**Random Forest**

Random forest là thuật toán supervised learning, có thể giải quyết cả bài toán hồi quy và phân loại. Với random forest, chúng ta sẽ xây dựng nhiều cây quyết định bằng decision tree, tuy nhiên mỗi cây quyết định sẽ có những yếu tố ngẫu nhiên và kết quả dự đoán được tổng hợp từ các cây quyết định.

**Xây dựng thuật toán random forest**

Giả sử bộ dữ liệu của chúng ta có n mẫu và mỗi mẫu có d thuộc tính (feature).

Để xây dựng mỗi cây quyết định chúng ta sẽ làm như sau:

1. Lấy ngẫu nhiên n mẫu từ bộ dữ liệu với kĩ thuật [Bootstrapping](https://en.wikipedia.org/wiki/Bootstrapping_(statistics)), hay còn gọi là **random sampling with replacement**. Cụ thể khi lấy ra 1 mẫu dữ liệu, chúng ta không loại bỏ luôn mẫu đó khỏi không gian mẫu mà tiếp tục lặp lại cho tới khi có đủ n mẫu dữ liệu. Khi dùng kĩ thuật này thì tập n dữ liệu mới được tạo ra có thể có những dữ liệu bị trùng nhau.
2. Sau khi sample được n dữ liệu từ bước 1, chúng ta sẽ chọn ngẫu nhiên ở k thuộc tính (k < n). Chúng ta thu được bộ dữ liệu mới gồm n dữ liệu và mỗi dữ liệu có k thuộc tính.
3. Dùng thuật toán Decision Tree để xây dựng cây quyết định với bộ dữ liệu ở bước 2.

Do tính chất ngẫu nhiên (random) trong quá trình lựa chọn mẫu dữ liệu nên kết quả là các cây quyết định trong thuật toán Random Forest có thể khác nhau.

Thuật toán Random Forest là tập hợp gồm nhiều cây quyết định, mỗi cây được xây dựng dùng thuật toán Decision Tree trên tập dữ liệu khác nhau và dùng tập thuộc tính khác nhau. Sau đó kết quả dự đoán của thuật toán Random Forest sẽ được tổng hợp từ các cây quyết định.

Khi dùng thuật toán Random Forest, chúng ta cần chú ý tới một số thuộc tính như: số lượng cây quyết định sẽ xây dựng, số lượng thuộc tính dùng để xây dựng cây. Ngoài ra, vẫn có các thuộc tính của thuật toán Decision Tree để xây dựng cây như độ sâu tối đa, số phần tử tối thiểu trong 1 node để có thể tách.

**Tại sao thuật toán Random Forest có thể tốt hơn so với một mô hình Decision tree thông thường**

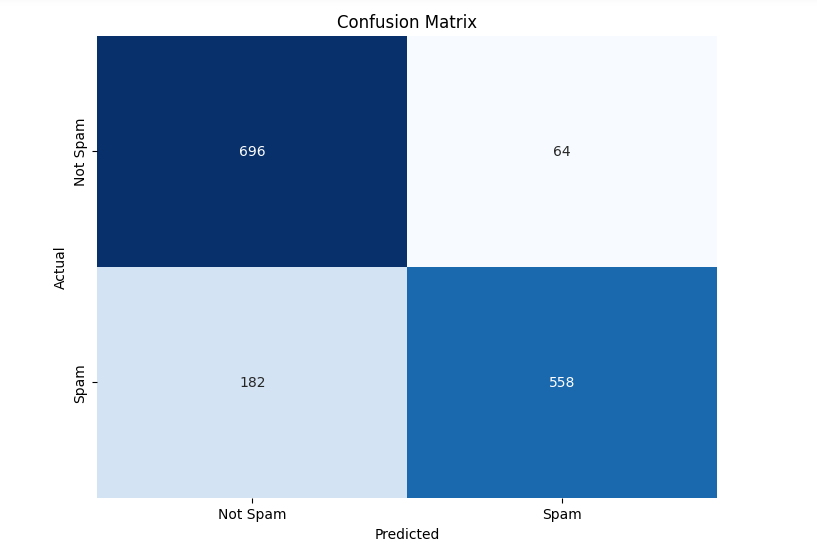
Trong thuật toán Decision Tree, khi xây dựng cây quyết định nếu để độ sâu tùy ý thì cây sẽ phân loại đúng hết các dữ liệu trong tập training, tuy nhiên lại không chắc thể hiện đúng trên tập test (high variance). Chính vì sự phức tạp trong mô hình có thể dẫn đến overfitting. Tuy nhiên, đối với thuật toán Random Forest gồm nhiều cây quyết định, mỗi cây quyết định đều có những yếu tố ngẫu nhiên:

