

Chương 1: Khái niệm về khai thác dữ liệu

1. Giới thiệu

Việc khai thác dữ liệu thường được mô tả như một quá trình lấy các thông tin có giá trị, xác thực từ những cơ sở dữ liệu lớn. Nói cách khác, việc khai thác dữ liệu bắt nguồn từ các dạng mẫu và khuynh hướng tồn tại trong dữ liệu. Các mẫu và khuynh hướng này có thể được gom lại với nhau và được định nghĩa như là một mô hình khai thác. Các mô hình này có thể được áp dụng cho các kịch bản nghiệp vụ riêng biệt như:

- Dự đoán việc bán hàng.
- Chuyển thư đến các khách hàng được chỉ định.
- Xác định các sản phẩm nào có khả năng được bán với nhau.
- Tìm các trình tự mà khách hàng chọn các sản phẩm.

Một khái niệm quan trọng là xây dựng mô hình khai thác là một phần của một tiến trình lớn hơn bao gồm từ việc xác định các vấn đề cơ bản mà mô hình sẽ giải thích, đến việc triển khai mô hình này vào môi trường làm việc. Tiến trình này có thể được định nghĩa bằng việc triển khai 6 bước cơ bản sau:

Bước 1: Xác định vấn đề.

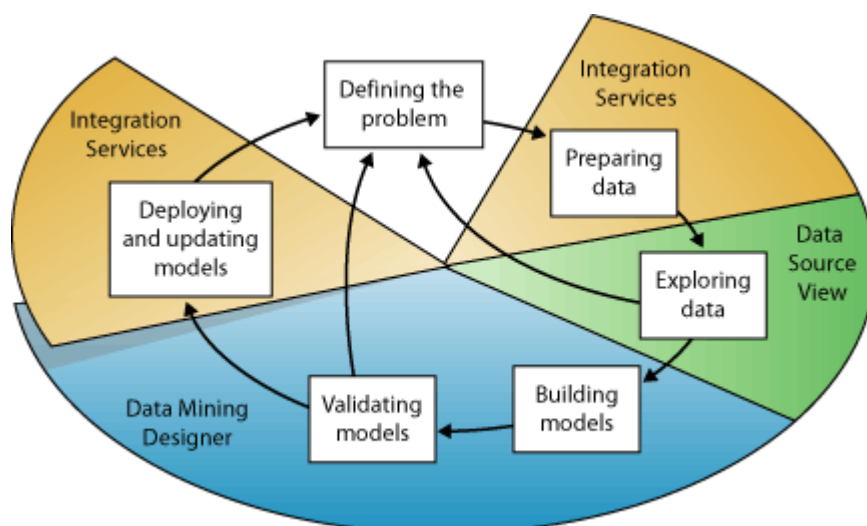
Bước 2. Chỉnh sửa dữ liệu.

Bước 3. Thăm dò dữ liệu.

Bước 4. Xây dựng mô hình.

Bước 5. Thăm dò và thông qua các mô hình.

Bước 6. Triển khai và cập nhật các mô hình. Biểu đồ sau mô tả mối quan hệ giữa mỗi bước trong tiến trình, và có thể sử dụng công nghệ trong Microsoft SQL Server 2005 để hoàn thành từng bước.



Hình 1.1: Mô tả mối quan hệ giữa các bước trong tiến trình

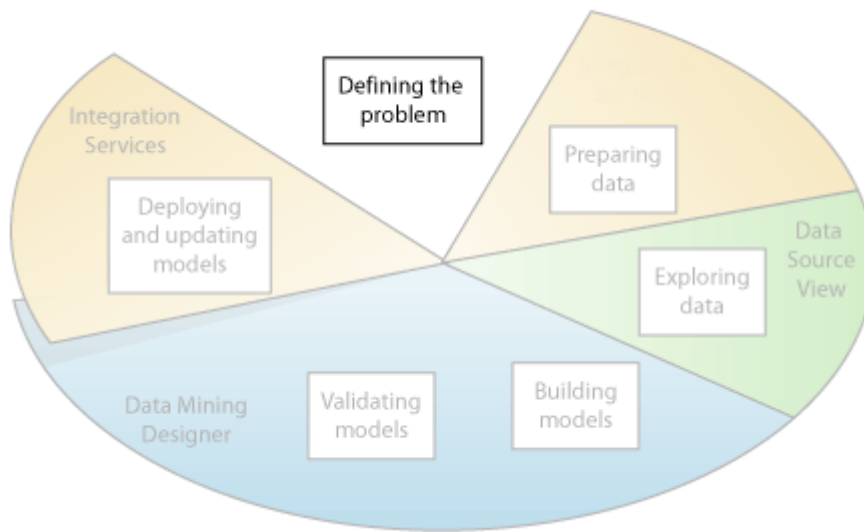
Mặc dù tiến trình được minh họa trong biểu đồ là hình tròn, nhưng mỗi bước không trực tiếp dẫn đến bước tiếp theo. Tạo ra một mô hình khai thác dữ liệu là một tiến trình động và lặp lại. Sau khi thăm dò dữ liệu, có thể nhận ra rằng dữ liệu không đủ để tạo ra mô hình khai thác thích hợp, do đó sẽ phải tìm thêm dữ liệu. Có thể xây dựng nhiều mô hình và nhận ra là chúng không giải quyết được các vấn đề đã đưa ra khi định nghĩa vấn đề, và do đó phải xác định lại vấn đề đó. Có thể cập nhật các mô hình sau khi chúng được triển khai bởi vì nhiều dữ liệu hơn sẽ trở nên hiệu quả. Điều này quan trọng để hiểu rằng tạo ra một mô hình khai thác dữ liệu là một tiến trình, và mỗi bước trong tiến trình có thể được lặp lại nhiều lần khi cần thiết để tạo ra một mô hình tốt.

SQL Server 2005 cung cấp một môi trường hội nhập để tạo ra và làm việc với mô hình khai thác dữ liệu, gọi là Business Intelligence Development Studio. Môi trường này bao gồm các thuật toán khai thác dữ liệu và các công cụ mà làm cho việc xây dựng giải pháp toàn diện cho các dự án khác nhau dễ hơn.

2. Các bước trong tiến trình khai thác dữ liệu

2.1. Xác định vấn đề

Bước đầu tiên trong tiến trình khai thác dữ liệu (được in đậm trong biểu đồ bên dưới (Hình 1.2)), là để xác định rõ ràng các vấn đề nghiệp vụ:



Hình 1.2: Xác định các vấn đề

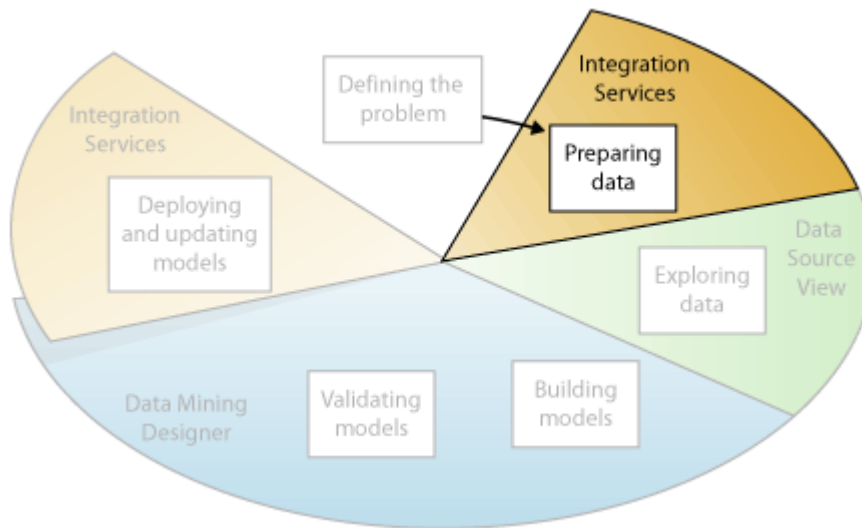
Bước này bao gồm việc phân tích các yêu cầu nghiệp vụ, xác định phạm vi của vấn đề, xác định điểm quan trọng bằng mô hình nào sẽ đánh giá, và xác định mục tiêu cuối cùng cho dự án khai thác dữ liệu. Những công việc này thông dịch thành các câu hỏi như:

- Đang tìm kiếm gì?
- Dự đoán các thuộc tính nào của dataset?
- Đang tìm những dạng quan hệ nào?
- Muốn dự đoán từ mô hình khai thác dữ liệu hay chỉ tìm các dạng mẫu và kết hợp yêu thích.
- Dữ liệu được phân bố như thế nào?
- Các cột liên quan như thế nào, hay nếu có nhiều bảng thì mối quan hệ của chúng như thế nào?

Để trả lời những câu hỏi này, có thể phải tìm hiểu về dữ liệu thực tế, điều tra nhu cầu của người dùng nghiệp vụ cùng với sự quan tâm về dữ liệu thực tế. Nếu dữ liệu không cung cấp được cho nhu cầu người dùng, có thể phải xác định lại dự án.

2.2. Chỉnh sửa dữ liệu

Bước thứ hai trong tiến trình khai thác dữ liệu (được in đậm trong mô hình bên dưới (Hình 1.3)), để củng cố và chỉnh sửa lại dữ liệu được xác định trong bước xác định vấn đề:



Hình 1.3: Chỉnh sửa dữ liệu

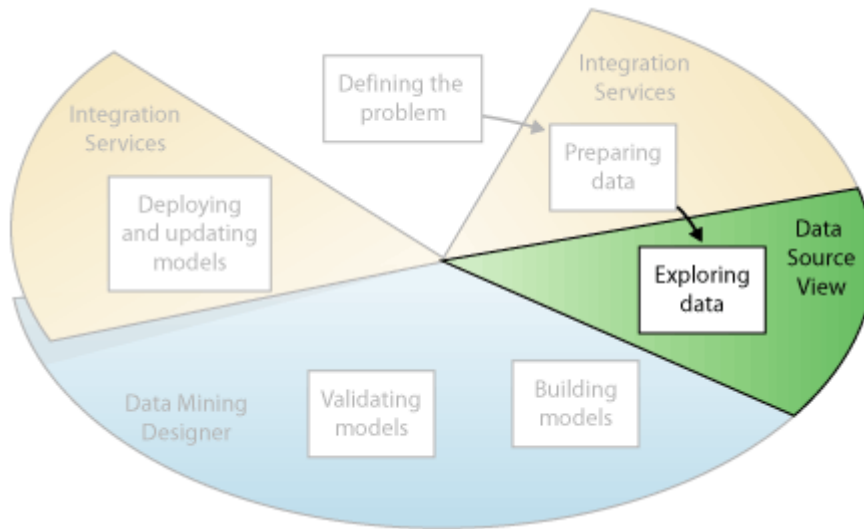
Microsoft SQL Server 2005 Integration Services (SSI) chứa tất cả các công cụ, bao gồm việc thay đổi dữ liệu rõ ràng và vững chắc hơn. Dữ liệu có thể được chứa ở nhiều nơi trong công ty và được định dạng khác nhau, hay có thể có những mâu thuẫn như bị rạn nứt hay mất một số mục nào đó.

Ví dụ: Dữ liệu có thể chỉ ra rằng khách hàng đã mua hàng hóa trước khi khách hàng đó được sinh ra, hay khách hàng đi mua sắm tại cửa hàng cách nhà khoảng 2000 dặm.

Trước khi bạn bắt đầu xây dựng mô hình, phải sửa chữa các vấn đề này. Diễn hình như đang làm việc với một số lượng lớn các dataset và không thể đọc lướt qua tất cả các giao tác. Do đó, phải sử dụng các dạng tự động, như Integration Services, để khảo sát tất cả dữ liệu và tìm ra các mâu thuẫn.

2.3. Khảo sát dữ liệu

Bước thứ ba trong tiến trình khai thác dữ liệu (được in đậm trong mô hình bên dưới (Hình 1.4)) là khảo sát các dữ liệu đã được sửa chữa



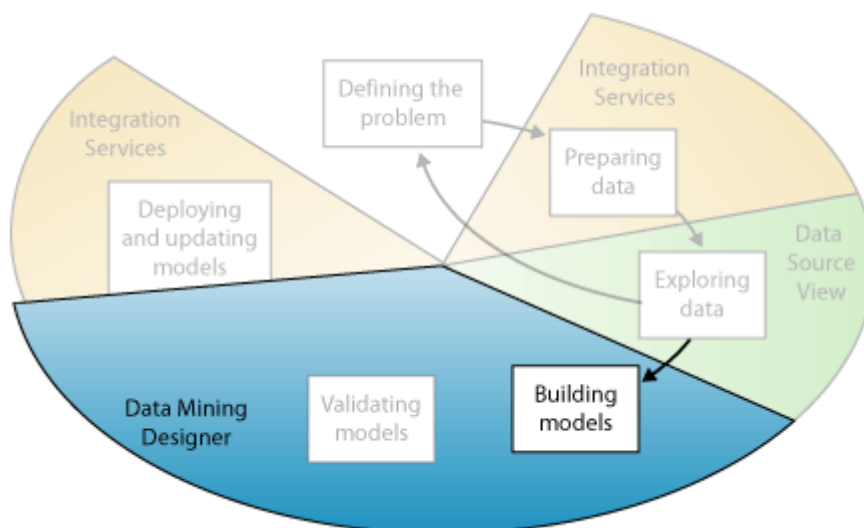
Hình 1.4: Khảo sát dữ liệu

Phải hiểu dữ liệu để đưa ra một quyết định thích hợp khi tạo ra các mô hình. Các kỹ thuật khảo sát bao gồm tính toán các giá trị nhỏ nhất và lớn nhất, tính toán độ trung bình và độ chênh lệch, và nhìn vào thuộc tính của dữ liệu. Sau đó, khảo sát dữ liệu, có thể quyết định xem rằng dataset có chứa các dữ liệu bị rạn nứt hay không, và sau đó có thể nghĩ ra các chiến thuật để giải quyết vấn đề.

Data Source View Designer trong BI Develop Studio chứa nhiều công cụ mà có thể sử dụng để khảo sát dữ liệu.

2.4. Xây dựng mô hình

Bước thứ tư trong tiến trình khai thác dữ liệu (được in đậm trong mô hình bên dưới (Hình 1.5)) để xây dựng mô hình khai thác.



Hình 1.5: Xây dựng mô hình

Trước khi xây dựng mô hình, phải phân chia ngẫu nhiên các dữ liệu đã được sửa chữa thành các dataset thử. Sử dụng các dataset thử này để xây dựng mô hình, và dataset thử này để kiểm tra độ chính xác của mô hình bằng cách ghi lại các query nghi ngờ. Có thể sử dụng Percentage Sampling Transformation trong Integration Services để phân chia dataset.

Sẽ sử dụng kiến thức thu được từ bước khảo sát dữ liệu để giúp cho việc xác định và tạo ra mô hình khai thác. Một mô hình tiêu biểu chứa các cột dữ liệu đưa vào, và các cột xác định, và các cột dự đoán. Có thể xác định những cột này sau đó trong một mô hình mới bằng cách sử dụng ngôn ngữ DataMining Extensions (DMX), hay Data Mining Wizard trong BI Development Studio.

Sau khi xác định cấu trúc của mô hình khai thác, xử lý nó, đưa vào các cấu trúc với các dạng mẫu mô tả mô hình. Điều này được hiểu như là “training” một mô hình. Các mẫu mô hình được tìm thấy bằng cách lướt qua các dữ liệu gốc thông qua các thuật toán. SQL Server 2005 chứa các thuật toán khác nhau cho mỗi dạng của mô hình mà thường xây dựng. Có thể sử dụng các tham số để điều chỉnh từng thuật toán.

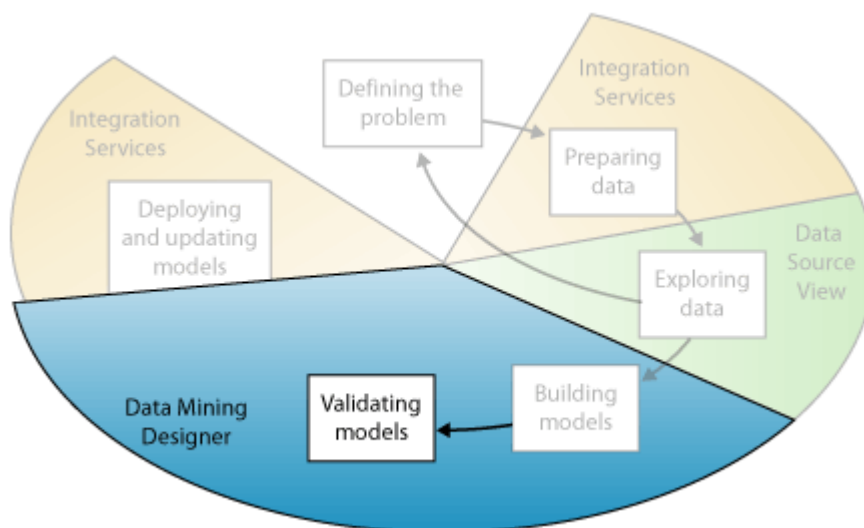
Mô hình khai thác được xác định bằng các đối tượng cấu trúc khai thác dữ liệu, đối tượng mô hình khai thác dữ liệu, và thuật toán khai thác dữ liệu.

Microsoft SQL Server 2005 Analysis Services (SSAS) bao gồm các thuật toán sau:

- Microsoft Decision Trees Algorithm
- Microsoft Clustering Algorithm.
- Microsoft Naive Bayes Algorithm.
- Microsoft Association Algorithm.
- Microsoft Sequence Clustering Algorithm.
- Microsoft Time Series Algorithm.
- Microsoft Neural Network Algorithm (SSAS).
- Microsoft Logistic Regression Algorithm.
- Microsoft Linear Regression Algorithm.

2.5. Khảo sát và thông qua các mô hình

Bước thứ năm trong tiến trình khai thác dữ liệu (được in đậm trong mô hình bên dưới (Hình 1.6)) để khảo sát các mô hình mà xây dựng và kiểm tra hiệu quả của chúng.



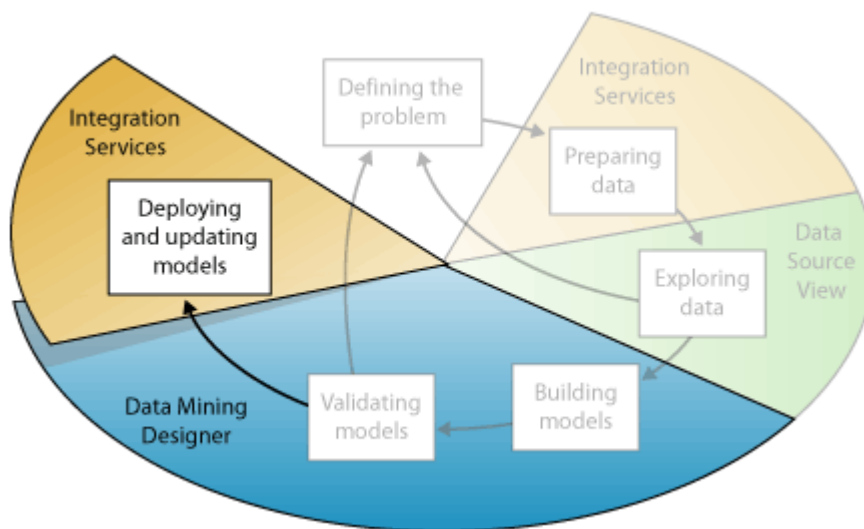
Hình 1.6: Khảo sát và thông qua mô hình

Không muốn đưa một mô hình vào môi trường sản xuất mà chưa có sự kiểm tra hoạt động của nó. Ngoài ra ta có thể đã tạo ra nhiều mô hình và sẽ phải quyết định mô hình nào sẽ thi hành tốt nhất. Nếu không có mô hình nào tạo ra trong bước xây dựng mô hình sản xuất tốt, sẽ phải trở lại bước trước đó trong tiến trình, hay có thể phải xác định lại vấn đề hay phải nghiên cứu lại dữ liệu trong dataset gốc.

Có thể khảo sát các khuynh hướng và các mẫu mô hình mà các thuật toán tìm ra bằng cách sử dụng cái nhìn tổng quan trong Data Mining Designer trong BI Development Studio. Cũng có thể kiểm tra các mô hình này tạo ra dự đoán tốt như thế nào bằng cách sử dụng các công cụ trong designer như lift chart và classification matrix. Những công cụ này yêu cầu các dữ liệu thử mà phân chia từ dataset gốc trong bước xây dựng mô hình.

2.6. Triển khai và cập nhật các mô hình

Bước cuối cùng trong tiến trình khai thác dữ liệu (được in đậm trong mô hình bên dưới (Hình 1.7)) để triển khai vào môi trường sản xuất các mô hình đã hoạt động tốt nhất.



Hình 1.7: Triển khai và cập nhật mô hình

Sau khi các mô hình khai thác tồn tại trong môi trường sản xuất, có thể thực thi nhiều công việc dựa trên nhu cầu. Sau đây là một vài công việc có thể thi hành:

- Sử dụng các mô hình để tạo các dự đoán, mà có thể sử dụng sau đó để tạo ra các quyết định nghiệp vụ. SQL Server cung cấp ngôn ngữ DMX mà có thể dùng để tạo ra các query dự đoán, và Prediction Query Builder để giúp xây dựng các query.
- Đưa chức năng khai thác dữ liệu trực tiếp vào ứng dụng. Có thể bao gồm Analysis Management Objects (AMO) hay một assembly bao gồm việc thiết lập các đối tượng mà ứng dụng có thể sử dụng để tạo, thay đổi, xử lý và xóa các cấu trúc khai thác và mô hình khai thác. Như một sự lựa chọn, có thể gửi XML cho Analysis (XMLA) các mẫu tin trực tiếp đến Analysis Service.
- Sử dụng Integration Service để tạo ra các đóng gói mà trong đó mô hình khai thác được sử dụng để phân chia thông minh các dữ liệu nguồn vào thành nhiều bảng. Ví dụ, nếu một cơ sở dữ liệu tiếp tục được cập nhật với các khách hàng tiềm năng, có thể sử dụng mô hình khai thác với Integration Services để phân chia dữ liệu đầu vào khách hàng, người chi trả cho các sản phẩm và những khách hàng dường như không chi trả cho các sản phẩm.
- Tạo báo cáo để người dùng trực tiếp nêu query với mô hình khai thác tồn tại.

Cập nhật mô hình là một phần trong chiến lược triển khai. Khi dữ liệu nhập vào tổ chức càng nhiều thì phải xử lý lại các mô hình, bằng cách đó sẽ cải thiện hiệu quả của chúng.

Chương 2: Các thuật toán khai thác dữ liệu

1. Giới thiệu chung

Thuật toán khai thác dữ liệu là một kỹ thuật để tạo ra các mô hình khai thác. Để tạo ra một mô hình, một thuật toán đầu tiên phải phân tích thiết lập của dữ liệu, tìm kiếm các mẫu đặc trưng và xu hướng. Thuật toán sau đó sử dụng những kết quả của việc phân tích này để xác định các tham số của mô hình khai thác.

Mô hình khai thác mà một thuật toán tạo ra có thể có nhiều dạng khác nhau, bao gồm:

- Việc thiết lập các luật mô tả làm cách nào các sản phẩm được gom nhóm lại với nhau thành một thao tác.
- Cây quyết định dự đoán một khách hàng cụ thể sẽ mua một sản phẩm hay không.
- Mô hình toán học dự đoán việc mua bán.
- Thiết lập các nhóm mô tả các case trong dataset liên quan đến nhau như thế nào.

Microsoft SQL Server 2005 Analysis Services (SSAS) cung cấp nhiều thuật toán cho các giải pháp khai thác dữ liệu của bạn. Các thuật toán này là tập con của tất cả các thuật toán có thể được dùng cho việc khai thác dữ liệu. Bạn cũng có thể sử dụng các thuật toán của hãng thứ ba tuân theo các đặc tả OLE DB for Data Mining.

2. Giới thiệu các thuật toán:

Microsoft khi phát triển SQL Server 2005 AS, họ đã hoàn thiện các thuật toán thường sử dụng trong DataMining 1 cách hoàn chỉnh nhất so với SQL Server 2000 AS, bao gồm : MS(Microsoft) Decision Tree, MS Clustering, MS Naïve Bayes, MS Time Series, MS Association, MS Sequence Clustering, MS Neural Network, MS Linear Regression, MS Logistic Regression .

Việc ứng dụng các thuật toán này ra sao sẽ được trình bày ở phần sau.

2.1 Microsoft Decision Tree:

Thuật toán Microsoft Decision Tree hỗ trợ cả việc phân loại và hồi quy, và tạo rất tốt các mô hình dự đoán. Sử dụng thuật toán này có thể dự đoán cả các thuộc tính rời rạc và liên tục.

