ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC KINH TẾ
KHOA THƯƠNG MẠI ĐIỆN TỬ
******

THUẬT TOÁN PHÂN CỤM: ỨNG DỤNG PHÂN CỤM PHÂN VÙNG VÀO PHÂN KHÚC KHÁCH HÀNG DỰA TRÊN MÔ HÌNH RFMTS TRONG CRM

Giảng viên hướng dẫn: Lê Diên Tuấn
Lớp: 46K29.2
Nhóm: 8
Tên Sinh Viên: Nguyễn Thanh Luân (0935270307)
Trần Thị Quỳnh Chi
Lê Trần Bảo Ngọc
Phan Thị Thảo Ngân

Nguyễn Thanh Luân	**100%**
Trần Thị Quỳnh Chi	**100%**
Lê Trần Bảo Ngọc	**100%**
Phan Thị Thảo Ngân	**100%**

Đà Nẵng, ngày 17 tháng 11 năm 2023

Contents

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

1. MỞ ĐẦU:

1.1 Tính cấp thiết của đề tài:

Ngày nay, vấn đề khan hiếm dữ liệu đã được thay thế bằng vấn đề dư thừa dữ liệu. Thách thức hiện tại là phải sử dụng hiệu quả dữ liệu trong quy trình quản lý quan hệ khách hàng (CRM) và phải chọn các kỹ thuật phân tích dữ liệu phù hợp. Có đa dạng các vấn đề trong CRM: phân khúc tệp khách hàng, marketing one - one, quản lý các khiếu nại và thiết lập sự hài lòng, phân tích giỏ hàng,... Nhưng cũng sẽ có đa dạng các kỹ thuật giúp giải quyết vấn đề tương ứng: phân cụm, phân loại, hồi quy, dự báo,... Hầu hết hiện nay, dữ liệu phi cấu trúc chiếm đại đa số vì thế mà việc nghiên cứu phương pháp học không giám sát cùng với kỹ thuật phân cụm trở nên tiềm năng. Không những vậy, nó còn có thể ứng dụng rộng rãi trong nhiều lĩnh vực.

1.2 Mục tiêu nghiên cứu:

Tìm hiểu các lý thuyết liên quan đến các kỹ thuật phân cụm, chỉ ra ưu nhược điểm của 2 kỹ thuật phổ biến trong lĩnh vực machine learning và khai phá dữ liệu là K-means và C-means. Bên cạnh đó bài nghiên cứu cũng phân tích các thuật toán, cách tiếp cận, giải thích bằng cách áp dụng vào 1 bài toán phân cụm thực tế, đánh giá hiệu quả và so sánh kết quả phân cụm được tạo bởi 2 kỹ thuật phân cụm nêu trên để từ đó đưa ra các kết luận.

2. CƠ SỞ LÝ THUYẾT:

2.1 Tìm hiểu về quản trị quan hệ khách hàng - CRM:

CRM, hoặc quản lý mối quan hệ khách hàng, là một hệ thống và chiến lược quan trọng trong doanh nghiệp. Nó tập trung vào xây dựng, duy trì và nâng cao mối quan hệ với khách hàng để tạo giá trị và tăng sự hài lòng của khách hàng. CRM giúp tổ chức quản lý thông tin khách hàng hiệu quả bằng cách tổ chức, lưu trữ và phân tích dữ liệu từ nhiều nguồn khác nhau. [1]

Thông qua việc thu thập và phân tích dữ liệu, CRM cung cấp cái nhìn toàn diện về khách hàng, giúp nhân viên tương tác với khách hàng một cách cá nhân hóa và đáp ứng nhu cầu của họ tốt nhất. CRM mang lại nhiều lợi ích cho doanh nghiệp bằng cách cải thiện hiểu biết về khách hàng, tăng cường tương tác và giao tiếp trong doanh nghiệp, và cung cấp các công cụ quản lý chiến dịch tiếp thị, bán hàng và dịch vụ khách hàng. Với sự phát triển của công nghệ và quy mô kinh doanh, CRM đã trở thành một công cụ quản lý không thể thiếu cho các doanh nghiệp hiện đại, giúp tạo ra giá trị bền vững và cạnh tranh trong thị trường ngày càng khốc liệt.

2.2 Phân loại khách hàng trong CRM dựa trên RFMTS

Một số mô hình RFM mới đã được phát triển để nâng cao hiệu quả so với mô hình RFM truyền thống. Chúng bổ sung các biến số để kiểm tra tính chính xác của kết quả. Một trong những mô hình đó là RFMTS, được phát triển từ mô hình RFM thông thường bằng cách tính thêm hai biến số bổ sung là T (Inter-purchase Time) và S (Satisfaction).

Cụ thể, RFMTS là viết tắt của:

- **Recency**: **là khoảng thời gian giữa giao dịch gần nhất tới hiện tại là bao lâu**? Là yếu tố quan trọng để đánh giá mức độ tương tác gần nhất của khách hàng với doanh nghiệp. Ví dụ, một khách hàng mua hàng hàng tháng sẽ có mức Frequency cao hơn so với một khách hàng chỉ mua hàng một lần mỗi năm. Khoảng thời gian này càng lớn thì khách hàng có khả năng cao rời bỏ công ty (churn customer - tỷ lệ rời bỏ), đồng nghĩa với việc doanh nghiệp sẽ mất càng nhiều thời gian để chăm sóc và thuyết phục họ quay lại. Nhưng ngược lại khoảng thời gian càng nhỏ thì khả năng tiếp cận sẽ càng cao. Frequency trong mô hình RFM được sử dụng để phân loại khách hàng thành các nhóm khác nhau như [2]:
- **Frequency**: Tần suất giao dịch của khách hàng với doanh nghiệp là bao nhiêu trong 1 khoảng thời gian nhất định, đánh giá mức độ tương tác lặp lại? Ví dụ, một khách hàng mua hàng hàng tháng sẽ có mức Frequency cao hơn so với một khách hàng chỉ mua hàng một lần mỗi năm. Khách hàng mua hàng càng thường xuyên sẽ càng có khả năng phản hồi nhanh với các chiến dịch của và trở thành khách hàng trung thành. Frequency trong mô hình RFM được sử dụng để phân loại khách hàng thành các nhóm khác nhau như:
- **Monetary**: đo giá trị tiền hoặc giá trị đơn hàng mà khách hàng đã chi tiêu trong mỗi lần mua hàng, thể hiện mức độ giá trị mà khách hàng mang lại cho doanh nghiệp. Tùy vào nhu cầu phân tích dữ liệu của doanh nghiệp cho từng chiến dịch marketing, giá trị tiền được đo bằng tổng số tiền mà

khách hàng đã chi trả hoặc tổng giá trị đơn hàng mỗi lần giao dịch. Monetary trong RFM Segmentation được sử dụng để phân loại khách hàng thành các nhóm khác nhau như:

- **Inter-purchase Time**: Biến số bổ sung T được định nghĩa là khoảng thời gian giữa hai lần mua hàng liên tiếp của một khách hàng trong cùng một cửa hàng hoặc trên cùng một trang web, được tính theo công thức:

$$T = L/(F-1) = (Tn - T1)/(F-1)$$

Trong đó:

T: Interpurchase Time

L: Chu kỳ mua sắm

F: Tần suất

T1: Lần mua đầu

Tn: Lần mua cuối

- **Satisfaction**: Trong kinh doanh, sự hài lòng của khách hàng là một yếu tố vô cùng quan trọng, trực tiếp quyết định đến sự thành bại của một doanh nghiệp. Doanh nghiệp muốn thành công xây dựng một hệ thống khách hàng tiềm năng và trung thành thì không thể bỏ qua việc nắm bắt mức độ hài lòng của khách hàng. Nhận thấy tầm quan trọng của sự hài lòng khách hàng, biến số S đã được thêm vào mô hình RFMT và được tính bằng cách lấy tổng điểm đánh giá của tất cả các khách hàng chia cho tổng số đánh giá. Gọi RS_0 …RS_n là điểm đánh giá do khách hàng c đưa ra, do đó giá trị Sc có thể được tính như sau:

$$Sc = \frac{\sum_{i=0}^{n} RS_i}{n+1}$$

Thay vì sử dụng mô hình RFM như trước đây để phân loại và thu hút khách hàng dựa trên hành vi mua hàng của họ, việc thêm khía cạnh về thời gian tồn tại của họ với doanh nghiệp và mức độ hài lòng trung bình của khách hàng sẽ giúp ích hơn cho môi trường chuyển đổi nhanh, nơi khách hàng có thể thay đổi sở thích mua hàng hoặc thậm chí là thay đổi doanh nghiệp khác để mua những thứ họ cần.

Dựa trên những số liệu của mô hình RFMTS, chúng ta có thể theo dõi sự phân bố tổng thể của tất cả các khách hàng theo từng yếu tố và sau đó nhóm chúng lại với nhau để phân tích và phân khúc khách hàng. Hiểu rõ các phân khúc có thể giúp doanh nghiệp điều chỉnh sản phẩm, hoạt động bán hàng, tiếp thị và đầu tư tốt hơn.

2.3 Giới thiệu phân cụm:

Phân cụm dữ liệu (Data Clustering), còn được gọi là phân tích cụm, phân tích phân đoạn hoặc phân tích phân loại, là quá trình nhóm các đối tượng dữ liệu thành các nhóm (cụm) dựa trên sự tương tự hoặc sự liên quan giữa chúng. Mỗi cụm bao gồm các đối tượng dữ liệu có tính chất tương tự trong cụm đó và khác biệt so với các đối tượng trong các cụm khác.

Phân cụm dữ liệu được sử dụng trong nhiều ứng dụng khác nhau:

- *Phân tích khách hàng*
- *Phân tích hình ảnh và video*
- *Phân tích dữ liệu y tế*
- *Phân tích dữ liệu văn bản*

Có rất nhiều thuật toán trong phân cụm:

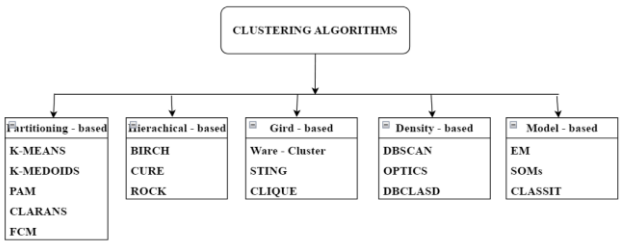


Figure 1.Các kỹ thuật phân cụm

2.4 Phân cụm phân vùng:

Phân cụm dựa trên phân vùng (Partition-based clustering): là một kỹ thuật phân cụm dữ liệu phổ biến. Các phương pháp phân cụm dựa trên phân vùng chia dữ liệu thành các phân vùng (cụm) riêng biệt dựa trên các điểm dữ liệu và các đặc trưng tương đồng hoặc khác biệt giữa chúng. Mỗi phân vùng thường được đại diện bởi một điểm trung tâm hoặc một tập hợp các điểm đại diện..

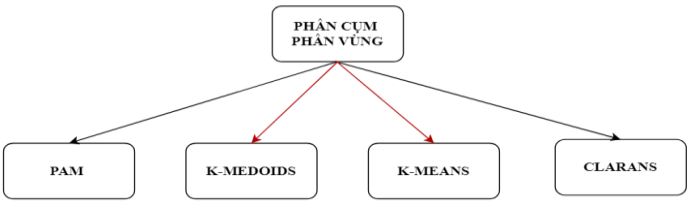


Figure 2. Kỹ thuật phân cụm phân vùng

Bài nghiên cứu của nhóm sẽ tập trung vào việc phân cụm dựa trên phân vùng, tập trung làm rõ ở 2 kỹ thuật là Kmeans và Kmedoids.

