VIETNAM NATIONAL UNIVERSITY HCMC
UNIVERSITY OF INFORMATION TECHNOLOGY

BÁO CÁO CUỐI KỲ

CS116.M12.KHCL

SUPPORT VECTOR MACHINE

Ngày 24 tháng 12 năm 2021

Nhóm thực hiện:
Lương Phạm Bảo 19521242
Nguyễn Gia Thống 19520993
Phạm Ngọc Dương 19521412

MỤC LỤC

1 Giới thiệu sơ lược về SVM

1.1 Lịch sử hình thành

SVM được phát triển vào năm 1963 bởi Vladimir Vapnik và Alexey Chervonenkis, Vapnik đã tiếp tục cải tiến phương pháp phân loại này vào những năm 1990. Tại thời điểm này (90s) cho đến đầu những năm 2000, khi AI lần thứ hai rơi vào thời kì băng giá, neural networks với những hạn chế về activation function của nó dần được thay thế bởi SVM và nhiều nhà khoa học về machine learning đã chuyển sang nghiên cứu SVM trong thời gian đó. Các kỹ thuật về kernel cũng phát triển giúp SVMs giải quyết được cả các vấn đề về việc dữ liệu không phân biệt tuyến tính.

1.2 Ý tưởng

Support Vector Machine (SVM) là một thuật toán Machine Learning có giám sát được sử dụng để phân loại dữ liệu thành các nhóm riêng biệt. SVM xây dựng một siêu phẳng hoặc một tập hợp siêu phẳng trong không gian đa chiều hoặc vô hạn chiều, có thể được sử dụng để phân loại, hồi quy hoặc các tác vụ khác. Theo trực quan, để phân loại tốt nhất, siêu phẳng được đặt càng xa điểm dữ liệu của tất cả các lớp (được gọi là margin function), vì nói chung, lề càng lớn thì phương sai càng nhỏ (sự khác biệt giữa training sets và testing sets). Trong nhiều trường hợp, ta không thể phân tách tuyến tính các lớp dữ liệu trong không gian ban đầu, do đó, đôi khi cần phải biến đổi các điểm dữ liệu trong không gian ban đầu thành một không gian mới nhiều chiều hơn, để dễ dàng tách chúng ra trong không gian mới.

Cho đến nay, SVM đã được chứng minh đạt được kết quả tốt trong nhiều lĩnh vực như nhận diện khuôn mặt, phân loại văn bản và siêu văn bản, nhận dạng chữ viết tay, sinh hoá,... Do đó hoàn toàn có cơ sở để tin rằng SVM sẽ đóng một vai trò quan trọng trong các bài toán phân loại trong tương lai.

2 Thuật toán SVM

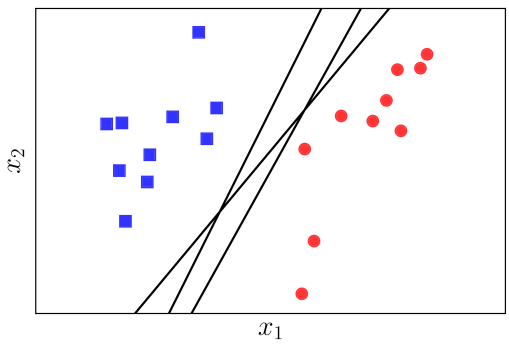
2.1 Vấn đề của các thuật toán máy học phân loại trước khi SVM ra đời

Mạng nơ-ron nhân tạo (ANN) là một trong những mô hình máy học đầu tiên được phát minh vào những năm 1940 với kiểu cấu trúc mô phỏng hệ thống nơ-ron sinh học của não người. Nó được sử dụng lần đầu tiên vào những năm 1980 và kể từ đó đã được sử dụng cho nhiều ứng dụng liên quan đến kỹ thuật, phần lớn là do khả năng của nó trong việc trích xuất các mối tương quan phức tạp và phi tuyến tính giữa các đặc trưng của dữ liệu. Tuy nhiên, những ANN giai đoạn này trên thực tế chỉ cho kết quả tốt khi có một số lượng lớn dữ liệu huấn luyện có sẵn. Những mô hình mạng nơ-ron này thường tổng quát hóa kém trên các bộ dữ liệu và đưa ra những kết quả cục bộ hay vì kết quả tối ưu toàn cục.

Nguyên nhân các mô hình mạng nơ-ron này tổng quát hóa kém với các bộ dữ liệu chính là do overfitting. Overfitting là khi các thuật toán máy học được huấn luyện trên một bộ dữ liệu có kích thước lớn và xây dựng một mô hình hàm với độ phức tạp cao, để giảm tối đa sai số, mất đi tính tổng quát của các kết quả dự đoán từ dữ liệu. Chính vì thế, độ chính xác của mô hình trên bộ dữ liệu huấn luyện của một mô hình overfit sẽ rất cao, nhưng sẽ thấp với những bộ dữ liệu mới mà mô hình không dùng để huấn luyện.

Một hướng tiếp cận đơn giản để giải quyết vấn đề mô hình bị overfit chính là làm giảm độ phức tạp của mô hình. Tuy việc này sẽ làm giảm kết quả của mô hình ở giai đoạn huấn luyện, nó sẽ giúp tổng quát hóa mô hình và tăng khả năng dự đoán trên các bộ dữ liệu mới.Và khi gặp các bài toán mà chúng ta cần phải làm giảm độ phức tạp của mô hình, thì việc xây dựng và sử dụng hàm chính quy (regularization function) để giám sát và loại bỏ bớt những thuộc tính không cần thiết là một kỹ thuật hiệu quả để giải quyết vấn đề này. Đây chính là thuyết Vapnik-Chervonenkis và là ý tưởng cơ bản để xây dựng nên các thuật toán SVM.

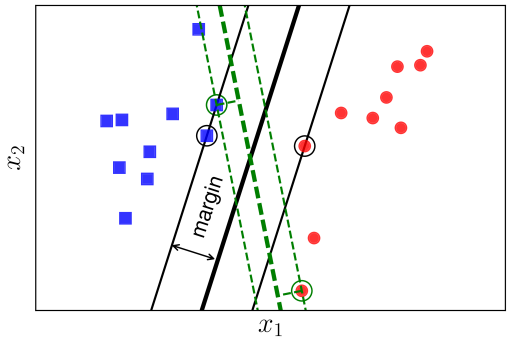
2.2 Định nghĩa siêu phẳng



Hình 1: Vô số các mặt phẳng phân chia thoả mãn

SVM sẽ tìm một đường thẳng (hoặc một siêu phẳng) lý tưởng phân cách hai lớp dữ liệu. Các vectors tạo nên siêu phẳng còn được gọi là support vectors. Nói theo cách khác, siêu phẳng là một tập hợp các điểm thoả mãn $wx+b=0$.

Nhằm phân chia hai lớp một cách rõ ràng và giúp các dự đoán chính xác hơn cho các điểm dữ liệu mới, ta cần tìm ra mặt phẳng phân chia tối ưu nhất sao cho khoảng cách từ điểm gần nhất của mỗi lớp tới đường phân chia là như nhau và tối đa hoá khoảng cách lề giữa hai lớp.



Hình 2: Lề của hai lớp là bằng nhau và lớn nhất có thể

2.3 Khoảng cách từ một điểm đến siêu phẳng

Trong không gian hai chiều, khoảng cách từ một điểm dữ liệu (x_0, y_0) tới đường thẳng $w_1x+w_2y+b=0$ được xác định như sau:

$$\frac{|w_1x_0+w_2y_0+b|}{\sqrt{w_1^2+w_2^2}}$$

Tương tự với không gian ba chiều, khoảng cách từ một điểm dữ liệu (x_0, y_0, z_0) đến đường thẳng $w_1x+w_2y+w_3z+b=0$ được xác định như sau:

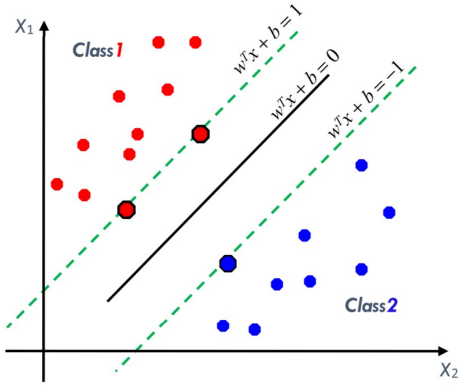
$$\frac{|w_1x_0+w_2y_0+w_3z_0+b|}{\sqrt{w_1^2+w_2^2+w_3^2}}$$

Tổng quát lại, khoảng cách từ một vector có toạ độ $(x_1, x_2, \ldots, x_d)$ tới siêu phẳng $w_1x_1+w_2x_2+\cdots+w_dx_d+b=0$ được xác định bởi:

$$\frac{|w_1x_1+w_2x_2+\cdots+w_dx_d+b|}{\sqrt{w_1^2+w_2^2+\cdots+w_d^2}} = \frac{|\mathbf{w}^T\mathbf{x}_0+b|}{\|\mathbf{w}\|_2}$$

với $\mathbf{x}_0 = [x_1, x_2, \ldots, x_d]^T, \mathbf{w} = [w_1, w_2, \ldots, w_d]^T$.

2.4 SVM Hard Margin - Dữ liệu phân biệt tuyến tính



Hình 3: Siêu phẳng dùng để phân loại dữ liệu của hai phân lớp và margin của các lớp này. Support Vectors (là các điểm dữ liệu được in đậm) của mỗi lớp được dùng để tối đa hóa margin của các lớp dữ liệu với siêu phẳng.

Mục đích của việc sử dụng Support Vector Machine hay SVM, là nhằm tìm được bộ phân lớp hay là hàm quyết định, có khả năng phân tách những dữ liệu mới mà mô hình chưa từng gặp một cách chính xác với độ tổng quát cao ở giai đoạn testing. Với dữ liệu hai lớp (nhị phân), bộ phân lớp có thể là một đường thẳng tuyến tính với margin (khoảng cách lề) tối đa với các điểm dữ liệu thuộc cả hai lớp. Bộ phân loại tuyến tính này thường được gọi là siêu phẳng tối ưu (optimal hyperplane). Đường thẳng tuyến tính phân lớp cho bộ dữ liệu huấn luyện, x_i ($i = 1, 2, 3, ..., n$) được định nghĩa như sau:

$$\omega^T x + b = 0, \tag{1}$$

trong đó, ω là một vector với n-chiều và b là tham số bias. Siêu phẳng này phải chứa hai đặc tính cụ thể: (1) là nó phải phân chia bộ dữ liệu sao cho có ít các điểm dữ liệu bị phân lớp sai nhất có thể. (2) Khoảng cách từ siêu phẳng đến điểm dữ liệu của mỗi phân lớp gần nó nhất phải là tối đa. Với hai điều kiện này, các điểm dữ liệu thuộc mỗi lớp chỉ có thể nằm bên trái (y = 1) hay bên phải (y =-1) của siêu phẳng. Do vậy, hai giá trị margin, hay có thể xem như giá trị biển diễn sự phân chia giữa các lớp trong bộ dữ liệu (Hình 3), được định nghĩa như sau:

$$\omega^T x + b \begin{cases} \geq 1 \text{ for } y_i = 1 \\ \leq -1 \text{ for } y_i = -1 \end{cases} \tag{2}$$

Tuy vậy, vùng tổng quát của siêu phẳng có thể nằm ở vị trí bất kì trong khoảng từ -1 đến 1 và có rất nhiều margin có thể coi là margin của mỗi phân lớp. Để có thể tìm được siêu phẳng tốt nhất, chúng ta phải tính và tìm giá trị cực đại của khoảng cách (d) giữa các margin:

$$d(\omega, b; x) = \frac{\left|(\omega^T x + b - 1) - (\omega^T x + b + 1)\right|}{\|\omega\|} = \frac{2}{\|\omega\|} \tag{3}$$

Tìm giá trị cực đại của margin cũng sẽ tương đương với việc tìm cực tiểu của vector chiều ω hay tìm cực tiểu của $\frac{1}{2}\omega^T\omega$[19]. Bài toán lồi tổng quát để xác định siêu phẳng tối ưu sẽ được biểu diễn như sau:

$$\begin{aligned} \text{Min}_{\omega,b} &= \frac{1}{2}\omega^T\omega \\ \text{s.t.} \quad & y_i(\omega^T x + b) \geq 1 \end{aligned} \tag{4}$$

Và sau khi áp điều kiện rằng buộc cho margin của hai phân lớp với hệ số Lagrange (α), ta được:

$$L_p(\omega, b, \alpha) = \frac{1}{2}\omega^T\omega - \sum_{i=1}^{N} \alpha_i \left\{ y_i \left[\omega^T x_i + b \right] - 1 \right\} \tag{5}$$

Điểm yên ngựa (saddle point) hay điểm minimax, là điểm trên bề mặt đồ thị của hàm trong đó các sườn theo hướng trực giao đều bằng 0, nhưng không phải là điểm cực trị cục bộ của hàm. Để tìm điểm yên ngựa cho phương trình ở trên, ta cần phải thỏa các điều kiện $Karush - Kuhn - Tucker$ (KKT) sau:

$$\frac{\partial L}{\partial \omega} = 0 \;\Rightarrow\; \omega_0 = \sum_{i=1}^{N} \alpha_i x_i y_i \tag{6}$$

$$\frac{\partial L}{\partial b} = 0 \;\Rightarrow\; \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{7}$$

với α_i có giá trị khác không khi và chỉ khi giá trị đầu vào x_i tương ứng là một support vector[2]. Support vectors là những điểm dữ liệu được chọn làm ranh giới của mỗi phân lớp dữ liệu để có thể tính giá trị margin với siêu phẳng (điểm dữ liệu in đậm Hình 3).

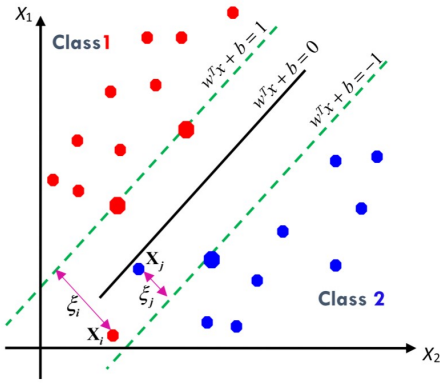
Thế phương trình (6) và (7) vào phương trình (5), ta sẽ có được phương trình tổng quát của SVM với trường hợp dữ liệu phân biệt tuyến tính, với hai điều kiện rằng buộc:

$$\begin{aligned} \text{Max } L_d(\alpha) &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ &\text{s.t} \begin{cases} \alpha_i \geq 0 \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \end{aligned} \tag{8}$$

Phương trình và những ràng buộc này được dùng để tìm các support vectors và dữ liệu đầu vào tương ứng. Ta tính được giá trị ω của siêu phẳng từ phương trình (6) và bias b từ phương trình (9), viết dưới dạng trung bình như sau:

$$b_0 = \frac{1}{N} \sum_{S=1}^{N} (y_S - \omega^T x_S) \tag{9}$$

2.5 SVM Soft Margin - Dữ liệu không phân biệt tuyến tính



Hình 4: Dữ liệu không phân biệt tuyến tính và các biến lệch (slack variable) được định nghĩa để hạn chế sai số phân lớp của siêu phẳng.

Luôn có những trường hợp dữ liệu không thể phân tách tuyến tính do các thuộc tính của bộ dữ liệu thiếu tương đồng. Tuy vậy, SVM tuyến tính vẫn có thể cho ra được lời giải tốt nếu chúng ta có thể định nghĩa được một hàm penalty,nhằm tính được khoảng cách ξ_i giữa các điểm dữ liệu phân loại sai của mỗi phân lớp tính từ margin của phân lớp đó và đi tìm giá trị cực tiểu của nó (Hình 4). Hàm penalty lúc này được định nghĩa như sau:

$$F(\xi) = \sum_{i=1}^{N} \xi_i \tag{10}$$

Khi này, hàm tối ưu hóa lồi đã đề cập trước đó (4) cho trường hợp dữ liệu không phân biệt tuyến tính, với hàm penalty vừa định nghĩa sẽ trở thành:

$$\begin{aligned} \text{Min}_{\omega,b} &= \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{N}\xi_i \\ \text{s.t. } & y_i(\omega^T x + b) \geq 1 - \xi_i \end{aligned} \tag{11}$$

với C là tham số "đánh đổi" được thêm vào để cực đại hóa margin và cực tiểu hóa sai số phân lớp.

