ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VŨ VĂN LUÂN

RỪNG NGẪU NHIÊN CẢI TIẾN CHO LỰA CHỌN THUỘC TÍNH VÀ PHÂN LOẠI DỮ LIỆU GEN

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

HÀ NỘI, 2017

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VŨ VĂN LUÂN

RỪNG NGẪU NHIÊN CẢI TIẾN CHO LỰA CHỌN THUỘC TÍNH VÀ PHÂN LOẠI DỮ LIỆU GEN

Ngành : Công nghệ thông tin
Chuyên ngành : Kỹ thuật phần mềm
Mã số : 60480103

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC:

TS. Nguyễn Thanh Tùng

HÀ NỘI, 2017

LỜI CAM ĐOAN

Tôi xin cam đoan những kiến thức trình bày trong luận văn này là do tôi tìm hiểu, nghiên cứu và trình bày theo cách hiểu của bản thân dưới sự hướng dẫn trực tiếp của của Tiến sĩ Nguyễn Thanh Tùng.

Tất cả những tham khảo từ các nghiên cứu liên quan đều được nêu nguồn gốc một cách rõ ràng từ danh mục tài liệu tham khảo của luận văn. Trong luận văn, không có việc sao chép tài liệu, công trình nghiên cứu của người khác mà không chỉ rõ về tài liệu tham khảo. Mọi sao chép không hợp lệ, vi phạm quy chế đào tạo tôi xin chịu hoàn toàn trách nhiệm.

TÁC GIẢ LUẬN VĂN

Vũ Văn Luân

LỜI CẢM ƠN

Để hoàn thành được luận văn thạc sỹ này, trước hết tôi xin gửi lời cảm ơn sâu sắc nhất đến TS Nguyễn Thanh Tùng. Thầy đã cung cấp cho tôi những kiến thức, những tài liệu, những phương pháp khi nghiên cứu một vấn đề mang tính khoa học. Thầy thường xuyên đưa ra và giúp tôi có những ý tưởng khi làm luận văn. Tôi xin chân thành cảm ơn thầy về sự hỗ trợ chân thành và nhiệt tình trong suốt thời gian qua. Tôi cũng xin cảm ơn PGS. TS. Hoàng Xuân Huấn, với sự giúp đỡ của Thầy qua những lần thảo luận đã giúp tôi hoàn thành được luận văn đúng hạn.

Tôi xin chân thành cảm ơn các thầy, cô giáo trong Bộ môn Công nghệ phần mềm, Khoa Công nghệ thông tin – Phòng Đào tạo sau đại học – Nghiên cứu Khoa học, Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội đã tạo mọi điều kiện tốt nhất để tôi hoàn thành khóa học này. Đồng thời, tôi cũng xin cảm ơn gia đình, bạn bè, những người luôn khuyến khích và giúp đỡ tôi trong mọi hoàn cảnh khó khăn. Tôi xin cảm ơn cơ quan và các đồng nghiệp đã hết sức tạo điều kiện cho tôi trong suốt thời gian tôi học tập và rèn luyện tại trường Đại học Công nghệ - Đại học Quốc gia Hà Nội.

TÁC GIẢ LUẬN VĂN

Vũ Văn Luân

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Chữ viết tắt	Tiếng Anh	Nghĩa Tiếng Việt
RF	Random Forest	Rừng ngẫu nghiên
RRF	Regularized Random Forest	Rừng ngẫu nhiên có kiểm soát
GRRF	Guided Regularized Random Forests	Rừng ngẫu nhiên điều hướng
SNP	Single Nucleotide Polymorphism	
GWAS	Genome-wide association studies	
KDD	Knowledge Discovery and Data Mining	Phát hiện tri thức và khai phá dữ liệu
SVM	Support Vector Machine	

DANH MỤC CÁC HÌNH VẼ

DANH MỤC CÁC BẢNG

MỞ ĐẦU

Hiện nay, kỹ thuật phân loại dữ liệu được sử dụng rông rãi trong hầu hết các lĩnh vực khác nhau của trí tuệ nhân tạo như phân loại văn bản, phân loại chữ viết tay, phân loại hình ảnh, phân loại gen,…

Mỗi gen đảm nhận một chức năng nào đó và có mối liên hệ với các gen khác. Việc phân loại gen chính là xác định vị trí tương đối của chúng với các gen khác. Bài toán phân loại dữ liệu gen có nhiệm vụ xác định chức năng của gen. Thông thường, mỗi gen mã hóa một protein tương ứng. Các protein này đảm nhiệm những vai trò hay chức năng khác nhau trong cơ thể các sinh vật. Các chức năng của gen/protein rất đa dạng, từ đóng vai trò trong các phản ứng sinh hóa của tế bào, tới tương tác và điều hòa sự hoạt động của các gen khác. Việc xác định chức năng của gen cũng như sản phẩm của gen là nhiệm vụ quan trọng của sinh học phân tử và tin sinh học.

Trong thực tế có rất nhiều phương pháp phân loại dữ liệu, mỗi phương pháp lại có những đặc điểm riêng phù hợp với từng đối tượng dữ liệu cần phân loại. Luận văn này sẽ trình bày về phương pháp rừng ngẫu nhiên để giải quyết bài toán phân loại dữ liệu gen.

Mục đích nghiên cứu

Mục tiêu nghiên cứu của luận văn là tìm hiểu các thuật toán về lựa chọn thuộc tính trong bài toán phân loại như rừng ngẫu nhiên, rừng ngẫu nhiên có kiểm soát, rừng ngẫu nhiên điều hướng. Từ đó đề xuất một phương pháp cải tiến để nâng cao hiệu quả của thuật toán rừng ngẫu nhiên điều hướng.

Đối tượng và phạm vi nghiên cứu

Đối tương nghiên cứu của đề tài là các bộ dữ liệu gen trong đó, số lượng cá thể gồm 50% bệnh nhân mắc bệnh và 50% không mắc bệnh, dùng để đối chứng. Phạm vi nghiên cứu của luận văn tập trung vào mô hình rừng ngẫu nhiên dùng để phân loại dữ liệu gen đã cho. Từ đó đề xuất cải tiến để nâng cao hiệu quả của mô hình rừng ngẫu nhiên trong việc phân loại dữ liệu gen

Phương pháp nghiên cứu

Phương pháp nghiên cứu khi thực hiện luận văn là tìm hiểu từ cơ sở lý thuyết chung về khai phá dữ liệu và lựa chọn thuộc tính, sau đó tìm hiểu về thuật toán rừng ngẫu nhiên và một số cải tiến của nó. Từ đó đề xuất một cải tiến nhằm nâng cao hiệu của thuật toán rừng ngẫu nhiên.

Đóng góp mới của luận văn

Luận văn này đã đề xuất được một cải tiến phương pháp tính độ quan trọng của thuộc tính cho GRRF nhằm nâng cao hiệu quả cho bài toán phân loại dữ liệu gen. Từ đó thực nghiệm trên các bộ dữ liệu gen nhằm chứng minh hiệu quả của cải tiến. Chi tiết kỹ thuật sẽ được trình bày ở các mục tiếp theo.

Ngoài phần kết luận và các phụ lục, phần còn lại của luận văn được chia thành 4 chương chính:

Chương 1: Giới thiệu về khai phá dữ liệu, lựa chọn thuộc tính

Chương 2: Cây quyết định và Rừng ngẫu nhiên

Chương 3: Rừng ngẫu nhiên cải tiến cho bài toán lực chọn thuộc tính trong dữ liệu có số chiều cao.

Chương 4 : Thực nghiệm trên môi trường R và đánh giá kết quả.

CHƯƠNG 1. GIỚI THIỆU VỀ KHAI PHÁ DỮ LIỆU VÀ LỰA CHỌN THUỘC TÍNH

1.1. Khai phá dữ liệu

1.1.1. Tổng quan về khai phá dữ liệu

Trong cuộc sống hiện nay sự phát triển mạnh mẽ của công nghệ thông tin và truyền thông, nhu cầu lưu trữ dữ liệu và trao đổi thông tin trong xã hội ngày càng tăng lên mạnh mẽ. Tuy nhiên, đi cùng với lượng dữ liệu và thông tin ngày càng khổng lồ mà chúng ta có được thì việc biến đổi những dữ liệu thô có sẵn đó thành tri thức trở thành một đòi hỏi tất yếu trong đời sống hàng ngày. Từ nhu cầu thực tế trên, đòi hỏi chúng ta phải tìm kiếm và ứng dụng các kỹ thuật nhằm "khai phá" những thông tin hữu ích, những tri thức có ích từ những nguồn dữ liệu khổng lồ hiện có.

Phát hiện tri thức và khai phá dữ liệu (Knowledge Discovery and Data Mining - KDD) là những công việc liên quan đến việc trích, lọc những thông tin có ích từ các nguồn dữ liệu [4]. Khai phá dữ liệu là một tập các kỹ thuật được sử dụng một cách tự động nhằm khám phá những tri thức có ích ở dạng tiềm năng trong nguồn dữ liệu đã có.

Ở đây chúng ta có thể coi khai phá dữ liệu là cốt lõi của quá trình phát hiện tri thức. Quá trình phát hiện tri thức gồm các bước:

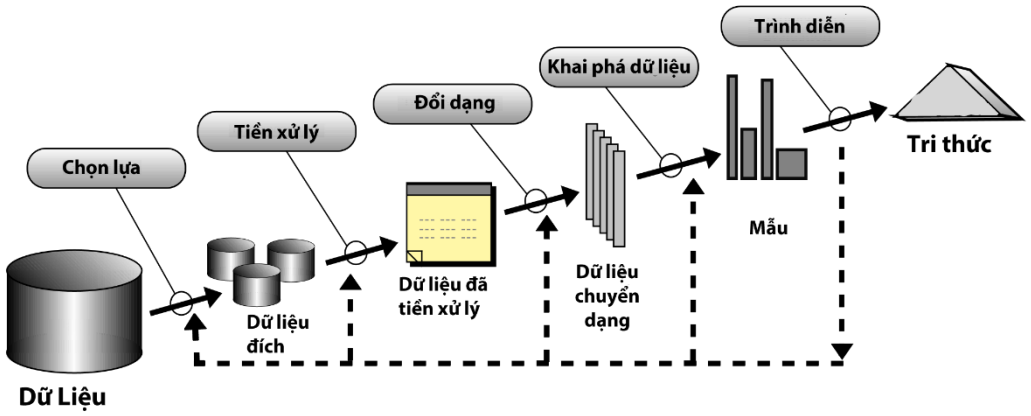
Bước 1: Trích chọn dữ liệu (data selection): Là bước trích chọn những tập dữ liệu cần được khai phá từ các tập dữ liệu lớn (databases, data ware houses).

Bước 2: Tiền xử lý dữ liệu (data preprocessing): Là bước làm sạch dữ liệu (xử lý dữ liệu không đầy đủ, dữ liệu nhiễu, dữ liệu không nhất quán,…v.v), rút gọn dữ liệu (sử dụng các phương pháp thu gọn dữ liệu, histograms, lấy mẫu…v.v), rời rạc hóa dữ liệu (dựa vào histograms, entropy, phân khoảng,...v.v). Sau bước này, dữ liệu sẽ nhất quán, đầy đủ, được rút gọn và được rời rạc hóa.

Bước 3: Biến đổi dữ liệu (data transformation): Là bước chuẩn hóa và làm mịn dữ liệu để đưa dữ liệu về dạng thuận lợi nhất nhằm phục vụ cho các kỹ thuật khai thác ở bước sau.

Bước 4: Khai phá dữ liệu (data mining): Đây là bước quan trọng và tốn nhiều thời gian nhất của quá trình khám phá tri thức, áp dụng các kỹ thuật khai phá (phần lớn là các kỹ thuật của machine learning) để khai phá, chọn lựa được các mẫu (pattern) thông tin, các mối liên hệ đặc biệt trong dữ liệu.

Bước 5: Đánh giá và biểu diễn tri thức (knowledge representation & evaluation): Dùng các kỹ thuật hiển thị dữ liệu để trình bày các mẫu thông tin (tri thức) và mối liên hệ đặc biệt trong dữ liệu đã được khai thác ở bước trên biểu diễn theo dạng gần gũi với người sử dụng như đồ thị, cây, bảng biểu, luật,…v.v. Đồng thời bước này cũng đánh giá những tri thức khám phá được theo những tiêu chí nhất định. Trong giai đoạn khai phá dữ liệu, có thể cần sự tương tác của người dùng để điều chỉnh và rút ra các tri thức cần thiết nhất. Các tri thức nhận được cũng có thể được lưu và sử dụng lại.



Hình 1.1.1: Quá trình phát hiện tri thức

Việc khai phá dữ liệu có thể được tiến hành trên một lượng lớn dữ liệu có trong CSDL, các kho dữ liệu hoặc trong các loại lưu trữ thông tin khác.

Các mẫu đáng quan tâm có thể được đưa đến người dùng hoặc được lưu trữ trong một cơ sở tri thức.

1.1.2. Nhiệm vụ chính của khai phá dữ liệu

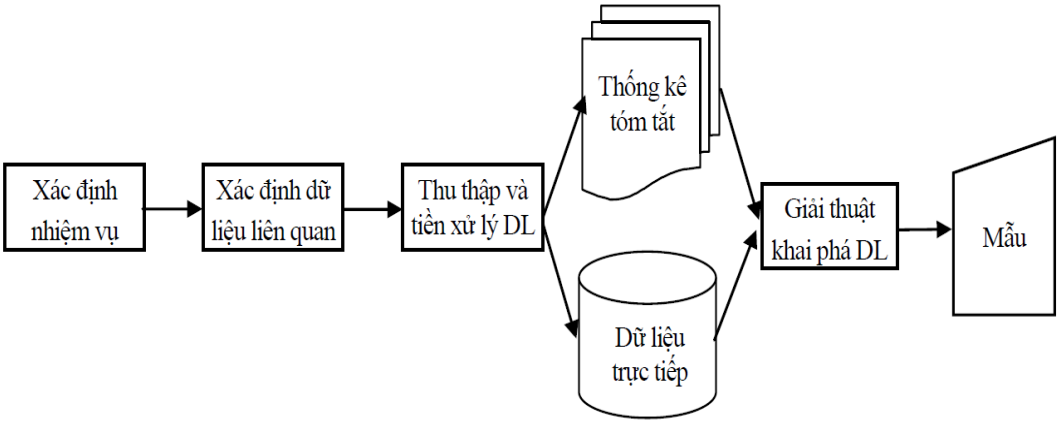
Giảm chiều dữ liệu : Giảm chiều dữ liệu là việc giảm chiều của không gian tìm kiếm dữ liệu, giảm chi phí thu thập và lưu trữ dữ liệu, nâng cao hiệu quả của việc khai phá dữ liệu và làm đơn giản hóa các kết quả khai phá dữ liệu.

Phân nhóm và phân loại : Phân loại và phân nhóm là hai nhiệm vụ có mối quan hệ tương đối gần nhau trong khai phá dữ liệu. Một lớp là một tập các đối tượng có cùng một số đặc điểm hoặc mối quan hệ nào đó, tất cả các đối tượng trong lớp này được phân vào trong cùng một loại tên nhằm mục đích là để phân biệt với các lớp khác. Một cụm là một tập các đối tượng tương tự nhau về mặt vị trí. Các cụm thường được tạo ra nhằm mục đích để sau đó tiến hành phân loại các đối tượng.

Trích chọn luật : Trích chọn luật tìm kiếm và đưa ra dữ liệu bằng cách tất cả các dữ liệu được đưa ra dựa trên các suy diễn/các quyết định mà các suy diễn/quyết định này được xây dựng từ các tri thức thu thập được từ dữ liệu đó. Đối với người sử dụng các kết quả của khai phá dữ liệu họ chỉ mong muốn có một cách giải thích đơn giản là tại sao có các kết quả phân loại đó, thuộc tính nào ảnh hưởng đến kết quả khai phá dữ liệu…Tuy nhiên, bằng các tham số phân loại rất khó để có thể diễn giải các tri thức đó theo cách mà người sử dụng có thể dễ dàng hiểu được.

