|  |  |
| --- | --- |
| **logo moi** | BỘ GIÁO DỤC VÀ ĐÀO TẠO  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM** |

**BÁO CÁO TRÍ TUỆ NHÂN TẠO**

**TÌM HIỂU VỀ RANDOM FOREST VÀ ÁP DỤNG VÀO BÀI TOÁN DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH TIM MẠCH**

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 Mechine 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 kĩ 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 kĩnh 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**

Table of Contents

[**LỜI MỞ ĐẦU** 2](#_Toc154740858)

[**LỜI CẢM ƠN** 3](#_Toc154740859)

[**NHẬN XÉT & ĐÁNH GIÁ CỦA GIẢNG VIÊN HƯỚNG DẪN** 4](#_Toc154740860)

[**MỤC LỤC** 5](#_Toc154740861)

[**DANH MỤC HÌNH ẢNH** 8](#_Toc154740862)

[**CHƯƠNG 1: TỔNG QUAN** 9](#_Toc154740863)

[**1.1** **Giới thiệu về Machine Learning** 9](#_Toc154740864)

[**1.2** **Giới thiệu về Classification** 10](#_Toc154740865)

[**1.3** **Giới thiệu sơ lược Decision Tree** 11](#_Toc154740866)

[**1.4** **Giới thiệu về thuật toán Random Forest** 12](#_Toc154740867)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT** 14](#_Toc154740868)

[**2.1** **Decision Tree** 14](#_Toc154740869)

[**2.1.1** **Định nghĩa** 14](#_Toc154740870)

[**2.1.2** **Một số thuật toán xây dựng Decision Tree** 14](#_Toc154740871)

[**2.1.3** **Đặc điểm của Decision Tree** 19](#_Toc154740872)

[**2.2** **Thuật Toán Random Forest** 20](#_Toc154740873)

[**2.2.1** **Định nghĩa** 20](#_Toc154740874)

[**2.2.2** **Mô hình thuật toán** 20](#_Toc154740875)

[**2.2.3** **Thuật toán** 22](#_Toc154740876)

[**2.2.4** **Ví dụ minh họa** 22](#_Toc154740877)

[**2.2.5** **Đặc điểm của Random Forest** 22](#_Toc154740878)

[**2.2.6** **Ứng dụng của thuật toán Random Forest** 23](#_Toc154740879)

[**2.3** **Phương pháp lấy mẫu Bootstrap** 23](#_Toc154740880)

[**2.3.1** **Giới thiệu Bootstrap** 23](#_Toc154740881)

[**2.3.2** **Thuật toán Bootstrap** 23](#_Toc154740882)

[**2.3.3** **Đặc điểm của Bootstrap** 25](#_Toc154740883)

[**2.3.4** **Bootstrap trong Random Forest** 25](#_Toc154740884)

[**2.4** **Tham số và siêu tham số trong model Random Forest** 26](#_Toc154740885)

[**3.1** **Tham số (Parameter)** 26](#_Toc154740886)

[**3.2** **Siêu tham số (Hyperparameter)** 26](#_Toc154740887)

[**CHƯƠNG 3: BÀI TOÁN DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH TIM MẠCH** 36](#_Toc154740888)

[**CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 37](#_Toc154740889)

[**TÀI LIỆU THAM KHẢO** 38](#_Toc154740890)

[References 38](#_Toc154740891)

