

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**UIT**  
TRƯỜNG ĐẠI HỌC  
CÔNG NGHỆ THÔNG TIN

# **BÁO CÁO ĐỒ ÁN**

**MÔN HỌC: LẬP TRÌNH PYTHON CHO MÁY HỌC**

**ĐỀ TÀI**

**TÌM HIỂU VỀ THUẬT TOÁN RANDOM FOREST  
CLASSIFIER**

**Giảng viên hướng dẫn:** TS. Nguyễn Vinh Tiệp

**Sinh viên thực hiện:** Trương Quốc Bình - 19521270  
Nguyễn Hữu Hưng – 19521571  
Nguyễn Quan Huy - 19521622

**Lớp:** CS116.M12.KHCL

*Thành phố Hồ Chí Minh, tháng 12 năm 2021*

## MỤC LỤC

<b>LỜI MỞ ĐẦU.....</b>	<b>5</b>
<b>CHƯƠNG 1. TỔNG QUAN .....</b>	<b>6</b>
1.1. Giới thiệu Machine Learning.....	6
1.2. Giới thiệu về Classification .....	6
1.3. Giới thiệu sơ lược Decision Tree .....	7
1.4. Giới thiệu thuật toán Random Forest.....	7
<b>CHƯƠNG 2. CƠ SỞ LÝ THUYẾT .....</b>	<b>8</b>
<b>2.1. Decision Tree .....</b>	<b>8</b>
2.1.1. Giới thiệu.....	8
2.1.2. Một số thuật toán xây dựng Decision Tree .....	9
2.1.3. Đặc điểm của Decision Tree.....	9
<b>2.2. Thuật toán Random Forest .....</b>	<b>9</b>
2.2.1. Định nghĩa .....	9
2.2.2. Mô hình thuật toán.....	10
2.2.3. Thuật toán.....	10
2.2.4. Ví dụ minh họa .....	11
2.2.5. Đặc điểm của Random Forest.....	12
2.2.6. Ứng dụng của Random Forest .....	13
<b>2.3. Phương pháp lấy mẫu Bootstrap .....</b>	<b>13</b>
2.3.1. Giới thiệu Bootstrap.....	13
2.3.2. Thuật toán Bootstrap.....	14
2.3.3. Đặc điểm của Bootstrap .....	14
2.3.4. Bootstrap trong Random Forest.....	14
2.3.5. Xây dựng Classification bằng cách sử dụng Scikit-learn với thuật toán Random Forest Classifier.....	15
2.3.6. Các tính năng quan trọng trong Scikit-learn.....	17
<b>2.4. Tham số và siêu tham số trong Random Forest .....</b>	<b>20</b>
2.4.1. Tham số (Parameter).....	20
2.4.2. Siêu tham số (Hyperparameter) .....	20
<b>CHƯƠNG 3. BÀI TOÁN SỬ DỤNG RANDOM FOREST CLASSIFIER.....</b>	<b>28</b>
<b>3.1. Giới thiệu bài toán.....</b>	<b>28</b>
<b>3.2. Tập dữ liệu.....</b>	<b>28</b>

<b>3.3. Tiền xử lí dữ liệu.....</b>	<b>30</b>
<b>3.4. Train model .....</b>	<b>32</b>
<b>3.5. Thực nghiệm siêu tham số cho bài toán.....</b>	<b>34</b>
<b>3.6. Test dự đoán .....</b>	<b>35</b>
<b>3.7. So sánh với một số model thuật toán khác .....</b>	<b>36</b>
<b>CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>37</b>
<b>4.1. Kết luận.....</b>	<b>37</b>
<b>4.1. Hướng phát triển.....</b>	<b>37</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>38</b>

## DANH MỤC BẢNG, SƠ ĐỒ, HÌNH VẼ

Hình 1: Ví dụ Decision tree dùng trong Random Forest.....	8
Hình 2: Bảng dữ liệu cho Decision tree.....	8
Hình 3: Mô hình thuật toán Random Forest.....	10
Hình 4: Tập dữ liệu minh họa thuật toán Random Forest.....	11
Hình 5: Cây phân lớp trong rừng.....	12
Hình 6: Bootstrap trong Random Forest.....	15
Hình 7: Bảng Iris Dataset.....	16
Hình 8: Feature Important Score.....	19
Hình 9,10: Sơ đồ max_depth và hyperparameter value.....	21
Hình 11,12: Sơ đồ min_sample_split và hyperparameter value.....	22
Hình 13,14: Sơ đồ max_leaf_nodes và hyperparameter value.....	23
Hình 15,16 Sơ đồ min_samples_leaf và hyperparameter value.....	24
Hình 17: Sơ đồ n_estimators.....	25
Hình 18: Sơ đồ max_samples.....	26
Hình 19: Sơ đồ max_features.....	27
Hình 20: 5 dòng dữ liệu từ Heart.csv.....	28
Hình 21,22: Biểu đồ target, sex.....	30
Hình 23,24,25,26,27,28: Biểu đồ cột các giá trị tương ứng target, sex.....	31
Hình 29: Bảng thực nghiệm siêu tham số.....	35
Hình 30,31:Bảng so sánh accuracy model và confusion matrix.....	36

## LỜI MỞ ĐẦU

Trong những năm gần đây, các lĩnh vực nghiên cứu của 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 lâu và đạt rất nhiều thành tựu. Nhiều thuật toán học máy được ứng dụng trong thực tế mang lại hiệu quả trong các lĩnh vực như: nhận dạng chữ viết tay, nhận dạng khuôn mặt, máy tìm kiếm, . . . 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 rất quan trọng và cần thiết.

Trong đồ án này, mục tiêu của nhóm chúng tôi là tìm hiểu về Random Forest Classifier, giới thiệu sơ lược về Classification, rồi tập trung kỹ vào thuật toán Random Forest về lịch sử, các bước thực hiện, tham số và siêu tham số trong Random Forest và cách điều chỉnh nó, ưu và nhược điểm của thuật toán Random forest, một số ứng dụng thực tế của thuật toán Random Forest và cuối cùng là một bài toán liên quan sử dụng thuật toán Random Forest Classifier.

Chúng tôi cũng xin chân thành cảm ơn thầy Nguyễn Vinh Tiệp đã tận tình giảng dạy chúng tôi môn học này để em có thể hoàn thành báo cáo đồ án này một cách tốt nhất.

## Chương 1. TỔNG QUAN

### 1.1 Giới thiệu Machine Learning

Machine Learning là ngành khoa học về máy tính nhằm nghiên cứu, phát triển các thuật toán, thuật giải với 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 của thuật toán. Các loại thuật toán thường sử dụng như: Học có giám sát, học không giám sát, học nửa giám sát, học tăng cường, học theo nhóm... Machine Learning 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, Decision tree, K-Mean, SVM, Random Forest. . .

Machine Learning có hiện nay được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như: máy truy tìm dữ liệu, chuẩn đoán y khoa, phát hiện thẻ tín dụng giả, phân tích thị trường chứng khoán, phân loại các chuỗi DNA, nhận dạng tiếng nói và chữ viết, dịch tự động, chơi trò chơi và cử động rô-bốt. . .

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

### 1.2. Giới thiệu về Classification

Phân loại trong học máy và thống kê là một phương pháp học có giám sát (supervised learning), trong đó chương trình máy tính học từ dữ liệu được cung cấp cho nó và thực hiện các quan sát hoặc phân loại mới. Nó là một 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 hoặc 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, label or categories.

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

Có một loạt các thuật toán học máy để phân loại trong học máy: Naive Bayes Classifier, Stochastic Gradient Descent, K-Nearest Neighbor, Decision Tree, Random Forest, ... Mỗi loại thuật toán sẽ có ưu và nhược điểm riêng nhưng ta sẽ tập trung vào Random Forest Classifier sẽ được đề cập ở chương 2. Hơn nữa, chúng tôi cũng sẽ giới thiệu bài toán Dự đoán bệnh tim qua thuật toán Random Forest được đề cập ở chương 3.

### 1.3. Giới thiệu sơ lược về Decision Tree

Decision tree là thuật toán thuộc kỹ thuật học có giám sát. Việc huấn luyện Decision tree và sử dụng nó như một mô hình dự đoán, ánh xạ các quan sát của một mẫu để đưa ra kết luận về giá trị mục tiêu của mẫu. Decision tree dùng trong khai phá dữ liệu có hai loại chính:

- Cây phân loại: khi phân tích kết quả dự đoán thuộc về phân lớp dữ liệu.
- Cây hồi quy: khi phân tích, kết quả dự đoán có thể được coi là một số thực (ví dụ: giá của một ngôi nhà, hoặc thời gian nằm viện của bệnh nhân).

Ở chương 2, ta sẽ xem thử Decision tree có vai trò thế nào trong thuật toán Random forest.

### 1.4. Giới thiệu thuật toán Random Forest

Random forest (rừng ngẫu nhiên) là một phương pháp học tập tổng hợp (ensemble learning) để phân loại, hồi quy và các nhiệm vụ khác hoạt động bằng cách xây dựng vô số Decision tree tại thời điểm đào tạo và xuất ra lớp là chế độ của các lớp (phân loại) hoặc dự đoán trung bình / trung bình (hồi quy) của các cây riêng lẻ. Rừng quyết định ngẫu nhiên phù hợp với thói quen thích nghi với tập huấn luyện của chúng.

Thuật toán đầu tiên cho các khu rừng quyết định 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 được phát triển bởi Leo Breiman và Adele Cutler, người đã đăng ký “Random forest” làm nhãn hiệu (kể từ năm 2019, thuộc sở hữu của Minitab, Inc). Phần mở rộng kết hợp ý tưởng “bagging (đóng gói)” của Breiman và lựa chọn ngẫu nhiên các tính năng, được Ho giới thiệu đầu tiên và sau đó là độc lập bởi Amit và Geman 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 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 cả 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

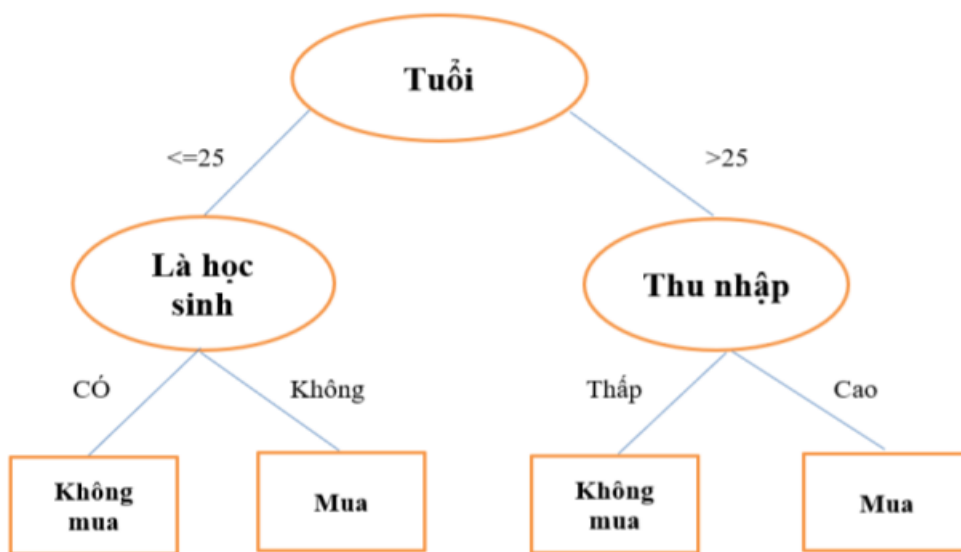
#### 2.1.1 Giới thiệu

Cây phân lớp được dùng trong Random Forest là Decision tree nhị phân có các thành phần:

- Các nút lá mang giá trị nhãn lớp.
- Các nút không phải nút lá là thuộc tính phân chia cây.
- Giá trị ghi trên nhánh là giá trị phân chia tại thuộc tính nút trên.

