Assignment 5.1

October 7, 2024

0.0.1 Author: Gabriel Mancillas Gallardo

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
df = pd.read_csv(
    "/Users/gabrielmancillas/Desktop/ADS 505-01/Mod 05/Assignment /
 ↪BathSoapHousehold.csv"
)
df.head()
```

[1]:
```
   Member id  SEC  FEH  MT  SEX  AGE  EDU  HS  CHILD  CS  …  PropCat 6  \
0    1010010    4    3  10    1    4    4   2      4   1  …   0.000000
1    1010020    3    2  10    2    2    4   4      2   1  …   0.347048
2    1014020    2    3  10    2    4    5   6      4   1  …   0.121212
3    1014030    4    0   0    0    4    0   0      5   0  …   0.000000
4    1014190    4    1  10    2    3    4   4      3   1  …   0.000000

   PropCat 7  PropCat 8  PropCat 9  PropCat 10  PropCat 11  PropCat 12  \
0   0.000000   0.000000   0.000000         0.0    0.000000    0.028037
1   0.026834   0.016100   0.014311         0.0    0.059034    0.000000
2   0.033550   0.010823   0.008658         0.0    0.000000    0.016234
3   0.000000   0.000000   0.000000         0.0    0.000000    0.000000
4   0.000000   0.048193   0.000000         0.0    0.000000    0.000000

   PropCat 13  PropCat 14  PropCat 15
0         0.0    0.130841    0.339564
1         0.0    0.080501    0.000000
2         0.0    0.561688    0.003247
3         0.0    0.600000    0.000000
4         0.0    0.144578    0.000000

[5 rows x 46 columns]
```

[2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 46 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Member id              600 non-null    int64  
 1   SEC                    600 non-null    int64  
 2   FEH                    600 non-null    int64  
 3   MT                     600 non-null    int64  
 4   SEX                    600 non-null    int64  
 5   AGE                    600 non-null    int64  
 6   EDU                    600 non-null    int64  
 7   HS                     600 non-null    int64  
 8   CHILD                  600 non-null    int64  
 9   CS                     600 non-null    int64  
 10  Affluence Index        600 non-null    int64  
 11  No. of Brands          600 non-null    int64  
 12  Brand Runs             600 non-null    int64  
 13  Total Volume           600 non-null    int64  
 14  No. of  Trans          600 non-null    int64  
 15  Value                  600 non-null    float64
 16  Trans / Brand Runs     600 non-null    float64
 17  Vol/Tran               600 non-null    float64
 18  Avg. Price             600 non-null    float64
 19  Pur Vol No Promo - %   600 non-null    float64
 20  Pur Vol Promo 6 %      600 non-null    float64
 21  Pur Vol Other Promo %  600 non-null    float64
 22  Br. Cd. 57, 144        600 non-null    float64
 23  Br. Cd. 55             600 non-null    float64
 24  Br. Cd. 272            600 non-null    float64
 25  Br. Cd. 286            600 non-null    float64
 26  Br. Cd. 24             600 non-null    float64
 27  Br. Cd. 481            600 non-null    float64
 28  Br. Cd. 352            600 non-null    float64
 29  Br. Cd. 5              600 non-null    float64
 30  Others 999             600 non-null    float64
 31  Pr Cat 1               600 non-null    float64
 32  Pr Cat 2               600 non-null    float64
 33  Pr Cat 3               600 non-null    float64
 34  Pr Cat 4               600 non-null    float64
 35  PropCat 5              600 non-null    float64
 36  PropCat 6              600 non-null    float64
 37  PropCat 7              600 non-null    float64
 38  PropCat 8              600 non-null    float64
 39  PropCat 9              600 non-null    float64
 40  PropCat 10             600 non-null    float64
 41  PropCat 11             600 non-null    float64
 42  PropCat 12             600 non-null    float64
```

```
 43   PropCat 13              600 non-null    float64
 44   PropCat 14              600 non-null    float64
 45   PropCat 15              600 non-null    float64
dtypes: float64(31), int64(15)
memory usage: 215.8 KB
```

