TRƯỜNG ĐẠI HỌC THỦY LỢI

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

HỌC PHẦN: KHAI PHÁ DỮ LIỆU

**ĐỀ TÀI:**

**Khai phá dữ liệu bằng phương pháp phân lớp để phân loại mức độ hài lòng của khách hàng trên sàn thương mại điện tử**

*Giảng viên hướng dẫn: Trần Mạnh Tuấn*

Nhóm sinh viên thực hiện:

1. Nguyễn Ngọc Bách - 2151163668

2. Trương Quốc Bảo - 2151163669

3. Lê Mạnh Duy - 2151163677

4. Nguyễn Văn Nam - 2151160525

**Hà Nội, năm 2024**

**MỤC LỤC**

[**I. MÔ TẢ BÀI TOÁN 4**](#_ddc3gh3sapj5)

[A. Lý do chọn đề tài 4](#_q59woivbsvjw)

[B. Quy trình thực hiện 4](#_7eg09hdbuned)

[C. Phân tích dữ liệu thô 5](#_d8pigik1wnk4)

[**II. QUY TRÌNH KHAI PHÁ DỮ LIỆU 5**](#_7bp3w57s5s4l)

[A. Tiền xử lý dữ liệu 5](#_t25xgm6gyah0)

[1. Làm sạch dữ liệu 6](#_1x4ezyau3id9)

[a) Loại bỏ thuộc tính dư thừa 6](#_q4z4vupxu0z0)

[b) Kiểm tra giá trị nhiễu (outliers) 6](#_f9n08sg0w5fg)

[c) Xử lý dữ liệu bị thiếu 7](#_f6eajeyp97ft)

[d) Rời rạc hóa dữ liệu 8](#_hu96z7a807z)

[2. Tích hợp dữ liệu 14](#_5i8j4swr1kx0)

[3. Biến đổi dữ liệu (chuẩn hóa) 14](#_2zt9pi1f0p5d)

[B. Phân tích dữ liệu 14](#_jwib4848elas)

[**III. KHAI PHÁ LUẬT KẾT HỢP 16**](#_2gbrmtoyrxm9)

[A. Khai phá luật kết hợp 16](#_pwcjr3ht9ysf)

[B. Quy trình thực hiện thuật giải Apriori 16](#_ayhim939srd3)

[C. Khai phá luật kết hợp bằng Apriori trong weka 17](#_69vxj6ofnvbo)

[D. Khai phá luật kết hợp bằng Apriori trong python 19](#_xftnmrox743)

[E. Nhận xét đánh giá 23](#_mrbuwhzg7nne)

[**IV. PHƯƠNG PHÁP PHÂN LỚP - NAIVE BAYES 24**](#_jdva1nz2tsfy)

[A. Thuật toán Naive Bayes 24](#_p8vadtd3qicx)

[B. Các loại phân phối thường dùng 25](#_x86ig9ldsnh0)

[1. Gaussian Naive Bayes 25](#_7826baarkzgs)

[2. Multinomial Naive Bayes 25](#_o96agu69q2xh)

[3. Bernoulli Naive Bayes 26](#_o6xvrcy6zcsv)

[C. Phân lớp Naive Bayes trong weka 27](#_705nd4ou5ru0)

[D. Phân lớp Naive Bayes trong python 29](#_gkcbpbtkh0ls)

[E. Biểu diễn tri thức 38](#_2mnd3rf7fbiw)

[F. Sản phẩm Demo 38](#_buf772i8u90)

[**KẾT LUẬN 44**](#_a9f78i6adu4)

[**TÀI LIỆU THAM KHẢO 45**](#_yc6d9w9sign)

**BẢNG PHÂN CHIA CÔNG VIỆC**

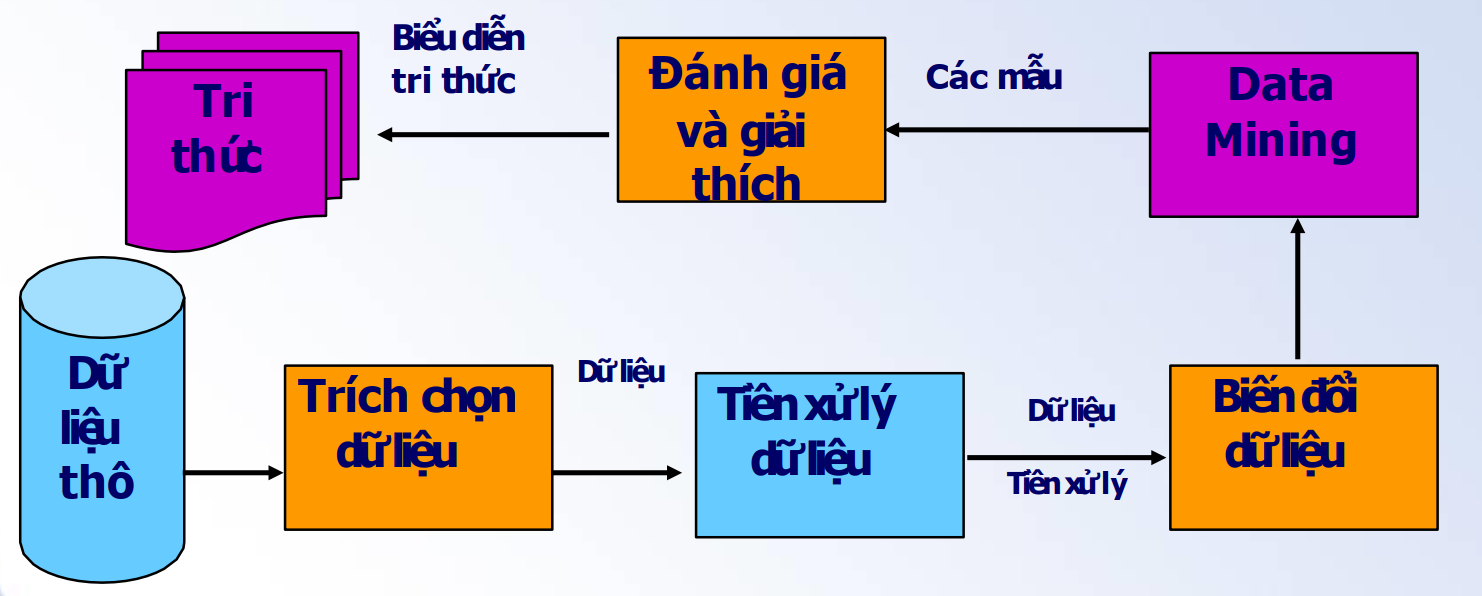
| **Người phụ trách** | **Công việc** |
| --- | --- |
| Nguyễn Ngọc Bách | * Lên ý tưởng * Tiền xử lý dữ liệu * Xây dựng luật kết hợp |
| Nguyễn Văn Nam | * Xây dựng thuật toán Naive Bayes * Làm giao diện |
| Trương Quốc Bảo | * Lý thuyết * Xây dựng thuật toán Naive Bayes |
| Lê Mạnh Duy | * Xây dựng luật kết hợp * Làm giao diện |

# MÔ TẢ BÀI TOÁN

## Lý do chọn đề tài

Sự chuyển mình của ngành thương mại điện tử về phía số lượng người dùng đòi hỏi việc áp dụng các phương pháp tiên tiến để đánh giá mức độ hài lòng. Phân lớp người dùng và nhận diện sớm phân loại các sản phẩm phù hợp là một trong những nhiệm vụ cấp thiết. Trong bối cảnh này, việc sử dụng mô hình Naive Bayes có thể cung cấp cái nhìn sâu sắc và giải pháp hữu hiệu cho vấn đề phức tạp này. Đề tài **"Khai phá dữ liệu bằng phương pháp phân lớp để phân loại mức độ hài lòng của khách hàng trên sàn thương mại điện tử"** không chỉ phản ánh sự kết hợp giữa lý thuyết và thực tiễn mà còn mang lại cơ hội để chúng em nghiên cứu và phát triển kỹ năng giải quyết vấn đề cụ thể, góp phần vào sự phát triển bền vững của ngành thương mại điện tử trong kỷ nguyên số.

## Quy trình thực hiện



*Hình 1: Quy trình thực hiện khai phá dữ liệu*

Quy trình thực hiện khai phá bao gồm 6 bước:

* Bước 1: Tạo tập tin dữ liệu đầu vào
* Bước 2: Tiền xử lý, làm sạch tập dữ liệu
* Bước 3: Chọn tác vụ khai phá dữ liệu (phân lớp)
* Bước 4: Khai phá dữ liệu: tìm kiếm tri thức
* Bước 5: Đánh giá mẫu tìm được
* Bước 6: Biểu diễn tri thức

Ở bài này, nhóm em tóm tắt các bước thành những mục sau:

* Thu thập dữ liệu
* Tiền xử lý dữ liệu
* Biến đổi dữ liệu

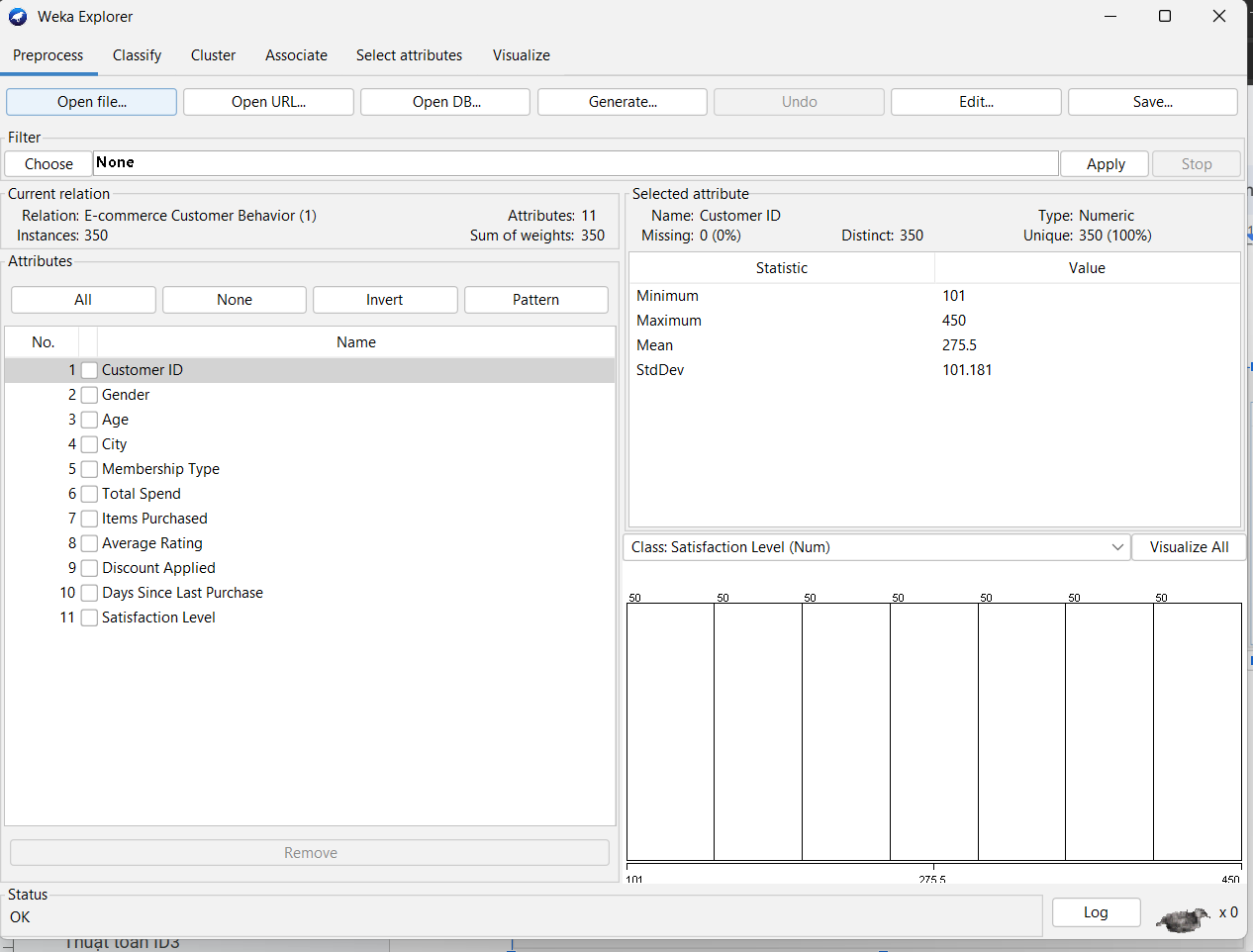
## Phân tích dữ liệu thô

* Nguồn dữ liệu thô: <https://www.kaggle.com/datasets/uom190346a/e-commerce-customer-behavior-dataset>
* Hiểu dữ liệu: Bộ dữ liệu cung cấp thông tin để phân tích và dự đoán mức độ hài lòng của khách hàng. Từ đó ta đưa ra được những sản phẩm phù hợp.
  + Số Lượng Bản Ghi: 350.
  + Số Lượng Thuộc Tính: 11.
  + Bao gồm thông tin về: Giới tính, Tuổi, Thành phố, Loại thành viên, Tổng chi tiêu, Các mặt hàng đã mua, Đánh giá trung bình, Áp dụng giảm giá, Số ngày kể từ lần mua cuối cùng,
  + **Satisfaction Level**: Mức độ hài lòng.