Sử dụng Decision tree: Mỗi đường đi từ nút gốc đến lá là một luật, áp dụng luật thỏa mãn cho một đối tượng để cho giá trị nhãn lớp của đối tượng đó.

Ví dụ biểu diễn một Decision tree cho việc xác định một người có mua máy tính không dựa vào các thuộc tính: 'tuổi', 'là học sinh' và 'thu nhập' với bảng dữ liệu bên dưới:



Hình 1: Ví dụ về Decision Tree dùng trong Random Forest

	Tuổi	Là học sinh	Thu nhập	Mua máy tính
1	23	Không	Cao	Có mua
2	35	Có	Cao	Có mua
3	15	Có	Thấp	Không
4	40	Không	Thấp	Không

Hình 2: Bảng dữ liệu cho Decision Tree



Từ Decision tree trên ta có các luật như:

- Nếu “Tuổi  $\leq 30$ ” và “là học sinh = có” thì “mua máy tính” = ‘không’.
- Nếu “Tuổi  $\leq 30$ ” và “là học sinh = không” thì “mua máy tính” = ‘có mua’.

Để sử dụng Decision tree trong phân lớp ta đặt đối tượng đó vào cây, áp dụng các luật để xác định nhãn lớp.

**Ví dụ:**

Xét Decision tree trên và một đối tượng X (Tuổi = ‘32’, là học sinh = ‘không’, thu nhập = ‘cao’)

Như vậy theo Decision tree trên ta có một luật: Nếu “Tuổi  $> 30$ ” và “ thu nhập = cao ” thì mua máy tính = ‘ có mua’. Vậy X có giá trị nhãn lớp “mua máy tính” = ‘có mua’ theo luật trên.

### 2.1.2 Một số thuật toán xây dựng Decision Tree

Một số thuật toán thường được dùng để xây dựng Decision tree như:

- ID3 (Iterative Dichotomiser 3): đề xuất bởi Quinlan năm 1980, sử dụng Information Gain để lựa chọn thuộc tính phân chia tại mỗi nút.
- C4.5: mở rộng từ ID3 được Ross Quinlan đề xuất năm 1993, dùng Gain Ratio trong phân chia cây.
- CART: (Classification and Regression Trees): Sử dụng chỉ số Gini để phân chia cây. Các số đo Information Gain, Gain Ratio và Gini là các số đo lựa chọn thuộc tính.

### 2.1.3. Đặc điểm của Decision Tree

Decision tree có một số đặc điểm sau:

- Việc chuẩn bị dữ liệu cho một Decision tree là đơn giản. Các kỹ thuật khác thường đòi hỏi chuẩn hóa dữ liệu, cần tạo các biến phụ và loại bỏ các giá trị rỗng.
  - Decision tree có thể xử lý cả dữ liệu có giá trị bằng số và dữ liệu có giá trị phân loại.
  - Có thể thẩm định một mô hình Decision tree bằng các kiểm tra thống kê.
  - Decision tree có thể xử lý tốt một lượng dữ liệu lớn trong thời gian ngắn. Có thể dùng máy tính cá nhân để phân tích các lượng dữ liệu lớn trong một thời gian đủ ngắn để đưa ra quyết định dựa trên phân tích của Decision tree.
  - Nhược điểm của Decision tree là khó giải quyết được những vấn đề có dữ liệu phụ thuộc thời gian liên tục, dễ xảy ra lỗi khi có quá nhiều tính toán để xây dựng mô hình Decision tree.
- Quá trình xây dựng Decision tree sẽ chọn thuộc tính tốt nhất để phân nhánh cây, một thuộc tính gọi là tốt tùy vào số đo được sử dụng khi phân chia tập mẫu tại mỗi nút, các số đo lựa chọn thuộc tính khi phân nhánh được trình bày trong mục 2.4.

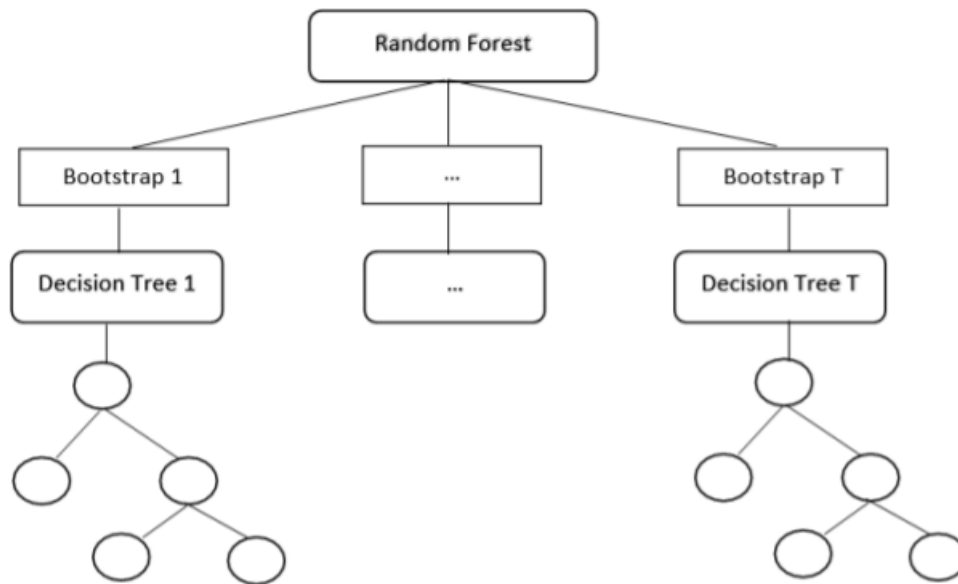
## 2.2 Thuật toán Random Forest

### 2.2.1 Định nghĩa

Random Forest là thuật toán học máy dựa trên kỹ thuật lắp ghép, kết hợp 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 các cây trong rừng làm kết quả cuối cùng của thuật toán.

### 2.2.2 Mô hình thuật toán



Hình 3: 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

### 2.2.3 Thuật toán

Thuật toán Random Forest cho phân lớp và hồi quy

#### A. Huấn luyện

- \* Cho tập huấn luyện  $S$
- \* Đối với mỗi cây trong rừng  $N_{cay}$

- Xây dựng Cây phân lớp  $T_k$  với tập  $S_k$ ;  $S_k$  được lấy Bootstrap từ  $S$
- Tại mỗi nút của cây, chọn thuộc tính tốt nhất để phân chia cây từ tập thuộc tính  $F_i$  được chọn ngẫu nhiên trong tập thuộc tính  $F$
- Mỗi cây được xây dựng đến độ sâu tối đa (không cắt nhánh)

**B. Dự đoán:** Thuật toán dự đoán nhãn lớp cho một đối tượng theo

- Phân lớp: sử dụng kết quả số đông của các cây.
- Hồi quy: lấy giá trị trung bình kết quả tất cả các cây

#### 2.2.4 Ví dụ minh họa

Cho tập dữ liệu gồm 5 bộ, mỗi bộ có 4 thuộc tính và một giá trị nhãn lớp như hình 4 bên dưới

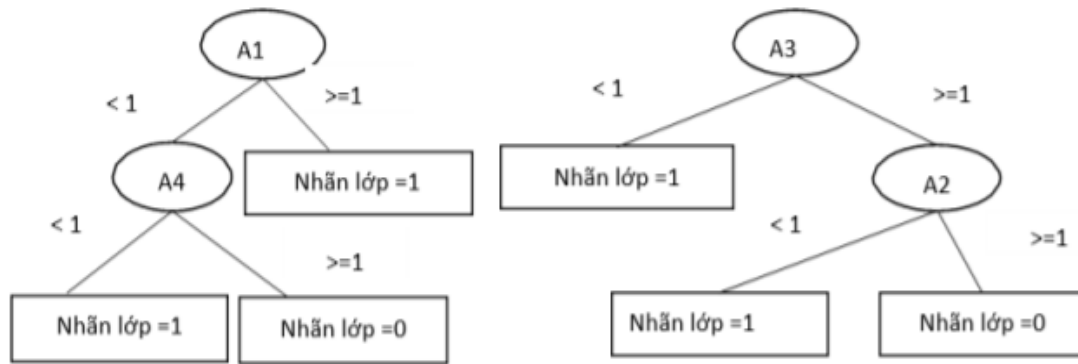
Thuộc tính STT	A1	A2	A3	A4	Nhãn lớp
1	1	0	1	1	1
2	0	0	0	0	1
3	1	0	0	0	1
4	0	1	1	1	0
5	1	1	0	1	1

Hình 4: Tập dữ liệu minh họa thuật toán Random Forest

Với số cây là 2, số thuộc tính lựa chọn phân chia là 2, thuật toán thực hiện như sau:

Từ tập  $S$ , dùng Bootstrap lấy mẫu có hoàn lại xây dựng tập con  $S_k$ , với  $jS_kj = 4$  được:  $S_1$  gồm các bộ 1, 2, 3, 4;  $S_2$  gồm các bộ 1,4,3,5 (theo thứ tự trong bảng).

Xây dựng 2 cây  $T_1$ ;  $T_2$  với các tập dữ liệu tương ứng  $S_1$ ,  $S_2$



Hình 5: Cây phân lớp trong rừng

Dự đoán nhãn lớp cho bộ  $X(1; 0; 1; 0)$ :

- Cây T1 cho  $X$  có giá trị nhãn lớp là 1

- Cây T2 cho  $X$  có giá trị nhãn lớp là 1

Vậy  $X(1; 0; 1; 0)$  có nhãn lớp là 1 do có 2 cây cho giá trị 1 và 0 cây cho giá trị 0 nên giá trị 1 sẽ là kết quả cuối cùng vì số đông cây cho giá trị này.

Nếu có nhiều nhãn lớp có cùng số cây cho ra thì có thể chọn bất kì nhãn lớp nào làm kết quả, thường chọn giá trị đầu tiên hoặc giá trị chiếm đa số trong tập mẫu  $D$ .

### 2.2.5. Đặc điểm của Random Forest

#### a. Ưu điểm:

- Thuật toán giải quyết tốt các bài toán có nhiều dữ liệu nhiễu, thiếu giá trị. Do cách chọn ngẫu nhiên thuộc tính nên các giá trị nhiễu, thiếu ảnh hưởng không lớn đến kết quả.

- Có những sự ước lượng nội tại như độ chính xác của mô hình phỏng đoán hoặc độ mạnh và liên quan giữa các thuộc tính (Out of bag)

- Dễ dàng thực hiện song song. Thay vì một máy thực hiện cả thuật toán, ta có thể sử dụng nhiều máy để xây dựng các cây sau đó ghép lại thành rừng.

- Các sai số được giảm thiểu do kết quả của Random Forest được tổng hợp thông qua nhiều người học (Cây phân lớp).

- Việc chọn ngẫu nhiên tại mỗi bước trong Random Forest sẽ làm giảm mối tương quan giữa các người học trong việc tổng hợp các kết quả.

- Lỗi chung của một rừng các cây phân lớp phụ thuộc vào lỗi riêng của từng cây trong rừng cũng như mối tương quan giữa các cây.

