UỶ BAN NHÂN DÂN

THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC SÀI GÒN

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN

<Cải tiến thuật toán phân lớp kết hợp>

Mã số đề tài: SV2023-127

Thuộc nhóm ngành khoa học: Công nghệ thông tin

Chủ nhiệm đề tài: Trần Anh Tuấn

Thành viên tham gia: Trần Anh Tuấn

Giảng viên hướng dẫn: TS. Nguyễn Quốc Huy

Thành phố Hồ Chí Minh, 05/2024

UỶ BAN NHÂN DÂN

THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC SÀI GÒN

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN

<Cải tiến thuật toán phân lớp kết hợp>

Mã số đề tài: SV2023-127

Xác nhận của Khoa	**Giáo viên hướng dẫn**	**Chủ nhiệm đề tài**
(ký, họ tên)	*(ký, họ tên)*	*(ký, họ tên)*

Thành phố Hồ Chí Minh, 05/2024

3. Mục lục

4. Danh mục bảng biểu, hình ảnh

5. Danh mục các chữ viết tắt

SPMF	Sequential Pattern Mining Framework
ACAC	Associative classification based on all-confidence
ACCF	Associative classification based on closed frequent itemsets
ACN	An associative classifier with negative rules
CBA	Classification based on Associations
CMAR	Classification based on multiple class-association rules
FSAC	Feature selection association rule classification
Minconf	Min confidence
Minsup	Min support
MinAllconf	Min all-confidence
Rulesup	Rule support
Conf	Confidence
Condsup	Confidence support
Num	Number
FS	Feature selection
CAR	Classification association rules

6. Bản tóm tắt đề tài

Trong thời đại ngày nay, với sự phát triển mạnh mẽ của công nghệ thông tin, cơ sở dữ liệu giao tác, hoặc còn được biết đến với tên gọi dữ liệu giao dịch, đã trở thành một phần quan trọng không thể thiếu trong hệ thống thông tin của các tổ chức và doanh nghiệp. Dữ liệu này thường chứa thông tin về các giao dịch mua bán, hoạt động tài chính, hoặc thậm chí là hành vi người dùng trên các nền tảng trực tuyến. Với khối lượng dữ liệu lớn và tính phức tạp, việc phân loại và phân

tích dữ liệu giao dịch đã trở thành một thách thức lớn đối với các nhà nghiên cứu và nhà phát triển.

Trong ngữ cảnh này, các mô hình phân loại dựa trên luật kết hợp đã nổi lên như một phương pháp mạnh mẽ để giải quyết vấn đề phân loại dữ liệu giao dịch. Tuy nhiên, khi đối diện với tập dữ liệu lớn, các thuật toán phân loại này thường gặp phải những thách thức về hiệu suất và thời gian xử lý. Quá trình xây dựng và đánh giá mô hình phân loại kết hợp trở nên cồng kềnh và tốn kém, đồng thời cũng có thể mắc phải vấn đề về độ chính xác và khả năng áp dụng thực tiễn.

Vì vậy, trong nỗ lực tối ưu hóa hiệu suất và độ chính xác của các mô hình phân loại dựa trên luật kết hợp, công trình nghiên cứu này đề xuất một số phương pháp cải tiến. Các phương pháp này bao gồm việc tinh chỉnh thuật toán để tối ưu hóa quá trình rút trích luật, sử dụng kỹ thuật chọn lọc quy tắc quan trọng để tăng cường tính hiệu quả của mô hình, và tối ưu hóa các quy trình đánh giá mô hình để đảm bảo sự chính xác và đồng nhất của kết quả.

Bằng cách này, mục tiêu của công trình nghiên cứu không chỉ là cải thiện độ chính xác của các mô hình phân loại dựa trên luật kết hợp, mà còn là giảm thiểu chi phí thực hiện và tăng cường khả năng áp dụng thực tiễn của chúng trong các bối cảnh thực tế. Đặc biệt, việc áp dụng thành công các phương pháp cải tiến này có thể đem lại những lợi ích to lớn trong các lĩnh vực như quản lý rủi ro tài chính, phân tích thị trường, và xử lý dữ liệu lớn.

7. Nội dung

Phần mở đầu

1. Lý do chọn đề tài:

Trong thời đại hiện đại, máy tính và trí tuệ nhân tạo đã trở thành những công cụ quan trọng không thể thiếu trong nhiều lĩnh vực, từ y tế đến tài chính, từ tự động hóa đến quản lý dữ liệu. Trong lĩnh vực trí tuệ nhân tạo, học máy dựa trên luật (rule-based machine learning) là một trong những phương pháp phổ biến để xây dựng hệ thống thông minh.

Tuy nhiên, khi dữ liệu trở nên phức tạp và đa dạng, việc tạo ra các luật đơn lẻ để mô hình hóa một cách chính xác trở nên khó khăn và không hiệu quả. Đồng thời, việc kết hợp nhiều tập luật (rule sets) cũng đặt ra một thách thức lớn trong việc tối ưu hóa hiệu suất của hệ thống.

Vì vậy, tôi đã chọn đề tài nghiên cứu về "cải tiến thuật toán phân lớp kết hợp" với mong muốn tạo ra một phương pháp có thể tối ưu hóa và kết hợp hiệu quả các tập luật, giúp cải thiện đáng kể hiệu suất và chính xác của các hệ thống dựa trên luật trong học máy và trí tuệ nhân tạo. Qua đó, đề tài này không chỉ cho thấy được các cách tiếp cận làm tối ưu các tập luật trong quá trình tiền xử lý mà còn mang lại đóng góp quan trọng cho các ứng dụng trong nhiều lĩnh vực thực tiễn.

2. Lịch sử nghiên cứu vấn đề

Vấn đề cải tiến thuật toán phân lớp kết hợp đã thu hút sự quan tâm của nhiều nhà nghiên cứu trong lĩnh vực trí tuệ nhân tạo và học máy từ những năm đầu của thế kỷ 21. Trong quá khứ, các phương pháp tiếp cận ban đầu thường dựa trên việc sử dụng các thuật toán tối ưu hóa cổ điển, chẳng hạn như thuật toán di truyền (genetic algorithms) và tìm kiếm theo hướng đạo hàm (gradient-based search methods), để tìm ra tổ hợp tối ưu của các luật.

Tuy nhiên, những phương pháp đó thường gặp phải các vấn đề như việc rơi vào các cực tiểu cục bộ hoặc không khả thi khi xử lý các không gian tìm kiếm lớn hoặc không liên tục. Do đó, nghiên cứu trong lĩnh vực này tiếp tục phát triển với

việc áp dụng các phương pháp mới như học tăng cường (reinforcement learning) và học sâu (deep learning) để tối ưu hóa và kết hợp các tập luật một cách hiệu quả hơn.

3. Mục đích và nhiệm vụ nghiên cứu

Mục đích chính của nghiên cứu này là phát triển và áp dụng các phương pháp cải tiến thuật toán phân lớp kết hợp nhằm cải thiện hiệu suất và chính xác của các hệ thống dựa trên luật trong lĩnh vực trí tuệ nhân tạo và học máy. Mục tiêu cuối cùng là tạo ra các hệ thống thông minh linh hoạt và hiệu quả hơn trong việc đưa ra quyết định và dự đoán trong các ứng dụng thực tiễn.

Về nhiệm vụ nghiên cứu :

- Phân tích và Đánh giá Phương pháp Hiện có: Đầu tiên, nghiên cứu sẽ tiến hành phân tích và đánh giá các phương pháp hiện có trong tối ưu hóa và kết hợp các tập luật.
- Phát triển Phương pháp Mới: Tiếp theo, nghiên cứu sẽ tập trung vào việc phát triển các phương pháp mới và cải tiến để tối ưu hóa hiệu suất của hệ thống dựa trên luật. Điều này có thể bao gồm việc áp dụng các kỹ thuật, thuật toán khác vào.
- Thử nghiệm và Đánh giá: Sau khi phát triển các phương pháp mới, nghiên cứu sẽ tiến hành thử nghiệm và đánh giá chúng trên các tập dữ liệu thực tế. Quá trình này giúp đánh giá hiệu suất và tính khả thi của các phương pháp đề xuất so với các phương pháp hiện có.
- Ứng dụng và Triển khai: Cuối cùng, nghiên cứu sẽ xem xét cách áp dụng và triển khai các phương pháp cải tiến thuật toán phân lớp kết hợp vào các ứng dụng thực tế, như hệ thống hỗ trợ quyết định trong y tế, tài chính, hoặc tự động hóa công nghiệp.

4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu trong đề tài này bao gồm các hệ thống dựa trên luật trong lĩnh vực trí tuệ nhân tạo và học máy. Cụ thể, nghiên cứu tập trung vào việc tối ưu hóa và kết hợp các tập luật trong các hệ thống này. Đối tượng có thể bao gồm:

- Hệ thống hỗ trợ quyết định trong y tế, tài chính, hoặc quản lý dữ liệu.
- Hệ thống dự đoán và phân loại trong lĩnh vực máy học và thị giác máy tính.

Phạm vi của nghiên cứu được xác định bởi các khía cạnh sau:

- Phạm vi Địa lý: Nghiên cứu không giới hạn về địa lý và có thể áp dụng trên toàn thế giới, tùy thuộc vào việc áp dụng cụ thể của các phương pháp tối ưu hóa và kết hợp tập luật.
- Phạm vi Thời gian: Nghiên cứu tập trung vào việc phát triển các phương pháp mới và cải tiến trong thời gian gần đây, nhưng có thể áp dụng cho các hệ thống hiện đại và cũ hơn tùy thuộc vào tính ứng dụng của chúng.
- Phạm vi Ứng dụng: Nghiên cứu tập trung vào việc áp dụng các phương pháp cải tiến thuật toán phân lớp kết hợp cho các hệ thống dựa trên luật trong nhiều lĩnh vực ứng dụng, bao gồm như đã đề cập ở trên: y tế, tài chính, máy học, thị giác máy tính, công nghiệp và tự động hóa.

5. Phương pháp nghiên cứu

Về phương pháp nghiên cứu, đề tài nghiên về "cải tiến thuật toán phân lớp kết hợp" sẽ được tiếp cận trên các cơ sở sau :

- Tổng quan Lý thuyết: Bước đầu tiên trong phương pháp nghiên cứu là tiến hành một tổng quan về các lý thuyết và phương pháp liên quan đến tối ưu hóa tập luật và học máy. Điều này bao gồm việc nghiên cứu các công trình đã công bố, bài báo khoa học, và sách vở trong lĩnh vực này để hiểu rõ về các phương pháp hiện có và các thách thức đang tồn tại.
- Phát triển Phương Pháp: Tiếp theo, nghiên cứu sẽ tập trung vào việc phát triển các phương pháp mới và cải tiến để cải tiến thuật toán phân lớp kết hợp. Điều này có thể bao gồm việc áp dụng các kỹ thuật, thuật toán và sử dụng các công cụ và ngôn ngữ lập trình, thư viện phù hợp như SPMF sẽ được sử dụng trong công trình nghiên cứu này.
- Thử nghiệm và Đánh giá: Sau khi phát triển các phương pháp mới, nghiên cứu sẽ tiến hành thử nghiệm và đánh giá chúng trên các tập dữ liệu thực tế. Điều này bao gồm việc sử dụng các tập dữ liệu chuẩn hoặc tự thu thập dữ liệu để đánh giá hiệu suất và tính khả thi của các phương pháp đề xuất.

- So sánh và Phân tích Kết quả: Kế tiếp, nghiên cứu sẽ so sánh và phân tích kết quả của các phương pháp mới với các phương pháp hiện có để đánh giá tính hiệu quả và ưu điểm của chúng. Điều này có thể bao gồm việc đo lường hiệu suất, thời gian chạy, và sự ổn định của các phương pháp.
- Phân tích và Tổng hợp Kết quả: Cuối cùng, nghiên cứu sẽ phân tích và tổng hợp các kết quả thu được từ các thử nghiệm và đánh giá để rút ra các kết luận và đề xuất hướng phát triển tương lai trong lĩnh vực cải tiến thuật toán phân lớp kết hợp.

6. Những đóng góp mới của đề tài

Đề tài có thể đề xuất và phát triển các phương pháp tối ưu hóa mới và tiên tiến hơn để tối ưu hóa hiệu suất của các hệ thống dựa trên luật. Đề tài có thể tìm ra cách kết hợp các tập luật một cách hiệu quả hơn, giúp cải thiện chính xác và tính linh hoạt của các hệ thống dựa trên luật. Đề tài có thể đóng góp vào việc xây dựng cơ sở lý thuyết mới và phát triển các phương pháp mới để giải quyết các vấn đề thực tiễn trong lĩnh vực cải tiến thuật toán phân lớp kết hợp, đồng thời cung cấp các công cụ và kỹ thuật cho cộng đồng nghiên cứu. Đề tài có thể đánh dấu một bước tiến quan trọng trong việc áp dụng cải tiến thuật toán phân lớp kết hợp vào nhiều lĩnh vực ứng dụng khác nhau, từ y tế, tài chính, đến tự động hóa công nghiệp và quản lý dữ liệu. Cuối cùng, đề tài có thể đóng góp vào việc nâng cao hiệu quả và tính ứng dụng của các hệ thống thông minh trong thực tế, giúp tạo ra những ứng dụng thông minh và linh hoạt hơn trong nhiều lĩnh vực khác nhau.

7. Cấu trúc của đề tài

Ngoài phần mở đầu, kết luận, danh mục tài liệu tham khảo và phụ lục, đề tài gồm có 5 chương:

Chương 1: Giới thiệu Đề tài cải tiến thuật toán phân lớp kết hợp.

Chương 2: Giới thiệu Các công trình nghiên cứu liên quan.

Chương 3: Các hướng tiếp cận cải tiến thuật toán phân lớp kết hợp.

Chương 4: Cài đặt và thực nghiệm.

Phần nội dung:

Chương 1: Giới thiệu Đề tài cải tiến thuật toán phân lớp kết hợp

1.1. Khái niệm về luật kết hợp trong phân lớp

Trong lĩnh vực khai phá dữ liệu, tập luật kết hợp (associative rules)[1] là một khái niệm quan trọng, đóng vai trò trong việc phân tích mối quan hệ giữa các mục trong tập dữ liệu. Tập luật kết hợp là một phần của quy trình khai phá dữ liệu, bao gồm việc thu thập dữ liệu, xử lý và biến đổi dữ liệu, chọn lọc thuộc tính quan trọng và phân tích các mẫu và mối quan hệ trong dữ liệu.

Tập luật kết hợp giúp chúng ta hiểu rõ hơn về các mẫu xuất hiện đồng thời trong dữ liệu và từ đó có thể áp dụng để dự đoán, tối ưu hóa quy trình kinh doanh và đưa ra các quyết định chiến lược. Một tập luật kết hợp thường được biểu diễn dưới dạng "nếu A xảy ra, thì B cũng xảy ra" hoặc "nếu A và B xảy ra cùng một lúc, thì C cũng xảy ra". Ví dụ, trong một cửa hàng bán lẻ, một tập luật kết hợp có thể là "Nếu khách hàng mua sữa và bánh mì, họ cũng sẽ mua trứng".

