# LỜI CAM ĐOAN

Tôi xin cam đoan rằng luận văn **“Khai thác luật phân lớp kết hợp trên dữ liệu mất cân bằng về lớp”** là công trình nghiên cứu của riêng tôi. Các số liệu, kết quả nêu trong Luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện Luận văn này đã được cảm ơn và các thông tin trích dẫn trong Luận văn đã được chỉ rõ nguồn gốc.

**Học viên thực hiện Luận văn**

*(Ký và ghi rõ họ tên)*

# LỜI CÁM ƠN

Lời đầu tiên, tôi xin chân thành cám ơn PGS.TS. Quản Thành Thơ đã tận tình truyền đạt và hướng dẫn tôi trong suốt thời gian thực hiện luận văn này. Thầy luôn tận tâm giúp đỡ, định hướng cho tôi trong suốt thời gian nghiên cứu khoa học. Thầy đã giúp tôi tiếp cận với khoa học và biết cách sáng tạo trong khoa học, với những điều mới trong xã hội và đạt được thành công trong bài nghiên cứu của mình.

Tiếp theo tôi xin bày tỏ lòng biết ơn đến PGS.TS. Võ Đình Bảy, Ban Giám hiệu, quý thầy cô trong Khoa Công nghệ thông tin, Viện đào tạo Sau Đại học trường Đại học Công nghệ TPHCM đã cung cấp những kiến thức quý báu cho tôi trong suốt quá trình học tập và nghiên cứu tại trường.

Cuối cùng, tôi xin chân thành gửi lời cám ơn đến gia đình, bạn bè và những người thân luôn quan tâm và giúp đỡ tôi trong suốt thời gian học tập và nghiên cứu hoàn thành luận văn này.

Vì thời gian có hạn và kiến thức còn hạn chế, nên luận văn khó tránh khỏi những thiếu sót, rất mong nhận được sự đóng góp ý kiến quý báu của quý thầy cô, anh chị và các bạn.

**Phan Tiến Dũng**

# TÓM TẮT

Kỹ thuật phân lớp đang đóng một vai trò quan trọng trong các hệ thống hỗ trợ ra quyết định. Trong thời gian gần đây có rất nhiều thuật toán tập trung vào vấn đề dự đoán lớp trên cơ sở luật kết hợp. Một trong các điểm yếu của phân lớp dựa vào luật phân lớp kết hợp là chọn ngưỡng độ hỗ trợ tối thiểu, đặc biệt là đối với các dữ liệu có sự mất cân bằng về lớp một cách rõ rệt: nếu chọn ngưỡng quá cao dẫn đến các lớp chứa ít mẫu sẽ không phổ biến, vì vậy sẽ không có luật nào chứa lớp này, nếu chọn ngưỡng thấp để khai thác được các luật chứa lớp thiểu số thì số lượng luật của lớp đa số vẫn áp đảo nên cũng ảnh hưởng đến giai đoạn dự đoán lớp. Để giải quyết vấn đề này, Nguyễn và đồng sự (năm 2016) đã đề xuất phương pháp phân cụm dữ liệu bằng thuật toán K-means để cân bằng số mẫu của mỗi lớp, sau đó sử dụng thuật toán CAR-Miner để khai thác tập luật phân lớp[17]. Xuất phát từ những hạn chế của phương pháp phân cụm K-means trong quá trình cân bằng số mẫu thuộc về các lớp của dữ liệu và nhược điểm của thuật toán CAR-Miner, luận văn đề xuất một phương pháp mới kết hợp K-means và HAC làm tăng độ tương tự của dữ liệu sau khi phân cụm nhằm mục đích tăng độ chính xác cho giai đoạn dự đoán lớp sau này, đồng thời áp dụng thuật toán CAR-Miner-Diff để giải quyết hạn chế của CAR-Miner trong nghiên cứu ở [17]. Các kết quả so sánh thuật toán trên CSDL chuẩn cũng sẽ được đưa ra trong luận văn để minh họa cho phương pháp đề xuất.

Bên cạnh đó, hiện nay nằm trong xu thế hiện đại hóa, việc khai thác cơ sở dữ liệu về an toàn giao thông, cụ thể là dữ liệu xử phạt vi phạm hành chính trên lĩnh vực trật tư an toàn giao thông (TTATGT) đường bộ mới chỉ dừng lại ở việc lưu trữ thông tin và thống kê báo cáo là chính, chưa khai thác hết các tác dụng mà cơ sở dữ liệu mang lại. Chúng ta chưa thấy được cơ sở dữ liệu có được đó đã phản ánh được những vấn đề gì về tình hình vi phạm TTATGT: những hành vi nào thường xuyên vi phạm, thời gian nào người điều khiển vi phạm nhiều nhất, địa bàn nào nổi cộm nhất, những hành vi nào gây tai nạn nhiều nhất, đặc biệt là khả năng dự báo gây tai nạn đối với những trường hợp tương tự… Để có thể khai thác kho dữ liệu khổng lồ của lực lượng cảnh sát giao thông về xử phạt vi phạm hành chính và xử lý tai nạn phục vụ cho công tác tham mưu, đề xuất các chính sách đảm bảo duy trì trật tự an toàn giao thông trên địa bàn các tỉnh, luận văn sẽ áp dụng phương pháp đề xuất xây dựng một chương trình ứng dụng vào việc dự đoán khả năng gây tai nạn dựa trên nhận diện về giá trị thuộc tính của các đối tượng khác nhau.

# ABSTRACT

Classification techniques are playing an important role in decision support systems. In recent times, many algorithms have focused on class prediction based on association rules. One of the weaknesses of the classification based on association rules is to choose the minimum support threshold, especially for data with markedly unbalanced class: if the threshold is too high, classes with few samples will not be frequent, so there will be no rules that contain this class, if users choose the low threshold to mine rules containing minority classes, the number of rules of the majority class will still be overwhelming, affecting the class prediction stage. To solve this problem, Nguyen et al. (2016) proposed the method of clustering original data by K-means algorithm to balance the number of samples of each class, then using the CAR-Miner algorithm to mine classification association rules [17]. Due to the limitations of K-means clustering method in balancing the number of samples belonging to classes and the disadvantages of CAR-Miner algorithm, the thesis proposes a new method which combines K- means and HAC to increase the similarity of data after clustering, thus increasing the accuracy for later class prediction. The thesis uses CAR-Miner-Diff algorithm to solve drawbacks of CAR-Miner in previous research. The algorithm comparison results on standard databases will also be presented in the thesis to illustrate the proposed method.

In addition, in the trend of modernization, the harnessing of the database on traffic safety, especially the data on sanctioning of administrative violations in the field of traffic safety just stops at storing information and reporting statistics, not fully exploiting the effects that the database brings. We have not yet seen the hidden information which is reflected through the database, such as which actions are frequent violations, the time when citizens violate the most, locations where citizens violate the most, which behaviors cause the most accidents, especially the ability to forecast accidents in similar cases... In order to mine the huge data warehouse of the traffic police force on sanctioning of administrative violations and handling accidents for advising and proposing policies to ensure maintenance of safety, the thesis will apply the proposed method of building an program for predicting the possibility of accidents based on the identification of attribute values of different objects.

# MỤC LỤC

[LỜI CAM ĐOAN i](#_Toc29891260)

[LỜI CÁM ƠN ii](#_Toc29891261)

[TÓM TẮT iii](#_Toc29891262)

[ABSTRACT v](#_Toc29891263)

[MỤC LỤC vii](#_Toc29891264)

[DANH MỤC CÁC TỪ VIẾT TẮT x](#_Toc29891265)

[DANH MỤC CÁC BẢNG xi](#_Toc29891266)

[DANH MỤC CÁC HÌNH xii](#_Toc29891267)

[MỞ ĐẦU 1](#_Toc29891268)

[1. Lý do chọn đề tài 1](#_Toc29891269)

[2. Mục tiêu nghiên cứu 2](#_Toc29891270)

[3. Đối tượng và phạm vi nghiên cứu 2](#_Toc29891271)

[4. Cấu trúc luận văn 2](#_Toc29891272)

[CHƯƠNG 1: TỔNG QUAN 4](#_Toc29891273)

[1.1. Giới thiệu về khai thác dữ liệu 4](#_Toc29891274)

[1.1.1. Khái niệm 4](#_Toc29891275)

[1.1.2. Các dạng dữ liệu có thể khai thác 4](#_Toc29891276)

[1.1.3. Quá trình khai thác dữ liệu 5](#_Toc29891277)

[1.1.4. Các hướng tiếp cận cơ bản 7](#_Toc29891278)

[1.1.5. Một số lĩnh vực ứng dụng 7](#_Toc29891279)

[1.2. Giới thiệu về khai thác luật kết hợp 8](#_Toc29891280)

[1.2.1. Tổng quan 8](#_Toc29891281)

[1.2.2. Một số khái niệm cơ bản 9](#_Toc29891282)

[1.2.3. Một số tính chất với tập mục phổ biến 10](#_Toc29891283)

[1.3. Giới thiệu về khai thác luật phân lớp kết hợp 11](#_Toc29891284)

[1.3.1. Một số định nghĩa và khái niệm 11](#_Toc29891285)

[1.3.2. So sánh giữa luật kết hợp và luật phân lớp kết hợp 12](#_Toc29891286)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 15](#_Toc29891287)

[2.1. Một số phương pháp khai thác luật kết hợp 15](#_Toc29891288)

[2.1.1. Giải thuật Apriori 15](#_Toc29891289)

[2.1.2. Giải thuật FP-Growth 18](#_Toc29891290)

[2.1.3. Phương pháp IT-tree 23](#_Toc29891291)

[2.2. Một số kỹ thuật phân lớp trong khai thác dữ liệu 25](#_Toc29891292)

[2.2.1. Giới thiệu về phân lớp 25](#_Toc29891293)

[2.2.2. Phân lớp dữ liệu với cây quyết định (Decision tree) 27](#_Toc29891294)

[2.2.3. Phân lớp dữ liệu bằng giải thuật học quy nạp ILA (Inductive Learning Algorithm) 28](#_Toc29891295)

[2.2.4. Phân lớp dữ liệu với mạng Naive Bayes 29](#_Toc29891296)

[2.2.5. Phân lớp dữ liệu với mạng neural 30](#_Toc29891297)

[2.3. Một số phương pháp khai thác luật phân lớp kết hợp 31](#_Toc29891298)

[2.3.1. Thuật toán CBA (Classification Based on Associations) 32](#_Toc29891299)

[2.3.2. Thuật toán CMAR (Classification Based on Multiple Rules) 35](#_Toc29891300)

[2.3.3. Thuật toán CAR-Miner 41](#_Toc29891301)

[2.3.4. Thuật toán CAR-Miner-Diff 45](#_Toc29891302)

[CHƯƠNG 3: PHƯƠNG PHÁP CẢI TIẾN ĐỀ XUẤT 50](#_Toc29891303)

[3.1. Các phương pháp phân cụm dữ liệu phổ biến 50](#_Toc29891304)

[3.1.1. Giới thiệu về kỹ thuật phân cụm 50](#_Toc29891305)

[3.1.2. Ứng dụng của phân cụm dữ liệu 51](#_Toc29891306)

[3.1.3. Kiểu dữ liệu dựa trên kích thước miền và độ đo tương tự 51](#_Toc29891307)

[3.1.4. Một số kỹ thuật phân cụm phổ biến 53](#_Toc29891308)

[3.2. Phương pháp cải tiến đề xuất 58](#_Toc29891309)

[CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ 63](#_Toc29891310)

[4.1. Môi trường và cơ sở dữ liệu thực nghiệm 63](#_Toc29891311)

[4.1.1. Môi trường thực nghiệm 63](#_Toc29891312)

[4.1.2. Giới thiệu về Pyqt5 63](#_Toc29891313)

[4.1.3. Giới thiệu về giao diện chương trình 65](#_Toc29891314)

[4.1.4. Cơ sở dữ liệu thực nghiệm 67](#_Toc29891315)

[4.2. Kết quả so sánh về độ chính xác 68](#_Toc29891316)

[4.3. Kết quả so sánh về thời gian thực hiện thuật toán 71](#_Toc29891317)

[4.4. Kết quả so sánh về độ tương tự sau khi phân cụm 73](#_Toc29891318)

[4.5. Khai thác dữ liệu xử phạt vi phạm hành chính giao thông 74](#_Toc29891319)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 76](#_Toc29891320)

[5.1. Kết luận 76](#_Toc29891321)

[5.2. Hướng phát triển 76](#_Toc29891322)

[DANH MỤC TÀI LIỆU THAM KHẢO 78](#_Toc29891323)

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| AC | Associative Classification |
| AGNES | Agglomerative Nesting |
| CAR-Miner | Class-Association Rule Miner |
| CAR-Miner-Diff | Class-Association Rule Miner using the Difference of Obidsets |
| CBA | Classification Based on Associations |
| CBA-RG | Classification Based on Associations Rule Generator |
| CMAR | Classification Based on Multiple Rules |
| CPAR | Classification Based on Predictive Association Rules |
| CSDL | Cơ sở dữ liệu |
| ECR-CARM | Equivalence Class-Rules tree based on Association Rule Mining |
| ECR-tree | Equivalence Class-Rules tree |
| FP-tree | Frequent Pattern tree |
| HAC | Hierarchical Agglomerative Clustering |
| ID3 | Iterative Dichotomiser 3 |
| ILA | Inductive Learning Algorithm |
| KDD | Knowledge Discovery in Database |
| MCAR | Multi-class Classification Based on Association Rules |
| MECR-tree | Modified Equivalence Class-Rules tree |
| MLP | Multilayer Perceptron |
| MMAC | Multi-class, Multi-label Associative Classification |
| OID | Object Identifier |
| TTATGT | Trật tự an toàn giao thông |

# DANH MỤC CÁC BẢNG

[Bảng 1.1: Một CSDL huấn luyện mẫu [10] 12](#_Toc29891354)

[Bảng 2.1: Minh họa kết quả bước 1 của thuật toán FP-Growth 19](#_Toc29891355)

[Bảng 2.2: Danh sách L các mục đã được sắp xếp theo độ hỗ trợ giảm dần 20](#_Toc29891356)

[Bảng 2.3: Cơ sở dữ liệu mẫu 24](#_Toc29891357)

[Bảng 2.4: Định dạng dữ liệu dọc của CSDL 24](#_Toc29891358)

[Bảng 2.5: Dữ liệu huấn luyện T[3] 36](#_Toc29891359)

[Bảng 2.6: Các luật được khai thác[3] 38](#_Toc29891360)

[Bảng 3.1: Ví dụ về dữ liệu mất cân bằng về lớp[17] 60](#_Toc29891361)

[Bảng 3.2: Bảng CSDL sau khi đã cân bằng lớp[17] 61](#_Toc29891362)

[Bảng 4.1: CSDL chuẩn thực nghiệm 68](#_Toc29891363)

[Bảng 4.2: Kết quả thực nghiệm về độ chính xác trên CSDL Breast cancer 68](#_Toc29891364)

[Bảng 4.3: Kết quả thực nghiệm về độ chính xác trên CSDL Chess 69](#_Toc29891365)

[Bảng 4.4: Kết quả thực nghiệm về độ chính xác trên CSDL Diabetes 69](#_Toc29891366)

[Bảng 4.5: Kết quả thực nghiệm về độ chính xác trên CSDL Tic-tac-toe 69](#_Toc29891367)

[Bảng 4.6: Kết quả thực nghiệm về thời gian thực thi trên CSDL Breast cancer 71](#_Toc29891368)

[Bảng 4.7: Kết quả thực nghiệm về thời gian thực thi trên CSDL Chess 71](#_Toc29891369)

[Bảng 4.8: Kết quả thực nghiệm về thời gian thực thi trên CSDL Diabetes 72](#_Toc29891370)

[Bảng 4.9: Kết quả thực nghiệm về thời gian thực thi trên CSDL Tic-tac-toe 72](#_Toc29891371)

[Bảng 4.10: Kết quả thực nghiệm về độ tương tự khi phân cụm 74](#_Toc29891372)

# DANH MỤC CÁC HÌNH

[Hình 1.1: Quá trình khai thác dữ liệu [21] 5](#_Toc29891325)

[Hình 2.1: Minh họa thuật toán Apriori [22] 17](#_Toc29891326)

[Hình 2.2: Quá trình xây dựng cây từ tác vụ 1, 2, 3 [23] 20](#_Toc29891327)

[Hình 2.3: Tiếp tục xây dựng cây từ tác vụ 4, 5, 6 [23] 20](#_Toc29891328)

[Hình 2.4: Các đầu mục liên kết và FP-tree với CSDL [23] 21](#_Toc29891329)

[Hình 2.5: Cây IT-tree được tạo ra từ CSDL mẫu[23] 25](#_Toc29891330)

[Hình 2.6: Một ví dụ về cây quyết định [22] 28](#_Toc29891331)

[Hình 2.7: Thuật toán CBA-RG [1] 34](#_Toc29891332)

[Hình 2.8: Cây FP-tree[3] 37](#_Toc29891333)

[Hình 2.9: Cấu trúc cây cho tập luật khai thác[3] 39](#_Toc29891334)

[Hình 2.10: Thuật toán CAR-Miner[9] 43](#_Toc29891335)

[Hình 2.11: Cây MECR-tree xây dựng từ bảng 1.1.[9] 44](#_Toc29891336)

[Hình 2.12: Thuật toán CAR-Miner-Diff[10] 47](#_Toc29891337)

[Hình 3.1: Hình dạng cụm dữ liệu được khai phá bời K-means[21] 55](#_Toc29891338)

[Hình 3.2: Cấu trúc cây cụm dữ liệu được khai phá bởi HAC[21] 58](#_Toc29891339)

[Hình 3.3: Các bước kết hợp thuật toán K-means với Car-Miner 59](#_Toc29891340)

[Hình 3.4: Các bước kết hợp thuậ toán K-means + HAC với Car-Miner-Diff 62](#_Toc29891341)

[Hình 4.1: Giao diện chính chương trình 65](#_Toc29891342)

[Hình 4.2: Giao diện chương trình Kết quả test 66](#_Toc29891343)

[Hình 4.3: Giao diện chương trình dự đoán ATGT 67](#_Toc29891344)

[Hình 4.4: So sánh về độ chính xác trên CSDL Breast cancer 69](#_Toc29891345)

[Hình 4.5: So sánh về độ chính xác trên CSDL Chess 70](#_Toc29891346)

[Hình 4.6: So sánh về độ chính xác trên CSDL Chess 70](#_Toc29891347)

[Hình 4.7: So sánh về độ chính xác trên CSDL Tic-tac-toe 70](#_Toc29891348)

[Hình 4.8: So sánh về thời gian thực thi trên CSDL Breast cancer 72](#_Toc29891349)

[Hình 4.9: So sánh về thời gian thực thi trên CSDL Chess 72](#_Toc29891350)

[Hình 4.10: So sánh về thời gian thực thi trên CSDL Diabetes 73](#_Toc29891351)

[Hình 4.11: So sánh về thời gian thực thi trên CSDL Tic-tac-toe 73](#_Toc29891352)

[Hình 4.12: Kết quả so sánh độ tương tự khi phân cụm 74](#_Toc29891353)

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Phương pháp phân lớp dựa trên luật kết hợp đã được nghiên cứu và chứng minh là tốt hơn các phương pháp dựa vào luật truyền thống như ILA, ID3… Tuy nhiên, trên thực tế, các dữ liệu bị mất cân bằng về lớp khá phổ biến tức là sẽ có một số lớp với số mẫu vượt trội hơn so với những lớp còn lại, điều này ảnh hưởng không nhỏ đến quá trình xây dựng bộ phân lớp và dự đoán lớp. Đặc biệt trong quá trình xây dựng bộ phân lớp, nếu chúng ta chọn ngưỡng độ hỗ trợ tối thiểu *(minSup)* không phù hợp sẽ dẫn đến việc các mẫu thuộc các lớp thiểu số sẽ không phổ biến hoặc các luật được khai thác sẽ thiên về các mẫu thuộc lớp đa số nhiều hơn.

Để giải quyết vấn đề trên, Nguyễn và đồng sự (năm 2016) đã đề xuất phương pháp phân cụm dữ liệu bằng thuật toán K-means để cân bằng số mẫu của mỗi lớp, sau đó sử dụng thuật toán CAR-Miner để khai thác tập luật phân lớp[17]. Qua nghiên cứu đã chứng minh được sự cải thiện đáng kể về độ chính xác so sánh giữa có và không có thực hiện cân bằng lớp. Tuy nhiên, tôi nhận thấy rằng với nghiên cứu này vẫn còn tồn tại những hạn chế của kỹ thuật phân cụm K-means và thuật toán CAR-Miner:

- Phân cụm K-means: ưu điểm là thời gian thực hiện tương đối nhanh nhưng không đảm bảo được độ tương tự giữa các thành phần trong cụm là đủ tốt, không thể xử lý nhiễu và ngoại lai.

- CAR-Miner là một thuật toán khai thác luật phân lớp khá hiệu quả dựa trên cấu trúc cây MECR-tree (Modified Equivalence Class-Rules tree). Tuy nhiên thuật toán này lại tiêu tốn nhiều bộ nhớ cho việc lưu trữ các Obidset (tập các nhận dạng đối tượng chứa itemset) của itemset và đòi hỏi thời gian tính toán cho giai đoạn giao giữa các tập Obidset với nhau, thời gian này trở nên đáng kể khi xét trong một cơ sở dữ liệu lớn.

Chính vì vậy tôi chọn đề tài “**Khai thác luật phân lớp kết hợp trên dữ liệu mất cân bằng về lớp**” góp phần nghiên cứu giải quyết những hạn chế trên, đồng thời từ đó ứng dụng phương pháp đề xuất vào dữ liệu thực tế về an toàn giao thông.

## 2. Mục tiêu nghiên cứu

Xuất phát từ những hạn chế của phương pháp phân cụm K-means trong quá trình cân bằng số mẫu thuộc về các lớp của dữ liệu, trong khuôn khổ của đề tài luận văn tôi đề xuất một phương pháp mới làm tăng độ tương tự của dữ liệu sau khi phân cụm nhằm mục đích tăng độ chính xác cho giai đoạn dự đoán lớp sau này, đồng thời áp dụng thuật toán CAR-Miner-Diff để giải quyết nhược điểm của CAR-Miner trong nghiên cứu ở [17].

Bên cạnh đó, để có thể khai thác kho dữ liệu khổng lồ của lực lượng cảnh sát giao thông về xử phạt vi phạm hành chính phục vụ cho công tác tham mưu, đề xuất các chính sách đảm bảo duy trì trật tự an toàn giao thông trên địa bàn các tỉnh, luận văn sẽ hướng đến áp dụng phương pháp đề xuất vào việc dự đoán khả năng gây tai nạn dựa trên nhận diện về giá trị thuộc tính của các đối tượng khác nhau.

## 3. Đối tượng và phạm vi nghiên cứu

- Đối tượng nghiên cứu: luận văn tập trung nghiên cứu thực hiện các thuật toán trên một số dữ liệu chuẩn như: diabetes, breast-cancer, tic-tac-toe, chess (được lấy từ website UCI http://mlearn.ics.uci.edu) nhằm mục đích so sánh độ chính xác và thời gian thực hiện trên nhiều dữ liệu khác nhau, đồng thời thuật toán đề xuất sẽ được áp dụng để khai thác trên dữ liệu xử phạt vi phạm hành chính giao thông của phòng Cảnh sát giao thông đường bộ - đường sắt công an tỉnh Đắk Nông.

