

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THÔNG THÔNG TIN



ĐỒ ÁN MÔN HỌC
DỮ LIỆU LỚN

ĐỀ TÀI
ỨNG DỤNG THUẬT TOÁN PHÂN CỤM
NHẰM PHÂN TÍCH DỮ LIỆU KHÁCH HÀNG

Giảng viên: ThS. Nguyễn Hồ Duy Tri

Lớp: IS405.P11.HTCL

Thành viên: Nhóm 3

Trần Thị Kim Anh – 21520596

Hồ Quang Lâm – 21521049

Lê Thị Lê Trúc – 21521586

Lê Minh Chánh – 21521882

Thành phố Hồ Chí Minh, tháng 12 năm 2024

LỜI CẢM ƠN

Đầu tiên, nhóm chúng em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới giảng viên Nguyễn Hồ Duy Tri – người đã giảng dạy và chia sẻ rất nhiều kiến thức cũng như các ví dụ thực tiễn trong các bài giảng. Thầy đã hướng dẫn cho chúng em làm bài tập, sửa chữa và đóng góp nhiều ý kiến quý báu giúp chúng em hoàn thành tốt báo cáo môn học của mình.

Bộ môn Dữ liệu lớn là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Tuy nhiên, do vốn kiến thức chuyên môn còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ. Mặc dù chúng em đã cố gắng hết sức nhưng chắc chắn bài báo cáo khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, chúng em rất mong nhận được sự góp ý, chỉ bảo thêm của thầy nhằm hoàn thiện những kiến thức của mình để nhóm chúng em có thể dùng làm hành trang thực hiện tiếp các đề tài khác trong tương lai cũng như là trong học tập và làm việc sau này.

Một lần nữa, nhóm xin gửi đến thầy, bạn bè lời cảm ơn đặc biệt chân thành và tốt đẹp nhất!

Thành phố Hồ Chí Minh, tháng 12 năm 2024

Nhóm sinh viên thực hiện

Trần Thị Kim Anh

Hồ Quang Lâm

Lê Thị Lê Trúc

Lê Minh Chánh

NHẬN XÉT CỦA GIẢNG VIÊN

MỤC LỤC

LỜI CẢM ƠN	2
NHẬN XÉT CỦA GIẢNG VIÊN	3
Phần 1. GIỚI THIỆU ĐỀ TÀI.....	9
1.1. Lý do chọn đề tài	9
1.2. Tổng quan về dữ liệu.....	10
1.2.1. Nguồn dữ liệu.....	10
1.2.2. Mô tả bộ dữ liệu.....	10
1.2.3. Thống kê	11
1.3. Mô tả bài toán.....	21
Phần 2. PHƯƠNG PHÁP VÀ THUẬT TOÁN SỬ DỤNG	22
2.1. Tiền xử lý dữ liệu.....	22
2.1.1. StringIndexer	22
2.1.2. Z-score	23
2.1.3. PCA (Principal Component Analysis).....	24
2.1.3.1. Vì sao cần giảm chiều dữ liệu?	24
2.1.3.2. Các bước thực hiện thuật toán PCA	26
2.2. Khai thác dữ liệu	28
2.2.1. Khoảng cách Euclidean.....	28
2.2.1.1. Công thức khoảng cách Euclidean một chiều	28
2.2.1.2. Công thức khoảng cách Euclidean hai chiều.....	28
2.2.1.3. Công thức khoảng cách Euclidean cho bất kì kích thước	29
2.2.2. Phương sai.....	29
2.2.3. Thuật toán K-Means	30

2.2.3.1. Giới thiệu	30
2.2.3.2. K-means hoạt động như thế nào?	32
2.2.3.3. Cách chọn k tối ưu - Elbow	35
2.2.4. Thuật toán DHC (Divisive Hierachical Clustering)	37
2.2.4.1. Hierarchical Clustering (phân cụm phân cấp)	37
2.2.4.2. Divisive Hierachical Clustering	39
2.3. Các phương pháp đánh giá	41
2.3.1. Silhouette Coefficient	41
2.3.2. Calinski-Harabasz Index	42
Phần 3. TIỀN XỬ LÝ DỮ LIỆU	43
3.1. Các thư viện được sử dụng	43
3.2. Đọc dữ liệu đầu vào	45
3.3. Kiểm tra các biến phân loại trong Dataframe	47
3.4. Xóa các cột bị thiếu hoặc dư thừa	48
3.5. Xử lý một số giá trị ngoại lai	50
3.6. Kiểm tra độ tương quan của bộ Dataset	52
3.7. Mã hóa các đặc trưng (features) phân loại bằng StringIndexer	54
3.8. Chuẩn hóa (scale) các đặc trưng bằng Z – Score	55
3.9. Thực hiện giảm chiều dữ liệu bằng PCA	56
Phần 4. THUẬT TOÁN KHAI THÁC DỮ LIỆU	59
4.1. Đặc tả thuật toán	59
4.1.1. Thuật toán K – Means	59
4.1.2. Thuật toán Divisive Hierarchical Clustering	60
4.2. Thực hiện thuật toán	60

4.2.1.	Thuật toán K – means	60
4.2.1.1.	Tạo các hàm tính toán trong K – means	60
4.2.1.2.	Tạo hàm chạy thuật toán K – Means	61
4.2.2.	Thuật toán Divisive Hierarchical Clustering.....	67
4.2.2.1.	Tạo các hàm tính toán trong DHC.....	67
4.2.2.2.	Tạo hàm chạy thuật toán DHC	68
4.3.	VISUALIZE.....	70
4.3.1.	Thuật toán K – Means.....	70
4.3.2.	Thuật toán Divisive Hierarchical Clustering.....	78
4.4.	Bảng so sánh	86
Phần 5.	KẾT LUẬN.....	87
5.1.	Kết quả đạt được	87
5.2.	Hạn chế.....	87
5.3.	Hướng phát triển	88
TÀI LIỆU THAM KHẢO		89
PHÂN CÔNG CÔNG VIỆC		90

DANH MỤC HÌNH ẢNH

<i>Hình 1.1. Biểu đồ phân phối giữa các khách hàng đã ngưng và còn sử dụng thẻ</i>	...14
<i>Hình 1.2. Biểu đồ phân phối giới tính khách hàng</i>	14
<i>Hình 1.3. Biểu đồ phân phối trình độ đại học của khách hàng</i>	15
<i>Hình 1.4. Biểu đồ phân phối tình trạng hôn nhân của khách hàng</i>	16
<i>Hình 1.5. Biểu đồ phân phối mức thu nhập khách hàng</i>	17
<i>Hình 1.6. Biểu đồ phân phối loại thẻ sử dụng</i>	18
<i>Hình 1.7. Biểu đồ phân phối số người phụ thuộc trong gia đình</i>	18
<i>Hình 1.8. Biểu đồ phân phối số tháng không hoạt động trong 12 tháng qua</i>	19
<i>Hình 1.9. Biểu đồ phân phối số lần khách đã liên lạc trong 12 tháng qua</i>	20
<i>Hình 1.10. Biểu đồ phân phối của từng đặc tính</i>	21
<i>Hình 2.1. Ví dụ về tập dữ liệu huyền thoại Iris</i>	25
<i>Hình 2.2. Kết quả bộ dữ liệu ví dụ sau khi thực hiện PCA</i>	25
<i>Hình 2.3. Ví dụ về phân cụm dữ liệu</i>	31
<i>Hình 2.4. Minh họa cho việc phân chia lãnh hải nếu có 5 đảo (chấm đen)</i>	32
<i>Hình 2.5. Minh họa khởi tạo tâm cho cụm</i>	33
<i>Hình 2.6. Minh họa gán điểm dữ liệu cho cụm gần nhất</i>	33
<i>Hình 2.7. Minh họa khởi tạo lại tâm từ các cụm đã chia</i>	34
<i>Hình 2.8. Minh họa kết quả sau khi tìm được tâm tối ưu – Phân cụm thành công</i>	35
<i>Hình 2.9. Biểu đồ kết quả của phương pháp Elbow</i>	36
<i>Hình 2.10. Minh họa kết quả thuật toán Silhouette</i>	37
<i>Hình 2.11. Đồ thị minh họa biểu diễn phân cụm phân cấp (Dendrogram)</i>	38
<i>Hình 2.12. Minh họa các bước xử lý của thuật toán DHC</i>	40
<i>Hình 3.1. Sơ đồ các bước tiền xử lý dữ liệu</i>	43
<i>Hình 4.1. Sơ đồ đặc tả các bước chạy thuật toán K-Means</i>	59
<i>Hình 4.2. Sơ đồ đặc tả các bước chạy thuật toán DHC</i>	60
<i>Hình 4.3. Kết quả Elbow Method trên bộ dữ liệu đề tài</i>	63
<i>Hình 4.4. Trực quan 3D phân cụm bằng thuật toán K-Means trên bộ dữ liệu đề tài</i>	66

Hình 4.5. Trực quan 3D kết quả phân cụm thuật toán DHC trên bộ dữ liệu đề tài .70

<i>Bảng 1. Bảng mô tả chi tiết các thuộc tính bộ dữ liệu</i>	10
<i>Bảng 2. Bảng thống kê trên từng thuộc tính</i>	11
<i>Bảng 3. Bảng so sánh kết quả của 2 thuật toán phân cụm</i>	86

Phần 1. GIỚI THIỆU ĐỀ TÀI

1.1. Lý do chọn đề tài

Trong bối cảnh ngành ngân hàng và tài chính ngày càng trở nên cạnh tranh, việc hiểu rõ và phân khúc khách hàng đã trở thành một yếu tố thiết yếu giúp các doanh nghiệp phát triển bền vững. Đề tài "**Phân Tích Phân Cụm Khách Hàng Thủ Tín Dụng**" được nhóm chúng tôi lựa chọn không chỉ vì tính ứng dụng cao mà còn vì những lợi ích thiết thực mà nó mang lại. Ngành ngân hàng đang chuyển mình mạnh mẽ với việc tối ưu hóa trải nghiệm khách hàng, và việc phân tích phân khúc khách hàng sẽ giúp doanh nghiệp xây dựng các chiến lược tiếp thị và dịch vụ phù hợp với từng nhóm đối tượng. Bằng cách hiểu rõ hơn về nhu cầu, sở thích và hành vi tiêu dùng của khách hàng, các doanh nghiệp có thể thiết kế các sản phẩm và dịch vụ đáp ứng tốt hơn, từ đó nâng cao sự hài lòng và giữ chân khách hàng hiệu quả hơn.

Bộ dữ liệu "**Predicting Credit Card Customer Segmentation**" mà chúng tôi sử dụng trong nghiên cứu này có quy mô lớn với 10.127 dòng và 23 thuộc tính phong phú, bao gồm thông tin nhân khẩu học, hành vi chi tiêu và mối quan hệ với nhà cung cấp thẻ. Những thông tin này không chỉ cung cấp cái nhìn sâu sắc về đặc điểm của khách hàng mà còn cho phép chúng tôi áp dụng các phương pháp phân tích hiện đại, đặc biệt là thuật toán K-Means, để phân nhóm khách hàng. K-Means là một phương pháp hiệu quả trong việc phân cụm dữ liệu lớn, giúp xác định các nhóm có đặc điểm tương đồng. Kết quả từ phân tích này sẽ giúp doanh nghiệp nhận diện rõ ràng các phân khúc khách hàng khác nhau, từ đó đưa ra các kế hoạch và chiến lược tiếp thị phù hợp với từng nhóm.

Cuối cùng, thông qua việc phân tích và hiểu rõ các phân khúc khách hàng, doanh nghiệp có thể phát triển các sản phẩm và dịch vụ đáp ứng tốt hơn nhu cầu của từng nhóm. Điều này không chỉ gia tăng doanh thu mà còn tạo ra lợi thế cạnh tranh bền vững trong thị trường thẻ tín dụng. Chúng tôi tin rằng đề tài này không chỉ mang lại giá trị học thuật mà còn có ý nghĩa thực tiễn to lớn, hỗ trợ doanh nghiệp trong việc tối ưu hóa chiến lược tiếp thị và nâng cao trải nghiệm khách hàng. Chúng tôi hy vọng

rằng kết quả nghiên cứu sẽ đóng góp vào việc xây dựng các chiến lược kinh doanh hiệu quả và phát triển bền vững trong ngành ngân hàng.

1.2. Tổng quan về dữ liệu

1.2.1. Nguồn dữ liệu

- Tên bộ dữ liệu: Predicting Credit Card Customer Segmentation
- Người công bố: **The Devastator**
- Đường dẫn đến Kaggle: [Predicting Credit Card Customer Segmentation](#)
- Bộ dữ liệu gồm **10.127 dòng x 23 thuộc tính** chứa các thông tin khách hàng được thu thập từ danh mục thẻ tín dụng tiêu dùng bao gồm các thông tin nhân khẩu học, mối quan hệ với nhà cung cấp thẻ, hành vi chi tiêu và các chỉ số dự đoán từ danh mục thẻ tín dụng tiêu dùng

1.2.2. Mô tả bộ dữ liệu

Bảng 1. Bảng mô tả chi tiết các thuộc tính bộ dữ liệu

STT	Thuộc tính	Kiểu dữ liệu	Mô tả
1	CLIENTNUM	Số nguyên	Mã định danh khách hàng
2	Attrition_Flag	Boolean	Khách hàng đã rời bỏ hay chưa
3	Customer_Age	Số nguyên	Tuổi của khách hàng
4	Gender	Kí tự	Giới tính khách hàng
5	Dependent_count	Số nguyên	Số người phụ thuộc của khách hàng
6	Education_Level	Chuỗi	Trình độ học vấn
7	Marital_Status	Chuỗi	Tình trạng hôn nhân
8	Income_Category	Chuỗi	Loại thu nhập của khách hàng
9	Card_Category	Chuỗi	Loại thẻ của khách hàng
10	Months_on_book	Số nguyên	Khách hàng đã có trong sổ bao lâu
11	Total_Relationship_Count	Số nguyên	Tổng số mối quan hệ khách hàng với nhà cung cấp thẻ tín dụng
12	Months_Inactive_12_mon	Số nguyên	Số tháng khách hàng không hoạt động trong 12 tháng qua

13	Contacts_Count_12_mon	Số nguyên	Số lượng liên hệ mà khách hàng đã có trong 12 tháng qua
14	Credit_Limit	Số nguyên	Hạn mức tín dụng của khách hàng
15	Total_Revolving_Bal	Số nguyên	Tổng số dư quay vòng
16	Avg_Open_To_Buy	Số nguyên	Trung bình hạn mức tín dụng mở để mua trong 12 tháng qua
17	Total_Amt_Chng_Q4_Q1	Số thực	Thay đổi số tiền giao dịch (Q4 so với Q1)
18	Total_Trans_Amt	Số nguyên	Tổng số tiền giao dịch
19	Total_Trans_Ct	Số nguyên	Tổng số giao dịch
20	Total_Ct_Chng_Q4_Q1	Số thực	Thay đổi số lượng giao dịch (Q4 so với Q1)
21	Avg_Utilization_Ratio	Số thực	Tỷ lệ sử dụng trung bình
22	Naive_Bayes_Classifier_1	Số thực	Bộ phân loại Naive Bayes dự đoán liệu có rời bỏ hay không dựa trên các đặc điểm
23	Naive_Bayes_Classifier_2		

1.2.3. Thống kê

- Nhóm sẽ thực hiện việc trực quan hóa dữ liệu nhằm làm rõ hơn các đặc điểm của bộ dữ liệu, đồng thời thực hiện nhiệm vụ khác trong môi trường PySpark.

Bảng 2. Bảng thống kê trên từng thuộc tính

Thuộc tính	Count	Mean	Stddev	Min	Max
Clientnum	10127	7.392	3.69	708082083	828343083
Customer_Age	10127	46.326	8.017	26	73
Dependent_count	10127	2.346	1.299	0	5
Months_on_book	10127	35.928	7.986	13	56
Total_Relationship_Count	10127	3.813	1.554	1	6

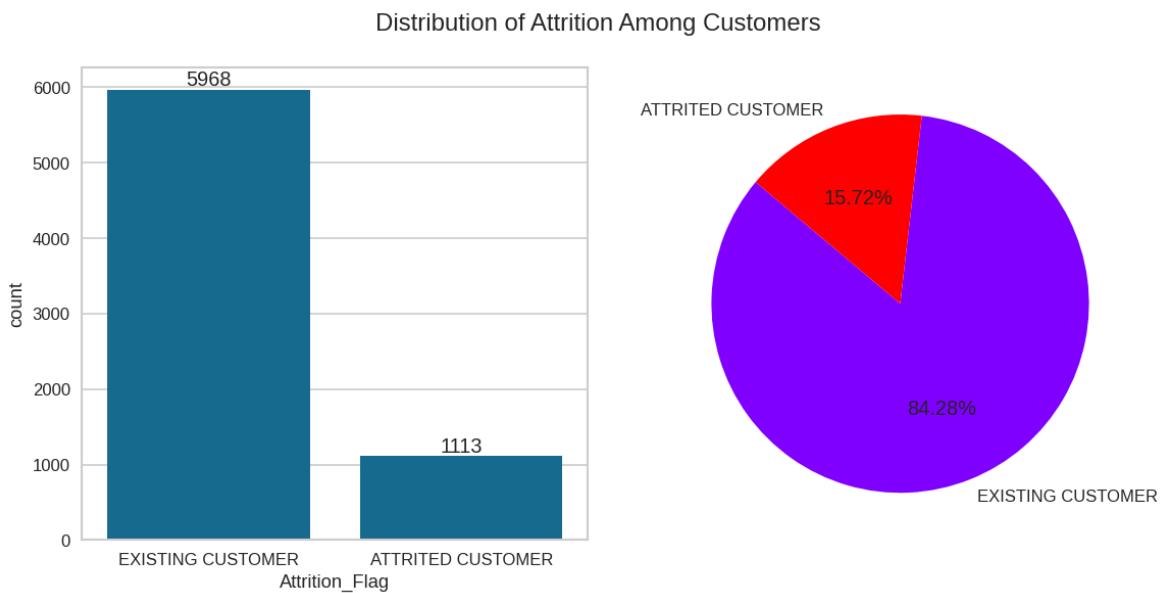
Months_Inactive _12_mon	10127	2.341	1.011	0	6
Contacts_Count _12_mon	10127	2.456	1.106	0	6
Credit_Limit	10127	8631.954	9088.777	1438.3	34516
Total_Revolving_Bal	10127	1162.814	814.987	0	2517
Avg_Open_To_Buy	10127	7469.134	9090.685	3	34516
Total_Amt_Chng _Q4_Q1	10127	0.759	0.219	0	3.397
Total_Trans_Amt	10127	4404.086	3397.13	510	18484
Total_Trans_Ct	10127	64.859	23.472	10	139
Total_Ct_Chng _Q4_Q1	10127	0.712	0.238	0	3.714
Avg_Utilization_Ratio	10127	0.275	0.276	0	0.999
Naive_Bayes _Classifier_1	10127	0.16	0.365	7.6642E-6	0.99958
Naive_Bayes _Classifier_2	10127	0.84	0.365	4.1998E-4	0.99999

⇒ Nhận xét:

- Bộ dữ liệu không chứa giá trị NULL hoặc trống, đảm bảo tính đầy đủ và độ tin cậy của thông tin.
- Độ tuổi trung bình của khách hàng được ước tính là 46 tuổi, với độ tuổi dao động từ 26 đến 73, cho thấy sự đa dạng trong nhóm khách hàng.
- Trung bình, mỗi khách hàng có khoảng 2 người phụ thuộc, cho thấy ảnh hưởng của các yếu tố gia đình đến quyết định tài chính.
- Thời gian sử dụng dịch vụ của khách hàng, hay thời gian "on books", trung bình là 36 tháng, cho thấy sự gắn bó của họ với nhà cung cấp thẻ.

-
- Mỗi khách hàng thường duy trì khoảng 4 mối quan hệ với các nhà cung cấp thẻ tín dụng, điều này phản ánh mức độ tương tác và sự lựa chọn của họ trong thị trường.
 - Hạn mức tín dụng trung bình mà các tổ chức tài chính cung cấp lên tới 8.492,77 USD. Tuy nhiên, độ lệch chuẩn lớn cho thấy sự biến động đáng kể trong hạn mức tín dụng, với khoảng giá trị trải dài từ 1.438 USD đến 34.516 USD.
 - Tổng số tiền giao dịch trung bình của một khách hàng đạt 4.394,30 USD, cho thấy mức độ chi tiêu khá cao.
 - Cuối cùng, mỗi khách hàng thực hiện trung bình 65 giao dịch, điều này cho thấy tần suất hoạt động tài chính của họ trong hệ thống.
- Khám phá các thuộc tính thuộc Customer (Attrition, Gender, Education level, Marital status, Income category, Category of card, number of dependent, number of months inactiveness, number of contacts customer)

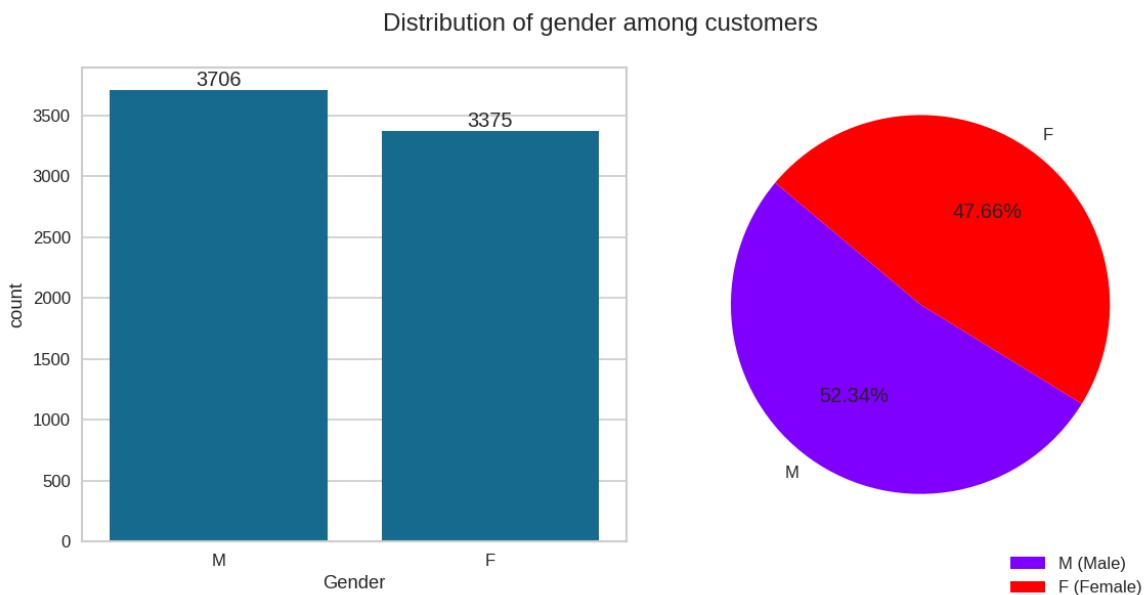
- Phân phối tỷ lệ ngưng sử dụng thẻ tín dụng giữa các khách hàng:



Hình 1.1. Biểu đồ phân phối giữa các khách hàng đã ngưng và còn sử dụng thẻ

⇒ Những biểu đồ trên đã thể hiện chính xác tỷ lệ phần trăm khách hàng ngưng sử dụng và khách hàng còn sử dụng thẻ tín dụng. Khoảng 16% khách hàng đã ngừng sử dụng thẻ tín dụng.

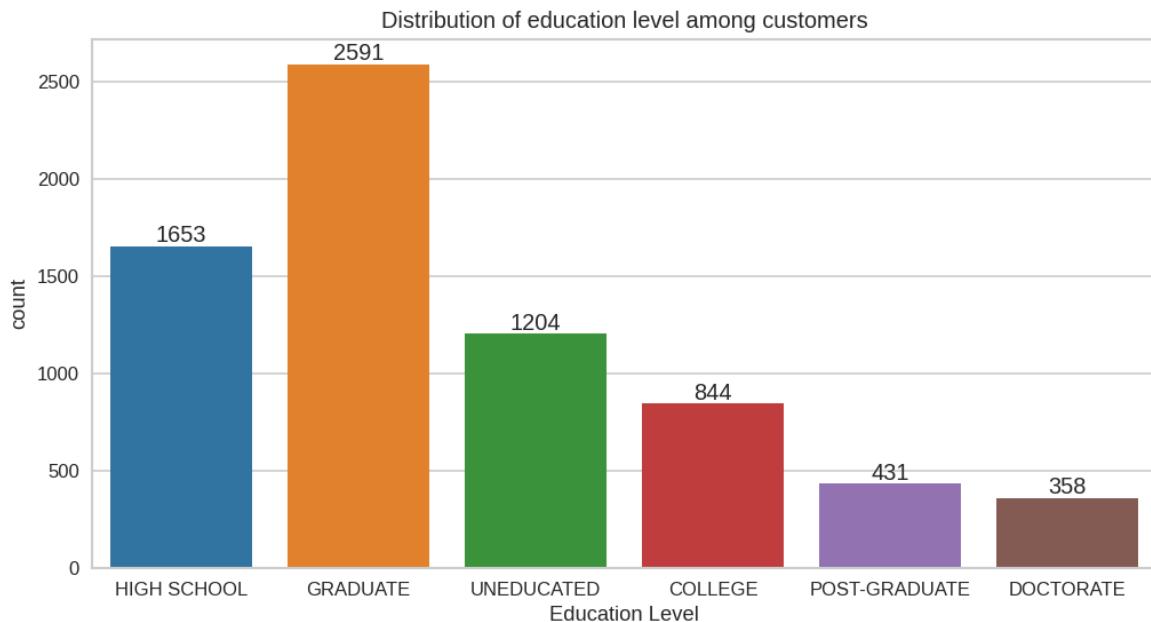
- Phân phối giới tính giữa các khách hàng:



Hình 1.2. Biểu đồ phân phối giới tính khách hàng

⇒ Chúng ta có được một tỷ lệ khá cân bằng dựa trên giới tính của khách hàng.

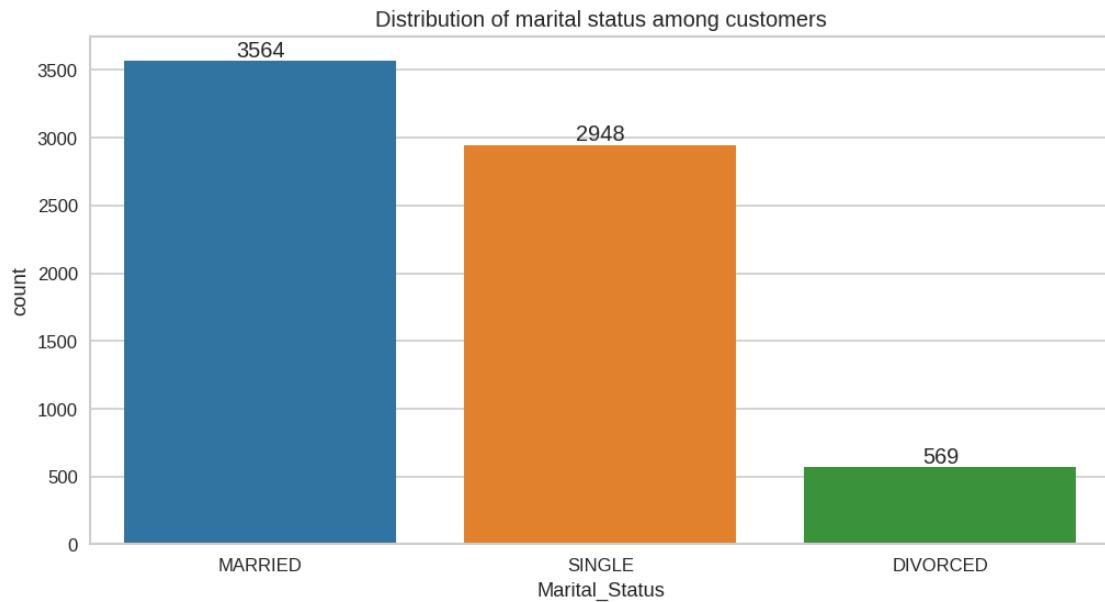
- Phân phối trình độ học vấn giữa các khách hàng:



Hình 1.3. Biểu đồ phân phối trình độ đại học của khách hàng

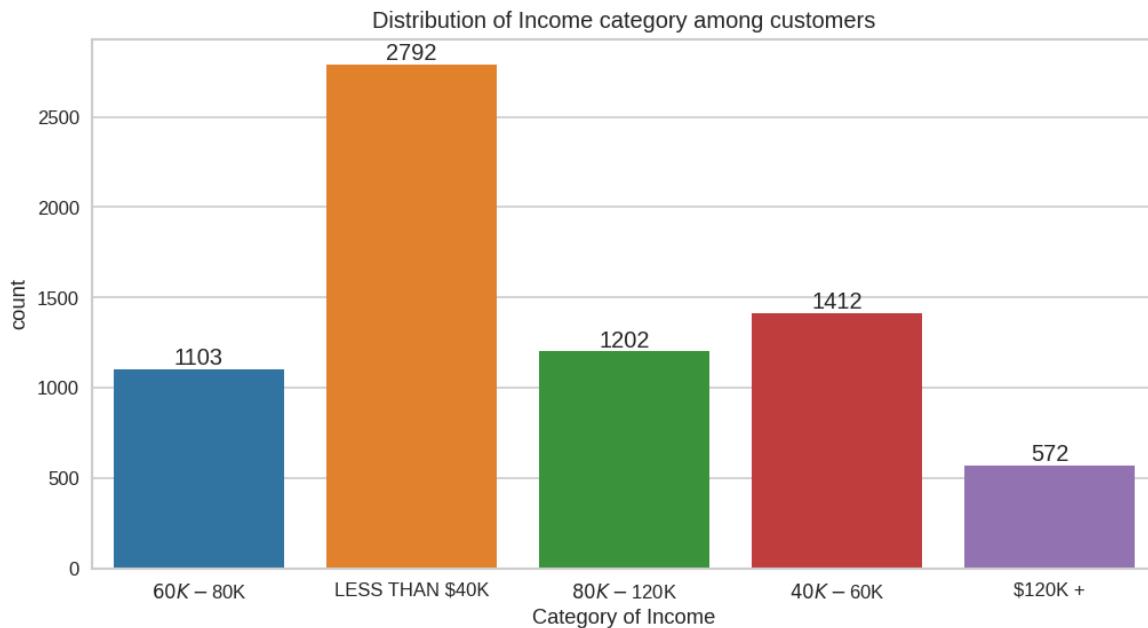
⇒ Phân phối trình độ học vấn cho thấy nhóm "Graduate" chiếm ưu thế với 2.591 người, tiếp theo là "High School" với 1.653 người, trong khi các nhóm còn lại có số lượng thấp hơn. Điều này cho thấy phần lớn khách hàng có trình độ học vấn từ tốt nghiệp trở lên.

