TRƯỜNG ĐẠI HỌC PHENIKAA

KHOA CÔNG NGHỆ THÔNG TIN

HỌC PHẦN: ĐỒ ÁN CƠ SỞ

ĐỀ TÀI: DỰ BÁO CHUỖI THỜI GIAN VỚI MACHINE LEARNING

Giảng viên hướng dẫn: Nguyễn Văn Thiệu

Khóa	Sinh viên	Mã số sinh viên
K14	Nguyễn Thị Ngọc	20010788
K14	Nguyễn Văn Cường	20010758

LỜI MỞ ĐẦU

Time Series Analysis & Forecasting là 1 lớp mô hinh quan trọng trong thống kê, kinh tế lượng và machine learning. Sở dĩ chúng được gọi lớp mô hình này là chuỗi thời gian (time series) là vì mô hình được áp dụng trên các chuỗi đặc thù có yếu tố thời gian. Một mô hình chuỗi thời gian thường dự báo dựa trên giả định rằng các qui luật trong quá khứ sẽ lặp lại ở tương lai. Do đó xây dựng mô hình chuỗi thời gian là chúng ta đang mô hình hóa mối quan hệ trong quá khứ giữa biến độc lập (biến đầu vào) và biến phụ thuộc (biến mục tiêu). Dựa vào mối quan hệ này để dự đoán giá trị trong tương lai của biến phụ thuộc.

Do là dữ liệu chịu ảnh hưởng bởi tính chất thời gian nên chuỗi thời gian thường xuất hiện những qui luật đặc trưng như: yếu tố chu kỳ, mùa vụ và yếu tố xu hướng. Đây là những đặc trưng thường thấy và xuất hiện ở hầu hết các chuỗi thời gian.

- Yếu tố chu kì, mùa vụ là những đặc tính lặp lại theo chu kì
- Yếu tố xu hướng (trend) thể hiện đà tăng hoặc giảm của chuỗi trong tương lai

MỤC LỤC

I. Time series

1. Time series data là gì?

Chuỗi thời gian là một chuỗi thông tin gắn một khoảng thời gian cho mỗi giá trị. Giá trị có thể là bất cứ thứ gì có thể đo lường được, phụ thuộc vào thời gian theo một cách nào đó, chẳng hạn như giá cả, độ ẩm hoặc số lượng người. Miễn là các giá trị chúng tôi ghi lại rõ ràng, bất kỳ phương tiện nào cũng có thể được đo bằng chuỗi thời gian.

_ Một số tính năng nổi bật của chuỗi thời gian là gì?

+ Khoảng thời gian

Đối với người mới bắt đầu, không có bất kỳ hạn chế nào về tổng khoảng thời gian của một chuỗi thời gian. Nó có thể là một phút, một ngày, một tháng hay thậm chí là một thế kỷ. Tất cả những gì cần thiết là điểm bắt đầu và điểm kết thúc. Tất nhiên, thường có nhiều điểm ở giữa và khoảng thời gian ngăn cách hai điểm liên tiếp được gọi là "khoảng thời gian". Ví dụ: nếu dữ liệu được ghi lại một lần mỗi ngày từ ngày 1/1/2000 đến đêm giao thừa năm 2009, thì một khoảng thời gian duy nhất sẽ là một ngày, trong khi toàn bộ khoảng thời gian sẽ là một thập kỷ.

+ Tính thường xuyên

"Tần suất" của tập dữ liệu cho chúng ta biết tần suất các giá trị của tập dữ liệu được ghi lại. Để có thể phân tích chuỗi thời gian một cách có ý nghĩa, tất cả các khoảng thời gian phải **bằng nhau và được xác định rõ ràng** . Đổi lại, điều này dẫn đến tần suất **không đổi**. Tần số này là phép đo thời gian và có thể nằm trong khoảng từ vài phần nghìn giây đến vài thập kỷ. Tuy nhiên,

những thứ chúng ta thường gặp nhất là hàng ngày, hàng tháng, hàng quý và hàng năm.

2. Phân tích Time Series

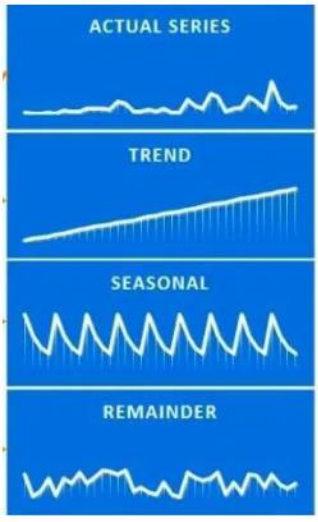
Một dữ liệu chuỗi thời gian thường được phân rã thành 4 thành phần con như sau:

Trend: chỉ ra xu hướng tổng quan của dữ liệu theo thời gian: lên hoặc xuống, tăng hoặc giảm

Seasonality: chỉ ra các xu hướng theo mùa vị, chỉ ra các pattern theo tháng, theo quý

Cycle: thành phần chu kỳ, nó khác seasonality ở chỗ thành phần này có sự vận động trong khoảng thời gian dài hơn (nhiều năm)

Irregular remainder: thành phần nhiễu còn lại sau khi trích xuất hết các thành phần ở trên, nó chỉ ra sự bất thường của các điểm dữ liệu

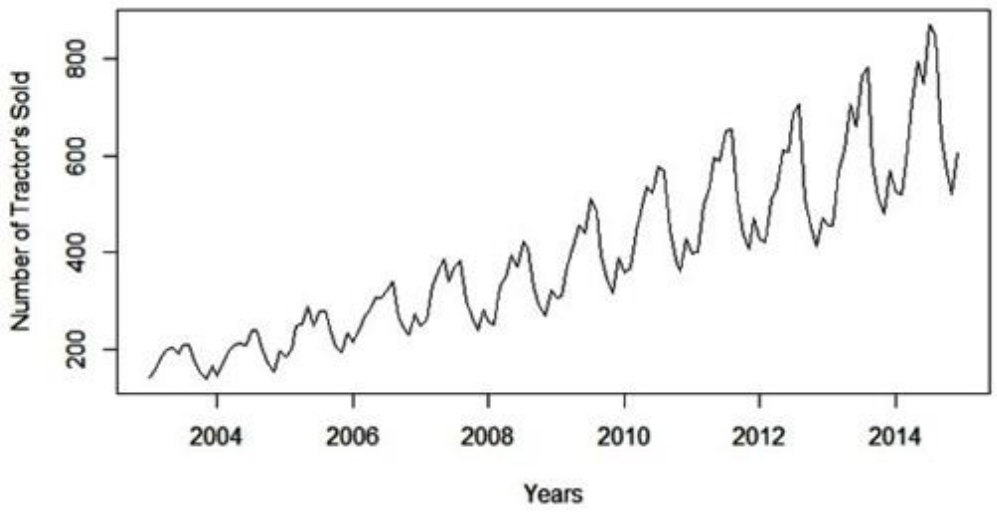


Thay vì đưa ra dự báo trên Actual Series – là một đường nhìn rất gập ghềnh và khó dự đoán, ta thực hiện trên các thành phần con nhìn có tính quy luật với xu

hướng vận động rõ ràng hơn rồi kết hợp các thành phần này lại với nhau, giống như cách ta phân rã ABC*D thành các đường hình sin A,B,C,D như ở trên vậy!

a. Ví dụ về Time Series.

Bộ dữ liệu này lưu trữ số lượng máy kéo bán ra hàng tháng trong khoảng thời gian từ 2003 đến 2015. Chúng ta cùng nhìn vào biểu diễn trên đồ thị của dữ liệu nhé!