Một trong những thuật toán phổ biến được sử dụng để khai phá tập luật kết hợp là thuật toán Apriori. Thuật toán này dựa trên một giả định quan trọng được gọi là "nguyên tắc Apriori", nó cho rằng mọi tập con của một tập phổ biến cũng phải phổ biến. Trong bất kì thuật toán mang tính chất luật kết hợp luôn phải tồn tại 2 loại giá trị là : Độ hỗ trợ (support) đo lường tần suất xuất hiện của một tập luật trong dữ liệu, ví dụ xét luật x -> y thì độ hổ trợ của x -> y = số lần xuất hiện cùng nhau của x và y. Độ tin cậy (confidence) đo lường mức độ chắc chắn của một tập luật, ví dụ xét luật a -> b thì độ tin cậy của a-> b = số lần xuất hiện cùng nhau của a và b chia cho tổng số lần xuất hiện của a.

Tập luật kết hợp đóng vai trò quan trọng trong việc phân tích dữ liệu và đưa ra quyết định thông minh bằng cách xây dựng các mô hình dư đoán trong nhiều lĩnh vực khác nhau. khái niệm về luật kết hợp là một phương pháp phân lớp mạnh mẽ dựa trên việc kết hợp các quy luật đơn lẻ để tạo ra dự đoán cuối cùng. Bằng cách sử dụng các mô hình phân lớp sử dụng luật kết hợp, chúng ta có thể khám phá và hiểu rõ hơn về các mối quan hệ ẩn sau dữ liệu, từ đó tạo ra giá trị và cơ hội mới cho doanh nghiệp và xã hội.

1.2. Hạn chế của thuật toán phân lớp kết hợp

Luật kết hợp, một phương pháp phổ biến trong việc phân lớp dữ liệu giao dịch, đã chứng tỏ sự hiệu quả của mình trong nhiều tình huống. Tuy nhiên, như mọi phương pháp, nó cũng tồn tại nhiều hạn chế. Sự đa dạng của các loại dữ liệu, sự không chắc chắn và mất mát thông tin, cùng với khả năng mô hình bị overfitting đều là những thách thức mà thuật toán phân lớp kết hợp phải đối mặt.

Do việc hoạt động theo cơ chế Apriori một cơ chế mang thiên hướng vét cạn với độ phức tạp của thuật toán lớn. Chính vì lẽ đó, trong những tình huống phải huấn luyện mô hình sử dụng luật kết hợp để phân lớp trên các tập dữ liệu lớn. Các thuật toán đó gặp rất nhiều khó khăn khi thời gian xử lý rất lâu và tiêu tốn một lượng lớn tài nguyên bộ nhớ. Và trên các tập dữ liệu cực kỳ phức tạ có nhiều thuộc tính, mỗi thuộc tính có sự phong phú và đa dạng về giá trị, việc xây dựng mô hình phân lớp dựa trên luật kết hợp gần như là việc bất khả thi.

Ngoài ra, độ chính xác của các thuật toán luật kết hợp cũng có thể không được đảm bảo. Điều này có thể xuất phát từ việc chúng dễ bị ảnh hưởng bởi dữ liệu nhiễu hoặc dữ liệu thiếu sót. Mặc dù các biện pháp như kiểm soát overfitting được thực hiện, nhưng vẫn có thể xuất hiện những quy tắc không chính xác hoặc không cần thiết, ảnh hưởng đến khả năng dự đoán chính xác của mô hình.

Đề tài nghiên cứu này nhằm mục tiêu đưa ra các phương pháp cải tiến để vượt qua các hạn chế của luật kết hợp trong việc phân lớp trên các dữ liệu giao dịch. Bằng cách tập trung vào việc tối ưu hóa quá trình kết hợp các luật, xử lý thông tin không chắc chắn một cách hiệu quả hơn, và giảm thiểu hiện tượng overfitting, chúng tôi hy vọng sẽ mang lại những tiến bộ đáng kể trong việc nâng cao độ chính xác và hiệu suất của các mô hình phân lớp dữ liệu giao dịch.

Chương 2: Giới thiệu Các công trình nghiên cứu liên quan.

2.1 Giới thiệu chung về các thuật toán phân lớp kết hợp

Kể từ khi khái niệm tập luật kết hợp được ra đời năm 1993 bởi tác giả Agrawal et al đã có rất nhiều công trình nghiên cứu liên quan, sử dụng ý tưởng về

luật kết hợp. Một số thuật toán phân lớp sử dụng ý tưởng luật kết hợp có thể kể đến như : ACCF[2], ACN[3], CBA[4],… trong đó có 1 thuật toán có tên là ACAC[5](ACAC: Associative classification based on all-confidence) của các tác giả Z. Huang, Z. Zhou, T. He, and X. Wang xuất bản năm 2011 sẽ được nói tới 1 cách chi tiết trong công trình nghiên cứu này.

Nhìn Chung, các thuật toán đó đều phải thỏa 2 yếu tố, sử dụng tính chất Apriori và sử dụng 2 ngưỡng giá trị là Độ hỗ trợ (support), Độ tin cậy (confidence). Điểm khác nhau giữa các thuật toán, là nằm ở việc sử dụng thêm 1 số ngưỡng giá trị khác hoặc sử dụng 1 số các hoạt động khác tạo nên điểm khác biệt giữa các thuật toán đó với nhau. Ví dụ thuật toán ACAC sẽ sử dụng thêm ngưỡng All-Confidence thay vì sử dụng ngưỡng giá trị Pearson correlation như trong bài báo ACN.

2.2 ACAC thuật toán đại diện cho họ thuật toán phân lớp bằng luật kết hợp.

Associative Classification based on All-Confidence(ACAC) là một phương pháp phân loại hiệu quả trong lĩnh vực học máy và khai phá dữ liệu. Cũng giống như các thuật toán khác trong họ CAR, ACAC phát triển từ khái niệm về tập luật kết hợp với tính chất apriori, AC-AllConf là một trong những thuật toán tiên tiến nhất trong việc xây dựng các mô hình phân loại từ dữ liệu có cấu trúc.

Thuật toán ACAC nổi bật với việc sử dụng chỉ số "All-Confidence", một phép đo đặc biệt đo lường mức độ chắc chắn của các tập luật kết hợp đối với tất cả các nhãn phân loại có thể. Điều này giúp ACAC xây dựng mô hình phân loại chính xác và đáng tin cậy, dựa trên các mối quan hệ phức tạp giữa các thuộc tính trong dữ liệu. Giá trị All-Confidence sẽ dược tính dựa trên công thức này:

$$\text{Allconf(X)} = \frac{\sup(X)}{\max(\sup(ai1),\ldots\ldots,\sup(aik))} \quad (1)$$

Một trong những điểm mạnh của ACAC là khả năng hiệu quả trong việc xử lý dữ liệu có cấu trúc phức tạp và đa dạng. Điều này làm cho ACAC trở thành

một công cụ hữu ích cho việc phân loại trong nhiều lĩnh vực, từ y tế đến tài chính, từ marketing đến dự báo xu hướng.

Thuật toán ACAC không chỉ cung cấp kết quả phân loại chính xác mà còn cho phép giải thích dễ dàng về quyết định phân loại được đưa ra. Điều này làm cho ACAC trở thành một công cụ hữu ích cho việc hiểu rõ hơn về dữ liệu và các quan hệ phức tạp giữa các thuộc tính.

Tóm lại, ACAC là một thuật toán phân loại mạnh mẽ và linh hoạt, mang lại kết quả chính xác và giải thích được, và có thể được áp dụng vào nhiều bài toán phân loại thực tế trong các lĩnh vực khác nhau. Đối với những nghiên cứu và ứng dụng trong học máy và khai phá dữ liệu, ACAC là một công cụ quan trọng và hiệu quả.

Cũng giống như các thuật toán, mô hình dư đoán sử dụng tập luật kết hợp để phân lớp khác khác. ACAC cũng có giai đoạn: giai đoạn đầu tiên thuật toán sẽ tạo ra tập luật dựa trên tập dữ liệu huấn luyện và giai đoạn sau sẽ dùng tập luật được tạo ra trước đó để đánh giá dự đoán, phân loại nhãn. Mỗi mốt giai đoạn sẽ được mô tả như sau thông qua mã giả trong bài báo An Efficient Algorithm That Optimizes The classification Association Rule Set như sau:

Với Giai đoạn 1 tạo ra tập luật sẽ được thực hiện trên thuật toán như sau:

❖ **ALGORITHM 1** ACAC-RG(T)

Require: Training data set; Minimum Support (min- Sup); Minimum confidence (minConf); Minimum all-confidence (minAllConf)

Ensure: R is set of rules for predicting class labels for objects

1: $C_1 \leftarrow init\_pass(T);$

2: ruleSelection (C_1, F_1, R)

3: ***for*** $k \leftarrow 2;\ F_{k-1} \neq 0;\ k++$ ***do***

4: $C_k \leftarrow candidateGen\ (F_{k-1});$

5: $supportCount(C_k);$

6: $ruleSelection(C_k, F_k, R);$

7: end for

8: **return** R

Với Giai đoạn 2 dư đoán, gán nhãn dựa trên các luật đã tìm được, ở bước này thuật toán ACAC sẽ sử dụng những công thức như sau trong quá trình tính toán dự đoán nhãn.

Tính toán giá trị Info(entropy) cho từng thuộc tính dựa trên công thức sau:

$$\text{Info}(X) = \frac{1}{log2k} \sum_{i=k}^{n} P(C_i|X)\log_2 P(C_i|X) \quad (2)$$

Trong đó k là số lớp, $P(C_i|X)$ là xác suất X phù hợp với C_i. Công thức entropy này được sử dụng để tính công thức (3). một công thức khác để tính toán xem các quy tắc sẽ phù hợp với nhãn nào:

$$S(r_i) = 0.9\left(1 - \frac{\sum \sup(Xi) Info(Xi)}{\sum \sup(Xi)}\right) + 0.1\frac{n}{ntot}$$

Trong đó X_i là tập quy tắc con của r_i ($r_i \in R$), n_{tot} là tổng của các quy tắc được gắn với đối tượng mới, n là số quy tắc trong tập R. Sau khi tính giá trị của $S(r_i)$, ACAC sử dụng nhóm có $S(r_i)$ cao nhất để gán nhãn cho dữ liệu mới. Việc phân loại quá trình theo công thức (3) được mô tả trong thuật toán ACClassifier.

❖ **ALGORITHM 2** ACPredict(R,D)

Input: R is the rule set obtains from ACAC-RG, Dnew (the new dataset),
Output: D' (the new dataset with label)
1: *D' = empty*
2: **for each** t_i in *D* **do**
3: v_1 = S(**R**, t_i, Yes),
4: v_2 = S(**R**, t_i, No),
5: **if** v1 > v2 **then**
6: t_i [Label] ← Yes //assigned Yes
7: **else**
8: t_i [Label] ← No //assigned No
9: **end if**
10: *D'*← *D'* + t_i
11: **end for**
12: **return** *D'*

2.3 Ví dụ minh họa về cách hoạt động thuật toán ACAC

Giả sử, ta có tập dữ liệu huấn luyện gồm có 8 dòng dữ liệu và 5 thuộc tính(cột) trong đó thuộc tính cuối(class) là thuộc tính nhãn. Trong ví dụ này, ta chọn ngưỡng hổ trợ(support threshold) = 2, ngưỡng allconfidence = 50%, và ngưỡng tin cậy(confidence threshold) = 100%. Để dễ trình này giá trị phần trăm sẽ được quy chiếu thành giá trị trong khoảng [0,1].

Bảng 2.1:Tập dữ liệu huấn luyện

A	B	C	D	class
32	55	80	83	90
33	52	80	85	89
33	52	80	85	89
33	55	79	82	90
34	55	79	82	89
34	55	77	82	89
32	55	80	88	90
33	55	79	82	90

Như đã nói trên thuật toán ACAC sẽ có 2 phần là chọn lọc luật và dự đoán, dựa trên đoạn mã giả thuật toán 1 đa sẽ có các bước thực hiện như sau:

- Bước đầu ta sẽ tìm tập phổ biến F1, R1 với F1 chứa các luật với tiền tố là 1 giá trị và vượt qua 2 ngưỡng độ hổ trợ (support) và All-Confidence những không vượt qua ngưỡng tin cậy (confidence). Trong khi đó R1 chứa các luật với tiền tố là 1 giá trị và vượt qua cả 3 ngưỡng.

Bảng 2.2:Tập ứng viên tiền tố 1 giá trị F1

item	class	condsup	rulesup	allconf	conf
33	90	4	2	100	50
33	89	4	2	100	50
55	90	6	4	100	33
55	89	6	2	100	67
79	90	3	2	100	67
79	89	3	1	100	33
80	90	4	2	100	50
80	89	4	2	100	50
82	90	4	2	100	50
82	89	4	2	100	50

Bảng 2.1:Tập luật R1

item	class	condsup	rulesup	allconf	conf
32	90	2	2	100	100
34	89	2	2	100	100
52	89	2	2	100	100
85	89	2	2	100	100

- Ở bước tiếp theo, với tập luật ứng viên F1 thu được, Thuật toán ACAC sẽ dùng các luật trong tập đó để sinh ra tập ứng viên F2, R2. F2 là tập chứa các luật với tiền tố là 2 giá trị và vượt qua 2 ngưỡng độ hổ trợ(support) và All-Confidence những không vượt qua ngưỡng tin cậy(confidence). Ví dụ tập F1 có 2 luật: 55 -> 90 có rulesup = 4 và79 -> 90 có rulesup = 2 sẽ được kết hợp để tạo nên luật : (55, 79) -> 90 với rulesup = 2, all-conf = 2/max(4,2) = 50% thỏa điều kiện được vào tập F2. Trong khi đó R2 chứa các luật với tiền tố là 2 giá trị và vượt qua cả 3 ngưỡng.

Bảng 2.2: Tập ứng viên tiền tố 2 giá trị F2

item	class	condsup	rulesup	allconf	conf
55 79	90	3	2	50	67
55 82	89	4	2	50	50
55 82	90	4	2	100	50
79 82	90	3	2	100	67

Bảng 2.3: Tập luật R2

item	class	condsup	rulesup	allconf	conf
33 55	90	2	2	100	100
33 79	90	2	2	100	100
33 80	89	2	2	100	100
33 82	90	2	2	100	100
33 80	90	2	2	100	100

Chương 3: Các hướng tiếp cận cải tiến thuật toán phân lớp kết hợp

Trong công trình nghiên cứu này sẽ có các chiến lược khác nhau nhằm cải thiện hiệu xuất của các thuật toán sử dụng luật kết hợp nói chung và thuật toán ACAC nói riêng. Chiến lượt đầu tiên sẽ là lựa chọn đặc trưng tốt nhất và loại bỏ đi các đặt trưng có chất lượng thấp bằng giá trị Entropy. Hai chiến lược tiếp sau đó, là sử dụng thuật toán leo đồi hoặc thuật toán luyện thép để tìm ra tập thuộc

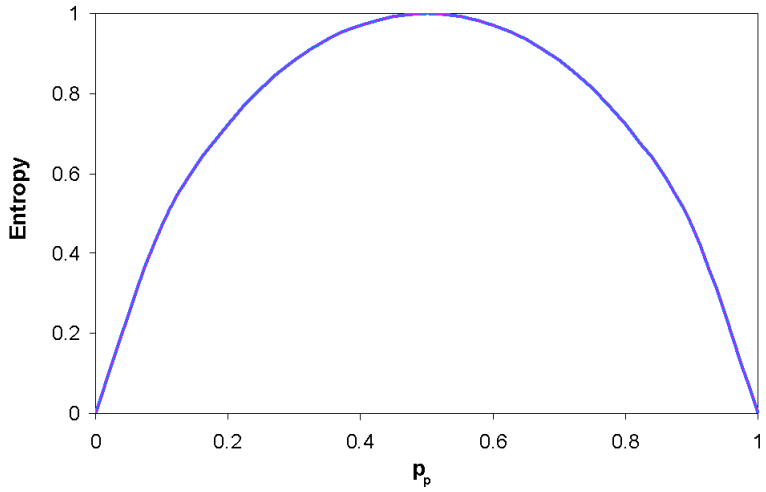
tính tốt nhất cho việc xây dựng mô hình phân lớp dựa trên kết quả từ chiến lượt đầu tiên. Và chiến lược cuối cùng trong công trình nghiên cứu này là sử dụng đa luồng xử lý trong việc tính toán các thông số trong việc đánh giá luật nhằm tăng tốc độ xử lý.

