

BÁO CÁO TRÍ TUỆ NHÂN TẠO

TÌM HIỂU VỀ RANDOM FOREST VÀ ÁP DỤNG VÀO BÀI TOÁN DỰ ĐOÁN CỤ THỂ

Ngành: **CÔNG NGHỆ THÔNG TIN**

Nhóm sinh viên thực hiện:

Tên	Lớp	MSSV
Nguyễn Thanh Siêu	21DTHE4	2180609116
Trần Việt Đức	21DTHE4	2180606946
Đặng Lê Ngọc Trạng	21DTHE4	2180608126

Giảng viên hướng dẫn: **TS. Vũ Thanh Hiền**

TP. Hồ Chí Minh, 2023

LỜI MỞ ĐẦU

Trong những năm gần đây, các lĩnh vực nghiên cứu ngành Khoa Học Máy Tính phát triển hết sức mạnh mẽ, nhiều thuật toán ra đời với nhiều hướng nghiên cứu khác nhau. Trong đó Machine Learning là một hướng nghiên cứu đã xuất hiện từ những năm 1959-1960 và đạt được nhiều thành tựu. Nhiều thuật toán học máy được ứng dụng trong thực tế và mang lại hiệu quả trong một số lĩnh vực như: nhận diện hình ảnh/ khuôn mặt, tự động nhận diện giọng nói, chữ viết, lĩnh vực tài chính – ngân hàng, chăm sóc sức khỏe,... Các thuật toán được dùng phổ biến như: Decision tree, mạng Nơ-ron nhân tạo, K-Mean, Random Forest,... Mỗi thuật toán đều có một số tham số và các tham số này ảnh hưởng rất lớn đến kết quả của thuật toán, vì vậy việc tối ưu các tham số là vô cùng cần thiết.

Trong đồ án này, mục tiêu của nhóm chúng tôi là tìm hiểu về Random Forest và áp dụng vào bài toán cụ thể “Bài toán dự đoán khả năng mắc bệnh tim mạch”. Đồ án tập trung về lịch sử, cơ sở lý thuyết, tham số và siêu tham số trong Random Forest và cách điều chỉnh nó, ưu và nhược điểm của thuật toán Random Forest, cùng một số ứng dụng thực tế của thuật toán Random Forest và cuối cùng là giải thuật của bài toán liên quan về Random Forest – bài toán dự đoán khả năng mắc bệnh tim mạch.

Tất cả những tham khảo từ các nghiên cứu liên quan đều được nêu nguồn gốc rõ ràng từ danh mục tài liệu tham khảo của đồ án. Trong đồ án, không có việc sao chép tài liệu, công trình nghiên cứu của người khác mà không chỉ rõ về tài liệu tham khảo. Mọi sao chép không hợp lệ chúng tôi xin cam đoan sẽ chịu toàn bộ trách nhiệm.

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin chân thành cảm ơn TS.Vũ Thanh Hiền hướng dẫn đồ án môn học cho chúng em.

Sau thời gian học tập và tìm hiểu dưới sự hướng dẫn tận tình của thầy, chúng em đã rút ra được nhiều kinh nghiệm quý báu về khả năng phân tích và thiết kế cũng như kỹ năng làm việc mà không chỉ đơn giản là đọc trong sách, vở có thể có được và một lần nữa chúng em xin gửi lời cảm ơn sâu sắc đến Thầy đã dạy bảo và hướng dẫn những kiến thức chuyên môn cần có để chúng em áp dụng tốt nhất những gì đã được học trong suốt thời gian qua.

Trong quá trình thực hiện và làm báo cáo, do còn thiếu nhiều kinh nghiệm thực tế không tránh được những sai sót. Chúng em rất mong nhận được những ý kiến đóng góp của Thầy để giúp đỡ chúng em trong kinh vững này được hoàn thiện hơn. Đó là hành trang quý giá giúp chúng em hoàn thiện kiến thức của mình trong chặng đường sau này.

Chúng em xin chân thành cảm ơn và trân trọng kính chào!

NHẬN XÉT & ĐÁNH GIÁ CỦA GIẢNG VIÊN HƯỚNG DẪN

Qua quá trình học tập:

Giáo viên hướng dẫn có một số nhận xét, đánh giá như sau:

1/- Quá trình học tập

.....

.....

.....

.....

2/- Thực hiện báo cáo

.....

.....

.....

.....

Đánh giá chung

.....

.....

.....

Kết quả đạt được:

Điểm đánh giá việc thực hiện báo cáo...../10

TP. HCM, Ngày tháng năm 20

GIẢNG VIÊN HƯỚNG DẪN

TS. Vũ Thanh Hiền

MỤC LỤC

LỜI MỞ ĐẦU	2
LỜI CẢM ƠN	3
NHẬN XÉT & ĐÁNH GIÁ CỦA GIẢNG VIÊN HƯỚNG DẪN	4
MỤC LỤC	5
DANH MỤC HÌNH ẢNH	7
CHƯƠNG 1: TỔNG QUAN	10
1.1 Giới thiệu về Machine Learning	10
1.2 Giới thiệu về Classification	11
1.3 Giới thiệu sơ lược Decision Tree	12
1.4 Giới thiệu về thuật toán Random Forest	13
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	15
2.1 Decision Tree	15
2.1.1 Định nghĩa	15
2.1.2 Một số thuật toán xây dựng Decision Tree	15
2.1.3 Đặc điểm của Decision Tree	20
2.2 Thuật Toán Random Forest	21
2.2.1 Định nghĩa	21
2.2.2 Mô hình thuật toán	21
2.2.3 Thuật toán	23
2.2.4 Đặc điểm của Random Forest	23
2.2.5 Ứng dụng của thuật toán Random Forest	25

2.3	Phương pháp lấy mẫu Bootstrap.....	25
2.3.1	Giới thiệu Bootstrap.....	25
2.3.2	Thuật toán Bootstrap.....	26
2.3.3	Đặc điểm của Bootstrap.....	28
2.3.4	Bootstrap trong Random Forest.....	28
2.4	Tham số và siêu tham số trong model Random Forest	28
2.4.1	Tham số (Parameter)	29
2.4.2	Siêu tham số (Hyperparameter)	29
CHƯƠNG 3: BÀI TOÁN DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH TIM MẠCH		
	40
3.1	Giới thiệu bài toán	41
3.2	Tập dữ liệu.....	42
3.3	Tiền xử lý dữ liệu	45
3.4	Tiến hành xây dựng các mô hình	49
3.5	Nghiệm thu kết quả.....	50
3.6	Sử dụng mô hình đối với tập test và đánh giá kết quả	55
CHƯƠNG 4: KẾT LUẬN.....		60
TÀI LIỆU THAM KHẢO		61

DANH MỤC HÌNH ẢNH

Hình 1.1 Mô hình học máy.....	11
Hình 1.2 Phân lớp nhận dạng email spam.....	12
Hình 1.3 Ảnh minh họa Decision Tree.....	13
Hình 1.4 Hình ảnh giới thiệu về thuật toán Random Forest.....	14
Hình 2.1 Mô hình Decision Tree quyết định đi chơi.....	15
Hình 2.2 Bảng quan sát sự thành công của một bộ phim.....	16
Hình 2.3 Mô hình thể hiện kết quả thành công của bộ phim	17
Hình 2.4 Hình vẽ biểu diễn sự thay đổi của hàm Entropy	18
Hình 2.5 Mô hình thuật toán Random Forest.....	22
Hình 2.6 Xây dựng thuật toán Random Forest.....	23
Hình 2.7 Hình ảnh về tập dữ liệu	27
Hình 2.8 Bootstrap trong Random Forest	28
Hình 2.9 Sơ đồ max_depth.....	30
Hình 2.10 Sơ đồ hyperparameter value.....	30
Hình 2.11 Sơ đồ min_sample_split	31
Hình 2.12 Sơ đồ min_sample_split hyperparameter value	32
Hình 2.13 Sơ đồ max_leaf_nodes	33
Hình 2.14 Sơ đồ max_leaf_nodes hyperparameter value.....	34
Hình 2.15 Sơ đồ min_sample_leaf.....	35
Hình 2.16 Sơ đồ min_samples_leaf parameter value.....	36
Hình 2.17 Mô hình n_estimators và hyper parameter để thấy rõ hiệu suất và mức độ trì trệ khi tăng số lượng cây lên	37
Hình 2.18 Mô hình max_samples.....	38
Hình 2.19 Sự thay đổi của mô hình max_features ảnh hưởng đến hiệu suất mô hình Random Forest.....	39

Hình 3. 1 Hình người đang có nguy cơ mắc bệnh tim	40
Hình 3. 2 Dòng dữ liệu từ Heart.csv	42
Hình 3. 3 Số dòng số cột và kiểm tra dữ liệu có bị missing value	44
Hình 3. 4 Thông tin các cột và các thuộc tính của cột dữ liệu	45
Hình 3. 5 Thống kê cơ bản các thuộc tính.....	45
Hình 3. 6 Biểu đồ trực quan các dữ liệu.....	46
Hình 3. 7 Biểu đồ trực quan dữ liệu về rối loạn về máu và có mắc bệnh tim hay không	46
Hình 3. 8 Trực quan độ tương quan giữa các cột.....	47
Hình 3. 9 Dòng kết quả khi hợp nhất dữ liệu	48
Hình 3. 10 Trích xuất các đặc trưng bằng cách sử dụng kỹ thuật feature selection	49
Hình 3. 11 Train model cho tập dữ liệu.....	50
Hình 3. 12 Nghiệm thu kết quả các mô hình trên tập train	51
Hình 3. 13 Biểu đồ heatmap của logitis	51
Hình 3. 14 Độ chính xác của logitis	52
Hình 3. 15 Model Decision Tree thông số kỹ thuật	52
Hình 3. 16 Độ chính xác của Decision Tree.....	52
Hình 3. 17 Model Random và thông số kỹ thuật	53
Hình 3. 18 Confuse Matrix của Random model.....	53
Hình 3. 19 Độ chính xác của Random Forest.....	53
Hình 3. 20 Model KNN thông số kỹ thuật	54
Hình 3. 21 Confuse matrix của KNN	54
Hình 3. 22 Độ chính xác của KNeighbors.....	54
Hình 3. 23 Model Logitis thông số kỹ thuật trên tập test.....	55
Hình 3. 24 Confuse matrix của logitis model trên tập test	55
Hình 3. 25 Độ chính xác của logitis	56
Hình 3. 26 Model decision tree thông số kỹ thuật trên tập test.....	56
Hình 3. 27 Confuse matrix của Decision Tree trên tập test	56
Hình 3. 28 Độ chính xác của Decision Tree.....	57

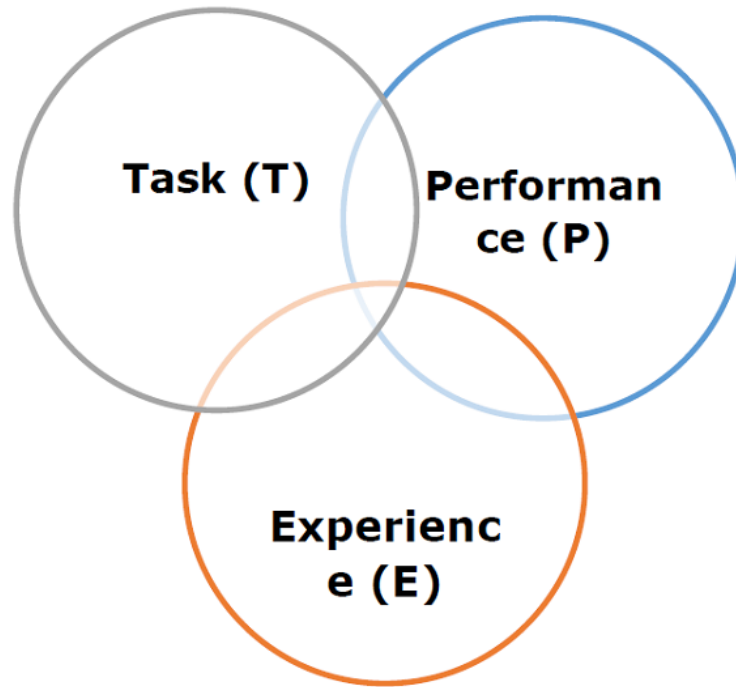
Hình 3. 29 Model Random thông số kỹ thuật trên tập test	57
Hình 3. 30 Confuse Matrix của Random Model trên tập test	57
Hình 3. 31 Độ chính xác của Random Forest trên tập test.....	58
Hình 3. 32 Model KNN thông số kỹ thuật trên tập test	58
Hình 3. 33 Confuse Matrix của KNN trên tập test.....	58
Hình 3. 34 Độ chính xác của kneight	59

CHƯƠNG 1: TỔNG QUAN

1.1 Giới thiệu về Machine Learning

Chúng ta đang sống trong “thời đại dữ liệu” được làm giàu với sức mạnh tính toán tốt hơn với nhiều tài nguyên lưu trữ hơn. Dữ liệu thông tin này đang tăng lên từng ngày, nhưng thách thức thực sự là hiểu được tất cả dữ liệu. Các doanh nghiệp và tổ chức đang cố gắng giải quyết vấn đề này bằng cách xây dựng các hệ thống thông minh sử dụng các khái niệm và phương pháp luận từ Khoa học dữ liệu, Khai thác dữ liệu và Máy học.

Machine Learning là ngành khoa học máy tính nhằm nghiên cứu, phát triển các thuật toán, thuật giải với các mục đích đưa tri thức vào máy tính, cụ thể hơn là những thuật giải dựa trên các tập dữ liệu và rút ra các quy luật từ chúng, làm cho máy tính có thể giải được các bài toán mà các thuật toán bình thường khó có thể thực hiện như tìm kiếm, nhận dạng, dự đoán. Các thuật toán học máy được phân loại theo kết quả thuật toán. Các loại thuật toán thường được sử dụng như: học có giám sát (supervised learning), học không giám sát (unsupervised learning), học bán giám sát (semi-supervised learning), học tăng cường (reinforcement learning), học tiến hóa (evolutionary learning),... Machine Learning đang ngày càng phát triển, ngoài những thuật toán ra đời sớm thì sau này có nhiều thuật toán ra đời như: Mạng nơ-ron nhân tạo, Mạng Bayes, Học bằng quy trình Markov, Decision tree, K-Mean, SVM, Random Forest, Học luật bằng quy nạp,...



Hình 1.1 Mô hình học máy

Machine Learning hiện nay được áp dụng rộng rãi và trong nhiều lĩnh vực khác nhau bao gồm máy truy tìm dữ liệu, chuẩn đoán y khoa, phát hiện thẻ tín dụng giả, phân tích thị trường chứng khoán, phân loại chuỗi DNA, nhận dạng tiếng nói và chữ viết, dịch tự động, trò chơi điện tử, cử động rô-bốt (robot locomotion).

Thuật toán Random Forest là một trong những thuật toán Machine Learning ra đời muộn hơn các thuật toán Machine Learning khác, chính vì vậy thuật toán Random Forest kế thừa được ưu điểm của các thuật toán khác đồng thời khắc phục được những hạn chế về mặt số lượng dữ liệu cũng như độ phức tạp của dữ liệu.

1.2 Giới thiệu về Classification

Phân loại (Classification) trong học máy và thống kê là một trong phương pháp học có giám sát (supervised learning), trong đó chương trình máy tính học từ dữ liệu cung cấp cho nó và thực hiện các quan sát hoặc phân loại mới. Nó là quá trình phân loại một tập hợp dữ liệu nhất định thành các lớp. Nó có thể được thực hiện trên cả dữ

liệu có cấu trúc và không có cấu trúc. Quá trình bắt đầu với việc dự đoán class/category của các điểm dữ liệu đã cho. Các lớp thường được gọi là target, lable hoặc categories.

Các vấn đề phân loại phổ biến nhất là: Nhận dạng giọng nói, nhận diện khuôn mặt, nhận dạng chữ viết tay, phát hiện email spam, phân loại tài liệu,... Nó có thể là bài toán phân loại nhị phân hoặc bài toán nhiều lớp.



Hình 1.2 Phân lớp nhận dạng email spam

Có một loạt các thuật toán học máy để phân loại trong trong học máy: Naive Bayes Classifier, Stochastic Gradient Descent, K-Nearest Neighbor, Decision Tree, Random forest,... Mỗi loại thuật toán sẽ có ưu nhược điểm riêng nhưng trong đồ án này chúng ta sẽ tập trung vào Random Forest sẽ được đề cập chương 2 và chương 3.