Nhắc lại một chút thì Time Series Data có thể được phân rã thành 4 thành phần: Trend, Seasonality, Cycle, Irregular remainder. Cycle chỉ ra xu hướng vận động trong một khoảng thời gian dài (thông thường chu kỳ rơi vào 7 năm hoặc hơn), do dữ liệu của chúng ta không trải đủ rộng nên chúng ta sẽ chỉ còn quan tâm đến 3 thành phần còn lại là Trend, Seasonality và Irregular remainder. Model dự đoán của chúng ta sẽ dựa trên hàm xác định sự phụ thuộc của sale dựa trên các yếu tố nêu trên:

$$Y_t = f(Trend_t\ , Seasonality_t\ , Remainder_t)$$

Chẳng hạn ta có thể xác định sale thông qua tích của các thành phần:

$$Y_t = Trend_t \times Seasonality_t \times Remainder_t$$

- Trend

Để phát hiện ra xu hướng trong dữ liệu, ta dùng một phương pháp phổ biến gọi là trung bình trượt. Giống như việc bạn là quần áo để loại bỏ những nếp nhăn, thì phương pháp này sẽ giúp loại bỏ các đường zigzag lên xuống của đồ thị để xuất ra một xu hướng ổn định trong dữ liệu thông qua việc lấy giá trị trung bình của các giá trị liền kề trong một khoảng thời gian. Công thức của trung bình trượt được xác định như sau:

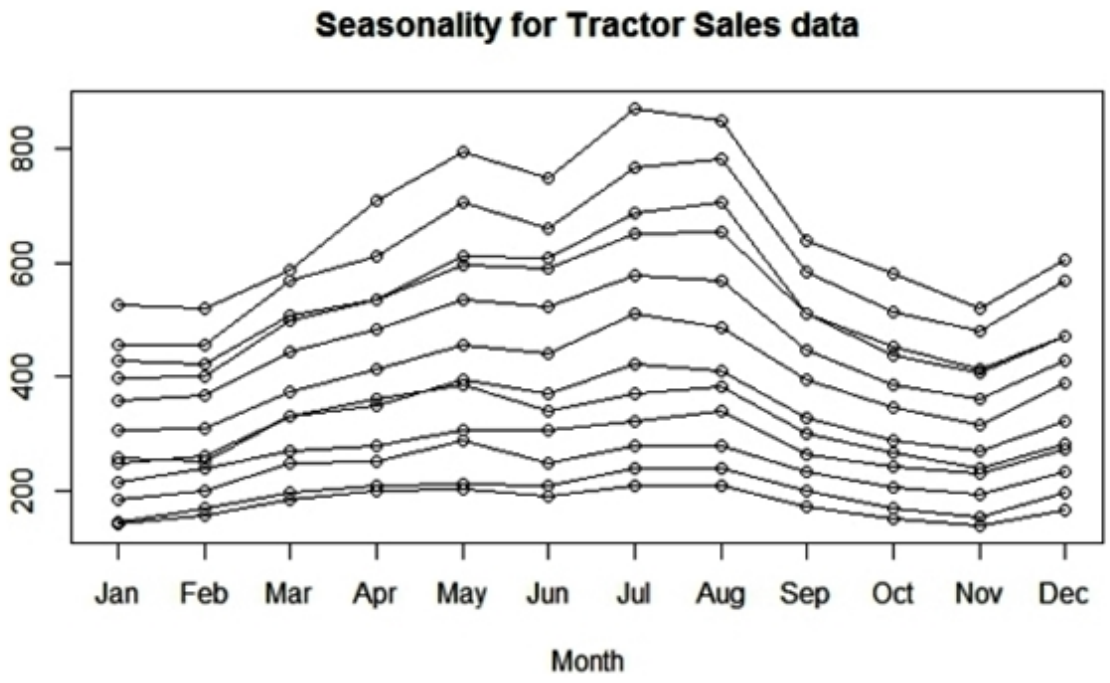
$$Moving\ Average = \frac{\sum_{i=-m}^{m} Y_{t+i}}{2m}$$

Chúng ta thử lấy trung bình trượt với những khoảng thời gian khác nhau trên bộ dữ liệu ban đầu:

Đường màu vàng biểu diễn dữ liệu ban đầu, còn đường màu xanh biểu diễn giá trị trung bình trượt. Như chúng ta thấy với giá trị khung thời gian bằng 12 (m=6) cho ta đồ thị gần như tuyến tính với giá trị tăng dần khi đi dọc theo trục tọa độ, đây chính là dạng Trend ta cần phát hiện. Các giá trị khung thời gian nhỏ hơn xuất ra đồ thị vẫn còn các đường lên xuống đan xen nên không cho thấy xu hướng rõ ràng của dữ liệu.

- Seasonality

Để khảo sát yếu tố mùa vụ trong dữ liệu, trước hết ta thử thể hiện số lượng máy kéo bán ra theo từng tháng trong mỗi năm.



Như ta có thể thấy, sự vận động của giá trị số lượng bán ra theo từng tháng trong một năm là có sự tương đồng giữa các năm với nhau khi đồ thị xếp chồng có cùng dạng. Lượng máy kéo bán ra gần như đạt đỉnh vào tháng 7 hoặc 8, thấp dần về đầu và cuối năm.

Thành phần mùa vụ cho từng tháng được tính bằng giá trị trung bình của tháng đó (tính từ 2003 đến 2015) sau khi đã loại bỏ Trend

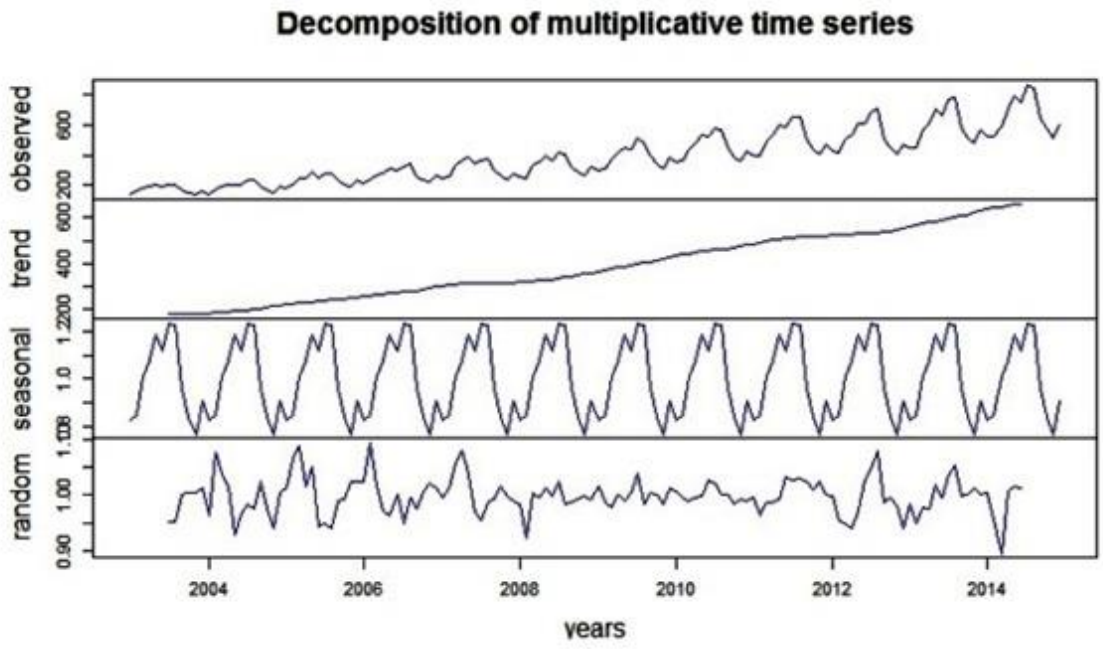
- Irregular remainder

Thành phần này chính là thành phần còn lại sau khi đã loại bỏ Trend và Seasonality:

$$Remainder_t = \frac{Y_t}{Trend_t \times Seasonality_t}$$

Chúng ta mong muốn thành phần Irregular remainder này phải giống như là nhiễu trắng, tức là nó không thể hiện bất kỳ một pattern nào (truely random). Nó thể hiện chúng ta đã xuất hết các information từ dữ liệu chuỗi thời gian ban đầu, giống như việc xay nước mía thì chúng ta mong muốn lấy được hết phần nước, và nhiễu trắng giống như phần bã còn lại – không thể lấy được thêm nước từ phần bã đó.

