Dự đoán khả năng nhiễm Covid-19 dựa trên nhiều phương pháp Máy Học *

Nguyễn Huy Khánh
MSSV: 20521451

Võ Duy Khang
MSSV: 20521441

Lê Nguyễn Tiến Đạt
MSSV: 20521167

***Tóm tắt nội dung*—Covid-19 thường được biết đến là "chủng 2019 coronavirus", việc xuất hiện của dịch bệnh mới này đã hủy hoại nhiều nền kinh tế trên toàn thế giới.Do không có các cách hiệu quả để dự đoán khả năng bị nhiễm Covid-19, các quan chức chính phủ và các nhà chính trị của các quốc gia trên toàn thế giới đều gặp khó khăn trong việc thực hiện các biện pháp phòng ngừa nhằm giảm thiểu rủi ro. Bài báo này trình bày những mô hình máy học được đào tạo để dự đoán khả năng nhiễm Covid-19 bằng cách sử dụng các thuật toán khá phổ biến hiện nay như KNN, Decision Tree và XGBoost. Bằng cách so sánh các chỉ số đánh giá của 3 mô hình với nhau, chúng tôi suy luận ra rằng mô hình hoạt động tốt hơn với thời gian đào tạo ít hơn, tỷ lệ chính xác cao hơn.**

I. Introduction

A. Bài toán

Covid-19 lần đầu được phát hiện ở Vũ Hán, Trung Quốc và dần dần nó đã lan rộng ra toàn thế giới trong một thời gian vô cùng ngắn ngủi. WHO đã gọi đây là đại dịch toàn cầu vào ngày 11 tháng 3 năm 2020. SARS-CoV-2(Hội chứng hô hấp cấp tính cực độ do virus Corona 2) gây ra căn bệnh Covid-19.Người già, bệnh nhân tiểu đường, béo phì và bệnh tim mạch có nhiều khả năng bị ảnh hưởng bởi virus này. Vì đại dịch Covid-19 là một trong những cuộc khủng hoảng nghiêm trọng nhất trên toàn thế giới, nếu không có các biện pháp phòng ngừa(như tiêm, phong tỏa, vv) thì không thể giảm tỷ lệ tử vong do căn bệnh. Bài viết này nhằm mục đích cung cấp phương pháp có thể dự đoán khả năng nhiễm Covid-19 của một người để có thể thực hiện các bước cần thiết để giảm mức độ nghiêm trọng của dịch bệnh. Trong bài báo này, 3 mô hình đã được đề xuất có thể dự đoán khả năng nhiễm Covid-19 dựa trên nhiều phân tích đã được ghi lại. Các mô hình được đề xuất được xây dựng lần lượt bằng ba thuật toán phân loại như KNN, Decision Tree và XGBoost và các số liệu đánh giá của các mô hình đưa ra khá ổn. Bài báo này sẽ thúc đẩy các nhà nghiên cứu khác tìm ra các giải pháp tốt hơn để dự báo được khả năng nhiễm Covid-19 của một cá nhân hay một tập thể.

B. Dataset

Dataset được sử dụng trong bài báo cáo này được lấy từ Kaggle và có tên là "Covid-19 dataset". Tác giả của bộ dataset là Meir Nizri và được cung cấp chính thức bởi chính phủ Mexico.

Covid-19 luôn là 1 đề tài bức thiết cần được giải quyết và có những phương pháp để hạn chế sự lây lan của đại dịch này, vì vậy nhóm em quyết định làm đề tài này để giúp cho việc phòng tránh Covid-19 của mọi người được hiệu quả hơn. Nhóm chúng em chọn dataset này bởi vì nó chứa rất nhiều thông tin hữu ích cho việc dự đoán khả năng nhiễm bệnh Covid-19 của một người. Dataset chứa một lượng lớn thông tin liên quan đến các bệnh nhân được ẩn danh. Bộ dự liệu khi chưa qua xử lý bao gồm 21 features và 1.048.576 bệnh nhân.Trong các cột Boolean feature, 1 mang nghĩa là có và 2 mang nghĩa là 0, các giá trị như 97 và 99 là các dữ liệu null.