Trong việc xây dựng mô hình, thuật toán này sẽ khảo sát sự ảnh hưởng của mỗi thuộc tính trong tập dữ liệu và kết quả của thuộc tính dự đoán. Và tiếp đến nó sử dụng các thuộc tính input (với các quan hệ rõ ràng) để tạo thành 1 nhóm phân hoá gọi là các node. Khi 1 node mới được thêm vào mô hình, 1 cấu trúc cây sẽ được thiết lập. Node đỉnh của cây sẽ miêu tả sự phân tích (bằng thống kê) của các thuộc tính dự đoán thông qua các mẫu. Mỗi node thêm vào sẽ được tạo ra dựa trên sự sắp xếp các trường của thuộc tính dự đoán, để so sánh với dữ liệu input. Nếu 1 thuộc tính input được coi là nguyên nhân của thuộc tính dự đoán (to favour one state over another), 1 node mới sẽ thêm vào mô hình. Mô hình tiếp tục phát triển cho đến lúc không còn thuộc tính nào, tạo thành 1 sự phân tách(split) để cung cấp một dự báo hoàn chỉnh thông qua các node đã tồn tại. Mô hình đòi hỏi tìm kiếm một sự kết hợp giữa các thuộc tính và trường của nó, nhằm thiết lập một sự phân phối không cân xứng giữa các trường trong thuộc tính dự đoán. Vì thế cho phép dự đoán kết quả của thuộc tính dự đoán một cách tốt nhất.

2.2 Microsoft Clustering:

Thuật toán này sử dụng kỹ thuật lặp để nhóm các bản ghi từ 1 tập hợp dữ liệu vào một liên cung cùng có đặc điểm giống nhau. Sử dụng liên cung này có thể khám phá dữ liệu, tìm hiểu về các quan hệ đã tồn tại, mà các quan hệ này không dễ dàng tìm được một cách hợp lý thông qua quan sát ngẫu nhiên. Thêm nữa, có thể dự đoán từ các mô hình liên cung đã được tạo bởi thuật toán.

Ví dụ : Xem xét một nhóm người sống ở cùng một vùng, có cùng một loại xe, ăn cùng một loại thức ăn và mua cùng một sản phẩm. Đây là một liên cung của dữ liệu, một liên cung khác có thể bao gồm những người cùng đến một nhà hàng, cùng mức lương, và được đi nghỉ ở nước ngoài 2 lần trong năm. Hãy quan sát những liên cung này được phân phối ra sao? Ta có thể biết rõ hơn sự ảnh hưởng của các bản ghi

trong 1 tập hợp dữ liệu. Cũng như sự ảnh hưởng này có ảnh hưởng gì đến kết quả của thuộc tính dự đoán?

2.3 Microsoft Naïve Bayes :

Thuật toán này xây dựng mô hình khai thác nhanh hơn các thuật toán khác, phục vụ việc phân loại và dự đoán. Nó tính toán khả năng có thể xảy ra trong mỗi trường hợp lệ của thuộc tính input, gán cho mỗi trường 1 thuộc tính có thể dự đoán. Mỗi trường này có thể sau đó được sử dụng để dự đoán kết quả của thuộc tính dự đoán dựa vào những thuộc tính input đã biết. Các khả năng sử dụng để sinh ra các mô hình được tính toán và lưu trữ trong suốt quá trình xử lý của khối lập phương (cube: các mô hình được dựng lên từ các khối lập phương). Thuật toán này chỉ hỗ trợ các thuộc tính hoặc là rời rạc hoặc liên tục, và nó xem xét tất cả các thuộc tính input độc lập. Thuật toán này cho ta 1 mô hình khai thác đơn giản (có thể được coi là điểm xuất phát của DataMining), bởi vì hầu như tất cả các tính toán sử dụng trong khi thiết lập mô hình, được sinh ra trong xử lý của cube (mô hình kích thước hợp nhất), kết quả được trả về nhanh chóng. Điều này tạo cho mô hình 1 lựa chọn tốt để khai phá dữ liệu khám phá các thuộc tính input được phân bố trong các trường khác nhau của thuộc tính dự đoán như thế nào?

2.4 Microsoft Time Series : (chuỗi thời gian)

Thuật toán này tạo ra những mô hình được sử dụng để dự đoán các biến tiếp theo từ OLAP và các nguồn dữ liệu quan hệ.

Ví dụ : Sử dụng thuật toán này để dự đoán bán hàng và lợi nhuận dựa vào các dữ liệu quá khứ trong 1 cube .

Sử dụng thuật toán này có thể chọn 1 hoặc nhiều biến để dự đoán (nhưng các biến là phải liên tục). Có thể có nhiều trường hợp cho mỗi mô hình. Tập các trường hợp xác định vị trí của 1 nhóm, như là ngày tháng khi xem việc bán hàng thông qua vài tháng hoặc vài năm trước.

Một trường hợp có thể bao gồm 1 tập các biến (ví dụ như bán hàng tại các cửa hàng khác nhau). Thuật toán này có thể sử dụng sự tương quan của thay đổi biến số (cross-variable) trong dự đoán của nó.

Ví dụ: Bán hàng trước kia tại 1 cửa hàng có thể rất hữu ích trong việc dự báo bán hàng hiện tại tại những cửa hàng.

2.5 Microsoft Association :

Thuật toán này được thiết kế đặc biệt để sử dụng trong phân tích giỏ thị trường (basket market).

Market basket (chỉ số giỏ thị trường: tức là ta sẽ dùng tất cả các loại hàng hoá đang có trên thị trường (1 siêu thị chẳng hạn) ta nhân giá cả của nó với chỉ số của hàng hoá (ví dụ gạo x 10, thịt x 20...) để tính chỉ số CPI (consumer price index). Nếu chỉ số CPI của ngày hôm nay cao hơn so với ngày hôm qua thì xảy ra lạm phát).

Thuật toán này sẽ xem xét mỗi cặp biến/giá trị (như là sản phẩm/xe đạp) là 1 item. 1 Itemset là 1 tổ hợp các item trong 1 transaction đơn lẻ. Thuật toán sẽ lướt qua tập hợp dữ liệu để cố gắng tìm kiếm các itemset nhằm vào việc xuất hiện trong nhiều transaction. Tham chiếu Support sẽ định nghĩa có bao nhiêu transaction mà itemset sẽ xuất hiện trước khi nó được cho là quan trọng.

Ví dụ: 1 itemset phổ biến có thể gồm {Gender="Male", Marital Status = "Married", Age="30-35"}. Mỗi itemset có 1 kích thước là tổng số của mỗi item mà nó có (ở ví dụ này là 3).

Thường thì những mô hình kết hợp làm việc dựa vào các tập dữ liệu chứa các bảng ẩn, như kiểu một danh sách khách hàng ẩn (nested) theo sau là 1 bảng mua bán. Nếu 1 bảng ẩn tồn tại trong tập dữ liệu, mỗi khoá ẩn (như 1 sản phẩm trong bảng mua bán) được xem như 1 item.

Thuật toán này cũng tìm các luật kết hợp với các Itemset. Một luật trong 1 mô hình kết hợp kiểu như $A, B \Rightarrow C$ (kết hợp với 1 khả năng có thể xảy ra). Khi tất cả A, B, C là những Itemset phổ biến. Dấu " \Rightarrow " nói rằng C được dự đoán từ A và B. Khả

năng giới hạn là 1 biến mà xác định khả năng nhỏ nhất tức là khi 1 luật có thể được xét đến. Khả năng này cũng được gọi là 1 “sự tin cậy” trong văn phong DataMining.

Mô hình kết hợp rất hữu ích trong cross-sell và collaborative-filtering .

Ví dụ : Bạn có thể sử dụng mô hình kết hợp để dự đoán các hạng mục mà khách hàng muốn mua dựa vào các danh mục hàng hoá khác trong basket của họ.

2.6 Microsoft Sequence Clustering:

Thuật toán này phân tích các đối tượng dữ liệu có trình tự, các dữ liệu này bao gồm 1 chuỗi các giá trị rời rạc. Thường thì thuộc tính trình tự của 1 chuỗi ảnh hưởng tới 1 tập các sự kiện của 1 trật tự rõ ràng. Bằng cách phân tích sự chuyển tiếp giữa các tình trạng của 1 chuỗi, thuật toán có thể dự đoán tương lai trong các chuỗi có quan hệ với nhau. Thuật toán này là sự pha trộn giữa thuật toán chuỗi và thuật toán liên cung. Thuật toán nhóm tất cả các sự kiện phức tạp với các thuộc tính trình tự vào 1 phân đoạn dựa vào sự giống nhau của những chuỗi này. Một đặc trưng sử dụng chuỗi sự kiện cho thuật toán này là phân tích khách hàng web của 1 cổng thông tin (portal site). 1 Cổng thông tin là 1 tập các tên miền liên kết như: tin tức, thời tiết, giá tiền, mail, và thể thao.. . Mỗi khách hàng được liên kết với 1 chuỗi các click web trên các tên miền này. Thuật toán này có thể nhóm các khách hàng web về 1 hoặc nhiều nhóm dựa trên kiểu hành động của họ. Những nhóm này có thể được trực quan hoá, cung cấp 1 bản chi tiết để biết được mục đích sử dụng trang web này của khách hàng.

2.7 Microsoft Neural Network:

Trong MS SQL server 2005 AS, thuật toán này tạo các mô hình khai thác hồi quy và phân loại bằng cách xây dựng đa lớp perceptron của các neuron. Giống như thuật toán cây quyết định, đưa ra mỗi tình trạng của thuộc tính có thể dự đoán. Thuật toán này tính toán khả năng có thể của mỗi trạng thái có thể của thuộc tính input. Thuật toán sẽ xử lý toàn thể các trường hợp. Sự lặp đi lặp lại so sánh các dự đoán phân loại của các trường với sự phân loại của các trường đã biết. Sai số từ sự phân loại ban đầu (của phép lặp ban đầu) của toàn bộ các trường hợp được trả về network và được sử dụng để thay đổi sự thực thi của network cho các phép lặp kế theo, v.v.. Có thể sau đó sử dụng những khả năng này để dự đoán kết quả của các thuộc tính dự đoán, dựa trên

thuộc tính input. 1 sự khác biệt chính giữa thuật toán này và thuật toán Cây quyết định là các kiến thức xử lý là những tham số network tối ưu nhằm làm nhỏ nhất các lỗi có thể trong khi cây quyết định tách các luật, mục đích để cực đại hoá thông tin có lợi. Thuật toán này hỗ trợ cả các thuộc tính rời rạc và liên tục.

2.8 Microsoft Linear Regression :

Thuật toán này là 1 thể hiện đặc biệt của thuật toán cây quyết định, thu được bởi vô hiệu hoá sự chia tách (toàn bộ công thức hồi quy được xây dựng trên 1 node gốc). Thuật toán này hỗ trợ quyết định của các thuộc tính liên tục.

2.9 Microsoft Logistic Regression :

Thuật toán này là 1 sự thể hiện đặc biệt của thuật toán neural network, thu được bằng cách loại ra các lớp ẩn. Thuật toán này hỗ trợ quyết định cả thuộc tính liên tục và không liên tục.

Tóm lại :

AS bao gồm những kiểu thuật toán sau:

- Thuật toán phân loại: Dự đoán 1 hoặc nhiều biến rời rạc (không liên tục), dựa trên các thuộc tính trong tập hợp dữ liệu (Microsoft Decision Trees Algorithm).
- Thuật toán hồi quy: Dự đoán 1 hoặc nhiều biến liên tục, kiểu như những lợi nhuận và những tổn thất, dựa trên các thuộc tính khác nhau của tập hợp DL (Microsoft Time Series Algorithm).
- Thuật toán phân đoạn: Chia dữ liệu thành 2 nhóm, hoặc các liên cung, hoặc các danh mục có thuộc tính giống nhau (Microsoft Clustering Algorithm).
- Thuật toán kết hợp: Tìm những sự tương quan giữa các thuộc tính khác nhau trong 1 tập hợp dữ liệu. Ứng dụng phổ biến nhất của loại thuật toán này là tạo ra các luật kết hợp, có thể được dùng trong market basket (Microsoft Association Algorithm).

Thuật toán phân tích tiến trình: Tổng kết những tiến trình thường xảy ra hoặc ít xảy ra trong dữ liệu (Microsoft Sequence Clustering Algorithm).

3. Đưa ra thuật toán :

Chọn một thuật toán đúng để sử dụng cho các nghiệp vụ riêng biệt là một nhiệm vụ khó khăn. Khi ta có thể sử dụng các thuật toán khác nhau để thực thi cùng một nghiệp vụ, mỗi thuật toán tạo ra một kết quả khác nhau, và một vài thuật toán có thể tạo ra nhiều hơn một kết quả.

Ví dụ 1: Có thể sử dụng thuật toán Microsoft Decision Trees không chỉ để dự đoán mà còn là một cách để giảm số lượng cột trong dataset, bởi vì cây quyết định có thể xác định các cột mà không ảnh hưởng đến mô hình khai thác cuối cùng.

Ta cũng không phải sử dụng các thuật toán độc lập trong giải pháp khai thác dữ liệu đơn giản, có thể sử dụng một vài thuật toán để khảo sát dữ liệu, và sau đó sử dụng các thuật toán khác để dự đoán kết quả rời rạc dựa trên dữ liệu này.

Ví dụ 2: Có thể sử dụng thuật toán gom nhóm, nhận ra các mẫu, đưa dữ liệu vào nhóm đồng nhất, và sau đó sử dụng các kết quả để tạo ra mô hình cây quyết định tốt hơn.

Ví dụ 3: Như bằng cách sử dụng thuật toán cây hồi quy để lấy thông tin dự đoán về tài chính, và thuật toán dựa trên luật để thực thi việc khảo sát thị trường.

Các mô hình khai thác có thể dự đoán các giá trị, đưa ra bảng tóm tắt dữ liệu, và tìm ra sự tương quan ẩn. Để giúp cho việc lựa chọn thuật toán cho giải pháp khai thác dữ liệu. Bảng 2.1 dưới đây cung cấp các gợi ý cho việc lựa chọn thuật toán nào cho các công việc cụ thể nào:

Bảng 2.1: Lựa chọn thuật toán cho giải pháp khai thác dữ liệu

Công việc	Thuật toán sử dụng
Dự đoán một thuộc tính rời rạc Ví dụ: Dự đoán người nhận thư của cuộc vận động sẽ mua sản phẩm hay không	Thuật toán Microsoft Decision Trees Thuật toán Microsoft Naïve Bayes Thuật toán Microsoft Clustering Thuật toán Microsoft Neural Network (SSAS)
Dự đoán thuộc tính liên tục Ví dụ: Dự đoán doanh thu năm tiếp theo.	Thuật toán Microsoft Decision Trees Thuật toán Microsoft Time Series
Dự đoán một trình tự. Ví dụ: Thực hiện phân tích một clickstream cho một web site của công ty.	Thuật toán Microsoft Sequence Clustering
Tìm nhóm của những mục chọn (item) trong các các giao tác (transaction). Ví dụ: Sử dụng phân tích thị trường để đưa thêm các sản phẩm cho khách hàng	Thuật toán Microsoft Association Thuật toán Microsoft Decision Trees
Tìm những mục (item) giống nhau. Ví dụ: Phân chia các dữ liệu vào các nhóm để hiểu dễ hơn các mối quan hệ giữa các thuộc tin	Thuật toán Microsoft Clustering Thuật toán Microsoft Sequence Clustering

Chương 3: Microsoft Association Rules

1. Giới thiệu về Microsoft Association Rules

Nếu đặt ta vào vai trò là người quản lý của siêu thị. Một trong những trách nhiệm ta là đảm bảo rằng phải bán được một số lượng rất lớn sản phẩm. Mục tiêu chính là việc bán được nhiều hơn và mang lại nhiều lợi nhuận hơn so với những người quản lý khác có cùng vị trí. Hiểu được nhu cầu mua sắm của khách hàng là bước đầu tiên để đạt được mục tiêu này.

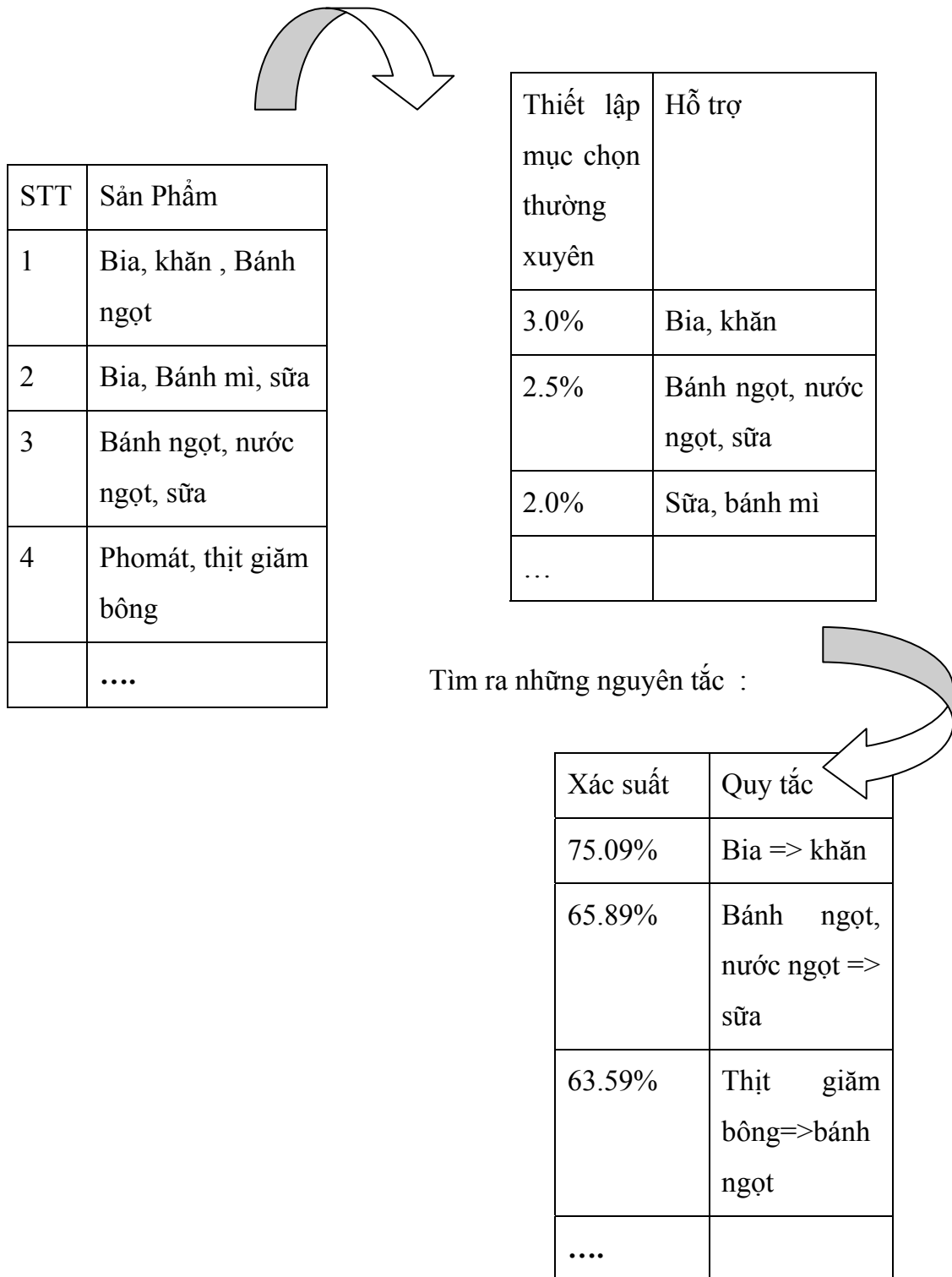
Sử dụng thuật toán luật kết hợp để thực hiện phân tích giỏ hàng trên sự giao dịch của khách hàng, có thể biết được những sản phẩm nào thường được bán cùng với nhau và làm thế nào một sản phẩm đặc biệt được bán cùng với những sản phẩm khác. Chẳng hạn, có thể thấy rằng 5% trong số những khách hàng mua cà *ketchup*, dưa chua(*pickles*), cùng với *hotdogs*, và 75% của những khách hàng này đã mua *ketchup* và *hot dogs* thì cũng mua dưa chua. Hiện tại với những thông tin này ta có thể nắm được công việc. Ta có thể thay đổi cách bố trí để bán được nhiều hàng hơn. Ta có thể dùng sự hiểu biết của mình để quản lý cấp độ của hàng hóa. Ta có thể xác định liệu dưa chua, *hot dogs* và cà *ketchup* để sẵn trong giỏ có nhiều lợi nhuận hoặc ít lợi nhuận hơn khi không xếp chúng sẵn trong giỏ. Nếu mang lại lợi nhuận nhiều hơn, ta có thể thực hiện một chương trình đặc biệt để khuyến khích mua những loại mặt hàng này.

Thêm vào đó, có thể ta muốn hiểu rõ hơn về những khách hàng của cửa hàng mình. Với thẻ ưu đãi, ta có thể rút trích ra được một vài thông tin của khách hàng. Ta có thể biết được rằng khoảng 15% khách hàng nữ của bạn có thẻ ưu đãi, 75% những khách hàng này cho thuê nhà của họ và dọn đến ở gần cửa hàng. Trong khi những mẫu hàng có thể có nguồn gốc từ truy vấn SQL chuẩn, nên có sự ghi nhận hàng trăm hoặc hàng ngàn câu truy vấn để thăm dò đến tất cả những sự kết hợp của hàng hóa có thể xảy ra. Kiểu dữ liệu thăm dò này được tạo ra một cách dễ dàng với thuật toán kết hợp.

2. Nguyên tắc của Microsoft Association Rules

Thuật toán kết hợp chỉ là một phương tiện đếm tương quan. *The Microsoft Association Algorithm* liên quan đến *priori association family* (họ ưu tiên kết hợp), nó là thuật toán rất phổ biến và hiệu quả trong việc tìm kiếm các danh mục phổ biến (việc thiết lập những giá trị thuộc tính phổ biến). Có 2 bước trong thuật toán kết hợp, ví dụ minh họa ở hình 3.1. Bước đầu tiên của thuật toán, là một giai đoạn tính toán chuyên sâu, để tìm kiếm các danh mục phổ biến (*find frequent itemsets*). Bước thứ hai là tạo ra luật kết hợp trên danh mục phổ biến. Bước này đòi hỏi ít tốn thời gian hơn bước đầu.

Finding frequent itemsets (Việc tìm những danh mục phổ biến)



Hình 3.1 : 2 bước thực hiện của thuật toán tìm luật kết hợp

*** Tìm hiểu cơ bản về thuật toán tìm luật kết hợp:**

Trước khi tìm hiểu về nguyên tắc của thuật toán, phần này sẽ giới thiệu một số khái niệm cơ bản về thuật toán kết hợp. Phần tiếp theo trình bày định nghĩa những khái

niệm và những giới hạn mà bạn cần hiểu trước khi thực hiện những thuật toán cơ bản này :

2.1. Itemset

Một itemset là một tập hợp những (*item*) danh mục. Mỗi item là một giá trị thuộc tính. Trong cái giỏ hàng chẳng hạn, một *itemset* chứa một tập hợp (*set*) của những sản phẩm như bánh ngọt, nước ngọt, và sữa. Chẳng hạn khảo sát tỉ mỉ nhân khẩu khách hàng, một *itemset* chứa một tập hợp của những giá trị thuộc tính như {Giới tính = 'nam', trình độ học vấn = 'cử nhân'}. Mỗi *itemset* có một kích thước, là số lượng *item* được chứa trong 1 *itemset*. Kích thước của *itemset* { bánh ngọt, nước ngọt, sữa} là 3.

Frequent itemsets là các tập hợp mục chọn tương đối phổ biến trong *dataset*. Giới hạn thông thường dành cho một itemset được định nghĩa là sử dụng *support*, được nhắc lại trong phần kế tiếp.

Chú ý : Để được chính xác hơn thì sữa, nước ngọt và bánh tất cả đều là những thuộc tính. Những giá trị của nó là ở hệ nhị phân: ở dạng không có (*missing*) hay ở dạng có (*existing*). Chúng ta dùng (sữa, bánh ngọt, nước ngọt) để làm mẫu cho đơn giản, với { Bánh ngọt = có, nước ngọt = có, và sữa = có}.

2.2. Support

Sử dụng *Support* để đánh giá mức độ phổ biến của một *itemset*. *Support* của một *itemset* {A, B} được tạo thành dựa trên tổng số lượng giao dịch của cả A và B.

$$\text{Support}(\{A, B\}) = \text{NumberofTransactions}(A, B)$$

Minimum_Support là một tham số giới hạn mà ta cần chỉ định trước khi xử lý một kiểu kết hợp, nghĩa là chỉ vì ta rất thích những itemset và qui tắc này mà tái hiện lại ít nhất là một lượng nhỏ của dataset hỗ trợ, khác so với luật.

Chú ý: *Minimum_Support* đại diện cho một số trường hợp xuất hiện giới hạn thường xuyên của itemset. Tuy nhiên, nhiều người thấy nó có ích để tạo một giá trị phần trăm thay vì những số đếm được trên thực tế dành cho tham số này. Chẳng hạn, *Minimum_Support*=0.03 có nghĩa rằng giới hạn thường xuyên là 3%. Trong *Microsoft*

Association Rules, nếu một người dùng chỉ định tham số này là số nguyên, thuật toán coi trường hợp thực tế là giới hạn (*threshold*). Nếu một người dùng nhập vào một số float (nhỏ hơn 1.0) cho tham số này, thuật toán coi như nó là giới hạn phần trăm (*percentage*).

2.3. (Probability)xác suất (Confidence)-độ tin cậy

Probability-xác suất là một đặc tính của một quy tắc kết hợp. Xác suất của quy tắc $A \Rightarrow B$ được tính toán sử dụng *support* của itemset $\{A, B\}$ bị chia bởi *support* của $\{A\}$. Xác suất này cũng được gọi là *confidence*-độ tin cậy trong cùng những nghiên cứu của *data mining*.