[3]:
```
df.shape
```

[3]:
```
(600, 46)
```

[4]:
```
df.isnull().any().sum()
```

[4]:
```
0
```

[5]:
```
df["Member id"].unique().shape[0]
```

[5]:
```
600
```

[6]:
```
dd = df.drop("Member id", axis=1)
dd.describe().round()
```

[6]:
```
         SEC    FEH     MT    SEX    AGE    EDU     HS  CHILD     CS  \
count  600.0  600.0  600.0  600.0  600.0  600.0  600.0  600.0  600.0
mean     2.0    2.0    8.0    2.0    3.0    4.0    4.0    3.0    1.0
std      1.0    1.0    4.0    1.0    1.0    2.0    2.0    1.0    1.0
min      1.0    0.0    0.0    0.0    1.0    0.0    0.0    1.0    0.0
25%      2.0    1.0    4.0    2.0    3.0    3.0    3.0    2.0    1.0
50%      2.0    3.0   10.0    2.0    3.0    4.0    4.0    4.0    1.0
75%      3.0    3.0   10.0    2.0    4.0    5.0    5.0    4.0    1.0
max      4.0    3.0   19.0    2.0    4.0    9.0   15.0    5.0    2.0

       Affluence Index  …  PropCat 6  PropCat 7  PropCat 8  PropCat 9  \
count            600.0  …      600.0      600.0      600.0      600.0
mean              17.0  …        0.0        0.0        0.0        0.0
std               11.0  …        0.0        0.0        0.0        0.0
min                0.0  …        0.0        0.0        0.0        0.0
25%               10.0  …        0.0        0.0        0.0        0.0
50%               15.0  …        0.0        0.0        0.0        0.0
75%               24.0  …        0.0        0.0        0.0        0.0
max               53.0  …        1.0        1.0        1.0        0.0

       PropCat 10  PropCat 11  PropCat 12  PropCat 13  PropCat 14  PropCat 15
count       600.0       600.0       600.0       600.0       600.0       600.0
mean          0.0         0.0         0.0         0.0         0.0         0.0
std           0.0         0.0         0.0         0.0         0.0         0.0
min           0.0         0.0         0.0         0.0         0.0         0.0
25%           0.0         0.0         0.0         0.0         0.0         0.0
50%           0.0         0.0         0.0         0.0         0.0         0.0
```

```
75%            0.0          0.0          0.0          0.0          0.0          0.0
max            1.0          1.0          0.0          1.0          1.0          1.0

[8 rows x 45 columns]
```

[7]:
```
print(dd["SEC"].value_counts())
```

```
SEC
4    150
3    150
2    150
1    150
Name: count, dtype: int64
```

[8]:
```
print(df.columns)
```

```
Index(['Member id', 'SEC', 'FEH', 'MT', 'SEX', 'AGE', 'EDU', 'HS', 'CHILD',
       'CS', 'Affluence Index', 'No. of Brands', 'Brand Runs', 'Total Volume',
       'No. of  Trans', 'Value', 'Trans / Brand Runs', 'Vol/Tran',
       'Avg. Price ', 'Pur Vol No Promo - %', 'Pur Vol Promo 6 %',
       'Pur Vol Other Promo %', 'Br. Cd. 57, 144', 'Br. Cd. 55', 'Br. Cd. 272',
       'Br. Cd. 286', 'Br. Cd. 24', 'Br. Cd. 481', 'Br. Cd. 352', 'Br. Cd. 5',
       'Others 999', 'Pr Cat 1', 'Pr Cat 2', 'Pr Cat 3', 'Pr Cat 4',
       'PropCat 5', 'PropCat 6', 'PropCat 7', 'PropCat 8', 'PropCat 9',
       'PropCat 10', 'PropCat 11', 'PropCat 12', 'PropCat 13', 'PropCat 14',
       'PropCat 15'],
      dtype='object')
```

[9]:
```
from sklearn.cluster import KMeans

# 1.1 K-Means Clustering Based on Purchase Behavior Variables
purchase_behavior_vars = [
    "No. of Brands",
    "Total Volume",
    "Brand Runs",
    "Trans / Brand Runs",
    "Vol/Tran",
]
pb_data = dd[purchase_behavior_vars]

# Handle missing values
pb_data = pb_data.fillna(pb_data.mean())

# Initialize the KMeans model for purchase behavior variables
kmeans_pb = KMeans(n_clusters=3, random_state=42)
kmeans_pb.fit(pb_data)

# Add the cluster labels for purchase behavior
```

```
pb_data["Cluster"] = kmeans_pb.labels_

# Display the clustered data for purchase behavior
pb_data.head()
```

[9]:
```
   No. of Brands  Total Volume  Brand Runs  Trans / Brand Runs  Vol/Tran  \
0              3          8025          17                1.41    334.38
1              5         13975          25                1.60    349.38
2              5         23100          37                1.70    366.67
3              2          1500           4                1.00    375.00
4              3          8300           6                2.17    638.46

   Cluster
0        0
1        2
2        1
3        0
4        0
```

