MARKET SEGMENT ANALYSIS

Gyanpriya Misra

FEYNN LABS

Content

Introduction

1. Marketing: Definition and Purpose

Marketing is the process by which companies identify, anticipate, and satisfy customer needs and desires through the promotion and sale of products or services. It is not just about advertising or selling; it includes market research, product design, pricing, distribution, and customer service.

The purpose of marketing is to create value for both the company and its customers by meeting consumer needs while driving business growth. It ensures that the right product is available at the right place, at the right time, and at the right price, while also fostering customer relationships and building brand loyalty.

2. Marketing Planning: Strategic and Tactical

Marketing planning involves two key aspects: **strategic planning** and **tactical planning**.

Strategic planning refers to the long-term vision and goals of a company, focusing on market positioning, product development, and competitive analysis. It defines the overall direction and mission of the company in the market.

Tactical planning, on the other hand, deals with the specific actions and short-term goals that support the strategic plan. It includes developing marketing campaigns, pricing strategies, promotional efforts, and distribution channels to ensure the strategic goals are met. Strategic marketing sets the course, while tactical marketing executes the plan.

3. Market Segmentation: Definition, Benefits, and Costs

Market segmentation is the process of dividing a broad market into smaller, more defined groups of consumers with similar needs, preferences, or behaviours. This allows companies to tailor their products and marketing efforts to meet the specific needs of each segment.

The **benefits** of segmentation include more targeted marketing, better customer satisfaction, and higher return on investment (ROI). It allows businesses to focus resources more effectively and build stronger relationships with their customers.

However, segmentation also comes with **costs**, such as the need for more complex marketing strategies, increased product variations, and potentially higher production and distribution costs.

4. Market Segmentation Analysis: Layers

a. Conducting High-Quality Market Segmentation Analysis

Market segmentation analysis involves extracting distinct market segments from a larger customer base. This process helps companies identify the unique characteristics of each segment and tailor marketing strategies accordingly.

b. Enabling High-Quality Market Segmentation Analysis

High-quality segmentation requires good data collection, exploring that data, profiling the identified segments, and describing them in detail. Accurate data is essential for understanding customer behaviour, needs, and preferences. Segment profiling helps businesses understand how to approach each group, while segment descriptions clarify the unique features of each segment.

c. Making it Happen in Practice

Implementing market segmentation involves several steps. Companies must first decide whether to pursue segmentation. They then define the ideal market segment and select the most attractive segments to target. A

customized marketing mix is developed for each segment, and effectiveness is assessed through continuous monitoring to ensure the strategy remains relevant as market conditions evolve.

5. Market Segment Analysis: Approaches

a. Based on Organizational Constraints

Organizations need to consider internal constraints, such as budget, resources, and capabilities, when deciding how to approach market segmentation.

b. Based on Choice of Segmentation Variables

Segmentation variables can be demographic (age, income), geographic (location), psychographic (lifestyle, personality), or behavioural (purchase habits). The choice of variables depends on the product, market, and goals of the business.

c. Data Structure and Data-Driven Approaches

Data-driven segmentation uses statistical methods, such as cluster analysis, to analyse customer data and identify distinct segments. This approach is more precise and data-backed compared to traditional methods, which may rely on intuition or experience.

6. Market Segment Analysis: Step by Step

1. **Weigh Advantages and Disadvantages**: The organization must evaluate whether the benefits of segmentation outweigh the costs and decide whether to proceed.
2. **Specify Ideal Market Segment**: Define the characteristics of the ideal customer group.
3. **Collect or Compile Data**: Once the segment characteristics are specified, data is collected or compiled from existing sources.
4. **Explore Data**: This step involves examining the data to identify patterns and insights that can inform segmentation.
5. **Extract Market Segments**: Use data analysis techniques to uncover distinct segments within the market.
6. **Profile Segments**: Each identified segment is analysed to understand its key characteristics.
7. **Describe Segments**: Detailed descriptions of the segments are developed, highlighting customer needs and preferences.
8. **Select Target Segments**: Based on analysis, one or more market segments are chosen for targeting.
9. **Develop a Customized Marketing Mix**: A tailored marketing strategy is created for the selected segment(s), focusing on product, price, place, and promotion.
10. **Evaluate and Monitor**: The effectiveness of the segmentation strategy is continuously evaluated, and adjustments are made as the market evolves or segment characteristics change.
11. This step-by-step approach ensures that companies can effectively identify and target the most profitable market segments, while also remaining agile and responsive to changes in the marketplace.

