ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT

BÁO CÁO QUÁ TRÌNH
PHÂN TÍCH DỮ LIỆU VỚI R/PYTHON

ĐỀ TÀI: TÌM HIỂU THUẬT TOÁN RANDOM FOREST

Giảng viên hướng dẫn	:	ThS. Nguyễn Phát Đạt
Lớp học phần	:	222IS2901
Nhóm thực hiện	:	Nhóm 11

TP.Hồ Chí Minh, tháng 02 năm 2023

DANH SÁCH THÀNH VIÊN NHÓM 11

STT	Họ và tên	MSSV	Chức vụ
1	Đinh Văn An	K204110855	Nhóm trưởng
2	Nguyễn Gia Hưng	K204110568	Thành viên
3	Phạm Nguyễn Hiền Phương	K204110579	Thành viên
4	Lê Thị Hồng Xuân	K204110591	Thành viên
5	Phạm Thị Minh Hòa	K204110567	Thành viên

MỤC LỤC

DANH MỤC HÌNH ẢNH

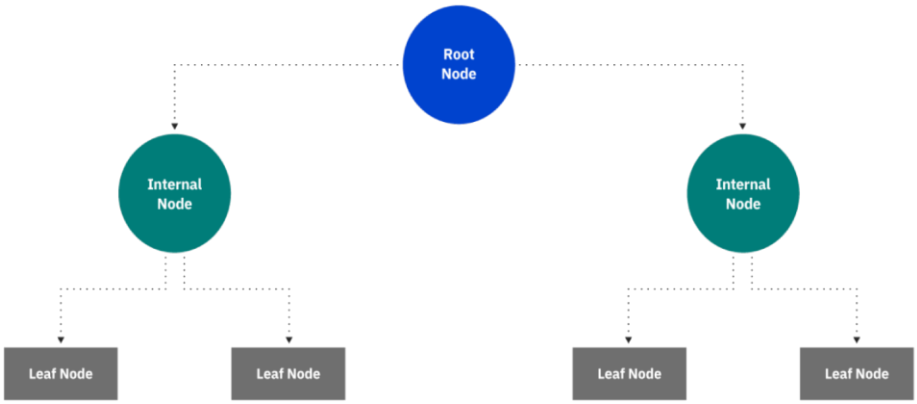
DANH MỤC BẢNG

CHƯƠNG 1: TỔNG QUAN VỀ LÝ THUYẾT

1.1. Tổng quát về Decision Tree

1.1.1 Định nghĩa

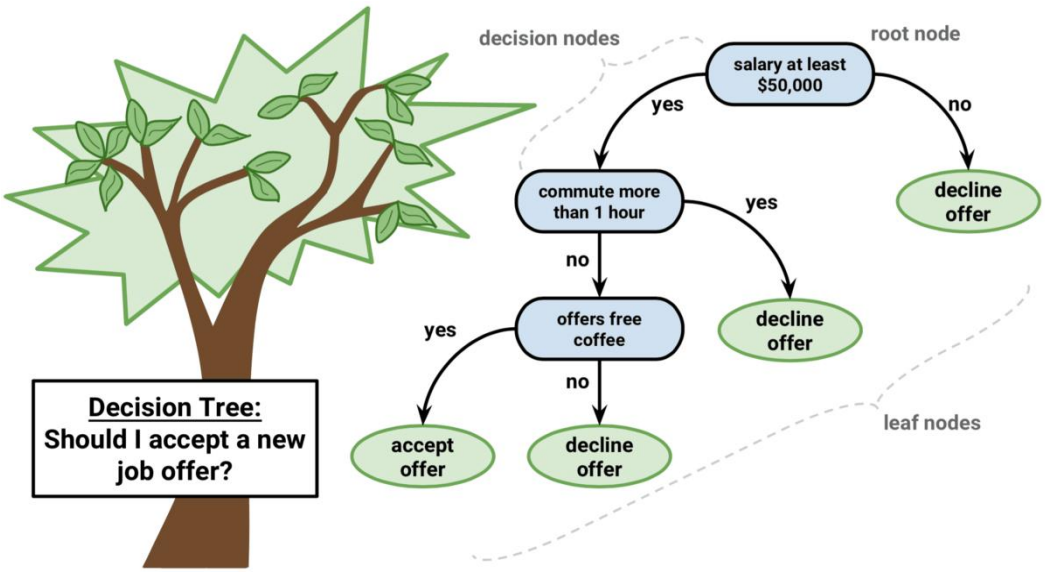
Decision Tree là một phương pháp học có giám sát phi tham số được sử dụng để phân loại và hồi quy. Mục tiêu là tạo ra một mô hình dự đoán giá trị của biến mục tiêu bằng cách học các quy tắc quyết định đơn giản được suy ra từ các đặc trưng dữ liệu. Nó có cấu trúc cây, phân cấp, bao gồm nút gốc, các nhánh, nút bên trong và nút lá.



Hình 1.1: Sơ đồ minh họa decision tree

(Nguồn ảnh: What is a Decision Tree | IBM; 1.10. Decision Trees — scikit-learn 1.2.1 documentation)

Sơ đồ bên trên cho thấy, một decision tree bắt đầu bằng nút gốc, nút này không có nhánh nào hướng đến. Các nhánh đi ra từ nút gốc sau đó sẽ đi vào các nút trong (không là nút gốc và nút lá), còn gọi là nút quyết định. Dựa trên các đặc điểm có sẵn, cả hai loại nút đều tiến hành đánh giá để tạo ra các tập hợp đồng nhất, được ký hiệu bằng các nút lá. Các nút lá biểu diễn tất cả các kết quả có thể xảy ra trong tập dữ liệu. Ví dụ, giả sử một người đang cố gắng đánh giá liệu có nên chấp nhận lời mời công việc hay không, họ có thể sử dụng các quy tắc quyết định sau để đưa ra quyết định:



Hình 1.2: Ví dụ decision tree cho việc chấp nhận lời mới công việc

(Nguồn ảnh: Decision Trees (brookewenig.com))

Loại cấu trúc sơ đồ dòng này cũng tạo ra một biểu diễn dễ hiểu của quá trình ra quyết định, giúp các nhóm khác nhau trong tổ chức hiểu rõ hơn về lý do tại sao một quyết định đã được đưa ra.

1.1.2 Ưu Điểm

Dễ hiểu: Logic Boolean và biểu diễn hình ảnh của decision tree làm cho chúng dễ hiểu và áp dụng hơn. Tính phân cấp của decision tree cũng làm cho việc xem xét các thuộc tính quan trọng hơn dễ dàng hơn, điều này thường không rõ ràng với các thuật toán khác, như mạng nơ-ron.

Không yêu cầu chuẩn bị dữ liệu: decision tree có nhiều đặc điểm, làm cho nó linh hoạt hơn so với các phân loại khác. Nó có thể xử lý nhiều loại dữ liệu - ví dụ: giá trị rời rạc hoặc liên tục và giá trị liên tục có thể được chuyển đổi thành giá trị phân loại thông qua việc sử dụng ngưỡng. Thêm vào đó, nó cũng có thể xử lý giá trị bị thiếu.

Linh hoạt hơn: decision tree có thể được sử dụng cho cả các tác vụ phân loại và hồi quy, làm cho nó linh hoạt hơn một số thuật toán khác. Nó cũng không nhạy cảm với các mối quan hệ cơ bản giữa các thuộc tính; điều này có nghĩa là nếu hai biến tương quan cao, thuật toán sẽ chỉ chọn một trong các đặc trưng để phân chia.

1.1.3 Nhược điểm

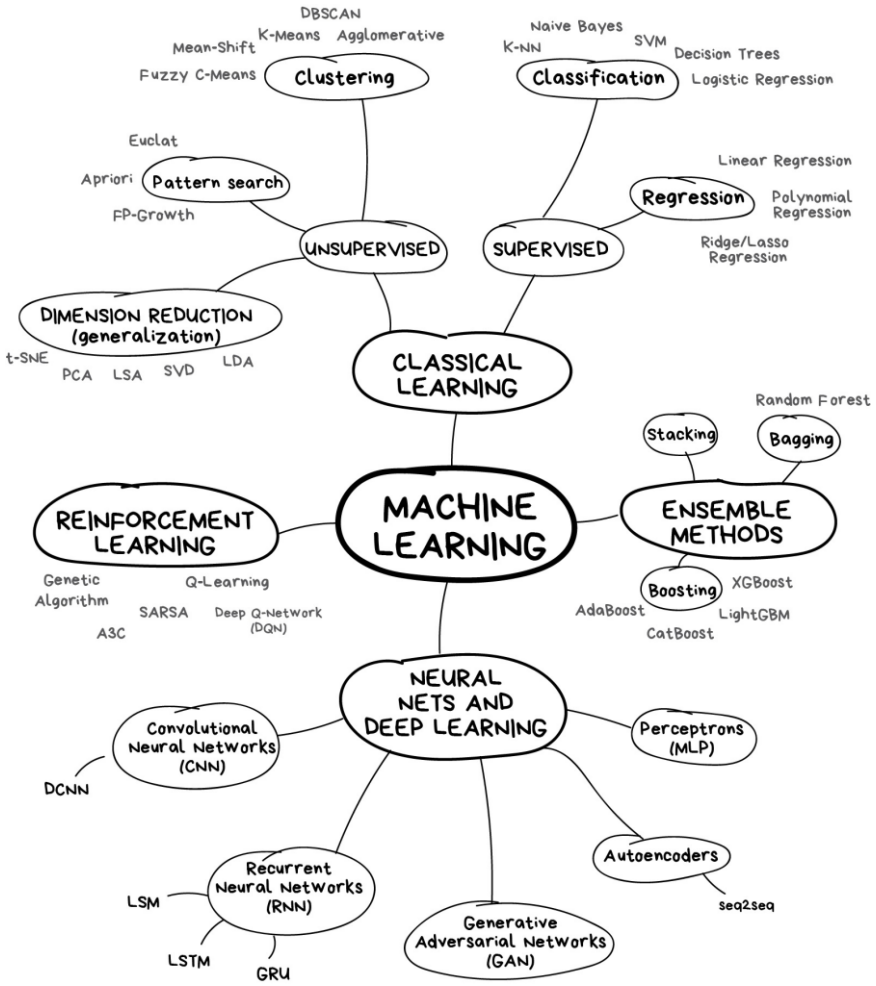
Ước lượng có độ biến thiên cao: Sự thay đổi nhỏ trong dữ liệu có thể tạo ra một decision tree hoàn toàn khác. Bagging, hay việc lấy trung bình các ước lượng, có thể là một

phương pháp để giảm sự biến thiên của decision tree. Tuy nhiên, phương pháp này có giới hạn vì có thể dẫn đến các dự đoán trùng lắp cao.

Tốn kém hơn: Vì decision tree tiến hành tìm kiếm tham lam trong quá trình xây dựng, nó có thể tốn kém hơn để huấn luyện so với các thuật toán khác.

Có xu hướng overfitting: decision tree phức tạp thì có xu hướng bị overfitting và không thể làm tốt việc tổng quát hóa dữ liệu mới. Trường hợp này có thể tránh được thông qua các quy trình pre-pruning hoặc post-pruning. Pre-pruning là dừng sự phát triển của cây khi không đủ dữ liệu, trong khi post-pruning loại bỏ các nhánh con có dữ liệu không đủ sau khi cây được xây dựng.

