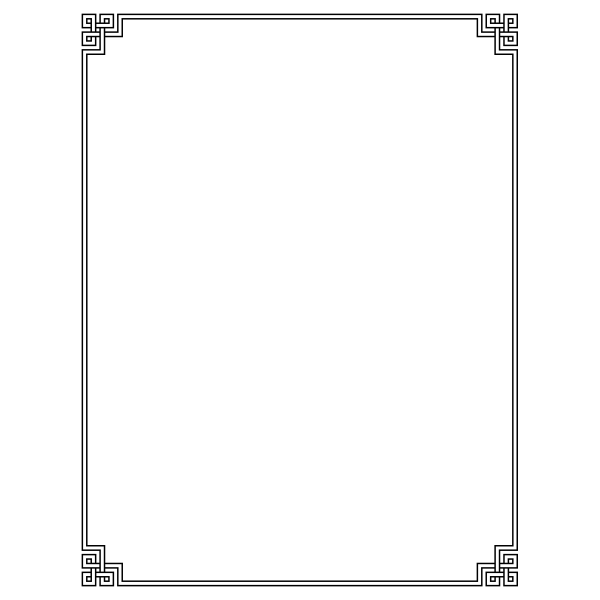
**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN TÍNH TOÁN MỀM**

**ĐỀ TÀI:**

**ỨNG DỤNG MẠNG NƠ RON MỜ TRONG DỰ ĐOÁN KHÁCH HÀNG TRUNG THÀNH DỰA TRÊN ĐẶC ĐIỂM NHÂN KHẨU HỌC VÀ HÀNH VI**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:**  **Sinh viên:**  **Lớp:** | TS. Trần Mạnh Tuấn  Nguyễn Thị Phương Anh  Nguyễn Thị Thanh Hải  64HTTT1 |

Hà Nội, 2025

LỜI NÓI ĐẦU

Trong thời đại 4.0 hiện nay, thời đại của công nghệ hóa, hiện đại hóa, những công việc đời thường cũng dần được số hóa, chẳng hạn như như mua sắm cũng không nằm ngoài phạm vi này. Song, việc nền tảng thương mại điện tử phát triển chóng mặt lại đặt ra một bài toán lớn cho các doanh nghiệp. Với bối cảnh thị trường cạnh tranh ngày càng gay gắt, việc giữ chân khách hàng trung thành chính là yếu tố quyết định.

Để giải quyết bài toán đó, từ những số liệu thực tế và nhu cầu ứng dụng công nghệ vào thực tiễn, chúng em đã quyết định triển khai đề tài *“Ứng dụng mạng nơ ron mờ trong dự đoán khách hàng trung thành dựa trên đặc điểm nhân khẩu học và hành vi”,* với mục tiêu xây dựng một mô hình dự đoán có khả năng học hỏi và xử lý dữ liệu mờ, từ đó hỗ trợ doanh nghiệp nhận diện nhóm khách hàng tiềm năng và đưa ra các chiến lược giữ chân khách hàng phù hợp.

Mục tiêu chính của chúng em là áp dụng và kết hợp mạng nơ ron mờ – một phương pháp kết hợp mạng nơron và logic mờ – để phát hiện các yếu tố ảnh hưởng lớn đến khả năng khách hàng quay lại, đồng thời xây dựng mô hình có độ chính xác và khả năng tổng quát cao.

Chúng em tin rằng đề tài sẽ mang lại những giá trị thiết thực cho lĩnh vực thương mại điện tử, chăm sóc khách hàng và quản trị doanh nghiệp trong kỷ nguyên dữ liệu lớn.

Chúng em xin chân thành cảm ơn thầy Trần Mạnh Tuấn đã tận tình hướng dẫn và đồng hành cùng chúng em trong quá trình thực hiện đề tài *“Ứng dụng mạng nơ ron mờ trong dự đoán khách hàng trung thành dựa trên đặc điểm nhân khẩu học và hành vi”.*

Chúng em xin chân thành cảm ơn!

MỤC LỤC

[CHƯƠNG 1 GIỚI THIỆU BÀI TOÁN 1](#_heading=h.1fob9te)

[1.1 Đặt vấn đề 1](#_heading=h.3znysh7)

[1.2 Phương pháp thực hiện 1](#_heading=h.2et92p0)

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 3](#_heading=h.tyjcwt)

[2.1 Mạng nơ ron mờ 3](#_heading=h.lnxbz9)

[2.1.1 Lớp mờ hóa 4](#_heading=h.w5naf4h4k0h6)

[2.1.2 Lớp luật 5](#_heading=h.wskzv5x8zkfb)

[2.1.3 Lớp chuẩn hóa 6](#_heading=h.2uicpis4wvnl)

[2.1.4 Lớp giải mờ 7](#_heading=h.5yu87pxod8h5)

[2.1.5 Lớp tổng hợp 7](#_heading=h.wkx2gfj0sn6h)

[2.2 Mô hình mạng nơron mờ trong dự đoán khách hàng trung thành 7](#_heading=h.2srqnzuhiqtq)

[2.2.1 Bài toán đặt ra 7](#_heading=h.5fx0jx2gzjrj)

[2.2.2 Lý do chọn mạng nơron mờ 8](#_heading=h.4l9r74ycmfgc)

[2.3 Đánh giá kết quả mô hình 8](#_heading=h.35nkun2)

[2.3.1 Tính chính xác 8](#_heading=h.1ksv4uv)

[2.3.2 Ma trận nhầm lẫn 8](#_heading=h.44sinio)

[2.3.3 Precision và recall 9](#_heading=h.z337ya)

[CHƯƠNG 3 DỮ LIỆU VÀ TIỀN XỬ LÝ DỮ LIỆU 10](#_heading=h.4i7ojhp)

[3.1 Nguồn dữ liệu 10](#_heading=h.2xcytpi)

[3.2 Tiền xử lý dữ liệu 15](#_heading=h.3as4poj)

[3.2.1 Kết hợp dữ liệu từ các bảng rời rạc 15](#_heading=h.n7gvc98uemd8)

[3.2.2 Khám phá dữ liệu sau kết hợp 16](#_heading=h.z0adq0i3quch)

[3.2.3 Làm sạch dữ liệu 18](#_heading=h.h1257xubqp8t)

[3.2.4 Chuẩn hóa dữ liệu 24](#_heading=h.6tt2wcmfqiph)

[3.2.5 Phân chia dữ liệu 25](#_heading=h.y62unbf77p92)

[CHƯƠNG 4 XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH 27](#_heading=h.2w5ecyt)

[4.1 Thiết kế và xây dựng mạng nơron mờ (ANFIS) 27](#_heading=h.e2elyg3b2h53)

[4.1.1 Phân tích dữ liệu và thuộc tính đầu vào 27](#_heading=h.1baon6m)

[4.1.2 Lựa chọn hàm thuộc 28](#_heading=h.pnqmbhlibkyu)

[4.1.3 Xây dựng lớp mờ hóa 30](#_heading=h.bjq5w49lxqr1)

[4.1.4 Xây dựng tập luật mờ 32](#_heading=h.nqi0ov4ysj5b)

[4.1.5 Khởi tạo ANFIS 34](#_heading=h.98ac93q6tdgv)

[4.1.6 Chạy mô hình 37](#_heading=h.64pe5vg68gvb)

[4.2 Đánh giá và tinh chỉnh mô hình 38](#_heading=h.3xw7ihup82az)

[4.2.1 Kết quả thực nghiệm ban đầu 38](#_heading=h.exewp05unpmi)

[4.2.2 Tinh chỉnh mô hình 40](#_heading=h.1x8y3xy0cmuf)

[4.2.2.1 Giảm số lượng thuộc tính 40](#_heading=h.wkp32kq8t81)

[4.2.2.2 Kết quả sau khi tinh chỉnh và đánh giá 42](#_heading=h.jv55k1n9pvfg)

[4.2.2.3 Tăng cường mẫu và đánh giá 44](#_heading=h.bl7lcvuy9sws)

[4.2.2.4 Kết luận 47](#_heading=h.tberts37n5ob)

[CHƯƠNG 5 TRIỂN KHAI MÔ HÌNH 48](#_heading=h.279ka65)

[TÀI LIỆU THAM KHẢO 51](#_heading=h.1d96cc0)

[PHỤ LỤC 53](#_heading=h.3x8tuzt)

DANH MỤC CÁC HÌNH ẢNH

[Hình 3.1. Dữ liệu ban đầu - user\_info 10](#_heading=h.2bn6wsx)

[Hình 3.2. Dữ liệu ban đầu - user\_log 11](#_heading=h.5h5ou63r7me7)

[Hình 3.3. Dữ liệu ban đầu - train 11](#_heading=h.2bn6wsx)

[Hình 3.4. Dữ liệu ban đầu - test 12](#_heading=h.2bn6wsx)

[Hình 3.5. Code chuyển kiểu của cột datetime từ string -> datetime 15](#_heading=h.djz6ldshei5m)

[Hình 3.6. Code tạo bảng đếm các hành vi của người dùng với từng gian hàng theo thời gian 15](#_heading=h.r626yiv4fur3)

[Hình 3.7. Code tạo bảng data, chứa dữ liệu chính để xử lý bài toán 16](#_heading=h.u8p7wy58yttu)

[Hình 3.8. Code khám phá tổng quan dữ liệu 16](#_heading=h.fh0a36ajnzko)

[Hình 3.9. Bảng mô tả thông tin dữ liệu 17](#_heading=h.wfrpsqfx2o6t)

[Hình 3.10. Bảng mô tả 5 dòng đầu của dữ liệu 17](#_heading=h.5yggk9wy0ij7)

[Hình 3.11. Bảng mô tả tổng giá trị null theo từng cột của dữ liệu 18](#_heading=h.owojwrmfqsek)

[Hình 3.12. Code xóa các giá trị NaN trong dữ liệu 18](#_heading=h.toh2w8qvlxor)

[Hình 3.13. Code loại bỏ cột phone 18](#_heading=h.iofx1ckzh1xm)

[Hình 3.14. Phân phối dữ liệu cột age 20](#_heading=h.9ulxlifgw2rt)

[Hình 3.15. Code xử lý ngoại lai bằng phương pháp IQR cho cột click 22](#_heading=h.r9go5dswyuxd)

[Hình 3.16. Code thực hiện mã hóa các giá trị liên tục 24](#_heading=h.r31e5ayaniur)

[Hình 3.17. Bảng mô tả 5 dòng đầu dữ liệu sau quá trình làm sạch dữ liệu 24](#_heading=h.7jmyjilw2fa9)

[Hình 3.18. Code chuẩn hóa theo Min-Max Scaling 25](#_heading=h.5pvpz0n4mvc0)

[Hình 3.19. Kết quả chuẩn hóa 25](#_heading=h.yxiq3xrcl1kx)

[Hình 3.20. Code phân chia dữ liệu 26](#_heading=h.ibtvyh6cgp0s)

[Hình 4.1. Các biểu đồ trực quan hóa theo hàm Gauss 30](#_heading=h.2l5d6e8wxmi6)

[Hình 4.2. Lớp mờ hóa - hàm Gauss 31](#_heading=h.uyhna020gefy)

[Hình 4.3. Code xây dựng tập luật 33](#_heading=h.ihchyhn68dk2)

[Hình 4.4. Mô hình ANFIS 35](#_heading=h.p82oo3g4fs1s)

[Hình 4.5. Lớp chuẩn hóa 35](#_heading=h.hfybosqaag51)

[Hình 4.6. Lớp giải mờ 36](#_heading=h.e564dy7pi83t)

[Hình 4.7. Lớp tổng hợp 36](#_heading=h.dl3qwa4ebmpu)

[Hình 4.8. Khởi tạo mô hình 37](#_heading=h.k8ra1hbpsm2n)

[Hình 4.9. Chạy mô hình 37](#_heading=h.w41e0y414amh)

[Hình 4.10. Kết quả thực hiện 38](#_heading=h.tgagl453i9r3)

[Hình 4.11. Ma trận nhầm lẫn 39](#_heading=h.ps5fsch3ilhc)

[Hình 4.12. Đưa 6 thuộc tính vào RandomForest 40](#_heading=h.j2js2ltf4qai)

[Hình 4.13. Biểu đồ độ quan trọng của các thuộc tính 41](#_heading=h.pcupntn1nlia)

[Hình 4.14. Kết quả sau khi tinh chỉnh 43](#_heading=h.wde02wcxdjpq)

[Hình 4.15. Ma trận nhầm lẫn 43](#_heading=h.z120keljqm5m)

[Hình 4.16. Thêm sample 44](#_heading=h.i2tc6fm0yvis)

[Hình 4.17. Kết quả sau khi thực hiện over-sampling 44](#_heading=h.vhx6qhh7zk77)

[Hình 4.18. Kết quả sau khi thực hiện over-sampling 45](#_heading=h.w6nzmp4s1z8e)

[Hình 4.19. Kết quả sau khi thực hiện over-sampling 46](#_heading=h.9mum4d98sivh)

DANH MỤC BẢNG BIỂU

[Bảng 3.1. Bảng mô tả ý nghĩa các thuộc tính trong file user\_info 12](#_heading=h.qsh70q)

[Bảng 3.2. Bảng mô tả ý nghĩa các thuộc tính trong file user\_log 13](#_heading=h.oxki3u6k67ou)

[Bảng 3.3. Bảng mô tả ý nghĩa các thuộc tính trong file train 14](#_heading=h.5y6n10vq6gz1)

[Bảng 3.4. Bảng mô tả ý nghĩa các thuộc tính trong file test 14](#_heading=h.bga93husvv2f)

[Bảng 4.1 Bảng minh họa thuộc tính đầu vào 27](#_heading=h.rf1yldxivl1h)

[Bảng 4.2 Bảng so sánh các chiến lược hàm thuộc 29](#_heading=h.ajl6gvhivj7f)

# GIỚI THIỆU BÀI TOÁN

## Đặt vấn đề

Trong những năm gần đây, với sự phát triển nhanh chóng của thương mại điện tử và chuyển đổi số, nhu cầu nghiên cứu hành vi tiêu dùng và duy trì khách hàng trung thành trở nên ngày càng cấp thiết đối với doanh nghiệp. Khách hàng trung thành không chỉ giúp giảm chi phí marketing, duy trì doanh thu ổn định mà còn là nguồn lan tỏa thương hiệu hiệu quả thông qua các kênh truyền miệng và mua lại thường xuyên.

Tuy nhiên, việc nhận diện và dự đoán khách hàng trung thành không phải là bài toán đơn giản, bởi nó chịu ảnh hưởng từ nhiều yếu tố như độ tuổi, giới tính, thu nhập, lịch sử mua sắm, hành vi phản hồi, mức độ tương tác,... Các mô hình phân tích truyền thống gặp hạn chế trong việc xử lý các dữ liệu phi tuyến tính, không đầy đủ hoặc có tính mơ hồ. Do đó, cần thiết phải có các phương pháp học máy linh hoạt và có khả năng thích ứng cao.

Mạng nơ ron mờ (Fuzzy Neural Network - FNN), là sự kết hợp giữa mạng nơ ron nhân tạo và logic mờ, được xem là một trong những hướng tiếp cận hiệu quả trong bài toán dự đoán hành vi và phân loại khách hàng. Nghiên cứu của Wang và cộng sự cho thấy rằng FNN có khả năng mô hình hóa hành vi tiêu dùng và cải thiện độ chính xác trong dự đoán khách hàng trung thành lên đến 92% trong môi trường thương mại điện tử [1]. Tại Việt Nam, các nghiên cứu cũng chỉ ra rằng việc ứng dụng mô hình dự đoán dựa trên hành vi và nhân khẩu học giúp doanh nghiệp hiểu rõ hơn về nhu cầu khách hàng và nâng cao hiệu quả chiến lược tiếp thị [2].

## Phương pháp thực hiện

1. Thu thập dữ liệu đầu vào
2. Tiền xử lý dữ liệu
3. Lựa chọn mô hình: Sử dụng Mạng nơron mờ (Fuzzy Neural Network - FNN)
4. Huấn luyện mô hình: Huấn luyện mô hình dựa trên tập huấn luyện
5. Đánh giá mô hình: Đánh giá độ hiệu quả và kết quả của mô hình
6. Triển khai mô hình

**Dữ liệu đầu vào:**

Bộ dữ liệu được công bố trong DAZONE - Cuộc thi Phân Tích Dữ Liệu, dữ liệu được thu thập gồm hơn 50 triệu bản ghi, từ khoảng 5000 cửa hàng, hơn 200 nghìn khách hàng và hơn 1 triệu sản phẩm. Bộ dữ liệu là thông tin tổng hợp của khách hàng từ một sàn thương mại điện thử trong một khoảng thời gian cố định, bao gồm: Danh sách các gian hàng (merchants), thông tin người dùng, lịch sử hành vi người dùng với các gian hàng như xem sản phẩm, mua hàng,v.v… theo thời gian.

Nguồn dữ liệu: [DAZONE 2025](https://drive.google.com/drive/folders/1d0Ptgy3d1neQcG4TXu0wYcxRl1fTMVhR)

# CƠ SỞ LÝ THUYẾT

Bài toán dự đoán khách hàng trung thành có mục tiêu đầu ra là nhãn khách hàng có quay lại (Yes) hay không (No) trong vòng 6 tháng. Với những dữ liệu về hành vi người dùng và hành vi nhân khẩu học tương đối phức tạp trong việc rời rạc hóa, ở đây, ta ứng dụng mạng nơron mờ để giải quyết bài toán.

## Mạng nơ ron mờ

Mạng nơ ron mờ (Neuro-Fuzzy Network) là sự kết hợp của logic mờ (fuzzy logic) và mạng nơron nhân tạo (artificial neural network – ANN), vừa có khả năng xử lý tốt các dữ liệu thô, học theo cấu trúc của mạng nơ ron, vừa có khả năng suy luận qua các biến ngôn ngữ của logic mờ. Bên cạnh đó, việc kết hợp cả logic mờ và mạng nơron cũng giúp khắc phục nhược điểm của logic mờ là không có khả năng điều chỉnh với môi trường dữ liệu mới và nhược điểm không xử lý được các dữ liệu quá phức tạp của mạng nơron.

Hệ suy diễn mờ có 3 loại chính là: hệ suy diễn mờ Mamdani, hệ suy diễn mờ Takagi - Sugeno, hệ suy diễn mờ Tsukamoto [3]. Hệ suy diễn mờ đánh giá, định lượng các suy diễn giống như suy diễn của con người và có cách xử lý biến ngôn ngữ như ở con người. Tuy nhiên, trong hệ suy diễn mờ, các tham số là cố định nên quá trình suy diễn không mềm dẻo và chính xác, bởi hệ suy diễn mờ không có hoặc ít tập trung vào quá trình học.

Một trong những ứng dụng phổ biến nhất của mạng nơron mờ là dự báo, đặc biệt trong bài toán dự đoán khách hàng trung thành - có đầu vào là những thông tin định lượng và định tính như đặc điểm nhân khẩu học (tuổi, giới tính, nghề nghiệp, tình trạng hôn nhân) và hành vi tương tác của người dùng (số lần xem sản phẩm, hành động mua hàng, số lượng sản phẩm yêu thích, tần suất thêm vào giỏ hàng) thường không có yếu tố phân loại rõ ràng, tồn tại trong miền giá trị liên tục, mơ hồ và phụ thuộc vào ngữ cảnh. Ví dụ, một người 32 tuổi có thể vừa được xem là “trẻ” vừa là “trung niên” tùy theo cách phân loại. Một ví dụ khác về hành vi như hành vi “xem nhiều nhưng không mua” có thể biểu hiện sự do dự hoặc không quan tâm, khó lượng hóa bằng mô hình học máy thông thường.

Chính vì vậy, mạng nơron mờ phát huy ưu điểm trong việc xử lý dữ liệu không chắc chắn (thông qua logic mờ) và tự học từ dữ liệu lớn (thông qua khả năng học của mạng nơron). Mô hình này cho phép ánh xạ linh hoạt giữa các đầu vào mờ với đầu ra là xác suất hoặc khả năng quay lại mua hàng của khách hàng, từ đó hỗ trợ doanh nghiệp ra quyết định về chăm sóc, tiếp thị cá nhân hóa và giữ chân khách hàng hiệu quả.

Mạng Nơron mờ là mạng nơron có chức năng năng tương tự hệ suy diễn mờ trong đó

các luật mờ dạng IF-THEN được huấn luyện qua các tham số của hàm thuộc và trọng

số của mạng.

Một ví dụ điển hình của mạng nơron mờ là ANFIS (Adaptive Neuro-Fuzzy Inference System) – hệ suy luận mờ dựa trên mạng thích nghi. Trong đó, mạng nơron được sử dụng để tối ưu hóa các tham số mờ như hàm thuộc, hệ luật, và trọng số.

ANFIS (Adaptive Neuro-Fuzzy Inference System) là một kiến trúc mô hình lai (hybrid), kết hợp một cách thông minh những ưu điểm của Mạng Nơ-ron Nhân tạo (Artificial Neural Networks) và Hệ thống Logic Mờ (Fuzzy Logic Systems). Về bản chất, ANFIS sử dụng kiến trúc của một mạng nơ-ron truyền thẳng nhiều lớp để hiện thực hóa một hệ thống suy luận mờ, từ đó tạo ra một mô hình vừa có khả năng học từ dữ liệu, vừa có khả năng diễn giải giống như con người.

Cấu trúc của ANFIS gồm 5 lớp: lớp mờ hóa, lớp luật, lớp chuẩn hóa, lớp giải mờ, lớp tổng hợp [2].

Ngoài ra, mạng nơron mờ đa tác động (MANFIS) là một cải tiến của ANFIS, trong đó nhiều mạng ANFIS được kết nối song song để xử lý dữ liệu phức tạp hơn, nhiều đầu ra hơn và tăng độ chính xác trong các bài toán phi tuyến tính [5].

### Lớp mờ hóa

Lớp mờ hóa: ánh xạ đầu vào sang tập mờ thông qua các hàm thuộc như Gaussian, tam giác, hình thang,...

Đây là bước ánh xạ giá trị số của biến đầu vào sang mức độ thuộc về các tập mờ – ví dụ như “trẻ”, “trung niên”, “cao”, “thấp”, v.v…

Giả sử đầu vào có tập mờ , được biểu diễn dưới dạng:

𝜇𝐴j() với j = 1, 2, ..., m *(2-1)*

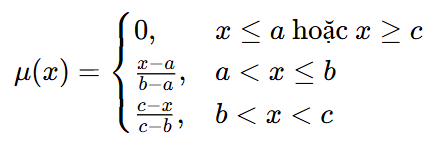
Trong đó:

𝜇𝐴j() là độ thuộc của giá trị vào tập mờ Aj

Các hàm thuộc có thể biểu diễn dưới dạng tam giác, hình thang, Gauss, Bell

Với hàm thuộc tam giác, ta có:

Hàm thuộc được xây dựng theo cấu trúc tam giác (trimf) như sau:

 *(2-2)*

Trong đó:

a: điểm bắt đầu (độ thuộc = 0)

b: đỉnh tam giác (độ thuộc = 1)

c: điểm kết thúc (độ thuộc = 0)

Với hàm Gauss:

 (*2-3)*

Trong đó:

μ (mu - tâm): Vị trí trung tâm của đường cong, nơi có độ thuộc bằng 1

σ (sigma - độ lệch chuẩn): Kiểm soát độ rộng của đường cong

(σ càng nhỏ, đường cong càng hẹp và dốc; σ càng lớn, đường cong càng rộng và phẳng)

Ở đây tâm và độ lệch chuẩn được khởi tạo như những trọng số có thể huấn luyện (trainable weights). Điều này cho phép mạng tự động học và điều chỉnh vị trí cũng như hình dạng của các hàm thuộc trong quá trình huấn luyện để phù hợp nhất với dữ liệu, thay vì phải dựa vào kiến thức của chuyên gia

### Lớp luật

Lớp luật: biểu diễn các luật IF–THEN. Mỗi luật xác định một mức độ kích hoạt dựa trên kết hợp của các điều kiện mờ.

Trong mô hình mạng nơron mờ, mỗi nơron ở lớp luật tương ứng với một luật dạng:

*(2-4)*

Trong đó

là tập mờ của biến trong luật thứ k

là đầu ra của luật thứ k

có thể là hằng số hoặc hàm tuyến tính

### Lớp chuẩn hóa

Lớp chuẩn hóa: Chuẩn hóa các mức độ kích hoạt và tính toán đầu ra suy luận.

Giả sử có N luật mờ, mỗi luật i có mức độ kích hoạt (firing strength) là ​ được tính bằng tích các độ thuộc:

𝜇𝐴i()𝜇Bi()… *(2-5)*

Lớp chuẩn hóa tính toán trọng số chuẩn hóa cho từng luật như sau:

*(2-6)*

Trong đó:

là mức độ kích hoạt chuẩn hóa của luật thứ i

là các trọng số chuẩn hóa có tổng bằng 1

Mục đích của việc chuẩn hóa là biến tất cả mức kích hoạt thành tỷ lệ tương đối, từ đó dễ dàng kết hợp lại ở bước tiếp theo.

Mỗi nơron trong lớp này tương ứng với một luật mờ. Ví dụ: nơron R1 tương ứng với Luật 1 sẽ nhận input từ nơron A1 và B1, sau đó sử dụng toán tử tích (phép giao - AND) để tính toán đầu ra.

### Lớp giải mờ

Lớp giải mờ: đưa ra kết quả cuối cùng (dạng phân loại hoặc xác suất)

Mục đích: Tính toán kết quả đầu ra (vế THEN) cho mỗi luật.

Mỗi nơ-ron trong lớp này tính toán kết quả của một luật dựa trên mô hình Sugeno bậc nhất: fi​ = pi​x + qi​y + ri​. Sau đó, kết quả này được nhân với độ kích hoạt đã được chuẩn hóa của luật đó ().

Tham số: Các tham số của lớp này được gọi là tham số hệ quả (consequent parameters), bao gồm các hệ số pi​, qi​, ri​. Đây chính là các tham số mà mạng nơ-ron chủ yếu học trong quá trình huấn luyện.

### Lớp tổng hợp

Mục đích: Tổng hợp kết quả từ tất cả các luật để đưa ra một giá trị đầu ra rõ (crisp output) duy nhất.

Hoạt động: Lớp này chỉ có một nơ-ron duy nhất, thực hiện phép cộng tất cả các đầu ra từ lớp giải mờ

## Mô hình mạng nơron mờ trong dự đoán khách hàng trung thành

Một trong những ứng dụng phổ biến nhất của mạng nơron mờ là dự báo hành vi, trong đó bao gồm cả việc dự đoán khách hàng có quay lại mua hàng hay không.

### Bài toán đặt ra

Bài toán dự đoán khách hàng trung thành là một bài toán phân loại nhị phân: đầu ra là "Yes" nếu khách hàng quay lại trong vòng 6 tháng, "No" nếu không. Các đầu vào có thể bao gồm:

* Thông tin nhân khẩu học: độ tuổi, giới tính, nghề nghiệp, tình trạng hôn nhân, nhà mạng,...
* Hành vi người dùng: số lần click, số lần thêm vào giỏ hàng, mua hàng, số lượng thêm sản phẩm vào yêu thích,...

Những thông tin này đa phần không có ranh giới rõ ràng, ví dụ: một người 30 tuổi có thể vừa thuộc nhóm “trẻ” vừa “trung niên”; hoặc số lần mua hàng 5 là cao hay trung bình? Chính vì vậy, việc xử lý mờ là cần thiết để phản ánh đúng bản chất dữ liệu.

