Shape, square

Description automatically generated

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**

-----🙞🙜🕮🙞🙜-----

**Icon

Description automatically generated with medium confidence**

**Khai Phá Dữ Liệu**

***ĐỀ TÀI:* *Xây dựng kho dữ liệu về máy tính xách tay (laptop).***

***Khai phá kho dữ liệu này cho mục đích tư vấn cho khách hàng***

***chọn phù hợp với nhu cầu***

**Giảng viên hướng dẫn: Nguyễn Ngọc Duy**

**Lớp: D18CQIS01-N**

**Sinh viên thực hiện: Châu Văn Hậu - N18DCCN058**

**Nguyễn Đỗ Yến Chi - N18DCCN022**

**Tạ Thương Thương - N18DCCN226**

TP. Hồ Chí Minh, 03/2022

**MỤC LỤC**

[GIỚI THIỆU ĐỀ TÀI 1](#_Toc99619931)

[**1. Lý do chọn đề tài** 1](#_Toc99619932)

[**2. Mục tiêu nghiên cứu** 1](#_Toc99619933)

[**3. Đối tượng và phạm vi nghiên cứu** 2](#_Toc99619934)

[**4. Phương pháp nghiên cứu** 2](#_Toc99619935)

[**5. Một số hạn chế dự kiến và cách khắc phục** 2](#_Toc99619936)

[**6. Nội dung báo cáo** 2](#_Toc99619937)

[CHƯƠNG I: GIỚI THIỆU CÔNG CỤ 3](#_Toc99619938)

[VÀ ĐỐI TƯỢNG CHÍNH 3](#_Toc99619939)

[**1. Giới thiệu công cụ** 3](#_Toc99619940)

[1.1 Giới thiệu về WEKA 3](#_Toc99619941)

[1.2 Các chức năng chính của WEKA 3](#_Toc99619942)

[1.3 Môi trường Explore 4](#_Toc99619943)

[**2. Giới thiệu dữ liệu** 7](#_Toc99619944)

[2.1 Tổng quan về doanh nghiệp 7](#_Toc99619945)

[2.2 Nghiệp vụ 7](#_Toc99619946)

[2.3 Xác định thuộc tính và mô tả dữ liệu 7](#_Toc99619947)

[CHƯƠNG II: CƠ SỞ LÝ THUYẾT 17](#_Toc99619948)

[**1. Quy trình khai phá dữ liệu** 17](#_Toc99619949)

[**2. Giới thiệu bài toán phân lớp dữ liệu** 18](#_Toc99619950)

[**3. Thuật toán Naive Bayes** 20](#_Toc99619951)

[3.1 Định lý Bayes 20](#_Toc99619952)

[3.2 Phân lớp Naive Bayes 21](#_Toc99619953)

[**4. Thuật toán Random Forest** 23](#_Toc99619954)

[**5.** **Đánh giá các giải thuật phân lớp** 25](#_Toc99619955)

[CHƯƠNG III: THỰC NGHIỆM VÀ ĐÁNH GIÁ 27](#_Toc99619956)

[**1.** **Thu thập và xử lý dữ liệu** 27](#_Toc99619957)

[**2.** **Phân lớp dữ liệu với thuật toán Random Forest** 30](#_Toc99619958)

[2.1 Cấu hình để phân lớp dữ liệu: 30](#_Toc99619959)

[2.2 Kết quả phân lớp dựa trên thuật toán Random Forest 30](#_Toc99619960)

[**3.** **Phân lớp dữ liệu với thuật toán Naive Bayes** 31](#_Toc99619961)

[3.1 Cấu hình để phân lớp dữ liệu: 31](#_Toc99619962)

[Kết quả phân lớp dựa trên thuật toán Naive Bayes 32](#_Toc99619963)

[**4.** **Lựa chọn các thuộc tính dữ liệu dựa trên độ tương quan** 33](#_Toc99619964)

[**5.** **Kết luận** 37](#_Toc99619965)

[CHƯƠNG IV: TỔNG KẾT 38](#_Toc99619966)

[TÀI LIỆU THAM KHẢO 39](#_Toc99619967)

# GIỚI THIỆU ĐỀ TÀI

## **1. Lý do chọn đề tài**

Trong thời đại ngày nay, sử dụng tri thức đã trở thành động lực chủ chốt cho tăng trưởng kinh tế quốc gia, cho tăng cường năng lực cạnh tranh của doanh nghiệp. Đồng thời, dung lượng dữ liệu số tăng rất nhanh chóng, chúng đã trở thành nguồn tài nguyên ẩn thông tin và tri thức có tiềm năng hữu ích cho phát triển kinh tế và tăng cường năng lực cạnh tranh. Nghiên cứu và triển khai các phương pháp tự động phát hiện các mẫu mới, có giá trị, hữu ích tiềm năng và hiểu được trong khối dữ liệu đồ sộ, khắc phục hiện tượng giàu về dữ liệu mà nghèo về thông tin, hướng tới mục tiêu tăng cường tài nguyên tri thức là hết sức cần thiết và có ý nghĩa. Theo đó, việc định nghĩa khai phá dữ liệu được hiểu theo nghĩa đơn giản chính là việc rút trích thông tin hay tri thức mới và có ích từ nguồn dữ liệu khổng lồ.

Trong lĩnh vực thương mại, tính cạnh tranh kinh doanh là rất cao, cho nên việc phân tích dữ liệu để đưa đến người dùng những dịch vụ tốt hơn, có nhiều tiện ích cho khách hàng hơn là điều cần thiết. Cụ thể trong chuyện mua bán laptop, khi đưa vào thông tin của một chiếc máy nào đó, người ta đồng thời phải đưa ra quyết định để biết loại máy đó thuộc nhóm nhu cầu nào. Tuy nhiên, với mỗi thông tin được đưa vào, người đánh giá, xếp loại chúng vào những nhóm nhu cầu phải có kiến thức, phải là chuyên gia am hiểu về các dòng máy, am hiểu về các thông tin chi tiết của máy. Như vậy nếu ta đưa vào rất nhiều thông tin, người này sẽ phải tốn rất nhiều thời gian để xem xét và đưa ra quyết định, như vậy sẽ dẫn đến nguy cơ mất cơ hội trong kinh doanh. Do đó, việc khai phá dữ liệu nhằm giúp đỡ con người trong công việc luôn được thúc đẩy mạnh mẽ, làm sao với các dữ liệu đã thu thập, những hành động tương ứng từ dữ liệu đó có thể mang lại lợi ích kịp thời, tối đa cho doanh nghiệp.

Với những lý do trên cùng với sự cho phép của thầy Nguyễn Ngọc Duy, nhóm chúng em quyết định chọn đề tài ***“Xây dựng kho dữ liệu về máy tính xách tay (laptop). Khai phá kho dữ liệu này cho mục đích tư vấn cho khách hàng chọn phù hợp với nhu cầu.”*** nhằm đưa ra đề xuất laptop phù hợp, mang đến trải nghiệm mua hàng tốt nhất cho khách hàng, nâng cao vị thế cạnh tranh của doanh nghiệp.

## **2. Mục tiêu nghiên cứu**

- Nghiên cứu lý thuyết về khai phá dữ liệu, mô hình thuật toán liên quan đến lĩnh vực khai phá dữ liệu.

- Tìm hiểu và sử dụng công cụ khai phá dữ liệu Weka.

- Ứng dụng của thuật toán phân lớp Random Forest và Naïve Bayes vào việc hỗ trợ ra quyết định trong môi trường mua bán trực tuyến.

- Đưa ra giải pháp nhằm phân loại nhu cầu mua laptop của khách hàng.

## **3. Đối tượng và phạm vi nghiên cứu**

- Đối tượng nghiên cứu: dữ liệu về laptop thuộc trang thương mại điện tử phongvu.com

- Phạm vi nghiên cứu: trang thương mại điện tử phongvu.com

## **4. Phương pháp nghiên cứu**

***Phương pháp thực nghiệm khoa học:*** Dựa vào tập dữ liệu đã thu thập được, tiến hành tiền xử lý và sử dụng các thuật toán phân lớp, đánh giá tính hiệu quả của thuật toán dựa trên tập dữ liệu kiểm thử. Xác định thuật toán phân lớp nào phù hợp với bộ dữ liệu hơn.

**5. Một số hạn chế dự kiến và cách khắc phục**

- Chưa tìm hiểu hết tất cả các tính năng của công cụ hỗ trợ WEKA để giải quyết các bài toàn thực tế.

- Sử dụng Weka chỉ tốt khi dữ liệu được làm sạch và chuẩn bị tốt, điều này không xảy ra trong đời sống thực.

## **6. Nội dung báo cáo**

Báo cáo được trình bày trong phạm vi 3 chương, bao gồm:

*Chương 1: Giới thiệu công cụ và đối tượng chính* trình bày về việc xây dựng bộ dữ liệu cùng công cụ sử dụng đối với dữ liệu đó

*Chương 2: Cơ sở lý thuyết* đưa ra quy trình khai phá dữ liệu, những khái niệm, phương pháp đối với 2 thuật toán phân lớp là Random Forest và Naive Bayes.

*Chương 3: Thực nghiệm và đánh giá* dựa trên việc sử dụng Weka trong việc phân lớp dữ liệu, cấu hình thuật toán và đánh giá kết quả thu được.

Dù trong quá trình làm bài, nhóm chúng em đã cố gắng thu thập, nghiên cứu và tổng hợp song chắc chắn vẫn còn không ít khiếm khuyết. Chúng em móng muốn nhận được những ý kiến đánh giá hữu ích từ thầy để kiến thức về môn học “Khai phá dữ liệu” ngày càng trở nên hoàn thiện.

Chúng em xin chân thành cảm ơn sự giúp đỡ từ phía thầy trong thời gian qua.

# CHƯƠNG I: GIỚI THIỆU CÔNG CỤ

# VÀ ĐỐI TƯỢNG CHÍNH

## **1. Giới thiệu công cụ**

### *1.1 Giới thiệu về WEKA*

WEKA (Waikato Environment for Knowledge Analysis) là một phần mềm khai phá dữ liệu mã nguồn mở được phát triển bởi Đại học Waikato ở New Zealand. WEKA cũng là tên một loài chim chỉ có trên một hòn đảo của New Zealand. WEKA được xây dựng bằng ngôn ngữ Java với mục tiêu xây dựng một công cụ hiện đại phát triển các kỹ thuật học máy và ứng dụng vào các bài toán khai phá dữ liệu trong thực tế.

Năm 1993, Đại học Waikato khởi động dự án nghiên cứu và xây dựng phiên bản đầu tiên của WEKA. Năm 1997, Đại học Waikato quyết định xây dựng lại WEKA từ đầu bằng ngôn ngữ Java và có cài đặt các thuật toán mô hình hóa. Đến năm 2005, WEKA được nhận giải thưởng SIGKDD Data Mining and Knowledge Discovery Service Award.

WEKA được chia sẻ rộng rãi trên website http://www.cs.waikato.ac.nz/~ml/weka/index.html. Hiện tại, phiên bản ổn định mới nhất của Weka là Weka 3.8. Ngoài ra, Đại học Waikato còn cung cấp phiên bản đang phát triển Weka 3.9. Đối với mỗi phiên bản, Weka được cung cấp đầy đủ các phiên bản cho hệ điều hành Windows, Mac OS X, Linux. Lưu ý, máy tính cần phải phiên bản Java cần thiết để để chạy một phiên bản Weka cụ thể. Với bản Weka 3.8 hiện tại, máy tính cần cài đặt phiên bản Java 1.7 trở lên.

WEKA được xây dựng với hơn 600 lớp, tổ chức thành 10 packages, mỗi package thực hiện một nhiệm vụ trong quá trình khai phá dữ liệu. Các lớp, packages này được mô tả một cách chi tiết trong tài liệu hướng dẫn sử dụng của nhà cung cấp. Giao diện đồ họa người sử dụng của WEKA được phát triển theo hướng trực quan và dễ sử dụng.