2.4.1 Tổng quan về thuật toán K-means:

K - Means clustering được biết đến là một trong những thuật toán tuy đơn giản nhưng đặc biệt hiệu quả trong Machine Learning - bài toán Unsupervised Learning. Trong thực tế, không phải lúc nào cũng có dữ liệu được gắn nhãn một cách hoàn chỉnh. Đôi khi chúng ta chỉ có dữ liệu thô không nhãn và cần phải tự điều chỉnh, phân loại để lấy thông tin hữu ích. Thuật toán phân cụm ra đời đã hỗ trợ rất nhiều cho bài toán đó. [3]

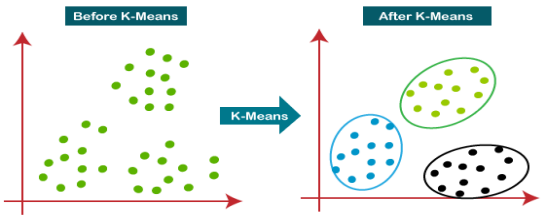


Figure 3. Phân cụm K-Means

a. Phân tích toán học:

X = [x1 ,x2,x3,…,xn] – với N là số điểm dữ liệu.

M = [m1,m2,m3,..,mK] – là K điểm trung tâm .

Với mỗi điểm xi đặt y i = [yi1, yi2 , yi3,…,yiK] là label vector của nó. Có nghĩa chỉ có 1 phần tử trong yI = 1 còn các phần tử còn lại bằng 0. Ví dụ, nếu 1 điểm dữ liệu có label vector là [1,0,0…,0] thì nó thuộc cluster 1, là [0,1,0…,0] thì thuộc cluster 2. Cách mã hóa như này được gọi là biểu diễn one-hot. Có thể viết dưới dạng toán học như sau:

$$y_{ik} \in \{0, 1\}, \quad \sum_{k=1}^{K} y_{ik} = 1 \quad (1)$$

Nếu ta coi center mK là center (hoặc representative) của mỗi cluster và ước lượng tất cả các điểm được phân vào cluster này bởi mk, thì một điểm dữ liệu xi được phân vào cluster k sẽ bị sai số là (xi−mk). Chúng ta mong muốn sai số này có trị tuyệt đối nhỏ nhất nên ta sẽ tìm cách để đại lượng sau đây đạt giá trị nhỏ nhất:

$$\|\mathbf{x}_i - \mathbf{m}_k\|_2^2$$

Hơn nữa, vì xi được phân vào cluster k nên yik =1,yij =0, ∀j≠k. Khi đó, biểu thức bên trên sẽ được viết lại là:

$$y_{ik}\|\mathbf{x}_i - \mathbf{m}_k\|_2^2 = \sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Sai số cho toàn bộ dữ liệu sẽ là:

$$\mathcal{L}(\mathbf{Y}, \mathbf{M}) = \sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Chúng ta cần tối ưu bài toán sau:

$$\mathbf{Y}, \mathbf{M} = \arg\min_{\mathbf{Y},\mathbf{M}} \sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Rất khó tìm được điểm tối ưu vì nó càng có thêm nhiều điều kiện ràng buộc. Tuy nhiên, trong 1 số trường hợp, chúng ta vẫn có thể tìm được nghiệm gần đúng hoặc điểm cực tiểu. Có 2 cách để giải quyết bài toán:

- ***Cố định M và tìm Y:***

Khi các tâm là cố định, chúng ta tìm nhãn cho các điểm dữ liệu để hàm mất mát đạt giá trị nhỏ nhất.

$$\mathbf{y}_i = \arg\min_{\mathbf{y}_i} \sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Đơn giản hơn:

$$j = \arg\min_{j} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Ta có thể kết luận rằng mỗi điểm xi thuộc vào cluster có tâm gần nó nhất sẽ giúp hàm mất mát đạt giá trị min.

- ***Cố định Y tìm M:***

Giả sử đã có nhãn cho từng điểm, tìm tâm mới cho cụm sao cho hàm mất mát đạt giá trị min.

$$\mathbf{m}_j = \arg\min_{\mathbf{m}_j} \sum_{i=1}^{N} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2.$$

Ta tìm nghiệm bằng cách giải đạo hàm bằng 0, vì hàm cần tối ưu là 1 hàm liên tục và có đạo hàm xác định tại mọi điểm. Đặt l(mj) là hàm bên trong 'argmin', ta có đạo hàm:

$$\frac{\partial l(\mathbf{m}_j)}{\partial \mathbf{m}_j} = 2\sum_{i=1}^{N} y_{ij}(\mathbf{m}_j - \mathbf{x}_i)$$

Sau khi giải phương trình ta có:

$$\mathbf{m}_j \sum_{i=1}^{N} y_{ij} = \sum_{i=1}^{N} y_{ij}\mathbf{x}_i$$

$$\Rightarrow \mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

b. Tư tưởng thuật toán:

Input: Dữ liệu X và số lượng cụm cần tìm là k.

Output: Các điểm ở vị trí trung tâm C và gắn nhãn cho từng điểm dữ liệu trong X.

Bước 1: Chọn số cụm k muốn tạo.

Bước 2: Khởi tạo k điểm ban đầu làm center của các cụm.

Bước 3: Lặp lại các bước sau cho đến khi tiêu chí dừng (thường là sự thay đổi của các center rất nhỏ hoặc đạt số lần lặp tối đa đã xác định):

- Phân mỗi điểm dữ liệu vào cụm có center gần nó nhất.
- Cập nhật center của từng cụm bằng cách tính trung bình của tất cả các điểm dữ liệu thuộc vào cụm đó.

- Kiểm tra xem việc gán dữ liệu vào từng cụm có thay đổi so với lần lặp trước hay không. Nếu không có thay đổi đáng kể hoặc không có sự thay đổi thì dừng thuật toán

Bước 4: Kết thúc thuật toán khi tiêu chí dừng được đáp ứng hoặc số lần lặp đạt giới hạn tối đa.

Các phương pháp để tính được khoảng cách sẽ được trình bày sau.

Bên cạnh ưu điểm là dễ dàng sử dụng thì cũng có 1 số nhược điểm như sau:

- Nhạy cảm với điểm khởi tạo ban đầu: K-Means phụ thuộc rất nhiều vào việc chọn các điểm khởi tạo ban đầu. Một lựa chọn không tốt có thể dẫn đến các kết quả kém chất lượng hoặc rơi vào các cụm tối ưu cục bộ.
- Không linh hoạt về hình dạng cụm: K-Means giả định rằng các cụm có hình dạng lồi và có kích thước tương đồng. Trong thực tế, dữ liệu có thể có các cụm hình dạng khác nhau và có kích thước không đồng đều.
- Độ chính xác bị giới hạn: K-Means có thể bị kết thúc sớm nếu không có sự thay đổi đáng kể trong các cụm sau một số lần cập nhật. Điều này có thể dẫn đến kết quả không phải lúc nào cũng là tối ưu.
- Ảnh hưởng lớn bởi nhiễu: K-Means dễ bị ảnh hưởng bởi dữ liệu nhiễu và ngoại lai, do đó có thể tạo ra các cụm sai lệch.
- Không phù hợp cho dữ liệu phi tuyến tính: K-Means hoạt động tốt cho các cụm hình dạng lồi, nhưng không phù hợp cho dữ liệu có cụm phi tuyến tính hoặc các cụm chồng chéo.
- Cần xác định trước số cụm: K-Means yêu cầu bạn xác định trước số lượng cụm.

2.4.2 Tổng quan về thuật toán K-Medoids:

Thuật toán K-Medoids là một phương pháp phân cụm trong lĩnh vực máy học, đặc biệt hiệu quả khi xử lý dữ liệu chứa giá trị ngoại lai.

a. Tư tưởng thuật toán

Ý tưởng chính của K-medoids là để tìm ra k cụm với n đối tượng thì k-medoids chọn ngẫu nhiên ***k đối tượng vào k cụm***, coi mỗi đối tượng này là tâm của cụm. Phân bổ các đối tượng còn lại vào cụm mà sự khác nhau của nó với đối tượng tâm của cụm là gần nhất. Sau đó lặp lại quá trình: Thay đổi đối tượng tâm của mỗi cụm sao cho chất lượng của cụm được cải thiện. Chất lượng của cụm được lượng giá bởi một hàm đo sự khác nhau giữa một đối tượng và đối tượng tâm của cụm chứa nó. Quá trình lặp cho đến khi không còn sự thay đổi nào về lực lượng cũng như hình dạng của các cụm.

b. 4 trường hợp chọn tâm:

Để chọn một đối tượng không là đối tượng tâm O_{random} thay thế tốt cho một đối tượng tâm $O_j$thì mỗi đối tượng p xét theo 4 trường hợp sau đây:

- Trường hợp 1: p đang thuộc vào cụm có tâm là O_j (từ nay gọi là cụm O_j). Nếu O_j được thay thế bởi O_{random} và p gần nhất với $O_{i(i\#j)}$ thì p được gán lại vào O_i
- Trường hợp 2: p đang thuộc vào O_j . Nếu O_j được thay thế bởi Orandom và p gần nhất với O_{random} thì p được gán lại vào Orandom.
- Trường hợp 3: p đang thuộc vào $O_{i(i\#j)}$. Nếu O_j được thay thế bởi O_{random} và p vẫn gần nhất với O_i thì không thay đổi gì cả. Tức là p vẫn thuộc O_i .
- Trường hợp 4: p đang thuộc vào $O_{i(i\#j)}$. NếuO_j được thay thế bởi O_{random} và p gần nhất với O_{random} thì p được gán lại vào O_{random}.

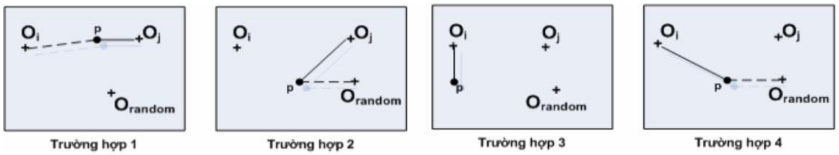


Figure 4. 4 trường hợp chọn tâm tron K-Medoids

c. Phân tích toán học:

Thuật toán K-Medoids có cách hoạt động gần như giống với K-Means, nhưng thay vì chọn trung bình cụm làm đại diện, nó chọn một trong các điểm dữ liệu gốc làm đại diện, được gọi là "exemplar". Việc chọn exemplars thay vì trung bình cụm làm đại diện có thể quan trọng trong các ứng dụng cụ thể.

Nhìn chung về mặt toán học thì K-medoids cũng tương tự như K-means là chúng ta tìm nhãn cho các điểm dữ liệu để hàm mất mát đạt giá trị nhỏ nhất:

$$L = \sum_{i=1}^{n} \quad \sum_{j=1}^{k} \quad w_{ij} d_{(x_i;m_j)}$$

Trong đó:

n là số lượng điểm dữ liệu

k là số lượng cụm

w_{ij} là một biến nhị phân, bằng 1 nếu điểm x_i thuộc cụm j, và bằng 0 nếu không

$d_{(x_i;m_j)}$ là khoảng cách giữa điểm x_i và medoid m_j

d. Các bước thực hiện thuật toán:

Input: Số nguyên k và CSDL gồm n đối tượng cần phân cụm.

Output: Một tập gồm k cụm mà tổng giá trị của sự khác nhau của tất cả các đối tượng đến đối tượng tâm của nhóm chứa nó là nhỏ nhất. Thuật toán:

Bước 1: Chọn k đối tượng bất kỳ vào k cụm. Coi mỗi đối tượng này là tâm của nhóm.

Bước 2: Lặp

Bước 3: Gán mỗi đối tượng còn lại vào một cụm mà nó gần với đối tượng tâm của cụm nhất.

Bước 4: Chọn ngẫu nhiên một đối tượng không là đối tượng tâm, O_{random}

Bước 5: Tính lại giá trị, S, đối với việc đổi O_j với O_{random}.

Bước 6: Nếu S<0 thì đổi Oj với O_{random} để tạo ra một tập với đối tượng tâm mới.

