







### **Mục lục**

[**Mục lục 2**](#_heading=h.gjdgxs)

[**1 Giới thiệu 3**](#_heading=h.30j0zll)

[1.1 Mục đích nghiên cứu 3](#_heading=h.1fob9te)

[1.2 Tổ chức bài báo. 3](#_heading=h.3znysh7)

[**2. Chuẩn bị dữ liệu. 4**](#_heading=h.2et92p0)

[2.1 Mô tả dữ liệu. 4](#_heading=h.tyjcwt)

[2.2 Đọc tập dữ liệu 5](#_heading=h.3dy6vkm)

[2.3 Data Cleaning 6](#_heading=h.1t3h5sf)

[**3. Phân tích dữ liệu thăm dò 8**](#_heading=h.4d34og8)

[**4. Xử lý dữ liệu 14**](#_heading=h.2s8eyo1)

[**5. Loại Mô Hình Dự Đoán và Kỹ Thuật Mô Hình Hóa 19**](#_heading=h.17dp8vu)

[5.1 Logistic Regression 19](#_heading=h.3rdcrjn)

[5.2 Support Vector Machine 20](#_heading=h.26in1rg)

[5.3 Multinomial Naive Bayes 21](#_heading=h.lnxbz9)

[5.4 Decision Tree Classifier 22](#_heading=h.35nkun2)

[5.5 K-Nearest Neighbors 26](#_heading=h.1ksv4uv)

[5.6 Random Forest Classifier 26](#_heading=h.44sinio)

[5.7 AdaBoost Classifier 28](#_heading=h.2jxsxqh)

[5.8 Bagging Classifier 29](#_heading=h.z337ya)

[5.9 Extra Trees Classifier 29](#_heading=h.3j2qqm3)

[5.10 Gradient Boosting Classifier 30](#_heading=h.1y810tw)

[5.11 XGBoost Classifier 31](#_heading=h.4i7ojhp)

[**6. Xây dựng mô hình và đánh giá. 31**](#_heading=h.2xcytpi)

[6.1 Phân tách tập dữ liệu 31](#_heading=h.1ci93xb)

[6.2 Modeling Approach 32](#_heading=h.3whwml4)

[6.3 Mô hình 32](#_heading=h.2bn6wsx)

[6.3.1 Logistic Regression 32](#_heading=h.qsh70q)

[6.3.2 Support Vector Machine 34](#_heading=h.3as4poj)

[6.3.3 K-Nearest Neighbors 36](#_heading=h.1pxezwc)

[6.3.4 Multinomial Naive Bayes 38](#_heading=h.49x2ik5)

[6.3.5 Decision Tree Classifier 40](#_heading=h.2p2csry)

[6.3.6 Random Forest Classifier 42](#_heading=h.147n2zr)

[6.3.7 AdaBoost Classifier 44](#_heading=h.3o7alnk)

[6.3.8 Bagging Classifier 46](#_heading=h.23ckvvd)

[6.3.9 Extra Trees Classifier 48](#_heading=h.ihv636)

[6.3.10 Gradient Boosting Classifier 50](#_heading=h.32hioqz)

[6.3.11 XGBoost Classifier 52](#_heading=h.1hmsyys)

[**7. Kết luận. 54**](#_heading=h.41mghml)

[7.1. Nhận xét và đánh giá 54](#_heading=h.2grqrue)

[7.2. Hiện thực hóa lên website 57](#_heading=h.vx1227)

[**8. Bảng phân công 60**](#_heading=h.pw769e7v7fyx)

[**9. Tài liệu tham khảo. 60**](#_heading=h.3fwokq0)

### **1 Giới thiệu**

#### **1.1 Mục đích nghiên cứu**

Ngày nay tin nhắn SMS spam không chỉ là phiền toái mà còn có thể làm mất thời gian và gây phiền hà cho người dùng. Việc phát hiện và ngăn chặn spam có thể bảo vệ người dùng khỏi việc nhận các tin nhắn không mong muốn hoặc lừa đảo. Một số tin nhắn spam có thể chứa các liên kết độc hại hoặc yêu cầu thông tin cá nhân. Bằng cách phát hiện và loại bỏ chúng, nghiên cứu này giúp bảo vệ thông tin cá nhân và an toàn cho người dùng. Việc loại bỏ tin nhắn spam giúp cải thiện trải nghiệm sử dụng điện thoại di động, giúp người dùng tập trung vào các thông điệp quan trọng và chất lượng hơn. Các công ty viễn thông, nhà cung cấp dịch vụ di động và người dùng cá nhân đều có lợi ích từ việc áp dụng các cảnh báo spam SMS. Điều này giúp tối ưu hóa tài nguyên mạng và giảm thiểu các chi phí liên quan đến việc xử lý tin nhắn spam. Việc áp dụng mô hình học máy vào việc phát hiện tin nhắn spam không chỉ giúp cải thiện khả năng phát hiện mà còn là cơ hội để phát triển và thử nghiệm các mô hình mới, từ đó nâng cao hiệu suất và sức mạnh của hệ thống nhận diện.

Tóm lại, nghiên cứu này không chỉ giúp cải thiện trải nghiệm người dùng mà còn đóng góp vào việc xây dựng các công cụ và kỹ thuật ngăn chặn spam SMS hiệu quả hơn trong cộng đồng người dùng điện thoại di động.

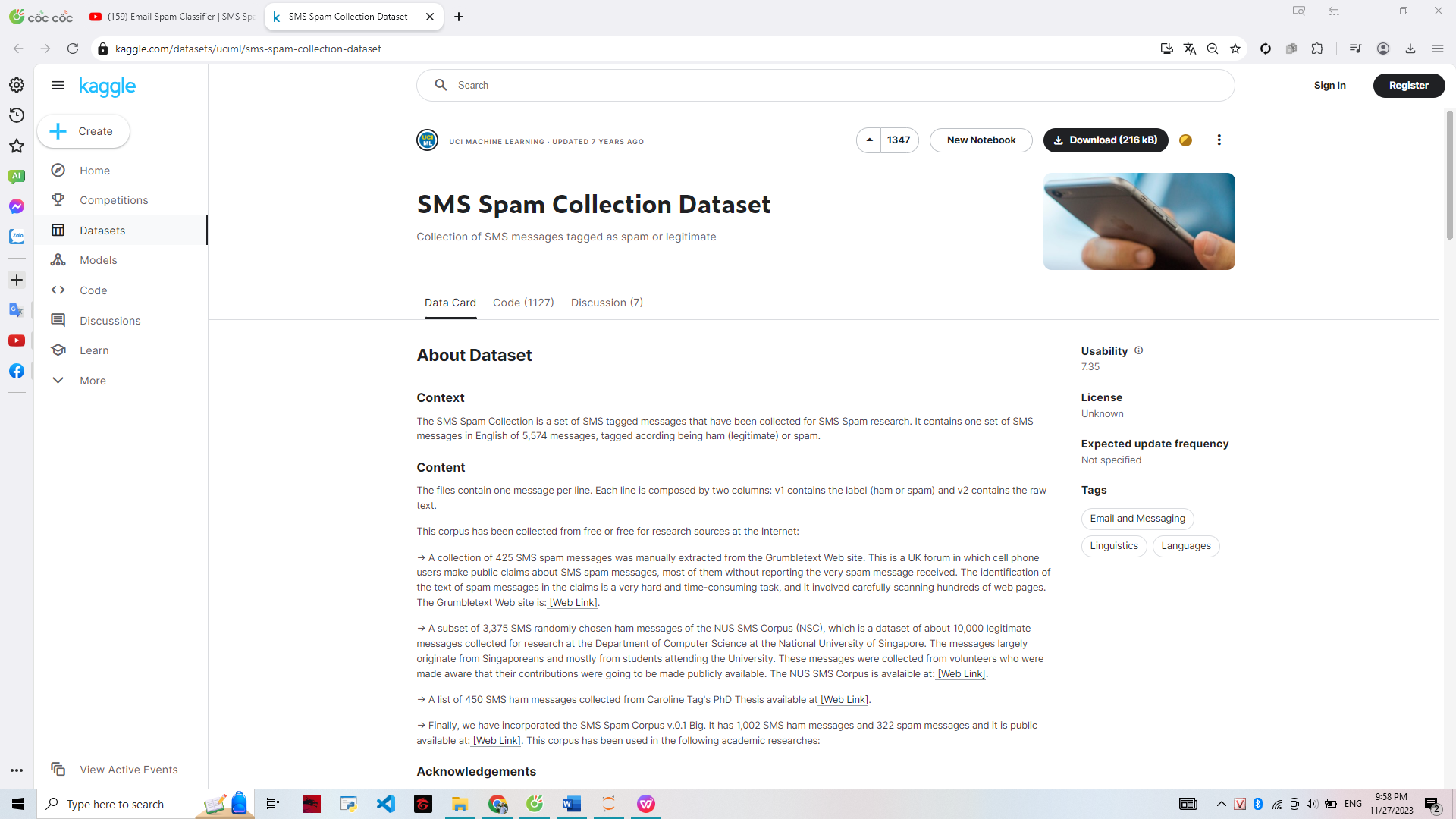
#### **1.2 Tổ chức bài báo.**

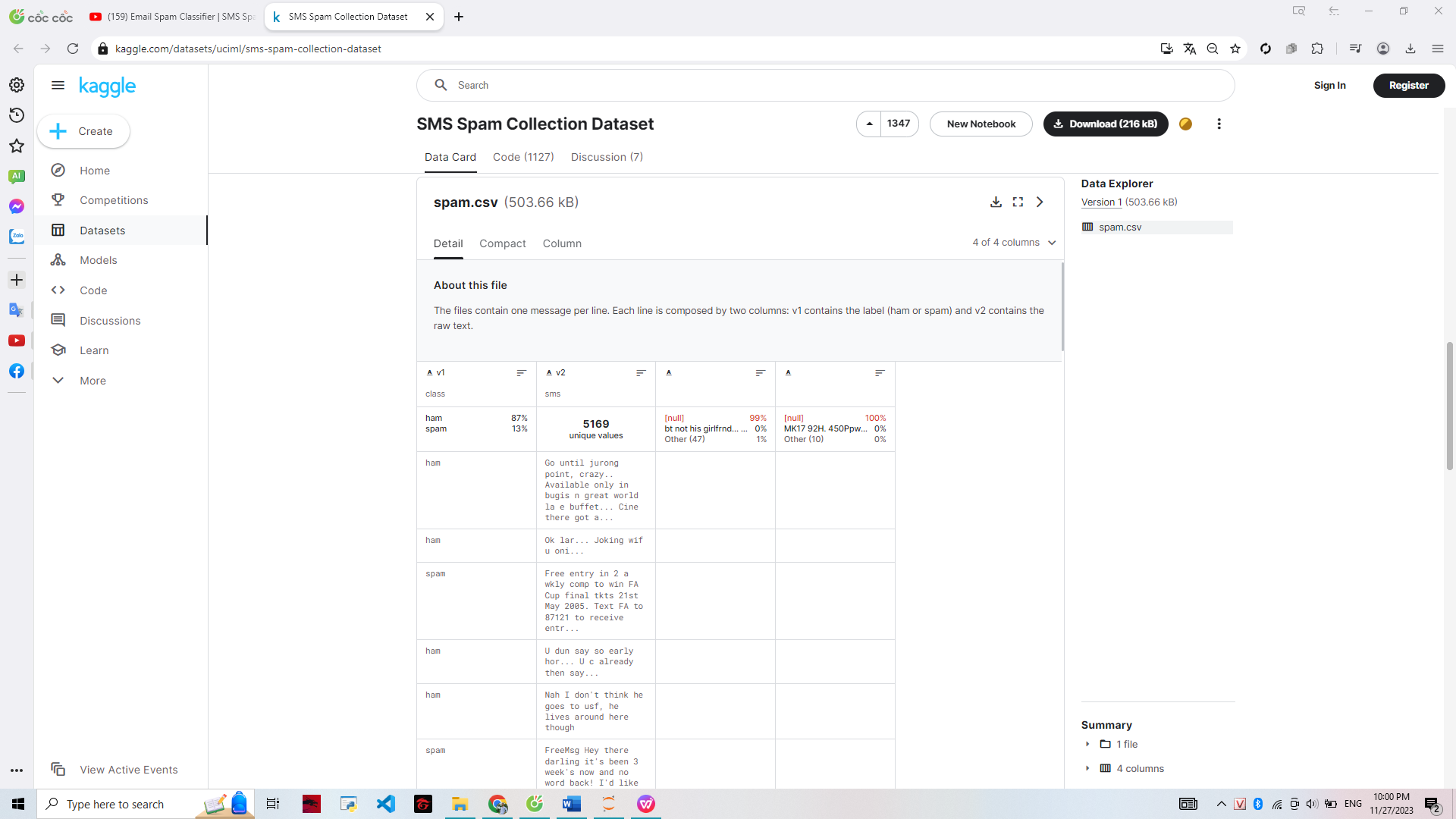
Bài viết này được tổ chức như sau: trong phần tiếp theo, phần 2, chúng em sẽ tiến hành chuẩn bị dữ liệu, thảo luận về loại vấn đề của chúng em. Tiếp theo trong phần 3, chúng em phân tích dữ liệu, thảo luận về loại vấn đề của chúng em và loại dự đoán học máy nên được áp dụng; chúng em cũng liệt kê các kỹ thuật dự đoán sẽ được sử dụng. Phần 4, Chúng em xử lý dữ liệu, làm sạch dữ liệu, loại bỏ ngoại lệ. Trong phần 5, chúng em chọn các thuật toán để thực hiện các kỹ thuật. Trong phần 6, chúng em xây dựng mô hình dựa trên các thuật toán này; chúng em cũng đào tạo và kiểm tra từng mô hình. Trong phần 7, chúng em phân tích và so sánh kết quả thu được từ phần 6 và kết luận bài viết.

### **2. Chuẩn bị dữ liệu.**

#### **2.1 Mô tả dữ liệu.**

Dataset name: SMS Spam Collection Dataset





SMS Spam Collection Dataset là một tập hợp các tin nhắn được gắn thẻ SMS đã được thu thập để nghiên cứu Thư rác SMS. Nó chứa một bộ tin nhắn SMS bằng tiếng Anh gồm 5.574 tin nhắn, được gắn thẻ là ham (hợp pháp) hoặc spam (thư rác).

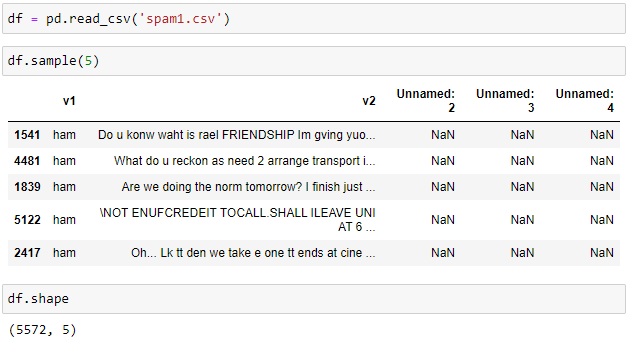
Các tập tin chứa một tin nhắn trên mỗi dòng. Mỗi dòng gồm có hai cột: v1 chứa nhãn (ham hoặc spam) và v2 chứa văn bản nội dung tin nhắn.