# QUY TRÌNH KHAI PHÁ DỮ LIỆU

## Tiền xử lý dữ liệu

Đọc file dữ liệu trên weka:

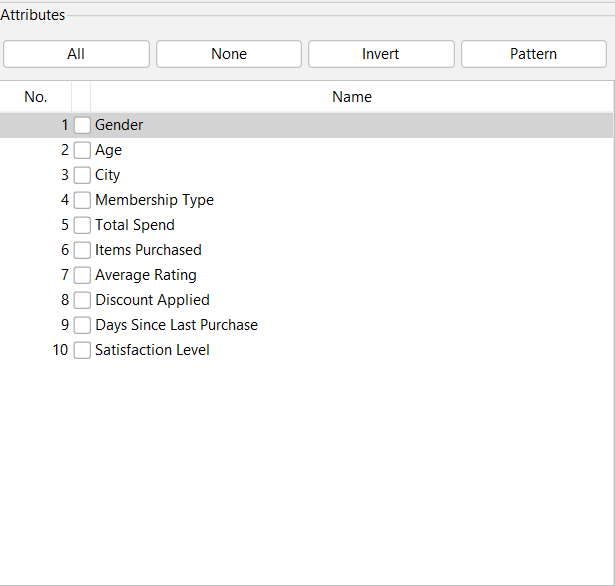


*Hình 2: Đọc file dữ liệu trên weka*

### Làm sạch dữ liệu

#### Loại bỏ thuộc tính dư thừa

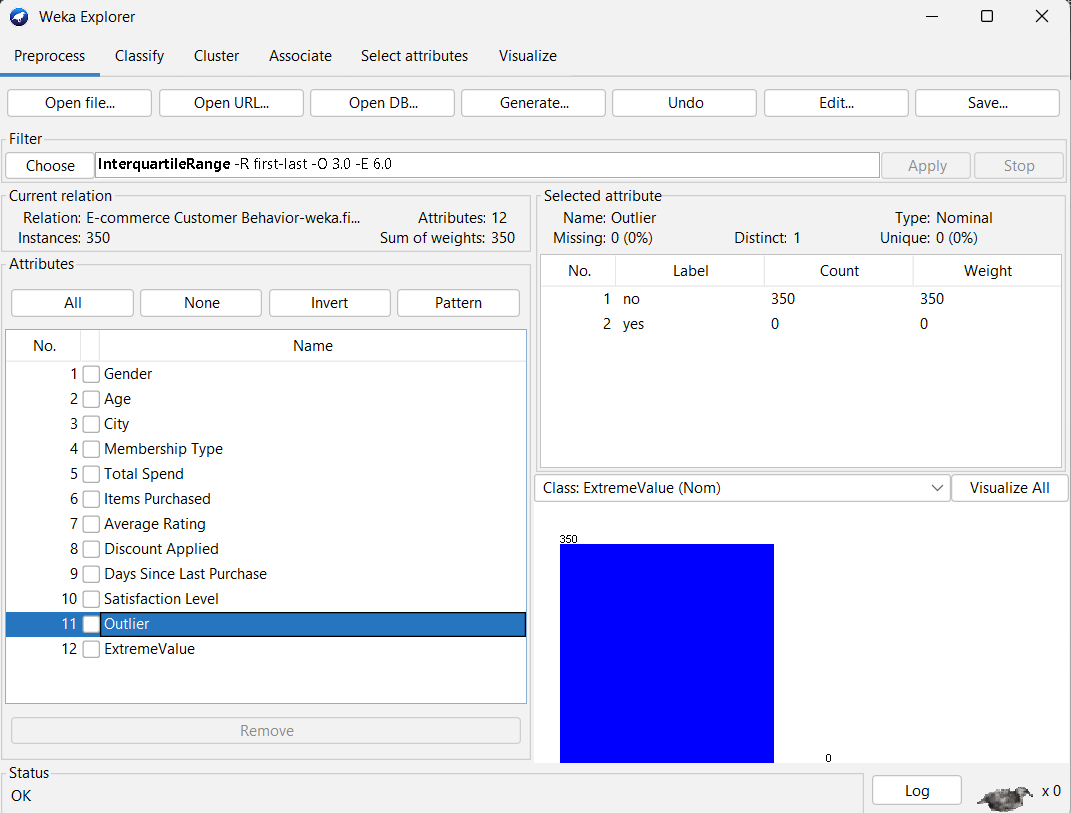
* Loại bỏ các thuộc tính dư thừa, không cần thiết sau: Customer ID



*Hình 3: Các thuộc tính còn lại sau xóa*

#### Kiểm tra giá trị nhiễu (outliers)

* Sử dụng bộ lọc **InterquartileRange** để phát hiện các giá trị ngoại lệ (outlier) và giá trị cực đoan (extreme values).
* Chọn **Filter -> weka -> filters -> unsupervised -> attribute -> InterquartileRange**

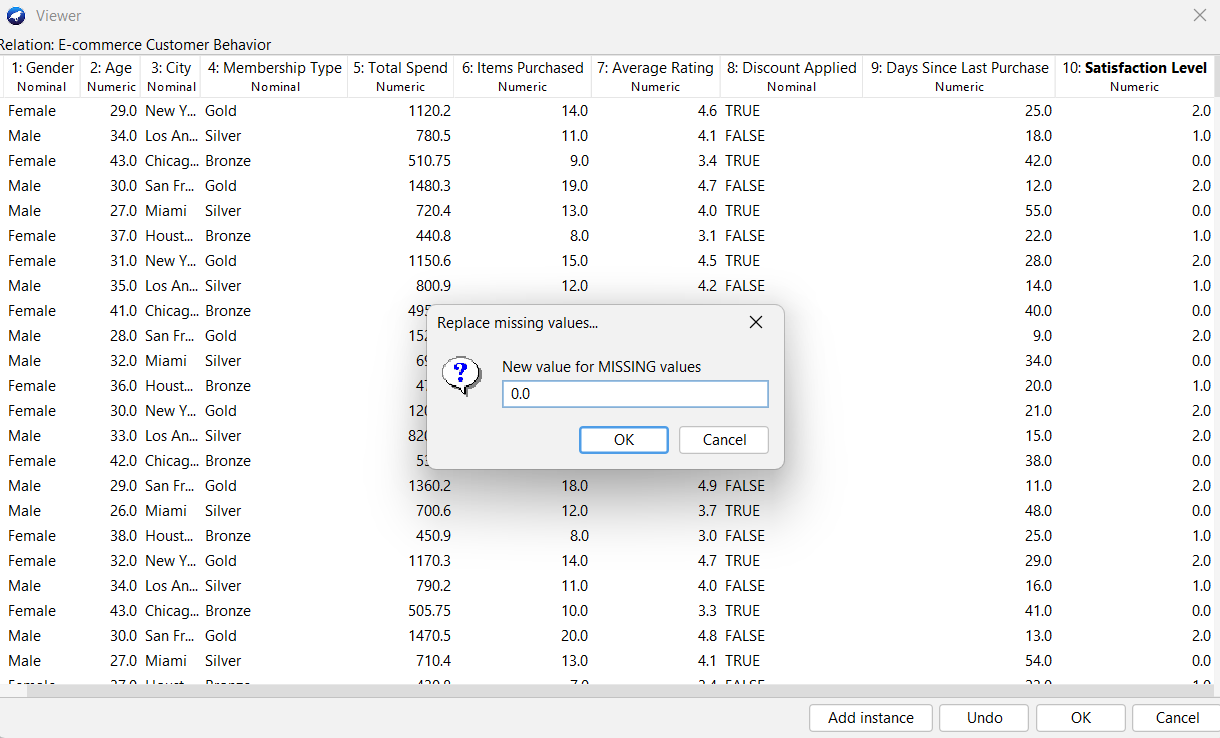


*Hình 4: Kiểm tra giá trị nhiễu (outliers)*

=> Nhận thấy không có phần tử ngoại lệ và cực đoan nên không cần loại bỏ.

#### Xử lý dữ liệu bị thiếu

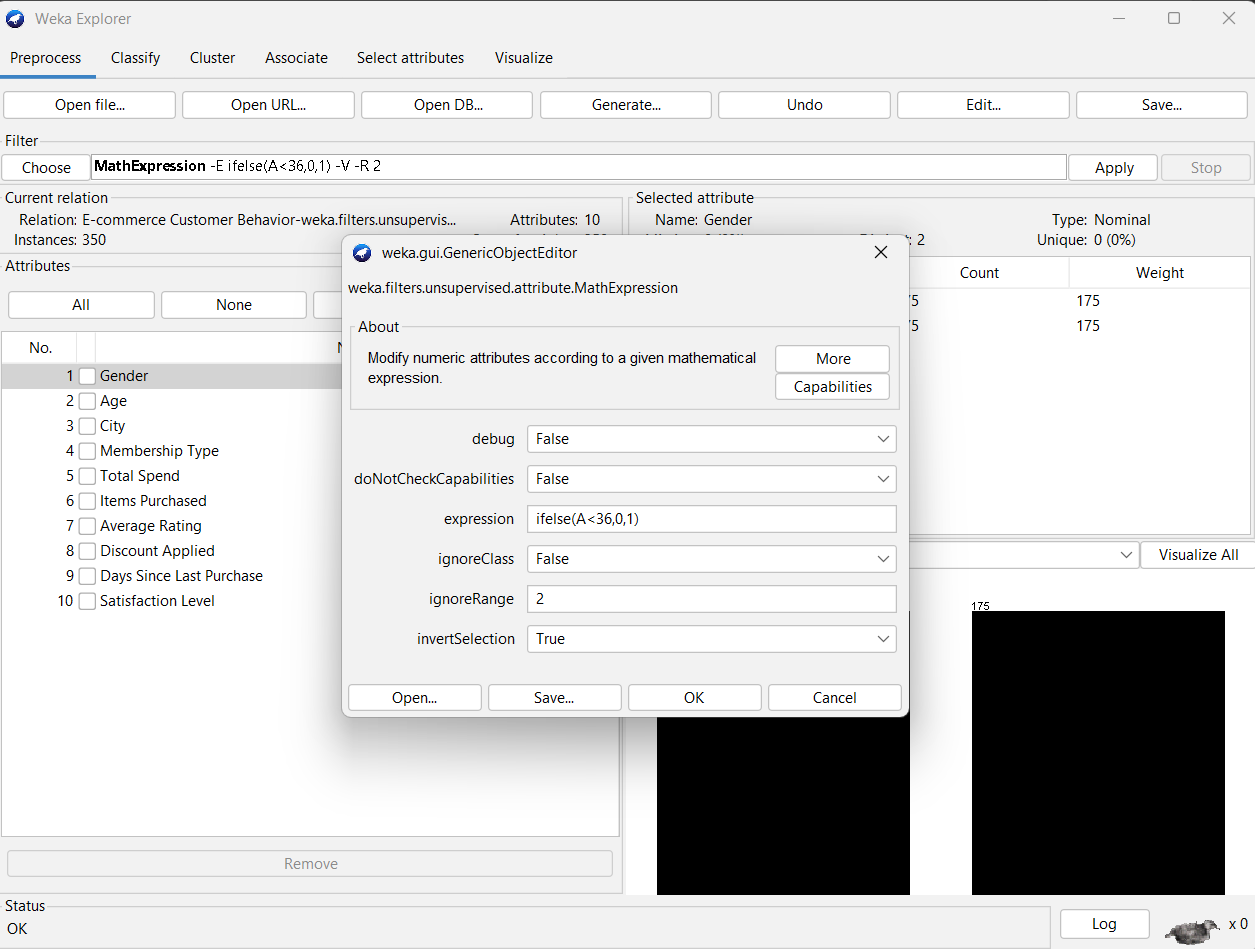
* Các trường thuộc tính có missing value (1%): Satisfied Level
* Chọn **Edit -> chuột phải vào cột Satisfied Level -> Set missing value to -> chọn 0.0**