1.1.3. Quá trình khai phá dữ liệu

Các giải thuật khai phá dữ liệu thường được miêu tả như những chương trình hoạt động trực tiếp trên tệp dữ liệu. Với các phương pháp học máy và thống kê trước đây, thường thì bước đầu tiên là các giải thuật nạp toàn bộ tệp dữ liệu vào trong bộ nhớ. Khi chuyển sang các ứng dụng công nghiệp liên quan đến việc khai phá các kho dữ liệu lớn, mô hình này không thể đáp ứng được. Không chỉ bởi vì nó không thể nạp hết dữ liệu vào trong bộ nhớ mà còn vì khó có thể chiết xuất dữ liệu ra các tệp đơn giản để phân tích được. Quá trình khai phá dữ liệu được thể hiện bởi mô hình sau:



Hình 1.1.2: Quá trình khai phá dữ liệu

+ Xác định nhiệm vụ: Xác định chính xác vấn đề cần giải quyết.

+ Xác định các dữ liệu liên quan dùng để xây dựng giải pháp.

+ Thu thập các dữ liệu có liên quan và xử lý chúng thành dạng sao cho giải thuật khai phá dữ liệu có thể hiểu được. Ở đây có thể gặp một số vấn đề: dữ liệu phải được sao ra nhiều bản (nếu được chiết xuất vào các tệp), quản lý tập các tệp dữ liệu, phải lặp đi lặp lại nhiều lần toàn bộ quá trình (nếu mô hình dữ liệu thay đổi v.v…).

+ Chọn thuật toán khai phá dữ liệu thích hợp và thực hiện việc khai phá dữ liệu: nhằm tìm được các mẫu (pattern) có ý nghĩa dưới dạng biểu diễn tương ứng với các ý nghĩa đó.

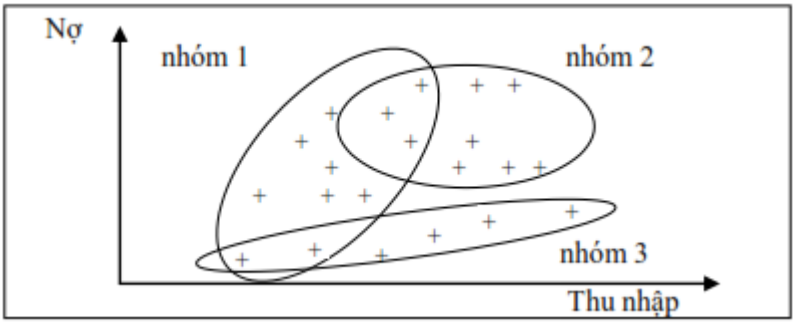
1.2. Một số kỹ thuật khai phá dữ liệu

Mục đích của khai phá dữ liệu là chiết xuất ra các tri thức có lợi cho kinh doanh hay cho nghiên cứu khoa học… Do đó, ta có thể xem mục đích của khai phá dữ liệu sẽ là mô tả các sự kiện và dự đoán. Các mẫu khai phá dữ liệu phát hiện được nhằm vào mục đích này. Dự đoán liên quan đến việc sử dụng các biến hoặc các đối tượng (bản ghi) trong cơ sở dữ liệu để chiết xuất ra các mẫu, dự đoán được những giá trị chưa biết hoặc những giá trị tương lai của các biến đáng quan tâm. Mô tả tập trung vào việc tìm kiếm các mẫu mô tả dữ liệu mà con người có thể hiểu được.

Để đạt được những mục đích này, nhiệm vụ chính của khai phá dữ liệu bao gồm như sau:

1.2.1. Phân nhóm dữ liệu

Phân nhóm là kỹ thuật khai phá dữ liệu. Sự phân nhóm dữ liệu là quá trình lọc không được giám sát, là quá trình nhóm những đối tượng vào trong những lớp tương đương, đến những đối tượng trong một nhóm là tương đương nhau, chúng phải khác với những đối tượng trong những nhóm khác. Trong phân loại dữ liệu, một bản ghi thuộc về lớp nào là phải xác định trước, trong khi phân nhóm không xác định trước. Trong phân nhóm, những đối tượng được nhóm lại cùng nhau dựa vào sự giống nhau của chúng. Sự giống nhau giữa những đối tượng được xác định bởi những chức năng giống nhau. Thông thường những sự giống về định lượng như khoảng cách hoặc độ đo khác được xác định bởi những chuyên gia trong lĩnh vực của mình.

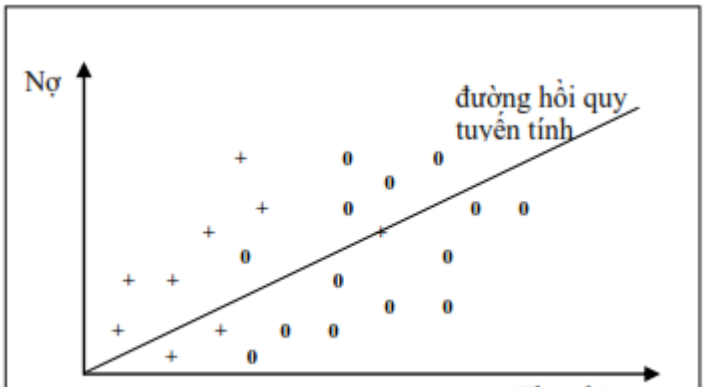


Hình 1.2.1: Mẫu kết quả của nhiệm vụ phân nhóm dữ liệu

Đa số các ứng dụng phân nhóm được sử dụng trong sự phân chia thị trường. Với sự phân nhóm khách hàng vào trong từng nhóm, những doanh nghiệp có thể cung cấp những dịch vụ khác nhau tới nhóm khách hàng một cách thuận lợi. Ví dụ, dựa vào chi tiêu, số tiền trong tài khoản và việc rút tiền của khách hàng, một ngân hàng có thể xếp những khách hàng vào những nhóm khác nhau. Với mỗi nhóm, ngân hàng có thể cho vay những khoản tiền tương ứng cho việc mua nhà, mua xe,… Trong trường hợp này ngân hàng có thể cung cấp những dịch vụ tốt hơn và cũng chắc chắn rằng tất cả các khoản tiền cho vay đều có thể thu hồi được. Ta có thể tham khảo một khảo sát toàn diện về kỹ thuật và thuật toán phân nhóm trong.

Hồi qui (Regression):

Là việc xây dựng mô hình máy tính từ một tập dữ liệu với biến đích có giá trị thực. Bài toán hồi qui tương tự như phân loại, điểm khác nhau là biến đích có dạng số trong khi bài toán phân loại có biến đích kiểu rời rạc. Việc dự báo các



Hình 1.2.2: Mẫu kết quả của nhiệm vụ hồi quy

giá trị số thường được làm bởi các phương pháp thống kê cổ điển chẳng hạn như hồi qui tuyến tính. Tuy nhiên, phương pháp mô hình hóa cũng được sử dụng

Hồi quy được ứng dụng trong nhiều lĩnh vực, ví dụ: dự đoán số lượng sinh vật phát quang hiện thời trong khu rừng bằng cách dò tìm vi sóng bằng thiết bị cảm biến từ xa; dự đoán khả năng tử vong của bệnh nhân khi biết các kết quả xét nghiệm chẩn đoán; dự đoán nhu cầu tiêu thụ một sản phẩm mới bằng một hàm chi tiêu quảng cáo… hình 1.2.2 chỉ ra mẫu kết quả hồi quy tuyến tính đơn giản, ở đây tổng số nợ được điều chỉnh cho phù hợp giống như một hàm thu nhập tuyến tính. Việc điều chỉnh này là không đáng kể bởi vì chỉ tồn tại một tương quan yếu giữa hai biến.

Tổng hợp (summarization):

Là công việc liên quan đến các phương pháp tìm kiếm một mô tả cô đọng cho tập con dữ liệu. Các kỹ thuật tổng hợp thường được áp dụng trong việc phân tích dữ liệu có tính thăm dò và báo cáo tự động.

Mô hình hóa phụ thuộc (dependency modeling):

Là việc tìm kiếm mô tả các phụ thuộc quan trọng giữa các biến. Mô hình phụ thuộc tồn tại hai mức:

+ Mức cấu trúc của mô hình (thường dưới dạng đồ thị) xác định các biến phụ thuộc cục bộ vào các biến khác;

+ Mức định lượng của mô hình xác định mức độ phụ thuộc của biến. Những phụ thuộc này thường được biểu thị dưới dạng luật.

Quan hệ phụ thuộc cũng có thể biểu diễn dưới dạng mạng tin cậy. Đó là đồ thị có hướng không có dạng chu trình, các nút biểu diễn thuộc tính và trọng số chỉ liên kết phụ thuộc giữa các nút đó.

Phát hiện sự thay đổi và độ lệch (change and deviation dectection):

Nhiệm vụ này tập trung vào khám phá những thay đổi có ý nghĩa trong dữ liệu dựa vào các giá trị chuẩn hay độ đo đã biết trước, phát hiện độ lệch đáng kể giữa nội dung của tập con dữ liệu và nội dung mong đợi. Hai mô hình độ lệch thường dùng là lệch theo thời gian và lệch theo nhóm. Độ lệch theo thời gian là sự thay đổi có nghĩa của dữ liệu theo thời gian. Độ lệch theo nhóm là sự khác nhau giữa dữ liệu trong hai tập con dữ liệu, tính cả trường hợp tập con của đối tượng này thuộc tập con kia, nghĩa là xác định dữ liệu trong một nhóm con của đối tượng có khác nhau đáng kể so với toàn bộ đối tượng.

1.2.2. Phân loại dữ liệu

Khái niệm phân loại dữ liệu được Han và Kamber đưa ra năm 2000 [20]. Phân loại dữ liệu là xây dựng một mô hình mà có thể phân các đối tượng thành những lớp để dự đoán giá trị bị mất tại một số thuộc tính của dữ liệu hay tiên đoán giá trị của dữ liệu sẽ xuất hiện trong tương lai.

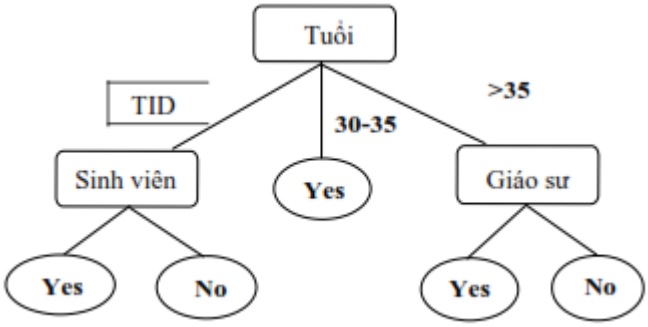
Quá trình phân loại dữ liệu được thực hiện qua hai bước:

Bước thứ nhất: Dựa vào tập hợp dữ liệu huấn luyện, xây dựng một mô hình mô tả những đặc trưng của những lớp dữ liệu hoặc những khái niệm, đây là quá trình học có giám sát, học theo mẫu được cung cấp trước.

Bước thứ hai: Từ những lớp dữ liệu hoặc những khái niệm đã được xác định trước, dự đoán giá trị của những đối tượng quan tâm.

Một kỹ thuật phân loại dữ liệu được Han và Kamber đưa ra là cây quyết định. Mỗi nút của cây đại diện một quyết định dựa vào giá trị thuộc tính tương ứng. Kỹ thuật này đã được nhiều tác giả nghiên cứu và đưa ra nhiều thuật toán.

Một ví dụ tiêu biểu về cây quyết định:



Hình 1.2.3: Ví dụ về cây quyết định

Trong hình 1.2.3 là một cây quyết định cho lớp mua laptop, chỉ ra một khách hàng sẽ mua hay không mua một laptop. Mỗi nút lá đại diện một lớp mà đánh giá mua laptop là Yes hay No. Sau khi mô hình này được xây dựng, chúng ta có thể dự đoán việc có thể mua một laptop hay không dựa vào những thuộc tính khách hàng mới là tuổi và nghề nghiệp. Cây quyết định có thể ứng dụng rộng rãi trong nhiều hoạt động của đời sống thực.

1.3.Lựa chọn thuộc tính

1.3.1. Vai trò của lựa chọn thuộc tính trong khai phá dữ liệu

Hiện nay có rất nhiều phương thức được sử dụng trong khai phá dữ liệu như phân loại, ước lượng, phân nhóm khái niệm và số, mối quan hệ, phân tích tuần tự …Tuy nhiên, đại bộ phận các phương thức này đều gặp vấn đề khi dữ liệu quá lớn hoặc quá nhỏ. Dữ liệu sẽ không là quá nhiều khi nó được thu thập cho một mục đích nhất định. Tuy nhiên, trên thực tế điều này rất ít khi có được bởi vì thông thường các dữ liệu thường được thu thập cho một mục đích chung nào đó hơn là được thu thập cho một mục đích riêng biệt cụ thể. Do vậy, đối với một nhiệm vụ khai phá dữ liệu cụ thể, nếu số lượng các dữ liệu không phù hợp là lớn thì dữ liệu này sẽ ảnh hưởng rất lớn đến kết quả cũng như tốc độ thực hiện của việc khai phá dữ liệu trên. Đối với những trường hợp này, thì việc áp dụng một thuật toán chọn lựa thuộc tính để loại bỏ các dữ liệu không phù hợp là hết sức cần thiết.

Chọn lựa thuộc tính là một quá trình để tìm ra một tập con tốt nhất các thuộc tính theo một số tiêu chí nào đó. Ứng dụng của trích chọn thuộc tính là hữu ích đối với cả ba bước của chu trình phát hiện tri thức: tiền sử lý, khai phá và hậu sử lý. Chọn lựa thuộc tính đóng vai trò như là một công cụ quan trọng trong khai phá dữ liệu và là thể hiện của khoảng cách giữa sự xuất hiện các vấn đề mới và sự xuất hiện các lĩnh vực mới. Một số vấn đề mà trước đây không thể giải quyết được trong thống kê nay đã giải quyết được bởi học máy, và khai phá dữ liệu xuất hiện như một lĩnh vực mới chỉ quan tâm đến việc xử lý các vấn đề chưa xử lý được trong thống kê và học máy. Các công cụ mới được phát triển ngày càng nhiều, tuy nhiên mỗi công cụ lại có những hạn chế rất định do đó chúng ta mong muốn có thể "nối" được những công cụ này để tận dụng được "sức mạnh" của chúng. Chọn lựa thuộc tính có thể được xem như là một bộ công cụ đặc biệt để "nối" và cải thiện hiệu năng của các công cụ hiện có. Xem xét hai thành phần chính của phát hiện tri thức là các giải thuật khai phá và tập dữ liệu, các công cụ có rất nhiều cho cả hai thành phần này. Chọn lựa thuộc tính là các công cụ thao tác dữ liệu làm cho dữ liệu phù hợp với giải thuật khai phá mà không thay đổi bản chất của dữ liệu và là một công cụ hữu hiệu cho việc cải tiến các phương thức khai phá dữ liệu.

1.3.2. Chọn lựa thuộc tính trong bài toán phân loại

Nhiệm vụ cơ bản của việc phân loại là phân chia một tập các đối tượng thành n hữu hạn lớp đã biết trước. Tập đối tượng cần phân loại được đặc trưng bởi một tập các thuộc tính chứa các thông tin cần thiết liên quan đến các lớp, trong đó mỗi tập các thuộc tính được đại diện bởi một tập các thuộc tính – giá trị. Với một tập dữ liệu bao gồm một tập các đối tượng đã được phân loại (thường gọi là tập tập huấn) nhiệm vụ đặt ra là từ tập huấn luyện cho trước xây dựng một bộ phân loại cho các dữ liệu tương tự.

CHƯƠNG 2. CÂY QUYẾT ĐỊNH VÀ RỪNG NGẪU NHIÊN

2.1.Khái niệm chung

2.1.1. Phân loại và dự đoán

Kho dữ liệu luôn chứa rất nhiều các thông tin hữu ích có thể dùng cho việc ra các quyết định liên quan đến điều hành, định hướng của một đơn vị, tổ chức. Phân loại và dự đoán là hai dạng của quá trình phân tích dữ liệu được sử dụng để trích rút các mô hình biểu diễn các lớp dữ liệu quan trọng hoặc dự đoán các dữ liệu phát sinh trong tương lai. Kỹ thuật phân tích này giúp cho chúng ta hiểu kỹ hơn về các kho dữ liệu lớn. Ví dụ chúng ta có thể xây dựng một mô hình phân loại để xác định một giao dịch cho vay của ngân hàng là an toàn hay có rủi ro hoặc xây dựng mô hình dự đoán để phán đoán khả năng chi tiêu của các khách hàng tiềm năm dựa trên các thông tin liên quan đến thu nhập của họ. Rất nhiều các phương pháp phân loại và dự đoán được nghiên cứu trong các lĩnh vực máy học, nhận dạng mẫu và thống kê. Hầu hết các thuật toán đều có hạn chế về bộ nhớ với các giả định là kích thước dữ liệu đủ nhỏ. Kỹ thuật khai phá dữ liệu gần đây đã được phát triển để xây dựng các phương pháp phân loại và dự đoán phù hợp hơn với nguồn dữ liệu có kích thước lớn.