1.2 Tìm hiểu về Ensemble Methods và Bagging



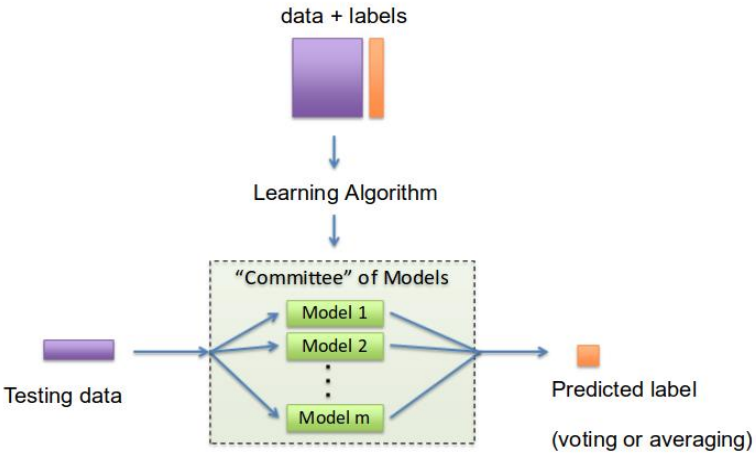
Hình 1.3: Tổng quan các loại mô hình trong học máy

(Nguồn ảnh: [Machine Learning for Everyone — In simple words. With real-world examples. Yes, again — vas3k])

1.2.1 Ensemble method

1.2.1.1 Định nghĩa

Ensemble method (Phương pháp tổng hợp) là các kỹ thuật nhằm cải thiện độ chính xác của kết quả và khả năng dự đoán các model bằng cách kết hợp nhiều model thay vì sử dụng một model duy nhất. Kết hợp các model cải thiện đáng kể độ chính xác của kết quả.



Hình 1.4: Tổng quan về Ensemble method

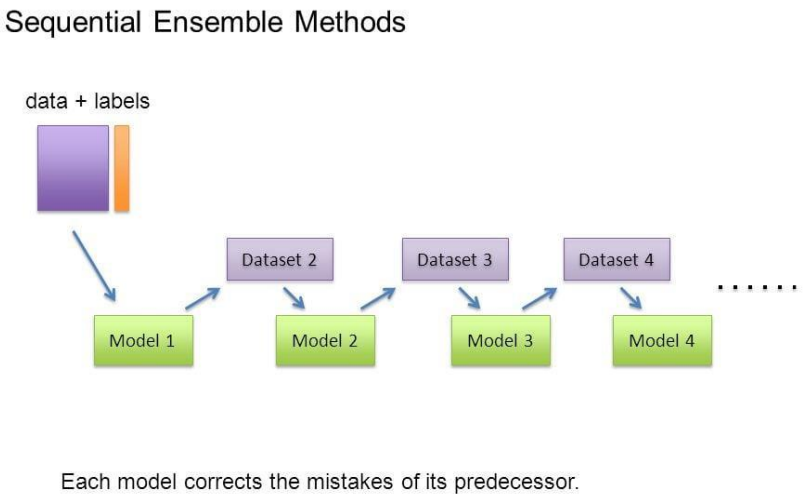
Các Ensemble method là phương pháp lý tưởng cho các bài toán hồi quy (classification) và bài toán phân loại (regression) bằng cách làm giảm độ lệch (reduce bias) và phương sai (reduce variance) để tăng độ chính xác của các model. Bởi vì không có một model nào là hoàn hảo khi đi độc lập, chúng sẽ xuất hiện hiện tượng high bias hoặc high variance. Bias và variance tồn tại bổ sung cho nhau, vì vậy sự tăng của bias sẽ làm giảm variance và ngược lại. Do đó, để tạo một model tốt cần phải tạo sự cân bằng giữa bias và variance, đây được gọi là đánh đổi độ lệch và phương sai (bias-variance tradeoff).

- Reduce variance: nếu các tập huấn luyện hoàn toàn độc lập, reduce variance sẽ giúp tính trung bình tập hợp vì điều này sẽ giảm variance mà không ảnh hưởng đến bias và giảm độ nhạy đối với các điểm dữ liệu riêng lẻ.
- Reduce bias: đối với các model đơn giản, trung bình của các mô hình có công suất (capacity) lớn hơn nhiều so với mô hình độc lập. Các model lấy trung bình có thể giảm đáng kể bias bằng cách tăng công suất và kiểm soát phương sai bằng cách khớp từng thành phần một.

1.2.1.2 Phân loại

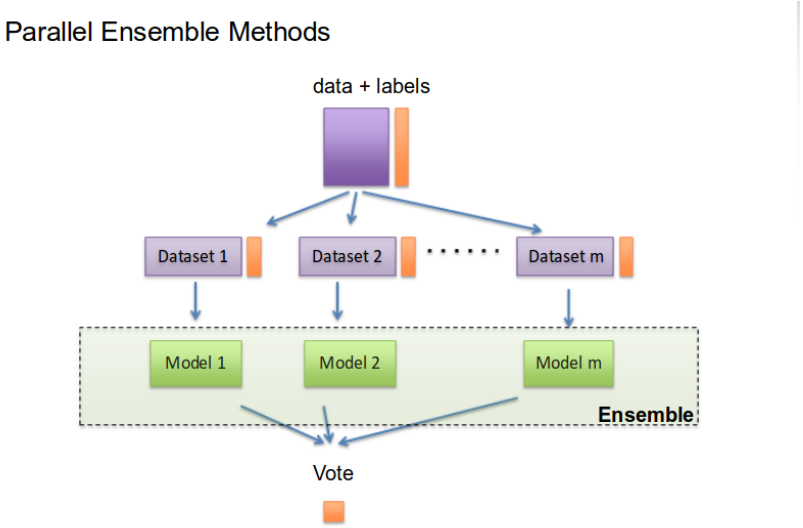
Các Ensemble method được chia thành 2 nhóm chính: tuần tự theo chuỗi (sequential ensemble method) và tuần tự song song (parallel ensemble method).

- Sequential ensemble method: Phương pháp này dựa trên sự phụ thuộc của lần học tiếp theo vào lần học trước. Sau mỗi bước training, các trọng số được gán lại. Data được phân loại sai sẽ tăng trọng số để nhấn mạnh những data khó. Bằng cách này, các lần học tiếp theo sẽ tập trung vào dữ liệu khó trong quá trình training, phương pháp này gọi là Boosting.



Hình 1.5: Tổng quan về Sequential ensemble method

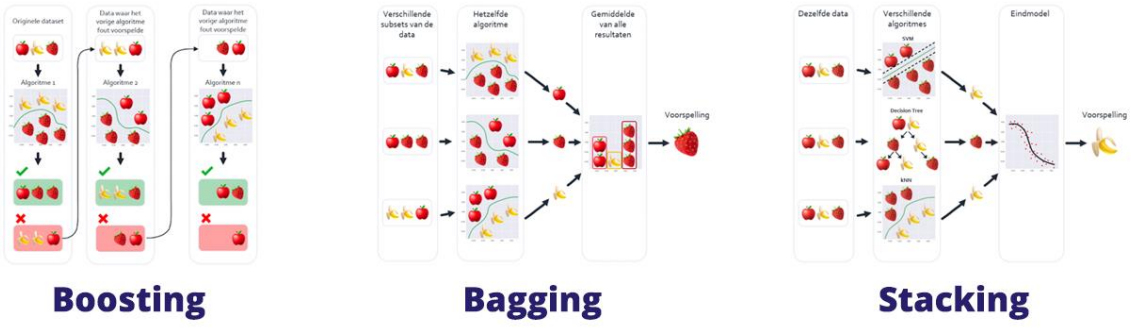
- Parallel ensemble method: Phương pháp này tận dụng kết quả dự đoán độc lập của từng model, sử dụng voting method tìm kiếm dự đoán có số phiếu chọn cao nhất và chọn dự đoán đó. Một ví dụ điển hình cho phương pháp này là Random forest.



Hình 1.6: Tổng quan về Parallel ensemble method

Các Ensemble method được sử dụng nhiều nhất hiện nay là bagging, boosting và stacking. Trong đó, bagging được sử dụng để giảm variance, boosting để giảm bias và stacking được dùng để cải thiện dự đoán của bài toán.

- **Bagging:** Xây dựng một lượng lớn các model (thường là cùng loại) trên những subsamples khác nhau từ tập training dataset. Những model này sẽ được training độc lập và song song với nhau nhưng đầu ra của chúng sẽ được trung bình cộng để cho ra kết quả dự đoán cuối cùng đối với regression hay chọn kết quả có số phiếu bầu cao nhất đối với classification.
- **Boosting:** Xây dựng một lượng lớn các model (thường là cùng loại). Mỗi model sau sẽ học cách sửa những lỗi của model trước từ đó tạo thành một chuỗi các model mà model sau sẽ tốt hơn model trước bởi trọng số (weight) được cải thiện qua mỗi model (cụ thể ở đây là trọng số của những dữ liệu dự đoán đúng sẽ không đổi, còn trọng số của những dữ liệu dự đoán sai sẽ được tăng thêm). Kết quả cuối cùng sẽ là kết quả trả về tốt nhất.
- **Stacking:** xây dựng một số models (thường là khác loại) và một meta model (supervisor model), mô hình này sẽ học cách kết hợp kết quả dự báo của một số mô hình một cách tốt nhất.



Hình 1.7: Phân biệt Boosting, Bagging, Stacking

(Nguồn ảnh: Ensemble Methods: the 3 methods simply explained)

1.2.2 Bagging

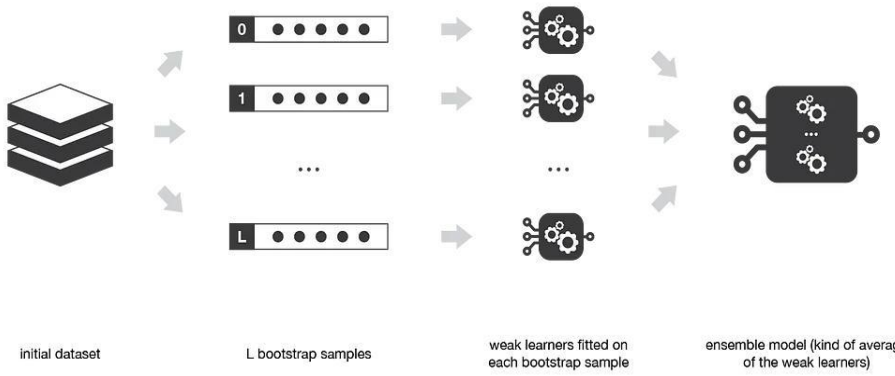
Bagging hay được biết đến là dạng viết tắt của Bootstrap aggregating, thường được áp dụng trong các bài toán phân loại và hồi quy. Nó làm tăng độ chính xác của các mô hình, giúp giảm variance ở mức độ lớn. Việc giảm variance làm tăng độ chính xác, làm giảm tình

trạng overfitting của một số model dự đoán. Quy trình diễn ra của Bagging gồm 3 bước cơ bản sau:

Bước 1. Bootstrapping: Bagging sử dụng kỹ thuật lấy mẫu bootstrapping để tạo đa dạng các mẫu (sample). Phương pháp lấy mẫu lại này tạo ra các tập hợp con khác nhau của training dataset bằng cách chọn ngẫu nhiên các điểm dữ liệu (data point) và thay thế (replacement).

Bước 2. Parallel training: Các bootstrap samples này sau đó được đào tạo độc lập và song song với nhau bằng cách sử dụng weak hoặc base learners.