Hình 1: Biểu tượng phần mềm Weka

### *1.2 Các chức năng chính của WEKA*

WEKA cung cấp 5 môi trường làm việc nhằm hỗ trợ người sử dụng hai chức năng chính là khai phá dữ liệu và thực nghiệm, đánh giá các mô hình học máy. Cụ thể:

* Explorer: Môi trường cho phép tiến hành khai phá dữ liệu với các tính năng tiền xử lý dữ liệu (Preprocess), phân lớp (Classify), phân cụm (Cluster), khai thác luật kết hợp (Associate). Ngoài ra, nó còn cung cấp thêm tính năng hỗ trợ lựa chọn thuộc tính (Select attributes) và mô hình hóa dữ liệu (Visualize).
* Experimenter: Môi trường cho phép thực nghiệm (Setup, Run), so sánh, phân tích (Analyse) các mô hình học máy.
* KnowledgeFlow: Môi trường này hỗ trợ các tính năng cơ bản giống như Explorer nhưng với một giao diện kéo thả để hỗ trợ học tập gia tăng.
* Simple CLI: Cung cấp một giao diện dòng lệnh đơn giản cho phép thực thi trực tiếp các lệnh của WEKA cho các hệ điều hành không cung cấp giao diện dòng lệnh riêng.
* Workbench: Môi trường này là sự kết hợp của 4 môi trường nêu trên, người sử dụng có thể tùy ý chuyển đổi mà không cần phải quay lại cửa sổ “Weka GUI Chooser”

Ảnh có chứa văn bản, ảnh chụp màn hình, con chim

Mô tả được tạo tự động

Hình 2: Giao diện chính của phần mềm Weka

### *Môi trường Explore*

Preprocess: Để chọn và thay đổi (xử lý) dữ liệu làm việc.

Classify: Để huấn luyện và kiểm tra các mô hình học máy (phân loại, hoặc hồi quy/dự đoán

Cluster: Để học các nhóm từ dữ liệu (phân cụm)

Associate: Để khám phá các luật kết hợp từ dữ liệu

Select attributes: Để xác định và lựa chọn các thuộc tính liên quan (quan trọng) nhất của dữ liệu.

Visualize: Để xem (hiển thị) biểu đồ tương tác 2 chiều đối với dữ liệu.

#### 1.3.1 Tiền xử lý dữ liệu

Trong qui trình khai phá dữ liệu, công việc xử lý dữ liệu trước khi đưa vào các mô hình là rất cần thiết, bước này làm cho dữ liệu có được ban đầu qua thu thập dữ liệu (gọi là dữliệu gốc ordinal data) có thể áp dụng được (thích hợp) với các mô hình khai phá dữ liệu cụ thể. Các công việc cụ thể của tiền xử lý dữ liệu bao gồm những công việc như:

* Filtering Attributes: Chọn các thuộc tính phù hợp với mô hình
* Filtering samples: Lọc các mẫu (instances, patterns) dữ liệu cho mô hình
* Clean Data: Làm sạch dữ liệu như xóa bỏ các dữ liệu bất thường (Outlier)
* Transformation: Chuyển đổi dữ liệu cho phù hợp với các mô hình như chuyển đổi dữ liệu từ numeric qua nomial hay ordinal.
* Discretization (rời rạc hóa dữ liệu): Nếu bạn có dữ liệu liên tục nhưng một vài mô hình chỉ áp dụng cho các dữ liệu rời rạc (như luật kết hợp chẳn hạn) thì bạn phải thực hiện việc rời rạc hóa dữ liệu.

Dữ liệu có thể được nhập vào (imported) từ một tập tin có khuôn dạng ARFF, CSV.

Dữ liệu cũng có thể được đọc vào từ một địa chỉ URL hoặc từ một cơ sở dữ liệu thông qua JDBC

Các công cụ tiền xử lý dữ liệu của WEKA được gọi là *filters*:

* Rời rạc hóa (Discretization)
* Chuẩn hóa (Normalization)
* Lấy mẫu (Re-sampling)
* Lựa chọn thuộc tính (Attribute selection)
* Chuyển đổi (Transfornfing) và kết hợp (Combining) các thuộc tính

#### 1.3.2 Các bộ phân lớp (Classier)

a. Định nghĩa

Các bộ phân lớp (Classifiers) của WEKA tương ứng với các mô hình dự đoán các đại lượng kiểu định danh (phân lớp) hoặc các đại lượng kiểu số (hồi quy/ dự đoán)

Các kĩ thuật phân lớp được hỗ trợ bởi WEKA:

* + Naive Bayes classifier and Bayesian networks
  + Decision trees
  + Support vector machines
  + Neural networks

*b. Sử dụng Classifier*

Lựa chọn các tùy chọn bộ phân lớp (classifier)

Lựa chọn các tùy chọn cho việc kiểm tra (test options)

* Use training set: Bộ phân loại học được sẽ được đánh giá trên tập học
* Supplied test set: Sử dụng một tập dữ liệu khác (với tập học) để cho việc đánh giá.
* Cross-validation: Tập dữ liệu sẽ được chia đều thành k tập(folds) có kích thước xấp xỉ nhau, và bộ phân loại học được sẽ được đánh giá bởi phương pháp cross-validation.
* Percentage split: Chỉ định tỷ lệ phân chia tập dữ liệu đối với việc đánh giá.

Classifier output hiển thị các thông tin quan trọng:

* Run information Các tùy chọn đối với mô hình học tên của tập
* Run information. Các tùy chọn đối với mô hình học, tên của tập dữ liệu, số lượng các ví dụ, các thuộc tính, và f.f. thí nghiệm
* Classifier model (full training set). Biểu diễn (dạng text) của bộ phân lớp học được
* Predictions on test data. Thông tin chi tiết về các dự đoán của bộ phân lớp đối với tập kiểm tra
* Summary. Các thống kê về mức độ chính xác của bộ phân lớp, đối với f.f. thí nghiệm đã chọn
* Detailed Accuracy By Class. Thông tin chi tiết về mức độ chính xác của bộ phân lớp đối với mỗi lớp.
* Confusion Matrix. Các thành phần của ma trận này thể hiện số lượng các ví dụ kiểm tra (test instances) được phân lớp đúng và phân lớp sai.

#### 1.3.3 Các bộ phân cụm (Cluster builders)

a. Định nghĩa

Các bộ phân cụm của WEKA tương ứng với các mô hình tìm các nhóm của các ví dụ tương tự đối với một tập dữ liệu.

Các kĩ thuật phân cum được hỗ trợ bởi Weka:

* + Expectation maximization (EM)
  + K-Means

Các bộ phân cụm có thể hiển thị kết quả và so sánh với cụm ( lớp ) thực tế

b. Sử dụng Cluster

Lựa chọn một bộ phân cụm (cluster builder)

Lựa chọn chế độ phân cụm (cluster mode)

* + Use training set. Các cụm học được sẽ được kiểm tra đối với tập học.
  + Supplied test set. Sử dụng một tập dữ liệu khác để kiểm tra các cụm học được.
  + Percentage split. Chỉ định tỷ lệ phân chia tập dữ liệu ban đầu cho việc xây dựng tập kiểm tra.
  + Classes to clusters evaluation. So sánh độ chính xác của các cụm học được đối với các lớp được chỉ định.

Store clusters for visualization: Lưu lại các bộ phân lớp trong bộ nhớ, để có thể hiện thị sau đó.

Ignore attributes: Lựa chọn các thuộc tính sẽ không tham gia vào quá trình học các cụm.

#### 1.3.4 Luật kếp hợp (Associate)

Lựa chọn một mô hình (giải thuật) phát hiện luật kết hợp

Associator output hiển thị các thông tin quan trọng

* + Run information: Các tùy chọn đối với mô hình phát hiện luật kết hợp, tên của tập dữ liệu, số lượng các ví dụ, các thuộc tính
  + Associator model (full training set): Biểu diễn (dạng text) của tập các luật kết hợp phát hiện được

Độ hỗ trợ tối thiểu (minimum support)

Độ tin cậy tối thiểu (minimum confidence)

Kích thước của các tập mục thường xuyên (large/frequent itemsets)

Liệt kê các luật kết hợp tìm được.

## **2. Giới thiệu dữ liệu**

### *2.1 Tổng quan về doanh nghiệp*

Tên doanh nghiệp: Phong Vũ

Địa chỉ: 1A Lê Văn Việt, Phường Hiệp Phú, TP. Thủ Đức, TP.HCM

Điện thoại: 02873046867

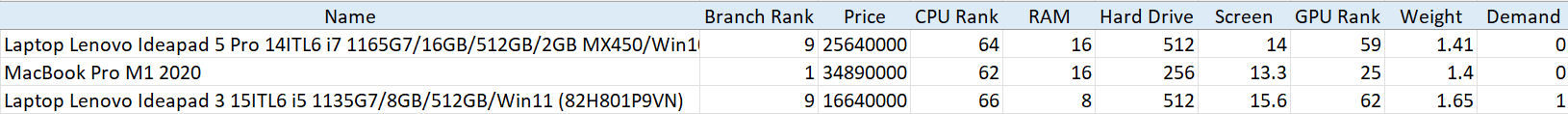
Phong Vũ là chuỗi cửa hàng lớn ở Việt Nam buôn bán đa dạng các mặt hàng và phong phú về mẫu mã trong lĩnh vực Công nghệ thông tin

### *2.2 Nghiệp vụ*

Nhập hàng: Khi nhân viên nhập hàng laptop từ nhà phân phối nếu có một sản phẩm mới chưa tồn tại trong kho thì nhân viên sẽ tiến hành nhập lưu thông tin sản phẩm: tên sản phẩm, hãng sản xuất và các thông số khác… Dựa vào các thông số đó thì chuyên gia sẽ phân tích và nhận định chiếc laptop đó chuyên phục vụ nhu cầu nào của khách hàng.

Tư vấn mua hàng: Khách hàng đến cửa hàng muốn được tư vấn mua laptop, dựa vào nhu cầu của khách hàng mà và sự phân tích của chuyên gia mà cửa hàng sẽ giới thiệu những sản phẩm theo nhu cầu đó.

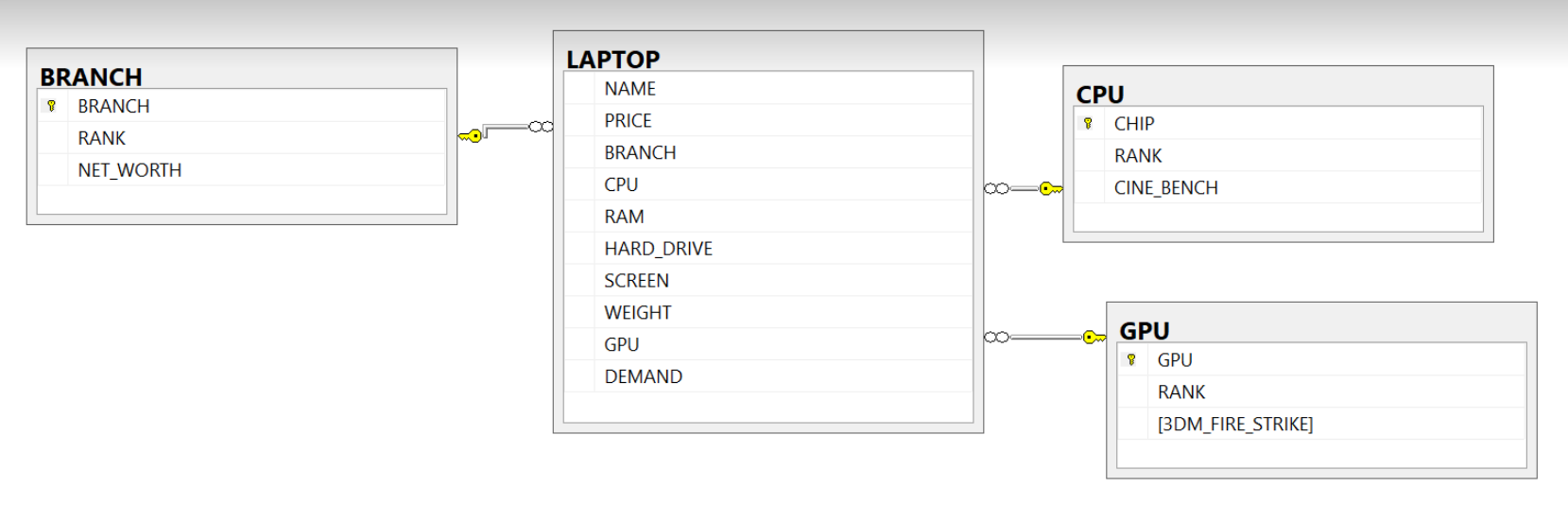
### *2.3 Xác định thuộc tính và mô tả dữ liệu*



