ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA HỆ THỐNG THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

DỮ LIỆU LỚN – IS405.P11.HTCL

PHÂN CỤM KHÁCH HÀNG VỚI KMEANS

GVHD: ThS. Nguyễn Hồ Duy Tri

Trương Vĩnh Thuận	- 21522653
Hoàng Quốc Việt	- 21522790
Phùng Thiên Phúc	- 21521297

HO CHI MINH CITY, 2024

Mục lục

Danh mục hình ảnh

CHƯƠNG 1. LÝ DO CHỌN ĐỀ TÀI

Phân cụm khách hàng là một trong những phương pháp phân tích dữ liệu phổ biến, giúp các doanh nghiệp khai thác tối đa giá trị từ dữ liệu khách hàng. Bằng cách nhóm các khách hàng có đặc điểm hoặc hành vi tương tự lại với nhau, doanh nghiệp có thể áp dụng các chiến lược quản lý và phát triển hiệu quả hơn.

Một số ứng dụng phổ biến của việc phân cụm khách hàng:

- **Chiến lược tiếp thị cá nhân hóa:** Tạo các chiến dịch quảng cáo và tiếp thị phù hợp với từng nhóm khách hàng.
- **Dự đoán giá trị lâu dài của khách hàng (CLV):** Phân nhóm khách hàng để dự đoán và tối ưu hóa chiến lược giữ chân khách hàng.
- **Phân tích nhu cầu khách hàng:** Nhận biết các nhu cầu ẩn trong dữ liệu khách hàng.
- **Định giá sản phẩm và dịch vụ:** Phân khúc thị trường dựa trên thu nhập hoặc hành vi tiêu dùng.

CHƯƠNG 2. DỮ LIỆU

Tên dataset: Customer Segmentation : Clustering

Nguồn dữ liệu: https://www.kaggle.com/datasets/vishakhdapat/customer-segmentation-clustering

Mô tả dữ liệu: Dữ liệu cung cấp thông tin chi tiết về hồ sơ khách hàng, cho phép doanh nghiệp phân tích đặc điểm và hành vi của khách hàng. Dữ liệu bao gồm thông tin nhân khẩu học (như năm sinh, trình độ học vấn, tình trạng hôn nhân), cấu trúc hộ gia đình (số lượng trẻ nhỏ, thanh thiếu niên), hành vi mua sắm (chi tiêu theo từng danh mục sản phẩm, kênh mua hàng) và tương tác của khách hàng (phản hồi chiến dịch tiếp thị, khiếu nại, số lượt truy cập website).

ID	Year_Birth	Education	Marital_Sta	Income	Kidhome	Teenhome	Dt_Custom	Recency	MntWines	MntFruits	MntMeatPr	MntFishPr	MntSweetP	MntGoldPr	NumDeals	NumWebP	NumCatal	NumStoreI	NumWebV	AcceptedC	AcceptedC	AcceptedC
5524	1957	Graduatior	Single	58138	0	0	4/9/2012	58	635	88	546	172	88	88	3	8	10	4	7	0	0	0
2174	1954	Graduatior	Single	46344	1	1	8/3/2014	38	11	1	6	2	1	6	2	1	1	2	5	0	0	0
4141	1965	Graduatior	Together	71613	0	0	21-08-201	26	426	49	127	111	21	42	1	8	2	10	4	0	0	0
6182	1984	Graduatior	Together	26646	1	0	########	26	11	4	20	10	3	5	2	2	0	4	6	0	0	0
5324	1981	PhD	Married	58293	1	0	19-01-201	94	173	43	118	46	27	15	5	5	3	6	5	0	0	0
7446	1967	Master	Together	62513	0	1	9/9/2013	16	520	42	98	0	42	14	2	6	4	10	6	0	0	0
965	1971	Graduatior	Divorced	55635	0	1	13-11-201	34	235	65	164	50	49	27	4	7	3	7	6	0	0	0
6177	1985	PhD	Married	33454	1	0	8/5/2013	32	76	10	56	3	1	23	2	4	0	4	8	0	0	0
4855	1974	PhD	Together	30351	1	0	6/6/2013	19	14	0	24	3	3	2	1	3	0	2	9	0	0	0
5899	1950	PhD	Together	5648	1	1	13-03-201	68	28	0	6	1	1	13	1	1	0	0	20	1	0	0
1994	1983	Graduatior	Married		1	0	15-11-201	11	5	5	6	0	2	1	1	1	0	2	7	0	0	0
387	1976	Basic	Married	7500	0	0	13-11-201	59	6	16	11	11	1	16	1	2	0	3	8	0	0	0
2125	1959	Graduatior	Divorced	63033	0	0	15-11-201	82	194	61	480	225	112	30	1	3	4	8	2	0	0	0
8180	1952	Master	Divorced	59354	1	1	15-11-201	53	233	2	53	3	5	14	3	6	1	5	6	0	0	0
2569	1987	Graduatior	Married	17323	0	0	########	38	3	14	17	6	1	5	1	1	0	3	8	0	0	0
2114	1946	PhD	Single	82800	0	0	24-11-201	23	1006	22	115	59	68	45	1	7	6	12	3	0	0	1
9736	1980	Graduatior	Married	41850	1	1	24-12-201	51	53	5	19	2	13	4	3	3	0	3	8	0	0	0
4939	1946	Graduatior	Together	37760	0	0	31-08-201	20	84	5	38	150	12	28	2	4	1	6	7	0	0	0
6565	1949	Master	Married	76995	0	1	28-03-201	91	1012	80	498	0	16	176	2	11	4	9	5	0	0	0
2278	1985	2n Cycle	Single	33812	1	0	########	86	4	17	19	30	24	39	2	2	1	3	6	0	0	0
9360	1982	Graduatior	Married	37040	0	0	8/8/2012	41	86	2	73	69	38	48	1	4	2	5	8	0	0	0
5376	1979	Graduatior	Married	2447	1	0	6/1/2013	42	1	1	1725	1	1	1	15	0	28	0	1	0	0	0
1993	1949	PhD	Married	58607	0	1	23-12-201	63	867	0	86	0	0	19	3	2	3	9	8	0	1	0
4047	1954	PhD	Married	65324	0	1	########	0	384	0	102	21	32	5	3	6	2	9	4	0	0	0
1409	1951	Graduatior	Together	40689	0	1	18-03-201	69	270	3	27	39	6	99	7	7	1	5	8	0	0	0

Hình 1 Dữ liệu ban đầu

Chi tiết dữ liệu: Dữ liệu gồm có 2240 dòng và 29 cột

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa
1	Id	Số nguyên	Mã định danh cho khách hàng
2	Year_Birth	Số nguyên	Năm sinh của khách hàng
3	Education	Chuỗi ký tự	Trình độ học vấn cao nhất mà khách hàng đạt được
4	Marital_Status	Chuỗi ký tự	Tình trạng hôn nhân của khách hàng
5	Income	Số nguyên	Thu nhập hàng năm của khách hàng
6	Kidhome	Số nguyên	Số trẻ nhỏ trong hộ gia đình của khách hàng

7	Teenhome	Số nguyên	Số thanh thiếu niên trong hộ gia đình của khách hàng
8	Dt_Customer	Date	Ngày khách hàng bắt đầu tham gia hoặc được ghi nhận vào cơ sở dữ liệu của công ty
9	Recency	Số nguyên	Số ngày kể từ lần mua hàng hoặc tương tác gần nhất của khách hàng
10	MntWines	Số nguyên	Tổng số tiền khách hàng đã chi tiêu cho rượu vang
11	MntFruits	Số nguyên	Tổng số tiền khách hàng đã chi tiêu cho trái cây
12	MntMeatProducts	Số nguyên	Tổng số tiền khách hàng đã chi tiêu cho các sản phẩm từ thịt
13	MntFishProducts	Số nguyên	Tổng số tiền khách hàng đã chi tiêu cho các sản phẩm từ cá
14	MntSweetProducts	Số nguyên	Tổng số tiền khách hàng đã chi tiêu cho đồ ngọt
15	MntGoldProds	Số nguyên	Tổng số tiền khách hàng đã chi tiêu cho các sản phẩm vàng
16	NumDealsPurchases	Số nguyên	Số lần khách hàng mua hàng với mức giá khuyến mãi hoặc ưu đãi
17	NumWebPurchases	Số nguyên	Số lần khách hàng mua hàng qua trang web của công ty
18	NumCatalogPurchases	Số nguyên	Số lần khách hàng mua hàng qua danh mục sản phẩm (catalog)
19	NumStorePurchases	Số nguyên	Số lần khách hàng mua hàng tại các cửa hàng vật lý
20	NumWebVisitsMonth	Số nguyên	Số lần khách hàng truy cập vào trang web của công ty trong một tháng