### Lý do chọn mạng nơron mờ

Theo các nghiên cứu so sánh mô hình trong tài liệu [5], mạng nơron mờ thường vượt trội so với các mô hình như mạng nơron thuần túy hay hồi quy logistic, đặc biệt là trong các bài toán có dữ liệu mơ hồ, thiếu rạch ròi. Bên cạnh đó, mạng nơron mờ - kết hợp của logic mờ và mạng nơron mờ - có những ưu điểm của từng mô hình được kết hợp cũng như giải quyết được nhược điểm của chúng:

* Logic mờ giúp diễn giải dữ liệu không chắc chắn bằng ngôn ngữ tự nhiên.
* Mạng nơron giúp tự động tối ưu luật và các tham số mờ từ dữ liệu.
* ANFIS hoặc MANFIS giúp mở rộng khả năng học cho các bài toán nhiều biến đầu vào phức tạp.

Ví dụ ứng dụng luật mờ:

IF Tuổi is Trung niên AND số lượt mua is Cao THEN khách hàng is Trung thành

IF hành vi is Click nhiều AND không mua hàng THEN khách hàng is Không trung thành

Các luật mờ này được khởi tạo từ chuyên gia hoặc dữ liệu, sau đó được mạng huấn luyện và điều chỉnh thông qua thuật toán lan truyền ngược hoặc hybrid.

## Đánh giá kết quả mô hình

### Tính chính xác

### Ma trận nhầm lẫn

* Pi (true positive): Số lượng dữ liệu thuộc lớp ci được phân loại chính xác vào lớp ci
* FPi (false positive): Số lượng dữ liệu bên ngoài bị phân loại nhầm vào lớp ci
* TNi (true negative): Số lượng dữ liệu không thuộc lớp ci được phân loại (chính xác)
* FNi (false negative): Số lượng dữ liệu thuộc lớp ci bị phân loại nhầm (vào các lớp khác ci)T

Bảng 2.3. Minh họa ma trận nhầm lẫn

|  |  |  |  |
| --- | --- | --- | --- |
| Lớp ci | | Được phân loại bởi hệ thống | |
| Thuộc | Không thuộc |
| Nhãn lớp đúng | Thuộc | TPi | FNi |
| Không thuộc | FPi | TNi |

### Precision và recall

Precision đối với lớp ci: Tổng số các ví dụ thuộc lớp ci phân loại chính xác được chia cho tổng số các ví dụ được phân loại vào lớp ci

Precision(ci) = TPi / (TPi +FPi)

Recall đối với lớp ci: Tổng số các ví dụ thuộc lớp ci phân loại chính xác được chia cho tổng số các ví dụ thuộc lớp ci

Recall(ci) = TPi / (TPi +FNi)

Tiêu chí đánh giá F1 là sự kết hợp của 2 tiêu chí đánh giá Precision và Recall

F1 = 2\*(Precision \* Recall)/(Precision + Recall)

# DỮ LIỆU VÀ TIỀN XỬ LÝ DỮ LIỆU

## Nguồn dữ liệu

Bộ dữ liệu được công bố trong cuộc thi DAZONE ([DAZONE - Cuộc thi phân tích dữ liệu](https://www.facebook.com/DAZONE.RCS/)), dữ liệu được thu thập gồm hơn 50 triệu bản ghi, từ khoảng 5000 cửa hàng, hơn 200 nghìn khách hàng và hơn 1 triệu sản phẩm.

Nguồn dữ liệu: [BỘ DỮ LIỆU](https://drive.google.com/drive/folders/1d0Ptgy3d1neQcG4TXu0wYcxRl1fTMVhR)

Bộ dữ liệu là thông tin tổng hợp của khách hàng từ một sàn thương mại điện thử trong một khoảng thời gian cố định, trong đó bao gồm: Danh sách các gian hàng (merchants), thông tin người dùng, lịch sử hành vi người dùng với các gian hàng như xem sản phẩm, mua hàng, v.v… theo thời gian.

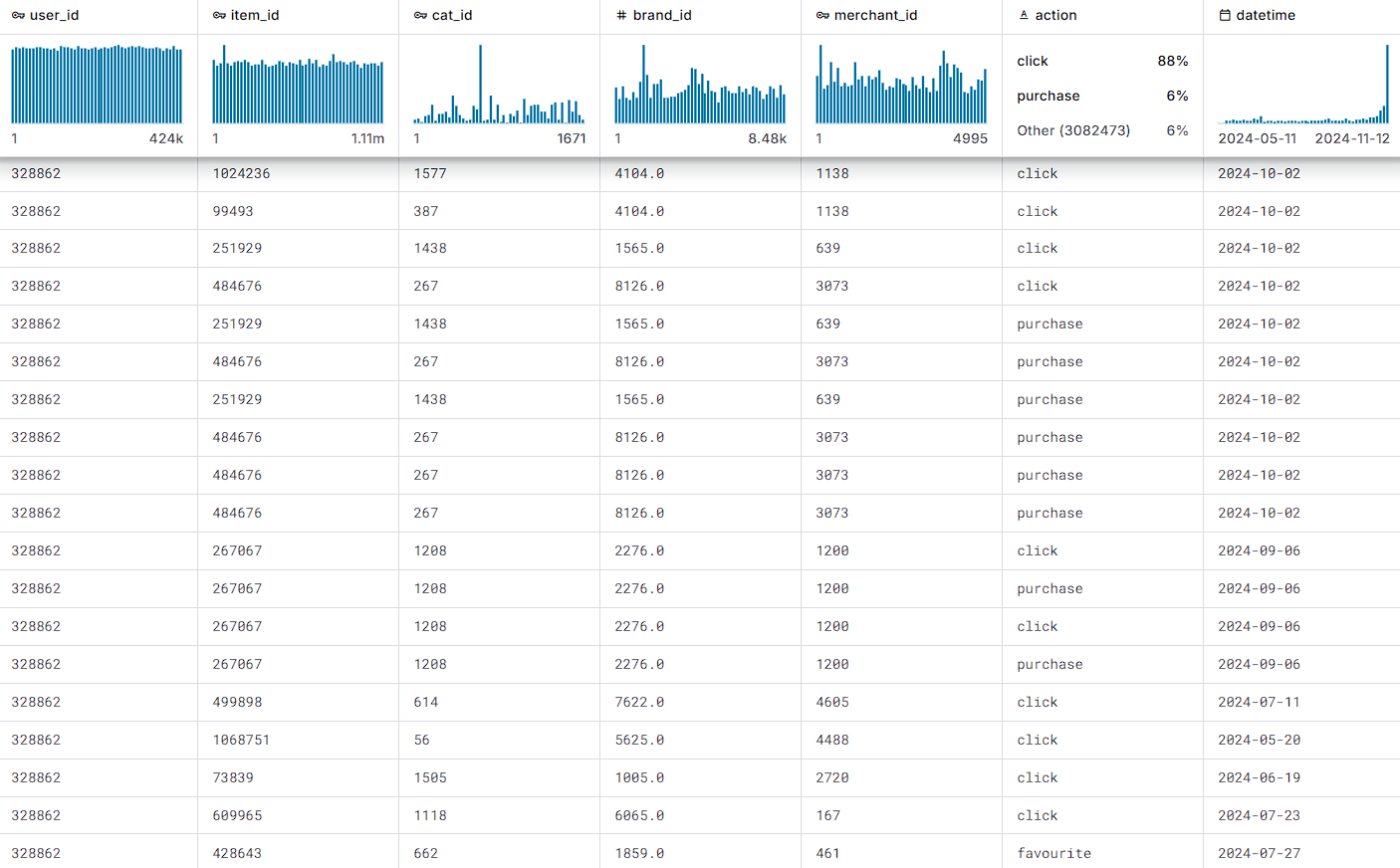
Bộ dữ liệu được chia thành 4 file dữ liệu:

* user\_info: thông tin của người mua hàng, bao gồm các đặc điểm nhân khẩu học
* user\_log: ghi nhận hành vi của người dùng đối với các gian hàng, mặt hàng
* train: tập dữ liệu train, bao gồm nhãn
* test: tập dữ liệu test, bao gồm nhãn

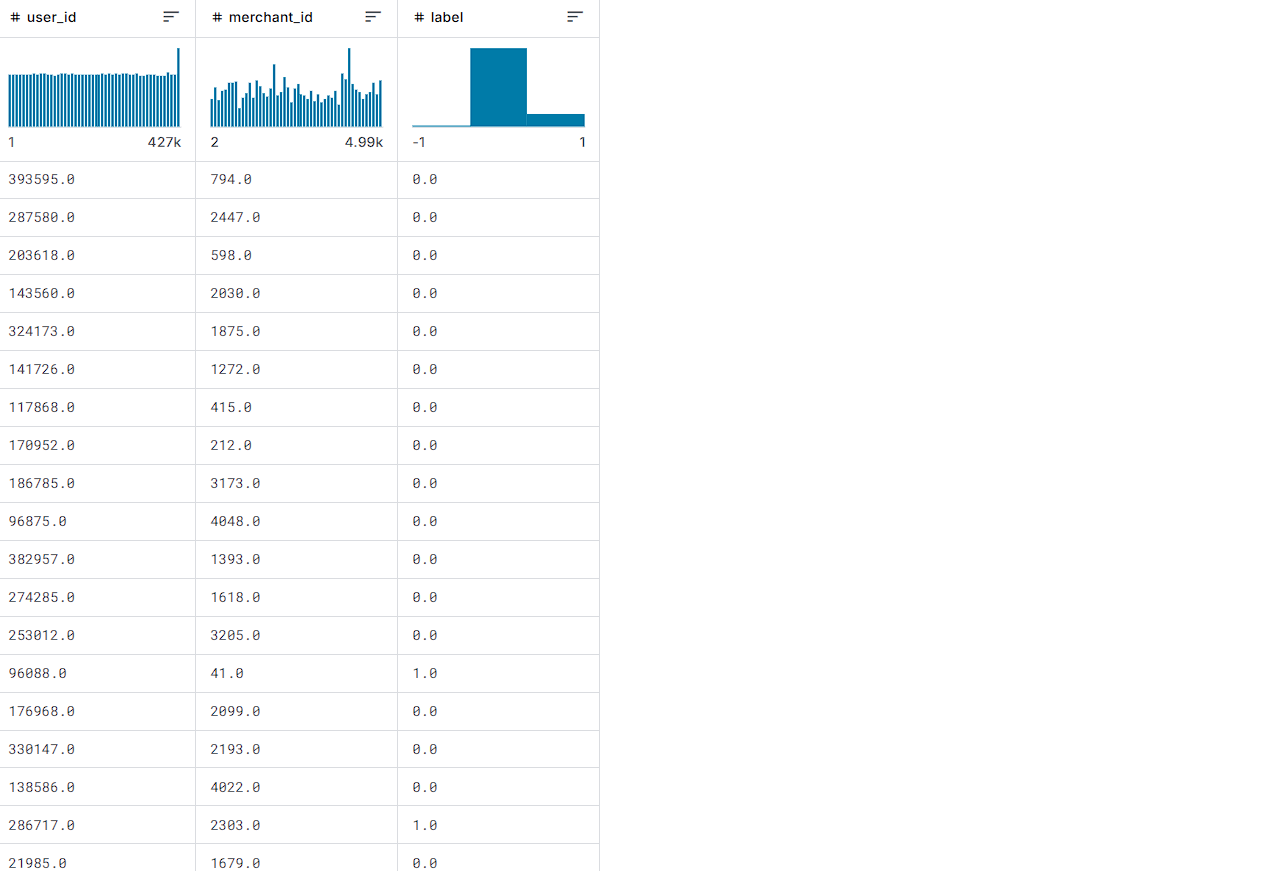
Để đọc dữ liệu ban đầu một cách trực quan, ta sử dụng Kaggle:



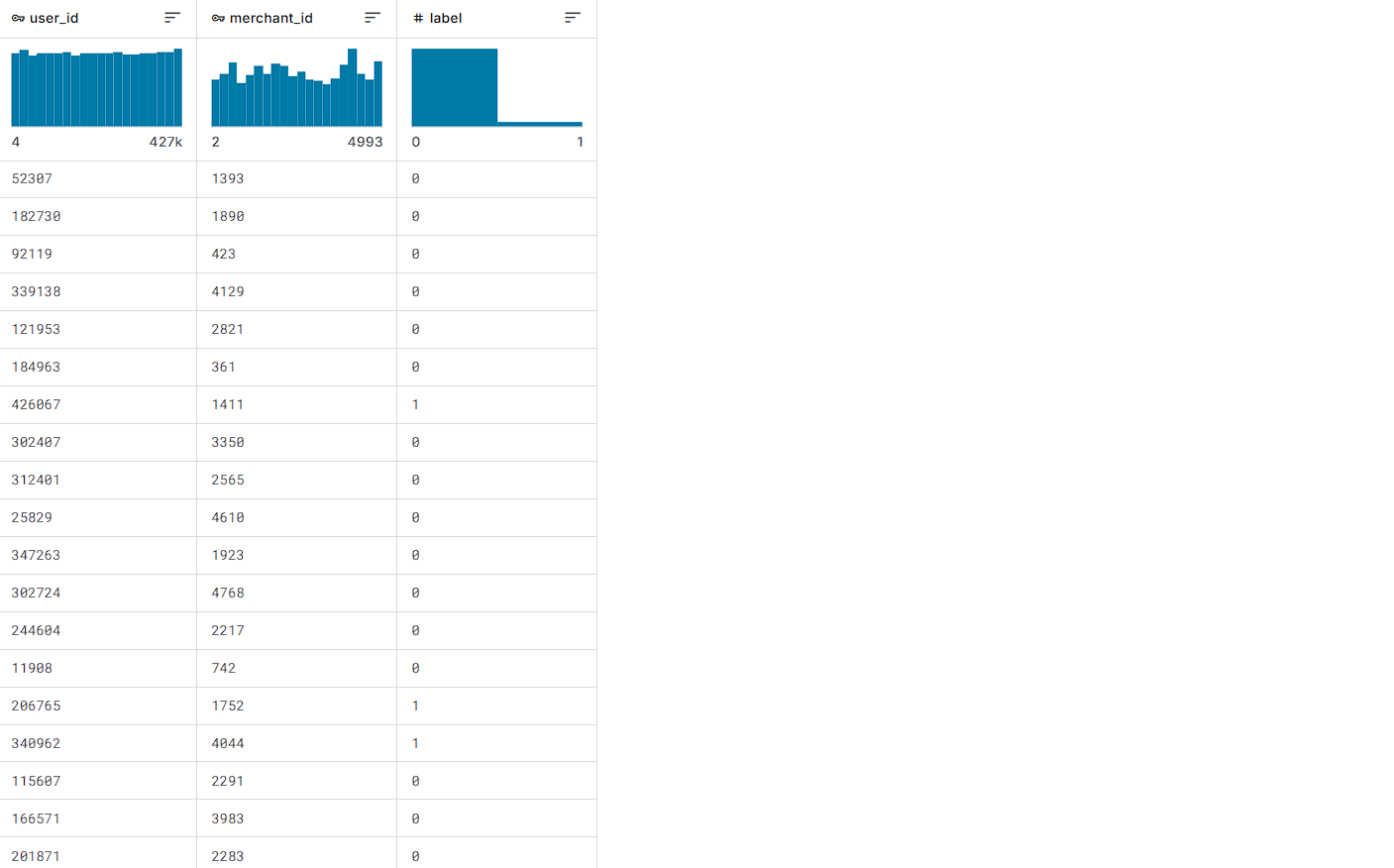
###### Dữ liệu ban đầu - user\_info



###### Dữ liệu ban đầu - user\_log



###### Dữ liệu ban đầu - train



###### Dữ liệu ban đầu - test

Dữ liệu gốc được đẩy lên Kaggle: [DAZONE 2025](https://www.kaggle.com/datasets/rmxrmx8112/dazone-2025?utm_medium=social&utm_campaign=kaggle-dataset-share&utm_source=facebook&fbclid=IwY2xjawKxTKhleHRuA2FlbQIxMQBicmlkETFYZ2d5c3Y2QTV2Y3p3TDROAR6CZ1ex2JiyFMyqew8ypDSvizAnLLvVaeCPic4T4KJAZfvhjl9y26PmR3Hy1A_aem_ZGwVhwKR8MkR6nTqpmq7YA)

**Giải thích thuộc tính**

##### Bảng 3.1. Bảng mô tả ý nghĩa các thuộc tính trong file user\_info

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Mô tả** | **Ý nghĩa** |
| 1 | user\_id | ID người mua hàng | Được sử dụng để phân biệt từng cá nhân trong dataset |
| 2 | age | Độ tuổi của người mua hàng, thể hiện dưới dạng số nguyên | Độ tuổi theo nhóm tuổi có thể là yếu tố ảnh hưởng tới quyết định mua hàng |
| 3 | sex | Giới tính của người mua hàng | Giới tính có thể là yếu tố quyết định đối với mỗi loại sản phẩm |
| 4 | phone | Số điện thoại của người mua hàng |  |
| 5 | job | Nghề nghiệp của người mua hàng | Nghề nghiệp có thể liên quan tới quyết định mua hàng |
| 6 | carrier | Nhà mạng của người mua hàng | Nhà mạng quyết định tốc độ truy cập mạng |
| 7 | marital\_status | Tình trạng hôn nhân của người mua hàng | Tình trạng hôn nhân có thể liên quan tới quyết định mua hàng |

##### Bảng 3.2. Bảng mô tả ý nghĩa các thuộc tính trong file user\_log

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Mô tả** | **Ý nghĩa** |
| 1 | user\_id | ID người mua hàng | Được sử dụng để phân biệt từng cá nhân trong dataset |
| 2 | item\_id | ID sản phẩm | Sản phẩm ảnh hưởng tới khả năng khách hàng có quay lại hay không |
| 3 | cagetory\_id | ID nhóm sản phẩm | Từng nhóm sản phẩm sẽ hướng đến những đối tượng khách hàng khác nhau |
| 4 | merchant\_id | ID gian hàng bán sản phẩm | Có thể đánh giá hiệu suất bán hàng trên các sàn thương mại điện tử |
| 5 | brand\_id | ID thương hiệu sản phẩm | Thương hiệu sản phẩm có thể ảnh hưởng đến độ tiếp cận và độ tin tưởng của khách hàng |
| 6 | action\_type | Hành động của người dùng, trong đó có click (click vào xem sản phẩm), add to cart (thêm sản phẩm vào giỏ hàng), purchase (mua hàng), add to favourtie (thêm vào sản phẩm yêu thích) | Hành vi người dùng có yếu tố quyết định đối với khả năng khách hàng quay lại |
| 7 | datetime | Thời gian ghi nhận hành động trên hệ thống | Thời gian ghi nhận hành động đối với từng hành động click/add to cart/purchase/add to favourite |

##### Bảng 3.3. Bảng mô tả ý nghĩa các thuộc tính trong file train

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Mô tả** | **Ý nghĩa** |
| 1 | user\_id | ID người mua hàng | Được sử dụng để phân biệt từng cá nhân trong dataset |
| 2 | merchant\_id | ID gian hàng bán sản phẩm | Có thể đánh giá hiệu suất bán hàng trên các sàn thương mại điện tử |
| 3 | label | Khách hàng có quay lại mua hàng trong vòng 6 tháng hay không | Mục tiêu của bài toán |

##### Bảng 3.4. Bảng mô tả ý nghĩa các thuộc tính trong file test

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Mô tả** | **Ý nghĩa** |
| 1 | user\_id | ID người mua hàng | Được sử dụng để phân biệt từng cá nhân trong dataset |
| 2 | merchant\_id | ID gian hàng bán sản phẩm | Có thể đánh giá hiệu suất bán hàng trên các sàn thương mại điện tử |
| 3 | label | Khách hàng có quay lại mua hàng trong vòng 6 tháng hay không | Mục tiêu của bài toán |

## Tiền xử lý dữ liệu

### Kết hợp dữ liệu từ các bảng rời rạc

Bộ dữ liệu tuy phong phú nhưng không liên kết mà rời rạc thành các bảng, nên để có được dữ liệu hoàn chỉnh để áp dụng phân tích thì phải kết hợp dữ liệu.

Trước tiên, ta chuyển kiểu của cột datetime từ kiểu chuỗi (string) sang kiểu datetime cho phù hợp định dạng. 

###### Code chuyển kiểu của cột datetime từ string -> datetime

Vì dữ liệu của các hành vi như *add-to-cart, click, favourite, purchase* trong bảng *user\_log* đang được thu thập là hành vi của người dùng theo từng gian hàng, nghĩa là mỗi người dùng có thể sẽ có nhiều hành vi giống nhau với cùng 1 gian hàng ở nhiều thời điểm khác nhau, nên muốn biết người dùng đã tương tác như thế nào với từng gian hàng, ta tiến hành gom nhóm thành 1 bảng hành vi, có chứa tổng số lần các hành vi mà người dùng tương tác cùng thời gian tương ứng.



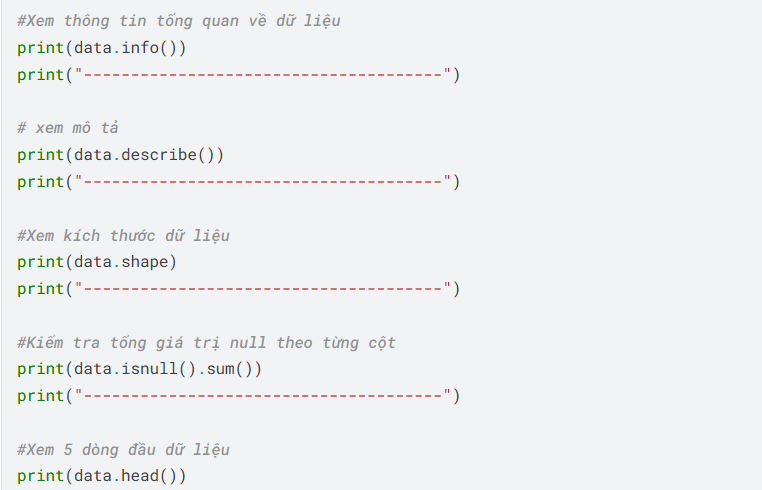
###### Code tạo bảng đếm các hành vi của người dùng với từng gian hàng theo thời gian

Sau đó sẽ ghép các bảng hành vi này với bảng *user\_info* và bảng *train* để được dữ liệu hoàn chỉnh có chứa thông tin về nhân khẩu học và hành vi tương tác của từng người với các gian hàng, có nhãn để phân loại có phải khách hàng trung thành hay không.

###### Code tạo bảng data, chứa dữ liệu chính để xử lý bài toán

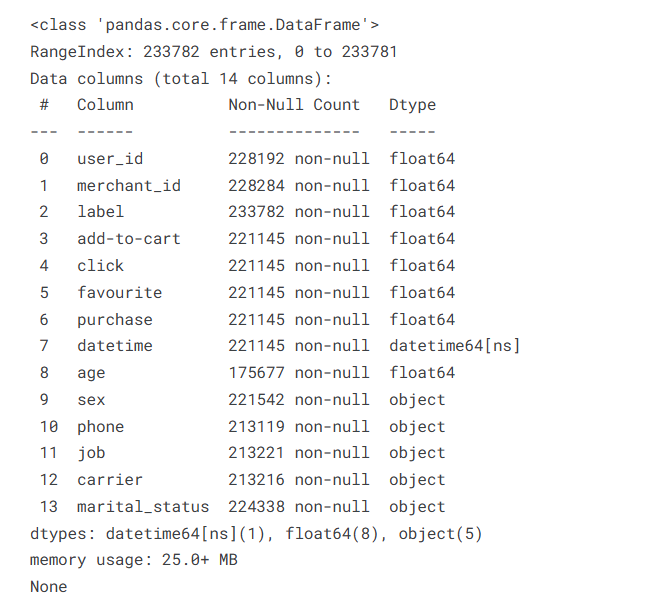
### Khám phá dữ liệu sau kết hợp

Để có những hướng xử lý tiếp theo, trước tiên sẽ khám phá tổng quan về dữ liệu vừa kết hợp được.



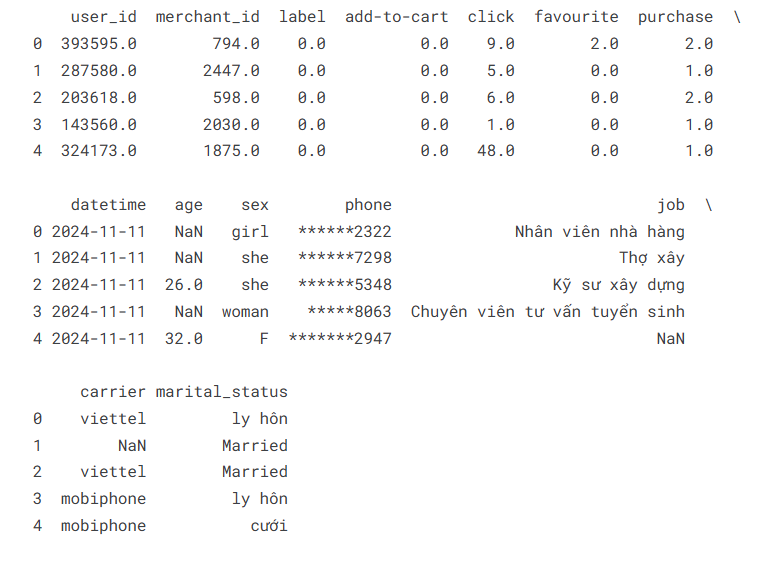
###### Code khám phá tổng quan dữ liệu

Kết quả:

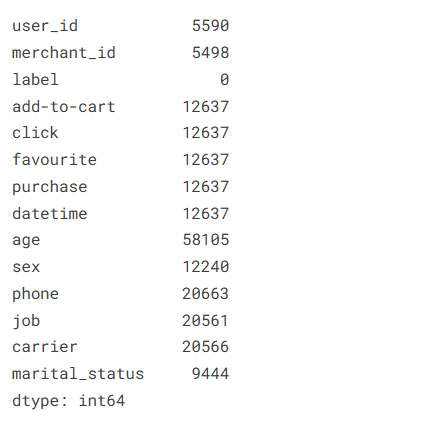


###### Bảng mô tả thông tin dữ liệu

Cho thấy dữ liệu có 233782 dòng, có 14 cột, trong đó 13 cột thuộc tính là *user\_id, merchant\_id, datetime, add-to-cart, click, favourite, purchase, age, sex, job, carrier, marital\_status* và cột nhãn *label*



###### Bảng mô tả 5 dòng đầu của dữ liệu



###### Bảng mô tả tổng giá trị null theo từng cột của dữ liệu

Như vậy, qua xem xét tổng quan thấy được dữ liệu sau khi kết hợp đã giảm được độ lớn tương đối. Điều này là phù hợp để huấn luyện mô hình dự đoán khách hàng trung thành trên những dữ liệu đã gán nhãn so với toàn bộ tập dữ liệu ban đầu. Đồng thời thấy được bộ dữ liệu cũng chứa nhiều giá trị liên tục và giá trị NaN, đa ngôn ngữ cần phải xử lý trước khi huấn luyện.

Phần tiếp theo chúng em sẽ tiến hành làm sạch dữ liệu.

### Làm sạch dữ liệu

Các thao tác làm sạch dữ liệu bao gồm xóa giá trị NaN (các giá trị null), loại bỏ cột *phone* không có ý nghĩa trong thống kê. Bên cạnh đó, để xử lý vấn đề đa ngôn ngữ trong các cột có giá trị kiểu liên tục, tức là có nhiều giá trị có ý nghĩa giống nhau, phương pháp là gom nhóm những giá trị chung ý nghĩa này.



