BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH

TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

DỰ ĐOÁN KHẢ NĂNG ĐĂNG KÝ GỬI KỲ HẠN THÔNG QUA TIẾP THỊ DI ĐỘNG BẰNG PHƯƠNG PHÁP PHÂN LỚP LUẬT KẾT HỢP

Học phần: MÁY HỌC

Chuyên ngành: Khoa học dữ liệu – Khóa 47

Nhóm Sinh Viên:

Họ và tên	MSSV
Đỗ Quang Thiên Phú	31211024191
Lê Trần Khánh Phú	31211024087
Lê Duy Phụng	31211027604
Phạm Minh Phước	31211027663
Trương Vũ Phương Quỳnh	31211027668

Giảng Viên: TS. Nguyễn An Tế

TP. Hồ Chí Minh, ngày 26 tháng 11 năm 2023

LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn đến khoa Công nghệ thông tin kinh doanh – Trường Công nghệ và Thiết kế, Đại học Kinh tế TP. Hồ Chí Minh đã đưa học phần máy học vào chương trình đào tạo để chúng em có cơ hội được tiếp cận với nhiều kiến thức mới lạ, có ý nghĩa cho ngành nghề sau này.

Tiếp theo, chúng em xin bày tỏ lòng biết ơn đến các nguồn học liệu được tham khảo trong đồ án. Những tài liệu này đã cung cấp cho chúng em những thông tin quan trọng, là nền tảng giúp chúng em tiếp cận đề tài dễ dàng hơn và hiểu hơn về chủ đề mà mình. Đồng thời, chúng em xin gửi lời cảm ơn đến tất cả các bạn học đã cùng chúng em chia sẻ, trao đổi và góp ý trong quá trình thực hiện đồ án. Sự giúp đỡ của các bạn đã giúp chúng em hoàn thiện đề tài một cách toàn diện hơn.

Đặc biệt, chung em muốn gửi lời cảm ơn sâu sắc nhất đến TS. Nguyễn An Tế, giảng viên bộ môn Máy học. Người đã trực tiếp giảng dạy và hướng dẫn nhóm chúng em trong suốt quá trình thực hiện đồ án. Sự tận tâm, nhiệt tình và kiến thức chuyên môn sâu rộng của thầy đã giúp chúng em hiểu rõ hơn về các khái niệm, phương pháp và các kỹ thuật trong máy học cũng như các môn học liên quan khác, từ đó có thể hoàn thành đồ án một cách tốt nhất.

Mặc dù chúng em đã cố gắng nhưng chắc chắn đồ án không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý từ thầy để có thể hoàn thiện phát triển đề tài đồ án môn học hơn trong tương lai.

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Từ gốc
CBA	Classification Based on Association
CMAR	Classification based on Multiple Association Rules
CPAR	Classification based on Predictive Association Rules
RG	Rule Generate
CB	Classifier Builder
CARs	Class Association Rules
prCARs	prunning Class Association Rules

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1. GIỚI THIỆU VẤN ĐỀ NGHIÊN CỨU

Huy động vốn là một trong những hoạt động quan trọng của các ngân hàng thương mại, là yếu tố quyết định đến quy mô hoạt động, quy mô tín dụng của các ngân hàng. Nguồn vốn của ngân hàng hình thành từ nhiều nguồn khác ngay, trong đó chiếm tỷ lệ lớn nhất là nguồn vốn huy động từ tiền gửi tiết kiệm của khách hàng. Huy động tiền gửi tiết kiệm đóng một vai trò quan trọng trong hoạt động của các Ngân hàng thương mại vì nguồn tiền gửi tiết kiệm là một nguồn vốn cơ bản, cốt lõi và có tính ổn định cao. Trong những năm vừa qua, các Ngân hàng thương mại đã có nhiều nỗ lực trong hoạt động huy động tiền gửi tiết kiệm và đã đạt được nhiều thành tựu trong hoạt động này. Tuy nhiên, vẫn có nhiều mặt bất cập, hạn chế. Do đó, đối với các ngân hàng thương mại, việc tìm hiểu thực trạng huy động vốn khách hàng, từ đó đề xuất các giải pháp thúc đẩy quyết định gửi tiền tiết kiệm của khách hàng là vô cùng cần thiết.

Ngày nay, với sự phát triển mạnh mẽ của kỹ thuật công nghệ và internet. Theo báo cáo của Wearesocial, tính đến tháng 01/2020 số lượng người sử dụng internet tại Việt Nam đạt 68.17 triệu người dùng chiếm 70% tổng dân số. Điều này làm cho digital ngày càng trở nên phát triển và là một kênh quảng cáo hiệu quả của các doanh nghiệp. Các ngân hàng thương mại cổ phần không ngừng triển khai các chiến lược digital marketing để nâng cao uy tín, sự tin cậy đối với ngân hàng. Đây là một lợi thế nhằm nâng cao khả năng tiếp cận với khách hàng. Nhưng đây cũng là một rào cản đối với những tệp khách hàng không sử dụng internet hoặc ít có khả năng tương tác trên không gian mạng, thì những hình thức truyền thống như tư vấn trực tiếp tại chi nhánh giao dịch, tư vấn tiếp thị qua điện thoại di động là những hình thức được ưu tiên. Trong đó, hình thức tiếp thị thông qua điện thoại di động là tiện lợi hơn cả với những khách hàng ở vùng sâu vùng xa có nhu cầu gửi tiền tiết kiệm nhưng không được tiếp cận với internet, không biết đến những dịch vụ gửi tiền tiết kiệm của ngân hàng và khó khăn trong việc di chuyển

Nhận thấy vấn đề này, các ngân hàng thương mại cần xây dựng nên một tệp khách hàng có thể có nhu cầu đăng ký tiền gửi có kỳ hạn. Tệp khách hàng này cần phải đáp ứng hiệu quả cho các vấn đề đặt ra phía trên để tránh lãng phí nguồn lực cho các khách hàng không có nhu cầu nhận tư vấn thông qua tiếp thị điện thoại.

Để thực hiện mục tiêu dự đoán khả năng đăng ký gửi kỳ hạn của khách hàng,

nghiên cứu của nhóm sử dụng thuật toán Classification using pattern frequents, thuật toán này chủ yếu tập trung vào việc phân loại dữ liệu dựa trên các quy luật kết hợp. Trong bối cảnh huy động vốn từ tiền gửi tiết kiệm, việc áp dụng mô hình này giúp xác định chính xác mức độ quan tâm của khách hàng đối với việc đăng ký gửi tiền tiết kiệm có kỳ hạn thông qua các thông tin cá nhân và cuộc gọi tiếp thị.

Phương pháp phân lớp luật kết hợp không chỉ tận dụng sức mạnh của máy học mà còn giúp hiểu rõ hơn về mối quan hệ giữa các yếu tố ảnh hưởng đến quyết định của khách hàng. Điều này mang lại sự linh hoạt và hiệu quả trong việc xây dựng mô hình dự đoán, đồng thời giúp tối ưu hóa quy trình phân loại, đặc biệt là khi có sự biến động lớn trong dữ liệu. Đồng thời, sự kết hợp giữa công nghệ thông tin và thuật toán máy học mở ra những tiềm năng mới để cải thiện trải nghiệm của khách hàng và tối ưu hóa hoạt động huy động vốn của ngân hàng thương mại.

Trong thời buổi các công nghệ trí tuệ nhân tạo, sử dụng các ứng dụng công nghệ thông tin vào vận hành được các doanh nghiệp áp dụng rộng rãi nhất là trong lĩnh vực tài chính ngân hàng, công nghệ thông tin không chỉ giúp tăng chất lượng dịch vụ đối với khách hàng mà còn hỗ trợ vô cùng lớn cho việc ra quyết định của người quản lý. Ứng dụng công nghệ thông tin như vậy gọi là Hệ hỗ trợ ra quyết định (Decision support systems - DSS), một nội dung khác gần với DSS là Business Intelligence (BI) sử dụng công nghệ như data warehouses và khai phá dữ liệu để hỗ trợ ra quyết định, cho phép khai thác bán tự động để khám phá và dự đoán tri thức từ dữ liệu thô. Cụ thể, phân loại chính là nhiệm vụ khai phá dữ liệu phổ biến nhất và mục tiêu của nó là xây dựng một mô hình dữ liệu vận hành để học những chức năng chưa biết được đưa vào thông qua các biến, là đặc điểm của một đối tượng cùng với nhãn lớp đầu ra. Điều này sẽ giúp cho việc xây dựng tệp khách hàng tiềm năng được nêu ở trên một cách nhanh chóng, chính xác, hiệu quả với hướng tiếp cận mới lạ so với phương pháp phân loại truyền thống đó là phân loại sử dụng tập phổ biến.

Đề tài "***DỰ ĐOÁN KHẢ NĂNG ĐĂNG KÝ GỬI KỲ HẠN THÔNG QUA TIẾP THỊ DI ĐỘNG BẰNG PHƯƠNG PHÁP PHÂN LỚP LUẬT KẾT HỢP***" sẽ tập trung vào phân loại khả năng đăng ký tiền gửi có kỳ hạn của khách hàng thông qua điện thoại tiếp thị dựa trên thông tin cá nhân của khách hàng và thông tin của cuộc gọi tiếp thị dựa trên mô hình máy học. Điều này sẽ góp phần giải quyết khó khăn của các ngân hàng thương mại, tăng khả năng cạnh tranh và trải nghiệm của khách hàng đối với các dịch

vụ gửi tiền tiết kiệm của ngân hàng thương mại tại Việt Nam.

2. MỤC TIÊU NGHIÊN CỨU

Xây dựng mô hình học máy có giám sát phân loại sử dụng mẫu phổ biến để dự đoán khả năng đăng ký tiền gửi có kỳ hạn của khách hàng thông qua điện thoại tiếp thị. Mục tiêu cụ thể:

- Thứ nhất, phân tích và đánh giá các yếu tố ảnh hưởng đến quyết định đăng ký tiền gửi có kỳ hạn của khách hàng thông qua điện thoại tiếp thị;
- Thứ hai, đánh giá kết quả mô hình học máy có giám sát phân lớp sử dụng mẫu phổ hiến (Classification using Frequent Pattern) và các mô hình khác như …
- Thứ ba, đề xuất các giải pháp nhằm thu hút tiền gửi có kỳ hạn, góp phần gia tăng huy động vốn tại các ngân hàng thương mại tại Việt Nam.

3. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

3.1. Đối tượng nghiên cứu

Trong đề tài, mô hình dự đoán được xây dựng tập trung vào đối tượng là danh sách khách hàng để tiếp thị thông qua điện thoại di động cho dịch vụ đăng ký tiền gửi có kỳ hạn của một ngân hàng. Đầu ra của mô hình cho biết các khách hàng được đánh giá là tiềm năng, có thể đăng ký dịch vụ sau khi gọi tiếp thị, giúp ngân hàng đưa ra được những chiến lược phù hợp.

3.2. Phạm vi nghiên cứu

Dữ liệu phân tích là bộ dữ liệu có kích thước trung bình (hơn 10000 dòng) đến từ một ngân hàng ở Bồ Đào Nha. Dữ liệu được thu thập từ năm 2008 đến năm 2013, do đó bộ dữ liệu có cả những tác động của cuộc khủng hoảng tài chính toàn cầu vào năm 2008. Dữ liệu thu thập bao gồm các thông tin như: thông tin cá nhân, hiệu quả trong chiến dịch telemarketing trước, số liệu trong chiến dịch telemarketing lần này.

4. PHƯƠNG PHÁP NGHIÊN CỨU

Phương pháp nghiên cứu định lượng: Cơ sở lý thuyết về xác suất thống kê, các kiến thức về trực quan hóa dữ liệu, tiền xử lý dữ liệu. Nghiên cứu các giải thuật khai phá dữ liệu, đặc biệt là các giải thuật khai phá luật kết hợp như: Thuật toán Apriori, Thuật toán ECLAT, Thuật toán FP-Growth… và ứng dụng của chúng vào bài toán phân lớp (Classification using Associations) với các phương pháp như: CBA, CMAR, CPAR…

5. KẾT CẤU ĐỀ TÀI

Đề tài này gồm 5 chương:

- ***Chương 1:*** Trình bày tổng quan đề tài, tính cần thiết của việc phân tích, dự đoán khả năng đăng ký tiền gửi có kỳ hạn thông qua điện thoại tiếp thị.
- ***Chương 2:*** Trình bày tổng quát về cơ sở lý thuyết: mẫu phổ biến, luật kết hợp… Cụ thể là bài toán Classification based on Associations (CBA) và các hướng tiếp cận.
- ***Chương 3:*** Phát biểu bài toán cần giải quyết trong đề tài. Giới thiệu và mô tả tập dữ liệu được sử dụng.
- ***Chương 4:*** Xây dựng và đánh giá các mô hình dự đoán khả năng đăng ký tiền gửi có kỳ hạn của khách hàng thông qua điện thoại tiếp thị.
- ***Chương 5:*** Trình bày kết luận, đưa ra các hạn chế và hướng phát triển cho đề tài.

CHƯƠNG 2: PHÂN LỚP SỬ DỤNG MẪU PHỔ BIẾN

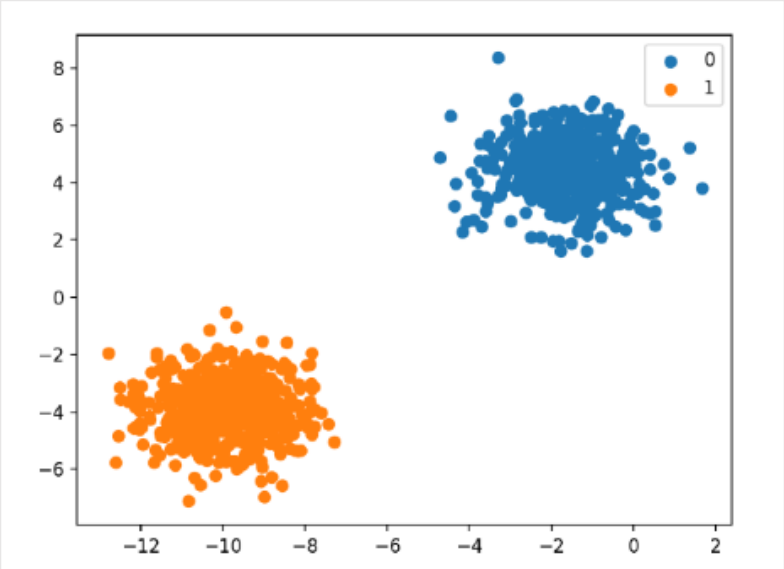
1. BÀI TOÁN PHÂN LỚP

Theo phương thức học, các thuật toán Machine Learning thường được chia làm 4 nhóm: Supervised learning (Học có giám sát), Unsupervised learning (Học không giám sát), Semi-supervised learning (Học bán giám sát) và Reinforcement learning (Học củng cố).

Học có giám sát là các thuật toán dự đoán đầu ra của một dữ liệu mới dựa trên các cặp đã biết từ trước. Cặp dữ liệu còn được gọi là (feature, target). Supervised learning là nhóm phổ biến nhất trong các thuật toán máy học. Khi có một tập hợp biến đầu vào X và một tập hợp nhãn tương ứng Y, các cặp $(x_1, y_1) \in X, Y$ được gọi là tập training data (dữ liệu huấn luyện). Từ tập này, ta cần xây dựng một hàm số ánh xạ mỗi phần tử từ tập X sang một phần tử tương ứng của tập Y. Mục đích là tìm hàm f thật tốt để khi có một dữ liệu x mới, chúng ta có thể tính được nhãn tương ứng của nó $y = f(x)$. Thuật toán học có giám sát được tiếp tục chia nhỏ thành hai loại chính là Phân loại (Classification) và Hồi quy (Regression). Trong đề tài này chúng ta tìm hiểu về bài toán Phân loại.

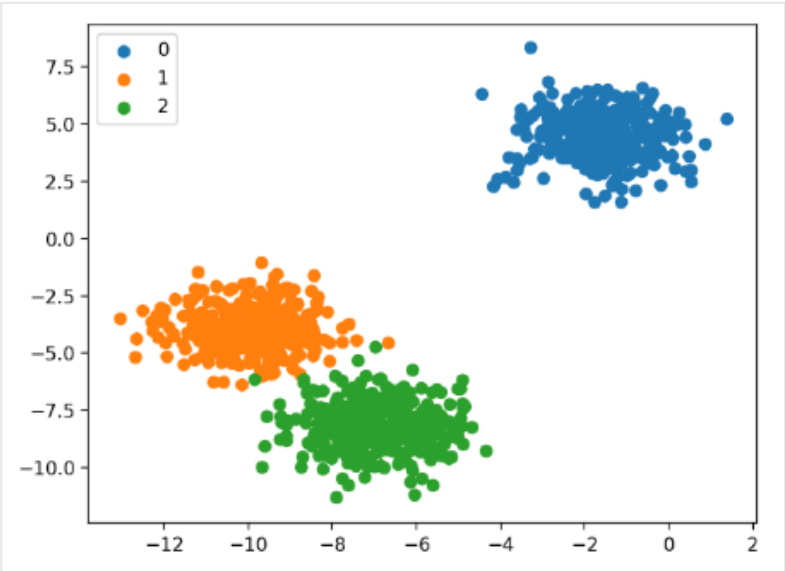
Bài toán được gọi là classification nếu các label của dữ liệu đầu vào được chia thành một số hữu hạn nhóm. Ví dụ: Xác định một email có phải spam hay không, xác định xem một khách hàng có khả năng thanh toán nợ hay không. Một số thuật toán được sử dụng trong bài toán Classification:

- Phân lớp nhị phân (Binary classification) là bài toán phân lớp dữ liệu cơ bản nhưng có nhiều ứng dụng thực tiễn. Phân lớp nhị phân sẽ phân chia dữ liệu thành hai lớp độc lập. Một số bài toán thực tiễn của phân lớp nhị phân như phân lớp dữ liệu liên quan đến bệnh tập: nhiễm bệnh hay không nhiễm bệnh, phân loại tin nhắn: tin nhắn rác và tin nhắn thông thường, phân tích đánh giá phản hồi của khách hàng: tích cực và tiêu cực,... Một số thuật toán được sử dụng cho bài toán phân lớp nhị phân là: Hồi quy Logistic (Logistic Regression), kNN (k-nearest neighbors), cây quyết định (decision tree), máy vector hỗ trợ (Support Vector Machine - SVM), Naive Bayes…



Hình 1: Đồ thị phân tán (scatter plot) của lớp nhị phân

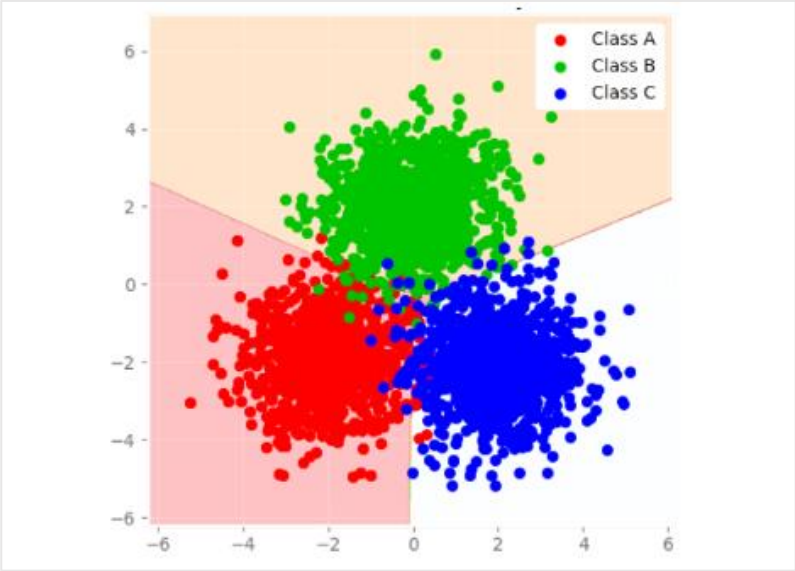
- Trong phân lớp đa lớp (multi-class classification), dữ liệu được phân chia thành nhiều nhóm khác nhau dựa vào các nhãn được chỉ định. Số lượng các nhãn có thể rất lớn và phụ thuộc vào từng bài toán phân lớp. Một ví dụ thực tiễn của bài toán phân lớp đa lớp là bài toán dịch thuật văn bản. Mỗi một từ trong câu có thể được dự đoán liên quan đến bài toán phân lớp đa lớp. Trong đó, số lượng các lớp tương ứng với số lượng từ vựng trong từ điển. Một số thuật toán phổ biến cho phân lớp đa lớp như kNN, cây quyết định, Naive Bayes, rừng ngẫu nhiên (Random Forest). Một số phương pháp sử dụng cho phân lớp nhị phân có thể mở rộng để áp dụng cho bài toán phân lớp đa lớp.