- Phạm vi nghiên cứu: nghiên cứu và vận dụng các thuật toán khai thác luật phân lớp kết hợp đã có từ trước đến nay. Trong đó, tập trung đi sâu phân tích thuật toán CAR-Miner-Diff do Nguyễn và đồng sự đề xuất [17]. Nắm bắt và cải tiến kỹ thuật phân cụm đã được trình bày trong nghiên cứu [19] nhằm cải thiện về độ chính xác và thời gian thực thi. Từ đó tiến hành cài đặt thực nghiệm trên các dữ liệu chuẩn để kiểm chứng kết quả và áp dụng trên dữ liệu thực tế về xử phạt vi phạm hành chính giao thông.

## 4. Cấu trúc luận văn

Luận văn tổ chức thành 05 chương có nội dung như sau:

**- Chương 1:** Giới thiệu tổng quan về khai thác dữ liệu, khai thác luật kết hợp, khai thác luật phân lớp kết hợp.

**- Chương 2:** Trình bày cơ sở lý thuyết về một số phương pháp khai thác luật kết hợp, một số kỹ thuật phân lớp, một số phương pháp khai thác luật phân lớp kết hợp.

**- Chương 3:** Trình bày về các phương pháp phân cụm dữ liệu phổ biến, thuật toán Car-Miner-Diff, phương pháp cải tiến đề xuất.

**- Chương 4:** Trình bày thực nghiệm và đánh giá kết quả về độ chính xác, thời gian thực thi, độ tương tự, chương trình ứng dụng dự đoán ATGT.

**- Chương 5:** Trình bày kết luận và hướng phát triển của luận văn.

# TỔNG QUAN

## 1.1. Giới thiệu về khai thác dữ liệu

### 1.1.1. Khái niệm

Khai thác dữ liệu là một khái niệm ra đời vào những năm cuối của thập niên 1980. Nó là quá trình khám phá những thông tin tiềm ẩn trong các cơ sở dữ liệu và có thể xem như là một bước trong quá trình khám phá tri thức. Khai thác dữ liệu là giai đoạn quan trọng nhất trong tiến trình khai phá tri thức từ cơ sở dữ liệu, các tri thức này hỗ trợ trong việc ra quyết định. Như vậy, những thông tin có giá trị tiềm ẩn trong kho cơ sở dữ liệu sẽ được trích xuất ra và sử dụng một cách hữu ích nhờ khai thác dữ liệu. Chức năng của khai thác dữ liệu gồm có gom nhóm, phân loại, dự báo, dự đoán và phân tích các liên kết [20].

Năm 1989, Fayyad, Smyth và Piatestsky-Shapiro đã dùng khái niệm *Phát hiện tri thức từ CSDL* (Knowledge Discovery in Database - KDD) trong đó khai thác dữ liệu là một giai đoạn rất đặc biệt trong toàn bộ quá trình, nó sử dụng các kỹ thuật để tìm ra các mẫu từ dữ liệu.

Khai thác dữ liệu là quá trình tìm kiếm các mẫu mới, những thông tin tiềm ẩn  
mang tính dự đoán trong các khối dữ liệu lớn. Những công cụ khai thác dữ liệu có thể phát hiện những xu hướng trong tương lai, các tri thức mà khai thác dữ liệu mang lại cho các doanh nghiệp, các cơ quan quản lý chính sách có thể đưa ra các quyết định kịp thời và trả lời những câu hỏi trong lĩnh vực kinh doanh mà trước đây tốn khá nhiều thời gian để xử lý.

### 1.1.2. Các dạng dữ liệu có thể khai thác

Có rất nhiều kiểu dữ liệu có thể được khai thác và các dạng dữ liệu điển hình mà chúng ta có thể tiến hành khai thác là:

- Cơ sở dữ liệu quan hệ: là các cơ sở dữ liệu tác nghiệp được tổ chức theo mô  
hình dữ liệu quan hệ. Hầu hết các hệ quản trị cơ sở dữ liệu đều hỗ trợ dữ liệu dạng này như Oracle, SQL Server, MS Access, ...

- Cơ sở dữ liệu đa chiều: là các kho dữ liệu được tập hợp, chọn lọc từ nhiều nguồn dữ liệu khác nhau. Dạng dữ liệu này mang tính lịch sử (tức là có tính thời gian) và chủ yếu phục vụ cho quá trình phân tích cũng như khai phá tri thức nhằm hỗ trợ cho việc ra quyết định.

- Cơ sở dữ liệu dạng giao dịch (transactional database): là dạng cơ sở dữ liệu tác nghiệp nhưng bản ghi thường là các giao dịch. Dạng dữ liệu này thường phổ biến trong lĩnh vực thương mại và ngân hàng.

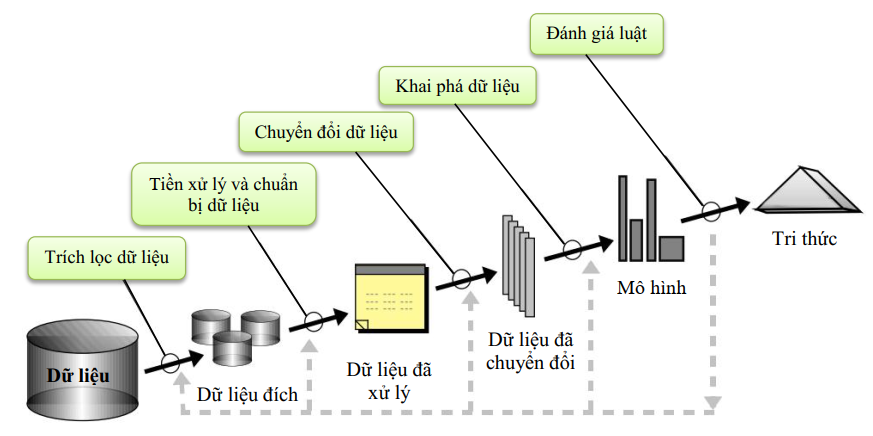
- Cơ sở dữ liệu quan hệ - đối tượng : là dạng cơ sở dữ liệu lai giữa hai mô hình  
quan hệ và hướng đối tượng.

- Dữ liệu không gian và thời gian (spatial, time - series data): là dạng dữ liệu có tích hợp thuộc tính về không gian như dữ liệu bản đồ hoặc dữ liệu tích hợp thuộc tính về thời gian như dữ liệu chứng khoán.

- Cơ sở dữ liệu đa phương tiện: là dạng dữ liệu âm thanh, hình ảnh, phim ảnh, văn bản hay www, ... Dạng dữ liệu này phổ biến trên Internet do sự ứng dụng rộng rãi của nó.

### 1.1.3. Quá trình khai thác dữ liệu

Quá trình khai thác dữ liệu sẽ tiến hành qua 06 giai đoạn:



Hình .: Quá trình khai thác dữ liệu [21]

Bắt đầu của quá trình là kho dữ liệu thô và kết thúc với tri thức được trích xuất ra. Đây là một quá trình rất phức tạp gặp khá nhiều khó khăn như: quản lý các tập dữ liệu, phải lặp đi lặp lại các quá trình,..

*1.1.3.1. Gom dữ liệu (Gathering)*

Tập hợp dữ liệu là bước đầu tiên trong quá trình khai thác dữ liệu. Đây là bước được khai thác trong một cơ sở dữ liệu, một kho dữ liệu và thậm chí các dữ liệu từ các nguồn ứng dụng Web.

*1.1.3.2. Trích lọc dữ liệu (Selection)*

Ở giai đoạn này dữ liệu được lựa chọn hoặc phân chia theo một số tiêu chuẩn nào đó, ví dụ chọn tất cả những người có chức vụ là Trưởng phòng và có trình độ đại học.

*1.1.3.3. Làm sạch, tiền xử lý và chuẩn bị trước dữ liệu (Cleansing, Pre-processing and Preparation)*

Giai đoan thứ ba này là giai đoạn hay bị bỏ quên, nhưng thực tế nó là một bước rất quan trọng trong quá trình khai thác dữ liệu. Một số lỗi thường mắc phải trong khi gom dữ liệu là tính không đủ chặt chẽ và logic. Vì vậy, dữ liệu thường chứa các giá trị vô nghĩa và không có khả năng kết nối dữ liệu, hay thiếu dữ liệu trên một số trường nào đó. Giai đoạn này sẽ tiến hành xử lý những dạng dữ liệu không chặt chẽ nói trên. Những dữ liệu dạng này được xem như thông tin dư thừa, không có giá trị. Bởi vậy, đây là một quá trình rất quan trọng vì dữ liệu này nếu không được *“làm sạch - tiền xử lý - chuẩn bị trước”* thì sẽ dẫn đến kết quả sai lệch trong quá trình khai thác dữ liệu.

*1.1.3.4. Chuyển đổi dữ liệu (Transformation)*

Tiếp theo là giai đoạn chuyển đổi dữ liệu, dữ liệu đưa ra có thể sử dụng và điều khiển được bởi việc tổ chức lại nó. Dữ liệu đã được chuyển đổi phù hợp với mục đích khai thác.

*1.1.3.5. Khai thác dữ liệu (Data mining)*

Đây là bước mang tính tư duy trong khai thác dữ liệu. Ở giai đoạn này nhiều thuật toán khác nhau đã được sử dụng để trích ra các mẫu từ dữ liệu. Thuật toán thường dùng là những thuật toán phân lớp, phân cụm, hay dựa trên luật kết hợp...

*1.1.3.6. Đánh giá kết quả mẫu/luật (Evaluation of Result/Rule)*

Đây là giai đoạn cuối trong quá trình khai thác dữ liệu. Ở giai đoạn này, các mẫu dữ liệu được trích xuất ra bởi phần mềm khai thác dữ liệu. Không phải bất cứ mẫu dữ liệu nào cũng đều hữu ích, đôi khi có trường hợp bị sai lệch. Vì vậy, cần phải ưu tiên những tiêu chuẩn đánh giá để trích xuất ra các tri thức (Knowledge) cần thiết.

### 1.1.4. Các hướng tiếp cận cơ bản

Khai thác dữ liệu có thể được phân chia theo các hướng tiếp cận chính sau:

- Phân lớp và dự đoán (classification and prediction): xếp một đối tượng vào một trong những lớp đã biết. Ví dụ: phân lớp theo học lực của học sinh. Đối với hướng tiếp cận này thường áp dụng một số kỹ thuật như học máy, cây quyết định, mạng neuron nhân tạo ..., bài toán này còn đựoc gọi là học có giám sát.

- Phân cụm (clustering): Sắp xếp các đối tượng theo từng cụm nhưng  
số lượng và tên các cụm chưa biết trước. Bài toán phân cụm còn được gọi là học  
không giám sát.

- Luật kết hợp (Association rules): là dạng biểu diễn tri thức ở dạng khá đơn giản. Ví dụ: “70% số vụ tai nạn xe máy do thanh niên điều khiển thì có tới 80% là do trong hơi thở có nồng độ cồn vượt quá ngưỡng quy định”. Hướng tiếp cận này được ứng dụng trong nhiều lĩnh vực như kinh doanh, y học, giáo dục ...

- Khai thác chuỗi theo thời gian: Cũng tương tự như khai thác dữ liệu bằng luật kết hợp nhưng có thêm tính thứ tự và tính thời gian. Hướng tiếp cận này được ứng dụng nhiều trong lĩnh vực tài chính và thị trường chứng khoán bởi chúng có tính dự đoán cao.

- Mô tả khái niệm (concept description and summarization): Bài toán này thiên  
về mô tả, tổng hợp và tóm tắt khái niệm. Ví dụ: tóm tắt văn bản.

### 1.1.5. Một số lĩnh vực ứng dụng

Phát hiện tri thức và khai thác dữ liệu liên quan đến nhiều ngành, nhiều lĩnh vực như: thống kê, trí tuệ nhân tạo, cơ sở dữ liệu, thuật toán học, tính toán song song và tốc độ cao, thu thập tri thức cho các hệ chuyên gia, quan sát dữ liệu,…Đặc biệt phát hiện tri thức và khai thác dữ liệu rất gần gũi với các lĩnh vực thống kê, sử dung các phương pháp thống kê để mô hình dữ liệu và phát hiện các mẫu, luật,…

Các ứng dụng của phát hiện tri thức và khai thác dữ liệu bao gồm:

•Thông tin thương mại.

•Phân tích dữ liệu marketing, khách hàng.

•Phân tích đầu tư.

•Phê duyệt vay vốn.

* Phát hiện gian lận.
* Thông tin kỹ thuật.
* Điều khiển và lập lịch trình, ....

## 1.2. Giới thiệu về khai thác luật kết hợp

### 1.2.1. Tổng quan

Luật kết hợp là một trong những phương pháp khai phá dữ liệu phổ biến, đặc biệt là việc khai thác những thông tin tiềm ẩn mang tính dự đoán trong các cơ sở dữ liệu lớn. Phương pháp này ra đời và phát triển mạnh trong những năm gần đây. Lần đầu tiên được Rakesh Agrawal, Tomasz Imielinski, Arun Swami đề xuất năm 1994. Sau đó đến năm 1996 được Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, A.Inkeri Verkanmo tiếp tục phát triển và cải tiến. Đến nay những nghiên cứu về luật kết hợp tập trung xây dựng thuật toán khai phá luật kết hợp mới, hiệu quả hoặc cải tiến, phát triển các thuật toán hiệu quả hơn. Mỗi luật kết hợp được thể hiện dưới dạng một mệnh đề gồm có tiền đề và kết luận. Trong đó tiền đề, kết luận có thể bao gồm một hay nhiều tập mục (thuộc tính đối tượng, trường dữ liệu trong cơ sở dữ liệu). Và đối với mỗi luật ta thường quan tâm tới các độ đo sau:

*- Độ hỗ trợ của tập mục (Support):* đó là số phần trăm của các tác vụ (giao dịch, bản ghi, hồ sơ) có chứa tập mục đó.

*- Độ tin cậy của luật (Confidence):* đó là số phần trăm các tác vụ (giao dịch, bản ghi, hồ sơ) có chứa các tập mục nằm trong tiền đề thì cũng chứa các tập mục nằm trong phần kết luận của luật.

Luật kết hợp là những luật có dạng như:

- 25% người dân gốc Tây Nguyên và trong độ tuổi từ 20 - 30 , 15% số vụ tai nạn xảy ra là người dân Tây Nguyên và trong độ tuổi từ 20 - 30 .

- 35% vụ chạy xe quá tốc độ, trong đó 18% số nạn nhân vừa chạy xe quá tốc độ vừa say bia rượu.

Trong các ví dụ nêu trên tiền đề của luật là: “gốc Tây Nguyên”, “độ tuổi từ 20 - 30”, “chạy xe quá tốc độ”, trong khi đó “tai nạn” và “say bia rượu” là kết luận của luật.

Các con số như: 25%, 35% là độ hỗ trợ của luật (support ), còn 15% và 18% là độ tin cậy của luật (confidence).

Thông tin do luật kết hợp mang lại như trên thường mang tính đúc kết, những mối liên hệ chưa được biết trước và mang tính dự đoán, dự báo những mỗi liên hệ tiềm ẩn trong cơ sở dữ liệu tác nghiệp.

Luật kết hợp là dạng luật khá đơn giản nhưng mang rất nhiều ý nghĩa. Thông tin mà dạng luật này đem lại là rất đáng kể và hỗ trợ không nhỏ trong quá trình ra quyết định. Những luật kết hợp phổ biến và mang nhiều thông tin từ cơ sở dữ liệu nghiệp vụ là một trong nhiều hướng tiếp cận chính của lĩnh vực khai thác dữ liệu.

### 1.2.2. Một số khái niệm cơ bản

Gọi:

I *=* {I1, I2,..., Im} là tập m thuộc tính riêng biệt, mỗi thuộc tính gọi là một mục.

D là một CSDL quan hệ, trong đó mỗi bản ghi là một tác vụ và chứa các tập mục.

T là tập các tác vụ.

*1.2.2.1. Luật kết hợp*

Luật kết hợp (Association Rule): Một luật kết hợp là quan hệ có dạng X →Y, trong đó X và Y là các tập mục thỏa mãn điều kiện: X ⊆I, Y ⊆I, X ∩Y = ∅.

Đối với luật kết hợp X →Y, X được gọi là tiền đề và Y được gọi là kết luận của luật.

*1.2.2.2. Độ hỗ trợ của một tập mục (support)*

Độ hỗ trợ (support) của một tập mục X trong tập các tác vụ D, ký hiệu: *Sup(X)* là tỉ số giữa số tác vụ T (của D) chứa X và tổng số các tác vụ của D. Một cách khái quát ta có thể biểu diễn như sau:



Độ hỗ trợ của một tập mục có giá trị giữa 0 và 1, tức là 0 ≤*Sup(X)* ≤1, ∀X

*1.2.2.3. Tập mục phổ biến (frequent itemset)*

Tập mục X thỏa điều kiện: *Sup(X)* ≥*minSup* (trong đó ngưỡng *minSup* là một giá trị cho trước) được gọi là tập mục phổ biến với độ hỗ trợ tối thiểu *minSup*.

*1.2.2.4. Các độ đo của một luật*

* Độ hỗ trợ của một luật (support):

Cho luật r = X →Y, độ hỗ trợ của luật r ký hiệu là *Sup(r)* được xác định như sau: *Sup(r)* = *Sup(X∪Y)*.

* Độ tin cậy của một luật (confidence):

Luật r = X →Y có độ tin cậy conf trong cơ sở dữ liệu D nếu conf là tỉ số giữa số các tác vụ T (của D) chứa X thì cũng chứa Y và tổng số các tác vụ của D chứa X. Hay đó chính là xác suất có điều kiện P(Y/X). Ta ký hiệu độ tin cậy của luật r là *Conf(r)*. Độ tin cậy của một luật cũng có giá trị giữa 0 và 1.

*Sup* (X →Y) = P (X ∪Y)

*Conf* (X →Y) = P (Y/X) = *Sup*(X∪Y)/*Sup*(X)

* Luật kết hợp mạnh (strong):

Các luật thỏa mãn cả hai ngưỡng là độ hỗ trợ tối thiểu *minSup* và độ tin cậy tối thiểu được gọi là luật kết hợp mạnh, tức là:

*Sup*(X→Y) = P(X ∪Y) ≥*minSup*

*Conf*(X→Y) = P(Y/X) = *Sup*(X∪Y)/*Sup*(X)≥*minConf*

Người ta thường biểu diễn bằng % thay cho các giá trị từ 0 đến 1.

### 1.2.3. Một số tính chất với tập mục phổ biến

Tính chất 1: *Độ hỗ trợ của tập con*

Nếu A ⊆B với A, B là các tập mục thì *Sup*(A) ≥*Sup*(B).

Điều này là hiển nhiên vì tất cả các tác vụ trong D hỗ trợ B thì cũng hỗ trợ A.

Tính chất 2: *Một tập chứa một tập không phổ biến thì cũng là tập không phổ biến*

Nếu tập A không là tập phổ biến, tức là *Sup*(A) < *minSup* thì tập B chứa A cũng không là tập phổ biến vì:

*Sup*(B) ≤*Sup*(A) < *minSup* (theo tính chất 1)

Tính chất 3: *Các tập con của tập phổ biến cũng là tập phổ biến*

Nếu B là tập phổ biến trong D tức là: *Sup*(B) ≥*minSup*. Khi đó mọi tập con A của B cũng là tập phổ biến vì *Sup*(A) ≥*Sup*(B)≥*minSup* (theo tính chất 1).

Nếu tập A = {i1, i2,…, ik} là tập phổ biến thì mọi tập con có (k-1) mục của nó cũng là phổ biến. Lưu ý trường hợp ngược lại là không đúng.

## 1.3. Giới thiệu về khai thác luật phân lớp kết hợp

### 1.3.1. Một số định nghĩa và khái niệm

Khai thác luật phân lớp dựa vào khai thác luật kết hợp (**C**lass **A**ssociaton **R**ule**s** - CARs) là tìm một tập con của các luật kết hợp có trong cơ sở dữ liệu. Mỗi luật trong tập con này chứa vế phải là giá trị của thuộc tính lớp [22].

Mục tiêu của khai thác luật phân lớp dựa vào khai thác luật kết hợp là:

- Khai thác tập CARs thỏa ngưỡng độ hỗ trợ tối thiểu (*minSup*) và ngưỡng độ tin cậy tối thiểu (*minConf*).

- Xây dựng bộ phân lớp từ CARs.

Một cách hình thức, bài toán khai thác luật phân lớp kết hợp CARs được phát biểu như sau:

Gọi *D* là tập dữ liệu huấn luyện với n thuộc tính *A1, A2,...,An* và *|D|* đối tượng (mẫu). Gọi *C = {c1, c2,..., ck}* là tập các nhãn lớp. Mỗi giá trị của thuộc tính *Ai* và thuộc tính lớp *C* được ký hiệu bởi các ký tự thường *a* và *c* tương ứng.

Định nghĩa 1: Một tập itemset là một tập các cặp (thuộc tính, giá trị) được ký hiệu bởi {*(Ai1,ai1), (Ai2,ai2),..., (Aim,aim)*}.

Định nghĩa 2: Một luật phân lớp kết hợp r có dạng {*(Ai1,ai1), (Ai2,ai2),..., (Aim,aim)*} → *c,* trong đó {*(Ai1,ai1), (Ai2,ai2),..., (Aim,aim)*} là một itemset và *c∈ C* là một nhãn lớp.

Định nghĩa 3: Khả năng xảy ra của luật r, kí hiệu *ActOcc(r)*, là số dòng trên *D* chứa vế trái của *r.*

Định nghĩa 4: Đếm độ hỗ trợ của luật *r*, kí hiệu *Sup(r)*, là số dòng trên *D* chứa vế trái và vế phải của *r*.

Định nghĩa 5: Độ tin cậy của luật *r* là tỉ số giữa *Sup(r)* chia cho *ActOcc(r),* nghĩa là 

Bảng .: Một CSDL huấn luyện mẫu [10]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OID** | **A** | **B** | **C** | **Lớp** |
| 1 | a1 | b1 | c1 | y |
| 2 | a1 | b2 | c1 | n |
| 3 | a2 | b2 | c1 | n |
| 4 | a3 | b3 | c1 | y |
| 5 | a3 | b1 | c2 | n |
| 6 | a3 | b3 | c1 | y |
| 7 | a1 | b3 | c2 | y |
| 8 | a2 | b2 | c2 | n |

Ví dụ: Xét luật *r* : {(*A,a1*)} → *y* từ CSDL Bảng 1.1, ta có *ActOcc(r) = 3* và *Sup(r) =* 2. Do có 3 đối tượng có giá trị *A = a1,* trong đó 2 đối tượng có lớp là *y,* suy ra độ tin cậy của luật là *Conf(r) = 2/3.*

### 1.3.2. So sánh giữa luật kết hợp và luật phân lớp kết hợp

Luật phân lớp kết hợp (CARs) là tập con của luật kết hợp, vì trong đó thực sự có chứa các thuộc tính lớp và không phải là lớp, thêm vào đó cà hai loại luật đều được biểu diễn dưới dạng luật If - Then. Tuy nhiên, luật kết hợp và luật phân lớp kết hợp cũng có những điểm khác biệt như sau:

- Hai dạng luật có những mục đích biểu diễn khác nhau. Luật kết hợp thường được sử dụng như các công cụ mô tả. Mặt khác, luật phân lớp kết hợp được sử dụng như một phương tiện để dự đoán nhãn lớp cho những dữ liệu mới.

