ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN HỌC MÁY NÂNG CAO

Cao học khóa 2022

Đề tài:

ÁP DỤNG DECISION TREE CHO CÁC ĐO LƯỜNG Y KHOA KHÔNG CHẮC CHẮN

Dựa trên bài báo:

Decision Tree Learning for Uncertain Clinical Measurements
Cecília Nunes, Hélène Langet, Mathieu De Craene, Oscar Camara, Bart Bijnens, Anders Johnsson,
IEEE Transactions on Knowledge and Data Engineering

GIÁO VIÊN HƯỚNG DẪN: TS. Trần Thái Sơn

HỌC VIÊN THỰC HIỆN: Đoàn Minh Hòa – 22C15028

Võ Hoài Danh - 22C15025

Lê Thị Cẩm Thi – 22C15044

Nguyễn Quốc Thắng – 22C15043

TPHCM - 4/2023

MỤC LỤC

PHÂN CÔNG CÔNG VIỆC

STT	MSSV	Họ và tên	Phân công công việc	Vị trí trang báo cáo thực hiện
1	**22C15028**	Đoàn Minh Hòa	- Tìm hiểu và phụ trách phần: Tóm tắt đồ án; 1. Giới thiệu; 2. Những nghiên cứu gần đây. - Tổng hợp nội dung của các thành viên và biên soạn.	**Từ:** Trang #1 (tất cả trang). **Đến:** Trang #9 (…sẽ mượt mà thay đổi p(y|x)).
2	**22C15025**	Võ Hoài Danh	- Tìm hiểu và phụ trách phần: 3. Phương pháp tiếp cận. - Rà soát, kiểm tra và góp ý nội dung.	**Từ:** Trang #9 (3. Phương pháp tiếp cận). **Đến:** Trang #13 (…ảnh hưởng của nhiễu Gauss).
3	**22C15044**	Lê Thị Cẩm Thi	- Tìm hiểu và phụ trách phần: 4: Kết quả thực nghiệm; 5. Kết quả. - Rà soát, kiểm tra và góp ý nội dung.	**Từ:** Trang #13 (4. Kết quả thực nghiệm). **Đến:** Trang #19 (…so với PLT).
4	**22C15045**	Nguyễn Quốc Thắng	- Tìm hiểu và phụ trách phần: 6. Kiểm chứng thực nghiệm. - Rà soát, kiểm tra và góp ý nội dung.	**Từ:** Trang #19 (6. Kiểm chứng thực nghiệm). **Đến:** Trang #25 (…tăng nhiều nhất có thể).

TÓM TẮT ĐỒ ÁN

Bài báo khoa học giới thiệu việc ứng dụng Cây quyết định (Decision Tree – DT) trong lĩnh vực y tế, cụ thể là hỗ trợ các chuyên gia y tế đưa ra các quyết định chuẩn đoán. Dữ liệu trong lĩnh vực y tế thường không chắc chắn, có độ chính xác thấp hoặc không đầy đủ, do đó việc xử lý các dữ liệu này là một thách thức vô cùng lớn. Cây quyết định là một công cụ hữu ích để hỗ trợ đưa ra các quyết định trong lĩnh vực y tế, nhờ khả năng giải thích rõ ràng, giúp các chuyên gia y tế đưa ra các quyết định dựa trên thông tin có sẵn, có thể nói DT là một trong những mô hình giải thích được rõ ràng và cách mô hình đưa ra kết quả, giúp cho các chuyên gia có thể xem xét cách mô hình hoạt động dẫn đến kết quả dự đoán.

Tuy nhiên, DT có một số hạn chế khi làm việc với dữ liệu y tế. Việc mô hình DT học các dữ liệu y tế dẫn đến việc kém khả năng tổng quát hóa và khả năng dự đoán không chính xác trong nhiều trường hợp. Vì thế, nhiều phương pháp đã được đề xuất như sử dụng Probabilistic Decision (Quyết định xác suất) tại các nút trong cây quyết định để làm cho tỉ lệ chính xác cao hơn với dữ liệu y tế đầu vào, nhưng hiệu quả trong các giai đoạn khác nhau của việc học của mô hình DT vẫn chưa thực sự rõ ràng.

Bài báo giới thiệu một phương pháp DT xác suất để mô hình hóa sự không chắc chắn đo lường dữ liệu y tế như một phân phối nhiễu. Cách tiếp cận của phương pháp gồm (1) tìm kiếm các ngưỡng chia, (2) khi chia các trường hợp huấn luyện và (3) khi tạo ra dự đoán cho các dữ liệu mới. Kết quả cho thấy phương pháp này giúp giảm kích thước của cây quyết định một cách đáng kể, trong khi vẫn giữ được độ chính xác, đối với khi nhiễu tăng lên, tuy nhiên không có lợi ích trong việc xử lý nhiễu trong quá trình đánh giá.

1. GIỚI THIỆU

Khai thác dữ liệu lâm sàng (Clinical data mining) là việc áp dụng các phương pháp và công nghệ máy tính để nghiên cứu và đưa ra các quyết định trong lĩnh vực y tế. Tuy nhiên, sự ảnh hưởng của nó phụ thuộc vào độ chính xác của các mô hình và khả năng tích hợp vào phần mềm lâm sàng. Mặc dù, có nhiều thông công về các phương pháp, nhưng vẫn còn khá nhiều hạn chế, ví dụ các dữ liệu y tế được xem là dữ liệu không chắc chắn, có độ chính xác thấp và không đầy đủ, điều này ảnh hưởng đến kết quả và hiệu suất của mô hình và giảm khả năng tổng quát hóa.

Uncertain clinical data là dữ liệu lâm sàng không chắc chắn. Điều này có thể bao gồm những dữ liệu mà không có đủ thông tin để đưa ra kết luận rõ ràng, hoặc những dữ liệu mâu thuẫn với những nghiên cứu khác hoặc dữ liệu chưa được xác thực. Những dữ liệu không chắc chắn này có thể ảnh hưởng đến quyết định điều trị và chăm sóc bệnh nhân, vì vậy việc đánh giá và quản lý chúng là rất quan trọng trong lâm sàng.

Để cải thiện chất lượng của dữ liệu đòi hỏi phải có nhiều tài nguyên, cần nhiều máy móc đo lường hiện đại,…và điều này hoàn toàn không khả thi. Nhóm tác giả, chấp nhận dữ liệu y tế đầu vào không chắn chắn, bằng cách cải tiến mô hình và đưa ra các phương pháp mới hơn kết hợp với kiến thức của các các chuyên gia thì sẽ là một trong những phương pháp tối ưu để hỗ trợ quyết định lâm sàng. Quá trình để đưa ra một quyết định lâm sàng sẽ bao gồm việc tham khảo dự đoán của các mô hình trên dữ liệu không chắc chắn kết hợp với kinh nghiệm thực tế của các chuyên gia y tế.

Trong nghiên cứu này, nhóm tác giả đề cập đến mô hình Cây quyết định (DTs), đây là một cấu trúc biểu diễn tri thức, trong đó các quyết định tại các nút trong cây dẫn đến dự đoán ở các nút lá. Các thuật toán phổ biến của DTs bao gồm ID3, C4.5, CART và CHAID.

So với các phương pháp dự đoán khác thì DTs có thể được giải thích dưới dạng các chuỗi quyết định, điều này có nghĩa là nhìn vào kết quả dự đoán chúng ta có thể truy xuất được cách mà mô hình đưa ra dự đoán, mở rộng hơn nữa thì trong những năm gần đây việc XAI (Explainable AI) trở nên khá phổ biến, đặc biệt trong lĩnh vực y tế vì các chuyên gia y tế không thể chỉ nhìn vào kết quả dự đoán của mô hình để đưa ra quyết định cuối cùng, mà phải hiểu cách mô hình thực hiện để đưa ra dự đoán. Và DTs được xem là một trong những mô hình rất tốt trong lĩnh vực XAI để phục vụ việc giải thích mô hình. Mặc dù, hiện nay các mạng học sâu (deep neural networks) đang khá phổ biến với độ chính xác tốt hơn, nhưng quá trình đưa ra kết quả thường là một "hộp đen".

Bên cạnh đó, các phương pháp như Random Forest hay Gradient Boosting cũng hỗ trợ khá tốt để cải thiện tính chính xác và ổn định của DTs. Trong bài báo này, nhóm tác giả tập trung vào một mô hình DTs đơn lẻ có tính giải thích cao, giải thích rõ quá trình đưa ra quyết định dự đoán và đảm bảo rằng các quyết định được giải thích một cách đầy đủ, *đặc biệt trong bối cảnh hiện nay các quốc gia đang rất quan tâm vấn đề "Xác định trách nhiệm pháp lý của AI trong quá trình dự đoán".*

Quá trình học của DTs từ các dữ liệu, đo lường không chắc chắn có thể dẫn đến việc quá khớp và không thể dự đoán chính xác trong các dữ liệu mới (Các dữ liệu chưa được huấn luyện), một mô hình có khả năng tổng quát hóa tốt, có nghĩa không chỉ tốt trên dữ liệu huấn luyện mà còn tốt trên dữ liệu mới, không được sử dụng trong huấn luyện, còn đối với trường hợp này thì khả năng DTs không thể tổng quát hóa tốt. Thực tế thì DTs là một cấu trúc lưu trữ kiến thức, trong đó quyết định tại một nút trong cây dẫn đến một dự đoán ở các nút lá của cây. Mỗi lần thực hiện một kiểm tra tại một nút trong DTs, đặc trưng của dữ liệu sẽ được so sánh với một ngưỡng cứng (hard threshold), những sai sót nhỏ trong đo lường có thể dẫn đến việc các trường hợp đi vào một nhánh khác của cây, dẫn đến kết quả dự đoán sai. Vì thế, để giải quyết cho các trường hợp sai số này, một số thuật toán sẽ sử dụng ngưỡng mềm (soft threshold) thay vì ngưỡng cứng để làm cho cây quyết định đưa ra kết quả tốt hơn đối với các dữ liệu không chắc chắn.

