ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
——————— * ———————

# BÀI CÁO PROJECT 2

## THUẬT TOÁN
## RỪNG NGẪU NHIỆN

**Sinh viên thực hiện** **Mã sinh viên**
Trịnh Quốc Đạt 20194248

Giảng viên hướng dẫn : PGS.TS. **Trịnh Văn loan**

*Hà Nội, tháng 8 năm 2022*

# Nội dung

# TÓM TẮT

Rừng ngẫu nhiên là một kỹ thuật máy học được sử dụng để giải quyết các vấn đề hồi quy và phân loại. Nó sử dụng phương pháp tập hợp, là một kỹ thuật kết hợp nhiều bộ phân loại để cung cấp giải pháp cho các vấn đề phức tạp.Một thuật toán rừng ngẫu nhiên bao gồm nhiều cây quyết định. 'Rừng' được tạo bởi thuật toán rừng ngẫu nhiên được đào tạo thông qua mô hình tổng hợp và lấy mẫu tái lặp. Lấy mẫu tái lặp là một thuật toán tổng hợp giúp cải thiện độ chính xác của các thuật toán học máy. Thuật toán rừng ngẫu nhiên đưa ra kết quả dự đoán dựa trên các dự đoán của cây quyết định. Nó dự đoán bằng cách lấy giá trị trung bình từ các cây khác nhau. Tăng số lượng cây sẽ tăng độ chính xác của kết quả. Thuật toán rừng ngẫu nhiên loại bỏ những hạn chế của thuật toán cây quyết định. Nó làm giảm vấn đề quá khớp bộ dữ liệu và tăng độ chính xác. Nó tạo ra các dự đoán mà không yêu cầu nhiều cấu hình khi lập trình.

# I. Giới thiệu

Với nhu cầu tính toán phức tạp hơn, chúng ta không thể dựa vào các thuật toán đơn giản. Thay vào đó, chúng ta phải sử dụng các thuật toán có khả năng tính toán cao hơn và một trong những thuật toán như vậy là rừng ngẫu nhiên. Rừng ngẫu nhiên là một thuật toán học máy có giám sát được xây dựng từ các thuật toán cây quyết định. Thuật toán này được áp dụng trong các ngành khác nhau như ngân hàng và thương mại điện tử để dự đoán hành vi và kết quả.

Bài báo cáo này cung cấp một cái nhìn tổng quan về thuật toán rừng ngẫu nhiên và cách thức hoạt động của nó. Qua đó, bài báo cáo sẽ giới thiệu qua mô hình cây quyết định là cơ sở của mô hình rừng ngẫu nhiên ở phần 2. Trong phần 3, bài báo cáo trình bày các tính năng của thuật toán và cách nó được sử dụng trong các ứng dụng thực tế. Đồng thời chỉ ra những ưu điểm và nhược điểm của thuật toán này. Để hiểu kỹ hơn cách thức hoạt động của thuật toán, ở phần 4, bài báo cáo sẽ minh họa thuật toán bằng python.

# II. Cây quyết định

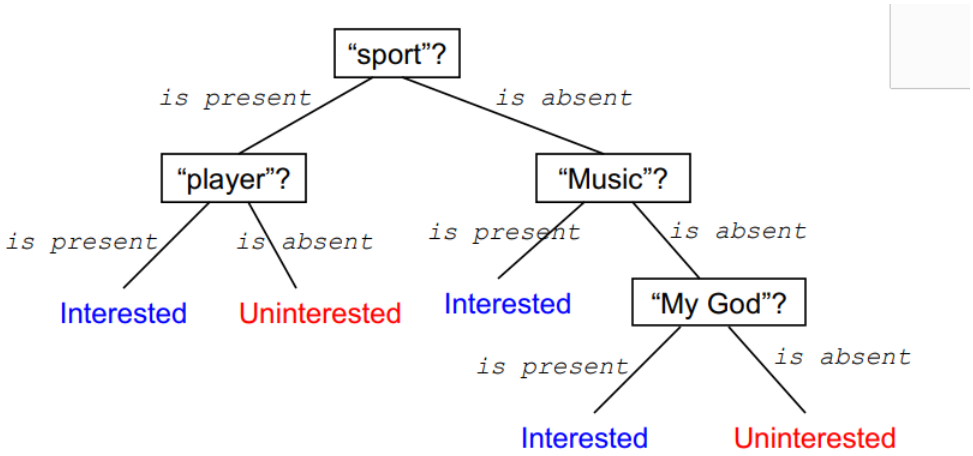
## 1. Giới thiệu về cây quyết định

Cây quyết định là một kỹ thuật học máy có giám sát, mô hình hóa các quyết định, kết quả và dự đoán bằng cách sử dụng cấu trúc cây. Một cây quyết định sử dụng một tập các luật nếu-thì để xác định cách tách, phân loại và trực quan hóa một tập dữ liệu dựa trên

các điều kiện khác nhau. Mô hình cây quyết định là một mô hình được sử dụng khá phổ biến và hiệu quả trong cả hai lớp bài toán phân loại và dự báo của học có giám sát. Khác với những thuật toán khác trong học có giám sát, mô hình cây quyết định không tồn tại phương trình dự báo. Mọi việc chúng ta cần thực hiện đó là tìm ra một cây quyết định dự báo tốt trên tập huấn luyện và sử dụng cây quyết định này dự báo trên tập kiểm tra.

