

# Project 2: Wholesale Customers

## Business Objective/Problem

- Chiến lược kinh doanh các sản phẩm của công ty đến đối tượng khách hàng bán buôn có hiệu quả chưa cao => Cần tìm ra giải pháp giúp cải thiện hiệu quả quảng bá, từ đó giúp tăng doanh thu bán hàng, cải thiện mức độ hài lòng của khách hàng.

## Triển khai dự án

### Bước 1: Business Understanding

- Dựa vào mô tả nói trên (hoặc sau khi đặt ra các câu hỏi cụ thể cho các đối tượng có liên quan) => xác định được vấn đề: Hiện tại: Công ty có nhiều khách hàng bán buôn nhưng chưa nhóm khách hàng vào các nhóm khác nhau để tiện theo dõi, đưa ra những chiến lược phù hợp.
- => Mục tiêu/ Vấn đề: Xây dựng mô hình phân nhóm khách hàng.
- Note: Sử dụng thuật toán phân nhóm trên tập dữ liệu Khách hàng bán buôn là để hiểu hành vi của từng khách hàng. Điều này sẽ cho phép chúng ta nhóm những khách hàng có hành vi tương tự thành một cụm. Hành vi của khách hàng sẽ được xác định bởi số tiền họ đã chi tiêu cho từng loại sản phẩm, cũng như kênh và khu vực nơi họ mua sản phẩm.

### Bước 2: Data Understanding/ Acquire

- Toàn bộ dữ liệu được đỗ ra và lưu trữ trong tập tin Wholesal\_customers\_data.csv với 440 record.
- Mô tả dữ liệu: <http://archive.ics.uci.edu/ml/datasets/Wholesale+customers> (<http://archive.ics.uci.edu/ml/datasets/Wholesale+customers>)

```
In [ ]: #!pip install pandas-profiling==2.7.1
```

```
In [ ]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
import pandas_profiling as pp

from scipy.stats import chi2_contingency
from scipy.stats import chi2

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```
In [ ]: import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

```
In [ ]: pd.options.display.float_format = '{:.2f}'.format
```

```
In [ ]: # from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)

# %cd '/content/gdrive/My Drive/HTDS/Topic_4/demo/'
```

```
In [ ]: data = pd.read_csv("Wholesal_customers_data.csv")
```

```
In [ ]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Channel          440 non-null    int64  
 1   Region           440 non-null    int64  
 2   Fresh            440 non-null    int64  
 3   Milk             440 non-null    int64  
 4   Grocery          440 non-null    int64  
 5   Frozen           440 non-null    int64  
 6   Detergents_Paper 440 non-null    int64  
 7   Delicassen       440 non-null    int64  
dtypes: int64(8)
memory usage: 27.6 KB
```

```
In [ ]: data.head()
```

```
Out[7]:   Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicassen
```

0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

### Bước 3: Data preparation/ Prepare

- Data Consolidation: hợp nhất dữ liệu từ các nguồn, lựa chọn các dữ liệu phù hợp: chỉ có một nguồn dữ liệu là file trên

#### First EDA

```
In [ ]: pp.ProfileReport(data)
```

```
Summarize dataset: 0% | 0/20 [00:00<?, ?it/s]  
Generate report structure: 0% | 0/1 [00:00<?, ?it/s]  
Render HTML: 0% | 0/1 [00:00<?, ?it/s]
```

```
Out[8]:
```

```
In [ ]: data.describe()
```

```
Out[9]:   Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicassen
```

count	440.00	440.00	440.00	440.00	440.00	440.00	440.00	440.00
mean	1.32	2.54	12000.30	5796.27	7951.28	3071.93	2881.49	1524.87
std	0.47	0.77	12647.33	7380.38	9503.16	4854.67	4767.85	2820.11
min	1.00	1.00	3.00	55.00	3.00	25.00	3.00	3.00
25%	1.00	2.00	3127.75	1533.00	2153.00	742.25	256.75	408.25
50%	1.00	3.00	8504.00	3627.00	4755.50	1526.00	816.50	965.50
75%	2.00	3.00	16933.75	7190.25	10655.75	3554.25	3922.00	1820.25
max	2.00	3.00	112151.00	73498.00	92780.00	60869.00	40827.00	47943.00

- Data Cleaning: kiểm tra xem có dữ liệu không liên quan, dữ liệu null, dữ liệu outlier, hay không? => Nếu có thì xử lý.

\* Dataset có các cột đều cần thiết để giải quyết bài toán => dữ liệu đã liên quan.

\* Kiểm tra dữ liệu null:

```
In [ ]: # kiểm tra dữ liệu null  
print(data.isnull().sum())  
# => Không có dữ liệu null
```

```
Channel          0  
Region          0  
Fresh            0  
Milk             0  
Grocery          0  
Frozen           0  
Detergents_Paper 0  
Delicassen       0  
dtype: int64
```

- Phân tích đơn biến: trực quan hóa, kiểm tra dữ liệu outlier

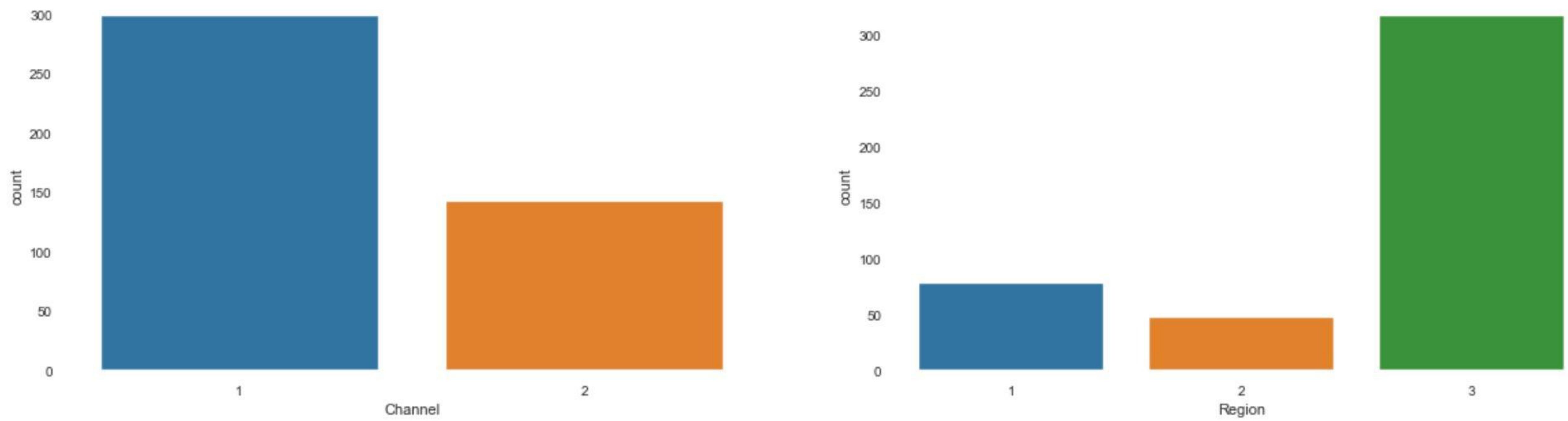
```
In [ ]: ### Biến phân loại  
data.groupby("Channel").size()
```

```
Out[11]: Channel  
1    298  
2    142  
dtype: int64
```

```
In [ ]: data.groupby("Region").size()
```

```
Out[12]: Region  
1     77  
2     47  
3    316  
dtype: int64
```

```
In [ ]: plt.figure(figsize=(20,5))
plt.subplot(1, 2, 1)
sns.countplot(x="Channel", data=data)
plt.subplot(1, 2, 2)
sns.countplot(x="Region", data=data)
plt.show()
```