2. NHỮNG NGHIÊN CỨU GẦN ĐÂY

Một số phương pháp để xử lý dữ liệu không chắc chắn như Fuzzy DTs, *đây là một phương pháp sử dụng logic mờ để xử lý sự không chắc chắn trong quá trình đưa ra quyết định, ở đây thuật toán sử dụng các giá trị mờ để mô tả sự không chắc chắn của dữ liệu, thay vì chỉ có giá trị đúng hoặc sai như trong logic nhị phân, thì trong fuzzy logic các giá trị có thể nằm giữa đúng và sai, được biểu diễn dưới dạng các giá trị mờ, trong trường hợp này Fuzzy DTs sẽ thiết lập các hàm fuzzy để diễn tả sự không chắc chắn xung quanh ngưỡng. Ngoài ra, các thuật toán rời rạc hóa (discretization algorithm) cũng có thể được sử dụng để xử lý trong trường hợp này, thuật toán này là các phương pháp trong khoa học dữ liệu để chuyển đổi dữ liệu liên tục thành dữ liệu rời rạc, thuật toán này chia các giá trị liên tục thành khoảng giá trị rời rạc để đại diện cho dữ liệu, giúp giảm chiều dữ liệu*. Tuy nhiên, cả hai phương pháp này đều có chi phí tính toán cao và độ chính xác của DTs thường phụ thuộc vào thuật toán được sử dụng để xây dựng. Probabilities DTs cũng là một phương pháp hiệu quả trong việc xử lý dữ liệu y tế không chắc chắn do tính ngẫu nhiên hoặc nhiễu ngẫu nhiên. Giải thích rõ hơn về Probabilites DTs, thì đây là một loại cây quyết định được sử dụng để xử lý các dạng dữ liệu không chắc chắn và nhiễu. Thay vì chỉ sử dụng các ngưỡng cứng như DTs thông thường, thì các thuật toán Probabilities DTs sử dụng các giải pháp thống kê để đo lường mức độ không chắc chắn của dữ liệu đầu vào, và tính toán xác suất cho từng nhánh của cây để dự đoán kết quả. Các phương pháp trên, phần nào đã chỉ ra được lợi ích của DTs trong việc cải thiện hiệu suất với dữ liệu không chắc chắn, tuy nhiên các chiến lược và phương pháp trên chưa thực sự hiệu quả và độ chính xác chưa cao. Vì thế, nhóm tác giả trong nghiên cứu đã đề xuất một phương pháp Probabilities DTs để xử lý dữ liệu không chắc chắn, được mô hình hóa dưới dạng phân phối của nhiễu trong các đo lường thực tế. Khác với các phương pháp trước đó, phân phối này được xem xét như sau:

(1) Trong quá trình huấn luyện mô hình, quá trình tìm kiếm ngưỡng tốt nhất tại các nút gọi là tìm kiếm mềm (Soft search – SS);

(2) Trong quá trình truyền dữ liệu trong qua trình huấn luyện qua DTs, gọi là truyền dẫn mềm (Soft training propagation – STP);
(3) Trong quá trình kiểm tra đánh giá một DTs đã hoàn thành, gọi là đánh giá mềm (Soft Evaluation – SE).

Để đảm bảo tính trực quan, phương pháp sử dụng cây quyết định đơn biến, trong đó mỗi nút chứa một DT dựa trên một biến đơn. SS được đề xuất để giữa cho chi phí tính toán được kiểm soát. Nhóm tác giả giải quyết vấn đề tích hợp kiến thức về sự không chắc chắn đến từ kinh nghiệm lâm sàng và phân tích dữ liệu lâm sàng. Làm rõ tác động của nhiễu đến các giai đoạn học khác nhau bằng cách sử dụng mô hình nhiều chuẩn, tác giả thực hiện các thí nghiệm riêng biệt để nghiên cứu tác động của việc làm mất mác dữ liệu trong dữ liệu huấn luyện và dữ liệu kiểm tra. Trong bài báo nghiên cứu, sử dụng thuật toán ID3 và C4.5 để giải thích về xác suất.

2.1. Decision Tree

Trong phần này, chúng tôi sẽ giải thích lại một số khái niệm về Cây quyết định (Decision Tree), một số thuật toán phổ biến của DTs là ID3, C4.5 và CART. Sau đó, sẽ đi trực tiếp vào cách tiếp cận của nhóm tác giả trong bài nghiên cứu. Cũng giống như những giải thuật khác thì DTs cũng là một thuật toán của Học máy có thể thực hiện hai tác vụ phân loại và hồi quy, thậm chí là cả tác vụ đa đầu ra. Thuật toán này khá mạnh mẽ và có khả năng khớp những dữ liệu phức tạp, DTs cũng là một trong những nền tảng của Rừng ngẫu nhiên (Random Forest) một trong những thuật toán mạnh mẽ nhất hiện nay. Một trong những đặc điểm của DTs đó là chúng không đòi hỏi ta phải chuẩn bị dữ liệu nhiều. Ta thậm chí không cần phải co giãn hoặc căn giữa giá trị của các đặc trưng.

Mô hình DTs rất dễ hiểu và quyết định mà chúng ta gọi là diễn dịch dễ dàng. Các mô hình như vậy thường được gọi là các mô hình hộp trắng (white box). Ngược lại, các mô hình sau này như Rừng ngẫu nhiên hay mạng neural thường được xem là mô hình hộp đen (black box). Mô hình hộp đen có thể đưa ra dự đoán tốt và có thể dễ dàng kiểm tra các phép tính dẫn đến dự đoán đó. Tuy nhiên, căn cứ mà mô hình đưa ra dự đoán lại khó có thể được giải thích bằng các từ ngữ đơn giản. Ví dụ, nếu một mạng neural cho rằng một người nào đó xuất hiện trong bức ảnh, rất khó để biết yếu tố nào góp phần vào dự đoán đó: Liệu đó là do mô hình nhận ra mắt, miệng hay mũi của họ? Nhận ra giày hay thậm chí là chiếc ghế người đó đang ngồi? Trái lại, DTs cung cấp các quy tắc phân loại rõ ràng và đơn giản mà thậm chí có thể được sử dụng thủ công nếu cần.

Xét trường hợp ta có dữ liệu đầu vào là X với các chiều X_i, j = 1, ..., M và kết quả đầu ra là Y và phân phối không xác định P_XY(x, y). Trong học có giám sát, mục tiêu là học một mô hình dự đoán đầu ra là Y dựa trên đầu vào là X, không phải học toàn bộ phân phối xác suất P_XY(x,y). Một cây quyết định sẽ học mối quan hệ đầu vào và đầu ra bằng cách tạo ra các quy tắc quyết định dựa trên các giá trị đặc trưng đầu vào. Tuy nhiên, vì cây quyết định chỉ tập trung vào các quyết định tại các điểm nút, nên nó có thể bỏ qua các mối quan hệ phức tạp giữa các đặc trưng và đầu ra trong P_XY(x,y). Do đó, có thể sẽ bị giới hạn trong việc giải quyết các vấn đề phức tạp. Tuy nhiên, DTs vẫn là một phương

pháp học máy phổ biến và hiệu quả trong nhiều tình huống, đặc biệt khi dữ liệu huấn luyện có kích thước lớn và có tính chất phân loại hoặc hồi quy đơn giản.

Một số thuật toán của DTs gồm ID3, C4.5, CART, trong đó:

- *CART (Classification and Regression Tree) là một thuật toán tham lam (greedy algorithm). Thuật toán này tìm kiếm tham lam cách chia tối ưu tại mức đầu tiên rồi lặp lại quá trình ở các mức sau đó. Nó không hề kiểm tra xem cách chia đó có giúp độ pha tạp thấp nhất có thể tại các mức dưới hay không. Một thuật toán tham lam thường tìm ra nghiệm tốt nhưng không đảm bảo đó là nghiệm tối ưu.*
- *ID3 (Iterative Dichotomiser 3) là một thuật toán của DTs dùng cho các bài toán phân loại. ID3 sử dụng information gain để tìm ra các đặc trưng tốt nhất để phân chia tập dữ liệu ở mỗi nút. Tuy nhiên, ID3 chỉ hoạt động với các đặc trưng là numerical và categorical, không xử lý được các giá trị bị mất (missing value).*
- *C4.5 là phiên bản tiếp theo của ID3 được cải tiến để hỗ trợ xử lý các giá trị bị mất (missing value), numerical và categorial. C4.5 sử dụng gain ratio thay cho information gain để tối ưu hóa chọn các đặc trưng và giảm thiểu tác động của các đặc trưng bị mất đi. Ngoài ra, C4.5 còn hỗ trợ DTs với nhiều đầu ra (multi-class) và có phương pháp tổng hợp tốt hơn để xử lý các giá trị liên tục.*

Theo tìm hiểu thì hiện không có một thuật toán nào là tối ưu nhất, mỗi thuật toán đều có ưu điểm và hạn chế riêng, và sự lựa chọn của thuật toán phụ thuộc vào nhiều yếu tố như đặc điểm của dữ liệu, mục tiêu của bài toán và nhu cầu thực tế của người sử dụng. Ví dụ, CART được sử dụng phổ biến trong các bài toán có đầu ra là các biến liên tục, trong khi C4.5 thì thường được sử dụng cho các bài toán có biến rời rạc. ID3 là một thuật toán cũ hơn và đã được nâng cấp thành C4.5, vì vậy ít được sử dụng hơn trong các bài toán thực tiễn.

2.2. Split Search

Trong phần này tác giả cũng giới thiệu chi tiết thuật toán ID3, ý tưởng chính xây dựng cây theo phương pháp từ trên xuống (top-down), tức là bắt đầu từ nút gốc và tiếp tục phân tách tập dữ liệu huấn luyện để tạo các nút con cho đến khi đạt được điều kiện dừng. Trong quyết định nhị phân, mỗi nút được xác định bởi một hàm gọi là hàm chia nhánh (branch function), được ký hiệu là $B(n)(\mathbf{x})$ với n là nút hiện tại của cây, x là dữ liệu đang xét. Hàm $B(n)(\mathbf{x})$ sẽ quyết định mẫu dữ liệu x sẽ thuộc về nhánh trái (nL) hay nhánh phải (nR) của nút n, dựa trên giá trị của thuộc tính được chọn và ngưỡng được định nghĩa.

$$B^{(n)}(\mathbf{x}) = \begin{cases} n_L & \text{if } x_{j^{(n)}} < \tau^{(n)} \\ n_R & \text{otherwise,} \end{cases} \qquad (1)$$

Trong công thức (1), giá trị của chia nhánh sẽ được định nghĩa dựa trên giá trị của thuộc tính j(n) trong mẫu dữ liệu x và ngưỡng được định nghĩa là $\tau(n)$. Công thức (1) cho thấy nếu giá trị của thuộc tính x_j(n) của mẫu dữ liệu x nhỏ hơn ngưỡng $\tau(n)$ thì mẫu dữ

liệu đó sẽ được đi theo nhánh trái và ngược lại sẽ đi theo nhánh phải. Quá trình tìm kiếm phân tách tốt nhất của tập dữ liệu huấn luyện được gọi là class seperation.

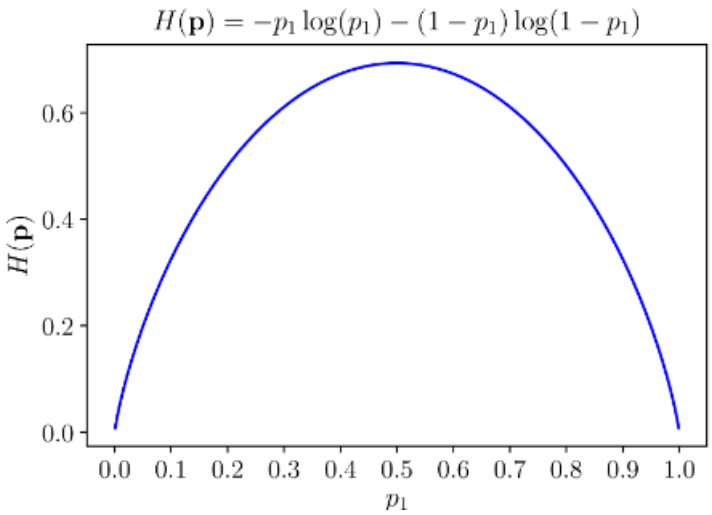
Class seperation trong DTs là một độ đo lường được sử dụng để tìm cách chia tập dữ liệu sao cho các lớp hoặc phân lớp (classes) khác nhau được tách biệt tốt nhất. Nói cách khác, class seperation đo lường độ tách biệt giữa các tập dữ liệu. Khi chọn một đặc trưng/ thuộc tính để chia tập dữ liệu, thuật toán DTs sẽ cố gắng tìm các thuộc tính và ngưỡng chia sao cho độ tách biệt giữa các lớp là lớn nhất. Điều này giúp giảm thiểu sự lẫn hỗn loạn giữa các lớp và tăng độ chính xác của mô hình.