Dataset source: [SMS Spam Collection Dataset](https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset)

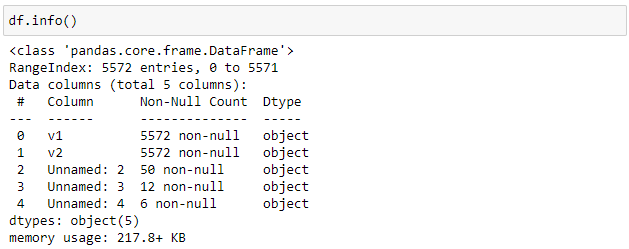
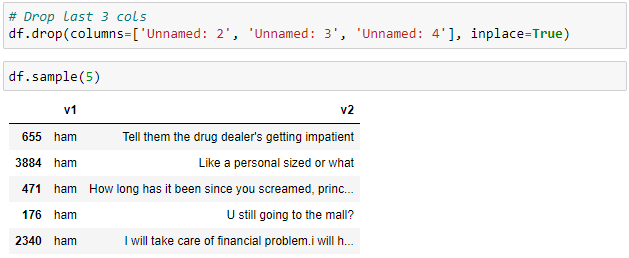
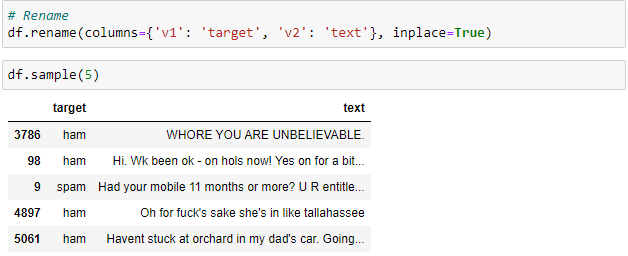
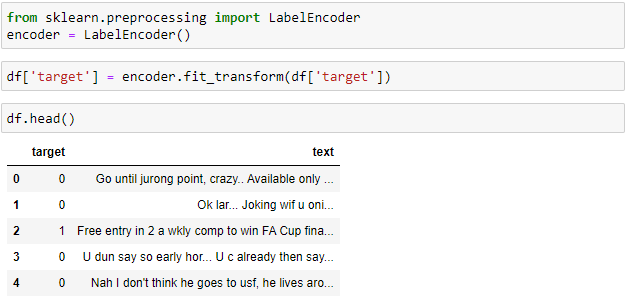
#### **2.2 Đọc tập dữ liệu**

* Thêm thư viện cần thiết

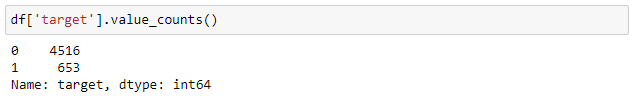
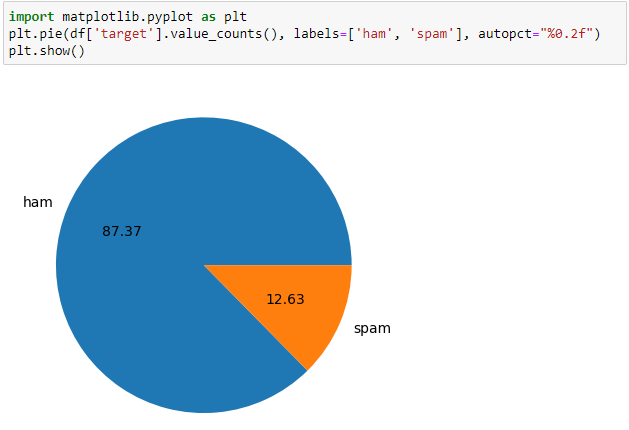
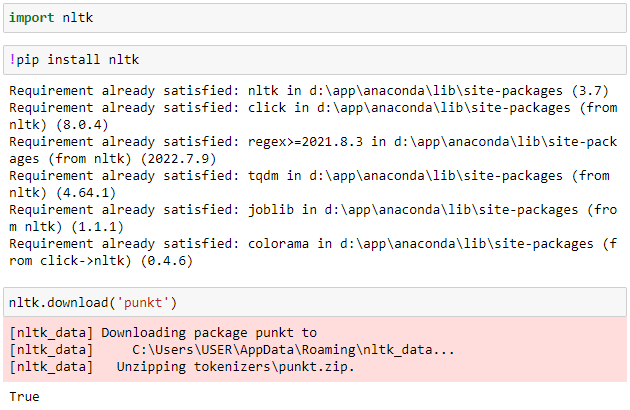
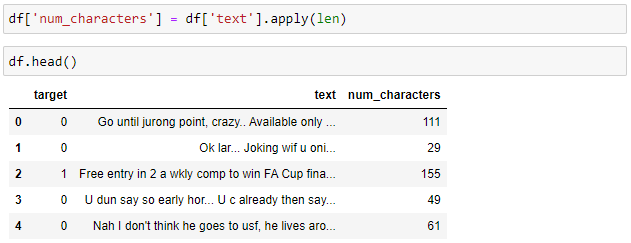
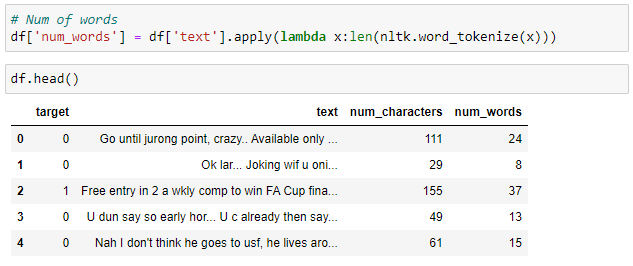
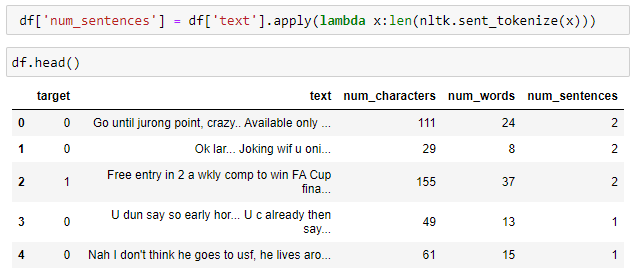
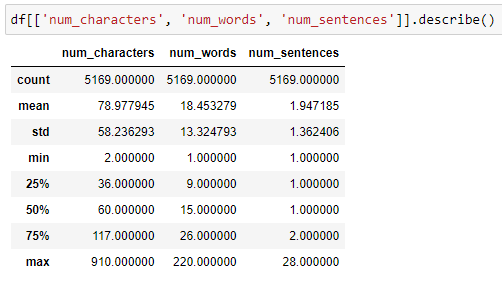
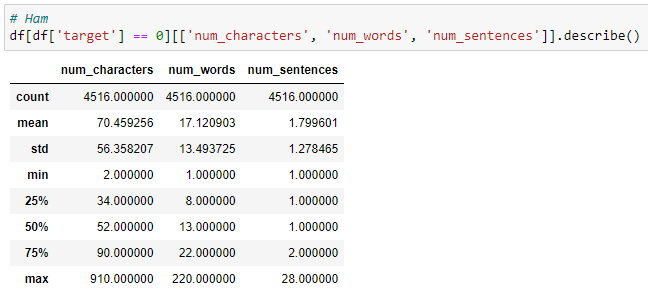
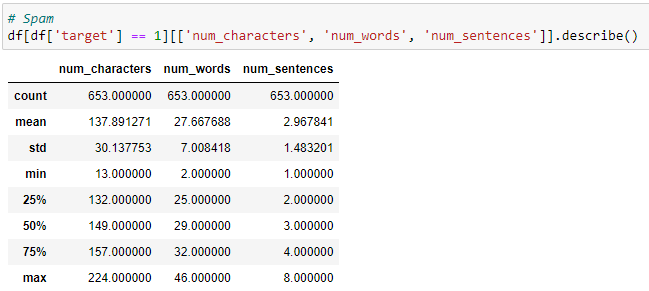
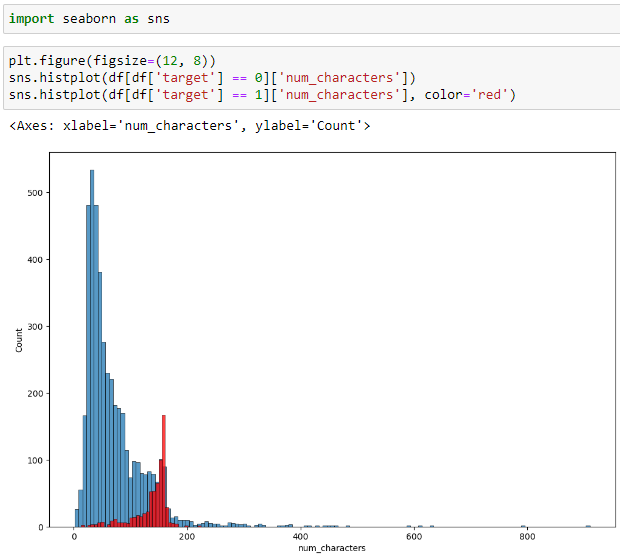


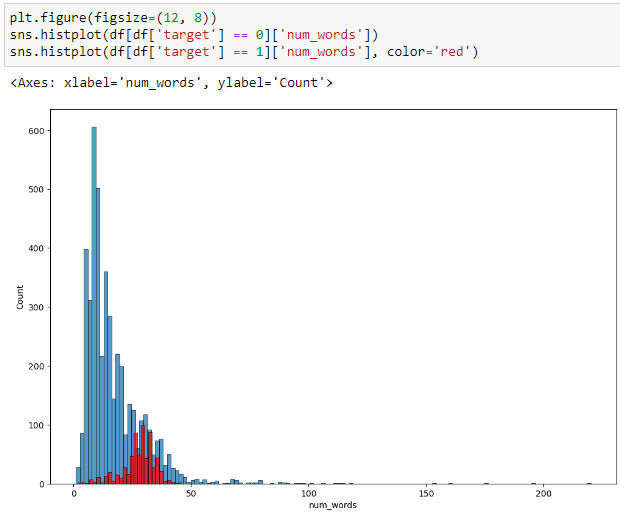
* Đọc file và xem các dòng dữ liệu

#### **2.3 Data Cleaning**

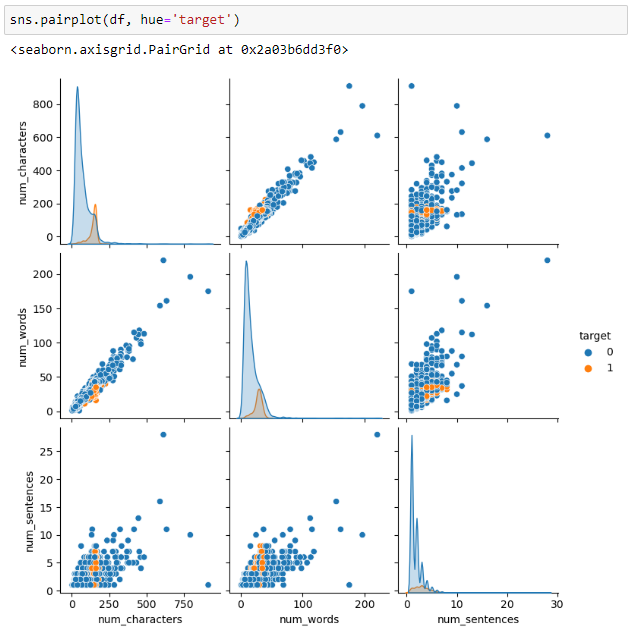
* Xem thông tin dữ liệu
* Xóa 3 cột Unnamed: 2, 3, 4
* Đổi tên 2 cột v1 và v2
* Thêm thư viện sklearn và chuyển đổi nhãn ham và spam thành giá trị số
* Kiểm tra giá trị bị thiếu
* Kiểm tra giá trị bị trùng lặp và loại bỏ

### **3. Phân tích dữ liệu thăm dò**

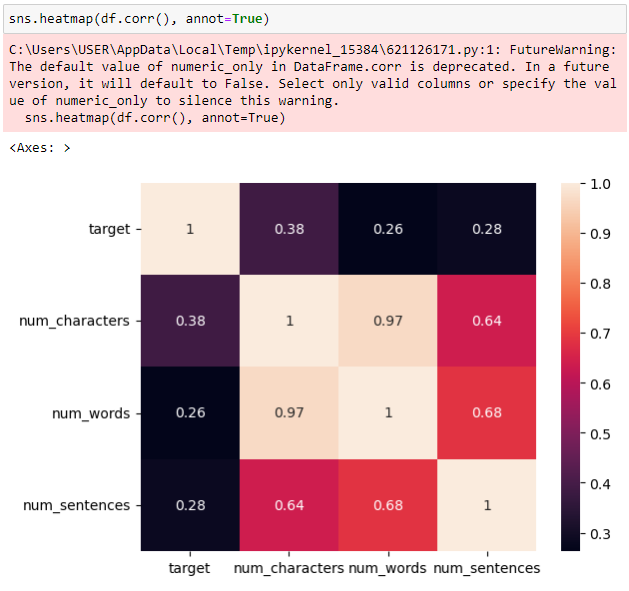
* Xem xét cột ‘target’. Ta thấy có 4516 dòng là ham và 653 dòng là spam.
* Sử dụng biểu đồ tròn để quan sát dữ liệu.
* Thêm thư viện NLTK và bộ dữ liệu ‘punkt’
* Tạo cột ‘num\_characters’ và đếm số lượng ký tự trong câu
* Tạo cột ‘num\_words’ và đếm số lượng từ trong câu
* Tạo cột ‘num\_sentences’ và đếm số lượng câu trong câu
* Tạo tóm tắt thống kê mô tả của các giá trị
* Tạo tóm tắt thống kê mô tả của giá trị của ‘ham’
* Tạo tóm tắt thống kê mô tả của giá trị của ‘spam’
* Thêm thư viện ‘seaborn’ và trực quan hóa dữ liệu ‘num\_characters’.
* Nhìn vào biểu đồ ta có thể thấy tin nhắn ‘ham’ thường có ít ký tự hơn ‘spam’
* Trực quan hóa dữ liệu ‘num\_words’.
* Tương tự biểu đồ của ‘num\_characters’, các từ của’ham’ ít hơn ‘spam’



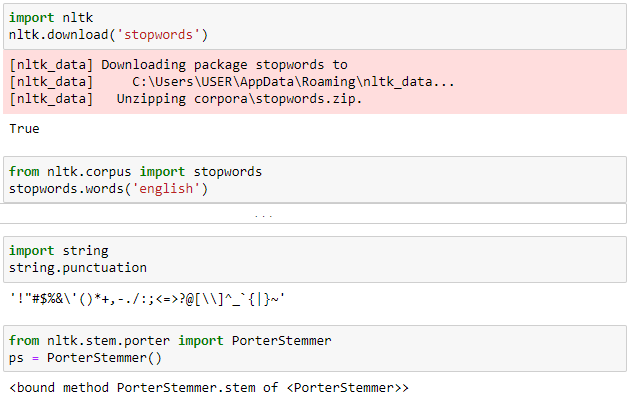
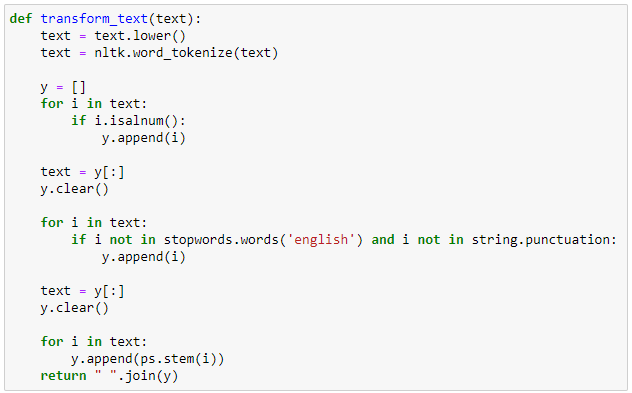
* Vẽ biểu đồ ‘Pairplot’
* Từ biểu đồ ta có thể thấy giữa ‘num\_words’ và ‘num\_characters’ có mối quan hệ tuyến tính.



* Vẽ biểu đồ ‘heatmap’.
* Từ biểu đồ ta có thể thấy giữa ‘num\_words’ và ‘num\_characters’ có độ tương quan rất cao.



### **4. Xử lý dữ liệu**

* Thêm thư viện cần thiết.
* Tạo hàm ‘transform\_text’.

Giải thích hàm ‘transform\_text’.

* text = text.lower(): Chuyển đổi tất cả các ký tự trong văn bản thành chữ thường để đồng nhất dữ liệu.
* text = nltk.word\_tokenize(text): Sử dụng tokenizer từ thư viện NLTK để chia văn bản thành danh sách các từ (tokens).

Tạo một danh sách rỗng y và lặp qua từng từ trong danh sách text:

* if i.isalnum(): y.append(i): Thêm các từ chỉ chứa các ký tự chữ cái hoặc số vào danh sách y.

Sao chép nội dung của danh sách y vào text và làm rỗng danh sách y.

