**KHAI PHÁ DỮ LIỆU**

**Câu 1:** Tổng quan về khai phá dữ liệu

* Khái niệm KPDL:
* Khai phá dữ liệu (data mining) là một tập hợp, một hệ thống các phương pháp tính toán, thuật toán được áp dụng cho các cơ sở dữ liệu lớn và phức tạp mục đích là loại bỏ các chi tiết ngẫu nhiên, chi tiết ngoại lệ, khám phá các mẫu, mô hình, quy luật tiềm ẩn các thông tin có giá trị trong bộ dữ liệu. Data mining là thành quả công nghệ tiên tiến ngày nay, là quá trình khám phá các kiến thức vô giá bằng cách phân tích khối lượng lớn dữ liệu đồng thời lưu trữ chúng ở nhiều cơ sở dữ liệu khác nhau.
* Các lĩnh vực liên quan:
* Ứng dụng trong các lĩnh vực tài chính, ngân hàng
* Trong lĩnh vực y tế, chăm sóc sức khỏe
* Trong lĩnh vực viễn thông
* Trong lĩnh vực Sales và Maeketing
* Trong lĩnh vực E-commerce
* Trong lĩnh vực bán lẻ
* Ngoài ra: Lĩnh vực sinh học; Lĩnh vực bảo mật, phòng chống tội phạm; Lĩnh vực giáo dục;
* Phân biệt KPDL với tìm kiếm thông thường:

|  |  |  |
| --- | --- | --- |
| Nội dung | KPDL | Tìm kiếm thông thường |
| Định nghĩa | Là một tập hợp các kỹ thuật được sử dụng để tự động khai thác và tìm ra các mối quan hệ lẫn nhau của dữ liệu trong một tập hợp dữ liệu khổng lồ và phức tạp, đồng thời cũng tìm ra các mẫu tiềm ẩn trong tập dữ liệu đó. | Là các hệ thống được xây dựng có khả năng tiếp nhận các yêu cầu tìm kiếm của người dùng(thường là một tập hợp các từ khóa), sau đó phân tích và tìm kiếm trong cơ sở dữ liệu đã có sẵn và đưa ra các kết quả là các trang web cho người dùng sử dụng. |
| Ưu điểm | -Chắt lọc thông tin  -Độ chính xác cao  -Cập nhật lại các tham số tìm kiếm tăng hiệu quả cho lần tìm kiếm sau. | * Nhanh, ít tốn thời gian. |
| Nhược điểm | - Tốn thời gian học máy. | -Là các hệ tìm kiếm tự động, không có cơ chế phản hồi từ người sử dụng để cập nhật các tham số tìm kiếm nhằm tăng hiệu quả cho lần tìm kiếm sau.  - không lọc đầu vào.  - Chưa quan tâm đến bản chất của xử lý văn bản, vấn đề từ đồng nghĩa, đa nghĩa.  - Độ chính xác không cao. |

* Phát biểu quy trình:

**1.** **Làm sạch dữ liệu (data cleaning & preprocessing)s:** Loại bỏ nhiễu và các dữ liệu không cần thiết.

**2.** **Tích hợp dữ liệu (data integration):** quá trình hợp nhất dữ liệu thành những kho dữ liệu (data warehouses & data marts) sau khi đã làm sạch và tiền xử lý (data cleaning & preprocessing).

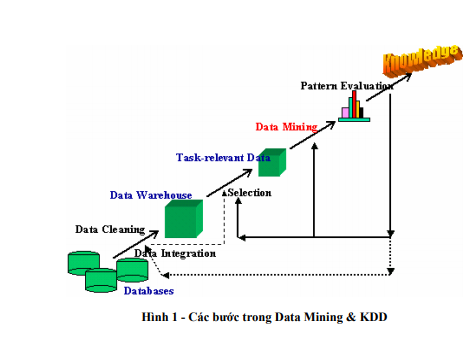
**3.** **Trích chọn dữ liệu (data selection):** trích chọn dữ liệu từ những kho dữ liệu và sau đó chuyển đổi về dạng thích hợp cho quá trình khai thác tri thức. Quá trình này bao gồm cả việc xử lý với dữ liệu nhiễu (noisy data), dữ liệu không đầy đủ (incomplete data), .v.v.

**4.** **Chuyển đổi dữ liệu:** Các dữ liệu được chuyển đổi sang các dạng phù hợp cho quá trình xử lý

**5.** **Khai phá dữ liệu(data mining):** Là một trong các bước quan trọng nhất, trong đó sử dụng những phương pháp thông minh để chắt lọc ra những mẫu dữ liệu.

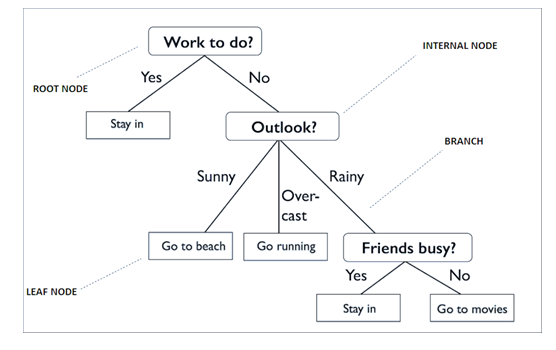
**6.** **Ước lượng mẫu (knowledge evaluation):** Quá trình đánh giá các kết quả tìm được thông qua các độ đo nào đó.

**7. Biểu diễn tri thức (knowledge presentation):** Quá trình này sử dụng các kỹ thuật để biểu diễn và thể hiện trực quan cho người dùng.



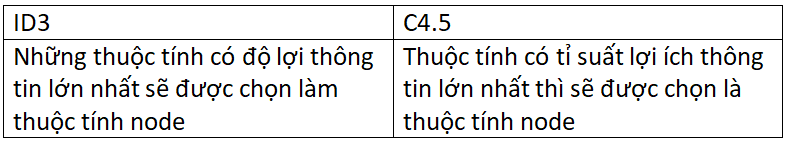
**Câu 2:** Kỹ thuật phân lớp

* Khái niệm:
* Là phương pháp dự báo, cho phép phân loại một đối tượng vào một hoặc một số lớp cho trước.
* Các miền ứng dụng:
* Bài toán nhận dạng khuôn mặt.
* Nhận diện giọng nói.
* Phát hiện email spam
* Cây quyết định: Nêu ý tưởng
* Khái niệm: Cây quyết định (Decision Tree) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật (series of rules). Các thuộc tính của đối tượng (ngoại trừ thuộc tính phân lớp – Category attribute) có thể thuộc các kiểu dữ liệu khác nhau (Binary, Nominal, ordinal, quantitative values) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary(nhị phân)hoặc Ordinal.
* Ý tưởng: cây quyết định mô tả một cấu trúc cây, trong đó, các lá đại diện cho các phân loại còn cành đại diện cho các kết hợp của các thuộc tính dẫn tới phân loại đó. Một cây quyết định có thể được học bằng cách chia tập hợp nguồn thành các tập con dựa theo một kiểm tra giá trị thuộc tính. Quá trình này được lặp lại một cách đệ quy cho mỗi tập con dẫn xuất. Quá trình đệ quy 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.



* “Root node” : Điểm ngọn Chứa giá trị của biến đầu tiên được dùng để phân nhánh.
* “Internal node” : Các điểm bên trong thân cây là các biến chứa các thuộc tính, giá trị dữ liệu được dùng để xét cho phân nhánh tiếp theo.
* “Leaf node” : Là các lá cây chứa các giá trị của biến phân loại sau cùng.
* “Branch”: Là quy luật phân nhánh, nói đơn giản là mối quan hệ giữa giá trị của biến độc lập và giá trị của biến mục tiêu.
* So sánh ID3 và C4.5:
* Thuật toán ID3: Trong ID3, tổng có trọng số của entropy tại các leaf-node sau khi xây dựng decision tree được coi là hàm mất mát của decision tree đó. Các trọng số ở đây tỉ lệ với số điểm dữ liệu được phân vào mỗi node. Công việc của ID3 là tìm các cách phân chia hợp lý (thứ tự chọn thuộc tính hợp lý) sao cho hàm mất mát cuối cùng đạt giá trị càng nhỏ càng tốt. Như đã đề cập, việc này đạt được bằng cách chọn ra thuộc tính sao cho nếu dùng thuộc tính đó để phân chia, entropy tại mỗi bước giảm đi một lượng lớn nhất. Bài toán xây dựng một decision tree bằng ID3 có thể chia thành các bài toán nhỏ, trong mỗi bài toán, ta chỉ cần chọn ra thuộc tính giúp cho việc phân chia đạt kết quả tốt nhất. Mỗi bài toán nhỏ này tương ứng với việc phân chia dữ liệu trong một non-leaf node.
* 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 Nghiên cứu các thuật toán phân lớp dữ liệu dựa trên cây quyết định 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.

|  |  |  |
| --- | --- | --- |
|  | **ID3** | **C4.5** |
|  | -Là một thuật toán decision tree được áp dụng cho các bài toán classification mà tất cả các thuộc tính đều ở dạng categorical. | - 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. |
|  | -Sử dụng lượng thông tin ứng với biến số phân loại sau đó dùng kỹ thuật tham lam. | -không cần biến số phân loại lượng đặc trưng |
|  |  | -Output theo dạng if-then, không hiển thị những phần cành không cần thiết. |
|  |  | Xử lý thuộc tính thực: rời rạc hóa hoặc rẽ nhánh |