```
In [ ]: # chi-squared test with similar proportions
```

```
In [ ]: # contingency table: Ho: Pclass and Sex independent
table = pd.crosstab(data['Channel'], data['Region'])
```

```
Out[15]:
```

Region	1	2	3
Channel			
1	59	28	211
2	18	19	105

```
In [ ]: stat, p, dof, expected = chi2_contingency(table)
print('dof=%d' % dof)
print(expected)
```

dof=2  
[[ 52.15 31.83181818 214.01818182]  
 [ 24.85 15.16818182 101.98181818]]

```
In [ ]: # interpret test-statistic
prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, stat))
```

probability=0.950, critical=5.991, stat=4.349

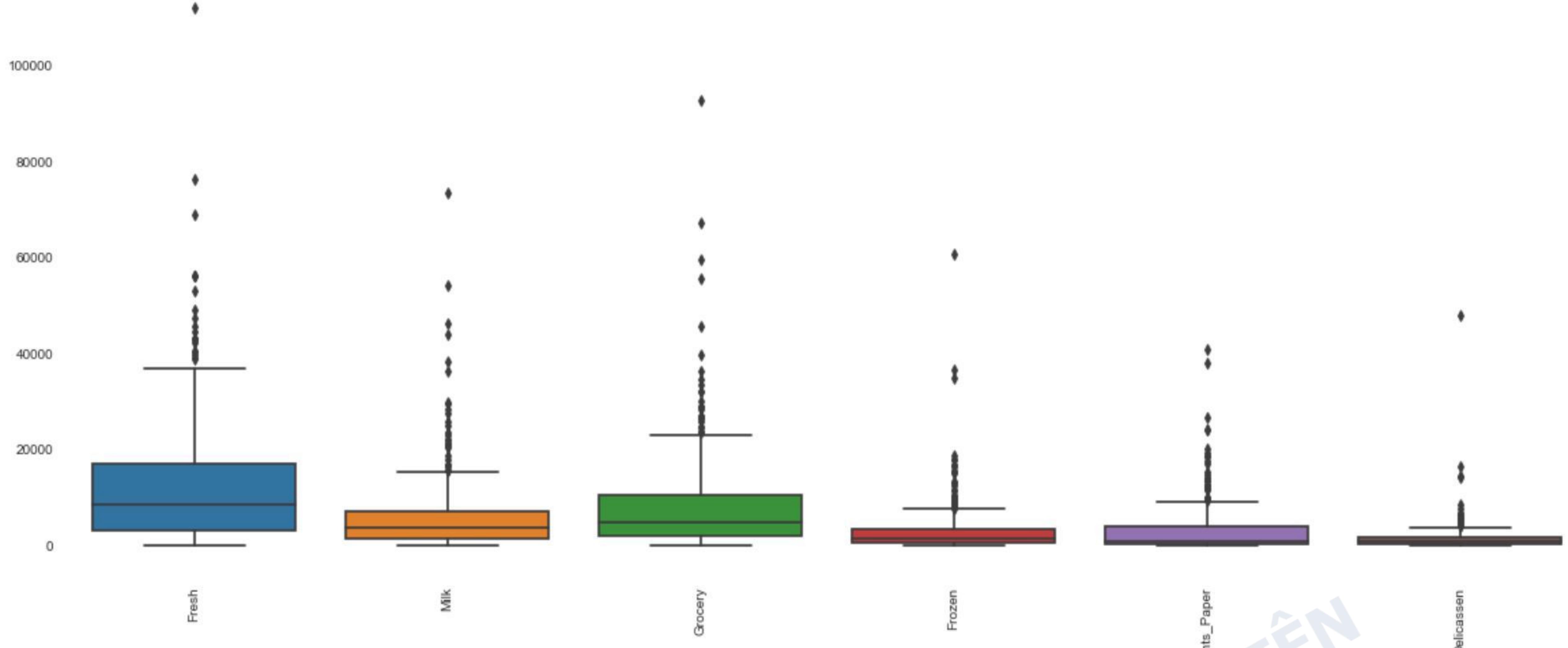
```
In [ ]: # interpret p-value
alpha = 1.0 - prob
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

significance=0.050, p=0.114  
Independent (fail to reject H0)

```
In [ ]: # Không có mối quan hệ gì giữa Channel và Region
```

```
In [ ]: ### Biến Liên tục
```

```
In [ ]: fig, ax = plt.subplots(figsize=(20,8))
sns.boxplot(data = data[["Fresh", "Milk", "Grocery", "Frozen", "Detergents_Paper", "Delicassen"]],  
            ax=ax)
plt.xticks(rotation=90)
plt.show()
```

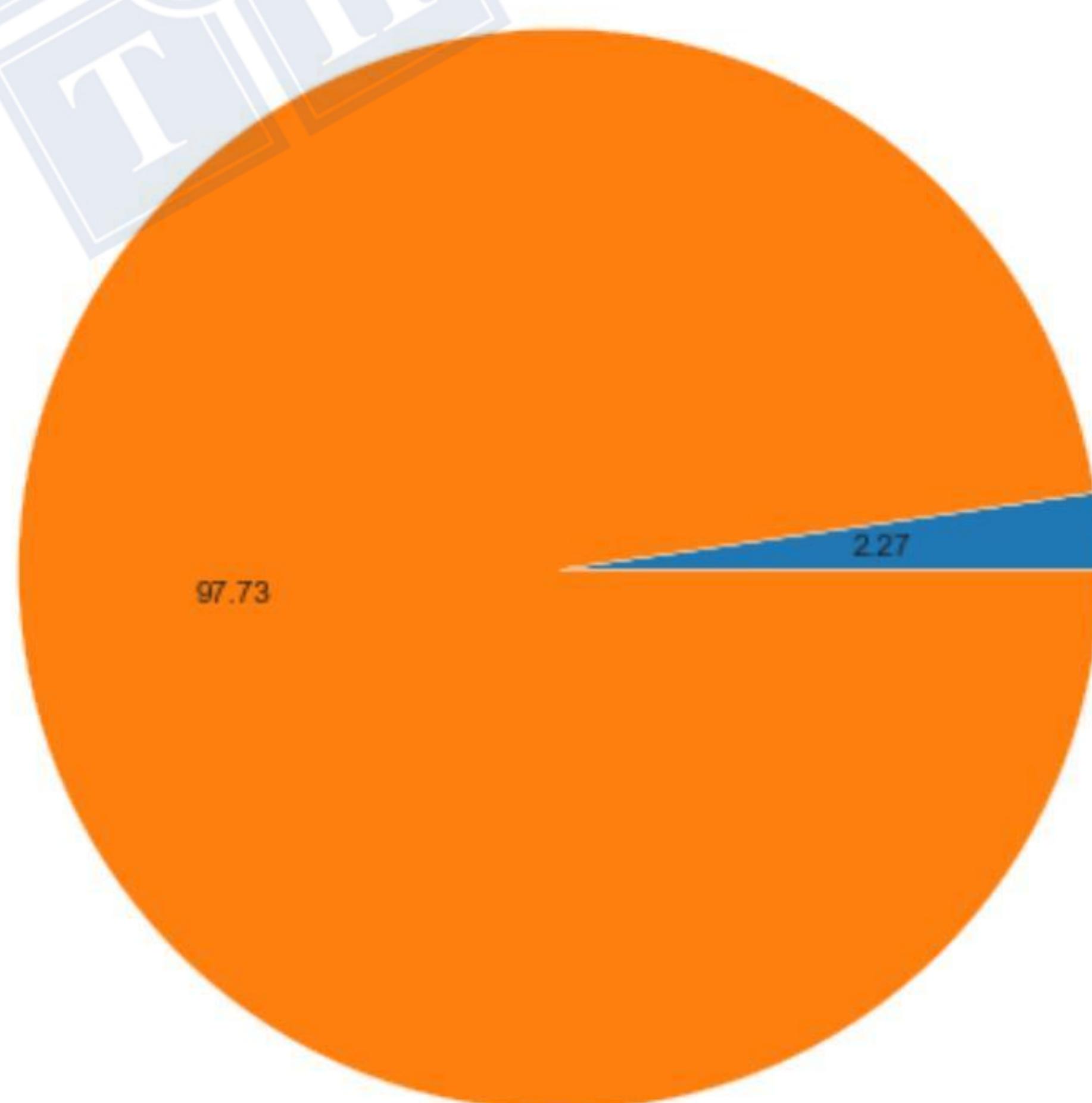


```
In [ ]: # Có outlier ở các cột dữ liệu liên quan đến số tiền
```

```
In [ ]: # Count outlier samples
outliers = {}
for i in range(data.shape[1]):
    min_t = data[data.columns[i]].mean() - (3 * data[data.columns[i]].std())
    max_t = data[data.columns[i]].mean() + (3 * data[data.columns[i]].std())
    count = 0
    for j in data[data.columns[i]]:
        if j < min_t or j > max_t:
            count += 1
    outliers[data.columns[i]] = [count,data.shape[0]-count]
print(outliers)
```

