# PHÂN LỚP THỜI TIẾT DỰA TRÊN DỮ LIỆU THỜI TIẾT CỦA THÀNH PHỐ HỒ CHÍ MINH NĂM 2020

Lê Quang Thắng
*Khoa Khoa học máy tính*
*Trường Đại học Công nghệ Thông tin*
Thành phố Hồ Chí Minh, Việt Nam
16521099@gm.uit.edu.vn

Phan Thanh Hải
*Khoa Khoa học máy tính*
*Trường Đại học Công nghệ Thông tin*
Thành phố Hồ Chí Minh, Việt Nam
18520705@gm.uit.edu.vn

Trương Tấn Sang
*Khoa Mạng máy tính và truyền thông*
*Trường Đại học Công nghệ Thông tin*
Thành phố Hồ Chí Minh, Việt Nam
18521336@gm.uit.edu.vn

***Tóm tắt*—Thời tiết có ảnh hưởng nhất định tới nền kinh tế và giao thông vận tải. Hiện nay, đã có nhiều hệ thống dự báo thời tiết dựa trên các dữ liệu về thời tiết như nhiệt độ, độ ẩm, tốc độ gió… Trong báo cáo này, nhóm chúng em xây dựng mô hình dự đoán thời tiết dựa trên dữ liệu thời tiết của thành phố Hồ Chí Minh năm 2020. Đây là bài toán phân loại đa lớp với 4 nhãn: clear, partially cloudly, overcast và rain. Thuật toán mà nhóm sẽ sử dụng để huấn luyện mô hình là: Decision Tree, Random Forest, Logistic Regression, SVM và K-nearest Neighbor. Nhóm sử dụng các lớp và hàm có sẵn trong thư viện scikit-learn để hiện thực phần code. Kết quả đem lại có thể cho ta thấy hiệu quả của thuật toán Random Forest và Decision Tree trong việc xây dựng mô hình cho bài toán phân lớp thời tiết.**

***Từ khóa—Phân lớp thời tiết, Decision Tree, Random Forest, Logistic Regression, SVM, K-nearest Neighbor***

## I. GIỚI THIỆU

Dự báo thời tiết luôn là một lĩnh vực có sức ảnh hưởng rất lớn đến các hoạt động kinh tế – xã hội của mỗi quốc gia. Nếu chúng ta có được khả năng dự báo thời tiết chính xác, mỗi quốc gia sẽ luôn có những giải pháp kịp thời để ứng phó với các loại thời tiết khác nhau, tránh được những ảnh hưởng tiêu cực của thời tiết. Ví dụ: Một số cây trồng, vật nuôi chỉ phù hợp với một hoặc một vài dạng thời tiết nhất định, như vậy nếu ta có thể dự đoán thành công, ta có thể đảm bảo được những cây trồng, vật nuôi đó được khỏe mạnh, tránh đi những tổn thất do các hiện tượng thời tiết cực đoan gây ra.

Chính vì thế, nhóm đã dành thời gian nghiên cứu và phát triển mô hình máy học để thực hiện việc dự đoán tình trạng thời tiết ở Thành phố Hồ Chí Minh.

Do ảnh hưởng của thời tiết có liên quan mật thiết đến hầu hết các vấn đề kinh tế - xã hội, mô hình trên sẽ có ứng dụng rộng rãi trong đời sống của công dân Thành phố Hồ Chí Minh như hỗ trợ thông tin để sắp xếp các buổi gặp mặt, tổ chức các sự kiện giải trí, thể thao, v.v…

Đầu vào của bài toán là các thông số như như thời gian, nhiệt độ, độ ẩm, sức gió, tầm nhìn xa,… Đầu ra của bài toán cho biết tình trạng thời tiết (trời nhiều mây, trời mây…) dựa trên thông số đầu vào của bài toán.

Trong phần I, nhóm giới thiệu mục tiêu cụ thể của đồ án, bài toán máy học, ứng dụng. Phần II thì tập trung giới thiệu về tập dữ liệu có sẵn, phân tích sơ bộ tập dữ liệu. Phần III trình bày tóm tắt các thuật toán máy học mà nhóm sử dụng để huấn luyện mô hình cho bài toán. Phần IV trình bày quá trình tiền xử lí dữ liệu và huấn luyện mô hình sử dụng thuật các thuật máy học đã được trình bày ở phần III. Phần V trình bày độ đo đánh giá cho bài toán và kết quả đạt được. Phần VI phân tích lỗi và đưa ra hướng phát triển của nhóm trong tương lai.

## II. DỮ LIỆU

Bộ dữ liệu phục vụ cho bài toán được nhóm thu thập từ trang web Visual Crossing [1] – một trang web cung cấp các thông tin lịch sử thời tiết ở khắp nơi trên thế giới. Bộ dữ liệu là toàn bộ thông tin thời tiết của thành phố Hồ Chí Minh trong năm 2020 được lấy mẫu theo từng giờ.

