Covid-19 Risk Prediction

Trương Thành Thắng
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20521907@gm.uit.edu.vn

Ngô Văn Tấn Lưu
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20521591@gm.uit.edu.vn

Huỳnh Viết Tuấn Kiệt
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20521494@gm.uit.edu.vn

Nguyễn Văn Toàn
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20522028@gm.uit.edu.vn

Ngô Ngọc Sương
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20521852@uit.edu.vn

Trần Văn Lực
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20521587@uit.edu.vn

Nguyễn Đức Anh Phúc
Vietnam National University
University of Information Technology
Ho Chi Minh, Viet Nam
20520276@uit.edu.vn

***Abstract*—Bệnh vi-rút corona là một bệnh truyền nhiễm gây ra bởi một loại vi-rút corona mới được phát hiện. Người lớn tuổi và những người mắc các bệnh nền sẽ có nhiều khả năng phát triển bệnh nặng hơn. Trong những thời điểm khó khăn này, việc có thể dự đoán những nguồn lực mà một cá nhân có thể cần sẽ giúp các cơ quan chức năng lên lịch các nguồn lực cần thiết để cứu sống bệnh nhân. Bởi vậy, chúng tôi đã tiến hành xây dựng nhiều mô hình máy học (Decision Trees, Naive Bayes, Linear Discriminant Analysis, K-Nearest Neighbor, Logistic Regression, Neral Networks, Support Vector Machine) có thể dự đoán bệnh nhân covid-19 nguy kịch từ các triệu chứng, tình trạng và tiền sử bệnh của người đó dựa trên dữ liệu từ Covid-19 Dataset. Kết quả là chúng tôi có nhiều phân tích để các mô hình đạt độ chính xác cao trong bài toán này và mô hình đạt kết quả cao nhất là Decision Tree với f1 score = 60.14%.**

***Index Terms*—Covid-19, Classification, Covid-19 Dataset, Machine Learning, Decision Trees, Naive Bayes, Linear Discriminant Analysis, K-Nearest Neighbor, Logistic Regression, Neural Networks, Support Vector Machine.**

I. Introduction

Bệnh vi-rút corona (Covid-19) là một bệnh truyền nhiễm gây ra bởi một loại vi-rút corona mới được phát hiện. Hầu hết những người bị nhiễm vi-rút Covid-19 đều phát triển bệnh hô hấp từ nhẹ đến trung bình và hồi phục mà không cần điều trị đặc hiệu. Người lớn tuổi và những người mắc các bệnh nền như bệnh tim mạch, tiểu đường, bệnh hô hấp mãn tính và ung thư có nhiều khả năng phát triển bệnh nặng hơn.

Một trong những vấn đề chính mà các nhà cung cấp dịch vụ chăm sóc sức khỏe phải đối mặt trong suốt đại dịch là thiếu nguồn lực y tế và kế hoạch phù hợp để phân bổ các nguồn lực đó một cách hiệu quả. Trong những thời điểm khó khăn này, việc có thể dự đoán những nguồn lực mà một cá nhân có thể cần khi họ xét nghiệm dương tính hoặc thậm chí trước khi họ xét nghiệm dương tính sẽ giúp các cơ quan chức năng lên lịch các nguồn lực cần thiết để cứu sống bệnh nhân.

Vì vậy, trong đồ án này chúng tôi tiến hành xây dựng các mô hình máy học (Decision Trees Naive Bayes Linear Discriminant Analysis K-Nearest Neighbor, Logistic Regression, Neral Networks và Support Vector Machine) để có thể dự đoán liệu bệnh nhân covid-19 có nguy kịch hay không dựa trên các triệu chứng, tình trạng và tiền sử bệnh hiện tại của họ. Kết quả là chúng tôi có nhiều phân tích để các mô hình đạt độ chính xác cao trong bài toán này và mô hình đạt kết quả cao nhất là Decision Tree với f1 score = 60.14%.

Mô hình bài toán:

Fig. 1. Mô hình bài toán

Source code: https://github.com/erwin24092002/PROJECT---Covid19_Risk_Prediction

Phần còn lại của bài báo bao gồm:

- Phần II - Dataset: Giới thiệu bộ dữ liệu được sử dụng trong bài toán - Covid-19 Dataset, phân tích và trình bày cách làm sạch và xử lí các giá trị thiếu hụt trong bộ dữ liệu gốc để tạo ra bộ dữ liệu hoàn chỉnh cho việc giải quyết bài toán.
- Phần III - Methodology: Trình bày kiến thức cơ bản của các phương pháp được chúng tôi chọn để thực nghiệm và các điều chỉnh tham số của chúng. Bao gồm có 7 phương

pháp: Decision Trees, Naive Bayes, Linear Discriminant Analysis, K-Nearest Neighbor, Logistic Regression, Neral Networks, Support Vector Machine.

- Phần IV - Experiment: Nội dung phần này trình bày phương thức đánh giá, độ đo được sử dụng trong thực nghiệm. Đồng thời đưa ra những đánh giá, phân tích dựa trên kết quả đạt được trên mỗi mô hình.
- Phần V - Conclusion: Tổng kết những việc đã làm được trong đồ án. Đúc kết kinh nghiệm chọn và huấn luyện mô hình trong bài toán Covid-19 Risk Prediction trên Covid-19 Dataset.

II. Dataset

A. Covid-19 Dataset

Là bộ dữ liệu mang thông tin về triệu chứng, trạng thái và tiền sử bệnh của các bệnh nhân Covid-19 được cung cấp bởi chính phủ Mexico, bao gồm 1.048.575 điểm dữ liệu với 21 đặc trưng, cụ thể Bảng I.

B. Data Analysis and Preprocessing

Nhìn chung, Covid-19 Dataset là bộ dữ liệu lớn (1.048.575 điểm dữ liệu), mỗi điểm dữ liệu có nhiều đặc trưng (21 đặc trưng), hầu hết đặc trưng là kiểu boolean (16 đặc trưng), có 3 đặc trưng có thông tin bị thiếu hụt là INTUBED, PREGNANT, ICU (các đặc trưng có màu đỏ trong Bảng I) và vẫn chưa có nhãn phân lớp dữ liệu cho bài toán (AT_RISK). Ta lần lượt xử lí các vấn đề trên như sau.

1) Xử lí thông tin bị thiếu hụt của PREGNANT: Thực tế, SEX và PREGNANT có liên quan với nhau, ta tiến hành đếm số lượng nhãn PREGNANT trong SEX.

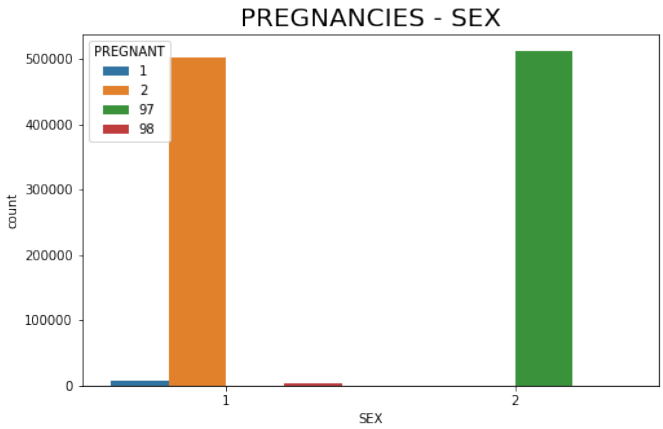


Fig. 2. Số lượng mỗi nhãn PREGNANT trong mỗi lớp của SEX

Từ Hình 2 ta thấy tất cả nhãn 97 đều thuộc lớp Nam, mà Nam không thể mang thai, vậy có nghĩa nhãn 97 trong PREGNANT nên có giá trị là 2 (không có thai), ta sửa toàn bộ giá trị 97 trong PREGNANT thành 2. Tất cả nhãn 98 đều thuộc lớp Nữ, vì vậy ta không biết được nhãn 98 trong PREGNANT mang giá trị 1 (có thai) hay là 2 (không có thai). Và số lượng nhãn 98 cũng không đáng kể, vì vậy ta tiến hành xoá toàn bộ điểm dữ liệu mang nhãn 98 này.

2) Xử lí thông tin bị thiếu hụt của INTUBED và ICU: Đối với 2 đặc trưng này ta không thể dựa vào giá trị đặc trưng khác để điền vào giá trị trống. Đồng thời chúng đều có giá trị đặc biệt với việc quyết định bệnh nhân có trong trường hợp nguy hiểm hay không (sẽ nói việc này ở Phần II.B.2). Vì vậy ta không thể bỏ qua thông tin bị thiếu hụt hoặc loại bỏ cả 3 đặc trưng này khỏi bộ dữ liệu. Trong trường hợp này chúng tôi sử dụng mô hình máy học để dự đoán và điền vào những thông tin còn thiếu trong đặc trưng INTUBED và ICU.