Lặp qua từng từ trong danh sách text:

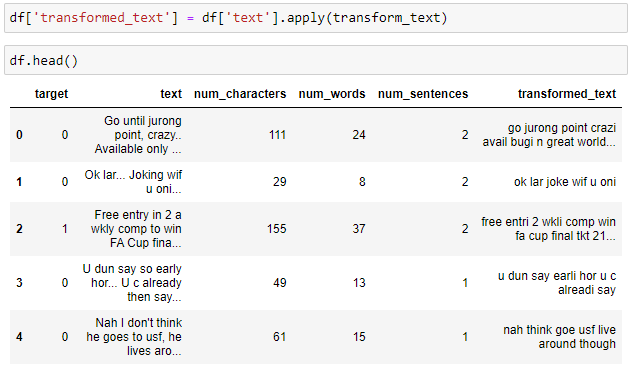
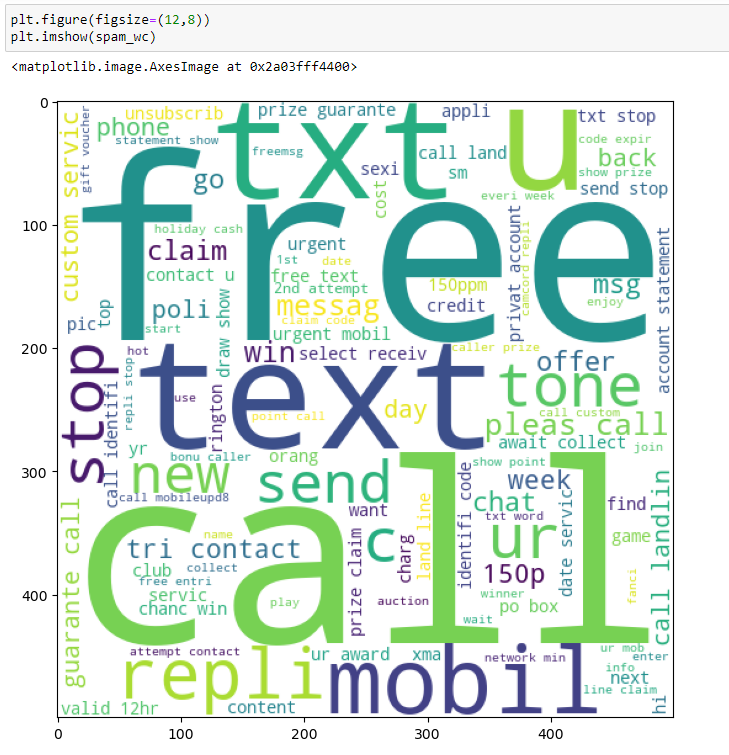
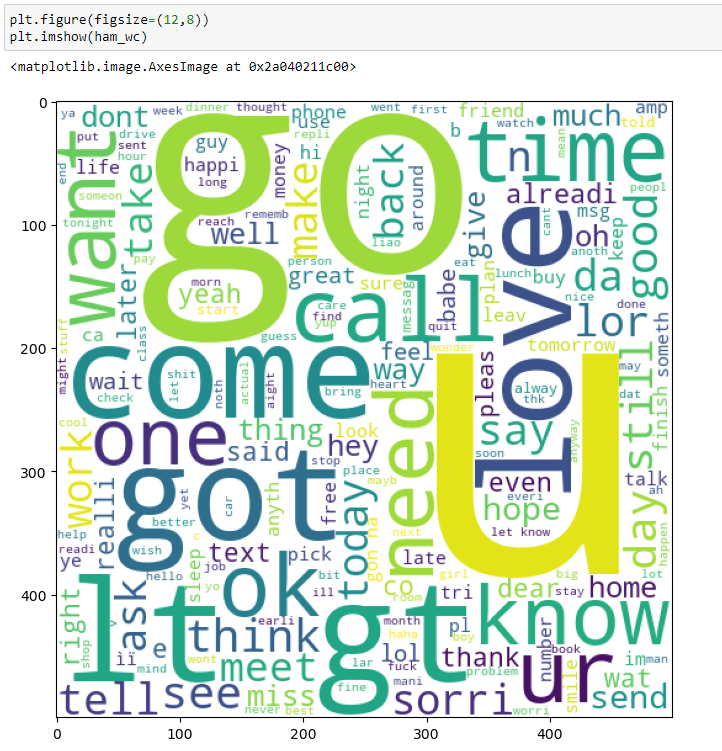
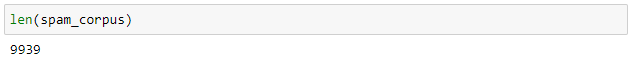
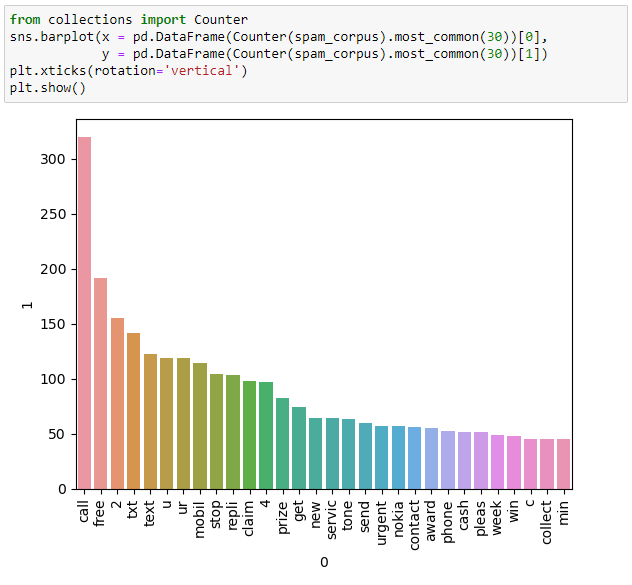
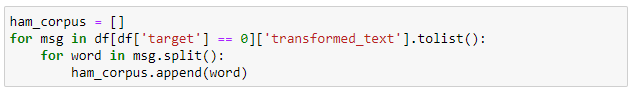
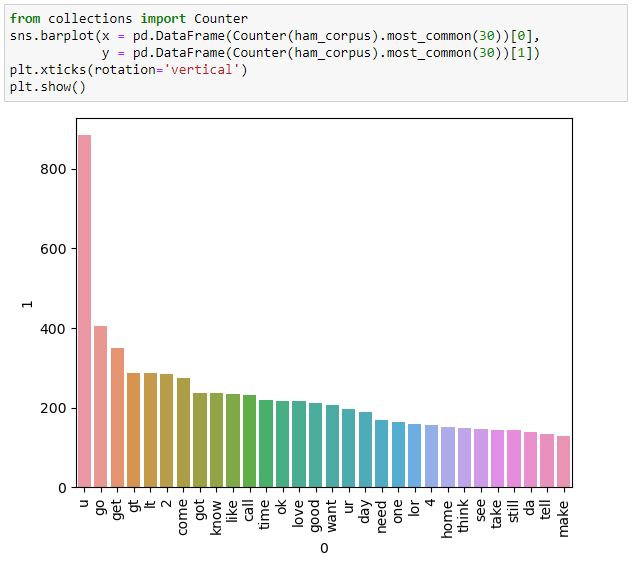
* if i not in stopwords.words('english') and i not in string.punctuation: y.append(i): Loại bỏ từ dừng (stopwords) và dấu câu khỏi danh sách y.

Sao chép nội dung của danh sách y vào text và làm rỗng danh sách y.

Lặp qua từng từ trong danh sách text:

* y.append(ps.stem(i)): Áp dụng quá trình stemming bằng cách sử dụng ps.stem(i) để giảm các từ về dạng gốc.

Trả về chuỗi kết quả, được tạo bằng cách nối các từ trong danh sách y thành một chuỗi, cách nhau bởi khoảng trắng.

* Áp dụng hàm ‘transform\_text’ và xem kết quả.
* Thêm thư viện WordCloud để hiển thị tần suất xuất hiện của các từ.
* Tạo Word Cloud cho ‘spam’.
* Trực quan hóa cho ‘spam\_wc’.
* Theo những gì quan sát được các từ có tần suất xuất hiện nhiều nhất trong ‘spam’ là ‘call’, ‘free’, ‘txt’, ‘text’,...
* Tạo WordCloud cho ‘ham’.
* Trực quan hóa cho ‘ham\_wc’.
* Theo những gì quan sát được các từ có tần suất xuất hiện nhiều nhất trong ‘ham’ là ‘go’, ‘u’, ‘come’, ‘got’,...
* Tạo ra danh sách ‘spam\_corpus’ chứa tất cả các từ từ văn bản được đánh dấu là "spam" trong DataFrame ‘df’.
* Tính số lượng phần tử trong ‘spam\_corpus’.
* Vẽ biểu đồ cột (bar plot) cho 30 từ xuất hiện nhiều nhất trong ‘spam\_corpus’.
* Tạo ra danh sách ‘ham\_corpus’ chứa tất cả các từ từ văn bản được đánh dấu là "ham" trong DataFrame ‘df’.
* Tính số lượng phần tử trong ‘ham\_corpus’.
* Vẽ biểu đồ cột (bar plot) cho 30 từ xuất hiện nhiều nhất trong ‘ham\_corpus’.

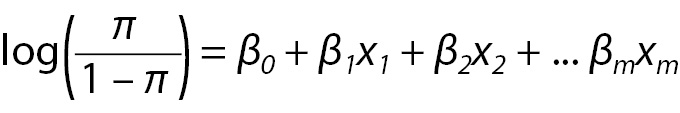
### **5. Loại Mô Hình Dự Đoán và Kỹ Thuật Mô Hình Hóa**

#### **5.1 Logistic Regression**

**Định nghĩa**

Logistic regression hoạt động rất giống với linear regression, nhưng với biến đáp ứng nhị phân. Lợi thế lớn nhất khi so sánh với Mantel-Haenszel OR là bạn có thể sử dụng các biến giải thích liên tục và dễ dàng xử lý hơn hai biến giải thích đồng thời. Mặc dù có vẻ tầm thường, đặc điểm cuối cùng này là rất cần thiết khi chúng ta quan tâm đến tác động của nhiều biến giải thích đối với biến đáp ứng. Nếu chúng ta xem xét nhiều biến giải thích độc lập, chúng ta sẽ bỏ qua sự đồng biến giữa các biến và phải chịu các tác động confounding, như đã được chứng minh trong ví dụ trên khi tác động của điều trị đối với xác suất tử vong bị che giấu một phần bởi tác động của tuổi tác.

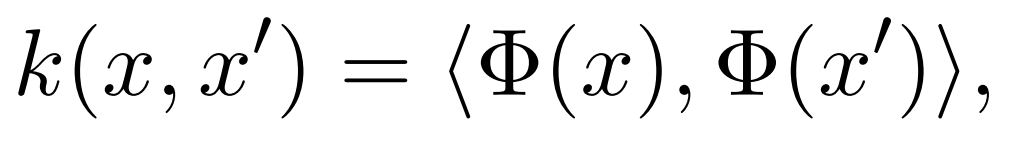
Logistic regression sẽ mô hình hóa cơ hội của một kết quả dựa trên các đặc điểm cá nhân. Bởi vì cơ hội là một tỷ lệ, nên những gì sẽ được mô hình hóa thực sự là logarit của cơ hội được đưa ra bởi:



Với ký hiệu π biểu thị xác suất của một sự kiện (ví dụ: tử vong trong ví dụ trước), và βi là các hệ số hồi quy liên quan đến nhóm tham chiếu và các biến giải thích xi. Tại thời điểm này, cần nhấn mạnh một khái niệm quan trọng. Nhóm tham chiếu, được biểu thị bởi β0, được cấu thành bởi những cá nhân thể hiện mức tham chiếu của mỗi và tất cả các biến x1 ... m. Để minh họa, xem xét ví dụ trước của chúng tôi, đây là những cá nhân lớn tuổi hơn đã nhận được điều trị tiêu chuẩn. Sau đó, chúng ta sẽ thảo luận về cách đặt mức tham chiếu.

#### **5.2 Support Vector Machine**

Support Vector Machines (SVMs) sử dụng ánh xạ ẩn Φ của dữ liệu đầu vào vào không gian đặc trưng kích thước cao được xác định bởi hàm kernel, tức là hàm trả về tích nội [Φ(x), Φ(x’)] giữa hình ảnh của hai điểm dữ liệu x, x’ trong không gian đặc trưng. Sau đó, việc học diễn ra trong không gian đặc trưng và các điểm dữ liệu chỉ xuất hiện bên trong tích điểm với các điểm khác. Điều này thường được gọi là "kernel trick" (Schölkopf và Smola 2002). Chính xác hơn, nếu sử dụng phép chiếu Φ: X → H, thì tích điểm [Φ(x), Φ(x’)] có thể được biểu diễn bởi hàm kernel k



Điều này đơn giản hơn về mặt tính toán so với việc chiếu x và x’ một cách rõ ràng vào không gian đặc trưng H. Một đặc tính thú vị của SVMs và các hệ thống dựa trên kernel khác là một khi hàm kernel hợp lệ đã được chọn, người ta thực tế có thể làm việc trong các không gian có bất kỳ kích thước nào mà không có chi phí tính toán bổ sung đáng kể, vì ánh xạ đặc trưng không bao giờ được thực hiện một cách hiệu quả. Trong thực tế, người ta thậm chí không cần biết những tính năng nào đang được sử dụng.

Một ưu điểm khác của SVMs và phương pháp kernel là người ta có thể thiết kế và sử dụng kernel cho một vấn đề cụ thể có thể áp dụng trực tiếp cho dữ liệu mà không cần quá trình trích xuất tính năng. Điều này đặc biệt quan trọng trong các vấn đề mà nhiều cấu trúc của dữ liệu bị mất bởi quá trình trích xuất tính năng (ví dụ: xử lý văn bản).

Huấn luyện SVM để phân loại, hồi quy hoặc phát hiện mới lạ liên quan đến việc giải quyết vấn đề tối ưu hóa bậc hai. Sử dụng trình giải quyết vấn đề bậc hai tiêu chuẩn để huấn luyện SVM sẽ liên quan đến việc giải quyết một vấn đề QP lớn ngay cả đối với một tập dữ liệu có kích thước vừa phải, bao gồm việc tính toán ma trận m × m trong bộ nhớ (m số điểm huấn luyện). Điều này sẽ nghiêm trọng hạn chế kích thước của các vấn đề có thể áp dụng SVM. Để xử lý vấn đề này, các phương pháp như SMO (Platt 1998), chunking (Osuna, Freund, và Girosi 1997) và SVM đơn giản (Vishwanathan, Smola, và Murty 2003) tồn tại, chúng tính toán lặp lại giải pháp của SVM và có độ phức tạp về không gian O(N^k) trong đó k nằm giữa 1 đến 2.5 và có độ phức tạp về không gian tuyến tính.

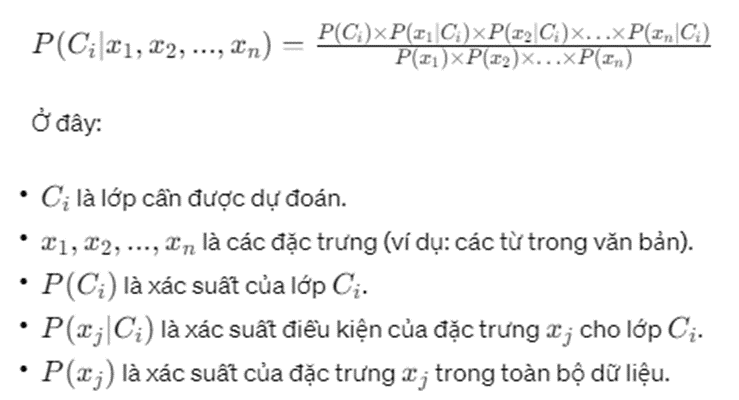
#### **5.3 Multinomial Naive Bayes**

Thuật toán Multinomial Naive Bayes là một thuật toán machine learning được sử dụng chủ yếu trong các bài toán phân loại văn bản. Nó là một biến thể của thuật toán Naive Bayes, một mô hình thống kê dựa trên nguyên tắc Bayes.

Mục tiêu chính của thuật toán Multinomial Naive Bayes là phân loại các văn bản vào các danh mục hoặc lớp khác nhau dựa trên nội dung của chúng. Đặc điểm của thuật toán này là giả sử rằng mỗi đặc trưng (từ vựng, từ ngữ) trong văn bản ảnh hưởng độc lập đối với các đặc trưng khác, điều này là giả định "ngây thơ" (naive) của mô hình Naive Bayes.

Thuật toán Multinomial Naive Bayes thường được sử dụng trong các tình huống mà dữ liệu đầu vào có thể được biểu diễn dưới dạng bộ đếm, chẳng hạn như số lần xuất hiện của từng từ trong một văn bản. Đối với bài toán phân loại văn bản, các đặc trưng thường là các từ trong văn bản và tần suất xuất hiện của chúng.

Công thức cơ bản của Naive Bayes cho bài toán phân loại có thể được biểu diễn như sau:

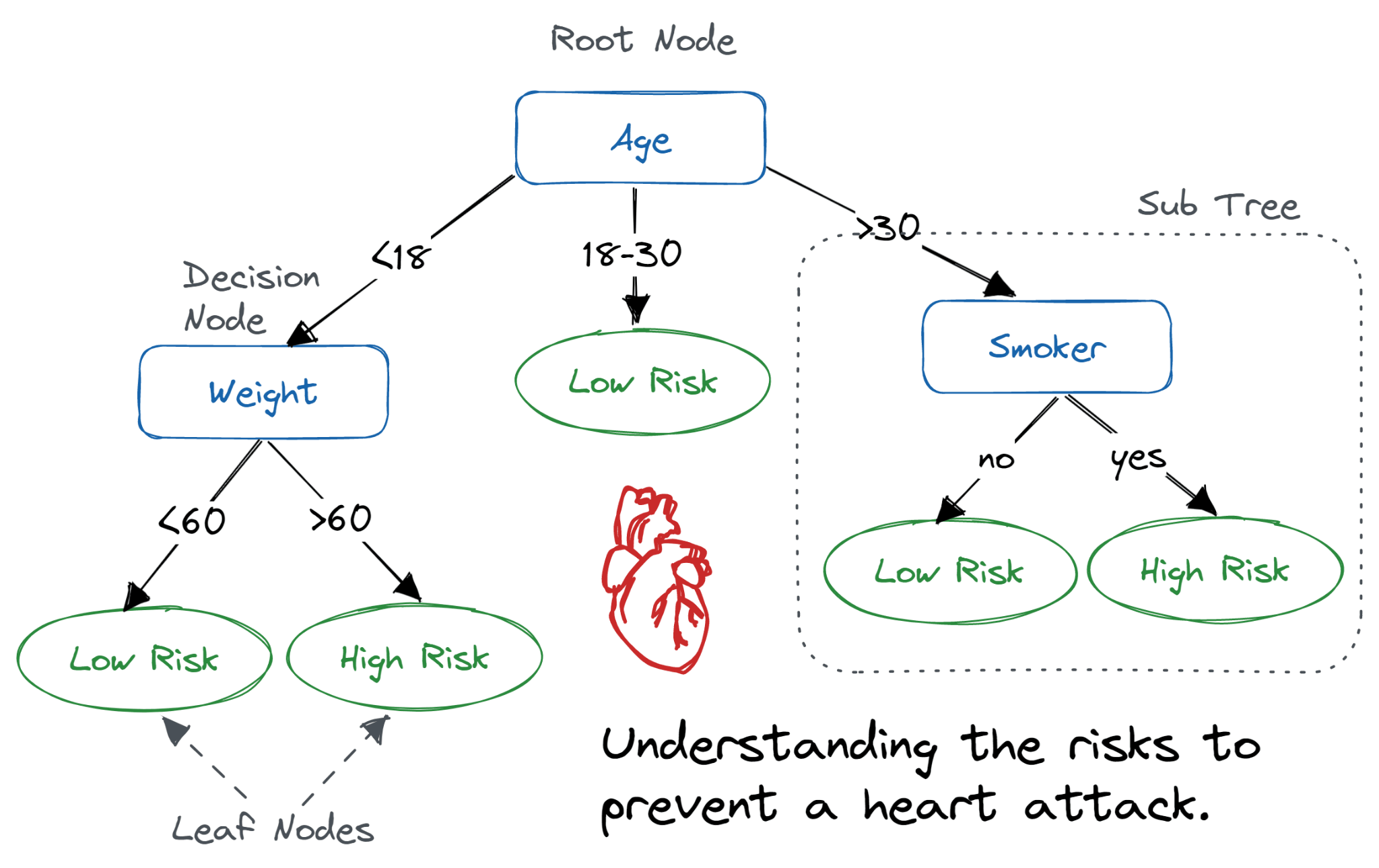


Mặc dù giả định ngây thơ của mô hình có thể không phản ánh đúng thực tế, nhưng Multinomial Naive Bayes thường cho kết quả tốt trên nhiều tập dữ liệu văn bản thực tế.