Bước 7: Đến khi không có sự thay đổi nào nữa thì dừng.

Ví dụ: Trong không gian hai chiều cho n = 10 điểm, cần chia thành k =2 cụm. Các bước thực hiện của thuật toán K-medoids được chỉ ra trong hình 3:

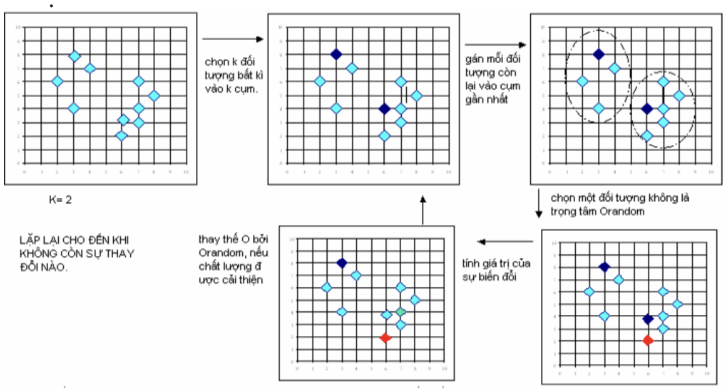


Figure 5. Phân cụm K-Medoids

Đầu tiên, chọn hai điểm bất kì vào hai cụm (điểm màu đen), rồi xét các điểm còn lại và đưa chúng vào một trong hai cụm với điểm tâm lần lượt là hai điểm đã chọn ban đầu.

Tiếp theo, chọn một điểm bất kì khác điểm tâm (điểm màu xám). Tính giá của phép chuyển đổi điểm tâm từ điểm màu trắng -> điểm màu xám. Nếu giá này chất lượng hơn thì coi điểm xám là tâm của cụm mới và thực lặp lại quá trình đó cho đến khi không còn sự thay đổi nào.

Nhận xét: Thuật toán K-medoids mạnh hơn thuật toán K-means trong các trường hợp dữ liệu có nhiễu vì K-medoids chịu ảnh hưởng ít hơn của nhiễu và các giá trị chênh lệch so với giá trị trung bình. Tuy nhiên cả hai thuật toán này đều yêu cầu đưa vào số lượng cụm k.

e. Ưu và nhược điểm của thuật toán:

- Ưu điểm: K-medoids hoạt động tương đối hiệu quả với bộ dữ liệu còn nhiều giá trị nhiễu.
- Nhược điểm:
  - Mất nhiều thời gian chạy với bộ dữ liệu có kích thước lớn.
  - Phải xác định được trước số cụm.
  - Tốn chi phí và kém hiệu quả.

2.5 Các phương pháp xác định và đánh giá chất lượng cụm:

Trong các thuật toán phân cụm thì chúng ta cần phải xác định trước số cụm. Câu hỏi đặt ra ở đây là số lượng cụm cần phân chia tốt nhất đối với một bộ dữ liệu cụ thể ? Cơ bản, không có phương pháp xác định giá trị chính xác nào cho số cụm cụ thể, nhưng một giá trị ước lượng đúng có thể thu được bằng cách sử dụng các phương pháp sau: Elbow, Silhouette, Partition Coeficient và Partition Entropy

2.5.1. Elbow

Elbow là cách giúp ta lựa chọn được số lượng K cụm phù hợp dựa vào đồ thị trực quan hoá bằng cách nhìn vào sự suy giảm của hàm biến dạng và lựa chọn ra điểm khuỷ tay (elbow point). Điểm khuỷ tay là điểm mà ở đó tốc độ suy giảm của hàm biến dạng sẽ thay đổi nhiều nhất. Tức là kể từ sau vị trí này thì gia tăng thêm số lượng cụm cũng không giúp hàm biến dạng giảm đáng kể. Nếu thuật toán phân chia theo số lượng cụm tại vị trí này sẽ đạt được tính chất phân cụm một cách tổng quát nhất mà không gặp các hiện tượng vị khớp (overfitting). [4]

Commented [1]: rút gọn

Quy trình triển khai Elbow method như sau:

Bước 1: Triển khai thuật toán phân cụm (ví dụ k-mean) với các số cụm k thay đổi (ví dụ từ 1 đến 10)

Bước 2: Với mỗi giá trị k, tính giá trị WSS

$WSS = WSS_1 + WSS_2 + \dots$

$$WSS = \sum_{i=0}^{n} dis(x_i - c_i)^2$$

Bước 3: Vẽ Elbow curve theo các giá trị k.

Bước 4: Dựa vào Elbow curve chọn số k thích hợp, là vị trí ở khúc cua.

2.5.2. Silhouette

Silhouette có thể được sử dụng để nghiên cứu khoảng cách tách biệt giữa các cụm kết quả. Biểu đồ Silhouette hiển thị thước đo mức độ gần nhau của mỗi điểm trong một cụm với các điểm trong các cụm lân cận và do đó cung cấp cách đánh giá các tham số như số lượng cụm một cách trực quan. Biện pháp này có phạm vi [-1, 1].Các hệ số Silhouette gần +1 cho thấy mẫu ở xa các cụm lân cận. Giá trị 0 cho biết mẫu nằm trên hoặc rất gần ranh giới quyết định giữa hai cụm lân cận và giá trị âm cho biết các mẫu đó có thể đã được gán sai cụm.

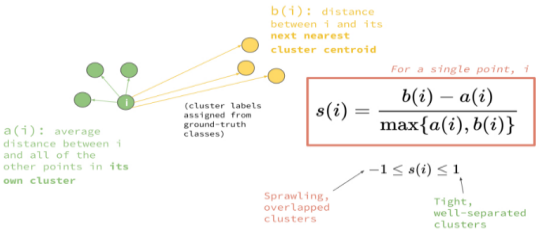


Figure 6.Quy trình tính chỉ số Silhouette cho một điểm dữ liệu điển hình X

2.5.3. Davies-Bouldin index

Chỉ số Davies-Bouldin index (DBI) là một phép đo internal về sự tách biệt giữa các cụm nhằm đánh giá mức độ tối ưu của thuật toán phân cụm bằng cách tính toán sự tương đồng giữa các điểm dữ liệu trong cùng một cụm và sự khác biệt giữa các cụm khác nhau. Phân cụm được đánh giá là tốt khi chỉ số DBI càng nhỏ (càng gần 0) - điều này cho thấy mỗi cụm hoàn toàn tách biệt và không có sự chồng chéo giữa chúng. [1]

$$DB = \frac{1}{n}\sum_{i=1}^{n} Max_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$

Trong đó:

- N: số cụm
- c_x: trọng tâm của cụm x
- σ_x: trung bình khoảng cách của tất cả các phần tử trong cụm x tới trọng tâm c_x
- $d(c_i , c_j)$: khoảng cách giữa hai trọng tâm của cụm i và j

Quy trình tính toán:

- Bước 1: Tính toán trung tâm của các cụm bằng cách lấy giá trị trung bình của tất cả các điểm dữ liệu trong cụm
- Bước 2: Tính toán khoảng cách trung bình từ các điểm trong cụm đến tâm cụm
- Bước 3: Tính toán khoảng cách giữa các cặp trung tâm cụm
- Bước 4: Tính toán ma trận tương đồng giữa các cụm và lấy ra giá trị lớn nhất
- Bước 5: Tính DBI bằng cách lấy giá trị trung bình tổng của độ tương đồng giữa các cụm.

2.5.4. Calinski-Harabasz Index:

Đây là một phương pháp đánh giá internal trong bài toán phân cụm khi mà không có nhãn thực. CHI được biết đến như là một tỷ lệ về phương sai nhằm đánh giá mức độ tương tự của một đối tượng với cụm của chính nó (cohension) với cụm khác (separation). Cohesion được tính dựa trên khoảng cách từ các điểm tòng cụm đến trung tâm của cụm, với separation thì được tính dựa trên khoảng cách tâm của cụm đến với global centroid (global centroid là khoảng cách trung bình giữa các cặp tâm cụm với nhau). [2]

$$CH = \left[\frac{\sum_{k=1}^{K} n_k \|c_k - c\|^2}{K-1}\right] / \left[\frac{\sum_{k=1}^{K}\sum_{i=1}^{n_k} \|d_i - c_k\|^2}{N-K}\right]$$

Trong đó:

- K: số cụm
- N: tổng số điểm trong dữ liệu
- n_k: số lượng điểm trong cụm k
- c_k: tâm của cụm thứ k
- c: global centroid
- d_i: vector điểm thứ i trong cụm k

Thuật toán được đánh giá là phân cụm tốt khi chỉ số CH của nó lớn - nghĩa là cụm dày đặc và phân tách tốt giữa các cụm. Tuy nhiên thì thuật toán này lại không có giá trị giới hạn, điều này phụ thuộc vào bài toán, giả thuyết và mục tiêu của người dùng hướng đến. Chính vì vậy mà khi lựa chọn số cụm sao cho phù hợp thì thuật toán này mong muốn có những điểm khúc khuỷa (điểm cực tiểu) và từ đó hình thành lên những điểm đỉnh (điểm cực đại). Nếu giá trị CHI chỉ đi ngang hoặc lên hoặc xuống thì cần phải lựa chọn một phương pháp khác để thay thế, vì lúc này đối với phương pháp CHI nó không thể diễn giải được một lựa chọn như thế nào là tốt.

Quy trình tính toán:

- Bước 1: Tính BSS (between-cluster sum of squares) trung bình là trung bình tổng lượng biến động giữa các cụm - đo lường sự tách biệt giữa các cụm.
- Bước 2: Tính WSS (within-cluster sum of squares) trung bình là trung bình tổng lượng biến động bên trọng mỗi cụm - đo lường độ giống nhau bên trong mỗi cụm.
- Bước 3: Tính CHI bằng cách lấy thương giữa BSS và WSS

3. QUY TRÌNH TRIỂN KHAI:

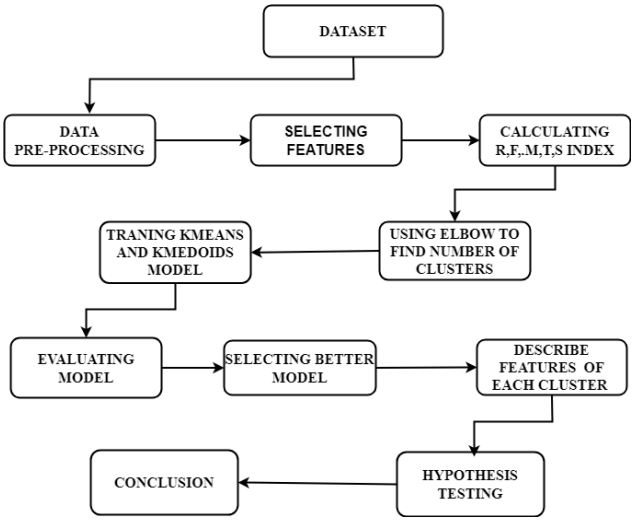


Figure 7. Framework

3.1 Thu thập dữ liệu:

3.3.1. Thu thập dữ liệu:

Dữ liệu được lấy từ Kaggle có tên là Brazilian E-Commerce Public Dataset với nội dung là về các đơn đặt hàng được thực hiện tại Olist Store. Tập dữ liệu có thông tin về 100 nghìn đơn đặt hàng từ năm 2016 đến năm 2018 được thực hiện tại nhiều thị trường ở Brazil. Các tính năng của nó cho phép xem đơn đặt hàng từ nhiều khía cạnh: từ trạng thái đơn hàng, giá cả, thanh toán và hiệu suất vận chuyển hàng hóa đến vị trí của khách hàng, thuộc tính sản phẩm và cuối cùng là các bài đánh giá do khách hàng viết.