Hình 2: Đồ thị phân tán (scatter plot) của lớp đa lớp

- Phân lớp đa nhãn (multi-label classification) cũng liên quan đến nhiệm vụ phân lớp dựa trên nhiều nhãn khác nhau, nhưng một hoặc nhiều nhãn khác nhau có

thể được sử dụng để tiên đoán cho mỗi một mẫu. Một ví dụ về phân lớp đa nhãn là phân lớp ảnh. Khi đó, trong một bức ảnh cho trước có thể chứa nhiều đối tượng khác nhau như người, xe, hoa, v.v. Mô hình phân lớp đa nhãn có thể tiên đoán sự hiện diện của nhiều đối tượng đó. Các thuật toán sử dụng cho phân lớp nhị phân và phân lớp đa lớp không thể áp dụng trực tiếp cho bài toán phân lớp đa nhãn. Các phương pháp phổ biến thường được dùng cho phân lớp đa nhãn là cây quyết định đa nhãn, rừng ngẫu nhiên đa nhãn.



Hình 3: Đồ thị phân tán (scatter plot) của lớp đa nhãn

2. MẪU PHỔ BIẾN – LUẬT KẾT HỢP

2.1. Khai phá mẫu phổ biến

Khai thác mẫu phổ biến trong khai thác dữ liệu là quá trình xác định các mẫu hoặc liên kết trong một tập dữ liệu xảy ra thường xuyên. Khai phá mẫu phổ biến là một nhiệm vụ thiết yếu trong khai thác dữ liệu nhằm khám phá các mẫu hoặc tập lặp đi lặp lại trong một tập dữ liệu được cung cấp. Khai thác tập phổ biến được Agrawal đề xuất lần đầu tiên vào năm 1993 trong một bài viết về phân tích thị trường ứng dự kỹ thuật khai thác luật kết hợp. Nhiệm vụ của luật khai thác luật kết hợp là tìm ra các mối liên hệ của các mặt hàng khác nhau được khách hàng chọn mua, thông qua việc xác định sự xuất hiện đồng thời của các mặt hàng đó trong cùng hóa đơn mua hàng tại cửa hàng. Ví dụ như dựa vào việc phân tích hóa đơn mua hàng, người ta phát hiện ra rằng những người mua tã thường mua kèm bia trong cùng hóa đơn. Thông tin về những sản phẩm xuất hiện trong cùng hóa đơn có thể được sử dụng để phân tích thị trường riêng biệt của từng cửa hàng, lên kế hoạch kinh doanh, sắp xếp gian hàng sao cho tiện dụng nhất cho

khách hàng, phân nhóm khách hàng… Kể từ khi xuất hiện, bài toán khai thác dữ liệu phổ biến đã nhận được sự quan tâm của giới nghiên cứu với hàng trăm công trình khác nhau, từ nghiên cứu thuật toán khai thác cho đến các lĩnh vực ứng dụng. Mặc dù có nhiều công trình được đề xuất để giải quyết vấn đề khai thác tập phổ biến, nhưng thiết kế phương pháp khai thác hiệu quả với từng loại dữ liệu vẫn là bài toán còn nhiều thách thức. Đến nay, có nhiều phương pháp khai thác tập phổ biến

- Phương pháp Apriori (Agrawal et al., 1994)
- Phương pháp IT-tree (Zaki et al., 1997)
- Phương pháp FP-tree (Han et al., 2000)
- Ngoài ra còn có một số phương pháp được đề xuất như: LCM, DCI, PrePost…

2.2. Khai phá luật kết hợp

Ví dụ trong một cửa hàng tạp hóa, bạn sẽ muốn biết được một khách hàng (KH) khi đến cửa hàng của mình sẽ mua những món đồ gì. Để làm được điều này cần nhìn vào dữ liệu về các giao dịch của KH, qua đó sẽ thấy được tần suất xuất hiện của các món đồ khách hàng đã mua và những món đồ nào thường được KH mua trong cùng một lần mua sắm. Ví dụ, KH thường mua bia lon và đồ nhậu mỗi lần đến mua sắm tại cửa hàng tạp hóa của bạn, điều này thể hiện 2 đồ vật này đã tạo ra mối quan hệ (association rule) và mối quan hệ này có thể hỗ trợ bạn trong việc tiên đoán hành vi mua sắm tiếp theo của KH khi mua hàng.

Giả dụ việc khách hàng mua bia lon và đồ nhậu tại cửa hàng của bạn (dựa vào dữ liệu về giao dịch của KH) có xác suất là 2% tổng số giao dịch và cơ hội để KH mua thêm đồ nhậu khi đã mua bia là 80% vậy thì hàng vi mua sắm bia lon và đồ nhậu được miêu tả như sau: Bia lon → đồ nhậu[support = 2%, confidence = 80%]

- Support: tần suất để hành vi mua sắm xuất hiện trong toàn bộ các giao dịch mua sắm của KH
- Confidence: cơ hội xảy ra việc mua sắm đồ vật tiếp theo trong chuỗi đồ mua sắm của hành vi
- Mức support tối thiểu (minimum support threshold): tần suất thấp nhất của hành vi để thỏa mãn được sự quan tâm của người phân tích
- Mức Confidence tối thiểu (minimum confidence threshold): mức thấp nhất của cơ hội mua sắm để thỏa mãn được sự quan tâm của người phân tích

Về nguyên tắc, Association rules phải trải qua 2 bước:

- Tìm tất cả các giao dịch (tập các đồ vật) phổ biến: các giao dịch này phải thỏa mãn điều kiện về mức support tối thiểu.
- Tạo ra những Association rules mạnh từ các giao dịch phổ biến: các giao dịch có association rules mạnh phải thỏa mãn cả điều kiện về support tối thiểu và confidence tối thiểu.

Ví dụ về khai phá luật kết hợp bằng thuật toán ECLAT, [minSup, minConf] = [15%, 50%] với dữ liệu giao dịch:

ID giao dịch	Danh sách các đồ vật
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Bảng 1: Ví dụ về khai phá luật kết hợp bằng thuật toán ECLAT

BƯỚC 1: TÌM CÁC GIAO DỊCH (TẬP ĐỒ VẬT) PHỔ BIẾN

1. Đồ vật trong VDF:

Đồ vật	Giao dịch
I1	{T100; T400; T500; T700; T800; T900}
I2	{T100; T200; T300; T400; T600; T800; T900}
I3	{T300; T500; T600; T700; T800; T900}
I4	{T200; T400}
I5	{T100; T800}

Bảng 2: 1-Itemsets

2. Đồ vật trong VDF:

Đồ vật	Giao dịch
{I1,I2}	{T100; T400; T800; T900}
{I1,I3}	{T500; T700; T800; T900}
{I1,I4}	{T400}
{I1,I5}	{T100; T800}
{I2,I3}	{T300; T600; T800; T900}
{I2,I4}	{T200; T400}
{I2,I5}	{T100; T800}
{I3,I4}	Ø
{I3,I5}	{T800}
{I4,I5}	Ø

Bảng 3: 2-Itemsets

Loại bỏ các tập đồ vật có số giá trị nhỏ hơn minFreq = 15% * 9 = 1.35

3. Đồ vật trong VDF:

Đồ vật	Giao dịch
{I1,I2,I3}	{T800; T900}
{I1,I2,I5}	{T100; T800}

Bảng 4: 3-Itemsets

BƯỚC 2: TẠO LUẬT KẾT HỢP MẠNH TỪ CÁC GIAO DỊCH PHỔ BIẾN:

A → B	A	B	Confidence
{I1,I2} → {I3}	{I1,I2}	{I3}	50%
{I1,I2} → {I5)	{I1,I2}	{I5)	50%
{I3} → {I1,I2}	{I3}	{I1,I2}	33.33%
{I5} → {I1,I2}	{I5}	{I1,I2}	100%
…	…	…	…

Bảng 5: Luật kết hợp mạnh từ các giao dịch phổ biến

Như vậy, ta có thể rút ra các luật kết hợp lớn hơn hoặc bằng minConf = 50% là các luật kết hợp mạnh như: {I1,I2} → {I3}; {I3} → {I1,I2}; {I5} → {I1,I2}...

Khai phá luật kết hợp có thể ứng dụng trong các lĩnh vực sau:

- Market Basket Analysis: Khai phá luật kết hợp thường được sử dụng trong phân tích giỏ hàng để xác định các mẫu trong dữ liệu bán hàng. Các nhà bán lẻ có thể sử dụng các thông tin này để tạo ra các chiến lược tiếp thị và khuyến mãi nhằm mục tiêu tăng doanh số bán hàng.
- Quản trị quan hệ khách hàng (CRM): Các doanh nghiệp có thể sử dụng các luật liên kết để xác định các mẫu trong hành vi của khách hàng. Điều này có thể được sử dụng để cải thiện khả năng giữ chân khách hàng và tăng độ trung thành của khách hàng.
- Phát hiện gian lận: Cũng có thể sử dụng khai phá luật kết hợp để phát hiện hoạt động gian lận, chẳng hạn như gian lận thẻ tín dụng hoặc gian lận bảo hiểm. Thuật toán có thể xác định các mẫu trong dữ liệu chỉ ra hành vi gian lận, cho phép các doanh nghiệp thực hiện hành động thích hợp.
- Hệ thống khuyến nghị: Sử dụng khai phá luật kết hợp trong các hệ thống khuyến nghị để đề xuất sản phẩm hoặc dịch vụ dựa trên lịch sử mua hàng hoặc duyệt web trước đó của khách hàng như trong phân tích giỏ hàng thị trường.

Chính những lợi ích trên mà khai phá luật kết hợp cung cấp nhiều ưu thế cho các doanh nghiệp, như những lợi thế được nhóm đưa ra dưới đây:

- Khám phá mẫu: Sử dụng khai phá luật kết hợp để khám phá các mẫu và mối quan hệ ẩn giữa các dữ liệu có thể không rõ ràng . Điều này có thể cung cấp cái nhìn sâu sắc về hành vi khách hàng, xu hướng thị trường…
- Phân tích dữ liệu hiệu quả: Đây là một cách hiệu quả để phân tích các bộ dữ liệu lớn và xác định các mẫu có thể khó phân biệt. Nó có thể nhanh chóng sàng lọc một lượng lớn dữ liệu và xác định các mối quan hệ và các mối tương quan quan trong.
- Những yếu tố trên góp phần cho doanh nghiệp ra quyết định hiệu quả, cải thiện được trải nghiệm khách hàng và tăng ưu thế cạnh tranh trên thị trường.

Mặc dù có những ưu điểm nhưng khai phá luật kết hợp cũng có một số hạn chế và bất lợi như sau:

- Khám phá sai: Khai phá luật liên kết có thể tạo ra một số lượng lớn các luật liên

kết. Ở đây, các luật có thể có ích hoặc không có ý nghĩa, một số là ngẫu nhiên và có thể không đại diện cho các mẫu hoặc mối quan hệ thực tế trong dữ liệu.

- Phạm vi hạn chế: Khai phá luật liên kết được thiết kế chủ yếu để xác định mối quan hệ nhị phân giữa các biến và có thể không phát hiện được các mẫu hoặc các mối quan hệ phức tạp hơn. Nó cũng có thể bỏ lỡ các mối quan hệ quan trọng không được ghi lại bởi dữ liệu.
- Vấn đề chất lượng dữ liệu: Chính hai khuyết điểm trên đã đưa ra một yêu cầu về chất lượng dữ liệu cao, đáng tin cậy để tạo ra kết quả chính xác cho việc khai phá luật kết hợp. Nếu dữ liệu không đầy đủ, không chính xác hoặc không nhất quán, các luật tạo ra có thể không đáng tin cây hoặc gây hiểu nhầm.
- Tính toán chuyên sâu: Các thuật toán trong khai phá luật kết hợp có thể chuyên sâu về mặt tính toán, đặc biệt là khi xử lý các tập dữ liệu lớn. Điều này có thể dẫn đến thời gian xử lý lâu và đòi hỏi nhiều tài nguyên máy tính.

Nhìn chung, khi khai phá luật kết hợp có thể cung cấp nhiều hiểu biết có giá trị về dữ liệu. Tuy nhiên, cần xem xét cẩn thận dữ liệu và áp dụng các kỹ thuật phân tích thích hợp có thể giúp đảm bảo rằng các kết quả được tạo ra bởi khai phá luật kết hợp chính xác và mang nhiều ý nghĩa

3. PHÂN LỚP SỬ DỤNG MẪU PHỔ BIẾN VÀ LUẬT KẾT HỢP

Phân lớp kết hợp lần đầu tiên được đề xuất bởi Bing Liu và cộng sự. Ông định nghĩa một mô hình với các luật là “the right-hand side is constrained to be the attribute of the classification class”. Có thể hiểu, phần phía bên phải trong luật kết hợp là thuộc tính của biến mục tiêu. Mô hình phân lớp kết hợp là một mô hình học có giám sát sử dụng các luật kết hợp để gán giá trị mục tiêu.

Mô hình được tạo ra bởi phân lớp kết hợp và được sử dụng để gán nhãn các quan sát mới bao gồm các luật kết hợp tạo ra nhãn. Do đó, chúng cũng được hiểu: nếu một quan sát đáp ứng các điều kiện nhất định (được chỉ định ở phía bên trái của các luật, hay còn được gọi là …), nó được gán nhãn, đánh giá theo các mục ở phía bên phải ở các luật kết hợp. Hầu hết các mô hình phân lớp kết hợp đọc danh sách một cách tuần tự và áp dụng luật trùng khớp đầu tiên để gán nhãn cho quan sát mới. Phân lớp luật kết hợp kế thừa một số chỉ số từ luật kết hợp như là Support hoặc Confidence, có thể sử dụng các chỉ số này để xếp hạng hoặc lọc các luật kết hợp trong mô hình và đánh giá chất lượng của chúng.

Nhìn chung, phân lớp kết hợp bao gồm các bước sau:

1. Khai phá dữ liệu cho các tập phổ biến, nghĩa là tìm ra các cặp thuộc tính - giá trị thường xuất hiện trong tập dữ liệu.
2. Phân tích các tập phổ biến để sinh ra các luật kết hợp cho mỗi lớp, điều này phải thỏa mãn điều kiện về độ tin cậy (confidence) và độ hỗ trợ (support) đề ra.
3. Tổ chức các luật để tạo ra một bộ phân loại dựa trên luật kết hợp.

Có nhiều phương pháp phân lớp luật kết hợp khác nhau. các phương pháp phân loại dựa trên luật kết hợp khác nhau chủ yếu là ở cách tiếp cận được sử dụng để khai thác tập phổ biến và các phân tích, sử dụng các kết hợp để phân loại. Dưới đây là một số phương pháp phổ biến:

- ***CBA (Classification based on Associations):*** Sử dụng các kỹ thuật luật kết hợp để phân loại dữ liệu, được chứng minh là chính xác hơn các kỹ thuật phân loại truyền thông. Tuy nhiên, phương pháp này nhạy cảm với ngưỡng hỗ trợ tối thiểu (minimum support). Khi ngưỡng này thấp hơn được chỉ định, thì một số lượng lớn các luật sẽ được tạo ra. (Liu et al., 1998)
- ***CMAR (Classification based on Multiple Association Rules):*** Được đề xuất lần đầu vào năm 2001 bởi Wenmin Li và công sự thông qua bài nghiên cứu *"CMAR: accurate and efficient classification based on multiple class-association rules"*. Khác biệt giữa CMAR so với CBA là ở chiến lược trong khai phá các tập phổ biến và xây dựng bộ phân loại. Nó cũng sử dụng một số chiến lược pruning với sự trợ giúp của cấu trúc cây để lưu trữ và truy xuất luật kết hợp một cách hiệu quả. CMAR áp dụng một biến thể của thuật toán FP-Growth để tìm ra một bộ các luật kết hợp thỏa mãn minimum support và minimum confidence. CMAR sử dụng FP-tree để duy trì việc phân bổ nhãn lớp giữa các tuple thỏa mãn mỗi tập phổ biến. Bằng cách này, nó có thể kết hợp việc tạo ra luật kết hợp cùng với khai thác tập phổ biến trong một bước. Giả sử, có một bộ X được cấp để phân loại và chỉ có một luật thỏa mãn thì chúng ta chỉ cần gán nhãn lớp của luật kết hợp. Còn nếu có nhiều hơn một luật kết hợp thỏa mãn X, CBA sẽ gán nhãn lớp có confidence cao nhất trong số các luật kết hợp. Thay vào đó, CMAR sẽ xem xét đến nhiều luật kết hợp khi đưa ra dự đoán, phương pháp này chia các luật kết hợp thỏa mãn thành các nhóm theo nhãn lớp. Tất cả các luật trong mỗi nhóm đều có chung nhãn và mỗi nhóm có một nhãn lớp riêng biệt. Sau đó, CMAR sử dụng trọng số đo lường Chi-Square để tìm ra nhóm

các luật mạnh nhất, dựa trên tương quan thống kê của các luật trong mỗi nhóm và gán nhãn X theo nhóm mạnh nhất. Bằng cách này, nó xem xét đến nhiều luật so với là một luật có độ tin cậy cao nhất khi dự đoán nhãn của một tuple mới. Bằng thực nghiệm, CMAR có độ chính xác trung bình cao hơn một chút so với CBA. Thời gian chạy, khả năng mở rộng và sử dụng bộ nhớ của nó được cho là hiệu quả hơn (Li et al., 2001).