###### Code xóa các giá trị NaN trong dữ liệu



###### Code loại bỏ cột phone

Kết quả sau khi xóa dữ liệu còn 146370 dòng, 13 cột. Có 3 cột chúng ta cần gom nhóm là *sex, job* và *marital\_status*.

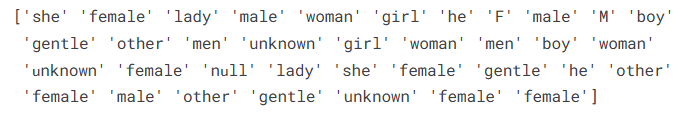
Các bước thực hiện là:

1. Định nghĩa ánh xạ để chuẩn hóa các giá trị
2. Áp dụng ánh xạ để chuẩn hóa cột
3. Tính mode của các giá trị đã được chuẩn hóa
4. Thay thế các giá trị *Unknown* bằng mode

Sau khi thực hiện gom nhóm các giá trị đã được chuẩn hóa về cùng nhóm. Hình ảnh code thực hiện chi tiết, chúng em để ở phần phụ lục.

\* Với cột *sex:*

* Trước khi biến đổi có các giá trị:

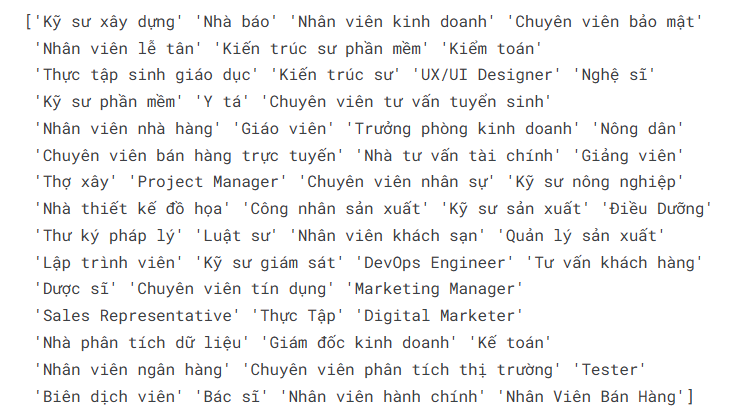


* Sau khi biến đổi các giá trị được phân thành 2 nhóm:

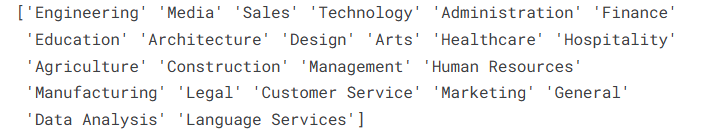


\* Với cột *job*:

* Trước khi biến đổi có các giá trị:



* Sau khi biến đổi có các giá trị (các ngành nghề được có điểm chung được phân về 1 nhóm, sau cùng có 23 nhóm)



\* Với cột *marital\_status:*

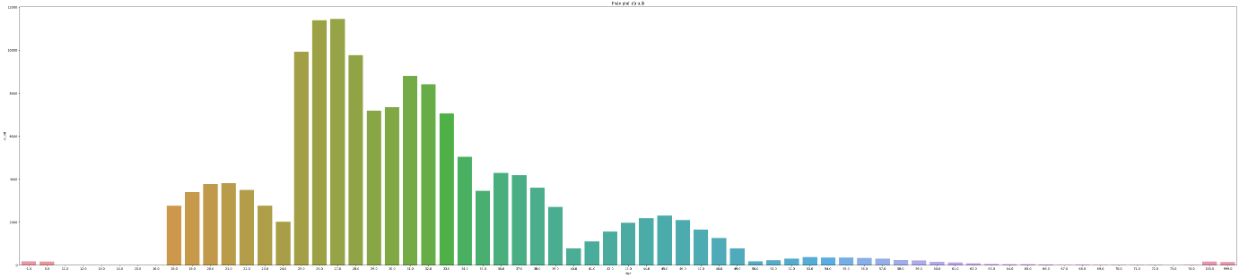
* Trước khi biến đổi có các giá trị:



* Sau khi biến đổi:



Dưới đây là biểu đồ thể hiện phân phối dữ liệu liệu cho cột *age:*

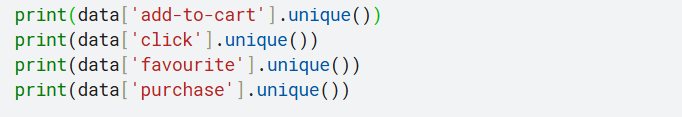


###### Phân phối dữ liệu cột age

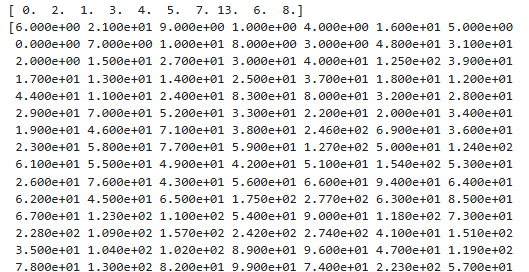
Dựa vào biểu đồ, bằng mắt thường ta quan sát được các giá trị ngoại lai là -1, 0, 150, 999 và để không ảnh hưởng đến độ chính xác, ta sẽ xóa những giá trị này đi bằng lệnh



Tiếp theo sẽ xử lý đến các cột có giá trị rời rạc, gồm cột *click, add-to-cart, purchase, favourite*. Dùng hàm unique() để xem các cột này có những giá trị gì.

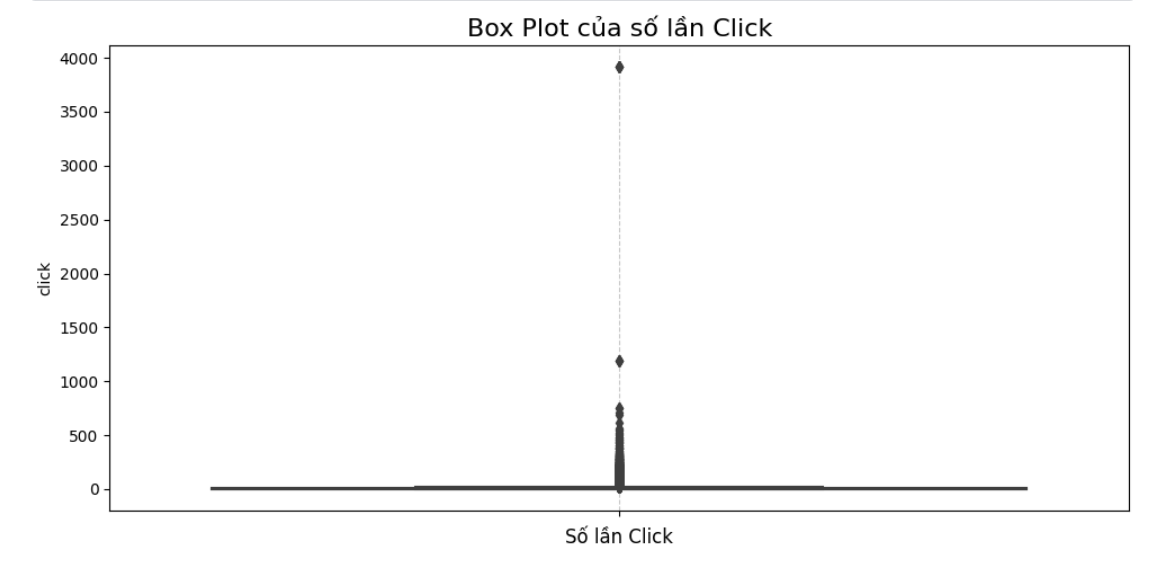


Kết quả:

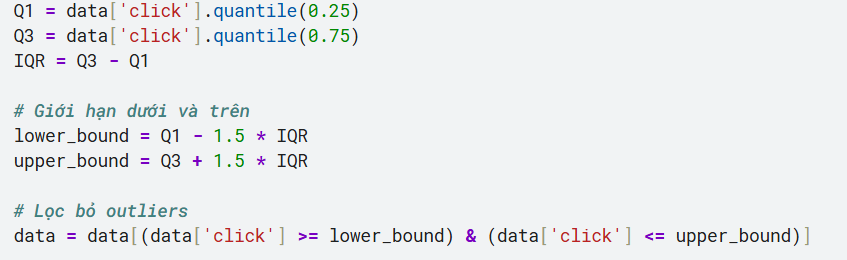




Khi xem xét kết quả trả về nhận thấy rằng chỉ có cột *click* có chứa giá trị bất thường, nên ta sẽ tập trung xử lý cột này. Dùng biểu đồ box plot để kiểm tra mô tả phân bố số lần click của người dùng trong tập dữ liệu.

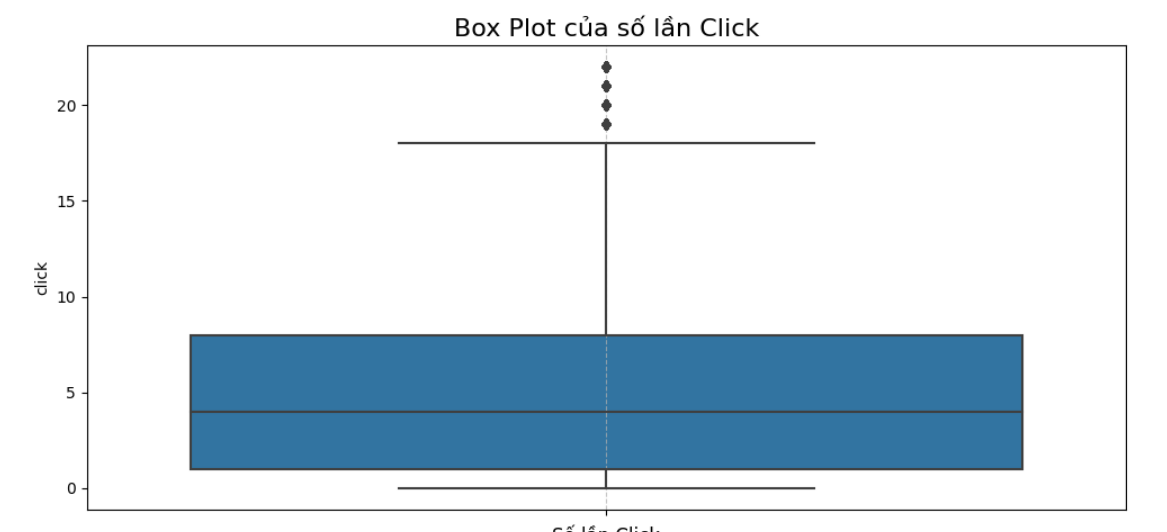


Qua biểu đồ cho thấy rằng phần lớn người dùng có số lần click thấp, nhưng vẫn tồn tại một số trường hợp đặc biệt với số lần click rất cao (4000 lần), tạo ra các giá trị ngoại lai. Những ngoại lai này có thể bắt nguồn từ hành vi người dùng bất thường, lỗi hệ thống, hoặc ảnh hưởng từ các chiến dịch quảng cáo riêng biệt. Để đảm bảo độ chính xác và tính đại diện của dữ liệu trong quá trình phân tích, các giá trị ngoại lai này sẽ được xử lý bằng cách sử dụng phương pháp IQR (Interquartile Range) nhằm loại bỏ những điểm nằm ngoài khoảng giá trị hợp lý.

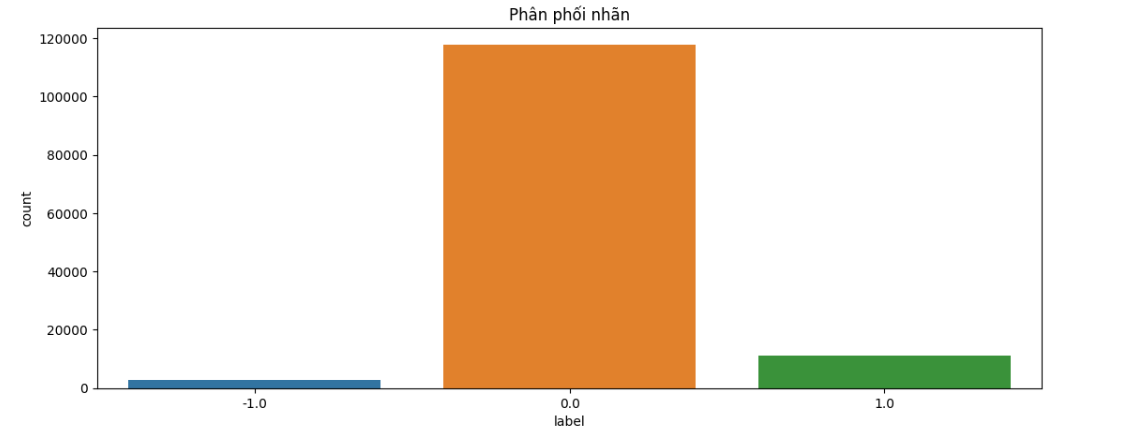


###### Code xử lý ngoại lai bằng phương pháp IQR cho cột click

Kết quả sau khi xử lý bằng IQR, biểu đồ Boxplot cho thấy phân bố dữ liệu trở nên rõ ràng và cân đối hơn, tập trung trong khoảng từ 0 đến dưới 20 lần click, với một số lượng nhỏ các giá trị ngoại lai nhẹ tuy vẫn xuất hiện nhưng chấp nhận được. Trung vị và các khoảng tứ phân vị được thể hiện rõ ràng hơn, cho thấy cấu trúc dữ liệu ổn định và đáng tin cậy hơn để tiến hành các bước phân tích tiếp theo.



Chúng ta cũng cần kiểm tra xem cột *label* có nhãn gây nhiễu không bằng cách kiểm tra phân phối nhãn.



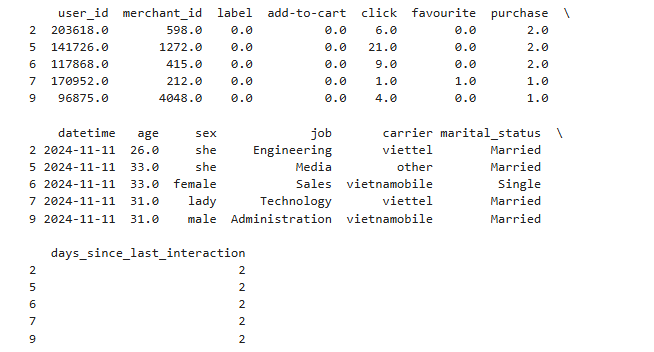
Biểu đồ cho thấy chỉ có 1 số ít nhãn gây nhiễu là -1 nên có thể xóa các nhãn này đi mà ko ảnh hưởng đến kết quả thống kê. Code thực hiện:



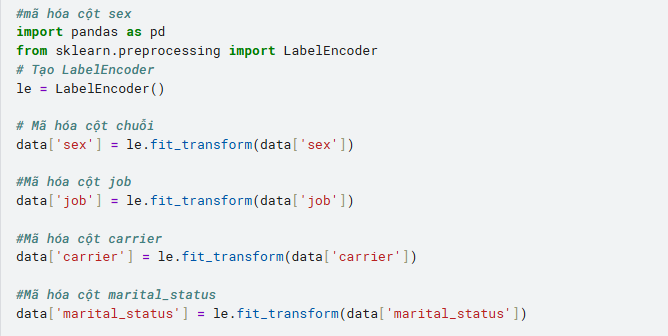
Với cột *datetime,* các giá trị hiện đang là dạng năm-ngày-tháng, thể hiện thời gian mà người dùng tương tác với từng gian hàng, nên để dễ thống kê, từ các giá trị ngày giờ cụ thể thành số ngày cuối cùng kể từ lần cuối người dùng tương tác với gian hàng. Để rõ ý nghĩa, tên cột cũng được đổi thành *days\_since\_last\_interaction.* Code thực hiện:



Sau các thao tác trên, về cơ bản dữ liệu đã sạch rồi, dưới đây là 5 dòng đầu của dữ liệu:

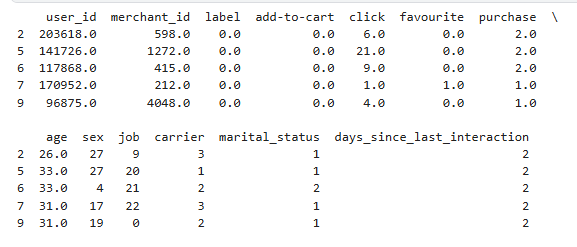


Tuy nhiên dữ liệu này vẫn chưa thể xử dụng ngay vì vẫn còn các giá trị liên tục (kiểu string). Nên bước cuối cùng trong quá trình làm sạch dữ liệu là mã hóa dữ liệu cho các giá trị này. Code thực hiện:



###### Code thực hiện mã hóa các giá trị liên tục

Kết quả dữ liệu sau cùng đã được làm sạch:

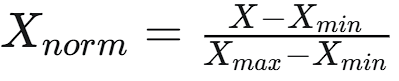


###### Bảng mô tả 5 dòng đầu dữ liệu sau quá trình làm sạch dữ liệu

### Chuẩn hóa dữ liệu

Với mục tiêu sử dụng mạng ANFIS, ta đưa tất cả các thuộc tính số (*add-to-cart, click, favourite, purchase, age, days\_since\_last\_interactio*n) về cùng một thang đo, ở đây là từ 0 đến 1:

Sử dụng phương pháp Min-Max Scaling:

 *(4-1)*

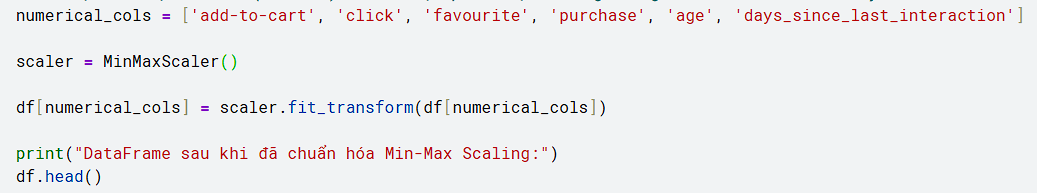
Trong đó:

X là giá trị gốc

Xnorm là giá trị sau khi chuẩn hóa

Xmin và Xmax là giá trị nhỏ nhất và lớn nhất của tập dữ liệu

Sử dụng Python:

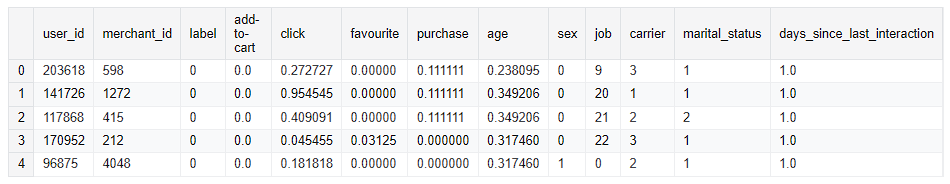


###### Code chuẩn hóa theo Min-Max Scaling

Ta sử dụng hàm MinMaxScaler() có sẵn trong thư viện sklearn và fit\_transform để học các giá trị min/max

Trong đó, fit sẽ "học" các giá trị nhỏ nhất (min) và lớn nhất (max) từ mỗi cột được chỉ định trong df[numerical\_cols] và transform sẽ dựa trên các giá trị min và max đã học, nó áp dụng công thức Xnorm ở trên để biến đổi tất cả các giá trị trong các cột đó về thang đo [0, 1].

Kết quả chuẩn hóa:



###### Kết quả chuẩn hóa

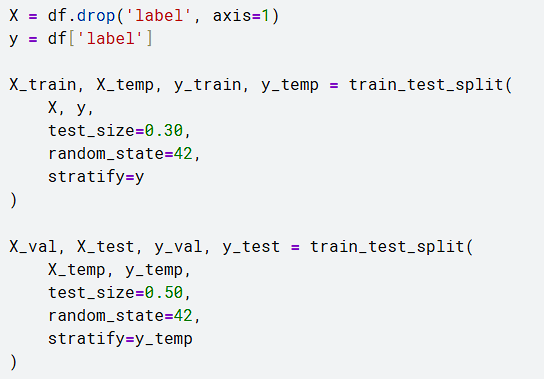
### Phân chia dữ liệu

Ta sử dụng hàm train\_test\_split của thư viện scikit-learn (vì hàm này chỉ chia dữ liệu thành hai phần một lúc) quy trình này được thực hiện trong 2 bước:

Bước 1: Chia toàn bộ dữ liệu thành 2 phần: 70% cho tập huấn luyện (train) và 30% còn lại cho một tập tạm (temp).

Bước 2: Chia tập tạm (temp) đó thành hai phần bằng nhau: 50% của nó cho tập kiểm định (validation) và 50% còn lại cho tập kiểm tra (test). (50% của 30% sẽ là 15% so với tổng thể).

Sử dụng Python:



###### Code phân chia dữ liệu

Như vậy, sau khi trộn và phân chia tập dữ liệu ban đầu, ta có:

* (X\_train, y\_train): Dùng để huấn luyện mô hình.
* (X\_val, y\_val): Dùng để theo dõi và tinh chỉnh các siêu tham số trong quá trình huấn luyện.
* (X\_test, y\_test): Dùng để đánh giá hiệu suất cuối cùng của mô hình sau khi đã huấn luyện xong.

# XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH

## Thiết kế và xây dựng mạng nơron mờ (ANFIS)

### Phân tích dữ liệu và thuộc tính đầu vào

Sau khi tiến hành tiền xử lý dữ liệu, tập dữ liệu sử dụng cho mô hình mạng nơron mờ bao gồm 129067 dòng (mỗi dòng tương ứng với một tương tác người dùng – gian hàng) và 13 thuộc tính. Dữ liệu này được tổng hợp từ nhiều nguồn: thông tin người dùng, hành vi tương tác, đặc điểm gian hàng và nhãn đầu ra.

##### Bảng 4.1 Bảng minh họa thuộc tính đầu vào

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Vai trò** |
| 1 | user\_id | ID người mua | Định danh – không đưa vào mô hình |
| 2 | merchant\_id | ID gian hàng | Định danh – không đưa vào mô hình |
| 3 | add-to-cart | Số lần thêm vào giỏ hàng | Đặc trưng hành vi |
| 4 | click | Số lần click vào sản phẩm | Đặc trưng hành vi |
| 5 | favourite | Số lần thêm sản phẩm vào mục yêu thích | Đặc trưng hành vi |
| 6 | purchase | Số lần mua hàng | Đặc trưng hành vi - quan trọng |
| 7 | age | Tuổi người dùng | Nhân khẩu học – mờ hóa theo nhóm tuổi |
| 8 | sex | Giới tính (0: Nam, 1: Nữ) | Nhân khẩu học |
| 9 | job | Mã hóa nghề nghiệp | Nhân khẩu học – cần gom nhóm |
| 10 | carrier | Mã hóa nhà mạng | Có thể ảnh hưởng đến tần suất truy cập |
| 11 | marital\_status | Mã hóa tình trạng hôn nhân (1: Độc thân, 2: Đã kết hôn) | Nhân khẩu học |
| 12 | days\_since\_last\_interaction | Số ngày kể từ tương tác gần nhất | Đặc trưng hành vi thời gian |
| 13 | label | Nhãn đầu ra (0: không quay lại, 1: quay lại trong 6 tháng) | Mục tiêu của bài toán |

Trong số các thuộc tính trên, các thuộc tính như *user\_id* và *merchant\_id* chỉ mang tính định danh và không được đưa vào mô hình học nên ta loại bỏ các thuộc tính định danh không có giá trị dự đoán như *user\_id* và *merchant\_id*

Giữ lại các thuộc tính còn lại: *add-to-cart, click, favourite, purchase, age, sex, job, carrier, marital\_status, days\_since\_last\_interaction*

### Lựa chọn hàm thuộc

Đối với các thuộc tính *add-to-cart, click, favourite, purchase, age, days\_since\_last\_interaction*, ranh giới giữa các khái niệm là không rõ ràng. Ví dụ, không có một độ tuổi chính xác nào mà một người đột ngột chuyển từ "Trẻ" sang "Trung niên" một cách rời rạc. Tuy vậy, logic mờ cho phép một giá trị (ví dụ: tuổi 28) có thể thuộc về cả hai nhóm với các mức độ khác nhau (ví dụ: 80% "Trẻ" và 20% "Trung niên"), giúp mô hình nắm bắt được sự chuyển đổi mềm dẻo này.

Các thuộc tính còn lại (*sex, job, carrier, marital\_status*) đã được mã hóa theo từng giá trị phân loại, chúng đại diện cho các phạm trù riêng biệt, rõ ràng và không có sự mơ hồ hay tính liên tục (ví dụ: một người thuộc hoặc không thuộc một ngành nghề nào đó, không có khái niệm "hơi hơi là kỹ sư"). Việc mờ hóa các biến này là không cần thiết và không mang lại giá trị cho mô hình.

Như vậy, sẽ có 6 thuộc tính được đưa vào mô hình: *add-to-cart, click, favourite, purchase, age, days\_since\_last\_interaction.*

Với mỗi thuộc tính, ta sử dụng 3 hàm thuộc có dạng: “Ít”, “Trung bình”, “Nhiều” tương ứng với tần suất của giá trị tương ứng với thuộc tính đó. Thuộc tính tuổi sẽ được chia là “Trẻ”, “Trung niên”, “Già”.

Việc lựa chọn 3 hàm liên thuộc ban đầu cho mỗi đầu vào nhằm mục đích đạt được độ chi tiết cao hơn trong việc biểu diễn các biến đầu vào. Tuy nhiên, độ chi tiết này phải trả giá bằng sự gia tăng theo cấp số nhân của các luật, có thể dẫn đến tích số tiệm cận không (triệt tiêu gradient - khiến mô hình ngừng học), ta sẽ cần tối ưu trong việc xây dựng hàm thuộc và tập luật mờ.

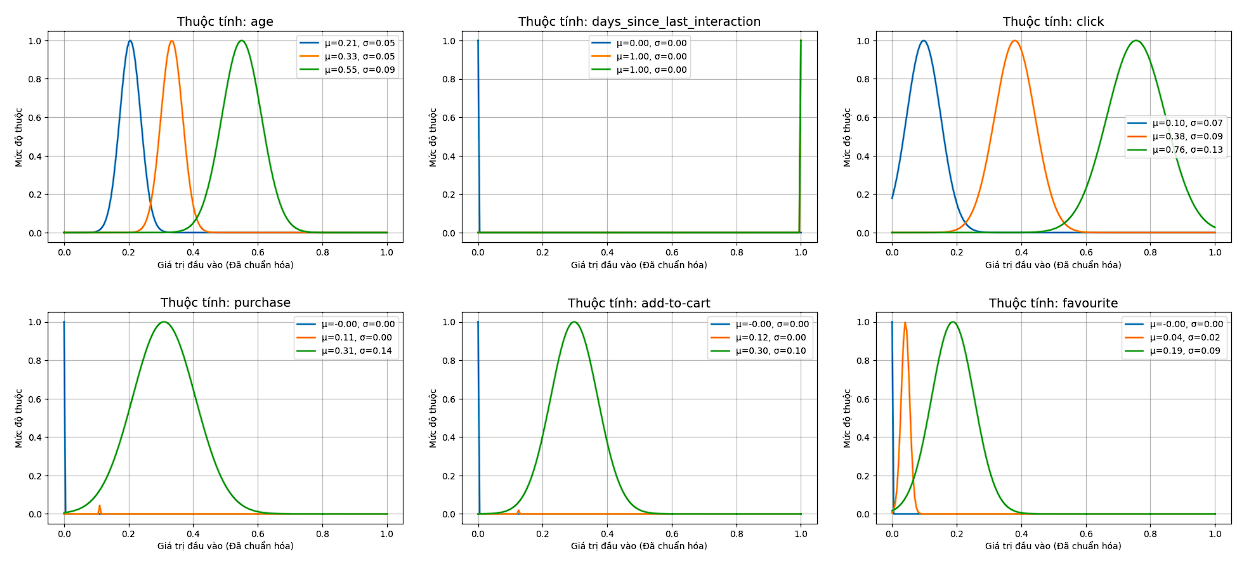
Việc lựa chọn hình dạng hàm liên thuộc ảnh hưởng đến độ mượt của hàm ANFIS tổng thể và đến các gradient. Với 3 hàm thuộc, ta có thể sử dụng hàm tam giác. Các hàm mượt hơn (như Gaussian) có thể cung cấp các gradient liên tục và nhiều thông tin hơn, có khả năng giúp bộ tối ưu hóa điều hướng không gian lỗi hiệu quả hơn và tránh bị mắc kẹt trong các cực tiểu cục bộ sắc nét hoặc cao nguyên.