Nó được miêu tả như sau :

$$\text{Probability}(A \Rightarrow B) = \text{Probability}(B|A) = \text{Support}(A, B) / \text{Support}(A)$$

Minimum_Probability xác suất tối thiểu là một tham số giới hạn mà ta cần chỉ định trước khi tiến hành chạy thuật toán. Nghĩa là chỉ vì người dùng thích thú với những quy tắc mà nó có một xác suất cao hơn xác suất tối thiểu. Xác suất tối thiểu không có tác động trên *itemsets*, nhưng nó có ảnh hưởng đến qui tắc.

Chú ý : Thậm chí chúng ta không đề cập đến xác suất của một *itemset*. Ta có thể sử dụng công thức sau:

$$\text{Probability}(\{A, B\}) = \text{NumberOfTransactions}(A, B) / \text{TotalNumberOfTransactions}$$

Tid	Mặt hàng mua
1	A, B, C
2	A, C
3	A, D
4	B, E, F



Giả sử min support = 50% và min confidence = 50%

Tập phổ biến	Độ tin cậy
{A}	3 = 75%
{B} và {C}	2 = 50%
{D}, {E} và {F}	1 = 25%
{A, C}	2 = 50%
{A, B}, {A, D}, {B, C}, {B, E} và {B, F}	1 = 25%

Chúng ta có luật $A \rightarrow C$ [50%, 66.6%] và $C \rightarrow A$ [50%, 100%]

2.4. Importance(tầm quan trọng)

Importance cũng được coi là một điểm đáng quan tâm hoặc phần nâng cao trong một vài tài liệu. *Importance* có thể dùng để xử lý những *itemset* và những quy tắc. *Importance* của một *itemset* được thể hiện qua công thức sau :

$$\text{Importance}(\{A, B\}) = \text{Probability}(A, B) / (\text{Probability}(A) * \text{Probability}(B))$$

Nếu *importance* = 1, A và B là các *item* độc lập. Có nghĩa là lượng bán của sản phẩm A và lượng bán của sản phẩm B là 2 trường hợp độc lập. Nếu *importance* < 1, thì A và B không tương quan. Nghĩa là nếu một khách hàng mua A, thì không chắc

anh ấy sẽ mua B. Nếu $importance > 1$, thì A và B chắc chắn tương quan với nhau. Điều này có nghĩa là một khách hàng mua A, thì chắc chắn anh ấy cũng sẽ mua B.

$$Importance (A \Rightarrow B) = \log (p(B|A)/p(B|not A))$$

Một $importance = 0$ nghĩa là ở đây không có sự kết hợp giữa A và B. Một điểm $Importance$ xác thực có nghĩa là xác suất của B tăng lên khi A là *true*. Điểm $importance$ không xác thực nghĩa là xác suất của B giảm khi A là *true*.

Bảng 3.1 đưa ra những điểm tương quan của Sandwich và Hamburger được lấy từ một cơ sở dữ liệu mua bán. Mỗi giá trị khối đặc trưng cho số lượng giao dịch. Chẳng hạn, lấy ra 5 trong số 100 giao dịch buôn bán bao gồm một khách hàng mua cả Sandwich và Hamburger.

Bảng 3.1 Đếm sự tương quan của Sandwich và Hamburger

	Hamburger	not Hamburger	TOTAL
Sandwich	5	15	20
Not Sandwich	65	15	80
Total	70	30	100

Trong những điều sau đây, chúng ta sẽ dùng những định nghĩa trước đó để tính toán *Support*, *probability* (xác suất), and *importance* của *itemsets* và những luật liên quan đến Sandwich và Hamburger:

$$Support (\{Hamburger\}) = 70$$

$$Support (\{Sandwich\}) = 20$$

$$Support (\{Hamburger, Sandwich\}) = 5$$

$$Probability (\{Hamburger\}) = 70/100 = 0.7$$

$$Probability (\{Sandwich\}) = 20/100 = 0.2$$

$$Probability (\{Hamburger, Sandwich\}) = 5/100 = 0.05$$

$$Probability (Hamburger | Sandwich) = 5/20 = 0.4$$

$$\text{Probability (Sandwich | Hamburger)} = 5/70 = 0.071$$

$$\text{Importance (\{Hamburger, Sandwich\})} = 0.05 / (0.7 * 0.2) = 0.357$$

Từ *Importance* của *itemset* { Hamburger, Sandwich } = 0.357 < 1, chúng ta có thể thấy rằng Hamburger và Sandwich không tương quan với nhau tức là không xảy ra với một số trường hợp khách hàng vừa mua Hamburger và mua cả Sandwich.

Chú ý: Tạo các tập phổ biến luôn chậm hơn và phải sử dụng support. Việc tạo các luật kết hợp từ các tập phổ biến thì nhanh hơn và phải sử dụng độ tin cậy (confidence).

2.5 Các dạng luật kết hợp

2.5.1 Luật Boolean: luật liên quan đến mỗi kết hợp giữa có xuất hiện và không xuất hiện của các phần tử.

Ví dụ: Khách có mua mặt hàng A hay không mua mặt hàng A?

2.5.2 Luật định lượng: luật có liên quan đến mỗi kết hợp giữa các phần tử hay các thuộc tính định lượng (tuổi, thu nhập, chiều cao, cân nặng v.v...).

2.5.3 Luật một chiều: Các thuộc tính trong luật chỉ qui về một đại lượng.

Ví dụ: Mua Bia, mua Khoai tây → mua Bánh mì

2.5.4 Luật nhiều chiều: Các thuộc tính trong luật qui về hai hay nhiều đại lượng.

Ví dụ: Quốc gia=Pháp => thu nhập =cao [50%,100%]

2.5.5 Luật 1 cấp: Mỗi kết hợp giữa các phần tử hay thuộc tính của cùng một cấp. VD: Bia, Khoai tây chiên → Bánh mì [0.4%,52%]

2.5.6 Luật nhiều cấp: Mỗi kết hợp giữa các phần tử hay thuộc tính của nhiều cấp khác nhau. VD: Bia:Heneiken, Khoai tây chiên → Bánh mì [0.1%,74%]

3. Cách sử dụng Microsoft Association Rules

3.1. Finding Frequent Itemsets (Tìm những itemset phổ biến)

Finding frequent itemset là phần cốt lõi của việc sử dụng thuật toán kết hợp. Trước tiên cần phải chỉ định ngưỡng phổ biến khi sử dụng tham số *minimum_Support*,

ví dụ, $minimum_support = 2\%$. Điều này có nghĩa là ta quan tâm đến việc phân tích riêng những items này khi nó xuất hiện ít nhất là 2% trong những giỏ hàng.

Thuật toán tìm tất cả các *danh mục phổ biến* với $size = 1$ trong lần lặp đầu tiên (những sản phẩm phổ biến này với $support$ thì hay hơn $Minimum_Support$). Thuật toán được thực hiện dựa trên nguyên tắc quét *dataset* và đếm $support$ của mỗi *item* riêng lẻ. Lần lặp thứ hai tìm kiếm những *danh mục* có $size = 2$. Trước khi tiến hành lần lặp lần thứ 2, thuật toán phát sinh một tập hợp những *itemset tham gia* (*candidate*) của 2 $size$ dựa trên kết quả của lần lặp đầu tiên (*itemset phổ biến* có kích thước là 1). Một lần nữa, thuật toán quét *dataset* và đếm $support$ dành cho mỗi *itemset tham gia* được tạo ra. Đến đoạn cuối của quá trình lặp, nó lựa chọn những *itemset tham gia* này với $support$ ít hơn $Minimum_Support$ để lấy danh sách của những *itemset phổ biến* với $size = 2$.

Thuật toán lặp lại một thủ tục tương tự để tìm kiếm những *itemset phổ biến* với kích thước 3, 4, 5...cho đến khi không *itemsets* nào thỏa mãn tiêu chuẩn $Minimum_Support$.

Hình 3.2 Minh họa quá trình của việc xác định những *itemset phổ biến* $Minimum_Support$ được xác lập lên đến 250/1000. Trong lần lặp thứ 1, phomat và bánh ngọt được lọc ra ngoài. Ở lần lặp thứ 2, *itemset tham gia* là { khăn giấy, sữa} bị loại ra. Đến lần lặp thứ 3, *itemset tham gia* là {bia, khăn giấy, bánh mì} có đủ $support$; ngược lại *itemset tham gia* là { bia, sữa, bánh mì } được lọc ra ngoài. Mã giả trình bày sau đây là qui trình chính cho việc tạo ra những *itemset phổ biến*:

F: result set of all frequent itemsets (kết quả tập hợp của những *itemset* thường xuyên)

F[k]: set of frequent itemsets of size k (tập hợp của những *itemset* có kích thước k)

C[k]: set of candidate itemsets of size k (tập hợp những *itemset tham gia* có size là k)

SetOfItemsets generateFrequentItemsets(Integer minimumSupport){

```

F[1] = {frequent items};
for (k=1, F[k] <> 0; k++) {
    C[k+1] = generateCandidates(k, F[k]);
    for each transaction t in databases {
        For each candidate c in C[k+1] {
            if t contains c then c.count++
        }
    } //Scan the dataset.

    for each candidate c in C[k+1] {
        //Select the qualified candidates
        if c.count >= Minimum_Support F[k+1] = F[k+1] U {c}
    }
}

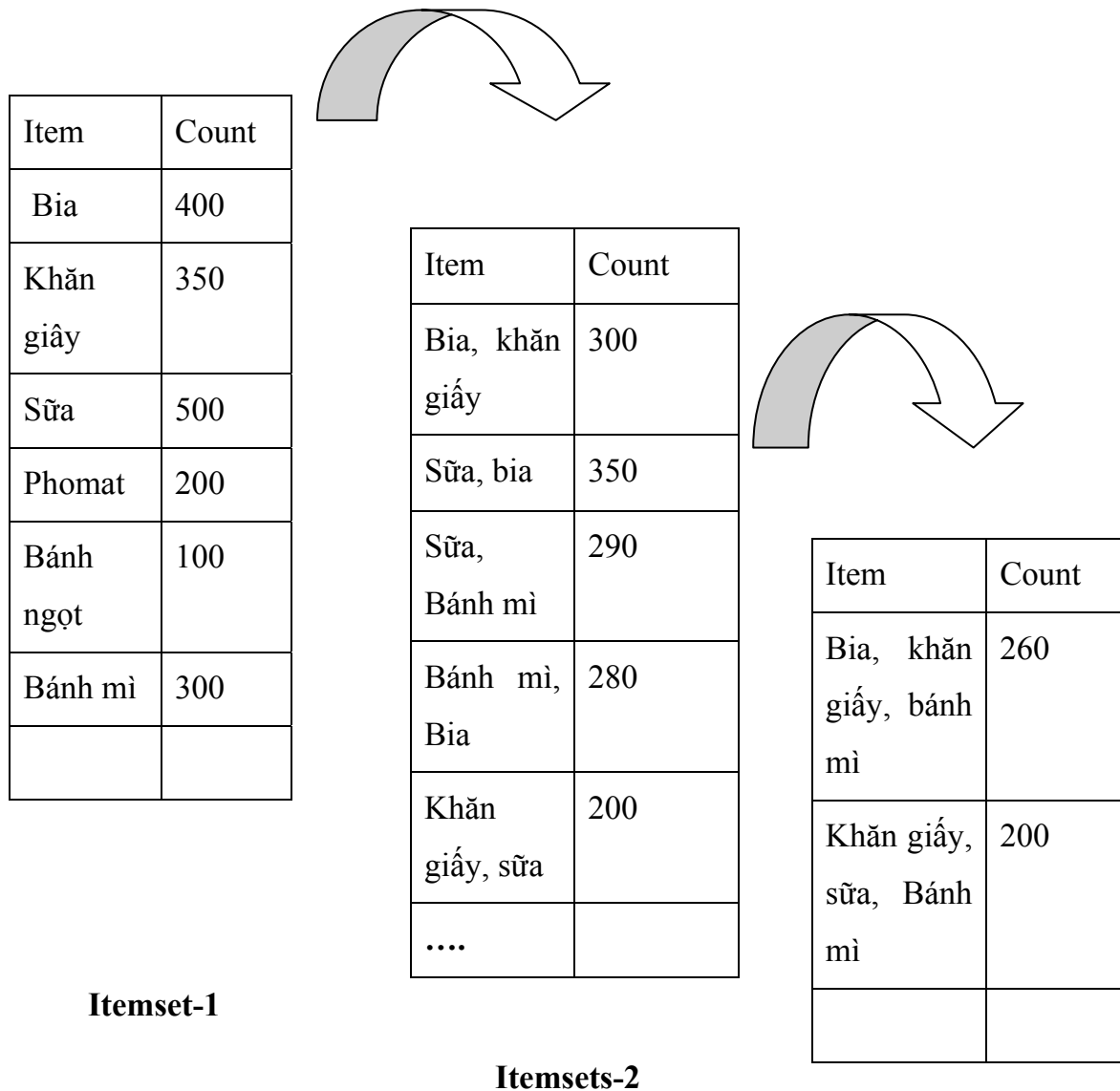
//Union all frequent itemsets of different size
while k >= 1 do {
    F = F U F[k];
    k--;
}

return F;
}

```

Một khi có những *itemset phổ biến*, *generateCandidates* là một hàm trả về tất cả các *itemset tham gia* với size = k+1. Một đặc tính quan trọng của một *itemset phổ biến* là mỗi tập hợp con của nó cũng phải là *itemset thường xuyên*.

Ví dụ: Nếu { bia, khăn giấy, bánh mì } là một *itemset phổ biến*, {bia}, {khăn giấy}, {bánh mì}, {bia, khăn giấy}, {bia, bánh mì}, {khăn giấy, bánh mì} cũng phải là những *itemse phổ biến*.



Mỗi tập hợp con của nó cũng phải là *danh mục phổ biến*.

Hình 3.2 Tìm các danh mục phổ biến

Câu lệnh kết hợp SQL sau đây có thể dùng để tạo ra *itemset* tham gia

Ck+ 1 từ *itemsets* tham gia Fk.

Insert into Ck+1

Select x1.a1, x1.a2, ..., x1.ak, x2.ak

From Fk as x1, Fk as X2

Where

//match the itemset prefixes of size k-1

x1.a1 = x2.a1 And

x1.a2 = x2.a2 And

...

x1.ak-1 = x2.ak-1 And

x1.ak < x2.ak

Câu lệnh SQL này tạo ra các *itemset tham gia* với tiền tố của *itemset size k*. Tuy nhiên, nó không đảm bảo rằng tất cả tập hợp con của *itemsets tham gia* này là *những itemset phổ biến*. Vì vậy, chúng ta cần phải lược bớt những candidate chứa những tập hợp con không phổ biến (*infrequent*) bằng việc sử dụng những thủ tục sau :

Boolean hasInfrequentSubset(Itemset c, SetofItemsets F)

```
{
    For each (k-1) subset s of c {
        If s not in F then return true;
    }
    return false;
}
```

Sự phát sinh và việc đếm tính tương quan của những *itemset tham gia* tốn nhiều thời gian (*time-consuming*). Trong một số trường hợp, nó có thể phát sinh một số lượng khổng lồ của *tập ứng viên*.

Ví dụ : Giả sử có *support* 10,000 sản phẩm (một siêu thị có tầm cỡ trung bình). Nếu *minimum support* đủ thấp, thuật toán sẽ phát sinh trên 10^7 *candidate 2 itemsets*. Nhiều kỹ thuật tối ưu có sẵn trong giai đoạn này, chẳng hạn, *Microsoft Association Rules* cất giữ những *itemset* trong một cấu trúc cây dữ liệu để tiết kiệm bộ nhớ .

Một vài Thuật toán kết hợp phát sinh những *itemset* phổ biến mà không có sự phát sinh của *candidate*.

Chú ý: Xử lý thuật toán kết hợp thì rất dễ làm ảnh hưởng đến tham số *Minimum_Support*. Khi giá trị của nó được thiết lập quá thấp (nhỏ hơn 1%), thời gian xử lý (*processing time*) và yêu cầu bộ nhớ sẽ cấp số mũ lên. Điều này nhờ vào lượng lớn của những *frequent itemset* hạn chế và *frequent itemset candidates*.

Những *dataset* lớn với nhiều *items* riêng biệt, chúng ta nên tránh việc thiết lập những tham số này quá nhỏ.

Số của những *item* cũng quyết định đến sự thực thi của xử lý. Khi ở đây có quá nhiều các *item* độc nhất, gom nhóm chúng thành những loại. Chẳng hạn, khối lượng lưu trữ có thể là 1 tá *JellyBeans* khác, ta có thể nhóm các *Jellybeans* này thành một loại *Jellybeans* đơn. Điều này có thể làm giảm bớt tổng số của các *items* và như vậy làm giảm bớt thời gian xử lý.

3.2 Generating Association Rules (việc tạo ra luật kết hợp)

Bước tiếp theo trong quy trình thuật toán kết hợp là phát sinh luật kết hợp. Ta tìm được luật kết hợp từ: bánh ngọt \geq sữa, và ta quan tâm đến những luật này mà nó có sự tương thích cao. Tạo ra những luật này ta cần đếm *itemset* { bánh ngọt, sữa } cũng như việc đếm bánh ngọt và sữa (*itemsets* 1). Trong trường hợp tổng quát ta cần những *itemset* đến bên trái của mũi tên, với *itemset* dọc theo phía tay trái bao gồm tất cả những *itemsets* trong luật.

Khi những luật được tạo ra từ *itemset*, mỗi *item* trong luật tự động thỏa mãn những điều kiện hỗ trợ tối thiểu. Thủ tục bên dưới phát ra tất cả những luật kết hợp đủ điều kiện:

For each frequent *itemset* f , generate all the subset x and its complimentary set $y = f - x$

If $\text{Support}(f)/\text{Support}(x) > \text{Minimum_Probability}$, then $x \Rightarrow y$ is a qualified association rule with probability = $\text{Support}(f)/\text{Support}(x)$

Thuộc tính tiếp theo sau có thể được sử dụng để làm nhanh tiến trình phát ra luật:

If $a, b, c \Rightarrow d$ has probability lower than the minimum probability, rule $a, b \Rightarrow c, d$ doesn't have enough probability neither.

Chú ý: Xét trên mặt phải của quy luật, *Microsoft Association Algorithm* không tạo ra nhiều *item*. Tuy nhiên nếu ta muốn có nhiều sự khuyến cáo, ta có thể sử dụng một truy vấn dự báo dựa vào mô hình kết hợp, mà có thể trả về nhiều *item*.

3.3 Sự dự đoán

Trong 1 mô hình kết hợp, nếu một cột được dùng cho việc nhập dữ liệu, giá trị của nó chỉ có thể được dùng trong những *itemset phổ biến* và trên mặt trái của luật kết hợp. Nếu một cột được dùng để tạo sự dự đoán, trạng thái của cột có thể được sử dụng trong các *itemset phổ biến* và trên cả mặt trái và phải của luật kết hợp. Nếu một cột là *chỉ dự đoán* (*predict-only*), tình trạng của nó có thể xuất hiện trong các *itemset phổ biến* và trên mặt phải của luật.

Nhiều thuật toán kết hợp trong các *gói khai thác dữ liệu thương mại* ngừng tại việc tìm kiếm các quy luật và các *itemset*: Thuật toán kết hợp *Microsoft* có thể thực hiện những sự dự đoán sử dụng những quy luật này. Kết quả của sự dự đoán thường là 1 tập hợp *item* để giới thiệu. Ta có thể xây dựng một mẫu kết hợp không chỉ dựa vào giỏ hàng mà còn dựa vào nhân khẩu của khách hàng.

Ví dụ: Ta có thể bao gồm giới tính, tình trạng hôn nhân, quyền sở hữu nhà như thuộc tính từng cấp độ trong việc khai thác cấu trúc và bao gồm những giỏ hàng như một bảng lồng nhau trong cùng cấu trúc. Trong trường hợp này, ta phân tích những mẫu hàng mua sắm không chỉ dựa vào mối quan hệ với *itemset* mà còn dựa vào nhân khẩu. Chẳng hạn, có thể tìm thấy một luật dự đoán rằng 65% khách hàng nữ vừa mua bia vừa mua khăn giấy, và 20% khách hàng nam vừa mua khăn giấy vừa mua rượu vang.

Những luật có thể đưa ra sự dự đoán. Cho một khách hàng nam, bạn có thể giới thiệu danh sách các loại rượu. Nếu một khách hàng nam mua bia trong khi mua sắm,

bạn có thể giới thiệu cả rượu và khăn giấy. Tuy nhiên, không phải *itemsets* nào cũng được kết hợp vào luật. Chẳng hạn, không có luật mà có *itemset* { bia, khăn giấy, bánh mì, sữa } trên mặt trái. Danh sách giới thiệu sẽ làm gì cho khách hàng mua bia, khăn giấy, bánh mì và sữa. Ở đây sẽ có một phương thức mà *Microsoft Association Algorithm* sử dụng để thực thi sự dự đoán kết hợp:

- Cho một danh sách *item*, tìm tất cả luật trên mặt trái ứng với việc cho *item* hoặc bất kỳ tập hợp con nào của việc cho *item*. Đưa ra những luật đó để được một danh sách giới thiệu.
- Nếu không có một luật thích hợp nào hay chỉ có vài *item* được giới thiệu để đưa ra những thông tin được biểu hiện bằng con số ở lẽ để dự đoán và trả về *n item* phổ biến nhất.
- Sắp xếp các *item* đó từ bước 1 đến bước 2 dựa vào khả năng có thể xảy ra.

Chú ý: Số của thuật toán kết hợp này dựa vào tham số *Minimum_Probability* (tất nhiên, mỗi item trong một luật phải là một item phổ biến). Chẳng hạn, khi *Minimum_Probability* được đặt tới 30%, điều này có nghĩa là 30% của khách hàng vừa mua A vừa mua B, $A \rightarrow B$ và đây là một luật đủ điều kiện.

3.4 Tham số thuật toán

Thuật toán kết hợp rất nhạy cảm với việc cài đặt tham số thuật toán. Sau đây là danh sách những tham số cho *Microsoft Association Algorithm*.

- *Minimum_Support* là tham số giới hạn. Nó khai báo item yêu cầu hỗ trợ tối thiểu phải thấy đủ điều kiện như một *itemset phổ biến*. Giá trị của nó trong khoảng từ 0 đến 1. Giá trị mặc định là 0.03. Nếu giá trị này được đặt quá thấp. Ví dụ: 0.001 – thuật toán mất nhiều thời gian xử lý và đòi hỏi nhiều bộ nhớ.

Nếu *Minimum_Support* được đặt lớn hơn 1, nó được xem như giới hạn cho một số những trường hợp thay vì phần trăm.

- *Maximum_Support* là tham số giới hạn. Nó xác định một ngưỡng hỗ trợ tối thiểu của *itemset phổ biến*. Giá trị của nó trong khoảng từ 0 đến 1, Giá trị mặc định là 0.001. Tham số này có thể được dùng để lọc ra những *item* hay xảy ra.

Nếu *Maximum_Support* được thiết lập lớn hơn 1, nó được xem như giới hạn cho một số trường hợp thay vì tỷ lệ phần trăm.

- *Minimum_Probability* là tham số giới hạn. Nó xác định khả năng tối thiểu cho một luật kết hợp. Giá trị của nó trong khoảng từ 0 đến 1. Mặc định là 0.4.

- *Minimum_Importance* là tham số giới hạn cho luật kết hợp. Những luật ít quan trọng hơn *Minimum_Importance* được tìm ra.

- *Minimum_Itemset_Size* chỉ rõ kích thước nhỏ nhất của một *itemset*. Mặc định là 0. Đôi khi không cần chú ý đến số lớn của một item nhỏ hơn. Chẳng hạn, có thể chỉ quan tâm trong *itemset* có kích thước lớn hơn 4.

Việc giảm bớt *Minimum_Itemset_Size* sẽ không giảm bớt thời gian tiến trình bởi vì thuật toán phải bắt đầu với *itemset* kích thước 1 và tăng kích thước lên từng bước.

- *Maximum_Itemset_Count* xác định số lớn nhất của các *itemset*. Nếu không được chỉ ra, thuật toán sẽ tạo ra tất cả các *itemset* dựa vào *Minimum_Support*. Tham số này tránh việc tạo ra số lớn nhất của các *itemset*. Khi có quá nhiều *itemset*, thuật toán chỉ giữ top n *itemset* dựa vào số điểm quan trọng của các *itemset*.

- *Optimized_Prediction_Count* được dùng để đặt số các item giới thiệu được hỏi bởi câu truy vấn dự đoán. Mặc định thuật toán sử dụng các luật với chiều dài là 2 cho dự đoán. Có thể tăng số này lên để có chất lượng dự đoán tốt hơn.

3.5 Sử dụng thuật toán

Nguồn gốc của *Microsoft Association Algorithm* và danh sách những tham số điều chỉnh. Xây dựng vài mẫu kết hợp sử dụng thuật toán này.

3.5.1 Truy vấn DMX

Giả sử ta có 2 bảng: *Customer* và *Purchase*. Bảng *Customer* chứa thông tin nhân khẩu khách hàng. Nó bao gồm những thuộc tính như *Gender*, *Age*, *marital status*, *profession*, vv.... Bảng *Purchase* là một bảng thực thi chứa danh sách các movies mỗi khách hàng đã mua trong cửa hàng. Có 2 cột trong bảng *Purchase*: *Customer_ID*

và *Movie_Name*. Xây dựng một mẫu kết hợp để phân tích mối quan hệ quanh movie và nhân khẩu.