1) Sex (giới tính): 1 có nghĩa là phụ nữ, 2 là đàn ông
2) Age : tuổi của bệnh nhân
3) Classification: giá trị từ 1-3 có nghĩa là bệnh nhân đang trong giai đoạn điều trị covid-19 với nhiều cấp độ khác nhau, 4 và cao hơn là bệnh nhân không nhiễm Covid-19 hay kết quả test không đáng tin.
4) Patient type: 1 là bệnh nhân được trả về bệnh viện, 2 là đang được chăm sóc tại bệnh viện
5) Pneumonia: Bệnh nhân có bị viêm phổi hay không
6) Pregnancy: Bệnh nhân có thai hay không.
7) Diabetes: bệnh nhân có bị tiểu đường hay không.
8) Copd: Cho biết bệnh nhân có mắc bệnh phổi tắc nghẽn mạn tính hay không.
9) Asthma: bệnh nhân có bị hen suyễn hay không.
10) Inmsupr: bệnh nhân có bị ức chế miễn dịch hay không.
11) Hypertension: bệnh nhân có tăng huyết áp hay không.
12) Cardiovascular: bệnh nhân có bệnh liên quan đến tim mạch hay không.
13) Renal chronic: bệnh nhân có bệnh thận mạn hay không.
14) Other disease: bệnh nhân có bệnh khác hay không.
15) Obesity: bệnh nhân có béo phì hay không.
16) Tobacco: cho dù bệnh nhân là người sử dụng thuốc lá.
17) Usmr: Cho biết bệnh nhân đã điều trị cho các đơn vị y tế tuyến 1, 2 hay 3.
18) Medical unit: loại tổ chức của Hệ thống Y tế Quốc gia cung cấp dịch vụ chăm sóc.
19) Intubed: liệu bệnh nhân có được kết nối với máy thở hay không.
20) Icu: Cho biết liệu bệnh nhân đã được đưa vào Đơn vị Chăm sóc Đặc biệt hay chưa.
21) Date died: Nếu bệnh nhân chết cho biết ngày chết và 9999-99-99 nếu không.

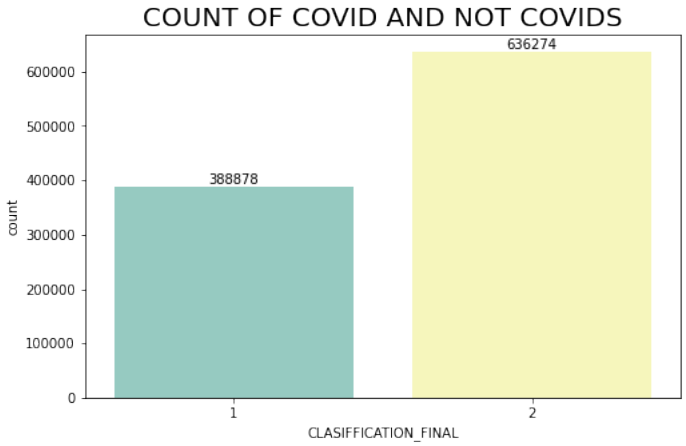
C. Data Preprocessing

Ở cột pregnancies, 98 có nghĩa là giá trị null cho phụ nữ và 97 là giá trị null cho đàn ông, vì vậy ta sẽ chuyển đổi 97 thành 2 vì đàn ông không thể có bầu. Tiếp theo, Intubed và ICU feature có quá nhiều giá trị null vì vậy ta sẽ xóa chúng đi.Về cột

Identify applicable funding agency here. If none, delete this.

classification-final cho biết khả năng nhiễm bệnh covid-19 của người đó, ta có 1-2-3 là người đó bị mắc covid, còn 4 tới 7 là bình thường, vì vậy ta sẽ chuyển class 1-2-3 thành class 1, còn class từ 4 tới 7 sẽ thành 2.

Số lượng người mắc Covid-19 và bình thường được biểu thị trực quan qua biểu đồ sau:



Hình 1: Số lượng người dính Covid-19 và người bình thường

Ta sẽ tách cột die date ra làm hai cột riêng biệt là MONTH và YEAR để cho việc tiện xử lý và trực quan hóa dữ liệu. Dưới đây là số lượng người chết được sắp theo năm và tháng:

Cuối cùng ta sẽ drop cột date die, month và year đi để dataset đỡ nặng. Sau đó giảm các đơn vị trong cột CLASSIFICATION-FINAL xuống 1 đơn vị để trở về 2 class là 0 và 1.