##### Bảng 4.2 Bảng so sánh các chiến lược hàm thuộc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Chiến lược** | **Số lượng luật dự kiến** | **Ưu điểm** | **Nhược điểm** | **Tác động dự kiến đến học** |
| Tam giác | 729 | Đơn giản, dễ triển khai | Tích số tiệm cận không nghiêm trọng, phức tạp quá mức, khó học, không giải thích được | Học không hiệu quả, lỗi không giảm, dự đoán không thay đổi |
| Dựa trên phân cụm (FCM/Subtractive) với Gaussian/Bell | Tối ưu hóa theo dữ liệu (có thể < 64) | Khởi tạo dựa trên dữ liệu thực tế, hàm liên thuộc phù hợp hơn, có khả năng giảm số luật tối ưu, hội tụ nhanh hơn, gradient mượt hơn | Phức tạp hơn trong triển khai, yêu cầu thư viện/công cụ phân cụm | Học hiệu quả cao, hội tụ nhanh, hiệu suất phân loại tốt, khả năng tổng quát hóa cao |

Dựa vào bảng so sánh trên, thay vì các hàm liên thuộc hình tam giác được định nghĩa thủ công hoặc chung chung, ta sẽ sử dụng các thuật toán phân cụm K-Means, v.v… để tự động suy ra các tham số hàm liên thuộc tối ưu (tâm và độ trải) từ dữ liệu huấn luyện với hàm Gauss.

Trực quan hóa bằng biểu đồ (màu xanh lam biểu thị cho tập mờ có giá trị Thấp, tương ứng với giá trị μ nhỏ nhất; màu cam biểu thị cho tập mờ có giá trị Trung bình, tương ứng với giá trị μ ở giữa; màu xanh lá biểu thị cho tập mờ có giá trị Cao, tương ứng với giá trị μ lớn nhất):



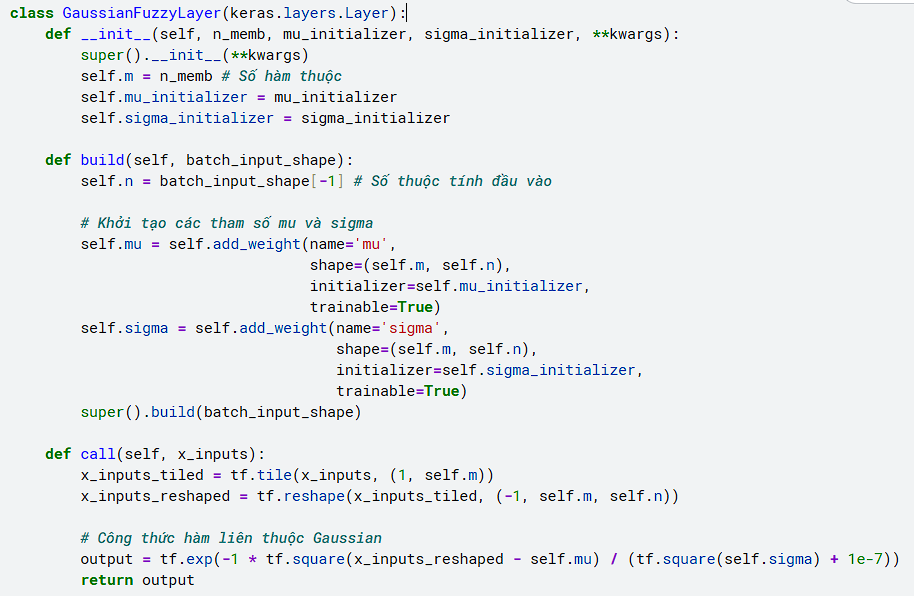
###### Các biểu đồ trực quan hóa theo hàm Gauss

Nhận xét:

* Đối với age và click: 3 đường cong hình chuông (dạng Gauss) được phân bổ khá đều và rõ ràng. Điều này cho thấy dữ liệu của hai thuộc tính này được phân bố tương đối tốt trên toàn dải giá trị. K-Means đã dễ dàng tìm thấy ba cụm tự nhiên, có thể tương ứng với các nhóm "thấp", "trung bình" và "cao". Đây là kết quả trực quan hóa lý tưởng.
* Đối với days\_since\_last\_interaction, purchase, add-to-cart, và favourite: Có một hoặc nhiều đường cong trông giống như những đường thẳng đứng (gai nhọn), đặc biệt là ở giá trị 0.0. Cho thấy rằng dữ liệu của các thuộc tính này bị chênh lệch rất nhiều. Ví dụ, với thuộc tính purchase (mua hàng), phần lớn khách hàng có giá trị là 0 (không mua hàng). Do đó, K-Means đã xác định một cụm rất lớn ngay tại giá trị 0, và độ lệch chuẩn (sigma) của cụm này gần như bằng 0, tạo ra một đường cong cực kỳ hẹp và cao. Các cụm còn lại được tìm thấy trong số ít các giá trị khác 0.

### Xây dựng lớp mờ hóa

Thiết kế lớp GaussianFuzzyLayer trên code:



###### Lớp mờ hóa - hàm Gauss

Khởi tạo (\_\_init\_\_): Khi tạo lớp, ta đưa số hàm thuộc mong muốn vào n\_memb (ở đây ta sẽ nhập là 3 theo phân tích ở mục 4.1.2), sau đó, ta khởi tạo giá trị ban đầu cho mu (tâm) và sigma (độ lệch chuẩn)

Khi xây dựng (build) mô hình, hàm này cần xác định số lượng thuộc tính đầu vào (self.n) và tạo ra hai ma trận trọng số là self.mu và self.sigma tương ứng với số hàm thuộc và số thuộc tính. Cả hai ma trận này đều được đánh dấu là trainable=True, cho phép quá trình lan truyền ngược (backpropagation) cập nhật chúng

Trong mỗi lượt dự đoán, ta thực hiện tính toán đầu ra

Bước 1: Điều chỉnh hình dạng đầu vào: Các dòng tf.tile và tf.reshape được dùng để sao chép và định hình lại tensor đầu vào. Mục đích là để có thể dễ dàng thực hiện phép tính giữa mỗi giá trị đầu vào và tất cả các hàm thuộc tương ứng của nó.

Bước 2: Áp dụng công thức Gauss:



x\_inputs\_reshaped - self.mu: Tính khoảng cách từ mỗi điểm dữ liệu đến tâm của mỗi hàm thuộc.

tf.square( ): Bình phương khoảng cách.

(tf.square(self.sigma) + 1e-7): Bình phương độ rộng (phương sai) (Số 1e-7 rất nhỏ được thêm vào để đảm bảo tính ổn định về mặt số học, tránh chia cho 0 nếu sigma tiến quá gần về 0 trong quá trình huấn luyện)

tf.exp( ): Tính hàm mũ để hoàn thành công thức.

Kết quả: Đầu ra của lớp là một tensor chứa các giá trị độ thuộc (từ 0 đến 1) của mỗi mẫu dữ liệu đối với mỗi hàm thuộc của từng thuộc tính. Tensor này sau đó sẽ được đưa vào các lớp tiếp theo của mạng ANFIS

### Xây dựng tập luật mờ

ANFIS sẽ tự động tạo ra một tập luật đầy đủ dựa trên tất cả các kết hợp có thể có từ các hàm liên thuộc đã định nghĩa ở phần trước và sử dụng phương pháp tổ hợp toàn phần để tạo ra luật (kết hợp mọi khả năng của các tập mờ từ tất cả các đầu vào).

Với 6 thuộc tính đầu vào (*age, days\_since\_last\_interaction, click, purchase, add-to-cart, favourite*) và mỗi thuộc tính có 3 tập mờ, tổng số luật mà ANFIS sẽ tự động tạo ra là: 3×3×3×3×3×3=36=729 luật

Nhiệm vụ của quá trình huấn luyện ANFIS là sử dụng dữ liệu (X\_train, y\_train) để học ra giá trị tối ưu cho tất cả đầu ra (THEN) của từng luật và đồng thời tinh chỉnh lại các hàm liên thuộc đã định nghĩa ban đầu

Thiết kế trên code:

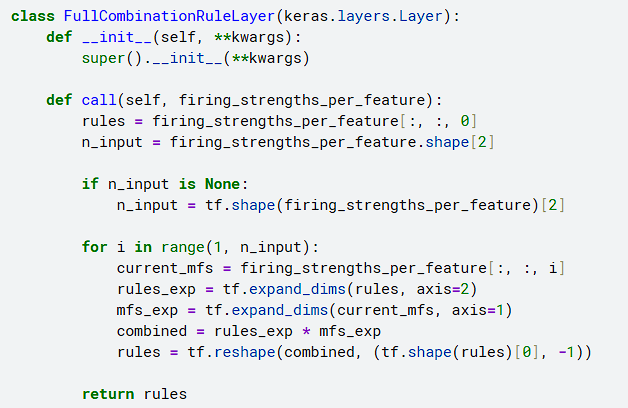
Ở đây, hàm FullCombinationRuleLayer() là Lớp Luật kết hợp (lớp 2), lớp này nhận các độ thuộc và tạo ra sức mạnh kích hoạt cho mỗi luật bằng cách lấy mọi tổ hợp có thể có của các hàm thuộc từ các đầu vào.

FullCombinationRuleLayer tạo ra các luật và tính toán độ kích hoạt (firing strength) cho mỗi luật.

Bên trong lớp FullCombinationRuleLayer, phép nhân (combined = rules\_exp \* mfs\_exp) được sử dụng làm toán tử T-norm để kết hợp các giá trị thuộc của các mệnh đề trong phần "NẾU" của một luật. Ví dụ, một luật có thể có dạng:

IF age is "Trẻ" AND days\_since\_last\_interaction is "Gần đây" AND ... THEN ...

Kết quả: Đầu ra của lớp này là một vector w có kích thước là 729, trong đó mỗi phần tử đại diện cho độ kích hoạt của một trong 729 luật đối với một mẫu dữ liệu đầu vào.



###### Code xây dựng tập luật

Trong đó:

Đầu vào của lớp này là một tensor 3D với kích thước (batch\_size, n\_mfs, n\_input)

batch\_size: Số lượng mẫu dữ liệu được xử lý cùng lúc.

n\_mfs: Số lượng hàm thành viên (tập mờ) cho mỗi biến đầu vào.

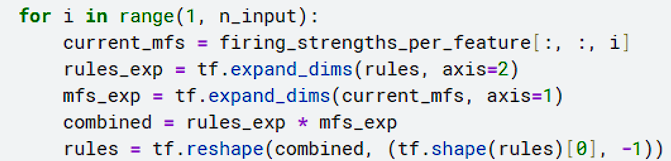
n\_input: Số lượng biến/đặc trưng đầu vào.

Giá trị trong tensor này là "độ kích hoạt" hay giá trị thành viên (membership value) của mỗi mẫu dữ liệu đối với từng tập mờ của từng biến đầu vào. Đây là kết quả từ lớp đầu tiên của ANFIS (lớp mờ hóa - Fuzzification Layer)

Khởi tạo luật (rule):



Quá trình bắt đầu bằng cách lấy độ kích hoạt của tất cả các tập mờ thuộc về biến đầu vào đầu tiên (i=0). rules lúc này có kích thước (batch\_size, n\_mfs)



current\_mfs = firing\_strengths\_per\_feature[:, :, i] ⇒ Lấy độ kích hoạt của các tập mờ cho biến đầu vào hiện tại.

rules\_exp = tf.expand\_dims(rules, axis=2) ⇒ Mở rộng tensor rules hiện tại bằng cách thêm một chiều ở cuối. Nếu rules có kích thước (batch\_size, k), nó sẽ trở thành (batch\_size, k, 1).

mfs\_exp = tf.expand\_dims(current\_mfs, axis=1) ⇒ Mở rộng tensor current\_mfs bằng cách thêm một chiều ở giữa. Nếu current\_mfs có kích thước (batch\_size, n\_mfs), nó sẽ trở thành (batch\_size, 1, n\_mfs).

combined = rules\_exp \* mfs\_exp ⇒ Nhờ vào cơ chế broadcasting của TensorFlow, phép nhân này sẽ kết hợp mỗi luật trong rules\_exp với mỗi tập mờ trong mfs\_exp.

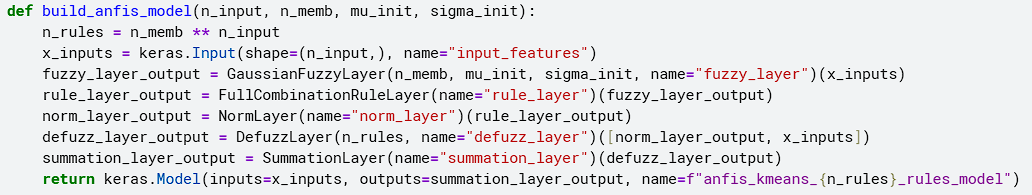
Ví dụ: Nếu rules\_exp có kích thước (batch, k, 1) và mfs\_exp có (batch, 1, n\_mfs), kết quả combined sẽ có kích thước (batch, k, n\_mfs). Nó thực hiện phép nhân Descartes (Cartesian product) giữa các độ kích hoạt.

rules = tf.reshape(combined, (tf.shape(rules)[0], -1)) ⇒ Làm phẳng kết quả combined thành một tensor 2D. Số luật mới sẽ là k \* n\_mfs. Tensor rules giờ đây chứa độ kích hoạt của tất cả các luật đã được kết hợp từ biến đầu vào 0 đến i.

Đầu ra: Sau khi vòng lặp kết thúc, tensor rules cuối cùng sẽ có kích thước (batch\_size, total\_rules), trong đó total\_rules = n\_mfs ^ n\_input. Mỗi giá trị trong tensor này là độ kích hoạt của một luật mờ hoàn chỉnh

### Khởi tạo ANFIS

Mô hình ANFIS hoàn chỉnh được tạo nên từ 5 lớp như đã nêu ở cơ sở lý thuyết



###### Mô hình ANFIS

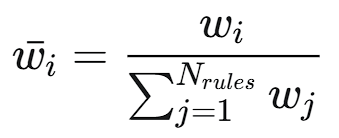
Trong đó:

* Lớp 1: fuzzy\_layer\_output là lớp mờ hóa (GaussianFuzzyLayer)

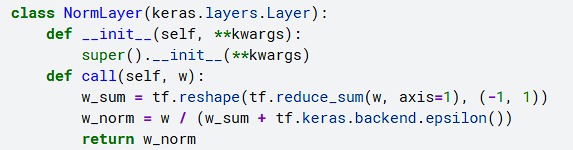
Lớp này nhận đầu vào và tính toán độ thuộc bằng hàm Gaussian.

Các tham số mu (tâm) và sigma (độ rộng) được tự động học

* Lớp 2: rule\_layer\_output là lớp luật (FullCombinationRuleLayer) nhận các giá trị thuộc và tính toán độ kích hoạt w cho mỗi luật trong số 729 luật (đã trình bày ở mục 4.1.3)
* Lớp 3: norm\_layer\_output là lớp chuẩn hóa (NormLayer) nhận vector độ kích hoạt w và chuẩn hóa nó bằng cách chia mỗi giá trị cho tổng của tất cả các giá trị

 *(4-2)*

Kết quả là các độ kích hoạt đã được chuẩn hóa w\_norm



###### Lớp chuẩn hóa

* Lớp 4: defuzz\_layer\_output là lớp giải mờ (DefuzzLayer)



###### Lớp giải mờ

Lớp này tạo ra một ma trận trọng số consequence\_params có kích thước (n\_input + 1, n\_rules), tức là (6+1, 729). Mỗi cột của ma trận này tương ứng với các tham số kết luận (p\_0, p\_1, ..., p\_6) cho một luật mờ theo mô hình Sugeno bậc nhất

Đối với mỗi luật, lớp này tính toán giá trị kết luận f\_i là một hàm tuyến tính của các đầu vào:

 *(4-3)*

Đầu ra của lớp là tích của độ kích hoạt đã chuẩn hóa và kết luận của mỗi luật:

* Lớp 5: summation\_layer\_output là lớp tổng hợp (SummationLayer): Lớp này tính tổng kết quả của tất cả các luật để đưa ra một giá trị đầu ra duy nhất



###### Lớp tổng hợp

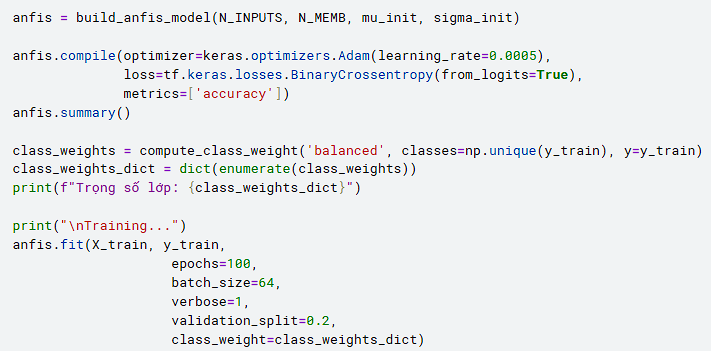
### Chạy mô hình

Khởi tạo các giá trị ban đầu



###### Khởi tạo mô hình

Chạy mô hình với số lần lặp là 100, tốc độ học là 0.005



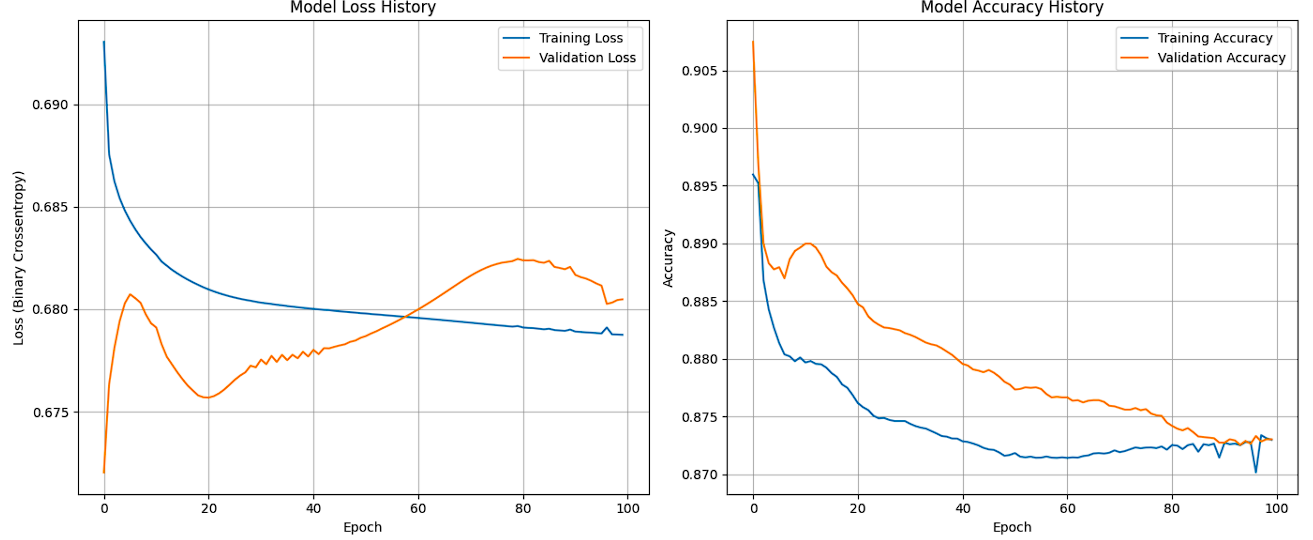
###### Chạy mô hình

BCEWithLogitsLoss: Hàm lỗi này kết hợp hàm kích hoạt Sigmoid và hàm lỗi Binary Cross-Entropy (BCE) thành một lớp duy nhất

Do phần lớn dữ liệu có đầu ra là “0” (Không trung thành), ta sử dụng class\_weight gán một trọng số cao hơn cho lớp thiểu số ('1') và một trọng số thấp hơn cho lớp đa số ('0') để giảm mất cân bằng dữ liệu

## Đánh giá và tinh chỉnh mô hình

### Kết quả thực nghiệm ban đầu



###### Kết quả thực hiện

Biểu đồ biểu diễn độ mất mát (bên trái):

* Training Loss (Màu xanh dương): Đường này liên tục giảm trong suốt quá trình huấn luyện. Điều này cho thấy mô hình đang học tốt từ tập dữ liệu huấn luyện và tối ưu hóa thành công hàm mất mát ⇒ Đây là một dấu hiệu tốt về khả năng học của mô hình.
* Validation Loss (Màu cam): Đường này ban đầu giảm, đạt đến điểm cực tiểu vào khoảng epoch thứ 20-25. Sau điểm này, Validation Loss bắt đầu tăng trở lại

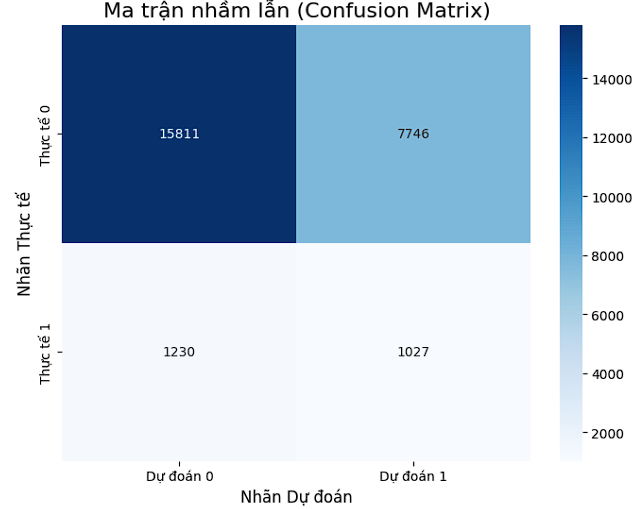
⇒ Vấn đề: Sự phân kỳ giữa hai đường loss là dấu hiệu kinh điển của overfitting. Trong khi mô hình ngày càng "giỏi" hơn trên dữ liệu huấn luyện (loss giảm), nó lại ngày càng "tệ" đi trên dữ liệu xác thực (loss tăng). Mô hình bắt đầu học cả nhiễu và các chi tiết không quan trọng của tập huấn luyện.

Biểu đồ độ chính xác (bên phải)

* Training Accuracy (Màu xanh dương): Đường này tăng dần và ổn định ở mức cao. Điều này xác nhận rằng mô hình đang khớp tốt với dữ liệu huấn luyện
* Validation Accuracy (Màu cam): Đường này ban đầu tăng, đạt đỉnh rất sớm (khoảng epoch 5-10) rồi sau đó giảm dần

⇒ Vấn đề: Tương tự như biểu đồ loss, sự chênh lệch ngày càng lớn giữa độ chính xác huấn luyện và độ chính xác xác thực khẳng định lại vấn đề overfitting. Mô hình mất dần khả năng tổng quát hóa, dẫn đến độ chính xác trên dữ liệu mới ngày càng giảm

Ma trận nhầm lẫn:



###### Ma trận nhầm lẫn

Với True Negatives (TN) = 15811, False Positives (FP) = 7746, False Negatives (FN) = 1230, True Positives (TP) = 1027

Ta có

Accuracy (Độ chính xác tổng thể):

(TP + TN) / (Tổng số) = (1027 + 15811) / 25814 = 65.2%

Precision (Độ chính xác cho lớp Positive):

TP / (TP + FP) = 1027 / (1027 + 7746) = 11.7% (Rất thấp)

⇒ Precision quá thấp do “0” chiếm đa số trong tập dữ liệu. Điều này dẫn đến việc khi mô hình dự đoán một trường hợp là "Positive", có tới 88.3% khả năng mô hình dự đoán sai

Recall (Độ phủ / Độ nhạy):

TP / (TP + FN) = 1027 / (1027 + 1230) = 45.5% (Thấp)

⇒ Mô hình đã bỏ sót hơn một nửa (54.5%) các trường hợp "Positive" thực tế

F1-Score:

2 \* (Precision \* Recall) / (Precision + Recall) = 18.6% (Rất thấp)

⇒ Chỉ số này là trung bình điều hòa của Precision và Recall, cho thấy sự cân bằng yếu kém giữa hai yếu tố này. Một F1-Score thấp như vậy chứng tỏ mô hình không hiệu quả trong việc nhận diện lớp Positive

Kết luận: Mô hình này không thể sử dụng trong thực tế vì nó dự đoán lớp Positive rất thiếu chính xác và bỏ sót quá nhiều trường hợp quan trọng

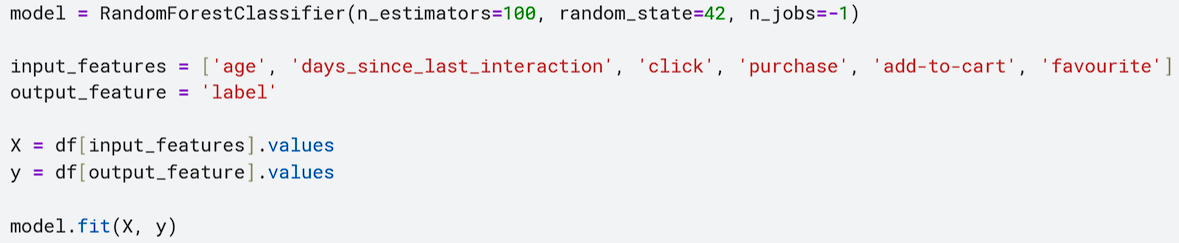
### Tinh chỉnh mô hình

#### Giảm số lượng thuộc tính

Như đã thấy theo kết quả thực nghiệm trước đó, mô hình ANFIS với quá nhiều thuộc tính đầu vào (dẫn đến số luật lớn) hoạt động không hiệu quả. Ta cân nhắc tới việc giảm thiểu số thuộc tính đầu vào.

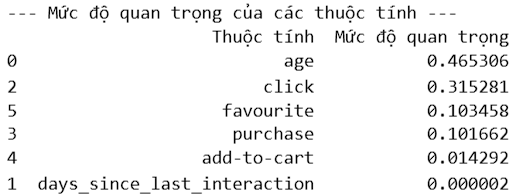
Random Forest là một mô hình học máy tập hợp, xây dựng hàng trăm cây quyết định khác nhau. Nó có khả năng đo lường mức độ quan trọng của từng thuộc tính đầu vào một cách khách quan bằng cách tính toán mức độ đóng góp trung bình của mỗi thuộc tính vào việc làm cho các cây quyết định trở nên chính xác hơn (thông qua chỉ số Suy giảm Độ ô nhiễm Gini)

Dựa vào đó, ta có thể loại bỏ các thuộc tính gây nhiễu và xây dựng lại một mô hình ANFIS tinh gọn hơn, tập trung vào những thông tin có giá trị nhất, với hy vọng cải thiện hiệu suất phân loại.