Đoạn sau tạo một mẫu về việc phân tích kết hợp sử dụng *Gender*, *Marital_Status* và *purchase movie*:

```
Create Mining Model MovieAssociation (
    Customer_Id long key,
    Gender text discrete predict,
    Marital_Status text discrete predict,
    MoviePurchase table predict (
        Movie_Name text key
    )
)

Using Microsoft_Association_Rules (Minimum_Support = 0.02,
Minimum_Probability = 0.40)
```

Mặc dù hầu hết các giỏ hàng chứa các bảng lồng nhau, nó có thể sử dụng một thuật toán kết hợp để phân tích bảng nguyên nhân cho sự thăm dò dữ liệu đã cải tiến. Sau là một mẫu để phân tích bảng *Customer*, hơn nữa nó giúp bạn khám phá *dataset* và tìm các cách đặt giá trị thuộc tính chung. Thuật toán kết hợp không chấp nhận những thuộc tính liên tục bởi vì *counting engine* mà đếm sự tương quan quanh trạng thái thuộc tính riêng lẻ. Ta cần tạo thuộc tính liên tục trong việc khai thác mẫu riêng lẻ, như minh họa ở đây:

```
Create Mining Model CustomerExploration (
    Customer_Id long key,
    Gender text discrete predict,
    Marital_Status text discrete predict
    Education text discrete predict,
```

Home_Ownership text discrete predict

)

Using Microsoft_Association_Rules (Minimum_Support = 0.05,

Minimum_Probability = 0.75)

Sự trình bày một mô hình huấn luyện phần lớn tùy thuộc vào cấu trúc mô hình, chứ không phụ thuộc vào thuật giải. Sau đây là phần trình bày cho mô hình huấn luyện *MovieAssociation*:

Insert into MovieAssociation (Customer_Id, Gender, Marital_Status,

MoviePurchase (Customer_Id, Movie_Name))

OPENROWSET ('MSDataShape', 'data

provider= SQLOLEDB;Server=myserver;UID=mylogging; PWD=mypass' ,

'Shape

{Select Customer_Id, Gender, Marital_Status From Customers }

Append (

{Select Customer_Id, Movie_Name From Purchases }

Relate Customer_Id to Customer_Id) as MoviePurchase')

Sau khi mô hình được xử lý, ta có thể đưa ra truy vấn để lấy lại các *itemset* và những luật từ nội dung. Ta làm điều này bằng việc đưa ra nội dung trên các kiểu nút cho các luật và các *itemset*, là 7 và 8, theo thứ tự định sẵn:

//retrieving all the frequent itemsets

Select Node_Description from MovieAssociation.Content

Where Node_Type = 7

//retrieving all the rules

Select Node_Description from MovieAssociation.Content

Where Node_Type = 8

Nếu ta chỉ có thông tin nhân khẩu khách hàng và đưa giới thiệu hình ảnh dựa vào *Gender, Marital_Status and Age*, ta có thể sử dụng đoạn truy vấn dự đoán sau:

```
Select t.CustomerID, Predict (MoviePurchase, 5) as Recommendation
```

```
From MovieAssociation
```

```
Natural Prediction Join
```

```
OPENROWSET ('MSDataShape', 'data provider=SQLOLEDB;
```

```
Server=myserver;UID=mylogging; PWD=mypass' ,
```

```
'Select CustomerID, Gender, Marital_Status, Age from NewCustomer')
```

```
as
```

```
t
```

Predict (MoviePurchase, 5) trả về top 5 *movies* trong một cột bảng dựa vào khả năng có thể xảy ra. Kiểu truy vấn này được gọi một sự truy vấn kết hợp.

Đôi khi, ta không những biết về nhân khẩu khách hàng, mà còn biết một ít về *movies* một khách hàng vừa được mua. Ta có thể sử dụng đoạn truy vấn sau để đưa ra nhiều sự giới thiệu chính xác:

```
Select t.CustomerID, Predict (MoviesPurchase, 5) as Recommendation
```

```
From MovieAssociation
```

```
PREDICTION JOIN
```

```
Shape {
```

```
OPENROWSET ('SQLOLEDB',
```

```
'Integrated Security=SSPI; Data Source=localhost;Initial Catalog=
```

```
MovieSurvey',
```

```
'
```

```
Select CustomerID, Gender, Marital_Status, Age
```

```
From Customer Order By CustomerID'))}
```

```
Append ({
```

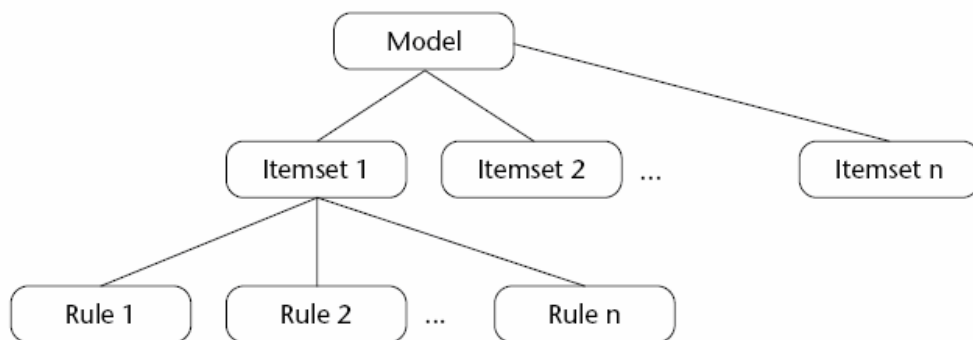
```

OPENROWSET ('SQLOLEDB',
    'Integrated Security=SSPI; Data Source=localhost; Initial Catalog =
    MovieSurvey',
    'Select CustomerID, Movie
    From Movies Order By CustomerID'})}
Relate CustomerID to CustomerID)
As MoviePurchase As t
On
    MovieAssociation.Gender = t.Gender
And    MovieAssociation.Marital_Status = t.Marital_Status
And
    MovieAssociation.MoviesPurchas.Movie_Name=t.MoviePurchase.Movie

```

3.5.2. Nội dung mô hình

Nội dung của một mô hình kết hợp được trình bày trong Hình 3.3.



Hình 3.3: Mô hình tìm luật kết hợp

Có 3 mức độ. Mức top có một nút đơn đại diện cho mô hình. Mức thứ hai chứa những nút đại diện cho những *itemset* đủ điều kiện với sự hỗ trợ kết hợp của chúng. *Distribution rowset* của các nút *itemset* chứa thông tin chi tiết về *itemset*, với mỗi hàng đại diện cho một *item* cá nhân. Mức thứ ba chứa những nút đại diện cho luật đủ điều

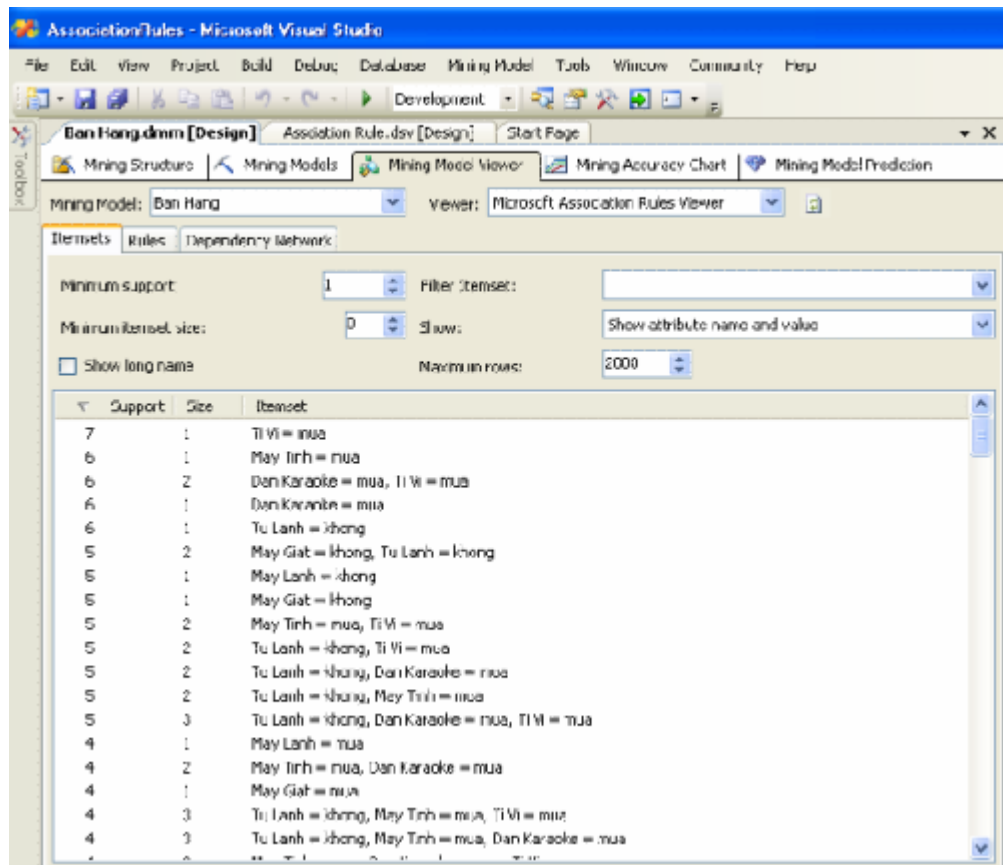
kiện. Cha của những nút luật mà t là *itemset* đại diện cho mặt trái *item* của luật. Mặt phải của luật luôn có một *item* đơn lẻ, mà được lưu trữ trong *Distribution rowset*.

3.5.3. Mô hình phiên dịch (Demo từ CSDL các mặt hàng được bán tại siêu thị điện máy)

Sau khi mô hình kết hợp được xử lý, bạn có thể duyệt nội dung mô hình sử dụng trình *Association viewer*. Trình này chứa 3 tabs: *Itemsets*, *Rules*, *Dependency Net*.

Tab Itemsets (Hình 3.4) trình bày những *itemset* thường dùng được khám phá bởi thuật toán kết hợp. Phần chính của màn hình là một hệ thống trình diễn danh sách các *itemset* phổ biến, kích thước và những *support* của chúng. Đôi khi nếu *Minimum_Support* được đặt quá thấp, có thể có nhiều *itemset*. Đôi khi những danh sách thả xuống thì có thể cho phép bạn lọc ra những *itemset* này *support* và kích thước các *itemset*.

Tab Rules (Hình 3.5) trình bày những luật kết hợp đủ khả năng. Phần chính của tab là hệ thống các luật. Nó trình bày tất cả các luật đủ điều kiện, những điểm quan trọng và có thể của chúng. Điểm quan trọng được thiết kế để đo tác dụng của một luật. Điểm quan trọng càng tăng thì chất lượng của luật càng tốt. Tương tự như *Tab Itemset*, thì *Tab Rules* chứa một vài danh sách thả xuống và các tập tin text cho chọn lọc các luật.



AssociationRules - Microsoft Visual Studio

File Edit View Project Build Debug Database Mining Model Tools Window Community Help

Development

Ban Hang.dmm [Design] Association Rule.dsv [Design] Start Page

Mining Structure Mining Models Mining Model Viewer Mining Accuracy Chart Mining Model Prediction

Mining Model: Ban Hang Viewer: Microsoft Association Rules Viewer

Items Rules Dependency Network

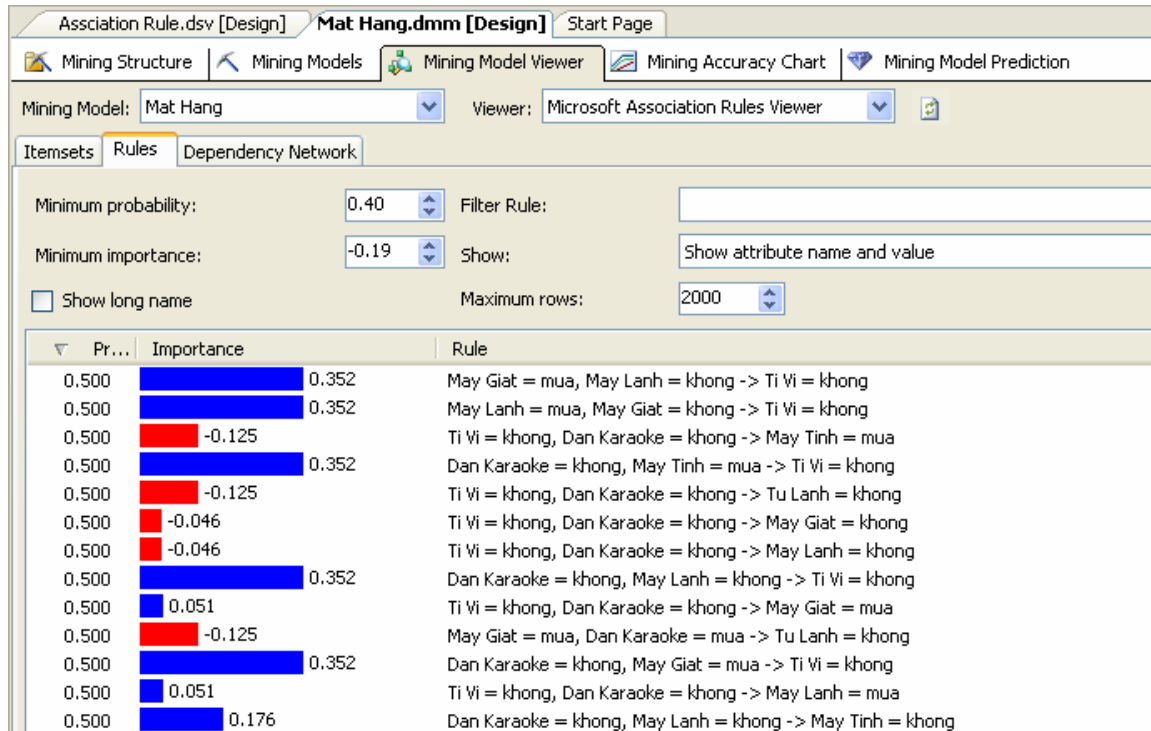
Minimum support: 1 Filter Itemset: Show attribute name and value

Minimum itemset size: 0 Show: 2000

Show long name Maximum rows: 2000

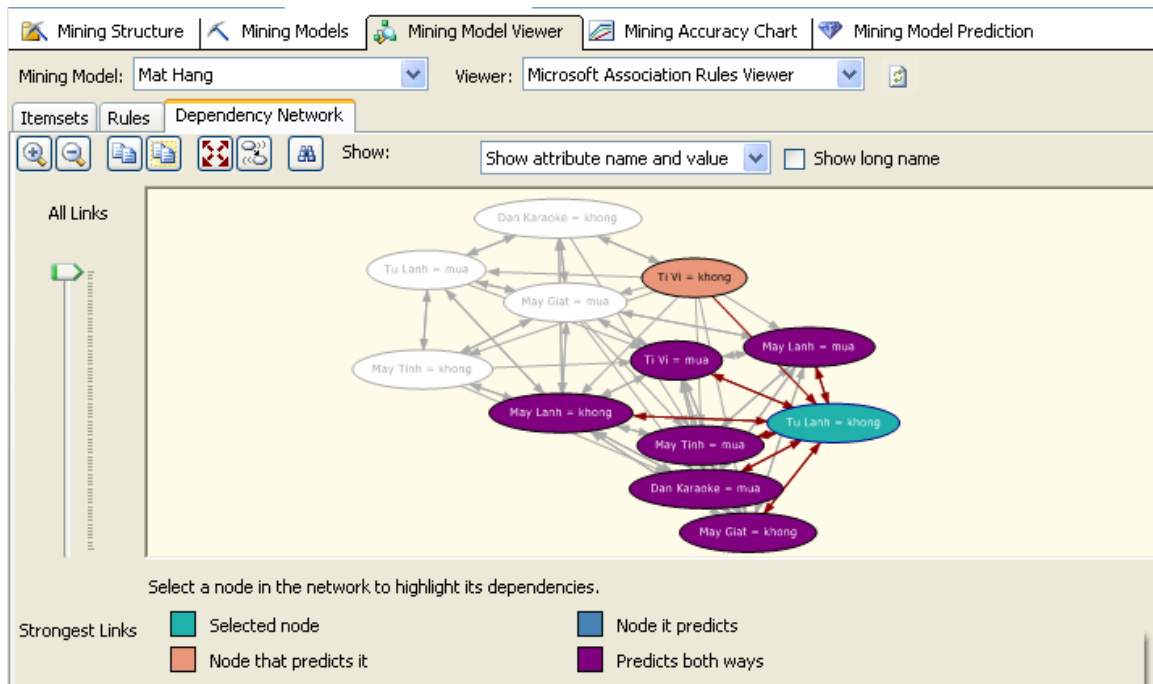
Support	Size	Itemset
7	1	Ti Vi = mua
6	1	Máy tính = mua
6	2	Dan Karaoke = mua, Ti Vi = mua
6	1	Dan Karaoke = mua
6	1	Tủ Lạnh = không
5	2	Máy Giặt = không, Tủ Lạnh = không
5	1	Máy Lạnh = không
5	1	Máy Giặt = không
5	2	Máy Tính = mua, Ti Vi = mua
5	2	Tủ Lạnh = không, Ti Vi = mua
5	2	Tủ Lạnh = không, Dan Karaoke = mua
5	2	Tủ Lạnh = không, Máy Tính = mua
5	3	Tủ Lạnh = không, Dan Karaoke = mua, Ti Vi = mua
4	1	Máy Lạnh = mua
4	2	Máy Tính = mua, Dan Karaoke = mua
4	1	Máy Giặt = mua
4	3	Tủ Lạnh = không, Máy Tính = mua, Ti Vi = mua
4	3	Tủ Lạnh = không, Máy Tính = mua, Dan Karaoke = mua

Hình 3.4. Những itemset phổ biến



Hình 3.5: Những luật có khả năng kết hợp

Tab thứ 3 của sự kết hợp là *Dependency Net viewer* (Hình 3.6). Mỗi nút trong viewer đại diện cho 1 *item*, chẳng hạn. Mỗi đỉnh đại diện cho một cặp luật kết hợp. *Slider* được kết hợp với điểm quan trọng. Mặc định, nó trình diễn trên 60 nút. Ta có thể *add* các nút ẩn vào biểu đồ sử dụng nút *Search* trong *Toolbar*. Và cũng có thể lọc ra những đỉnh yếu sử dụng *slider*. Nếu muốn có nhiều nút và đỉnh trong *dependency net*, ta có thể hạ thấp giá trị của *Minimum_Probability* và xử lý lại mô hình.



Hình 3.6: Mô hình mạng kết hợp

Tóm tắt:

Trong chương này, chúng ta đã có được cái nhìn tổng quan về *Microsoft Association algorithm* và cách sử dụng của nó, ta biết về bộ từ khóa của thuật toán kết hợp bao gồm: *itemset*, *rule*, *support*, *probability*, và *importance* và nguồn gốc của tiến trình thuật toán kết hợp. Có 2 bước trong thuật toán này: khai báo các *danh mục phổ biến* và đưa ra luật. Những luật có thể được sử dụng cho dự đoán.

Ta biết cách truy vấn DMX để sử dụng với mô hình kết hợp. Những truy vấn này đưa ra những giới thiệu dựa trên những cái có thể hoặc có thể điều chỉnh. Kết quả của truy vấn này có thể được dùng trong ứng dụng *cross-selling*./.

Chương IV: Decision Tree

1. Khái niệm cây quyết định:

Cây quyết định là một cây trong đó:

- Nút trong: tên thuộc tính được chọn để phân lớp.
- Nhánh: các giá trị tương ứng của thuộc tính được chọn ở bước đó.
- Nút lá: một nút lá là một nhãn phân lớp hay là một trong các giá trị của thuộc tính kết quả.

Một cây quyết định (decision tree) là một đồ thị mô tả các dự đoán về kết quả có thể xảy ra của sự vật, hiện tượng trong đời sống, từ đó đưa ra những kế hoạch, chiến lược phù hợp nhằm nâng cao hiệu quả công việc. Các cây quyết định được dùng để hỗ trợ quá trình ra quyết định. Cây quyết định là một dạng đặc biệt của cấu trúc cây.

Cây quyết định là một kiểu mô hình dự báo (predictive model), nghĩa là một ánh xạ từ các quan sát về một sự vật/hiện tượng tới các kết luận về giá trị mục tiêu của sự vật/hiện tượng. Mỗi một nút trong (internal node) tương ứng với một biến, đường nối giữa nó với nút con của nó thể hiện một giá trị cụ thể cho biến đó. Mỗi nút lá đại diện cho giá trị dự đoán của biến mục tiêu.

Cây quyết định mô tả một cấu trúc cây, trong đó, các lá đại diện cho các phân loại (thuộc tính xuất) và cành đại diện cho các đường đi của các thuộc tính dẫn tới phân loại đó (thuộc tính dẫn). Quá trình tìm lá được lặp lại một cách đệ qui cho mỗi tập con dẫn xuất. Quá trình đệ qui hoàn thành khi không thể tiếp tục thực hiện việc chia tách được nữa, hay khi một phân loại đơn có thể áp dụng cho từng phần tử của tập con dẫn xuất.

Việc tạo quyết định có rất nhiều ứng dụng ví dụ như hệ thống thư tín của công ty chứa đựng một mô hình mà có thể chính xác tiên đoán thành viên nào của nhóm trực sẽ trả lời cho một yêu cầu nhất định mà họ không cần quan tâm mô hình này hoạt động như thế nào. Trong một số những trường hợp khác khả năng **giải thích cho việc** đưa ra quyết định là vấn đề chính yếu. Trong một số ứng dụng, sự phân loại hay sự tiên đoán là vấn đề hết sức quan trọng.

2. Tổng Quan Về Thuật Toán

Thuật toán Microsoft Decision Tree hỗ trợ cho cả việc phân loại và hồi quy, tạo rất tốt các mô hình dự đoán. Sử dụng thuật toán này có thể dự đoán cả các thuộc tính rời rạc và liên tục.

Trong việc xây dựng mô hình, thuật toán này sẽ khảo sát sự ảnh hưởng của mỗi thuộc tính trong tập dữ liệu và kết quả của thuộc tính dự đoán. Tiếp đến nó sẽ sử dụng các thuộc tính input (các quan hệ rõ ràng) để tạo thành 1 nhóm phân hoá gọi là các node. Khi các 1 node mới được thêm vào mô hình thì 1 cấu trúc cây sẽ được thiết lập. **Node đỉnh của cây sẽ miêu tả sự phân tích** (thống kê) của các thuộc tính dự đoán thông qua các mẫu. Mỗi node thêm vào sẽ được tạo ra dựa trên sự sắp xếp các trường của thuộc tính dự đoán, để so sánh với các dữ liệu input. Nếu 1 thuộc tính input được coi là nguyên nhân của thuộc tính dự đoán thì 1 node mới sẽ thêm vào mô hình. Mô hình tiếp tục phát triển cho đến lúc không còn thuộc tính nào, tạo thành 1 sự phân tách (split) để cung cấp 1 dự báo hoàn chỉnh thông qua các node đã tồn tại. Mô hình đòi hỏi tìm kiếm 1 sự kết hợp giữa các thuộc tính và trường của nó, nhằm thiết lập 1 sự phân phối không cân xứng giữa các trường trong thuộc tính dự đoán. Vì vậy, nó cho phép dự đoán kết quả của thuộc tính dự đoán 1 cách tốt nhất.

Thuật toán Microsoft Decision Trees là một thuật toán phân loại và hồi quy được cung cấp bởi Microsoft SQL Server 2005 Analysis Services (SSAS) sử dụng trong mô hình dự đoán cho cả thuộc tính rời rạc và liên tục.

Đối với thuộc tính rời rạc, thuật toán đưa ra các dự đoán dựa trên các mối quan hệ giữa các cột nhập vào trong dataset. Nó sử dụng các giá trị, trạng thái, các cột của chúng để dự đoán trạng thái cột mà bạn chỉ định hay dự đoán. Đặc biệt, thuật toán nhận biết các cột nhập vào tương quan với cột dự đoán. Ví dụ, trong một kịch bản, để dự đoán những khách hàng nào có khả năng mua xe đạp, nếu có 9 trong số 10 khách hàng trẻ hơn mua xe đạp, trong khi có 2 trong số 10 khách hàng lớn tuổi hơn mua, thuật toán sẽ suy luận ra tuổi dự đoán tốt cho việc mua xe đạp. Cây quyết định tạo ra các dự đoán dựa trên xu hướng đi tới kết quả cụ thể.

Đối với thuộc tính liên tục, thuật toán sử dụng hồi quy tuyến tính để xác định cây quyết định phân chia ở đâu.

Nếu có nhiều hơn một cột được thiết lập để dự đoán, hay nếu dữ liệu nhập vào chứa bảng xếp vào nhau được thiết lập để dự đoán, thuật toán xây dựng cây quyết định riêng biệt cho mỗi cột dự đoán.