#### **5.4 Decision Tree Classifier**

Cây quyết định là một cấu trúc cây giống như sơ đồ, trong đó mỗi nút bên trong đại diện cho một đặc trưng (hoặc thuộc tính), nhánh đại diện cho quy tắc quyết định và mỗi nút lá đại diện cho kết quả.

Nút trên cùng trong cây quyết định được gọi là nút gốc. Nó học cách phân chia dựa trên giá trị thuộc tính. Nó phân chia cây theo cách đệ quy được gọi là phân chia đệ quy. Cấu trúc giống như sơ đồ này giúp bạn đưa ra quyết định. Hình ảnh trực quan của nó giống như một sơ đồ flowchart dễ dàng mô phỏng tư duy của con người. Đó là lý do tại sao cây quyết định dễ hiểu và dễ giải thích.



Cây quyết định là một thuật toán học máy loại hộp trắng. Nó chia sẻ logic ra quyết định nội bộ, không có sẵn trong các thuật toán loại hộp đen như với mạng nơ ron. Thời gian huấn luyện của nó nhanh hơn so với thuật toán mạng nơ ron.

Độ phức tạp của thời gian của cây quyết định là hàm của số lượng bản ghi và thuộc tính trong dữ liệu được cung cấp. Cây quyết định là phương pháp phân phối miễn phí hoặc không tham số không phụ thuộc vào giả định phân phối xác suất. Cây quyết định có thể xử lý dữ liệu kích thước cao với độ chính xác tốt.

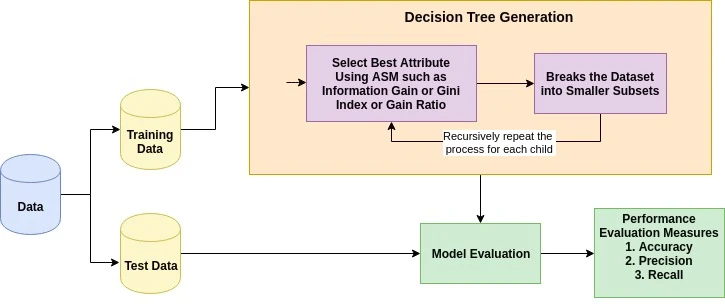
Ý tưởng cơ bản đằng sau bất kỳ thuật toán cây quyết định nào như sau:

1. Chọn thuộc tính tốt nhất sử dụng Các thước đo lựa chọn thuộc tính (ASM) để chia tách các bản ghi.

2. Làm cho thuộc tính đó trở thành một nút quyết định và chia nhỏ tập dữ liệu thành các tập con nhỏ hơn.

3. Bắt đầu xây dựng cây bằng cách lặp lại quá trình này một cách đệ quy cho từng con cho đến khi một trong những điều kiện sau được đáp ứng:

* Tất cả các tuple thuộc về cùng một giá trị thuộc tính.
* Không còn thuộc tính nào còn lại.
* Không còn trường hợp nào nữa.



Đo lường lựa chọn thuộc tính là một phương pháp heuristic để lựa chọn tiêu chuẩn phân chia dữ liệu theo cách tốt nhất có thể. Nó cũng được gọi là quy tắc phân chia vì nó giúp chúng ta xác định điểm ngắt đối với các tuple trên một nút nhất định. ASM cung cấp thứ hạng cho từng tính năng (hoặc thuộc tính) bằng cách giải thích tập dữ liệu đã cho. Thuộc tính điểm số tốt nhất sẽ được chọn làm thuộc tính phân chia (Nguồn). Trong trường hợp của một thuộc tính có giá trị liên tục, các điểm phân chia cho các nhánh cũng cần được xác định. Các biện pháp lựa chọn phổ biến nhất là Information Gain (Độ lợi thông tin), Gain Ratio (Tỷ lệ lợi ích) và Gini Index (Chỉ số Gini).

Claude Shannon đã phát minh ra khái niệm entropy, đo lường sự không tinh khiết của tập đầu vào. Trong vật lý và toán học, entropy được gọi là ngẫu nhiên hoặc sự không tinh khiết trong một hệ thống. Trong lý thuyết thông tin, nó đề cập đến sự không tinh khiết trong một nhóm các ví dụ. Độ lợi thông tin là sự giảm entropy. Độ lợi thông tin tính toán sự khác biệt giữa entropy trước khi phân chia và entropy trung bình sau khi phân chia tập dữ liệu dựa trên các giá trị thuộc tính được cung cấp. Thuật toán cây quyết định ID3 (Iterative Dichotomiser) sử dụng độ lợi thông tin.



Độ lợi thông tin được tính toán bằng công thức sau:

Trong đó:

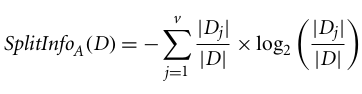
* Info(D) là lượng thông tin trung bình cần thiết để xác định nhãn lớp của một tuple trong D.
* |Dj|/|D| đóng vai trò trọng số của phân vùng thứ j.
* InfoA(D) là thông tin dự kiến cần thiết để phân loại một tuple từ D dựa trên sự phân vùng bởi A.

Thuộc tính A với độ lợi thông tin cao nhất, Gain(A), được chọn làm thuộc tính phân chia tại nút N().

Độ lợi thông tin bị thiên vị cho thuộc tính có nhiều kết quả. Nó có nghĩa là nó thích thuộc tính có số lượng lớn các giá trị riêng biệt. Ví dụ, hãy xem xét một thuộc tính có định danh duy nhất, chẳng hạn như customer\_ID, có info(D) bằng không vì phân vùng thuần túy. Điều này tối đa hóa độ lợi thông tin và tạo ra sự phân vùng vô ích.

C4.5, một cải tiến của ID3, sử dụng phần mở rộng cho độ lợi thông tin được gọi là tỷ lệ lợi ích. Tỷ lệ lợi ích xử lý vấn đề thiên vị bằng cách chuẩn hóa độ lợi thông tin bằng cách sử dụng Split Info. Java triển khai thuật toán C4.5 được gọi là J48, có sẵn trong công cụ khai thác dữ liệu WEKA.

Tỷ lệ lợi ích được tính toán bằng công thức sau:



Trong đó:

* |Dj|/|D| đóng vai trò trọng số của phân vùng thứ j.
* v là số lượng giá trị riêng biệt trong thuộc tính A.

Thuộc tính có tỷ lệ lợi ích cao nhất được chọn làm thuộc tính phân chia (Nguồn).

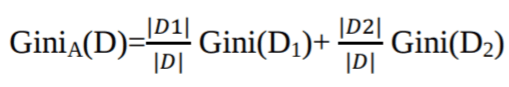
Một thuật toán cây quyết định khác là CART (Classification and Regression Tree) sử dụng phương pháp Gini để tạo điểm phân chia.

Chỉ số Gini được tính toán bằng công thức sau:



Trong đó pi là xác suất mà một tuple trong D thuộc về lớp Ci.

Chỉ số Gini xem xét sự phân chia nhị phân cho từng thuộc tính. Bạn có thể tính toán tổng có trọng số của sự không tinh khiết của mỗi phân vùng. Nếu sự phân chia nhị phân trên thuộc tính A phân vùng dữ liệu D thành D1 và D2, thì chỉ số Gini của D là:



Trong trường hợp của thuộc tính có giá trị rời rạc, tập con cung cấp chỉ số gini tối thiểu cho tập con được chọn làm thuộc tính phân chia. Trong trường hợp của các thuộc tính có giá trị liên tục, chiến lược là chọn từng cặp giá trị liền kề làm điểm phân chia có thể, và một điểm có chỉ số gini nhỏ hơn được chọn làm điểm phân chia.



Thuộc tính có chỉ số Gini tối thiểu được chọn làm thuộc tính phân chia.

#### **5.5 K-Nearest Neighbors**

Thuật toán K-Nearest Neighbors (K-NN) là một thuật toán machine learning thuộc lớp bài toán giám sát (supervised learning), được sử dụng chủ yếu cho các tác vụ phân loại (classification) và dự đoán (regression). Ý tưởng cơ bản của K-NN là dự đoán nhãn của một điểm dữ liệu mới dựa trên nhãn của các điểm dữ liệu "k gần nhất" trong tập dữ liệu đã được đào tạo.

Cụ thể, quy trình hoạt động của thuật toán KNN như sau:

**Chọn giá trị K:** Xác định giá trị K, tức là số lượng láng giềng gần nhất mà thuật toán sẽ xem xét để đưa ra dự đoán cho một điểm dữ liệu mới.

**Tính khoảng cách:** Sử dụng một phép đo khoảng cách (thường là khoảng cách Euclidean) để đo độ tương đồng giữa điểm dữ liệu mới và các điểm dữ liệu trong tập huấn luyện.

**Xác định láng giềng gần nhất:** Chọn ra K điểm dữ liệu gần nhất với điểm dữ liệu mới dựa trên khoảng cách tính được.

**Đa số phiếu bầu:** Đếm số lượng láng giềng thuộc mỗi nhóm (đối với bài toán phân loại) hoặc tính trung bình giá trị (đối với bài toán dự đoán). Nhãn hoặc giá trị của điểm dữ liệu mới sẽ được dự đoán dựa trên đa số phiếu bầu hoặc giá trị trung bình này.

Thuật toán KNN có thể hoạt động hiệu quả đối với các tập dữ liệu lớn có cấu trúc tương đối đơn giản. Tuy nhiên, nó có nhược điểm khi sử dụng trên các tập dữ liệu lớn hoặc có nhiều chiều (nhiều đặc trưng), vì đòi hỏi nhiều thời gian tính toán khi cần xem xét tất cả các láng giềng gần nhất.

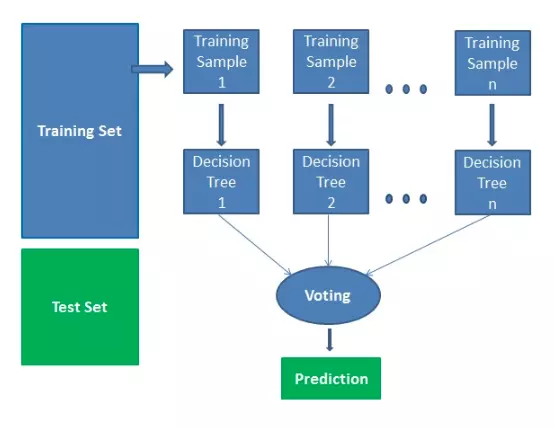
#### **5.6 Random Forest Classifier**

Random Forests là thuật toán học có giám sát (supervised learning). Nó có thể được sử dụng cho cả phân lớp và hồi quy. Nó cũng là thuật toán linh hoạt và dễ sử dụng nhất. Một khu rừng bao gồm cây cối. Người ta nói rằng càng có nhiều cây thì rừng càng mạnh. Random forests tạo ra cây quyết định trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu. Nó cũng cung cấp một chỉ báo khá tốt về tầm quan trọng của tính năng. Random forests có nhiều ứng dụng, chẳng hạn như công cụ đề xuất, phân loại hình ảnh và lựa chọn tính năng. Nó có thể được sử dụng để phân loại các ứng viên cho vay trung thành, xác định hoạt động gian lận và dự đoán các bệnh. Nó nằm ở cơ sở của thuật toán Boruta, chọn các tính năng quan trọng trong tập dữ liệu.

Về mặt kỹ thuật, nó là một phương pháp tổng hợp (dựa trên cách tiếp cận phân chia và chinh phục) của các cây quyết định được tạo ra trên một tập dữ liệu được chia ngẫu nhiên. Bộ sưu tập phân loại cây quyết định này còn được gọi là rừng. Cây quyết định riêng lẻ được tạo ra bằng cách sử dụng chỉ báo chọn thuộc tính như tăng thông tin, tỷ lệ tăng và chỉ số Gini cho từng thuộc tính. Mỗi cây phụ thuộc vào một mẫu ngẫu nhiên độc lập. Trong bài toán phân loại, mỗi phiếu bầu chọn và lớp phổ biến nhất được chọn là kết quả cuối cùng. Trong trường hợp hồi quy, mức trung bình của tất cả các kết quả đầu ra của cây được coi là kết quả cuối cùng. Nó đơn giản và mạnh mẽ hơn so với các thuật toán phân loại phi tuyến tính khác.

Nó hoạt động theo bốn bước:

* Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
* Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
* Hãy bỏ phiếu cho mỗi kết quả dự đoán.
* Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.



Ưu điểm: Random forests được coi là một phương pháp chính xác và mạnh mẽ vì số cây quyết định tham gia vào quá trình này. Nó không bị vấn đề overfitting. Lý do chính là nó mất trung bình của tất cả các dự đoán, trong đó hủy bỏ những thành kiến. Thuật toán có thể được sử dụng trong cả hai vấn đề phân loại và hồi quy. Random forests cũng có thể xử lý các giá trị còn thiếu. Có hai cách để xử lý các giá trị này: sử dụng các giá trị trung bình để thay thế các biến liên tục và tính toán mức trung bình gần kề của các giá trị bị thiếu. Bạn có thể nhận được tầm quan trọng của tính năng tương đối, giúp chọn các tính năng đóng góp nhiều nhất cho trình phân loại. Nhược điểm: Random forests chậm tạo dự đoán bởi vì nó có nhiều cây quyết định. Bất cứ khi nào nó đưa ra dự đoán, tất cả các cây trong rừng phải đưa ra dự đoán cho cùng một đầu vào cho trước và sau đó thực hiện bỏ phiếu trên đó. Toàn bộ quá trình này tốn thời gian. Mô hình khó hiểu hơn so với cây quyết định, nơi bạn có thể dễ dàng đưa ra quyết định bằng cách đi theo đường dẫn trong cây.

#### **5.7 AdaBoost Classifier**

AdaBoost (Adaptive Boosting) là một thuật toán machine learning được sử dụng chủ yếu trong bài toán phân loại (classification). Nó thuộc họ thuật toán "boosting," một kỹ thuật mà nó cố gắng xây dựng một mô hình mạnh từ nhiều mô hình yếu (weak learners).

Ý tưởng chính của AdaBoost là tập trung vào việc cải thiện hiệu suất của mô hình bằng cách tăng cường sức mạnh của các weak learners, sau đó kết hợp chúng lại để tạo thành một mô hình tổng cộng mạnh mẽ. Các weak learners thường là các mô hình đơn giản và nhanh chóng đào tạo, chẳng hạn như cây quyết định có độ sâu thấp.

Dưới đây là cách AdaBoost thực hiện quá trình đào tạo:

**Khởi tạo trọng số:** Mỗi điểm dữ liệu trong tập huấn luyện được gán một trọng số khởi tạo.

**Đào tạo weak learner:** Một weak learner (thường là một cây quyết định) được đào tạo trên tập dữ liệu hiện tại, với trọng số được áp dụng để tập trung vào các điểm dữ liệu khó phân loại.

**Đánh giá hiệu suất của weak learner:** Weak learner được sử dụng để dự đoán trên toàn bộ tập huấn luyện, và hiệu suất của nó được đánh giá.

**Cập nhật trọng số:** Trọng số của các điểm dữ liệu được cập nhật dựa trên hiệu suất của weak learner. Các điểm dữ liệu được phân loại sai sẽ có trọng số tăng lên để chú ý đến trong các vòng lặp tiếp theo.

**Lặp lại quá trình:** Quá trình từ bước 2 đến bước 4 được lặp lại cho một số lượng vòng lặp (số lượng weak learners được xác định trước).

