Phân khúc khách hàng bằng RFM và Kmean

Từ những khách hàng tiêu dùng lớn cho đến những khách hàng rời bỏ doanh nghiệp, tất cả những khách hàng đều có nhu cầu và mong muốn đa dạng. Doanh nghiệp muốn khách hàng chi tiêu nhiều hơn từ những chiến dịch tiếp thị chương trình, sản phẩm mới tới khách hàng theo những cách khác nhau. Tuy nhiên, câu hỏi đặt ra là làm thế nào để đưa ra được các chiến dịch tiếp thị phù hợp với những nhóm khách hàng đang có nhu cầu để từ đó tăng tỷ lệ phản hồi từ khách hàng và từ đó tăng doanh số bán hàng. Bài toán đặt ra là làm thế nào để có thể phân khúc khách hàng một cách tương đối chính xác dựa trên "hành vi giao dịch lịch sử" của khách hàng, thuật toán RFM sẽ giúp chúng ta giải quyết vấn đề này một cách nhanh chóng và hiệu quả.

1. RFM (Recently – Frequently – Montery Value)

Để phân khúc khách hàng, có một vài metrics thường sử dụng như là:

+ Khách hàng mua hàng lần cuối khi nào? (R)

+ Tần suất khách hàng mua hàng nhiều hay ít (F)

+ Khách hàng chi bao nhiêu tiền để mua sản phẩm của công ty (M)

Recently: Để tính ngày mua hàng gần nhất: ta lấy ngày hiện tại trừ đi ngày mua hàng gần nhất của khách hàng, ta sẽ được một giá trị gọi là khoảng cách từ ngày mua hàng gần nhất đến hiện tại của khách hàng. Số này càng nhỏ thì khách hàng càng mua hàng gần hơn

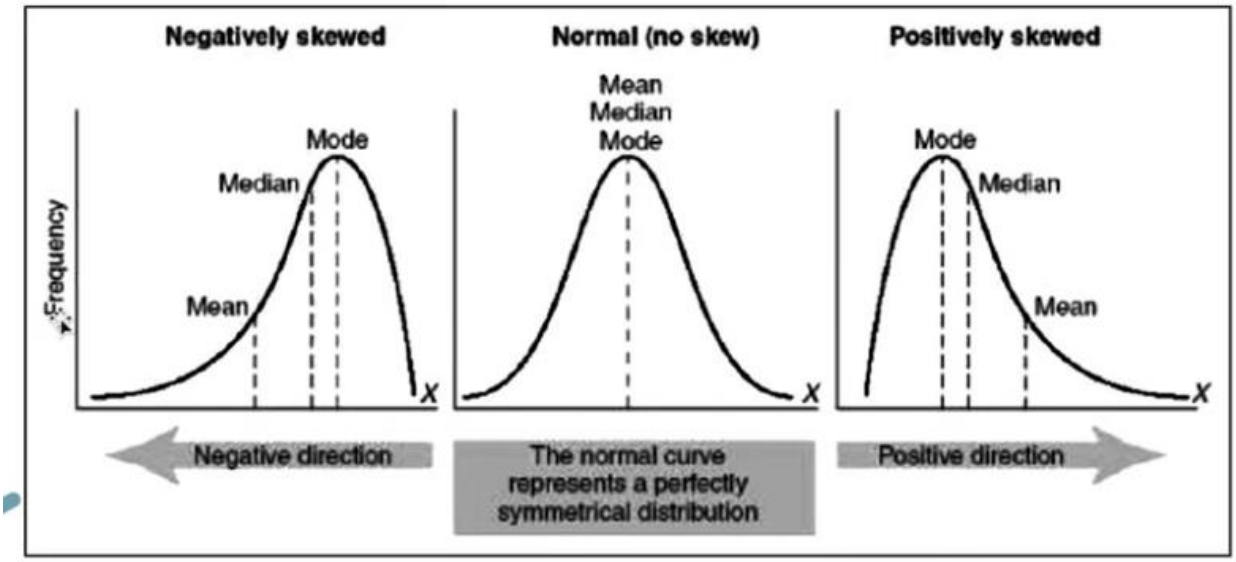
Frequently: Đếm số lượt mua hàng của khách hàng trong khoảng thời gian ta trích xuất dữ liệu.