21	AcceptedCmp3	Số nguyên	Chỉ báo nhị phân (1 hoặc 0) cho biết khách hàng có chấp nhận chiến dịch tiếp thị thứ ba hay không
22	AcceptedCmp4	Số nguyên	Chỉ báo nhị phân (1 hoặc 0) cho biết khách hàng có chấp nhận chiến dịch tiếp thị thứ tư hay không
23	AcceptedCmp5	Số nguyên	Chỉ báo nhị phân (1 hoặc 0) cho biết khách hàng có chấp nhận chiến dịch tiếp thị thứ năm hay không
24	AcceptedCmp1	Số nguyên	Chỉ báo nhị phân (1 hoặc 0) cho biết khách hàng có chấp nhận chiến dịch tiếp thị đầu tiên hay không
25	AcceptedCmp2	Số nguyên	Chỉ báo nhị phân (1 hoặc 0) cho biết khách hàng có chấp nhận chiến dịch tiếp thị thứ hai hay không
26	Complain	Số nguyên	Chỉ báo nhị phân (1 hoặc 0) cho biết khách hàng có từng khiếu nại hay không
27	Z_CostContact	Số nguyên	Chi phí cố định khi liên hệ với khách hàng
28	Z_Revenue	Số nguyên	Doanh thu cố định từ mỗi phản hồi chiến dịch thành công
29	Response	Số nguyên	Chỉ báo nhị phân (1 hoặc 0) cho biết khách hàng có phản hồi với chiến dịch tiếp thị hay không

Bảng 1 Mô tả dữ liệu

CHƯƠNG 3. MÔ TẢ BÀI TOÁN

Bài toán của đề tài là phân cụm khách hàng dựa trên dữ liệu hành vi và đặc điểm cá nhân nhằm hỗ trợ các doanh nghiệp tối ưu hóa chiến lược kinh doanh và tiếp thị. Cụ thể, mục tiêu là sử dụng thuật toán phân cụm KMeans để nhóm các khách hàng có đặc điểm tương đồng thành các cụm riêng biệt. Điều này giúp doanh nghiệp hiểu rõ hơn về các nhóm khách hàng của mình, từ đó đưa ra các chiến lược tiếp thị và chăm sóc phù hợp nhằm tăng doanh số và sự hài lòng của khách hàng.

❖ **Các bước thực hiện bài toán bao gồm:**

1) *Thu thập dữ liệu:* Sử dụng dữ liệu khách hàng từ nguồn dữ liệu mở Kaggle.
2) *Xử lý tiền dữ liệu:* Tiền xử lý dữ liệu để loại bỏ các giá trị ngoại lệ, chuẩn hóa dữ liệu số và mã hóa dữ liệu phân loại nhằm đảm bảo dữ liệu đồng nhất và phù hợp cho phân cụm.
3) *Áp dụng thuật toán phân cụm KMeans:* Sử dụng thuật toán KMeans để nhóm các khách hàng dựa trên các đặc điểm đã chuẩn bị. Quy trình này bao gồm khởi tạo các centroid, gán cụm cho từng điểm dữ liệu và lặp lại quá trình đến khi hội tụ.
4) *Tối ưu hóa số lượng cụm:* Sử dụng phương pháp Elbow để xác định số lượng cụm (k) tối ưu, đảm bảo sự cân bằng giữa độ chi tiết của cụm và tính tổng quát của mô hình.
5) *Đánh giá kết quả:* Đánh giá hiệu quả phân cụm bằng cách sử dụng độ đo như SSE (Sum of Squared Errors), từ đó phân tích và so sánh các cụm để đưa ra kết luận hợp lý.

CHƯƠNG 4. PHÂN TÍCH DỮ LIỆU

4.1 Làm sạch dữ liệu

- Sau khi xóa các giá trị null, dữ liệu còn lại 2216 dòng

```
df = df.dropna()
print("After removing NA values:", df.count())
```

```
After removing NA values: 2216
```

Hình 2 Làm sạch dữ liệu - Xóa các giá trị Null

- Chuyển đổi dữ liệu Date thành số ngày liên tục, giúp mô hình dễ dàng xử lý dữ liệu

```
from pyspark.sql.functions import to_date

# Chuyển đổi sang DateType thành định dạng dd-MM-yyyy
df = df.withColumn("Dt_Customer", to_date(col("Dt_Customer"), "dd-MM-yyyy"))
```

```
newest_date = df.agg(spark_max("Dt_Customer")).first()[0]
oldest_date = df.agg(spark_min("Dt_Customer")).first()[0]
print("The newest customer's enrolment date:", newest_date)
print("The oldest customer's enrolment date:", oldest_date)
```

```
The newest customer's enrolment date: 2014-06-29
The oldest customer's enrolment date: 2012-07-30
```

```
from pyspark.sql.functions import datediff, lit
# Tạo cột Customer_For, tính số ngày từ ngày mới nhất đến ngày của mỗi khách hàng
df = df.withColumn("Customer_For", datediff(lit(newest_date), col("Dt_Customer")))
```

Hình 3 Làm sạch dữ liệu - Chuyển đổi dữ liệu date thành số ngày từ ngày khách hàng đăng ký đến ngày mới nhất của dữ liệu

- Định dạng lại các cột:
  - Tạo cột Age để lấy tuổi của khách hàng
  - Tạo cột Spent cho tổng chi tiêu các hạng mục khác nhau
  - Tạo cột Living_With (Partner/Alone) dựa trên cột Marital_Status, vì cột Marital Status phân loại quá nhiều giá trị không cần thiết

- Cột Children tính tổng số con trong gia đình
- Cột Family_Size tính tổng số thành viên trong gia đình
- Cột Is_Parent để xác định khách hàng có phải là cha mẹ không
- Gom nhóm các giá trị trong Education (UnderGraduate/ Graduate/ PostGraduate)
- Đổi tên 1 số cột giúp dữ liệu gọn gàng hơn

```
from pyspark.sql.functions import when, expr

# Tạo cột Age (tuổi của khách hàng)
df = df.withColumn("Age", lit(2024) - col("Year_Birth"))

# Tính tổng chi tiêu cho các mục khác nhau
df = df.withColumn("Spent",
                   col("MntWines") + col("MntFruits") + col("MntMeatProducts") +
                   col("MntFishProducts") + col("MntSweetProducts") + col("MntGoldProds"))

# Xác định tình trạng sống qua Marital_Status
df = df.withColumn("Living_With",
                   when(col("Marital_Status").isin("Married", "Together"), "Partner")
                   .otherwise("Alone"))

# Tạo cột tổng số con trong hộ gia đình
df = df.withColumn("Children", col("Kidhome") + col("Teenhome"))

# Tạo cột tổng số thành viên trong hộ gia đình
df = df.withColumn("Family_Size",
                   when(col("Living_With") == "Alone", 1)
                   .otherwise(2) + col("Children"))

# Tạo cột xác định khách hàng có phải là cha mẹ hay không
df = df.withColumn("Is_Parent", when(col("Children") > 0, 1).otherwise(0))

# Nhóm trình độ học vấn thành ba nhóm
df = df.withColumn("Education",
                   when(col("Education").isin("Basic", "2n Cycle"), "Undergraduate")
                   .when(col("Education") == "Graduation", "Graduate")
                   .when(col("Education").isin("Master", "PhD"), "Postgraduate"))

# Đổi tên các cột để rõ ràng hơn
df = df.withColumnRenamed("MntWines", "Wines") \
       .withColumnRenamed("MntFruits", "Fruits") \
       .withColumnRenamed("MntMeatProducts", "Meat") \
       .withColumnRenamed("MntFishProducts", "Fish") \
       .withColumnRenamed("MntSweetProducts", "Sweets") \
       .withColumnRenamed("MntGoldProds", "Gold")
```

