BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC KINH TẾ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

UEH
UNIVERSITY

ĐỒ ÁN MÔN HỌC

MÁY HỌC

Đề tài: ***PHÂN CỤM KHÁCH HÀNG SỬ DỤNG THẺ TÍN DỤNG BẰNG GIẢI THUẬT BRICH.***

Giảng viên: TS. Nguyễn An Tế

Mã lớp học phần: 23C1INF50904401

Thành viên: Nguyễn Đình Toàn - 31211027676

Nguyễn Thị Phương Thảo - 31211027671

Lý Gia Thuận - 31211026754

Phạm Quốc Thuận - 31211027674

Huỳnh Thị Ngọc Trâm - 31211027679

Thành phố Hồ Chí Minh , ngày 11 tháng 12 năm 2023

Lời cảm ơn **3**
Chương I: Tổng quan đề tài **4**
1.1 Giới thiệu đề tài: 4
1.2 Mục tiêu nghiên cứu: 4
1.3 Phương pháp nghiên cứu: 4
1.4 Tài nguyên sử dụng: 4
1.5 Ngôn ngữ sử dụng:: 4
Chương II: Tổng quan bộ dữ liệu **5**
2.1 Mô tả tổng quan bộ dữ liệu: 5
2.2 Mô tả thuộc tính: 5
Chương III: Tổng quan thuật toán **7**
3.1 Giới thiệu tổng quan về các thuật toán phân cụm: 7
3.2 Giới thiệu về thuật toán Birch: 8
3.2.1 Giới thiệu tổng quan về Hierarchical Clustering: 8
3.2.2 Giới thiệu sơ bộ về thuật toán: 9
3.2.3 Các điểm đặc trưng của thuật toán BIRCH 11
3.3 Cấu trúc giải thuật : 12
3.4 Ví dụ minh họa: 14
3.5 Hướng áp dụng vào đề tài: 17
Chương IV: Tiền xử lý dữ liệu **18**
4.1 Phân tích dữ liệu được thu thập: 18
4.2 Tiền xử lý dữ liệu: 22
4.2.1 Xử lý các cột dữ liệu: 22
4.2.2 Xử lý missing data: 22
4.2.3 Xử lý Outliers: 26
Chương V: Xây dựng và đánh giá mô hình **28**
5.1 Xây dựng mô hình: 28
5.1.1 Chuẩn hóa mô hình: 28
5.1.2 Chọn Threshold và Branching Factor tốt nhất (Cross-Validation): 28
5.1.3 Chọn số cụm tốt nhất: 31
5.1.4 Đưa vào mô hình phân cụm: 37
5.1.5 Biểu diễn kết quả đạt được: 38
5.2 Đánh giá kết quả phân cụm: 39
5.2.1 Đánh giá: 39
5.2.2 Hướng phát triển cho bộ dữ liệu: 43

Thành viên	Phân công nhiệm vụ	Mức độ hoàn thành
Nguyễn Đình Toàn	- Tiền xử lý dữ liệu. - Xây dựng mô hình. - Biểu diễn trực quan, nhận xét sau khi phân cụm.	**100%**
Nguyễn Thị Phương Thảo	- Tổng quan đề tài. - Tổng quan về thuật toán BIRCH. - Hướng áp dụng vào đề tài.	**100%**
Lý Gia Thuận	- Tiền xử lý dữ liệu. - Xây dựng mô hình. - Lý thuyết và đánh giá mô hình.	**100%**
Phạm Quốc Thuận	- Lời cảm ơn. - EDA, Phân tích đơn biến, đa biến. - Tinh chỉnh bài báo cáo.	**100%**
Huỳnh Thị Ngọc Trâm	- Tổng quan bộ dữ liệu. - Tổng quan về các thuật toán phân cụm. - Cấu trúc giải thuật.	**100%**

Lời cảm ơn

Nhóm chúng em xin gửi lời cảm ơn sâu sắc và lòng biết ơn của mình đến giảng viên hướng dẫn, TS. Nguyễn An Tế. Cảm ơn thầy đã giúp chúng em có một hành trình học tập khó quên trong môn Máy học.

Cảm ơn thầy đã hướng dẫn chúng em không chỉ về lý thuyết mà còn về ứng dụng thực tế của Machine Learning. Sự hiểu biết sâu rộng của thầy về lĩnh vực này đã giúp chúng em có những nhận thức đúng đắn về môn học và về lĩnh vực mình đang theo đuổi. Qua sự hướng dẫn của thầy chúng em đã biết cách áp dụng kiến thức đã học được vào thực tế.

Thầy không chỉ là người hướng dẫn mà còn là người tạo cho chúng em những động lực giúp chúng em vượt qua những thử thách và phát triển bản thân. Sự nhiệt huyết và lòng tận tâm của thầy là nguồn động lực lớn giúp chúng tôi vươn lên.

Trong quá trình hoàn thành đồ án này, có thể chúng em còn nhiều thiếu sót. Chúng em mong có được sự góp ý từ thầy để ngày càng hoàn thiện bản thân cho những dự án mà chúng em tham gia sau này.

Một lần nữa, chúng em cảm ơn và biết ơn thầy sâu sắc, TS. Nguyễn An Tế.

Chương I: Tổng quan đề tài

1.1 Giới thiệu đề tài:

Trong thời đại số ngày nay, việc sử dụng các phương tiện thanh toán công nghệ đang ngày càng thay thế các phương thức thanh toán truyền thống. Các công cụ như thẻ ngân hàng, thẻ tín dụng, và ví điện tử đang chiếm ưu thế hơn vì sự tiện lợi trong các giao dịch của người tiêu dùng. Trong số này, thẻ tín dụng đặc biệt là một phương tiện thanh toán phổ biến, được sử dụng rộng rãi trong các giao dịch mua sắm hàng ngày.

Nhận thức về sự quan trọng của việc này, nhóm đã tiến hành phân tích trên bộ dữ liệu "Customer Dataset", chứa thông tin của khoảng 9000 khách hàng sử dụng thẻ tín dụng trong 6 tháng. Mục tiêu của nhóm là thông qua phân tích dữ liệu, hiểu rõ hơn về hành vi của khách hàng khi sử dụng thẻ tín dụng, và đồng thời phân loại họ thành các nhóm có đặc điểm chung. Điều này giúp doanh nghiệp có cái nhìn tổng quan về thị trường, từ đó xây dựng chiến lược tiếp thị hiệu quả và phù hợp.

1.2 Mục tiêu nghiên cứu:

Bộ dữ liệu này cung cấp thông tin chi tiết về người dùng thẻ tín dụng trong khoảng thời gian 6 tháng. Qua quá trình xử lý, phân tích, và gom cụm dữ liệu, mục tiêu của nhóm là phân loại khách hàng sử dụng thẻ tín dụng thành các nhóm dựa trên hành vi của họ. Điều này giúp tạo ra cái nhìn rõ ràng về đặc điểm chung trong từng nhóm, từ đó đề xuất các chiến lược tiếp thị phù hợp và hiệu quả hơn.

1.3 Phương pháp nghiên cứu:

- Phân tích tổng quan các biến
- Sử dụng thuật toán BIRCH để phân cụm khách hàng

1.4 Tài nguyên sử dụng:

- Bộ dữ liệu Customer Dataset lấy từ Kaggle

1.5 Ngôn ngữ sử dụng::

- Ngôn ngữ lập trình phân tích dữ liệu Python

Chương II: Tổng quan bộ dữ liệu

2.1 Mô tả tổng quan bộ dữ liệu:

Bộ dữ liệu "Customer Dataset" được thu thập để ghi lại hành vi sử dụng thẻ tín dụng của khách hàng trong thời gian 6 tháng. Tập dữ liệu gồm có 8950 dòng, mỗi dòng là bản ghi về thông tin tín dụng của một khách hàng được mô tả thông qua 18 thuộc tính liên quan.

2.2 Mô tả thuộc tính:

Cụ thể ý nghĩa của các thuộc tính trong bộ dữ liệu như sau:

STT	Tên thuộc tính	Ý nghĩa
1	**cust_id**	Mã khách hàng
2	**balance**	Số dư tài khoản
3	**balance_frequency**	Tần suất cập nhật tài khoản
4	**purchases**	Tổng số tiền giao dịch từ tài khoản
5	**oneoff_purchases**	Số tiền tối đa trong 1 lần mua
6	**installments_purchases**	Số tiền trả góp là số tiền mà khách đã chi cho các giao dịch trả góp
7	**cash_advance**	Tiền mặt trả trước
8	**purchases_frequency**	Tần suất mua hàng
9	**oneoff_purchases_frequency**	Tần suất mua sắm trả trong một lần (không trả góp)
10	**purchases_installments_frequency**	Tần suất mua sắm trả góp
11	**cash_advance_frequency**	Tần suất thanh toán tiền mặt trả trước
12	**cash_advance_trx**	Số lượng giao dịch "tiền mặt trả trước" (biến rời rạc)
13	**purchases_trx**	Số lượng mua sắm
14	**credit_limit**	Hạn mức tín dụng
15	**payments**	Số tiền thanh toán

16	**minimun_payments**	Số tiền thanh toán ít nhất
17	**prc_full_payment**	Tỷ lệ thanh toán đầy đủ của người dùng
18	**tenure**	Thời hạn tín dụng

Bảng 1. Ý nghĩa các thuộc tính

Chương III: Tổng quan thuật toán

3.1 Giới thiệu tổng quan về các thuật toán phân cụm:

Gom cụm (clustering) là một phương pháp học máy không giám sát (unsupervised learning) nhằm gom các nhóm dữ liệu có cùng đặc trưng lại thành cụm. Các cụm có thể rời hoặc không rời nhau và cũng cần đến tri thức lĩnh vực để đưa ra ý nghĩa của các cụm.

Các ứng dụng của việc phân cụm có thể kể đến như khám phá dữ liệu, phân tích phân khúc khách hàng, phát hiện outliers, phân chia hình ảnh… Ngoài ra, phân cụm dữ liệu có thể là một bước tiền xử lý hữu ích trước khi xây dựng mô hình học máy bằng thuật toán khác, bằng cách xây dựng các mô hình riêng biệt cho từng cụm giúp cải thiện độ chính xác và hiệu quả của mô hình.

Trong thời đại dữ liệu bùng nổ như hiện nay, các thuật toán phân cụm cũng đang dần lấy lại được sự chú ý. Tuy nhiên, lại có hai vấn đề:

- Thứ nhất là vấn đề về hiệu suất của thuật toán gom cụm, vì đặc trưng độ phức tạp thời gian của thuật toán cao ($O(N^2)$ hoặc O(NM) với N là số lượng điểm dữ liệu, M là số lượng cụm) nên khó có thể xử lý tốt và nhanh đối với lượng dữ liệu lớn.
- Vấn đề thứ hai ta cần quan tâm là việc xác định số lượng cụm. Theo phương pháp thông thường trước đây là dùng một trong những thuật toán phân cụm yêu cầu số lượng cụm làm tham số đầu vào, rồi lần lượt chạy thuật toán với mỗi giá trị k. Về sau, ra đời phương pháp *Elbow* của *Catherine A. Sugar* (năm 1998) được dùng để xác định số k tối ưu nhất. Tuy nhiên tất cả những phương pháp này đều làm tăng đáng kể thời gian tính toán, đặc biệt là khi không có nhiều thông tin để xác định trước phạm vi số cụm có thể nhỏ. Một số thuật toán phân cụm có thể tìm thấy số lượng cụm trực tiếp, mà không cần phải chạy thuật toán cho tất cả các số cụm .

Loại thuật toán	**Mô tả tổng quan**	**Các thuật toán phổ biến**
Density-based *(phân cụm dựa trên mật độ)*	Phương pháp học máy không giám sát, dựa trên mật độ của các điểm dữ liệu mà gom chúng thành cụm. Chúng phân tách thành các vùng có mật độ cao và gán các điểm dữ liệu trong vùng này vào cùng một cụm	DBSCAN, OPTICS
Distribution-based *(phân cụm dựa trên phân phối)*	Dữ liệu được chia thành cụm dựa trên xác suất cao nhất mà nhóm dữ liệu thuộc về cùng kiểu phân phối. Thông thường, các thuật toán phân cụm dựa trên phân phối sẽ giả định dữ liệu tuân theo phân phối chuẩn (*Gauss*).	Expectation - Maximization (EM)

Centroid-based *(phân cụm phân vùng)*	Là thuật toán phân cụm lặp dựa trên tính tương đồng giữa các điểm dữ liệu và tâm của các cụm	K-Means, CLARANS
Hierarchical Clustering *(phân cụm phân cấp)*	Thuật toán phân cụm phân cấp, thay thế cho phương phân cụm phân vùng (centroid-based), với phương pháp này không cần phải xác định trước k cụm. Thay vào đó thuật toán này sẽ tạo ra một cấu trúc dạng cây (dendrogram), bằng cách cắt cây ở mức mong muốn thì có thể chọn được số cụm mong muốn	BIRCH, BANG

Bảng 2. Các loại thuật toán Phân cụm [1]

Mỗi loại thuật toán sẽ có những ưu và nhược điểm riêng, tùy thuộc vào mục đích và bộ dữ liệu thực hiện.

3.2 Giới thiệu về thuật toán Birch:

3.2.1 Giới thiệu tổng quan về Hierarchical Clustering:

Hierarchical Clustering là các phương pháp phân cụm phân cấp, phương pháp này sắp xếp một tập dữ liệu đã thành một cấu trúc dạng cây. Cây phân cấp có thể được xây dựng theo 2 phương pháp là *Agglomerative* và *Divisive*.

Phương pháp	Hướng giải quyết	Ưu điểm	Nhược điểm
Divisive	Là phương pháp tiếp cận từ trên xuống, bắt đầu từ một nhóm duy nhất chứa tất cả các đối tượng (nút gốc) của cây, sau đó tiếp tục phân tách hoặc chia nhỏ nút này thành các nhóm con nhỏ hơn để tạo ra các nút cấp thấp cho để khi tất cả các nút chỉ còn lại một đối tượng hoặc đáp ứng tiêu chí kết thúc đã cho.	Có thể xử lý dữ liệu bất thường: vì bắt đầu với một cụm duy nhất và sau đó chia nhỏ cụm này thành các cụm nhỏ hơn, cho phép các cụm bất thường được phát hiện và phân tách khỏi các cụm bình thường. Các quyết định phân chia được đưa ra dựa trên tất cả các kết quả, trong khi phương pháp tiếp cận từ dưới lên chỉ dựa trên các cụm lân cận.	Khó xác định điểm chia tách ở mỗi cụm.

Agglomerative	Là phương pháp tiếp cận từ dưới lên, bắt đầu với việc hợp nhất các nhóm nhỏ thành các nhóm lớn hơn cho đến khi tất cả các điểm dữ liệu được nhóm lại thành một nhóm duy nhất hoặc đáp ứng tiêu chí kết thúc đã cho.	Dễ triển khai: vì có thể tiếp tục hợp nhất các cụm có khoảng cách nhỏ nhất (trong mỗi lần lặp) thành các cụm lớn hơn. Có thể sử dụng nhiều cách tính khoảng cách khác nhau: Single Linkage, Complete Linkage, Average Linkage. [2]	Không thể hoàn tác các bước đã thực hiện, nếu trong các lần lặp trước đó, các đối tượng đã bị gán vào các cụm không chính xác, thì trong các lần lặp sau, việc này không thể được hoàn tác để điều chỉnh. Nói cách khác, một khi quá trình hợp nhất được thực hiện, sự quyết định đó sẽ được giữ nguyên Khó xác định số lượng cụm: vì phải phụ thuộc vào các tính khoảng cách được sử dụng. Độ phức tạp về thời gian là $O(n^3)$ [3] với n là số điểm dữ liệu.