* **VD:**

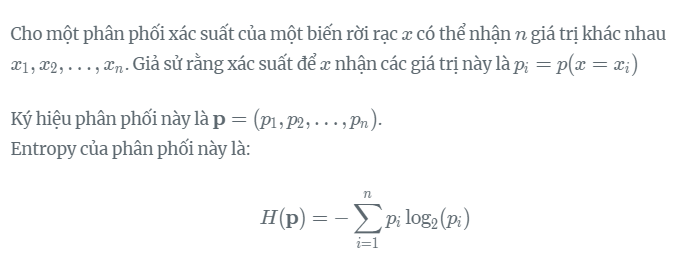
**ID3:** Ta có tập Training Data như bảng dưới:

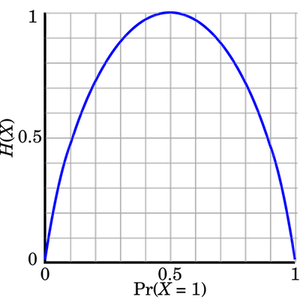
| **ID** | **Engine** | **Type** | **Color** | **4WD** | **Want?** |
| --- | --- | --- | --- | --- | --- |
| 1 | 2000cc | SUV | Silver | Yes | Yes |
| 2 | 1000cc | Sedan | Silver | Yes | Yes |
| 3 | 2000cc | Sport | Blue | No | No |
| 4 | 1000cc | SUV | Blue | No | Yes |
| 5 | 2000cc | Sedan | Silver | Yes | No |
| 6 | 2000cc | Sport | Blue | Yes | Yes |
| 7 | 1000cc | Sedan | Blue | No | Yes |
| 8 | 1000cc | SUV | Silver | No | Yes |

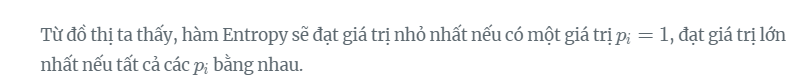
Data của ta có 4 thuộc tính: Engine, Type, Color, 4WD.  
Để tính toán được DT, ta cần phân chia các thuộc tính vào các node đánh giá. Vậy làm sao để biết được thuộc tính nào quan trọng, nên đặt ở gốc, thuộc tính nào ở nhánh…

Trong thuật toán ID3, các thuộc tính được đánh giá dựa trên Hàm số Entropy, hàm số phổ biến trong toán học xác suất.

**Hàm số Entropy**

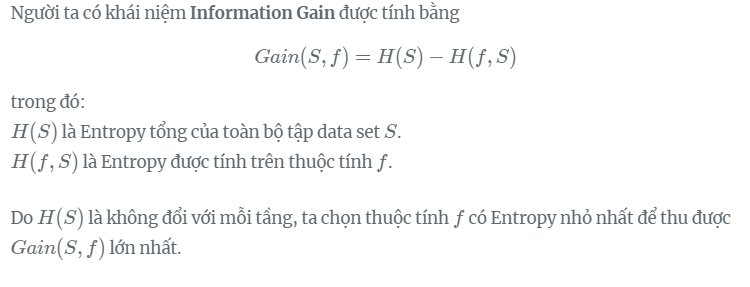
Hàm Entropy được biểu diễn dưới dạng đồ thị như sau:



Hàm Entropy càng lớn thì độ ngẫu nhiên của các biến rời rạc càng cao (càng không tinh khiết).

Với cây quyết định, ta cần tạo cây như thế nào để cho ta nhiều thông tin nhất, tức là Entropy là cao nhất.

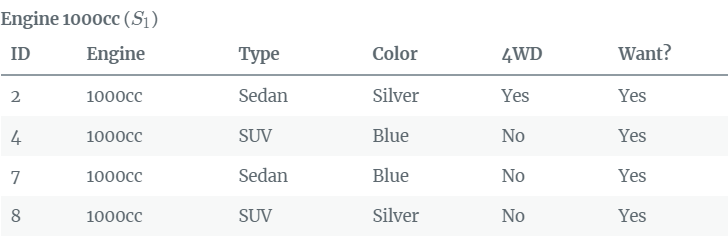
**Information Gain**

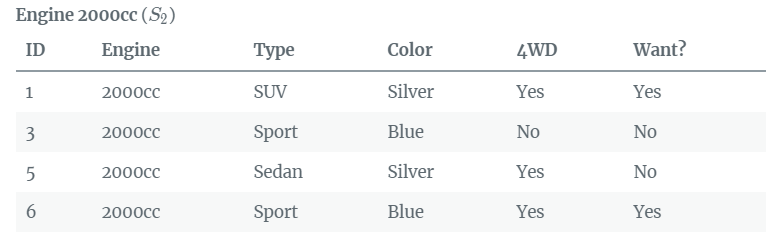
Bài toán của ta trở thành, tại mỗi tầng của cây, cần chọn thuộc tính nào để độ giảm Entropy là thấp nhất.  
**Tính Entropy của các thuộc tính**

**Xét thuộc tính Engine**

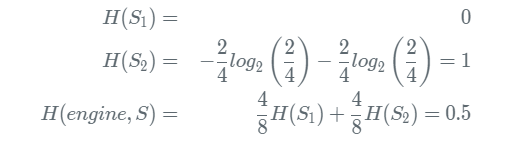
Thuộc tính này có thể nhận 1 trong 2 giá trị 1000cc, 2000cc, tương ứng với 2 child node.  
Gọi tập hợp các điểm trong mỗi child node này lần lượt là S\_1*S*1​, S\_2*S*2​.

Sắp xếp lại theo thuộc tính **Engine** ta có 2 bảng nhỏ.



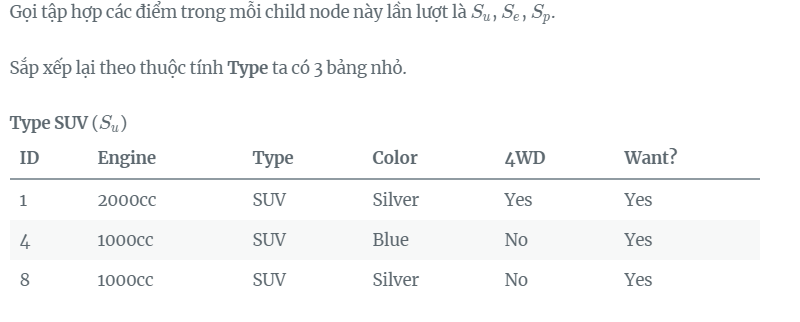


Child node ứng với Engine 1000cc sẽ có Entropy = 0 do tất cả các giá trị đều là **Yes**.  
Ta chỉ việc tính Entropy với Engine 2000cc. Sau đó tính Entropy trung bình.  
Cụ thể như sau:

****

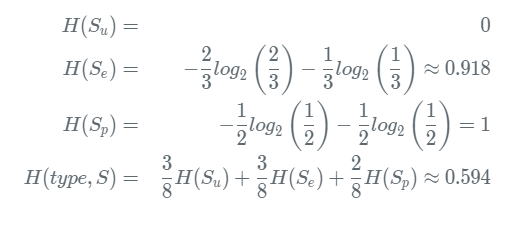
**Xét thuộc tính Type**

Thuộc tính này có thể nhận 1 trong 3 giá trị SUV, Senda, Sport tương ứng với 3 child node.



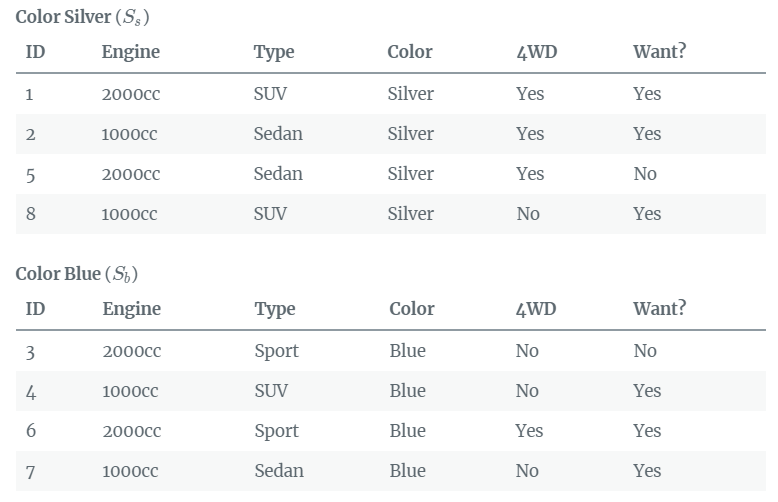
  
Tương tự, ta lần lượt tính Entropy như bên dưới:

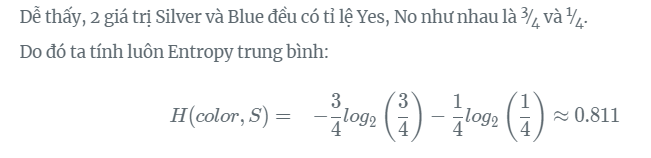
**Xét thuộc tính Color**

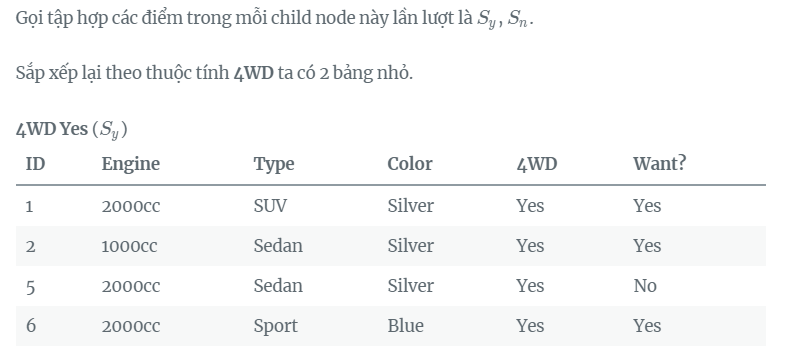
****

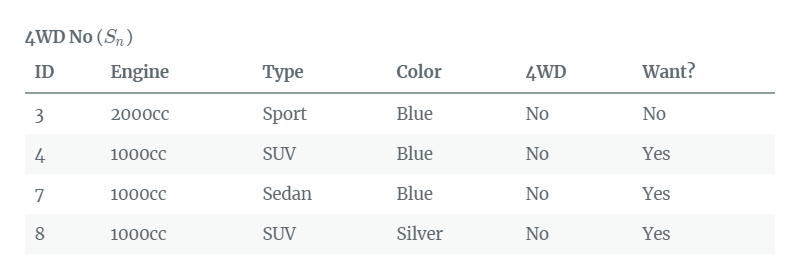
Thuộc tính này có thể nhận 1 trong 2 giá trị Silver, Blue tương ứng với 2 child node.

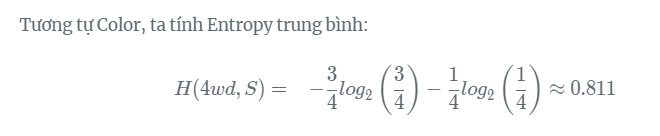
  
Sắp xếp lại theo thuộc tính **Color** ta có 2 bảng nhỏ.



**Xét thuộc tính 4WD**

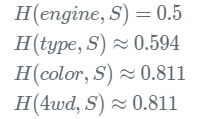
Thuộc tính này có thể nhận 1 trong 2 giá trị Yes, No tương ứng với 2 child node.  






**Chọn thuộc tính có Entropy nhỏ nhất**

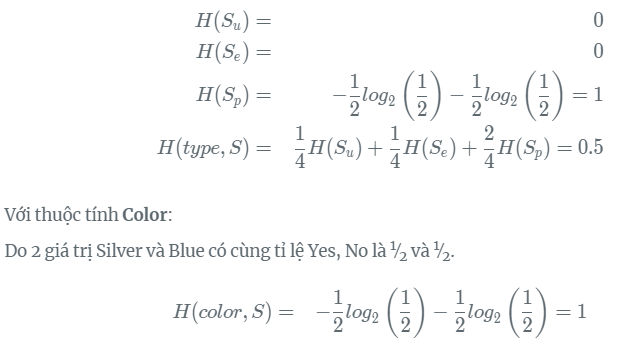
Sau khi tính Entropy trung bình của 4 thuộc tính ta thu được:

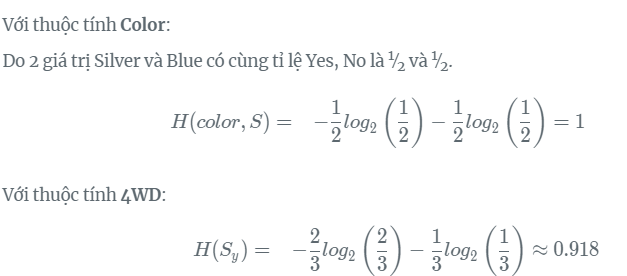


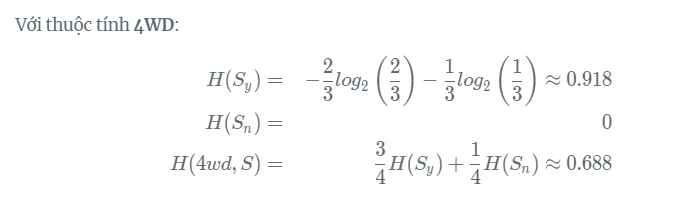
Thuộc tính **Engine** có giá trị Entropy nhỏ nhất nên ta chọn là node đánh giá đầu tiên.  
Với Engine 1000cc, tất cả các data đều có giá trị Yes, vì vậy ta thu được node là Yes ở nhánh 1000cc.  
Ta tiếp tục tính cho nhánh Engine 2000cc với tập data nhỏ hơn là

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Tương tự ta lần lượt tính Entropy cho 3 thuộc tính là: Type, Color, 4WD

Với thuộc tính **Type**:  




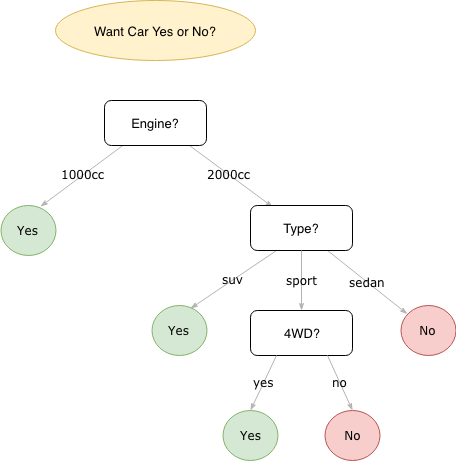


Vậy ta chọn **Type** là node đánh giá tiếp theo.

Với trường hợp Type là SUV hoặc Sedan, ta có ngay node lá vì chỉ có một kết quả.  
Với trường hợp Type là Sport, do thuộc tính Color là giống nhau với tất cả data, ta chọn node đánh giá tiếp theo là **4WD**.

**Kết quả**

Ta thu được Decision Tree như hình bên dưới.



**Kiểm tra (Validation)**

Ta sẽ tiến hành kiểm tra mô hình DT ta vừa tạo được bằng tập Test Data như bên dưới:

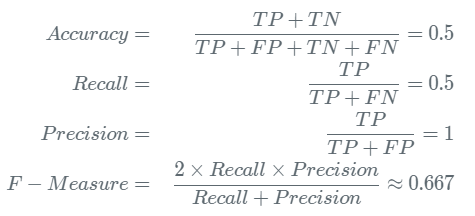
| **ID** | **Engine** | **Type** | **Color** | **4WD** | **Want?** |
| --- | --- | --- | --- | --- | --- |
| 9 | 2000cc | Sedan | Silver | Yes | Yes |
| 10 | 2000cc | Sport | Silver | No | No |
| 11 | 1000cc | SUV | Blue | Yes | Yes |
| 12 | 2000cc | Sedan | Blue | No | Yes |

Ta có bảng mapping đánh giá kết quả như sau:

| **Actual Result** | **Estimate Result** | **Validation** |
| --- | --- | --- |
| Yes | Yes | True Positive (TP) |
| No | No | True Negative (TN) |
| Yes | No | False Negative (FN) |
| No | Yes | False Positive (FP) |

Dựa vào DT ta vừa tạo được, ta tiến hành đánh giá như sau:

| **ID** | **Actual Result** | **Estimate Result** | **Validation** |
| --- | --- | --- | --- |
| 9 | Yes | No | False Negative (FN) |
| 10 | No | No | True Negative (TN) |
| 11 | Yes | Yes | True Positive (TP) |
| 12 | Yes | No | False Negative (FN) |

Các thông số áp dụng để đánh giá được tính như sau:  


Nhìn chung Decision Tree tìm được có độ chính xác không cao khi chạy thử với Test Data.  
Nguyên nhân chính có lẽ là do tập Training Data quá ít.

* **C4.5:**
* **Rừng ngẫu nhiên:**
* ***Khái niệm:*** Là một bộ nhận dạng bao gồm một tập bộ phân lớp cơ sở dạng cây quyết định được kết hợp theo phương thức bỏ phiếu.
* Khi dữ liệu chiều cao, xây dựng cây lâu và chất lượng nhận dạng (Phân lớp /hồi quy) thấp. Người ta dùng rừng ngẫu nhiên.
* ***Ý tưởng:*** Rừng ngẫu nhiên hoạt động bằng cách đánh giá nhiều cây quyết định ngẫu nhiên, và lấy kết quả được đánh giá tốt nhất trong số kết quả trả về.

Rừng ngẫu nhiên là một thuật toán học có giám sát. Như tên gọi của nó, Rừng ngẫu nhiên sử dụng các cây (tree) để làm nền tảng.

Rừng ngẫu nhiên là một tập hợp của các Decision Tree, mà mỗi cây được chọn theo một thuật toán dựa vào ngẫu nhiên.

Random Forest hoạt động bằng cách đánh giá nhiều Cây quyết định ngẫu nhiên, và lấy ra kết quả được đánh giá tốt nhất trong số kết quả trả về.