2.1.1.1. Phân loại

Quá trình phân loại thực hiện nhiệm vụ xây dựng mô hình các công cụ phân loại giúp cho việc gán nhãn phân loại cho các dữ liệu. Ví dụ nhãn "An toàn" hoặc "Rủi ro" cho các yêu cầu vay vốn; "Có" hoặc "Không" cho các thông tin thị trường… Các nhãn dùng phân loại được biểu diễn bằng các giá trị rời rạc trong đó việc sắp xếp chùng là không có ý nghĩa.

Phân loại dữ liệu gồm hai quá trình. Trong quá trình thứ nhất một công cụ phân loại sẽ được xây dựng để xem xét nguồn dữ liệu. Đây là quá trình học, trong đó một thuật toán phân loại được xây dựng bằng cách phân tích hoặc "học" từ tập dữ liệu huấn luyện được xây dựng sẵn bao gồm nhiều bộ dữ liệu. Một bộ dữ liệu X biểu diễn bằng một vector p chiều, $X = (x_1, x_1, \dots, x_p)$, đây là các giá trị cụ thể của một tập p thuộc tính của nguồn dữ liệu $\{A_1, A_1, \dots, A_p\}$. Mỗi bộ được giả sử rằng nó thuộc về một lớp được định nghĩa trước với các nhãn xác định

Quá trình đầu tiên của phân loại có thể được xem như việc xác định ánh xạ hoặc hàm y = f(X), hàm này có thể dự đoán nhãn y cho bộ X. Nghĩa là với mỗi lớp dữ liệu chúng ta cần học (xây dựng) một ánh xạ hoặc một hàm tương ứng

Trong bước thứ hai, mô hình thu được sẽ được sử dụng để phân loại. Để đảm bảo tính khách quan nên áp dụng mô hình này trên một tập kiểm thử hơn là làm trên tập dữ liệu huấn luyện ban đầu. Tính chính xác của mô hình phân loại trên tập dữ liệu kiểm thử là số phần trăm các bộ dữ liệu kiểm tra được đánh nhãn đúng bằng cách so sánh chúng với các mẫu trong bộ dữ liệu huấn luyện. Nếu như độ chính xác của mô hình dự đoán là chấp nhận được thì chúng ta có thể sử dụng nó cho các bộ dữ liệu với thông tin nhãn phân loại chưa xác định.

2.1.1.2. Dự đoán

Dự đoán dữ liệu là một quá trình gồm hai bước, nó gần giống với quá trình phân loại. Tuy nhiên để dự đoán, chúng ta bỏ qua khái niệm nhãn phân loại bởi vì các giá trị được dự đoán là liên tục (được sắp xếp) hơn là các giá trị phân loại. Ví dụ thay vì phân loại xem một khoản vay có là an toàn hay rủi do thì chúng ta sẽ dự đoán xem tổng số tiền cho vay của một khoản vay là bao nhiêu thì khoản vay đó là an toàn.

Có thể xem xét việc dự đoán cũng là một hàm y = f(X), trong đó X là dữ liệu đầu vào, và đầu ra là một giá trị y liên tục hoặc sắp xếp được. Việc dự đoán và phân loại có một vài điểm khác nhau khi sử dụng các phương pháp xây dựng mô hình. Giống với phân loại, tập dữ liệu huấn luyện sử dụng để xây dựng mô hình dự đoán không được dùng để đánh giá tính chính xác. Tính chính xác của mô hình dự đoán được đánh giá dựa trên việc tính độ lệch giá các giá trị dự đoán với các giá trị thực sự nhận được của mỗi bộ kiểm tra X.

2.1.2. Cây quyết định

Cây quyết định là một kiểu mô hình dự báo (predictive model), nghĩa là một ánh xạ từ các quan sát về một sự vật/hiện tượng tới các kết luận về giá trị mục tiêu của sự vật/hiện tượng.

Cây quyết định có cấu trúc hình cây và là một sự tượng trưng của một phương thức quyết định cho việc xác định lớp các sự kiện đã cho. Mỗi nút của cây chỉ ra một tên lớp hoặc một phép thử cụ thể, phép thử này chia không gian

các dữ liệu tại nút đó thành các kết quả có thể đạt được của phép thử. Mỗi tập con được chia ra là không gian con của các dữ liệu được tương ứng với vấn đề con của sự phân loại. Sự phân chia này thông qua một cây con tương ứng. Quá trình xây dựng cây quyết định có thể xem như là một chiến thuật chia để trị cho sự phân loại đối tượng [4] [5]. Một cây quyết định có thể mô tả bằng các khái niệm nút và đường nối các nút trong cây.

Mỗi nút của cây quyết định có thể là:

- Nút lá (leaf node) hay còn gọi là nút trả lời (answer node), nó biểu thị cho một lớp các trường hợp (bản ghi), nhãn của nó là tên của lớp.

- Nút không phải là lá (non-leaf node) hay còn gọi là nút trong (inner node), nút này xác định một phép thử thuộc tính (attribute test), nhãn của nút này có tên của thuộc tính và sẽ có một nhánh (hay đường đi) nối nút này đến cây con (sub-tree) ứng với mỗi kết quả có thể có của phép thử. Nhãn của nhánh này chính là giá trị của thuộc tính đó. Nút không phải lá nằm trên cùng là nút gốc (root node).

Một cây quyết định sử dụng để phân loại dữ liệu bằng cách bắt đầu đi từ nút gốc của cây và đi xuyên qua cây theo các nhánh cho tới khi gặp nút lá, khi đó ta sẽ được lớp của dữ kiện đang xét.

2.2.Các thuật toán học cây quyết định

2.2.1. Thuật toán CLS

2.2.1.1. Khái niệm

Đây có thể nói là thuật toán được xây dựng dựa trên ý tưởng đầu tiên về cách thức xây dựng một cây quyết định. Thuật toán này được Hunt trình bày trong Concept Learning System (CLS) vào cuối thập niên 50 của thế kỷ trước [6]. Ý tưởng xây dựng cây quyết định này được gọi là thuật toán CLS mà tư tưởng chủ đạo của nó là chia để trị.

2.2.1.2. Mô tả thuật toán

Thuật toán được trình bày như sau:

Bước 1. Tạo một nút T, nút này chứa tất cả các mẫu trong tập huấn luyện.
Bước 2. Nếu tất cả các mẫu trong T đều có giá trị Yes (positive) đối với

thuộc tính quyết định thì gán nhãn cho nút T là Yes và dừng.
Bước 3. Nếu tất cả các mẫu trong T đều có giá trị No (Negative) đối với thuộc tính quyết định thì gán nhãn cho nút T là No và dừng.
Bước 4. Trong trường hợp ngược lại thì:

4.1. Chọn một thuộc tính X có các giá trị là v_1, v_2,…, v_n, làm nhãn cho T
4.2. Chia T thành các tập con T_1, T_2, ….,T_n dựa theo các giá trị của X.
4.3. Tạo n nút con $T_i (i = 1 \dots n)$, với T là cha của chúng.
4.4. Tạo các nhánh nối từ T đến các T_i, các nhánh này có nhãn tương ứng là các giá trị vi của thuộc tính X

Bước 5. Đệ quy cho mỗi nút con T_i

Bảng 2.2.1: Mô tả thuật toán CLS

2.2.1.3. Đánh giá thuật toán

Chúng ta có thể thấy rằng, tại bước 4 của thuật toán CLS, nếu chọn thuộc tính khác nhau sẽ cho chúng ta cây quyết định có hình dáng khác nhau. Điều này có nghĩa là việc chọn thuộc tính sẽ ảnh hưởng đến độ phức tạp của cây và cho ta các kết quả khác nhau. Vấn đề đặt ra là làm thế nào để chọn thuộc tính được coi là tốt nhất ở mỗi lần chọn, để cuối cùng ta có cây tối ưu. Thuật toán sau đây sẽ giải quyết vấn đề đó.

2.2.2. Thuật toán ID3

2.2.2.1. Khái niện

Thuật toán ID3 [7] học cây quyết định cho bài toán học khái niệm bằng cách xây dựng bắt đầu từ gốc và phát triển dần đến các nút lá. Mỗi nút gốc hoặc nút trong biểu thị một thuộc tính kiểm tra, các cạnh biểu thị giá trị kiểm tra của thuộc tính tương ứng. Thuộc tính tốt nhất của tập dữ liệu đào tạo được chọn làm nút gốc theo tiêu chuẩn cực đại lượng thu hoạch thông tin (Information gain). Để định nghĩa thu hoạch thông tin, ta cần đến khái niệm Entropy của một tập.

a) Entropy

Định nghĩa entropy của một tập S được đề cập đến trong lý thuyết thông tin là số lượng mong đợi các bit cần thiết để mã hóa thông tin về lớp của một thành viên rút ra một cách ngẫu nhiên từ tập S. Trong trường hợp tối ưu, mã có độ dài ngắn nhất. Theo lý thuyết thông tin, mã có độ dài tối ưu là mã gán $-\log_2 p$ bits cho thông điệp có xác suất là p.

Trong trường hợp S là tập mẫu, thì thành viên của S là một mẫu, mỗi mẫu thuộc một lớp hay có một giá trị phân loại.

- Entropy có giá trị nằm trong khoảng [0,1],
- Entropy(S) = 0 ⇔ tập mẫu S chỉ toàn mẫu thuộc cùng một loại, hay S là thuần nhất.
- Entropy(S) = 1 ⇔ tập mẫu S có các mẫu thuộc các loại khác nhau với độ pha trộn là cao nhất.
- 0 < Entropy(S) < 1 ⇔ tập mẫu S có số lượng mẫu thuộc các loại khác nhau là không bằng nhau.

Tập S là tập dữ liệu rèn luyện, trong đó thuộc tính phân loại có hai giá trị, giả sử là âm (-) và dương (+)

- p_+ là phần các mẫu dương trong tập S.
- p_- là phần các mẫu âm trong tập S.

Entropy của S là đại lượng xác định bởi:

$$\text{Entropy(S)} = -p_+ log_2\, p_+ - p_- log_2\, p_-$$

Một cách tổng quát hơn, nếu các mẫu của tập S thuộc nhiều hơn hai loại, giả sử là có c giá trị phân loại thì công thức entropy tổng quát là:

$$\text{Entropy(S)} = \sum_{i=1}^{C} - p_i \log_2 p_i$$

b) Thu hoạch thông tin

Thu hoạch thông tin của một thuộc tính A ứng với một tập mẫu S được ký hiệu là Gain(S,A), đo sự giảm kỳ vọng entropy theo thuộc tính này, được xác định như sau: Một cách chính xác hơn, Gain (S,A) của thuộc tính A, trên tập S, được định nghĩa như sau:

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Trong đó, Values(A) là một tập các giá trị mà thuộc tính A có thể nhận, và S_v là tập hợp con của S nhận giá trị v ở thuộc tính này: S_v = {s ∈S A(s)=v}. Số hạng đầu trong vế phải của công thức trên là entropy của toàn bộ tập S còn số hạng thứ hai là giá trị kỳ vọng của entropy sau khi tách S ra bằng thuộc tính A. Chính vì vậy Gain(S,A) là chiết giảm entropy mong đợi dựa trên các giá trị biết được của thuộc tính A.

2.2.2.2. Mô tả thuật toán

Thuật toán ID3 thực hiện như sau. Trước hết, mở một nút gốc cho cây, nếu nhãn của tập mẫu có giá trị như nhau thì nút này được gán nhãn chung này, còn nếu tập thuộc tính là rỗng thì nút được gán nhãn theo đa số và thuật toán kết thúc. Ngược lại, thuộc tính tốt nhất được chọn làm nút gốc của cây và tạo ra các nhánh từ nút gốc tương ứng với các giá trị của thuộc tính được chọn.

Sau đó, tạo ra một nút con cho mỗi nhánh ứng với tập các mẫu huấn luyện sẽ được sắp xếp phù hợp với các nút con, tức là, xếp các mẫu vào nhánh tương ứng với giá trị của mẫu theo thuộc tính này và thuộc tính này cũng bị loại khỏi danh sách tập thuộc tính.

Toàn bộ quá trình trên được lặp lại cho các tập mẫu mới ở các nút con để phát triển cây cho tới khi dữ liệu thuần nhất hoặc các thuộc tính đã kiểm tra hết thì nút trở thành nút lá vá gán nhãn tương ứng hoặc theo đa số. Kết quả cho một cây quyết định phân loại đúng (hoặc gần đúng nếu mẫu có nhiễu) các tập mẫu đào tạo

Bước 0. Khởi tạo: D, tập nhãn, tập thuộc tính;

Bước 1. Tạo nút gốc (Root) cho cây;

Bước 2. Gán nhãn cho nút nếu dữ liệu thuần nhất hoặc tập thuộc tính là rỗng.

2.1. Nếu mọi mẫu đều dương tính thì nhãn nút gốc = +

2.2. Nếu tất cả các mẫu là âm tính thì nhãn nút gốc = -

2.3. Nếu tập các thuộc tính là rỗng trả về cây một nút gốc có nhãn = giá trị phổ biến nhất của thuộc tính đích trong tập các mẫu;

2.4. Các trường hợp khác sang bước 3;

Bước 3.

3.1. Xác định thuộc tính phân loại tập mẫu tốt nhất trong tập thuộc tính;

3.2. A thuộc tính phân loại tốt nhất;

3.4. Với mỗi giá trị có thể vi của thuộc tính A, thực hiện:

3.4.1. Thêm một nhánh mới dưới nút gốc với mỗi điều kiện A=v_i;

3.4.2. Xác định Examples$_{v_i}$= {x tập mẫu: x có giá trị v_i ở thuộc tính A};

3.4.3. Nếu Examples$_{v_i}$ là rỗng thì thêm dưới nhánh một nút lá có nhãn

là nhãn phổ biến nhất của các mẫu trong tập mẫu;

3.4.4. Ngược lại, trở lại bước 1 với khởi tạo:

(D = Examples$_{v_i}$, tập nhãn, tập thuộc tính – {A});

Bảng 2.2.2: Mô tả thuật toán ID3

2.2.2.3. Đánh giá thuật toán

ID3 cho ta một phương pháp hiệu quả để tìm một giả thuyết phù hợp với tập mẫu huấn luyện dưới dạng tập luật được biểu diễn bằng cây quyết định. Các luật phân loại này tiện dùng trong các cơ sở tri thức của các hệ thông minh. Thuật toán có hiệu quả cả khi dữ liệu có nhiễu nhờ quyết định theo đa số khi có xung đột ở nút lá. Tuy nhiên vẫn còn những vấn đề sau cần được khắc phục.

1) ID3 tìm kiếm giả thuyết trên toàn bộ không gian các hàm có giá trị rời rạc nên không gian giả thuyết này luôn chứa hàm đích. Tuy nhiên, khi gặp thuộc tính có giá trị liên tục thì thuật toán không áp dụng được.

2) ID3 thực hiện xây dựng cây từ đơn giản đến phức tạp theo phương thức tìm kiếm leo đồi được định hướng nhờ tiêu chuẩn thuộc tính có Thu hoạch thông

tin cực đại trong không gian giả thuyết. Tiêu chuẩn này chưa chắc đã tốt, Chẳng hạn, đối với dữ liệu nhân sự có thuộc tính ngày sinh khác nhau cho mỗi mẫu thì thuộc tính này thường có lượng Thu hoạch thông tin lớn nhất nên cho ta cây độ sâu bằng 1 và không có ý nghĩa. Có cách gì để khắc phục không?

3) ID3 không bao giờ quay lui trong quá trình tìm kiếm nên nó dễ mắc phải nhược điểm thường gặp của việc tìm kiếm leo đồi là tập trung vào các giải pháp tối ưu địa phương mà không phải là các giải pháp tối ưu toàn cục.