Hình 3: Mẫu dữ liệu

Branch Rank: thứ tự xếp hạng của thương hiệu sản phẩm. Thuộc tính này được quy đổi từ tên thương hiệu dựa vào giá trị tài sản ròng (Net worth) (Bảng 1)

Dữ liệu được lưu trữ trong có sở dữ liệu có mô hình như sau:



Bảng 1: Bảng quy đổi tên thương hiệu sang thứ hạng theo chiều giảm Gía trị tài sản ròng

|  |  |  |
| --- | --- | --- |
| **Thương hiệu** | **Hạng** | **Net worth (tỷ USD)** |
| Apple | 1 | 3000 |
| Microsoft | 2 | 2000 |
| Sony | 3 | 188.91 |
| Dell | 4 | 115.86 |
| Huawei | 5 | 108 |
| Samsung | 6 | 107 |
| Hp | 7 | 70.45 |
| Toshiba | 8 | 52.55 |
| Lenovo | 9 | 34.37 |
| LG | 10 | 30.67 |
| Msi | 11 | 24.9 |
| Xiaomi | 12 | 20 |
| Asus | 13 | 6.36 |
| Acer | 14 | 4.26 |
| Gigabyte | 15 | 2 |
| CyerpowerPC | 16 | 1 |
| Razer | 17 | 1 |
| i-life Degital | 18 | 0.1 |
| iBall | 19 | 0.1 |
| Nuvision | 20 | 0.055 |
| Jumper | 21 | 0.05 |
| Chuwi | 22 | 0.01 |

Price: mức giá được bán ra khi sản phẩm ở thời điểm thu thập dữ liệu (3/2022), tính theo VND.

CPU Rank: thứ tự xếp hạng CPU cấu thành nên sản phẩm. Được quy đổi từ tên của CPU dựa vào điểm đánh giá hiệu suất đa lõi từ phần mềm Cinebench R20 (Bảng 2)

Bảng 2: Bảng quy đổi tên CPU sang thứ hạng theo chiều giảm hiệu suất Cinebench R20

|  |  |  |
| --- | --- | --- |
| **Chip (CPU)** | **Xếp hạng** | **Cinebench 20 (M)** |
| Intel Core i7-12700H | 1 | 7158 |
| Intel Core i9-12900H | 2 | 6371 |
| Intel Core i7-10700K | 3 | 5292 |
| AMD Ryzen 9 5900HS | 4 | 5132 |
| AMD Ryzen 7 6800H | 5 | 5074 |
| AMD Ryzen 9 5900HX | 6 | 5019 |
| AMD Ryzen 7 5800H | 7 | 4881 |
| Intel Core i7-11800H | 8 | 4856 |
| Intel Core i9-11900H | 9 | 4757 |
| Intel Core i7-11850H | 10 | 4743 |
| AMD Ryzen 7 5800HS | 11 | 4723 |
| Intel Core i9-11980HK | 12 | 4600 |
| Intel Core i9-11950H | 13 | 4432 |
| AMD Ryzen 7 4800H | 14 | 4420 |
| AMD Ryzen 9 4900H | 15 | 4332 |
| AMD Ryzen 9 4900HS | 16 | 4288 |
| Intel Core i9-9980HK | 17 | 4150 |
| Intel Core i7-11600H | 18 | 4073 |
| Intel Xeon W-10885M | 19 | 4065 |
| AMD Ryzen 7 5800U | 20 | 3906 |
| AMD Ryzen 7 4800HS | 21 | 3869 |
| Intel Core i5-11260H | 22 | 3800 |
| Intel Core i7-10875H | 23 | 3792 |
| Intel Core i9-10980HK | 24 | 3773 |
| AMD Ryzen 7 5850U | 25 | 3671 |
| AMD Ryzen 5 5600H | 26 | 3657 |
| Intel Core i7-10870H | 27 | 3588 |
| Intel Core i5-11500H | 28 | 3578 |
| AMD Ryzen 7 5700U | 29 | 3446 |
| Intel Core i9-10885H | 30 | 3400 |
| Intel Core i9-9880H | 31 | 3391 |
| Intel Core i5-10500H | 32 | 3390 |
| Intel Core i5-11400H | 33 | 3334 |
| AMD Ryzen 5 4600H | 34 | 3333 |
| AMD Ryzen 7 4800U | 35 | 3259 |
| Apple M1 Max | 36 | 3197 |
| AMD Ryzen 7 PRO 4750U | 37 | 3014 |
| AMD Ryzen 5 5600U | 38 | 3000 |
| Intel Core i9-8950HK | 39 | 2924 |
| Intel Core i7-10750H | 40 | 2880 |
| AMD Ryzen 5 5500U | 41 | 2726 |
| Intel Core i7-8850H | 42 | 2724 |
| Apple M1 Pro (10-core) | 43 | 2721 |
| Intel Xeon E-2276M | 44 | 2700 |
| AMD Ryzen 7 4700U | 45 | 2680 |
| Intel Core i7-11370H | 46 | 2645 |
| AMD Ryzen 5 PRO 4650U | 47 | 2623 |
| Intel Core i7-9750H | 48 | 2565 |
| Intel Core i7-11375H | 49 | 2552 |
| AMD Ryzen 5 4600U | 50 | 2544 |
| Apple M1 Pro (8-core) | 51 | 2502 |
| Intel Core i5-11300H | 52 | 2395 |
| Intel Core i7-8750H | 53 | 2335 |
| Intel Core i7-9850H | 54 | 2334 |
| AMD Ryzen 5 4500U | 55 | 2241 |
| Intel Core i7-11390H | 56 | 2193 |
| Intel Core i7-10810U | 57 | 2144 |
| Intel Core i7-1195G7 | 58 | 2142 |
| Intel Core i7-1185G7 | 59 | 2139 |
| Intel Core i7-10710U | 60 | 2128 |
| Intel Core i5-9300HF | 61 | 2111 |
| Apple M1 | 62 | 2052 |
| AMD Ryzen 3 5300U | 63 | 2039 |
| Intel Core i7-1165G7 | 64 | 2033 |
| Intel Core i5-1155G7 | 65 | 1980 |
| Intel Core i5-1135G7 | 66 | 1970 |
| Intel Core i5-10300H | 67 | 1960 |
| Intel Core i5-1145G7 | 68 | 1959 |
| Intel Core i5-10400H | 69 | 1926 |
| Intel Core i5-9300H | 70 | 1900 |
| Intel Core i5-11320H | 71 | 1868 |
| Intel Core i7-7820HK | 72 | 1805 |
| Intel Core i7-7700HQ | 73 | 1772 |
| Intel Core i7-8557U | 74 | 1762 |
| Intel Core i7-1068G7 | 75 | 1739 |
| Intel Core i7-8705G | 76 | 1733 |
| AMD Ryzen 5 3550H | 77 | 1728 |
| AMD Ryzen 7 3750H | 78 | 1725 |
| Intel Core i7-1160G7 | 79 | 1717 |
| Intel Core i5-8300H | 80 | 1689 |
| Intel Core i5-8259U | 81 | 1685 |
| Intel Core i5-12450H | 82 | 1639 |
| AMD Ryzen 3 4300U | 83 | 1599 |
| Intel Core i5-1135G4 | 84 | 1566 |
| AMD Ryzen 7 3780U | 85 | 1580 |
| Intel Core i5-8350U | 86 | 1555 |
| AMD Ryzen 7 3700U | 87 | 1551 |
| Intel Core i7-1065G7 | 88 | 1532 |
| AMD Ryzen 5 3580U | 89 | 1529 |
| Intel Core i7-1180G7 | 90 | 1527 |
| Intel Core i7-1060G7 | 91 | 1521 |
| Intel Core i7-10610U | 92 | 1521 |
| AMD Ryzen 7 PRO 3700U | 93 | 1515 |
| Intel Core i5-8257U | 94 | 1494 |
| AMD Ryzen 5 PRO 3500U | 95 | 1483 |
| Intel Core i5-1035G7 | 96 | 1467 |
| Intel Core i7-10510U | 97 | 1442 |
| Intel Core i5-8365U | 98 | 1430 |
| Intel Core i5-10310U | 99 | 1429 |
| AMD Ryzen 5 3500U | 100 | 1427 |
| Intel Core i5-10210U | 101 | 1414 |
| Intel Core i7-8665U | 102 | 1413 |
| Intel Core i7-8565U | 103 | 1381 |
| Intel Core i5-8265U | 104 | 1373 |
| AMD Ryzen 5 3450U | 105 | 1365 |
| AMD Ryzen 5 PRO 2500U | 106 | 1352 |
| Intel Core i5-1035G1 | 107 | 1329 |
| Intel Core i5-1035G4 | 108 | 1307 |
| Intel Core i7-8650U | 109 | 1289 |
| Intel Core i3-1115G4 | 110 | 1185 |
| Intel Core i7-8550U | 111 | 1130 |
| Intel Core i5-8250U | 112 | 1085 |
| AMD Ryzen 3 PRO 2300U | 113 | 1008 |
| AMD Ryzen 3 2300U | 114 | 1008 |
| AMD Ryzen 7 PRO 2700U | 115 | 935 |
| AMD Ryzen 3 3300U | 116 | 921 |
| Intel Core i3-10110U | 117 | 862 |
| Intel Core i3-1005G1 | 118 | 822 |
| Intel Core i5-7300U | 119 | 800 |
| Intel Core i3-8145U | 120 | 798 |
| Intel Core i7-7560U | 121 | 738 |
| Intel Pentium Silver N6000 | 122 | 736 |
| Intel Core i7-7500U | 123 | 735 |
| AMD Ryzen 3 3250U | 124 | 735 |
| Intel Core i5-6200U | 125 | 732 |
| AMD Ryzen 3 3200U | 126 | 728 |
| Intel Core i7-7600U | 127 | 686 |
| Intel Core i5-7200U | 128 | 645 |
| Intel Core i5-8200Y | 129 | 641 |
| Intel Core i3-1000NG4 | 130 | 587 |
| Intel Core i5-L16G7 | 131 | 576 |
| Intel Celeron N4100 | 132 | 560 |
| AMD Ryzen 3 2200U | 133 | 544 |
| AMD Athlon Gold 3150U | 134 | 524 |
| Intel Pentium Silver N5030 | 135 | 516 |
| Intel Pentium Gold 5405U | 136 | 513 |
| Intel Core i7-8500Y | 137 | 512 |
| Intel Core i7-7Y75 | 138 | 491 |
| Intel Pentium Silver N5000 | 139 | 478 |
| Intel Pentium 4417U | 140 | 472 |
| Intel Celeron N5100 | 141 | 451 |
| Intel Core i3-7100U | 142 | 445 |
| Intel Core m3-8100Y | 143 | 413 |
| Intel Celeron N4500 | 144 | 413 |
| AMD 3020e | 145 | 370 |
| Intel Pentium N4200 | 146 | 366 |
| Intel Celeron N4120 | 147 | 364 |
| AMD Ryzen 5 2500U | 148 | 350 |
| Intel Celeron N3450 | 149 | 346 |
| Intel Celeron N4020 | 150 | 311 |
| AMD A9-9425 | 151 | 284 |
| AMD A6-9225 | 152 | 273 |
| Intel Celeron N3350 | 153 | 164 |
| Intel Celeron 3867U | 154 | 159 |
| AMD A6-9220e | 155 | 110 |
| AMD A4-9120e | 156 | 77 |

RAM: độ lớn bộ nhớ tạm của máy, đơn vị tính Gigabyte.