Bước 3. Aggregation: Cuối cùng, tùy thuộc vào nhiệm vụ (tức là hồi quy hoặc phân loại) sẽ có cách riêng để lựa chọn đầu ra tối ưu hơn.[1]



Hình 1.8: Quy trình cơ bản của Bagging

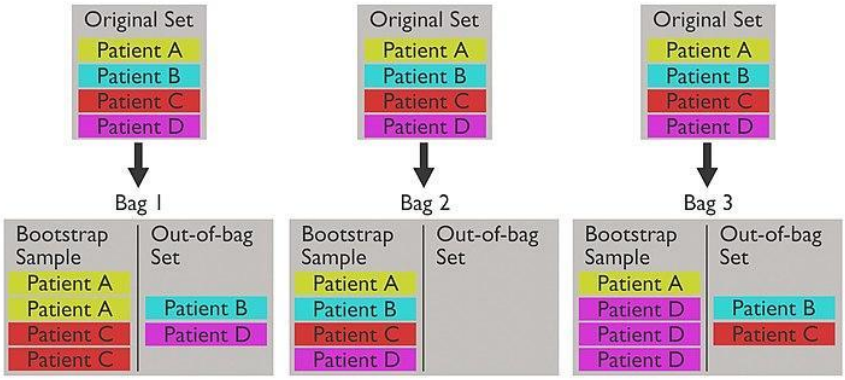
(Nguồn ảnh: Ensemble methods: bagging, boosting and stacking | by Joseph Rocca | Towards Data Science)

1.2.2.1 Bootstrapping

Bootstrapping là một phương pháp rất nổi tiếng trong thống kê được giới thiệu bởi Efron vào năm 1979 [15]. Phương pháp này được thực hiện như sau: từ một quần thể ban đầu lấy ra một mẫu L = (x1, x2,..,xn) gồm n thành phần để tính toán các tham số mong muốn. *Bootstrapping* trong bagging là một kỹ thuật đặc biệt trong việc xây dựng model. Kỹ thuật này được sử dụng để tạo ra các tập dữ liệu con ngẫu nhiên từ tập dữ liệu huấn luyện ban đầu.

Đối với *bootstrapping*, một tập dữ liệu con được tạo ra bằng cách lấy một lượng lớn các mẫu ngẫu nhiên từ tập dữ liệu huấn luyện ban đầu. Các sample được lấy ra có thể có trùng lặp, điều này cho phép cùng một sample xuất hiện nhiều lần trong tập dữ liệu con.

Những dữ liệu mới này có phân phối giống nhau và gần như độc lập, kết quả cuối cùng sẽ không quá ảnh hưởng. Quá trình này được thực hiện nhiều lần để tạo ra các tập dữ liệu con khác nhau, mỗi tập dữ liệu con được sử dụng để huấn luyện với model độc lập và song song. Trong bagging, việc tính trung bình đầu ra các base model này cũng sẽ hỗ trợ làm giảm variance.



Hình 1.9: Sampling với replacement và out-of-bag dataset

(Nguồn ảnh: Out-of-bag_error)

Trong hình trên, ta có thể thấy sẽ xuất hiện một số dữ liệu được lấy trùng lặp trong Bootstrap sample và có một số dữ liệu trong tập dữ liệu gốc không có trong bootstrapped dataset. Những dữ liệu đó được gọi là **Out-of-bag dataset**. Trong các thuật toán Bagging, có một kỹ thuật quan trọng sử dụng các Out-of-bag samples này, **Out-of-bag error** (hay còn được gọi là Out-of-bag estimate), kỹ thuật này được dùng để đo lường lỗi hoặc hiệu suất của các model trong mỗi thời kỳ nhằm giảm thiểu số lượng lỗi của model khi kết thúc.[2]

Sơ lược Bootstrapping dưới dạng toán học:

Ta có, L bootstrap samples (tương ứng với L bộ dữ liệu) có kích thước B

$$\{z_1^1, z_2^1, ..., z_B^1\}, \{z_1^2, z_2^2, ..., z_B^2\}, ..., \{z_1^L, z_2^L, ..., z_B^L\} \qquad z_b^l \equiv b\text{-th observation of the } l\text{-th bootstrap sample}$$

Tương ứng với L bộ dữ liệu là L model "yếu".

$$w_1(.), w_2(.), ..., w_L(.)$$

[3]

Việc kết hợp các model "yếu" ta được một đầu ra của một model "mạnh" khắc phục điểm yếu của model "yếu". Quy trình này được gọi là aggregating và sẽ được trình bày rõ trong mục tiếp theo.

1.2.2.2 Aggregating

Sau quá trình training song song model được diễn ra, các model sẽ kết hợp lại với nhau tạo ra một đầu ra tối ưu. Như đã nhắc từ các phần trên, cách thức để đưa ra đầu ra của bài toán hồi quy và bài toán phân loại sẽ khác nhau.

Trong trường hợp hồi quy, giá trị trung bình được lấy cho tất cả các đầu ra được dự đoán bởi các individual classifiers; điều này được gọi là bỏ phiếu mềm (soft voting). Đối với bài toán phân loại, lớp nào có đa số phiếu bầu cao nhất được chấp nhận; điều này được gọi là bỏ phiếu cứng (hard voting) hoặc bỏ phiếu theo đa số (majority voting).

$$s_L(.) = \frac{1}{L}\sum_{l=1}^{L} w_l(.) \qquad \text{(simple average, for regression problem)}$$

$$s_L(.) = \arg\max_k [card(l|w_l(.) = k)] \qquad \text{(simple majority vote, for classification problem)}$$

1.2.2.3 Công thức toán bagging

Chúng ta có Y là biến phụ thuộc (response variables), P là biến phụ thuộc (predictor variables), $X=X_1, X_2,...,X_p$ với N là số dòng của Dataset (records). Thuật toán của phương pháp được trình bày dưới dạng sau:

1. Khởi tạo M, số lượng model phù hợp và n là số lượng records để chọn với (n<N). Đặt lần lặp m = 1.
2. Lấy bootstrap resample (với sampling có thay thế) của n records từ dữ liệu huấn luyện để tạo thành subsample Y_m và X_m.
3. Huấn luyện một mô hình sử dụng Y_m và X_m để tạo một bộ quy tắc quyết định $\hat{f}_m(\mathbf{X})$.
4. Tăng bộ đếm model m = m + 1. Nếu m <= M, chuyển sang bước 2.

Trong trường hợp dự đoán xác suất Y = 1, bagged estimate được đưa ra bởi:

$$\hat{f} = \frac{1}{M}\left(\hat{f}_1(\mathbf{X}) + \hat{f}_2(\mathbf{X}) + \cdots + \hat{f}_M(\mathbf{X})\right)$$

Hình 1.10: Thuật toán của Bagging

(Nguồn ảnh:Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and ... - Peter Bruce, Andrew Bruce, Peter Gedeck - Google Sách)

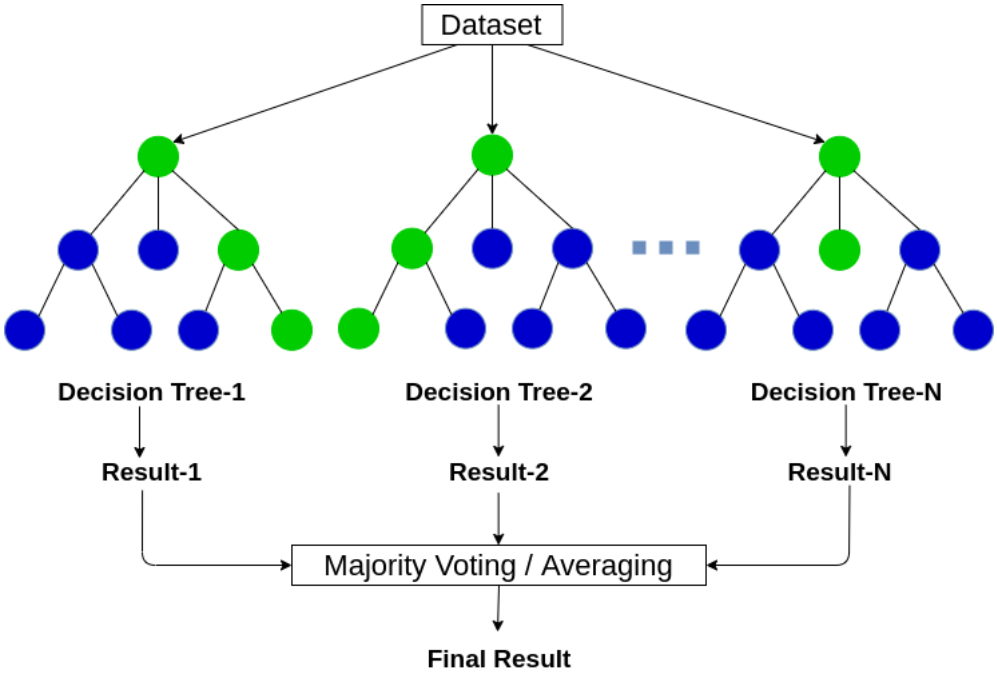
1.3 Tìm hiểu Random Forest

1.3.1 Giới thiệu về Random Forest

1.3.1.1 Định nghĩa

Random forest (bởi Leo Breiman, 2001) là một phương pháp thống kê mô hình hóa bằng máy (machine learning statistic) dùng để phục vụ các mục đích phân loại, tính hồi quy và các nhiệm vụ khác bằng cách xây dựng nhiều cây quyết định (Decision tree).

Random Forest có khả năng tìm ra thuộc tính nào quan trọng hơn so với những thuộc tính khác. Trên thực tế, nó còn có thể chỉ ra rằng một số thuộc tính là không có tác dụng trong Decision tree.[4]



Hình 1.11: Random forest

(Nguồn ảnh: random-forests-understanding)

Từ ảnh trên, có thể thấy Random forest là một phần mở rộng của Bagging. Nó còn khắc phục được lỗi của Bagging là kết quả dự báo có tính tương quan và greedy của decision tree. Ở random forest là một model được tạo từ nhiều Decision tree. Random forest có hai lý do chính để nó có tên gọi "Random"

- Lấy ngẫu nhiên các sampling từ các training data point trong quá trình xây tree.
- Các Subset ngẫu nhiên của các feature được xem xét khi tách node.

1.3.1.2 Thuật toán Random forest

Với b=1 đến B ta có:

a. Tạo ra một bootstrap sample Z* có kích thước N từ training data.
b. Xây một random forest T_b thành bootstrapped data, bằng cách lặp lại quy trình các bước sau cho từng node đầu cuối của tree, cho đến khi đạt được kích thước node tối thiểu n_{min}.
    i. Chọn ngẫu nhiên m biến từ p biến
    ii. Chọn biến/điểm phân chia (split-point) tốt nhất trong số biến m
    iii. Tách node thành 2 node con.

Output của tập hợp các tree $\{T_b\}_1^B$. Để tạo một dự đoán tại một điểm mới x

$$\textit{Regression: } \hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x).$$

$$\textit{Classification: Let } \hat{C}_b(x) \text{ be the class prediction of the } b\text{th random-forest tree. Then } \hat{C}_{\text{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B.$$

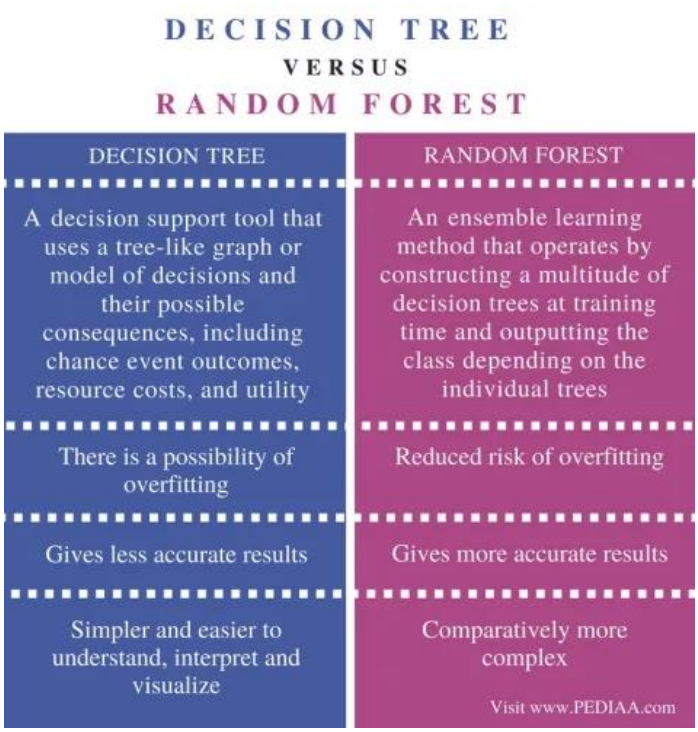
[5]

Để có kết quả đầu ra cuối cùng của Random forest, tất cả các đầu ra sẽ được major voting trong bài toán phân loại và tính trung bình cộng trong bài toán hồi quy.