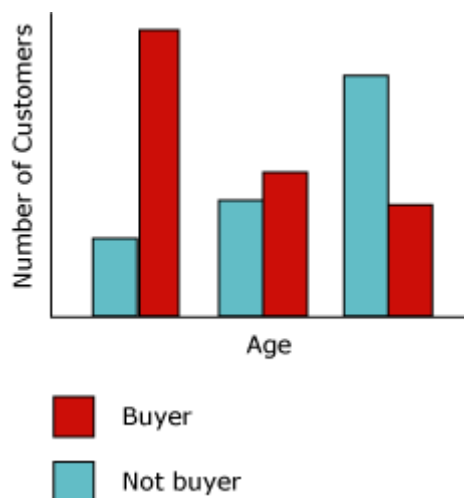
3. Cách thi hành thuật toán trong SQL SERVER 2005

Thuật toán Microsoft Decision Trees xây dựng mô hình khai thác dữ liệu bằng cách tạo ra một loạt đường rẽ, gọi là các node, trên một cây. Thuật toán thêm các node vào mô hình liên tục một cột nhập vào được tìm thấy tương quan đáng kể với cột dự đoán. Cách mà thuật toán xác định đường rẽ khó hay dễ phụ thuộc vào việc dự đoán cột liên tiếp hay rời rạc.

Mô hình dữ liệu tạo cây quyết định để giải quyết vấn đề phải chứa một cột khóa, các cột input và một cột dự đoán. Thuật toán Microsoft Decision Trees cung cấp các dạng nội dung cột input, các dạng nội dung cột dự đoán, và các modeling lag.

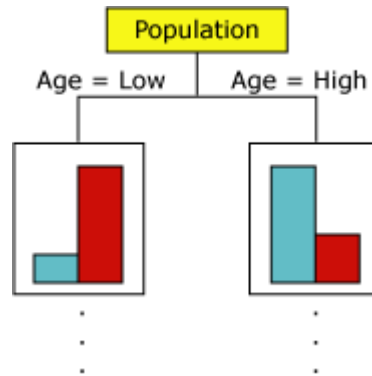
3.1. Thi hành dựa vào cột dự đoán rời rạc

Cách mà thuật toán Microsoft Decision Trees xây dựng cây cho cột dự đoán rời rạc có thể được minh họa bằng biểu đồ. Biểu đồ sau chỉ ra các cột dự đoán, Bike Buyer (mua xe đạp), ngược lại với cột nhập vào, Age. Biểu đồ chỉ ra tuổi của người giúp phân biệt người đó sẽ mua xe đạp hay không.



Hình 4.1: Biểu đồ dự đoán người mua xe đạp dựa vào độ tuổi

Sự tương quan được chỉ ra trong biểu đồ làm cho thuật toán Microsoft Decision Trees tạo ra node mới trong mô hình.

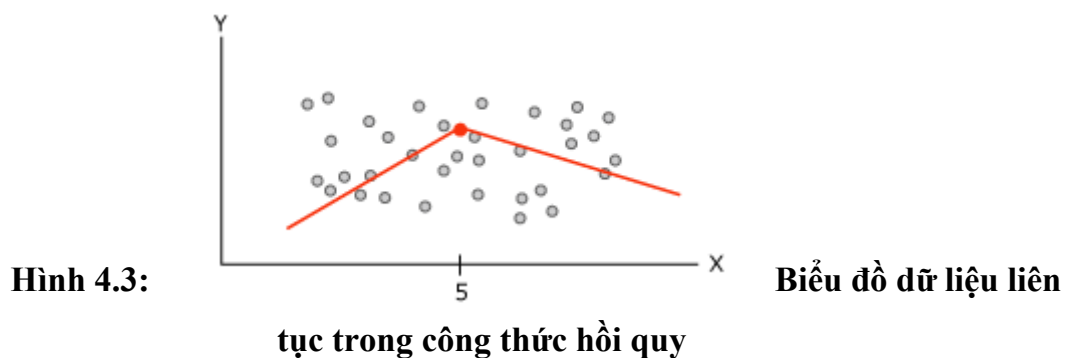


Hình 4.2: Biểu đồ cây quyết định được tạo ra

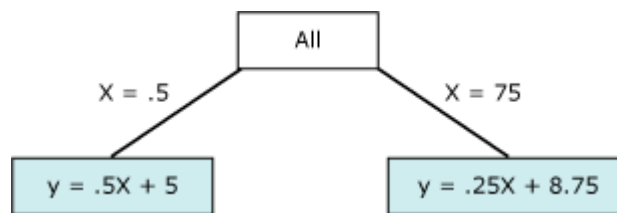
Khi thuật toán thêm một node mới vào mô hình, cấu trúc cây được hình thành. Node trên cùng của cây mô tả sự phân tích cột dự đoán cho mẫu toàn diện của khách hàng. Khi mô hình tiếp tục được phát triển, thuật toán đi đến tất cả các cột.

3.2. Thi hành dựa vào cột dự đoán liên tục

Khi thuật toán Microsoft Decision Trees xây dựng một cây dựa trên cột dự đoán liên tiếp, mỗi node chứa một công thức hồi quy. Sự phân chia xảy ra tại mỗi điểm của non-linearity trong công thức hồi quy. Ví dụ, xem sơ đồ sau.



Biểu đồ chứa dữ liệu có thể được mô hình hóa bằng cách sử dụng một đường đơn hoặc hai đường liên kết với nhau. Tuy nhiên, một đường đơn trình bày dữ liệu kém hơn. Thay vào đó, nếu bạn dùng hai đường, mô hình sẽ làm việc **tốt hơn cho một** dữ liệu tương đương. Tại điểm mà hai đường gặp nhau là điểm non-linearity, và là điểm mà tại đó một nút trong mô hình cây quyết định có thể phân chia. Ví dụ, nút phù hợp với điểm của non-linearity trong biểu đồ trước được mô tả bằng biểu đồ sau. Hai biểu thức trình bày hai biểu thức hồi quy cho hai đường.



Hình 4.4: Biểu đồ cây quyết định của cột dự đoán liên tục

4. CHI TIẾT THUẬT TOÁN

4.1.Tạo Cây

Cây quyết định được tạo thành bằng cách lần lượt chia (đệ quy) một tập dữ liệu thành các tập dữ liệu con, mỗi tập con được tạo thành chủ yếu từ các phần tử của cùng một lớp.

Các nút không phải là nút lá là các điểm phân nhánh của cây. Việc phân nhánh tại các nút có thể dựa trên việc kiểm tra một hay nhiều thuộc tính để xác định việc phân chia dữ liệu. Chúng ta chỉ xét việc phân nhánh nhị phân vì cho cây chính xác hơn.

4.2 Entropy và Information Gain

Đây là các công thức để tính toán cho việc chọn thuộc tính để phân nhánh cây quyết định. Việc chọn thuộc tính nào tại một nút để phân nhánh có thể dựa trên các chỉ số như Index hay Entropy.

Giả sử thuộc tính dự đoán có m giá trị phân biệt (tức là có m lớp C_i , $i=1, \dots, m$), S có s mẫu tin, s_i là số các mẫu tin trong S thuộc lớp C_i .

Index được tính như sau:

$$I(s_1, s_2, \dots, s_m) = 1 - \sum_{i=1}^m p_i^2$$

Entropy được tính như sau:

$$I(s_1, s_2, \dots, s_m) = - \sum_{i=1}^m p_i \log_2(p_i)$$

Giả sử thuộc tính A có n giá trị phân biệt $\{a_1, a_2, \dots, a_n\}$. Gọi S_j là tập con của S có giá trị của thuộc tính A là a_j , s_{ij} là số các mẫu tin thuộc lớp C_i trong tập S_j . Nếu phân nhánh theo thuộc tính A thì.

$$G(A) = \sum_{j=1}^n \frac{s_{1j} + \dots + s_{mj}}{s} I(s_{1j}, \dots, s_{mj})$$

Lúc đó ta có được chỉ số Gain, và ứng với thuộc tính A là.

$$\text{Gain}(A) = I(s_1, s_2, \dots, s_m) - G(A)$$

Dựa vào chỉ số Gain ta chọn thuộc tính để phân nhánh cho cây quyết định. $G(A)$ càng nhỏ thì các tập con càng đồng nhất hơn. Bởi vậy chúng ta chọn thuộc tính cho $\text{Gain}(A)$ lớn nhất để phân nhánh.

Sau khi đã chọn được thuộc tính tốt nhất, chúng ta tạo thêm một nút phân nhánh cho cây, gán nhãn cho nút là thuộc tính được chọn và tiến hành việc phân chia tập S.

4.3 Ví Dụ:

Dữ liệu: Ví dụ ta có các mẫu tin với các thông tin như sau:

<u>Color</u>	<u>Size</u>	<u>Shape</u>	<u>Edible?</u>
Yellow	Small	Round	+
Yellow	Small	Round	-
Green	Small	Irregular	+
Green	Large	Irregular	-
Yellow	Large	Round	+
Yellow	Small	Round	+
Yellow	Small	Round	+
Yellow	Small	Round	+
Green	Small	Round	-
Yellow	Large	Round	-
Yellow	Large	Round	+
Yellow	Large	Round	-
Yellow	Large	Round	-
Yellow	Large	Round	-
Yellow	Small	Irregular	+
Yellow	Large	Irregular	+

Bảng 4.1: Bảng dữ liệu mẫu cho ví dụ

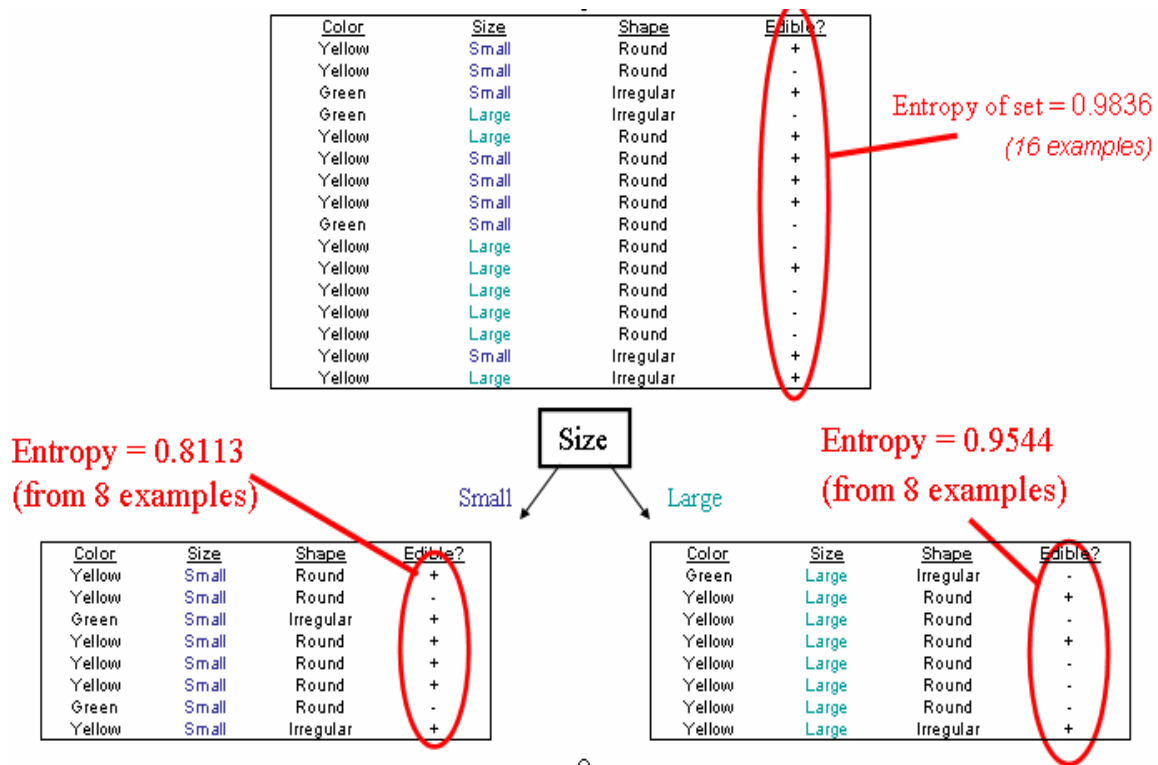
Trong việc phân lớp cho bảng dữ liệu trên, ta chọn cột thuộc tính là Edible. Và trong 16 mẫu tin trên có 9 mẫu tin với Edible là + và 7 mẫu tin có Edible là - . Vậy áp dụng các công thức ta tính được Entropy và Gain của tập dữ liệu trên là.

$$entropy(all_data) =$$

$$I(all_data) = - \left[\left(\frac{9}{16} \right) \log_2 \left(\frac{9}{16} \right) + \left(\frac{7}{16} \right) \log_2 \left(\frac{7}{16} \right) \right]$$

$$Entropy = 0,9836.$$

Để tính được chỉ số Gain thì dĩ nhiên ta phải phân chia nhánh cho cây quyết định qua một cột nào đó. Ở ví dụ này ta dùng SIZE để tạo nhánh tại nút gốc.



Hình 4.5: Tạo nhánh cho cây quyết định tại nút gốc

Chúng ta tạo nhánh cho cây tại nút gốc thông qua thuộc tính Size sau đó ta tính entropy cho các tập dữ liệu con thì ta được.

- Entropy_Small=0,8113 (Từ 8 mẫu tin)
- Entropy_Large=0,9544 (Từ 8 mẫu tin)

$$\text{expected_entropy}(\text{size}) = I(\text{size}) = \frac{8}{16}(0.8113) + \frac{8}{16}(0.9544) = 0.8828$$

Ta có: $I(\text{parent})$ là chỉ số entropy tại nút gốc, $I(\text{Size})$ là chỉ số entropy của các tập con khi phân chia cây tại nút gốc theo thuộc tính Size. Lúc đó ta có chỉ số $\text{Gain}(\text{Size})$ là:

$$\text{Gain}(\text{Size}) = I(\text{parent}) - I(\text{Size}) = 0,9836 - 0,8828 = 0,1008$$

Vậy ta có số $\text{Gain} = 0,1008$ khi chọn cách phân chia tập dữ liệu gốc với thuộc tính Size để phân nhánh đầu tiên.

Trên đây là ví dụ để mô tả cho việc tính toán các công thức phục vụ cho việc tạo cây quyết định nhằm một mục đích nào đó. Vậy, với tập dữ liệu đó thì qua quá trình tính toán để chọn thuộc tính tạo nhánh phù hợp là gì.

Đầu tiên ta tính entropy của tập dữ liệu ban đầu, sau đó ta tính entropy của các tập dữ liệu con khi tạo nhánh dựa vào cột nào đó. Và quá trình tính toán cho ta kết quả như sau.

$$I(\text{all_data}) = 0.9836$$

$$I(\text{Size}) = 0.8829$$

$$G(\text{Size}) = 0.1007$$

$$\text{Size} = \text{small}, +2, -6;$$

$$I(\text{size} = \text{small}) = 0.8112$$

$$\text{Size} = \text{large}, +3, -5; \quad I(\text{size} = \text{large}) = 0.9544$$

$$I(\text{Color}) = 0.9532$$

$$G(\text{Color}) = 0.0304$$

$$\text{Color} = \text{green}, +1, -2;$$

$$I(\text{Color} = \text{green}) = 0.9183$$

$$\text{Color} = \text{yellow}, +8, -5; \quad I(\text{Color} = \text{yellow}) = 0.9612$$

$$I(\text{Shape}) = 0.9528$$

$$G(\text{Shape}) = 0.0308$$

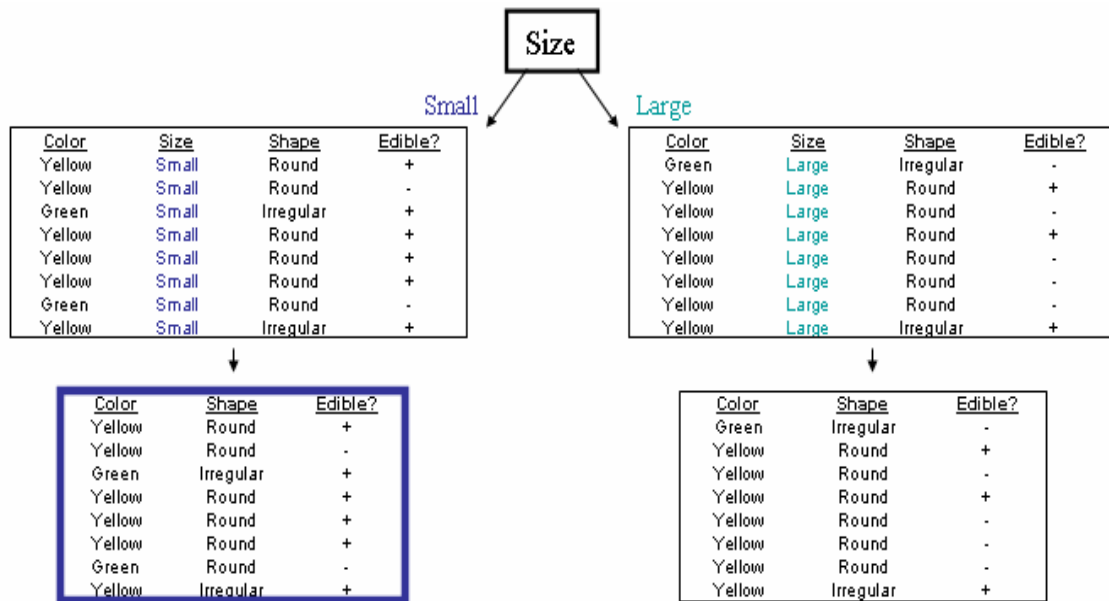
$$\text{Shape} = \text{regular}, +6, -6;$$

$$I(\text{Shape} = \text{regular}) = 1.0$$

$$\text{Shape} = \text{irregular}, +3, -1;$$

$$I(\text{Shape} = \text{irregular}) = 0.8113$$

Vậy qua kết quả tính được như trên ta thấy Gain (Size) là lớn nhất, vậy ta sẽ chọn Size làm thuộc tính để phân nhánh tại nút gốc. Để tiếp tục tạo, tia cây quyết định ta đệ quy quá trình trên cho mỗi tập con. Và kết quả sau khi phân nhánh bậc 1 như sau.



Hình 4.6: Kết quả phân nhánh tại nút

5. Ưu điểm của cây quyết định:

Cây quyết định là phương pháp có một số ưu điểm:

- Cây quyết định dễ hiểu. Người ta có thể hiểu mô hình cây quyết định sau khi được giải thích ngắn.
- Việc chuẩn bị dữ liệu cho một cây quyết định là cơ bản hoặc không cần thiết. Các kỹ thuật khác thường đòi hỏi chuẩn hóa dữ liệu, cần tạo các biến phụ (dummy variable) và loại bỏ các giá trị rỗng.
- Cây quyết định có thể xử lý cả dữ liệu có giá trị bằng số và dữ liệu có giá trị là tên thể loại. Các kỹ thuật khác thường chuyên để phân tích các bộ dữ liệu chỉ gồm một loại biến. Chẳng hạn, các luật quan hệ chỉ có thể dùng cho các biến tên, trong khi mạng nơ-ron chỉ có thể dùng cho các biến có giá trị bằng số.
- Cây quyết định là một mô hình hộp trắng. Nếu có thể quan sát một tình huống cho trước trong một mô hình, thì có thể dễ dàng giải thích điều kiện đó bằng logic Boolean. Mạng nơ-ron là một ví dụ về mô hình hộp đen, do lời giải thích cho kết quả quá phức tạp để có thể hiểu được.

- Có thể thẩm định một mô hình bằng các kiểm tra thống kê. Điều này làm cho ta có thể tin tưởng vào mô hình.
- Cây quyết định có thể xử lý tốt một lượng dữ liệu lớn trong thời gian ngắn. Có thể dùng máy tính cá nhân để phân tích các lượng dữ liệu lớn trong một thời gian đủ ngắn để cho phép các nhà chiến lược đưa ra quyết định dựa trên phân tích của cây quyết định.

6. Mô hình phiên dịch (Demo từ CSDL khảo sát tình hình chung của khu vực về mức sống, thu nhập, nhập khẩu...v.v.).

Sau khi các mô hình đã được xử lý, ta có thể xem chọn Mining Model Viewer trong editor để duyệt lại chúng. Sử dụng combo box Mining Model ở đầu thẻ để kiểm tra lại cấu trúc mô hình.

(1) Microsoft Decision Trees Model

Trong thẻ Mining Model Viewer mặc định đang mở mô hình **KhaoSatmuc song**, cấu trúc mô hình đầu tiên. Trong phần Tree viewer thì chứa 2 thẻ là Decision Tree và Dependency Network.

(a) Decision Tree

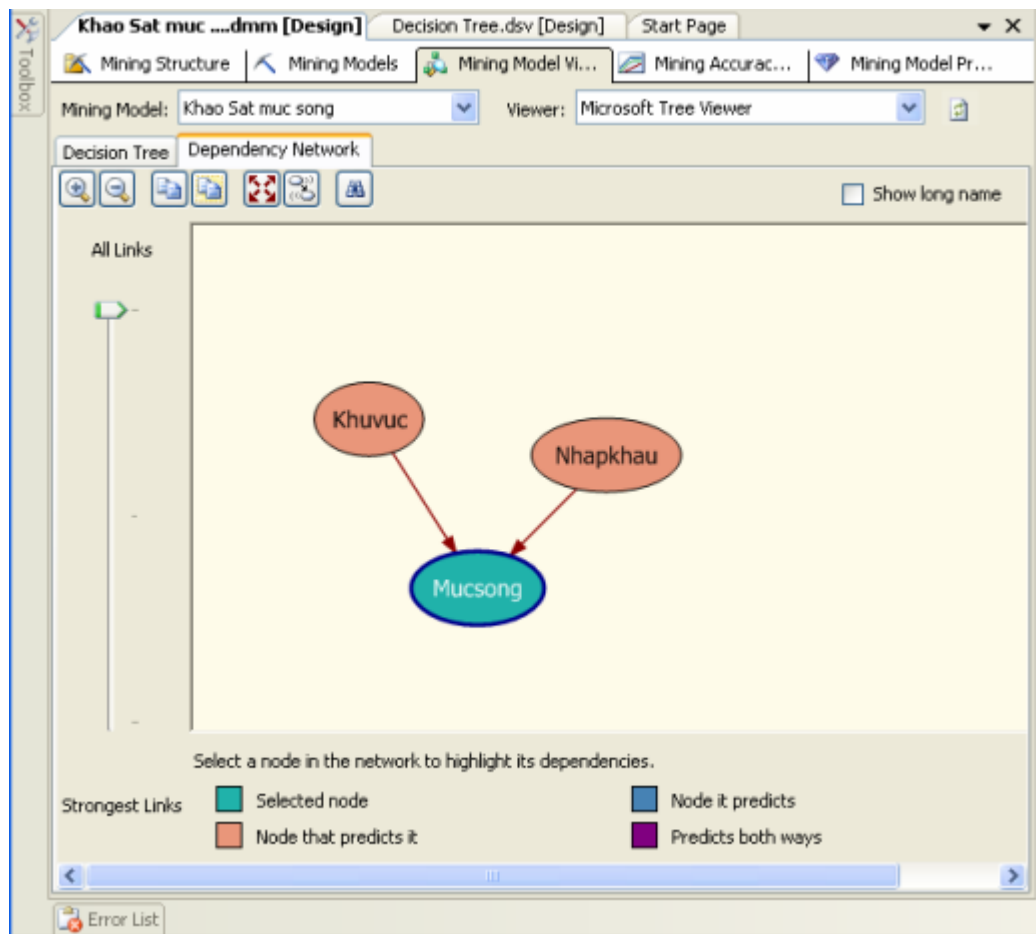
Từ thẻ Decision Tree ta sẽ kiểm tra tất cả các mô hình cây để chuẩn bị cho mô hình khai thác. Sẽ không có mô hình cây nào cho mỗi thuộc tính có khả năng dự đoán trong mô hình trừ khi nó được lựa chọn theo yêu cầu. Bởi vì mô hình chỉ chứa duy nhất một thuộc tính dự đoán nên sẽ không có kiểu cây ở đây. Nếu có sự hiện diện nhiều cây ta sẽ chọn mục Tree để xem được những cây khác.

Tree viewer mặc định hiển thị nhánh đầu tiên, nếu cây có ít hơn ba cấp nhánh, Tree viewer sẽ hiển thị hết. Ta có thể xem chi tiết cây hơn bằng cách chọn thanh trực Show Level hoặc Default Expansion.

Cách làm như sau:

1. Trực Show Level đến mức 5.
2. Từ danh sách Background, chọn 1 (Mức sống = cao)

Thẻ Dependency Network hiển thị thông tin về mối liên hệ giữa các thuộc tính có khả năng tạo nên quyết định trong mô hình KPDL.



Hình 4.8: Mô hình tạo mối liên hệ giữa các thuộc tính tạo cây

Node trung tâm là **Mucsong**, nó thể hiện các thuộc tính dự đoán trong mô hình dữ liệu còn các node xung quanh thể hiện thuộc tính bị tác động bởi thuộc tính dự đoán. Di chuyển con trỏ bên trái làm cho hình ảnh sẽ rõ hơn.

e) Kiểm tra độ chính xác của các mô hình

Đến đây thì các mô hình đã được xử lý và khám phá. Thẻ Mining Accuracy Chart sử dụng dữ liệu kiểm tra tách biệt với tập dữ liệu huấn luyện gốc để so sánh dự đoán với kết quả đã biết. Những kết quả này được sắp xếp và vẽ lên đồ thị thể hiện khả năng dự đoán của các mô hình. Mô hình lý tưởng có khả năng dự đoán chính xác đến 100% theo thời gian.

