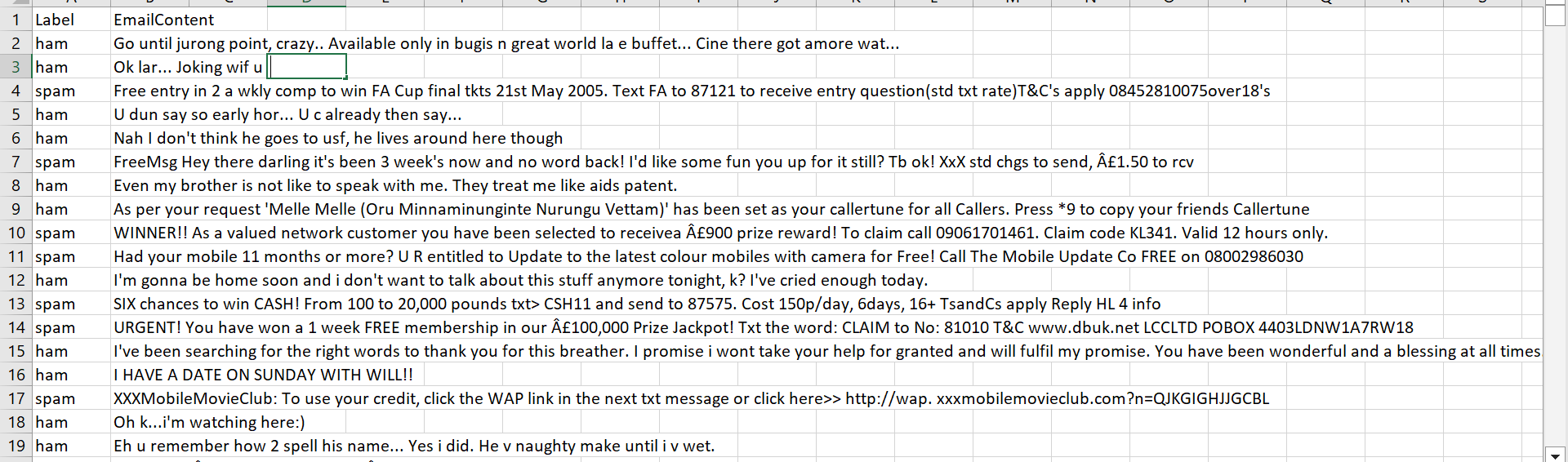
Dữ liệu chuẩn bị:

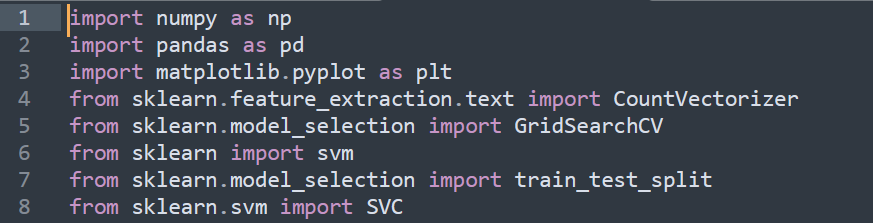


1. Label — Ham or Spam
2. EmailContent — Actual Email

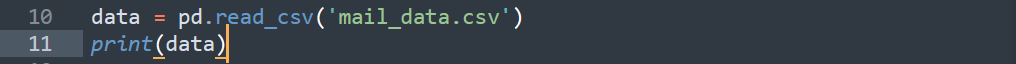
Vì vậy, về cơ bản, mô hình sẽ nhận dạng mẫu và sẽ dự đoán xem thư là thư rác hay thư thật.