Để giải quyết bài toán tối ưu với điều kiện ràng buộc của margin này, ta có thể dùng hệ số Lagrange $(\alpha(\alpha_i \geq 0), \beta(\beta_i \geq 0))$. Ta có phép toán không ràng buộc từ phương trình (11) như sau:

$$L_p(\omega, b, \xi, \alpha, \beta) = \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left\{y_i\left[\omega^T x_i + b\right] - 1 + \xi_i\right\} - \sum_{i=1}^{N}\beta_i\xi_i \tag{12}$$

Nghiệm của phương trình trên là tối ưu khi thỏa các điều kiện KKT sau:

$$\frac{\partial L}{\partial \omega} = 0 \;\Rightarrow\; \omega = \sum_{i=0}^{N}\alpha_i y_i x_i \tag{13}$$

$$\frac{\partial L}{\partial b} = 0 \;\Rightarrow\; \sum_{i=1}^{N}\alpha_i y_i = 0 \tag{14}$$

$$\frac{\partial L}{\partial \xi} = 0 \;\Rightarrow\; \alpha_i + \beta_i = C \tag{15}$$

Thế các phương trình từ (13) đến (15) vào phương trình (12), bài toán kép với Soft Margin SVM có dạng:

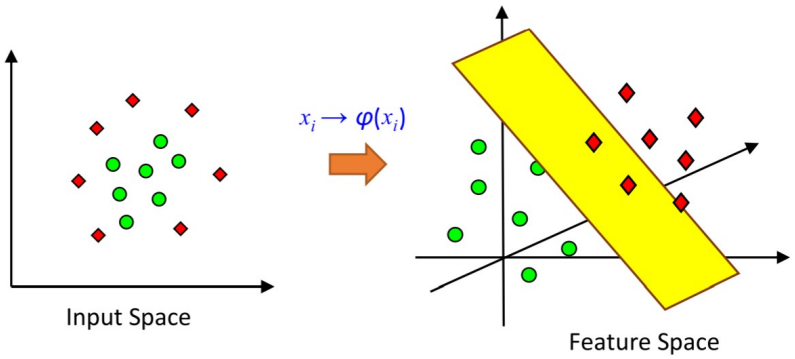
$$\begin{aligned} \text{Max } L_d(\alpha) &= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ \text{s.t} &\begin{cases} 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{N}\alpha_i y_i = 0 \end{cases} \end{aligned} \tag{16}$$

Điểm khác biệt giữa phương trình này với phương trình (8) tương tự chính là điều kiện ràng buộc lên hệ số Lagrange α luôn phải nhỏ hơn hoặc bằng tham số penalty C

2.6 Kernel trong SVM - Dữ liệu không tuyến tính

2.6.1 Không gian thuộc tính

Khi đi tìm siêu phẳng tối ưu cho dữ liệu đầu vào, chúng ta đang hướng đến việc tăng tính tổng quát của mô hình. Tuy nhiên, nếu chưa xác định được siêu phẳng tối ưu, thì ngay cả với những dữ liệu phân biệt tuyến tính, mô hình sẽ không đạt được tính tổng quát với dữ liệu đã cho. Lúc này, dữ liệu đầu vào sẽ được ánh xạ vào không gian tích vô hướng có chiều lớn hơn, còn được gọi là không gian thuộc tính (feature space) hay không gian Hilbert, để giảm thiểu vấn đề này. Lúc này, dữ liệu vẫn là phi tuyến tính trong không gian đầu vào, nhưng lúc này mô hình SVM có thể được dùng để phân tách chúng trong không gian thuộc tính. Không gian thuộc tính được sử dụng để phân tách dữ liệu trên một không gian lớn chiều hơn hơn, như ở Hình 5.



Hình 5: Ánh xạ dữ liệu đầu vào lên không gian thuộc tính có chiều cao hơn nhằm tăng khả năng tổng quát hóa dữ liệu của mô hình.

Ở đây, dữ liệu đầu vào x sẽ được biểu diễn dưới dạng $\varphi(x)$ trong không gian thuộc tính theo điều kiện Mercer. Tuy nhiên, chúng ta không cần diễn giải rõ biểu diễn của dữ liệu đầu vào trong không gian thuộc tính, mà chỉ cần tính tích vô hướng của chúng. Ta có thể dùng hàm Kernel để tìm ra kết quả này:

$$\varphi(x_i, x_j) = K(x_i, x_j) \tag{17}$$

Nhờ đó ta có thể áp dụng SVM để giải quyết các vấn đề phi tuyến. Tuy nhiên, phương trình trên chỉ đúng khi và chỉ khi với một hàm vector phi tuyến $g(x)$, điều kiện sau được thỏa mãn:

$$\int K(x_i, x_j)\, g(x_i) g(x_j) dx_i dx_j \geq 0 \tag{18}$$

2.6.2 Kernel Trick

Từ đó, ta có thể xét việc dùng hàm kernel để tính được tích vô hướng của dữ liệu đầu vào trong không gian thuộc tính cho công thức đối ngẫu tổng quát (19) đã đề cập để giải quyết bài toán phân loại dữ liệu không phân tách tuyến tính:

$$\begin{aligned} \text{Max } L_d(\alpha) &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\ &\text{s.t} \begin{cases} 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \end{aligned} \tag{19}$$

Tuy nhiên, việc đi tìm siêu phẳng tối ưu lúc này sẽ khá phức tạp khi mà vector trọng số ω là một ẩn số, thể hiện ở phương trình (20) trong không gian thuộc tính:

$$\omega = \sum_{i=1}^{N} y_i \alpha_i \varphi(x_i) \tag{20}$$

Lúc này, ta có thể dùng kernel trick. Cách tiếp cận này cho phép ta giải quyết vài toàn mà không cần biết giá trị cụ thể của tham số ω. Bằng cách thế phương trình giá trị ω (20) vào phương trình (9), ta sẽ tính được giá trị của tham số bias b bằng hàm kernel:

$$b = y_i - \sum_{i,j=1}^{N_{SV}} \alpha_i y_i K(x_i, x_j) \tag{21}$$

Và siêu phẳng sẽ được định nghĩa như sau:

$$d(x) = \omega^T \varphi(x) + b \tag{22}$$

Do hàm quyết định tối ưu đã có thể được tính bằng việc thế phương trình (20) vào phương trình (22) với một hàm kernel phù hợp [30], nên phương trình tổng quát của siêu phẳng được phát biểu như sau:

$$d(x) = \sum_{i=1}^{N} y_i \alpha_i K(x_i, x_j) + b \tag{23}$$

Giá trị cụ thể của trọng số ω lúc này đã được thay thế, và các mô hình thuật toán SVM có thể được dùng để giải quyết các vấn đề phi tuyến bằng việc chọn một hàm kernel phù hợp với dữ liệu bài toán.

2.7 Các bài toán khi nào nên sử dụng SVM (NLP, CV, ML)

Thông qua thực nghiệm và tham khảo các tài liệu thì ta thấy rằng SVM sẽ hoạt động rất hiệu quả với các dữ liệu có ít số chiều. Ngoài ra, SVM còn rất hiệu quả trong các bài toán image classification (trong cuộc thi VN Face Recognition, mô hình tốt nhất đã sử dụng mô hình phân loại SVM multiclass sau khi tuning các parameters và chỉ sử dụng một mạng rút trích đặc trưng bình thường là VGG16) hoặc với bài toán Text Classification, sentiment analysic

2.8 Ứng dụng thực tế

Với những ưu điểm trong việc phân loại và tìm ra các quan hệ trong những bộ dữ liệu phức tạp, nhiều thuộc tính với độ nhiễu cao, thuật toán SVM đã được sử dụng trong khá nhiều vĩnh vực nghiên cứu và ứng dụng thực tiễn trong sản suất kinh doanh. Một số những ứng dụng phổ biến và thành công với thuật toán SVM có thể kể đến như: các ứng dụng phân loại như nhận diện chữ viết tay, hypertext, nhận diện khuôn mặt.

Được ứng dụng trong ngành nghiên cứu sinh học y khoa, đặc biệt là lĩnh vực nghiên cứu ung thư: Ung thư là một loại bênh di truyền chúng ta có thể nghiên cứu và đánh giá dựa trên các đặc tính và cấu trúc gen của chúng. Chính vì vậy SVM đã được dùng trong việc phân loại, chẩn đoán các loại bênh ung thư,và các biến chủng, tình trạng ung thư khác nhau trên nhiều bộ dữ liệu từ những năm 2000. Không những vậy, một ứng dụng khác sử dụng SVM chính là quá trình thử nghiệm và khai phá thuốc điều trị ung thư khi mà nó được dùng để phân loại và phát hiện các thành phần thuốc chủ động và thụ động trong tương tác so với nhãn thực của chúng.

Các ngành dầu khí, khai khoán, địa chất, đào mỏ và xây dựng: Ứng dụng SVM trong xây dựng các mô hình dự đoán độ thẩm thấu của tầng đá trong khai thác tham dò dầu khí, ước lượng vận tốc sóng âm qua các tầng đá để lựa chọn vị trí khoan dò, hay ước lượng sức bền vật liệu.

3 Ưu nhược điểm của SVM

3.1 Ưu điểm

- SVM có các tham số được chính quy hóa nên sẽ giảm thiệu việc bị overfitting.
- SVM hoạt động trên dữ liệu không có cấu trúc hoặc bán cấu trúc như văn bản, hình ảnh và cây.
- SVM có thể xử lý được dữ liệu phi tuyến tính bằng cách dùng kernel trick của nó để đưa dữ liệu vào không gian nhiều chiều hơn để phân chia tuyến tính.
- Mô hình SVM không có quá nhiều siêu tham số.
- Bởi vì mô hình SVM chỉ sử dụng một tập con của các điểm dữ liệu (các support vector) để tìm được biên quyết định nên sẽ tiết kiệm được bộ nhớ rất nhiều.

3.2 Nhược điểm

- SVM không phù hợp để giải quyết các bài toán có số điểm dữ liệu quá lớn vì thời gian huấn luyện sẽ lâu.
- Việc chọn được kernel phù hợp để giải một bài toán là chuyện không dễ dàng.
- Để tinh chỉnh tham số cho mô hình SVM là một công việc khó cho dù nó chỉ có hai siêu tham số: C, gamma và kernel.
- SVM có thể phân lớp ngay cả khi số chiều của dữ liệu lớn hơn số điểm dữ liệu mà ta có tuy nhiên kết quả sẽ không tốt cho lắm.
- Nếu như có các điểm dữ liệu nhiễu hoặc các lớp bị chồng lên nhau thì SVM sẽ không thể phân lớp tốt.
- SVM không phải là mô hình phân lớp dựa trên xác suất nên không đưa ra kết quả theo xác suất.

3.3 So sánh với mô hình Deep Learning

Ta sẽ đề cập về cách mà SVM và Neural Networks xử lý cùng một tập dữ liệu về độ chính xác và lượng thời gian cần thiết để train. Cả SVM và NNs đều có thể giải quyết cùng một vấn đề phân loại dựa trên cùng một tập dữ liệu. Điều này có nghĩa là nhìn một cách khách quan từ nhiều góc độ và yếu tố khác nhau thì chưa thể kết luận mô hình nào hoạt động tốt hơn.

Tuy nhiên, điều quan trọng hơn là cả hai đều hoạt động với độ chính xác có thể so sánh được với cùng một tập dữ liệu trong trường môi trường và tài nguyên train như nhau. Nhưng NNs có xu hướng hoạt động tốt hơn SVM nếu được cung cấp càng nhiều tài nguyên để train. Mặc dù vậy, thời gian cần thiết để train và tiếp cận hai mô hình là rất khác nhau đối với cùng một tập dữ liệu. Dưới đây là những điểm khác biệt:

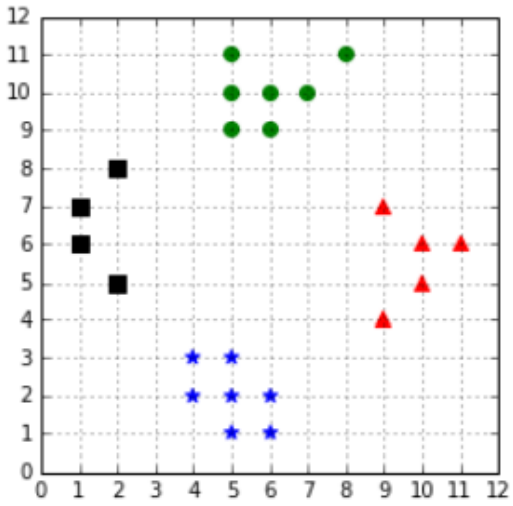
1. Kết cấu mô hình.
2. Lượng data cần cho quá trình train.
3. Quá trình train mô hình.
4. Tối ưu các tham số.
5. Ảnh hưởng của việc khởi tạo ngẫu nhiên các trọng số lên mô hình.

4 Sử dụng SVM cho phân loại nhiều lớp

4.1 Giới thiệu

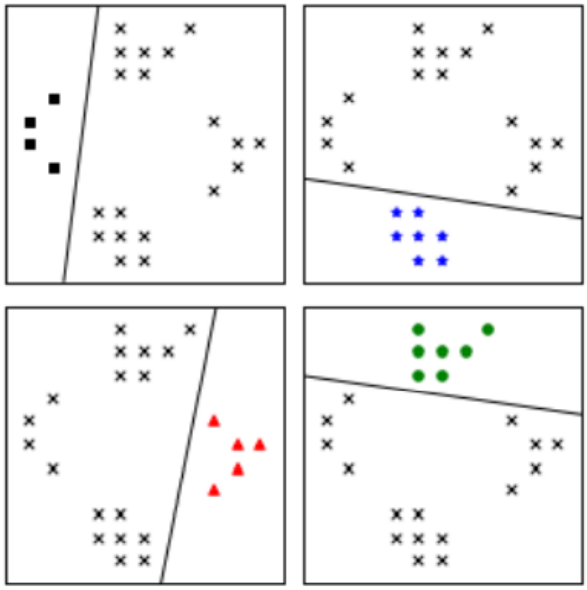
Liệu có khả thi khi ta áp dụng SVM vào những bài toán có nhiều hơn hai lớp cần được phân loại? Như ta đã biết, SVM là thuật toán dùng để phân loại hai lớp nhị phân, thế nhưng ta vẫn có khá nhiều cách tiếp cận dựa trên ý tưởng này để giải quyết những trường hợp có nhiều hơn hai lớp.