Bảng 3. Hai phương pháp phân cụm phân cấp

3.2.2 Giới thiệu sơ bộ về thuật toán:

Thuật toán BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*) là một thuận toán phân cụm dữ liệu sử dụng chiến lược tiếp cận từ trên xuống, được thiết kế để xử lý các tập dữ liệu lớn mà không yêu cầu lưu trữ toàn bộ dữ liệu trong bộ nhớ. Thuật toán này được giới thiệu lần đầu năm 1996 ở bài báo khoa học có tên: "BIRCH: An Efficient Data Clustering Method for Very Large Database" [4] của nhóm tác giả: **Tian Zhang, Raghu Ramakrishnan và Miron Livny.**

Đặc trưng phân cụm [5] (Clustering Feature) và cây CF (CF Tree) là nền tảng cốt lõi của thuật toán BIRCH. Đặc trưng phân cụm (CF) của một cụm là một vector 3 chiều tóm tắt thông tin về các cụm con và được định nghĩa như sau:

CF = (n, LS, SS)

- n: Số lượng đối tượng trong cụm
- LS: Tổng các giá trị thuộc tính của các đối tượng trong cụm

$$LS = \sum_{i=1}^{n} x_i$$

- SS: Tổng bình phương các giá trị thuộc tính của các đối tượng trong cụm

$$SS = \sum_{i=1}^{n} x_i^2$$

⇒ Các bộ ba này được gọi là các đặc trưng của cụm CF và được lưu giữ trong một cây được gọi là cây CF (CF tree). Việc CF tóm tắt thông tin về một cụm chỉ bằng 3 giá trị thay vì lưu trữ tất cả các điểm dữ liệu giúp tăng hiệu quả hơn về mặt lưu trữ, ngoài ra nó vẫn chứa đủ những thông tin để tính toán các phép đo cần thiết để đưa ra các quyết định phân cụm trong BIRCH.

Ngoài ra, có thể tính được các thống kê khác của cụm như là:

- Xo (*Centroid*): Tâm của cụm:

$$X_0 = \frac{LS}{n}$$

- R (*Radius*): Bán kính của cụm là khoảng cách trung bình của các điểm dữ liệu đến trọng tâm.

$$R = \sqrt{\frac{\sum_{i=1}^{n} (x_i - X_0)^2}{n}} = \sqrt{\frac{n(SS) - 2LS^2 - n(LS)}{n^2}}$$

- D (*Diameter*): Đường kính của cụm là khoảng cách trung bình theo cặp giữa các điểm dữ liệu trong một cụm.

$$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} (x_i - x_j)^2}{n(n-1)}} = \sqrt{\frac{2n(SS) - 2LS^2}{n(n-1)}}$$

- Tổng của CF1, CF2 tuân theo định lý "Additivity Theorem":

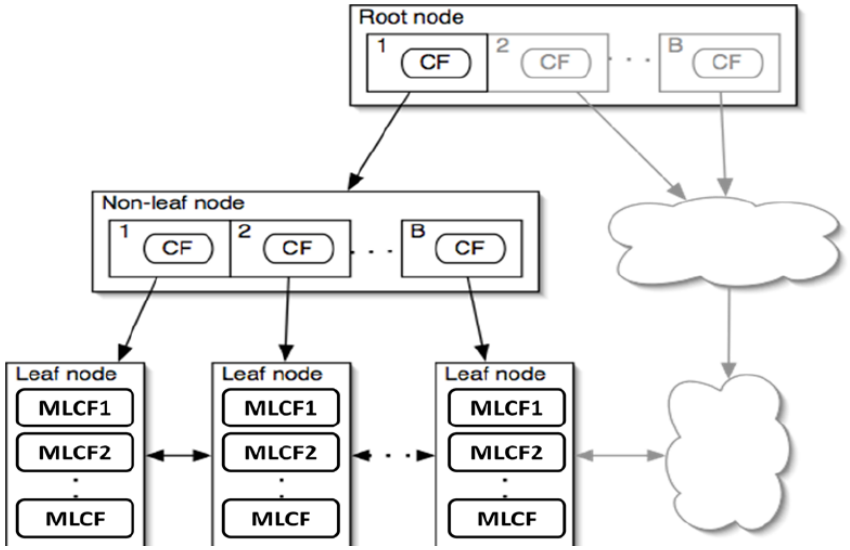
$$CF_1 + CF_2 = (N_1 + N_2, \vec{LS}_1 + \vec{LS}_2, SS_1 + SS_2)$$

- Khoảng cách giữa các cụm có thể đo bằng khoảng cách Euclidean, Manhattan,...

Thuật toán BIRCH hoạt động bằng cách xây dựng một cấu trúc được gọi là cây CF (Clustering Feature Tree). Cây CF là một cấu trúc dữ liệu được sử dụng để lưu trữ và truy cập các CF một cách hiệu quả. Cây CF là một cây cân bằng chiều cao và được đặc trưng bởi hai tham số:

- Ngưỡng T (*Threshold*): Tham số này xác định đường kính (hoặc bán kính) tối đa của các cụm con được lưu trữ ở các leaf node.
- Hệ số nhánh B (*Branching Factor*): Tham số này xác định số lượng con tối đa cho non-leaf node.

⇒ Mỗi leaf node biểu diễn một cụm được tạo thành từ tất cả các cụm con và tất cả các cụm con trong leaf node phải thỏa mãn yêu cầu với đường kính (hoặc bán kính) phải nhỏ hơn giá trị ngưỡng T. Nhờ tính chất này, cây CF biểu diễn rất tinh gọn tập dữ liệu vì mỗi mục con trong một leaf node không phải là một điểm dữ liệu đơn lẻ mà là một cụm con. Kích thước tổng thể của cây sẽ phụ thuộc vào hai tham số là ngưỡng T và hệ số nhánh B. T càng lớn thì cây càng nhỏ, B càng lớn thì cây càng ngắn nhưng tán lại rộng hơn.



Hình 1. Ảnh minh họa CF tree [4]

Kết luận: Các khái niệm về Đặc trưng phân cụm (Clustering Feature) và cây CF (Clustering Feature Tree) là những nền tảng của thuật toán BIRCH. CF cho phép lưu trữ và tính toán thông tin về các cụm một cách hiệu quả, trong khi cây CF giúp tổ chức và truy cập thông tin này một cách nhanh chóng.

3.2.3 Các điểm đặc trưng của thuật toán BIRCH [4]

a. Ưu điểm:

Thuật toán BIRCH có cách thức xây dựng sự phân cụm phù hợp với các bộ dữ liệu rất lớn bằng những ưu điểm có thể kể đến là:

Xử lý tập dữ liệu lớn: Một trong những đặc điểm quan trọng của BIRCH là khả năng xử lý tập dữ liệu lớn. Thuật toán này sử dụng một cấu trúc dữ liệu cây gọi là cây CF (Clustering Feature) để biểu diễn tập dữ liệu mà không yêu cầu lưu trữ toàn bộ dữ liệu trong bộ nhớ.

Tính cục bộ: BIRCH là một phương pháp phân cụm cục bộ, có thể đưa ra quyết định phân cụm mà không cần quét lại toàn bộ dữ liệu cũng như các cụm hiện tại. BIRCH dựa trên các hàm đo khoảng cách, chẳng hạn như khoảng cách Euclidean, để xác định mức độ gần của các điểm dữ liệu. Các yếu tố trên giúp tiết kiệm đáng kể về thời gian và bộ nhớ.

Tính linh hoạt và hiệu quả: BIRCH có khả năng phân cụm một cách linh hoạt và hiệu quả. Nó sử dụng hai tham số quan trọng: hệ số phân nhánh B và ngưỡng T, để cân bằng giữa kích thước của cây và độ chính xác phân cụm.

Giữ lại thông tin quan trọng: CF tree giúp giữ lại các đặc trưng quan trọng của tập dữ liệu. Điều này giúp giảm thiểu việc mất mát thông tin quan trọng trong quá trình phân cụm.

Độ phức tạp: BIRCH chỉ cần quét bộ dữ liệu một lần sẽ tạo ra được phân cụm tốt, và một hoặc nhiều lần bổ sung được áp dụng để cải thiện chất lượng hơn nữa. Nên độ phức tạp của nó là O(n) với n là số đối tượng dữ liệu.

b. **Nhược điểm:** [6]

Tuy nhiên, cũng có một số hạn chế của thuật toán BIRCH, bao gồm sự giảm chất lượng phân cụm so với một số thuật toán khác trong một số trường hợp, đặc biệt là khi dữ liệu có các cụm mang hình dạng và kích thước khác nhau.

⇒ **BIRCH có xu hướng phù hợp cho việc xử lý các tập dữ liệu lớn với nhiều dòng nhưng ít cột. Điều này là do BIRCH sử dụng cấu trúc cây CF (Clustering Feature), cấu trúc này giúp giữ lại thông tin quan trọng về tập dữ liệu mà không yêu cầu lưu trữ tất cả các điểm dữ liệu chi tiết. Và với đặc điểm những bộ dữ liệu như vậy, khi có số chiều của dữ liệu thấp, BIRCH có thể giúp giảm gánh nặng bộ nhớ và tăng hiệu suất, vì nó chỉ cần duyệt qua dữ liệu một lần mà không bị ảnh hưởng bởi việc quá nhiều chiều gây ra sự phức tạp và mất liên kết.**

3.3 Cấu trúc giải thuật :

Bước 1: Các điểm dữ liệu được chuyển thành CF (clustering feature), vector CF = (n. LS, SS)

Bước 2: Cây CF được khởi tạo để tập hợp lại các CFs ở bước 1.

Nhập ngưỡng B: số lượng điểm dữ liệu tối đa được phép thêm vào một cụm leaf node trước khi cụm đó được chia thành 2 cụm.

Nhập ngưỡng T: đường kính (bán kính) tối đa của cụm con được lưu trữ ở leaf node (ngưỡng được sử dụng để xác định liệu một điểm dữ liệu có nên được thêm vào một cụm hiện có hay tạo ra cụm mới)

Bước 3: Tính toán số lượng leaf L: số lượng nhánh tối đa của một leaf node

BIRCH tự động tính toán L dựa trên các tham số T, B

Bước 4: Tuần tự quét các điểm dữ liệu

Bước 5: So sánh centroid của điểm dữ liệu với các CF:

Với mỗi điểm dữ liệu, BIRCH sẽ so sánh centroid của điểm đó với từng CF tại nút gốc, sau đó gán điểm dữ liệu cho CF gần nhất

Bước 6: Di chuyển xuống child non-leaf node của CF node được chọn ở bước 5. Tiếp tục so sánh centroid của điểm dữ liệu đó với từng child non-leaf node l, và gán điểm dữ liệu cho CF gần nhất

Nếu CF đó vẫn còn child non-leaf thì tiếp tục thực hiện di chuyển và so sánh.

Bước 7. Kiểm tra và phân cụm:

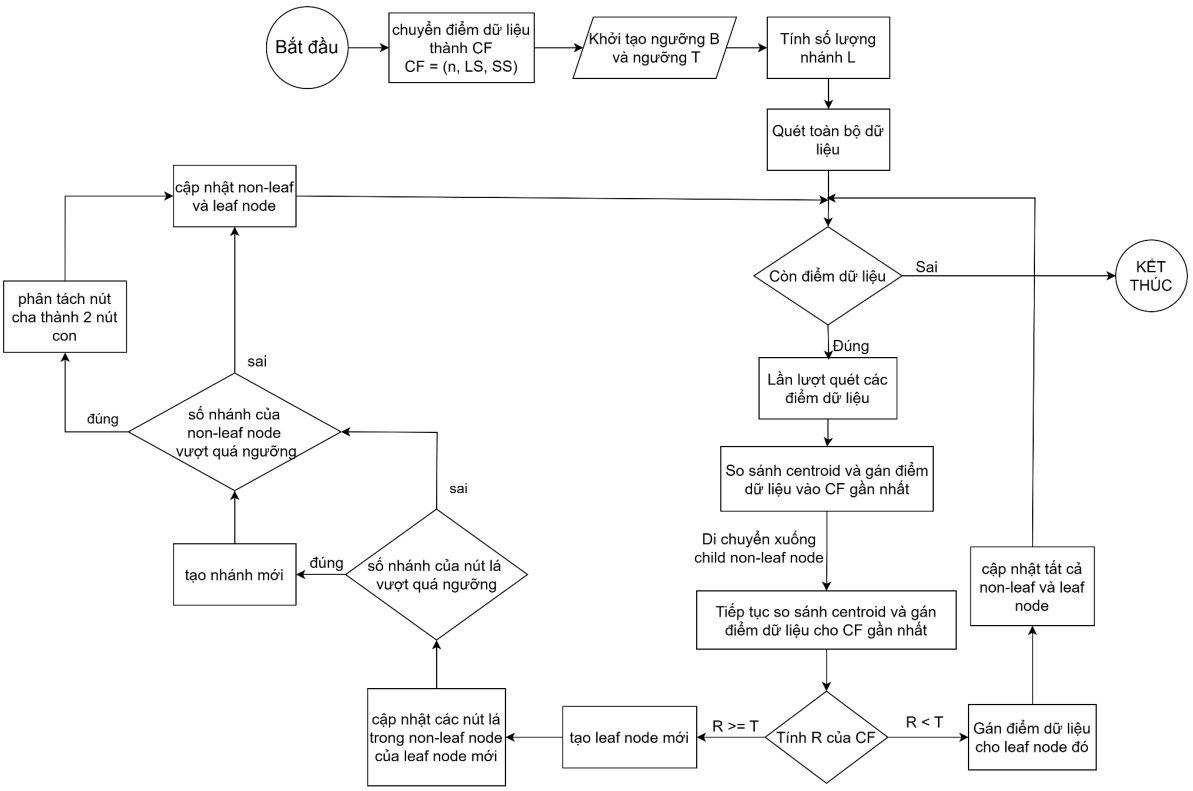
- Nếu bán kính R của leaf node được chọn có thể chứa điểm dữ liệu mới mà không vượt quá ngưỡng T, điểm dữ liệu sẽ được gán cho nút lá đó. Cả nút lá và các nút cha của nó sẽ được cập nhật để tính đến điểm dữ liệu mới.
    - Tính R của leaf node được chọn
    - R < T => gán điểm dữ liệu cho leaf node đó. Cập nhật lại leaf node và non-leaf node
- Nếu bán kính của nút lá vượt quá ngưỡng T khi thêm điểm dữ liệu mới, một nút lá mới sẽ được tạo ra chỉ chứa điểm dữ liệu mới đó. Các nút cha cũng sẽ được cập nhật.
    - R > T => tạo thêm nút lá mới
    - Cập nhật lại các leaf node và non-leaf node của nút lá mới tạo

Bước 8. Chia nhánh leaf node: Nếu số nhánh của một nút **lá** (m) vượt quá một ngưỡng nhất định, BIRCH sẽ tạo thêm nhánh mới (B) cho nút đó.