- ***CPAR (Classification based on Predictive Association Rules):*** Thực hiện một cách tiếp cận khác để tạo ra luật kết hợp, dựa trên thuật toán tạo ra luật để phân loại được gọi là FOIL. FOIL xây dựng các luật để phân biệt các giá trị dương với các bộ giá trị âm. Đối với các bài toán đa lớp, FOIL được áp dụng cho từng lớp. Nghĩa là đối với một lớp (ví dụ là lớp C), tất cả các tuple của lớp C được xem xét là tuple dương, trong khi phần còn lại được coi là tuples âm. Mỗi khi một luật kết hợp được tạo ra, tất các các tuple dương mà nó thỏa mãn hay hàm chứa sẽ bị xóa cho đến khi tất cả các tuple dương trong tập dữ liệu đều được bao phủ. Bằng cách này, sẽ ít luật được tạo ra hơn. CPAR sẽ làm nhẹ bước này bằng cách cho phép các tuple bị bao phủ vẫn được xem xét những sẽ giảm trọng số. Quá trình này được lặp lại cho mỗi lớp. Trong quá trình phân loại, CPAR sử dụng luật khác hơn so với CMAR. Nếu có nhiều hơn một luật kết hợp thỏa mãn bộ X mới thì các luật đó sẽ được chia thành các nhóm theo lớp tương tự như CMAR. Tuy nhiên, CPAR sẽ sử dụng k luật tốt nhất thay vì tất cả các luật của nhóm. Bằng cách xem xét k luật tốt nhất thay vì tất cả các luật kết hợp của nhóm, nó sẽ tránh được ảnh hưởng của các luật có chỉ số đánh giá thấp hơn. Độ chính xác của CPAR được chứng minh là gần bằng CMAR. Nhưng vì CPAR tạo ra ít luật kết hợp hơn CMAR nên nó cho thấy hiệu quả tốt hơn nhiều với các tập dữ liệu huấn luyện lớn. (Yin et al., 2003)

Trong đề tài "Dự đoán khả năng đăng ký gửi kỳ hạn thông qua tiếp thị di động bằng phương pháp phân lớp luật kết hợp" nhóm sẽ sử dụng phương pháp CBA để tiếp cận bài toán Classification using Frequent Pattern đơn giản. Chi tiết phương pháp sẽ được nhóm mô tả chi tiết trong phần sau.

4. CLASSIFICATION BASED ON ASSOCIATIONS (CBA)

Thuật toán CBA (Classification Based on Associations) là một trong những thuật toán sớm nhất và đơn giản nhất đề phân lớp kết hợp. CBA sử dụng cách tiếp cận lặp lại để khai thác itemsets phổ biến, tương tự như với giải thuật Apriori. Số lần thực hiện lặp

lại bằng với độ dài của luật dài nhất được tìm thấy. Bộ luật hoàn chỉnh thỏa mãn độ tin cậy tối thiểu và ngưỡng hỗ trợ tối thiểu, sau đó được phân tích để đưa vào bộ phân loại. CBA sử dụng phương pháp heuristic để xây dựng bộ phân loại, trong đó các luật được sắp xếp theo mức độ ưu tiên giảm dần dựa trên độ tin cậy và độ hỗ trợ của chúng. Thực nghiệm cho thấy CBA chính xác trên nhiều tập dữ liệu. (Jiawei Han et al., 2011)

Bài nghiên cứu "Integrating Classification and Association Rule Mining" của Bing Liu và cộng sự mô tả chi tiết hơn về thuật toán này. Mục tiêu của bài nghiên cứu là xây dựng trình phân loại dựa trên CARs (Class Association Rules), thuật toán này được gọi là thuật toán CBA, chính là phương pháp CBA chúng ta đang tìm hiểu .Thuật toán bao gồm hai phần, một là sinh luật (được gọi là CBA-RG), phần này dựa trên giải thuật Apriori để tìm ra các luật kết hợp bên trong bộ dữ liệu (Agrawal and Srikant 1994). Và phần còn lại là xây dựng bộ phân loại (được gọi là CBA-CB).

4.1. Thuật toán CBA-RG

Quy trình chính của CBA-RG là tìm tất cả các ruleitem có độ hỗ trợ lớn hơn minsup (ngưỡng hỗ trợ tối thiểu). Một ruleitem có dạng:

$$< condset, y >$$

condset là bộ các item, $y \in Y$ là nhãn lớp. Độ hỗ trợ của condset (condsupCount) là số các trường hợp trong tập dữ liệu có chưa condset. Độ hỗ trợ của ruleitem (rulesupCount) là số trường hợp trong tập dữ liệu có chưa condset và được gán nhãn lớp y. Mỗi ruleitem về cơ bản sẽ biểu diễn một luật kết hợp:

$$condset \rightarrow y$$

độ hỗ trợ là $(rulesupCount \ / \ |D|) * 100\%$ với $|D|$ là kích thước của tập dữ liệu và độ tin cậy là $(rulesupCount/condsupCount) * 100\%$. Các ruleitem thỏa mãn minsup được gọi là các ruleitem phổ biến, phần còn lại được gọi là các ruleitem không phổ biến.

Thuật toán CBA-RG tạo ra tất cả các ruleitem phổ biến bằng cách thực hiện nhiều lần truyền dữ liệu. Trong lần đầu tiên, nó tính độ hỗ trợ (support) của từng item riêng lẻ và xác định xem nó có phổ biến không. Trong mỗi lần sau, nó bắt đầu với tập hợp các item được xác định là phổ biến ở lần trước. Nó sử dụng tập hợp này để tạo ra các ruleitem mới có thể phổ biến được gọi là các ruleitem ứng viên. Chỉ số support sẽ được tính toán xuyên suốt quá trình truyền dữ liệu. Ở cuối quá trình, nó sẽ xác định ruleitem ứng viên nào là phổ biến. Từ tập hợp các ruleitem phổ biến này tạo ra được các luật kết hợp (CARs).

4.2. Xây dựng bộ phân loại (CBA-CB)

Sau khi tất cả các luật kết hợp (CARs) được tìm thấy, trình phân lớp sử dụng các luật này được xây dựng. Có nhiều phương pháp khả thi để xây dựng một trình phân lớp dựa trên các luật. Đối với CBA, một bộ các luật có độ tin cậy cao được chọn từ CARs để tạo thành mô hình phân lớp. Sự chọn lựa các luật kết hợp được dựa trên tổng thứ tự được xác định trên các luật.

Quy ước: Có 2 luật i và j ($i \neq j$). Ta nói i có độ ưu tiên (precedence) cao hơn j nếu:

1. Độ tin cậy của i cao hơn j;
2. Độ tin cậy i và j bằng nhau, nhưng độ hỗ trợ của i cao hơn j;
3. Độ tin cậy và độ hỗ trợ của i bằng j, nhưng i được tạo ra trước j.

Gọi R là tập của CARs và D là tập dữ liệu huấn luyện. Ý tưởng cơ bản của xây dựng trình phân lớp của CBA là chọn một bộ các luật có quyền ưu tiên cao trong R để phủ tập D. Một phân lớp CBA có dạng:

$$< r_1, r_2, \dots, r_n, default\_class >$$

với ri thuộc R, ra khác rb nếu b > a.. Khi phân loại một trường hợp chưa từng thấy, quy tắc đầu tiên thỏa mãn trường hợp đó sẽ gán nhãn cho nó. Nếu không có quy tắc nào áp dụng cho trường hợp này thì nó sẽ được gán một nhãn lớp mặc định. Mã giả phiên bản đơn giản của thuật toán xây dựng như trình bày ở trên:

```
R = sort(R); /* according the precedence ϕ */
for each rule r ∈ R in sequence do
   if there are still training examples in D AND r classifies
         at least one example correctly then
      delete all training examples covered by r from D;
      add r to the classifier
   end
end
add the majority class as the default class to the classifier.
```

Hình 4: Pseudocode của bước CBA-CB:M1 trong thuật toán CBA

(Liu et el., 2001)

Xây dựng bộ phân loại trong CBA có hai phiên bản: đầu tiên là phiên bản đơn giản như mã giả ở trên (hay còn được gọi là CBA-CB-M1), phiên bản còn lại là phiên bản cải tiến (hay còn được gọi là CBA-CB-M2). Các bước của phiên bản M1 cụ thể là:

- ***Bước 1:*** Sắp xếp bộ các luật được sinh ra ở bước CBA-RG theo tổng thứ tự (độ

tin cậy, độ hỗ trợ, thứ tự luật xuất hiện). Để chắc chắn rằng ta sẽ chọn được luật có độ ưu tiên cao nhất.

- ***Bước 2:*** Với mỗi luật r, chúng ta sẽ duyệt qua tập D để tìm ra trường hợp sẽ được giải quyết bởi r (thỏa mãn các condset của luật). Ta đánh dấu luật r khi r phân lớp đúng cho một trường hợp, khi đó r được xem là luật có tiềm năng và được thêm vào tập C. Và trường hợp đã được phân lớp sẽ loại bỏ khỏi tập dữ liệu. Các trường hợp chưa được phân lớp sẽ được gán cho một nhãn mặc định. Cuối cùng là tính toán sai số bằng tổng số phân lớp sai của luật r. Khi không còn luật hay trường hợp huấn luyện nào nữa thì hoàn tất bước chọn luật.
- ***Bước 3:*** Tìm luật mà có ít sai số nhất, các luật thêm vào C sau luật này đều bị loại bỏ. Các luật còn lại và nhãn lớp mặc định là bộ phân lớp được sử dụng.

Thuật toán M1 tuy đơn giản nhưng không hiệu quả, đặc biệt khi bộ dữ liệu không ở trên bộ nhớ chính, điều này khiến thuật toán phải duyệt qua nhiều lần bộ dữ liệu. Thuật toán cải tiến M2 được trình bày thông qua hai lần thực hiện trên D. Thuật toán M2 bao gồm ba giai đoạn:

- ***Giai đoạn 1:*** Với mỗi trường hợp d, chúng ta tìm luật có độ ưu tiên cao nhất khi gán nhãn đúng và gán nhãn sai cho d (gọi là *cRule* và *wRule*). Nếu cRule ưu tiên cao hơn *wRule* thì dữ liệu được phân loại bởi cRule. Nếu wRule có độ ưu tiên cao hơn cRule thì ta ghi lại cả *cRule* và *wRule* để xử lý sau.
- ***Giai đoạn 2:*** Xác định tất cả các quy tắc phân loại sai d có độ ưu tiên cao hơn cRule của d. Nếu *wRule* đã được đánh dấu (nghĩa là đã được chọn làm cRule của một dữ liệu khác), *wRule* sẽ phân loại dữ liệu. Nếu *wRule* chưa được đánh dấu, thì ta sẽ thêm *wRule* vào danh sách các quy tắc có thể thay thế cRule để phân loại dữ liệu.
- ***Giai đoạn 3:*** Chọn tập hợp cuối cùng của các quy tắc để tạo bộ phân loại.
    - Bước 1: Xếp hạng các quy tắc theo độ ưu tiên và loại bỏ các quy tắc không còn phân loại chính xác bất kỳ dữ liệu nào.
    - Bước 2: Loại bỏ các quy tắc gây ra nhiều lỗi hơn.

5. SO SÁNH VỚI THUẬT TOÁN KHÁC

Trong bài nghiên cứu "CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules" của Wenmin Li và nnk. thuật toán CBA và CMAR

được so sánh hiệu suất khi thực hiện kiểm thử trên 26 tập dữ liệu từ UIC Machine Learning Repository:

Data set	# attr	# cls	# rec	C4.5	CBA	*CMAR*
Anneal	38	6	898	94.8	97.9	97.3
Austral	14	2	690	84.7	84.9	86.1
Auto	25	7	205	80.1	78.3	78.1
Breast	10	2	699	95	96.3	96.4
Cleve	13	2	303	78.2	82.8	82.2
Crx	15	2	690	84.9	84.7	84.9
Diabetes	8	2	768	74.2	74.5	75.8
German	20	2	1000	72.3	73.4	74.9
Glass	9	7	214	68.7	73.9	70.1
Heart	13	2	270	80.8	81.9	82.2
Hepatic	19	2	155	80.6	81.8	80.5
Horse	22	2	368	82.6	82.1	82.6
Hypo	25	2	3163	99.2	98.9	98.4
Iono	34	2	351	90	92.3	91.5
Iris	4	3	150	95.3	94.7	94
Labor	16	2	57	79.3	86.3	89.7
Led7	7	10	3200	73.5	71.9	72.5
Lymph	18	4	148	73.5	77.8	83.1
Pima	8	2	768	75.5	72.9	75.1
Sick	29	2	2800	98.5	97	97.5
Sonar	60	2	208	70.2	77.5	79.4
Tic-tac	9	2	958	99.4	99.6	99.2
Vehicle	18	4	846	72.6	68.7	68.8
Waveform	21	3	5000	78.1	80	83.2
Wine	13	3	178	92.7	95	95
Zoo	16	7	101	92.2	96.8	97.1
Average				83.34	84.69	85.22

Bảng 6: So sánh CBA và CMAR về độ chính xác

(Wenmin Li et al., 2001)

Kết quả cho thấy dù là thuật toán ra đời sớm hơn nhưng CBA vẫn thể hiện tốt hơn CMAR ở 9 trong tổng số 26 bộ dữ liệu khi đánh giá trên accuracy. Các bộ dữ liệu còn lại chênh lệch không đáng kể, trung bình độ chính xác của CBA là 84.69% trong khi CMAR là 85.22%. Tuy nhiên khi xét đến bộ nhớ chính sử dụng và thời gian chạy, CMAR đạt 77.12% tiết kiệm dung lượng chính sử dụng so với CBA. Khi so sánh thời gian chạy trên 6 bộ dữ liệu, CBA dành phần lớn thời gian chạy cho Input/Ouput, kết quả được hiển thị ở hai hình dưới đây:

Dataset	# attr	# cls	# rec	CBA mem (M)	*CMAR* mem (M)	saving (%)
Auto	25	7	205	488	101	79.30%
Hypo	25	2	3163	330	90	72.73%
Iono	34	2	351	334	86	74.25%
Sick	29	2	2800	321	85	73.52%
Sonar	60	2	208	590	88	85.09%
Vehicle	18	4	846	590	88	85.09%
Average				393.33	90	77.12%

Bảng 7: So sánh CBA và CMAR về mức sử dụng bộ nhớ chính

(Wenmin Li et al., 2001)

Dataset	# attr	# cls	# rec	CBA runtime	*CMAR* runtime
Auto	25	7	205	612s	408s
Hypo	25	2	3163	92s	19s
Iono	34	2	351	150s	89s
Sick	29	2	2800	74s	13s
Sonar	60	2	208	226s	145s
Vehicle	18	4	846	284s	192s

Bảng 8: Thời gian chạy của CBA và CMAR

(Wenmin Li et al., 2001)

Trong bài nghiên cứu "CPAR: Classification based on Predictive Association Rules" của Xiaoxin Yin và Jiawei Han, so sánh hiệu suất khi thực hiện kiểm thử trên 26 tập dữ liệu từ UIC Machine Learning Repository như của Wenmin Liu với ba phương pháp CBA, CMAR và CPAR, có thể thấy thuật toán CPAR có accuracy trung bình thấp hơn CMAR, chênh lệch không nhiều so với CBA. Tuy nhiên, về thời gian chạy CPAR thể hiện tốt hơn rất nhiều so với thuật toán CMAR, điều này là do số lượng luật kết hợp được sinh ra ở phương pháp CPAR ít hơn CMAR do sử dụng kỹ thuật k luật tốt nhất thay vì tất cả các luật như CMAR

	RIPPER	CMAR	CPAR
Arithmetic average	0.218	30.24	0.555
Geometric average	0.036	2.877	0.105

Bảng 9: Thời gian chạy của RIPPER, CMAR and CPAR

(Yin & Han., 2003)

	RIPPER	CMAR	CPAR
Arithmetic average	8.20	305	244
Geometric average	5.74	185	106

Bảng 10: Số luật của RIPPER, CMAR and CPAR

(Yin & Han., 2003)

Dataset	c4.5	ripper	cba	cmar	cpar
anneal	94.8	95.8	97.9	97.3	98.4
austral	84.7	87.3	84.9	86.1	86.2
auto	80.1	72.8	78.3	78.1	82.0
breast	95.0	95.1	96.3	96.4	96.0
cleve	78.2	82.2	82.8	82.2	81.5
crx	84.9	84.9	84.7	84.9	85.7
diabetes	74.2	74.7	74.5	75.8	75.1
german	72.3	69.8	73.4	74.9	73.4
glass	68.7	69.1	73.9	70.1	74.4
heart	80.8	80.7	81.9	82.2	82.6
hepatic	80.6	76.7	81.8	80.5	79.4
horse	82.6	84.8	82.1	82.6	84.2
hypo	99.2	98.9	98.9	98.4	98.1
iono	90.0	91.2	92.3	91.5	92.6
iris	95.3	94.0	94.7	94.0	94.7
labor	79.3	84.0	86.3	89.7	84.7
led7	73.5	69.7	71.9	72.5	73.6
lymph	73.5	79.0	77.8	83.1	82.3
pima	75.5	73.1	72.9	75.1	73.8
sick	98.5	97.7	97.0	97.5	96.8
sonar	70.2	78.4	77.5	79.4	79.3
tic-tac	99.4	98.0	99.6	99.2	98.6
vehicle	72.6	62.7	68.7	68.8	69.5
waveform	78.1	76.0	80.0	83.2	80.9
wine	92.7	91.6	95.0	95.0	95.5
zoo	92.2	88.1	96.8	97.1	95.1
Average	83.34	82.93	84.69	85.22	85.17

Bảng 11: Độ chính xác của C4.5, RIPPER, CBA, CMAR and CPAR

(Yin & Han., 2003)

Khi so sánh các thuật toán phân lớp sử dụng luật kết hợp với thuật toán máy học có giám sát truyền thống, cụ thể là phương pháp cây quyết định trong nghiên cứu "Classification Based on Association Rules for Adaptive Web Systems" của Saddys Segrera1 và María N. Moreno. Khi thực hiện kiểm thử với 26 bộ dữ liệu từ UCI, những bộ sử dụng trong các nghiên cứu trước đã đưa ra bằng chứng về tính vượt trội của

phương pháp CPAR so với các thuật toán khác của phân loại kết hợp. Hầu hết các phương pháp phân lớp dựa trên luật kết hợp có kết quả tốt hơn phương pháp cây quyết định ngoại trừ CPAR. Điều này cho thấy đôi khi thuật toán CPAR không ổn định so với hai thuật toán CBA và CMAR.

Algorithms	Accuracy (%)	Execution time (seconds)
Decision tree J48	72.34	8.00
CBA	85.94	17.02
CMAR	91.06	1 519.09
FOIL	84.79	1.42
CPAR	49.85	0.18

Bảng 12: So sánh các thuật toán phân lớp sử dụng luật kết hợp với cây quyết định

(Saddys & María, 2001)

CHƯƠNG 3: TỔNG QUAN NGHIÊN CỨU

1. PHÁT BIỂU BÀI TOÁN

Đây là một bài toán dự đoán xem khách hàng trong tương lai có đăng ký tiền gửi có kỳ hạn hay không, sử dụng phương pháp phân lớp dựa trên luật kết hợp.