Thuật toán được sử dụng — SVM

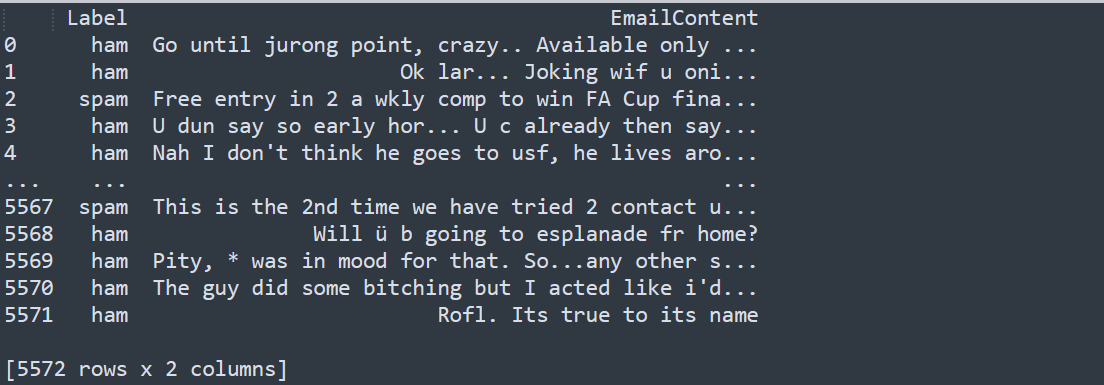
Import các thư viện cần thiết:



Đọc và in ra dữ liệu để kiểm tra



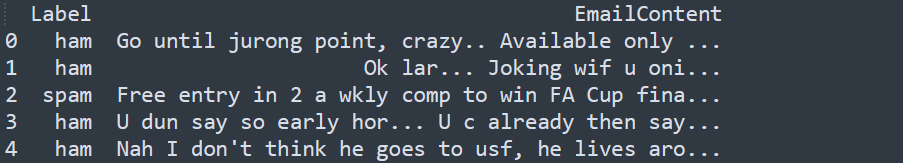
Kết quả:



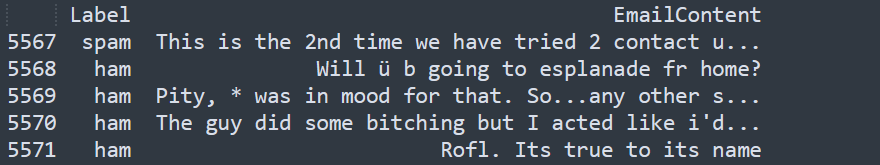
Hiển thị các thông tin của dữ liệu:



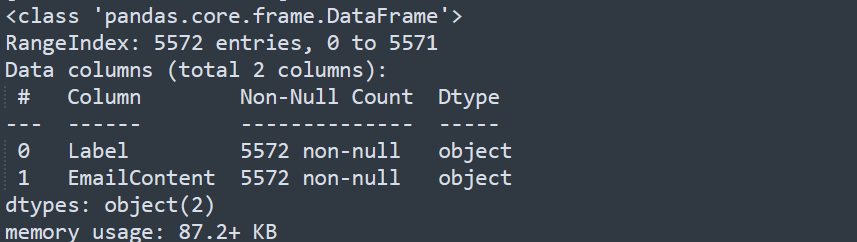
data.head()



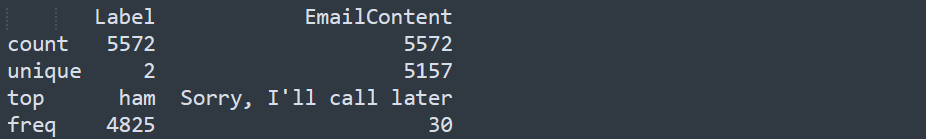
data.tail()



data.info()



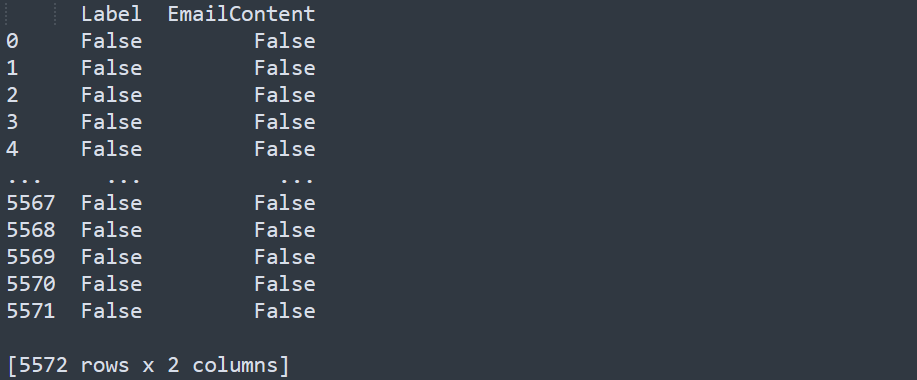
data.describe()