Trong ID3 sẽ sử dụng Entropy & Information Gain để xây dựng cây quyết định. Đây là thước đo của sự biến đổi, hỗn loạn hoặc ngẫu nhiên. Entropy của phân phối được định nghĩa như sau:

$$H_\alpha(Y) = -\sum_y p(y) log_\alpha p(y),$$

Trường hợp này, xét α = 2 thì đây sẽ là một bài toán phân lớp nhị phân, ở đây p(y) tương ứng với số dữ liệu nhãn y chia có tổng số dữ liệu của bài toán.

Ví dụ: Với n = 2, thì p là tinh khiết nhất, tức là một trong hai giá trị p_i bằng 1 hoặc 0, entropy của phân phối H(Y) = 0. Khi p vẫn đục nhất tức là cả hai giá trị p_i =0.5, hàm entropy đạt giá trị cao nhất.



Hình 1 : Mô hình đường cong Entropy

Trong thuật toán ID3, tăng tính thuần khiết của lớp tương ứng với việc giảm entropy H(Y) trong nút trái và phải so với n. Nói cách khác, j(n) và τ (n) được chọn để tối đa hóa thông tin giữa B(n) và Y. Giải thích thêm về Entropy trong thuật toán ID3, được sử dụng để đo lường độ không đồng nhất của các lớp trong tập dữ liệu. Entropy có giá trị càng cao khi dữ liệu càng không đồng nhất, tức là các lớp có phân bố đều trong tập dữ liệu. Ngược lại, entropy có giá trị càng thấp khi dữ liệu càng đồng nhất, tức là các lớp có phân bố gần như đồng đều hoặc có một lớp chiếm đa số trong tập dữ liệu.

$$j^{(n)}, \tau^{(n)} = \arg\max_{j,\tau} I(B^{(n)}; Y). \qquad (2)$$

Sau khi tính toán các đặc trưng so với Y, ví dụ bài toán có n đặc trưng, thì áp dụng công thức tính Entropy ở trên để tính Entropy cho n đặc trưng đó, sau đó áp dụng công thức bên dưới để tính information gain của n đặc trưng và so sánh chọn đặc trưng có Information Gain lớn nhất (hoặc có nghĩa là Entropy thấp nhất). H(Y) là Entropy của toàn bộ dữ liệu, còn H(Y|B(n)) là Entropy của từng đặc trưng trên toàn bộ dữ liệu.

$$I(B^{(n)}; Y) = H(Y) - H(Y|B^{(n)}).$$

Trong DTs, I(B(n); Y) là information gain, đây là một đại lượng đo lường sự thay đổi entropy của biến phụ thuộc Y sau khi tập huấn luyện được phân chia bằng một biến độc lập B(n) tại một nút n cụ thể, information gain càng lớn thì dữ liệu càng đồng nhất.

$$\begin{aligned} j^{(n)}, \tau^{(n)} &= \arg\min_{j,\tau} H(Y|B^{(n)}) \\ &= \arg\min_{j,\tau} \sum_{b \in \{n_L, n_R\}} p(b) H(Y|B^{(n)} = b), \end{aligned} \qquad (3)$$

Phương trình (3) giải thích cách chọn thuộc tính và ngưỡng phân chia tối ưu cho nút n bằng cách tối thiểu hóa entropy có điều kiện của Y dựa trên đặc trưng phân chia B(n).

$$H(Y|B^{(n)} = b) = -\sum_{y=1}^{K} p(y|b) log_2 p(y|b). \qquad (4)$$

Trong phương trình (4), H(Y|B(n) =b) là entropy có điều kiện của Y cho biết B(n)=b. Entropy có điều kiện này đo lường lượng thông tin còn lại của Y sau khi biết B(n)=b.

Các ước lượng có độ tin cậy cao nhất của xác suất trong các phương trình (3) và (4) được tính dựa trên tập dữ liệu huấn luyện. Cụ thể, xác suất p^(b) được ước lượng bằng số các ví dụ trong tập huấn luyện thuộc n, chia cho tổng số ví dụ trong tập huấn luyện. Trong khi đó, xác xuất p^(y|b) được ước lượng bằng số lượng các ví dụ trong tập huấn luyện thuộc n có B(n)=b và Y=y, chia cho số lượng các ví dụ trong tập huấn luyện thuộc n có B(n)=b.

$$\hat{p}(b) = \frac{N_n(b)}{N_n} \text{ and } \hat{p}(y|b) = \frac{N_n(y,b)}{N_n(b)}, \qquad (5)$$

Công thức (5) đang giải thích cách tính các ước lượng giá trị xác suất tối đa của các biến trong phương trình (3) và (4). Cụ thể, p^(b) là ước lượng tối đa xác suất của biến chia nhánh B(n) được tính bằng cách đếm số lượng các mẫu trong thập huấn luyện thuộc node n rồi chia cho tổng số mẫu trong tập huấn luyện. p^(y|b) là ước lượng tối đa xác suất của

biến mục tiêu Y được tính bằng cách đếm số lượng các mẫu trong tập huấn luyện thuộc node n có giá trị B(n) bằng b và giá trị Y bằng y, rồi chia cho tổng số mẫu trong tập huấn luyện có giá trị B(n) bằng b. Trong bài báo, nhóm tác giả xem xét các biến dạng số và sử dụng phân chia nhị phân và sử dụng thông tin thu được làm tiêu chí phân chia. Trong phiên bản C4.5

Ở đây, nhóm tác giả có sử dụng maximum-likelihood, đây là một phương pháp ước lượng tham số của một phân phối xác suất dựa trên dữ liệu quan sát được. Phương pháp này tìm cách tối đa hóa xác suất dữ liệu được sinh ra từ phân phối đó, giúp tìm ra các giá trị của tham số tối ưu nhất. Đây là một phương pháp ước lượng tham số phổ biến trong thống kê và các lĩnh vực liên quan đến xác suất thống kê

2.3. Probabilistic interpretation of the split

Ở phần này, tác giả đưa ra cách tiếp cận xác suất thống kê thông qua phương pháp Bayesian, phương pháp này sử dụng để xác định xác suất của một sự kiện dựa trên kiến thức trước đó và bằng cách tích hợp các giả định trước đó vào quá trình dự đoán. Trong bài toán DTs, giả sử chúng ta có xác suất p(y|x) của y dựa trên mẫu x, từ đó chúng ta có thể đưa ra dự đoán về y. Chúng ta có thể ước tính p(y|x) từ DTs gốc tại nút n, với các nút con nL và nR. Công thức (6) được dựa trên phương pháp trung bình hóa mô hình Bayesian, giúp chuyển đổi sự không chắc chắn của mô hình khác nhau, trong trường hợp này là hai cây con nL và nR, thành sự không chắc chắn trong việc dự đoán lớp của một trường hợp xác định.

$$\begin{aligned}\hat{p}(y|\mathbf{x}) &= \textstyle\sum_{b\in\{n_L,n_R\}} p(y|b,\mathbf{x})p(b|\mathbf{x}) \\ &= \textstyle\sum_{b\in\{n_L,n_R\}} p(y|b)p(b|\mathbf{x}) \\ &= p(y|n_L)p(n_L|\mathbf{x}) + p(y|n_R)p(n_R|\mathbf{x}),\end{aligned} \tag{6}$$

Propapilistics DTs sử dụng ý tưởng này để thay thế bằng cách thể hiện sự không chắc chắn về mẫu quan sát x. Sự không chắc chắn này được mô hình hóa bằng posterior p(nL|x), được gọi là hàm gating, g_n(x). Ước tính của p(y|x) được mô tả theo công thức (7).

$$\hat{p}(y|\mathbf{x}) = p(y|n_L)g_n(\mathbf{x}) + p(y|n_R)\big(1 - g_n(\mathbf{x})\big). \tag{7}$$

Nếu chúng ta giả sử giá trị quan sát chính xác, nút n sẽ thực hiện một phân chia cứng như trong phương trình (1).

$$g_n(\mathbf{x}) = \mathbf{1}(x_{j^{(n)}} < \tau^{(n)}),$$

Trong trường hợp này, nếu x_j(n) gần với *τ(n)* các biến thể nhỏ về giá trị của nó có thể thay đổi đáng kể ước tính p(y|x), và gây ra dự đoán sai. Propabilistics DTs thay vào đó mô hình sự không chắc chắn của từng biến như một phân phối nhiễu và g_n(x) trở thành

hầm mật độ tích lũy (CDF) của phân phối được chọn. Một biến thể nhỏ của x_j(n) xung quanh giá trị ngưỡng sau đó sẽ mượt mà thay đổi p(y|x).

3. PHƯƠNG PHÁP TIẾP CẬN

Tác giả đề xuất xem các quan sát sẽ có nhiễu quanh giá trị quan sát và dùng một phân phối nhiễu để biểu diễn giá trị quan sát thay vì chỉ dùng chính giá trị quan sát (một giá trị cố định quan sát được).

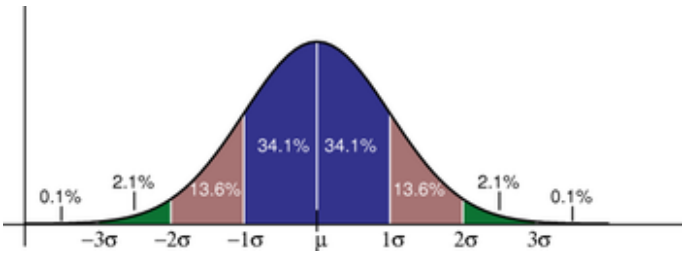
Tác giả chia làm 3 cách áp dụng phương pháp xác suất này:

(1) Áp dụng trong quá trình tìm kiếm ngưỡng phân tách (Quá trình chọn ngưỡng để quyết định mô hình Cây Quyết Định (DT) sẽ đi theo nhánh nào khi đứng tại một node nào đó trong cây quyết định) - gọi là "Soft search (SS)"

(2) Áp dụng trong quá trình training - gọi là "soft training propagation (STP)"

(3) Áp dụng trong quá trình kiểm tra đánh giá cây quyết định đã xây dựng – gọi là "soft evaluation (SE)"

Dựa vào thực nghiệm thì tác giả xác định thuộc tính quan sát X có nhiễu và nhiễu sẽ có phân phối chuẩn là [1]EX ~ N(0; σ2).