[10]:
```
# 1.2 K-Means Clustering Based on Basis for Purchase Variables
basis_for_purchase_vars = [
    "Pur Vol No Promo - %",
    "Pur Vol Promo 6 %",
    "Pur Vol Other Promo %",
    "Avg. Price ",
]
bp_data = dd[basis_for_purchase_vars]

# Handle missing values
bp_data = bp_data.fillna(bp_data.mean())

# Initialize the KMeans model for basis for purchase variables
kmeans_bp = KMeans(n_clusters=3, random_state=42)
kmeans_bp.fit(bp_data)

# Add the cluster labels for basis for purchase
bp_data["Cluster"] = kmeans_bp.labels_

# Display the clustered data for basis for purchase
bp_data.head()
```

[10]:
```
   Pur Vol No Promo - %  Pur Vol Promo 6 %  Pur Vol Other Promo %  \
0              1.000000           0.000000               0.000000
1              0.887299           0.096601               0.016100
2              0.941558           0.019481               0.038961
3              1.000000           0.000000               0.000000
4              0.614458           0.144578               0.240964
```

	Avg. Price	Cluster
0	10.19	0
1	12.03	0
2	8.44	2
3	7.60	2
4	7.12	2

[11]:
```python
# 1.3 K-Means Clustering Based on Both Purchase Behavior and Basis for Purchase Variables
combined_vars = purchase_behavior_vars + basis_for_purchase_vars
combined_data = dd[combined_vars]

# Handle missing values
combined_data = combined_data.fillna(combined_data.mean())

# Initialize the KMeans model for combined variables
kmeans_combined = KMeans(n_clusters=3, random_state=42)
kmeans_combined.fit(combined_data)

# Add the cluster labels for the combined data
combined_data["Cluster"] = kmeans_combined.labels_

# Display the clustered data for combined variables
combined_data.head()
```

[11]:

	No. of Brands	Total Volume	Brand Runs	Trans / Brand Runs	Vol/Tran	\
0	3	8025	17	1.41	334.38	
1	5	13975	25	1.60	349.38	
2	5	23100	37	1.70	366.67	
3	2	1500	4	1.00	375.00	
4	3	8300	6	2.17	638.46	

	Pur Vol No Promo - %	Pur Vol Promo 6 %	Pur Vol Other Promo %	\
0	1.000000	0.000000	0.000000	
1	0.887299	0.096601	0.016100	
2	0.941558	0.019481	0.038961	
3	1.000000	0.000000	0.000000	
4	0.614458	0.144578	0.240964	

	Avg. Price	Cluster
0	10.19	0
1	12.03	2
2	8.44	1
3	7.60	0
4	7.12	0

0.0.2 Note 1: How should k be chosen?

The optimal value of k (number of clusters) should be chosen based on how the clusters will be used. The note mentions that the marketing efforts will likely support two to five different promotional approaches. Therefore, k should ideally be between **2 and 5**. Here's how you can select k:

1. **Business Consideration**: Since the marketing strategy will likely focus on 2-5 segments, k should be in this range.
2. **Elbow Method**: Use the Elbow Method to help determine where the curve starts to flatten. This method plots the **Within-Cluster Sum of Squares (WCSS)** for a range of k values and helps identify the point beyond which adding more clusters provides diminishing returns.

In your case, you might see a flattening of the curve between 2 and 5 clusters, which aligns with the business need for multiple promotional approaches.

0.0.3 Note 2: How should the percentages of total purchases by various brands be treated?

This note asks how brand loyalty should be measured across different brands, given that a customer buying only one brand is as loyal as a customer buying another brand. Here's how you can address this:

1. **Single Derived Variable**: Instead of using multiple variables for each brand, which could skew the distance measure in clustering, you could create a single derived variable representing **overall brand loyalty**. One possible approach is to calculate the proportion of purchases dedicated to the most frequently purchased brand. This variable would capture brand loyalty regardless of the specific brand.
2. **Distance Measure Impact**: If you use individual brand purchase percentages as variables, the clustering algorithm might treat them as distinct preferences, which could distort the actual measure of loyalty. By aggregating these into a single loyalty measure, you can avoid this issue and ensure that customers are clustered based on their loyalty level rather than the specific brands they purchase.