- Phân phối tình trạng hôn nhân giữa các khách hàng:



Hình 1.4. Biểu đồ phân phối tình trạng hôn nhân của khách hàng

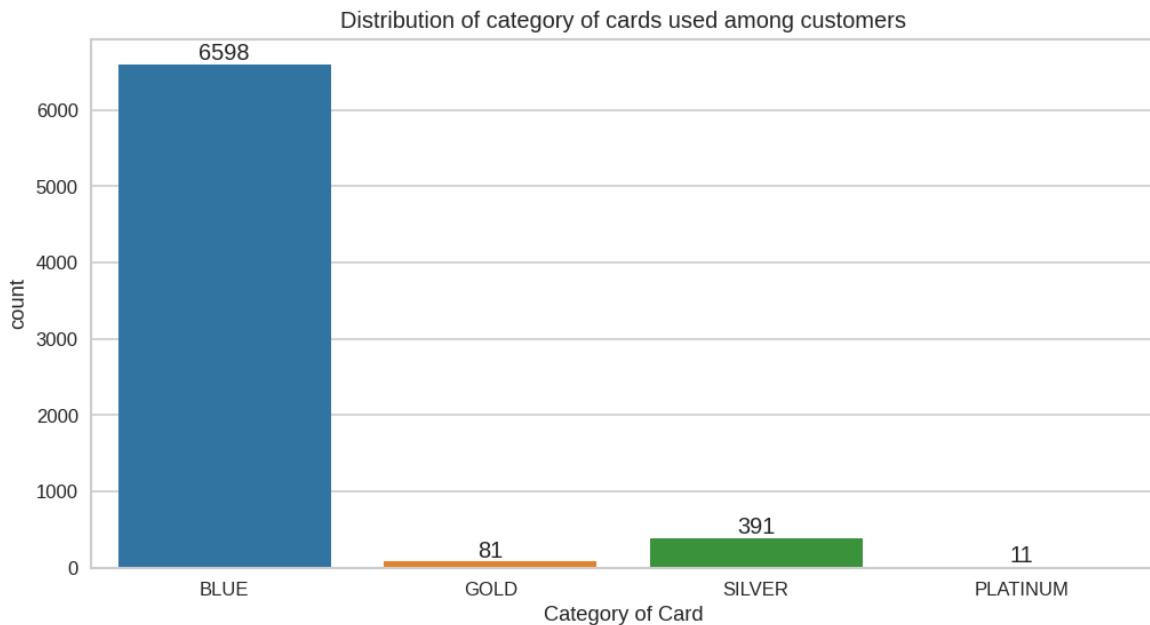
- ⇒ Biểu đồ cho thấy phân bố tình trạng hôn nhân của khách hàng. Nhóm "Married" (Đã kết hôn) chiếm ưu thế với 3.564 người, trong khi nhóm "Single" (Độc thân) có 2.948 người. Nhóm "Divorced" (Ly hôn) là nhóm ít nhất với chỉ 569 người, cho thấy phần lớn khách hàng đang trong mối quan hệ hôn nhân ổn định.
- Phân phối nhóm thu nhập giữa các khách hàng:



Hình 1.5. Biểu đồ phân phối mức thu nhập khách hàng

- ⇒ Biểu đồ cho thấy phân phối nhóm thu nhập của khách hàng, với nhóm "Less than \$40K" (Dưới 40.000 USD) chiếm ưu thế nhất với 2.792 người, tiếp theo là nhóm "\$40K - \$60K" (40.000 - 60.000 USD) với 1.412 người. Các nhóm còn lại có số lượng thấp hơn, cho thấy đa số khách hàng có thu nhập dưới 40.000 USD.

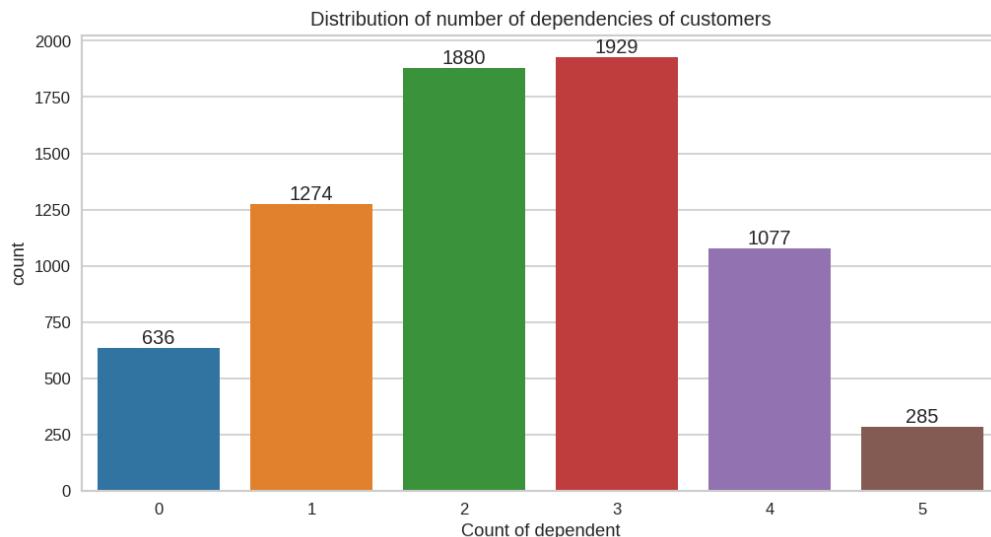
- Phân phối nhóm thẻ sử dụng giữa các khách hàng:



Hình 1.6. Biểu đồ phân phối loại thẻ sử dụng

⇒ Biểu đồ cho thấy phân bố các loại thẻ sử dụng của khách hàng, với nhóm "Blue" (Thẻ xanh) vượt trội với 6.598 thẻ. Các nhóm "Silver" (Thẻ bạc) và "Gold" (Thẻ vàng) chỉ có 391 và 81 thẻ, trong khi nhóm "Platinum" (Thẻ bạch kim) rất ít, chỉ có 11 thẻ. Điều này cho thấy hầu hết khách hàng sử dụng thẻ xanh (đây là thẻ cơ bản nhất của dịch vụ cung cấp thẻ).

- Phân phối số lượng người phụ thuộc của khách hàng:



Hình 1.7. Biểu đồ phân phối số người phụ thuộc trong gia đình

⇒ Biểu đồ cho thấy phân bố số lượng người phụ thuộc của khách hàng. Nhóm có 2 người phụ thuộc chiếm ưu thế nhất với 1.929 người, tiếp theo là nhóm có 3 người phụ thuộc với 1.880 người. Nhóm có 1 người phụ thuộc cũng khá phổ biến với 1.274 người. Các nhóm còn lại (0, 4 và 5 người phụ thuộc) có số lượng thấp hơn, cho thấy rằng đại đa số khách hàng thường có từ 2 đến 3 người phụ thuộc.

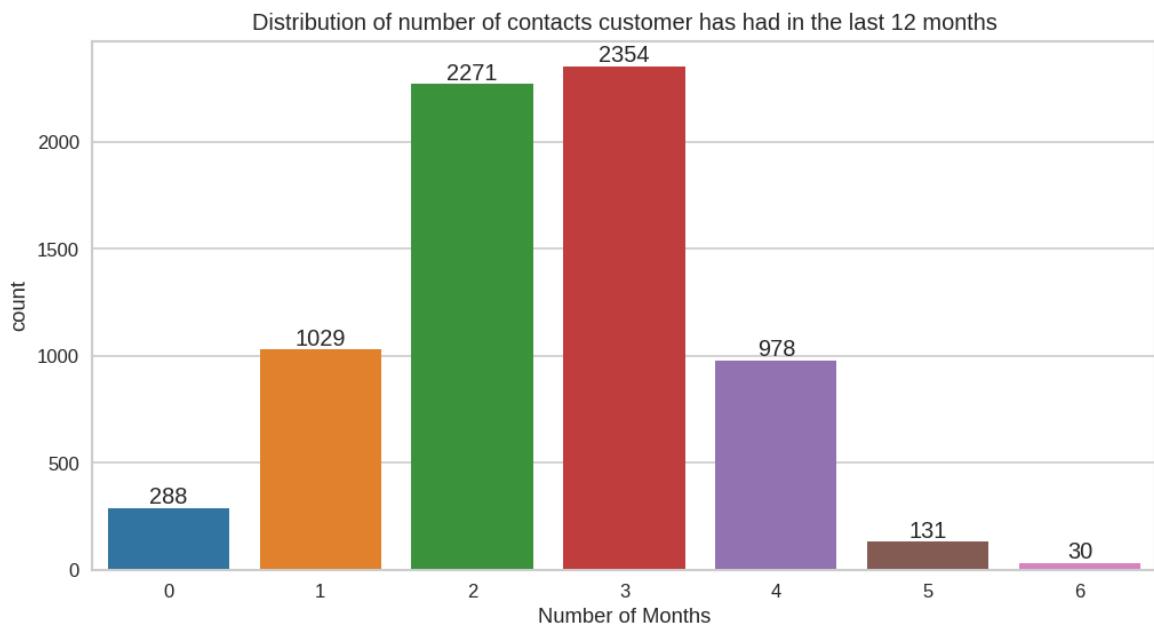
- Phân phối số tháng không hoạt động trong 12 tháng qua:



Hình 1.8. Biểu đồ phân phối số tháng không hoạt động trong 12 tháng qua

⇒ Biểu đồ cho thấy phân bố số tháng không hoạt động của khách hàng trong 12 tháng qua. Nhóm có 4 tháng không hoạt động chiếm số lượng lớn nhất với 2.724 người, tiếp theo là nhóm có 3 tháng không hoạt động với 2.320 người. Nhóm có 1 tháng không hoạt động có 1.525 người, trong khi nhóm không hoạt động (0 tháng) chỉ có 19 người. Các nhóm có 5 và 6 tháng không hoạt động có số lượng rất thấp, cho thấy rằng phần lớn khách hàng chỉ có từ 1 đến 4 tháng không hoạt động.

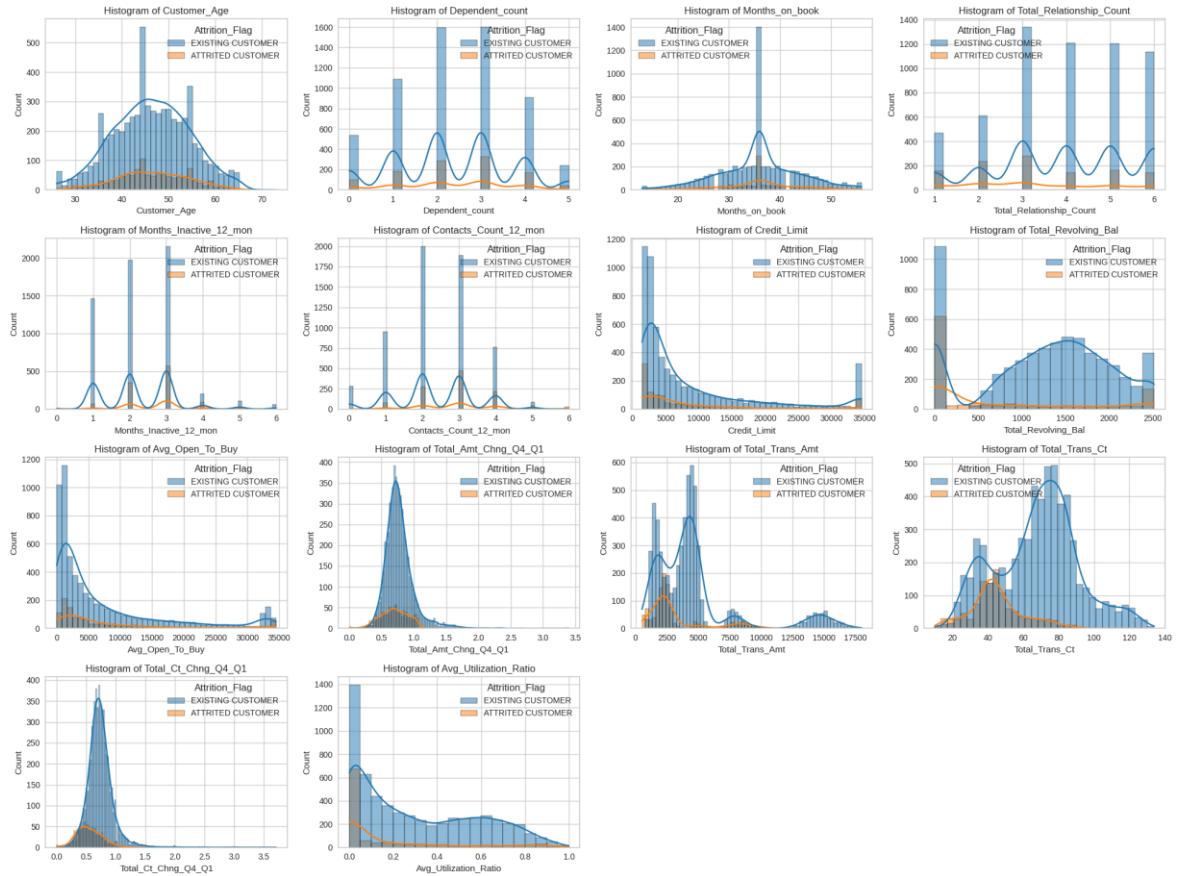
- Phân phối số lần khách hàng đã liên lạc trong 12 tháng qua:



Hình 1.9. Biểu đồ phân phối số lần khách đã liên lạc trong 12 tháng qua

⇒ Biểu đồ này cho thấy sự phân bố số lượng liên lạc mà khách hàng đã có trong 12 tháng qua. Các cột thể hiện số lượng liên lạc theo từng tháng, với tháng thứ 3 có số lượng liên lạc cao nhất (2354), tiếp theo là tháng thứ 2 (2271). Số lượng liên lạc giảm dần ở các tháng còn lại, đặc biệt là tháng thứ 5 và tháng thứ 6 với số lượng rất thấp (131 và 30). Điều này có thể chỉ ra rằng khách hàng thường xuyên liên lạc nhiều hơn trong những tháng đầu tiên.

- Giờ đây, chúng ta sẽ xem phân phối của từng đặc tính trên:



Hình 1.10. Biểu đồ phân phối của từng đặc tính

1.3. Mô tả bài toán

Với bộ dữ liệu như trên, trong phạm vi đồ án, nhóm sẽ tiến hành thực hiện phương pháp phân cụm bằng thuật toán K-Means và DHC dựa trên các thuộc tính của khách hàng để tìm ra các cụm khách hàng có cùng đặc điểm với nhau. Từ kết quả thu được có thể giúp cho doanh nghiệp có các kế hoạch, chiến lược phù hợp cho từng phân khúc khách hàng.

Phần 2. PHƯƠNG PHÁP VÀ THUẬT TOÁN SỬ DỤNG

2.1. Tiền xử lý dữ liệu

Tiền xử lý dữ liệu là một phần của phân tích dữ liệu và khai thác dữ liệu. Nó liên quan đến việc chuyển đổi dữ liệu nguồn thô thành định dạng dễ hiểu cho máy móc, tránh các sai sót trong quá trình phân tích.

Dữ liệu thô có thể chứa các sự không nhất quán, lặp lại và các lỗi khác. Nó có thể không đầy đủ và mỗi tập dữ liệu có thể có định dạng riêng. Với tiền xử lý dữ liệu, bạn đang loại bỏ những sự không nhất quán này và chuẩn hóa định dạng, đảm bảo tất cả các tập dữ liệu có thiết kế đồng nhất, không có lỗi.

2.1.1. StringIndexer

- **StringIndexer** là một tính năng quan trọng của **PySpark** giúp chuyển đổi các cột chuỗi phân loại trong DataFrame thành các chỉ số số. Việc chuyển đổi này là cần thiết vì hầu hết các thuật toán học máy không thể làm việc trực tiếp với dữ liệu chuỗi.
- **StringIndexer** chỉ định một chỉ mục duy nhất cho mỗi giá trị chuỗi riêng biệt trong cột đầu vào và ánh xạ nó tới một cột đầu ra mới có chỉ mục số nguyên.
- **StringIndexer** xử lý các giá trị chuỗi trong cột đầu vào dựa trên tần suất xuất hiện của chúng trong tập dữ liệu. Theo mặc định, nhãn có tần suất cao nhất sẽ nhận chỉ số 0, nhãn có tần suất cao thứ hai nhận chỉ số 1, và cứ tiếp tục như vậy cho đến hết.
- Nếu hai nhãn có cùng tần suất, chỉ số sẽ được gán dựa trên thứ tự bảng chữ cái. Tuy nhiên, bạn cũng có thể thiết lập thứ tự tùy chỉnh cho các nhãn bằng cách sử dụng tham số '**stringOrderType**'.

Ví dụ: Giả sử ta có một biến với giá trị “Có” và “Không”. **StringIndexer** sẽ gán chỉ số 0 cho "Không" và chỉ số 1 cho "Có".

- **Lợi ích của StringIndexer:**

- **Nâng cao hiệu suất:** **StringIndexer** biến đổi dữ liệu danh mục thành dạng số, giúp các thuật toán học máy hoạt động hiệu quả hơn, từ đó cải thiện độ chính xác và tốc độ xử lý.

- **Đơn giản hóa quá trình xử lý:** Công cụ này giảm bớt các bước phức tạp trong tiền xử lý dữ liệu, giúp người dùng dễ dàng hơn trong việc chuẩn bị dữ liệu cho phân tích.
- **Khả năng linh hoạt:** **StringIndexer** có thể áp dụng cho nhiều loại biến danh mục khác nhau, bao gồm cả những biến có nhiều giá trị, làm cho nó trở thành công cụ hữu ích trong nhiều lĩnh vực phân tích dữ liệu.

2.1.2. Z-score

- **Z-score** là một phép đo thống kê định lượng khoảng cách giữa một điểm dữ liệu và giá trị trung bình của một tập dữ liệu. Nó được biểu thị bằng số độ lệch chuẩn. **Z-score** cho biết độ lệch chuẩn một điểm dữ liệu nằm cách xa giá trị trung bình của phân phối.

$$\text{Z-score} = \frac{x - \mu}{\sigma}$$

Trong đó:

x : giá trị đang được đánh giá

μ : giá trị trung bình

σ : độ lệch chuẩn

Ý nghĩa:

Giá trị Z-score > 0 cho biết giá trị lớn hơn giá trị trung bình.

Giá trị Z-score < 0 cho biết giá trị nhỏ hơn giá trị trung bình.

Giá trị Z-score ≈ 0 cho biết giá trị gần với giá trị trung bình.

- **Lợi ích của Z-Score:**

- Z-score dựa trên giá trị trung bình và độ lệch chuẩn, là những thống kê vững chắc giúp giảm thiểu ảnh hưởng của các giá trị ngoại lai và độ lệch trong dữ liệu của bạn.
- Thêm vào đó, Z-score có thể cải thiện hiệu suất và độ chính xác của một số thuật toán, như hồi quy tuyến tính, hồi quy logistic, phân cụm k-means và phân tích thành phần chính.

- Z-score làm cho dữ liệu của bạn dễ hiểu và so sánh hơn vì chúng có giá trị trung bình bằng không và độ lệch chuẩn bằng một. Điều này cho phép bạn dễ dàng thấy mối quan hệ giữa từng giá trị và phân phối của đặc tính đó.

- **Nhược điểm của Z-Score:**

- Việc sử dụng Z-score trong mô hình dự đoán có thể gặp một số nhược điểm, chẳng hạn như mất đi ý nghĩa và quy mô ban đầu của dữ liệu. Z-score biến đổi dữ liệu thành một đơn vị và khoảng giá trị khác, điều này có thể khiến việc diễn giải và giải thích kết quả trở nên khó khăn.
- Z-score giả định rằng các đặc tính phân phối theo quy luật chuẩn, điều này có thể không đúng với tất cả các tập dữ liệu, dẫn đến việc bóp méo các mối quan hệ và mâu thuẫn giữa các đặc tính.
- Z-score phụ thuộc vào giá trị trung bình và độ lệch chuẩn, có thể bị ảnh hưởng bởi các giá trị ngoại lai, giá trị thiểu hoặc đo lường sai, có khả năng gây ra lỗi và thiên lệch trong dữ liệu của bạn. Do đó, việc làm sạch và xác thực dữ liệu trước khi sử dụng Z-score là rất quan trọng.

2.1.3. PCA (Principal Component Analysis)

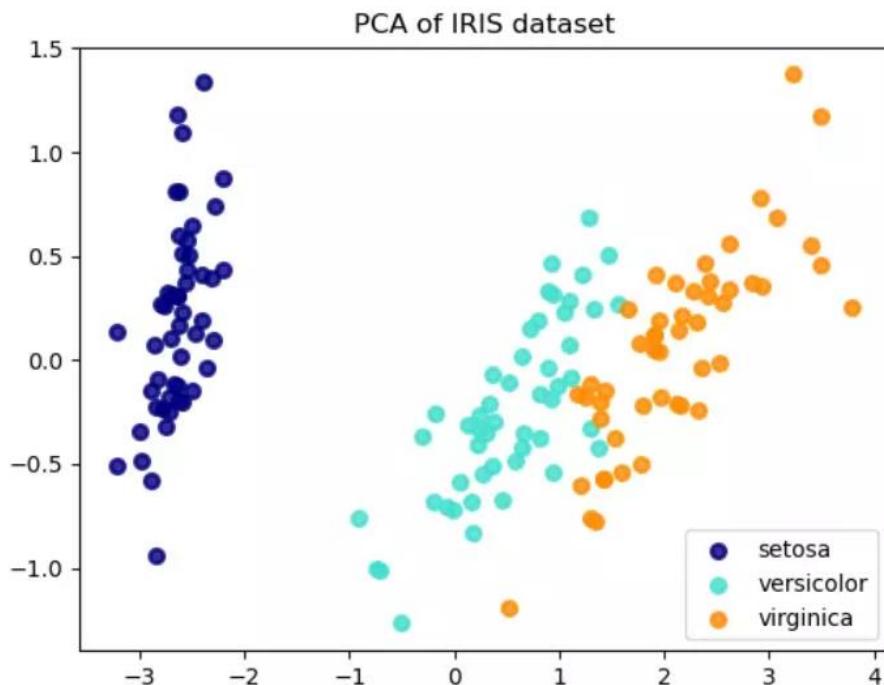
2.1.3.1. Vì sao cần giảm chiều dữ liệu?

- Trong các bài toán học máy thì dữ liệu có kích thước rất lớn. Máy tính có thể hiểu và thực thi các thuật toán trên dữ liệu này, tuy nhiên đối với con người để "nhìn" dữ liệu nhiều chiều thật sự là rất khó. Vì vậy bài toán giảm chiều dữ liệu ra đời giúp đưa ra cái nhìn mới cho con người về dữ liệu nhiều chiều.
- Ngoài để trực quan dữ liệu, các phương pháp giảm chiều dữ liệu còn giúp đưa dữ liệu về một không gian mới giúp khai phá các thuộc tính ẩn mà trong chiều dữ liệu ban đầu không thể hiện rõ, hoặc đơn giản là giảm kích thước dữ liệu để tăng tốc độ thực thi cho máy tính.
- **Ví dụ:** Tập dữ liệu huyền thoại Iris bao gồm 4 thuộc tính và 3 nhãn tương ứng với 3 loài hoa. Rất khó để có thể nhận biết rằng 4 thuộc tính này có phân tách với nhau theo mỗi loài hay không vì cần biểu diễn không gian này trên dữ liệu 4 chiều.

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa

Hình 2.1. Ví dụ về tập dữ liệu huyền thoại Iris

⇒ Vì vậy, thuật toán giảm chiều dữ liệu giúp đưa về không gian 2 chiều để dễ dàng trực quan hóa trên hệ toan độ Oxy, đổi lại là chúng ta phải chấp nhận mất mát đi một lượng thông tin. Và đây là kết quả:

*Hình 2.2. Kết quả bộ dữ liệu ví dụ sau khi thực hiện PCA*

Nhìn vào đây chúng ta có thể dễ dàng phân tích hơn, có thể thấy lớp nào dễ nhầm lẫn với nhau, mức độ tách biệt giữa các lớp, ...

2.1.3.2. Các bước thực hiện thuật toán PCA

- **Bước 1:** Chuẩn hóa dữ liệu
 - Trong quá trình này, phạm vi biến được tính toán và chuẩn hóa để phân tích sự đóng góp của từng biến một cách bình đẳng.
 - Việc tính toán các biến ban đầu sẽ giúp phân loại các biến đang chi phối các biến khác trong phạm vi nhỏ.
 - Điều này sẽ giúp có được kết quả thiên lệch vào cuối quá trình phân tích.
 - Để biến đổi các biến có cùng chuẩn, bạn có thể làm theo công thức sau:

$$Z = \frac{\text{Value} - \text{Mean}}{\text{Standard Deviation}}$$

Trong đó:

$$\text{Mean} = \frac{\text{Tổng của các giá trị}}{\text{Tổng số giá trị}}$$

$$\text{Standard Deviation} = \sqrt{\frac{\sum(x - \text{mean})^2}{n}}$$

x: Giá trị trong một tập dữ liệu

n: Số lượng giá trị trong tập dữ liệu

- **Bước 2:** Tính toán ma trận hiệp phương sai
 - Ở bước này, bạn sẽ biết được các biến của dữ liệu cho sẵn thay đổi như thế nào theo giá trị trung bình được tính toán.
 - Bất kỳ biến số nào liên quan cũng có thể được sắp xếp vào cuối bước này.
 - Để phân tách các biến có mối quan hệ chặt chẽ với nhau, bạn tính toán ma trận hiệp phương sai bằng cách sử dụng công thức cho sẵn.
 - Lưu ý: Ma trận hiệp phương sai là ma trận đối xứng N x N chứa các hiệp phương sai của tất cả các tập dữ liệu có thể có.
 - Ma trận hiệp phương sai của dữ liệu hai chiều được đưa ra như sau:

$$\begin{matrix} & x & y & z \\ x & var(x) & cov(x, y) & cov(x, z) \\ y & cov(x, y) & var(y) & cov(y, z) \\ z & cov(x, z) & cov(y, z) & var(z) \end{matrix}$$

$$S = \frac{1}{N-1} X^T X$$

Trong đó:

X : ma trận dữ liệu có kích thước $N \times N$

X^T : ma trận chuyển vị của X

N : số lượng mẫu của X

⇒ Ý nghĩa: Ma trận này phản ánh sự biến thiên giữa các thuộc tính trong dữ liệu.

- **Bước 3:** Tìm trị riêng và vecto riêng:

- Các vector riêng và trị riêng của ma trận hiệp phương sai sẽ xác định các trục chính của dữ liệu. Ma trận hiệp phương sai C có các trị riêng λ và vector riêng v thỏa mãn phương trình:

$$\lambda = C \cdot v$$

- Phương trình này cũng có thể được viết lại dưới dạng:

$$(C - \lambda I) \cdot v = 0$$

- Từ phương trình trên, ta có thể tìm thấy trị riêng λ , và tương tự vector riêng có thể được tìm được thông qua phương trình $Cv = \lambda v$

⇒ Sắp xếp các trị riêng theo thứ tự giảm dần và chọn ra k vector riêng tương ứng với k trị riêng lớn nhất. Các vector này chính là trục chính mới của dữ liệu.

- **Bước 4:** Chọn các trị riêng lớn nhất:

- Chọn k trị riêng lớn nhất: Sắp xếp các trị riêng theo thứ tự giảm dần và chọn k trị riêng lớn nhất.
- Tạo ma trận U : Ma trận U sẽ chứa các vector riêng tương ứng với k trị riêng đã chọn. U có kích thước $(N_{feature}, k)$

- **Bước 5:** Ánh xạ không gian ban đầu thành không gian k chiều

- Tính toán: Lấy ma trận dữ liệu đã chuẩn hóa X nhân với ma trận U

$$X_{new} = X \cdot U$$

- Kết quả: Dữ liệu đã được giảm chiều xuống k

2.2. Khai thác dữ liệu

2.2.1. Khoảng cách Euclidean

Khoảng cách Euclid là độ dài của đường ngắn nhất giữa hai điểm trong bất kỳ chiều nào. Nó cũng được gọi là khoảng cách trực giao hoặc khoảng cách Pythagore. Khoảng cách Euclid thường được sử dụng trong các thuật toán học máy, bao gồm: hồi quy tuyến tính, k-nearest neighbor và k-means clustering.

2.2.1.1. Công thức khoảng cách Euclidean một chiều

- Trong không gian một chiều, khoảng cách Euclid không chỉ trực quan mà còn dễ tính toán.

$$d(a, b) = |a - b|$$

- Nếu các đồ thị trên một đường thẳng là 3, 5 và 7, thì khoảng cách giữa 3 và 5 là 2, và giữa 3 và 7 là 4. Tuy nhiên, thay vì tuân theo công thức trên, chúng ta sẽ sử dụng công thức sau:

$$d(a, b) = \sqrt{(a - b)^2}$$

- Về mặt toán học, điều này tương đương với công thức trước đó. Tuy nhiên, chúng ta sẽ sử dụng công thức này để chuẩn hóa số liệu khi chúng ta đi vào các chiều cao hơn.