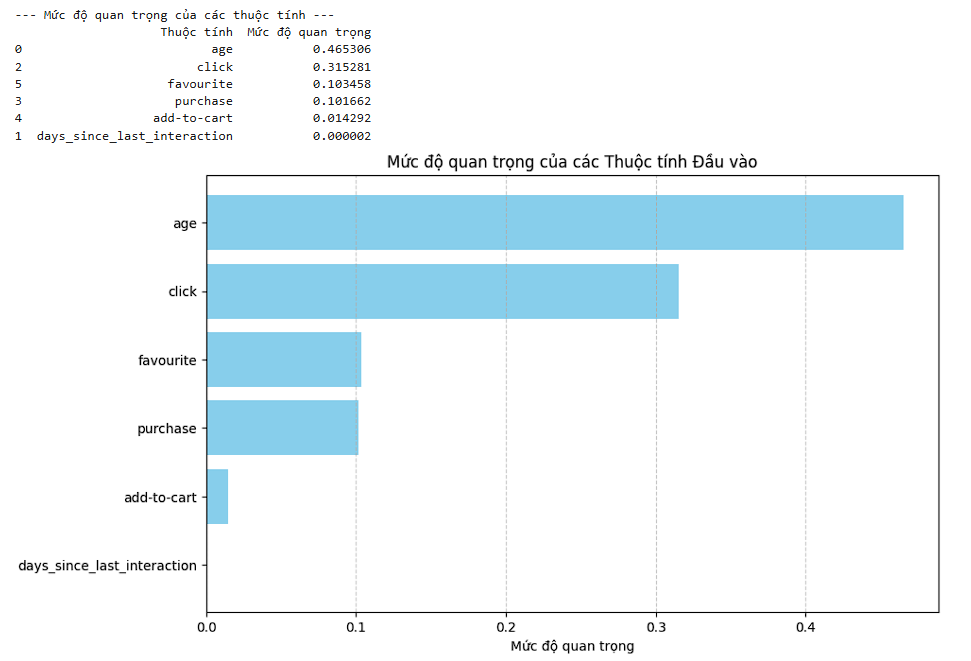


###### Đưa 6 thuộc tính vào RandomForest

Kết quả thu được:



Trực quan hóa trên biểu đồ:



###### Biểu đồ độ quan trọng của các thuộc tính

Kết quả này cho thấy một sự phân cấp rất rõ ràng về mức độ ảnh hưởng của các thuộc tính đến việc dự đoán của mô hình. Chúng ta có thể chia chúng thành ba nhóm chính:

* Nhóm Quan trọng Nhất: *age* = 0.465, *click* = 0.315

Hai thuộc tính này chiếm tổng cộng gần 78% mức độ quan trọng của toàn bộ mô hình. Điều này có nghĩa là khi các cây quyết định trong Random Forest được xây dựng, *age* và *click* là hai yếu tố được sử dụng nhiều nhất và hiệu quả nhất để phân chia dữ liệu và đưa ra dự đoán

*age* là thuộc tính có ảnh hưởng lớn nhất. Điều này cho thấy các nhóm tuổi khác nhau có hành vi hoàn toàn khác biệt trong bối cảnh bài toán

*click* là yếu tố quan trọng thứ hai, cho thấy hành động nhấp chuột là một tín hiệu mạnh mẽ thể hiện sự quan tâm hoặc ý định của người dùng

* Nhóm Quan trọng vừa: *favourite* = 0.103, *purchase* = 0.101

Hai thuộc tính này có mức độ quan trọng gần như nhau và đóng góp một phần đáng kể vào mô hình, nhưng kém xa so với hai thuộc tính top đầu

Hành động "thêm vào yêu thích" và "lịch sử mua hàng" rõ ràng là những chỉ báo hữu ích về hành vi của người dùng, nhưng có thể chúng không mang tính quyết định bằng các yếu tố nhân khẩu học cơ bản như tuổi tác hoặc hành vi thể hiện sự quan tâm ban đầu như click

* Nhóm Ít Quan trọng: *add-to-cart* = 0.014, *days\_since\_last\_interaction* = 0.000002

*add-to-cart* có mức độ quan trọng rất thấp. Điều này hơi bất ngờ nhưng có thể được giải thích rằng thông tin từ hành động này có thể đã được chứa đựng trong click và purchase

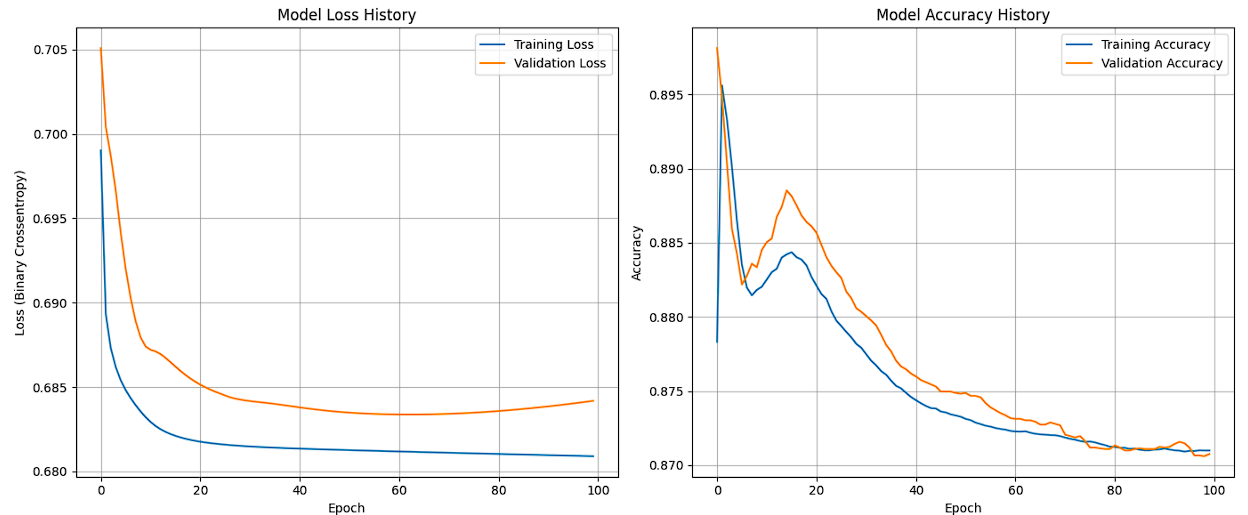
*days\_since\_last\_interaction* có mức độ quan trọng gần như bằng không. Đây là một phát hiện rất giá trị. Nó cho thấy mô hình không tìm thấy mối liên hệ nào giữa việc một người dùng đã không tương tác trong bao lâu và kết quả dự đoán. Nói cách khác, "sự mới mẻ" của tương tác không phải là yếu tố quyết định trong trường hợp này

Như vậy, ta sẽ giảm số lượng thuộc tính đưa vào mô hình bằng cách loại nhỏ 2 thuộc tính ít quan trọng nhất là *add-to-cart* và *days\_since\_last\_interaction*

**

#### Kết quả sau khi tinh chỉnh và đánh giá

Sau khi điều chỉnh số lượng thuộc tính đầu vào, ta thu được:



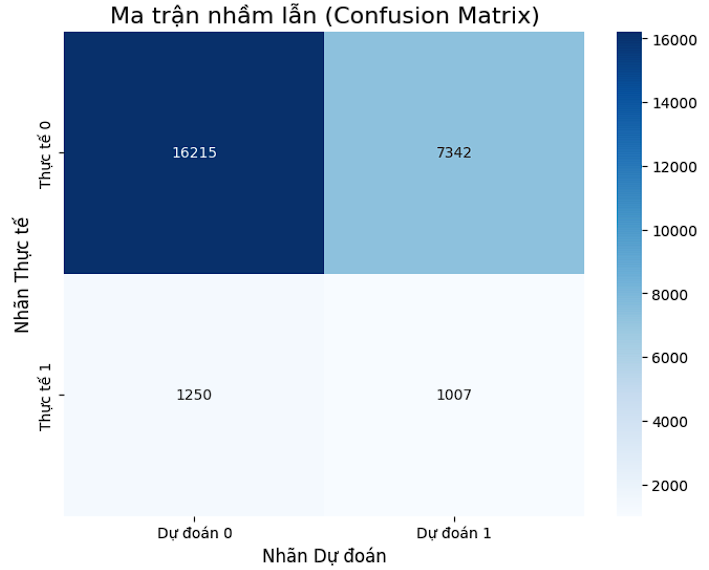
###### Kết quả sau khi tinh chỉnh

Biểu đồ biểu diễn độ mất mát (bên trái) có sự cải thiện rõ rệt nhất so với trước đó

Training Loss (xanh dương) và Validation Loss (cam) bây giờ cùng nhau đi xuống và hội tụ. Đường Validation Loss không còn tăng vọt lên ở các epoch sau nữa. Độ mất mát (loss) giảm cũng cho thấy kết quả khả quan hơn trong kết quả dự đoán của mô hình

Có thể thấy việc loại bỏ thuộc tính nhiễu đã giúp mô hình tập trung vào những tín hiệu quan trọng.

Với biểu đồ Accuracy (bên phải): Hai đường Training và Validation Accuracy bây giờ rất gần nhau, có thể thấy rằng mô hình đã bớt overfitting



###### Ma trận nhầm lẫn

Tuy nhiên, vẫn có độ chênh lệch lớn giữa giá trị “0” và “1’ trong dữ liệu:

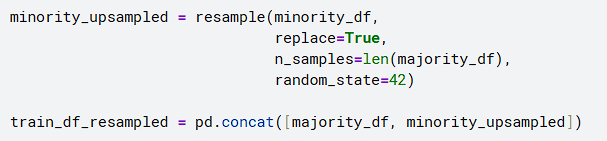
Lớp 0 (Negative): 16215 + 7342 = 23,557 mẫu

Lớp 1 (Positive): 1250 + 1007 = 2,257 mẫu

Lớp 0 xuất hiện nhiều hơn lớp 1 tới hơn 10 lần, điều này dẫn đến việc mô hình không học đủ các đặc trưng của lớp 1 và sẽ bỏ sót rất nhiều trường hợp Positive (Recall thấp) và khi nó dự đoán là Positive thì lại sai rất nhiều (Precision thấp)

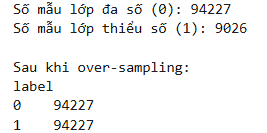
#### Tăng cường mẫu và đánh giá

Over-sampling là kỹ thuật giải quyết vấn đề này bằng cách tăng số lượng mẫu trong lớp thiểu số (lớp ít hơn). Mục tiêu là làm cho tập dữ liệu huấn luyện trở nên "cân bằng" hơn, để mô hình có cơ hội học hỏi đồng đều từ tất cả các lớp. Ở đây ta sử dụng Random Over-sampling, “sao chép” những dòng có *label* thiểu số (“1”) để cân bằng với “0”



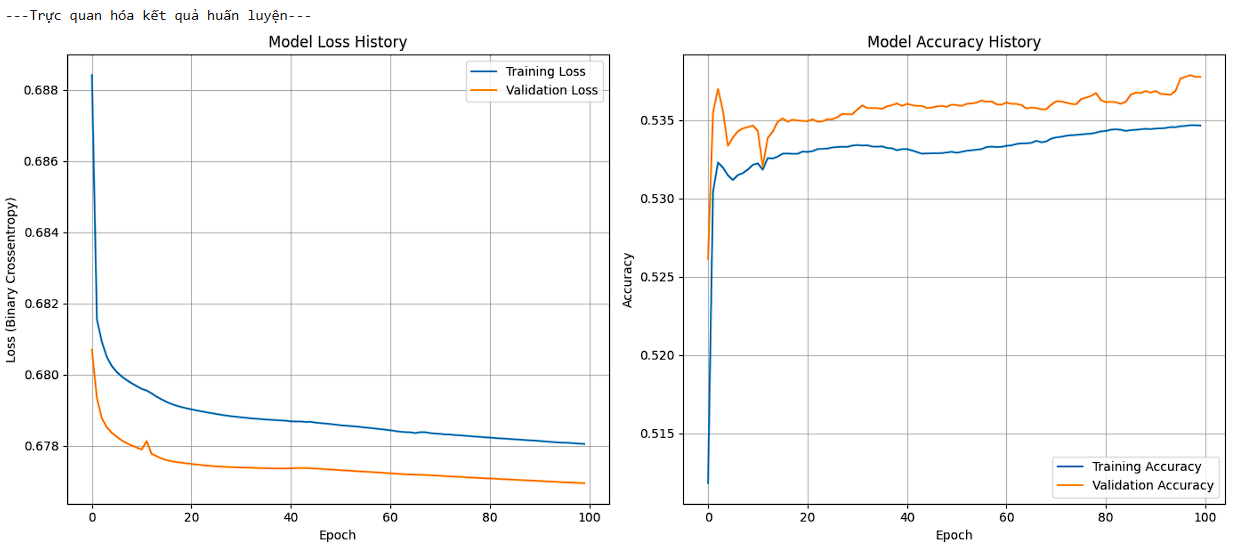
###### Thêm sample

Kết quả thu được:



###### Kết quả sau khi thực hiện over-sampling

Kết quả sau khi mô hình áp dụng over-sampling:



###### Kết quả sau khi thực hiện over-sampling

Về Hàm Mất mát (Loss):

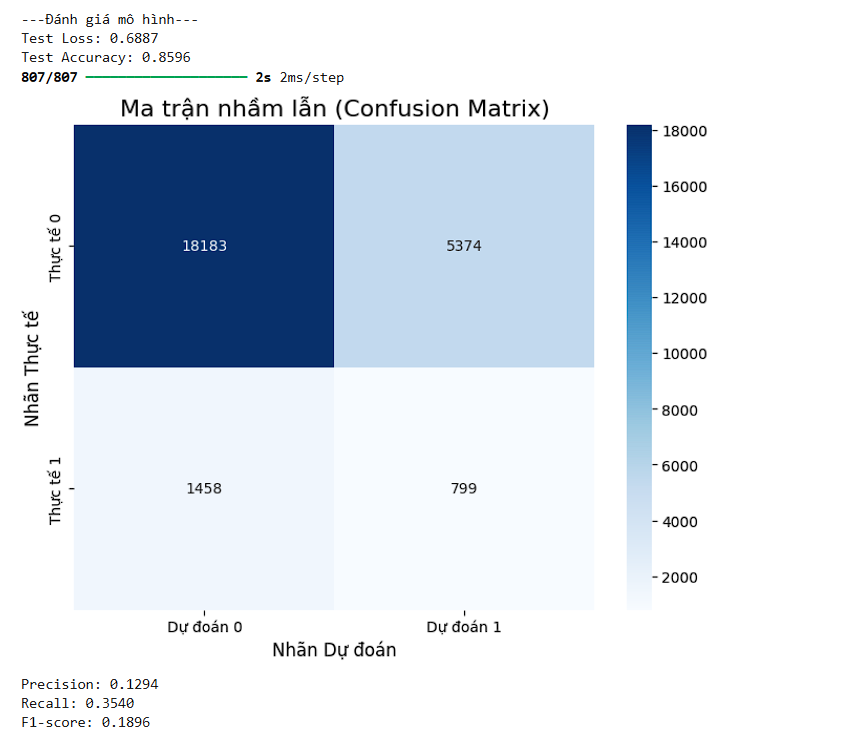
* Biểu đồ "Model Loss History" cho thấy một xu hướng học tập rất tích cực. Cả đường Training Loss (tổn thất trên tập huấn luyện) và Validation Loss (tổn thất trên tập kiểm định) đều cùng nhau giảm dần và hội tụ
* Đường Validation Loss duy trì ở mức thấp và ổn định, thậm chí còn thấp hơn Training Loss ở các epoch cuối. Đây là minh chứng rõ ràng cho thấy mô hình không bị học quá khớp (overfitting). Việc loại bỏ các thuộc tính nhiễu và cân bằng dữ liệu đã giúp mô hình tổng quát hóa tốt hơn.

Về Độ chính xác (Accuracy):

* Tương tự, biểu đồ "Model Accuracy History" cho thấy hai đường Training và Validation Accuracy có xu hướng gần nhau và cùng tăng trưởng.
* Độ chính xác trên tập kiểm định (Validation Accuracy) duy trì ở mức cao và ổn định, củng cố thêm cho kết luận rằng mô hình học một cách bền vững.

Đánh giá hiệu suất mô hình

* Accuracy (Độ chính xác) = 86.0%: Chỉ số này cao nhưng có thể gây hiểu lầm do sự mất cân bằng lớn trong tập test (lớp "không trung thành" chiếm đa số). Nó không phản ánh đúng khả năng nhận diện lớp thiểu số.
* Precision (Độ chính xác) = 12.9%: Khi mô hình dự đoán một khách hàng là "trung thành", nó chỉ đúng trong khoảng 13% các trường hợp.
* Recall = 35.4%: Mô hình có khả năng "bắt" được khoảng 35% trong tổng số các khách hàng thực sự trung thành.
* F1-score = 19.0%: Là trung bình điều hòa của Precision và Recall, cho thấy một sự cân bằng còn khiêm tốn giữa hai chỉ số



###### Kết quả sau khi thực hiện over-sampling

Về Ma trận nhầm lẫn:

* True Positives (TP - Đúng Positives): 799 – Mô hình đã nhận diện chính xác 799 khách hàng trung thành.
* False Negatives (FN - Sai Negatives): 1458 – Mô hình đã bỏ sót 1458 khách hàng trung thành
* False Positives (FP - Sai Positives): 5374 – Mô hình đã dự đoán nhầm 5374 khách hàng không trung thành thành khách hàng trung thành
* True Negatives (TN - Đúng Negatives): 18183 – Mô hình nhận diện rất tốt các khách hàng không trung thành.

#### Kết luận

Mô hình ANFIS sau khi được tối ưu hóa đã cho thấy sự ổn định cao và khả năng tổng quát hóa tốt, không còn hiện tượng overfitting. Các kỹ thuật lựa chọn đặc trưng và over-sampling đã chứng minh được hiệu quả rõ rệt trong việc ổn định quá trình học. Tuy nhiên, khả năng dự đoán chính xác lớp khách hàng trung thành (lớp thiểu số) vẫn còn là một thách thức, với chỉ số F1-score cao nhất đạt 19%.

# TRIỂN KHAI MÔ HÌNH

Để đưa mô hình ANFIS vào ứng dụng thực tế, một ứng dụng web dưới dạng dashboard đã được xây dựng bằng framework Flask của Python. Mục tiêu của ứng dụng là thu thập dữ liệu từ trang web bán hàng theo thời gian thực và chuyển nó thành dạng csv và nhận lại kết quả dự đoán về mức độ trung thành một cách nhanh chóng. Quy trình này đảm bảo dữ liệu mới, không có trong tập huấn luyện, được xử lý một cách nhất quán để mang lại kết quả đáng tin cậy.

Vì thời gian hạn chế, nên trong bài báo cáo chỉ dừng lại ở việc sử dụng file csv có sẵn.

Luồng hoạt động của hệ thống được thực hiện như sau:

**Bước 1:** Giao diện người dùng và nhập liệu

Giao diện: Người dùng tương tác với một giao diện web (dashboard) đơn giản, nhấn nút "Dự đoán". Dữ liệu sẽ được gửi đến backend của ứng dụng Flask để xử lý.

**Bước 2:** Xử lý Backend và Tiền xử lý dữ liệu

Khi nhận được yêu cầu từ giao diện, ứng dụng Flask sẽ thực hiện vai trò điều phối, bắt đầu bằng việc chuẩn bị dữ liệu.

Quá trình xử lý:

* Nhận dữ liệu: Tiếp nhận dữ liệu đầu vào dưới dạng csv.
* Tải quy trình làm sạch dữ liệu: Ứng dụng tải lại quy trình làm sạch dữ liệu để nhất quán trong cách xử lý dữ liệu.

**Bước 3:** Thực thi Mô hình và Đưa ra Dự đoán (Model Execution)

Sau khi dữ liệu đã được tiền xử lý, nó sẽ được đưa vào mô hình ANFIS để tính toán.

Quá trình dự đoán:

* Tải mô hình: Ứng dụng tải lại kiến trúc và trọng số của mô hình ANFIS.
* Dự đoán Logit: Dữ liệu đã chuẩn hóa được đưa vào mô hình thông qua phương thức .predict( ), trả về một giá trị đầu ra dạng logit.

**Bước 4:** Xử lý Kết quả và Trả về Giao diện (Post-processing & Display)

Giá trị logit thô từ mô hình được chuyển đổi thành một kết quả có ý nghĩa và dễ hiểu cho người dùng.

Quá trình xử lý: Áp dụng hàm Sigmoid để biến đổi logit thành một xác suất (từ 0 đến 1), thể hiện mức độ tự tin của mô hình rằng khách hàng thuộc lớp "trung thành".

Hiển thị kết quả:

* Nếu xác suất > 0.5410, ứng dụng sẽ trả về kết quả là "Trung thành" (Nhãn 1).
* Nếu xác suất <= 0.5410, kết quả sẽ là "Không trung thành" (Nhãn 0).

Kết quả cuối cùng này, cùng với xác suất cụ thể, sẽ được gửi ngược lại giao diện dashboard và hiển thị một cách trực quan cho người dùng cuối.

**TỔNG KẾT**

Qua quá trình thực hiện đề tài *“Ứng dụng mạng nơ ron mờ trong dự đoán khách hàng trung thành dựa trên đặc điểm nhân khẩu học và hành vi”*, nhóm chúng em đã đạt được những mục tiêu đề ra và thu được nhiều kiến thức, kinh nghiệm quý báu. Đề tài không chỉ là bài tập ứng dụng mà còn là cơ hội để chúng em tiếp cận một quy trình hoàn chỉnh trong lĩnh vực khoa học dữ liệu, từ việc xử lý dữ liệu thô đến việc xây dựng, tối ưu và đánh giá một mô hình học máy phức tạp - mạng nơron mờ (mạng ANFIS).

Chúng em cũng đã học được cách phân chia công việc một cách logic, cùng nhau giải quyết các vấn đề phát sinh để đạt được mục tiêu chung trong quá trình làm việc nhóm. Những điều này không chỉ giúp chúng em hoàn thành tốt đề tài mà còn là kinh nghiệm quan trọng cho các dự án và công việc trong tương lai.

# TÀI LIỆU THAM KHẢO

[1] Kumar, V., & Reinartz, W. (2016). Creating enduring customer value. *Journal of Marketing, 80*(6), 36–68. https://doi.org/10.1509/jm.15.0414

[2] Trần, M. T. (n.d.). *Slide bài giảng Tính toán mềm*. Đại học Thủy lợi.

[3] Siddique, N., & Adeli, H. (2013). *Computational intelligence: Synergies of fuzzy logic, neural networks and evolutionary computing*. Wiley.

[4] Rusli, N. M., Ibrahim, Z., & Janor, R. M. (2008). Predicting students’ academic achievement: Comparison between logistic regression, artificial neural network, and Neuro-fuzzy. In *Proceedings of the IEEE International Symposium on Information Technology* (pp. 1–6). IEEE. https://doi.org/10.1109/ISIT.2008.4631945

[5] Ngân, T. T., Thủy, Đ. T., & Dương, M. T. (2022). Ứng dụng mạng nơron mờ đa tác động trong tư vấn học sinh trung học phổ thông dựa trên quá trình học tập. *Tạp chí Khoa học và Công nghệ – Đại học Thái Nguyên, 227*(06), 25–31.