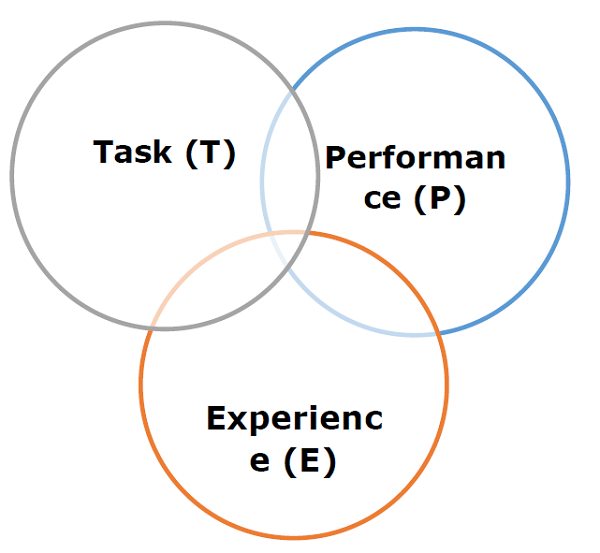
# **DANH MỤC HÌNH ẢNH**

# **CHƯƠNG 1: TỔNG QUAN**

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-supervidsed learning), học tặng 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 nhu: 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

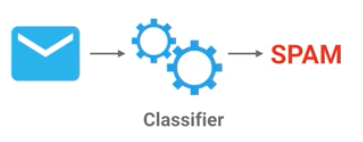
Machine Learning hiện nay được áp dụng rộng rãi và trong nhiều lĩnh vực khác nhau bao gồm máy truy tìm dữ liệu, chuẩn đoán y khoa, phát hiện thẻ tín dụng giả, phân tích thị trường chứng khoán, phân loại chuỗi DNA, nhận dạng tiếng nói và chữ vieetsm dịch tự động, trò chơi điện tử, cử động rô-bốt (robot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

Phân loại 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

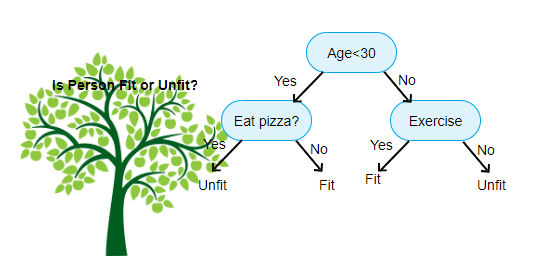
Có một loạt các thuật toán học máy để phân loại trong trong học máy: Naïve Bayes Classifier, Stochastic Gradient Descent, K-Nearest Neighbor, Decision Tree, Random forest,… Mỗi loại thuật toán sẽ có ưu nhược điểm riêng nhưng trong đồ án này chúng ta sẽ tập trung vào Random Forest sẽ được đề cập chương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

1. **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. **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 dụng một tập hợp các Decision Tree với phương sai có kiểm soát.

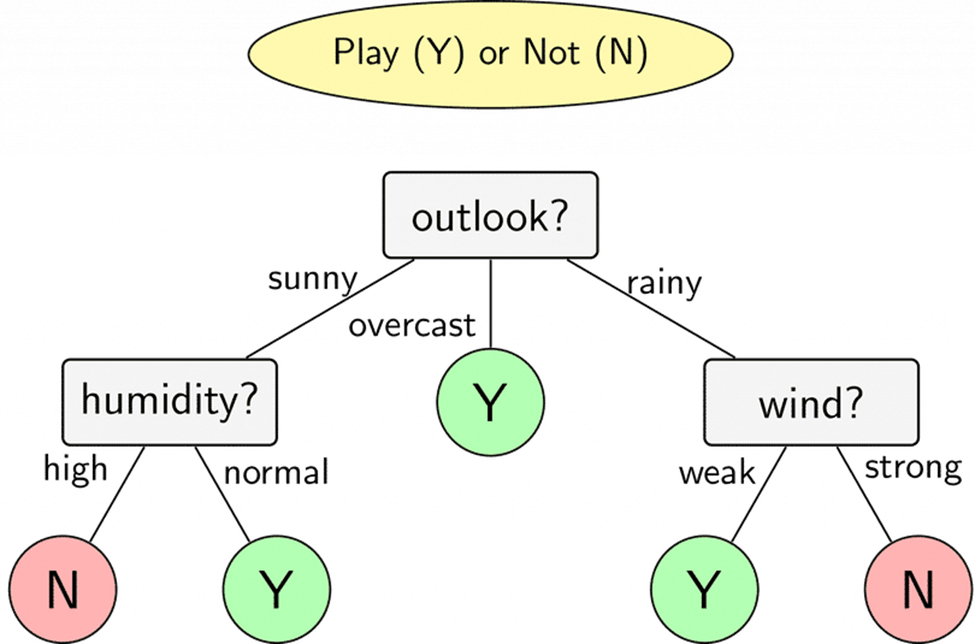
Random Forest được mô hình hóa như tập các cây phân lớp. Tuy nhiên Random Forest sử dụng các mẫu ngẫu nhiên cho các cây cũng như việc chọn lựa các thuộc tính ngẫu nhiên khi phân chia cây. Thuật toán Random Forest tỏ ra chính xác và nhanh hơn khi huấn luyện trên không gian dữ liệu lớn với nhiều thuộc 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 kĩ hơn về Random Forest ở chương 2.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