****

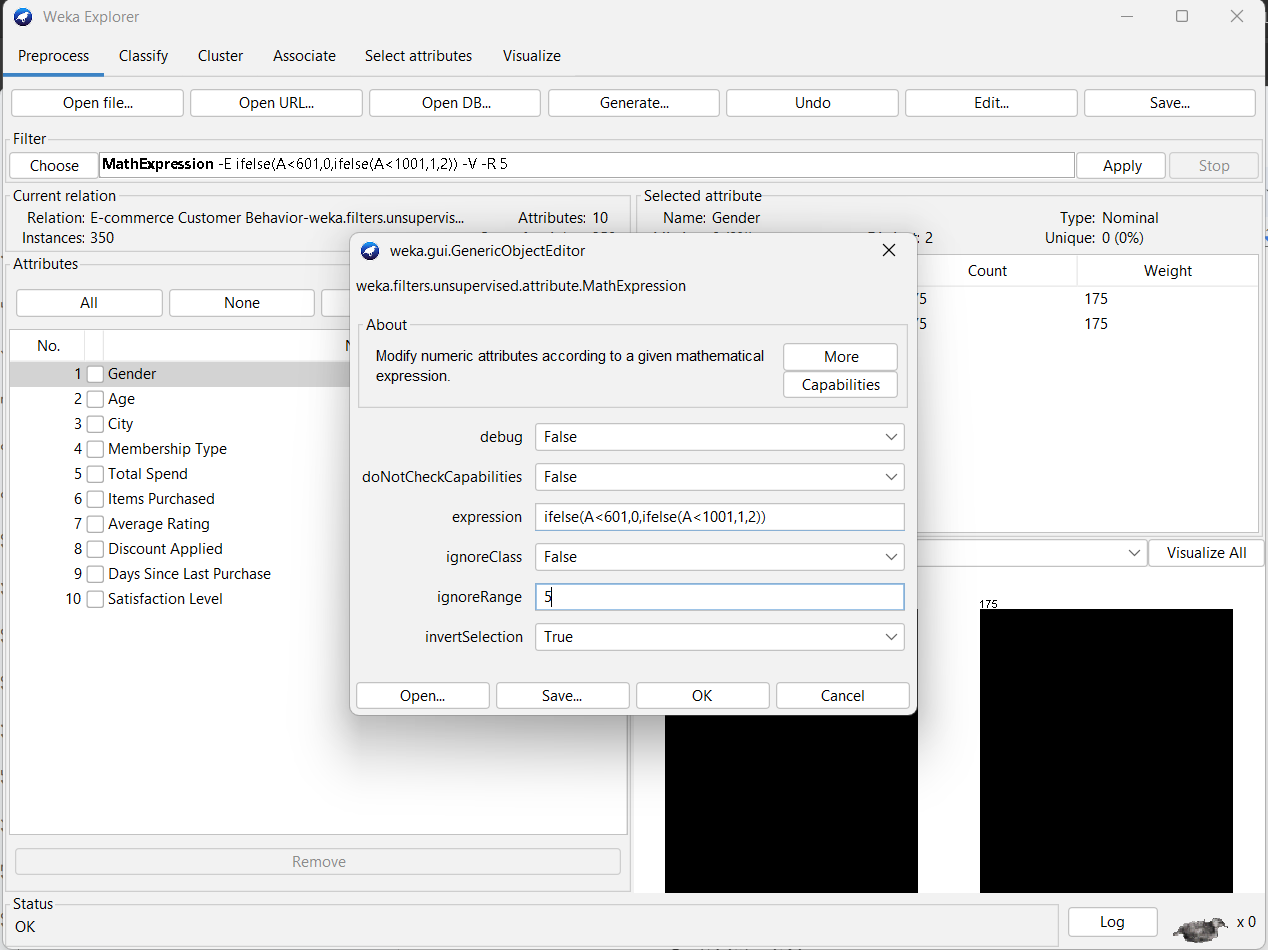
*Hình 5: Xử lý dữ liệu bị thiếu*

#### Rời rạc hóa dữ liệu

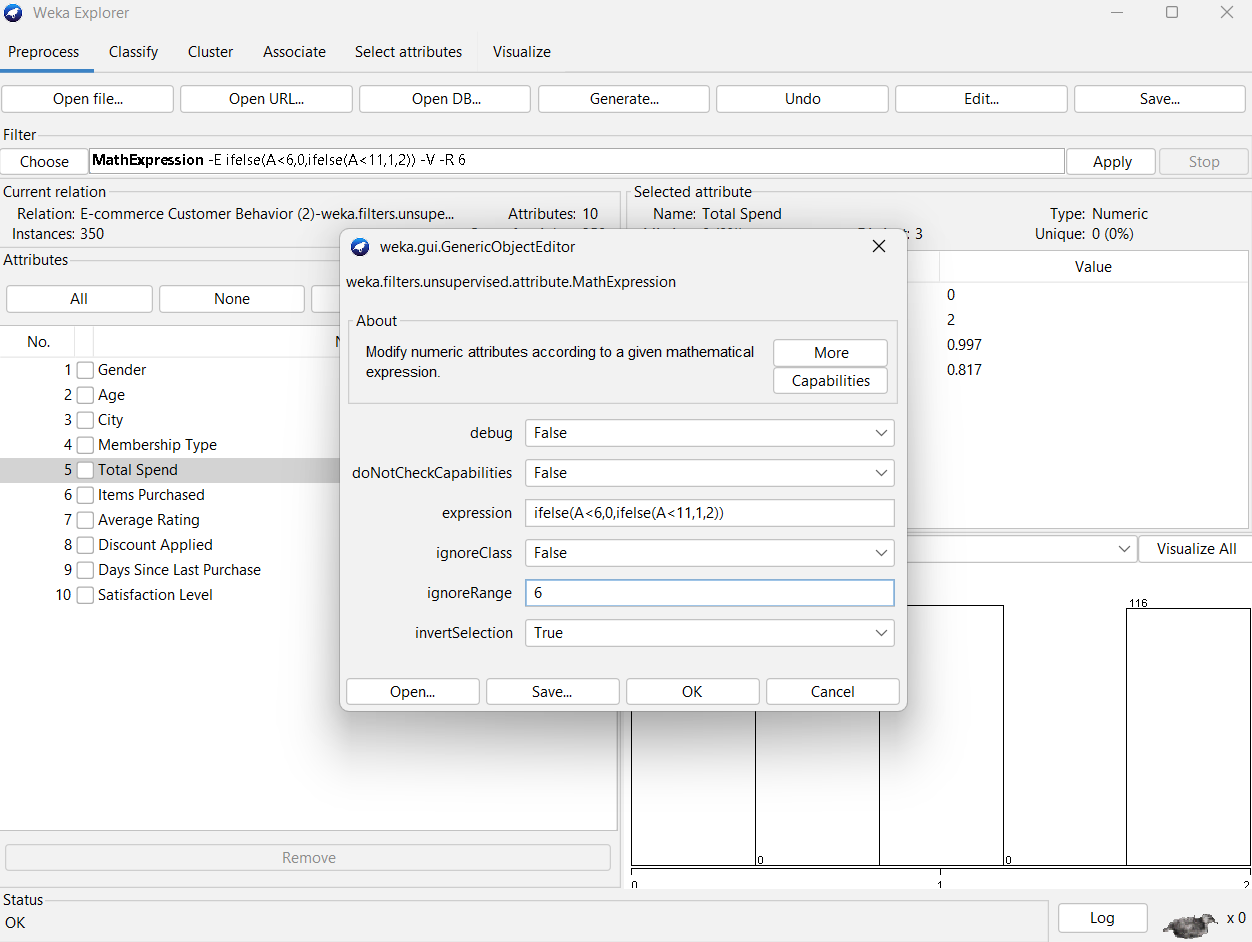
* MathExpression 5 thuộc tính Age, Total Spend, Items Purchased, Average Rating, Days Since Last Purchase
* Chọn **Filter -> weka -> filters -> unsupervised -> attribute -> MathExpression**
* Chia Age thành 2 nhóm
* 26-35: young adults.
* 36-43: middle-aged adults.
* Chia Total Spend thành 3 nhóm
* Low spending (410 - 600)
* Average spending (601 - 1000)
* High spending (1001 - 1520)
* Chia Items Purchased thành 3 nhóm
* Little product (2 - 5)
* Average product (6 - 10)
* Many product (11 - 21)
* Chia Average Rating thành 3 nhóm
* Low rating (3 - 3.9)
* Average rating (4 - 4.4)
* High rating (4.5 - 4.9)
* Chia Days Since Last Purchase thành 3 nhóm
* Recent purchase (9 - 21 days)
* Regular purchase (22 - 42 days)
* Infrequent purchase (43 - 63 days)

****

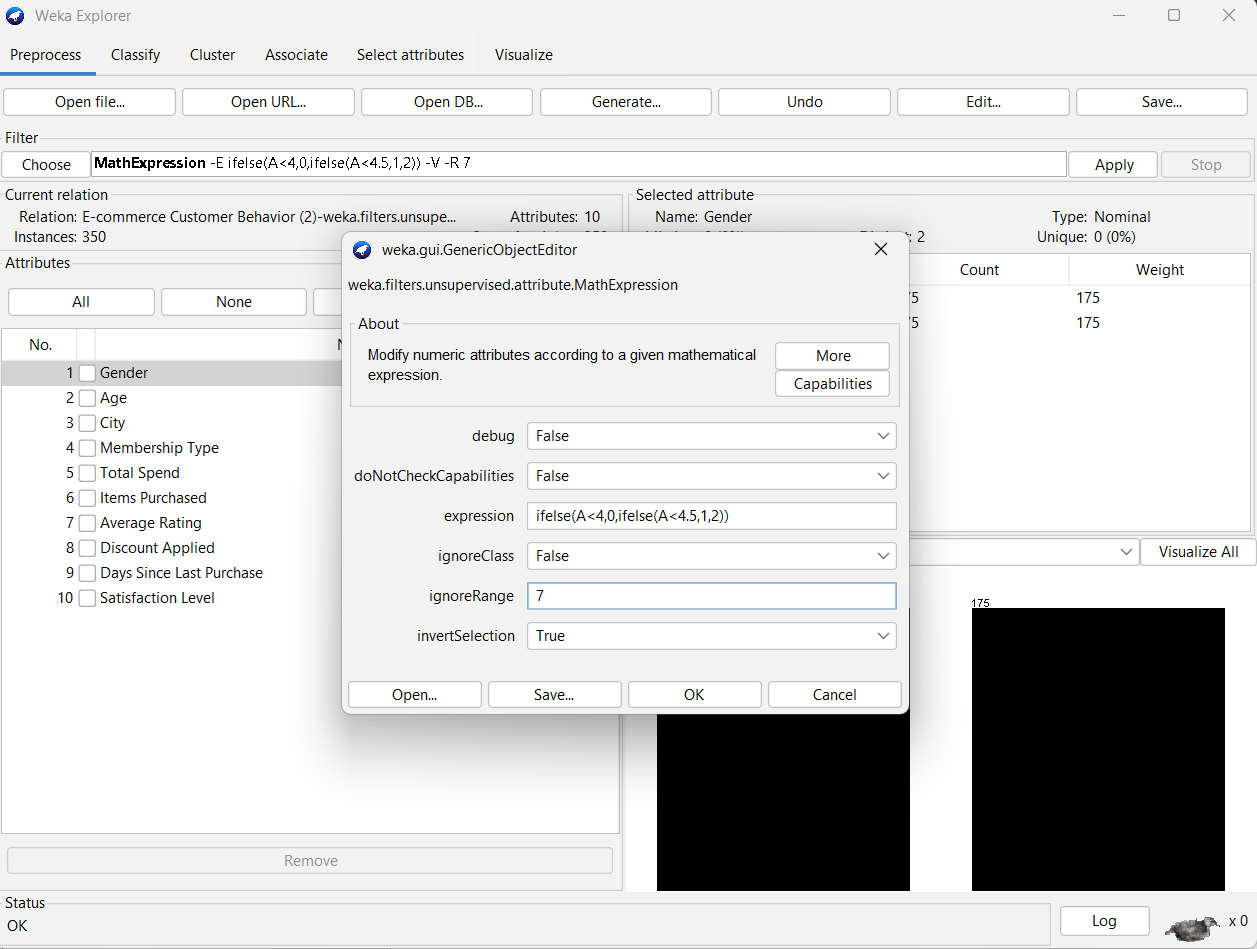
*Hình 6: Rời rạc hóa dữ liệu Age*

****

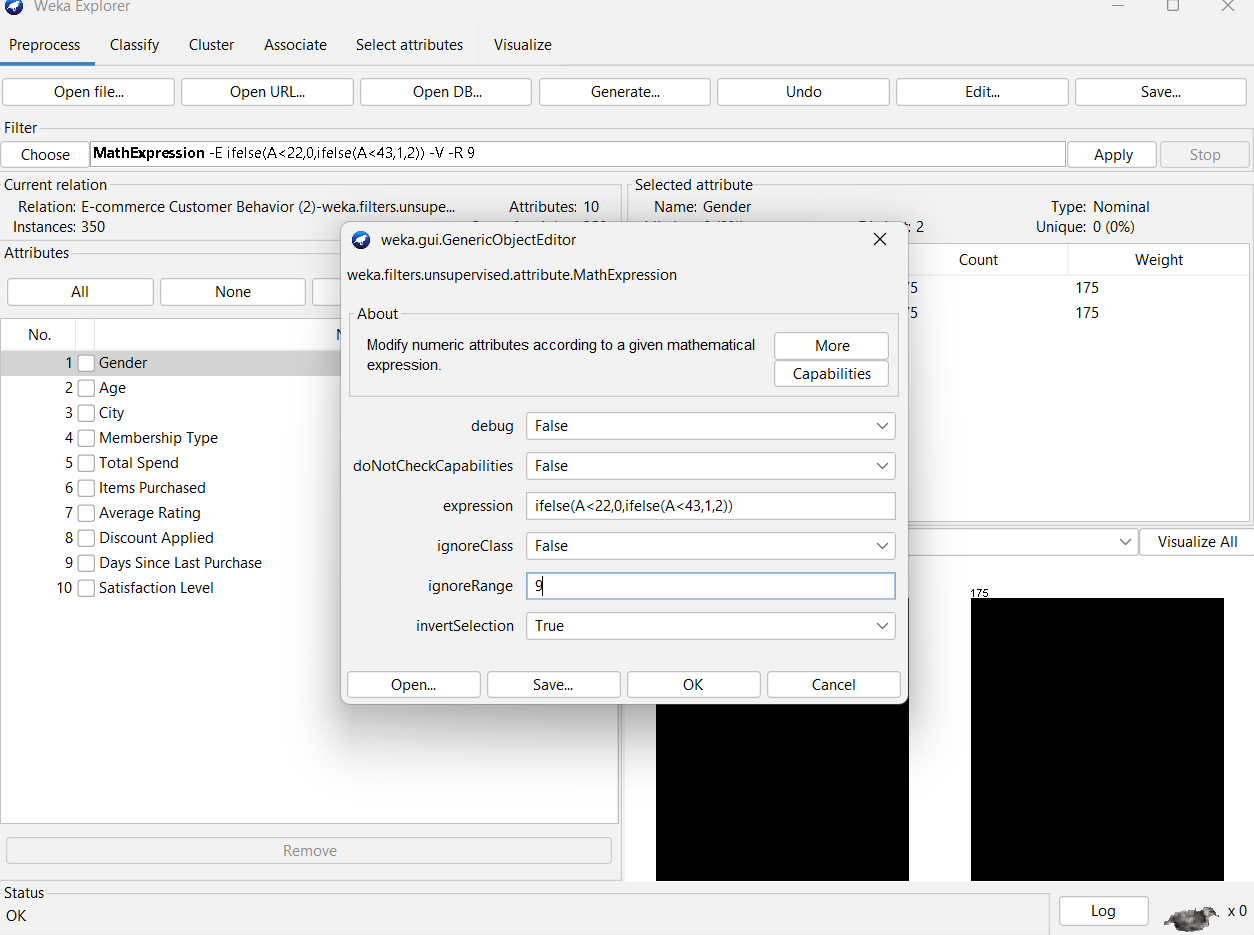
*Hình 7: Rời rạc hóa dữ liệu Total Spend*

****

*Hình 8: Rời rạc hóa dữ liệu Item Purchased*

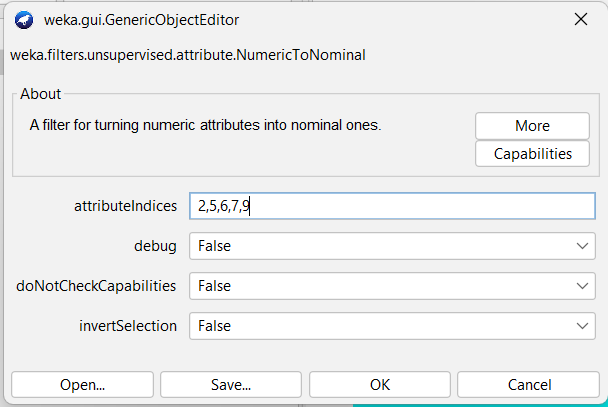
****

*Hình 9: Rời rạc hóa dữ liệu Average Rating*

****

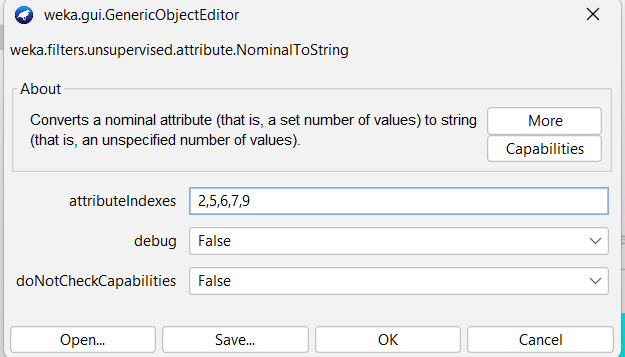
*Hình 10: Rời rạc hóa dữ liệu Days Since Last Purchase*