3.1 Cải thiện thuật toán phân lớp kết hợp ACAC bằng cách tiếp cận lựa chọn đặc trưng bằng ngưỡng giá trị entropy.

3.1.1 Giới thiệu giá trị Entropy.

Entropy là một khái niệm quan trọng trong lĩnh vực khoa học máy tính, đặc biệt là trong việc đo lường sự không chắc chắn hoặc độ không tự do của thông tin. Trong thống kê và lý thuyết thông tin, entropy được sử dụng để đo lường mức độ không chắc chắn của một biến ngẫu nhiên. Đối với hệ thống xử lý ngôn ngữ tự nhiên chẳng hạn, entropy có thể được sử dụng để đánh giá độ đa dạng của văn bản hoặc đo lường sự hỗn loạn của một từ trong một ngữ cảnh nhất định.

Entropy cũng là một thành phần quan trọng trong các thuật toán học máy, như trong việc xây dựng các cây quyết định hoặc các mô hình học tập có giám sát khác. Trong các thuật toán này, entropy thường được sử dụng để đo lường sự thuần túy hoặc đồng nhất của các nhóm dữ liệu, giúp tối ưu hóa quá trình phân loại hoặc dự đoán.



Hình 3.1: Đồ thị mô tả miền giá trị Entropy(trục tung) và xác suất phân loại tương ứng với từng giá trị entropy (trục hoành).

Dựa vào hình minh họa trên ta thấy, miền giá trị của Entropy thuộc [0,1]. Càng gần giá trị 0, thuộc tính cho thấy xác suất ra 1 trong 2 nhãn cao. Ngược lại, giá trị Entropy càng gần 1, thì xuất suất ra 1 trong 2 nhãn của thuộc tính là không chắc chắn gần như xác suất là 50-50. Tóm lại, entropy là một khái niệm quan trọng trong khoa học máy tính nói chung, và việc xây dựng mô hình dự đoán bằng luật kết hợp nói riêng. Đóng vai trò quan trọng trong việc đo lường sự không chắc chắn và tự do của thông tin, cũng như trong việc tối ưu hóa các quy trình xử lý dữ liệu và học máy.

3.1.2 Sử dụng Entropy trong việc lựa chọn đặc trưng trên thuật toán phân lớp ACAC.

Với ý nghĩa của giá trị Entropy, công trình nghiên cứu này sẽ sử dụng nó cho giai đoạn tiền xử lý dữ liệu đầu vào nhằm loại bỏ đi các thuộc tính không có đóng góp nhiều trong việc dự đoán nhãn sau khi huấn luyện mô hình, qua đó sẽ giảm đáng kể thời gian xử lý cũng như tăng độ chính xác của mô hình, kỹ thuật này còn được là chọn lọc đặt trưng hay trong tiếng anh gọi là feature selection(FS). Công trình này đưa ra thuật toán FSAC trong bài báo An Efficient Algorithm That Optimizes The Classification Association Rule Set[6] để kết hợp ngưỡng giá trị entropy vào thuật toán ACAC như sau:

❖ **ALGORITHM 3** FSAC

1: minSup, minConf, minAllConf;
2: D: dataset (D_{train}, D_{test});
3: e ≔ entropy threshold;
4: D_e ≔ attrribute a_i from D_{train} where entropy(a_i, label) ≤ e;
5: neighbor = [e,e ± Δ];
6: e' ≔ random(neighbor);
7: $D_{e'}$ ≔ attrribute a_i from D_{train} where entropy(a_i,label) ≤ e;
8: **repeat**
9: $model_e$ ≔ ACAC-RG(D_e);
10: $accuracy_e$ ≔ $model_e$.predict(D_{test});
11: neighbor = [e,e ± Δ];
12: e' ≔ random(neighbor);
13: $model_{e'}$ ≔ ACAC-RG(D_e);

14: $accuracy_{e'} := model_{e'}.predict(D_{test})$;
15: **if** ($accuracy_e \leq accuracy_{e'}$) **then**
16: e:=e'
17: **end if**
18: ***until*** $accuracy_e \leq accuracy_{e'}$
19: ***return*** $model_e$

Thuật toán FSAC tối ưu hóa ACAC bằng cách loại bỏ các thuộc tính dư thừa, dòng 13 là ACAC gốc thuật toán, dòng 14 tính toán độ chính xác của mô hình thu được từ dòng 13. Mã từ dòng 8 đến dòng 17 lặp lại công việc cải tiến cho đến khi độ chính xác của mô hình đạt mức tối ưu. Trong mỗi lần lặp, dựa trên entropy hiện tại, chúng ta chọn entropy lân cận (dòng 11, 12) e', thuật toán xác định tập tính năng De bao gồm các thuộc tính có entropy liên quan đến thuộc tính đích nhỏ hơn e'. Nói cách khác, thuật toán cố gắng loại bỏ thuộc tính dư thừa khỏi tập huấn luyện trong mỗi lần lặp. Dòng 15 kiểm tra xem có loại bỏ đúng, nếu đúng thì cập nhật giá trị tốt hơn cho entropy hiện tại. Thuật toán sẽ tìm ra kết quả tốt nhất mô hình ở ngoài vòng lặp (bộ phân loại luật kết hợp tối ưu) như dòng 18. Ở dòng 2, D là tập dữ liệu được chia thành hai nhóm Dtrain và Dtest theo tỷ lệ 80/20. Dựa trên Dtrain, chúng tôi tìm thấy bộ phân lớp luật kết hợp và Dtest được sử dụng để kiểm tra độ chính xác của bộ phân loại luật kết hợp.

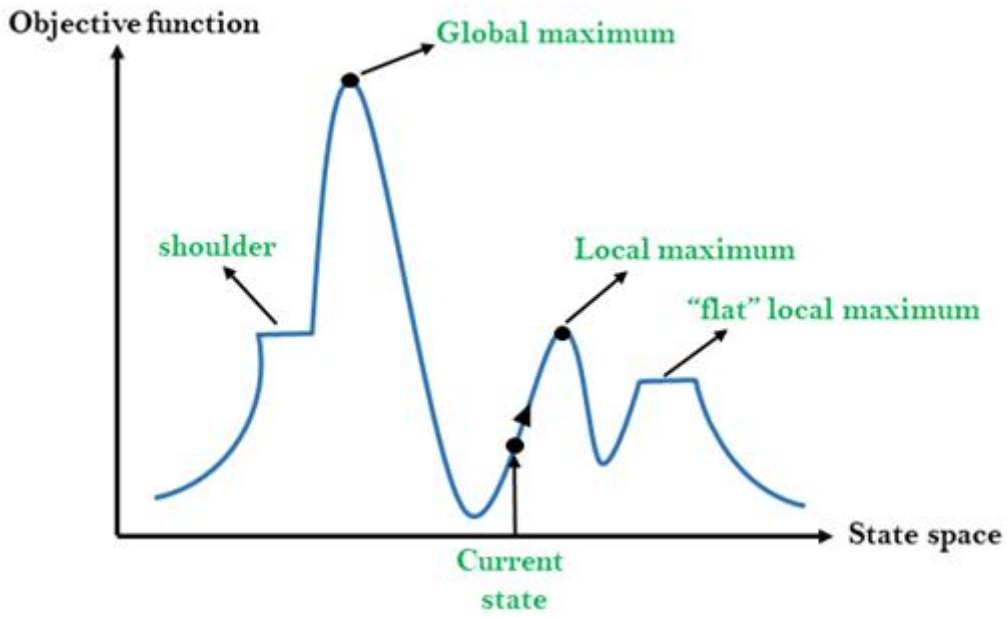
3.2 Sử dụng thuật toán leo đồi trong việc cải thiện thuật toán phân lớp kết hợp ACAC

3.2.1 Giới thiệu thuật toán leo đồi.

Thuật toán leo đồi[7], hay còn gọi là thuật toán hill climbing, là một phương pháp tìm kiếm cục bộ được sử dụng rộng rãi trong giải quyết các bài toán tối ưu hóa. Ý tưởng cơ bản của thuật toán này là khám phá và di chuyển từng bước trong không gian tìm kiếm, mục tiêu là tìm ra giải pháp tốt nhất trong khu vực lân cận của giải pháp hiện tại.

Thuật toán bắt đầu bằng việc chọn ngẫu nhiên một giải pháp khởi đầu, sau đó lặp đi lặp lại quá trình cải thiện giải pháp này bằng cách di chuyển đến các giải pháp lân cận có giá trị hàm mục tiêu cao hơn. Quá trình này được tiếp tục cho đến

khi không có giải pháp lân cận nào cải thiện được nữa, khi đó thuật toán kết thúc và trả về giải pháp tốt nhất mà nó đã tìm thấy.



Hình 3.2: Hình minh họa thuật toán leo đồi

Tuy thuật toán leo đồi có thể dễ dàng bị mắc kẹt ở điểm cực tiểu cục bộ và không thể tìm ra giải pháp tối ưu toàn cục, nhưng nó vẫn được ứng dụng rộng rãi trong các bài toán có không gian tìm kiếm lớn và phức tạp. Đồng thời, nhiều biến thể và kỹ thuật kết hợp được phát triển để cải thiện hiệu suất của thuật toán này trong các bài toán khác nhau.

3.2.2 Kết hợp thuật toán leo đồi vào thuật toán ACAC.

Dựa trên ý tưởng về việc chọn lọc, thay thế trong mô hình quản lý tìm ra đội hình mạnh nhất trong các môn thể thao tập thể nói chung. Kết hợp với ý tưởng của thuật toán leo đồi, phần sau của công trình nghiên cứu đưa ra cách tiếp cận dựa trên kết quả thu được trong các tiếp cận ở chương 1. Cách tiếp cận này sẽ có 3 bước như sau:

- Bước 1: chọn 1 ngưỡng Entropy loại bỏ các thuộc tính không có đóng góp trong việt phân loại nhãn. Sau đó chọn 1 ngưỡng giá trị entropy để chia tập dữ liệu đó thành 2 tập là tập tốt nhất (bestAttr) và tập dự bị (subAttri).

- Bước 2: Sử dụng thuật toán leo đồi vào thuật toán ACAC trên tập thuộc tính bước thu được ở bước 1. Bằng cách chọn số lượng ngẫu nhiên thuộc tính trong tập tốt nhất, với số lượng gọi là delta trong tập hiện tại và đổi với delta thuộc tính của tập dự bị, lập đi lập lại cho đến khi không tìm được tập có độ chính xác tốt nhất. Điều này đã dược bài báo A meta-heuristic approach for enhancing performance of associative classification[8] trình bày 1 cách rất chi tiết và đây là đoạn mã giá minh họa cho ý tưởng này.

❖ **ALGORITHM 2** Phase2.1(Hill Climbing)

Input: The training dataset, initValid, delta
Output: bestAttr(all best attributes)

```
1: for i ← 0; i ≤ n; i + + do
2:     newAttr ≔ GenAttributes(bestAttr, subAttr, delta);
3:     validation ≔ AC(new Attr);
4:     if validation > initValid thenif
5:         initValid ≔ validation;
6:         i ≔ 0;
7:         bestAttr ≔ new Attr
8:         subAttr ≔ allAttr - bestAttr
9:     end if
10: end for
11: return bestAttr
```

- Bước 3: kết hợp 1 thuộc tính trong tập còn lại vào tập sau bước 2.

3.3 Sử dụng thuật toán luyện thép trong việc cải thiện thuật toán phân lớp kết hợp ACAC.

3.3.1 Giới thiệu thuật toán luyện thép.

Như đã đề cập về điểm hạn chế của thuật toán leo đồi ở phần trước, thuật toán leo đồi có thể dễ dàng bị mắc kẹt ở điểm cực tiểu cục bộ và không thể tìm ra giải pháp tối ưu toàn cục, nhưng nó vẫn được ứng dụng rộng rãi trong các bài toán có không gian tìm kiếm lớn và phức tạp. Chính vì thế Chương 4 của công trình nghiên cứu này sẽ đề cập đến 1 thuật toán luyện thép[9].

Thuật toán luyện thép, hay Simulated Annealing, là một phương pháp tìm kiếm cục bộ được lấy cảm hứng từ quá trình làm nóng và làm nguội của kim loại

trong công nghệ luyện thép. Mục tiêu của thuật toán này là cải thiện điểm yếu của thuật toán leo đồi bằng cách chấp nhận các giải pháp kém hơn một cách tạm thời, để tránh bị mắc kẹt ở các điểm cực tiểu cục bộ và tìm ra giải pháp tối ưu toàn cục.

Thuật toán luyện thép hoạt động dựa trên một nguyên lý chính: trong quá trình tìm kiếm, nó có thể chấp nhận các giải pháp có giá trị hàm mục tiêu kém hơn giải pháp hiện tại một cách ngẫu nhiên. Việc này giúp thuật toán thoát khỏi các điểm cực tiểu cục bộ và có khả năng khám phá không gian tìm kiếm rộng hơn.

Thuật toán luyện thép bắt đầu với một nhiệt độ ban đầu cao, tương ứng với khả năng chấp nhận các giải pháp xấu. Sau mỗi bước lặp, nhiệt độ được giảm dần theo một lịch trình xác định trước, điều này làm giảm khả năng chấp nhận các giải pháp xấu. Quá trình này tiếp tục cho đến khi nhiệt độ giảm về mức thấp nhất, khi đó thuật toán kết thúc và trả về giải pháp tốt nhất mà nó đã tìm thấy.

Bằng cách kết hợp việc chấp nhận các giải pháp xấu tạm thời và giảm dần khả năng này theo thời gian, thuật toán luyện thép đã chứng tỏ hiệu suất tốt trong việc tìm kiếm giải pháp tối ưu toàn cục cho các bài toán tối ưu hóa.

3.3.2 Kết hợp thuật toán luyện thép vào thuật toán ACAC.

Như đã nói Thuật toán luyện thép là 1 dạng cải tiến của thuật toán leo đồi nên cách kết hợp thuật toán luyện theo vào thuật toán ACAC cũng sẽ được thực hiện gần như tương tự. Phần sau của công trình nghiên cứu mô tả cách tiếp cận dựa trên kết quả thu được trong các tiếp cận ở chương 1. Cách tiếp cận này sẽ có 3 bước như sau:

- Bước 1: chọn 1 ngưỡng Entropy loại bỏ các thuộc tính không có đóng góp trong việt phân loại nhãn.
- Bước 2: Ở bước này sẽ thực hiện giống với các tiếp cận sử dụng thép. Điều này cũng đã dược bài báo A meta-heuristic approach for enhancing performance of associative classification trình bày 1 cách rất chi tiết và đây là đoạn mã giả minh họa cho ý tưởng này. Thuật toán leo đồi vào ACAC chỉ khác là ở bước này thuật toán được sử dụng là luyện.