1. **Decision Tree**
2. **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 đinh, hay chỉ với cái tên ngắn gọn là cây quyết định.

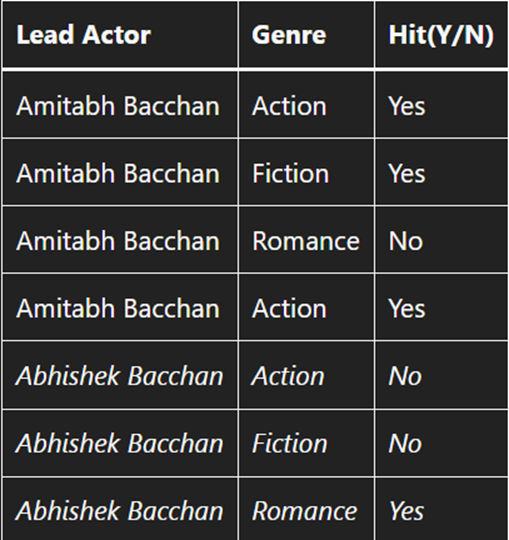


Hình 2.1. Mô hình Decision Tree quyết định đi chơi

1. **Một số thuật toán xây dựng Decision Tree**

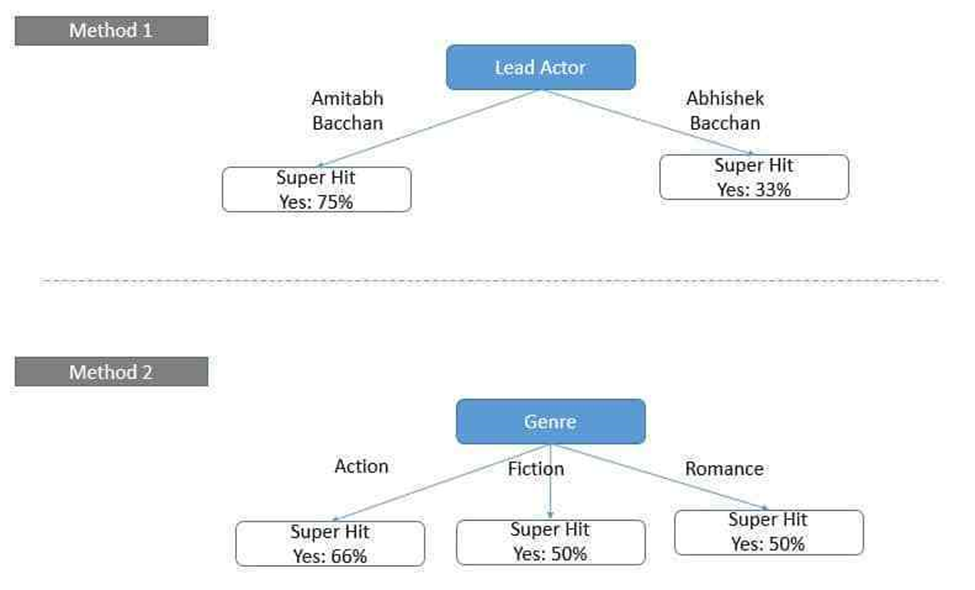
Thuật toán ID3(J.R.Quinlan 1933) sử dụng phương pháp tham lam tìm kiếm từ trên xuống thông qua không gian của các nhánh có thể không có backtracking. ID3 sử dụng Entropy và Information Gain để xây dựng Decision Tree.