Để đánh giá và tìm ra mô hình thích hợp cho việc dự đoán thông tin còn thiếu,dựa trên những điểm dữ liệu có sẵn chúng tôi thực nghiệm nhiều mô hình với config khác nhau và đánh giá chúng bằng độ đo f1 score (bộ dữ liệu mất cân bằng giữa các lớp trong mỗi INTUBED và ICU, vì vậy không thể dùng accuracy để đánh giá). Cuối cùng mô hình chúng tôi chọn là Decision Tree (max_depth=20) đạt f1 score cao nhất là 0.25 và 0.36 lần lượt trên ICU và INTUBED.

3) Tạo nhãn phân lớp dữ liệu cho bài toán - AT_RISK: Bài toán của ta là bài toán dự đoán bệnh nhân nguy kịch, vì vậy ta cần phải có đặc trưng thể hiện điều này (AT_RISK). Qua quá trình xem xét mức độ nghiêm trọng của bệnh nhân trong thực tế, chúng tôi định nghĩa một bệnh nhân nguy kịch là bệnh nhân thuộc một trong 3 tình trạng là: ICU (được đưa vào Đơn vị Chăm sóc Đặc biệt), INSTUBED (được kết nối với máy thở) và DATE_DIED (Bệnh nhân mất).

$$AT\_RISK = \begin{cases} 1 & \text{Nếu } ICU + INTUBED + DATE\_DIED > 0 \\ 0 & \text{Trường hợp còn lại} \end{cases}$$

Sau khi sử dụng, ta xoá đi các đặc trưng ICU, INSTUBED và DATA_DIED

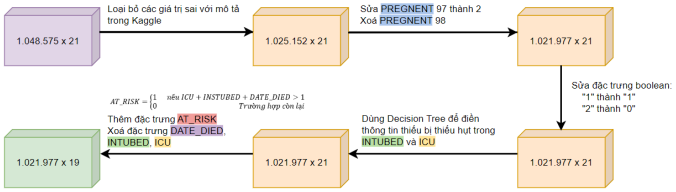


Fig. 3. Quá trình xử lí dữ liệu. Khối màu tím là Covid-19 Dataset ban đầu, khối màu xanh là bộ dữ liệu sau quá trình xử lí.

Sau quá trình xử lí dữ liệu (Hình 3), bộ dữ liệu còn 1.021.977 điểm dữ liệu (đã loại bỏ 26.598 điểm dữ liệu) với 19 đặc trưng, không còn trường hợp thiếu hụt thông tin và đặc trưng cần dự đoán là AT_RISK. Bộ dữ liệu sau xử lí bị mất cân bằng về số lượng điểm dữ liệu giữa 2 lớp trong AT_RISK, tỉ lệ "nguy kịch":"không nguy kịch" là 1:9.

III. Methodology

A. Decision Trees

1) Theory: Decision tree là một công cụ hỗ trợ quyết định sử dụng mô hình quyết định dạng cây và dự đoán các kết quả

TABLE I: Mô tả đặc trưng và Thống kê số lượng điểm dữ liệu thiếu hụt trong Covid-19 Dataset

No.	Feature	Description	Values	Missing Data
1	**USMER**	Cho biết cấp độ đơn vị y tế mà bệnh nhân được điều trị.	Đơn vị y tế cấp [1, 2, 3].	0
2	**MEDICAL_UNIT**	Loại tổ chức của Hệ thống Y tế Quốc gia cung cấp dịch vụ chăm sóc.	Loại [1-13].	0
3	**SEX**	Cho biết giới tính của bệnh nhân.	"1" là giới tính nữ, "2" là giới tính nam.	0
4	**PATIENT_TYPE**	Nhập viện hay không nhập viện.	"1" là nhập viện, "2" là không nhập viện.	0
5	**DATE_DIED**	Ngày bệnh nhân mất.	Ngày mất [dd/mm/yyyy] của nạn nhân hoặc [9999-99-99] nếu không chết.	0
6	**INTUBED**	Cho biết liệu bệnh nhân có được kết nối với máy thở hay không.	"1" là được kết nối, "2" là không được kết nối, giá trị "97" hoặc cao hơn là dữ liệu không được cung cấp.	855869
7	**PNEUMONIA**	Bệnh nhân có viêm phổi hay không.	"1" là bị viêm phổi, "2" là không bị viêm phổi, giá trị "99" là dữ liệu không được cung cấp.	0
8	**AGE**	Cho biết tuổi của bệnh nhân.	Giá trị [0-121]	0
9	**PREGNANT**	Cho biết liệu bệnh nhân có thai hay không.	"1" là có thai, "2" là không có thai, giá trị "97" hoặc cao hơn là dữ liệu không được cung cấp.	523511
10	**DIABETES**	Cho biết liệu bệnh nhân có bị tiểu đường hay không.	"1" là bị tiểu đường, "2" là không bị tiểu đường, giá trị "98" là dữ liệu không được cung cấp.	0
11	**COPD**	Cho biết liệu bệnh nhân có bị mắc bệnh phổi tắc nghẽn mãn tính hay không.	"1" là mắc bệnh, "2" là không mắc bệnh, giá trị "98" là dữ liệu không được cung cấp.	0
12	**ASTHMA**	Cho biết liệu bệnh nhân có bị hen suyễn hay không	"1" là bị hen suyễn, "2" không bị hen suyễn, giá trị "98" là dữ liệu không được cung cấp.	0
13	**INMSUPR**	Cho biết liệu bệnh nhân có bị ức chế miễn dịch hay không.	"1" là bị ức chế miễn dịch, "2" là không bị ức chế miễn dịch, giá trị "98" là dữ liệu không được cung cấp.	0
14	**HIPERTENSION**	Cho biết liệu bệnh nhân có tăng huyết áp hay không.	"1" là tăng huyết áp, "2" là không tăng huyết áp, giá trị "98" là dữ liệu không được cung cấp.	0
15	**OTHER_DISEASE**	Cho biết liệu bệnh nhân có bệnh khác hay không.	"1" là có bệnh khác, "2" là không có bệnh khác, giá trị "98" là dữ liệu không được cung cấp.	0
16	**CARDIOVASCULAR**	Cho biết liệu bệnh nhân có bệnh liên quan đến tim hoặc mạch máu hay không.	"1" là có bệnh, "2" là không có bệnh, giá trị "98" là dữ liệu không được cung cấp.	0
17	**OBESITY**	Cho biết liệu bệnh nhân có bị béo phì hay không.	"1" là bị béo phì, "2" là không bị béo phì, giá trị "98" là dữ liệu không được cung cấp.	0
18	**RENAL_CHRONIC**	Cho biết liệu bệnh nhân có bệnh thận mãn tính hay không.	"1" là có bệnh, "2" là không có bệnh, giá trị "98" là dữ liệu không được cung cấp.	0
19	**TOBACCO**	Cho biết liệu bệnh nhân có sử dụng thuốc lá hay không.	"1" là có sử dụng thuốc lá, "2" là không sử dụng thuốc lá, giá trị "98" là dữ liệu không được cung cấp.	0
20	**CLASIFFICATION_FINAL**	Cho biết kết quả xét nghiệm Covid.	Giá trị [1-3] có nghĩa là bệnh nhân được chẩn đoán mắc covid ở các mức độ khác nhau. 4 trở lên có nghĩa là bệnh nhân không phải là người mang covid hoặc xét nghiệm không kết luận được.	0
21	**ICU**	Cho biết liệu bệnh nhân đã được đưa vào Đơn vị Chăm sóc Đặc biệt hay chưa.	"1" là đã được đưa vào, "2" là chưa được đưa vào, giá trị "97" hoặc cao hơn là dữ liệu không được cung cấp.	856032

có thể xảy ra của chúng. Đây là một phương pháp mà thuật toán quyết định của nó được biểu diễn bằng các câu lệnh điều kiện tạo thành cấu trúc dữ liệu dạng cây.

Quá trình xây dựng Decision Tree tương đối đơn giản. Decision Tree là tập hợp các node, mỗi node được xây dựng bằng một chiến lược phân chia cụ thể. Ở mỗi bước đi, Decision Tree sẽ được xây dựng bằng cách quyết định đặc trưng nào sẽ là yếu tố quyết định tại một node. Quá trình này được diễn ra bằng cách tính lượng thông tin nhiễu chứa đựng ở mỗi đặc trưng tại một cấp. Có nhiều thuật toán để tính lượng thông tin nhiễu đó như Entropy hay Gini Impurity. Đặc trưng nào có chỉ số mang lại thông tin nhiễu thấp nhất sẽ được chọn làm yếu tố phân định tại node đó. Sau có Decision Tree tiến hành chia lại dữ liệu sau sau khi được phân định tại node vừa chọn sau đó tiếp tục lựa chọn các đặc trưng còn lại cho lượng dữ liệu còn lại. Quá trình này được lặp đi lặp lại cho đến khi không còn đặc trưng nào để lựa chọn.