Cây quyết định có 2 loại:

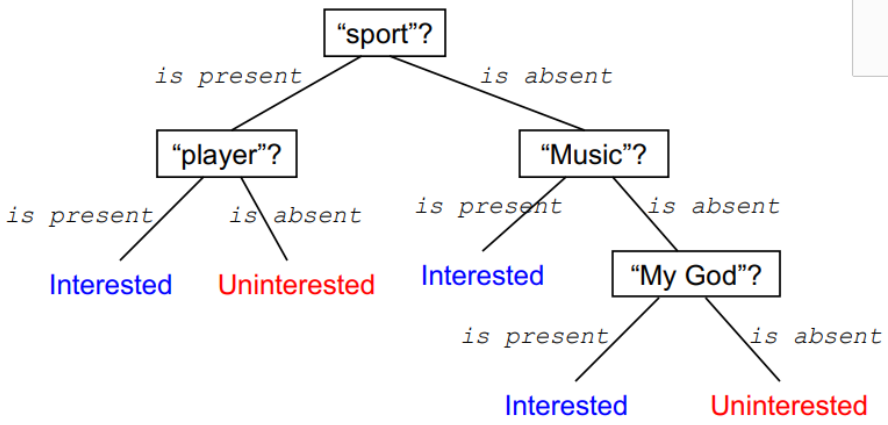
- Cây hồi quy (Regression tree) ước lượng các hàm giá có giá trị là số thực thay vì được sử dụng cho các nhiệm vụ phân loại. (ví dụ: ước tính giá một ngôi nhà hoặc khoảng thời gian một bệnh nhân nằm viện) [2].
- Cây phân loại (Classification tree) ước lượng các hàn có giá trị rời rạc – hàm phân lớp. Kết quả dự đoạn là lớp rời rạc mà dữ liệu thuộc về [2].




*Hình 1. Ví dụ về cây quyết định [1]*

Trong hình 1, ta có thể thấy cách quây quyết định đưa ra kết quả phán đoán. Nếu tập giá trị là (…,"sport",…,"player",…) thì cây quyết định sẽ trả về Interested. Hoặc nếu tập giá trị là (…,"My God",…,) thì cũng sẽ trả về Interested.

Trong cây phân loại, mỗi nút biểu diễn một thuộc tính, mỗi nhánh xuất phát từ một nút sẽ tương ứng với một giá trị của thuộc tính nút đó và mỗi lá của cây sẽ biểu diễn một nhãn lớp cần phân loại. Từ cách biểu diễn vậy, mỗi đường đi từ nút gốc đến nút lá sẽ tương ứng bởi một kết hợp của các phép kiểm tra giá trị thuộc tính. Từ đó, cây quyết định có thể hiểu là một phép tuyển của các kết hợp này.




["sport" is present) ∧ ("player" is present)] ∨

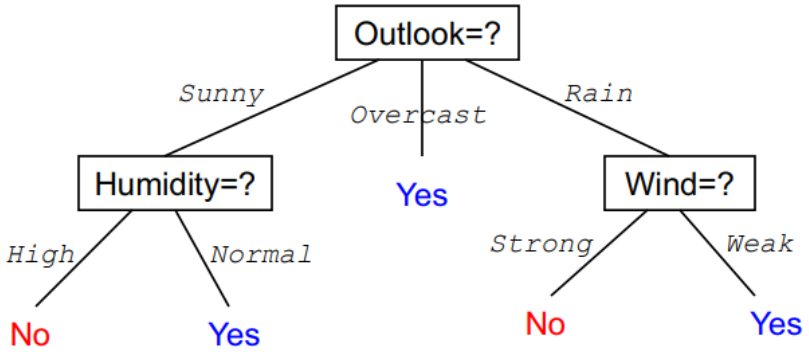
["sport" is absent) ∧ ("Music" is present)] ∨

["sport" is absent) ∧ ("Music" is absent) ∧ ("My God" is present)]

*Hình 2. Cây quyết định là phép tuyển của các kết hợp [1]*

Từ hình 2, ta có thể thấy cây quyết định được biểu diễn bởi các phép tuyển, phép hội của các phép thử thuộc tính: "sport" is present, "player" is present, "music" is present, "My God" is present. Vì thế nếu biểu thức này trả về 1, thì cây quyết định sẽ trả về nhãn Interested, ngược lại nếu biểu thức này trả về 0, cây quyết định sẽ trả về Uninterested.