* Mã giả cho hoạt động của rừng ngẫu nhiên:

1. Chọn ngẫu nhiên “k” features từ tập “m” features.

Để ý k << m

1. Từ tập “k” features, tính toán ra node “d” là tốt nhất cho Node phân loại.
2. Chia các node con theo node tốt nhất vừa tìm được
3. Lặp lại bước 1-3 cho đến khi đạt đến k node
4. Lặp lại bước 1-4 để tạo ra “n” cây

* **Áp dụng với dữ liệu cho trước(dùng công cụ):**
* **KNN:**
* ***Khái niệm:*** KNN là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai phá dữ liệu và học máy.
* ***Ý tưởng:*** Ý tưởng là không học một điều gì từ tập dữ liệu học, mọi tính toán được thực hiện khi nó cần dự đoán nhãn của dữ liệu mới. Lớp của một đối tượng dữ liệu mới có thể dự đoán từ các lớp của k hàng xóm gần nhất.
* KNN giả định rằng những thứ tượng tự nhau hay có tính chất gần giống nhau sẽ nằm ở vị trí gần nhau. Với giả định như vậy, KNN xây dựng trên các công thức toán học phục vụ để tính khoảng cách giữa 2 điểm dữ liệu (gọi là data points) để xem xét mức độ giống nhau giữa chúng.

- Công thức tính khoảng cách giữa 2 điểm:

**Thuật toán K-NN được mô tả như sau:**

1.    Xác định giá trị tham số K (số láng giềng gần nhất)

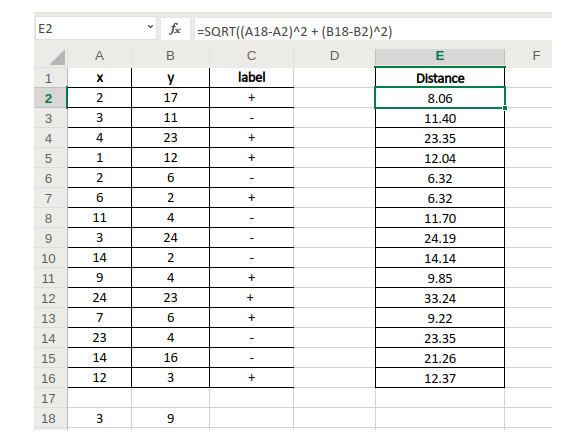
2.    Tính khoảng cách giữa đối tượng cần phân lớp (Query Point) với tất cả các đối tượng trong training data (thường sử dụng khoảng các Euclidean)

3.    Sắp xếp khoảng cách theo thứ tự tăng dần và xác định K láng giềng gần nhất với Query Point

4.    Lấy tất cả các lớp của K láng giềng gần nhất đã xác định

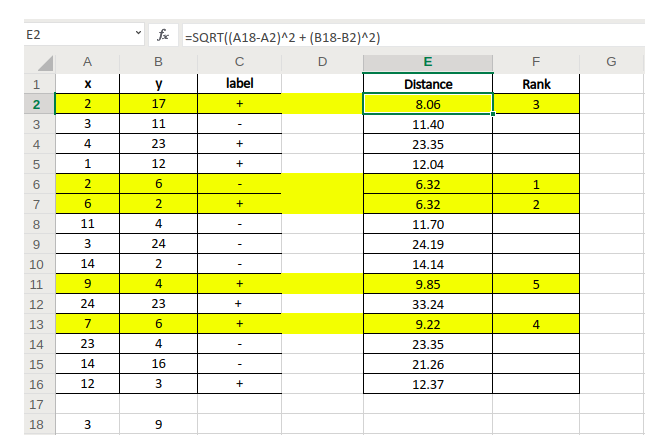
5.    Dựa vào phần lớn lớp của láng giềng gần nhất để xác định lớp cho Query Point

* **Áp dụng với dữ liệu cho trước:**

****

Giả sử ta có tập dữ liệu D có gắn nhãn gồm 15 điểm như trên ảnh.

1. Điểm cần dự đoán nhãn A(3,9)
2. Ta tính khoảng cách từ điểm A đến các điểm dữ liệu trong D bằng công thức Euclidian.
3. Ta chọn K= 5, và tìm ra 5 điểm có khoảng cách gần với điểm A nhất.
4. Trong 5 điểm ta thấy có 4 điểm mang nhãn (+) và 1 điểm mang nhãn (-).
5. Vậy ta có thể đưa ra kết luận là điểm A cần dự đoán mang nhãn (+).

******

# Ưu điểm

1. Thuật toán đơn giản, dễ dàng triển khai.
2. Độ phức tạp tính toán nhỏ.
3. Xử lý tốt với tập dữ liệu nhiễu

# Nhược điểm

1. Với K nhỏ dễ gặp nhiễu dẫn tới kết quả đưa ra không chính xác
2. Cần nhiều thời gian để thực hiện do phải tính toán khoảng cách với tất cả các đối tượng trong tập dữ liệu.
3. Cần chuyển đổi kiểu dữ liệu thành các yếu tố định tính.

***Tự phát biểu bài toán và cho dữ liệu:***

* ***Naïve Bayes:***
* ***Khái niệm:*** Bayes là một thuật toán phân lớp được mô hình hóa dựa trên định lý Bayes trong xác suất thống kê.
* ***Ý tưởng:***

Xét bài toán phân lớp có C classes 1…C. Giả sử có một điểm dữ liệu x. Tính xác suất để điểm dữ liệu này rơi vào class c, nói cách khác, tính:

Hoặc viết gọn thành:

* **VD**:

### Ví dụ 1:

    Phân các bệnh nhân thành 2 lớp ung thư và không ung thư. Giả sử xác suất để một người bị ung thư là 0.008 tức là P(cancer) = 0.008; và P(nocancer) = 0.992. Xác suất để bệnh nhân ung thư có kết quả xét nghiệm dương tính là 0.98 và xác suất để bệnh nhân không ung thư có kết quả dương tính là 0.03 tức là P(+/cancer) = 0.98, P(+/nocancer) = 0.03. Bây giờ giả sử một bệnh nhân có kết quả xét nghiệm dương tính. Ta có:  
P(+/canncer)P(cancer) = 0.98 \* 0.008 = 0.0078  
P(+/nocancer)P(nocancer) = 0.03 \* 0.992 = 0.0298  
Như vậy, P(+/nocancer)P(nocancer) >> P(+/cancer)P(cancer).  
Do đó ta xét đoán rằng, bệnh nhân là không ung thư.

### Ví dụ 2:

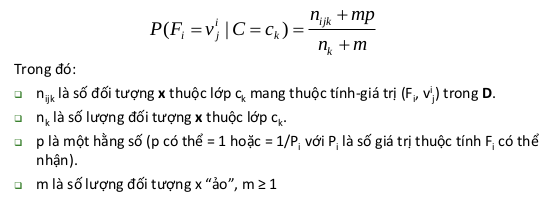
Cơ sở dữ liệu khách hàng:

| **ID** | **Tuổi** | **Thu nhập** | **Sính viên** | **Đánh giá tín dụng** | **Mua máy tính** |
| --- | --- | --- | --- | --- | --- |
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle | low | yes | excellent | yes |
| 8 | youth | medum | no | fair | yes |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle | medium | no | excellent | yes |
| 13 | middle | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

Giả sử ta có một khách hàng mới X có các thuộc tính  
X = (age = youth, income = medium, student = yes, credit\_rating = fair)  
Bây giớ cần xác định xem khách hàng X có thuộc lớp Cyes (mua máy tính) hay không, ta tính toán như sau:  
P(Cyes) = 9/14 = 0.357  
Các xác suất thành phần:  
P(age = youth|Cyes) = 2/9 = 0.222  
P(age = youth|Cno) = 3/5 = 0.6  
P(income = medium|Cyes) = 4/9 = 0.444  
P(income = medium|Cno) = 2/5 = 0.4  
P(student = yes|Cyes) = 6/9 = 0.667  
P(student = yes|Cno) = 1/5 = 0.2  
P(credit\_rating = fair|Cyes) = 6/9 = 0.667  
P(credit\_rating = fair|Cno) = 2/5 = 0.2  
Cuối cùng:  
P(X|Cyes) = 0.222 \* 0.444 \* 0.667 \* 0.667 = 0.044  
P(X|Cno) = 0.60.4 \* 0.2 \* 0.4 = 0.019  
P(X|Cyes)\*P(Cyes) = 0.044 \* 0.643  
P(X|Cno)\*P(Cno) =0.019 \* 0.357 = 0.007

    Từ kết quả này ta thấy P(X |Cyes)P(Cyes) có giá trị lớn nhất, do đó thuật toán Bayes sẽ kết luận là khách hàng X sẽ mua máy tính.

* **Khắc phục vấn đề xác suất điều kiện bằng zero**
* Nếu trong dữ liệu huấn luyện không có đối tượng X nào có thuộc tính lớp Ck có thuộc tính Fi nhận một giá trị cụ thể vij, xác suất điều kiện P(Fi = xij | Ck) sẽ bằng 0.
* Khi phân lớp, nếu có một đối tượng nào mang thuộc tính này thì xác suất phân vào lớp Ck luôn bằng 0.
* Khắc phục bằng cách ước lượng theo công thức sau:



* **Ưu điểm**

1. Giả định độc lập: hoạt động tốt cho nhiều bài toán/miền sữ liệu và ứng dụng.  
   Đơn giản nhưng đủ tốt để giải quyết nhiều bài toán như phân lớp văn bản, lọc spam,..
2. Cho phép kết hợp tri thức tiền nghiệm (prior knowledge) và dữ liệu quan sát được (obserwed data).  
   Tốt khi có sự chệnh lệch số lượng giữa các lớp phân loại.
3. Huấn luyện mô hình (ước lượng tham số) dễ và nhanh.