2) Parameter Tuning: Trong số các tham số của Decision Tree được thực hiện bởi thư viện Python là Sklearn. Tham số có ảnh hưởng nhất là tham số criterion. Tham số này được sử dụng để lựa chọn chiến lược phân chia. Có hai chiến lược có thể sử dụng là "Gini Impurity" và "Entropy" với công thức như sau:

- Gini Impurity:

$$Gini = 1 - \sum_j p_j^2$$

- Entropy:

$$Entropy = -\sum_j p_j \cdot log_2 \cdot p_j$$

Với j là xác xuất của lớp j

Gini Impurity đo tần suất của một đặc trưng mà tại đó bất kỳ phần tử nào của tập dữ liệu sẽ bị dán nhãn sai khi nó được gán nhãn ngẫu nhiên. Trong khi Entropy là thước đo thông tin cho biết sự hỗn loạn của các đặc trưng với mục tiêu. Dựa vào công thức, có thể thấy Entropy sẽ tính được lượng tạp chất kỹ hơn do có sử dụng hàm logarith, giúp cho mô hình đạt kết quả cao hơn nhưng đồng thời sẽ tăng thời tính toán. Chi tiết được thể hiện ở bảng kết quả.

B. Naive Bayes

1) Theory: Thuật toán Naive Bayes là một trong những thuật toán học máy phân loại phổ biến giúp phân loại dữ liệu dựa trên tính toán giá trị xác suất có điều kiện. Thực hiện định lý Bayes để tính toán và sử dụng các mức lớp được biểu diễn dưới dạng các giá trị đặc trưng hoặc vectơ của các yếu tố dự đoán để phân loại.

Thuật toán Naive Bayes là một thuật toán nhanh cho các bài toán phân loại. Thuật toán này rất phù hợp cho các trường hợp sử dụng dự đoán theo thời gian thực, dự đoán nhiều lớp, hệ thống đề xuất, phân loại văn bản và phân tích tình cảm. Thuật toán Naive Bayes có thể được xây dựng bằng cách sử dụng phân phối Gaussian, Multinomial và Bernoulli.

- Naive Bayes:

$$P(y|X) = \frac{P(X|y) * P(y)}{P(X)}$$

- Trong đó:
  $P(y)$ được gọi là Prior Probability. $P(y)$ là xác suất để một điểm bất kỳ rơi vào lớp y. Được tính bằng công thức:

$$P(y) = \frac{\text{Số lượng điểm dữ liệu thuộc lớp } y}{\text{Tổng số lượng điểm dữ liệu}}$$

  $P(X)$ được gọi là Marginal Probability. $P(X)$ là xác suất để lấy được điểm X trong tập dữ liệu.

$$P(X) = \frac{\text{Số lượng điểm dữ liệu tương tự với } X}{\text{Tổng số lượng điểm dữ liệu}}$$

  $P(X|y)$ được gọi là Likelihood. $P(X|y)$ được định nghĩa là xác suất để bắt gặp một điểm dữ liệu tương tự X trong các điểm dữ liệu thuộc lớp y. Công thức cụ thể:

$$P(X|y) = \frac{\text{Số lượng điểm trong lớp } y \text{ tương tự với } X}{\text{Số lượng điểm trong lớp } y}$$

  $P(y|X)$ được gọi là Posterior Probability.

Ưu điểm

- Thuật toán này hoạt động nhanh chóng và có thể tiết kiệm rất nhiều thời gian.
- Naive Bayes phù hợp để giải các bài toán dự đoán nhiều lớp.

- Nếu giả định về tính độc lập của các tính năng là đúng, thì nó có thể hoạt động tốt hơn các mô hình khác và yêu cầu ít dữ liệu đào tạo hơn nhiều.
- Naive Bayes phù hợp hơn với các biến đầu vào phân loại hơn là các biến số.

Nhược điểm

- Naive Bayes cho rằng tất cả các yếu tố dự đoán (hoặc tính năng) đều độc lập, hiếm khi xảy ra trong đời thực. Điều này giới hạn khả năng áp dụng của thuật toán này trong các trường hợp sử dụng trong thế giới thực.
- Thuật toán này phải đối mặt với 'vấn đề tần số bằng không' khi nó gán xác suất bằng không cho một biến phân loại có danh mục trong tập dữ liệu thử nghiệm không có sẵn trong tập dữ liệu huấn luyện. Sẽ là tốt nhất nếu bạn sử dụng một kỹ thuật làm mịn để khắc phục vấn đề này.
- Ước tính của nó có thể sai trong một số trường hợp, vì vậy bạn không nên quá coi trọng kết quả xác suất của nó.

2) Parameter Tuning: Scikit-Learning cung cấp ba mô hình triển khai Naive Bayes: Gaussian, Bernoulli, Multinomial. Sự khác biệt duy nhất là phân phối xác suất được thông qua.

Multinomial Naive Bayes: Trình phân loại được sử dụng rộng rãi để phân loại tài liệu giúp duy trì số lượng từ thường xuyên có trong tài liệu.

Bernoulli Naive Bayes: Được sử dụng cho dữ liệu rời rạc, trong đó các tính năng chỉ ở dạng nhị phân.

Gaussian Naive Bayes: Được sử dụng khi chúng tôi đang xử lý dữ liệu liên tục và sử dụng phân phối Gaussian.

Nhận thấy phần kết quả có 2 lớp dữ liệu là nguy kịch và không nguy kịch. Phù hợp để sử dụng mô hình Bernoulli.

- Nếu X là biến ngẫu nhiên có phân phối Bernoulli, thì nó chỉ có thể nhận hai giá trị (để đơn giản, hãy gọi chúng là 0 và 1) và xác suất của chúng là:
  $P(X) = p$ if $X = 1$
  $P(X) = q$ if $X = 0$
  Where $q = 1 - p$ and $0 < p < 1$
- Quy tắc quyết định cho Bernoulli dựa trên:

$$P(x_i|y) = P(x_i = 1|y)x_i + (1 - P(x_i = 1|y))(1 - x_i)$$

C. Linear Discriminant Analysis

1) Theory: Các kỹ thuật giảm kích thước dữ liệu rất quan trọng trong các ứng dụng Máy học, Khai thác dữ liệu và Truy xuất thông tin mà công việc chính là loại bỏ các đặc trưng dư thừa và phụ thuộc bằng cách thay đổi tập dữ liệu trên không gian có số chiều thấp hơn. Trong đó, kỹ thuật phổ biến nhất là PCA (Principle Component Analysis) - phương pháp chỉ sử dụng các vector mô tả dữ liệu (unsupervised learning) để giảm chiều dữ liệu sao cho lượng thông tin về dữ liệu được giữ lại là nhiều nhất. Tuy nhiên, trong nhiều bài toán, ta không cần giữ lại lượng thông tin lớn nhất mà chỉ cần giữ lại thông tin cần thiết cho riêng bài toán đó. Vì vậy LDA (Linear Discriminant Ananlysis) được ra đời nhằm tìm ra không gian biểu diễn dữ liệu hiệu quả nhất.

LDA là một phương pháp giảm chiều dữ liệu có khai thác mối liên quan giữa dữ liệu và nhãn của dữ liệu (supervised learning). LDA có thể được coi là một phương pháp giảm chiều dữ liệu, cũng có thể được coi là một phương pháp phân lớp, và cũng có thể được áp dụng đồng thời cho cả hai, tức giảm chiều dữ liệu sao cho việc phân lớp hiệu quả nhất. Số chiều của dữ liệu mới là nhỏ hơn hoặc bằng $C - 1$ trong đó C là số lượng lớp dữ liệu.

Ý tưởng cơ bản của LDA là tìm một không gian mới với số chiều nhỏ hơn không gian ban đầu sao cho hình chiếu của các điểm trong cùng lớp lên không gian mới này là gần nhau trong khi hình chiếu của các điểm của các lớp khác nhau là khác nhau. Hàm mục tiêu (công thức áp dụng cho dữ liệu chỉ có 2 lớp):

$$J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

Trong đó:

- m_1, m_2 là vector kỳ vọng của mỗi lớp trong không gian đang xét, khoảng cách giữa 2 kỳ vọng $(m_1 - m_2)^2$, khoảng cách càng lớn tức dữ liệu giữa 2 lớp càng khác nhau nhiều.
- s_1, s_2 là độ lệch chuẩn của mỗi lớp trong không gian đang xét, giá trị càng nhỏ chứng tỏ dữ liệu trong mỗi lớp ít phân tán, giống nhau.

Thuật toán LDA sẽ đi tìm không gian dữ liệu mới sao cho hàm mục tiêu $J(w)$ đạt giá trị lớn nhất.