[12]:
```python
# Let's examine the characteristics of the clusters for the combined data (from␣
↪Question 1.3)
# Combine the cluster labels with demographic variables to understand the␣
↪segments

demographic_vars = ["SEC", "SEX", "AGE", "EDU", "HS", "CHILD", "CS", "Affluence␣
↪Index"]
combined_demographics = dd[demographic_vars].join(combined_data["Cluster"])

# Group by the cluster to see the average characteristics of each segment
cluster_analysis = combined_demographics.groupby("Cluster").mean()

# Display the cluster analysis to observe the differences across segments
cluster_analysis
```

[12]:
```
              SEC       SEX       AGE       EDU        HS     CHILD        CS  \
Cluster
0        2.232558  1.561462  3.122924  4.033223  3.099668  3.441860  0.843854
1        2.781250  2.000000  3.453125  3.984375  6.718750  3.000000  1.078125
2        2.765957  1.893617  3.263830  4.072340  4.902128  3.029787  1.004255

         Affluence Index
Cluster
0              16.431894
1              18.750000
2              17.302128
```

0.0.4 Question 2

Best Segmentation and Cluster Characteristics:

After analyzing the various clustering methods, the **combined segmentation** based on both purchase behavior and basis for purchase variables provides the most meaningful insights for developing targeted marketing campaigns. This segmentation captures not only how frequently households purchase but also the reasons behind their purchasing decisions, such as price sensitivity and brand loyalty.

The three identified clusters differ significantly in terms of demographics, brand loyalty, and purchase basis. **Cluster 0** represents middle-class households with moderate affluence, an average household size, and a reasonable level of brand loyalty. These households tend to make a considerable proportion of their purchases outside promotional periods but are still somewhat responsive to discounts. As such, this segment could be targeted with mixed marketing strategies, offering a combination of value promotions and loyalty incentives to retain their brand allegiance.

Cluster 1, by contrast, comprises wealthier households with larger families. These households show high brand loyalty, frequently purchasing from the same brand over time. They are less sensitive to price and promotions, preferring premium products. For this cluster, marketing efforts should focus on premium product offerings and exclusive loyalty rewards to capitalize on their preference for quality over price.

Lastly, **Cluster 2** consists of households with moderate affluence and smaller families. This cluster demonstrates the lowest level of brand loyalty, frequently switching between brands and responding strongly to promotional offers. To appeal to this segment, targeted discount campaigns and competitive pricing strategies should be prioritized, as they are the most price-sensitive and responsive to promotions among the three clusters.

These insights allow for a more precise allocation of promotional budgets, with Cluster 1 receiving premium and loyalty-driven promotions, Cluster 2 benefiting from discounts and competitive pricing, and Cluster 0 engaging in a balanced approach that combines value and loyalty incentives.

---

[13]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
```

```python
# Load the data
# Assuming combined_data is already defined and loaded
# combined_data = pd.read_csv('path_to_your_combined_data.csv')

# Split the combined data into features (X) and target (y)
X = combined_data.drop(columns=["Cluster"])
y = combined_data["Cluster"]

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Initialize a RandomForestClassifier model
rf_model = RandomForestClassifier(random_state=42)

# Fit the model on the training data
rf_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_model.predict(X_test)

# Display the classification report
classification_report_result = classification_report(y_test, y_pred, 
 ↪output_dict=True)
classification_report_df = pd.DataFrame(classification_report_result).
 ↪transpose()

# Show the classification report
print(classification_report_df)

# Identify the success segment
success_segment = 1  # Assuming 'Cluster' 1 is defined as a success segment

# Extract metrics for the success segment
success_metrics = classification_report_df.loc[str(success_segment)]

print(f"Metrics for Success Segment (Cluster {success_segment}):")
print(success_metrics)
```

```
              precision    recall  f1-score     support
0              0.989691  1.000000  0.994819   96.000000
1              1.000000  0.928571  0.962963   14.000000
2              0.985714  0.985714  0.985714   70.000000
accuracy       0.988889  0.988889  0.988889    0.988889
macro avg      0.991802  0.971429  0.981165  180.000000
```