* Chuyển về chữ có nghĩa, NumbericToNominal 2,5,6,7,9



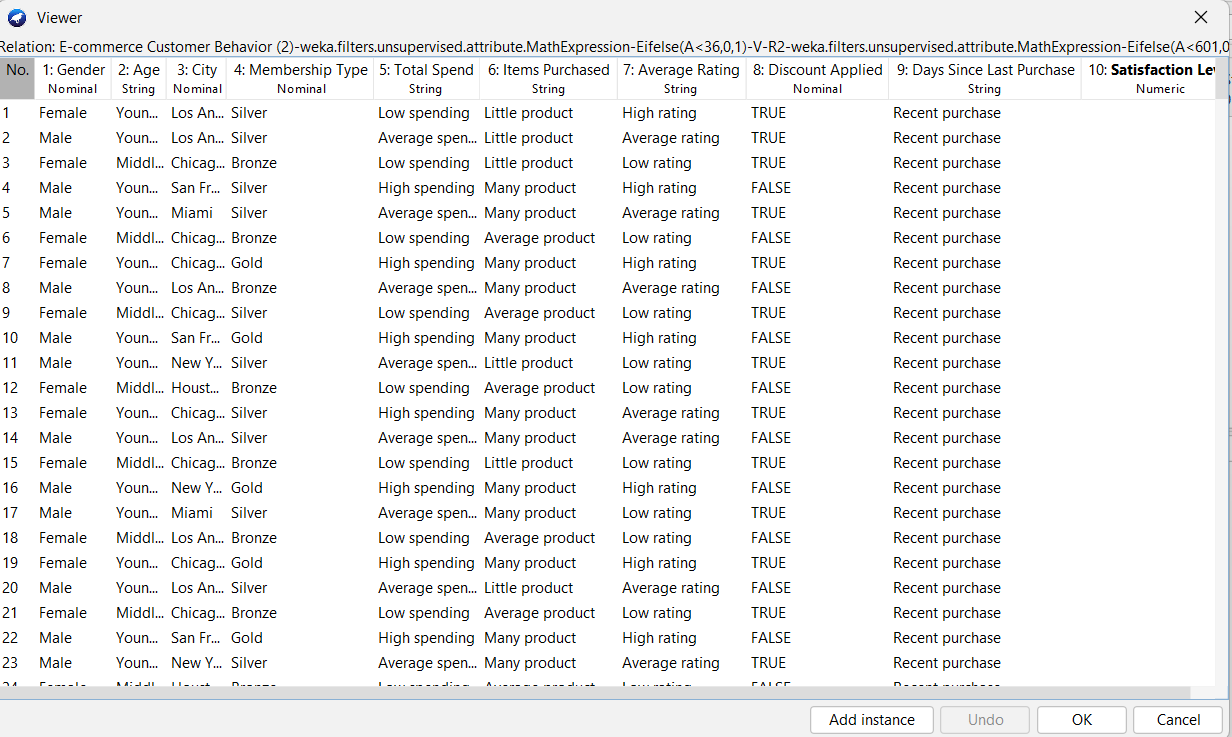
*Hình 10: Chuyển kiểu từ NumericToNominal* 2,5,6,7,9

* NominalToString 2,5,6,7,9



*Hình 11: Chuyển kiểu từ NumericToNominal* 2,5,6,7,9

* Chọn edit và thay đổi từng giá trị trong thuộc tính
* Chia Age thành 2 nhóm
* 0 - Young adults
* 1 - Middle-aged adults
* Chia Total Spend thành 3 nhóm
* 0 - Low spending
* 1 - Average spending
* 2 - High spending
* Chia Items Purchased thành 3 nhóm
* 0 - Little product
* 1 - Average product
* 2 - Many product
* Chia Average Rating thành 3 nhóm
* 0 - Low rating
* 1 - Average rating
* 2 - High rating
* Chia Days Since Last Purchase thành 3 nhóm
* 0 - Recent purchase
* 1 - Regular purchase
* 2 - Infrequent purchase



*Hình 12: Dữ liệu sau khi rời rạc hóa*

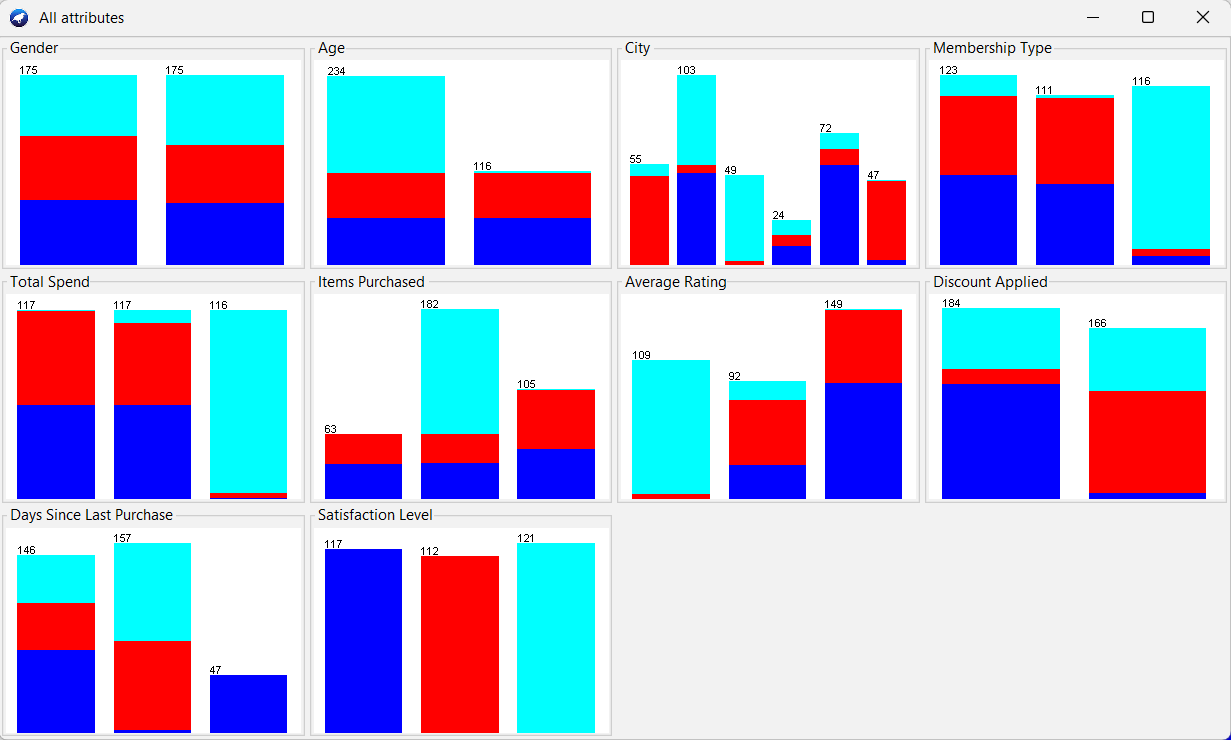
### Tích hợp dữ liệu

* Là quá trình trộn dữ liệu từ các nguồn khác nhau vào một kho dữ liệu sẵn sàng cho quá trình khai phá dữ liệu.
* Liên quan đến cấu trúc và tính không thuần nhất (heterogeneity) về ngữ nghĩa (semantics) của dữ liệu.
* Hỗ trợ việc giảm và tránh dư thừa và không nhất quán về dữ liệu > cải thiện tính chính xác và tốc độ quá trình khai phá dữ liệu.

=> Do dữ liệu lấy từ 1 nguồn nên quy trình này bỏ qua.

### Biến đổi dữ liệu (chuẩn hóa)

* Là quá trình biến đổi hay kết hợp dữ liệu vào những dạng thích hợp cho quá trình khai phá dữ liệu.
* Chuẩn hóa dữ liệu Numeric sang Nominal và String sang Nominal.
* Chọn **Filter -> weka -> filters -> unsupervised -> attribute -> StringToNominal**
* Chọn **Filter -> weka -> filters -> unsupervised -> attribute -> NumbericToNominal**



*Hình 13: Các thuộc tính sau chuẩn hóa sang Nominal*

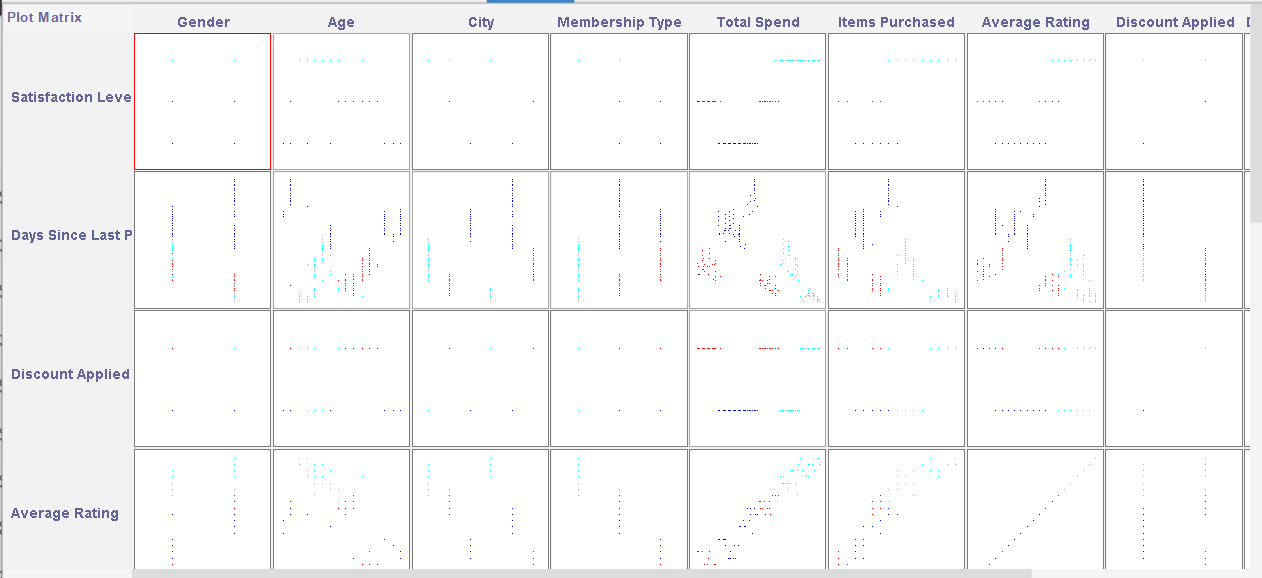
## Phân tích dữ liệu

Phân tích dữ liệu được thực hiện để có cái nhìn sâu sắc hơn về bản chất của dữ liệu và để khám phá các mối liên kết giữa các thuộc tính, từ đó giúp ta hiểu rõ hơn về các mối quan hệ và sự tương tác giữa chúng.

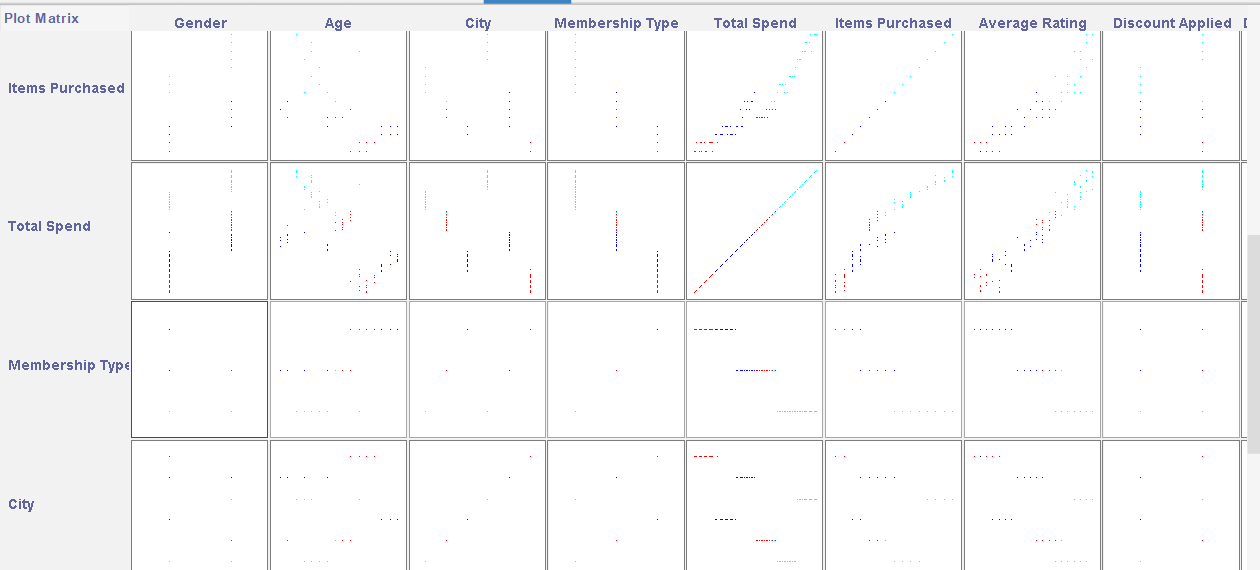
* **Thống kê mô tả:** Dữ liệu được đưa vào khai phá gồm **9 thuộc tính**, **350 mẫu dữ liệu** trên **weka**

| **STT** | **Tên thuộc tính** | **Ý nghĩa** |
| --- | --- | --- |
| 1 | Gender | Giới tính |
| 2 | Age | Tuổi |
| 3 | City | Thành phố sinh sống |
| 4 | Membership\_Type | Loại thành viên |
| 5 | Total Spend | Tổng chi tiêu của khách hàng |
| 6 | Items Purchased | Tổng số mặt hàng đã mua |
| 7 | Average rating | Thể hiện xếp hạng trung bình mà khách hàng đưa ra cho các mặt hàng đã mua, đo lường mức độ hài lòng |
| 8 | Discount Applied | Khách hàng có áp dụng giảm giá hay không |
| 8 | Days Since Last Purchase | Số ngày trôi qua kể từ lần mua hàng gần nhất |
| 9 | Satisfaction Level | Mức độ hài lòng của khách hàng |