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.



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)

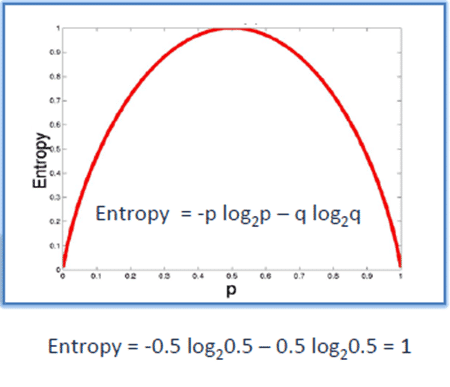
Entropy trong Decision Tree:

* 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 (-log2p) 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]
  + Entropy (S) = 0 ⬄ tập mẫu S chỉ toàn mẫu thuộc cùng lớp hoặc S là thuần nhất
  + Entropy(S) = 1 ⬄ tập mẫu S có các mẫu thuộc các lớp khác nhau với độ pha trộn là cao nhất
  + 0 < Entropy(S) < 1 ⬄ 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:

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

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



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: pi = 0 hoặc pi = 1
  + P vẩn đục: pi = 0.5, khi đó hàm Entropy đạt đỉnh cao nhất

Information Gain trong Decision Tree:

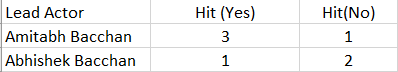
* Thuộc tính A gồm các giá trị {a1, a2,…,av}, và phân chia D thành v tập con {D1, D2,…,Dv}. Thông tin cần thiết để chia D theo A.
* Information Gain dựa trên phân chia theo thuộc tính A của tập D

Gain(A) = Info(D) – InfoA(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”



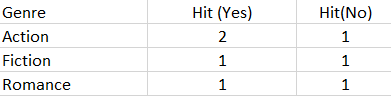
* + - * |D| = 7, n = 2, C1 = “Yes”, C2 = “No”
      * |C1,D| = 4, |C2,D| = 3
      * Entropy để phân lớp một mẫu trong D là

Info(D) = log2 log2 ≈ 0.99

* + - Tính Information Gain đối với “Lead Actor”
      * InfoLead Actor(D) = log2 log2 ) +

log2 log2 ) ≈ 0.69

* + - * Gain (Lead Actor) = Info(D) – InfoLead Actor(D) ≈ 0.99 -0.69 ≈ 0.3
    - Tính Information Gain đối với “Genre”



* + - * InfoGenre(D) = log2 log2 ) +

log2 log2 ) +

log2 log2 ) ≈ 0.55

* + - * GainGenre(D) = Info(D) - InfoGenre = 0.99 – 0.55 ≈ 0.44

🡺Chọn “Genre” làm thuộc tính phân hạch 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

GainRatio(A) = với

* Chọn thuộc tính với độ đo Gain Ratio lớn nhất là thuộc tính phân hoạch áp dụng với ví dụ trên ta có