1.3.1.3 Mục đích của thuật toán

Decision tree mô hình hóa các mối quan hệ phức tạp, nhưng nó lại quá phụ thuộc vào training data, bất kỳ sự xuất hiện của một data mới cũng có thể gây ra hiện tượng Overfitting ở decision tree. Mặc dù chúng đào tạo các mô hình thường chính xác, nhưng decision tree thường cho thấy mức độ biến thiên lớn giữa các mẫu dữ liệu khác nhau từ cùng một tập dữ liệu. Kết quả là, decision tree được biết đến với việc thể hiện variance cao và bias thấp.

Vì vậy, mục tiêu của random forest là lấy một tập hợp các decision tree có variance cao, bias thấp và cố gắng cân bằng lại variance và bias nhằm hạn chế tình trạng overfitting. Bằng cách tổng hợp các đầu ra khác nhau của các decision tree riêng lẻ, random forest làm giảm variance có thể gây ra lỗi trong decision tree. Thông qua, major voting chúng ta có thể tìm thấy đầu ra trung bình được đưa ra bởi hầu hết các cây riêng lẻ. Điều này làm giảm variance để model ít có khả năng tạo ra kết quả xa giá trị thực hơn.



Hình 1.12: Sự khác biệt giữa decision tree và random forest

(Nguồn ảnh: Difference-between-decision-tree-and-random-forest)

1.3.2 Phân tích Random Forest Method

1.3.2.1 Quy trình của Random Forest Model

Để diễn tả quy trình của Random forest model, chúng ta sẽ lấy ví dụ của một bài toán dự đoán phân loại. Ở bài toán này, sẽ có 4 feature bao gồm chest pain (đau ngực), good blood circ (tuần hoàn máu tốt), blocked arteries (tắc động mạch), weight (cân nặng) để dự đoán bệnh nhân có khả năng bị bệnh tim hay không.

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Hình 1.13: Tập dữ liệu gốc

(Nguồn ảnh:Random Forests Part 1 - Building, Using and Evaluating)

Từ tập dữ liệu gốc, tạo các bootstrapped dataset có kích thước bằng với dữ liệu gốc, lưu ý rằng tất cả các sample trong một bootstrapped dataset đều được chọn ngẫu nhiên vì vậy việc một sample có thể được chọn nhiều lần là được cho phép. Việc sử dụng bootstrapping trong random forest giúp mô hình tránh được overfitting, tăng tính đa dạng và ổn định của các Decision tree trong model. Nó cũng cho phép model ước tính các giá trị thiếu hoặc bị lỗi trong tập dữ liệu huấn luyện, giảm thiểu hiện tượng overfitting và tăng khả năng tổng quát hóa của mô hình.

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Hình 1.14: Bootstrapped dataset được tạo từ tập dữ liệu gốc

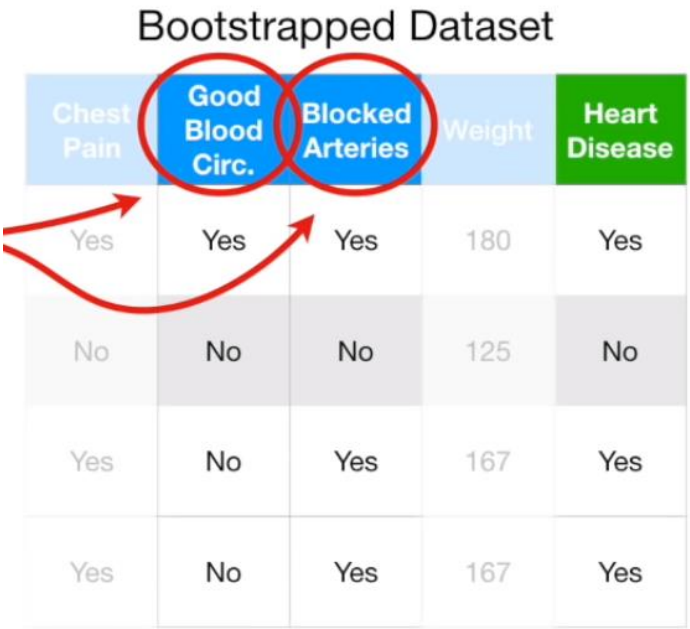
(Nguồn ảnh:Random Forests Part 1 - Building, Using and Evaluating)

Random forest thay đổi thuật toán theo cách mà cây con được học để các dự đoán kết quả từ tất cả các cây con có ít mối tương quan hơn. Random forest tinh chỉnh đơn giản hơn

các decision tree. Trong thuật toán CART, khi chọn split-point, thuật toán phải xem qua tất cả các biến (feature) và tất cả các giá trị của biến để chọn split-point tối ưu nhất. Random forest thay đổi quy trình này sao cho thuật toán học được giới hạn trong một mẫu ngẫu nhiên các tính năng (Random feature selection) để tìm kiếm. Vì vậy, để tạo được một decision tree từ Bootstrapped dataset, nhưng chỉ sử dụng tập hợp con ngẫu nhiên các biến (hoặc cột) trong mỗi bước. Số lượng random feature được chọn thường sẽ được tính như sau:

- Đối với Regression, giá trị mặc định tốt nhất: m=p/3
- Đối với Classification, giá trị mặc định tốt nhất: m= sqrt(p) [5]

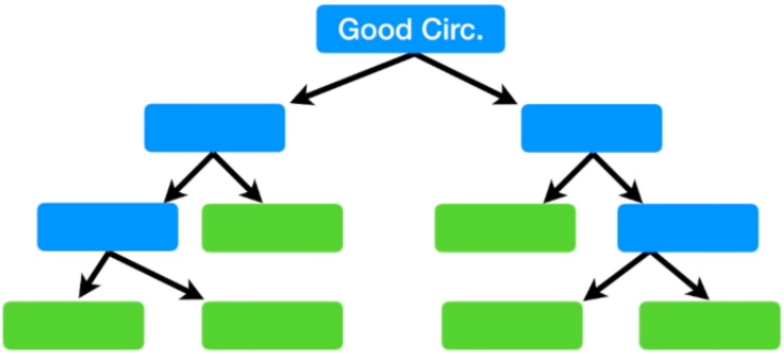
Trong đó m là số lượng random features selection có thể được tìm kiếm tại một split-point và p là số lượng biến đầu vào.



Hình 1.15: Các feature được sử dụng để xây decision tree sau khi random feature

(Nguồn ảnh:Random Forests Part 1 - Building, Using and Evaluating)

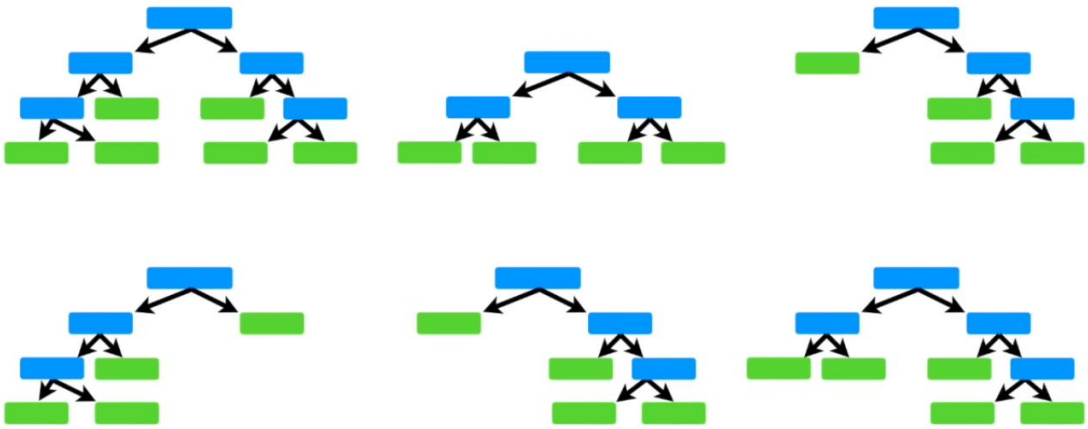
Sau khi đã chọn được random feature để làm split-point, chúng ta sẽ tiến hành tạo những decision tree được tinh giản.



Hình 1.16: Decision tree được tạo sau khi đã random feature

(Nguồn ảnh:Random Forests Part 1 - Building, Using and Evaluating)

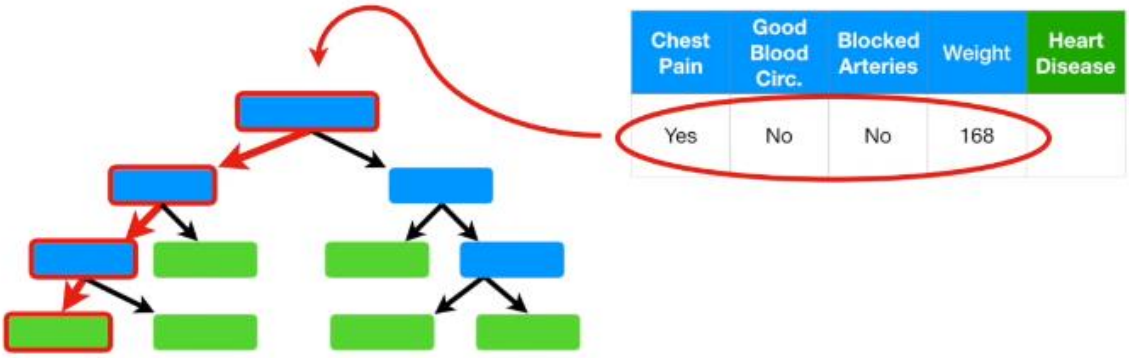
Lặp lại các bước trên và tạo ra những decision tree khác bằng bootstrapped dataset mới và chỉ dùng một tập nhỏ các biến ngẫu nhiên (random features). Sự đa dạng giúp cho random forest tối ưu hơn so với một decision tree, nó khắc phục được hiện tượng overfitting của decision tree. Số lượng tree càng lớn thì kết quả đầu ra càng cao và đi theo đó sự phức tạp của model cũng sẽ tăng lên vì vậy sẽ tốn nhiều tài nguyên cũng như thời gian. Vì vậy, nên cân nhắc số lượng decision tree sao cho vừa đủ với bài toán được đưa ra.



Hình 1.17: 6 decision tree mới được tạo ra từ bộ bootstrapped dataset khác nhau

(Nguồn ảnh:Random Forests Part 1 - Building, Using and Evaluating)

Trong ví dụ, có một bệnh nhân mới xuất hiện với các feature đã có data, yêu cầu đặt ra ở đây là dự đoán bệnh nhân này có bị bệnh tim hay không. Data của bệnh nhân sẽ được test trên từng decision tree và quá trình này sẽ diễn ra song song. Ở hình ảnh dưới, sau khi test data, kết quả thu được là bệnh nhân có nguy cơ bị bệnh tim.