3.1.2. Tổng quan dữ liệu:

Trong bộ dữ liệu thu được nhóm sẽ lấy ra 4 bảng như sau:

Table 1. Mô tả dữ liệu

Bảng 1 : Olist - order - payments - dataset		
	Columns_name	**Description**
1	order_id	Mã đơn hàng
2	payment_sequential	Thứ tự thanh toán của từng loại hình
3	payment_type	Phương thức thanh toán
4	payment_installments	Kì hạn trả góp
5	payment_value	Giá trị đơn hàng
Bảng 2 : Olist - order - reviews - dataset		
6	review_id	Mã đánh giá
7	review_score	Điểm từ 1-5 do khách hàng đưa ra trong bản khảo sát mức độ hài lòng

8	review_comment_title	Tiêu đề nhận xét của khách hàng
9	review_comment_message	Tin nhắn bình luận từ bài đánh giá của KH
10	review_creation_date	Ngày gửi bảng khảo sát cho KH
11	review_answer_timestamp	Hiển thị thời gian trả lời khảo sát
Bảng 3 : Olist - order - customer - dataset		
12	customer_id	Mã khách hàng
13	customer_unique_id	Mã định danh khách hàng
14	customer_zip_code_prefix	Mã zip nơi ở của khách hàng
15	customer_city	Thành phố khách hàng
16	customer_state	Khu vực khách hàng
Bảng 4 : Olist - order - dataset		
17	order_status	Tình trạng đơn hàng
18	order_purchase_timestamp	Thời gian mua hàng
19	order_approved_at	Thời gian phê duyệt thanh toán
20	order_delivered_carrier_date	Thời gian đăng đơn hàng
21	order_delivered_customer_date	Thời gian giao hàng thực tế cho KH
22	order_estimated_delivery_date	Thời gian giao hàng ước tính

3.2 Tiền xử lý dữ liệu

3.2.1. Nối các bảng thành một tập dữ liệu thống nhất:

Tiến hành nối dữ liệu với bảng fact được chọn là olist_order_dataset và các bảng dim lần lượt là olist_order_reviews_dataset (được nối trên trường order_id), bảng dim olist_order_payments_dataset (được nối trên trường order_id), và cuối cùng là bảng dim olist_order_customer_dataset (được nối trên trường customer_ID).

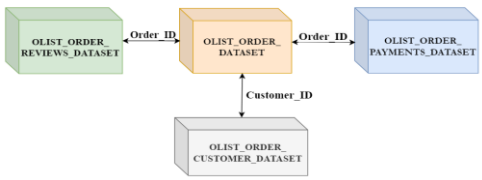


Figure 8. Nối dữ liệu

3.2.2. Chuyển đổi kiểu dữ liệu

Tiếp tục kiểm tra kiểu dữ liệu và số bản ghi nhận thấy dữ liệu thời gian đang ở dạng văn bản và tiến hành chuyển đổi thành kiểu dữ liệu datetime các cột như: order_purchase_timestamp, order_approved_at. Và không tồn tại trường dữ liệu nào bị thiếu giá trị (tồn tại giá trị null).

```
data_delivered['order_purchase_timestamp'] = pd.to_datetime(data_delivered['order_purchase_timestamp'])
data_delivered['order_approved_at'] = pd.to_datetime(data_delivered['order_approved_at'])
```

Figure 9. Chuyển đổi kiểu dữ liệu

3.2.3. Loại bỏ giá trị ngoại lai

Sau khi tiến hành tính các giá trị R,F,M,T,S nhận thấy trường M và R tồn tại giá trị outlier. Tiến hành loại bỏ outlier bằng phương pháp tứ phân vị. Cuối cùng chuẩn hóa dữ liệu bằng phương pháp Z-score giảm chi phí tính toán với thuật toán phân cụm.

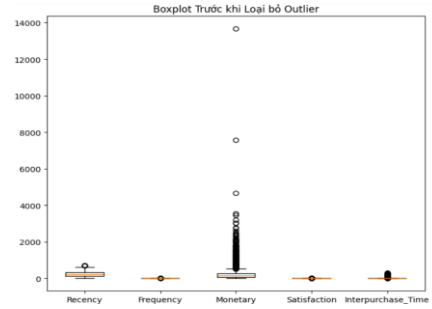


Figure 10. Hình ảnh phân phối của các đặc trưng trước khi loại bỏ outlier

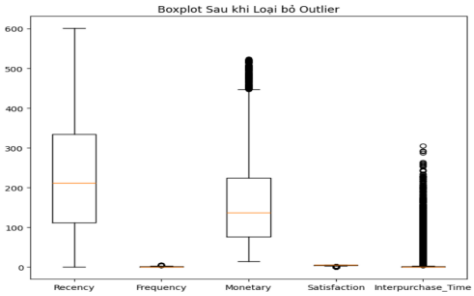


Figure 11. Hình ảnh phân phối của các đặc trưng sau khi loại bỏ outlier

3.2.4 Chia dữ liệu

Tiến hành thực nghiệm và so sánh giữa hai thuật toán K-means và K-mediods tuy nhiên với thuật toán K-mediods được tính toán dựa trên ma trận pair-wise yêu cầu số dòng và cột tương ứng với số điểm là rất lớn. Vì vậy để thực nghiệm nhóm đã tiến hành cắt bớt dữ liệu nhưng đảm bảo được phân phối của tổng thể của từng đặc trưng và tiến hành thực nghiệm.

3.3 Ứng dụng dữ liệu vào phân cụm khách hàng

3.3.1 Phân cụm khách hàng với thuật toán Kmeans

Để phân cụm khách hàng bằng thuật toán K-means, chúng ta thực hiện các bước sau:

Chuẩn bị dữ liệu:

Xác định Biến X:

Xác định các biến cần sử dụng để phân loại khách hàng. Trong trường hợp này, RFMTS đã được chuẩn hóa.

Chuẩn hóa Dữ liệu:

Chuẩn hóa dữ liệu nếu cần thiết để đảm bảo rằng các biến có cùng thang đo. Điều này giúp K-means hoạt động hiệu quả hơn.

```
from sklearn.preprocessing import StandardScaler
data = df[['Recency','Frequency','Monetary','Satisfaction','Interpurchase_Time']]
X_kmeans = data.values
scaler = StandardScaler()
X_kmeans = scaler.fit_transform(X_kmeans)
```

Xác định số cụm:

Xác định Số Cụm bằng Phương pháp Elbow:

- Chạy thuật toán K-means với một loạt các số cụm khác nhau.
- Đánh giá Elbow Method để xác định số lượng cụm tốt nhất.

```
from sklearn.cluster import KMeans
# khai báo danh sách K – số lượng cụm:
K = range(3,11)
# tạo dsách trống để lưu teuwx giá trị distortions cho từng gtri K
distortions = []
# Sử dụng vòng lặp for để lặp qua từng giá trị K
for k in K:
# khởi tạo đối tượng km với số lượng cụm qua từng vòng lặp
    km = KMeans(n_clusters = k)
# sử dụng dữ liệu đã chuẩn hóa
    km = km.fit(X_kmeans)
# tính tổng bình phương khoảng cách từ mỗi điểm đến tâm cụm
    distortions.append(pd.Series({'K': k,
                                  'distortions': km.inertia_}))

inertias= pd.concat(distortions, axis=1).T.set_index('K')
# tạo hình vẽ mới với kích thước
# tạo biểu đồ đường bằng cách sử dụng hàm plot
# trục X: inertias – đại diện cho số cụm
# trục Y: giá trị tương ứng với từng cụm
plt.plot(inertias.index, inertias['distortions'], '-o')
plt.title('Phương pháp Elbow')
plt.xlabel('Số lượng cụm (k)')
plt.ylabel('inertia')
plt.xticks(K)
plt.show()
```

Đánh giá kết quả:

- Chạy K-means với số cụm được xác định từ Elbow Method.
- Kết hợp với các chỉ số đánh giá như Silhouette Score, Calinski-Harabasz Index, và Davies-Bouldin Index để so sánh và đánh giá chất lượng của việc phân cụm.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import calinski_harabasz_score, davies_bouldin_score, silhouette_score


# Tạo subplot cho ba chỉ số
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

calinski_scores1 = []
davies_scores1 = []
silhouette_scores1 = []

for k in range(3,11):
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X_kmeans)
    labels = kmeans.labels_
    # Tính các chỉ số cho mỗi giá trị k
    calinski_score = calinski_harabasz_score(X_kmeans, labels)
    davies_score = davies_bouldin_score(X_kmeans, labels)
    silhouette_score_value = silhouette_score(X_kmeans, labels)

    calinski_scores1.append(calinski_score)
    davies_scores1.append(davies_score)
    silhouette_scores1.append(silhouette_score_value)
```

Trực quan hoá giá trị của các chỉ số trên theo K cụm

```
axes[0].plot(range(3,11), calinski_scores1, marker='o')
axes[0].set_title('Calinski-Harabasz Index')
axes[0].set_xlabel('Number of Clusters (k)')
axes[0].set_ylabel('Score')

axes[1].plot(range(3,11), davies_scores1, marker='o')
axes[1].set_title('Davies-Bouldin Index')
axes[1].set_xlabel('Number of Clusters (k)')
axes[1].set_ylabel('Score')

axes[2].plot(range(3,11), silhouette_scores1, marker='o')
axes[2].set_title('Silhouette Score')
axes[2].set_xlabel('Number of Clusters (k)')
axes[2].set_ylabel('Score')

# Hiển thị biểu đồ
plt.tight_layout()
plt.show()
```

3.3.2 Phân cụm khách hàng với thuật toán Kmedoids

Tương tự như Kmeans thì quy trình phân cụm khách hàng bằng thuật toán KMedoids cũng bao gồm các bước:

Chuẩn bị dữ liệu:

Xác định Biến X:

Xác định các biến cần sử dụng để phân loại khách hàng. Trong trường hợp này, RFMTS đã được chuẩn hóa.

Chuẩn hóa Dữ liệu:

Chuẩn hóa dữ liệu nếu cần thiết để đảm bảo rằng các biến có cùng thang đo. Điều này giúp KMedoids hoạt động hiệu quả hơn.

```
from sklearn.preprocessing import StandardScaler
data = df[['Recency','Frequency','Monetary','Satisfaction','Interpurchase_Time']]
X = data.values
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

Xác định số cụm:

Xác định Số Cụm bằng Phương pháp Elbow:

- Chạy thuật toán KMedoids với một loạt các số cụm khác nhau.
- Đánh giá Elbow Method để xác định số lượng cụm tốt nhất.

```python
from sklearn_extra.cluster import KMedoids
k_values = range(3, 11)

# Danh sách inertia tương ứng với mỗi số lượng cụm
inertia_values = []

# Tính inertia cho mỗi số lượng cụm
for k in k_values:
    kmedoids = KMedoids(n_clusters=k, random_state=0)
    kmedoids.fit(X)
    inertia_values.append(kmedoids.inertia_)

# Trực quan hóa giá trị inertia
plt.plot(k_values, inertia_values, '-o', color='red')
plt.xlabel('Số lượng cụm (k)')
plt.ylabel('Inertia')
plt.title('Phương pháp Elbow')
plt.xticks(k_values)
plt.show()
```

Đánh giá kết quả:

- Chạy K-means với số cụm được xác định từ Elbow Method.
- Kết hợp với các chỉ số đánh giá như Silhouette Score, Calinski-Harabasz Index, và Davies-Bouldin Index để so sánh và đánh giá chất lượng của việc phân cụm.

```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import calinski_harabasz_score, davies_bouldin_score, silhouette_score


# Tạo subplot cho ba chỉ số
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

calinski_scores = []
davies_scores = []
silhouette_scores = []

for k in range(3,11):
    kmedoids = KMedoids(n_clusters=k, random_state=42)
    kmedoids.fit(X)
    labels = kmedoids.labels_
    # Tính các chỉ số cho mỗi giá trị k
    calinski_score = calinski_harabasz_score(X, labels)
    davies_score = davies_bouldin_score(X, labels)
    silhouette_score_value = silhouette_score(X, labels)

    calinski_scores.append(calinski_score)
    davies_scores.append(davies_score)
    silhouette_scores.append(silhouette_score_value)
```