Sau khi học được cây quyết định, chúng ta có thể dự đoán được nhãn cho dữ liệu cần phán đoán bằng cách sử dụng các thuộc tính của nó để đi từ rễ xuống lá.




*Hình 3. Dự đoán cây quyết định. [1]*

Dựa vào cây quyết định ở hình 3 ta có thể dự đoán tập hợp các giá trị thuộc tính (Outlook=Overcast, Temperature=Hot, Humidity=High, Wind=Weak) sẽ trả về nhãn Yes. Hoặc tập hợp

các giá trị thuộc tính tính (Outlook=Rain, Temperature=Mild, Humidity=High, Wind=Strong) sẽ trả về nhãn No.

## 2. Học cây quyết định

Học cây quyết định là một trong các phương pháp học quy náp được dùng phổ biến nhất, được áp dùng thành công trong rất nhiều bài toán ứng dụng thực tế. Một cây quyết định sử dụng thuộc toán tham lam được học bằng cách chia tập hợp dữ liệu nguồn thành các tập con dựa trên một phép kiểm tra giá trị thuộc tính. Việc phân chia này được dựa trên một tập hợp quy tắc phân tách các đặc điểm phân loại. Quá trình này được lặp lại một cách đệ quy cho mỗi tập con dẫn xuất. Quá trình đệ quy hoàn thành khi không thể tiếp tục thực hiện việc chia tách được nữa, hay khi một phân loại đơn có thể áp dụng cho từng phần tử của tập con [2].

Những thuộc tính giúp chúng ta phân tách cây tốt được gọi là những thuộc tính có tính tách biệt. Để đánh giá tính tách biệt các thuộc tính ta sẽ sử dụng các chỉ số đánh giá như: Độ lợi thông tin ( infomation gain), Chỉ số Gini (Gini index), độ lệch chuẩn (standard deviation).

### 2.1. Infomation gain

Để tính được Infomation gain, chúng ta cần phải sử dụng Entropy. Entropy xác định tính ngẫu nhiên trong thông tin được xử lý và dùng để đo lường mức độ không chắc chắn trong thông tin đó. Entropy càng cao thì thuộc tính đó sẽ càng khó phán đoán. Entropy của tập s gồm c lớp:

$$entropy(s) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

Trong đó $p_i$ là xác tỉ lệ của phần tử có lớp i trong s.

Information gain giúp cho ta đo đạc độ giảm entropy khi ta chia thuộc tính thành nhiều thành tập con. Vì thế nó thể hiện lượng thông tin trung bình bị mất khi chi S theo A. Infomation gain của thuộc tính A trong tập S được tính:

$$Gain(S, a) = entropy(S) - \sum_{v \in vals(A)} \frac{|s_v|}{|s|} entropy(S_v)$$

Trong đó, $s_v$ là tập dữ liệu nà thuộc tính a có giá trị bằng v.

## 2.2. Gini index

Gini index là chỉ số đo lường mức độ đồng nhất hay mức độ nhiễu loạn thông tin, hay sự khác biệt về các giá trị mà mỗi điểm dữ liệu trong một tập hợp con hoặc một nốt trên cây quyết định có. Gini index ngoài áp dụng cho cây phân loại còn có thể dùng cho cây hồi quy. Công thức tổng quát của hệ số Gini

$$gini(t) = 1 - \sum_{j} [p(j|t)]^2$$

Với t là một nút bất kỳ có chứa các điểm dữ liệu mang thuộc tính j của biến mục tiêu và p chính là xác suất để một điểm dữ liệu trong t có thuộc tính j, nếu tất cả các điểm dữ liệu đều mang thuộc tính j thì lúc này p = 1 và gini(t) = 0, nút t lúc đó sẽ được công nhận là nút "tinh khiết"

Công thức trên để tính độ đồng nhất của một nút, vậy khi chúng ta có nhiều cách phân nhánh, mỗi cách có thể phân ra một số nút nhất định, lúc này chúng có thêm công thức thứ hai để tìm ra cách phân tách tốt nhất.

$$Gini_{split} = \sum_{i=1}^{k} \frac{n_i}{n} gini(i)$$

Với $n_i$ là số điểm dữ liệu có trong nút con, nút của nhánh được phân còn n là số quan sát số điểm dữ liệu có trong nút cha, nút được dùng để phân nhánh. Hệ số gini split càng nhỏ tức cách phân nhánh càng tối ưu.

## 2.3. Độ lệch chuẩn

Khi xây dựng cây hồi quy ngoài gini index, chúng ta còn có thể sử dụng độ lệch chuẩn. Việc phân nhánh trong lúc trong lúc xây dựng mô hình cây quyết định đó chính là việc phân chia tập dữ liệu thành các tập con trên cơ sở chúng sẽ đồng nhất về giá trị sau cùng của biến mục tiêu. Nếu tất cả các đối tượng dữ liệu trong nút đồng nhất về giá trị của biến mục tiêu thì độ lệch chuẩn sẽ bằng không. Để hiểu độ lệch chuẩn, trước tiên chúng ta phải hiểu phương sai.