Nguồn dữ liệu của bài toán được thu thập từ một ngân hàng Bồ Đào Nha, đã thực hiện các cuộc chiến dịch tiếp thị thông qua điện thoại trong khoảng từ tháng 5 năm 2008 đến tháng 6 năm 2013. Đã có tổng cộng 52.944 cuộc gọi tiếp thị qua điện thoại để chào mời khách hàng mở tài khoản tiền gửi có lãi suất hấp dẫn, với mỗi liên hệ có lưu lại rất nhiều thuộc tính về thông tin liên lạc, thông tin khách hàng.

Bài toán dựa trên các thông tin về khách hàng, xây dựng mô hình dự đoán giúp xác định các khách hàng có khả năng đăng ký sản phẩm cao để ngân hàng có chiến lược tiếp thị trọng điểm, từ đó tối ưu hóa chi phí và doanh thu.

2. HƯỚNG GIẢI QUYẾT VẤN ĐỀ

[Liu et al., 1998] đã đề xuất kết hợp khai phá luật phân loại (classification rule mining) và khai phá luật kết hợp (association rule mining) để xây dựng mô hình dự đoán. Cụ thể:

- **Bước 1:** Tiền xử lý dữ liệu
  - Xử lý giá trị bị thiếu: loại bỏ hoặc thay thế bằng giá trị trung bình
  - Xử lý giá trị ngoại lai: loại bỏ hoặc thay thế bằng giá trị biên gần nhất
  - Làm rời rạc các thuộc tính liên tục thành các khoảng giá trị rời rạc bằng thuật toán entropy discretization
- **Bước 2: Khai phá luật kết hợp**
  - Áp dụng thuật toán CBA-RG để tìm tất cả các luật kết hợp liên quan đến lớp (CARs) với minsup = 1% và minconf = 50%.
  - Loại bỏ các luật không mang nhiều thông tin dự đoán bằng pruning.
- **Bước 3:** Xây dựng mô hình phân loại
  - Sử dụng thuật toán CBA-CB để chọn các luật có độ tin cậy và khả năng bao phủ cao nhất để lấy làm luật phân loại.
  - Xếp các luật theo thứ tự ưu tiên dựa trên độ tin cậy và hỗ trợ.
  - Lựa chọn luật dừng dựa trên sai số trên tập huấn luyện.

3. TÌM HIỂU BỘ DỮ LIỆU

Đây là tập dữ liệu tiếp thị ngân hàng kinh điển được tải lên lần đầu trong Kho Dữ

liệu Học máy UCI. Tập dữ liệu cung cấp thông tin về chiến dịch tiếp thị của một tổ chức tài chính mà ta sẽ phải phân tích để tìm ra cách nhìn nhằm cải thiện các chiến lược tiếp thị trong tương lai cho ngân hàng.

Để tìm ra các chiến lược tốt nhất để cải thiện các chiến dịch tiếp thị cũng như giải đáp câu hỏi làm thế nào để tổ chức tài chính thực hiện các chiến dịch tiếp thị có hiệu quả hơn trong tương lai, ta cần phân tích chiến dịch tiếp thị gần đây nhất của ngân hàng và xác định các mẫu sẽ giúp chúng ta đưa ra kết luận để phát triển các chiến lược trong tương lai.

Bộ dữ liệu được thu thập là thông tin về khách hàng ngân hàng gồm 17 thuộc tính, bao gồm các thông tin cơ bản về khách hàng, lịch sử giao dịch và lịch sử tiếp cận khách hàng thông qua các chiến dịch tiếp thị trước đó. Biến target deposit: cho biết cuộc gọi có thành công không/khách hàng có đăng ký tiền gửi kỳ hạn hay không, được thể hiện bằng các giá trị "yes" (có) và "no" (không). Đây là biến target cần dự đoán.

- **Dữ liệu khách hàng**

  1 - Age: tuổi (numeric)

  2 - Job: loại công việc (danh mục: "admin", "unknown","unemployed", "management", "housemaid", "entrepreneur", "student", "blue-collar", "self-employed", "retired", "technician", "services")

  3 - Marital: tình trạng hôn nhân (danh mục: "married", "divorced", "single")

  4 - Education: trình độ học vấn (danh mục: "unknown", "secondary", "primary", "tertiary")

  5 - Default/mặc định: có nợ xấu tín dụng không? ("yes", "no")

  6 - Balance/số dư: số dư trung bình hàng năm, tính bằng đồng Euro(numeric)

  7 - Housing/nhà ở: có khoản vay mua nhà không? ("yes", "no")

  8 - Loan/khoản vay: có khoản vay cá nhân không? ("yes", "no")

- **Có liên quan đến cuộc gọi cuối cùng:**

9 - Contact/liên hệ: loại liên hệ truyền thông (danh mục: "unknown", "telephone", "cellular")

10 - Day/ngày: ngày liên hệ cuối cùng của tháng (numeric)

11 - Month/tháng: tháng liên hệ cuối cùng của năm (danh mục: "jan", "feb", "mar", …, "nov", "dec")

12 - Duration/thời lượng: thời lượng liên hệ cuối cùng, tính bằng giây (numeric)

- **Các thuộc tính khác:**

13 - Campaign/chiến dịch: số lượng liên hệ được thực hiện trong chiến dịch này và cho khách hàng này (numeric, bao gồm cả lần liên hệ cuối)

14 - Pdays: số ngày đã trôi qua kể từ khi khách hàng được liên hệ lần cuối từ chiến dịch tiếp thị trước đó (numeric, -1 có nghĩa là khách hàng chưa từng được liên hệ trước đây)

15 - Previous: số lượng liên hệ được thực hiện trước chiến dịch này và cho khách hàng này (numeric)

16 - Poutcome: kết quả của chiến dịch tiếp thị trước đó (danh mục: "unknown", "other", "failure", "success")

- **Biến đầu ra (biến target):**

17 - deposit - khách hàng có đăng ký tiền gửi có kỳ hạn không? ("yes", "no")

CHƯƠNG 4: DỰ ĐOÁN KHẢ NĂNG ĐĂNG KÝ GỬI KỲ HẠN BẰNG PHƯƠNG PHÁP PHÂN LỚP LUẬT KẾT HỢP

1. PHÂN TÍCH KHÁM PHÁ DỮ LIỆU

Phân tích khám phá dữ liệu (EDA) là một bước quan trọng trong quá trình phân tích dữ liệu. EDA giúp có cái nhìn đầu tiên về dữ liệu, từ đó mường tượng được độ phức tạp của dữ liệu và vạch ra những bước đầu tiên cần làm.

Trong biểu diễn trực quan dữ liệu, EDA được thực hiện bằng cách sử dụng các kỹ thuật trực quan hóa dữ liệu để khám phá các đặc điểm và mối quan hệ của dữ liệu. Các kỹ thuật này giúp hiểu rõ hơn về dữ liệu, từ đó có thể đưa ra các quyết định sáng suốt hơn trong quá trình phân tích.

Có hai lệnh được sử dụng phổ biến để khám phá dữ liệu trong data visualization là:

- ***head()*** - Xem một số hàng đầu tiên của bộ dữ liệu

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0	unknown	yes
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0	unknown	yes

Hình 5: Một số hàng đầu tiên của bộ dữ liệu

- ***info()*** - Cung cấp thông tin tóm tắt của dữ liệu, bao gồm số lượng hàng, số lượng cột, kiểu dữ liệu của mỗi cột, v.v.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        11162 non-null  int64
 1   job        11162 non-null  object
 2   marital    11162 non-null  object
 3   education  11162 non-null  object
 4   default    11162 non-null  object
 5   balance    11162 non-null  int64
 6   housing    11162 non-null  object
 7   loan       11162 non-null  object
 8   contact    11162 non-null  object
 9   day        11162 non-null  int64
 10  month      11162 non-null  object
 11  duration   11162 non-null  int64
 12  campaign   11162 non-null  int64
 13  pdays      11162 non-null  int64
 14  previous   11162 non-null  int64
 15  poutcome   11162 non-null  object
 16  deposit    11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

Hình 6: Thông tin tóm tắt của dữ liệu

- Bộ dữ liệu bao gồm 11162 dòng, 17 cột. Trong bộ dữ liệu có 10 biến phân loại, 7 biến số.
- **Biến features**: age, job, marital, education, default, balance, housing, loan, contant, day, month, duration, campaign, pdays, previous, outcome.
- **Biến target**: deposit

Xem số lượng danh mục của mỗi thuộc tính, ta sử dụng phương thức ***nunique()***

```
age            76
job            12
marital         3
education       4
default         2
balance      3805
housing         2
loan            2
contact         3
day            31
month          12
duration     1428
campaign       36
pdays         472
previous       34
poutcome        4
deposit         2
```

Quan sát các biến số và biến phân loại:

```
df_num = df.select_dtypes(include=[int , float])
df_cate = df.select_dtypes(include=[object])

print("Biến numerical:\n" + '\n'.join(df_num.columns[1:]))
print("\nBiến categorical:\n" + '\n'.join(df_cate.columns[1:]))
```

Ta thu được kết quả như sau:

```
Biến numerical:
balance
day
duration
campaign
pdays
previous

Biến categorical:
marital
education
default
housing
loan
contact
month
poutcome
deposit
```

Như vậy trong bộ dữ liệu sẽ gồm:

- 6 biến numerical: balance, day, duration, campaign, pdays, previous.
- 9 biến categorical: marital, education, default, housing, loan, contact, month, poutcome, deposit.

Sử dụng phương thức ***describe()*** để thống kê mô tả các biến numerical và categorical:

	age	balance	day	duration	campaign	pdays	previous
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407	0.832557
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282	2.292007
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000	0.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000	0.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000	0.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000	1.000000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000	58.000000

Hình 7: Thống kê mô tả cho các biến numerical

Nhận xét: Tổng kết cho kết quả thống kê:

- Độ tuổi trung bình trong tập dữ liệu là khoảng 41 tuổi bên cạnh đó, độ tuổi phân bố có vẻ khá đồng đều, với giá trị trung vị và phân vị 25, phân vị 75 không chênh lệch nhiều.
- Số dư trung bình trong tài khoản ngân hàng là khoảng 1.528, tuy nhiên, ta có thể thấy độ lệch chuẩn khá là cao khoảng 3.225 và giá trị tối đa là 81.204.
- Số lượng cuộc gọi thường xuyên diễn ra trong khoảng giữa tháng, trung bình là khoảng 15. Dữ liệu phân phối đều qua các ngày trong tháng.
- Thời lượng trung bình của cuộc gọi là khoảng 371,99 giây (gần 6 phút). Có sự biến động lớn trong thời lượng cuộc gọi, từ 2 giây đến 3.881 giây.
- Trung bình mỗi khách hàng được liên hệ khoảng 2 lần. Phần lớn mọi người được liên hệ khá ít trong chiến dịch, với phân vị 75 là 3.
- Trung bình số ngày kể từ lần liên hệ trước đó là khoảng 51 ngày. Có vẻ như phần lớn các các khách hàng không được trước đó, do giá trị -1 xuất hiện ở phân vị 25, 50 và 75.
- Trung bình số lần liên hệ trước đó là khoảng 0,83 lần. Đa số đa số các khách hàng

không được liên hệ trước đó (75% có giá trị là 0).

	job	marital	education	default	housing	loan	contact	month	poutcome	deposit
count	11162	11162	11162	11162	11162	11162	11162	11162	11162	11162
unique	12	3	4	2	2	2	3	12	4	2
top	management	married	secondary	no	no	no	cellular	may	unknown	no
freq	2566	6351	5476	10994	5881	9702	8042	2824	8326	5873

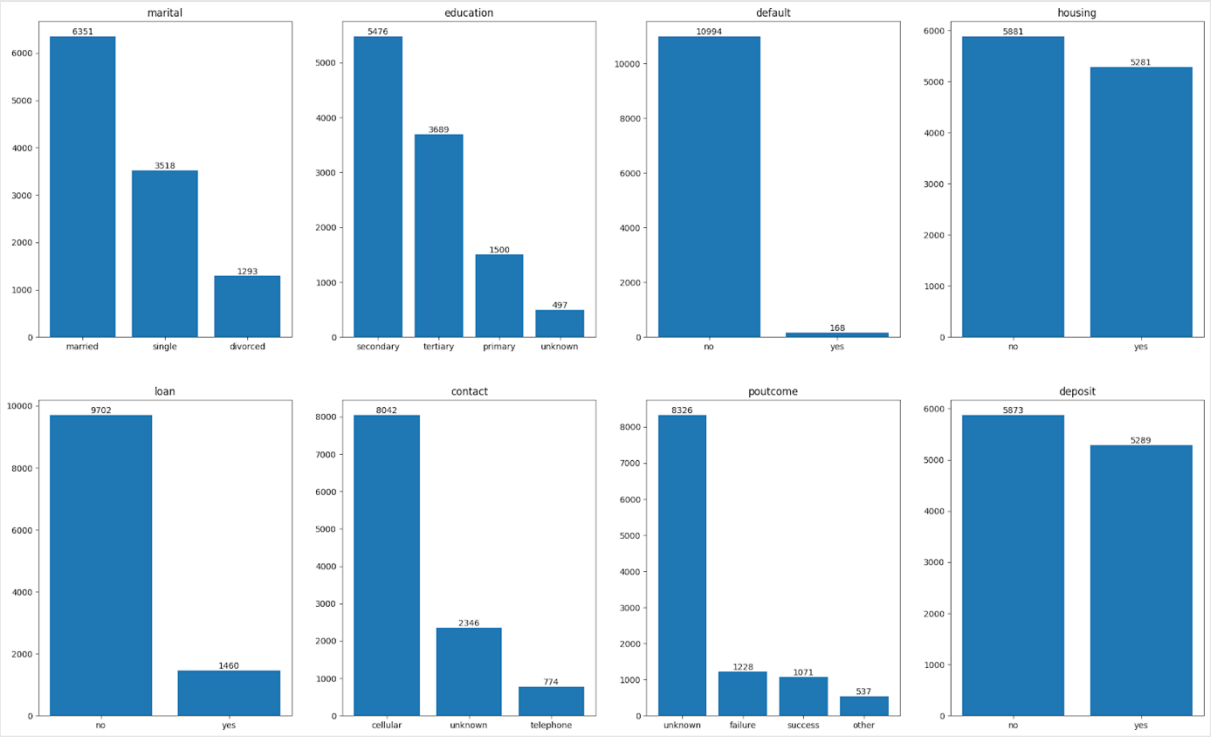
Hình 8: Thống kê các biến Categorical

Thông qua bảng kết quả trên, có thể thấy một số thông tin về phân phối và xu hướng của các biến categorical trong tập dữ liệu:

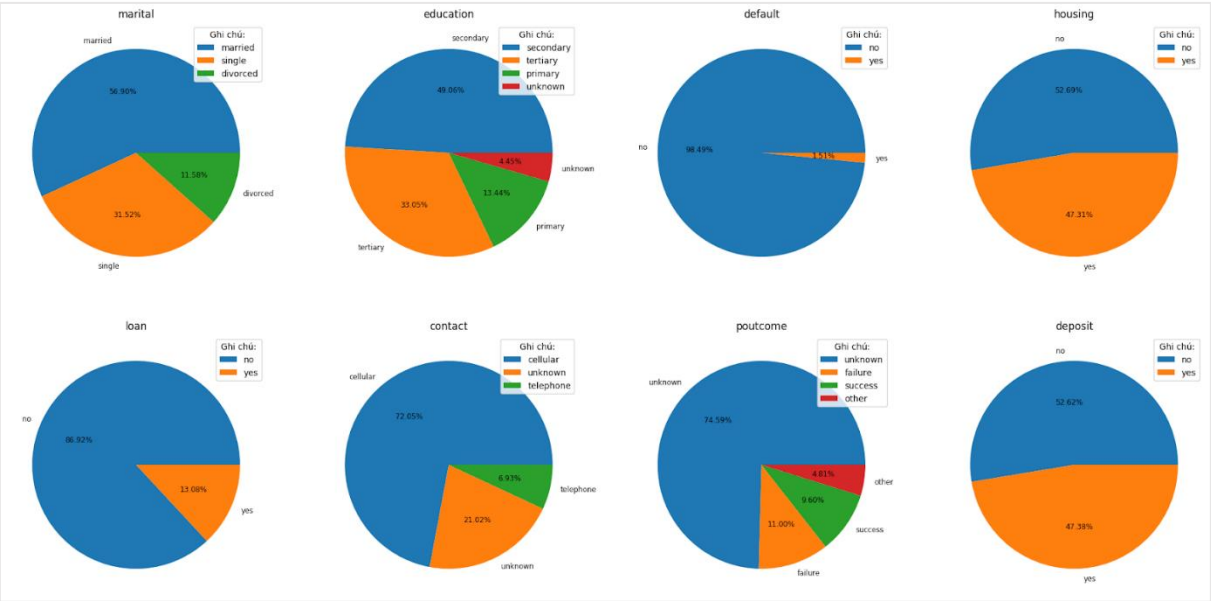
- Cột "job" cho thấy có 12 công việc khác nhau trong tập dữ liệu, với "management" xuất hiện nhiều nhất (2566 lần).
- Cột "marital" cho thấy có 3 tình trạng hôn nhân khác nhau trong tập dữ liệu, với "married" xuất hiện nhiều nhất (6351 lần).
- Cột "education" cho thấy có 3 trình độ học vấn khác nhau trong tập dữ liệu, với "secondary" xuất hiện nhiều nhất (5476 lần).
- Cột "default" thể hiện người đó có thẻ tín dụng hay không trong tập dữ liệu, với "no" xuất hiện nhiều nhất (10994 lần).
- Cột "housing" thể hiện người đó có vay mua nhà hay không trong tập dữ liệu, với "no" xuất hiện nhiều nhất (5881 lần).
- Cột "loan" thể hiện người đó có khoản vay cá nhân hay không trong tập dữ liệu, với "no" xuất hiện nhiều nhất (9702 lần).
- Cột "contact" cho thấy có 2 hình thức liên lạc trong tập dữ liệu, "cellular" là hình thức xuất hiện nhiều nhất (8042 lần).
- Cột "month" thể hiện các tháng đã liên lạc cuối cùng trong năm trong tập dữ liệu, với "may" xuất hiện nhiều nhất (2824 lần).
- Cột "poutcome" thể hiện kết quả của chiến dịch tiếp thị trước đó trong tập dữ liệu, với "unknown" xuất hiện nhiều nhất (8326 lần).
- Cột "deposit" thể hiện khách hàng đã đăng ký gửi tiền có kỳ hạn hay chưa trong tập dữ liệu, với "no" xuất hiện nhiều nhất (5873 lần).