Nhóm đã lược bỏ cột Wind Gust, Snow và Snow Depth vì các thuộc tính này toàn giá trị rỗng.

Bộ dữ liệu hiện tại gồm 7 cột, gồm 8784 mẫu:

- Date time (thời gian): là thời gian lấy mẫu, trong đó bao gồm thông tin ngày ở trước và thông tin giờ ở sau.
- Temperature (nhiệt độ): là nhiệt độ trung bình tại thời điểm lấy mẫu, đơn vị là °C.
- Wind Speed (tốc độ gió): là tốc độ gió lớn nhất tại thời điểm lấy mẫu, đơn vị là km/h.
- Visibility (tầm nhìn): là khoảng cách xa nhất mà con người có thể quan sát bằng mắt các vật thể trên nền trời, tùy thuộc vào độ trong suốt của khí quyển [2], đơn vị là km.
- Cloud Cover (tình trạng mây che phủ): là phần của bầu trời che khuất bởi những đám mây khi quan sát được từ một vị trí cụ thể [3].
- Relative Humidity (độ ẩm tương đối): là tỉ lệ giữa giữa áp suất riêng phần của hơi nước và áp suất hơi bão hòa của nước ở cùng nhiệt độ [4], đơn vị là %.
- Conditions (trạng thái thời tiết): là trạng thái của thời tiết hiện tại như nhiều mây, mưa, trời u ám, trời có sấm…

Thuộc tính cần được phân lớp chính là thuộc tính Conditions. Thuộc tính này gồm 4 giá trị chính: clear (trời thoáng đãng), partially cloudly (trời nhiều mây), overcast (trời u ám) và rain (trời mưa). Bảng dưới đây trình bày tỉ lệ của các giá trị này trong bộ dữ liệu:

BẢNG 1. TỈ LỆ PHÂN BỐ CÁC GIÁ TRỊ TRONG THUỘC TÍNH CONDITIONS TRONG BỘ DỮ LIỆU

| Nhãn | Số lượng | Tỉ lệ |
|---|---|---|
| Clear | 6802 | 77.5% |
| Partially cloudy | 1548 | 17.6% |
| Overcast | 219 | 2.50% |
| Rain | 209 | 2.40% |

Có thể thấy bộ dữ liệu có sự chênh lệch rất lớn giữa tỉ lệ phân bố các giá trị: giá trị chiếm số lượng nhiều nhất và chiếm hơn 3/4 toàn bộ điểm dữ liệu là rain, trong khi giá trị chiếm số lượng thấp nhất chỉ chiếm hơn 2% là rain. Đây là một bộ dữ liệu không cân bằng.

Bộ dữ liệu được tiến hành chia thành 2 tập là train và test với tỉ lệ 8:2, trong đó tập train có 7020 mẫu với 1756 mẫu. Nhóm sử dụng `train_test_split()` trong scikit-learn để tiến hành việc chia tập dữ liệu với tỉ lệ phù hợp sau cho tỉ lệ của các nhãn trong tập train và tập test gần bằng nhau.

## III. PHƯƠNG PHÁP MÁY HỌC SỬ DỤNG

Dưới đây là phần trình bày tóm tắt 5 phương pháp máy học mà nhóm sẽ sử dụng để huấn luyện mô hình cho bài toán:

### A. Decision Tree

Trong lĩnh vực máy học, Decision Tree là một kiểu mô hình dự báo (predictive model), nghĩa là một ánh xạ từ các quan sát về một sự vật/hiện tượng tới các kết luận về giá trị mục tiêu của sự vật/hiện tượng. Mỗi một nút trong (internal node) tương ứng với một biến; đường nối giữa nó với nút con của nó thể hiện một giá trị cụ thể cho biến đó. Mỗi nút lá đại diện cho giá trị dự đoán của biến mục tiêu, cho trước các giá trị của các biến được biểu diễn bởi đường đi từ nút gốc tới nút lá đó [5].

Một cách đơn giản, Decision Tree là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật bao gồm:

- *Root node*: điểm ngọn chứa giá trị đầu tiên để phân nhánh.
- *Internal node*: các điểm bên trong thân cây chứa các giá trị, thuộc tính dùng để xét đến những lần phân nhánh tiếp theo.
- *Leaf node*: là các lá cây chứa các giá trị phân loại sau cùng.
- *Branch*: là quy luật phân nhánh, có thể xem là mối quan hệ giữa giá trị của biến độc lập (internal node) và biến mục tiêu (leaf node).

Tóm lại, cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết.

Luật để xác định nút gốc của cây quyết định được dựa trên một độ đo "không thuần khiết" (impurity) gồm 2 thuật toán chính:

*1) Độ đo GINI – Thuật toán CART (Classification And Regression Tree)*

CART sử dụng công thức Gini Index để thực hiện việc quyết định dữ liệu nào được chọn làm gốc. Công thức Gini Index là chỉ số đo lường mức độ đồng nhất, mức độ nhiễu loạn thông tin, hay sự khác biết về các giá trị mà mỗi điểm dữ liệu trong tập hợp con, hoặc một nhánh (node) trên Decision Tree có.