Trực quan hoá giá trị của các chỉ số trên theo K cụm

```
axes[0].plot(range(3,11), calinski_scores, marker='o',color='red')
axes[0].set_title('Calinski-Harabasz Index')
axes[0].set_xlabel('Number of Clusters (k)')
axes[0].set_ylabel('Score')

axes[1].plot(range(3,11), davies_scores, marker='o',color='red')
axes[1].set_title('Davies-Bouldin Index')
axes[1].set_xlabel('Number of Clusters (k)')
axes[1].set_ylabel('Score')

axes[2].plot(range(3,11), silhouette_scores, marker='o',color='red')
axes[2].set_title('Silhouette Score')
axes[2].set_xlabel('Number of Clusters (k)')
axes[2].set_ylabel('Score')

# Hiển thị biểu đồ
plt.tight_layout()
plt.show()
```

3.3.3 So sánh 2 thuật toán

Nhìn chung thì các bước thực hiện

	K-MEANS	K-MEDOIDS
Phương thức	Thuật toán K-Means sử dụng **trung bình của các điểm dữ liệu** trong cụm để xác định trung tâm của cụm.	Trong K-Medoids, medoid được sử dụng thay vì trung tâm cụm. Medoid là một điểm dữ liệu thực tế trong tập dữ liệu
Độ phức tạp	K-Means có độ phức tạp tính toán thấp hơn K-Medoids. Việc tính toán trung bình các điểm dữ liệu trong cụm là tgương đối đơn giản và nhanh chóng	K-Medoids có độ phức tạp tính toán cao hơn K-Means do việc tính toán khoảng cách giữa từng cặp điểm dữ liệu trong cụm. Điều này tốn nhiều thời gian và tài nguyên tính toán hơn.
Quy mô	Quy mô dữ liệu lớn	Quy mô dữ liệu nhỏ
Hiệu quả	Hiệu quả cao hơn	Hiệu quả thấp hơn
Tính thực thi	Dễ cài đặt	Phức tạp
Dữ liệu Outlier	Không hiệu quả khi có outlier	Ít bị ảnh hưởng
Yêu cầu số cụm	Có	Có

Table 2. So sánh hai thuật toán

4. KẾT QUẢ DỰ ÁN:

4.1 Kết quả phân cụm từ K-Means:

Một bước cơ bản cho các thuật toán phân cụm không giám sát là xác định số cụm tối ưu mà dữ liệu có thể được phân cụm vào đó. Ở đây, nhóm sẽ dùng một trong những phương pháp phổ biến nhất là Elbow để xác định giá trị tối ưu của k cụm. Ở biểu đồ bên dưới, có thể thấy số cụm bằng 5 là thích hợp nhất để phân cụm dữ liệu.

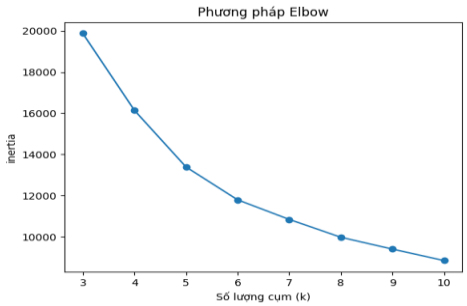


Figure 12. Xác định số cụm K-means bằng Elbow

Ngoài phương pháp Elbow, nhóm cũng dùng thêm 3 chỉ số là Silhouette Score, Davies-Bouldin Index và Calinski-Harabasz Index để xác thực phân cụm cũng như để xác định số cụm tối ưu cho thuật toán K-means. Cũng như Elbow thì các chỉ số này cũng cho ra số cụm tốt nhất là 5, ngoại trừ Silhouette cho ra số cụm là 4. Điều này có thể giải thích vì theo Yannis Poulakis, chỉ số Silhouette có thể không phải là chỉ mục để đánh giá mọi giải pháp phân cụm và trong một số trường hợp thì việc so sánh chỉ dựa trên chỉ số này xu hướng khiến chúng ta chọn sai số cụm mà tại đó thuật toán không thật sự hoạt động tốt nhất.

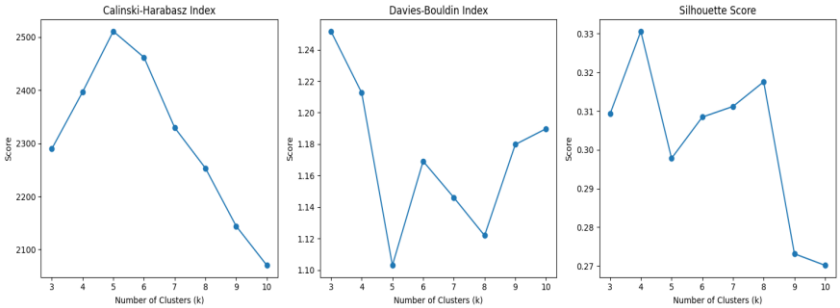


Figure 13. Xác định số cụm K-means bằng DBI, Silhouette, Calinski

Sau khi đã lựa được số cụm tốt nhất thì chúng ta sẽ tiến hành phân cụm dữ liệu bằng KMeans với số cụm là 5 (cụm 0 - cụm 4):

	Recency	Frequency	Monetary	Satisfaction	Interpurchase_Time	Cluster
0	169	1	129.76	1.0	1	1
1	233	1	36.68	5.0	1	0
2	153	1	107.44	3.0	1	0
3	183	1	217.74	1.0	1	1
4	463	1	130.56	5.0	1	4
5	277	1	65.78	2.0	1	1
6	520	1	143.73	5.0	1	4
7	313	1	433.98	1.0	1	1
8	227	2	354.37	5.0	86	2
9	270	1	85.04	4.0	1	4

Figure 14. Kết quả phân cụm bằng K-means

4.2 Kết quả phân cụm từ K-Medoids

Tương tự như K-means, ở K-medoids cũng xem xét số cụm tốt nhất bằng phương pháp Elbow. Tuy nhiên, theo hình vẽ bên dưới rất khó để xác định đâu là số cụm tối ưu nhất cho thuật toán khi giá trị được cân nhắc giữa K=5 hoặc K=9. Vì vậy nhóm sẽ kết hợp thêm các giá trị đánh giá để tìm ra giá trị K phù hợp.

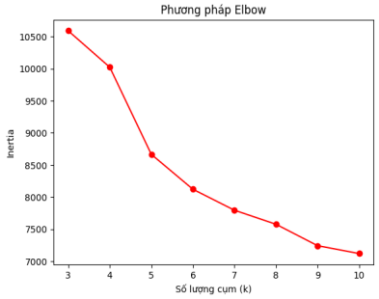


Figure 15. Xác định số cụm K-medoids bằng Elbow

Tương tự như Elbow, 2 chỉ số Silhouette Score và Calinski-Harabasz Index cũng cho ra số cụm bằng 5, còn Davies-Bouldin Index cho ra số cụm tốt nhất bằng 7 chênh lệch với số cụm bằng 5 là …%. Dựa vào kết quả của tất cả các chỉ số thì ở thuật toán K-meadoids số cụm tối ưu được chọn là 5.

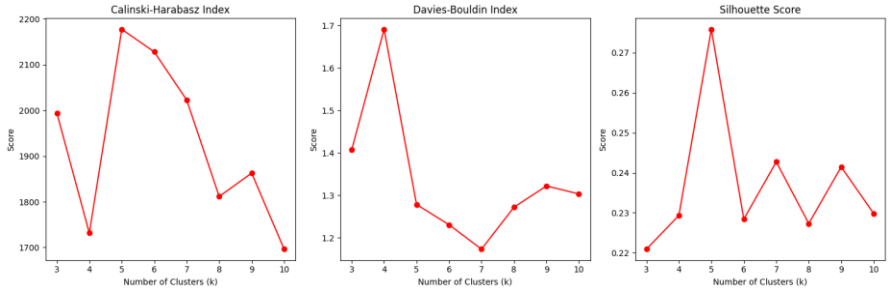


Figure 16. Xác định số cụm K-medoids bằng DBI, Silhouette, Calinski

Sau khi đã lựa được số cụm tốt nhất thì chúng ta sẽ tiến hành phân cụm dữ liệu bằng K-medoids với số cụm là 5 (cụm 0 - cụm 4):

	Recency	Frequency	Monetary	Satisfaction	Interpurchase_Time	Cluster
0	169	1	129.76	1.0	1	2
1	233	1	36.68	5.0	1	4
2	153	1	107.44	3.0	1	2
3	183	1	217.74	1.0	1	2
4	463	1	130.56	5.0	1	0
5	277	1	65.78	2.0	1	2
6	520	1	143.73	5.0	1	0
7	313	1	433.98	1.0	1	2
8	227	2	354.37	5.0	86	1
9	270	1	85.04	4.0	1	0

Figure 17. Kết quả phân cụm bằng K-medoids

4.3 So sánh và đánh giá kết quả

4.3.1. So sánh

Kết quả so sánh các chỉ số tại số cụm bằng 5 tại 2 thuật toán cho ra kết quả K-means cao hơn K-medoids ở Silhouette Score, Calinski-Harabasz Index và thấp hơn ở chỉ số Davies-Bouldin Index. Vì vậy có thể kết luận rằng ở tập dữ liệu mẫu RFMTS này thuật toán K-means cho ra kết quả tốt hơn thuật toán K-medoids.

	K-Means	K-Medoids
Silhouette Score	0.297495	0.275831
Davies-Bouldin Index	1.102509	1.278269
Calinski-Harabasz Index	2510.789402	2177.334206

Table 3. So sánh kết quả hai thuật toán

Tiến hành phân cụm khách hàng bằng K-means cho ra được 5 cụm với số lượng khách hàng của mỗi cụm như sau:

Cluster	Customer
0	2085
1	851
2	574
3	1632
4	1554

Table 4. Số lượng khách hàng ở mỗi cụm

4.3.2. Đánh giá

Mỗi cụm đại diện cho một phân khúc thị trường. Để đưa ra quyết định đúng đắn và phát triển các chiến lược tiếp thị hiệu quả, cần phân tích và hiểu rõ đặc điểm của từng cụm. Các đặc điểm RFMTS của từng cụm được phân tích trong phần này.

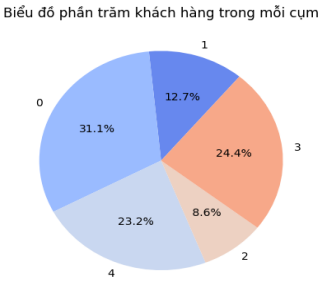


Figure 18. Biểu đồ phần trăm số lượng khách hàng ở mỗi cụm

Cluster	Recency_mean	Frequency_mean	Monetary_mean	InterpurchaseTime_mean	Satisfaction_mean	Value_counts
0	126.111271	1.016787	112.670149	1.192806	4.646043	2085
1	242.450059	1.166863	148.486898	2.547591	1.598433	851
2	130.822300	2.076655	230.154774	127.395470	4.366007	574
3	229.756740	1.998162	263.231716	16.893382	4.423814	1632
4	404.536680	1.074646	117.679311	1.374517	4.616742	1554

Figure 19. Gía trị trung bình của các chỉ số RFMTS tại mỗi cụm

Cluster	RankR	RankF	RankM	RankT	RankS
0	1.0	1.0	1.0	1.0	5.0
1	4.0	3.0	3.0	3.0	1.0
2	2.0	5.0	4.0	5.0	2.0
3	3.0	4.0	5.0	4.0	3.0
4	5.0	2.0	2.0	2.0	4.0

Figure 20. Gía trị xếp hạng ở các chỉ số RFMTS tại mỗi cụm

Cụm	**Mô tả cụm**	**Loại khách hàng**	**Hành động**
0	R thấp, F thấp, M thấp, T ngắn, S cao	Khách hàng tích cực	Tăng cường giao tiếp, tạo ưu đãi đặc biệt, xây dựng chương trình khách hàng thân thiết, tạo trải nghiệm mua sắm dễ dàng, gửi thông tin sản phẩm mới, …
1	R cao, F trung bình, M trung bình, T trung bình, S thấp	Khách hàng rời bỏ	Gửi thông tin về các sản phẩm mới, và gửi ưu đãi để khuyến khích khách hàng mua hàng lại
2	R thấp, F cao, M cao, T dài, S thấp	Khách hàng tiềm năng	Tạo trải nghiệm mua hàng tốt hơn, tập trung cải thiện quá trình trải nghiệm mua hàng của họ
3	R trung bình, F cao, M cao, T dài, S trung	Khách hàng trung thành	Đây là khách hàng đóng góp rất lớn cho doanh nghiệp tuy nhiên cần tìm hiểu lý do

	bình		khiến họ thất vọng và cải thiện điều đó.
4	R cao, F thấp, M thấp, T ngắn, S cao	Khách hàng hỗ trợ	Với tập khách hàng này nên tiếp cận và nhấn mạnh rằng sản phẩm, dịch vụ của công ty sẽ mang lại giải pháp tuyệt vời cho họ, giúp họ vượt qua những hạn chế khác.