LDA hoạt động rất tốt nếu các lớp là linearly separable (có tính chất phần tán tuyến tính). Chất lượng mô hình giảm đi rõ rệt nếu các classes là không linearly separable. Điều này dễ hiểu vì khi đó, chiếu dữ liệu lên phương nào thì cũng bị chồng lấn, và việc tách biệt không thể thực hiện được như ở không gian ban đầu.

2) Parameter Tuning: Trong bài toán Phát hiện bệnh nhân nguy kịch, có 2 lớp dữ liệu ("nguy kịch" và "không nguy kịch"), vì vậy LDA chỉ có thể giảm chiều dữ liệu về 1. Sau khi giảm chiều dữ liệu, chúng tôi huấn luyện các mô hình (Decision Tree, Naive Bayes, K-nearest Neighbor, Logistic Regression, Support Vector Machine với config đạt kết quả tốt nhất trong đồ án này) thực hiện phân lớp để phân tích sự tác động của LDA lên hiệu suất các mô hình.

D. K-Nearest Neighbor

1) Theory:

a) Khái niệm: KNN (K-Nearest Neighbors) là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai phá dữ liệu và học máy. Ý tưởng của thuật toán này là nó không học một điều gì từ tập dữ liệu học (nên KNN được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán nhãn của dữ liệu mới.

Lớp (nhãn) của một đối tượng dữ liệu mới có thể dự đoán từ các lớp (nhãn) của k hàng xóm gần nó nhất.

Thuật toán KNN cho rằng những dữ liệu tương tự nhau sẽ tồn tại gần nhau trong một không gian, từ đó công việc của chúng ta là sẽ tìm k điểm gần với dữ liệu cần kiểm tra nhất.

Việc tìm khoảng cách giữa 2 điểm củng có nhiều công thức có thể sử dụng, tùy trường hợp mà chúng ta lựa chọn cho phù hợp.

Đây là 3 cách cơ bản để tính khoảng cách 2 điểm dữ liệu x, y có k thuộc tính

- Euclidean
$$\sqrt{\sum_{1}^{k}(x_i - y_i)^2}$$
- Manhattan
$$\sum_{1}^{k}|x_i - y_i|$$
- Minkowski
$$\left(\sum_{1}^{k}\left(|x_i - y_i|\right)^q\right)^{\frac{1}{q}}$$

Các bước trong KNN

1) Ta có D là tập các điểm dữ liệu đã được gắn nhãn và A là dữ liệu chưa được phân loại.
2) Đo khoảng cách (Euclidian, Manhattan, Minkowski, Minkowski hoặc Trọng số) từ dữ liệu mới A đến tất cả các dữ liệu khác đã được phân loại trong D.
3) Chọn K (K là tham số mà bạn định nghĩa) khoảng cách nhỏ nhất.
4) Kiểm tra danh sách các lớp có khoảng cách ngắn nhất và đếm số lượng của mỗi lớp xuất hiện.
5) Lấy đúng lớp (lớp xuất hiện nhiều lần nhất).
6) Lớp của dữ liệu mới là lớp mà bạn đã nhận được ở bước 5.

Ưu điểm

- Thuật toán đơn giản, dễ dàng triển khai.
- Độ phức tạp tính toán của quá trình training là bằng 0.
- Xử lý tốt với tập dữ liệu nhiễu

Nhược điểm

- Với K nhỏ dễ gặp nhiễu dẫn tới kết quả đưa ra không chính xác.
- Cần nhiều thời gian để thực hiện do phải tính toán khoảng cách với tất cả các đối tượng trong tập dữ liệu.
- Cần chuyển đổi kiểu dữ liệu thành các yếu tố định tính.
- Hoạt động kém hơn ở tập dữ liệu có mật độ thấp (Curse of dimensionality).

2) Parameter Tuning: KNN đắt về mặt chi phí tính toán vì nó tải toàn bộ tập dữ liệu vào bộ nhớ để phân loại. Khi số lượng tính năng của tập dữ liệu rất cao, nó có thể bị ảnh hưởng bởi tính đa chiều và có thể hoạt động kém.

a) **n_neighbors** : Số neighbors được sử dụng theo mặc định cho neighbors queries. Gía trị tuning của tham số này sẽ được rút ngắn phạm vi tuning thông qua việc sử dung GridSearchCV từ đó tìm được **n_neighbors** tối ưu nhất.

b) **algorithm** :Thuật toán dùng để tính neighbors gần nhất. Thông thường trong KNN sử dụng algorithm là Brute Force có thể là phương pháp chính xác nhất do xem xét tất cả các điểm dữ liệu. Do đó, không có điểm dữ liệu nào được gán cho một cụm sai. Đối với các tập dữ liệu nhỏ, Brute Force là hợp lý, tuy nhiên với tập dữ liệu Covid-19 Risk Prediction có 1 000 000 điểm dữ liệu sử dụng KD Tree hoặc Ball Tree là những lựa chọn tuning tốt hơn do tốc độ và hiệu quả của chúng trên tập dữ liệu lớn như vậy.

c) **metric** : Metric dùng để tính toán khoảng cách 2 điểm dữ liệu gồm 3 cách cơ bản để tính khoảng cách đã được trình bày ở trên là Minkowski,Manhattan,Euclidean.

d) Values tuning:
   - n_neighbors : np.arange(7,14)
   - algorithm : ['ball_tree','kd_tree']
   - metric : ['euclidean','manhattan','minkowski']

   Tuning tool: **GridSearchCV**

   Bộ tham số tối ưu cho thuật toán khi áp dụng GridSearchCV :
   - n_neighbors : 13
   - algorithm : kd_tree
   - metric : manhattan

E. Logistic Regression

1) Theory: Logistic Regression là thuật toán học có giám sát (supervised learning) đơn giản nhưng lại rất hiệu quả trong bài toán phân loại (Classification).

Mục đích của Logistic Regression là ước tính xác suất của các sự kiện, bao gồm xác định mối quan hệ giữa các tính năng từ đó đự đoán xác suất của các kết quả, nên đối với Logistic Regression ta sẽ có:

- Input: Vector dữ liệu mẫu $[x_1 x_2 ... x_i]^T$, trong đó $x_i \in R$ là thành phần thứ i của mẫu dữ liệu đầu vào.
- Output: giá trị cần dự đoán y, $y \in \{0, 1\}$.

Trong hồi qui tuyến tính chúng ta dựa vào một hàm hồi qui giả thuyết $h_w(x) = w^T x$ để dự báo biến mục tiêu liên tục y. Vì giá trị của y có thể vượt ngoài khoảng $[0, 1]$ nên trong Logistic Regression cần một hàm số có tác dụng chiếu giá trị dự báo lên không gian xác suất nằm trong khoảng $[0, 1]$ và đồng thời tạo ra tính phi tuyến cho phương trình hồi qui nhằm giúp nó có đường biên phân chia giữa hai nhóm tốt hơn. Đó chính là hàm Sigmoid có miền giá trị tự 0 đến 1

$$f(x) = \frac{1}{1 + e^{-x}}$$

Logistic Regression sử dụng hàm mất mát có tên là Cross-Entropy:

$$L(w) = -\sum_{1}^{N}(y_i * log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i))$$

Dưới góc nhìn của graphic model thì mô hình Logistic regression có dạng như sau:

Đồ thị trên sẽ bao gồm hai bước:

- Bước 1: Kết hợp tuyến tính. Mỗi một node (hình tròn) đại diễn cho 1 biến đầu vào. Các cạnh là hình mũi tên có hướng thể hiện hướng tính toán của đồ thị. Đầu vào sẽ là node ở gốc mũi tên và đầu ra là node ở ngọn mũi tên. Giá trị này sẽ được điều tiết bằng cách nhân với hệ

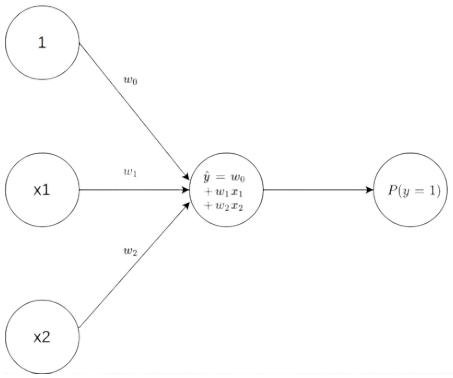


Fig. 4. Biểu diễn dạng đồ thị của mô hình Hồi quy Luận lý.

số w_i. Cuối cùng ta sẽ kết hợp tuyến tính các nodes đầu vào để tính ra đầu ra $haty$.

- Bước 2: Biểu diễn hàm Sigmoid. Giá trị $haty$ lại tiếp tục được đưa qua hàm σ để tính ra xác suất $P(y = 1)$ ở output.

Ưu điểm:

- Dễ đào tạo và triển khai hơn so với các phương pháp khác.
- Hoạt động tốt đối với các tập dữ liệu có thể phân tách tuyến tính.
- Nó có thể giải thích các hệ số của mô hình như là các chỉ số về tầm quan trọng của tính năng.