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2 \quad (1)$$

Nếu tất cả các quan sát, điểm dữ liệu thuộc về một node và có chung 1 thuộc tính bất kì thì node này thể hiện sự đồng nhất và không có nhiễu loạn thông tin, lúc này Gini của node sẽ bằng 0, ngược lại Gini sẽ tiến sát đến 1.

Như vậy, với độ đo Gini, cách chia tối ưu của thuật toán sẽ có chỉ số Gini Index thấp nhất (tiến sát về 0) và máy tính sẽ tìm kiếm min của tất cả cách chia có thể của bộ dữ liệu.

*2) Độ đo ENTROPY – Thuật toán ID3 (Interactive Dichotomister)*

Entropy là thuật ngữ thuộc Nhiệt động lực học, là thước đo của sự biến đổi, hỗn loạn hoặc ngẫu nhiên. Năm 1948, Shannon đã mở rộng khái niệm Entropy sang lĩnh vực nghiên cứu, thống kê với công thức bên dưới [6].

- Với một phân phối xác suất của một biến rời rạc $x$ có thể nhận $n$ giá trị khác nhau $x_1, x_2, \dots, x_n$.
- Giả sử rằng xác suất để $x$ nhận các giá trị này là $p_i = p(x = x_i)$.
- Ký hiệu phân phối này là $p = (p_1, p_2, \dots, p_n)$. Entropy của phân phối này được định nghĩa là:

$$Entropy = -\sum_{i=1}^{n} p_i \log_2 p_i \quad (2)$$

Cách tính giá trị information gain:

Giả sử ta đang làm việc với một non-leaf node với các điểm dữ liệu tạo thành một tập $S$ với số phần tử là $|S| = N$. Tiếp theo, giả sử thuộc tính được chọn là $x$. Dựa trên $x$, các điểm dữ liệu trong $S$ được phân ra thành $k$ child node $S_1, S_2, \dots, S_k$ với số điểm trong mỗi child node lần lượt là $m_1, m_2, \dots, m_k$ Ta định nghĩa:

$$E(x, S) = \sum_{i=1}^{k} \frac{m_i}{N} E(S_k) \quad (3)$$

Trong đó, $E(S_k)$ là giá trị entropy tại mỗi nút con.

Công thức trên là tổng có trọng số entropy của mỗi nút con. Việc lấy trọng số này là quan trọng vì các nút thường có số lượng điểm khác nhau.

Tiếp theo, ta định nghĩa information gain dựa trên thuộc tính $x$:

$$Gain(x, S) = E(S) - E(x, S) \quad (4)$$

Việc tính toán giá trị entropy và information gain rất quan trọng trong bài toán về xây dựng và cắt tỉa cây, tức là khi tách ra thành các nút con thì cần chọn ra thuộc tính nào tốt nhất để tách. Thuộc tính này có ảnh hưởng đến kết quả sau cùng.

Bài toán của chúng ta là tìm ra thuộc tính sao cho giá trị gain information là cao nhất (đồng nghĩa với tổng có trọng số entropy ở mỗi nút con phải là thấp nhất).

### B. Random Forest

Random Forests là thuật toán học có giám sát (supervised learning). Nó có thể được sử dụng cho cả phân lớp và hồi quy. Nó cũng là thuật toán linh hoạt và dễ sử dụng nhất. Một khu rừng bao gồm cây cối. Random Forests tạo ra cây quyết định

trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu.

Random Forests có thể được mô tả với một mã giả sau [7]:

- *Bước 1:* Chọn ngẫu nhiên *k* đặc trưng từ tập *n* đặc trưng ($k \leq m$).
- *Bước 2:* Từ tập *k* đặc trưng đó, chọn ra node tốt nhất cho node phân loại.
- *Bước 3:* Chia các node con theo node tốt nhất tìm được.
- *Bước 4:* Lặp lại bước 1 đến 3 cho đến khi đạt *k* nodes.

Ta có thể thay đổi các giá trị như số thuộc tính tối đa, số nhánh cây,… đồng thời ta cũng có thể thay đổi số lượng thuộc tính *k* đầu vào để tinh chỉnh mô hình.

### C. Logistic Regression

Logistic regression là một phương pháp thống kê cho phép phân tích một tập dữ liệu mà trong đó có một hoặc nhiều hơn các biến độc lập có thể tác động tới sự xuất hiện của một kết cục. Kết cục đó được đo bởi một biến nhị phân (trong đó chỉ có hai kết cục có thể xảy ra). Với một vài thuộc tính có số nhãn có hạn, ta có thể sử dụng Logistic Regression như là một thuật toán phân lớp tốt.