* **`count`** cho biết số lượng dòng không rỗng trong cột. Trong trường hợp này, cả hai cột "Label" và "EmailContent" đều có 5572 dòng không rỗng.
* **`unique`** chỉ ra số lượng giá trị duy nhất trong cột. Ở đây, cột "Label" chỉ có hai giá trị duy nhất ("ham" và "spam"), trong khi cột "EmailContent" có 5157 giá trị duy nhất.
* **`top`** hiển thị giá trị xuất hiện nhiều nhất trong cột. Trong trường hợp này, giá trị "ham" xuất hiện nhiều nhất trong cột "Label", và "Sorry, I'll call later" xuất hiện nhiều nhất trong cột "EmailContent".
* **`freq`** cho biết số lần xuất hiện của giá trị phổ biến nhất (top). Ở đây, giá trị "ham" xuất hiện 4825 lần trong cột "Label", và "Sorry, I'll call later" xuất hiện 30 lần trong cột "EmailContent".

Kết quả này cung cấp thông tin quan trọng về phân bố và tính chất của các cột số trong DataFrame, giúp bạn hiểu và khám phá dữ liệu.

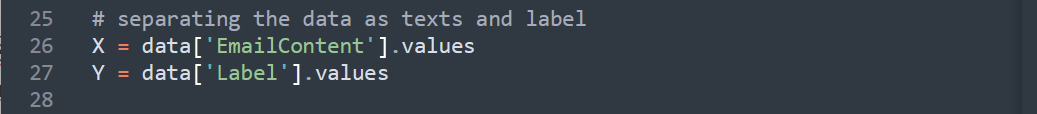
data.isnull()



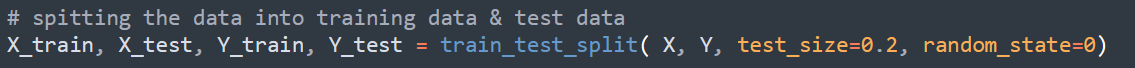
* Có hai cột trong DataFrame mới là "Label" và "EmailContent".
* Mỗi cột đều có 5572 hàng.
* Các giá trị trong DataFrame mới là False, vì nói rằng không có ô dữ liệu nào trong **`data`** có giá trị null.
* False cho biết rằng ô dữ liệu tương ứng không phải là giá trị null.

Kết quả từ **`data.isnull()`** chỉ ra rằng không có giá trị null nào trong DataFrame **`data`**. Điều này cho thấy dữ liệu của bạn không bị thiếu giá trị null và có đầy đủ dữ liệu trong các cột "Label" và "EmailContent".

Gán giá trị:



Chia dữ liệu thành hai tập: tập huấn luyện (training set) và tập kiểm tra (test set) dựa trên dữ liệu của cột "EmailContent" và cột "Label" trong DataFrame `data`.



* **`X`** là mảng có giá trị là cột "EmailText" trong DataFrame **`data`**. Đây là tập dữ liệu đầu vào mà chúng ta muốn sử dụng để huấn luyện mô hình.
* **`Y`** là mảng có giá trị là cột "Label" trong DataFrame **`data`**. Đây là tập dữ liệu đầu ra tương ứng với các mẫu dữ liệu trong **`X`**.
* **`train\_test\_split()`** là hàm trong sklearn để chia dữ liệu thành hai tập: tập huấn luyện và tập kiểm tra.
* **`X\_train`** là mảng chứa mẫu dữ liệu đầu vào cho tập huấn luyện.
* **`X\_test`** là mảng chứa mẫu dữ liệu đầu vào cho tập kiểm tra.
* **`Y\_train`** là mảng chứa mẫu dữ liệu đầu ra tương ứng với tập huấn luyện.
* **`Y\_test`** là mảng chứa mẫu dữ liệu đầu ra tương ứng với tập kiểm tra.
* **`test\_size=0.2`** chỉ định tỷ lệ dữ liệu kiểm tra là 0.2, tức là 20% dữ liệu được sử dụng cho kiểm tra và 80% dữ liệu được sử dụng cho huấn luyện.
* **`random\_state=0`** là một giá trị cố định dùng để cố định việc phân chia dữ liệu thành hai tập đối với việc chạy lại chương trình.