Hình 4 Làm sạch dữ liệu - Format lại 1 số cột cần thiết

- Xóa các cột dữ liệu không cần thiết

```
# Xóa các cột không cần thiết
columns_to_drop = ["Marital_Status", "Dt_Customer", "Z_CostContact", "Z_Revenue", "Year_Birth"]
df = df.drop(*columns_to_drop)
```

Hình 5 Làm sạch dữ liệu - Xóa các cột dữ liệu không cần thiết

- Kết quả sau khi làm sạch dữ liệu

```
+----+-----------+------+--------+---------+-------+-----+------+----+------+----+----------------+--------------+-----------------+-----------------+-----------------+
|  ID|  Education|Income|Kidhome|Teenhome|Recency|Wines|Fruits|Meat|Fish|Sweets|Gold|NumDealsPurchases|NumWebPurchases|NumCatalogPurchases|NumStorePurchases|NumWebVisitsMonth|
+----+-----------+------+--------+---------+-------+-----+------+----+----+------+----+----------------+--------------+-----------------+-----------------+-----------------+
|5524|   Graduate| 58138|      0|       0|     58|  635|    88| 546| 172|    88|  88|               3|             8|               10|                4|                7|
|2174|   Graduate| 46344|      1|       1|     38|   11|     1|   6|   2|     1|   6|               2|             1|                1|                2|                5|
|4141|   Graduate| 71613|      0|       0|     26|  426|    49| 127| 111|    21|  42|               1|             8|                2|               10|                4|
|6182|   Graduate| 26646|      1|       0|     26|   11|     4|  20|  10|     3|   5|               2|             2|                0|                4|                6|
|5324|Postgraduate| 58293|      1|       0|     94|  173|    43| 118|  46|    27|  15|               5|             5|                3|                6|                5|
+----+-----------+------+--------+---------+-------+-----+------+----+----+------+----+----------------+--------------+-----------------+-----------------+-----------------+
only showing top 5 rows
```

```
+------------+------------+------------+------------+------------+-------+--------+-------------+---+-----+-----------+--------+-----------+---------+
|AcceptedCmp3|AcceptedCmp4|AcceptedCmp5|AcceptedCmp1|AcceptedCmp2|Complain|Response|Customer_For|Age|Spent|Living_With|Children|Family_Size|Is_Parent|
+------------+------------+------------+------------+------------+-------+--------+-------------+---+-----+-----------+--------+-----------+---------+
|           0|           0|           0|           0|           0|      0|       1|          663| 67| 1617|      Alone|       0|          1|        0|
|           0|           0|           0|           0|           0|      0|       0|          113| 70|   27|      Alone|       2|          3|        1|
|           0|           0|           0|           0|           0|      0|       0|          312| 59|  776|    Partner|       0|          2|        0|
|           0|           0|           0|           0|           0|      0|       0|          139| 40|   53|    Partner|       1|          3|        1|
|           0|           0|           0|           0|           0|      0|       0|          161| 43|  422|    Partner|       1|          3|        1|
+------------+------------+------------+------------+------------+-------+--------+-------------+---+-----+-----------+--------+-----------+---------+
```

Hình 6 Kết quả sau khi làm sạch dữ liệu

4.2 Phân tích dữ liệu

4.2.1 Với dữ liệu phân loại

➢ Thực hiện đếm các giá trị trong các cột phân loại:

+ Với Education: Graduate (1116), PostGraduate (846), Undergraduate (254)

+ Với Living_With: Partner (1430), Alone (786)

```
from pyspark.sql.functions import col, count

categorical_cols = ["Education", "Living_With"]

for c in categorical_cols:
    print(f"Frequency table for {c}:")
    df.groupBy(c).count().orderBy("count", ascending=False).show()
```

```
Frequency table for Education:
+-------------+-----+
|    Education|count|
+-------------+-----+
|     Graduate| 1116|
| Postgraduate|  846|
|Undergraduate|  254|
+-------------+-----+

Frequency table for Living_With:
+-----------+-----+
|Living_With|count|
+-----------+-----+
|    Partner| 1430|
|      Alone|  786|
+-----------+-----+
```

Hình 7 Phân tích dữ liệu - Đếm số các giá trị trong các cột phân loại

➢ Thực hiện tìm giá trị phổ biến nhất và giá trị ít phổ biến nhất

```
from pyspark.sql import Row

# Hàm tìm giá trị phổ biến nhất và hiếm nhất của một cột
def get_most_and_least_common(df, col_name):
    counts = df.groupBy(col_name).count().orderBy("count", ascending=False)
    most_common = counts.first()
    least_common = counts.orderBy("count", ascending=True).first()
    return most_common, least_common

# Danh sách các cột để phân tích
attributes = ['Education', 'Living_With', 'Income', 'Recency', 'Wines', 'Fruits', 'Meat', 'Fish', 'Sweets',
              'Gold', 'Spent', 'Customer_For', 'Age', 'NumDealsPurchases',
              'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
              'NumWebVisitsMonth', 'Children', 'Family_Size', 'Is_Parent',
              'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
              'AcceptedCmp2', 'Complain', 'Response']

rows = []

# Tính giá trị phổ biến nhất và hiếm nhất cho từng cột
for attr in attributes:
    most_common, least_common = get_most_and_least_common(df, attr)
    rows.append(Row(statistic=f"most_common_{attr}", value=most_common[attr], count=most_common['count']))
    rows.append(Row(statistic=f"least_common_{attr}", value=least_common[attr], count=least_common['count']))

# Tạo DataFrame từ kết quả
stats_df = spark.createDataFrame(rows)
```

Hình 8 Phân tích dữ liệu - Hàm tìm giá trị phổ biến/ ít phổ biến nhất

➢ Kết quả giá trị phổ biến nhất, hiếm nhất của Education và Living_With

STT	Thuộc tính	Giá trị phổ biến nhất	Giá trị ít phổ biến nhất
1	Education	Graduate (1116)	Undergraduate (254)
2	Living Status	Partner (1430)	Alone (786)