Phương sai là trung bình cộng của bình phương các độ lệch giữa các giá trị của từng quan sát và số trung bình cộng của dãy số. Độ lệch chuẩn chính là căn bậc 2 của phương sai. Phương sai lớn phản ánh khuynh hướng phân tán nhiều, và độ biến thiên cao của dữ liệu, độ lệch chuẩn đại diện cho một giá trị trung bình, là chênh lệch giữa giá trị của mỗi quan sát so với trung bình cộng

của dãy số do đó cũng thể hiện được độ biến thiên, độ lệch chuẩn càng cao thì dãy số phân tán nhiều và ngược lạ. Do đó nếu trong một nút, các đối tượng dữ liệu có giá trị định lượng của biến mục tiêu quá chênh lệch hay khác biệt thì độ lệch chuẩn sẽ rất lớn, cách phân nhánh này có thể không hiệu quả và ngược lại. Giá trị của biến mục tiêu sau cùng tại các nút lá sẽ là giá trị trung bình của các đối tượng dữ liệu bên trong nút. Công thức độ lệch chuẩn:

$$\text{Standard deviation} = \sqrt{\frac{\sum f x^2}{\sum f} - \left(\frac{\sum f x}{\sum f}\right)^2}$$

### 3. Ưu điểm nhược điểm

- Ưu điểm:
  + Thuật toán đơn giản để hiểu, diễn giải và trực quan vì ý tưởng chủ yếu được sử dụng trong cuộc sống hàng ngày của chúng ta. Đầu ra của cây quyết định có thể được con người giải thích một cách dễ dàng
  + Rất hữu ích để giải quyết các vấn đề liên quan đến quyết định.
  + Việc chuẩn bị dữ liệu cho một cây quyết định là cơ bản hoặc không cần thiết. Các kỹ thuật khác thường đòi hỏi chuẩn hóa dữ liệu, cần tạo các biến phụ và loại bỏ các giá trị rỗng.
- Nhược điểm:
  + Quá khớp: Đây là vấn đề chính của cây quyết định. Nó thường dẫn đến việc quá khớp dữ liệu mà cuối cùng dẫn đến dự đoán sai. Để khớp với dữ liệu (ngay cả dữ liệu nhiễu), nó tiếp tục tạo ra các nút mới và cuối cùng cây trở nên quá phức tạp để diễn giải, dẫn đến mất khả năng tổng quát hóa
  + Phương sai cao: Như đã đề cập ở trên, cây quyết định thường dẫn đến việc quá khớp dữ liệu. Chính vì thế, có rất nhiều khả năng sai lệch cao trong đầu ra, dẫn đến nhiều sai sót trong ước tính cuối cùng và cho thấy kết quả không chính xác cao. Quá khớp dẫn đến phương sai cao
  + Không ổn định: Việc thêm một điểm dữ liệu mới có thể dẫn đến việc tạo lại cây tổng thể và tất cả các nút cần được tính toán lại và tạo lại

# III. Rừng ngẫu nhiên

## 1. Ý tưởng cốt lõi

Ở phần trên, chúng ta đã tìm hiểu về cây quyết định, để giải quyết vấn đề chính của cây quyết định là quá khớp, chúng ta sẽ

dùng thuật toán rừng ngẫu nhiên. Nếu như sức mạnh của một cây quyết định là yếu thì hợp sức của nhiều cây quyết định sẽ trở nên mạnh mẽ hơn. Ý tưởng của sự hợp sức đó đã hình thành nên mô hình rừng ngẫu nhiên. Mô hình rừng cây được huấn luyện dựa trên sự phối hợp giữa luật tổng hợp và quá trình lấy mẫu tái lặp (bootstrap aggregation/bagging).

Với phương pháp tổng hợp, thuật toán sẽ tạo ra nhiều cây quyết định mà mỗi cây quyết định được huấn luyện dựa trên nhiều mẫu con khác nhau và kết quả dự báo là bầu cử từ toàn bộ những cây quyết định. Như vậy một kết quả dự báo được tổng hợp từ nhiều mô hình nên kết quả của chúng sẽ không bị chệch. Đồng thời kết hợp kết quả dự báo từ nhiều mô hình sẽ có phương sai nhỏ hơn so với chỉ một mô hình. Điều này giúp cho mô hình khắc phục được hiện tượng quá khớp. Thông thường những kết quả từ phương pháp kết hợp sẽ tốt hơn so với chỉ sử dụng một mô hình bởi chúng ta đang vận dụng trí thông minh đám đông. Điều này đã được kiểm chứng ở nhiều lớp mô hình khác nhau trong thực nghiệm.