Phân tích đơn biến

Ta quan sát các thuộc tính phân loại có ít danh mục sau



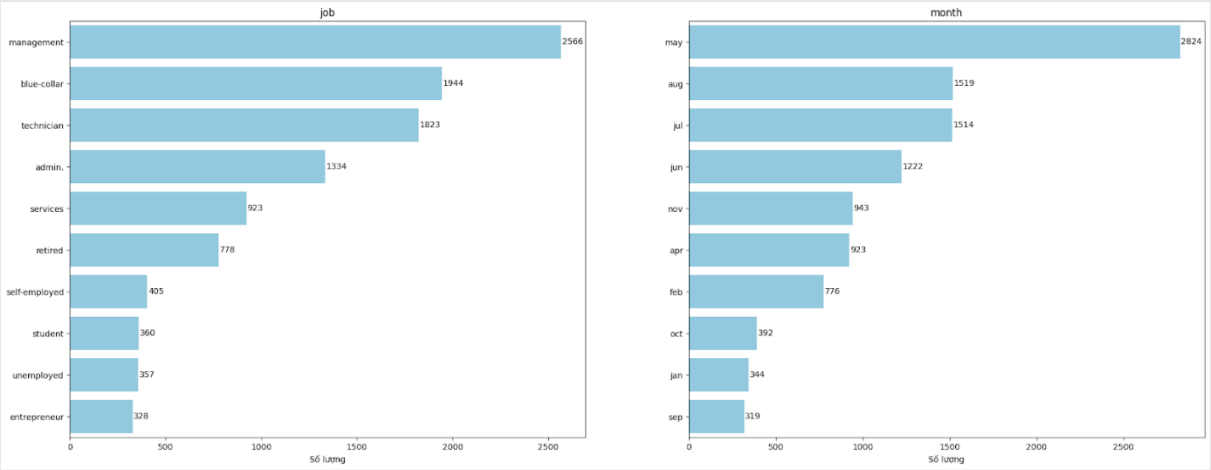
Hình 9: Biểu đồ cột thể hiện số lượng của mỗi danh mục trong 1 biến



Hình 10: Biểu đồ tròn thể hiện phần trăm của mỗi danh mục trong 1 biến

- Khi nhìn vào các biểu đồ, ta có thể thấy rằng các khách hàng tiếp cận được chủ yếu là những người đã kết hôn (chiếm 56.9% trong tổng số) và gần một nửa đang có nhu cầu vay tiền mua nhà (chiếm 47.31% trong tổng số).
- Chủ yếu các cuộc tư vấn được liên hệ trực tiếp qua di động (chiếm 72.05%)
- Chiếm phần lớn khách hàng được tư vấn không có thẻ tín dụng hay khoản vay cá nhân nào (chiếm 98.49% ở cột Default và chiếm 86.92% ở cột “loan”)

Tiếp theo, Ta quan sát các thuộc tính phân loại có nhiều danh mục



Hình 11: Biểu đồ cột thể hiện số lượng của mỗi danh mục trong "job" và "month"

Dựa vào biểu đồ cột trên, ta có thể thấy những người đã tư vấn chủ yếu là những người công nhân và quản lý. Bên cạnh đó, khoảng thời gian liên lạc nhiều nhất với khách hàng đã tư vấn chủ yếu trong tháng 5.

2. TIỀN XỬ LÝ DỮ LIỆU

2.1. Quan sát dữ liệu bị thiếu

Trong bộ dữ liệu, tất cả giá trị bị thiếu trong một số thuộc tính phân loại đã được mã hóa thành "unknown".

Ta quan sát những giá trị bị thiếu được gán bằng nhãn "unknown":

```
def unknown_ratio(column):
    return column.apply(lambda x: x == "unknown").sum()
df.apply(unknown_ratio)
```

Kết quả thu được:

```
age            0
job           70
marital        0
education    497
default        0
balance        0
housing        0
loan           0
contact     2346
day            0
month          0
duration       0
campaign       0
pdays          0
previous       0
poutcome    8326
deposit        0
```

```
dtype: int64
```

Ta sẽ tiến hành xóa những cột có "unknown" trên 80% dữ liệu và xóa những dòng chứa 3 giá trị bị thiếu, còn những cột còn thiếu còn lại coi như là nhãn "unknown".

2.2. Quan sát giá trị nhiễu

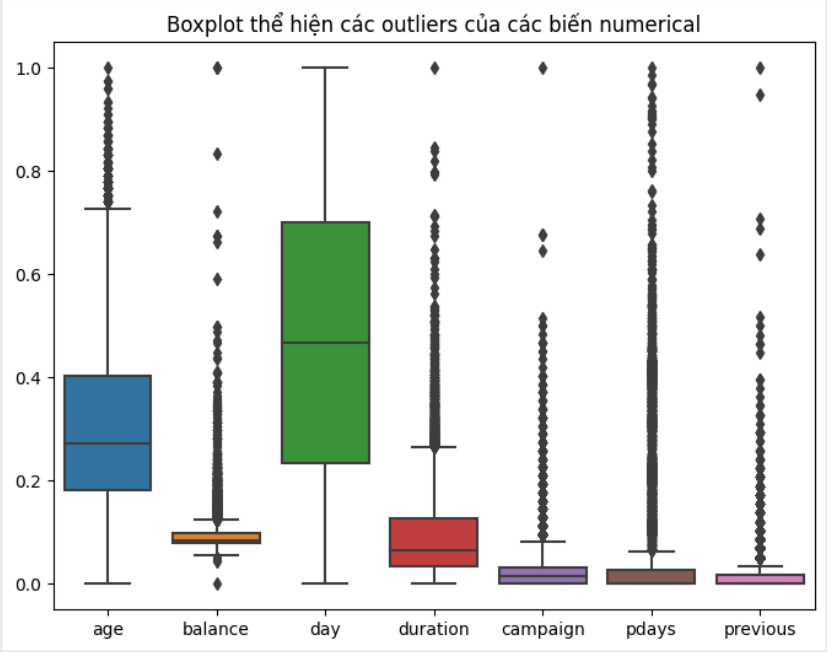
Để quan sát các giá trị nhiễu, trước hết ta sẽ chuẩn hóa các dữ liệu numerical về Minmaxscaler để có thể dễ dàng xem các điểm dữ liệu.

```
# Kiểm tra outliers
numerical = df.select_dtypes(include=['number'])
numerical_columns = numerical.columns
scaler = MinMaxScaler()
scaler.fit(numerical)
numerical = scaler.transform(numerical)
numerical = pd.DataFrame(numerical, columns = numerical_columns)
numerical.head()
```

Ta truyền vào các cột numerical để quan sát giá trị nhiễu

```
plt.subplots(figsize = (8,6))
sns.boxplot(numerical)
plt.title("Boxplot thể hiện các outliers của các biến numerical")
plt.show()
```

Kết quả thu được:



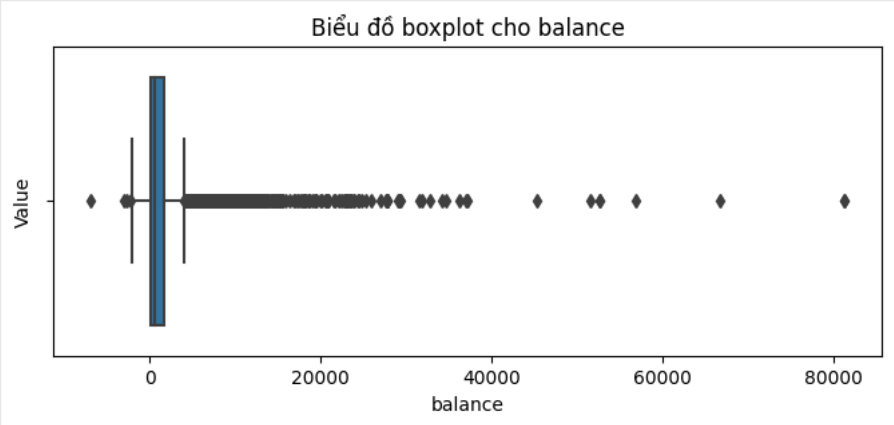
Hình 12: Boxplot thể hiện các outliers của các biến numerical

Quan sát các thuộc tính nhiễu trên, ta có thể thấy cột day và cột age đều thỏa mãn nằm trong phạm vi hợp lý (như cột age thể hiện độ tuổi từ 18 - 95 tuổi và cột day thể

hiện từ ngày 1 đến ngày 31). Ta sẽ tiến hành xử lý các outliers cho các cột còn lại gồm: balance, duration, campaign, pdays, previous.

2.2.1. Balance

```
box = 'balance'
plt.subplots(figsize = (8,3))
plt.title(f'Biểu đồ boxplot cho {box}')
plt.xlabel(box)
sns.boxplot(x=box, data=df)
plt.ylabel('Value')
plt.show()
```



Hình 13: Biểu đồ boxplot cho balance

Tính toán số lượng outliers:

```
Q1 = df[box].quantile(0.25)
Q3 = df[box].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outlier = df[(df[box] < lower_bound) | (df[box] > upper_bound)]
print(f'Giá trị ngoại lai chiếm
{round(outlier.shape[0]/df.shape[0]*100,2)}% bộ dữ liệu')
```

Kết quả thu được

```
Giá trị ngoại lai chiếm 9.46% bộ dữ liệu
```

Vì có nhiều giá trị ngoại lai nên không thể xử lý bằng cách loại bỏ outliers. Xóa bỏ đi giá trị quá lớn và giá trị quá nhỏ. Cụ thể là xóa bỏ đi những giá trị lớn hơn 40000, và nhỏ hơn lower_bound vì những giá trị này chiếm chưa tới 0.5% bộ dữ liệu

```
# Đếm số lần xuất hiện của giá trị lớn hơn 40000 trong cột "balance"
balance_greater_than_40000 = (df['balance'] > 40000)
print(f"Số lần xuất hiện của giá trị lớn hơn 40000 trong cột
```

```
'balance': {balance_greater_than_40000.sum()}")
df[balance_greater_than_40000]['deposit'].value_counts()
```

```
Số lần xuất hiện của giá trị lớn hơn 40000 trong cột 'balance': 8
yes    5
no     3
Name: deposit, dtype: int64
```

```
# Đếm số lần xuất hiện của giá trị nhỏ hơn lower_bound trong cột
"balance"
balance_less_than_lower_bound = (df['balance'] < lower_bound)
print(f"Số lần xuất hiện của giá trị nhỏ hơn đáy giới hạn trong cột
'balance': {balance_less_than_lower_bound.sum()}")
df[balance_less_than_lower_bound]['deposit'].value_counts()
```

```
Số lần xuất hiện của giá trị nhỏ hơn đáy giới hạn trong cột 'balance':
4
no     3
yes    1
Name: deposit, dtype: int64
```

```
# Bỏ đi những giá trị nhỏ nhỏ hơn lower_bound và giá trị lớn hơn
40000.
df.drop(df[balance_greater_than_40000 |
balance_less_than_lower_bound].index, inplace=True)
```

Sau khi xem 3 kết quả trên thì ta có thể thấy có 12 giá trị mà ta cần xử lý, trong đó có 8 giá trị lớn hơn 40.000 và 4 giá trị nhỏ hơn lower_bound chiếm chưa tới 0.5% bộ dữ liệu.

Tiếp theo, ta tính phần trăm cho mỗi giá trị của cột “deposit” trong outliers còn lại so với cả bộ dữ liệu

```
# Đếm số lượng mỗi giá trị trong cột 'deposit' cho cả df và outlier
df_counts = df['deposit'].value_counts()
outlier_counts = outlier['deposit'].value_counts()
# Tính phần trăm cho mỗi giá trị của cột 'deposit' trong outlier so
với bộ dữ liệu
percentage_outlier = round((outlier_counts / df_counts) * 100,2)
percentage_outlier
```

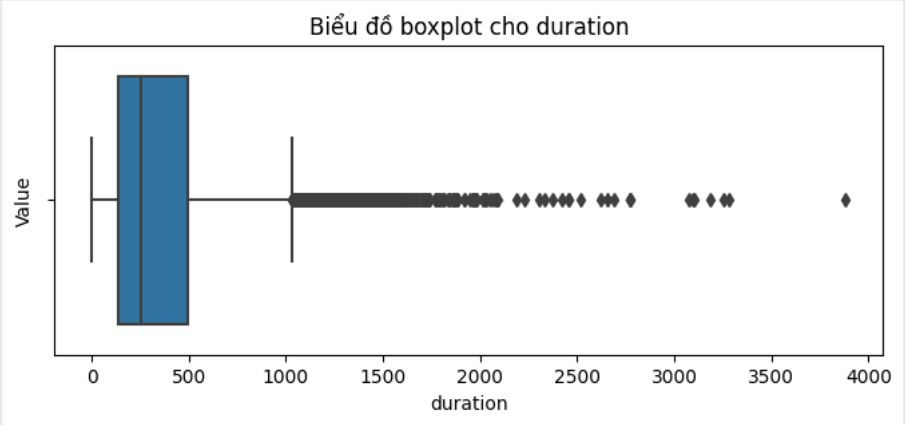
```
no      7.69
yes    11.42
Name: deposit, dtype: float64
```

Ta thấy “yes” ở cột deposit trong outliers chiếm 11,42% trong toàn bộ dữ liệu

nên ta có thể thấy rằng dữ liệu trong outliers có thể khai thác được, trong bài toán này ta tập trung vào những dữ liệu có deposit là "yes".

2.2.2. Duration

```
box = 'duration'
plt.subplots(figsize = (8,3))
plt.title(f'Biểu đồ boxplot cho {box}')
plt.xlabel(box)
sns.boxplot(x=box, data=df)
plt.ylabel('Value')
plt.show()
```



Hình 14: Biểu đồ boxplot cho duration

Tính toán số lượng outlier:

```
Q1 = df[box].quantile(0.25)
Q3 = df[box].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outlier = df[(df[box] < lower_bound) | (df[box] > upper_bound)]
print(f'Giá trị ngoại lai chiếm
{round(outlier.shape[0]/df.shape[0]*100,2)}% bộ dữ liệu')
```

Kết quả thu được:

```
Giá trị ngoại lai chiếm 5.7% bộ dữ liệu
```

Đếm số lượng cuộc gọi dài hơn 1 tiếng:

```
# Đếm số cuộc gọi trên 60 phút
duration_greater_than_60m = (df['duration'] > 60*60)
print(f"Số cuộc gọi trên 60 phút: {duration_greater_than_60m.sum()}")
df[duration_greater_than_60m]['deposit'].value_counts()
```

Kết quả thu được:

```
Số cuộc gọi trên 60 phút: 1
yes    1
Name: deposit, dtype: int64
```

Tiến hành bỏ những cuộc gọi trên 1 tiếng, sau đó tính toán phần trăm cho mỗi giá trị của cột "deposit" trong outliers.

```
# Bỏ đi một cuộc gọi trên 1 tiếng
df.drop(df[duration_greater_than_60m].index, inplace=True)
```

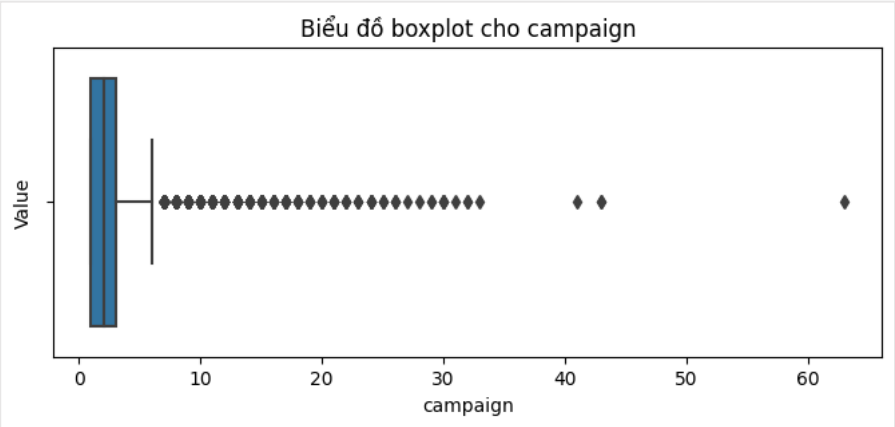
```
# Tính phần trăm cho mỗi giá trị của cột 'deposit' trong outlier so với bộ dữ liệu
percentage_outlier = round((outlier_counts / df_counts) * 100,2)
percentage_outlier
```

Kết quả thu được:

```
no      1.09
yes    10.81
Name: deposit, dtype: float64
```

Tỷ lệ những cuộc gọi càng dài thì chiếm phần tỷ lệ trăm đồng ý ký gửi kỳ hạn cao hơn, nên những giá trị outliers này có giá trị phân tích nên giữ lại.

2.2.3. Campaign



Hình 15: Biểu đồ boxplot cho campaign

Tính toán số lượng outlier:

```
Q1 = df[box].quantile(0.25)
Q3 = df[box].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outlier = df[(df[box] < lower_bound) | (df[box] > upper_bound)]
print(f'Giá trị ngoại lai chiếm
{round(outlier.shape[0]/df.shape[0]*100,2)}% bộ dữ liệu')
```

Kết quả thu được:

```
Giá trị ngoại lai chiếm 5.38% bộ dữ liệu
```

Đếm số lượng khách hàng được liên hệ trên 25 lần:

```
# Đếm số khách hàng được liên hệ trên 25 lần
campaign_greater_than_25 = (df['campaign'] >= 25)
print(f"Số khách hàng được liên hệ trên 30 lần:
{campaign_greater_than_25.sum()}")
df[campaign_greater_than_25]['deposit'].value_counts()
```

Kết quả thu được:

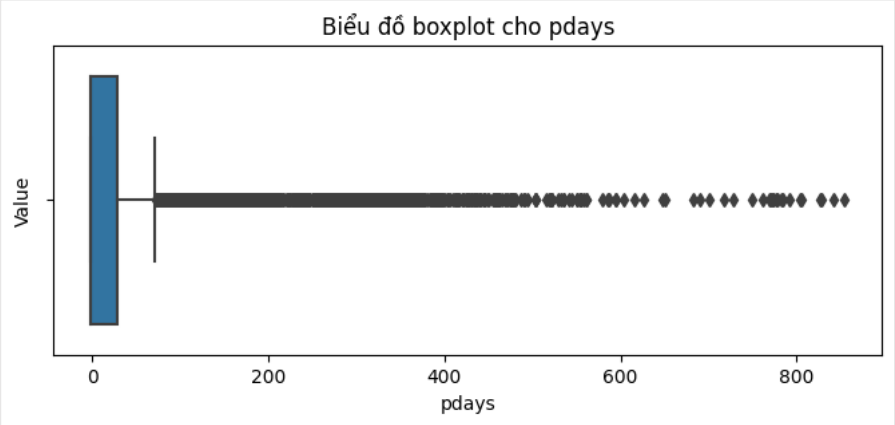
```
Số khách hàng được liên hệ trên 30 lần: 22
no     20
yes     2
Name: deposit, dtype: int64
```

Ta tiến hành bỏ những hành khác đã được liên hệ trên 25 lần

```
# Bỏ đi những khách hàng được liên hệ trên 30 lần
df.drop(df[campaign_greater_than_25].index, inplace=True)
```

Còn lại các outliers là những khách hàng được liên hệ dưới 25 lần ta sẽ giữ lại vì các outliers này vẫn có thể khai thác được.