Ngược lại, bước 8 không được thực hiện

Bước 9. Tách và gộp lại CF: Nếu B vượt quá giá trị xác định, BIRCH sẽ tách nút cha của CF và sau đó gộp lại với CF mới được tạo ra ở mức cao hơn. Các nút cha cũng sẽ được cập nhật để tính đến điểm dữ liệu mới.

Nếu cây CF không còn thay đổi nữa, BIRCH kết thúc.



Hình 2. Sơ đồ giải thuật thuật toán BIRCH [7]

3.4 Ví dụ minh họa:

Xét tập dữ liệu một chiều với các điểm dữ liệu là {22; 9; 12; 15; 18; 27; 11} [8]. Giả sử giá trị tham số B (Branching Factor) là 2 và giá trị tham số T (Threshold) là 5. Nhóm tiến hành phân cụm như sau:

- Điểm dữ liệu đầu tiên là 22, nó được chèn trực tiếp vào một nút mới.

(1, 22, 484)	

CF = (n, LS, SS) = (1, 22, 22^2) = (1, 22, 484).
Tâm = 22 và D (đường kính) không xác định vì chỉ mới có một điểm dữ liệu.

- Điểm dữ liệu thứ 2 là 9, điểm dữ liệu này sẽ được chèn thử vào cụm hiện có.
  CF = (n, LS, SS) = (2, 22 + 9, $22^2 + 9^2$) = (2, 31, 565)
  Nhóm tiếp tục đi tính D để xem đường kính của cụm có thỏa mãn ngưỡng T (là 5) hay không.

$$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2}{n(n-1)}} = \sqrt{\frac{2n(SS) - 2LS^2}{n(n-1)}} = \sqrt{\frac{2*2(565) - 2(31)^2}{2*1}} = 13$$

Có thể thấy D = 13 > T = 5 nên điểm dữ liệu hiện tại {9} không thể chèn vào cụm hiện có và cần phải tạo một cụm mới.

(1, 22, 484)	(1, 9, 81)

Tâm: 22 và 9

- Điểm dữ liệu tiếp theo là 12, điểm dữ liệu này sẽ được chèn thử vào cụm có tâm gần nó nhất là (1, 9, 81) với tâm là 9.
  CF = (n, LS, SS) = (2, 9 + 12, $9^2 + 12^2$) = (2, 21, 225).
  Dựa vào công thức tính đường kính, nhóm tính được D = 3 < T = 5, vì vậy điểm dữ liệu hiện tại {12} được chèn vào cụm hiện có và CF được cập nhật.

(1, 22, 484)	(2, 21, 225)

Tâm: 22 (cụm con đầu tiên) và $\frac{Ls}{n} = \frac{21}{2}$ = 10.5 (cụm con thứ hai)

- Điểm dữ liệu tiếp theo là 15, điểm dữ liệu này sẽ được thử chèn vào cụm có tâm gần nó nhất là (2, 21, 225) với tâm là 10.5
  CF = (n, LS, SS) = (3, 21 + 15, 225 + 15^2) = (3, 36, 450)
  Dựa vào công thức tính đường kính, nhóm tính được D = 4.24 < T = 5, vì vậy điểm dữ liệu hiện tại {15} được chèn vào cụm hiện có và CF được cập nhật

(1, 22, 484)	(3, 36, 450)

Tâm: 22 (cụm con đầu tiên) và $\frac{Ls}{n} = \frac{36}{3}$ = 12 (cụm con thứ hai)

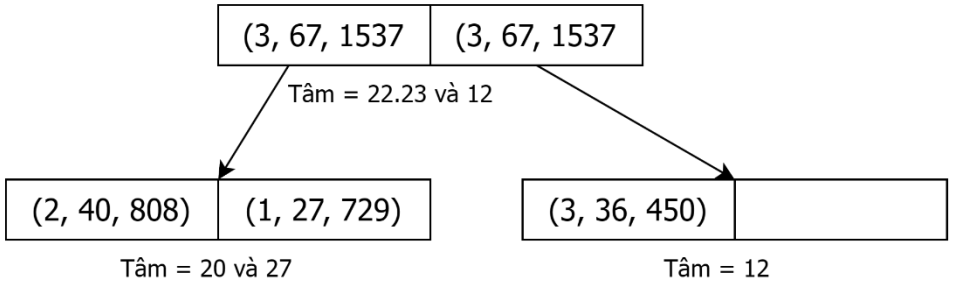
- Điểm dữ liệu tiếp theo là 18, điểm dữ liệu này sẽ được thử chèn vào cụm có tâm gần nhất của nó là (1, 22, 484) với tâm là 22
  CF = (n, LS, SS) = (2, 22 + 18, $22^2 + 18^2$) = (2, 40, 808)
  Dựa vào công thức tính đường kính, nhóm tính được D = 4 < T = 5, vì vậy điểm dữ liệu hiện tại {18} được chèn vào cụm hiện có và CF được cập nhật

(2, 40, 808)	(3, 36, 450)

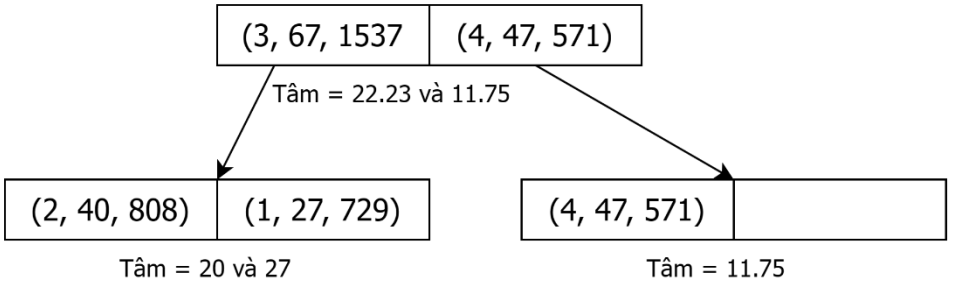
Tâm: LS/n = 40/2 = 20 (cụm con đầu tiên) và $\frac{Ls}{n} = \frac{36}{3}$ = 12 (cụm con thứ hai)

- Điểm dữ liệu tiếp theo là 27, điểm dữ liệu này sẽ được chèn vào cụm có tâm gần nhất của nó là (2, 40, 808) với tâm là 20.
  CF = (n, LS, SS) = (3, 40 + 27, 808 + 27^2) = (3, 67, 1537)
  Dựa vào công thức tính đường kính, nhóm tính được D = 6.37 > T = 5, vì vậy điểm dữ liệu hiện tại {27} không thể chèn vào cụm hiện có phải tạo một cụm mới, nhưng

leaf node không còn đủ chỗ vì nó chỉ có thể chứa 2 CF (Branching Factor = 2), vì vậy leaf node bị tách và một cấp mới được thêm vào cây CF.



- Điểm dữ liệu tiếp theo là 11, trước tiên sẽ so sánh với tâm ở gốc, có thể thấy điểm dữ liệu ở gần cụm (3, 36, 450) với tâm là 12.
  CF = (n, LS, SS) = (4, 36 + 11, 450 + 11^2) = (4, 47, 571)
  Dựa vào công thức tính đường kính, nhóm tính được D = 3.54 < T = 5, vì vậy điểm dữ liệu hiện tại {11} được chèn vào cụm hiện có và CF được cập nhật



Cụm	**Tâm**	**Điểm dữ liệu**
C1	20	{22, 18}
C2	27	{27}
C3	11.75	{9, 12, 15, 11}

Nhận xét: Các điểm dữ liệu có đặc điểm gần giống nhau sẽ được nhóm lại trong cùng một cụm, tạo ra cấu trúc phân cụm tự nhiên của tập dữ liệu. Qua ví dụ trên có thể thấy quá trình phân cụm đã tạo ra được trực tiếp 3 cụm là 3 leaf node, mỗi leaf node chứa các thông tin về các điểm dữ liệu thuộc về cụm tương ứng bao gồm số lượng điểm dữ liệu, tổng các giá trị thuộc tính của các điểm dữ liệu trong cụm, tổng bình phương các giá trị thuộc tính của các điểm dữ liệu trong cụm từ đó có thể tính được tâm và đường kính của cụm.

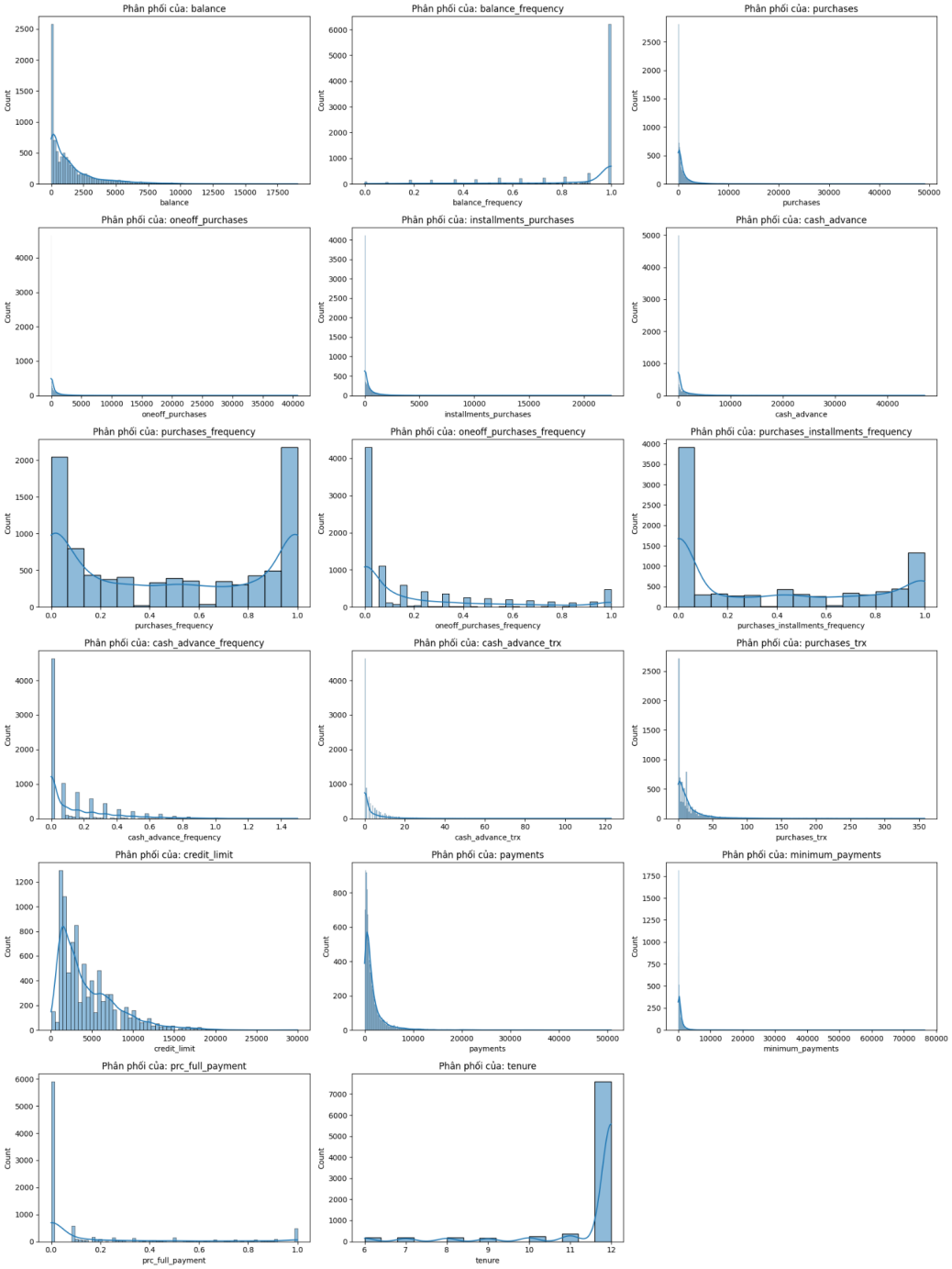
3.5 Hướng áp dụng vào đề tài:

Dựa trên các kiến thức được học hỏi, tìm hiểu trong quá trình học và thực hiện đồ án của môn Máy học, nhóm nhận thấy thuật toán phân cụm phân cấp BIRCH là một lựa chọn phù hợp và hiệu quả cho việc phân cụm dữ liệu trong bộ dữ liệu "Customer Dataset". Mục tiêu của nhóm là gom cụm các khách hàng các nhóm khách hàng có thói quen sử dụng thẻ tín dụng và hạn mức sử dụng tương tự nhau vào các nhóm, từ đó có thể giúp các doanh nghiệp có thể ứng dụng, đề xuất, xây dựng các chiến lược tiếp thị phù hợp với từng nhóm nhằm nâng cao trải nghiệm của khách hàng, đưa ra các chính sách hợp lý từ đó giúp nâng cao doanh thu.

Chương IV: Tiền xử lý dữ liệu

4.1 Phân tích dữ liệu được thu thập:

Vì bộ dữ liệu có kích thước khá lớn nên sẽ gặp đôi chút khó khăn trong EDA. Nhưng ta vẫn sẽ thực hiện một số phân tích với các biến để rút ra một số đặc trưng phục vụ cho quá trình phân cụm. Đầu tiên ta sẽ xem xét phân phối của các biến có mặt trong bộ dữ liệu này.

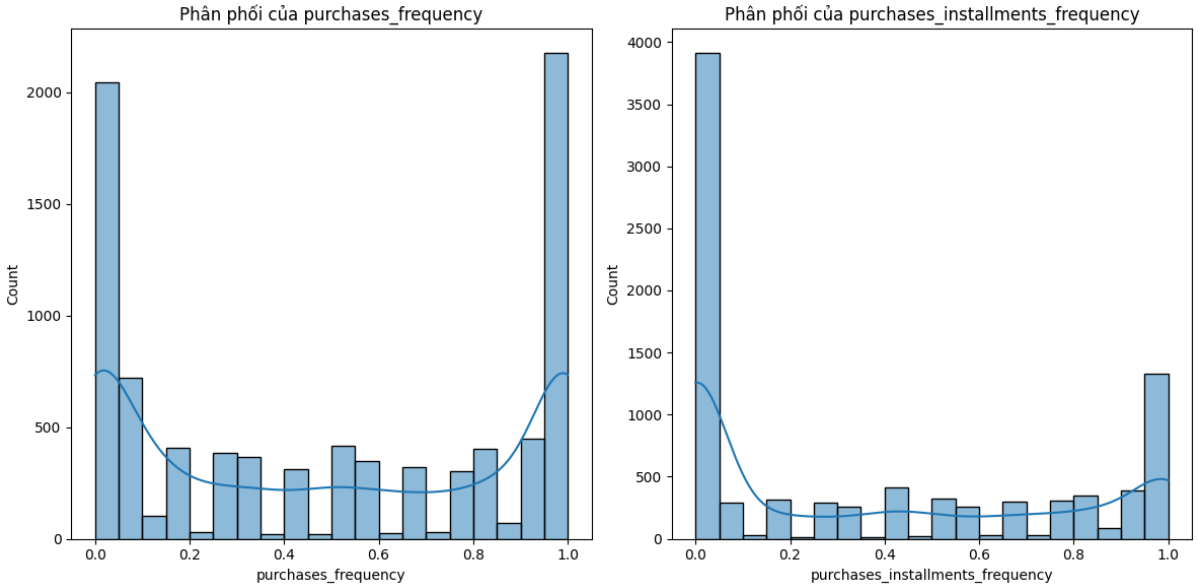


Hình 3. Biểu đồ phân phối của các biến

Tổng quan về phân phối của bộ dữ liệu, ta nhận thấy rằng có sự lệch lớn, cho thấy độ biến động cao. Dựa vào đặc điểm này, dự đoán là bộ dữ liệu sẽ chứa nhiều giá trị ngoại lai

(outliers), điều này sẽ đặt ra thách thức trong quá trình xử lý dữ liệu. Việc xử lý outliers là một bước quan trọng trong tiền xử lý dữ liệu để đảm bảo tính đồng nhất và chất lượng của dữ liệu.