Hình 1.18: Test data trên từng decision tree

(Nguồn ảnh:Random Forests Part 1 - Building, Using and Evaluating)

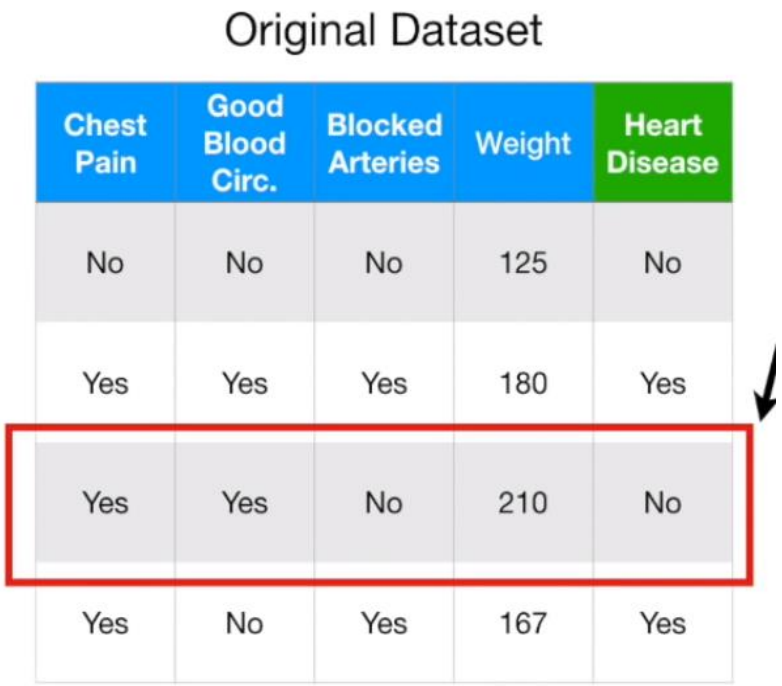
Lặp lại các bước với các decision tree còn lại, kết quả thu được sẽ được tổng hợp bằng cách tiến hành major voting. Đầu ra có sự xuất hiện nhiều sẽ được chọn làm kết quả đầu ra cuối cùng. Quy trình tương tự với quá trình aggregation của bagging, có thể thấy random forest là một phiên bản mở rộng của bagging.

Hình 1.19: Kết quả dự đoán của bệnh nhân sau khi major voting các đầu ra

(Nguồn ảnh:Random Forests Part 1 - Building, Using and Evaluating)

Nhắc lại, việc tạo ra các bootstrap sample (sampling với replacement), tạo sự xuất hiện trùng lặp của một số sample và một số sample trong tập dữ liệu gốc không được xuất hiện trong bootstrapped dataset. Những Out-of-bag sample này sẽ sẽ không nằm trong bootstrapped dataset để tạo decision tree tuy nhiên nó được tận dụng để làm một tính năng

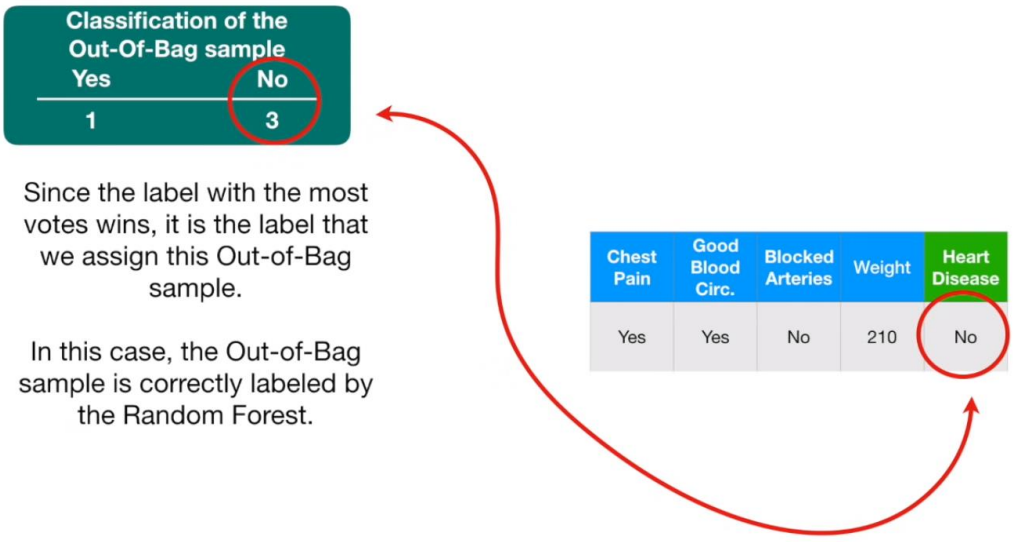
quan trọng trong Random forest. Đa phần, OOB sẽ chiếm ⅓ số lượng sample của tập dữ liệu ban đầu.



Hình 1.20: Out-of-bag sample của dataset sau khi tạo bootstrapped dataset đầu tiên

(Nguồn ảnh:Random Forests Part 1 - Building, Using and Evaluating)

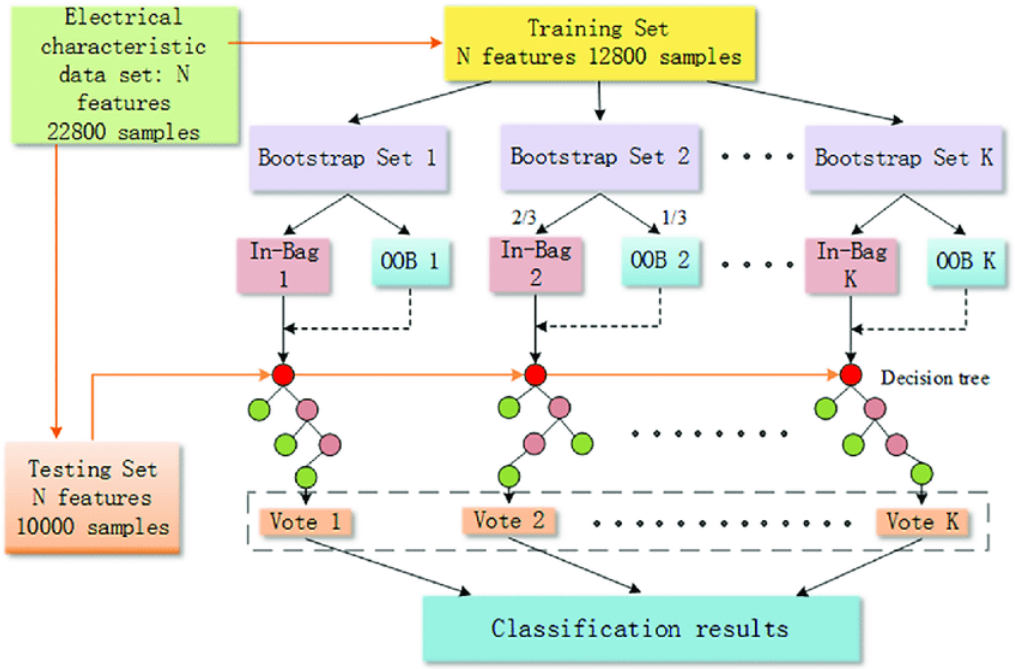
Test thử từng OOB sample này trên từng decision tree được tạo ra từ trước. Chúng ta có thể đo lường được mức độ chính xác của Random forest với tỉ lệ các Out-of-bag sample được phân loại chính xác. Tỷ lệ các Out-of-bag sample không được phân loại chính xác được gọi là Out-of-bag error (hay là Out-of-bag estimate).



Hình 1.21: Testing Out-of-bag sample trên từng decision tree

(Nguồn ảnh:Random Forests Part 1 - Building, Using and Evaluating)

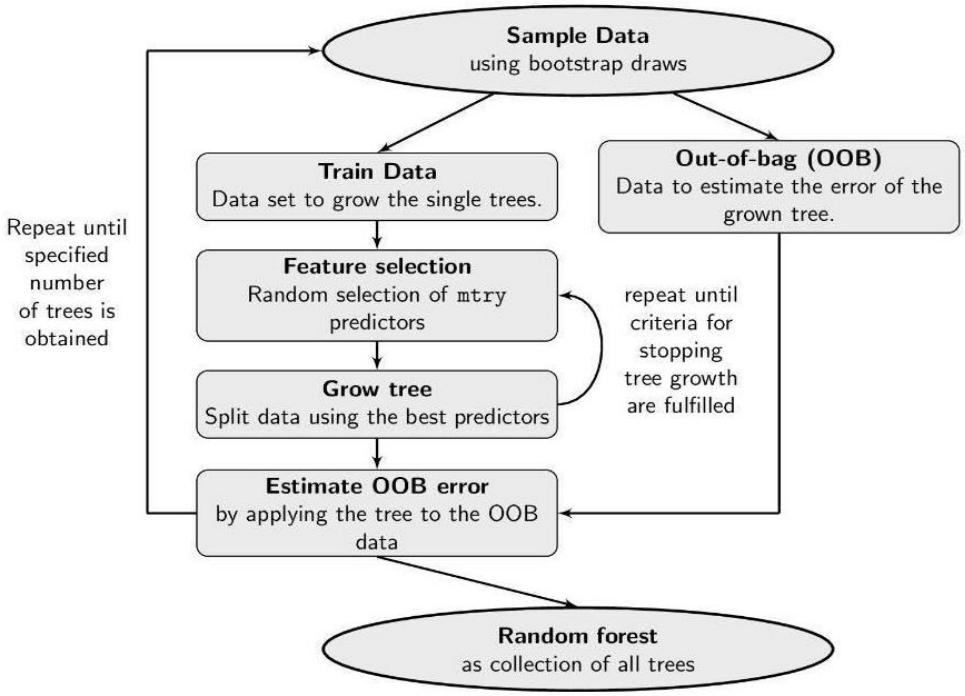
Nhắc lại, kỹ thuật OOB error này được dùng để đo lường lỗi hoặc hiệu suất của các model trong mỗi thời kỳ nhằm giảm thiểu số lượng lỗi của model khi kết thúc. Điều đó có nghĩa là model dưới cùng có nhiều lỗi hơn, thì OOB error cho model dưới cùng càng cao. Ngược lại với OOB error là OOB score, OOB score càng cao thì model cuối cùng càng ít lỗi.



Hình 1.22: Quy trình tổng quan của Random forest model

(Nguồn ảnh: Out-of-bag-oob-score-for-bagging-in-data-science)

Tóm lại, tổng thể quy trình diễn ra không quá phức tạp tuy nhiên nó đòi hỏi nhiều tài nguyên và thời gian để đưa ra một kết quả đầu ra tốt nhất.



Hình 1.23: Lưu đồ Random forest algorithm

(Nguồn ảnh: Overview of Random Forest Methodology and PracticalGuidance with Emphasis on Computational Biology andBioinformatics)

1.3.2.2 Variable importance

Sau quá trình training một random forest, chúng ta sẽ cần giải đáp một câu hỏi "Biến nào sẽ có vai trò quan trọng ảnh hưởng đến kết quả dự đoán của model nhất". Các biến có tầm quan trọng cao (variable importance) là động lực của kết quả và giá trị của chúng có tác động đáng kể đến các giá trị kết quả. Ngược lại, các biến có tầm quan trọng thấp có thể bị bỏ qua khỏi model, làm cho nó đơn giản hơn và nhanh hơn để phù hợp và dự đoán.[6]

Có hai thước đo tầm quan trọng được đưa ra cho mỗi biến trong random forest:

- Thước đo đầu tiên dựa trên độ chính xác giảm (decrease accuracy) bao nhiêu khi biến bị loại trừ. Phương pháp này được tính toán từ Out-of-bag sample (đây là phương pháp xác thực cross-validated).
- Thước đo thứ hai dựa trên sự giảm Gini impurity (decrease Gini) khi một biến được chọn để tách node. Đo lường mức độ tinh khiết của các node. Thước đo này được tính toán dựa trên training dataset nên nó có độ tin cậy ít hơn so với tính toán từ OOB data.

Với độ tin cậy thấp hơn nhưng tại sao vẫn sử dụng phép đo độ giảm Gini impurity? Ở RandomForest chỉ tính toán Gini impurity này: Gini impurity là sản phẩm phụ của thuật toán, trong khi độ chính xác của model theo biến yêu cầu tính toán bổ sung. Trong những bài toán yêu cầu độ phức tạp cao thì việc tính toán thêm có thể không phù hợp. Ngoài ra, decrease Gini làm sáng tỏ các biến mà random forest đang sử dụng để thực hiện các quy tắc split của nó.

Việc so sánh sự khác biệt giữa decrease Gini và decrease accuracy của model có thể gợi ý các cách để cải thiện mô hình.[7]

1.3.2.3 Proximity plots

Proximity plots là một công cụ trực quan giúp hiểu rõ hơn về mức độ tương đồng giữa các mẫu dữ liệu trong Random Forest. Các proximity plots được tạo ra bằng cách tính toán ma trận proximity (tương đồng) giữa các mẫu dữ liệu trong Random Forest, sau đó thực hiện phân tích trực quan trên ma trận này.