Logistic Regression cũng giống như Linear Regression nhưng sử dụng hàm sigmoid để dự đoán kết quả.

Hàm ước lượng xác suất trong Logistic Regression [8]:

$$\hat{p} = h_\theta(x) = \sigma(x^T\theta) \tag{5}$$

Trong đó $\sigma(-)$ là hàm sigmoid, đầu ra của nó sẽ là khoảng giá trị từ 0 đến 1.

$$\sigma(t) = \frac{1}{1 + \exp(-t)} \tag{6}$$

Chúng ta nhận xét đây là một hàm số đặc biệt, với mọi giá trị của *t*, hàm luôn nằm trong khoảng [0, 1].

Sau khi đã tính được xác suất $\hat{p} = h_\theta(x)$, ta sẽ dễ dàng xác định được *x* thuộc về lớp nào:

$$\hat{y} = \begin{cases} 0 & nếu\ \hat{p} < 0.5 \\ 1 & nếu\ \hat{p} \geq 0.5 \end{cases} \tag{7}$$

Vì Logistic Regression là một thuật toán chỉ áp dụng được trên bài toán phân lớp nhị phân nên đối với bài toán phân lớp đa lớp thì ta cần sử dụng 1 trong 2 chiến thuật sau:

- *One-vs-one*: Xây dựng rất nhiều bộ phân lớp nhị phân cho từng cặp lớp. Bộ thứ nhất phân biệt class 1 và class 2, bộ thứ hai phân biệt class 1 và class 3,… Khi có một dữ liệu mới vào, đưa nó vào toàn bộ các bộ phân lớp nhị phân trên. Kết quả cuối cùng có thể được xác định bằng cách xem lớp nào mà điểm dữ liệu đó được phân vào nhiều nhất (major voting). Với Logistic Regression thì ta có thể tính tổng các xác suất tìm được sau mỗi bộ phân lớp nhị phân.
- *One-vs-rest*: Phương pháp được sử dụng nhiều nhất là one-vs-rest (một số tài liệu gọi là one-vs-all, one-against-rest, hoặc one-against-all). Cụ thể, nếu có *n* lớp thì ta sẽ xây dựng *n* bộ phân lớp, mỗi bộ phân lớp tương ứng với một lớp. Bộ phân lớp thứ nhất giúp phân biệt lớp 1 với lớp không phải là lớp 1, tức xem một điểm có thuộc lớp 1 hay không, hoặc xác suất để một điểm rơi vào lớp 1 là bao nhiêu. Tương tự như thế, bộ phân lớp thứ hai sẽ phân biệt lớp 2 với lớp không phải là lớp 2,… Kết quả cuối cùng có thể được xác định bằng cách xác định class mà một điểm rơi vào với xác suất cao nhất. Hàm Logistic Regression trong thư viện sklearn có thể được dùng trực tiếp để áp dụng vào các bài toán phân lớp đa lớp với phương pháp one-vs-rest [9].

### D. SVM

SVM (viết tắt tên tiếng Anh là Support Vector Machine) là một khái niệm trong thống kê và khoa học máy tính cho một tập hợp các phương pháp học có giám sát liên quan đến nhau để phân loại và phân tích hồi quy. SVM dạng chuẩn nhận dữ liệu vào và phân loại chúng vào hai lớp khác nhau.

Một mô hình SVM là một cách biểu diễn các điểm trong không gian và lựa chọn ranh giới giữa hai thể loại sao cho khoảng cách từ các ví dụ luyện tập tới ranh giới là xa nhất có thể.

Trong bài toán trên 2 chiều, ta cần dùng thuật toán để ánh xạ bộ data đó vào không gian nhiều chiều hơn (*n* chiều), từ đó tìm ra siêu mặt phẳng (hyperplane) để phân chia.

Margin (biên) là khoảng cách giữa siêu phẳng (trong trường hợp không gian 2 chiều là đường thẳng) đến 2 điểm dữ liệu gần nhất tương ứng với 2 phân lớp.

SVM cố gắng tối ưu thuật toán bằng các tìm cách cực đại hóa giá trị margin này, từ đó tìm ra siêu phẳng đẹp nhất để phân 2 lớp dữ liệu.

Bài toán của chúng ta trở thành tìm ra biên của 2 lớp dữ liệu sao cho khoảng cách giữa 2 đường này là lớn nhất.