- Các giải thuật khác nhau được sử dụng để khai thác hai dạng luật khác nhau này. Khai thác luật kết hợp thường bao gồm hai bước: (1) Tìm kiếm tập mục phổ biến, nghĩa là, tất cả các tập mục itemset có độ hỗ trợ lớn hơn hoặc bằng một ngưỡng cho trước, (2) giai đoạn sinh luật kết hợp dựa trên các tập mục phổ biến. Khai thác luật phân lớp kết hợp cũng bao gồm hai bước: (1) Sử dụng phương pháp heuristic để chọn các cặp giá trị thuộc tính hình thành nên các điều kiện của luật, (2) sử dụng các phương pháp, chiến lược cắt tỉa ứng viên để tránh các phân biệt nhỏ, nghĩa là các luật với tiền đề quá cụ thể. Bước thứ hai cắt tỉa ứng viên phải được thực hiện mặc dù càng nhiều luật cụ thể sẽ dẫn đến độ chính xác cao hơn cho dữ liệu huấn luyện, tuy nhiên nó lại không đáng tin cậy đối với những dữ liệu chưa biết, trường hợp này gọi là *overfitting*. [22]

- Thuật toán khai thác luật kết hợp thường sinh nhiều luật hơn thuật toán khai thác luật phân lớp kết hợp. Một thuật toán để khai thác luật kết hợp sẽ sinh tất cả các luật đáp ứng yêu cầu về độ hỗ trợ và độ tin cậy. Điều kiện không có cắt tỉa và xếp hạng, các thuật toán khác nhau để khai thác luật kết hợp sẽ có kết quả giống nhau. Ngược lại, hầu hết các thuật toán để khai thác luật phân lớp kết hợp sinh luật bảo đảm đủ để bao phủ các dữ liệu huấn luyện thay vì phải sinh tất cả các luật có thể sinh được từ tập dữ liệu. Do đó, các thuật toán khác nhau cho ra các tập luật khác nhau.

- Các thuật toán để tạo ra hai dạng luật được đánh giá khác nhau. Vì kết quả của các thuật toán khai thác luật kết hợp là như nhau, thời gian chạy và tiêu thụ bộ nhớ chính là những tiêu chuẩn quan trọng hàng đầu để so sánh. Đối với khai thác luật phân lớp kết hợp, việc so sánh chủ yếu dựa trên độ chính xác dự đoán của luật trên dữ liệu kiểm tra.

- Hai dạng luật được đánh giá theo những cách khác nhau. Luật kết hợp thường được đánh giá bởi người dùng, trong khi luật phân lớp kết hợp thông thường đánh giá bằng cách áp dụng chúng vào dữ liệu kiểm tra.

# CƠ SỞ LÝ THUYẾT

## 2.1. Một số phương pháp khai thác luật kết hợp

### 2.1.1. Giải thuật Apriori

*2.1.1.1. Giới thiệu:*

Apriori là thuật toán khai thác các tập mục phổ biến cho các luật kết hợp boolean. Trong phần này, tôi sẽ trình bày thuật toán Apriori do Rakesh Agrawal, Tomasz Imielinski, Arun Swami đề xuất lần đầu tiên vào năm 1994.

Vấn đề phát hiện tất cả các luật kết hợp có độ hỗ trợ và độ tin cậy vượt qua  
ngưỡng xác định nào đó (ngưỡng này do người dùng đưa ra, nó chính là độ hỗ  
trợ tối thiểu *minSup* và độ tin cậy tối thiểu *minConf*) có thể được phân chia thành hai vấn đề chính sau đây:

- Tìm tất cả các tập mục phổ biến với *minSup* cho trước. Thuật toán Apriori chủ yếu giải quyết vấn đề này.

- Sử dụng các tập mục phổ biến để sinh ra các luật kết hợp với *minConf* cho trước.

Ý tưởng chính ở đây là nếu S và X là tập mục phổ biến và X ⊂S, khi đó ta có thể xem X →Y (ở đây Y = S\X) có phải là luật mong muốn hay không bằng cách tính độ tin cậy c = *Conf*(X→Y) = *Sup*(X∪Y)/*Sup*(X) = *Sup*(S)/*Sup*(X). Nếu c≥*minConf* thì khi đó X →Y trở thành luật mong muốn.

Thuật toán Apriori sinh ra các tập ứng cử trong một lần duyệt bằng việc sử dụng các tập mục đã được thấy là phổ biến trong lần duyệt trước mà không cần quan tâm đến các tác vụ trong cơ sở dữ liệu. Sở dĩ như vậy là vì bất kỳ tập con nào của tập mục phổ biến cũng là tập mục phổ biến (theo tính chất 2 mục 1.2.3). Vì vậy, các tập ứng cử có (k-1) mục và xóa đi các tập ứng cử viên nếu nó chứa bất kỳ một tập con nào mà không là phổ biến. Thủ tục này dẫn đến khả năng tỉa bớt một số các tập ứng viên không thỏa mãn tính chất, hay nói cách khác là khá hiệu quả trong việc rút gọn không gian tìm kiếm.

*2.1.1.2. Mô tả thuật toán:*

Mục đích của thuật toán: Tìm các tập mục phổ biến dựa trên các ứng cử viên.

Đầu vào:

- Cơ sở dữ liệu: D

- Ngưỡng độ hỗ trợ tối thiểu: *minSup*

Đầu ra:

- Tập các tập mục phổ biến.

Nội dung của thuật toán:

*Gọi Ck*: Tập các ứng viên có kích thước k

*Lk*: Các tập phổ biến có kích thước k

Quét cơ sở dữ liệu để tìm các tập 1 mục phổ biến L1:

***L1*** = {i ∈ *I:* σ(i) ≥ *minSup*}

for (*k* = 2; *Lk-1* !=∅; *k*++) do

***Ck*** = {các ứng viên được tạo từ *Lk-1* }

for each *t* ∈ D do

for each *c* ∈ Ck do

if *c* ⊆ *t then c.*count++

***Lk*** = {c*Ck | c*.count ≥ *minSup*}

**FIs** = ∪k*L*k; *//Tất cả các tập mục phổ biến đã khai thác được [23]*

Các bước khai thác tập mục phổ biến theo thuật toán Apriori như sau:

- Bước 1: Xây dựng danh sách các ứng viên (k-1)-itemsets và sau đó trích chọn ra danh sách tập mục phổ biến của (k-1)-itemsets dùng ngưỡng *minSup.*

- Bước 2: Sau đó sử dụng các tập mục phổ biến (k-1)-itemsets này để xác định danh sách ứng viên và tập phổ biến của k-itemsets.

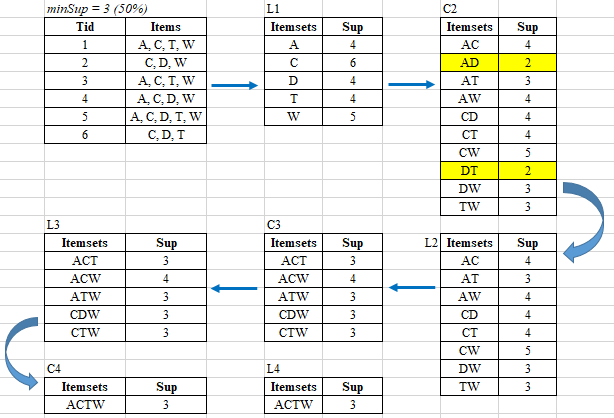
- Bước 3: Loại bỏ các tập mục không phổ biến.

- Bước 4: Lặp lại cho đến khi danh sách ứng viên và tập phổ biến của (k-1)-itemsets rỗng.

- Bước 5: Trả về danh sách các tập mục phổ biến đã khai thác được.

Ta có thể xem chi tiết mô tả thuật toán qua ví dụ cụ thể hình 2.1:

- Đầu vào là CSDL với 06 giao dịch, danh sách các item {A,C,D,T,W}, ngưỡng *minSup* = 3.

**

Hình .: Minh họa thuật toán Apriori [22]

Với mỗi tập mục phổ biến thu được, ta có thể sinh ra tất cả các tập con không rỗng. Với mỗi tập con này sẽ sinh ra các luật kết hợp dạng X →Y.

Với mỗi luật kết hợp, lần lượt duyệt qua tất cả các giao dịch, ta sẽ chọn các luật có độ tin cậy *Conf(r)* ≥ *minConf*.

Ví dụ: với tập mục phổ biến I = {A,C,T}

- Các tập con của I: {A}, {C}, {T}, {A,C},{A,T},{C,T}

- Ta sẽ có các luật kết hợp sau:

{A} → {C,T}; {C} → {A,T}; {T} → {A,C};

{A,C} → {T}; {A,T} → {C}; {C,T} → {A}

Với minConf = 80%. Ta có 1 luật kết hợp thỏa mãn là: {A,T} → {C} *(Các thông số conf được tính theo mục 1.2.2.4).*

*2.1.1.3. Đánh giá thuật toán:*

- Lựa chọn giá trị ngưỡng *minSup*: Giá trị *minSup* quá thấp sẽ sinh ra nhiều  
tập mục phổ biến. Điều này sẽ làm tăng số lượng tập mục phải xét.

- Số lượng các mục trong cơ sở dữ liệu (các giao dịch): Cần thêm bộ nhớ để  
lưu giá trị độ hỗ trợ với mỗi mục. Nếu số lượng các tập mục phổ biến mức 1 tăng lên thì chi phí duyệt các giao dịch cũng tăng.

- Kích thước của cơ sở dữ liệu (các giao dịch): Giải thuật phải duyệt cơ sở  
dữ liệu nhiều lần, do đó chi phí tính toán của Apriori tăng lên khi số lượng các giao  
dịch tăng lên.

- Kích thước trung bình của các giao dịch: Khi kích thước (số lượng các  
mục) trung bình của các giao dịch tăng lên, thì độ dài tối đa của các tập mục phổ biến cũng tăng lên.

### 2.1.2. Giải thuật FP-Growth

*2.1.2.1. Giới thiệu:*

Thuật toán có tên FP-Growth được giới thiệu bởi Jiawei Han, Jian Pei và Yiwen năm 2000. Thuật toán này hiệu quả hơn Apriori bởi 3 kỹ thuật chính sau:

- Nó mở rộng cấu trúc cây tiền tố được gọi là cây mẫu phổ biến (frequent pattern hoặc gọi tắt là FP-tree) dùng để nén tới cấu trúc dữ liệu nhỏ hơn. Các nút của cây được sắp xếp sao cho các nút phổ biến hơn xuất hiện thường xuyên hơn để có thể dễ dàng chia sẻ với các nút xuất hiện ít hơn.

- Phương pháp khai phá phát triển (growth) từng đoạn dựa trên FP-tree gọi là  
phương pháp FP-Growth. Bắt đầu từ mẫu phổ biến có độ dài 1-itemset, FP-Growth xem xét mẫu phụ thuộc của nó như là cơ sở dữ liệu con bao gồm tập các mục phổ biến có cùng hậu tố, xây dựng FP-tree phụ thuộc tương ứng của nó và thực hiện khai phá đệ quy trên cây này. Mẫu phát triển là nhận được qua việc nối mẫu hậu tố với một đoạn mới được sinh ra từ FP-tree phụ thuộc. Khai phá dựa trên FP-tree thực hiện theo cách phát triển các đoạn mẫu để tránh chi phí trong việc sinh ra số lớn các ứng cử.

- Kỹ thuật tìm kiếm được dùng ở đây là dựa vào nguyên tắc chia để trị, phân rã  
nhiệm vụ khai phá thành tập các nhiệm vụ nhỏ và luôn chỉ cần duyệt cơ sở dữ liệu 2 lần.

*2.1.2.2. Mô tả thuật toán:*

Thuật toán FP-Growth được thực hiện như sau:

- Đầu tiên, duyệt cơ sở dữ liệu để tính độ hỗ trợ của từng mục (đếm số lần xuất  
hiện của từng mục).

- Loại những mục không thỏa mãn độ hỗ trợ *minSup*.

- Các mục còn lại được sắp theo thứ tự giảm dần của độ hỗ trợ (tức là giảm dần theo số lần xuất hiện trong cơ sở dữ liệu) ta nhận được danh sách L các mục đã sắp xếp.

- Duyệt cơ sở dữ liệu lần thứ 2, với mỗi tác vụ *t* ta loại các mục không thỏa độ hỗ trợ, các mục còn lại theo thứ tự giống như xuất hiện trong L (tức là thứ tự giảm dần theo độ hỗ trợ) được xếp vào cây FP-tree (xây dựng FP-tree tại lần duyệt thứ 2).

- Cuối cùng, tìm các mẫu phổ biến trên cây FP-tree đã xây dựng được mà  
không cần duyệt cơ sở dữ liệu thêm một lần nào nữa.

Ta xét ví dụ cụ thể được cho trong bảng 2.1 sau để minh họa cho thuật toán này. Giả sử ta xét với cơ sở dữ liệu có độ hỗ trợ tối thiểu *minSup = 50%.*

Bảng .: Minh họa kết quả bước 1 của thuật toán FP-Growth

|  |  |  |
| --- | --- | --- |
| **Tid** | **Các mục** | **Các mục phổ biến đã sắp** |
| 1 | A, C, T, W | C, W, A, T |
| 2 | C, D, W | C, W, D |
| 3 | A, C, T, W | C, W, A, T |
| 4 | A, C, D, W | C, W, A, D |
| 5 | A, C, D, T, W | C, W, A, D, T |
| 6 | C, D, T | C, D, T |

*- Bước 1:*Duyệt cơ sở dữ liệu, đếm số lần xuất hiện của từng mục, loại các mục không thỏa độ hỗ trợ tối thiểu.

*- Bước 2:*Sắp xếp các mục đáp ứng yêu cầu về độ hỗ trợ theo thứ tự giảm dần của độ hỗ trợ, ta nhận được danh sách L sau:

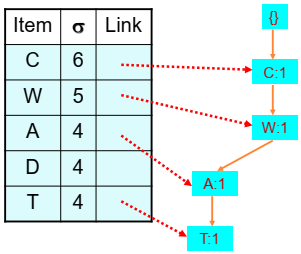
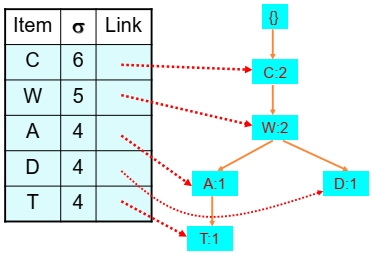
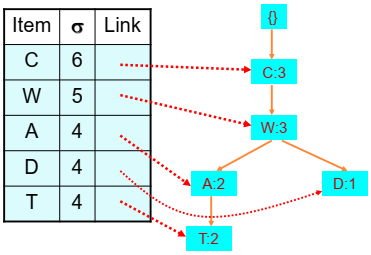
*- Bước 3:*Duyệt lại cơ sở dữ liệu lần thứ hai và xây dựng cây FP-tree. Cây này được xây dựng như sau:

+ Khởi tạo cây T, gốc của cây có nhãn null.

+ Duyệt cơ sở dữ liệu lần thứ 2, với mỗi tác vụ, loại các mục không phổ biến, các mục còn lại sắp theo thứ tự giảm dần của số lần xuất hiện (xem bảng 2.2), dãy các mục phổ biến đó được thêm vào cây cùng với thay đổi số đếm của các mục trên cây cho phù hợp. Quá trình xây dựng cây như hình 2.2 và hình 2.3.

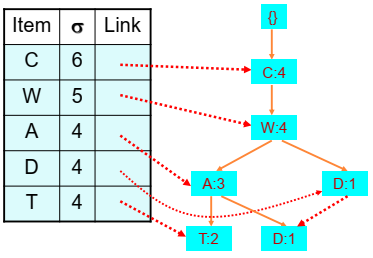
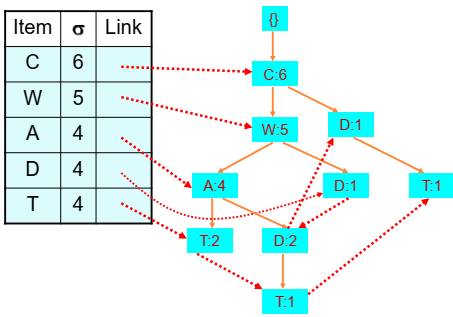
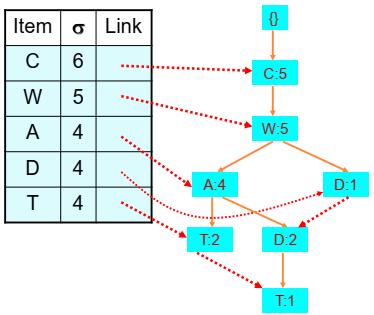
Bảng .: Danh sách L các mục đã được sắp xếp theo độ hỗ trợ giảm dần

| **Mục** | **Tần số** |
| --- | --- |
| C | 6 |
| W | 5 |
| A | 4 |
| D | 4 |
| T | 4 |

*(a) (b) (c)*

Hình .: Quá trình xây dựng cây từ tác vụ 1, 2, 3 [23]

(a) (b) (c)

Hình .: Tiếp tục xây dựng cây từ tác vụ 4, 5, 6 [23]

+ Trong đó:

\* Hình 2.2a: Cây FP-tree được xây dựng khi đưa tác vụ thứ 1 vào.

\* Hình 2.2b: Cây FP-tree được xây dựng khi đưa thêm tác vụ thứ 2 vào.

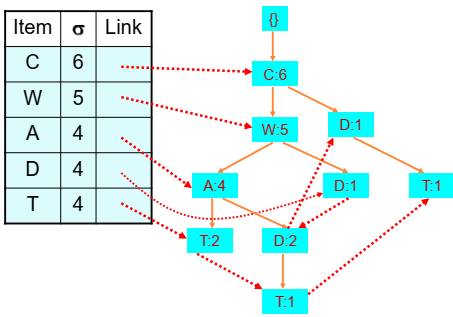
\* Hình 2.2c: Cây FP-tree được xây dựng khi đưa thêm tác vụ thứ 3 vào.

\* Hình 2.3a: Cây FP-tree được xây dựng khi đưa thêm tác vụ thứ 4 vào.

\* Hình 2.3b: Cây FP-tree được xây dựng khi đưa thêm tác vụ thứ 5 vào.

\* Hình 2.3c: Cây FP-tree được xây dựng khi đưa thêm tác vụ thứ 6 vào.

Cuối cùng ta có cây FP-tree sau khi xây dựng xong là:



Hình .: Các đầu mục liên kết và FP-tree với CSDL [23]

- Gốc của cây có nhãn null, các đường đi trên cây biểu diễn item prefixs.

- Các liên kết trên cây: liên kết các mục xuất hiện có tên giống nhau.

- Mỗi nút (trừ nút gốc) bao gồm:

+ Tên mục (item identifier).

+ Số đếm (count).

+ Liên kết đến nút tiếp theo trên cây có cùng tên (Node link).

- Bảng các đầu mục phổ biến (header table): Bắt đầu cho các liên kết.

Sau khi đã xây dựng xong cây, ta dùng thuật toán FP-Growth để tìm các tập mục phổ biến, cụ thể như sau:

- Bắt đầu từ dưới lên (bắt đầu từ mục có tần suất nhỏ tới các mục có tần suất lớn hơn) của bảng header và cây. Ví dụ: Đầu tiên xét mục T, sau đó đến D… như sau:

***Mục T:***

Có 3 đường:

- C:6, W:5, A:4, T:2

- C:6, W:5, A:4, D:5, T:1

- C:6, D:1, T:1

Chiếu trên nút T, ta có CSDL cục bộ như sau:

- {CWA:2, CWAD:1, CD:1}

Các tập phổ biến chứa T được xác định dựa vào CSDL cục bộ này.

Việc tìm các tập phổ biến là tìm các tập con của tập {C, W, A}. Ta có các tập con: {∅,A:3,W:3,C:4,AW:3,AC:3,WC:3, AWC:3}.

Chiếu trên T sinh ra các tập phổ biến là: {T:4, TA:3, TW:3, TC:4, TAW:3, TAC:3, TWC:3, TAWC:3}.

Tương tự đối với các mục D, A, W và C tìm theo cách đệ quy các mẫu phổ biến trên FP-tree phụ thuộc, đầu tiên cho D sau đó cho A, W và C.

Sau khi đã khai thác tất cả các tập mục phổ biến đối với từng mục, ta tiến hành khai thác các luật kết hợp như mục 2.1.1.2.

*2.1.2.3. Đánh giá thuật toán:*

Như đã phân tích ở trên, thuật toán này hiệu quả hơn thuật toán Apriori, thực hiện rất tốt cho mẫu phổ biến ngắn cũng như dài:

*- Độ phức tạp về thời gian:* Chỉ duyệt cơ sở dữ liệu 2 lần, thời gian xây dựng cây là O(n), với n là số các tác vụ của cơ sở dữ liệu, tức là tuyến tính với số các tác vụ.

*- Độ phức tạp về không gian:* O(n) với n là số các tác vụ của cơ sở dữ liệu, độ sâu của cây được giới hạn bởi kích thước của tác vụ lớn nhất.

- Thuật toán không bao giờ bị ngắt bởi một mẫu dài nào của mọi tác vụ. Cây FP-tree duy trì đầy đủ thông tin cho khai thác các mẫu phổ biến. Đồng thời thuật toán cũng rút gọn hợp lý các thông tin cần thiết - các mục không phổ biến đã bị loại bỏ ngay*.*

- Kỹ thuật sắp theo trật tự giảm dần của tần số, điều đó dẫn đến các mục phổ biến hơn được chia sẻ nhiều hơn. Cây FP-tree không bao giờ lớn hơn cơ sở dữ liệu gốc.

- Là phương pháp không sinh ứng viên, thường rất hiệu quả trên các CSDL có mật độ trùng lắp dữ liệu cao.

### 2.1.3. Phương pháp IT-tree

*2.1.3.1. Cấu trúc IT-tree và các lớp tương đương:*

Cho X ⊆*I* (với I là tập tất cả các item đơn trong CSDL), mỗi nút trên IT-tree gồm 2 thành phần Itemset-Tidset: X×*t*(X) được gọi là IT-pair, thực chất là một lớp tiền tố. Các nút con của X thuộc về lớp tương đương của X vì chúng chia sẻ chung tiền tố X (*t*(X) là tập các giao dịch có chứa X).

Nhận xét về IT-tree:

- σ(X) = |t(X)|: độ phổ biến của tập mục X chính bằng lực lượng của tập *t*(X).

- Chỉ cần kết hợp các phần tử trên cùng một mức của lớp tương đương là đủ để sinh ra các tập phổ biến.

*2.1.3.2. Thuật toán tìm tập mục phổ biến Eclat:*

**ECLAT()**

[∅] = {i∈*I*| σ(i)≥ *minSup*}

ENUMERATE\_FREQUENT([∅])

**ENUMERATE\_FREQUENT([*P*])**

for all *li* ∈ [*P*] do

[*Pi*] = ∅

for all *lj* ∈ [*P*] with j > i do

*X = li* ∪ *lj*

*T = t*(*li*) ∩ *t*(*lj*)

if |*T*| ≥ *minSup* then

[*Pi*] = [*Pi*] ∪ {*X*×*T*}

ENUMERATE\_FREQUENT([*Pi*])

Trong đó *t*(*X*) = {y∈T | *X* xuất hiện trong giao dịch y} được gọi là Tidset của *X*.

Ta sẽ minh họa thuật toán qua ví dụ: Xét CSDL mẫu ở bảng 2.3.