Khi lấy mẫu tái lặp, chúng ta sẽ xây dựng tập huấn luyện cho từng cây bằng cách lấy mẫu có thay thế từ dữ liệu gốc. Như vậy nếu sử dụng dự báo là trung bình kết hợp từ nhiều mô hình cây quyết định thì phương sai có thể giảm nhiều lần so với chỉ sử dụng một mô hình duy nhất. Trong một mô hình rừng cây, số lượng các cây quyết định là rất lớn. Do đó phương sai dự báo từ mô hình có thể giảm gấp nhiều lần và tạo ra một dự báo ổn định hơn. Vì mỗi cây quyết định trong thuật toán rừng ngẫu nhiên không dùng tất cả dữ liệu huấn luyện, cũng như không dùng tất cả các thuộc tính của dữ liệu để xây dựng cây nên mỗi cây có thể sẽ dự đoán không tốt, khi đó mỗi mô hình cây quyết định không bị quá khớp mà có thể chưa khớp, hay nói cách khác là mô hình có độ chệch cao. Tuy nhiên, kết quả cuối cùng của thuật toán rừng ngẫu nhiên lại tổng hợp từ nhiều cây quyết định, thế nên thông tin từ các cây sẽ bổ sung thông tin cho nhau, dẫn đến mô hình có độ chệch thấp và phương sai thấp, hay mô hình có kết quả dự đoán tốt.

Quy trình trên đã mô tả thuật toán bagging cơ bản cho nhiều cây. Ngoài ra, rừng ngẫu nhiên còn sử dụng thêm một mô hình bagging khác: features bagging/ bagging thuộc tính. Trong mô hình này, tại mỗi lần phân nhánh, thuật toán sẽ lựa chọn tập hợp con ngẫu nhiên thuộc tính cho từng nhánh. Lý do cho quá trình này là để giảm sự tương quan của các cây trong mô hình lấy mẫu tái lặp thông thường

trong trường hợp một vài thuộc tính có "khả năng phán đoán" tốt hơn [3].

## 2. Thuật toán

Thuật toán rừng ngẫu nhiên có ba siêu tham số chính, cần được thiết lập trước khi huấn luyện. Đó chính là kích thuớc nút, số lượng cây và số lượng đối tượng được lấy mẫu. Từ đó, mô hình rừng ngẫu nhiên có thể được sử dụng để giải quyết các vấn đề hồi quy hoặc phân loại.

Thuật toán rừng ngẫu nhiên được tạo thành từ một tập hợp các cây quyết định và mỗi cây trong tập hợp bao gồm một mẫu dữ liệu được rút ra từ một tập huấn luyện có thay thế, được gọi là mẫu tái lặp. Trong số mẫu huấn luyện đó, một phần ba được dành làm dữ liệu thử nghiệm, được gọi là mẫu nằm ngoái túi. Để tăng tính tương quan cao giữa các cây và làm cho mỗi cây phân chia một cách ngẫu nhiên nhất có thể chúng ta sẽ sử dung features bagging. Tùy thuộc vào vấn đề cần giải quyết mà việc đưa ra dự đoán sẽ khác nhau. Đối với bài toán hồi quy, kết quả dự đoán của các cây quyết định riêng lẻ sẽ được tính trung bình và đối với bài toán phân loại, kết quả dự báo sẽ là bầu cử từ các cây quyết định. Cuối cùng, mẫu ngoài túi sau đó được sử dụng để xác nhận chéo, hoàn thiện dự đoán đó.

## 3. Ưu điểm, nhược điểm và ứng dụng

### 3.1 Ưu điểm

Nếu như mô hình cây quyết định thường bị nhạy cảm với dữ liệu ngoại lai thì mô hình rừng cây được huấn luyện trên nhiều tập dữ liệu con khác nhau, trong đó có những tập được loại bỏ dữ liệu ngoại lai, điều này giúp cho mô hình ít bị nhạy cảm với dữ liệu ngoại lai hơn.

Sự kết hợp giữa các cây quyết định giúp cho kết quả ít bị chệch và phương sai giảm. Như vậy chúng ta giảm thiểu được hiện tượng quá khớp ở mô hình rừng cây, một điều mà mô hình cây quyết định thường xuyên gặp phải.

Cuối cùng các bộ dữ liệu được sử dụng từ những cây quyết định đều xuất phát từ dữ liệu huấn luyện nên quy luật học giữa các cây quyết định sẽ gần tương tự như nhau và tổng hợp kết quả giữa chúng không có xu hướng bị chệch.

Với mô hình rừng ngẫy nhiên, các cây được tạo độc lập từ các tập dữ liệu và thuộc tính khác nhau. Vì thế nên người lập trình có thể tận dụng tối đa sức mạnh của CPU để xây dựng mô hình.

## 3.2 Nhược điểm

Mặc dù mô hình rừng ngẫu nhiên thường đạt độ chính xác cao hơn mô hình cây quyết định nhưng nó lại hi sinh đi tính trực quan, dễ hiểu của cây quyết định. Đây là một trong những đặc tính quan trọng nhất của cây quyết định. Nó giúp người lập trình đảm bảo được mô hình đã học được từ dữ liệu và cho phép người dùng tin tưởng vào các phán đoán của mô hình.