2.2.4. Pdays



Hình 16: Biểu đồ boxplot cho pdays

Tính toán số lượng outlier:

```
Q1 = df[box].quantile(0.25)
Q3 = df[box].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outlier = df[(df[box] < lower_bound) | (df[box] > upper_bound)]
print(f'Giá trị ngoại lai chiếm
```

```
{round(outlier.shape[0]/df.shape[0]*100,2)}% bộ dữ liệu')
```

Kết quả thu được:

```
Giá trị ngoại lai chiếm 24.29% bộ dữ liệu
```

Đếm số lượng khách hàng chưa liên hệ trên 2 năm

```
# Đếm số khách hàng chưa liên hệ trên 2 năm
pdays_greater_than_2years = (df['pdays'] > 2*365)
print(f"Số khách hàng chưa liên hệ trên 2 năm:
{pdays_greater_than_2years.sum()}")
df[pdays_greater_than_2years]['deposit'].value_counts()
```

Kết quả thu được:

```
Số khách hàng chưa liên hệ trên 2 năm: 17
yes    12
no      5
Name: deposit, dtype: int64
```

Ta tính toán phần trăm cho mỗi giá trị của cột “deposit” trong outliers so với toàn bộ dữ liệu:

```
# Đếm số lượng mỗi giá trị trong cột 'deposit' cho cả df và outlier
df_counts = df['deposit'].value_counts()
outlier_counts = outlier['deposit'].value_counts()
# Tính phần trăm cho mỗi giá trị của cột 'deposit' trong outlier so
với bộ dữ liệu
percentage_outlier = round((outlier_counts / df_counts) * 100,2)
percentage_outlier
```

Kết quả thu được:

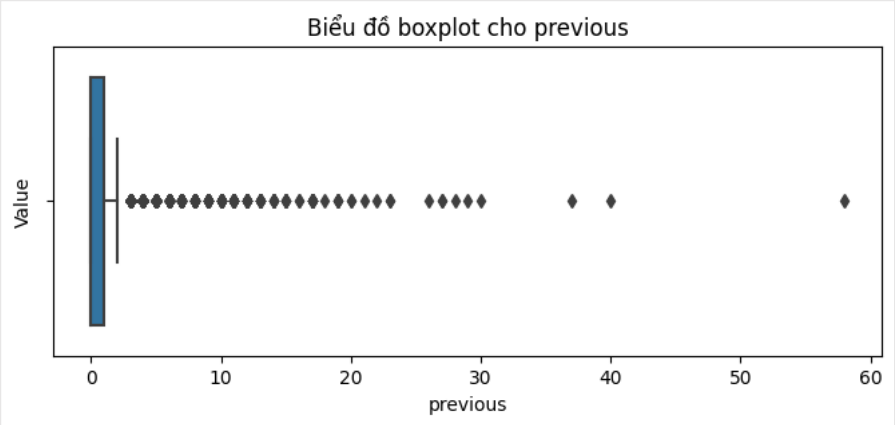
```
no     15.20
yes    34.34
Name: deposit, dtype: float64
```

Sau đó tiến hành bỏ đi những khách hàng chưa liên hệ trên 2 năm vì thông tin về khách hàng đã cũ:

```
df.drop(df[pdays_greater_than_2years].index, inplace=True)
```

Ta giữ lại những outliers là những khách chưa liên hệ dưới 2 năm, thông tin của họ vẫn có thể còn khả dụng, khi liên hệ tư vấn cho những người này thì tỷ lệ họ đồng ý ký gửi kỳ hạn vẫn chấp nhận được.

2.2.5. Previous



Hình 17: Biểu đồ boxplot cho previous

Tính toán số lượng outlier:

```
Q1 = df[box].quantile(0.25)
Q3 = df[box].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outlier = df[(df[box] < lower_bound) | (df[box] > upper_bound)]
print(f'Giá trị ngoại lai chiếm
{round(outlier.shape[0]/df.shape[0]*100,2)}% bộ dữ liệu')
```

Kết quả thu được:

```
Giá trị ngoại lai chiếm 11.27% bộ dữ liệu
```

Tính toán phần trăm cho mỗi giá trị của “deposit” trong outliers so với bộ dữ liệu.

```
# Đếm số lượng mỗi giá trị trong cột 'deposit' cho cả df và outlier
df_counts = df['deposit'].value_counts()
outlier_counts = outlier['deposit'].value_counts()
# Tính phần trăm cho mỗi giá trị của cột 'deposit' trong outlier so
với bộ dữ liệu
percentage_outlier = round((outlier_counts / df_counts) * 100,2)
percentage_outlier
```

Kết quả thu được:

```
no      6.66
yes    16.37
Name: deposit, dtype: float64
```

Đếm số khách hàng được liên hệ trước chiến dịch này trên 30 lần.

```
previous_greater_than_30 = (df['previous'] > 30)
print(f"Số khách hàng được liên hệ trước chiến dịch này trên 30 lần:
{previous_greater_than_30.sum()}")
```

```
df[previous_greater_than_30]['deposit'].value_counts()
```

Kết quả thu được:

```
Số khách hàng được liên hệ trước chiến dịch này trên 30 lần: 3
no     2
yes    1
Name: deposit, dtype: int64
```

Ta tiến hành bỏ đi 1 giá trị cao bất thường và giữ lại các outliers khác

```
df.drop(df[previous_greater_than_30].index, inplace=True)
```

Ta chỉ bỏ đi 1 giá trị outliers cao bất thường, và giữ các outliers còn lại vì các outlier này cũng có thể khai thác được vì tỷ lệ "yes" của deposit khá cao, ta có thể khai thác được.

2.3. Thu gọn dữ liệu

Ta tiến hành bỏ cột day, month thì không có thông tin về năm và thông tin về ngày tháng không có giá trị.

```
df.drop(df[['day','month']],axis=1, inplace=True)
```

2.4. Chuyển dạng dữ liệu

2.4.1. Age

- 18-27: Độ tuổi thanh niên, bắt đầu phát triển sự nghiệp, tài chính chưa mạnh.
- 28-44: Độ tuổi lập gia đình và bắt đầu sinh con cái, con cái phụ thuộc. Nền tảng tài chính chưa mạnh, chưa có nhiều tích lũy.
- 45-64: Con cái đã lớn, bắt đầu bước sang tuổi trưởng thành, nền tảng tài chính ổn đinh, có tích lũy.
- 65-84: Độ tuổi bắt đầu về hưu. Con cái đã trưởng thành và lập gia đình, không có người phụ thuộc.
- 85+: Giai đoạn của người cao niên, có thể mắc các bệnh và không có khả năng tạo ra thu nhập.

Rời rạc hóa cho cột "age":

```
bins = [18, 28, 45, 65, 85, 100]
labels = ['18_27', '28_44', '45_64', '65_84', '85_100']

df['age_bins'] = pd.cut(df['age'], bins=bins, labels=labels,
right=False)
df['age_bins'].value_counts()
```

Kết quả thu được:

```
28_44      6315
45_64      3436
18_27       927
65_84       399
85_100       24
Name: age_bins, dtype: int64
```

2.4.2. Balance

Đặt nhãn cho từng giá trị ở cột “balance”

```
Q1 = df['balance'].quantile(0.25)
Q3 = df['balance'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

bins = [-float('inf'), 0, round(Q3), round(upper_bound), 10000, 20000,
float('inf')]

# Đặt nhãn cho từng khoảng giá trị
labels = ['debt', 'low', 'moderate', 'high', 'very_high',
'extremely_high']

# Sử dụng cut để thực hiện custom binning
df['balance_bins'] = pd.cut(df['balance'], bins=bins, labels=labels)

# Hiển thị DataFrame sau khi thực hiện rời rạc hóa
print(df[['balance', 'balance_bins']].head())
```

Kết quả thu được:

```
   balance balance_bins
0     2343     moderate
1       45          low
2     1270          low
3     2476     moderate
4      184          low
```

```
df['balance_bins'].value_counts()
```

```
low               6875
moderate          1734
debt              1451
high               822
very_high          170
extremely_high      49
Name: balance_bins, dtype: int64
```

2.4.3. Duration

Đặt nhãn cho từng giá trị ở cột "duration":

```
bins = [0, 901, 1801, 2701, 3601]
labels = ['under_15m', '15_30m', '30_45m', '45_60m']
df['duration_bins'] = pd.cut(df['duration'], bins=bins, labels=labels,
right=False)
# Hiển thị DataFrame sau khi thực hiện rời rạc hóa
print(df[['duration', 'duration_bins']].head())
```

Kết quả thu được:

```
   duration duration_bins
0      1042        15_30m
1      1467        15_30m
2      1389        15_30m
3       579     under_15m
4       673     under_15m
```

```
df['duration_bins'].value_counts()
```

```
under_15m    10169
15_30m         873
30_45m          50
45_60m           9
Name: duration_bins, dtype: int64
```

2.4.4. Campaign

Rời rạc hóa cho cột "campaign" theo nhóm 5

```
bins = list(range(df['campaign'].min(), (df['campaign'].max()+4), 5))
labels = [f"{i}_{i+4}" for i in range(df['campaign'].min(),
df['campaign'].max(), 5)]
df['campaign_bins'] = pd.cut(df['campaign'], bins=bins, labels=labels,
right=False)

# Hiển thị DataFrame sau khi rời rạc hóa
print(df[['campaign', 'campaign_bins']].head())
```

Kết quả thu được:

```
   campaign campaign_bins
0         1          1_5
1         1          1_5
2         1          1_5
3         1          1_5
4         2          1_5
```

```
df['campaign_bins'].value_counts()
```

```
1_5       10260
6_10        653
11_15       127
16_20        41
21_25        20
Name: campaign_bins, dtype: int64
```

2.4.5. Pdays

Đối với cột "pdays", ta đổi -1 thành no và rời rạc hóa từ 1 ngày đến 3 tháng, từ 3 tháng đến 6 tháng, từ 6 tháng tới 12 tháng và 1 - 2 năm.

```
bins = [-1, 1, 91, 181, 366, 365*2+1]
labels = ['no', 'under_3_months', '3_6months', '6months_1year',
'1_2years']
df['pdays_bins'] = pd.cut(df['pdays'], bins=bins, labels=labels,
right=False)
# Hiển thị DataFrame sau khi rời rạc hóa
print(df[['pdays', 'pdays_bins']].head())
```

Kết quả thu được:

```
   pdays pdays_bins
0     -1         no
1     -1         no
2     -1         no
3     -1         no
4     -1         no
```

```
df['pdays_bins'].value_counts()
```

```
no                8286
6months_1year     1303
3_6months          948
under_3_months     329
1_2years           235
Name: pdays_bins, dtype: int64
```

2.4.6. Previous

Ta Binning cột previous nhưng vẫn giữ nguyên giá trị 0 vì bản chất là 0 liên lạc nên ta sẽ rời rạc hóa từ 1 đến giá trị lớn nhất)

```
bins = [0, 1, 11, 21, 31]
labels = ['no', '1_10', '11_20', '21_30']
df['previous_bins'] = pd.cut(df['previous'], bins=bins, labels=labels,
right=False)

# Hiển thị DataFrame sau khi rời rạc hóa
```

```
print(df[['previous', 'previous_bins']].head())
```

Kết quả thu được:

```
   previous previous_bins
0         0            no
1         0            no
2         0            no
3         0            no
4         0            no
```

```
df['previous_bins'].value_counts()
```

```
no        8286
1_10      2725
11_20       80
21_30       10
Name: previous_bins, dtype: int64
```

2.5. Dữ liệu sau khi tiền xử lý

Ta tạo 1 biến khác chứa các biến features sau khi được chuyển dạng dữ liệu:

```
columns_cba = ['age_bins', 'job', 'marital', 'education', 'default',
'balance_bins', 'housing', 'loan', 'contact', 'duration_bins',
'campaign_bins', 'pdays_bins', 'previous_bins', 'deposit']
df_cba = df[columns_cba].reset_index(drop=True)
```

Ta đổi tên các cột có chứa bins

```
columns_with_bins = [col for col in df_cba.columns if '_bins' in col]
df_cba.rename(columns=lambda x: x.replace('_bins', ''), inplace=True)
df_cba
```

Sau khi tiến hành tiền xử lý, ta có dữ liệu dưới đây:

	age	job	marital	education	default	balance	housing	loan	contact	duration	campaign	pdays	previous	deposit
0	45_64	admin.	married	secondary	no	moderate	yes	no	unknown	15_30m	1_5	no	no	yes
1	45_64	admin.	married	secondary	no	low	no	no	unknown	15_30m	1_5	no	no	yes
2	28_44	technician	married	secondary	no	low	yes	no	unknown	15_30m	1_5	no	no	yes
3	45_64	services	married	secondary	no	moderate	yes	no	unknown	under_15m	1_5	no	no	yes
4	45_64	admin.	married	tertiary	no	low	no	no	unknown	under_15m	1_5	no	no	yes
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11096	28_44	blue-collar	single	primary	no	low	yes	no	cellular	under_15m	1_5	no	no	no
11097	28_44	services	married	secondary	no	low	no	no	unknown	under_15m	1_5	no	no	no
11098	28_44	technician	single	secondary	no	low	no	no	cellular	under_15m	1_5	no	no	no
11099	28_44	technician	married	secondary	no	debt	no	yes	cellular	under_15m	1_5	3_6months	1_10	no
11100	28_44	technician	married	secondary	no	debt	no	no	cellular	under_15m	1_5	no	no	no

Hình 18: Dữ liệu sau khi tiền xử lý

3. XÂY DỰNG MÔ HÌNH

3.1. Thuật toán CBA với thư viện pyARC

3.1.1. Xây dựng mô hình

Đầu tiên chia tập dữ liệu thành tập train và tập test, với tỷ lệ tập train là 80%, tập test là 20%.

```
# Tạo tập huấn luyện và tập dữ liệu còn lại
data_train, data_test = train_test_split(df_cba, test_size=0.2,
random_state=42)

data_train = data_train.reset_index(drop=True)
data_test = data_test.reset_index(drop=True)
```

Trước khi đưa vào mô hình cần chuyển đổi sang dạng transactionDB.

```
# Chuyển đổi dữ liệu
txns_train = TransactionDB.from_DataFrame(data_train)
txns_test = TransactionDB.from_DataFrame(data_test)
```

Thực hiện Grid Search với các tham số algorithms = ["m1", "m2"], confidences = [0.4, 0.5, 0.6], supports = [0.05, 0.1, 0.3] để tạo bảng đánh giá, và xem xét chọn kết quả tốt và phù hợp với mục tiêu của bài toán.

```
# Các giá trị tham số bạn muốn thử nghiệm
algorithms = ["m1", "m2"]
confidences = [0.4, 0.5, 0.6]
supports = [0.05, 0.1, 0.3]

# Tạo DataFrame để lưu kết quả
results_df = pd.DataFrame(columns=["Algorithm", "Confidence",
"Support", "Accuracy", "Precision", "Recall", "F1"])

# Duyệt qua tất cả các tổ hợp của các tham số
for algorithm, confidence, support in product(algorithms, confidences,
supports):
    # Khởi tạo và huấn luyện mô hình với các tham số
    cba_model = CBA(algorithm=algorithm, confidence=confidence,
support=support).fit(txns_train)

    # Dự đoán trên tập kiểm tra
    y_pred = cba_model.predict(txns_test)

    # Tính toán các độ đo
    accuracy = cba_model.rule_model_accuracy(txns_test)
    precision = precision_score(data_test["deposit"],
y_pred,pos_label='yes')
    recall = recall_score(data_test["deposit"], y_pred,
pos_label='yes')
    f1 = f1_score(data_test["deposit"], y_pred, pos_label='yes')

    # Thêm kết quả vào DataFrame
```

```
    results_df = results_df.append({
        "Algorithm": algorithm,
        "Confidence": confidence,
        "Support": support,
        "Accuracy": accuracy,
        "Precision": precision,
        "Recall": recall,
        "F1": f1
    }, ignore_index=True)

# In bảng kết quả
print(results_df)
```

Ta thu được kết quả giảm dần theo cột F1 như sau:

	Algorithm	Confidence	Support	Accuracy	Precision	Recall	F1
0	m1	0.4	0.05	0.705988	0.710071	0.655399	0.681641
6	m1	0.6	0.05	0.705988	0.710071	0.655399	0.681641
3	m1	0.5	0.05	0.705988	0.710071	0.655399	0.681641
1	m1	0.4	0.10	0.676722	0.657324	0.682629	0.669737
4	m1	0.5	0.10	0.676722	0.657324	0.682629	0.669737
7	m1	0.6	0.10	0.676722	0.657324	0.682629	0.669737
9	m2	0.4	0.05	0.703287	0.753750	0.566197	0.646649
12	m2	0.5	0.05	0.702386	0.753769	0.563380	0.644815
15	m2	0.6	0.05	0.702386	0.753769	0.563380	0.644815
5	m1	0.5	0.30	0.661864	0.712737	0.493897	0.583472
8	m1	0.6	0.30	0.661864	0.712737	0.493897	0.583472
2	m1	0.4	0.30	0.661864	0.712737	0.493897	0.583472
11	m2	0.4	0.30	0.661864	0.712737	0.493897	0.583472
14	m2	0.5	0.30	0.661864	0.712737	0.493897	0.583472
17	m2	0.6	0.30	0.661864	0.712737	0.493897	0.583472
10	m2	0.4	0.10	0.656911	0.747145	0.430047	0.545888
13	m2	0.5	0.10	0.656911	0.747145	0.430047	0.545888
16	m2	0.6	0.10	0.656911	0.747145	0.430047	0.545888

Hình 19: Bảng đánh giá tham số

Dựa vào bảng kết quả trên, ta thấy với các tham số algorithm = 'm1', confidence = 0.4, support = 0.05, các chỉ số đánh giá đều cao hơn các tham số còn lại. Vậy ta sử dụng tham số này đưa vào mô hình.

```
algorithm = 'm1'
confidence = 0.4
support = 0.05

# Khởi tạo và huấn luyện lại mô hình với bộ tham số tốt nhất
cba = CBA(algorithm=algorithm, confidence=confidence, support=support)
cba_model = cba.fit(txns_train)
y_pred = cba_model.predict(txns_test)
```

Xem lại các luật mà mô hình đã học được:

```
# Xem các luật kết hợp đã được học
```

```
# Truy cập danh sách các luật đã học được
rules = cba_model.clf.rules

# In ra nội dung của từng luật
for rule in rules:
    print(rule)
```

```
CAR {pdays=no,duration=15_30m} => {deposit=yes} sup: 0.06 conf: 0.90
len: 3, id: 77939
CAR {duration=15_30m} => {deposit=yes} sup: 0.07 conf: 0.90 len: 2,
id: 77951
CAR
{pdays=no,contact=unknown,duration=under_15m,marital=married,housing=y
es} => {deposit=no} sup: 0.07 conf: 0.89 len: 6, id: 79066
CAR {contact=unknown,marital=married,duration=under_15m,housing=yes}
=> {deposit=no} sup: 0.07 conf: 0.88 len: 5, id: 79098
CAR
{pdays=no,contact=unknown,education=secondary,duration=under_15m,marit
al=married} => {deposit=no} sup: 0.06 conf: 0.88 len: 6, id: 79274
…
CAR {pdays=no,age=28_44,marital=married,duration=under_15m} =>
{deposit=no} sup: 0.15 conf: 0.73 len: 5, id: 88409
CAR
{pdays=no,education=secondary,duration=under_15m,balance=low,marital=m
arried} => {deposit=no} sup: 0.09 conf: 0.73 len: 6, id: 87178
```

Mô hình đã học được 76 luật kết hợp với ngưỡng hỗ trợ tối thiểu là 5% và ngưỡng tin cậy tối thiểu và 40%.