2.2.1.2. Công thức khoảng cách Euclidean hai chiều

- Lý do tại sao khoảng cách Euclid còn được gọi là khoảng cách Pythagore là khi chúng ta ở trong không gian 2D, chúng ta sử dụng định lý Pythagoras để tính cạnh huyền nhằm tìm khoảng cách giữa hai điểm.
- Lấy một tam giác trong đó a là một cạnh, b là cạnh còn lại và c là cạnh huyền. Tính khoảng cách c, trong hình bên cạnh, chúng ta tính khoảng cách 1D của a và b và sử dụng định lý Pythagore để tính giá trị c.
- Nếu ta nói điểm ở góc vuông là 0 thì các điểm ở hai đầu của đường thẳng có khoảng cách c trở thành $(a, 0)$ và $(0, b)$.
- Sau đó, khoảng cách 1D a và b sẽ là $\sqrt{a^2}$ và $\sqrt{b^2}$. Làm theo tương tự, ta sẽ tạo c thành

$$\sqrt{a^2 + b^2}$$

- Nói chung, nếu tọa độ của các điểm tương đương là (a_1, a_2) và (b_1, b_2) , thì khoảng cách giữa chúng sẽ như sau:

$$d((a_1, a_2), (b_1, b_2)) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2}$$

2.2.1.3. Công thức khoảng cách Euclidean cho bất kỳ kích thước

- Tương tự như trên, chúng ta có thể chứng minh bằng hình học (ít nhất là đối với 3D) rằng công thức tính Khoảng cách Euclid như sau:

$$d((a_1, a_2, a_3, \dots, a_n), (b_1, b_2, b_3, \dots, b_n)) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + (b_3 - a_3)^2 + \dots + (b_n - a_n)^2}$$

2.2.2. Phương sai

- Để tránh tổng các độ lệch bằng 0 và loại bỏ ảnh hưởng của kích thước mẫu người ta tính tổng bình phương các độ lệch và chia cho kích thước mẫu trừ 1 (hiệu chỉnh). Ta có kết quả là "trung bình bình phương các độ lệch" và gọi là phương sai mẫu (sample variance).

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

Trong đó:

s^2 : Phương sai mẫu, đo lường sự biến thiên của dữ liệu trong mẫu.

n : Kích thước mẫu, tức là số lượng quan sát trong tập dữ liệu.

x_i : Giá trị của quan sát thứ i trong mẫu

\bar{x} : Giá trị trung bình của mẫu

$(x_i - \bar{x})^2$: Độ lệch bình phương của mỗi giá trị so với giá trị trung bình, thể hiện sự biến thiên của từng giá trị

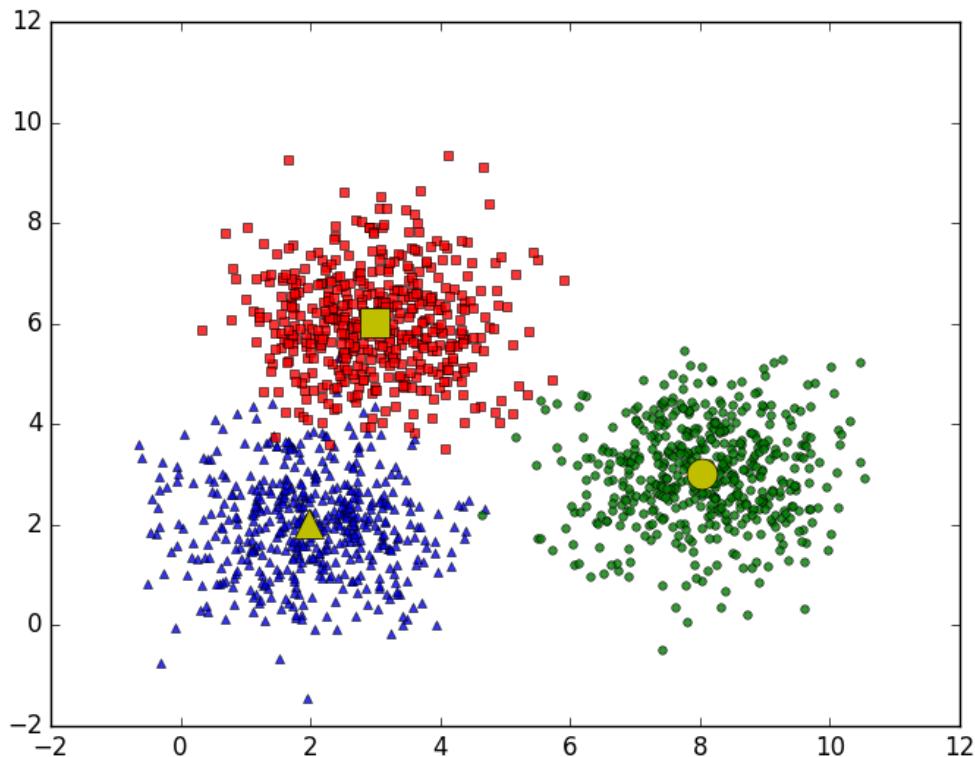
- Phương sai là tham số rất tốt để đo lường sự biến thiên (hay phân tán) của dữ liệu trong mẫu vì nó đo quan sát độ lệch của các giá trị từ giá trị trung bình, loại bỏ ảnh hưởng của kích thước mẫu và là một hàm trơn (smooth function). Tuy nhiên, điều này yêu cầu phương sai là không cố định đơn vị với giá trị trung bình. Đơn vị tính của phương sai là bình phương đơn vị giá trị trung

bình. Chẳng hạn, nếu đơn vị là thời gian chạy thì phương sai là giây bình phương, điều này khiến cho việc so sánh trở nên khó khăn hơn.

2.2.3. Thuật toán K-Means

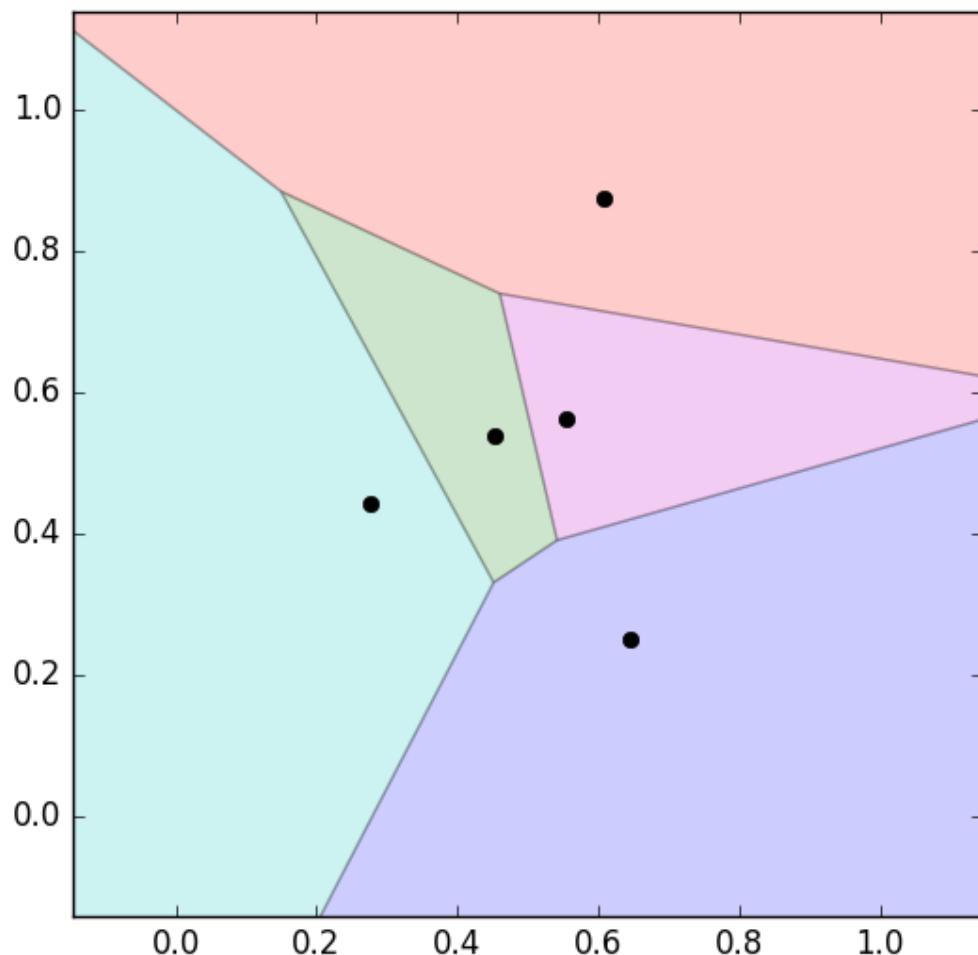
2.2.3.1. Giới thiệu

- Trong thuật toán K-Means Clustering, chúng ta không biết nhãn (label) của từng điểm dữ liệu. Mục đích là làm thế nào để phân dữ liệu thành các cụm (cluster) khác nhau sao cho dữ liệu trong cùng cụm có tính chất giống nhau.
- **Ví dụ:** Một công ty muốn tạo ra những chính sách ưu đãi cho những nhóm khách hàng khác nhau dựa trên sự tương tác giữa mỗi khách hàng với công ty đó (số năm là khách hàng; số tiền khách hàng đã chi trả cho công ty; độ tuổi; giới tính; thành phố; nghề nghiệp; ...). Giả sử công ty đó có rất nhiều dữ liệu của rất nhiều khách hàng nhưng chưa có cách nào chia toàn bộ khách hàng đó thành một số nhóm/cụm khác nhau. Nếu một người biết Machine Learning được đặt câu hỏi này, phương pháp đầu tiên anh (chị) ta nghĩ đến sẽ là K-Means Clustering. Vì nó là một trong những thuật toán đầu tiên mà anh ấy tìm được trong các cuốn sách, khóa học về Machine Learning. Sau khi đã phân ra được từng nhóm, nhân viên công ty đó có thể lựa chọn ra một vài khách hàng trong mỗi nhóm để quyết định xem mỗi nhóm tương ứng với nhóm khách hàng nào. Phần việc cuối cùng này cần sự can thiệp của con người, nhưng lượng công việc đã được rút gọn đi rất nhiều.
- Ý tưởng đơn giản nhất về cluster (cụm) là tập hợp các điểm ở gần nhau trong một không gian nào đó (không gian này có thể có rất nhiều chiều trong trường hợp thông tin về một điểm dữ liệu là rất lớn). Hình bên dưới là một ví dụ về 3 cụm dữ liệu (tù giờ sẽ viết gọn là cluster).



Hình 2.3. Ví dụ về phân cụm dữ liệu

- Giả sử mỗi cluster có một điểm đại diện (center) màu vàng. Và những điểm xung quanh mỗi center thuộc vào cùng nhóm với center đó. Một cách đơn giản nhất, xét một điểm bất kỳ, ta xét xem điểm đó gần với center nào nhất thì nó thuộc về cùng nhóm với center đó. Tới đây, chúng ta có một bài toán thú vị: Trên một vùng biển hình vuông lớn có ba đảo hình vuông, tam giác, và tròn màu vàng như hình trên. Một điểm trên biển được gọi là thuộc lãnh hải của một đảo nếu nó nằm gần đảo này hơn so với hai đảo kia. Hãy xác định ranh giới lãnh hải của các đảo.
- Hình dưới đây là một hình minh họa cho việc phân chia lãnh hải nếu có 5 đảo khác nhau được biểu diễn bằng các hình tròn màu đen:



Hình 2.4. Minh họa cho việc phân chia lánh hải nếu có 5 đảo (chấm đen)

2.2.3.2. K-means hoạt động như thế nào?

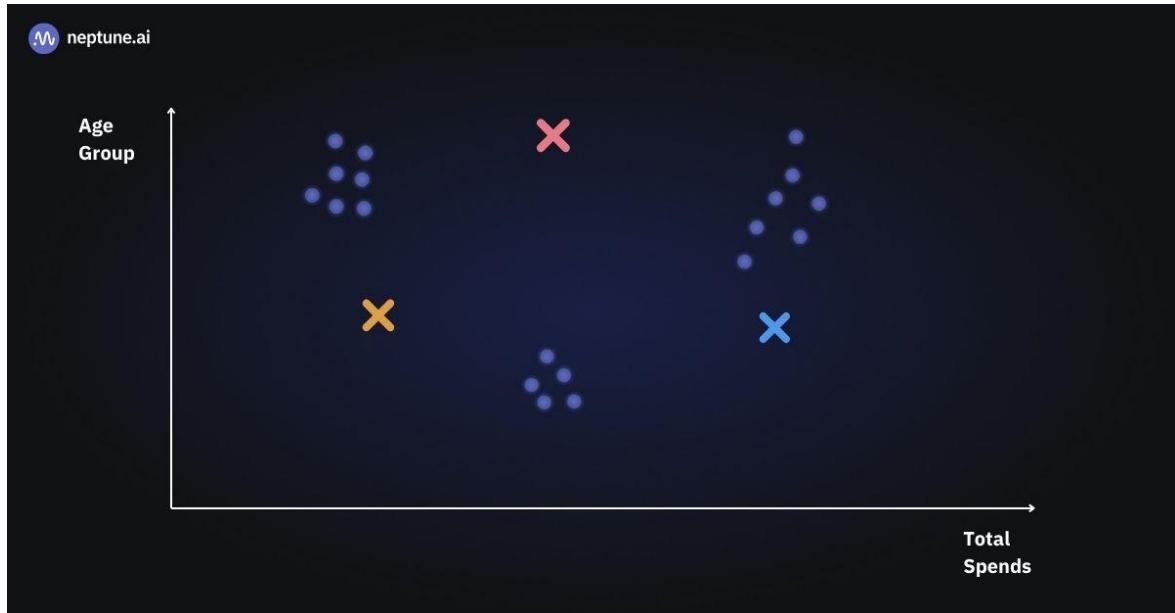
Hãy lấy một ví dụ để hiểu cách K-means hoạt động từng bước. Thuật toán có thể được chia thành 4-5 bước.

- **Bước 1:** Chọn số lượng cụm

Bước đầu tiên là xác định số lượng cụm K mà chúng ta sẽ nhóm dữ liệu. Hãy chọn K=3.

- **Bước 2:** Khởi tạo tâm

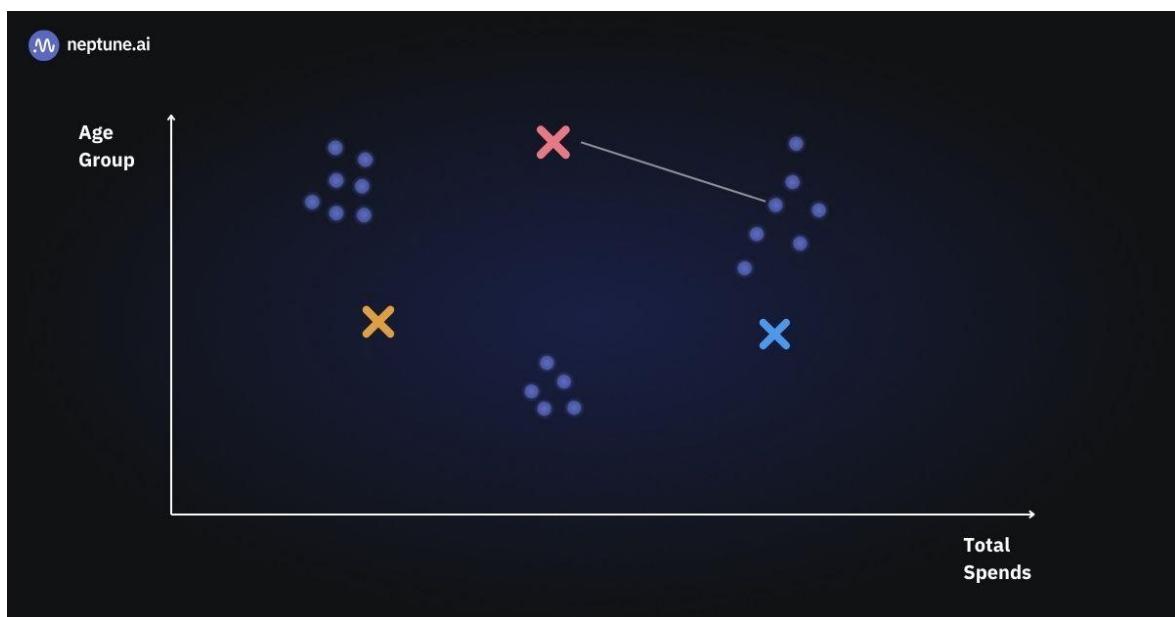
Tâm là tâm của một cụm nhưng ban đầu, tâm chính xác của các điểm dữ liệu sẽ không được biết, do đó, chúng ta chọn các điểm dữ liệu ngẫu nhiên và xác định chúng là tâm cho mỗi cụm. Chúng ta sẽ khởi tạo 3 tâm trong tập dữ liệu.



Hình 2.5. Minh họa khởi tạo tâm cho cụm

- **Bước 3:** Gán điểm dữ liệu cho cụm gần nhất

Bây giờ các tâm đã được khởi tạo, bước tiếp theo là gán các điểm dữ liệu X_n cho tâm cụm gần nhất của chúng là C_k



Hình 2.6. Minh họa gán điểm dữ liệu cho cụm gần nhất

Ở bước này, trước tiên chúng ta sẽ tính khoảng cách giữa điểm dữ liệu X và tâm C bằng phép đo khoảng cách Euclid.

Euclidean

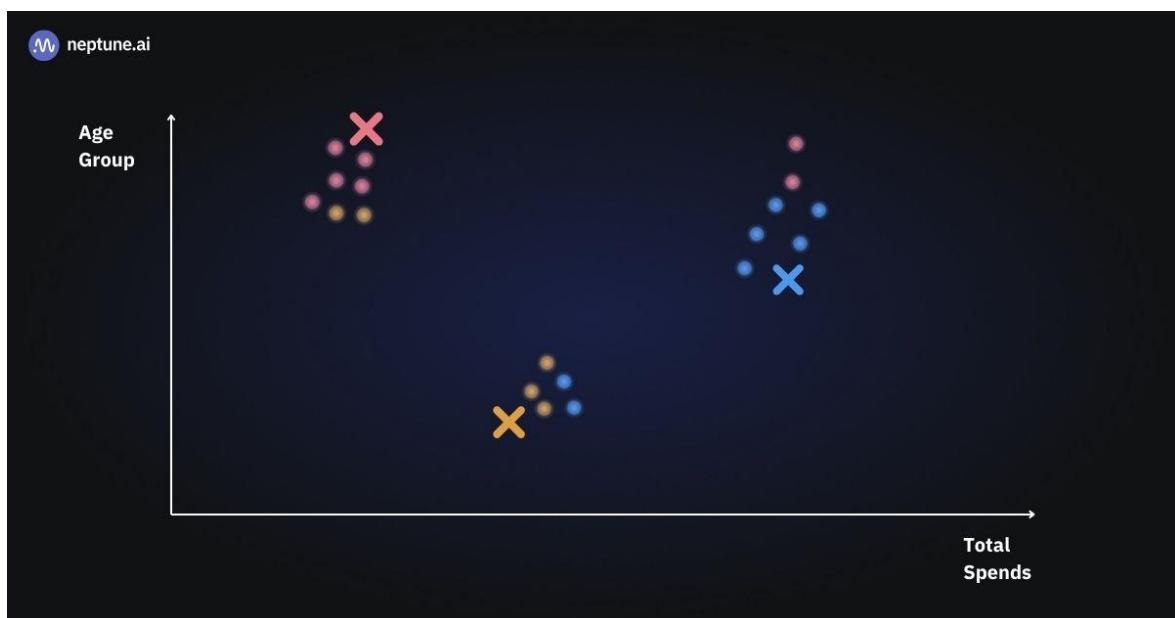
$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Sau đó chọn cụm cho các điểm dữ liệu có khoảng cách giữa điểm dữ liệu và tâm là nhỏ nhất.

- **Bước 4:** Khởi tạo lại tâm

Tiếp theo, chúng ta sẽ khởi tạo lại trọng tâm bằng cách tính giá trị trung bình của tất cả các điểm dữ liệu của cụm đó.

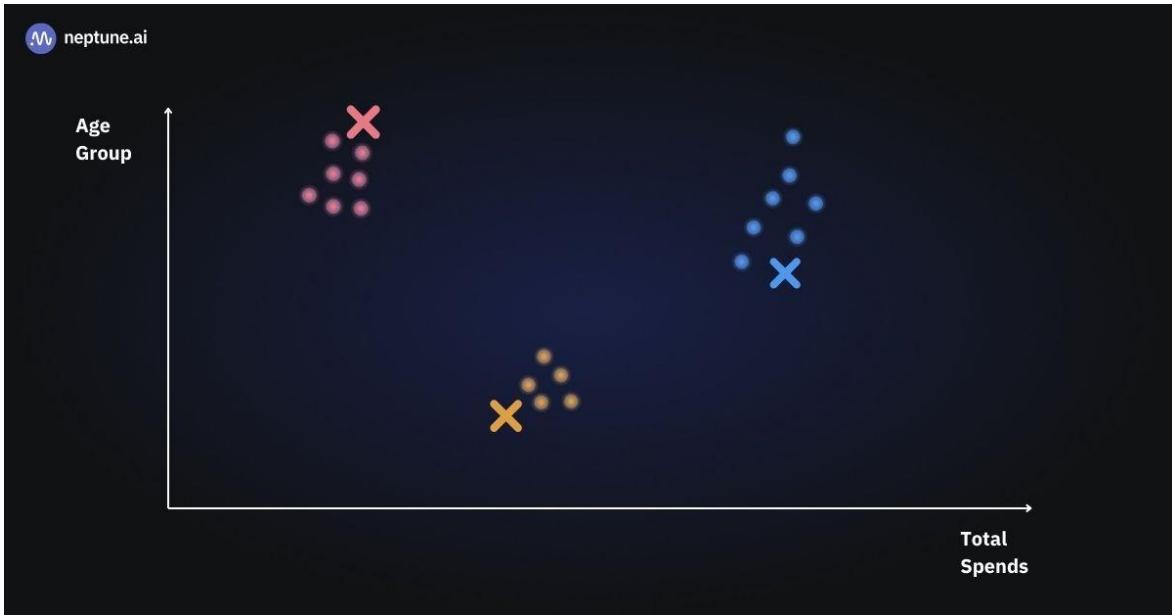
$$C_i = \frac{1}{|N_i|} \sum x_i$$



Hình 2.7. Minh họat khởi tạo lại tâm từ các cụm đã chia

- **Bước 5:** Lặp lại bước 3 và bước 4

Chúng ta sẽ tiếp tục lặp lại bước 3 và 4 cho đến khi có được trọng tâm tối ưu và việc chỉ định điểm dữ liệu cho các cụm chính xác không còn thay đổi nữa.

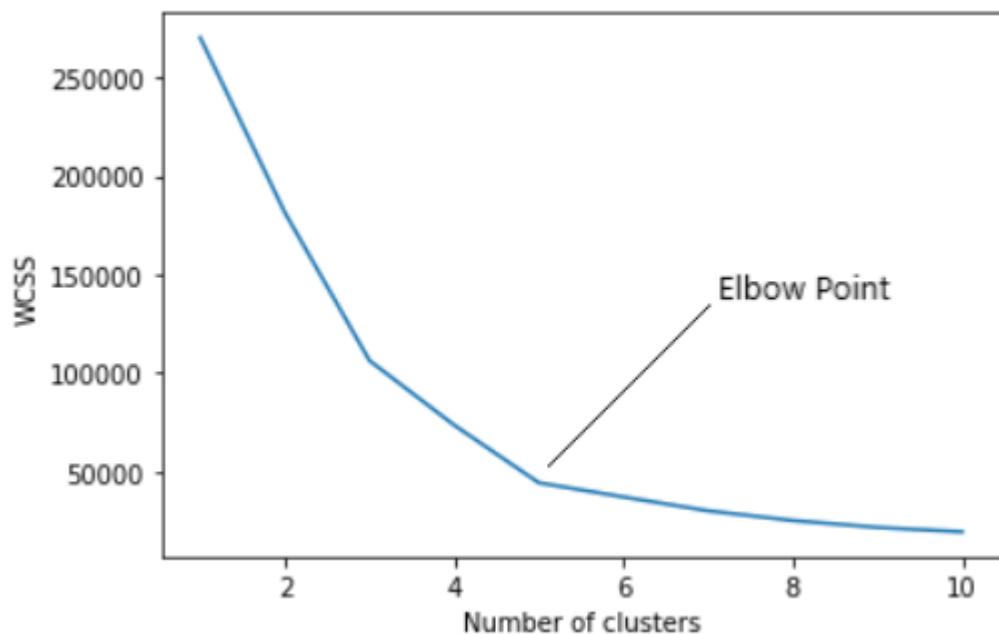


Hình 2.8. Minh họat kết quả sau khi tìm được tâm tối ưu – Phân cụm thành công

2.2.3.3. Cách chọn k tối ưu - Elbow

- Phương pháp **Elbow** là một kỹ thuật được sử dụng trong phân tích dữ liệu và học máy để xác định số lượng cụm tối ưu trong một tập dữ liệu. Phương pháp này bao gồm việc vẽ biểu đồ phương sai được giải thích bởi các số lượng cụm khác nhau và xác định điểm "elbow", tại đó tỷ lệ phương sai giảm mạnh và ổn định, gợi ý số lượng cụm phù hợp để phân tích hoặc đào tạo mô hình.
- Phương pháp này là một kỹ thuật trực quan được sử dụng để xác định giá trị K tốt nhất cho thuật toán phân cụm K-means. Trong phương pháp này, một đồ thị được gọi là đồ thị khuỷu tay sẽ vẽ các giá trị tổng bình phương trong cụm (**WCSS - Within-Cluster Sum of Squares**) so với các giá trị K khác nhau. Giá trị K tối ưu được xác định tại điểm mà đồ thị uốn cong như một khuỷu tay.
- Trong phương pháp **Elbow**, chúng ta thực sự thay đổi số lượng cụm (K) từ 1 đến 10. Đối với mỗi giá trị K, chúng ta tính toán **WCSS** (Within-Cluster Sum of Squares). WCSS là tổng khoảng cách bình phương giữa mỗi điểm và tâm cụm trong một cụm. Khi chúng ta vẽ biểu đồ WCSS với giá trị K, biểu đồ sẽ có hình dạng giống như một cái khuỷu tay (Elbow). Khi số lượng cụm tăng lên, giá trị WCSS sẽ bắt đầu giảm. Giá trị WCSS lớn nhất khi K = 1.

- Khi phân tích đồ thị, chúng ta có thể thấy rằng đồ thị sẽ thay đổi nhanh chóng tại một điểm nhất định và tạo ra hình dạng khuỷu tay. Từ điểm này, đồ thị di chuyển gần như song song với trục hoành (trục X). Giá trị K tương ứng với điểm này là giá trị tối ưu của K hoặc số lượng cụm tối ưu.



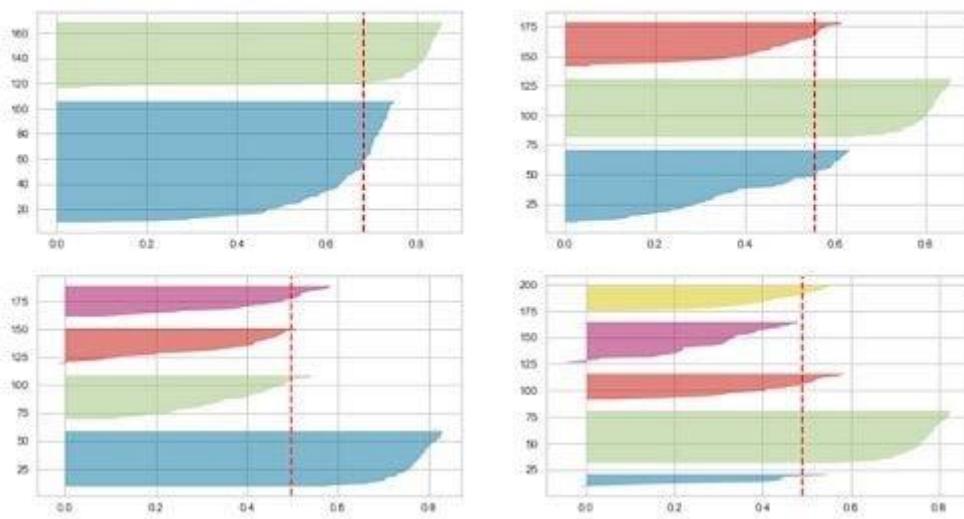
Hình 2.9. Biểu đồ kết quả của phương pháp Elbow

- Một số lưu ý về phương pháp Elbow:
 - Có thể mang tính chủ quan: “Khuỷu tay” không phải lúc nào cũng là một khúc gãy rõ ràng, có thể khó xác định điểm chính xác một cách khách quan.
 - Không phải lúc nào cũng là phương pháp tốt nhất: Phương pháp Elbow có thể không phù hợp cho tất cả các tập dữ liệu, đặc biệt là những tập có độ chiều cao hoặc các cụm có hình dạng không đều.
 - Cân nhắc các yếu tố khác: Mặc dù phương pháp Elbow là một điểm khởi đầu hữu ích, nhưng cũng quan trọng để xem xét các yếu tố khác như khả năng giải thích của các cụm và kiến thức miền của dữ liệu khi chọn giá trị k cuối cùng.
- Ngoài Elbow, còn có thuật toán Silhouette - kỹ thuật rất hữu ích để tìm K tối ưu cho phân cụm K-means

```

For n_clusters = 2 The average silhouette_score is : 0.6810461692117462
For n_clusters = 3 The average silhouette_score is : 0.5528190123564895
For n_clusters = 4 The average silhouette_score is : 0.49805050499728737
For n_clusters = 5 The average silhouette_score is : 0.4887488870931056

```



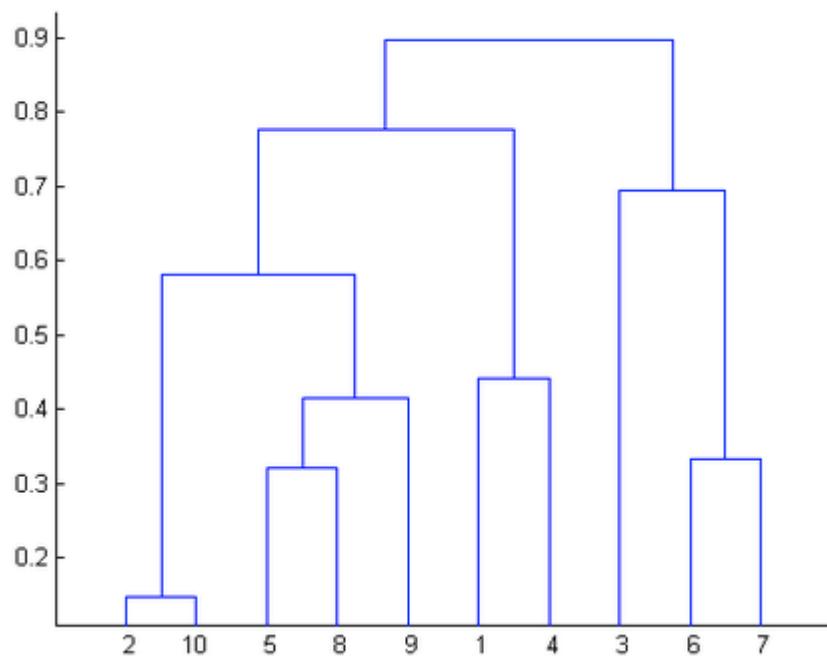
Hình 2.10. Minh họat kết quả thuật toán Silhouette

⇒ Tùy theo dataset, chúng ta sẽ chọn phương pháp phù hợp.