❖ **ALGORITHM 3** Phase2.2 (Simulated Annealing)

Input: The training dataset, initValid, delta, T, k

Output: bestAttr

```
1: for i ← 0; i ≤ n; i++ do
2:     new Attr := GenAttributes(bestAttr, subAttr, delta);
3:     validation := AC(new Attr);
4:     if validation > initValid then
5:         initValid := validation;
6:         bestAttr := new Attr;
7:         subAttr := allAttr - bestAttr;
8:         i := 0;
9:     else
10:        Δf = | validation - initValid |
11:        r := random r(0,1)
12:        if r > exp(-Δf / kT) then
13:            initValid := validation;
14:            bestAttr := new Attr
15:            subAttr := allAttr - bestAttr
16:            i := 0;
17:        end if
18:    end if
19: end for
20: return bestAttr
```

- Bước 3: kết hợp 1 thuộc tính trong tập còn lại vào tập thuộc tính trước đó

3.4 Cải thiện thuật toán phân lớp kết hợp ACAC bằng các tiếp cận đa luồng xử lý.

3.4.1 Khái niệm đa luồng xử lý

Kế thừa từ các ý tưởng về hướng tiếp cận đã được đề cập trước đó. Các cách tiếp cận chú trọng việc loại bỏ bớt các thuộc tính không có đóng góp trong quá trình dự đoán nhãn qua đó giúp cải thiện tốc độ xử lý và độ chính xác của mô hình sử dụng luật kết hợp. Ở cách tiếp cận cuối cùng của công trình nghiên cứu này, chúng tôi đưa ra 1 cách tiếp cận sử dụng đa luồng xử lý trong việc tính toán từng luật.

Việc sử dụng đa luồng xử lý trong thuật toán phân loại theo luật kết hợp không chỉ là một tiến bộ đáng kể mà còn là một chiến lược thông minh để tăng tốc độ xử lý. Bằng cách này, chúng ta có thể tận dụng hiệu suất của các bộ xử lý đa nhân (multicore processors) và thậm chí là các hệ thống phân tán.

Khi áp dụng đa luồng, thuật toán có thể chia nhỏ các nhiệm vụ phân loại thành các tác vụ nhỏ hơn và thực hiện chúng đồng thời trên nhiều luồng khác nhau. Điều này giúp tận dụng tối đa nguồn lực xử lý có sẵn và giảm thời gian chờ đợi giữa các bước tính toán.

Hơn nữa, việc phân tán tính toán qua nhiều luồng cũng giúp tránh được các chướng ngại về khả năng xử lý một lượng lớn dữ liệu. Bằng cách này, chúng ta có thể xử lý các tập dữ liệu lớn một cách hiệu quả hơn mà không lo lắng về việc gây tắc nghẽn hoặc quá tải hệ thống.

3.4.2 Sử dụng Đa luồng xử lý trên các thuật toán phân lớp

Thay vì chỉ sử dụng thuật toán ACAC để áp dụng chiến lược cải thiện độ chính xác như các phần trên, ở cách tiếp cận này sẽ sửa dụng đa luồng xử lý nhiều vài thuật toán sử dụng ý tưởng luật kết hợp.

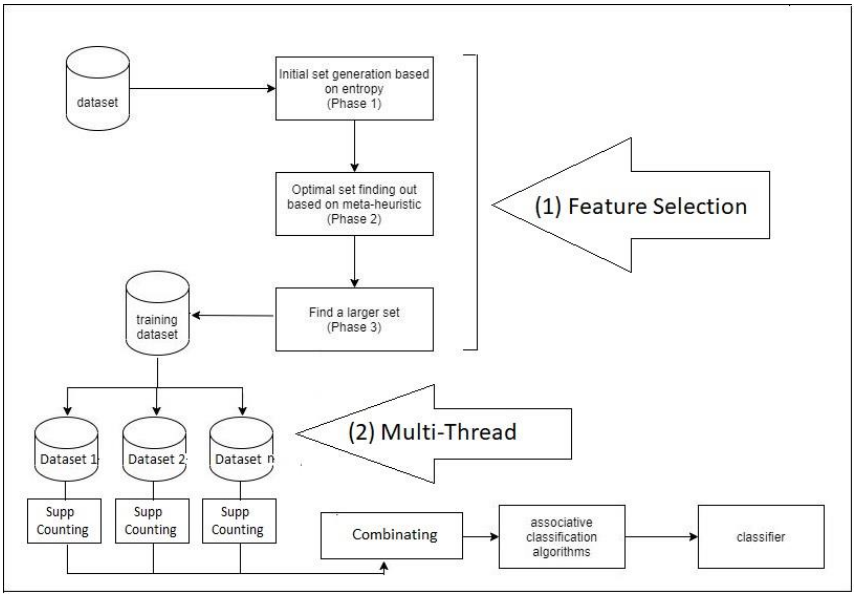
Bảng 3.1: Mô tả việc chưa/ đã áp dụng kỹ thuật lựa chọn đặc trưng trên các thuật toán luật kết hợp trên tập dữ liệu Mushroom

	ACAC	CBA	CMAR
Độ chính xác	0.5429/0.9951	0.9828/0.9397	0.8012/0.8816
Số luật	55/9	22/8	453/14
Thời gian(ms)	1120/169	945343/146	1500/200

Bảng 3.2: Mô tả việc chưa/ đã áp dụng kỹ thuật lựa chọn đặc trưng trên các thuật toán luật kết hợp trên tập dữ liệu Mail Spam

	ACAC	CBA	CMAR
Độ chính xác	NA/0.9305	NA/0.848	NA/0.6916
Số luật	NA/278	NA/184	NA/158
Thời gian(ms)	NA/13292	NA/146	NA/200

Nhằm kế thừa kết quả của các cách tiếp cận trước đó Các tiếp cận đa luồng xử lý này sẽ đươc kết hợp vào thuật toán ACAC sau khi tìm ra được tập thuộc tính tốt nhất.



Hình 3.3: Biểu đồ mô tả cách kết hợp đa luồng xử lý vào thuật toán ACAC

Thuật toán ruleSelection(C_k,F_k,R) trong thuật toán ACAC gốc sẽ được thay đổi như sau khi sử dụng các tiếp cận đa luồng xử lý:

❖ **ALGORITHM 1** ruleSelection (C_k,F_k,R)

1: **for** *all candidate* c ∈ C_k **do**
2: **for** *all* class[i] **do**
3: compute c.allconf[i], c.conf[i];// parallelize by using MultithreadAC
4: **if** c.condsup/|T| ≥ minsup and c.allconf[i] ≥ minallconf **then**
5: **if** c.conf[i] ≥ minconf **then**
6: *push* c → class[i] into R
7: **else**
8: *push c* into F_k
9: **end if**
10: **end if**
11: **end for**
12:**end for**

❖ **ALGORITHM 2** MultithreadAC

Input: rule(rule), subDatasets[n] (n sub-dataset was divided), n(number of threads)
Output: ruleDone(rule after evaluation)

1: ruleEvaluated ≔ []
2: **for** *i* ← *0; i* ≤ *n*−*1; i*++ **do**
3: ruleEvaluated ≔ ruleEvaluated **U** Evaluation(rule, subDataset[i]);
4: **end for**
5: ruleDone ≔Aggregating(ruleEvaluated);
6: **return** ruleDone;

❖ **ALGORITHM 3** Aggregating

Input: ruleEvaluated[n] (Evaluation of rule on each sub-dataset)
Output: evaluated of rule after combinating

1: rule ≔ ruleEvaluated[0];
2: **for** *i* ← *1; i* ≤ *n - 1;i*++ **do**
3: rule ≔ rule + ruleEvaluated[i];
4: **end for**
5: **return** rule;

Bản chất cách tiếp cận đa luồng xử lý này chỉ đơn giản là chia nhỏ quá trình tính toán của từng luật, mỗi khi xét 1 luật sẽ chia bộ xử lý từ 1 thành n bộ xử lý, sau đó khi cả n bộ xử lý hoàn thành sẽ được tích hợp lại qua thuật toán số 3 Aggregating trên để kết hợp lại.

Ví dụ về cách tiếp cận này với bảng 1.1, với cách tiếp cận ACAC truyền thống để xét 1 thuật có tiền tố 1 giá trị A -> B thì sẽ mất 8 bước làm nhưng với các tiếp cận đa luồng xử lý thì thời gian thực hiện của mỗi luật sẽ được thực hiện như sau. Tập dữ liệu trong bảng 1.1 được chia thành 3 phần. Phần 1 có 3 dòng, phần 2 có 3 dòng, phần 3 có 2 dòng. Chi phí để quét dữ liệu thay vì 8 như tuần tự, theo cách đa luồng sẽ là max(3,3,2) = 3 + chi phí tạo luồng xử lý. Với những tập dữ liệu có nhiều dòng dữ liệu cách tiếp cận này sẽ cực kì hiệu quả.

Chương 4: Thực nghiệm và cài đặt

4.1 Giới thiệu môi trường, công cụ và dữ liệu thực nghiệm.

Ở công trình nghiên cứu này, mọi thực nghiệm sẽ được thực hiện trên thư viên SPMF dựa trên ngôn ngữ JAVA. Môi trường thí nghiệm được xử lý tập trung trên các máy tính có cấu hình Intel(R), CPU Core(TM) i7-6820HQ @ 2.70GHz

(8CPU), 2.7GHz và hệ điều hành Windows 10 hệ thống. Mỗi cách tiếp cận cải tiến thuật toán phân lớp ACAC sẽ dược thực hiện trên các bộ dữ liệu khác nhau. Chung quy lại sẽ có tổng cộng 5 tập dữ liệu được sử dụng trong tất cả giai đoạn thực nghiệm. Tỉ lệ chia dữ liệu cho tập huấn luyện và tập đánh giá trên mỗi tập dữ liệu sẽ được chia theo tỉ lệ tiêu chuẩn là 80% dòng dữ liệu đầu tiên sẽ dùng cho việc huấn luyện, 20% còn lại cho phần đánh giá.

Tập dữ được nói đến đầu tiên là tập dữ liệu Mushroom[10]. Bởi vì dữ liệu này cũng được sử dụng bởi bản gốc Thuật toán ACAC và cũng là tập dữ liệu kinh điện cho các mô hình, thuật toán phân lớp bằng luật kết hợp. Tập dữ liệu này có 8125 dòng và 23 cột. Trong tập dữ liệu, thuộc tính đầu tiên là thuộc tính nhãn đươc đặt tên là “class” để phân loại nấm độc hoặc không độc (ăn được=e, độc=p). Trong 22 thuộc tính còn lại có rất nhiều các thuộc tính không cần thiết trong phân loại, và chúng cần được loại bỏ. Tập dữ liệu này sẽ được dùng trong phần thực nghiệm của các cách tiếp cận đã nêu trên.

Phần thực nghiệm còn được triển khai thử nghiệm trên tập dữ liệu phức tạp hơn là Email Spam [11], có 4602 dòng và 58 cột. Trong đó thuộc tính cuối cùng là thuộc tính nhãn có tên 'spam' để phân loại email là spam hoặc ham (spam=1, ham=0). Tập dữ liệu này sẽ được dùng trong phần thực nghiệm của các cách tiếp cận đã nêu trên.

Tập dữ liệu tiếp theo được sử dụng là Default of credit card clients[12]. Tập dữ liệu này có 24 cột và 30000 dòng. So với 2 tập dữ liệu liệu vừa đề cập trên, tập dữ liệu này có số lượng gấp khoảng 5 lần 2 tập dữ liệu trên hứa hẹn sẽ có đóng góp lớn. Tập dữ liệu này sẽ được dùng trong phần thực nghiệm của cách tiếp cận Đa luồng xử lý trong thuật toán phân lớp bằng luật kết hợp.

Tập dữ liệu cuối cùng được sử dụng trong phần thực nghiệm này là Smoking and Drinking Dataset with body signal [13]. Tập dữ liệu này có 24 thuộc tính và 1 triệu dòng dữ liệu. Thuộc tính nhãn có tên là “default payment next month” để phân loại khách hàng sử dụng thẻ tín dụng là 0 hoặc 1 (Yes = 1, No = 0). Nếu so về số lượng dòng dữ liệu với 3 tập dữ liệu trên tập dữ liệu này cho thấy

sự vượt trội. Tập dữ liệu này sẽ được dùng trong phần thực nghiệm của cách tiếp cận Đa luồng xử lý trong thuật toán phân lớp bằng luật kết hợp.

4.2 Thực nghiệm cách tiếp cận lựa chọn đặc trưng bằng ngưỡng giá trị entropy

4.2.1 Lựa chon đặt trưng trên tập dữ liệu Mushroom.

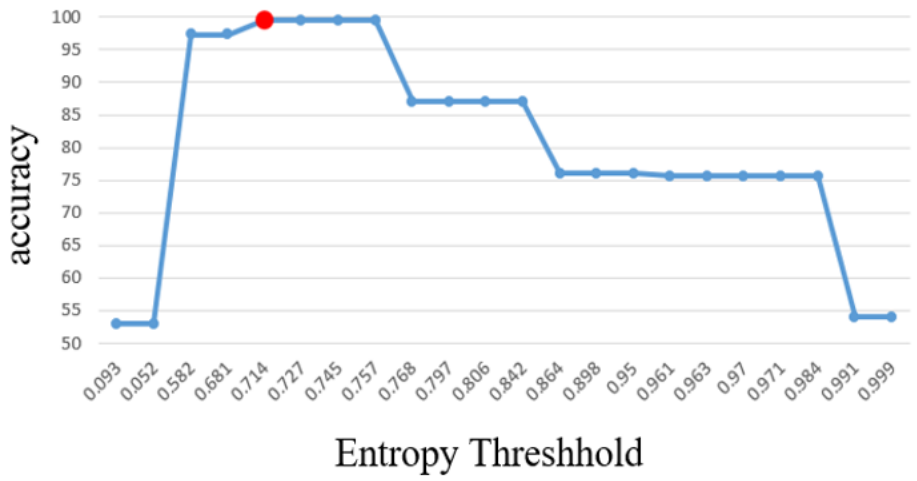
Dựa trên thuật toán FSAC đã nêu trên, Nếu độ chính xác tăng thì việc loại bỏ thuộc tính là chính xác và việc này được thực hiện cho đến khi độ chính xác đạt cao nhất giá trị. Nói cách khác, quá trình sẽ dừng khi độ chính xác bị giảm đi. Để so sánh một cách công bằng với ACAC, bài báo này cũng sử dụng cùng ngưỡng (minsup = 0.1, minconf = 0.8, minallconf = 0.5) được sử dụng trên ACAC gốc. FSAC cải thiện ACAC bằng cách sử dụng entropy để loại bỏ các thuộc tính không cần thiết trên tập dữ liệu Mushroom. Bảng 4.1 tại ngưỡng entropy = 0.714 cho thấy thuật toán FSAC đã loại bỏ 17 thuộc tính dư thừa (không cần thiết cho việc phân loại), giảm bộ quy tắc ban đầu từ 55 quy tắc xuống còn 9 quy tắc, và thời gian xử lý nhanh hơn đáng kể (131 ms so với 2384 ms). Đặc biệt độ chính xác phân loại tăng lên đáng kể (54,28% so với 99,51%). Người đọc có thể làm, từng bước một cách đơn giản thí nghiệm giảm entropy như thể hiện trong bảng 4.1.