Nhóm của chúng em đã nhận thấy một số biến có thể có ảnh hưởng đáng kể đến quá trình phân cụm. Trong số đó, "purchases_frequency" và "purchases_installments_frequency" là hai biến nổi bật. Đối với cả hai biến này, chúng ta có thể quan sát một đặc điểm đáng chú ý - có sự xuất hiện của hai đỉnh dữ liệu tại vị trí gần hai đầu mút của phân phối. Mỗi đỉnh dữ liệu có thể được coi là một cụm đặc trưng trong bộ dữ liệu.

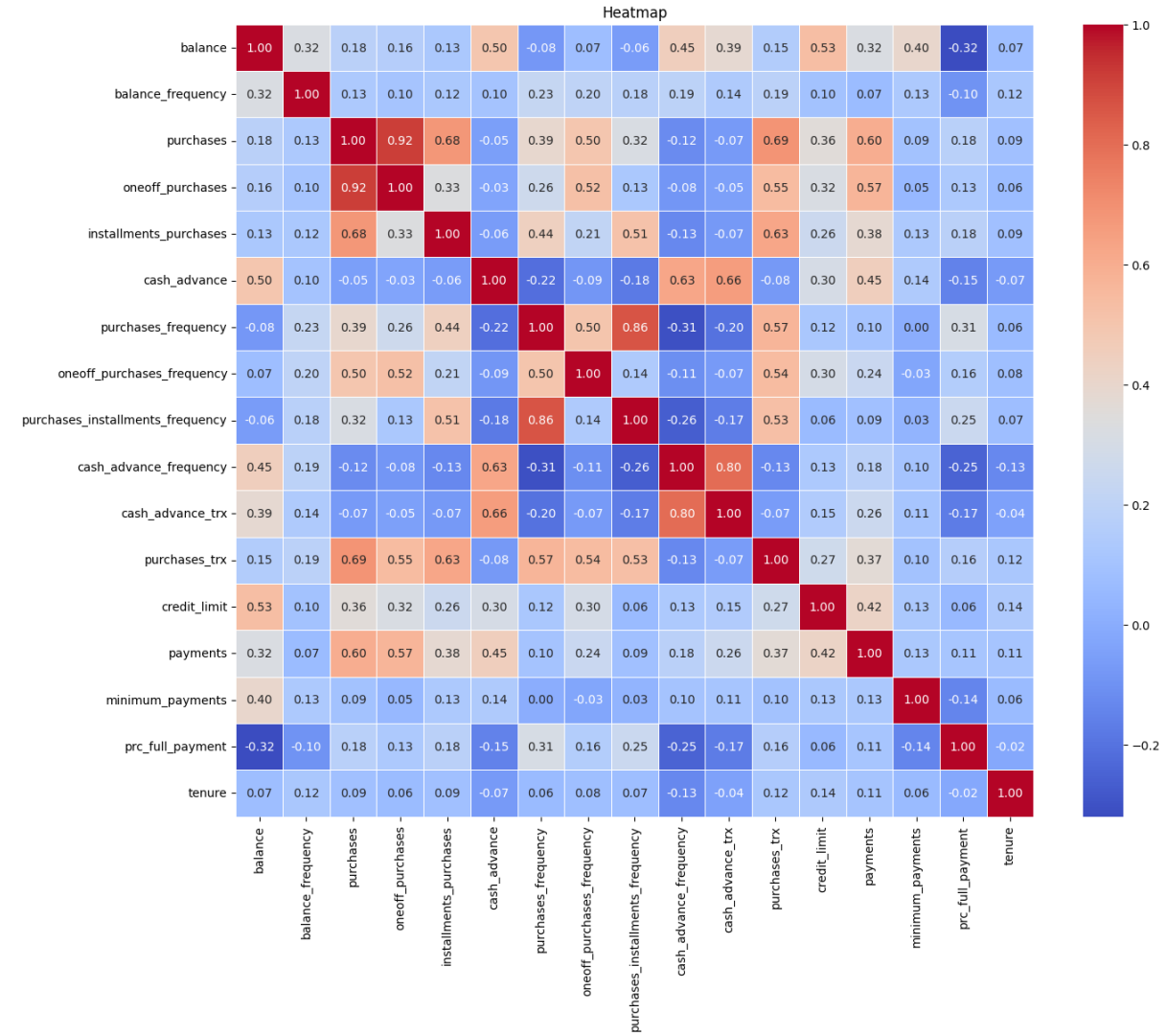


Hình 4. Biểu đồ phân phối của biến purchases_frequency và purchases_installments_frequency

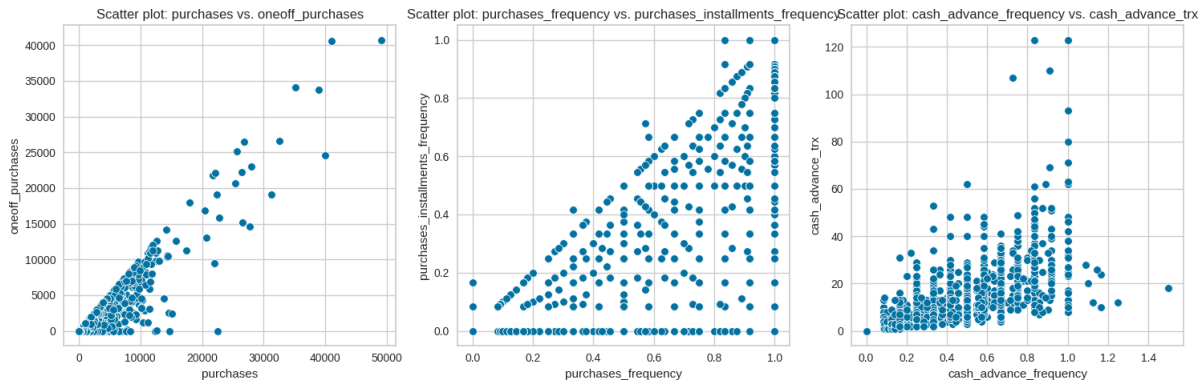
Điều này gợi ý rằng có thể có hai nhóm chính trong dữ liệu của chúng ta, và việc phân cụm có thể được thực hiện dựa trên sự tương đồng hoặc sự khác biệt giữa các đỉnh dữ liệu này. Điều này mang lại cơ hội để tập trung nghiên cứu sâu hơn vào những khía cạnh quan trọng của dữ liệu và xác định cụm đặc trưng một cách rõ ràng, có thể giúp chúng ta hiểu rõ hơn về bản chất và đặc điểm của từng nhóm.

Tiếp theo, dựa vào heatmap nhóm nhận thấy có một vài cặp biến có độ tương quan cao như:

- “purchases_installments_frequency” với “purchases_frequency”
- “purchases” với “oneoff_purchases”
- “cash_advance_frequency” với “cash_advance_trx”



Hình 5. Heatmap thể hiện tương quan của các biến



Hình 6. Scatter plot thể hiện tương quan giữa các cặp biến

Do các cặp biến này có tương quan quá cao, nhóm nhận thấy có thể là biểu hiện của đa cộng tuyến. Để không làm ảnh hưởng đến kết quả phân cụm, nhóm sẽ kiểm tra đa cộng tuyến bằng VIF

```
# Danh sách các cặp biến cần kiểm tra
variable_pairs = [('purchases', 'oneoff_purchases'),
                  ('purchases_frequency',
'purchases_installments_frequency'),
                  ('cash_advance_frequency', 'cash_advance_trx')]

# Tính toán VIF và hiển thị kết quả
for pair in variable_pairs:
    x = df[list(pair)]
    correlation_coefficient = x.corr().iloc[0, 1]
    vif = pd.DataFrame()
    vif["Variable"] = x.columns
    vif["VIF"] = [variance_inflation_factor(x.values, i) for i in
range(x.shape[1])]

    # Hiển thị kết quả
    print(f"\nPair: {pair[0]} vs {pair[1]}")
    print("Correlation coefficient:", correlation_coefficient)
    print("Variance Inflation Factor (VIF):")
    print(vif)
```

output:

```
Cặp: purchases vs oneoff_purchases
Correlation coefficient: 0.9168445587151489
Variance Inflation Factor (VIF):
           Variable      VIF
0         purchases  6.88149
1  oneoff_purchases  6.88149

Cặp: purchases_frequency vs purchases_installments_frequency
Correlation coefficient: 0.8629336372699634
Variance Inflation Factor (VIF):
                          Variable       VIF
0              purchases_frequency  7.001987
1  purchases_installments_frequency  7.001987

Cặp: cash_advance_frequency vs cash_advance_trx
Correlation coefficient: 0.7995607573993837
Variance Inflation Factor (VIF):
                 Variable       VIF
0  cash_advance_frequency  3.374459
1        cash_advance_trx  3.374459
```

Cả hai cặp 'purchases' vs 'oneoff_purchases' và 'purchases_frequency' vs 'purchases_installments_frequency' đều có giá trị VIF cao (khoảng 6.88 và 7), cho thấy mức độ đa cộng tuyến ở mức trung bình. Điều này làm tăng tăng khả năng sai sót trong quá trình phân cụm.

Tương quan cao và đa cộng tuyến có thể ảnh hưởng đến quá trình phân cụm bằng cách làm mất đi sự độc lập giữa các biến và tăng độ nhòa lẫn giữa các cụm. Kết quả có thể là các cụm không rõ ràng hoặc không thể phân biệt rõ ràng.

4.2 Tiền xử lý dữ liệu:

4.2.1 Xử lý các cột dữ liệu:

Thông qua phần đơn biến, chúng ta đã hiểu được rằng có một biến có tên "cust_id" chứa những mã nhận dạng của mỗi khách hàng. Tuy nhiên những giá trị này có vẻ như là không có ý nghĩa về mặt thống kê. Hãy kiểm tra điều này:

```
df['cust_id'].duplicated().sum()
```

```
0
```

```
df['cust_id'].nunique()
```

```
8950
```

Có thể thấy được rằng những giá trị "cust_id" chỉ xuất hiện một lần và không hề xuất hiện lại. Vậy nên những giá trị mà nó biểu diễn không có ý nghĩa gì ngoài liệt kê thứ tự các khách hàng và điều này thì không mang nhiều mục đích nghiên cứu. Vì thế về cột cust_id thì sẽ loại bỏ ra khỏi mô hình vì không mang ý nghĩa thống kê.

```
df = df.drop('cust_id', axis=1)
```

4.2.2 Xử lý missing data:

```
df.isnull().sum()
missing_data_summary=pd.DataFrame({'Missing Count':
df.isnull().sum(),
                                   'Missing Percentage':
df.isnull().mean()*100})
print(missing_data_summary)
```

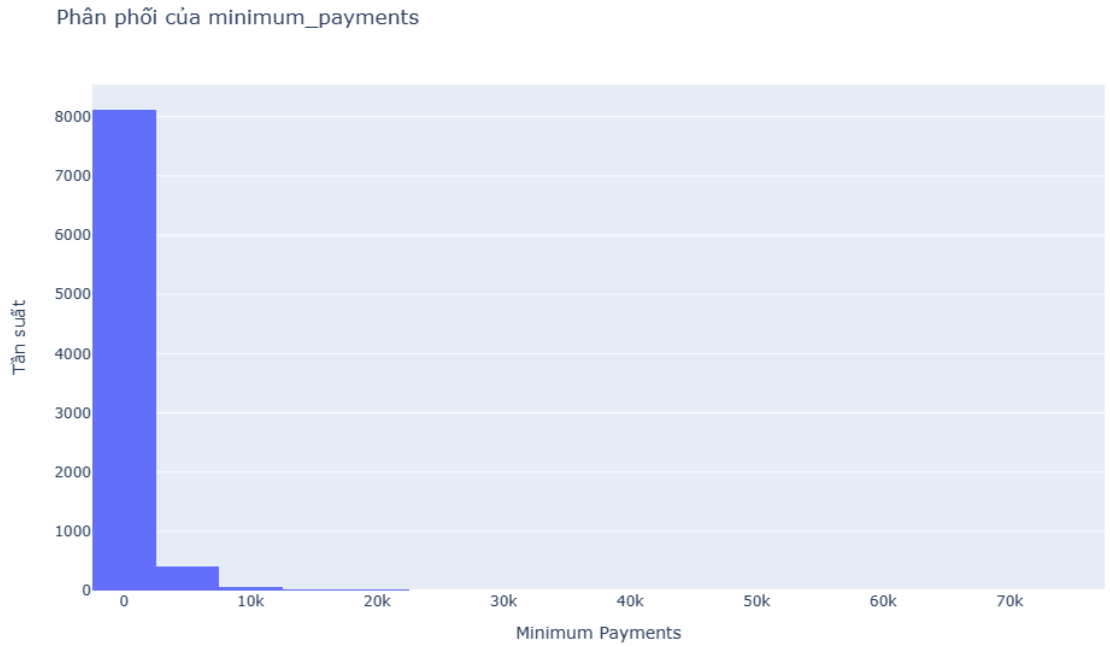
```
                                  Missing Count  Missing Percentage
cust_id                                       0            0.000000
balance                                       0            0.000000
balance_frequency                             0            0.000000
purchases                                     0            0.000000
oneoff_purchases                              0            0.000000
installments_purchases                        0            0.000000
cash_advance                                  0            0.000000
purchases_frequency                           0            0.000000
oneoff_purchases_frequency                    0            0.000000
```

purchases_installments_frequency	0	0.000000
cash_advance_frequency	0	0.000000
cash_advance_trx	0	0.000000
purchases_trx	0	0.000000
credit_limit	**1**	**0.011173**
payments	0	0.000000
minimum_payments	**313**	**3.497207**
prc_full_payment	0	0.000000
tenure	0	0.000000

Có thể thấy ở đây rằng xuất hiện 2 cột chứa missing values là "credit_limit" và "minimun_payments".
Với cột credit_limit thì với chỉ 1 trên tổng số 8950 dòng là có dữ liệu bị thiếu thì chúng ta sẽ quyết định loại bỏ dòng bị thiếu này ra khỏi bộ dữ liệu vì nó không đáng kể so với tổng thể.

```
# Credit_limit chỉ 1/8950 => Drop
df = df.drop(df[df.credit_limit.isna()].index, axis=0)
```

Với cột còn lại là "minimum_payments" thì lượng giữ liệu bị thiếu mặc dù không nhiều nhưng cũng đóng góp gần 3.5% dữ liệu của cột nên chúng ta sẽ tìm cách để thay thế những giá trị này. Đầu tiên hãy xem phân phối của nó.