* Mối tương quan của dữ liệu



*Hình 14a: Mối tương quan của một số thuộc tính với nhau*



*Hình 14b: Mối tương quan của một số thuộc tính với nhau*

# KHAI PHÁ LUẬT KẾT HỢP

## Khai phá luật kết hợp

* Khai phá luật kết hợp là một lý thuyết trong lĩnh vực khai phá dữ liệu và học máy. Nó được sử dụng để tìm kiếm và khám phá các quy tắc, mẫu hoặc mối quan hệ tồn tại giữa các thuộc tính hoặc biến trong dữ liệu mà không cần dựa vào bất kỳ thông tin ngoại vi nào. Thay vào đó, lý thuyết này dựa trên kiến thức tiên phong (a priori knowledge) về dữ liệu để tìm ra những mẫu kết hợp (combination patterns) quan trọng.
* Trong khai phá luật kết hợp, đầu tiên, thuật toán xác định tần suất xuất hiện của các mẫu hoặc quy tắc trong dữ liệu. Sau đó, nó sử dụng các ngưỡng (thresholds) để chọn ra những mẫu quan trọng đáng chú ý. Ngưỡng này thường được đặt dựa trên một số tiêu chí như tần suất xuất hiện, độ tin cậy (confidence) và độ hỗ trợ (support) của mẫu.
* Công cụ phổ biến được sử dụng trong khai phá luật kết hợp là thuật toán Apriori. Thuật toán này sử dụng một phương pháp đệ quy để liệt kê toàn bộ không gian mẫu kết hợp có thể có và tìm ra các mẫu có độ hỗ trợ lớn hơn ngưỡng đã định. Nhờ vào kiến thức tiên phong về dữ liệu, thuật toán Apriori có thể giảm số lượng mẫu cần kiểm tra và tăng hiệu suất của quá trình khai phá.

## Quy trình thực hiện thuật giải Apriori

* **Chuẩn bị dữ liệu:** Tiền xử lý dữ liệu là một bước quan trọng trước khi áp dụng Apriori. Điều này bao gồm loại bỏ dữ liệu trùng lặp, xử lý các giá trị thiếu (nếu có) và chuyển đổi dữ liệu thành định dạng phù hợp cho việc áp dụng thuật toán Apriori.
* **Xác định ngưỡng:** Ngưỡng (threshold) được xác định để quyết định mức độ quan trọng của mẫu kết hợp. Ngưỡng có thể được đặt dựa trên tần suất xuất hiện của mẫu hoặc các yếu tố khác như độ tin cậy và độ hỗ trợ.
* **Tạo tập ứng viên (Candidate Generation):** Trong bước này, tập ứng viên chứa các mẫu kết hợp có kích thước từ 1 đến k. Ban đầu, tập ứng viên 1-đề cập đến các thuộc tính hoặc biến đơn lẻ. Sau đó, các tập ứng viên có kích thước lớn hơn được tạo ra từ các tập ứng viên có kích thước nhỏ hơn bằng cách kết hợp các mẫu kết hợp đã biết.
* **Đánh giá độ hỗ trợ (Support Evaluation):** Trong bước này, độ hỗ trợ của các ứng viên được tính toán. Độ hỗ trợ là tỷ lệ phần trăm của số lần xuất hiện của mẫu trong tập dữ liệu so với tổng số mẫu. Các ứng viên có độ hỗ trợ lớn hơn ngưỡng đã định sẽ được giữ lại.
* **Loại bỏ mẫu không quan trọng:** Ở bước này, các ứng viên không đáp ứng ngưỡng được loại bỏ. Điều này giúp giảm số lượng ứng viên cần xem xét và tăng hiệu suất của quá trình khai phá.
* **Lặp lại các bước trên:** Các bước 3-5 được lặp lại cho đến khi không còn ứng viên mới hoặc không còn mẫu đáp ứng ngưỡng mới được tạo ra. Kết quả là tập hợp các mẫu kết hợp quan trọng được tìm thấy trong dữ liệu.
* **Hiển thị và phân tích kết quả:** Cuối cùng, kết quả của quá trình khai phá được hiển thị và phân tích. Các mẫu kết hợp quan trọng có thể được sắp xếp theo độ hỗ trợ hoặc độ tin cậy để đưa ra thông tin hữu ích về mối quan hệ giữa các thuộc tính hoặc biến trong dữ liệu.

## Khai phá luật kết hợp bằng Apriori trong weka

* Chọn minsup = 0.33 và minconf = 0.9.
* Tập phổ biến thu được
* Số lượng tập phổ biến ở lần lặp 1: 17
* Số lượng tập phổ biến ở lần lặp 2: 18
* Số lượng tập phổ biến ở lần lặp 3: 5
* Tổng: 40 tập phổ biến
* Luật kết hợp

1. Age=Young adults Satisfaction Level=2 ==> Items Purchased=Many product <conf:(1)> lift:(1.92) lev:(0.16) [57] conv:(57.12)

* **Tuổi thanh niên và hài lòng thì tổng số sản phẩm từng mua nhiều.**

2. Total Spend=Low spending ==> Gender=Female

<conf:(1)> lift:(2) lev:(0.17) [58] conv:(58.5)

* **Tổng chi tiêu thấp thường là nữ giới.**

3. Total Spend=Average spending ==> Age=Young adults

<conf:(1)> lift:(1.5) lev:(0.11) [38] conv:(38.78)

* **Tổng chi tiêu trung bình thường là thanh niên.**

4. Total Spend=High spending ==> Items Purchased=Many product

<conf:(1)> lift:(1.92) lev:(0.16) [55] conv:(55.68)

* **Tổng chi tiêu cao thì tổng số sản phẩm từng mua nhiều.**

5. Total Spend=Low spending Average Rating=Low rating ==> Gender=Female <conf:(1)> lift:(2) lev:(0.17) [57] conv:(58)

* **Tổng chi tiêu thấp và đánh giá thấp thường là nữ giới.**

6. Gender=Female Average Rating=Low rating ==> Total Spend=Low spending <conf:(1)> lift:(2.99) lev:(0.22) [77] conv:(77.22)

* **Nữ giới và đánh giá thấp thường có tổng chi tiêu thấp.**

7. Gender=Male Total Spend=Average spending ==> Age=Young adults <conf:(1)> lift:(1.5) lev:(0.11) [38] conv:(38.45)

* **Nam giới và tổng chi tiêu trung bình thường là thanh niên.**

8. Items Purchased=Many product ==> Age=Young adults

<conf:(0.99)> lift:(1.49) lev:(0.17) [59] conv:(30.16)

* **Tổng số sản phẩm từng mua nhiều thường là thanh niên.**

9. Gender=Male ==> Age=Young adults

<conf:(0.99)> lift:(1.49) lev:(0.16) [57] conv:(29)

* **Nam giới thường là thanh niên.**

10. Gender=Male Items Purchased=Many product ==> Age=Young adults <conf:(0.99)> lift:(1.48) lev:(0.11) [40] conv:(20.55)

* **Nam giới và tổng số sản phẩm từng mua nhiều thường là thanh niên.**

11. Satisfaction Level=2 ==> Items Purchased=Many product

<conf:(0.99)> lift:(1.91) lev:(0.16) [57] conv:(29.04)

* **Hài lòng thường tổng sản phẩm đã mua nhiều.**

12. Items Purchased=Many product Satisfaction Level=2 ==> Age=Young adults <conf:(0.99)> lift:(1.48) lev:(0.11) [38] conv:(19.89)

* **Tổng sản phẩm đã mua nhiều và hài lòng thường là thanh niên.**

13. Gender=Male Days Since Last Purchase=Recent purchase ==> Age=Young adults

<conf:(0.99)> lift:(1.48) lev:(0.11) [38] conv:(19.55)

* **Nam giới và mua hàng gần đây thường là thanh niên.**

14. Total Spend=Average spending ==> Gender=Male

<conf:(0.99)> lift:(1.98) lev:(0.16) [57] conv:(29.25)

* **Tổng chi tiêu trung bình thường là nữ giới.**

15. Total Spend=Low spending ==> Average Rating=Low rating

<conf:(0.99)> lift:(2.33) lev:(0.19) [66] conv:(33.6)

* **Tổng chi tiêu thấp thường đánh giá thấp.**

16. Gender=Female Total Spend=Low spending ==> Average Rating=Low rating <conf:(0.99)> lift:(2.33) lev:(0.19) [66] conv:(33.6)

* **Nữ giới và tổng chi tiêu thấp thường đánh giá thấp.**

17. Total Spend=Low spending ==> Gender=Female Average Rating=Low rating <conf:(0.99)> lift:(2.99) lev:(0.22) [77] conv:(39.11)

* **Tổng chi tiêu thấp thường là nam giới đánh giá thấp.**

18. Age=Young adults Total Spend=Average spending ==> Gender=Male <conf:(0.99)> lift:(1.98) lev:(0.16) [57] conv:(29.25)

* **Thanh niên và tổng chi tiêu trung bình thường là nam giới.**

19. Total Spend=Average spending ==> Gender=Male Age=Young adults <conf:(0.99)> lift:(1.99) lev:(0.17) [57] conv:(29.42)

* **Tổng chi tiêu trung bình thường là nam thanh niên.**

20. Satisfaction Level=2 ==> Age=Young adults

<conf:(0.98)> lift:(1.47) lev:(0.11) [38] conv:(13.37)

* **Hài lòng thường là thanh niên.**

21. Satisfaction Level=2 ==> Age=Young adults Items Purchased=Many product <conf:(0.98)> lift:(1.9) lev:(0.16) [56] conv:(19.48)

* **Hài lòng thường là thanh nhiên và tổng sản phẩm từng mua nhiều.**

22. Membership Type=Silver ==> Age=Young adults

<conf:(0.96)> lift:(1.43) lev:(0.1) [35] conv:(6.79)

* **Thành viên Bạc thường là thanh niên.**

## Khai phá luật kết hợp bằng Apriori trong python

**Class Apriori và Association Rule**

import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from itertools import combinations

class AprioriAlgorithm\_AssociationRule:

def \_\_init\_\_(self, min\_supp, min\_conf):

self.min\_supp = min\_supp

self.min\_conf = min\_conf

self.C = []

self.L = []

self.TPB = []

self.TPB\_ = []

def get\_unique\_values(self, array):

unique\_values = []

for row in array:

for value in row:

if value not in unique\_values:

unique\_values.append(value)

return unique\_values

def tp\_list(self, C\_list):

List = []

for i in C\_list:

List.append(list(i))

return List

def get\_C(self, L, k):

C = []

if k == 2:

C = list(combinations(L, k))

C = self.tp\_list(C)

else:

C\_test = []

for i in range(len(L)):

for j in range(i + 1, len(L)):

if L[i][:-1] == L[j][:-1]:

item = sorted(list(set(L[i]) | set(L[j])))

C\_test.append(item)

count = 0

for item in C\_test:

combinations\_list = list(combinations(item, len(item) - 1))

combinations\_list = self.tp\_list(combinations\_list)

for i in combinations\_list:

if i in L:

count += 1

if count == len(combinations\_list):

C.append(item)

count = 0

return C

def set\_L(self, C, df, k):

L = []

for item in C:

if k == 1:

Supp\_L = df[item].mean()

else:

Supp\_L = df[item].all(axis=1).mean()

if Supp\_L >= self.min\_supp:

L.append(item)

self.TPB.append(item)

if k > 1:

self.TPB\_.append(item)

return L

def generate\_rules(self, frequent\_itemsets):

rules = []

for itemset in frequent\_itemsets:

for i in range(1, len(itemset)):

subsets = list(combinations(itemset, r=i))

for subset in subsets:

antecedent = subset

consequent = tuple(set(itemset) - set(subset))

confidence = self.support(itemset) / self.support(antecedent)

if confidence >= self.min\_conf:

rules.append((antecedent, consequent, confidence))

return rules

def support(self, itemset):

count = 0

for transaction in self.transactions:

if set(itemset).issubset(set(transaction)):

count += 1

return count

def find\_rules(self, data):

self.data = data

self.transactions = data.values.tolist()

self.transactions = [[str(item) for item in transaction] for transaction in self.transactions]

te = TransactionEncoder()

te\_ary = te.fit(self.transactions).transform(self.transactions)

self.df = pd.DataFrame(te\_ary, columns=te.columns\_)

self.C = self.get\_unique\_values(self.transactions)

self.C.sort()

k = 1

while True:

self.L = self.set\_L(self.C, self.df, k)

if len(self.L) <= k:

break

print("Size of set of large itemsets L(", k, "): ", len(self.L))

k += 1

self.C = self.get\_C(self.L, k)

self.LKH = self.generate\_rules(self.TPB\_)

def get\_frequent\_itemsets(self):

return self.TPB

def get\_association\_rules(self):

return self.LKH

**Lấy data từ weka, chọn minsupp = 0.33 và minconf = 0.9**

data = pd.read\_csv('E-commerce Customer Behavior weka.csv')

min\_supp = 0.33

min\_conf = 0.9

apriori = AprioriAlgorithm\_AssociationRule(min\_supp, min\_conf)

apriori.find\_rules(data)

frequent\_itemsets = apriori.get\_frequent\_itemsets()

association\_rules = apriori.get\_association\_rules()

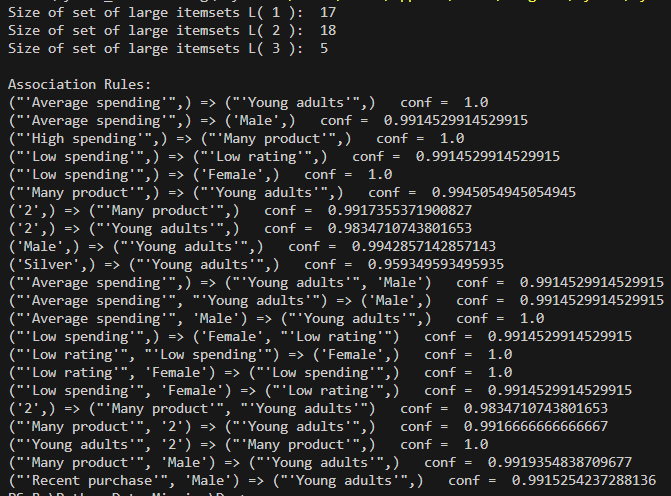
# print("\nFrequent Itemsets:\n", frequent\_itemsets)

print("\nAssociation Rules:")

for rule in association\_rules:

print(rule[0], "=>", rule[1]," conf = ", rule[2])

* Kết quả



*Hình 15: Kết quả khai phá luật kết hợp trong Python*

=> Nhận thấy: Kết quả giống weka nên chúng ta bỏ bước biểu diễn tri thức

## E. Nhận xét đánh giá

Phần khai phá luật kết hợp Apriori trong bài toán "Khai phá dữ liệu bằng phương pháp phân lớp để phân loại mức độ hài lòng của khách hàng trên sàn thương mại điện tử" có một số hạn chế và điểm yếu cần được cải thiện.