Bảng 4.1: Giá trị entropy của từng thuộc tính trên tập dữ liệu Mushroom

Num	Tên thuộc tính	entropy	Độ chính xác(%)
1	odor	0.093	52.98
2	spore-print	0.52	52.98
3	gill-color	0.582	97.35
4	ring-type	0.681	97.35
5	**stalk-surf-abo**	**0.714**	**99.51**
6	stalk-surf-bel	0.727	99.51
7	stalk-color-abo	0.745	99.51
8	stalk-color-bel	0.757	99.51
9	gill-size	0.768	87.08
10	population	0.797	87.08
11	bruises	0.806	87.08
12	habitat	0.842	87.08
13	stalk-root	0.864	76.06
14	gill-spacing	0.898	76.06
15	cap-shape	0.95	76.06
16	ring-number	0.961	75.57

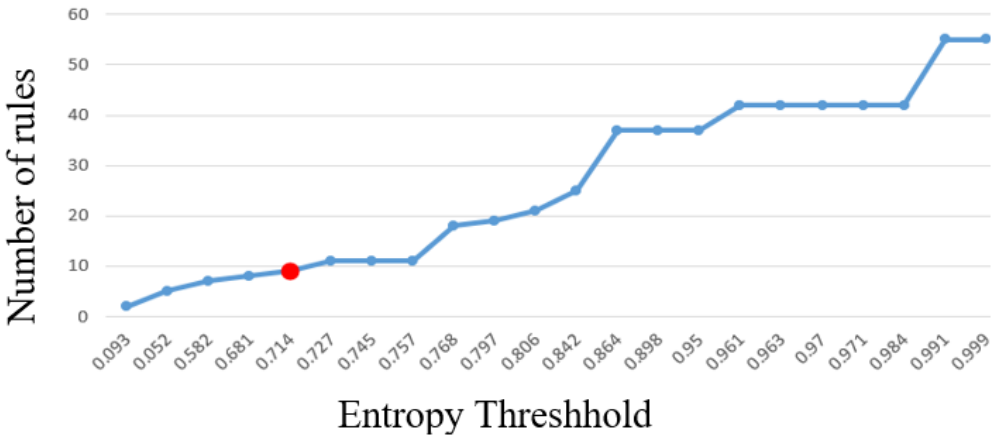
17	cap-color	0.963	75.57
18	cap-surface	0.97	75.57
19	veil-color	0.971	75.57
20	*gill*	0.984	75.57
21	stalk-shape	0.991	54.28
22	veil-type	0.999	54.28

Từ kết quả ở Bảng 4.1, ta có đồ thị là hình 4.1. Trong biểu đồ này, trục tung biểu thị độ chính xác, trục hoành biểu thị entropy của mỗi thuộc tính với thuộc tính đích. Đồ thị có điểm tối đa của độ chính xác phân loại là 99,51% từ điểm entropy là 0,757, tương ứng với 8 thuộc tính. Tuy nhiên, thuật toán được chạy liên tục với các giá trị entropy tương ứng (0,745, 0,727, 0,714) để loại bỏ thêm 3 thuộc tính và phân loại giá trị vẫn không giảm. Độ chính xác của việc phân loại chỉ giảm khi loại bỏ thêm một thuộc tính. Vậy thuật toán dừng ở giá trị entropy là 0,714 và có độ chính xác phân loại cao nhất của 99,51%.

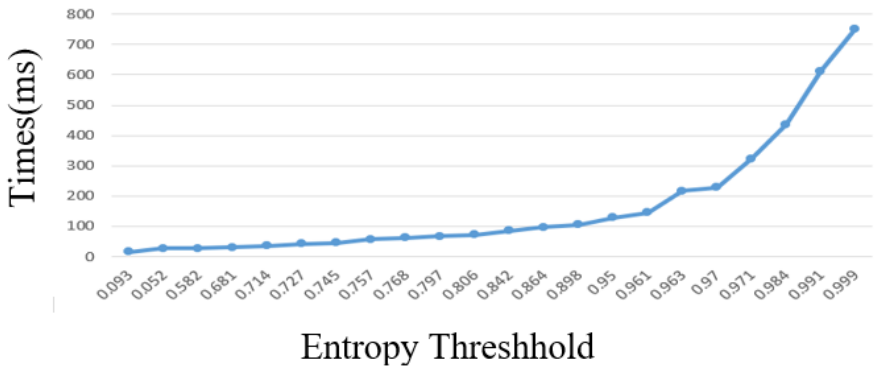


Hình 4.1: Biểu đồ mô tả độ chính xác của thuật toán ACAC + FS theo từng ngưỡng entropy trên tập dữ liệu Mushroom.

Cũng trong thí nghiệm này ở Bảng 4.1, chúng ta có đồ thị là Hình 4.2. Trong biểu đồ này, trục tung mô tả số quy tắc được sử dụng để phân loại, theo chiều ngang trục biểu thị entropy của từng thuộc tính so với thuộc tính đích. Biểu đồ cho thấy càng nhiều Các thuộc tính dư thừa trong tập huấn luyện sẽ bị loại bỏ (thuộc tính không cần thiết để phân loại), số các quy tắc trong tập phân lớp luật kết hợp được giảm bớt nhiều hơn. Đồ thị trong Hình 4.3 có trục tung mô tả thời gian xử lý và trục hoành biểu thị entropy của từng thuộc tính tương ứng với thuộc tính đích. Cũng trong đồ thị này, thời gian xử lý cũng đáng kể giảm.



Hình 4.2: Biểu đồ mô tả số lượng luật của thuật toán ACAC + FS theo từng ngưỡng entropy trên tập dữ liệu Mushroom.



Hình 4.3: Biểu đồ mô tả tốc độ xử lý của thuật toán ACAC theo từng ngưỡng entropy trên tập dữ liệu Mushroom.

4.2.2 Lựa chon đặt trưng trên tập dữ liệu EmailSpam.

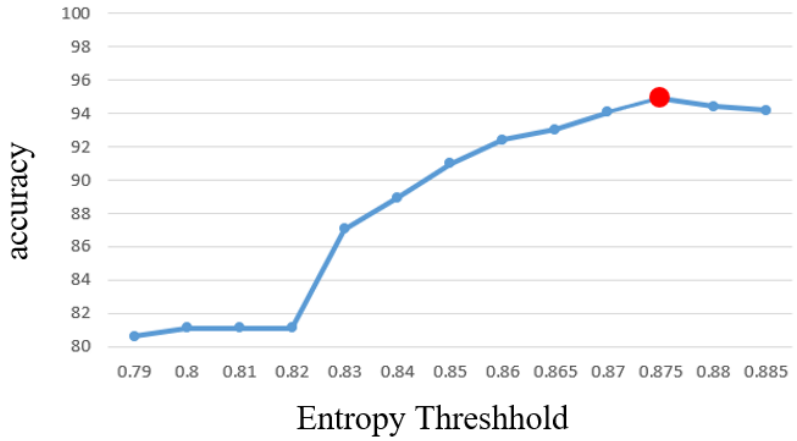
Việc kết hợp ngưỡng Entropy vào thuật toán ACAC trên tập dữ liệu Mushroom đã cho thấy sự cải thiện hiệu suất vượt trội của thuật toán ACAC. Tuy nhiên, Mushroom vẫn là một dữ liệu nhỏ so với nhiều loại dữ liệu cần phân loại trong thực tế. Để mạnh mẽ hơn, Phần thực nghiệm của cách tiếp cận này sẽ triển khai thử nghiệm trên tập dữ liệu phức tạp hơn là Email Spam. Thí nghiệm này cũng sử dụng các ngưỡng tương tự chỉ khác Mushroom, dữ liệu Mailspam sẽ thay minconf = 0.9 thay vì 0.8 (minsup = 0.1, minconf = 0.9, minallconf = 0.5).

Bảng 4.2: Giá trị entropy của từng thuộc tính trên tập dữ liệu Mail Spam

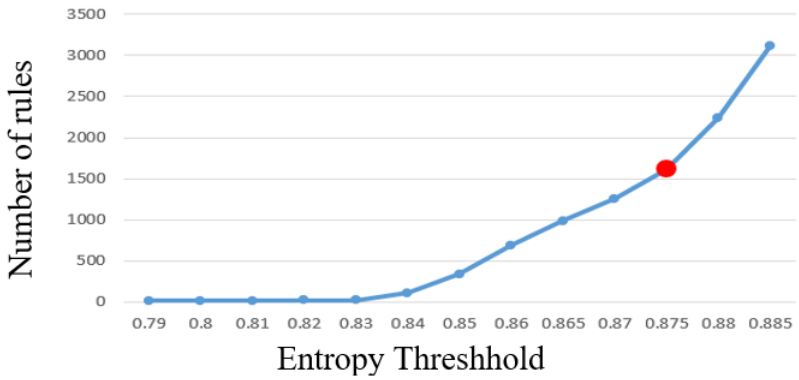
Entropy	Số thuộc tính	rules	Thời gian (ms)	Độ chính xác(%)
0.88	31	2239	3.5E+11	94.4

0.875	**29**	**1612**	**11E+9**	**94.9**
0.87	28	1254	261000000	94.06
0.865	27	989	51000000	93.04
0.86	26	692	10000000	92.4
0.85	24	338	2200000	90.99
0.84	22	109	35170	88.93
0.83	17	24	343	87.84
0.81	14	17	151	81.11
0.79	13	13	145	80.61

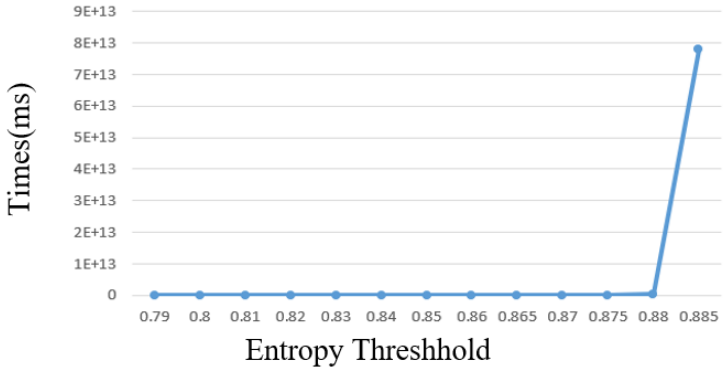
Bảng 4.2 mô tả thuật toán FSAC được triển khai 12 lần từ giá trị entropy ban đầu là 0,79 tăng dần và kết quả thực nghiệm trên Hình 4.4 minh họa độ chính xác phân loại ở entropy 0,875 có độ chính xác là 94,9%, số lượng thuộc tính là 29 so với số ban đầu là 57. Độ chính xác giảm nếu giá trị entropy tăng. Hình 4.5 minh họa số lượng quy tắc ở mức entropy tối ưu 0,875 là 1612, khi dữ liệu phức tạp (Mail Spam), số lượng quy tắc tăng đáng kể so với dữ liệu Mushroom. Hình 4.6 mô tả thời gian xử lý mà nó tiêu tốn đáng kể ở số lượng thuộc tính từ 29 đến 31.



Hình 4.4: Biểu đồ mô tả độ chính xác của thuật toán ACAC + FS theo từng ngưỡng entropy trên tập dữ liệu MailSpam.



Hình 4.5: Biểu đồ mô tả số lượng luật của thuật toán ACAC + FS theo từng ngưỡng entropy trên tập dữ liệu MailSpam.



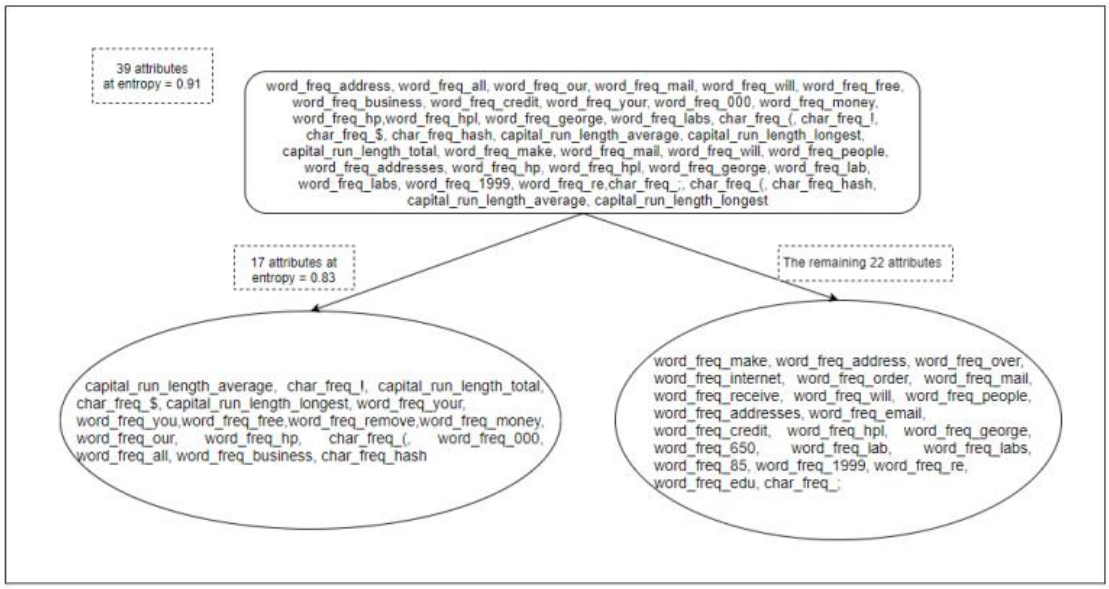
Hình 4.6: Biểu đồ mô tả tốc độ xử lý của thuật toán ACAC + FS theo từng ngưỡng entropy trên tập dữ liệu MailSpam.

Tóm lại, các tiếp cận cải thiện tập luật kết hợp qua thuật toán FSAC cải thiện đáng kể thuật toán ACAC dựa trên ý tưởng loại bỏ thuộc tính dư thừa trong tập dữ liệu huấn luyện. Để xác định xem thuộc tính có vai trò trong việc phân loại hay không, Bài báo đã sử dụng thước đo entropy giữa thuộc tính dữ liệu và thuộc tính đích. Thông qua quá trình thử nghiệm về tập dữ liệu Mushroom và MailSpam, nó đã được chỉ ra rằng thuộc tính với entropy cao được ưu tiên loại bỏ, thì độ chính xác phân loại sẽ tăng lên, tập quy tắc phân loại được giảm bớt và quá trình xử lý thời gian cũng giảm đi. Đối với dữ liệu lớn như Thư rác, việc xử lý thời gian vẫn còn khá lớn. Theo hướng tiếp theo của cải thiện, chúng ta sẽ tìm thấy mối quan hệ giữa bộ phân lớp luật kết hợp ban đầu và bộ phân lớp ấy nhỏ hơn sau loại bỏ một thuộc tính. Từ đó, có thể tìm một phương pháp để giảm chi phí tính toán lại bộ luật kết hợp nhỏ hơn.