Hình 7. Biểu đồ phân phối của minimum_payments

Có thể thấy đây là một phân phối có giá trị chủ yếu là 0 và kéo dài tới 70. Tuy nhiên, nhìn vào biểu đồ giá trị phân phối này không cho chúng ta đủ dữ kiện để thay thế missing values. Vậy hãy thử đánh giá với 3 cách để thay thế missing value quen thuộc cho dữ liệu dạng số như thế này là:

- Thay thế bằng giá trị median vì mean bị ảnh hưởng nhiều khi phân phối không đồng đều.

```
df1 = df.copy()
# Tính median của cột 'minimum_payments'
median_value = df1['minimum_payments'].median()

# Thay thế giá trị còn thiếu bằng median
df1['minimum_payments'].fillna(median_value, inplace=True)
```

- Dùng nội suy tuyến tính (interpolate)

```
df2 = df.copy()
# Dùng nội suy tuyến tính bằng interpolate
df2['minimum_payments'] = df2.minimum_payments.interpolate()
```

- Và thuật toán học máy và trong trường hợp này là KNN.

```
from sklearn.impute import KNNImputer
# Tạo mô hình
imputer = KNNImputer()

# Chia dữ liệu thành dữ liệu có và không có giá trị còn thiếu
data_with_values = df3.dropna(subset=['minimum_payments'])
data_missing_values = df3[df3['minimum_payments'].isnull()]

# Huấn luyện mô hình KNNImputer và dự đoán
imputer.fit(data_with_values.drop('minimum_payments', axis=1))
imputed_values =
imputer.transform(data_missing_values.drop('minimum_payments',
axis=1))
data_missing_values['minimum_payments'] = imputed_values

# Gộp dữ liệu đã điền giá trị còn thiếu vào DataFrame chính
df3.loc[df3['minimum_payments'].isnull(), 'minimum_payments'] =
data_missing_values['minimum_payments'].values
```

So sánh kết quả:

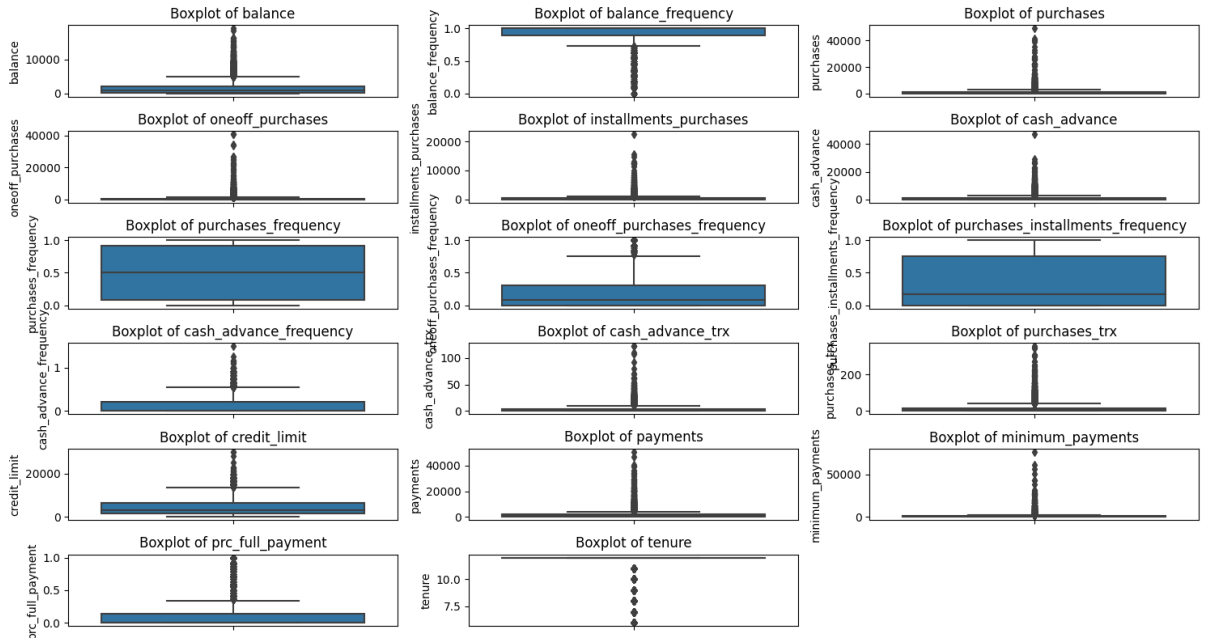
Hình 8. Trực quan kết quả các phương pháp xử lí missing values

- Qua các giá trị heatmap của từng cách thay thế missing values thì chúng ta thấy được những sự tương quan hầu như không bị ảnh hưởng sau khi xử lý. Điều này cho thấy được sự tích cực khi những giá trị mới được bổ sung không làm ảnh hưởng tới tổng quan mô hình. Tuy nhiên trong 3 cách đã sử dụng thì an toàn và hợp lý nhất vẫn là KNNImputer vì:
    - Những cách như median hay nội suy sẽ bị ảnh hưởng bởi sự phân phối không đồng đều và có nhiều giá trị cao vượt ngưỡng như đã thấy ở biểu đồ phân phối
    - Và KNNImputer sử dụng những cột còn lại để có thể bổ sung giá trị bị thiếu, điều này giúp những giá trị được điền sẽ có ý nghĩa hơn là việc chỉ sử dụng giá trị trong chỉ riêng cột "minimun_payments" như 2 phương pháp trên.

⇒ Vậy chúng ta sẽ sử dụng KNNImpute để bổ sung giá trị bị thiếu cho cột "minimun_payments"

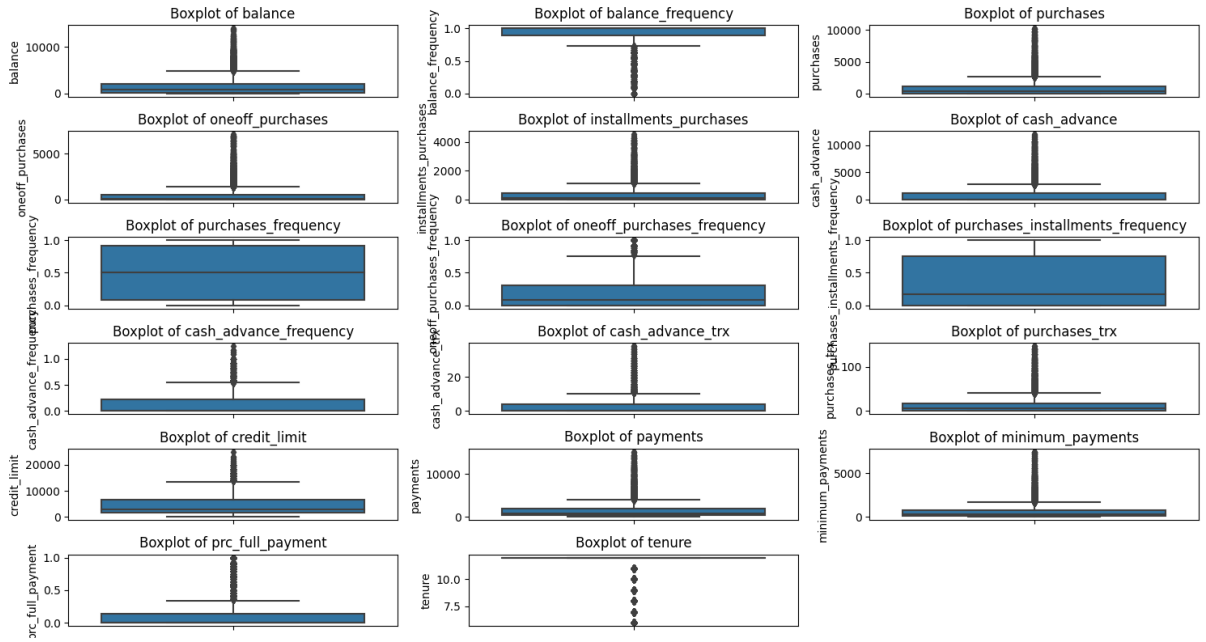
4.2.3 Xử lý Outliers:

Đầu tiên cùng xem qua tổng quan về các biến numerical:



Hình 9. Biểu đồ boxplot trực quan Outliers của các biến

Về mặt tương quan thì những giá trị ngoại lai có tồn tại ở hầu hết tất cả các cột trừ "purchases_freq" và "purchases_installments_freq". Ngoài ra có những cột đã được scale về trong khoảng {0; 1}, cũng như những giá trị khác trong các cột đều trong miền hợp lý, phù hợp với ngữ cảnh của bộ dữ liệu. Vậy nên chúng ta sẽ phải cân nhắc khi xử lý những giá trị quá thưa thớt và giữ lại những giá trị khác để đảm bảo sự nguyên vẹn cho bộ dữ liệu. Và với những biến không mang phân phối chuẩn như trên thì chúng ta sẽ chọn cách xử lý outliers bằng phương pháp tứ phân vị là hợp lý. Tuy nhiên sẽ có những điều chỉnh nhất định để bảo toàn sự trọn vẹn của bộ dữ liệu (Chỉ xóa đi những khách hàng có chỉ số quá tách biệt với nhóm còn lại).



Hình 10. Biểu đồ boxplot trực quan sau khi xử lý Outliers

Có thể thấy được rằng những giá trị tách biệt khỏi các biến đã được loại bỏ và dữ lại những dữ liệu cần thiết cho sự toàn vẹn của bộ dữ liệu.

Chương V: Xây dựng và đánh giá mô hình

5.1 Xây dựng mô hình:

5.1.1 Chuẩn hóa mô hình:

```
df.head()
```

	balance	balance_frequency	purchases	oneoff_purchases	installments_purchases	cash_advance	purchases_frequency	oneoff_purchases_frequency
0	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166667	0.000000
1	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000000	0.000000
2	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000000	1.000000
3	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083333	0.083333
4	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083333	0.083333

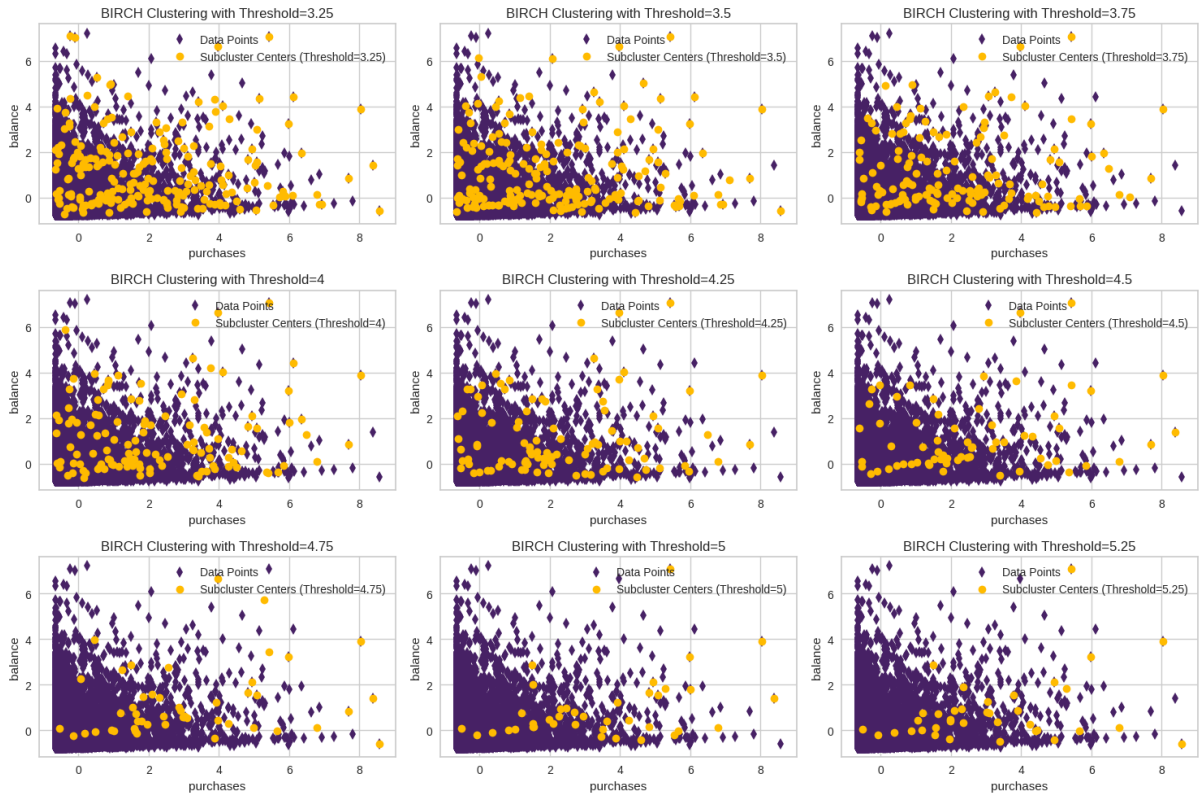
Với tổng quan các giá trị hiện hữu trong DataFrame thì các thuộc tính là tần suất hoặc có dữ liệu được gán nhãn sẽ không được chuẩn hóa cho nên nhóm sẽ chọn các thuộc tính định lượng cần thiết để chuẩn hóa.

```
from sklearn.preprocessing import StandardScaler

# # Assume df is your DataFrame
columns_to_scale = ['balance', 'purchases',
'oneoff_purchases','installments_purchases',
'cash_advance','cash_advance_trx', 'credit_limit', 'payments',
'minimum_payments', 'tenure']


scaler = StandardScaler()
df_scaled = df.copy()
df_scaled[columns_to_scale]=
scaler.fit_transform(df[columns_to_scale])
```

5.1.2 Chọn Threshold và Branching Factor tốt nhất (Cross-Validation):

```
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 10))
axes = axes.flatten()
threshold_values = [3.25, 3.5, 3.75, 4, 4.25, 4.5, 4.75, 5, 5.25]

for i, threshold_value in enumerate(threshold_values):
    brc = Birch(threshold=threshold_value)
    # Fit vào mô hình
    brc.fit(df_scaled)
```

```
    # Tạo ra tọa độ của các cụm con
    c = brc.subcluster_centers_

    # Biểu diễn dựa trên purchases và balance
      axes[i].plot(df_scaled['purchases'], df_scaled['balance'], 'd',
label='Data Points')
    axes[i].plot(c[:, df.columns.get_loc('purchases')],
                 c[:, df.columns.get_loc('balance')],
                                    'o', label=f'Subcluster Centers
(Threshold={threshold_value})')
                      axes[i].set_title(f'BIRCH Clustering with
Threshold={threshold_value}')
    axes[i].set_xlabel('purchases')
    axes[i].set_ylabel('balance')
    axes[i].legend()

plt.tight_layout()
plt.show()
```



Hình 11. Scatter thể hiện mức độ phân cụm với các chỉ số Threshold khác nhau [9]

Khi Threshold có giá trị càng nhỏ thì có càng nhiều các cụm được tạo ra, việc quá nhiều cụm được tạo ra có thể dẫn đến việc không lý giải được cấu trúc tổng quát hóa trên toàn cục của

mô hình, hay có thể được hiểu là overfitting. Ngược lại, khi có quá ít cụm được tạo ra thì sẽ dẫn đến các cụm sẽ không phản ánh được sự khác biệt giữa khác nhóm và cấu trúc cục bộ của dữ liệu sẽ bị mơ hồ, có thể hiểu là Underfitting. Nên việc chọn Threshold từ 3.25 đến 5.25 là các mức ngưỡng tương đối để tránh việc phân cụm bị Overfit hoặc Underfit.