Mặc dù thuật toán Apriori đã được áp dụng thành công để tạo ra các quy tắc liên kết từ dữ liệu khách hàng, nhưng độ chính xác và độ tin cậy của các quy tắc này có thể còn hạn chế. Có thể có các quy tắc không quan trọng hoặc không có ý nghĩa thực tế mà vẫn được tạo ra, dẫn đến sự mất mát và nhiễu trong kết quả khai phá.

Ngoài ra, việc sử dụng Apriori để xác định các quy tắc liên kết và sau đó áp dụng một phương pháp phân loại khác có thể dẫn đến mất mát thông tin quan trọng. Có thể có một sự không phù hợp giữa quy tắc liên kết và phương pháp phân loại được sử dụng, làm giảm hiệu suất của mô hình phân loại và chất lượng dự đoán.

Hơn nữa, việc áp dụng kết quả khai phá vào thực tế cũng có thể gặp khó khăn. Cần xem xét xem liệu các quy tắc liên kết tìm thấy có thể được áp dụng và thực hiện trong môi trường thương mại điện tử thực tế hay không. Các yếu tố như tính khả thi, tính thực tế và tính áp dụng của kết quả khai phá cần được đánh giá một cách cẩn thận.

Tóm lại, mặc dù phần khai phá luật kết hợp Apriori đã mang lại một số kết quả và nhận thức về mô hình mua hàng và mức độ hài lòng của khách hàng trên sàn thương mại điện tử, nhưng cần chú ý đến các hạn chế và điểm yếu của phương pháp này.

# PHƯƠNG PHÁP PHÂN LỚP - NAIVE BAYES

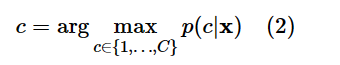
## Thuật toán Naive Bayes

* Xét bài toán classification với C classes 1, 2,…,C. Giả sử có một điểm x∈Rd .Hãy tính xác suất để điểm dữ liệu này rơi vào class c. Nói cách khác, hãy tính:

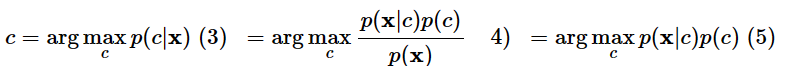


hay viết gọn lại thành p(c|x)

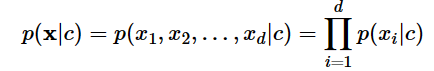
* Tức tính xác suất để đầu ra là class c biết rằng đầu vào vector x
* Biểu thức này, nếu tính được, sẽ giúp chúng ta xác định được xác suất để điểm dữ liệu rơi vào mỗi class. Từ đó có thể giúp xác định class của điểm dữ liệu đó bằng cách chọn ra class có xác suất cao nhất:



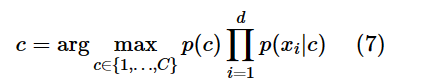
* Biểu thức (2) thường khó được tính trực tiếp. Thay vào đó, quy tắc Bayes thường được sử dụng:



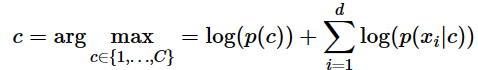
* Từ (3) sang (4) là vì quy tắc Bayes. Từ (4) sang (5) là vì mẫu số p(x) không phụ thuộc vào c
* Thành phần còn lại p(x|c), tức phân phối của các điểm dữ liệu trong class c, thường rất khó tính toán vì x là một biến ngẫu nhiên nhiều chiều, cần rất rất nhiều dữ liệu training để có thể xây dựng được phân phối đó. Để giúp cho việc tính toán được đơn giản, người ta thường giả sử một cách đơn giản nhất rằng các thành phần của biến ngẫu nhiên x là [độc lập với nhau](https://machinelearningcoban.com/2017/07/09/prob/#-independence), nếu biết c. Tức là:



* Ở bước **training**, các phân phối p(c) và p(xi|c), i = 1, 2,..., d sẽ được xác định dựa vào training data
* Ở bước **test**, với một điểm dữ liệu mới x class của nó sẽ được xác định bởi:



* Khi d lớn và các xác suất nhỏ, biểu thức ở vế phải sẽ là một số rất nhỏ, khi tính toán có thể gặp sai số. Để giải quyết việc này, biểu thức thường được viết lại dưới dạng tương đương bằng cách lấy log của vế phải:

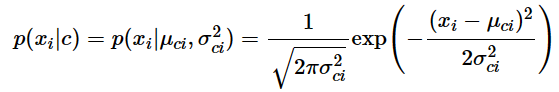


* Việc tính toán p(xi|c) phụ thuộc vào loại dữ liệu. Có ba loại phổ biến là: Gaussian Naive Bayes, Multinomial Naive Bayes, và Bernoulli Naive.

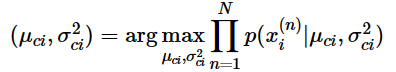
## B. Các loại phân phối thường dùng

### Gaussian Naive Bayes

* Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục.
* Với mỗi chiều dữ liệu i và 1 class c, xi tuân theo một phân phối chuẩn có kỳ vọng μci và phương sai σci2:

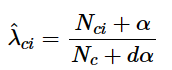


* Trong đó, bộ tham số θ = {μci, σci2} được xác định bằng Maximum Likelihood:



### Multinomial Naive Bayes

* Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of Words. Lúc này, 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, p(xi|c) tỉ lệ với tần suất từ thứ i (hay feature thứ i cho trường hợp tổng quát) xuất hiện trong các văn bản của class c. Giá trị này có thể được tính bằng cách:
* Trong đó:
* Nci: Tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c
* Nc: là tổng số từ (kể cả lặp) xuất hiện trong class c. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c. Có thể suy ra rằng: 
* Cách tính này có một hạn chế là nếu có một từ mới chưa bao giờ xuất hiện trong class c thì biểu thức (10) sẽ bằng 0 và vế phải của biểu thức (7) sẽ bằng 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:



* Với α là một số dương, thường bằng 1, để tránh trường hợp tử số bằng 0. Mẫu số được cộng với dα để đảm bảo tổng xác suất 
* Như vậy, mỗi class c sẽ được mô tả bởi bộ các số dương có tổng bằng 1: 

### Bernoulli Naive Bayes

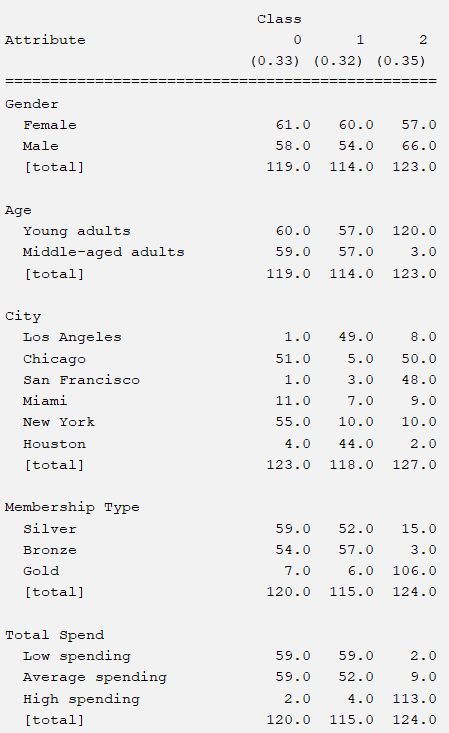
* Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị binary - bằng 0 hoặc 1. Ví dụ: cùng với loại văn bản nhưng thay vì đếm tổng số lần xuất hiện của 1 từ trong văn bản, ta chỉ cần quan tâm từ đó có xuất hiện hay không.
* Khi đó, p(xi|c) được tính bằng:

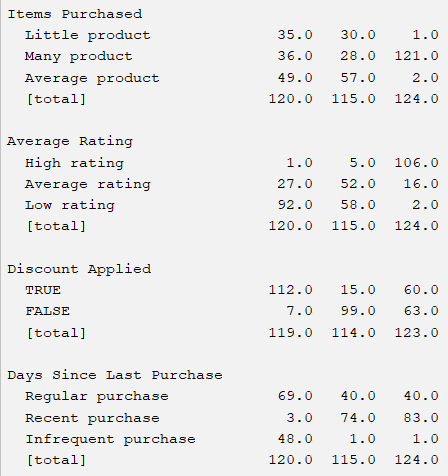


* Vì p(i|c) có thể được hiểu là xác suất từ thứ i xuất hiện trong các văn bản của class c

## C. Phân lớp Naive Bayes trong weka

* Kết quả

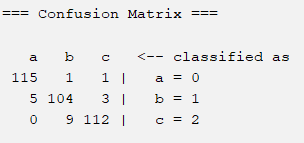




* Nhận xét những xác suất mẫu có độ ảnh hưởng cao:
* Mẫu Young adults (Age) thuộc lớp 2 (Satisfaction Level) có xác suất lên đến 97.6%
* Mẫu Gold (Membership) thuộc lớp 2 (Satisfaction Level) có xác suất lên đến 85.5%
* Mẫu High spending (Total Spend) thuộc lớp 2 (Satisfaction Level) có xác suất lên đến 91.1%
* Mẫu Many product (Items Purchased) thuộc lớp 2 (Satisfaction Level) có xác suất lên đến 97.6%
* Mẫu Low rating (Average Rating) thuộc lớp 0 (Satisfaction Level) có xác suất lên đến 76.7%
* Mẫu High rating (Average Rating) thuộc lớp 2 (Satisfaction Level) có xác suất lên đến 97.6%
* Mẫu High rating (Average Rating) thuộc lớp 2 (Satisfaction Level) có xác suất lên đến 85.5%
* Mẫu TRUE (Discount Applied) thuộc lớp 0 (Satisfaction Level) có xác suất lên đến 94.1%
* Mẫu FALSE (Discount Applied) thuộc lớp 1 (Satisfaction Level) có xác suất lên đến 86.8%
* Nhận xét con số đánh giá chất lượng mô hình:
* Tổng số mẫu được phân loại đúng là 331 trong tổng số 350 mẫu, tỷ lệ chính xác là 94.57%.
* Số mẫu bị phân loại sai là 19, chiếm tỷ lệ 5.43%.



* Nhận xét ma trận tương quan
* Lớp 0: Có tổng cộng 117 mẫu thuộc lớp 0. Trong số đó, 115 mẫu được phân loại đúng, 1 mẫu bị phân loại nhầm thành lớp 1 và 1 mẫu bị phân loại nhầm thành lớp 2.
* Lớp 1: Có tổng cộng 112 mẫu thuộc lớp 1. Trong số đó, 104 mẫu được phân loại đúng, 5 mẫu bị phân loại nhầm thành lớp 0 và 3 mẫu bị phân loại nhầm thành lớp 2.
* Lớp 2: Có tổng cộng 121 mẫu thuộc lớp 2. Trong số đó, 112 mẫu được phân loại đúng, 9 mẫu bị phân loại nhầm thành lớp 0 và không có mẫu nào bị phân loại nhầm thành lớp 1.



## D. Phân lớp Naive Bayes trong python

**Thư viện hỗ trợ thuật toán Naive Bayes**

* import pandas as pd
* import numpy as np
* from collections import defaultdict
* from tabulate import tabulate

**Class Naive Bayes**

class NaiveBayesClassifier:

def \_\_init\_\_(self):

# Phương thức khởi tạo được gọi khi một đối tượng mới của lớp được tạo ra

# Khởi tạo biến thành viên class\_probabilities với giá trị None

self.class\_probabilities = None

# Khởi tạo biến thành viên feature\_probabilities với giá trị None

self.feature\_probabilities = None

def fit(self, X, y):

# Lấy số lượng mẫu và số lượng đặc trưng từ dữ liệu đầu vào

n\_samples, n\_features = X.shape

# Xác định các lớp duy nhất trong tập dữ liệu và số lượng lớp

self.classes = np.unique(y)

n\_classes = len(self.classes)

# Khởi tạo một từ điển để lưu trữ xác suất cho từng lớp

self.class\_probabilities = defaultdict(float)

# Khởi tạo một từ điển lồng để lưu trữ xác suất của từng thuộc tính cho mỗi lớp

self.feature\_probabilities = defaultdict(lambda: defaultdict(float))

# Tính xác suất của từng lớp

for c in self.classes:

# Tạo mặt nạ để chọn các mẫu thuộc lớp c hiện tại

mask = (y == c)

# Tính xác suất của lớp c sử dụng Laplace Smoothing

self.class\_probabilities[c] = (np.sum(mask) + 1) / (n\_samples + n\_classes) # Laplace Smoothing

# Tính xác suất của từng thuộc tính cho mỗi lớp

for i in range(n\_features):

# Lấy các giá trị duy nhất của đặc trưng thứ i

feature\_values = np.unique(X[:, i])

# Tính xác suất của giá trị feature cho lớp c

for value in feature\_values:

# Tạo mặt nạ để chọn các mẫu có giá trị feature là value

feature\_mask = (X[:, i] == value)

# Tính xác suất của giá trị feature là value cho lớp c sử dụng Laplace Smoothing

self.feature\_probabilities[c][i, value] = (np.sum(mask & feature\_mask) + 1) / (np.sum(mask) + len(feature\_values)) # Laplace Smoothing

def predict(self, X):

# Kiểm tra xem mô hình đã được huấn luyện chưa

if self.class\_probabilities is None or self.feature\_probabilities is None:

raise RuntimeError("Classifier has not been trained.")