4.3 Thực nghiệm cách tiếp cận kết hợp thuật toán leo đồi vào thuật toán phân lớp ACAC.

Thí nghiệm chỉ được so sánh trên tập dữ liệu MailSpam. Lý do cho việc không sử dụng tập dữ liệu Mushroom trước đó vào tập dữ liệu này là vì sau khi dùng kỹ thuật rút trích đặt trưng được mô tả trước đó. Thuật toán ACAC có độ chính xác tiệm cận tuyệt đối.

Để loại bỏ những thuộc tính không có vai trò phân lớp, thực nghiệm loại bỏ các thuộc tính có Entropy > 0.9 bằng cách tiếp cận chọn lọc đặc trưng trước đó. Qua đó số thuộc tính giảm từ 57 xuống còn 39 thuộc tính, thời gian xử lý và cải thiện các chỉ số đánh giá. Mặc dù đã loại bỏ 18 thuộc tính nhưng số thuộc tính vẫn là quá lớn cho việc tìm ra tập luật kết hợp phân lớp. Trong giai đoạn 1, thực nghiệm tìm ra tập thuộc tính ban đầu sao cho thời gian xử lý vẫn chấp nhận được và độ chính xác hợp lý. Và tập thuộc tính ban đầu tìm được từ thuật toán Phase 1 trong thực nghiệm này là 22 thuộc tính có độ Entropy ≤ 0.84, tập 17 thuộc tính còn lại được dùng trong phép thử tối ưu trong giai đoạn 2.

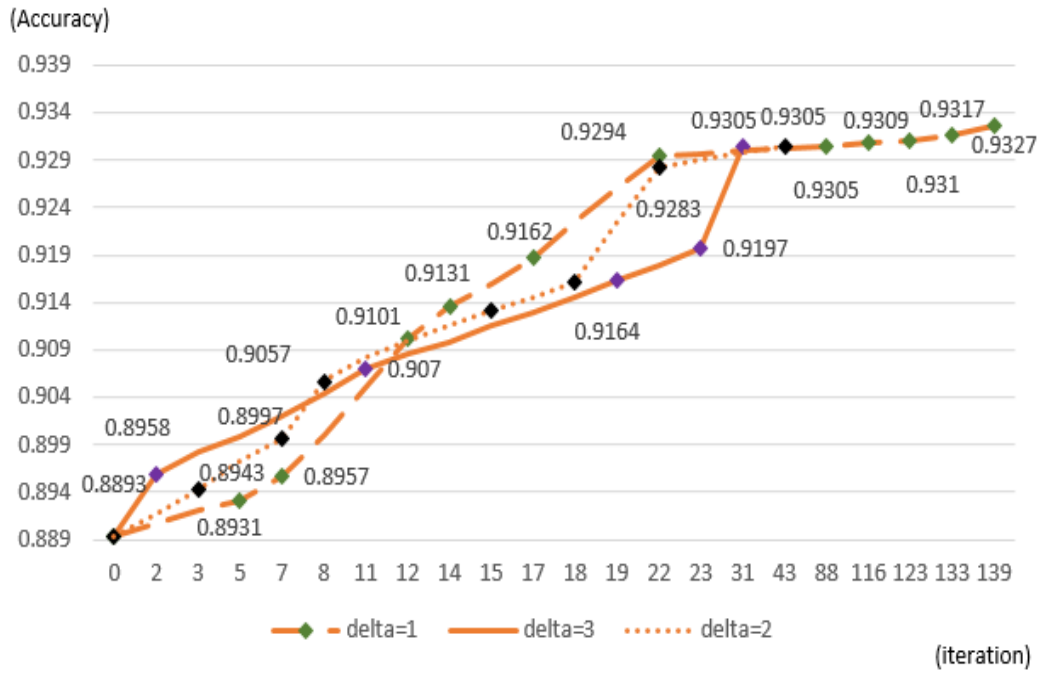


Hình 4.7: Hình minh họa việc chia dữ liệu với mốc Entropy = 0.84 trên tập MailSpam

Bước 1 dừng tại giá trị Entropy = 0.84 và đạt 22 thuộc tính vì cân bằng giữa hiệu suất và thời gian xử lý, khi giá trị Entropy = 0.85 lúc này số thuộc tính là 24 (xem bảng 2.2, cột 2) thì chi phí về thời gian thực hiện tăng lên đáng kể mặt dù độ

chính xác vẫn còn tăng. Hình 4.7 mô tả kết quả sau khi thực hiện xong thuật toán giai đoạn 1 là chia tập dữ liệu ban đầu thành 2 tập con. Tập best_Attributes (22 thuộc tính thu được từ thuật toán Phase 1) được cho là tập thuộc tính tương đối tốt nhưng chưa tối ưu và tập sub_Attributes (bao gồm các thuộc tính còn lại).

Sau khi chọn được tập thuộc tính ban đầu (best_Attributes) là tập 22 thuộc tính tại ngưỡng entropy = 0.84. bước tiếp theo đó chính là tìm ra tập 22 thuộc tính tối ưu hơn 22 thuộc tính ban đầu. Để thực hiện được điều đó, phần thực nghiệm sẽ được tiến hành chạy thuật toán Phase 2.1 đã nêu trên. Phần thực nghiệm này chọn giá trị delta lần lượt là 1, 2 và 3 với mỗi giá trị delta khác nhau sẽ cho những kết quả khác nhau như trong hình 4.8. Theo phương pháp leo đồi vẫn còn nhiều hạn chế, chưa thể kết luận giá trị delta ảnh hưởng như thế nào đến việc tối ưu, nhưng trong thực nghiệm này giá trị delta = 1 mang lại kết quả tốt nhất với tập thuộc tính tối ưu có độ chính xác là 0.9327 so với các giá trị còn lại. Giá trị delta = 2 trong thực nghiệm này có kết quả kém nhất (tốn 88 lần lặp so với 31 lần lặp khi delta = 3) về mặt thời gian cũng như độ chính xác.



Hình 4.8: Kết quả sau khi thực hiển thuật toán leo đồi trên tập MailSpam với Entropy = 0.84

Qua giai đoạn 2, Ta đã chọn ra được tập luật gồm có 22 thuộc tính có độ chính xác = 0.9305 (tương đối tốt vì thời gian thực hiện của delta = 3 chỉ là 31 lần

lặp so với thời gian delta = 1 mất 139 lần lặp) và 17 thuộc tính còn lại. Ở giai đoạn 3, thuật toán Phase 3 tìm ra tập 23 thuộc tính có độ chính xác cao hơn 0.9305. Để cụ thể hóa mục tiêu trên, phần thực nghiệm này sẽ kết hợp tập 22 thuộc tính và từng thuộc tính trong 17 thuộc tính còn lại. mỗi tập 23 thuộc tính sẽ cho những kết quả có thể khác nhau như trong bảng 2.2. Khi thực nghiệm Phase 3, ta có được 2 thuộc tính "word_freq_mail" và thuộc tính "char_freq_;" giúp cho tập thuộc tính tối ưu có kết quả tốt hơn.

Bảng 4.3: Kết quả sự kết hợp của tập dữ liệu thu được sau khi áp dụng thuật toán leo đồi với từng thuộc tính trong tập còn lại trên tập MailSpam.

Num	entropy	Tên thuộc tính	Thời gian (ms)	Số luật	Độ chính xác(%)
1	0.30840	capital_run_length_average	3195068	640	0.9305
2	0.65802	capital_run_length_longest	2991905	640	0.9305
3	0.78659	word_freq_hp	26753570	604	0.9305
4	0.78937	char_freq_(	3680278	605	0.9305
5	0.83027	word_freq_george	31908135	604	0.9305
6	**0.83569**	**word_freq_mail**	**54212858**	**957**	**0.9349**
7	0.83673	word_freq_hpl	59496226	604	0.9305
8	0.83897	word_freq_will	23804363	917	0.9305
9	0.83950	char_freq_hash	14989621	861	0.9305
10	0.87196	word_freq_re	17217569	703	0.9305
11	0.87707	word_freq_people	25790388	700	0.9305
12	0.87805	word_freq_1999	20146323	604	0.9305
13	0.88005	word_freq_make	71690336	645	0.9305
14	**0.88494**	**char_freq_;**	**19102413**	**1039**	**0.9392**
15	0.88756	word_freq_addresses	22861710	725	0.9305
16	0.89806	word_freq_labs	32724505	604	0.9305
17	0.90796	word_freq_lab	28166730	604	0.9305

Có một điều thú vị là sau khi kết thúc giai đoạn 1, thuật toán Phase 1 dừng tại giá trị Entropy = 0.84. Trong bảng 2.2, có 9 thuộc tính đầu tiên có giá trị Entropy < 0.84 và 8 thuộc tính cuối cùng > 0.84. Trong 9 thuộc tính đầu tiên có thuộc tính "word_freq_mail" có giá trị Entropy = 0.83569 giúp cho tập thuộc tính tối ưu từ Phase2.1 tăng độ chính xác phân lớp đến 0.9349. Trong 8 thuộc tính cuối cùng, có thuộc tính "char_freq_;" có giá trị Entropy = 0.88494 giúp cho tập thuộc tính tối ưu từ Phase2.1 tăng độ chính xác phân lớp đến 0.9392. Đây cũng là điều hạn chế của phương pháp lọc đặc trưng (Filter Method) mà bài báo đã trình bày ở trên.

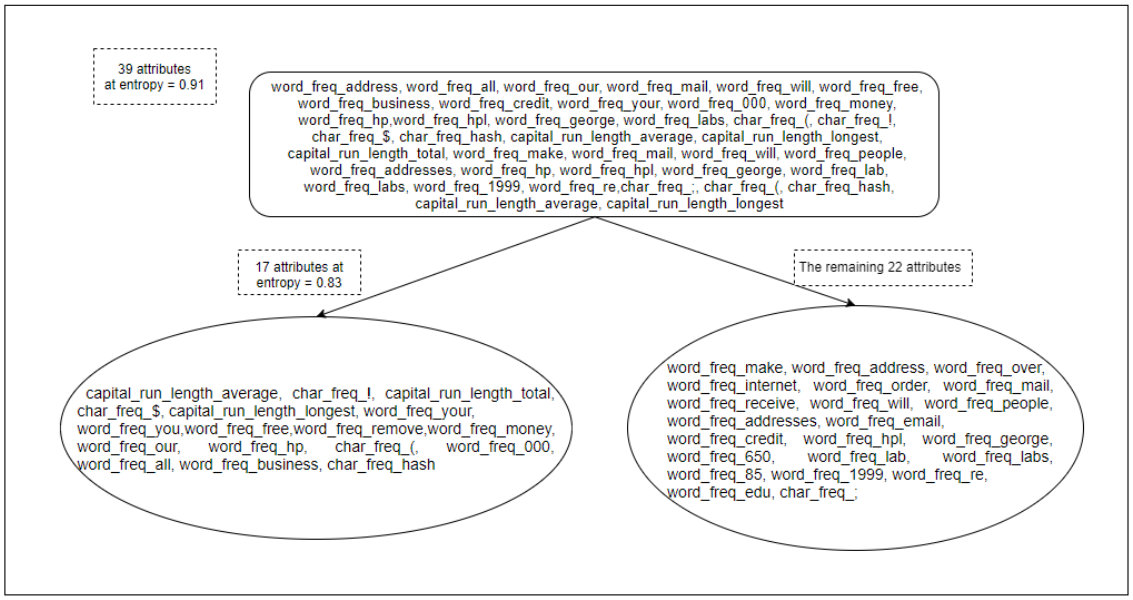
Cách tiếp cận sử dụng thuật toán leo đồi vào ACAC như đã đề cập là phương pháp lựa chọn đặc trưng theo hướng học có giám sát. Giai đoạn một thực hiệu việc lọc đặc trưng theo độ đo Entropy, giai đoạn hai thực hiện việc lựa chọn đặc trưng tối ưu theo nhóm nhỏ mô phỏng hình thức Leo đồi, Giai đoạn ba mở rộng tập tối ưu nhằm tìm bộ phân lớp có kết quả chính xác hơn. Cả ba giai đoạn đều có điểm mạnh và điểm hạn chế riêng. Tuy nhiên, theo phương pháp thực hiện linh hoạt như mô tả rõ trong thực nghiệm. Kết quả cuối cùng vẫn tìm ra tập thuộc tính tối ưu về mặt phân lớp cũng như thời gian xử lý chấp nhận được.

Cách tiếp cận này có thể mở rộng để đạt kết quả cao hơn khi vượt qua hạn chế của Leo đồi bằng các phương pháp khác như Luyện thép, Di truyền, Bầy đàn. Bên cạnh đó ta cũng có thể tiến hành trên các tập dữ liệu phức tạp hơn nhằm đánh giá độ ổn định cũng như tính chắc chắn của phương pháp.

4.4 Thực nghiệm cách tiếp cận kết hợp thuật toán Luyện thép vào thuật toán phân lớp ACAC.

Như đã nói thuật toán luyện thép là 1 dạng cải tiến của thuật toán leo đồi, nên phần thực nghiệm này vẫn sẽ chỉ tập trung so sánh trên tập dữ liệu MailSpam. Ngoài ra, phần thực nghiệm này cũng sẽ so sánh giữa cách tiếp cận này và cách tiếp cận bằng thuật toán leo đồi để nêu lên sự hiệu quả của thuật toán luyện thép trong việc cải tiến thuật toán phân lớp kết hợp ACAC.

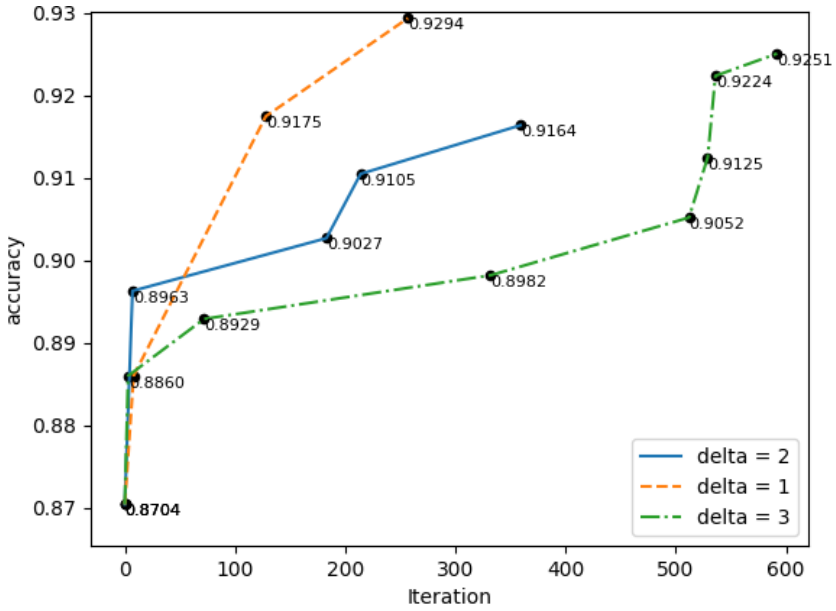
Cũng giống như cách tiếp cận bằng thuật toán leo đồi, nhưng thay vì chọn tập ban đầu là tập thuộc tính có ngưỡng Entropy nhỏ hơn hoặc bằng 0.84 thì ở các tiếp cận bằng thuật toán luyện thép sẽ chọn Ngưỡng Entropy = 0.83 làm mốc chia tập 37 thuộc tính tại ngưỡng entropy = 0.91 thành 2 tập dữ liệu.