SplitInfoGenre(D) = ≈ 0.98

* GainRatio(Lead Actor) = ≈ 0.58

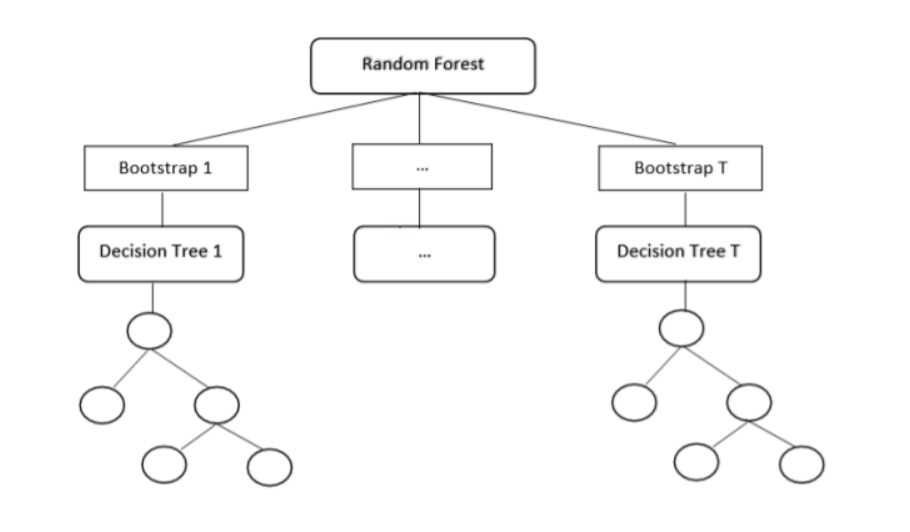
1. **Đặc điểm của Decision Tree**

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| 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:   * Mô hình sinh ra các quy tắc dễ hiểu cho người đọc tạo ra bộ luật với mỗi nhánh lá là một luật của cây. * 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. | Kèm với đó, cây quyết định cũng có những nhược điểm:   * Mô hình cây quyết định phụ thuộc rất lớn vào dữ liệu của bạn. Thâm chí với một sự thay đổi nhỏ trong toàn bộ dữ liệu, cấu trúc mô hình cây quyết định thay đổi hoàn toàn. * Cây quyết định hay gặp vấn đề overfitting. |

1. **Thuật Toán Random Forest**
2. **Định nghĩa**

Random Forests hay Random Decision Forests là thuật toán học máy dựa trên kĩ 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.

1. **Mô hình thuật toán**



741Hình 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 l\ iệu: 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 voting để đưa ra dự đoán cuối cùng dựa vào kết quả tính trung bình của trọng số.c/963.

1. **Thuật toán**

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

1. **Ví dụ minh họa**

1. **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 bang 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.

1. **Ứng dụng của thuật toán Random Forest**

Được ứng dụng chủ yếu vào bốn lĩnh vực sau:

* Ngân hàng: Xác định rủi roc ho 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.

1. **Phương pháp lấy mẫu Bootstrap**
2. **Giới thiệu Bootstrap**

Boostrap 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 Boostrap. 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.

1. **Thuật toán Bootstrap**

Thuật toán Bootstrap lấy mẫu ngẫu nhiên có hoàn lại

1. 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

1. 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.



*Hình . Hình ảnh về tập dữ liệu minh họa cho phương pháp lấy mẫu bootstrap*

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 diểm dữ liệu có khả năng được chọn một lần.

Mẫu dữ liệu Boostrap 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 overfiting. 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.