**Kết hợp các weak learners:** Các weak learners được kết hợp lại để tạo thành một mô hình tổng cộng. Các weak learners có hiệu suất cao hơn sẽ được đánh trọng số cao hơn trong dự đoán cuối cùng.

AdaBoost thường là một trong những thuật toán phân loại mạnh mẽ và linh hoạt, có thể tốt trên nhiều loại dữ liệu.

#### **5.8 Bagging Classifier**

Bagging Classifier là một kỹ thuật trong Machine Learning, chủ yếu được sử dụng trong bài toán phân loại. "Bagging" viết tắt của cụm từ "Bootstrap Aggregating". Kỹ thuật này hoạt động bằng cách xây dựng nhiều mô hình dự đoán độc lập và sau đó kết hợp kết quả của chúng để tạo ra dự đoán cuối cùng.

Cụ thể, Bagging Classifier tạo ra nhiều tập con dữ liệu từ dữ liệu huấn luyện bằng cách lấy ngẫu nhiên với việc thay thế (bootstrap sampling). Sau đó, nó huấn luyện một mô hình dự đoán trên mỗi tập con này. Các mô hình này có thể là bất kỳ thuật toán học máy nào, chẳng hạn như Decision Trees, Random Forests, hoặc SVMs.

Cuối cùng, khi cần dự đoán kết quả cho một dữ liệu mới, Bagging Classifier kết hợp kết quả dự đoán từ các mô hình con thông qua việc lấy trung bình (đối với dự đoán hồi quy) hoặc sử dụng phiếu bầu (đối với dự đoán phân loại) để tạo ra dự đoán cuối cùng. Kỹ thuật này thường giúp cải thiện tính tổng quát và giảm phương sai của mô hình.

#### **5.9 Extra Trees Classifier**

Extra Trees Classifier là một thuật toán học máy thuộc loại "ensemble learning", giống như Random Forests, được sử dụng cho bài toán phân loại. "Extra Trees" là viết tắt của "Extremely Randomized Trees".

Thuật toán này tương tự như Random Forest, nhưng có một số điểm khác biệt quan trọng trong cách mà nó tạo ra các cây quyết định.

* Randomization trong việc chọn đặc trưng và điểm cắt: Trong Extra Trees, không chỉ có việc sử dụng mẫu ngẫu nhiên từ tập dữ liệu để xây dựng cây như trong Random Forest, mà còn áp dụng việc chọn ngẫu nhiên các đặc trưng để chia nhánh cây quyết định (feature splitting) và chọn ngẫu nhiên điểm cắt (thresholds). Điều này tạo ra sự đa dạng lớn hơn trong các cây quyết định, đồng thời giảm thiểu sự overfitting.
* Cắt tỉa cây một cách ngẫu nhiên: Extra Trees thường không thực hiện việc cắt tỉa (pruning) trên cây quyết định như các thuật toán khác. Thay vào đó, nó cho phép cây phát triển đầy đủ và sau đó sử dụng chúng mà không lo lắng về việc cắt tỉa để giảm overfitting.

Với việc kết hợp các cây quyết định được tạo ra một cách ngẫu nhiên và sự đa dạng cao, Extra Trees Classifier thường có khả năng tìm ra các mô hình tổng hợp có hiệu suất tốt và khá linh hoạt khi xử lý nhiễu và dữ liệu không hoàn hảo.

#### **5.10 Gradient Boosting Classifier**

Gradient Boosting Classifier là một thuật toán học máy thuộc lớp các phương pháp tăng cường (boosting methods) được sử dụng cho bài toán phân loại. Được phát triển dựa trên ý tưởng xây dựng các mô hình dự đoán theo cách tuần tự để cố gắng cải thiện hiệu suất dự đoán.

Cách hoạt động chính của Gradient Boosting Classifier là:

* Xây dựng cây yếu (weak learners): Thuật toán bắt đầu bằng việc xây dựng một cây quyết định đơn giản (cây yếu) dự đoán mục tiêu. Cây yếu này thường là một decision tree với mức độ phức tạp thấp, thường chỉ có một vài tầng cây.
* Tối ưu hóa theo độ lỗi: Sau khi cây yếu được tạo ra, Gradient Boosting sẽ tìm cách điều chỉnh dự đoán của mô hình để giảm sai số (loss) giữa dự đoán hiện tại và giá trị thực tế. Thuật toán sẽ tập trung vào việc dự đoán những điểm dữ liệu mà cây trước đó dự đoán sai.
* Tích hợp các cây dự đoán: Sau khi tính toán sự khác biệt giữa dự đoán và giá trị thực tế, Gradient Boosting tiếp tục xây dựng một cây mới để dự đoán sai số này. Quá trình này tiếp tục tuần tự với mục tiêu là giảm thiểu sai số toàn cục.
* Kết hợp các mô hình yếu thành một mô hình mạnh: Gradient Boosting kết hợp tất cả các cây yếu dự đoán lại với nhau theo một cách thông minh để tạo ra một mô hình mạnh hơn. Mô hình này là sự tổng hợp của các cây yếu, với mỗi cây cố gắng cải thiện những điểm yếu của các cây trước đó.

Gradient Boosting Classifier thường tạo ra các mô hình mạnh có khả năng dự đoán tốt và hiệu quả trên nhiều loại dữ liệu khác nhau. Điều này làm cho nó trở thành một trong những phương pháp phổ biến và mạnh mẽ trong Machine Learning.

#### **5.11 XGBoost Classifier**

XGBoost (Extreme Gradient Boosting) Classifier là một thuật toán Machine Learning thuộc lớp các phương pháp tăng cường (boosting methods), được phát triển bởi Tianqi Chen. Được biết đến với hiệu suất cao và khả năng tốt trong việc xử lý các bài toán phân loại và hồi quy trên các tập dữ liệu lớn.

Các đặc điểm chính của XGBoost Classifier bao gồm:

Tối ưu hóa hàm mất mát (Loss function): XGBoost tập trung vào tối ưu hóa hàm mất mát bằng cách sử dụng phương pháp Gradient Descent. Nó thực hiện việc cải thiện mô hình bằng cách di chuyển theo hướng giảm thiểu hàm mất mát, từ đó tăng cường khả năng dự đoán.

Regularization: XGBoost cũng có khả năng xử lý overfitting bằng cách sử dụng regularization term trong quá trình tối ưu hóa mô hình. Điều này giúp kiểm soát độ phức tạp của mô hình và ngăn chặn việc quá mức hóa dữ liệu huấn luyện.

Cây quyết định (Decision Trees) tối ưu hóa: XGBoost xây dựng cây quyết định tối ưu hóa bằng cách sử dụng thuật toán Gradient Boosting, tập trung vào việc tạo ra các cây quyết định có chiều sâu thấp mà vẫn giữ được khả năng tổng quát hóa.

Parallel Computing: Thuật toán này cũng được tối ưu hóa để thực hiện tính toán song song (parallel computing), làm tăng tốc độ huấn luyện và dự đoán trên dữ liệu lớn.

XGBoost đã trở thành một trong những thuật toán phổ biến và mạnh mẽ trong cộng đồng Machine Learning, đặc biệt là khi xử lý các bài toán phức tạp và dữ liệu lớn. Nó thường được sử dụng trong các cuộc thi Machine Learning và các ứng dụng thực tế do hiệu suất và khả năng áp dụng rộng rãi của nó.

### **6. Xây dựng mô hình và đánh giá.**

#### **6.1** **Phân tách tập dữ liệu**

Như thường lệ đối với các vấn đề về học máy có giám sát, chúng ta cần một tập dữ liệu huấn luyện để huấn luyện mô hình của mình và một tập dữ liệu thử nghiệm để đánh giá mô hình. Vì vậy, chúng tôi sẽ chia tập dữ liệu của mình một cách ngẫu nhiên thành hai phần, một phần để huấn luyện và phần còn lại để thử nghiệm. Để làm được điều đó, chúng ta sẽ sử dụng một hàm khác từ Scikit-Learn có tên là train\_test\_split()

Chúng em đã chỉ định kích thước của bộ thử nghiệm là 20% của toàn bộ tập dữ liệu. Điều này để lại 80% cho tập dữ liệu huấn luyện. Bây giờ chúng em có bốn tập hợp con: x\_train, x\_test, y\_train và y\_test. Sau này, chúng em sẽ sử dụng x\_train và y\_train để huấn luyện mô hình của mình, đồng thời sử dụng x\_test và y\_test để kiểm tra và đánh giá mô hình. x\_train và x\_test đại diện cho các tính năng (dự đoán); y\_train và y\_test đại diện cho mục tiêu. Từ giờ trở đi, chúng em sẽ gọi x\_train và y\_train là tập dữ liệu huấn luyện và x\_test và y\_test là tập dữ liệu thử nghiệm.

#### **6.2 Modeling Approach**

Đối với mỗi kỹ thuật được đề cập trong phần trước, chúng em sẽ làm theo các bước sau để xây dựng mô hình:

* Chọn một thuật toán thực hiện kỹ thuật tương ứng
* Tìm kiếm tổ hợp tham số hiệu quả cho thuật toán đã chọn
* Tạo mô hình bằng cách sử dụng các tham số tìm thấy
* Huấn luyện (khớp) mô hình trên tập dữ liệu huấn luyện
* Kiểm tra mô hình trên tập dữ liệu thử nghiệm và nhận kết quả

#### **6.3 Mô hình**

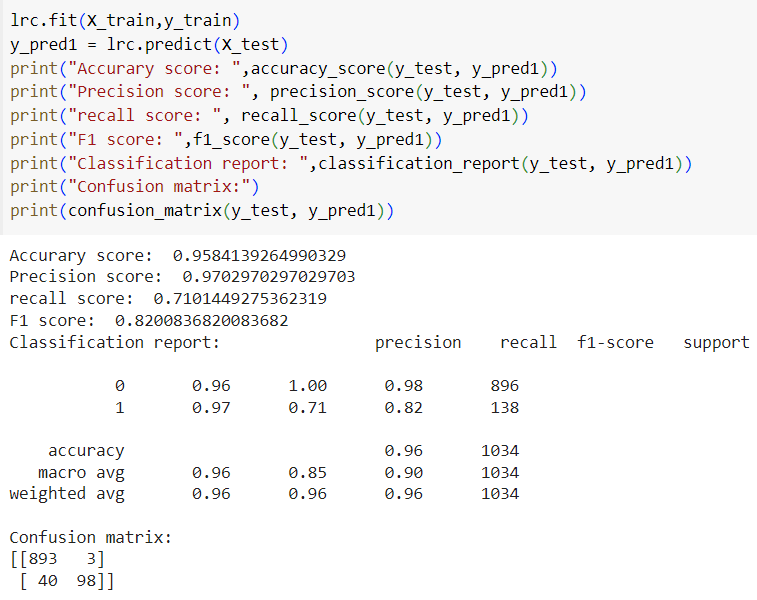
##### **6.3.1 Logistic Regression**

Đối với Logistic Regression (LRC), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp.

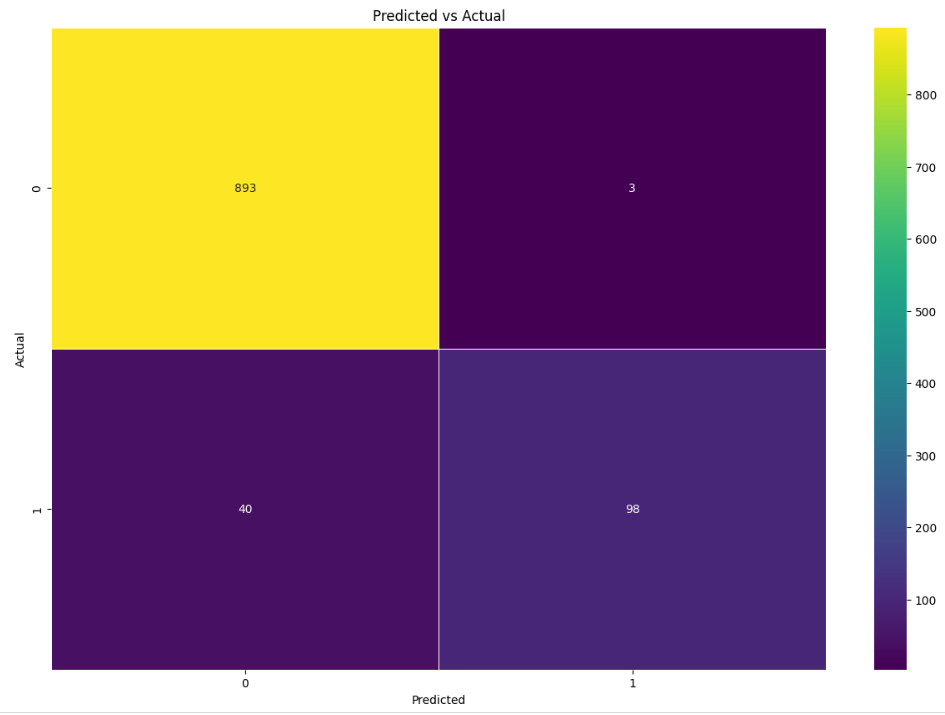
Mô hình Logistic Regression có cú pháp như sau:



Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác:



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.



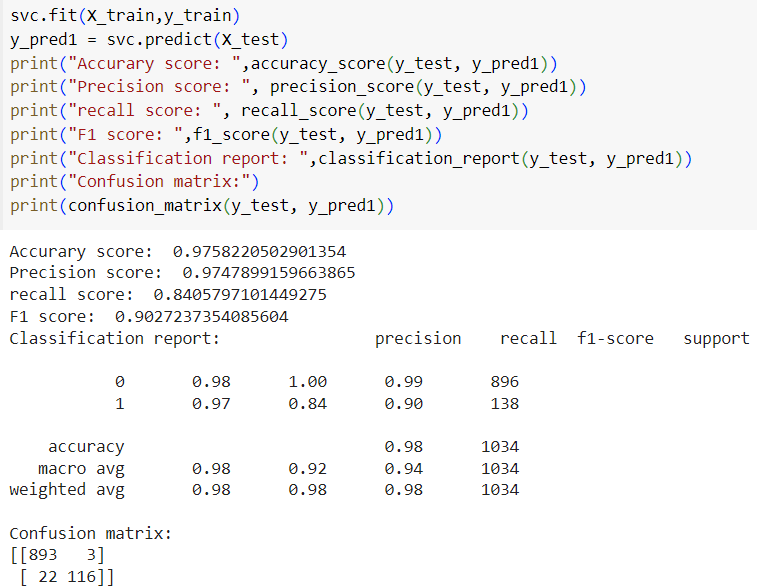
##### **6.3.2 Support Vector Machine**

Đối với Vector Machine (SVC), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp.

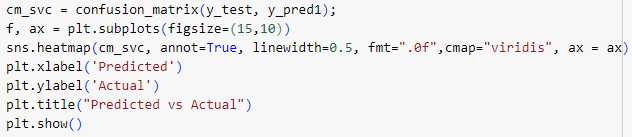
Mô hình Support Vector Machine có cú pháp như sau:

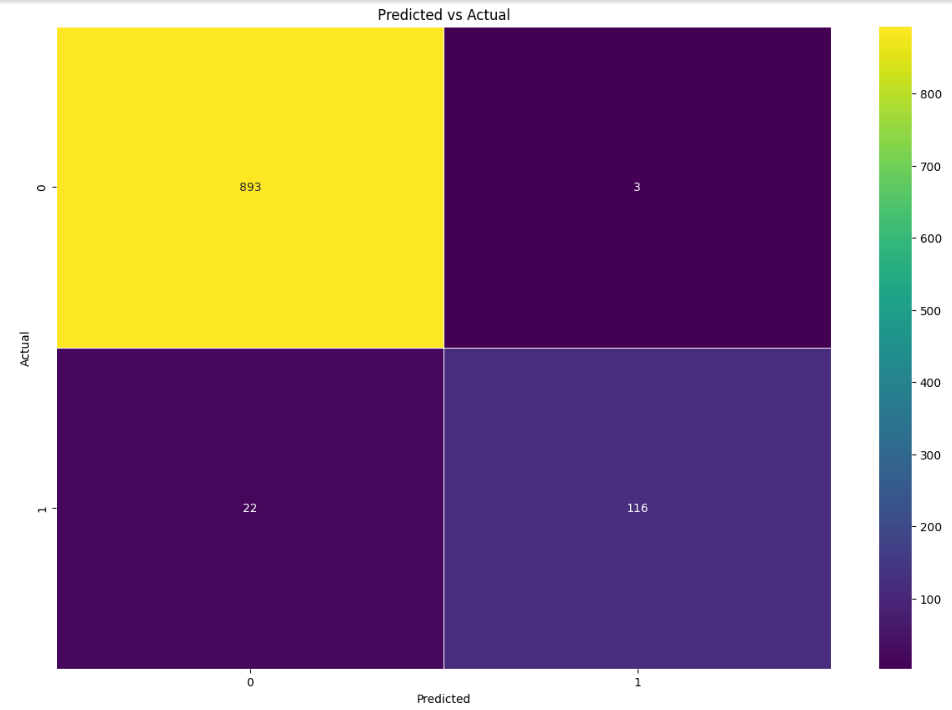


Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác:



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.