Việc chia dữ liệu thành tập huấn luyện và tập kiểm tra giúp đánh giá hiệu suất của mô hình trong quá trình phát triển và sử dụng mô hình trên dữ liệu chưa được nhìn thấy.

Chuyển đổi dữ liệu từ dạng văn bản (string) sang dạng số nguyên (integer).

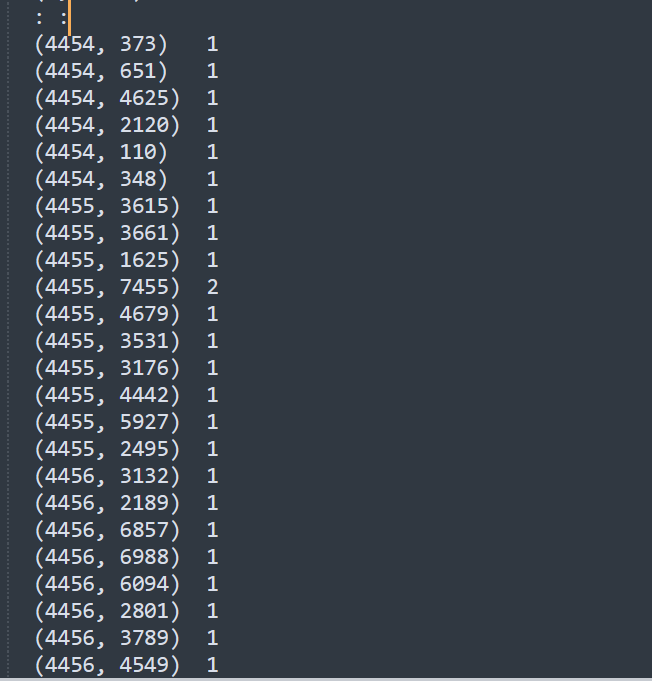
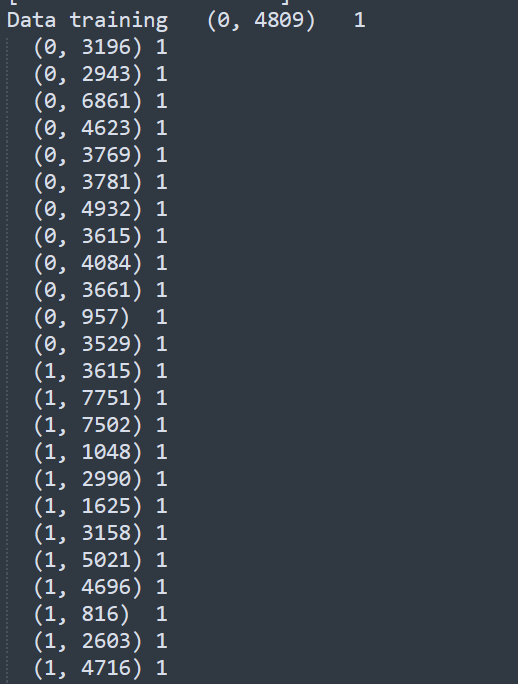