Cụ thể biểu diễn 3 thành phần này trên đồ thị, ta thu được hình sau:



Đây chính là ý tưởng cơ bản cho lớp các thuật toán Time Series Forecasting. Thực thế thì những thuật toán phổ biến cho dữ liệu chuỗi thời gian như ARIMA, Holt-Winters seasonal method sẽ phức tạp hơn thế này. Việc trích xuất các thành phần con từ dữ liệu gốc không phải là đơn giản giống như việc pha màu và phả tách các màu ban đầu ra riêng.

II. Decision Tree

1. Định nghĩa

Decision Tree là 1 mô hình supervised learing, có thể được áp dụng vào cả hai bài toán Classification và Regression. Việc xây dựng 1 Decision Tree trên dữ liệu huấn luyện cho trước là việc đi xác định các câu hỏi và thứ tự của chúng.

Mô phỏng việc con người suy nghĩ và giải quyết các vấn đề

2. Các loại cây quyết định

Có 2 loại cây quyết định là Classification Tree (cây phân loại) và Regression Tree (cây hồi quy)

2.1. Classification Tree

Gini index tương tự như information gain, dùng để đánh giá xem việc phân chia ở node điều kiện có tốt hay không.

Để tính Gini index, trước hết mình sẽ tính chỉ số Gini, chỉ số Gini tính ở từng node.

Đầu ra là 1 giá trị yes/no

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

Trong đó C là số lớp cần phân loại, $p_i = \frac{n_i}{N}$, n_i là số lượng phần tử ở lớp thứ i. Còn N là tổng số lượng phần tử ở node đó, $N = \sum_{i=1}^{N} n_i \Rightarrow \sum_{i=1}^{N} p_i = 1$.

Do $0 <= p_i <= 1 \forall i$ và $\sum_{i=1}^{N} p_i = 1$ nên:

$$\sum_{i=1}^{C}(p_i)^2 <= (\sum_{i=1}^{C} p_i)^2 = 1 \Rightarrow Gini \geq 0, \text{ dấu bằng xảy ra khi } \exists j : p_j = 1 \text{ và } p_k = 0 \; \forall k \neq j$$

$$\sum_{i=1}^{C}(p_i)^2 >= \frac{(\sum_{i=1}^{C} p_i)^2}{C} = \frac{1}{C} \Rightarrow Gini \leq \frac{C-1}{C}, \text{ dấu bằng xảy ra khi } p_j = \frac{1}{C} \; \forall j$$

Tính tất cả rồi so sánh xem gini nào nhỏ nhất thì lấy giá trị đó làm node root

Sau đó sử dụng bên yes để tính gini tiếp cho đến khi chỉ số gini của node dưới cao hơn chỉ số gini ở node trên thì dừng lại.

chú ý: nếu dữ liệu ở dạg số thì tinh trung bình 2 số liên tiếp rồi xây dựng cây như trên

2.2. Regression Tree

Đầu ra là 1 số

3. Ưu điểm và nhược điểm của Decision Tree

3.1. Ưu điểm

_ Dễ visualine và dễ hiểu:

_ Hữu ích trong việc thăm dò dữ liệu: biết được quan hệ giữa các biến, phân biệt dữ liệu dễ hơn, có thể tạo ra các biến mới hoặc đưa 1 số tính năng vào nhóm.

_ Yêu cầu ít phải là sạch dữ liệu: miễn nhiễm với các dữ liệu ngoại lai và dữ liệu bị thiếu nên cần phải làm sạch dữ liệu ít hơn, xử lý được cả dữ liệu chữ và số.

3.2. Nhược điểm

_ Overfitting: cây quyết định đơn lẻ có xu hướng trang bị quâ mức các dữ liệu cần giải quyết bằng cách thiết lập Ronstraints trên các thông số mô hình (chiều cao của cây). Hoạt động tốt với dữ liệu train nhưng khi có dư liệu mới vào thì không hiệu quả

_ Nhạy cảm với dữ liệu nhiễu: có thể có quá nhiều nhiễu.

_ Không phù hợp và chính xác với dữ liệu liên tục: cây quyết định bị sai lệch với dữ liệu không cân bằng -> cần cân bằng dữ liệu trước khi tạo cây quyết định

_ Đối với Regression: các giá trị dự đoán là giá trị trung bình nên khi dự đoán không dự đoán chính xác được giá trị đầu ra (do mất 1 số thông tin)

Giải pháp: 1 số thuật toán đóng gói hoặc tăng cường

Purning: cắt đi 1 số node tránh overfitting

Bagging, Random Forest, Boosting

III. Random Forest

1. Tổng quan Random Forest

Data Analysis và Machine learning đã trở thành một phần cảu khoa học hiện đại, với nhiều vai trò quan trọng giúp cung cấp các mô hình tự động để dự đoán các bài toán mang tính thực tế dựa trên các thông tin được quan sát trong quá khứ. Việc sử dụng các thuật toán yêu cầu về hiểu biết các cơ chế, đặc tính để hiểu và giải thích được các kết quả của thuật toán. Bài này được viết để cung cấp các kiến thức chuyên sâu về thuật toán Random Forest nhằm sáng tỏ những khả năng của thuật toán đáp ứng cho thực tế. Thuật toán đầu tiên của Random Forest được tạo ra vào năm 1995 bởi Tin Kam Ho bằng sử dụng phương pháp đóng gói các thuộc tính (là một phương pháp học tập nhằm giảm mối tương quan giữa các bộ ước lượng trong 1 tập hợp) theo công thức của Ho, là một cách để thực hiện phương pháp "phân biệt ngẫu nhiê" để phân loại do Eugene Kleinberg đề xuất. Random Forest là thuật toán học có giám sát (supervised learning) đóng vai trò quan trọng trong việc xây dựng và giải quyết các bài toán thực tế. RF khá đa dụng khi có thể áp dụng để giải được 2 bài toán là classification và regression. RF là sự kết hợp của việc tạo ra các dataset con ngẫu nhiên có hoàn lại và xây dựng cây quyết định theo cách chọn đặc trưng ngẫu nhiên. Do đó RF có thể tránh được overfifiting và giảm bias

2. Phương pháp xây xây dựng Random Forest

2.1. Tìm hiểu về RF

Sau khi kết hợp lại cả 2 y tưởng Bagging bootstrap, Split-variable randomization thì gọi là Random Forest