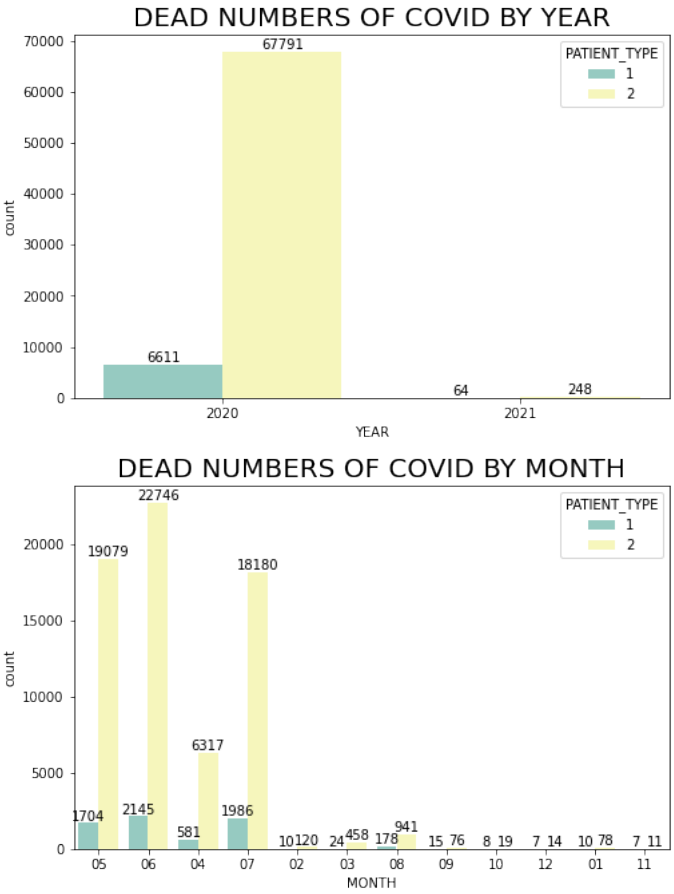
Class	Number
0	388878
1	636274

D. Các thuật toán

Decision Tree (**Võ Duy Khang** phụ trách) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượngncó thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary) , Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal

XGBoost (**Lê Nguyễn Tiến Đạt** phụ trách) là viết tắt của Extreme Gradient Boosting. Đây là thuật toán state-of-the-art nhằm giải quyết bài toán supervised learning cho độ chính xác khá cao.

KNN (**Nguyễn Huy Khánh** phụ trách) là một thuật supervised learning classifier, sử dụng khoảng cách gần để phân loại hoặc dự đoán về việc nhóm một điểm dữ liệu riêng lẻ. Mặc dù nó có thể được sử dụng cho các vấn đề hồi quy hoặc phân loại, nhưng nó thường được sử dụng như một thuật toán phân loại, dựa trên giả định rằng các điểm tương tự có thể được tìm thấy gần nhau.



Hình 2: Số lượng người chết do dính covid-19 theo năm và ngày

II. Methods

A. XGBOOST

XGBoost là viết tắt của Extreme Gradient Boosting. Đây là thuật toán state-of-the-art nhằm giải quyết bài toán supervised learning cho độ chính xác khá cao bên cạnh mô hình Deep learning như chúng ta từng tìm hiểu.

Nếu Deep learning chỉ nhận đầu vào là raw data dạng numerical (ta thường phải chuyển đổi sang n-vector trong không gian số thực) thì XGBoost nhận đầu vào là tabular datasets với mọi kích thước và dạng dữ liệu bao gồm cả categorical mà dạng dữ liệu này thường được tìm thấy nhiều hơn trong business model, đây là lý do đầu tiên tại sao các cá nhân tham gia Kaggle thường sử dụng.

Bên cạnh đó, XGboost có tốc độ huấn luyện nhanh, có khả năng scale để tính toán song song trên nhiều server, có thể tăng tốc bằng cách sử dụng GPU, nhờ vậy mà Big Data không phải là vấn đề của mô hình này. Vì thế, XGBoost thường được sử dụng và đã giành được nhiều chiến thắng trong các cuộc thi tại Kaggle.

Chi tiết thuật toán: Một phương pháp đào tạo được thuật toán XGBoost sử dụng cho hàm mục tiêu tối ưu hóa. Do đó, mỗi bước trong quá trình tối ưu hóa phụ thuộc vào bước trước

về mặt kết quả. Biểu thức toán học của hàm mục tiêu của mô hình XGBoost được liệt kê dưới đây:

$$\sum_{k=1}^{n} .l(y_k, y'_k)^{i-1} + f_i(x_k))) + R(f_i) + C \quad (1)$$

trong đó số hạng mất lần lặp thứ t được cho là l, số hạng không đổi là C và phép chính quy hóa tham số R của mô hình được mô tả thêm như sau:

$$R(f_i) = \gamma.T_i + \frac{\lambda}{2}.\sum_{j=1}^{T} .\omega_i^2 \quad (2)$$

Mức độ đơn giản của cấu trúc cây tỷ lệ thuận với giá trị của g và l thông số tùy biến. Giá trị của các tham số càng lớn thì càng đơn giản cấu trúc cây và ngược lại. Đạo hàm bậc nhất và bậc hai của mô hình, g và h tương ứng, được cho như sau:

$$g_j = \delta_({y'_k}^{i-1}).l(y_k, {y'_k}^{i-1}) \quad (3)$$

$$h_j = (\delta_({y'_k}^{i-1}))^2.l(y_k, {y'_k}^{i-1}) \quad (4)$$

Các công thức sau đây được sử dụng để thu được kết quả:

$$W_j^* = -\frac{\sum .g_t}{\sum .h_t + \lambda} \quad (5)$$

$$F_o^* = -\frac{1}{2}.\sum_{j=1}^{T} .\frac{(\sum .g)^2}{\sum .h + \lambda} + \gamma.T \quad (6)$$

Trong đó điểm hàm mất mát được cho là

$$F_o^* \quad (7)$$

Các trọng số của nó là

$$W_j^* \quad (8)$$

Các siêu tham số trong XGBoost::

1. **learning rate:** Làm cho mô hình tốt hơn bằng cách thu nhỏ trọng số trên mỗi bước Các giá trị thường được sử dụng nằm trong khoảng 0,01-0,2
2. **n-estimators:** Số lượng cây mà model sẽ duyệt qua
3. **max-depth:** Độ sâu của một cây, thường dùng để tránh over-fitting
4. **min-child-weight:** Xác định số lượng tối thiểu các trọng số của các biến đc quan sát trong một mẫu, thường dùng để tránh over-fitting.
5. **gamma:** Chỉ định mức giảm tổn thất tối thiểu cần thiết để thực hiện phân tách. Các giá trị có thể khác nhau tùy thuộc vào hàm mất mát và cần được điều chỉnh.
6. **sub-sample:** Giá trị càng thấp càng giúp thuật toán được tốt hơn, dùng để tránh over-fitting nhưng giá trị nhỏ cũng có thể dẫn đến under-fitting
7. **colsample-bytree:** Biểu hiện tỷ lệ cột được chọn ngẫu nhiên để làm mẫu cho cây.

Áp dụng thuật toán XGBoost vào bài toán, sau khi tuning tham số bằng phương pháp RandomizeSearchCV ta có được các tham số tốt nhất cho model như sau:

- learning rate: 0.1
- max-depth: 9
- min-child-weight: 7
- gamma: 0.1
- colsample-bytree: 0.4
- sub-sample: 1

B. Decision Tree

Decision tree là gì : -Là một phương pháp học có giám sát phi tham số được sử dụng để phân loại và hồi quy -Là cây mà mỗi nút biểu diễn một đặc trưng(tính chất), mỗi nhánh(branch) biểu diễn một quy luật(rule) và mỗi lá biểu biễn một kết quả (giá trị cụ thể hay một nhánh tiếp tục) -Mục tiêu là tạo ra một mô hình dự đoán giá trị của biến mục tiêu bằng cách học các quy tắc quyết định đơn giản được suy ra từ các tính năng dữ liệu.

Xây dựng Decision tree:

- Bước 1: Bắt đầu cây với nút gốc (Đặt tên: S), nút này chứa tập dữ liệu hoàn chỉnh.
- Bước 2: Tìm thuộc tính tốt nhất trong tập dữ liệu bằng cách sử dụng Phép đo lựa chọn thuộc tính (ASM).
- Bước 3: Chia S thành các tập con chứa các giá trị có thể có cho các thuộc tính tốt nhất.
- Bước 4: Tạo nút cây quyết định chứa thuộc tính tốt nhất.
- Bước 5: Tạo một cách đệ quy cây quyết định mới bằng cách sử dụng các tập con của tập dữ liệu đã tạo ở bước -3. Tiếp tục quá trình này cho đến khi đạt đến một giai đoạn mà bạn không thể phân loại thêm các nút và được gọi là nút cuối cùng là nút lá.

Trong khi thực hiện cây Quyết định, vấn đề chính nảy sinh là làm thế nào để chọn thuộc tính tốt nhất cho nút gốc và cho các nút con. Vì vậy, để giải quyết những vấn đề như vậy có một kỹ thuật được gọi là thước đo lựa chọn thuộc tính hoặc ASM. Bằng phép đo này, chúng ta có thể dễ dàng chọn thuộc tính tốt nhất cho các nút của cây. Có hai kỹ thuật phổ biến cho ASM, đó là:
o Gini index (CART)+ Information Gain