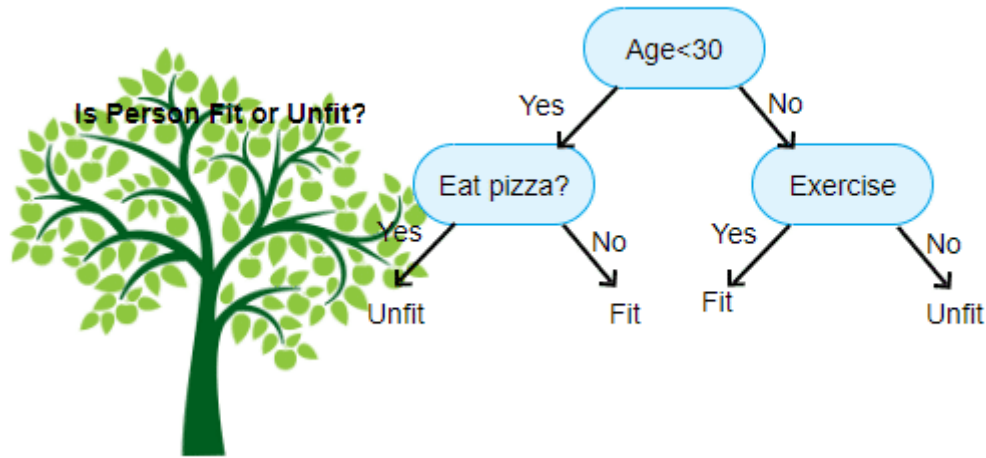
1.3 Giới thiệu sơ lược Decision Tree

Decision Tree là một trong những công cụ mạnh mẽ nhất của các thuật toán học tập có giám sát. Việc huấn luyện Decision Tree và sử dụng nó như một mô hình dự đoán, ánh xạ từ các quan sát về một sự vật/hiện tượng với các kết luận về giá trị mục tiêu của sự vật/hiện tượng.

Decision Tree có hai loại chính:

- Cây phân loại (Classification Tree): nếu y là một biến phân loại như: giới tính (nam hay nữ), kết quả một trận đấu (thắng hay thua).

- Cây hồi quy (Regression tree): ước lượng hàm đó có giá trị là số thực thay vì được sử dụng cho các nhiệm vụ phân loại. Ví dụ ước tính giá của một ngôi nhà hoặc khoảng thời gian một bệnh nhân nằm viện.

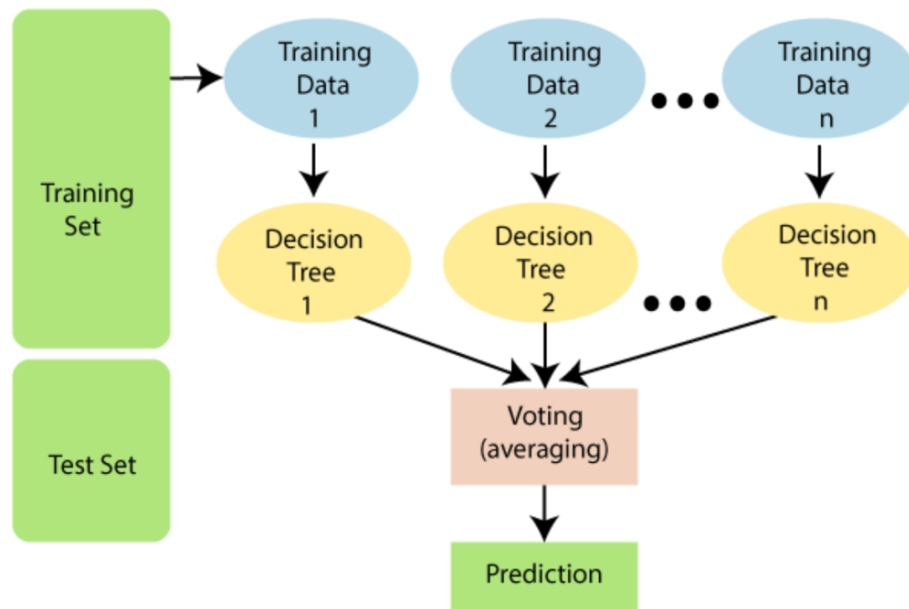


Hình 1.3 Ảnh minh họa Decision Tree

Ở chương 2, chúng ta sẽ xem thử Decision Tree có vai trò quan trọng như thế nào trong thuật toán Random Forest.

1.4 Giới thiệu về thuật toán Random Forest

Random Forest hay rừng ngẫu nhiên là một thuật toán học có giám sát được sử dụng cho cả phân loại cũng như hồi quy. Tuy nhiên, nó chủ yếu được sử dụng cho các bài toán phân loại. Như chúng ta đã biết một khu rừng được tạo thành từ cây cối và nhiều cây cối hơn có nghĩa là khu rừng vững chắc hơn. Tương tự, thuật toán Random Forest được tạo ra từ các cây quyết định trên các mẫu dữ liệu sau đó lấy dự đoán từ mỗi người trong số chúng và cuối cùng chọn giải pháp tốt nhất bằng cách bỏ phiếu. Đây là một phương pháp tổng hợp tốt hơn một cây quyết định đơn lẻ vì nó giảm sự phù hợp quá mức bằng cách lấy trung bình kết quả.



Hình 1.4 Hình ảnh giới thiệu về thuật toán Random Forest

Thuật toán đầu tiên cho các khu rừng ngẫu nhiên được tạo ra bởi Tin Kam Ho bằng cách sử dụng phương pháp không gian con ngẫu nhiên, theo công thức của Ho là một cách để thực hiện phương pháp “phân biệt ngẫu nhiên” để phân loại do Eugene Kleinberg đề xuất. Một phần mở rộng của thuật toán là được phát triển bởi Leo Breiman và Adele Cutler, người đã đăng kí “Random Forest” làm nhãn hiệu (kể từ năm 2019, thuộc sở hữu của Minitab, Inc). Phần mở rộng ý tưởng “bagging (đóng gói)” của Breiman và lựa chọn ngẫu nhiên các tính năng được Ho giới thiệu lần đầu tiên và sau là độc lập bởi Amit và German để xây dựng một tập hợp các Decision Tree với phương sai có kiểm soát.

Random Forest được mô hình hóa như tập các cây phân lớp. Tuy nhiên Random Forest sử dụng các mẫu ngẫu nhiên cho các cây cũng như việc chọn lựa các thuộc tính ngẫu nhiên khi phân chia cây. Thuật toán Random Forest tỏ ra chính xác và nhanh hơn khi huấn luyện trên không gian dữ liệu lớn với nhiều thuộc tính, việc sử dụng kết quả dự đoán của tất cả các cây trong rừng khi phân lớp hoặc hồi quy giúp cho kết quả thuật toán chính xác hơn. Chúng ta sẽ tìm hiểu kỹ hơn về Random Forest ở chương 2.

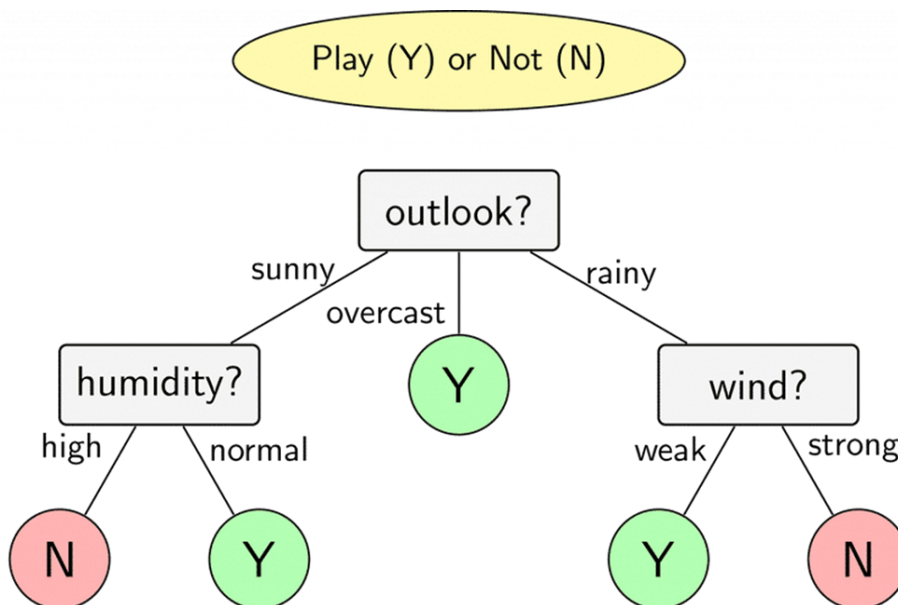
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Decision Tree

2.1.1 Định nghĩa

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.

Decision Tree là mô hình Supervised Learning, có thể được áp dụng vào cả hai bài toán Classification và Regression. Mỗi một nút trong (internal node) tương ứng với một biến, đường nối giữa nó với 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 biến được biểu diễn đường đi từ nút gốc tới nút lá đó. Kỹ thuật học máy dùng trong cây quyết định được gọi là học bằng cây quyết định, hay chỉ với cái tên ngắn gọn là cây quyết định.



Hình 2.1 Mô hình Decision Tree quyết định đi chơi

2.1.2 Một số thuật toán xây dựng Decision Tree

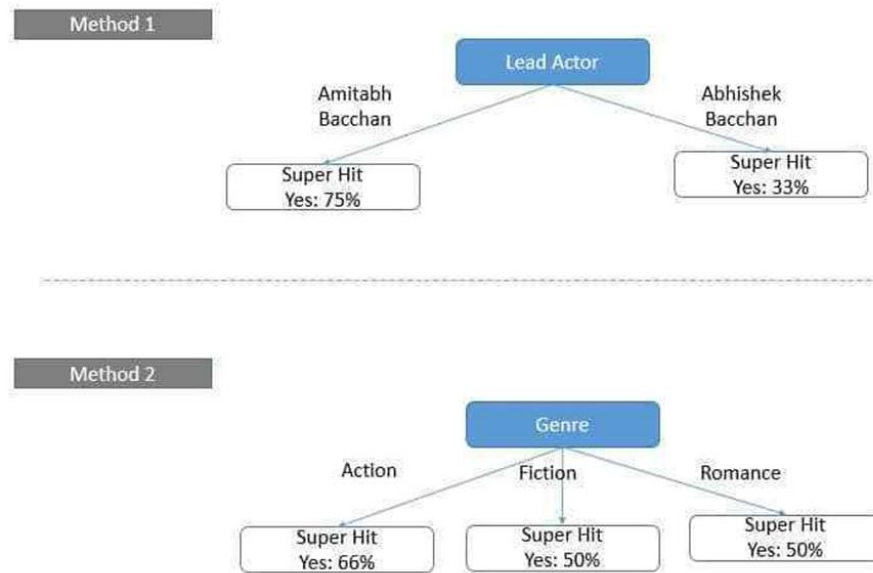
Thuật toán ID3 (J.R.Quinlan 1933) sử dụng phương pháp tham lam tìm kiếm từ trên xuống thông qua không gian của các nhánh có thể không có backtracking. ID3 sử dụng Entropy và Information Gain để xây dựng Decision Tree.

Ví dụ: Bạn muốn xem xét sự thành công của một bộ phim thông qua hai yếu tố: Diễn viên chính của phim và thể loại phim.

Lead Actor	Genre	Hit(Y/N)
Amitabh Bacchan	Action	Yes
Amitabh Bacchan	Fiction	Yes
Amitabh Bacchan	Romance	No
Amitabh Bacchan	Action	Yes
Abhishek Bacchan	Action	No
Abhishek Bacchan	Fiction	No
Abhishek Bacchan	Romance	Yes

Hình 2.2 Bảng quan sát sự thành công của một bộ phim

Giả sử muốn xác định độ thành công của một bộ phim chỉ dựa trên một yếu tố, sẽ có hai cách thực hiện như sau: qua diễn viên chính và qua thể loại phim.



Hình 2.3 Mô hình thể hiện kết quả thành công của bộ phim

Qua sơ đồ trên ta thấy rõ, với phương pháp thứ nhất, ta phân loại được rõ ràng, trong khi phương pháp hai có một kết quả lộn xộn hơn. Và tương tự, cây quyết định sẽ thực hiện như trên khi thực hiện việc chọn các biến.

Có rất nhiều hệ số khác nhau mà cây quyết định sử dụng phân chia. Dưới đây là hai hệ số phổ biến là Information Gain và Gain Ratio (ngoài ra còn hệ số Gini).

Entropy trong Decision Tree:

- Số lượng mong đợi các bit cần thiết để mã hóa thông tin về lớp của một thành viên rút ra một cách ngẫu nhiên từ tập S.
- Trong trường hợp tối ưu, mã có độ dài ngắn nhất.
- Trong lý thuyết thông tin, mã có độ dài tối ưu là mã gán $(-\log_2 p)$ bit cho thông điệp (message) có xác suất là p.
- Trong trường hợp S là tập mẫu dữ liệu thì thành viên của S là một mẫu dữ liệu, mỗi mẫu dữ liệu thuộc một lớp có một giá trị phân lớp.
- Entropy có giá trị nằm trong đoạn $[0,1]$:
 - o Entropy $(S) = 0 \Leftrightarrow$ tập mẫu S chỉ toàn mẫu thuộc cùng lớp hoặc S là thuần nhất.

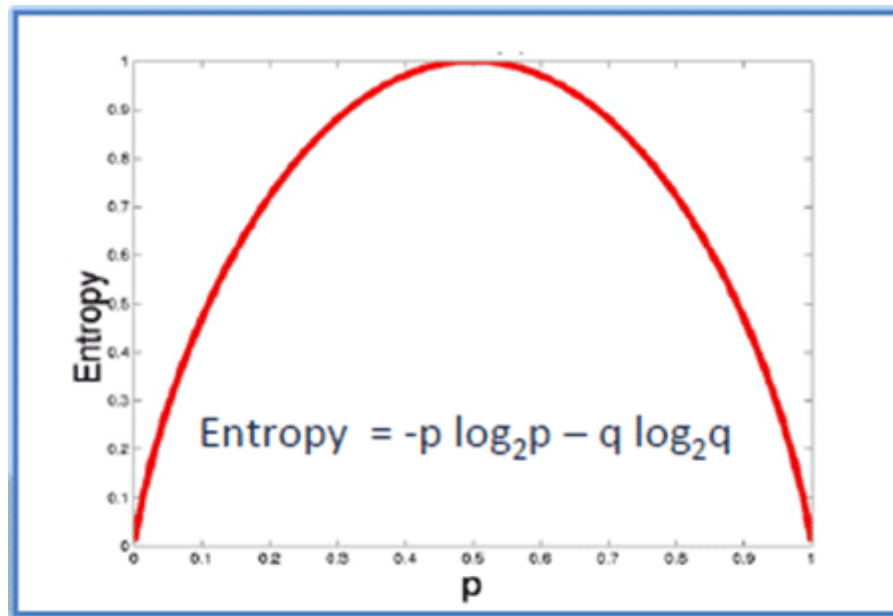
- $\text{Entropy}(S) = 1 \Leftrightarrow$ tập mẫu S có các mẫu thuộc các lớp khác nhau với độ pha trộn là cao nhất.
- $0 < \text{Entropy}(S) < 1 \Leftrightarrow$ tập mẫu S có số lượng mẫu thuộc các lớp khác nhau là không bằng nhau.

Entropy cần để phân lớp một mẫu trong CSDL D là:

$$\text{Info}(D) = - \sum_{i=1}^n (p_i \log_2 [(p)_i])$$

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

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



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

Hình 2.4 Hình vẽ 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: $p_i = 0$ hoặc $p_i = 1$
- P vẩn đục: $p_i = 0.5$, khi đó hàm Entropy đạt đỉnh cao nhất.