4) ID3 chỉ duy trì một giả thuyết đơn hiện thời khi tìm kiếm trong không gian của các cây quyết định nên nó đánh mất các khả năng mô tả rõ ràng tất cả các giả thuyết phù hợp. Chẳng hạn, nó không có khả năng xác định xem có bao nhiêu cây quyết định có thể lựa chọn phù hợp với dữ liệu huấn luyện hay đưa ra những vấn đề mới đòi hỏi phải giải quyết tối ưu giữa các giả thuyết đang cạnh tranh này.

5) Phương pháp dựa trên thống kê mà ID3 sử dụng để phát triển giả thuyết hiện thời của nó có ưu điểm là dùng thông tin của tất cả các mẫu, nhờ đó mà ít gặp lỗi hơn là các thuật toán học trên từng mẫu riêng lẻ. Vì vậy ID3 có thể được mở rộng một cách dễ dàng để dùng được cho dữ liệu huấn luyện có nhiễu bằng cách chấp nhận các giả thuyết không phù hợp lắm với dữ liệu huấn luyện. Tuy nhiên thuật toán chưa cho cách xử lý dữ liệu bị mất.

6) Chiến lược tìm kiếm của thuật toán ID3 có hai đặc điểm chính: Ưa các cây ngắn hơn là các cây dài, ưu tiên hơn cho các cây quyết định nhỏ gọn hơn so với các quyết định phức tạp; Ưa những cây mà các thuộc tính có lượng thu hoạch thông tin cao nhất nằm ở gần nút gốc nhất. Thuật toán tìm kiếm trên toàn bộ không gian giả thuyết nên cây tìm được có thể rất lớn, khó áp dụng khi tập dữ liệu đào tạo lớn và có nhiều thuộc tính. Một câu hỏi đặt ra là: nếu cây quá lớn thì ta có thể "tỉa" cho nhỏ và làm đơn giản đi ra sao?

7) Khi dữ liệu chứa nhiễu thì có thể xảy ra hiện tượng phù hợp trội (overfitting), tức là cây quyết định tìm được phù hợp tốt trên dữ liệu đào tạo nhưng có thể đoán nhận tồi hơn cây khác đối với mẫu mới. Thuật toán ID3 xử lý theo phương pháp thống kê nhưng ta chưa cho biết cách tránh hiện tượng phù hợp.

2.2.3. Thuật toán C4.5

2.2.3.1. Khái niệm

Trong thực tế, hầu hết các dữ liệu của các ứng dụng không chỉ liên quan đến các thuộc tính có giá trị rời rạc, rõ ràng mà còn liên quan đến các giá trị thuộc tính dạng liên tục, thay đổi theo thời gian… Vì thế đối với thuật toán xây dựng cây quyết định sử dụng ID3 khó có thể giải quyết được các bài toán mà dữ liệu ở dạng này. Đối với các thuộc tính dạng số, liên tục thì chúng ta phải giới hạn bằng các phép toán tách nhị phân khi xây dựng cây quyết định. Các mẫu phải được sắp xếp theo các giá trị thuộc tính và các giá trị Information Gain được tính toán cho mỗi vị trí tách cây có thể. Các phép tách được xác định chính xác giữa hai mẫu với giá trị khác nhau. Để tránh việc sắp xếp lại, tại mỗi nút phải lưu giữ thứ tự sắp xếp với mỗi tập con bởi vì tập con đó theo mỗi thuộc tính số có thể được sử dụng [6].

Thuật toán C4.5 do Quinlan đưa ra năm 1993 nhằm cải tiến thuật toán ID3 [18]. Thuật toán C4.5 sinh ra một cây quyết định phân loại đối với tập dữ liệu đã cho bằng cách phân chia đệ quy dữ liệu. Cây quyết định được triển khai theo chiến lược chiều sâu trước (Depth First).

Thuật toán này xét tất cả các phép thử có thể phân chia tập dữ liệu đã cho bằng cách đệ quy và xét tất cả các phép thử có thể phân chia tập dữ liệu đã cho và chọn ra một phép thử cho độ đo thu được tốt nhất. Độ đo thu được cũng là độ đo sự hiệu quả của một thuộc tính trong thuật toán triển khai cây quyết định. Đối với mỗi thuộc tính có giá trị liên tục thì các phép thử nhị phân bao gồm mọi giá trị riêng biệt của thuộc tính được xét. Để thu thập các giá trị entropy gain của tất cả các phép thử nhị phân có hiệu quả thì tập dữ liệu huấn luyện thuộc về một nút trong phép thử phải được sắp xếp các giá trị của thuộc tính liên tục và các entropy gain của phép tách nhị phân được dựa trên mỗi giá trị riêng biệt trong lần duyệt qua dữ liệu đã sắp xếp [3] [6].

2.2.3.2. Mô tả thuật toán

Thuật toán này do Quinlan [18] đề xuất để khắc phục các nhược điểm của chính của ID3. Thuật toán này thực hiện theo lược đồ của ID3 nhưng có các cải tiến sau:

1) Ngoài việc áp dụng tiêu chuẩn Thu hoạch thông tin cực đại, C4.5 còn đề xuất sử dụng tiêu chuẩn Tỷ lệ thu hoạch thông tin cực đại (Gainratio) để dùng cho các trường hợp mà tiêu chuẩn trước áp dụng không tốt (nhược điểm thứ nhất của ID3).

2) Áp dụng kỹ thuật chặn sớm sự phát triển của cây dựa trên thống kê để tránh phù hợp trội và cây không quá lớn.

3) Đề xuất giải pháp xử lý trường hợp mẫu có thuộc tính thiếu giá trị

4) Đề xuất phương pháp áp dụng cho thuộc tính nhận giá trị liên tục.

2.2.3.3. Đánh giá thuật toán

Giờ chúng ta cùng xem xét nhưng cải tiến đạt được của thuật toán C 4.5 so với thuật toán ID3

1) Chọn độ đo Gain Ratio

Thuật toán ID3 sử dụng độ đo Information Gain để tìm thuộc tính phân loại tốt nhất nhưng xu hướng của Information Gain là ưu tiên chọn thuộc tính có nhiều giá trị làm thuộc tính phân loại. Ta đã thấy rằng khi có một thuộc tính nhận quá nhiều giá trị thì lượng Thu hoạch thông tin của thuộc tính này thường lớn nhưng lại cho ít thông tin khi chọn nó làm nút gốc. Kết quả cho cây quyết định phức tạp hơn, sinh ra nhiều luật hơn. Trong thuật toán C4.5, tác giả Quinlan đã giải quyết vấn đề này bằng cách sử dụng 1 độ đo khác là Gain Ratio, làm giảm ảnh hưởng của các thuộc tính có nhiều giá trị.

Tỷ lệ thu hoạch này phạt các thuộc tính có nhiều giá trị bằng cách thêm vào vào một hạng tử gọi là thông tin chia (SplitInformation), đại lượng này rất nhạy cảm với việc đánh giá tính rộng và đồng nhất khi chia tách dữ liệu theo giá trị thuộc tính:

$$SplitInformation(S,A) = -\sum_{i=1}^{k} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|} \qquad (2.2.3.1)$$

Trong đó S_1 đến S_k là k tập con của các mẫu huấn luyện khi phân chia S theo k giá trị của thuộc tính A. Lưu ý rằng *SplitInformation(S,A)* thực tế là entropy của *S* ứng với giá trị của thuộc tính A. Khi đó *Tỷ lệ thu hoạch thông tin* của tập *S* đối với thuộc tính *A* được xác định như sau:

$$\text{GainRatio (S,A)}=\frac{\text{Gain(S,A)}}{\text{SplitInfor mation (S, A)}} \qquad (2.2.3.2)$$

Khi sử dụng *GainRatio* thay cho *Gain* để lựa chọn các thuộc tính thì nảy sinh vấn đề là mẫu số ở biểu thức (2.2.3.2) có thể bằng 0 hoặc rất nhỏ khi $|S_i| \approx |S|$. Khi áp dụng, người ta thường khắc phục hiện tượng này nhờ kết hợp cả hai tiêu chuẩn: Đầu tiên, tính Gain cho mỗi thuộc tính, sau đó áp dụng GainRatio cho các thuộc tính có giá trị Gain trên trung bình để chọn thuộc tính tốt nhất.

2) Xử lý giá trị thuộc tính bị thiếu của mẫu

Trong ứng dụng, nhiều khi dữ liệu quan sát được của ta có thể có các mẫu bị thiếu giá trị của một số thuộc tính, chẳng hạn, sử dụng bệnh án của bệnh nhân để dự đoán bệnh và phương án điều trị những kết quả xét nghiệm sinh hóa ở một số bệnh nhân không có. Trong trường hợp này, các giá trị có được thường dùng để ước lượng giá trị thuộc tính thiếu theo hai cách:

- **Cách thứ nhất:** Gán cho giá trị bị thiếu của mẫu bằng giá trị phổ biến nhất của các mẫu học tại nút đang xét vá cùng lớp với nó.
- **Cách thứ hai**: Áp dụng cho trường hợp có nhiều mẫu bị thiếu giá trị ở cùng một thuộc tính. Lúc đó người ta gán giá trị ngẫu nhiên cho tập mẫu bị thiếu này với phân bố bằng tần suất xuất hiện của giá trị tương ứng trong tập mẫu tại nút đang xét.

Chẳng hạn, giả sử thuộc tính A giá trị boolean và nếu nút đang xét chứa 70 mẫu đã biết với giá trị thuộc tính này bằng 1 và 30 mẫu đã biết có giá trị thuộc tính này bằng 0. Khi đó ta ước lượng xác suất để A(x)=1 là 0,7 và xác suất để A(x) = 0 là 0,3. Bằng cách này, các mẫu thiếu giá trị thuộc tính này ở nút đang xét được được tạo ra với tỷ lệ khoảng 0,7 xuống nhánh ứng với thuọc tính A=1 còn với tỷ lệ khoảng 0,4 xuống nhánh thuộc tính A=0 của cây và dùng chúng để tính thu hoạch thông tin cho bước phát triển.

3) Xử lý các thuộc tính có kiểu giá trị liên tục

Trong thuật toán ID3 không phân biệt thuộc tính kiểu giá trị liên tục và thuộc tính kiểu giá trị rời rạc, mà chỉ xem thuộc tính kiểu giá trị liên tục như một thuộc tính có nhiều giá trị, và phạm phải khuyết điểm trên là ưu tiên chọn thuộc

tính này làm thuộc tính phân loại. Giả sử thuộc tính A có các giá trị v_1 , v_2,…, v_N thuật toán C4.5 đã giải quyết vấn đề này như sau:

- Trước tiên, sắp xếp các giá trị của thuộc tính A tăng dần mẫu như từ v_1 , v_2,…, v_N

- Chia giá trị của thuộc tính A thành N-1 "ngưỡng", giá trị "$ngưỡng$" = $\frac{v_i + v_{i+1}}{2}$

- Tính Information Gain ứng với N-1 "ngưỡng".

- Chọn "ngưỡng" có Information Gain cực đại làm "ngưỡng" tốt nhất của A, Gain(S, A) là giá trị Gain cực đại của "ngưỡng" chọn để rẽ nhánh.

2.2.4. Kết luận

Cây quyết đinh là một tiếp cận hữu hiệu cho các bài toán phân loại với giá trị thuộc tính rời rạc. Thuật toán ID3 phát triển cây từ gốc xuống lá theo kiểu ăn tham để chọn thuộc tính phát triển và không quay lui nên có một số nhược điểm mà các thuật toán khác tìm cách khắc phục.

Phương pháp học bằng cây quyết định có thể áp dụng hiểu quả cả khi dữ liệu quan sát được bị nhiễu hoặc có mẫu bị thiếu giá trị thuộc tính. Thuật toán C4.5 cải tiến ID3 dùng được cho cả các bài toán dữ liệu có thuộc tính liên tục, đặc biệt, có thể dùng để xây dựng cây hồi quy.

2.3. Thuật toán Rừng ngẫu nhiên (Random Forest)

2.3.1. Khái niệm

2.3.1.1. Phương pháp Bootrap

Phương pháp Bootrap là một phương pháp rất nổi tiếng trong thống kê được giới thiệu bởi Bradley Efron vào năm 1979 [17]. Phương pháp này chủ yếu dùng để ước lượng lỗi chuẩn (standard errors), độ lệch (bias) và tính toán khoảng tin cậy (confidence interval) cho các tham số. Phương pháp này được thực hiện như sau: từ một tập ban đầu lấy ra một mẫu D = (x_1, x_1,… x_N) gồm N thành phần, tính toán các tham số mong muốn. Trong các bước tiếp theo lặp lại b lần việc tạo ra mẫu L_b cũng gồm N phần từ D bằng cách lấy lại mẫu với sự thay thế các thành phần trong mẫu ban đầu sau đó tính toán các tham số mong muốn.

2.3.1.2. Phương pháp Bagging

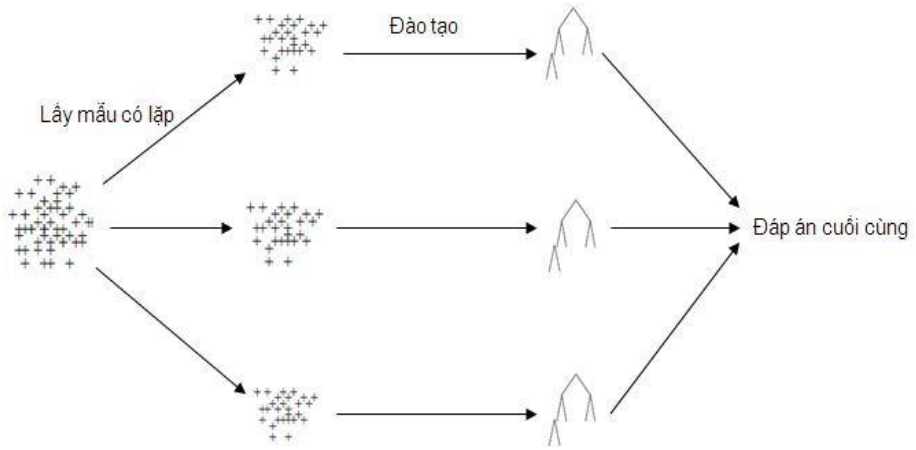
1) Mô hình hoạt động của Bagging

Bagging (Bootstrap aggregating) là tổng hợp các bootstrap sử dụng cách tiếp cận xây dựng mỗi bộ phân loại một cách độc lập với nhau, sau đó sử dụng phương pháp bỏ phiếu để chọn ra kết quả cuối cùng của bộ kết hợp. Tức là mỗi bộ phân loại cơ bản sẽ được xây dựng độc lập với các bộ phân loại khác bằng cách thay đổi tập dữ liệu huấn luyện đầu vào, thay đổi các đặc trưng trong tập huấn luyện. Bagging tạo ra các bộ phân loại từ các tập mẫu con có lặp từ tập mẫu ban đầu (sử dụng bootstrap lấy mẫu có hoàn lại) và một thuật toán học máy, mỗi tập mẫu sẽ tạo ra một bộ phân loại cơ bản.

Các bộ phân loại sẽ được kết hợp bằng phương pháp bỏ phiếu theo số đông. Tức là khi có một mẫu cần được phân loại, mỗi bộ phân loại sẽ cho ra một kết quả. Và kết quả nào xuất hiện nhiều nhất sẽ được lấy làm kết quả của bộ kết hợp

2) Thuật toán Bagging

Bagging tạo ra N tập huấn luyện được chọn có lặp từ tập dữ liệu huấn luyện ban đầu. Trong đó các mẫu huấn luyện có thể được chọn hơn một lần hoặc không được chọn lần nào. Từ mỗi tập huấn luyện mới, Bagging cho chạy với một thuật toán học máy L để sinh ra M bộ phân loại cơ bản h_m. Khi có một mẫu phân loại mới, kết quả của bộ kết hợp sẽ là kết quả nhận được nhiều nhất khi chạy M bộ phân loại cơ bản.



Hình 2.3.1: Mô hình hoạt động của Bagging

Trong hình 2.3.1, bộ 3 mũi tên bên trái mô tả việc lấy mẫu 3 lần có lặp. Bộ 3 mũi tên tiếp theo mô tả việc gọi thuật toán học mô hình trên 3 ví dụ để tạo ra 3 mô hình cơ bản.