{'Channel': [0, 440], 'Region': [0, 440], 'Fresh': [7, 433], 'Milk': [9, 431], 'Grocery': [7, 433], 'Frozen': [6, 434], 'Detergents\_Paper': [10, 430], 'Delicassen': [4, 436]}

```
In [ ]: plt.figure(figsize=(8,8))
plt.pie(outliers["Detergents_Paper"], autopct=".2f")
plt.show()
```



```
In [ ]: # Số lượng outlier rất ít, nhiều nhất là cột Detergents_Paper với 10 mẫu outlier (~2.3%)  
# Có thể không cần loại bỏ outlier???
```

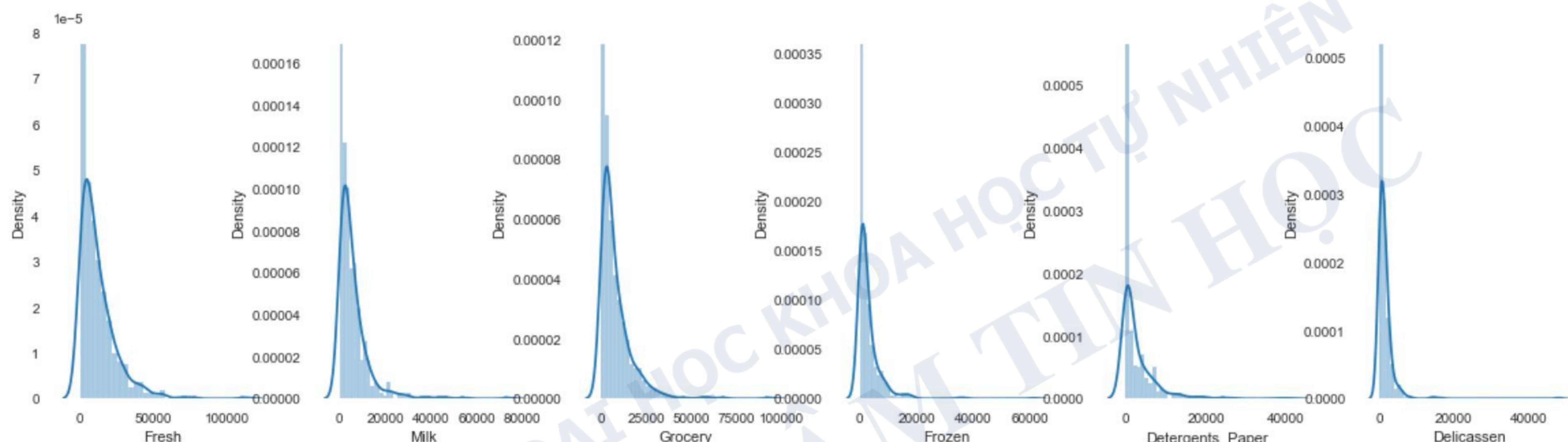
- Data Transformation: chuẩn hóa dữ liệu, bổ sung thuộc tính mới

- \* Customer Segmentation là một bài toán phân nhóm dựa trên sự tương tự về các thuộc tính và sẽ tính khoảng cách để biết mẫu này “gần” mẫu kia hay không bằng công thức tính khoảng cách Euclidean
- \* Đối với các thuật toán cần tính khoảng cách thì dữ liệu trên các cột phải đảm bảo tính công bằng khi tính toán. Các cột dữ liệu liên tục có cùng một thang đo. Nếu không cần chuẩn hóa dữ liệu.
- \* Muốn chuẩn hóa dữ liệu: cần kiểm tra phân phối của dữ liệu.

```
In [ ]: data.columns
```

```
Out[26]: Index(['Channel', 'Region', 'Fresh', 'Milk', 'Grocery', 'Frozen',
   'Detergents_Paper', 'Delicassen'],
  dtype='object')
```

```
In [ ]: plt.figure(figsize=(20,5))
plt.subplot(1, 6, 1)
sns.distplot(data.Fresh)
plt.subplot(1, 6, 2)
sns.distplot(data.Milk)
plt.subplot(1, 6, 3)
sns.distplot(data.Grocery)
plt.subplot(1, 6, 4)
sns.distplot(data.Frozen)
plt.subplot(1, 6, 5)
sns.distplot(data.Detergents_Paper)
plt.subplot(1, 6, 6)
sns.distplot(data.Delicassen)
plt.show()
```



```
In [ ]: data[['Fresh', 'Milk', 'Grocery', 'Frozen',
   'Detergents_Paper', 'Delicassen']].skew()
```

```
Out[28]: Fresh      2.56
Milk       4.05
Grocery    3.59
Frozen     5.91
Detergents_Paper  3.63
Delicassen  11.15
dtype: float64
```

```
In [ ]: data[['Fresh', 'Milk', 'Grocery', 'Frozen',
   'Detergents_Paper', 'Delicassen']].kurtosis()
```

```
Out[29]: Fresh      11.54
Milk       24.67
Grocery    20.91
Frozen     54.69
Detergents_Paper  19.01
Delicassen  170.69
dtype: float64
```

- Phân phối lệch phải
  - Có thể dùng log scale
- Và/ hoặc từ những kết quả trên ta thấy:
  - Dữ liệu không theo phân phối Gaussian
  - Dữ liệu có outlier => Dùng RobustScaler để chuẩn hóa (nếu có làm sau này)
- Hoặc có thể loại outlier rồi dùng MinMaxScaler
  
- Tuy nhiên, Dữ liệu trong bài này là cùng một thang đo (số tiền khách hàng đã mua). => Có thể dùng dữ liệu gốc. Sau khi làm xong có thể thử nghiệm thêm với dữ liệu scale.