Information Gain trong Decision Tree:

- Thuộc tính A gồm các giá trị $\{a_1, a_2, \dots, a_v\}$, và phân chia D thành v tập con $\{D_1, D_2, \dots, D_v\}$. Thông tin cần thiết để chia D theo A.

$$Info_A(D) = \sum_{j=1}^v \left(a_n \frac{|D_j|}{|D|} \times [Info(D)]_j \right)$$

- Information Gain dựa trên phân chia theo thuộc tính A của tập D

$$Gain(A) = Info(D) - Info_A(D)$$

- Chọn thuộc tính với Information Gain lớn nhất là thuộc tính phân hạch
- Với ví dụ về thành công của một bộ phim trên, ta tính được hệ số Entropy như sau:

- Trong 7 mẫu có 4 mẫu “Yes” và 3 mẫu “No”

Lead Actor	Hit (Yes)	Hit(No)
Amitabh Bacchan	3	1
Abhishek Bacchan	1	2

→ $|D| = 7, n = 2, C_1 = \text{“Yes”}, C_2 = \text{“No”}$

→ $|C_{1,D}| = 4, |C_{2,D}| = 3$

→ Entropy để phân lớp một mẫu trong D là

$$Info(D) = -\frac{4}{7}\log_2\frac{4}{7} - \frac{3}{7}\log_2\frac{3}{7} \approx 0.99$$

- Tính Information Gain đối với “Lead Actor”

$$\rightarrow Info_{Lead Actor}(D) = \frac{4}{7} \left(-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} \right) +$$

$$\frac{3}{7} \left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} \right) \approx 0.69$$

$$\rightarrow Gain(Lead Actor) = Info(D) - Info_{Lead Actor}(D) \approx 0.99 - 0.69 \approx 0.3$$

- Tính Information Gain đối với “Genre”

Genre	Hit (Yes)	Hit(No)
Action	2	1
Fiction	1	1
Romance	1	1

$$\rightarrow Info_{Genre}(D) = \frac{3}{7} \left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} \right) +$$

$$2/7 \left(-\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} \right) +$$

$$2/7 \left(-\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} \right) \approx 0.55$$

$$\rightarrow \text{Gain}_{\text{Genre}}(D) = \text{Info}(D) - \text{Info}_{\text{Genre}} = 0.99 - 0.55 \approx 0.44$$

→ Chọn “Genre” làm thuộc tính phân hoạch và được cây quyết định

C4.5 là thuật toán cải tiến ID3:

- Độ đo Information Gain có xu hướng thiên vị cho các thuộc tính có nhiều giá trị:
 - Một số trường hợp các mẫu chia thuần nhất và không có ích cho việc phân lớp.
 - Cần chuẩn hóa độ đo Information Gain.
- C4.5 sử dụng độ đo Gain Ratio để khắc phục vấn đề của ID3.
- $\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}_A(D)}$ với

$$\text{SplitInfo}_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right) \text{ chọn thuộc tính với độ đo Gain}$$

Ratio lớn nhất là thuộc tính phân hoạch áp dụng với ví dụ trên ta có

$$\rightarrow \text{SplitInfo}_{\text{Genre}}(D) = -\frac{3}{7} \log_2 \left(\frac{3}{7} \right) - \frac{2}{7} \log_2 \left(\frac{2}{7} \right) - \frac{2}{7} \log_2 \left(\frac{2}{7} \right)$$

$$\text{ainRatio(Lead Actor)} = \frac{0.3}{0.52} \approx 0.58$$

2.1.3 Đặc điểm của Decision Tree

Ưu điểm	Nhược điểm
<p>Cây quyết định là một thuật toán đơn giản và phổ biến. Thuật toán này được sử dụng rộng rãi bởi những lợi ích của nó:</p> <ul style="list-style-type: none"> - Mô hình sinh ra các quy tắc dễ hiểu cho người đọc tạo ra 	<p>Kèm với đó, cây quyết định cũng có những nhược điểm:</p> <ul style="list-style-type: none"> - Mô hình cây quyết định phụ thuộc rất lớn vào dữ liệu của bạn. Thậm chí

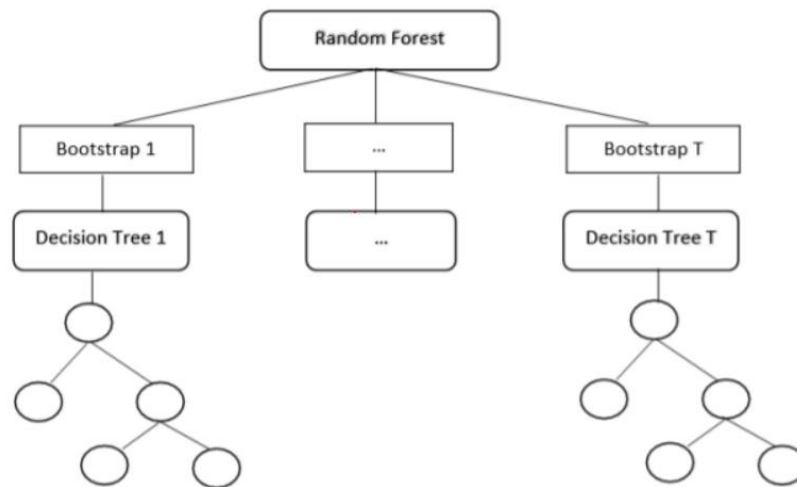
<p>bộ luật với mỗi nhánh lá là một luật của cây.</p> <ul style="list-style-type: none"> - Dữ liệu đầu vào có thể là dữ liệu missing, không cần chuẩn hóa hoặc biến tạo giả. - Có thể làm việc với cả dữ liệu số và dữ liệu phân loại. - Có thể xác thực mô hình bằng cách sử dụng kiểm tra thống kê. - Có khả năng là việc với dữ liệu lớn. 	<p>với một sự thay đổi nhỏ trong toàn bộ dữ liệu, cấu trúc mô hình cây quyết định thay đổi hoàn toàn.</p> <ul style="list-style-type: none"> - Cây quyết định hay gặp vấn đề overfitting.
---	--

2.2 Thuật Toán Random Forest

2.2.1 Định nghĩa

Random Forests hay Random Decision Forests là thuật toán học máy dựa trên kỹ thuật lắp ghép, kết hợp với các cây phân lớp. Random Forest xây dựng cây phân lớp bằng cách lựa chọn ngẫu nhiên một nhóm nhỏ các thuộc tính tại mỗi nút của cây để phân chia cho mức tiếp theo của cây phân lớp. Ngoài ra tập mẫu của mỗi cây cũng được lựa chọn ngẫu nhiên bằng phương pháp Bootstrap từ tập mẫu ban đầu. Số lượng các cây phân lớp trong rừng là không hạn chế và thuật toán sử dụng kết quả dự đoán của tất cả cây trong rừng làm kết quả cuối cùng của thuật toán.

2.2.2 Mô hình thuật toán



Hình 2.5 Mô hình thuật toán Random Forest

Các kí hiệu:

- Random Forest Rừng ngẫu nhiên: Tập cây phân lớp
- Bootstrap i Phương pháp Bootstrap tạo tập huấn luyện cho cây thứ i
- Decision Tree i Cây phân lớp thứ i trong rừng
- Nút trong cây phân lớp

Thuật toán Random Forest được xây dựng từ nhiều Decision Tree, tuy nhiên mỗi Decision Tree sẽ khác nhau. Sau đó, kết quả dự đoán sẽ được tổng hợp từ những kết quả Decision Tree đó.

Quá trình hoạt động cơ bản:

- Tạo các tập con dữ liệu: Sử dụng Bootstrap Aggregating để tạo ra các tập con dữ liệu từ tập dữ liệu huấn luyện.
- Xây dựng cây quyết định: Với mỗi tập con dữ liệu được tạo ra, một Decision Tree được xây dựng.
- Sử dụng phương pháp voting: Mỗi cây quyết định được gán một trọng số dựa trên độ chính xác của nó trên tập dữ liệu huấn luyện. Random Forest sử dụng

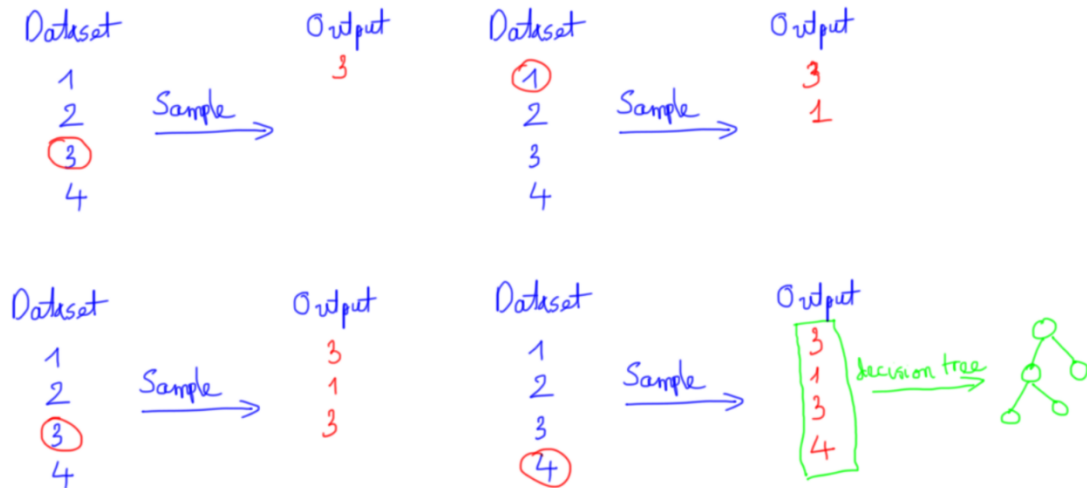
dùng voting để đưa ra dự đoán cuối cùng dựa vào kết quả tính trung bình của trọng số.

2.2.3 Thuật toán

Giả sử bộ dữ liệu của mình có n dữ liệu (sample) và mỗi dữ liệu có d thuộc tính (feature).

Để xây dựng mỗi cây quyết định mình sẽ làm như sau:

- Lấy ngẫu nhiên n dữ liệu từ bộ dữ liệu với kỹ thuật [Bootstrapping](#), hay còn gọi là **random sampling with replacement**. Tức khi mình sample được 1 dữ liệu thì mình không bỏ dữ liệu đấy ra mà vẫn giữ lại trong tập dữ liệu ban đầu, rồi tiếp tục sample cho tới khi sample đủ n dữ liệu. Khi dùng kỹ thuật này thì tập n dữ liệu mới của mình có thể có những dữ liệu bị trùng nhau.



Hình 2.6 Xây dựng thuật toán Random Forest

- Sau khi sample được n dữ liệu từ bước 1 thì mình chọn ngẫu nhiên ở k thuộc tính ($k < n$). Giờ mình được bộ dữ liệu mới gồm n dữ liệu và mỗi dữ liệu có k thuộc tính.

Do quá trình xây dựng mỗi cây quyết định đều có yếu tố ngẫu nhiên (random) nên kết quả là các cây quyết định trong thuật toán Random Forest có thể khác nhau.

Công thức:

$$y_{\text{pred}} = \text{argmax}(p_1, p_2, \dots, p_n)$$

- Trong đó:

- y_{pred} là giá trị dự đoán của điểm dữ liệu mới.
- p_1, p_2, \dots, p_n là các khả năng dự đoán của điểm dữ liệu mới, được tính bởi các cây quyết định trong rừng.

Cụ thể hơn, khả năng dự đoán của một cây quyết định cho một điểm dữ liệu mới được tính bằng công thức sau:

$$p = P(y = \text{class} \mid x)$$

- Trong đó:

- p là khả năng dự đoán.
- y là giá trị dự đoán, là một trong các lớp.
- x là điểm dữ liệu mới.

Khả năng dự đoán này được tính bằng cách sử dụng phương pháp bỏ phiếu.

Ví dụ, nếu có 5 cây quyết định trong rừng, và 3 cây quyết định dự đoán điểm dữ liệu mới là lớp 1, 2 cây quyết định dự đoán là lớp 2, thì khả năng dự đoán của điểm dữ liệu mới là lớp 1 là 3/5.

Công thức này cho thấy rằng, mô hình Random Forest dự đoán giá trị của một điểm dữ liệu mới bằng cách sử dụng khả năng dự đoán của các cây quyết định trong rừng. Khả năng dự đoán này được tính bằng cách sử dụng phương pháp bỏ phiếu.

2.2.4 Đặc điểm của Random Forest

- Ưu điểm:
 - Có thể sử dụng cả bài toán hồi quy và phân loại.
 - Có thể xử lý các tập dữ liệu lớn một cách hiệu quả và đưa ra dự đoán tốt có thể hiểu được một cách dễ dàng.
 - Có thể sử dụng nhiều máy chạy song song để chạy một thuật toán.
 - Có thể giải quyết tốt các bài toán có dữ liệu lớn, thiếu giá trị vì có cách chọn ngẫu nhiên thuộc tính nên các dữ liệu nhiều hoặc thiếu giá trị không gây ảnh hưởng tới kết quả.
- Nhược điểm:
 - Có thể bị chậm và không hiệu quả với các dự đoán theo thời gian.
 - Dữ liệu huấn luyện cần được đa dạng hóa và cân bằng về số nhãn lớp. Việc không cân bằng nhãn lớp làm cho kết quả có thể bị lệch về số đông nhãn lớp.
 - Thời gian huấn luyện có thể dài tùy số cây và số thuộc tính phân chia.

2.2.5 Ứng dụng của thuật toán Random Forest

Được ứng dụng chủ yếu vào bốn lĩnh vực sau:

- Ngân hàng: Xác định rủi ro cho vay và được sử dụng để phát hiện kẻ lừa đảo.
- Y học: Dự đoán xu hướng bệnh tật và nguy cơ của bệnh.
- Bất động sản: Xác định khu vực có thể sử dụng đất tương tự.
- Thương mại điện tử: Dự đoán các xu hướng quan tâm, sở thích của khách hàng dựa trên hành vi tiêu dùng của khách hàng trong quá khứ.

2.3 Phương pháp lấy mẫu Bootstrap

2.3.1 Giới thiệu Bootstrap

Bootstrap là một kỹ thuật lấy mẫu lại thống kê liên quan tới việc lấy mẫu ngẫu nhiên một tập dữ liệu có thay thế hay tạo mẫu dữ liệu giả để tạo các mẫu dữ liệu giả từ mẫu dữ liệu ban đầu. Nó thường được sử dụng như một phương tiện để định lượng sự không chắc chắn liên quan tới mô hình học máy.

Bootstrap được xem là phương pháp chuẩn trong phân tích thống kê và đã làm nên một cuộc cách mạng trong thống kê vì có thể giải quyết được nhiều vấn đề mà trước đây tưởng không có thể giải được.

Trong Random Forest, mỗi Decision Tree được tạo ra từ một mẫu dữ liệu giả được tạo ra bằng phương pháp Bootstrap. Điều này có nghĩa là mỗi cây sẽ học từ một tập dữ liệu khác nhau, chứa một tập hợp con ngẫu nhiên của các điểm dữ liệu ban đầu.

2.3.2 Thuật toán Bootstrap

Thuật toán Bootstrap lấy mẫu ngẫu nhiên có hoàn lại

A. Lấy mẫu

Cho tập dữ liệu D với N mẫu và K mẫu muốn lấy ra

Thực hiện K lần

- Lấy ngẫu nhiên một mẫu k từ tập D , ghi lại chỉ số k
- Đặt k lại tập D

Cuối cùng được k mẫu ngẫu nhiên

B. Sử dụng

Sử dụng tập ngẫu nhiên làm tập mẫu huấn luyện cho Decision Tree

Ví dụ:

Giả sử chúng ta có một tập dữ liệu như hình bên dưới.

	A	B	C	D	E	F
1	Tên	Giới tính	Năm sinh	Quê Quán	Điểm thi	
2	Anh	Nam	1999	Trà Vinh	8	
3	Bình	Nam	1998	Đồng Nai	10	
4	Đạt	Nam	2000	Trà Vinh	9	
5	Diệu	Nữ	2001	Bình Thuận	7	
6	Hiển	Nam	1996	Bình Thuận	9	
7	Hiếu	Nam	2000	Trà Vinh	7	
8	Hương	Nữ	2001	Vũng Tàu	8	
9	My	Nữ	1999	Bến Tre	7	
10	Nghi	Nữ	1996	An Giang	9	
11	Ngọc	Nữ	1999	Trà Vinh	9	
12	Nhi	Nữ	2001	Bến Tre	10	
13	Nhu	Nữ	2001	Trà Vinh	9	
14	Thuỷ	Nữ	2000	Bình Thuận	7	
15	Trân	Nữ	1998	Tiền Giang	8	
16	Tuyền	Nữ	1997	Bến Tre	9	
17						

Hình 2.7 Hình ảnh về tập dữ liệu

Để tạo một mẫu dữ liệu bootstrap từ tập dữ liệu này, chúng ta sẽ tạo một mẫu ngẫu nhiên gồm 10 điểm dữ liệu, với mỗi điểm dữ liệu có khả năng được chọn một lần.

Mẫu dữ liệu Bootstrap sau đây có thể sẽ được tạo:

[1,2,3,4,5,6,7,8,9,10]

Mẫu dữ liệu có thể được sử dụng để tạo cây quyết định mới. Cây quyết định này sẽ học từ điểm dữ liệu 1,2,3,4,5,6,7,8,9 và 10.

Quá trình này sẽ được lặp lại nhiều lần để tạo ra nhiều cây quyết định khác nhau. Mỗi cây quyết định sẽ học từ một tập dữ liệu khác nhau, chứa một tập hợp con ngẫu nhiên của các điểm dữ liệu ban đầu.

Việc sử dụng các cây quyết định Bootstrap để tạo rừng ngẫu nhiên giúp giảm thiểu overfitting. Điều này là do mỗi cây quyết định sẽ học từ một tập dữ liệu khác

nhau, khiến chúng ít khả năng bị ảnh hưởng bởi các điểm dữ liệu bất thường trong tập dữ liệu ban đầu.