Step 1: Deciding (not) to Segment

Implications of Committing to Market Segmentation

Market segmentation has become a fundamental marketing strategy used by many organizations, but it is not always the best decision to pursue. The decision to adopt market segmentation requires careful consideration of the long-term implications. The key implication is that market segmentation is a long-term commitment rather than a short-term tactic. It demands significant changes in organizational structure, product development, pricing, and communication strategies. As Cahill (2006) suggests, segmenting a market incurs costs, such as research, surveys, and designing multiple marketing materials. The expected increase in sales must outweigh the expenses associated with developing and implementing the segmentation strategy.

Organizations may also need to modify their internal structure, focusing on market segments rather than individual products. Strategic business units could be created to manage the needs of different segments. Given the extensive commitment and resources required, the decision to pursue market segmentation should be made at the executive level and communicated across all organizational units.

Implementation Barriers

Various barriers can impede the successful implementation of a market segmentation strategy. Leadership is crucial—without senior management's commitment, resources, and proactive involvement, the strategy is likely to fail. Organizational culture also plays a significant role. Resistance to change, lack of market orientation, poor communication, and short-term thinking can all hinder implementation.

Training is essential to ensure both management and the segmentation team understand the market segmentation process. Lack of formal marketing expertise or data analysis skills can also be a major stumbling block. Structural and financial limitations, unclear objectives, and inadequate planning further contribute to the difficulties of implementing segmentation.

Checklist for Moving Forward

Before committing to market segmentation, organizations should ask themselves the following questions:

- Is the organization market-oriented?
- Is the organization genuinely willing to change?
- Does the organization take a long-term perspective?
- Is the organization open to new ideas?
- Is communication across organizational units effective?
- Can the organization make significant structural changes?
- Are there sufficient financial resources to support the strategy?

If the answer to these questions is "yes," then the organization can proceed with segmentation. Additionally, senior management must commit to the strategy, both financially and operationally. Lastly, ensure that the market segmentation concept and its implications are fully understood across the organization.

Step 2: Specifying the Ideal Target Segment

1. Segment Evaluation Criteria

In market segmentation analysis, evaluating potential target segments involves two critical sets of criteria: knock-out and attractiveness criteria. These criteria guide an organization in selecting the most suitable market segments for targeting.

a. Knock-Out Criteria

Knock-out criteria are non-negotiable features that eliminate market segments that do not meet essential requirements. These criteria include:

- **Homogeneity**: Segment members must be similar.
- **Distinctness**: Segment members must be different from those in other segments.
- **Size**: The segment should be large enough to justify the investment.
- **Match to Organizational Strengths**: The organization must have the capability to meet the segment's needs.
- **Identifiability**: It must be possible to identify segment members.
- **Reachability**: The organization should be able to communicate with the segment.

These criteria ensure that segments failing to meet these basic requirements are eliminated early in the segmentation process. Knock-out criteria must be defined and agreed upon by senior management, the segmentation team, and an advisory committee. Certain criteria, like the minimum viable segment size, may need to be specified in detail.

b. Attractiveness Criteria

Once a segment meets knock-out criteria, its attractiveness is assessed. Unlike knock-out criteria, attractiveness criteria are not binary but are rated on a scale. Segments are evaluated based on:

- **Profitability**
- **Growth potential**
- **Competitive intensity**
- **Strategic fit**
- **Accessibility**
- **Cost efficiency**

Attractiveness criteria help the organization determine how viable and desirable each segment is for targeting. The segmentation team must weigh each criterion according to the organization’s strategic goals.

2. Implementing a Structured Process

A structured process for segment evaluation ensures consistency and collaboration across the organization. One effective approach is the **segment evaluation plot**, which maps segment attractiveness on one axis and organizational competitiveness on the other.

Key Steps for Implementation:

- **Cross-Functional Team Involvement**: Representatives from different units ensure that the chosen criteria reflect the organization's varied perspectives. These stakeholders will also help implement the segmentation strategy.
- **Early Selection of Criteria**: Choosing attractiveness criteria early helps ensure that relevant data is collected and that segment selection is streamlined.
- **Assigning Weights to Criteria**: The team distributes 100 points across attractiveness criteria to indicate their relative importance, ensuring a quantitative basis for decision-making.