Ma trận proximity trong Random Forest được tính bằng cách đếm số lần hai mẫu dữ liệu được đặt vào cùng một nhánh trong decision tree.

Sau khi tính toán ma trận proximity, các proximity plots có thể được tạo ra để hiển thị các mẫu dữ liệu trong không gian hai chiều. Các mẫu dữ liệu có tính tương đồng cao sẽ được đặt gần nhau hơn trong không gian này, trong khi các mẫu dữ liệu khác nhau sẽ được đặt xa nhau hơn.

Proximity plots giúp cho việc khám phá các mối tương đồng giữa các mẫu dữ liệu trong Random Forest, và có thể được sử dụng để giải thích kết quả của mô hình.

1.3.3 Giải quyết Overfitting

Random Forest là một mô hình dự đoán mạnh mẽ, có khả năng giảm thiểu hiện tượng overfitting, đặc biệt là khi so sánh với các mô hình dự đoán phức tạp khác. Có một số cách mà Random Forest giải quyết overfitting như sau:

1. Tập con ngẫu nhiên: Trong quá trình huấn luyện Random Forest, một tập con ngẫu nhiên các mẫu được chọn từ tập dữ liệu để xây dựng từng decision tree. Điều này giúp giảm khả năng mô hình bị overfitting do việc quá tập trung vào các mẫu hiếm hơn, kể cả khi dữ liệu đầu vào có chứa nhiễu.

2. Random Feature Selection: Trong mỗi decision tree, chỉ một tập con ngẫu nhiên các features được sử dụng để xây dựng cây. Điều này giúp giảm khả năng mô hình bị overfitting do việc quá tập trung vào các tính năng quan trọng hơn.
3. Features importance: Random Forest cung cấp một đánh giá về mức độ quan trọng của các feature đầu vào. Những feature không quan trọng có thể được loại bỏ để giảm thiểu độ phức tạp của mô hình và giảm khả năng bị overfitting.
4. Tính toán trung bình của nhiều decision tree: Random Forest tính toán trung bình của các decision tree để dự đoán kết quả cuối cùng. Điều này giúp giảm thiểu hiện tượng overfitting bằng cách loại bỏ sự khác biệt lớn giữa các dự đoán từ mỗi cây riêng lẻ.

Tóm lại, Random Forest là một mô hình dự đoán mạnh mẽ, có khả năng giảm thiểu hiện tượng overfitting bằng cách sử dụng các kỹ thuật như tập con ngẫu nhiên, random feature selection, feature importance và tính toán trung bình của nhiều decision tree.

1.4 Mô tả dữ liệu phù hợp

1.4.1 Tính chất dữ liệu đầu vào

Tính chất của dữ liệu đầu vào cho thuật toán Random forest bao gồm:

• Số lượng đặc trưng: Thuật toán Random forest có khả năng xử lý các tập dữ liệu có số lượng đặc trưng lớn. Tuy nhiên, khi số lượng đặc trưng quá lớn, nó có thể gây ra hiện tượng overfitting.

• Đặc trưng dạng số hoặc hạng mục: Thuật toán Random forest có khả năng xử lý các tập dữ liệu có đặc trưng dạng số hoặc hạng mục.

• Dữ liệu bị khuyết: Thuật toán Random forest có khả năng xử lý các tập dữ liệu bị khuyết. Nó có thể sử dụng các phương pháp như mean imputation hoặc median imputation để điền giá trị bị khuyết.

Ngoài ra, Random forest còn có khả năng xử lý dữ liệu phi tuyến tính, các tương tác giữa các đặc trưng và các giá trị nhiễu trong dữ liệu. Tuy nhiên, nếu tập dữ liệu có quá nhiều giá trị nhiễu hoặc bị lệch thì Random forest có thể không cho kết quả tốt.

1.4.2 Hạn chế trong quá trình xử lý dữ liệu

Mặc dù Random forest là một thuật toán rất mạnh và có thể xử lý nhiều loại dữ liệu, tuy nhiên, nó cũng có một số hạn chế trong việc xử lý dữ liệu.

Dữ liệu mà thuật toán Random forest không thể xử lý bao gồm:

1. Dữ liệu không được chẩn chu: Thuật toán Random forest đòi hỏi dữ liệu phải được chuẩn chu và không có các giá trị nhiễu. Nếu dữ liệu bị nhiễu quá nhiều hoặc không được chẩn chu, Random forest có thể cho ra kết quả không chính xác.
2. Dữ liệu có quá nhiều giá trị bị khuyết: Thuật toán Random forest có thể xử lý dữ liệu bị khuyết nhưng nếu tỷ lệ giá trị bị khuyết quá lớn, nó có thể gây ra các vấn đề về chất lượng kết quả.
3. Dữ liệu có tính tuyến tính cao: Random forest là một thuật toán phi tuyến tính, do đó nó không phù hợp với các tập dữ liệu có tính tuyến tính cao.
4. Dữ liệu có độ phức tạp quá lớn: Khi tập dữ liệu quá lớn và phức tạp, Random forest có thể không thể xử lý được trong thời gian hợp lý hoặc gặp phải các vấn đề về trang bị phần cứng.
5. Dữ liệu có tính không cân bằng: Nếu dữ liệu không cân bằng về số lượng mẫu giữa các nhóm, Random forest có thể không cho kết quả tốt.
6. Dữ liệu có nhiều biến độc lập, bao gồm cả các biến liên tục và rời rạc: Random Forest là một phương pháp học máy dựa trên việc kết hợp các decision tree (Decision Trees). Vì vậy, nó hoạt động tốt với dữ liệu có nhiều biến độc lập (independent variables) bởi vì nó cho phép decision tree xác định mối quan hệ giữa các biến độc lập và mục tiêu (target variable).
7. Dữ liệu có sự tương quan và tương tác giữa các biến đầu vào: Random Forest có khả năng xác định các tương tác giữa các biến đầu vào, do đó nó là một giải pháp tốt cho dữ liệu có các biến đầu vào tương tác với nhau. Ví dụ, nếu bạn đang phân loại khách hàng của một công ty, dữ liệu có thể bao gồm các biến như tuổi, thu nhập, giới tính và mức độ hài lòng với sản phẩm của công ty. Các biến này có thể tương tác với nhau, ví dụ như một khách hàng trẻ tuổi có thể có thu nhập thấp hơn, nhưng hài lòng hơn với sản phẩm của công ty.
8. Dữ liệu có các giá trị bị khuyết (missing values): Random Forest có khả năng xử lý các giá trị bị khuyết mà không cần phải loại bỏ các quan sát liên quan đến các giá trị đó. Nó có thể đưa ra dự đoán bằng cách sử dụng các giá trị còn lại và các decision tree tạo ra từ các biến khác.
9. Dữ liệu có nhiễu (noise) hoặc outliers: Random Forest là một phương pháp khá linh hoạt và không nhạy cảm với nhiễu và outliers. Vì vậy, nó là một lựa chọn tốt cho các bộ dữ liệu có nhiễu hoặc outliers.

1.4.3 Hạn chế về dữ liệu

Một số hạn chế về dữ liệu khi sử dụng thuật toán Random Forest:

1. Dữ liệu có số lượng quan sát quá nhỏ: Random Forest thường đòi hỏi một số lượng lớn các quan sát để đưa ra dự đoán chính xác. Nếu bộ dữ liệu quá nhỏ, nó có thể dẫn đến hiện tượng overfitting hoặc underfitting.
2. Dữ liệu có phân phối không đồng đều giữa các lớp: Nếu dữ liệu có sự mất cân bằng giữa các lớp, nghĩa là số lượng quan sát cho mỗi lớp không đồng đều, Random Forest có thể không hoạt động tốt. Điều này có thể dẫn đến việc mô hình bị phân loại sai vì một số lớp có quan sát ít hơn.
3. Dữ liệu có tính chu kỳ: Random Forest không thể dự đoán các giá trị trong tương lai nếu dữ liệu có tính chu kỳ. Ví dụ, nếu bạn đang cố gắng dự đoán giá cổ phiếu theo thời gian, Random Forest có thể không phù hợp.
4. Dữ liệu có sự tương quan cao giữa các biến đầu vào: Nếu dữ liệu có sự tương quan cao giữa các biến đầu vào, Random Forest có thể không tạo ra các decision tree khác nhau đủ để tạo ra các dự đoán khác nhau. Trong trường hợp này, việc sử dụng một mô hình hồi quy tuyến tính hoặc mô hình Neural Network có thể tốt hơn.
5. Dữ liệu có đặc tính thời gian: Nếu dữ liệu có đặc tính thời gian như trường hợp của chuỗi thời gian (time series), Random Forest không phù hợp. Thay vào đó, bạn có thể sử dụng mô hình hồi quy ARIMA hoặc các mô hình khác để dự đoán chuỗi thời gian.

Tóm lại, Random Forest có thể không phù hợp với một số dạng dữ liệu nhất định, ví dụ như dữ liệu có số lượng quan sát quá nhỏ, dữ liệu có tính chu kỳ, dữ liệu có sự tương quan cao giữa các biến đầu vào, và dữ liệu có đặc tính thời gian

1.5 Các biến số liên quan

1.5.1 Các biến số quan trọng trong xây dựng mô hình Random Forest

Các biến số quan trọng trong việc xây dựng Random Forest bao gồm:

Số lượng cây trong Random Forest (number of trees): Đây là số lượng decision tree được xây dựng trong mô hình Random Forest. Việc tăng số lượng cây sẽ làm tăng độ chính xác của mô hình, nhưng đồng thời cũng làm tăng thời gian huấn luyện và độ phức tạp của mô hình.

Số lượng biến đầu vào (features) được chọn để xây dựng mỗi cây: Đây là số lượng biến được chọn ngẫu nhiên từ tập dữ liệu gốc để sử dụng trong quá trình xây dựng mỗi

decision tree. Số lượng biến đầu vào thường được đặt bằng căn bậc hai (sqrt) hoặc logarit tự nhiên (ln) của tổng số lượng biến đầu vào để đảm bảo tính đa dạng của các cây trong Random Forest.[8]

Số lượng mẫu được chọn ngẫu nhiên từ tập dữ liệu gốc để xây dựng mỗi cây (subsample): Đây là số lượng mẫu được chọn ngẫu nhiên từ tập dữ liệu gốc để sử dụng trong quá trình xây dựng mỗi cây. Việc chọn một lượng mẫu nhỏ hơn từ tập dữ liệu gốc có thể giúp giảm overfitting và cải thiện tính tổng quát của mô hình.

Phương pháp chọn biến đầu vào cho mỗi cây: Trong thuật toán Random Forest, có thể chọn một trong hai phương pháp (input variables) được gọi là "Feature selection methods" và "Feature importance methods" [9] để chọn biến đầu vào.

- Phương pháp chọn đặc trưng (Feature selection methods): Phương pháp này dựa trên việc lựa chọn tập con các biến đầu vào (input variables) để sử dụng trong quá trình huấn luyện (training) của mô hình Random Forest. Có nhiều cách để lựa chọn tập con này, ví dụ như "Sequential Forward Selection" (chọn lần lượt các biến quan trọng nhất), "Sequential Backward Selection" (loại lần lượt các biến ít quan trọng nhất) hoặc "Recursive Feature Elimination" (loại lần lượt các biến ít quan trọng và tính toán lại các biến quan trọng của tập con còn lại).
- Phương pháp đánh giá đặc trưng (Feature importance methods): Phương pháp này đánh giá mức độ quan trọng của từng biến đầu vào trong quá trình xây dựng mô hình. Các biến quan trọng sẽ có ảnh hưởng lớn đến kết quả dự đoán của mô hình, trong khi các biến ít quan trọng thì không. Phương pháp đánh giá này có thể dùng các độ đo như Gini importance, Permutation importance hoặc Mean decrease accuracy để tính toán độ quan trọng của các biến.