# Lấy số lượng mẫu và số lượng đặc trưng từ dữ liệu đầu vào

n\_samples, n\_features = X.shape

predictions = []

# Dự đoán cho từng mẫu

for i in range(n\_samples):

# Lấy mẫu thứ i

sample = X[i, :]

# Tạo một từ điển để lưu trữ điểm số cho từng lớp

class\_scores = defaultdict(float)

# Tính điểm số cho từng lớp

for c in self.classes:

# Điểm số cho lớp c bằng log của xác suất lớp c

class\_scores[c] = np.log(self.class\_probabilities[c])

# Tính điểm số cho từng thuộc tính của mẫu

for j in range(n\_features):

feature\_value = sample[j]

# Cộng thêm điểm số của giá trị đặc trưng vào điểm số của lớp c

class\_scores[c] += np.log(self.feature\_probabilities[c][j, feature\_value])

# Chọn lớp có điểm số cao nhất là kết quả dự đoán cho mẫu thứ i

predicted\_class = max(class\_scores, key=class\_scores.get)

predictions.append(predicted\_class)

return predictions

# Trả về xác suất của các đặc trưng cho từng lớp sau khi mô hình đã được huấn luyện.

def get\_feature\_probabilities(self):

if self.feature\_probabilities is None:

raise RuntimeError("Classifier has not been trained.")

return self.feature\_probabilities

# Đánh giá hiệu suất của mô hình dự đoán.

def evaluate(self, X, y):

y\_pred = self.predict(X)

# Tính toán các chỉ số đánh giá

n\_samples = len(y)

correct = np.sum(y\_pred == y)

incorrect = n\_samples - correct

accuracy = correct / n\_samples

kappa = (accuracy - (1 / len(self.classes))) / (1 - (1 / len(self.classes)))

# Tính toán ma trận nhầm lẫn

confusion\_matrix = np.zeros((len(self.classes), len(self.classes)), dtype=int)

for true\_class, pred\_class in zip(y, y\_pred):

confusion\_matrix[true\_class][pred\_class] += 1

return accuracy, incorrect, kappa, confusion\_matrix

**Chạy chương trình**

# Đọc dữ liệu từ file CSV

def read\_data(file\_path):

data = pd.read\_csv(file\_path)

X = data.drop('Satisfaction Level', axis=1).values

y = data['Satisfaction Level'].values

feature\_names = list(data.columns)[:-1]

return X, y, feature\_names

# Chương trình chính

def main():

# Đường dẫn đến file CSV

file\_path = 'E-commerce Customer Behavior weka.csv'

# Đọc dữ liệu từ file CSV

X, y, feature\_names = read\_data(file\_path)

# Tạo mô hình Naive Bayes và huấn luyện

nbc = NaiveBayesClassifier()

nbc.fit(X, y)

# Dự đoán nhãn cho dữ liệu huấn luyện và đánh giá mô hình

accuracy, incorrect, kappa, confusion\_matrix = nbc.evaluate(X, y)

# Hiển thị kết quả với tên cột

for c, probabilities in nbc.get\_feature\_probabilities().items():

print("Class:", c)

data = []

for (feature\_index, feature\_value), probability in probabilities.items():

data.append([feature\_names[feature\_index], feature\_value, probability])

print(tabulate(data, headers=['Feature', 'Value', 'Probability'], tablefmt='orgtbl'))

# Tính các chỉ số đánh giá bổ sung

mean\_absolute\_error = np.mean(np.abs(nbc.predict(X) - y))

root\_mean\_squared\_error = np.sqrt(np.mean((nbc.predict(X) - y) \*\* 2))

relative\_absolute\_error = mean\_absolute\_error / np.mean(np.abs(y))

root\_relative\_squared\_error = root\_mean\_squared\_error / np.mean(np.abs(y))

# Hiển thị kết quả đánh giá

print("=== Stratified cross-validation ===")

print("=== Summary ===")

print("Correctly Classified Instances ", accuracy \* len(y), " ", accuracy \* 100, "%")

print("Incorrectly Classified Instances ", incorrect, " ", (incorrect / len(y)) \* 100, "%")

print("Kappa statistic ", kappa)

print("Mean absolute error ", mean\_absolute\_error)

print("Root mean squared error ", root\_mean\_squared\_error)

print("Relative absolute error ", relative\_absolute\_error \* 100, "%")

print("Root relative squared error ", root\_relative\_squared\_error \* 100, "%")

print("Total Number of Instances ", len(y))

print("\n=== Detailed Accuracy By Class ===\n")

detailed\_accuracy = pd.DataFrame({

'Class': np.arange(len(nbc.classes)),

'TP Rate': np.diag(confusion\_matrix) / np.sum(confusion\_matrix, axis=1),

'FP Rate': (np.sum(confusion\_matrix, axis=0) - np.diag(confusion\_matrix)) / np.sum(confusion\_matrix, axis=1),

'Precision': np.diag(confusion\_matrix) / np.sum(confusion\_matrix, axis=0),

'Recall': np.diag(confusion\_matrix) / np.sum(confusion\_matrix, axis=1),

'F-Measure': 2 \* (np.diag(confusion\_matrix) / np.sum(confusion\_matrix, axis=0) \* np.diag(confusion\_matrix) / np.sum(confusion\_matrix, axis=1)) / (np.diag(confusion\_matrix) / np.sum(confusion\_matrix, axis=0) + np.diag(confusion\_matrix) / np.sum(confusion\_matrix, axis=1)),

'MCC': ((np.diag(confusion\_matrix) \* np.sum(confusion\_matrix, axis=1) - np.sum(confusion\_matrix, axis=0) \* np.sum(confusion\_matrix, axis=0)) /

np.sqrt((np.sum(confusion\_matrix, axis=0) + np.sum(confusion\_matrix, axis=1)) \*

np.sum(confusion\_matrix, axis=0) \* np.sum(confusion\_matrix, axis=1)))

})

print(tabulate(detailed\_accuracy, headers='keys', tablefmt='orgtbl'))

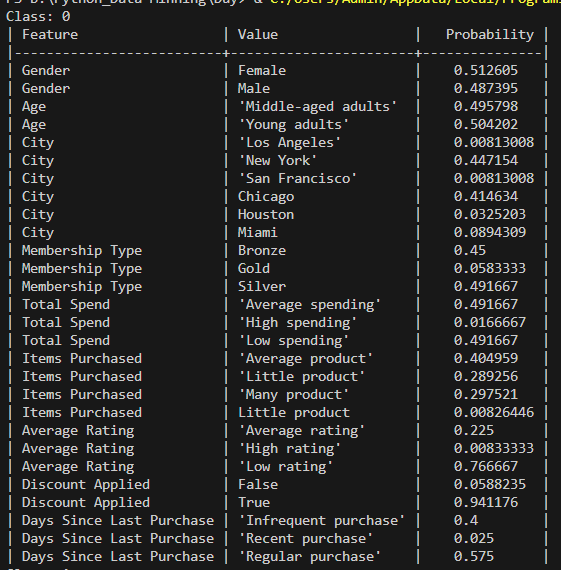
print("\n=== Confusion Matrix ===\n")

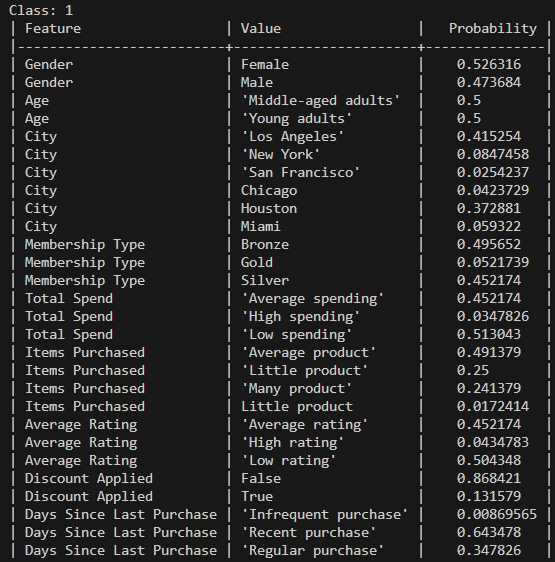
print(tabulate(confusion\_matrix, headers=['a', 'b', 'c'], tablefmt='orgtbl'))

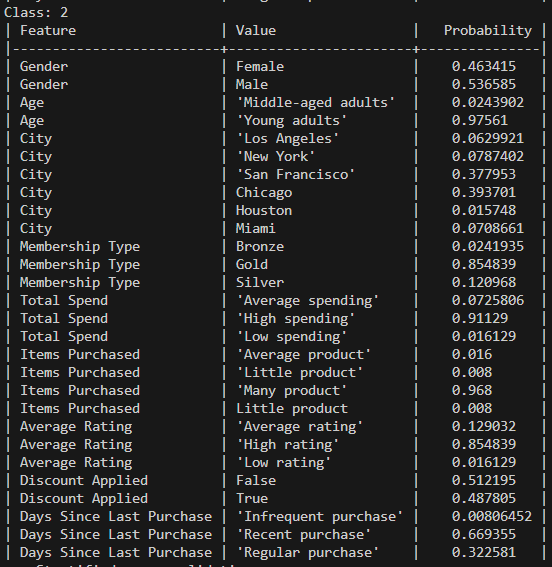
if \_\_name\_\_ == '\_\_main\_\_':

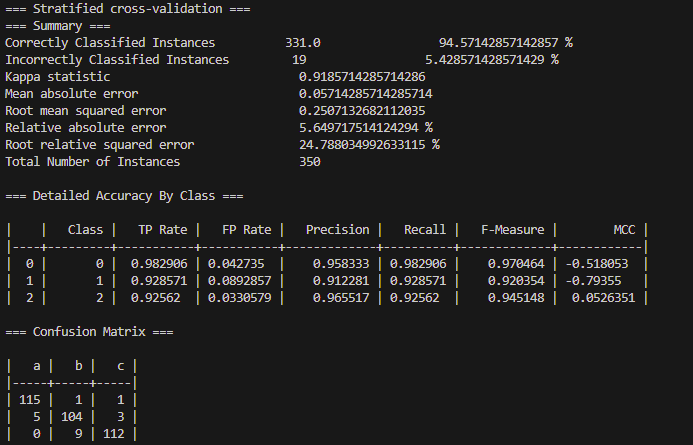
main()