3. Checklist for Target Segment Specification

1. **Convene a segmentation team meeting**.
    - Discuss and agree on knock-out criteria, such as homogeneity, distinctness, size, and reachability.
    - Ensure the automatic elimination of non-compliant market segments.
2. **Present knock-out criteria to the advisory committee** for discussion and adjustment if needed.
3. **Review attractiveness criteria**.
    - Select six or fewer criteria that reflect market attractiveness.
4. **Distribute 100 points across the attractiveness criteria**.
    - Reflect the relative importance of each criterion.
5. **Discuss and finalize weightings** with the team.
6. **Present criteria and weights to the advisory committee** for approval or adjustment.

Effective market segmentation requires careful evaluation of target segments based on knock-out and attractiveness criteria. By implementing a structured process and involving multiple stakeholders, organizations can select the ideal target segments that align with both market opportunities and organizational strengths.

Step 3: Collecting Data

1. Segmentation Variables

Market segmentation relies on empirical data to identify or create distinct segments within a market. Segmentation variables are critical in both commonsense and data-driven segmentation, as they serve to split a sample into relevant market segments.

- **Commonsense Segmentation**: It typically relies on a single segmentation variable to create segments. This approach uses descriptor variables (e.g., socio-demographics) to provide a detailed description of each segment, which is essential for tailoring marketing strategies.
- **Data-Driven Segmentation**: Unlike commonsense segmentation, this method uses multiple segmentation variables. It helps identify segments based on shared characteristics, which may not be immediately apparent from individual segmentation variables. For example, a segment of tourists may share common preferences, like seeking relaxation or adventure, rather than having socio-demographic characteristics in common.

Data for segmentation can come from various sources, including survey studies, observations (e.g., scanner data), and experimental studies. The best data source reflects actual consumer behaviour, as surveys may sometimes fail to capture true behavioural patterns.

Empirical Data Sources

- **Survey Studies**: While surveys are common for market segmentation, they may not always represent actual behavior, especially in cases where respondents provide socially desirable answers.
- **Observation Data**: Purchase behavior and loyalty programs provide reliable data that reflects actual consumer behavior.
- **Experimental Studies**: These can yield data that simulate real-world behavior, helping understand potential market segments more accurately.

2. Segmentation Criteria

Choosing the right segmentation criteria is critical before collecting data. Unlike segmentation variables, which refer to individual measured values (e.g., survey items), segmentation criteria are broader and relate to the type of information used for segmentation. Common criteria include geographic, socio-demographic, psychographic, and behavioral factors.

2.1 Geographic Segmentation

This criterion is based on consumers' locations, such as countries, regions, or cities. It is a traditional and simple approach to segmentation, especially useful in industries like tourism. For example, a national tourism organization may treat tourists from different countries as separate segments due to differences in language and culture.

- **Advantages**: Easy to assign consumers to geographic units, enabling targeted communication.
- **Disadvantages**: Geographic proximity does not always translate into shared preferences. Consumers from the same location may still have vastly different interests, which limits the usefulness of geographic segmentation in certain industries.

2.2 Socio-Demographic Segmentation

This approach uses criteria like age, gender, income, and education to create market segments. Certain industries, such as luxury goods, cosmetics, or tourism, benefit from socio-demographic segmentation.

- **Advantages**: Easy to determine segment membership. In some cases, socio-demographic factors explain product preferences (e.g., families with children choosing family vacation destinations).
- **Disadvantages**: Socio-demographics often explain only a small part of consumer behavior. Demographics alone may not capture the full range of preferences and behaviors relevant to a product.

2.3 Psychographic Segmentation

Psychographic segmentation groups consumers based on psychological criteria such as beliefs, interests, and motivations. This method provides deeper insights into the underlying reasons behind consumer behavior. For example, tourists who are interested in learning about other cultures may prefer destinations rich in cultural heritage.

- **Advantages**: Reflects the true motivations behind consumer behavior, making it highly effective in industries like tourism or lifestyle products.
- **Disadvantages**: It requires complex data collection, and the reliability of psychographic measures must be ensured for effective segmentation.