Sau khi đã chọn được khoảng giá trị Threshold phù hợp cho bài toán thì nhóm đã tạo một đoạn code để có thể tìm ra được giá trị Threshold và Branching_factor có điểm Silhouette cao nhất để làm tham số cho thuật toán áp dụng vào bộ dữ liệu. Ở đây nhóm quyết định sẽ tìm Branching_factor trong khoảng từ {20,30,40,50} khi bộ dữ liệu cũng không quá nhiều dòng và các ngưỡng Braching_factor này sẽ đảm bảo cho cây có sự cân bằng và các cụm không bị phân phối quá lệch.

```
def find_best_birch_parameters(df, threshold_values,
branching_factors):
    best_result = {'threshold': None, 'branching_factor': None,
'silhouette_score': -1}

    for threshold in threshold_values:
        for branching_factor in branching_factors:

            # Phân cụm sử dụng BIRCH với giá trị threshold và
branching factor hiện tại
            birch_model = Birch(threshold=threshold,
branching_factor=branching_factor)
            df['cluster'] = birch_model.fit_predict(df_scaled)

            # Đánh giá hiệu suất sử dụng silhouette score
            silhouette_avg = silhouette_score(df_scaled,
df['cluster'])

            # Kiểm tra xem có phải là kết quả tốt nhất không
            if silhouette_avg > best_result['silhouette_score']:
                best_result['threshold'] = threshold
                best_result['branching_factor'] = branching_factor
                best_result['silhouette_score'] = silhouette_avg
    df.drop(columns='cluster',inplace=True)
    return best_result

# Sử dụng hàm để tìm giá trị tốt nhất cho cả threshold và branching
factor
threshold_values_to_test = np.linspace(3.25, 4.5, 6)
branching_factors_to_test = [20,30,40,50]  # Các giá trị branching
```

```
factor để thử nghiệm
best_parameters = find_best_birch_parameters(df_scaled,
threshold_values_to_test, branching_factors_to_test)

print("Best Parameters:")
print("Threshold:", best_parameters['threshold'])
print("Branching Factor:", best_parameters['branching_factor'])
print("Silhouette Score:", best_parameters['silhouette_score'])
```

Output:

```
Best Parameters:
Threshold: 4.0
Branching Factor: 50
Silhouette Score: 0.7633783702273668
```

- **Nhận xét:**

Sau khi đã tìm được khoảng Threshold và cho chạy với các ngưỡng Branching_factor như trên thì kết quả cho thấy được ở giá trị Threshold là 4.0 và Branching_factor là 50 thì cho ra được điểm số Silhouette Score tốt nhất. Và nhóm sẽ sử dụng 2 kết quả này để tiến hành phân cụm cho bộ dữ liệu.

5.1.3 Chọn số cụm tốt nhất:

Cơ sở lý thuyết:

Việc xác định số lượng K cụm là tốt nhất ở thuật toán BIRCH không thể dựa vào Elbow K method như ở thuật toán phân cụm K-means truyền thống vì:

- Kỹ thuật "Elbow Method" thường được sử dụng để chọn số lượng cụm K tối ưu bằng cách xác định điểm "khuỷu" trên đồ thị, nơi mà việc tăng thêm số lượng cụm không còn mang lại sự cải thiện đáng kể về tổng bình phương sai số trong cụm (Within-Cluster Sum of Squares - WCSS).
- Thuật toán BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) được thiết kế để làm việc với các tập dữ liệu rất lớn và phức tạp mà ở đó việc tính toán WCSS không hiệu quả về mặt tính toán và rất hao tốn bộ nhớ, hoặc thậm chí không khả thi. Cụ thể:
    - **Về độ phức tạp thức toán**: BIRCH không yêu cầu duyệt qua toàn bộ tập dữ liệu để tính toán khoảng cách giữa các điểm và tâm cụm, điều này làm giảm độ phức tạp tính toán so với K-Means, nơi phương pháp Elbow yêu cầu tính toán WCSS một cách rõ ràng.
    - **Phân cấp cụm**: BIRCH xây dựng một cấu trúc phân cấp thông qua CF Tree, điều này cho phép phân cụm mà không cần xác định số lượng cụm trước. Thuật toán sau đó có thể sử dụng các phương pháp như cắt bớt cây CF để tạo ra số lượng cụm cuối cùng dựa trên yêu cầu phân tích cụ thể.
    - **Độ nhạy với thứ tự dữ liệu**: BIRCH có thể nhạy cảm với thứ tự của dữ liệu do cách nó cập nhật cây CF. Điều này có nghĩa là các đặc điểm của dữ liệu có

thể ảnh hưởng đến chất lượng của các cụm được tạo ra, khiến việc áp dụng một phương pháp tĩnh như Elbow Method trở nên kém hiệu quả.

- **Đa dạng của cụm**: BIRCH cho phép sự đa dạng về kích thước và hình dạng của cụm, điều mà Elbow Method không tính đến. Elbow Method dựa trên giả định rằng tất cả các cụm có cùng mức độ quan trọng và đặc tính.

Trong việc đánh giá hiệu quả của quá trình phân cụm, việc sử dụng các chỉ số như Silhouette, Davies-Bouldin (Db) index, và Calinski-Harabasz (Ch) index là phổ biến.

1. **Silhouette Coefficient:**

Silhouette coefficient cung cấp một cái nhìn chi tiết về cách mỗi điểm dữ liệu được phân vào cụm. Điều này hữu ích để xác định liệu các điểm có được gom nhóm một cách hợp lý hay không. Giá trị silhouette cao cho thấy rằng các điểm trong cùng một cụm gần gũi nhau hơn so với các điểm trong cụm khác, đánh dấu sự phân chia rõ ràng giữa các cụm. Silhouette Coefficient cho phép so sánh chất lượng phân cụm giữa các lần chạy khác nhau của thuật toán hoặc giữa các thuật toán khác nhau.

Cách tính: Tính bằng cách sử dụng khoảng cách trung bình giữa mỗi điểm và tất cả các điểm khác trong cùng cụm (a), và khoảng cách trung bình tới các điểm trong cụm gần nhất (b). Giá trị Silhouette gần 1 chỉ ra rằng các điểm được phân loại chính xác vào cụm của chúng.

- $a(i)$: Khoảng cách trung bình từ điểm i đến các điểm trong cụm
- $b(i)$: Khoảng cách trung bình nhỏ nhất từ điểm i đến các điểm trong cụm khác, mà i không thuộc về
- Công thức:

$$s(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$$

2. **Davies-Bouldin Index (DB index)** ***(David L. Davies and Donald W. Bouldin, 1979)*****:**

DB index cung cấp một chỉ số về việc các cụm có được tạo ra với các điểm dữ liệu nhóm chặt chẽ bên trong mình và cách biệt rõ ràng với các cụm khác hay không. Chỉ số này phổ biến trong việc đánh giá nhanh chất lượng phân cụm. Davies-Bouldin Index Có thể sử dụng để xác định số lượng cụm lý tưởng bằng cách tìm giá trị DB index thấp nhất.

Cách tính: Dựa trên tỷ lệ giữa khoảng cách trung bình trong cùng một cụm và khoảng cách giữa các cụm. Giá trị thấp hơn của DB index chỉ ra chất lượng phân cụm cao hơn.

Đối với mỗi cụm C_i:

- $centroid_i$: Tâm của cụm C_i
- S_i: Khoảng cách trung bình của tất cả các điểm trong cụm C_i đến tâm cụm

Đối với mỗi cụm C_i và C_j:

- K: là số cụm
- M_{ij}: Khoảng cách giữa $centroid_i$ và $centroid_j$

Chỉ số Davies-Bouldin cho cụm C_i:

$$DB_i = \max_{j \neq i} \left(\frac{S_i + S_j}{M_{ij}} \right)$$

Chỉ số Davies-Bouldin chung cho tập dữ liệu:

$$DB = \frac{1}{k} \sum_{i=1}^{k} DB_i$$

3. **Calinski-Harabasz Index (CH index)** ***(Caliński Tadeusz and Jerzy Harabasz, 1974)*****:**

CH index giúp hiểu rõ liệu các cụm có được tách biệt tốt so với nhau hay không, một yếu tố quan trọng trong phân cụm hiệu quả. Ngoài việc đánh giá sự tách biệt, chỉ số này cũng giúp xác định liệu các điểm trong cùng một cụm có đồng nhất với nhau hay không. Có thể sử dụng Ch index để so sánh chất lượng của các mô hình phân cụm khác nhau hoặc để điều chỉnh các tham số của một mô hình.

Cách tính: Dựa vào tỉ số giữa tổng phương sai giữa các cụm (khoảng cách giữa các cụm) và tổng phương sai trong cùng cụm. Giá trị lớn của CH index thường chỉ ra chất lượng phân cụm tốt.

- *BSS (Between-Cluster Sum of Squares)*: Tổng phương sai giữa các cụm.
- *WSS (Within-Cluster Sum of Squares)*: Tổng phương sai trong cụm.
- *N*: Tổng số điểm dữ liệu.
- *k*: Số lượng cụm.

Chỉ số Calinski-Harabasz:

$$CH = \frac{BSS/(k-1)}{WSS/(N-k)}$$

* Tóm lại:

- Một chỉ số **DB index <u>nhỏ</u>** cho thấy rằng các cụm có sự tách biệt cao và độ nhóm chặt bên trong cụm tốt, tức là mỗi cụm đều có các điểm dữ liệu gần gũi với nhau và cách biệt rõ ràng so với các cụm khác. Điều này chỉ ra chất lượng phân cụm tốt.
- Một chỉ số **CH index <u>cao</u>** biểu thị một chất lượng phân cụm tốt, với sự phân biệt rõ ràng giữa các cụm và độ đồng nhất cao bên trong mỗi cụm. Chỉ số này càng cao, tỷ lệ giữa tổng biến thiên giữa các cụm và tổng biến thiên trong cùng một cụm càng lớn, điều này cho thấy sự hiệu quả trong việc phân chia dữ liệu thành các cụm rõ ràng.
- Một giá trị **Silhouette Coefficient <u>cao</u>** chỉ ra rằng các điểm dữ liệu được phân loại chính xác vào cụm của chúng, với mức độ gần gũi cao với các điểm khác trong cùng cụm và cách biệt rõ ràng so với các điểm trong cụm gần nhất. Một giá trị cao (gần 1) là lý tưởng, cho thấy chất lượng phân cụm tốt.

Kết hợp ba chỉ số này cung cấp một cái nhìn toàn diện về chất lượng của quá trình phân cụm, từ việc đánh giá sự đồng nhất bên trong từng cụm (Silhouette), đến việc đánh giá sự tách biệt và nhóm chặt của các cụm (DB index và CH index). Điều này rất quan trọng đối với BIRCH và các thuật toán phân cụm khác, nhất là khi làm việc với dữ liệu lớn và phức tạp.

<u>Áp dung:</u>

Khởi tạo ba danh sách silhouette_coef, db_index_list, ch_index_list dựa trên ba thang đo (Silhouette Coefficient, Davies Bouldin Index và Calinski Harabasz).

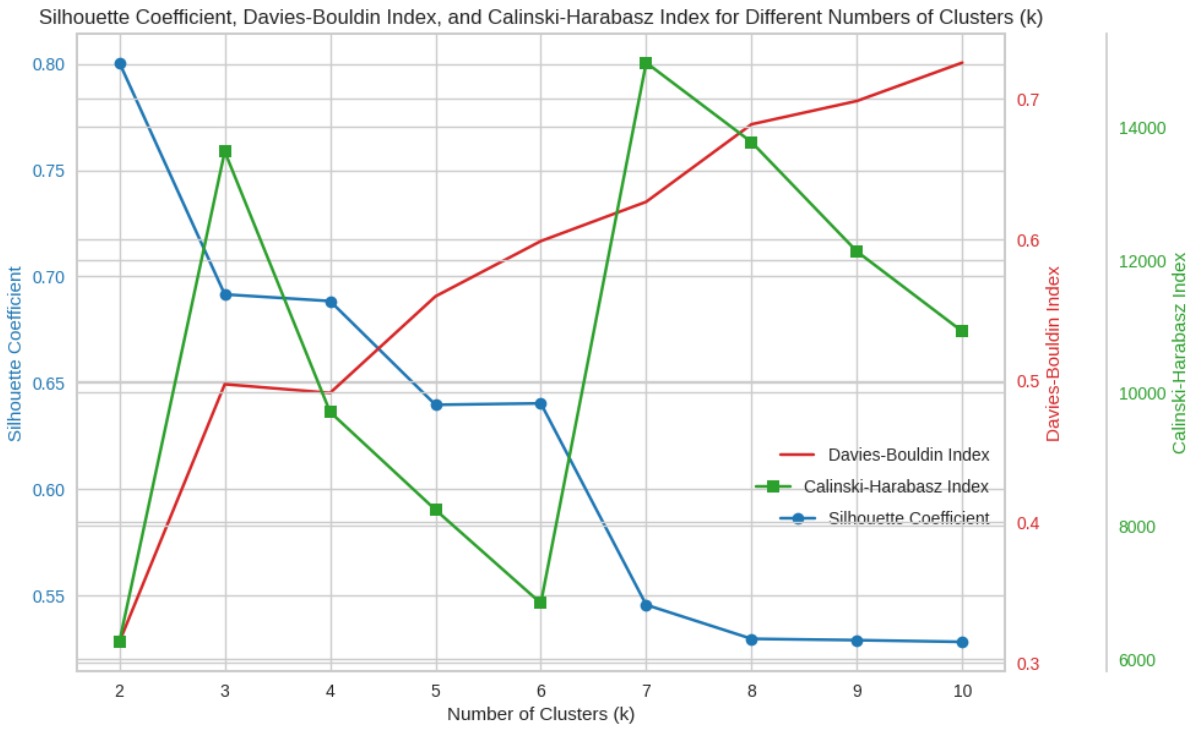
```
from sklearn.metrics import calinski_harabasz_score

silhouette_coef = []
db_index_list = []
ch_index_list = []

for k in range(2, 11):
    birch = Birch(n_clusters=k, threshold=4.0, branching_factor=50)
    birch.fit(df_scaled)

    # Silhouette Score
    silhouette_score_value = silhouette_score(df_scaled,
birch.labels_)
    silhouette_coef.append(silhouette_score_value)

    # Davies-Bouldin Index
    labels = birch.labels_
    db_index = davies_bouldin_score(df_scaled, labels)
    db_index_list.append(db_index)

    # Calinski-Harabasz Index
    calinski_index = calinski_harabasz_score(df_scaled, labels)
```

```
    ch_index_list.append(calinski_index)
```

Sau đó đánh giá dữ liệu đã được chuẩn hóa với K từ khoảng 2 đến 10 cụm và gán vào ba danh sách thang điểm đã đã khởi tạo.