Montery Value: tính tổng tiền tất cả mỗi khách hàng. (nhân đơn giá với số lượng và tính tổng)

Từ bảng hóa đơn lịch sử mua sắp khách hàng, ta tính các thông số R, F, M này cho mỗi khách hàng. Coi đó là các Feature để phân nhóm khách hàng.

2. Chỉ số Skewness

Chỉ số Skewness dùng để đánh giá dữ liệu có phân phối chuẩn hay không hay bị lệch trái, lệch phải.



Lý tưởng: chỉ số Skewness = 0. Dùng các biện pháp transform để đưa chỉ số Sknewness về càng gần 0 càng tốt.

Một số phương pháp:

+ log transformation

+ square root transformation

+ box-cox transformation

+ cube root transformation (căn bậc 3)

3. Lập trình

1. Import một số thư viện cần thiết

```
[2]  import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
```

2. Đọc dữ liệu file excel:

```
[3] df = pd.read_excel("/content/gdrive/MyDrive/Machine_Learning/Customer_Segmentation/data.xlsx")
```

```
[4] df.head(10)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 08:34:00	1.69	13047.0	United Kingdom

```
[5] # Đếm thử số lượng dòng
    df.count()
```

```
InvoiceNo      541909
StockCode      541909
Description    540455
Quantity       541909
InvoiceDate    541909
UnitPrice      541909
CustomerID     406829
Country        541909
dtype: int64
```

Làm bài toán Sample với khoảng 10000 dòng:

```
[6] # Loại bỏ giá trị nan, lấy những giá trị mà CustomerID không bị nan
    df_not_nan = df[df['CustomerID'].notna()]
    # Sau đó lấy 10000 dòng dữ liệu để làm sample
    df_not_nan = df_not_nan.sample(10000, random_state = 42)
    df_not_nan.head(10)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
47912	540456	48185	DOORMAT FAIRY CAKE	2	2011-01-07 12:14:00	7.95	13534.0	United Kingdom
342630	566891	23013	GLASS APOTHECARY BOTTLE TONIC	4	2011-09-15 13:51:00	3.95	14894.0	United Kingdom
288183	C562139	21313	GLASS HEART T-LIGHT HOLDER	-4	2011-08-03 10:10:00	0.85	12921.0	United Kingdom
325368	565438	22382	LUNCH BAG SPACEBOY DESIGN	4	2011-09-04 13:56:00	1.65	17229.0	United Kingdom
331450	566016	21212	PACK OF 72 RETROSPOT CAKE CASES	24	2011-09-08 12:20:00	0.55	15144.0	United Kingdom
234751	557598	22558	CLOTHES PEGS RETROSPOT PACK 24	2	2011-06-21 11:33:00	1.65	14667.0	United Kingdom
283269	561704	21533	RETROSPOT LARGE MILK JUG	1	2011-07-29 11:07:00	4.95	14606.0	United Kingdom
532936	581001	22726	ALARM CLOCK BAKELIKE GREEN	24	2011-12-07 08:07:00	3.75	12583.0	France
78561	542890	84865	NEW BAROQUE BLACK PHOTO ALBUM	2	2011-02-01 13:25:00	8.50	15311.0	United Kingdom
424855	573287	21314	SMALL GLASS HEART TRINKET POT	8	2011-10-28 14:42:00	2.10	14194.0	United Kingdom

3. Tính toán RFM

Tính R: Recently: Khách hàng nào mua hàng gần nhất thì R nhỏ, mua hàng càng lâu thì R lớn

```
# Tính toán RFM

# Tính R = Recently: Khách hàng nào mua hàng gần nhất thì R nhỏ, mua hàng càng lâu thì R lớn

df_not_nan['InvoiceDate'].describe()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: FutureWarning: Treating datetime data as categorical rather than num
  """
count                   10000
unique                   6430
top       2011-12-05 17:17:00
freq                       16
first     2010-12-01 09:34:00
last      2011-12-09 12:16:00
Name: InvoiceDate, dtype: object
```