Hình 2 : Hình dạng của một phân phối chuẩn

3.1. Soft search

Tìm kiếm một ngưỡng chia (τ) tại một node "n" cụ thể trong DTs. Chúng ta bắt đầu với việc xét một biến X (hay ***thuộc tính*** *X). (Chú ý phân biệt giữa mẫu x và thuộc tính x trong bài viết, vì chúng ta chỉ xét trên một thuộc tính nên chúng ta bỏ qua chỉ số thể hiện thuộc tính). Và giả định rằng* ***thuộc tính*** *X này sẽ nhận D giá trị riêng biệt, được sắp xếp tăng dần.*

$$\{x^{(1)}, ..., x^{(D)}\} \text{ with } x^{(i)} < x^{(i+1)}, i = 1, ..., D-1$$

Chú ý: D phải nhỏ hơn số lượng mẫu trong tập "data" vì nếu ngược lại thì tập data không có đủ các trường hợp.

[1] https://vi.wikipedia.org/wiki/Ph%C3%A2n_ph%E1%BB%91i_chu%E1%BA%A9n

Ví dụ: Chỉ số BMI (lấy từ file "HCV-Egy-Data.csv" [2])

Age	Gender	BMI	Fever	Nause
56	1	35	2	

Hình 3 : Chỉ số BMI

BMI sẽ nhận giá trị từ 18,5-24,9 đối với người bình thường, <18,5 với người gầy, 25-40 vời người thừa cân (chỉ số càng lớn thì càng thừa cân). Trong trường hợp này biến X ~ chỉ số BMI, X sẽ nhận giá trị trong đoạn [0,40] (khoảng giá trị này ta có thể chọn, tùy theo kinh nghiệm và bài toán cụ thể, ta cũng có thể loại bỏ một số giá trị của X không có hoặc ít khi xảy ra, ví dụ chỉ số BMI = 0, chỉ số BMI < 10 vì chúng ko thể xảy ra trong thực tế).

Ví dụ: X ∈ {0, 0.5 ,1 , 1.5, .. , 40} chú ý số giá trị của X có thể nhận phải nhỏ hơn tổng số mẫu tập dataset. $x^{(0)} = 0$, $x^{(1)} = 0.5$ ...chúng ta không thể chia X thành 40 giá trị có thể nhận trong khi tập mẫu của chúng ta chỉ có 39 mẫu.

Quay lại vấn đề, bây giờ chúng ta tìm kiếm ngưỡng chia tại một node n cụ thể trong DTs.

$N(x^{(i)})$ *là số lượng mẫu mang giá trị* $x^{(i)}$.

$N(y, x^{(i)})$ *là số lượng mẫu thuộc class y.*

Đặt τ *là ứng cử viên cho ngưỡng chia của thuộc tính X tại node "n" . Trong phương pháp C4.5, khi tìm kiếm ngưỡng chia, vì phương pháp giả định rằng phép đo là tuyệt đối chính xác, nên ta có* $g(\mathbf{x}) = \mathbf{1}(x < \tau)$ *(xem trong phương trình (7) của bài báo).*

$$\hat{p}(y|\mathbf{x}) = p(y|n_L)g_n(\mathbf{x}) + p(y|n_R)\big(1 - g_n(\mathbf{x})\big). \qquad (7)$$

Nghĩa là xác suất dự đoán mẫu x phân vào class y sẽ chính bằng xác suất cây quyết định ***hoặc*** *rẽ trái* ***hoặc*** *rẽ phải tại node n. Bây giờ chúng ta tìm hiểu cách thức áp dụng xác suất khi xem các phép đo có nhiễu và không chắc chắn. Mục tiêu là chúng ta đi tìm hai giá trị (5) theo phương pháp xác suất của tác giả để* ***Maximum*** *"Information Gain" ở phương trình (2), hay* ***Minimum*** *độ hỗn loạn thông tin ở phương trình (3).*

$$\hat{p}(b) = \frac{N_n(b)}{N_n} \text{ and } \hat{p}(y|b) = \frac{N_n(y,b)}{N_n(b)}, \qquad (5)$$

Xem xét hàm chọn ***g(x)*** *(hàm chọn xem DT sẽ rẽ nhánh trái ($x \leq \tau$), $g(x)=1$ hay rẽ sang nhánh phải ($x > \tau$) tại node n) sẽ không còn mang giá trị hoặc 1, hoặc 0 mà* ***g(x)*** *sẽ là hàm tích lũy (CDF) [3]của phân phối chuẩn tại ngưỡng τ, g(x) từ bây giờ có thể mang giá trị*

---

[2] Link download: https://archive.ics.uci.edu/ml/machine-learning-databases/00503/HCV-Egy-Data.zip

[3] *(CDF) là hàm xác suất của biến X nhận giá trị bằng hoặc nhỏ hơn ngưỡng τ .*

Nguồn: https://vi.wikipedia.org/wiki/Ph%C3%A2n_ph%E1%BB%91i_chu%E1%BA%A9n

liên tục thuộc [0,1]. Ta thay thế tổng các g(x) trong phương trình (8) bằng tổng của các CDF của phân phối chuẩn, ta được phương trình (9):

$$\rho(\tau) = \sum_{\mathbf{x}\in\mathcal{D}} p(n_L|\mathbf{x}) = \sum_{\mathbf{x}\in\mathcal{D}} g_n(\mathbf{x}) = \sum_{\mathbf{x}\in\mathcal{D}} \int_{-\infty}^{\tau} K(\tau - x, \sigma)d\tau, \tag{9}$$

Ở đây ρ(τ) tương đương với p^(nL) trong phương trình (8).

$\int_{-\infty}^{\tau} K(\tau - x, \sigma)d\tau$ *Chính là công thức tính phân phối tích lũy (CDF) của phân phối chuẩn. Trong công thức này chúng ta xem x là giá trị trung bình của một quan sát (giá trị này có nhiễu gauss) và tính diện tích của phần đồ thị hàm gauss mà có* $x \leq \tau$.

Do cách tính như vậy nên ứng với mỗi mẫu $x \in \mathcal{D}$ *thì kết quả tính hàm g(x) mang giá trị thuộc đoạn [0,1].*

- *x càng xa và ở bên phải của τ thì CDF sẽ tiến về 0.*
- *x càng xa và ở bên trái của τ thì CDF sẽ tiến về 1.*
- *x càng gần τ thì CDF sẽ tiến gần về 0.5.*

Tóm lại, cách tính như vậy thì khi các mẫu x gần với ngưỡng chia thì mẫu x đó không được đếm là 1 mẫu (vì dùng hàm tính lũy để tính nên x khi đó ví dụ chỉ được tính là 0.5 nếu x trùng với τ). Các điểm x ở "đủ" xa ngưỡng chia τ thì vẫn được tính là 1 điểm. Khi chọn ngưỡng τ chúng ta cũng có thể chọn bất cứ giá trị nào phù hợp mà không cần chọn giá trị τ phải thuộc giá trị có trong dataset. τ có thể mang bất cứ giá trị nào trong đoạn giá trị của thuộc tính X.

Tác giả chọn τ đi từ τ_{min} *đến* τ_{max}

$$\tau = \tau_{min},\ \tau_{min} + \delta\sigma,\ \tau_{min} + 2\delta\sigma,\ \ldots,\ \tau_{max} \tag{11}$$

$$\tau_{min} = x^{(1)} - \frac{\omega}{2}\sigma \text{ and } \tau_{max} \leq x^{(D)} + \frac{\omega}{2}\sigma$$

<u>Cách thức tác giả "dò tìm" ngưỡng τ</u>: Tác giả thiết lập 2 vòng lặp chạy từ bên trái của τ và bên phải của τ, ρL(τ) and ρR(τ). Với điểm bắt đầu chạy từ τ_{min}, $\rho L(\tau) = 0$, $\rho R(\tau) = N$ *(với N là tổng số mẫu mà DT nhìn thấy tại node "n"). Mỗi lần τ tăng lên một đoạn δσ thì ρL(τ) tăng 1 khoảng Δρ(τ) và , ρR(τ) cũng giảm tương ứng 1 khoảng Δρ(τ), Δρ(τ) được tính bởi công thức bên dưới.*

$$\Delta\rho(\tau) = \begin{cases} \sum_{i=1}^{D} N(x^{(i)})\Phi\left(\frac{\tau - x^{(i)}}{\sigma}\right), & \text{if } \tau = \tau_{min} \\ \sum_{i=1}^{D} N(x^{(i)})\left[\Phi\left(\frac{\tau - x^{(i)}}{\sigma}\right) - \Phi\left(\frac{\tau - \delta\sigma - x^{(i)}}{\sigma}\right)\right], & \text{if } \tau_{min} < \tau < \tau_{max} \\ \sum_{i=1}^{D} N(x^{(i)})\left[1 - \Phi\left(\frac{\tau - \delta\sigma - x^{(i)}}{\sigma}\right)\right], & \text{if } \tau = \tau_{max} \end{cases}$$

Giải thích công thức:

Với $\tau_{min} < \tau < \tau_{max}$: $\Delta\rho(\tau) = \rho(\tau_t) - \rho(\tau_{t-1})$ (biểu thức là hiệu của 2 $\rho(\tau)$ ở điểm τ và điểm trước đó τ-$\delta\sigma$).

Với $\tau = \tau_{min}$: thì $\rho(\tau_{t-1}) \sim 0$ vì tất cả các điểm $x^{(i)}$ đều nằm bên phải của τ.

Với $\tau = \tau_{max}$: thì $\rho(\tau) \sim 1$ vì tất cả các điểm $x^{(i)}$ đều nằm bên trái của τ nên ta có kết quả tính $\Delta\rho(\tau)$ như ở trên.

Ở trên chúng ta đã tính được p(b) trong phương trình (5) tương ứng với $\rho(\tau)$ trong phương trình (9).

$$\hat{p}(b) = \frac{N_n(b)}{N_n} \text{ and } \hat{p}(y|b) = \frac{N_n(y,b)}{N_n(b)}, \qquad (5)$$

Chúng ta áp dụng cách tương tự cho ước lượng p(y|b) hay $\hat{p}(y|n_L)$

Ta đi tính tổng xác suất $\rho(y, \tau)$ (là $N_n(y,b)$ trong phương trình (5)) và tính $\Delta\rho(y,\tau)$, ta chỉ cần thay N(x(i)) bằng N(y, x(i)) vào phương trình (12), mọi thứ còn lại giữ nguyên. Sau khi đã có cách tính tập các giá trị của $\rho(\tau)$ hay p(b) và $\hat{p}(y|n_L)$ *hay p(y|b) ta áp dụng vào phương trình:*

$$\begin{aligned} j^{(n)}, \tau^{(n)} &= \arg\min_{j,\tau} H(Y|B^{(n)}) \\ &= \arg\min_{j,\tau} \sum_{b\in\{n_L,n_R\}} p(b)H(Y|B^{(n)} = b), \end{aligned} \qquad (3)$$

where

$$H(Y|B^{(n)} = b) = -\sum_{y=1}^{K} p(y|b)log_2 p(y|b). \qquad (4)$$

Và tìm kiếm giá trị τ thỏa mãn Minimum hàm $H(Y|B^{(n)})$ ở trên (Như đã nói ở phần đầu, chúng ta xét cách tìm ngưỡng chia "τ" trên 1 thuộc tính xác định cụ thể nên chúng ta bỏ qua chỉ số thuộc tính "j").