Hard Drive: độ lớn ổ cứng lưu trữ của laptop, đơn vị tính Gigabyte.

Screen: độ lớn của màn hình laptop, là độ dài đường chéo màn hình của máy tính theo đơn vị inch (2,54 cm = 1 inch).

GPU Rank: thứ tự xếp hạng của card đồ hoạ của chiếc laptop. Được quy đổi từ tên của card đồ hoạ của máy dựa vào điểm đánh giá đồ hoạ 3D từ phần mềm 3DMark (Bảng 3)

Bảng 3: Bảng quy đổi GPU của laptop sang thứ hạng theo chiều giảm điểm đánh giá 3DMark

|  |  |  |
| --- | --- | --- |
| **GPU** | **Xếp hạng** | **3DM Fire Strike** |
| 2x NVIDIA GeForce GTX 1070 (8GB GDDR5, SLI) | 1 | 33487 |
| NVIDIA GeForce RTX 3080 | 2 | 31897 |
| AMD Radeon RX 6800M | 3 | 31044 |
| NVIDIA GeForce RTX 3070 Ti Max-Q | 4 | 27133 |
| Card tích hợp 24 nhân GPU | 5 | 27100 |
| AMD Radeon RX 6700M | 6 | 27000 |
| NVIDIA GeForce RTX 3070 | 7 | 26996 |
| NVIDIA GeForce RTX 2080 SUPER | 8 | 25628 |
| NVIDIA GeForce RTX 2080 | 9 | 24480 |
| Card tích hợp 16 nhân GPU | 10 | 24000 |
| AMD Radeon RX 6600M | 11 | 23403 |
| NVIDIA GeForce RTX 3060 | 12 | 22643 |
| NVIDIA GeForce RTX 2080 SUPER Max-Q | 13 | 22105 |
| NVIDIA GeForce GTX 1080 | 14 | 21304 |
| NVIDIA RTX A3000 | 15 | 19951 |
| NVIDIA GeForce RTX 2070 | 16 | 18946 |
| NVIDIA GeForce RTX 2080 Max-Q | 17 | 18372 |
| NVIDIA GeForce RTX 2070 SUPER Max-Q | 18 | 18275 |
| NVIDIA GeForce GTX 1080 Max-Q | 19 | 18121 |
| NVIDIA GeForce RTX 2060 | 20 | 17465 |
| NVIDIA GeForce RTX 2070 Max-Q | 21 | 16775 |
| NVIDIA GeForce GTX 1070 | 22 | 16746 |
| NVIDIA GeForce RTX 3050 Ti | 23 | 15319 |
| NVIDIA GeForce RTX 2060 Max-Q | 24 | 15074 |
| Card tích hợp 8 nhân GPU | 25 | 15000 |
| NVIDIA RTX A2000 | 26 | 14932 |
| NVIDIA Quadro RTX 3000 | 27 | 14667 |
| NVIDIA GeForce GTX 1660 Ti | 28 | 14624 |
| AMD Radeon RX 5600M | 29 | 14609 |
| NVIDIA GeForce GTX 1070 Max-Q | 30 | 14120 |
| NVIDIA GeForce RTX 3050 | 31 | 13345 |
| NVIDIA GeForce GTX 1660 Ti Max-Q | 32 | 12760 |
| NVIDIA Quadro P4000 | 33 | 12284 |
| AMD Radeon RX 5500M | 34 | 12276 |
| NVIDIA GeForce GTX 1060 | 35 | 11650 |
| NVIDIA RTX A2000 | 36 | 11052 |
| AMD Radeon RX 5300M | 37 | 10789 |
| NVIDIA GeForce GTX 1060 Max-Q | 38 | 10569 |
| NVIDIA GeForce GTX 1650 Ti | 39 | 9965 |
| NVIDIA GeForce GTX 1650 Ti Max-Q | 40 | 9739 |
| NVIDIA GeForce GTX 980M | 41 | 9672 |
| Card tích hợp 7 nhân GPU | 42 | 9576 |
| NVIDIA Quadro P3000 | 42 | 9463 |
| NVIDIA GeForce GTX 1650 | 43 | 9327 |
| NVIDIA Quadro M5500 | 44 | 9172 |
| NVIDIA Quadro T2000 Max-Q | 45 | 8642 |
| NVIDIA Quadro T2000 | 46 | 8573 |
| NVIDIA GeForce GTX 1650 Max-Q | 47 | 7866 |
| NVIDIA T600 | 48 | 7841 |
| NVIDIA Quadro T1000 Max-Q | 49 | 7821 |
| NVIDIA GeForce GTX 1050 Ti | 50 | 7686 |
| NVIDIA GeForce GTX 970M | 51 | 7507 |
| NVIDIA GeForce GTX 1050 Ti Max-Q | 52 | 7383 |
| AMD Radeon RX Vega M GL | 53 | 7159 |
| Intel Iris Xe Graphics G7 | 54 | 6837 |
| NVIDIA GeForce GTX 1050 | 55 | 6532 |
| AMD Radeon RX 560X | 56 | 6414 |
| NVIDIA Quadro M2200 | 57 | 5871 |
| NVIDIA GeForce GTX 965M | 58 | 5702 |
| NVIDIA GeForce MX450 | 59 | 5346 |
| NVIDIA Quadro T500 | 60 | 5053 |
| NVIDIA Quadro P620 | 61 | 4551 |
| Intel Iris Xe | 62 | 4485 |
| NVIDIA GeForce MX350 | 63 | 4434 |
| NVIDIA GeForce GTX 960M | 64 | 4378 |
| NVIDIA Quadro M1200 | 65 | 4218 |
| NVIDIA GeForce GTX 860M | 66 | 3890 |
| NVIDIA GeForce MX350 | 67 | 3754 |
| NVIDIA GeForce MX330 | 68 | 3728 |
| NVIDIA GeForce MX250 | 69 | 3674 |
| AMD Radeon RX Vega 8 | 70 | 3662 |
| NVIDIA GeForce MX150 | 71 | 3574 |
| NVIDIA GeForce MX330 | 72 | 3556 |
| NVIDIA GeForce GTX 950M | 73 | 3553 |
| AMD Radeon RX Vega 7 | 74 | 3536 |
| NVIDIA Quadro M1000M | 75 | 3506 |
| NVIDIA Quadro P520 | 76 | 3400 |
| NVIDIA Quadro P520 | 77 | 3251 |
| NVIDIA Quadro M620 | 78 | 3222 |
| AMD Radeon Pro WX 3200 | 79 | 3131 |
| NVIDIA GeForce GTX 850M | 80 | 3116 |
| AMD Radeon RX 640 | 81 | 2962 |
| AMD Radeon Vega 9 | 82 | 2763 |
| AMD Radeon Pro WX 3100 | 83 | 2565 |
| NVIDIA GeForce MX230 | 84 | 2563 |
| Intel UHD Graphics Xe 750 | 85 | 2500 |
| AMD Radeon RX Vega 6 | 86 | 2448 |
| AMD Radeon RX Vega 8 | 87 | 2364 |
| NVIDIA Quadro M520 | 88 | 2364 |
| NVIDIA GeForce MX130 | 89 | 2340 |
| AMD Radeon RX Vega 5 | 90 | 2276 |
| AMD Radeon RX Vega 10 | 91 | 2248 |
| NVIDIA GeForce 940MX | 92 | 2230 |
| Intel Iris Plus | 93 | 2154 |
| AMD FirePro W5130M | 94 | 2116 |
| AMD Radeon 530 | 95 | 2065 |
| NVIDIA GeForce 940M | 96 | 2041 |
| NVIDIA GeForce GT 750M | 97 | 1904 |
| AMD Radeon R9 M275 | 98 | 1893 |
| AMD Radeon 620 | 99 | 1733 |
| NVIDIA GeForce 930MX | 100 | 1685 |
| AMD Radeon R7 | 101 | 1661 |
| NVIDIA GeForce 940MX | 102 | 1637 |
| AMD FirePro W4190M | 103 | 1576 |
| Intel Iris Plus Graphics 640 | 104 | 1546 |
| Intel UHD Graphics | 105 | 1493 |
| AMD Radeon 530 | 106 | 1416 |
| NVIDIA GeForce 840M | 107 | 1412 |
| AMD Radeon 520 | 108 | 1403 |
| Intel Iris Graphics 540 | 109 | 1403 |
| AMD Radeon R7 | 110 | 1328 |
| AMD Radeon RX Vega 3 | 111 | 1243 |
| Intel HD Graphics 630 | 112 | 1203 |
| AMD Radeon R6 | 113 | 1179 |
| Intel UHD Graphics 630 | 114 | 1175 |
| Intel UHD Graphics 620 | 115 | 1114 |
| AMD Radeon R5 (Carrizo-L) | 116 | 1003 |
| Intel HD Graphics 530 | 117 | 997 |
| Intel UHD Graphics | 118 | 955 |
| AMD Radeon R5 M335 | 119 | 919 |
| Intel UHD Graphics 615 | 120 | 864 |
| Intel HD Graphics 620 | 121 | 855 |
| AMD Radeon R7 | 122 | 851 |
| NVIDIA GeForce 820M | 123 | 828 |
| AMD Radeon R5 M335 | 124 | 820 |
| AMD Radeon R5 | 125 | 758 |
| Intel HD Graphics 4400 | 126 | 685 |
| Intel HD Graphics 5500 | 127 | 631 |
| Intel HD Graphics 610 | 128 | 616 |
| Intel HD Graphics 520 | 129 | 603 |
| Intel HD Graphics 615 | 130 | 594 |
| Intel HD Graphics 510 | 131 | 581 |
| Intel HD Graphics 4600 | 132 | 565 |
| AMD Radeon R4 | 133 | 551 |
| Intel HD Graphics 515 | 134 | 508 |
| Intel UHD Graphics | 135 | 462 |
| Intel HD Graphics 400 | 136 | 450 |
| Intel UHD Graphics 600 | 137 | 407 |
| Intel HD Graphics 5300 | 138 | 382 |
| Intel HD Graphics | 139 | 358 |
| Intel HD Graphics 505 | 140 | 355 |
| Intel HD Graphics 405 | 141 | 326 |
| AMD Radeon R3 | 142 | 279 |
| AMD Radeon R2 | 143 | 270 |
| Intel HD Graphics 500 | 144 | 265 |
| Intel HD Graphics 5000 | 145 | 206 |
| Intel HD Graphics | 146 | 150 |

*Weight*: cân nặng của laptop tính theo đơn vị kilogram.

*Demand*: nhãn tượng trưng cho nhu cầu của chiếc laptop được chuyển sang bit 0-1(tượng trưng cho 2 nhu cầu: 0 – Kỹ thuật, 1- Văn phòng).

# CHƯƠNG II: CƠ SỞ LÝ THUYẾT

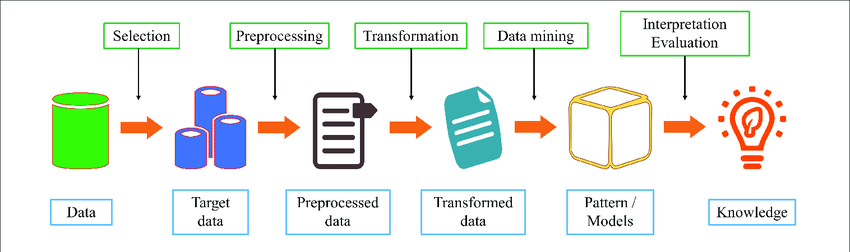
## **1. Quy trình khai phá dữ liệu**

Thông tin là một khái niệm trừu tượng, được thể hiện dưới nhiều dạng thức khác nhau. Thông tin có thể được phát sinh, lưu trữ, biến đổi trong những vật mang tin. Dữ liệu là sự biểu diễn thông tin và được thể hiện bằng các tín hiệu vật lý. Dữ liệu là một dãy các bit các số, sử dụng các bit để đo lường các thông tin và xem nó như là các dữ liệu đã được lọc bỏ các dư thừa, được rút gọn tới mức tối thiểu để đặc trưng một cách cơ bản cho dữ liệu.