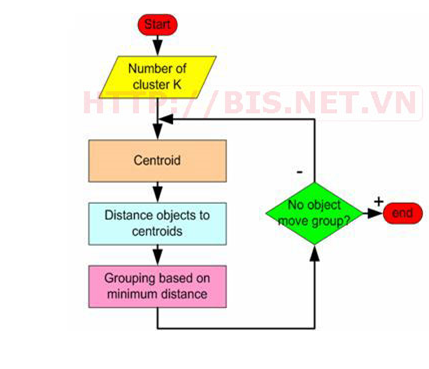
* **Nhược điểm**

1. Giả định độc lập (ưu điểm cũng chính là nhược điểm)  
   hầu hết các trường hợp thực tế trong đó có các thuộc tính trong các đối tượng thường phụ thuộc lẫn nhau.
2. Vấn đề zero (đã nêu cách giải quyết ở phía trên)
3. Mô hình không được huẩn luyện bằng phượng pháp tối ưu mạnh và chặt chẽ.  
   Tham số mủa mô hình là các ước lượng xác suất điều kiện đơn lẻ.  
   Không tính đến sự tương tác giữa các ước lượng này.

***CÂU 3:KỸ THUẬT PHÂN CỤM***

* ***Khái niệm:***
* Phân cụm là kỹ thuật rất quan trọng trong khai phá dữ liệu, nó thuộc lớp các phương pháp **Unsupervised Learning** trong Machine Learning. Có rất nhiều định nghĩa khác nhau về kỹ thuật này, nhưng về bản chất ta có thể hiểu phân cụm là các qui trình tìm cách nhóm các đối tượng đã cho vào các cụm (clusters), sao cho các đối tượng trong cùng 1 cụm tương tự (similar) nhau và các đối tượng khác cụm thì không tương tự (Dissimilar) nhau.
* Mục đích của phân cụm là tìm ra bản chất bên trong các nhóm của dữ liệu. Các thuật toán phân cụm (Clustering Algorithms) đều sinh ra các cụm (clusters). Tuy nhiên, không có tiêu chí nào là được xem là tốt nhất để đánh hiệu của của phân tích phân cụm, điều này phụ thuộc vào mục đích của phân cụm như: data reduction, “natural clusters”, “useful” clusters, outlier detection
* ***Các miền ứng dụng:***
* Ứng dụng:
* Marketing: Giúp các nhà tiếp thị nhóm các khách hàng của họ, từ đó đưa ra những chương trình tiếp thị phù hợp với từng nhóm.
* Sinh vật: Xác định các sinh vật, phân loại GEN
* Địa lý: Phân lớp động vật và thực vật, đưa ra các đặc trưng của chúng
* ***Kmen:*** Là thuật toán rất quan trọng và được sử dụng phổ biến trong kỹ thuật phân cụm.
* ***Ý tưởng:*** Tư tưởng chính của thuật toán K-Means là tìm cách phân nhóm các đối tượng (objects) đã cho vào K cụm (K là số các cụm được xác đinh trước, K nguyên dương) sao cho tổng bình phương khoảng cách giữa các đối tượng đến tâm nhóm (centroid ) là nhỏ nhất.
* ***Áp dụng với dữ lieuj cho trước:***
* ***Tự phát biểu bài toán và cho dữ liệu:***

***Thuật toán K-Means được mô tả như sau:***



***Thuật toán K-Means thực hiện qua các bước chính sau:***

1.    Chọn ngẫu nhiên K tâm (centroid) cho K cụm (cluster). Mỗi cụm được đại diện bằng các tâm của cụm.

2.    Tính khoảng cách giữa các đối tượng (objects) đến K tâm (thường dùng khoảng cách Euclidean)

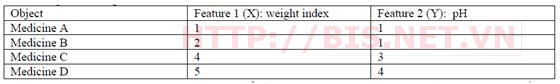
3.    Nhóm các đối tượng vào nhóm gần nhất

4.    Xác định lại tâm mới cho các nhóm

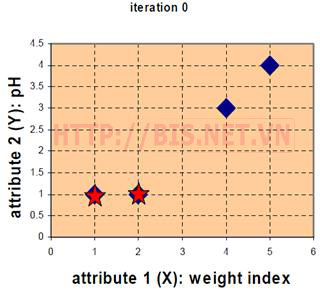
5.    Thực hiện lại bước 2 cho đến khi không có sự thay đổi nhóm nào của các đối tượng

***Ví dụ minh họa thuật toán K-Mean:***

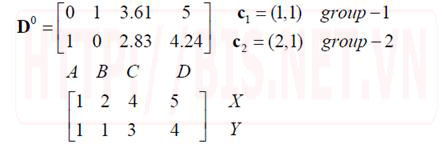
Giả sử ta có 4 loại thuốc A,B,C,D, mỗi loại thuộc được biểu diễn bởi 2 đặc trưng X và Y như sau. Mục đích của ta là nhóm các thuốc đã cho vào 2 nhóm (K=2) dựa vào các đặc trưng của chúng.



**Bước 1.** Khởi tạo tâm (centroid) cho 2 nhóm. Giả sử ta chọn A là tâm của nhóm thứ nhất (tọa độ tâm nhóm thứ nhất c1(1,1)) và B là tâm của nhóm thứ 2 (tạo độ tâm nhóm thứ hai c2 (2,1)).



**Bước 2.** Tính khoảng cách từ các đối tượng đến tâm của các nhóm (Khoảng cách Euclidean)



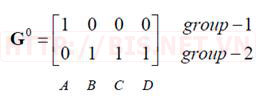
Mỗi cột trong ma trận khoảng cách (D) là một đối tượng (cột thứ nhất tương ứng với đối tượng A, cột thứ 2 tương ứng với đối tượng B,…). Hàng thứ nhất trong ma trận khoảng cách biểu diễn khoảng cách giữa các đối tượng đến tâm của nhóm thứ nhất (c1) và hàng thứ 2 trong ma trận khoảng cách biểu diễn khoảng cách của các đối tượng đến tâm của nhóm thứ 2 (c2).

Ví dụ, khoảng cách từ loại thuốc C=(4,3) đến tâm c1(1,1) là 3.61  và đến tâm c2(2,1) là 2.83 được tính như sau:





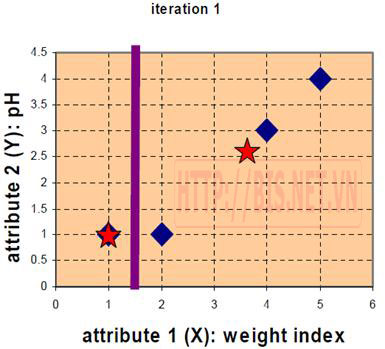
**Bước 3.** Nhóm các đối tượng vào nhóm gần nhất



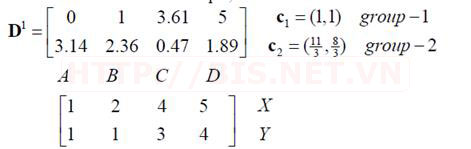
Ta thấy rằng  nhóm 1 sau vòng lặp thứ nhất gồm có 1 đối tượng A và nhóm 2 gồm các đối tượng còn lại B,C,D.

**Bước 5.** Tính lại tọa độ các tâm cho các nhóm mới dựa vào tọa độ của các đối tượng trong nhóm. Nhóm 1 chỉ có 1 đối tượng A nên tâm nhóm 1 vẫn không đổi, c1(1,1). Tâm nhóm 2 được tính như sau:

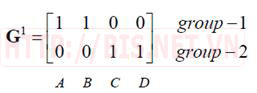




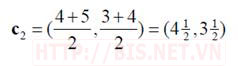
**Bước 6.** Tính lại khoảng cách từ các đối tượng đến tâm mới

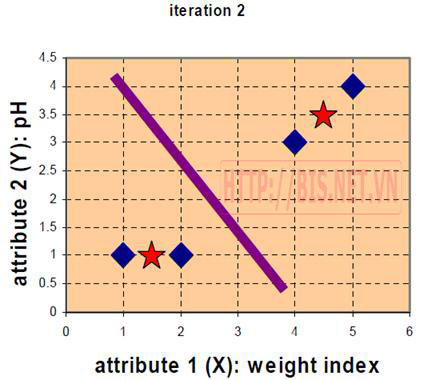


**Bước 7.** Nhóm các đối tượng vào nhóm

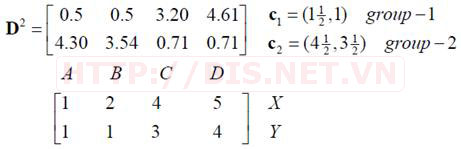


Bước 8. Tính lại tâm cho nhóm mới

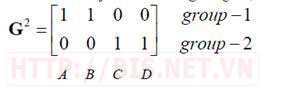




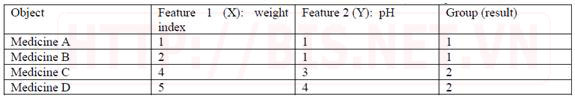
**Bước 8.** Tính lại khoảng cách từ các đối tượng đến tâm mới



**Bước 9.** Nhóm các đối tượng vào nhóm



Ta thấy G2 = G1 (Không có sự thay đổi nhóm nào của các đối tượng) nên thuật toán dừng và kết quả phân nhóm như sau:



Thuật toán K-Means có ưu điểm là đơn giản, dễ hiểu và cài đặt. Tuy nhiên, một số hạn chế của K-Means là hiệu quả của thuật toán phụ thuộc vào việc chọn số nhóm K (phải xác định trước) và chi phí cho thực hiện vòng lặp tính toán khoảng cách lớn khi số cụm K và dữ liệu phân cụm lớn.