Bagging trả lại hàm h(x) được bỏ phiếu lớn nhất trong các $h_1, h_2,..., h_M$. phân loại các mẫu mới bằng việc trả lại lớp y trong tập các lớp có thể Y. Trong hình 2.3.1, có 3 bộ phân loại cơ bản để bỏ phiếu ra đáp án cuối cùng. Trong bagging, các tập huấn luyện M được tạo ra khác nhau. Nếu sự khác nhau này đủ để dẫn đến sự khác nhau của M mô hình cơ bản trong khi hiệu năng của các mô hình đủ tốt thì thì bộ kết hợp có hiệu năng tốt hơn các mô hình cơ bản.

2.3.1.3. Học tập thể

Với mỗi bài toán phân loại hoặc hồi quy cụ thể, người ta thường có nhiều thuật toán học để khi xây dựng bộ học. Cùng một thuật toán, có thể chọn các tham số khác nhau hoặc sử dụng tập dữ liệu huấn luyện khác nhau nên cho các bộ phân loại khác nhau.

Những thuật toán cho cùng lớp bài toán thường tuân theo luật "không có bữa trưa miễn phí (no free lunch theory)", tức là không có thuật toán tốt hơn hẳn các thuật toán khác mà mỗi thuật toán có ưu /nhược điểm riêng, khi thực hiện phân loại thì mỗi bộ huấn luyện theo thuật toán tương ứng có những lớp mẫu được phân loại tốt và tồi khác nhau. Kết hợp hợp lý các bộ phân loại có thể cho ta bộ phân loại mới có nhiều ưu điểm hơn, cách kết hợp này gọi là học máy tập thể (ensemble learning).

Như vậy, mỗi cách học cho ta một bộ phân loại cơ sở, nhờ kết hợp các bộ phân loại thành phần có được mà ta có một bộ phân loại tốt hơn. Các bộ phân loại cơ sở này thường được xây dựng theo cách tiếp cận sau đây:

1) Dùng các thuật toán huấn luyện khác nhau. Các thuật toán này sử dụng các giả thuyết khác nhau về dữ liệu, các bộ học có thể phụ thuộc tham số hoặc không. Khi kết hợp các bộ học, ta được giải phóng khỏi các giả thiết áp đặt này.

2) Mỗi bộ học dùng cách chọn đặc trưng khác nhau. Chẳng hạn chúng ta dùng một thuật toán để phân biệt chữ viết tay nhưng cách chọn đặc trưng có thể là nội dung ảnh hay qua phép biến đổi nào đó.

3) Có thể sử dụng cùng một thuật toán nhưng có tham số khác nhau. Chẳng hạn đều sử dụng thuật toán k-láng giềng gần nhất nhưng với số lượng cây k khác nhau.

4) Cùng một thuật toán nhưng sử dụng các tập dữ liệu huấn luyện khác nhau.

Thông thường thì các bộ phân loại được xây dựng theo hai cách cách tiếp cận đầu có thời gian chạy khác nhau và bộ phân loại chính xác hơn thường đòi hỏi thời gian xử lý nhiều hơn.

Khi có các bộ phân loại cơ sở, bộ phân loại tập thể được kết hợp theo các kiểu tôpô đa dạng để cho ta những bộ mới tốt hơn các bộ thành phần. Trong đó phương thức kết hợp đơn giản và dễ dùng nhất là phương pháp bỏ phiếu.

2.3.1.4. Phương pháp bỏ phiếu

Một cách đơn giản để kết hợp các bộ học cơ sở là dùng phương pháp bỏ phiếu nhờ kiến trúc song song, đầu ra được quyết định nhờ kết quả tổng hợp có trọng số của các bộ phân loại thành phần. Đối với đối tượng x cần gán nhãn, nếu mỗi bộ học cơ sở C_i cho quyết định q_i với trọng số ý kiến w_i tương ứng thì đầu ra của bộ kết hợp đối với mẫu này được tính theo công thức:

$q(x)=\sum_{i=1}^{N} w_1 q_i(x)$ (2.3 a) cho bài toán hồi quy,

và theo đa số có trọng số của tập cho $\{w_1 q_i(x)\}_{i=1}^{N}$ (2.3 b)

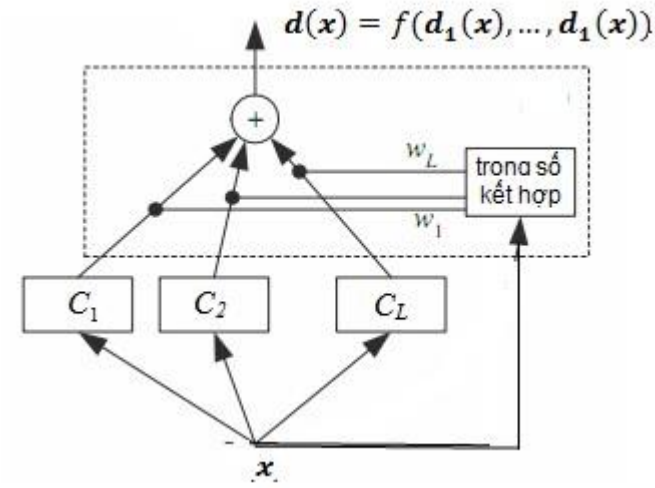
bài toán phân loại,

Trong đó $\sum_{i=1}^{N} w_i = 1$ (N : Số lượng mẫu)

Các trọng số có thể chọn bằng nhau. Tổng quát hơn, ta có thể quyết định bằng một hàm tổng hợp phi tuyến f nào đó:

$$q(x) = f(q_1(x),\ldots,q_1(x))$$

Sơ đồ quyết định tổng quát của quyết định theo hình thức bỏ phiếu được mô tả trong hình 2.3.3.

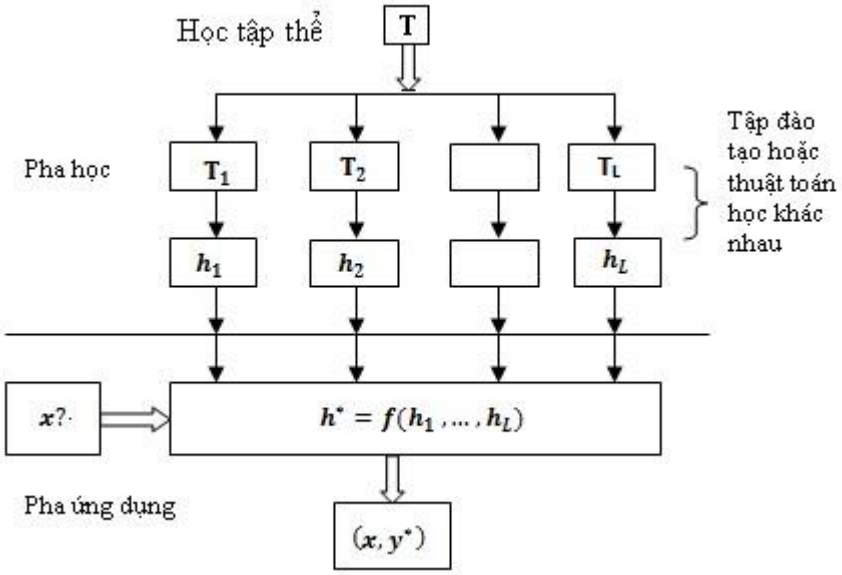


Hình 2.3.2: Sơ đồ kết hợp các bộ phân loại nhờ bỏ phiếu

Việc huấn luyện các bộ thành phần của bộ học tập thể nay có thể sử dụng một trong các phương thức sau:

- N thuật toán huấn luyện khác nhau.
- Một thuật toán nhưng M tập dữ liệu đào tạo hay tham số khác nhau.
- Một thuật toán nhưng dùng tập dữ liệu với tập đặc trưng khác nhau.
- Kết hợp các phương thức trên.

Việc học tập thể T bao gồm các quá trình huấn luyện T_i cho bộ học C_i để cho giả thuyết h_i tương ứng và chúng được kết hợp thành giả thuyết h^*. Khi ứng dụng nhận dạng mẫu x, giả thuyết h* sẽ cho ta nhãn y^* h^* (x) như minh họa trong hình 2.3.5.



Hình 2.3.3: Sơ đồ học tập thể các bộ học

2.3.1.5. Rừng ngẫu nhiên

Random Forest (rừng ngẫu nhiên) [10] là phương pháp học tập thể (ensemble) để phân loại, hồi quy được phát triển bởi Leo Breiman tại đại học California, Berkeley. Breiman cũng đồng thời là đồng tác giả của phương pháp CART [15]. Random Forest (RF) là phương pháp cải tiến của phương pháp tổng hợp bootstrap (bagging). RF sử dụng 2 bước ngẫu nhiên, một là ngẫu nhiên theo mẫu (sample) dùng phương pháp bootstrap có hoàn lại (with replacement), hai là lấy ngẫu nhiên một lượng thuộc tính từ tập thuộc tính ban đầu. Các tập dữ liệu con (sub-dataset) được tạo ra từ 2 lần ngẫu nhiên này có tính đa dạng cao, ít liên quan đến nhau, giúp giảm lỗi phương sai (variance). Các cây CART được xây dựng từ tập các tập dữ liệu con này tạo thành rừng. Khi tổng hợp kết quả, RF dùng phương pháp bỏ phiếu (voting) cho bài toán phân loại và lấy giá trị trung bình (average) cho bài toán hồi quy. Việc kết hợp các mô hình CART này để cho kết quả cuối cùng nên RF được gọi là phương pháp học tập thể.

Đối với bài toán phân loại, cây CART sử dụng công thức Gini như là một hàm điều kiện để tính toán điểm tách nút của cây. Số lượng cây là không hạn chế, các cây trong RF được xây dựng với chiều cao tối đa.

Trong những năm gần đây, RF được sử dụng khá phổ biến bởi những điểm vượt trội của nó so với các thuật toán khác: xử lý được với dữ liệu có số lượng các thuộc tính lớn, có khả năng ước lượng được độ quan trọng của các thuộc tính, thường có độ chính xác cao trong phân loại (hoặc hồi quy), quá trình học nhanh. Trong RF, mỗi cây chỉ chọn một tập nhỏ các thuộc tính trong quá trình xây dựng (bước ngẫu nhiên thứ 2), cơ chế này làm cho RF thực thi với tập dữ liệu có số lượng thuộc tính lớn trong thời gian chấp nhận được khi tính toán. Người dùng có thể đặt mặc định số lượng các thuộc tính để xây dựng cây trong rừng, thông thường giá trị mặc định tối ưu là $\sqrt{p}$ cho bài toán phân loại và $p/3$ với các bài toán hồi quy (p là số lượng tất cả các thuộc tính của tập dữ liệu ban đầu). Số lượng các cây trong rừng cần được đặt đủ lớn để đảm bảo tất cả các thuộc tính đều được sử dụng một số lần. Thông thường là 500 cây cho bài toán phân loại, 1000 cây cho bài toán hồi quy. Do sử dụng phương pháp bootstrap lấy mẫu ngẫu nhiên có hoàn lại nên các tập dữ liệu con có khoảng 2/3 các mẫu không trùng nhau dùng để xây dựng cây, các mẫu ngày được gọi là in-bag. Khoảng 1/3 số mẫu còn lại gọi là out-of-bag, do không tham gia vào việc xây dựng cây nên RF dùng luôn các mẫu out-of-bag này để kiểm thử và tính toán độ quan trọng thuộc tính của các cây CART trong rừng.

2.3.2. Thuật toán Rừng ngẫu nhiên

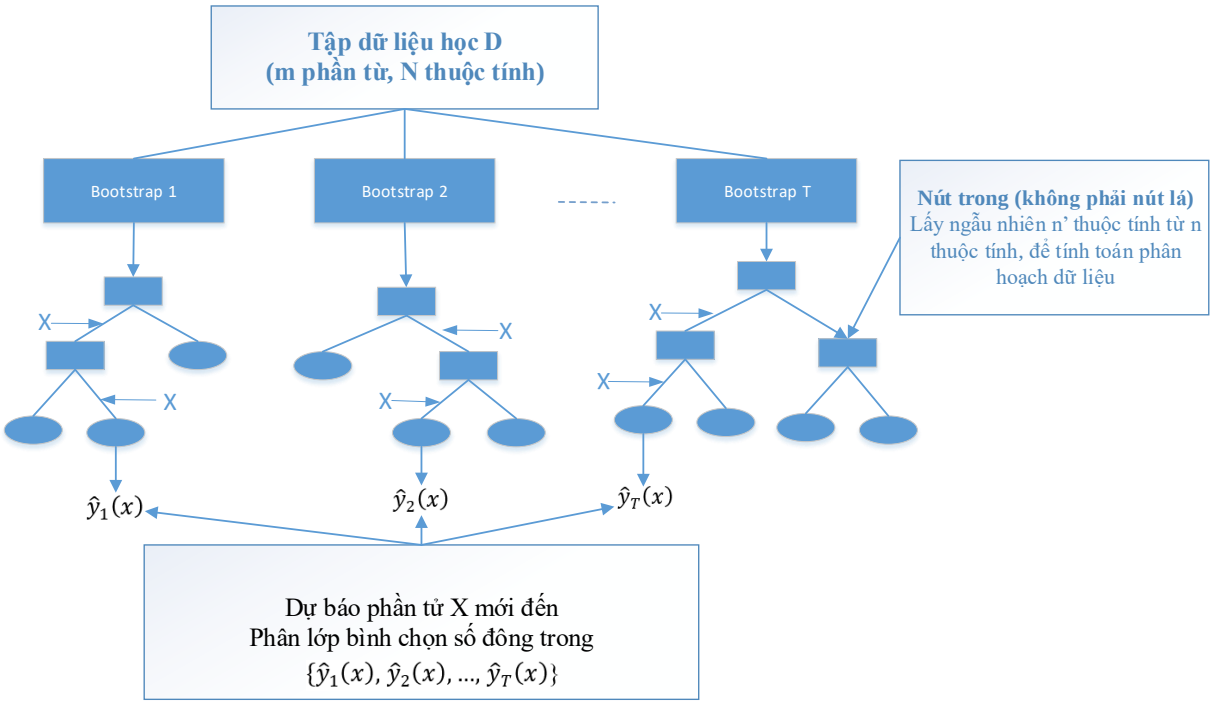
2.3.2.1. Mô tả thuật toán

Tóm tắt thuật toán Random Forest cho phân loại dữ liệu:

Bước 1: Từ tập dữ liệu huấn luyện D, ta tạo dữ liệu ngẫu nhiên (mẫu bootstrap).

Bước 2: Sử dụng các tập con dữ liệu lấy mẫu ngẫu nhiên $D_1, D_2, \ldots, D_k$ xây dựng nên các cây $T_1, T_2, \ldots, T_k$.

Bước 3: Kết hợp các cây: sử dụng chiến lược bình chọn theo số đông với bài toán phân loại hoặc lấy trung bình các giá trị dự đoán từ các cây với bài toán hồi quy.



Hình 2.3.4: Thuật toán Random Forest

Quá trình học của Random Forest bao gồm việc sử dụng ngẫu nhiên giá trị đầu vào, hoặc kết hợp các giá trị đó tại mỗi node trong quá trình dựng từng cây quyết định. Trong đó Random Forest có một số thuộc tính mạnh như :

(1) Độ chính xác của RF tương đối cao.

(2) Thuật toán giải quyết tốt các bài toán có nhiều dữ liệu nhiễu.

(3) Thuật toán chạy nhanh hơn so với bagging.

(4) Có những sự ước lượng nội tại như độ chính xác của mô hình dự đoán hoặc độ mạnh và liên quan giữa các thuộc tính.

(5) Dễ dàng thực hiện song song.

(6) Tuy nhiên để đạt được các tính chất mạnh trên, thời gian thực thi của thuật toán khá lâu và phải sử dụng nhiều tài nguyên của hệ thống.

Tính chất thứ 4 được quan tâm rất nhiều và là tính chất được sử dụng để giải quyết bài toán trích chọn thuộc tính. Sau khi thực hiện học sẽ thu được một danh sách các thuộc được xếp hạng dựa theo một trong hai tiêu chí. Tiêu chí thứ nhất là thu được sau quá trình kiểm tra độ chính xác sử dụng các mẫu out-of-bag. Tiêu chí thứ hai là mức độ dầy đặc tại các node khi phân chia thuộc thuộc tính, và được tính trung bình trên tất cả các cây.