3.1.2. Đánh giá mô hình

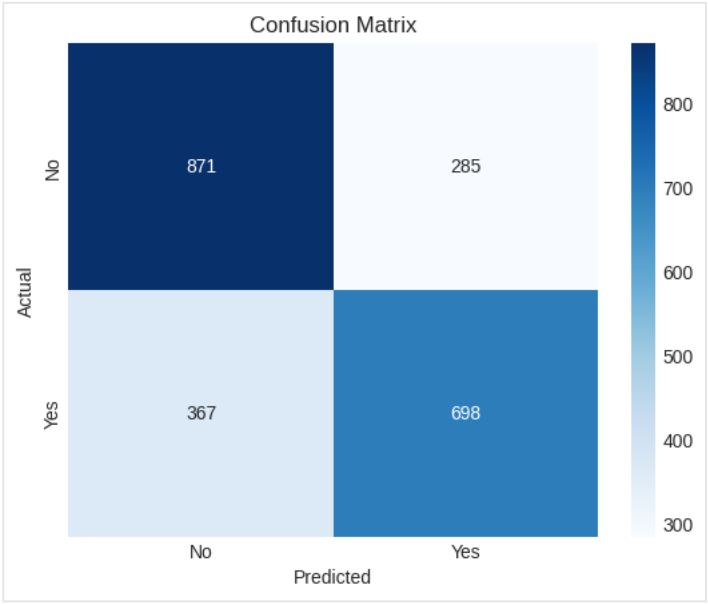
Lập hàm đánh giá bao gồm các độ đo precision, recall và f1 srcore.

```
def evaluate_classification(y_true, y_pred):

    # Tính các thước đo chính
    precision = precision_score(y_true, y_pred,pos_label='yes')
    recall = recall_score(y_true, y_pred, pos_label='yes')
    f1 = f1_score(y_true, y_pred, pos_label='yes')


    ## Hiển thị giá trị cá chỉ số
    print(f'Precision = {(precision * 100):.1f}%')
    print(f'Recall    = {(recall * 100):.1f}%')
    print(f'F1 Score  = {(f1 * 100):.1f}%')
```

Sử dụng ma trận nhầm lẫn để đánh giá đánh giá hiệu quả của một mô hình phân loại nhị phân.

```
# Tính confusion matrix
conf_matrix = confusion_matrix(data_test['deposit'], y_pred)

# Vẽ ma trận nhầm lẫn bằng seaborn
```

```
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=['No', 'Yes'], yticklabels=['No', 'Yes'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



Hình 20: Ma trận nhầm lẫn đánh giá đánh giá hiệu quả của mô hình phân loại nhị phân

Kết quả ma trận:

- Có 698 khách hàng được dự đoán đúng là đăng ký gửi tiền (deposit=yes).
- Có 871 khách hàng được dự đoán đúng là không đăng ký gửi tiền (deposit=no).
- Có 285 khách hàng được dự đoán là đăng ký gửi tiền, nhưng thực tế lại không đăng ký.
- Có 367 khách hàng được dự đoán là không đăng ký gửi tiền, nhưng thực tế lại đăng ký.

```
# Các chỉ số đánh giá
accuracy = cba_model.rule_model_accuracy(txns_test)
print(f'Accuracy={(accuracy*100):.1f}%')
evaluate_classification(data_test['deposit'], y_pred)
```

```
Accuracy = 70.6%
Precision = 71.0%
Recall    = 65.5%
F1 Score  = 68.2%
```

Các chỉ số đánh giá:

- Accuracy: khoảng 70.6% tỷ lệ dự đoán đúng trên toàn bộ dữ liệu. Mặc dù kết quả

Có vẻ cao, nhưng không đủ để đánh giá hiệu suất của mô hình.

- Precision: 71.0% Mức này cũng khá tốt, với khoảng 71.0% khách hàng được dự đoán có đăng ký gửi là chính xác.
- Recall: 65.5% Mô hình dự đoán đúng 65,5% các trường hợp thực sự sẽ đăng ký tiền gửi có kỳ hạn. Đây là một kết quả khá tốt.
- F1 Score: 68.2%. Đây cũng là một kết quả khá tốt, precision và recall không mất cân bằng.

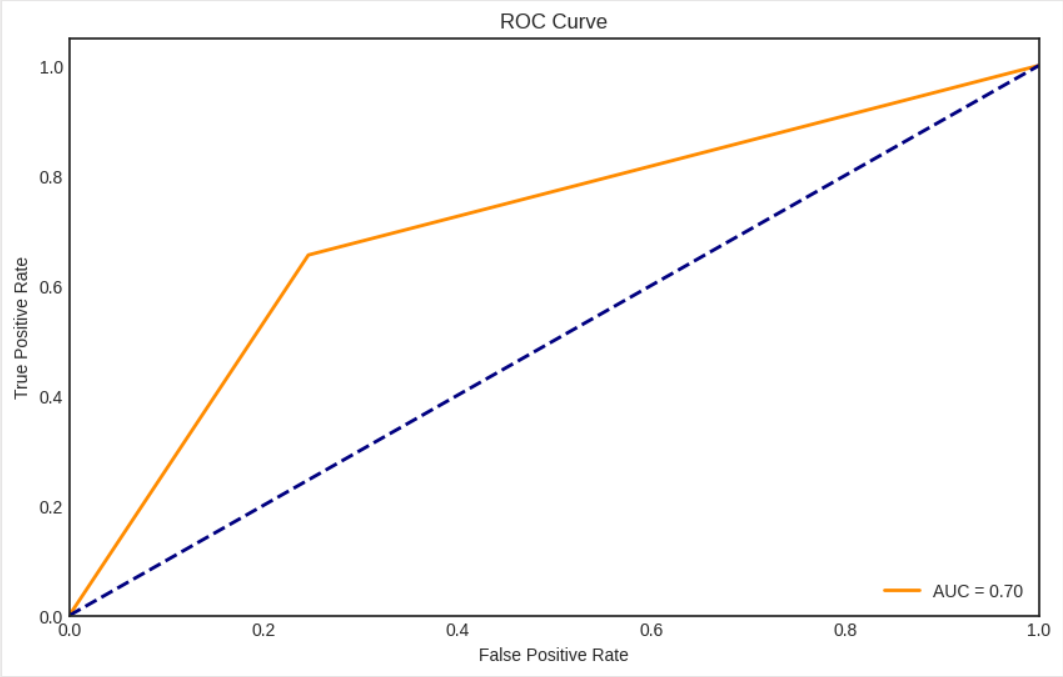
Tổng quan:

Có vẻ như mô hình có khả năng dự đoán đúng một phần đáng kể các trường hợp không đăng ký gửi tiền có kỳ hạn (TN), nhưng còn một số lượng không nhỏ dự đoán sai (FP). Nhìn chung, các chỉ số trên cho thấy mô hình phân lớp có hiệu suất khá tốt. Mô hình có thể dự đoán chính xác khoảng 70% các trường hợp.

Tính chỉ số AUC, trước khi tính chỉ số AUC thì chuyển encoding biến target với "yes" là 1, "no" là 0:

```
# Chuyển các giá trị yes, no sang 1, 0
data_test['deposit_encoded'] = data_test['deposit'].map({'yes': 1,
'no': 0})
y_pred_encoded = [1 if value == 'yes' else 0 for value in y_pred]
```

```
## ROC và AUC: phân lớp NHỊ PHÂN


fpr, tpr, thresholds = roc_curve(data_test['deposit_encoded'],
y_pred_encoded)
auc = metrics.auc(fpr, tpr) ## nếu gọi trực tiếp auc(fpr, tpr) sẽ bị
báo lỗi:  'numpy.float64' object is not callable
# Vẽ đường ROC
plt.figure(figsize=(10, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC = {auc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()
```



Hình 21:Đường cong ROC của thuật toán CBA

Với chỉ số AUC là 0.7 có thể được xem là một hiệu suất trung bình, kết quả này khá khả quan.

Những chỉ số đánh giá trên cho thấy mô hình khá tốt, nhưng những kết quả này chỉ dựa trên việc huấn luyện và đánh giá trên một tập train và một tập test nên chưa đủ để đánh giá mô hình có thật sự tốt hay không, nên thực hiện cross validation để chia mô hình ra 10 phần và thực hiện đánh giá với mỗi tập train và test.

```
block_size = int(len(df_cba) / 10)
split_point = [k * block_size for k in range(0, 10)]
split_point.append(len(df_cba))
accuracies = []
recalls = []

for k in range(len(split_point) - 1):
    print("\nRound %d:" % (k+1))
    data_test = df_cba[split_point[k]: split_point[k + 1]]
    data_train = df_cba[0: split_point[k]].append(
        df_cba[split_point[k + 1]:]
    )
    txns_train = TransactionDB.from_DataFrame(data_train)
    txns_test = TransactionDB.from_DataFrame(data_test)
    time_start = time.time()
    cba_model = cba.fit(txns_train)
    y_pred = cba_model.predict(txns_test)
    time_end = time.time()
    print(f"\ttime taken: {(time_end - time_start):.4f}s")
    accuracy = cba_model.rule_model_accuracy(txns_test)
    recall = recall_score(data_test['deposit'], y_pred,
pos_label='yes')
    print(f"\taccuracy: {accuracy:.4f}")
    print(f"\trecall: {recall:.4f}")
    accuracies.append(accuracy)
```

```
    recalls.append(recall)

print("")
# print(accuracies)
print(f"Mean accuracy = {np.mean(accuracies):.4f}")
print(f"Mean recall = {np.mean(recalls):.4f}")
```

Kết quả từng vòng như sau:

```
Round 1:
      time taken: 11.4792s
      accuracy: 0.4225
      recall: 0.4225

Round 2:
      time taken: 4.7989s
      accuracy: 0.6324
      recall: 0.6333

Round 3:
      time taken: 7.0443s
      accuracy: 0.8117
      recall: 0.8126

Round 4:
      time taken: 5.7154s
      accuracy: 0.8000
      recall: 0.8009

Round 5:
      time taken: 6.3700s
      accuracy: 0.6946
      recall: 0.7167
Round 6:
      time taken: 3.8371s
      accuracy: 0.6009
      recall: 0.0000

Round 7:
      time taken: 3.9766s
      accuracy: 0.6297
      recall: 0.0000

Round 8:
      time taken: 3.6901s
      accuracy: 0.6090
      recall: 0.0000

Round 9:
      time taken: 3.7968s
      accuracy: 0.6360
      recall: 0.0000

Round 10:
      time taken: 3.4582s
      accuracy: 0.6013
      recall: 0.0000
```

```
Mean accuracy = 0.6438
Mean recall = 0.3386
```

Tổng quan kết quả cross validation:

- Độ chính xác trung bình: 0.6438
- Tỷ lệ dự đoán đúng trung bình: 0.3386
- Độ lệch chuẩn của độ chính xác: 0.1046
- Độ lệch chuẩn của tỷ lệ dự đoán đúng: 0.3533

Phân tích từng vòng:

- Các vòng 3 và 4 đạt độ chính xác cao nhất (0.8117 và 0.8000) và tỷ lệ dự đoán đúng cao nhất (0.8126 và 0.8009).
- Các vòng 6, 7, 8 và 9 có độ chính xác thấp (0.6009, 0.6297, 0.6090 và 0.6360) và tỷ lệ dự đoán đúng bằng 0.
- Các vòng 1, 2 và 5 có độ chính xác và tỷ lệ dự đoán đúng nằm ở mức trung bình.

Điểm cần lưu ý:

- Độ lệch chuẩn của độ chính xác và tỷ lệ dự đoán đúng khá cao, điều này cho thấy sự không ổn định đáng kể giữa các vòng cross validation.
- Độ chính xác của mỗi vòng lặp dao động từ khoảng 0.4225 đến 0.8117 và recall cũng dao động từ 0.0000 đến 0.8126, thật sự có sự biến động lớn giữa các vòng lặp, điều này cho thấy **mô hình không tốt với mọi bộ dữ liệu**. Đối với một mô hình phân loại, recall là một chỉ số quan trọng, đặc biệt là nếu việc bỏ sót các trường hợp Positive là rủi ro lớn trong việc tìm kiếm những khách hàng tiềm năng.

3.2. Mô hình CBA tự xây dựng

Với mục đích để hiểu rõ thuật toán CBA, nhóm đã xây dựng một thuật toán CBA theo như đề xuất của của Bing Liu trong bài nghiên cứu "Integrating Classification and Association Rule Mining" năm 1998. Nhóm sẽ thực hiện 3 công việc là: Khai phá luật kết hợp, xây dựng bộ phân lớp và cuối cùng là kiểm thử, đánh giá mô hình.

Đầu tiên, nhóm sẽ chia bộ dữ liệu thành 2 tập train và tập test với tỷ lệ lần lượt 80:20 trên toàn bộ dữ liệu. Sau đó, các tập dữ liệu sẽ được đưa vào các list transaction để tiến hành chuyển dạng dữ liệu sau đó. Trong đó label của tập test sẽ được đưa vào

một list riêng với list các feature.

```
# Chia tập dữ liệu thành tập train và tập test theo tỉ lệ 80:20
train_dataset, test_dataset = train_test_split(df_cba, test_size=0.2,
random_state=42)
train_dataset = train_dataset.reset_index(drop=True)
test_dataset = test_dataset.reset_index(drop=True)

# Đưa dữ liệu tập train và tập test vào các list
test_label = []
train_transaction = []
test_transaction = []

for _, row in train_dataset.iterrows():
    temp = [f"{column}={value}" for column, value in row.iteritems()]
    train_transaction.append(temp)

for _, row in test_dataset.iterrows():
    temp = [f"{column}={value}" for column, value in row.iteritems()]
    test_label.append(temp[13:])
    test_transaction.append(temp[:13])
```

3.2.1. Khai phá luật kết hợp

Chuyển dạng dữ liệu thành dạng bảng ma trận boolean bằng TransactionEncoder(). Thực hiện với hiện với các tập train và test.

```
# Chuyển dạng dữ liệu thành dạng bảng ma trận boolean
te = TransactionEncoder()
te_ary = te.fit(train_transaction).transform(train_transaction)
df_train = pd.DataFrame(te_ary, columns=te.columns_)

te_ary = te.fit(test_transaction).transform(test_transaction)
df_test = pd.DataFrame(te_ary, columns=te.columns_)
```

Ta thu được kết quả dữ liệu như sau

Tiếp theo, ta sử dụng thuật toán Apriori để tìm luật kết hợp mạnh theo các bước dưới đây, minSup và minConf lần lượt là 0.05 và 0.6, bên cạnh đó ta thiết lập giá trị lift tối thiểu là 1:

```
# Khai phá tập phổ biến bằng thuật toán Apriori
df_apriori = apriori(df_train, min_support=0.05, use_colnames=True)
df_apriori.itemsets
```

```
# Khai phá luật kết hợp với minconf = 0.6 và minlift = 1
rule_APRIORI = association_rules(df_apriori, metric="confidence",
min_threshold= 0.6)
rule_APRIORI = rule_APRIORI[rule_APRIORI.lift >
1].sort_values('confidence', ascending=False)
```

Tạo 2 danh sách luật, mỗi danh sách chứa các luật phân cho 2 lớp riêng biệt:

```
# Tạo danh sách các luật với consequents có chứa biến target
```

```
deposit_y_rules =
rule_APRIORI[rule_APRIORI['consequents'].apply(lambda x: 'deposit=yes'
in x)]
deposit_n_rules =
rule_APRIORI[rule_APRIORI['consequents'].apply(lambda x: 'deposit=no'
in x)]
```

3.2.2. Xây dựng trình phân lớp

```
def predict_with_rules(instance, rules):
    predicted_class = []
    max_confidence = 0
    temp_pred = []
    for _, rule in rules.iterrows():
      if all(item in instance for item in rule['antecedents']):
        if rule['confidence'] > max_confidence:
          temp_pred = rule['consequents']
          max_confidence = rule['confidence']
    predicted_class = list(temp_pred)
    return predicted_class

# Testing on test dataset
test_instances = df_test.apply(lambda row: [col for col in
df_test.columns if row[col]], axis=1)
predictions = test_instances.apply(lambda instance:
predict_with_rules(instance, deposit_y_rules))
```

Hàm predict_with_rule(instance, rules) thực hiện phân lớp với các luật kết hợp cho biến target deposit là yes. Lần lượt với mỗi trường hợp, ta duyệt tất cả các luật qua trường hợp. Nếu trường hợp có chứa luật kết hợp thì gán nhãn cho luật có độ chắc chắn cao nhất. Ta thu được một list dự đoán các trường hợp có deposit là yes. Tiếp theo, ta sẽ gán những trường hợp còn lại chưa được phân lớp cho nhãn deposit là no.

```
final_prediction = []
for i in predictions:
  if 'deposit=yes' in i:
    final_prediction.append(['deposit=yes'])
  else:
    final_prediction.append(['deposit=no'])
len(final_prediction)
```

3.2.3. Đánh giá mô hình

Thực hiện đánh giá mô hình qua các chỉ số Accuracy, Precision, Recall và F1 Score:

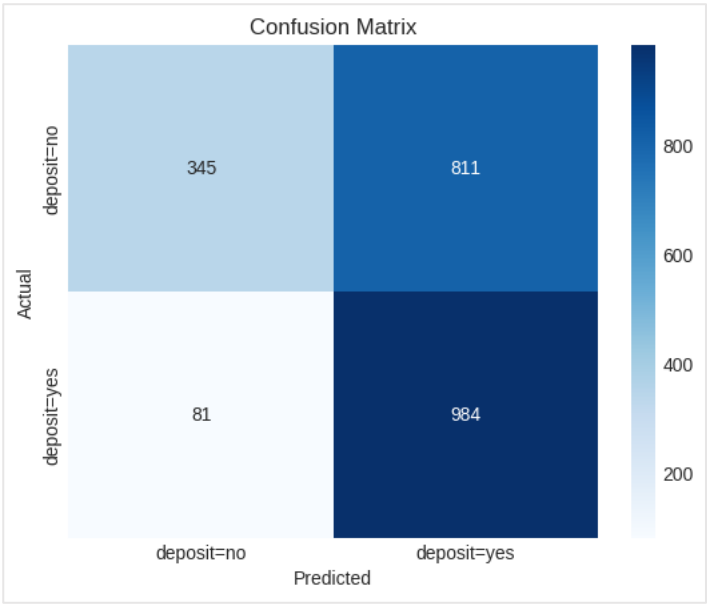
```
def evaluate(y_true, y_pred):

    # Tính các thước đo chính
```

```
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred,
pos_label='deposit=yes')
    recall = recall_score(y_true, y_pred, pos_label='deposit=yes')
    f1 = f1_score(y_true, y_pred, pos_label='deposit=yes')

    ## Hiển thị giá trị cá chỉ số
    print(f'Accuracy = {(accuracy * 100):.1f}%')
    print(f'Precision = {(precision * 100):.1f}%')
    print(f'Recall    = {(recall * 100):.1f}%')
    print(f'F1 Score  = {(f1 * 100):.1f}%')

evaluate(y_true, y_pred)
```

```
Accuracy = 59.8%
Precision = 54.8%
Recall    = 92.4%
F1 Score  = 68.8%
```

Qua kết quả các chỉ số ta có thể thấy, chỉ số Accuracy là 59,8% cho thấy mô hình dự đoán kết quả chưa thực sự chính xác. Ngoài ra, chỉ số Recall rất cao là 92,4% trong khi chỉ số Precision chỉ khoảng 54,8%. Cho thấy mô hình dự đoán phủ được nhiều trường hợp deposit là yes trong thực tế, nhưng lại có nhiều dự đoán deposit là yes nhưng deposit là no trong thực tế. Quan sát ma trận nhầm lẫn để thấy rõ hơn những điều trên:

```
conf_matrix_cba = confusion_matrix(y_true, y_pred)

# Vẽ ma trận nhầm lẫn bằng seaborn
sns.heatmap(conf_matrix_cba,    annot=True,    fmt='d',    cmap='Blues',
xticklabels=['deposit=no',  'deposit=yes'],  yticklabels=['deposit=no',
'deposit=yes'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



Hình 22: Ma trận nhầm lẫn mô hình CBA tự xây dựng

Để giải thích cho điều này, do nhóm sử dụng luật kết hợp phân lớp deposit là yes để phân lớp trước và gán deposit là no cho những trường hợp còn lại. Nên số trường hợp được gán nhãn yes rất nhiều trong khi deposit = no lại rất ít. Tuy nhiên, nếu xét cả 2 danh sách luật (tức là tất cả luật được sinh ra bởi Apriori) thì thời gian thực hiện thuật toán sẽ rất lâu (hơn 1 giờ đồng hồ). Nên ta có thể đánh giá mô hình này không được hiệu quả và có thể thực hiện phiên bản cải tiến của CBA (hay CBA-M2).