Xem mô tả của cột InvoideDate, ta thấy kiểu dữ liệu của nó đang ở dạng Object không phải dạng date.

```python
# Tính toán RFM

######## Tính R = Recently: Khách hàng nào mua hàng gần nhất thì R nhỏ, mua hàng càng lâu thì R lớn

# Chuyển từ string sang date
df_not_nan['InvoiceDate'] = pd.to_datetime(df_not_nan['InvoiceDate'], format='%Y-%m-%d %H:%M:%S')

# Lấy ngày lớn nhất trong InvoideDate + 1

import datetime
current_date = max(df_not_nan['InvoiceDate']) + datetime.timedelta(days=1)

######## Tính M = Moneytary Value
df_not_nan['TotalPay'] = df_not_nan['Quantity'] * df_not_nan['UnitPrice']

# Group by CustomerID để tính RFM
df_customers = df_not_nan.groupby(['CustomerID']).agg(
    {'InvoiceDate': lambda x: (current_date - x.max()).days,
     'InvoiceNo' : 'count',
     'TotalPay' : 'sum'
     }
)
```

Bây giờ ta có một data frame customer với cột InvoiceDate thể hiện khách hàng mua hàng xa hay gần với thời điểm hiện tại, InvoiceNo thể hiện số lượng đơn hàng khách hàng đã mua, TotalPay là tổng tiền khách hàng đã chi để mua hàng.

```python
df_customers.head()
```

CustomerID	InvoiceDate	InvoiceNo	TotalPay
12347.0	40	5	133.20
12348.0	249	2	120.88
12349.0	19	2	312.75
12352.0	73	5	80.85
12354.0	233	2	33.30

Rename các cột lại cho đúng ý nghĩa:

```python
[14] df_customers.rename(columns={'InvoiceDate':'Recently', 'InvoiceNo':'Frequently', 'TotalPay':'MonetaryValue'}, inplace = True)
```

```
df_customers.rename(columns={'InvoiceDate':'Recently', 'InvoiceNo':'Frequently', 'TotalPay':'MonetaryValue'})
```

CustomerID	Recently	Frequently	MonetaryValue
12347.0	40	5	133.20
12348.0	249	2	120.88
12349.0	19	2	312.75
12352.0	73	5	80.85
12354.0	233	2	33.30
...	...	...	...
18265.0	73	2	39.60
18272.0	3	11	206.17
18274.0	18	2	-4.65
18283.0	4	21	78.03
18287.0	43	4	68.28

2690 rows × 3 columns

Bây giờ ta có thể dùng để đưa vào model KNN để phân cụm khách hàng, tuy nhiên hiệu quả sẽ không cao vì:

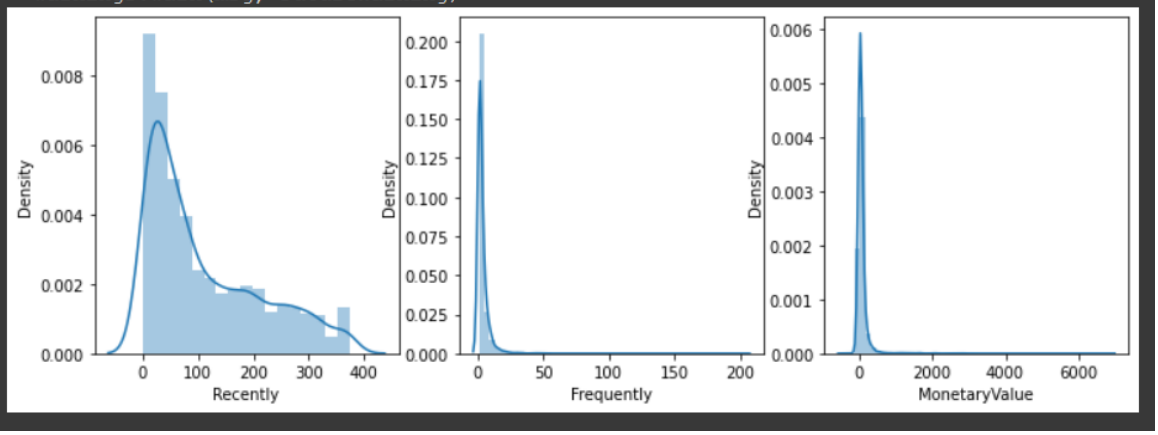
+ Các dữ liệu đang chưa được scale về một khoảng giống nhau.