Entropy trong Cây quyết định (Decision Tree): Entropy là khái niệm chính của thuật toán này, giúp xác định một tính năng hoặc thuộc tính cung cấp thông tin tối đa về một lớp được gọi là thuật toán Độ lợi thông tin hoặc thuật toán ID3. Bằng cách sử dụng phương pháp này, chúng ta có thể giảm mức entropy từ nút gốc xuống nút lá. Công thức toán học :

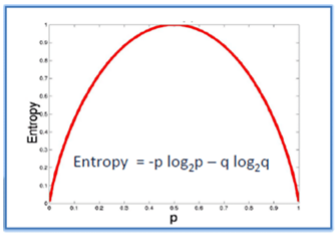
$$E(S) = \sum_{i=1}^{c} .(-p.log_2.p_i) \quad (9)$$

p ', biểu thị xác suất của E(S), biểu thị entropy. Tính năng hoặc thuộc tính có mức tăng ID3 cao nhất được sử dụng làm gốc để phân tách. Với một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau x1,x2,. . . ,xn. Giả sử rằng xác suất để x nhận các giá trị này là pi=p(x=xi). Ký hiệu phân phối này là p=(p1 ,p2 ,. . . ,pn). Entropy của phân phối này được định nghĩa là:

$$H(p) = -\sum_{n=1}^{n} .p_i.log(p_i) \quad (10)$$

Giả sử bạn tung một đồng xu, entropy sẽ được tính như sau:

$$H = -[0.5.ln(0.5) + 0.5.ln(0.5)] \quad (11)$$



Entropy = -0.5 $\log_2$0.5 – 0.5 $\log_2$0.5 = 1

Hình 3: Hình vẽ trên biểu diễn sự thay đổi của hàm entropy. Ta có thể thấy rằng, entropy đạt tối đa khi xác suất xảy ra của hai lớp bằng nhau.

• P tinh khiết: Pi = 0 hoặc Pi = 1

• P vẩn đục: Pi = 0.5, khi đó hàm Entropy đạt đỉnh cao nhất

Giải thuật ID3 xây dựng cây quyết định được trình bày như sau:

- 1. Chọn A <= thuộc tính quyết định "tốt nhất" cho nút kế tiếp
- 2. Gán A là thuộc tính quyết định cho nút
- 3. Với mỗi giá trị của A, tạo nhánh con mới của nút
- 4. Phân loại các mẫu huấn luyện cho các nút lá
- 5. Nếu các mẫu huấn luyện được phân loại hoàn toàn thì NGƯNG, Ngược lại, lặp với các nút lá mới. (Thuộc tính tốt nhất ở đây là thuộc tính có entropy trung bình thấp nhất theo thuộc tính)

Information Gain trong Cây quyết định (Decision Tree) Information Gain dựa trên sự giảm của hàm Entropy khi tập dữ liệu được phân chia trên một thuộc tính. Để xây dựng một cây quyết định, ta phải tìm tất cả thuộc tính trả về Infomation gain cao nhất. Để xác định các nút trong mô hình cây quyết định, ta thực hiện tính Infomation Gain tại mỗi nút theo trình tự sau:

- Bước 1: Tính toán hệ số Entropy của biến mục tiêu S có N phần tử với Nc phần tử thuộc lớp c cho trước:

$$H(S) = -\sum_{c=1}^{c}(\frac{N_c}{N}).log(\frac{N_c}{N}) \quad (12)$$

- Bước 2: Tính hàm số Entropy tại mỗi thuộc tính: với thuộc tính x, các điểm dữ liệu trong S được chia ra K child node S1, S2, . . . , SK với số điểm trong mỗi child node lần lượt là m1, m2 ,. . . , mK , ta có:

$$H(x,S) = \sum_{k=1}^{k}.(\frac{m_k}{N}).H(S_k) \quad (13)$$

- Bước 3: Chỉ số Gain Information được tính bằng:

$$H(x,S) = H(S) - H(x,S) \quad (14)$$

Trong các thuật toán Decision tree, với phương pháp chia trên, ta sẽ chia mãi các node nếu nó chưa tinh khiết. Như vậy, ta sẽ thu được một tree mà mọi điểm trong tập huấn luyện đều được dự đoán đúng (giả sử rằng không có hai input giống nhau nào cho output khác nhau). Khi đó, cây có thể sẽ rất phức tạp (nhiều node) với nhiều leaf node chỉ có một vài điểm dữ liệu. Như vậy, nhiều khả năng overfitting sẽ xảy ra. Để tránh trường hợp này, ta có thể dừng cây theo một số phương pháp sau đây:

- nếu node đó có entropy bằng 0, tức mọi điểm trong node đều thuộc một class.
- nếu node đó có số phần tử nhỏ hơn một ngưỡng nào đó. Trong trường hợp này, ta chấp nhận có một số điểm bị phân lớp sai để tránh overfitting. Class cho leaf node này có thể được xác định dựa trên class chiếm đa số trong node.
- nếu khoảng cách từ node đó đến root node đạt tới một giá trị nào đó. Việc hạn chế chiều sâu của tree này làm giảm độ phức tạp của tree và phần nào giúp tránh overfitting.
- nếu tổng số leaf node vượt quá một ngưỡng nào đó.
- nếu việc phân chia node đó không làm giảm entropy quá nhiều (information gain nhỏ hơn một ngưỡng nào đó).