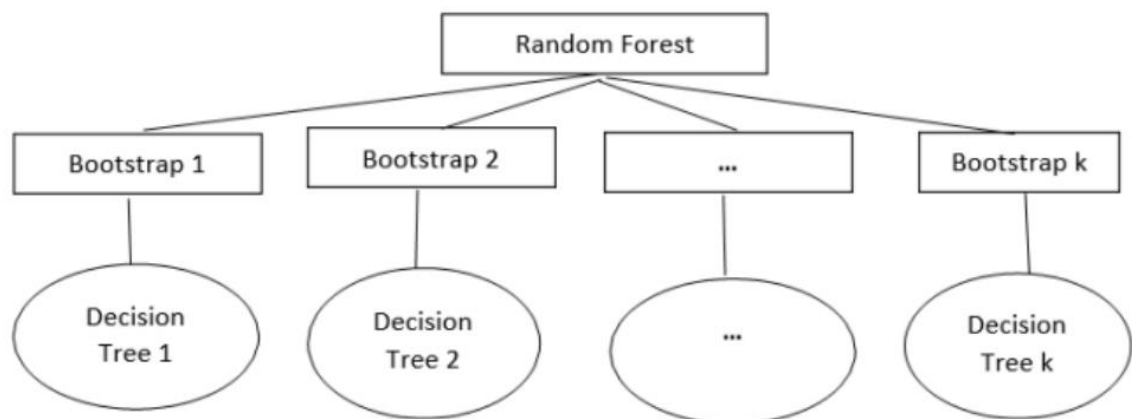
2.3.3 Đặc điểm của Bootstrap

Phương pháp chọn ngẫu nhiên có hoàn lại nhằm mục đích tạo ra nhiều mẫu ngẫu nhiên từ một mẫu và qua cách chọn này, tập hợp những mẫu có thể đại diện cho một quần thể.

Bootstrap có thể cung cấp thông tin chi tiết về phân bố của số trung bình, khoảng tin cậy cũng như xác suất của số trung bình dựa trên một mẫu duy nhất.

2.3.4 Bootstrap trong Random Forest

Trong Random Forest thuật toán Bootstrap được sử dụng để tạo mẫu ngẫu nhiên cho từng cây p. Vậy mỗi cây sẽ có một tập ngẫu nhiên riêng biệt. Ngoài ra còn sử dụng để đánh giá nội tại của thuật toán (Out-of-bag).



Hình 2.8 Bootstrap trong Random Forest

2.4 Tham số và siêu tham số trong model Random Forest

Sau đây chúng tôi sẽ giới thiệu một số tham số và siêu tham số được dùng trong Random Forest

2.4.1 Tham số (Parameter)

a. `n_jobs`

Tham số này cho động cơ biết có bao nhiêu bộ vi xử lý được phép sử dụng. Giá trị “-1” có nghĩa là không có hạn chế trong khi giá trị bằng “1” có nghĩa là nó chỉ có thể sử dụng một bộ xử lý.

b. `random_state`

Tham số này làm cho một giải pháp dễ dàng sao chép. Một giá trị xác định của `random_state` sẽ luôn tạo ra kết quả giống nhau nếu được cung cấp với cùng tham số và dữ liệu huấn luyện.

c. `oob_score`

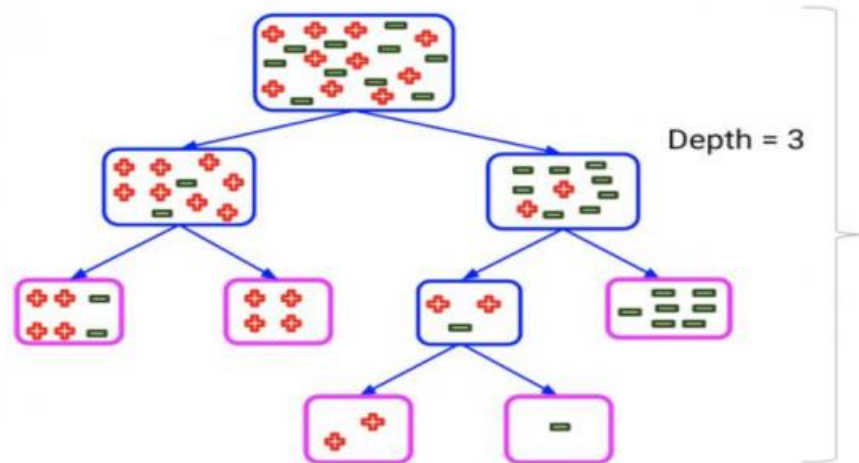
Đây là phương pháp xác nhận chéo rừng ngẫu nhiên. Nó rất giống kỹ thuật xác nhận, tuy nhiên, điều này nhanh hơn rất nhiều. Phương pháp này chỉ cần gắn thẻ cho mọi quan sát được sử dụng trong các stress khác nhau. Và sau đó, nó tìm ra điểm bầu chọn tối đa cho mọi quan sát chỉ dựa trên những cây không sử dụng quan sát cụ thể này để đào tạo.

2.4.2 Siêu tham số (Hyperparameter)

Trong học máy, siêu tham số là một tham số có giá trị được đặt trước khi quá trình học bắt đầu. Ngược lại, các giá trị của tham số khác có được thông qua huấn luyện.

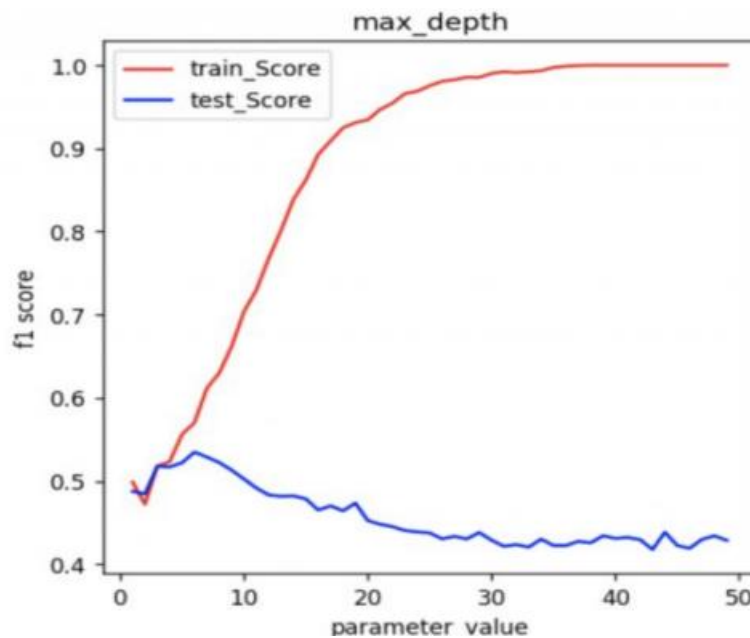
a. `max_depth`

Các `max_depth` của một cây trong Random Forest được định nghĩa là con đường dài nhất giữa nút gốc và nút lá:



Hình 2.9 Sơ đồ *max_depth*

Sử dụng tham số *max_depth*, có thể giới hạn độ sâu mà chúng tôi muốn mọi cây trong khu rừng ngẫu nhiên của mình phát triển.



Hình 2.10 Sơ đồ *hyperparameter value*

Trong biểu đồ này, chúng ta có thể thấy rõ rằng khi độ sâu tối đa của cây quyết định tăng lên, hiệu suất của mô hình trên tập huấn luyện sẽ tăng liên tục. Mặc khác, khi giá trị *max_depth* tăng lên, hiệu suất trên tập thử nghiệm sẽ tăng ban đầu nhưng sau một thời điểm nhất định, nó bắt đầu giảm nhanh chóng.

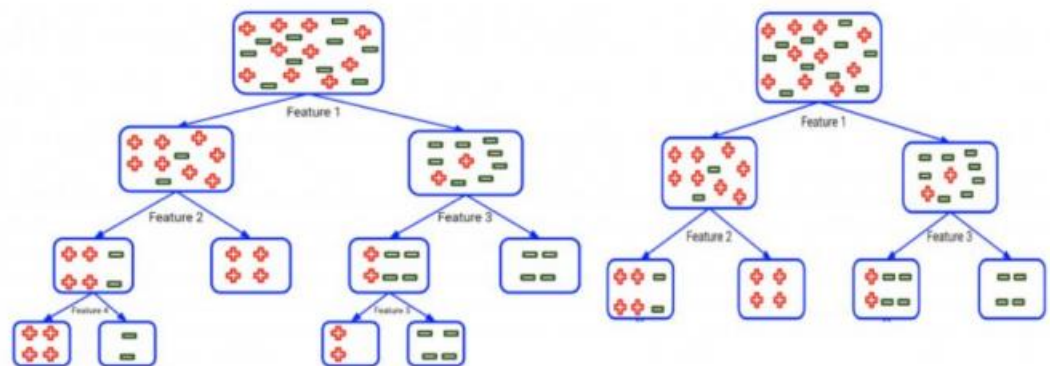
Cây bắt đầu trang bị quá mức cho tập huấn luyện và do đó không thể tổng quát hóa các điểm không nhìn thấy trong tập thử nghiệm.

b. `min_sample_split`

min_sample_split – một tham số cho cây quyết định trong một khu rừng ngẫu nhiên về số lượng quan sát cần thiết tối thiểu trong bất kỳ nút nhất định nào mà ta có thể tách nó ra.

Giá trị mặc định của *minimum_sample_split* được gán cho 2. Điều này, có nghĩa là nếu bất kỳ nút đầu nào có nhiều hơn 2 quan sát và không phải là nút thuần túy, chúng ta có thể chia nó thành các nút con.

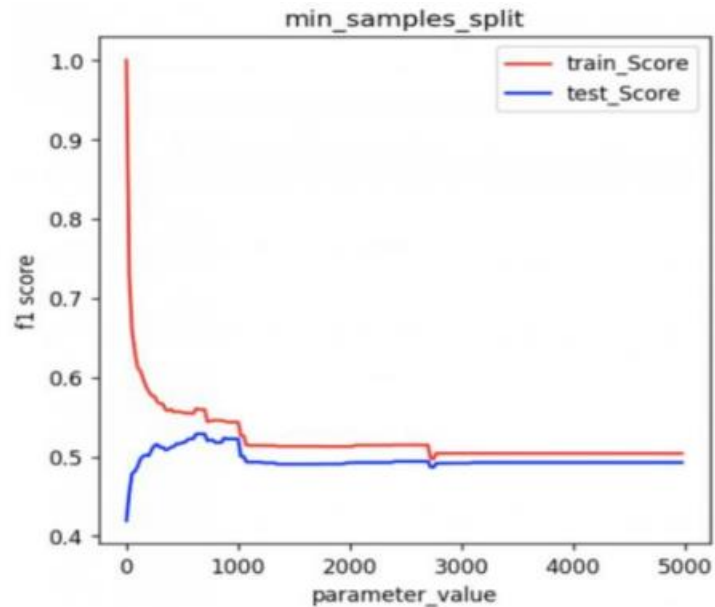
Việc có giá trị mặc định là 2 đặt ra vấn đề rằng một cây thường được tiếp tục lách cho đến khi các nút hoàn toàn thuần khiết. Kết quả là cây phát triển về kích thước và do đó làm quá tải dữ liệu.



Hình 2.11 Sơ đồ *min_sample_split*

Bằng cách tăng giá trị của *min_sample_split* chúng ta có thể giảm số lần phân tách xảy ra trong cây quyết định do đó ngăn mô hình không được trang bị quá mức. Trong ví dụ trên, nếu chúng ta tăng giá trị *min_sample_split* từ 2 lên 6, thì cây bên trái sẽ giống như cây bên phải.

Bây giờ, hãy xem ảnh hưởng của `min_samples_split` đối với hiệu suất của mô hình. Biểu đồ dưới đây được vẽ bằng cách xem xét rằng tất cả những tham số khác vẫn giữ nguyên và chỉ giá trị của `min_sample_split` được thay đổi:



Hình 2.12 Sơ đồ `min_sample_split` hyperparameter value

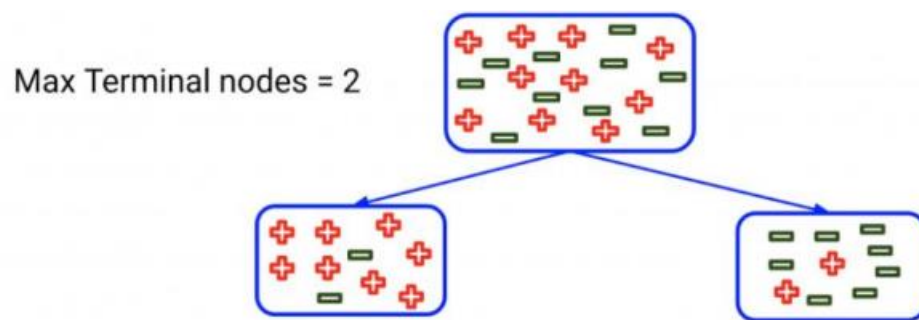
Khi tăng giá trị của siêu tham số `min_sample_split`, chúng ta có thể thấy rõ rằng đối với giá trị nhỏ của tham số, có sự khác biệt đáng kể giữa điểm đào tạo và điểm kiểm tra. Nhưng khi giá trị của tham số tăng lên, sự khác biệt giữa điểm đào tạo và điểm kiểm tra sẽ giảm.

Nhưng có một điều cần phải lưu ý. Nhưng giá trị tham số tăng quá nhiều, sẽ có sự sụt giảm tổng thể về cả điểm đào tạo và điểm kiểm tra. Điều này là do thực tế là yêu cầu tối thiểu của việc tách một nút quá cao nên không có sự phân tách đáng kể nào được quan sát. Kết quả là, khu rừng ngẫu nhiên kém chất lượng.

c. `max_leaf_nodes`

Tiếp theo, hãy chuyển sang một siêu tham số khác trong Random Forest được gọi là *max_leaf_nodes*. Siêu tham số này đặt ra một điều kiện về việc tách nút trong cây và do đó hạn chế sự phát triển của cây. Nếu sau khi tách mà chúng ta có nhiều nút đầu cuối hơn so với đầu cuối đã chỉ định, nó sẽ dừng việc tách và cây sẽ không phát triển thêm.

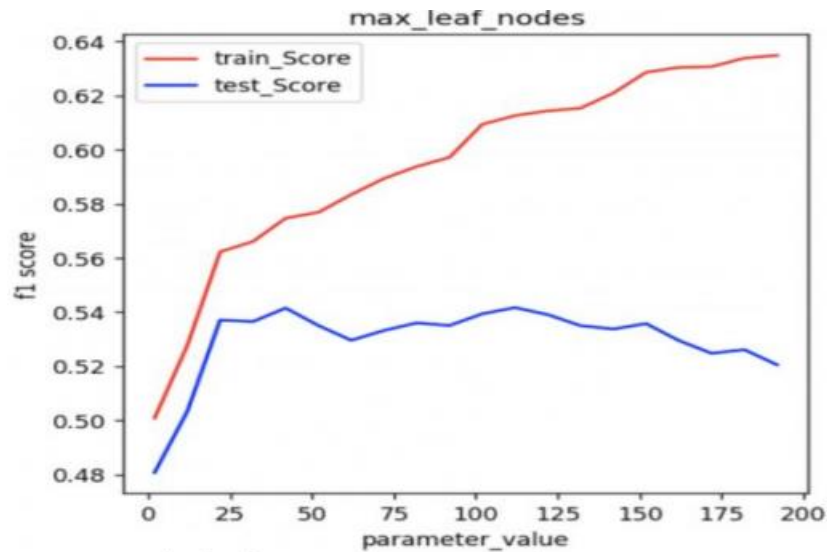
Giả sử chúng ta đặt các nút đầu cuối tối đa là 2 trong trường hợp này. Vì chỉ có một nút, nó sẽ cho phép cây phát triển thêm.



Hình 2.13 Sơ đồ *max_leaf_nodes*

Bây giờ, sau lần tách đầu tiên, bạn có thể thấy rằng có 2 nút ở đây và chúng tôi đã đặt các nút đầu cuối tối đa là 2. Do đó, cây sẽ kết thúc ở đây và sẽ không phát triển thêm. Đây là cách thiết lập các nút đầu cuối tối đa và *max_leaf_nodes* có thể giúp chúng tôi ngăn chặn việc trang bị quá mức.

Lưu ý rằng nếu giá trị của *max_leaf_nodes* quá nhỏ, thì rừng ngẫu nhiên có khả năng được trang bị thấp hơn. Hãy xem tham số này ảnh hưởng như thế nào đến hiệu suất của mô hình rừng ngẫu nhiên:



Hình 2.14 Sơ đồ `max_leaf_nodes` hyperparameter value

Chúng ta có thể thấy rằng khi giá trị tham số rất nhỏ, cây không được trang bị đầy đủ và khi giá trị tham số tăng lên, hiệu suất của cây qua cả thử nghiệm và huấn luyện đều tăng. Theo biểu đồ này, cây bắt đầu quá mức khi giá trị tham số vượt quá 25.

d. `min_samples_leaf`

Trọng tâm trong đồ án này là `min_sample_leaf`. Siêu thông số Random Forest này chỉ định số lượng mẫu tối thiểu cần có trong nút lá sau khi tách một nút.