Bảng .: Cơ sở dữ liệu mẫu

|  |  |
| --- | --- |
| **Mã giao dịch** | **Nội dung giao dịch** |
| 1 | A, C, T, W |
| 2 | C, D, W |
| 3 | A, C, T, W |
| 4 | A, C, D, W |
| 5 | A, C, D, T, W |
| 6 | C, D, T |

Bảng .: Định dạng dữ liệu dọc của CSDL

|  |  |
| --- | --- |
| **Mã danh mục** | **Các giao dịch chứa danh mục** |
| A | 1, 3, 4, 5 |
| C | 1, 2, 3, 4, 5, 6 |
| D | 2, 4, 5, 6 |
| T | 1, 3, 5, 6 |
| W | 1, 2, 3, 4, 5 |

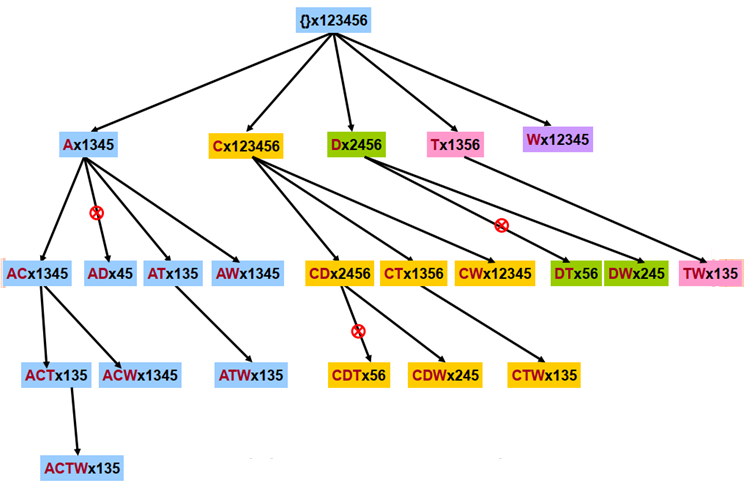
Ta chuyển đổi bảng CSDL mẫu bảng 2.3 sang bảng định dạng dữ liệu dọc bảng 2.4, ứng với mỗi mục là tập các giao dịch chứa nó hay Tidset, ví dụ: mục A có tập Tidset là {1,3,4,5}.

Thực hiện lần lượt giao các Tidset của các mục còn lại với Tidset của A để tạo ra Tidset của các tập mục 2 itemset, ví dụ: t(A) = 1345; t(AD) = t(A) ∩ t(D) = 1345 ∩ 2456 = 45.

Giả sử ngưỡng minSup = 50% tức minSup = 3, tập mục {A,D} sẽ không phổ biến vì |t(AD| = 2 < minSup.

Thực hiện đệ quy tiếp tục đối với các tập mục phổ biến vừa được tạo ra, ta có cây IT-tree như hình 2.5, ta thấy có 19 tập phổ biến thỏa minSup = 50%.

Sau khi đã khai thác tất cả các tập mục phổ biến, ta tiến hành khai thác các luật kết hợp như mục 2.1.1.2.



Hình .: Cây IT-tree được tạo ra từ CSDL mẫu[23]

*2.1.3.3. Đánh giá thuật toán:*

Thuật toán dựa vào phần giao giữa các Tidset để tính nhanh độ phổ biến nên chỉ quét CSDL 1 lần.

Do thuật toán không sinh ứng viên nên hiệu quả khai thác thường cao hơn so với các họ thuật toán sinh ứng viên.

Khi số tập phổ biến lớn, thời gian khai thác luật lớn ⇒Cần phương pháp khai thác hiệu quả hơn.

## 2.2. Một số kỹ thuật phân lớp trong khai thác dữ liệu

### 2.2.1. Giới thiệu về phân lớp

Phân lớp dữ liệu là dạng phân tích dữ liệu nhằm rút trích các mô hình mô tả  
các lớp dữ liệu hoặc dự đoán xu hướng dữ liệu.[22]

Quá trình gồm hai bước:

- Bước học (giai đoạn huấn luyện): ta xây dựng một mô hình phân lớp mô tả một bộ xác định trước các lớp dữ liệu hay khái niệm. Ở đó một thuật toán phân lớp xây dựng nên mô hình phân lớp bằng cách phân tích hoặc “học hỏi từ” một *tập huấn luyện* hình thành từ các bản ghi trong CSDL và các nhãn lớp liên kết với chúng. Mỗi bản ghi ***X*** được biểu diễn bởi một *vector thuộc* *tính* gồm *n*-chiều,***X***= (*x1*, *x2*, …, *xn*), môtả *n* giá trị đo đã thực hiện trên bản ghi đó từ *n* thuộctính tương ứng của CSDL *A1, A2, …,* *An*. Mỗi bản ghi ***X*** được giả định là thuộc về một lớp được quy định trước, như được xác định bởi một thuộc tính khác của CSDL gọi là *thuộc tính nhãn lớp*.

Thuộc tính nhãn lớp này có trị rời rạc và không có thứ tự. Nó *có tính phân loại* ở chỗ mỗi giá trị của nó đóng vai trò một *loại* hoặc *lớp*. Các bản ghi riêng tạo thành tập huấn luyện được gọi là các *bản ghi huấn luyện* và được lấy mẫu ngẫu nhiên từ CSDL đang phân tích. Trong ngữ cảnh phân lớp, các bản ghi của CSDL có thể được các tài liệu khác nhau gọi là *mẫu*, *ví dụ*, *thể hiện*, *điểm dữ liệu*, hoặc *đối tượng*.

Do nhãn lớp của mỗi bản ghi huấn luyện đều được cung cấp sẵn, nên bước này còn được gọi là *học có giám sát*, tức là việc học của mô hình phân lớp là “có giám sát” ở chỗ nó được cho biết là mỗi bản ghi huấn luyện thuộc về lớp nào. Nó trái với *học không giám sát*, trong đó ta không biết nhãn lớp của mỗi bản ghi huấn luyện vàcó thể cả số lượng hoặc tập hợp các lớp cần học nữa.

- Bước phân lớp (classification): mô hình tìm được ở bước học sẽ được dùng cho việc phân lớp những dữ liệu mới. Trước hết, ta đánh giá *độ chính xác dự đoán* (predictive accuracy) của mô hình phân lớp ấy. Nếu như ta dùng tập huấn luyện để đánh giá độ chính xác của mô hình phân lớp thì việc đánh giá này nhiều khả năng là quá lạc quan, bởi vì mô hình ấy có khuynh hướng *quá khớp* dữ liệu (tức là trong quá trình học, nó có thể sáp nhập luôn một số trường hợp bất thường của dữ liệu huấn luyện, vốn không hiện diện trong tổng thể tập dữ liệu nói chung). Vì thế, ta dùng một *tập kiểm tra*, bao gồm các *bản ghi kiểm tra* và các nhãn lớp liên kết với chúng, đểthực hiện việc đánh giá này. Các bản ghi kiểm tra này độc lập với các bản ghi huấn luyện, nghĩa là chúng không được dùng để xây dựng mô hình phân lớp.

*Độ chính xác* của một mô hình phân lớp trên một tập kiểm tra xác định là tỷ lệcác bản ghi của tập kiểm tra được phân loại đúng đắn bởi mô hình phân lớp ấy. Nhãn lớp liên kết của mỗi bản ghi kiểm tra được so sánh với sự dự đoán lớp của mô hình phân lớp học ứng với bản ghi đó. Nếu độ chính xác của mô hình phân lớp được xem là chấp nhận được, thì mô hình phân lớp có thể được dùng để phân lớp những bản ghi dữ liệu trong tương lai mà nhãn lớp chưa biết.

Chúng ta có các giải thuật phân lớp dữ liệu phổ biến như:

- Phân lớp dữ liệu với cây quyết định (decision tree).

- Phân lớp dữ liệu với mạng Naive Bayes.

- Phân lớp dữ liệu với mạng neural.

- Phân lớp dữ liệu với k phần tử gần nhất (k-nearest neighbor).

- Phân lớp dữ liệu dựa trên thuật toán di truyền (genetic algorithms).

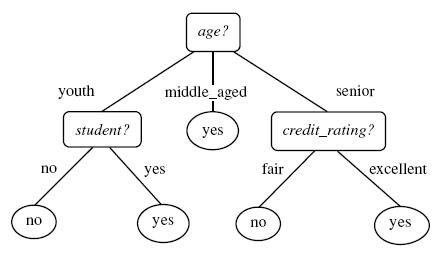
- Phân lớp dữ liệu với lý thuyết tập thô (rough sets).

- Phân lớp dữ liệu với lý thuyết tập mờ (fuzzy sets).

### 2.2.2. Phân lớp dữ liệu với cây quyết định (Decision tree)

*Phương pháp cây quyết định* là việc tìm kiếm các cây quyếtđịnh từ những bản ghi huấn luyện đã có nhãn lớp. Mỗi cây quyết định (decision tree) là một cấu trúc cây kiểu lưu đồ, trong đó mỗi *nút trong* biểu thị một sự kiểm tra trên một thuộc tính nào đó, mỗi *nhánh* biểu diễn một kết quả của sự kiểm tra đó, còn mỗi *nút lá* chứa một nhãn lớp. Nút ở trên cùng của cây là *nút gốc*, chứa tên của thuộc tínhcần kiểm tra. Hình 2.6 cho thấy một cây quyết định tiêu biểu. Các nút trong được biểu diễn bằng các hình chữ nhật, còn các nút lá được biểu diễn bằng các hình bầu dục. Một số thuật toán cây quyết định chỉ sinh ra các *cây nhị* *phân* (trong đó mỗi nút trong rẽ nhánh đến hai nút khác), trong khi những thuật toán cây quyết định khác có thể sinh ra những cây không nhị phân.

Sử dụng các cây quyết định cho phân lớp: cho một bản ghi ***X*** nào đó mà ta chưa biết nhãn lớp liên kết với nó, các giá trị thuộc tính của bản ghi đó được kiểm tra so với cây quyết định. Dựa theo những cuộc kiểm tra đó, ta lần ra một đường đi từ gốc đến một nút lá chứa kết quả dự đoán lớp dành cho bản ghi ấy. Các cây quyết định có thể dễ dàng được chuyển đổi thành các *luật phân lớp*, tức hình thức phân loại đơn giản và dễ hiểu nhất đối với con người.

  
30





Hình .: Một ví dụ về cây quyết định [22]

Giới thiệu một số độ đo:

- Information Gain (được dùng trong ID3):

 (2.1)

Trong đó: *Info(D)*: Lượng thông tin cần để phân loại một phần tử D.

pi: xác suất để một phần tử bất kỳ trong D thuộc về lớp Ci , với i = 1,..,m.

- Gain Ratio (được dùng trong C4.5):

 (2.2)

 (2.3)

### 2.2.3. Phân lớp dữ liệu bằng giải thuật học quy nạp ILA (Inductive Learning Algorithm)

Thuật giải ILA được dùng để xác định các luật phân lớp cho tập hợp các mẫu học. Thuật giải này thực hiện theo cơ chế lặp, để tìm luật riêng đại diện cho tập mẫu của từng lớp. Sau khi xác định được luật, thuật giải sẽ loại bỏ các mẫu mà luật này bao hàm, đồng thời thêm luật mới này vào tập luật. Kết quả có được là một danh sách có thứ tự các luật.[22]

Mô tả thuật giải ILA:

- Bước 1: Chia bảng con có chứa m mẫu thành n bảng con. Một bảng con ứng với một giá trị của thuộc tính phân lớp (Lặp lại từ bước 2 đến bước 8 cho mỗi bảng con).

- Bước 2: Khởi tạo số lượng thuộc tính kết hợp j với j = 1.

- Bước 3: Với mỗi bảng con đang xét, phân chia các thuộc tính của nó thành một danh sách các thuộc tính kết hợp, mỗi thành phần của danh sách có j thuộc tính phân biệt.

- Bước 4: Với mỗi kết hợp các thuộc tính trong danh sách trên, đếm số lần xuất hiện các giá trị cho các thuộc tính trong kết hợp đó ở các dòng chưa bị khóa của bảng đang xét nhưng nó không được xuất hiện cùng giá trị ở những bảng con khác. Chọn ra một kết hợp trong danh sách sao cho nó có giá trị tương ứng xuất hiện nhiều nhất và được gọi là Max\_combination.

- Bước 5: Nếu Max\_combination = 0 thì j = j+1 quay lại bước 3.

- Bước 6: Khóa các dòng ở bảng con đang xét mà tại đó giá trị bằng với giá trị tạo ra Max\_combination.

- Bước 7: Thêm vào R luật mới với giả thuyết là các giá trị tạo ra Max\_combination kết nối các bộ này bằng phép AND, kết luận là giá trị của thuộc tính quyết định trong bảng con đang xét.

- Bước 8: Nếu tất cả các dòng đều khóa:

+ Nếu còn bảng con thì qua bảng con tiếp theo và quay lại bước 2.

+ Ngược lại chấm dứt thuật toán.

Ngược lại quay lại bước 4.

### 2.2.4. Phân lớp dữ liệu với mạng Naive Bayes

Các mô hình phân lớp dựa theo Bayes (Bayesian classifier) là loại mô hình phân lớp theo lý thuyết thống kê. Chúng có thể dự đoán xác suất của các thành viên lớp, chẳng hạn xác suất để một bản ghi nhất định thuộc về một lớp cụ thể nào đó. Phân lớp dựa theo Bayes căn cứ vào nền tảng lý thuyết là định lý Bayes (được đặt theo tên của Thomas Bayes, nhà toán học Anh vào thế kỷ 18).

Thuật toán phân lớp Naïve Bayes (NB) giả định rằng ảnh hưởng của một giá trị thuộc tính nào đó trên một lớp nhất định là độc lập với các giá trị của các thuộc tính khác. Giả định này được gọi là *sự độc lập theo điều kiện lớp* (class-conditional independence). Người ta giả định như vậy để đơn giản hóa khối lượng tính toán cần thiết, và vì lý do này, nó được gọi là dạng “ngây thơ” (naïve).

Chi tiết của việc phân lớp dữ liệu bằng mạng NB có thể được tham khảo ở [22].

* Ưu điểm:

- Về thời gian học (tức thời gian xây dựng mô hình): ít hơn so với phương pháp quy nạp cây quyết định, và ít hơn rất nhiều so với mạng neural, nhất là đối với dữ liệu rời rạc.

- Hiệu năng phân lớp (độ chính xác và tốc độ) cao khi dùng với CSDL lớn.

- Thuật toán dễ hiểu và dễ hiện thực.

* Nhược điểm:

- Do NB giả định là các thuộc tính độc lập với nhau, nên khi các thuộc tính có sự phụ thuộc lẫn nhau (ví dụ, trong giáo dục có một số môn học có ý nghĩa tiên quyết đối với một số môn học khác) thì phương pháp NB trở nên thiếu chính xác.

- NB không sinh ra được những mô hình phân lớp dễ hiểu đối với người dùng không chuyên về khai thác dữ liệu.

### 2.2.5. Phân lớp dữ liệu với mạng neural

Lĩnh vực học bằng các *mạng neural nhân tạo* (Artificial Neural Network – ANN), ban đầu được khởi xướng bởi các nhà tâm lý học và các nhà sinh học thần kinh muốn tìm cách xây dựng và kiểm tra những mô hình tính toán tương tự với mạng lưới các tế bào thần kinh (neuron) của con người. Một mạng neural nhân tạo, đôi khi còn được gọi là *multilayer perceptron* (MLP), là một tập hợp các *nút* xuất/nhập nối kết với nhau, trong đó mỗi đường nối kết có một trọng số liên kết với nó. Trong giai đoạn học, mạng này học bằng cách điều chỉnh các trọng số để dự đoán được nhãn lớp đúng đắn của các bản ghi nhập vào.

Chi tiết của việc phân lớp dữ liệu bằng các NN có thể được tham khảo ở [22].

* Ưu điểm:

- Các mô hình học được từ mạng neural nhân tạo có khả năng chịu đựng đối với dữ liệu nhiễu cao cũng như khả năng phân lớp được những mẫu hình mà chúng chưa từng được huấn luyện.

- Chúng rất thích hợp đối với dữ liệu nhập và xuất có trị liên tục.

- Các thuật toán mạng neural vốn có sẵn tính song song; có thể dùng các kỹ thuật song song hóa để tăng tốc quá trình tính toán.

- Ngoài ra, gần đây đã có nhiều kỹ thuật được xây dựng để rút trích ra các luật phân lớp dễ hiểu từ các mạng neural học được.

* Nhược điểm:

- Học bằng mạng neural đòi hỏi thời gian huấn luyện phải dài, vì thế thích hợp hơn với các ứng dụng nào chấp nhận được điều này.

- Thuật toán xây dựng mạng neural nhân tạo cần một số tham số mà thường thì chỉ được xác định tốt nhất thông qua thí nghiệm, như cấu trúc.

- Các mô hình học bằng mạng neural đã bị chỉ trích vì tính khó hiểu của chúng; con người khó diễn giải được ý nghĩa biểu tượng đằng sau các trọng số học và ý nghĩa của các “đơn vị ẩn” trong mạng.

## 2.3. Một số phương pháp khai thác luật phân lớp kết hợp

Quá trình phân lớp dựa trên luật thường rất hiệu quả và mang tính giải thích cao. Một số phương pháp khai thác tập luật đã được đề xuất như phương pháp học trên cây quyết định (Quinlan 1986, 1992)[11], thuật toán học quy nạp ILA (Tolun và Abu-Soud, 1998; Tolun và đồng sự, 1999)[14], thuật toán phân lớp kết hợp (AC) (Liu và đồng sự, 1998)[1]. Một số nghiên cứu cho thấy thuật toán phân lớp dựa trên luật kết hợp cho ra kết quả chính xác hơn so với các phương pháp phân lớp truyền thống khác (Veloso và đồng sự, 2006, 2007, 2011)[5][6][7]. Do đó, trong thời gian gần đây có rất nhiều thuật toán tập trung vào vấn đề dự đoán lớp trên cơ sở luật kết hợp. Khai thác luật phân lớp kết hợp (Mining CARs) là khai thác ra được tất cả các luật liên quan đến lớp của dữ liệu thỏa mãn ngưỡng hỗ trợ tối thiểu (*minSup*) và ngưỡng độ tin cậy tối thiểu (*minConf*).

### 2.3.1. Thuật toán CBA (Classification Based on Associations)

Trong giải thuật này, ta xử lý tất cả các thuộc tính một cách thống nhất. Với mỗi một thuộc tính rời rạc, tất cả các giá trị hiện hữu được ánh xạ tới tập hợp các số nguyên dương liên tiếp. Với mỗi một thuộc tính liên tục, phạm vi giá trị của nó sẽ được rời rạc hóa thành các khoảng và các khoảng này cũng được ánh xạ thành các số nguyên dương liên tiếp. Với các ánh xạ này, chúng ta có thể coi tập dữ liệu là tập hợp các cặp (*thuộc tính, giá trị thuộc tính*) và nhãn lớp. Chúng ta gọi mỗi cặp (*thuộc tính, giá trị thuộc tính*) là một tập mục *item*.

Ta gọi *D* là tập dữ liệu ban đầu,  *I* là tập tất cả các mục có trong dữ liệu *D* và *Y* là tập các nhãn lớp. Ta nói tác vụ *d*∈*D* chứa tập mục *X**I* nếu *X**d.*

Một luật phân lớp (CAR) là luật có dạng *X*→*y*, trong đó *X**I* và *y*∈*Y.* Luật *X*→*y* tồn tại trong dữ liệu *D* với độ tin cậy *c* nếu có *c%* số tác vụ có chứa itemset *X* trong *D* có nhãn lớp là *y.* Luật *X*→*y* có độ hỗ trợ *s* nếu có *s%* số tác vụ trong *D* chứa itemset *X* và được gán nhãn lớp *y.*

Mục tiêu của chúng ta là sinh ra tập các luật phân lớp kết hợp CARs thỏa mãn ngưỡng hỗ trợ tối thiểu *minSup* và độ tin cậy tối thiểu *minConf,* sau đó sẽ xây dựng bộ phân lớp từ tập CARs đó. Trong luận văn này, tôi chỉ trình bày về bước khai thác luật phân lớp kết hợp, bước xây dựng bộ phân lớp có thể tham khảo chi tiết ở [1].

*2.3.1.1. Thuật toán CBA-RG (CBA Rule Generator):*

Đây là thuật toán dựa trên nền tảng của thuật toán khai thác dữ liệu nổi tiếng Apriori (thuật toán khai thác luật kết hợp của Agrawal và Srikant 1994).

* Một số khái niệm cơ bản của thuật toán:

- Hoạt động chính của thuật toán CBA-RG là tìm tất cả các luật thỏa mãn ngưỡng minSup. Luật có dạng: <*condset,y*>, trong đó *condset* là tập các mục item, *y*∈*Y* là nhãn lớp. Độ hỗ trợ của *condset* (*condsupCount*) là số lượng tác vụ trong *D* chứa *condset.* Độ hỗ trợ của luật (*rulesupCount*) là số lượng tác vụ trong D chứa condset và được gán nhãn lớp y.

- Trong mỗi luật *condset*→*y:*

+ Độ hỗ trợ *support* = , trong đó |*D*| là kích thước của tập dữ liệu.

+ Độ tin cậy *confidence* = 

- Những luật thỏa mãn ngưỡng minSup được gọi là luật phổ biến, những luật còn lại gọi là không phổ biến. Xét ví dụ cụ thể, ta có luật:

<{(A, 1), (B, 1)}, (class: 1)>

Trong đó A và B là các thuộc tính. Giả sử độ hỗ trợ của condset {(A, 1), (B, 1)} là 3, độ hỗ trợ của luật là 2 và tổng số tác vụ trong *D* là 10, như vậy độ hỗ trợ *support* của luật là 20% và độ tin cậy *confidence* của luật là 66.7% theo... Nếu ngưỡng minSup là 10%, ta nói luật này phổ biến.

Đối với tất cả các luật có cùng *condset*, luật nào có độ tin cậy *confidence* cao hơn sẽ được chọn gọi là luật khả thi đại diện cho tập luật này. Nếu có nhiều hơn một luật có cùng độ tin cậy cao nhất, chúng ta sẽ lựa chọn ngẫu nhiên lấy ra một luật. Ví dụ, chúng ta có hai luật có cùng *condset* như sau:

i) <{(A, 1), (B, 1)}, (class: 1)>

ii) <{(A, 1), (B, 1)}, (class: 2)>

Giả sử độ hỗ trợ của condset là 3. Độ hỗ trợ support của luật 1 là 2, của luật 2 là 1, hay nói cách khác của luật 1 là 66.7%, của luật 2 là 33.3%. Với 2 luật này, chúng ta chỉ chọn một luật khả thi (giả sử tập dữ liệu có 10 tác vụ):

(A, 1), (B, 1)→(class, 1) [*supp* = 20%, *conf* = 66.7%]

Nếu độ tin cậy lớn hơn *minConf*, ta nói luật này có tính chính xác. Tập các luật phân lớp kết hợp (CARs) vì thế bao gồm tất cả các luật khả thi phổ biến và chính xác.