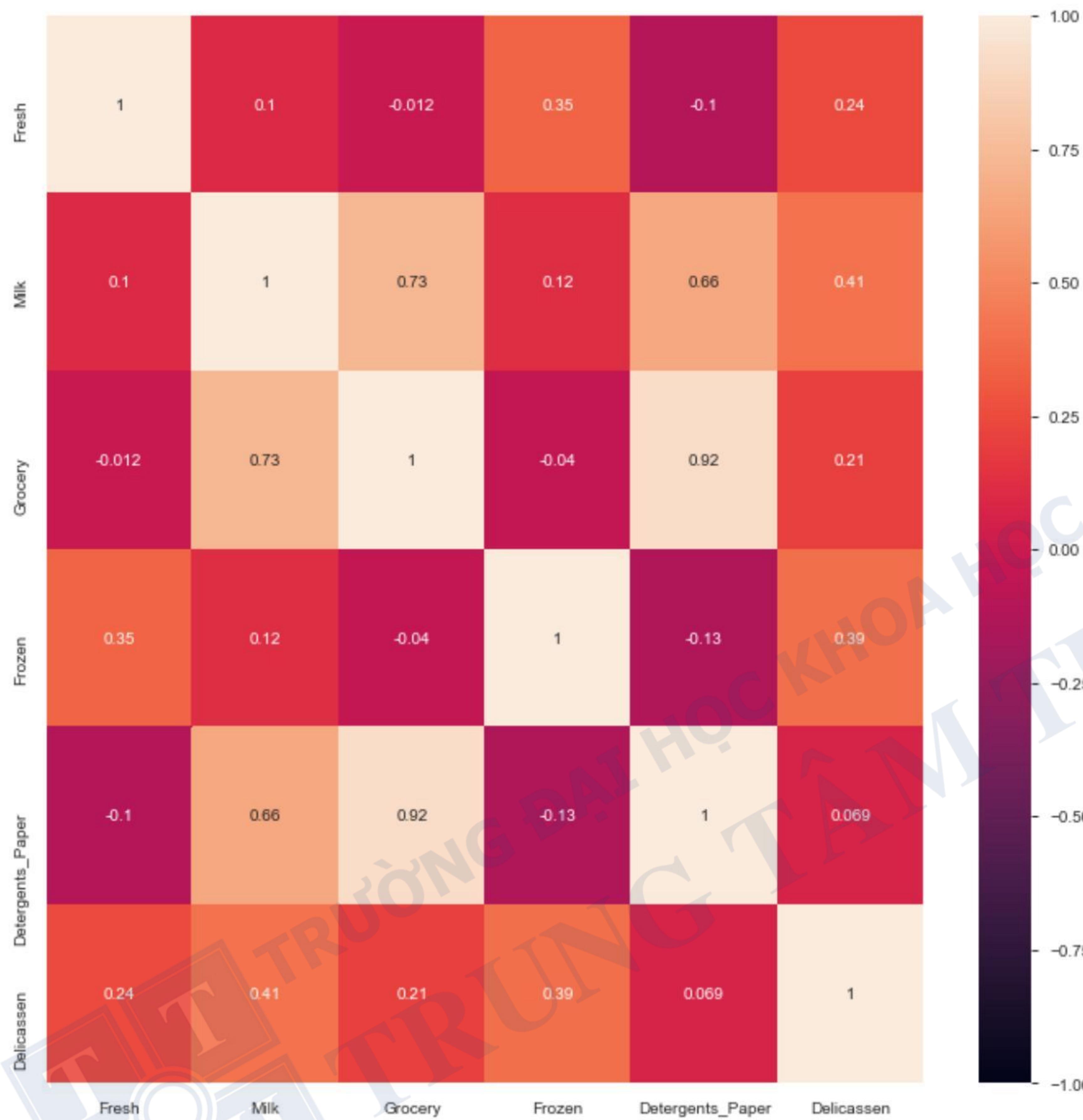
```
In [ ]: # Mối quan hệ giữa các biến liên tục
corr = data[["Fresh", "Milk", "Grocery", "Frozen", "Detergents_Paper", "Delicassen"]].corr()
```

```
In [ ]: corr
```

```
Out[31]:
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
Fresh	1.00	0.10	-0.01	0.35	-0.10	0.24
Milk	0.10	1.00	0.73	0.12	0.66	0.41
Grocery	-0.01	0.73	1.00	-0.04	0.92	0.21
Frozen	0.35	0.12	-0.04	1.00	-0.13	0.39
Detergents_Paper	-0.10	0.66	0.92	-0.13	1.00	0.07
Delicassen	0.24	0.41	0.21	0.39	0.07	1.00

```
In [ ]: fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(corr, vmin=-1, vmax=1, annot=True)
plt.show()
```



- Milk và Grocery có mối quan hệ khá cao với nhau (~0.73)
- Detergents\_Paper và Grocery có mối quan hệ rất cao với nhau (~0.92)
- Có thể gom 3 thuộc tính này để phân cụm trước

```
In [ ]: data.columns
```

```
Out[34]: Index(['Channel', 'Region', 'Fresh', 'Milk', 'Grocery', 'Frozen',
       'Detergents_Paper', 'Delicassen'],
       dtype='object')
```