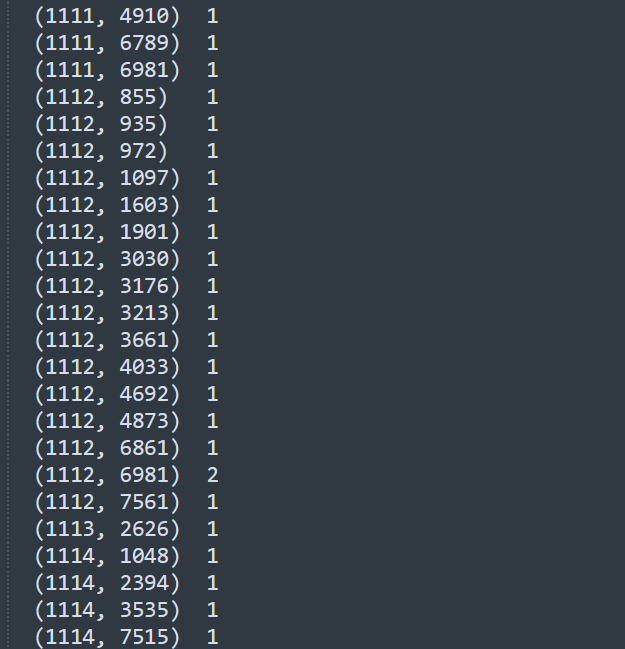
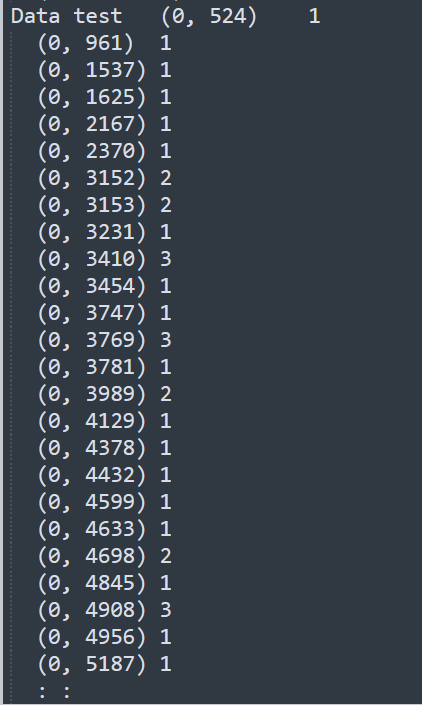
* Đầu tiên, ta khởi tạo một đối tượng **`CountVectorizer`** với **`cv = CountVectorizer()`**.
* Sau đó, ta sử dụng phương thức **`fit\_transform()`** để huấn luyện và chuyển đổi dữ liệu huấn luyện **`X\_train`**. Phương thức **`fit\_transform()`** sẽ tính toán và tạo ra ma trận từ dữ liệu huấn luyện, và đồng thời trả về ma trận đã chuyển đổi.
* Tiếp theo, ta sử dụng phương thức **`transform()`** để chuyển đổi dữ liệu kiểm tra **`X\_test`**. Phương thức này chỉ chuyển đổi dữ liệu theo cấu trúc đã được huấn luyện từ dữ liệu huấn luyện, mà không tính toán lại.

Kết quả là, **`X\_train`** và **`X\_test`** sau khi chuyển đổi sẽ là những ma trận số nguyên thể hiện sự xuất hiện của các từ trong dữ liệu văn bản. Điều này cho phép chúng ta sử dụng các thuật toán học máy và mô hình học máy để xử lý và dự đoán trên dữ liệu văn bản.



Kết quả:

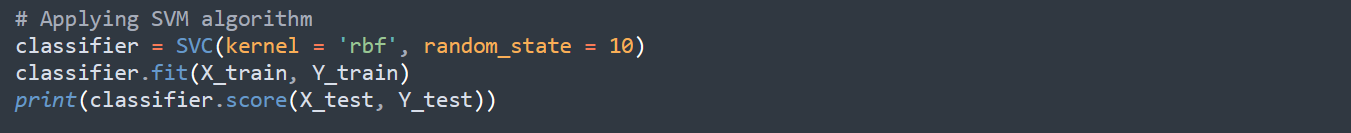




Dữ liệu huấn luyện (**`X\_train`**) được biểu diễn dưới dạng ma trận ma trận thưa (sparse matrix). Mỗi phần tử trong ma trận biểu diễn cho một từ trong từ điển của các từ xuất hiện trong nội dung email. Cặp (i, j) trong ma trận cho biết từ có chỉ số (index) j xuất hiện bao nhiêu lần trong mẫu email có chỉ số (index) i. Ví dụ, (0, 4809) 1 có nghĩa là từ có chỉ số (index) 4809 xuất hiện 1 lần trong mẫu email đầu tiên.

Tương tự, dữ liệu kiểm tra (**`X\_test`**) cũng được biểu diễn dưới dạng ma trận thưa với số lần xuất hiện các từ tương ứng trong mỗi mẫu email.