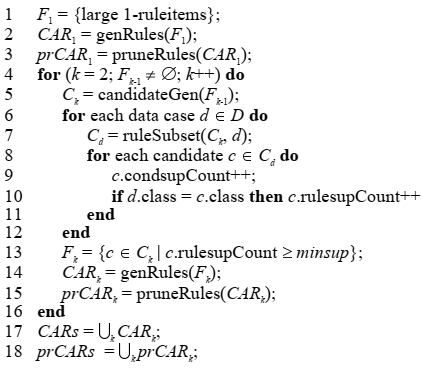
* Thuật toán CBA-RG:

Thuật toán CBA-RG sinh ra tất cả các luật phổ biến bằng cách quét nhiều lần trên dữ liệu ban đầu. Trong lần lặp đầu tiên, nó tính toán độ hỗ trợ của các luật chứa item đơn và chọn ra luật nào là phổ biến. Trong những lần lặp tiếp theo, nó sẽ bắt đầu với các luật phổ biến đã tìm được trong những lần lặp trước đó, sử dụng các luật này để sinh ra các ứng viên mới. Độ hỗ trợ thực tế cho các luật ứng viên này được tính toán trong suốt quá trình lặp trên dữ liệu. Kết thúc mỗi vòng lặp, nó sẽ quyết định ứng viên nào thực sự phổ biến. Từ những luật phổ biến này sẽ sinh ra các luật phân lớp kết hợp (CARs).

Cho *k-ruleitem* ký hiệu cho 1 luật mà *condset* của nó có *k* items, *Fk* ký hiệu cho tập *k-ruleitems* phổ biến. Mỗi luật của tập này có dạng:

<*(condset, condsupCount), (y, rulesupCount)*>

Gọi Ck là tập ứng viên k-ruleitems, chi tiết thuật toán được thể hiện ở hình 2.7.



Hình .: Thuật toán CBA-RG [1]

*2.3.1.2. Đánh giá thuật toán:*

- Thuật toán có sự khác biệt rõ ràng với các thuật toán phân lớp trước đó như C4.5 chỉ sinh ra một số lượng nhỏ các luật thiên lệch, thuật toán CBA sinh ra tập luật phân lớp hoàn thiện hơn.

- Bộ phân lớp được xây dựng trên tập luật mang tính toàn cục vì được sinh ra trên toàn bộ dữ liệu huấn luyện sử dụng trong khi những phương pháp trước đã loại bỏ bớt một số tác vụ được phủ bởi luật tốt nhất được tìm thấy.

- Hạn chế của cách tiếp cận này là thuật toán sinh ra quá nhiều ứng viên và phải quét cơ sở dữ liệu (CSDL) nhiều lần dẫn đến việc tiêu thụ thời gian và bộ nhớ khá lớn.

### 2.3.2. Thuật toán CMAR (Classification Based on Multiple Rules)

*2.3.2.1. Các khái niệm cơ bản:*

Thuật toán CBA trước đây vẫn tồn tại một số nhược điểm nhất định như:

- Không dễ dàng để xác định luật nào hiệu quả nhất trong việc phân loại cho các tác vụ mới.

- Tập dữ liệu huấn luyện thường sinh ra một tập luật khá lớn. Nó gây ra thử thách không nhỏ cho việc lưu trữ, truy hồi, cắt tỉa và sắp xếp một lượng lớn luật hiệu quả cho quá trình phân lớp.

Năm 2001, Li và đồng sự đã đề xuất giải thuật dựa trên cấu trúc cây FP-tree để giải quyết các nhược điểm trên. Thứ nhất, thay vì dựa trên các luật đơn cho việc phân lớp, CMAR xác định nhãn lớp bằng 1 tập luật. Thuật toán sẽ chọn một tập hợp nhỏ các luật có độ tương quan và tin cậy cao. Để tránh sự thiên lệch, nó phát triển một kỹ thuật được gọi là trọng số  biểu thị cho một thông số đo độ mạnh của luật dưới sự phân bố lớp và độ hỗ trợ điều kiện. Thứ hai, để cải thiện độ chính xác và hiệu suất, CMAR áp dụng một kiến trúc mới CR-Tree nhằm lưu trữ gọn nhẹ và truy hồi một lượng lớn luật cho việc phân lớp một cách hiệu quả nhất. Thứ ba, để tăng tốc cho việc khai thác các luật hoàn thiện, CMAR sử dụng biến thể của thuật toán FP-Growth trước đây.

Gọi *C* = *{c1,..., cm}* là tập các nhãn lớp. Một tập dữ liệu huấn luyện là một tập các đối tượng dữ liệu, trong đó mỗi đối tượng tồn tại một nhãn lớp *cobj*∈*C* tương ứng.

Mẫu *P* =  là tập các thuộc tính-giá trị trong đó (*1≤j≤k*), . Một đối tượng dữ liệu *obj* được gọi là hợp với mẫu *P* nếu và chỉ nếu với (*1≤j≤k*), *obj* có giá trị là của thuộc tính .

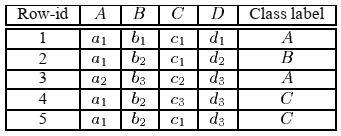
Cho tập dữ liệu huấn luyện *T*, gọi *c* là nhãn lớp. Với luật *R*: *P* → *c*, số đối tượng dữ liệu trong *T* hợp với mẫu *P* và có nhãn lớp *c* được gọi là độ hỗ trợ của luật *R*, ký hiệu *sup(R)*. Tỉ số giữa số đối tượng hợp với *P* và có nhãn lớp *c* với tổng số đối tượng hợp với mẫu *P* được gọi là độ tin cậy của luật *R*, kí hiệu *conf(R)*.

*2.3.2.2. Giai đoạn khai thác luật:*

CMAR đầu tiên khai thác trên tập dữ liệu huấn luyện để tìm ra tập luật hoàn thiện thỏa ngưỡng hỗ trợ và độ tin cậy cho trước. Để thuật toán khai thác có khả năng mở rộng và hiệu quả cao, CMAR sử dụng biến thể của phương pháp FP-Growth. FP-Growth là thuật toán khai thác mẫu phổ biến nhanh hơn thuật toán Apriori truyền thống, đặc biệt là trong những tập dữ liệu lớn có ngưỡng hỗ trợ thấp hay mẫu khai thác dài. Chúng ta có thể theo dõi ý tưởng của thuật toán CMAR qua ví dụ cụ thể sau:

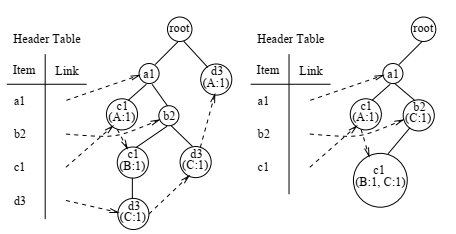
- Xét ví dụ: cho bảng dữ liệu huấn luyện *T* như bảng 2.5. giả sử ngưỡng hỗ trợ là 2 và ngưỡng độ tin cậy là 50%.

Bảng .: Dữ liệu huấn luyện T[3]



- Đầu tiên, thuật toán CMAR quét dữ liệu *T* một lần để tìm ra tập các giá trị thuộc tính xuất hiện ít nhất 2 lần trong *T*. Tập giá trị tìm được là *F* = {*a1, b2, c1, d3*} được gọi là tập mục phổ biến, tất cả những giá trị còn lại không thỏa ngưỡng hỗ trợ sẽ bị loại.

- Sau đó, CMAR sắp xếp giá trị thuộc tính trong *F* theo thứ tự độ hỗ trợ giảm dần, chẳng hạn *F-list* = *a1 - b2 - c1 - d3.* CMAR sẽ quét dữ liệu lần thứ hai để xây dựng cây FP-tree từ *F-list* như hình 2.8a.



(a) (b)

Hình .: Cây FP-tree[3]

- FP-tree là cây tiền tố, với mỗi giao tác trong tập dữ liệu huấn luyện, những giá trị thuộc tính xuất hiện trong *F-list* được rút trích ra và sắp xếp theo *F-list*. Ví dụ: với bộ đầu tiên (*a1, c1*) được rút ra và thêm vào nhánh bên trái ngoài cùng của cây, nhãn lớp được đính kèm theo nút cuối cùng trên đường đi.

- Bộ dữ liệu trong tập dữ liệu huấn luyện chia sẻ tiền tố. Ví dụ: bộ thứ hai có giá trị thuộc tính (*a1, b2, c1)* trong F-list và chia sẻ tiền tố a1 với bộ dữ liệu thứ nhất. Vì thế, nó cũng sẽ chia sẻ đường dẫn phụ *a1* với nhánh bên trái ngoài cùng. Tất cả các nút với cùng giá trị thuộc tính sẽ được nối với nhau như một hàng bắt đầu từ *header table*.

- Tiếp theo, dựa vào *F-list*, tập luật phân lớp kết hợp có thể được chia thành 4 tập con không trùng lắp: (1) tập chứa *d3*; (2) tập chứa *c1* nhưng không có *d3*­; (3) tập chứa *b2* nhưng không chứa *d3* và *c1*; và (4) những tập chỉ chứa *a1*. CMAR sẽ lần lượt tìm những tập con này.

- Để tìm ra các tập con luật chứa *d3,* thuật toán xem xét các nút có *d3* và đi ngược từ dưới lên lấy dữ liệu chiếu trên nút *d3* này, ta sẽ có 3 bộ dữ liệu sau: *(a1, b2, c1, d3)* : *C*, *(a1, b2, d3)*: *C*, *d3*: *A*. Một cách đệ quy, khi chiếu lên *d3, a1* và *b2* là những giá trị thuộc tính phổ biến vì thỏa ngưỡng hỗ trợ. Dựa trên thông tin về sự phân phối nhãn lớp, chúng ta có luật *a1b2d3* → *C* với độ hỗ trợ là 2 và độ tin cậy là 100%.

- Sau khi tìm kiếm luật chứa d3, tất cả các nút của *d3* sẽ được nhập lên nút cha tương ứng, tức là thông tin nhãn lớp gắn với nút *d3* sẽ có trong nút cha của nó. Cây FP-tree được rút lại như hình 2.8b. Các tập con còn lại được khai thác tương tự.

Có 2 điểm khác nhau trong thuật toán CMAR và FP-Growth:

- Thứ nhất, CMAR tìm tất cả các mẫu phổ biến và sinh luật trong một bước, khác với khai thác luật kết hợp truyền thống phải thực hiện qua hai bước là khai thác tập phổ biến thỏa ngưỡng hỗ trợ và sinh luật thỏa mãn độ tin cậy tối thiểu.

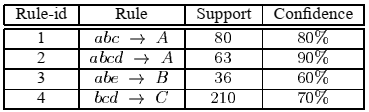
- Thứ hai, CMAR sử dụng phân bố nhãn lớp để cắt tỉa. Với bất kì mẫu *P* nào, gọi *c* là lớp trội nhất trong tập các đối tượng *obj* hợp với *P*. Nếu số lượng đối tượng có nhãn lớp *c* và hợp *P* nhỏ hơn ngưỡng hỗ trợ thì sẽ không cần tiếp tục khai thác bất kì tập cha *P’* nào của *P* nữa vì bất kì luật *P’* → *c* nào cũng sẽ không thỏa mãn ngưỡng hỗ trợ cho trước.

*2.3.2.3. Giai đoạn lưu trữ luật trong cây CR-tree:*

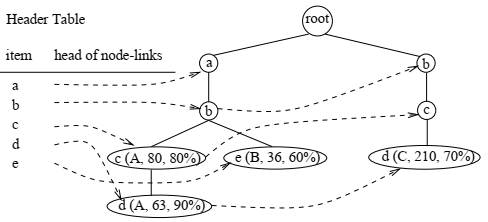
Khi một luật được sinh ra, nó được lưu trữ trong cây CR-tree, đó là cấu trúc cây tiền tố. Chúng ta có thể xem qua ví dụ sau:

- Ví dụ: sau khi khai thác tập dữ liệu huấn luyện, 4 luật được tìm thấy trong bảng 2.6.

Bảng .: Các luật được khai thác[3]



- Cây CR-tree được xây dựng như hình 2.9:



Hình .: Cấu trúc cây cho tập luật khai thác[3]

- Cây CR-tree có 1 nút gốc. Tất cả các giá trị thuộc tính xuất hiện phía bên trái và được sắp xếp theo tần suất xuất hiện.

- Với luật thứ nhất, *abc* → *A* được thêm vào cây như là một đường đi từ nút gốc. Nhãn lớp cũng như là độ hỗ trợ, độ tin cậy của luật kí hiệu là (A, 80, 80%) được gắn tại nút cuối trên đường đi tức nút *c*. Các luật khác cũng thêm vào cây tương tự. Tất cả các nút có cùng giá trị thuộc tính được nối với nhau thành hàng, đầu của mỗi hàng được lưu trong bảng *Header table*.

Cấu trúc cây CR-tree có những ưu điểm sau:

- Là một cấu trúc nhỏ gọn. Nó có thể khai thác được tất cả các luật và vì thế có thể tiết kiệm vùng nhớ để lưu trữ.

- Tự cấu trúc cây là một chỉ mục cho các luật, khi muốn truy hồi luật chứa giá trị thuộc tính nào, ta chỉ cần xét các kết nối nút chứa giá trị đó từ *header table* và tiếp tục xét từ dưới lên trên.

- Cắt tỉa luật và sử dụng luật cho phân lớp rất nhanh chóng.

*2.3.2.4. Giai đoạn cắt tỉa luật:*

Số lượng luật sinh ra có thể quá nhiều, chúng ta cần cắt tỉa bớt luật để giảm thông tin thừa và nhiễu.

Cho 2 luật *R1* và *R2*, *R1* có thứ hạng cao hơn *R2* kí hiệu là *R1* > *R2* nếu và chỉ nếu (1) *conf(R1)* > *conf(R2)*; (2) *conf(R1)* = *conf(R2)* nhưng *sup(R1)* > *sup(R2)*; hay (3) *conf(R1)* = *conf(R2)*, *sup(R1)* = *sup(R2)* nhưng *R1* có ít giá trị thuộc tính trong phần trái của nó hơn *R2*. Thêm nữa, *R1*: *P* → *c* được gọi là luật chung của *R2*: *P’* → *c’* nếu và chỉ nếu *P* là tập con của *P’*.

CMAR áp dụng những phương pháp sau để cắt tỉa luật:

- Đầu tiên, sử dụng các luật chung và có độ tin cậy cao để cắt tỉa những luật cụ thể hơn và có độ tin cậy thấp.

- Thứ hai, chỉ chọn những luật tương quan tích cực. Chỉ những luật với giá trị  thỏa mức ngưỡng cụ thể được sử dụng cho việc phân lớp sau này, các luật còn lại không thỏa sẽ bị loại bỏ. Lý do cho việc cắt tỉa này là chúng ta sử dụng những luật phản ánh mạnh để phân lớp và loại bớt nhiễu.

- Thứ ba, cắt tỉa luật dựa trên việc phủ dữ liệu, CMAR chọn ra một tập con luật chất lượng cao bằng cách dùng một ngưỡng phủ dữ liệu. Nó khác phương pháp CBA ở chỗ thay vì loại bỏ một đối tượng dữ liệu từ tập dữ liệu huấn luyện ngay khi được phủ bởi các luật lựa chọn, chúng ta vẫn để nó ở đó đến khi được phủ bởi tối thiểu *δ* luật, điều đó sẽ cho phép nhiều luật được lựa chọn hơn dẫn đến việc dự đoán sẽ chính xác hơn sau này.

Giai đoạn phân lớp có thể tham khảo chi tiết ở [3].

*2.3.2.5. Đánh giá thuật toán:*

- Thuật toán CMAR phân lớp dựa trên phân tích trọng số  thực thi trên đa luật kết hợp dẫn đến độ chính xác phân lớp tổng thể tốt hơn.

- Nó cắt tỉa luật một cách hiệu quả dựa trên độ tin cậy, độ phủ dữ liệu và tính tương quan.

- Hiệu suất của nó đạt được bằng cách mở rộng phương pháp khai thác mẫu phổ biến FP-Growth, xây dựng cấu trúc cây phân bố lớp kết hợp FP-tree và áp dụng vào cấu trúc cây CR-tree để lưu trữ và truy hồi luật một cách hiệu quả nhất.

- Tuy nhiên, khi khai thác luật bằng phương pháp này, chúng ta phải xác định nhãn lớp tương ứng với mỗi mẫu trong nhánh của cây. Việc này làm cho cây trở nên phức tạp hơn và việc khai thác tiêu thụ nhiều thời gian hơn cho việc dùng phép chiếu trên cây với các dữ liệu huấn luyện lớn.

### 2.3.3. Thuật toán CAR-Miner

Để giải quyết hạn chế của thuật toán ECR-CARM dựa trên cấu trúc ECR-tree, thuật toán CAR-Miner ra đời. Thuật toán ECR-CARM tiêu tốn nhiều thời gian cho việc kiểm tra ứng viên vì mỗi nút trong cây chứa tất cả các giá trị của một thuộc tính. CAR-Miner là một thuật toán cải tiến của thuật toán ECR-CARM đã được phát triển bởi Nguyễn và các đồng sự vào năm 2013[9]. CAR-Miner khai thác luật phân lớp kết hợp dựa trên cấu trúc cây MECR-tree. Trong cấu trúc cây MECR-tree, mỗi nút trong cây đại diện cho các tập itemset.

*2.3.3.1. Một số khái niệm cơ bản:*

Khai thác luật phân lớp dựa vào khai thác luật kết hợp (**C**lass **A**ssociaton **R**ule**s** - CARs) là tìm một tập con của các luật kết hợp có trong cơ sở dữ liệu. Mỗi luật trong tập con này chứa vế phải là giá trị của thuộc tính lớp.

Mục tiêu của khai thác luật phân lớp dựa vào khai thác luật kết hợp là:

- Khai thác tập CARs thỏa ngưỡng độ hỗ trợ tối thiểu (*MinSup*) và ngưỡng độ tin cậy tối thiểu (*MinConf*).

- Xây dựng bộ phân lớp từ CARs.

Một cách hình thức, bài toán khai thác luật phân lớp kết hợp CARS được phát biểu như sau:

Gọi *D* là tập dữ liệu huấn luyện với n thuộc tính *A1, A2,...,An* và *|D|* đối tượng (mẫu). Gọi *C = {c1, c2,..., ck}* là tập các nhãn lớp. Mỗi giá trị của thuộc tính *Ai* và thuộc tính lớp *C* được ký hiệu bởi các ký tự thường *a* và *c* tương ứng.

Các định nghĩa và khái niệm đã được phát biểu tại mục 1.3.1.

Cấu trúc cây MECR-tree (Modification of Equivalence Class Rule tree) là cấu trúc cây cải tiến từ cấu trúc ECR-tree, mỗi nút trong cây chỉ chứa một tập itemset với các thông tin sau:

- *Obidset*: tập các nhận dạng đối tượng tác vụ chứa itemset.

- (*c1, c2,..., ck*): danh sách các số nguyên, trong đó *ci* là số bản ghi trong *Obidset* thuộc về lớp *ci.*

- pos:: số nguyên dương lưu trữ vị trí của lớp với số đếm cao nhất, tức là .

- Ví dụ: xét itemset *X* = {*(A, a3), (B, b3)*} từ bảng 1.1, vì *X* được chứa trong tác vụ *4* và *6*, tất cả đều thuộc lớp *y* nên cấu trúc của nút sẽ là  hay đơn giản hơn là . Thông số *pos* là 1 (được gạch dưới tại vị trí 1 của nút này) vì số lượng lớp *y* là đa số (2 so với 0). Chúng ta sử dụng bit cho việc lưu trữ thuộc tính của tập itemset để tiết kiệm bộ nhớ. Ví dụ, *AB* có thể được đại diện là 11 ở dạng bit và vì thế giá trị của thuộc tính này là 3, với định dạng này chúng ta có thể sử dụng toán tử trên bit để hợp itemset nhanh hơn.

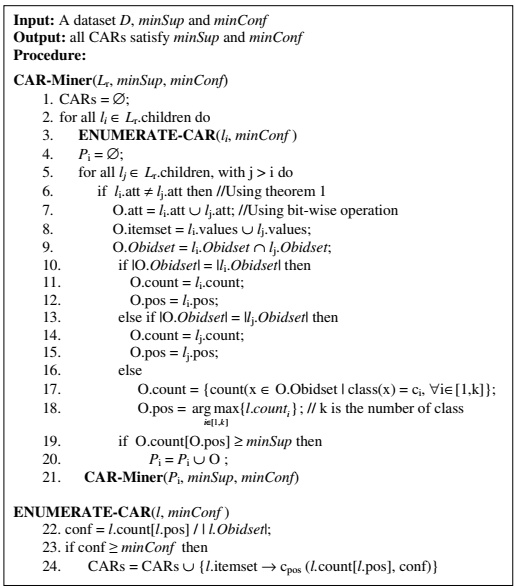
*2.3.3.2. Thuật toán khai thác luật:*

***Định lý 1:*** Cho 2 nút  và , nếu *att1*= *att2* và values1 ≠ values2 thì *Obidset1* ∩ *Obidset2* = ∅, tức là nếu 2 itemset *X* và *Y* có cùng thuộc tính thì chúng ta có thể bỏ qua bước kết hợp thành itemset mới *XY* vì *sup(XY)* = 0.

***Định lý 2:*** Cho 2 nút  và , nếu itemset1  itemset2 và  thì .

Bằng cách sử dụng 2 định lý trên, chúng ta có một thuật toán khai thác luật phân lớp kết hợp hiệu quả như hình 2.10. Với định lý 1, chúng ta không cần hợp 2 nút có cùng thuộc tính, và với định lý 2, chúng ta không cần tính toán thông tin cho một số nút con.

Đầu tiên, nút gốc *Lr* chứa danh sách các nút con chứa các item đơn phổ biến, sau đó hàm CAR-Miner sẽ được gọi với tham số *Lr* để khai thác tất cả các luật CARs từ dữ liệu *D.*



Hình .: Thuật toán CAR-Miner[9]

Xét ví dụ cụ thể từ bảng 1.1 để mô tả thuật toán với *minSup* = 10%, *minConf* = 60%, sử dụng nút  để minh họa quá trình kết với các *lj* sau đó trong danh sách *Lr*

- Với nút , vì li và lj có cùng thuộc tính và khác giá trị nên bỏ qua nút này không thực hiện gì thêm.

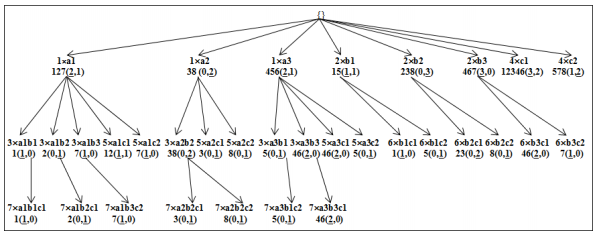
- Với nút , vì 2 thuộc tính khác nhau nên chúng ta sẽ tiếp tục thuật toán tính ra  hay 11 ở dạng bit; , và . Vì O.count[O.pos] = 0 < *minSup* nên O không được thêm vào tập *Pi*.

- Với nút , vì 2 thuộc tính khác nhau nên chúng ta sẽ tiếp tục thuật toán tính ra  hay 11 ở dạng bit; , và . Vì  nên thuật toán sao chép thông tin của nút *li* sang cho nút O. Có nghĩa là  và O.pos = 1; vì O.count[O.pos] = 2 > *minSup* nên O được thêm vào tập *Pi*  

- Thực hiện tương tự với các nút *lj* còn lại, ta sẽ có tập *Pi* = . Thuật toán sẽ tiếp tục được gọi đệ quy với thông số *Pi*, *minSup*, *minConf*.