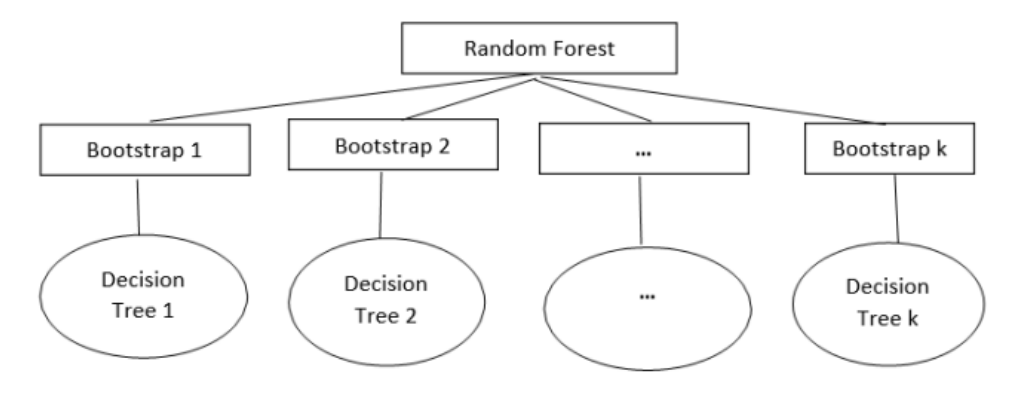
1. **Đặc điểm của Bootstrap**

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 cây cũng như xác suất của số trung bình dựa trên một mẫu duy nhất.

1. **Bootstrap trong Random Forest**

Trong Random Forest thuật toán Boostrap 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 . Bootstrap trong Random Forest*

1. **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

1. **Tham số (Parameter)**
   1. **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.

* 1. **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.

* 1. **oob\_score**

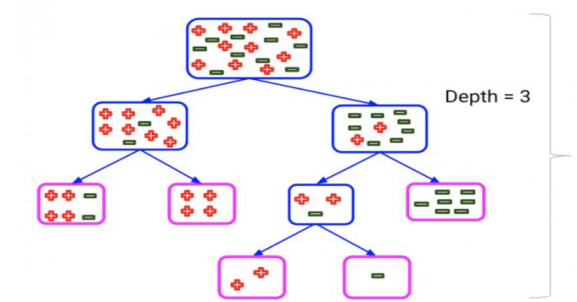
Đây là phương pháp xác nhận chéo rừng ngẫu nhiên. Nó rất giống kĩ thuật xác nhận, tuy nhiên, điều này nhanh hơn rất nhiều. Phương pháp này chỉ cần gắn thẻ cho mọi quan sát được sử dụng trong các stress khác nhau. Và sau đó, nó tìm ra điểm bầu chọn tối đa cho mọi quan sát chỉ dựa trên những cây không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

1. **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.

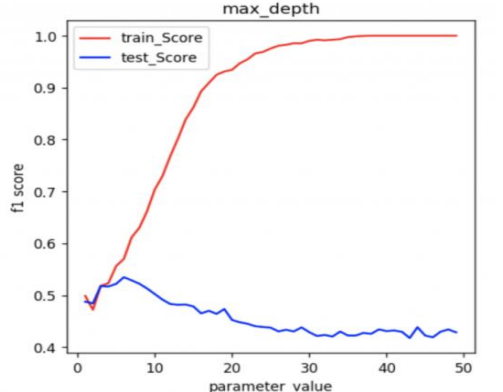
1. **max\_depth**

Các *max\_depth* của một cây trong Random Forest được đỉnh nghĩa là con đường dài nhất giữa nút gốc và nút lá:



Hình .*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 .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.

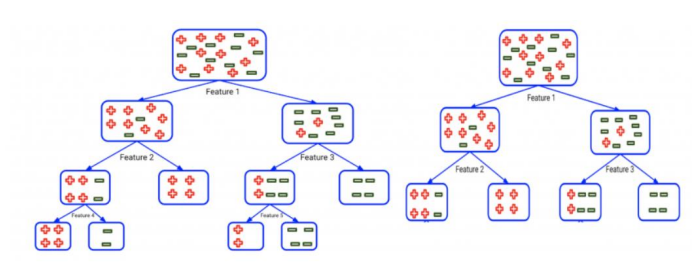
Cây bắt đầu trang bị quá mức cho tập huấn luyện và do đó không thể tổng quát hóa các điểm không nhìn thấy trong tập thử nghiệm.

1. **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.