Qua những tìm hiểu trên về giải thuật RF ta có nhận xét rằng RF là một phương pháp phân loại tốt do:

(1) Trong RF các phương sai (variance) được giảm thiểu do kết quả của RF được tổng hợp thông qua nhiều bộ học (learner).

(2) Việc chọn ngẫu nhiên tại mỗi bước trong RF sẽ làm giảm mối tương quan (correlation) giữa các bộ phận lớp trong việc tổng hợp các kết quả.

Ngoài ra, chúng ta cũng thấy rằng lỗi chung của một rừng các cây phân loại phụ thuộc vào lỗi riêng của từng cây trong rừng cũng như mỗi tương quan giữa các cây.

2.3.2.2. Đặc điểm của thuật toán rừng ngẫu nghiên

2.3.2.3. Out – Of – Bag (OOB)

Do sử dụng phương pháp bootstrap lấy mẫu ngẫu nhiên có hoàn lại nên các tập dữ liệu con có khoảng 2/3 các mẫu không trùng nhau dùng để xây dựng cây, các mẫu ngày được gọi là in-bag. Khoảng 1/3 số mẫu còn lại gọi là out-of-bag, do không tham gia vào việc xây dựng cây nên RF dùng luôn các mẫu out-of-bag này để kiểm thử và tính toán độ quan trọng thuộc tính của các cây CART trong rừng cũng như sử dụng để ước lượng lỗi tạo ra từ việc kết hợp các kết quả từ các cây tổng hợp trong random forest.

Trong random forest OOB được tính như sau: Giả sử có một phương pháp cho việc xây dựng một bộ phân loại từ bất kỳ tập huấn luyện nào. Cho một tập huấn luyện D ban đầu, sử dụng phương pháp bootstrap xây dựng được tập huấn luyện D_k, sau đó xây dựng các bộ phân loại h(x, D_k) và sử dụng các bộ phân loại này "bỏ phiếu" để xây dựng một tập tham số dự báo. Đối với mỗi cặp y, x trong tập huấn luyện, việc tổng hợp các lá phiếu chỉ được thực hiện trên những bộ phân loại đối với những tập D_k không chứa y, x. Chúng ta gọi tính toán trên là out-of-

bag classifier. Sử dụng dữ liệu out-of-bag để ước tính tỷ lệ lỗi trong RF là việc tính toán tỉ lệ lỗi của out-of-bag classifier trên tập huấn luyện D_k. Cách tính trên có thể được hiểu một cách đơn giản như sau: Gửi các "đối tượng" trong OOB xuống cây và "đếm" số các dự đoán đúng, ta gọi kết quả của tính toán này là R_{OOB} (Risk out of bag)

2.3.2.4. Độ quan trọng thuộc tính

Theo Breiman [8] có một cách nhìn nữa về rừng ngẫu nhiên: bao gồm một tổ hợp các cây quyết định không cắt nhánh. Mỗi cây quyết định được xây dựng bởi thuật toán CART [8] trên tập mẫu bootstrap (lấy mẫu ngẫu nhiên có hoàn lại) từ tập dữ liệu ban đầu. Tại mỗi nút, một phân hoạch tốt nhất được thực hiện dựa trên thông tin trong một không gian con các thuộc tính được chọn ngẫu nhiên từ không gian thuộc tính ban đầu. RF tổng hợp kết quả dự đoán của các cây quyết định làm kết quả cuối cùng.

Ưu điểm của RF là xây dựng cây không thực hiện việc cắt nhánh từ các tập dữ liệu con khác nhau dùng kỹ thuật boostrap có hoàn lại, do đó thu được những cây với lỗi bias thấp. Bên cạnh đó, mối quan hệ tương quan giữa các cây quyết định cũng được giảm thiểu nhờ việc xây dựng các không gian con thuộc tính một cách ngẫu nhiên. Do đó, việc kết hợp kết quả của một số lượng lớn những cây quyết định độc lập có bias thấp, phương sai cao sẽ giúp RF đạt được cả độ lệch thấp và phương sai thấp. Sự chính xác của RF phụ thuộc vào chất lượng dự đoán của các cây quyết định và mức độ tương quan giữa các cây quyết định.

Cho một tập dữ liệu huấn luyện (tập mẫu) chứa N mẫu dữ liệu, p thuộc tính X_j (j = 1,2,...,p) và $Y \in \{1, 2, \ldots, C\}$ với $C \geq 2$ là biến phụ thuộc. RF dùng chỉ số Gini để đo tính hỗn tạp của tập mẫu. Trong quá trình xây dựng các cây quyết định, RF phát triển các nút con từ một nút cha dựa trên việc đánh giá chỉ số Gini của một không gian con mtry các thuộc tính được chọn ngẫu nhiên từ không gian thuộc tính ban đầu. Thuộc tính được chọn để tách nút t là thuộc tính làm cực tiểu độ hỗn tạp của các tập mẫu sau khi chia. Công thức tính chỉ số Gini cho nút t như sau:

$$\text{Gini}(t) = \sum_{c=1}^{C} \Phi_c(t)[1-\Phi_c(t)] \quad (2.3.1)$$

trong đó $\Phi c(t)$ là tần suất xuất hiện của lớp $c \in C$ trong nút t.

Gọi s là một giá trị trong thuộc tính X_j tách nút t thành 2 nút con: nút trái t_L và nút phải t_R tùy thuộc vào $X_j \leq$ s hoặc X_j>s; t_L ={$X_j \in$t, $X_j \leq$ s} và t_R ={$X_j \in$t, X_j>s}.

Khi đó, tổng độ đo chỉ số Gini của 2 nút t_L và t_R sau khi dùng thuộc tính X_j tách nút t tại s là:

$$\Delta Gini(s,t)=p(t_L)Gini(t_L)+ p(t_R)Gini(t_R) \quad (2.3.2)$$

Để đạt được điểm chia tốt, tại mỗi nút RF sẽ tìm tất cả các giá trị có thể của tất cả mtry biến để tìm ra điểm s có độ đo $\Delta Gini(s,t)$ nhỏ nhất làm điểm phân tách nút t. Thuộc tính chứa điểm phân tách nút t được gọi là thuộc tính tách nút t.

Gọi $IS_k(X_j)$, IS_{X_j} lần lượt là độ đo sự quan trọng của thuộc tính X_j trong một cây quyết định T_k ($k = 1 \ldots K$) và trong một rừng ngẫu nhiên. Công thức tính $IS_k(X_j)$ và IS_{X_j} như sau:

$$IS_k(X_j)=\sum_{t \in T_k} \Delta Gini(X_j,t) \quad (2.3.3)$$

$$IS_{X_j}=\frac{1}{K}\sum_{k=1}^{K} IS_k(X_j) \quad (2.3.4)$$

Chuẩn hóa min-max để chuyển độ đo sự quan trọng thuộc tính về đoạn [0, 1], theo công thức (2.3.5) :

$$VI_{X_j} = \frac{IS_{X_j} - \min_{j=1}^{M}(IS_{X_j})}{\max_{j=1}^{M}\left(IS_{X_j}\right) - \min_{j=1}^{M}(IS_{X_j})} \quad (2.3.5)$$

Độ đo sự quan trọng của các thuộc tính đã chuẩn hóa theo công thức (2.3.5) được dùng để lựa chọn thuộc tính trong các mô hình cải tiến được đề cập ở chương 3.

CHƯƠNG 3. RỪNG NGẪU NHIÊN CẢI TIẾN CHO BÀI TOÁN LỰA CHỌN THUỘC TÍNH TRONG DỮ LIỆU CÓ SỐ CHIỀU CAO

3.1. Rừng ngẫu nhiên kiểm soát có điều hướng

3.1.1. Rừng ngẫu nhiên có kiểm soát

Năm 2012, Deng và Runger [11] đề xuất mô hình cây có kiểm soát (Regularized Trees) giúp cải thiện việc lựa chọn thuộc tính trên cây quyết định. Mô hình mở rộng cho tập hợp cây và nhóm tác giả đặt là rừng ngẫu nhiên có kiểm soát (Regularized Random Forest- RRF).

Ý tưởng của RRF là hạn chế lựa chọn thuộc tính mới để phân tách nút. Nếu thuộc tính mới X_j có độ quan trọng tương đương với thuộc tính X'_j (X'_j là một thuộc tính đã từng được chọn để phân tách), thì RRF ưu tiên chọn thuộc tính X'_j. Thuộc tính mới X_j chỉ được chọn nếu như nó có chỉ số *Gini* nhỏ hơn tất cả các thuộc tính đã được chọn trong các nút trước (xét trong mô hình rừng).

Để thực hiện ý tưởng trên, RRF gán hệ số phạt λ cho $\Delta Gini\left(X_j, t\right)$ đối với các X_j chưa được chọn chia tách nút khi dựng cây từ dữ liệu huấn luyện. Gọi F là tập các thuộc tính đã được sử dụng ở các lần chia tách trước trong mô hình rừng ngẫu nhiên. Độ đo mới dùng để chọn thuộc tính phân tách nút *t* được tính như sau:

$$\Delta Gini_R\left(X_j, t\right) = \begin{cases} \lambda\, \Delta Gini\left(X_j, t\right) & với\ \ X_j \notin F \\ \Delta Gini\left(X_j, t\right) & với\ X_j \in F \end{cases} \qquad (3.1.1)$$

Trong đó: $\lambda \in [0,1]$ là hệ số phạt. Giá trị λ càng nhỏ thì phạt càng cao. Tại nút gốc của cây đầu tiên *F* được gán giá trị rỗng ($F=\varnothing$). RRF sử dụng chỉ số $\Delta Gini_R\left(X_j, t\right)$ để tách nút. Thuộc tính X_j được thêm vào F nếu như nó có chỉ số $\Delta Gini_R\left(X_j, t\right)$ nhỏ hơn $min(\Delta Gini_R(X_i, t))$ với $X_i \in F$.

Bằng thực nghiệm, Deng và Runger cho thấy tiếp cận RRF làm tăng hiệu năng của RF nguyên bản [8] (do RRF không chỉ so độ quan trọng của một thuộc tính trong cây hiện thời mà so trên tất cả các cây đã được xây dựng trước đó để chọn thuộc tính). Vì vậy, RRF làm giảm bias trong quá trình lựa chọn thuộc tính của RF. Tuy nhiên, tại mỗi nút của cây, RRF đánh giá các thuộc tính dựa trên chỉ số *Gini* được tính toán trong một phần nhỏ của tập dữ liệu huấn luyện nhưng lại

so sánh với tất cả thuộc tính đã được chọn tại các cây trong rừng. Điều đó dẫn đến RRF có thể chọn phải những thuộc tính không tốt để dựng cây.

Năm 2013, Deng và Runger [11] đã thiết lập được giới hạn trên cho số giá trị Gini phân biệt trong bài toán phân loại nhị phân có N mẫu là N(N+2)/4-1. Vì vậy, khi N nhỏ dẫn đến số giá trị Gini phân biệt nhỏ. Với bài toán chiều cao, sẽ có rất nhiều giá trị $Gini(X_j,t)$ giống nhau, nên rất khó để phân biệt thuộc tính nào là quan trọng hơn. Ví dụ, đối với bài toán phân hoạch nhị phân, tại một nút chỉ có 10 mẫu thì sẽ có khoảng 29 giá trị Gini phân biệt nhau. Trong tập dữ liệu huấn luyện, nếu có 10000 thuộc tính thì sẽ có khoảng 1000 - 29 = 971 thuộc tính đạt giá trị *Gini* giống nhau. Nếu những chỉ số *Gini* giống nhau này là những giá trị $Gini_{min}$ thì RRF sẽ chọn ngẫu nhiên một trong số các thuộc tính có chỉ số *Gini* đạt *min* để tách nút t. Như vậy, RRF có thể chọn phải những thuộc tính không hoặc ít liên quan đến biến đích để phân hoạch dữ liệu. Vì vậy, đối với các tập dữ liệu có dung lượng mẫu nhỏ, số chiều rất cao (cao hơn nhiều so với dung lượng mẫu) thì cách trích chọn thuộc tính của RRF cho hiệu quả không cao.

3.1.2. Rừng ngẫu nhiên kiểm soát có điều hướng

Trong phương pháp rừng ngẫu nhiên có kiểm soát, Deng [12] đã thay đổi cách tính độ đo quan trọng của mỗi thuộc tính do đó RRF làm giảm độ lệch (bias) so với RF nguyên bản. Tuy nhiên các chỉ số đo độ quan trọng thuộc tính này được đánh giá dựa trên một phần của dữ liệu huấn luyện tại mỗi nút của cây so với tất cả các thuộc tính đã được chọn để xây dựng cây trong rừng. Mặt khác đối với các tập dữ liệu có số mẫu nhỏ, số chiều lớn thì có rất nhiều các thuộc tính có cùng độ đo. Với N mẫu thì số lượng tối đa các thuộc tính có các chỉ số Gini khác nhau trong bài toán phân loại nhị phân là (N(N+ 2)/4)-1 [12]. Ví dụ ta có 30 mẫu có số chiều là 3.000, như vậy có lớn nhất là 239 thuộc tính có độ đo khác nhau và 3.000-239 = 2.761 thuộc tính cùng độ đo. Chính vì vậy RRF phải chọn ngẫu nhiên một trong các thuộc tính đó để tách nút. Các thuộc tính này có thể là những thuộc tính không tốt (không hoặc ít có liên quan đến biến đích) dẫn đến khả năng dự đoán của rừng RRF không cao.

Xuất phát từ lý do trên, Deng [12] đã đề xuất phương pháp rừng ngẫu nhiều kiểm soát có điều hướng (Guided Regularized Random Forests, GRRF) để khắc phục nhược điểm của RRF. Ở phương pháp GRRF các tác giả tính độ quan trọng thuộc tính dựa trên độ quan trọng thuộc tính được tạo ra bởi RF gốc trên toàn bộ

tập dữ liệu ban đầu (dựa theo độ quan trọng của từng nút được tách khi xây dựng cây). Do vậy chỉ số Gini của các thuộc tính có độ quan trọng khác nhau sẽ có giá trị khác nhau. Khi đó với các bài toán có số mẫu nhỏ, số chiều lớn như dữ liệu gen, GRRF sẽ chọn được các thuộc tính tách nút tốt hơn và kết quả phân loại cũng tốt hơn [12]. Nếu như RRF gán hệ số phạt như nhau cho tất cả các thuộc tính mới thì GRRF sử dụng những thuộc tính có độ quan trọng lớn từ RF truyền thống để "điều hướng" quá trình lựa chọn thuộc tính mới khi phân tách nút trong quá trình dựng cây. Thuộc tính có độ quan trọng cao (importance score) thì được gán giá trị λ cao, ngược lại thuộc tính có độ đo quan trọng thấp được gán giá trị λ thấp. Tiếp cận này sử dụng độ quan trọng thuộc tính được tạo ra bởi RF nguyên bản trên toàn bộ tập dữ liệu ban đầu làm trọng số cho các thuộc tính nên đã cải thiện được chất lượng của chỉ số Gini, các thuộc tính có độ quan trọng khác nhau sẽ có giá trị Gini khác nhau. Điều này giúp GRRF có thể chọn được các thuộc tính phân tách tốt hơn trong bài toán phân tích dữ liệu mẫu nhỏ, số chiều cao, nhiều nhiễu. Thực nghiệm trên các tập dữ liệu gen, Deng và Runger cho thấy GRRF mang lại hiệu quả phân loại tốt hơn khi so sánh với RF, RRF, varSelRF và C4.5 [11].

Nếu như RRF gán hệ số phạt λ bằng nhau cho tất cả các thuộc tính mới, thì GRRF căn cứ độ quan trọng của các thuộc tính dựa trên RF nguyên bản (tính theo công thức (5) từ dữ liệu *out of bag*) để gán hệ số phạt λ_j khác nhau đối với các thuộc tính khác nhau. Thuộc tính có độ quan trọng cao thì gán giá trị λ cao (phạt ít), ngược lại gán giá trị λ thấp (phạt nhiều).