2.2.4. Thuật toán DHC (Divisive Hierarchical Clustering)

2.2.4.1. Hierarchical Clustering (phân cụm phân cấp)

- Thuật toán phân cụm K-means cho thấy cần phải cấu hình trước số lượng cụm cần phân chia. Ngược lại, phương pháp phân cụm phân cấp (Hierarchical Clustering) không yêu cầu khai báo trước số lượng cụm. Thay vào đó, thuật toán chỉ yêu cầu xác định trước thước đo về sự khác biệt giữa các cụm (không giao nhau), dựa trên sự khác biệt từng cặp giữa các quan sát trong hai cụm. Theo phương pháp này, chúng tạo ra những biểu diễn phân cấp trong đó các cụm ở mỗi cấp của hệ thống phân cấp được tạo bằng cách hợp nhất các cụm ở cấp độ thấp hơn bên dưới. Ở cấp thấp nhất, mỗi cụm chứa một quan sát. Ở cấp cao nhất, chỉ có một cụm chứa tất cả dữ liệu. Các cấp của biểu diễn phân cụm được thể hiện trong đồ thị dendrogram bên dưới.



Hình 2.11. Đồ thị minh họa biểu diễn phân cụm phân cấp (Dendrogram)

- Thuật toán phân cụm phân cấp được xây dựng trên bộ dữ liệu có kích thước N thì sẽ trải qua tổng cộng N bước phân chia. Có hai chiến lược phân chia chính phụ thuộc vào chiều di chuyển trên biểu đồ dendrogram mà chúng ta sẽ tìm hiểu bên dưới:
 - **Chiến lược hợp nhất:** Chiến lược này sẽ đi theo chiều bottom-up (từ dưới lên trên). Quá trình phân cụm bắt đầu ở dưới cùng tại các node lá (còn gọi là leaf node hoặc terminal node). Ban đầu mỗi quan sát sẽ được xem là một cụm tách biệt được thể hiện bởi một node lá. Ở mỗi level chúng ta sẽ tìm cách hợp một cặp cụm thành một cụm duy nhất nhằm tạo ra một cụm mới ở level cao hơn tiếp theo. Cụm mới này tương ứng với các node quyết định (non-leaf node). Như vậy sau khi hợp cụm thì số lượng cụm ít hơn. Một cặp được chọn để hợp nhất sẽ là những cụm trung gian không giao nhau.
 - **Chiến lược phân chia:** Chiến lược này sẽ thực hiện theo chiều top-down. Tức là phân chia bắt đầu từ node gốc của đồ thị. Node gốc bao gồm toàn bộ các quan sát, tại mỗi level chúng ta phân chia một cách đệ qui các cụm đang tồn tại tại level đó thành hai cụm mới. Phép phân chia được tiến hành

sao cho tạo thành hai cụm mới mà sự tách biệt giữa chúng là lớn nhất. Sự tách biệt này sẽ được đo lường thông qua một thước đo khoảng cách mà ta sẽ tìm hiểu kĩ hơn bên dưới.

⇒ Như vậy đồ thị của chiến lược phân chia và chiến lược hợp nhất đều là cây nhị phân, chúng chỉ khác biệt về chiều thực hiện thuật toán. Node gốc của cây nhị phân sẽ bao gồm toàn bộ các quan sát và cây nhị phân bao gồm N node là đại diện cho N quan sát từ bộ dữ liệu. Mỗi một node quyết định bao gồm hai node con. Quá trình phân chia thì hai node con thể hiện kết quả được phân chia từ node cha và quá trình hợp nhất thì node cha là thể hiện kết quả sau khi gộp hai node con.

2.2.4.2. Divisive Hierarchical Clustering

- Chiến lược phân chia chưa được nghiên cứu và phát triển rộng rãi trong các bài toán phân cụm như hợp nhất. Trong sklearn cũng chưa có module phát triển cho phương pháp này. Nó được giới thiệu lần đầu trong một tài liệu của Gersho và Grey, 1992 về kỹ thuật nén. Chiến lược phân chia sẽ bắt đầu từ một cụm gồm toàn bộ các quan sát bên trong cụm và sau đó phân chia đệ qui những cụm đang tồn tại thành hai cụm con tại mỗi bước theo hướng top-down.
- Đầu tiên thuật toán sẽ chọn ra một điểm từ toàn bộ tập dữ liệu S sao cho điểm này thoả mãn điều kiện trung bình khoảng cách từ điểm đó tới toàn bộ những điểm còn lại là nhỏ nhất. Chúng ta đưa điểm này vào tập S1, tập còn lại gồm N-1 điểm là tập S2. Tiếp theo ta sẽ thực hiện các lượt phân chia sao cho mỗi một lượt lựa chọn ra một điểm xi từ tập S2 đưa sang S1. Điểm này cần thoả mãn hai điều kiện:
 - Trung bình khoảng cách từ điểm đó tới toàn bộ các điểm còn lại trong S1 phải là nhỏ nhất. Điều đó có nghĩa là xi là điểm tách biệt nhất so với phần còn lại của S1.

$$\mathbf{x}_i = \arg \max_{\mathbf{x}_i} \frac{1}{|\mathcal{S}_1| - 1} \sum_{j=1, j \neq i}^{|\mathcal{S}_1|} d(\mathbf{x}_i, \mathbf{x}_j)$$

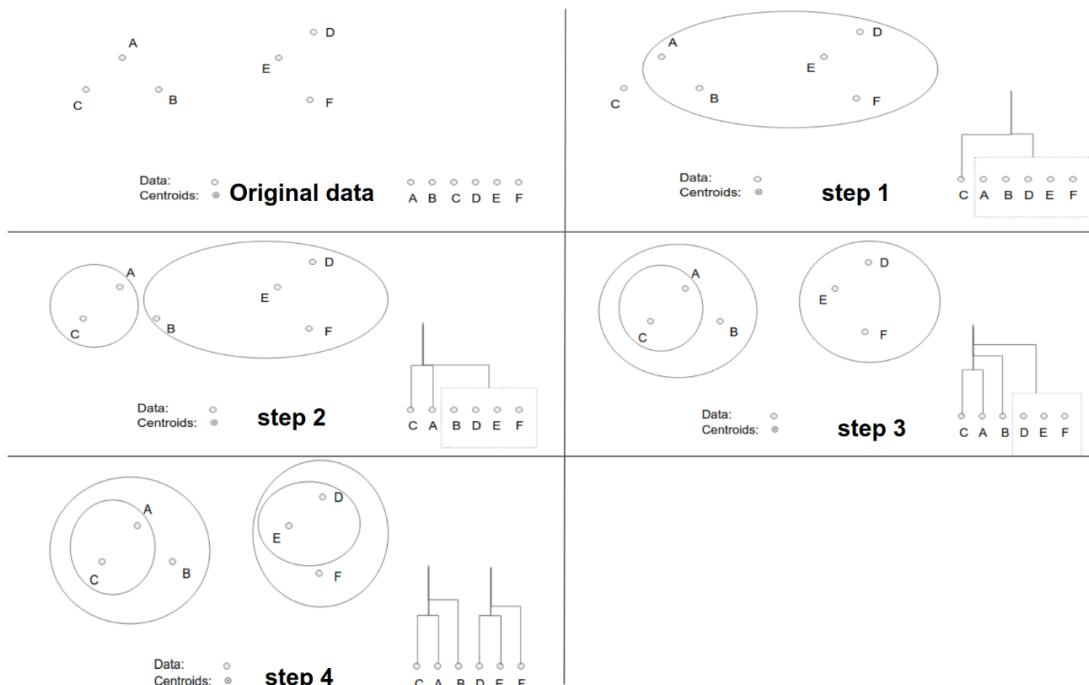
- Khoảng cách tối thiểu từ xi tới các điểm trong S2 phải lớn hơn khoảng cách tối thiểu tới các điểm trong S1. Điều này nhằm mục đích khiến cho điểm xi phải gần với cụm S2 hơn cụm S1.

$$d(\mathbf{x}_i, \mathcal{S}_1) \geq d(\mathbf{x}_i, \mathcal{S}_2)$$

Trong đó:

$$d(\mathbf{x}_i, \mathcal{S}_k) = \min_{\mathbf{x}_j, \mathbf{x}_j \in \mathcal{S}_k} d(\mathbf{x}_i, \mathbf{x}_j)$$

- Quá trình chuyển cụm sẽ kết thúc khi không còn điểm nào thoả mãn hai điều kiện trên. Khi đó chúng ta lại thực hiện đệ qui lại quá trình trên trên từng tập S1 và S2
- Chúng ta cùng diễn giải lại quá trình này thông qua hình minh họa bên dưới:



Hình 2.12. Minh họa các bước xử lý của thuật toán DHC

- Giải thích:
 - Ở bước 1 chúng ta sẽ lựa chọn ra điểm C là điểm đầu tiên thuộc cụm mới dựa trên khoảng cách so với các điểm còn lại là xa nhất. Sau bước 1 ta thu được tập $S1=\{C\}$ và $S2=\{A,B,D,E,F\}$.
 - Tại bước 2 lựa chọn trong số các điểm thuộc $S2$ ra điểm mà có khoảng cách xa nhất so với những điểm còn lại sao cho điểm này gần với C hơn so với các điểm thuộc tập $S2$, đó chính là điểm A. Di chuyển điểm này sang $S1$.
 - Bước 3 chúng ta lại tiếp tục thực hiện như vậy và lựa chọn được điểm B để đưa sang $S1$.
 - Ở bước thứ 4 ta sẽ dừng quá trình chuyển cụm cho các điểm thuộc $S2$ vì thuật toán đã đạt sự hội tụ về hai cụm. Khi đó ta lại tiếp tục tiến hành đệ qui thuật toán trên từng cụm con.

2.3. Các phương pháp đánh giá

2.3.1. Silhouette Coefficient

- Một chỉ số để đánh giá chất lượng phân cụm được gọi là Silhouette Coefficient. Có thể được áp dụng cho các thuật toán phân cụm. Silhouette Coefficient dao động từ -1 đến 1, trong đó Silhouette Coefficient cao hơn có nghĩa là các cụm rõ ràng hơn. Silhouette Coefficient gần 1 có nghĩa là một điểm nằm gần hơn với các điểm trong cùng một cụm so với các cụm lân cận. Giá trị 0 có nghĩa là điểm đó nằm ở ranh giới giữa hai cụm lân cận. Thực tế, các giá trị âm cho thấy rằng các điểm có thể đã bị gán sai vào cụm khác.
- Để tính toán Silhouette Coefficient, độ gắn kết của cụm (cluster cohesion) và độ tách biệt của cụm (cluster separation) phải được tính. Độ gắn kết là khoảng cách trung bình giữa một điểm và các điểm khác trong cùng cụm, trong khi độ tách biệt là khoảng cách trung bình giữa một điểm và tất cả các điểm trong cụm gần nhất khác. Silhouette Coefficient có thể được tính toán như sau.

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

Trong đó:

a(i) là khoảng cách trung bình từ i tới tất cả các node trong cùng cụm với i.

b(i) là khoảng cách trung bình ngắn nhất từ i tới bất kỳ cụm nào không có i.

2.3.2. Calinski-Harabasz Index

- Calinski-Harabasz Index (CH) (được giới thiệu bởi Calinski và Harabasz vào năm 1974) có thể được sử dụng để đánh giá mô hình khi không có nhãn thực tế, nơi việc xác thực mức độ tốt của phân cụm được thực hiện dựa trên các đại lượng và đặc trưng vốn có của tập dữ liệu. Calinski-Harabasz Index (còn được gọi là tiêu chí tỷ lệ phương sai) là một thước đo mức độ tương đồng của một đối tượng với cụm của nó (độ gắn kết) so với các cụm khác (độ tách biệt). Ở đây, độ gắn kết được ước lượng dựa trên khoảng cách từ các điểm dữ liệu trong một cụm đến tâm cụm của nó, trong khi độ tách biệt được dựa trên khoảng cách của các tâm cụm từ tâm toàn cầu.
- Calinski-Harabasz Index cho K cụm trên một tập dữ liệu $D = [d_1, d_2, d_3, \dots, d_N]$ được tính như sau:

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

Trong đó:

n_k : là số điểm của cụm thứ k,

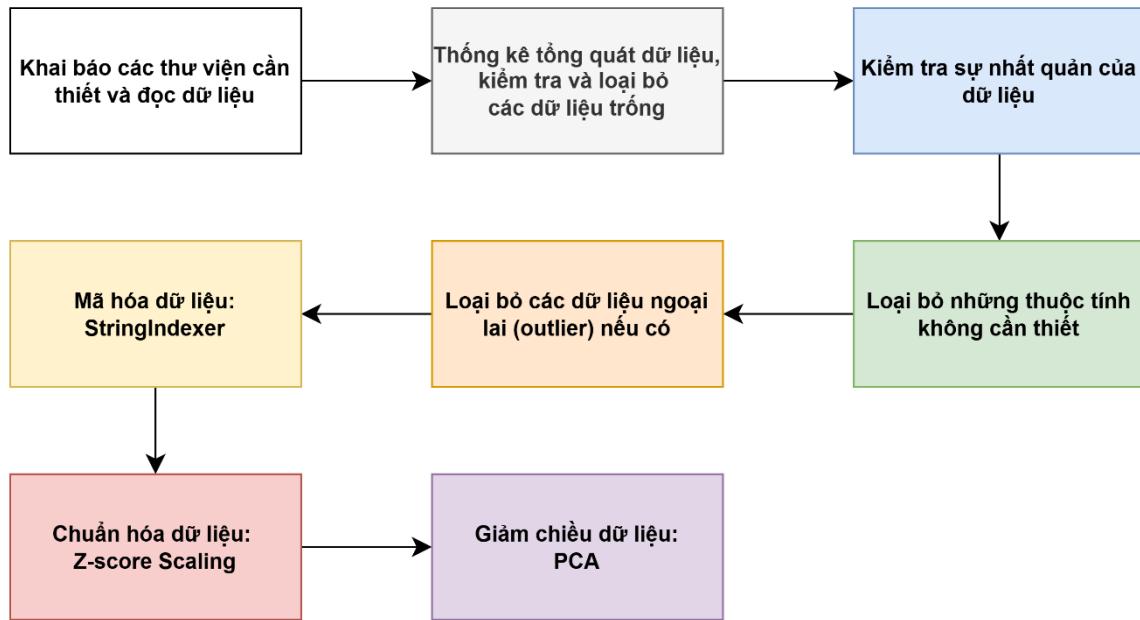
c_k : là số tâm cụm của cụm thứ k

c: là tâm toàn cầu

N: là tổng số điểm dữ liệu.

- ⇒ Giá trị Calinski-Harabasz Index cao hơn có nghĩa là các cụm dày đặc và tách biệt tốt, mặc dù không có giá trị "chấp nhận được" nào. Chúng ta cần chọn giải pháp mang lại đỉnh hoặc ít nhất là một khuỷu tay rõ ràng trên biểu đồ đường của các Calinski-Harabasz Index. Mặt khác, nếu đường biểu diễn mượt mà (ngang hoặc tăng/giảm), thì không có lý do gì để ưu tiên một giải pháp hơn các giải pháp khác.

Phần 3. TIỀN XỬ LÝ DỮ LIỆU



Hình 3.1. Sơ đồ các bước tiền xử lý dữ liệu

3.1. Các thư viện được sử dụng

Khai báo thư viện đồng thời bắt đầu một Spark Session



```
1  from pyspark.sql import SparkSession
2  from pyspark.sql.functions import *
3  from pyspark.sql.window import Window
4  from pyspark.sql.types import *
5
6  #Tạo kết nối Spark session
7  spark = SparkSession.builder \
8      .appName("DataProcessing") \
9      .getOrCreate()
10
11 #Import thư viện cần thiết
12 import numpy as np
13 import datetime
14 import matplotlib
15 import matplotlib.pyplot as plt
16 from matplotlib import colors
17 import seaborn as sns
18 from math import comb
19 from collections import defaultdict
20 from sklearn.preprocessing import LabelEncoder, StandardScaler
21 from yellowbrick.cluster import KElbowVisualizer
22 from mpl_toolkits.mplot3d import Axes3D
23 import plotly.express as px
24 from matplotlib.colors import ListedColormap
25 import warnings
26 import sys
27 if not sys.warnoptions:
28     warnings.simplefilter("ignore")
29 np.random.seed(42)
30
31 # PySpark MLlib cho machine learning
32 from pyspark.ml.feature import StringIndexer, PCA
33 from pyspark.ml.clustering import KMeans as SparkKMeans
34 from pyspark.ml.evaluation import ClusteringEvaluator
35 from pyspark.ml.feature import VectorAssembler
36 from pyspark.ml.linalg import Vectors
```

3.2. Đọc dữ liệu đầu vào

- Import dữ liệu và đọc dữ liệu BankChurners.csv bằng **spark.read.csv**



```
1 # Đọc dữ liệu từ thư mục CSV
2 # data = spark.read.format("csv") \
3 #     .option("header", "true") \
4 #     .option("inferSchema", "true") \
5 #     .load("/home/kanh/Desktop/Project/dataset/")
6
7 # Xác định đường dẫn file
8 file_path = "/content/BankChurners.csv"
9
10 # Đọc dữ liệu từ file CSV
11 data = spark.read.csv(file_path, header=True, inferSchema=True)
```

- Xem dữ liệu



```
1 #Hiển thị số lượng datapoint
2 num_data_points = data.count()
3 print(f"Number of datapoints: {num_data_points}")
4
5 #Hiển thị 5 dòng đầu tiên
6 data.show(5)
```

- Khai báo các hàm **col**, **count**, **isnan**, và **when** từ thư viện **pyspark.sql.functions** để đếm số lượng dữ liệu không Null ở từng cột. Đồng thời, sử dụng **printSchema()** để kiểm tra tính hợp lý của các kiểu dữ liệu trong từng cột.



```

1 #In so do cua DataFrame
2 data.printSchema()
3
4 # Mo ta dataframe de tinh xac suat
5 data.select(*[f.name for f in data.schema.fields if f.dataType.typeName() in ['integer', 'double']]).describe().show()
6
7 # Dem cac gia tri khac NULL
8 from pyspark.sql.functions import col, count, isnan, when
9
10 def count_not_null(c):
11     return count(when(col(c).isNotNull() & ~isnan(col(c)), c))
12
13 non_null_counts = data.agg(*(count_not_null(c).alias(c) for c in data.columns))
14 non_null_counts.show()

```

```

root
|-- CLIENTNUM: integer (nullable = true)
|-- Attrition_Flag: string (nullable = true)
|-- Customer_Age: integer (nullable = true)
|-- Gender: string (nullable = true)
|-- Dependent_count: integer (nullable = true)
|-- Education_Level: string (nullable = true)
|-- Marital_Status: string (nullable = true)
|-- Income_Category: string (nullable = true)
|-- Card_Category: string (nullable = true)
|-- Months_on_book: integer (nullable = true)
|-- Total_Relationship_Count: integer (nullable = true)
|-- Months_Inactive_12_mon: integer (nullable = true)
|-- Contacts_Count_12_mon: integer (nullable = true)
|-- Credit_Limit: double (nullable = true)
|-- Total_Revolving_Bal: integer (nullable = true)
|-- Avg_Open_To_Buy: double (nullable = true)
|-- Total_Amt_Chng_Q4_Q1: double (nullable = true)
|-- Total_Trans_Amt: integer (nullable = true)
|-- Total_Trans_Ct: integer (nullable = true)
|-- Total_Ct_Chng_Q4_Q1: double (nullable = true)
|-- Avg_Utilization_Ratio: double (nullable = true)
|-- Naive_Bayes_classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1: double (nullable = true)
|-- Naive_Bayes_classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2: double (nullable = true)

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|summary|CLIENTNUM|Customer_Age|Dependent_count|Months_on_book|Total_Relationship_Count|Months_Inactive_12_mon|Contacts_Count_12_mon|Credit_Limit|Total_Revolv
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| count | 10127 | 10127 | 10127 | 10127 | 10127 | 10127 | 10127 | 10127 | 10127 |
| mean | 7.391776063336625E8 | 46.32596030413745 | 2.3462032191172115 | 35.928409203120874 | 3.8125802310654686 | 2.3411671768539546 | 2.4553174681544387 | 8631.9536980834848 | 1162.8140
| stddev | 3.690378345023116E7 | 8.016814032549046 | 1.29890834890379 | 7.98641633087208 | 1.55440786533883 | 1.0106223994182844 | 1.1062251426359249 | 9088.776650223148 | 814.9873
| min | 708082083 | 26 | 0 | 13 | 1 | 0 | 0 | 1438.3 |
| max | 828343083 | 73 | 5 | 56 | 6 | 6 | 6 | 34516.0 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

⇒ Nhận xét:

- Bộ dữ liệu không chứa giá trị NULL hoặc trống, đảm bảo tính đầy đủ và độ tin cậy của thông tin.
- Độ tuổi trung bình của khách hàng được ước tính là 46 tuổi, với độ tuổi dao động từ 26 đến 73, cho thấy sự đa dạng trong nhóm khách hàng.
- Trung bình, mỗi khách hàng có khoảng 2 người phụ thuộc, cho thấy ảnh hưởng của các yếu tố gia đình đến quyết định tài chính.
- Thời gian sử dụng dịch vụ của khách hàng, hay thời gian "on books", trung bình là 36 tháng, cho thấy sự gắn bó của họ với nhà cung cấp thẻ.
- Mỗi khách hàng thường duy trì khoảng 4 mối quan hệ với các nhà cung cấp thẻ tín dụng, điều này phản ánh mức độ tương tác và sự lựa chọn của họ trong thị trường.
- Hạn mức tín dụng trung bình mà các tổ chức tài chính cung cấp lên tới 8.492,77 USD. Tuy nhiên, độ lệch chuẩn lớn cho thấy sự biến động đáng

kê trong hạn mức tín dụng, với khoảng giá trị trải dài từ 1.438 USD đến 34.516 USD.

- Tổng số tiền giao dịch trung bình của một khách hàng đạt 4.394,30 USD, cho thấy mức độ chi tiêu khá cao.
- Cuối cùng, mỗi khách hàng thực hiện trung bình 65 giao dịch, điều này cho thấy tần suất hoạt động tài chính của họ trong hệ thống.

3.3. Kiểm tra các biến phân loại trong Dataframe

- Theo mô tả, ta có các biến phân loại như sau: Education level, Marital Status, Income Category, Card Category.



```
1 #Xem bien phan loai Education_level
2 Education_counts = data.groupBy("Education_Level").count()
3 Education_counts.show()
4
5 #Xem bien phan loai Marital_Status
6 Marital_counts = data.groupBy("Marital_Status").count()
7 Marital_counts.show()
8
9 #Xem bien phan loai Income_Category
10 Income_counts = data.groupBy("Income_Category").count()
11 Income_counts.show()
12
13 #Xem bien phan loai Card_Category
14 Card_counts = data.groupBy("Card_Category").count()
15 Card_counts.show()
```

```
+-----+-----+
|Education_Level|count|
+-----+-----+
|    High School| 2013|
|      Unknown| 1519|
| Uneducated| 1487|
| Post-Graduate| 516|
|   Doctorate| 451|
|   Graduate| 3128|
|   College| 1013|
+-----+-----+
+-----+-----+
|Marital_Status|count|
+-----+-----+
|      Unknown| 749|
|     Married| 4687|
| Divorced| 748|
|   Single| 3943|
+-----+-----+
+-----+-----+
|Income_Category|count|
+-----+-----+
|      $120K +| 727|
| $60K - $80K| 1402|
| $80K - $120K| 1535|
|      Unknown| 1112|
| $40K - $60K| 1790|
| Less than $40K| 3561|
+-----+-----+
+-----+-----+
|Card_Category|count|
+-----+-----+
|    Platinum| 20|
|      Silver| 555|
|       Blue| 9436|
|      Gold| 116|
+-----+-----+
```

3.4. Xóa các cột bị thiếu hoặc dư thừa

- Ta sẽ thực hiện xóa các khoảng trắng và chuyển về chữ in hoa, xóa các dòng có giá trị “UNKNOWN” hoặc NULL trong cột chuỗi và chuyển về chữ in hoa.



```
1 # Chuẩn hóa giá trị các cột chuỗi: xóa khoảng trắng, chuyển về chữ in hoa
2 for column in data.columns:
3     if isinstance(data.schema[column].dataType, StringType): # Chỉ áp dụng cho cột kiểu string
4         data = data.withColumn(column, trim(upper(col(column))))
5
6 # Xóa các dòng có giá trị 'UNKNOWN' hoặc null chỉ trong cột chuỗi
7 columns_to_check = [col(column) for column in data.columns if isinstance(data.schema[column].dataType, StringType)]
8 filter_condition = ~((columns_to_check[0] == "UNKNOWN") | columns_to_check[0].isNull()) # Điều kiện cho cột đầu tiên
9
10 for column in columns_to_check[1:]: # Thêm điều kiện cho các cột còn lại
11     filter_condition &= ~((column == "UNKNOWN") | column.isNull())
12
13 data = data.filter(filter_condition)
14
15 # Hiển thị DataFrame sau khi xử lý
16 print("DataFrame sau khi loại bỏ 'Unknown' và null:")
17 data.show()
```

CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon
768805383	EXISTING CUSTOMER	45	M	3	HIGH SCHOOL	MARRIED	\$60K - \$80K	BLUE	39	5	
818770008	EXISTING CUSTOMER	49	F	5	GRADUATE	SINGLE	LESS THAN \$40K	BLUE	44	6	
713982108	EXISTING CUSTOMER	51	M	3	GRADUATE	MARRIED	\$80K - \$120K	BLUE	36	4	
709106358	EXISTING CUSTOMER	48	M	3	UNEDUCATED	MARRIED	\$60K - \$80K	BLUE	21	5	
713061558	EXISTING CUSTOMER	44	M	2	GRADUATE	MARRIED	\$40K - \$60K	BLUE	36	3	
710930508	EXISTING CUSTOMER	37	M	3	UNEDUCATED	SINGLE	\$60K - \$80K	BLUE	36	5	
719661558	EXISTING CUSTOMER	48	M	2	GRADUATE	SINGLE	\$80K - \$120K	BLUE	36	6	
710599683	EXISTING CUSTOMER	56	M	1	COLLEGE	SINGLE	\$80K - \$120K	BLUE	36	3	
712396908	EXISTING CUSTOMER	57	F	2	GRADUATE	MARRIED	LESS THAN \$40K	BLUE	48	5	
709967358	EXISTING CUSTOMER	48	M	4	POST-GRADUATE	SINGLE	\$80K - \$120K	BLUE	36	6	
806160108	EXISTING CUSTOMER	61	M	1	HIGH SCHOOL	MARRIED	\$40K - \$60K	BLUE	56	2	
806165208	EXISTING CUSTOMER	47	M	1	DOCTORATE	DIVORCED	\$60K - \$80K	BLUE	42	5	
708508758	ATTRITED CUSTOMER	62	F	0	GRADUATE	MARRIED	LESS THAN \$40K	BLUE	49	2	
784725333	EXISTING CUSTOMER	41	M	3	HIGH SCHOOL	MARRIED	\$40K - \$60K	BLUE	33	4	
771071958	EXISTING CUSTOMER	41	F	3	GRADUATE	SINGLE	LESS THAN \$40K	BLUE	28	6	
806624208	EXISTING CUSTOMER	47	M	4	HIGH SCHOOL	MARRIED	\$40K - \$60K	BLUE	42	6	
712991888	EXISTING CUSTOMER	53	M	2	UNEDUCATED	MARRIED	\$60K - \$80K	BLUE	48	2	
709029408	EXISTING CUSTOMER	41	M	4	GRADUATE	MARRIED	\$60K - \$80K	BLUE	36	4	
788658483	EXISTING CUSTOMER	53	F	2	COLLEGE	MARRIED	LESS THAN \$40K	BLUE	38	5	
787937058	EXISTING CUSTOMER	58	M	0	GRADUATE	MARRIED	\$80K - \$120K	BLUE	49	6	

- Ta xem số dòng dữ liệu sau khi xóa:



```
1 #Hien thi so luong datapoint
2 num_data_points = data.count()
3 print(f"Number of datapoints: {num_data_points}")
```

Number of datapoints: 7081

- Sau đó, ta tiếp tục xóa các thuộc tính không cần thiết:



```
1 #Xoa cac thuoc tinh khong can thiet
2 data = data.drop(
3     "CLIENTNUM",
4     "Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1",
5     "Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2",
6 )
```

summary	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon
count	7081	7081	7081	7081	7081	7081	7081	7081	7081	7081	7081
mean	NULL	46.347691004095466	NULL	2.33780539471826	NULL	NULL	NULL	NULL	NULL	3.819375794379325	2.34
stddev	NULL	8.04122486197091	NULL	1.291649051957562	NULL	NULL	NULL	NULL	NULL	1.544442823830733	0.99
min	ATTRITED CUSTOMER	26	F	0	COLLEGE	DIVORCED	\$120K +	BLUE	13	1	
max	EXISTING CUSTOMER	73	M	5	UNEDUCATED	SINGLE	LESS THAN \$40K	SILVER	56	6	

3.5. Xử lý một số giá trị ngoại lai

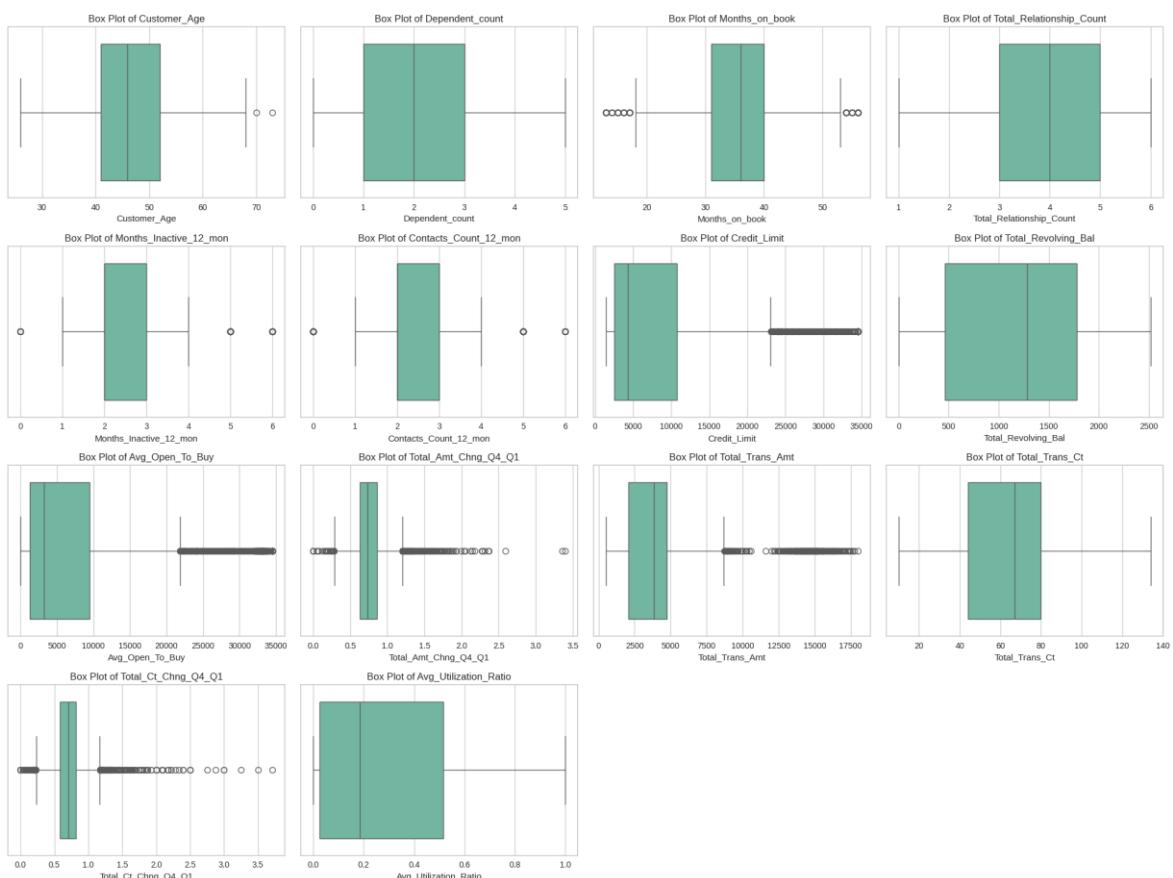
- Xác định ngoại lai:



```

1 fig, axes = plt.subplots(4, 4, figsize=(20, 15)) # Tạo lưới biểu đồ 4x4
2 k = 0 # Biến đếm để duyệt qua danh sách cột
3
4 for i in range(4): # Duyệt qua các hàng
5     for j in range(4): # Duyệt qua các cột
6         if k < len(columns_list): # Nếu còn cột cần vẽ
7             sns.boxplot(
8                 data=data_pandas,
9                 x=columns_list[k],
10                ax=axes[i, j],
11                palette='Set2'
12             )
13             axes[i, j].set_title(f'Box Plot of {columns_list[k]}') # Đặt tiêu đề biểu đồ
14             k += 1 # Chuyển sang cột tiếp theo
15         else: # Nếu hết cột, ẩn các ô còn lại
16             axes[i, j].axis('off')
17
18 plt.tight_layout() # Tự động điều chỉnh khoảng cách giữa các biểu đồ

```



⇒ Nhận xét:

- Những thuộc tính có nhiều outlier rõ rệt và cần xử lý:
 - Credit_Limit: Có rất nhiều outlier ở phần giá trị cao. Việc này có thể ảnh hưởng lớn đến các mô hình phân tích.
 - Avg_Open_To_Buy: Rất nhiều giá trị lớn vượt xa ngưỡng Q3, gây méo phân phối.
 - Total_Amt_Chng_Q4_Q1: Xuất hiện một số outlier giá trị rất cao và thấp.
 - Total_Ct_Chng_Q4_Q1: Có outlier ở cả hai phía dưới Q1 và trên Q3.
 - Total_Trans_Amt: Một vài outlier rất cao nằm cách xa Q3.
 - Months_on_book: Có outlier ở cả hai phía dưới Q1 và trên Q3.
- Những thuộc tính có outlier nhưng có thể giữ lại:
 - Customer_Age: Số lượng outlier rất ít và không gây ảnh hưởng lớn.
 - Months_Inactive_12_mon: Các outlier khá nhỏ và không đáng kể.
 - Contacts_Count_12_mon: Có một vài outlier nhưng vẫn nằm trong khoảng hợp lý.



```

1 # Danh sách các cột cần xử lý outlier
2 col_To_Process_Outlier = [
3     'Months_on_book',
4     'Credit_Limit',
5     'Avg_Open_To_Buy',
6     'Total_Amt_Chng_Q4_Q1',
7     'Total_Trans_Amt',
8     'Total_Ct_Chng_Q4_Q1'
9 ]
10
11 # Loại bỏ outliers cho từng cột
12 for column in col_To_Process_Outlier:
13     # Tính toán Q1 và Q3
14     bounds = data.approxQuantile(column, [0.25, 0.75], 0.001) # approxQuantile để lấy Q1 và Q3
15     Q1, Q3 = bounds[0], bounds[1]
16     IQR = Q3 - Q1
17
18     # Tính ngưỡng Lower và Upper
19     lower_bound = Q1 - 1.5 * IQR
20     upper_bound = Q3 + 1.5 * IQR
21
22     # Lọc dữ liệu trong khoảng giá trị hợp lệ
23     data = data.filter((col(column) >= lower_bound) & (col(column) <= upper_bound))
24
25 # In số lượng dòng sau khi xử lý outliers
26 print("Số dòng sau khi loại bỏ outliers cho tất cả các cột:", data.count())

```

Số dòng sau khi loại bỏ outliers cho tất cả các cột: 4935

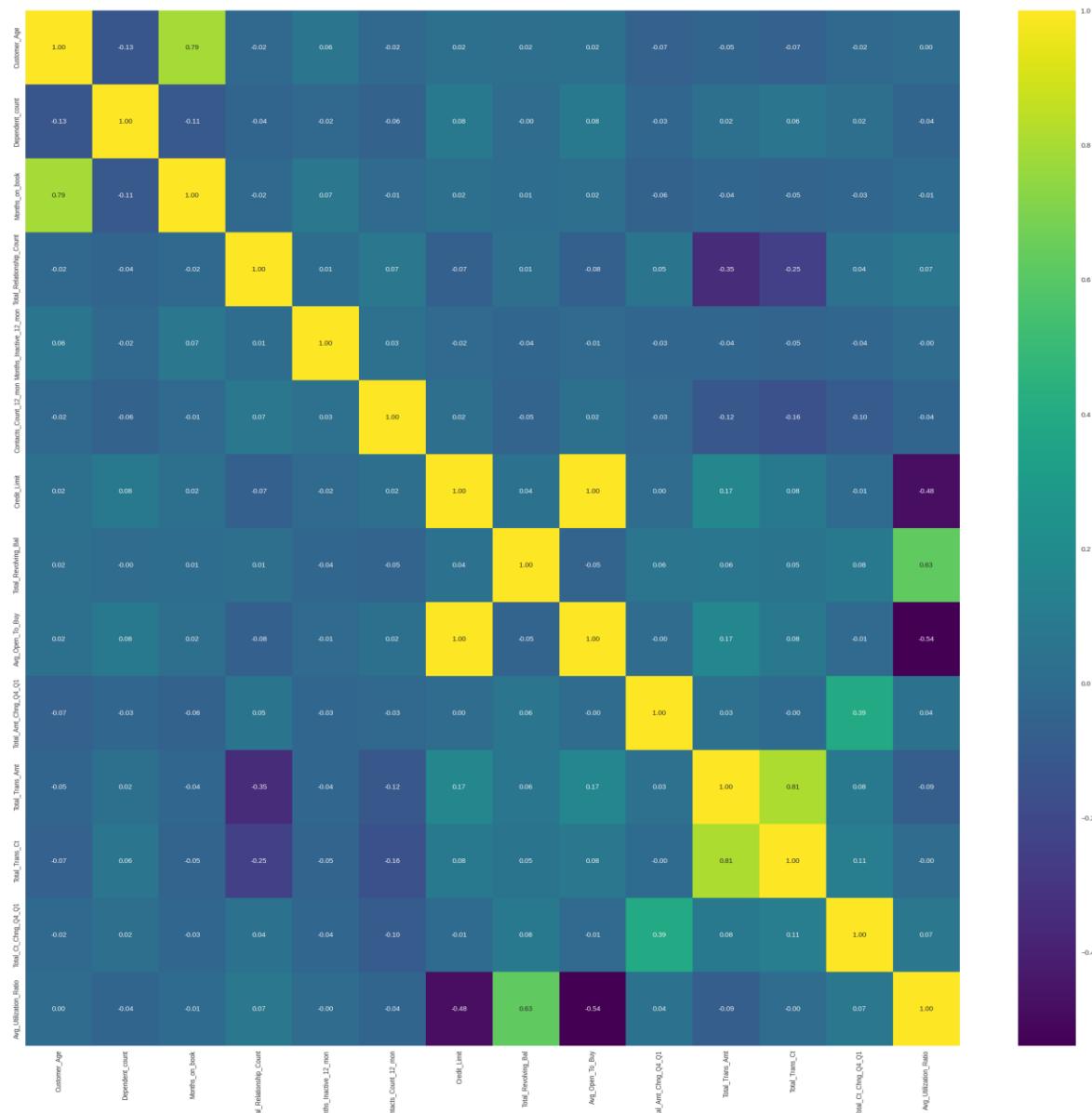
3.6. Kiểm tra độ tương quan của bộ Dataset



```

1 data_numeric = data_pandas.select_dtypes(include=['int', 'double'])
2 corrrmat = data_numeric.corr()
3 # Plot the heatmap
4 plt.figure(figsize=(30, 30))
5 #sns.heatmap(corrrmat, annot=True, cmap='coolwarm', center=0)
6 sns.heatmap(corrrmat, annot=True, annot_kws={'size':10}, fmt='.2f', cmap='viridis')
7 plt.show()

```



⇒ Nhận xét:

- **Tuổi của khách hàng (Customer Age):** có mối quan hệ tương quan dương cao (0.79) với số tháng hoạt động (**Months on Book**). Điều này là do khi khách hàng lớn tuổi hơn, họ thường duy trì tài khoản trong thời gian dài hơn so với khách hàng trẻ tuổi.
- **Tổng số tiền giao dịch (Total Transaction Amount)** có mối tương quan dương mạnh (0.81) với tổng số lượng giao dịch (**Total Transaction Count**). Điều này hợp lý vì khi một khách hàng thực hiện nhiều giao dịch, số lượng giao dịch sẽ tăng theo.

- **Tỷ lệ sử dụng tín dụng (Utilization Ratio)** là số tiền tín dụng quay vòng mà khách hàng sử dụng chia cho tổng số tín dụng sẵn có. Do đó, Tỷ lệ sử dụng trung bình (**Average Utilization Ratio**) có mối tương quan dương cao (0.63) với tổng hóa đơn quay vòng (**Total Revolving Bill**). Đây cũng là lý do tại sao Giới hạn tín dụng (**Credit Limit**) lại có mối tương quan âm đáng kể (-0.48) với tỷ lệ sử dụng trung bình, vì khi giới hạn tín dụng cao hơn, tỷ lệ sử dụng tín dụng thường giảm.
- "**Open to Buy**" là sự chênh lệch giữa giới hạn tín dụng được cấp cho tài khoản của chủ thẻ và số dư hiện tại trên tài khoản đó. Vì tỷ lệ sử dụng tín dụng (Utilization Ratio) gián tiếp phụ thuộc vào giới hạn tín dụng, nên "Open to Buy" cũng gián tiếp phụ thuộc vào tỷ lệ sử dụng tín dụng. Đây là lý do tại sao chúng có mối tương quan âm.

3.7. Mã hóa các đặc trưng (features) phân loại bằng StringIndexer

- Xem lại các biến phân loại (các biến có kiểu dữ liệu không phải Integer)



```
1 col_types = data.dtypes
2 categorical_cols = [col_name for col_name, data_type in col_types if data_type == 'string']
3 print(f"Các biến phân loại trong tập dữ liệu: {categorical_cols}")
```

⇒ Kết quả: Các biến phân loại trong tập dữ liệu: ['Attrition_Flag', 'Gender', 'Education_Level', 'Marital_Status', 'Income_Category', 'Card_Category']



```
1 # Khởi tạo StringIndexer cho từng cột
2 indexers = [StringIndexer(inputCol=col_name, outputCol=col_name + "_index").fit(data) for col_name in categorical_cols]
3
4 # Áp dụng StringIndexer lên DataFrame
5 for indexer in indexers:
6     data = indexer.transform(data)
7
8 # Chuyển đổi kiểu dữ liệu từ float sang int
9 for col_name in categorical_cols:
10    data = data.withColumn(col_name + "_index", col(col_name + "_index").cast("int"))
11
12 # Xóa các cột gốc (dạng chuỗi)
13 data = data.drop(*categorical_cols)
14
15 print("Tất cả thuộc tính bây giờ là kiểu số")
```

⇒ Kết quả: Tất cả thuộc tính bây giờ là kiểu số

	Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revolving_Bal	Avg_Open_To_Buy	Total_Amt_Chng_Q4_Q1	
57	2	48	5	2	2	2436.0	680	1756.0	1.15		
62	0	49	2	3	3	1438.3	0	1438.3	1.04		
41	3	28	6	1	2	7768.0	1669	6099.0	0.79		
47	4	42	6	0	0	4785.0	1362	3423.0	0.73		
58	0	49	6	2	2	12555.0	1696	10859.0	0.51		
55	1	36	4	2	1	3520.0	1914	1686.0	0.51		
52	1	40	5	1	1	4745.0	1227	3518.0	0.62		
54	1	40	2	3	1	1438.3	808	630.3	0.99		
49	3	36	2	2	0	2882.0	2363	439.0	0.71		
39	1	33	5	3	3	5926.0	1251	4675.0	0.94		
53	2	44	4	2	2	14734.0	1634	13100.0	0.98		
57	2	52	5	3	3	6584.0	1817	4767.0	0.6		
44	2	20	6	3	3	2084.0	1468	616.0	1.00		
55	2	42	5	3	3	2216.0	1034	1182.0	0.75		
53	2	36	5	3	2	5272.0	1515	3797.0	0.85		
37	3	29	4	4	2	7098.0	1801	5237.0	0.73		
48	3	30	6	1	2	2536.0	1823	713.0	0.70		
52	2	47	5	3	0	3085.0	1910	1175.0	0.92		
43	3	35	5	2	3	4026.0	0	4026.0	0.48		
45	1	36	4	4	3	6576.0	0	6576.0	0.57		

- Đổi tên các cột phân loại vừa đổi từ chữ sang số



```
1 # Đổi tên
2 data = data.withColumnRenamed("Gender_index", "Gender")
3 data = data.withColumnRenamed("Attrition_Flag_index", "Attrition_Flag")
4 data = data.withColumnRenamed("Education_Level_index", "Education_Level")
5 data = data.withColumnRenamed("Marital_Status_index", "Marital_Status")
6 data = data.withColumnRenamed("Income_Category_index", "Income_Category")
7 data = data.withColumnRenamed("Card_Category_index", "Card_Category")
```

3.8. Chuẩn hóa (scale) các đặc trưng bằng Z – Score



1 print(data.columns)

- Xem tên các cột cần chuyển hóa:
 - Kết quả: ['Customer_Age','Dependent_count','Months_on_book', 'Total_Relationship_Count', 'Months_Inactive_12_mon', 'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal', 'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt', 'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio', 'Attrition_Flag', 'Gender', 'Education_Level', 'Marital_Status', 'Income_Category', 'Card_Category']



```

1 # Tính toán giá trị trung bình và độ lệch chuẩn cho mỗi đặc trưng
2 # Tạo một mảng chứa giá trị trung bình và độ lệch chuẩn của mỗi đặc trưng
3 # Tính toán giá trị trung bình và độ lệch chuẩn cho mỗi đặc trưng
4 newdata = data
5 summary = newdata.select(
6     [mean(c).alias(c + '_mean') for c in newdata.columns] +
7     [stddev(c).alias(c + '_stddev') for c in newdata.columns]
8 ).collect()[0]
9
10 # Tạo dictionary để lưu trữ giá trị trung bình và độ lệch chuẩn
11 means = {col_name: summary[col_name + '_mean'] for col_name in newdata.columns}
12 std devs = {col_name: summary[col_name + '_stddev'] for col_name in newdata.columns}
13 print(means)
14 print(std devs)

```

{'Customer_Age': 46.3732277963526, 'Dependent_count': 2.3775075987841947, 'Months_on_book': 36.03262411347518, 'Total_Relationship_Count': 3.952988855116515, 'Months_Inactive_12_mon': 0.0000000000000001, 'Customer_Age': 7.64056851627595, 'Dependent_count': 1.2723640534248515, 'Months_on_book': 7.251151279342349, 'Total_Relationship_Count': 1.4961844062617884, 'Months_Inactive_12_mon': 0.0000000000000001}

- Kết quả:



```

1 # Kết quả như ý muốn nên chạy cho toàn bộ dataset
2 for col_name in newdata.columns:
3     newdata = newdata.withColumn(col_name, (col(col_name) - means[col_name]) / std devs[col_name])
4
5 # Hiển thị kết quả
6 newdata.show()

```

Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revolving_Bal	Avg
1.398832069319065	-0.2966977869015152	1.6504104555946064	0.6997874997905097	-0.36526942637339965	-0.40313419047554205	-0.6459080626473273	-0.563987843003467	-0.515
2.0452336350464786	-1.8685749510013294	3.788319500499691	-1.3053129326458166	0.6400873048896248	0.5048837145891771	-0.917899305172076	-1.3990505617349107	-0.6861
-0.7032529409544788	-1.107722425970833	1.3681543106026184	-1.370626157637618	-0.40313419047554205	0.8076925237641013	0.6565371991659271	0.6619	
0.8822093790293764	-1.2751793771982989	0.8229556461640999	1.3681543106026184	-2.3798288989012393	-2.219170954060498	-0.005527759029396..	0.2735309423675105	0.066115
1.5217123824748675	-1.8686749510013294	1.788319500499691	1.3681543106026184	-0.36526942637339965	-0.40313419047554205	2.1127161568130716	0.6836941012296315	1.9570
1.1290714430461595	-1.0826363689514222	-0.00449916326640...	0.03142068897840094	-0.36526942637339965	-1.3115209554062611	-0.35038986390952	0.9514053845588002	-0.5604
0.7364305036174512	-1.0826363689514222	0.5471373763539306	0.6997874997905097	-1.370626157637618	-1.3115209554062611	-0.01643248987463...	0.10774643204898861	-0.04827
0.9981911299812568	-1.0826363689514222	0.5471373763539306	-1.3053129326458166	0.6400873048896248	-1.3115209554062611	-0.917899305172076	-0.4067995666433128	-0.8259
0.34378956418874307	0.48924879514839186	-0.00449916326640...	-0.3053129326458166	-0.36526942637339965	-2.219170954060498	-0.5461297778336396	1.5027923855441865	-0.8780
-0.4650125674208242	-1.0826363689514222	-0.4182265679816608	0.6997874997905097	0.6400873048896248	0.5048837145891771	0.3055296805214345	0.1372192338833925	0.274
0.867310816760354	-0.2966977869015152	1.0987739159742684	0.03142068897840094	-0.36526942637339965	-0.40313419047554205	2.706751354398387	0.6075560298240881	2.5668
1.390832069319065	-0.2966977869015152	2.2028469952149444	0.6997874997905097	0.6400873048896248	0.5048837145891771	0.4849124985744668	0.8322861438114177	0.29950
-0.31061200152577856	-0.2966977869015152	-2.211045321747759	1.3681543106026184	0.6400873048896248	0.5048837145891771	-0.7418696917577592	0.4037204838027944	-0.8298
1.1290714430461595	-0.2966977869015152	0.8229556461640999	0.6997874997905097	0.6400873048896248	0.5048837145891771	-0.7058840808413472	-0.12926464603609943	-0.6758
0.867310816760354	-0.2966977869015152	-0.00449916326640...	0.6997874997905097	0.6400873048896248	-0.40313419047554205	1.2723733552649333	0.4614200540618354	0.02475
-1.2267741935260896	0.48924879514839186	-0.9969831076919987	0.03142068897840094	1.64544048361532463	-0.40313419047554205	0.6086811966657624	0.8126376992551485	0.4274
0.34378956418874307	0.48924879514839186	-0.8319539726969143	1.3681543106026184	-1.370626157637618	-0.40313419047554205	-0.618646236195008	0.8396543422700187	-0.8834
0.7364305036174512	-0.2966977869015152	1.51250132068952	0.6997874997905097	0.6400873048896248	-0.40313419047554205	0.9464932559197329	-0.6777	
-0.441204923146686733	0.48924879514839186	-0.1424082981749198	0.6997874997905097	0.5048837145891771	0.212440522063739965	-0.212440522063739965	-0.212440522063739965	0.09794
-0.17973168838286782	-1.0826363689514222	-0.00449916326640...	0.03142068897840094	1.64544048361532463	0.5048837145891771	0.4827315524583206	-0.3990505617349107	0.7917

3.9. Thực hiện giảm chiều dữ liệu bằng PCA

- Chuyển dữ liệu về dạng vector



```

1 # Sử dụng VectorAssembler để kết hợp các cột đặc trưng thành một cột duy nhất dạng vector
2 assembler = VectorAssembler(inputCols=newdata.columns, outputCol="features")
3 df_features = assembler.transform(newdata)

```

Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revolving_Bal	Avg.
1, 390832069331965	-0, 2966977869015152	1, 6504104555946064	0, 6997874997905097	-0, 40313419047554205	-0, 6459080626473273	-0, 563987843093467	-0, 51%	
2, 0452336350464786	-1, 8685749510013294	1, 788319590499691	-1, 3053129326458166	0, 6400873048896248	0, 5048837145891771	-0, 9178993051572076	-1, 3990505617349107	
-0, 70252940954788	0, 48924079514839186	-1, 107722425078033	1, 3681543106026184	-1, 370626157637618	-0, 40313419047554205	0, 8876925237641013	0, 6505371991659271	
0, 08202893790293764	1, 2751793771982989	0, 829556461640996	1, 3681543106026184	-2, 3759828889012393	2, 21917000060498	-0, 00552775929390...1	0, 066115	
1, 5217123824748675	1, 8685749510013204	1, 788319590499691	1, 3681543106026184	0, 3652694263739965	-0, 40313419047554205	2, 1127161560320751	0, 682360410322051	
1, 1290714430461595	-1, 0826363689514222	-0, 00449916326640...1	0, 03142068897840094	-0, 3652694263739965	-1, 3111520955402611	-0, 35038986390952	0, 0514053845588002	
0, 7374305036174512	-1, 0826363689514222	0, 547137376539306	0, 6997874997905097	-1, 370626157637618	-1, 3111520955402611	-0, 0162248987463...1	0, 1977464320480861	
0, 9981911299032568	-1, 0826363689514222	0, 547137376539306	-1, 3053129326458166	0, 6400873048896248	-1, 3111520955402611	-0, 9178993051572076	-0, 60611	
0, 34378956418874307	0, 48924079514839186	-0, 00449916326640...1	-1, 3053129326458166	-0, 3652694263739965	2, 21917000060498	-0, 5461297778336396	0, 66198	
-0, 9650125672408242	-1, 0826363689514222	-0, 418226567266408	0, 6997874997905097	0, 6400873048896248	0, 5048837145891771	0, 305296805214435	0, 2735309422675105	
0, 867310816760354	-1, 0826363689514222	-0, 418226567266408	0, 03142068897840094	0, 3652694263739965	-0, 40313419047554205	0, 305296805214435	0, 066115	
1, 390832069331965	0, 4966977869015152	2, 2620890997840444	0, 6997874997905097	0, 6400873048896248	0, 5048837145891771	0, 305296805214435	0, 2735309422675105	
-0, 31061200152577056	0, 4966977869015152	-2, 211043231747759	1, 3681543106026184	0, 6400873048896248	0, 5048837145891771	-0, 741869617577592	0, 4037024838927444	
1, 1290714430461595	0, 4966977869015152	0, 829556461640996	0, 6997874997905097	0, 6400873048896248	0, 5048837145891771	-0, 7058480808413472	-0, 12926401603606943	
0, 867310816760354	0, 4966977869015152	-0, 00449916326640...1	0, 6997874997905097	0, 6400873048896248	0, 5048837145891771	0, 4614200540618354	0, 02475%	
-1, 26777419352608996	0, 48924079514839186	-0, 9968831077601987	0, 03142068897840094	1, 64544846361532463	-0, 40313419047554205	0, 8126376905251485	0, 4274:	
0, 34378956418874307	0, 48924079514839186	-0, 819539726969143	1, 3681543106026184	-1, 370626157637618	-0, 40313419047554205	-0, 6186462361955001	0, 8396543442708187	
0, 7364303903174512	-0, 2966977869015152	3, 51250132068952	0, 6997874997905097	0, 6400873048896248	-2, 21917000060498	0, 4689788808749846	0, 6777%	
-0, 44149231466866733	0, 48924079514839186	-0, 1424082981714918	-0, 3652694263739965	0, 5048837145891771	0, 46448837145891771	-0, 21244502206327412	-1, 3990505617349107	
-0, 17973168838286782	-1, 0826363689514222	-0, 00449916326640...1	0, 03142068897840094	1, 64544846361532463	0, 5048837145891771	0, 482731552458326	0, 3990505617349107	

- Sử dụng hàm locPcs để tạo ra các UDF để trích xuất các giá trị thành phần chính từ vector pca_features



```

1 # Tạo hàm triết xuất user-defined function để lọc ra các PCs
2 def locPcs(index):
3     # Chỗ này lấy giá trị thứ i của vector rồi trả về kiểu float
4     def locIndex(vector):
5         return float(vector[index])
6         # Chỗ này trả về một hàm do người dùng xác định udf
7     return udf(locIndex, DoubleType())
8
9 # Sử dụng hàm locPcs để add vào dữ liệu từ trước 3 cột (tương ứng với 3 PCs)
10 pca_df = pca_result.withColumn("pc1", locPcs(0)(col("pca_features")))\ \
11     .withColumn("pc2", locPcs(1)(col("pca_features")))\ \
12     .withColumn("pc3", locPcs(2)(col("pca_features")))

```

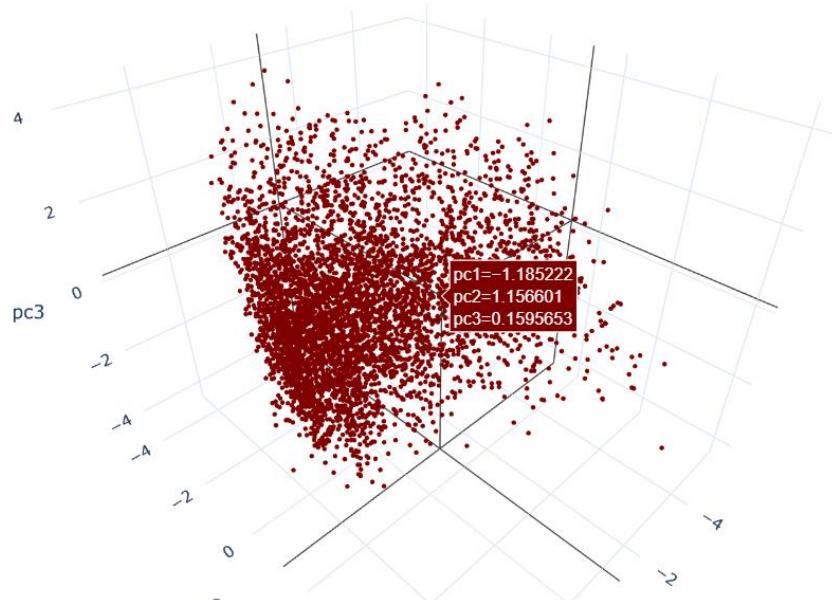
Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revolving_Bal	Avg.
1, 390832069331965	-0, 2966977869015152	1, 6504104555946064	0, 6997874997905097	-0, 40313419047554205	-0, 6459080626473273	-0, 563987843093467	-0, 51%	
2, 0452336350464786	-1, 8685749510013294	1, 788319590499691	-1, 3053129326458166	0, 6400873048896248	0, 5048837145891771	-0, 9178993051572076	-0, 60611	
-0, 70252940954788	0, 48924079514839186	-1, 107722425078033	1, 3681543106026184	-1, 370626157637618	-0, 40313419047554205	0, 8876925237641013	0, 66198	
0, 08202893790293764	1, 2751793771982989	0, 829556461640996	1, 3681543106026184	-2, 3759828889012393	2, 21917000060498	-0, 00552775929390...1	0, 066115	
1, 5217123824748675	1, 8685749510013204	1, 788319590499691	1, 3681543106026184	0, 3652694263739965	-0, 40313419047554205	2, 1127161560320751	0, 682360410322051	
1, 1290714430461595	-1, 0826363689514222	-0, 00449916326640...1	0, 03142068897840094	-0, 3652694263739965	-1, 3111520955402611	-0, 35038986390952	0, 0514053845588002	
0, 7374305036174512	-1, 0826363689514222	0, 547137376539306	0, 6997874997905097	-1, 370626157637618	-1, 3111520955402611	-0, 0162248987463...1	0, 1977464320480861	
0, 9981911299032568	-1, 0826363689514222	0, 547137376539306	-1, 3053129326458166	0, 6400873048896248	-0, 40313419047554205	-0, 9178993051572076	-0, 60611	
0, 34378956418874307	0, 48924079514839186	-0, 00449916326640...1	-1, 3053129326458166	-0, 3652694263739965	-1, 3111520955402611	-0, 9178993051572076	0, 04027%	
-0, 9650125672408242	-1, 0826363689514222	-0, 418226567266408	0, 6997874997905097	0, 6400873048896248	-0, 40313419047554205	0, 80876925237641013	0, 6505371991659271	
0, 867310816760354	-1, 0826363689514222	-0, 418226567266408	0, 03142068897840094	0, 6997874997905097	-0, 6186462361955001	0, 8396543442708187	0, 0834%	
1, 390832069331965	0, 4966977869015152	2, 2620890997840444	0, 6997874997905097	0, 6400873048896248	0, 5048837145891771	0, 21244502206327412	0, 3990505617349107	
-0, 31061200152577056	0, 4966977869015152	-2, 211043231747759	1, 3681543106026184	0, 6400873048896248	0, 5048837145891771	0, 487024838927444	0, 3990505617349107	
1, 1290714430461595	0, 4966977869015152	0, 829556461640996	0, 6997874997905097	0, 6400873048896248	0, 5048837145891771	0, 12926401603606943	0, 0758%	
0, 867310816760354	0, 4966977869015152	-0, 00449916326640...1	0, 6997874997905097	0, 6400873048896248	0, 5048837145891771	-0, 7058480808413472	-0, 12926401603606943	
-1, 26777419352608996	0, 48924079514839186	-0, 9968831077601987	0, 03142068897840094	1, 64544846361532463	-0, 40313419047554205	0, 8126376905251485	0, 04274:	
0, 34378956418874307	0, 48924079514839186	-0, 819539726969143	1, 3681543106026184	-1, 370626157637618	-0, 40313419047554205	-0, 6186462361955001	0, 0834%	
0, 7364303903174512	-0, 2966977869015152	3, 51250132068952	0, 6997874997905097	0, 6400873048896248	0, 5048837145891771	0, 9464932599197329	0, 0677%	
-0, 44149231466866733	0, 48924079514839186	-0, 1424082981714918	-0, 3652694263739965	0, 5048837145891771	0, 21244502206327412	0, 3990505617349107	0, 09794%	
-0, 17973168838286782	-1, 0826363689514222	-0, 00449916326640...1	0, 03142068897840094	1, 64544846361532463	0, 5048837145891771	0, 482731552458326	0, 3990505617349107	

- Trực quan hóa dữ liệu với không gian 3 chiều