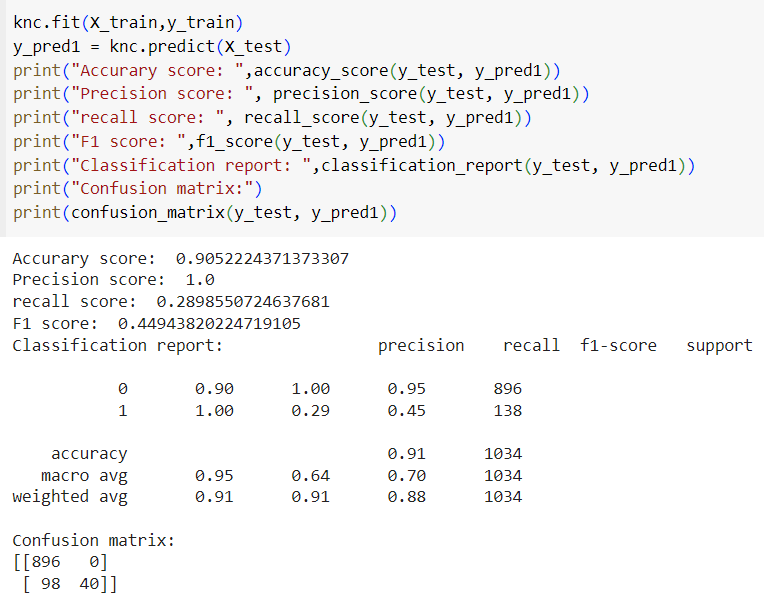
##### **6.3.3 K-Nearest Neighbors**

Đối với K-Nearest Neighbors (KNN), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp.

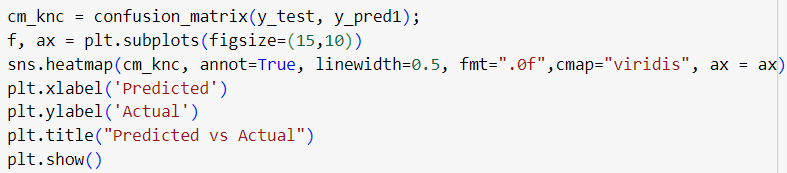
Mô hình The K-Nearest Neighbors có cú pháp như sau:

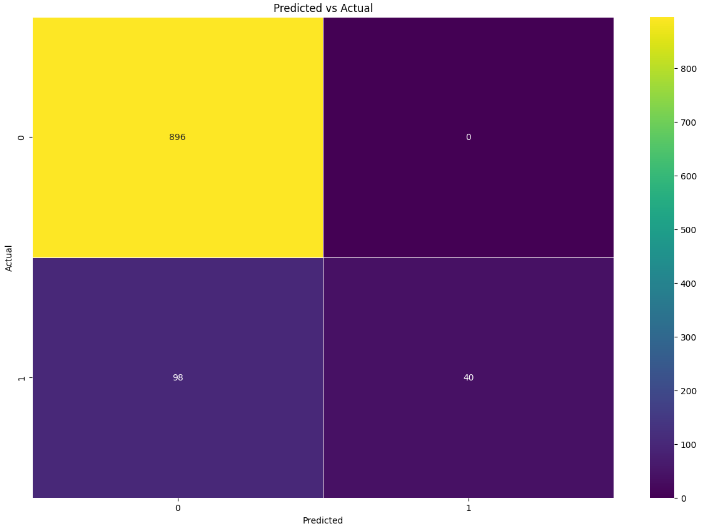


Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác:



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.





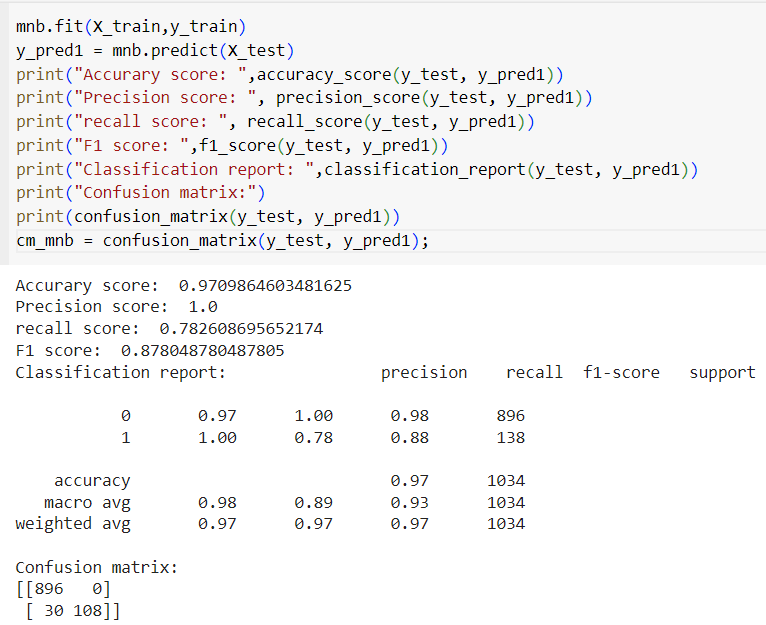
##### **6.3.4 Multinomial Naive Bayes**

Đối với Multinomial Naive Bayes (MNB), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp.

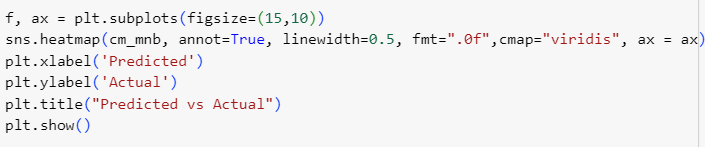
Mô hình Multinomial Naive Bayes có cú pháp như sau:

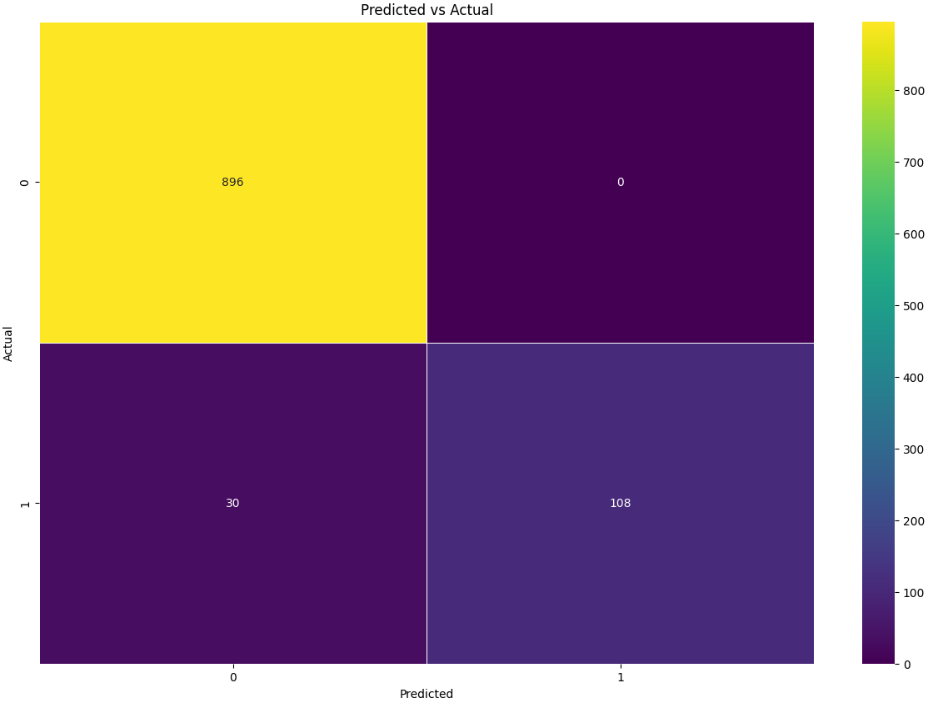


Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.





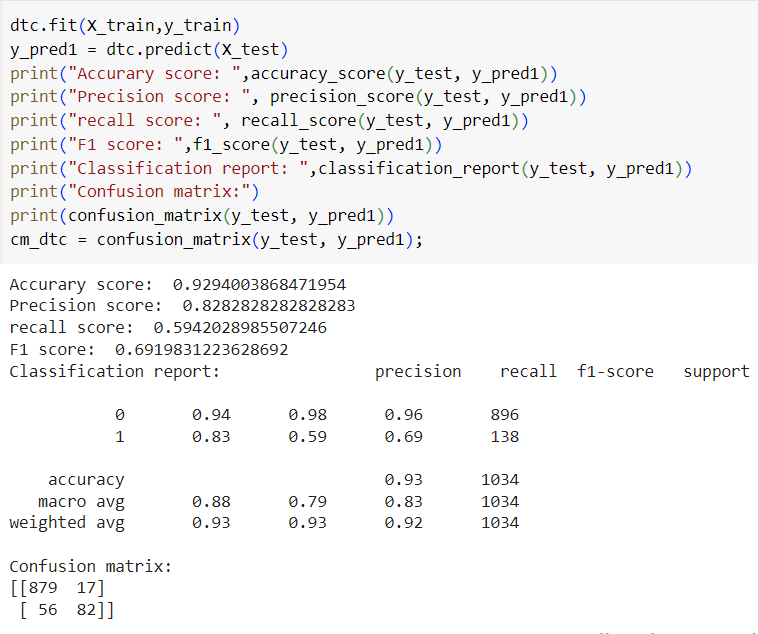
##### **6.3.5 Decision Tree Classifier**

Đối với decision Tree Classifier (DTC), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp.,

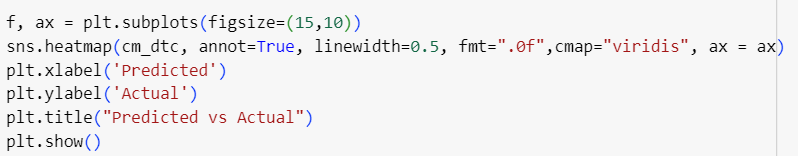
Mô hình hồi quy logistic có cú pháp như sau:

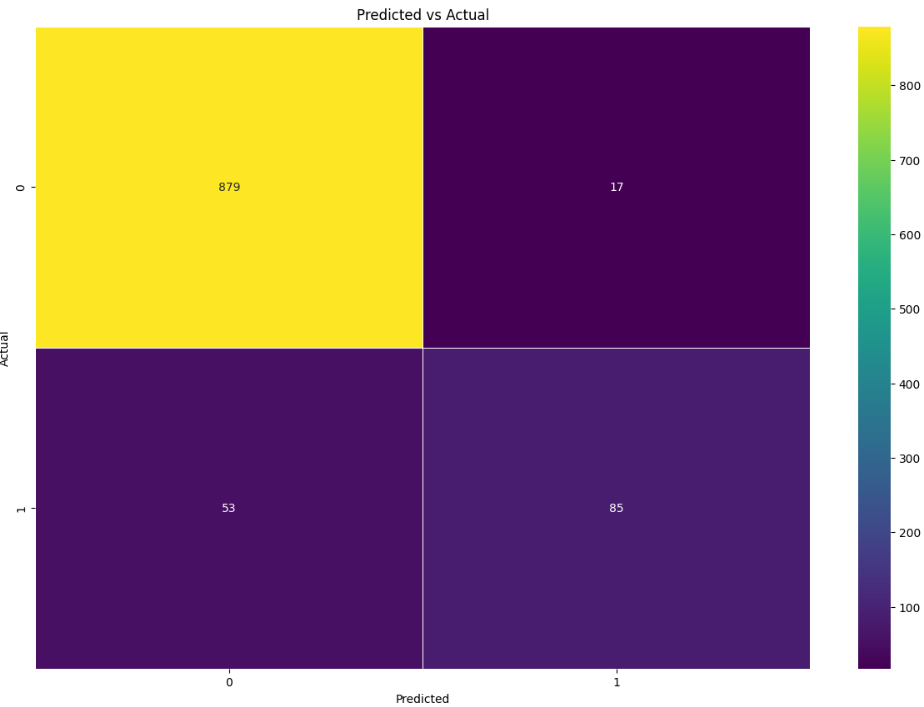


Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác:



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.





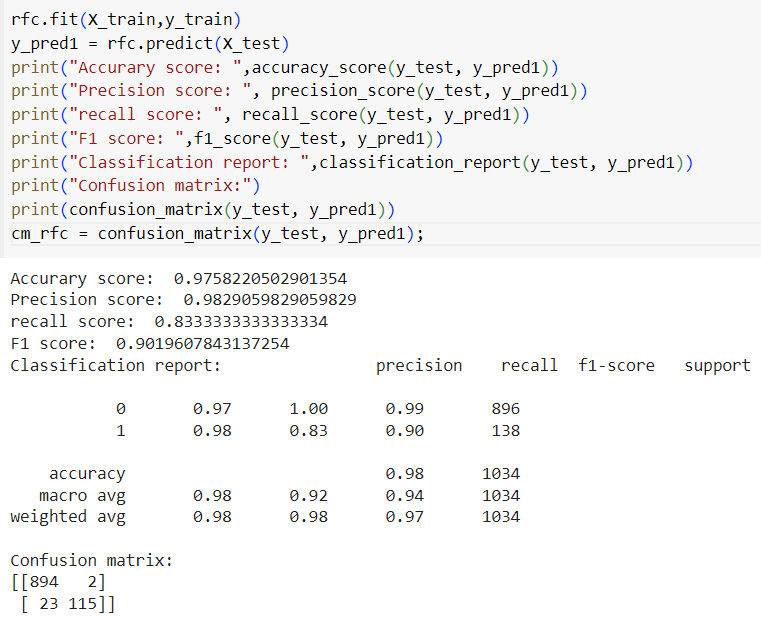
##### **6.3.6 Random Forest Classifier**

Đối với Forest Classifier (RFC), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp..

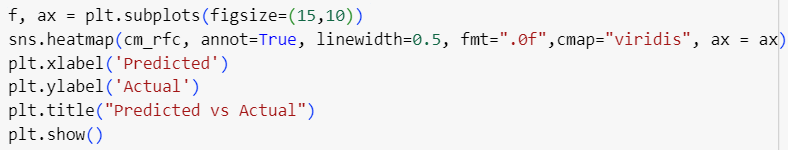
Mô hình Random Forest Classifier có cú pháp như sau:

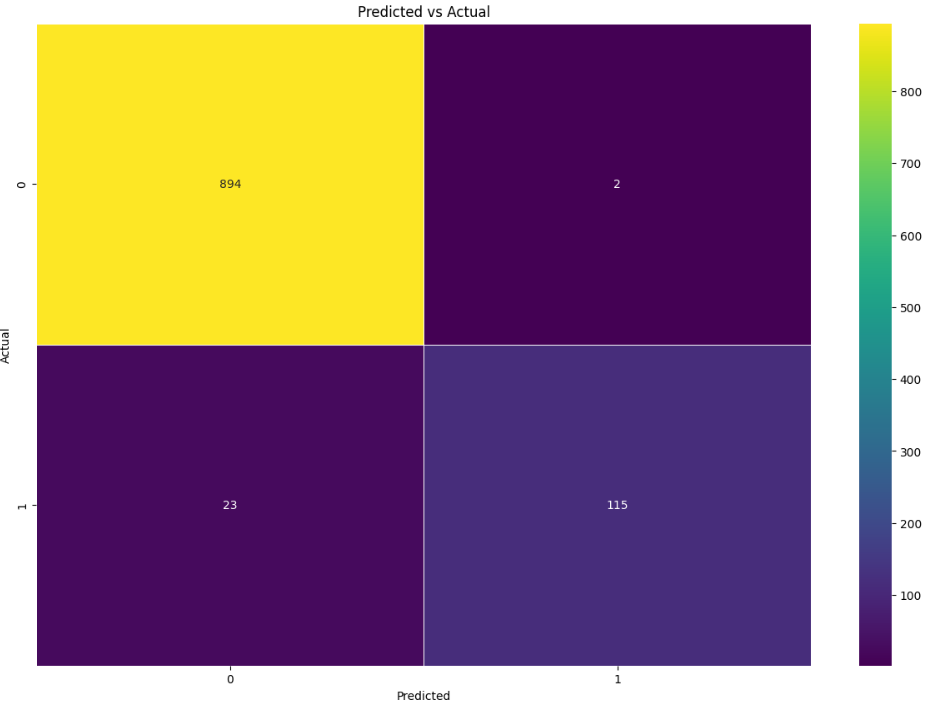


Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác:



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.