```
In [ ]: data_sub = data[['Milk', 'Grocery', 'Detergents_Paper']]
```

```
In [ ]: data_sub.head()
```

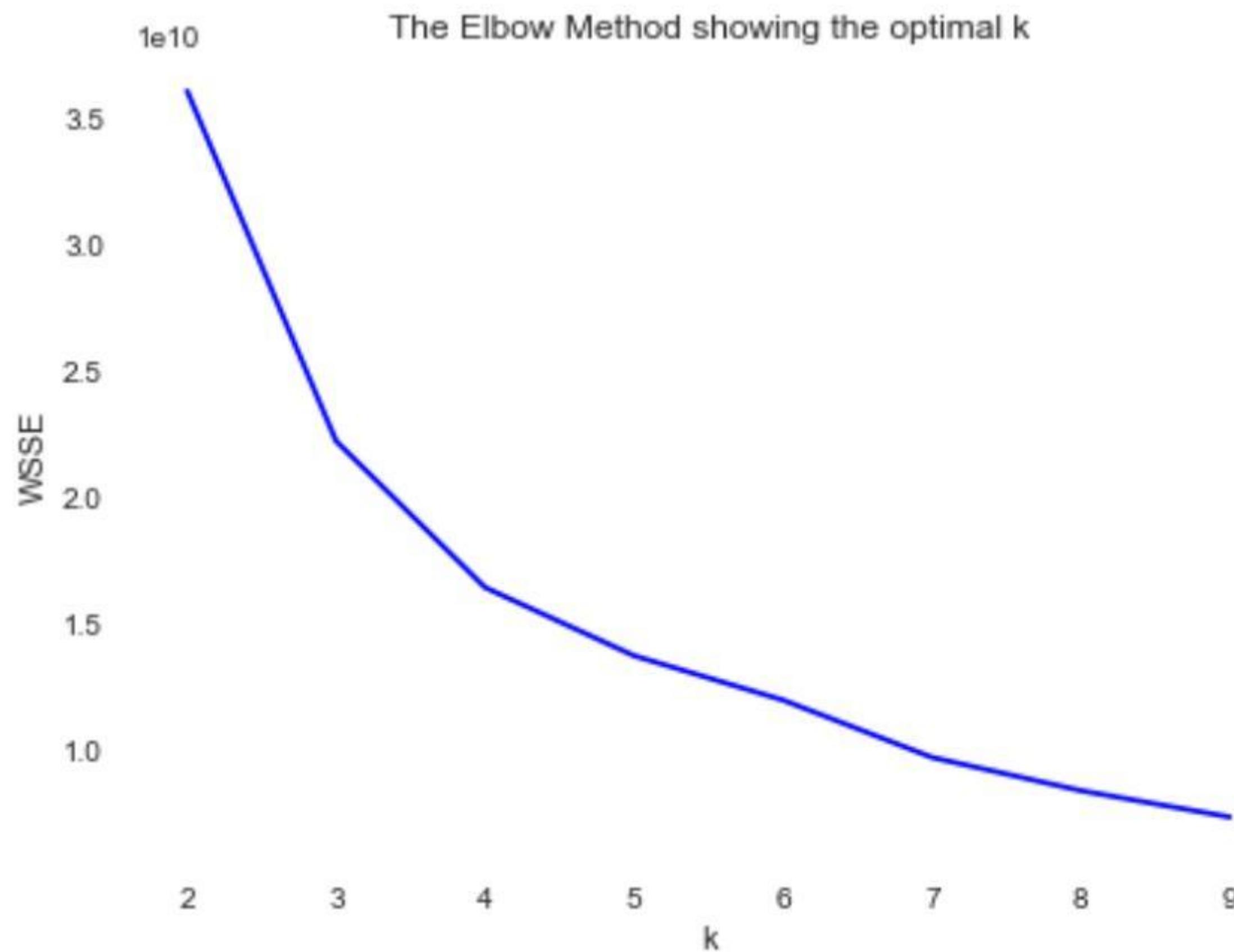
```
Out[36]:
```

	Milk	Grocery	Detergents_Paper
0	9656	7561	2674
1	9810	9568	3293
2	8808	7684	3516
3	1196	4221	507
4	5410	7198	1777

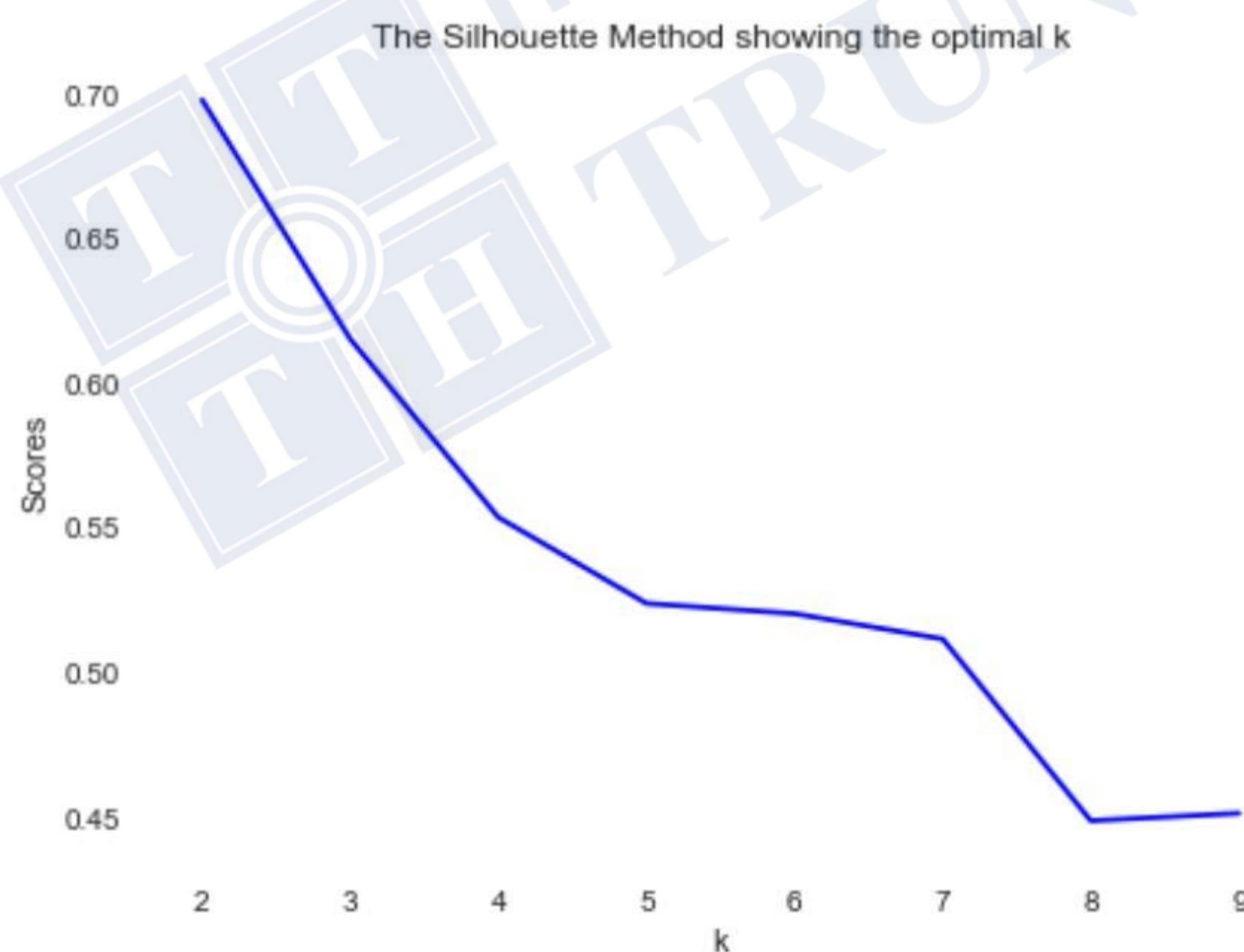
## Bước 4&5: Modeling & Evaluation/ Analyze & Report

```
In [ ]: # Select k
K = range(2,10)
wsses = []
silhouette = []
for k in K:
    est_kmeans = KMeans(n_clusters=k, random_state = 0)
    est_kmeans.fit(data_sub)
    pred_kmeans = est_kmeans.predict(data_sub)
    x = silhouette_score(data_sub, pred_kmeans, metric='euclidean')
    silhouette.append(x)
    wsses.append(est_kmeans.inertia_)
```

```
In [ ]: # Trực quan elbow
plt.figure(figsize=(7,5))
plt.plot(K, wsses, 'b')
plt.xlabel('k')
plt.ylabel('WSSE')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



```
In [ ]: # Trực quan silhouette_score
plt.figure(figsize=(7,5))
plt.plot(K, silhouette, 'b')
plt.xlabel('k')
plt.ylabel('Scores')
plt.title('The Silhouette Method showing the optimal k')
plt.show()
```



```
In [ ]: # Chọn k=4
kmeans = KMeans(n_clusters = 4, random_state = 0)
kmeans.fit(data_sub)
pred_kmeans = kmeans.predict(data_sub)
```

```
In [ ]: centroids = kmeans.cluster_centers_
labels = kmeans.labels_

print("Centroids in normal:")
print(centroids)

print("Labels:")
print(labels)
```

```
Centroids in normal:
[[ 8536.33043478 12381.89565217 4741.20869565]
 [ 2651.94117647 3283.92733564 755.98961938]
 [43460.6 61472.2 29974.2]
 [18869.83870968 26394.4516129 11427.93548387]]
Labels:
[0 0 0 1 1 1 0 1 0 0 0 1 0 1 0 1 1 1 3 0 1 1 1 3 1 0 1 1 1 1 1 0 1
 0 0 1 1 1 0 3 0 3 3 2 0 3 1 1 1 0 1 1 3 0 1 0 1 2 1 0 1 3 1 0 1 1 1 0 1 1
 0 1 1 3 1 1 1 0 0 1 1 2 2 1 1 1 1 1 3 1 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 3 1
 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 3 1 1
 1 1 1 1 1 1 0 0 1 0 0 0 1 1 3 0 0 0 1 1 1 1 0 3 1 0 1 0 1 1 1 1 0 3 0 3 1
 1 1 0 0 0 1 1 1 0 1 1 0 1 1 3 3 0 1 1 3 1 1 1 0 1 3 1 0 0 0 3 1 0 1 1 0
 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3 1 0 0 1 1 1 1
 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1
 1 1 0 1 1 3 0 0 3 0 3 1 1 0 1 1 0 1 0 1 1 3 1 1 1 0 1 0 1 1 1 1 1 3 1
 2 1 0 1 1 1 0 0 1 3 1 1 0 0 1 3 1 3 1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1
 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1
 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 0 1 0 0 1 1 1 1 1 0 1 0 1 1 3 1 1]
```