Lift chart giúp phân biệt giữa mô hình gần đúng về cấu trúc với việc xác định mô hình cho dự đoán tốt nhất.

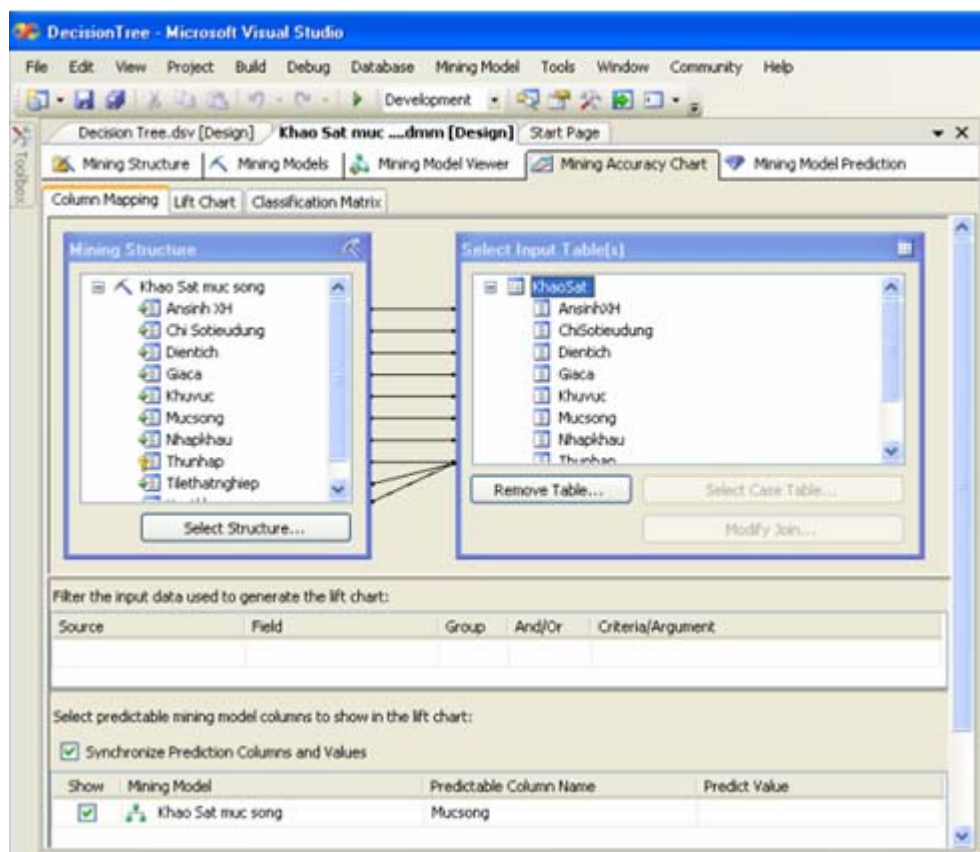
Từ thẻ Mining Accuracy Chart ta tạo mới theo 3 bước sau đây:

(1.1) Ánh xạ các cột dữ liệu

Bước đầu tiên là tạo ánh xạ từ các cột dữ liệu trong mô hình KPDL đang xét với các cột trong dữ liệu kiểm tra, nếu các cột này được đặt cùng tên thì công cụ sẽ tự động tạo các mối quan hệ.

Cách thực hiện ánh xạ

- Từ bảng Select Input Table(s) , click vào Select case table.
- Bảng Select Table mở ra, ta chọn ra dữ liệu để kiểm tra. (Trong trường hợp của chúng ta bảng **KhaoSat** được chọn tương đối giống so với bảng **KhaoSatmucsong** trong mô hình).
- Trong bảng Select Table, chọn **DecisionTree DW** từ data source.
- Chọn **KhaoSat** từ Table/View rồi OK.
- Các cột có cùng tên sẽ tự động ánh xạ nhau theo như hình.



Hình 4.9: Ánh xạ các cột dữ liệu

Một câu truy vấn dự đoán sẽ được tạo ra cho mỗi mô hình trong cấu trúc dựa vào việc ánh xạ các cột dữ liệu này. Ta có thể xóa bỏ một ánh xạ bằng cách click chọn vào đường nối giữa chúng rồi DELETE, cũng có thể tạo ánh xạ bằng cách kéo từ bảng Mining Structure sang bảng Select Input Tables(s).

(1.2) Lọc các dòng dữ liệu nhập

Việc xử lý lọc dữ liệu nhập thực hiện qua lưới dữ liệu dưới mục Filter the input data used to generate the lift chart, nằm ngay dưới 2 bảng dữ liệu ánh xạ bước trên. Lưới dữ liệu này hỗ trợ kéo thả giống như trên bảng dữ liệu của cơ sở dữ liệu quan hệ SQL, ta cũng có thể lọc dữ liệu qua các toán tử Criteria/Argument ở phần cột sau cùng của lưới dữ liệu.

(1.3) Lựa chọn mô hình, cột dự đoán, các giá trị

Bước này ta lựa chọn mô hình để đưa vào lift chart và cột dự đoán để so sánh. Mặc định thì tất cả mô hình trong cấu trúc mining sẽ được chọn, ta **có thể bỏ** qua mô

hình nào đó. Có thể tạo ra 2 loại biểu đồ, nếu chọn một giá trị dự đoán ta sẽ thấy một biểu đồ có một đường nâng các đường mô hình lên, còn nếu ta không chọn một giá trị dự đoán nào thì biểu đồ sẽ khác, nó chỉ cho thấy độ chính xác của mô hình (xem hình phần Xem lift chart).

Thực hiện:

Với mỗi mô hình, trong phần Predictable Column Name, ta chọn **Mucsong** Với mỗi mô hình, trong cột Predict Value, chọn 1 (Mức sống = cao) hoặc 0 (Mức sống = thấp).

Hiển thị độ chính xác của mô hình:

Trong mục Predictable Column Name, chọn **Mucsong**

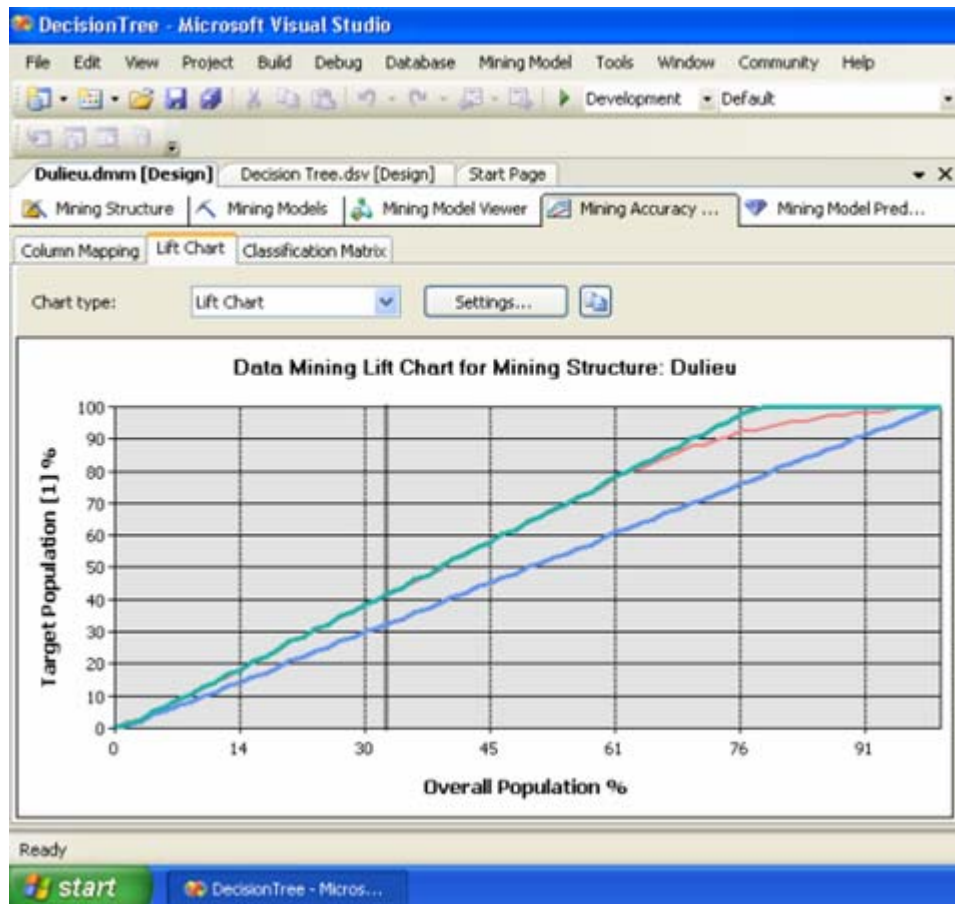
Cứ để cột Predict Value rỗng.

Nếu chọn Synchronize Prediction Columns and Values thì cột dự đoán sẽ được đồng bộ với mỗi mô hình trong cấu trúc mining.

(1.4) Hiển thị Lift Chart

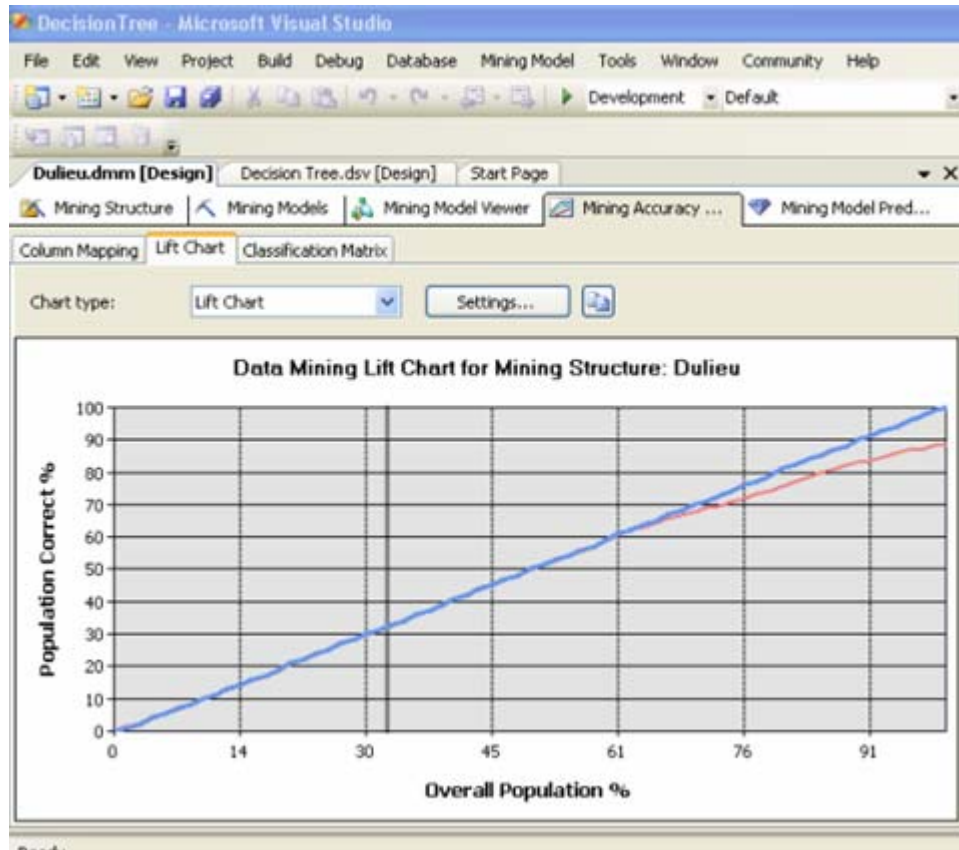
Chọn vào thẻ Lift Chart để hiển thị, khi chọn vào thẻ này, một câu truy vấn chạy trên server, cơ sở dữ liệu của cả cấu trúc mining và bảng input. Kết quả dự đoán được so sánh với kết quả thật đã biết và được sắp xếp theo khả năng rồi đưa lên biểu đồ.

Nếu chọn một giá trị dự đoán ta sẽ thấy một biểu đồ có một đường nâng các đường mô hình lên



Hình 4.10: Độ chính xác của mô hình khi chọn giá trị dự đoán

Còn nếu ta không chọn một giá trị dự đoán nào thì biểu đồ sẽ khác, nó chỉ cho thấy độ chính xác của mô hình



Hình 4.11: Độ chính xác của mô hình khi không chọn giá trị dự đoán

f) Tạo dự đoán và kết quả

Nếu đã hài lòng với mô hình KPDL ta bắt đầu tạo câu truy vấn dự đoán DMX sử dụng công cụ Prediction Query Builder. Prediction Query Builder có 3 cách dùng là Design, Query và Result. Nó tương tự như Access Query Builder và ta có thể thực hiện việc kéo thả để tạo câu truy vấn.

(1) Tạo câu truy vấn

Ta chọn mô hình mining và bảng input

1. Trong Mining Model, chọn Select model.

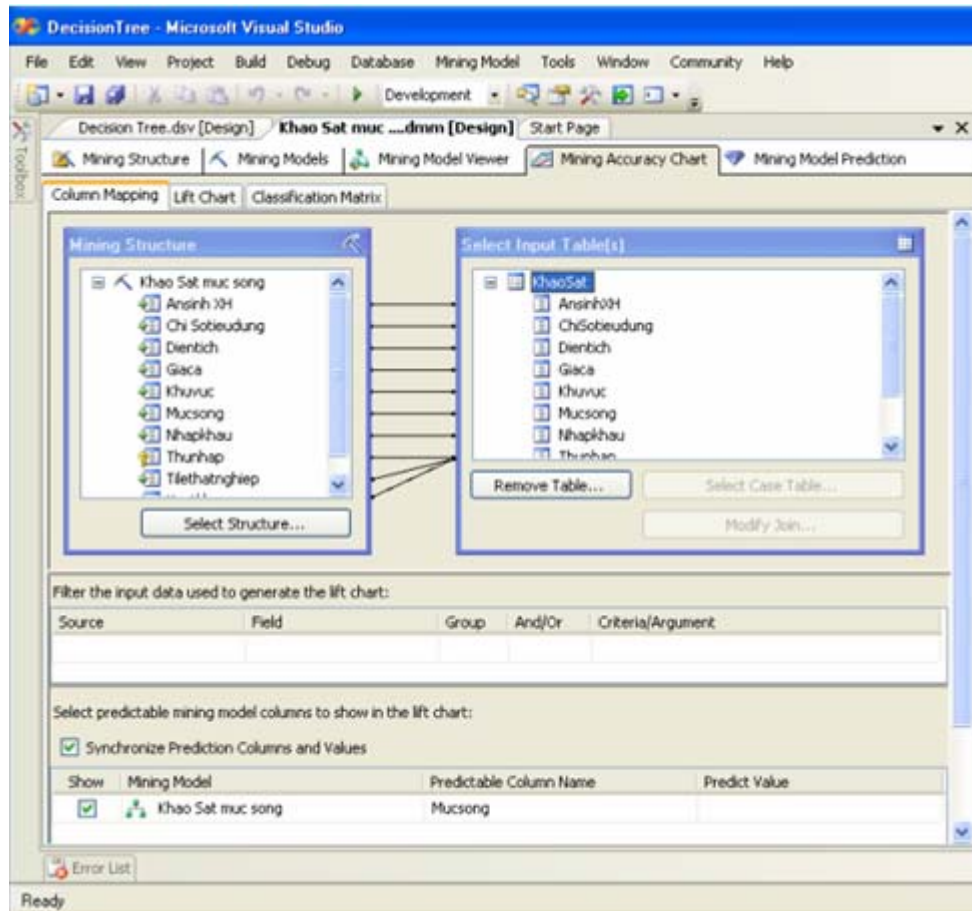
Hộp thoại Select Mining Model mở ra. Mặc định mô hình đầu tiên sẽ được chọn

2. Chọn **KhaoSat**.

3. Trên bảng Select Input Table(s), chọn mục Select case table.

4. Trong hộp thoại Select Table duyệt cây để chọn bảng **KhaoSat** nằm trong DecisionTree data source view.

Sau khi chọn bảng input thì Prediction Query Builder mặc định ánh xạ giữa các cột có cùng tên với nhau.



Hình 4.12: Ánh xạ dữ liệu để tạo dự đoán

5. Trong cột Source, chọn ô trong dòng trống đầu tiên và sau đó chọn vào **KhaoSat** table.

6. Ở cột Field, cạnh ô ở bước 5 chọn **ThunhapKey**.

Tạo ID duy nhất cho câu truy vấn dự đoán để ta có thể xác định **Mucsong**

7. Chọn ô kế tiếp trong cột Source, chọn mô hình **KhaoSat**.

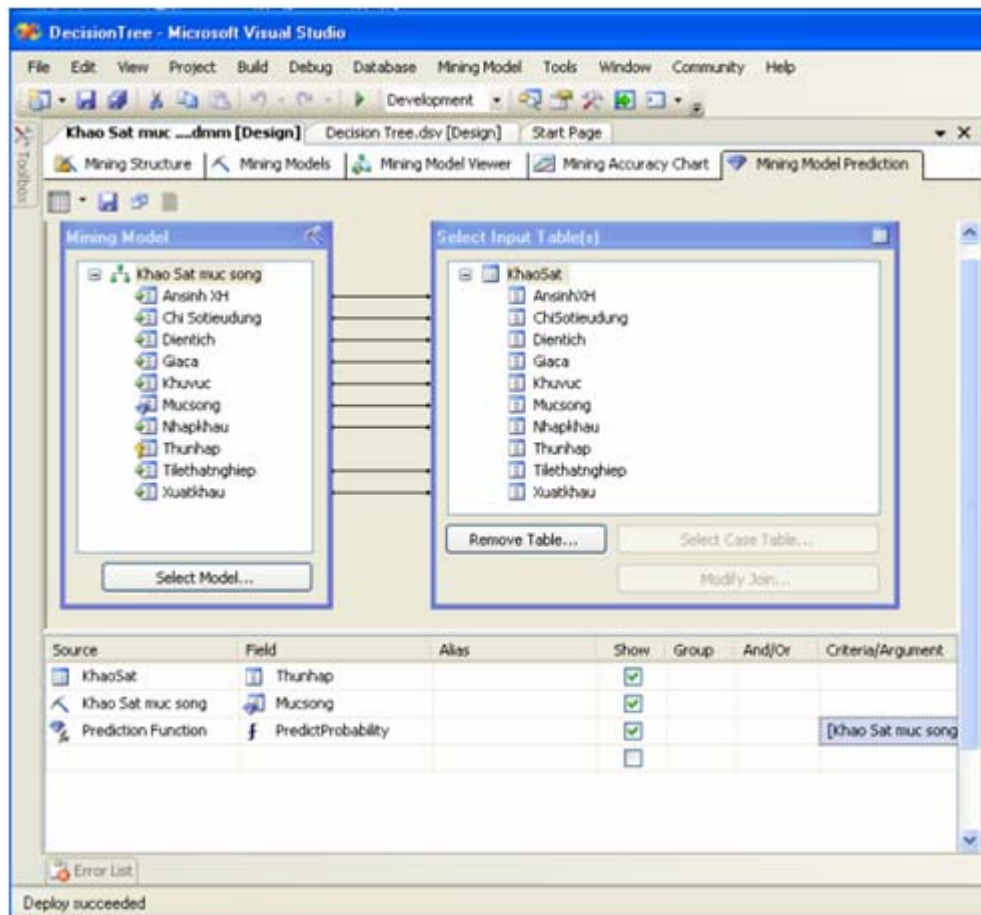
8. Ở ô Field, chọn **Mucsong**.

9. Chọn ô kế tiếp dưới cột Source, và chọn Prediction Function.

10. Trong Prediction Function, ở cột Field, chọn PredictProbability.

Prediction functions cho biết cách mà mô hình đưa ra dự đoán và khả năng dự đoán đúng. Ta có thể bổ sung thêm thông số để phục vụ cho chức năng này ở cột Criteria/Argument.

11. Ở cột Criteria/Argument, gõ vào **[KhaoSat].[Mucsong]**.

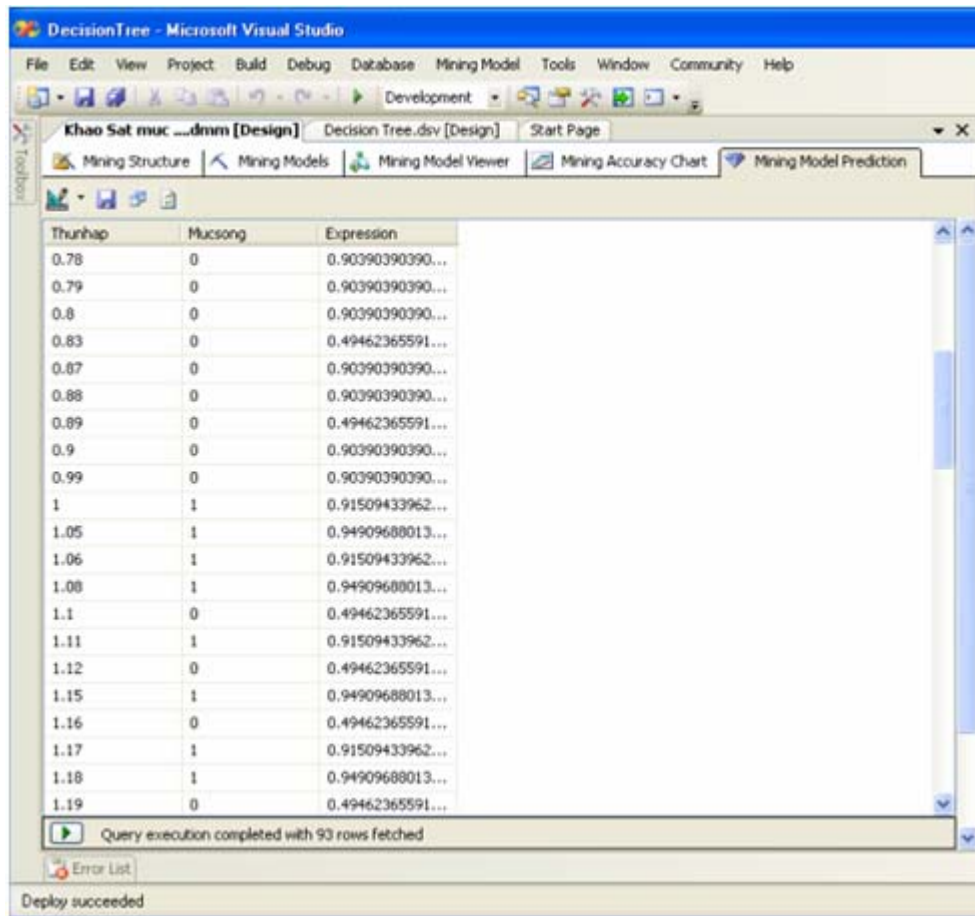


Hình 4.13: Tạo dự đoán

Xem câu truy vấn DMX phát sinh tự động bởi Prediction Query Builder bằng cách click vào icon ở góc trên bên trái view. Ta có thể chỉnh sửa câu truy vấn và chạy lại. Việc sử dụng Prediction Query Builder giống như cách dùng view trên SQL Server.

(2) *Xem kết quả*

Xem kết quả dự đoán bằng cách click vào nút mũi tên cạnh icon ở trên góc trái của thẻ, rồi chọn Result, kết quả hiển thị như hình:



The screenshot shows the 'DecisionTree - Microsoft Visual Studio' application window. The main area displays a table with three columns: 'Thunhap', 'Mucsong', and 'Expression'. The table contains 20 rows of data. Below the table, a status bar indicates 'Query execution completed with 93 rows fetched'. At the bottom, a message says 'Deploy succeeded'.

Thunhap	Mucsong	Expression
0.78	0	0.90390390390...
0.79	0	0.90390390390...
0.8	0	0.90390390390...
0.83	0	0.49462365591...
0.87	0	0.90390390390...
0.88	0	0.90390390390...
0.89	0	0.49462365591...
0.9	0	0.90390390390...
0.99	0	0.90390390390...
1	1	0.91509433962...
1.05	1	0.94909688013...
1.06	1	0.91509433962...
1.08	1	0.94909688013...
1.1	0	0.49462365591...
1.11	1	0.91509433962...
1.12	0	0.49462365591...
1.15	1	0.94909688013...
1.16	0	0.49462365591...
1.17	1	0.91509433962...
1.18	1	0.94909688013...
1.19	0	0.49462365591...

Hình 4.14: Kết quả dự đoán

Ba cột **Thunhap**, **Mucsong**, và **Expression** thể hiện tình hình khu vực, và khả năng dự đoán đúng. Ta sẽ sử dụng kết quả này để kiểm soát tình hình kinh tế.

CHƯƠNG 5: Kết luận – Hướng phát triển

5.1 Các mục tiêu đã thực hiện trong đề tài

Sau 1 thời gian thực hiện đề tài, ta có thể đáp ứng các mục tiêu mà đề tài đặt ra:

- Khai thác được khả năng tiềm ẩn của dữ liệu.
- Hiểu được dữ liệu để đưa ra quyết định khi tạo ra các mô hình dự đoán. Các kĩ thuật khảo sát bao gồm tính toán các giá trị nhỏ nhất và lớn nhất, tính toán độ trung bình và độ chênh lệch, và nhìn vào thuộc tính dữ liệu.
- Dựa vào cơ sở dữ liệu tiếp tục được cập nhật với khách hàng tiềm năng.
- Sử dụng các mô hình để tạo các dự đoán, mà có thể sử dụng sau đó để tạo ra các quyết định nghiệp vụ.
- Đưa chức năng khai thác dữ liệu trực tiếp vào ứng dụng.
- Tạo báo cáo để người dùng trực tiếp nêu query với mô hình khai thác tồn tại.