Nhược điểm:

- Không dự đoán được kết quả liên tục.
- Dự đoán có thể không chính xác nếu kích thước mẫu quá nhỏ.

2) Parameter Tuning: Thực hiện tuning trên 2 tham số penalty và solver để biết với giá trị nào của tham số thì mô hình đạt kết quả cao nhất

- penalty: Được sử dụng để xác định các tiêu chuẩn được sử dụng trong tối ưu loss
- solver: Đây là thuật toán được sử dụng trong bài toán tối ưu
- Values tuning:
  - penalty : [l1,l2]
  - solver : ['liblinear','saga','lbfgs','newton-cg']

F. Neural Networks

1) Theory: Neural network là một chuỗi các thuật toán được kết hợp dùng để giải quyết xử lí mối quan hệ của một tập dữ liệu như cách mà não bộ của chúng ta xử lí. Theo định nghĩa này, chúng ta có thể thấy được rằng neural network là một hệ thống các nơ ron thần kinh xi-náp được sắp xếp một cách "chằn chịt" để xử lí những thông tin phi tuyến, logic và cực kì phức tạp.

Neural network sẽ có nhiều nút (nodes) connect với nhau và với mỗi nút sẽ có nhiều dữ liệu đầu vào. Các dữ liệu đầu vào (input) được biến đổi bằng cách tính tổng các dữ liệu với ma trận trọng số (weight) tương ứng, sau đó, các hàm phi tuyến (activation function) sẽ phi tuyến hóa các chuỗi công thức để tính toán trạng thái trung gian. Với mỗi nút thì dữ liệu truyền vào và bộ weight là khác nhau nên ta có thể hiểu rằng mỗi nút sẽ đại diện cho một "đặc trưng" nào đó có trong bộ dữ liệu, khi được huấn luyện đủ nhiều thì mỗi một nút sẽ có vai trò như là một "bộ nhận diện" đặc trưng tương ứng. Một Neural network sẽ có nhiều nút ở mỗi lớp (layer), các nút trên cùng một lớp sẽ không connect với nhau nhưng các nút khác lớp sẽ connect pair-wise với nhau và các đầu ra (output) của các lớp này sẽ là input của các lớp phía sau. Neural network sẽ

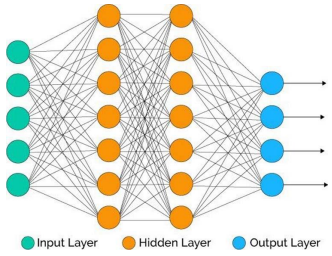


Fig. 5. Mô hình của một Neural network với kích thước của input layer, hidden layer 1, hidden layer 2 và output layer lần lượt là 5, 7, 7, 4

trở thành một mô hình hồi quy tuyến tính (linear regression) mà việc mô hình hóa một bài toán phức như nhận diện khuôn mặt, phân loại văn bản, etc. dưới dạng tuyến tính sẽ là cho mô hình khó có thể đạt được hiệu quả cao. Với mạng Neural network, chúng ta có thể mô hình hóa như sau:

$$Y = Activation((weight \times input) + bias)$$

Các hàm Activation cơ bản hiện nay có như:

- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- Tanh: $tanh(x) = \frac{2}{1+e^{-2x}} - 1$
- RELU: $f(x) = max(0, x)$
- Leaky RELU: $f(x) = ax, 0 > x; = x, 0 \leqslant x$
- Softmax: $a(x) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}; j = 1, .., k$

Việc xác định hàm Activation phụ thuộc vào bài toán mà chúng ta cài đặt mạng Neural để giải quyết. Khi chúng ta muốn phân loại $(0, 1)$, chúng ta thường sử dụng các hàm sigmoid. Còn để phân loại nhiều lớp thì hàm softmax là lựa chọn cài đặt hợp lí nhất. RELU là hàm phổ biến nhất và chỉ dùng trong những hidden layers để phi tuyến hóa chuỗi các phương trình lan truyền.

Quy trình để huấn luyện một Neural network trong một iteration:

1) Lấy một tập data có kích thước là 1 batch_size dùng để huấn luyện cho 1 iteration
2) Input (feature, ground_truth) sẽ được reshape thành véc-tơ đặc trưng (feature vector) có kích thước cố định
3) Lan truyền thuận (Forward pass) feature vector đó qua các lớp (qua 1 layer là sử dụng 1 phép matmul) để ra kết quả α ở lớp cuối cùng
4) Tính giá trị hàm lỗi (Loss) $\mathcal{L}(\alpha, ground\_truth)$
5) Tính giá trị đạo hàm của hàm lỗi $\mathcal{L}$ vừa tính được theo các trọng số có trong mạng Neural

6) Cập nhật trọng số cho mạng Neural bằng công thức Gradient Descend và empty gradient (Pytorch)

Neural network có thể hoạt động hiệu quả hơn bộ não con người bằng cách trích xuất (extract) những đặc trưng đặc biệt của bộ dữ liệu mà khó để có thể con người nhìn ra được. Ngoài ra, Neural network còn được ứng dụng rất nhiều trong nhiều lĩnh vực khác nhau như y tế, chứng khoán, bất động sản hay kể cả ngân hàng vì tính dễ thiết kế và độ hiệu quả cao. Tuy nhiên, vì phải sử dụng những phép nhân ma trận vô cùng phức tạp mà Neural network cần GPU nhân CUDA để tăng tốc tính toán nên cũng là một bài toán khó đối với công việc bảo trì (maintenance). Ngoài ra, Neural network là một Black Box nên việc giải thích tại sao có thể đưa ra output cụ thể như thế nào là một việc vô cùng thách thức, đặc biệt hơn vì là một Black Box nên Neural network có thể dễ dàng bị Back-door attack bởi những hacker bằng việc cài nhiễu huấn luyện (noise samples) hay thay đổi cấu trúc nơ ron(neural structure), etc. Chính vì thế mà hiện nay người ta đang nghiên cứu về Explainable AI để giải quyết vấn đề này bằng cách cấu trúc mô hình sẽ giải thích cho chúng ta rằng vì sao lại đưa ra output như vậy.

2) Parameter Tuning: Configuration gốc: Vì có khá nhiều tổ hợp tuning, nên chúng ta chỉ tune từng hyper-parameter theo configuration gốc để tránh trường hợp số lượng kết quả tăng theo hàm mũ.

a) Optimizer: Đây là thuật toán tối ưu (optimizer) dùng để đi tìm cực trị địa phương của hàm loss trong quá trình huấn luyện Neural network. Việc sử dụng optimizer nào phụ thuộc rất nhiều vào kiến trúc của mạng và bài toán mà chúng ta đang giải quyết. Đối với những bài toán đơn giản, chỉ cần dùng GD hay SGD là mô hình có thể hội tụ nhưng cũng có những bài toán phức tạp trên những bộ dữ liệu phức tạp, việc sử dụng optimizer nào cho hợp lí lại là một nghệ thuật trong việc huấn luyện mô hình Neural network. Không phải cứ sử dụng những optimizer phức tạp trên những bài toán đơn giản sẽ đạt được kết quả tốt và ngược lại, việc này cần đuược xác minh qua quá trình thực nghiệm.

b) Learning rate: Đây là hệ số học để cập nhật ma trận trọng số theo công thức của gradient descend với mỗi batch được truyền vào. Độ lớn của learning rate sẽ ảnh hưởng trực tiếp tới tốc độ hội tụ của hàm loss tới điểm cực trị toàn cục. Nếu tăng learning rate quá lớn, mô hình khó có thể đạt đến được cực trị đại phương của của hàm Loss do bước nhảy giữa các weight là quá lớn. Nếu giảm learning rate đi quá nhỏ thì việc tìm bộ weight sao cho hàm Loss đạt đến cực trị địa phương là vô cùng lâu vì các bước nhảy giữa các weight là quá nhỏ (biên độ biến đổi của bộ weight quá nhỏ-chưa đủ lớn).

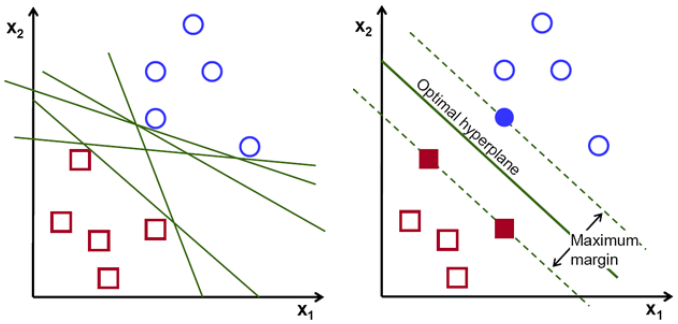
c) Hàm loss: Đây là hàm lỗi dùng để huấn luyện Neural network. Cũng tương tự như optimizer, việc sử dụng hàm Loss cho bài toán hay một cấu trúc phụ thuộc vào quá trình thực nghiệm. Tuy nhiên, có những hàm Loss được sinh ra để giải quyết những bài toán có những đặc trưng riêng biệt như Cross-Entropy Loss sẽ cho ra kết quả rất tốt nếu được dùng cho các bài toán phân loại, còn Smooth L1 thì được thiết kế cho những bài toán hồi quy.

d) Hàm Activation: Hàm kích hoạt để phi tuyến hóa kết quả của từng layer. Tuỳ bài toán khi chúng ta tìm hiểu về các cấu trúc mạng cụ thể, các activation khác nhau sẽ được sử dụng, tuỳ vào độ sâu của mạng, output mong muốn, thậm chí là dữ liệu của bài toán. Chúng ta không thể nói hàm nào tốt hơn khi chưa xét đên điều kiện cụ thể cũng giống như hàm Loss và Optimizer.