Trong bài báo này, tác giả đã chọn độ lệch σ theo kinh nghiệm của các bác sĩ khi quan sát trên nhiều mẫu dữ liệu, σ sẽ thay đổi tùy theo từng loại thuộc tính và chỉ có thể xác định σ dựa trên số liệu thống kê.

3.2. Soft training propagation

Đối với quá trình trainning, tác giả đề xuất phương pháp ***Soft training propagation (STP)*** *trong việc* ***chia*** *tập dữ liệu huấn luyện. Trong cách huấn luyện DT bình thường, tại mỗi node của DTs (xét ở 1 level của DT) thì mô hình DTs sẽ nhìn thấy một số lượng mẫu cố định, tức là N_n mang giá trị cố định tại node n. Trong phương pháp (STP) tác giả dùng hàm chọn g(x) (tương tự như trong phương pháp SS) để đánh trọng số cho các mẫu huấn luyện, một mẫu bây giờ không chỉ thuộc về một nhánh mà có thể thuộc về cả 2 nhánh, chúng ta có sự giao thoa việc đếm các mẫu gần ngưỡng chia τ, do đó làm tăng số lượng*

mẫu được nhìn thấy tại một node của DTs, việc này làm tăng số lượng tính toán "information Gain" tại level d của DT lên nhiu lần, cụ thể là tăng lên $2^d|D|M$ (là số mẫu DT nhìn thấy tại level d), do với mỗi giá trị của τ ta lại có 1 cách chia tập training khac nhau.

Ví dụ: Chỉ số BMI trong file "HCV-Egy-Data.csv"

Với ngưỡng cứng là 30: BMI <= 30 ta có 869 mẫu, BMI>30 ta có 516 mẫu.

Khi áp dụng hàm g(x) có dạng phân phối chuẩn.

Với ngưỡng mềm là 30: BMI <=30 ta có 900 mẫu > 869 mẫu, BMI>30 ta có 600 mẫu > 516 mẫu. (kết quả này chỉ là giả định).

3.3. Soft Evaluation

Sự không chắc chắn cũng được tính đến trong quá trình đánh giá mô hình DTs sau huấn luyện được gọi là ước lượng mềm. Thực hiện bằng cách cài đặt hàm chọn g(x) như trong phương trình (13) $g_n(\mathbf{x}) = F_{\mathcal{E}_X}(\tau - x) = \Phi\left(\frac{\tau - x}{\sigma}\right).$ (13) *với giá trị độ lệch* $\sigma = u_e.x_{mean}$ *, trong đó* u_e *là hệ số không chắc chắn và* x_{mean} *là giá trị trung bình của x.*

(Các giá trị u_e, x_{mean} *đều ước lượng từ tập huấn luyện)*

$$\hat{p}(y|\mathbf{x}) = p(y|n_L)\Phi\left(\frac{\tau - x}{\sigma}\right) + p(y|n_R)\left(1 - \Phi\left(\frac{\tau - x}{\sigma}\right)\right). \quad (14)$$

Cách tiếp cận này điều chỉnh giá trị xác suất $\hat{p}(y|\mathbf{x})$ *và phản ánh được ảnh hưởng của nhiễu Gauss.*

4. KẾT QUẢ THỰC NGHIỆM

Tác giả đề xuất một số thử nghiệm để đánh giá hiệu quả của các mô hình học máy trong các giai đoạn học tập khác nhau và ảnh hưởng của việc hỏng quá trình đào tạo với dữ liệu thử nghiệm có nhiễu. Mục tiêu của thử nghiệm là cải thiện hiệu suất dự đoán của các mô hình học máy bằng cách tối đa hóa độ chính xác tổng quát và giảm độ phức tạp của mô hình. Độ phức tạp của mô hình được đánh giá bằng cách đếm số lượng lá trong cây quyết định.

Các phương pháp SS, STP và SE được so sánh với thuật toán C4.5 độc lập. Các chiến lược cắt tỉa C4.5 và thiếu giá trị được sử dụng đồng nhất trong tất cả các thử nghiệm. Việc cắt tỉa được mở rộng với hiệu chỉnh Laplace và được khuyến nghị để ưu tiên khám phá các cây quyết định nhỏ hơn với độ chính xác tương đương. Nhóm tác giả cũng mở rộng việc đánh giá các phương pháp tiếp cận PLT và UDT trên nhiều bộ dữ liệu và loại nhiễu khác nhau.

4.1. Mô tả dữ liệu và hệ số tin cậy cắt xén – pruning confidence factor:

Tác giả sử dụng Bảng dữ liệu 1. Trong bảng 1 là dữ liệu lâm sàng từ các nguồn mở, bao gồm các kho lưu trữ máy học (ML) và KEEL của UCI. Các tính năng khác thường đã bị loại trừ. Chúng tôi đã tổng hợp 5 bộ dữ liệu bổ sung bằng cách sử dụng phương pháp

điều chỉnh của Guyon, có sẵn trong phân loại thực hiện của Scikit-learning. Các bộ dữ liệu được tạo với các thuộc tính khác nhau.

Vì tối ưu kích thước DT phụ thuộc vào vấn đề và tập dữ liệu, nên hệ số tin cậy cắt xén không được tối ưu hóa. Thay vào đó, đối với mỗi tập dữ liệu, hệ số tin cậy được cố định sao cho kích thước DT trung bình mà C4.5 đạt được thông qua xác thực chéo cross-validation (CV) là 15 lá. Điều này tương ứng với một cây nhị phân có gần 4 cấp độ, được coi là một số quyết định có thể quản lý/có thể giải thích được trong một lâm sàng trực quan hướng dẫn. Một số bộ dữ liệu quá nhỏ để đạt được 15 lá, vì vậy hệ số tin cậy của chúng được đặt thành 10 hoặc 5 lá. Hệ số tin cậy được cố định trên tất cả các thử nghiệm.

TABLE 1: Classification datasets used in the experiments.

Dataset	Source	No. variables	No. instances	No. classes
Hepatitis	KEEL	6	155	2
Heart disease	UCI ML	8	303	2
Pima Indians diabetes	UCI ML	8	768	2
South African heart	UCI ML	8	462	2
Breast cancer Wisconsin	UCI ML	39	569	2
Dermatology	UCI ML	34	366	6
Haberman's breast cancer	UCI ML	3	306	2
Indian liver patient	UCI ML	9	582	2
BUPA liver disorders	UCI ML	6	345	2
Vertebral column (2 classes)	UCI ML	12	310	2
Vertebral column (3 classes)	UCI ML	12	310	3
Thyroid gland	UCI ML	5	215	3
Oxford Parkinson's disease	UCI ML	22	194	2
SPECTF	UCI ML	44	266	2
Thoracic surgery	UCI ML	3	470	2
Synthetic 1	Guyon	15	30×500	2
Synthetic 2	Guyon	15	30×400	2
Synthetic 3	Guyon	20	30×300	2
Synthetic 4	Guyon	25	30×200	3
Synthetic 5	Guyon	20	30×250	3

Hình 4 : Bảng tập các dữ liệu theo nghiên cứu của nhóm tác giả

4.2. Thí nghiệm

Tác giả tiến hành thực nghiệm như sau: Đối với mỗi tập dữ liệu thực, 30 hoán vị train-test ngẫu nhiên đã được tạo, chứa lần lượt 70% và 30% dữ liệu. Sử dụng lấy mẫu phân tầng để duy trì tỷ lệ các lớp. Đối với mỗi tập dữ liệu tổng hợp, các phiên bản riêng biệt được tạo và chia thành 30 bộ khác nhau, sau đó được chia thành 70%-30% mẫu cho tập train và tập test.

Các thí nghiệm được thực hiện với các mức độ nhiễu khác nhau được thêm vào dữ liệu Train (Thí nghiệm 1) hoặc Test (Thí nghiệm 2). Nhiễu được thêm vào một biến X được lấy mẫu từ $N(0, n\,\bar{x})$, với n là $hệ\ số\ nhiễu\ -\ noise\ factor$ và $\bar{x}$ là giá trị trung bình của tập huấn luyện của X. Nhóm tác giả áp dụng cùng một n cho tất cả các biến trong tập dữ liệu và tất cả tính ngẫu nhiên đã được tạo với hạt giống cố định cho khả năng tái sản xuất.

Các yếu tố us, ut, ue của các phương pháp SS, STP và SE, cũng như tham số UDT w, kiểm soát độ lệch chuẩn của mô hình không chắc chắn. Như vậy, các tham số này được điều chỉnh thông qua CV trên mỗi trong số 30 hoán vị tập huấn luyện, cho từng tập dữ liệu

và cài đặt nhiễu. Việc tìm kiếm được thực hiện trong khoảng [0,02, 0,5] và với gia số 0,02. Giá trị trung bình của 30 giá trị thu được cho mỗi tham số là giá trị được chọn cuối cùng. Các tham số SS ω và δ lần lượt được đặt thành 6.0 và 0.1. Các thí nghiệm ban đầu cho thấy rằng chúng không ảnh hưởng đáng kể đến kết quả, với điều kiện là ω đủ lớn để chứa phần lớn mật độ của phân bố độ không đảm bảo và δ đủ nhỏ để đảm bảo một tập hợp lớn đối tượng bị phân tách. Các tham số PLT $\tau -$ và $\tau +$ được suy ra bằng phương pháp do Quinlan đề xuất [29]. Sau đây, chúng tôi tóm tắt các bước đánh giá và điều chỉnh mô hình cụ thể cho từng thử nghiệm.

4.2.1. Thí nghiệm 1: Thêm nhiễu vào dữ liệu huấn luyện

Thí nghiệm 1 nghiên cứu các phương pháp có nhiễu được thêm vào dữ liệu huấn luyện, vì vậy nhóm tác giả biểu thị hệ số nhiễu huấn luyện là n_{train}.

Cho mỗi bộ dataset, 70%-30% train-test hoán vị, n_{train} and mô hình:

1. Giữ lại 30% cho tập test

2. Nếu mô hình có tham số để thiết lập ($u_s, u_t, , u_e, hoặc\ w$), sử dụng the 70% cho tập huấn luyện để điều chỉnh nó bằng CV:

    (a) Tính 10 nếp gấp CV phân tầng.

    (b) Thêm nhiễu vào các nếp gấp đào tạo CV, được phân phối dưới dạng $N(0, n_{train}\ \bar{x})$. Không thêm nhiễu vào các nếp gấp xác thực CV.

    (c) Điều chỉnh tham số để tối đa hóa độ chính xác của CV.

3. Thêm nhiễu vào 70% tập training, có phân phối là $N(0, n_{train}\ \bar{x})$..

4. Đào một cái cây bằng cách sử dụng tập huấn luyện có nhiễu, và giá trị tham số đã chọn, nếu có.

5. Đánh giá cây trên tập kiểm tra 30%, trong đó không có nhiễu đã được thêm vào.

C4.5 and PLT không trãi qua bước 2.