[6] Etezadi, H. (n.d.). *Using ANFIS for capacitive deionization* [Kaggle notebook]. Kaggle.<https://www.kaggle.com/code/hamedetezadi/using-anfis-for-capacitive-deionization/notebook>

[7] Elazab, A., Hu, Q., Wang, M., Wu, S., & Lei, B. (2020). An optimized neuro-fuzzy classification model for breast cancer diagnosis. *Computers in Biology and Medicine, 126*, 104036

[8] Power, J. (n.d.). *membership.py – ANFIS implementation in PyTorch* [Python script]. GitHub.<https://github.com/jfpower/anfis-pytorch/blob/master/membership.py>

[9] Ariff, Y. (n.d.). *Fuzzy C-means using scikit-fuzzy for image clustering* [Jupyter notebook]. GitHub.<https://github.com/ariffyasri/fuzzy-c-means/blob/master/fuzzy-c-means-scikit-fuzzy-image.ipynb>

[10] Khánh, P. Đ. (2020, February 17). *Imbalanced data: Cách xử lý dữ liệu mất cân bằng trong học máy* [Blog post]. <https://phamdinhkhanh.github.io/2020/02/17/ImbalancedData.html>

# PHỤ LỤC

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

userlog = pd.read\_csv('/kaggle/input/dazone-2025/user\_log.csv')

userinfo = pd.read\_csv('/kaggle/input/dazone-2025/user\_info.csv')

train = pd.read\_csv('/kaggle/input/dazone-2025/train.csv')

test = pd.read\_csv('/kaggle/input/dazone-2025/test.csv')

In [181]:

*#Kiểm tra sơ bộ dữ liệu*

print("Thông tin User Info: ")

print(userinfo.info())

print("Thông tin User Log: ")

print(userlog.info())

Thông tin User Info:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 424170 entries, 0 to 424169

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 user\_id 424170 non-null int64

1 age 328803 non-null float64

2 sex 418652 non-null object

3 phone 402962 non-null object

4 job 402962 non-null object

5 carrier 402962 non-null object

6 marital\_status 423813 non-null object

dtypes: float64(1), int64(1), object(5)

memory usage: 22.7+ MB

None

Thông tin User Log:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 54925330 entries, 0 to 54925329

Data columns (total 7 columns):

# Column Dtype

--- ------ -----

0 user\_id int64

1 item\_id int64

2 cat\_id int64

3 brand\_id float64

4 merchant\_id int64

5 action object

6 datetime object

dtypes: float64(1), int64(4), object(2)

memory usage: 2.9+ GB

None

In [182]:

*#Kiểm tra chi tiết bộ User Log*

print ("5 dòng đầu dữ liệu: ")

print(userlog.head())

5 dòng đầu dữ liệu:

user\_id item\_id cat\_id brand\_id merchant\_id action datetime

0 328862 323294 833 2661.0 2882 click 2024-08-29

1 328862 844400 1271 2661.0 2882 click 2024-08-29

2 328862 575153 1271 2661.0 2882 click 2024-08-29

3 328862 996875 1271 2661.0 2882 click 2024-08-29

4 328862 1086186 1271 1049.0 1253 click 2024-08-29

In [183]:

*#Chuyển kiểu của cột datetime từ string -> datetime*

userlog['datetime'] = pd.to\_datetime(userlog['datetime'], format='%Y-%m-**%d**')

print(userlog.info())

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 54925330 entries, 0 to 54925329

Data columns (total 7 columns):

# Column Dtype

--- ------ -----

0 user\_id int64

1 item\_id int64

2 cat\_id int64

3 brand\_id float64

4 merchant\_id int64

5 action object

6 datetime datetime64[ns]

dtypes: datetime64[ns](1), float64(1), int64(4), object(1)

memory usage: 2.9+ GB

None

In [184]:

*# Bảng đếm hành vi*

log\_agg = userlog.groupby(['user\_id', 'merchant\_id', 'action'])['item\_id'] \

.count() \

.unstack(fill\_value=0) \

.reset\_index()

*# Bảng thời gian tương tác gần nhất*

recent\_time = userlog.groupby(['user\_id', 'merchant\_id'])['datetime'].max().reset\_index()

*# Gộp vào bảng hành vi*

log\_agg = pd.merge(log\_agg, recent\_time, on=['user\_id', 'merchant\_id'], how='left')

*## Kết hợp hành vi với label*

data = pd.merge(train, log\_agg, on=['user\_id', 'merchant\_id'], how='left')

*# Kết hợp thêm đặc điểm nhân khẩu học*

data = pd.merge(data, userinfo, on='user\_id', how='left')

1. Khám phá dữ liệu sau kết hợp

In [186]:

*#Xem thông tin tổng quan về dữ liệu*

print(data.info())

print("--------------------------------------")

*# xem mô tả*

print(data.describe())

print("--------------------------------------")

*#Xem kích thước dữ liệu*

print(data.shape)

print("--------------------------------------")

*#Kiểm tra tổng giá trị null theo từng cột*

print(data.isnull().sum())

print("--------------------------------------")

*#Xem 5 dòng đầu dữ liệu*

print(data.head())

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 233782 entries, 0 to 233781

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 user\_id 228192 non-null float64

1 merchant\_id 228284 non-null float64

2 label 233782 non-null float64

3 add-to-cart 221145 non-null float64

4 click 221145 non-null float64

5 favourite 221145 non-null float64

6 purchase 221145 non-null float64

7 datetime 221145 non-null datetime64[ns]

8 age 175677 non-null float64

9 sex 221542 non-null object

10 phone 213119 non-null object

11 job 213221 non-null object

12 carrier 213216 non-null object

13 marital\_status 224338 non-null object

dtypes: datetime64[ns](1), float64(8), object(5)

memory usage: 25.0+ MB

None

--------------------------------------

user\_id merchant\_id label add-to-cart \

count 228192.000000 228284.000000 233782.000000 221145.000000

mean 215325.445888 2534.346529 0.120146 0.023817

min 1.000000 2.000000 -1.000000 0.000000

25% 107427.250000 1318.000000 0.000000 0.000000

50% 215538.000000 2482.000000 0.000000 0.000000

75% 322423.250000 3891.000000 0.000000 0.000000

max 426778.000000 4993.000000 1.000000 13.000000

std 124210.967345 1453.292964 0.382867 0.212005

click favourite purchase \

count 221145.000000 221145.000000 221145.000000

mean 9.335603 0.396183 1.346967

min 0.000000 0.000000 1.000000

25% 2.000000 0.000000 1.000000

50% 4.000000 0.000000 1.000000

75% 10.000000 0.000000 1.000000

max 3917.000000 107.000000 10.000000

std 24.626155 1.454088 0.872070

datetime age

count 221145 175677.000000

mean 2024-11-11 00:00:01.172081664 32.091452

min 2024-11-11 00:00:00 -1.000000

25% 2024-11-11 00:00:00 26.000000

50% 2024-11-11 00:00:00 30.000000

75% 2024-11-11 00:00:00 35.000000

max 2024-11-12 00:00:00 999.000000

std NaN 31.915003

--------------------------------------

(233782, 14)

--------------------------------------

user\_id 5590

merchant\_id 5498

label 0

add-to-cart 12637

click 12637

favourite 12637

purchase 12637

datetime 12637

age 58105

sex 12240

phone 20663

job 20561

carrier 20566

marital\_status 9444

dtype: int64

--------------------------------------

user\_id merchant\_id label add-to-cart click favourite purchase \

0 393595.0 794.0 0.0 0.0 9.0 2.0 2.0

1 287580.0 2447.0 0.0 0.0 5.0 0.0 1.0

2 203618.0 598.0 0.0 0.0 6.0 0.0 2.0

3 143560.0 2030.0 0.0 0.0 1.0 0.0 1.0

4 324173.0 1875.0 0.0 0.0 48.0 0.0 1.0

datetime age sex phone job \

0 2024-11-11 NaN girl \*\*\*\*\*\*2322 Nhân viên nhà hàng

1 2024-11-11 NaN she \*\*\*\*\*\*7298 Thợ xây

2 2024-11-11 26.0 she \*\*\*\*\*\*5348 Kỹ sư xây dựng

3 2024-11-11 NaN woman \*\*\*\*\*8063 Chuyên viên tư vấn tuyển sinh

4 2024-11-11 32.0 F \*\*\*\*\*\*\*2947 NaN

carrier marital\_status

0 viettel ly hôn

1 NaN Married

2 viettel Married

3 mobiphone ly hôn

4 mobiphone cưới

/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1458: RuntimeWarning: invalid value encountered in greater

has\_large\_values = (abs\_vals > 1e6).any()

/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1459: RuntimeWarning: invalid value encountered in less

has\_small\_values = ((abs\_vals < 10 \*\* (-self.digits)) & (abs\_vals > 0)).any()

/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1459: RuntimeWarning: invalid value encountered in greater

has\_small\_values = ((abs\_vals < 10 \*\* (-self.digits)) & (abs\_vals > 0)).any()

1. Làm sạch dữ liệu

3.1. Xóa tất cả các giá trị null

In [187]:

data = data.dropna()

print(data.shape)

(146370, 14)

3.2. Loại bỏ cột phone vì không cần thiết

In [188]:

data = data.drop(columns=['phone'])

print(data.shape)

print(data.columns)

(146370, 13)

Index(['user\_id', 'merchant\_id', 'label', 'add-to-cart', 'click', 'favourite',

'purchase', 'datetime', 'age', 'sex', 'job', 'carrier',

'marital\_status'],

dtype='object')

3.3. Gom nhóm các giá trị trong các cột dạng chuỗi

In [189]:

*#Cột sex*

*#Kiểm tra xem có những nhãn nào*

print(data['sex'].unique())

['she' 'female' 'lady' 'male' 'woman' 'girl' 'he' 'F' 'mаle' 'M' 'boy'

'gentle' 'other' 'men' 'unknown' 'gіrl' 'wοman' 'mеn' 'bοy' 'womаn'

'սnknown' 'femаle' 'nսll' 'lаdy' 'shе' 'fеmale' 'gеntle' 'hе' 'othеr'

'femalе' 'malе' 'οther' 'gentlе' 'unknοwn' 'femаlе' 'fеmalе']

In [190]:

*# Bước 1: Định nghĩa ánh xạ để chuẩn hóa giá trị giới tính*

sex\_mapping = {

'gentle': 'Male', 'gеntle': 'Male', 'gentlе': 'Male', *# Các biến thể của 'gentle'*

'lady': 'Female', 'lаdy': 'Female', *# Các biến thể của 'lady'*

'she': 'Female', 'shе': 'Female', *# Các biến thể của 'she'*

'female': 'Female', 'fеmale': 'Female', 'femаle': 'Female', 'femalе': 'Female', 'fеmalе': 'Female', 'fеmаle': 'Female', *# Các biến thể của 'female'*

'girl': 'Female', 'gіrl': 'Female', *# Các biến thể của 'girl'*

'male': 'Male', 'M': 'Male', 'mаle': 'Male', 'malе': 'Male', 'mаlе': 'Male', *# Các biến thể của 'male'*

'boy': 'Male', 'bοy': 'Male', *# Các biến thể của 'boy'*

'woman': 'Female', 'wοman': 'Female', 'womаn': 'Female', 'wοmаn': 'Female', *# Các biến thể của 'woman'*

'F': 'Female', *# Viết tắt của 'Female'*

'he': 'Male', 'hе': 'Male', *# Các biến thể của 'he'*

'men': 'Male', 'mеn': 'Male', *# Các biến thể của 'men'*

'unknown': 'Unknown', 'սnknown': 'Unknown', 'unknοwn': 'Unknown', *# Các biến thể của 'unknown'*

'other': 'Other', 'othеr': 'Other', 'οther': 'Other', *# Các biến thể của 'other'*

'nսll': 'Unknown' *# Xem 'null' như 'Unknown'*

}

*# Bước 2: Áp dụng ánh xạ để chuẩn hóa cột sex*

data['sex'] = data['sex'].map(sex\_mapping).fillna('Unknown') *# Ánh xạ và gán NaN thành 'Unknown'*

*# Bước 3: Tính mode của các giá trị đã chuẩn hóa (loại trừ 'Unknown' và 'Other')*

mode\_sex = data[data['sex'].isin(['Female', 'Male'])]['sex'].mode()[0]

*# Bước 4: Thay thế 'Unknown' và 'Other' bằng mode*

data['sex'] = data['sex'].replace(['Unknown', 'Other'], mode\_sex)

*#Kiểm tra lại nhãn sau khi đồng nhất*

print(data['sex'].unique())

['Female' 'Male']

In [191]:

*#Cột job*

*#Kiểm tra xem có những nhãn nào*

print(data['job'].unique())

['Kỹ sư xây dựng' 'Nhà báo' 'Nhân viên kinh doanh' 'Chuyên viên bảo mật'

'Nhân viên lễ tân' 'Kiến trúc sư phần mềm' 'Kiểm toán'

'Thực tập sinh giáo dục' 'Kiến trúc sư' 'UX/UI Designer' 'Nghệ sĩ'

'Kỹ sư phần mềm' 'Y tá' 'Chuyên viên tư vấn tuyển sinh'

'Nhân viên nhà hàng' 'Giáo viên' 'Trưởng phòng kinh doanh' 'Nông dân'

'Chuyên viên bán hàng trực tuyến' 'Nhà tư vấn tài chính' 'Giảng viên'

'Thợ xây' 'Project Manager' 'Chuyên viên nhân sự' 'Kỹ sư nông nghiệp'

'Nhà thiết kế đồ họa' 'Công nhân sản xuất' 'Kỹ sư sản xuất' 'Điều Dưỡng'

'Thư ký pháp lý' 'Luật sư' 'Nhân viên khách sạn' 'Quản lý sản xuất'

'Lập trình viên' 'Kỹ sư giám sát' 'DevOps Engineer' 'Tư vấn khách hàng'

'Dược sĩ' 'Chuyên viên tín dụng' 'Marketing Manager'

'Sales Representative' 'Thực Tập' 'Digital Marketer'

'Nhà phân tích dữ liệu' 'Giám đốc kinh doanh' 'Kế toán'

'Nhân viên ngân hàng' 'Chuyên viên phân tích thị trường' 'Tester'

'Biên dịch viên' 'Bác sĩ' 'Nhân viên hành chính' 'Nhân Viên Bán Hàng']

*# Bước 1: Định nghĩa ánh xạ để chuyển các công việc sang tiếng Anh*

job\_mapping = {

'Thực tập sinh giáo dục': 'Education Intern',

'DevOps Engineer': 'DevOps Engineer',

'Kiến trúc sư phần mềm': 'Software Architect',

'UX/UI Designer': 'UX/UI Designer',

'Trưởng phòng kinh doanh': 'Sales Manager',

'Lập trình viên': 'Programmer',

'Nhà phân tích dữ liệu': 'Data Analyst',

'Y tá': 'Nurse',

'Quản lý sản xuất': 'Production Manager',

'Kỹ sư giám sát': 'Supervisory Engineer',

'Digital Marketer': 'Digital Marketer',

'Nghệ sĩ': 'Artist',

'Thực Tập': 'Intern',

'Điều Dưỡng': 'Nurse',

'Kiến trúc sư': 'Architect',

'Nông dân': 'Farmer',

'Chuyên viên nhân sự': 'HR Specialist',

'Nhân viên kinh doanh': 'Sales Representative',

'Nhân viên hành chính': 'Administrative Staff',

'Nhà báo': 'Journalist',

'Nhân viên nhà hàng': 'Restaurant Staff',

'Bác sĩ': 'Doctor',

'Chuyên viên tư vấn tuyển sinh': 'Admissions Counselor',

'Kỹ sư nông nghiệp': 'Agricultural Engineer',

'Thợ xây': 'Construction Worker',

'Giảng viên': 'Lecturer',

'Chuyên viên bảo mật': 'Security Specialist',

'Kỹ sư xây dựng': 'Civil Engineer',

'Kỹ sư phần mềm': 'Software Engineer',

'Nhà thiết kế đồ họa': 'Graphic Designer',

'Project Manager': 'Project Manager',

'Nhà tư vấn tài chính': 'Financial Consultant',

'Nhân viên khách sạn': 'Hotel Staff',

'Kiểm toán': 'Auditor',

'Thư ký pháp lý': 'Legal Secretary',

'Công nhân sản xuất': 'Production Worker',

'Nhân viên lễ tân': 'Receptionist',

'Chuyên viên tín dụng': 'Credit Specialist',

'Giáo viên': 'Teacher',

'Sales Representative': 'Sales Representative',

'Tester': 'Tester',

'Kỹ sư sản xuất': 'Manufacturing Engineer',

'Luật sư': 'Lawyer',

'Chuyên viên bán hàng trực tuyến': 'Online Sales Specialist',

'Chuyên viên phân tích thị trường': 'Market Analyst',

'Nhân Viên Bán Hàng': 'Sales Representative',

'Giám đốc kinh doanh': 'Sales Director',

'Biên dịch viên': 'Translator',

'Dược sĩ': 'Pharmacist',

'Nhân viên ngân hàng': 'Bank Employee',

'Tư vấn khách hàng': 'Customer Service Representative',

'Marketing Manager': 'Marketing Manager',

'Kế toán': 'Accountant'

}

*# Bước 2: Định nghĩa ánh xạ để phân loại công việc vào các lĩnh vực*

job\_category\_mapping = {

'Education Intern': 'Education',

'DevOps Engineer': 'Technology',

'Software Architect': 'Technology',

'UX/UI Designer': 'Design',

'Sales Manager': 'Sales',

'Programmer': 'Technology',

'Data Analyst': 'Data Analysis',

'Nurse': 'Healthcare',

'Production Manager': 'Manufacturing',

'Supervisory Engineer': 'Engineering',

'Digital Marketer': 'Marketing',

'Artist': 'Arts',

'Intern': 'General',

'Architect': 'Architecture',

'Farmer': 'Agriculture',

'HR Specialist': 'Human Resources',

'Sales Representative': 'Sales',

'Administrative Staff': 'Administration',

'Journalist': 'Media',

'Restaurant Staff': 'Hospitality',

'Doctor': 'Healthcare',

'Admissions Counselor': 'Education',

'Agricultural Engineer': 'Agriculture',

'Construction Worker': 'Construction',

'Lecturer': 'Education',

'Security Specialist': 'Technology',

'Civil Engineer': 'Engineering',

'Software Engineer': 'Technology',

'Graphic Designer': 'Design',

'Project Manager': 'Management',

'Financial Consultant': 'Finance',

'Hotel Staff': 'Hospitality',

'Auditor': 'Finance',

'Legal Secretary': 'Legal',

'Production Worker': 'Manufacturing',

'Receptionist': 'Administration',

'Credit Specialist': 'Finance',

'Teacher': 'Education',

'Tester': 'Technology',

'Manufacturing Engineer': 'Engineering',

'Lawyer': 'Legal',

'Online Sales Specialist': 'Sales',

'Market Analyst': 'Marketing',

'Sales Director': 'Sales',

'Translator': 'Language Services',

'Pharmacist': 'Healthcare',

'Bank Employee': 'Finance',

'Customer Service Representative': 'Customer Service',

'Marketing Manager': 'Marketing',

'Accountant': 'Finance'

}

*# Bước 3: Áp dụng ánh xạ để chuyển đổi sang tiếng Anh*

data['job'] = data['job'].map(job\_mapping).fillna('Unknown')

*# Bước 4: Phân loại công việc vào các lĩnh vực*

data['job'] = data['job'].map(job\_category\_mapping).fillna('Unknown')

*# Bước 5: Tính mode của lĩnh vực (loại trừ 'Unknown')*

mode\_category = data[data['job'] != 'Unknown']['job'].mode()[0]

*# Bước 6: Thay thế 'Unknown' bằng mode*

data['job'] = data['job'].replace('Unknown', mode\_category)

*# Bước 7: Kiểm tra các giá trị trong cột job sau khi làm sạch*

print("**\n**Các giá trị duy nhất trong cột job sau khi làm sạch:")

print(data['job'].unique())

Các giá trị duy nhất trong cột job sau khi làm sạch:

['Engineering' 'Media' 'Sales' 'Technology' 'Administration' 'Finance'

'Education' 'Architecture' 'Design' 'Arts' 'Healthcare' 'Hospitality'

'Agriculture' 'Construction' 'Management' 'Human Resources'

'Manufacturing' 'Legal' 'Customer Service' 'Marketing' 'General'

'Data Analysis' 'Language Services']

In [193]:

import numpy as np

*# Bước 1: Thay thế giá trị 'nan' (chuỗi) và NaN bằng 'other'*

data['carrier'] = data['carrier'].replace(['nan', np.nan], 'other')

*# Bước 2: Kiểm tra các giá trị duy nhất trong cột carrier sau khi làm sạch*

print("**\n**Các giá trị duy nhất trong cột carrier sau khi làm sạch:")

print(data['carrier'].unique())

Các giá trị duy nhất trong cột carrier sau khi làm sạch:

['viettel' 'other' 'vietnamobile' 'mobiphone' 'vinaphone']

*#Kiểm tra xem cột marital\_status có những giá trị nào*

print(data["marital\_status"].unique())

['Married' 'Single' 'cưới' 'Unmarried' 'độc thân' 'ly hôn' 'Divorced']

In [195]:

data['marital\_status'] = data['marital\_status'].replace({

'cưới': 'Married',

'độc thân': 'Single',

'Unmarried': 'Single',

'ly hôn': 'Divorced'

})

data = data.dropna(subset=['marital\_status'])

print(data["marital\_status"].unique())

['Married' 'Single' 'Divorced']

In [196]:

import matplotlib.pyplot as plt

import seaborn as sns

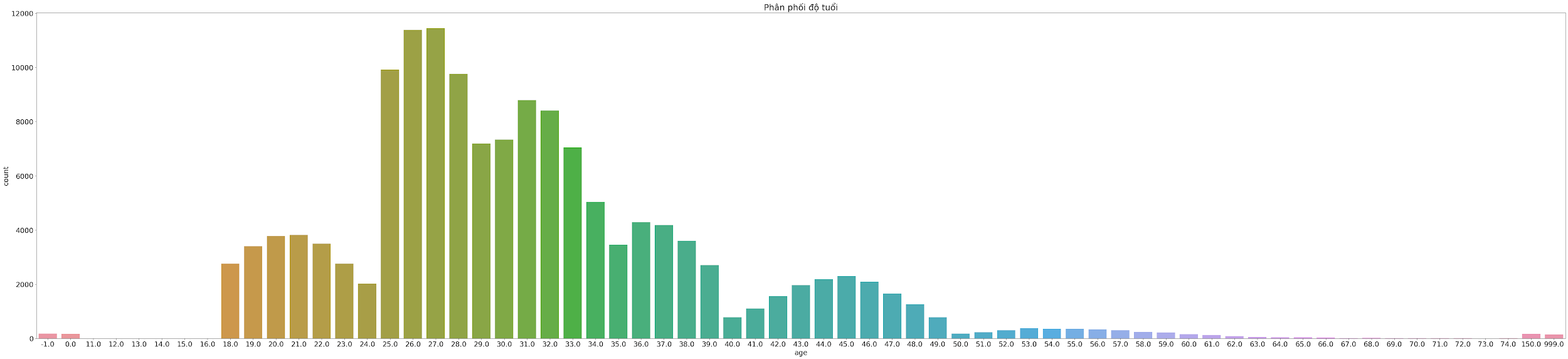
*#Xem phân phối dữ liệu cho cột tuổi*

plt.rcParams['figure.figsize'] = (70, 15)

sns.countplot(x='age', data=data)

plt.title('Phân phối độ tuổi')

plt.show()



In [197]:

*#Xóa các giá trị ngoại lai*

data.drop(data[data['age'].isin([-1, 0, 150, 999])].index, inplace=True)

In [198]:

print(data['add-to-cart'].unique())

print(data['click'].unique())

print(data['favourite'].unique())

print(data['purchase'].unique())

[ 0. 2. 1. 3. 4. 5. 7. 13. 6. 8.]

[6.000e+00 2.100e+01 9.000e+00 1.000e+00 4.000e+00 1.600e+01 5.000e+00

0.000e+00 7.000e+00 1.000e+01 8.000e+00 3.000e+00 4.800e+01 3.100e+01

2.000e+00 1.500e+01 2.700e+01 3.000e+01 4.000e+01 1.250e+02 3.900e+01

1.700e+01 1.300e+01 1.400e+01 2.500e+01 3.700e+01 1.800e+01 1.200e+01

4.400e+01 1.100e+01 2.400e+01 8.300e+01 8.000e+01 3.200e+01 2.800e+01

2.900e+01 7.000e+01 5.200e+01 3.300e+01 2.200e+01 2.000e+01 3.400e+01

1.900e+01 4.600e+01 7.100e+01 3.800e+01 2.460e+02 6.900e+01 3.600e+01

2.300e+01 5.800e+01 7.700e+01 5.900e+01 1.270e+02 5.000e+01 1.240e+02

6.100e+01 5.500e+01 4.900e+01 4.200e+01 5.100e+01 1.540e+02 5.300e+01

2.600e+01 7.600e+01 4.300e+01 5.600e+01 6.600e+01 9.400e+01 6.400e+01

6.200e+01 4.500e+01 6.500e+01 1.750e+02 2.770e+02 6.300e+01 8.500e+01

6.700e+01 1.230e+02 1.100e+02 5.400e+01 9.000e+01 1.180e+02 7.300e+01

2.280e+02 1.090e+02 1.570e+02 2.420e+02 2.740e+02 4.100e+01 1.510e+02

3.500e+01 1.040e+02 1.020e+02 8.900e+01 9.600e+01 4.700e+01 1.190e+02

7.800e+01 1.300e+02 8.200e+01 9.900e+01 7.400e+01 2.230e+02 5.700e+01

1.670e+02 9.500e+01 6.000e+01 1.070e+02 7.900e+01 1.740e+02 6.800e+01

2.180e+02 5.080e+02 2.880e+02 9.300e+01 9.200e+01 3.130e+02 1.080e+02

2.750e+02 2.020e+02 1.220e+02 1.150e+02 1.160e+02 1.210e+02 7.200e+01

7.120e+02 2.540e+02 1.830e+02 7.500e+01 1.560e+02 1.010e+02 6.130e+02

4.450e+02 8.400e+01 1.550e+02 1.530e+02 8.100e+01 8.600e+01 2.240e+02

2.870e+02 1.060e+02 1.410e+02 1.590e+02 1.280e+02 1.340e+02 1.370e+02

2.850e+02 1.380e+02 1.430e+02 1.580e+02 8.700e+01 1.320e+02 1.600e+02

3.110e+02 1.630e+02 9.800e+01 2.980e+02 2.000e+02 1.260e+02 3.300e+02

1.500e+02 1.360e+02 2.760e+02 1.030e+02 1.470e+02 9.100e+01 1.120e+02

8.800e+01 1.880e+02 1.170e+02 1.290e+02 1.000e+02 1.310e+02 1.490e+02

2.340e+02 1.920e+02 2.170e+02 1.140e+02 1.990e+02 2.080e+02 1.200e+02

1.110e+02 1.760e+02 1.960e+02 9.700e+01 1.520e+02 3.070e+02 1.770e+02

1.660e+02 4.820e+02 1.680e+02 1.970e+02 4.100e+02 1.390e+02 1.400e+02

1.870e+02 1.610e+02 1.840e+02 3.917e+03 2.010e+02 1.980e+02 1.050e+02

4.390e+02 2.130e+02 4.580e+02 1.330e+02 1.130e+02 1.620e+02 5.600e+02

1.950e+02 3.060e+02 1.780e+02 4.380e+02 2.050e+02 2.270e+02 5.560e+02

1.700e+02 1.480e+02 1.810e+02 2.710e+02 1.420e+02 2.450e+02 4.320e+02

2.840e+02 1.460e+02 1.640e+02 2.150e+02 3.250e+02 1.850e+02 3.140e+02

1.690e+02 2.070e+02 2.260e+02 2.350e+02 1.189e+03 2.560e+02 5.040e+02

3.080e+02 3.850e+02 1.730e+02 2.060e+02 1.440e+02 6.860e+02 2.290e+02

3.480e+02 3.940e+02 1.350e+02 2.330e+02 4.760e+02 3.790e+02 2.660e+02

5.230e+02 2.480e+02 1.710e+02 2.160e+02 3.050e+02 2.630e+02 2.530e+02

2.200e+02 2.470e+02 1.720e+02 2.690e+02 3.220e+02 2.210e+02 2.640e+02

1.800e+02 3.440e+02 2.440e+02 2.040e+02 2.800e+02 1.450e+02 2.490e+02

1.890e+02 1.790e+02 1.650e+02 1.900e+02 1.820e+02 2.920e+02 2.370e+02

4.240e+02 3.240e+02 2.820e+02 7.490e+02 2.120e+02 3.560e+02 1.910e+02

3.820e+02 2.550e+02 4.740e+02 1.940e+02 5.450e+02 3.320e+02 2.110e+02

3.780e+02 4.060e+02 3.000e+02 2.570e+02 2.250e+02 3.770e+02 2.950e+02

1.860e+02 2.810e+02 3.040e+02 2.670e+02 3.580e+02 2.380e+02 3.160e+02]

[ 0. 1. 3. 4. 2. 8. 5. 6. 7. 15. 9. 13. 10. 36. 11. 26. 27. 16.

18. 14. 32. 30. 12. 17. 24. 19. 20. 21. 28. 29. 25. 50. 22. 31. 51. 23.

41. 70. 48. 59. 54. 57.]

[ 2. 1. 4. 5. 3. 7. 6. 9. 8. 10.]