Công thức tính độ quan trọng cho các thuộc tính mới tại nút t trong GRRF như sau:

$$Gain_R(X_j, t) = \begin{cases} \lambda_j Gain(X_j, t) & với\ X_j \notin F \\ Gain(X_j, t) & với\ X_j \in F \end{cases} \qquad (3.1.2)$$

$\lambda_j \epsilon (0,1]$ là hệ số phạt gán cho các X_j *(j=1,2,...,M).* Giá trị λ_j dựa vào độ quan trọng của X_j trong RF:

$$\lambda_j = (1 - \gamma)\lambda_0 + \gamma VI_{X_j} \qquad (3.1.3)$$

Trong đó, $\lambda_0 \ \epsilon(0,1]$ là hệ số điều khiển mức độ điều hướng, $\gamma \ \epsilon[0,1]$ điều chỉnh độ quan trọng của thuộc tính đã chuẩn hóa và được gọi là hệ số quan trọng. Khi $\gamma = 0$ GRRF trở thành RRF.

Để giảm tham số cho GRRF, Deng và George Runger chọn $\lambda_0 = 1$, ta có:

$$\lambda_j = (1 - \gamma) + \gamma VI_{X_j} = 1 - \gamma(1 - VI_{X_j}) \qquad (3.1.4)$$

Như vậy, GRRF đã kế thừa được những ưu điểm RRF và khắc phục được phần nào hạn chế của RRF trong quá trình lựa chọn thuộc tính phân loại tại các nút có dung lượng mẫu nhỏ.

3.2. Cải tiến trọng số thuộc tính cho GRRF

Trong mục này, phương pháp tính độ quan trọng của thuộc tính trình bày trong [16] được áp dụng để tính trọng số từng gen cho GRRF lựa chọn khi dựng cây. Từ tập dữ liệu có M gen ban đầu, ta bổ sung thêm M gen "rác" bằng cách hoán vị các giá trị của từng gen nhằm mục đích phá hủy quan hệ của các biến so với biến đích. Ý tưởng của phương pháp này như sau. Ta muốn kiểm tra độ quan trọng của 1 gen trong M gen ban đầu, ta dùng RF lần lượt tính độ quan trọng của từng gen này với gen "rác", việc này thực hiện với số lần hữu hạn lặp lại sau đó kiểm thử độ quan trọng của gen thật với gen rác bằng một kiểm định thống kê, chẳng hạn t-test. Giá trị p thu được sau kiểm định là dấu hiệu cho thấy độ quan trọng của gen đang xét so với gen rác, giá trị p càng nhỏ chứng tỏ độ quan trọng của gen càng lớn.

Áp dụng để tính độ quan trọng của từng gen để điều hướng cho GRRF, ta thực hiện RF với số lần lặp hữu hạn R để tính độ quan trọng của 2M gen, sau đó ta thực hiện phương pháp kiểm định thống kê độ quan trọng của từng gen so với độ quan trọng của các gen bổ sung. Với những gen có độ quan trọng ngang bằng những gen "rác", ta gán với trọng số bằng 0, ngược lại ta sẽ lấy giá trị p từ kết quả kiểm định thống kê để làm trọng số cho GRRF. Những trọng số này được sử dụng để điều hướng cho GRRF trong quá trình lựa chọn gen khi xây dựng cây phân loại trong GRRF.

Cho một tập huấn luyện D, tập dữ liệu gen được biểu diễn là $S_x = \{X_j; j = 1,2,\dots M\}$. Gen rác được tạo ra từ các gen X_j trong S_x bằng cách hoán đổi ngẫu

nhiên tất cả các giá trị của X_j để được một gen rác A_j tương ứng. Cho $S_A = \{A_j\}_1^M$ dữ liệu gen mở rộng, tập dữ liệu huấn luyện được ký hiệu là $S_{X,A} = \{S_X, S_A\}$.

Chạy R lần mô hình rừng ngẫu nhiên RF được thực hiện trên tập dữ liệu $S_{X,A}$ với số lượng gen gấp hai lần dữ liệu ban đầu. Với mỗi lần chạy r (r = 1÷R), tính độ quan trọng VI_X^r và VI_A^r cho các gen và đặt chúng vào dòng thứ r của ma trận V_{Rx2M} ta có 1 ma trận gồm R hàng và 2M cột chứa giá trị là độ quan trọng của từng gen (bảng 3.2.2).

TT	VI_{X_1}	VI_{X_2}	…	VI_{X_M}	$VI_{A_{M+1}}$	$VI_{A_{M+2}}$	…	$VI_{A_{2M}}$
1	$VI_{x_{1,1}}$	$VI_{x_{1,2}}$	…	$VI_{x_{1,M}}$	$VI_{a_{1,(M+1)}}$	$VI_{a_{1,(M+2)}}$	…	$VI_{a_{1,2M}}$
2	$VI_{x_{2,1}}$	$VI_{x_{2,2}}$	…	$VI_{x_{2,M}}$	$VI_{a_{2,(M+1)}}$	$VI_{a_{2,(M+2)}}$	…	$VI_{a_{2,2M}}$
.	.							.
.	.							.
.	.							.
R	$VI_{x_{R,1}}$	$VI_{x_{R,2}}$	…	$VI_{x_{R,M}}$	$VI_{a_{R,M+1}}$	$VI_{a_{R,M+2}}$	…	$VI_{a_{R,2M}}$

Bảng 3.2.1: Ma trận mô tả độ quan trọng thuộc tính của tất cả các gen thật và gen rác

Ký hiệu độ quan trọng từng gen của tập S_A tại lần lặp thứ r là $VI_{X,A}^r = \{VI_X^r, VI_A^r\}$ ở đây VI_X^r và VI_A^r là độ quan trọng từng gen của S_X và S_A tại lần lặp thứ r. Tiếp tục lặp lại quá trình R lần (r=1..R) để tính R hàng cho ma trận VI_{X_j}={ $VI_{X_j}^t$}$_1^R$ và VI_{A_j}={ $VI_{A_j}^t$}$_1^R$. Nửa bên phải ma trận của bảng 3.2.1 lưu trữ độ quan trọng của các gen rác, xét các cột từ M+1 đến 2M với từng hàng r tương ứng, ta lấy ra từng giá trị lớn nhất để có được 1 dãy VI_A^{max}. Tiến hành kiểm định t-test từng cột VI_{A_j} (j=1..M) đo độ quan trọng của từng gen ban đầu so sánh nó với dãy VI_A^{max} . Đối với mỗi gen X_j, tiến hành tính t-test như sau:

$$t_j = \frac{\overline{VI_{X_j}} - \overline{VI_A^{max}}}{\sqrt{s_1^2/n_1 + s_2^2/n_2}} \qquad (3.3.1)$$

Trong đó s_1^2 và s_2^2 là các ước lượng không chệch của phương sai hai mẫu , $n_1 = n_2 = R$

Để kiểm tra ý nghĩa thống kê, sự phân bố của t_j trong (3.3.1) được tính gần đúng như một phân phối Student thông thường với các bậc tự do *df* được tính như sau :

$$df = \left[\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right]^2 \Big/ \left[\frac{(s_1^2/n_1)^2}{n_1 - 1} + \frac{(s_2^2/n_2)^2}{n_2 - 1}\right] \quad (3.3.2)$$

Tính được t-test và *df*, có thể tính toán giá trị p (p-value) cho từng gen và thực hiện kiểm nghiệm giả thuyết trên $\overline{VI_{X_j}} > \overline{VI_A^{max}}$. Ta có thể xác định được các gen quan trọng từ kiểm định t-test dựa trên giá trị p nhận được.

Giá trị p của một gen thu được từ kiểm định t-test cho thấy tầm quan trọng của gen trong dự đoán biến đích. Giá trị p của một gen càng nhỏ thì mức độ quan trọng của gen tương ứng sẽ càng cao, đóng góp lớn khi dự đoán biến đích. Tính tất cả các giá trị p cho tất cả các gen, sau đó ta đặt 1 ngưỡng để phân loại độ quan trọng của các gen ra 2 mức, quan trọng và không quan trọng, chẳng hạn đặt ngưỡng đó là η, ví dụ η = 0.05. Bất kỳ gen nào có giá trị p lớn hơn η được coi là một gen có mức độ quan trọng kém, trọng số của nó được gán bằng 0. Ngược lại, trọng số được tính bằng công thức sau:

$$\theta_j = \frac{1}{R}\sum_{r=1}^{R} VI_{X_j}^R \quad (3.3.3)$$

Trọng số $\{\theta_1, \theta_2, \dots, \theta_M\}$ được sử dụng cho GRRF điều hướng lựa chọn các gen khi xây dựng cây trong rừng.

Để GRRF lựa chọn được các gen có độ quan trọng cao khi dựng cây, các trọng số mới đã tính bởi công thức (3.3.3) được sử dụng và thay thế cho độ quan trọng thuộc tính từ RF nguyên bản với một gen X_j $(j = 1 \dots M)$. Trong GRRF, hệ số phạt λ được sử dụng để điều hướng cho việc lựa chọn gen khi dựng cây. Với trọng số thu được đã trình bày ở trên, công thức áp dụng bởi GRRF sử dụng trọng số θ_j với gen X_j tại nút t được tính như sau:

$$\Delta R(X_j, t) = \begin{cases} \lambda \, \mathrm{R}\,(X_j, t) & với \;\; X_j \notin F \\ R(X_j, t) & với \; X_j \in F \end{cases} \qquad (3.3.4)$$

Trong đó F là tập hợp các gen đầu vào đã được sử dụng trong rừng ngẫu nhiên và $\lambda \in [0,1]$. Giá trị của λ không giống nhau cho tất cả các gen đầu vào bởi vì nó được khởi tạo dựa trên các trọng số θ_j trong công thức (3.3.3).

CHƯƠNG 4. THỰC NGHIỆM TRÊN MÔI TRƯỜNG R VÀ ĐÁNH GIÁ KẾT QUẢ

4.1. Dữ liệu thực nghiệm

Bảng 4.1.1 gồm 10 bộ dữ liệu gen được dùng trong thực nghiệm để đánh giá hiệu quả của phương pháp GRRF có cải tiến cách tính trọng số, ký hiệu eGRRF, kết quả thực nghiệm tiến hành bởi eGRRF được so sánh với các phương pháp RF, GRRF, và SVM. Thông tin về 10 bộ dữ liệu gen này được mô tả trong Bảng 4.1.1. Trong đó, số lượng cá thể đã được gán nhãn gồm 50% bệnh nhân mắc bệnh và 50% không mắc bệnh, dùng để đối chứng.

Phương pháp kiểm tra chéo 5-fold cũng được sử dụng để đánh giá hiệu quả của các mô hình. Tập dữ liệu ban đầu được chia làm 5 phần kích thước bằng nhau. Tiến hành lặp 5 lần : mỗi lần lấy 1 phần dùng làm dữ liệu thử nghiệm và 4 phần còn lại dùng làm dữ liệu huấn luyện.

TT	Tập dữ liệu	Số lượng gen	Số lượng cá thể
1	Brain_Tumor1	5,921	90
2	Brain_Tumor2	10,386	50
3	DLBCL	5,470	77
4	Prostate_Tumor	10,510	102
5	Tumors.11	12,543	174
6	Tumors.14	15,010	308
7	EMBRYONAL_TUMOURS_C	7,130	60
8	Leukemia1	5,328	72
9	Leukemia2	11,226	72
10	Lung_Cancer	12,601	203

Bảng 4.1.1: Mô tả các tập dữ liệu thực nghiệm

4.2. Kết quả thực nghiệm

Các thực nghiệm được tiến hành trên 10 bộ dữ liệu gen. Trong phần thực nghiệm, 4 mô hình được tiến hành và so sánh kết quả để đánh giá độ chính xác của mô hình cải tiến eGRRF, 3 mô hình so sánh là: mô hìnhRF nguyên bản của Breiman [10], mô hình rừng ngẫu nhiên kiểm soát có điều hướng (GRRF) của Deng và Ranger [12], mô hình máy véc-tơ hỗ trợ (SVM) với nhân tuyến tính. Phương pháp kiểm tra chéo 5-fold cũng được sử dụng để đánh giá hiệu quả của mô hình eGRRF và các mô hình đối chứng trên các tập dữ liệu gen.

Bài toán phân loại dữ liệu Gen được mô tả như sau:

Input: Tập dữ liệu huấn luyện Gen $S_X = \{\{X_j\}_{j=1}^{M}, Y\}$ có N mẫu dữ liệu và M thuộc tính (Gen). Có 2 loại bệnh và không bệnh tương ứng với hai nhãn {0, 1}

Output: Tìm/học một hàm cho thuộc tính bệnh (hàm phân loại) đối với các giá trị của các gen khác.

Độ đo đánh giá hiệu quả của mô hình được tính dựa trên tổng các gen được dự đoán chúng chia cho tổng số gen có trong tập kiểm thử (testing data), giá trị càng gần 1 nghĩa là mô hình có hiệu năng tốt, ngược lại giá trị gần về 0 khi hiệu năng dự đoán của mô hình không tốt.

Trong phần thực nghiệm độ đo độ chính xác thuật toán được tính theo công thức sau:

$$\text{Acc} = \frac{1}{N_t}\sum_{i=1}^{N_t} \text{I}(\text{Q}(x_i,y_i) - \max_{j\neq y_j} \text{Q}(x_i,j) > 0)$$

Trong đó I(.) là hàm dấu hiệu và $\text{Q}(x_i,y_i) = \sum_{K=1}^{K} \text{I}(h_K(x_i)=j)$ là số lượng cây quyết định lựa chon x_i thuộc vào lớp j , N_t là số mẫu trong D_t

Đầu tiên để đánh giá hiệu quả của mô hình eGRRF và các mô hình rừng ngẫu nhiên khác khi số lượng cây trong rừng biến thiên, kích thước không gian con thuộc tính được đặt cố định là *mtry* = $\sqrt{M}$ và thay đổi số lượng cây K={20, 50, 100, 200, 500, 1000}. Với 5 lần kiểm tra chéo được thực hiện với mỗi K

khác nhau, sau đó lấy kết quả trung bình 5 lần chạy để đánh giá độ chính xác của các mô hình, kết quả được liệt kê như sau:

STT	Tập dữ liệu	Phương pháp	K					
			20	**50**	**100**	**200**	**500**	**1000**
1	Brain_Tumor1	eGRRF	0.83	**0.89**	**0.9**	**0.88**	**0.87**	0.87
		GRRF	**0.85**	0.83	0.88	0.86	0.82	**0.88**
		RF	0.85	0.81	0.87	0.82	0.83	0.83
2	Brain_Tumor2	eGRRF	**0.88**	**0.8**	**0.86**	**0.86**	**0.84**	**0.86**
		GRRF	0.74	0.73	0.82	0.76	0.78	0.81
		RF	0.72	0.72	0.76	0.76	0.74	0.79
3	DLBCL	eGRRF	**0.92**	**0.92**	**0.93**	**0.92**	**0.92**	**0.94**
		GRRF	0.86	0.91	**0.93**	0.91	0.88	0.91
		RF	0.90	0.88	0.86	0.89	0.88	0.90
4	Prostate_Tumor	eGRRF	**0.94**	**0.93**	**0.94**	**0.92**	**0.92**	**0.92**
		GRRF	0.88	0.88	0.93	0.91	**0.92**	**0.92**
		RF	0.91	0.88	0.92	0.90	0.90	0.91
5	Tumors.11	eGRRF	**0.86**	**0.91**	**0.93**	**0.93**	**0.92**	**0.91**
		GRRF	0.84	0.89	0.89	0.89	0.89	0.86
		RF	0.87	0.88	0.87	0.87	0.88	0.87
6	Tumors.14	eGRRF	0.48	0.53	0.53	0.55	0.58	0.58
		GRRF	0.56	**0.63**	**0.67**	**0.64**	**0.66**	**0.66**
		RF	**0.63**	0.62	0.64	0.60	**0.66**	0.65
7	EMBRYONAL_ TUMOURS_C	eGRRF	**0.70**	**0.76**	**0.75**	**0.78**	**0.74**	**0.78**
		GRRF	0.58	0.68	0.58	0.67	0.61	0.63
		RF	0.58	0.63	0.62	0.65	0.60	0.68
8	Leukemia1	eGRRF	0.94	**0.99**	**0.97**	**0.97**	**0.96**	**0.97**
		GRRF	**0.95**	0.93	**0.97**	**0.97**	**0.96**	0.94
		RF	0.93	0.93	0.96	0.93	0.96	0.94
9	Leukemia2	eGRRF	**0.94**	**0.96**	**0.96**	**0.95**	0.96	0.95
		GRRF	0.90	0.94	0.93	0.91	0.96	**0.97**
		RF	0.93	0.94	**0.96**	0.92	**0.97**	**0.97**
10	Lung_Cancer	eGRRF	**0.94**	**0.94**	**0.94**	**0.95**	**0.95**	**0.95**
		GRRF	0.91	**0.94**	**0.94**	0.93	0.93	0.92
		RF	0.91	0.93	0.91	0.92	0.91	0.92