Bảng 2 Bảng các giá trị phổ biến, ít phổ biến nhất của cột phân loại

4.2.2 Với dữ liệu số

➢ Thực hiện tính count, mean, stddev, min, max của các cột số

```
def printMedian(numeric_cols):
    for c in numeric_cols:
        median = df.approxQuantile(c, [0.5], 0.001)[0]
        print(f"Median of {c}: {median}")
```

```
numeric_cols = ['Income', 'Recency', 'Wines', 'Fruits', 'Meat', 'Fish']

df.select(numeric_cols).describe().show()
printMedian(numeric_cols)
```

```
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|summary|            Income|          Recency|            Wines|           Fruits|             Meat|             Fish|
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|  count|              2216|             2216|             2216|             2216|             2216|             2216|
|   mean| 52247.25135379061|49.01263537906137|305.09160649819495|26.356046931407942|166.99593862815885|37.63763537906137|
| stddev|25173.076660901414|28.94835165862499|  337.327920116672|39.793916518238674|224.28327275898198|54.75208241485154|
|    min|              1730|                0|                0|                0|                0|                0|
|    max|            666666|               99|             1493|              199|             1725|              259|
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+

Median of Income: 51369.0
Median of Recency: 49.0
Median of Wines: 174.0
Median of Fruits: 8.0
Median of Meat: 68.0
Median of Fish: 12.0
```

```
+-------+-----------------+-----------------+-----------------+-----------------+-----------------+------------------+
|summary|           Sweets|             Gold|            Spent|      Customer_For|              Age| NumDealsPurchases|
+-------+-----------------+-----------------+-----------------+-----------------+-----------------+------------------+
|  count|             2216|             2216|             2216|             2216|             2216|              2216|
|   mean|27.028880866425993|43.96525270758123|607.0753610108303| 353.5212093862816| 55.17960288808664|2.3235559566787005|
| stddev| 41.07204566698224|51.81541437447363|602.9004763523358|202.43466744562002|11.985554097352384|1.9237156460114953|
|    min|                0|                0|                5|                0|               28|                 0|
|    max|              262|              321|             2525|              699|              131|                15|
+-------+-----------------+-----------------+-----------------+-----------------+-----------------+------------------+

Median of Sweets: 8.0
Median of Gold: 24.0
Median of Spent: 396.0
Median of Customer_For: 355.0
Median of Age: 54.0
Median of NumDealsPurchases: 2.0
```

```
+-------+------------------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|summary|   NumWebPurchases|NumCatalogPurchases|NumStorePurchases|NumWebVisitsMonth|         Children|       Family_Size|        Is_Parent|
+-------+------------------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|  count|              2216|              2216|             2216|             2216|             2216|             2216|             2216|
|   mean|  4.08528880866426|  2.671028880866426|5.800992779783393|5.319043321299639| 0.947202166064982| 2.592509025270758| 0.7143501805054152|
| stddev|2.7409510810147255| 2.9267336360080134|3.250784785670525|2.425358540606443|0.7490619124789906|0.9057218788428764|0.4518253244556944|
|    min|                 0|                 0|                0|                0|                0|                1|                0|
|    max|                27|                28|               13|               20|                3|                5|                1|
+-------+------------------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+

Median of NumWebPurchases: 4.0
Median of NumCatalogPurchases: 2.0
Median of NumStorePurchases: 5.0
Median of NumWebVisitsMonth: 6.0
Median of Children: 1.0
Median of Family_Size: 3.0
Median of Is_Parent: 1.0
```

```
+-------+------------------+------------------+------------------+------------------+------------------+------------------+------------------+
|summary|      AcceptedCmp3|      AcceptedCmp4|      AcceptedCmp5|      AcceptedCmp1|      AcceptedCmp2|          Complain|          Response|
+-------+------------------+------------------+------------------+------------------+------------------+------------------+------------------+
|  count|              2216|              2216|              2216|              2216|              2216|              2216|              2216|
|   mean|0.07355595667870037|0.074007220216606 5|0.07310469314079422|0.06407942238267147|0.013537906137184115|0.009476534296028 88|0.15027075812274368|
| stddev|0.26110580876067324|0.2618417283260366|0.26036702668143424| 0.2449496396591494| 0.11558832299800825| 0.09690700384872 11|0.35741727012906527|
|    min|                 0|                 0|                 0|                 0|                 0|                 0|                 0|
|    max|                 1|                 1|                 1|                 1|                 1|                 1|                 1|
+-------+------------------+------------------+------------------+------------------+------------------+------------------+------------------+

Median of AcceptedCmp3: 0.0
Median of AcceptedCmp4: 0.0
Median of AcceptedCmp5: 0.0
Median of AcceptedCmp1: 0.0
Median of AcceptedCmp2: 0.0
Median of Complain: 0.0
Median of Response: 0.0
```

Hình 9 Phân tích dữ liệu - Tính count, mean, stddev, min, max của các cột số

- Nhận xét: Một số cột cho giá trị Mean và Median vẫn còn chênh lệch có thể do các giá trị ngoại lệ => Cần thực hiện xử lý ngoại lệ
- Tính các giá trị phổ biến nhất, ít phổ biến nhất tương tự với thuộc tính phân loại. Ta có kết quả sau:

STT	Thuộc tính	Giá trị phổ biến nhất	Giá trị ít phổ biến nhất
1	Income	7500 (12)	82582 (1)
2	Recency	56 (37)	44 (11)
3	Wines	2 (42)	463 (1)
4	Fruits	0 (395)	101 (1)
5	Meat	7 (53)	471 (1)
6	Fish	0 (379)	81 (1)
7	Sweets	0 (413)	128 (1)
8	Gold	1 (71)	148 (1)
9	Spent	22 (18)	148 (1)
10	Customer_For	667 (12)	148 (1)
11	Age	48 (89)	84 (1)
12	NumDealsPurchases	1 (960)	12 (3)
13	NumWebPurchases	2 (368)	27 (1)
14	NumCatalogPurchases	0 (576)	22 (1)
15	NumStorePurchases	3 (484)	1 (6)
16	NumWebVisitsMonth	7 (387)	13 (1)
17	Children	1 (1117)	3 (50)

18	Family_Size	3 (880)	5 (31)
19	Is_Parent	1 (1583)	0 (633)
20	AcceptedCmp1	0 (2053)	1 (163)
21	AcceptedCmp2	0 (2052)	1 (164)
22	AcceptedCmp3	0 (2054)	1 (162)
23	AcceptedCmp4	0 (2074)	1 (142)
24	AcceptedCmp5	0 (2186)	1 (30)
25	Complain	0 (2195)	1 (21)
26	Response	0 (1883)	1 (333)

Hình 10 Bảng các giá trị phổ biến, ít phổ biến nhất của cột số

CHƯƠNG 5. TIỀN XỬ LÝ DỮ LIỆU

5.1 Đối với dữ liệu phân loại

Mã hóa cột dữ liệu phân loại là một bước quan trọng trong quá trình xử lý dữ liệu, đặc biệt đối với các mô hình máy học. Lý do là vì các mô hình thường chỉ làm việc với dữ liệu dạng số, trong khi dữ liệu phân loại lại ở dạng văn bản hoặc giá trị không số.

❖ **Mã hóa one-hot (One-Hot Encoding) [1]**

Mã hóa one-hot là một phương pháp chuyển đổi dữ liệu phân loại thành các cột nhị phân (binary columns). Mỗi giá trị riêng biệt của một biến phân loại được biểu diễn bằng một cột riêng, trong đó:

+ Giá trị tương ứng được đặt là 1.
+ Các giá trị còn lại được đặt là 0.

```
from pyspark.sql.functions import when

# Các cột phân loại
categorical_cols = ["Education", "Living_With"]

# Mã hóa one-hot cho cột 'Education'
df = df.withColumn("Education_Graduate", when(col("Education") == "Graduate", 1).otherwise(0))
df = df.withColumn("Education_Postgraduate", when(col("Education") == "Postgraduate", 1).otherwise(0))
df = df.withColumn("Education_Undergraduate", when(col("Education") == "Undergraduate", 1).otherwise(0))

# Mã hóa one-hot cho cột 'Living_With'
df = df.withColumn("Living_With_Alone", when(col("Living_With") == "Alone", 1).otherwise(0))
df = df.withColumn("Living_With_Partner", when(col("Living_With") == "Partner", 1).otherwise(0))

# Loại bỏ các cột phân loại gốc
df = df.drop(*categorical_cols)

df.show(5)
```

Hình 11 Tiền xử lý dữ liệu - Code mã hóa one-hot cho dữ liệu phân loại

```
+-----------------+---------------------+----------------------+-----------------+-------------------+
|Education_Graduate|Education_Postgraduate|Education_Undergraduate|Living_With_Alone|Living_With_Partner|
+-----------------+---------------------+----------------------+-----------------+-------------------+
|                1|                    0|                     0|                1|                  0|
|                1|                    0|                     0|                1|                  0|
|                1|                    0|                     0|                0|                  1|
|                1|                    0|                     0|                0|                  1|
|                0|                    1|                     0|                0|                  1|
+-----------------+---------------------+----------------------+-----------------+-------------------+
```

Hình 12 Tiền xử lý dữ liệu - Kết quả sau khi thực hiện mã hóa one-hot cho dữ liệu phân loại

5.2 Đối với dữ liệu số

5.2.1 Thực hiện xử lý các giá trị ngoại lệ

Ngoại lệ (Outliers) là các giá trị bất thường hoặc nằm cách biệt so với phần lớn các giá trị còn lại trong dữ liệu. Những giá trị này có thể xuất hiện do lỗi nhập liệu, quy trình thu thập dữ liệu không chính xác, hoặc do bản chất tự nhiên của dữ liệu. Việc xử lý ngoại lệ là cần thiết vì chúng có thể:

+ Gây ảnh hưởng tiêu cực đến mô hình học máy (như tăng độ lệch hoặc sai số).
+ Làm sai lệch kết quả phân tích thống kê.

❖ **Phương pháp sử dụng IQR (Interquartile Range) [2]**

IQR là một kỹ thuật để phát hiện và xử lý ngoại lệ, dựa trên thống kê phân vị của dữ liệu.