#### b. Hạn chế:

- Dữ liệu huấn luyện cần được đa dạng hóa và cân bằng về số nhãn lớp. Việc không cân bằng nhãn lớp khiến kết quả dự đoán của thuật toán có thể lệch về số đông nhãn lớp.

- Thời gian huấn luyện của rừng có thể kéo dài tùy số cây và số thuộc tính phân chia.

### **c. Out of bag**

Random Forest sử dụng Bootstrap để tạo tập mẫu cho các cây. Khi tập mẫu được rút ra từ một tập huấn luyện của một cây với sự thay thế thì theo ước tính có khoảng 1/3 các phần tử không có nằm trong mẫu này. Điều này có nghĩa là chỉ có khoảng 2/3 các phần tử trong tập huấn luyện tham gia vào trong các tính toán, và 1/3 các phần tử này được gọi là dữ liệu out-of-bag. Dữ liệu out-of-bag được sử dụng để ước lượng lỗi tạo ra từ việc kết hợp các kết quả từ các cây tổng hợp trong Random Forest cũng như dùng để ước tính độ quan trọng thuộc tính. Hơn nữa có thể sử dụng chính tập huấn luyện để kiểm thử mô hình từ thuật toán trước khi đưa vào ứng dụng.

### **2.2.6. Ứng dụng của thuật toán Random Forest**

Trong đề án này, nhóm chúng tôi đưa ra bốn ứng dụng của việc sử dụng thuật toán Random Forest: Ngân hàng, Y học, Thị trường chứng khoán và Thương mại điện tử:

- Đối với ứng dụng trong ngân hàng: thuật toán Random Forest được sử dụng để tìm khách hàng trung thành, có nghĩa là những khách hàng có thể vay nhiều tiền và trả lãi cho ngân hàng một cách hợp lý, và những khách hàng lừa đảo, có nghĩa là những khách hàng có hồ sơ xấu như không trả được nợ, cho vay không đúng hạn hoặc có những hành động nguy hiểm.

- Đối với ứng dụng trong y học: thuật toán Random Forest có thể được sử dụng để vừa xác định sự kết hợp chính xác của các thành phần trong y học, vừa để xác định bệnh bằng cách phân tích hồ sơ bệnh án của bệnh nhân.

- Đối với ứng dụng trên thị trường chứng khoán: thuật toán Rừng Ngẫu nhiên có thể được sử dụng để xác định hành vi của cổ phiếu và mức lỗ hoặc lợi nhuận dự kiến.

- Đối với ứng dụng trong thương mại điện tử: thuật toán Random Forest có thể được sử dụng để dự đoán liệu khách hàng có thích sản phẩm được giới thiệu hay không, dựa trên trải nghiệm của những khách hàng tương tự.

## **2.3 Phương pháp lấy mẫu Bootstrap**

### **2.3.1 Giới thiệu Bootstrap**

Bootstrap là tập hợp một số kỹ thuật phân tích dựa vào nguyên lý chọn mẫu có hoàn lại để ước tính các thông số trong thống kê. Phương pháp Bootstrap do nhà thống kê học Bradley Efron thuộc đại học Stanford (Mỹ) phát triển từ cuối thập niên 1979 nhưng đến khi máy tính được sử dụng phổ biến thì phương pháp này mới trở thành phương pháp phổ biến trong phân tích thống kê và được ứng dụng rộng rãi trong rất nhiều lĩnh vực khoa học. 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 như không giải được.

### **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 mẫu ngẫu nhiên làm tập mẫu huấn luyện cho Decision tree.

Ví dụ: Giả sử có 15,000 bệnh nhân nghi ngờ bị nhiễm độc, cần xác định mẫu máu của các bệnh nhân này. Nếu lấy hết 15,000 đem đi xét nghiệm thì tốn kém và mất thời gian nên chỉ lấy 1,000 bệnh nhân làm đại diện cho cả 15,000 bệnh nhân.

Cần lấy 1,000 mẫu từ 15,000 bệnh nhân bằng phương pháp Bootstrap, ta tiến hành như sau:

Lặp 1,000 lần thao tác:

- Lấy ngẫu nhiên một bệnh nhân từ 15,000 bệnh nhân.
- Ghi lại chỉ số bệnh nhân được chọn.
- Đặt bệnh nhân vào lại 15,000 bệnh nhân.

Cuối cùng có được là chỉ số của 1,000 bệnh nhân được chọn.

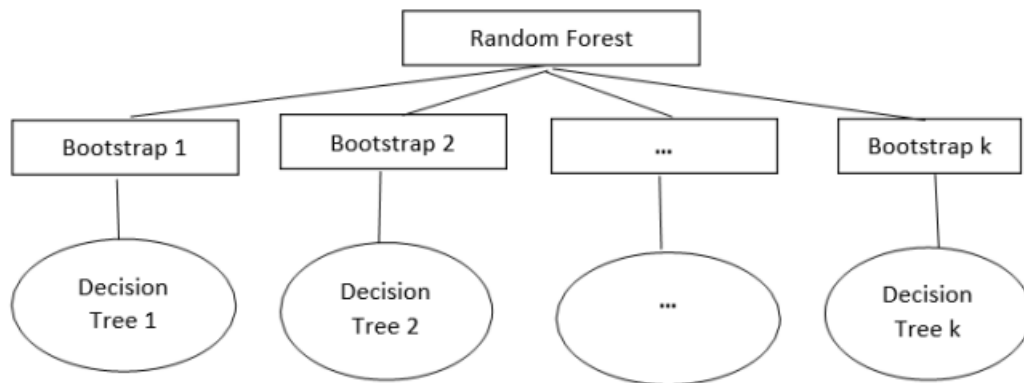
Chú ý rằng trong cách chọn ngẫu nhiên có hoàn lại như trên, có thể có một số bệnh nhân được chọn hơn 1 lần.

### **2.3.3 Đặc điểm của Bootstrap**

- Phương pháp chọn mẫu 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 mẫu ngẫu nhiên riêng biệt. Ngoài ra còn sử dụng để đánh giá nội tại của thuật toán (Out-of-bag).



Hình 6. Bootstrap trong Random Forest

### 2.3.5 Xây dựng một trình phân loại (classification) bằng cách sử dụng Scikit-learn với Random Forest Classifier

Ta sẽ xây dựng một mô hình trên tập dữ liệu hoa iris, đó là một bộ phân loại rất nổi tiếng. Nó bao gồm chiều dài vách ngăn, chiều rộng vách ngăn, chiều dài cánh hoa, chiều rộng cánh hoa và loại hoa. Có ba loài hoặc lớp: setosa, versicolor và virginia.

Ta sẽ xây dựng một mô hình để phân loại loại hoa. Tập dữ liệu có sẵn trong thư viện scikit-learning hoặc bạn có thể tải xuống từ UCI Machine Learning Repository.

Bắt đầu bằng cách nhập thư viện datasets từ scikit-learn và tải tập dữ liệu iris bằng load\_iris ().

---

```

#Import scikit-learn dataset library
from sklearn import datasets

#Load dataset
iris = datasets.load_iris()
  
```

---

Ta có thể in tên mục tiêu và đối tượng địa lý, để đảm bảo bạn có tập dữ liệu phù hợp, như vậy:

---

```

# print the label species(setosa, versicolor, virginica)
print(iris.target_names)

# print the names of the four features
print(iris.feature_names)
['setosa' 'versicolor' 'virginica']

['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
  
```

---

Ta nên luôn khám phá dữ liệu của mình một chút để biết bạn đang làm việc với cái gì. Tại đây, ta có thể thấy năm hàng đầu tiên của tập dữ liệu được in, cũng như biến mục tiêu cho toàn bộ tập dữ liệu.

[illegible]

Ở đây, có thể tạo một DataFrame của tập dữ liệu iris theo cách sau:

```
# Creating a DataFrame of given iris dataset.
import pandas as pd
data=pd.DataFrame({
    'sepal length':iris.data[:,0],
    'sepal width':iris.data[:,1],
    'petal length':iris.data[:,2],
    'petal width':iris.data[:,3],
    'species':iris.target
})
data.head()
```

	petal length	petal width	sepal length	sepal width	species
0	1.4	0.2	5.1	3.5	0
1	1.4	0.2	4.9	3.0	0
2	1.3	0.2	4.7	3.2	0
3	1.5	0.2	4.6	3.1	0
4	1.4	0.2	5.0	3.6	0

Trước tiên, chúng ta tách các cột thành các biến phụ thuộc và độc lập (hoặc các tính năng và nhãn). Sau đó, chia các biến đó thành một tập huấn luyện và kiểm tra.



---

```

# Import train_test_split function
from sklearn.model_selection import train_test_split

X=data[['sepal length', 'sepal width', 'petal length', 'petal width']] #
    Features
y=data['species'] # Labels

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
    # 70% training and 30% test

```

---

Sau khi tách, ta sẽ đào tạo mô hình trên tập huấn luyện và thực hiện các dự đoán trên tập kiểm tra.

---

```

#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
#sau khi đào tạo, kiểm tra tính chính xác bằng cách sử dụng giá trị thực
    te và dự đoán.
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

('Accuracy:', 0.9333333333333335)
#bạn cũng có thể đưa ra dự đoán cho một mục, ví dụ:
sepal length = 3
sepal width = 5
petal length = 4
petal width = 2

```

---

Bây giờ ta có thể dự đoán loại hoa nào.

---

```

clf.predict([[3, 5, 4, 2]])
array([2])

```

---

Ở đây, 2 cho biết loại hoa Virginica.

### 2.3.6 Các tính năng quan trọng trong Scikit-learn

Ở đây, ta đang tìm các tính năng quan trọng hoặc chọn các tính năng trong tập dữ liệu IRIS. Trong quá trình tìm hiểu, ta có thể thực hiện tác vụ này trong các bước sau:

Đầu tiên, ta cần tạo một mô hình Random Forests. Thứ hai, sử dụng biến quan trọng của tính năng để xem điểm quan trọng của đối tượng địa lý. Thứ ba, hình dung các điểm số này bằng thư viện.

---

```

from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=None, max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
oob_score=False, random_state=None, verbose=0,
warm_start=False)

import pandas as pd

feature_imp =
    pd.Series(clf.feature_importances_,index=iris.feature_names).sort_values(ascending=
feature_imp

petal width (cm) 0.458607

petal length (cm) 0.413859

sepal length (cm) 0.103600

sepal width (cm) 0.023933

dtype: float64

```

---

Ta cũng có thể hình dung tầm quan trọng của đối tượng địa lý. Hình ảnh dễ hiểu và dễ hiểu.

Để hiển thị, ta có thể sử dụng kết hợp matplotlib và seaborn. Bởi vì seaborn được xây dựng trên đầu trang của matplotlib, nó cung cấp một số chủ đề tùy chỉnh và cung cấp các loại cốt truyện bổ sung. Matplotlib là một superset của seaborn và cả hai đều quan trọng không kém cho visualizations tốt. Thứ ba, hình dung những điểm số bằng cách sử dụng thư viện seaborn.