4.2 Hướng tiếp cận

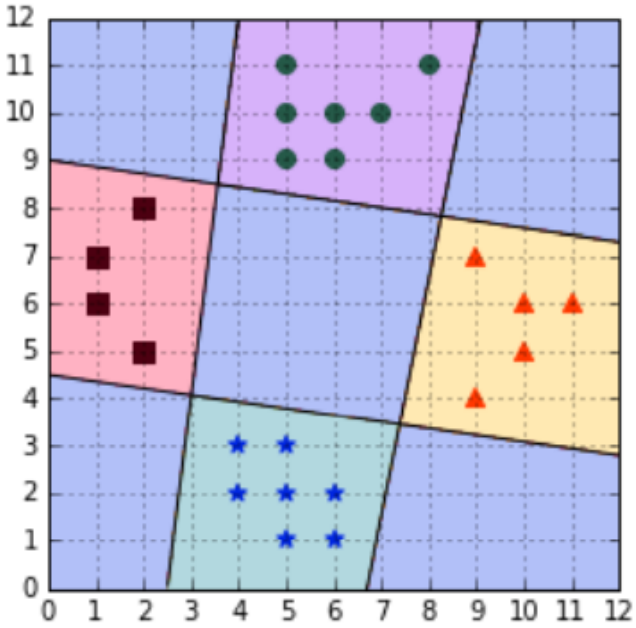


Hình 6: Bài toán phân loại 4 lớp

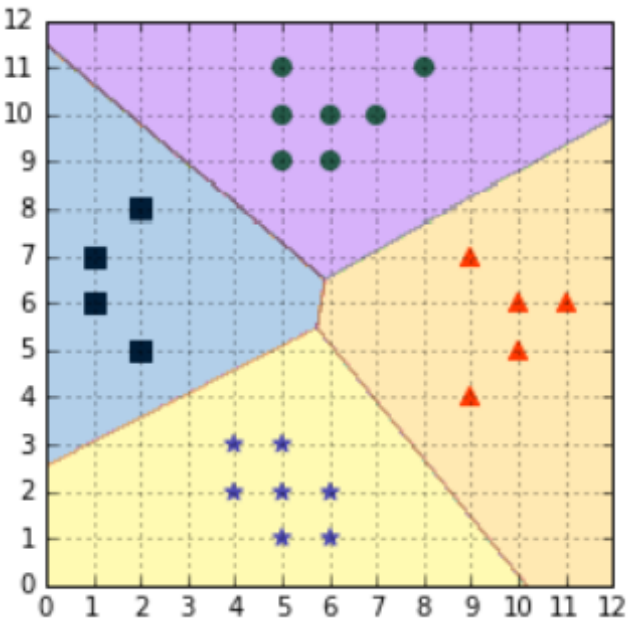
Đầu tiên, phương pháp **One-against-all** sẽ tách biệt từng lớp với những lớp còn lại và áp dụng heuristic để tránh những dự đoán không chính xác đối với những điểm dữ liệu mới.



Hình 7: Hướng tiếp cận One-against-all

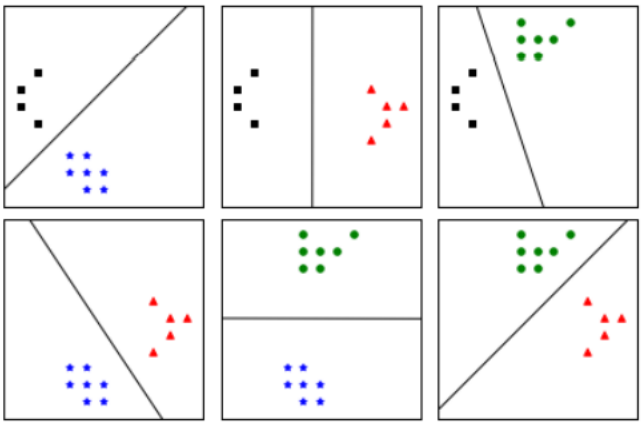


Hình 8: One-against-all trước khi áp dụng heuristic

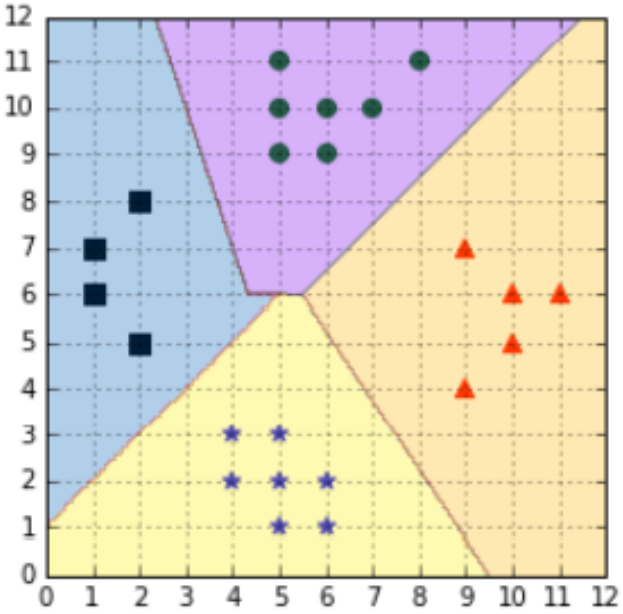


Hình 9: One-against-all sau khi áp dụng heuristic

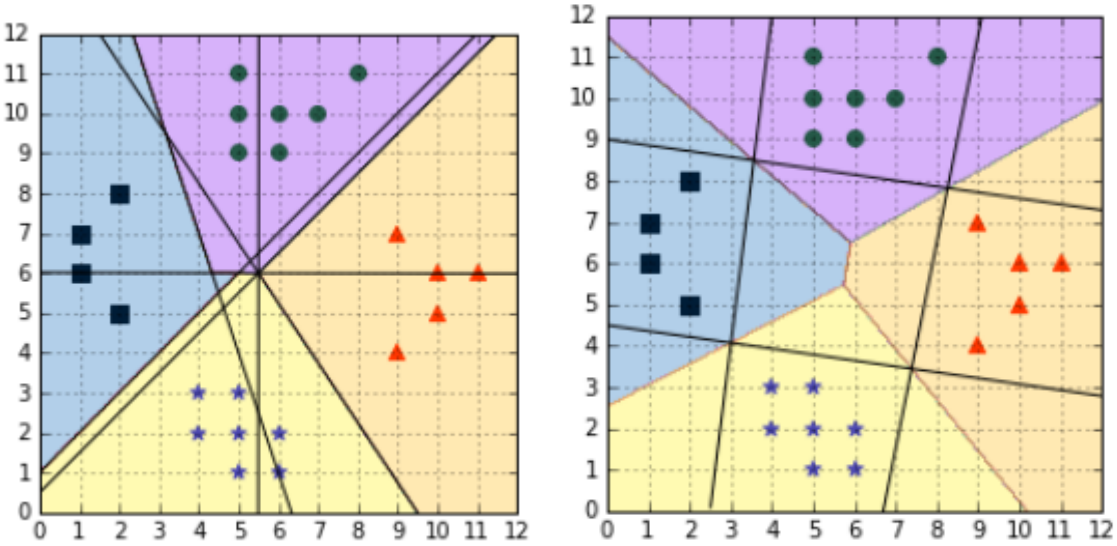
Tiếp đến, phương pháp **One-against-one** sẽ lấy các lớp theo từng cặp và phân loại chúng. Áp dụng mô hình voting scheme để dự đoán những điểm dữ liệu mới.



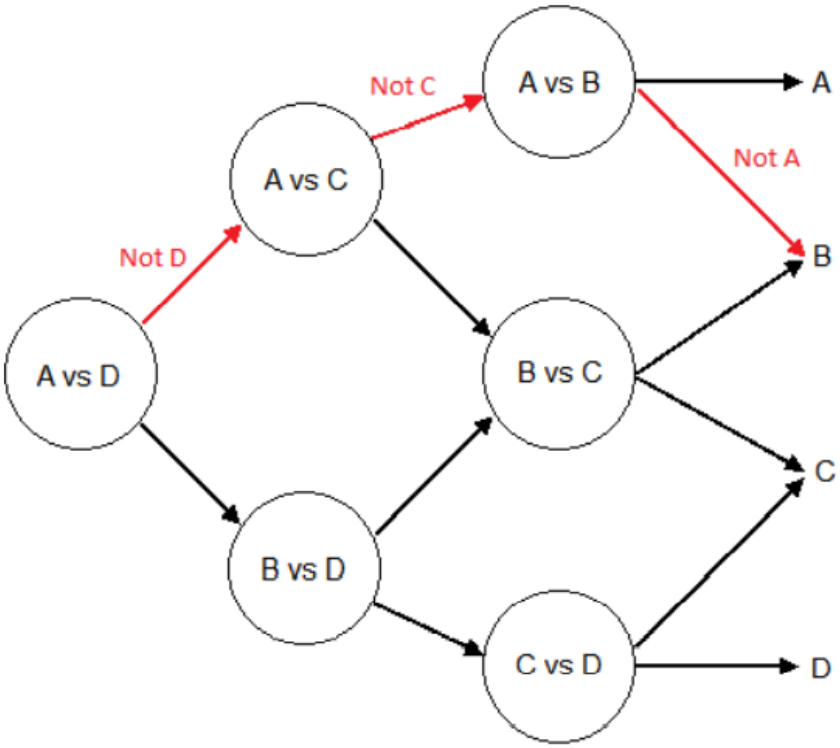
Hình 10: One-against-one



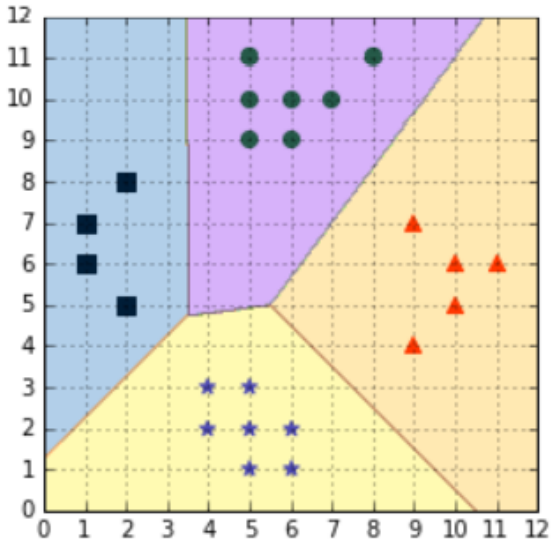
Hình 11: Dự đoán dựa trên mô hình voting scheme



Hình 12: So sánh One-against-one (trái) với One-against-all (phải)



Hình 13: Áp dụng đồ thị có hướng Acyclic để đưa ra dự đoán



Hình 14: Thuật toán Crammer & Singer

4.3 Tổng quan

Bảng 1: Các hướng tiếp cận phân loại đa lớp trong SVM

Method name	One-against-all	One-against-one	Weston and Watkins	DAGSVM	Crammer and Singer
First SVMs usage	1995	1996	1999	2000	2001
Approach	Use several binary classifiers	Use several binary classifiers	Solve a single optimization problem	Use several binary classifiers	Solve a single optimization problem
Training approach	Train a single classifier for each class	Train a classifier for each pair of classes	Decomposition method	Same as one-against-one	Decomposition method
Number of trained classifiers (K is the number of classes)	K	$\frac{K(K-1)}{2}$	1	$\frac{K(K-1)}{2}$	1
Testing approach	Select the class with the biggest decision function value	"Max-Wins" voting strategy	Use the classifier	Use a DAG to make predictions on K-1 classifiers	Use the classifier
scikit-learn class	LinearSVC	SVC	Not available	Not available	LinearSVC
Drawbacks	Class imbalance	Long training time for large K	Long training time	Not available in popular libraries	Long training time

5 Các siêu tham số của mô hình SVM

5.1 Kernel

5.1.1 Tính chất hàm kernel

Các hàm kernel cần có các tính chất như sau:

- Đối xứng: $k(x,z) = k(z,x)$. Điều này luôn được thỏa mãn vì tích vô hướng của hai vector có tính chất đối xứng.
- Về mặt lý thuyết các hàm kernel phải thỏa mãn điều kiện Mercer:

$$\sum_{n=1}^{N}\sum_{m=1}^{N} k(x_m,x_n)c_n c_m \geq 0, \quad \forall c_i \in R, i = 1,2,...,N \tag{24}$$

  Tính chất này đảm bảo cho hàm mục tiêu của bài toán đối ngẫu (19) là hàm lồi.

- Trong thực tế vẫn có một số hàm kernel không thỏa mãn điều kiện Mercer nhưng vẫn cho kết quả chấp nhận được. Những hàm số này vẫn được gọi là kernel.

5.1.2 Một số hàm kernel thông dụng

Một số hàm kernel thông dụng:

1. Linear: Ở trường hợp này hàm kernel chính là tích vô hướng của hai vector.

$$k(x,z) = x^T z$$

2. Polynomial (Đa thức):

$$k(x,z) = (r + \gamma x^T z)^d$$

Với d là một số dương để chỉ bậc của đa thức.

3. Radial Basic Function: viết tắt là kernel *'rbf'* hay còn gọi là Gaussian kernel được sử dụng nhiều nhất trong thực tế và là lựa chọn mặc định trong sklearn.

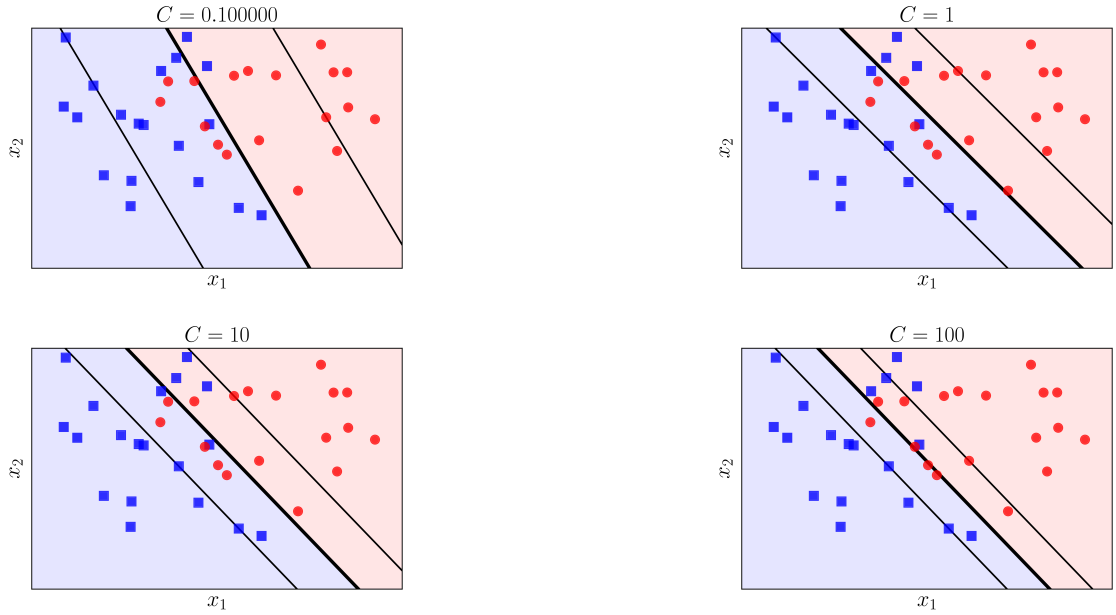
$$k(x,z) = \exp(-\gamma \|x - z\|_2^2), \quad \gamma > 0$$

4. Sigmoid:

$$k(x,z) = \tanh(\gamma x^T z + r)$$

5.2 Hằng số C

Hằng số C được dùng để điều chỉnh tầm quan trọng giữa margin và sự hy sinh. Hằng số này được xác định từ trước bởi người lập trình hoặc có thể được xác định bởi cross-validation. Hàng số C càng lớn cho biết mỗi sự hy sinh sẽ mang ý nghĩa rất lớn vì thế SVM sẽ ưu tiên giảm thiểu sự hy sinh hơn là tạo ra một margin lớn. Ngược lại C càng nhỏ thì tức là sự hy sinh sẽ bớt quan trọng đi từ đó SVM sẽ ưu tiên đi tìm một margin lớn hơn là giảm thiểu sự hy sinh.



Hình 15: Ảnh hưởng của C lên nghiệm của Soft Margin SVM

5.3 Gamma

Gamma là một tham số thể hiện tầm ảnh hưởng của các điểm dữ liệu huấn luyện. Nếu gamma càng nhỏ thì tức là tầm ảnh hưởng của các điểm dữ liệu sẽ xa và các điểm xa nhau cũng có thể được phân vào cùng một lớp. Ngược lại, nếu gamma nhỏ thì các điểm dữ liệu phải nằm gần nhau mới được phân vào cùng một lớp. Tuy gamma là thể hiện tầm ảnh hưởng các các điểm dữ liệu huấn luyện nhưng thực sự chỉ các support vector mới đóng góp trong việc tìm ra biên quyết định vì thế ta có thể hiểu gamma là tham số thể hiện tầm ảnh hưởng của các support vector.