**Kết quả**









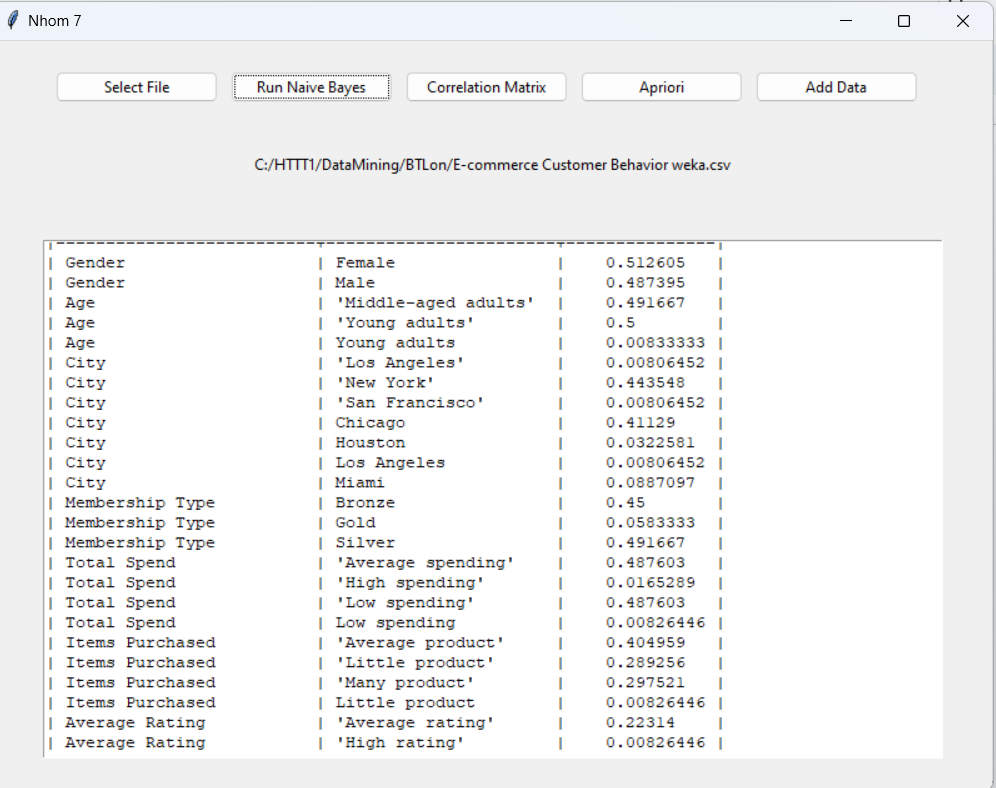
## 

## E. Biểu diễn tri thức

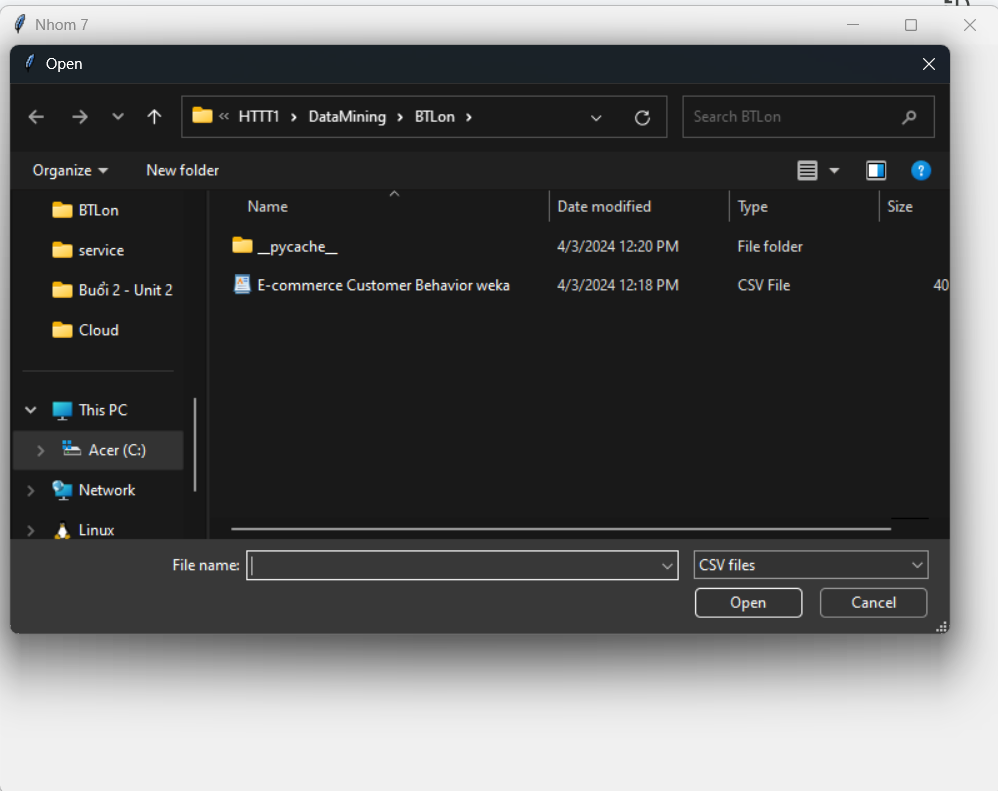
* Thanh niên thường hài lòng (97.6%)
* Thành viên hạng Gold thường hài lòng (85.5%)
* Tổng chi tiêu cao thường hài lòng (91.1%)
* Tổng sản phẩm đã mua nhiều thường hài lòng (97.6%)
* Đánh giá thấp thường không hài lòng (76.7%)
* Đánh giá cao thường hài lòng (97.6%)
* Áp dụng giảm giá thường không hài lòng (94.1%)
* Không áp dụng giảm giá thì cảm thấy bình thường (86.8%)

## F. Sản phẩm Demo

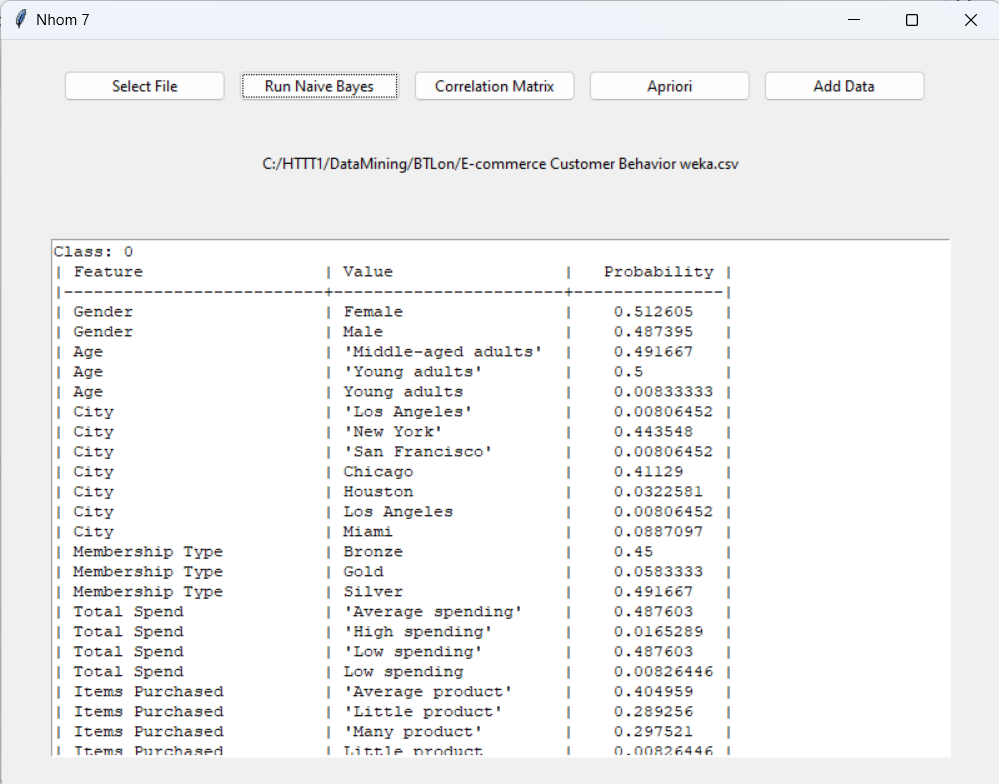
* Giao diện sản phẩm:



* Các chức năng:
* Select file:



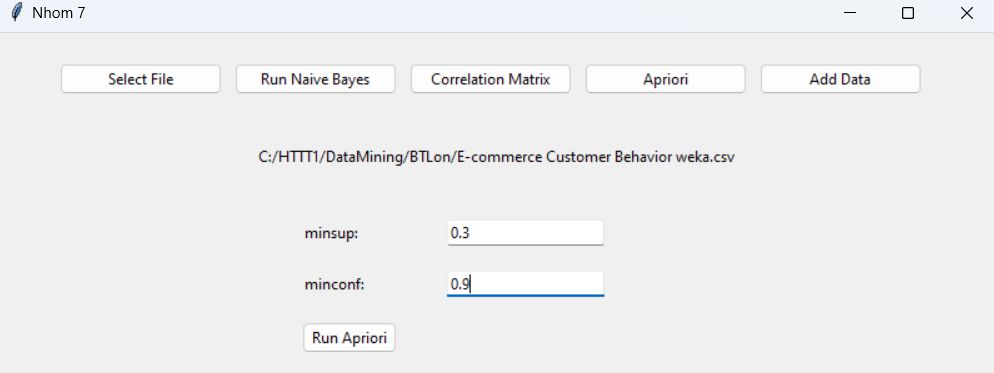
* Chạy thuật toán Naive Bayes:

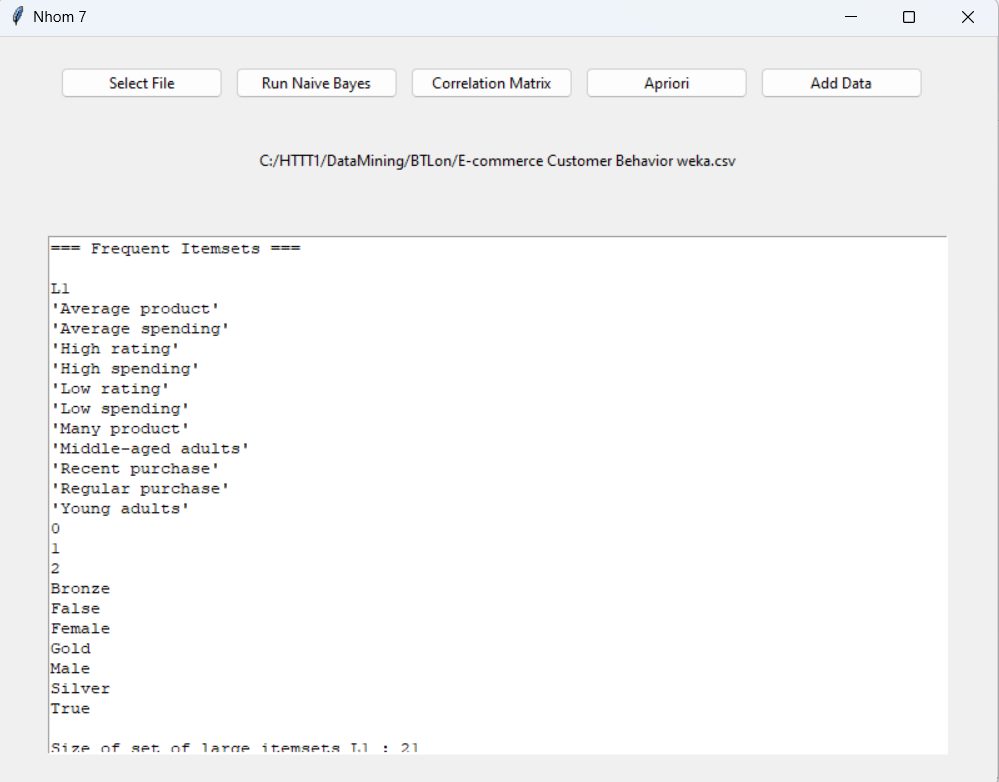


* Vẽ đồ thị tương quan:

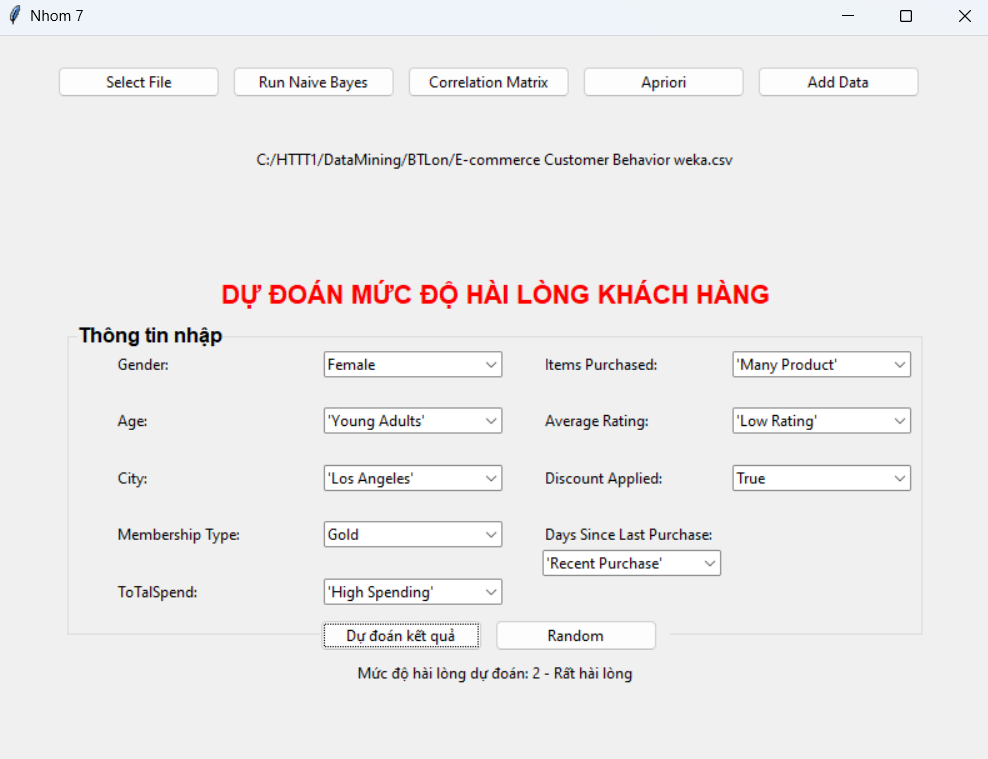


* Chạy thuật toán Apriori:





* Thêm dữ liệu để dự đoán:



# KẾT LUẬN

Hiện nay, mức độ hài lòng của khách hàng trên sàn thương mại điện tử đóng vai trò quan trọng trong việc đánh giá và định hình thành công của một doanh nghiệp. Với sự phát triển nhanh chóng của công nghệ và sự phổ biến của mua sắm trực tuyến, khách hàng có nhiều lựa chọn hơn bao giờ hết và đánh giá chất lượng dịch vụ trở thành một yếu tố quan trọng trong quyết định mua hàng.

Trong bối cảnh này, việc áp dụng phân lớp dữ liệu và luật kết hợp đóng vai trò quan trọng để phân loại và dự đoán mức độ hài lòng của khách hàng trên sàn thương mại điện tử. Phân lớp dữ liệu là một phương pháp mạnh mẽ cho phép chúng ta xây dựng mô hình dự đoán dựa trên các thuộc tính khác nhau. Đồng thời, việc áp dụng luật kết hợp giúp chúng ta tìm ra các quy tắc và mối quan hệ giữa các thuộc tính để phân loại khách hàng vào các nhóm mức độ hài lòng khác nhau. Việc sử dụng cả phân lớp và luật kết hợp mang lại khả năng dự đoán và hiểu rõ hơn về mức độ hài lòng của khách hàng trên sàn thương mại điện tử.

Tổng kết, việc khai phá dữ liệu bằng phương pháp phân lớp và áp dụng luật kết hợp đóng vai trò quan trọng trong việc phân loại mức độ hài lòng của khách hàng trên sàn thương mại điện tử. Nhờ vào việc xác định được các yếu tố ảnh hưởng và quy tắc phân loại, doanh nghiệp có thể cải thiện trải nghiệm mua sắm và đáp ứng mong đợi của khách hàng một cách tốt hơn. Việc nắm vững thông tin về khách hàng và áp dụng phân lớp cũng như luật kết hợp sẽ giúp doanh nghiệp xây dựng một hệ thống quản lý chất lượng hiệu quả, từ đó tạo điều kiện để tăng cường sự hài lòng và tín nhiệm của khách hàng trong lĩnh vực thương mại điện tử.

# TÀI LIỆU THAM KHẢO

* Bài giảng Khai phá dữ liệu - thầy **Trần Mạnh Tuấn**
* <https://viblo.asia/p/ml-from-scratch-thuat-toan-phan-loai-naive-bayes-viblo-aNj4vXOqL6r>
* <https://trituenhantao.io/kien-thuc/phan-1-phan-loai-naive-bayes-ly-thuyet/>
* <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html>
* [Machine Learning cơ bản (machinelearningcoban.com)](https://machinelearningcoban.com/2017/08/08/nbc/)

# 