Sau đó, nhóm sẽ thực hiện trực quan hóa sự phân bố các giá trị của các thang đo với số lượng K cụm tăng dần:

```
fig, ax1 = plt.subplots(figsize=(10, 6))

# Silhouette Score
ax1.set_xlabel('Number of Clusters (k)')
ax1.set_ylabel('Silhouette Score', color='tab:blue')
line1, = ax1.plot(range(2, 11), silhouette_coef, color='tab:blue',
marker='o', label='Silhouette Score')
ax1.tick_params(axis='y', labelcolor='tab:blue')
ax1.legend(loc='lower right',bbox_to_anchor=(1, 0.2))

# Davies-Bouldin Index
ax2 = ax1.twinx()
ax2.set_ylabel('Davies-Bouldin Index', color='tab:red')
line2, = ax2.plot(range(2, 11), db_index_list, color='tab:red',
marker='x', label='Davies-Bouldin Index')
ax2.tick_params(axis='y', labelcolor='tab:red')
ax2.legend(loc='lower right', bbox_to_anchor=(1, 0.3))

# Calinski-Harabasz Index
ax3 = ax1.twinx()
ax3.spines['right'].set_position(('outward', 60))
ax3.set_ylabel('Calinski-Harabasz Index', color='tab:green')
line3, = ax3.plot(range(2, 11), ch_index_list, color='tab:green',
marker='s', label='Calinski-Harabasz Index')
ax3.tick_params(axis='y', labelcolor='tab:green')
ax3.legend(loc='lower right',bbox_to_anchor=(1, 0.25))

fig.tight_layout()
plt.title('Silhouette Score, Davies-Bouldin Index, and
Calinski-Harabasz Index for Different Numbers of Clusters (k)')
plt.show()
```

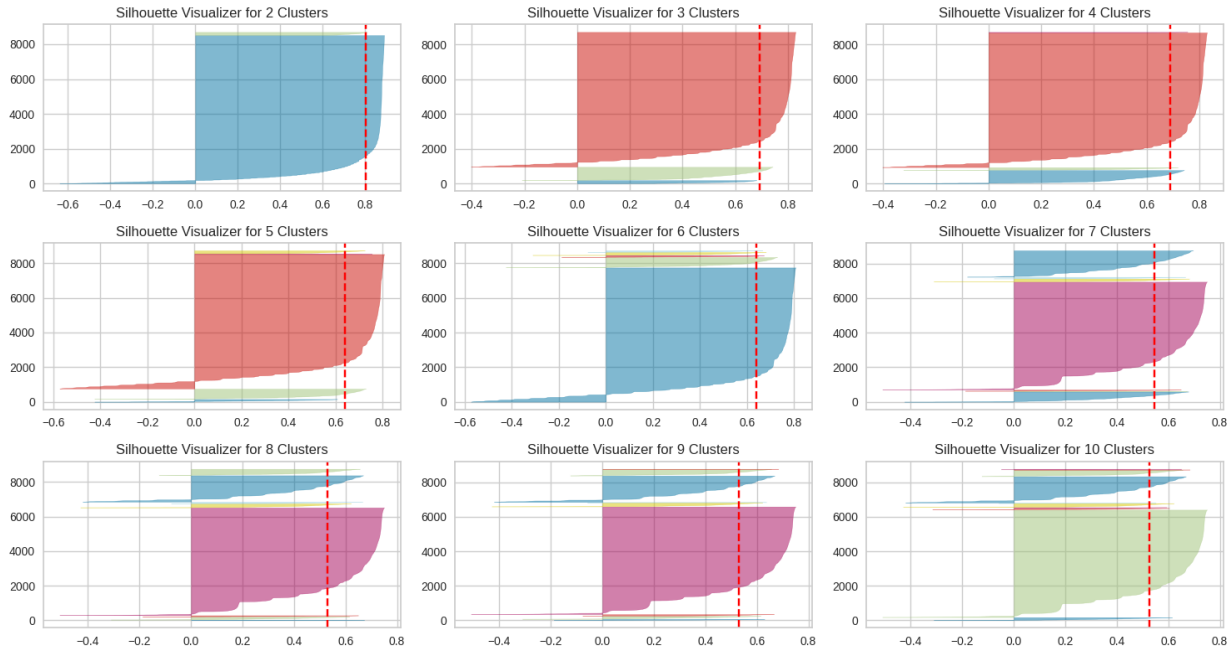


Hình 12. Biểu đồ trực quan các giá trị của các thang đo với số K tăng dần

Thông qua ba chỉ số ta thấy được số cụm K = 3 có thể được coi là số lượng cụm tốt nhất khi chỉ số CH index của nó cao thứ hai (sau K = 10), Silhouette coefficient cao thứ hai (sau SC = 2), DB index cũng là khá thấp so với các cụm còn lại.

Tiếp tục trực quan kết quả phân cụm theo Biểu đồ Silhouette:

```
cluster_range = range(2, 11)
fig, ax = plt.subplots(3, 3, figsize=(15, 8))
ax = ax.flatten()

for i, k in enumerate(cluster_range):
    ax[i].set_title(f'Silhouette Visualizer for {k} Clusters')
    visualizer = SilhouetteVisualizer(Birch(n_clusters=k,
threshold=4.0, branching_factor=50), colors='yellowbrick', ax=ax[i])
    visualizer.fit(df_scaled)
    visualizer.silhouette_score_ = silhouette_coef[i]

plt.tight_layout()
plt.show()
```



Hình 13. Biểu đồ Silhouette

- **Nhận xét:**

Qua hai biểu đồ để hiệu chỉnh số lượng các cụm thì ta thấy K = 3 là số cụm tốt nhất khi mà nó ổn định ở cả ba chỉ số đánh giá và các cụm cũng được biểu diễn tương đối dễ nhận biết ở biểu đồ Silhouette với điểm đạt đến xấp xỉ 0.7.

5.1.4 Đưa vào mô hình phân cụm:

Xây dựng mô hình phân cụm dựa trên bộ dữ liệu đã chuẩn hóa với các tham số tốt nhất đã được chọn (K = 3, threshold = 4.0 và branching_factor=50):

```
birch_model = Birch(n_clusters=3, threshold=4, branching_factor=50)
birch_labels = birch_model.fit_predict(df_scaled)
```

Tạo một Dataframe Copy từ dữ liệu gốc để gán các kết quả phân cụm vào, việc làm này sẽ đảm bảo cho dữ liệu gốc không bị ảnh hưởng và nhằm để biểu diễn trực tiếp kết quả phân cụm trên dữ liệu ban đầu:

```
df_c = df.copy()
df_c['cluster'] = birch_labels
```

Đếm tổng số lượng các giá trị dữ liệu ở từng nhóm:

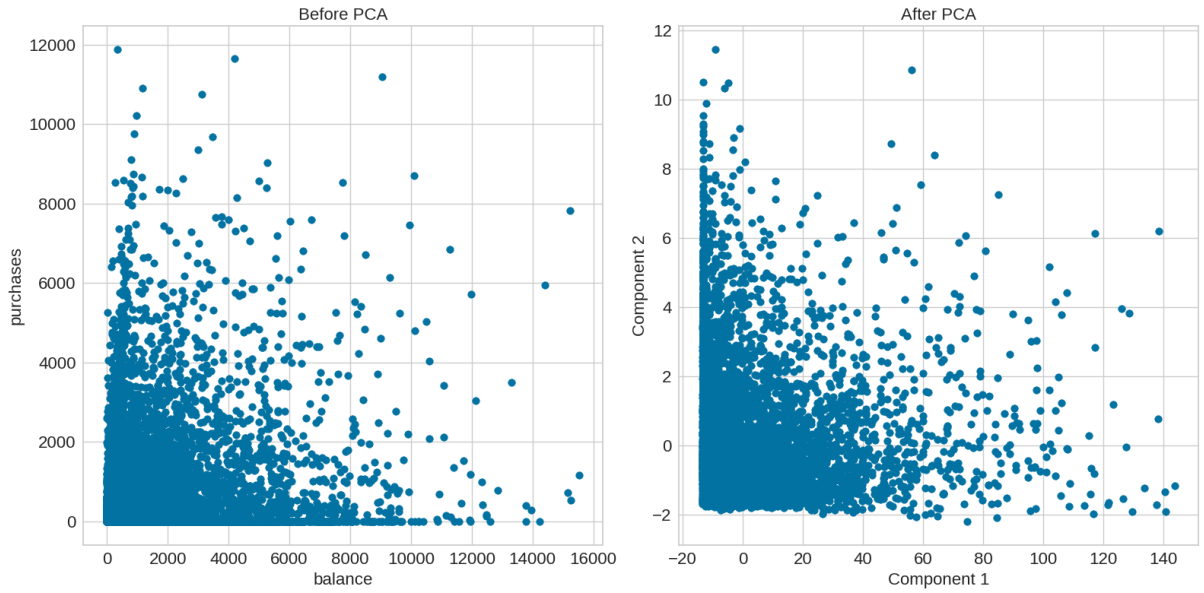
```
df_c['cluster'].value_counts()

OUTPUT:
2    7749
1     759
```

```
0      176
Name: cluster, dtype: int64
```

Có thể thấy ở cụm số 2 có nhiều dữ liệu được phân vào nhất với 7649 giá trị, tiếp đó là cụm số 1 với 759 giá trị và cụm 0 với ít giá trị quan sát được nhất là 176 giá trị.

5.1.5 Biểu diễn kết quả đạt được:

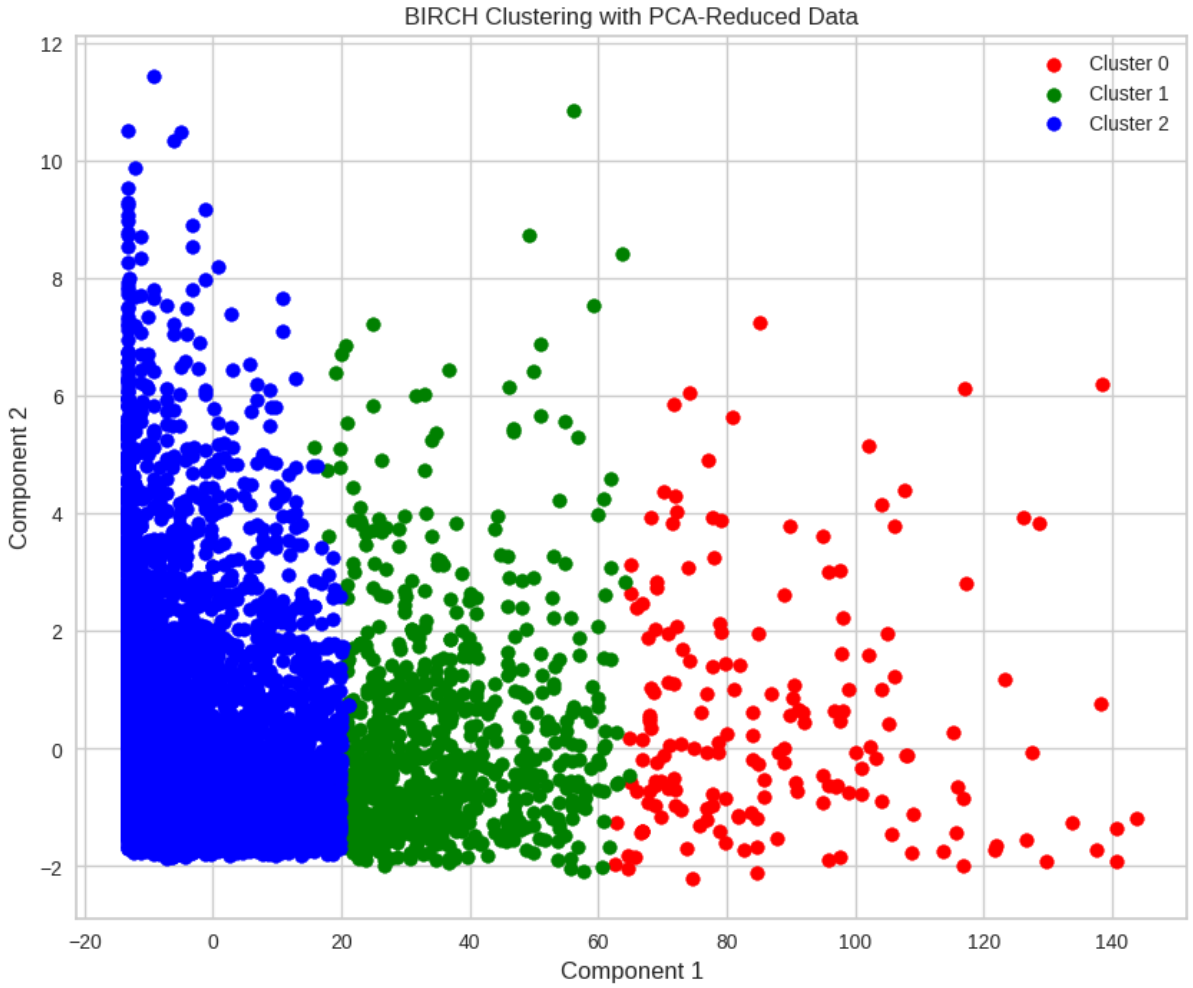
Để biểu diễn một cách tổng quan, nhóm quyết định sử dụng PCA để giảm chiều dữ liệu xuống còn 2 chiều. Ở đây, PCA (hay Phân tích thành phần chính) là một thuật toán học máy đơn giản giúp giảm chiều dữ liệu bằng cách xác định các thành phần chính mang nhiều ý nghĩa nhất, đó là các chiều có biến động lớn nhất trong không gian dữ liệu. Quá trình này giữ nguyên thông tin quan trọng, đồng thời thuận tiện hóa quá trình phân tích và hiểu dữ liệu.