1. Lấy ngẫu nhiên dữ liệu để xây dựng cây quyết định.
2. Lấy ngẫu nhiên các thuộc tính để xây dựng cây quyết định.

Do mỗi cây quyết định trong thuật toán Random Forest không dùng tất cả dữ liệu training, cũng như không dùng tất cả các thuộc tính của dữ liệu để xây dựng cây nên mỗi cây có thể sẽ dự đoán không tốt, khi đó mỗi mô hình cây quyết định không bị overfitting mà có thế bị underfitting, hay nói cách khác là mô hình có high bias. Tuy nhiên, kết quả cuối cùng của thuật toán Random Forest lại tổng hợp từ nhiều cây quyết định, thế nên thông tin từ các cây sẽ bổ sung thông tin cho nhau, dẫn đến mô hình có low bias và low variance, hay mô hình có kết quả dự đoán tốt.

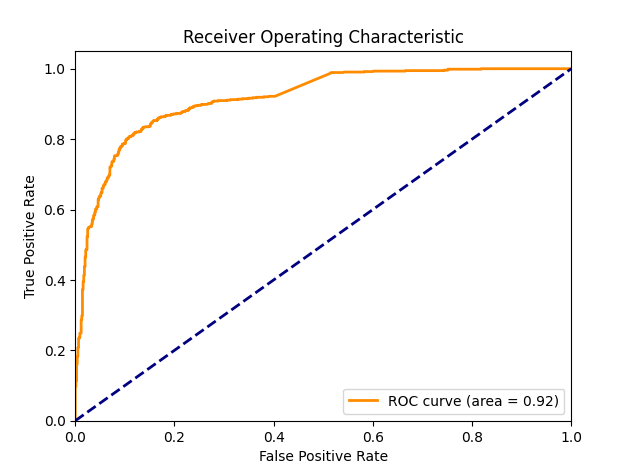
**Kết quả**

Với việc áp dụng random forest cho bài toán phân loại comment spam, chúng ta tiến hành sử dụng GridSearchCV và cross validation để đưa ra các tham số và model tốt nhất trong grid search. Kết quả thu được trên tập test với accuracy 83.6%.



Tỉ lệ dự đoán đúng chỉ đạt 83.6%, mô hình đã bỏ qua khá nhiều comment spam.

Đối với giá trị AUC = 92, mô hình có khả năng phân biệt khá tốt giữa 2 lớp.



**SVM**

**Giới thiệu về SVM**

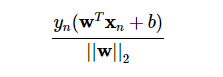
SVM là thuật toán học có giám sát, thường được dùng cho bài toán phân loại. SVM có khả năng phân loại dữ liệu có số chiều lớn. Ý tưởng của thuật toán là tìm một siêu phẳng để phân chia các điểm dữ liệu của các lớp khác nhau trong không gian.

Để tìm được siêu phẳng, chúng ta cần giải quyết bài toán tối ưu khoảng cách (margin) giữa các điểm dữ liệu của 2 lớp gần nhất. Và các điểm dữ liệu nằm gần siêu phẳng được gọi là các vector hỗ trợ.

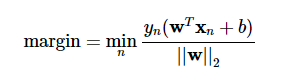
Đối với dữ liệu có phân bố tuyến tính, chúng ta có thể sử dụng mô hình linear SVM. Đối với dữ liệu có phân bố phi tuyến tính, chúng ta có thể sử dụng các mô hình SVM có kernel phi tuyến tính.