Cập nhật mô hình là một phần trong chiến lược triển khai. Khi dữ liệu nhập vào tổ chức càng nhiều thì phải xử lý lại các mô hình, bằng cách đó sẽ cải thiện hiệu quả của chúng.

- Việc thiết lập các luật mô tả làm cách nào các sản phẩm được gom nhóm lại với nhau thành một thao tác.
- Cây quyết định dự đoán một khách hàng cụ thể sẽ mua một sản phẩm hay không.
- Mô hình toán học dự đoán việc mua bán.

5.2 Hướng phát triển

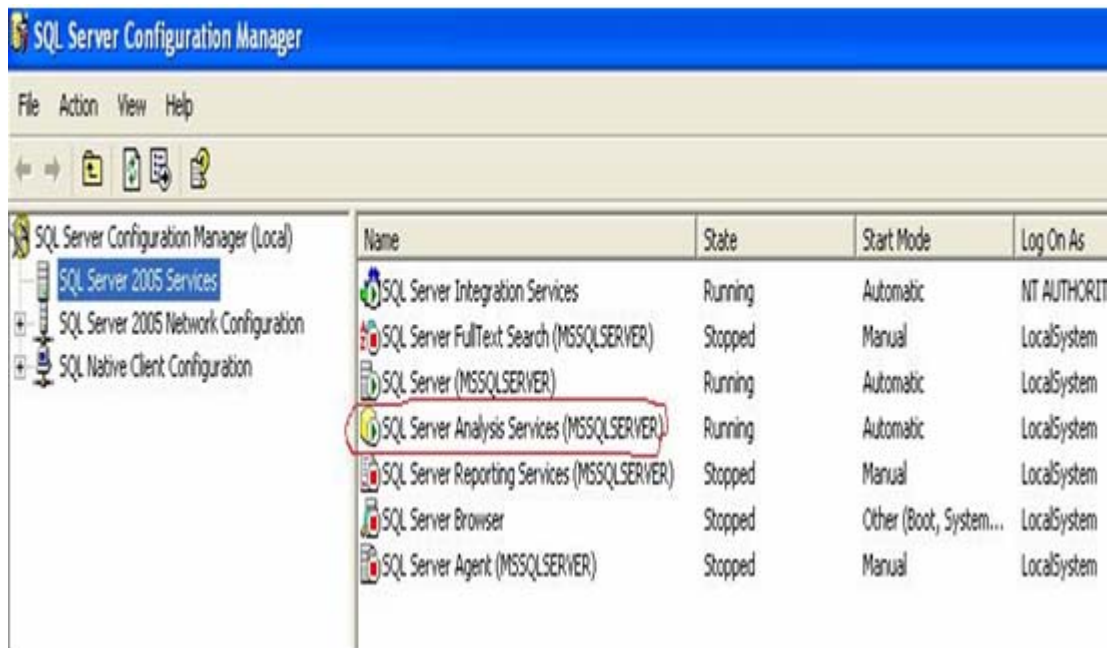
Trên cơ sở đã thực hiện, đề tài của em có các hướng phát triển như sau:

- Có thể nghiên cứu đi sâu vào cơ sở dữ liệu với các thuật toán trong SQL Server 2005 để đưa ra các luật tốt nhất cho dự đoán.

PHỤ LỤC: HƯỚNG DẪN CÁC BƯỚC DEMO CHƯƠNG TRÌNH

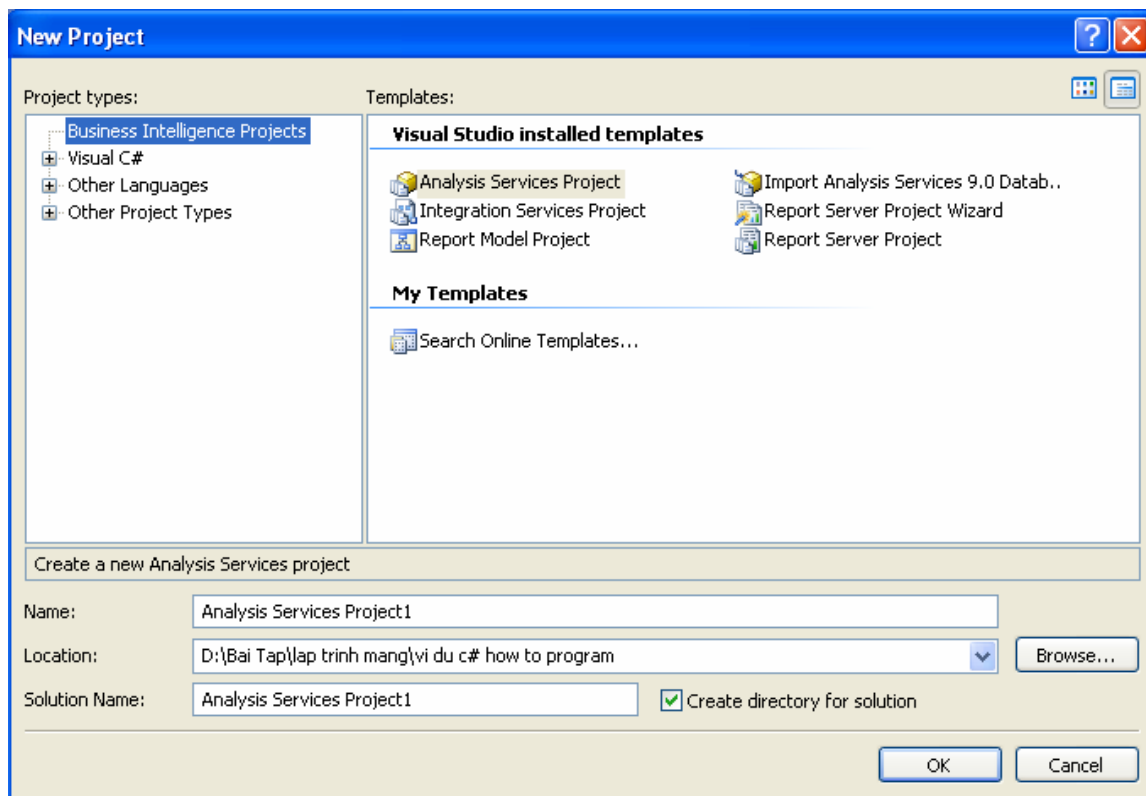
A. Yêu cầu hệ thống trước khi chạy:

- Cài đặt SQL Server 2005
- Khi cài đặt SQLServer 2005 nhớ cài đặt thêm bộ *Business Intelligence Development Studio*. Business Intelligence Development Studio là ngôn ngữ dùng để tạo và thực thi chương trình.
- Bạn phải chắc rằng dịch vụ phân tích đã được chạy



B. Quá trình chạy Demo chương trình

- Chạy *Business Intelligence Development Studio*. Chọn newProjects -> Business Intelligence Projects -> Analysis Services Project

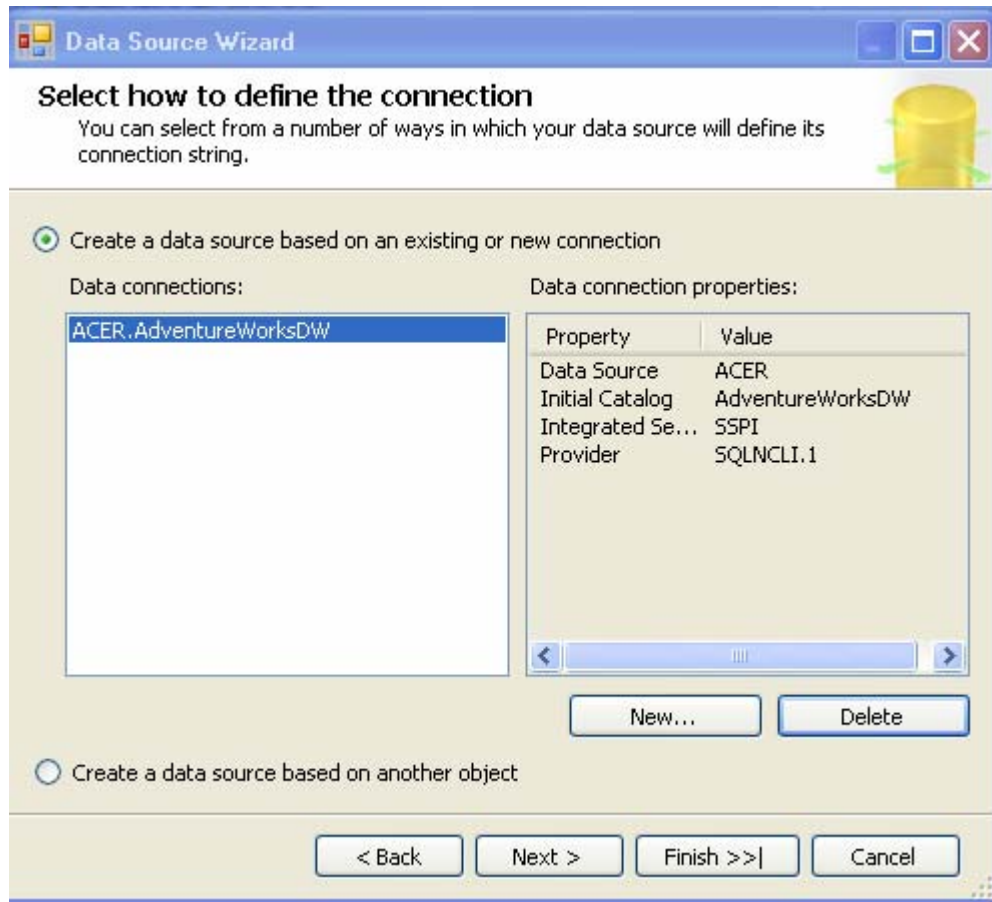


- **Tạo data source**

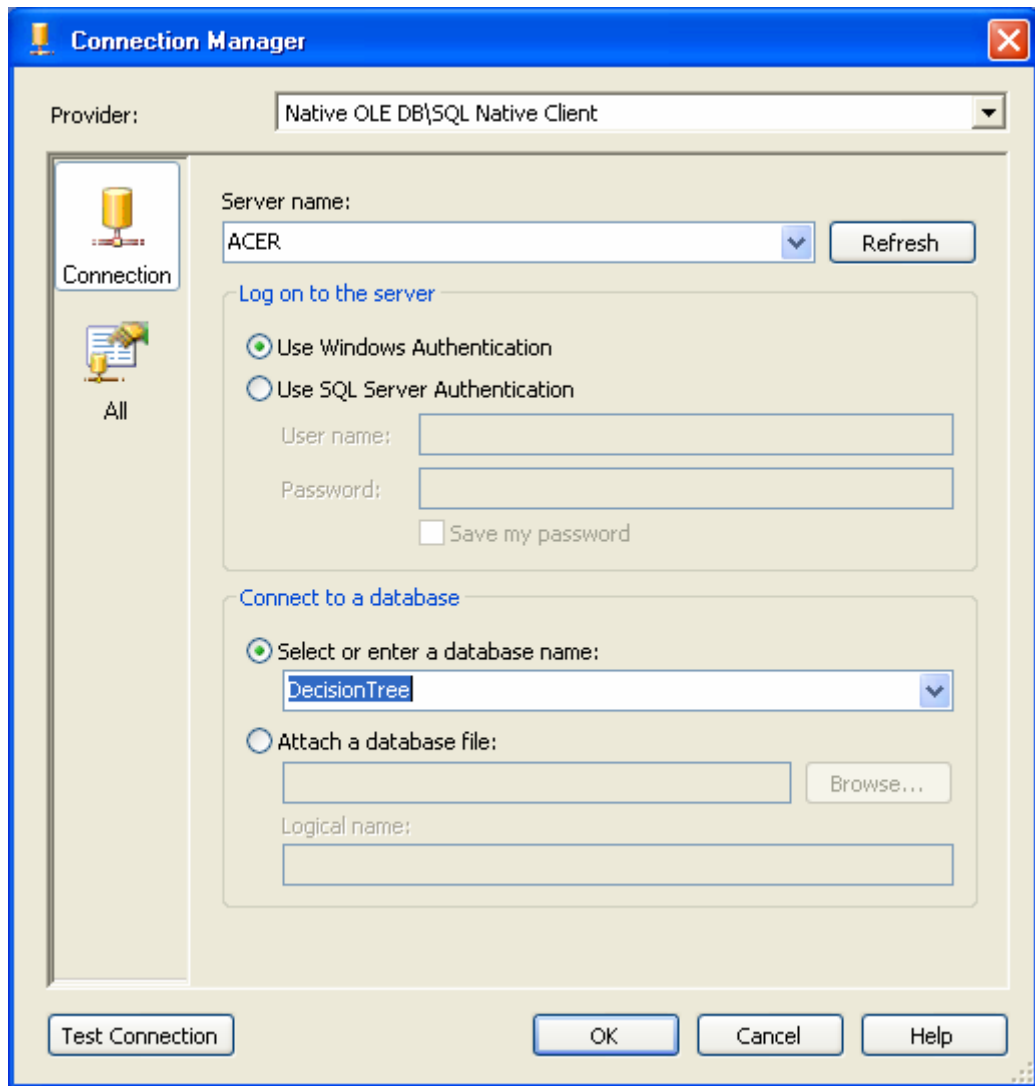
Click phải lên data source -> New Datasource



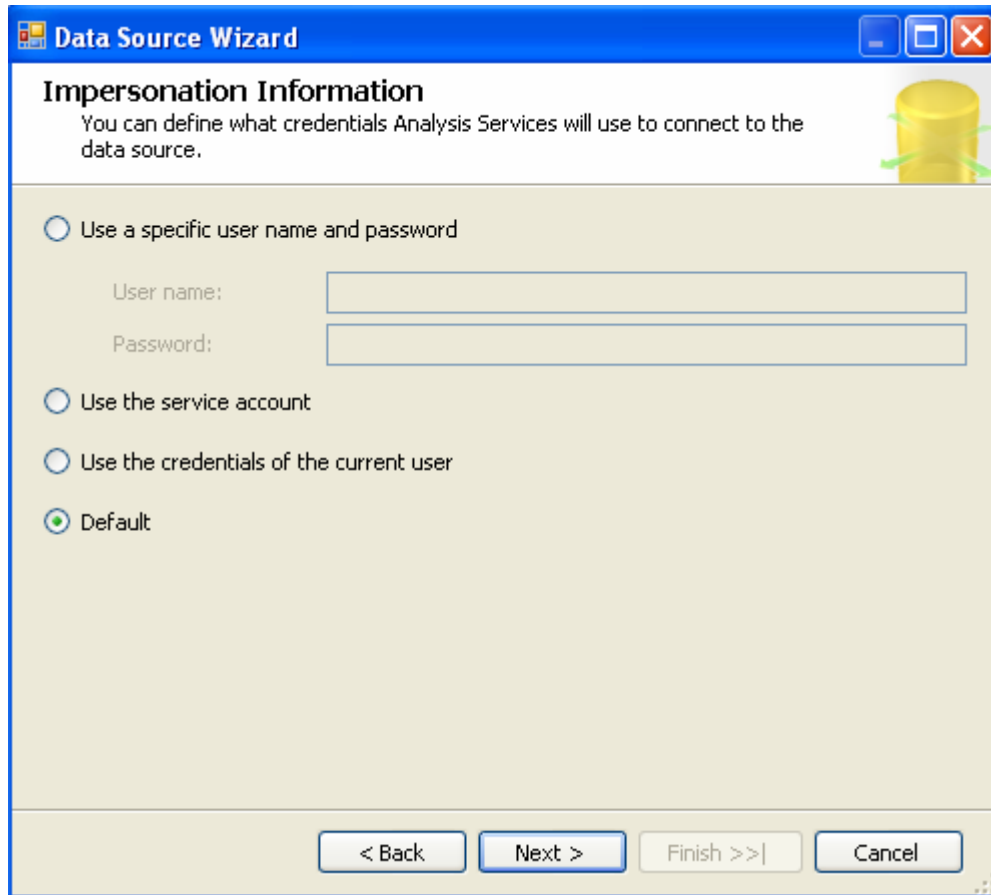
Click Next



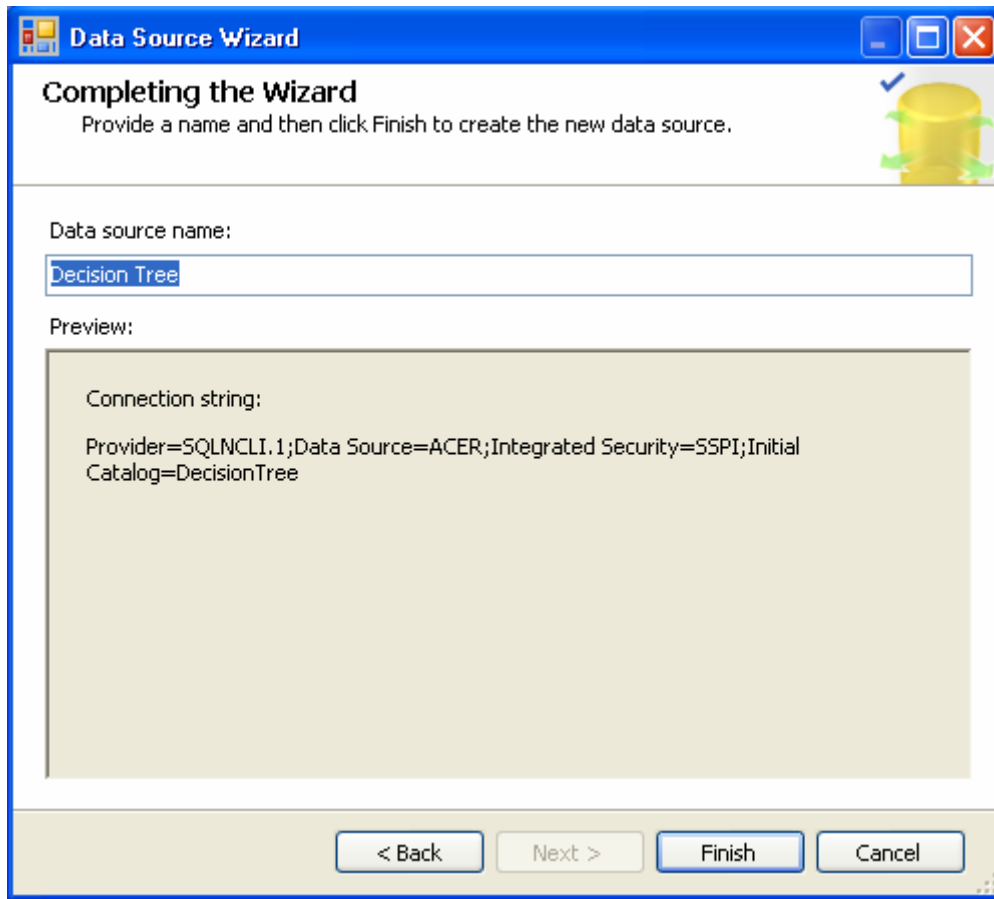
Chọn New và khai báo các thông số kết nối



Click OK



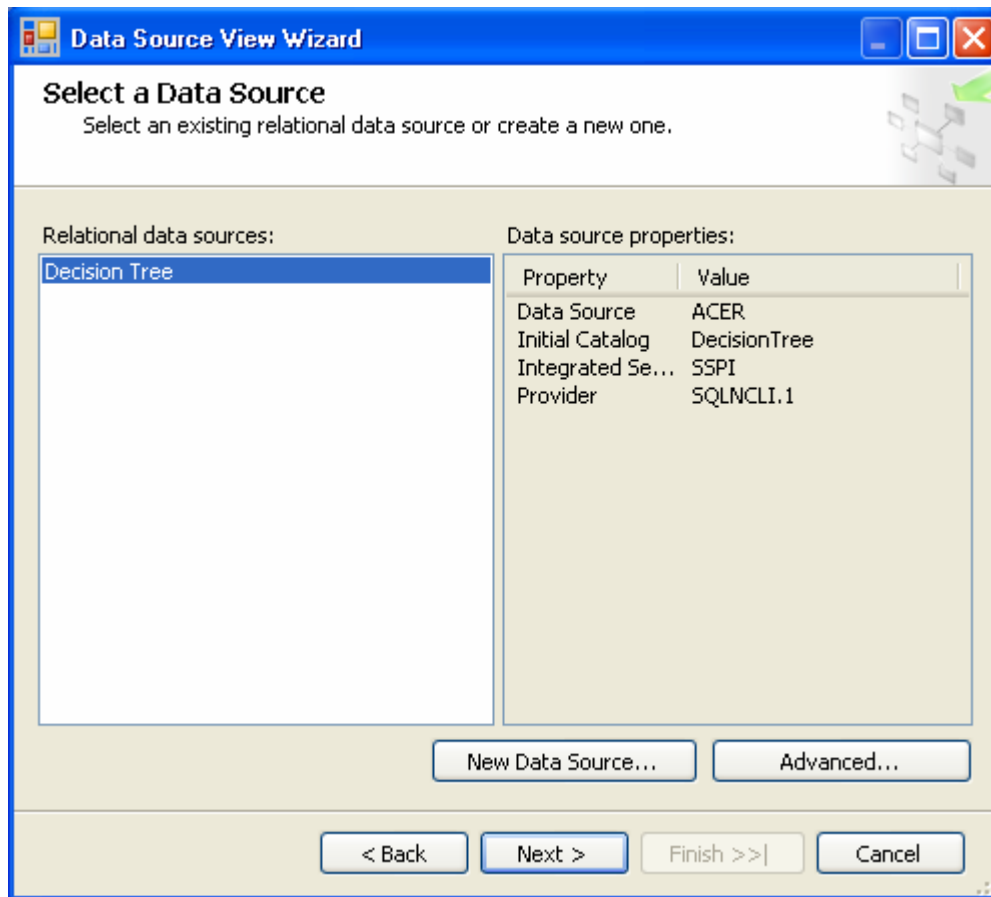
Click Next và đặt tên cho data source



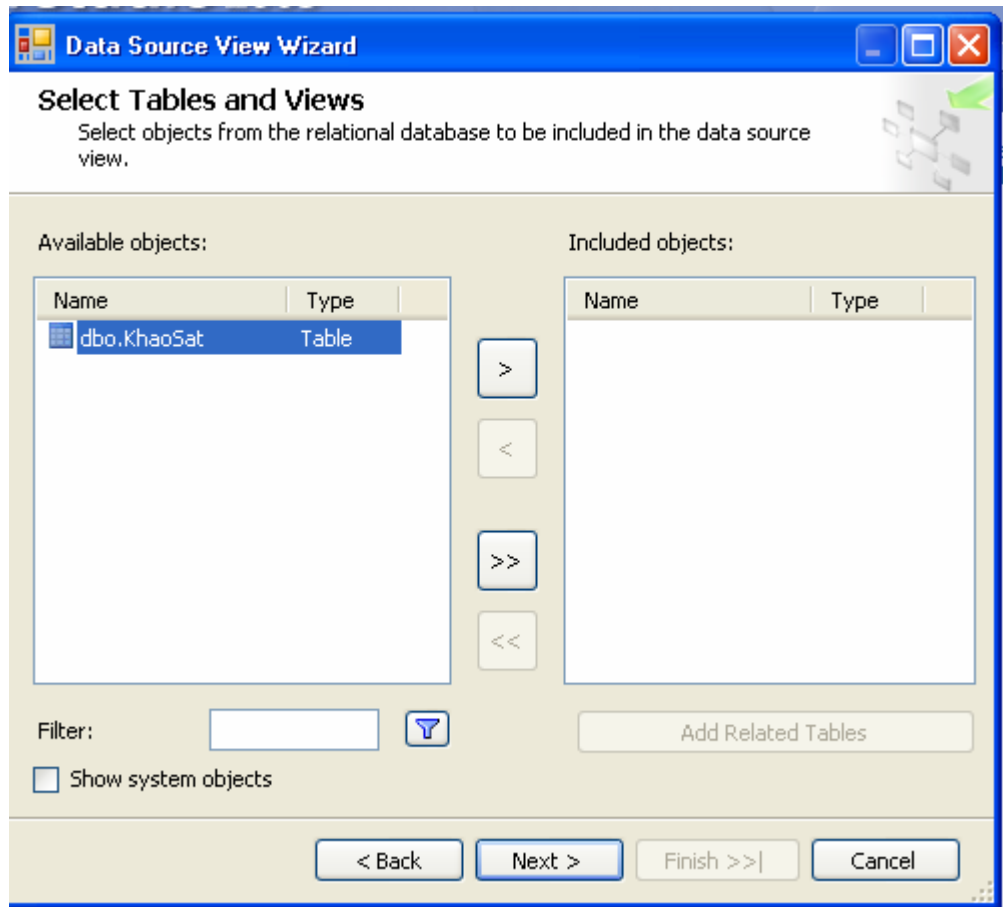
Click Finish

- **Tạo Data Source Views**

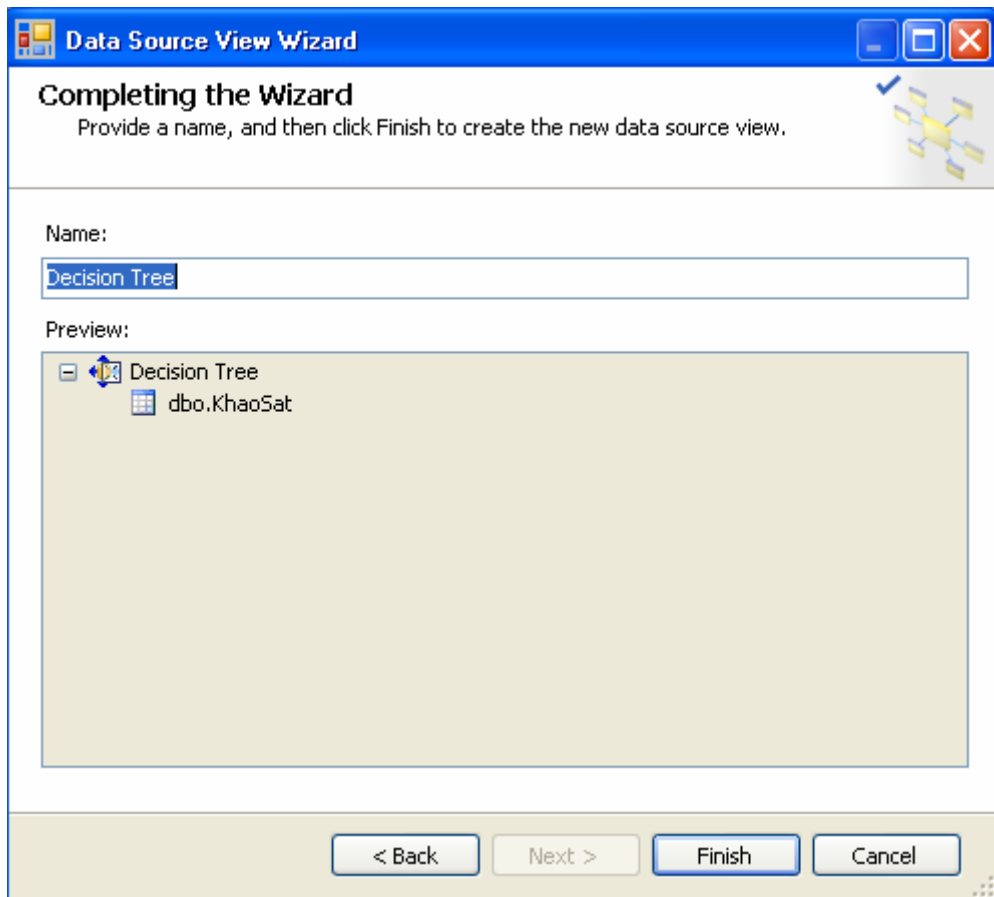
Click phải Data Source Views chọn New Data Source View. Chọn Data Source tồn tại.