Tuy nhiên, trong thực tế, thường không cần lựa chọn tập con biến đầu vào hoặc đánh giá độ quan trọng của từng biến một cách rõ ràng, mà ta có thể dùng tất cả các biến có sẵn và để mô hình Random Forest tự động chọn biến quan trọng. Khi đó, các biến ít quan trọng sẽ được loại bỏ trong quá trình xây dựng mô hình tự động, và chỉ các biến quan trọng mới được sử dụng để dự đoán kết quả.

Các tham số để điều chỉnh quá trình xây dựng cây: Các tham số này bao gồm số lượng nhánh của mỗi cây, số lượng lá của mỗi cây, độ sâu tối đa của mỗi cây, giá trị ngưỡng để chia nhánh, v.v. Các tham số này cần được điều chỉnh để đạt được hiệu suất tốt nhất cho mô hình Random Forest.[10]

Trong Random Forest, các biến số này sẽ được sử dụng để xây dựng các decision tree độc lập với nhau, và sau đó kết hợp kết quả từ các cây để đưa ra dự đoán cuối cùng.

1.5.2 Các siêu tham số quan trọng

Các siêu tham số trong random forest được sử dụng để tăng khả năng dự đoán của mô hình hoặc để làm cho mô hình nhanh hơn. Hãy xem các siêu tham số của chức năng random forest tích hợp sẵn của sklearns.

1.5.2.1 Tăng sức mạnh dự đoán

n_estimators: để chỉ định số cây trong rừng trước khi lấy phiếu bầu hoặc giá trị trung bình của các dự đoán. Số lượng cây càng cao thì mô hình càng mạnh và dự đoán ổn định hơn, nhưng cũng làm chậm quá trình tính toán.

max_features: chỉ định số lượng biến đầu vào tối đa được sử dụng để xây dựng các cây quyết định trong mô hình Random Forest.

min_sample_leaf: giúp xác định số lượng lá tối thiểu cần thiết để tách một nút bên trong.

1.5.2.2 Tăng tốc độ của mô hình

n_jobs: để chỉ định số lượng bộ xử lý được phép sử dụng. Nếu giá trị là 1, chỉ có thể sử dụng một bộ xử lý, trong khi giá trị "-1" có nghĩa là không có giới hạn.

random_state: Siêu tham số random_state giúp đảm bảo rằng mô hình sẽ tạo ra cùng một kết quả khi nó được huấn luyện trên cùng một dữ liệu và với cùng một siêu tham số.

Cuối cùng, có *oob_score* (còn được gọi là lấy mẫu oob) là một phương pháp xác thực chéo random forest, trong đó khoảng một phần ba dữ liệu không được sử dụng để đào tạo mô hình và có thể được sử dụng để đánh giá hiệu suất của nó.

1.6 Kiểm định Random forest

Để kiểm định cho các mô hình học máy với các chỉ số phù hợp, chúng ta cần quan tâm đầu ra của mô hình là các giá trị phân loại rời rạc, các giá trị liên tục hay xếp hạng [11]. Random Forest có thể giải quyết cả bài toán Regression và Classification nên đối với mỗi loại bài toán khác nhau sẽ có phương pháp kiểm định khác nhau.

Đối với bài toán classification, mô hình có thể được kiểm định bằng các chỉ số dựa vào ma trận nhầm lẫn như Precision, Recall, F-measure, hoặc kiểm định giữa các mô hình với

nhau qua đường cong ROC (Receiver Operating Characteristic), AUC (Area Under ROC Curve) [11].

Không giống các bài toán classification, thật khó để yêu cầu bài toán regression có thể đưa ra chính xác giá trị so với kết quả thực tế, thay vào đó chúng ta quan tâm rằng sự dự đoán của bài toán regression gần với kết quả thực tế như thế nào. Đối với bài toán regression, các chỉ số như R-squared, MSE (Mean Square Error)/RMSE (Root Mean Square Deviation), MAE (Mean Absolute Error) [12] [13] có thể được áp dụng để thực hiện kiểm định.

CHƯƠNG 2: BÀI TOÁN ỨNG DỤNG

2.1 Mô tả bài toán

Đối với doanh nghiệp, việc giữ chân một khách hàng cũ sẽ ít tốn kém chi phí hơn nhiều so với tìm kiếm một khách hàng mới. Để tối ưu hóa chi phí và lợi luận cho doanh nghiệp, bài toán dự đoán khách hàng rời bỏ/ bỏ dùng dịch vụ là một bài toán cần thiết nhằm phục vụ ra quyết định kịp thời giữ chân các khách hàng có nguy cơ cao sẽ hủy bỏ sử dụng dịch vụ của doanh nghiệp. Nhờ sự giúp đỡ của mô hình học máy Random Forest, nhóm thực nghiệm dự đoán khách hàng ngưng sử dụng dịch vụ trong lĩnh vực viễn thông và sẽ được mô tả chi tiết ở các phần sau của báo cáo này.

2.2 Mô tả dữ liệu

Dữ liệu phục vụ cho huấn luyện mô hình học máy Random Forest trong bài toán dự đoán khách hàng ngưng sử dụng dịch vụ là bộ dữ liệu từ trang Kaggle.

Link dataset: Telco Customer Churn | Kaggle

Bộ dữ liệu này có tổng cộng 7043 record đại diện cho 7043 khách hàng của công ty viễn thông với 21 cột dữ liệu mô tả 4 nhóm dữ kiện chính:

- Khách hàng đã dừng sử dụng dịch vụ viễn thông trong tháng vừa rồi hay không
- Dịch vụ mà người dùng đã đăng ký sử dụng
- Thông tin tài khoản khách hàng
- Thông tin nhân khẩu học của khách hàng

Cột	Ý nghĩa
customerID	ID của khách hàng
gender	giới tính của khách hàng
SeniorCitizen	tuổi của người dùng có từ 65 trở lên không
Dependents	khách hàng có đang bảo hộ cho ai không (cha mẹ, con cái…)
tenure	số tháng khách hàng đã sử dụng dịch vụ của công ty
PhoneService	khách hàng có dùng dịch vụ điện thoại hay không
MultipleLines	khách hàng có đăng ký dùng nhiều tuyến điện thoại hay không
InternetService	dịch vụ internet mà khách hàng dùng
OnlineSecurity	khách hàng có đăng ký dùng dịch vụ bảo mật trực tuyến không

OnlineBackup	khách hàng có đăng ký dùng dịch vụ backup trực tuyến không
DeviceProtection	khách hàng có đăng ký gói bảo vệ thiết bị đi kèm không
TechSupport	khách hàng có đăng ký dịch vụ hỗ trợ kỹ thuật hay không
StreamingTV	khách hàng có sử dụng Internet để truyền phát chương trình truyền hình hay không
StreamingMovies	khách hàng có sử dụng Internet để truyền phát phim hay không
Contract	loại hợp đồng của khách hàng
PaperlessBilling	khách hàng chọn nhận hóa đơn phi giấy tờ hay không
PaymentMethod	phương thức thanh toán của khách hàng
MonthlyCharges	mức thanh toán hàng tháng của khách hàng
TotalCharges	tổng thanh toán từ trước đến giờ của khách hàng
Churn	khách hàng đã hủy dùng dịch vụ hay chưa

Bảng 2.1: Mô tả dữ liệu bài toán

2.3 Các bước thực hiện

Tải dữ liệu và chuyển đổi kiểu dữ liệu phù hợp

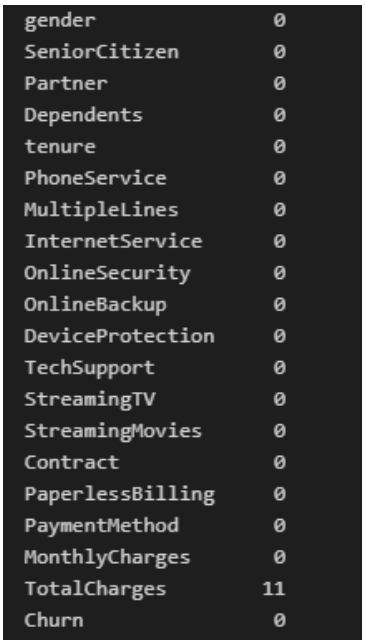
Tải dữ liệu lên dataframe và chuyển kiểu dữ liệu của các cột dữ liệu định lượng về dạng số như cột tenure, MonthlyCharges, TotalCharges, chuyển kiểu dữ liệu của các cột dữ liệu định tính về dạng object.

```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7043 non-null   object
 1   SeniorCitizen     7043 non-null   object
 2   Partner           7043 non-null   object
 3   Dependents        7043 non-null   object
 4   tenure            7043 non-null   int64
 5   PhoneService      7043 non-null   object
 6   MultipleLines     7043 non-null   object
 7   InternetService   7043 non-null   object
 8   OnlineSecurity    7043 non-null   object
 9   OnlineBackup      7043 non-null   object
 10  DeviceProtection  7043 non-null   object
 11  TechSupport       7043 non-null   object
 12  StreamingTV       7043 non-null   object
 13  StreamingMovies   7043 non-null   object
 14  Contract          7043 non-null   object
 15  PaperlessBilling  7043 non-null   object
 16  PaymentMethod     7043 non-null   object
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7043 non-null   float64
 19  Churn             7043 non-null   object
```

Hình 2.1: Kiểu dữ liệu sau khi chuyển đổi

Xử lý giá trị không phù hợp sau khi chuyển đổi kiểu dữ liệu

Khi chuyển đổi kiểu dữ liệu, nhận thấy có 11 record có giá trị NaN ở cột TotalCharges



gender	0
SeniorCitizen	0
Partner	0
Dependents	0
tenure	0
PhoneService	0
MultipleLines	0
InternetService	0
OnlineSecurity	0
OnlineBackup	0
DeviceProtection	0
TechSupport	0
StreamingTV	0
StreamingMovies	0
Contract	0
PaperlessBilling	0
PaymentMethod	0
MonthlyCharges	0
TotalCharges	11
Churn	0

Hình 2.2: Số lượng record chứa giá trị không hợp lệ ở các cột

Sau khi kiểm tra, các dòng dữ liệu chứa TotalCharges là NaN đồng thời có tenure = 0 (đồng nghĩa với việc người dùng chưa sử dụng dịch vụ tháng nào) nên ta có thể xử lý bằng cách lấp đầy các giá trị NaN này bằng giá trị 0

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
488	Female	0	Yes	Yes	0	No	No phone service
753	Male	0	No	Yes	0	Yes	No
936	Female	0	Yes	Yes	0	Yes	No
1082	Male	0	Yes	Yes	0	Yes	Yes
1340	Female	0	Yes	Yes	0	No	No phone service
3331	Male	0	Yes	Yes	0	Yes	No
3826	Male	0	Yes	Yes	0	Yes	Yes
4380	Female	0	Yes	Yes	0	Yes	No
5218	Male	0	Yes	Yes	0	Yes	No
6670	Female	0	Yes	Yes	0	Yes	Yes
6754	Male	0	No	Yes	0	Yes	Yes

Hình 2.3: Các record có giá trị không hợp lệ ở cột TotalCharges

Chuẩn bị dữ liệu phù hợp với mô hình huấn luyện

Thực hiện chuyển biến định tính thành giá trị số và chuẩn bị dữ liệu thành 2 nhóm: nhóm dữ liệu có scale và nhóm dữ liệu không scale trên các cột dữ liệu định lượng.