3.3. Mô hình cây quyết định

3.3.1. Tiền xử lý dữ liệu

Trước khi đưa dữ liệu vào mô hình cây quyết định, thực hiện tiền xử lý dữ liệu bằng cách ánh xạ các thuộc tính liên tục đã được rời rạc hóa đến các số nguyên dương tiên tiếp, và ánh xạ các thuộc tính phân loại đến các số nguyên dương liên tiếp.

Dữ liệu số

- Biến Age

```
# Áp dụng label encoding cho cột age_group (vì Label_encoding gán theo
thứ tự tăng dần)
label_encoder = LabelEncoder()
df['age_encoded'] = label_encoder.fit_transform(df['age_bins'])+1
print(dict(zip(label_encoder.classes_,
label_encoder.transform(label_encoder.classes_)+1)))
```

```
{'18_27': 1, '28_44': 2, '45_64': 3, '65_84': 4, '85_100': 5}
```

- Biến campaign

```
# Áp dụng label encoding cho cột campaign_bins (vì Label_encoding gán
```

```
theo thứ tự tăng dần)
df['campaign_encoded'] =
label_encoder.fit_transform(df['campaign_bins'])+1
print(dict(zip(label_encoder.classes_,
label_encoder.transform(label_encoder.classes_)+1)))
```

```
{'11_15': 1, '16_20': 2, '1_5': 3, '21_25': 4, '6_10': 5}
```

- Biến balance

```
mapping_dict = {'debt': 0, 'low': 1, 'moderate': 2, 'high': 3,
'very_high': 4, 'extremely_high': 5}
df['balance_encoded'] = df['balance_bins'].map(mapping_dict)
```

- Biến duration

```
mapping_dict = {'under_15m': 1, '15_30m': 2, '30_45m': 3, '45_60m': 4}
df['duration_encoded'] = df['duration_bins'].map(mapping_dict)
```

- Biến pdays

```
mapping_dict = {'no': 0, 'under_3_months': 1, '3_6months': 2,
'6months_1year': 3, '1_2years': 4}
df['pdays_encoded'] = df['pdays_bins'].map(mapping_dict)
```

- Biến previous

```
mapping_dict = {'no': 0, '1_10': 1, '11_20': 2, '21_30': 3}
df['previous_encoded'] = df['previous_bins'].map(mapping_dict)
```

Dữ liệu phân loại

- Biến job

```
# Thực hiện frequent encoding
mapping_dict = df['job'].value_counts().to_dict()
df['job_encoded'] = df['job'].map(mapping_dict)
```

- Biến marital

```
# Thực hiện Label Encoding đối với cột marital, với single=0,
divorced=1, married=2
mapping_dict = {'single': 1, 'divorced': 2, 'married': 3}
df['marital_encoded'] = df['marital'].map(mapping_dict)
```

- Biến education

```
# Thực hiện Label Encoding đối với cột education_encoded, với
unknown=0, primary=1, secondary=2, tertiary=3
mapping_dict = {'unknown': 0, 'primary': 1, 'secondary': 2,
'tertiary': 3}
df['education_encoded'] = df['education'].map(mapping_dict)
```

- Biến default

```
mapping_dict = {'no': 0, 'yes': 1}
```

```
df['default_encoded'] = df['default'].map(mapping_dict)
```

- Biến housing

```
mapping_dict = {'no': 0, 'yes': 1}
df['housing_encoded'] = df['housing'].map(mapping_dict)
```

- Biến loan

```
mapping_dict = {'no': 0, 'yes': 1}
df['loan_encoded'] = df['loan'].map(mapping_dict)
```

- Biến contact

```
mapping_dict = {'unknown': 0, 'telephone': 1, 'cellular': 2}
df['contact_encoded'] = df['contact'].map(mapping_dict)
```

-

Biến deposit

```
mapping_dict = {'no': 0, 'yes': 1}
df['deposit_encoded'] = df['deposit'].map(mapping_dict)
```

Lấy những cột đã encoding và thu được dữ liệu sau khi xử lý như sau:

	age_encoded	job_encoded	marital_encoded	education_encoded	default_encoded	balance_encoded	housing_encoded	loan_encoded	contact_encoded	duration_encoded	campaign_encoded	pdays_encoded	previous_encoded	deposit_encoded
0	3	1331	3	2	0	2	1	0	0	2	3	0	0	1
1	3	1331	3	2	0	1	0	0	0	2	3	0	0	1
2	2	1813	3	2	0	1	1	0	0	2	3	0	0	1
3	3	918	3	2	0	2	1	0	0	1	3	0	0	1
4	3	1331	3	3	0	1	0	0	0	1	3	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11096	2	1935	1	1	0	1	1	0	2	1	3	0	0	0
11097	2	918	3	2	0	1	0	0	0	1	3	0	0	0
11098	2	1813	1	2	0	1	0	0	2	1	3	0	0	0
11099	2	1813	3	2	0	0	0	1	2	1	3	2	1	0
11100	2	1813	3	2	0	0	0	0	2	1	3	0	0	0

11101 rows × 14 columns

Hình 23: Dữ liệu sau khi xử lý

3.3.2. Xây dựng mô hình cây quyết định và đánh giá

Chia tập dữ liệu thành 2 tập train và test theo tỷ lệ 80:20:

```
## Các features:
X = df_tree.drop(['deposit_encoded'], axis = 1)

## Biến target: y_encoded
y = df_tree.deposit_encoded

## Chia tập dữ liệu thành training, test sets theo tỷ lệ 80:20
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 42)
```

Xây dựng mô hình:

```
## Xây dựng mô hình Decision Tree
clf   = DecisionTreeClassifier()
model = clf.fit(X, y) # huấn luyện để tạo  mô hình
## Kiểm thử mô hình
y_pred_tree = clf.predict(X_test)
```

Đánh giá mô hình bằng các chỉ số:

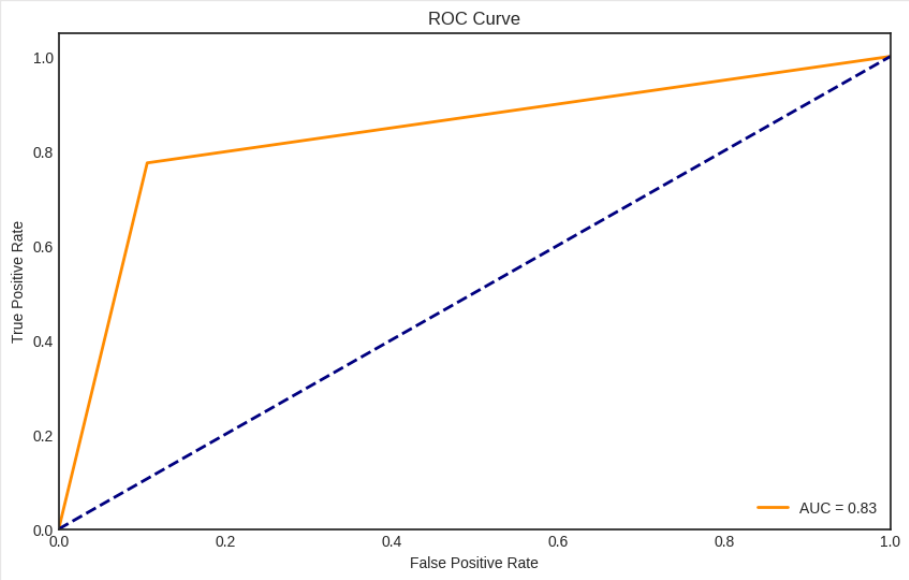
```
## Đánh giá mô hình bằng các chỉ số
scores = classification_eval(y_test, y_pred_tree)

## ROC và AUC: phân lớp NHỊ PHÂN
fpr, tpr, thresholds = roc_curve(y_test, y_pred_tree)
auc = metrics.auc(fpr, tpr) ## nếu gọi trực tiếp auc(fpr, tpr) sẽ bị
báo lỗi:  'numpy.float64' object is not callable

## Hiển thị giá trị cá chỉ số
print(f'accuray   = {(scores[0] * 100):.1f}%')
print(f'precision = {(scores[1] * 100):.1f}%')
print(f'recall    = {(scores[2] * 100):.1f}%')
print(f'f1        = {(scores[3] * 100):.1f}%')
print(f'AUC       = {(auc * 100):.1f}%')
```

Thu được kết quả như sau:

```
accuray   = 83.7%
precision = 87.0%
recall    = 77.5%
f1        = 82.0%
AUC       = 83.4%
```

Vẽ đường AUC:

```
# Vẽ đường ROC
plt.figure(figsize=(10, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC = {auc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()
```



Hình 24: Đường cong ROC của mô hình cây quyết định

Thực hiện cross validation:

```
# Cross validation
scores = cross_val_score(clf, X, y, cv=5)
scores.mean()
```

```
0.6466140566500764
```

Quan sát các chỉ số đánh giá cây quyết định:

Ta thấy chỉ số accuracy là 83.7% với mô hình cây quyết định, so sánh với mô hình trên thì có thể thấy rằng độ chính xác của mô hình cây quyết định đang tốt hơn nhiều so với mô hình phân lớp dựa trên luật kết hợp.

Bên cạnh accuracy, ta có các chỉ số khác như recall, f1, precision hay AUC đều tốt hơn nhiều so với mô hình phân lớp dựa trên luật kết hợp.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. KẾT LUẬN

Nhìn chung, mô hình có độ chính xác (Accuracy) trung bình là 0.7060, đây là một ưu điểm và cho thấy khả năng phân loại chính xác và ổn định qua các chu kỳ cross-validation.

Bên cạnh đó, độ chính xác của Positive (Precision) là 71.0%, một con số đáng chú ý. Điều này có nghĩa là mô hình có khả năng nhận diện chính xác các trạng thái Positive, đồng thời giả mạo Positive ở mức thấp. Đây là điểm quan trọng trong các ứng dụng thực tế, nơi việc tránh bị giả mạo là quan trọng để đảm bảo tính chính xác của dự đoán.

Ngoài ra, thời gian thực hiện Cross Validation giảm dần qua các vòng lặp mà vẫn đảm bảo sự hiệu quả của mô hình, cho thấy tính linh hoạt và hiệu suất ổn định của mô hình trong quá trình đào tạo và đánh giá. Thời gian thực hiện mỗi vòng lặp giảm dần cũng đồng nghĩa với việc mô hình có khả năng xử lý dữ liệu một cách hiệu quả, giảm thiểu tối đa thời gian và nguồn lực cần thiết cho quá trình huấn luyện.

2. HẠN CHẾ

Độ nhạy thấp (Low Recall): Độ nhạy trung bình của mô hình chỉ đạt mức 0.3386, chỉ số tương đối thấp, đồng nghĩa với việc mô hình có xu hướng bỏ sót một lượng lớn các trường hợp Positive. Trong bối cảnh mà bài toán này ưu tiên độ nhạy cao để đảm bảo khả năng dự đoán đối với những khách hàng tiềm năng, chỉ số độ nhạy khá thấp này trở thành một hạn chế lớn đối với mô hình.

Sự biến động đáng kể trong độ chính xác và recall qua các vòng lặp của cross-validation là một thách thức đáng chú ý. Sự biến động này không chỉ ảnh hưởng đến tính ổn định của mô hình mà còn làm giảm độ tin cậy của các kết quả đánh giá. Việc này cần được xem xét và có thể đòi hỏi các điều chỉnh hoặc cải thiện trong quá trình huấn luyện.

Điều đặc biệt đáng lo ngại là trong một số vòng lặp của cross-validation, độ nhạy (recall) của mô hình bằng 0. Nó chỉ ra rằng mô hình không nhận diện được bất kỳ trường hợp Positive nào trong các vòng lặp đó, gây ra mất mát thông tin lớn và làm suy giảm độ tin cậy của mô hình.

Tóm gọn, kết quả chỉ ra rằng thuật toán CBA không tối ưu với bộ dữ liệu và có hiệu suất còn kém hơn so với thuật toán cây quyết định. Qua đây ta cần xem xét sự thay

thế hoặc tinh chỉnh thuật toán để cải thiện hiệu suất phân loại.

Cuối cùng, phương pháp phân lớp kết hợp (CBA) dù có mục tiêu là nhằm nâng cao hiệu quả khai thác luật dựa trên độ chính xác tiên đoán trên phần dữ liệu dùng để kiểm tra nhưng chỉ dựa vào một mình độ tin cậy để đánh giá chất lượng của một luật trong việc phân lớp một mẫu mới là chưa đủ.

Vì vậy thuật toán được cải tiến của phân lớp kết hợp CBA được đề xuất bởi Liu et al 1998 đã giải quyết hai hạn chế về sử dụng một độ hỗ trợ duy nhất cho toàn bộ cơ sở dữ liệu và việc khai thác các luật dài dòng. Hạn chế đầu tiên được giải quyết bằng cách sử dụng nhiều độ hỗ trợ tối thiểu khác nhau tùy vào tỷ lệ từng lớp trong cơ sở dữ liệu, và hạn chế thứ hai được giải quyết bằng cách tích hợp CBA với cây quyết định và phương pháp Bayesian Naive

3. HƯỚNG PHÁT TRIỂN

Tối ưu hóa mô hình: Xem xét cấu trúc mô hình hiện tại để có thể tối ưu hóa. Có thể cần thay đổi kiến trúc mạng, số lượng lớp, hoặc kích thước lớp để cải thiện khả năng học của mô hình. Tiến hành điều chỉnh siêu tham số, chẳng hạn như tốc độ học và kích thước mẫu, để đạt được sự cân bằng giữa độ nhạy và độ chính xác và giảm biến động trong quá trình cross-validation.

Tăng cường dữ liệu: Nếu có thể, xem xét việc tăng cường dữ liệu bằng cách thêm dữ liệu mới hoặc tạo ra biến thay thế để cải thiện khả năng tổng quát hóa của mô hình. Điều này có thể giúp mô hình học được các đặc trưng quan trọng hơn và giảm khả năng overfitting.

Lựa chọn các thuật toán khác: Kết quả cho thấy thuật toán cây quyết định cho ra các kết quả tốt hơn nhiều so với thuật toán CBA trên bộ dữ liệu này. Thực hiện thử nghiệm với một số phương pháp đã được đề xuất để xây dựng một bộ phân lớp với các luật kết hợp chất lượng cao như phân lớp kết hợp dựa trên luật kết hợp đa lớp MCAR, phương pháp khai thác kết hợp phổ biến PAM, phân lớp dựa trên luật kết hợp đa nhãn CMAR,... để xem xét liệu có thuật toán nào phù hợp và đem lại hiệu suất cao hơn trên dữ liệu cụ thể này. Những kỹ thuật này sử dụng những cách tiếp cận khác nhau để thực hiện các giai đoạn phát hiện các mẫu thường xuyên, các luật, quy định cấp bậc, tỉa luật dư thừa hoặc có hại (các luật dẫn đến việc phân lớp không chính xác) và phân lớp đối tượng thử nghiệm mới.

TÀI LIỆU THAM KHẢO

Rathi, P. (2020). Banking dataset - marketing targets. Retrieved from https://www.kaggle.com/datasets/prakharrathi25/banking-dataset-marketing-targets/data?fbclid=IwAR35nvVOtVqdqXwaDaCmZFL0sL2jUAUDp33-_OBuSyrv-ySBVEYn5vvPMAo

Moro, Sérgio & Cortez, Paulo & Rita, Paulo. (2014). A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems. 62. 10.1016/j.dss.2014.03.001.

Aditya. (2023, March 26). Association rule mining explained with examples. Retrieved from https://codinginfinite.com/association-rule-mining-explained-with-examples/#htoc-advantages-of-association-rule-mining

Han, Jiawei & Kamber, Micheline & Pei, Jian. (2012). Data Mining: Concepts and Techniques. 10.1016/C2009-0-61819-5.

I. T. S. A. Agrawal R, "Mining association rules between sets of items in large databases," in The 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD'93) , Washington, 1993.

R. a. R. S. Agarwal, "Fast algorithms for mining association rules," in The 20th International Conference on Very, Santiago de Chile, 1994.

Ali, K., Manganaris, S. and Srikant, R.: Partial classification using association rules, Proceedings of the Third ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD-97, ACM, 115-118, (1997).

Bing Liu, Wynne Hsu, and Yiming Ma. 1998. Integrating classification and association rule mining. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98). AAAI Press, 80–86.

Wenmin Li, Jiawei Han and Jian Pei, "CMAR: accurate and efficient classification based on multiple class-association rules," Proceedings 2001 IEEE International Conference on Data Mining, San Jose, CA, USA, 2001, pp. 369-376, doi: 10.1109/ICDM.2001.989541.

Yin, Xiaoxin & Han, Jiawei. (2003). CPAR: Classification Based on Predictive Association Rules. In: SDM'03. 3. 10.1137/1.9781611972733.40.

Liu, Bing & Ma, Yiming & Wong, Ching-Kian. (2001). Classification Using Association Rules: Weaknesses and Enhancements. Massive Computing. 10.1007/978-

1-4615-1733-7_30.

Segrera, Saddys & Moreno García, María. (2007). Classification Based on Association Rules for Adaptive Web Systems. 44. 10.1007/978-3-540-74972-1_58.

PHÂN CÔNG CÔNG VIỆC

STT	Họ và tên	MSSV	Công việc	Đánh giá
1	Đỗ Quang Thiên Phú	31211024191	- Chương 1: Mở đầu - Chương 2: Phân lớp sử dụng mẫu phổ biến - Chương 4 (3): Xây dựng mô hình	100%
2	Lê Trần Khánh Phú	31211024087	- Chương 4 (3): Xây dựng mô hình	100%
3	Lê Duy Phụng	31211027604	- Chương 3: Tổng quan nghiên cứu - Chương 5: Kết luận và hướng phát triển	100%
4	Phạm Minh Phước	31211027663	- Chương 4 (1): Phân tích khám phá dữ liệu - Chương 4 (2): Tiền xử lý dữ liệu	100%
5	Trương Vũ Phương Quỳnh	31211027668	- Lời cảm ơn - Tổng hợp word - Làm slide thuyết trình	100%