4.2.2. Thí nghiệm 2: Nhiễu được thêm vào dữ liệu thử nghiệm

Thí nghiệm 2 đánh giá giá trị của các mô hình được xây dựng trên dữ liệu mà không thêm nhiễu trong việc dự đoán nhãn của các trường hợp thử nghiệm nhiễu. Bây giờ chúng ta ký hiệu n là n_{test}

Đối với mỗi bộ dữ liệu, 70%-30% hoán vị train-test, n_{test} và mô hình:

1. Cầm bộ test 30%

2. Nếu mô hình có tham số để đặt ($u_s, u_t, , u_e,$ hoặc w), hãy sử dụng tập huấn luyện 70% để điều chỉnh nó theo CV:

    (a) Tính 10 nếp gấp CV phân tầng.

(b) Không thêm nhiễu vào các nếp gấp đào tạo CV.

Thêm nhiễu vào các nếp gấp xác thực CV, phân phối được gọi là $N(0, n_{test}\ \bar{x})$.

(c) Điều chỉnh tham số để tối đa hóa độ chính xác của CV.

3. Tìm hiểu một cây bằng cách sử dụng tập huấn luyện 70% ban đầu và giá trị tham số đã chọn, nếu có

4. Thêm nhiễu vào tập kiểm tra 30%, như $N(0, n_{test}\ \bar{x})$.

5. Đánh giá cây trên noise bộ test 30%.

Như trước đây, C4.5 và PLT không trải qua bước 2. Lưu ý rằng tiếng ồn được thêm vào nếp gấp xác thực CV.

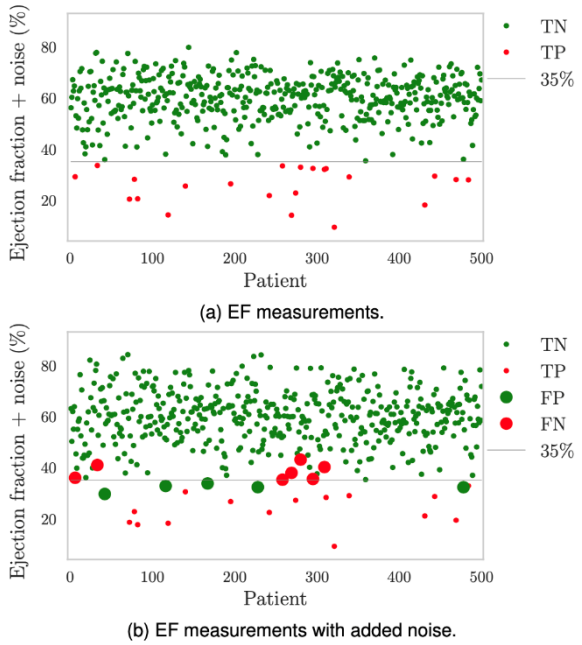
5. KẾT QUẢ

Phần 5.1 minh họa SS, STP và SE trên một biến duy nhất. Trong phần 5.2, chúng tôi sẽ trình bày kết quả của thí nghiệm 1 và 2.

5.1. Minh họa về một biến

Nhóm tác giả lấy ước tính phân suất tống máu *ejection fraction* (EF) của *Data Science Bowl Cardiac Challenge*. EF là một biến có tầm quan trọng quan trọng trong tim mạch. Liệu pháp thiết bị cấy ghép được khuyến cáo chính thức cho *EF < 35%*. Do đó, chúng tôi coi các trường hợp có *EF < 35%* là đủ điều kiện tích cực, tức là *Y = 1(EF < 35%)*, như trong Hình 5a. Việc thêm nhiễu ngẫu nhiên vào các dữ liệu này dẫn đến 7 kết quả âm tính giả - **False Negatives** (FN) và 5 kết quả dương tính giả **False Positives** (FP), như trong Hình 5b.

Tìm kiếm mềm: Trong Phần 3.4, nhóm tác giả đã thúc đẩy SS như một cách để tăng tập hợp các phân tách ứng cử viên và làm mịn information gain. Bây giờ chúng tôi quan sát điều này trên dữ liệu thực. Hình 6a hiển thị số lượng bệnh nhân cho mỗi loại và giá trị $EF, N(y, x^{(i)})$, là một biểu đồ histogram. Hình 6b hiển thị cùng một dữ liệu với nhiễu. Hình 6c cho thấy mức tăng mật độ SS $\Delta\rho(y, \tau)$ và SS information gain.



Hình 5 : Dữ liệu của Data Science Bowl Cardiac

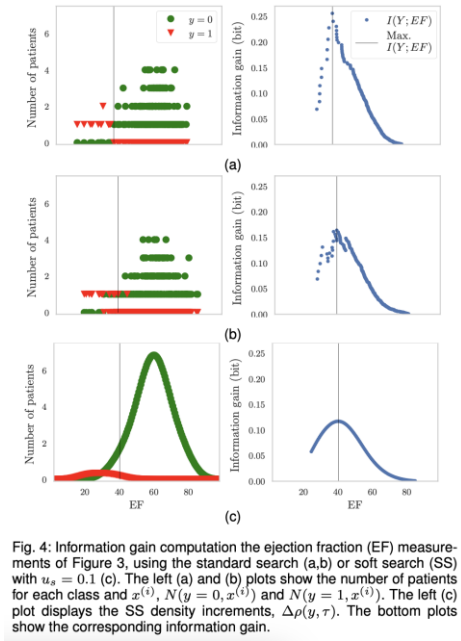


Fig. 4: Information gain computation the ejection fraction (EF) measurements of Figure 3, using the standard search (a,b) or soft search (SS) with $u_s = 0.1$ (c). The left (a) and (b) plots show the number of patients for each class and $x^{(i)}$, $N(y=0, x^{(i)})$ and $N(y=1, x^{(i)})$. The left (c) plot displays the SS density increments, $\Delta\rho(y,\tau)$. The bottom plots show the corresponding information gain.

Hình 6 : Biểu đồ hiển thị kết quả

Các phương pháp Soft search như SS hoặc UDT làm tăng tập hợp các phân tách đại diện. Tập dữ liệu này có 500 thành phần. Sử dụng UDT với hệ số lấy mẫu lại $s = 100$ làm tăng số lượng tính toán thu được thông tin từ 5e2 lên 5e4. Cài đặt các hệ số SS là $w = 8$, $\delta = 0{,}05$ và $u_s = 0{,}1$ giới hạn con số này ở mức tối đa xấp xỉ 2,3e3.

Soft training propagation - STP: Để minh họa cách STP thay đổi các ước tính xác suất của lớp, nhóm tác giả xem xét dữ liệu nhiễu của Hình 3b. Nhóm tác giả đã học được single-node two-leaf bằng cách sử dụng quá trình truyền huấn luyện tiêu chuẩn - standard training propagation và STP với $u_p = 0{,}1$. Kết quả cho thấy rằng các ước tính xác suất của lớp ít cực đoan hơn khi STP được sử dụng, vì chúng phản ánh sự lựa chọn phân phối nhiễu.

Soft evaluation: Kết quả cho thấy 7 FN và 5 FP được tạo ra bởi nhiễu được thêm vào bộ dữ liệu EF. Bảng 3 cho thấy ước tính xác suất của các ví dụ bị phân loại sai đó, thu được bằng cách sử dụng đánh giá cứng hoặc mềm, với $u_s = 0{,}1$. Số lượng FN và FP khác nhau vì thuật toán đã học được ngưỡng 35,37% thay vì 35%.

5.2. Kết quả thí nghiệm của nhóm tác giả

Số lượng lá trung bình, độ chính xác của thử nghiệm và thời gian chạy được tính toán trên 30 hoán vị thử nghiệm đào tạo, cho từng thử nghiệm và tập dữ liệu. Sự khác biệt tuyệt đối so với C4.5, tính trung bình trên tất cả các bộ dữ liệu.

Tuy nhiên, vì không thể so sánh trực tiếp các kết quả tuyệt đối của các bộ dữ liệu riêng biệt nên nhóm tác giả tập trung vào các số liệu chuẩn hóa. Kết quả của từng tập dữ liệu và phương pháp được chuẩn hóa theo đường cơ sở của tập dữ liệu. Kết quả cơ sở thu

được bằng C4.5 mà không có nhiễu bổ sung và được ước tính với 30 hoán vị. Việc tiêu chuẩn hóa bao gồm việc trừ đi giá trị trung bình của đường cơ sở và chia cho độ lệch chuẩn của đường cơ sở. Ví dụ. đường cơ sở của bộ dữ liệu bệnh tim có trung bình 15.0 lá và độ lệch chuẩn 3.0 lá. Trong trường hợp này, các kết quả chuẩn hóa 0.0 , −1.0 và 1.5 lần lượt chuyển thành 15.0, 12.0 và 19.5 lá. Theo bài báo, bảng 4 và 5 hiển thị các số liệu chuẩn hóa, tính trung bình trên tất cả các bộ dữ liệu và Hình 5 hiển thị các ô vuông tương ứng. Thời gian tính toán chỉ mang tính biểu thị vì các thử nghiệm được chạy trên một cụm và thông số kỹ thuật của máy có thể thay đổi đôi chút.

Tất cả các phương pháp đều cho thấy số lượng lá tăng lên và giảm độ chính xác của phép thử khi hệ số nhiễu tăng lên. Độ chính xác giảm rõ rệt hơn đối với các dự đoán được thực hiện trên dữ liệu thử nghiệm ồn ào, như đã thấy trong Bảng 5.

SS, STP và UDT thể hiện sự duy trì hoặc cải thiện đáng kể về mặt thống kê về độ chính xác so với C4.5, trong tất cả các tình huống nhiễu. Trong Bảng 4, chúng ta thấy rằng STP có độ chính xác cao hơn so với các phương pháp khác cho tất cả n_{train} và n_{test}. Đối với các phương pháp SS và STP, việc duy trì độ chính xác đi kèm với việc giảm đáng kể về mặt thống kê số lượng lá so với C4.5 và UDT. Việc giảm kích thước cây SS có ý nghĩa thống kê đối với các hệ số nhiễu lớn hơn 0.00. STP đã giảm thêm kích thước cây, có ý nghĩa đối với tất cả n_{train} và n_{test}. Việc duy trì độ chính xác của UDT so với C4.5 đi kèm với sự gia tăng số lượng lá. Phương pháp này chậm hơn đáng kể so với SS và STP.

TABLE 2: Class probability estimates at leaves n_L and n_R of the trees learned on the noisy ejection fraction (EF) dataset of Figure 3b. Decision trees (DTs) learned using standard or soft training propagation (STP).

	Standard training propagation	STP
$\hat{p}(y=0\|n_L)$	0.000	0.310
$\hat{p}(y=1\|n_L)$	1.000	0.790
$\hat{p}(y=0\|n_R)$	0.981	0.979
$\hat{p}(y=1\|n_R)$	0.018	0.021

TABLE 3: Class probability estimates for the misclassified examples of the noisy ejection fraction (EF) data in Figure 3b, estimated by the tree learned with C4.5 on the non-noisy data in Figure 3a.