Bước 1: Tính IQR

- Q1 (quartile 1): Giá trị tại phân vị thứ 25% (25% dữ liệu nhỏ hơn hoặc bằng Q1).
- Q3 (quartile 3): Giá trị tại phân vị thứ 75% (75% dữ liệu nhỏ hơn hoặc bằng Q3).
- IQR (Interquartile Range): Là khoảng cách giữa Q3 và Q1, được tính bằng:

$$\boldsymbol{IQR = Q_3 - Q_1}$$

Bước 2: Xác định giới hạn

- Lower Bound (giới hạn dưới):

$$\boldsymbol{LowerBound = Q_1 - 1.5 * IQR}$$

- Upper Bound (giới hạn trên):

$$\boldsymbol{UpperBound = Q_3 + 1.5 * IQR}$$

Bước 3: Phát hiện và xử lý ngoại lệ

- Bất kỳ giá trị nào:
  + Nhỏ hơn Lower Bound => được coi là ngoại lệ dưới.
  + Lớn hơn Upper Bound => được coi là ngoại lệ trên.
- Xử lý:
  + Gán giá trị nhỏ hơn Lower Bound thành Lower Bound.
  + Gán giá trị lớn hơn Upper Bound thành Upper Bound.
  + Giữ nguyên các giá trị trong khoảng [Lower Bound, Upper Bound].

```python
from pyspark.sql.functions import col, when

# Giả sử numeric_cols là danh sách các cột số trong DataFrame df
# Đây là những cột mà bạn muốn xử lý ngoại lệ.
numeric_cols = ['Income', 'Recency', 'Wines', 'Fruits', 'Meat', 'Fish', 'Sweets',
                'Gold', 'Spent', 'Customer_For', 'Age', 'NumDealsPurchases',
                'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
                'NumWebVisitsMonth', 'Children', 'Family_Size']

# Xử lý ngoại lệ cho mỗi cột số
for c in numeric_cols:
    # Tính Q1 và Q3 sử dụng approxQuantile
    # Sai số 0.001 để đảm bảo độ chính xác tương đối cao
    q1, q3 = df.approxQuantile(c, [0.25, 0.75], 0.001)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    # Cắt giá trị ngoại lệ:
    # - Nếu giá trị < lower_bound => gán bằng lower_bound
    # - Nếu giá trị > upper_bound => gán bằng upper_bound
    # Giữ nguyên nếu nằm trong khoảng [lower_bound, upper_bound]
    df = df.withColumn(
        c,
        when(col(c) < lower_bound, lower_bound)
        .when(col(c) > upper_bound, upper_bound)
        .otherwise(col(c))
    )
```

Hình 13 Tiền xử lý dữ liệu - Code xử lý giá trị ngoại lệ

5.2.2 Thực hiện chuẩn hóa Z-score [3]

Chuẩn hóa Z-Score là một phương pháp phổ biến trong xử lý dữ liệu số để đưa dữ liệu về một dạng phân phối chuẩn (mean = 0, standard deviation = 1). Kỹ thuật này giúp giảm sự khác biệt về tỷ lệ giữa các cột và làm cho các mô hình học máy hoạt động hiệu quả hơn.

Chuẩn hóa Z-Score được tính bằng công thức:

$$\boldsymbol{z = \frac{x - \mu}{\sigma}}$$

Trong đó:

z: Điểm chuẩn hóa (Z-Score).

x: Giá trị gốc của dữ liệu.

μ: Giá trị trung bình (mean) của cột.

σ: Độ lệch chuẩn (standard deviation) của cột.

CHƯƠNG 6. KHAI THÁC DỮ LIỆU VỚI KMEANS

6.1 Thuật toán Kmeans

6.2.1 Lý do lựa chọn thuật toán

❖ **Đơn giản và hiệu quả**

K-Means là thuật toán phân cụm phổ biến và đơn giản nhất, dễ dàng triển khai và hiểu rõ. Thuật toán có thể xử lý nhanh với dữ liệu lớn, vì số bước tính toán là khá ít và yêu cầu tài nguyên tính toán không quá lớn.

❖ **Tính ứng dụng cao trong phân nhóm khách hàng**

K-Means rất hiệu quả khi các nhóm khách hàng có sự phân bổ khá đồng đều và không có quá nhiều sự chồng chéo giữa các nhóm.

Thuật toán giúp dễ dàng nhận diện các nhóm khách hàng chính để phát triển các chiến lược marketing, chăm sóc khách hàng hoặc tạo ra các sản phẩm, dịch vụ phù hợp.

❖ **Dễ dàng áp dụng và điều chỉnh**

K-Means rất dễ điều chỉnh để phù hợp với các đặc điểm cụ thể của bài toán phân nhóm khách hàng. Các tham số như số cụm (k) có thể được điều chỉnh thông qua phương pháp lựa chọn như Elbow method để tìm số cụm tối ưu.

6.2.2 Định nghĩa

K-Means là một thuật toán phân cụm không giám sát (unsupervised clustering) phổ biến, được sử dụng để nhóm các đối tượng dữ liệu thành k cụm (clusters) dựa trên sự tương đồng giữa chúng. Mỗi cụm được đại diện bởi một tâm (centroid), thường là trung bình của tất cả các điểm trong cụm.

6.2.3 Mục tiêu của thuật toán

Tối thiểu hóa tổng bình phương khoảng cách (Sum of Squared Errors - SSE) từ các điểm dữ liệu đến centroid của cụm tương ứng:

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

6.2.4 Các bước thực hiện

1) Khởi tạo:

Chọn ngẫu nhiên k điểm làm centroid ban đầu.

2) Phân cụm:

Với mỗi điểm dữ liệu, tính khoảng cách đến các centroid và gán nó vào cụm gần nhất.

3) Cập nhật centroid:

Tính trung bình tất cả các điểm trong mỗi cụm để cập nhật centroid.

4) Kiểm tra hội tụ:

Thuật toán dừng nếu: Centroid không thay đổi hoặc số vòng lặp đạt giới hạn tối đa.

5) Kết quả:

k cụm với mỗi cụm có một tập hợp các điểm dữ liệu gần nhau.

6.2 Cài đặt thuật toán

```python
import math

def euclidean_distance(p1, p2):
    return math.sqrt(sum((a - b)**2 for a, b in zip(p1, p2)))

def assign_to_cluster(point, centroids):
    distances = [euclidean_distance(point, c) for c in centroids]
    cluster_id = distances.index(min(distances))
    return cluster_id

def compute_sse(rdd_points, centroids):
    return rdd_points.map(lambda x: min([euclidean_distance(x[1], c)**2 for c in centroids])).sum()
```

Hình 14 Định nghĩa các hàm cần thiết cho thuật toán

```
# Định nghĩa phạm vi các giá trị k để thử
k_values = range(2, 11)  # Từ k=2 đến k=10

# Khởi tạo danh sách để lưu kết quả SSE cho mỗi k
sse_results = []

# Định nghĩa các tham số chung
max_iterations = 20
convergence_threshold = 1e-4

# Lặp qua từng giá trị k để thực hiện K-Means và tính SSE
for k in k_values:
    print(f"Running K-Means for k={k}")

    # Khởi tạo centroids ngẫu nhiên từ dữ liệu
    centroids = rdd_points.map(lambda x: x[1]).takeSample(False, k, seed=42)

    for i in range(max_iterations):
        # Gán cụm cho mỗi điểm
        clusters = rdd_points.map(lambda x: (assign_to_cluster(x[1], centroids), (x[1], 1)))

        # Tính centroid mới bằng cách lấy trung bình các điểm trong cụm
        new_centroids = clusters.reduceByKey(
            lambda a, b: ([x + y for x, y in zip(a[0], b[0])], a[1] + b[1])
        ).mapValues(
            lambda x: [val / x[1] for val in x[0]]
        )
```

```
        # Chuyển centroids thành list để dễ dàng so sánh
        new_centroids_list = new_centroids.collect()
        new_centroids_dict = dict(new_centroids_list)

        # Kiểm tra hội tụ: tính khoảng cách giữa centroids cũ và mới
        max_distance = 0
        for i, centroid in enumerate(centroids):
            if i in new_centroids_dict:
                distance = euclidean_distance(centroid, new_centroids_dict[i])
                max_distance = max(max_distance, distance)

        # Kiểm tra hội tụ
        if max_distance < convergence_threshold:
            print(f"  Converged at iteration: {i+1}")
            centroids = [new_centroids_dict[key] for key in sorted(new_centroids_dict.keys())]
            break

        centroids = [new_centroids_dict[key] for key in sorted(new_centroids_dict.keys())]

    # Tính SSE cho k hiện tại (giữ RDD phân tán)
    sse = rdd_points.map(lambda x: min([euclidean_distance(x[1], c)**2 for c in centroids])).sum()
    print(f"  SSE for k={k}: {sse}")

    # Lưu kết quả
    sse_results.append((k, sse))
```

Hình 15 Thuật toán Kmeans

❖ **Luồng hoạt động của thuật toán**

1) Khởi tạo giá trị k (số cụm) và các tham số chung

- `k_values = range(2, 11)`: Đây là phạm vi các giá trị của k mà bạn muốn thử nghiệm (từ 2 đến 10). Mỗi giá trị k đại diện cho số lượng cụm (clusters) mà thuật toán sẽ phân chia dữ liệu thành.
- `sse_results = []`: Một danh sách rỗng để lưu trữ kết quả SSE (Sum of Squared Errors) cho mỗi giá trị của k.
- `max_iterations = 20`: Số lần tối đa thuật toán sẽ lặp lại để cải thiện các centroids.
- `convergence_threshold = 1e-4`: Ngưỡng hội tụ, nghĩa là khi khoảng cách giữa các centroids cũ và mới nhỏ hơn giá trị này, thuật toán sẽ dừng lại.