Tuning siêu tham số cho mô hình cây quyết định

- criterion='gini',
- splitter='best',
- max-depth=None,
- min-samples-split=2,
- min-samples-leaf=1,
- max-features=None,
- max-leaf-nodes=None,
- min-impurity-decrease=0.0,
- min-impurity-split=None

Trong đó:

- criterion: Là hàm số để đo lường chất lượng phân chia ở mỗi node. Có hai lựa chọn là gini và entropy.
- max-depth: Độ sâu tối đa cho một cây quyết định. Đối với mô hình bị quá khớp thì chúng ta cần giảm độ sâu và vị khớp thì gia tăng độ sâu.
- min-samples-split: Kích thước mẫu tối thiểu được yêu cầu để tiếp tục phân chia đối với node quyết định. Được sử dụng để tránh kích thước của node lá quá nhỏ nhằm giảm thiểu hiện tượng quá khớp.
- min-samples-leafchỉ định số lượng mẫu tối thiểu cần có tại một nút lá .
- max-features: Số lượng các biến được lựa chọn để tìm kiếm ra biến phân chia tốt nhất ở mỗi lượt phân chia.
- max-leaf-nodes: Số lượng các node lá tối đa của cây quyết định. Thường được thiết lập khi muốn kiểm soát hiện tượng quá khớp.
- min-impurity-decrease: Chúng ta sẽ tiếp tục phân chia một node nếu như sự suy giảm của độ tinh khiết nếu phân chia lớn hơn ngưỡng này. im
- min-impurity-split: Ngưỡng dừng sớm để kiểm soát sự gia tăng của cây quyết định. Thường được sử dụng để tránh hiện tượng quá khớp. Chúng ta sẽ tiếp tục chia node nếu độ tinh khiết cao hơn ngưỡng này.

C. K-Nearest Neighbors

Thuật toán k-nearest neighbors, còn được gọi là KNN hoặc k-NN, là một thuật supervised learning classifier, sử dụng khoảng cách gần để phân loại hoặc dự đoán về việc nhóm một điểm dữ liệu riêng lẻ. Mặc dù nó có thể được sử dụng cho các vấn đề hồi quy hoặc phân loại, nhưng nó thường được sử dụng như một

thuật toán phân loại, dựa trên giả định rằng các điểm tương tự có thể được tìm thấy gần nhau. Thuật toán KNN cũng là một phần của họ mô hình "lazy learning", nghĩa là nó chỉ lưu trữ một tập dữ liệu đào tạo thay vì trải qua một giai đoạn đào tạo. Điều này cũng có nghĩa là tất cả các tính toán xảy ra khi phân loại hoặc dự đoán đang được thực hiện. Vì nó chủ yếu dựa vào bộ nhớ để lưu trữ tất cả dữ liệu đào tạo của nó, nên nó còn được gọi là phương pháp học tập dựa trên cá thể hoặc dựa trên bộ nhớ.

Chi tiết thuật toán: mục tiêu của thuật toán k- nearest neighbors là xác định các điểm gần nhất của một điểm truy vấn nhất định, để có thể gán nhãn lớp cho điểm đó. Để làm được điều này, KNN có một vài yêu cầu:

Xác định cách tính khoảng cách (distance metric): Để xác định điểm dữ liệu nào gần nhất với một điểm truy vấn nhất định, cần phải tính toán khoảng cách giữa điểm truy vấn và các điểm dữ liệu khác. Các số liệu khoảng cách này giúp hình thành các ranh giới quyết định, phân vùng các điểm truy vấn thành các vùng khác nhau (thường ranh giới quyết định được hiển thị bằng sơ đồ Voronoi). Một số các distance metric thường được dùng:

Khoảng cách Euclidean:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} .(y_i - x_i)^2} \tag{15}$$