2.4 Behavioral Segmentation

Behavioral segmentation focuses on actual or reported behavior, such as purchase history, frequency of purchases, or spending patterns. Behavioral segmentation provides the most relevant insights when the behavior of interest is used as the basis for segmenting consumers.

- **Advantages**: Behavioral data directly reflects consumer actions, making it a powerful tool for identifying market segments based on relevant behaviours.
- **Disadvantages**: Access to behavioral data may be limited, especially when trying to include potential customers who have not yet purchased the product.

3. Data from Survey Studies

Survey data is commonly used for market segmentation due to its accessibility and affordability. However, it often contains biases that can compromise the quality of the segmentation. Addressing these biases and understanding the limitations of survey data is essential for ensuring robust segmentation outcomes.

3.1 Choice of Variables

Choosing appropriate variables for market segmentation is crucial to ensure that the analysis is both effective and efficient. In data-driven segmentation, relevant variables must be included, while unnecessary variables (referred to as "noisy" or "masking" variables) should be avoided, as they can dilute the quality of the segmentation. Including too many variables can overwhelm respondents, leading to fatigue, which negatively affects the quality of their responses. Moreover, redundant or irrelevant variables increase the complexity of the analysis without adding meaningful insights.

In traditional survey research, questions are often repeated or varied slightly to measure consistency (following psychometric principles). However, when applied to segmentation, this redundancy can interfere with the algorithm's ability to identify correct segments. Therefore, surveys should be carefully designed to ask only the most relevant questions. Exploratory research, such as qualitative interviews, can be an effective way to identify critical variables before conducting a larger quantitative survey.

3.2 Response Options

The response options provided in a survey determine the type of data (binary, ordinal, metric) that will be generated, which influences the suitability of different data analytic techniques. Binary and metric data are preferred for segmentation analysis as they allow the use of standard distance measures without assumptions. Ordinal data, while common, poses challenges due to undefined distances between response options. If nuanced responses are necessary,

visual analogue scales (sliders) can be used to capture continuous data, which can be treated as metric data for segmentation purposes.

3.3 Response Styles

Response styles, such as tendencies to agree with all items (acquiescence bias) or to use extreme answers, can distort segmentation results. Segmentation algorithms often fail to differentiate between genuine responses and those influenced by response styles. This can lead to the identification of misleading segments that do not accurately reflect consumer behavior. To mitigate the effects of response styles, it is critical to minimize these biases during data collection and perform additional analyses to ensure that identified segments are not artifacts of response styles.

3.4 Sample Size

There is no universal rule for the ideal sample size in market segmentation analysis, but guidelines have been developed for specific algorithms and data types. One rule of thumb suggests that the sample size should be at least 2^p (or five times that), where *p* is the number of segmentation variables. Larger sample sizes generally improve the accuracy of segment identification, but the benefit diminishes as the sample size grows beyond a certain point. A recommended guideline is to have at least 100 respondents for each segmentation variable to ensure the reliability of the segmentation results. Additionally, issues such as unequal segment sizes and overlap between segments can make it more difficult for algorithms to extract the correct segments, emphasizing the importance of collecting high-quality, unbiased data.

4. Data from Internal Sources

Internal data, such as purchase histories or loyalty program data, can provide valuable insights for market segmentation. The primary advantage of internal data is that it reflects actual consumer behavior rather than self-reported intentions, making it more reliable. However, internal data may over-represent existing customers, limiting its usefulness for understanding potential future customers or untapped market segments. This limitation highlights the need for complementing internal data with external data sources to capture a broader market perspective.

5. Data from Experimental Studies

Experimental data, such as responses to advertisements or results from conjoint analysis, can also be used for market segmentation. This type of data is particularly useful for understanding consumer preferences and the factors that influence decision-making. For example, conjoint analysis reveals how different product attributes influence consumer choice, providing a basis for segmenting the market based on preferences for specific product features. Experimental data is valuable because it allows marketers to observe how consumers behave in controlled scenarios, offering insights into how they might respond in the real world.