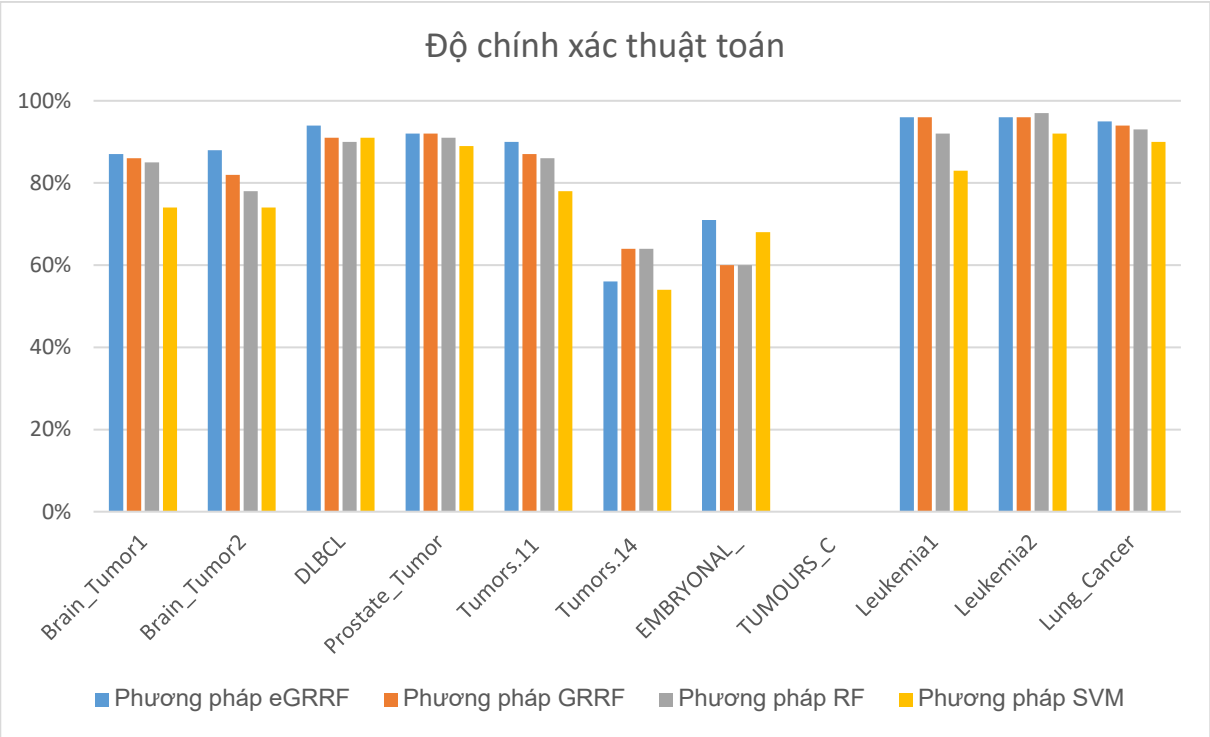
Bảng 4.2.1: So sánh các phương pháp với số lượng cây K thay đổi. Các giá trị có font đậm là kết quả tốt nhất của mô hình.

Trong bảng 4.2.1 so sánh các phương pháp RF, GRRF, eGRRF ta thấy với số lượng cây thay đổi trong hầu hết các trường hợp mô hình eGRRF cho độ chính xác cao hơn so với các phương pháp khác, chẳng hạn với bộ dữ liệu Leukemia2 khi số lượng cây thay đổi thì độ chính xác của thuật toán vẫn đạt được độ chính xác từ 95-96%, và với bộ dữ liệu Lung_Cancer đạt được từ 94-95% tương ứng.

Bảng 4.2.2 liệt kê kết quả phân loại gen của 4 mô hình với các tham số đầu vào cố định (tối ưu cho mô hình), 2 cột cuối của bảng trình bày số lượng gen trung bình được chọn bởi eGRRF và GRRF. Các gen được chọn được xem là các gen có độ quan trọng cao hơn các gen còn lại khi tham gia xây dựng mô hình rừng ngẫu nhiên. Các gen được chọn là kết quả quan trọng cho bài toán lựa chọn gen, mô hình nào chọn được số lượng gen ít nhưng có độ chính xác phân loại gen cao thì đó là mô hình tốt. Trong phần thực nghiệm này, các tham số tối ưu $mTry = \sqrt{M}$ và số cây trong rừng K=500 được dặt giá trị cố định khi thực hiện các mô hình rừng ngẫu nhiên (eGRRF, GRRF, RF), giá trị $C = 2^{-5}$ đặt cố định cho mô hình SVM tuyến tính. Tương tự, phương pháp kiểm tra chéo được thực hiện 5 lần rồi lấy kết quả trung bình để đánh giá độ chính xác của các mô hình.

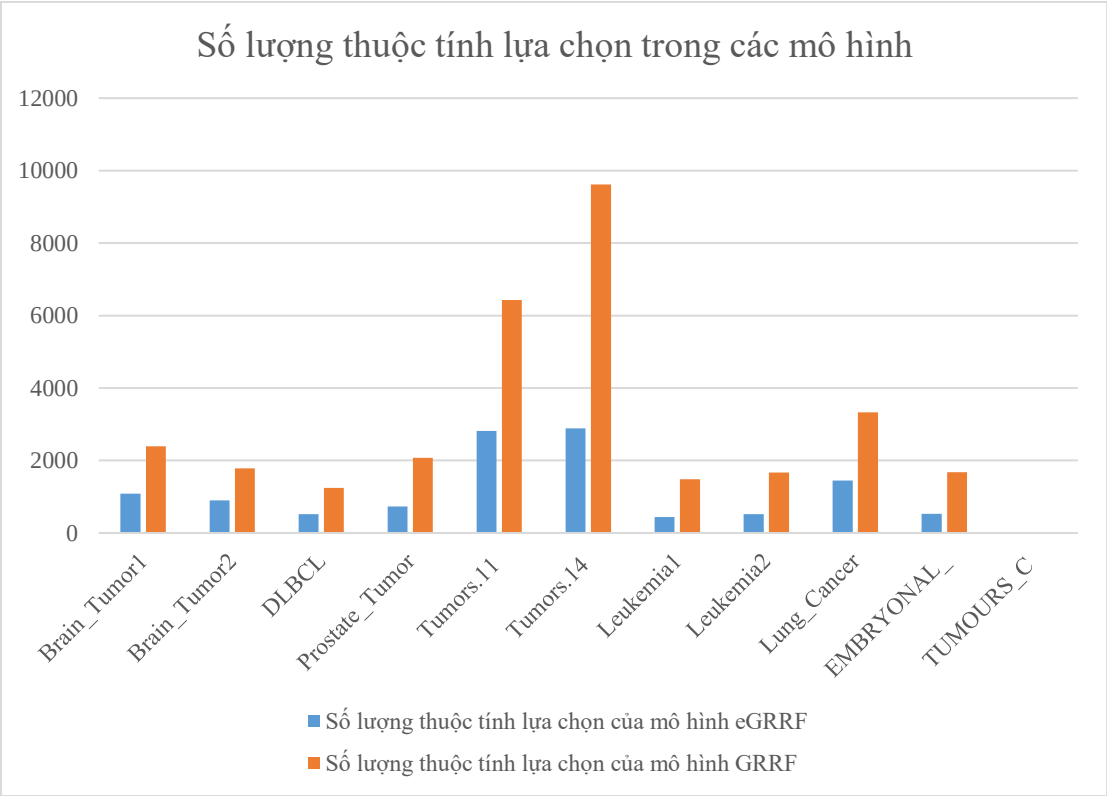
STT	Tập dữ liệu	Phương pháp				Số lượng thuộc tính lựa chọn	
		eGRRF	GRRF	RF	SVM	FS.eGRRF	FS.GRRF
1	Brain_Tumor1	**0.87**	0.86	0.85	0.74	1084.6	2393.8
2	Brain_Tumor2	**0.88**	0.82	0.78	0.74	896.6	1782
3	DLBCL	**0.94**	0.91	0.90	0.91	520.8	1243
4	Prostate_Tumor	**0.92**	**0.92**	0.91	0.89	729.6	2077.2
5	Tumors.11	**0.90**	0.87	0.86	0.78	2819.8	6431
6	Tumors.14	0.56	**0.64**	**0.64**	0.54	2886.6	9620.6
7	EMBRYONAL_ TUMOURS_C	**0.71**	0.60	0.60	0.68	532.6	1673.8
8	Leukemia1	**0.96**	**0.96**	0.92	0.83	437.4	1482.8
9	Leukemia2	**0.96**	**0.96**	0.97	0.92	524.4	1670.4
10	Lung_Cancer	**0.95**	0.94	0.93	0.90	1446	3327.8

Bảng 4.2.2: So sánh các mô hình với tham số cố định tối ưu mTry= $\sqrt{M}$, K=500



Hình 4.2.1: Biểu đồ so sánh độ chính xác của các thuật toán

Trong bảng 4.2.2 và hình 4.2.1 ta thấy với các tham số tối ưu cho từng mô hình thì với mô hình eGRRF vẫn cho một giá trị dự đoán chính xác cao hơn so với phương pháp RF, GRRF và cả SVM. Như với bộ dữ liệu Leukemia1 và Leukemia2 với mô hình eGRRF thì kết quả dự đoán chính xác đến 96%. Điều đó cho thấy eGRRF sử dụng những thuộc tính có độ quan trọng lớn từ RF truyền thống để "hướng dẫn" quá trình lựa chọn thuộc tính mới phân tách nút làm giảm số chiều cho các tập gen dẫn đến làm tăng hiệu quả phân loại trên 10 bộ dữ liệu gen. Cột FS.eGRRF liệt kê số lượng gen được chọn để xây dựng mô hình eGRRF và cột FS.GRRF thống kê số lượng gen của GRRF chọn được sau 5 lần chạy theo phương pháp 5-fold. Ta có thể thấy, số lượng gen mà eGRRF chọn được ít hơn nhiều so với GRRF trên tất cả 10 bộ dữ liệu nhưng kết quả phân loại vẫn có độ chính xác cao hơn, kết quả được minh họa rõ hơn ở hình 4.2.2. Mô hình eGRRF đạt được kết quả phân loại tốt chứng tỏ rằng phương pháp tạo trọng số mới cho các gen đã trình bày ở trên cải thiện rõ rệt cho bài toán phân loại và lựa chọn gen, đặc biệt là kiểu dữ liệu luôn gây khó khăn lớn cho các mô hình máy học khi số chiều rất lớn nhưng cỡ mẫu nhỏ.



Hình 4.2.2: So sánh số lượng thuộc tính được lựa chọn trong các mô hình

Như vậy, với những kết quả thực nghiệm ở trên ta thấy mô hình eGRRF cho kết quả dự đoán có độ chính xác cao và khả năng trích chọn gen hiệu quả hơn hẳn RF, GRRF, SVM. Những kết quả này một lần nữa chứng minh bằng thực nghiệm, mô hình eGRRF đã cải thiện đáng kể độ chính xác phân loại so với các mô hình khác là RF, SVM và GRRF. Mô hình rừng ngẫu nhiên eGRRF có cải tiến cách tạo trọng số có thể được xem là mô hình hữu hiệu dùng cho phân tích dữ liệu gen nói chung.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong khuôn khổ của luận văn, cơ sở lý thuyết về học máy và một số thuật toán áp dụng giải bài lựa chọn thuộc tính đã được tìm hiểu. Chúng tôi cũng đã tập trung nghiên cứu về thuật toán Random Forest và các biến thể cải tiến của Random Forest như rừng ngẫu nhiên có kiểm soát RRF, rừng ngẫu nhiên kiểm soát có điều hướng GRRF. Từ những tìm hiểu này này chúng tôi đề xuất hướng cải tiến cách đánh trọng số cho GRRF nhằm tăng hiệu quả của thuật toán phân loại đặc biệt với dữ liệu có số chiều cao. Để chứng minh tính hiệu quả của mô hình cải tiến, thực nghiệm được tiến hành trên 10 bộ dữ liệu gen.

Từ những kết quả thực nghiệm đạt được trên 10 bộ dữ liệu gen thấy rằng độ chính xác của mô hình cải tiến eGRRF tương đối ổn định và đạt hiệu năng cao so với các phương pháp RF, RRF, cũng như phương pháp GRRF. Qua đó, có thể đóng góp thêm một chọn lựa cho các nhà phát triển ứng dụng khi phát triển các ứng dụng liên quan đến phân loại dữ liệu.

Với những đóng góp trong luận văn này, chúng tôi hi vọng đã góp phần giải quyết một phần nhỏ liên quan đến bài toán khai phá dữ liệu nói chung cũng như bài toán phân loại dữ liệu nói riêng. Tôi cũng hi vọng từ các đóng góp của mình có thể xây dựng lên các hệ thống đánh giá và dự đoán áp dụng một cách thiết thực vào đời sống xã hội.

TÀI LIỆU THAM KHẢO

Tài liệu tiếng Việt

[1] Hoàng Xuân Huấn, "Giáo trình học máy", Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội, 2015.

[2] Hoàng Thị Hà , Nguyễn Thanh Tùng, "Cải tiến phương pháp rừng ngẫu nhiên có điều hướng để áp dụng cho dữ liệu SNP", Kỷ yếu Hội nghị Quốc gia lần thứ VIII về Nghiên cứu cơ bản và ứng dụng Công Nghệ thông tin (FAIR); Hà Nội, ngày 9-10/7/2015

Tài liệu tiếng Anh

[3] M. Stratton, "Genome-wide association study of 14 000 cases of seven common diseases and 3000 shared," The Journal of Nature, vol. 447, no. 7145, p. 661–678, 2007.

[4] L. NikhilR.Pal, "Advanced Techniques in Knowledge Discovery and DataMining," *Springer,* 2005.

[5] H. J. a. K. M., Data Mining: Concepts and Techniques, Morgan Kaufman, Academic Press, 2001.

[6] H. T. Bao, Knowledge Discovery and Data Mining Techniques and, http://www.jaist.ac.jp/~bao/.

[7] U. P.E, Article: Incremental induction of Decision Trees, Univerity of Massacuhsetts, 1989.

[8] B. P. Hofer J., Distributed Decision Tree Induction within the Grid Data Mining Framework GridMiner-Core, Institute for Software Science,AUT, March 2004.

[9] Q. J.R, Machine Learning 1, Boston - Manufactured in The Netherlands: Kluwer Academic Publishers, 1986.

[10] L. Breiman, "Random Forests," *Machine Learning Journal Paper,* vol. 45, 2001.

[11] H. Deng and G. Runger, "Feature selection via regularized trees," in *International Joint Conference on Neural Networks(IJCNN)*, 2012.

[12] H. Deng and G. Runger, "Gene selection with guided regularized random forest," Journal of Pattern Recognition, vol. 46, pp. 3483-3489, 2013.

[13] M. K. e. a. Halushka, "Patterns of single-nucleotide polymorphisms in candidate genes for blood-pressure," Nature Genet., vol. 22, p. 239–247, 1999.

[14] Y. Y. Y. L. a. M. K. N. Q. Wu, "Snp selection and classification of genome-wide snpdata using stratified," The Journal of IEEE Transactions on NanoBioscience, vol. 11, no. 3, p. 216–227, 2012.

[15] Leo Breiman, Jerome Friedman, Charles J. Stone, R.A. Olshen, Classification and Regression Trees, Taylor & Francis, 1984.

[16] Nguyen, Thanh-Tung, Joshua Z. Huang, and Thuy Thi Nguyen. "Two-level quantile regression forests for bias correction in range prediction." Machine Learning 101.1-3 (2015): 325-343.

[17] Bradley Efron, Bootstrap Methods: Another Look at the Jackknife, The Annals of Statistics, 1979.

[18] Thanh-Tung Nguyen, Huong Nguyen, “Classifying gene data with regularized,” 2005.

[19] Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[20] Han Jiawei, Micheline Kamber, Data Mining: Concepts and Techniques, 2000.