Hãy hiểu `min_sample_leaf` bằng cách sử dụng một ví dụ. Giả sử chúng tôi đã đặt các mẫu tối thiểu cho một nút đầu cuối là 5:

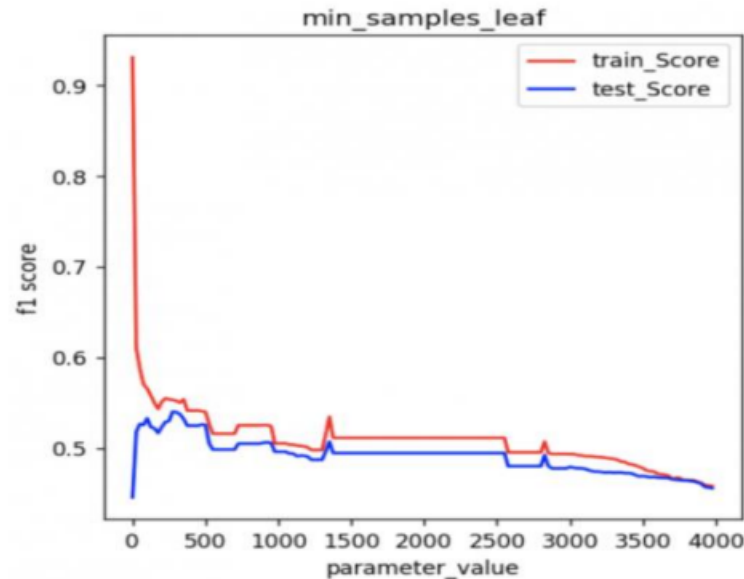


Hình 2.15 Sơ đồ *min_sample_leaf*

Cây bên trái tượng trưng cho cây không bị bó buộc. Ở đây, các nút được đánh dấu bằng màu xanh lá cây thỏa mãn điều kiện vì chúng có tối thiểu 5 mẫu. Do đó, chúng sẽ được coi là nút lá hoặc nút cuối.

Tuy nhiên, nút đỏ chỉ có 3 mẫu và do đó nó sẽ không được coi là nút lá. Nút cha của nó sẽ trở thành nút lá. Đó là lý do tại sao cây bên phải biểu thị kết quả khi chúng ta đặt các mẫu tối thiểu cho nút đầu cuối là 5.

Vì vậy, chúng tôi đã kiểm soát sự phát triển của cây bằng cách đặt tiêu chí của mẫu tối thiểu cho các nút đầu cuối. Như đã đoán, tương tự như hai siêu tham số được đề cập ở trên, siêu tham số này cũng giúp ngăn chặn việc trang bị quá mức khi giá trị tham số tăng lên.



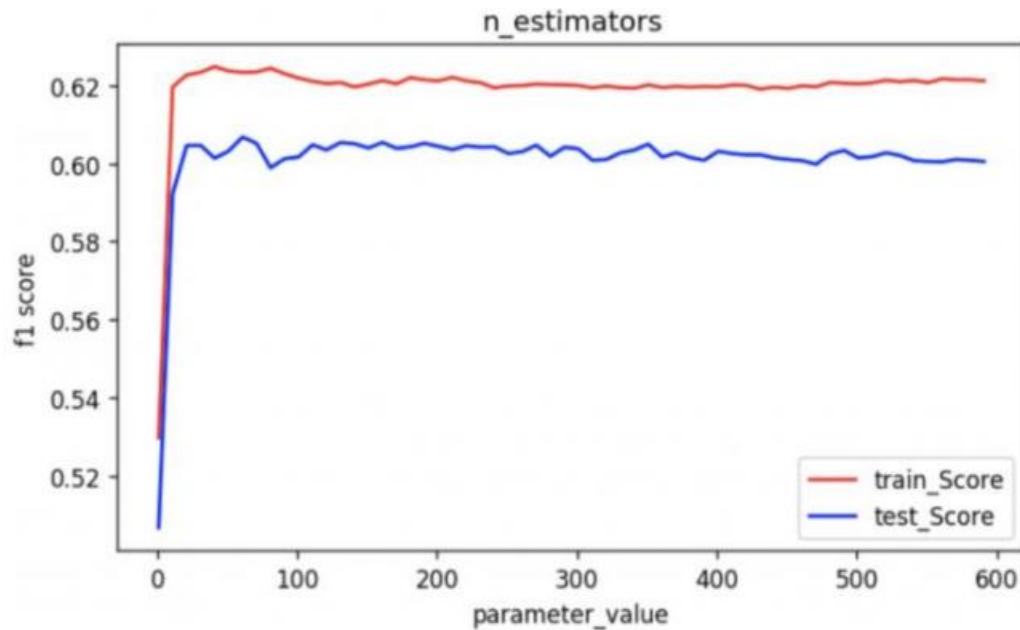
Hình 2.16 Sơ đồ *min_samples_leaf* parameter value

Chúng ta có thể thấy rõ rằng mô hình Random Forest đang overfitting khi giá trị tham số rất thấp (khi tham số < 100), nhưng hiệu suất của mô hình nhanh chóng tăng lên và khắc phục vấn đề overfitting ($100 < \text{giá trị tham số} < 400$). Nhưng khi giá trị tiếp tục tăng giá trị của tham số (> 500), thì mô hình sẽ dần trôi về phía mức độ thích hợp.

e. *n_estimators*

Thuật toán Random Forest không khác là gì ngoài một nhóm cây. Nhưng cần phải xem xét có bao nhiêu cây trong rừng? Đây là một câu hỏi phổ biến mà các nhà khoa học về dữ liệu mới sẽ hỏi. Và nó là một trong những hợp lệ.

Có thể nói rằng nhiều cây hơn sẽ có thể tạo ra một kết quả tổng quát hơn, phải không? Nhưng bằng cách chọn nhiều cây hơn, độ phức tạp và thời gian của mô hình cũng sẽ phải tăng lên. Và để thấy hiểu rõ hơn có thể xem qua biểu đồ này. Hiệu suất mô hình tăng mạnh và sau đó gây ra trì trệ ở mức độ nhất định.



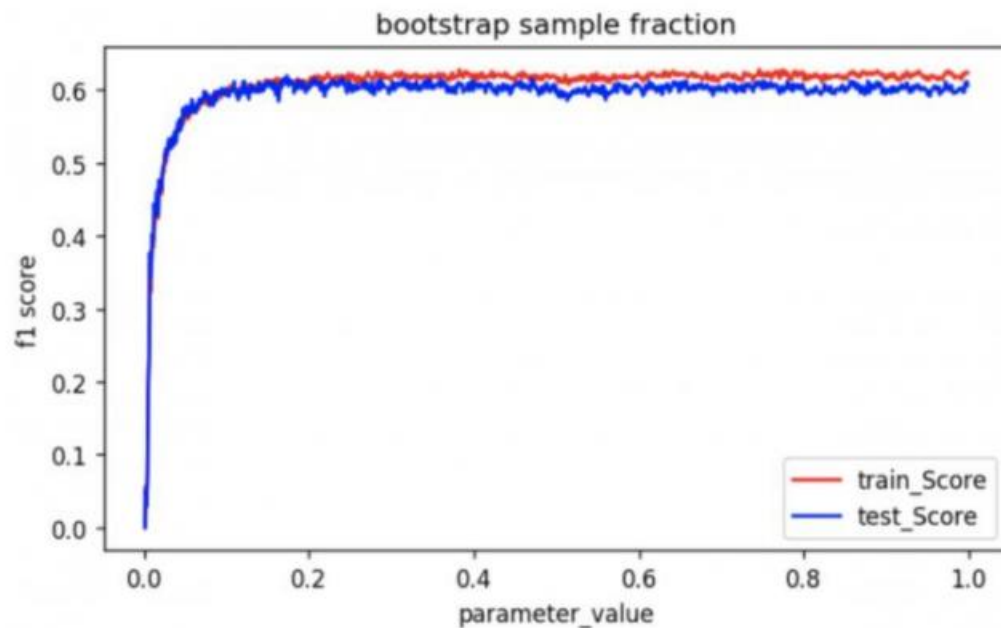
Hình 2.17 Mô hình $n_estimators$ và hyper parameter để thấy rõ hiệu suất và mức độ trì trệ khi tăng số lượng cây lên

Điều này có nghĩa là chọn một số lượng lớn các công cụ ước lượng trong một mô hình rừng ngẫu nhiên không phải là ý tưởng tốt nhất. Mặc dù nó sẽ không làm suy giảm mô hình, nhưng nó có thể giúp tiết kiệm sự phức tạp trong tính toán và ngăn việc sử dụng bình cứu hỏa trên CPU của máy tính.

f. max_samples

Các max_samples hyperparameter xác định những gì phần của tập dữ liệu ban đầu được trao cho bất kỳ cây cá thể.

Hình dưới đây giải đáp cho thắc mắc dữ liệu nhiều hơn luôn tốt hơn. Liệu điều đó có thật sự hợp lý không?



Hình 2.18 Mô hình *max_samples*

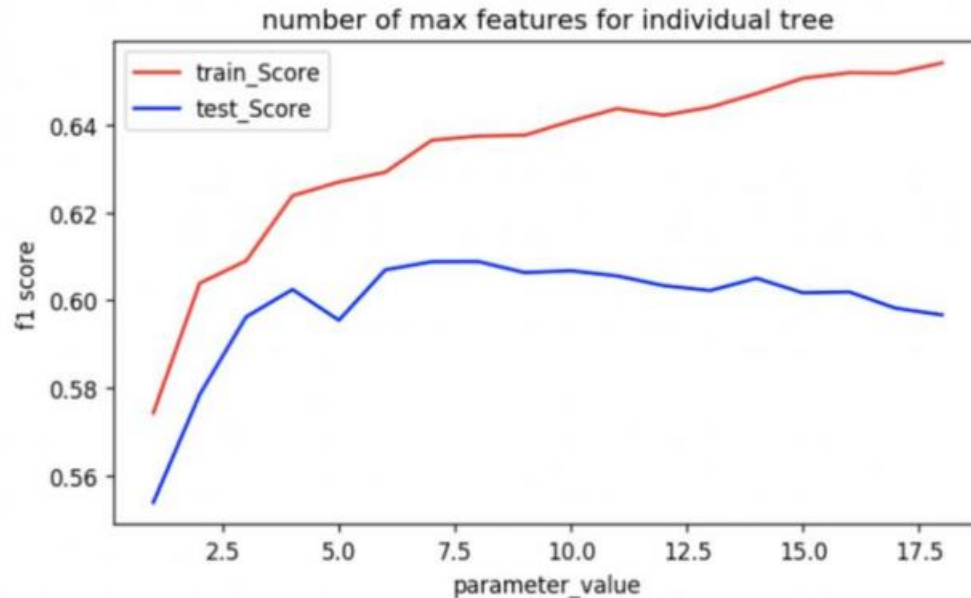
Có thể nhìn thấy được hiệu suất của mô hình tăng mạnh và sau đó bão hòa rất nhanh. Vì vậy không nhất thiết phải cung cấp cho mỗi decision tree của random forest toàn bộ dữ liệu. Theo mô hình trên có thể nhận thấy được hiệu suất mô hình đạt mức tối đa khi dữ liệu được cung cấp nhỏ hơn 0.2 phần của dữ liệu tập ban đầu.

Mặc dù phần này sẽ khác với tập dữ liệu này sang tập dữ liệu khác, nhưng chúng ta có thể phân bổ một phần nhỏ hơn của dữ liệu được khởi động ở mỗi cây quyết định. Do đó, thời gian đào tạo của mô hình Random Forest được giảm một cách đáng kể.

g. **max_features**

Cuối cùng, chúng ta sẽ quan sát hiệu ứng của siêu tham số *max_features*. Điều này giống với số lượng tính năng tối đa được cung cấp cho mỗi cây trong một khu rừng ngẫu nhiên.

Khu rừng ngẫu nhiên một số mẫu ngẫu nhiên từ các đối tượng địa lý để tìm ra sự phân chia tốt nhất. Vì vậy hãy xem sự thay đổi của tham số này có thể ảnh hưởng như thế nào đến hiệu suất của mô hình Random Forest.



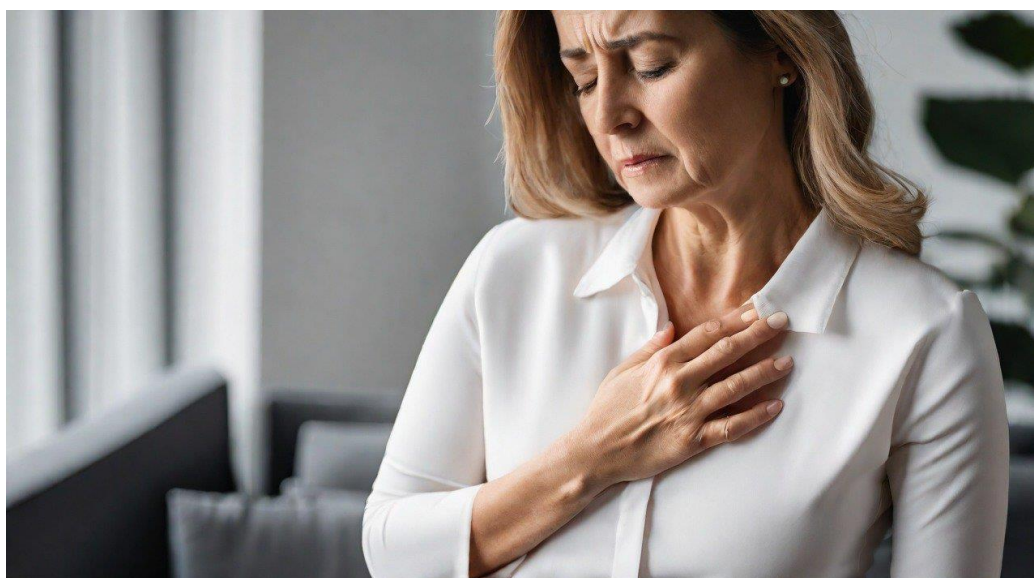
Hình 2.19 Sự thay đổi của mô hình `max_features` ảnh hưởng đến hiệu suất mô hình Random Forest

Chúng ta có thể thấy rằng hiệu suất của mô hình ban đầu tăng lên khi số lượng `max_feature` tăng lên. Tuy nhiên, sau một thời điểm nhất định, `train_score` tiếp tục tăng nhưng `test_score` lại dần bão hòa, thậm chí là có xu hướng giảm dần khi về cuối. Điều đó cho thấy mô hình đang bắt đầu quá mức.

Lý tưởng nhất, hiệu suất tổng thể của mô hình là giá trị cao nhất gần với 6 của các tính năng tối đa. Đó là một quy ước tốt để xem xét giá trị mặc định của tham số này, được đặt thành căn bậc hai của số lượng đối tượng có trong tập dữ liệu. Số lượng `max_features` lý tưởng thường có xu hướng nằm gần với giá trị này.

CHƯƠNG 3: BÀI TOÁN DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH TIM MẠCH

Theo thống kê của Tổ chức Y tế thế giới (WHO), hàng năm có khoảng 17,9 triệu người tử vong do bệnh tim, trong đó có 85% trường hợp gây ra bởi nhồi máu cơ tim và đột quỵ. Tại Việt Nam, hằng năm có gần 200.000 người chết vì bệnh tim, con số này cao hơn so với tỷ lệ tử vong do ung thư. Đáng chú ý, các loại bệnh như động mạch não, mạch vành và động mạch ngoại biên đang trở nên phổ biến ở những người trẻ trong khi trước đây, các bệnh trên chỉ thường xuất hiện ở người cao tuổi.



Hình 3. 1 Hình người đang có nguy cơ mắc bệnh tim

Tuy nhiên, nhiều người trẻ thường tỏ ra chủ quan, cho rằng không có nguy cơ mắc bệnh tim mạch, do đó họ không thực hiện các biện pháp phòng ngừa hoặc tầm soát sớm. Điều này dẫn đến những biến chứng nghiêm trọng ảnh hưởng đến năng suất hoạt động của xã hội. Hơn nữa, trường hợp bệnh tim mạch bẩm sinh thường không được chuẩn đoán và điều trị kịp thời trong những năm đầu sau khi sinh, khiến cho tỷ lệ mắc bệnh tim mạch ở người trẻ gia tăng kinh ngạc.

Chính vì lý do đó mà nhóm chúng tôi đã quyết định chọn bài toán “Dự đoán khả năng mắc bệnh tim mạch” bởi sự ý nghĩa và đây cũng là vấn đề mà chúng tôi nên quan tâm với chính bản thân mình, cũng như những người xung quanh.