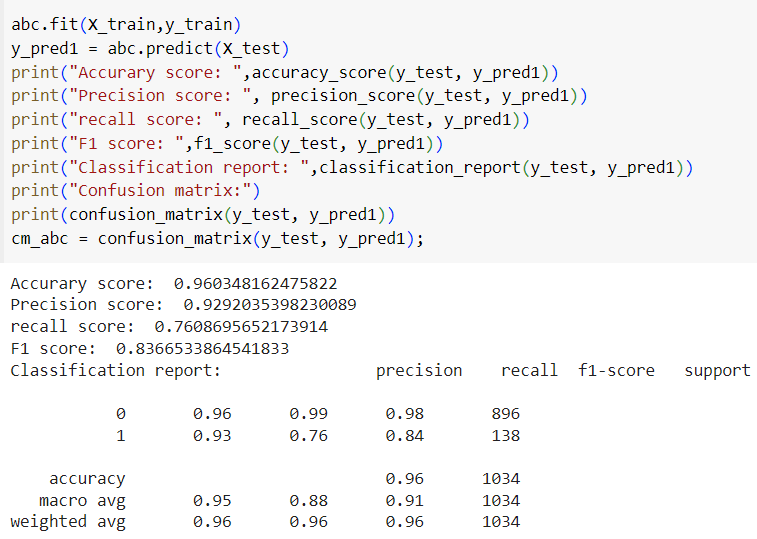
##### **6.3.7 AdaBoost Classifier**

Đối với Bagging AdaBoost Classifier (ABC), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp.

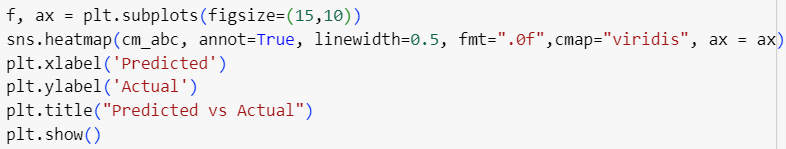
Mô hình Bagging AdaBoost Classifier có cú pháp như sau:

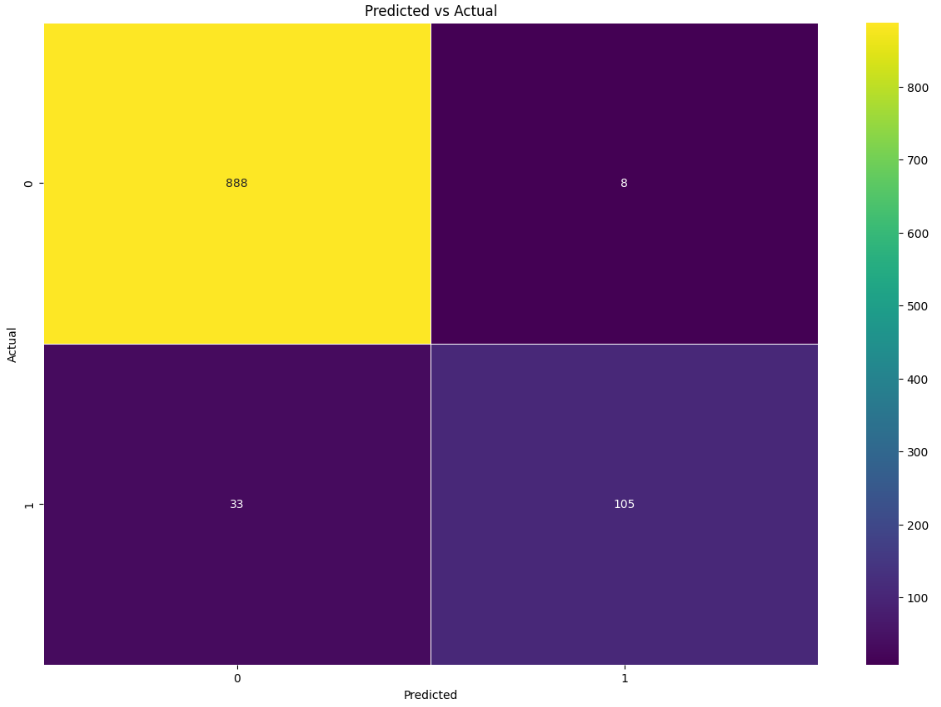


Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác:



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.





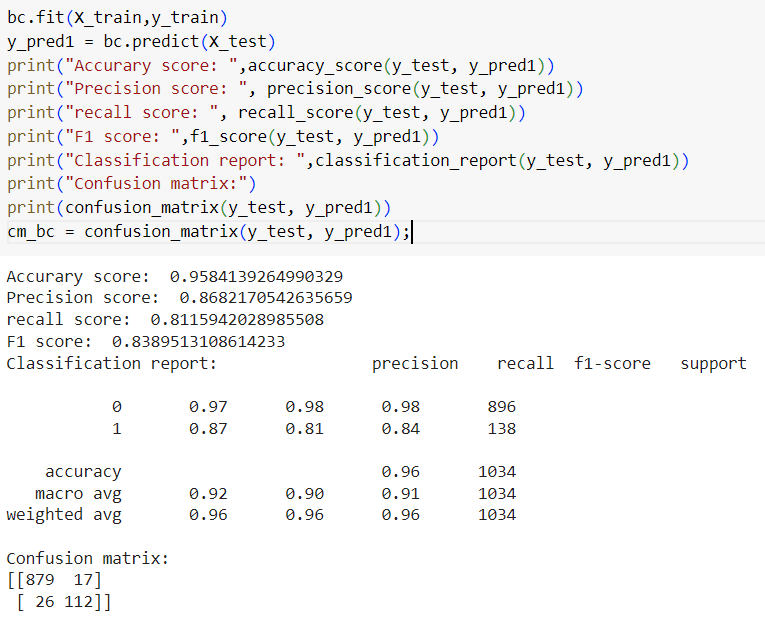
##### **6.3.8 Bagging Classifier**

Đối với Bagging Classifier (BC), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp.

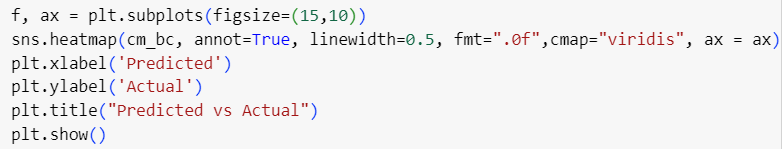
Mô hình Bagging Classifier có cú pháp như sau:

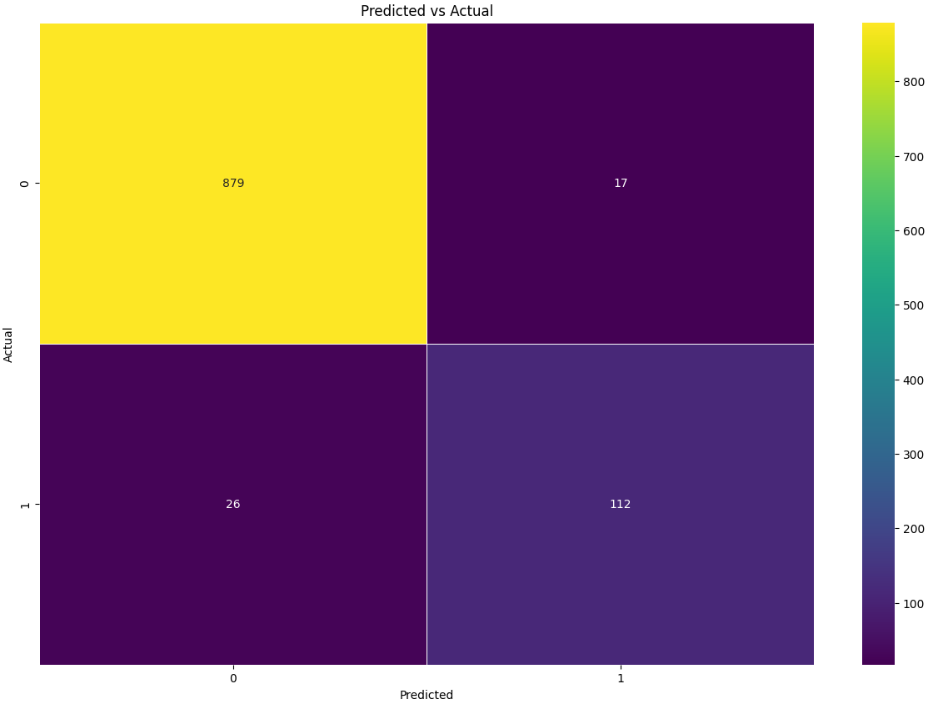


Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác:



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.





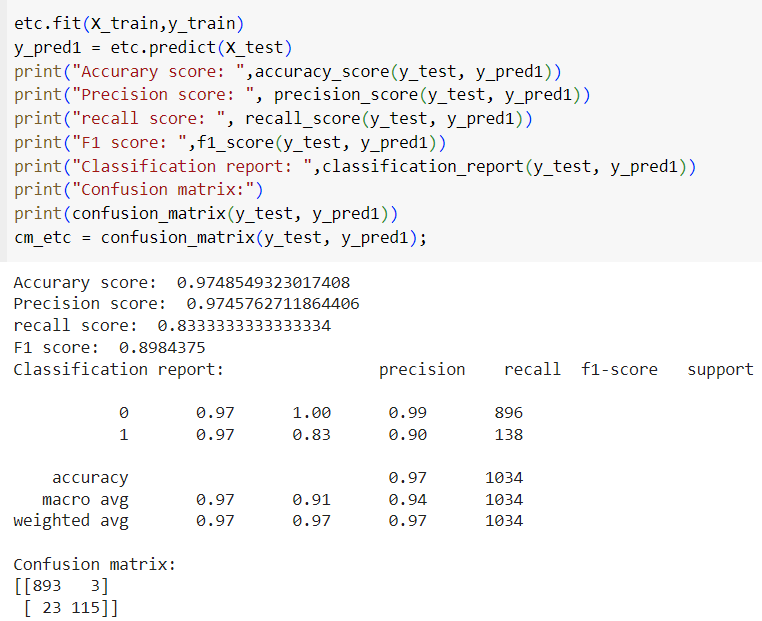
##### **6.3.9 Extra Trees Classifier**

Đối với Extra Trees Classifier (ETC), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp.

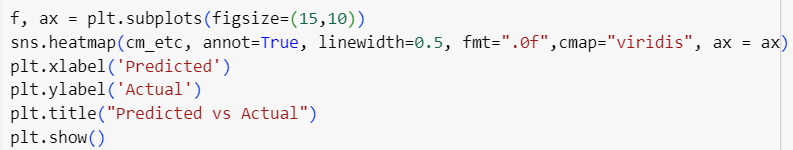
Mô hình Extra Trees Classifier có cú pháp như sau:

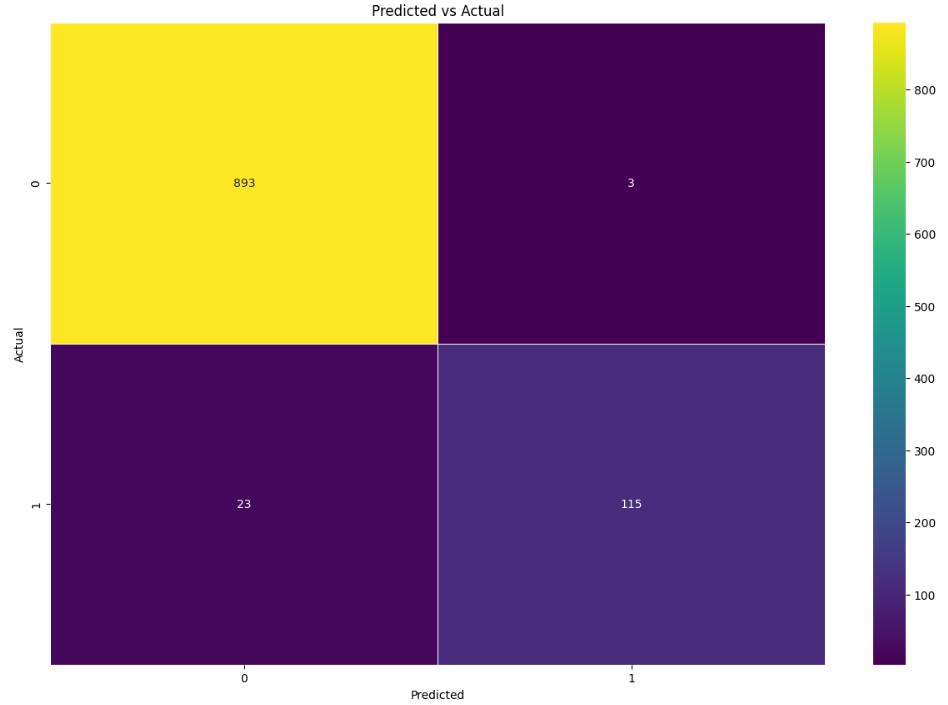


Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác:



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.





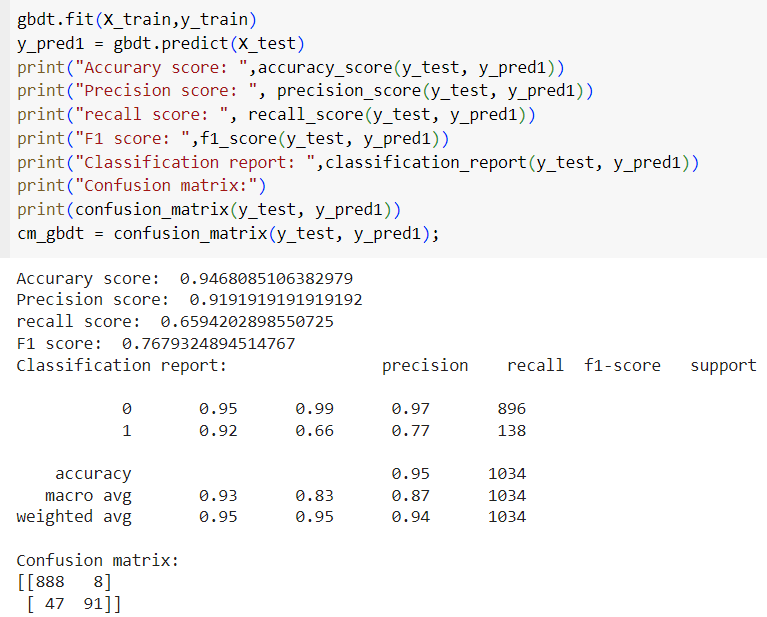
##### **6.3.10 Gradient Boosting Classifier**

Đối với Gradient Boosting Classifier (GBC), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp.

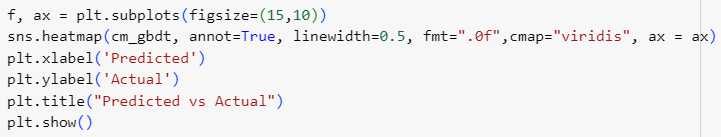
Mô hình hồi Gradient Boosting Classifier có cú pháp như sau:

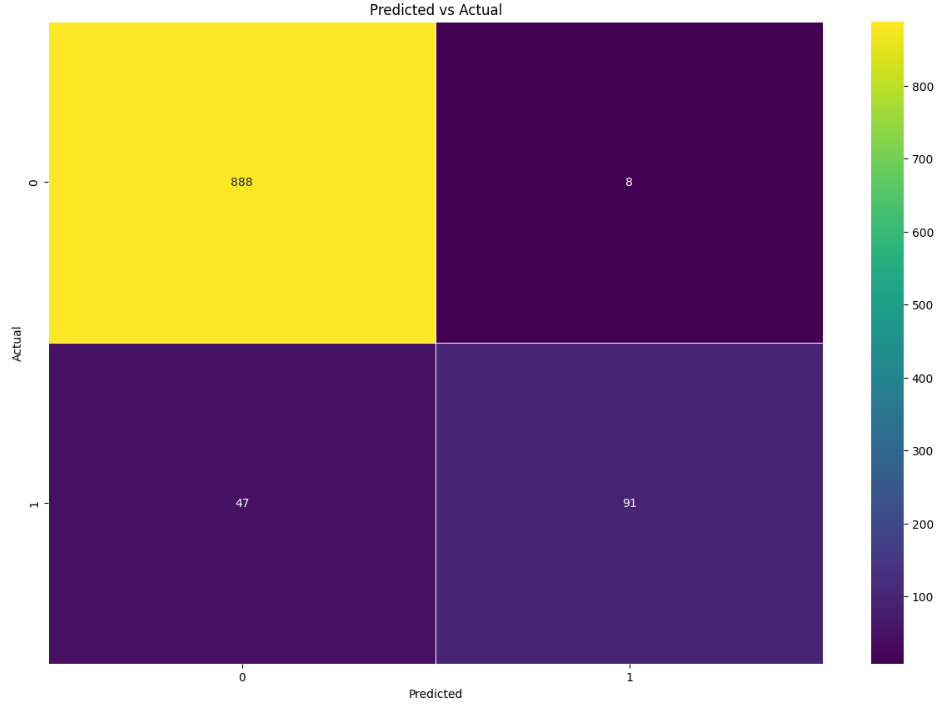


Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác:



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.