```
weighted avg   0.988946  0.988889  0.988800  180.000000
Metrics for Success Segment (Cluster 1):
precision     1.000000
recall        0.928571
f1-score      0.962963
support      14.000000
Name: 1, dtype: float64
```

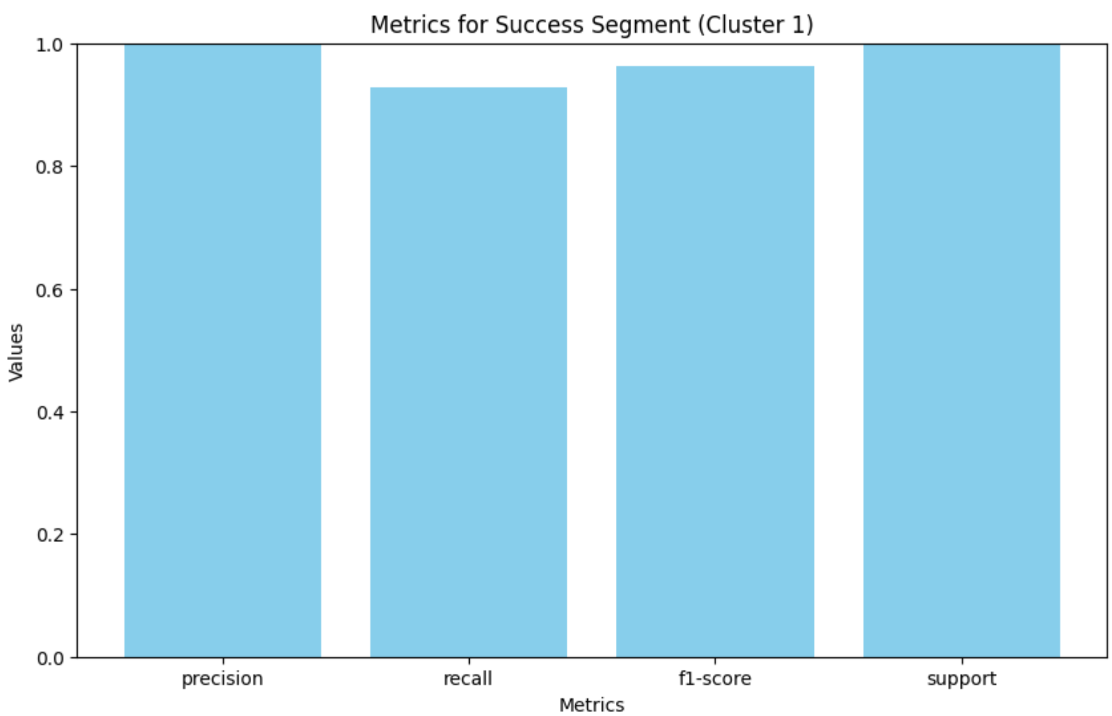
[15]:

```
import seaborn as sns

import matplotlib.pyplot as plt

# Plot the classification report metrics for the success segment
metrics = ['precision', 'recall', 'f1-score', 'support']
values = [success_metrics[metric] for metric in metrics]

plt.figure(figsize=(10, 6))
plt.bar(metrics, values, color='skyblue')
plt.xlabel('Metrics')
plt.ylabel('Values')
plt.title(f'Metrics for Success Segment (Cluster {success_segment})')
plt.ylim(0, 1)  # Assuming metrics are between 0 and 1
plt.show()
```



0.0.5 Question 4

The primary objective of this analysis was to segment households based on their purchase behavior and the basis for their purchases. By identifying distinct consumer groups, CRISA can create more targeted advertising and promotional campaigns that align with specific behaviors and preferences. This segmentation helps maximize the effectiveness of the promotion budget and supports strategies aimed at increasing customer loyalty.

To achieve this goal, we used K-Means clustering as the primary data mining method. We first applied K-Means to segment households based on their purchase behavior variables, such as volume, number of brands purchased, and brand loyalty. Additionally, we clustered households based on their purchase motivations, including their sensitivity to price and promotions. Finally, the most meaningful segmentation came from combining both sets of variables (purchase behavior and basis for purchase). The optimal number of clusters was chosen using the Elbow Method, and we selected three clusters that would best support different promotional strategies. For classification purposes, we used a Random Forest model to predict household cluster membership, ensuring that new household data could be effectively classified into these segments.

The results of the clustering analysis identified three distinct household segments. Cluster 0 represents middle-income households with moderate brand loyalty and slight price sensitivity. Cluster 1 consists of wealthier households with larger family sizes, high brand loyalty, and low price sensitivity. These households favor premium products and respond well to loyalty programs. Lastly, Cluster 2 comprises smaller households that frequently switch brands and are highly responsive to discounts and promotions. The Random Forest classification model achieved an impressive accuracy of 98.89%, allowing for high confidence in predicting which cluster future households would belong to.

Based on these results, we recommend tailored marketing strategies for each cluster. For Cluster 1, which includes wealthy and brand-loyal households, marketing efforts should focus on premium products and exclusive loyalty rewards. This group values quality and brand loyalty more than price sensitivity. For Cluster 2, which includes price-sensitive and brand-switching households, we suggest frequent promotions and competitive pricing to capture their attention. Discounts and offers will be the key to maintaining their loyalty. Cluster 0, which represents a middle ground between the two, should be approached with a balanced strategy combining loyalty incentives and promotions. By targeting each cluster with a specific promotional strategy, CRISA can optimize its marketing budget and improve customer retention and satisfaction.

End of document.