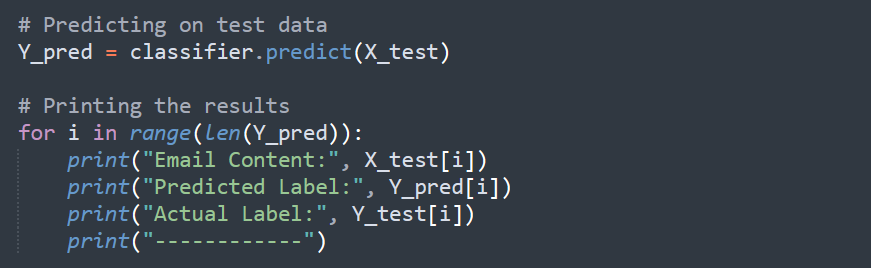
Mô hình SVM:



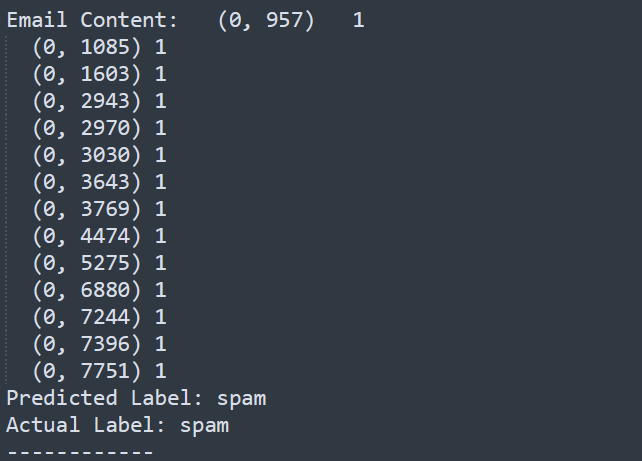
Các dòng code này tạo ra một mô hình SVM, huấn luyện nó trên dữ liệu huấn luyện đã được mã hóa và in ra độ chính xác của mô hình trên dữ liệu kiểm tra.

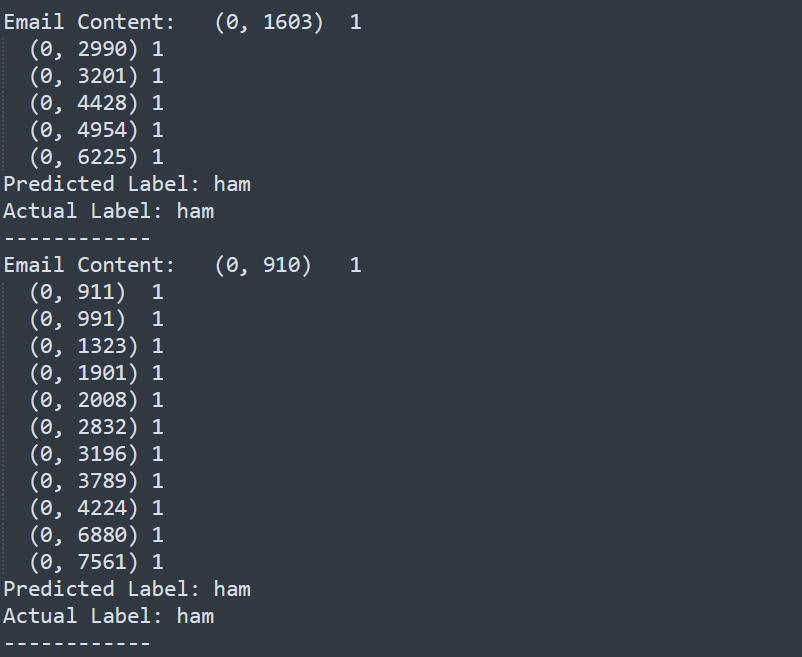
Kết quả : 0.9766816143497757

Sử dụng phương thức predict() của mô hình SVM và so sánh kết quả dự đoán với nhãn thực tế.



Kết quả:





Nguồn tham khảo : https://becominghuman.ai/spam-mail-detection-using-support-vector-machine-cdb57b0d62a8