Table 5. Mô tả cụm

4.4 Kiểm định kết quả

Ở phần đánh giá và so sánh kết quả giữa 2 thuật toán nhóm đã sử dụng phương pháp học máy để đưa ra các chỉ số đánh giá nhằm chọn ra thuật toán có hiệu suất mô hình tốt nhất. Ở phần này, sau khi đã chọn được K-means để phân cụm khách hàng dựa tiêu chí R,F,M,T,S thì nhóm sẽ sử dụng phương pháp thống kê để kiểm định kết quả phân cụm ở thuật toán này.

Ở phần kiểm định này, nhóm sử dụng kiểm định MANOVA để xem xét liệu có sự khác biệt nào trong các chỉ số R,F,M,T,S giữa 5 cụm đã phân loại ở trên.

Cặp giả thuyết:

- H0: Không có sự khác biệt có ý nghĩa nào giữa các cụm với các giá trị R,F,M,T,S
- H1: Có sự khác biệt ý nghĩa giữa ít nhất hai cụm đối với ít nhất một giá trị trong R,F,M,T,S

Kết quả chạy được từ python:

```
                  Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept        Value  Num DF   Den DF   F Value   Pr > F
----------------------------------------------------------------
         Wilks' lambda 0.0978 5.0000 6690.0000 12336.6750 0.0000
        Pillai's trace 0.9022 5.0000 6690.0000 12336.6750 0.0000
Hotelling-Lawley trace 9.2202 5.0000 6690.0000 12336.6750 0.0000
   Roy's greatest root 9.2202 5.0000 6690.0000 12336.6750 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
        Cluster         Value  Num DF   Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda 0.4795 5.0000 6690.0000 1452.2138 0.0000
         Pillai's trace 0.5205 5.0000 6690.0000 1452.2138 0.0000
 Hotelling-Lawley trace 1.0854 5.0000 6690.0000 1452.2138 0.0000
    Roy's greatest root 1.0854 5.0000 6690.0000 1452.2138 0.0000
================================================================
```

Figure 21. Kiểm định kết quả bằng MANOVA

Nhìn vào bảng kết quả ta có thể thấy các chỉ số như Pillai's Trace, Wilks' Lambda, Hotelling's Trace, và Roy's Largest Root. Đây là các thống kê MANOVA chính, được tính toán từ ma trận phương sai giữa các biến phụ thuộc giữa các nhóm. Khoảng giá trị giữa các chỉ số này sẽ được thể hiện như sau:

Pillai's Trace:

Khoảng giá trị: từ 0 đến 1.

Giá trị gần 0: Cho thấy không có sự khác biệt ý nghĩa giữa các nhóm.

Giá trị gần 1: Cho thấy có sự khác biệt đáng kể giữa các nhóm.

Wilks' Lambda:

Khoảng giá trị: từ 0 đến 1.

Giá trị gần 0: Cho thấy có sự khác biệt đáng kể giữa các nhóm.

Giá trị gần 1: Cho thấy không có sự khác biệt ý nghĩa giữa các nhóm.

Hotelling's Trace:

Khoảng giá trị: từ 0 đến vô cùng.

Giá trị tăng lên: Cho thấy sự khác biệt giữa các nhóm tăng lên.

Roy's Greatest Root:

Khoảng giá trị: từ 0 đến vô cùng.

Giá trị tăng lên: Cho thấy sự khác biệt giữa các nhóm tăng lên.

Đối với kết quả chung (Intercept):

Các chỉ số Pillai's Trace, Wilks' Lambda, Hotelling's Trace, và Roy's Largest Root cho thấy có sự khác biệt giữa các cụm với các biến phụ thuộc(R,F,M,T,S). Bên cạnh đó giá trị p-value (Pr > F) rất thấp (0.0000) và p-value<0.05. Do đó, chúng ta bác bỏ giả thuyết H1 và chấp nhận giả thuyết H0

Kết quả cho từng Cluster:

Pr > F (P-value): 0.0000 bé hơn 0.05 vì vậy chúng ta bác bỏ giả thuyết không có sự khác biệt giữa các nhóm đối với ít nhất một biến phụ thuộc cho từng cluster.

5. Kết luận

Từ những khó khăn của doanh nghiệp, người bán hàng, … về cách thức khai thác dữ liệu và đưa ra chính sách hiệu quả. Nhóm đã tiến hành xây dựng các thông số R,F,M,T,S dựa trên các giao dịch với mục tiêu phân tích về hành vi và đặc trưng của khách hàng. Thực nghiệm phân cụm khách hàng dựa trên các thuật toán hiện đại như K-means, K-mediods và nhận thấy, với tập dữ liệu mẫu của nhóm sử dụng thì K-means sẽ có các chỉ số đánh giá tốt hơn so với K-mediods. Đồng thời nhóm cũng đã phân tích được cách thức lựa chọn phương pháp phân cụm hiệu quả với hướng tiếp cận phân vùng, và cũng đã đưa ra được mặt ưu và nhược điểm của từng thuật toán sử dụng, điều này làm rõ ràng hơn về cách lựa chọn thuật toán phù hợp cho bài toán phân cụm trong CRM.

Minh chứng rõ hơn cho hiệu quả phân cụm, khi thực hiện kiểm định MANOVA và kiểm định cho ra kết quả giữa các cụm có sự khác biệt rất lớn về ý nghĩa giữa các biến, cho thấy mức độ tương đồng giữa các cụm là rất bé (gần như không tồn tại).

REFERENCES

[1] "CRM là gì? Customer Relationship Management System quan trọng như thế nào?".

[2] "Phân tích RFM là gì," 2023.

[3] M. L. c. bản, "Bài 4: K-means Clustering," 2017.

[4] B. Saji, "Elbow Method for Finding the Optimal Number of Clusters in K-Means," 2023.

[5] 5 9 2023. [Online]. Available: https://www.geeksforgeeks.org/davies-bouldin-index/.

[6] 25 4 2022. [Online]. Available: https://www.geeksforgeeks.org/calinski-harabasz-index-cluster-validity-indices-set-3/.

[7] "Các độ đo tương tự trong khai phá dữ liệu," 22 October 2018. [Online]. Available: http://bis.net.vn/forums/p/1872/9576.aspx. [Accessed 8 November 2023].

[8] [Online]. Available: https://en.wikipedia.org/wiki/Davies%E2%80%93Bouldin_index#:~:text=The%20Davies%E2%80%93Bouldin%20index%20.

In [1]:
```
# import library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
from sklearn.preprocessing import StandardScaler
from sklearn_extra.cluster import KMedoids
from sklearn.cluster import KMeans
from sklearn.metrics import calinski_harabasz_score, davies_bouldin_score,
from statsmodels.multivariate.manova import MANOVA
import math

import warnings
```

Description

Using the clustering method to segment customers based on the RFMTS model helps businesses better understand the behavior and distinctive characteristics of each customer segment. This can aid in customizing marketing strategies, customer care, and relationship management to optimize the value derived from each customer segment.

Table of contents

I. Merge sub-datasets into fact dataset

In [2]:
```
# read datasets
Payments = pd.read_csv('olist_order_payments_dataset.csv')Orders =
pd.read_csv('olist_orders_dataset.csv')
Customers = pd.read_csv('olist_customers_dataset.csv') Reviews =
pd.read_csv('olist_order_reviews_dataset.csv')
```

In [3]:
```
# merge data with inner join method
data = Orders.merge(Customers, on="customer_id").merge(Payments, on="order
```

In [4]:
```
# check null value on each column
data.isnull().sum()[data.isnull().sum() > 0]
```

Out[4]:
```
order_approved_at                  171
order_delivered_carrier_date      1861
order_delivered_customer_date     3030
review_comment_title             91681
review_comment_message           60862
dtype: int64
```

In [5]:
```
# get only data delivered status
data_delivered = data[data.order_status == 'delivered'].reset_index(drop=T
#only delivered orders
```

In [6]:
```
# # check null value on each column
data_delivered.isnull().sum()[data_delivered.isnull().sum() > 0]
```

Out[6]:
```
order_approved_at                  14
order_delivered_carrier_date        2
order_delivered_customer_date       8
review_comment_title            89025
review_comment_message          59830
dtype: int64
```

In [7]:
```
# Drop unneccesary columns
data_delivered = data_delivered.drop(['order_delivered_carrier_date', 'ord
                                      'review_creation_date','review_answe
```

II. Clean data

In [8]:
```
data_delivered.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100649 entries, 0 to 100648
Data columns (total 17 columns):
 #   Column                       Non-Null Count   Dtype  
---                                 --------------------------------------
 0   order_id                        100649  non-null   object 
 1   customer_id                     100649  non-null   object 
 2   order_status                    100649  non-null   object 
 3   order_purchase_timestamp        100649  non-null   object 
 4   order_approved_at               100635  non-null   object 
 5   customer_unique_id              100649  non-null   object 
 6   customer_zip_code_prefix        100649  non-null   int64  
 7   customer_city                   100649  non-null   object 
 8   customer_state                  100649  non-null   object 
 9   payment_sequential              100649  non-null   int64  
 10  payment_type                    100649  non-null   object 
 11  payment_installments            100649  non-null   int64  
 12  payment_value                   100649  non-null   float64
 13  review_id                       100649  non-null   object 
 14  review_score                    100649  non-null   int64  
 15  review_comment_title            11624 non-null     object 
 16  review_comment_message          40819 non-null     object 
dtypes: float64(1), int64(4), object(12)
memory usage: 13.1+ MB
```

In [9]:
```
data_delivered.columns
```

Out[9]:
```
Index(['order_id', 'customer_id', 'order_status', 'order_purchase_timesta
mp',
       'order_approved_at', 'customer_unique_id', 'customer_zip_code_pref
ix',
       'customer_city', 'customer_state', 'payment_sequential', 'payment_
type',
       'payment_installments', 'payment_value', 'review_id', 'review_scor
e',
       'review_comment_title', 'review_comment_message'],
      dtype='object')
```

In [10]:
```
# convert object to datetime
data_delivered['order_purchase_timestamp'] = pd.to_datetime(data_delivered
data_delivered['order_approved_at'] = pd.to_datetime(data_delivered['order
```

In [11]:
```
data_delivered.head()
```

Out[11]:

	order_id	customer_id	order_status	ord
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
3	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	
4	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	

In [22]:
```
# dữ liệu sau nối
# data_delivered.to_csv('Group_8_46K29.2.csv')
```

III. EDA - Feature Selection

In [12]:
```
Latest_Date = data_delivered['order_purchase_timestamp'].max()
# get lastest date in datasets
```