5.4 Tham số degree

Degree là bậc của hàm Polynomial kernel. Nếu sử dụng các kernel khác thì tham số này sẽ không có tác động lên mô hình.
Giá trị mặc định: **int, default = 3**

5.5 coef0

Coef0 cho phép ta chỉnh sửa thuật ngữ độc lập trong hàm kernel. Nó chỉ có ý nghĩa trong hàm **'poly'** hoặc **'sigmoid'**. Coef0 có giá trị mặc định là 0.
Giá trị mặc định: **float, default=0.0**

5.6 Tham số shrinking

Tham số cho phép ta chọn có sử dụng hàm heuristic co lại hay không nó sẽ giúp giảm thời gian huấn luyện cho mô hình nếu như số lượng vòng lặp huấn luận quá nhiều.
Giá trị mặc định: **bool, default=True**

5.7 Probability

Tham số probability này sẽ kích hoạt chế độ đo theo xác suất của mô hình và có thể gọi hàm predict_proba để lấy xác suất phân lớp của từng lớp.
Giá trị mặc định: **bool, default=True**

5.8 Tham số tol

Tham số này là mốc dừng trong quá trình tối ưu hóa hàm mục tiêu của SVM. Nếu kết quả hàm mục tiêu giảm nhỏ hơn mức tol ở hai vòng lặp liên tiếp nhau thì thuật toán sẽ dừng và xem là đã hội tụ.
Giá trị mặc định: **float, default=1e-3**

5.9 Tham số cache_size

Tham số này điều chỉnh lượng bộ nhớ cache cung cấp cho kernel có tác dụng tăng tốc độ xử lý của mô hình lên nếu lượng dữ liệu quá lớn.
Giá trị mặc định: **float, default=200**

5.10 Tham số class_weight

Tham số này sẽ thay đổi giá trị tham số C của lớp thứ *i* thành class_weight[i]C. Nếu để trống thì toàn bộ class_weight của các lớp được gán bằng 1. Ngoài ra, tham số này còn có thể truyền vào một từ điển hoặc chọn chế độ **'balanced'** để gán trọng số tỉ lệ nghịch với số lượng điểm của lớp.
Giá trị mặc định: **dict or 'balanced', default=None**

5.11 Tham số verbose

Tham số này sẽ cho phép ta chọn có hiển thị thông tin của mô hình sau khi huấn luyện hoàn thành hay không. Nếu chọn hiển thị nó sẽ cho biết sau bao nhiêu vòng lặp thì hàm mục tiêu sẽ hội tụ và giá trị hàm mục tiêu là bao nhiêu, số lượng support-vector trong mô hình, ...
Giá trị mặc định: **bool, default=False**

5.12 Tham số max_iter

Tham số này giới hạn số vòng lặp trong quá trình tối ưu hàm mục tiêu. Ta có thể gán một số lượng vòng lặp nhất định hoặc không giới hạn số lượng vòng lặp bằng cách gán giá trị -1.
Giá trị mặc định: **int, default=-1**

5.13 Tham số decision_function_shape

Tham số này cho phép lựa chọn hình dạng của biên quyết định như đã đề cập ở phần Multi-Class Classification ở trên.
Giá trị mặc định: **'ovo', 'ovr', default='ovr'**

5.14 Tham số break_ties

Nếu chọn decision_function_shape = 'ovr' và số lớp lớn hơn 2 thì lúc phân chia sẽ có thể tạo ra 1 vùng mà ở đó các lớp sẽ trang chấp với nhau. Nếu gán tham số break_ties = False thì tất cả các điểm trong vùng tranh chấp sẽ được phân vào cùng 1 lớp. Ngược lại, nếu là True thì mô hình sẽ tạo ra biên quyết định non-convex trong vùng đó cho các lớp.
Giá trị mặc định: **bool, default=False**

5.15 Tham số random_state

Tham số này sẽ chọn một trạng thái ngẫu nhiên đã được cài đặt sẵn và toàn bộ các phép lấy mẫu ngẫu nhiên đều lấy tự bộ số cài sẵn. Điều này giúp cho mỗi lần chạy mô hình bộ số ngẫu nhiên không bị thay đổi giúp ta dễ dàng theo dõi hoạt động của mô hình. Giá trị mặc định: **int, RandomState instance or None, default=None**

6 Thực nghiệm

Để đánh giá thực tế khả năng của thuật toán phân lớp SVM, nhóm đã tiến hành thực nghiệm trên một dữ liệu y khoa về dự đoán nguy cơ bị đột quỵ (*Stroke Prediction Dataset*), nhằm xét kết quả dự đoán của mô hình thuật toán SVM cũng như khi áp dụng một số các kĩ thuật tinh chỉnh siêu tham số phổ biến trên bộ dữ liệu này.

6.1 Giới thiệu về bài toán

Đây là một bài toán phân lớp (Classification) nhằm xác định (phân lớp) bệnh nhân có bị đột quỵ hay không. Bộ dữ liệu này chứa thông tin của các bệnh nhân nhập viện cấp cứu và có chẩn đoán nghi ngờ đột quỵ, với xác suất thật sự bị đột quỵ vào khoảng $\frac{1}{20}$. Mục tiêu của mô hình xây dựng chính là xác định đến mức có thể tất cả những người thực sự bị đột quỵ và đồng thời tăng tối đa tỉ lệ $\frac{1}{X}$ giữa những người bị và không bị đột quỵ ($X > 20$) trong những người được mô hình dự đoán bị đột quỵ.

6.1.1 Kiến thức chung về đột quỵ

Tai biến mạch máu não hay đột quỵ hoặc nhồi máu não là một tình trạng y tế trong đó lưu lượng máu đến não giảm đi dẫn đến việc chết tế bào. Có hai loại đột quỵ chính: thiếu máu cục bộ, do thiếu lưu lượng máu và xuất huyết, do chảy máu. Cả hai kết quả là các phần của não không hoạt động được.

Đột quỵ do thiếu máu cục bộ thường gây ra do tắc nghẽn mạch máu, mặc dù cũng có những nguyên nhân ít phổ biến hơn. Đột quỵ xuất huyết là do chảy máu trực tiếp vào não hoặc vào khoảng trống giữa màng não. Chảy máu có thể xảy ra do phình động mạch não bị vỡ.

Yếu tố nguy cơ chính của đột quỵ là huyết áp cao. Các yếu tố nguy cơ khác bao gồm hút thuốc lá, béo phì, cholesterol trong máu cao, đái tháo đường, rung tâm nhĩ, các bệnh thận giai đoạn cuối và những yếu tố sinh lí khác. (nguồn - Tai biến mạch máu não – Wikipedia tiếng Việt)

Huyết áp cao, cholesterol cao, hút thuốc lá, béo phì, tiểu đường là những nguyên nhân hàng đầu dẫn đến đột quỵ. Cứ 3 người trưởng thành ở Mỹ thì có 1 người mắc ít nhất một trong những tình trạng hoặc thói quen này. Thông tin này có thể giúp ích khi xây dựng mô hình. (nguồn - https://www.cdc.gov/stroke/facts.htm)

6.2 Bộ dữ liệu

6.2.1 Thông tin thuộc tính của bộ dữ liệu

Bộ dữ liệu này chứa 5110 hàng thông tin, tương ứng với dữ liệu của 5110 bệnh nhân. Dữ liệu của bệnh nhân sẽ gồm 12 trường thông tin, cho ta biết được sự xuất hiện của một số yếu tố quan trọng ảnh hưởng đến nguy cơ đột quỵ: tăng huyết áp, mức độ chỉ số khối cơ thể, bệnh tim, mức đường trung bình, tình trạng hút thuốc, tuổi tác, và được mô hình dùng làm dữ liệu đầu vào để dự đoán nguy cơ bị đột quỵ của bệnh nhân, với trường dữ liệu label *stroke*.

Thông tin chi tiết: Dữ liệu gồm thông tin của 5110 bệnh nhân với 12 thuộc tính.

- *id*: Mã định danh độc nhất của bệnh nhân.
- *gender*: Dữ liệu giới tính với 3 giá trị phân loại: *Male*, *Female*, *Other*.
- *age*: Dữ liệu tuổi của bệnh nhân.
- *hypertension*: Thông tin về bệnh lí cao huyết áp của bệnh nhân. 0 nếu bệnh nhân không bị cao huyết áp, 1 nếu bệnh nhân bị cao huyết áp.
- *heart_disease*: Thông tin về bệnh tim của bệnh nhân. 0 nếu bệnh nhân không bị bệnh tim, 1 nếu bệnh nhân bị bệnh tim.
- *ever_married*: Dữ liệu kết hôn với 2 giá trị phân loại: *No*,*Yes*.
- *work_type*: Dữ liệu công việc của bênh nhận, 5 giá trị: *children*, *Govt_job*, *Never_worked*, *Private*, *Self¯employed*.
- *Residence_type*: Dữ liệu vùng dân cư của bệnh nhân: *Rural*, *Urban*.
- *avg_glucose_level*: Lượng đường trung bình trong máu của bệnh nhân.
- *bmi*: Mức độ chỉ số khối cơ thể của bệnh nhân.
- *smoking_status*: Thông tìn về tình trạng hút thuốc của bệnh nhân: *formerly smoked*, *never smoked*, *smokes*, *Unknown*.
- *stroke*: Thông tin nhãn dữ liệu, gồm 2 giá trị phân loại: 1 nếu bệnh nhân bị đột quỵ, 0 nếu không đột quỵ.

6.3 Mô tả và phân tích bộ dữ liệu

6.3.1 Thống kê cơ bản

Bộ dữ liệu thực tế sẽ chứa các thông tin như sau:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
...	...	...	...	...	...	...	...	...	...	...	...	...
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	0
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

Sau quá trình tiền xử lí dữ liệu, nhóm đã thực hiện một số thống kê cơ bản:

Thống kê trên bộ dữ liệu với người không bị đột quỵ

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	4861.000000	4861.000000	4861.000000	4861.000000	4861.000000	4700.000000	4861.0
mean	36487.236371	41.971545	0.088871	0.047110	104.795513	28.823064	0.0
std	21120.133386	22.291940	0.284586	0.211895	43.846069	7.908287	0.0
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.0
25%	17762.000000	24.000000	0.000000	0.000000	77.120000	23.400000	0.0
50%	36958.000000	43.000000	0.000000	0.000000	91.470000	28.000000	0.0
75%	54497.000000	59.000000	0.000000	0.000000	112.830000	33.100000	0.0
max	72940.000000	82.000000	1.000000	1.000000	267.760000	97.600000	0.0

và kết quả thống kê với tập những người bị đột quỵ

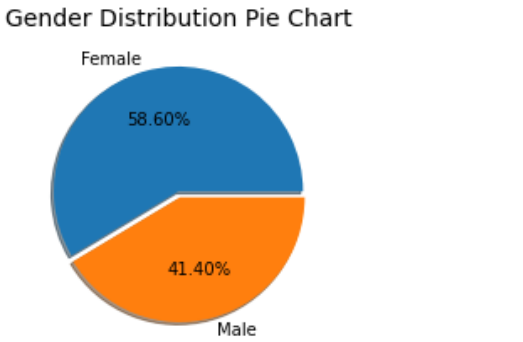
	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	249.000000	249.000000	249.000000	249.000000	249.000000	209.000000	249.0
mean	37115.068273	67.728193	0.265060	0.188755	132.544739	30.471292	1.0
std	21993.344872	12.727419	0.442254	0.392102	61.921056	6.329452	0.0
min	210.000000	1.320000	0.000000	0.000000	56.110000	16.900000	1.0
25%	17013.000000	59.000000	0.000000	0.000000	79.790000	26.400000	1.0
50%	36706.000000	71.000000	0.000000	0.000000	105.220000	29.700000	1.0
75%	56669.000000	78.000000	1.000000	0.000000	196.710000	33.700000	1.0
max	72918.000000	82.000000	1.000000	1.000000	271.740000	56.600000	1.0

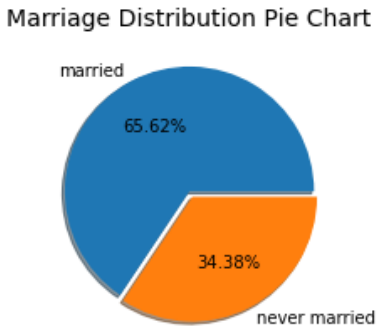
Có thể thấy sự khác biệt lớn giữa hai phân lớp đột quỵ và không đột quỵ khi độ tuổi trung bình, lượng đường trung bình, và số người bị bệnh tim và cao huyết áp, đều cao hơn đáng kể ở nhóm những người bị đột quỵ so với người không bị đột quỵ

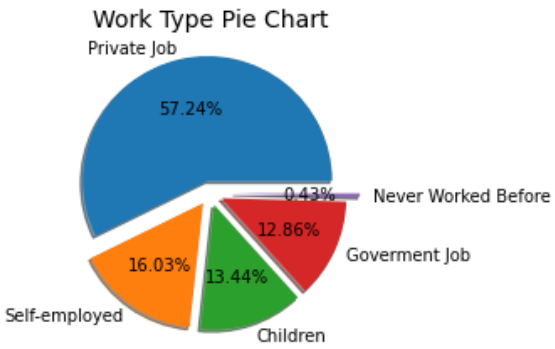
6.3.2 Trực quan hóa bộ dữ liệu

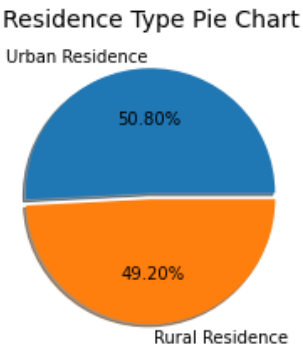
Nhóm sau đó đã tiến hành trực quan hóa bộ dữ liệu để thể hiện rõ hơn sự phân bố và cá mối liên hệ giữa các điểm và trường dữ liệu, với biểu đồ bánh thể hiện phân bố dữ liệu với các trường dữ liệu phân loại:

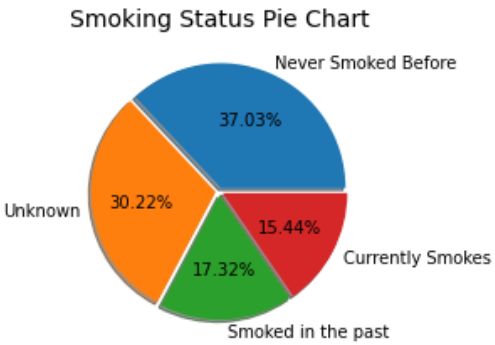
- Giới tính
- Kết hôn
- Vùng dân cư
- Công việc
- Bệnh tim
- Hút thuốc
- Cao huyết áp
- Đột quỵ