+ Chưa biết phân phối dữ liệu.

4. Vẽ phân phối để kiểm tra dữ liệu và áp dụng các biện pháp Transform

```
[16] # Vẽ phân phối để kiểm tra
     fig, ax = plt.subplots(1, 3, figsize=(12, 4)) # tạo 1 dòng, 3 cột

     sns.distplot(df_customers['Recently'], ax = ax[0]) # vẽ vào cột đầu tiên
     sns.distplot(df_customers['Frequently'], ax = ax[1]) # vẽ vào cột thứ hai
     sns.distplot(df_customers['MonetaryValue'], ax = ax[2]) # vẽ vào cột thứ ba
     plt.show();
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `dis
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `dis
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `dis
  warnings.warn(msg, FutureWarning)
```



Ta thấy dữ liệu chưa phân bố chuẩn. Ta phải áp dụng các biện pháp Transform.

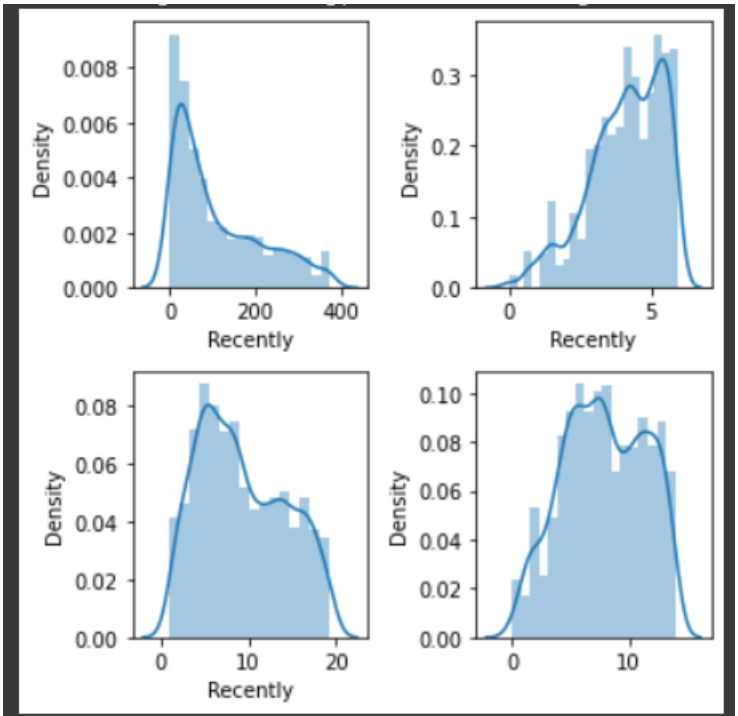
Hàm kiểm tra xem áp dụng phương pháp nào để ra được phân phối chuẩn:

```
from scipy import stats
def analyze_skewness(x):
    fig, ax = plt.subplots(2, 2, figsize=(5,5))
    sns.distplot(df_customers[x], ax=ax[0,0])
    sns.distplot(np.log(df_customers[x]), ax=ax[0,1])
    sns.distplot(np.sqrt(df_customers[x]), ax=ax[1,0])
    sns.distplot(stats.boxcox(df_customers[x])[0], ax=ax[1,1])
    plt.tight_layout()
    plt.show()

    print(df_customers[x].skew().round(2))
    print(np.log(df_customers[x]).skew().round(2))
    print(np.sqrt(df_customers[x]).skew().round(2))
    print(pd.Series(stats.boxcox(df_customers[x])[0]).skew().round(2))
```

Với cột Recently: Xem các phân bố khi áp dụng các biện pháp Transformation bằng lệnh:

```
analyze_skewness('Recently')
```

Kết quả: ta thấy box-cox transformation có chỉ số skewness là -0,1 gần với 0 nhất nên nó là tốt nhất.