---

```

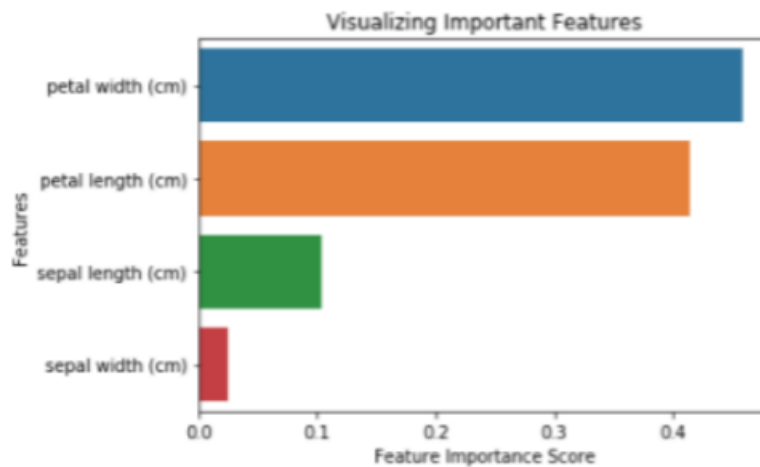
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')

```

---

```
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

---



Tạo mô hình trên các tính năng được chọn Tại đây, ta có thể loại bỏ tính năng "chiều rộng sepal" vì nó có tầm quan trọng rất thấp và chọn 3 tính năng còn lại.

---

```
# Import train_test_split function
from sklearn.cross_validation import train_test_split
# Split dataset into features and labels

X=data[['petal length', 'petal width','sepal length']] # Removed feature
      "sepal length"
y=data['species']

'# Split dataset into training set and test set'
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.70,
      random_state=5) # 70% training and 30% test'
```

---

Sau khi chia nhỏ, ta sẽ tạo ra một mô hình trên các tính năng tập huấn được chọn, thực hiện các dự đoán về các tính năng bộ thử đã chọn và so sánh các giá trị thực tế và được dự đoán.

---