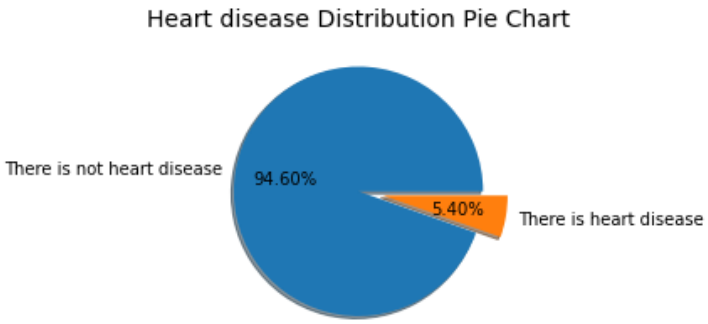


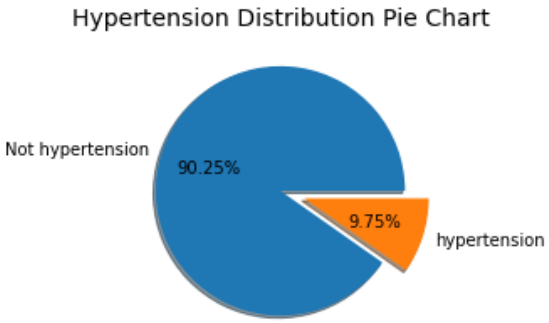


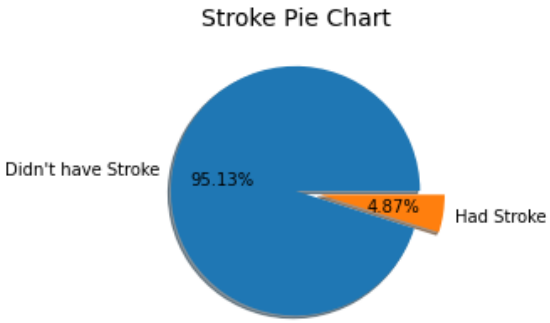






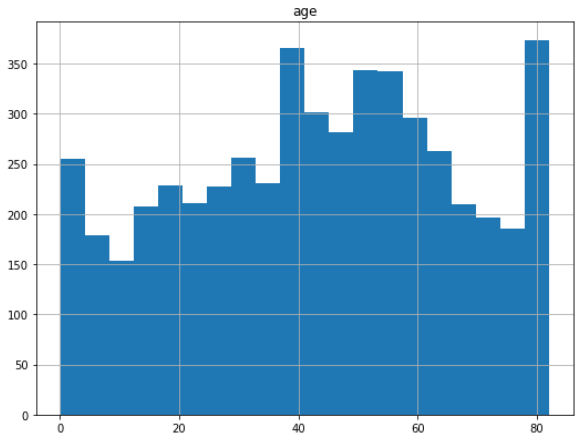


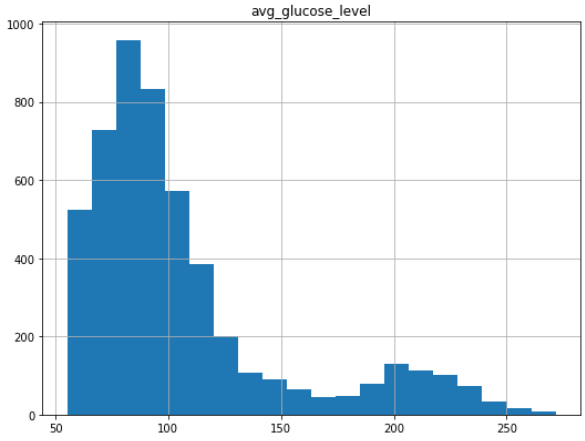


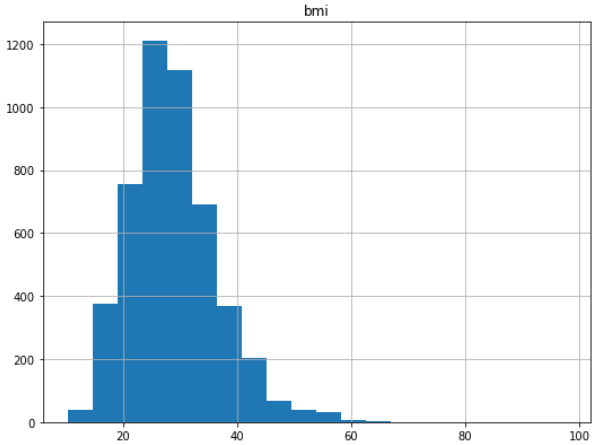


và biểu đồ histogram cho các trường dữ liệu số:

- Tuổi
- BMI
- Mức đường trung bình



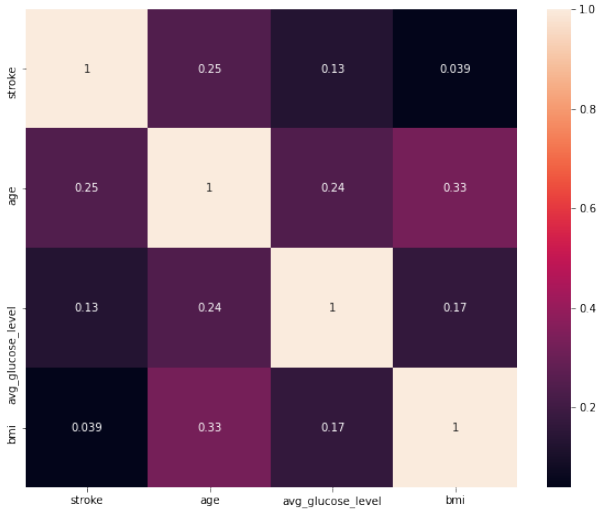




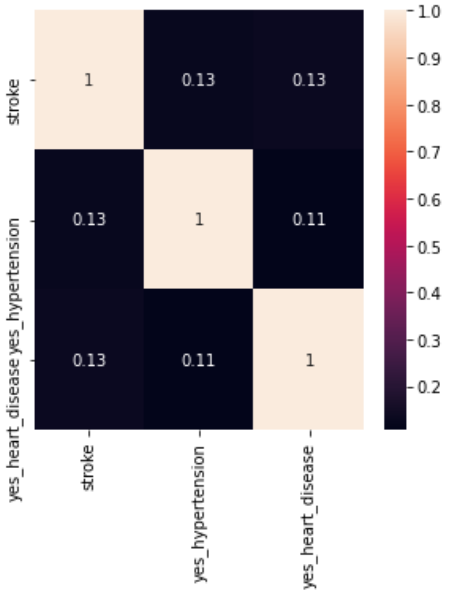
6.3.3 Phân tích dữ liệu

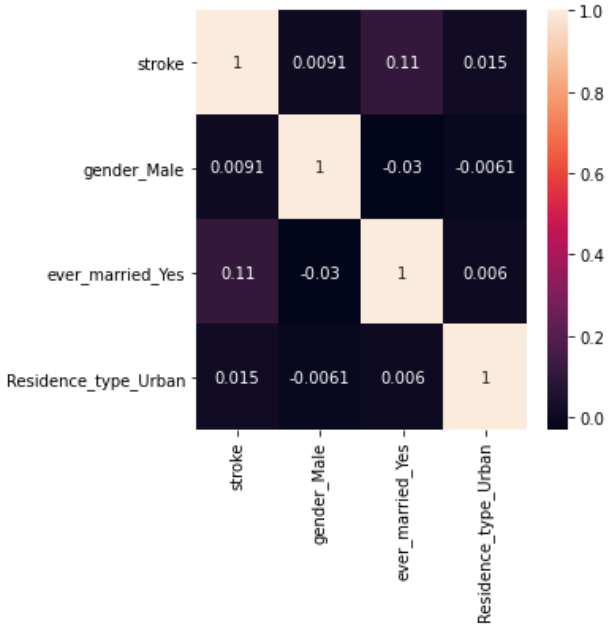
Từ phân phối, rõ ràng cứ 5 người trong số 100 người thì bị đột quỵ từ dữ liệu lấy mẫu và là một phân phối dữ liệu không cân bằng. Để có được cái nhìn sơ lược về quan hệ giữa các trường dữ liệu với nhau, ta tìm ma trận hệ số tương quan tuyến tính Pearson:

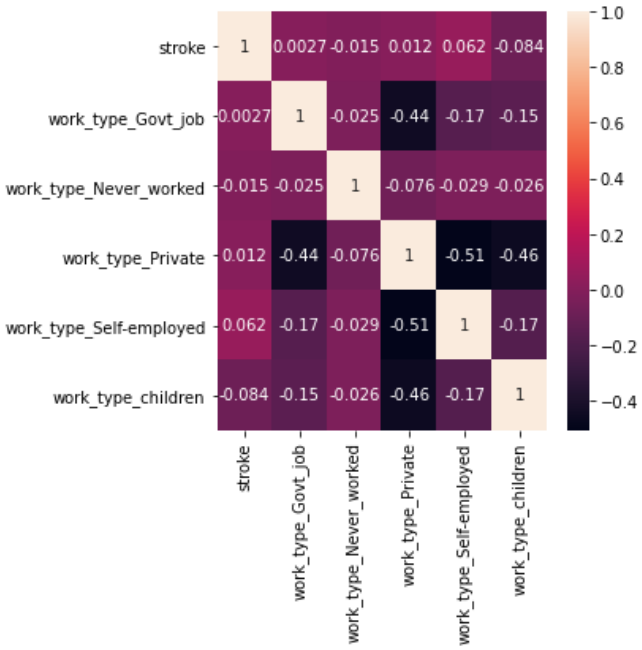
Heatmap hệ số tương quan của các trường giá trị số với thuộc tính đột quỵ:

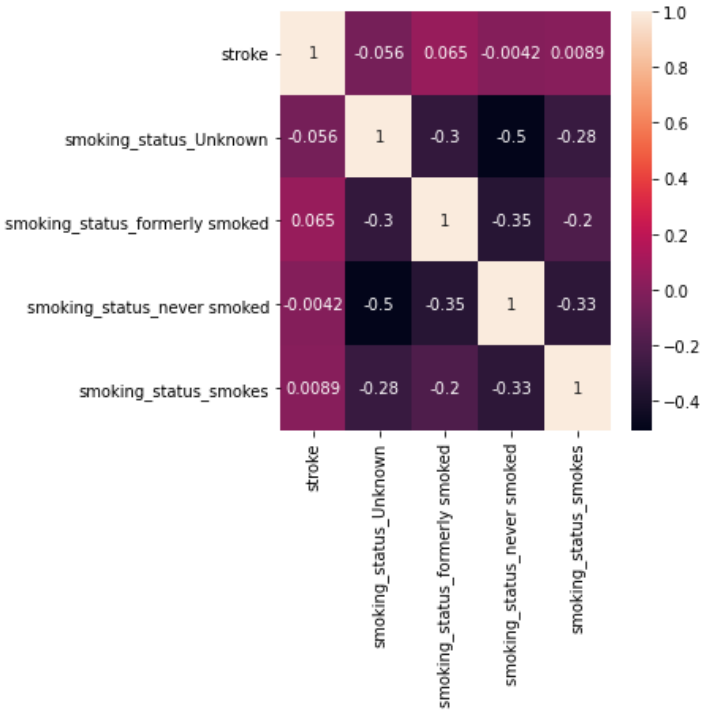


Heatmap hệ số tương quan giữa các cặp giá trị của trường dữ liệu phân lớp với thuộc tính đột quỵ:



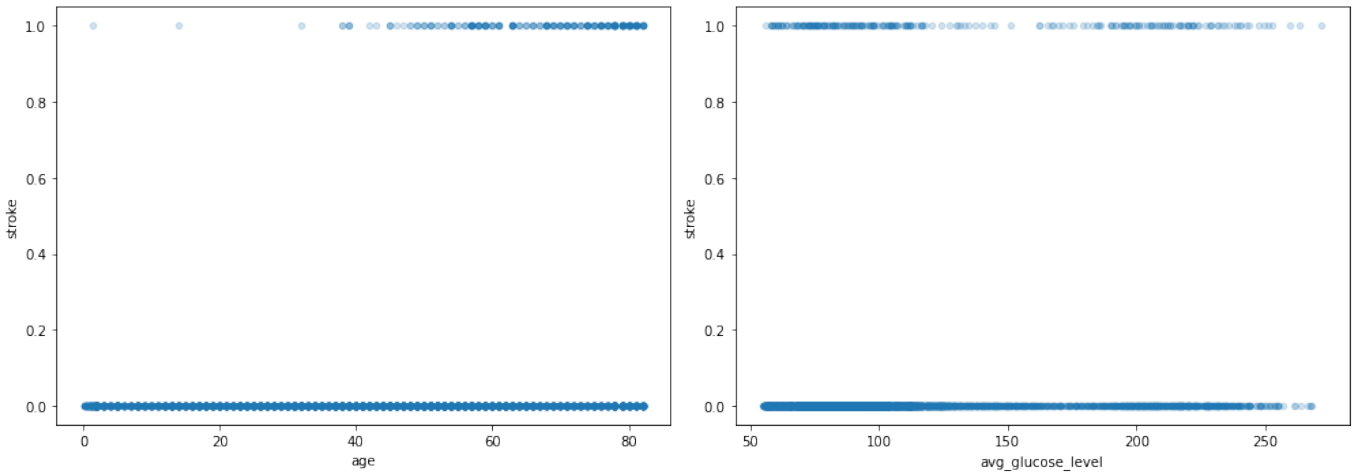






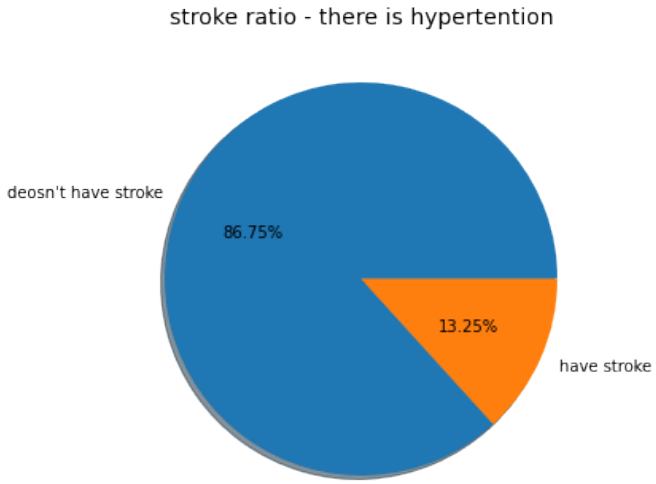
Ta có thể thấy những cặp thuộc tính có tính tương quan với đột quỵ có hệ số > 0.1 bao gồm: Tuổi, mức đường trung bình, cao huyết áp, bệnh tim và kết hôn. Khi đi vào đánh giá từng cặp thuộc tính dữ liệu với nhau, chúng ta bắt đầu xem xét ảnh hưởng của chúng đến nguy cơ đột quỵ.

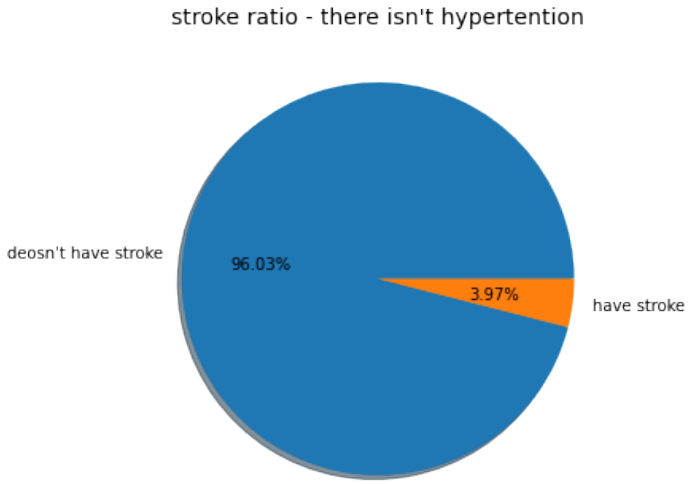
Với các trường dữ liệu có giá trị số: tuổi và mức đường trung bình



Có thể kết luận sơ bộ rằng yếu tố tuổi càng cao càng làm tăng nguy cơ đột quỵ; và mức đường trong máu an toàn nằm ở mức từ 120 đến 180, ngoài ra thì tăng hay giảm quá lớn đều có nguy cơ dẫn đến đột quỵ cao.