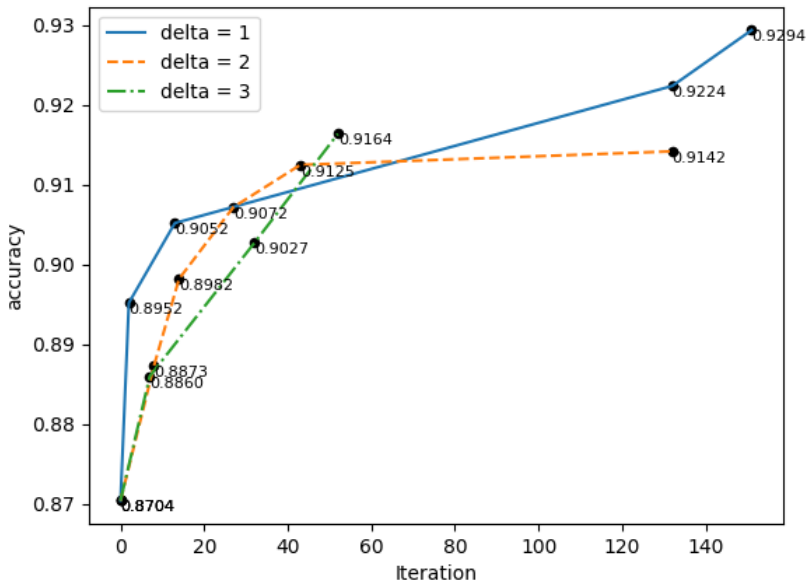
Hình 4.9: Hình minh họa việc chia dữ liệu với mốc Entropy = 0.83 trên tập MailSpam

Sau khi chọn tập thuộc tính ban đầu, có 17 thuộc tính ở ngưỡng entropy = 0,83. Bước tiếp theo là tìm một tập hợp gồm 17 thuộc tính tối ưu hơn hơn 17 thuộc tính ban đầu. Để làm được điều đó, thí nghiệm phần chọn giá trị delta lần lượt là 1, 2 và 3, mỗi phần giá trị delta khác nhau sẽ cho kết quả khác nhau. Trên này thử nghiệm, chọn thuật toán leo đồi và luyện thép để triển khai trong tìm kiếm siêu dữ liệu. Trong Hình 4.10, nó cho thấy những thay đổi cao nhất (độ chính xác) của tập dữ liệu qua các lần lặp của chương trình khi kết hợp thuật toán luyện thép. Tại delta = 1, số lượng thay đổi cực đại khi tìm một tập hợp có giá trị cao hơn độ chính xác so với bộ hiện tại là 7 lần, con số này ở mức delta = 3 là 3 lần và delta = 2 là 4 lần. Sau đây bộ dữ liệu có xu hướng có độ chính xác cao hơn bản gốc (0,8704) với mỗi đỉnh mới được tìm thấy và giá trị cao nhất mức cao nhất có thể đạt được ở delta=3 qua các lần lặp là 0,9294. Vì delta = 3, tìm đỉnh có độ chính xác = 0,9294 mất ít thời gian hơn và cũng có độ chính xác cao hơn delta = 2 hoặc đồng bằng = 3. Kết quả của thuật toán luyện thép (hình 4.10) trên ACAC cho thấy một cách tiếp cận cải tiến để cải thiện nhược điểm của thuật toán leo đồi chúng ta có thể tham khảo biểu đồ so sánh các chỉ số của từng phương pháp trên cùng một tập dữ liệu (17 thuộc tính) ở các giá trị delta khác nhau.

Hình 4.10: Kết quả sau khi thực hiển thuật toán luyện thép trên tập MailSpam với Entropy = 0.83



Hình 4.10: Kết quả sau khi thực hiển thuật toán luyện thép trên tập MailSpam với Entropy = 0.83



Hình 4.11: Kết quả sau khi thực hiển thuật toán leo đồi với Entropy =0.83

Từ giai đoạn 2, chúng tôi đã chọn một bộ quy tắc bao gồm 17 thuộc tính có độ chính xác = 0,9294 (đỉnh cao nhất) và 22 thuộc tính còn lại. Trong giai đoạn 3, chúng ta sẽ tìm thấy bộ 18 thuộc tính có độ chính xác cao hơn 0,9294. Để cụ thể hóa mục tiêu trên, phần thực nghiệm này sẽ kết hợp tập hợp 17 thuộc tính và mỗi thuộc tính còn lại 22 thuộc tính. Mỗi bộ 18 thuộc tính sẽ cho ra những giá trị khác

nhau kết quả có thể xảy ra. Trong trường hợp này chọn ‘word_freq_addresses’ vì hiệu suất của nó Sau khi trải qua 3 giai đoạn, Chúng ta đã thiết lập các thuộc tính tốt nhất để có thể sử dụng thuật toán phân loại luật kết hợp để xây dựng mô hình dự đoán.

Bảng 4.4: Kết quả sự kết hợp của tập dữ liệu thu được sau khi áp dụng thuật toán luyện thép với từng thuộc tính trong tập còn lại Trên tập Mail.

Entropy	Tên thuộc tính	Thời gian (ms)	Số luật	Độ chính xác(%)
0.308	capital_run_length_a	7862	170	9294
0.589	capital_run_length_t	7165	170	0.9294
0.658	capital_run_length_l	6891	170	0.9294
0.786	word_freq_hp	18390	170	0.9294
0.789	char_freq_(	7029	169	0.9294
0.815	word_freq_all	17124	219	0.9294
0.830	word_freq_george	19007	170	0.9294
0.835	word_freq_mail	19316	332	0.9305
0.836	word_freq_address	18231	225	0.9294
0.836	word_freq_hpl	18023	170	0.9294
0.838	word_freq_will	12262	267	0.9294
0.846	word_freq_email	13710	244	0.9294
0.866	**word_freq_credit**	**13292**	**278**	**0.9305**
0.871	word_freq_re	12716	194	0.9294
0.877	word_freq_people	17010	185	0.9294
0.878	word_freq_1999	18207	170	0.9294
0.884	char_freq_;	17159	327	0.9305
0.887	word_freq_addresses	17589	204	0.9305
0.892	word_freq_edu	18668	211	0.9294
0.898	word_freq_labs	17062	170	0.9294
0.898	word_freq_85	17108	170	0.9294
0.901	word_freq_650	17400	171	0.9294

Cách tiếp cận bằng thuật toán luyện thép hay leo đồi vẫn duy trì hầu hết các độ chính xác phân loại như nhau, nhưng thời gian chạy và quy tắc việc cắt tỉa hiệu quả hơn khi sử dụng thuật toán luyện thép do ý tưởng thuật toán đã cãi thiện những điểm hạn chế của thuật toán leo đồi. So sánh bảng 4.2 và 4.3 ta thấy, trên cùng một phân loại độ chính xác (0.9305 so với 0.9304), thời gian xử lý (13292

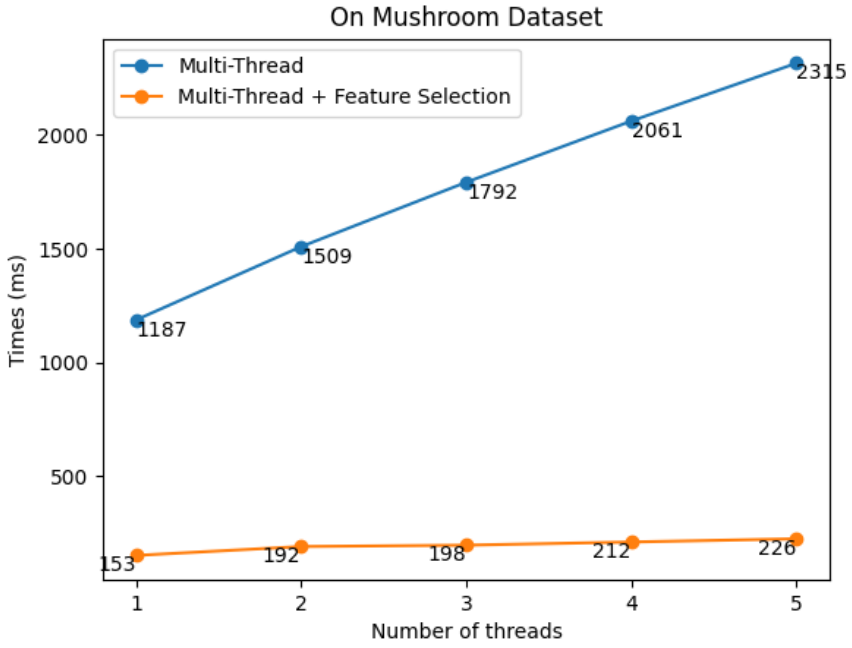
so với 51000000) nhanh hơn 3750 lần, số lượng quy tắc nhiều giảm (278 so với 989). Thời gian chạy quan trọng hơn số lượng quy tắc, đó là lý do tại sao chúng tôi chọn 278 quy tắc thay vì quy tắc 204.

4.5 Thực nghiệm cách tiếp cận đa luồng xử lý vào thuật toán ACAC.

Phần thực nghiệm cuối, công trình nghiên cứu này sẽ tiến hành và đánh giá cách tiếp cận đa luồng xử lý vào thuật toán ACAC trên cả 4 tập dữ liệu Mushroom, MailSpam, Default of credit card clients và Smoking and Drinking Dataset with body signal, các tập dữ liệu này đã được miêu tả chi tiết trước đó.

4.5.1 Đa luồng xử lý trên tập dữ liệu Mushroom.

Hình 4.12 mô tả kết quả của bảng 4.4, đây là kết quả thực nghiệm của tập dữ liệu Tập dữ liệu Mushroom. Kết quả tốt nhất vẫn là một luồng, bởi vì tập dữ liệu này chỉ có 8125 dòng dữ liệu. Quá ít trường hợp để tận dụng lợi thế của đa chủ đề. Dung lượng bộ nhớ tăng theo số lượng luồng.



Hình 4.12: Biểu đồ mô tả tốc độ xử lý trên từng số lượng luồng xử lý khác nhau trên tập Mushroom

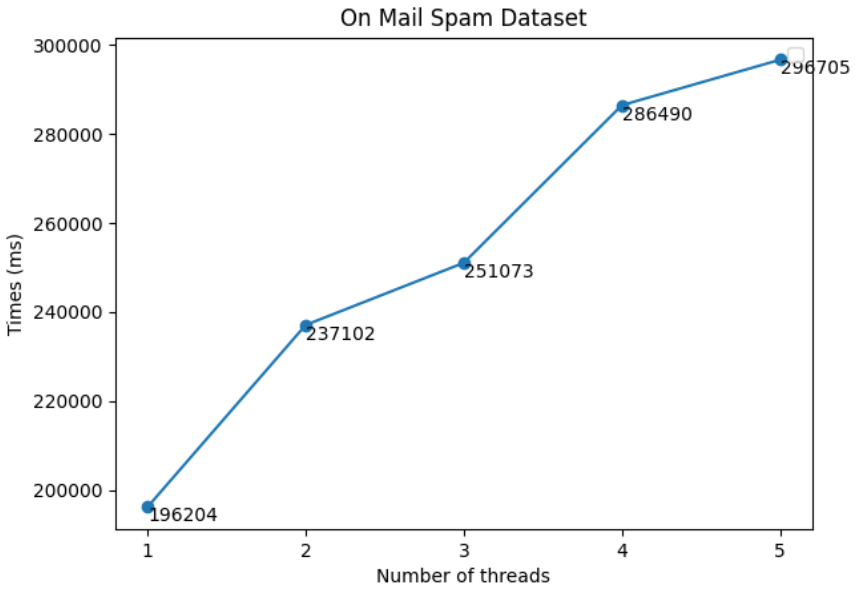
Bảng 4.5: Kết quả khi sử dụng Đa luồng xử lý vào ACAC trên tập Mushroom

Số lượng luồng	Đa luồng (ms)	Đa luồng + FS (ms)	Bộ nhớ(MB)

1	1187	153	1839.2
2	1509	192	1873.1
3	1792	198	1920.7
4	2061	212	2013.5
5	2315	223	2022.0

4.5.2 Sử dụng đa luồng xử lý vào thuật toán ACAC trên tập dữ liệu MailSpam.

Hình 5.13 mô tả kết quả của bảng 4.5, đây là kết quả thử nghiệm của tập dữ liệu Spammail Dataset. Kết quả tốt nhất vẫn là một luồng, vì dữ liệu này chỉ có 6025 dòng dữ liệu. Quá ít trường hợp để tận dụng tính năng đa luồng. Dung lượng bộ nhớ tăng theo số lượng luồng.



Hình 4.13: mô tả tốc độ xử lý trên từng số lượng luồng xử lý khác nhau trên tập MailSpam

Bảng 4.6: Kết quả khi sử dụng Đa luồng xử lý vào ACAC trên tập MailSpam

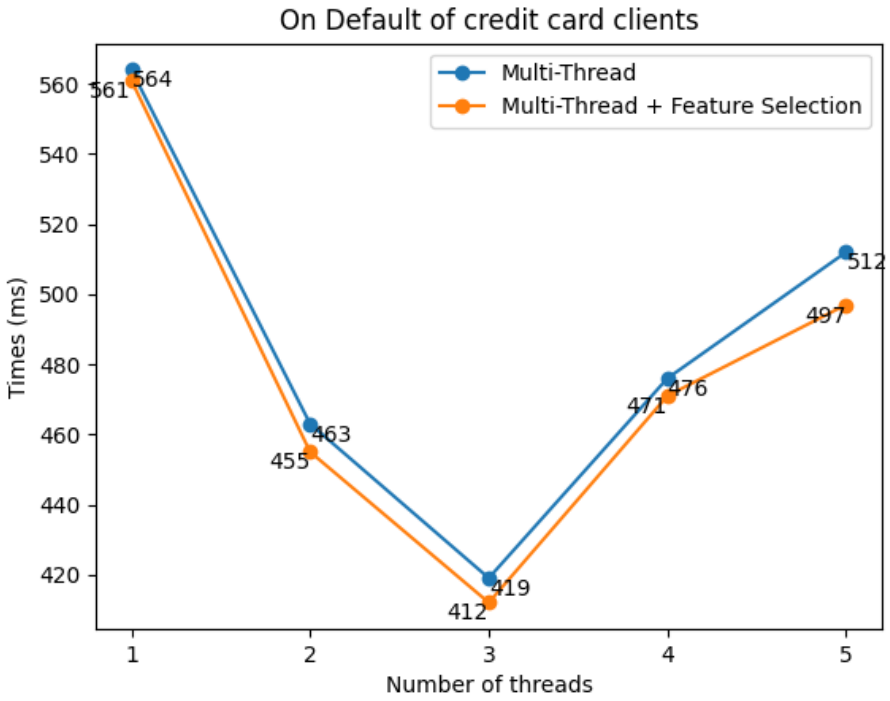
Số lượng luồng	Đa luồng + FS (ms)	Bộ nhớ(MB)
1	196204	2204.7
2	237102	2240.5

3	251073	2251.2
4	286490	2283.4
5	296705	2297.4

4.5.3 Sử dụng đa luồng xử lý vào thuật toán ACAC trên tập dữ liệu Default of credit card clients.

Hình 5.14 mô tả kết quả của bảng 4.6, đây là kết quả thực nghiệm của tập dữ liệu "Mặc định của khách hàng thẻ tín dụng", kết quả tốt nhất là khi số luồng là 3. Dung lượng bộ nhớ tăng theo số lượng luồng. Trong này thử nghiệm, chúng tôi thấy mặc dù kết quả đã cho thấy đa luồng xử lý đã cho kết quả tốt hơn so với việc sử dụng 1 luồng. Nhưng kết quả vẫn chưa thể hiện rõ rệt về kết quả của đa luồng xử lý do 30000 dòng dữ liệu tuy nhiều gấp 4-5 lần 2 tập dữ liệu đã sử dụng xuyên suốt các cách tiếp cận trên. Nhưng số lượng dòng dữ liệu của tập dữ liệu Default of credit card clients này vẫn chưa thật sự lý tưởng cho việc sự dụng đa luồng xử lý.