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
0	0	0	1	0	1	0	1	0	0	2	0
1	1	0	0	0	34	1	0	0	2	0	2
2	1	0	0	0	2	1	0	0	2	2	0
3	1	0	0	0	45	0	1	0	2	0	2
4	0	0	0	0	2	1	0	1	0	0	0

Hình 2.4: Bộ dữ liệu chưa scale

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtecti
0	0	0	1	0	-1.277445	0	1	0	0	2	
1	1	0	0	0	0.066327	1	0	0	2	0	
2	1	0	0	0	-1.236724	1	0	0	2	2	
3	1	0	0	0	0.514251	0	1	0	2	0	
4	0	0	0	0	-1.236724	1	0	1	0	0	

Hình 2.5: Bộ dữ liệu đã được sacle

Huấn luyện mô hình và đánh giá

2.4 Kết quả và đánh giá

Sau khi huấn luyện mô hình, kết quả thu được về độ chính xác và ma trận nhầm lẫn của mô hình trên 2 tập dữ liệu được thể hiện như sau

- Tập dữ liệu không có scale

  Độ chính xác: 0.8041163946061036

	precision	recall	f1-score	support
0	0.84	0.91	0.87	1036
1	0.67	0.52	0.58	373
accuracy			0.80	1409
macro avg	0.75	0.71	0.73	1409
weighted avg	0.79	0.80	0.80	1409

Hình 2.6: Ma trận nhầm lẫn của Random Forest trên bộ dữ liệu không scale

- Tập dữ liệu có scale

  Độ chính xác: 0.8034066713981547

	precision	recall	f1-score	support
0	0.84	0.91	0.87	1036
1	0.67	0.52	0.58	373
accuracy			0.80	1409
macro avg	0.75	0.71	0.73	1409
weighted avg	0.79	0.80	0.79	1409

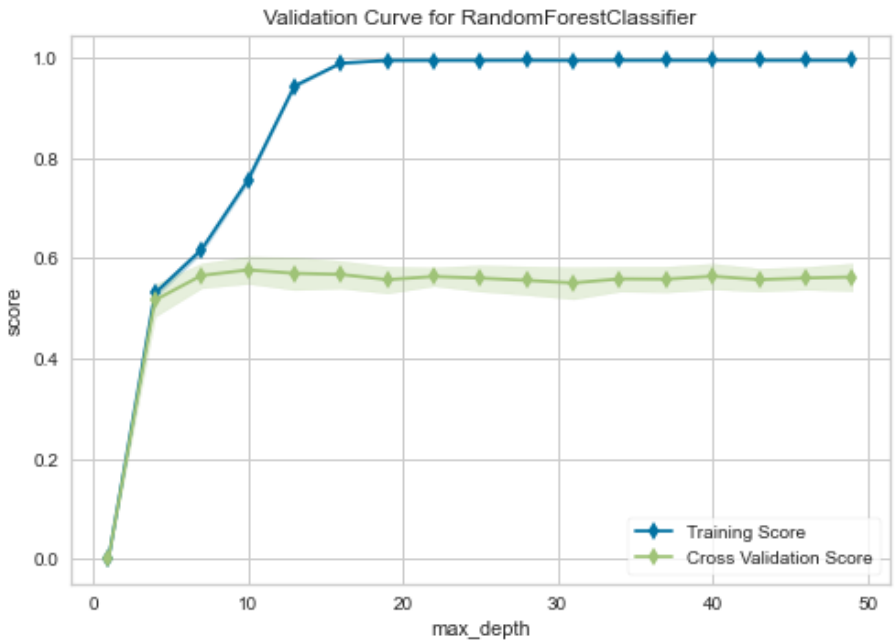
Hình 2.7: Ma trận nhầm lẫn của Random forest trên bộ dữ liệu có scale

Đánh giá mức độ quan trọng của các biến trong mô hình, các biến định lượng có mức độ quan trọng lớn nhất.

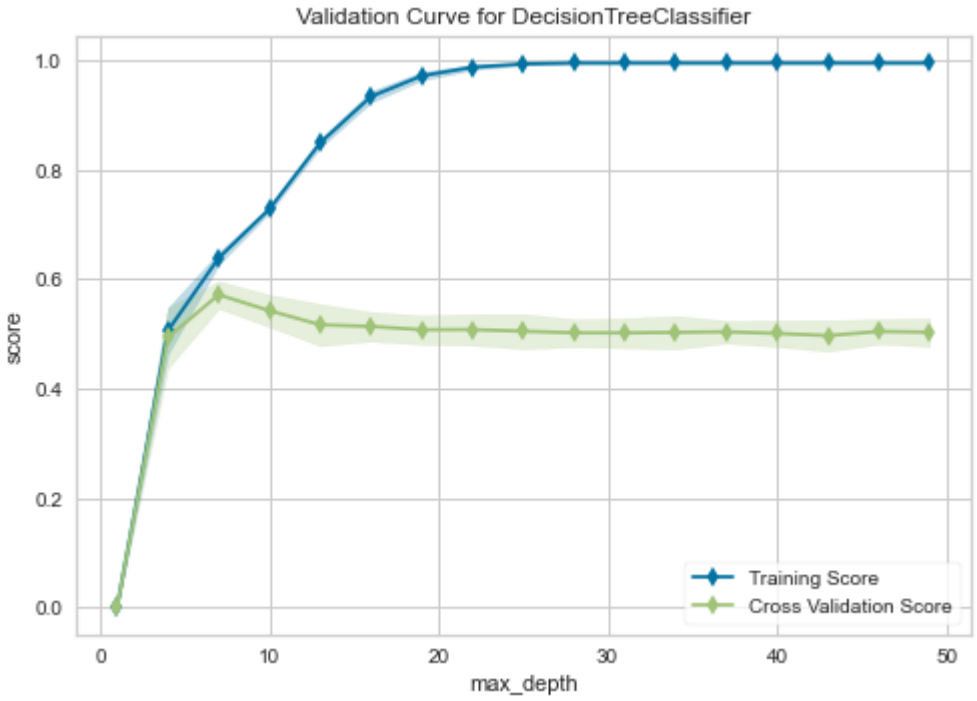
	feature	importance
18	TotalCharges	0.157582
4	tenure	0.155399
17	MonthlyCharges	0.143339
14	Contract	0.135200
8	OnlineSecurity	0.073763
11	TechSupport	0.061691
16	PaymentMethod	0.042494
7	InternetService	0.040885
9	OnlineBackup	0.027658
15	PaperlessBilling	0.023622
6	MultipleLines	0.020458
10	DeviceProtection	0.019308
0	gender	0.016970
13	StreamingMovies	0.016134
3	Dependents	0.015767
2	Partner	0.014948
1	SeniorCitizen	0.014796
12	StreamingTV	0.013869
5	PhoneService	0.006117

Hình 2.8: Độ quan trọng của các biến trong mô hình Random forest

Khi so sánh giữa Random Forest và Decision Tree cho cùng bài toán và tập dữ liệu, đánh giá trên sự thay đổi của max_depth của mô hình, ta có thể thấy mức độ tối ưu hơn của Random Forest qua 2 biểu đồ dưới đây.



Hình 2.9: Validcation curve của Random Forest



Hình 2.10: Validation curve của Decision tree

Ở mô hình Random Forest, khi càng tăng max_depth, điểm F1 có tăng tới ngưỡng, sau đó điểm training tăng nhưng không làm tăng điểm cross validation. Ở mô hình Decision Tree, khi càng tăng max_depth, điểm F1 có tăng tới ngưỡng nhưng sau đó điểm training tăng, điểm cross validation giảm.

CHƯƠNG 3: TỔNG KẾT

3.1 Ưu, nhược điểm của Random forest

3.1.1 Ưu điểm

Ưu điểm của Random Forest:

- Hạn chế vấn đề overfitting so với giải thuật gốc Decision Tree.
- Xử lý được các dataset lớn, high-dimensional data một cách hiệu quả.
- Đa chức năng khi có thể vừa sử dụng cho bài toán regression và classification với độ chính xác cao.
- Dễ dàng đánh giá được các biến nào là quan trọng đối với mô hình.
- Xử lý được dataset có missing-value

3.1.2 Nhược điểm [14]

- Vì cần phải xây dựng và dựa trên nhiều decision tree nên thời gian là yếu tố đánh đổi, không phù hợp với các dự đoán thời gian thực.
- Kém hiệu quả với các dataset nhỏ hoặc các low-dimensional data

3.2 Kết luận

Random Forest là một trong những mô hình dự đoán được sử dụng phổ biến trong Machine Learning và có khả năng giải quyết nhiều bài toán khác nhau. Với việc sử dụng nhiều decidion tree. Random Forest cung cấp một dự đoán chính xác và đáng tin cậy hơn so với một decion tree đơn lẻ.

Một trong những lợi ích lớn nhất của Random Forest là khả năng giảm thiểu hiện tượng overfitting, một vấn đề thường gặp trong các mô hình dự đoán phức tạp. Các kỹ thuật như tập con ngẫu nhiên, random feature selection, feature importance và tính toán trung bình của nhiều decision giúp giảm thiểu độ phức tạp của mô hình và giảm khả năng bị overfitting.

Các ứng dụng của Random Forest rất đa dạng, bao gồm phân loại, dự đoán và phân tích hồi quy. Với số lượng cây decision tree lớn, Random Forest có thể giải quyết các bài toán với dữ liệu lớn và phức tạp, và cho phép các chuyên gia và nhà khoa học dữ liệu tìm ra những đặc điểm và quy luật quan trọng của dữ liệu đó.

Tóm lại, Random Forest là một mô hình dự đoán mạnh mẽ và phổ biến trong machine learning, có khả năng giải quyết hiện tượng overfitting, và được sử dụng rộng rãi trong nhiều lĩnh vực ứng dụng khác nhau. Nó là một công cụ mạnh mẽ giúp các nhà khoa học dữ liệu và chuyên gia.

TÀI LIỆU THAM KHẢO

1. *What is Bagging? | IBM*. [cited 2023 01/03/2023]; Available from: https://www.ibm.com/topics/bagging.
2. Shukla, P. *Out of Bag (OOB) Score for Bagging in Data Science*. 2022 [cited 2023 1/3/2023]; Available from: https://www.analyticsvidhya.com/blog/2022/11/out-of-bag-oob-score-for-bagging-in-data-science/.
3. Hoang, P.M. *Ensemble learning và các biến thể (P1)*. 2020 [cited 2023 1/3/2023]; Available from: https://viblo.asia/p/ensemble-learning-va-cac-bien-the-p1-WAyK80AkKxX.
4. Hải, P.M. and N.N. Quang, *Khái niệm về phương pháp random forest trong cuộc cách mạng machine learning và định hướng ứng dụng trong lĩnh vực viễn thám.* Tạp chí Khoa học Đo đạc và Bản đồ, 2019(39): p. 15-19.
5. University, M. *Random Forests*. [cited 2023 1/3/2023]; Available from: https://www.math.mcgill.ca/yyang/resources/doc/randomforest.pdf.
6. Hoare, J. *How is Variable Importance Calculated for a Random Forest?* [cited 2023 1/3/2023]; Available from: https://www.displayr.com/how-is-variable-importance-calculated-for-a-random-forest/.
7. Bruce, P., A. Bruce, and P. Gedeck, *Practical statistics for data scientists: 50+ essential concepts using R and Python*. 2020: O'Reilly Media.
8. James, G., et al., *An introduction to statistical learning*. Vol. 112. 2013: Springer.
9. Cutler, D.R., et al., *Random forests for classification in ecology*. Ecology, 2007. **88**(11): p. 2783-2792.
10. Hastie, T., et al., *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. 2009: Springer.
11. Tao, C. *How to Evaluate a Classification Machine Learning Model*. 2020; Available from: https://towardsdatascience.com/how-to-evaluate-a-classification-machine-learning-model-d81901d491b1.
12. Wu, S. *3 Best metrics to evaluate Regression Model?* 2020; Available from: https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b.

13. Ahmed, M. *Ways to Evaluate your Regression Model*. 2021; Available from: https://linguisticmaz.medium.com/evaluating-regression-models-cb02ba075e16.
14. *What Are The Disadvantages Of Random Forest?* 2022 [cited 2023 1/3/2023]; Available from: https://www.rebellionresearch.com/what-are-the-disadvantages-of-random-forest.