệu. Khi bạn cần dự đoán dữ liệu mới, mỗi mô hình sẽ dự đoán và các dự đoán được tính trung bình để ước lượng tốt hơn giá trị đầu ra thật sự.

Random forest là một sự tinh chỉnh trên phương pháp tiếp cận này, nơi các decision trees được tạo ra để thay vì chọn các điểm phân chia tối ưu, việc phân chia tối ưu được thực hiện bằng cách đưa ra sự ngẫu nhiên.

Các mô hình được tạo cho mỗi mẫu dữ liệu khác biệt nhiều so với chúng lẽ ra phải như thế, tuy nhiên vẫn chính xác theo những cách độc đáo và khác biệt. Kết hợp dự đoán của chúng dẫn đến một ước tính tốt hơn về giá trị đầu ra cơ bản.

Nếu bạn nhận được kết quả tốt với một thuật toán có độ biến thiên cao (như decision trees), bạn thường có thể nhận được kết quả tốt hơn bằng cách bagging thuật toán đó.

****

**10. Boosting và AdaBoost**

Boosting là một kĩ thuật đồng bộ nhằm cố gắng tạo ra một phương pháp phân loại mạnh từ một số phương pháp phân loại yếu. Điều này được thực hiện bằng cách xây dựng mô hình từ dữ liệu đào tạo, sau đó tạo ra một mô hình thứ hai cố gắng sửa lỗi từ mô hình đầu tiên. Các mô hình được thêm vào cho đến khi tập đào tạo được dự đoán hoàn hảo hoặc thêm một số mô hình tối đa.

AdaBoost là thuật toán boosting thành công đầu tiên được phát triển để phân loại nhị phân. Đây là điểm khởi đầu tốt nhất để hiểu về boosting. Các phương pháp boosting hiện đại xây dựng trên AdaBoost, đáng chú ý nhất là các máy boosting gradient ngẫu nhiên.

AdaBoost được sử dụng với các decision trees ngắn. Sau khi cây đầu tiên được tạo ra, hiệu suất của cây trên mỗi trường hợp huấn luyện được sử dụng để đo độ chú ý của cây kế tiếp được tạo nên chú ý đến từng trường hợp đào tạo. Dữ liệu đào tạo khó dự đoán sẽ có trọng lượng hơn, trong khi những trường hợp dễ dự đoán có ít trọng lượng hơn. Các mô hình được tạo theo thứ tự tuần tự, mỗi lần cập nhật các trọng số trên các trường hợp đào tạo ảnh hưởng đến việc học được thực hiện bởi cây kế tiếp trong chuỗi. Sau khi tất cả các cây được xây dựng, dự đoán được thực hiện cho dữ liệu mới, và hiệu suất của mỗi cây được đo lường bằng cách nó đã được về dữ liệu đào tạo.

Bởi vì rất nhiều sự chú ý được đưa ra để sửa sai lầm theo thuật toán, điều quan trọng là bạn phải có dữ liệu sạch với các giá trị biên.