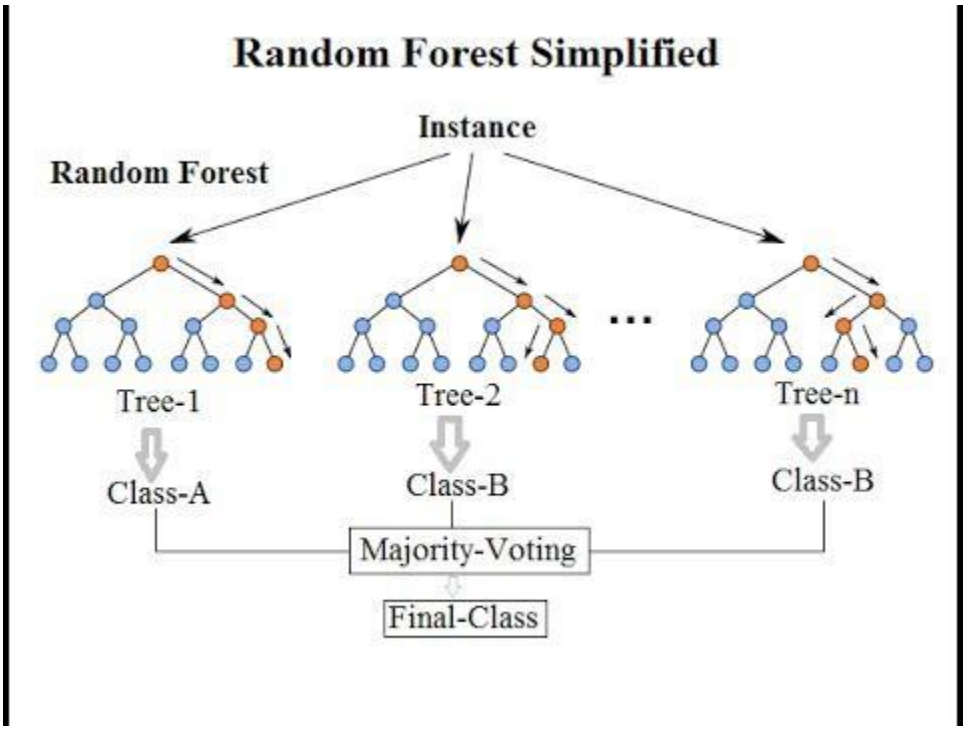
Như tên gọi thì random forest bao gồm 2 phần:

Random: Ngẫu nhiên

Forest: Rừng. Ở đây là nhiều cây quyết định (Decision tree)

Đơn vị của RF là thuật toán cây quyết định với số lượng hàng trăm. Mỗi cây quyết định được tạo ra một cách ngẫu nhiên từ việc: Tái chọn mẫu (bootstrap) và chỉ dùng một phần nhỏ tập biến ngẫu nhiên từ toàn bộ các biến trong dữ liệu. Ở trạng thái sau cùng, mô hình RF thường hoạt động rất chính xác, nhưng đổi lại, ta không thể nào hiểu được cơ chế hoạt động bên trong mô hình vì cấu trúc quá phức tạp.

Trong quá khứ, chúng ta thường chấp nhận đánh đổi tính tường minh để đạt được tín chính xác. Từ mô hình Random Forest, chúng ta chỉ có thể làm một số khảo sát hạn chế, bao gồm vai trò tương đối của các biến (features) và vẽ các biểu đồ 2 chiều thể hiện ranh giới các vùng phân loại.



Hình 1:

2.2. Ý tưởng về khái niệm Bootstrap trong Random Forest

Bootstrap là phương pháp tại chọn ngẫu nhiên và độc lập có trùng lặp từ tập dữ liệu train gốc thành nhiều tập mẫu hơn. Đơn thuần nó chỉ là việc lấy mẫu ngẫu nhiên có trùng lặp những hàng của tập dữ liệu train. Khi mà lấy mẫu có cho phép trùng lặp thì mỗi hàng có thể lặp lại nhiều lần và một số hàng thì bị vắng mặt. Cái ý tưởng của nó là muốn tạo ra một tập dữ liệu mới mà có giữ được một số những tính chất đặc trưng của tập train (dữ liệu gốc ban đầu) vì thế mà chúng ta có thể huấn luyện một mô hình giống nhau cho nhiều tập dữ liệu để lấy ra đặc tính tiêu biểu nhất.

2.3. Ý tưởng của Split-variable randomization

Thuật toán cây quyết định thì nó sẽ bắt đầu bằng cách tìm toàn bộ tất cả những biến đặc tính trong bootstrap sample và sau đó sẽ tìm cái nào là tốt nhất mà biến này có thể dùng để chia tách ra những nhóm có tính đồng nhất cao nhất. Tức là

khả năng phân biệt lớn nhất. Thì vì cái này xảy ra như nhau ở từng cây thì nó sẽ dẫn đến sự không mấy khác biệt nhau về cấu trức ở mỗi cây quyết định trong rừng cây dẫn đến sự gọi là tree correlation như đã nói ở trên. Vì vậy thuật toán Random Forest phải khống chế cá chuyện nầy bằng cách là không đi tìm toàn bộ những biến đặc tính mà chỉ chọn từng cụm ngẫu nhiên trong tổng số biến, rồi đi tìm ra biến tốt nhất trong cụm đó để thực hiện việc chia tách

3. Xây dựng thuật toán Random Forest

3.1. Thuật toán Random Forest

Chuẩn bị tập dữ liệu

Ta có bốn dữ báo

Cân nặng

Lưu lượng máu

Động mạch bị chặn

Tức ngực

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Abnormal	No	No	130	No
Normal	Yes	Yes	195	Yes
Normal	No	Yes	218	No
Abnormal	Yes	Yes	180	Yes

Hình 2: Tập dữ liệu mẫu

Các biến số trên hình 2 dữ đoán liệu một người có bị bệnh tim hay không. Ta sẽ sử dựng Random Forest để dự đoán

3.2. Các bước xây dựng Random Forest

Bước 1: Tạo tập dữ liệu khởi động

Bootstrapping là một phương pháp ước tính được sử dụng để đưa ra dư đoán trên tập dữ liệu bằng cách lấy mẫu lại. Để tạo một tập dữ liệu khởi động, chúng ta

chọn ngẫu nhiên các mẫu từ tập dữ liệu gốc. Một điểm cần lưu ý ở đây là chúng ta có thể chọn cùng một mẫu nhiều lần

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Abnormal	No	No	130	No
Normal	Yes	Yes	195	Yes
Normal	No	Yes	218	No
Abnormal	Yes	Yes	180	Yes

Tập dữ liệu trên đã được chọn mẫu từ tập dữ liệu gốc

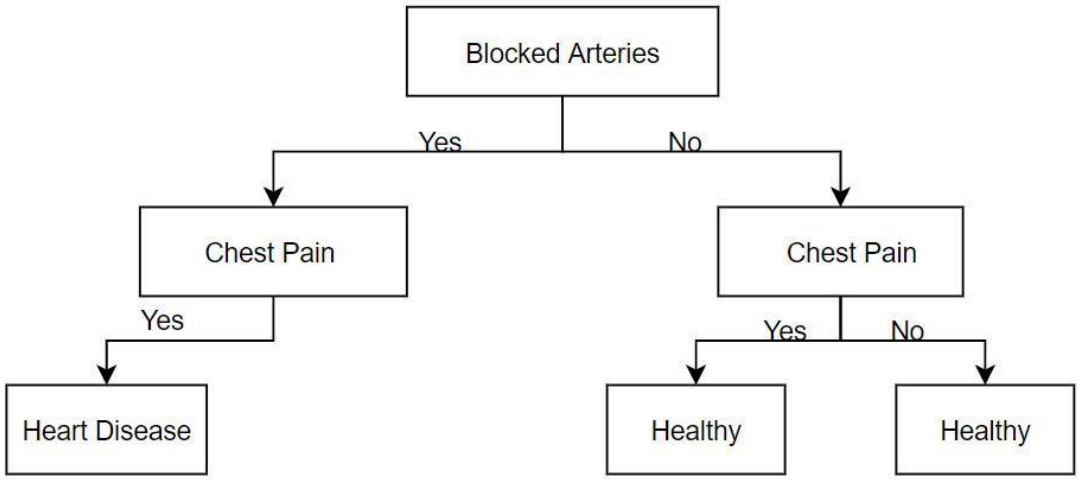
Bước 2: Tạo cây quyết định (Decision Tree)