3.1 Giới thiệu bài toán

Bệnh tim là một tập hợp các tình trạng ảnh hưởng đến sức khỏe của trái tim và hiệu suất hoạt động của các mạch máu chịu trách nhiệm cung cấp oxy, dưỡng chất đến nuôi cơ tim. Các loại bệnh tim mạch thường gặp bao gồm: rối loạn nhịp tim, cao huyết áp, bệnh mạch vành, các bệnh về van tim cùng nhiều loại khác.

Một số yếu tố nguy cơ góp phần vào sự phát triển của bệnh tim mạch bao gồm:

- Tuổi tác: người cao tuổi có nguy cơ cao.
- Giới tính: Nam giới thường có nguy cơ mắc bệnh cao hơn so với phụ nữ. Tuy nhiên, đến thời kỳ mãn kinh, tỷ lệ mắc bệnh tim của phụ nữ sẽ tăng lên.
- Tiền sử bệnh gia đình: Nếu trong gia đình từng có người mắc bệnh tim mạch, đặc biệt là người thân trực hệ, bạn có nguy cơ mắc bệnh cao hơn.
- Hút thuốc lá: Nicotine trong thuốc lá có thể làm thắt chặt mạch máu và carbon monoxide có thể gây hại cho lớp lót bên trong mạch máu, dẫn đến xơ vữa động mạch. Người hút thuốc thường có nguy cơ mắc bệnh cao hơn so với người không hút thuốc.
- Chế độ ăn uống không lành mạnh: Một chế độ ăn uống giàu chất béo, nhiều muối, đường và cholesterol có thể tăng nguy cơ mắc bệnh tim mạch.
- Huyết áp cao: Huyết áp không kiểm soát được có thể gây làm dày và cứng động mạch, hạn chế sự lưu thông của máu.
- Mức cholesterol cao trong máu: Cao huyết áp cholesterol có thể dẫn đến tích tụ mảng bám và xơ vữa động mạch.
- Tiểu đường: Tiểu đường tăng nguy cơ mắc bệnh tim mạch.
- Béo phì: Sự thừa cân cùng với béo phì tăng nguy cơ các yếu tố nguy cơ.
- Ít vận động: Thiếu tập thể dục thường kết nối với nhiều dạng bệnh tim mạch và yếu tố nguy cơ khác.

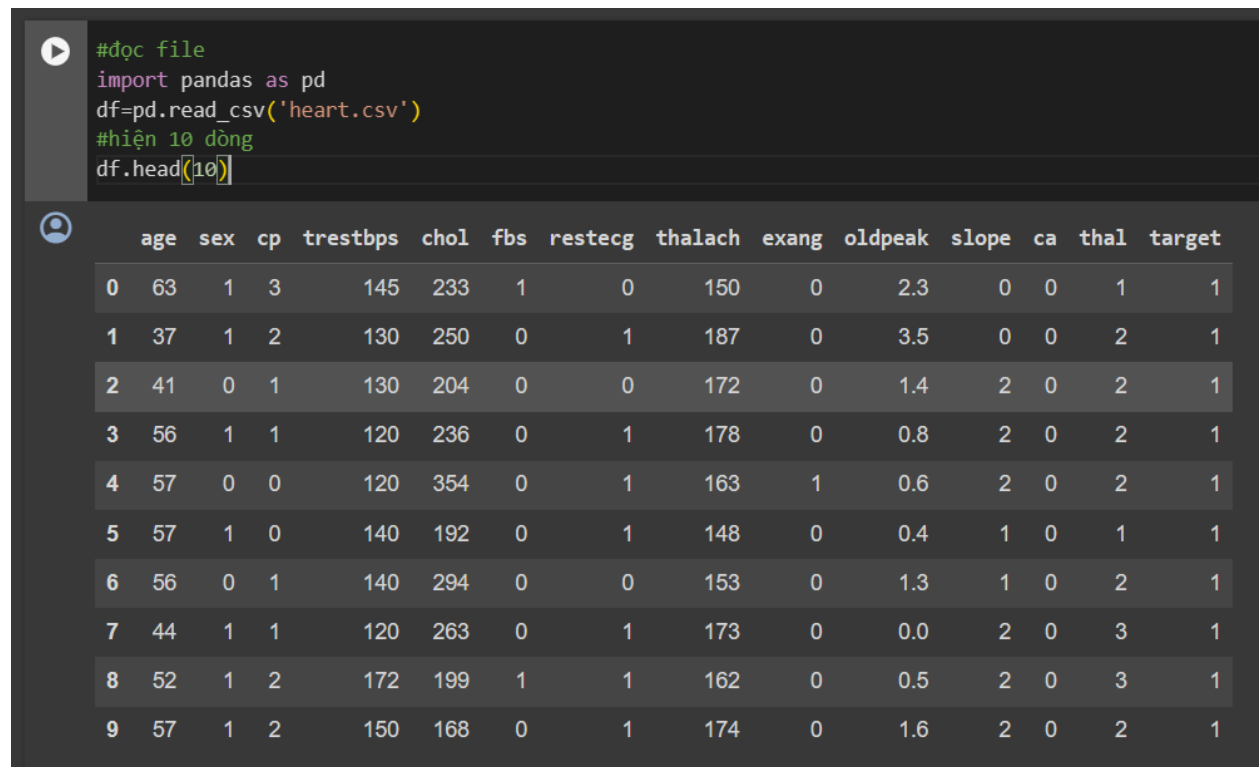
- Căng thẳng: Căng thẳng không được quản lý có thể gây hỏng động mạch và làm tăng nguy cơ mắc bệnh tim mạch cùng với các yếu tố nguy cơ khác.
- Sức khỏe răng miệng kém: Nếu răng và nướu không khỏe mạnh, vi khuẩn có thể xâm nhập vào máu và di chuyển đến tim, gây viêm nội tâm mạc.

Từ một số yếu tố trên chúng tôi quyết định chọn để dùng, lấy dữ liệu để dự đoán nguy cơ mắc bệnh tim.

Nguồn dữ liệu và code của bài toán được lấy từ Kho lưu trữ github của nhóm <https://github.com/tsdevtool/13-Group-AI-CMP163-HUTECH-21DTHE4.git> và tệp heart.csv được lấy từ nguồn dữ liệu trên website Kaggle (<https://www.kaggle.com/search?q=heart.csv>).

3.2 Tập dữ liệu

Tập dữ liệu trong file Heart.csv gồm 303 dòng dữ liệu và cột dữ liệu.



```
#đọc file
import pandas as pd
df=pd.read_csv('heart.csv')
#hiện 10 dòng
df.head(10)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

Hình 3. 2 Dòng dữ liệu từ Heart.csv

* Thông tin 14 cột dữ liệu:

- **age**: Tuổi của người đó tính theo năm
- **sex**: Giới tính của một người (1 = nam, 0 = nữ)
- **cp**: Đau ngực đã trải qua (Giá trị 1: đau thắt ngực điển hình, Giá trị 2: đau thắt ngực không điển hình, Giá trị 3: đau không đau thắt ngực, Giá trị 4: không có triệu chứng)
- **trestbps**: Huyết áp lúc nghỉ của người đó (mm Hg)
- **chol**: Phép đo cholesterol của người đó tính bằng mg / d
- **fbs**: Đường huyết lúc đói của người đó (> 120 mg / dl, 1 = true; 0 = false)
- **restecg**: Đo điện tâm đồ khi nghỉ ngơi (0 = bình thường, 1 = có bất thường sóng ST-T, 2 = hiển thị phì đại thất trái có thể xảy ra hoặc xác định theo tiêu chí của Estes)
- **thalach**: Nhịp tim tối đa của người đó đạt được
- **exang**: Đau thắt ngực do tập thể dục (1 = có; 0 = không)
- **oldpeak**: ST bị suy giảm do tập thể dục so với khi nghỉ ngơi ('ST' liên quan đến các vị trí trên đồ thị ECG.)
- **slope**: độ dốc của đoạn ST bài tập đỉnh cao (Giá trị 1: dốc lên, Giá trị 2: bằng phẳng, Giá trị 3: dốc xuống)
- **ca**: Số lượng tàu chính (0-3)
- **thal**: Một rối loạn về máu được gọi là thalassemia (3 = bình thường; 6 = khiếm khuyết cố định; 7 = khiếm khuyết có thể hồi phục)
- **target**: Bệnh tim (0 = không, 1 = có).

*** Tiến hành phân tích thuộc tính dữ liệu:**

```
#thông tin về số dòng và số cột của bộ dữ liệu
print("Số dòng của bộ dữ liệu là ",df.shape[0])
print()
print("Số cột của bộ dữ liệu là ",df.shape[1])
```

Số dòng của bộ dữ liệu là 303

Số cột của bộ dữ liệu là 14

```
#kiểm tra dữ liệu bị thiếu- missing value
df.isna().sum()
```

```
age      0
sex      0
cp       0
trestbps 0
chol     0
fbs      0
restecg  0
thalach  0
exang    0
oldpeak  0
slope    0
ca       0
thal     0
target   0
dtype: int64
```

Hình 3. 3 Số dòng số cột và kiểm tra dữ liệu có bị missing value

```
#Thông tin về các cột, các thuộc tính của bộ dữ liệu
df.info()
'''Hầu hết tất cả dữ liệu được mã hóa thành số nên các cột dữ liệu toàn bộ là số'''

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   age         303 non-null    int64
 1   sex         303 non-null    int64
 2   cp          303 non-null    int64
 3   trestbps    303 non-null    int64
 4   chol        303 non-null    int64
 5   fbs         303 non-null    int64
 6   restecg     303 non-null    int64
 7   thalach     303 non-null    int64
 8   exang       303 non-null    int64
 9   oldpeak     303 non-null    float64
10  slope       303 non-null    int64
11  ca          303 non-null    int64
12  thal        303 non-null    int64
13  target      303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
'Hầu hết tất cả dữ liệu được mã hóa thành số nên các cột dữ liệu toàn bộ là số'
```

Hình 3. 4 Thông tin các cột và các thuộc tính của cột dữ liệu

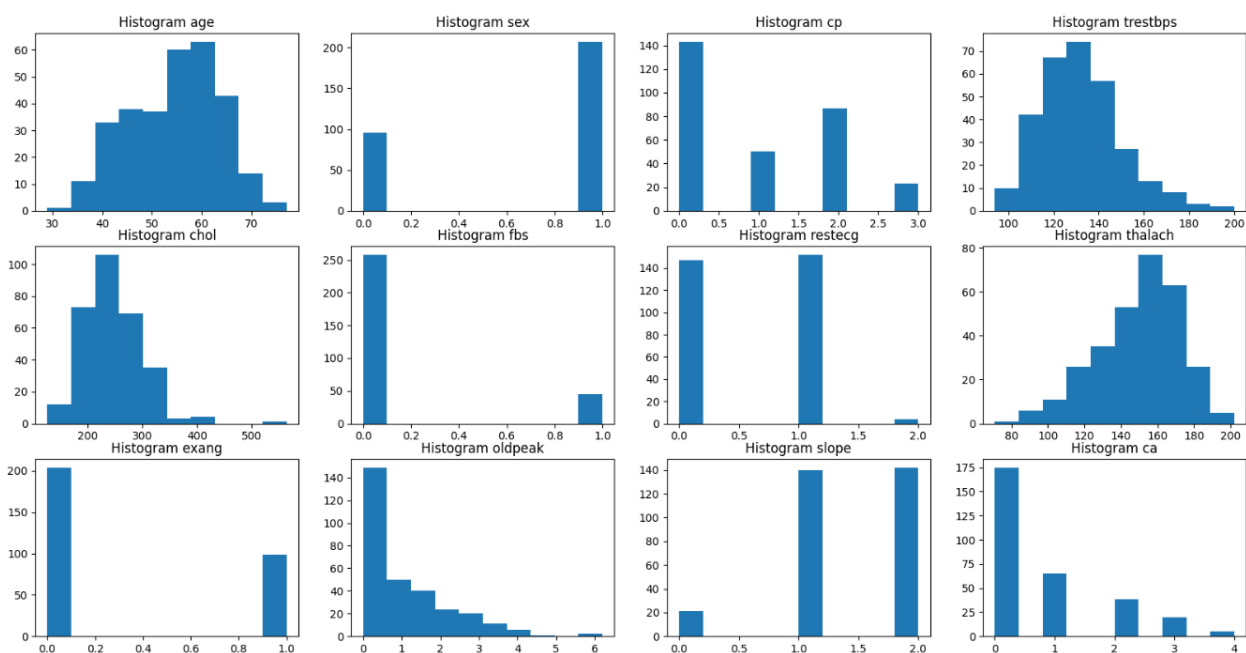
```
#Thống kê mô tả cơ bản các thuộc tính
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

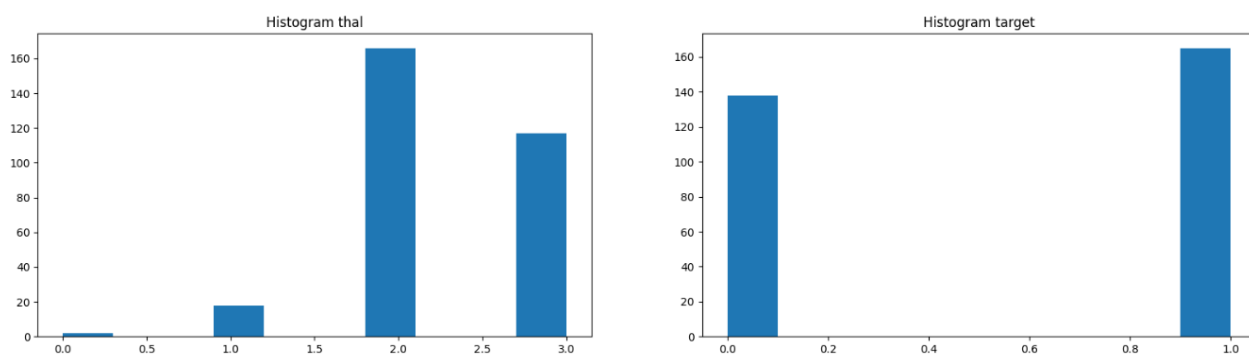
Hình 3. 5 Thống kê cơ bản các thuộc tính

3.3 Tiền xử lý dữ liệu

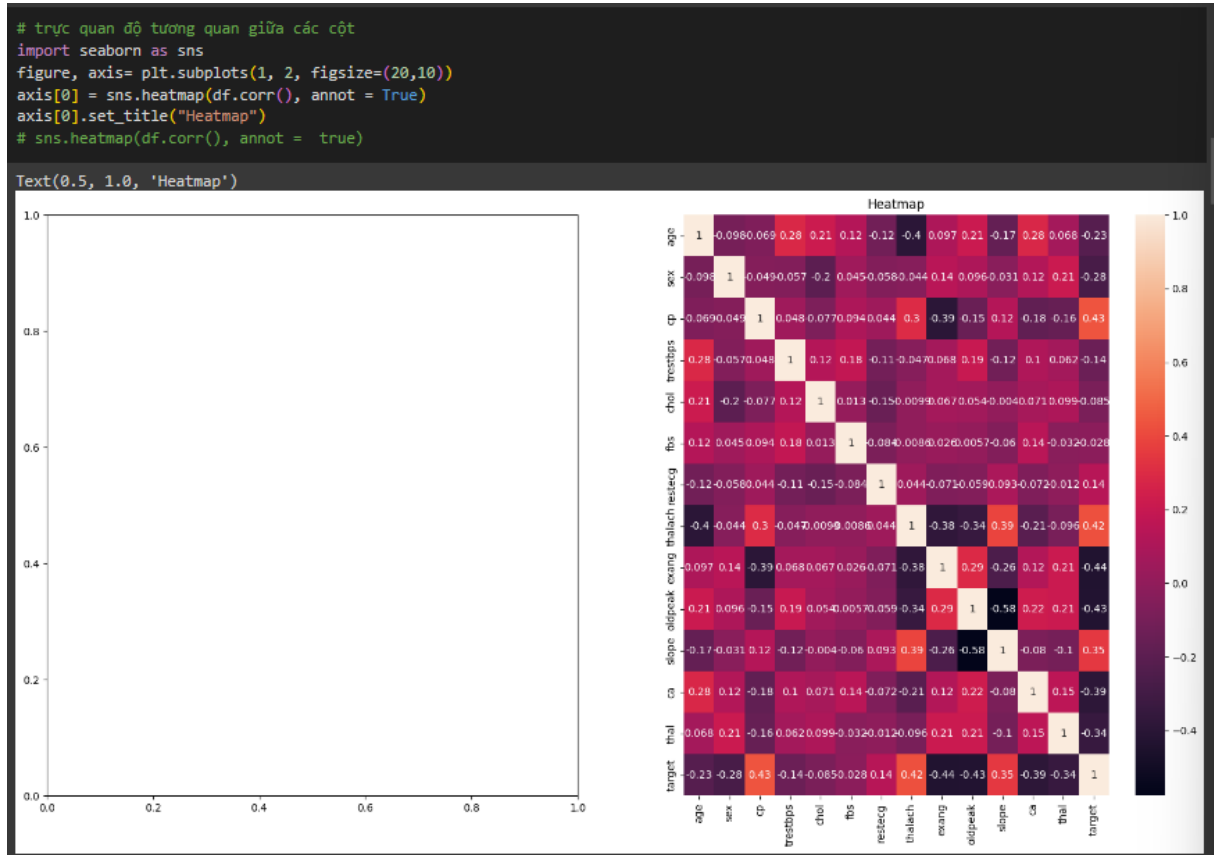
* Trực quan dữ liệu:



Hình 3. 6 Biểu đồ trực quan các dữ liệu



Hình 3. 7 Biểu đồ trực quan dữ liệu về rối loạn về máu và có mắc bệnh tim hay không



Hình 3. 8 Trực quan độ tương quan giữa các cột

- Như đã nói ở đề tài thì cột target là cột cần dự đoán (thuộc tính nhãn) và các cột còn lại là các đặc trưng đầu vào.
- Phần tiền xử lý dữ liệu: là một bộ dữ liệu sạch, từ phần tìm hiểu dữ liệu bên trên có thể thấy bộ dữ liệu đã được mã hóa kèm theo đó là không có giá trị bị thiếu (missing value).
- Để xây dựng được mô hình có kết quả tốt cần có các bước chuẩn bị tốt. Đối với bài toán dự đoán khả năng mắc bệnh tim (có hoặc không) thì việc cân bằng số nhãn sẽ giúp việc huấn luyện mô hình chính xác hơn, tránh trường hợp bị overfitting. Như biểu đồ histogram vẽ ở phần trên có thể thấy 2 nhãn 1 và 0 chênh lệch không nhiều nhưng đảm bảo việc chính xác cho mô hình

nhóm chúng tôi quyết định dùng kỹ thuật undersampling data – nghĩa là lấy số lượng 2 nhãn bằng nhau.