```
from sklearn.ensemble import RandomForestClassifier
#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)
#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)
'# prediction on test set'
y_pred=clf.predict(X_test)
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
'# Model Accuracy, how often is the classifier correct?'
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
('Accuracy:', 0.95238095238095233)
```

---

Ta có thể thấy rằng sau khi loại bỏ các tính năng ít quan trọng nhất (chiều dài sepal), độ chính xác tăng lên. Điều này là do bạn đã xóa dữ liệu gây nhiễu và tiếng ồn, dẫn đến độ chính xác tăng lên. Một số lượng ít tính năng hơn cũng làm giảm thời gian đào tạo.

Phần kết luận Trong hướng dẫn này, chúng ta đã biết được Random Forests là gì, nó hoạt động như thế nào, tìm ra các tính năng quan trọng, so sánh giữa Random Forests và Decision tree, lợi thế và bất lợi. Bạn cũng đã học xây dựng mô hình, đánh giá và tìm các tính năng quan trọng trong scikit-learn.

## **2.4. Tham số và siêu tham số trong model Random Forest**

Tôi sẽ giới thiệu 1 số tham số và siêu tham số dùng được trong Random Forest. (Phần thực nghiệm siêu tham số sẽ được làm trong chương 3.5)

### **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 “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. Cá nhân tôi đã tìm thấy một tập hợp với nhiều mô hình của các trạng thái ngẫu nhiên khác nhau và tất cả các tham số tối ưu đôi khi hoạt động tốt hơn trạng thái ngẫu nhiên riêng lẻ.

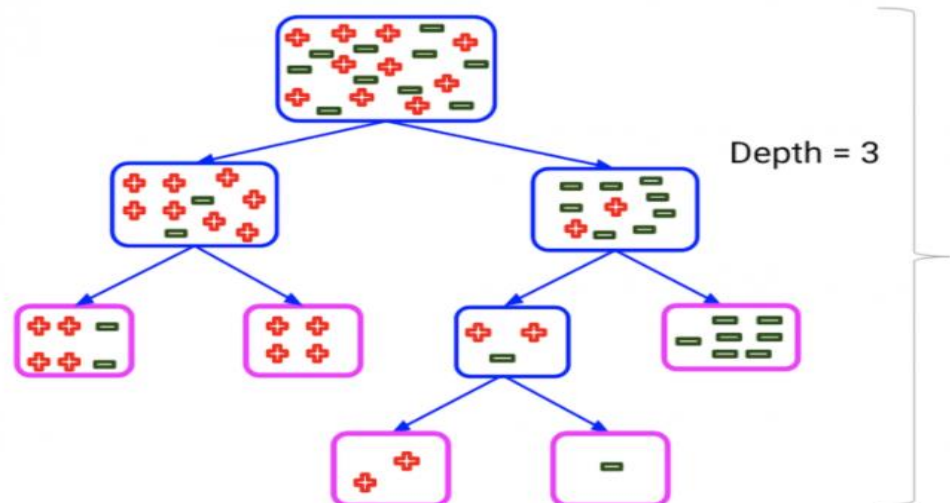
#### **c. oob\_score**

Đây là một phương pháp xác nhận chéo rừng ngẫu nhiên. Nó rất giống với một 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 tress 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 tự đào tạo.

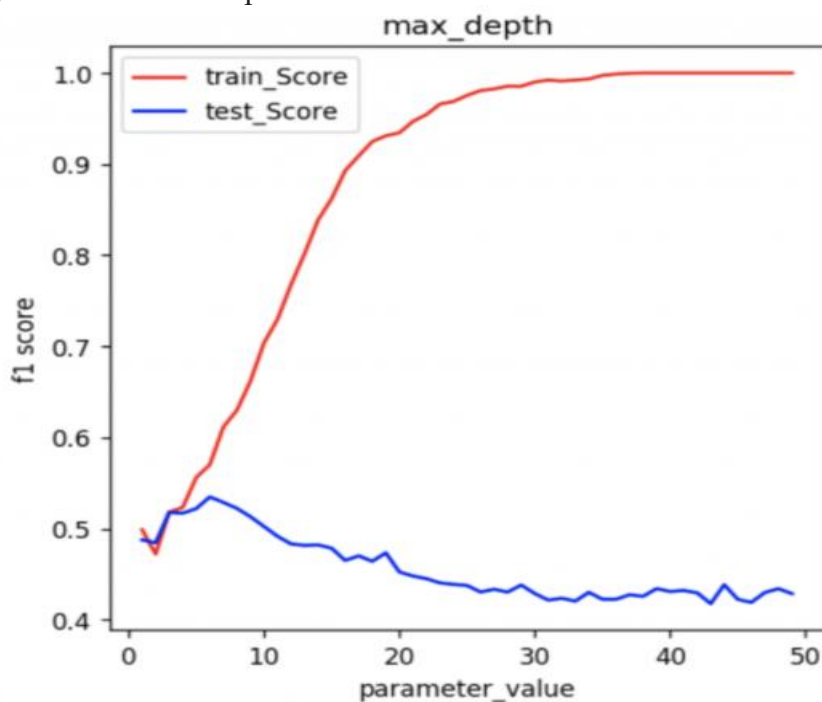
### **2.4.2. Siêu tham số (Hyperparameter)**

#### **a. max\_depth**

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



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



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. Mặt 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.

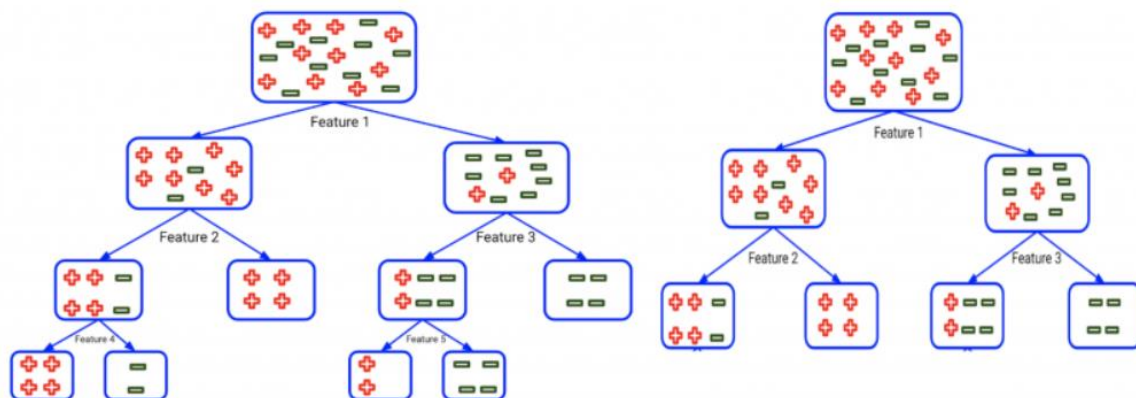
Cây bắt đầu trạng bị quá mức cho tập huấn luyện và do đó không thể tổng quát hóa các điểm không nhìn thấy trong tập thử nghiệm.

## 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 để tách nó ra.

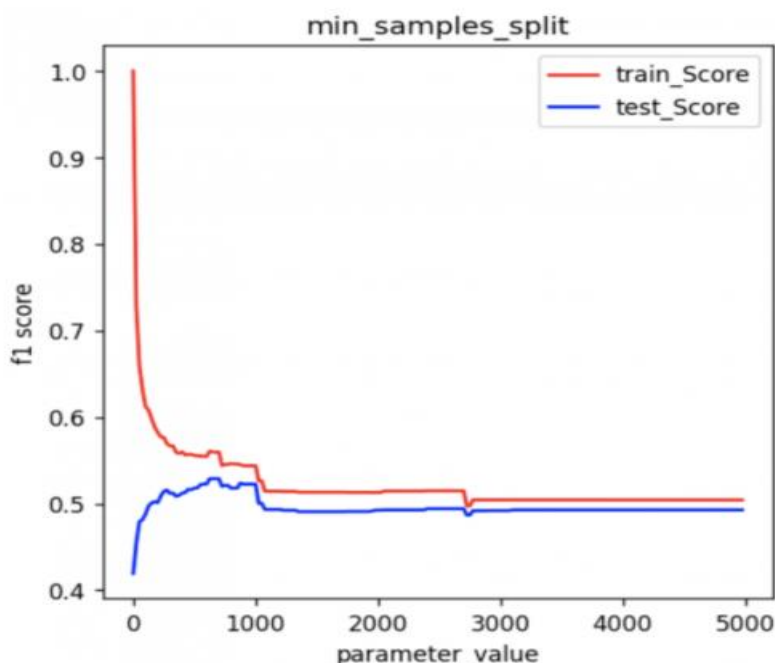
Giá trị mặc định của `Minimum_sample_split` được gán cho 2. Điều này có nghĩa là nếu bất kỳ nút đầu cuối nào có nhiều hơn hai quan sát và không phải là nút thuần túy, chúng ta có thể chia nó thành các nút con.

Việc có giá trị mặc định là 2 đặt ra vấn đề rằng một cây thường tiếp tục tách 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.



Bằng cách tăng giá trị của `min_sample_split`, chúng ta có thể giảm số lần phân tách xảy ra trong cây quyết định và 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 các tham số khác vẫn giữ nguyên và chỉ giá trị của `min_samples_split` được thay đổi:



Khi tăng giá trị của siêu tham số `min_sample_split`, chúng ta có thể thấy rõ ràng đối với giá trị nhỏ của các 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 bạn cần lưu ý. Khi giá trị tham số tăng quá nhiều, sẽ có sự sụt giảm

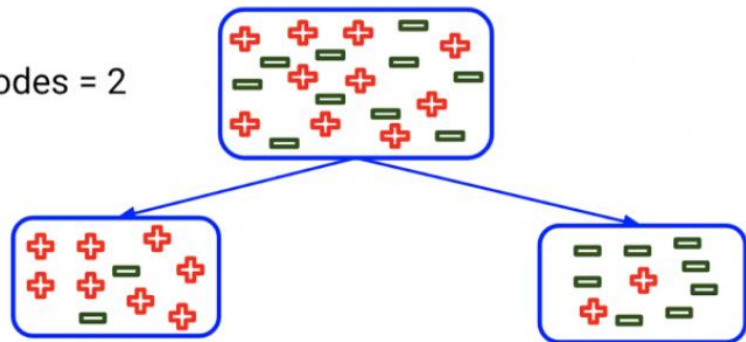
tổng thể về cả điểm đào tạo và điểm kiểm tra. Điều này là do thực tế là yêu cầu tối thiểu của việc tách một nút quá cao nên không có sự phân tách đáng kể nào được quan sát. Kết quả là, khu rừng ngẫu nhiên bắt đầu kém chất lượng.

### c. max\_leaf\_nodes

Tiếp theo, hãy chuyển sang một siêu tham số trong Random Forest khác được gọi là max\_leaf\_nodes. Siêu tham số này đặt ra một điều kiện về việc tách các 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 số nút đầu cuối đã chỉ định, nó sẽ dừng việc tách và cây sẽ không phát triển thêm.

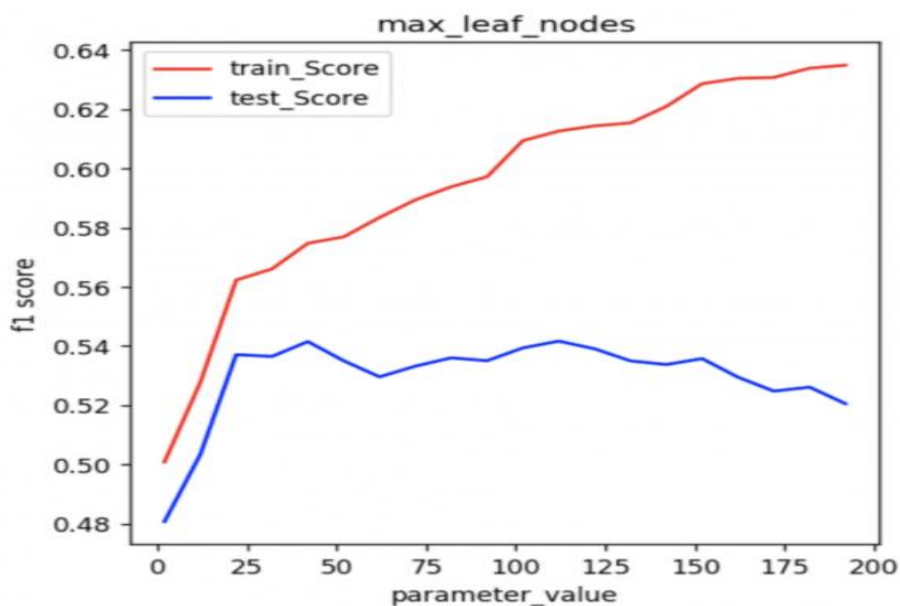
Giả sử chúng ta đặt các nút đầu cuối tối đa là 2 trong trường hợp này. Vì chỉ có một nút, nó sẽ cho phép cây phát triển thêm:

Max Terminal nodes = 2



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 hoặc 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 rất 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:



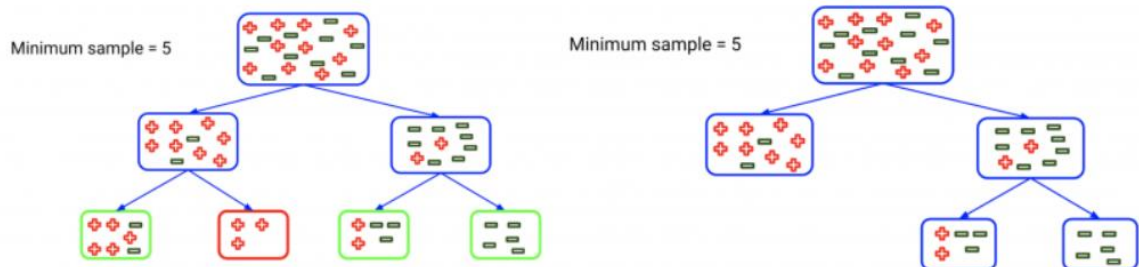
Chúng ta có thể thấy rằng khi giá trị tham số rất nhỏ, cây không được trang bị đầy đủ và khi giá trị tham số tăng lên, hiệu suất của cây qua cả thử nghiệm và huấn luyện đều tăng. Theo biểu đồ này, cây bắt đầu quá mức khi giá trị tham số vượt quá 25.



#### d. min\_samples\_leaf

Đã đến lúc chuyển trọng tâm của chúng ta sang min\_sample\_leaf. Siêu thông số Rừng Ngẫu nhiên này chỉ định số lượng mẫu tối thiểu cần có trong nút lá sau khi tách một nút.

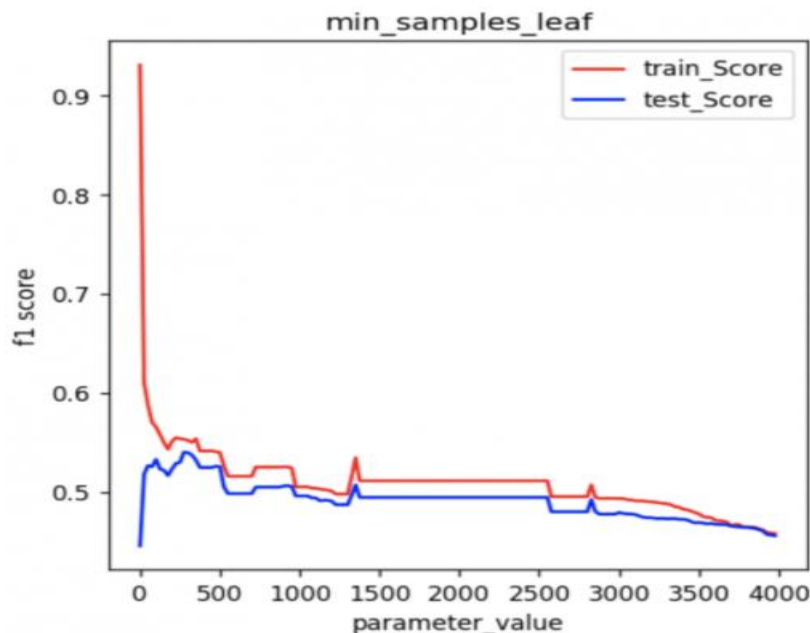
Hãy hiểu min\_sample\_leaf bằng cách sử dụng một ví dụ. Giả sử chúng tôi đã đặt các mẫu tối thiểu cho một nút đầu cuối là 5:



Cây bên trái tượng trưng cho cây không bị bó buộc. Ở đây, các nút được đánh dấu bằng màu xanh lá cây cho thấy 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 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.



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

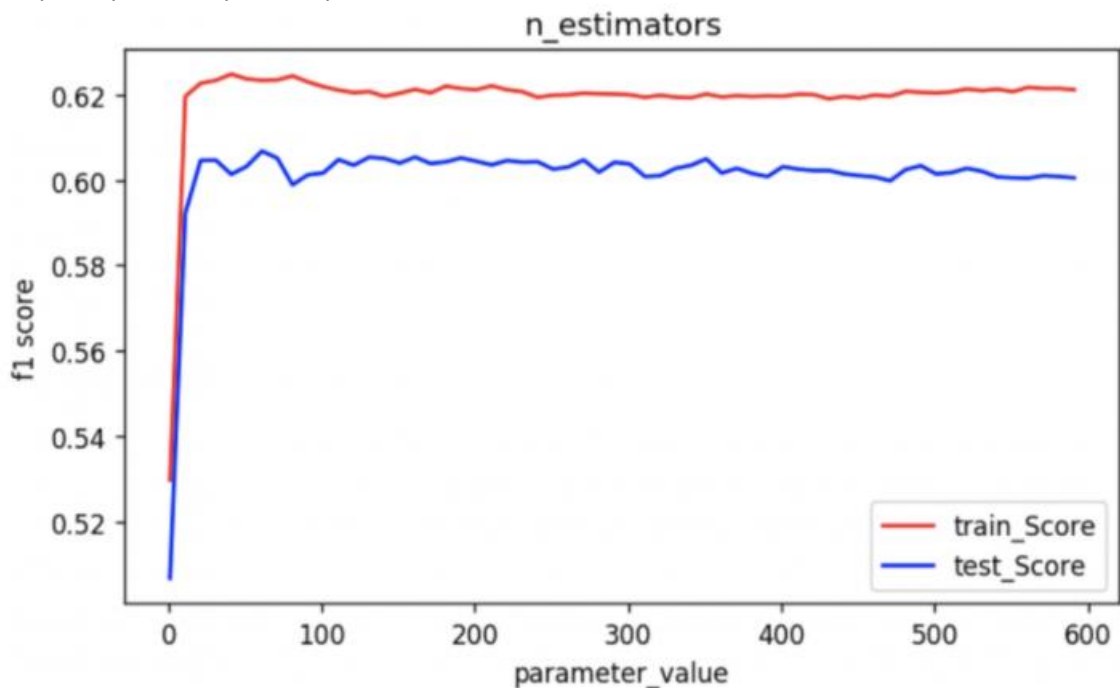


### e. n\_estimators

Chúng ta biết rằng thuật toán Random Forest không là gì khác ngoài một nhóm các cây. Nhưng chúng ta nên xem xét có bao nhiêu cây? Đó là một câu hỏi phổ biến mà các nhà khoa học về dữ liệu mới hỏi. Và nó là một trong những hợp lệ!

Chúng ta 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 về thời gian của mô hình Rừng ngẫu nhiên cũng tăng lên.

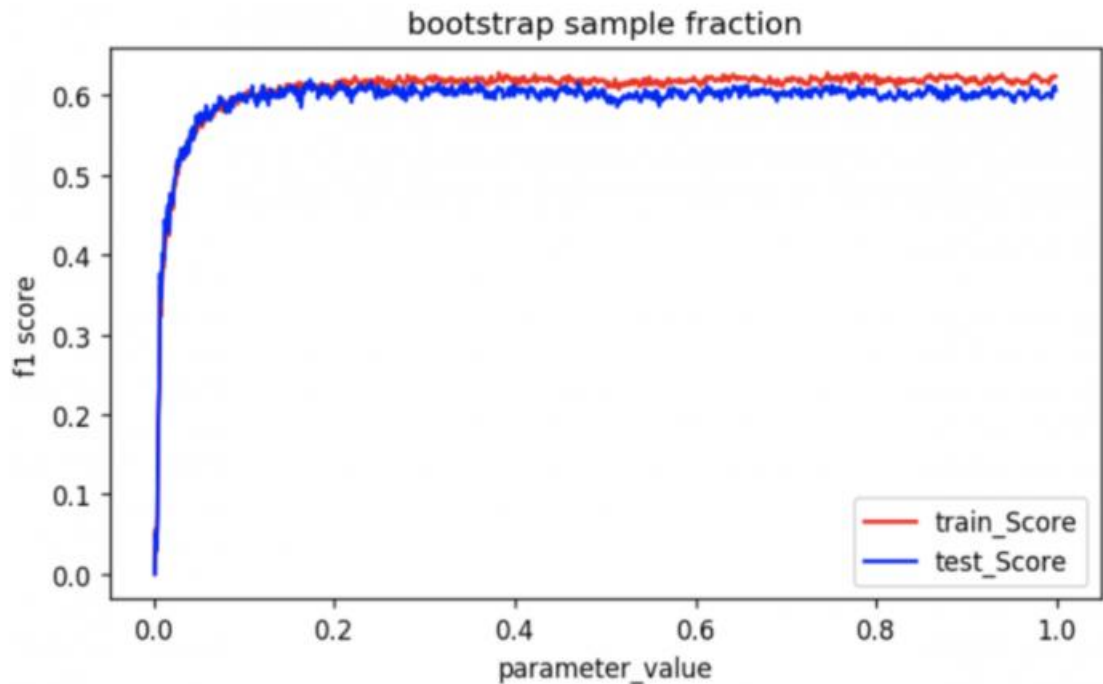
Trong biểu đồ này, chúng ta có thể thấy rõ rằng hiệu suất của mô hình tăng mạnh và sau đó đình trệ ở một mức độ nhất định:



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 bạn tiết kiệm sự phức tạp trong tính toán và ngăn việc sử dụng bình cứu hỏa trên CPU của bạn!

### 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. Bạn có thể nghĩ rằng nhiều dữ liệu hơn luôn tốt hơn. Hãy thử xem điều đó có hợp lý không.



Chúng ta có thể thấy rằng hiệu suất của mô hình tăng mạnh và sau đó bão hòa khá nhanh. Bạn có thể tìm ra chìa khóa rút ra từ hình dung này là gì không?

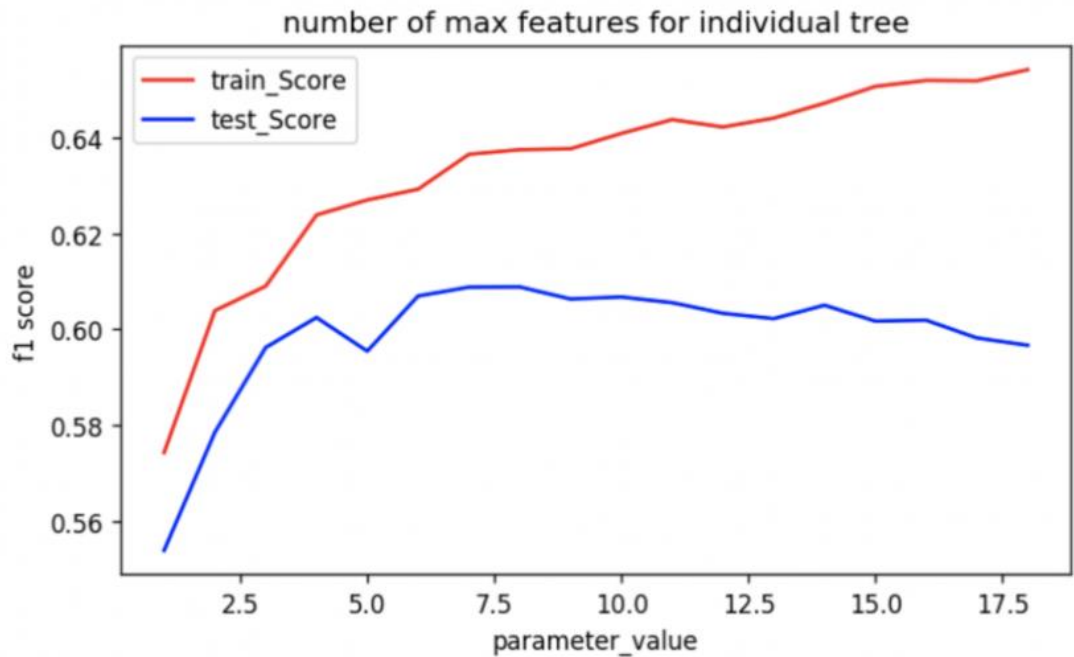
Không nhất thiết phải cung cấp cho mỗi cây quyết định của Rừng ngẫu nhiên toàn bộ dữ liệu. Nếu bạn nhận thấy, hiệu suất mô hình đạt mức tối đa khi dữ liệu được cung cấp nhỏ hơn 0,2 phần của tập dữ liệu ban đầu. Điều đó thật đáng kinh ngạc!

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 dữ liệu được khởi động cho mỗi cây quyết định. Do đó, thời gian đào tạo của mô hình Rừng ngẫu nhiên được giảm 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 các tính năng tối đa được cung cấp cho mỗi cây trong một khu rừng ngẫu nhiên.

Chúng tôi biết rằng khu rừng ngẫu nhiên chọn một số mẫu ngẫu nhiên từ các đối tượng địa lý để tìm ra sự phân chia tốt nhất. Hãy xem sự thay đổi của tham số này có thể ảnh hưởng đến hiệu suất của mô hình Random Forest của chúng ta như thế nào.



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` bão hòa và thậm chí bắt đầu giảm dần về cuối, điều đó rõ ràng có nghĩa là mô hình 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 CLASSIFIER TRONG RANDOM FOREST

### 3.1. Giới thiệu bài toán

Bệnh tim mô tả một loạt các tình trạng ảnh hưởng đến tim của bạn. Các bệnh thuộc phạm vi bệnh tim bao gồm các bệnh về mạch máu, chẳng hạn như bệnh mạch vành, các vấn đề về nhịp tim (loạn nhịp tim) và các dị tật tim mà bạn sinh ra (dị tật tim bẩm sinh), và nó liên quan đến một số những bệnh khác.

Bệnh tim mạch thường đề cập đến các tình trạng liên quan đến các mạch máu bị thu hẹp hoặc tắc nghẽn có thể dẫn đến đau tim, đau ngực (đau thắt ngực) hoặc đột quỵ. Các bệnh tim khác, chẳng hạn như những bệnh ảnh hưởng đến cơ, van hoặc nhịp tim của bạn, cũng được coi là các dạng bệnh tim.

Bệnh tim là một trong những nguyên nhân lớn nhất gây ra bệnh tật và tử vong cho người dân trên thế giới. Dự đoán bệnh tim mạch được coi là một trong những môn học quan trọng nhất trong phân tích dữ liệu lâm sàng. Số lượng dữ liệu trong ngành chăm sóc sức khỏe là rất lớn. Khai thác dữ liệu biến bộ sưu tập lớn dữ liệu chăm sóc sức khỏe thô thành thông tin có thể giúp đưa ra các quyết định và dự đoán sáng suốt.

Báo cáo và code của bài toán sẽ được đăng lên Kho lưu trữ Github của nhóm - [github](#).

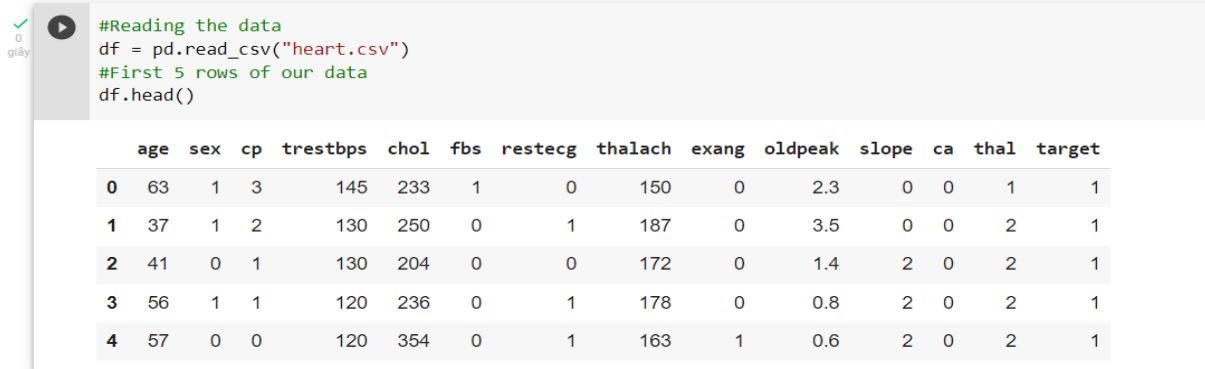
### 3.2. Tập dữ liệu

Trong đồ án này, tôi sẽ áp dụng phương pháp Machine Learning với model Random Forest Classifier (và cuối cùng là so sánh với 1 số model khác) để phân loại xem một người có đang bị bệnh tim hay không, bằng cách sử dụng một trong những tập dữ liệu được sử dụng nhiều nhất - tập dữ liệu Heart.csv lấy trên website: <https://www.kaggle.com/>

Tập dữ liệu trong file Heart.csv gồm 303 dòng dữ liệu và 14 cột dữ liệu.