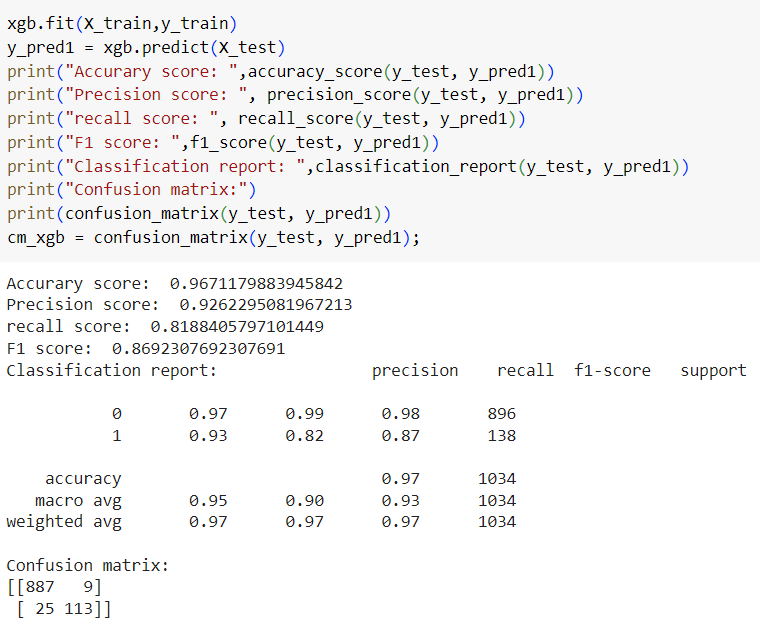
##### **6.3.11 XGBoost Classifier**

Đối với XGBoost Classifier (XGB), chúng em sẽ sử dụng triển khai do gói Scikit-Learn cung cấp.

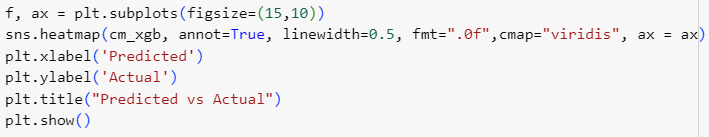
Mô hình The XGBoost Classifier có cú pháp như sau:

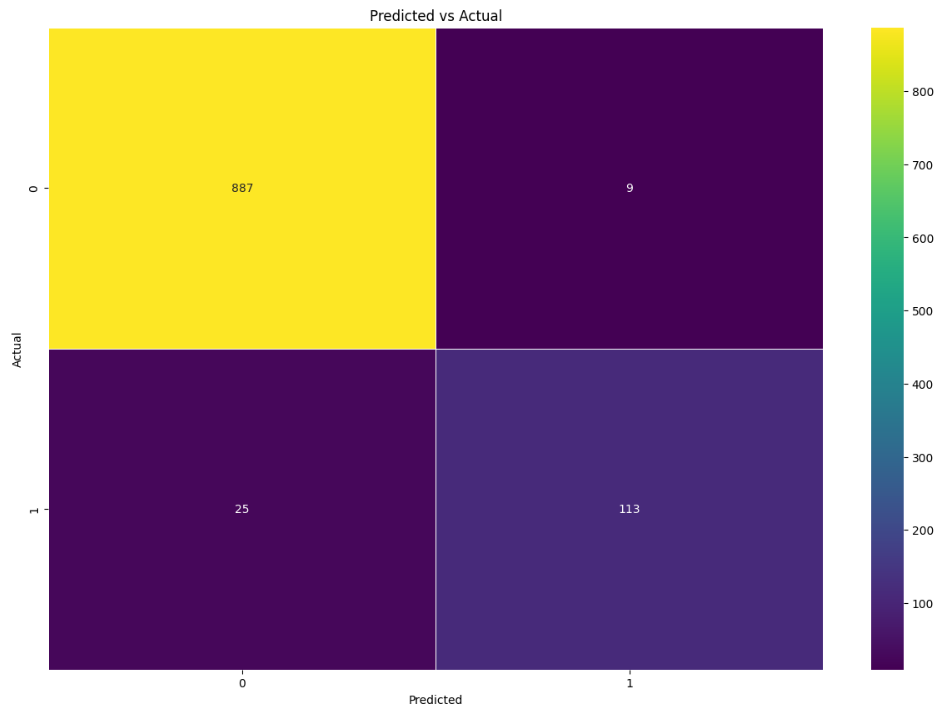


Tiếp theo, chúng em huấn luyện mô hình của mình bằng cách sử dụng tập huấn luyện (x\_train và y\_train). Sau đó, chúng em kiểm tra mô hình của mình trên x\_test. Cuối cùng, chúng em đánh giá hiệu suất của mô hình bằng cách lấy điểm chính xác:



Sau đó, chúng em sẽ vẽ một bản đồ nhiệt để xem xét ma trận nhầm lẫn giữa Giá trị Dự đoán và Giá trị Thực tế.



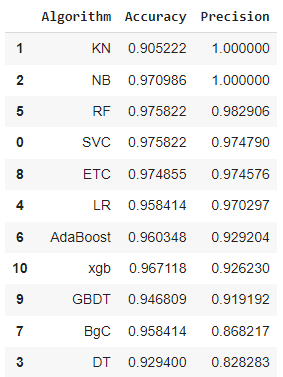


### **7. Kết luận.**

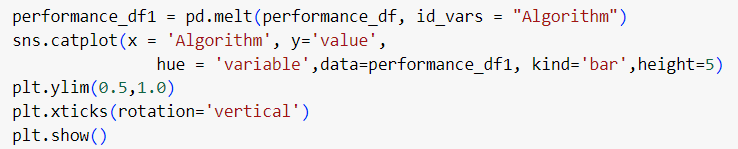
#### **7.1. Nhận xét và đánh giá**

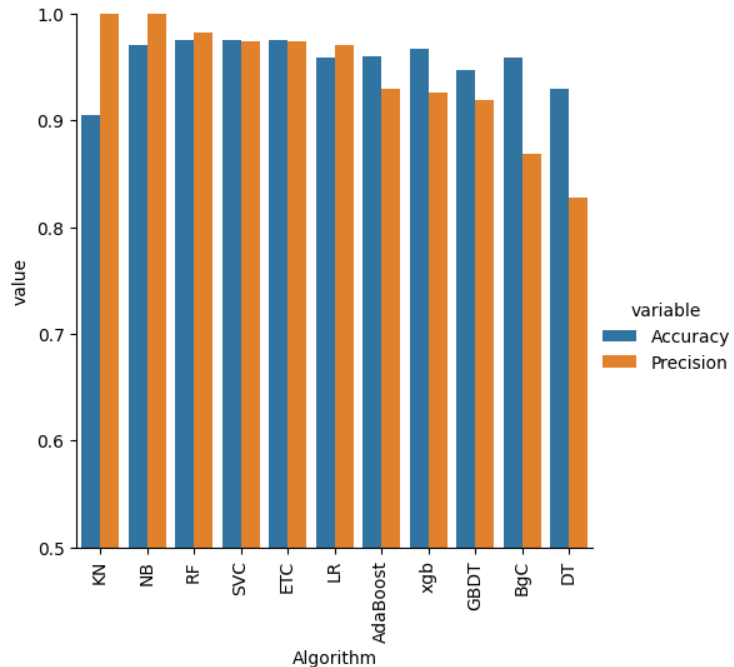
Trong phần trước, chúng em đã tạo nhiều mô hình: đối với mỗi mô hình, chúng em đã huấn luyện (lắp) mô hình đó vào dữ liệu huấn luyện của mình (x\_train và y\_train), sau đó thử nghiệm mô hình trên dữ liệu thử nghiệm của chúng em (x\_test) và cuối cùng, chúng em đánh giá mô hình hiệu suất bằng cách so sánh các dự đoán của mô hình với các giá trị thực trong y\_test. Chúng em đã sử dụng điểm chính xác để đánh giá hiệu suất của mô hình.

Sử dụng kết quả thu được ở phần trước, chúng em trình bày một bảng hiển thị Điểm chính xác cho từng mô hình khi áp dụng cho bộ kiểm tra x\_test.



Chúng em cũng trình bày một biểu đồ trực quan hóa nội dung bảng:





Nhìn vào bảng và đồ thị có thể thấy mô hình Decision Tree (DT) có độ chính xác nhỏ nhất là 92,94% và Precision 82,82%, tiếp theo là mô hình Bagging Classifier (BGC) với độ chính xác là 95,84% và Precision 86,82%. Sau đó, Gradient Boosting Classifier (GBDC) có độ chính xác 94,68% và Precision 91,91%. Mô hình Adaboost Classifier (AdaBoost) có độ chính xác là 96,03% và Precision 92,92%. Mô hình XGB Classifier (XGB) có độ chính xác là 96,71% và Precision 92,62%. Mô hình Logistic Regression (LR) có độ chính xác 95,84% và Precision 97,02%. Mô hình KNeighborsClassifier (KN) có độ chính xác 90,52% và Precision 100%. Cuối cùng là 4 mô hình có độ chính xác cao nhất đó là Mô hình Extra Trees Classifier (ETC) có độ chính xác 97,48% và Precision 97,45%. Mô hình Support Vector (SV) có độ chính xác 97,58% và Precision 97,47%. Mô hình Random Forest Classifier (RFC) có độ chính xác 97,58% và Precision 98,29%. Cuối cùng là mô hình có độ chính xác cao nhất MultinomialNB (NB) với độ chính xác là 97,09% và Precision 100%

Vì vậy, trong thử nghiệm của chúng em, mô hình tốt nhất là MultinomialNB (NB). Mô hình tồi tệ nhất là Decision Tree (DT).

#### **7.2. Hiện thực hóa lên website**

Dựa vào đánh giá độ chính xác của mỗi mô hình bọn em đã quyết định chọn Mô hình MultinomialNB (NB) để xây dựng website nhận diện tin nhắn spam thông qua tin nhắn sms

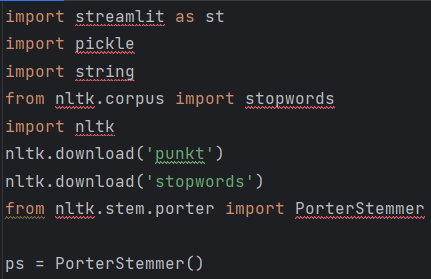
* Xuất file vectorizer và model để hỗ trợ dự đoán sms



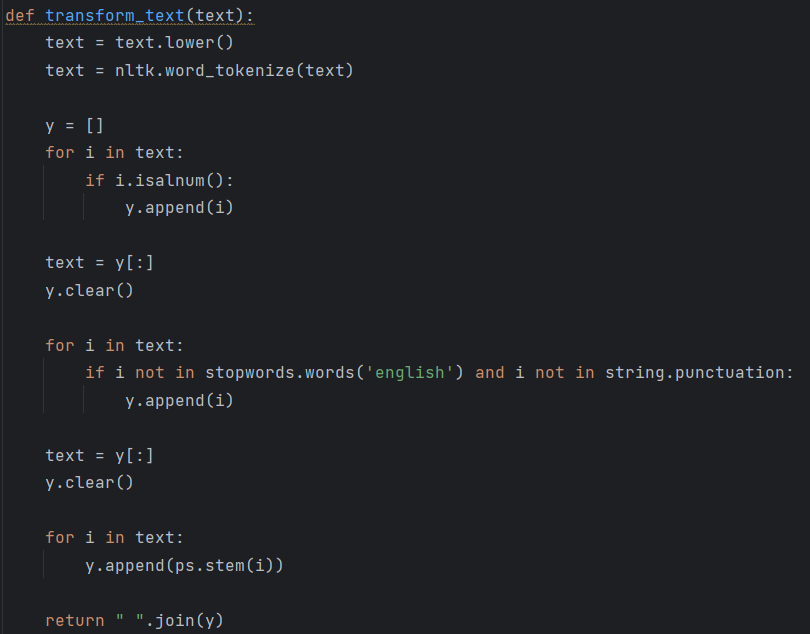
- vectorizer.pkl: File này được sử dụng để chuẩn bị hoặc biến đổi dữ liệu trước khi đưa vào mô hình máy học.

- model.pkl: File này chứa mô hình đã được huấn luyện. Dùng để dự đoán trên dữ liệu mới mà không cần phải huấn luyện lại từ đầu.

- Import các thư viện cần thiết



* Tạo hàm transfrom\_text



- Chuyển văn bản thành chữ thường (text.lower()): Đưa tất cả các ký tự trong văn bản về dạng chữ thường.

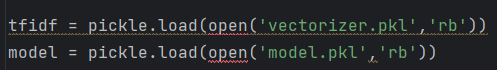
- Tokenization (nltk.word\_tokenize(text)): Phân tách văn bản thành các từ hoặc token.

- Loại bỏ các ký tự không phải chữ cái và số (if i.isalnum()): Loại bỏ các ký tự đặc biệt, chỉ giữ lại các từ chứa chữ cái hoặc số.

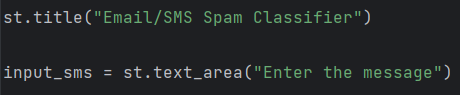
- Loại bỏ stop words và dấu câu (if i not in stopwords.words('english') and i not in string.punctuation): Stop words là các từ phổ biến trong ngôn ngữ như "the", "is",... không mang nhiều ý nghĩa trong việc phân loại hoặc xử lý văn bản nên được loại bỏ. Dấu câu cũng thường không cần thiết trong quá trình xử lý.

- Stemming (ps.stem(i)): Áp dụng việc rút gọn từ về dạng gốc của chúng (stemming) để đưa các từ về dạng cơ bản nhất.

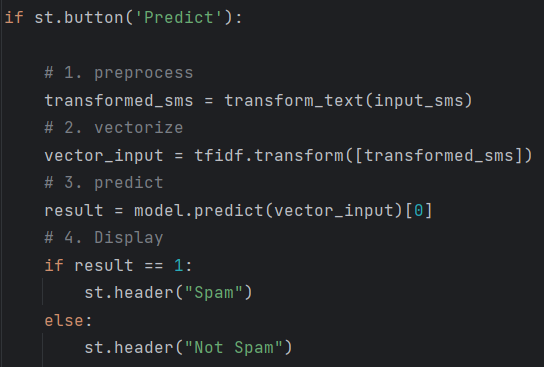
* Định nghĩa 2 file vectorizer và model



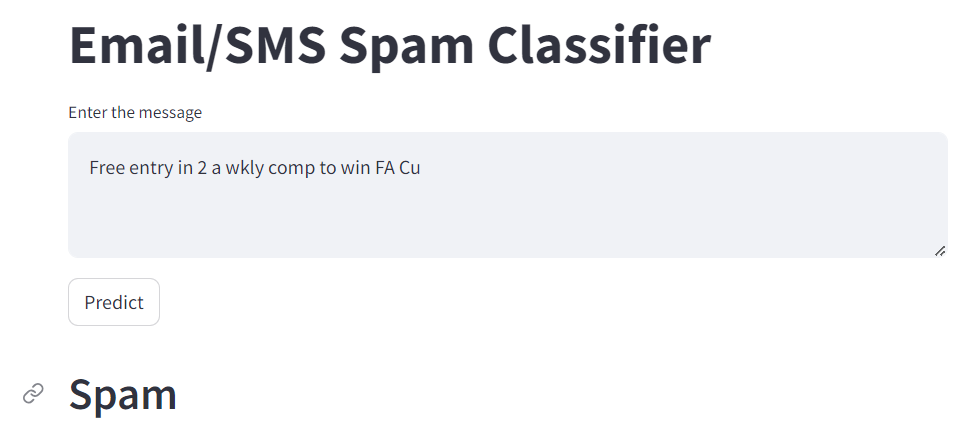
* Tạo tiêu đề trang web và input sms



* Bắt sự kiện cho nút button predict



* Cho dữ liệu input vào hàm transform\_text để xử lý dữ liệu
* Biến đổi dữ liệu thông qua file vectorize
* Cuối cùng là dự đoán dữ liệu thông qua model MultinomialNB (NB)
* Nếu kết quả trả về 1 thì xuất ra ‘Spam’ ngược lại thì ‘Not Spam’
* Giao diện trang web



### **8. Bảng phân công**

|  | Dương Bảo Tâm | Huỳnh Đăng Nghĩa | Lê Quang Nhật | Nguyễn Ngọc Hải Sơn |
| --- | --- | --- | --- | --- |
| Slide thuyết trình | x |  | x | x |
| Word báo cáo | x | x | x |  |
| Video demo |  | x |  | x |
| Mức độ hoàn thành | 100% | 100% | 100% | 100% |

### **9. Tài liệu tham khảo.**

1. [Understanding logistic regression analysis - Biochemia Medica (biochemia-medica.com)](https://www.biochemia-medica.com/en/journal/24/1/10.11613/BM.2014.003/fullArticle)

2. [(PDF) A COMPARISON OF LOGISTIC REGRESSION AND GEOGRAPHICALLY WEIGHTED LOGISTIC REGRESSION (GWLR) ON COVID-19 DATA IN WEST SUMATRA (researchgate.net)](https://www.researchgate.net/publication/374343025_A_COMPARISON_OF_LOGISTIC_REGRESSION_AND_GEOGRAPHICALLY_WEIGHTED_LOGISTIC_REGRESSION_GWLR_ON_COVID-19_DATA_IN_WEST_SUMATRA)

3. [Support Vector Machines in R | Journal of Statistical Software (jstatsoft.org)](https://www.jstatsoft.org/article/view/v015i09)

4. [# Phân lớp bằng Random Forests trong Python (viblo.asia)](https://viblo.asia/p/phan-lop-bang-random-forests-trong-python-djeZ1D2QKWz)

5. [Python Decision Tree Classification Tutorial: Scikit-Learn DecisionTreeClassifier | DataCamp](https://www.datacamp.com/tutorial/decision-tree-classification-python)