Ngoài ra, trong các bài toán với nhiều lớp phân loại, rừng ngẫu nhiên có thể không thể tăng độ chính xác của phương pháp học cơ sở.

## 3.3 Ứng dụng

Thuật toán rừng ngẫu nhiên đã được áp dụng trên một số lĩnh vực, cho phép chúng ta đưa ra các quyết định tốt hơn. Một số lĩnh vực bao gồm:

- Tài chính: Đây là một thuật toán được ưa thích hơn các thuật toán khác vì nó giảm thời gian dành cho các tác vụ quản lý và xử lý trước dữ liệu. Trong lĩnh vực tài chính, mô hình rừng ngẫu nhiên có thể được sử dụng để dự đoán các khoản nợ thế chấp và xác định hoặc ngăn chặn lừa đảo. Do đó, thuật toán có thểxác định xem khách hàng có khả năng vỡ nợ hay không. Để phát hiện lừa đảo, nó có thể phân tích một loạt các giao dịch và xác định xem chúng có khả năng là giao dịch lừa đảo không. Một ứng dụng khác trong tài chính đó là có thể đào tạo rừng ngẫu nhiên để ước tính xác suất khách hàng chấm dứt tài khoản của họ dựa trên lịch sử giao dịch và tính thường xuyên của các giao dịch. Nếu chúng ta áp dụng mô hình này cho tất cả người dùng hiện tại, chúng ta có thể dự đoán tình hình ngừng hoạt động trong vài tháng tới. Điều này cung cấp cho công ty những thông tin kinh doanh cực kỳ hữu ích giúp công ty xác định các điểm nghẽn và xây dựng mối quan hệ đối tác lâu dài với khách hàng.

- Chăm sóc sức khỏe: Thuật toán rừng ngẫu nhiên có các ứng dụng trong sinh học tính toán, cho phép các bác sĩ giải quyết các vấn đề như phân loại biểu hiện gen, phát hiện dấu ấn sinh học và chú thích trình tự. Do đó, các bác sĩ có thể đưa ra các ước tính về phản ứng của thuốc đối với các loại thuốc cụ thể.

- Thương mại điện tử: Thuật toán ngày càng được sử dụng rộng rãi trong thương mại điện tử để dự báo doanh số bán hàng. Sử dụng mô hình rừng ngẫu nhiên trên dữ liệu khách hàng như độ tuổi, giới tính, sở thích sẽ cho phép chúng ta dự đoán khách hàng nào sẽ mua và khách hàng nào không mua với độ chính xác tương đối cao. Và bằng cách nhắm mục tiêu chiến dịch quảng cáo đến những người mua tiềm năng, chúng ta sẽ tối ưu hóa chi tiêu tiếp thị và tăng doanh số bán hàng

# IV. Cách triển khai

Chúng ta sẽ tạo rừng ngẫu nhiên cho bài toán hồi quy. Để bắt đầu chúng ta sẽ xác định các tham số của rừng ngẫu nhiên:

1. x: Các biến độc lập của tập huấn luyện
2. y: Các biến phụ thuộc tương ứng cần thiết cho việc học có giám sát
3. n_trees: Số lượng cây không tương để tạo ra rừng ngẫu nhiên.
4. n_features: Số lượng thuộc tính để lấy mẫu và chuyển cho mỗi câu. Đây là nơi chúng ta sẽ sử dụng đóng bao thuộc tính. Nó có thể nhận giá trị là sqrt, log2, hoặc một số nguyên. Trong trường hợp giá trị là sqrt, số thuộc tính mà được lấy mẫu cho mỗi cây là căn bậc 2 của tổng số thuộc tính.
5. sample_size: Số hàng dữ liệu của tập huấn luyện được chọn ngẫu nhiên và chuyển cho mỗi cây. Nó thường có giá trị bằng giá trị của tổng số hàng trong dữ liệu huấn luyện nhưng có thể dùng giá trị bé hơn để tăng hiệu suất và giảm tương quan của cây trong một số trường hợp
6. depth: Độ sâu của từng cây quyết định, độ sao lớn nghĩa là số lần phân nhánh nhiều hơn, tăng xu hướng quá khớp. Nhưng vì chúng ta tập hợp nhiều cây không tương quan nên các cây bị quá khớp hầu như không làm ảnh hưởng đến toàn bộ rừng
7. min_leaf: Số lượng hàng dữ liệu tối thiểu cần thiết trong một nút để phân nhánh.