Khoảng cách Manhattan:

$$d(x,y) = (\sum_{i=1}^{m} .|x_i - y_i|) \tag{16}$$

Khoảng cách Minkowski:

$$d(x,y) = (\sum_{i=1}^{n} .|x_i - y_i|)^{\frac{1}{p}} \tag{17}$$

Xác định k Giá trị k trong thuật toán k-NN xác định số lượng hàng xóm sẽ được kiểm tra để xác định phân loại của một điểm truy vấn cụ thể. Việc xác định k có thể là một hành động cân bằng vì các giá trị khác nhau có thể dẫn đến overfitting hoặc underfitting. Các giá trị k thấp hơn có thể có phương sai cao, nhưng độ lệch thấp và các giá trị k lớn hơn có thể dẫn đến độ lệch cao và phương sai thấp hơn. Việc lựa chọn k phần lớn sẽ phụ thuộc vào dữ liệu đầu vào vì dữ liệu có nhiều ngoại lệ hoặc nhiễu hơn sẽ có khả năng hoạt động tốt hơn với giá trị k cao hơn. Nhìn chung, nên chọn k lẻ để tránh ràng buộc trong phân loại và các chiến thuật xác thực chéo có thể giúp chọn k tối ưu cho tập dữ liệu.

Trọng số của hàng xóm gần nhất Có 2 cách gán trọng số cho k hàng xóm gần nhất:

- Cách 1: k hàng xóm gần nhất có trọng số bằng nhau
- Cách 2: trọng số của hàng xóm gần nhất i bằng nghịch đảo khoảng cách của hàng xóm đó

$$W_i = \sum_{i}^{k} . \frac{1}{d_i} \tag{18}$$

Trong trường hợp này, các hàng xóm gần điểm truy vấn hơn sẽ có ảnh hưởng lớn hơn các hàng xóm ở xa hơn.

Các siêu tham số trong KNN:

- n-neighbors hay còn gọi là k được đề cập ở trên
- metrics là công thức tính khoảng cách giữa các điểm
- weights là cách gán trọng số cho các hàng xóm gần nhất của điểm truy vấn

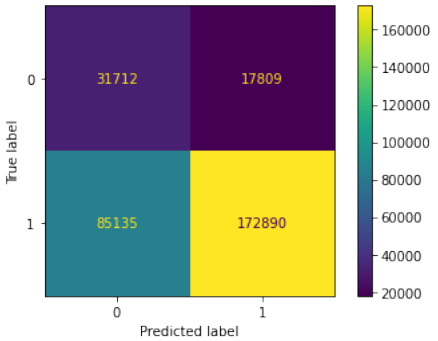
Áp dụng thuật toán KNN vào bài toán và tuning tham số bằng phương pháp RandomizeSearchCV ta có được siêu tham số tốt nhất cho mô hình như sau:

- n-neighbors: 20
- metrics: 'uniform'
- weights: 'manhattan'

III. Models evaluation

Sau khi tune tham số lần lượt vào 3 model là Decision Tree, KNN và XGBoost, ta thu được độ chính xác lên tới 0.94 từ Decision Tree.Độ hiệu quả của các model được bằng các độ đo như f1-score, precision, recall. Như bảng dưới đây, ta có thể thấy phương pháp XGBoost thu được độ chính xác chỉ gần 0.67, mặt khác độ chính xác của KNN là 0.64 thấp nhất trong 3 phương pháp mà ta sử dụng.

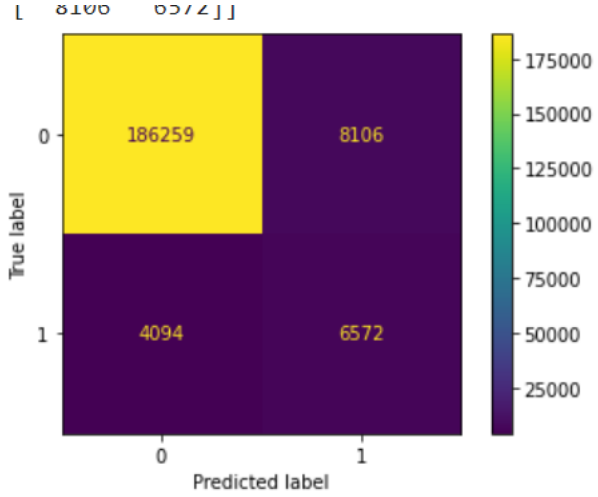
Classifier	Accuracy	Precision	F1-Score
Decision Tree	0.94	0.61	0.93
XGboost	0.67	0.9	0.77
KNN	0.64	0.64	0.33