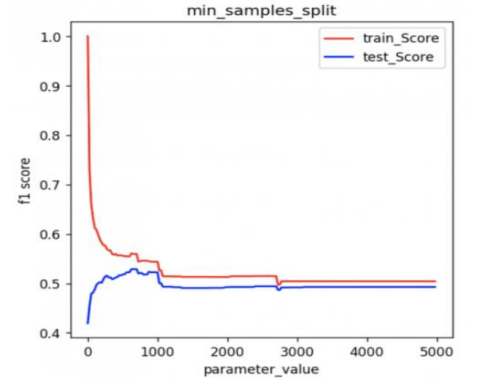
Việc có giá trị mặc định là 2 đặt ra vấn đề rằng một cây thường được tiếp tục lách cho đến khi các nút hoàn toàn thuần khiết. Kết quả là cây phát triển về kích thước và do đó làm quá tải dữ liệu.



*Hình . 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 .Sơ đồ hyperparameter value*

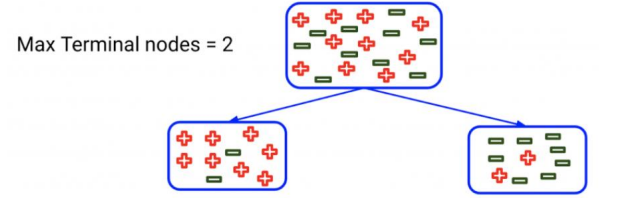
Khi tăng giá trị của siêu tham số min\_sample\_split, chúng ta có thể thấy rõ rằng đối với giá trị nhỏ của tham số, có sự khác biệt đáng kể giữa điểm đào tạo và điểm kiểm tra. Nhưng khi giá trị của tham số tăng lên, sự khác biệt giữa điểm đào tạo và điểm kiểm tra sẽ giảm.

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.

1. **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.

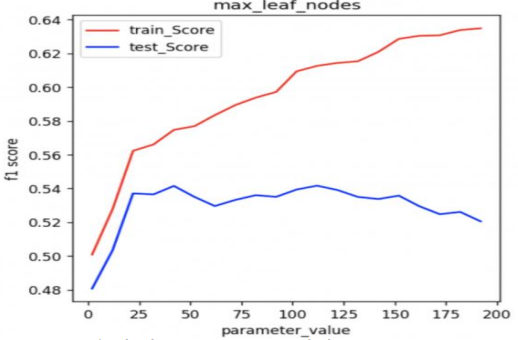
Giả sử chúng ta đặt các nút đầu cuối tối đa là 2 trong trường hợp này. Vì chỉ có một nút, nó sẽ cho phép cây phát triển thêm.



*Hình . 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:



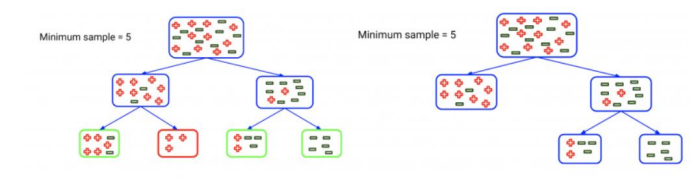
*Hình .Sơ đồ hyperparameter value*

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.

1. **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:

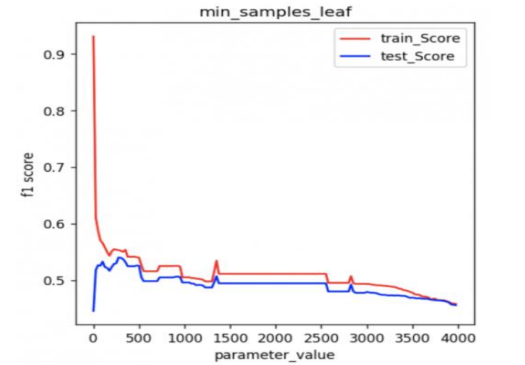


*Hình . Sơ đồ min\_sample\_leaf*

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 lí 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.