In [13]:

```
RFMTS = data_delivered.groupby('customer_unique_id').agg({
'order_purchase_t'order_id': lambd'payment_value': 'review_score': l
                                                        })

# get only payment value > 0
RFMTS = RFMTS.query('payment_value > 0')

# rename columns to
RFMTS.rename(columns={  'order_purchase_timestamp': 'Recency',
                        'order_id': 'Frequency',
                        'payment_value': 'Monetary',
                    'review_score': 'Satisfaction'}, inplace=True)

RFMTS
```

Out[13]:

customer_unique_id	Recency	Frequency	Monetary	Satisfaction
0000366f3b9a7992bf8c76cfdf3221e2	111	1	141.90	5.0
0000b849f77a49e4a4ce2b2a4ca5be3f	114	1	27.19	4.0
0000f46a3911fa3c0805444483337064	536	1	86.22	3.0
0000f6ccb0745a6a4b88665a16c9f078	320	1	43.62	4.0
0004aac84e0df4da2b147fca70cf8255	287	1	196.89	5.0
...	...	...	...	...
fffcf5a5ff07b0908bd4e2dbc735a684	446	1	2067.42	5.0
fffea47cd6d3cc0a88bd621562a9d061	261	1	84.58	4.0
ffff371b4d645b6ecea244b27531430a	567	1	112.46	5.0
ffff5962728ec6157033ef9805bacc48	118	1	133.69	5.0
ffffd2657e2aad2907e67c3e9daecbeb	483	1	71.56	5.0

92754 rows × 4 columns

In [14]:

```
# calculate interpurchase-Time
RFMTS["Shopping_Cycle"] = data_delivered.groupby('customer_unique_id').agg
                                                        {'order_purchase_timestamp

# cal T feature
RFMTS["Interpurchase_Time"] = (RFMTS["Shopping_Cycle"] / RFMTS["Frequency"
```

In [15]:
```
RFMTS.Interpurchase_Time.value_counts(1) * 100
```

Out[15]:
```
1      97.920305
4       0.057140
2       0.056062
5       0.049594
7       0.046359
          ...
203     0.001078
199     0.001078
194     0.001078
291     0.001078
206     0.001078
Name: Interpurchase_Time, Length: 233, dtype: float64
```

In [16]:
```
RFMTS = RFMTS.drop(['Shopping_Cycle'], axis = 1)
RFMTS
```

Out[16]:

customer_unique_id	Recency	Frequency	Monetary	Satisfaction	Interpurcha
0000366f3b9a7992bf8c76cfdf3221e2	111	1	141.90	5.0	
0000b849f77a49e4a4ce2b2a4ca5be3f	114	1	27.19	4.0	
0000f46a3911fa3c0805444483337064	536	1	86.22	3.0	
0000f6ccb0745a6a4b88665a16c9f078	320	1	43.62	4.0	
0004aac84e0df4da2b147fca70cf8255	287	1	196.89	5.0	
...	...	...	...	...	
fffcf5a5ff07b0908bd4e2dbc735a684	446	1	2067.42	5.0	
fffea47cd6d3cc0a88bd621562a9d061	261	1	84.58	4.0	
ffff371b4d645b6ecea244b27531430a	567	1	112.46	5.0	
ffff5962728ec6157033ef9805bacc48	118	1	133.69	5.0	
ffffd2657e2aad2907e67c3e9daecbeb	483	1	71.56	5.0	

92754 rows × 5 columns

In [17]:
```
RFMTS.describe()
```

Out[17]:

	Recency	Frequency	Monetary	Satisfaction	Interpurchase_Time
count	92754.000000	92754.000000	92754.000000	92754.000000	92754.000000
mean	236.789907	1.033174	165.641621	4.153708	2.228863
std	152.593047	0.208416	226.497851	1.280061	11.765354
min	0.000000	1.000000	9.590000	1.000000	1.000000
25%	113.000000	1.000000	63.090000	4.000000	1.000000
50%	218.000000	1.000000	107.825000	5.000000	1.000000
75%	345.000000	1.000000	182.910000	5.000000	1.000000
max	694.000000	15.000000	13664.080000	5.000000	305.000000

In [18]:
```
RFMTS.reset_index().to_csv('RFMTS_data.csv', index=False)
```

IV. Split sample dataset

In [20]:
```
# read data
data = pd.read_csv('RFMTS_data.csv')df =
data

# filter with frequency -- 1
data = data[data.Frequency == 1]
```

Out[20]:

	customer_unique_id	Recency	Frequency	Monetary	Satisfaction	Inter
0	0000366f3b9a7992bf8c76cfdf3221e2	111	1	141.90	5.0	
1	0000b849f77a49e4a4ce2b2a4ca5be3f	114	1	27.19	4.0	
2	0000f46a3911fa3c0805444483337064	536	1	86.22	3.0	
3	0000f6ccb0745a6a4b88665a16c9f078	320	1	43.62	4.0	
4	0004aac84e0df4da2b147fca70cf8255	287	1	196.89	5.0	
...	...	...	...	...	...	
92749	fffcf5a5ff07b0908bd4e2dbc735a684	446	1	2067.42	5.0	
92750	fffea47cd6d3cc0a88bd621562a9d061	261	1	84.58	4.0	
92751	ffff371b4d645b6ecea244b27531430a	567	1	112.46	5.0	
92752	ffff5962728ec6157033ef9805bacc48	118	1	133.69	5.0	
92753	ffffd2657e2aad2907e67c3e9daecbeb	483	1	71.56	5.0	

92754 rows × 6 columns

In [21]:
```
## split data with 95% F = 1
cus_lst = data.customer_unique_id.to_list()
random_sample = random.sample(cus_lst, int(len(cus_lst) * 0.95))

filtered_df = df[~df['customer_unique_id'].isin(random_sample)]
filtered_df.head()
```

Out[21]:

	customer_unique_id	Recency	Frequency	Monetary	Satisfaction	Interpu
29	0011c98589159d6149979563c504cb21	389	1	117.94	5.0	
32	0015752e079902b12cd00b9b7596276b	26	1	74.82	5.0	
34	00191a9719ef48ebb5860b130347bf33	497	1	58.86	3.0	
58	002ae492472e45ad6ebeb7a625409392	324	1	218.66	3.0	
80	0036b4a3d09ad551a5188c2e374da402	384	1	162.77	4.0	

In [22]:
```
filtered_df.to_csv('RFMTS_data1.csv', index=False)
# extract data info new file
```

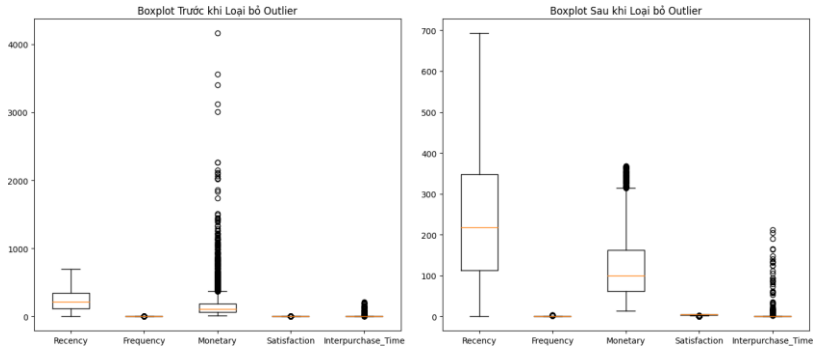
23123

In [23]:
```
df = pd.read_csv('RFMTS_data1.csv')
df.head()
```

Out[23]:

	customer_unique_id	Recency	Frequency	Monetary	Satisfaction	Interpur
0	0011c98589159d6149979563c504cb21	389	1	117.94	5.0	
1	0015752e079902b12cd00b9b7596276b	26	1	74.82	5.0	
2	00191a9719ef48ebb5860b130347bf33	497	1	58.86	3.0	
3	002ae492472e45ad6ebeb7a625409392	324	1	218.66	3.0	
4	0036b4a3d09ad551a5188c2e374da402	384	1	162.77	4.0	

In [24]:
```
# boxplot graph before remove outliers
data = df[['Recency','Frequency','Monetary','Satisfaction','Interpurchase_
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
plt.boxplot(data.values, labels=data.columns)
plt.title("Boxplot Trước khi Loại bỏ Outlier")

# remove outlier R, M attribute with quantile range
for column in ['Recency', 'Monetary']:Q1 =
    data[column].quantile(0.25) Q3 =
    data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    data = data[(data[column] >= lower_bound) & (data[column] <= upper_bou

# box plot graph after remove outlier
plt.subplot(1, 2, 2)
plt.boxplot(data.values, labels=data.columns)
plt.title("Boxplot Sau khi Loại bỏ Outlier")
plt.tight_layout()
plt.show()
```



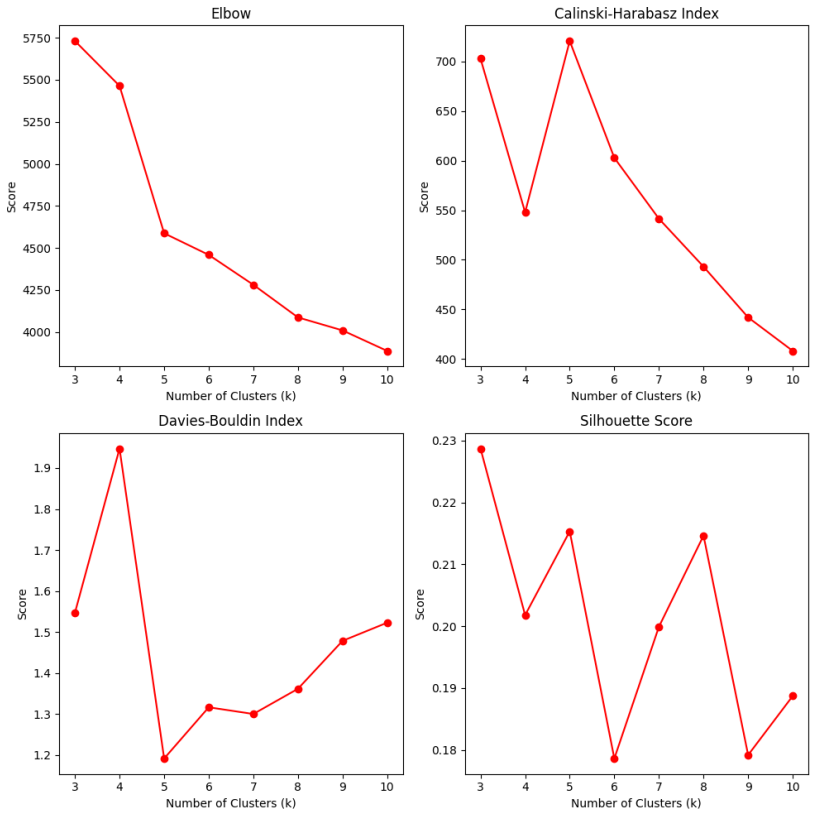
In [25]:
```
X = data.values
# assign data matrix to X variable

scaler = StandardScaler()
X = scaler.fit_transform(X)
# normalize data by z-score method
```

1. Clustering with K-mediods

In [26]:
```
k_values = range(3, 11)

# list storage value of methods
inertia_values = []
calinski_scores = []
davies_scores = []
silhouette_scores = []

for k in k_values:
    kmedoids = KMedoids(n_clusters=k, random_state=42)
    kmedoids.fit(X)
    inertia_values.append(kmedoids.inertia_)
    # calculate inertia values for elbow methods

    labels = kmedoids.labels_
    calinski_score = calinski_harabasz_score(X, labels)
    calinski_scores.append(calinski_score)
    davies_score = davies_bouldin_score(X, labels)
    davies_scores.append(davies_score)
    silhouette_score_value = silhouette_score(X, labels)
    silhouette_scores.append(silhouette_score_value)
```