```
1 # Chuyển dữ liệu về dạng pandas dataframe để có thể trực quan hóa
2 plot_pca_df = pca_df.toPandas()
3
4 # Xác định cột x, y, z tương ứng với các PCs
5 x = plot_pca_df["pc1"]
6 y = plot_pca_df["pc2"]
7 z = plot_pca_df["pc3"]
8
9 # Tạo biểu đồ 3D
10 import plotly.express as px
11 px.defaults.template = "plotly_white"
12 fig = px.scatter_3d(plot_pca_df, x, y, z, color_discrete_sequence=['maroon'])
13 fig.update_traces(marker=dict(size=2))
14 fig.update_layout(title="3D Projection Of Data In The Reduced Dimension", height=600, width=800)
15
16 # Hiển thị đồ thị
17 fig.show()
```

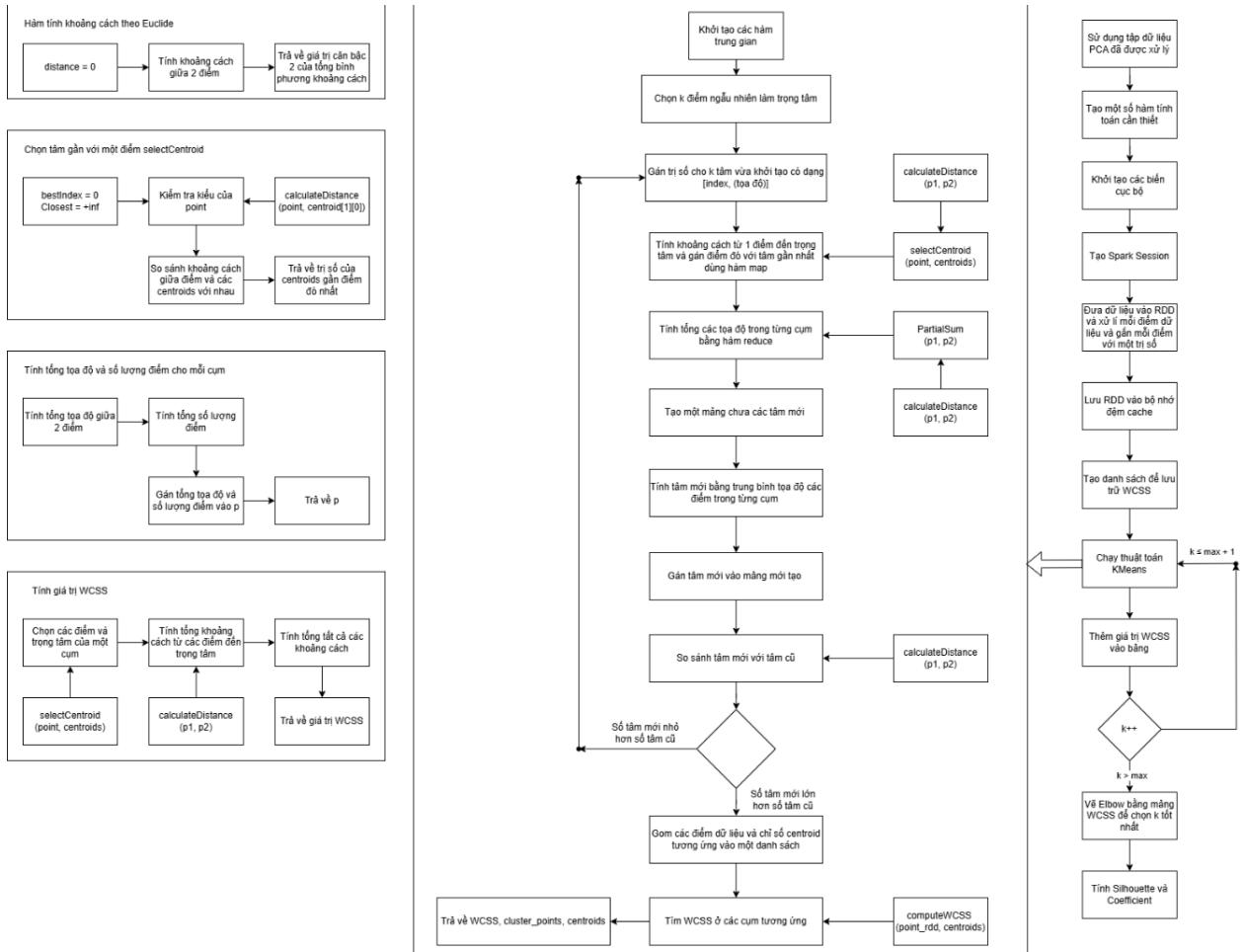
3D Projection Of Data In The Reduced Dimension



Phần 4. THUẬT TOÁN KHAI THÁC DỮ LIỆU

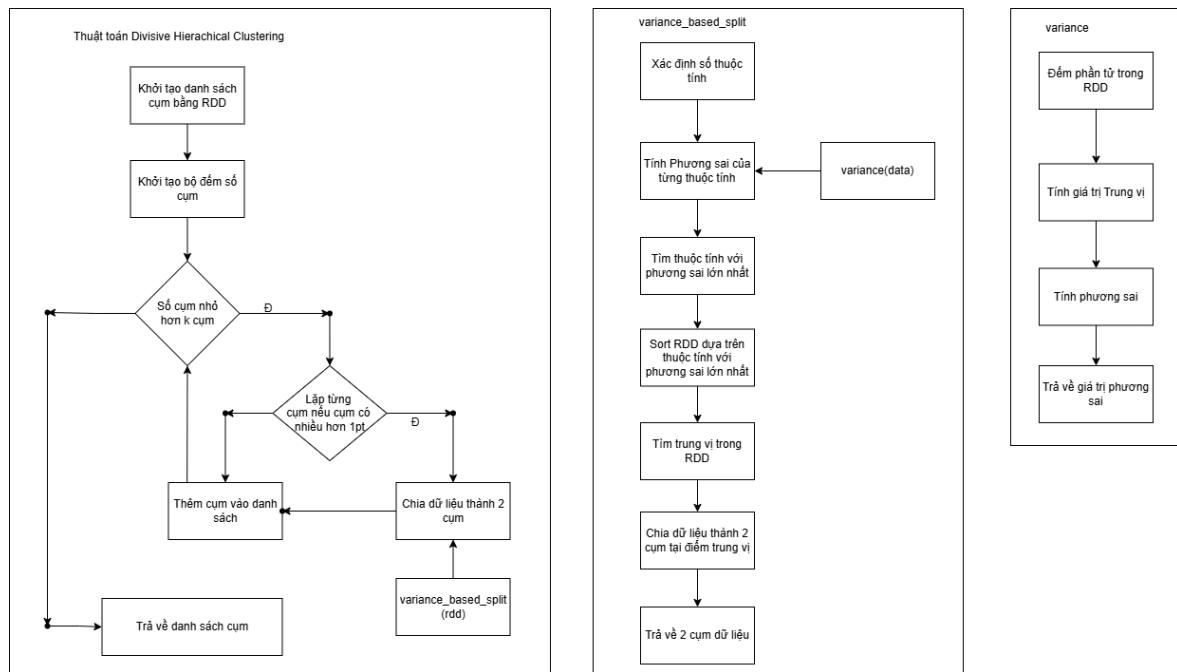
4.1. Đặc tả thuật toán

4.1.1. Thuật toán K – Means



Hình 4.1. Sơ đồ đặc tả các bước chạy thuật toán K-Means

4.1.2. Thuật toán Divisive Hierarchical Clustering



Hình 4.2. Sơ đồ đặc tả các bước chạy thuật toán DHC

4.2. Thực hiện thuật toán

4.2.1. Thuật toán K – means

4.2.1.1. Tạo các hàm tính toán trong K – means

- Tạo hàm tính khoảng cách giữa các điểm bằng công thức tính Euclidean tính khoảng cách trong không gian đa chiều



```

1 # Tạo hàm tính khoảng cách theo công thức Euclidean
2 def calculateDistance(p1, p2):
3     distance = 0
4     for i in range(len(p1)):
5         distance += (p1[i] - p2[i])**2
6     return np.sqrt(distance)

```

- Tạo hàm chọn trọng tâm bằng cách so sánh khoảng cách giữa điểm point và tất cả các trọng tâm xem điểm point được đưa vào gần với trọng tâm nào nhất.



```

1 # Tạo hàm chọn trọng tâm bằng cách so sánh khoảng cách giữa điểm point và tất cả các trọng tâm
2 # để xem điểm point được đưa vào gần với trọng tâm nào nhất
3 def selectCentroid(point, centroids):
4     bestIndex = 0
5     Closest = float("inf")
6     for centroid in centroids:
7         if isinstance(point, float):
8             tempDist = calculateDistance(point, centroid[1][0])
9         else:
10            tempDist = calculateDistance(point, centroid[1])
11        if tempDist < Closest:
12            Closest = tempDist
13            bestIndex = centroid[0]
14    return bestIndex

```

- Tính tổng các hợp phần của các điểm



```

1 from operator import add
2 # Tính tổng các hợp phần của các điểm
3 def partialSum(p1,p2):
4     coordinates_sum = list(map(add, p1[0], p2[0]))
5     points_number = p1[1] + p2[1]
6     p = [coordinates_sum, points_number]
7     return p

```

- Tính chỉ số WCSS



```

1 #Tính chỉ số WCSS
2 def computeWCSS(point_rdd, centroids):
3     #Chọn điểm dữ liệu và tâm centroids gần nó trong cụm
4     wcss = point_rdd.map(lambda point: (selectCentroid(point[0], centroids), point[0]))\
5                 .map(lambda x: calculateDistance(x[1], centroids[x[0]][1])**2)\\
6                 .sum()
7     return wcss

```

4.2.1.2. Tạo hàm chạy thuật toán K – Means

- Giai đoạn map: Tính toán khoảng cách từ mỗi điểm đến các centroid và gán điểm đó cho centroid gần nhất. Tạo cặp khóa key-value với key là chỉ số centroid và value là giá trị của điểm dữ liệu.

- Giai đoạn reduce: Tập hợp các điểm thuộc cùng một centroid và tính toán centroid mới bằng cách lấy trung bình tọa độ của các điểm dữ liệu.
- Giai đoạn tiếp theo: Lặp lại quá trình map và reduce cho đến khi các centroid hội tụ.



```

1 def kmean(k,lines,points_rdd):
2 # Khởi tạo các biến trung gian
3     convergedCentroids = 0 #Số trọng tâm đã hội tụ
4     centroids = [] # Trọng tâm hiện tại
5     new_centroids = [] # Trọng tâm mới được tính
6     iterations = 0 # Số vòng lặp
7     maxIterations = 100 # Số vòng lặp tối đa
8     wcss_values = [] # Biến lưu WCSS
9
10    # Khởi tạo các trọng tâm ban đầu
11    tmp = [line.split(",") for line in lines.takeSample(False, k)] #takeSample(False, k) chọn ngẫu nhiên K điểm từ dữ liệu để làm trọng tâm ban đầu.
12    for index, centroid in enumerate(tmp):
13        centroids += [[index, [float(string) for string in centroid]]] #centroids là danh sách các trọng tâm [index, tọa độ]
14    print("Các trọng tâm được chọn: ", centroids)
15    while maxIterations > iterations: #Lặp cho đến khi các trọng tâm hội tụ hoặc đạt đến số vòng lặp tối đa
16        iterations += 1
17
18        # MAP: Gán dữ liệu vào cụm gần nhất
19        # Hàm selectCentroid(point[0], centroids) tính khoảng cách giữa một điểm với tất cả các trọng tâm và chọn trọng tâm gần nhất.
20        mapped_rdd = points_rdd.keyBy(lambda point: selectCentroid(point[0], centroids))
21
22        # REDUCE: Tính trung bình để cập nhật trọng tâm mới
23        # reduceByKey kết hợp tất cả các điểm thuộc cùng một cụm để tính tổng tọa độ và số lượng điểm.
24        # Hàm partialSum(p1, p2) cộng dồn tọa độ và số lượng điểm trong cụm
25        reduced_rdd = mapped_rdd.reduceByKey(lambda p1, p2: partialSum(p1, p2))
26        reduced_points = reduced_rdd.collect() # Một danh sách chứa tổng tọa độ và số lượng điểm cho từng cụm

```



```

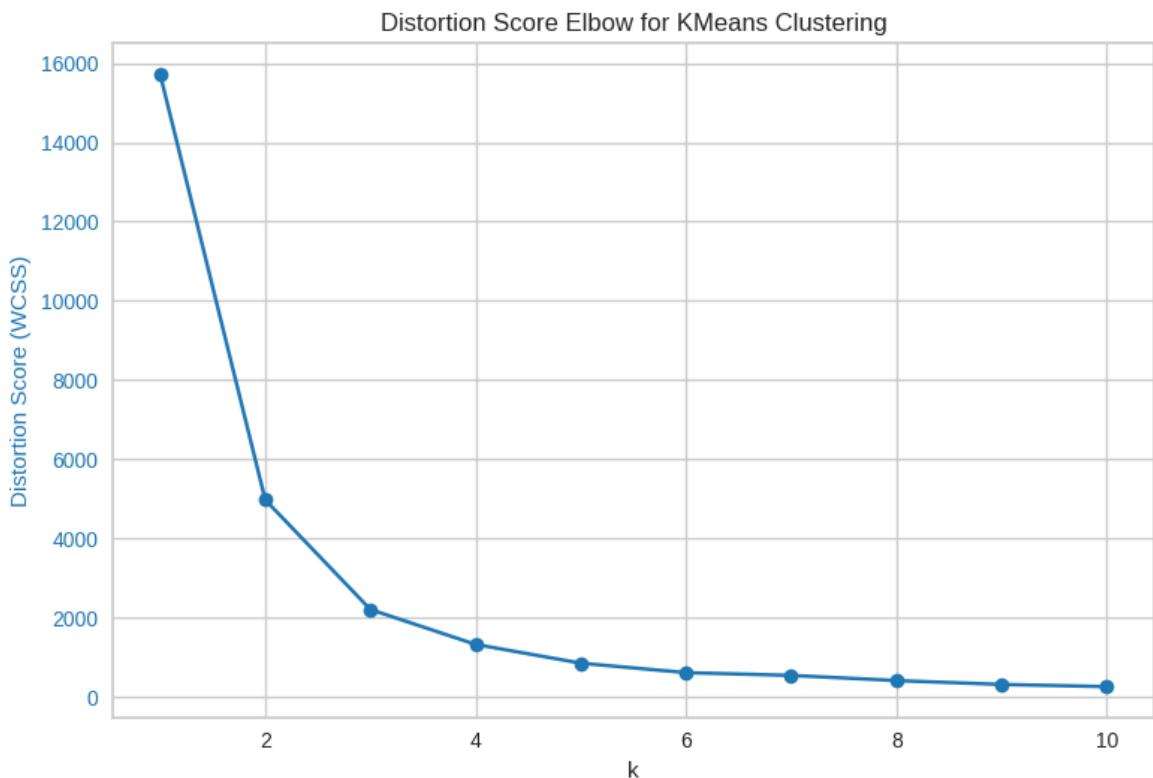
1 #Cập nhật trọng tâm mới
2     # Mỗi cụm sẽ tính tọa độ trung bình bằng cách lấy tổng tọa độ chia cho số lượng điểm
3     new_centroids = []
4     for index, reduced_point in enumerate(reduced_points):
5         converted_point = list(reduced_point)
6         centroid_index = converted_point[0]
7         centroid_coordinates = np.array(converted_point[1][0]) / converted_point[1][1]
8         new_centroid = [centroid_index, centroid_coordinates] #chứa trọng tâm mới được cập nhật
9         new_centroids.append(new_centroid)
10
11     # Kiểm tra hội tụ
12     # So sánh k/c trọng tâm cũ với trọng tâm mới = calculateDistance
13     # Nếu k/c < threshold thì trọng tâm đã hội tụ, sau đó đếm trọng tâm hội tụ
14     convergedCentroids = 0
15     for index, centroid in enumerate(centroids):
16         distance = calculateDistance(centroid[1], new_centroids[index][1])
17         if distance < threshold:
18             convergedCentroids += 1
19     centroids = new_centroids #Cập nhật lại các trọng tâm hiện tại bằng trọng tâm mới đã tính ở trên
20
21     #Dừng vòng lặp nếu hội tụ
22     percentage = len(centroids) / 100
23     if convergedCentroids > percentage:
24         print(f"Centroids converged for k = {k} in {iterations} iterations")
25         break
26     clustered_points = mapped_rdd.map(lambda x: (x[1], x[0])).collect()
27     wcss = computeWCSS(points_rdd, centroids)
28     return wcss,clustered_points,centroids #Trả về chỉ số WCSS, các cụm điểm và trọng tâm cuối cùng

```

- Ta tìm hệ số K:



```
1 import matplotlib.pyplot as plt
2
3 # Vẽ biểu đồ
4 fig, ax1 = plt.subplots()
5
6 # Trục y thứ nhất: WCSS
7 color = 'tab:blue'
8 ax1.set_xlabel('k')
9 ax1.set_ylabel('Distortion Score (WCSS)', color=color)
10 ax1.plot(range(1, max_k + 1), wcss_values, marker='o', color=color)
11 ax1.tick_params(axis='y', labelcolor=color)
12
13 # Tiêu đề và chú thích
14 plt.title('Distortion Score Elbow for KMeans Clustering')
15 fig.tight_layout()
16 plt.legend()
17 plt.show()
```



Hình 4.3. Kết quả Elbow Method trên bộ dữ liệu để tài

⇒ Nhận xét:

- Theo biểu đồ Elbow trên, để xác định giá trị tối ưu cho k trong thuật toán K-Means thì sẽ chọn điểm mà đường cong bắt đầu phẳng lại (điểm khuỷu)
 - Dựa trên biểu đồ thì giá trị của k trong khoảng 3 hoặc 4, vì sau đó độ chêch giảm chậm lại. Tuy nhiên ta sẽ chọn k = 4 vì khi từ 4 tới 5 Distortion chỉ giảm một khoảng nhỏ hơn so với từ 3 tới 4.
 - Nhóm sẽ chọn k = 4 để tính.
- Hàm tính toán Silhouette Coefficient



```
1 def silhouette_coefficient(clustered_points, centroids):  
2     """Tính toán silhouette coefficient."""  
3     silhouette_values = []  
4     for point_cluster, _ in clustered_points:  
5         point, cluster_index = point_cluster  
6         a = 0  
7         b = float('inf')  
8         cluster_distance_sum = 0  
9         cluster_size = 0  
10        for other_point_cluster, other_cluster_index in clustered_points:  
11            other_point, _ = other_point_cluster  
12            if cluster_index == other_cluster_index:  
13                cluster_distance_sum += calculateDistance(point, other_point)  
14                cluster_size += 1  
15            else:  
16                distance = calculateDistance(point, centroids[other_cluster_index][1])  
17                if distance < b:  
18                    b = distance  
19                a = cluster_distance_sum / cluster_size if cluster_size > 1 else 0  
20                silhouette = (b - a) / max(a, b)  
21                silhouette_values.append(silhouette)  
22    return np.mean(silhouette_values)
```

- Hàm tính WCSS, BCSS, Calinski – Harabasz Index



```

1 def calinski_harabasz_index(clustered_points, centroids):
2     """Tính chỉ số Calinski-Harabasz."""
3     # Tính toán WCSS (Within Cluster Sum of Squares)
4     wcss = 0
5     for point_cluster, _ in clustered_points:
6         point, cluster_index = point_cluster
7         wcss += calculateDistance(point, centroids[cluster_index][1]) ** 2
8
9     # Tính toán BCSS (Between Cluster Sum of Squares)
10    bcss = 0
11    centroid_means = np.mean([centroid[1] for centroid in centroids], axis=0)
12    for centroid in centroids:
13        bcss += len(clustered_points) * calculateDistance(centroid[1], centroid_means) ** 2
14
15    # Tính Calinski-Harabasz Index
16    k = len(centroids)
17    n = len(clustered_points)
18    calinski_harabasz = (bcss / (k - 1)) / (wcss / (n - k))
19    return calinski_harabasz

```

- Tính các chỉ số Silhouette Coefficient và Calinski – Harabasz Index



```

1 # Tạo một SparkSession
2 spark = SparkSession.builder \
3     .appName("WCSSK4") \
4     .getOrCreate()
5 sc = SparkContext.getOrCreate()
6
7 # Tải và xử lý dữ liệu
8 lines = sc.textFile(filename, minPartitions = 20)
9 # Xử lý dữ liệu đưa về một
10 points_rdd = lines.map(lambda line: [[float(string) for string in line.split(',')], 1])
11 points_rdd.cache()
12
13 k = 4 # Số lượng cụm
14 wcss, clustered_points, centroids = kmean(k, lines, points_rdd)
15 print(clustered_points[0][0])
16 print(clustered_points[1])
17 print(clustered_points[2])
18 print("Hệ số silhouette_coefficient: ", silhouette_coefficient(clustered_points, centroids))
19 print("Hệ số calinski harabasz: ", calinski_harabasz_index(clustered_points, centroids))
20

```

các trọng tâm được chọn: [[0, [2.091753060150042, 1.1855056212675756, -1.1903466366201465]], [1, [-1.0052313686969925, -0.584055635270197, -1.505772079693236]], [2, [-2.496235180316404, [-0.630217357803575, -4.187782637646889, 0.4853744699830927], 1], [[[[-0.5928806163868663, -0.3955403511377794, -0.9570191731033288], 1], 1], [[[[-0.5705047630525357, -0.0700904321571848, 1.0893449018943395], 1], 1], Hệ số silhouette_coefficient: -0.4196991319507463
Hệ số calinski harabasz: 5696.976231008187

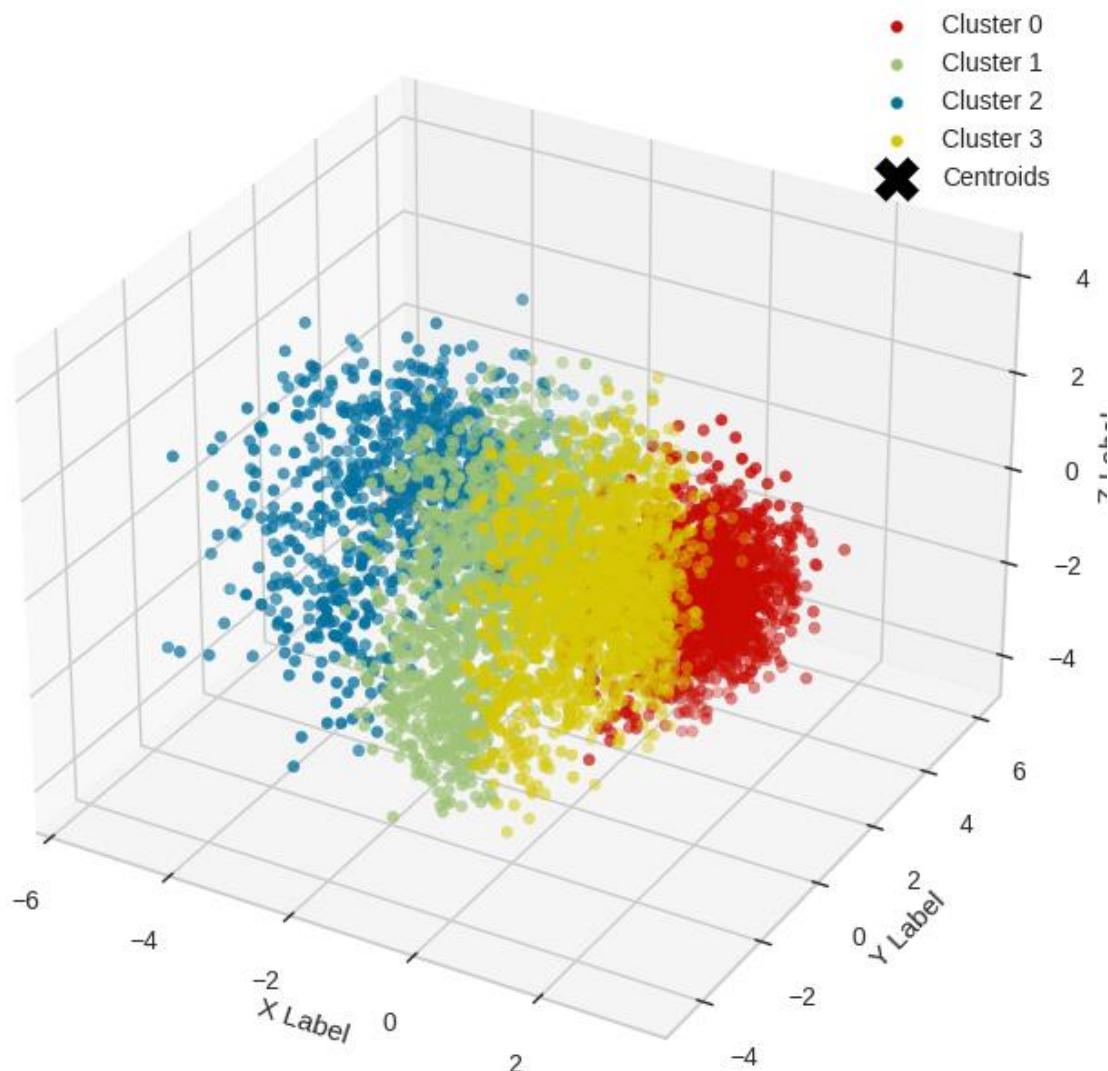
- Trực quan hóa 3D của cụm và trung tâm

```

● ● ●
1 # Vẽ biểu đồ 3D của các cụm
2 fig = plt.figure(figsize=(10, 8))
3 ax = fig.add_subplot(111, projection='3d')
4
5 colors = ['r', 'g', 'b', 'y', 'c', 'm']
6 # Vẽ các điểm trong từng cụm
7 for cluster_index in range(k):
8     cluster_points = [point[0] for point in clustered_points if point[1] == cluster_index]
9     cluster_points = np.array(cluster_points)
10    ax.scatter(cluster_points[:, 0], cluster_points[:, 1], cluster_points[:, 2], c=colors[cluster_index % len(colors)], label=f'Cluster {cluster_index}')
11
12 # Vẽ các trọng tâm
13 centroid_coords = np.array([centroid[1] for centroid in centroids])
14 ax.scatter(centroid_coords[:, 0], centroid_coords[:, 1], centroid_coords[:, 2], s=300, c='black', marker='X', label='Centroids')
15
16 ax.set_xlabel('X Label')
17 ax.set_ylabel('Y Label')
18 ax.set_zlabel('Z Label')
19 plt.legend()
20 plt.title("3D Visualization of Clusters and Centroids")
21 plt.show()

```

3D Visualization of Clusters and Centroids



Hình 4.4. Trực quan 3D phân cụm bằng thuật toán K-Means trên bộ dữ liệu để tài

4.2.2. Thuật toán Divisive Hierarchical Clustering

4.2.2.1. Tạo các hàm tính toán trong DHC

- Hàm tính phương sai:



```

1 def variance(data):
2     count = data.count()
3     # Tính giá trị trung bình
4     mean = data.reduce(lambda x, y: x + y) / count
5     # Tính phương sai
6     var = data.map(lambda x: (x - mean) ** 2).reduce(lambda x, y: x + y) / count
7     return var

```

- Hàm tách cụm dựa trên phương sai