Hình 4.14: mô tả tốc độ xử lý trên từng số lượng luồng xử lý khác nhau trên tập Default of credit card clients

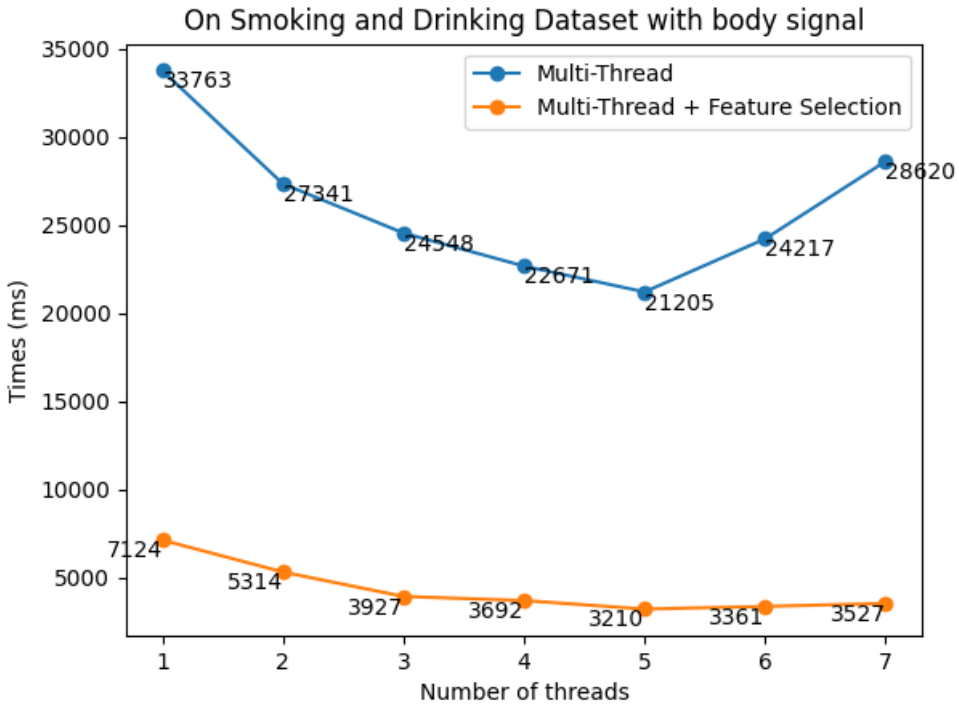


Bảng 4.7: Kết quả khi sử dụng Đa luồng xử lý vào ACAC trên tập Default of credit card clients

Số lượng luồng	Đa luồng (ms)	Đa luồng + FS (ms)	Bộ nhớ(MB)
1	564	561	1839.2
2	463	455	1873.1
3	419	412	1920.7
4	476	471	2013.5
5	512	497	2022.0

4.5.4 Sử dụng đa luồng xử lý vào thuật toán ACAC trên tập dữ liệu Smoking and Drinking Dataset with body signal.

Với số lượng gần 1 triệu dòng dữ liệu kết quả của cách tiếp cận đa luồng xử lý sẽ cho thấy được kết quả mong đợi khi kết hợp đa luồng xử lý vào thuật toán ACAC được thể hiện ở bảng 4.7 và trực quan hóa ở hình 4.15.



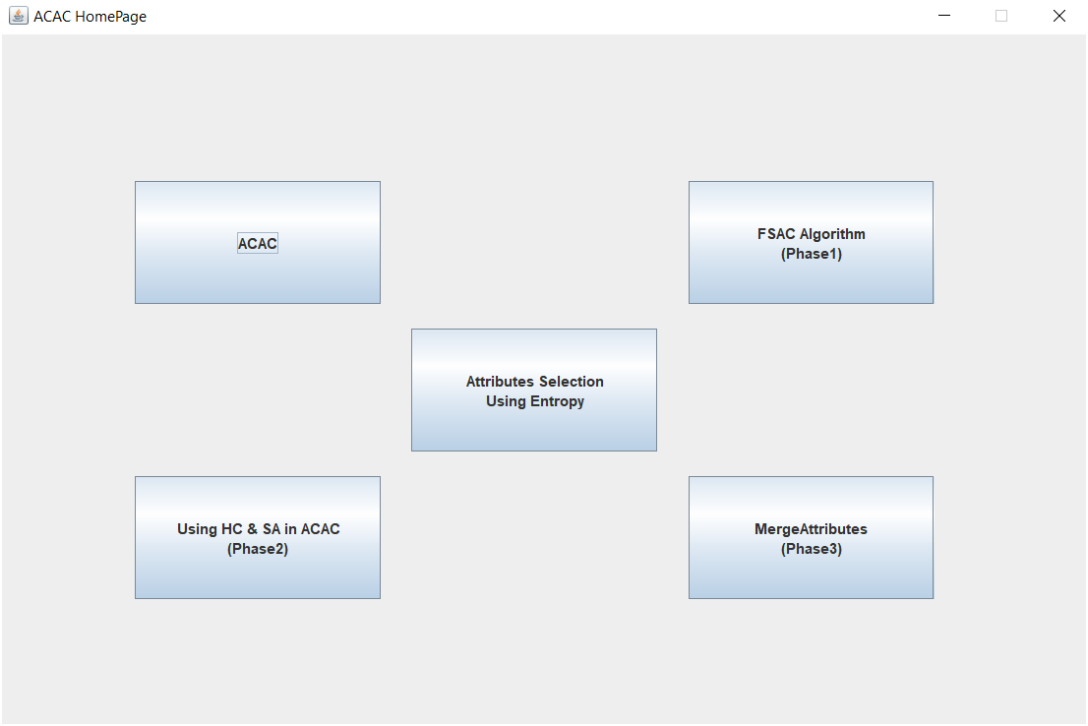
Hình 4.15: mô tả tốc độ xử lý trên từng số lượng luồng xử lý khác nhau tập Smoking and Drinking Dataset with body signal

Bảng 4.8: Kết quả khi sử dụng Đa luồng xử lý vào ACAC trên tập Smoking and Drinking Dataset with body signal

Số lượng luồng	Đa luồng (ms)	Đa luồng + FS (ms)	Bộ nhớ(MB)
1	33763	7124	3251.6
2	27341	5314	3432.0
3	24548	3926	3579.4
4	22671	3692	3664.7
5	21205	3210	3781.3
6	24217	3361	3858.9
7	28620	3527	3971.2

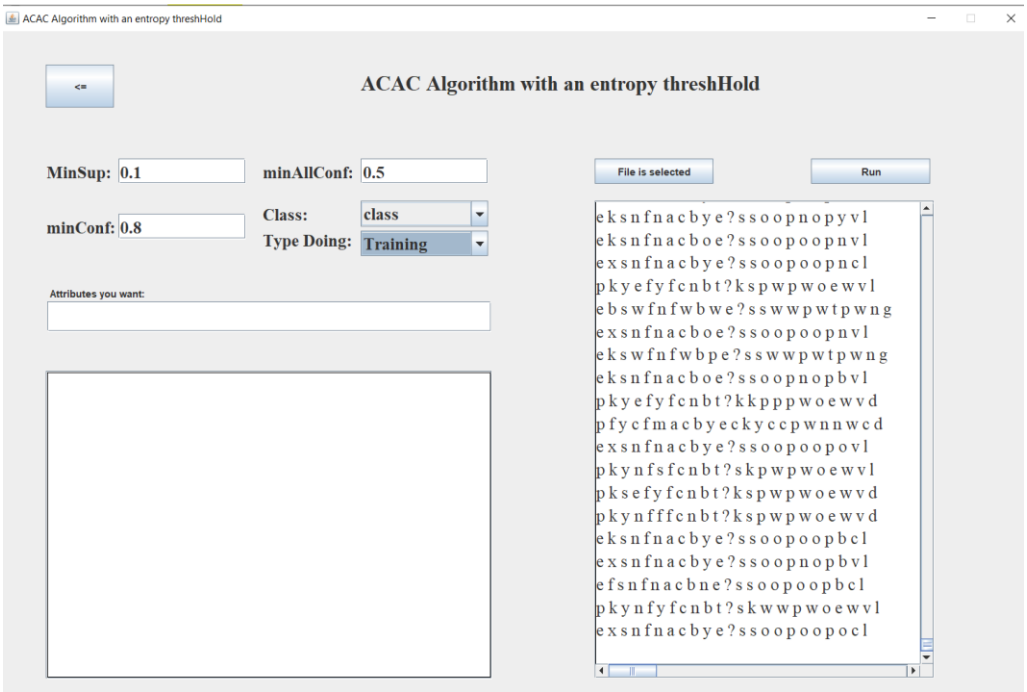
4.6 Cài đặt chương trình.

Qua các hướng tiếp cận nhằm mục đích cải tiến thuật toán phân lớp kết hợp, công trình nghiên cứu này đã đạt được những kết quả vô cùng tốt, nằm trong mong đợi kể từ thời điểm tiến hành xây dựng công trình này. Trong qua trình gần một năm nghiên cứu, qua các giai đoạn, qua các bước tiếp cận. Cứ mỗi khi có kết quả khả quan trong các tiếp cận đó. Công trình nghĩa cứu sẽ thực nghiệm trên SPMF trên nền tảng ngôn ngữ lập trình JAVA như đã nói, những tính tái sử dụng cũng như trải nghiệm của người dùng. Chính vì lý do đó công trình nghiên cứu này sẽ có thêm một phần mềm để thực nghiệm các cách tiếp cận trên. Qua đó vừa có thể chứng minh độ minh bạch của công trình nghiên cứu này vừa có thể cho thế hệ sau này kế thừa và học tập. Mã nguồn sẽ được lưu trữ trong dư án ScienceResearchSGU20_24[14].



Hình 4.16: mô tả tổng quát các chức năng trong ứng dụng

Trong ứng dụng này có những chức năng như thực hiện thuật toán ACAC nếu người dùng chon "ACAC".



Hình 4.17: Chức năng ACAC trên ứng dụng

Tại chức năng này người dùng có thể chọn file dữ liệu đầu vào, các thuộc tính cần dữ lại, nhãn dự đoán và các ngưỡng giá trị như trong mô tả thuật toán

ACAC. Và “Type Doing” cho biết mục đích của người dùng là huấn luyện hay dư đoán. Các chức năng còn lại giao diện cũng như các option chọn gần như là như nhau.Sử dụng ngưỡng Entropy cho lựa chọn đặt trưng nếu người dùng chọn “FSAC Algorithm”. Sử dụng thuật toán leo đồi nếu chọn “Using HC & SA in ACAC”. Sau khi thực nghiệm trên thuất toán leo đồi hoặc luyện thép người dùng có thể kết hợp các thuộc tính tốt nhất đó vời 1 trong các thuộc tính còn lại bằng chạy chọn chức năng MergeAttributes. Hoặc nếu chỉ muốn biết các ngưỡng giá trị entropy của từng thuộc tính người dùng có thể chọn “Attributes Selection Using Entropy”.

KẾT LUẬN VÀ KHUYẾN NGHỊ

Dữ liệu giao tác hay còn được biết đến là dữ liệu của các giao dịch chiếm tỷ trọng lớn trong cuộc sống hiện nay dẫn đến các mô hình sử dụng luật kết hợp đóng góp rất lớn cho việc xây dựng các mô hình dự đoán, phân loại. Các cách tiếp cận được đề cập trong công trình nghiên cứu này ít nhiều đã đóng góp lớn trong việc tối ưu tập luật kết hợp.

Ở các cách tiếp cận trên tuy đã tối ưu được luật kết hợp trong một số mô hình dư đoán trên các tập dữ liệu khác nhau. Nhưng chung quy lại, các tập dữ liệu ấy vẫn chỉ là những tập dữ liệu cục bộ không thật sự phù hợp với xu thế hiện nay khi dữ liệu của các doanh nghiệp hiện nay theo xu hướng là hệ cơ sở dữ liệu phân tán. Chính vì xu thế đó, dựa trên các kết quả ngoài mong đợi đạt được trong công trình này hướng nghiên cứu tiếp theo sau này sẽ tập trung tối ưu tập luật kết hợp của các mô hình trên các tập dữ liệu phân tán.

TÀI LIỆU THAM KHẢO

[1]. Foxiao Zhan, Xiaolan Zhu, Lei Zhang, Xuexi Wang, Lu Wang, Chaoyi Liu “Summary of Association Rules” IOP Conference Series: Earth and Environmental Science.

[2]. Li, D. Qin, and C. Yu, Associative classification based on closed frequent itemsets,2008, pp. 380-384.

[3]. G. Kundu, M. M. Islam, S. Munir, and M. F. Bari, An associative classifier with negative rules, in in 2008 11th IEEE International Conference on ComputationalScience and Engineering, July 2008, pp. 369-375.

[4]. B. Liu, W. Hsu, and Y. Ma, Integrating classification and association rule mining, Proc. 4th International Conference on Knowledge Discovery and Data Mining (KDD98),1998, pp. 80-86.

[5]. Z. Huang, Z. Zhou, T. He, and X. Wang, "ACAC: Associative classification based on all-confidence", 11 2011, pp. 289-293.

[6]. N. Q. Huy, T. A. Tuan, N. T. N. Thanh, “An efficient algorithm that optimizes the classification association rule set”, VNICT 26, pp. 13-19, 2023.

[7]. Leticia Hernando, Alexander Mendiburu, Jose A. Lozano, “Hill-Climbing Algorithm: Let's Go for a Walk Before Finding the Optimum”, IEEE 8-13 July 2018.

[8]. H.Q.Huy, T.A.Tuan, D.N.Tai, “A meta-heuristic approach for enhancing performance of associative classification”, ICTMAG 2024-01-22.

[9]. Darrall Henderson, Sheldon H Jacobson, Alan W. Johnson, “The Theory and Practice of Simulated Annealing” Handbook of Metaheuristics (pp.287-319) April 2006

[10]. Mushroom Classification Dataset, [Online].Available: https://www.kaggle.com/datasets/uciml/mushroom-classification

[11]. Spam Emails Dataset, [Online]. Available: https://www.kaggle.com/datasets/yasserh/spamemailsdataset

[12]. Default of credit card clients, [Online]. Available: https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients

[13]. Smoking and Drinking Dataset with body signal, [Online]. Available: https://www.kaggle.com/datasets/sooyoungher/smoking-drinking-dataset.

[14].ScienceResearchSGU20_24 Available: https://github.com/Azure2212/ScienceComputerSGU20_24.git.