G. Support Vector Machine

1) Theory:

a) Khái niệm: Support Vector Machine (SVM) là thuật toán học có giám sát (supervised learning algorithm) hầu hết được sử dụng cho bài toán Classification, nhưng nó cũng có thể được sử dụng cho bài toán Regression. Ý tưởng chính là dựa trên dữ liệu đã gắn nhãn (training data), thuật toán cố gắng tìm optimal hyperplane trong không gian N chiều (N - số lượng đặc trưng) có thể được sử dụng để phân loại các điểm dữ liệu mới.



(a) (b)

Fig. 6. Hyperlane và Support Vectors trong Support Vector Machine. (a) Các Hyperlanes có thể có. (b) Hyperlane tối ưu

b) Ưu nhược điểm khi sử dụng Support Vector Machine: Ưu điểm:

- Hiệu quả trên các bộ dữ liệu có nhiều đặc trưng, như dữ liệu tài chính hoặc y tế.
- Hiệu quả trong trường hợp số lượng đặc trưng lớn hơn số lượng điểm dữ liệu.
- Sử dụng một tập hợp con các điểm huấn luyện trong hàm quyết định được gọi là **Support vector** giúp bộ nhớ hiệu quả.
- Hyperplane chỉ bị ảnh hưởng bởi các **Support vectors**, do đó các giá trị outliers ít bị ảnh hưởng hơn

Tuy nhiên, SVMs có một số nhược điểm:

- SVM đòi hỏi thời gian huấn luyện lâu; dẫn đến nó không thực tế đối với các tập dữ liệu lớn.
- Khi tập dữ liệu chứa nhiều dữ liệu nhiễu, SVM hoạt động với hiệu suất kém.
- Việc lựa chọn **kernel function** có thể rất khó khăn.
- Vì Support vector classifiers hoạt động bằng cách đặt các điểm dữ liệu, bên trên và bên dưới **classifying hyperlane**, nên không có giá trị xác suất rõ ràng cho việc phân loại.

2) Parameters Tuning:

a) SVM Kernels: Kernels trong SVMs giúp xác định hình dạng hyperplane và decision boundary. Việc lựa chọn kernel phù hợp sẽ giảm số lượng số lượng tính toán phức tạp và giải quyết vấn đề overfitting trên bộ dữ liệu lớn. Do đó, kernel là một trong những tham số quan trong SVMs, và nghiên cứu này sẽ tuning các loại kernel khác nhau để tìm kernel phù hợp nhất cho model.

Bảng II chỉ ra các kernel được sử dụng cho thuật toán Support Vector Classifiers để giải quyết bài toán này.

TABLE II: Các kernel thông dụng trong Support Vector Classifier

Tên	Công thức	Kernel
Linear	$x^T z$	$linear$
Polynomial	$(r + \gamma x^T z)^d$	$poly$
Sigmoid	$\tanh(\gamma x^T z + r)$	$sigmoid$
Radial Basis Function	$\exp(-\gamma \|\|x - z\|\|_2^2)$	rbf

b) Regularization C parameter: Tham số C thêm một hình phạt (penalty) cho mỗi điểm dữ liệu bị phân loại sai. Nghiên cứu này thử nghiệm tìm tham số C tốt nhất cho mô hình SVMs trên bộ dữ liệu có kích thước lớn và đa chiều, hướng tới việc tìm bộ trọng số đơn giản có thể phân loại tập train chính xác và đảm bảo không xảy ra overfitting.

c) Values tuning:

- Kernel: [linear, poly, sigmoid, rbf]
- C: [0.1, 1.0, 10.0, 100.0]

Trong bộ siêu tham số của Support Vector Machine, việc lựa chọn kernel phù hợp là một điều quan trọng và cũng là vấn đề khó khăn nhất. Linear kernel không phù hợp trên bộ dữ liệu không thể phân tách tuyến tính (linear separable) dẫn đến thuật toán đạt hiệu suất kém, do đó linear kernel được thử nghiệm nhưng không thể ghi nhận được kết quả trong nghiên cứu này.

Như đã đề cập ở phần lý thuyết, SVMs đòi hỏi thời gian huấn luyện lớn so với các phương pháp Machine Learning khác trên bộ dữ liệu lớn. Bộ dữ liệu Covid-19 được sử dụng trong nghiên cứu này bao gồm hơn 1 triệu điểm dữ liệu, đủ lớn để gây thách thức cho phương pháp Support Vector Machine. Việc thử nghiệm các tham số cho SVMs trên 700.000 (tập train) điểm dữ liệu cần nhiều thời gian và tài nguyên máy tính, điều này nằm ngoài khả năng và phạm vi của nghiên cứu. Do đó, riêng việc thực nghiệm phương pháp SVMs trên Covid-19 dataset trong nghiên cứu này sẽ sử dụng 200.000 điểm dữ liệu để đảm bảo về mặt thời gian và vẫn được đánh giá trên cùng tập test so với các phương pháp khác trong nghiên cứu này.

3) GridSearchCV: GridSearchCV cũng được sử dụng để tìm bộ tham số tốt nhất cho phương pháp Support Vector Machine như một cách kiểm chứng lại kết quả đã thực nghiệm được ở bảng IV.

Tham số thử nghiệm:

- Kernel: [poly, sigmoid, rbf]
- C: [0.1, 1.0, 10.0, 100.0]

Dựa trên bộ tham số cung cấp, GridSearchCV thực hiện xác thực chéo 12 configs trên 3 bộ dữ liệu tương ứng với 3 lần chia tỉ lệ $random\_state$ khác nhau. Kết quả thực nghiệm được lấy trung bình qua 3 lần đo và thống kê ở bảng III.

TABLE III: Kết quả thực nghiệm GridSearchCV

Parameters	0.1	1	10	100
Poly	88.23	90.8	90.9	91.0
RBF	89.4	90.8	90.9	91.1
Sigmoid	80.7	80.2	79.8	79.8

IV. Experiment

A. Protocol

Bộ dữ liệu chia tỉ lệ train:test là 7:3, được chia 3 lần để thực nghiệm, random_state lần lượt là: 7, 14, 21.

Mỗi mô hình được thực nghiệm trên 3 bộ dữ liệu đã chia trên, sau đó tính trung bình cộng hiệu suất trên mỗi mô hình.

B. Evaluation Metrics

1) Accuracy: Đây là phương pháp đánh giá mô hình phân lớp đơn giản nhất. Phương pháp này được tính bằng cách lấy tỉ lệ giữa số lượng dự đoán đúng và tổng số lượng cần dự đoán.

$$Accuracy = \frac{\text{Số nhãn dự đoán đúng}}{\text{Tổng số nhãn cần dự đoán}}$$

2) Precision: Thể hiện mức độ uy tín của mô hình khi xác định một nhãn positive là chính xác (True Positive).

$$Precison = \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

3) Recall: Thể hiện khả năng bao quát của mô hình (tỉ lệ nắm bắt được đối tượng positive)

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

4) F1-Score: Các mô hình không thể so sánh với nhau bằng 2 con số là Precision, Recall được, vì vậy F1-Score ra đời, là trung bình điều hoà của Precision và Recall.

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

5) Time: Thời gian huấn luyện của mô hình.

Trong bài toán này, bộ dữ liệu Covid-19 là bộ dữ liệu mất cân bằng, vì vậy không thể đánh giá mô hình qua độ đo accuracy được, chúng tôi sẽ tập trung vào đánh giá độ chính xác quá F1-Score và thời gian huấn luyện của mô hình.

C. Result and Analysis

1) Decision Trees: Phương pháp Decision Tree với bản chất là thực hiện phân chia các đặc trưng thành các node để phân loại, do đó có thể coi thời gian thực hiện huấn luyện là rất ngắn do chỉ cần thực hiện tính toán dựa trên loại đặc trưng.