6. Checklist for Market Segmentation

1. **Assemble a market segmentation team**: Gather key stakeholders to discuss potential segmentation variables and strategies.
2. **Identify segmentation variables**: Determine which consumer characteristics will be used to extract groups from the data.
3. **Identify descriptor variables**: Select variables that will provide a detailed understanding of each segment after they are identified.
4. **Design data collection methods**: Ensure the data collection process captures valid, high-quality responses while minimizing biases and errors.
5. **Collect data**: Gather the necessary data to proceed with segmentation analysis.

Step 4: Exploring Data

1. A First Glimpse at the Data

Data exploration is a crucial step in any data-driven analysis, enabling analysts to understand the dataset's structure and characteristics before applying more complex algorithms. After data collection, exploratory data analysis (EDA) cleans and pre-processes the data to ensure consistency and quality. Additionally, EDA provides guidance on which algorithms are suitable for further analysis, such as market segmentation.

Technically, data exploration helps in:

- **Identifying measurement levels** of variables.
- **Investigating univariate distributions** (e.g., histograms and box plots).
- **Assessing dependencies between variables** through correlation analysis or cross-tabulation.

The results of EDA help in selecting the most appropriate segmentation methods for extracting meaningful insights. Functions like read.csv(), colnames(), dim(), and summary() in R provide quick overviews of the dataset structure, its dimensions, and basic statistics for both numeric and categorical data.

2. Data Cleaning

Before beginning the analysis, data cleaning is essential to ensure accuracy and reliability. Cleaning involves:

- Checking if values are recorded correctly.
- Validating if categorical variable labels are consistent.
- Ensuring numeric variables fall within an expected range (e.g., ages between 0 and 110).

For categorical variables, ensuring permissible values is crucial. For example, if gender is surveyed as male and female, then only these values should be present unless other options are specified.

Data cleaning should be performed using code, ensuring that all steps are fully documented and reproducible. This is especially important for scenarios where new data will be added regularly, ensuring that the same cleaning process can be applied repeatedly. Once the data is cleaned, it is typically saved using R’s save() function for future use.

3. Descriptive Analysis

Descriptive analysis is essential to gain an initial understanding of the dataset. Descriptive statistics provide insights into the distribution and properties of the data, helping to avoid misinterpretation during more complex analyses.

The summary() function in R gives numeric summaries of the data, such as ranges, quartiles, means, and counts of missing values. Visualizing data is equally important, with tools such as:

- **Histograms**: Visualize the frequency distribution of numeric data.
- **Boxplots**: Show the spread and potential outliers of a variable.
- **Scatter plots**: Assess relationships between pairs of numeric variables.
- **Bar plots**: Visualize the distribution of categorical variables.

- **Mosaic plots**: Illustrate relationships between categorical variables.

4. Pre-Processing

4.1 Categorical Variables

Categorical data often requires pre-processing to make it suitable for analysis. Two common techniques are:

1. **Merging levels**: This is done when categories are too differentiated and merging them helps in better segmentation.
2. **Converting categorical variables to numeric**: Ordinal scales (like Likert scales) can sometimes be converted to numeric if it makes sense, allowing for the application of statistical methods requiring numeric input.

Binary variables are often converted to numeric (e.g., 0/1) for use in machine learning models. This transformation is straightforward and preserves the data's meaning.

4.2 Numeric Variables

Standardization is critical when working with numeric variables, particularly in distance-based methods. For example, a variable with a range from 0 to 1 (e.g., a binary variable) will have less influence than a variable ranging from 0 to 1000 unless standardized. The most common standardization method involves subtracting the mean and dividing by the standard deviation to put all variables on the same scale.

In cases of extreme outliers, robust methods (e.g., using the median and interquartile range) can be used for better scaling.

5. Principal Components Analysis (PCA)

PCA is a dimensionality reduction technique often used in multivariate data analysis. It transforms a dataset into new variables (principal components) that are uncorrelated and ranked by their importance. The first component captures the most variability, followed by the second, and so on. While PCA does not reduce the data's dimensionality by itself, it helps in identifying which variables contribute the most variability and allows for plotting high-dimensional data in lower dimensions.

PCA is often used to simplify complex datasets by reducing the number of variables, while still retaining most of the information. However, when extracting market segments, using PCA for dimensionality reduction may lead to loss of valuable information. Instead, PCA can be used to identify highly correlated variables, which can then be removed to avoid redundancy.