Có thể xem tri thức như là các thông tin tích hợp, bao gồm các sự kiện và các mối quan hệ giữa chúng. Các mối quan hệ này có thể được hiểu ra, có thể được phát hiện, hoặc có thể được học.

Mục đích của phát hiện tri thức và KPDL là tìm ra các mẫu và các mô hình đang tồn tại trong các cơ sở dữ liệu nhưng vẫn còn bị che khuất bởi hàng “núi” dữ liệu. Quá trình phát hiện tri thức được mô tả tóm tắt như sau:

Hình 3: Quá trình phát hiện tri thức



Như hình trên, ta thấy quá trình khai phá dữ liệu cần thực hiện các công việc sau:

* Làm sạch dữ liệu (data cleaning): Do dữ liệu thường bị thiếu và không nhất quán sẽ làm chất lượng bộ dữ liệu bị kém đi, nó sẽ gây tác động tiêu cực nghiêm trọng đến quá trình khai thác thông tin. Do đó, xử lý dữ liệu trống hoặc không nhất quán là bước đầu tiên trong khai thác dữ liệu.
* Tích hợp dữ liệu (data integration): Nếu nguồn dữ liệu được nghiên cứu là khác nhau, thì nguồn dữ liệu đó phải được tổng hợp một cách nhất quán.
* Lựa chọn dữ liệu (selection): Lựa chọn chính xác các dữ liệu liên quan dựa trên nội dung nghiên cứu.
* Chuyển đổi dữ liệu (transformation): Chuyển đổi hoặc hợp nhất dữ liệu thành một bộ dữ liệu phù hợp để khai thác và tích hợp các thuộc tính mới hoặc các chức năng hữu ích cho quá trình khai thác dữ liệu.
* Khai phá dữ liệu (data mining): Lựa chọn mô hình phù hợp theo vấn đề và thực hiện các cải tiến mô hình nếu cần thiết.
* Đánh giá (interpretation evalution): Sau khi có được kiến thức từ dữ liệu, cần tiến hành lựa chọn các chỉ số thích hợp để đánh giá mô hình.

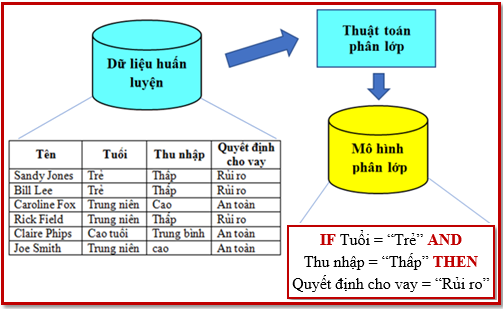
## **2. Giới thiệu bài toán phân lớp dữ liệu**

Phân lớp (Classification/ Categorization) là một trong những mối quan tâm nhiều nhất của con người trong quá trình làm việc với một tập hợp đối tượng. Điều này giúp con người có thể tiến hành việc sắp xếp, tìm kiếm các đối tượng một cách thuận lợi. Khi biểu diễn đối tượng vào các cơ sở dữ liệu, tính chất lớp vốn có của đối tượng trong thực tế thường được biểu diễn tương ứng bằng một thuộc tính “lớp” riêng biệt. Chẳng hạn trên các hệ thống bán hàng trực tuyến, nhiều loại hàng hoá thuộc những nhóm mặt hàng khác nhau như thiết bị điện tử, nhà cửa và đời sống, mẹ và bé hay thiết bị gia dụng… Trước đây, công việc gán các giá trị của thuộc tính lớp thường được làm một cách thủ công. Nhưng hiện nay, với sự bùng nổ của thông tin và các loại dữ liệu, việc gán các giá trị thủ công như vậy là rất khó khăn, chưa kể người gán nhãn phải có độ hiểu biết nhất định với thứ mà họ chuẩn bị gán nhãn, có thể nói công việc thủ công như thế ở thời điểm hiện tại là không thể thực hiện được. Do vậy các phương pháp phân lớp tự động là rất cần thiết và là một trong những chủ đề chính của khai phá dữ liệu.

Các cơ sở dữ liệu thường chứa rất nhiều thông tin ẩn, các thông tin có thể phục vụ cho quá trình phân lớp. Các giải thuật phân lớp thường phân tích dữ liệu nhằm tìm ra các mô hình mô tả các lớp dữ liệu, từ đó có thể quyết định được một phần tử dữ liệu mới là thuộc lớp nào.

Việc tìm ra lớp của một phần tử dữ liệu mới trong nhiều trường hợp có ý nghĩa rất quan trọng, nó hỗ trợ quá trình ra quyết định thông minh thậm chí là những quyết định mang tính sống còn. Ví dụ chúng ta đưa dữ liệu của khách hàng vào bộ phân lớp, bộ phân lớp có nhiệm vụ quyết định xem vị khách hàng đó có khả năng trả nợ ngân hàng khi họ vay nợ ngân hàng hay không, ngân hàng sẽ dựa vào kết quả đó để cân nhắc việc có nên cho khách hàng đó vay tiền ngân hàng hoặc không, sự quyết định đó là “an toàn” hay “mạo hiểm”. Hay một nhân viên tiếp thị trong một công ty buôn bán hàng điện tử thì rất muốn biết một khách hàng có khả năng mua chiếc máy tính đó hay không. Hữu ích hơn, một bác sĩ sẽ dựa trên những triệu chứng của bệnh nhân, máy tính phân tích được và dự đoán bệnh của bệnh nhân đó, bác sĩ sẽ dùng kết quả do máy tính dự đoán được cùng những kinh nghiệm trong thực tế của mình để phán đoán bệnh trạng và xây dựng cho bệnh nhân một phác đồ điều trị phù hợp.

Bản chất của bài toán phân lớp là dự đoán các nhãn (lớp) của phần tử dữ liệu đầu vào và các nhãn này là các giá trị rời rạc. Một số phương pháp điển hình là cây quyết định, luật phân lớp, mạng neuron. Thông thường, giải thuật phân lớp hoạt động thông qua 2 bước. Bước đầu tiên, nó sẽ phân tích dữ liệu đã gán nhãn để tìm ra mô hình phù hợp mô tả tập dữ liệu đó. Bước này được gọi là bước học và tập dữ liệu dùng cho bước này được gọi là tập dữ liệu huấn luyện (training data). Dữ liệu huấn luyện là một tập các phần tử dữ liệu có gán nhãn, hay còn gọi là bản ghi. Một bản ghi thường được biểu diến bởi một vector n chiều X = (x1, x2, … , xn), trong đó mỗi phần tử trong vector chưa một giá trị biểu diễn cho 1 thuộc tính (attribute). Vì nhãn của các phần tử dữ liệu được đi kèm với dữ liệu trong quá trình cho máy học tập, nên đây là công việc học có giám sát (supervised learning). Nói cách khác, các giải thuật phân lớp thuộc lớp giải thuật học có giám sát. Về bản chất, việc học của máy tạo ra một hàm tương ứng mô tả chung cho bộ dữ liệu, như vậy, mỗi phần tử mới được đưa vào sẽ được ánh xạ đến lớp tương ứng của nó. Sau khi học được hàm phân lớp, các giải thuật có thể dùng để dự đoán các dữ liệu mới. Tuy nhiên trước khi đem giải thuật vào ứng dụng trong thực tế, các giải thuật phải trải qua bước thứ 2 là bước kiểm tra hiệu năng từ kết quả cho máy học tập trước đó. Để tránh hiện tượng “quá vừa” (overfit), một tập dữ liệu khác được gọi là dữ liệu kiểm thử (testing data) sẽ được sử dụng để đo độ chính xác của giải thuật. Thông thường dữ liệu kiểm thử sẽ không chứa bất kỳ bản ghi nào giống với dữ liệu huấn luyện. Giống như dữ liệu huấn luyện, bộ dữ liệu kiểm thử cũng có nhãn đi kèm đối với từng bản ghi trong nó. Vậy các nhãn này có tác dụng gì? Nó dùng để so sánh với các nhãn có được từ các giải thuật phân lớp. Tỷ lệ dự đoán đúng nhãn của các giải thuật được gọi là độ chính xác (accuracy) của giải thuật. Khi chất lượng ohaan lớp của các giải thuật là chấp nhận được trong một miền dữ liệu cụ thể nào đó, ta có thể dùng chúng dự đoán lớp của các phần tử dữ liệu mới hoàn toàn chưa biết trước.



Hình 4: Quá trình phân lớp của bài toán vay ngân hàng

Hình trên minh hoạ quá trình phân lớp của bài toán vay tiền ngân hàng, với dữ liệu tương ứng, việc cho vay là “An toàn” hay “Rủi ro”.

## **3. Thuật toán Naive Bayes**

Bộ phân lớp Bayes là một giải thuật thuộc lớp giải thuật phân lớp thống kê, nó có thể dự đoán xác suất của một phần tử dữ liệu thuộc vào một lớp là bao nhiêu. Phân lớp Bayes được dựa trên định lý Bayes (định lý được đặt tên theo tên tác giả của nó là Thomas Bayes).

### *3.1 Định lý Bayes*

Gọi A, B là hai biến cố:

Với P(B) > 0:

Suy ra:

Công thức Bayes:

Công thức Bayes tổng quát:

Với P(A)>0 và {} là một hệ đầy đủ các biến cố:

* + - Tổng số xác suất của hệ bằng 1:
    - Từng đôi một xung khắc:

Khi đó ta có:

Trong đó ta gọi A là một chứng cứ (evidence) (trong bài toán phân lớp A sẽ là một phần tử dữ liệu), B là một giả thiết nào để cho A thuộc về một lớp C nào đó. Trong bài toán phân lớp chúng ta muốn xác định giá trị P(B/A) là xác suất để giả thiết B là đúng với chứng cứ A thuộc vào lớp C với điều kiện ra đã biết các thông tin mô tả A. P(B|A) là một xác suất hậu nghiệm (posterior probability hay posteriori probability) của B với điều kiện A.

Giả sử tập dữ liệu liệu khách hàng của chúng ta được mô tả bởi các thuộc tính tuổi và thu nhập, và một khách hàng X có tuổi là 25 và thu nhập là 2000$. Giả sử H là giả thiết khách hàng đõ sẽ mua máy tính, thì P(H|X) phản ánh xác xuất người dùng X sẽ mua máy tính với điều kiện ta biết tuổi và thu nhập của người đó.

Ngược lại P(H) là xác suất tiền nghiệm (prior probability hay priori probability) của H. Trong ví dụ trên, nó là xác suất một khách hàng sẽ mua máy tính mà không cần biết các thông tin về tuổi hay thu nhập của họ. Hay nói cách khác, xác suất này không phụ thuộc vào yếu tố X. Tương tự, P(X|H) là xác suất của X với điều kiện H (likelihood), nó là một xác suất hậu nghiệm. VÍ dụ, nó là xác suất người dùng X (có tuổi là 25 và thu nhập là $200) sẽ mua máy tính với điều kiện ta đã biết người đó sẽ mua máy tính. Cuối cùng P(X) là xác suất tiền nghiệm của X. Trong ví dụ trên, nó se là xác xuất một người trong tập dữ liệu sẽ có tuổi 25 và thu nhập $2000.

### *3.2 Phân lớp Naive Bayes*

Naive Bayes Classification là một trong những thuật toán được ứng dụng rất nhiều trong các lĩnh vực Machine learning dùng để đưa các dự đoán chính xác nhất dự trên một tập dữ liệu đã được thu thập, vì nó khá dễ hiểu và độ chính xác cao.