Tiếp theo ta cần xây dựng cây quyết định bằng cách sử dụng tập dữ liệu khởi động được tạo ở bước 1. Để tạo một khu rừng ngẫu nhiên (Random Forest) thì ta không xem xét toàn bộ dữ liệu mà chỉ sử dụng một tập hợp con ngẫu nhiên của các biến ở mỗi bước

Ví dụ trên ta sẽ xem xét hai biến ở mỗi bước. Ta sẽ bắt đầu ở nút gốc, chọn ngẫu nhiên hai biến làm ứng cử viên cho nút gốc

Giả sử chọn Dòng máu (*Blood Flow*) và Động mạch bị chặn (*Blocked Arteries*). Trong hai biến, ta chọn biến phân tách các mẫu tốt nhất. Vì lợi ích giả sử Động mạch bị chặn (*Blocked Arteries*) là một dự đoán quan trọng và chỉ định làm nút gốc

Tiếp theo là lặp lại quy trình tương tự cho mỗi nút nhánh. Ta lại chọn ngẫu nhiên hai biến làm ứng cử viên cho nhánh rồi lại chọn mẫu biến phân tách tốt nhất

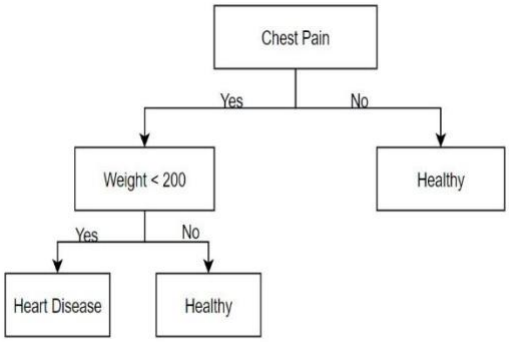


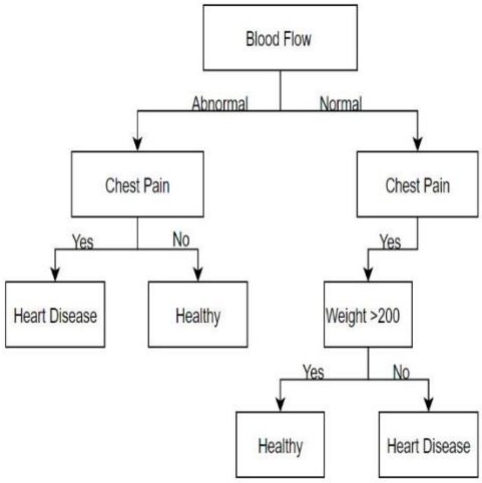
Bước 3: Quay lại bước 1 và lặp lại

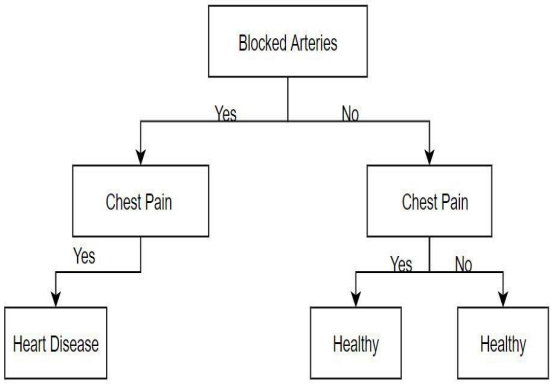
Như đã đề cập ở trước, rừng cây ngẫu nhiên (Random forest) là một tập hợp các cây quyết định. Mỗi cây quyết định dự đoán lớp đầu ra dựa trên các dư báo tương ứng được sử dụng trong cây. Kết quả cuối của tất cả các cây quyết định trong khu rừng ngẫu nhiên sẽ được ghi lại và lớp có đa số phiếu bầu được tính làm lớp đầu ra.

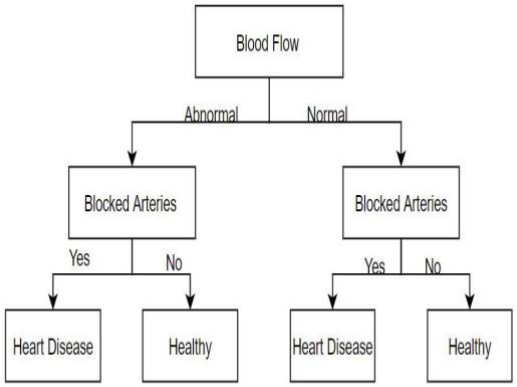
Do đó, cần phải tạo nhiều cây quyết định hơn bằng cách xem xét một tập hợp con của các biến dự báo ngẫu nhiên ở mỗi bước.

Quay lại bước 1, tạo một tập dữ liệu khởi động mới, sau đó xây dựng cây quyết định bằng cách chỉ xem xét một tập hợp con của các biến mỗi bước


Chest Pain
Yes
No
Weight < 200
Healthy
Yes
No
Heart Disease
Healthy


Blood Flow
Abnormal
Normal
Chest Pain
Chest Pain
Yes
No
Yes
Heart Disease
Healthy
Weight >200
Yes
No
Healthy
Heart Disease


Blocked Arteries
Yes
No
Chest Pain
Chest Pain
Yes
Yes
No
Heart Disease
Healthy
Healthy


Blood Flow
Abnormal
Normal
Blocked Arteries
Blocked Arteries
Yes
No
Yes
No
Heart Disease
Healthy
Heart Disease
Healthy

Việc lặp lại được thực hiện 100 lần, tạo ra nhiều cây quyết định với tính toán đầu ra ở mỗi cây

Có nhiều loại cây quyết định trong khu rừng ngẫu nhiên sẽ hiệu quả hơn là một cây riêng lẻ được tạo ra bằng cách sử dụng tất cả các chức năng và toàn bộ tập dữ liệu

Bước 4: Dự đoán kết quả của một điểm dữ liệu mới

Ta đã tạo ra một khu rừng ngẫu nhiên, giờ xem dự đoán của một bệnh nhân mới có bị bệnh tim hay không

Sơ đồ dưới có dữ liệu bệnh nhân mới. Ta sẽ cho chúng chạy dữ liệu xuống các cây quyết định

Cây đầu tiên cho thấy bệnh nhân bị bệnh tim nên cần theo dõi trong bảng thể hiện như hình

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Abnormal	No	Yes	185	

Tương tự chạy dữ liệu xuống các cây quyết định khác và theo dõi dữ đoán của từng cây

Sau đó kiểm tra lớp nào được đa số phiếu bầu, ta thấy lớp “YES” có nhiều phiếu bầu nhất do đó bệnh nhân mới là bị bệnh tim