- Luật sẽ được sinh ra bằng cách gọi hàm ENUMERATE-CAR(*li, minConf*). Ví dụ, với  thì conf = li.count[li.pos]/ |li.Obidset| = 2/2 = 1. Vì conf ≥ minConf (60%) nên ta thêm luật  vào tập luật CARs, nó có nghĩa là “Nếu A = a2 thì class = n” (độ hỗ trợ = 2, độ tin cậy = 100%).

- Kết quả xây dựng cây MECR-tree được thể hiện ở hình 2.11.



Hình .: Cây MECR-tree xây dựng từ bảng 1.1.[9]

*2.3.3.3.Đánh giá thuật toán:*

- Thuật toán CAR-Miner khai thác luật phân lớp kết hợp sử dụng cấu trúc cây giúp tiết kiệm bộ nhớ và thời gian khai thác luật, mỗi nút trên cây chứa thông tin cho việc tính toán nhanh độ hỗ trợ của luật ứng viên.

- Bằng cách sử dụng *Obidset* chúng ta có thể tính toán nhanh độ hỗ trợ của tập itemset.

- Ngoài ra, dựa trên các định lý, chúng ta có thể bỏ qua các bước tính toán một số thông tin cho rất nhiều nút trên cây.

- Hạn chế của thuật toán này là nó vẫn tiêu thụ khá nhiều bộ nhớ cho việc lưu trữ các *Obidset* của các tập itemset và đòi hỏi thời gian tính toán cho giai đoạn giao giữa các tập *Obidset* với nhau, thời gian này trở nên đáng kể khi chúng ta xét trong một cơ sở dữ liệu lớn.

### 2.3.4. Thuật toán CAR-Miner-Diff

Để giải quyết hạn chế của thuật toán CAR-Miner dựa trên cấu trúc cây MECR-tree, thuật toán CAR-Miner-Diff ra đời. Thuật toán CAR-Miner tiêu thụ khá nhiều bộ nhớ cho việc lưu trữ các *Obidset* của các tập itemset và đòi hỏi thời gian tính toán cho giai đoạn giao giữa các tập *Obidset* với nhau, thời gian này trở nên đáng kể khi chúng ta xét trong một cơ sở dữ liệu lớn. CAR-Miner-Diff là một thuật toán cải tiến của thuật toán CAR-Miner đã được phát triển bởi Nguyễn và các đồng sự vào năm 2015[10]. CAR-Miner-Diff thay vì lưu trữ phần giao giữa các tập Obidset, nó chỉ lưu trữ phần sai khác giữa các tập Obidset đó (gọi là *Diffset*), điều này dẫn đến bộ nhớ và tốc độ khai thác luật phân lớp kết hợp dựa trên cấu trúc cây được cải thiện.

*2.3.4.1. Một số khái niệm cơ bản:*

Các khái niệm cơ bản và định lý 1 cơ bản giống với thuật toán CAR-Miner đã được giới thiệu trong phần 2.3.3.1.

Trong thuật toán CAR-Miner, phần giao giữa 2 tập *Obidsets* được sử dụng để tính thông số *count* cho mỗi nút của cây; thuật toán CAR-Miner-Diff lại sử dụng chính phần khác biệt giữa 2 tập *Obidsets* để tính thông số *count*.

***Định nghĩa***: Cho 2 nút *X* và *Y* chứa *k*-itemsets có cùng tiền tố (*k-1*)-itemsets trong cây MECR-tree, trong đó *O1* là Obidset của *X* và *O2* là Obidset của *Y*. Giả sử rằng *Z* được sinh ra từ 2 nút trên. Gọi *d2O(Z)* là phần khác biệt giữa *O1* và *O2*, chúng ta có: *d2O(Z) = O1 \ O2.*

Ví dụ: Cho  ,  , như vậy *d2O(Z) = O1 \ O2* = {3, 8} \ {2, 3, 8} = ∅

***Định lý 2:*** Cho 2 nút *XY* và *XZ* (*X, Y và Z là (k-1)-itemsets*) chứa *k*-itemsets có cùng tiền tố *(k-1)*-itemset trong cây MECR-tree, chúng ta có *d2O(XYZ) = d2O(XZ) \ d2O(XY).*

Ví dụ: Cho  ,  , như vậy *d2O(3,a2b2) =* ∅. Tương tự chúng ta có *d2O(5,a2c1)* = {8}, *d2O(7,a2b2c1)* được tính như sau:

*d2O(7, a2b2c1) = d2O(5,a2c1) \ d2O(3,a2b2) = {8} \ ∅ = {8}*

Để tính toán thông số *count* của nút *XYZ*, chúng ta sử dụng *d2O(XYZ)* như sau:

*XYZ.count* = *XY.count* – {*#ci*­|*#ci* = số tác vụ trong *d2O(XYZ)* thuộc về lớp *ci*}

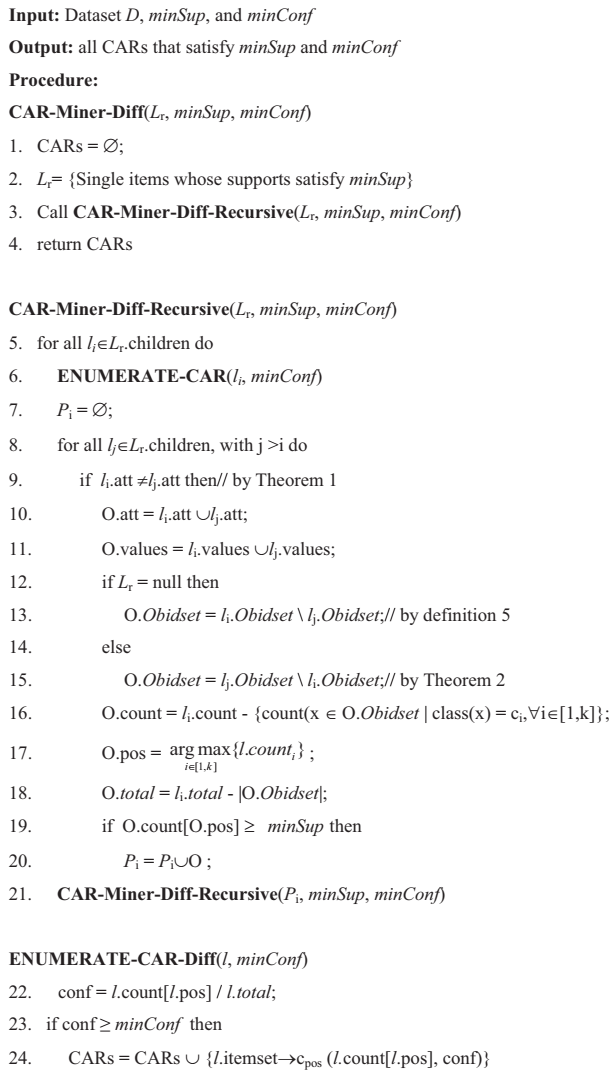
*2.3.4.2. Thuật toán khai thác luật:*

Bằng cách sử dụng định nghĩa, định lý và công thức trên, chúng ta có một thuật toán khai thác luật phân lớp kết hợp hiệu quả như hình 3.3.

Đầu tiên, nút gốc *Lr* chứa danh sách các nút con chứa các item đơn phổ biến, sau đó hàm CAR-Miner-Diff-Recursive sẽ được gọi với tham số *Lr*, *minSup* và *minConf* để khai thác tất cả các luật CARs từ dữ liệu *D.*

Hàm ENUMERATE-CAR-Diff(*l*, *minConf*) sinh luật từ nút *l*. Đầu tiên, nó sẽ tính ra độ tin cậy của luật, nếu thỏa mãn ngưỡng *minConf* thì thêm luật đó vào tập luật CARs. Lưu ý rằng, khi sử dụng *d2O,* |*l.Obidset|* chính là lực lượng của diffset *d2O* thay vì là Obidset. Biến total được sử dụng để lưu lực lượng của Obidset ban đầu, thông số này được tính thông qua *total* của nút cha *l­i*.

Xét ví dụ cụ thể từ bảng 1.1 để mô tả thuật toán với *minSup* = 10%, *minConf* = 60%, sử dụng nút  để minh họa quá trình kết với các *lj* sau đó trong danh sách *Lr*



Hình .: Thuật toán CAR-Miner-Diff[10]

- Với nút , vì li và lj có cùng thuộc tính và khác giá trị nên bỏ qua nút này không thực hiện gì thêm.

- Với nút , vì 2 thuộc tính khác nhau nên chúng ta sẽ tiếp tục thuật toán tính ra  (hay 11 ở dạng bit); , và => *O.count* = (0,2) – (0,2) = (0,0) và *O.pos* = 1. Vì *O.count* < *minSup* nên O không được thêm vào tập *Pi*.

- Với nút , vì 2 thuộc tính khác nhau nên chúng ta sẽ tiếp tục thuật toán tính ra  (hay 11 ở dạng bit); , và , O.count = (0,2) – (0,0) = (0,2) và O.pos = 2; vì O.count[O.pos] = 2 > *minSup* nên O được thêm vào tập *Pi*  

- Thực hiện tương tự với các nút *lj* còn lại, ta sẽ có tập *Pi* = . Thuật toán sẽ tiếp tục được gọi đệ quy với thông số *Pi*, *minSup*, *minConf*.

- Luật sẽ được sinh ra bằng cách gọi hàm ENUMERATE-CAR(*li, minConf*). Ví dụ, với  thì conf = li.count[li.pos]/ |li.total| = 2/2 = 1. Vì conf ≥ minConf (60%) nên ta thêm luật  vào tập luật CARs, nó có nghĩa là “Nếu A = a2 thì class = 2” (độ hỗ trợ = 2, độ tin cậy = 100%).

*2.3.4.3. Đánh giá thuật toán:*

- Thuật toán CAR-Miner-Diff khai thác luật phân lớp kết hợp sử dụng cấu trúc cây giúp tiết kiệm bộ nhớ và thời gian khai thác luật, mỗi nút trên cây chứa thông tin cho việc tính toán nhanh độ hỗ trợ của luật ứng viên.

- Bằng cách sử dụng khái niệm *Diffset* (phần khác biệt giữa 2 Obidset)chúng ta có thể giảm không gian lưu trữ và thời gian tính toán cho việc giao giữa 2 tập Obidset với nhau.

- Ngoài ra, thuật toán còn có thể được áp dụng để cắt tỉa luật một cách nhanh chóng.

# PHƯƠNG PHÁP CẢI TIẾN ĐỀ XUẤT

## 3.1. Các phương pháp phân cụm dữ liệu phổ biến

### 3.1.1. Giới thiệu về kỹ thuật phân cụm

Phân cụm là quá trình phân chia một tập dữ liệu ban đầu thành các cụm dữ liệu sao cho các đối tượng trong một cụm đó “tương tự” với nhau. Phân cụm là một kỹ thuật trong khai thác dữ liệu nhằm tìm kiếm, phát hiện các cụm, các mẫu dữ liệu tự nhiên, tiềm ẩn, quan trọng trong tập dữ liệu lớn từ đó cung cấp thông tin, tri thức hữu ích cho việc ra quyết định. Mục đích chính của phân cụm nhằm khám phá cấu trúc của mẫu dữ liệu để thành lập các nhóm dữ liệu từ tập dữ liệu lớn, theo đó nó cho phép người ta đi sâu vào phân tích và nghiên cứu cho từng cụm dữ liệu này nhằm khám phá và tìm kiếm các thông tin tiềm ẩn, hữu ích phục vụ cho việc ra quyết định. Phương pháp phân cụm hỗ trợ giai đoạn tiền xử lý dữ liệu, mô tả sự phân bố dữ liệu/đối tượng,… Ví dụ: “Trong dữ liệu tín dụng ngân hàng, phân nhóm khách hàng có khả năng trả nợ cao.

Một số vấn đề thường gặp trong phân cụm là dữ liệu “nhiễu” và “phần tử ngoại lai”. “Nhiễu” có thể là các đối tượng dữ liệu không chính xác hoặc các đối tượng dữ liệu khuyết thiếu thông tin về một số thuộc tính. Một trong các kỹ thuật xử lý nhiễu phổ biến là việc thay thế giá trị của các thuộc tính của đối tượng “nhiễu” bằng giá trị thuộc tính tương ứng của đối tượng dữ liệu gần nhất. “Phần tử ngoại lai” là những phần tử có sự khác biệt đáng kể đối với những phần tử còn lại. Có nhiều cách xác định phần tử ngoại lai, như xác định theo khoảng cách: sử dụng hàm đo khoảng cách giữa các phần tử trong tập dữ liệu, các phần tử ngoại lai là các phần tử cách khá xa so với các phần tử còn lại. Xác định theo thống kê: xác định các mô hình phân phối thống kê mà các phần tử phải tuân theo, “phần tử ngoại lai” là những phần tử không tuân theo các quy luật này. Xác định theo độ khác biệt: xác định những đặc trưng cơ bản của các cụm, “phần tử ngoại lai” sẽ có đặc trưng khác biệt lớn với những phần tử còn lại.

Những vấn đề cần giải quyết khi phân cụm dữ liệu:

- Biểu diễn dữ liệu.

- Xây dựng hàm tính độ tương tự.

- Xây dựng các tiêu chuẩn phân cụm.

- Xây dựng mô hình cho cấu trúc cụm dữ liệu.

- Xây dựng thuật toán phân cụm và xác lập các điều kiện khởi tạo.

- Xây dựng các thủ tục biểu diễn và đánh giá kết quả phân cụm.

### 3.1.2. Ứng dụng của phân cụm dữ liệu

Phân cụm dữ liệu là một trong những công cụ chính của khai thác dữ liệu được ứng dụng trong nhiều lĩnh vực như thương mại và khoa học. Các kỹ thuật phân cụm đã được áp dụng cho một số ứng dụng điển hình trong các lĩnh vực sau:

*- Thương mại:* phân cụm có thể giúp các thương nhân khám phá ra các nhóm khách hàng quan trọng có các đặc trưng tương đồng nhau và đặc tả họ từ các mẫu mua bán trong CSDL khách hàng, phát hiện và dự đoán các giao dịch gian lận...

*- Sinh học:* phân cụm được sử dụng để phân cụm các loại sinh vật, phân loại các Gen với chức năng tương đồng và thu được các cấu trúc trong các mẫu, phát hiện và dự đoán các biến dị.

*- Lập quy hoạch đô thị:* Nhận dạng các nhóm nhà theo kiểu và vị trí địa lý,…  
nhằm cung cấp thông tin cho quy hoạch đô thị.

*- Địa lý:* Phân cụm các động vật, thực vật và đưa ra đặc trưng của chúng theo vị trí địa lý.

*- Khai phá Web:* Phân cụm có thể khám phá các nhóm tài liệu quan trọng, có nhiều ý nghĩa trong môi trường Web. Các lớp tài liệu này trợ giúp cho việc khám phá tri thức từ dữ liệu Web, khám phá ra các mẫu truy cập của khách hàng đặc biệt hay khám phá ra cộng đồng Web,…

### 3.1.3. Kiểu dữ liệu dựa trên kích thước miền và độ đo tương tự

*- Thuộc tính liên tục:* Nếu miền giá trị của nó là vô hạn không đếm được, nghĩa là giữa hai giá trị tồn tại vô số giá trị khác. Ví dụ: các thuộc tính về màu, nhiệt độ, cường độ âm thanh, tuổi,…

*- Thuộc tính rời rạc:* Nếu miền giá trị của nó là tập hữu hạn hoặc đếm được. ví dụ: các thuộc tính về số serial của một cuốn sách, số thành viên trong một gia  
đình, các cấp học, phòng ban trong công ty…

Để phân cụm, người ta phải đi tìm cách thích hợp để xác định “khoảng cách”  
giữa các đối tượng, hay là phép đo tương tự dữ liệu. Đây là các hàm để đo sự giống  
nhau giữa các cặp đối tượng dữ liệu, thông thường các hàm này hoặc là để tính độ  
tương tự (Similar) hoặc là tính độ phi tương tự (Dissimilar) giữa các đối tượng dữ liệu.

Tất cả các độ đo dưới đây được xác định trong không gian metric. Một không  
gian metric là một tập trong đó có xác định các “khoảng cách” giữa từng cặp phần tử, với những tính chất thông thường của khoảng cách hình học. Nghĩa là, một tập X (các phần tử của nó có thể là những đối tượng bất kỳ) các đối tượng dữ liệu trong cơ sở dữ liệu D được gọi là một không gian metric nếu:

Với mỗi cặp phần tử x, y thuộc X đều có xác định, theo một quy tắc nào đó, một số thực δ(x, y), được gọi là khoảng cách giữa x và y.

*Quy tắc trên thoả mãn hệ tính chất sau:*

- δ(x, y) > 0 nếu x ≠ y ;

- δ(x, y) = 0 nếu x = y;

- δ(x, y) = δ(y, x) với mọi x, y;

- δ(x, y) ≤ δ(x, z) + δ(z, y).

Hàm δ(x, y) được gọi là một metric của không gian. Các phần tử của  
X được gọi là các điểm của không gian này. Mỗi phép đo độ tương tự sẽ phù hợp với mỗi kiểu dữ liệu khác nhau.

Sau khi chuẩn hoá, độ đo phi tương tự của hai đối tượng dữ liệu *x*, *y* được xác  
định bằng các metric như sau:

*- Khoảng cách Minskowski*:  với q là số nguyên dương.

*- Khoảng cách Euclidean*: , (trường hợp đặc biệt của khoảng cách Minskowski trong trường hợp *q* = 2).

*- Khoảng cách Manhattan*: , (trường hợp đặc biệt của khoảng cách Minskowski trong trường hợp *q* = 1).

*- Khoảng cách cực đại*: , (trường hợp đặc biệt của khoảng cách Minskowski trong trường hợp *q* → ∞).

### 3.1.4. Một số kỹ thuật phân cụm phổ biến

Các phương pháp phân cụm tiêu biểu:

- Phân hoạch (partitioning): các phân hoạch được tạo ra và đánh giá theo  
một tiêu chí nào đó.

- Phân cấp (hierarchical): phân rã tập dữ liệu/đối tượng có thứ tự phân cấp  
theo một tiêu chí nào đó.

- Dựa trên mật độ (density-based): dựa trên kết nối và hàm mật độ (density functions).

- Dựa trên lưới (grid-based): dựa trên một cấu trúc đa cấp cụ thể.

- Dựa trên mô hình (model-based): một mô hình giả thuyết được đưa ra cho  
mỗi cụm; sau đó hiệu chỉnh các thông số để mô hình phù hợp với cụm dữ liệu/đối  
tượng nhất.

Trong phần này, luận văn đi vào trình bày 2 kỹ thuật phân cụm tiêu biểu là K-means (phương pháp phân hoạch) và Hierarchical Agglomerative Clustering (phương pháp phân cấp).

*3.1.4.1. Kỹ thuật phân cụm K-means*

Thuật toán phân cụm K-means do MacQueen đề xuất trong lĩnh vực thống kê  
năm 1967, mục đích của thuật toán K-means là sinh ra k cụm dữ liệu {C1, C2,…, Ck} từ một tập dữ liệu ban đầu gồm n đối tượng trong không gian d chiều Xi = (*xi1, xi2,…, xid*), i = (1,n), sao cho hàm tiêu chuẩn:  đạt giá trị tối  
thiểu. Trong đó: *mi* là trọg tâm của cụm *Ci*, D là khoảng cách giữa hai đối tượng.

Trọng tâm của một cụm là một vector, trong đó giá trị của mỗi phần tử của nó là trung bình cộng các thành phần tương ứng của các đối tượng vector dữ liệu trong cụm đang xét. Tham số đầu vào của thuật toán là số cụm k, tập CSDL gồm n phần tử và tham số đầu ra của thuật toán là các trọng tâm của các cụm dữ liệu. Độ đo khoảng cách D giữa các đối tượng dữ liệu thường được sử dụng là khoảng cách Euclidean, bởi vì đây là mô hình khoảng cách dễ để lấy đạo hàm và xác định các cực trị tối thiểu. Hàm tiêu chuẩn và độ đo khoảng cách có thể được xác định cụ thể hơn tuỳ vào ứng dụng hoặc các quan điểm của người dùng.

Thuật toán K-means được chứng minh là hội tụ và có độ phức tạp tính toán là: . Trong đó: n là số đối tượng dữ liệu, k là số cụm dữ liệu, d là  
số chiều, τ là số vòng lặp, *Tflop* là thời gian để thực hiện một phép tính cơ sở như  
phép tính nhân, chia,…

***Thuật toán K-means bao gồm các bước cơ bản như sau:***

***INPUT:*** Một CSDL gồm n đối tượng và số các cụm k.

***OUTPUT:*** Các cụm Ci (i=1,..,k) sao cho hàm tiêu chuẩn E đạt giá trị tối thiểu.

***Bước 1:*** Khởi tạo

Chọn k đối tượng mj (j=1...k) là trọng tâm ban đầu của k cụm từ tập dữ liệu (việc lựa chọn này có thể là ngẫu nhiên hoặc theo kinh nghiệm).

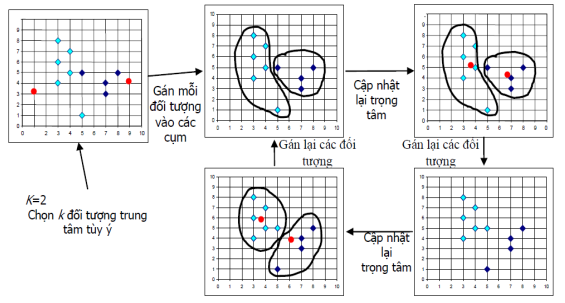
***Bước 2:*** Tính toán khoảng cách

Đối với mỗi đối tượng xi = (*xi1, xi2,…, xid*), tính toán khoảng cách từ nó tới mỗi  
trọng tâm mj (j=1,..,k), sau đó tìm trọng tâm gần nhất đối với mỗi đối tượng.

***Bước 3:*** Cập nhật lại trọng tâm

Đối với mỗi *j=1,..,k,* cập nhật trọng tâm cụm mj bằng cách xác định trung bình  
cộng của các vector đối tượng dữ liệu.

***Bước 4:*** Điều kiện dừng lặp các bước 2 và 3 cho đến khi các trọng tâm của cụm không thay đổi.



Hình .: Hình dạng cụm dữ liệu được khai phá bời K-means[21]

*Nhận xét:*

*-* Do K-means phân tích phân cụm đơn giản nên có thể áp dụng đối với  
tập dữ liệu lớn.

- K-means còn nhiều mặt hạn chế như: K-means chỉ áp dụng với dạng dữ liệu có thuộc tính số và có dạng hình cầu, K-means còn rất nhạy cảm với nhiễu và các phần tử ngoại lai trong dữ liệu. Ngoài ra, chất lượng phân cụm dữ liệu của thuật toán K-means phụ thuộc nhiều vào các tham số đầu vào như: số cụm k và k trọng tâm khởi tạo ban đầu. Trong trường hợp, các trọng tâm khởi tạo ban đầu mà quá lệch so với các trọng tâm cụm tự nhiên thì kết quả phân cụm của K-means là rất thấp, cụm dữ liệu được khám phá rất lệch so với các cụm trong thực tế.

- Hiện nay có rất nhiều thuật toán kế thừa tư tưởng của thuật toán K-means để  
khai phá dữ liệu mà có CSDL rất lớn như: K-medoid, PAM, CLARA, CLARANS, …

*3.1.4.2. Kỹ thuật phân cụm Hierarchical Agglomerative Clustering (HAC)*

Phân cụm theo phương pháp phân cấp là xây dựng nên một hệ thống phân cấp các cụm hay nói cách khác một cấu trúc cây các cụm gọi là *dendrogram*. Mỗi nút cụm chứa các cụm con; những điểm phân vùng cụm cùng cấp được bao phủ bởi các nút cha chung của chúng. Cách tiếp cận như vậy cho phép khám phá dữ liệu trên các mức chi tiết khác nhau. Phương pháp phân cụm phân cấp được phân loại thành dạng ***agglomerative*** (phân cụm từ dưới lên) và ***divisive*** (phân cụm từ trên xuống) [Jain & Dubes 1988; Kaufman & Rousseeuw 1990].

Phân cụm dạng ***agglomerative*** bắt đầu với các cụm một điểm (singleton) và hợp nhất đệ quy hai hoặc nhiều cụm thích hợp nhất. Phân cụm dạng ***divisive*** bắt đầu với một cụm của tất cả các điểm dữ liệu và phân chia đệ quy thành các cụm thích hợp nhất. Quá trình tiếp tục cho đến khi thỏa mãn một điều kiện dừng (thông thường là số *k* của cụm) đạt được.

Phương pháp phân cụm phân cấp liên quan đến ma trận khoảng cách (phi tương tự) hay độ tương tự giữa các điểm dữ liệu huấn luyện. Đôi khi được gọi là ma trận kết nối. Yêu cầu giữ một ma trận lớn như vậy trong bộ nhớ là không thực tế. Để thỏa mãn giới hạn này, các phương pháp khác nhau được sử dụng đưa vào ma trận kết nối thưa. Nó được thực hiện bằng cách bỏ qua các mục nhỏ hơn một ngưỡng nhất định, bằng cách chỉ sử dụng một tập hợp con nhất định của đại diện dữ liệu hoặc bằng cách với mỗi điểm như vậy chỉ giữ một số lượng nhất định điểm láng giềng gần nhất.

Sự phù hợp của một cụm phụ thuộc vào độ tương tự hay phi tương tự của các phần tử cụm. Điều này phản ánh một giả định chung rằng các cụm bao gồm các điểm tương tự. Một ví dụ quan trọng về độ phi tương tự giữa hai điểm là khoảng cách giữa chúng.

Để hợp nhất các tập con của các điểm thay vì các điểm riêng lẻ, khoảng cách giữa các điểm riêng lẻ phải được khái quát theo khoảng cách giữa các tập con. Ta dùng độ đo được gọi là ***linkage metric***. Loại linkage metric được sử dụng ảnh hưởng đáng kể đến các thuật toán phân cấp, vì nó phản ánh khái niệm cụ thể của sự gần nhau và kết nối. Các độ đo linkage metric giữa các cụm [Murtagh 1985, Olson 1995] bao gồm *single link*, *average link* và *complete link*. Sự khác biệt cơ bản độ đo (thông thường là khoảng cách) được tính cho mỗi cặp điểm với một điểm trong tập đầu và một điểm khác trong tập thứ hai. Một toán tử cụ thể như phép tối thiểu (*single link*), phép lấy trung bình (*average link*) hoặc phép cực đại (*complete link*) được áp dụng cho độ đo phi tương tự của một cặp như sau:



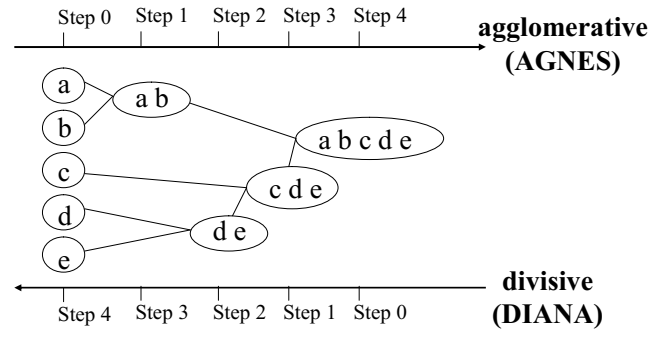
Các ví dụ ban đầu bao gồm thuật toán SLINK [Sibson 1973] thực hiện với *single link*, phương pháp Voorhees [Voorhees 1986] thực hiện với *average link* và thuật toán CLINK [Defays 1977] thực hiện với *complete link*. Trong đó, *single link* được tham chiếu đến nhiều nhất. Nó liên quan đến vấn đề tìm kiếm một cấu trúc cây bao trùm tối thiểu Euclide [Yao 1982] và có độ phức tạp *O(N2)*. Các phương pháp sử dụng khoảng cách liên cụm được xác định theo các cặp với các điểm trong cụm (tập con) tương ứng được gọi là phương pháp đồ thị (*graph*). Chúng không sử dụng bất kỳ đại diện cụm nào ngoài một tập các điểm. Mỗi phân vùng dữ liệu tương ứng với một phân vùng đồ thị. Phương pháp này có thể được gọi là phương pháp hình học (*geometric*) trong đó một cụm được đại diện bởi điểm trung tâm của nó. Nó dẫn đến các số đo về centroid, median và minimum variance linkage metric. Giả định các thuộc tính ở dạng số, điểm trung tâm được xác định là trọng tâm hoặc trung bình của hai cụm trung tâm kết tụ lại.

Tất cả các độ đo linkage metric ở trên có thể được lấy làm ví dụ trong công thức cập nhật của Lance-Williams [Lance & Williams 1967]



Trong đó *a, b, c* là các hệ số tương ứng với một liên kết cụ thể. Công thức này diễn tả một linkage metric giữa sự kết hợp của hai cụm và một cụm thứ ba với các thành phần cơ bản.

Phương pháp phân cụm phân cấp dựa trên linkage metric chịu sự phức tạp của thời gian, có độ phức tạp là *O(N2)* [Olson 1995]. Mặc dù độ phức tạp thời gian không thuận lợi, các thuật toán này vẫn được sử dụng rộng rãi. Một ví dụ là thuật toán AGNES (Agglomerative Nesting) [Kaufman & Rousseeuw 1990] được sử dụng trong S-Plus.



Hình .: Cấu trúc cây cụm dữ liệu được khai phá bởi HAC[21]

## 3.2. Phương pháp cải tiến đề xuất

Phương pháp phân cụm theo phân cấp thường được sử dụng để phân cụm dữ liệu bởi vì đặc tính đơn giản khi thực hiện cùng với hiệu quả tốt. Ngoài ra, phương pháp phân cụm phân cấp mang tính thông tin cao vì chúng cung cấp lược đồ dendogram sau khi phân cụm. Tuy nhiên, vấn đề chính của các phương pháp phân cụm phân cấp là chúng đòi hỏi thời gian thực hiện dài khi phân cụm một tập lớn dữ liệu.

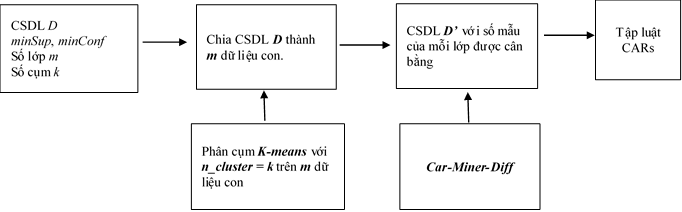
Phương pháp phân cụm phân hoạch có lợi thế trong các ứng dụng liên quan đến một tập lớn dữ liệu vì chúng phân cụm dữ liệu trực tiếp thành các cụm mà không tạo ra các lược đồ dendogram như phân cụm phân cấp. Do đó, phân cụm phân hoạch có thể phân cụm dữ liệu nhanh hơn nhiều. Thuật toán K-means là một trong những phương pháp phân cụm phân hoạch phổ biến nhất. Tuy nhiên, vì phương pháp K-means lại khá nhạy cảm với nhiễu nên hiệu quả phân cụm của nó bị hạn chế. Kỹ thuật phân cụm dựa trên học máy có thể học từ dữ liệu huấn luyện để cải thiện hiệu suất của chúng. Tuy nhiên, vì thời gian học thường khá lâu, các kỹ thuật học máy không phù hợp để phân cụm một tập lớn các dữ liệu.

Bởi vì phân cụm là một phương pháp học không giám sát, việc xác định số lượng cụm là khá quan trọng. Trong các phương pháp phân cụm theo phân cấp, người dùng có thể đặt ngưỡng dựa trên độ tương tự cụm làm điểm dừng cho quá trình phân cụm. Đối với các phương pháp khác, số cụm được xác định dựa trên quan sát thực nghiệm.

Bên cạnh đó, thông thường đối với một tập dữ liệu, người dùng chỉ sử dụng một phương pháp phân cụm cụ thể. Một vài trường hợp sử dụng phương pháp phân hoạch để dễ dàng thu về các cụm, tuy nhiên đối với các phương pháp này, chúng ta không thể dễ dàng kiểm soát chất lượng kết quả phân cụm và bắt buộc chúng ta phải xác định số cụm cụ thể. Một số khác sử dụng phương pháp phân cấp với kết quả phân cụm đạt độ chính xác cao hơn. Tuy nhiên, độ phức tạp của phương pháp phân cấp là O(n2) làm cho nó thực thi khá chậm đối với các dữ liệu lớn. Chính vì thế, chúng ta sẽ tiến hành kết hợp một vài kỹ thuật phân cụm với nhau để có thể tận dụng được ưu điểm của mỗi phương pháp cũng như cải thiện được kết quả chất lượng phân cụm.

Như chúng ta đã biết, trong các dữ liệu bị mất cân bằng về lớp, việc một số lớp chiếm đa số sẽ ảnh hưởng không nhỏ đến quá trình dự đoán dựa trên tập luật do gặp khó khăn trong việc chọn ngưỡng hỗ trợ tối thiểu. Nếu chọn ngưỡng quá cao dẫn đến các lớp chứa ít mẫu sẽ không phổ biến, vì vậy sẽ không có luật nào chứa lớp này, nếu chọn ngưỡng thấp để khai thác được các luật chứa lớp thiểu số thì số lượng luật của lớp đa số vẫn áp đảo nên cũng ảnh hưởng đến giai đoạn dự đoán lớp. Chính vì thế nên chúng ta sẽ thực hiện cân bằng số dữ liệu thuộc mỗi lớp trước, sau đó mới thực hiện khai thác luật.

Thuật toán K-means được kết hợp với thuật toán Car-Miner[17] bao gồm các bước cơ bản như sơ đồ hình 3.3**:**

****

Hình .: Các bước kết hợp thuật toán K-means với Car-Miner

***INPUT:*** Một CSDL *D*, *minSup*, *minConf*, số lớp m và số cụm *k*.

***OUTPUT:*** Tập luật *CARs* thỏa mãn *minSup* và *minconf*.

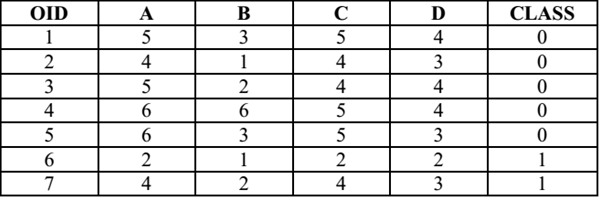
***Bước 1:*** : Chia CSDL *D* thành m dữ liệu con tương ứng với m giá trị của thuộc tính lớp. Gọi k là số dòng dữ liệu của dữ liệu con có số dòng dữ liệu ít nhất.

***Bước 2:*** Với mỗi dữ liệu con có số dòng lớn hơn k, tiến hành phân cụm K-means các dòng trong dữ liệu con đó thành k cụm. Mỗi cụm chỉ chọn một mẫu đại diện, trong bài báo này mẫu đại diện là phần tử gần trọng tâm của cụm nhất. Như vậy, kết quả mỗi dữ liệu con sẽ giữ lại *k* mẫu.

***Bước 3:*** Áp dụng thuật toán Car-Miner để khai thác trên tập dữ liệu tổng hợp từ m dữ liệu con.

Ví dụ: Chúng ta có bảng dữ liệu 3.1

Bảng .: Ví dụ về dữ liệu mất cân bằng về lớp[17]



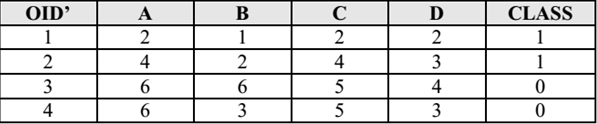
Ta thấy dữ liệu ở bảng trên có 5 mẫu thuộc lớp 0 và 2 mẫu thuộc lớp 1. Chúng ta sẽ cân bằng số mẫu thuộc lớp bằng phương pháp K-means như sau:

- Bước 1**:** Đầu tiên chúng ta chia cơ sở dữ liệu thành 2 bảng con tương ứng với 2 lớp 0 và 1. Do số dòng dữ liệu chứa lớp 1 là ít nhất (2 dòng) nên chọn số cụm là *k* = 2.

- Bước 2: Dùng kỹ thuật K-means gom các dòng có lớp là 0 thành 2 cụm. Mỗi cụm rút ra 1 dòng đại diện, ta có kết quả như bảng 3.2.

- Bước 3: Thực hiện khai thác luật phân lớp kết hợp bằng thuật toán khai thác luật phân lớp kết hợp với dữ liệu trong bảng 3.2.

Bảng .: Bảng CSDL sau khi đã cân bằng lớp[17]



Ngoài ra, một phương pháp phân cụm nên có số liệu để đánh giá và đảm bảo chất lượng của các cụm được tạo, chất lượng phân cụm càng tốt, các phần tử trong cụm càng tương tự nhau dẫn đến việc chọn ra mẫu đại diện cho cụm tốt hơn, luật sinh ra sẽ mang tính chính xác và khả thi hơn.

Trong luận văn này, chúng tôi sử dụng khái niệm độ tương tự trong cụm (*intra-cluster similarity*) để đo độ tương tự giữa các phần tử trong một cụm. Nếu giá trị này càng lớn, các phần tử trong cụm sẽ có độ tương tự cao hơn, do đó chất lượng phân cụm là tốt hơn.

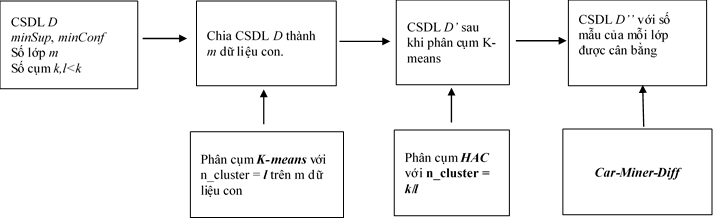
Gọi *C* là cụm có m phần tử, độ tương tự trong cụm của *C* (ký hiệu là ) là trung bình của độ tương tự giữa các mẫu trong *C* và được tính theo công thức sau:

 (3.1)

Trong đó, *Sim(Wi, Wj)* là độ tương tự giữa 2 mẫu trong một cụm.

K-means là một thuật toán phân cụm đơn giản, thời gian thực thi khá nhanh với độ phức tạp thuật toán là *O(n.k.d)* với *k* là số cụm, *n* là số mẫu và *d* là số lần lặp (Nazeer and Sebastian, 2009). Tuy nhiên, K-means không đảm bảo được độ tương tự trong cụm là đủ tốt. Ngược lại, thuật toán phân cụm HAC tuy có thời gian thực thi lâu hơn K-means do độ phức tạp *O(n2)* nhưng lại trả về kết quả phân cụm với độ tương tự trong cụm rất tốt[19].

Từ những ưu điểm và nhược điểm của K-means và HAC, chúng tôi đề xuất một mô hình phân cụm kết hợp cả hai phương pháp trên như hình 3.4



Hình .: Các bước kết hợp thuậ toán K-means + HAC với Car-Miner-Diff

***INPUT:*** Một CSDL *D*, *minSup*, *minConf*, số lớp m và số cụm *k, l*.

***OUTPUT:*** Tập luật *CARs* thỏa mãn *minSup* và *minconf*.

***Bước 1:*** : Chia CSDL *D* thành m dữ liệu con tương ứng với m giá trị của thuộc tính lớp. Gọi k là số dòng dữ liệu của dữ liệu con có số dòng dữ liệu ít nhất.

***Bước 2:*** Với mỗi dữ liệu con có số dòng lớn hơn k, tiến hành phân cụm K-means các dòng trong dữ liệu con đó thành l cụm nhỏ trước (trong thực nghiệm chọn *l = 10)*.

***Bước 3:*** Với mỗi cụm nhỏ ở bước 2, tiến hành phân cụm HAC các dòng dữ liệu trong mỗi cụm nhỏ đó thành *k/l* cụm. Mỗi cụm chỉ chọn một mẫu đại diện, trong bài báo này mẫu đại diện là phần tử gần trọng tâm của cụm nhất. Như vậy, kết quả m dữ liệu con sẽ giữ lại đúng hay gần đúng với *k* mẫu nhất.

***Bước 4:*** Áp dụng thuật toán Car-Miner-Diff để khai thác trên tập dữ liệu tổng hợp từ m dữ liệu con.

Chính vì chúng ta sử dụng phương pháp K-means ở bước đầu tiên nên đầu vào của thuật toán HAC sẽ là những cụm tương đối nhỏ, điều này làm cho thuật toán HAC chạy nhanh hơn nhiều trong bước thứ hai. Ngoài ra, việc phân cụm có thể được kiểm soát bởi HAC cho chất lượng phân cụm tốt hơn. Kết quả thực nghiệm sẽ được trình bày trong chương 4.

# THỰC NGHIỆM VÀ ĐÁNH GIÁ

## 4.1. Môi trường và cơ sở dữ liệu thực nghiệm

### 4.1.1. Môi trường thực nghiệm

Các thuật toán được cài đặt trên môi trường Python phiên bản 3.7.3, hệ điều hành Windows 10 Professional 64 bit, CPU Intel Core i3 1.7GHz, RAM 4GB, 500GB.

Luận văn sử dụng công cụ Pyqt5 là Python Interface của Qt để tạo ra các giao diện chương trình phục vụ mô phỏng dữ liệu và báo cáo.

### 4.1.2. Giới thiệu về Pyqt5

*4.1.2.1. Giới thiệu chung*

- Qt là một Application framework đa nền tảng viết trên ngôn ngữ C++ , được dùng để phát triển các ứng dụng trên desktop, hệ thống nhúng và mobile. Hỗ trợ cho các platform bao gồm: Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS và một số platform khác. PyQt là Python Interface của Qt, kết hợp của ngôn ngữ lập trình Python và thư viện Qt, là một thư viện bao gồm các thành phần giao diện điều khiển (**widgets**, **graphical control elements**).

- PyQt API bao gồm các module số lượng lớn các **classes**và **functions**hỗ trợ cho việc thiết kế các giao diện giao tiếp với người dùng của các phần mềm chức năng. Hỗ trợ với Python 2.x và 3.x.

- PyQt được phát triển bởi Riverbank Computing Limited, các class của PyQt5 được chia thành các module, bao gồm :

**+** **QtCore:** là module bao gồm phần lõi không thuộc chức năng GUI, ví dụ: dùng để làm việc với thời gian, file và thư mục, các loại dữ liệu, streams, URLs, mime type, threads hoặc processes.

**+ QtGui:** bao gồm các class dùng cho việc lập trình giao diện (windowing system integration), event handling, 2D graphics, basic imaging, fonts và text.

**+ QtWidgets:** bao gồm các class cho widget, ví dụ : button, hộp thoại, … được sử dụng để tạo nên giao diện người dùng cơ bản nhất.

**+ QtMultimedia:** thư viện cho việc sử dụng âm thanh, hình ảnh, camera,…

**+ QtBluetooth:** bao gồm các class giúp tìm kiếm và kết nối với các thiết bị có giao tiếp với phần mềm.

**+ QtNetwork:** bao gồm các class dùng cho việc lập trình mạng, hỗ trợ lập trình TCP/IP và UDP client , server hỗ trợ việc lập trình mạng.

**+ QtPositioning:** bao gồm các class giúp việc hỗ trợ xác định vị.

**+ Enginio:** module giúp các client truy cập các Cloud Services của Qt.

**+ QtWebSockets:** cung cấp các công cụ cho WebSocket protocol.

**+ QtWebKit:** cung cấp các class dùng cho làm việc với các trình duyệt Web , dựa trên thư viện WebKit2.

**+ QtWebKitWidgets:** các widget cho WebKit.

**+ QtXml:** các class dùng cho làm việc với XML file.

**+ QtSvg:** dùng cho hiển thị các thành phần của SVG file.

**+ QtSql:** cung cấp các class dùng cho việc làm việc với dữ liệu.

**+ QtTest:** cung cấp các công cụ cho phép test các đơn vị của ứng dụng với PyQt5.

*4.1.2.2. Giới thiệu các tool công cụ phần mềm phục vụ thiết kế với Pyqt5*

- **Qt Designer**: Qt sử dụng IDE tên Qt Creator với một tool thiết kế giao diện người dùng Qt Designer. Qt Designer có thể làm việc một mình độc lập với Qt Creator.

Qt Designer sử dụng XML .ui file  để lưu thiết kế và không sinh thêm bất kỳ mã nguồn nào của nó. User Interface Compiler (uic) đọc định dạng file XML (.ui) và xuất ra header file mã nguồn C++ tương ứng. Qt có một class QUiLoader cho phép một ứng dụng tải một file .ui và tạo một giao diện động tương ứng.

- **uic Python module**: PyQt5 không chứa class QUiLoader nhưng thay vào đó là module Python uic. Cũng giống như QUiLoader , module python uic này cũng tải định dạng file .ui và tạo giao diện động tương ứng. Giống như UIC (User Interface Compiler) module python uic này cũng sinh ra mã nguồn python tạo nên giao diện tương ứng.

- **Python3** : python là một ngôn ngữ lập trình bậc cao, linh hoạt, là ngôn ngữ thông dịch và là một ngôn ngữ động. Python do Guido van Rossum tạo ra năm 1990. Các phiên bản của Python trong quá trình phát triển: Python 1 – giai đoạn 1990 tới 1995; Python 2 – phát hành vào năm 2000; Python 3 , python 3000 hoặc py3k được phát hành 3/12/2008.

### 4.1.3. Giới thiệu về giao diện chương trình

*4.1.3.1. Giao diện chính của chương trình*



Hình .: Giao diện chính chương trình

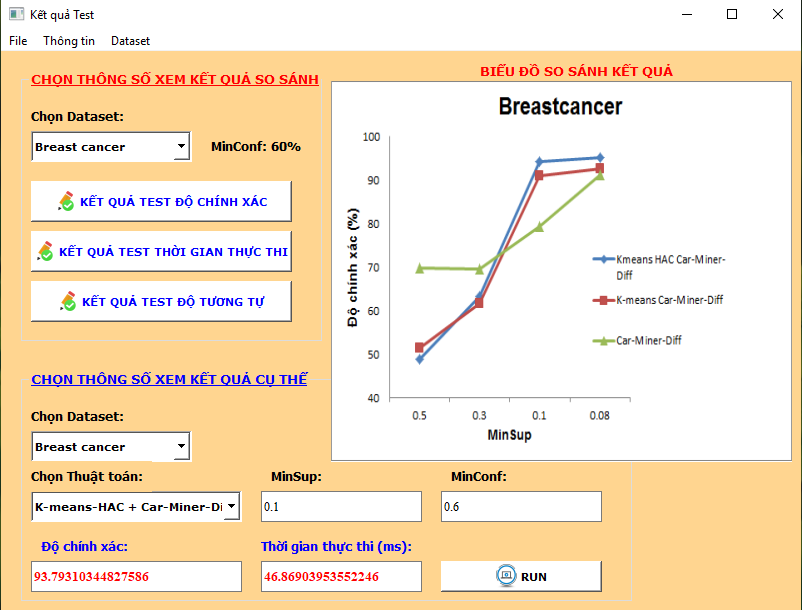
Giao diện chính chương trình báo cáo luận văn được thể hiện như hình 4.1, trong đó bao gồm thông tin chương trình, người dùng có thể chọn lựa như sau:

- Nhấn nút **“KẾT QUẢ TEST”**: để xem kết quả thực nghiệm của các thuật toán trên cơ sở dữ liệu chuẩn.