Chọn các view chạy chương trình



Đặt tên cho Data Source View



Click Finish

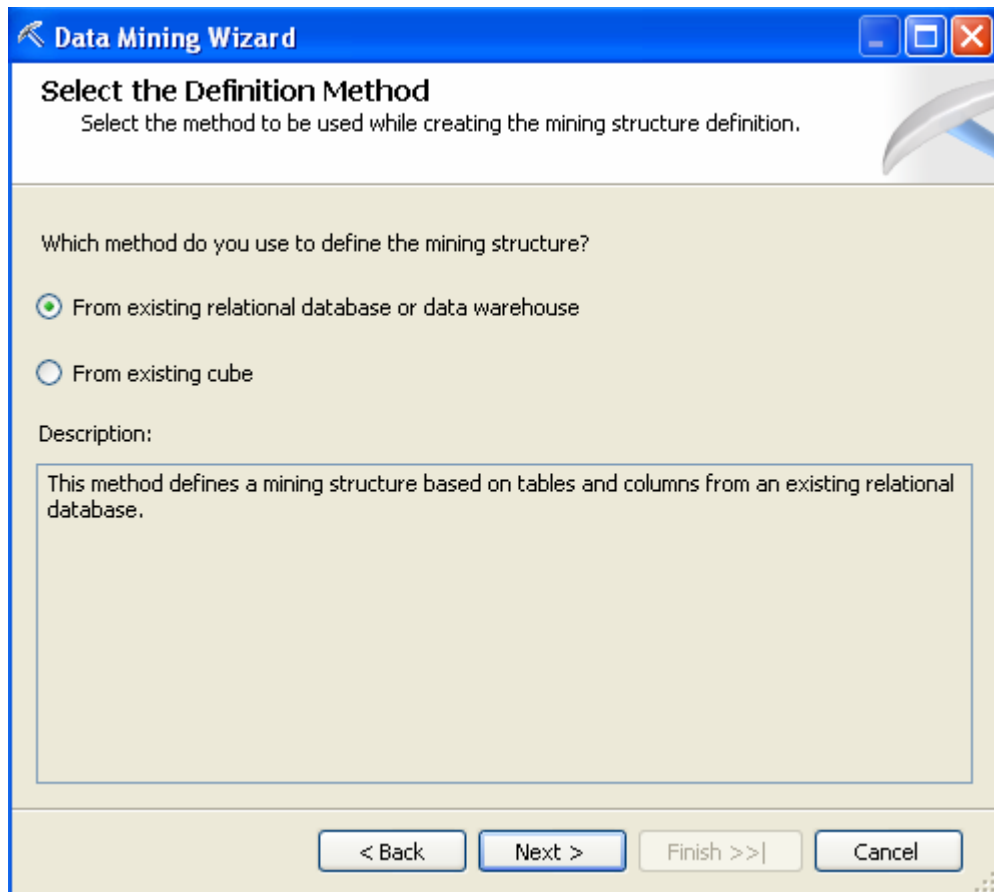
- Tạo relationship giữa các view: Nếu dữ liệu của chúng ta phục vụ cho việc thực thi mô hình, ta sẽ tạo mối quan hệ many-to-one giữa các view. Sau khi tạo quan hệ xong các view sẽ lồng vào nhau khi tạo mô hình.

- **Tạo 1 Mining Structures**

Click phải Mining Structures chọn New Mining Structures

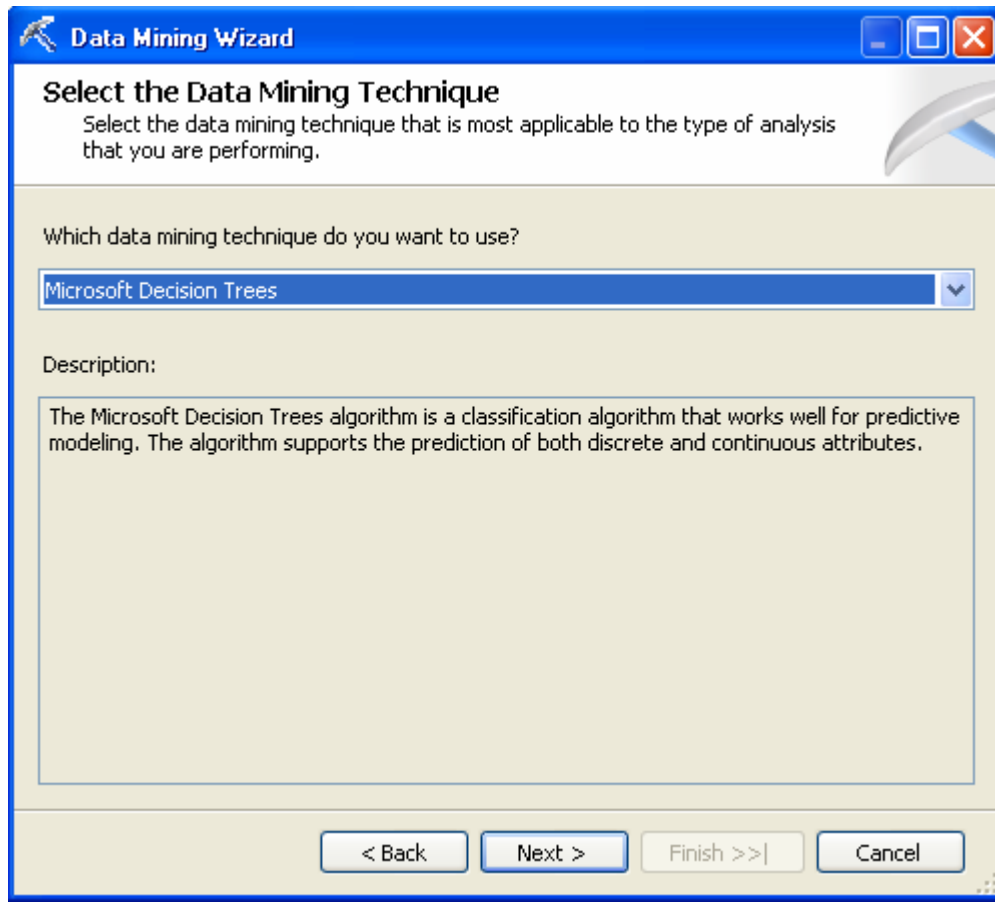


Click Next chọn **From existing relational database or data warehouse**

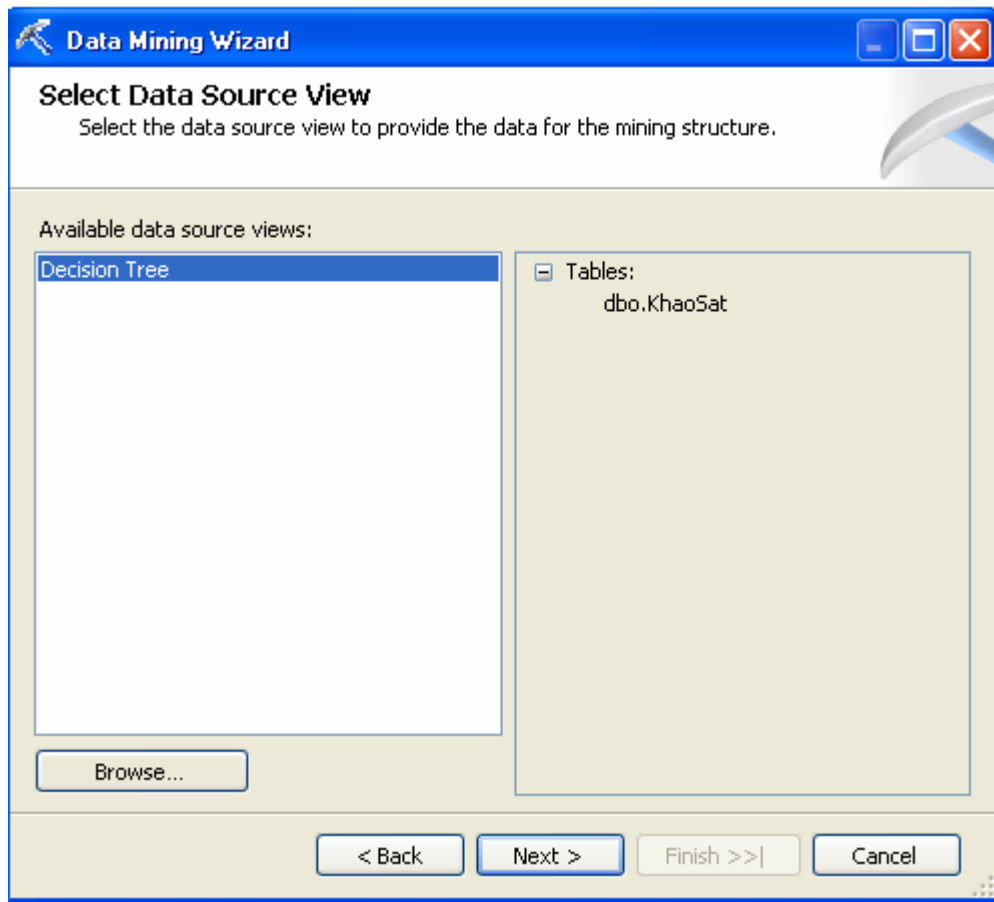


Click Next

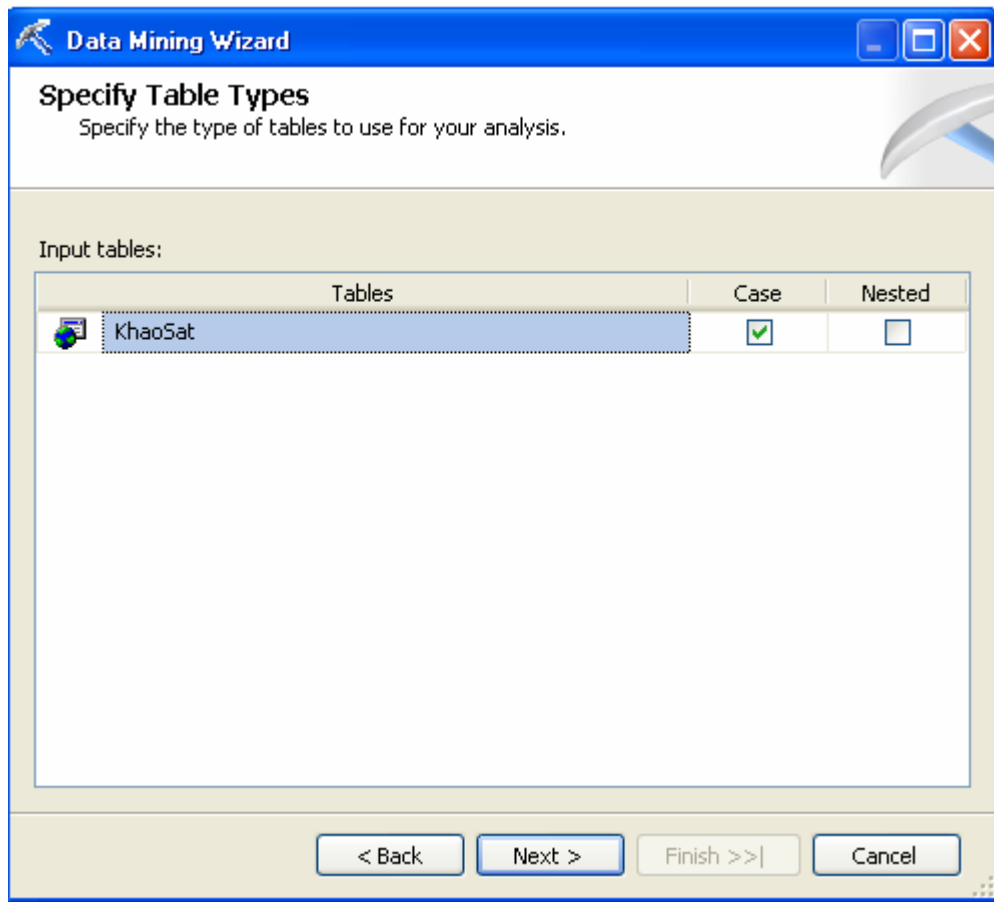
Chọn thuật toán sử dụng



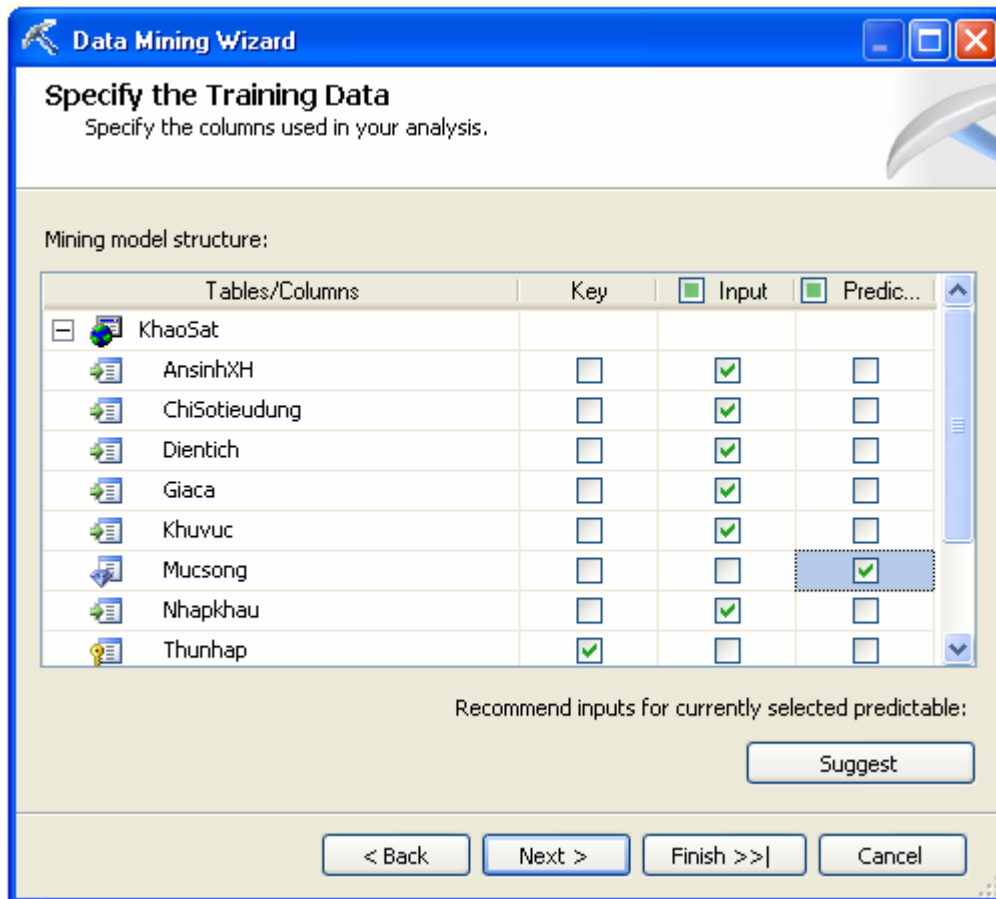
Chọn Data Source View



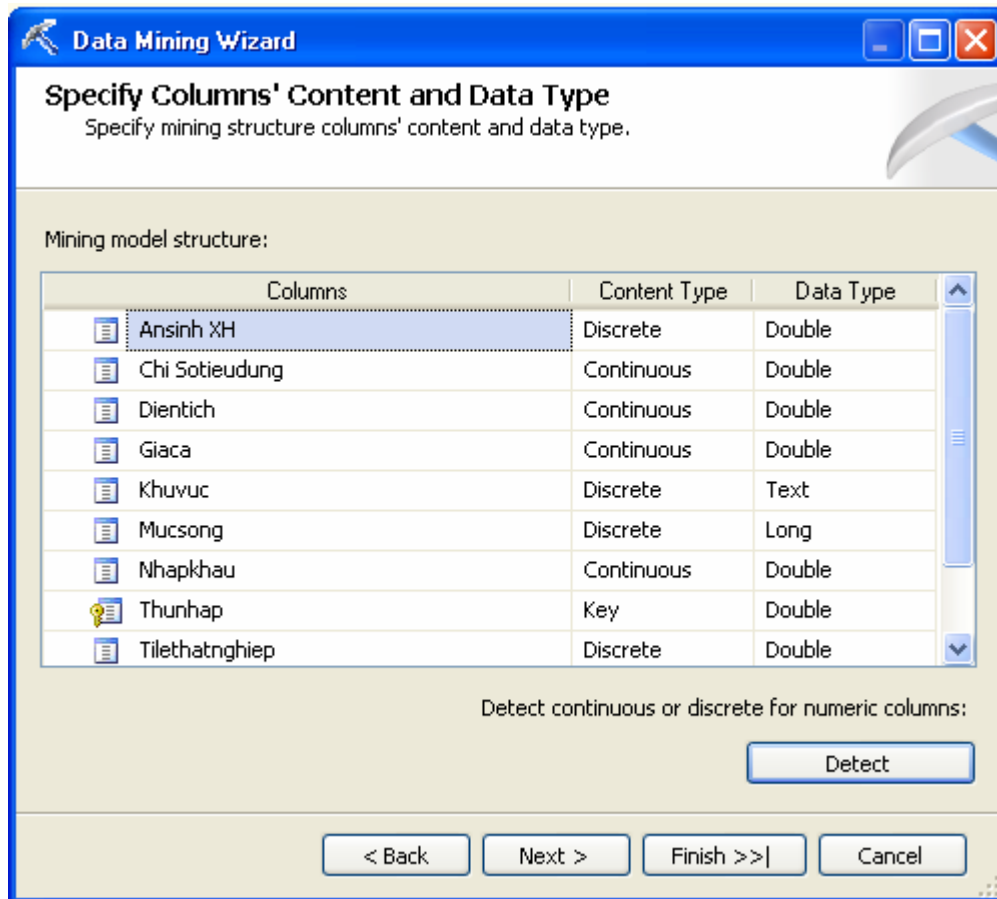
Click Next chọn bảng Case (cha) và bảng Nested (con)



Click Next



Click Next



The image shows a Windows-style dialog box titled "Data Mining Wizard" with a subtitle "Specify Columns' Content and Data Type". Below the subtitle is a description: "Specify mining structure columns' content and data type." The main area is labeled "Mining model structure:" and contains a table with three columns: "Columns", "Content Type", and "Data Type". The table lists ten columns: "Ansinh XH", "Chi Sotieudung", "Dientich", "Giaca", "Khuvuc", "Mucsong", "Nhapkhu", "Thunhap", and "Tilethatnghiep". Each column has a corresponding "Content Type" and "Data Type" listed. For example, "Ansinh XH" is Discrete with Data Type Double, while "Thunhap" is Key with Data Type Double. Below the table, there is a button labeled "Detect" with the text "Detect continuous or discrete for numeric columns:" above it. At the bottom of the dialog, there are four buttons: "< Back", "Next >", "Finish >>|", and "Cancel".

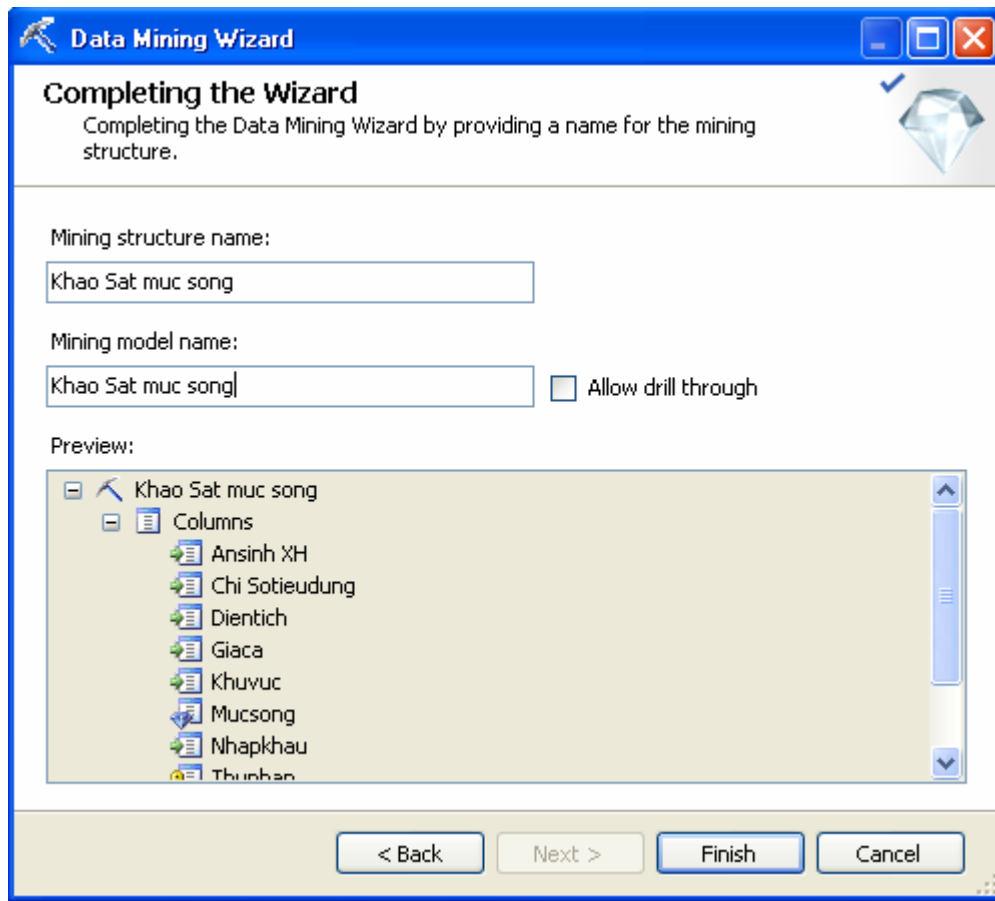
Columns	Content Type	Data Type
Ansinh XH	Discrete	Double
Chi Sotieudung	Continuous	Double
Dientich	Continuous	Double
Giaca	Continuous	Double
Khuvuc	Discrete	Text
Mucsong	Discrete	Long
Nhapkhu	Continuous	Double
Thunhap	Key	Double
Tilethatnghiep	Discrete	Double

Detect continuous or discrete for numeric columns:

Detect

< Back Next > Finish >>| Cancel

Click Next và đặt tên



Click Finish

TÀI LIỆU THAM KHẢO

- [1] Microsoft Corporation - MSDN – Nhà xb – năm xb**
- [2] Nguyễn Thiện Bằng (Chủ biên) – Phương Lan (Hiệu đính) – Khám phá SQL Server 2005 – Nhà xuất bản lao động xã hội – 2006**
- [3] Zhao Hui Tang – Jamie Mac Lennan – Data Mining With SQL Server 2005**