## 1. Lớp RandomForest

```
class RandomForest():
    def __init__(self, x, y, n_trees, n_features, sample_sz, depth = 10, min_leaf=5):
        np.random.seed(12)
        if n_features == 'sqrt':
            self.n_features = int(np.sqrt(x.shape[1]))
        elif n_features == 'log2':
            self.n_features = int(np.log2(x.shape[1]))
        else:
            self.n_features = n_features
        print(self.n_features, "sha: ",x.shape[1])

        self.x, self.y, self.sample_sz, self.depth, self.min_leaf  = x, y, sample_sz,
        depth, min_leaf

        self.trees = [self.create_tree() for i in range(n_trees)]

    def create_tree(self):
        idxs = np.random.permutation(len(self.y))[:self.sample_sz]
        f_idxs = np.random.permutation(self.x.shape[1])[:self.n_features]

        return DecisionTree(self.x[idxs], self.y[idxs], self.n_features, f_idxs,
                    idxs=np.array(range(self.sample_sz)), depth = self.depth,
                     min_leaf=self.min_leaf)

    def predict(self, x):
        return np.mean([t.predict(x) for t in self.trees], axis = 0)
```

1. `__init__:` hàm khởi tạo với các tham số cần thiết.
2. `create_tree`: hàm tạo một cây quyết định mới bằng cách gọi hàm tạo của lớp DecisionTree(lớp sẽ được giới thiệu ở bên dưới). Mỗi cây nhận được một tập hợp con ngẫu nhiên của các thuộc tính (features bagging) và một tập hợp các hàng dữ liệu ngẫu nhiên.
3. `predict`: hàm dự đoán của khu rừng ngẫu nhiên, trả về giá trị trung bình của tất cả các dự đoán của cây quyết định.

## 2. Lớp DecisionTree

```
def std_agg(cnt, s1, s2): return math.sqrt((s2/cnt) - (s1/cnt)**2)

class DecisionTree():
    def __init__(self, x, y, n_features, f_idxs, idxs, depth = 10, min_leaf = 5):
        self.x, self.y, self.idxs, self.min_leaf, self.f_idxs = x, y, idxs,
             min_leaf, f_idxs
        self.depth = depth
        self.n_features = n_features
        self.n, self.c = len(idxs), x.shape[1]
        self.val = np.mean(y[idxs])
        self.score = float('inf')

        self.find_varsplit()

    def find_varsplit(self):
        for i in self.f_idxs: self.find_better_split(i)

        if self.is_leaf: return

        x = self.split_col
        lhs = np.nonzero(x<=self.split)[0]
        rhs = np.nonzero(x>self.split)[0]

        lf_idxs = np.random.permutation(self.x.shape[1])[:self.n_features]
        rf_idxs = np.random.permutation(self.x.shape[1])[:self.n_features]

        self.lhs = DecisionTree(self.x, self.y, self.n_features, lf_idxs,
             self.idxs[lhs], depth=self.depth-1, min_leaf=self.min_leaf)
        self.rhs = DecisionTree(self.x, self.y, self.n_features, rf_idxs,
             self.idxs[rhs], depth=self.depth-1, min_leaf=self.min_leaf)
```

```
def find_better_split(self, var_idx):
        x, y = self.x[self.idxs,var_idx], self.y[self.idxs]
        sort_idx = np.argsort(x)
        sort_y,sort_x = y[sort_idx], x[sort_idx]
        rhs_cnt,rhs_sum,rhs_sum2 = self.n, sort_y.sum(), (sort_y**2).sum()
        lhs_cnt,lhs_sum,lhs_sum2 = 0,0.,0.

        for i in range(0, self.n - self.min_leaf - 1):
            xi,yi = sort_x[i],sort_y[i]
            lhs_cnt += 1; rhs_cnt -= 1
            lhs_sum += yi; rhs_sum -= yi
            lhs_sum2 += yi**2; rhs_sum2 -= yi**2
            if i<self.min_leaf or xi==sort_x[i+1]:
                continue
            lhs_std = std_agg(lhs_cnt, lhs_sum, lhs_sum2)
            rhs_std = std_agg(rhs_cnt, rhs_sum, rhs_sum2)
            curr_score = lhs_std*lhs_cnt + rhs_std*rhs_cnt
            if curr_score<self.score:
                self.var_idx, self.score, self.split = var_idx,curr_score,xi

    @property
    def split_name(self): return self.x.columns[self.var_idx]

    @property
    def split_col(self): return self.x[self.idxs,self.var_idx]

    @property
    def is_leaf(self): return self.score == float('inf') or self.depth <= 0


    def predict(self, x):
        return np.array([self.predict_row(xi) for xi in x])

    def predict_row(self, xi):
        if self.is_leaf: return self.val
        t = self.lhs if xi[self.var_idx]<=self.split else self.rhs
        return t.predict_row(xi)
```

1. `__init__`: hàm khởi tạo cây quyết định:
   a. `self.val = np.mean(y[idxs])`. Dự đoán của mỗi cây quyết định giá trị trung bình của tất cả các hàng dữ liệu mà nó đang nắm giữ. Biến self.val giữ dự đoán cho mỗi nút của cây. Đối với nút gốc, giá trị sẽ là giá trị trung bình của tất cả các quan sát vì nó chứa tất cả các hàng khi chưa thực hiện phân tách.
   b. `self.score = float('inf')`. Điểm của một nút được tính toán dựa trên cách nó phân chia "tốt" tập dữ liệu ban đầu. Chúng ta sẽ định nghĩa "tốt" ở bên dưới