* Self Oganizing Map:
* Ý tưởng: Self Organizing Map (SOM) hay Self-Organizing Feature Map (SOFM) là một mạng Neuron nhân tạo (Artificial Neural Networks – ANN), được huấn luyện (trained) sử dụng kỹ thuật [*Unsupervised learning*](http://en.wikipedia.org/wiki/Unsupervised_learning) để biểu diễn dữ liệu với số chiều (dimension) thấp hơn nhiều (thường là 2 chiều) so với dữ liệu đầu vào nhiều chiều (thường số chiều lớn). Kết quả của SOM gọi là bản đồ (Map). SOM là một ANN, tuy nhiên SOM khác với các ANN là không sử dụng các lớp ẩn (hidden layers) chỉ sử dụng input và output layer. SOM sử dụng khái niệm láng giềng (neighborhood) để giữ lại đặc trưng của các dữ liệu đầu vào trên bản đồ (có nghĩa là các training sample tương tự nhau thì được đặt gần nhau trên bản đồ). Ưu điểm chính của SOM là biểu diễn trực quan dữ liệu nhiều chiều vào không gian ít chiều hơn (thường là 2 chiều) và đặc trưng của dữ liệu đầu vào được giữ lại trên bản đồ.
* SOM gồm các thành phần sau:

**Output Layer:** Gồm các node (neurons) được bố trí trên một lưới (bản đồ) kích thước X xY. Mỗi neuron có vị trí xác định trên lưới, tại mỗi neuron lưu giữ một vector trọng số (weight vector) có số chiều bằng với số chiều của input vector.

**Input vector:**  Là các training sample có kích thước n

**Ma trận trọng số (weight matrix)** wijkết nối giữa input vector và các neurons

* ***VD:***

VD: Xét một mạng Kohonen sử dụng thuật toán SOM với kích thước 20x30=600 neural, độ phân giải bức ảnh đầu vào được tính bằng đơn vị megapixel tức là có tới hàng triệu điểm ảnh. Như vậy, riêng trong quá trình huấn luyện, việc tìm BMU đã phải duyệt qua khoảng 600 triệu lần các neural mà chưa tính đến các phép tính khoảng cách, so sánh, cập nhật trọng số…

* ***Ưu điểm:***
* Rất dễ hiểu, cài đặt đơn giản và làm việc rất tốt.
* Biểu diễn trực quan dữ liệu đa chiều vào không gian ít chiều hơn và đặc trưng của dữ liệu được giữ lại trên bản đồ.
* Hiệu quả trong quá trình phân tích đòi hỏi trí thông minh để đưa ra quyết định nhanh chóng. Nó giúp cho người phân tích hiểu vấn đề hơn trên một tập *dữ liệu tương đối lớn*.
* Xác định các cụm dữ liệu giúp cho việc tối ưu phân bổ nguồn lực

=> Thuật toán SOM được ứng dụng rất nhiều trong nhận dạng tiếng nói, điều khiển tự động, hóa-sinh trắc học, phân tích tài chính và xử lý ngôn ngữ tự nhiên…

* ***Nhược điểm:***
* Chi phí cho việc tính toán lớn khi số chiều của dữ liệu tăng lên
* Việc xác định ranh giới giữa các nhóm trên lớp ra Kohonen gặp nhiều khó khăn, trong trường hợp dữ liệu biến thiên liên tục, sự phân chia giữa các nhóm là rất nhỏ.
* Khối lượng tính toán là tương đối lớn, do vậy tốc độ xử lý của giải thuật cũng là một thách thức cần xét tới.

**Câu 4: Khai phá luật kết hợp**

* **Khái niệm : L**à một kĩ thuật quan trọng của khai phá dữ liệu. Mục tiêu nhằm phát hiện mối quan hệ giữa các mục dữ liệu trong cơ sở .
* Khai phá luật kết hợp (KPLKH) là một kỹ thuật quan trọng của KPDL. Mục tiêu nhằm phát hiện mối quan hệ giữa các mục dữ liệu trong CSDL. Mô hình đầu tiên của bài toán KPLKH là mô hình nhị phân (hay còn gọi là mô hình cơ bản) được R. Agrawal, T. Imielinski và A. Swami đề xuất vào năm 1993, xuất phát từ nhu cầu phân tích dữ liệu của cơ sở dữ liệu giao tác, phát hiện các mối quan hệ giữa các tập mục hàng hóa (Itemsets) đã bán được tại các siêu thị. Việc xác định các quan hệ này không phân biệt vai trò khác nhau cũng như không dựa vào các đặc tính dữ liệu vốn có của các mục dữ liệu mà chỉ dựa vào sự xuất hiện cùng lúc của chúng.
* **Các miền ứng dụng:**
* Thương mại
* Y học
* **Mục tiêu:**

Mục tiêu của phương pháp này là phát hiện và đưa ra các mối liên hệ giữa các giá trị dữ liệu trong cơ sở dữ liệu. Mẫu đầu ra của giải thuật KPDL là luật kết hợp tìm được.

* **VD:**
* Bài toán KPLKH có thể phát biểu như sau: Cho cơ sở dữ liệu giao tác DB, ngưỡng độ hỗ trợ tối thiểu minsup và ngưỡng độ tin cậy tối thiểu minconf.
* Yêu cầu: Tìm tất cả các luật kết hợp X→Y trên cơ sở dữ liệu DB sao cho sup(X→Y) > minsup và conf (X→Y) > minconf.
* KPLKH này được gọi là bài toán cơ bản hay bài toán nhị phân, vì ở đây, giá trị của mục dữ liệu trong cơ sở dữ liệu là 0 hoặc 1 (xuất hiện hay không xuất hiện).
* Bài toán KPLKH trong cơ sở dữ liệu chia thành hai bài toán con:
  + Tìm tất cả các tập mục thường xuyên: Một tập mục là thường xuyên được xác định qua tính độ hỗ trợ và thoả mãn độ hỗ trợ cực tiểu.
  + Sinh ra các luật kết hợp từ các tập mục thường xuyên đã tìm được thỏa mãn độ tin cậy tối thiểu cho trước.
* Khi KPLKH trong cơ sở dữ liệu DB thì mọi khó khăn nằm ở bài toán thứ nhất là tìm tập mục thường xuyên.
* Thuật ngữ [Khai phá dữ liệu](https://vietnambiz.vn/khai-pha-du-lieu-data-mining-la-gi-nhung-dac-diem-can-luu-y-20191130175442498.htm) ra đời vào cuối những năm 80 thế kỷ trước. Có nhiều định nghĩa khác nhau về khai phá dữ liệu, nhưng diễn đạt một cách dễ hiểu thì khai phá dữ liệu là quá trình tìm kiếm những thông tin (tri thức) có ích, tiềm ẩn và mang tính dự đoán trong các khối cơ sở dữ liệu lớn.
* **Apriori**

*a) Ý tưởng*

* Tìm tất cả frequent itemsets:
* k-itemset (itemsets gồm k items) được dùng để tìm (k+1)- itemset.
* Đầu tiên tìm 1-itemset (ký hiệu L1). L1 được dùng để tìm L2 (2-itemsets). L2 được dùng để tìm L3 (3-itemset) và tiếp tục cho đến khi không có k-itemset được tìm thấy.
* Từ frequent itemsets sinh ra các luật kết hợp mạnh (các luật kết hợp thỏa mãn 2 tham số min\_sup và min\_conf)
* Cải tiến Apriori : ý tưởng chung
* Giảm số lần duyệt CSDL
* Giảm số lượng tập ứng viên
* Qui trình tính độ phổ biến thuận tiện hơn
* Thuật toán

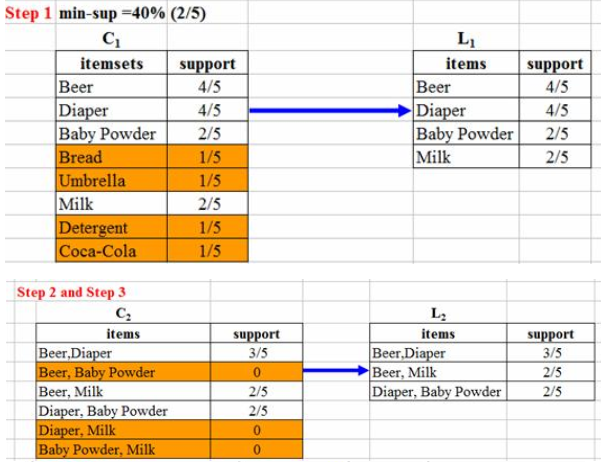
1. Duyệt (Scan)  toàn bộ transaction database để có được support S của 1-itemset, so sánh S với min\_sup, để có được 1-itemset (L1)
2. Sử dụng Lk-1 nối (join) Lk-1  để sinh ra candidate k-itemset. Loại bỏ các itemsets không phải là frequent itemsets thu được k-itemset
3. Scan transaction database để có được support của mỗi candidate k-itemset, so sánh S với min\_sup để thu được frequent k –itemset (Lk)
4. Lặp lại từ bước 2 cho đến khi Candidate set (C) trống (không tìm thấy frequent itemsets)
5. Với mỗi frequent itemset I, sinh tất cả các tập con s không rỗng của I
6. Với mỗi tập con s không rỗng của I, sinh ra các luật  **s => (I-s)** nếu độ tin cậy (Confidence)  của nó > =min\_conf