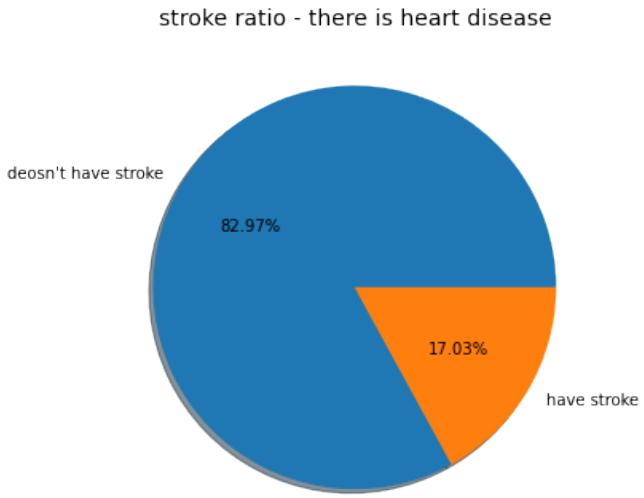
Cao huyết áp

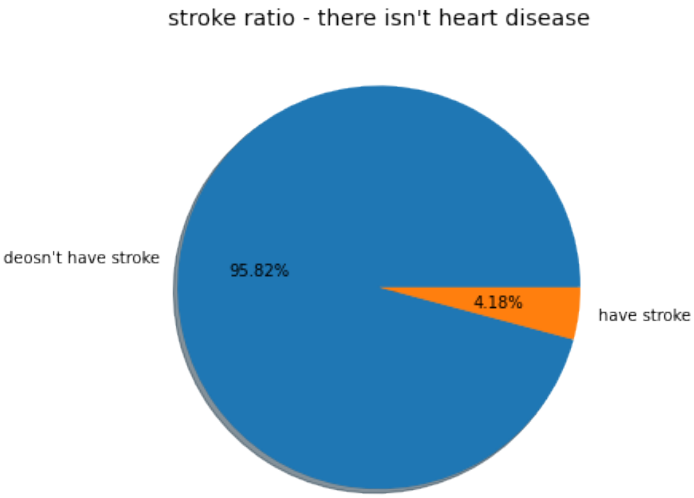




Bị cao huyết áp làm tăng nguy cơ đột quỵ từ 3.97% lên 13.25%, gấp hơn 3 lần so với bình thường.

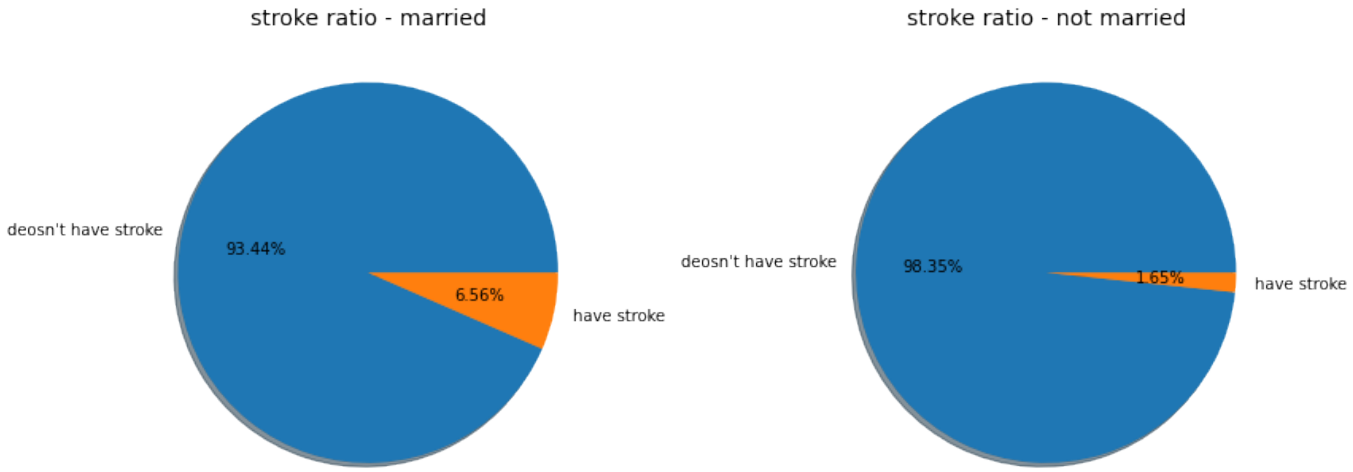
Bệnh tim nền





Người có bệnh nền tim mạch có nguy cơ đột quỵ cao hơn người không có bệnh gấp 4 lần, từ 4.18% lên 17.03%.

Kết hôn



Người đã kết hôn có nguy cơ bị đột quỵ cao hơn 4.91% với người chưa kết hôn.

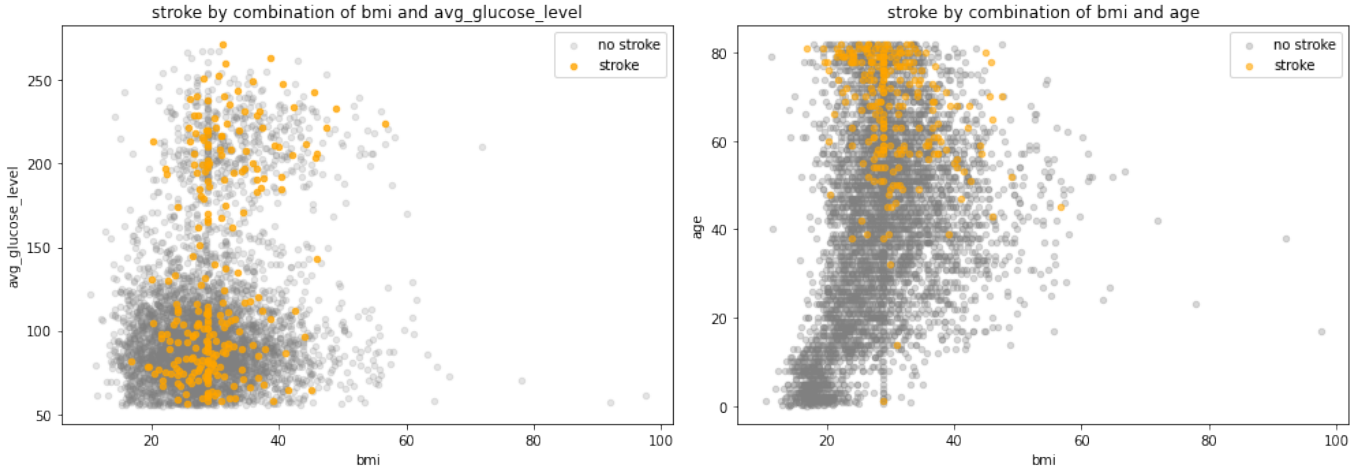
BMI

Tuy BMI không thể hiện tương quan rõ rệt với nguy cơ đột quỵ, nhưng khi xét đến hệ số tương quan giữa BMI với các thuộc tính còn lại của bộ dữ liệu, đồng thời đánh giá sự phân bố của các trường hợp đột quỵ với độ đo BMI, chúng ta cũng có thể nhận ra những sự liên kết nhất định.
Với kết quả hệ số tương quan giữa BMI và các trường giá trị > 0.15, ngưỡng để xem xét các thuộc tính có tương quan:

```
ever_married_Yes     0.335563
age                  0.325858
work_type_Private    0.204055
avg_glucose_level    0.168910
yes_hypertension     0.160147
```

Ta thấy xuất hiện hai thuộc tính tuổi và mức đường trung bình, các trường giá trị số mà ta có thể khảo sát thêm để tìm ra quy luật cho dữ liệu.

Và khi ta dựng scatter plot giữa BMI và hai trường giá trị tuổi và mức đường trung bình:

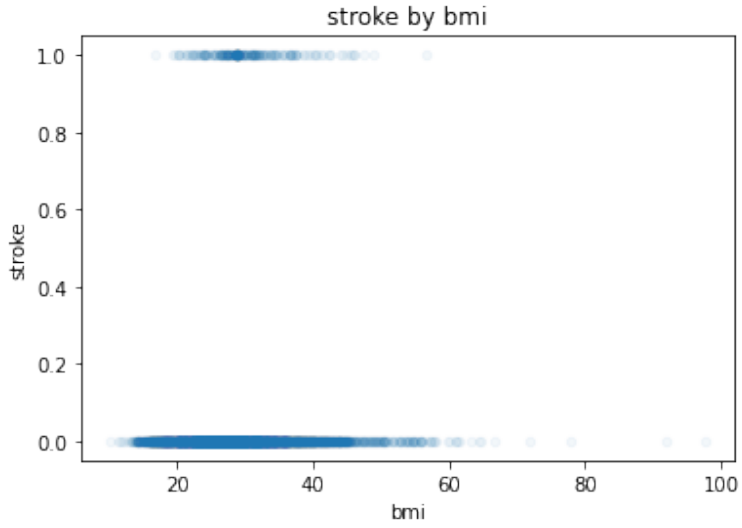


Với cặp thuộc tính BMI và tuổi, ta thấy rõ được sự tập trung của các điểm dữ liệu là trường hợp đột quỵ nằm trong vùng BMI $+-$ 30 với khoảng tuổi từ 50 trở lên. Đây là cặp thuộc tính có tính tương quan rất lớn đến việc xác định đột quỵ trong bộ dữ liệu này.

Với cặp thuộc tính BMI và mức đường trung bình, ta thấy được hai cụm tập trung các trường hợp đột quỵ, đều nằm trong khoảng BMI từ 20 đến 40 với tâm nằm ở BMI = 30, và mức đường nằm ở hai cực giá trị thấp và cao hơn khoảng từ 140 đến 160, tương đối giống với quan sát trước đó của chúng ta về mức đường trung bình.

Bên cạnh đó, scatter plot giữa BMI và các ca đột quỵ, có thể thấy mật độ tập trung cao nhất những trường hợp đột quỵ là ở BMI

trong khoảng miền giá trị 30, với các trường hợp đột quỵ với BMI trong khoảng giá trị 30 chiếm hơn 50% tổng số các trường hợp đột quỵ đã xảy ra.




```
-+30bmi with stroke cases : all stroke cases (rati
o) = 0.5180722891566265
```

6.3.4 Kết luận phân tích dữ liệu

Đầu tiên, nhóm đã xác định được trường dữ liệu có tính ảnh hưởng lớn đến việc xác định đột quỵ, với hệ số tương quan với thuộc tính nhãn *stroke* > 0.10 là tuổi, mức đường trung bình, bệnh tim nền và tình trạng kết hôn của bệnh nhân.

Quan trọng hơn, nhóm còn rút ra được những đặc điểm nổi bật với các điểm dữ liệu thuộc lớp đột quỵ và các trường giá trị như:

- Tuổi cao, kết hợp với độ đo BMI trong miền giá trị từ 20 đến 40, xoay quanh 30 là một chỉ báo khá chính xác cho các trường hợp đột quỵ.
- Mức đường trung bình nằm ngoài ngưỡng giá trị từ 120 đến 170 làm tăng nguy cơ đột quỵ.
- Hơn 50% số ca đột quỵ có giá trị BMI nằm quanh mức 30.
- Bị cao huyết áp làm tăng nguy cơ đột quỵ từ 3.97% lên 13.25%, gấp hơn 3 lần.
- Người có bệnh nền tim mạch có nguy cơ đột quỵ cao hơn người không có bệnh gấp 4 lần, từ 4.18% lên 17.03%.
- Người đã kết hôn có nguy cơ bị đột quỵ cao hơn 4.91% với người chưa kết hôn.

Ngoài ra có thể nhận thấy rằng bộ dữ liệu của nhóm bị mất cân bằng nghiêm trọng giữa các class . Nên nhóm sẽ sử dụng một phương pháp lấy mẫu để tăng cường dữ liệu là SMOTE (Synthetic Minority Oversampling TEchnique)

6.3.5 Sử dụng SMOTE để giải quyết mất cân bằng dữ liệu

Kỹ thuật SMOTE được giới thiệu bởi (Chawla, 2002). SMOTE là một kỹ thuật dựa trên các lân cận gần nhất được đánh giá bởi Euclidean distance giữa các điểm dữ liệu trong không gian đặc trưng của chúng. SMOTE hoạt động bằng cách chọn các điểm gần nhau trong không gian đối tượng và vẽ một đường thẳng giữa các điểm đối trong không gian đối tượng và tạo ra một mẫu mới trên đường thẳng mới
Theo mô tả của Chawla, class chiếm số lượng ít sẽ được lấy mẫu quá mức bằng cách lấy từng mẫu của từng lớp chiếm số lượng ít trong bộ dữ liệu bằng việc lấy các điểm dữ liệu ở trên đường thẳng được tạo ra từ các điểm thuộc lớp chiếm số lượng ít ở trên, việc lấy sẽ dựa trên k các lận cận gần nhất một cách ngẫu nhiên Nhược điểm chung của phương pháp này là các dữ liệu sau được tổng hợp được tạo ra mà không xem xét đến lớp đa số, có thể dẫn đến các điểm không rõ ràng nếu có sự chồng chéo mạnh giữa các lớp.

6.4 Hyper Parameter Tuning

Qua thực nghiêm trên dữ liệu của nhóm, nhóm nhận thấy có 3 siêu tham số có ảnh hướng rõ ràng nhất đối với mô hình, dó đó nhóm quyết định lựa chọn tập siêu tham số C,gamma, kernel để sử dụng cho việc tối ưu mô hình trên các tập dữ liệu. Miền giá trị của các siêu tham số được nhóm sử dụng đê tìm kiếm tối ưu như sau:

```
paramgrid= {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['linear','rbf',"sigmoid"]}
```

Trong quá trình tối ưu nhóm sẽ sử dụng toàn bộ dữu liệu và sử kfold cross validation với số **flold=5** để có thể đánh giá hiệu năng model một cách rõ ràng nhất

6.4.1 Phương pháp tối ưu hóa bằng Grid Search

Tìm kiếm theo lưới (Grid Search) là một cách truyền thống để thực hiện tối ưu hóa siêu tham số. Nó hoạt động bằng cách tìm kiếm toàn diện thông qua một tập con siêu tham số được chỉ định. Lợi ích của tìm kiếm lưới (grid search) là nó đảm bảo để tìm ra sự kết hợp tối ưu của các tham số được cung cấp. Hạn chế là nó rất tốn thời gian và tính toán tốn kém (Bảng số liệu)

6.4.2 Phương pháp tối ưu hóa bằng Random Search

Tìm kiếm ngẫu nhiên (Ramdom Search) khác với Grid Search chủ yếu ở chổ nó tìm kiếm tập hợp con được chỉ định của các siêu tham số một cách ngẫu nhiên thay vì toàn bộ các trường hợp. Lợi ích rõ ràng nhất là giảm thời gian xử lý tuy nhiên điểm hạn chế là phương pháp này không đảm bảo sẽ tìm ra sự kết hợp tối ưu của các siêu tham số của mô hình.

6.4.3 Phương pháp tối ưu hóa bằng GA Search

Ý tưởng của thuật toán di truyền là ta sẽ học cách tối ưu các tham số dựa trên các cá thể có chất lượng tốt để tiến hóa và lai ghép với nhau nhằm đạt các cá thể đời con có kết quả tốt nhất.

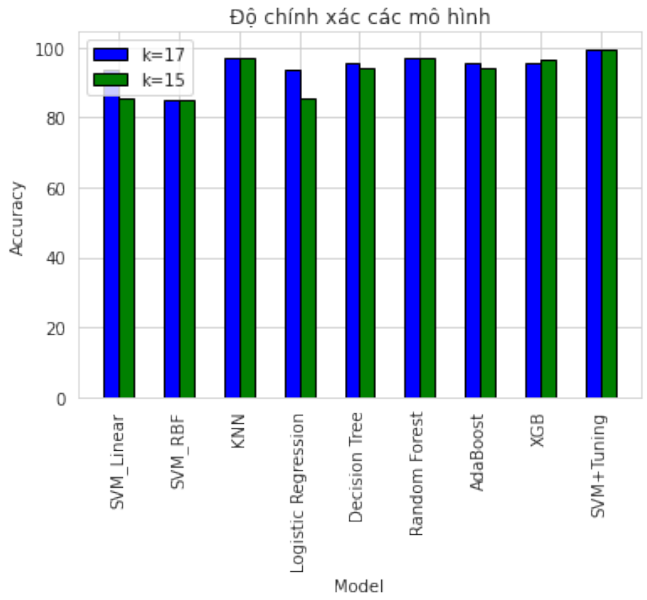
6.4.4 Kết quả thực nghiệm

Ở bài toán trên nhóm sẽ thực hiện các phần như sau. Huấn luyện các mô hình máy học khác nhau so với SVM chưa tuning tham số, sau đó là so sánh kết quả tuning các model SVM dựa trên các Search khác nhau.