Định lý Bayes cho phép chúng ta có thể tính toán một xác suất chưa biết dựa vào các xác suất có điều kiện khác. Với công thức tổng quát tính xác suất của biến cố A với điều kiện biến cố Bk xảy ra trước (hay được gọi là xác suất hậu nghiệm):

Với P(A) > 0 và {𝐵1, 𝐵2 , … , 𝐵𝑛} là một hệ đầy đủ các biến cố thỏa mãn tổng xác suất của hệ bằng 1 () và từng đôi một xung khắc

(𝑃(𝐵𝑖 ∩ 𝐵𝑗 ) = 0). Khi đó ta có:

(1)

Bộ phân lớp Naive bayes hoạt động như sau:

* Gọi D là tập dữ liệu huấn luyện, trong đó mỗi phần tử dữ liệu A chứa n giá trị thuộc tính B1, B2, ..., Bn được biểu diễn bằng một vector n thành phần {x1, x2, ..., xn}
* Giả sử có m lớp C1, C2, ..., Cm. Cho một phần tử dữ liệu A, bộ phân lớp sẽ gán nhãn cho A là lớp có xác suất hậu nghiệm lớn nhất. Cụ thể, bộ phân lớp Bayes sẽ dự đoán A thuộc vào lớp Ci nếu và chỉ nếu:

Giá trị này sẽ tính dựa trên định lý Bayes.

* Để tìm xác suất lớn nhất, ta nhận thấy các giá trị P(A) là giống nhau với mọi lớp nên không cần tính. Do đó ta chỉ cần tìm giá trị lớn nhất của P(A|Ci) \* P(Ci). Chú ý rằng P(Ci) được ước lượng bằng |Di|/|D|, trong đó Di là tập các phần tử dữ liệu thuộc lớp Ci. Nếu xác suất tiền nghiệm P(Ci) cũng không xác định được thì ta coi chúng bằng nhau P(C1) = P(C2) = ... = P(Cm), khi đó ta chỉ cần tìm giá trị lớn nhất.
* Khi số lượng các thuộc tính mô tả dữ liệu là lớn thì chi phí tính toán P(A|Ci) là rất lớn, do đó có thể giảm độ phức tạp của thuật toán Naive Bayes nếu giả thiết các thuộc tính độc lập nhau. Khi đó ta có thể tính:

𝑃(𝐴|𝐶𝑖 ) = 𝑃(𝑥1|𝐶𝑖) … . . 𝑃(𝑥𝑛|𝐶𝑖) (3)

NBC có thể hoạt động với các vector đặc trưng mà một phần là liên tục (sử dụng Gaussian Naive Bayes), phần còn lại ở dạng rời rạc (sử dụng Multinomial hoặc Bernoulli). Trong phần thực nghiệm, tác giả sử dụng MultinomialNB để xây dựng mô hình. Mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó.

Khi đó, tỉ lệ với tần suất từ thứ i (hay thuộc tính thứ i cho trường hợp tổng quát) xuất hiện trong các văn bản của lớp . Cách tính như sau:

(4)

Trong đó:

•là tổng số lần từ thứ I xuất hiện trong các văn bản của lớp , nó được tính bằng tổng của tất cả các thành phần thứ I của các vector thuộc tính ứng với lớp .

• là tổng số từ (kể cả lặp) xuất hiện trong lớp . Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào lớp .

Cách tính này có một hạn chế là nếu có một từ mới chưa bao giờ xuất hiện trong lớp thì biểu thức (1) sẽ bằng 0, điều này dẫn P(A|) = 0 bất kể các giá trị còn lại có lớn thế nào. Việc này sẽ dẫn đến kết quả không chính xác. Để giải quyết việc này, một kỹ thuật được gọi là Laplace smoothing được áp dụng như trong biểu thức (5):

(5)

Với là một số dương, thường bằng 1, để tránh trường hợp tử số bằng 0. Mẫu số được cộng với để đảm bảo tổng xác suất .

## **4. Thuật toán Random Forest**

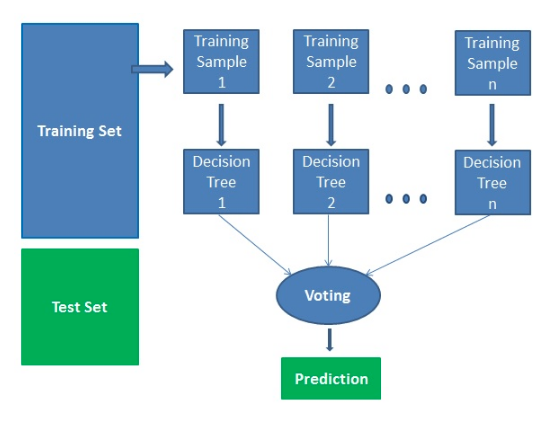
Như tên gọi của nó, Random Forest (RF) dựa trên cơ sở:

1. Random = Tính ngẫu nhiên.
2. Forest = nhiều cây quyết định (decision tree).

Đơn vị của RF là thuật toán cây quyết định, với số lượng hàng trăm. Mỗi cây quyết định được tạo ra một cách ngẫu nhiên từ việc: Tái chọn mẫu (bootstrap, random sampling) và chỉ dùng một phần nhỏ tập biến ngẫu nhiên (random features) từ toàn bộ các biến trong dữ liệu. Ở trạng thái sau cùng, mô hình RF thường hoạt động rất chính xác, nhưng đổi lại, ta không thể nào hiểu được cơ chế hoạt động bên trong mô hình vì cấu trúc quá phức tạp. RF do đó là một trong số những mô hình hộp đen (black box).

Nó hoạt động theo bốn bước:

1. Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
2. Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
3. Hãy bỏ phiếu cho mỗi kết quả dự đoán.
4. Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.



Hình 5: Các bước hoạt động của Random Forest

Ưu điểm: Random forests được coi là một phương pháp chính xác và mạnh mẽ vì số cây quyết định tham gia vào quá trình này. Nó không bị vấn đề overfitting. Lý do chính là nó mất trung bình của tất cả các dự đoán, trong đó hủy bỏ những thành kiến. Thuật toán có thể được sử dụng trong cả hai vấn đề phân loại và hồi quy. Random forests cũng có thể xử lý các giá trị còn thiếu. Có hai cách để xử lý các giá trị này: sử dụng các giá trị trung bình để thay thế các biến liên tục và tính toán mức trung bình gần kề của các giá trị bị thiếu. Bạn có thể nhận được tầm quan trọng của tính năng tương đối, giúp chọn các tính năng đóng góp nhiều nhất cho trình phân loại. Nhược điểm: Random forests chậm tạo dự đoán bởi vì nó có nhiều cây quyết định. Bất cứ khi nào nó đưa ra dự đoán, tất cả các cây trong rừng phải đưa ra dự đoán cho cùng một đầu vào cho trước và sau đó thực hiện bỏ phiếu trên đó. Toàn bộ quá trình này tốn thời gian. Mô hình khó hiểu hơn so với cây quyết định, nơi bạn có thể dễ dàng đưa ra quyết định bằng cách đi theo đường dẫn trong cây.

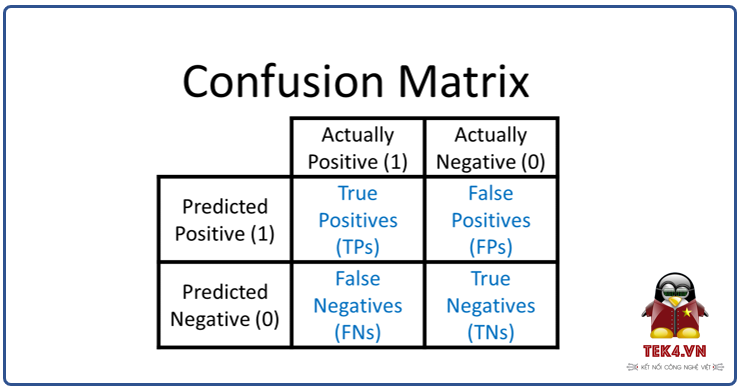
Random forests cũng cung cấp một chỉ số lựa chọn tính năng tốt. Scikit-learn cung cấp thêm một biến với mô hình, cho thấy tầm quan trọng hoặc đóng góp tương đối của từng tính năng trong dự đoán. Nó tự động tính toán điểm liên quan của từng tính năng trong giai đoạn đào tạo. Sau đó, nó cân đối mức độ liên quan xuống sao cho tổng của tất cả các điểm là 1.

Điểm số này sẽ giúp chúng ta chọn các tính năng quan trọng nhất và thả các tính năng quan trọng nhất để xây dựng mô hình.

Random forests sử dụng tầm quan trọng của gini hoặc giảm tạp chất trung bình (MDI) để tính toán tầm quan trọng của từng tính năng. Gini tầm quan trọng còn được gọi là tổng giảm trong tạp chất nút. Đây là mức độ phù hợp hoặc độ chính xác của mô hình giảm khi bạn thả biến. Độ lớn càng lớn thì biến số càng có ý nghĩa. Ở đây, giảm trung bình là một tham số quan trọng cho việc lựa chọn biến. Chỉ số Gini có thể mô tả sức mạnh giải thích tổng thể của các biến. Random Forests và cây quyết định Random Forests là một tập hợp của nhiều cây quyết định. Cây quyết định sâu có thể bị ảnh hưởng quá mức, nhưng Random forests ngăn cản việc lấp đầy bằng cách tạo cây trên các tập con ngẫu nhiên. Cây quyết định nhanh hơn tính toán. Random forests khó giải thích, trong khi cây quyết định có thể diễn giải dễ dàng và có thể chuyển đổi thành quy tắc.

## **Đánh giá các giải thuật phân lớp**

Như đã đề cập ở trên, trước khi đưa bộ phân lớp vào ứng dụng, chúng ta cần phải biết được độ chính xác của nó có đáp ứng được yêu cầu trong miền dữ liệu cụ thể nào đó hay không. Để tính toán các độ đo đánh giá, ta sử dụng ma trận nhầm lẫn (confusion matrix). Đó là một bố cục bảng cụ thể cho phép hình dung hiệu suất của một thuật toán bằng đồ thị trực quan của 2 yếu tố thực tế và dự đoán.



Hình 6: Các yếu tố của Confusion Matrix

Các yếu tố của Confusion Matrix*:*

(Cột dọc là lớp thực tế, cột ngang là lớp được dự đoán dựa vào giải thuật phân lớp)

* TP (True Positive) là số lượng các phần tử được dự đoán đúng lớp 1.
* FP (False Positive) số lượng các phần tử bị đoán nhầm từ lớp 1 sang 0.
* FN (False Negative) số lượng các phần tử bị đoán nhầm lớp 1 sang 0.
* TN (True Negative) là số lượng các phần tử được dự đoán đúng lớp 0.

Các chỉ số đánh giá liên quan đến Confusion Matrix:

* Accuracy

Nó được tính bằng cách chia tổng số dự đoán đúng cho tất cả các dự đoán.

* Recall

Việc recall là biện pháp để kiểm tra các kết quả dự đoán tích cực chính xác trong tổng số các kết quả tích cực.

* Precision

Precision kiểm tra xem có bao nhiêu kết quả thực sự là kết quả tích cực trong tổng số các kết quả được dự đoán tích cực.

* F-measure

Chúng ta thấy rằng cả precision và recall đều quan trọng. Có lúc thì cái này quan trọng hơn cái kia. Vậy trong thực tế, ta sẽ lấy cái nào làm cái chính? Ta phải điều chỉnh sao cho cả hai cái này thật sự hợp lý? Thật khó trả lời cho các câu hỏi này! Đó là lý do vì sao F-measure ra đời!