## Report

```
In [ ]: data_sub['Group'] = pd.Series(labels)
data_sub.head()
```

```
c:\program files\python36\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy)
    """Entry point for launching an IPython kernel.
```

```
Out[42]:
```

	Milk	Grocery	Detergents_Paper	Group
0	9656	7561	2674	0
1	9810	9568	3293	0
2	8808	7684	3516	0
3	1196	4221	507	1
4	5410	7198	1777	1

```
In [ ]: data_sub.groupby("Group").size()
```

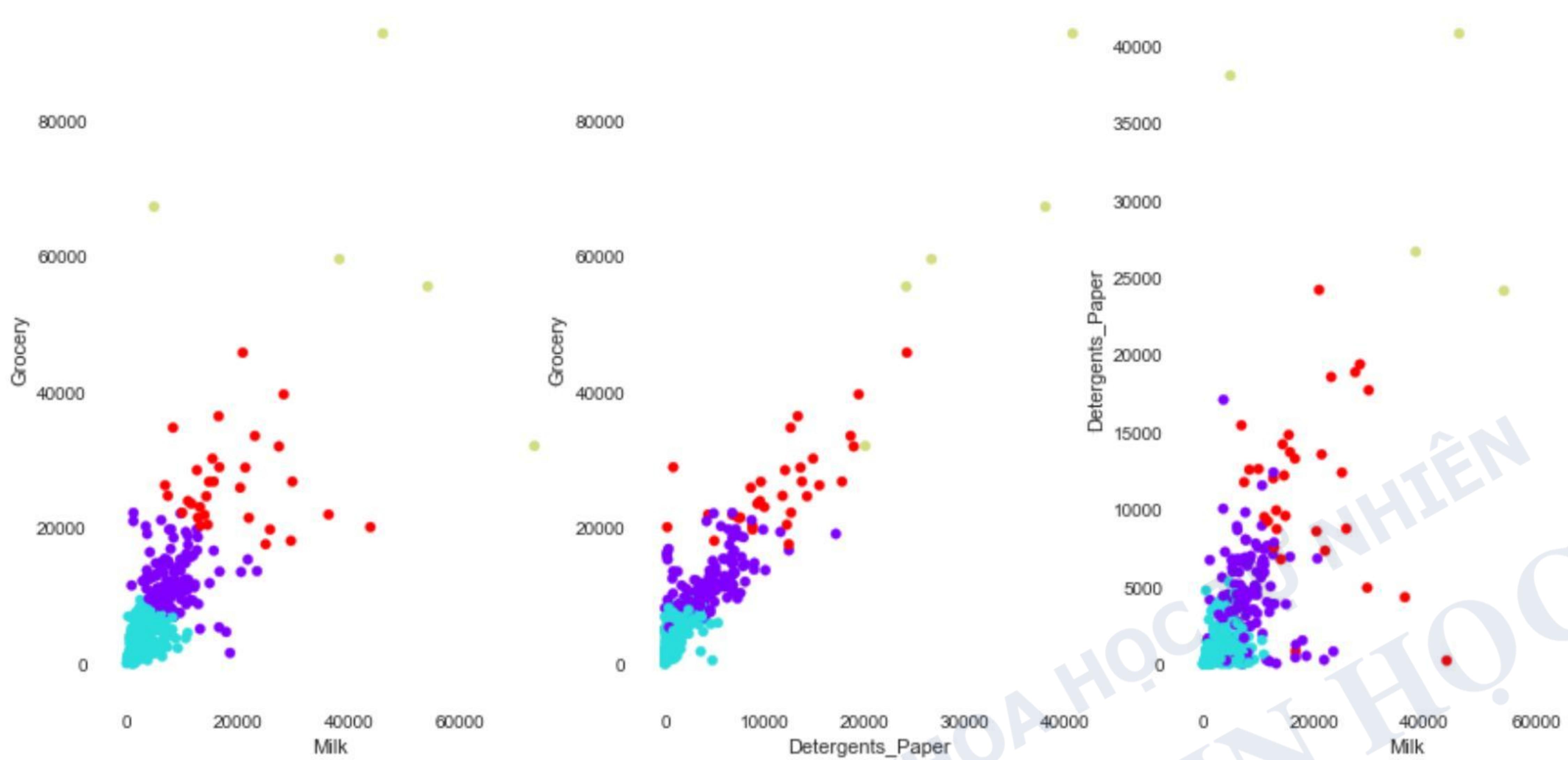
```
Out[43]:
```

Group	size
0	115
1	289
2	5
3	31

```
dtype: int64
```

```
In [ ]: # Một nhóm có số Lượng Là 5: Có đặc điểm gì đặc biệt cho nhóm này không?
```

```
In [ ]: plt.figure(figsize=(15,7))
plt.subplot(1, 3, 1)
plt.scatter(data.Milk, data.Grocery, c=pred_kmeans, s=30, cmap='rainbow')
plt.xlabel('Milk')
plt.ylabel('Grocery')
plt.subplot(1, 3, 2)
plt.scatter(data.Detergents_Paper, data.Grocery, c=pred_kmeans, s=30, cmap='rainbow')
plt.xlabel('Detergents_Paper')
plt.ylabel('Grocery')
plt.subplot(1, 3, 3)
plt.scatter(data.Milk, data.Detergents_Paper, c=pred_kmeans, s=30, cmap='rainbow')
plt.xlabel('Milk')
plt.ylabel('Detergents_Paper')
plt.show()
```



```
In [ ]: i = 0
print(data_sub.columns)
for cluster in centroids:
    print("*** Centroid Cluster: " + str(i))
    print(round(cluster[0],2), round(cluster[1],2), round(cluster[2],2))
    i = i+1
```

```
Index(['Milk', 'Grocery', 'Detergents_Paper', 'Group'], dtype='object')
*** Centroid Cluster: 0
8536.33 12381.9 12381.9
*** Centroid Cluster: 1
2651.94 3283.93 3283.93
*** Centroid Cluster: 2
43460.6 61472.2 61472.2
*** Centroid Cluster: 3
18869.84 26394.45 26394.45
```

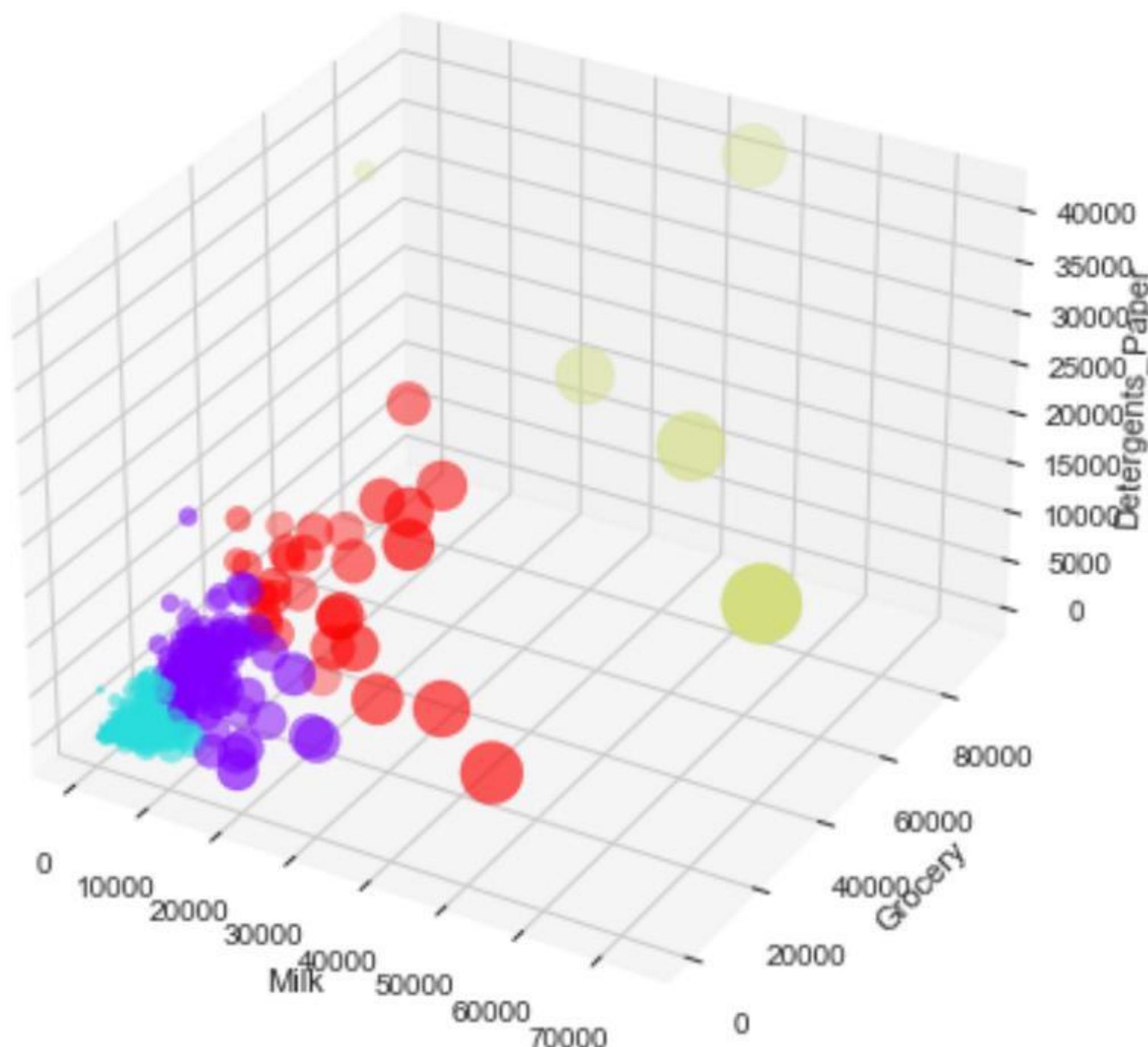
```
In [ ]: #%matplotlib notebook
```

```
In [ ]: fig = plt.figure(figsize=(7, 7))

ax = fig.add_subplot(111, projection='3d')

ax.scatter(data_sub.Milk, data_sub.Grocery, data_sub.Detergents_Paper,
           cmap='rainbow',
           s=data_sub.Milk/100,
           c=data_sub.Group
           )
ax.set_xlabel('Milk')
ax.set_ylabel('Grocery')
ax.set_zlabel('Detergents_Paper')

Out[48]: Text(0.5, 0, 'Detergents_Paper')
```



```
In [ ]: # Giải thích cụm:
for i in range(len(centroids)):
    print("----- Group ", i, "-----")
    data_filter = data_sub.loc[(data_sub['Group']==i)]
    print("Number of samples:", data_filter.shape[0])
    print(data_filter.describe().loc[['count', 'mean', 'min', 'max']])

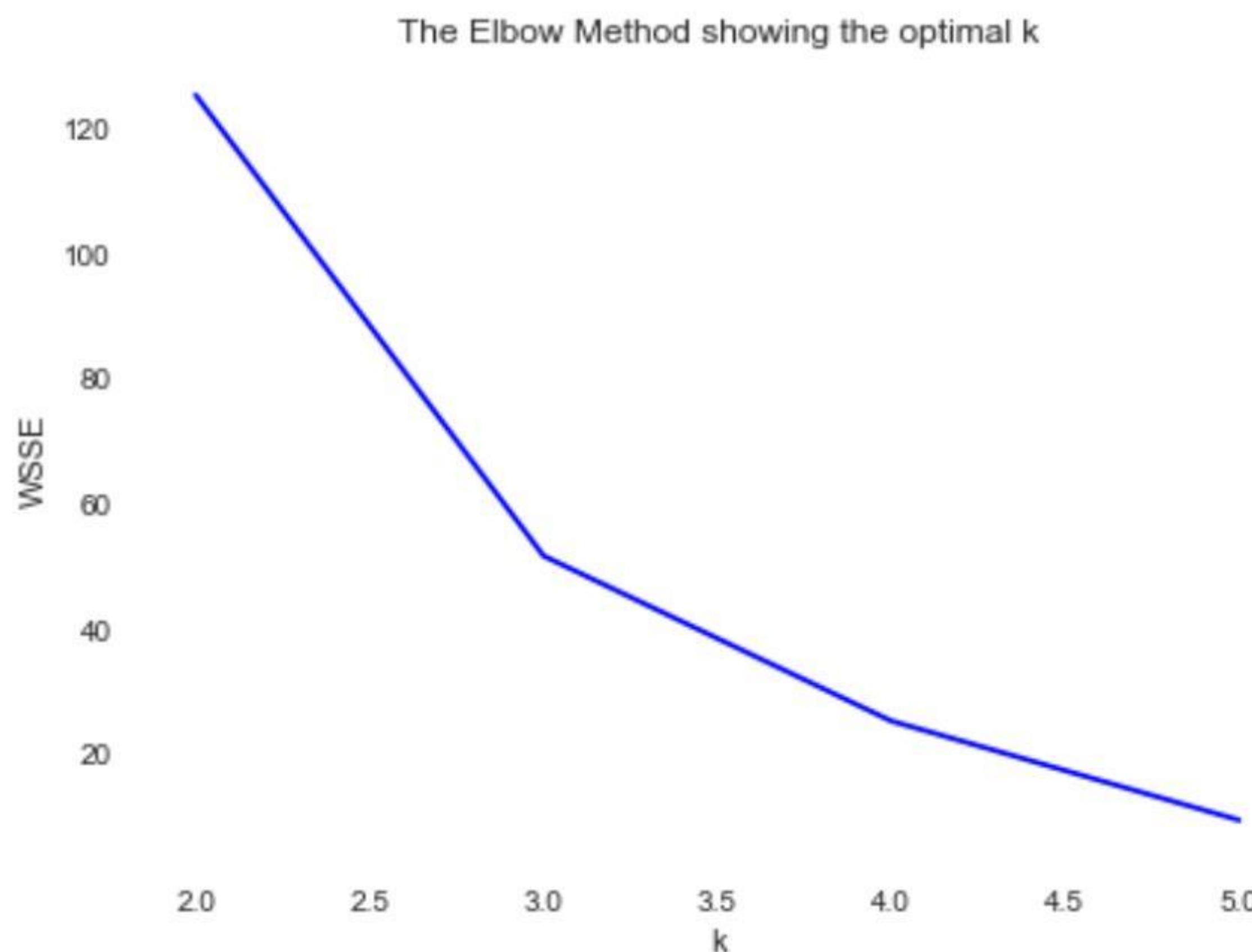
----- Group  0 : -----
Number of samples: 115
      Milk   Grocery  Detergents_Paper  Group
count  115.00  115.00       115.00  115.00
mean   8536.33 12381.90      4741.21  0.00
min    944.00 1660.00        20.00  0.00
max   23527.00 22272.00     17120.00  0.00
----- Group  1 : -----
Number of samples: 289
      Milk   Grocery  Detergents_Paper  Group
count  289.00  289.00       289.00  289.00
mean  2651.94  3283.93      755.99  1.00
min   55.00   3.00         3.00  1.00
max  11006.00 9464.00      5316.00  1.00
----- Group  2 : -----
Number of samples: 5
      Milk   Grocery  Detergents_Paper  Group
count   5.00   5.00         5.00  5.00
mean  43460.60 61472.20     29974.20  2.00
min   4980.00 32114.00     20070.00  2.00
max  73498.00 92780.00     40827.00  2.00
----- Group  3 : -----
Number of samples: 31
      Milk   Grocery  Detergents_Paper  Group
count  31.00  31.00       31.00  31.00
mean  18869.84 26394.45     11427.94  3.00
min   6964.00 17645.00      239.00  3.00
max  43950.00 45828.00     24231.00  3.00
```

```
In [ ]: # Yêu cầu:
# Tiếp tục thực hiện việc phân tích, phân cụm theo các thuộc tính khác
# Có thể thử nghiệm scale dữ liệu và đánh giá kết quả
```

Ví dụ: Theo Channel & Region

```
In [ ]: K = range(2,6)
wsses = []
silhouette = []
for k in K:
    est_kmeans = KMeans(n_clusters=k, random_state = 0)
    est_kmeans.fit(data[["Channel", "Region"]])
    pred_kmeans = est_kmeans.predict(data[["Channel", "Region"]])
    x = silhouette_score(data_sub, pred_kmeans, metric='euclidean')
    silhouette.append(x)
    wsses.append(est_kmeans.inertia_)
```

```
In [ ]: # Trục quan elbow
plt.figure(figsize=(7,5))
plt.plot(K, wsses, 'b')
plt.xlabel('k')
plt.ylabel('WSSE')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



```
In [ ]: # Chọn k=4
kmeans = KMeans(n_clusters = 4, random_state = 0)
kmeans.fit(data[["Channel", "Region"]])
pred_kmeans = kmeans.predict(data[["Channel", "Region"]])
```

```
In [ ]: centroids = kmeans.cluster_centers_
labels = kmeans.labels_

print("Centroids in normal:")
print(centroids)

print("Labels:")
print(labels)

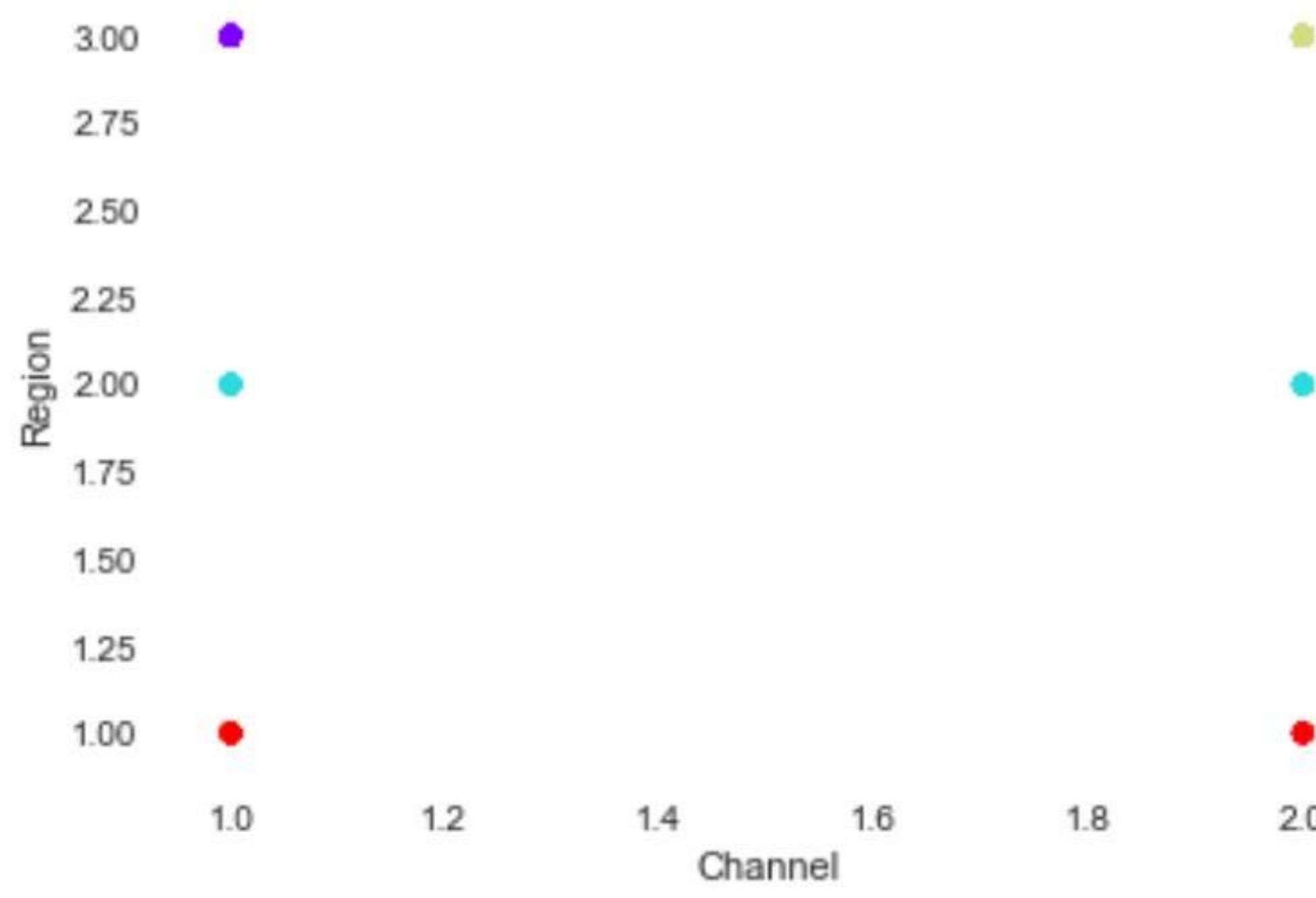
Centroids in normal:
[[1.        3.
  1.40425532 2.
  2.        3.
  1.23376623 1.        ]]
Labels:
[2 2 2 0 2 2 2 2 0 2 2 2 2 2 0 2 0 2 0 2 0 0 2 2 2 0 0 2 0 0 0 0 0 2 0
 2 2 0 0 0 2 2 2 2 2 2 0 0 2 2 0 0 2 2 2 0 0 2 2 2 0 2 0 2 0 0 0 0 0 2 0
 2 0 0 2 0 0 0 2 2 0 2 2 2 0 0 0 0 0 2 0 2 0 2 0 0 0 2 2 2 0 0 0 2 2 2 2 0
 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 2 2 0 2 2 2 0 0 2 2 2 2 0 0 0 2 2 0 2 0 2 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 2 0 0 0 2 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 2 2 0 2 0 0 2 2 0 2 0 2 0 2 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0
 2 0 0 2 0 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 2 2 0 0 0 0 0 2 2 0 2 0 0 2 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

```
In [ ]: clusters = pd.Series(labels)
```

```
In [ ]: clusters.value_counts()
```

```
Out[59]: 0    211
2    105
3     77
1     47
dtype: int64
```

```
In [ ]: plt.scatter(data.Channel, data.Region, c=pred_kmeans, s=30, cmap='rainbow')
plt.xlabel('Channel')
plt.ylabel('Region')
plt.show()
```



#### Giải thích cụm: Khách hàng chia theo cụm

- Channel 1 và Region 1
- Channel 2 và Region 2
- Channel 1 và Region 3
- Channel 2 và Region 3
- Với
  - REGION: Lisbon 77, Oporto 47, Other Region 316
  - CHANNEL: Horeca 298, Retail 142

#### Bước 6: Deployment & Feedback/ Act

- Đưa ra chiến dịch quảng cáo, bán hàng, chăm sóc khách hàng phù hợp cho mỗi nhóm