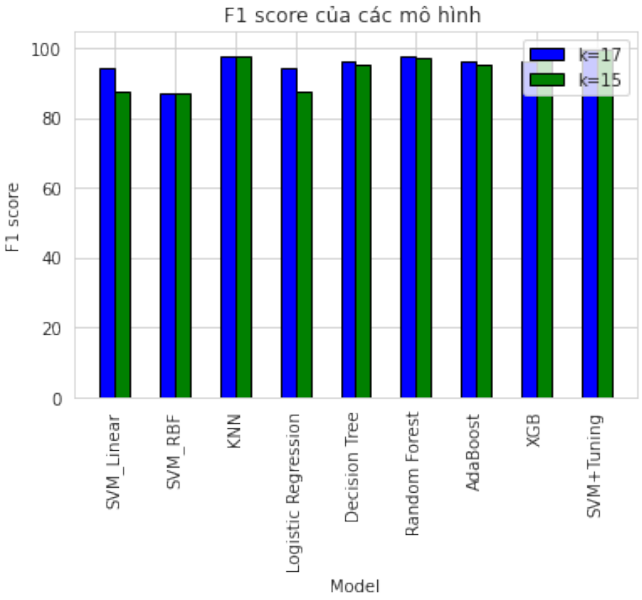
	C	gamma	kernel	acc_score	total_time
Grid Search	1	0.1	rbf	99.7	2.2s
Random Search	100	0.1	rbf	99.7	2.3s
GA Search	100	0.001	rbf	96.5	0.7s

Model	K=17	K=15	K=17	K=15
	ACC	ACC	F1	F1
SVM_Linear	93.77	85.66	94.38	87.63
SVM_RBF	85.05	85.05	87	87.22
KNN	97	97	97.39	97.39
Logistic Regression	93.61	85.58	94.25	87.54
Decision Tree	95.47	94.37	95.97	95.01
Random Forest	97.01	97.27	97.33	96.96
AdaBoost	95.55	94.28	96	94.92
XGB	95.55	96.71	96.08	97.05
SVM + Tuning	**99.43**	**99.51**	**99.49**	**99.56**

Nhận xét: Ta có thể thấy rằng kết quả ban đầu của SVM khi chưa tuning là chưa tốt bởi vì các tham số mặc định của SVM sẽ ưu tiên việc tạo ra các margin lớn (giúp cho tăng tương thích với nhiều trường hợp data), vì thế nó sẽ làm cho margin thật lớn và sẽ chấp nhận bỏ qua nhiều điểm dữ liệu dẫn đến độ chính xác thấp.



Hình 16: Độ chính xác của các mô hình



Hình 17: F1-Score của các mô hình

6.4.5 Thực nghiệm với mô hình SVM phân loại đa lớp (Multiclass SVM)

Ngoài kết quả với việc thực nghiệm trên bộ dữ liệu phân lớp nhị phân, nhóm đã tiến hành thực nghiệm thêm với bộ dữ liệu phân loại các loại kính (*Glass*; *Identification*; *Dataset*) nhằm khảo sát và so sánh kết quả của mô hình thuật toán SVM với các vấn đề phân loại đa lớp.

6.4.6 Giới thiệu về bộ dữ liệu và thông tin thuộc tính

Đây là một bài toán phân lớp nhằm xác định (phân lớp) chất liệu kính với một trong bảy loại kính khác nhau. Bộ dữ liệu chứa này sẽ chứa 214 hàng thông tin, tương ứng với dữ liệu của 214 đơn vị kính được khảo sát. Các dữ liệu này chính là các tính chất của kính, bao gồm chiết suất và tỉ lệ hàm lượng của các thành phần hóa học tạo thành hợp chất kính như ma-giê, natri, nhôm, sắt và các thành phần khác, được mô hình dùng làm dữ liệu đầu vào để dự đoán phân lớp kính của dữ liệu được khảo sát, với trường dữ liệu *Type*. Thông tin các trường thuộc tính:

- *RI*: Độ đo chiết suất của kính (*Refractive*; *Index*)
- *Na*: Hàm lượng Natri dưới dạng tỉ lệ phần trăm thành phần của hợp chất kính
- *Mg*: Hàm lượng Ma-giê dưới dạng tỉ lệ phần trăm thành phần của hợp chất kính
- *Al*: Hàm lượng nhôm dưới dạng tỉ lệ phần trăm thành phần của hợp chất kính
- *Si*: Hàm lượng Silic dưới dạng tỉ lệ phần trăm thành phần của hợp chất kính
- *K*: Hàm lượng Kali dưới dạng tỉ lệ phần trăm thành phần của hợp chất kính
- *Ca*: Hàm lượng Canxi dưới dạng tỉ lệ phần trăm thành phần của hợp chất kính
- *Ba*: Hàm lượng Bari dưới dạng tỉ lệ phần trăm thành phần của hợp chất kính
- *Fe*: Hàm lượng sắt dưới dạng tỉ lệ phần trăm thành phần của hợp chất kính
- *Type*: Thông tin nhãn dữ liệu, gồm 7 giá trị phân loại:
    1. building_windows_float_processed
    2. building_windows_non_float_processed
    3. vehicle_windows_float_processed
    4. vehicle_windows_non_float_processed
    5. containers
    6. tableware
    7. headlamps

Chú ý: Trong bộ dữ liệu này không có điểm dữ liệu nào thuộc phân lớp 4.

6.4.7 Mô tả và phân tích cơ bản bộ dữ liệu

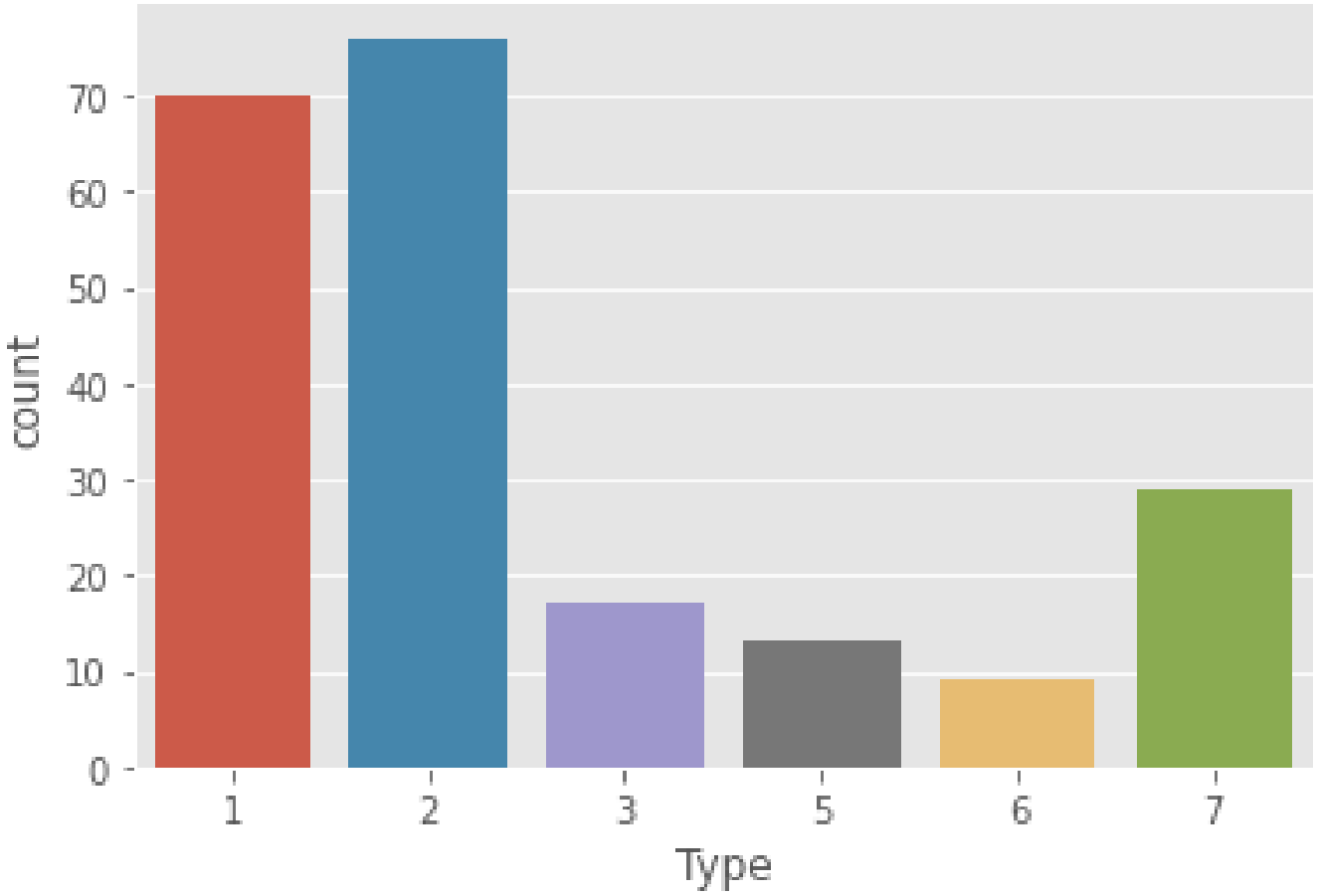
Bộ dữ liệu thực tế sẽ chứa các thông tin như sau:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

Thống kê sơ bộ trên bộ dữ liệu:

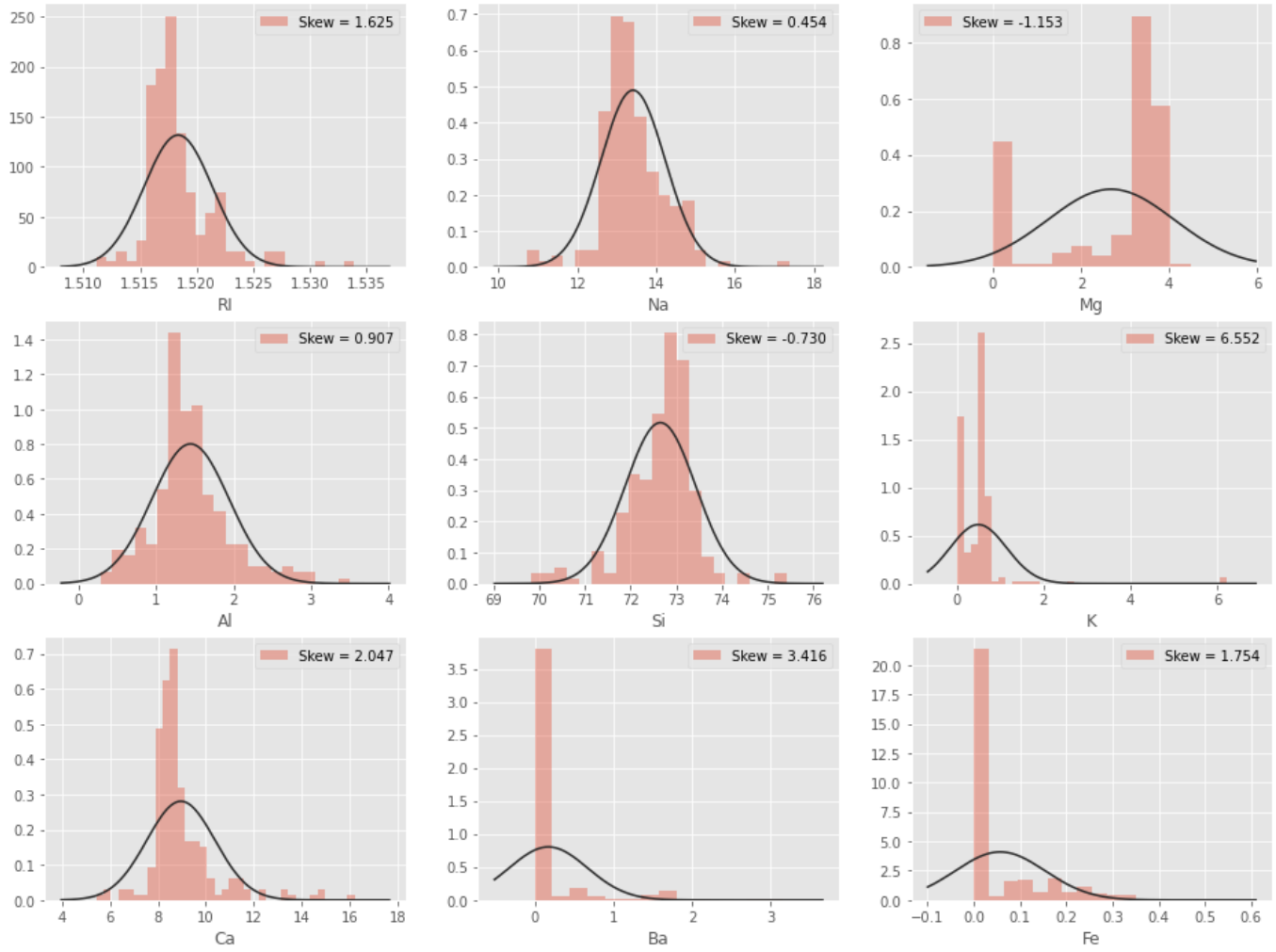
	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.175047	0.057009	2.780374
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.497219	0.097439	2.103739
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000	1.000000
25%	1.516523	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.000000	0.000000	1.000000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.000000	0.000000	2.000000
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.000000	0.100000	3.000000
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000	7.000000

Và khảo sát phân bố lớp của bộ dữ liệu:

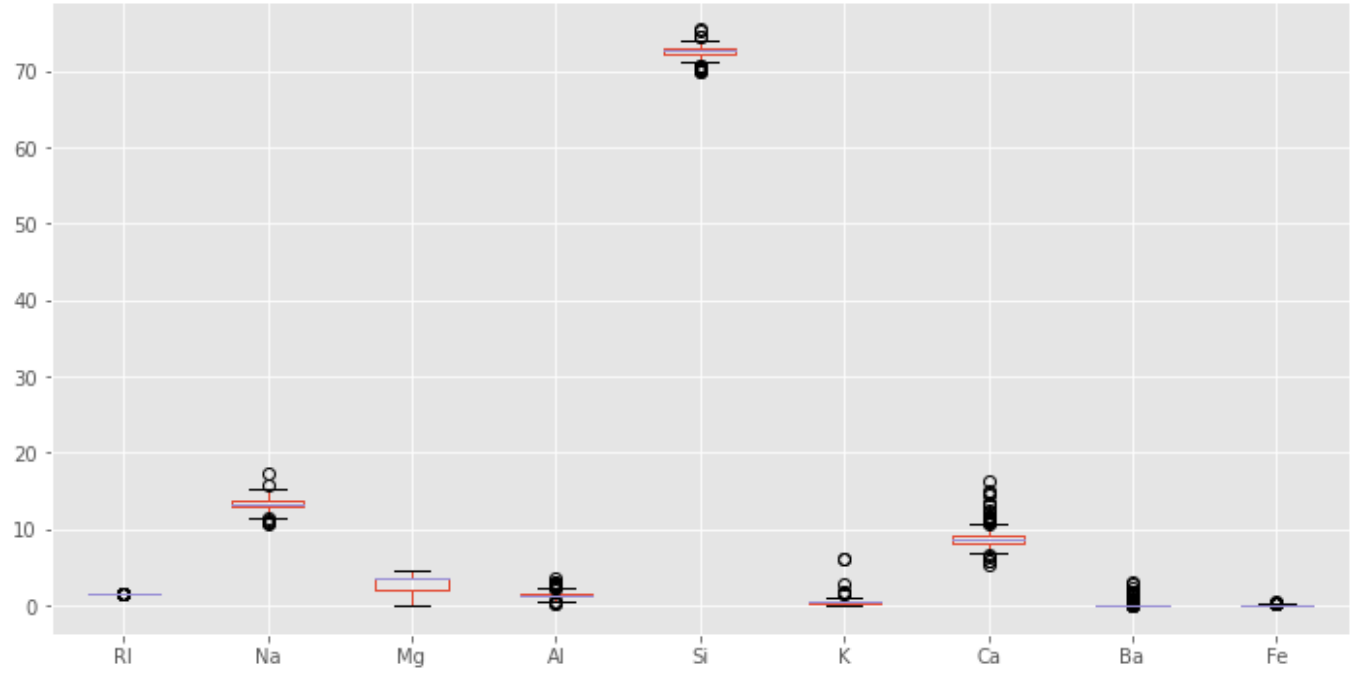


Có thể thấy, số liệu của các trường thuộc tính có cường độ khác nhau, ví dụ như trung bình của thành phần Silic và sắt lần lượt là 72.65 và 0.057, vì thế nhóm có thể phải cần chuẩn hóa dữ liệu. Bên cạnh đó, đây là một dữ iệu có phân bố lớp không đồng đều, khi phân lớp kính 1 và 2 chiếm tỉ lệ cao nhất các điểm dữ liệu, hơn 67% trên toàn bộ dữ liệu, và các lớp 3, 5, 6 có tỉ lệ thấp nhất.