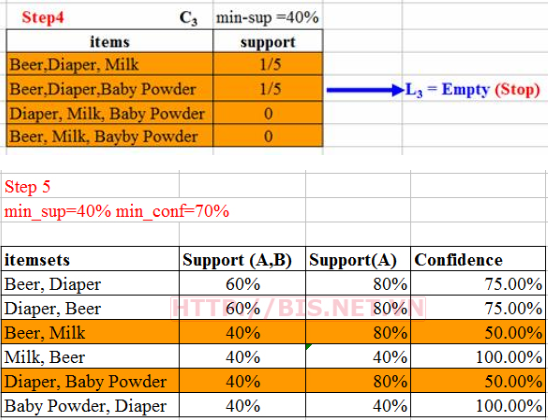
*b) Ví dụ*

**Giả sử có cơ sở dữ liệu bán hàng gồm 5 giao dịch sau:**



**Thuật toán Apriori tìm các luật kết hợp trong giao dịch như sau:**





Kết quả ta có các luật kết hợp sau (với min\_sup= 40%, min\_conf=70%)

R1: Beer → Diaper  (support =60%, confidence = 75%)

R2: Diaper → Beer (support =60%,confidence = 75%)

R3: Milk → Beer (support =40%, confidence = 100%)

R4: Baby Powder → Diaper (support =40%,confidence = 100%)

##### **Thuật toán FP\_growth**

* **Khái miệm:**Thuật toán Apriori có chi phí lớn nhưng lại kém hiệu quả. Để khắc phục nhược điểm này thì FP\_growth ra đời.
* Thuật toán FP-growth được xây dựng với 3 kỹ thuật chính:

(1) Nén dữ liệu thích hợp vào một cấu trúc cây gọi là cây FP-tree. Chỉ có các 1-tập mục (1-item) ở trong cây và các nút của cây được sắp xếp để các nút xuất hiện thường xuyên hơn có thể dễ dàng chia sẻ với các nút xuất hiện ít hơn.

(2) Thực hiện phương pháp khai phá phát triển (growth) từng đoạn dựa trên cây FP-tree gọi là phương pháp FP-growth.

(3) Kỹ thuật tìm kiếm được dùng ở đây là dựa vào sự phân chia, “chia để trị”, phân rã nhiệm vụ khai phá thành các nhiệm vụ nhỏ hơn.

Thuật toán FP-growth do nén toàn bộ CSDL lên một cấu trúc dữ liệu nhỏ hơn là cây FP-tree nên tránh được việc duyệt nhiều lần CSDL (thuật toán chỉ duyệt cơ sở dữ liệu 2 lần). Tiếp theo thuật toán khai phá cây bằng cách phát triển dần các mẫu mà không sinh các tập mục ứng viên, do đó tránh được khối lượng tính toán lớn. Phương pháp FP- growth đã chứng tỏ được tính hiệu quả của nó và có thể thực hiện khai phá cho cả các mẫu ngắn và dài, nhanh hơn thuật toán Apriori, luôn chỉ cần duyệt CSDL 2 lần.

Thuật toán FP- growth thực hiện như sau: Đầu tiên, thuật toán duyệt CSDL lần thứ nhất để tính độ hỗ trợ của từng mục (đếm số lần xuất hiện của từng mục).

Tiếp đến, những mục không đủ độ hỗ trợ bị loại. Các mục còn lại được sắp theo thứ tự giảm dần của độ hỗ trợ (cũng tức là giảm dần theo số lần xuất hiện trong CSDL), ta nhận được danh sách L các mục đã sắp.

Duyệt CSDL lần thứ hai, với mỗi giao tác t, loại các mục không đủ độ hỗ trợ, các mục còn lại theo thứ tự giống như xuất hiện trong L (tức là thứ tự giảm dần theo độ hỗ trợ) được cất vào cây FP-tree.

Phần tiếp theo thuật toán khai phá tìm các mẫu thường xuyên trên cây FP-tree đã xây dựng mà không cần duyệt lại CSDL nữa.

Để hiểu phương pháp này làm việc thế nào, ta xét khai phá CSDL giao tác DB sau với độ hỗ trợ tối thiểu minsup = 3/5.

Thuật toán kinh điển Apriori tìm tập mục thường xuyên theo cách sinh ra các ứng cử viên và duyệt CSDL để kiểm tra, thuật toán FP-growh không khai phá theo cách của thuật toán Apriori mà nén các giao tác của CSDL lên cấu trúc cây FP-Tree, sau đó thực hiện khai phá trên cây này. Thuật toán sinh luật từ tập mục thường xuyên cũng đã được trình bày cụ thể.

##### **Thuật toán FP\_growth**

* **Khái miệm:**Thuật toán Apriori có chi phí lớn nhưng lại kém hiệu quả. Để khắc phục nhược điểm này thì FP\_growth ra đời.
* Thuật toán FP-growth được xây dựng với 3 kỹ thuật chính:

(1) Nén dữ liệu thích hợp vào một cấu trúc cây gọi là cây FP-tree. Chỉ có các 1-tập mục (1-item) ở trong cây và các nút của cây được sắp xếp để các nút xuất hiện thường xuyên hơn có thể dễ dàng chia sẻ với các nút xuất hiện ít hơn.

(2) Thực hiện phương pháp khai phá phát triển (growth) từng đoạn dựa trên cây FP-tree gọi là phương pháp FP-growth.

(3) Kỹ thuật tìm kiếm được dùng ở đây là dựa vào sự phân chia, “chia để trị”, phân rã nhiệm vụ khai phá thành các nhiệm vụ nhỏ hơn.

Thuật toán FP-growth do nén toàn bộ CSDL lên một cấu trúc dữ liệu nhỏ hơn là cây FP-tree nên tránh được việc duyệt nhiều lần CSDL (thuật toán chỉ duyệt cơ sở dữ liệu 2 lần). Tiếp theo thuật toán khai phá cây bằng cách phát triển dần các mẫu mà không sinh các tập mục ứng viên, do đó tránh được khối lượng tính toán lớn. Phương pháp FP- growth đã chứng tỏ được tính hiệu quả của nó và có thể thực hiện khai phá cho cả các mẫu ngắn và dài, nhanh hơn thuật toán Apriori, luôn chỉ cần duyệt CSDL 2 lần.

Thuật toán FP- growth thực hiện như sau: Đầu tiên, thuật toán duyệt CSDL lần thứ nhất để tính độ hỗ trợ của từng mục (đếm số lần xuất hiện của từng mục).

Tiếp đến, những mục không đủ độ hỗ trợ bị loại. Các mục còn lại được sắp theo thứ tự giảm dần của độ hỗ trợ (cũng tức là giảm dần theo số lần xuất hiện trong CSDL), ta nhận được danh sách L các mục đã sắp.

Duyệt CSDL lần thứ hai, với mỗi giao tác t, loại các mục không đủ độ hỗ trợ, các mục còn lại theo thứ tự giống như xuất hiện trong L (tức là thứ tự giảm dần theo độ hỗ trợ) được cất vào cây FP-tree.

Phần tiếp theo thuật toán khai phá tìm các mẫu thường xuyên trên cây FP-tree đã xây dựng mà không cần duyệt lại CSDL nữa.

Để hiểu phương pháp này làm việc thế nào, ta xét khai phá CSDL giao tác DB sau với độ hỗ trợ tối thiểu minsup = 3/5.

Thuật toán kinh điển Apriori tìm tập mục thường xuyên theo cách sinh ra các ứng cử viên và duyệt CSDL để kiểm tra, thuật toán FP-growh không khai phá theo cách của thuật toán Apriori mà nén các giao tác của CSDL lên cấu trúc cây FP-Tree, sau đó thực hiện khai phá trên cây này. Thuật toán sinh luật từ tập mục thường xuyên cũng đã được trình bày cụ thể.

* **Đặc điểm:**
  + Không t ạo t ập itemsets d ự tuy ể n
* Không ki ểm tra xem li ệu itemsets d ự tuy ển có th ực là frequent itemsets S ử d ụng c ấu trúc d ữ li ệu nén d ữ li ệu t ừ t ập d ữ li ệ u
  + Gi ảm chi phí ki ểm tra t ập d ữ li ệ u
  + Chi phí ch ủ y ếu là đếm và xây d ựng cây FP-tree lúc đầ u
    - Hi ệu qu ả và co giãn t ốt cho vi ệc khám phá các frequent itemsets dài l ẫn ng ắ n
* **VD:**

Cho bảng dữ liệu bao gồm các giao dịch (tid) sau:

| **TID** | **Items** |
| --- | --- |
| 1 | f, a, c, d, g, i, m, p |
| 2 | a, b, c, f, l, m, o |
| 3 | b, f, h, j, o |
| 4 | b, c, k, s, p |
| 5 | a, f, c, e, l, p, m, n |

* Tìm các tập mục có độ hỗ trợ ≥ 0.6 (tức tần số sup. ≥ 3).