Sử dụng tất cả các cây định đưa ra quyết định được đọi là Bagging

Bước 5: Đánh giá mô hình

Bước cuối ta cần đánh giá mô hình rừng cây ngẫu nhiên

Trước khi tạo tập dữ liệu khởi động ta đã bổ sót mục nhập/mẫu. Trong mẫu dữ liệu thực khoảng 1/3 dữ liệu gốc không được bao gồm dữ liệu đầu

Đây là mục nhập không kết thúc tỏng tập dữ liệu đầu

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Normal	No	Yes	218	No

Mẫu ngoài túi – Trong rừng ngẫu nhiên

Tập dữ liệu này không bao gồm trong tập dữ liệu khởi động hay là tập dữ liệu Out-Of-Bag(OOB)

Tập dữ liệu Out-Of-Bag sử dụng để kiểm tra độ chính xác của mô hình vì mô hình không được tạo bằng dữ liệu OOB này, nó sẽ giúp ta hiểu rõ về việc mô hình có hiệu quả hay không

Như ví dụ trên, lớp đầu ra cho tập OOB là "NO". Nên rừng ngẫu nhiên của ta là chính xác, nếu chạy dữ liệu OOB xuống các cây quyết định nhận được đa số phiếu là "NO"

Có thể đo độ chính xác của một khu rừng ngẫu nhiên bằng tỉ lệ các mẫu OOB được phân loại chính xác

Tỷ lệ mẫu OOB được phân loại không chính xác được gọi là Lỗi ngoài túi

4. Ưu điểm và nhược điểm

4.1. Ưu điểm

- Một trong những lợi thế lớn nhất của Random Forest là tính linh hoạt của nó. Nó có thể sử dụng cho cả nhiệm classification và regression và cũng dễ dàng xem tầm quan trọng tương đối mà nó gán cho các tính năng đầu vào
- RF là một thuật toán rất tiện dụng vì các siêu tham số mặc định mà nó sử dụng thường tạo ra kết quả dự đoán tốt
- RF có khả năng xử lý các tập dữ liệu lớn với kích thước lớn
- RF nâng cao độ chính xác của mô hình và ngăn vấn đề bị quá mức

4.2. Nhược điểm

Hạn chế chính của RF là số lượng lớn cây có thể làm cho thuật toán quá chậm và không hiệu quả cho các dự đoán thời gian thực.

Được huấn luyện nhanh nhưng lại khá chậm để tạo ra các dự đoán

RF có thể sử dụng cho classification và regression nhưng nó không phù hợp với regression

IV. Code

_ môi trường: colab notebook

_ ngôn ngữ: python

1. Bài toán time series sử dụng decision tree

_ Import thư viện:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

from sklearn.metrics import mean_squared_error
```

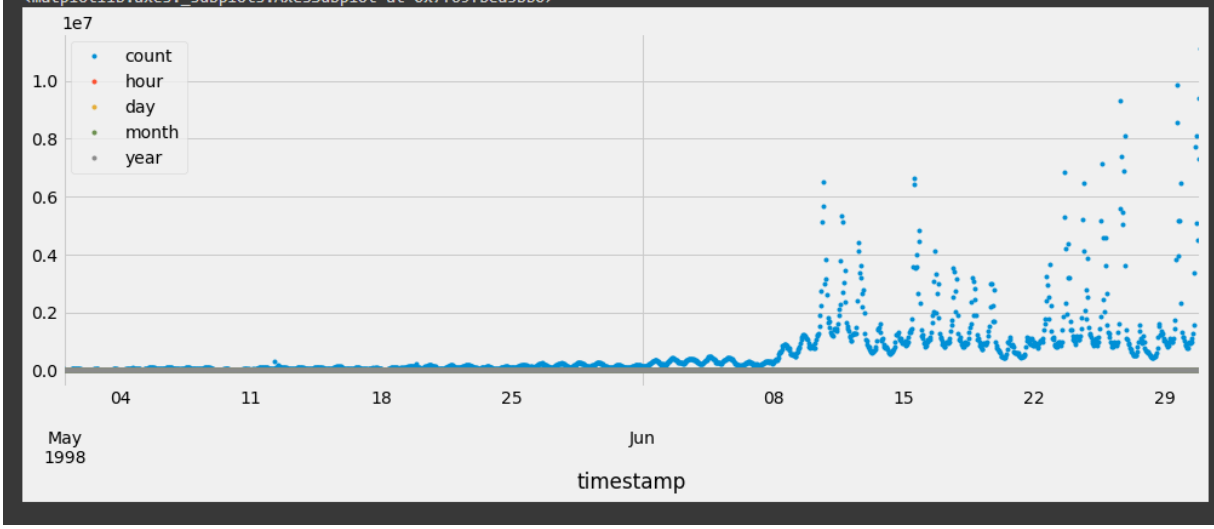
_ load data:

```
df = pd.read_csv("https://raw.githubusercontent.com/thieu1995/ai/master/data/worldcup/wc98_workload_hour.csv")
df = df.set_index('timestamp')
df.index = pd.to_datetime(df.index)
```

_ vẽ biểu đồ data:

```
1 df.plot(style='.', figsize=(15,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f69fbea5bb0>
```



_ tách cột timestamp thành các cột hour, day, month, year:

```
1 df['hour'] = df.index.hour
2 df['day'] = df.index.day
3 df['month'] = df.index.month
4 df['year'] = df.index.year
5 df
```

timestamp	count	hour	day	month	year
1998-04-30 21:00:00	324	21	30	4	1998
1998-04-30 22:00:00	51622	22	30	4	1998
1998-04-30 23:00:00	46813	23	30	4	1998
1998-05-01 00:00:00	39482	0	1	5	1998
1998-05-01 01:00:00	36221	1	1	5	1998
...	...	...	...	...	...
1998-06-30 18:00:00	4493125	18	30	6	1998
1998-06-30 19:00:00	7306176	19	30	6	1998
1998-06-30 20:00:00	9412516	20	30	6	1998
1998-06-30 21:00:00	11102603	21	30	6	1998
1998-06-30 22:00:00	129	22	30	6	1998

1466 rows × 5 columns

_ xem tổng quan của data

```
1 df.describe()
```

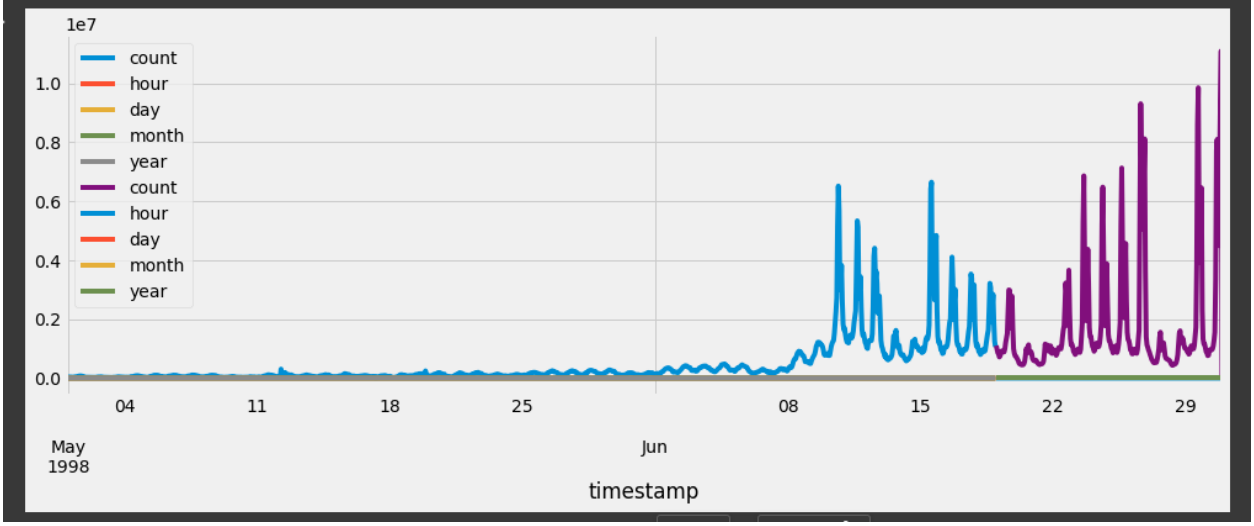
	count	hour	day	month	year
count	1.466000e+03	1466.000000	1466.000000	1466.000000	1466.0
mean	6.957752e+05	11.513643	15.773533	5.488404	1998.0
std	1.202282e+06	6.929692	8.819650	0.504115	0.0
min	1.290000e+02	0.000000	1.000000	4.000000	1998.0
25%	8.244250e+04	6.000000	8.000000	5.000000	1998.0
50%	2.009865e+05	12.000000	16.000000	5.000000	1998.0
75%	9.212275e+05	18.000000	23.000000	6.000000	1998.0
max	1.110260e+07	23.000000	31.000000	6.000000	1998.0

_ chia tập train, test:

```
train = df.loc[df.index < '1998-06-19']
test = df.loc[df.index >= '1998-06-19']
```

_ biểu đồ chia tệp train, test:

```
fig, ax = plt.subplots(figsize=(15,5))
train.plot(ax = ax, label='train set')
test.plot(ax = ax, label='test set')
plt.show()
```



_ xây dựng cây:

```
# xây dựng cây
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
clf_en = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)

# fit the model
clf_en.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
```

_ dự đoán kết quả kiểm tra với entropy, so sánh độ chính xác của tập train và tập test

```
[158]  1 y_pred_en = clf_en.predict(X_test)

[173]  1 from sklearn.metrics import accuracy_score
       2
       3 print('Model accuracy score with criterion entropy: {0:0.5f}'.format(accuracy_score(y_test, y_pred_en)))
       4

       1 y_pred_train_en = clf_en.predict(X_train)
       2 y_pred_train_en

array([[ 89318,     21,     30,      5,   1998],
       [ 89318,     21,     30,      5,   1998],
       [ 89318,     21,     30,      5,   1998],
       ...,
       [796512,     12,     10,      6,   1998],
       [796512,     12,     10,      6,   1998],
       [796512,     12,     10,      6,   1998]])
```

_ in ra điểm trên tập train và tập test:

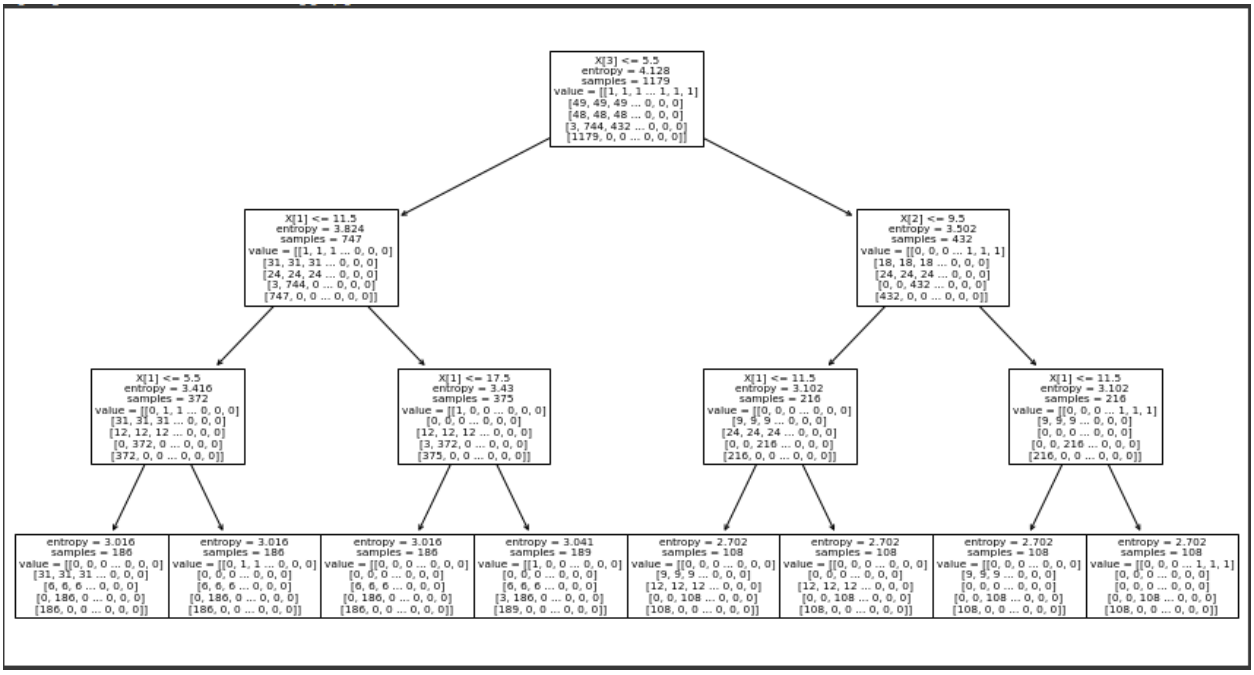
```
1 print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train_en)))
```

```
1 print('Training set score: {:.4f}'.format(clf_en.score(X_train, y_train)))
2
3 print('Test set score: {:.4f}'.format(clf_en.score(X_test, y_test)))
```

_ vẽ cây:

```
1 plt.figure(figsize=(15,8))
2
3 from sklearn import tree
4
5 tree.plot_tree(clf_en.fit(X_train, y_train))
```