Vì dữ liệu trong thế giới thực ít khi ở dạng phân tách tuyến tính, những kernel phi tuyến tính thường được sử dụng để biến đổi không gian điểm đang xét. Kernel tuyến tính có công thức như ở dưới công thức 6. Trong đó $x$ và $y$ là 2 vecto cần dự đoán, $C$ là một toán hạng chính quy (regularization term). Nếu $C$ quá lớn dẫn đến mô hình huấn luyện bị overfitting, hoặc nếu $C$ quá nhỏ dẫn đến mô hình huấn luyện bị underfitting.

$$K_{Linear} = x^T y + C \tag{8}$$

Với SVM phi tuyến, ta cũng có một kernel được sử dụng thông dụng như sau [10]:

Polynomial:

$$K_{Polynomial} = (\gamma x^T y + 1)^d \tag{9}$$

Gaussian RBF:

$$K_{Gaussian} = \exp(-\gamma \|x - y\|^2) \tag{10}$$

Sigmoid:

$$K_{Sigmoid} = \tanh(\gamma x^T y + r) \tag{11}$$

Trong các công thức trên xuất hiện siêu tham số $\gamma$. Giống như $C$, nếu $\gamma$ quá lớn có thể mô hình huấn luyện bị overfit, hoặc nếu $\gamma$ quá nhỏ có thể thể mô hình huấn luyện bị underfit.

### E. K-nearest Neighbor

K-nearest neighbor (kNN) là một trong những thuật toán học có giám sát đơn giản trong máy học. Khi huấn luyện mô

hình, thuật toán này không học một điều gì từ dữ liệu huấn luyện (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới.

Với kNN, trong bài toán phân lớp, nhãn của một điểm dữ liệu được suy ra trực tiếp từ *k* điểm dữ liệu gần nhất trong tập dữ liệu huấn luyện. Nhãn của một dữ liệu kiểm thử có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra nhãn. Một cách ngắn gọn, kNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách chỉ dựa trên thông tin của *k* điểm dữ liệu trong tập dữ liệu huấn luyện gần nó nhất (*k*-lân cận), không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu [10].

Để xác định khoảng cách giữa các điểm dữ liệu trong kNN, ta có thể sử dụng 2 công thức chính cơ bản:

- Euclidean:

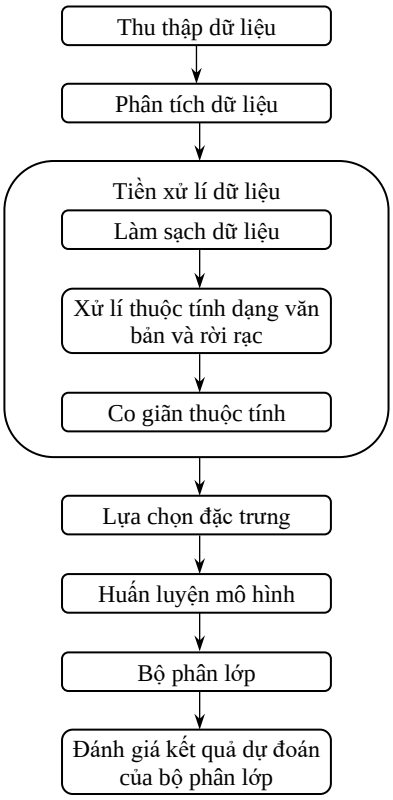
$$d_{Euclidean}(x,y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \quad (12)$$

- Manhattan:

$$d_{Manhattan}(x,y) = \sum_{i=1}^{n}|x_i - y_i| \quad (13)$$

Để tránh over/underfitting trong kNN, ta có thể thực hiện thay đổi công thức tính khoảng cách, tham số *k*,…

## IV. QUY TRÌNH XÂY DỰNG BỘ PHÂN LỚP




Hình 1: Quy trình xây dựng một bộ phân lớp thời tiết

Dưới đây là trình bày toàn bộ quy trình thực hiện đồ án của nhóm sau khi đã thu thập dữ liệu (phần thu thập dữ liệu và phân tích dữ liệu đã trình bày trong phần II).

### *A. Tiền xử lí dữ liệu*

*1) Làm sạch dữ liệu:* Trong bộ dữ liệu của nhóm thu thập được, có 8 mẫu chứa giá trị rỗng. Vì số lượng rất nhỏ (0.09%) nên nhóm quyết định sẽ bỏ đi những mẫu chứa giá trị rỗng.

*2) Xử lí thuộc tính dạng văn bản và rời rạc:* Hầu hết thuật toán máy học chỉ làm việc tốt với thuộc tính dạng số, do đó phải chuyển chuỗi thành số. Trong bộ dữ liệu có 2 thuộc tính dạng văn bản là Date time và Conditions. Nhóm sử dụng `OrdinalEncoder()` trong scikit-learn để chuyển giá trị của Conditions sang dạng số. Còn với thuộc tính Date time, thì nhóm tách thuộc tính này ra thành các thuộc tính Hour, Day, Month, Year.

*3) Co giãn thuộc tính:* Các thuật toán máy học thường không hoạt động tốt khi thuộc tính có khoảng giá trị quá khác biệt. Nhóm sử dụng phương pháp chuẩn hóa để co giãn thuộc tính. Ý tưởng của phương pháp này là đầu tiên trừ giá trị thuộc tính cho giá trị trung bình, sau đó chia cho độ lệch chuẩn để có phân phối có phương sai đơn vị. Nhóm sử dụng `StandardScaler()` của scikit-learn để hiện thực hóa phương pháp này trên code.