Với F-measure, chúng ta chỉ cần quan tâm đến một chỉ số duy nhất (thay vì hai – precision và recall). F-measure được tính thông qua precision và recall bởi công thức sau đây:

# CHƯƠNG III: THỰC NGHIỆM VÀ ĐÁNH GIÁ

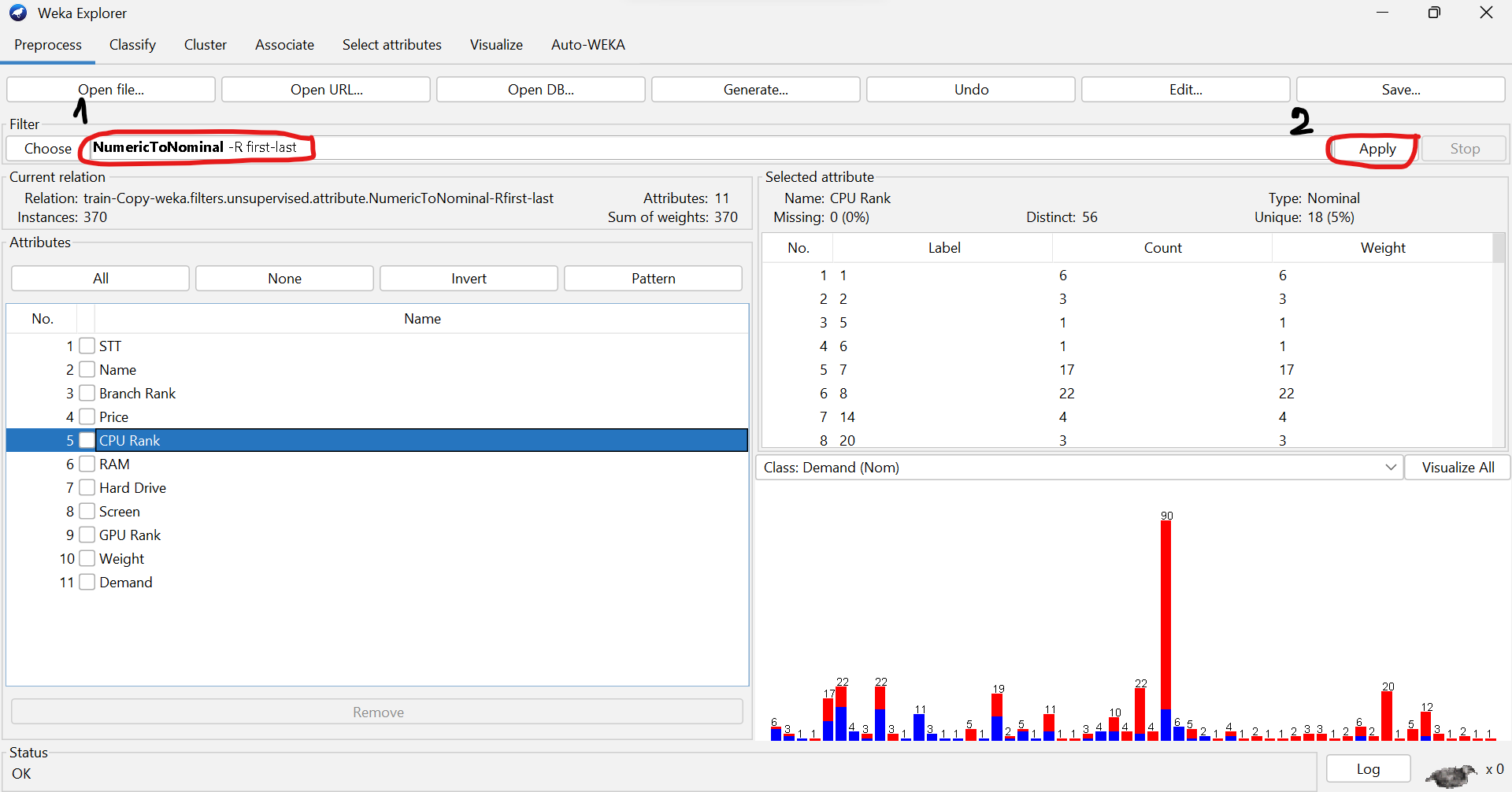
## **Thu thập và xử lý dữ liệu**

* Bộ dữ liệu thu thập được bao gồm:
* 9 Numeric: STT, Branch Rank, Price, CPU Rank, RAM, Hard Drive, Screen, GPU Rank, Demand
* 1 string: Weight
* 1 norminal: Name
* Bộ dữ liệu có dạng như hình 7:

Ảnh có chứa bàn

Mô tả được tạo tự động

Hình 7: Bộ dữ liệu

* Tiền xử lý dữ liệu: Đối với dữ liệu không chứa bất kỳ giá trị NA nào

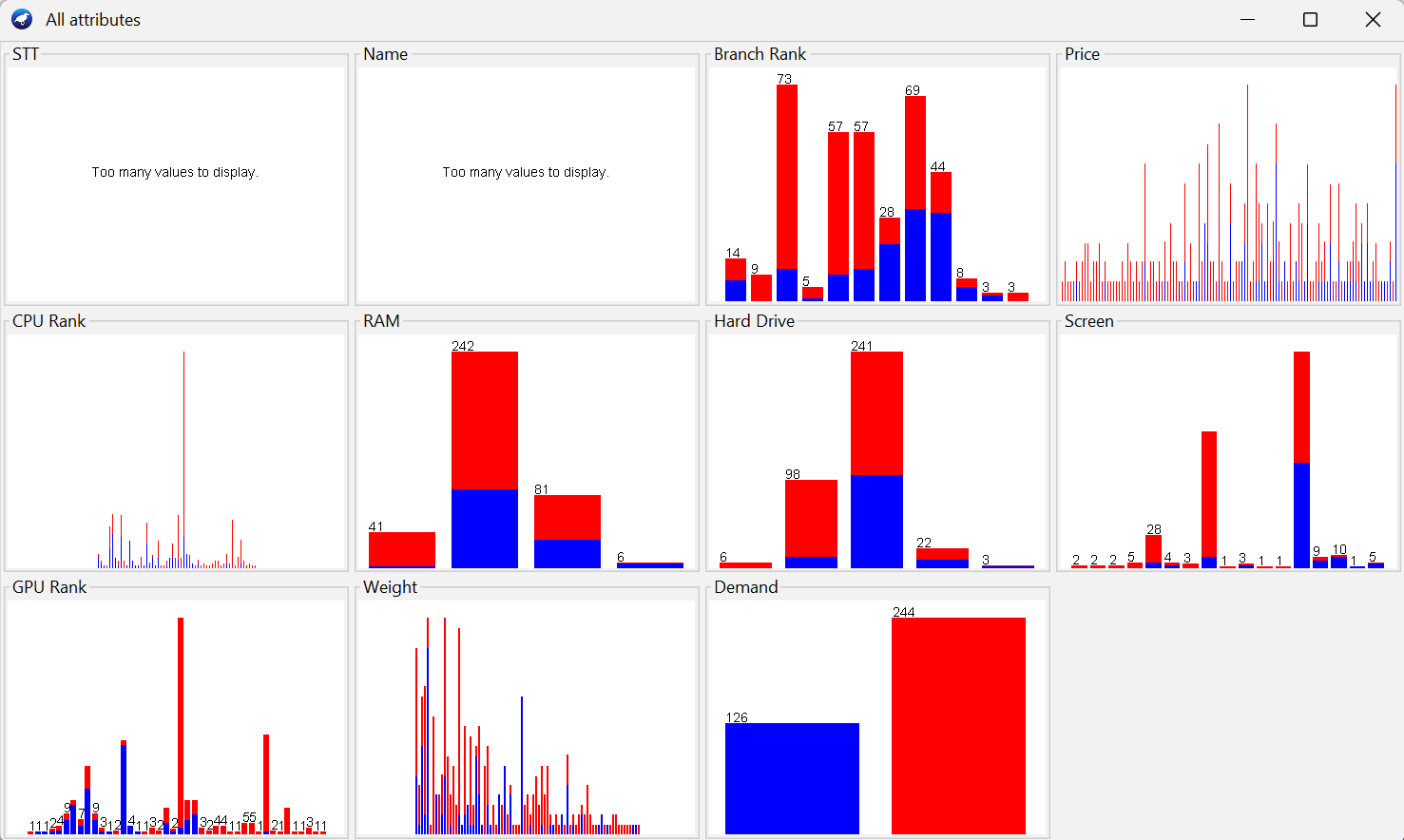
Hình 8: Chuyển đổi kiểu Numeric sang Nominal

Ảnh có chứa văn bản, trong nhà

Mô tả được tạo tự động

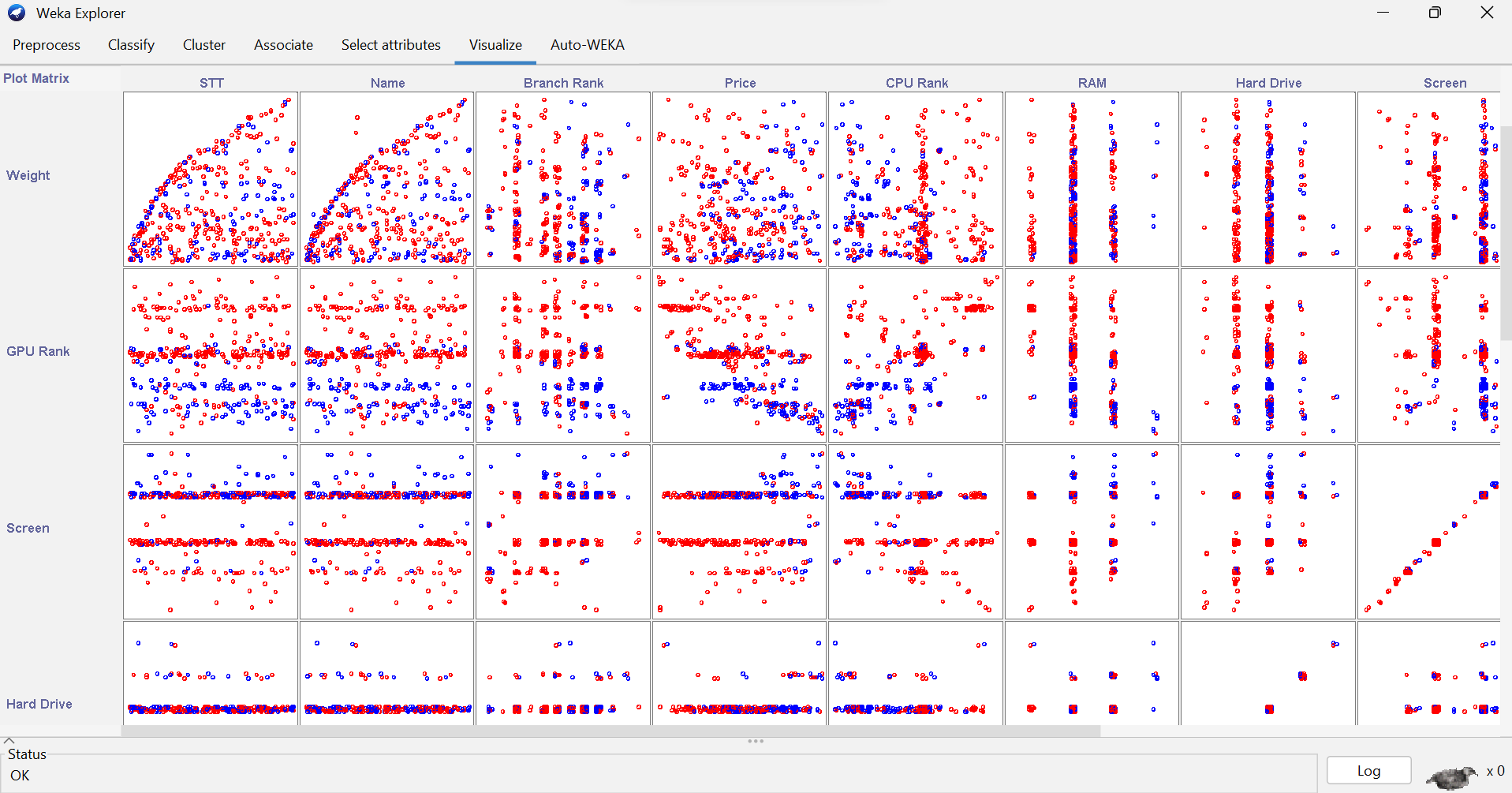
Hình 9: Chuyển đổi kiểu String sang Nominal

Chúng ta có thể nhanh chóng có được cái nhìn tổng quan về sự phân bố của tất cả các thuộc tính trong tập dữ liệu và sự phân chia của các phân phối theo lớp bằng cách nhấp chuột vào nút “Visualize All” phía trên biểu đồ đơn biến.