- Việc cho hết tất cả các thuộc tính mô tả vào mô hình không phải là cách hay. Cách hay nhất nhóm có thể đặc trưng đầu vào có hiệu quả cho mô hình là sử dụng độ tương quan, ở đồ thị heatmap bên trên ta thấy độ tương quan giữa các cột không thật sự cao nên cách này không khả thi. Nhóm chúng tôi quyết định sử dụng kỹ thuật feature selection để chọn lọc các đặc trưng tốt với mục đích tăng độ chính xác của mô hình lên cao nhất có thể.

```
# cân bằng dữ liệu
df['target'].value_counts()

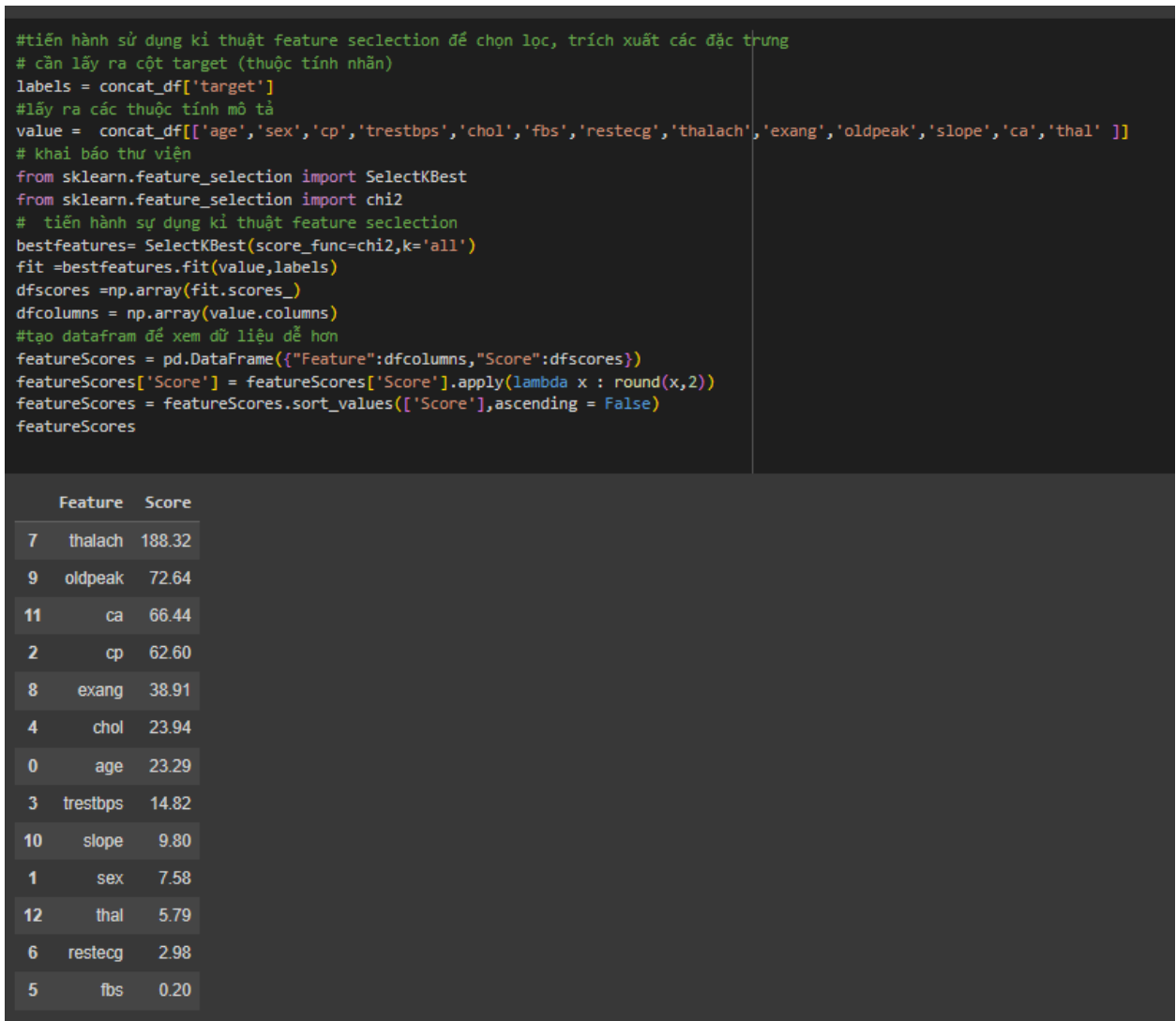
1    165
0    138
Name: target, dtype: int64

# lọc ra 2 giá trị của cột target
df_class_0 = df[df['target']==0]
df_class_1 = df[df['target']==1]

#tiến hành lấy 2 mẫu 1 và 0 đều giống nhau -138
df_class_0_under = df_class_0.sample(138)
# sau khi lấy mẫu bằng nhau , tiến hành hợp nhất dữ liệu
concat_df = pd.concat([df_class_1,df_class_0_under])
#hiện thị vài dòng dữ liệu để xem kết quả khi hợp nhất
concat_df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Hình 3. 9 Dòng kết quả khi hợp nhất dữ liệu



Hình 3. 10 Trích xuất các đặc trưng bằng cách sử dụng kĩ thuật feature selection

Nhận xét: Sau khi feature selection ta có thể thấy các cột: thalach, oldpeak, ca, cp cao hơn các cột còn lại nên nhóm quyết định chọn cột đó.

3.4 Tiến hành xây dựng các mô hình

Logitis Regression: Mô hình có hàm kích hoạt là Sigmoid trả ra giá trị 0-1. Đây là mô hình thường được nhắc trong các bài toán binary classification.

Decision Tree Classifier: Mô hình cây quyết định là một mô hình sử dụng khá phổ biến và hiệu quả của hai bài toán phân lớp và dự đoán của học có giám sát. Khác

với những kĩ thuật toán học khác của học có giám sát, mô hình cây quyết định không tồn tại phương trình dự báo. Mọi việc chúng ta cần thực hiện đó là tìm ra một cây quyết định dự báo tốt trên tập huấn luyện và sử dụng cây quyết định và dự đoán trên tập kiểm tra.

Random Forest Classifier: Ở thuật toán Random Forest ta sẽ xây dựng nhiều cây quyết định bằng thuật toán Decision Tree, tuy nhiên mỗi cây quyết định sẽ khác nhau. Sau đó kết quả dự đoán sẽ được tổng hợp từ các cây quyết định.

Kneighbors Classifier: KNN được cho là thuật toán đơn giản nhất trong máy học. Mô hình xây dựng được xây dựng chỉ bao gồm việc lưu trữ dữ liệu tập huấn (training dataset). Để dự đoán một điểm dữ liệu mới, thuật toán sẽ tìm ra những láng giềng trong tập huấn, đó là láng giềng (nearest neighbors).

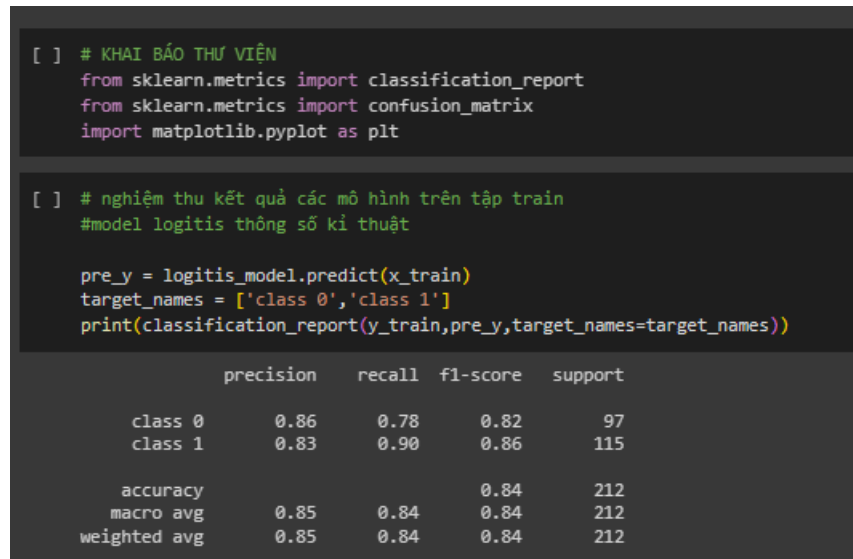
```
#khai báo thư viện
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# chia tập dữ liệu thành 2 tập train và test với tỉ lệ 7:3
x = concat_df[['thalach','oldpeak','cp','ca']]
y = concat_df['target']
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.3,random_state=42)
#để random_state - trộn dữ liệu lên tránh trường hợp overfitting

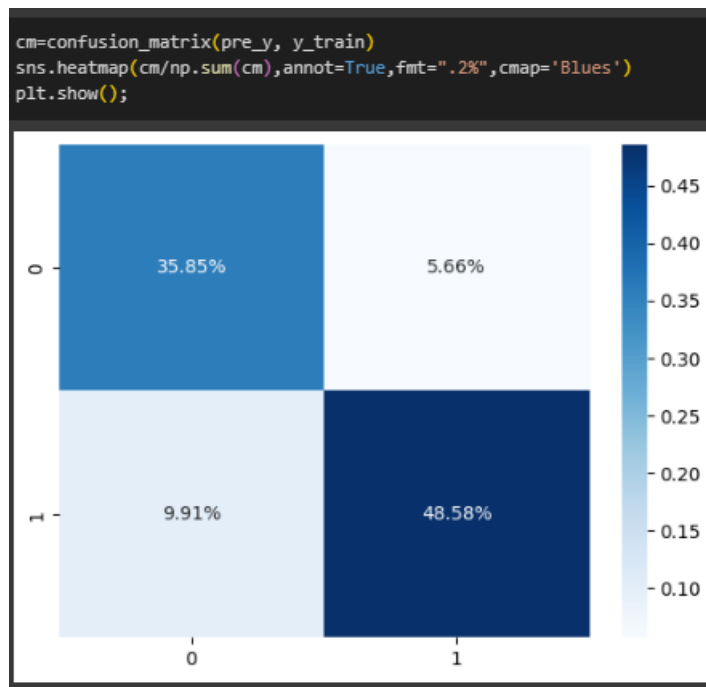
#train các mô hình
logitis_model = LogisticRegression().fit(x_train, y_train)
de_model = DecisionTreeClassifier().fit(x_train,y_train)
random_model = RandomForestClassifier().fit(x_train,y_train)
knn_model = KNeighborsClassifier().fit(x_train,y_train)
```

Hình 3. 11 Train model cho tập dữ liệu

3.5 Nghiệm thu kết quả



Hình 3. 12 Nghiệm thu kết quả các mô hình trên tập train



Hình 3. 13 Biểu đồ heatmap của logitis

```
import sklearn
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_train,pre_y)
print('độ chính xác của logitis là: ',ac*100,'%')

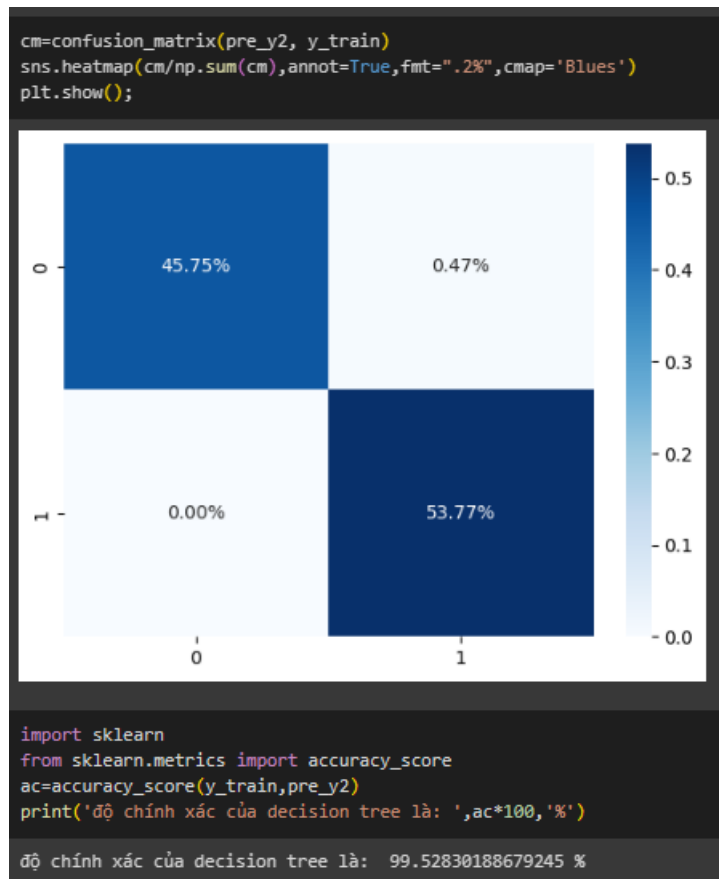
độ chính xác của logitis là: 84.43396226415094 %
```

Hình 3. 14 Độ chính xác của logitis

```
#model decisiontree thông số kĩ thuật
pre_y2=de_model.predict(x_train)
target_names = ['class 0','class 1']
print(classification_report(y_train,pre_y2,target_names=target_names))
```

	precision	recall	f1-score	support
class 0	0.99	1.00	0.99	97
class 1	1.00	0.99	1.00	115
accuracy			1.00	212
macro avg	0.99	1.00	1.00	212
weighted avg	1.00	1.00	1.00	212

Hình 3. 15 Model Decision Tree thông số kĩ thuật



Hình 3. 16 Độ chính xác của Decision Tree

```
#model random thông số kĩ thuật
pre_y4 = random_model.predict(x_train)
target_names = ['class 0','class 1']
print(classification_report(y_train,pre_y4,target_names=target_names))
```

	precision	recall	f1-score	support
class 0	1.00	0.99	0.99	97
class 1	0.99	1.00	1.00	115
accuracy			1.00	212
macro avg	1.00	0.99	1.00	212
weighted avg	1.00	1.00	1.00	212

Hình 3. 17 Model Random và thông số kĩ thuật



Hình 3. 18 Confuse Matrix của Random model

```
import sklearn
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_train,pre_y4)
print('độ chính xác của randomforest là: ',ac*100,'%')

độ chính xác của randomforest là: 99.52830188679245 %
```

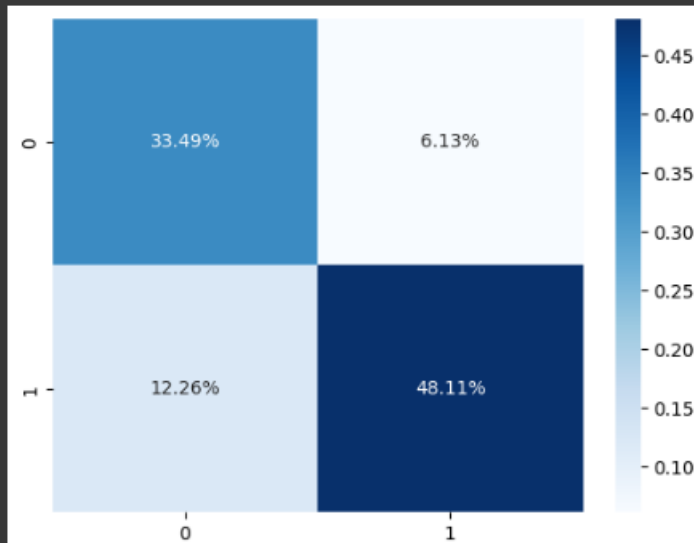
Hình 3. 19 Độ chính xác của Random Forest