### *B. Lựa chọn đặc trưng*

BẢNG 2. HỆ SỐ TƯƠNG QUAN GIỮA CÁC THUỘC TÍNH CÒN LẠI VỚI THUỘC TÍNH CONDITIONS

| Thuộc tính | Hệ số tương quan |
|---|---|
| Hour | –0.029 |
| Day | –0.0022 |
| Month | –0.086 |
| Year | 0 |
| Temperature | –0.13 |
| Wind Speed | –0.044 |
| Visibility | 0.012 |
| Cloud Cover | –0.36 |
| Relative Humidity | 0.17 |

Dựa vào bảng 2, nhóm không sử dụng thuộc tính Year làm đặc trưng đưa vào huấn luyện mô hình (do dữ liệu chỉ thu thập tại 1 năm duy nhất). Những thuộc tính còn lại được chọn để làm đặc trưng đưa vào huấn luyện mô hình.

### *C. Huấn luyện mô hình*

Sau khi tiền xử lí dữ liệu, nhóm quyết định chọn ra 5 thuật toán để huấn luyện mô hình: Decision Tree, Random Forest, Logistic Regression, SVM và K-nearest Neighbor.

Nhóm sử dụng các lớp có sẵn trong scikit-learn lần lượt là `DecisionTreeClassifier`, `RandomForestClassifier`, `LogisticRegression`, `SVC` và `KneighborsClassifier` để huấn luyện mô hình.

Với mỗi mô hình máy học có những siêu tham số riêng. Nhóm đã sử dụng `GridSearchCV()` của scikit-learn để đi tìm siêu tham số tốt nhất cho mô hình. Với phương pháp grid search, chương trình sẽ tự động thử tất cả các khả năng kết hợp, đánh giá theo cross validation.

Bảng 3 trình bày kết quả siêu tham số tốt nhất cho từng mô hình:

BẢNG 3. KẾT QUẢ TINH CHỈNH MÔ HÌNH SAU KHI DÙNG GRID SEARCH TRONG SCIKIT-LEARN

| Thuật toán | Tham số |
|---|---|
| Decision Tree | criterion = 'gini',<br>max_features = 7,<br>min_samples_leaf = 1,<br>min_samples_split = 2 |
| Random Forest | criterion = 'entropy',<br>max_features = 8,<br>min_samples_leaf = 1,<br>min_samples_split = 5,<br>n_estimators = 10 |
| Logistic Regression | C = 10000,<br>solver = 'newton-cg' |
| SVM | kernel = 'linear',<br>C = 100,<br>decision_function_shape = 'ovo', |
| K-nearest Neighbor | leaf_size = 5,<br>n_neighbors = 2,<br>p = 2,<br>weights = 'distance' |

## V. KẾT QUẢ

Vì bộ dữ liệu này không cân bằng, nên nhóm sẽ không dùng độ đo chính xác để đánh giá mô hình.

Công thức tính độ đo precision, recall đối với bài toán phân lớp nhị phân:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (14)$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (15)$$

Với bài toán phân lớp đa lớp, nhóm quyết định sử dụng độ đo macro-average và weighted-average trên precision, recall và F1 score được cho bởi các công thức dưới đây [11]. Trong đó, $c_i$ là số lượng lớp thứ $i$.

$$Precision_{macro} = \frac{\sum_{i=1}^{n} Precision_i}{n} \quad (16)$$

$$Recall_{macro} = \frac{\sum_{i=1}^{n} Recall_i}{n} \quad (17)$$

$$F1score = \frac{2.Precision_{macro}.Recall_{macro}}{Precision_{macro} + Recall_{macro}} \quad (18)$$

$$Precision_{weighted} = \frac{\sum_{i=1}^{n}(Precision_i.c_i)}{\sum_{i=1}^{n} c_i} \quad (19)$$

$$Recall_{weighted} = \frac{\sum_{i=1}^{n}(Recall_i.c_i)}{\sum_{i=1}^{n} c_i} \quad (20)$$

$$F1score_{weighted} = \frac{\sum_{i=1}^{n}(F1score_i.c_i)}{\sum_{i=1}^{n} c_i} \quad (21)$$

Bảng 4 trình bày kết quả của mô hình sau khi đã tinh chỉnh tham số:

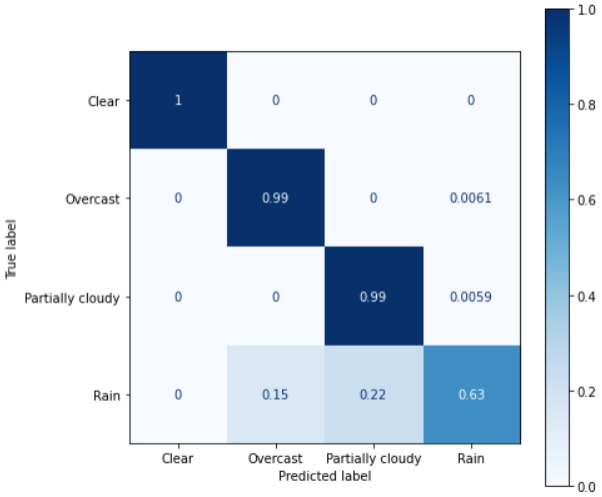
BẢNG 4. KẾT QUẢ ĐÁNH GIÁ MÔ HÌNH SAU KHI ĐÃ TINH CHỈNH THAM SỐ

| Mô hình | Macro-average | | | Weighted-average | | |
|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 |
| Decision Tree | 87.12 | 90.52 | 88.58 | 98.05 | 97.72 | 97.86 |
| Random Forest | 92.43 | 90.55 | 91.42 | 98.50 | 98.58 | 98.53 |
| Logistic Regression | 73.03 | 74.98 | 73.99 | 95.33 | 97.61 | 96.46 |
| SVM | 72.97 | 73.85 | 73.39 | 95.22 | 97.49 | 96.34 |
| K-nearest Neighbor | 69.92 | 67.22 | 68.39 | 93.58 | 93.51 | 93.52 |

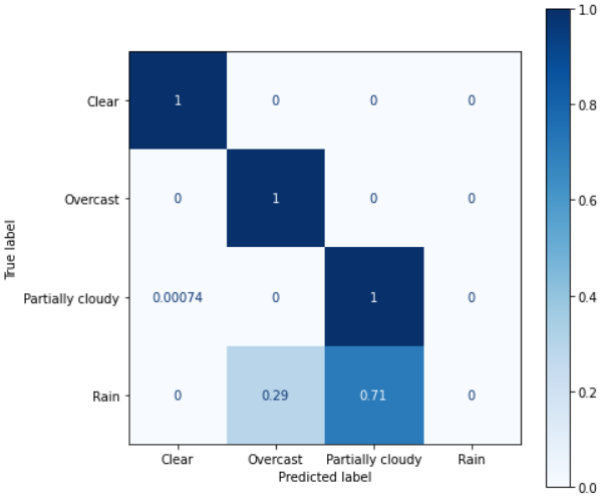
Như trên bảng, ta nhận thấy thuật toán Random Forest cho kết quả cao nhất. Tiếp theo đó là Decision Tree. Logistic Regression và SVM cho kết quả tương tự nhau. K-nearest Neighbor cho kết quả thấp nhất.

Đô đo macro-average và độ đo weighted-average khá chênh lệch chứng tỏ mô hình vẫn còn cho kết quả không đúng nhiều ở các lớp thiểu số

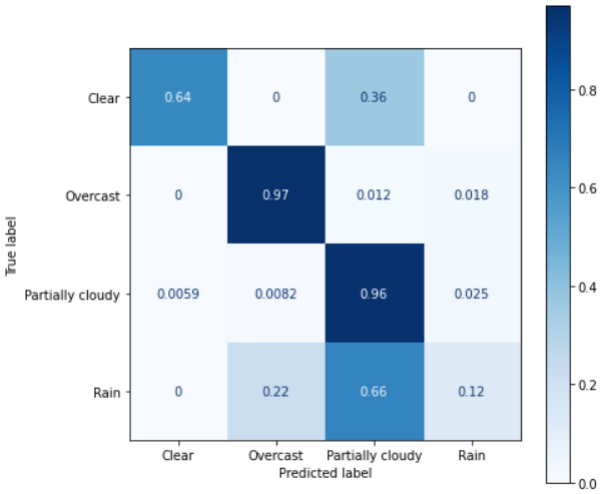
## VI. PHÂN TÍCH LỖI, HƯỚNG PHÁT TRIỂN




Hình 2: Confusion matrix của mô hình Random Forest (mô hình có kết quả cao nhất)




Hình 3: Confusion matrix của mô hình Logistic Regression (mô hình có kết quả tạm chấp nhận được)




Hình 4: Confusion matrix của mô hình kNN
(mô hình có kết quả thấp nhất)

Dựa vào kết quả confusion matrix ở 3 hình trên, ta nhận thấy nhãn rain được dự đoán sai khá nhiều. Dựa vào đặc điểm của dữ liệu thì có thể thấy nguyên nhân cho việc đoán sai này là do số mẫu mà có nhãn là rain quá ít. Do đó, một số thuật toán máy học không đem lại kết quả cao (Logistic Regression, SVM và K-nearest Neighbor). Vì vậy, ta cần có phương pháp lấy mẫu phù hợp cho dữ liệu bài toán trước khi đưa vào huấn luyện mô hình theo tỉ lệ thích hợp: có thể tăng số dữ liệu (oversampling) số dữ liệu thuộc lớp thiểu số hoặc giảm số dữ liệu (undersampling) số dữ liệu thuộc lớp đa số hoặc kết hợp cả hai.