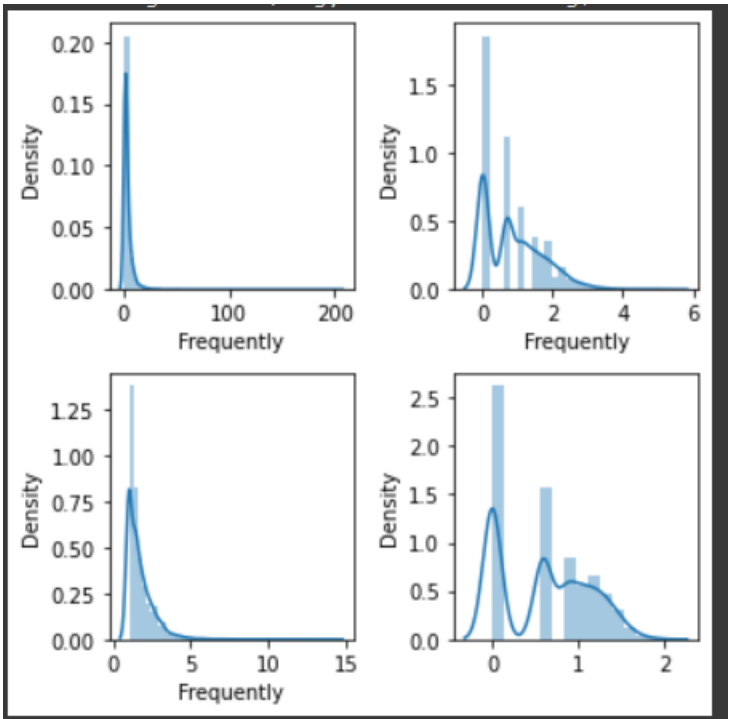



```
0.93
-0.72
0.32
-0.1
```

Với cột Frequently:

```
analyze_skewness('Frequently')
```

Kết quả: Ta thấy box-cox Transformation vẫn cho kết quả tốt nhất (skewness = 0,16)




```
14.77
0.85
3.67
0.16
```

Với cột MonetaryValue: vì có giá trị âm nên ta không dùng được các biện pháp Transform như box-cox, sqr root hay log mà ta phải dùng cube root (căn bậc 3).

Tiến hàng Transform dữ liệu:

```
[25] # Tiến hàng Transform dữ liệu

     # Tạo 1 dữ liệu riêng để Transform mà không làm thay đổi dữ liệu gốc
     df_customers_T  = pd.DataFrame()

     df_customers_T['Recently'] = stats.boxcox(df_customers['Recently'])[0]
     df_customers_T['Frequently'] = stats.boxcox(df_customers['Frequently'])[0]
     df_customers_T['MonetaryValue'] = pd.Series(np.cbrt(df_customers['Recently'])).values
```

```
df_customers_T.head()
```

	Recently	Frequently	MonetaryValue
0	6.152222	1.127547	3.419952
1	12.180045	0.591193	6.291195
2	4.402867	0.591193	2.668402
3	7.832068	1.127547	4.179339
4	11.907953	0.591193	6.153449

5. Scale dữ liệu

```
[28] # Tiến hành scale dữ liệu
     from sklearn.preprocessing import StandardScaler
     scaler = StandardScaler()
     scaler.fit(df_customers_T)

     df_customers_T = scaler.transform(df_customers_T)
```

```
pd.DataFrame(df_customers_T).head()
```

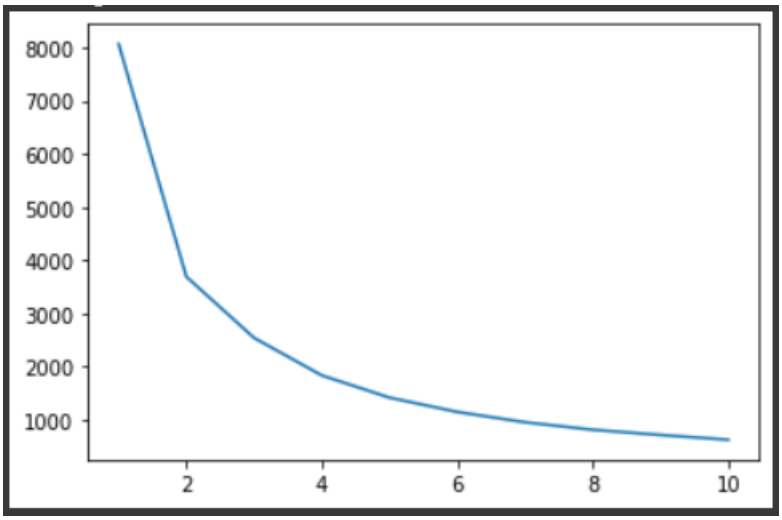
	0	1	2
0	-0.493794	1.012426	-0.533456
1	1.232949	-0.017412	1.262754
2	-0.994917	-0.017412	-1.003616
3	-0.012582	1.012426	-0.058394
4	1.155005	-0.017412	1.176583

6. Đưa dữ liệu vào K-means để phân khúc khách hàng

➔ *Tiến hành chọn số cụm bằng phương pháp Elbow:*

```
[31] from sklearn.cluster import KMeans

# Chọn K cluster bằng thuật toán Elbow

k_rng = range(1, 11) #10 gia tri k
sse = []

for k in k_rng:
  km = KMeans(n_clusters=k)
  km.fit(df_customers_T)
  sse.append(km.inertia_)

print(sse)
plt.plot(k_rng, sse)
```

Kết quả:



Chọn k sao cho khi k tăng lên 1 thì giá trị SSE không giảm đi quá nhiều. Vì vậy dựa vào đồ thị trên ta chọn k = 3 là phù hợp.

Áp dụng thuật toán phân cụm K-means:

```
[42] # Áp dụng model K-Means vào bài toán phân cụm khách hàng
     km = KMeans(n_clusters=3)
     y_predicted = km.fit_predict(df_customers_T)
     y_predicted
```

```
array([0, 1, 0, ..., 0, 0, 0], dtype=int32)
```

```
df_customers['Cluster'] = y_predicted
df_customers.head()
```

CustomerID	Recently	Frequently	MonetaryValue	Cluster
12347.0	40	5	133.20	0
12348.0	249	2	120.88	1
12349.0	19	2	312.75	0
12352.0	73	5	80.85	0
12354.0	233	2	33.30	1

Đặc trưng của từng nhóm khách hàng:

```
[52] # Đặc trưng của các cụm khách hàng
     df_customers.groupby('Cluster').agg(
         {
             'Recently': 'mean',
             'Frequently': 'mean',
             'MonetaryValue': 'mean'
         }
     ).round(2)
```

Cluster	Recently	Frequently	MonetaryValue
0	29.88	7.40	145.45
1	223.12	1.99	40.10
2	56.46	1.40	27.99

Nhóm khách hàng 0: Mua gần đây, số lần mua khá nhiều và tổng chi phí họ mua rất lớn → Một số khách hàng tiềm năng gần đây, cần có những biện pháp tri ân khách hàng.

Nhóm khách hàng 1: Mua hàng rất lâu rồi, số lần mua khá ít và tổng chi phí họ mua ở mức trung bình → đây có thể là nhóm khách hàng mới sử dụng dịch vụ nên e ngại mua số lượng lớn và họ chỉ dừng ở mức mua để dùng thử.

Nhóm khách hàng 2: Mua hàng khá gần, số lần mua ít và chi ít tiền.