Về hiệu quả trên bộ dữ liệu Covid-19, có thể nói Decision Tree đạt kết quả cao nhất so với các phương pháp khác cùng thực nghiệm do tính chất đơn giản của bộ dữ liệu đa số chứa các đặt trưng có kiểu dữ liệu boolean. Với kiểu dữ liệu này, khi xem xét một đặc trưng để chọn làm yếu tố phân chia thì Decision Tree chỉ cần tính toán mức độ phân chia của node dựa trên số lượng phân chia.

2) Naive Bayes: Thuật toán Naive Bayes về mặt thời gian có thể thấy nó nhanh hơn các thuật toán khác rất nhiều lần. Thời gian để đào tạo một mô hình của b dữ liệu trên chưa đến 1s. Nhanh nhất là MultinomialNB với 0.13s, nhỏ gấp nhiều lần so với thời gian đào tạo mô hình của thuật toán khác.

Với thuật toán Naive bayes, qua các kết quả thực nghiệm được có thể thấy mô hình MultinomialNB và BernoulliNB cho ra kết quả cao hơn GaussianNB cho thấy 2 mô hình trên hoạt động khá tốt trên dữ liệu rời rạc. Trong đó BernoulliNB là cao nhất ở cả 2 độ đo là f1 score và accuracy.

Kết quả trên cũng khá dể hiểu khi BernoulliNB Naive Bayes được sử dụng cho dữ liệu rời rạc và nó hoạt động trên phân phối Bernoulli. Đặc trưng chính của Bernoulli Naive Bayes là nó chỉ chấp nhận các đặc trưng dưới dạng giá trị nhị phân như đúng hoặc sai, có hoặc không, 0 hoặc 1...Vì vậy, khi nhìn vào bộ dữ liệu phần lớn đều có 2 lớp nên rất phù hợp để sử dụng BernoulliNB.

3) K-Nearest Neighbor: Với thuật toán KNN thông qua các kết quả thực nghiệm có thể thấy khi tuning n_neighbors = 5,9,13,30 thì accuracy của các model không chênh lệch quá nhiều điều đó cho thấy trong mô hình này ảnh hưởng của n_neighbors đến hiệu quả của mô hình KNN không quá nhiều nhưng nếu n_neighbors nhỏ như 5 hoặc quá lớn như 30 thì hiệu quả của mô hình sẽ không cao.

Đối với dữ liệu có nhiều chiều như bài toán này, thuật toán Ball Tree có thể là giải pháp tốt nhất.Hiệu suất của nó phụ thuộc vào số lượng dữ liệu đào tạo, kích thước và cấu trúc của dữ liệu. Nhưng tập dữ liệu Covid-19 Risk Prediction có nhiều điểm dữ liệu bị nhiễu do đó dẫn đến hiệu suất kém hơn so với KD Tree do Ball Tree không có cấu trúc. Qua đó có thể thấy bộ tham số tối ưu cho thuật toán KNN trong bài toán này là n_neighbors =13 ,algorithm : kd_tree và metric là manhattan.

Khi sử dụng thuật toán KNN cho bài toán này hiệu quả đạt được sẽ cao hơn nhiều thuật toán bởi vì với tập dữ liệu này KNN sẽ không bị ảnh hưởng bởi Curse of dimensionality (hoạt động kém ở tập dữ liệu nhiều chiều và ít dữ liệu hay tập dữ liệu có mật độ thấp) nhưng vì tập dữ liệu này đủ lớn để giữ nguyên mật độ điểm dữ liệu giúp thuật toán KNN đạt hiệu quả cao hơn cũng như tăng khả năng dự đoán chính xác hơn (Mọi thuật toán học máy đều cần một tập dữ liệu dày đặc để dự đoán chính xác trên toàn bộ không gian dữ liệu).

4) Logistic Regression: Qua kết quả thực nghiệm ta thấy khi tuning với penalty [l1, l2] và solver ['liblinear','saga','lbfgs','newton-cg'] thì accuracy hầu như không có sự chênh lệch. Điều này cho thấy penalty và solver không ảnh hưởng nhiều đến hiệu quả của mô hình Logistic Regression.

Tuy nhiên thời gian huấn luyện của chúng lại chênh lệch rất lớn. Trong số các solver đã thực nghiệm, solver 'liblinear' có thời gian huấn luyện nhanh nhất. Solver 'lbfgs' 'newton-cg' và 'saga' có thời gian huấn luyện khá lâu, đặc biệt là 'saga' thời gian huấn luyện là 44.23 với penalty 'l1' và 36.56 với penalty 'l2'.

Vậy khi sử dụng phương pháp Logistic Regression cho bài toán Classification, ta nên sử dụng solver 'liblinear' với penalty 'l2' sẽ cho kết quả tốt cả về độ chính xác và thời gian huấn luyện.

5) Neural Network: Với Neural Network, ta thấy được mô hình với cấu hình là SGD đạt kết quả vô cùng thấp với F1 Score là 22.84 thấp hơn rất nhiều so với các cấu hình khác. Điều này cũng vô cùng dễ hiểu bởi vì ràng buộc của thuật toán SGD là vô cùng ít, thêm vào đó là không có momentum để tăng tốc quá trình hội tụ nên dẫn đến cấu hình thực nghiệm đạt kết quả thấp hơn rất nhiều so với các cấu hình khác. Trái lại với SGD là AdamW, một phiên bản mở rộng hơn của Adam với việc tách weight decay ra và sử dụng 2 moment đồng thời nên thuật toán này có tính flexible, có thể dễ dàng thay đổi theo vector gradient. Do đó kết quả đạt được khi sử dụng AdamW là khá cao với F1 Score là 51.52.

Nhờ vào thực nghiệm dày đặc, ta có thể thấy được rằng việc sử dụng các hàm activation trong Neural Network là vô cùng quan trọng. Trong Table IV, kết quả của việc sử dụng / không sử dụng hàm activation chênh lệch vô cùng đáng kể khi nếu ta sử dụng RELU và Tanh thì đạt được kết quả F1 Score lần lượt là $50.99, 48.53$ nhưng nếu không dùng hàm activation để phi tuyến hóa các phương trình forward thì kết quả chỉ còn 42.28. Chính vì thế ta mới thấy được tầm quan trọng của các hàm activation trong việc xây dựng một mạng Neural và việc sử dụng chúng như thế nào cho hợp lí là một nghệ thuật.

Ngoài ra, kết quả cao nhất thuộc về cấu hình của hàm loss SmoothL1 với F1 Score 53.03. So với Binary Cross Entropy và MSE thì SmoothL1 tạo một criterion (Dùng để backward trong Pytorch) bình phương nếu sai số tuyệt đối của các phần tử giảm xuống thấp hơn β. So với MSE thì SmoothL1 ít bị ảnh hưởng bởi các outliers và trong một số trường hợp thì hàm loss này giúp giảm thiểu việc bị bùng nổ gradient tức là gradients có giá trị lớn hơn trong quá trình backpropagation, dẫn đến một số layers có giá trị weights cập nhật quá lớn, dẫn đến chúng khổng thể hội tụ được. Chính vì thế mà SmoothL1 được sử dụng rất là phổ biến trong các mô hình học sâu hiện đại ngày nay như Fast R-CNN, Faster R-CNN.

6) Support Vector Machine: Bảng IV cho thấy thời gian huấn luyện SVMs lâu hơn đáng kể so với các phương pháp khác mặc dù đã thực hiện giảm bớt số lượng mẫu huấn luyện (700.000 mẫu còn 200.000 mẫu). Kết quả này là do độ phức tạp quá trình huấn luyện SVMs phụ thuộc nhiều vào kích thước của tập dữ liệu.

Trong số các kernel đã thực nghiệm, kernel 'poly' và 'rbf' đạt độ chính xác cao nhất và cạnh tranh nhau. kernel 'rbf' có phần tốt hơn ở Accuracy hơn kém hơn ở độ đo F1 score so với kernel 'poly' ở các giá trị C khác nhau. Tuy nhiên, kernel 'poly' có xu hướng tăng thời gian huấn luyện cấp số nhân theo chiều tăng của tham số regularization C, đây là một nhược điểm lớn khi so với kernel 'rbf'. So với các phương pháp khác, SVMs thấp hơn về độ chính xác và thấp hơn đáng kể ở F1 score. Lý do thứ nhất có thể do tập dữ liệu ít hơn khi huấn luyện SVMs, thứ hai, SVMs không phù hợp với dữ liệu mất cân bằng, vấn đề về tối ưu hóa soft margin dẫn đến các hyperplane bị lệch về lớp thiểu số khi dữ liệu mất cân bằng.

Thực nghiệm dày đặc và kết quả của GridSearchCV ở bảng III chứng minh rằng kernel RBF đạt kết quả tốt nhất và thời

gian huấn luyện ổn định nhất. Kết quả này có thể giải thích là do kernel RBF hoạt động tương tự như thuật toán K-Nearest Neighborhood, nó có các ưu điểm của K-NN và khắc phục được vấn đề phức tạp về không gian vì RBF Kernel Support Vector Machines chỉ cần lưu trữ các Support Vectors trong quá trình huấn luyện chứ không phải toàn bộ tập dữ liệu.