6. Checklist for Data Exploration

- **Explore the data** for inconsistencies and contaminations.
- **Clean the data** to ensure all variables have permissible values.
- **Pre-process the data** to standardize or convert variables where necessary.
- Ensure you have enough sample size for each segmentation variable.
- **Check correlations** between segmentation variables and select a subset of uncorrelated ones.
- **Pass on cleaned data** for further analysis and segmentation.

This step-by-step exploration and preparation of data provide the foundation for more sophisticated analysis, ensuring the final models and insights are based on clean, well-understood data.

Step 5: Extracting Segments

5.1 Grouping Consumers

Market segmentation is an exploratory process, as consumer datasets are often unstructured. Consumers have diverse preferences, and the results of segmentation strongly depend on the assumptions and methods used. Segmentation methods, particularly clustering, shape the outcome by imposing a structure on the data.

Clustering Methods

Many segmentation techniques stem from cluster analysis. The suitability of any method depends on both the nature of the data and the specific clustering features required by the researcher. Different algorithms can produce different segmentation solutions, making it essential to explore multiple methods. For instance, **k-means clustering** tends to create compact, equally-sized clusters, which may overlook natural shapes like spirals in data. In contrast, **single linkage hierarchical clustering** can identify irregular clusters, such as snake-shaped or spiral ones. However, this does not mean one method is universally superior. Each algorithm has strengths and weaknesses based on the data structure.

No Universal Best Algorithm

No single segmentation method performs best in all cases. The effectiveness of an algorithm is highly dependent on the structure of the data. If consumer data is well-separated, algorithmic differences matter less. In cases of unstructured data, the algorithm can significantly influence the segmentation outcome by imposing its inherent structure. For this reason, researchers should investigate multiple algorithms before selecting a final solution.

Types of Methods

Segmentation extraction methods fall into two main categories:

- **Distance-based methods**: These rely on measuring the similarity or distance between observations to form groups of similar consumers.
- **Model-based methods**: These formulate statistical models for the market segments. Additionally, some methods combine multiple aims, such as variable selection and segment extraction in one step.

Data Characteristics & Algorithm Selection

Key factors in selecting a segmentation algorithm include the size of the dataset, the number of consumers, expected segments, and segment sizes. Larger datasets with many segmentation variables may require algorithms capable of variable selection during extraction. The scale of segmentation variables also influences the choice of algorithm. For example, data containing repeated measurements of consumers over time would need a model-based algorithm to account for its longitudinal nature.

Direct vs. Indirect Consumer Characteristics

When defining segments, it's important to specify the consumer characteristics that should be common within a segment. These can be direct (e.g., benefits sought) or indirect (e.g., price sensitivity). Indirect characteristics often require more complex models, such as regression models in a model-based approach.

Binary Segmentation Variables

When dealing with binary segmentation variables, it's essential to decide whether to treat these variables symmetrically or asymmetrically. In certain cases, such as vacation activities, treating them asymmetrically makes sense. **Biclustering** is an example of a method that handles binary variables asymmetrically, focusing on common 1s rather than both 0s and 1s.

Conclusion

Segmentation is highly dependent on both the data and the algorithm used. As no single method works best for all data types, researchers must explore and compare different approaches. Understanding the nature of the data and the intended segment characteristics is key to selecting the right algorithm for accurate and meaningful market segmentation.

5.2 Distance based Methods

Market segmentation involves grouping consumers with similar needs or behaviours. To accomplish this, we need to define a measure of similarity or dissimilarity, known as a distance measure. Several distance measures can be used in cluster analysis and market segmentation, with each calculating the distance between two data points (vectors). These measures allow us to group similar data points together effectively.

5.2.1 Distance Measures

A distance measure evaluates the distance between two vectors, resulting in a nonnegative value. Common distance measures include:

- **Euclidean Distance**: The most common distance measure, representing the straight-line distance between two points in a multidimensional space.
- **Manhattan (or Absolute) Distance**: Reflects distance on a grid, as if navigating through city streets. It sums the absolute differences along each dimension.
- **Asymmetric Binary Distance**: Used for binary data, focusing on where at least one of the vectors has a value of 1. It is useful when the presence of an attribute (1) is more informative than its absence (0).

Euclidean and Manhattan distances consider all dimensions of the data, but if dimensions are on different scales (e.g., spending amount vs. activity participation), it may be necessary to standardize the data.