##### **Lời giải**

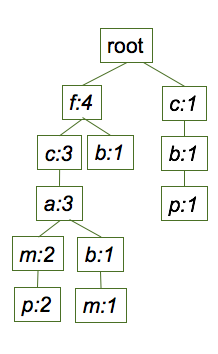
Đầu tiên tìm các item mức 1 có sup. ≥ 3, và sắp xếp theo thứ tự giảm dần:

| **Items** | **sup.** |
| --- | --- |
| F | 4 |
| C | 4 |
| A | 3 |
| B | 3 |
| M | 3 |
| P | 3 |

Tiếp theo sắp xếp các mục phổ biến mức 1 vừa tìm được theo thứ tự giảm dần trong mỗi giao dịch:

| **TID** | **Items** | **Items phổ biến** |
| --- | --- | --- |
| 1 | f, a, c, d, g, i, m, p | f, c, a, m, p |
| 2 | a, b, c, f, l, m, o | f, c, a, b, m |
| 3 | b, f, h, j, o | f, b |
| 4 | b, c, k, s, p | c, b, p |
| 5 | a, f, c, e, l, p, m, n | f, c, a, m, p |

Duyệt các Items phổ biến của mỗi giao dịch để xây dựng FP-Tree:



Hình 1. FP-Tree được xây dựng sau khi duyệt các items phổ biến của các giao dịch.

Tiếp theo, duyệt các item phổ biến mức 1 theo thứ tự tăng dần độ hỗ trợ là p, m, b, a, c, f. Với mỗi item, xây dựng các **cơ sở mẫu điều kiện** (conditional pattern-base) và sau đó là các **FP-Tree điều kiện** (conditional FP-Tree) của nó.

**Tính chất**: bất kì mẫu phổ biến nào có chứa mục Ii đều được chứa trên các nhánh (đường dẫn) của cây FP-Tree chứa Ii, số lần xuất hiện của mẫu chứa các nút trong đường dẫn tiền tố bằng số lần xuất hiện của nút Ii.

Bắt đầu với item p, cơ sở mẫu điều kiện của nó là tất cả các đường dẫn tiền tố của cây FP-Tree khi duyệt từ gốc root = null đến nút p, chính là fcam:2 và cb:1 (số theo sau là số lần xuất hiện của nút p tương ứng với mỗi tiền tố đó).

Tiếp theo ta xây dựng FP-Tree điều kiện từ mẫu này bằng cách trộn tất cả các đường dẫn và giữ lại các nút có tổng các số đếm ≥ sup. = 3:

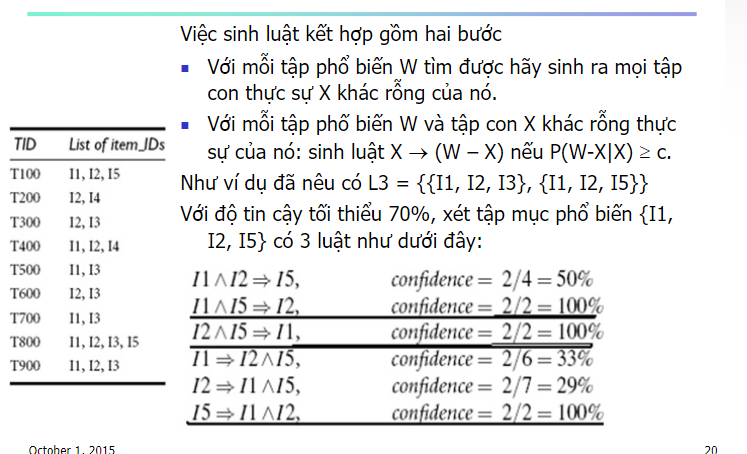
fcam:2 và cb:1 trộn lại thành f:2, c:3, a:2, m:2, b:1; chỉ có c:3 là thoả mãn điều kiện.

Do đó, các mẫu phổ biến chứa p là: p, cp.

Làm tương tự cho các item còn lại, ta sẽ tìm được các mẫu phổ biến chứa các item đó. Cuối cùng có bảng sau đây:

| **Item** | **Cơ sở mẫu điều kiện** | **FP-Tree điều kiện** | **Các mẫu phổ biến** |
| --- | --- | --- | --- |
| p | {fcam:2, cb:1} | {c:3}-p | p, cp |
| m | {fca:2, fcab:1} | {f:3, c:3, a:3}-m | m, fm, cm, am, fcm, cam, fam, fcam |
| b | {fca:1, f:1, c:1} | ∅ | b |
| a | {fc:3} | {f:3, c:3}-a | a, fa, ca, fca |
| c | {f:3} | {f:3}-c | c, fc |
| f | ∅ | ∅ | f |

* **Sinh luật kết hợp:**

****

**V)Các mô hình KPDL và các công cụ hỗ trợ KPDL**

*1.Liệt kê được các mô hình:*

-Mô hình tuyên bố gian lận(Fraud Claiming Models)

-Mô hình nhân bản khách hàng(Customer Clone Models)

-Mô hình đáp ứng(Response Models)

-Mô hình dự đoán doanh thu và lợi nhuận(Revenue and Profit Predictive Models)

\*Giải thích được các mô hình:

-Mô hình tuyên bố gian lận(Fraud Claiming Models):

Gian lận là thách thức mà nhiều ngành công nghiệp và đặc biệt là ngành bảo hiểm phải đối mặt. Các ngành công nghiệp này cần liên tục dự đoán bằng cách sử dụng dữ liệu thô để có thể hiểu và xử lý các khiếu nại gian lận. Chúng tôi có thể theo dõi các khiếu nại đến dưới dạng dữ liệu thô và xác định khả năng nó bị lừa đảo có thể dẫn đến tiết kiệm lớn cho công ty bảo hiểm.

-Mô hình nhân bản khách hàng(Customer Clone Models)

Mô hình nhân bản khách hàng có thể dự đoán khách hàng tiềm năng nào có khả năng đáp ứng cao dựa trên các đặc điểm của khách hàng tốt nhất của tổ chức.

-Mô hình đáp ứng(Response Models)

Các mô hình phản hồi khai thác dữ liệu dự đoán giúp các tổ chức xác định các mô hình sử dụng tách biệt cơ sở khách hàng của họ để tổ chức có thể thiết lập liên hệ với các khách hàng đó. Mô hình phản hồi này là phương pháp tốt nhất để dự đoán và xác định cơ sở khách hàng hoặc khách hàng tiềm năng cho mục tiêu cho một sản phẩm cụ thể mà việc cung cấp phù hợp với việc sử dụng mô hình được phát triển. Những loại mô hình này được áp dụng trong việc xác định các khách hàng có khả năng cao sở hữu đặc điểm của mục tiêu.

-Mô hình dự đoán doanh thu và lợi nhuận(Revenue and Profit Predictive Models)

Các mô hình Dự đoán doanh thu và lợi nhuận kết hợp các đặc điểm phản hồi hoặc không phản hồi với ước tính doanh thu nhất định, đặc biệt nếu kích thước đặt hàng, tỷ suất lợi nhuận khác nhau hoặc hóa đơn hàng tháng. Như chúng ta biết rằng không phải tất cả các phản hồi đều có cùng giá trị hoặc bằng nhau và mô hình có thể tăng các phản hồi không nhất thiết mang lại lợi nhuận cho chúng ta. Kỹ thuật dự đoán doanh thu và lợi nhuận chỉ ra rằng những người trả lời có khả năng cao tăng doanh thu hoặc lợi nhuận biên với phản ứng của họ so với những người trả lời khác. Đây là một số loại mô hình và có nhiều loại khác có thể giúp kết hợp dữ liệu cần thiết từ bộ dữ liệu thô.

các công cụ hỗ trợ KPDL:

#### 1.WEKA:

Đây là một công cụ tùy biến dựa trên JAVA, miễn phí sử dụng. Nó bao gồm trực quan hóa và phân tích dự đoán và kỹ thuật mô hình, phân cụm, liên kết, hồi quy và phân loại.

#### 2.Công cụ lập trình R:

Điều này được viết bằng C và FORTRAN và cho phép các nhà khai thác dữ liệu viết các tập lệnh giống như ngôn ngữ / nền tảng lập trình. Do đó, nó được sử dụng để làm phần mềm thống kê và phân tích để khai thác dữ liệu. Nó hỗ trợ phân tích đồ họa, cả mô hình tuyến tính và phi tuyến, phân loại, phân cụm và phân tích dữ liệu dựa trên thời gian.

#### 3. Python dựa trên Orange và NTLK:

Python rất phổ biến do dễ sử dụng và các tính năng mạnh mẽ của nó. Orange là một công cụ mã nguồn mở được viết bằng Python với các phân tích dữ liệu hữu ích, phân tích văn bản và các tính năng học máy được nhúng trong giao diện lập trình trực quan. NTLK, cũng được sáng tác bằng Python, là một công cụ khai thác dữ liệu xử lý ngôn ngữ mạnh mẽ, bao gồm khai thác dữ liệu, học máy và các tính năng cạo dữ liệu có thể dễ dàng được xây dựng cho các nhu cầu tùy chỉnh.

#### 4.Knime:

Được sử dụng chủ yếu cho tiền xử lý dữ liệu - tức là trích xuất, chuyển đổi và tải dữ liệu, Knime là một công cụ mạnh mẽ với GUI hiển thị mạng của các nút dữ liệu. Phổ biến trong số các nhà phân tích dữ liệu tài chính, nó có lớp lót dữ liệu mô-đun, thúc đẩy học máy và các khái niệm khai thác dữ liệu một cách tự do để xây dựng các báo cáo kinh doanh thông minh.