```
#model knn thông số kĩ thuật
pre_y5=knn_model.predict(x_train)
target_names= ['class 0','class 1']
print(classification_report(y_train,pre_y5,target_names=target_names))
```

	precision	recall	f1-score	support
class 0	0.85	0.73	0.78	97
class 1	0.80	0.89	0.84	115
accuracy			0.82	212
macro avg	0.82	0.81	0.81	212
weighted avg	0.82	0.82	0.81	212

Hình 3. 20 Model KNN thông số kĩ thuật

```
#confuse matrix của knn
cm=confusion_matrix(pre_y5, y_train)
sns.heatmap(cm/np.sum(cm),annot=True,fmt=".2%", cmap='Blues')
plt.show();
```



Hình 3. 21 Confuse matrix của KNN

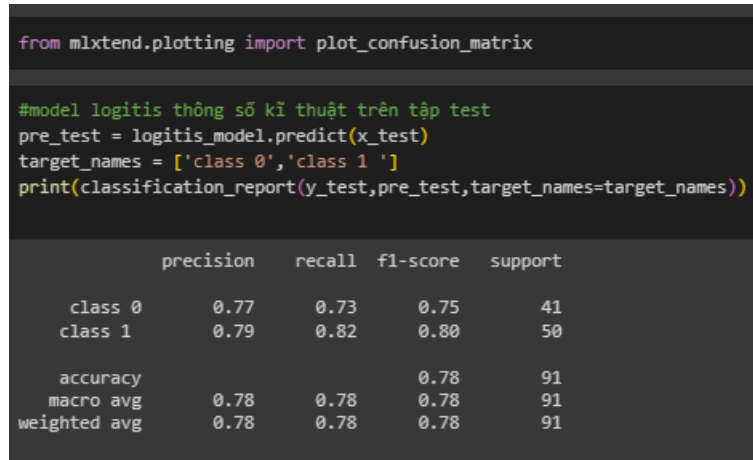
```
import sklearn
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_train,pre_y5)
print('độ chính xác của KNeighbors là: ',ac*100,'%')

độ chính xác của KNeighbors là: 81.60377358490565 %
```

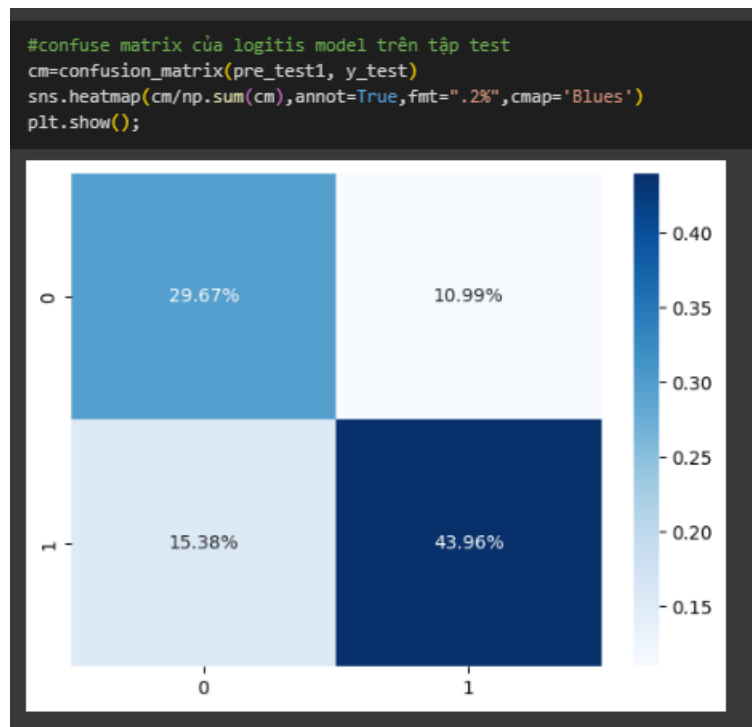
Hình 3. 22 Độ chính xác của KNeighbors

Nhận xét: Đối với tập train các mô hình tương đối cao đặc biệt là decision tree và random forest có kết quả gần như tuyệt đối.

3.6 Sử dụng mô hình đối với tập test và đánh giá kết quả



Hình 3. 23 Model Logitis thông số kĩ thuật trên tập test



Hình 3. 24 Confuse matrix của logitis model trên tập test

```
import sklearn
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_test,pre_test1)
print('độ chính xác của logitis là: ',ac*100,'%')

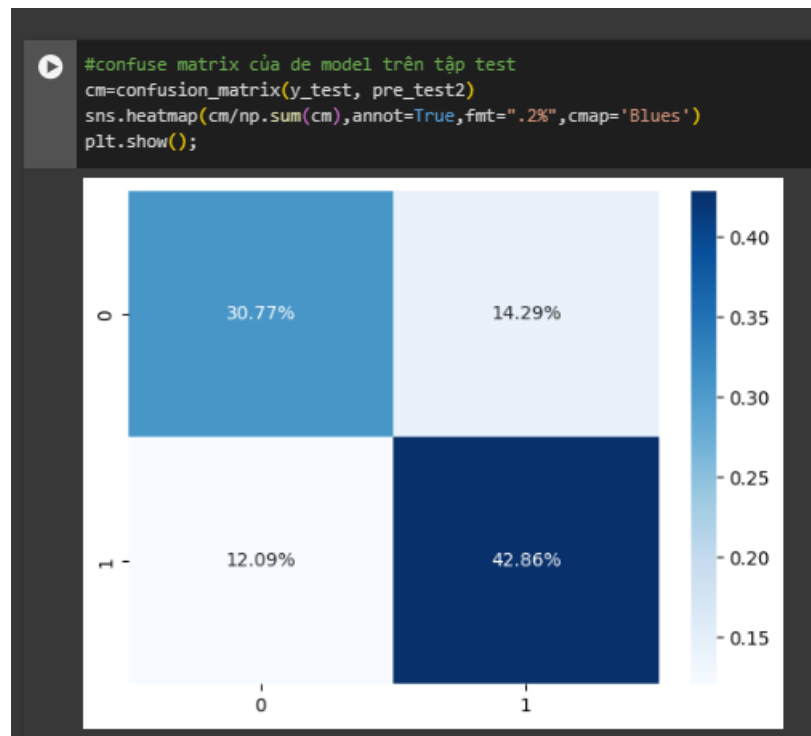
độ chính xác của logitis là: 73.62637362637363 %
```

Hình 3. 25 Độ chính xác của logitis

```
#model de thông số kĩ thuật trên tập test
pre_test2 = de_model.predict(x_test)
target_names = ['class 0','class 1 ']
print(classification_report(y_test,pre_test2,target_names=target_names))
```

	precision	recall	f1-score	support
class 0	0.72	0.68	0.70	41
class 1	0.75	0.78	0.76	50
accuracy			0.74	91
macro avg	0.73	0.73	0.73	91
weighted avg	0.74	0.74	0.74	91

Hình 3. 26 Model decision tree thông số kĩ thuật trên tập test



Hình 3. 27 Confuse matrix của Decision Tree trên tập test


```
import sklearn
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_test,pre_test2)
print('độ chính xác của decisiontree là: ',ac*100,'%')

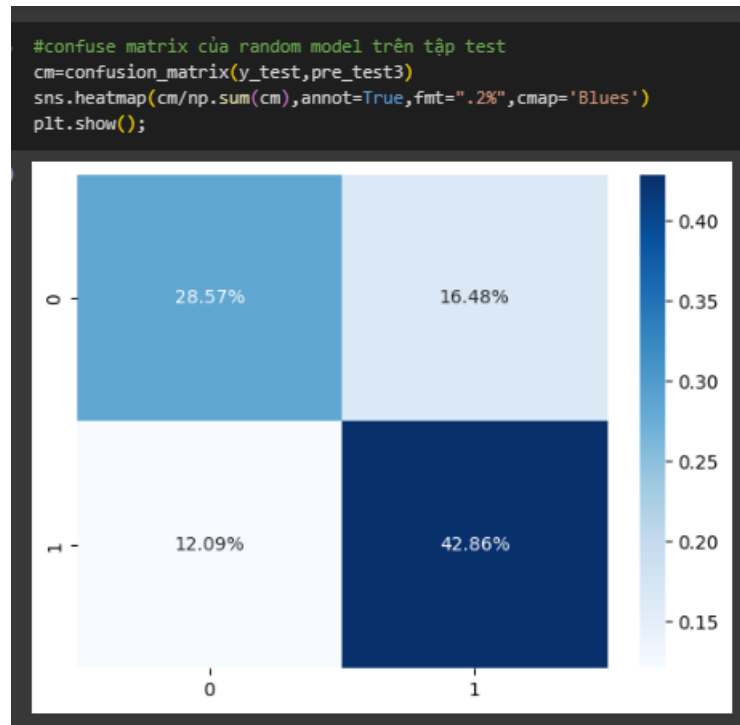
độ chính xác của decisiontree là: 73.62637362637363 %
```

Hình 3. 28 Độ chính xác của Decision Tree

```
#model random thông số kĩ thuật trên tập test
pre_test3= random_model.predict(x_test)
target_names=['class 0', 'class 1']
print(classification_report(y_test,pre_test3,target_names=target_names))
```

	precision	recall	f1-score	support
class 0	0.70	0.63	0.67	41
class 1	0.72	0.78	0.75	50
accuracy			0.71	91
macro avg	0.71	0.71	0.71	91
weighted avg	0.71	0.71	0.71	91

Hình 3. 29 Model Random thông số kĩ thuật trên tập test



Hình 3. 30 Confuse Matrix của Random Model trên tập test

```
import sklearn
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_test,pre_test3)
print('độ chính xác của randomforest là: ',ac*100,'%')

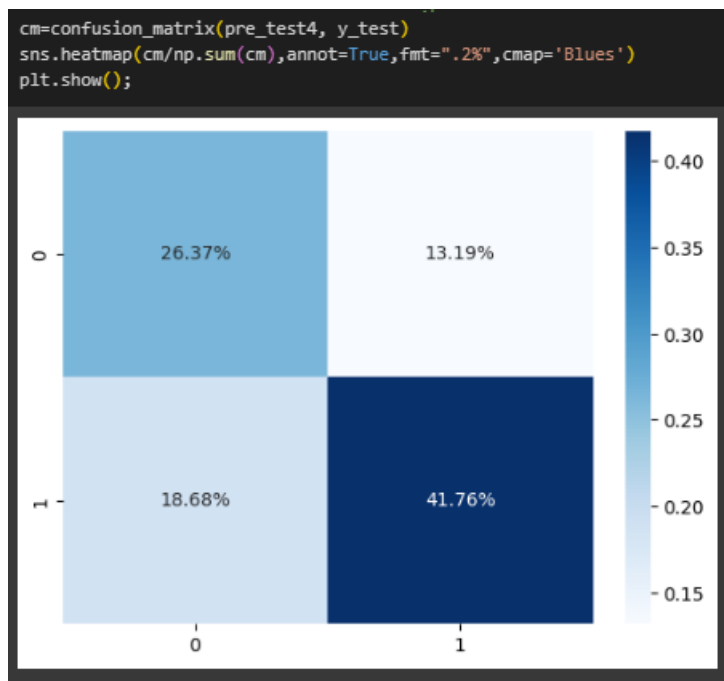
độ chính xác của randomforest là: 71.42857142857143 %
```

Hình 3. 31 Độ chính xác của Random Forest trên tập test

```
#model nn thông số kĩ thuật trên tập test
pre_test4 = knn_model.predict(x_test)
target_names = ['class 0','class 1 ']
print(classification_report(y_test,pre_test4,target_names=target_names))
```

	precision	recall	f1-score	support
class 0	0.67	0.59	0.62	41
class 1	0.69	0.76	0.72	50
accuracy			0.68	91
macro avg	0.68	0.67	0.67	91
weighted avg	0.68	0.68	0.68	91

Hình 3. 32 Model KNN thông số kĩ thuật trên tập test



Hình 3. 33 Confuse Matrix của KNN trên tập test

```
import sklearn
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_test,pre_test4)
print('độ chính xác của kneight là: ',ac*100,'%')
độ chính xác của kneight là: 68.13186813186813 %
```

Hình 3. 34 Độ chính xác của kneight

CHƯƠNG 4: KẾT LUẬN

Trong báo cáo đồ án chúng tôi đã tìm hiểu cơ sở lý thuyết một số thuật toán và thuật toán Random Forest. Ta có thể thấy Random Forest là một thuật toán hiệu quả, chính xác cao, cần được đầu tư phát triển cho các ứng dụng thực tiễn và đời sống. Có thể sau này có thể sẽ có cải thiện thuật toán bằng nhiều phương pháp khác và chúng tôi sẽ thấy được điều đó sau này.

Thông qua những nghiên cứu về thuật toán Random Forest và ứng dụng, cho thấy Random Forest là thuật toán hiệu quả, chính xác, cần phát triển mạnh để áp dụng nhiều vào trong đời sống. Bản thân chúng tôi đã hoàn thành bài báo cáo đồ án môn học này tuy nhiên vẫn còn nhiều thiếu sót, cần đào sâu, nghiên cứu và trau dồi nhiều kiến thức hơn.

TÀI LIỆU THAM KHẢO

- [1] "wikipedia," 23 20 2023. [Online]. Available: https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_m%C3%A1y. [Accessed 15 12 2023].
- [2] T. V. Hào, "123docz.net," [Online]. Available: <https://123docz.net/document/9898782-bao-cao-mon-lap-trinh-python-cho-may-hoc-random-forest-classifier.htm>. [Accessed 1 12 2023].
- [3] T. Mai, "TinoGroup," [Online]. Available: <https://tino.org/vi/machine-learning-la-gi/>. [Accessed 24 12 2023].
- [4] "wikipedia," 3 1 2023. [Online]. Available: https://vi.wikipedia.org/wiki/C%C3%A2y_quy%E1%BA%BFt_%C4%91%E1%BB%8Bnh. [Accessed 24 12 2023].
- [5] "iSolution," [Online]. Available: <https://isolution.pro/vi/t/machine-learning-with-python/classification-algorithms-random-forest/thuat-toan-phan-loai-rung-ngau-nhien>. [Accessed 24 12 2023].
- [6] "Trí Tuệ Nhân Tạo.io," 06 06 2019. [Online]. Available: <https://trituenhantao.io/kien-thuc/decision-tree/>. [Accessed 25 12 2023].
- [7] Simplilearn, "Simplilearn," 7 10 2023. [Online]. Available: <https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm>. [Accessed 25 12 2023].
- [8] H. Nguyễn, "pdfcoffee," [Online]. Available: <https://pdfcoffee.com/random-forest-pdf-free.html>. [Accessed 27 12 2023].
- [9] S. Sharoon, "Analytics Vidhya," 25 08 2023. [Online]. Available: <https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/?fbclid=IwAR3Ncz7qVKtqtK6WwUwPeNpJpWZ74xigGVzY8aUmEieILRd1Sy9u5>. [Accessed 29 12 2023].
- [10] T. Srivastava, "Analytics Vidhya," 22 08 2023. [Online]. Available: <https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/>. [Accessed 29 12 2023].

- [11] "QUANTSTART," [Online]. Available: <https://www.quantstart.com/articles/bootstrap-aggregation-random-forests-and-boosted-trees/>. [Accessed 29 12 2023].
- [12] "Vinmec International Hospital," [Online]. Available: <https://www.vinmec.com/vi/tim-mach/thong-tin-suc-khoe/nguyen-nhan-va-trieu-chung-cua--benh-tim-mach/>. [Accessed 30 12 2023].
- [13] T. h. M. Learning, "Youtube," 31 10 2022. [Online]. Available: <https://www.youtube.com/watch?v=gzrwwbdi80Y&t=1279s>. [Accessed 30 12 2023].
- [14] noeffortnomoney, "github," 14 12 2021. [Online]. Available: <https://github.com/noeffortnomoney/CS116.M12.KHCL/tree/main/Project>. [Accessed 30 12 2023].
- [15] a. rezaei, "kaggle," 07 2023. [Online]. Available: <https://www.kaggle.com/datasets/arezaei81/heartcsv>. [Accessed 31 10 2023].
- [16] Đ. T. Hoàng, "Youtube," 18 06 2022. [Online]. Available: <https://www.youtube.com/watch?v=HrnH2UylyfM&t=305s>. [Accessed 30 12 2023].