In [199]:

*# ----------------- Vẽ Box Plot cho cột 'click' -----------------*

plt.figure(figsize=(10, 5))

sns.boxplot(y=data['click'])

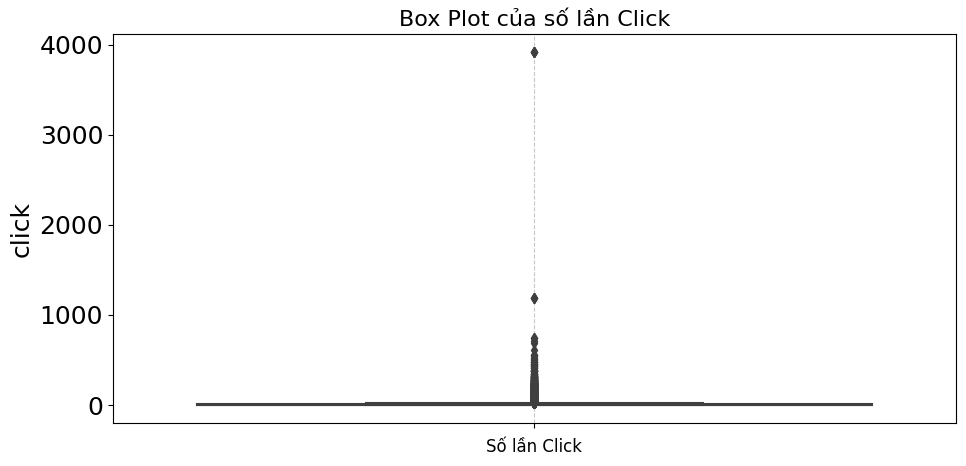
plt.title('Box Plot của số lần Click', fontsize=16)

plt.xlabel('Số lần Click', fontsize=12)

plt.grid(axis='x', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()



In [200]:

Q1 = data['click'].quantile(0.25)

Q3 = data['click'].quantile(0.75)

IQR = Q3 - Q1

*# Giới hạn dưới và trên*

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

*# Lọc bỏ outliers*

data = data[(data['click'] >= lower\_bound) & (data['click'] <= upper\_bound)]

*# ----------------- Vẽ Box Plot cho cột 'click' -----------------*

plt.figure(figsize=(10, 5))

sns.boxplot(y=data['click'])

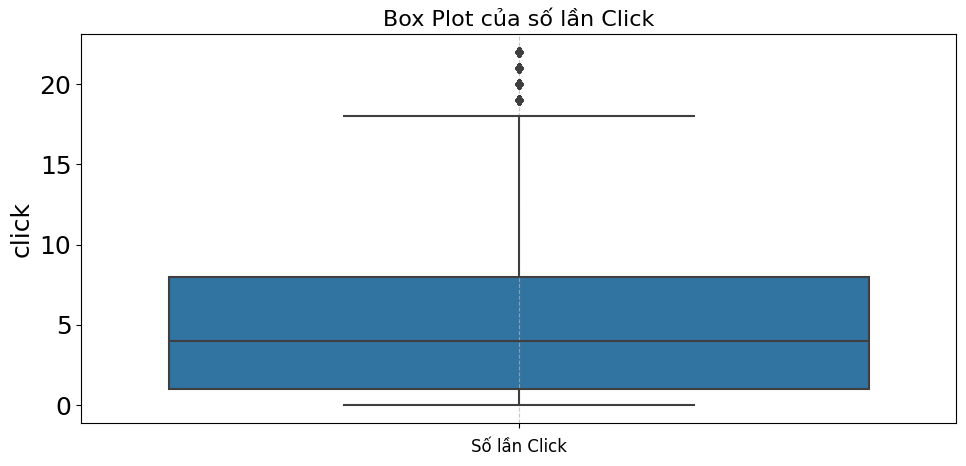
plt.title('Box Plot của số lần Click', fontsize=16)

plt.xlabel('Số lần Click', fontsize=12)

plt.grid(axis='x', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()



In [202]:

print(data['add-to-cart'].unique())

print(data['click'].unique())

print(data['favourite'].unique())

print(data['purchase'].unique())

[0. 2. 1. 4. 3. 5. 8. 7.]

[ 6. 21. 9. 1. 4. 16. 5. 0. 7. 10. 8. 3. 2. 15. 17. 13. 14. 18.

12. 11. 22. 20. 19.]

[ 0. 1. 2. 3. 5. 4. 7. 9. 6. 10. 8. 14. 16. 13. 11. 12. 20. 18.

19. 17. 25. 32. 15. 31. 23. 21.]

[ 2. 1. 4. 3. 7. 8. 5. 6. 9. 10.]

In [203]:

import matplotlib.pyplot as plt

import seaborn as sns

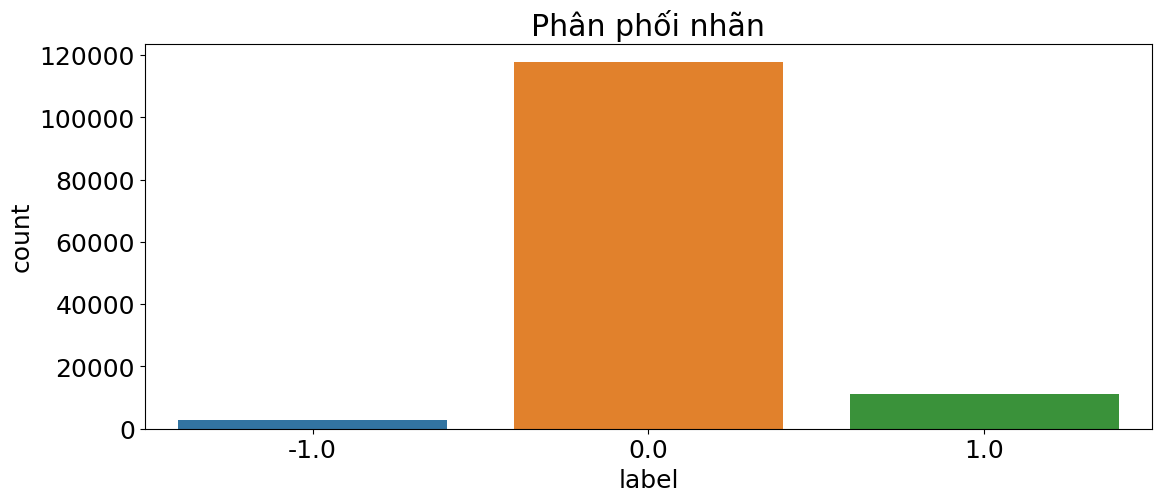
*#Xem phân phối dữ liệu cho cột tuổi*

plt.rcParams['figure.figsize'] = (13, 5)

sns.countplot(x='label', data=data)

plt.title('Phân phối nhãn')

plt.show()



In [204]:

*#Xóa các nhãn -1 đi*

data = data[data['label'] != -1]

In [205]:

print(data.shape)

print(data.head())

(129067, 13)

user\_id merchant\_id label add-to-cart click favourite purchase \

2 203618.0 598.0 0.0 0.0 6.0 0.0 2.0

5 141726.0 1272.0 0.0 0.0 21.0 0.0 2.0

6 117868.0 415.0 0.0 0.0 9.0 0.0 2.0

7 170952.0 212.0 0.0 0.0 1.0 1.0 1.0

9 96875.0 4048.0 0.0 0.0 4.0 0.0 1.0

datetime age sex job carrier marital\_status

2 2024-11-11 26.0 Female Engineering viettel Married

5 2024-11-11 33.0 Female Media other Married

6 2024-11-11 33.0 Female Sales vietnamobile Single

7 2024-11-11 31.0 Female Technology viettel Married

9 2024-11-11 31.0 Male Administration vietnamobile Married

In [206]:

import datetime

current\_date = max(data['datetime'] + datetime.timedelta(days=1))

data['days\_since\_last\_interaction'] = (current\_date - data['datetime']).dt.days

In [207]:

print(data.head())

user\_id merchant\_id label add-to-cart click favourite purchase \

2 203618.0 598.0 0.0 0.0 6.0 0.0 2.0

5 141726.0 1272.0 0.0 0.0 21.0 0.0 2.0

6 117868.0 415.0 0.0 0.0 9.0 0.0 2.0

7 170952.0 212.0 0.0 0.0 1.0 1.0 1.0

9 96875.0 4048.0 0.0 0.0 4.0 0.0 1.0

datetime age sex job carrier marital\_status \

2 2024-11-11 26.0 Female Engineering viettel Married

5 2024-11-11 33.0 Female Media other Married

6 2024-11-11 33.0 Female Sales vietnamobile Single

7 2024-11-11 31.0 Female Technology viettel Married

9 2024-11-11 31.0 Male Administration vietnamobile Married

days\_since\_last\_interaction

2 2

5 2

6 2

7 2

9 2

In [208]:

data = data.drop(columns=['datetime'])

3.4. Mã hóa các nhóm giá trị trong các cột kiểu string

In [209]:

*#mã hóa cột sex*

import pandas as pd

from sklearn.preprocessing import LabelEncoder

*# Tạo LabelEncoder*

le = LabelEncoder()

*# Mã hóa cột chuỗi*

data['sex'] = le.fit\_transform(data['sex'])

*#Mã hóa cột job*

data['job'] = le.fit\_transform(data['job'])

*#Mã hóa cột carrier*

data['carrier'] = le.fit\_transform(data['carrier'])

*#Mã hóa cột marital\_status*

data['marital\_status'] = le.fit\_transform(data['marital\_status'])

In [210]:

data.to\_csv('/kaggle/working/data.csv', index=False)

print("Đã lưu file merged thành tập data.csv")

Đã lưu file merged thành tập data.csv

1. Trực quan hóa dữ liệu

In [231]:

*# Danh sách hành vi*

behavior\_columns = ['click', 'purchase', 'favourite', 'add-to-cart']

*# Vẽ từng biểu đồ riêng biệt*

for behavior **in** behavior\_columns:

plt.figure(figsize=(15, 6))

sns.countplot(data=data, x=behavior, hue='label', palette='Set2')

plt.title(f'Số lượng khách hàng **{**behavior**}** ở nhóm trung thành và không trung thành theo các đặc điểm nhân khẩu học')

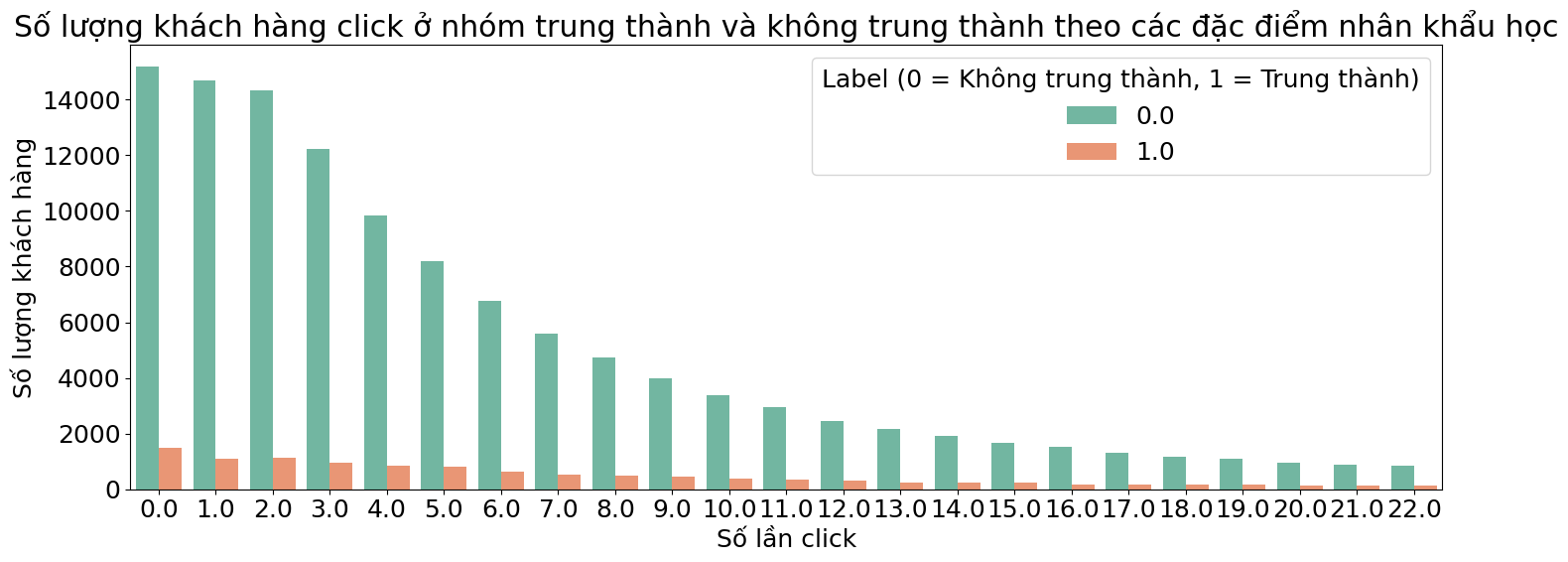
plt.xlabel(f'Số lần **{**behavior**}**')

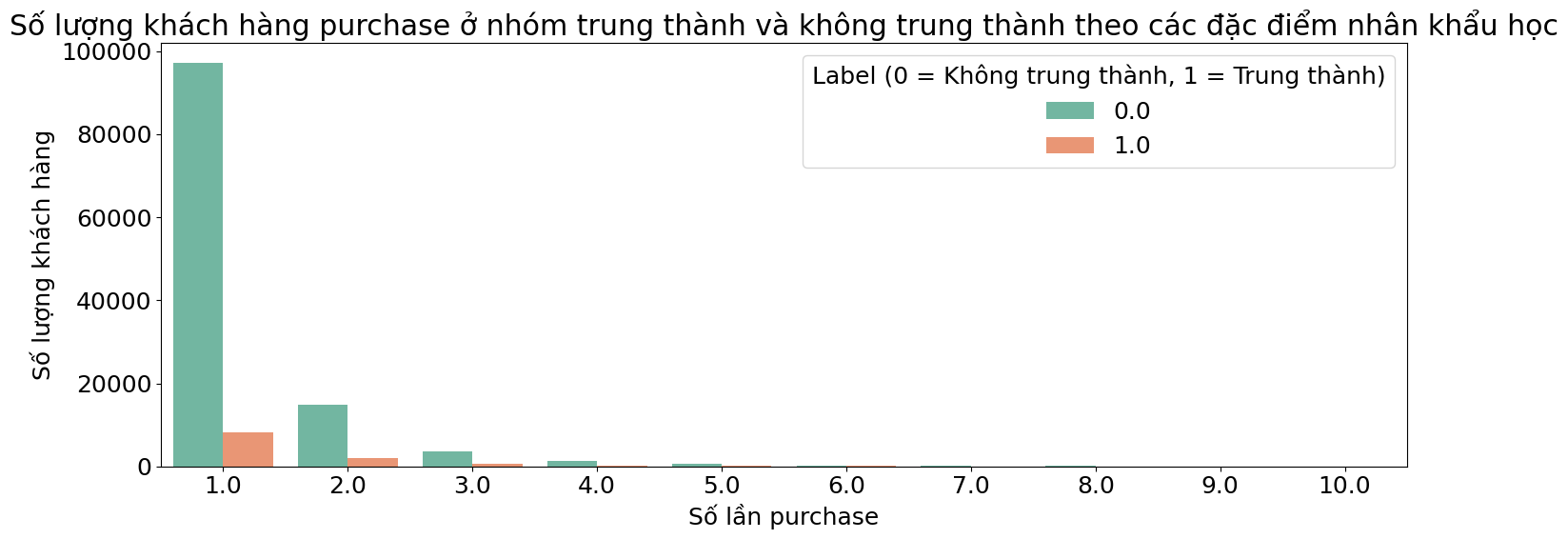
plt.ylabel('Số lượng khách hàng')

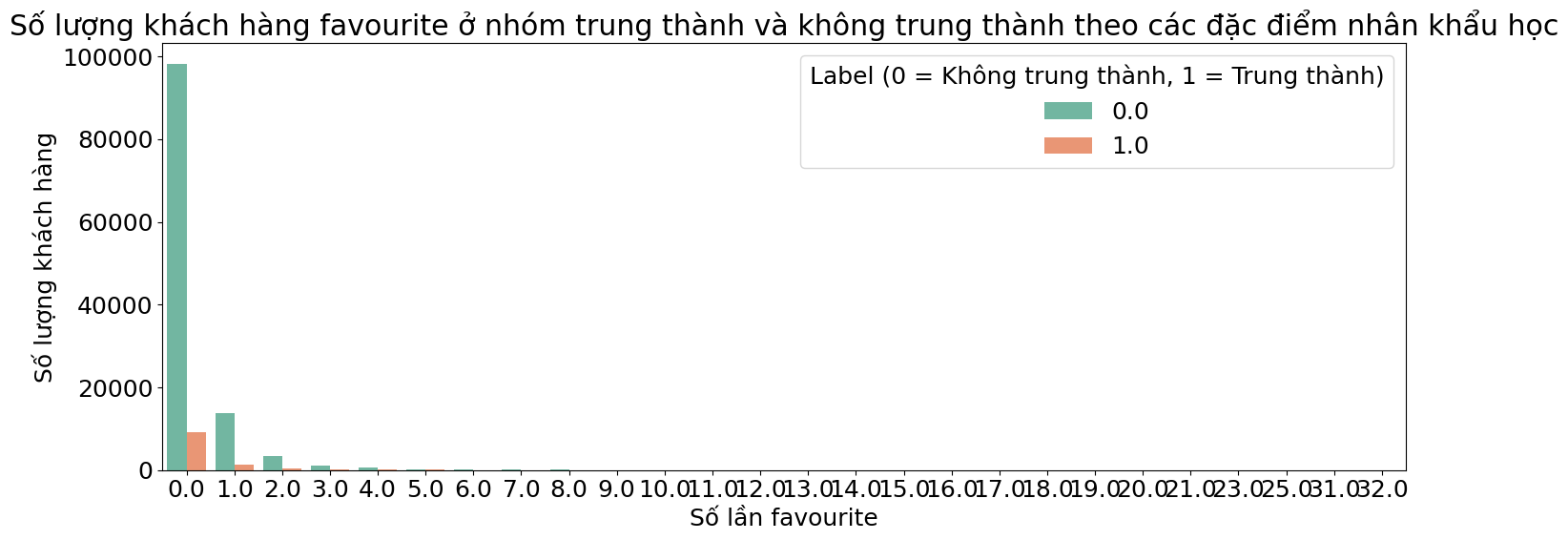
plt.legend(title='Label (0 = Không trung thành, 1 = Trung thành)')

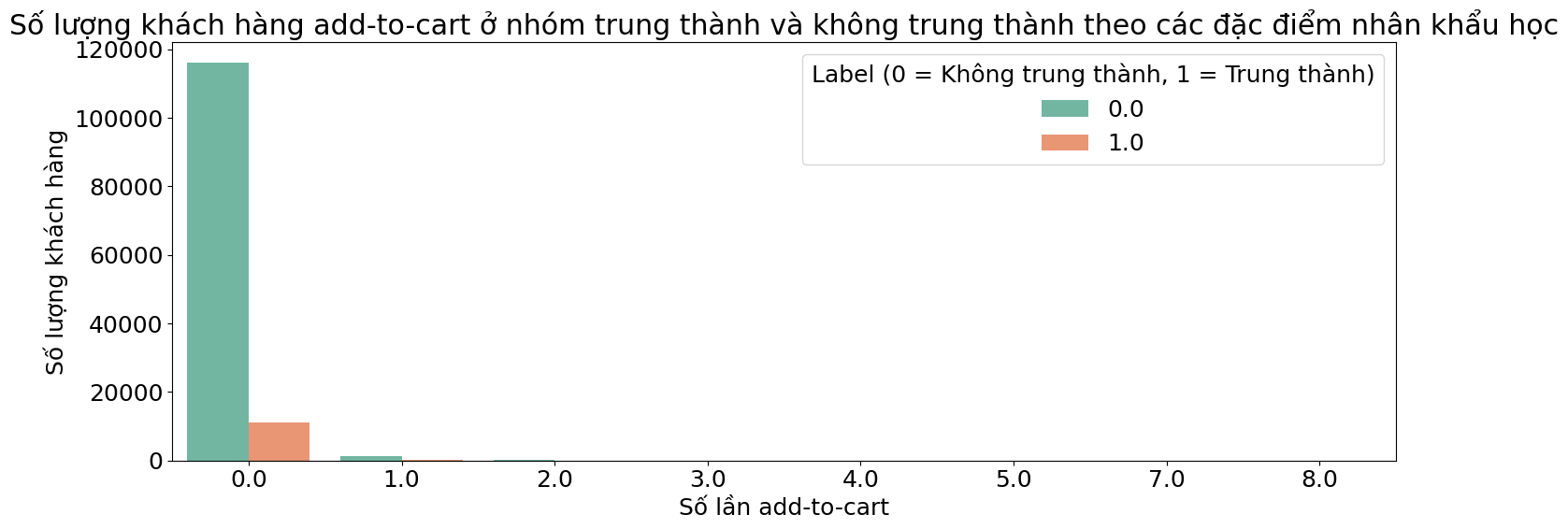
plt.tight\_layout()

plt.show()









In [234]:

*# Danh sách hành vi*

characteristics = ['age', 'sex', 'job', 'marital\_status']

*# Vẽ từng biểu đồ riêng biệt*

for behavior **in** characteristics:

plt.figure(figsize=(30, 6))

sns.countplot(data=data, x=behavior, hue='label', palette='Set2')

plt.title(f'Số lượng khách hàng **{**behavior**}** ở nhóm trung thành và không trung thành theo các đặc điể

điểm nhân khẩu học')

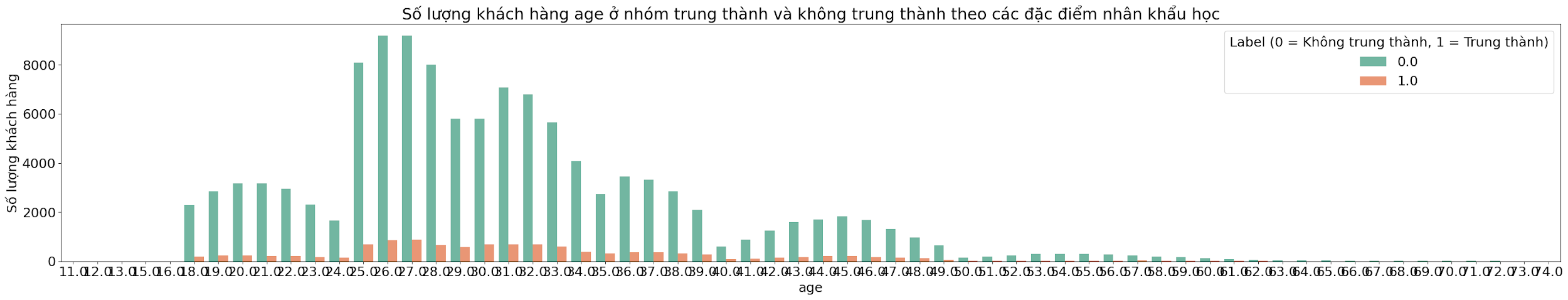
plt.xlabel(f'**{**behavior**}**')

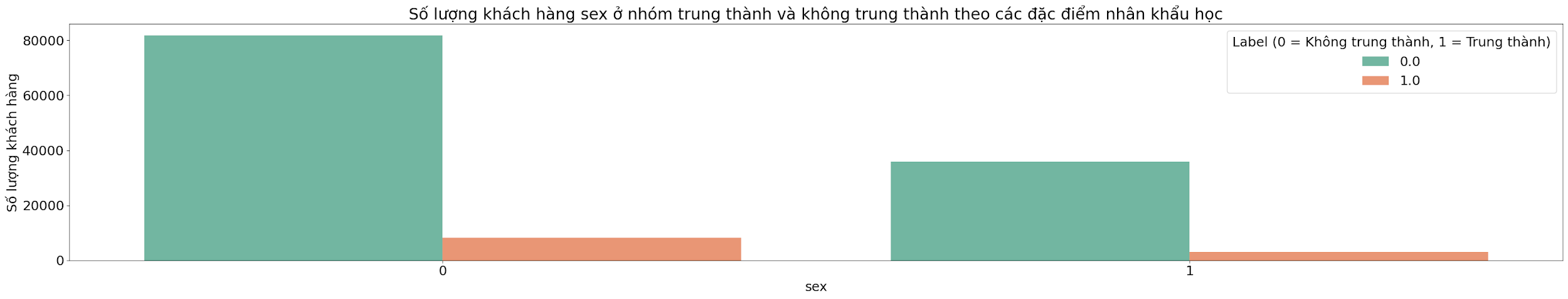
plt.ylabel('Số lượng khách hàng')

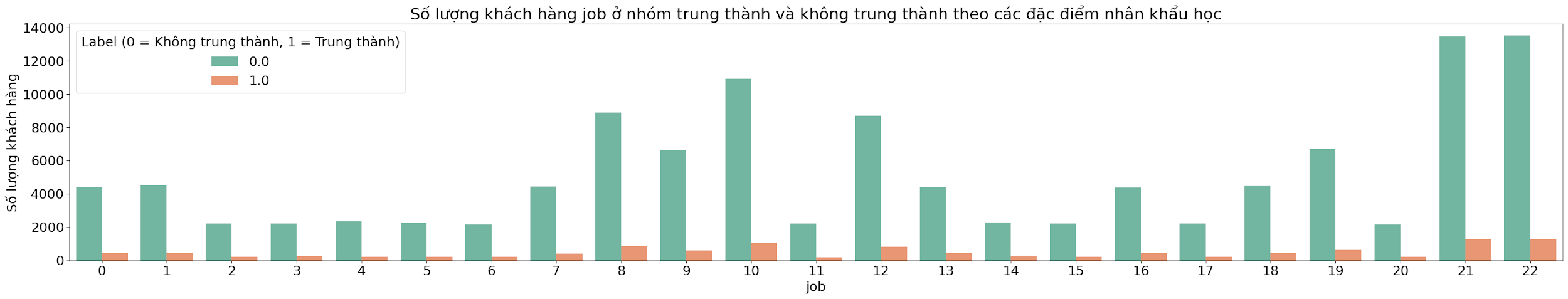
plt.legend(title='Label (0 = Không trung thành, 1 = Trung thành)')

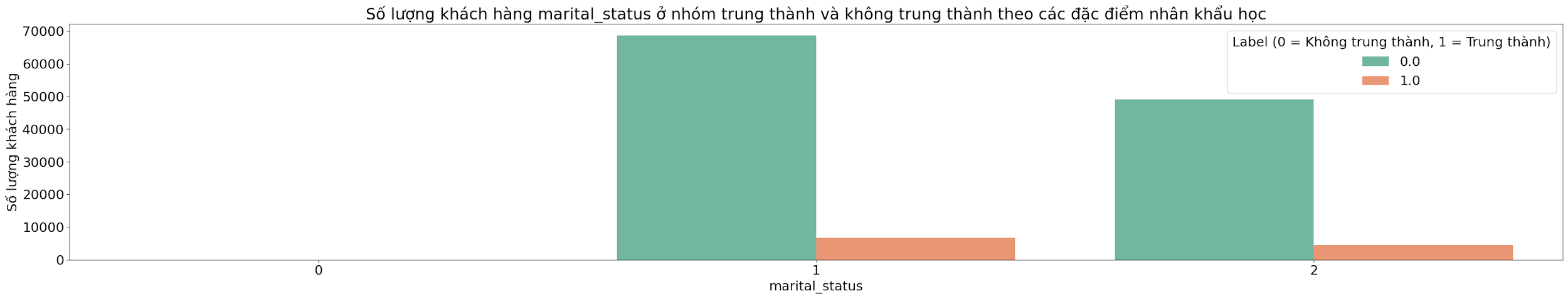
plt.tight\_layout()

plt.show()









In [236]:

plt.figure(figsize=(30, 6))

sns.countplot(data=data, x='days\_since\_last\_interaction', hue='label', palette='Set2')

plt.title(f'Số lượng khách hàng ở nhóm trung thành và không trung thành theo các đặc điểm nhân khẩu học')

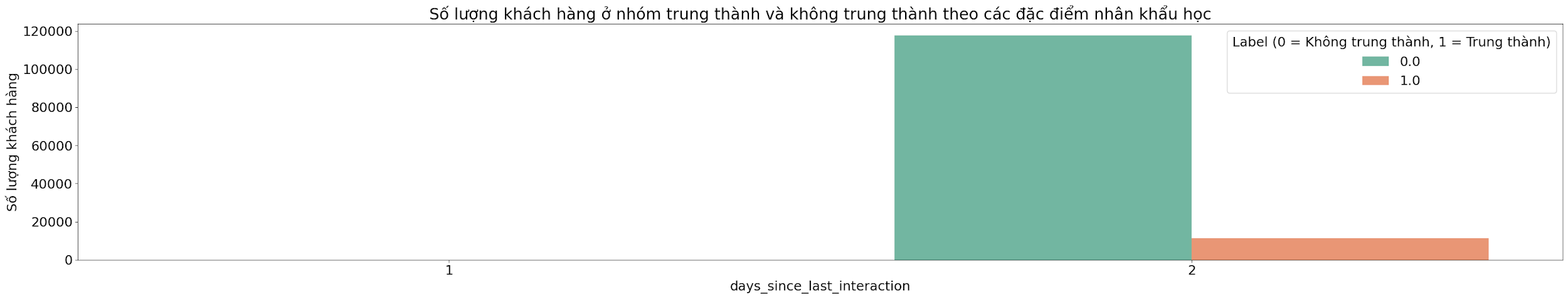
plt.xlabel('days\_since\_last\_interaction')

plt.ylabel('Số lượng khách hàng')

plt.legend(title='Label (0 = Không trung thành, 1 = Trung thành)')

plt.tight\_layout()

plt.show()