```
[43] df.shape
```

```
(303, 14)
```



```
#Reading the data
df = pd.read_csv("heart.csv")
#First 5 rows of our data
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 9. 5 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 = thiếu hụt cố định; 7 = thiếu hụt có thể hồi phục)
- **target**: Bệnh tim (0 = không, 1 = có)

**\* Reset tên 14 cột dữ liệu:**

```
df.columns = ['age', 'sex', 'chest_pain_type', 'resting_blood_pressure', 'cholesterol',  
              'fasting_blood_sugar', 'rest_ecg', 'max_heart_rate_achieved',  
              'exercise_induced_angina', 'st_depression', 'st_slope', 'num_major_vessels',  
              'thalassemia', 'target']
```

*Hình 10. Đặt lại tên cho 14 cột dữ liệu*

**\* Split Dataset:**

- + Dataset được chia làm 2 set theo tỉ lệ 8:2:
- 242 hàng dữ liệu cho Training set (80%).

- 61 hàng dữ liệu cho Validation set (20%).

```
✓ [62] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, stratify = Y, random_state=42)
0 giây

✓ [63] print(X.shape, X_train.shape, X_test.shape)
0 giây

(303, 13) (242, 13) (61, 13)
```

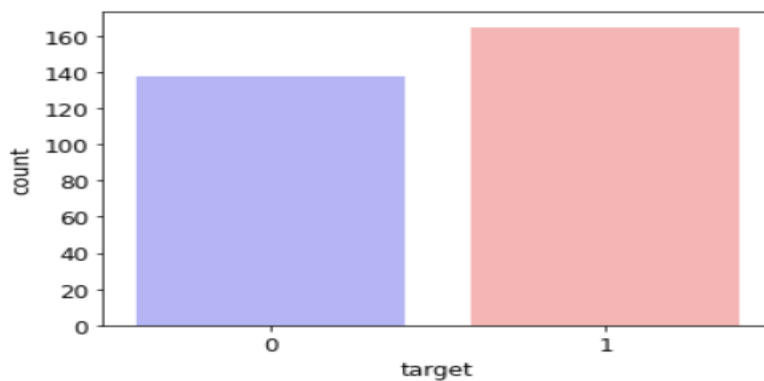
+ Ta cũng xử lý dữ liệu theo cột Target để dễ dàng hơn khi train model.

### 3.3 Tiền xử lý dữ liệu

#### \* Data Exploration

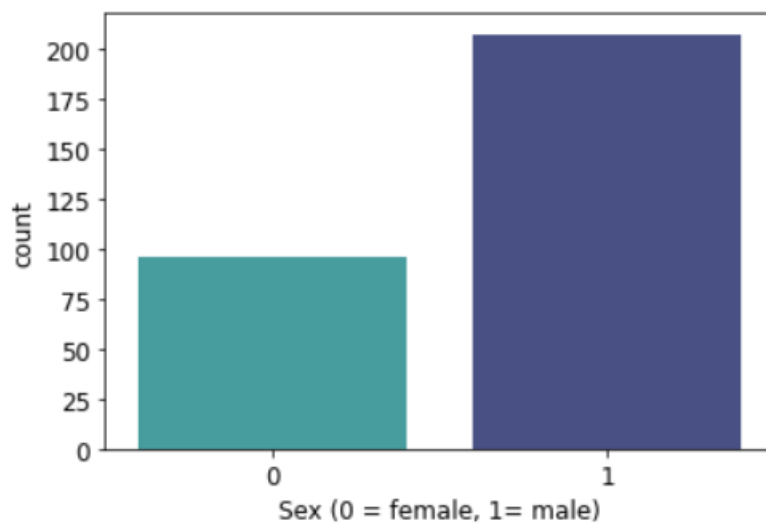
- Xử lý dữ liệu cột target: Xét có bệnh tim hay không?