In [27]:

```
# create graph for visualize evaluation metrics
fig, axes = plt.subplots(2, 2, figsize=(10, 10))

axes[0, 0].plot(range(3, 11), inertia_values, marker='o', color='red')axes[0,
0].set_title('Elbow')
axes[0, 0].set_xlabel('Number of Clusters (k)')axes[0,
0].set_ylabel('Score')

axes[0, 1].plot(range(3, 11), calinski_scores, marker='o', color='red')axes[0,
1].set_title('Calinski-Harabasz Index')
axes[0, 1].set_xlabel('Number of Clusters (k)')axes[0,
1].set_ylabel('Score')

axes[1, 0].plot(range(3, 11), davies_scores, marker='o', color='red')axes[1,
0].set_title('Davies-Bouldin Index')
axes[1, 0].set_xlabel('Number of Clusters (k)')axes[1,
0].set_ylabel('Score')

axes[1, 1].plot(range(3, 11), silhouette_scores, marker='o', color='red')axes[1,
1].set_title('Silhouette Score')
axes[1, 1].set_xlabel('Number of Clusters (k)')axes[1,
1].set_ylabel('Score')

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



In [28]:
```
n_clusters = 5
# choose the optimal number k

# detail clustering with optimal k
kmedoids = KMedoids(n_clusters=n_clusters, random_state=42)
kmedoids.fit(X)
labels = kmedoids.labels_

silhouette = silhouette_score(X, labels)
davies_bouldin = davies_bouldin_score(X, labels)
calinski_harabasz = calinski_harabasz_score(X, labels)

# create df for storage evaluation metrics
evaluation_df = pd.DataFrame(data=[[silhouette, davies_bouldin, calinski_h
                                    columns=['Silhouette Score', 'Davies-Bouldin

# show results
```

Out[28]:

	Silhouette Score	Davies-Bouldin Index	Calinski-Harabasz Index
0	0.215332	1.191895	720.704168

In [29]:
```
RFMTS_Segmentation = data.assign(Cluster=labels)
# assign cluster for each point
RFMTS_Segmentation
```

Out[29]:

	Recency	Frequency	Monetary	Satisfaction	Interpurchase_Time	Cluster
0	389	1	117.94	5.0	1	3
1	26	1	74.82	5.0	1	4
2	497	1	58.86	3.0	1	3
3	324	1	218.66	3.0	1	1
4	384	1	162.77	4.0	1	3
...	...	...	...	...	...	...
4633	141	1	188.12	1.0	1	0
4634	222	1	51.75	5.0	1	2
4635	174	1	85.37	5.0	1	2
4636	217	1	226.90	4.0	1	1
4637	244	1	265.80	4.0	1	1

4290 rows × 6 columns

In [30]:
```
RFMTS_Segmentation['Cluster'].value_counts()
```

Out[30]:
```
4    952
2    874
3    851
1    841
0    772
Name: Cluster, dtype: int64
```

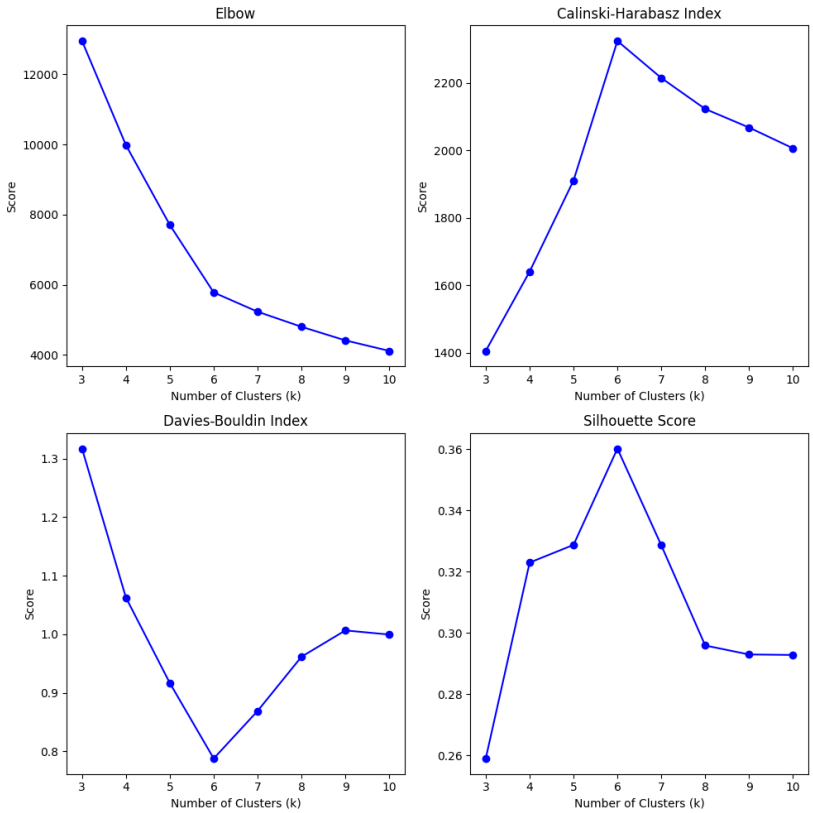
2. Test K-means

In [31]:
```
X_kmeans = X
```

In [32]:
```
k_values = range(3, 11)

# list storage value of methods
inertia_values = []
calinski_scores = []
davies_scores = []
silhouette_scores = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_kmeans)
    inertia_values.append(kmeans.inertia_)
    # calculate inertia values for elbow methods

    labels = kmeans.labels_
    calinski_score = calinski_harabasz_score(X_kmeans, labels)
    calinski_scores.append(calinski_score)
    davies_score = davies_bouldin_score(X_kmeans, labels)
    davies_scores.append(davies_score)
    silhouette_score_value = silhouette_score(X_kmeans, labels)
    silhouette_scores.append(silhouette_score_value)
```

In [33]:

```
# create graph for visualize evaluation metrics
fig, axes = plt.subplots(2, 2, figsize=(10, 10))

axes[0, 0].plot(range(3, 11), inertia_values, marker='o', color='blue')axes[0,
0].set_title('Elbow')
axes[0, 0].set_xlabel('Number of Clusters (k)')axes[0,
0].set_ylabel('Score')

axes[0, 1].plot(range(3, 11), calinski_scores, marker='o', color='blue')axes[0,
1].set_title('Calinski-Harabasz Index')
axes[0, 1].set_xlabel('Number of Clusters (k)')axes[0,
1].set_ylabel('Score')

axes[1, 0].plot(range(3, 11), davies_scores, marker='o', color='blue')axes[1,
0].set_title('Davies-Bouldin Index')
axes[1, 0].set_xlabel('Number of Clusters (k)')axes[1,
0].set_ylabel('Score')

axes[1, 1].plot(range(3, 11), silhouette_scores, marker='o', color='blue')axes[1,
1].set_title('Silhouette Score')
axes[1, 1].set_xlabel('Number of Clusters (k)')axes[1,
1].set_ylabel('Score')

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



In [34]:
```
n_clusters = 5 # choose the optimal number k

# detail clustering with optimal k
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
kmeans.fit(X_kmeans)
labels = kmeans.labels_

silhouette = silhouette_score(X_kmeans, labels)
davies_bouldin = davies_bouldin_score(X_kmeans, labels)
calinski_harabasz = calinski_harabasz_score(X_kmeans, labels)

# create df for storage evaluation metrics
evaluation_df = pd.DataFrame(data=[[silhouette, davies_bouldin, calinski_h
                                    columns=['Silhouette Score', 'Davies-Bouldin

# show results
```

Out[34]:

	Silhouette Score	Davies-Bouldin Index	Calinski-Harabasz Index
0	0.328756	0.916565	1910.610218

In [35]:
```
RFMTS_Segmentation = data.assign(Cluster=labels)
# assign cluster for each point
RFMTS_Segmentation.head()
```

Out[35]:

	Recency	Frequency	Monetary	Satisfaction	Interpurchase_Time	Cluster
0	389	1	117.94	5.0	1	2
1	26	1	74.82	5.0	1	4
2	497	1	58.86	3.0	1	2
3	324	1	218.66	3.0	1	0
4	384	1	162.77	4.0	1	2

In [36]:
```
RFMTS_Segmentation['Cluster'].value_counts()
```

Out[36]:
```
4    2000
2    1476
0     723
3      69
1      22
Name: Cluster, dtype: int64
```

3. Kiểm định MANOVA

In [37]:
```
# testing MANOVA
manova = MANOVA.from_formula('Recency + Frequency + Monetary + Interpurcha
                             data=RFMTS_Segmentation)
print(manova.mv_test())
```

```
                    Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept         Value  Num DF   Den DF    F Value   Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0569 5.0000 4284.0000 14208.2021 0.0000
         Pillai's trace  0.9431 5.0000 4284.0000 14208.2021 0.0000
 Hotelling-Lawley trace 16.5829 5.0000 4284.0000 14208.2021 0.0000
    Roy's greatest root 16.5829 5.0000 4284.0000 14208.2021 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
          Cluster         Value  Num DF   Den DF    F Value  Pr > F
----------------------------------------------------------------
            Wilks' lambda 0.2882 5.0000 4284.0000 2115.7815 0.0000
           Pillai's trace 0.7118 5.0000 4284.0000 2115.7815 0.0000
   Hotelling-Lawley trace 2.4694 5.0000 4284.0000 2115.7815 0.0000
      Roy's greatest root 2.4694 5.0000 4284.0000 2115.7815 0.0000
================================================================
```

4. Mô tả cụm trong Kmeans

In [38]:
```
print('Số lượng từng cụm: ')
RFMTS_Segmentation.groupby("Cluster").size()
```

```
Số lượng từng cụm:
```

Out[38]:
```
Cluster
0     723
1      22
2    1476
3      69
4    2000
dtype: int64
```

In [39]:
```
# calculate the means of each attribute of each cluster
clusters = RFMTS_Segmentation.groupby(['Cluster']).agg({'Recency':
            lambda x: x.mean(),
            'Frequency': lambda x: x.mean(),
            'Monetary': lambda x: x.mean(),
            'Interpurchase_Time': lambda x: x.mean(),
            'Satisfaction': lambda x: x.mean(),})

# count frequencies of each cluster
clusters['Value_counts'] = RFMTS_Segmentation.groupby(['Cluster'])['Clustclusters.columns
= ['Recency_mean', 'Frequency_mean', 'Monetary_mean',
                              'InterpurchaseTime_mean','Satisfaction_mean','Value_co
clusters.head(10)
```

Out[39]:

Cluster	Recency_mean	Frequency_mean	Monetary_mean	InterpurchaseTime_mean	Satisfac
0	227.629322	1.000000	122.913956	1.000000	
1	148.681818	2.090909	196.849091	126.136364	
2	396.759485	1.000000	121.900786	1.000000	
3	257.115942	2.072464	201.488406	12.913043	
4	124.068500	1.000000	114.037645	1.000000	

In [40]:
```
# ranking on each feature on global

clusters['RankR'] = clusters['Recency_mean'].rank(ascending=True)
clusters['RankF'] = clusters['Frequency_mean'].rank(ascending=True)
clusters['RankM'] = clusters['Monetary_mean'].rank(ascending=True)
clusters['RankT'] = clusters['InterpurchaseTime_mean'].rank(ascending=True
clusters['RankS'] = clusters['Satisfaction_mean'].rank(ascending=True)
```

In [41]:
```
cluster_rank = clusters[['RankR','RankF','RankM','RankT','RankS']]
# get sub target dataframe
```

In [42]:
```
# Visualize percent of the customer in each cluster

palette=sns.color_palette('coolwarm')
explodes = [0.25, 0.25, 0.25, 0.25, 0.25]
RFMTS_Segmentation["Cluster"].value_counts(sort=False).plot.pie(colors=pal
                                                                textprops={'fontsize': 10
                                                                autopct='%4.1f%%',
                                                                startangle= 50,
                                                                radius = 1,
                                                                shadow = False)
                                                               # explode = explodes)
plt.ylabel("")
plt.title('Biểu đồ phần trăm khách hàng trong mỗi cụm')plt.show()
```