```
[Text(0.5, 0.875, 'X[3] <= 5.5\nentropy = 4.128\nsamples = 1179\nvalue = [[1, 1, 1 ... 1, 1, 1]\n[49, 49, 49 ... 0, 0, 0]\n[48, 48, 48 ... 0, 0, 0]\n[3, 744, 432
... 0, 0, 0]\n[1179, 0, 0 ... 0, 0, 0]]'),
 Text(0.25, 0.625, 'X[1] <= 11.5\nentropy = 3.824\nsamples = 747\nvalue = [[1, 1, 1 ... 0, 0, 0]\n[31, 31, 31 ... 0, 0, 0]\n[24, 24, 24 ... 0, 0, 0]\n[3, 744, 0
... 0, 0, 0]\n[747, 0, 0 ... 0, 0, 0]]'),
 Text(0.125, 0.375, 'X[1] <= 5.5\nentropy = 3.416\nsamples = 372\nvalue = [[0, 1, 1 ... 0, 0, 0]\n[31, 31, 31 ... 0, 0, 0]\n[12, 12, 12 ... 0, 0, 0]\n[0, 372, 0
... 0, 0, 0]\n[372, 0, 0 ... 0, 0, 0]]'),
 Text(0.0625, 0.125, 'entropy = 3.016\nsamples = 186\nvalue = [[0, 0, 0 ... 0, 0, 0]\n[31, 31, 31 ... 0, 0, 0]\n[6, 6, 6 ... 0, 0, 0]\n[0, 186, 0 ... 0, 0,
0]\n[186, 0, 0 ... 0, 0, 0]]'),
 Text(0.1875, 0.125, 'entropy = 3.016\nsamples = 186\nvalue = [[0, 1, 1 ... 0, 0, 0]\n[0, 0, 0 ... 0, 0, 0]\n[6, 6, 6 ... 0, 0, 0]\n[0, 186, 0 ... 0, 0,
0]\n[186, 0, 0 ... 0, 0, 0]]'),
 Text(0.375, 0.375, 'X[1] <= 17.5\nentropy = 3.43\nsamples = 375\nvalue = [[1, 0, 0 ... 0, 0, 0]\n[0, 0, 0 ... 0, 0, 0]\n[12, 12, 12 ... 0, 0, 0]\n[3, 372, 0 ...
0, 0, 0]\n[375, 0, 0 ... 0, 0, 0]]'),
 Text(0.3125, 0.125, 'entropy = 3.016\nsamples = 186\nvalue = [[0, 0, 0 ... 0, 0, 0]\n[0, 0, 0 ... 0, 0, 0]\n[6, 6, 6 ... 0, 0, 0]\n[0, 186, 0 ... 0, 0,
0]\n[186, 0, 0 ... 0, 0, 0]]'),
 Text(0.4375, 0.125, 'entropy = 3.041\nsamples = 189\nvalue = [[1, 0, 0 ... 0, 0, 0]\n[0, 0, 0 ... 0, 0, 0]\n[6, 6, 6 ... 0, 0, 0]\n[3, 186, 0 ... 0, 0,
0]\n[189, 0, 0 ... 0, 0, 0]]'),
 Text(0.75, 0.625, 'X[2] <= 9.5\nentropy = 3.502\nsamples = 432\nvalue = [[0, 0, 0 ... 1, 1, 1]\n[18, 18, 18 ... 0, 0, 0]\n[24, 24, 24 ... 0, 0, 0]\n[0, 0, 432
... 0, 0, 0]\n[432, 0, 0 ... 0, 0, 0]]'),
 Text(0.625, 0.375, 'X[1] <= 11.5\nentropy = 3.102\nsamples = 216\nvalue = [[0, 0, 0 ... 0, 0, 0]\n[9, 9, 9 ... 0, 0, 0]\n[24, 24, 24 ... 0, 0, 0]\n[0, 0, 216
... 0, 0, 0]\n[216, 0, 0 ... 0, 0, 0]]'),
 Text(0.5625, 0.125, 'entropy = 2.702\nsamples = 108\nvalue = [[0, 0, 0 ... 0, 0, 0]\n[9, 9, 9 ... 0, 0, 0]\n[12, 12, 12 ... 0, 0, 0]\n[0, 0, 108 ... 0, 0,
0]\n[108, 0, 0 ... 0, 0, 0]]'),
 Text(0.6875, 0.125, 'entropy = 2.702\nsamples = 108\nvalue = [[0, 0, 0 ... 0, 0, 0]\n[0, 0, 0 ... 0, 0, 0]\n[12, 12, 12 ... 0, 0, 0]\n[0, 0, 108 ... 0, 0,
0]\n[108, 0, 0 ... 0, 0, 0]]'),
 Text(0.875, 0.375, 'X[1] <= 11.5\nentropy = 3.102\nsamples = 216\nvalue = [[0, 0, 0 ... 1, 1, 1]\n[9, 9, 9 ... 0, 0, 0]\n[0, 0, 0 ... 0, 0, 0]\n[0, 0, 216 ...
0, 0, 0]\n[216, 0, 0 ... 0, 0, 0]]'),
 Text(0.8125, 0.125, 'entropy = 2.702\nsamples = 108\nvalue = [[0, 0, 0 ... 0, 0, 0]\n[9, 9, 9 ... 0, 0, 0]\n[0, 0, 0 ... 0, 0, 0]\n[0, 0, 108 ... 0, 0,
0]\n[108, 0, 0 ... 0, 0, 0]]'),
 Text(0.9375, 0.125, 'entropy = 2.702\nsamples = 108\nvalue = [[0, 0, 0 ... 1, 1, 1]\n[0, 0, 0 ... 0, 0, 0]\n[0, 0, 0 ... 0, 0, 0]\n[0, 0, 108 ... 0, 0,
0]\n[108, 0, 0 ... 0, 0, 0]]')]
```



_ confusion metrics:

```
1 from sklearn.tree import DecisionTreeClassifier
2 y_pred = classifier.predict(X_test)
```

```
1 from sklearn.metrics import classification_report, confusion_matrix
2 print(confusion_matrix(y_test, y_pred))
3 print(classification_report(y_test, y_pred))
```

```
1 from sklearn.metrics import confusion_matrix
2
3 cfm = confusion_matrix(y_test, y_pred_en)
4
5 print('Confusion matrix\n\n', cfm)
```

_ Classification Report:

```
1 from sklearn.metrics import classification_report
2
3 print(classification_report(y_test, y_pred_en))
```

2. Bài toán time series sử dụng ranmdom forest

_ tạo khu rừng đầu tiên

```
1 from sklearn.ensemble import RandomForestClassifier
2 rf_model = RandomForestClassifier(n_estimators=50, max_features="auto", random_state=44)
3 rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=50, random_state=44)
```

_ dự đoán:

```
1 predictions = rf_model.predict(X_test)
2 predictions
```

```
array([[1144634,       0,      18,       6,    1998],
       [1028319,       1,      18,       6,    1998],
       [ 899553,       2,      18,       6,    1998],
       ...,
       [2839307,      20,      18,       6,    1998],
       [1892494,      20,      18,       6,    1998],
       [ 174319,      22,      18,       6,    1998]])
```

_ Có thể kiểm tra xác suất được chỉ định bởi mô hình của mình cho từng dự đoán bằng predict_proba():

```
1 rf_model.predict_log_proba(X_test)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:906: RuntimeWarning: divide by zero encountered in log
  proba[k] = np.log(proba[k])
[array([[       -inf,        -inf,        -inf, ...,        -inf,
                -inf,        -inf],
       [       -inf,        -inf,        -inf, ...,        -inf,
                -inf,        -inf],
       [       -inf,        -inf,        -inf, ...,        -inf,
                -inf,        -inf],
       ...,
       [       -inf,        -inf,        -inf, ...,        -inf,
                -inf,        -inf],
       [       -inf,        -inf,        -inf, ...,        -inf,
                -inf,        -inf],
       [-3.21887582, -3.21887582, -3.21887582, ...,        -inf,
                -inf,        -inf]]),
 array([[-0.61618614, -1.51412773, -2.81341072, ...,        -inf,
                -inf,        -inf],
       [-2.52572864, -0.54472718, -2.30258509, ...,        -inf,
                -inf,        -inf],
       [-3.91202301, -2.52572864, -0.82098055, ...,        -inf,
                -inf,        -inf],
       ...,
       [       -inf,        -inf,        -inf, ...,        -inf,
                -inf,        -inf],
       [       -inf,        -inf,        -inf, ..., -1.51412773,
```

3. Link code

https://colab.research.google.com/drive/1Er-LXNaazWd1rdjL6riXKwxu72Vs-Lo7#scrollTo=5vG4TktBdMS1&uniqifier=1

V. Tài liệu tham khảo

https://machinelearningcoban.com/tabml_book/ch_model/random_forest.html

https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

https://machinelearningcoban.com/tabml_book/ch_model/decision_tree.html

https://viblo.asia/p/tan-man-mot-chut-ve-time-series-data-p1-1VgZv6DpZAw

https://insights.magestore.com/posts/giai-thuat-time-series-forecasting

https://viblo.asia/p/time-series-data-gDVK2Qbv5Lj

VI. Lời cảm ơn

Cảm ơn thầy Nguyễn Văn Thiệu và các bạn đã hỗ trợ chúng em trong quá trình hoàn thành đồ án này