→ target = 1: Có bị bệnh tim  
target = 0: Không bị bệnh tim



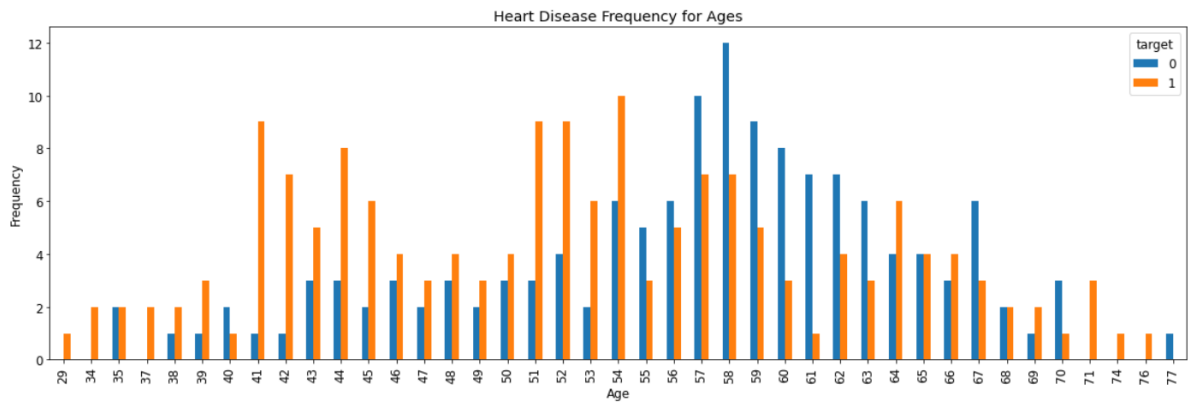
Hình 20. Biểu đồ target

- Xử lý dữ liệu cột sex: Xét giới tính Nam/Nữ



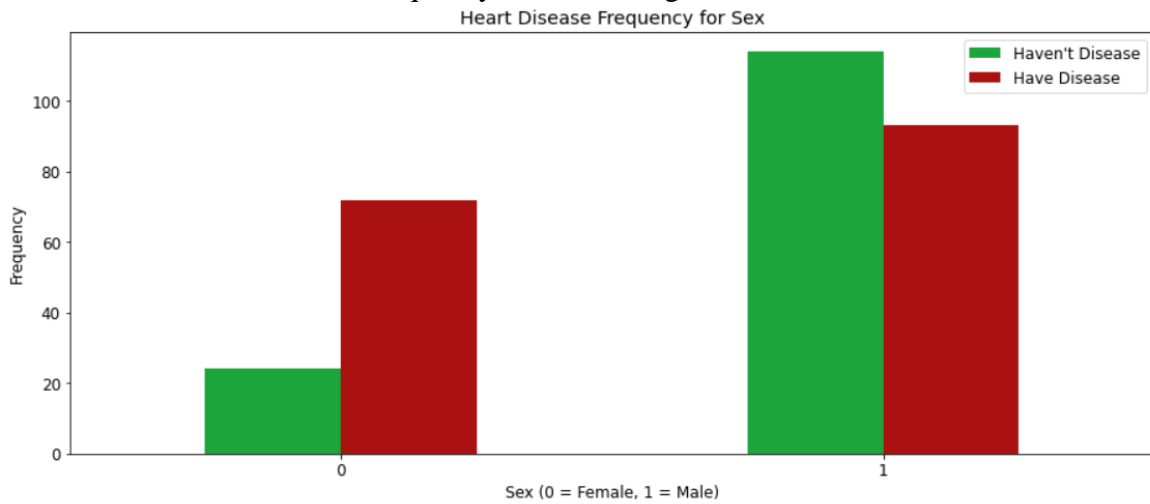
Hình 21. Biểu đồ Sex

- Tổng hợp dữ liệu cột theo Frequency, Age và Target:



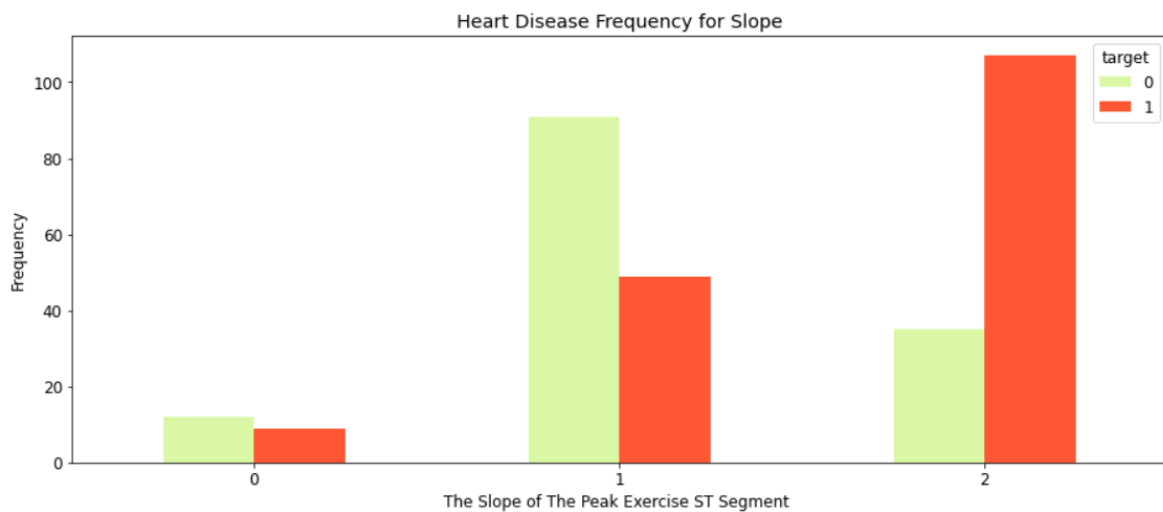
Hình 22. Biểu đồ Heart Disease Frequency for Ages

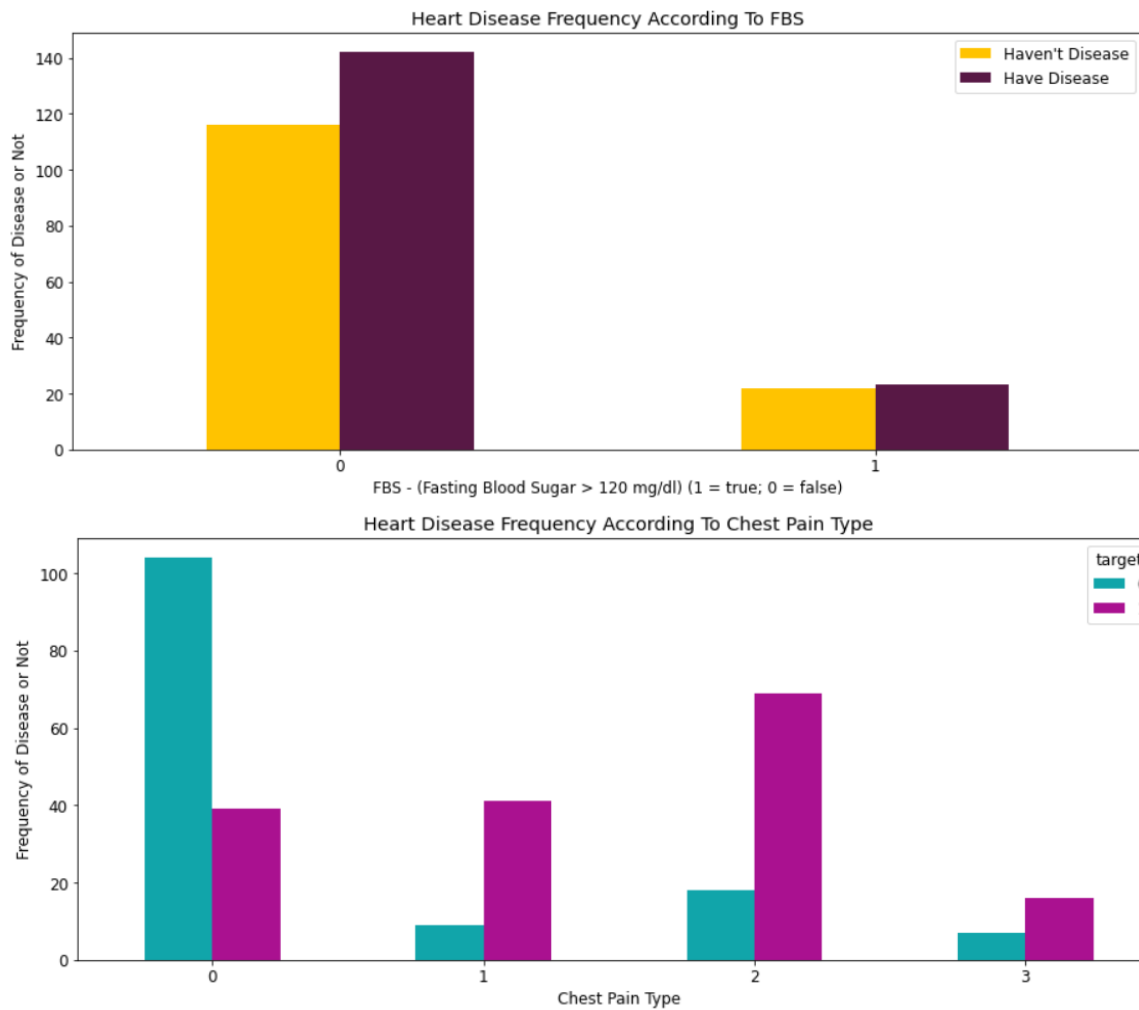
- Dữ liệu các cột theo Frequency, Sex and Target:



Hình 23. Biểu đồ Heart Disease Frequency for Sex

- Và 1 số biểu đồ dựa theo Sex và Target cho các cột Slope, FBS, Chest Pain Type:





- Mục đích của Data Exploration: là tiền xử lý dữ liệu để ta hiểu để ta có thể hiểu rõ được dữ liệu này, làm rõ vấn đề ở đâu rồi đồng bộ chúng để để dữ liệu train dữ liệu 1 cách dễ dàng hơn.

### 3.4. Train model

Sau khi chia tập dữ liệu thành 8:2, thì ta tiến hành train model.