Khảo sát phân bố của các thuộc tính trong bộ dữ liệu:

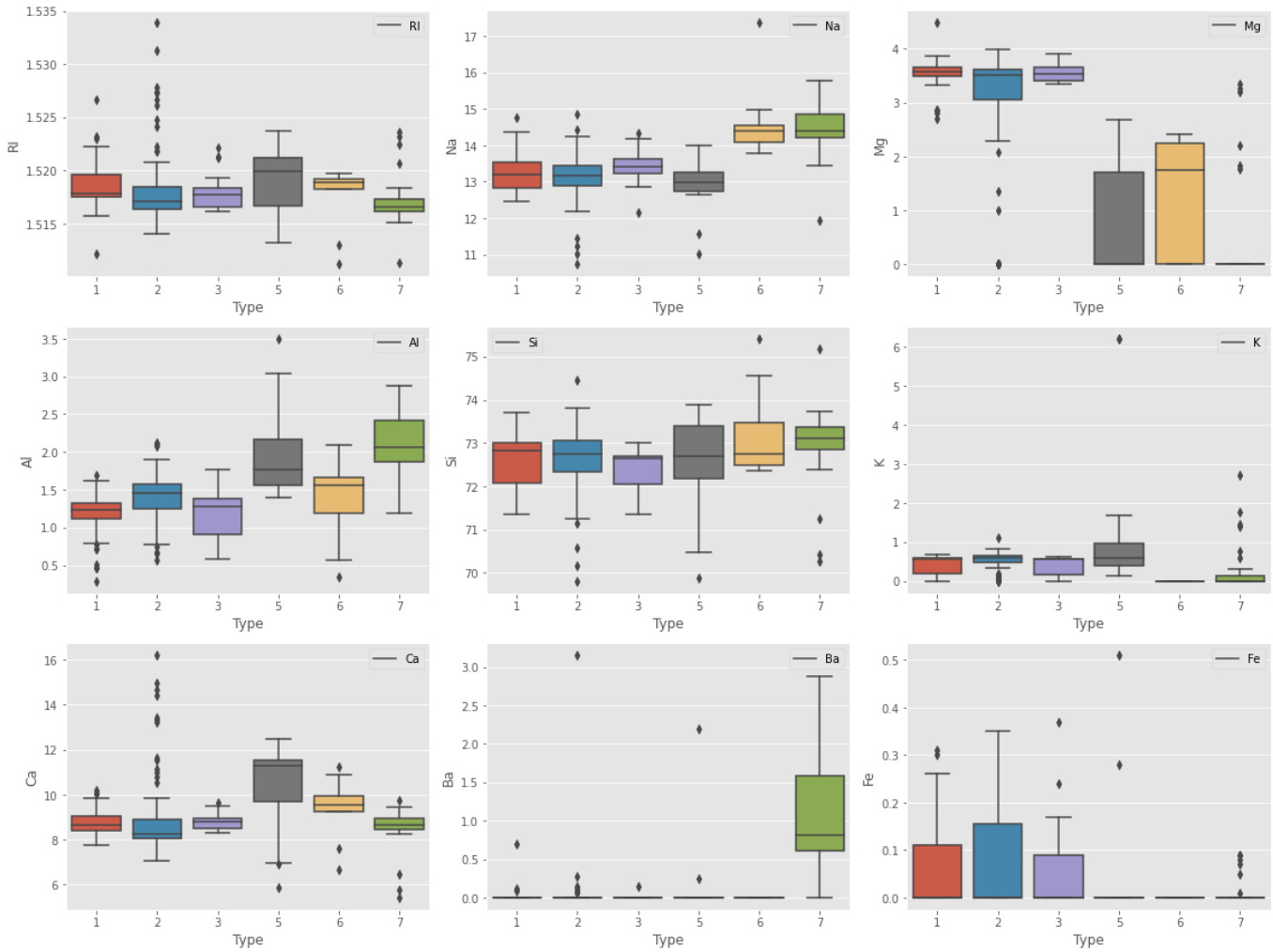


Và thể hiện phân bố này qua biểu đồ hộp:



Thì có thể quan sát được không có thuộc tính nào có phân phối chuẩn. Các trường Fe, Ba, Ca và K có độ phân bố lệch nhất, và với Silic có độ phân bố chuẩn nhất khi mà thành phần chính của hầu hết các loại kính là Silic. Tỉ trọng Silic, Natri và Canxi chiếm gần 90% tổng hợp thành của kính, với sắt là thành phần ít tỉ trọng nhất.

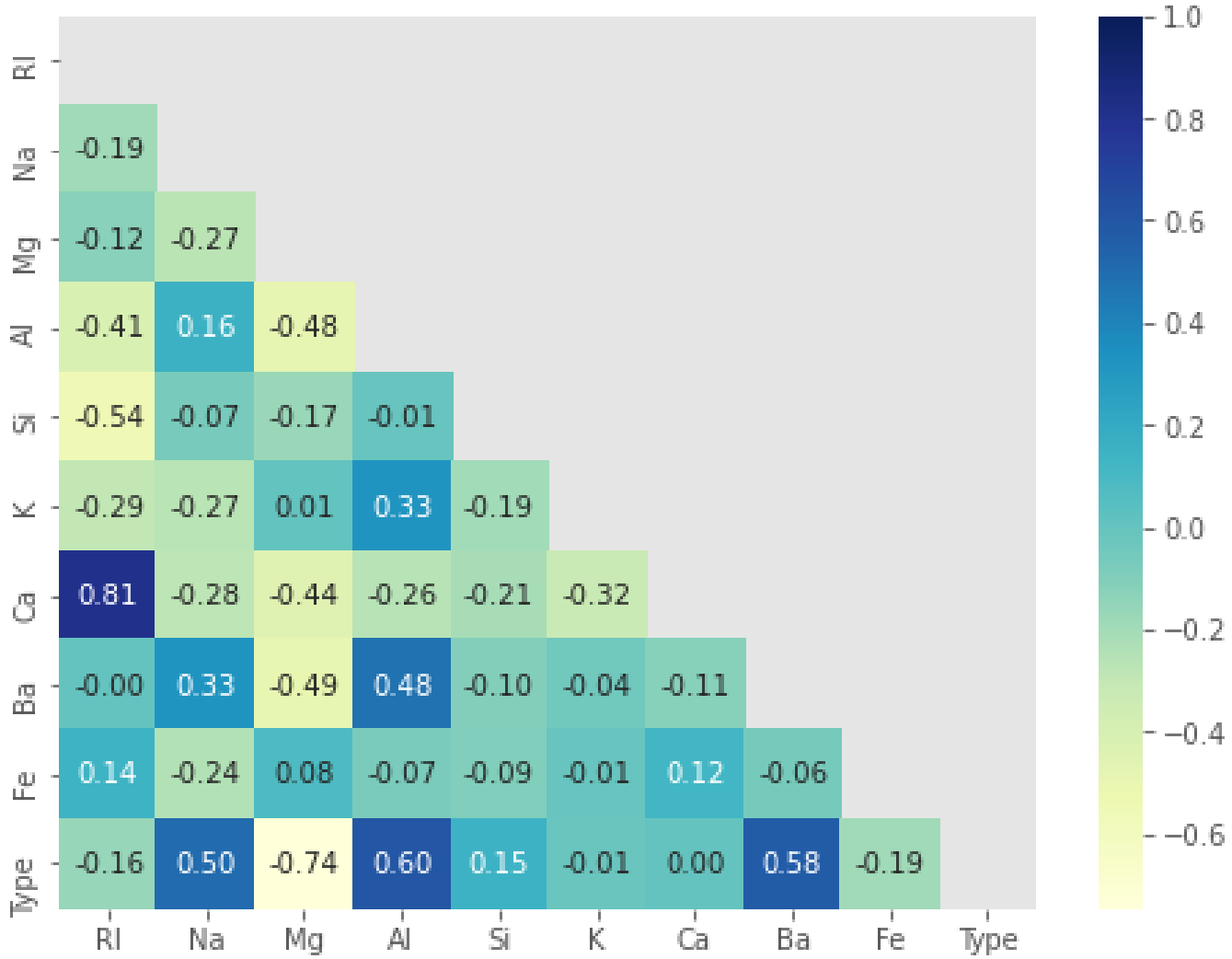
Và khi biểu diễn sự phân bố của các phân lớp theo thuộc tính qua biểu đồ hộp:



Những quan sát có thể rút ra được từ đây:

- Độ chiết suất có phân bố giao động từ khoảng 1.51 đến 1.54.
- Kính loại 6 và 7 có giá trị Na cao rõ rệt hơn các loại kính khác.
- Kính loại 1,2 và 3 có giá trị Mg cao hơn vượt trội so với các loại kính còn lại.
- Kính loại 5 và 7 có hàm lượng Al cao với miền giá trị tối đa gần gấp đôi các loại kính còn lại.
- Phần trăm Si phân bố tương đồng ở tất cả loại kính.
- Kính loại 6 không chứa K.
- Kính loại 5 và 6 có hàm lượng Ca cao hơn, đặc biệt là kính loại 5.
- Ba là một chỉ báo cho kính loại 7, với một số các điểm dữ liệu ngoại lệ ở các loại kính khác, trừ kính loại 6.
- Và Fe thường được dùng cho kính loại 1, 2 và 3.

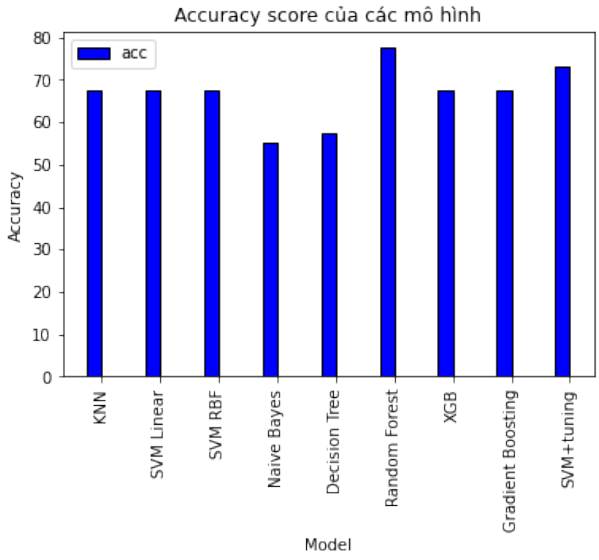
Và cuối cùng, khi xét hệ số tương quan để xác định hiệp phương sai hay mức ảnh hưởng giữa các thuộc tính với nhau:



Ta có thể thấy, các thuộc tính Na, Al và Ba là những thuộc tính ảnh hưởng đến phân lớp nhiều nhất, bên cạnh những phân tích ở trên.

6.4.8 Kết quả thực nghiệm SVM phân loại đa lớp

Dưới đây là kết quả các model phân loại đa lớp của nhóm. Nhóm chỉ sử dụng accuracy score để đánh giá vì bộ dữ liệu glass cho multi class có 6 class và dữ liệu ít bị mất cân bằng:



	C	gamma	kernel	acc_score
Grid Search	100	1	rbf	73.125
Random Search	100	1	rbf	73
GA Search	100	1	rbf	73.125

K Nearest Neighbor	67.5
Support Vector Machine (Linear Classifier)	67.5
Support Vector Machine (RBF Classifier)	67.5
Gaussian Naive Bayes	55
Decision Tree Classifier	57.5
Random Forest Classifier	**77.5**
Xgboost Classifier	67.5
GradientBoosting Classifier	67.5
SVM tuning	73.125

Ta có thể thấy mặc dù việc tuning tham số giúp cho SVM tốt hơn trong việc phân loại nhưng so với Random Forest vẫn chưa tốt bằng vì bộ dữ liệu trên có nhiều class dẫn đến khả năng tạo ra các margin để phân loại là chưa quá tốt vì phải hi sinh nhiều điểm dữ liệu

TÀI LIỆU THAM KHẢO

[0] Tiep, V. H. (2017, April 9). *Support Vector Machines*. Machine Learning Co Ban.
https://machinelearningcoban.com/2017/04/09/smv/

[1] Raoof G., & Nikoo F. (2017). *Handbook of neural computation* [E-book]. Sci-hub.
Chapter 27 - Support Vector Machine: Principles, Parameters, and Applications

[2] Alexandre, K. (2017, October 23). *Support Vector Machines Succinctly* [E-book].
https://www.syncfusion.com/ebooks/support_vector_machines_succinctly/
https://github.com/SyncfusionSuccinctlyE-Books/Support-Vector-Machines-Succinctly/tree/master/succinctly/multi_class

[3] Gabriele, D. L. (2020, September 9). *Support Vector Machines vs Neural Networks*. Baeldung.
https://www.baeldung.com/cs/svm-vs-neural-network/

[4] *Support Vector Machine*. (n.d.). Wikipedia.
https://en.wikipedia.org/wiki/Support-vector_machine

[5] Huang, S., et al (2017, December 26). *Applications of SVM learning in cancer genomics*. NCBI.
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5822181

[6] Hai D. M. (2018, March 22). *Support vector machine*. Hai's Blog.
https://dominhhai.github.io/vi/2018/03/ml-svm/

[7] *Stroke prediction dataset* (Version 1). (2020b). [Dataset]. Kaggle.
https://www.kaggle.com/fedesoriano/stroke-prediction-dataset

[8] *Glass Classification* (Version 1). (2017b). [Dataset]. Kaggle.
https://www.kaggle.com/uciml/glass
https://www.kaggle.com/jinxzed/glass-type-eda-pca-feature-engineering
https://www.kaggle.com/eliekawerk/glass-type-classification-with-machine-learning

[9] Chennoju, B. (2021). *Data storytelling AUC focus on strokes*. Kaggle.
https://www.kaggle.com/bhuvanchennoju/data-storytelling-auc-focus-on-strokes

[10] Kaminetsky, I. (2021). *Stroke Prediction Classification - KNN and RFR*. Kaggle.
https://www.kaggle.com/idankaminetsky/stroke-prediction-classification-knn-and-rfr#-B.-Basic-familiarity-with-the-Datast-