**Xây dựng bài toán tối ưu**

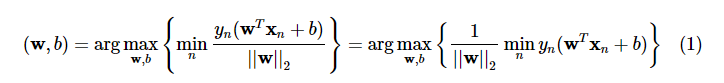
Giả sử các cặp dữ liệu của training set là (x1, y1), (x2, y2)…(xn, yn) với xi nằm trong không gian d chiều thể hiện đầu vào của dữ liệu và yi là nhãn của điểm dữ liệu đó. D là số chiều của dữ liệu và N là số điểm dữ liệu. Giả sử các nhãn yi được xác định bới yi = 1 hoặc yi = -1. Với một điểm quan sát bất kì, khoảng cách từ điểm đó tới siêu phẳng là:



Như vậy, margin được tính là khoảng cách gần nhất từ 1 điểm tới mặt phẳng đó.

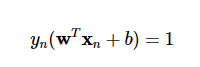


Bài toán tối ưu trong SVM là bài toán tìm **w** và **b** sao cho margin này đạt giá trị lớn nhất:



Quá trình giải bài toán này rất phức tạp, chính vì thế chúng ta sẽ đưa nó về bài toán đơn giản hơn.

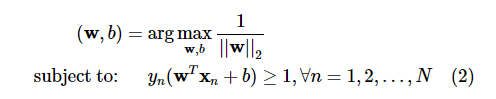
Nếu ta thay vector hệ số **w** bằng **kw**, **b** thành **kb** trong đó **k** là hằng số dương thì mặt phân chia không đổi, khoảng cách từ từng điểm đến mặt phân chia không đổi, tức margin không đổi. Dựa vào tính chất này, ta giả sử:



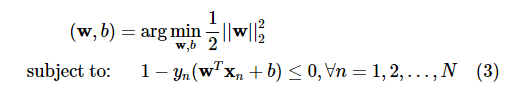
với những điểm nằm gần mặt phân chia nhất. Tổng quát với mọi n, ta có:



Ta có thể đưa bài toán tối ưu về bài toán tối ưu có ràng buộc:

****

Bằng vài phép biến đổi (nghịch đảo hàm mục tiêu, bình phương để có hàm khả vi, nhân với ½ để dễ đạo hàm hơn), ta có thể đưa về bài toán dưới đây:



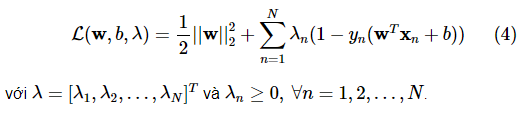
Xác định class cho điểm dữ liệu sau khi tìm được mặt phân cách **wTx + b = 0**:



**Bài toán đối ngẫu SVM**

Bài toán tối ưu trên thỏa mãn điều kiện Slater, như vậy bài toán trên thỏa mãn strong duality.

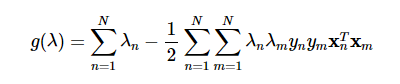
Ta sử dụng Lagrange cho bài toán (3):



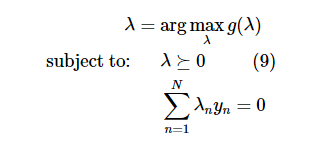
Hàm đối ngẫu Lagrange được định nghĩa như sau:



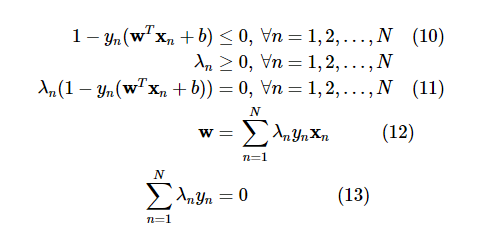
Ta có thể giải hàm trên sử dụng đạo hàm, chúng ta thu được:



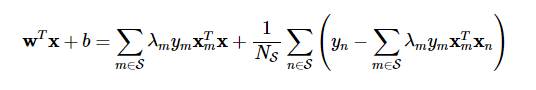
Với việc kết hợp hàm đối ngẫu Lagrange và các điều kiện ràng buộc của lambda, ta thu được bài toán đối ngẫu Lagrange:



Vì đây là một bài toán lồi và strong duality, nghiệm của bài toán sẽ thỏa mãn hệ KKT:



Bằng việc giải hệ tối ưu này, để xác định x thuộc class nào, chúng ta cần xác định dấu của biểu thức:

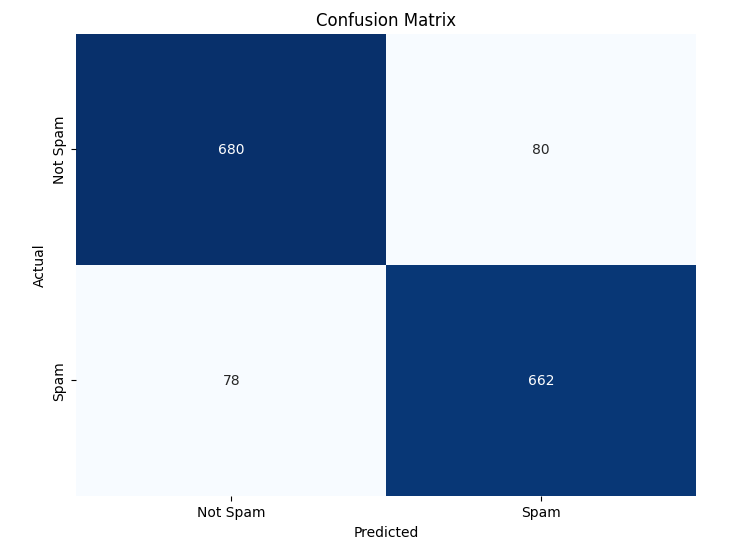


**Áp dụng cho bài toán**

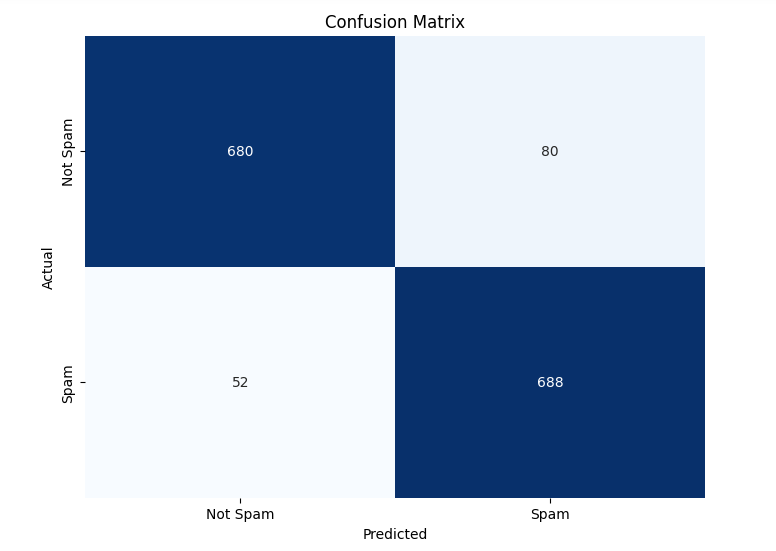
Chúng ta cũng thực hiện GridSearchCV để đưa ra tham số tốt nhất. Vì dữ liệu không có mối quan hệ tuyến tính, nên việc sử dụng các kernel phi tuyến tính sẽ đưa ra kết quả tốt hơn. Tuy nhiên, chúng ta vẫn sẽ thực hiện SVM tuyến tính và đưa ra nhận xét với 2 mô hình:

* Accuracy: SVM tuyến tính đạt 89.47%. SVM phi tuyến đạt 91.2%
* Confusion matrix

Với Linear SVM:

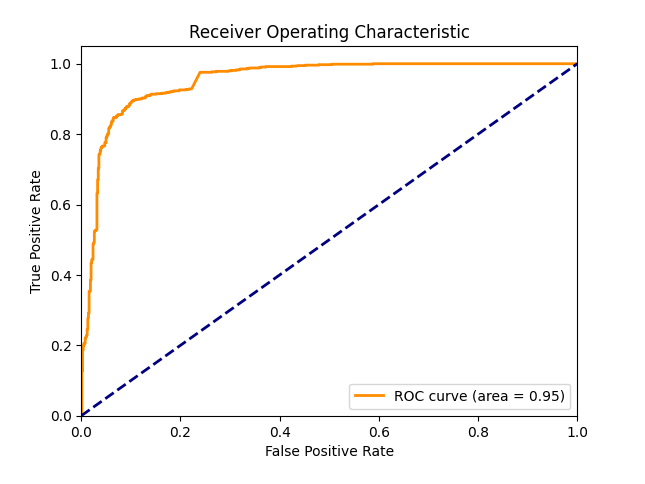


Với Non-linear SVM:



* ROC-AUC:

Với Linear SVM:



Với Non-linear SVM:

