

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**Nguyễn Thị Thùy Linh**

**NGHIÊN CỨU CÁC THUẬT TOÁN PHÂN LỚP DỮ LIỆU  
DỰA TRÊN CÂY QUYẾT ĐỊNH**

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**

**Ngành: Công nghệ thông tin**

**HÀ NỘI - 2005**

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**Nguyễn Thị Thùy Linh**

**NGHIÊN CỨU CÁC THUẬT TOÁN PHÂN LỚP DỮ LIỆU  
DỰA TRÊN CÂY QUYẾT ĐỊNH**

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**

**Ngành: Công nghệ thông tin**

**Cán bộ hướng dẫn: TS. Nguyễn Hải Châu**

**HÀ NỘI - 2005**

## TÓM TẮT NỘI DUNG

Phân lớp dữ liệu là một trong những hướng nghiên cứu chính của khai phá dữ liệu. Công nghệ này đã, đang và sẽ có nhiều ứng dụng trong các lĩnh vực thương mại, ngân hàng, y tế, giáo dục... Trong các mô hình phân lớp đã được đề xuất, cây quyết định được coi là công cụ mạnh, phổ biến và đặc biệt thích hợp với các ứng dụng khai phá dữ liệu. Thuật toán phân lớp là nhân tố trung tâm trong một mô hình phân lớp.

Khóa luận đã nghiên cứu vấn đề phân lớp dữ liệu dựa trên cây quyết định. Từ đó tập trung vào phân tích, đánh giá, so sánh hai thuật toán tiêu biểu cho hai phạm vi ứng dụng khác nhau là C4.5 và SPRINT. Với các chiến lược riêng về lựa chọn thuộc tính phát triển, cách thức lưu trữ phân chia dữ liệu, và một số đặc điểm khác, C4.5 là thuật toán phổ biến nhất khi phân lớp tập dữ liệu vừa và nhỏ, SPRINT là thuật toán tiêu biểu áp dụng cho những tập dữ liệu có kích thước cực lớn. Khóa luận đã chạy thử nghiệm mô hình phân lớp C4.5 với tập dữ liệu thực và thu được một số kết quả phân lớp có ý nghĩa thực tiễn cao, đồng thời đánh giá được hiệu năng của mô hình phân lớp C4.5. Trên cơ sở nghiên cứu lý thuyết và quá trình thực nghiệm, khóa luận đã đề xuất một số cải tiến mô hình phân lớp C4.5 và tiến tới cài đặt SPRINT.

## LỜI CẢM ƠN

Trong suốt thời gian học tập, hoàn thành khóa luận em đã may mắn được các thầy cô chỉ bảo, dìu dắt và được gia đình, bạn bè quan tâm, động viên.

Em xin được bày tỏ lòng biết ơn chân thành tới các thầy cô trường Đại học Công Nghệ đã truyền đạt cho em nguồn kiến thức vô cùng quý báu cũng như cách học tập và nghiên cứu khoa học.

Cho phép em được gửi lời cảm ơn sâu sắc nhất tới TS. Nguyễn Hải Châu, người thầy đã rất nhiệt tình chỉ bảo và hướng dẫn em trong suốt quá trình thực hiện khóa luận.

Với tất cả tấm lòng mình, em xin bày tỏ lòng biết ơn sâu sắc đến TS. Hà Quang Thụy đã tạo điều kiện thuận lợi và cho em những định hướng nghiên cứu. Em xin lời cảm ơn tới Nghiên cứu sinh Đoàn Sơn (JAIST) đã cung cấp tài liệu và cho em những lời khuyên quý báu. Em cũng xin gửi lời cảm ơn tới các thầy cô trong Bộ môn Các hệ thống thông tin, Khoa Công nghệ thông tin đã giúp em có được môi thực nghiệm thuận lợi.

Em cũng xin gửi tới các bạn trong nhóm Seminar “Khai phá dữ liệu và Tính toán song song” lời cảm ơn chân thành vì những đóng góp và những kiến thức quý báu em đã tiếp thu được trong suốt thời gian tham gia nghiên cứu khoa học.

Cuối cùng, em xin cảm ơn gia đình, bạn bè và tập thể lớp K46CA, những người đã luôn ở bên khích lệ và động viên em rất nhiều.

Hà Nội, tháng 6 năm 2005

Sinh viên

Nguyễn Thị Thùy Linh

# MỤC LỤC

|  |            |
|--|------------|
| <b>TÓM TẮT NỘI DUNG .....</b>  | <b>i</b>   |
| <b>LỜI CẢM ƠN .....</b>  | <b>ii</b>  |
| <b>MỤC LỤC .....</b>   | <b>iii</b> |
| <b>DANH MỤC BIỂU ĐỒ HÌNH VẼ.....</b>   | <b>v</b>   |
| <b>DANH MỤC THUẬT NGỮ' .....</b>   | <b>vii</b> |
| <b>ĐẶT VẤN ĐỀ.....</b>   | <b>1</b>   |
| <b>Chương 1. TỔNG QUAN VỀ PHÂN LỚP DỮ LIỆU DỰA TRÊN CÂY QUYẾT ĐỊNH.....</b>                        | <b>3</b>   |
| 1.1. Tổng quan về phân lớp dữ liệu trong data mining.....  | 3          |
| 1.1.1. Phân lớp dữ liệu.....   | 3          |
| 1.1.2. Các vấn đề liên quan đến phân lớp dữ liệu.....  | 6          |
| 1.1.3. Các phương pháp đánh giá độ chính xác của mô hình phân lớp .....                            | 8          |
| 1.2. Cây quyết định ứng dụng trong phân lớp dữ liệu .....  | 9          |
| 1.2.1. Định nghĩa .....  | 9          |
| 1.2.2. Các vấn đề trong khai phá dữ liệu sử dụng cây quyết định.....                               | 10         |
| 1.2.3. Đánh giá cây quyết định trong lĩnh vực khai phá dữ liệu.....                                | 11         |
| 1.2.4. Xây dựng cây quyết định.....  | 13         |
| 1.3. Thuật toán xây dựng cây quyết định.....   | 14         |
| 1.3.1. Tư tưởng chung .....  | 14         |
| 1.3.2. Tình hình nghiên cứu các thuật toán hiện nay .....  | 15         |
| 1.3.3. Song song hóa thuật toán phân lớp dựa trên cây quyết định tuần tự.....                      | 17         |
| <b>Chương 2. C4.5 VÀ SPRINT .....</b>  | <b>21</b>  |
| 2.1. Giới thiệu chung .....  | 21         |
| 2.2. Thuật toán C4.5.....  | 21         |
| 2.2.1. C4.5 dùng Gain-entropy làm độ đo lựa chọn thuộc tính “tốt nhất” .....                       | 22         |
| 2.2.2. C4.5 có cơ chế riêng trong xử lý những giá trị thiếu.....                                   | 25         |
| 2.2.3. Tránh “quá vừa” dữ liệu .....   | 26         |
| 2.2.4. Chuyển đổi từ cây quyết định sang luật .....  | 26         |
| 2.2.5. C4.5 là một thuật toán hiệu quả cho những tập dữ liệu vừa và nhỏ .....                      | 27         |
| 2.3. Thuật toán SPRINT .....   | 28         |
| 2.3.1. Cấu trúc dữ liệu trong SPRINT .....   | 29         |
| 2.3.2. SPRINT sử dụng Gini-index làm độ đo tìm điểm phân chia tập dữ liệu “tốt nhất” .....         | 31         |
| 2.3.3. Thực thi sự phân chia .....   | 34         |
| 2.3.4. SPRINT là thuật toán hiệu quả với những tập dữ liệu quá lớn so với các thuật toán khác..... | 35         |

|   |           |
|---|-----------|
| 2.4. So sánh C4.5 và SPRINT.....                            | 37        |
| <b>Chương 3. CÁC KẾT QUẢ THỰC NGHIỆM.....</b>               | <b>38</b> |
| 3.1. Môi trường thực nghiệm.....                            | 38        |
| 3.2. Cấu trúc mô hình phân lớp C4.5 release8:.....          | 38        |
| 3.2.1. Mô hình phân lớp C4.5 có 4 chương trình chính: ..... | 38        |
| 3.2.2. Cấu trúc dữ liệu sử dụng trong C4.5 .....            | 39        |
| 3.3. Kết quả thực nghiệm.....                               | 40        |
| 3.3.1. Một số kết quả phân lớp tiêu biểu: .....             | 40        |
| 3.3.2. Các biểu đồ hiệu năng .....                          | 47        |
| 3.4. Một số đề xuất cải tiến mô hình phân lớp C4.5 .....    | 54        |
| <b>KẾT LUẬN .....</b>                                       | <b>56</b> |
| <b>TÀI LIỆU THAM KHẢO.....</b>                              | <b>57</b> |

## DANH MỤC BIỂU ĐỒ HÌNH VẼ

|   |    |
|---|----|
| Hình 1 - Quá trình phân lớp dữ liệu - (a) Bước xây dựng mô hình phân lớp .....  | 4  |
| Hình 2 - Quá trình phân lớp dữ liệu - (b1)Ước lượng độ chính xác của mô hình.....   | 5  |
| Hình 3 - Quá trình phân lớp dữ liệu - (b2) Phân lớp dữ liệu mới .....   | 5  |
| Hình 4 - Ước lượng độ chính xác của mô hình phân lớp với phương pháp holdout .....  | 8  |
| Hình 5- Ví dụ về cây quyết định .....   | 9  |
| Hình 6 - Mã giả của thuật toán phân lớp dữ liệu dựa trên cây quyết định.....  | 14 |
| Hình 7 - Sơ đồ xây dựng cây quyết định theo phương pháp đồng bộ.....  | 18 |
| Hình 8 - Sơ đồ xây dựng cây quyết định theo phương pháp phân hoạch .....  | 19 |
| Hình 9 - Sơ đồ xây dựng cây quyết định theo phương pháp lai.....  | 20 |
| Hình 10 - Mã giả thuật toán C4.5 .....  | 22 |
| Hình 11 - Mã giả thuật toán SPRINT .....  | 28 |
| Hình 12 - Cấu trúc dữ liệu trong SLIQ.....  | 29 |
| Hình 13 - Cấu trúc danh sách thuộc tính trong SPRINT – Danh sách thuộc tính liên tục<br>được sắp xếp theo thứ tự ngay được tạo ra .....       | 30 |
| Hình 14 - Ước lượng các điểm phân chia với thuộc tính liên tục .....  | 32 |
| Hình 15 - Ước lượng điểm phân chia với thuộc tính rời rạc .....   | 33 |
| Hình 16 - Phân chia danh sách thuộc tính của một node .....   | 34 |
| Hình 17 - Cấu trúc của bảng băm phân chia dữ liệu trong SPRINT (theo ví dụ các hình<br>trước).....  | 35 |
| Hình 18 - File định nghĩa cấu trúc dữ liệu sử dụng trong thực nghiệm .....  | 39 |
| Hình 19 - File chứa dữ liệu cần phân lớp .....  | 40 |
| Hình 20 - Dạng cây quyết định tạo ra từ tập dữ liệu thử nghiệm.....   | 41 |
| Hình 21 - Ước lượng trên cây quyết định vừa tạo ra trên tập dữ liệu training và tập dữ<br>liệu test .....                                     | 42 |
| Hình 22 - Một số luật rút ra từ bộ dữ liệu 19 thuộc tính, phân lớp loại thiết lập chế độ<br>giao diện của người sử dụng (WEB_SETTING_ID)..... | 43 |
| Hình 23 - Một số luật rút ra từ bộ dữ liệu 8 thuộc tính, phân lớp theo số hiệu nhà sản<br>xuất điện thoại (PRODUCTER_ID) .....                | 44 |
| Hình 24 - Một số luật sinh ra từ tập dữ liệu 8 thuộc tính, phân lớp theo dịch vụ<br>điện thoại mà khách hàng sử dụng (MOBILE_SERVICE_ID)..... | 45 |
| Hình 25 - Ước lượng tập luật trên tập dữ liệu đào tạo .....   | 46 |

|   |    |
|---|----|
| Bảng 1 - Bảng dữ liệu tập training với thuộc tính phân lớp là buys_computer .....   | 24 |
| Bảng 2 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 2 thuộc tính.....  | 49 |
| Bảng 3 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 7 thuộc tính.....  | 50 |
| Bảng 4 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 18 thuộc tính..... | 51 |
| Bảng 5 - Thời gian sinh cây quyết định phụ thuộc vào số lượng thuộc tính .....  | 52 |
| Bảng 6 - Thời gian xây dựng cây quyết định với thuộc tính rời rạc và thuộc tính liên tục .....                                  | 53 |
| Bảng 7 - Thời gian sinh cây quyết định phụ thuộc vào số giá trị phân lớp.....   | 54 |

|  |    |
|--|----|
| Biểu đồ 1- So sánh thời gian thực thi của mô hình phân lớp SPRINT và SLIQ theo kích thước tập dữ liệu đào tạo .....                | 36 |
| Biểu đồ 2 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 2 thuộc tính.....  | 49 |
| Biểu đồ 3 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 7 thuộc tính.....  | 50 |
| Biểu đồ 4 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 18 thuộc tính..... | 51 |
| Biểu đồ 5 - Sự phụ thuộc thời gian sinh cây quyết định vào số lượng thuộc tính.....  | 52 |
| Biểu đồ 6 - So sánh thời gian xây dựng cây quyết định từ tập thuộc tính liên tục và từ tập thuộc tính rời rạc .....                | 53 |
| Biểu đồ 7 - Thời gian sinh cây quyết định phụ thuộc vào số giá trị phân lớp.....   | 54 |



## DANH MỤC THUẬT NGỮ

| STT | Tiếng Anh             | Tiếng Việt  |
|-----|-----------------------|---|
| 1   | training data         | dữ liệu đào tạo   |
| 2   | test data             | dữ liệu kiểm tra  |
| 3   | Pruning decision tree | Cắt, tỉa cây quyết định   |
| 4   | Over fitting data     | Quá vừa dữ liệu   |
| 5   | Noise                 | Dữ liệu lỗi   |
| 6   | Missing value         | Giá trị thiếu   |
| 7   | Data tuple            | Phần tử dữ liệu   |
| 8   | Case                  | Case (được hiểu như một data tuple, chứa một bộ giá trị của các thuộc tính trong tập dữ liệu) |

## ĐẶT VẤN ĐỀ

Trong quá trình hoạt động, con người tạo ra nhiều dữ liệu nghiệp vụ. Các tập dữ liệu được tích lũy có kích thước ngày càng lớn, và có thể chứa nhiều thông tin ẩn dạng những quy luật chưa được khám phá. Chính vì vậy, một nhu cầu đặt ra là cần tìm cách trích rút từ tập dữ liệu đó các luật về phân lớp dữ liệu hay dự đoán những xu hướng dữ liệu tương lai. Những quy tắc nghiệp vụ thông minh được tạo ra sẽ phục vụ đắc lực cho các hoạt động thực tiễn, cũng như phục vụ đắc lực cho quá trình nghiên cứu khoa học. Công nghệ phân lớp và dự đoán dữ liệu ra đời để đáp ứng mong muốn đó.

Công nghệ phân lớp dữ liệu đã, đang và sẽ phát triển mạnh mẽ trước những khao khát tri thức của con người. Trong những năm qua, phân lớp dữ liệu đã thu hút sự quan tâm các nhà nghiên cứu trong nhiều lĩnh vực khác nhau như học máy (*machine learning*), hệ chuyên gia (*expert system*), thống kê (*statistics*)... Công nghệ này cũng ứng dụng trong nhiều lĩnh vực thực tế như: thương mại, nhà băng, maketing, nghiên cứu thị trường, bảo hiểm, y tế, giáo dục...

Nhiều kỹ thuật phân lớp đã được đề xuất như: Phân lớp cây quyết định (Decision tree classification), phân lớp Bayesian (Bayesian classifier), phân lớp K-hàng xóm gần nhất (K-nearest neighbor classifier), mạng nơron, phân tích thống kê,... Trong các kỹ thuật đó, cây quyết định được coi là công cụ mạnh, phổ biến và đặc biệt thích hợp cho data mining [5][7]. Trong các mô hình phân lớp, thuật toán phân lớp là nhân tố chủ đạo. Do vậy cần xây dựng những thuật toán có độ chính xác cao, thực thi nhanh, đi kèm với khả năng mở rộng được để có thể thao tác với những tập dữ liệu ngày càng lớn.

Khóa luận đã nghiên cứu tổng quan về công nghệ phân lớp dữ liệu nói chung và phân lớp dữ liệu dựa trên cây quyết định nói riêng. Từ đó tập trung hai thuật toán tiêu biểu cho hai phạm vi ứng dụng khác nhau là C4.5 và SPRINT. Việc phân tích, đánh giá các thuật toán có giá trị khoa học và ý nghĩa thực tiễn. Tìm hiểu các thuật toán giúp chúng ta tiếp thu và có thể phát triển về mặt tư tưởng, cũng như kỹ thuật của một công nghệ tiên tiến đã và đang là thách thức đối với các nhà khoa học trong lĩnh vực data mining. Từ đó có thể triển khai cài đặt và thử nghiệm các mô hình phân lớp dữ liệu trên thực tế. Tiến tới ứng dụng vào trong các hoạt động thực tiễn tại Việt Nam, mà trước tiên là các hoạt động phân tích, nghiên cứu thị trường khách hàng.

Khóa luận cũng đã chạy thử nghiệm mô hình phân lớp C4.5 trên tập dữ liệu thực tế từ Tổng công ty bưu chính viễn thông. Qua đó tiếp thu được các kỹ thuật triển khai, áp dụng một mô hình phân lớp dữ liệu vào hoạt động thực tiễn. Quá trình chạy thử nghiệm đã thu được các kết quả phân lớp khả quan với độ tin cậy cao và nhiều tiềm năng ứng dụng. Các đánh giá hiệu năng của mô hình phân lớp cũng đã được tiến hành. Trên cơ sở đó, khóa luận đề xuất những cải tiến nhằm tăng hiệu năng của mô hình phân lớp C4.5 đồng thời thêm tiện ích cho người dùng.

Khóa luận gồm có 3 chương chính:

**Chương 1** đi từ tổng quan công nghệ phân lớp dữ liệu tới kỹ thuật phân lớp dữ liệu dựa trên cây quyết định. Các đánh giá về công cụ cây quyết định cũng được trình bày. Chương này cũng cung cấp một cái nhìn tổng quan về lĩnh vực nghiên cứu các thuật toán phân lớp dữ liệu dựa trên cây quyết định với nền tảng tư tưởng, tình hình nghiên cứu và phương hướng phát triển hiện nay.

**Chương 2** tập trung vào hai thuật toán tiêu biểu cho hai phạm vi ứng dụng khác nhau là C4.5 và SPRINT. Hai thuật toán này có những chiến lược riêng trong lựa chọn tiêu chuẩn phân chia dữ liệu cũng như cách thức lưu trữ phân chia dữ liệu...Chính những đặc điểm riêng đó mà C4.5 là thuật toán tiêu biểu phổ biến nhất với tập dữ liệu vừa và nhỏ, trong khi đó SPRINT lại là sự lựa chọn đối với những tập dữ liệu cực lớn.

**Chương 3** trình bày quá trình thực nghiệm với mô hình phân lớp C4.5 trên tập dữ liệu thực tế từ tổng công ty bưu chính viễn thông Việt Nam. Các kết quả thực nghiệm đã được trình bày. Từ đó khóa luận đề xuất các cải tiến mô hình phân lớp C4.5

## Chương 1. TỔNG QUAN VỀ PHÂN LỚP DỮ LIỆU DỰA TRÊN CÂY QUYẾT ĐỊNH

### 1.1. Tổng quan về phân lớp dữ liệu trong data mining

#### 1.1.1. Phân lớp dữ liệu

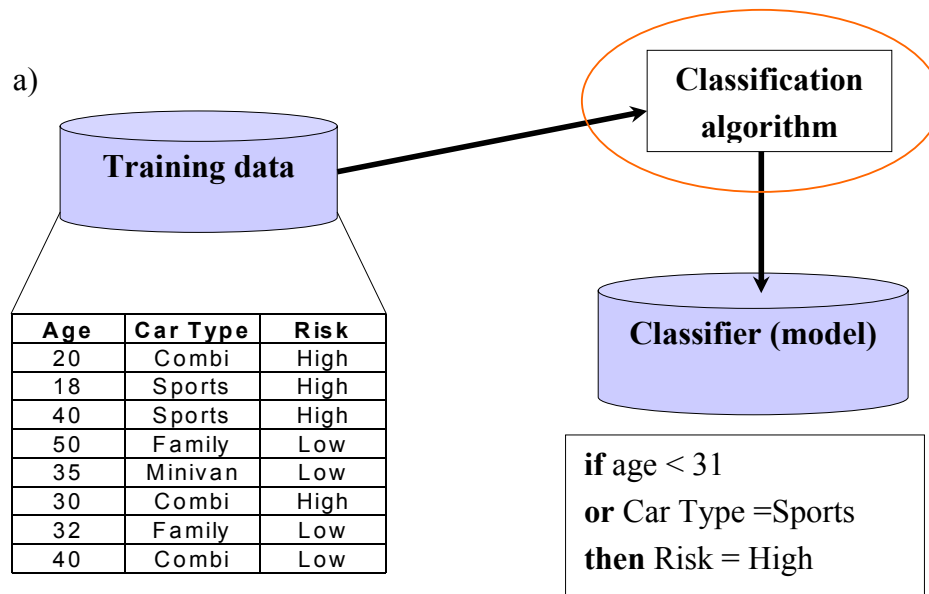
Ngày nay **phân lớp dữ liệu** (*classification*) là một trong những hướng nghiên cứu chính của khai phá dữ liệu. Thực tế đặt ra nhu cầu là từ một cơ sở dữ liệu với nhiều thông tin ẩn con người có thể trích rút ra các quyết định nghiệp vụ thông minh. Phân lớp và dự đoán là hai dạng của phân tích dữ liệu nhằm trích rút ra một mô hình mô tả các lớp dữ liệu quan trọng hay dự đoán xu hướng dữ liệu tương lai. Phân lớp dự đoán giá trị của những nhãn xác định (*categorical label*) hay những giá trị rời rạc (*discrete value*), có nghĩa là phân lớp thao tác với những đối tượng dữ liệu mà có bộ giá trị là biết trước. Trong khi đó, dự đoán lại xây dựng mô hình với các hàm nhận giá trị liên tục. Ví dụ mô hình phân lớp dự báo thời tiết có thể cho biết thời tiết ngày mai là mưa, hay nắng dựa vào những thông số về độ ẩm, sức gió, nhiệt độ,... của ngày hôm nay và các ngày trước đó. Hay nhờ các luật về xu hướng mua hàng của khách hàng trong siêu thị, các nhân viên kinh doanh có thể ra những quyết sách đúng đắn về lượng mặt hàng cũng như chủng loại bày bán... Một mô hình dự đoán có thể dự đoán được lượng tiền tiêu dùng của các khách hàng tiềm năng dựa trên những thông tin về thu nhập và nghề nghiệp của khách hàng. Trong những năm qua, phân lớp dữ liệu đã thu hút sự quan tâm các nhà nghiên cứu trong nhiều lĩnh vực khác nhau như học máy (*machine learning*), hệ chuyên gia (*expert system*), thống kê (*statistics*)... Công nghệ này cũng ứng dụng trong nhiều lĩnh vực khác nhau như: thương mại, nhà băng, marketing, nghiên cứu thị trường, bảo hiểm, y tế, giáo dục... Phần lớn các thuật toán ra đời trước đều sử dụng cơ chế dữ liệu cư trú trong bộ nhớ (*memory resident*), thường thao tác với lượng dữ liệu nhỏ. Một số thuật toán ra đời sau này đã sử dụng kỹ thuật cư trú trên đĩa cải thiện đáng kể khả năng mở rộng của thuật toán với những tập dữ liệu lớn lên tới hàng tỉ bản ghi.

Quá trình phân lớp dữ liệu gồm hai bước [14]:

- **Bước thứ nhất (*learning*)**

Quá trình học nhằm xây dựng một mô hình mô tả một tập các lớp dữ liệu hay các khái niệm định trước. Đầu vào của quá trình này là một tập dữ liệu có cấu trúc được mô tả bằng các thuộc tính và được tạo ra từ tập các bộ giá trị của các thuộc tính

đó. Mỗi bộ giá trị được gọi chung là một phần tử dữ liệu (*data tuple*), có thể là các mẫu (*sample*), ví dụ (*example*), đối tượng (*object*), bản ghi (*record*) hay trường hợp (*case*). Khoa luận sử dụng các thuật ngữ này với nghĩa tương đương. Trong tập dữ liệu này, mỗi phần tử dữ liệu được giả sử thuộc về một lớp định trước, lớp ở đây là giá trị của một thuộc tính được chọn làm *thuộc tính gán nhãn lớp* hay *thuộc tính phân lớp* (*class label attribute*). Đầu ra của bước này thường là các quy tắc phân lớp dưới dạng luật dạng if-then, cây quyết định, công thức logic, hay mạng nơron. Quá trình này được mô tả như trong hình 1

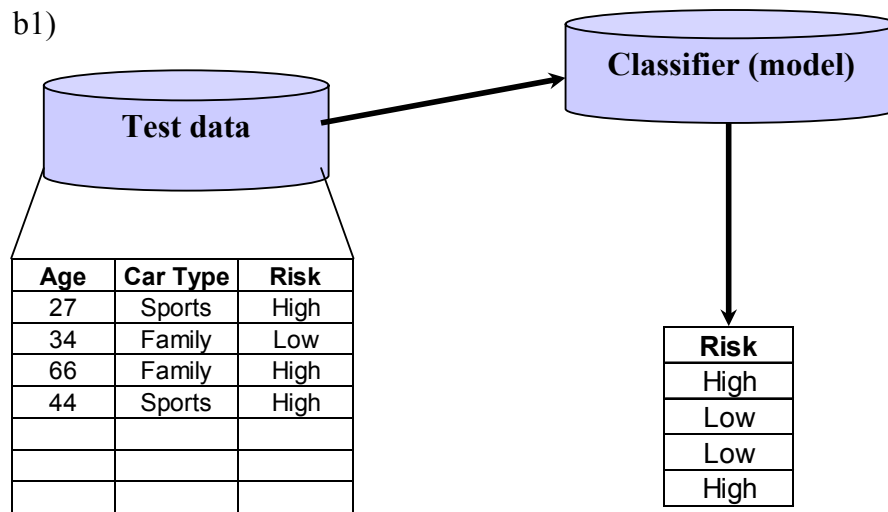


Hình 1 - Quá trình phân lớp dữ liệu - (a) Bước xây dựng mô hình phân lớp

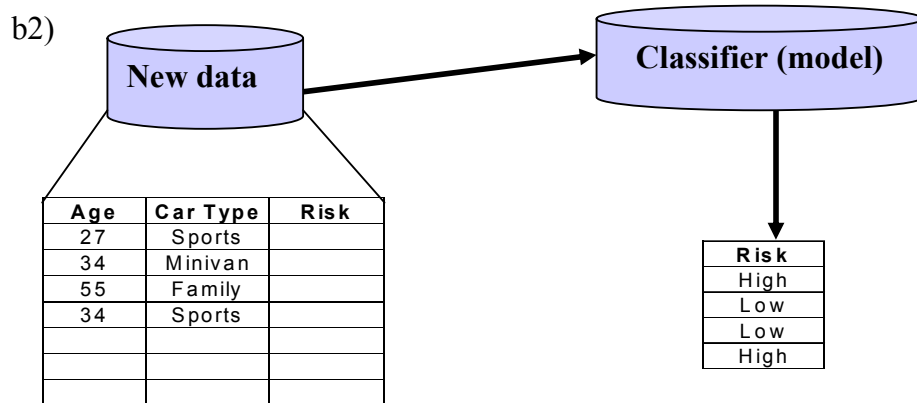
- **Bước thứ hai (classification)**

Bước thứ hai dùng mô hình đã xây dựng ở bước trước để phân lớp dữ liệu mới. Trước tiên độ chính xác mang tính chất dự đoán của mô hình phân lớp vừa tạo ra được ước lượng. **Holdout** là một kỹ thuật đơn giản để ước lượng độ chính xác đó. Kỹ thuật này sử dụng một tập dữ liệu kiểm tra với các mẫu đã được gán nhãn lớp. Các mẫu này được chọn ngẫu nhiên và độc lập với các mẫu trong tập dữ liệu đào tạo. Độ chính xác của mô hình trên *tập dữ liệu kiểm tra* đã đưa là tỉ lệ phần trăm các các mẫu trong tập dữ liệu kiểm tra được mô hình phân lớp đúng (so với thực tế). Nếu độ chính xác của mô hình được ước lượng dựa trên tập dữ liệu đào tạo thì kết quả thu được là rất khả quan vì mô hình luôn có xu hướng “quá vừa” dữ liệu. Quá vừa dữ liệu là hiện tượng kết quả phân lớp trùng khít với dữ liệu thực tế vì quá trình xây dựng mô hình phân lớp từ tập dữ liệu đào tạo có thể đã kết hợp những đặc điểm riêng biệt của tập dữ

liệu đó. Do vậy cần sử dụng một tập dữ liệu kiểm tra độc lập với tập dữ liệu đào tạo. Nếu độ chính xác của mô hình là chấp nhận được, thì mô hình được sử dụng để phân lớp những dữ liệu tương lai, hoặc những dữ liệu mà giá trị của thuộc tính phân lớp là chưa biết.



Hình 2 - Quá trình phân lớp dữ liệu - (b1) Ước lượng độ chính xác của mô hình



Hình 3 - Quá trình phân lớp dữ liệu - (b2) Phân lớp dữ liệu mới

Trong mô hình phân lớp, thuật toán phân lớp giữ vai trò trung tâm, quyết định tới sự thành công của mô hình phân lớp. Do vậy chìa khóa của vấn đề phân lớp dữ liệu là tìm ra được một thuật toán phân lớp nhanh, hiệu quả, có độ chính xác cao và có khả năng mở rộng được. Trong đó khả năng mở rộng được của thuật toán được đặc biệt chú trọng và phát triển [14].

Có thể liệt kê ra đây các kỹ thuật phân lớp đã được sử dụng trong những năm qua:

- Phân lớp cây quyết định (Decision tree classification)

- *Bộ phân lớp Bayesian (Bayesian classifier)*
- *Mô hình phân lớp K-hàng xóm gần nhất (K-nearest neighbor classifier)*
- *Mạng nơron*
- *Phân tích thống kê*
- *Các thuật toán di truyền*
- *Phương pháp tập thô (Rough set Approach)*

### **1.1.2. Các vấn đề liên quan đến phân lớp dữ liệu**

#### **1.1.2.1. Chuẩn bị dữ liệu cho việc phân lớp**

Việc tiền xử lý dữ liệu cho quá trình phân lớp là một việc làm không thể thiếu và có vai trò quan trọng quyết định tới sự áp dụng được hay không của mô hình phân lớp. Quá trình tiền xử lý dữ liệu sẽ giúp cải thiện độ chính xác, tính hiệu quả và khả năng mở rộng được của mô hình phân lớp.

Quá trình tiền xử lý dữ liệu gồm có các công việc sau:

##### ***Làm sạch dữ liệu***

Làm sạch dữ liệu liên quan đến việc xử lý với lỗi (*noise*) và giá trị thiếu (*missing value*) trong tập dữ liệu ban đầu. *Noise* là các lỗi ngẫu nhiên hay các giá trị không hợp lệ của các biến trong tập dữ liệu. Để xử lý với loại lỗi này có thể dùng kỹ thuật làm trơn. *Missing value* là những ô không có giá trị của các thuộc tính. Giá trị thiếu có thể do lỗi chủ quan trong quá trình nhập liệu, hoặc trong trường hợp cụ thể giá trị của thuộc tính đó không có, hay không quan trọng. Kỹ thuật xử lý ở đây có thể bằng cách thay giá trị thiếu bằng giá trị phổ biến nhất của thuộc tính đó hoặc bằng giá trị có thể xảy ra nhất dựa trên thống kê. Mặc dù phần lớn thuật toán phân lớp đều có cơ chế xử lý với những giá trị thiếu và lỗi trong tập dữ liệu, nhưng bước tiền xử lý này có thể làm giảm sự hỗn loạn trong quá trình học (xây dựng mô hình phân lớp).

##### ***Phân tích sự cần thiết của dữ liệu***

Có rất nhiều thuộc tính trong tập dữ liệu có thể hoàn toàn không cần thiết hay liên quan đến một bài toán phân lớp cụ thể. Ví dụ dữ liệu về ngày trong tuần hoàn toàn không cần thiết đối với ứng dụng phân tích độ rủi ro của các khoản tiền cho vay của ngân hàng, nên thuộc tính này là dư thừa. Phân tích sự cần thiết của dữ liệu nhằm mục đích loại bỏ những thuộc tính không cần thiết, dư

thừa khối quá trình học vì những thuộc tính đó sẽ làm chậm, phức tạp và gây ra sự hiểu sai trong quá trình học dẫn tới một mô hình phân lớp không dùng được.

### **Chuyển đổi dữ liệu**

Việc khái quát hóa dữ liệu lên mức khái niệm cao hơn đôi khi là cần thiết trong quá trình tiền xử lý. Việc này đặc biệt hữu ích với những *thuộc tính liên tục* (*continuous attribute* hay *numeric attribute*). Ví dụ các giá trị số của thuộc tính *thu nhập* của khách hàng có thể được khái quát hóa thành các dãy giá trị rời rạc: *thấp, trung bình, cao*. Tương tự với những *thuộc tính rời rạc* (*categorical attribute*) như *địa chỉ phố* có thể được khái quát hóa lên thành *thành phố*. Việc khái quát hóa làm cô đọng dữ liệu học nguyên thủy, vì vậy các thao tác vào/ ra liên quan đến quá trình học sẽ giảm.

#### **1.1.2.2. So sánh các mô hình phân lớp**

Trong từng ứng dụng cụ thể cần lựa chọn mô hình phân lớp phù hợp. Việc lựa chọn đó căn cứ vào sự so sánh các mô hình phân lớp với nhau, dựa trên các tiêu chuẩn sau:

- ***Độ chính xác dự đoán*** (*predictive accuracy*)

Độ chính xác là khả năng của mô hình để dự đoán chính xác nhãn lớp của dữ liệu mới hay dữ liệu chưa biết.

- ***Tốc độ*** (*speed*)

Tốc độ là những chi phí tính toán liên quan đến quá trình tạo ra và sử dụng mô hình.

- ***Sức mạnh*** (*robustness*)

Sức mạnh là khả năng mô hình tạo ra những dự đoán đúng từ những dữ liệu *noise* hay dữ liệu với những giá trị thiếu.

- ***Khả năng mở rộng*** (*scalability*)

Khả năng mở rộng là khả năng thực thi hiệu quả trên lượng lớn dữ liệu của mô hình đã học.

- ***Tính hiểu được*** (*interpretability*)

Tính hiểu được là mức độ hiểu và hiểu rõ những kết quả sinh ra bởi mô hình đã học.



- **Tính đơn giản** (*simplicity*)

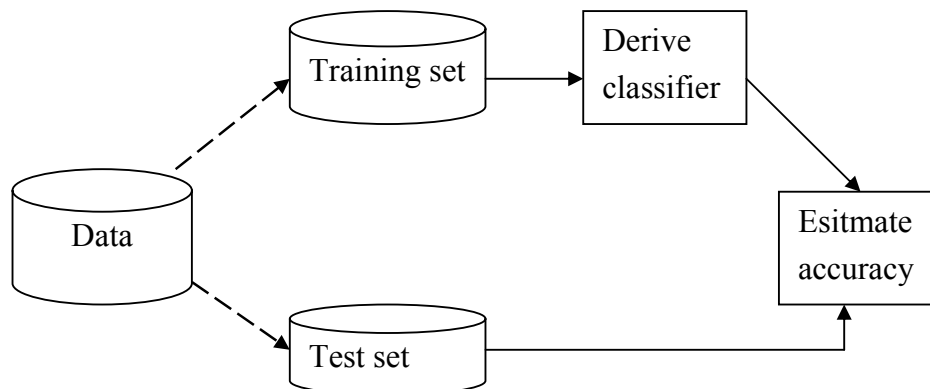
Tính đơn giản liên quan đến kích thước của cây quyết định hay độ cô đọng của các luật.

Trong các tiêu chuẩn trên, khả năng mở rộng của mô hình phân lớp được nhấn mạnh và trú trọng phát triển, đặc biệt với cây quyết định. [14]

### 1.1.3. Các phương pháp đánh giá độ chính xác của mô hình phân lớp

Ước lượng độ chính xác của bộ phân lớp là quan trọng ở chỗ nó cho phép dự đoán được độ chính xác của các kết quả phân lớp những dữ liệu tương lai. Độ chính xác còn giúp so sánh các mô hình phân lớp khác nhau. Khóa luận này đề cập đến 2 phương pháp đánh giá phổ biến là *holdout* và *k-fold cross-validation*. Cả 2 kỹ thuật này đều dựa trên các phân hoạch ngẫu nhiên tập dữ liệu ban đầu.

- Trong phương pháp *holdout*, dữ liệu đưa ra được phân chia ngẫu nhiên thành 2 phần là: tập dữ liệu đào tạo và tập dữ liệu kiểm tra. Thông thường 2/3 dữ liệu cấp cho tập dữ liệu đào tạo, phần còn lại cho tập dữ liệu kiểm tra [14].



Hình 4 - Ước lượng độ chính xác của mô hình phân lớp với phương pháp *holdout*

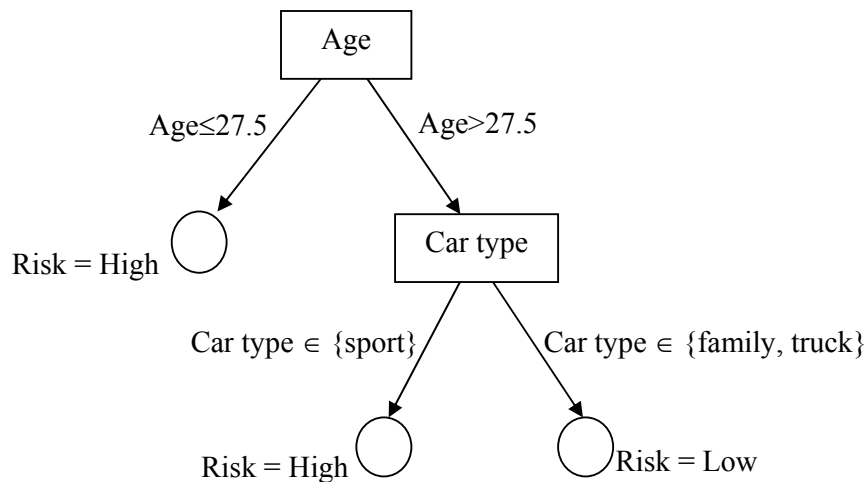
- Trong phương pháp *k-fold cross validation* tập dữ liệu ban đầu được chia ngẫu nhiên thành  $k$  tập con (*fold*) có kích thước xấp xỉ nhau  $S_1, S_2, \dots, S_k$ . Quá trình học và test được thực hiện  $k$  lần. Tại lần lặp thứ  $i$ ,  $S_i$  là tập dữ liệu kiểm tra, các tập còn lại hợp thành tập dữ liệu đào tạo. Có nghĩa là, đầu tiên việc dạy được thực hiện trên các tập  $S_2, S_3, \dots, S_k$ , sau đó test trên tập  $S_1$ ; tiếp tục quá trình dạy được thực hiện trên tập  $S_1, S_3, S_4, \dots, S_k$ , sau đó test trên tập  $S_2$ ; và cứ thế tiếp tục. Độ chính xác là toàn bộ số phân lớp đúng từ  $k$  lần lặp chia cho tổng số mẫu của tập dữ liệu ban đầu.

## 1.2. Cây quyết định ứng dụng trong phân lớp dữ liệu

### 1.2.1. Định nghĩa

Trong những năm qua, nhiều mô hình phân lớp dữ liệu đã được các nhà khoa học trong nhiều lĩnh vực khác nhau đề xuất như mạng neuron, mô hình thông kê tuyến tính / bậc 2, cây quyết định, mô hình di truyền. Trong số những mô hình đó, cây quyết định với những ưu điểm của mình được đánh giá là một công cụ mạnh, phổ biến và đặc biệt thích hợp cho data mining nói chung và phân lớp dữ liệu nói riêng [7]. Có thể kể ra những ưu điểm của cây quyết định như: xây dựng tương đối nhanh; đơn giản, dễ hiểu. Hơn nữa các cây có thể dễ dàng được chuyển đổi sang các câu lệnh SQL để có thể được sử dụng để truy nhập cơ sở dữ liệu một cách hiệu quả. Cuối cùng, việc phân lớp dựa trên cây quyết định đạt được sự tương tự và đôi khi là chính xác hơn so với các phương pháp phân lớp khác [10].

Cây quyết định là biểu đồ phát triển có cấu trúc dạng cây, như mô tả trong hình vẽ sau:



Hình 5- Ví dụ về cây quyết định

Trong cây quyết định:

- Gốc: là node trên cùng của cây
- Node trong: biểu diễn một kiểm tra trên một thuộc tính đơn (hình chữ nhật)
- Nhánh: biểu diễn các kết quả của kiểm tra trên node trong (mũi tên)
- Node lá: biểu diễn lớp hay sự phân phối lớp (hình tròn)

Để phân lớp mẫu dữ liệu chưa biết, giá trị các thuộc tính của mẫu được đưa vào kiểm tra trên cây quyết định. Mỗi mẫu tương ứng có một đường đi từ gốc đến lá và lá biểu diễn dự đoán giá trị phân lớp mẫu đó.

### 1.2.2. Các vấn đề trong khai phá dữ liệu sử dụng cây quyết định

Các vấn đề đặc thù trong khi học hay phân lớp dữ liệu bằng cây quyết định gồm: xác định độ sâu để phát triển cây quyết định, xử lý với những thuộc tính liên tục, chọn phép đo lựa chọn thuộc tính thích hợp, sử dụng tập dữ liệu đào tạo với những giá trị thuộc tính bị thiếu, sử dụng các thuộc tính với những chi phí khác nhau, và cải thiện hiệu năng tính toán. Sau đây khóa luận sẽ đề cập đến những vấn đề chính đã được giải quyết trong các thuật toán phân lớp dựa trên cây quyết định.

#### 1.2.2.1. Tránh “quá vừa” dữ liệu

Thế nào là “quá vừa” dữ liệu? Có thể hiểu đây là hiện tượng cây quyết định chứa một số đặc trưng riêng của tập dữ liệu đào tạo, nếu lấy chính tập training data để test lại mô hình phân lớp thì độ chính xác sẽ rất cao, trong khi đối với những dữ liệu tương lai khác nếu sử dụng cây đó lại không đạt được độ chính xác như vậy.

Quá vừa dữ liệu là một khó khăn đáng kể đối với học bằng cây quyết định và những phương pháp học khác. Đặc biệt khi số lượng ví dụ trong tập dữ liệu đào tạo quá ít, hay có noise trong dữ liệu.

Có hai phương pháp tránh “quá vừa” dữ liệu trong cây quyết định:

- Dừng phát triển cây sớm hơn bình thường, trước khi đạt tới điểm phân lớp hoàn hảo tập dữ liệu đào tạo. Với phương pháp này, một thách thức đặt ra là phải ước lượng chính xác thời điểm dừng phát triển cây.
- Cho phép cây có thể “quá vừa” dữ liệu, sau đó sẽ cắt, tỉa cây.

Mặc dù phương pháp thứ nhất có vẻ trực tiếp hơn, nhưng với phương pháp thứ hai thì cây quyết định được sinh ra được thực nghiệm chứng minh là thành công hơn trong thực tế. Hơn nữa việc cắt tỉa cây quyết định còn giúp tổng quát hóa, và cải thiện độ chính xác của mô hình phân lớp. Dù thực hiện phương pháp nào thì vấn đề mấu chốt ở đây là tiêu chuẩn nào được sử dụng để xác định kích thước hợp lý của cây cuối cùng.

### 1.2.2.2. Thao tác với thuộc tính liên tục

Việc thao tác với thuộc tính liên tục trên cây quyết định hoàn toàn không đơn giản như với thuộc tính rời rạc.

Thuộc tính rời rạc có *tập giá trị (domain)* xác định từ trước và là tập hợp các giá trị rời rạc. Ví dụ *loại ô tô* là một thuộc tính rời rạc với tập giá trị là: {xe tải, xe khách, xe con, taxi}. Việc phân chia dữ liệu dựa vào phép kiểm tra giá trị của thuộc tính rời rạc được chọn tại một ví dụ cụ thể có thuộc tập giá trị của thuộc tính đó hay không:  $value(A) \in X$  với  $X \subset domain(A)$ . Đây là phép kiểm tra logic đơn giản, không tốn nhiều tài nguyên tính toán. Trong khi đó, với thuộc tính liên tục (thuộc tính dạng số) thì tập giá trị là không xác định trước. Chính vì vậy, trong quá trình phát triển cây, cần sử dụng kiểm tra dạng nhị phân:  $value(A) \leq \theta$ . Với  $\theta$  là *hằng số ngưỡng (threshold)* được lần lượt xác định dựa trên từng giá trị riêng biệt hay từng cặp giá trị liên nhau (theo thứ tự đã sắp xếp) của thuộc tính liên tục đang xem xét trong tập dữ liệu đào tạo. Điều đó có nghĩa là nếu thuộc tính liên tục  $A$  trong tập dữ liệu đào tạo có  $d$  giá trị phân biệt thì cần thực hiện  $d-1$  lần kiểm tra  $value(A) \leq \theta_i$  với  $i = 1..d-1$  để tìm ra ngưỡng  $\theta_{best}$  tốt nhất tương ứng với thuộc tính đó. Việc xác định giá trị của  $\theta$  và tiêu chuẩn tìm  $\theta$  tốt nhất tùy vào chiến lược của từng thuật toán [13][1]. Trong thuật toán C4.5,  $\theta_i$  được chọn là giá trị trung bình của hai giá trị liên kề nhau trong dãy giá trị đã sắp xếp.

Ngoài ra còn một số vấn đề liên quan đến sinh tập luật, xử lý với giá trị thiếu sẽ được trình bày cụ thể trong phần thuật toán C4.5.

### 1.2.3. Đánh giá cây quyết định trong lĩnh vực khai phá dữ liệu

#### 1.2.3.1. Sức mạnh của cây quyết định

Cây quyết định có 5 sức mạnh chính sau [5]:

##### ***Khả năng sinh ra các quy tắc hiểu được***

Cây quyết định có khả năng sinh ra các quy tắc có thể chuyển đổi được sang dạng tiếng Anh, hoặc các câu lệnh SQL. Đây là ưu điểm nổi bật của kỹ thuật này. Thậm chí với những tập dữ liệu lớn khiến cho hình dáng cây quyết định lớn và phức tạp, việc đi theo bất cứ đường nào trên cây là dễ dàng theo nghĩa phổ biến và rõ ràng. Do vậy sự giải thích cho bất cứ một sự phân lớp hay dự đoán nào đều tương đối minh bạch.

### ***Khả năng thực thi trong những lĩnh vực hướng quy tắc***

Điều này có nghe có vẻ hiển nhiên, nhưng quy tắc quy nạp nói chung và cây quyết định nói riêng là lựa chọn hoàn hảo cho những lĩnh vực thực sự là các quy tắc. Rất nhiều lĩnh vực từ di truyền tới các quá trình công nghiệp thực sự chứa các quy tắc ẩn, không rõ ràng (*underlying rules*) do khá phức tạp và tối nghĩa bởi những dữ liệu lộn xộn (*noisy*). Cây quyết định là một sự lựa chọn tự nhiên khi chúng ta nghi ngờ sự tồn tại của các quy tắc ẩn, không rõ ràng.

### ***Dễ dàng tính toán trong khi phân lớp***

Mặc dù như chúng ta đã biết, cây quyết định có thể chứa nhiều định dạng, nhưng trong thực tế, các thuật toán sử dụng để tạo ra cây quyết định thường tạo ra những cây với số phân nhánh thấp và các test đơn giản tại từng node. Những test điển hình là: so sánh số, xem xét phần tử của một tập hợp, và các phép nối đơn giản. Khi thực thi trên máy tính, những test này chuyển thành các toán hàm logic và số nguyên là những toán hạng thực thi nhanh và không đắt. Đây là một ưu điểm quan trọng bởi trong môi trường thương mại, các mô hình dự đoán thường được sử dụng để phân lớp hàng triệu thậm chí hàng tỉ bản ghi.

### ***Khả năng xử lý với cả thuộc tính liên tục và thuộc tính rời rạc***

Cây quyết định xử lý “tốt” như nhau với thuộc tính liên tục và thuộc tính rời rạc. Tuy rằng với thuộc tính liên tục cần nhiều tài nguyên tính toán hơn. Những thuộc tính rời rạc đã từng gây ra những vấn đề với mạng neural và các kỹ thuật thống kê lại thực sự dễ dàng thao tác với các *tiêu chuẩn phân chia* (splitting criteria) trên cây quyết định: mỗi nhánh tương ứng với từng phân tách tập dữ liệu theo giá trị của thuộc tính được chọn để phát triển tại node đó. Các thuộc tính liên tục cũng dễ dàng phân chia bằng việc chọn ra một số gọi là ngưỡng trong tập các giá trị đã sắp xếp của thuộc tính đó. Sau khi chọn được ngưỡng tốt nhất, tập dữ liệu phân chia theo test nhị phân của ngưỡng đó.

### ***Thể hiện rõ ràng những thuộc tính tốt nhất***

Các thuật toán xây dựng cây quyết định đưa ra thuộc tính mà phân chia tốt nhất tập dữ liệu đào tạo bắt đầu từ node gốc của cây. Từ đó có thể thấy những thuộc tính nào là quan trọng nhất cho việc dự đoán hay phân lớp.

### 1.2.3.2. Điểm yếu của cây quyết định

Dù có những sức mạnh nổi bật trên, cây quyết định vẫn không tránh khỏi có những điểm yếu. Đó là cây quyết định không thích hợp lắm với những bài toán với mục tiêu là dự đoán giá trị của thuộc tính liên tục như thu nhập, huyết áp hay lãi xuất ngân hàng,... Cây quyết định cũng khó giải quyết với những dữ liệu thời gian liên tục nếu không bỏ ra nhiều công sức cho việc đặt ra sự biểu diễn dữ liệu theo các mẫu liên tục.

#### ***Để xảy ra lỗi khi có quá nhiều lớp***

Một số cây quyết định chỉ thao tác với những lớp giá trị nhị phân dạng *yes/no* hay *accept/reject*. Số khác lại có thể chỉ định các bản ghi vào một số lớp bất kỳ, nhưng dễ xảy ra lỗi khi số ví dụ đào tạo ứng với một lớp là nhỏ. Điều này xảy ra càng nhanh hơn với cây mà có nhiều tầng hay có nhiều nhánh trên một node.

#### ***Chi phí tính toán đắt để đào tạo***

Điều này nghe có vẻ mâu thuẫn với khẳng định ưu điểm của cây quyết định ở trên. Nhưng quá trình phát triển cây quyết định đắt về mặt tính toán. Vì cây quyết định có rất nhiều node trong trước khi đi đến lá cuối cùng. Tại từng node, cần tính một *độ đo* (hay *tiêu chuẩn phân chia*) trên từng thuộc tính, với thuộc tính liên tục phải thêm thao tác sắp xếp lại tập dữ liệu theo thứ tự giá trị của thuộc tính đó. Sau đó mới có thể chọn được một thuộc tính phát triển và tương ứng là một phân chia tốt nhất. Một vài thuật toán sử dụng tổ hợp các thuộc tính kết hợp với nhau có trọng số để phát triển cây quyết định. Quá trình cắt cụt cây cũng “đắt” vì nhiều cây con ứng cử phải được tạo ra và so sánh.

### 1.2.4. Xây dựng cây quyết định

Quá trình xây dựng cây quyết định gồm hai giai đoạn:

- Giai đoạn thứ nhất phát triển cây quyết định:

Giai đoạn này phát triển bắt đầu từ gốc, đến từng nhánh và phát triển quy nạp theo cách thức chia để trị cho tới khi đạt được cây quyết định với tất cả các lá được gán nhãn lớp.

- Giai đoạn thứ hai cắt, tỉa bớt các cành nhánh trên cây quyết định.

Giai đoạn này nhằm mục đích đơn giản hóa và khái quát hóa từ đó làm tăng độ chính xác của cây quyết định bằng cách loại bỏ sự phụ thuộc vào mức độ lỗi (noise)

của dữ liệu đào tạo mang tính chất thống kê, hay những sự biến đổi mà có thể là đặc tính riêng biệt của dữ liệu đào tạo. Giai đoạn này chỉ truy cập dữ liệu trên cây quyết định đã được phát triển trong giai đoạn trước và quá trình thực nghiệm cho thấy giai đoạn này không tốn nhiều tài nguyên tính toán, như với phần lớn các thuật toán, giai đoạn này chiếm khoảng dưới 1% tổng thời gian xây dựng mô hình phân lớp [7][1].

Do vậy, ở đây chúng ta chỉ tập trung vào nghiên cứu giai đoạn phát triển cây quyết định. Dưới đây là khung công việc của giai đoạn này:

- 
- 1) Chọn thuộc tính “tốt” nhất bằng một độ đo đã định trước
  - 2) Phát triển cây bằng việc thêm các nhánh tương ứng với từng giá trị của thuộc tính đã chọn
  - 3) Sắp xếp, phân chia tập dữ liệu đào tạo tới node con
  - 4) Nếu các ví dụ được phân lớp rõ ràng thì dừng.
- Ngược lại: lặp lại bước 1 tới bước 4 cho từng node con
- 

### **1.3. Thuật toán xây dựng cây quyết định**

#### **1.3.1. Tư tưởng chung**

Phần lớn các thuật toán phân lớp dữ liệu dựa trên cây quyết định có mã giả như sau:

```
Make Tree (Training Data T)
{
    Partition(T)
}
Partition(Data S)
{
    if (all points in S are in the same class) then
        return
    for each attribute A do
        evaluate splits on attribute A;
        use best split found to partition S into  $S_1, S_2, \dots, S_k$ 
        Partition( $S_1$ )
        Partition( $S_2$ )
        ...
        Partition( $S_k$ )
}
```

Hình 6 - Mã giả của thuật toán phân lớp dữ liệu dựa trên cây quyết định

Các thuật toán phân lớp như C4.5 (Quinlan, 1993), CDP (Agrawal và các tác giả khác, 1993), SLIQ (Mehta và các tác giả khác, 1996) và SPRINT (Shafer và các tác giả khác, 1996) đều sử dụng phương pháp của **Hunt** làm tư tưởng chủ đạo. Phương pháp này được Hunt và các đồng sự nghĩ ra vào những năm cuối thập kỷ 50 đầu thập kỷ 60.

#### ***Mô tả quy nạp phương pháp Hunt [1]:***

---

Giả sử xây dựng cây quyết định từ  $T$  là tập training data và các lớp được biểu diễn dưới dạng tập  $C = \{C_1, C_2, \dots, C_k\}$

Trường hợp 1:  $T$  chứa các case thuộc về một lớp đơn  $C_j$ , cây quyết định ứng với  $T$  là một lá tương ứng với lớp  $C_j$

Trường hợp 2:  $T$  chứa các case thuộc về nhiều lớp khác nhau trong tập  $C$ . Một kiểm tra được chọn trên một thuộc tính có nhiều giá trị  $\{O_1, O_2, \dots, O_n\}$ . Trong nhiều ứng dụng  $n$  thường được chọn là 2, khi đó tạo ra cây quyết định nhị phân. Tập  $T$  được chia thành các tập con  $T_1, T_2, \dots, T_n$ , với  $T_i$  chứa tất cả các case trong  $T$  mà có kết quả là  $O_i$  trong kiểm tra đã chọn. Cây quyết định ứng với  $T$  bao gồm một node biểu diễn kiểm tra được chọn, và mỗi nhánh tương ứng với mỗi kết quả có thể của kiểm tra đó. Cách thức xây dựng cây tương tự được áp dụng đệ quy cho từng tập con của tập training data.

Trường hợp 3:  $T$  không chứa case nào. Cây quyết định ứng với  $T$  là một lá, nhưng lớp gắn với lá đó phải được xác định từ những thông tin khác ngoài  $T$ . Ví dụ C4.5 chọn giá trị phân lớp là lớp phổ biến nhất tại cha của node này.

---

### **1.3.2. Tình hình nghiên cứu các thuật toán hiện nay**

Các thuật toán phân lớp dữ liệu dựa trên cây quyết định đều có tư tưởng chủ đạo là phương pháp Hunt đã trình bày ở trên. Luôn có 2 câu hỏi lớn cần phải được trả lời trong các thuật toán phân lớp dữ liệu dựa trên cây quyết định là:

1. Làm cách nào để xác định được thuộc tính tốt nhất để phát triển tại mỗi node?
2. Lưu trữ dữ liệu như thế nào và làm cách nào để phân chia dữ liệu theo các test tương ứng?

Các thuật toán khác nhau có các cách trả lời khác nhau cho hai câu hỏi trên. Điều này làm nên sự khác biệt của từng thuật toán.

Có 3 loại tiêu chuẩn hay chỉ số để xác định thuộc tính tốt nhất phát triển tại mỗi node



- Gini-index (Breiman và các đồng sự, 1984 [1]): Loại tiêu chuẩn này lựa chọn thuộc tính mà làm cực tiểu hóa độ không tinh khiết của mỗi phân chia. Các thuật toán sử dụng là CART, SLIQ, SPRINT.
- Information-gain (Quinlan, 1993 [1]): Khác với Gini-index, tiêu chuẩn này sử dụng entropy để đo độ không tinh khiết của một phân chia và lựa chọn thuộc tính theo mức độ cực đại hóa chỉ số entropy. Các thuật toán sử dụng tiêu chuẩn này là ID3, C4.5.
- $\chi^2$ -bảng thống kê các sự kiện xảy ra ngẫu nhiên:  $\chi^2$  đo độ tương quan giữa từng thuộc tính và nhãn lớp. Sau đó lựa chọn thuộc tính có độ tương quan lớn nhất. CHAID là thuật toán sử dụng tiêu chuẩn này.

Chi tiết về cách tính các tiêu chuẩn *Gini-index* và *Information-gain* sẽ được trình bày trong hai thuật toán C4.5 và SPRINT, chương 2.

Việc tính toán các chỉ số trên đôi khi đòi hỏi phải duyệt toàn bộ hay một phần của tập dữ liệu đào tạo. Do vậy các thuật toán ra đời trước yêu cầu toàn bộ tập dữ liệu đào tạo phải nằm *thường trú trong bộ nhớ (memory- resident)* trong quá trình phát triển cây quyết định. Điều này đã làm hạn chế khả năng mở rộng của các thuật toán đó, vì kích thước bộ nhớ là có hạn, mà kích thước của tập dữ liệu đào tạo thì tăng không ngừng, đôi khi là triệu là tỉ bản ghi trong lĩnh vực thương mại. Rõ ràng cần tìm ra giải pháp mới để thay đổi cơ chế lưu trữ và truy cập dữ liệu, năm 1996 SLIQ (Mehta) và SPRINT (Shafer) ra đời đã giải quyết được hạn chế đó. Hai thuật toán này đã sử dụng cơ chế lưu trữ dữ liệu *thường trú trên đĩa (disk- resident)* và cơ chế *sắp xếp trước một lần (pre- sorting)* tập dữ liệu đào tạo. Những đặc điểm mới này làm cải thiện đáng kể hiệu năng và tính mở rộng so với các thuật toán khác. Tiếp theo là một số thuật toán khác phát triển trên nền tảng SPRINT với một số bổ xung cải tiến như PUBLIC (1998) [11] với ý tưởng kết hợp hai quá trình xây dựng và cắt tỉa với nhau, hay ScalParC (1998) cải thiện quá trình phân chia dữ liệu của SPRINT với cách dùng bảng băm khác, hay thuật toán do các nhà khoa học trường đại học Minnesota (Mỹ) kết hợp với IBM đề xuất đã làm giảm chi phí vào ra cũng như chi phí giao tiếp toàn cục khi song song hóa so với SPRINT [2]. Trong các thuật toán đó SPRINT được coi là sáng tạo đột biến, đáng để chúng ta tìm hiểu và phát triển.

### **1.3.3. Song song hóa thuật toán phân lớp dựa trên cây quyết định tuần tự**

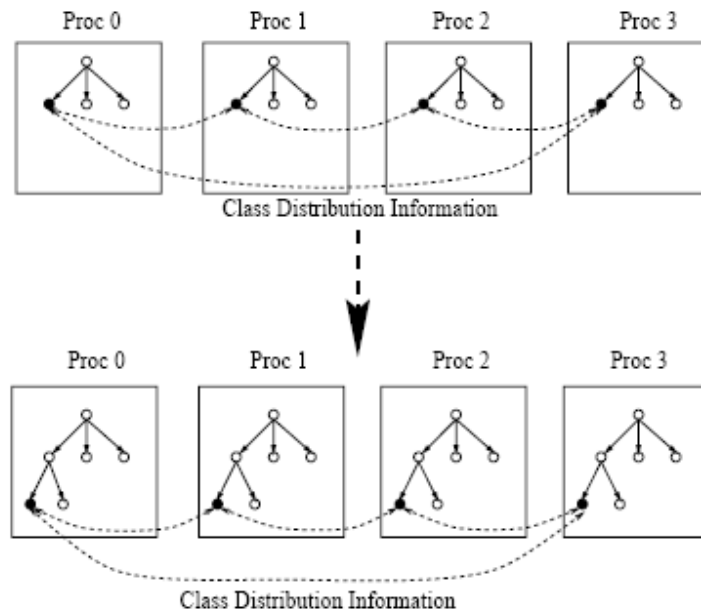
Song song hóa xu hướng nghiên cứu hiện nay của các thuật toán phân lớp dữ liệu dựa trên cây quyết định. Nhu cầu song song hóa các thuật toán tuần tự là một nhu cầu tất yếu của thực tiễn phát triển khi mà các đòi hỏi về hiệu năng, độ chính xác ngày càng cao. Thêm vào đó là sự gia tăng nhanh chóng về kích thước của dữ liệu cần khai phá. Một mô hình phân lớp chạy trên hệ thống tính toán song song có hiệu năng cao, có khả năng khai phá được những tập dữ liệu lớn hơn từ đó gia tăng độ tin cậy của các quy tắc phân lớp. Hiện nay, các thuật toán tuần tự yêu cầu dữ liệu thường trú trong bộ nhớ đã không đáp ứng được yêu cầu của các tập dữ liệu có kích thước TetaByte với hàng tỉ bản ghi. Do vậy xây dựng thuật toán song song hiệu quả dựa trên những thuật toán tuần tự sẵn có là một thách thức đặt ra cho các nhà nghiên cứu.

Có 3 chiến lược song song hóa các thuật toán tuần tự:

#### **Phương pháp xây dựng cây đồng bộ**

Trong phương pháp này, tất cả các bộ vi xử lý đồng thời tham gia xây dựng cây quyết định bằng việc gửi và nhận các thông tin phân lớp của dữ liệu địa phương. Hình 7 mô tả cơ chế làm việc của các bộ vi xử lý trong phương pháp này

Ưu điểm của phương pháp này là không yêu cầu việc di chuyển các dữ liệu trong tập dữ liệu đào tạo. Tuy nhiên, thuật toán này phải chấp nhận chi phí giao tiếp cao, và tải bất cân bằng. Với từng node trong cây quyết định, sau khi tập hợp được các thông tin phân lớp, tất cả các bộ vi xử lý cần phải đồng bộ và cập nhật các thông tin phân lớp. Với những node ở độ sâu thấp, chi phí giao tiếp tương đối nhỏ, bởi vì số lượng các mục training data được xử lý là tương đối nhỏ. Nhưng khi cây càng sâu thì chi phí cho giao tiếp chiếm phần lớn thời gian xử lý. Một vấn đề nữa của phương pháp này là tải bất cân bằng do cơ chế lưu trữ và phân chia dữ liệu ban đầu tới từng bộ vi xử lý.



Hình 7 - Sơ đồ xây dựng cây quyết định theo phương pháp đồng bộ

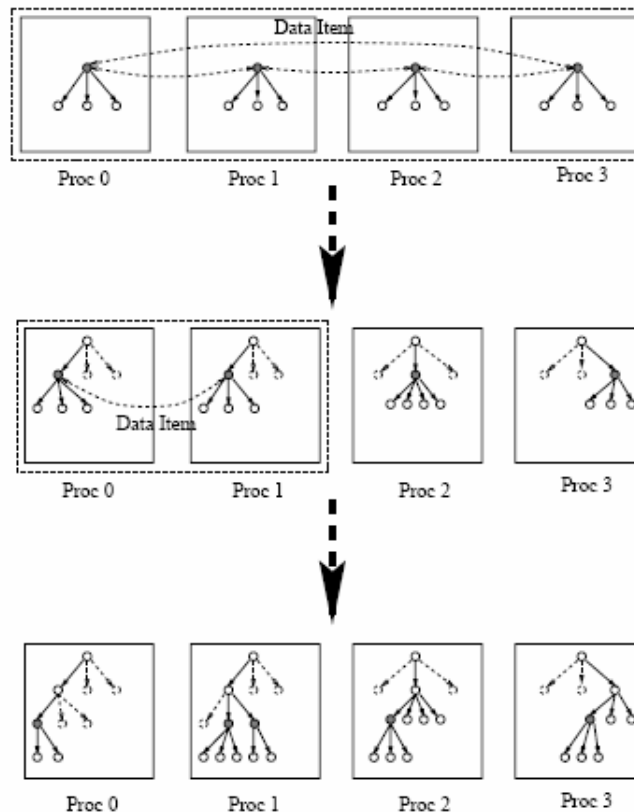
### Phương pháp xây dựng cây phân hoạch

Khi xây dựng cây quyết định bằng phương pháp phân hoạch các bộ vi xử lý khác nhau làm việc với các phần khác nhau của cây quyết định. Nếu nhiều hơn 1 bộ vi xử lý cùng kết hợp để phát triển 1 node, thì các bộ vi xử lý được phân hoạch để phát triển các con của node đó. Phương pháp này tập trung vào trường hợp 1 nhóm các bộ vi xử lý  $P_n$  cùng hợp tác để phát triển node  $n$ . Khi bắt đầu, tất cả các bộ vi xử lý cùng đồng thời kết hợp để phát triển node gốc của cây phân lớp. Khi kết thúc, toàn bộ cây phân lớp được tạo ra bằng cách kết hợp tất cả các cây con của từng bộ vi xử lý. Hình 8 mô tả cơ chế làm việc của các bộ vi xử lý trong phương pháp này.

Ưu điểm của phương pháp này là khi một bộ vi xử lý một mình chịu trách nhiệm phát triển một node, thì nó có thể phát triển thành một cây con của cây toàn cục một cách độc lập mà không cần bất cứ chi phí giao tiếp nào.

Tuy nhiên cũng có một vài nhược điểm trong phương pháp này, đó là: Thứ nhất yêu cầu di chuyển dữ liệu sau mỗi lần phát triển một node cho tới khi mỗi bộ vi xử lý chứa toàn bộ dữ liệu để có thể phát triển toàn bộ một cây con. Do vậy dẫn đến tốn kém chi phí giao tiếp khi ở phần trên của cây phân lớp. Thứ hai là khó đạt được tải cân bằng. Việc gán các node cho các bộ vi xử lý được thực hiện dựa trên số lượng các *case* trong các node con. Tuy nhiên số lượng các *case* gán với một node không nhất

thiết phải tương ứng với số lượng công việc cần phải xử lý để phát triển cây con tại node đó.



Hình 8 - Sơ đồ xây dựng cây quyết định theo phương pháp phân hoạch

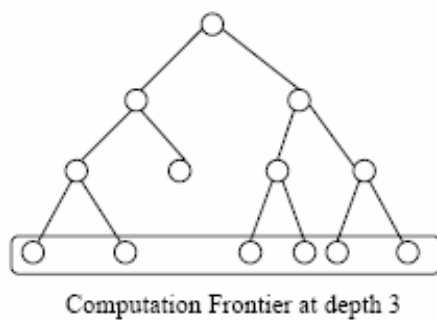
### Phương pháp lai

Phương pháp lai có tận dụng ưu điểm của cả 2 phương pháp trên. Phương pháp xây dựng cây đồng bộ chấp nhận chi phí giao tiếp cao khi biên giới của cây càng rộng. Trong khi đó, phương pháp xây dựng cây quyết định phân hoạch thì phải chấp nhận chi phí cho việc tải cân bằng sau mỗi bước. Trên cơ sở đó, phương pháp lai tiếp tục duy trì cách thức thứ nhất miễn là chi phí giao tiếp phải chịu do tuân theo cách thức thứ nhất không quá lớn. Khi mà chi phí này vượt quá một ngưỡng quy định, thì các bộ vi xử lý đang xử lý các node tại đường biên hiện tại của cây phân lớp được phân chia thành 2 phần (với giả thiết số lượng các bộ vi xử lý là lũy thừa của 2).

Phương pháp này cần sử dụng tiêu chuẩn để khởi tạo sự phân hoạch tập các bộ vi xử lý hiện tại, đó là:

$$\sum (\text{Chi phí giao tiếp}) \geq \text{Chi phí di chuyển} + \text{Tải cân bằng}$$

Mô hình hoạt động của phương pháp lai được mô tả trong hình 9.



Hình 9 - Sơ đồ xây dựng cây quyết định theo phương pháp lai

## Chương 2. C4.5 VÀ SPRINT

### 2.1. Giới thiệu chung

Sau đây là những giới thiệu chung nhất về lịch sử ra đời của hai thuật toán C4.5 và SPRINT.

C4.5 là sự kế thừa của của thuật toán học máy bằng cây quyết định dựa trên nền tảng là kết quả nghiên cứu của HUNT và các cộng sự của ông trong nửa cuối thập kỷ 50 và nửa đầu những năm 60 (Hunt 1962). Phiên bản đầu tiên ra đời là ID3 (Quinlan, 1979)- 1 hệ thống đơn giản ban đầu chứa khoảng 600 dòng lệnh Pascal, và tiếp theo là C4 (Quinlan 1987). Năm 1993, J. Ross Quinlan đã kế thừa các kết quả đó phát triển thành C4.5 với 9000 dòng lệnh C chứa trong một đĩa mềm. Mặc dù đã có phiên bản phát triển từ C4.5 là C5.0 - một hệ thống tạo ra lợi nhuận từ Rule Quest Research, nhưng nhiều tranh luận, nghiên cứu vẫn tập trung vào C4.5 vì mã nguồn của nó là sẵn dùng [13].

Năm 1996, 3 tác giả John Shafer, Rakesh Agrawal, Manish Mehta thuộc *IBM Almaden Research Center* đã đề xuất một thuật toán mới với tên gọi SPRINT (Scalable PaRallelization INduction of decision Trees). SPRINT ra đời đã loại bỏ tất cả các giới hạn về bộ nhớ, thực thi nhanh và có khả năng mở rộng. Thuật toán này được thiết kế để dễ dàng song song hóa, cho phép nhiều bộ vi xử lý cùng làm việc đồng thời để xây dựng một mô hình phân lớp đơn, đồng nhất [7]. Hiện nay SPRINT đã được thương mại hóa, thuật toán này được tích hợp vào trong các công cụ khai phá dữ liệu của IBM.

Trong các thuật toán phân lớp dữ liệu dựa trên cây quyết định, C4.5 và SPRINT là hai thuật toán tiêu biểu cho hai phạm vi ứng dụng khác nhau. C4.5 là thuật toán hiệu quả và được dùng rộng rãi nhất trong các ứng dụng phân lớp với lượng dữ liệu nhỏ cỡ vài trăm nghìn bản ghi. SPRINT một thuật toán tuyệt vời cho những ứng dụng với lượng dữ liệu khổng lồ cỡ vài triệu đến hàng tỉ bản ghi.

### 2.2. Thuật toán C4.5

Với những đặc điểm C4.5 là thuật toán phân lớp dữ liệu dựa trên cây quyết định hiệu quả và phổ biến trong những ứng dụng khai phá cơ sở dữ liệu có kích thước nhỏ. C4.5 sử dụng cơ chế lưu trữ dữ liệu thường trú trong bộ nhớ, chính đặc điểm này làm C4.5 chỉ thích hợp với những cơ sở dữ liệu nhỏ, và cơ chế sắp xếp lại dữ liệu tại

mỗi node trong quá trình phát triển cây quyết định. C4.5 còn chứa một kỹ thuật cho phép biểu diễn lại cây quyết định dưới dạng một danh sách sắp thứ tự các luật **if-then** (một dạng quy tắc phân lớp dễ hiểu). Kỹ thuật này cho phép làm giảm bớt kích thước tập luật và đơn giản hóa các luật mà độ chính xác so với nhánh tương ứng cây quyết định là tương đương.

Tư tưởng phát triển cây quyết định của C4.5 là phương pháp HUNT đã nghiên cứu ở trên. Chiến lược phát triển theo **độ sâu** (*depth-first strategy*) được áp dụng cho C4.5.

Mã giả của thuật toán C4.5:

```
(1)  ComputerClassFrequency(T);  
(2)  if OneClass or FewCases  
      return a leaf;  
      Create a decision node N;  
(3)  ForEach Attribute A  
      ComputeGain(A);  
(4)  N.test=AttributeWithBestGain;  
(5)  if N.test is continuous  
      find Threshold;  
(6)  ForEach T' in the splitting of T  
(7)  if T' is Empty  
      Child of N is a leaf  
      else  
(8)  Child of N=FormTree(T');  
(9)  ComputeErrors of N;  
      return N
```

Hình 10 - Mã giả thuật toán C4.5

Trong báo cáo này, chúng tôi tập trung phân tích những điểm khác biệt của C4.5 so với các thuật toán khác. Đó là cơ chế chọn thuộc tính để kiểm tra tại mỗi node, cơ chế xử lý với những giá trị thiếu, tránh việc “quá vừa” dữ liệu, ước lượng độ chính xác và cơ chế cắt tỉa cây.

### 2.2.1. C4.5 dùng Gain-entropy làm độ đo lựa chọn thuộc tính “tốt nhất”

Phần lớn các hệ thống học máy đều cố gắng để tạo ra 1 cây càng nhỏ càng tốt, vì những cây nhỏ hơn thì dễ hiểu hơn và dễ đạt được độ chính xác dự đoán cao hơn.

Do không thể đảm bảo được sự cực tiểu của cây quyết định, C4.5 dựa vào nghiên cứu tối ưu hóa, và sự lựa chọn cách phân chia mà có *độ đo lựa chọn thuộc tính* đạt giá trị cực đại.

Hai độ đo được sử dụng trong C4.5 là **information gain** và **gain ratio**.  $RF(C_j, S)$  biểu diễn tần xuất (*Relative Frequency*) các *case* trong  $S$  thuộc về lớp  $C_j$ .

$$RF(C_j, S) = |S_j| / |S|$$

Với  $|S_j|$  là kích thước tập các *case* có giá trị phân lớp là  $C_j$ .  $|S|$  là kích thước tập dữ liệu đào tạo.

Chỉ số thông tin cần thiết cho sự phân lớp:  $I(S)$  với  $S$  là tập cần xét sự phân phối lớp được tính bằng:

$$I(S) = - \sum_{j=1}^x RF(C_j, S) \log(RF(C_j, S)).$$

Sau khi  $S$  được phân chia thành các tập con  $S_1, S_2, \dots, S_t$  bởi test  $B$  thì **information gain** được tính bằng:

$$G(S, B) = I(S) - \sum_{i=1}^t \frac{|S_i|}{|S|} I(S_i).$$

Test  $B$  sẽ được chọn nếu có  **$G(S, B)$  đạt giá trị lớn nhất**.

Tuy nhiên có một vấn đề khi sử dụng  $G(S, B)$  ưu tiên test có số lượng lớn kết quả, ví dụ  $G(S, B)$  đạt cực đại với test mà từng  $S_i$  chỉ chứa một *case* đơn. Tiêu chuẩn **gain ratio** giải quyết được vấn đề này bằng việc đưa vào thông tin tiềm năng (*potential information*) của bản thân mỗi phân hoạch

$$P(S, B) = - \sum_{i=1}^t \frac{|S_i|}{|S|} \log \left( \frac{|S_i|}{|S|} \right).$$

Test  $B$  sẽ được chọn nếu có tỉ số giá trị **gain ratio** =  $G(S, B) / P(S, B)$  lớn nhất.

Trong mô hình phân lớp C4.5 release8, có thể dùng một trong hai loại chỉ số *Information Gain* hay *Gain ratio* để xác định thuộc tính tốt nhất. Trong đó *Gain ratio* là lựa chọn mặc định.



## Ví dụ mô tả cách tính information gain

- Với thuộc tính rời rạc

Bảng 1 - Bảng dữ liệu tập training với thuộc tính phân lớp là buys\_computer

| rid | age   | income | student | credit_rating | Class: buys_computer |
|-----|-------|--------|---------|---------------|----------------------|
| 1   | <30   | high   | no      | fair          | no                   |
| 2   | <30   | high   | no      | excellent     | no                   |
| 3   | 30-40 | high   | no      | fair          | yes                  |
| 4   | >40   | medium | no      | fair          | yes                  |
| 5   | >40   | low    | yes     | fair          | yes                  |
| 6   | >40   | low    | yes     | excellent     | no                   |
| 7   | 30-40 | low    | yes     | excellent     | yes                  |
| 8   | <30   | medium | no      | fair          | no                   |
| 9   | <30   | low    | yes     | fair          | yes                  |
| 10  | >40   | medium | yes     | fair          | yes                  |
| 11  | <30   | medium | yes     | excellent     | yes                  |
| 12  | 30-40 | medium | no      | excellent     | yes                  |
| 13  | 30-40 | high   | yes     | fair          | yes                  |
| 14  | >40   | medium | no      | excellent     | no                   |

Trong tập dữ liệu trên:  $s_1$  là tập những bản ghi có giá trị phân lớp là *yes*,  $s_2$  là tập những bản ghi có giá trị phân lớp là *no*. Khi đó:

- $I(S) = I(s_1, s_2) = I(9, 5) = -9/14 \cdot \log_2 9/14 - 5/14 \cdot \log_2 5/14 = 0.940$
- Tính  $G(S, A)$  với  $A$  lần lượt là từng thuộc tính:
  - $A = \text{age}$ . Thuộc tính age đã được rời rạc hóa thành các giá trị <30, 30-40, và >40.
    - Với age = “<30”:  $I(S_1) = I(s_{11}, s_{21}) = -2/5 \log_2 2/5 - 3/5 \log_2 3/5 = 0.971$
    - Với age = “30-40”:  $I(S_2) = I(s_{12}, s_{22}) = 0$
    - Với age = “>40”:  $I(S_3) = I(s_{13}, s_{23}) = 0.971$

$$\square \sum |S_i| / |S| * I(S_i) = 5/14 * I(S_1) + 4/14 * I(S_2) + 5/14 * I(S_3) = 0.694$$

$$\mathbf{Gain(S, age)} = I(s_1, s_2) - \sum |S_i| / |S| * I(S_i) = 0.246$$

Tính tương tự với các thuộc tính khác ta được:

- $A = \text{income}$ :  $\text{Gain}(S, \text{income}) = 0.029$
- $A = \text{student}$ :  $\text{Gain}(S, \text{student}) = 0.151$
- $A = \text{credit\_rating}$ :  $\text{Gain}(S, \text{credit\_rating}) = 0.048$

□ Thuộc tính **age** là thuộc tính có độ đo *Information Gain* lớn nhất. Do vậy **age** được chọn làm thuộc tính phát triển tại node đang xét.

- **Với thuộc tính liên tục**

Xử lý thuộc tính liên tục đòi hỏi nhiều tài nguyên tính toán hơn thuộc tính rời rạc. Gồm các bước sau:

1. Kỹ thuật **Quick sort** được sử dụng để sắp xếp các case trong tập dữ liệu đào tạo theo thứ tự tăng dần hoặc giảm dần các giá trị của thuộc tính liên tục  $V$  đang xét. Được tập giá trị  $V = \{v_1, v_2, \dots, v_m\}$
2. Chia tập dữ liệu thành hai tập con theo ngưỡng  $\theta_i = (v_i + v_{i+1})/2$  nằm giữa hai giá trị liên kế nhau  $v_i$  và  $v_{i+1}$ . Test để phân chia dữ liệu là test nhị phân dạng  $V \leq \theta_i$  hay  $V > \theta_i$ . Thực thi test đó ta được hai tập dữ liệu con:  $V_1 = \{v_1, v_2, \dots, v_i\}$  và  $V_2 = \{v_{i+1}, v_{i+2}, \dots, v_m\}$ .
3. Xét  $(m-1)$  ngưỡng  $\theta_i$  có thể có ứng với  $m$  giá trị của thuộc tính  $V$  bằng cách tính *Information gain* hay *Gain ratio* với từng ngưỡng đó. Ngưỡng có giá trị của *Information gain* hay *Gain ratio* lớn nhất sẽ được chọn làm ngưỡng phân chia của thuộc tính đó.

Việc tìm ngưỡng (theo cách tuyến tính như trên) và sắp xếp tập training theo thuộc tính liên tục đang xem xét đôi khi gây ra thất cổ chai vì tốn nhiều tài nguyên tính toán.

### 2.2.2. C4.5 có cơ chế riêng trong xử lý những giá trị thiếu

Giá trị thiếu của thuộc tính là hiện tượng phổ biến trong dữ liệu, có thể do lỗi khi nhập các bản ghi vào cơ sở dữ liệu, cũng có thể do giá trị thuộc tính đó được đánh giá là không cần thiết đối với *case* cụ thể.

Trong quá trình xây dựng cây từ tập dữ liệu đào tạo  $S$ ,  $B$  là test dựa trên thuộc tính  $A_a$  với các giá trị đầu ra là  $b_1, b_2, \dots, b_t$ . Tập  $S_0$  là tập con các *case* trong  $S$  mà có giá trị thuộc tính  $A_a$  không biết và  $S_i$  biểu diễn các *case* với đầu ra là  $b_i$  trong test  $B$ . Khi đó độ đo *information gain* của test  $B$  giảm vì chúng ta không học được gì từ các *case* trong  $S_0$ .

$$G(S, B) = \frac{|S - S_0|}{|S|} G(S - S_0, B).$$

Tương ứng với  $G(S, B)$ ,  $P(S, B)$  cũng thay đổi,

$$P(S, B) = -\frac{|S_0|}{|S|} \log \left( \frac{|S_0|}{|S|} \right) - \sum_{i=1}^t \frac{|S_i|}{|S|} \log \left( \frac{|S_i|}{|S|} \right).$$

Hai thay đổi này làm giảm giá trị của test liên quan đến thuộc tính có tỉ lệ giá trị thiếu cao.

Nếu test B được chọn, C4.5 không tạo một nhánh riêng trên cây quyết định cho  $S_0$ . Thay vào đó, thuật toán có cơ chế phân chia các case trong  $S_0$  về vác tập con  $S_i$  là tập con mà có giá trị thuộc tính test xác định theo trong số  $|S_i|/|S - S_0|$ .

### 2.2.3. Tránh “quá vừa” dữ liệu

“Quá vừa” dữ liệu là một khó khăn đáng kể đối với học bằng cây quyết định và những phương pháp học khác. Quá vừa dữ liệu là hiện tượng: nếu không có các case xung đột (là những case mà giá trị cho mọi thuộc tính là giống nhau nhưng giá trị của lớp lại khác nhau) thì cây quyết định sẽ phân lớp chính xác toàn bộ các case trong tập dữ liệu đào tạo. Đôi khi dữ liệu đào tạo lại chứa những đặc tính cụ thể, nên khi áp dụng cây quyết định đó cho những tập dữ liệu khác thì độ chính xác không còn cao như trước.

Có một số phương pháp tránh “quá vừa” dữ liệu trong cây quyết định:

- Dừng phát triển cây sớm hơn bình thường, trước khi đạt tới điểm phân lớp hoàn hảo tập dữ liệu đào tạo. Với phương pháp này, một thách thức đặt ra là phải ước lượng chính xác thời điểm dừng phát triển cây.
- Cho phép cây có thể “quá vừa” dữ liệu, sau đó sẽ cắt, tỉa cây

Mặc dù phương pháp thứ nhất có vẻ trực quan hơn, nhưng với phương pháp thứ hai thì cây quyết định được sinh ra được thử nghiệm chứng minh là thành công hơn trong thực tế, vì nó cho phép các tương tác tiềm năng giữa các thuộc tính được khám phá trước khi quyết định xem kết quả nào đáng giữ lại. C4.5 sử dụng kỹ thuật thứ hai để tránh “quá vừa” dữ liệu.

### 2.2.4. Chuyển đổi từ cây quyết định sang luật

Việc chuyển đổi từ cây quyết định sang **luật sản xuất** (*production rules*) dạng if-then tạo ra những quy tắc phân lớp dễ hiểu, dễ áp dụng. Các mô hình phân lớp biểu diễn các khái niệm dưới dạng các luật sản xuất đã được chứng minh là hữu ích trong nhiều lĩnh vực khác nhau, với các đòi hỏi về cả độ chính xác và tính hiệu được của mô hình phân lớp. Dạng output tập luật sản xuất là sự lựa chọn “khôn ngoan”. Tuy nhiên, tài nguyên tính toán dùng cho việc tạo ra tập luật từ tập dữ liệu đào tạo có kích thước lớn và nhiều giá trị sai là vô cùng lớn [12]. Khẳng định này sẽ được chứng minh qua kết quả thực nghiệm trên mô hình phân lớp C4.5

Giai đoạn chuyển đổi từ cây quyết định sang luật bao gồm 4 bước:

- **Cắt tỉa:**

Luật khởi tạo ban đầu là đường đi từ gốc đến lá của cây quyết định. Một cây quyết định có  $l$  lá thì tương ứng tập luật sản xuất sẽ có  $l$  luật khởi tạo. Từng điều kiện trong luật được xem xét và loại bỏ nếu không ảnh hưởng tới độ chính xác của luật đó. Sau đó, các luật đã cắt tỉa được thêm vào tập luật trung gian nếu nó không trùng với những luật đã có.

- **Lựa chọn**

Các luật đã cắt tỉa được nhóm lại theo giá trị phân lớp, tạo nên các tập con chứa các luật theo lớp. Sẽ có  $k$  tập luật con nếu tập training có  $k$  giá trị phân lớp. Từng tập con trên được xem xét để chọn ra một tập con các luật mà tối ưu hóa độ chính xác dự đoán của lớp gắn với tập luật đó.

- **Sắp xếp**

Sắp xếp  $K$  tập luật đã tạo ra từ trên bước theo tần số lỗi. Lớp mặc định được tạo ra bằng cách xác định các case trong tập training không chứa trong các luật hiện tại và chọn lớp phổ biến nhất trong các case đó làm lớp mặc định.

- **Ước lượng, đánh giá:**

Tập luật được đem ước lượng lại trên toàn bộ tập training, nhằm mục đích xác định xem liệu có luật nào làm giảm độ chính xác của sự phân lớp. Nếu có, luật đó bị loại bỏ và quá trình ước lượng được lặp cho đến khi không thể cải tiến thêm.

#### **2.2.5. C4.5 là một thuật toán hiệu quả cho những tập dữ liệu vừa và nhỏ**

C4.5 có cơ chế sinh cây quyết định hiệu quả và chặt chẽ bằng việc sử dụng độ đo lựa chọn thuộc tính tốt nhất là **information-gain**. Các cơ chế xử lý với giá trị lỗi, thiếu và chống “quá vừa” dữ liệu của C4.5 cùng với cơ chế cắt tỉa cây đã tạo nên sức mạnh của C4.5. Thêm vào đó, mô hình phân lớp C4.5 còn có phần chuyển đổi từ cây quyết định sang luật dạng *if-then*, làm tăng độ chính xác và tính dễ hiểu của kết quả phân lớp. Đây là tiện ích rất có ý nghĩa đối với người sử dụng.

### 2.3. Thuật toán SPRINT

Ngày nay dữ liệu cần khai phá có thể có tới hàng triệu bản ghi và khoảng 10 đến 10000 thuộc tính. Hàng Terabyte (100 M bản ghi \* 2000 trường \* 5 bytes) dữ liệu cần được khai phá. Những thuật toán ra đời trước không thể đáp ứng được nhu cầu đó. Trước tình hình đó, SPRINT là sự cải tiến của thuật toán SLIQ (Mehta, 1996) ra đời. Các thuật toán SLIQ và SPRINT đều có những cải tiến để tăng khả năng mở rộng của thuật toán như:

- Khả năng xử lý tốt với những thuộc tính liên tục và thuộc tính rời rạc.
- Cả hai thuật toán này đều sử dụng kỹ thuật **sắp xếp trước một lần** dữ liệu, và **lưu trữ thường trú** trên đĩa (*disk – resident data*) những dữ liệu quá lớn không thể chứa vừa trong bộ nhớ trong. Vì sắp xếp những dữ liệu lưu trữ trên đĩa là đắt [3], nên với cơ chế sắp xếp trước, dữ liệu phục vụ cho quá trình phát triển cây chỉ cần được sắp xếp một lần. Sau mỗi bước phân chia dữ liệu tại từng node, thứ tự của các bản ghi trong từng danh sách được duy trì, không cần phải sắp xếp lại như các thuật toán *CART*, và *C4.5* [13][12]. Từ đó làm giảm tài nguyên tính toán khi sử dụng giải pháp lưu trữ dữ liệu thường trú trên đĩa.
- Cả 2 thuật toán sử dụng những cấu trúc dữ liệu giúp cho việc xây dựng cây quyết định dễ dàng hơn. Tuy nhiên cấu trúc dữ liệu lưu trữ của SLIQ và SPRINT khác nhau, dẫn đến những khả năng mở rộng, và song song hóa khác nhau giữa hai thuật toán này.

Mã giả của thuật toán SPRINT như sau:

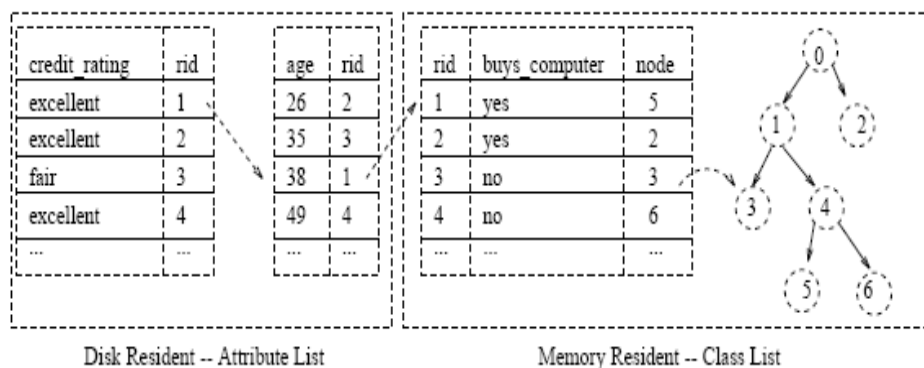
```
SPRINT algorithm:
Partition(Data S) {
    if (all points in S are of the same class) then
        return;
    for each attribute A do
        evaluate splits on attribute A;
    Use best split found to partition S into S1 & S2
    Partition(S1);
    Partition(S2);
}
Initial call: Partition(Training Data)
```

Hình 11 - Mã giả thuật toán SPRINT

### 2.3.1. Cấu trúc dữ liệu trong SPRINT

Kỹ thuật phân chia dữ liệu thành các danh sách thuộc tính riêng biệt lần đầu tiên được SLIQ (Supervised Learning In Quest) đề xuất. Dữ liệu sử dụng trong SLIQ gồm: nhiều danh sách thuộc tính lưu trữ thường trú trên đĩa (mỗi thuộc tính tương ứng với một danh sách), và một danh sách đơn chứa giá trị của class lưu trữ thường trú trong bộ nhớ chính. Các danh sách này liên kết với nhau bởi giá trị của thuộc tính *rid* (chỉ số bản ghi được đánh thứ tự trong cơ sở dữ liệu) có trong mỗi danh sách.

**SLIQ phân chia dữ liệu thành hai loại cấu trúc:[14][9]**



Hình 12 - Cấu trúc dữ liệu trong SLIQ

- Danh sách thuộc tính (*Attribute List*) thường trú trên đĩa. Danh sách này gồm trường thuộc tính và *rid* (*a record identifier*).
- Danh sách lớp (*Class List*) chứa các giá trị của thuộc tính phân lớp tương ứng với từng bản ghi trong cơ sở dữ liệu. Danh sách này gồm các trường *rid*, *thuộc tính phân lớp* và *node* (liên kết với node có giá trị tương ứng trên cây quyết định). Việc tạo ra trường con trỏ trỏ tới node tương ứng trên cây quyết định giúp cho quá trình phân chia dữ liệu chỉ cần thay đổi giá trị của trường con trỏ, mà không cần thực sự phân chia dữ liệu giữa các node. Danh sách lớp được lưu trữ thường trú trong bộ nhớ trong vì nó thường xuyên được truy cập, sửa đổi cả trong giai đoạn xây dựng cây, và cả trong giai đoạn cắt, tỉa cây. Kích thước của danh sách lớp tỉ lệ thuận với số lượng các bản ghi đầu vào. Khi danh sách lớp không vừa trong bộ nhớ, hiệu năng của SLIQ sẽ giảm. Đó là hạn chế của thuật toán SLIQ. Việc sử dụng cấu trúc dữ liệu thường trú trong bộ nhớ làm giới hạn tính mở rộng được của thuật toán SLIQ.

## SPRINT sử dụng danh sách thuộc tính cư trú trên đĩa

SPRINT khắc phục được hạn chế của SLIQ bằng cách không sử dụng danh sách lớp cư trú trong bộ nhớ, SPRINT chỉ sử dụng một loại danh sách là danh sách thuộc tính có cấu trúc như sau:

| RID | Age | Car Type | Risk |
|-----|-----|----------|------|
| 0   | 23  | family   | high |
| 1   | 17  | sport    | high |
| 2   | 43  | sport    | high |
| 3   | 68  | family   | low  |
| 4   | 32  | truck    | low  |
| 5   | 20  | family   | high |

| Car Type | RID | Risk |
|----------|-----|------|
| family   | 0   | high |
| sport    | 1   | high |
| sport    | 2   | high |
| family   | 3   | low  |
| truck    | 4   | low  |
| family   | 5   | high |

| Age | RID | Risk |
|-----|-----|------|
| 17  | 1   | high |
| 20  | 5   | high |
| 23  | 0   | high |
| 32  | 4   | low  |
| 43  | 2   | high |
| 68  | 3   | low  |

Hình 13 - Cấu trúc danh sách thuộc tính trong SPRINT – Danh sách thuộc tính liên tục được sắp xếp theo thứ tự ngay được tạo ra

## Danh sách thuộc tính

SPRINT tạo *danh sách thuộc tính* cho từng thuộc tính trong tập dữ liệu. Danh sách này bao gồm *thuộc tính*, *nhãn lớp* (Class label hay thuộc tính phân lớp), và chỉ số của bản ghi *rid* (được đánh từ tập dữ liệu ban đầu). Danh sách thuộc tính liên tục được sắp xếp thứ tự theo giá trị của thuộc tính đó ngay khi được tạo ra. Nếu toàn bộ dữ liệu không chứa đủ trong bộ nhớ thì tất cả các danh sách thuộc tính được lưu trữ trên đĩa. Chính do đặc điểm lưu trữ này mà SPRINT đã loại bỏ mọi giới hạn về bộ nhớ, và có khả năng ứng dụng với những cơ sở dữ liệu thực tế với số lượng bản ghi có khi lên tới hàng tỉ.

Các danh sách thuộc tính ban đầu tạo ra từ tập dữ liệu đào tạo được gắn với gốc của cây quyết định. Khi cây phát triển, các node được phân chia thành các node con mới thì các danh sách thuộc tính thuộc về node đó cũng được phân chia tương ứng và gắn vào các node con. Khi danh sách bị phân chia thì thứ tự của các bản ghi trong danh sách đó được giữ nguyên, vì thế các danh sách con được tạo ra không bao giờ phải sắp xếp lại. Đó là một ưu điểm của SPRINT so với các thuật toán trước đó.

## Biểu đồ (Histogram)

SPRINT sử dụng biểu đồ để lập bảng thống kê sự phân phối lớp của các bản ghi trong mỗi danh sách thuộc tính, từ đó dùng vào việc ước lượng điểm phân chia cho danh sách đó. Thuộc tính liên tục và thuộc tính rời rạc có hai dạng biểu đồ khác nhau.

- **Biểu đồ của thuộc tính liên tục**

SPRINT sử dụng 2 biểu đồ:  $C_{below}$  và  $C_{above}$ .  $C_{below}$  chứa sự phân phối của những bản ghi đã được xử lý,  $C_{above}$  chứa sự phân phối của những bản ghi chưa được xử lý trong danh sách thuộc tính. Hình II-3 minh họa việc sử dụng biểu đồ cho thuộc tính liên tục

- **Biểu đồ của thuộc tính rời rạc**

Thuộc tính rời rạc cũng có một biểu đồ gắn với từng node. Tuy nhiên SPRINT chỉ sử dụng một biểu đồ là *count matrix* chứa sự phân phối lớp ứng với từng giá trị của thuộc tính được xem xét.

Các danh sách thuộc tính được xử lý cùng một lúc, do vậy thay vì đòi hỏi các danh sách thuộc tính trong bộ nhớ, với SPRINT bộ nhớ chỉ cần chứa tập các biểu đồ như trên trong quá trình phát triển cây.

### 2.3.2. SPRINT sử dụng Gini-index làm độ đo tìm điểm phân chia tập dữ liệu “tốt nhất”

SPRINT là một trong những thuật toán sử dụng độ đo *Gini-index* để tìm thuộc tính tốt nhất làm thuộc tính test tại mỗi node trên cây. Chỉ số này được Breiman nghĩ ra từ năm 1984, cách tính như sau:

- Trước tiên cần định nghĩa:  $\text{gini}(S) = 1 - \sum p_j^2$

Trong đó:  $S$  là tập dữ liệu đào tạo có  $n$  lớp;  $p_j$  là tần xuất của lớp  $j$  trong  $S$  (là thương của số bản ghi có giá trị của thuộc tính phân lớp là  $p_j$  với tổng số bản ghi trong  $S$ )

- Nếu phân chia dạng nhị phân, tức là  $S$  được chia thành  $S_1, S_2$  (SPRINT chỉ sử dụng phân chia nhị phân này) thì chỉ số tính độ phân chia được cho bởi công thức sau:

$$\text{gini}_{\text{split}}(S) = n_1/n * \text{gini}(S_1) + n_2/n * \text{gini}(S_2)$$

Với  $n, n_1, n_2$  lần lượt là kích thước của  $S, S_1, S_2$ .



Ưu điểm của loại chỉ số này là các tính toán trên nó chỉ dựa vào thông tin về sự phân phối các giá trị lớp trong từng phần phân chia mà không tính toán trên các giá trị của thuộc tính đang xem xét.

Để tìm được điểm phân chia cho mỗi node, cần quét từng danh sách thuộc tính của node đó và ước lượng các phân chia dựa trên mỗi thuộc tính gắn với node đó. Thuộc tính được chọn để phân chia là thuộc tính có chỉ số  **$gini_{split}(S)$  nhỏ nhất**.

Điểm cần nhấn mạnh ở đây là khác với *Information Gain* chỉ số này được tính mà không cần đọc nội dung dữ liệu, chỉ cần biểu đồ biểu diễn sự phân phối các bản ghi theo các giá trị phân lớp. Đó là tiền đề cho cơ chế lưu trữ dữ liệu thường trú trên đĩa. Các biểu đồ của danh sách thuộc tính liên tục, hay rời rạc được mô tả dưới đây.

### Với thuộc tính liên tục

Với thuộc tính liên tục, các giá trị kiểm tra là các giá trị nằm giữa mọi cặp 2 giá trị liên kề của thuộc tính đó. Để tìm điểm phân chia cho thuộc tính đó tại một node nhất định, biểu đồ được khởi tạo với  $C_{below}$  bằng 0 và  $C_{above}$  là phân phối lớp của tất cả các bản ghi tại node đó. Hai biểu đồ trên được cập nhật lần lượt mỗi khi từng bản ghi được đọc. Mỗi khi con trỏ chạy *gini-index* được tính trên từng điểm phân chia nằm giữa giá trị vừa đọc và giá trị sắp đọc. Khi đọc hết danh sách thuộc tính ( $C_{above}$  bằng 0 ở tất cả các cột) thì cũng là lúc tính được toàn bộ các *gini-index* của các điểm phân chia cần xem xét. Căn cứ vào kết quả đó có thể chọn ra *gini-index* thấp nhất và tương ứng là điểm phân chia của thuộc tính liên tục đang xem xét tại node đó. Việc tính *gini-index* hoàn toàn dựa vào biểu đồ. Nếu tìm ra điểm phân chia tốt nhất thì kết quả đó được lưu lại và biểu đồ vừa gắn danh sách thuộc tính đó được khởi tạo lại trước khi xử lý với thuộc tính tiếp theo.

| Attribute List |       |     | Position of cursor in scan |  | State of Class Histograms |             |             |
|----------------|-------|-----|----------------------------|--|---------------------------|-------------|-------------|
| Age            | Class | tid |                            |  | cursor position 0:        | $C_{below}$ | $C_{above}$ |
| 17             | High  | 1   | ← position 0               |  |                           | 0           | 0           |
| 20             | High  | 5   |                            |  |                           | 4           | 2           |
| 23             | High  | 0   |                            |  | cursor position 3:        | 3           | 0           |
| 32             | Low   | 4   | ← position 3               |  |                           | 1           | 2           |
| 43             | High  | 2   |                            |  | cursor position 6:        | 4           | 2           |
| 68             | Low   | 3   | ← position 6               |  |                           | 0           | 0           |

Hình 14 - Ước lượng các điểm phân chia với thuộc tính liên tục

## Với thuộc tính rời rạc

Với thuộc tính rời rạc, quá trình tìm điểm phân chia tốt nhất cũng được tính toán dựa trên biểu đồ của danh sách thuộc tính đó. Trước tiên cần quét toàn bộ danh sách thuộc tính để thu được số lượng phân lớp ứng với từng giá trị của thuộc tính rời rạc, kết quả này được lưu trong biểu đồ *count matrix*. Sau đó, cần tìm tất cả các tập con có thể có từ các giá trị của thuộc tính đang xét, coi đó là điểm phân chia và tính *gini-index* tương ứng. Các thông tin cần cho việc tính toán chỉ số *gini-index* của bất cứ tập con nào đều có trong *count matrix*. Bộ nhớ cung cấp cho *count matrix* được thu hồi sau khi tìm ra được điểm phân chia tốt nhất của thuộc tính đó.

| Attribute List |       |     |  |  |
|----------------|-------|-----|--|--|
| Car Type       | Class | tid |  |  |
| family         | High  | 0   |  |  |
| sports         | High  | 1   |  |  |
| sports         | High  | 2   |  |  |
| family         | Low   | 3   |  |  |
| truck          | Low   | 4   |  |  |
| family         | High  | 5   |  |  |

⇒

| Count Matrix |   |   |
|--------------|---|---|
|              | H | L |
| family       | 2 | 1 |
| sports       | 2 | 0 |
| truck        | 0 | 1 |

Hình 15 - Ước lượng điểm phân chia với thuộc tính rời rạc

## Ví dụ mô tả cách tính chỉ số Gini-index

Với tập dữ liệu đào tạo được mô tả trong hình 13, việc tính chỉ số Gini-index để tìm ra điểm phân chia tốt nhất được thực hiện như sau:

- Với Thuộc tính liên tục **Age** cần tính điểm phân chia trên lần lượt các so sánh sau Age≤17, Age≤20, Age≤23, Age≤32, Age≤43, Age≤68

| Tuple count | High | Low |
|-------------|------|-----|
| Age≤17      | 1    | 0   |
| Age>17      | 3    | 2   |

$$G(\text{Age} \leq 17) = 1 - (1^2 + 0^2) = 0$$

$$G(\text{Age} > 17) = 1 - ((3/5)^2 + (2/5)^2) = 1 - (13/25) = 12/25$$

$$G_{\text{SPLIT}} = (1/6) * 0 + (5/6) * (12/25) = 2/5$$

| Tuple count | High | Low |
|-------------|------|-----|
| Age≤20      | 2    | 0   |
| Age>20      | 1    | 2   |

$$G(\text{Age} \leq 20) = 1 - (1^2 + 0^2) = 0$$

$$G(\text{Age} > 20) = 1 - ((1/2)^2 + (1/2)^2) = 1/2$$

$$G_{\text{SPLIT}} = (2/3) * 0 + (1/3) * (1/2) = 1/6$$

| Tuple count | High | Low |
|-------------|------|-----|
| Age≤23      | 3    | 0   |
| Age>23      | 1    | 2   |

$$G(\text{Age} \leq 23) = 1 - (1^2 + 0^2) = 0$$

$$G(\text{Age} > 23) = 1 - ((1/3)^2 + (2/3)^2) = 1 - (1/9) - (4/9) = 4/9$$

$$G_{\text{SPLIT}} = (3/6) * 0 + (3/6) * (4/9) = 2/9$$

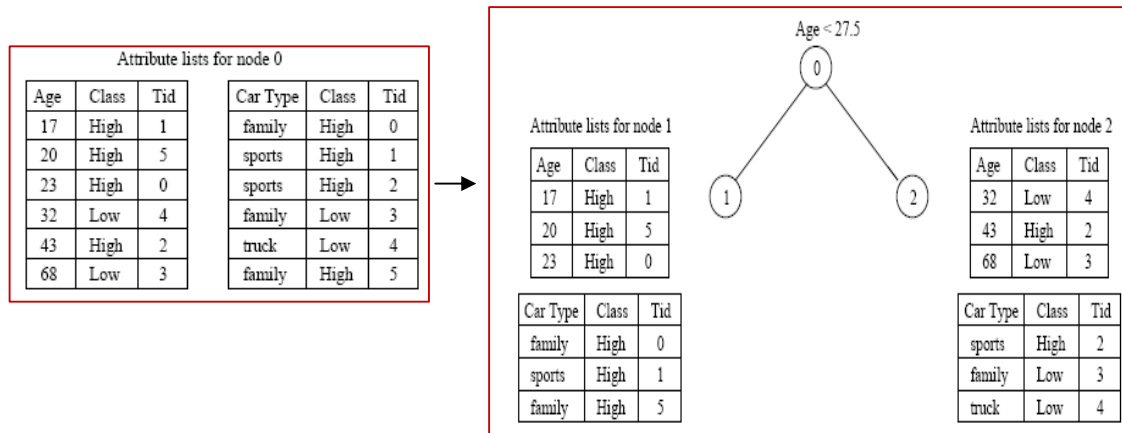
Tính toán tương tự với các test còn lại Age≤32, Age≤43, Age≤68

So sánh các  $G_{\text{SPLIT}}$  tìm được ứng với từng phân chia của các thuộc tính,  $G_{\text{SPLIT}}$  ứng với Age ≤23 có **giá trị nhỏ nhất**. Do vậy điểm phân chia là tại thuộc tính Age với giá trị phân chia =  $(23+32) / 2 = 27.5$ .

Kết quả ta có cây quyết định như hình 5 phần 1.2.1

### 2.3.3. Thực thi sự phân chia

Sau khi tìm ra điểm phân chia tốt nhất của node dựa trên so sánh *gini-index* của các thuộc tính có trên node đó, cần thực thi sự phân chia bằng cách tạo ra các node con và phân chia danh sách thuộc tính của node cha cho các node con.



Hình 16 - Phân chia danh sách thuộc tính của một node

Với thuộc tính được chọn (*Age* như trên hình vẽ) làm thuộc tính phân chia tại node đó, việc phân chia danh sách thuộc tính này về các node con khá đơn giản. Nếu đó là thuộc tính liên tục, chỉ cần cắt danh sách thuộc tính theo điểm phân chia thành 2 phần và gán cho 2 node con tương ứng. Nếu đó là thuộc tính rời rạc thì cần quét toàn bộ danh sách và áp dụng test đã xác định để chuyển các bản ghi về 2 danh sách mới ứng với 2 node con.

Nhưng vấn đề không đơn giản như vậy với những thuộc tính còn lại tại node đó (*Car Type* chẳng hạn), không có test trên thuộc tính này, nên không thể áp dụng các kiểm tra trên giá trị của thuộc tính để phân chia các bản ghi. Lúc này cần dùng đến một trường đặc biệt trong các danh sách thuộc tính đó là ***rids***. Đây chính là trường kết nối các bản ghi trong các danh sách thuộc tính. Cụ thể như sau: trong khi phân chia danh sách của thuộc tính phân chia (*Age*) cần chèn giá trị trường *rids* của mỗi bản ghi vào một **bảng băm** (*hash table*) để đánh dấu node con mà các bản ghi tương ứng (có cùng *rids*) trong các danh sách thuộc tính khác được phân chia tới. Cấu trúc của bảng băm như sau:

|            |            |   |   |   |   |   |   |
|------------|------------|---|---|---|---|---|---|
| Hash table | Rids       | 1 | 2 | 3 | 4 | 5 | 6 |
|            | Child node | L | R | R | R | L | L |

Hình 17 - Cấu trúc của bảng băm phân chia dữ liệu trong SPRINT (theo ví dụ các hình trước)

Phân chia xong danh sách của thuộc tính phân chia thì cũng là lúc xây dựng xong bảng băm. Danh sách các thuộc tính còn lại được phân chia tới các node con theo thông tin trên bảng băm bằng cách đọc trường *rids* trên từng bản ghi và trường *Child node* tương ứng trên bảng băm.

Nếu bảng băm quá lớn so với bộ nhớ, quá trình phân chia được chia thành nhiều bước. Bảng băm được tách thành nhiều phần sao cho vừa với bộ nhớ, và các danh sách thuộc tính phân chia theo từng phần bảng băm. Quá trình lặp lại cho đến khi bảng băm nằm trong bộ nhớ.

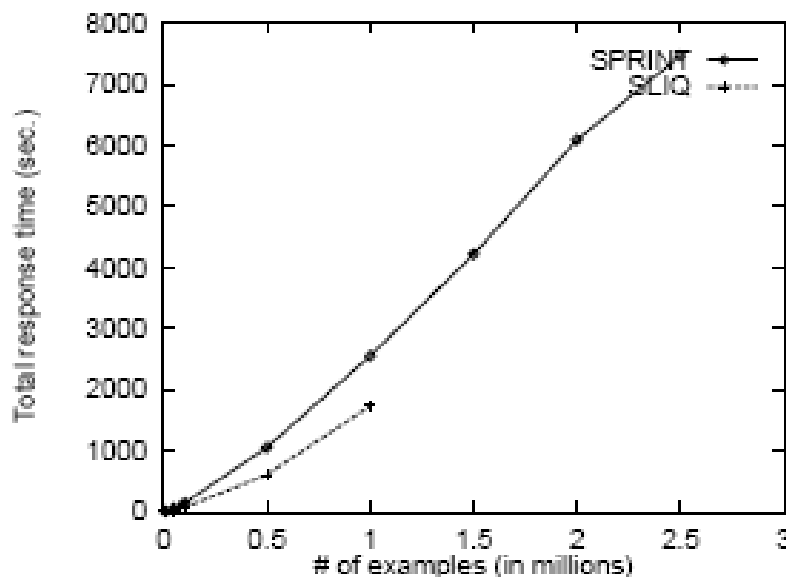
#### 2.3.4. SPRINT là thuật toán hiệu quả với những tập dữ liệu quá lớn so với các thuật toán khác

SPRINT ra đời không nhằm mục đích làm tốt hơn SLIQ [9] với những tập dữ liệu mà *danh sách lớp* nằm vừa trong bộ nhớ. Mục tiêu của thuật toán này là nhằm vào những tập dữ liệu quá lớn so với các thuật toán khác và có khả năng tạo ra một mô

hình phân lớp hiệu quả từ đó. Hơn nữa, SPRINT còn được thiết kế để dễ dàng song song hóa. Quả vậy, việc song song hóa SPRINT khá tự nhiên và hiệu quả với cơ chế xử lý dữ liệu song song. SPRINT đạt được chuẩn cho việc sắp xếp dữ liệu và tải cân bằng khối lượng công việc bằng cách phân phối đều danh sách thuộc tính thuộc tính cho  $N$  bộ vi xử lý của một máy theo kiến trúc *shared-nothing* [7]. Việc song song hóa SPRINT nói riêng cũng như song song hóa các mô hình phân lớp dữ liệu dựa trên cây quyết định nói chung trên hệ thống *Shared-memory multiprocessor (SMPs)* hay còn được gọi là hệ thống *shared-everything* được nghiên cứu trong [10].

Bên cạnh những mặt mạnh, SPRINT cũng có những mặt yếu. Trước hết đó là **bảng băm** sử dụng cho việc phân chia dữ liệu, có kích cỡ tỉ lệ thuận với số lượng đối tượng dữ liệu gắn với node hiện tại (số bản ghi của một danh sách thuộc tính). Đồng thời bảng băm cần được đặt trong bộ nhớ khi thi hành phân chia dữ liệu, khi kích cỡ bảng băm quá lớn, việc phân chia dữ liệu phải tách thành nhiều bước. Mặt khác, thuật toán này phải chịu chi phí vào-ra “trầm trọng”. Việc song song hóa thuật toán này cũng đòi hỏi chi phí giao tiếp toàn cục cao do cần đồng bộ hóa các thông tin về các chỉ số *Gini-index* của từng danh sách thuộc tính.

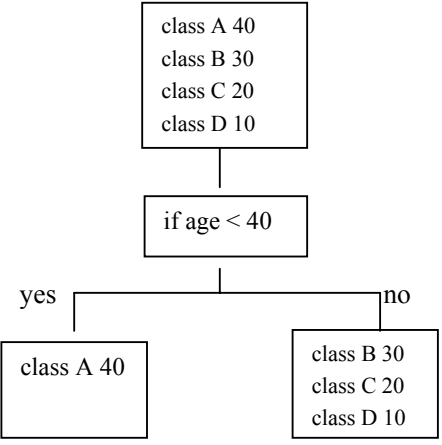
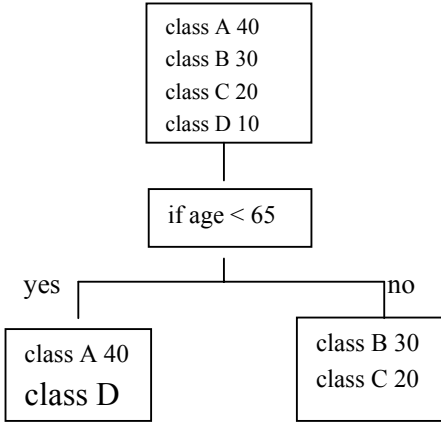
Ba tác giả của SPRINT đã đưa ra một số kết quả thực nghiệm trên mô hình phân lớp SPRINT so sánh với SLIQ [7] được thể hiện bằng biểu đồ dưới đây.



Biểu đồ 1- So sánh thời gian thực thi của mô hình phân lớp SPRINT và SLIQ theo kích thước tập dữ liệu đào tạo

Từ biểu đồ trên có thể thấy: với những tập dữ liệu nhỏ (<1 triệu cases) thì thời gian thực thi của SPRINT lớn hơn so với thời gian thực thi SLIQ. Điều này có thể giải thích do khi đó danh sách lớp khi dùng thuật toán SLIQ vẫn nằm vừa trong bộ nhớ. Nhưng với những tập dữ liệu lớn (>1 triệu cases) thì SLIQ không thể thao tác, trong khi với những tập dữ liệu khoảng hơn 2,5 triệu cases SPRINT vẫn thao tác dễ dàng. Lý do là SPRINT sử dụng cơ chế lưu trữ liệu thường trú hoàn toàn trên đĩa.

## 2.4. So sánh C4.5 và SPRINT

| Nội dung so sánh                                | C4.5   | SPRINT  |
|---|--|---|
| <b>Tiêu chuẩn lựa chọn thuộc tính phân chia</b> | <b>Gain-entropy</b><br>Có khuynh hướng làm cô lập lớp lớn nhất khỏi các lớp khác<br> | <b>Gini-index</b><br>Có khuynh hướng chia thành các nhóm lớp với lượng dữ liệu tương đương<br> |
| <b>Cơ chế lưu trữ dữ liệu</b>                   | Lưu trữ trong bộ nhớ (memory-resident)<br>-> Áp dụng cho những ứng dụng khai phá cơ sở dữ liệu nhỏ (hàng trăm nghìn bản ghi)   | Lưu trữ trên đĩa (disk-resident)<br>-> Áp dụng cho những ứng dụng khai phá dữ liệu cực lớn mà các thuật toán khác không làm được (hàng trăm triệu - hàng tỉ bản ghi)              |
| <b>Cơ chế sắp xếp dữ liệu</b>                   | Sắp xếp lại tập dữ liệu tương ứng với mỗi node   | Sắp xếp trước một lần. Trong quá trình phát triển cây, danh sách thuộc tính được phân chia nhưng thứ tự ban đầu vẫn được duy trì, do đó không cần phải sắp xếp lại.               |

## Chương 3. CÁC KẾT QUẢ THỰC NGHIỆM

Tác giả sử dụng mô hình phân lớp C4.5 release8 mã nguồn mở do J. Ross Quinlan viết, tại địa chỉ: <http://www.cse.unsw.edu.au/~quinlan/> để phân tích, đánh giá mô hình phân lớp C4.5 về kết quả phân lớp và các nhân tố ảnh hưởng đến hiệu năng của mô hình.

### 3.1. Môi trường thực nghiệm

Mã nguồn C.45 được cài đặt và chạy thử nghiệm trên Server 10.10.0.10 của Đại học Công Nghệ.

**Cấu hình của Server như sau:** bộ vi xử lý Intel ® Xeon™ 2.4GHz, có 2 bộ xử lý vật lý có thể hoạt động như 4 bộ xử lý logic theo công nghệ hyper-threading, cache size: 512KB, dung lượng bộ nhớ trong 1GB.

**Tập dữ liệu thử nghiệm** là tập dữ liệu chứa các thông tin về khách hàng sử dụng điện thoại di động đã đăng ký sử dụng *web portal*. Các trường trong tập dữ liệu gồm có: Các thông tin cá nhân như: Tên tuổi, giới tính, ngày sinh, vùng đăng ký sử dụng điện thoại, loại điện thoại sử dụng, version của loại điện thoại đó, số lần và thời gian truy cập web portal để sử dụng các dịch vụ như gửi tin nhắn, gửi logo hay ringtone... Tập dữ liệu có kích thước khoảng 120000 bản ghi dùng để training và khoảng 60000 bản ghi được sử dụng để làm tập dữ liệu test.

### 3.2. Cấu trúc mô hình phân lớp C4.5 release8:

#### 3.2.1. Mô hình phân lớp C4.5 có 4 chương trình chính:

- Chương trình sinh cây quyết định (**c4.5**)
- Chương trình sinh luật sản xuất (**c4.5rules**)
- Chương trình ứng dụng cây quyết định vào phân lớp những dữ liệu mới (**consult**)
- Chương trình ứng dụng bộ luật sản xuất vào phân lớp những dữ liệu mới (**consultr**)

Ngoài ra C4.5 còn có 2 tiện ích đi kèm phục vụ cho quá trình chạy thực nghiệm là:

- **csh shell script** cho kỹ thuật ước lượng độ chính xác của mô hình phân lớp *cross-validation* (**xval.sh**)
- Hai chương trình phụ thuộc đi kèm là (**xval-prep** và **average**).

Chi tiết hơn về mô hình phân lớp C4.5 có thể tham khảo tại địa chỉ:

<http://www2.cs.uregina.ca/~hamilton/courses/831/notes/ml/dtrees/c4.5/tutorial.html>

### 3.2.2. Cấu trúc dữ liệu sử dụng trong C4.5

Mỗi bộ dữ liệu dùng trong C4.5 gồm có 3 file:

#### 3.2.2.1. Filestem.names: định nghĩa bộ dữ liệu

|  |  |
|--|--|
| Thuộc tính phân lớp là: MOBILE-PRODUCER_ID   |  |
| 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 50, 94, 95, 97, 98, 100 class |  |
| Danh sách các thuộc tính là:   |  |
| GENDER:  | M, F   |
| CITY:  | AnGiang, BacGiang, BacNinh, BacLieu, BacCan, BaRiaVungTau, BenTre, BinhDinh, Binh    |
| COUNTRY:   | VIETNAM, OTHERS  |
| JOB:   | Accounting, Choose a job, ComputerRelated, Consulting, CustomerService, Education, E |
| MOBILE_PRODUCER_ID:  | continuous   |
| PRODUCTER_VERSION_ID:  | continuous   |
| MOBILE_NUMBER:   | continuous   |
| MOBILE_SERVICE_ID:   | 1, 2, 3, 4   |
| ACTIVE/REGISTRATION_DATE_MONTH:  | JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OC              |
| ACTIVE/REGISTRATION_DATE_YEAR:   | 2003, 2004, 2005   |
| ACCESS_TIMES:  | continuous   |
| LAST_ACCESS_DATE_1:  | MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY                       |
| LAST_ACCESS_DATE_MONTH:  | JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OC              |
| LAST_ACCESS_DATE_DAY:  | continuous   |
| LAST_ACCESS_DATE_YEAR:   | 2004, 2005   |
| NUMBER_MESSAGE_FREE:   | continuous   |
| WEB_SETTING_ID:  | 1, 2, 3  |
| BIRTHDAY_1:  | MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY                       |
| BIRTHDAY_MONTH:  | JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OC              |
| BIRTHDAY_DAY:  | continuous   |
| BIRTHDAY_YEAR:   | continuous   |
| PREF_LANG:   | VN, EN   |
| ACTIVE_TYPE:   | 1, 2   |

Hình 18 - File định nghĩa cấu trúc dữ liệu sử dụng trong thực nghiệm

Mô tả:

- Dòng trên cùng định nghĩa các giá trị phân lớp theo thuộc tính được chọn (ví dụ trên hình 18 là thuộc tính MOBILE\_PRODUCER\_ID)
- Các dòng tiếp theo là danh sách các thuộc tính cùng với tập giá trị của nó trong tập dữ liệu. Các thuộc tính liên tục được định nghĩa bằng từ khóa “continuous”
- Chú thích được định nghĩa sau dấu “|”

#### 3.2.2.2. Filestem.data: chứa dữ liệu training



```

1 M,HaNoi,VIETNAM,?,100,84904000001,1, APRIL,2004,3,TUESDAY, APRIL,2005,5,1, JANUARY,1972,VN,1,7
2 F,HaNoi,VIETNAM,?,508,84904000002,1, DECEMBER,2003,5,SATURDAY, APRIL,2005,5,1, JUNE,1974,VN,0,1
3 M,HaNoi,VIETNAM,Engineering,7,84904000003,1, DECEMBER,2003,0,FRIDAY, NOVEMBER,2004,5,1, APRIL,1976,VN,0,10
4 M,HaNoi,VIETNAM,Professional,508,84904000008,1, AUGUST,2004,0,WEDNESDAY, AUGUST,2004,5,1, JULY,1968,VN,0,1
5 M,HaNoi,VIETNAM,?,100,84904000011,1, AUGUST,2004,0,THURSDAY, AUGUST,2004,5,1, MAY,1979,VN,0,7
6 M,HaNoi,VIETNAM,ComputerRelated,938,84904000013,1, NOVEMBER,2004,0,SUNDAY, JANUARY,2005,5,1, FEBRUARY,1982,VN,0,12
7 F,HaNoi,VIETNAM,Operation,63,84904000037,1, JANUARY,2004,0,FRIDAY, MARCH,2005,5,1, NOVEMBER,1973,VN,0,1
8 M,HaNoi,VIETNAM,Sale,137,84904000038,1, JANUARY,2004,0,WEDNESDAY, MARCH,2005,5,1, NOVEMBER,1977,VN,0,1
9 M,HaNoi,VIETNAM,Executive,?,84904000040,1, DECEMBER,2003,3,SUNDAY, APRIL,2005,5,1, SEPTEMBER,1977,VN,0,11
10 M,ThaiBinh,VIETNAM,?,?,84904000046,1, MARCH,2005,0,?,?,5,1, MAY,1960,VN,0,?
11 M,VAP,VIETNAM,Other,?,84904000055,?, DECEMBER,2004,0,WEDNESDAY, MARCH,2005,5,1, APRIL,1982,?,1,?
12 M,HaNoi,VIETNAM,?,?,84904000060,1, JULY,2004,0,THURSDAY, JULY,2004,5,1, OCTOBER,1976,VN,0,?
13 M,HaNoi,VIETNAM,Executive,215,84904000068,1, DECEMBER,2003,0,WEDNESDAY, OCTOBER,2004,5,1, OCTOBER,1977,VN,0,2
14 M,ThaiNguyen,VIETNAM,Supervisory,179,84904000080,1, DECEMBER,2004,0,WEDNESDAY, DECEMBER,2004,5,1, AUGUST,1983,VN,0,2
15 F,HaNoi,VIETNAM,?,303,84904000082,2, MARCH,2004,0,SATURDAY, MARCH,2005,5,1, AUGUST,1982,VN,0,1

```

Hình 19 - File chứa dữ liệu cần phân lớp

*Filestem.data* có cấu trúc như sau: mỗi dòng tương ứng với một bản ghi (cases) trong cơ sở dữ liệu. Mỗi dòng một bộ giá trị theo thứ đã định của các thuộc tính định nghĩa trong *filestem.names*. Các giá trị ngăn cách nhau bởi dấu phẩy. Giá trị thiếu (*missing value*) được biểu diễn bằng dấu “?”.

### 3.2.2.3. Filestem.test: chứa dữ liệu test

File này chứa dữ liệu test trên mô hình phân lớp đã được tạo ra từ tập dữ liệu training, và có cấu trúc giống *filestem.data*

## 3.3. Kết quả thực nghiệm

### 3.3.1. Một số kết quả phân lớp tiêu biểu:

#### 3.3.1.1. Cây quyết định

Lệnh tạo cây quyết định

```
$ ./C4.5 -f ../Data/Classes/10-5/class -u >> ../Data/Classes/10-5/class.dt
```

Tham số tùy chọn:

- f: xác định bộ dữ liệu cần phân lớp
- u: tùy chọn cây được tạo ra được đánh giá trên tập dữ liệu test.
- v verb: mức độ chi tiết của output [0..3], mặc định là 0
- t trials: thiết lập chế độ interactive với trials là số cây thử nghiệm. Interactive là chế độ cho phép tạo ra nhiều cây thử nghiệm bắt đầu với một tập con dữ liệu được chọn ngẫu nhiên. Mặc định là chế độ batch với toàn bộ tập dữ liệu được sử dụng để tạo một cây quyết định duy nhất.

Cây quyết định có các node trong là các kiểm tra giá trị của thuộc tính được chọn để phát triển tại node đó. Lá của cây quyết định có định dạng: *Giá\_trị\_phân\_lớp (N/E) hoặc (N)*. Với N/E là tỉ lệ giữa tổng các case đạt tới lá đó với số case đạt tới lá đó nhưng thuộc về lớp khác (trong tập dữ liệu đào tạo).

```
190713 C4.5 [release 8] decision tree generator    Thu April 7 18:44:20 2005
190714 -----
190715
190716 Options:
190717   File stem <../Data/Attributes/test3-1>
190718
190719 Read 121218 cases (8 attributes) from ../Data/Attributes/test3-1.data
190720
190721 Decision Tree:
190722
190723 ACTIVE_TYPE = 0:
190724 | GENDER = M:
190725 | | CITY = CaoBang: 1 (4.0/2.0)
190726 | | CITY = WAP: ? (111.0/1.0)
190727 | | CITY = AnGiang:
190728 | | | JOB = Choose a job: 1 (0.0)
190729 | | | JOB = Consulting: 1 (1.3/0.2)
190730 | | | JOB = Executive: 1 (10.1/2.4)
190731 | | | JOB = Homemaker: 1 (0.0)
190732 | | | JOB = Operation: 1 (1.3/0.2)
190733 | | | JOB = Professional: 1 (0.0)
190734 | | | JOB = Research: 2 (1.3/0.3)
190735 | | | JOB = Retired: 1 (1.3/0.2)
190736 | | | JOB = Sale: 1 (11.4/2.6)
190737 | | | JOB = Unemployed: 1 (1.3/0.2)
190738 | | | JOB = Accounting:
190739 | | | | ACCESS_TIMES > 2 : 2 (2.3/0.3)
190740 | | | | ACCESS_TIMES <= 2 :
190741 | | | | | BIRTHDAY_YEAR <= 1982 : 1 (4.2/0.8)
190742 | | | | | BIRTHDAY_YEAR > 1982 : 5 (2.4/0.4)
190743 | | | | JOB = ComputerRelated:
```

Hình 20 - Dạng cây quyết định tạo ra từ tập dữ liệu thử nghiệm

|   |             |               |             |                 |    |
|---|-------------|---------------|-------------|-----------------|----|
| Evaluation on training data (131065 items): |             |               |             |                 |    |
| Before Pruning                              |             | After Pruning |             |                 |    |
| Size  | Errors      | Size          | Errors      | Estimate        |    |
| 238   | 1955( 1.5%) | 1             | 1961( 1.5%) | ( 1.5%)         | << |
| Evaluation on test data (65536 items):      |             |               |             |                 |    |
| Before Pruning                              |             | After Pruning |             |                 |    |
| Size  | Errors      | Size          | Errors      | Estimate        |    |
| 238   | 829( 1.3%)  | 1             | 830( 1.3%)  | ( 1.5%)         | << |
| (a)   | (b)         | (c)           | (d)         | <-classified as |    |
| 64706                                       |             |               |             | (a): class 1    |    |
| 432   |             |               |             | (b): class 2    |    |
| 398   |             |               |             | (c): class 3    |    |
|   |             |               |             | (d): class ?    |    |

Hình 21 - Ước lượng trên cây quyết định vừa tạo ra trên tập dữ liệu training và tập dữ liệu test

Sau khi cây quyết định được tạo ra, nó sẽ được ước lượng lại độ chính xác trên chính tập dữ liệu đào tạo vừa học được, và có thể được ước lượng trên tập dữ liệu test độc lập với dữ liệu training nếu có tùy chọn từ phía người dùng.

Các ước lượng được thực hiện trên cây khi chưa cắt tỉa và sau khi đã cắt tỉa. Mô hình C4.5 cũng cho phép truyền các tham số về mức độ cắt tỉa của cây, mặc định là cắt tỉa 25%.

### 3.3.1.2. Các luật sản xuất tiêu biểu

Lệnh tạo luật sản xuất khi đã có cây quyết định:

```
$ ./C4.5rules -f ../Data/Classes/10-5/class -u >> ../Data/Classes/10-5/class.r
```

Các tham số tùy chọn -f, -v, -u giống như với lệnh tạo cây quyết định.

Mỗi luật sinh ra gồm có 3 phần:

- Điều kiện phân lớp
- Giá trị phân lớp ( ->class ...)
- []: dự đoán độ chính xác của luật. Giá trị này được ước lượng trên tập training và test (nếu có tùy chọn -u khi sinh luật)

```
Rule 233:
  CITY = HaNoi
  JOB = Other
  BIRTHDAY_MONTH = APRIL
  BIRTHDAY_YEAR > 1981|
  BIRTHDAY_YEAR <= 1982
  -> class 1 [96.6%]

Rule 417:
  CITY = WAP
  -> class 1 [96.0%]

Rule 272:
  CITY = HaNoi
  JOB = ComputerRelated
  MOBILE_SERVICE_ID = 2
  ACTIVE/REGISTRATION_DATE_MONTH = MAY
  BIRTHDAY_MONTH = APRIL
  BIRTHDAY_YEAR > 1981
  -> class 1 [54.6%]
```

Hình 22 - Một số luật rút ra từ bộ dữ liệu 19 thuộc tính, phân lớp loại thiết lập chế độ giao diện của người sử dụng (WEB\_SETTING\_ID)

Việc đưa ra được các luật liên quan đến sở thích giao diện sử dụng của khách hàng giúp ích cho công việc thiết kế, cũng như tạo các loại giao diện phù hợp cho từng loại đối tượng khách hàng khác nhau. Ví dụ, **Rule 233** trong hình 22 cho thấy, nếu khách hàng đăng ký sử dụng dịch vụ tại *Hà Nội*, nghề nghiệp thuộc nhóm *Other* và sinh năm 1982 thì chế độ giao diện mà người đó sử dụng có mã số là 1. Kết luận này có độ chính xác là 96,6%.

```
Rule 1648:
  CITY = KhanhHoa
  JOB = CustomerService
  MOBILE_SERVICE_ID = 2
  BIRTHDAY_YEAR > 1979
  -> class 1 [93.0%]

Rule 1908:
  GENDER = M
  CITY = PhuYen
  JOB = Accounting
  -> class 1 [92.6%]

Rule 1021:
  JOB = Supervisory
  BIRTHDAY_YEAR > 1969
  BIRTHDAY_YEAR <= 1973
  -> class 1 [91.7%]

Rule 2892:
  CITY = DaNang
  JOB = Student
  ACTIVE/REGISTRATION_DATE_YEAR = 2004
  ACCESS_TIMES > 0
  ACCESS_TIMES <= 1
  BIRTHDAY_YEAR > 1982
  -> class 1 [91.7%]
```

Hình 23 - Một số luật rút ra từ bộ dữ liệu 8 thuộc tính, phân lớp theo số hiệu nhà sản xuất điện thoại (*PRODUCTER\_ID*)

Từ kết quả thực tế hình 23, từ **Rule 1021**, chúng ta có thể kết luận: nếu khách hàng làm công việc *Supervisory* và sinh trong khoảng từ năm 1969 đến 1973 thì loại điện thoại mà khách hàng dùng có số hiệu là 1 (là điện thoại SAMSUNG). Độ chính xác của kết luận này là 91,7%.

Những luật như trên giúp cho các nhân viên maketing có thể tìm ra được thị trường điện thoại di động đối với từng loại đối tượng khách hàng khác nhau, từ đó có các chiến lược phát triển sản phẩm hợp lý.

|  |
|--|
| Rule 661:<br>GENDER = F<br>JOB = Engineering<br>MOBILE_PRODUCER_ID = 4<br>ACTIVE/REGISTRATION_DATE_YEAR = 2004<br>-> class 2 [79.4%]   |
| Rule 709:<br>GENDER = M<br>JOB = Government<br>MOBILE_PRODUCER_ID = 1<br>ACTIVE/REGISTRATION_DATE_YEAR = 2004<br>ACCESS_TIMES > 3<br>ACCESS_TIMES <= 4<br>-> class 2 [79.4%] |
| Rule 993:<br>JOB = Supervisory<br>MOBILE_PRODUCER_ID = 7<br>-> class 2 [79.4%]   |
| Rule 1231:<br>GENDER = F<br>CITY = HaNoi<br>JOB = Other<br>ACTIVE/REGISTRATION_DATE_YEAR = 2004<br>BIRTHDAY_YEAR > 1978<br>-> class 2 [79.4%]                                |

Hình 24 - Một số luật sinh ra từ tập dữ liệu 8 thuộc tính, phân lớp theo dịch vụ điện thoại mà khách hàng sử dụng (MOBILE\_SERVICE\_ID)

Ví dụ từ **Rule 661**: nếu khách hàng là nam (F), nghề nghiệp *Engineering*, điện thoại sử dụng là Erricsion (MOBILE\_PRODUCER\_ID = 4) và đăng ký năm 2004, thì dịch vụ mà khách hàng đó sử dụng là gửi logo (MOBILE\_SERVICE\_ID = 2). Độ chính xác của luật này là 79,4%.

Từ những luật như vậy, ta có thể thống kê cũng như dự đoán được xu hướng sử dụng các loại dịch vụ của từng đối tượng khách hàng khác nhau. Từ đó có chiến lược phát triển dịch vụ khách hàng hiệu quả.

| Evaluation on training data (37837 items): |      |       |      |            |            |   |  |
|--|------|-------|------|------------|------------|---|--|
| Rule                                       | Size | Error | Used | Wrong      | Advantage  |   |  |
| 246  | 5    | 1.1%  | 129  | 0 (0.0%)   | 88 (88 0)  | 2 |  |
| 77   | 6    | 27.3% | 29   | 16 (55.2%) | -5 (11 16) | 2 |  |
| 79   | 5    | 29.3% | 2    | 0 (0.0%)   | 2 (2 0)    | 2 |  |
| 64   | 6    | 50.0% | 2    | 0 (0.0%)   | 2 (2 0)    | 2 |  |
| 68   | 4    | 50.0% | 2    | 0 (0.0%)   | 2 (2 0)    | 2 |  |
| 161  | 6    | 2.6%  | 2234 | 52 (2.3%)  | -4 (3 7)   | 1 |  |
| 321  | 1    | 4.1%  | 239  | 33 (13.8%) | -11 (4 15) | 1 |  |
| 223  | 6    | 45.4% | 5    | 1 (20.0%)  | 3 (4 1)    | 1 |  |
| 115  | 5    | 50.0% | 2    | 0 (0.0%)   | 0 (0 0)    | 1 |  |
| 126  | 5    | 50.0% | 2    | 0 (0.0%)   | 2 (2 0)    | 1 |  |
| 249  | 6    | 50.0% | 2    | 0 (0.0%)   | 0 (0 0)    | 1 |  |
| 260  | 5    | 50.0% | 2    | 0 (0.0%)   | 0 (0 0)    | 1 |  |
| 151  | 1    | 0.1%  | 2325 | 0 (0.0%)   | 0 (0 0)    | 0 |  |
| 158  | 2    | 0.1%  | 1454 | 0 (0.0%)   | 4 (4 0)    | 0 |  |
| 322  | 2    | 0.1%  | 2101 | 0 (0.0%)   | 0 (0 0)    | 0 |  |
| 267  | 4    | 0.2%  | 570  | 0 (0.0%)   | 4 (4 0)    | 0 |  |
| 228  | 2    | 0.2%  | 1508 | 5 (0.3%)   | 9 (9 0)    | 0 |  |
| 39   | 3    | 0.2%  | 975  | 2 (0.2%)   | 5 (5 0)    | 0 |  |
| 37   | 1    | 0.2%  | 2331 | 4 (0.2%)   | 9 (9 0)    | 0 |  |
| 81   | 2    | 0.2%  | 3939 | 10 (0.3%)  | 20 (21 1)  | 0 |  |
| 166  | 2    | 0.3%  | 1539 | 3 (0.2%)   | 7 (8 1)    | 0 |  |
| 30   | 2    | 0.3%  | 53   | 0 (0.0%)   | 1 (1 0)    | 0 |  |
| 69   | 1    | 0.3%  | 1808 | 8 (0.4%)   | 4 (5 1)    | 0 |  |

Hình 25 - Ước lượng tập luật trên tập dữ liệu đào tạo

Sau khi được tạo ra, tập luật được ước lượng lại trên tập training data, hay tập dữ liệu test (tùy chọn).

Mô tả các một số trường tiêu biểu:

- *Rule*: số hiệu của luật
- *Size*: Kích thước của luật (số các điều kiện so sánh trong phần điều kiện phân lớp)
- *Used*: số lượng cases trong tập training áp dụng luật đó. Trường này quy định tính phổ biến của luật.
- *Wrong*: số lượng case phân lớp sai -> tỉ lệ phần trăm lỗi

### Kết luận

Từ quá trình thực nghiệm, chúng tôi nhận thấy vai trò của quá trình tiền xử lý dữ liệu là rất quan trọng. Trong quá trình này, cần xác định chính xác những thông tin gì cần rút ra từ cơ sở dữ liệu đó, từ đó chọn thuộc tính phân lớp phù hợp. Sau đó việc

lựa chọn những thuộc tính liên quan là rất quan trọng, nó quyết định mô hình phân lớp có đúng đắn không, có ý nghĩa thực tế không và có thể áp dụng cho những dữ liệu tương lai hay không.

### 3.3.2. Các biểu đồ hiệu năng

Các tham số ảnh hưởng đến hiệu năng của mô hình phân lớp là [6]:

- Số các bản ghi trong tập dữ liệu đào tạo (N)
- Số lượng thuộc tính (A)
- Số các giá trị rời rạc của mỗi thuộc tính (nhân tố nhánh) (V)
- Số các lớp (C)

Chi phí xây dựng cây quyết định là tổng chi phí xây dựng từng node:

$$T = \sum t_{\text{node}}(i)$$

Chi phí tốn cho node  $i$  được tính bằng tổng các khoản chi phí riêng cho từng công việc:

$$t_{\text{node}}(i) = t_{\text{single}}(i) + t_{\text{freq}}(i) + t_{\text{info}}(i) + t_{\text{div}}(i)$$

Với:

- $t_{\text{single}}(i)$  là chi phí thực thi việc kiểm tra xem liệu tất cả các case trong tập dữ liệu đào tạo có thuộc về cùng một lớp không?
- $t_{\text{div}}(i)$  là chi phí phân chia tập dữ liệu theo thuộc tính đã chọn
- Việc lựa chọn thuộc tính có *Information gain* lớn nhất trong tập dữ liệu hiện tại là kết quả của việc tính *Information gain* của từng thuộc tính. Chi phí cho quá trình này bao gồm thời gian tính toán tần suất phân phối theo các giá trị phân lớp của từng thuộc tính ( $t_{\text{freq}}(i)$ ) và thời gian để tính *Information gain* từ các thông tin phân phối đó ( $t_{\text{info}}(i)$ ).

Có thể biểu diễn sự phụ thuộc của các khoản chi phí trên vào các tham số hiệu năng đã mô tả ở trên như sau:

$$t_{\text{freq}} = k_1 * A_i N_i$$

$$t_{\text{info}} = k_2 * C A_i V$$

$$t_{\text{div}} = k_3 * A_i$$

$$t_{\text{single}} = k_4 * N_i$$



Với  $k_j$  là hằng số có giá trị tùy theo từng ứng dụng cụ thể. Số lượng bản ghi ( $N_i$ ) và số lượng thuộc tính ( $A_i$ ) tương ứng với từng node phụ thuộc vào độ sâu của node đó và bản thân tập dữ liệu.

Việc xác định chính xác chi phí cho quá trình xây dựng cây quyết định (T) là rất khó và cần phải biết chính xác hình dáng của cây quyết định, điều này không thể xác định trong thời gian chạy. Chính vì vậy mà T được đơn giản hóa bằng cách dùng giá trị trung bình đi kèm với những giả sử về hình dáng của cây và giải các phương trình lặp cho từng thành phần riêng lẻ của mô hình [6].

Sau đây là các kết quả thực nghiệm đánh giá ảnh hưởng của các tham số hiệu năng như kích thước tập dữ liệu đào tạo, số lượng thuộc tính, thuộc tính liên tục, và số giá trị phân lớp tới mô hình phân lớp C4.5:

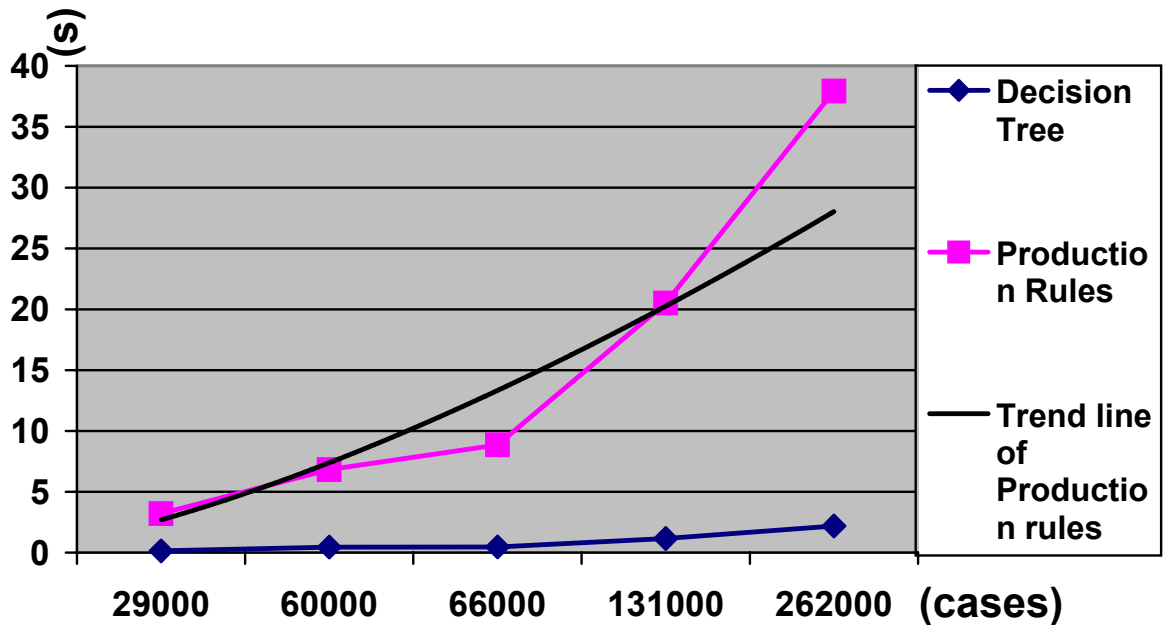
### 3.3.2.1. Thời gian thực thi phụ thuộc vào kích thước tập dữ liệu đào tạo

Các thử nghiệm đã được tiến hành trên nhiều tập dữ liệu với kích thước, số lượng thuộc tính và thuộc tính phân lớp khác nhau. Sau đây là các bảng kết quả và biểu đồ thể hiện sự phụ thuộc đang xét.

#### Thử nghiệm với tập dữ liệu 2 thuộc tính

Bảng 2 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 2 thuộc tính

| Kích thước tập dữ liệu    | 29000 | 60000 | 66000 | 131000 | 262000 |
|---------------------------|-------|-------|-------|--------|--------|
| Thời gian xây dựng (giây) |       |       |       |        |        |
| Decision Tree             | 0.15  | 0.46  | 0.47  | 1.17   | 2.2    |
| Production Rules          | 3.21  | 6.82  | 8.85  | 20.51  | 37.94  |

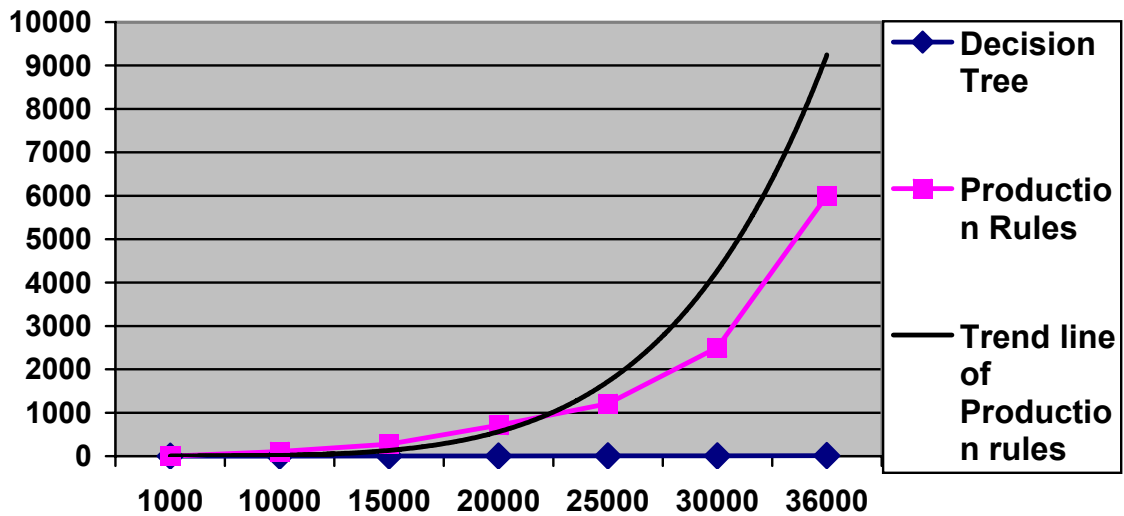


Biểu đồ 2 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 2 thuộc tính

### Thử nghiệm với tập dữ liệu 7 thuộc tính

Bảng 3 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 7 thuộc tính

| Kích thước tập dữ liệu    | 1000 | 10000 | 15000 | 20000 | 25000  | 30000  | 36000  |
|---------------------------|------|-------|-------|-------|--------|--------|--------|
| Thời gian xây dựng (giây) |      |       |       |       |        |        |        |
| Decision Tree             | 0.03 | 0.46  | 1.90  | 2.79  | 5.70   | 8.31   | 13.34  |
| Production Rules          | 0.13 | 107.1 | 276.2 | 709.9 | 1211.0 | 2504.8 | 5999.5 |

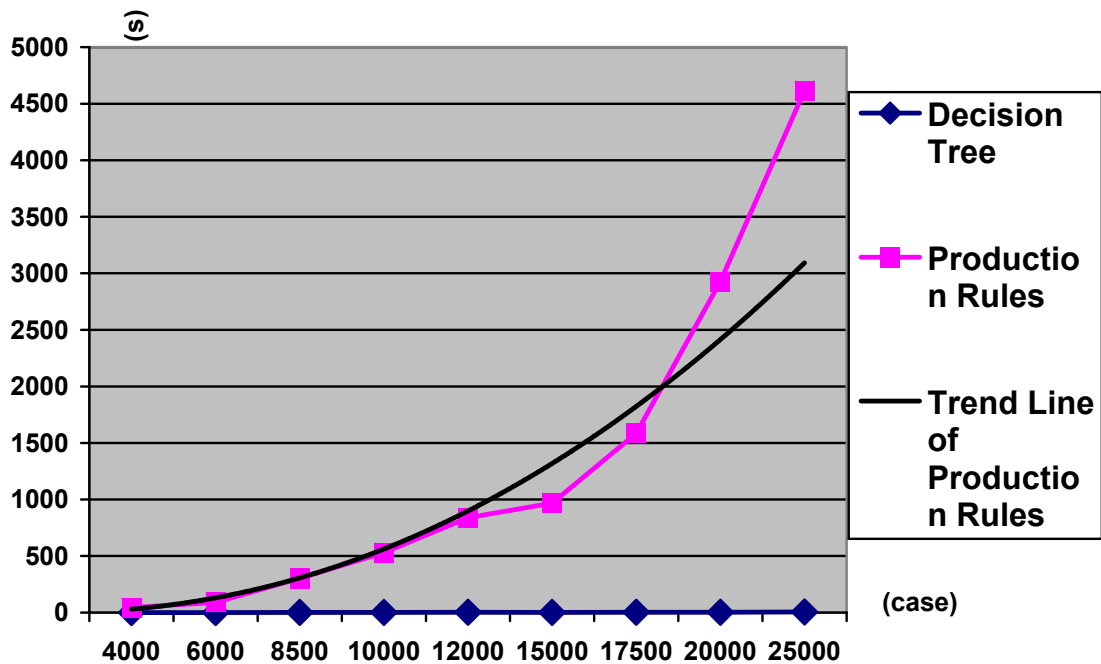


Biểu đồ 3 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 7 thuộc tính

### Thử nghiệm với tập dữ liệu 18 thuộc tính

Bảng 4 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 18 thuộc tính

| Kích thước tập dữ liệu xây dựng (giây) | 4000 | 6000  | 8500   | 10000  | 12000  | 15000  | 17500   | 20000   | 25000   |
|--|------|-------|--------|--------|--------|--------|---------|---------|---------|
| Decision Tree                          | 0.45 | 0.64  | 1.32   | 1.77   | 2.37   | 1.8    | 2.68    | 2.98    | 5.24    |
| Production Rules                       | 43.6 | 90.77 | 304.07 | 531.34 | 838.88 | 968.24 | 1584.63 | 2927.56 | 4617.23 |



Biểu đồ 4 - Thời gian xây dựng cây quyết định và tập luật sản xuất phụ thuộc vào kích thước tập dữ liệu đào tạo 18 thuộc tính

Các đánh giá sự phụ thuộc của thời gian thực thi vào kích thước tập dữ liệu đào tạo đã được tiến hành trên các tập dữ liệu với số lượng thuộc tính khác nhau. Có thể rút ra các kết luận sau:

- Kích thước tập dữ liệu càng lớn thì thời gian sinh cây quyết định cũng như thời gian sinh tập luật sản xuất càng lớn. Căn cứ vào các đường trendline của đường

biểu diễn thời gian sinh tập luật sản xuất được vẽ thêm trên các biểu đồ, chúng tôi dự đoán sự phụ thuộc trên được diễn đạt bằng hàm đa thức.