		Hard evaluation		SE	
	Patient	$\hat{p}(y=0\|\mathbf{x})$	$\hat{p}(y=1\|\mathbf{x})$	$\hat{p}(y=0\|\mathbf{x})$	$\hat{p}(y=1\|\mathbf{x})$
False negative	6	1.00	0.00	0.54	0.46
	33	1.00	0.00	0.83	0.17
	269	1.00	0.00	0.67	0.33
	280	1.00	0.00	0.91	0.09
	295	1.00	0.00	0.51	0.49
	309	1.00	0.00	0.79	0.21
False positive	42	0.00	1.00	0.16	0.84
	116	0.00	1.00	0.33	0.67
	167	0.00	1.00	0.39	0.61
	228	0.00	1.00	0.30	0.70
	359	0.00	1.00	0.49	0.51
	478	0.00	1.00	0.30	0.70

Hình 7 : Kết quả đánh giá

Trong thử nghiệm 1, độ chính xác thu được bởi SE và PLT bằng hoặc nhỏ hơn so với độ chính xác thu được thông qua đánh giá cứng, như trong Bảng 4. Điều này đặc biệt rõ ràng khi $n_{train} = 0$. Số lượng lá không thay đổi vì các phương pháp này không ảnh hưởng đến quá trình huấn luyện .

Ngược lại, Bảng 5 cho thấy độ chính xác của SE vượt trội so với đánh giá cứng ở Thí nghiệm 2 với $n_{train} \geq 0{,}05$. Tuy nhiên, sự gia tăng này không có ý nghĩa thống kê. Nó gợi ý rằng phân bố độ không đảm bảo được xem xét trong phương pháp SE nắm bắt tốt hơn nhiễu được thêm vào dữ liệu, so với PLT.

6. KIỂM CHỨNG THỰC NGHIỆM

Nhóm chúng tôi đã mô tả lại thực nghiệm 1 và thực nghiệm 2 của nhóm tác giả tại một node của DT, để kiểm chứng sự tối ưu cảu SS, STP và SE cho UDT.

6.1. Mô tả dữ liệu thực nghiệm

*Nhóm chúng tôi sử dụng dữ liệu **Hepatitis data set** thuộc KEEL dataset được tác giả đề cập trong bảng 1. Tập dữ liệu này chứa hỗn hợp các thuộc tính có giá trị nguyên và thực, với thông tin về bệnh nhân bị ảnh hưởng bởi bệnh Viêm gan. Nhiệm vụ là dự đoán xem những bệnh nhân này sẽ có tế bào ung thư gan hay không.*

6.2. Phương pháp tiến hành

***Bước 1**: Xây dựng hàm thêm nhiễu cho từng distance theo phân phối $N(0, n_{test}\ \bar{x})$ đối với những dữ liệu liên tục.*

- *Sử dụng function make_circles của sklearn để random noises trong ngưỡng $n_{test}\ \bar{x}$.*
- *Dùng hàm buildin function abs để lấy trị tuyệt đối của dữ liệu nhiễu.*

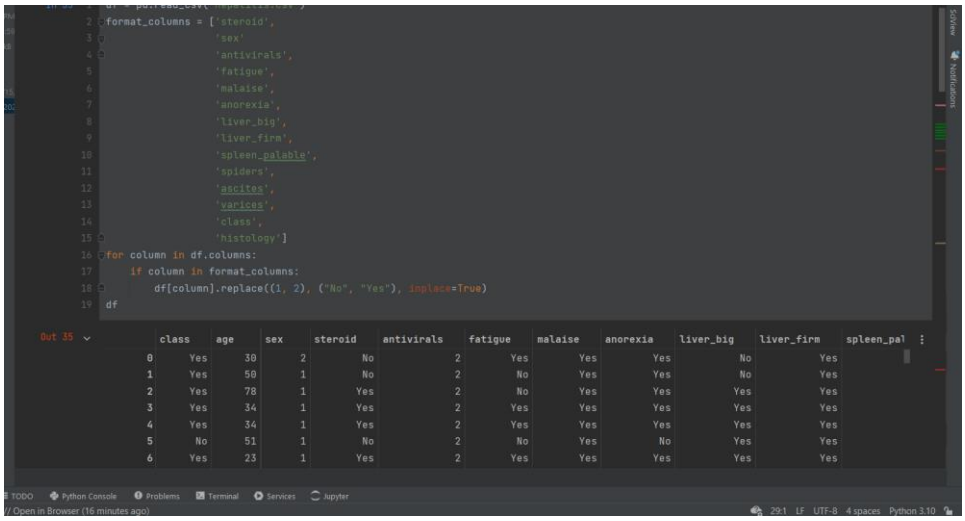
```
In 31
def seed_continuous_noises(distance, data):
    noises = make_circles(n_samples=30, noise=np.mean(data[distance]), random_state=1)
    return abs(noises[0][:, 1])

In 34
seed_continuous_noises('alk_phosphate', X_train)

Out 34
    alk_phosphate
0       89.967734
1       10.737972
2       97.174573
3        6.627700
4       39.038049
5        8.173140
6      178.208411
7      182.960786
30 rows × 1 columns  Open in new tab
```

***Bước 2**: Random dữ liệu nhiễu true false bằng hàm np.random.randint(2, size=30).*

***Bước 3**: Đưa các dữ liệu về "Yes", "No".*



```
format_columns = ['steroid',
                  'sex',
                  'antivirals',
                  'fatigue',
                  'malaise',
                  'anorexia',
                  'liver_big',
                  'liver_firm',
                  'spleen_palable',
                  'spiders',
                  'ascites',
                  'varices',
                  'class',
                  'histology']
for column in df.columns:
    if column in format_columns:
        df[column].replace((1, 2), ("No", "Yes"), inplace=True)
df
```

Out 35

	class	age	sex	steroid	antivirals	fatigue	malaise	anorexia	liver_big	liver_firm	spleen_pal
0	Yes	30	2	No	2	Yes	Yes	Yes	No	Yes	
1	Yes	50	1	No	2	No	Yes	Yes	No	Yes	
2	Yes	78	1	Yes	2	No	Yes	Yes	Yes	Yes	
3	Yes	34	1	Yes	2	Yes	Yes	Yes	Yes	Yes	
4	Yes	34	1	Yes	2	Yes	Yes	Yes	Yes	Yes	
5	No	51	1	No	2	No	Yes	No	Yes	Yes	
6	Yes	23	1	Yes	2	Yes	Yes	Yes	Yes	Yes	

Bước 4*: Thành lập một DF mới với 30 dữ liệu nhiễu.*

Bước 5*: Train C4.5 model và phương pháp SS của tác giả với 70% dữ liệu train có nhiễu, sau đó dùng 30% dữ liệu test để predict.*

- ***C4.5:*** *Ta thấy độ chính xác của model rơi vào khoảng 67%.*

```
config = {'algorithm': 'C4.5'}
model = chef.fit(noises_df, config=config, target_label='class')
```

```
-------------------------
Evaluate  train set
-------------------------
MAE:  0.03617571059431525
MSE:  0.018087855297157625
RMSE:  0.1344910974643215
RAE:  0.1559023911155809
RRSE:  0.30824118197058986
Mean:  0.7441860465116279
MAE / Mean:  4.861111111111112 %
RMSE / Mean:  18.072241221768202 %
```

```
predictions = []
for index, instance in X_test.iterrows():
    prediction = chef.predict(model, instance)
    predictions.append(prediction)
```

```
print("Accuracy:", metrics.accuracy_score(X_test['class'], predictions))
```

```
Accuracy: 0.6744186046511628
```

- ***SS:*** *Theo phương pháp của tác giả, thì giờ ta cần phai tìm một ngưỡng mềm, để tại ngưỡng đó information gain là lớn nhất.*

- *Đầu tiên ta phải xây dựng những hàm cần thiết cho việc xây dựng một cây nhị phân, như là tính:*

+ Entropy:

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i$$

```
def entropy(y):
    if isinstance(y, pd.Series):
        a = y.value_counts() / y.shape[0]
        e = np.sum(-a * np.log2(a + 1e-9))
        return e
```

+ Information Gain:

$$InformationGain_{Classification} = E(d) - \sum \frac{|s|}{|d|} E(s)$$

```
def variance(y):
    if len(y) == 1:
        return 0
    else:
        return y.var()

def information_gain(y, mask, func=entropy):
    a = sum(mask)
    b = mask.shape[0] - a

    if a == 0 or b == 0:
        ig = 0

    else:
        if y.dtypes != 'O':
            ig = variance(y) - (a / (a + b) * variance(y[mask])) - (b / (a +
b) * variance(y[-mask]))
        else:
            ig = func(y) - a / (a + b) * func(y[mask]) - b / (a + b) *
func(y[-mask])

    return ig
```

Giả sử ta tìm được information gain lớn nhất như trên ví dụ:

```
54  1  max_information_gain, best_split, _, _ = pdt.max_information_gain_split(noises_df['bilirubin'], noises_df['class'], )
    2  max_information_gain, best_split

54     (0.032616621221756244, 1.7731268328791447)

_   1
```

Tìm ngưỡng mềm: *Gom nhóm các dữ liệu có value giống nhau, gom các dữ liệu lại, sau đó chọn t bắt đầu bất kì, sau đó tính information gain từ 0 đến ngưỡng t. Làm tuần tự như vậy cho đến khi tìm được max information gain.*

```
def categorical_options(a):
    a = a.unique()

    opciones = []
```

```
    for L in range(0, len(a) + 1):
        for subset in itertools.combinations(a, L):
            subset = list(subset)
            opciones.append(subset)

    return opciones[1:-1]

def max_information_gain_split(x, y, func=entropy):
    split_value = []
    ig = []

    numeric_variable = True if x.dtypes != 'O' else False

    if numeric_variable:
        options = x.sort_values().unique()[1:]
    else:
        options = categorical_options(x)

    for val in options:
        mask = x < val if numeric_variable else x.isin(val)
        val_ig = information_gain(y, mask, func)
        ig.append(val_ig)
        split_value.append(val)

    if len(ig) == 0:
        return None, None, None, False

    else:
        best_ig = max(ig)
        best_ig_index = ig.index(best_ig)
        best_split = split_value[best_ig_index]
        return best_ig, best_split, numeric_variable, True
```

Để đảm bảo rằng decision tree ko bị quá cao, điều đó có thể dẫn đến việc bị overfitting, thì ta phải chỉ ra được chiều sâu tối đa của decision tree là bao nhiêu, số quan sát tối thiểu mà có thể tạo ra node mới, và số lượng các information gain tối thiểu sẽ tăng theo chiều cao của cây.