Vì hiện tại nhóm lấy gần hết thuộc tính trong bộ dữ liệu (trừ thuộc tính phân loại) để xây dựng mô hình nên ta cần đó phân tích sâu hơn về bộ dữ liệu và kiểm tra thực nghiệm để có thể lựa chọn bộ thuộc tính phù hợp để xây dựng mô hình.

Mô hình mà ta xây dựng dựa hoàn toàn vào một thành phố cụ thể (thành phố Hồ Chí Minh) có khí hậu nóng ẩm đặc trưng, do đó mô hình hiện tại chỉ phù hợp để dự đoán cho các thành phố có khí hậu tương tự thành phố Hồ Chí Minh mà không nên sử dụng cho các thành phố có khí hậu lạnh hơn khác.

Ngoài ra, vì thời gian hạn chế nên nhóm chỉ thực hiện huấn luyện mô hình trên các thuật toán batch learning. Dữ liệu thời tiết là dữ liệu thời gian thực nên cần sử dụng các thuật toán online learning để có thể cập nhật nhanh model khi có thêm mẫu dữ liệu mới.

Trong tương lai, nhóm có thể tiếp tục mở rộng bộ dữ liệu sang các năm trước đó để tiếp tục hoàn thành bộ dữ liệu và tinh chỉnh lại mô hình thuật toán phù hợp với bộ dữ liệu mở rộng đó.

## VII. KẾT LUẬN

Nhóm đã thu thập dữ liệu thời tiết của thành phố Hồ Chí Minh năm 2020 và huấn luyện được mô hình để dự đoán thời tiết dựa trên các thông số về thời gian, nhiệt độ, độ ẩm, v.v… Mô hình mà nhóm đã xây dựng đã có tính chính xác cao khi sử dụng thuật toán Decision Tree và Random Forest. Từ đó, nhóm có triển vọng để tiếp tục khắc phục các lỗi đã nêu ra, tiếp tục xây dựng hoàn thiện ứng dụng mà nhóm muốn hướng đến trong tương lai.

## TÀI LIỆU THAM KHẢO


[1] "Weather Forecast & Weather History Data", 2020. [Trực tuyến]. Địa chỉ: https://www.visualcrossing.com/weather-data [Truy cập 26/12/2020].

[2] "Tầm nhìn xa", 2018. [Trực tuyến]. Địa chỉ: https://vi.wikipedia.org/wiki/T%E1%BA%A7m_nh%C3%ACn_xa [Truy cập 28/12/2020].

[3] "Mây che phủ", 2020. [Trực tuyến]. Địa chỉ: https://vi.wikipedia.org/wiki/M%C3%A2y_che_ph%E1%BB%A7 [Truy cập 28/12/2020].

[4] "Độ ẩm tương đối", 2020. [Trực tuyến]. Địa chỉ: https://vi.wikipedia.org/wiki/%C4%90%E1%BB%99_%E1%BA%A9m_t%C6%B0%C6%A1ng_%C4%91%E1%BB%91i [Truy cập 28/12/2020].

[5] "THUẬT TOÁN CÂY QUYẾT ĐỊNH (P.2): CART (GINI INDEX)", 2019. [Trực tuyến]. Địa chỉ: https://bigdatauni.com/vi/tin-tuc/thuat-toan-cay-quyet-dinh-p-2-cart-gini-index.html [Truy cập 29/12/2020]

[6] "THUẬT TOÁN CÂY QUYẾT ĐỊNH (P.3): C4.5 (ENTROPY)", 2019. [Trực tuyến]. Địa chỉ: https://bigdatauni.com/vi/tin-tuc/thuat-toan-cay-quyet-dinh-p-3-c4-5-entropy.html [Truy cập 29/12/2020]

[7] "RANDOM FOREST, THẾ NÀO LÀ MỘT RỪNG NGẪU NHIÊN", 2018. [Trực tuyến]. Địa chỉ: https://couhpcode.wordpress.com/2018/01/24/random-forest-the-nao-la-mot-rung-ngau-nhien/ [Truy cập 29/12/2020]

[8] Aurélien Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow – Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd edition, O'Reilly Media, 2012.

[9] Vũ Hữu Tiệp, "Bài 12: Binary Classifiers cho các bài toán Classification", 2017. [Trực tuyến]. Địa chỉ: https://machinelearningcoban.com/2017/02/11/binaryclassifiers/ [Truy cập 29/12/2020]

[10] Vũ Hữu Tiệp, "Bài 6: K-nearest neighbors:, 2017. [Trực tuyến]. Địa chỉ: https://machinelearningcoban.com/2017/01/08/knn/ [Truy cập 29/12/2020]

[11] Ramit Pahwa, "Micro-Macro Precision, Recall and F-Score", 2015. [Trực tuyến]. Địa chỉ:https://medium.com/@ramit.singh.pahwa/micro-macro-precision-recall-and-f-score-44439de1a044 [Truy cập 30/12/2020].