2) Khởi tạo centroid và lặp qua các giá trị k

- Lặp qua từng giá trị k trong `k_values` để thực hiện K-Means cho mỗi giá trị của k.
  - Khởi tạo `centroids` ngẫu nhiên từ dữ liệu với `rdd_points.map(lambda x: x[1]).takeSample(False, k, seed=42)`, giúp chọn k điểm ngẫu nhiên làm centroids ban đầu.

3) Thuật toán K-Means

Trong mỗi vòng lặp:

- Gán mỗi điểm vào cụm gần nhất

`clusters = rdd_points.map(lambda x: (assign_to_cluster(x[1], centroids), (x[1], 1)))`: Mỗi điểm trong dữ liệu được gán vào cụm gần nhất, theo chỉ số centroid gần nhất với điểm đó. assign_to_cluster là hàm xác định cụm của một điểm dựa trên centroid gần nhất.

- Cập nhật centroid

`new_centroids = clusters.reduceByKey(...)`: Sau khi các điểm được phân loại vào các cụm, ta tính lại các centroid mới bằng cách lấy trung bình của các điểm trong mỗi cụm. Phép reduceByKey kết hợp các giá trị trong cùng một cụm và tính trung bình để tạo centroid mới.

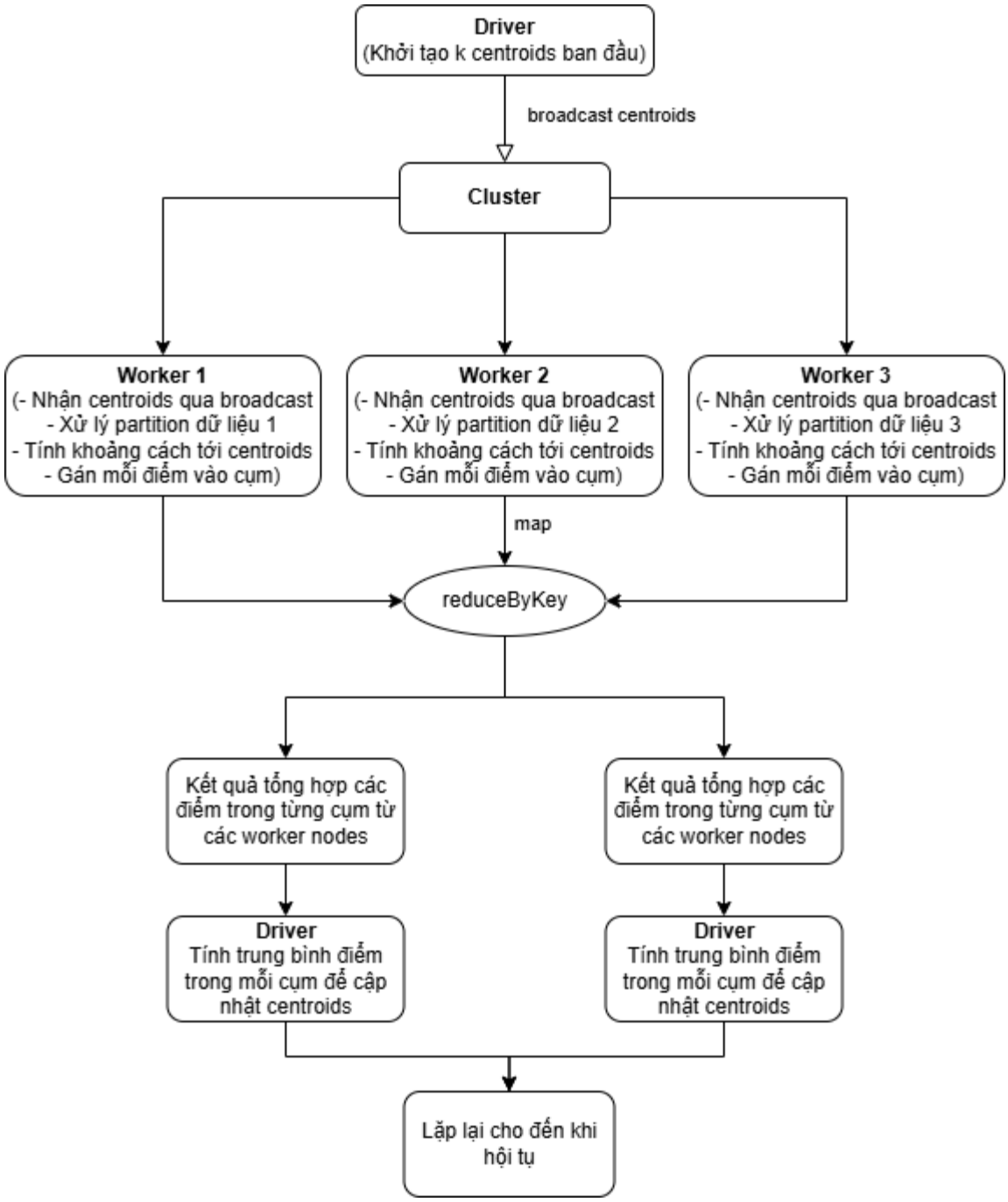
Sau đó, hàm `mapValues(lambda x: [val / x[1] for val in x[0]])` tính toán lại centroid cho mỗi cụm bằng cách chia tổng các giá trị điểm trong cụm cho số lượng điểm trong cụm đó.

- Kiểm tra hội tụ:

Tính khoảng cách giữa centroid cũ và mới: Mỗi lần cập nhật centroid mới, bạn kiểm tra khoảng cách giữa các centroid cũ và mới. Nếu khoảng cách lớn hơn một ngưỡng, thuật toán tiếp tục lặp lại các bước trên.

Hội tụ: Nếu khoảng cách giữa centroids cũ và mới nhỏ hơn ngưỡng hội tụ, thuật toán dừng lại và kết quả là các centroids cuối cùng.

6.3 Song song hóa giải thuật



Hình 16 Song song hóa giải thuật

❖ **Chi tiết quá trình:**

1) Khởi tạo centroids và phân phối (Broadcasting):

Driver: Bắt đầu thuật toán bằng cách khởi tạo số lượng cụm k và các centroid ban đầu. Sau đó, các centroid này sẽ được broadcast tới tất cả các worker nodes. Điều này giúp mỗi worker có thể xử lý dữ liệu của mình mà không cần phải truy cập trực tiếp vào Driver mỗi lần.

2) Phân chia dữ liệu thành các partition:

Dữ liệu sẽ được chia thành nhiều partition, mỗi partition này sẽ được phân phối đến một worker node. Mỗi worker node sẽ làm việc độc lập trên một phần dữ liệu của mình.

3) Xử lý dữ liệu trên các worker nodes:

Mỗi worker nhận được centroid từ Driver (qua broadcast) và xử lý dữ liệu trong partition của mình.

Mỗi điểm dữ liệu trong partition sẽ tính toán khoảng cách tới các centroid và gán điểm đó vào cụm gần nhất.

map sẽ được áp dụng để gán điểm vào cluster, và sau đó, kết quả sẽ được gửi lại về Driver thông qua các phép toán phân tán như reduceByKey.