```

1 def variance_based_split(rdd):
2     len_feature = len(rdd.first())
3     # Tính phương sai của các thuộc tính bằng hàm variance
4     variances = [variance(rdd.map(lambda x: x[i])) for i in range(len_feature)]
5     # Xác định thuộc tính với phương sai lớn nhất
6     max_variance = variances.index(max(variances))
7
8     # Sort data dựa trên thuộc tính có phương sai lớn nhất
9     sort_rdd = rdd.sortBy(lambda x: x[max_variance]).zipWithIndex().map(lambda x: (x[1], x[0]))
10    count = sort_rdd.count()
11
12    # Ktra count
13    if count == 0:
14        return rdd, spark.sparkContext.emptyRDD()
15
16    # Thực hiện tính điểm trung vị
17    median_index = count // 2
18    median_value = sort_rdd.lookup(median_index)[0][max_variance]
19
20    # Chia dữ liệu làm 2 dựa trên điểm trung vị
21    cluster1 = rdd.filter(lambda x: x[max_variance] <= median_value)
22    cluster2 = rdd.filter(lambda x: x[max_variance] > median_value)
23
24    return cluster1, cluster2

```

- Hàm phân cụm DHC



```

1  def divisive_clustering(rdd, max_clusters):
2      # Lưu dữ liệu vào một bộ dữ liệu (RDD, label của điểm)
3      clusters = [(rdd, 0)]
4      # Biến đếm cụm
5      cluster_counter = 1
6      # Vòng lặp xác định kích thước cụm có bé hơn k cụm
7      while len(clusters) < max_clusters:
8          # Tạo cụm mới
9          new_clusters = []
10         # Vòng lặp qua các cụm và nhãn trong clusters
11         for cluster, label in clusters:
12             # Thực hiện chia dữ liệu nếu kích thước dữ liệu lớn hơn 1
13             if cluster.count() > 1:
14                 # Chia dữ liệu bằng hàm variance_based_split
15                 c1, c2 = variance_based_split(cluster)
16                 # Thực hiện thêm vào cụm mới các cụm đã chia
17                 new_clusters.append((c1, label))
18                 new_clusters.append((c2, cluster_counter))
19                 # Tăng biến đếm thêm 1
20                 cluster_counter += 1
21             else:
22                 # Thực hiện thêm vào cụm mới (cluster, label)
23                 new_clusters.append((cluster, label))
24             # Gán cho biến clusters = new_clusters
25             clusters = new_clusters
26         return clusters

```

4.2.2.2. Tạo hàm chạy thuật toán DHC

- Truyền K = 4 (tìm được trên K – Means) vào thuật toán DHC



1 k = 4

2 dhc = divisive_clustering(dataRDD, k)

- Số phần tử của mỗi cụm sau khi phân cụm:

● ● ●

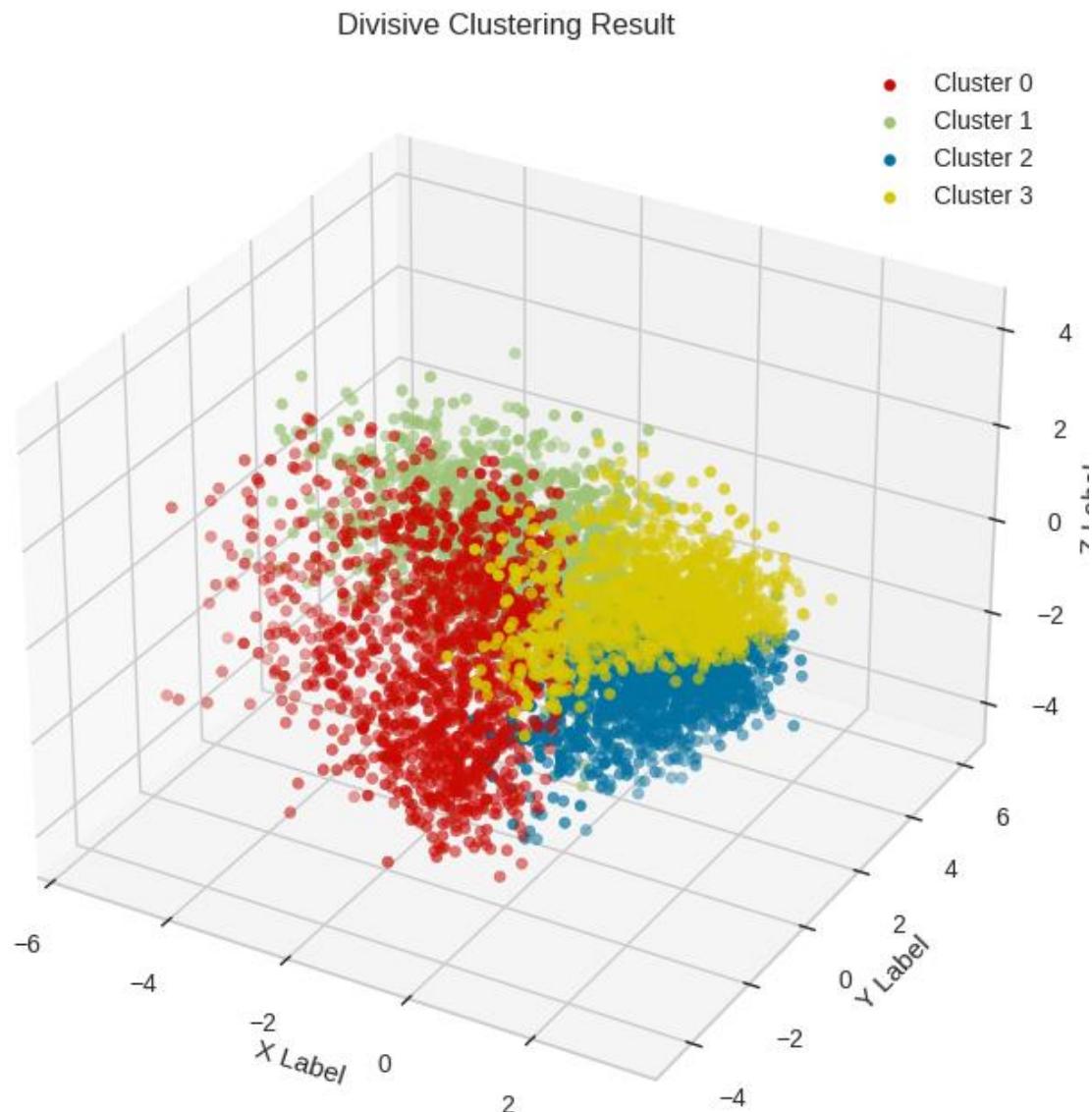
```
1 for i, cluster in enumerate(dhc):
2     cluster_rdd, label = cluster
3     print(f"Cluster {label} (label: {label}) has {cluster_rdd.count()} points")
```

```
Cluster 0 (label: 0) has 1235 points
Cluster 2 (label: 2) has 1233 points
Cluster 1 (label: 1) has 1234 points
Cluster 3 (label: 3) has 1233 points
```

- Trực quan hóa 3D của cụm và trung tâm

● ● ●

```
1 # Plot the clusters in 3D
2 fig = plt.figure(figsize=(10, 8))
3 ax = fig.add_subplot(111, projection='3d')
4
5 colors = ['r', 'g', 'b', 'y', 'c', 'm']
6 for i, cluster_tuple in enumerate(dhc):
7     cluster_rdd, label = cluster_tuple
8     cluster_data = cluster_rdd.collect()
9     cluster_data = [[row[0], row[1], row[2]] for row in cluster_data]
10    cluster_data = list(zip(*cluster_data))
11    ax.scatter(cluster_data[0], cluster_data[1], cluster_data[2], c=colors[i % len(colors)], label=f"Cluster {i}")
12
13 ax.set_xlabel('X Label')
14 ax.set_ylabel('Y Label')
15 ax.set_zlabel('Z Label')
16
17 plt.title('Divisive Clustering Result')
18 plt.legend()
19 plt.show()
```



Hình 4.5. Trục quan 3D kết quả phân cụm thuật toán DHC trên bộ dữ liệu để tài

4.3. VISUALIZE

4.3.1. Thuật toán K – Means

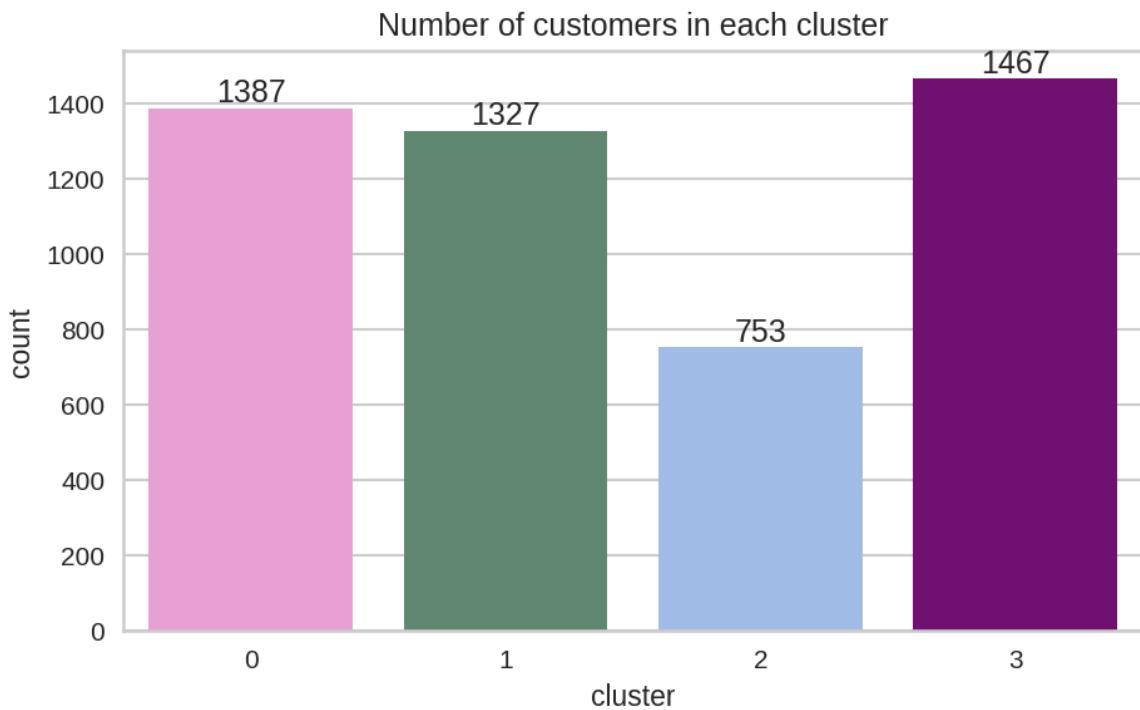
- Số lượng khách hàng trong mỗi cụm



```

1 plt.figure(figsize=(7,4),dpi=150)
2 cluster_colors = {0:"#f294d6",1:"#588c6e",2: "#94b8f2", 3: "#800080"}
3 palette = [cluster_colors[int(cluster)] for cluster in sorted(df['cluster'].unique())]
4 a = sns.countplot(data=df,x='cluster',palette=palette)
5 for i in a.containers: a.bar_label(i)
6 plt.title('Number of customers in each cluster');

```



- ⇒ Cluster 3 có số lượng khách hàng cao nhất (1476)
- Cluster 2 có số lượng thấp nhất (753)
- Hồ sơ của cụm dựa trên hạn mức tín dụng và tổng số tiền giao dịch



```

1 df["cluster"] = df["cluster"].astype(str)
2 fig = px.scatter(df.sort_values(by='cluster'), x="Credit_Limit", y="Total_Trans_Amt",
3                   color='cluster', color_discrete_sequence=cluster_colors, opacity=0.8, symbol='cluster')
4 fig.update_yaxes(title_text="Total Transaction Amount")
5 fig.update_xaxes(title_text="Credit Limit")
6 fig.update_traces(showlegend=True)
7 fig.update_layout(title="Cluster's Profile Based On Credit Limit and Total Transaction Amount")
8 fig.show()

```



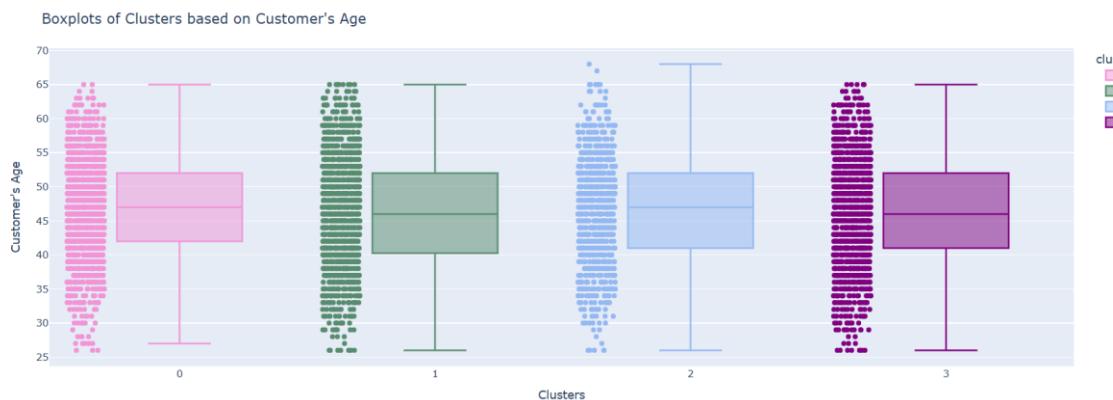
⇒ Giải thích:

- Cluster 0 (màu hồng):
 - Phân bố chủ yếu ở phía bên trái biểu đồ
 - Các khách hàng có tổng giao dịch nhiều nhất trong khoảng 4K-5K USD và hạn mức tiêu dùng từ 1438 - khoảng 5K
- Cluster 1 (màu xanh lá):
 - Tương tự với Cụm 0 nhưng phần góc dưới bên trái phân bố dày hơn
 - Thêm các khách hàng có tổng giao dịch trong khoảng từ 1K-3K USD
- Cluster 2 (màu xanh dương):
 - Phân bố tương tự với 2 cụm trên nhưng có phần đồng đều hơn
 - Hạn mức giao dịch từ 1438 - 5K phân bố nhiều và giảm dần
- Cluster 3 (màu tím):
 - Tương tự với Cụm 0, dày đặc nhất ở khu vực tổng lượng giao dịch từ 4K-5K USD và hạn mức từ 1438 - 5K

⇒ Nhận xét: Thuật toán phân cụm không được rõ ràng ở thuộc tính tổng lượng giao dịch và hạn mức của khách hàng.

- Sơ đồ các cụm dựa trên độ tuổi của khách hàng

```
● ● ●
1 fig = px.box(df.sort_values(by='cluster'), x="cluster", y="Customer_Age", color='cluster', color_discrete_sequence=cluster_colors, points='all')
2 fig.update_yaxes(title_text="Customer's Age")
3 fig.update_xaxes(title_text="Clusters")
4 fig.update_layout(title = "Boxplots of Clusters based on Customer's Age")
5 fig.show()
```



⇒ Nhận xét:

- Độ tuổi của khách hàng trong các cụm không có sự khác biệt đáng kể

- Khoảng tuổi rộng từ 26-65 tuổi, riêng cụm 2 lên đến 68 tuổi.
- Độ tuổi trung vị của các nhóm là 46 và 47 tuổi
- Sơ đồ các cụm dựa trên Tổng số giao dịch của khách hàng



```

1 fig = px.box(df.sort_values(by='cluster'), x="cluster", y="Total_Trans_Ct", color='cluster', color_discrete_sequence=cluster_colors, points='all')
2 fig.update_yaxes(title_text="Total Transaction Count")
3 fig.update_xaxes(title_text="Clusters")
4 fig.update_layout(title = "Boxplots of Clusters based on Customer's Total Transaction Count")
5 fig.show()

```



⇒ Nhận xét:

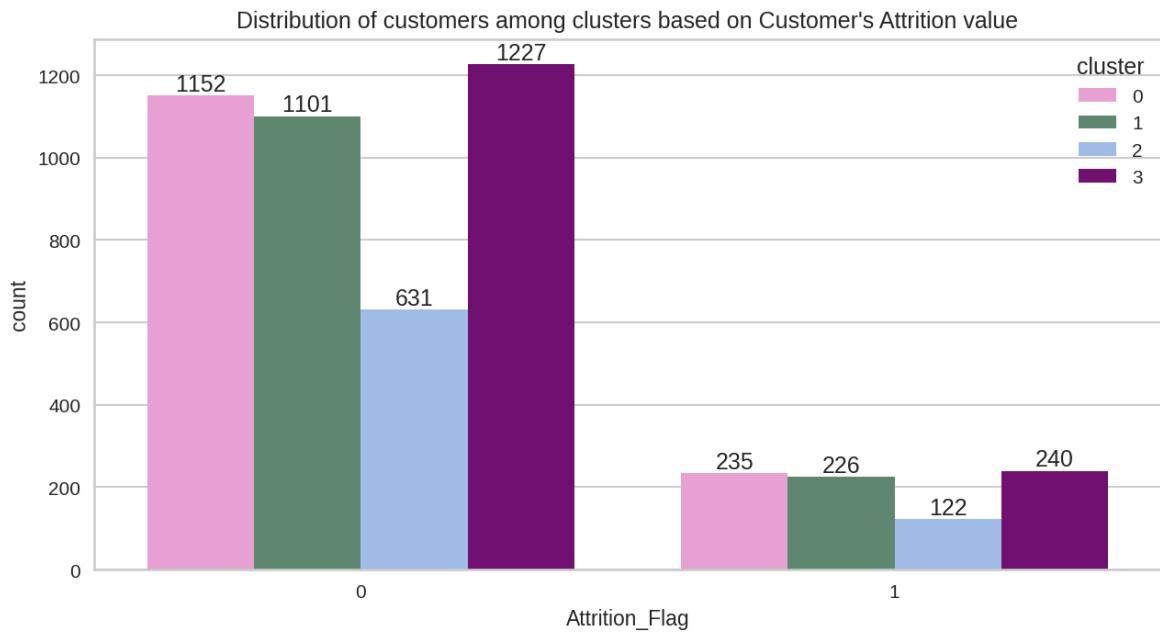
- Số lượng giao dịch của các cụm cũng có sự tương đương với nhau, với số lượng từ 10 và cao nhất là 113 giao dịch.
- Số lượng giao dịch trung vị lần lượt ở các cụm là 71, 61, 62, 66
- Phân bổ khách hàng giữa các cụm dựa trên Giá trị tiêu hao của Khách hàng



```

1 plt.figure(figsize=(10,5),dpi= 150)
2 palette = [cluster_colors[int(cluster)] for cluster in sorted(df['cluster'].unique())]
3 a = sns.countplot(data=df.sort_values(by='cluster'),x='Attrition_Flag',hue='cluster',palette=palette)
4 for i in a.containers: a.bar_label(i,)
5 plt.title ("Distribution of customers among clusters based on Customer's Attrition value");

```



⇒ Nhận xét:

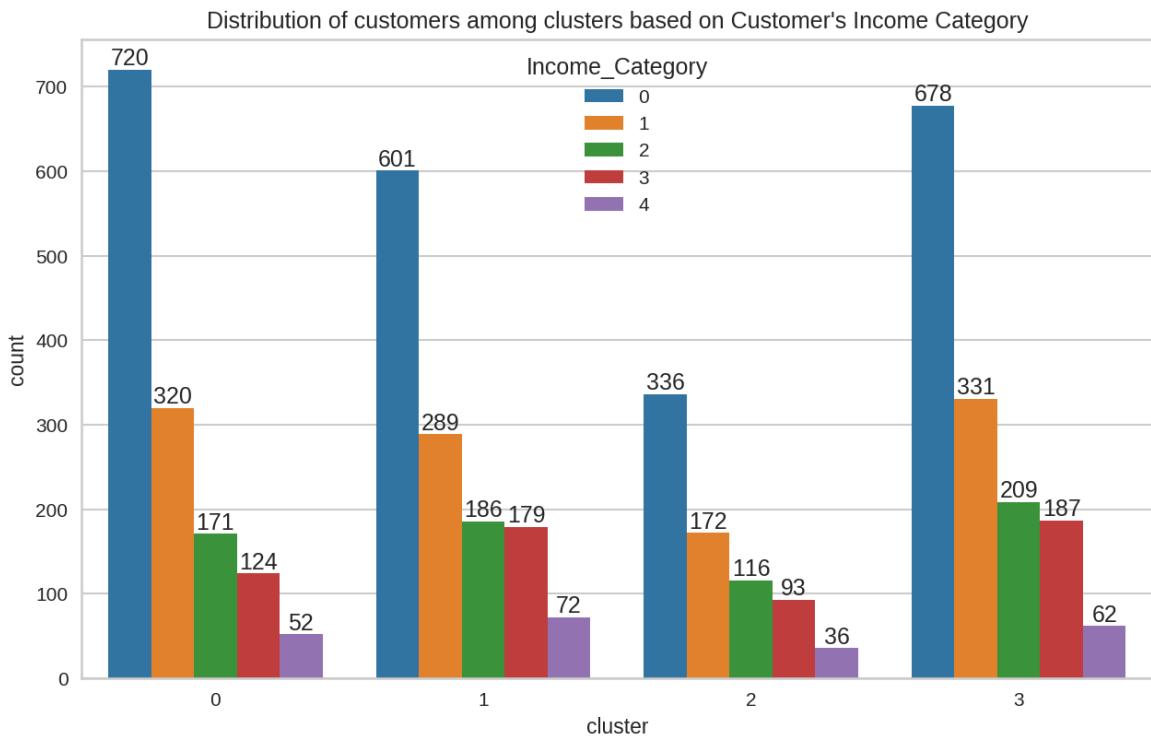
- Vì số lượng khách hàng trong mỗi cụm là tương đương nhau, trừ cụm 2. Tỉ lệ khách hàng ở lại của mỗi cũng gần tương đương nhau.
- Số lượng khách hàng còn ở lại (Attrition_Flag) của cụm 3 là cao nhất so với các cụm khác.
- Phân bổ khách hàng giữa các cụm dựa trên Khoảng thu nhập của Khách hàng



```

1 plt.figure(figsize=(10,6),dpi= 150)
2 palette = [cluster_colors[int(cluster)] for cluster in sorted(df['cluster'].unique())]
3 a = sns.countplot(data=df.sort_values(by='cluster'),x='cluster',hue='Income_Category',palette='tab10')
4 for i in a.containers: a.bar_label(i,)
5 plt.title ("Distribution of customers among clusters based on Customer's Income Category");

```



⇒ **Giải thích:** Các giá trị Income_Category:

- 0 (Màu xanh dương): Less than 40K
- 1 (Màu cam): 40K-60K
- 2 (Màu xanh lá): 60K-80K
- 3 (Màu đỏ): 80K-120K
- 4 (Màu tím): 120K+

⇒ **Nhận xét:**

- Nhóm thu nhập dưới 40K chiếm ưu thế trong tất cả các cụm, cho thấy rằng nhóm thu nhập này có đại diện cao nhất trong tập dữ liệu.
- Nhóm thu nhập 40K-60K là nhóm có đại diện nhiều thứ hai trong mỗi cụm.
- Các nhóm thu nhập 60K-80K, 80K-120K và trên 120K có số lượng khách hàng tương đối ít hơn trong tất cả các cụm, với đại diện ít nhất là nhóm thu nhập trên \$120K.
- Phân cụm không phân tách rõ ràng theo nhóm thu nhập, khách hàng từ các nhóm thu nhập thấp được phân bố đều trong tất cả các cụm.

➔ KẾT LUẬN:

➤ Cluster 0:

- Khách hàng trong nhóm này đa số là những người có thu nhập dưới 40K, số lượng khách hàng ở các mức thu nhập khác cũng cao hơn các nhóm khác.
- Tỉ lệ khách hàng rời bỏ trong nhóm này thấp

➤ Cluster 1:

- Số lượng khách hàng có thu nhập thấp hơn các nhóm.
- Tổng lượng giao dịch trung bình ở mức giữa, chứng tỏ các khách hàng này trong giai đoạn chi tiêu nhiều cho cuộc sống.

➤ Cluster 2:

- Chủ yếu là khách hàng mức thu nhập trung bình
- Tỉ lệ chưa rời bỏ là cao
- Mức chi tiêu rất đa dạng

➤ Cluster 3:

- Nhóm này có nhiều người thu nhập ở mức thấp và cũng nhiều người thu nhập mức cao so với các nhóm còn lại. Nhóm này có thu nhập cao nhất trong các cụm.
- Mức tuổi của nhóm này thấp hơn các nhóm còn lại

- Hồ sơ của cụm dựa trên tổng số lượng giao dịch và tổng số tiền giao dịch



```

1 fig = px.scatter(df, x="Total_Trans_Ct", y="Total_Trans_Amt",
2                   color='cluster', color_discrete_sequence=colors, opacity=0.8, symbol='cluster')
3 fig.update_yaxes(title_text="Total Transaction Amount")
4 fig.update_xaxes(title_text="Total Transaction Count")
5 fig.update_traces(showlegend=True)
6 fig.update_layout(title="Cluster's Profile Based On Total Transaction Count and Total Transaction Aoun")
7 fig.show()

```



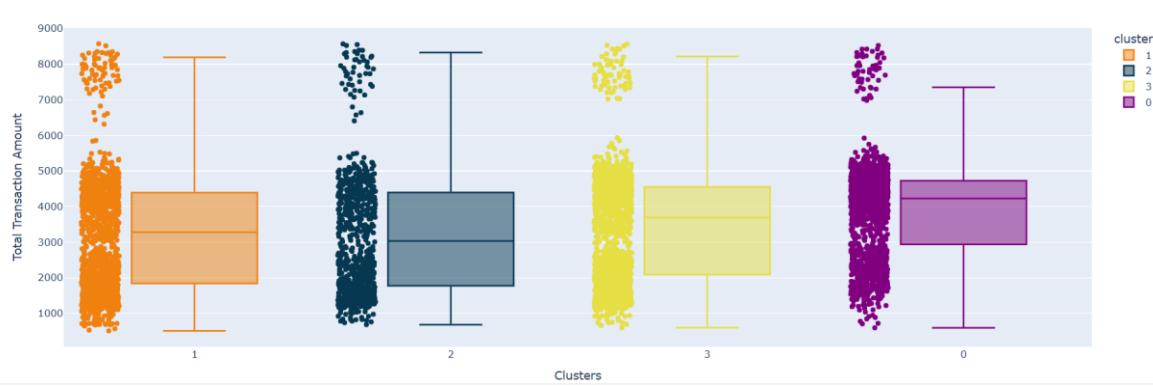
- Tổng số tiền giao dịch



```

1 fig = px.box(df, x="cluster", y="Total_Trans_Amt",
2                 color='cluster', color_discrete_sequence=colors, points='all')
3 fig.update_yaxes(title_text="Total Transaction Amount")
4 fig.update_xaxes(title_text="Clusters")
5 fig.show()

```



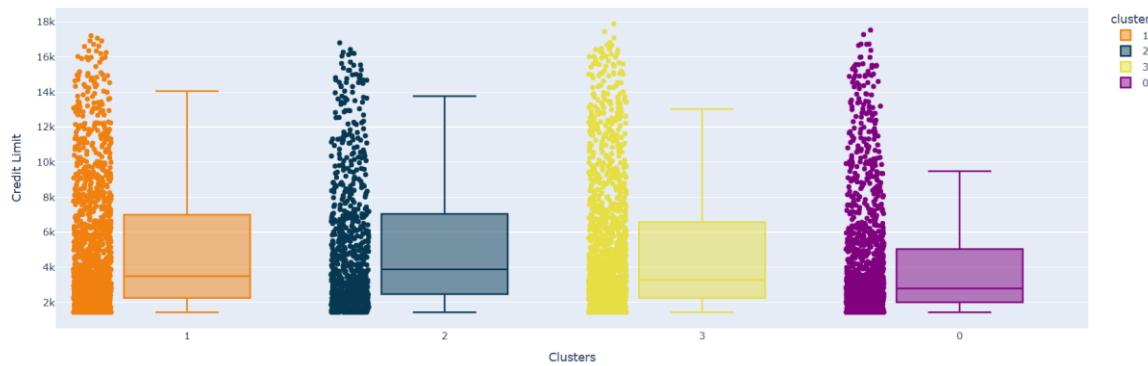
- Hạn mức tín dụng



```

1 fig = px.box(df, x="cluster", y="Credit_Limit", color='cluster', color_discrete_sequence=colors, points='all')
2 fig.update_yaxes(title_text="Credit Limit")
3 fig.update_xaxes(title_text="Clusters")
4 fig.show()

```



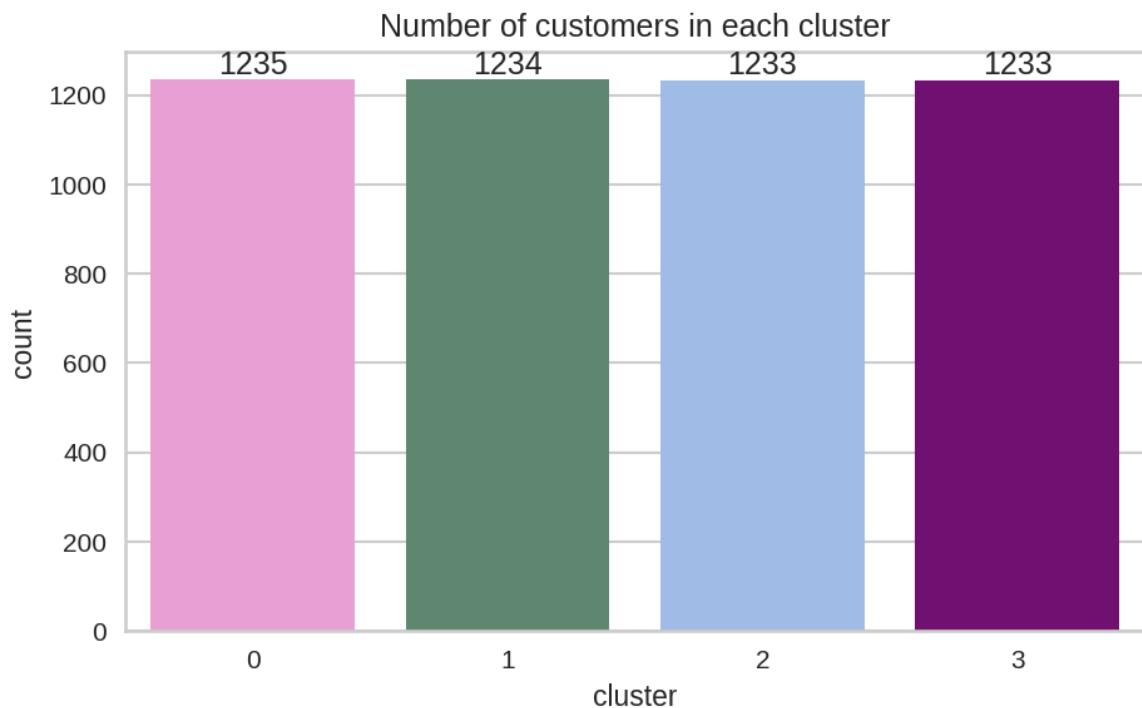
4.3.2. Thuật toán Divisive Hierarchical Clustering

- Số lượng khách hàng trong mỗi cụm