In [242]:

correlation\_matrix = data.corr()

plt.rcParams['font.size'] = 8 *# Đặt kích thước font cho số trên heatmap*

plt.figure(figsize=(10, 7)) *# Điều chỉnh kích thước biểu đồ*

ax = sns.heatmap(

data=correlation\_matrix, *# Dữ liệu ma trận tương quan*

cmap='gnuplot', *# Bảng màu*

annot=True, *# HIỂN THỊ GIÁ TRỊ SỐ TRÊN BIỂU ĐỒ*

fmt=".2f", *# Định dạng số thập phân (2 chữ số sau dấu phẩy)*

linewidths=.5 *# Độ rộng đường kẻ giữa các ô*

)

*# Xoay nhãn trục X và Y để dễ đọc hơn nếu có nhiều cột*

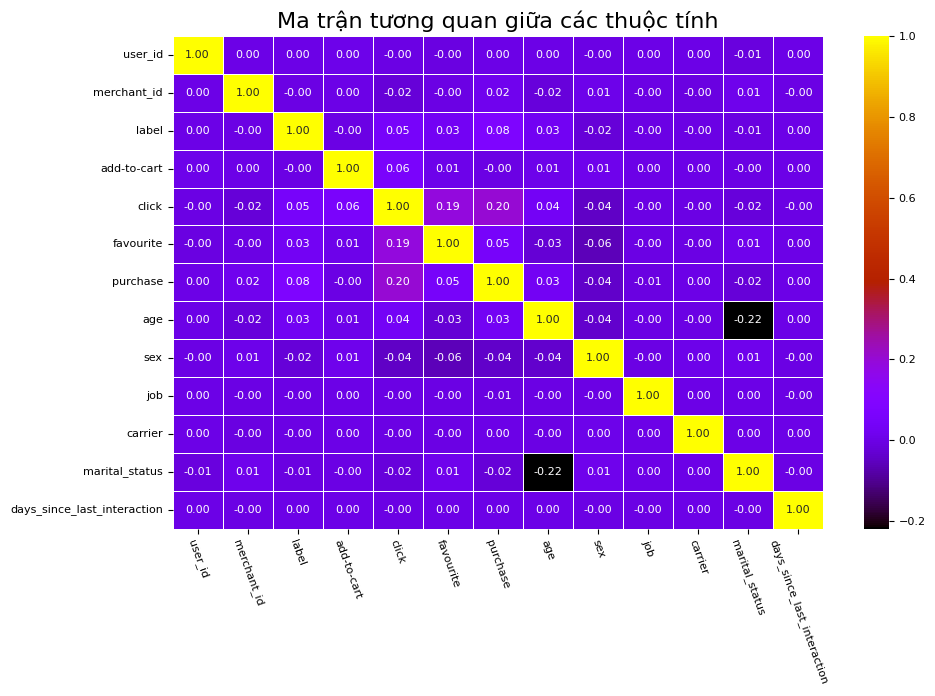
ax.tick\_params(axis='x', rotation=-70)

ax.tick\_params(axis='y', rotation=0)

plt.title('Ma trận tương quan giữa các thuộc tính', fontsize=16)

plt.tight\_layout() *# Đảm bảo bố cục chặt chẽ*

plt.show()



In [ ]:

numerical\_cols = ['add-to-cart', 'click', 'favourite', 'purchase', 'age', 'days\_since\_last\_interaction']

scaler = MinMaxScaler()

data[numerical\_cols] = scaler.fit\_transform(data[numerical\_cols])

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

import skfuzzy as fuzz

from skfuzzy import control as ctrl

import tensorflow as tf

from tensorflow import keras

from sklearn.metrics import precision\_score, recall\_score, f1\_score, roc\_auc\_score, confusion\_matrix, roc\_curve

from sklearn.utils.class\_weight import compute\_class\_weight

from sklearn.cluster import KMeans

from sklearn.ensemble import RandomForestClassifier

from sklearn.utils import resample

import os

df = pd.read\_csv('/kaggle/input/input-data/data.csv')

df.head(5)

Out[2]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | user\_id | merchant\_id | label | add-to-cart | click | favourite | purchase | age | sex | job | carrier | marital\_status | days\_since\_l… |
| 0 | 203618 | 598 | 0 | 0.0 | 0.272727 | 0.00000 | 0.111111 | 0.238095 | 0 | 9 | 3 | 1 | 1.0 |
| 1 | 141726 | 1272 | 0 | 0.0 | 0.954545 | 0.00000 | 0.111111 | 0.349206 | 0 | 20 | 1 | 1 | 1.0 |
| 2 | 117868 | 415 | 0 | 0.0 | 0.409091 | 0.00000 | 0.111111 | 0.349206 | 0 | 21 | 2 | 2 | 1.0 |
| 3 | 170952 | 212 | 0 | 0.0 | 0.045455 | 0.03125 | 0.000000 | 0.317460 | 0 | 22 | 3 | 1 | 1.0 |
| 4 | 96875 | 4048 | 0 | 0.0 | 0.181818 | 0.00000 | 0.000000 | 0.317460 | 1 | 0 | 2 | 1 | 1.0 |

In [3]:

model = RandomForestClassifier(n\_estimators=100, random\_state=42, n\_jobs=-1)

input\_features = ['age', 'days\_since\_last\_interaction', 'click', 'purchase', 'add-to-cart', 'favourite']

output\_feature = 'label'

X = df[input\_features].values

y = df[output\_feature].values

model.fit(X, y)

print("**\n**--- Mức độ quan trọng của các thuộc tính ---")

importances = model.feature\_importances\_

feature\_importance\_df = pd.DataFrame({

'Thuộc tính': input\_features,

'Mức độ quan trọng': importances

})

feature\_importance\_df = feature\_importance\_df.sort\_values(by='Mức độ quan trọng', ascending=False)

print(feature\_importance\_df)

plt.figure(figsize=(10, 6))

plt.barh(feature\_importance\_df['Thuộc tính'], feature\_importance\_df['Mức độ quan trọng'], color='skyblue')

plt.xlabel('Mức độ quan trọng')

plt.title('Mức độ quan trọng của các Thuộc tính Đầu vào')

plt.gca().invert\_yaxis()

plt.grid(axis='x', linestyle='--', alpha=0.7)

plt.show()

top\_features = feature\_importance\_df.head(3)['Thuộc tính'].tolist()

print(f"Dựa trên phân tích, 3 thuộc tính quan trọng nhất là: **{**top\_features**}**")

--- Mức độ quan trọng của các thuộc tính ---

Thuộc tính Mức độ quan trọng

0 age 0.465306

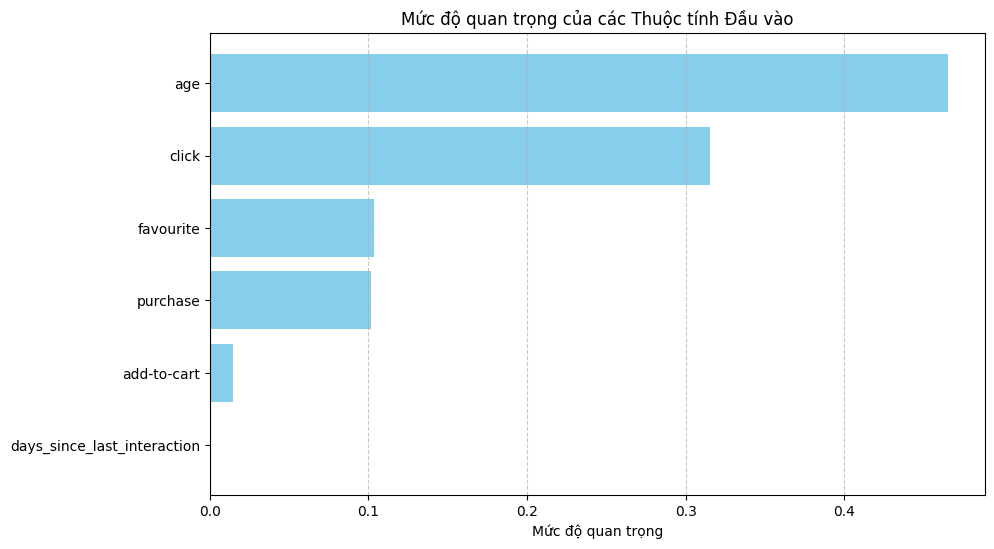
2 click 0.315281

5 favourite 0.103458

3 purchase 0.101662

4 add-to-cart 0.014292

1 days\_since\_last\_interaction 0.000002



Dựa trên phân tích, 3 thuộc tính quan trọng nhất là: ['age', 'click', 'favourite']

In [4]:

class GaussianFuzzyLayer(keras.layers.Layer):

def \_\_init\_\_(self, n\_memb, mu\_initializer, sigma\_initializer, \*\*kwargs):

super().\_\_init\_\_(\*\*kwargs)

self.m = n\_memb *# Số hàm thuộc*

self.mu\_initializer = mu\_initializer

self.sigma\_initializer = sigma\_initializer

def build(self, batch\_input\_shape):

self.n = batch\_input\_shape[-1] *# Số thuộc tính đầu vào*

*# Khởi tạo các tham số mu và sigma*

self.mu = self.add\_weight(name='mu',

shape=(self.m, self.n),

initializer=self.mu\_initializer,

trainable=True)

self.sigma = self.add\_weight(name='sigma',

shape=(self.m, self.n),

initializer=self.sigma\_initializer,

trainable=True)

super().build(batch\_input\_shape)

def call(self, x\_inputs):

x\_inputs\_tiled = tf.tile(x\_inputs, (1, self.m))

x\_inputs\_reshaped = tf.reshape(x\_inputs\_tiled, (-1, self.m, self.n))

*# Công thức hàm liên thuộc Gaussian*

output = tf.exp(-1 \* tf.square(x\_inputs\_reshaped - self.mu) / (tf.square(self.sigma) + 1e-7))

return output

In [5]:

def initialize\_params\_with\_kmeans(X\_train, n\_memb):

n\_input = X\_train.shape[1]

mu\_list, sigma\_list = [], []

for i **in** range(n\_input):

feature\_data = X\_train[:, i]

unique\_values = np.unique(feature\_data)

if len(unique\_values) < n\_memb:

min\_val, max\_val = np.min(feature\_data), np.max(feature\_data)

mus = np.linspace(min\_val, max\_val, n\_memb)

sigma\_val = (max\_val - min\_val) / (2 \* (n\_memb - 1)) if n\_memb > 1 **and** max\_val != min\_val else 0.1

sigmas = np.full(n\_memb, sigma\_val)

else:

feature\_data\_reshaped = feature\_data.reshape(-1, 1)

kmeans = KMeans(n\_clusters=n\_memb, random\_state=42, n\_init='auto').fit(feature\_data\_reshaped)

mus = kmeans.cluster\_centers\_.flatten()

sigmas = [np.std(feature\_data[kmeans.labels\_ == j]) + 1e-7 if len(feature\_data[kmeans.labels\_ == j]) >= 2 else np.std(feature\_data)/n\_memb + 1e-7 for j **in** range(n\_memb)]

sorted\_indices = np.argsort(mus)

mu\_list.append(mus[sorted\_indices])

sigma\_list.append(np.array(sigmas)[sorted\_indices])

return tf.constant\_initializer(np.array(mu\_list).T), tf.constant\_initializer(np.array(sigma\_list).T)

In [6]:

def visualize\_gauss():

feature\_names = ['age', 'click', 'purchase', 'favourite']

scaler = MinMaxScaler()

X\_scaled = scaler.fit\_transform(df[feature\_names])

N\_MEMB = 3

N\_INPUTS = len(feature\_names)

mu\_list = []

sigma\_list = []

for i **in** range(N\_INPUTS):

feature\_data = X\_scaled[:, i].reshape(-1, 1)

kmeans = KMeans(n\_clusters=N\_MEMB, random\_state=42, n\_init='auto').fit(feature\_data)

mus = kmeans.cluster\_centers\_.flatten()

sigmas = []

for j **in** range(N\_MEMB):

cluster\_points = X\_scaled[:, i][kmeans.labels\_ == j]

if len(cluster\_points) < 2:

sigma\_val = np.std(X\_scaled[:, i]) / N\_MEMB

else:

sigma\_val = np.std(cluster\_points)

sigmas.append(sigma\_val + 1e-7)

sorted\_indices = np.argsort(mus)

mu\_list.append(mus[sorted\_indices])

sigma\_list.append(np.array(sigmas)[sorted\_indices])

mu\_init\_values = np.array(mu\_list).T

sigma\_init\_values = np.array(sigma\_list).T

fuzzy\_layer = GaussianFuzzyLayer(

n\_memb=N\_MEMB,

mu\_initializer=tf.constant\_initializer(mu\_init\_values),

sigma\_initializer=tf.constant\_initializer(sigma\_init\_values)

)

dummy\_input = tf.zeros((1, N\_INPUTS), dtype=tf.float32)

\_ = fuzzy\_layer(dummy\_input)

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(20, 10))

fig.suptitle('Trực quan hóa các Hàm thuộc Gauss cho 4 thuộc tính', fontsize=20, y=1.02)

x\_range = np.linspace(0, 1, 200).astype(np.float32)

for i, ax **in** enumerate(axes.flatten()):

plot\_input = np.zeros((len(x\_range), N\_INPUTS), dtype=np.float32)

plot\_input[:, i] = x\_range

membership\_values = fuzzy\_layer(plot\_input).numpy()

for j **in** range(N\_MEMB):

mu\_val = mu\_init\_values[j, i]

sigma\_val = sigma\_init\_values[j, i]

ax.plot(x\_range, membership\_values[:, j, i], label=f'μ=**{**mu\_val**:**.2f**}**, σ=**{**sigma\_val**:**.2f**}**', lw=2)

ax.set\_title(f'Thuộc tính: **{**feature\_names[i]**}**', fontsize=14)

ax.set\_xlabel('Giá trị đầu vào (Đã chuẩn hóa)', fontsize=10)

ax.set\_ylabel('Mức độ thuộc', fontsize=10)

ax.legend()

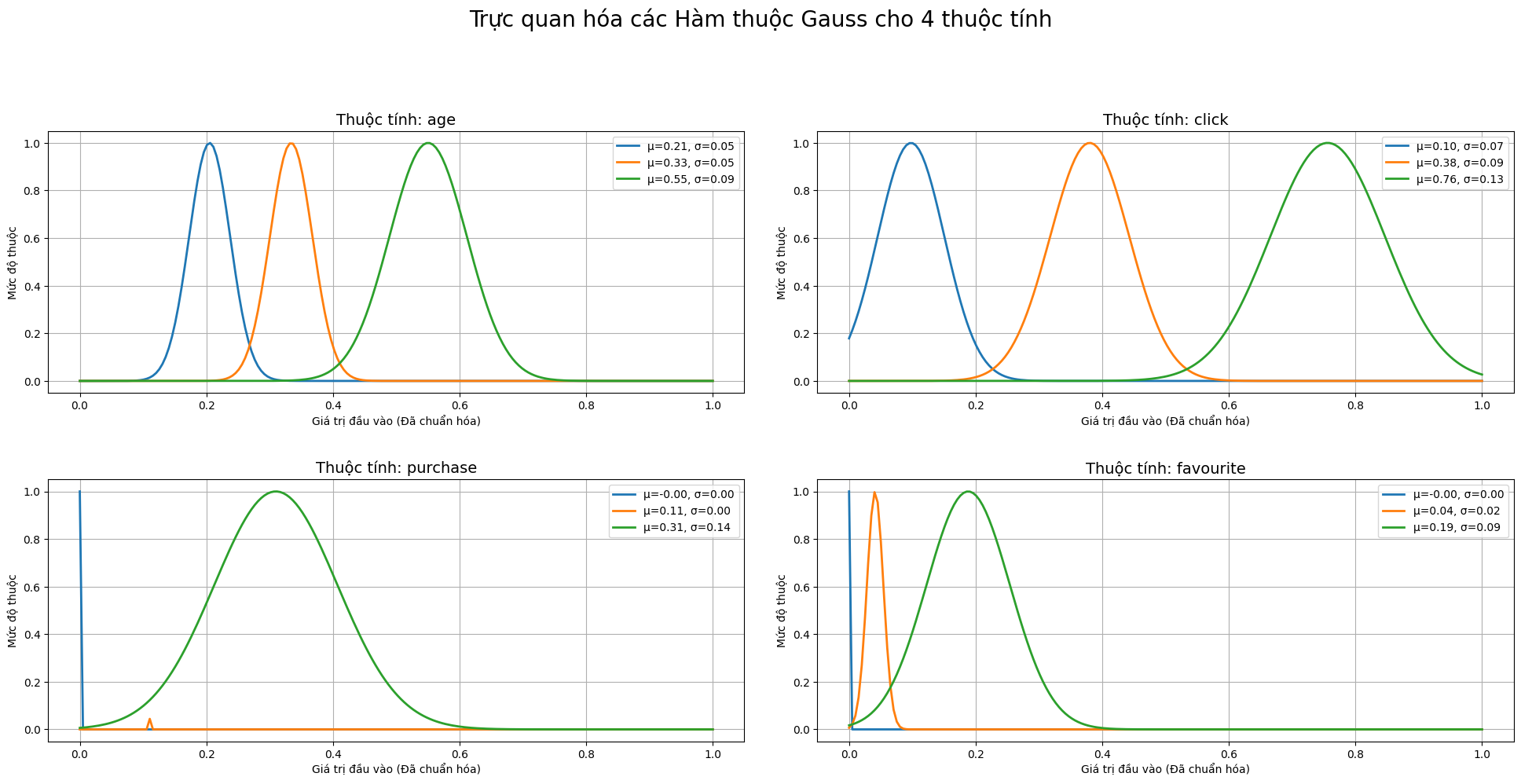
ax.grid(True)

plt.tight\_layout(pad=3.0)

plt.show()

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

visualize\_gauss()



In [7]:

class FullCombinationRuleLayer(keras.layers.Layer):

def \_\_init\_\_(self, \*\*kwargs):

super().\_\_init\_\_(\*\*kwargs)

def call(self, firing\_strengths\_per\_feature):

rules = firing\_strengths\_per\_feature[:, :, 0]

n\_input = firing\_strengths\_per\_feature.shape[2]

if n\_input **is** None:

n\_input = tf.shape(firing\_strengths\_per\_feature)[2]

for i **in** range(1, n\_input):

current\_mfs = firing\_strengths\_per\_feature[:, :, i]

rules\_exp = tf.expand\_dims(rules, axis=2)

mfs\_exp = tf.expand\_dims(current\_mfs, axis=1)

combined = rules\_exp \* mfs\_exp

rules = tf.reshape(combined, (tf.shape(rules)[0], -1))

return rules

In [8]:

class NormLayer(keras.layers.Layer):

def \_\_init\_\_(self, \*\*kwargs):

super().\_\_init\_\_(\*\*kwargs)

def call(self, w):

w\_sum = tf.reshape(tf.reduce\_sum(w, axis=1), (-1, 1))

w\_norm = w / (w\_sum + tf.keras.backend.epsilon())

return w\_norm

In [9]:

class DefuzzLayer(keras.layers.Layer):

def \_\_init\_\_(self, n\_rules, \*\*kwargs):

super().\_\_init\_\_(\*\*kwargs)

self.n\_rules = n\_rules

def build(self, batch\_input\_shape):

w\_norm\_shape, x\_inputs\_shape = batch\_input\_shape

n\_input = x\_inputs\_shape[-1]

self.consequence\_params = self.add\_weight(name='consequence\_params',

shape=(n\_input + 1, self.n\_rules),

initializer=keras.initializers.RandomUniform(minval=-1, maxval=1),

trainable=True)

super().build(batch\_input\_shape)

def call(self, inputs):

w\_norm, x\_inputs = inputs

x\_inputs\_with\_bias = tf.concat([x\_inputs, tf.ones((tf.shape(x\_inputs)[0], 1))], axis=1)

rule\_outputs = tf.matmul(x\_inputs\_with\_bias, self.consequence\_params)

weighted\_rule\_outputs = tf.multiply(w\_norm, rule\_outputs)

return weighted\_rule\_outputs

In [10]:

class SummationLayer(keras.layers.Layer):

def \_\_init\_\_(self, \*\*kwargs):

super().\_\_init\_\_(\*\*kwargs)

def call(self, input\_):

output = tf.reduce\_sum(input\_, axis=1, keepdims=True)

return output

In [11]:

def build\_anfis\_model(n\_input, n\_memb, mu\_init, sigma\_init):

n\_rules = n\_memb \*\* n\_input

x\_inputs = keras.Input(shape=(n\_input,), name="input\_features")

fuzzy\_layer\_output = GaussianFuzzyLayer(n\_memb, mu\_init, sigma\_init, name="fuzzy\_layer")(x\_inputs)

rule\_layer\_output = FullCombinationRuleLayer(name="rule\_layer")(fuzzy\_layer\_output)

norm\_layer\_output = NormLayer(name="norm\_layer")(rule\_layer\_output)

defuzz\_layer\_output = DefuzzLayer(n\_rules, name="defuzz\_layer")([norm\_layer\_output, x\_inputs])

summation\_layer\_output = SummationLayer(name="summation\_layer")(defuzz\_layer\_output)

return keras.Model(inputs=x\_inputs, outputs=summation\_layer\_output, name=f"anfis\_kmeans\_**{**n\_rules**}**\_rules\_model")

In [12]:

input\_features = ['age', 'click', 'favourite', 'purchase']

output\_feature = 'label'

X\_full = df[input\_features]

y\_full = df[output\_feature]

X\_train\_full, X\_test, y\_train\_full, y\_test = train\_test\_split(X\_full, y\_full, test\_size=0.2, random\_state=42, stratify=y\_full)

train\_df = pd.concat([X\_train\_full, y\_train\_full], axis=1)

majority\_df = train\_df[train\_df[output\_feature] == 0]

minority\_df = train\_df[train\_df[output\_feature] == 1]

print(f"Số mẫu lớp đa số (0): **{**len(majority\_df)**}**")

print(f"Số mẫu lớp thiểu số (1): **{**len(minority\_df)**}**")

minority\_upsampled = resample(minority\_df,

replace=True,

n\_samples=len(majority\_df),

random\_state=42)

train\_df\_resampled = pd.concat([majority\_df, minority\_upsampled])

print("**\n**Sau khi over-sampling:")

print(train\_df\_resampled[output\_feature].value\_counts())

X\_train\_resampled\_df = train\_df\_resampled[input\_features]

y\_train\_resampled = train\_df\_resampled[output\_feature]

scaler = MinMaxScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train\_resampled\_df)

X\_test\_scaled = scaler.transform(X\_test)

X\_train\_final, X\_val, y\_train\_final, y\_val = train\_test\_split(X\_train\_scaled, y\_train\_resampled, test\_size=0.2, random\_state=42, stratify=y\_train\_resampled)

Số mẫu lớp đa số (0): 94227

Số mẫu lớp thiểu số (1): 9026

Sau khi over-sampling:

label

0 94227

1 94227

Name: count, dtype: int64

In [13]:

np.random.seed(42)

tf.random.set\_seed(42)

N\_INPUTS = X\_train\_final.shape[1]

N\_MEMB = 3

mu\_init, sigma\_init = initialize\_params\_with\_kmeans(X\_train\_final, N\_MEMB)

anfis = build\_anfis\_model(N\_INPUTS, N\_MEMB, mu\_init, sigma\_init)

anfis.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.0005),

loss=tf.keras.losses.BinaryCrossentropy(from\_logits=True),

metrics=['accuracy'])

*# anfis.summary()*

*# class\_weights = compute\_class\_weight('balanced', classes=np.unique(y\_train), y=y\_train)*

*# class\_weights\_dict = dict(enumerate(class\_weights))*

*# print(f"Trọng số lớp: {class\_weights\_dict}")*

print("**\n**---Training---")

history = anfis.fit(X\_train\_final, y\_train\_final,

epochs=100,

batch\_size=64,

verbose=1,

validation\_data=(X\_val, y\_val))

print("**\n**---Đánh giá mô hình---")

test\_loss, test\_accuracy = anfis.evaluate(X\_test\_scaled, y\_test, verbose=0)

print(f"Test Loss: **{**test\_loss**:**.4f**}**")

print(f"Test Accuracy: **{**test\_accuracy**:**.4f**}**")

y\_pred\_logits = anfis.predict(X\_test\_scaled)

y\_pred\_probs = tf.sigmoid(y\_pred\_logits).numpy().flatten()

y\_pred\_class = (y\_pred\_probs > 0.5410).astype(int)

cm = confusion\_matrix(y\_test, y\_pred\_class)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

xticklabels=['Dự đoán 0', 'Dự đoán 1'],

yticklabels=['Thực tế 0', 'Thực tế 1'])

plt.title('Ma trận nhầm lẫn (Confusion Matrix)', fontsize=16)

plt.ylabel('Nhãn Thực tế', fontsize=12)

plt.xlabel('Nhãn Dự đoán', fontsize=12)

plt.show()

precision = precision\_score(y\_test, y\_pred\_class)

recall = recall\_score(y\_test, y\_pred\_class)

f1 = f1\_score(y\_test, y\_pred\_class)

print(f"Precision: **{**precision**:**.4f**}**")

print(f"Recall: **{**recall**:**.4f**}**")

print(f"F1-score: **{**f1**:**.4f**}**")

print("**\n**---Trực quan hóa kết quả huấn luyện---")

plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Model Loss History')

plt.xlabel('Epoch')

plt.ylabel('Loss (Binary Crossentropy)')

plt.legend()

plt.grid(True)

plt.subplot(1, 2, 2)

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy History')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

---Training---

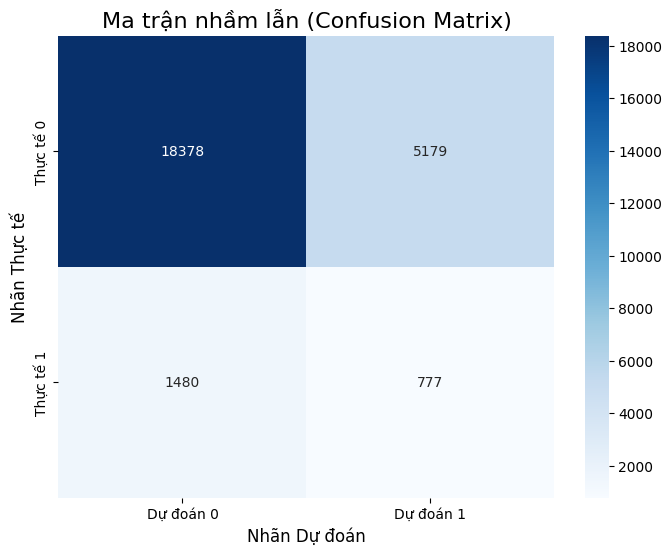
Epoch 100/100

**2356/2356** ━━━━━━━━━━━━━━━━━━━━ **7s** 3ms/step - accuracy: 0.5315 - loss: 0.6784 - val\_accuracy: 0.5342 - val\_loss: 0.6764

---Đánh giá mô hình---

Test Loss: 0.6821

Test Accuracy: 0.8713



Precision: 0.1305

Recall: 0.3443

F1-score: 0.1892

---Trực quan hóa kết quả huấn luyện---