- Nhấn nút **“DEMO ATGT”**: để xem kết quả thực nghiệm trên dữ liệu thực tế về an toàn giao thông.

- Nhấn nút **“EXIT”**: để thoát khỏi chương trình.

*4.1.3.2. Giao diện chương trình Kết quả test*



Hình .: Giao diện chương trình Kết quả test

Giao diện chính chương trình Kết quả test trên dữ liệu chuẩn được thể hiện như hình 4.2, trong đó chương trình được chia làm 02 phần:

- Phần I: Người dùng chọn thông số xem kết quả so sánh trên những dữ liệu chuẩn khác nhau; kết quả so sánh về độ chính xác, thời gian thực thi thuật toán và độ tương tự sẽ được hiển thị qua biểu đồ so sánh kết quả.

- Phần II: Người dùng có thể chọn thông số xem kết quả cụ thể. Bằng cách chọn các thông số về dataset, kiểu thuật toán, ngưỡng hỗ trợ minSup và ngưỡng độ tin cậy minConf, người dùng có thể xem kết quả hiển thị về độ chính xác và thời gian thực thi.

*4.1.3.3. Giao diện chương trình khai thác dữ liệu an toàn giao thông*

Giao diện chính chương trình Kết quả test trên dữ liệu chuẩn được thể hiện như hình 4.3.



Hình .: Giao diện chương trình dự đoán ATGT

Trong đó chương trình được chia làm 02 phần:

- Phần I: Người dùng chọn thông số xem kết quả dự đoán cụ thể thông qua các thông tin về: Ngày vi phạm, Giới tính, Độ tuổi vi phạm, Vùng miền nơi cư trú, Nghề nghiệp, Địa bàn xảy ra vi phạm, Có/Không vi phạm Lỗi về nồng độ cồn/tốc độ, Có/Không vi phạm Lỗi về tín hiệu rẽ, Có/Không vi phạm nhiều lỗi cùng lúc, để từ đó dựa vào tập luật sẽ có dự đoán khả năng gây tai nạn giao thông như thế nào.

- Phần II: Người dùng có thể dự đoán khả năng gây tai nạn giao thông từ một danh sách trong file ImportList.xlsx và xuất ra kết quả qua file ExportList.xlsx theo yêu cầu; kết quả cũng được thể hiện trên giao diện chương trình.

### 4.1.4. Cơ sở dữ liệu thực nghiệm

- Các CSDL chuẩn dùng để thực nghiệm được lấy từ website UCI http://mlearn.ics.uci.edu (Bảng 4.1).

- Dữ liệu xử phạt vi phạm hành chính giao thông và xử lý tai nạn của phòng Cảnh sát giao thông đường bộ - đường sắt công an tỉnh Đắk Nông.

Bảng .: CSDL chuẩn thực nghiệm

| **Tập dữ liệu** | **Số thuộc tính** | **Số lớp** | **Số mẫu** | **Mô tả** |
| --- | --- | --- | --- | --- |
| Breast cancer | 9 | 2 | 683 | - Lớp 0: 444 (65%) - Lớp 1: 239 (35%) |
| Chess | 10 | 2 | 1200 | - Lớp 0: 900 (75%) - Lớp 1: 300 (25%) |
| Diabetes | 8 | 2 | 1400 | - Lớp 0: 942 (67.3%) - Lớp 1: 458 (32.7%) |
| Tic-tac-toe | 9 | 2 | 958 | - Lớp 0: 332 (34.6%)  - Lớp 1: 626 (65.4%) |

## 4.2. Kết quả so sánh về độ chính xác

Để so sánh đánh giá kết quả về độ chính xác của 03 thuật toán: Car-Miner-Diff, K-means\_Car-Miner-Diff, K-means\_HAC\_Car-Miner-Diff, luận văn sử dụng 04 CSDL chuẩn để tiến hành cài đặt:

- Tập dữ liệu Breast cancer: ngưỡng độ tin cậy minConf = 60% và ngưỡng hỗ trợ minSup lần lượt thay đổi: 50%, 30%, 10%, 8%.

- Tập dữ liệu Chess: ngưỡng độ tin cậy minConf = 60% và ngưỡng hỗ trợ minSup lần lượt thay đổi: 50%, 30%, 10%, 8%.

- Tập dữ liệu Diabetes: ngưỡng độ tin cậy minConf = 60% và ngưỡng hỗ trợ minSup lần lượt thay đổi: 10%, 5%, 1%, 0.5%.

- Tập dữ liệu Tic-tac-toe: ngưỡng độ tin cậy minConf = 60% và ngưỡng hỗ trợ minSup lần lượt thay đổi: 30%, 10%, 8%, 5%.

Độ chính xác phân lớp trên các CSDL ở bảng 4.1 được thể hiện cụ thể qua bảng 4.2 đến 4.5.

Bảng .: Kết quả thực nghiệm về độ chính xác trên CSDL Breast cancer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CSDL Breast cancer: Độ chính xác (%)** | | | | | |
| **MINSUP** | **0.5** | **0.3** | **0.1** | **0.08** |
| **K-means\_HAC\_Car-Miner-Diff** | 48.8966 | 63.1724 | 94.2759 | 95.1724 |
| **K-means\_Car-Miner-Diff** | 51.3235 | 61.5441 | 90.8088 | 92.6471 |
| **Car-Miner-Diff** | 69.8049 | 69.561 | 79.3659 | 91.2195 |

Bảng .: Kết quả thực nghiệm về độ chính xác trên CSDL Chess

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CSDL Chess: Độ chính xác (%)** | | | | | |
| **MINSUP** | **0.5** | **0.3** | **0.1** | **0.08** |
| **K-means\_HAC\_Car-Miner-Diff** | 51.1111 | 77.2222 | 83.8889 | 85.5556 |
| **K-means\_Car-Miner-Diff** | 78.9474 | 78.9474 | 78.9474 | 78.9474 |
| **Car-Miner-Diff** | 75.5556 | 75.5556 | 75.5556 | 75.5556 |

Bảng .: Kết quả thực nghiệm về độ chính xác trên CSDL Diabetes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CSDL Diabetes: Độ chính xác (%)** | | | | | |
| **MINSUP** | **0.1** | **0.05** | **0.01** | **0.005** |
| **K-means\_HAC\_Car-Miner-Diff** | 75.6345 | 82.7411 | 86.802 | 87.3096 |
| **K-means\_Car-Miner-Diff** | 78.0749 | 78.0749 | 83.9572 | 84.492 |
| **Car-Miner-Diff** | 71.9048 | 76.6667 | 79.7619 | 80.7143 |

Bảng .: Kết quả thực nghiệm về độ chính xác trên CSDL Tic-tac-toe

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CSDL Tic-tac-toe: Độ chính xác (%)** | | | | | |
| **MINSUP** | **0.3** | **0.1** | **0.08** | **0.05** |
| **K-means\_HAC\_Car-Miner-Diff** | 53.2843 | 62.9902 | 74.0196 | 99.0196 |
| **K-means\_Car-Miner-Diff** | 50.7 | 70 | 69 | 88.5 |
| **Car-Miner-Diff** | 68.4722 | 67.3611 | 67.3611 | 75.6944 |

Kết quả thực nghiệm so sánh độ chính xác giữa 03 thuật toán được trình bày trong các hình từ 4.4 đến 4.7.

Hình .: So sánh về độ chính xác trên CSDL Breast cancer

Hình .: So sánh về độ chính xác trên CSDL Chess

Hình .: So sánh về độ chính xác trên CSDL Chess

Hình .: So sánh về độ chính xác trên CSDL Tic-tac-toe

**Nhận xét:** Các kết quả từ hình 4.4 đến hình 4.7 cho thấy đối với các CSDL mất cân bằng về lớp, phương pháp cải tiến K-means\_HAC\_Car-Miner-Diff cho kết quả tốt hơn về độ chính xác, đặc biệt đối với các ngưỡng minSup nhỏ.

## 4.3. Kết quả so sánh về thời gian thực hiện thuật toán

Để so sánh đánh giá kết quả về thời gian khai thác luật của 02 thuật toán: Car-Miner-Diff, K-means\_HAC\_Car-Miner-Diff, luận văn sử dụng 04 CSDL chuẩn để tiến hành cài đặt:

- Tập dữ liệu Breast cancer: ngưỡng độ tin cậy minConf = 60% và ngưỡng hỗ trợ minSup lần lượt thay đổi: 50%, 30%, 10%, 8%.

- Tập dữ liệu Chess: ngưỡng độ tin cậy minConf = 60% và ngưỡng hỗ trợ minSup lần lượt thay đổi: 50%, 30%, 10%, 8%.

- Tập dữ liệu Diabetes: ngưỡng độ tin cậy minConf = 60% và ngưỡng hỗ trợ minSup lần lượt thay đổi: 10%, 5%, 1%, 0.5%.

- Tập dữ liệu Tic-tac-toe: ngưỡng độ tin cậy minConf = 60% và ngưỡng hỗ trợ minSup lần lượt thay đổi: 30%, 10%, 8%, 5%.

Thời gian thực hiện thuật toán trên các CSDL ở bảng 4.1 được thể hiện cụ thể qua bảng 4.6 đến 4.9.

Bảng .: Kết quả thực nghiệm về thời gian thực thi trên CSDL Breast cancer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CSDL Breast cancer: Thời gian thực thi (ms)** | | | | | |
| **MINSUP** | **0.5** | **0.3** | **0.1** | **0.08** |
| **Kmeans HAC Car-Miner-Diff** | 0 | 3 | 43 | 63 |
| **Car-Miner-Diff** | 7 | 16 | 91 | 108 |

Bảng .: Kết quả thực nghiệm về thời gian thực thi trên CSDL Chess

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CSDL Chess: Thời gian thực thi (ms)** | | | | | |
| **MINSUP** | **0.5** | **0.3** | **0.1** | **0.08** |
| **Kmeans HAC Car-Miner-Diff** | 1 | 41 | 270 | 330 |
| **Car-Miner-Diff** | 4 | 43 | 330 | 400 |

Bảng .: Kết quả thực nghiệm về thời gian thực thi trên CSDL Diabetes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CSDL Diabetes: Thời gian thực thi (ms)** | | | | | |
| **MINSUP** | **0.1** | **0.05** | **0.01** | **0.005** |
| **Kmeans HAC Car-Miner-Diff** | 55 | 130 | 570 | 900 |
| **Car-Miner-Diff** | 83 | 180 | 580 | 950 |

Bảng .: Kết quả thực nghiệm về thời gian thực thi trên CSDL Tic-tac-toe

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CSDL Tic-tac-toe: Thời gian thực thi (ms)** | | | | | |
| **MINSUP** | **0.3** | **0.1** | **0.08** | **0.05** |
| **Kmeans HAC Car-Miner-Diff** | 0 | 67 | 107 | 210 |
| **Car-Miner-Diff** | 1 | 97 | 120 | 260 |

Kết quả thực nghiệm so sánh thời gian thực thi giữa 02 thuật toán được trình bày trong các hình từ 4.8 đến 4.11.

Hình .: So sánh về thời gian thực thi trên CSDL Breast cancer

Hình .: So sánh về thời gian thực thi trên CSDL Chess

Hình .: So sánh về thời gian thực thi trên CSDL Diabetes

Hình .: So sánh về thời gian thực thi trên CSDL Tic-tac-toe

**Nhận xét:** Các kết quả từ hình 4.8 đến hình 4.11 cho thấy đối với các CSDL mất cân bằng về lớp, phương pháp cải tiến K-means\_HAC\_Car-Miner-Diff do xử lý với ít mẫu hơn sau khi phân cụm nên cho kết quả tốt hơn về thời gian thực thi, đặc biệt đối với các ngưỡng minSup nhỏ.

## 4.4. Kết quả so sánh về độ tương tự sau khi phân cụm

Để so sánh đánh giá kết quả về độ tương tự giữa các phần tử sau khi phân cụm của 02 phương pháp: K-means và K-means\_HAC, luận văn đã tiến hành phân cụm trên 04 CSDL chuẩn: Breast cancer, Chess, Diabetes, Tic-tac-toe với số cụm n = .

Sử dụng công thức 3.1 để đánh giá độ tương tự sau khi phân cụm, kết quả cụ thể được thể hiện ở Bảng 4.10 và hình 4.11.

Bảng .: Kết quả thực nghiệm về độ tương tự khi phân cụm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Độ tương tự trong cụm** | | | | |
| **Dataset** | **Breast** | **Diabetes** | **Tic-tac-toe** | **Chess** |
| **K-means + HAC** | 0.725179602 | 0.311769548 | 0.387909446 | 0.5525347 |
| **K-means** | 0.575212808 | 0.264289639 | 0.330269319 | 0.3050754 |

Hình .: Kết quả so sánh độ tương tự khi phân cụm

**Nhận xét:** Kết quả từ hình 4.12 cho thấy sau khi phân cụm, phương pháp K-means\_HAC cho kết quả tốt hơn về độ tương tự của các phần tử trong cụm, dẫn đến chất lượng phân cụm tốt hơn phương pháp K-means.

## 4.5. Khai thác dữ liệu xử phạt vi phạm hành chính giao thông

Trong luận văn này, tác giả đã dựa vào phương pháp cải tiến đề xuất để xây dựng nên chương trình dự đoán an toàn giao thông. Với các thông số liên quan đến dữ liệu xử phạt vi phạm hành chính và dữ liệu về xử lý tai nạn đã thu thập được với 4368 mẫu dữ liệu, chương trình sẽ làm nhiệm vụ xử lý và dự đoán khả năng gây tai nạn khi các cá nhân có tình trạng tương tự như tập luật đã khai thác.

Ngoài ra từ tập luật khai thác được, chúng ta có thể thấy được những vấn đề về tình hình vi phạm TTATGT như những hành vi nào thường xuyên vi phạm, thời gian nào người điều khiển vi phạm nhiều nhất, địa bàn nào nổi cộm nhất, những hành vi nào gây tai nạn nhiều nhất, đặc biệt là khả năng dự báo gây tai nạn đối với những trường hợp tương tự… từ đó phục vụ cho công tác nghiệp vụ, hỗ trợ trong quá trình ra quyết định, đề xuất những chính sách quản lý hợp lý trên địa bản quản lý.

Chương trình có giao diện như hình 4.3, cụ thể như sau:

- Phần I: Người dùng chọn thông số xem kết quả dự đoán cụ thể thông qua các thông tin về:

+ Ngày vi phạm: Ngày trong tuần, Ngày cuối tuần.

+ Giới tính: Nam, Nữ.

+ Độ tuổi: 15 - 24 tuổi, 25 - 39 tuổi, Từ 40 tuổi trở lên.

+ Vùng miền: Tây Nguyên, Địa bàn khác.

+ Nghề nghiệp: Người lao động, Công chức/Giáo viên, Học sinh/Sinh viên.

+ Địa bàn vi phạm: Thị trấn, Địa bàn xã.

+ Lỗi nồng độ cồn, tốc độ: Không vi phạm, Có vi phạm.

+ Lỗi tín hiệu rẽ: Không vi phạm, Có vi phạm.

+ Vi phạm nhiều lỗi cùng lúc: Không vi phạm, Có vi phạm.

Sau khi chọn lựa các giá trị liên quan thuộc tính, người dùng nhấn nút “**RUN”** để chương trình dự đoán khả năng gây tai nạn giao thông như thế nào dựa vào tập luật đã khai thác được.

- Phần II: Người dùng có thể dự đoán khả năng gây tai nạn giao thông từ một danh sách trong file ImportList.xlsx bằng cách nhấn nút **“TẢI DỮ LIỆU”** và xuất ra kết quả qua file ExportList.xlsx bằng cách nhấn nút **“XUẤT DỮ LIỆU”** theo yêu cầu; kết quả cũng được thể hiện trên giao diện bảng của chương trình.

- Ngoài ra, người dùng có thể xem thông tin của các tập luật có độ tin cậy cao trong mục **“Xem luật”** của chương trình.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1. Kết luận

Luận văn đã đáp ứng được các nội dung đăng ký trong đề cương. Cụ thể luận văn đã đạt được một số kết quả sau:

- Tìm hiểu tổng quan những khái niệm liên quan đến khám phá tri thức, khai thác dữ liệu, khai thác luật kết hợp và khai thác luật phân lớp kết hợp.

- Tìm hiểu và so sánh ưu điểm, hạn chế của các thuật toán khai thác luật kết hợp và luật phân lớp kết hợp.

- Tìm hiểu về các kỹ thuật phân cụm dữ liệu K-means, Hierarchical Agglomerative Clustering (HAC).

- Luận văn đã đề xuất phương pháp cải tiến kết hợp hai kỹ thuật phân cụm K-means và HAC, cài đặt thuật toán trên các CSDL chuẩn, so sánh độ chính xác, thời gian thực hiện giữa phương pháp đề xuất và thuật toán CAR-Miner-Diff ban đầu để kiểm chứng; đồng thời luận văn cũng tiến hành so sánh độ tương tự sau khi phân cụm đối với 02 phương pháp K-means và K-means\_HAC.

- Ứng dụng phương pháp đề xuất xây dựng chương trình khai thác dữ liệu xử phạt vi phạm hành chính giao thông trên thực tế, dự đoán khả năng gây tai nạn khi được cung cấp một số thông tin thuộc tính liên quan.

## 5.2. Hướng phát triển

Trong tương lai, tác giả sẽ tiếp tục thử nghiệm trên nhiều loại CSDL hơn với số lớp tăng lên để đánh giá khả năng ứng dụng của phương pháp cải tiến đề xuất. Đồng thời, tác giả sẽ áp dụng phương pháp này với các loại phân lớp khác như cây quyết định, Ngoài ra, phương pháp này cũng sẽ được áp dụng vào các loại phân lớp khác như cây quyết định, ILA, mạng neural…

Mặc dù luận văn đã xây dựng được chương trình dự đoán ATGT trên bộ số liệu về xử lý vi phạm hành chính về trật tự ATGT và xử lý tai nạn giao thông thật song ứng dụng mới chỉ được cài đặt và phát triển theo các chuyên đề định sẵn đã nói ở trên, vẫn còn rất nhiều chuyên đề khai thác dữ liệu khác cần được cài đặt. Do đó nếu có điều kiện chương trình ứng dụng sẽ được phát triển mở rộng theo các chuyên đề mới phục vụ cho công tác nghiệp vụ của lực lượng công an nhân dân.

# DANH MỤC TÀI LIỆU THAM KHẢO

1. B. Liu, W. Hsu, Y. Ma (1998). Integrating classification and association rule mining. In Proc. of the 4th International Conference on Knowledge Discovery and Data Mining, New York, USA, pp. 80-86.
2. B. Liu, Y. Ma, C.K. Wong (2000). Improving an association rule based classifier. In: Proc. of *The 4th European Conference on Principles of Data Mining and Knowledge Discovery*, Lyon, France, pp. 80-86.
3. W. Li, J. Han, J. Pei (2001), CMAR: Accurate and efficient classification based on multiple class-association rules, 1st IEEE international conference on Data mining, pp. 369–376.
4. Yin X. and Han J., “CPAR: Classification based on Predictive Association Rules,” in Proceedings of SIAM International Conference on Data Mining, San Francisco, USA, pp. 331-335, 2003.
5. A. Veloso, W. Meira Jr., M.J. Zaki (2006). Lazy associative classification. In: Proc. of *The 2006 IEEE International Conference on Data Mining (ICDM’06)*, Hong Kong, China, pp. 645- 654.
6. A. Veloso, W. Meira Jr., M. Goncalves, H.M. Almeida, M.J. Zaki (2007). Multi-label lazy associative classification. In: Proc. of *The 11th European Conference on Principles of Data Mining and Knowledge Discovery*, Warsaw, Poland, pp. 605-612.
7. A. Veloso, W. Meira Jr., M. Goncalves, H.M. Almeida, M.J. Zaki (2011). Calibrated lazy associative classification. *Information Sciences* 181(13), pp. 2656-2670.
8. Vo B. and Le B. (2008), “A Novel Classification Algorithm based on Association rule Mining,” in Proceedings of Pacific Rim Knowledge  
   Acquisition Workshop, Viet Nam, pp. 61-75.
9. L. T. T. Nguyen, B. Vo, T. P. Hong, H. C. Thanh (2013). CAR-Miner: An efficient algorithm for mining class-association rules. Expert Systems with Applications, vol.40, no.6, pp. 2305-2311.
10. L. T. T. Nguyen, N. T. Nguyen (2015). An improved algorithm for mining class association rules using the difference of Obidsets. Expert Systems with Applications, vol.42, no.9, pp. 4361-4369.
11. R. Quinlan (1992), C4.5: programs for machine learning, Machine Learning, vol.16, pp. 235-240.
12. F. A. Thabtah, P. Cowling, Y. Peng (2004), MMAC: A new multi-class, multi-label associative classification approach, the 4th IEEE International Conference on Data mining, pp. 217-224.
13. F. Thabtah, P. Cowling, Y. Peng (2005), MCAR: Multi-class classification based on association rule, 3rd ACS/IEEE international conference on computer systems and applications, pp. 33–39.
14. M. R. Tolun, S. M. Abu-Soud (1998), ILA: an inductive learning algorithm for rule extraction, Expert Systems with Applications, vol.14, no.3, pp. 361– 370.
15. J. Wu (2012), Advances in K-means clustering: a data mining thinking. Springer Sceience & Business Media, pp. 17-35.
16. X. Wu et al. (2008), Top 10 algorithms in data mining. Knowledge and Information Systems, vol.14, no.1, pp. 1-37.
17. Nguyễn và đồng sự, *“Khai thác luật phân lớp kết hợp trên cơ sở dữ liệu mất cân bằng về lớp”, Kỷ yếu Hội nghị Khoa học Quốc gia lần thứ IX “Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR’9)”*, 2016.
18. Khanali and Vaziri, “*A Survey on Improved Algorithms for Mining  
    Association Rules”*, International Journal of Computer Applications (0975 – 8887) Volume 165 – No.9, 2017.
19. K.T.Huynh et al. (2017), “*A quality-controlled logic-based clustering approach for web service composition and verification”*, International Journal of Web Information Systems, Vol. 13 Issue: 2, pp.173-198.
20. Võ Thiện Khoa, “*Khai thác tập mục hữu ích cao trên cơ sở dữ liệu tăng trưởng”*, Luận văn tốt nghiệp thạc sĩ, 2015.
21. Nguyễn Đặng Thế Vinh, “*Ứng dụng khai phá dữ liệu chọn ngành nghề cho học sinh THPT”*, Luận văn tốt nghiệp thạc sĩ, 2014.
22. Jiawei Han and Micheline Kamber, *Data Mining: Concepts and Techniques*, 3rd Edition. Morgan Kaufmann Publishers, 2011.
23. PGS.TS. Võ Đình Bảy, *“Bài giảng Khai thác luật kết hợp”*, Đại học Công nghệ TPHCM.