Tóm lại, khi sử dụng phương pháp SVM cho bài toán phân loại, kernel RBF có thể được ưu tiên sử dụng, giá trị C được lựa chọn khác nhau bởi nó có sự đánh đổi giữa độ chính xác và thời gian huấn luyện. Mặt khác, nghiên cứu đã xác thực rằng SVMs không phù hợp cho bộ dữ liệu mất cân bằng và có quy mô lớn so với các phương pháp Machine Learning khác.

7) Linear Discriminant Analysis: Cuối cùng, các bộ tham số tốt nhất của mỗi mô hình trên sẽ được thực nghiệm trên bộ dữ liệu sau khi giảm chiều bởi LDA (Linear Discriminant Analysis). Về thời gian huấn luyện, tất cả mô hình đều được huấn luyện nhanh hơn so với ban đầu, đặc biệt là trên Logistic Regression (từ 44.23s giảm xuống 2.36s) bởi LDA đã làm cho kích thước dữ liệu giảm 18 lần (dữ liệu 18 chiều về 1 chiều). Về độ chính xác, hầu hết f1 score của các mô hình đều giảm, trong đó giảm nhiều nhất là trên Decision Trees (giảm khoảng 4%) cho thấy độ phân tán tuyến tính (linearly separable) của bộ Covid-19 là không cao.

Tóm lại, trong bài toán Covid-19 Risk Prediction trên bộ Covid-19 Dataset này, LDA được sử dụng với tác dụng giảm thời gian huấn luyện của mô hình. Nếu kết hợp thì hiệu quả nhất là kết hợp với Logistic Regression, vừa giảm thời gian huấn luyện nhiều mà độ chính xác giảm không đáng kể.

V. Conclusion

Dự báo chính xác về mức độ nguy kịch của COVID-19 là một trong những yếu tố chính giúp ngăn ngừa và hạn chế sự lây lan của đại dịch này bằng cách cung cấp thông tin liên quan để giúp các nhà quản lý bệnh viện đưa ra quyết định và quản lý phù hợp các nguồn lực, nhân viên sẵn có và tình trạng bệnh nhân. Nghiên cứu này nhằm mục đích xây dựng các mô hình học máy để cải thiện chất lượng phân loại bệnh nhân Covid-19 nguy kịch. Từ bộ dữ liệu Covid-19 ban đầu, chúng tôi đã cung cấp những bước phân tích, xử lí dữ liệu để phục vụ cho bài toán này, cung cấp các thực nghiệm khi tuning các siêu tham số trên 6 phương pháp Machine Learning và đưa ra những so sánh, phân tích, đánh giá để mô hình đạt hiệu quả tốt. Decision Trees đạt độ chính xác cao nhất nhưng vẫn tối ưu được tốc độ huấn luyện, trong khi Support Vector Machine gặp nhiều thách thức về mặt thời gian tính toán trên bộ dữ liệu quy mô lớn.

Acknowledgment

Sự thành công của nghiên cứu này nhờ vào những đóng góp của mỗi cá nhân:

- **Ngô Văn Tấn Lưu** - Decision Trees
- **Nguyễn Văn Toàn** - Naive Bayes
- **Trương Thành Thắng** - Linear Discriminant Analysis, Phân tích và xử lí dữ liệu
- **Trần Văn Lực** - K-Nearest Neighbor
- **Ngô Ngọc Sương** - Logistic Regression
- **Nguyễn Đức Anh Phúc** - Neural Networks
- **Huỳnh Viết Tuấn Kiệt** - Support Vector Machine

Chân thành cảm ơn sự đóng góp về mặt kiến thức, thực nghiệm và phân tích chi tiết của các thành viên trong nghiên cứu, góp một phần trong công cuộc chống sự bùng phát của đại dịch Covid-19 trong tương lai.

TABLE IV: Experimental Results

Method	Config	precision	recall	f1_score	accuracy	time
Decision Trees	criterion='gini'	62.65	56.17	59.24	91.82	3.25
	criterion='gini', max_depth=15	64.47	52.53	57.89	91.91	2.72
	criterion='gini', max_depth=20	63.63	56.90	60.01	92.00	3.31
	criterion='gini', max_depth=25	62.84	56.61	59.61	91.87	3.54
	criterion='entropy'	63.02	56.36	59.51	91.88	3.39
	criterion='entropy', max_depth=15	65.47	53.63	58.96	92.1	3.55
	criterion='entropy', max_depth=20	64.96	56.02	60.13	92.14	4.15
	criterion='entropy', max_depth=25	63.43	57.18	**60.14**	91.98	4.18
Naive Bayes	GaussianNB	40.54	68.58	50.96	86.03	0.24
	MultinomialNB	45.25	70.63	55.16	87.85	0.13
	BernoulliNB	49.15	64.42	55.38	89.18	0.27
Linear Discriminant Analysis	solver='svd', cls=Decision Trees	62.21	49.61	55.18	91.48	2.38 + 1.85
	solver='svd', cls=Naive Bayes	50.77	58.18	54.22	89.60	0.05 + 1.77
	solver='svd', cls=K-Nearest Neighborl	59.98	48.78	53.80	91.14	0.35 + 1.67
	solver='svd', cls=Logistic Regression	59.13	42.32	48.98	90.67	2.36 + 1.8
	solver='svd', cls=Neral Networks	58.87	39.90	47.49	90.68	17.9 + 1.7
	solver='svd', cls=Support Vector Machine	59.85	36.63	45.44	90.69	19239.73 + 1.64
K-Nearest Neighbor	n_neighbors=5,algorithm='kd_tree',metric='manhattan'	65.14	52.46	58.11	92.00	4.03
	n_neighbors=9,algorithm='kd_tree',metric='euclidean'	65.41	52.20	58.06	92.02	4.21
	n_neighbors=9,algorithm='kd_tree',metric='manhattan'	65.38	52.24	58.07	92.02	4.18
	n_neighbors=9,algorithm='kd_tree',metric='minkowski'	65.36	52.27	58.08	92.02	4.16
	n_neighbors=13,algorithm='kd_tree',metric='manhattan'	**66.15**	51.22	57.73	**92.06**	4.62
	n_neighbors=13,algorithm='ball_tree',metric='manhattan'	65.84	50.52	57.07	91.98	3.78
	n_neighbors=30,algorithm='kd_tree',metric='manhattan'	65.59	51.34	57.53	91.99	4.00
Logistic Regression	solver='liblinear', penalty='l1'	59.98	39.66	47.75	90.81	5.60
	solver='liblinear', penalty='l2'	59.98	39.63	47.72	90.81	3.86
	solver='saga', penalty='l1'	59.97	39.67	47.76	90.81	44.23
	solver='saga', penalty='l2'	59.97	39.67	47.75	90.81	36.56
	solver='lbfgs', penalty='l2'	59.96	39.59	47.69	90.81	13.62
	solver='newton-cg', penalty='l2'	59.97	39.66	47.75	90.81	21.62
Neural Networks	optmizer=Adam, learning_rate=0.001, loss=BCE, activate=RELU	60.19	44.23	50.99	91.02	26.7
	optmizer=SGD	69.82	13.65	22.84	90.26	25.3
	optmizer=AdamW	59.88	45.21	51.52	91.95	28.6
	learning_rate=0.01	63.32	35.85	45.78	91.03	26.8
	learning_rate=0.0001	58.84	38.29	46.39	90.66	26.75
	loss=MSE	60.91	42.18	49.84	91.04	28.4
	loss=SmoothL1	58.60	48.42	53.03	90.94	28.7
	activation=None	62.16	32.03	42.28	90.76	27.1
	activation=Tanh	62.02	39.86	48.53	91.07	28.20
Support Vector Machine	C=0.1, kernel='poly'	20.31	68.29	31.31	90.53	575
	C=0.1, kernel='rbf'	0.52	**85.94**	1.03	89.42	1243
	C=0.1, kernel='sigmoid'	3.95	4.42	4.17	80.7	1587
	C=1.0, kernel='poly'	25.16	66.71	36.54	90.72	1058
	C=1.0, kernel='rbf'	23.98	67.68	35.41	90.71	1193
	C=1.0, kernel='sigmoid'	4.51	4.64	4.58	80.02	854
	C=10.0, kernel='poly'	26.24	67.24	37.74	90.81	1989
	C=10.0, kernel='rbf'	29.06	67.8	40.31	90.86	1923
	C=10.0, kernel='sigmoid'	4.57	4.61	4.59	79.82	1333
	C=100.0, kernel='poly'	29.42	66.76	40.84	90.95	15250
	C=100.0, kernel='rbf'	29.27	67.27	40.79	90.97	2675
	C=100.0, kernel='sigmoid'	4.57	4.6	4.58	79.8	1350