```

1 plt.figure(figsize=(7,4),dpi=150)
2 cluster_colors = {0:"#f294d6",1:"#588c6e",2: "#94b8f2", 3: "#800080"}
3 palette = [cluster_colors[int(i)] for i in sorted(visualDHC['cluster'].unique())]
4 a = sns.countplot(data=visualDHC,x='cluster',palette=palette)
5 for i in a.containers: a.bar_label(i)
6 plt.title('Number of customers in each cluster');
    
```



⇒ Nhận xét: Các Cluster được phân chia với số lượng đồng đều nhau, chỉ chênh lệch 1-2 về số lượng

- Hồ sơ của cụm dựa trên hạn mức tín dụng và tổng số tiền giao dịch



```

1 visualDHC["cluster"] = visualDHC["cluster"].astype(str)
2 fig = px.scatter(visualDHC.sort_values(by='cluster'), x="Credit_Limit", y="Total_Trans_Amt",
3                   color='cluster', color_discrete_sequence=cluster_colors, opacity=0.8, symbol='cluster')
4 fig.update_yaxes(title_text="Total Transaction Amount")
5 fig.update_xaxes(title_text="Credit Limit")
6 fig.update_traces(showlegend=True)
7 fig.update_layout(title="Cluster's Profile Based On Credit Limit and Total Transaction Amount")
8 fig.show()

```



⇒ Nhận xét:

- Cluster 0 (màu hồng):
 - Phân bố chủ yếu ở góc dưới bên trái và dài dưới của biểu đồ
 - Khách hàng chủ yếu giá trị các giao dịch thấp trong khoảng 1K-2.5K, hạn mức tín dụng của khách hàng trong khoảng từ 1.5K đến 4K và rải rác giảm dần khi tăng lên
- Cluster 1 (màu xanh lá):
 - Phân bố ở khu vực giữa biểu đồ phía bên trái
 - Các khách hàng có khối lượng giao dịch ở mức trung bình khoảng 4K-5K với hạn mức tín dụng thẻ chủ yếu ở từ 1.5K đến 4K
- Cluster 2 (màu xanh dương):
 - Nằm ở dài dưới của biểu đồ nhưng phân bố thưa hơn so với các cluster khác.
 - Là các khách hàng có tổng lượng giao dịch từ 1K-5K, hạn mức chủ yếu từ 1438 USD trở lên

- Cluster 3 (màu tím)

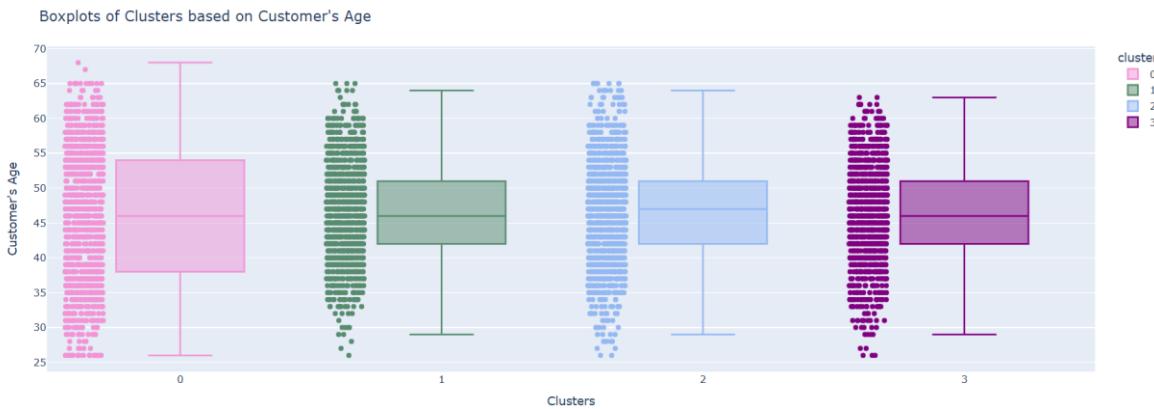
- Tập trung nhiều ở giữa biểu đồ góc bên trái, và 1 dài ở phía trên.
- Là các khách hàng có khối lượng giao dịch từ 4K-5K, một số nhỏ từ 7K-9K, hạn mức chủ yếu của các khách hàng từ 1.4K-dưới 4K USD
- Sơ đồ các cụm dựa trên độ tuổi của khách hàng



```

1 fig = px.box(visualDHC.sort_values(by='cluster'), x="cluster", y="Customer_Age", color='cluster', color_discrete_sequence=cluster_colors, points='all')
2 fig.update_xaxes(title_text="Customer's Age")
3 fig.update_xaxes(title_text="Clusters")
4 fig.update_layout(title = "Boxplots of Clusters based on Customer's Age")
5 fig.show()

```



⇒ Nhận xét:

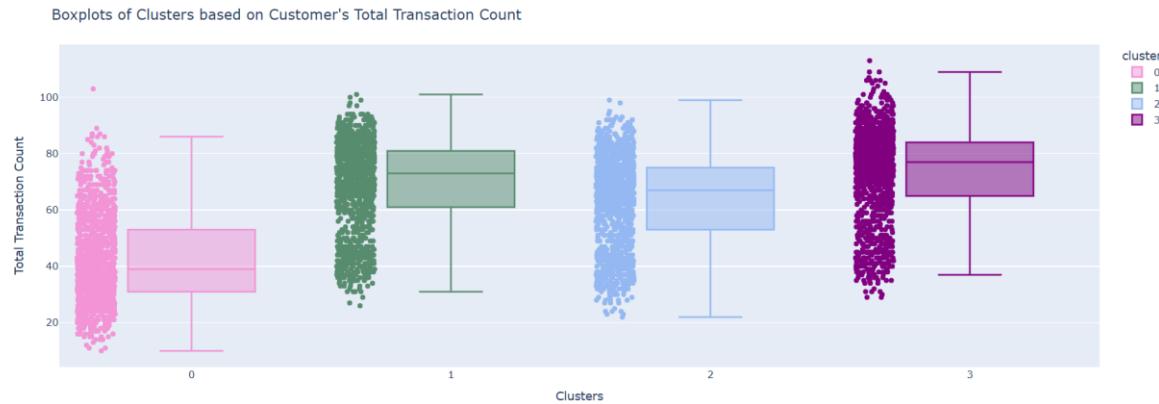
- Cluster 0 (màu hồng): Mức trung vị tuổi là 46 và phạm vi tuổi khá rộng (26-68) nhóm khách hàng chủ yếu là người trung niên
- Cluster 1 (màu xanh lá): Mức trung vị tuổi bằng với Cụm 0 và khoảng tuổi nhỏ hơn cho thấy độ biến động về tuổi của khách hàng ít hơn, tức là khách hàng trong cụm này có độ tuổi khá đồng đều, và trẻ hơn so với Cụm 0.
- Cluster 2 (màu xanh dương): Cụm này có biểu đồ gần tương đương với Cụm 1 nhưng độ tuổi trung vị là 47 tuổi
- Cluster 3 (màu tím): Cũng tương tự với 2 cụm trên, độ tuổi trung vị của là 46, khoảng tuổi cũng tương đương so với các cụm khác

⇒ Kết luận: Thuật toán đã đánh giá phân cụm độ tuổi chưa thực sự rõ, tuổi trong các cụm vẫn tương đương nhau, riêng cụm 0 có sự khác biệt hơn so với các cụm còn lại

- Sơ đồ các cụm dựa trên Tổng số giao dịch của khách hàng

● ● ●

```
1 fig = px.box(visualDHC.sort_values(by='cluster'), x="cluster", y="Total_Trans_Ct", color='cluster', color_discrete_sequence=cluster_colors, points='all')
2 fig.update_yaxes(title_text="Total Transaction Count")
3 fig.update_xaxes(title_text="Clusters")
4 fig.update_layout(title = "Boxplots of Clusters based on Customer's Total Transaction Count")
5 fig.show()
```



⇒ Nhận xét:

- Cluster 0 (màu hồng):
 - Số lượng giao dịch đa số nằm trong khoảng (10-86) và trung vị là 39 cho thấy cụm này có số lượng giao dịch thấp
 - Cụm này không có nhiều giá trị ngoại lai, riêng nhận thấy có khách hàng có số lượng giao dịch cao vượt trội so với các khách hàng khác (hơn 100)
- Cluster 1 (màu xanh lá): Số lượng giao dịch trong khoảng 31-101, giá trị trung vị là 73. Đây là cụm có số lượng giao dịch cao nhìn trong bộ dữ liệu
- Cluster 2 (màu xanh dương): Số lượng giao dịch cụm này khá cao (22-99) và rất ít giá trị ngoại lai
- Cluster 3 (màu tím): Đây là cụm có số lượng giao dịch cao nhất trong 4 cụm với giá trị cận trên lên đến 113. Nhưng cụm này vẫn có nhiều giá trị ngoại lai phân bố dày.

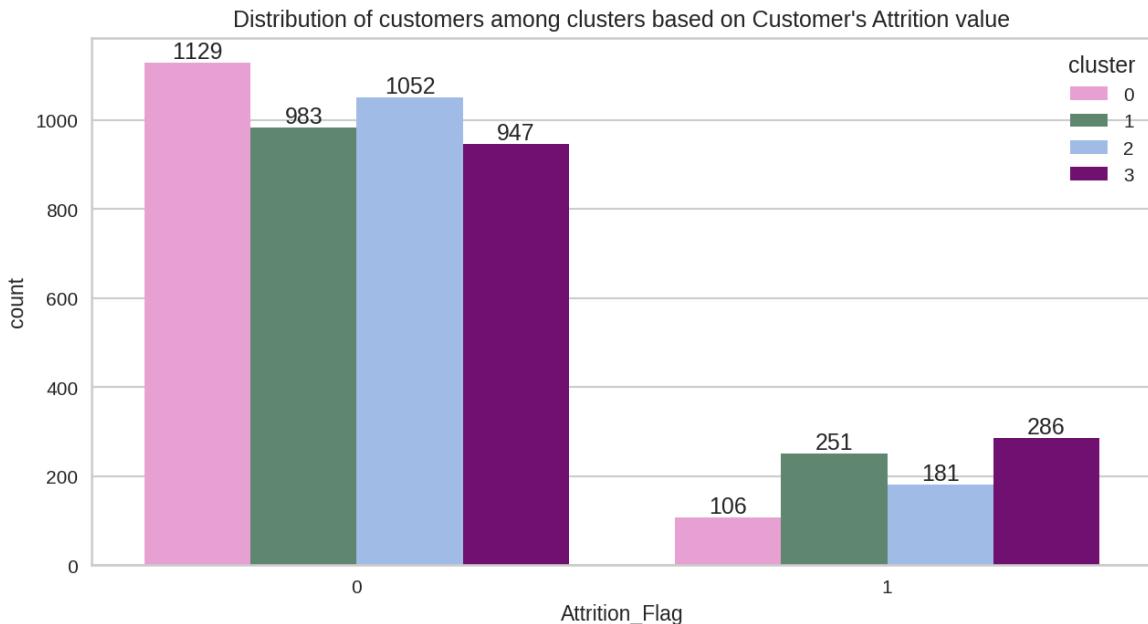
- Phân bô khách hàng giữa các cụm dựa trên Giá trị tiêu hao của Khách hàng



```

1 plt.figure(figsize=(10,5),dpi= 150)
2 palette = [cluster_colors[int(cluster)] for cluster in sorted(visualDHC['cluster'].unique())]
3 a = sns.countplot(data=visualDHC.sort_values(by='cluster'),x='Attrition_Flag',hue='cluster',palette=palette)
4 for i in a.containers: a.bar_label(i,)
5 plt.title ("Distribution of customers among clusters based on Customer's Attrition value");

```



⇒ Kết luận: Số lượng khách hàng trong 4 cụm là tương đương nhau, theo biểu đồ cho thấy số lượng khách hàng đã rời bỏ của cụm 0 là thấp nhất (`Attrition_Flag = 1`), và cụm 4 là cao nhất.

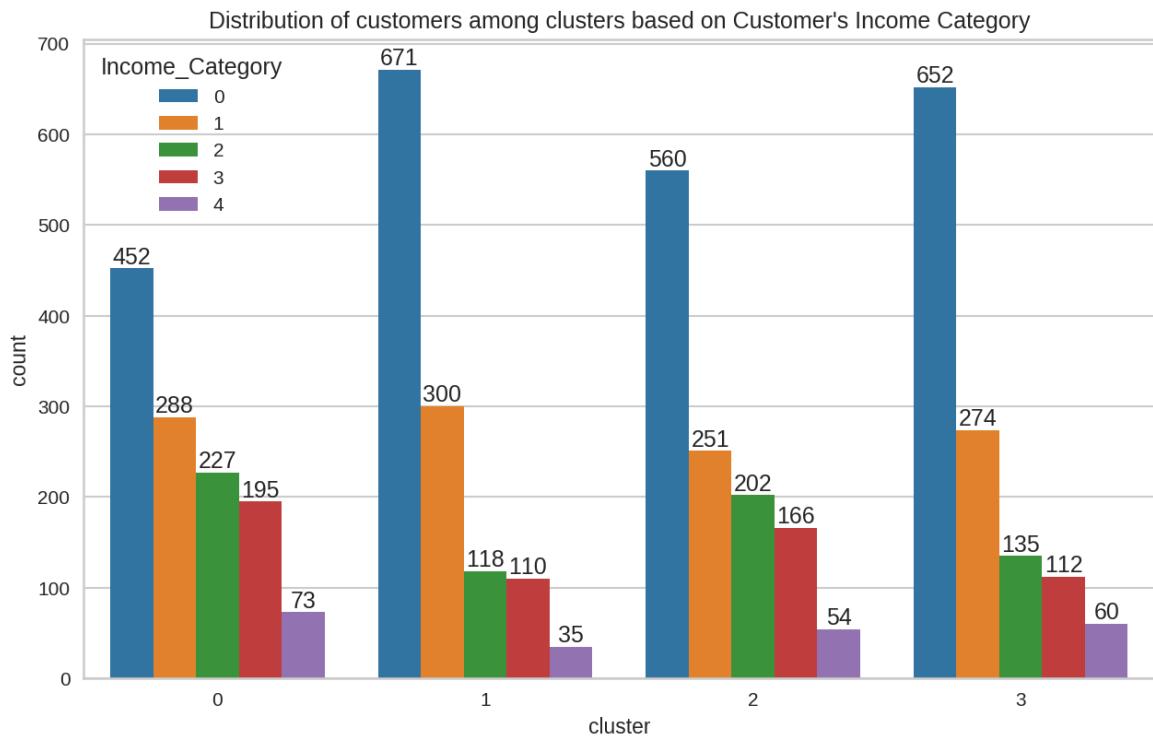
- Phân bô khách hàng giữa các cụm dựa trên Danh mục thu nhập của Khách hàng



```

1 plt.figure(figsize=(10,6),dpi= 150)
2 palette = [cluster_colors[int(cluster)] for cluster in sorted(visualDHC['cluster'].unique())]
3 a = sns.countplot(data=visualDHC.sort_values(by='cluster'),x='cluster',hue='Income_Category',palette='tab10')
4 for i in a.containers: a.bar_label(i,)
5 plt.title ("Distribution of customers among clusters based on Customer's Income Category");

```



⇒ Nhận xét:

- Nhóm thu nhập dưới 40K (Màu xanh dương): Chiếm ưu thế trong tất cả các cụm, cho thấy rằng nhóm thu nhập này có đại diện cao nhất trong tập dữ liệu.
- Nhóm thu nhập 40K-60K (Màu cam): Là nhóm có đại diện nhiều thứ hai trong mỗi cụm.
- Nhóm thu nhập 60K-80K (Màu xanh lá), 80K-120K (Màu đỏ), và trên 120K (Màu tím): Có số lượng khách hàng tương đối ít hơn trong tất cả các cụm, với nhóm thu nhập trên 120K có đại diện ít nhất.
- Phân cụm: Không phân tách rõ ràng theo nhóm thu nhập, khách hàng từ các nhóm thu nhập thấp được phân bố đều trong tất cả các cụm.

➔ KẾT LUẬN:**➤ Cluster 0:**

- Khách hàng trong nhóm này đa số là những người có thu nhập dưới 40K, số lượng khách hàng ở các mức thu nhập khác cũng cao hơn các nhóm khác.
- Số lượng khách hàng rời bỏ trong nhóm này thấp
- Số lượng giao dịch cũng thấp nhất trong 4 cụm, hạn mức giao dịch ở mức phổ thông và ổn định. Tổng lượng giao dịch của khách hàng trong nhóm còn thấp mặc dù hạn mức tín dụng có những khách cao.
- Độ tuổi của nhóm này đa dạng tuổi hơn các nhóm còn lại

➤ Cluster 1:

- Số lượng khách hàng có thu nhập thấp hơn các nhóm.
- Tổng lượng giao dịch trung bình ở mức giữa, chứng tỏ các khách hàng này trong giai đoạn chi tiêu nhiều cho cuộc sống.
- Độ tuổi của nhóm này ít đa dạng, phần lớn từ 42-51

➤ Cluster 2:

- Chủ yếu là khách hàng mức thu nhập trung bình
- Tỉ lệ chưa rời bỏ là cao
- Độ tuổi chính của nhóm này cao hơn các nhóm khác
- Mức chi tiêu rất đa dạng

➤ Cluster 3:

- Nhóm này có nhiều người thu nhập ở mức thấp và cũng nhiều người thu nhập mức cao so với các nhóm còn lại. Nhóm này có thu nhập cao nhất.
- Mức chi tiêu của khách hàng trong nhóm này cao, có thể lên tới trên 8K
- Nhưng số khách hàng đã rời bỏ của nhóm này cũng là cao nhất

- Hồ sơ của cụm dựa trên tổng số lượng giao dịch và tổng số tiền giao dịch



```

1 fig = px.scatter(visualDHC, x="Total_Trans_Ct", y="Total_Trans_Amt",
2                   color='cluster', color_discrete_sequence=colors, opacity=0.8, symbol='cluster')
3 fig.update_yaxes(title_text="Total Transaction Amount")
4 fig.update_xaxes(title_text="Total Transaction Count")
5 fig.update_traces(showlegend=True)
6 fig.update_layout(title="Cluster's Profile Based On Total Transaction Count and Total Transaction Amoun")
7 fig.show()

```



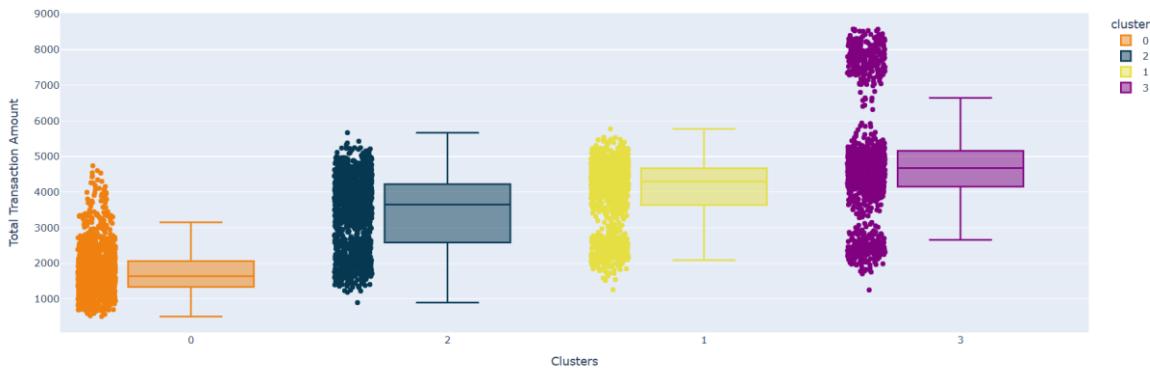
- Tổng số tiền giao dịch



```

1 fig = px.box(visualDHC, x="cluster", y="Total_Trans_Amt",
2               color='cluster', color_discrete_sequence=colors, points='all')
3 fig.update_yaxes(title_text="Total Transaction Amount")
4 fig.update_xaxes(title_text="Clusters")
5 fig.show()

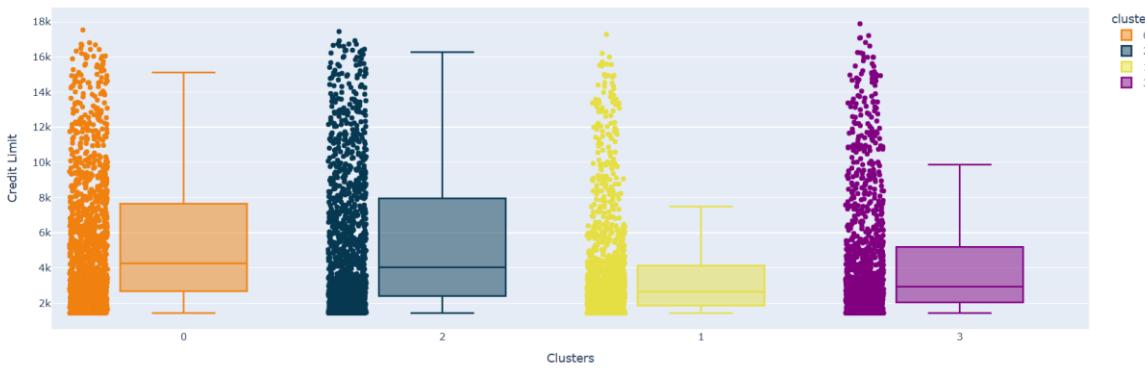
```



- Hạn mức tín dụng

● ● ●

```
1 fig = px.box(visualDHC, x="cluster", y="Credit_Limit", color='cluster', color_discrete_sequence=colors, points='all')
2 fig.update_yaxes(title_text="Credit Limit")
3 fig.update_xaxes(title_text="Clusters")
4 fig.show()
```



4.4. Bảng so sánh

Bảng 3. Bảng so sánh kết quả của 2 thuật toán phân cụm

	K - Mean	DHC
Silhouette Coefficient	-0.4196991319507463	0.2251481035332149
Calinski harabasz index	5696.976231008187	1821.4155161315641

Phần 5. KẾT LUẬN

5.1. Kết quả đạt được

- Nhận diện bài toán khai thác dữ liệu lớn: Trong bối cảnh dữ liệu ngày càng phát triển với quy mô lớn, việc khai thác và phân tích dữ liệu trở thành một thách thức quan trọng. Đề tài này tập trung vào việc áp dụng các giải thuật khai thác dữ liệu song song và phân tán, cho phép xử lý một lượng lớn thông tin một cách hiệu quả. Bằng cách này, có thể rút ra những thông tin giá trị từ dữ liệu khách hàng, hỗ trợ các quyết định chiến lược trong kinh doanh.
- Ứng dụng công nghệ học máy: Đề tài sử dụng các thuật toán phân cụm, nổi bật là K-means và Phân cụm Phân chia, để phân tích và nhóm các khách hàng dựa trên các đặc điểm cụ thể. K-means là một phương pháp phổ biến giúp phân loại điểm dữ liệu thành các cụm dựa trên sự tương đồng, trong khi Phân cụm Phân chia cung cấp một cái nhìn sâu sắc hơn về mối quan hệ giữa các nhóm. Những thuật toán này không chỉ giúp phát hiện các khuynh hướng trong hành vi mua sắm mà còn tạo ra các nhóm khách hàng tương đồng, từ đó hỗ trợ doanh nghiệp trong việc phát triển các chiến lược tiếp cận phù hợp.
- Tính ứng dụng cao trong thực tiễn kinh doanh: Kết quả nghiên cứu từ đề tài này có thể áp dụng trực tiếp vào các chiến lược kinh doanh cụ thể.
- Như vậy, đề tài không chỉ mang lại giá trị lý thuyết mà còn có tính ứng dụng cao, giúp doanh nghiệp có những quyết định chiến lược dựa trên dữ liệu, từ đó nâng cao khả năng cạnh tranh trên thị trường.

5.2. Hạn chế

- Bộ dữ liệu được thu thập ngoài lãnh thổ Việt Nam, do đó vẫn còn nhiều khía cạnh chưa phản ánh đầy đủ đặc điểm của người Việt Nam nếu muốn áp dụng vào thực tiễn tại Việt Nam.
- Nhóm sử dụng hai thuật toán phân cụm là K-Means và DHC để phân tích dữ liệu, nhưng cả hai phương pháp này đều gặp phải một số hạn chế liên quan đến việc tối ưu hóa.

- **K-Means:** Thuật toán này yêu cầu xác định số lượng cụm trước khi thực hiện, điều này có thể gây khó khăn trong quá trình áp dụng và dẫn đến những quyết định không chính xác về số lượng cụm tối ưu.
- **Divisive Hierarchical Clustering:** Phương pháp này phụ thuộc vào số lượng cụm sẽ được xác định, và quá trình chia dữ liệu dựa trên phương sai có thể gặp khó khăn trong việc xử lý các giá trị ngoại lai và các biến không mong muốn.
- Những hạn chế này nhấn mạnh rằng cần có thêm nghiên cứu và điều chỉnh để cải thiện tính hiệu quả của các thuật toán phân cụm, đặc biệt là khi áp dụng cho bối cảnh và đặc điểm văn hóa của người tiêu dùng Việt Nam.

5.3. Hướng phát triển

- Mở rộng bộ dữ liệu:
 - Thu thập dữ liệu địa phương: Thu thập dữ liệu thị trường Việt Nam để các phân tích phản ánh đúng đặc điểm và hành vi người tiêu dùng địa phương.
 - Kết hợp dữ liệu đa nguồn: Tích hợp dữ liệu từ nhiều nguồn khác nhau để có cái nhìn toàn diện hơn về hành vi khách hàng.
- Tối ưu hóa thuật toán:
 - Thử nghiệm với các phương pháp phân cụm khác: Khám phá các thuật toán phân cụm khác như DBSCAN hoặc Gaussian Mixture Models, có thể xử lý tốt hơn các giá trị ngoại lai và không yêu cầu xác định số cụm trước.
 - Sử dụng phương pháp học máy tự động: Áp dụng các kỹ thuật học máy tự động (AutoML) để tự động hóa quá trình lựa chọn và tối ưu hóa mô hình.
- Phát triển mô hình phân tích nâng cao:
 - Sử dụng phân tích đa biến: Kết hợp phân tích đa biến để hiểu rõ hơn về các mối quan hệ giữa các đặc điểm của khách hàng và hành vi mua sắm.
- Xây dựng mô hình dự đoán: Phát triển các mô hình dự đoán hành vi mua sắm dựa trên các phân khúc khách hàng đã xác định
- Tích hợp công nghệ mới:
 - Áp dụng trí tuệ nhân tạo và học sâu: Khám phá các công nghệ học sâu để phân tích dữ liệu phức tạp và cải thiện độ chính xác trong việc phân cụm.

TÀI LIỆU THAM KHẢO

- [1] *Data preprocessing: Definition & key steps* (2024). (n.d.). Visier.com. Retrieved December 15, 2024, from <https://www.visier.com/blog/data-preprocessing/>
- [2] Gorthy, S. (2024, April 1). *Euclidean distance explained*. Built In. <https://builtin.com/articles/euclidean-distance>
- [3] Huy, T. N. (2022, October 12). *ML From Scratch: Thuật toán giảm chiều dữ liệu PCA*. Viblo. <https://viblo.asia/p/ml-from-scratch-thuat-toan-giam-chieu-du-lieu-pca-7ymJXKMa4kq>
- [4] Nevil, S. (2003, November 18). *Z-Score: Meaning and Formula*. Investopedia. <https://www.investopedia.com/terms/z/zscore.asp>
- [5] Saji, B. (2021, January 20). *Elbow method for optimal cluster number in K-means*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/>
- [6] Sharma, N. (2022, July 22). *K-Means clustering explained*. Neptune.Ai. <https://neptune.ai/blog/k-means-clustering>
- [7] Vu, T. (2017, January 1). *Bài 4: K-means Clustering*. Tiep Vu's Blog. <https://machinelearningcoban.com/2017/01/01/kmeans/>
- [8] *What are the benefits and drawbacks of using z-scores to standardize your data for predictive modeling?* (2023, March 21). Linkedin.com; www.linkedin.com. <https://www.linkedin.com/advice/1/what-benefits-drawbacks-using-z-scores-standardize>
- [9] (N.d.-a). Machinelearningplus.com. Retrieved December 15, 2024, from <https://www.machinelearningplus.com/pyspark/pyspark-stringindexer/>
- [10] (N.d.-b). Sciencedirect.com. Retrieved December 15, 2024, from <https://www.sciencedirect.com/topics/computer-science/silhouette-coefficient>

PHÂN CÔNG CÔNG VIỆC

Công việc	Trần Thị Kim Anh (Nhóm trưởng)	Hồ Quang Lâm (Thành viên)	Lê Thị Lệ Trúc (Thành viên)	Lê Minh Chánh (Thành viên)
Giới thiệu đề tài			X	
Lý thuyết Phương pháp và thuật toán sử dụng			X	
Tìm hiểu data	X			
Tiền xử lý dữ liệu	X	X	X	X
Đặc tả thuật toán K-Means	X			
Đặc tả thuật toán DHC				X
Thực hiện Thuật toán	X	X	X	X
Visualize Data	X	X	X	X
Làm word báo cáo	X	X	X	X
Làm slide	X	X	X	X
Tổng kết	X		X	
Làm báo cáo word	X	X	X	X

Làm powerpoint	X			X
Thuyết trình	X	X	X	X
Mức độ hoàn thành	100%	100%	100%	100%