Ta cần tạo ra 3 functions:

1. *Tìm ra threshold tốt nhất để split dữ liệu*

```
def get_best_split(y, data):
    masks = data.drop(y, axis=1).apply(max_information_gain_split,
y=data[y])
    if sum(masks.loc[3, :]) == 0:
        return None, None, None, None

    else:
        masks = masks.loc[:, masks.loc[3, :]]

        split_variable = masks.iloc[0].astype(np.float32).idxmax()
        split_value = masks[split_variable][1]
        split_ig = masks[split_variable][0]
        split_numeric = masks[split_variable][2]

        return split_variable, split_value, split_ig, split_numeric
```

2. *Split tập dữ liệu thành 2 DF dựa trên threshold ở trên*

```
def make_split(variable, value, data, is_numeric):
    if is_numeric:
        data_1 = data[data[variable] < value]
```

```
        data_2 = data[(data[variable] < value) == False]

    else:
        data_1 = data[data[variable].isin(value)]
        data_2 = data[(data[variable].isin(value)) == False]

    return data_1, data_2
```

3. Đưa ra dự đoán

```
def make_prediction(data, target_factor):
    if target_factor:
        pred = data.value_counts().idxmax()
    else:
        pred = data.mean()

    return pred
```

Train SS decision tree

```
def train_tree(data, y, target_factor, max_depth=None,
min_samples_split=None, min_information_gain=1e-20, counter=0,
              max_categories=20):
    if counter == 0:
        types = data.dtypes
        check_columns = types[types == "object"].index
        for column in check_columns:
            var_length = len(data[column].value_counts())
            if var_length > max_categories:
                print
                'The variable ' + column + ' has ' + str(
                    var_length) + ' unique values, which is more than the
accepted ones: ' + str(max_categories)

    if max_depth == None:
        depth_cond = True

    else:
        if counter < max_depth:
            depth_cond = True

        else:
            depth_cond = False

    if min_samples_split == None:
        sample_cond = True

    else:
        if data.shape[0] > min_samples_split:
            sample_cond = True

        else:
            sample_cond = False

    if depth_cond & sample_cond:

        var, val, ig, var_type = get_best_split(y, data)

        if ig is not None and ig >= min_information_gain:

            counter += 1

            left, right = make_split(var, val, data, var_type)
```

```
            split_type = "<=" if var_type else "in"
            question = "{} {}  {}".format(var, split_type, val)
            subtree = {question: []}

            yes_answer = train_tree(left, y, target_factor, max_depth,
min_samples_split, min_information_gain, counter)

            no_answer = train_tree(right, y, target_factor, max_depth,
min_samples_split, min_information_gain, counter)

            if yes_answer == no_answer:
                subtree = yes_answer

            else:
                subtree[question].append(yes_answer)
                subtree[question].append(no_answer)

        else:
            pred = make_prediction(data[y], target_factor)
            return pred

    else:
        pred = make_prediction(data[y], target_factor)
        return pred

    return subtree
```

```
max_depth = 10
min_samples_split = 5
decisiones = pdt.train_tree(noises_df, 'class', True, max_depth, min_samples_split)
decisiones
```

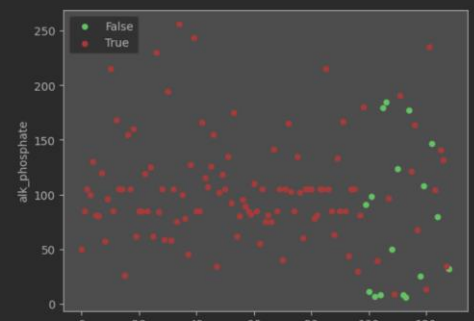
```
{'ascites <=  1': [{'bilirubin <=  0.5759003062397141': [1,
   {'albumin <=  3.945800537826263': [{'age <=  30.0': [1, 0]},
     {'age <=  90.83781345122324': [1, 0]}]}]},
  {'albumin <=  0.7816971811462268': [0,
    {'albumin <=  3.9': [{'sgot <=  95.87143614536438': [{'alk_phosphate <=  243.0': [{'protime <=  50.0': [{'albumin <=  3.8':
    [{'age <=  14.945929445139985': [0,
              1]},
            0]},
          1]},
        0]},
      {'malaise <=  1': [0, {'anorexia <=  1': [0, 1]}]}]},
    {'bilirubin <=  1.5650014211706278': [1,
      {'bilirubin <=  1.599939585877969': [0,
        {'sgot <=  85.10699277651857': [0, 1]}]}]}]}]}]}
```

Sau khi tạo cây ta sẽ tạo ra được một bộ decision rule như trên. Với kiến thức hiện tại thì nhóm chúng tôi chỉ có thể dựng được decision rules, và tìm được softsearch như trên bài báo, chứ chưa thể chạy predict một cây quyết định hoàn chỉnh.

6.3. Đánh giá kết quả

Đối với mô hình có nhiễu thì information gain của phương pháp softsearch cao hơn nhiều lần so với dữ liệu ban đầu.

```
colors = {False: 'green', True: 'red'}
fig, ax = plt.subplots()
ax.set_facecolor('silver')
grouped = noises_df.groupby('class')
for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x='patient', y='alk_phosphate', label=key, color=colors[key])
plt.show()
```



- *Dữ liệu ban đầu: Information Gain rơi vào khoảng 0.006, đây là giá trị cực thấp.*

```
information_gain, best_split, _, _ = pdt.max_information_gain_split(X_test['alk_phosphate'], X_test['class'])
(information_gain, best_split)
```

```
(0.006559332140727514, 85)
```

- *Dữ liệu sau khi thêm nhiễu: information gain lên 0.02.*

```
information_gain, best_split, _, _ = pdt.max_information_gain_split(noises_df['alk_phosphate'], noises_df['class'])
(information_gain, best_split)
```

```
(0.024287321830457634, 9.182115382333478)
```

Ta có thể tiếp tục lập đi lập lại quá trình và làm cho information gain tăng nhiều nhất có thể.

7. KẾT LUẬN

Trong bài báo này, tác giả trình bày phương pháp học Probabilitics DTs để xử lý dữ liệu không chắn chắn trong y tế, trong đó mô hình được phân chia thành 3 thành phần thuật toán độc lập. Trong phương pháp bao gồm quá trình học tìm kiếm ngưỡng tối ưu (SS), khi truyền dữ liệu huấn luyện qua cây (STP) và trong quá trình đánh giá khi thu được dự đoán cho dữ liệu không nhìn thấy (SE).

Kết luận của nghiên cứu cho thấy rằng việc xem xét đến sự không chắc chắn trong dữ liệu khi mô hình quyết định là rất quan trọng, đặc biệt trong lĩnh vực hỗ trợ quyết định y tế. Đồng thời, việc thêm nhiễu vào dữ liệu kiểm tra có tác động nghiêm trọng đến độ chính xác của mô hình so với việc thêm nhiễu vào dữ liệu huấn luyện, phương pháp của nhóm tác giả có thể giúp giảm kích thước của DTs và cải thiện độ chính xác so với phương pháp C4.5 truyền thống. Tuy nhiên, việc áp dụng các phương pháp này cần xác định được mô hình không chắc chắn phù hợp với dữ liệu cụ thể, thay vì giả định nhiễu là phân phối chuẩn đơn giản như trong nghiên cứu này.

Nghiên cứu cũng đưa ra một số hướng nghiên cứu tiếp theo, bao gồm xem xét phân phối nhiễu cụ thể cho từng lĩnh vực dữ liệu, nghiên cứu tính hiệu quả của các phương pháp mềm trên các loại dữ liệu khác nhau và xem xét các phương pháp xác định độ tin cậy của từng phép đo đối với cơ sở dữ liệu. Tổng thể, nghiên cứu này đóng góp quan trọng vào việc phát triển các phương pháp học máy cho hỗ trợ quyết định y tế và nhấn mạnh rằng việc xem xét sự không chắc chắn trong dữ liệu là rất quan trọng.

Tài liệu tham khảo

[1] Nunes, Cecília, et al. "Decision tree learning for uncertain clinical measurements." *IEEE Transactions on Knowledge and Data Engineering* 33.9 (2020): 3199-3211.

[2] V. L. Patel, E. H. Shortliffe, M. Stefanelli, P. Szolovits, M. R. Berthold, R. Bellazzi, and A. Abu-Hanna, "The coming of age of artificial intelligence in medicine," Artificial Intelligence in Medicine, vol. 46, no. 1, pp. 5–17, 2009.

[3] D. A. Clifton, K. E. Niehaus, P. Charlton, and G. W. Colopy, "Health Informatics via Machine Learning for the Clinical Management of Patients," Yearb Med Inform, vol. 10, no. 1, pp. 38–43, 2015.

[4] S. Tsang, B. Kao, K. Y. Yip, W.-S. Ho, and S. D. Lee, "Decision trees for uncertain data," IEEE transactions on knowledge and data engineering, vol. 23, no. 1, pp. 64–78, 2011.

[5] O. Irsoy, O. T. Yıldız, and E. Alpaydın, "Soft decision trees," in Pattern Recognition (ICPR), 2012 21st International Conference on. IEEE, 2012, pp. 1819–1822. [24] Y. Yuan, "Induction of fuzzy decision trees," Fuzzy Sets and Systems, vol. 69, no. 2, pp. 125–139, 1995.

[6] X. Wang, B. Chen, G. Qian, and F. Ye, "On the optimization of fuzzy decision trees," Fuzzy Sets and Systems, vol. 112, no. 1, pp. 117–125, may 2000.

[7] A. Segatori, F. Marcelloni, and W. Pedrycz, "On Distributed Fuzzy Decision Trees for Big Data," IEEE Transactions on Fuzzy Systems, pp. 1–1, 2017. [27] J. R. Quinlan, "Probabilistic decision trees," Machine learning: an artificial intelligence approach, vol. 3, pp. 140–152, 1990.

[8] J. R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1, no. 1, pp. 81–106, 1986.

[9] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers - A survey," IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, vol. 35, no. 4, pp. 476– 487, 2005.

[10] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian model averaging: a tutorial," Statistical science, pp. 382– 401, 1999.

[11] M. Lichman, "UCI Machine Learning Repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml [38] J. Alcal ´a-Fdez, A. Fern ´andez, J. Luengo, J. Derrac, S. Garc´ıa, L. S ´anchez, and F. Herrera, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," Journal of Multiple-Valued Logic and Soft Computing, vol. 17, no. 2-3, pp. 255–287, 2011.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[13] D. Jensen and T. Oates, "The effects of training set size on decision tree complexity," in Proceedings of the 14th International Conference on Machine Learning, 1999, pp. 254–262.

[14] National Heart, Lung, and Blood Institute, "Data Science Bowl Cardiac Challenge Data," 2015. [Online]. Available: www.kaggle.com/c/second-annual-data-science-bowl

[15] P. Ponikowski et al., "2016 ESC Guidelines for the diagnosis and treatment of acute and chronic heart failure: The Task Force for the diagnosis and treatment of acute and chronic heart failure of the European Society of Cardiology (ESC) Developed with the special contribution of

the Heart Failure Association (HFA) of the ESC," European heart journal, vol. 37, no. 27, pp. 2129–2200, 2016.
[16] J. Demsar, "Statistical Comparison of Classifiers over Multiple Data Sets," Journal of Machine Learning Research, vol. 7, no. 7, pp. 1–30, 2006.
[17] F. Wilcoxon, "Individual Comparisons by Ranking Methods," Biometrics Bulletin, vol. 1, no. 6, pp. 80–83, 1945.
[18] C. Clopper and E. Pearson, "The use of confidence or fiducial limits illustrated in the case of the binomial," Biometrika, vol. 26, no. 4, p. 404, 1934.