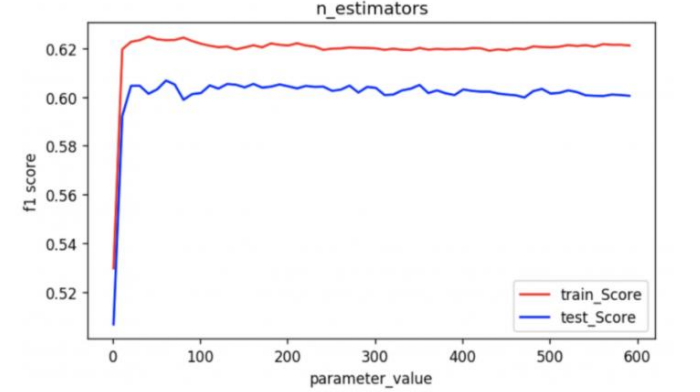
*Hình . Sơ đồ parameter value*

Chúng ta có thể thấy rõ rằng mô hình Random Forest đang overfiting 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 overfiting(100 < 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.

1. **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 thồi 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.



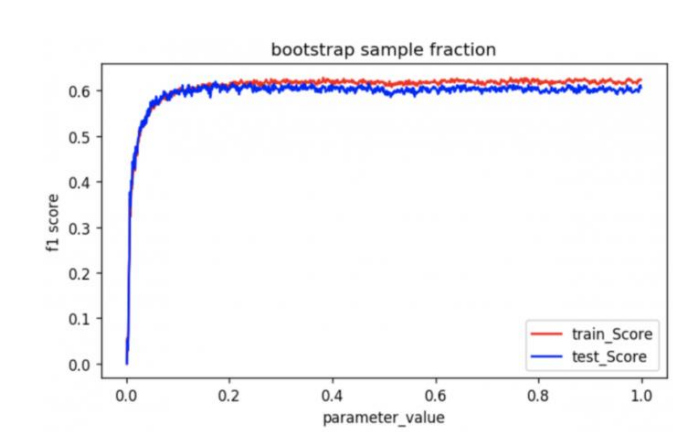
*Hình . Mô hình n\_estimators và hyper parameter để thấy rõ hiệu suất và mức độ trì trệ khi tăng số lượng cây lên*

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, những 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.

1. **max\_samples**

Các max\_samples hyperparameter xác được 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 . Mô hình max\_samples*

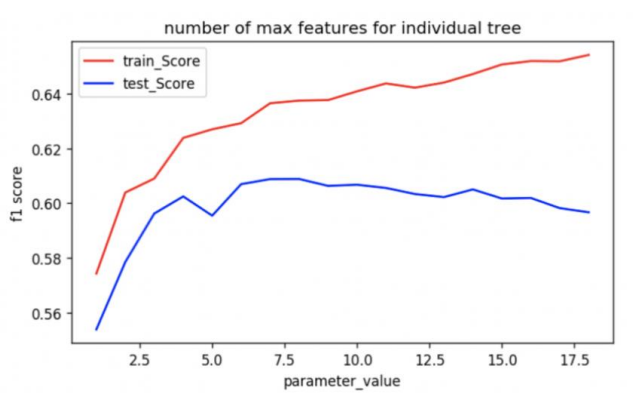
Có thể nhìn thấy được hiệu suất của mô hình tăng mạnh và sau đó bão hòa rất nhanh. Vì vậy không nhất thiết phải cung cấp cho mỗi decision tree của random forest toàn bộ dữ liệu. Theo mô hình trên có thể nhận thấy được hiệu suất mô hình đạt mức tối đa khi dữ liệu được cung cấp nhỏ hơn 0.2 phần của dữ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

1. **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 nhiện một 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 . 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 cảu 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\_feautes lý tưởng thường có xu hướng nằm gần với giá trị này.

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

1. **Giới thiệu bài toán**
2. **Tập dữ liệu**
3. **Tiền xử lý dữ liệu**
4. **Train model**
5. **Thực hiện tham số cho bài toán**
6. **Test dự đoán**
7. **So sánh model với một số model khác**

# **CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

# **TÀI LIỆU THAM KHẢO**

# References

|  |  |
| --- | --- |
| [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, "Analystics 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]. |