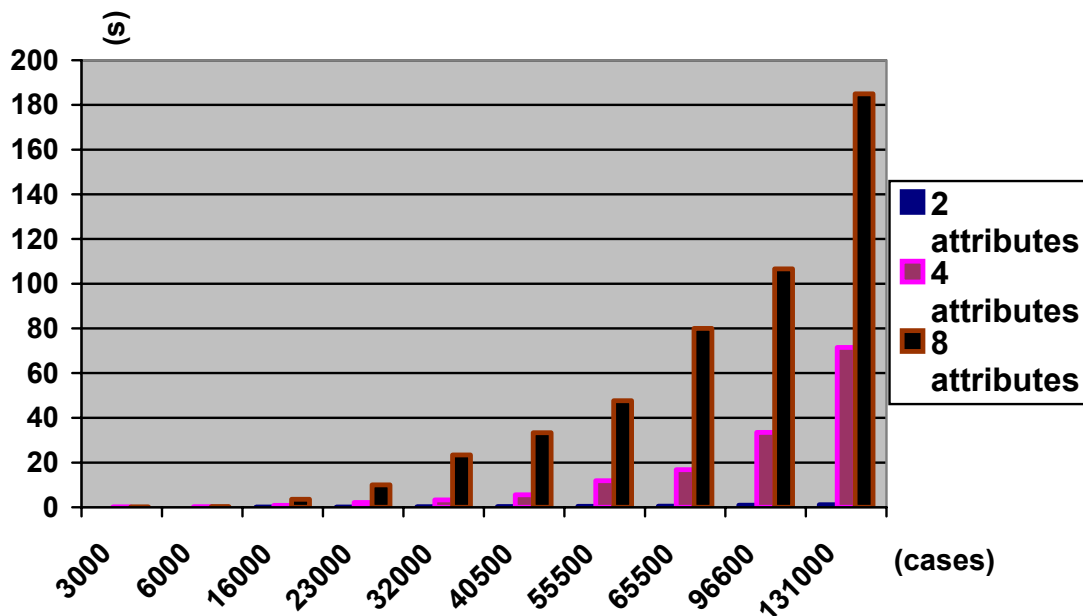
- Các biểu đồ trên cho thấy quá trình sinh luật sản xuất sau từ cây quyết định đã tạo ra tốn tài nguyên tính toán gấp nhiều lần so với quá trình sinh cây quyết định. Thực nghiệm cho thấy với những tập dữ liệu cỡ trăm nghìn bản ghi, thời gian sinh luật sản xuất là khá lâu (thông thường > 5 giờ). Đó cũng là một trong những lý do khiến C4.5 không thể áp dụng với những tập dữ liệu lớn. Tập dữ liệu đào tạo có càng nhiều thuộc tính thì sự chênh lệch về thời gian thực thi giữa 2 quá trình trên càng lớn.

### 3.3.2.2. Hiệu năng của C4.5 phụ thuộc vào số lượng thuộc tính

Để đánh giá sự phụ thuộc trên, các thử nghiệm đã tiến hành với 3 tập dữ liệu có 2, 4, và 8 thuộc tính rời rạc, với cùng thuộc tính phân lớp.

Bảng 5 - Thời gian sinh cây quyết định phụ thuộc vào số lượng thuộc tính

|              | 3000 | 6000 | 16000 | 23000 | 32000 | 40500 | 55500 | 65500 | 96600  | 131000 |
|--------------|------|------|-------|-------|-------|-------|-------|-------|--------|--------|
| 2 attributes | 0.01 | 0.02 | 0.05  | 0.1   | 0.18  | 0.25  | 0.39  | 0.47  | 0.89   | 1.17   |
| 4 attributes | 0.12 | 0.18 | 0.82  | 2.18  | 3.32  | 5.58  | 11.83 | 16.79 | 33.49  | 71.52  |
| 8 attributes | 0.14 | 0.3  | 3.56  | 9.99  | 23.40 | 33.36 | 47.62 | 80    | 106.61 | 185    |



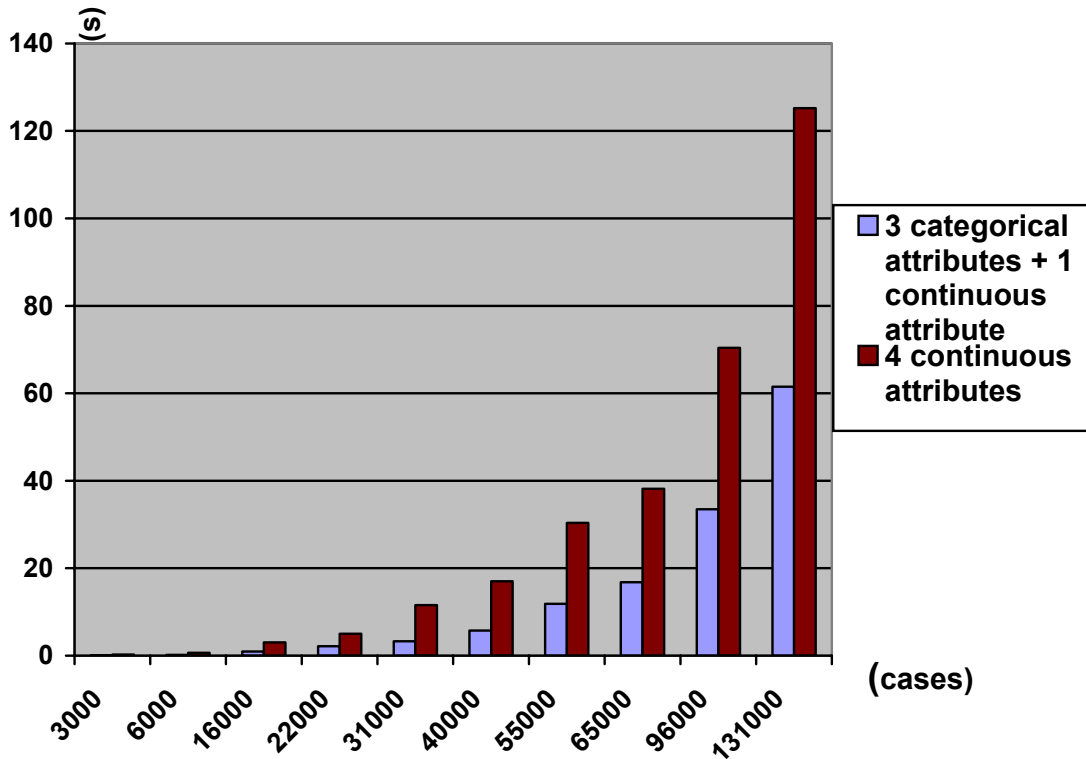
Biểu đồ 5 - Sự phụ thuộc thời gian sinh cây quyết định vào số lượng thuộc tính

Thời gian C4.5 xây dựng cây quyết định phụ thuộc vào số lượng thuộc tính qua các khoảng thời gian  $t_{freq}$ ,  $t_{info}$ ,  $t_{div}$ . Số thuộc tính càng nhiều thời gian tính toán để lựa chọn thuộc tính tốt nhất test tại mỗi node càng lớn, vì vậy thời gian sinh cây quyết định càng tăng. Do vậy C4.5 bị hạn chế về số lượng thuộc tính trong tập dữ liệu đào tạo [2]. Đây là một điểm khác biệt so với SPRINT

### 3.3.2.3. Hiệu năng của C4.5 khi thao tác với thuộc tính liên tục

Bảng 6 - Thời gian xây dựng cây quyết định với thuộc tính rời rạc và thuộc tính liên tục

|  | 3000 | 6000 | 16000 | 22000 | 31000 | 40000 | 55000 | 65000 | 96000 | 131000 |
|--|------|------|-------|-------|-------|-------|-------|-------|-------|--------|
| 3 thuộc tính rời rạc+<br>1 thuộc tính liên tục | 0.12 | 0.18 | 0.92  | 2.18  | 3.32  | 5.74  | 11.83 | 16.79 | 33.47 | 61.52  |
| 4 thuộc tính liên tục                          | 0.24 | 0.66 | 3.02  | 5.01  | 11.56 | 16.99 | 30.37 | 38.16 | 70.38 | 125.21 |



Biểu đồ 6 - So sánh thời gian xây dựng cây quyết định từ tập thuộc tính liên tục và từ tập thuộc tính rời rạc

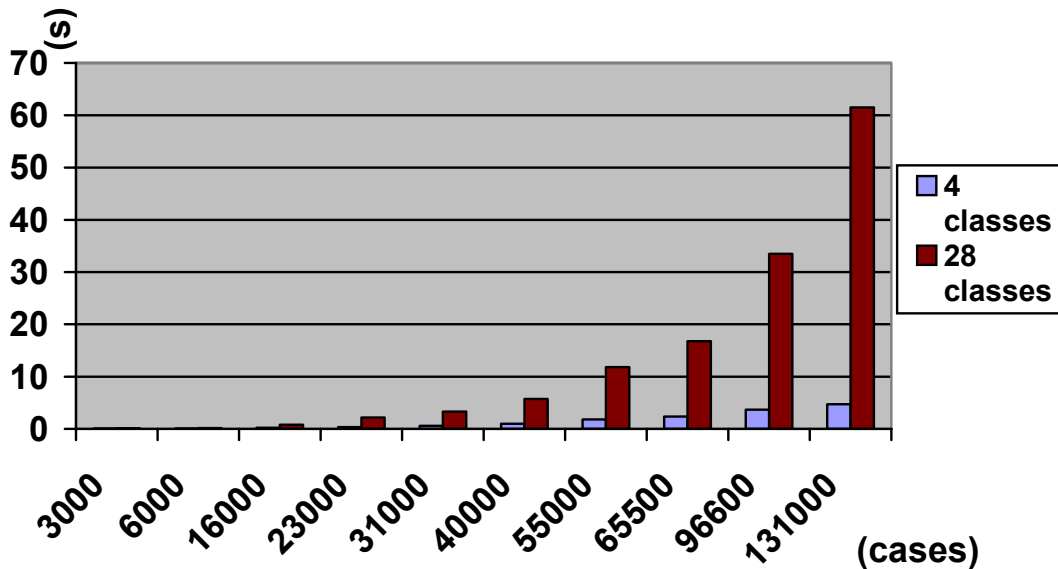
Như đã phân tích trong thuật toán C4.5 cũng như trong thuật toán phân lớp dữ liệu dựa trên cây quyết định nói chung, việc thao tác với thuộc tính liên tục chiếm nhiều tài nguyên tính toán hơn với thuộc tính rời rạc. Do vậy tập dữ liệu có nhiều

thuộc tính liên tục ảnh hưởng đáng kể đến thời gian sinh cây quyết định so với tập dữ liệu có nhiều thuộc tính rời rạc.

### 3.3.2.4. Hiệu năng của C4.5 khi thao tác với nhiều giá trị phân lớp

Bảng 7 - Thời gian sinh cây quyết định phụ thuộc vào số giá trị phân lớp

|            | 3000 | 6000 | 16000 | 23000 | 31000 | 40000 | 55000 | 65500 | 96600 | 131000 |
|------------|------|------|-------|-------|-------|-------|-------|-------|-------|--------|
| 4 classes  | 0.04 | 0.07 | 0.22  | 0.35  | 0.61  | 0.97  | 1.8   | 2.36  | 3.68  | 4.72   |
| 28 classes | 0.12 | 0.18 | 0.82  | 2.18  | 3.32  | 5.74  | 11.83 | 16.79 | 33.49 | 61.51  |



Biểu đồ 7 - Thời gian sinh cây quyết định phụ thuộc vào số giá trị phân lớp

Càng nhiều giá trị phân lớp thì thời gian tính *Information gain* cho từng thuộc tính ( $t_{info}$ ) càng nhiều. Do vậy thời gian sinh cây quyết định càng lâu.

### 3.4. Một số đề xuất cải tiến mô hình phân lớp C4.5

Từ quá trình nghiên cứu mô hình phân lớp C4.5 cũng như những so sánh với SPRINT để thấy được ưu nhược điểm của thuật toán. Và từ quá trình thực nghiệm chúng tôi đưa ra một số đề xuất cải tiến thuật toán C4.5

1. Sinh luật sản xuất là một tính năng mới của C4.5 so với các thuật toán khác. Hiện nay với cơ sở dữ liệu lớn, tập luật sinh ra là rất dài, ví dụ với tập training cỡ 30000 cases với 8 thuộc tính, tập luật có thể lên tới 3000 luật. Do đó việc xem và trích rút thông tin có ích trên tập luật là khó khăn. Trên thực tế đó, chúng tôi đề xuất tích hợp thêm vào C4.5 module trích chọn tập những luật “tốt

nhất” có là những *luật có độ chính xác chấp nhận được* (mức độ chính xác có thể do người dùng tùy chọn) và có *độ phổ biến cao* (là những luật mà áp dụng được trên nhiều case trong tập dữ liệu thử nghiệm).

2. Sinh luật sản xuất là một tính năng mới, đem lại nhiều lợi ích của C4.5 so với các thuật toán phân lớp dữ liệu khác. Nhưng quá trình sinh luật sản xuất tốn rất nhiều tài nguyên tính toán so với quá trình sinh cây quyết định. Do vậy cần song song hóa giai đoạn sinh luật để cải tiến hiệu năng của C4.5
3. C4.5 bị hạn chế về số lượng thuộc tính trong tập dữ liệu đào tạo, và độ chính xác của các cây quyết định hay các luật sinh ra nói chung là chưa cao. Cần tập trung sử dụng các phương pháp cải tiến độ chính xác của mô hình phân lớp như *bagging*, *boosting*.
4. C4.5 thao tác với thuộc tính liên tục lâu hơn thuộc tính rời rạc. Điều này có thể giải thích bởi: với thuộc tính liên tục có  $n$  giá trị đã sắp xếp, thuật toán cần độ đo phân chia tại  $(n-1)$  ngưỡng nằm giữa 2 giá trị liên nhau trong dãy sắp xếp. Từ đó mới có thể tìm ra được một ngưỡng tốt nhất để test trên thuộc tính đó. Trong tập dữ liệu đào tạo, thuộc tính liên tục càng nhiều giá trị, thì tài nguyên tính toán bỏ ra để thao tác với nó càng nhiều. Hiện nay đã có một số đề xuất cải tiến cách xử lý với thuộc tính liên tục [3][8], đó là một trong những hướng nghiên cứu đang nghiên cứu của đề tài.
5. Chúng tôi đề xuất cơ chế sắp xếp trước có sử dụng lược đồ phân phối lớp một lần như của SPRINT áp dụng vào C4.5. Từ đó tiến tới xây dựng cơ chế lưu trữ dữ liệu thường trú trên đĩa. Nếu thực hiện được sẽ làm tăng hiệu năng cũng như khả năng mở rộng của mô hình phân lớp C4.5.



## KẾT LUẬN

Trong khuôn khổ khóa luận tốt nghiệp này, chúng tôi đã nghiên cứu, phân tích, đánh giá các thuật toán phân lớp dữ liệu dựa trên cây quyết định. Tiêu biểu là 2 thuật toán C4.5 và SPRINT. C4.5 và SPRINT có cách thức lưu trữ dữ liệu và xây dựng cây quyết định dựa trên những độ đo khác nhau. Do đó hai thuật toán này có phạm vi ứng dụng vào các cơ sở dữ liệu có kích thước khác nhau.

C4.5 là thuật toán xử lý đầy đủ các vấn đề của quá trình phân lớp dữ liệu: lựa chọn thuộc tính tốt nhất, lưu trữ phân chia dữ liệu, xử lý giá trị thiếu, tránh quá vừa, cắt tỉa cây,... Với những lý do đó C4.5 đã trở thành thuật toán phổ biến nhất trong những ứng dụng vừa và nhỏ. Quá trình triển khai, cài đặt thử nghiệm cùng với các đánh giá hiệu năng mô hình phân lớp C4.5 đã được tiến hành. Và đã thu được nhiều kết quả có ý nghĩa thực tiễn, cũng như các kết quả gợi mở những hướng nghiên cứu tiếp theo.

SPRINT là một thuật toán tối ưu cho những cơ sở dữ liệu cực lớn. Những ưu điểm của SPRINT là tư tưởng của thuật toán khá đơn giản, có khả năng mở rộng cao, lại rất dễ dàng song song hóa. Do vậy cài đặt và triển khai SPRINT có ý nghĩa khoa học và có khả năng triển khai ứng dụng và đem lại nhiều lợi ích thực tế.

## TÀI LIỆU THAM KHẢO

- [1] Anurag Srivastava, Eui- Hong Han, Vipin Kumar, Vineet Singh. *Parallel Formulations of Decision-Tree Classification Algorithm*. Kluwer Academic Publisher, 1999.
- [2] Anurag Srivastava, Vineet Singh, Eui- Hong (Sam) Han, Vipin Kumar. *An Efficient, Scalable, Parallel Classifier for Data mining*.
- [3] Girija J. Narlikar. *A Parallel, Multithreaded Decision Tree Builder*. CMU-CS-98-184. [reports-archive.adm.cs.cmu.edu/ anon/1998/CMU-CS-98-184.pdf](http://reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-184.pdf)
- [4] Henrique Andrade, Tahsin Kurc, Alan Sussman, Joel Saltz. *Decision Tree Construction for Data Mining on Cluster of Shared-Memory Multiprocessors*. <http://citeseer.csail.mit.edu/178359.html>
- [5] Ho Tu Bao, Chapter 3: *Data mining with Decision Tree* – <http://www.netnam.vn/unescocourse/knowledge/knowledge.htm>
- [6] John Darlington, Moustafa M. Ghanem, Yike Guo, Hing Wing To. *Performance Model for Coordinating Parallel Data Classification*
- [7] John Shafer, Rakesh Agrawal, Manish Mehta. *SPRINT- A Scalable Parallel Classifier for Data mining*. In Proceedings of the 22<sup>nd</sup> International Conference on Very Large Database, India, 1996.
- [8] J. R. Quinlan. *Improvement of Continuous Attribute in C4.5*. In Journal of Artificial Intelligence Research 4 (1996) 77-90
- [9] Manish Mehta, Rakesh Agrawal, Jorma Rissanen. *SLIQ: A Fast Scalable Classifier for Data mining*. IBM Almaden Research Center, 1996.
- [10] Mohammed J. Zaki, Ching-Tien Ho, Rakesh Agrawal. *Parallel Classification for Data Mining on Shared-Memory Multiprocessors*. IBM Almaden Research Center, San Jose, CA 95120.
- [11] Rajeev Rastogi, Kyuseok Shim (Bell Laboratories). *PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning*, 1998. [www.vldb.org/conf/1998/p404.pdf](http://www.vldb.org/conf/1998/p404.pdf)
- [12] Richard Kufrin. *Generating C4.5 Production Rules in Parallel*. In Proceedings of Fourteenth National Conference on Artificial Intelligence, Providence RI, 1997

[www.almaden.ibm.com/software/quest/Publications/papers/vldb96\\_sprint.pdf](http://www.almaden.ibm.com/software/quest/Publications/papers/vldb96_sprint.pdf)

[13] Ron Kohavi, J. Ross Quinlan. *Decision Tree Discovery*, 1999

[14] The Morgan Kaufmann Series in Data Management Systems, Jim Gray. *Datamining- Concepts and Techniques, Chapter 7-Classification and Prediction*. Series Editor Morgan Kaufmann Publishers, August 2000