Train model:

- Sử dụng thư viện skicit-learn để gọi RandomForestClassifier.
- Các siêu tham số hyperparameter được chọn để tối ưu hóa hiệu suất của model.

```
[128] #Use the model Random Forest Classifier with hypermarameter - best optimization
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(bootstrap = True, max_depth=70, max_features = 'auto',
                              min_samples_leaf= 4,
                              min_samples_split = 10,
                              n_estimators= 400)
```



```
[129] model.fit(X_train, Y_train)

RandomForestClassifier(max_depth=70, min_samples_leaf=4, min_samples_split=10,
                       n_estimators=400)
```

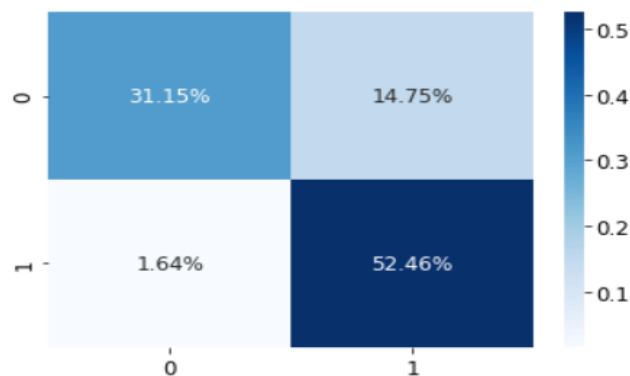
```
[179] Y_predict = model.predict(X_test)
```

- Sử dụng Confusion\_matrix rõ hơn về việc các điểm dữ liệu được phân loại đúng/sai như thế nào và đánh giá độ chính xác của model.

```
[131] confusion_matrix(Y_test, Y_predict)

array([[19,  9],
       [ 1, 32]])
```

```
[132] cm = confusion_matrix(Y_test, Y_predict)
sns.heatmap(cm/np.sum(cm),annot=True,fmt=".2%",cmap='Blues')
plt.show()
```



- Độ chính xác của model là : 83.6066%

```
[182] ac = accuracy_score(Y_test,Y_predict)
print('Random Forest Classifier Accuracy Score on Confusion_matrix: ',ac*100,'%')

Random Forest Classifier Accuracy Score on Confusion_matrix: 83.60655737704919 %
```

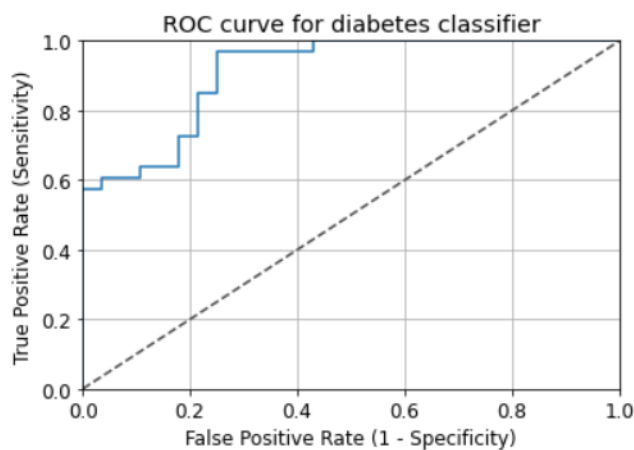
- Ta cũng sử dụng ROC curve để đánh giá kết quả dự đoán của model tốt hơn.

```

✓ [183] Y_pred_quant = model.predict_proba(X_test)[: , 1]
1 fpr, tpr, thresholds = roc_curve(Y_test, Y_pred_quant)
iây

fig, ax = plt.subplots()
ax.plot(fpr, tpr)
ax.plot([0, 1], [0, 1], transform=ax.transAxes, ls="--", c=".3")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.rcParams['font.size'] = 12
plt.title('ROC curve for diabetes classifier')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)

```



- Độ chính xác của model khi dùng ROC curve: 91.0173%

```

✓ 0 aucs = auc(fpr, tpr)
giây print('Random Forest Classifier Accuracy Score on ROC curve: ',aucs*100,'%')

Random Forest Classifier Accuracy Score on ROC curve: 91.01731601731602 %

```

- **Kết luận:** Với hyperparameter được chọn khi xài RandomForestClassifier() thì mô hình đã được tối ưu hóa, giúp tăng độ chính xác thông qua Confusion\_matrix và ROC curve.

### 3.5. Thực nghiệm siêu tham số cho bài toán

Ta có bảng có tham số được dùng để giúp model thêm tối ưu hơn, ở đây chúng tôi sẽ dùng các hyperparameter (siêu tham số). Dựa vào cơ sở lý thuyết về hyperparameter đã nói ở chương 2.5.

Accuracy (model RDF)	max_ depth	min_samples_ split	max_leaf_ nodes	min_samples_ leaf	n_estimators	max_features
81.97 %	70	10	none	4	400	none
83.61 %	80	20	25	5	400	7
81.97%	60	50	100	2	100	3
80.83 %	50	60	50	3	300	5
78.69%	40	80	150	6	200	4
72.13%	30	100	200	7	250	8
73.77%	100	150	250	8	350	9
81.97%	90	8	300	9	450	6
85.25%	200	25	500	3	500	2

- Từ bảng thực nghiệm cho model RandomForestClassifier() trên ta rút ra được các khoảng mà các siêu tham số nên chọn trong khi thực hiện 1 model RandomForestClassifier():

- + n\_estimators: [100,400,500] - số lượng cây trong khoảng trước
- + max\_features: [2,3] - số lượng tối đa các tính năng được xem xét để tách một nút
- + max\_depth [70,80,90,200] - số cấp tối đa trong mỗi cây quyết định
- + min\_samples\_split: [8,10,20,25] - số điểm dữ liệu tối thiểu được đặt trong một nút trước khi nút được tách
- + min\_samples\_leaf: [3,4,5] - số điểm dữ liệu tối thiểu được phép trong một nút lá.

### 3.6. Test dự đoán

Bước cuối cùng của bài toán này, xây dựng một cách test dự đoán về model, tôi sẽ sử dụng một số chỉ số phù hợp với các cột dữ liệu giống như bảng sau:

	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

- Tạo 1 danh sách input dữ liệu đầu vào (chỉ số phù hợp như bảng trên), lấy 1 số giá trị phù hợp ngẫu nhiên trừ cột target, sau đó thực hiện một số xử lý trên đó trước khi ta dự đoán.
- Thay đổi dữ liệu đầu vào input\_data() thành 1 mảng numpy.
- Reshape lại 1 mảng numpy đó để nó có thể predict 1 giá trị duy nhất (0 or 1).

```
[ ] input_data = (38,1,0,119,250,0,0,186,0,1.5,1,0,3)

#change the input_data to a numpy array
input_data_new = np.asarray(input_data)

#reshape the numpy array as we are predicting for only on instance
input_data_reshape = input_data_new.reshape(1, -1)

prediction = model.predict(input_data_reshape)

print(prediction)
```

[0]

- **Kết quả:**

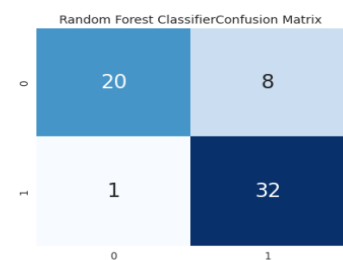
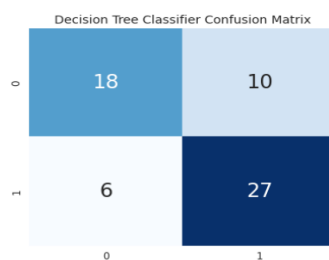
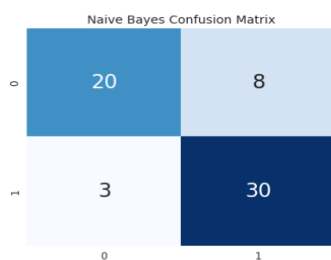
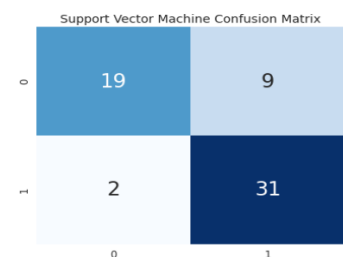
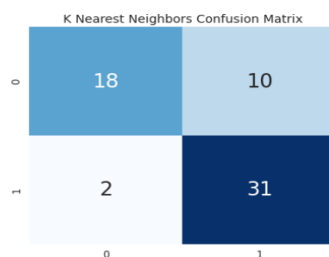
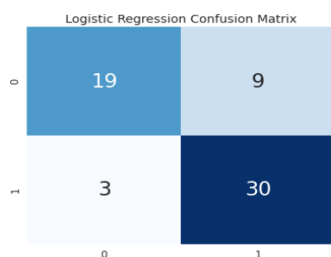
```
[ ] #Test the second of the row data in table heart disease (input_data = (38,1,0,119,250,0,0,186,0,1.5,1,0,3))
if (prediction[0] == 0):
    print("The person doesn't have a Heart Disease.")
else:
    print('The person has a Heart Disease.')
```

The person doesn't have a Heart Disease.

### 3.7 So sánh model với 1 số model khác

Model	Random Forest Classifier	Logistic Regression	KNN (max k)	Naive Bayes	SVM	Decision Tree Classifier
<b>Accuracy</b>	<b>85.25 %</b>	80.33 %	<b>86.89 %</b>	81.97 %	81.97 %	73.77 %

Các mô hình của chúng tôi hoạt động tốt nhưng tốt nhất trong số đó là KNN 86,89% và Random Forest Classifier với độ chính xác 85,25%. Hãy xem Confusion Matrix của các model.



## **Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

### **4.1 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 Classification. 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. Tuy nhiên thuật toán vẫn còn tồn tại một số nhược điểm cần phải được cải thiện sau này. Có thể sau này ta có thể cải thiện thuật toán bằng nhiều phương pháp khác và chúng tôi hi vọng 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 một thuật toán hiệu quả, chính xác cao, cần được đầu tư phát triển cho các ứng dụng thực tiễn và đời sống. Bản thân đã hoàn thành báo cáo đồ án tuy nhiên vẫn còn nhiều thiếu sót, cần đào sâu nghiên cứu và trau dồi thêm nhiều kiến thức.

### **4.2 Hướng phát triển**

Trong thời gian tới, đề tài có thể mở rộng như tối ưu thuật toán Random Forest trên các tham số khác như phương pháp lấy mẫu, cây phân lớp hoặc phát triển các ứng dụng có ích cho đời sống xã hội. Việc chuyển từ thuật toán sang ứng dụng cụ thể có thể gặp nhiều khó khăn, vì vậy cần có thời gian để phát triển ứng dụng, cải biến cấu trúc dữ liệu, giảm thời gian thực thi để nâng cao khả năng ứng dụng của thuật toán Random forest.

## TÀI LIỆU THAM KHẢO

- [1] Nguyễn Thành Đô, Tìm hiểu và tối ưu thuật toán Random Forest , ĐH Nha Trang
- [2] <https://viblo.asia/p/phan-lop-bang-random-forests-trong-python-djeZ1D2QKWz>
- [3] <https://machinelearningcoban.com/2017/08/31/evaluation/>
- [4] Gilles Louppe, Understanding random forest, Univesity of Liege, France
- [5] Leo Breiman, Random Forest, University of California, Berkeley, CA 9472, January 2001.
- [6] Eric Debreuve, An introduction to random forests, University Nice Sophia Antipolis / CNRS / InriaLabs: I3S / Inria CRI SA-M / iBV, 2011.
- [7] J. Ross Quinlan ,C4.5 Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning), 1993.
- [8] [https://en.wikipedia.org/wiki/Decision\\_tree](https://en.wikipedia.org/wiki/Decision_tree)
- [9] [https://en.wikipedia.org/wiki/Random\\_forest](https://en.wikipedia.org/wiki/Random_forest)
- [10] <http://en.wikipedia.org/wiki/AdaBoost>
- [11] <https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/?fbclid=IwAR3Ncz7qVKtqtK6WwUwPeNpJpWZ74xigGVzY8aUmEieILRd1Sy9u5BJL2T0>
- [12] <https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/>
- [13] <https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74>