4) Tính toán centroid mới:

Sau khi tất cả các worker nodes hoàn thành việc gán các điểm vào cụm, Driver sẽ thu thập kết quả từ các worker. Sau đó, tính toán lại centroid cho mỗi cụm bằng cách lấy trung bình các điểm trong cùng một cụm.

5) Kiểm tra hội tụ:

Sau mỗi vòng lặp, Driver tính toán sự thay đổi của centroid và kiểm tra xem thuật toán có hội tụ chưa. Nếu có, thuật toán kết thúc, nếu không, thuật toán sẽ tiếp tục với các centroid mới.

CHƯƠNG 7. ĐÁNH GIÁ KẾT QUẢ

7.1 Sum of Squared Errors (SSE)

7.1.1 Định nghĩa

SSE trong K-Means là tổng của các bình phương khoảng cách giữa mỗi điểm dữ liệu và trung tâm cụm của nó.

7.1.2 Công thức tính SSE

$$SSE = \sum_{k=1}^{K} \sum_{i=1}^{n_k} |x_i - c_k|^2$$

K: Số cụm (clusters).

n_k: Số lượng điểm trong cụm thứ

x_i: Điểm dữ liệu i thuộc cụm k

c_k: Centroid (tâm cụm) của cụm

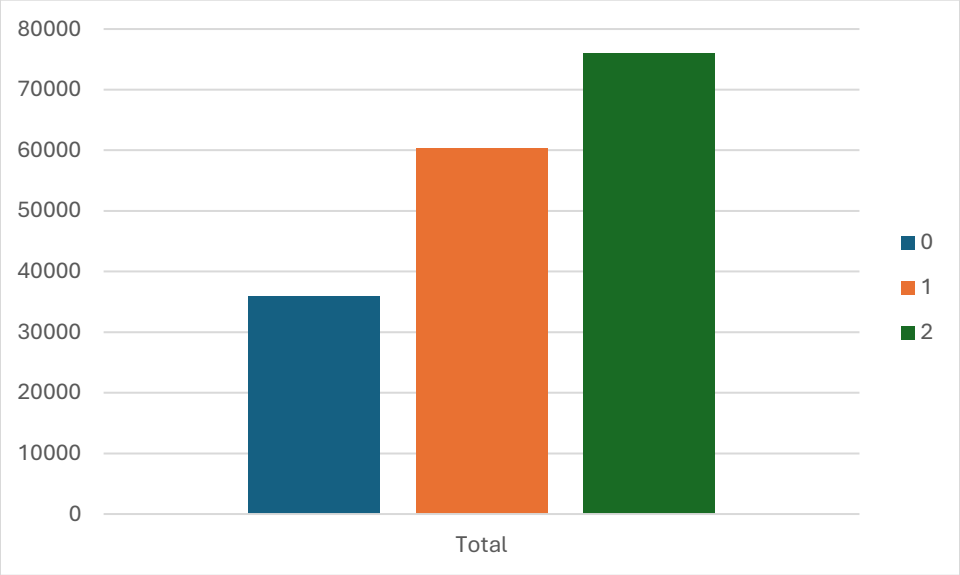
$|x_i - c_k|^2$: Khoảng cách Euclidean bình phương giữa điểm x_i và centroid c_k

```
Running K-Means for k=2
  Converged at iteration: 2
  SSE for k=2: 48215.17278440477
Running K-Means for k=3
  Converged at iteration: 3
  SSE for k=3: 43188.20242947265
Running K-Means for k=4
  SSE for k=4: 40848.80333983346
Running K-Means for k=5
  SSE for k=5: 38559.207199458324
Running K-Means for k=6
  Converged at iteration: 6
  SSE for k=6: 37493.3884607082
Running K-Means for k=7
  Converged at iteration: 7
  SSE for k=7: 36305.355804365776
Running K-Means for k=8
  SSE for k=8: 35394.322807038356
Running K-Means for k=9
  SSE for k=9: 33579.82712514724
Running K-Means for k=10
  SSE for k=10: 32721.91198359116
+---+------------------+
|  k|               SSE|
+---+------------------+
|  2| 48215.17278440477|
|  3| 43188.20242947265|
|  4| 40848.80333983346|
|  5|38559.207199458324|
|  6|  37493.3884607082|
|  7|36305.355804365776|
|  8|35394.322807038356|
|  9| 33579.82712514724|
| 10| 32721.91198359116|
+---+------------------+
```

Hình 17 Kết quả độ đo SSE theo số lượng cụm k

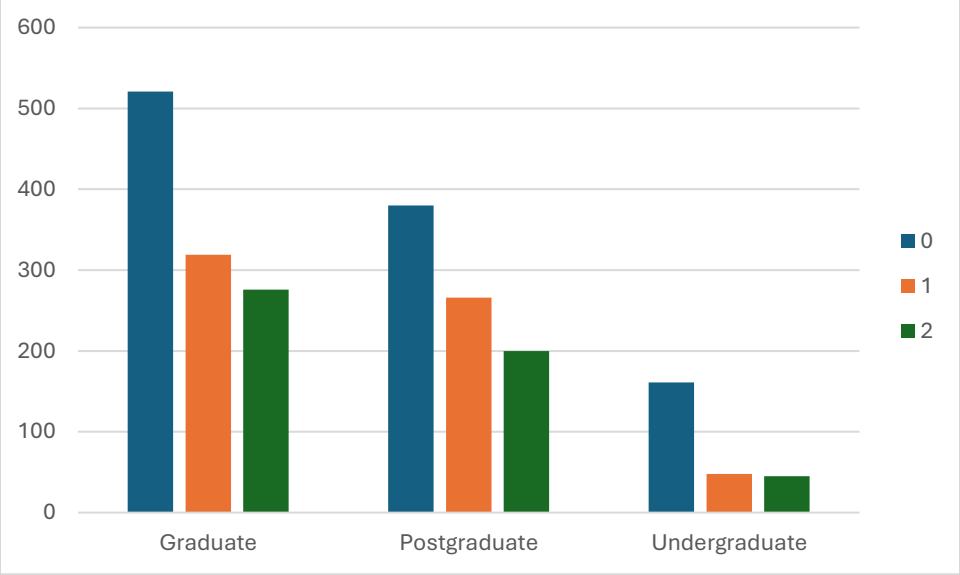
Kết luận: Dựa vào bảng độ đo theo số lượng cụm trên, chọn k = 3 là số lượng cụm tối ưu vì với các giá trị k còn lại, kết quả đánh giá SSE giảm không đáng kể

7.2 Phân tích đặc trưng của các cụm



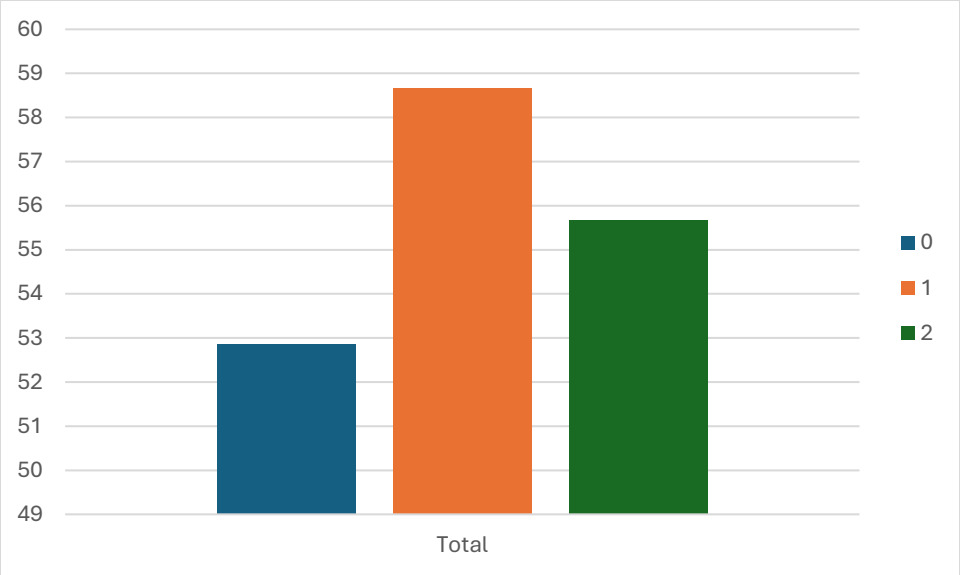
Hình 18 Thu nhập bình quân của 3 cụm

Nhận xét: Kết quả cho thấy cụm 0 cho thu nhập trung bình năm khoảng 35,000 USD; cụm 1 trung bình 60,000 USD; cụm 2 trên 70,000 USD



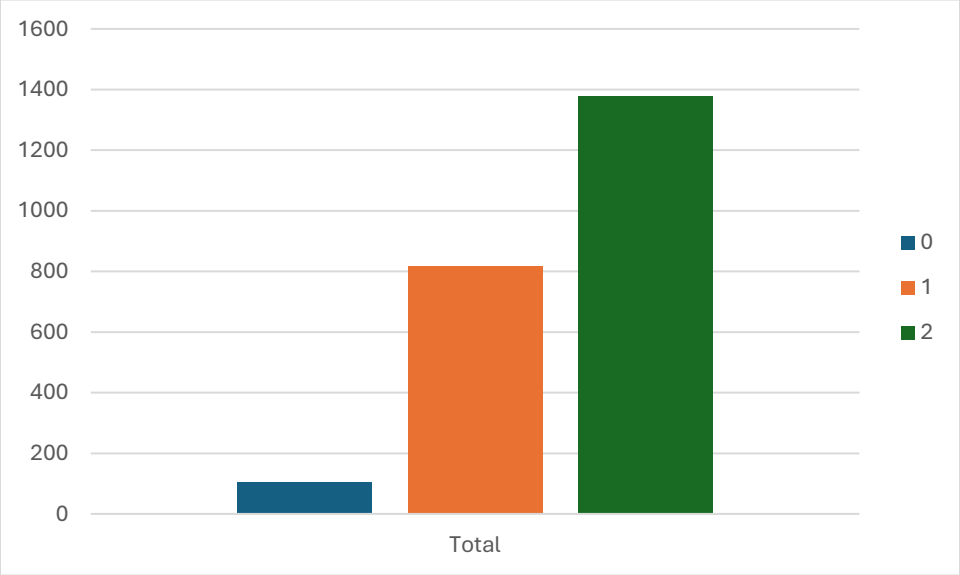
Hình 19 Trình độ học vấn của các cụm

Nhận xét: Kết quả cho thấy cụm 0 có số lượng Undergraduate vẫn còn tỷ trọng tương đối cao so với 2 cụm còn lại



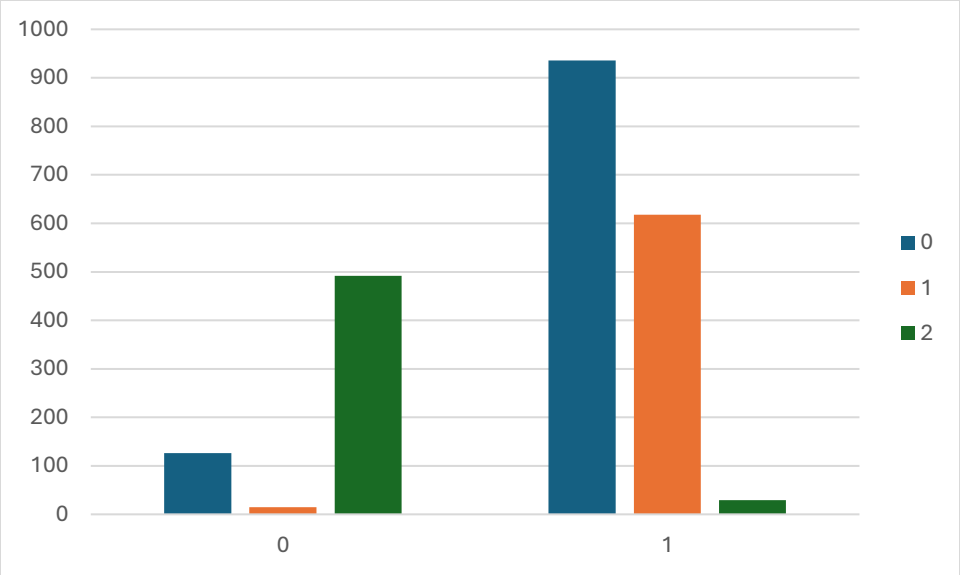
Hình 20 Tuổi trung bình của các cụm

Nhận xét: Tuổi trung bình của cụm 1 cao nhất với hơn 58 tuổi; tiếp theo là cụm 2 với hơn 55 tuổi và cuối cùng là cụm 0 với hơn 52 tuổi



Hình 21 Chi tiêu trung bình của các cụm

Nhận xét: Cụm 2 có chi tiêu trung bình cao nhất với xấp xỉ 1400 USD, tiếp theo là cụm 1 với 800 USD và cuối cùng là cụm 0 với khoảng 100 USD



Hình 22 Is_Parent theo các cụm

Nhận xét: Kết quả cho thấy cụm 0 và cụm 1 là các bậc cha mẹ là chủ yếu; trong khi cụm 2 thì ngược lại

CHƯƠNG 8. KẾT LUẬN

8.1 Ưu điểm

+ Đơn giản và dễ hiểu: K-Means là một thuật toán phân nhóm cơ bản và dễ thực hiện. Quá trình thuật toán có thể được hiểu rõ, từ việc gán điểm vào các cụm cho đến việc tính toán các centroid của các cụm.
+ Nhanh chóng với dữ liệu lớn: K-Means có thể xử lý lượng dữ liệu lớn một cách hiệu quả, đặc biệt khi số lượng cụm k không quá lớn và số lượng điểm dữ liệu cũng không quá lớn.
+ Dễ dàng triển khai: Thuật toán có thể được triển khai trên các nền tảng tính toán phân tán như Apache Spark, giúp tối ưu hóa quá trình phân nhóm cho dữ liệu lớn và phân tán.
+ Ứng dụng thực tiễn rộng rãi: K-Means rất phổ biến trong phân tích khách hàng, phân nhóm thị trường, phân tích hành vi người dùng, xác định các nhóm khách hàng tiềm năng dựa trên các đặc điểm như độ tuổi, thu nhập, sở thích,...
+ Tính linh hoạt: K-Means có thể áp dụng được cho nhiều loại dữ liệu khác nhau, miễn là các đặc điểm của khách hàng có thể biểu diễn dưới dạng các vector số.

8.2 Hạn chế

– Cần xác định trước số cụm k: Một trong những hạn chế lớn nhất của K-Means là bạn phải xác định số lượng cụm k trước khi thực hiện phân nhóm.
– Nhạy cảm với dữ liệu ngoại lệ (outliers): K-Means rất nhạy cảm với các điểm dữ liệu ngoại lai, vì chúng có thể ảnh hưởng lớn đến vị trí centroid, làm cho kết quả phân nhóm không chính xác.
– Chọn khởi tạo centroid ban đầu: Nếu các centroid ban đầu được khởi tạo không tốt, thuật toán có thể hội tụ về một nghiệm không tối ưu. Điều này có thể dẫn đến kết quả phân nhóm không chính xác.

8.3 Hướng phát triển

- Sử dụng các phương pháp khởi tạo centroid tốt hơn: Áp dụng các phương pháp khởi tạo centroid như K-Means++, giúp khởi tạo các centroid ban đầu sao cho hiệu quả hơn, từ đó giúp giảm thiểu khả năng bị dính vào các cực trị địa phương.
- Kết hợp với các phương pháp phân nhóm khác: Để vượt qua hạn chế về dữ liệu không đồng đều và hình dạng cụm phức tạp, có thể kết hợp K-Means với các thuật toán phân nhóm khác như DBSCAN hoặc Gaussian Mixture Models (GMM). Những thuật toán này có thể xử lý tốt hơn với các cụm có hình dạng không cầu và kích thước không đồng đều.

CHƯƠNG 9. TÀI LIỆU THAM KHẢO

[1] "statology," [Online]. Available: https://www.statology.org/label-encoding-vs-one-hot-encoding/.

[2] "geeksforgeeks," [Online]. Available: https://www.geeksforgeeks.org/interquartile-range-and-quartile-deviation-using-numpy-and-scipy/.

[3] "khanacademy," [Online]. Available: https://www.khanacademy.org/math/statistics-probability/modeling-distributions-of-data/z-scores/a/z-scores-review.