Hình 10: Sự phân bố của tất cả các thuộc tính

Đến bước này, chúng ta mới chỉ xem được các thuộc tính riêng lẻ. Khi các thuộc tính là số, chúng ta có thể tạo ra biểu đồ sự tương quan giữa các thuộc tính với nhau. Điều này rất hữu ích bởi nó có thể phần nào đánh giá được cột dữ liệu đó có thực sự cần thiết khi dùng nó trong việc phân lớp hay không.

Nhấp vào Visualize để quan sát và xem xét bộ dữ liệu.

Hình 11: Các biểu đồ thể hiện sự tương quan của các thuộc tính

Đối với những biểu đồ này (Hình 8), ta chỉ cần xem một nửa bởi chúng đối xứng với nhau qua một đường chéo. Để quan sát kỹ một biểu đồ, chúng ta nhấp chuột vào biểu đồ mong muốn và dùng các tuỳ chỉnh trong nó.

## **Phân lớp dữ liệu với thuật toán Random Forest**

### *2.1 Cấu hình để phân lớp dữ liệu:*

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 12: Cấu hình để thực hiện phân lớp

* Lựa chọn thuật toán phân lớp (Choose): Nhấp chọn vào thuật toán mong muốn.
* Lựa chọn các tùy chọn cho việc kiểm tra (test options)
* Use training set: Bộ phân loại học được sẽ được đánh giá trên tập học
* Supplied test set: Sử dụng một tập dữ liệu khác (với tập học) để cho việc đánh giá
* Cross-validation: Tập dữ liệu sẽ được chia đều thành k tập (folds) có kích thước xấp xỉ nhau, và bộ phân loại học được sẽ được đánh giá bởi phương pháp cross-validation
* Percentage split: Chỉ định tỷ lệ phân chia tập dữ liệu đối với việc đánh giá
* Lựa chọn nhãn để phân lớp là “Demand”.

### *Kết quả phân lớp dựa trên thuật toán Random Forest*

* Click vào Start để tiến hành phân lớp và được kết quả như hình 13.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 13: Kết quả phân lớp dữ liệu laptop với nhãn Demand bằng Random Forest

* Sử dụng kết quả đó áp dụng cho tập dữ liệu kiểm thử, ta có kết quả như hình 14

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 14: Kết quả test từ việc train của thuật toán Random Forest

## **Phân lớp dữ liệu với thuật toán Naive Bayes**

### *3.1 Cấu hình để phân lớp dữ liệu:*

* Các bước cấu hình (Hình 12):
* Bước 1: Chọn Choose để lựa chọn thuật toán Naive Bayes.
* Bước 2: Chọn như số 2 hình 12 để lựa chọn nhãn.
* Bước 3: Start để tiến hành train.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 12: Các bước cấu hình phân lớp theo thuật toán Naive Bayes

### *Kết quả phân lớp dựa trên thuật toán Naive Bayes*

Ảnh có chứa văn bản

Mô tả được tạo tự động

*Hình 13: Kết quả train theo thuật toán Naive Bayes*

Sử dụng kết quả đó áp dụng cho tập dữ liệu kiểm thử, ta có kết quả như hình 14Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 14: Kết quả kiểm thử trên kết quả của train với thuật toán Naive Bayes

## **Lựa chọn các thuộc tính dữ liệu dựa trên độ tương quan**

Lựa chọn thuộc tính dữ liệu phù hợp là quá trình giảm số lượng biến đầu vào khi phát triển mô hình dự đoán. Điều này vừa có tác dụng làm giảm chi phí tính toán, vừa nâng cao hiệu suất của mô hình. Thông thường, ta sẽ sử dụng tất cả các thuộc tính có liên quan đến nhãn để áp dụng cho bài toán phân lớp. Tuy nhiên, khi đưa toàn bộ chúng vào mô hình phân lớp đã chọn, thời gian để máy có thể học là rất lâu đồng thời kết quả dự đoán có độ chính xác thấp. Thay vì vậy, ta nên quyết định xem thuộc tính nào cần cho máy học, thuộc tính nào thì không. Ngoài ra, ta cũng có thể thu thập thêm các thuộc tính khác để bổ sung vào tập dữ liệu ban đầu. Chính vì sự biểu diễn tốt tập dữ liệu ban đầu mà độ chính xác của mô hình phân lớp được cải thiện đáng kể. Cũng như lập trình là một nghệ thuật, việc thêm bớt, chỉnh sửa các thuộc tính cũng là một nghệ thuật đòi hỏi người thực hiện phải có nhiều kinh nghiệm thực tế, va chạm nhiều với các bài toán. Sự trích chọn thuộc tính khiến chúng ta đạt được kết quả tốt mà không nhất định phải có một mô hình dự đoán tốt nhất.

Một số kỹ thuật phổ biến để chọn các thuộc tính có liên quan nhất trong tập dữ liệu là sử dụng tính tương quan. Weka hỗ trợ lựa chọn tính năng dựa trên tương quan với kỹ thuật CorrelationAttributeEval yêu cầu sử dụng phương pháp tìm kiếm Rank. Kết quả thực hiện như sau:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 1515: Kiểm tra tính tương quan

Như vậy, hệ số tương quan giữa Demand và Price là kém nhất. Ngoài ra, có một dẫn chứng có thể giúp ta kết luận việc loại bỏ thuộc tính price, đó là phát hiện luật kết hợp trong bộ dữ liệu.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 16: Áp dụng luật kết hợp

Những luật kết hợp phát hiện ra trong bộ dữ liệu hoàn toàn không có sự xuất hiện của thuộc tính “Weight” và “Price” (Hình 13). Điều này càng ủng hộ việc loại bỏ thuộc tính “Price” và cho máy học lại. Xem xét sự thay đổi:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 17: Kết quả học lại khi bỏ thuộc tính Price

Độ chính xác đã tăng lên xấp xỉ 1%. Tuy nhiên, đây vẫn chưa chính thức là một kết quả tốt. Chúng ta cần xem xét với kết quả học này, bộ dữ liệu mới được đưa vào có được gán nhãn tốt hơn hay không.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình 18: Kết quả test với việc học lại không có thuộc tính Price

Như ta đã thấy, độ chính xác của mô hình trong việc kiểm thử bởi bộ dữ liệu testing data có kết quả tốt hơn (76.8421% -> 82.1053%). Điều này phần nào chứng tỏ việc loại bỏ thuộc tính price có tác động tích cực đến mô hình phân lớp trong trường hợp này.

## **Kết luận**

Với bộ dữ liệu mà nhóm đưa ra được thu thập từ nhiều nguồn nên độ chính xác khi phân lớp chỉ mang tính tương đối. Nhưng qua việc phân lớp qua hai thuật toán Random Forest và Naive Bayes có thể thấy được đối với bộ dữ liệu như ban đầu được đưa ra với các thuộc tính đó thì thuật toán Naive Bayes thực hiện với độ chính xác cao hơn (Random Forest - 76.8421%, Naive Bayes - 82.1053%). Với độ chính xác đó, thuật toán xây dựng được từ bộ dữ liệu này có thể mang áp dụng vào việc phân lớp nhu cầu loại laptop, và nhân viên có thể thực hiện tư vấn cho khách hàng mua chiếc laptop theo nhu cầu của họ.

Để bộ dữ liệu ban đầu tăng thêm độ chính xác, nhóm cũng tìm hiểu và nhận ra rằng bỏ đi thuộc tính “Price” trong bộ dữ liệu gốc ban đầu thì độ chính xác tăng lên đáng kể (~ 6%). Từ đó, tạm thời ta có thể thấy giá cả không có tính tương quan cao đối với nhu cầu phục vụ của một chiếc laptop. Tuy nhiên, một thuộc tính muốn xét xem nó có ích cho các nhãn phân lớp hay không ta vẫn nên giữ lại. Bởi có nhiều trường hợp do dữ liệu quá ít hoặc do độ có ích của bản thân nó bị che lấp bớt, nên ta vẫn phải giữ lại xem xét nếu cần.

Sau khi thực hiện đề tài, nhóm chúng em nhận ra một số điểm hạn chế như sau:

* Thiếu dữ liệu: Sự thiếu dữ liệu được xét theo hai khía cạnh. Thứ nhất, bộ dữ liệu được xây dựng còn khá nhỏ, có thể chưa đủ lớn để tạo ra một mô hình mô tả kỹ càng bộ dữ liệu. Thứ hai, các trường thuộc tính đưa vào chưa đủ, có thể còn tồn tại những thuộc tính khác có ảnh hưởng đến việc phân nhãn cho dữ liệu.
* Quá tập trung vào việc chọn mô hình tốt: Tìm được mô hình tốt, phù hợp với ngữ cảnh của bài toán khai phá dữ liệu là định hướng chủ đạo của quá trình khai phá dữ liệu. Nhưng đối với bài toán khai phá dữ liệu, chúng ta cần tìm “lời giải đủ tốt” chứ không phải “lời giải tối ưu”, việc tìm mô hình nên có một mức độ phù hợp, song cũng cần thời gian và công thức cho các công việc khác của khai phá như tiền xử lý dữ liệu, trích chọn đặc trưng, biểu diễn, trực quan hoá dữ liệu.
* Dữ liệu sai: Trong quá trình thu thập, khó tránh khỏi tình trạng dữ liệu sai. Trong bài làm có thể mắc phải những dòng bị sai lệch thông tin do nguyên nhân chủ quan từ phía người thu thập dữ liệu.

# CHƯƠNG IV: TỔNG KẾT

Bài báo cáo đã đề cập đến các nội dung về kho dữ liệu và ứng dụng của lưu trữ và khai phá tri thức trong kho dữ liệu. Từ đó xây dựng nên kho dữ liệu về máy tính xách tay và thực hiện khai phá cho mục đích tư vấn cho khách hàng chọn phù hợp với nhu cầu.

Về mặt lý thuyết, khai phá tri thức bao gồm các bước: Hình thành, xác định và định nghĩa bài toán, thu thập và tiền xử lý dữ liệu, khai phá dữ liệu, rút ra các tri thức, sử dụng các tri thức phát hiện được. Phương pháp khai phá dữ liệu có thể là: phân lớp, cây quyết định, suy diễn… Các phương pháp trên có thể áp dụng trong dữ liệu thông thường. Về thuật toán khai phá tri thức, bài làm trình bày một số thuật toán và minh họa một thuật toán kinh điển: Random Forest và Naive Bayes.

Về mặt thử nghiệm, bài báo cáo giới thiệu kĩ thuật khai phá dữ liệu theo hai thuật toán Random Forest và Naive Bayes, áp dụng khai phá qua phần mềm Weka vào bài toán gán nhãn nhu cầu của laptop để phục vụ việc tư vấn cho khách hàng.

Qua quá trình thực nghiệm, độ chính xác mà bộ dữ liệu nhóm đưa ra chưa thực sự cao để có thể áp dụng vào thực tế mà chỉ mang tính chất tham khảo. Và để khắc phục cần phải chọn lọc các thuộc tính một cách kĩ càng, làm giàu dữ liệu hơn và dữ liệu được chuyên gia đánh giá sẽ có kết quả khả quan hơn.

Trong quá trình thực hiện bài, nhóm đã cố gắng tập trung tìm hiểu và tham khảo các tài liệu liên quan. Tuy nhiên, với thời gian và trình độ có hạn nên không tránh khỏi những hạn chế và thiếu sót. Nhóm rất mong nhận được các nhận xét và góp ý của thầy để hoàn thiện hơn các kết quả nghiên cứu.

# TÀI LIỆU THAM KHẢO

1. Nguyễn Nhật Quang (2010-2011), *Khai Phá Dữ Liệu*, Viện Công nghệ Thông tin và Truyền thông đại học Bách khoa Hà Nội.
2. Nguyễn Nhật Quang (2011-2012), *Học Máy,* Viện Công nghệ Thông tin và Truyền thông đại học Bách khoa Hà Nội.
3. Nguyễn Hà Nam – Nguyễn Trí Thành – Hà Quang Thuỵ (2016), *Giáo trình Khai Phá Dữ Liệu*, Nhà xuất bản Đại học Quốc gia Hà Nội.