2. `split_col`: Trả về cột tại `self.var_idx` với càng hàng dữ liệu được cung cấp bởi biến `self.idxs`. Về cơ bản, phân tách một cột/thuộc tính với các hàng dữ liệu đã chọn.
3. `is_leaf`: Hàm để xác định một nút có phải nút lá hay không. Một nút lá là nút không bị phân tách vì thế có `self.score` bằng vô hạn. Ngoài ra, trong tường hợp chúng ta đi đến độ sâu tối đa `self.depth <= 0`, nó cũng là nút lá vì chúng ta không thể đi sâu hơn.
4. `find_better_split`: Hàm tìm cách phân tách tốt nhất cho một cột/thuộc tính nhất định. Chúng ta so sánh các cách phân tách với nhau dựa trên độ "tốt" khi nó phân tách dữ liệu thành hai nửa. Chúng ta sẽ sử dụng độ lệch chuẩn để đo độ "tốt" này. Vì thế chúng ta muốn giảm thiểu trung bình của độ lệch chuẩn hai nửa. Sử dụng cách tiếp cận tham lam, chúng ta sẽ tìm phép tách bằng cách chia dữ liệu thành hai nửa cho mọi giá trị trong cột và tính trung bình độ lệch chuẩn của hai nửa để cuối cùng tìm ra giá trị nhỏ nhất.
   a. Hàm std_agg dùng để tính độ lệch chuẩn
   b. `curr_score = lhs_std*lhs_cnt + rhs_std*rhs_cnt`. Điểm cho mỗi lần lặp là trung bình có trọng số của độ lệch chuẩn của hai nửa với trọng số là số hàng dữ liệu trong mỗi nửa. Điểm thấp hơn tương đương với phương sai thấp hơn.
   c. 
```
if curr_score<self.score:
      self.var_idx, self.score, self.split =
var_idx,curr_score,xi
```
   Khi điểm hiện tại tốt hơn (nhỏ hơn điểm lưu trong `self.score`) chúng ta sẽ cập nhật cột trong `self.var_idx` và lưu giá trị tách trong `self.split`
5. `find_varsplit`:
   a. `for i in self.f_idxs: self.find_better_split(i).` Vòng lặp này đi qua từng cột/thuộc tính và cố gắng tìm cách phân tách tốt nhất cho cột đó. Khi vòng lặp kết thúc, chúng ta đã tìm được cách phân tách tốt nhất. Thuộc tính được dùng để phân tách sẽ được lưu ở biến `self.var_idx` và giá trị của thuộc tính dùng để phân tách được lưu trong biến `self.split`. Từ đó chúng ta sẽ đệ quy để tạo ra 2 nửa, cây quyết định phải và trái cho đến khi chúng ta đạt đến nút lá
   b. `if self.is_leaf: return`. Nếu chúng ta đạt đến nút là, chúng ta sẽ không cần tìm cách phân tách nữa.
   c. `self.lhs`. Giữ cây quyết định bên trái. `lhs` là chỉ số của phần tử hàng dữ liệu mà nó giữ mà có giá trị thuộc tính nhỏ hơn hoặc bằng giá trị dùng để tách `self.split`. Biến `lf_idxs` giữ

chỉ số của các thuộc tính mà cây quyết định bên trái nhận được (features bagging)

d. `self.rhs`. Giữ cây quyết định bên phải. `rhs là` chỉ số của hàng dữ liệu mà cây bên phải giữ. Nó được tính giống `lhs`. Biến `rf_idxs` tương tự `lf_idxs`

# V. Kết luận

Tổng kết lại, bản chất của mô hình rừng ngẫu nhiên là sự kết hợp giữa nhiều mô hình cây quyết định được huấn luyện trên các tập dữ liệu khác nhau được rút ra từ tập huấn luyện. Mô hình rừng cây có ưu điểm đó là giảm thiểu được hiện tượng quá khớp do có phương sai thấp và ít bị ảnh hưởng bởi nhiễu như mô hình cây quyết định. Khi huấn luyện mô hình, mô hình rừng cây cũng giúp chúng ta đánh giá nhanh tầm quan trọng của các biến đối với việc phân loại. Điều này cực kì hữu ích đối với những bộ dữ liệu có số chiều lớn.

Rừng ngẫu nhiên là một trong những kỹ thuật tốt nhất với hiệu suất cao được sử dụng rộng rãi trong các ngành công nghiệp khác nhau vì tính hiệu quả của nó.

# Tài liệu tham khảo

1. Bài giảng Cây quyết định và rừng ngẫu nhiên
2. Wikipedia Cây quyết định
3. Wikipedia Rừng ngẫu nhiên