Hình 14. Biểu đồ kết quả phân cụm trước và sau khi áp dụng PCA

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_scaled)

colors = ['red', 'green', 'blue']

# In ra các cụm
plt.figure(figsize=(10, 8))
for cluster in range(3):
    cluster_points = df_pca[birch_labels == cluster]
    plt.scatter(cluster_points[:, 0], cluster_points[:, 1],
color=colors[cluster], label=f'Cluster {cluster}')
```

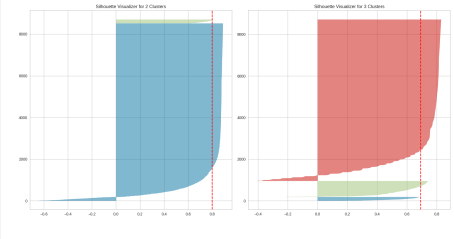
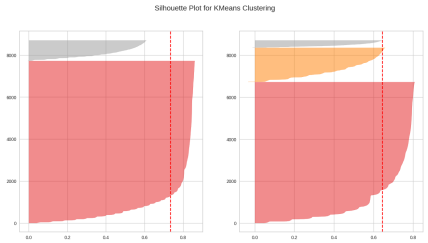
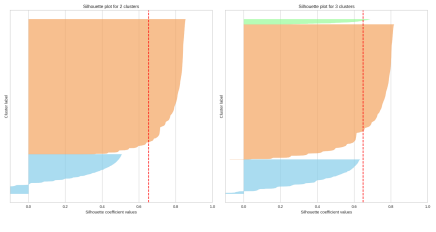
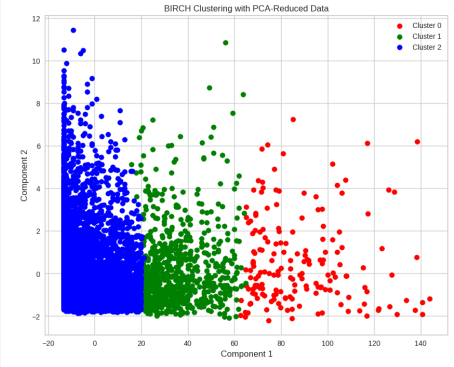
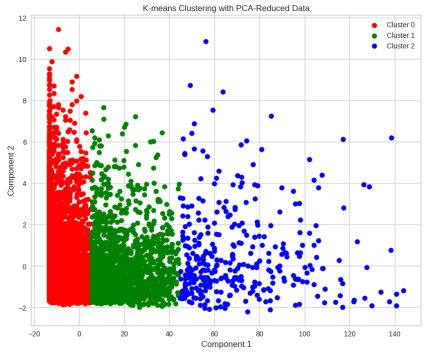
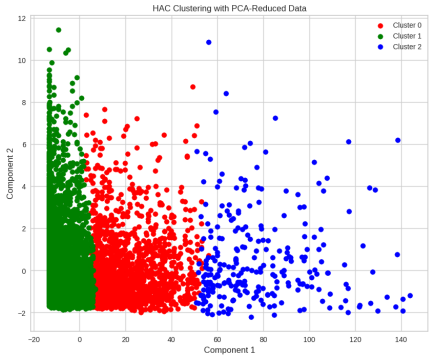
```
# In ra biểu đồ
plt.title('BIRCH Clustering with PCA-Reduced Data')
plt.legend()
plt.show()
```



Hình 15. Trực quan kết quả phân cụm

Có thể thấy rằng sau khi giảm chiều dữ liệu xuống còn 2 và biểu diễn, các cụm đã được phân chia khá tách biệt và trực quan. Mặc dù có một số điểm có sự phân chia chưa rõ ràng, và một số điểm nhiễu, nhưng trên tổng thể, đã có sự phân loại thành 3 nhóm riêng biệt.

❖ So sánh kết quả với 2 thuật toán HAC và K-Means

BIRCH	K-Means	HAC
		
		

```
Thời gian chạy BIRCH: 0.5237371921539307
Thời gian chạy HAC: 7.741003513336182
Thời gian chạy K-Means: 2.1560778617858887
```

Có thể thấy được rằng với những thuật toán khác nhau thì kết quả cho ra có phần khác biệt, nhận thấy điều này khi nhìn vào SilhouetteDiagram và kết quả phân cụm. Tuy nhiên về mặt điểm số Silhouette thì cũng không có quá nhiều sự khác biệt khi cả 3 đều đặt kết quả gần 0.65, là một kết quả cho thấy sự tách biệt giữa các cụm là rõ rệt.
Đây là kết quả có thể dự đoán được khi K-means thuộc nhóm **Centroid-based** *(phân cụm phân vùng)* ngược lại thì BIRCH và HAC thuộc nhóm **Hierarchical Clustering** *(phân cụm phân cấp)*. Tuy nhiên kết quả của HAC và BIRCH có sự khác biệt là do một thuật toán theo hướng bottom-up và một theo hướng top-down.

Và nhìn vào thời gian chạy của 3 thuật toán khi phân thành 3 cụm, ta thấy được ưu thế vượt trội của BIRCH khi có thời gian chạy ngắn nhất, sau đó là K-Means, cuối cùng là HAC. Từ đó đưa ra được những nhận xét sau:

- BIRCH đúng như cách nó được ra đời, xử lý rất tốt trên tập dữ liệu ít nhiều nhưng nhiều cột, mặc dù bộ dữ liệu này không lớn nhưng cũng cho thấy được ưu thế của BIRCH trong thời gian chạy thuật toán.
- K-Means có kết quả phân cụm khá ấn tượng về cả điểm silhouette và thời gian chạy. Tuy nhiên vẫn có phần chậm hơn BIRCH.

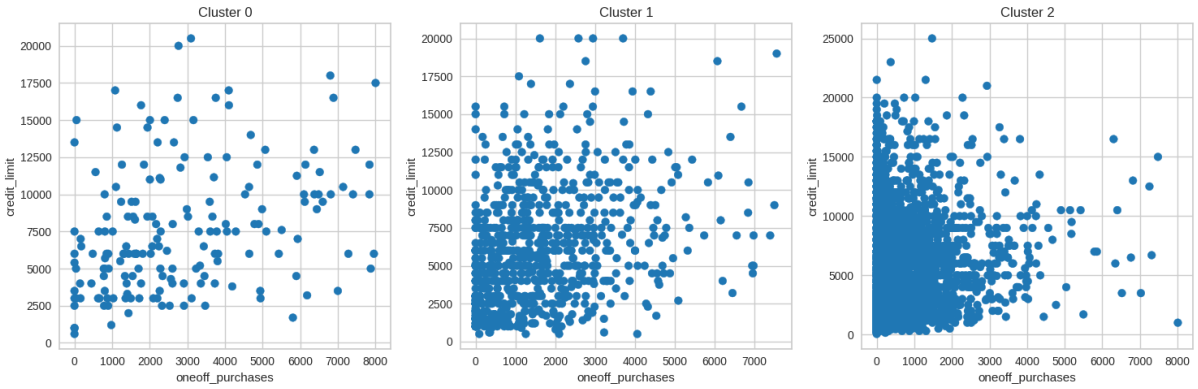
- HAC cung cấp một cái nhìn toàn cảnh về sự liên kết giữa các điểm dữ liệu, nhưng có thể tốn nhiều bộ nhớ cho dữ liệu lớn cũng vì vậy mà thường yêu cầu nhiều tài nguyên tính toán hơn do tính phức tạp của quá trình phân cụm phân cấp, đặc biệt là trên tập dữ liệu lớn. Và trong trường hợp này HAC cho thời gian chạy lâu nhất là điều dễ hiểu.

5.2 Đánh giá kết quả phân cụm:

5.2.1 Đánh giá:

Như đã đề cập trước đó, khi áp dụng phân cụm vào bộ dữ liệu này, nhóm mong muốn thăm dò một phần nào đó đặc điểm của những nhóm khách hàng đã được xác định, nhằm từ đó đưa ra những hành vi khách hàng của mỗi nhóm, giúp đưa ra các đề xuất chiến lược hoặc chính sách phù hợp với từng nhóm khách hàng. Tuy nhiên, vì mức độ hiểu biết về ngành thẻ tín dụng cũng như hành vi tiêu dùng có giới hạn, cùng với các kết quả chạy thuật toán chưa thực sự tối ưu, những nhận định từ phía nhóm có thể không chính xác và không mang tính quyết định như thông tin từ các chuyên gia trong lĩnh vực.
Do đó, phần này của đề án được phát triển để cung cấp cái nhìn sâu sắc hơn và tổng quan về kết quả của quá trình phân cụm. Mục tiêu là hiểu rõ hơn về các đặc điểm của từng nhóm khách hàng và từ đó cung cấp cơ sở cho việc đề xuất các chiến lược và chính sách mục tiêu, hỗ trợ thông tin trong việc đưa ra quyết định thông tin và chiến lược kinh doanh.

- Và ở đây nhóm đã thử với 2 biến là “credit_limit” và “oneoff_purchases”, mong muốn từ những cụm đã phân sẽ tìm ra được một đặc điểm nào đó của các nhóm khách hàng.



Hình 16. Trực quan kết quả phân cụm dựa trên 2 biến credit_limit và oneoff_purchases

- Nhận xét:

- Cụm 0: Qua biểu đồ thì cho thấy được những khách hàng ở nhóm này có sự phân bố thưa thớt và không có một đặc điểm gì nổi bật, sẽ cần quan sát thêm. Tuy nhiên có một điểm có thể nhận xét rằng đây là nhóm khách hàng có hạn mức tín dụng "credit_limit" từ 2500 trở lên và không ngại trong việc chi tiêu trong một lần trả "oneoff_purchases" khi rất ít giá trị ở ngưỡng 0.
- Cụm 1: Qua biểu đồ cho thấy được có một sự tuyến tính nhẹ ở đây, cho thấy được rằng những khách hàng ở nhóm này có số tiền chi tiêu trong một lần trả "oneoff_purchases" càng nhiều khi hạn mức tín dụng "credit_limit" càng cao. Đây là một điểm có thể lưu ý cho nhóm khách hàng này.
- Cụm 2: Nhóm khách hàng ở đây cho thấy được sự chi tiêu dè dặt hơn khi hạn mức tín dụng càng cao thì số tiền chi tiêu trong một lần lại có xu hướng giảm, và hầu như ít khi chi quá nhiều tiền trong một lần.

Column Name	Metrics	0	1	2
balance	mean	2811.395566	1863.774661	1407.951908
balance_frequency	mean	0.994835	0.975653	0.862860
purchases	mean	4796.540057	2802.559947	560.810430
oneoff_purchases	mean	2874.765568	1637.384717	321.307991
installments_purchases	mean	1921.774489	1165.768063	239.710810
cash_advance	mean	699.411786	623.406357	932.539596
purchases_frequency	mean	0.988163	0.957627	0.428560
oneoff_purchases_frequency	mean	0.730384	0.560178	0.150246
purchases_installments_frequency	mean	0.912067	0.813114	0.302432
cash_advance_frequency	mean	0.094697	0.092949	0.137764
cash_advance_trx	mean	2.789773	2.388669	3.080268
purchases_trx	mean	100.801136	48.002635	7.849271
credit_limit	mean	7689.204545	6137.417655	4106.294765
payments	mean	4357.312037	2788.870936	1287.183132
minimum_payments	mean	1166.227538	862.377309	624.858320
prc_full_payment	mean	0.253904	0.240897	0.141848
tenure	mean	11.971591	11.880105	11.462898

Hình 17. Bảng giá trị trung bình của các cụm trên từng biến [10]

Đối chiếu với bảng giá trị trung bình của các cụm cho thấy:

- Cụm 0: Có hạn mức tín dụng "credit_limit" lớn và đứng đầu trong 3 cụm, và cũng có lượng "oneoff_purchases" tương tự. Khá đúng với nhận định ở trên rằng đây là nhóm không ngại chi tiêu và khoản tiền trong một lần trả là lớn với 2874.
- Cụm 1: Có hạn mức tín dụng không thấp, và số tiền chi trả trong một lần cũng khá ấn tượng với 1637.
- Cụm 2: Như đã nói, đây là cụm chi tiêu khá ít khi chỉ có trung bình "oneoff_purchases" chỉ 321. Ít hơn trông thấy so với 2 cụm còn lại.

Ngoài ra, khi nhìn vào các chỉ số khác cho thấy:

- Cụm 0, 1: Đây là 2 nhóm khách hàng cho thấy mức độ thường xuyên sử dụng thẻ tín dụng khi trung bình các thông số: tần suất mua hàng "purchases_frequency", tần suất số dư tài khoản thay đổi "balance_frequency" hay số tiền các khoản vay trả góp

"installments_purchases" cao hơn hẳn so với cụm 2. Và vì thế họ cũng có hạn mức tín dụng "credit_limit" được ưu đãi cao hơn cụm còn lại.

- Cụm 2: Đây là nhóm khách hàng không dùng thẻ tín dụng thường xuyên, có thể thấy qua việc trung bình họ trả trước tiền mặt "cash_advance" và tần suất dùng tiền mặt "cash_advance_frequency" cao hơn 2 nhóm kia. Cũng như số tiền được thanh toán qua tài khoản "payments" và "purchases" thấp hơn hẳn nhóm còn lại. Tuy nhiên lượng tiền có trong tài khoản có nhóm này cũng không quá thấp so với cụm 1. Có vẻ họ chỉ sử dụng thẻ tín dụng cho việc cất giữ một khoản tiền và dùng khi cần thiết.

5.2.2 Hướng phát triển cho bộ dữ liệu:

Qua những nhận định trên thì nhóm có đưa ra một số chính sách dựa trên từng nhóm khách hàng như:

- Khách hàng ở cụm 0 là những khách hàng tiềm năng cần phải dữ chân và quan tâm tới. Khi những khách hàng trong cụm này rất thường xuyên sử dụng thẻ tín dụng, các khoản vay cũng như hạn mức tín dụng với các chỉ số tần suất cao nhất so với các cụm khác. Như kết quả phân cụm cho thấy được đây là nhóm khách hàng chiếm thiểu số vậy nên mở rộng thêm tệp khách hàng có mang những đặc điểm sử dụng thẻ tín dụng như vậy là một trong những điều cần quan tâm.
- Đối với cụm 1, đây chính là nhóm khách hàng có thể trở thành cụm 0 khi các xu hướng chỉ số cũng gần tương tự với nhóm trên. Tuy nhiên có thể yếu tố tài chính khiến họ không thể chi tiêu với số tiền và tần suất nhiều như cụm 0, nhưng đây vẫn là nhóm khách hàng có thể đem lại nhiều lợi ích tài chính khi chiếm số lượng tương đối và vẫn thường xuyên sử dụng thẻ để thanh toán.
- Ở cụm 2 thì với việc nhóm khách hàng có xu hướng ứng trước tiền mặt nên có thể đưa ra phương án cung cấp thẻ tín dụng đặc biệt với nhiều lợi ích khác nhau. Những lợi ích này có thể ở dạng ứng trước tiền mặt hoặc phí cho vay với lãi suất thấp, thời hạn sử dụng hoàn trả lâu, v.v. Ngoài ra, ngân hàng cũng có thể cung cấp các chương trình ngân hàng khác ngoài thẻ tín dụng, chẳng hạn như chương trình thanh toán sau với bên thứ ba, hợp tác hoặc các khoản vay cá nhân được cung cấp bởi các ngân hàng.

Vậy nhóm đã đưa ra được một vài sự phân tích cho những nhóm khách hàng đã được phân cụm sau khi sử dụng thuật toán BIRCH. Qua đây, nhóm thấy được BIRCH là một thuật toán phù hợp cho những bộ dữ liệu mang dạng số cũng như số chiều thấp, nhưng không bị ảnh hưởng nhiều bởi sự tăng trưởng dữ liệu. Qua bài đề án đã giúp nhóm hiểu và biết thêm về một thuật toán thú vị nhưng chưa được quá nhiều người biết tới nay.

TÀI LIỆU THAM KHẢO

[1] developers.google, "Clustering Algorithms".

[2] D. v. K.Ranjini, "Hierarchical Clustering Algorithm - A Comparative Study," 2011.

[3] A. H. v. S. S. Philipp Cimiano, "Comparing Conceptual, Divisive and Agglomerative Clustering for Learning Taxonomies from Text," 2004.

[4] R. R. v. M. L. Tian Zhang, "Birch: An Efficient Data Clustering Method for Very Large Database," 1996.

[5] A. D. F. v. J. Abreu, "A Data-Driven BIRCH Clustering Method for Extracting Typical Load Profiles for Big Data," 2018.

[6] W. L. v. Y. Jie, "Advanced Split BIRCH Algorithm in Reconfigurable Network," 2013.

[7] M. Z. S. S. Fanny Ramadhani, "Improve BIRCH algorithm for big data clustering," 2020.

[8] V. Dalal, "BIRCH Algorithm with working example," 2022.

[9] dmh126, "Birch algorithm does not cluster as expected".

[10] M. CAESAR, "Clustering for Effective Marketing Strategy".

[11] Giáo trình dạy học môn “Machine Learning” của đại học UEH, khoa Công nghệ Thông tin Kinh doanh, giảng viên “Nguyễn An Tế” 2023.