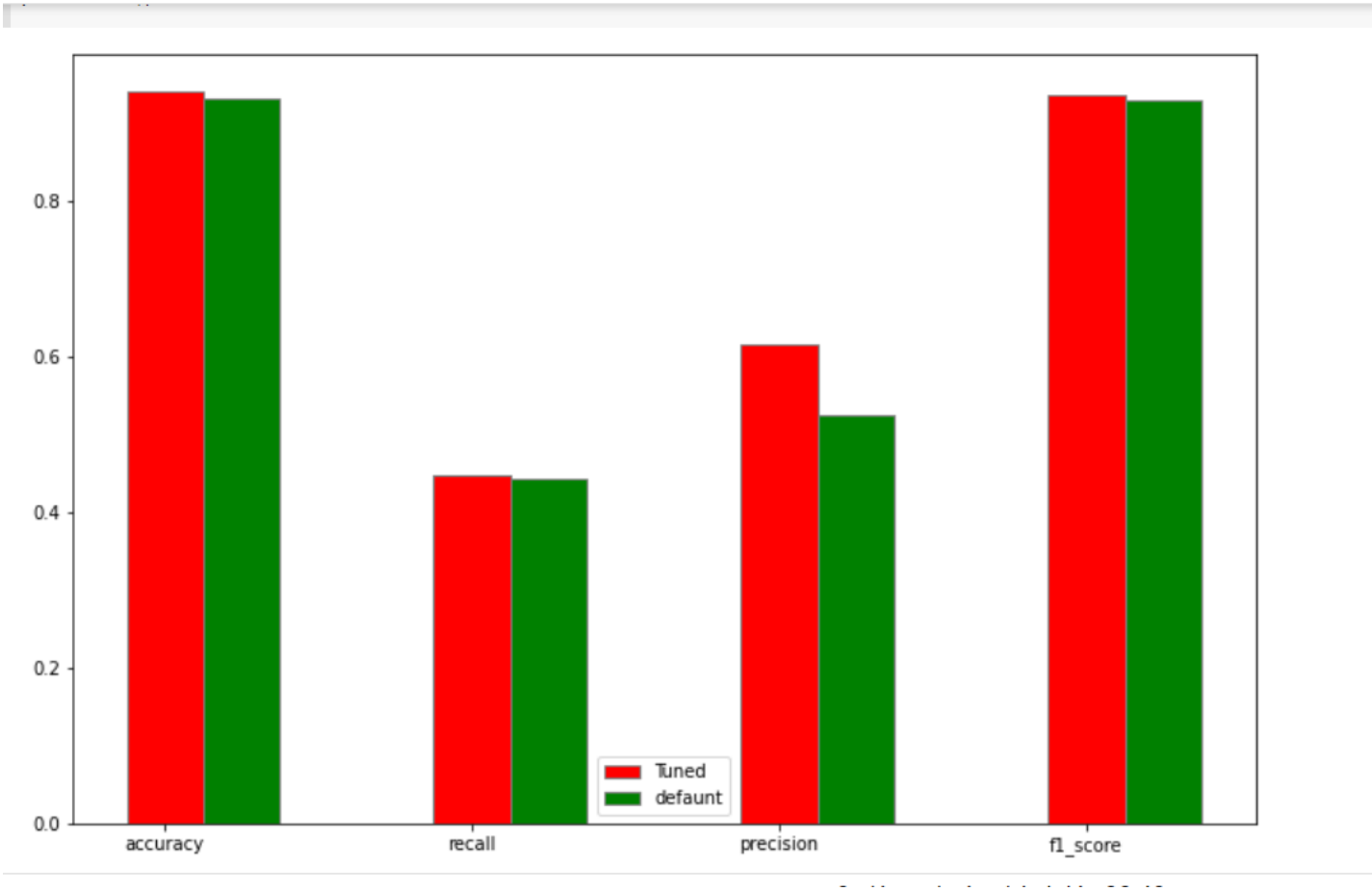
Hình 4: Confusion matrix của XGBoost

IV. Conclusion

Đồ án tập trung vào việc dự đoán khả năng nhiễm Covid-19 của một người dựa trên nhiều dữ liệu được thu thập như bệnh tình, giới tính, tuổi,sự hồi phục,...và thu được độ chính xác cao nhất là 0.94 với model Decision Tree. Mặc dù là đồ án được thực hiện chỉ trên một dataset có sẵn trên Kaggle, nó còn có thể hiệu quả trên bất cứ một bộ dataset nào có cùng số feature. Trong tương lai, có nhiều thuật toán May học có thể được thử nghiệm để cải thiện độ chính xác và có thể được mở rộng nghiên cứu thêm như dùng để chế tạo vaccine,..



Hình 5: Confusion matrix của Decision Tree



Hình 6: Performances trước và sau khi tune tham số