5.2.2 Hierarchical Methods

Hierarchical clustering is intuitive as it mimics how humans might group data. It can be:

- **Divisive**: Starting with all data in one cluster and splitting it iteratively.
- **Agglomerative**: Starting with each data point as its own cluster and merging them step by step.

The merging or splitting is based on a distance measure and a linkage method:

- **Single Linkage**: Distance between the closest members of two groups.
- **Complete Linkage**: Distance between the farthest members of two groups.
- **Average Linkage**: The average distance between all members of two groups.

A popular method, **Ward's clustering**, minimizes the increase in squared Euclidean distance when joining clusters, often producing compact, spherical clusters. Results are typically visualized with a **dendrogram**, a tree diagram representing the hierarchy of clusters.

5.2.3 Partitioning Methods

Partitioning clustering methods divide a data set into distinct, non-overlapping subsets (clusters) in such a way that objects within the same cluster are more similar to each other than to those in other clusters. Here are the primary partitioning methods:

5.2.3.1 K-Means and K-Centroid Clustering

K-Means Clustering:

- **Objective**: Minimize the sum of squared distances between each data point and its assigned cluster centroid.
- **Process**:
    1. **Initialization**: Select kkk initial centroids randomly from the data points.
    2. **Assignment**: Assign each data point to the nearest centroid, forming kkk clusters.
    3. **Update**: Recalculate centroids as the mean of all data points assigned to each cluster.
    4. **Repeat**: Iterate the assignment and update steps until the centroids stabilize or a maximum number of iterations is reached.
- **Pros**:
    - Simple and fast, especially for small to medium-sized data sets.
    - Easy to understand and implement.
- **Cons**:
    - Requires specifying the number of clusters kkk in advance.
    - Sensitive to initial centroid placement; may converge to local minima.
    - Assumes spherical clusters and equal variance.
    - Not suitable for clusters of different shapes and sizes.

K-Centroid Clustering:

- **Variation**: Similar to K-Means, but can use different distance metrics (e.g., Manhattan distance) and may involve alternative strategies for centroid computation, such as using medians instead of means.
- **Pros**:
    - Can be adapted to different distance metrics and cluster shapes.
- **Cons**:
    - Still requires specifying the number of clusters kkk.

5.2.3.2 "Improved" K-Means or Enhanced K-Means:

- **Initialization Improvements**:
    - **K-Means++**: A smarter initialization technique that selects initial centroids in a way that improves convergence and cluster quality by spreading them out.
- **Algorithm Variants**:
    - **Fuzzy C-Means**: A variation where each data point can belong to multiple clusters with varying degrees of membership.
    - **Kernel K-Means**: Applies the kernel trick to handle non-linearly separable clusters by transforming the data into a higher-dimensional space.
- **Pros**:
    - Improved convergence and better results compared to standard K-Means.
    - Can handle more complex clustering problems.
- **Cons**:
    - Still requires specifying kkk, and computational complexity can be higher than standard K-Means.

5.2.3.3 Hard Competitive Learning

- **Concept**: Uses competitive learning algorithms where neurons or clusters compete to represent data points.
- **Process**:
    1. **Initialization**: Initialize the weights of the neurons or clusters.
    2. **Competition**: Each neuron or cluster competes to be the best match for an input data point.
    3. **Update**: The winning neuron or cluster updates its weights to become more similar to the input data.
- **Pros**:
    - Can learn complex, non-linear decision boundaries.
    - Useful for unsupervised learning and clustering.
- **Cons**:
    - The training process can be sensitive to initial conditions and learning rates.
    - Requires careful tuning of parameters.

5.2.3.4 Neural Gas and Topology Representing Network

- **Concept**: An unsupervised learning algorithm that adapts the positions of prototypes (neurons) to model the data distribution.
- **Process**:
    1. **Initialization**: Initialize the prototypes randomly or based on data.
    2. **Adaptation**: Update the prototypes based on input data using a learning rate and a neighborhood function.
- **Pros**:
    - Can model complex data distributions.
    - Adapts the topology of the network based on data.
- **Cons**:
    - Computationally intensive.
    - Requires tuning of parameters such as learning rate and neighbourhood function.

Topology Representing Network (TRN):

- **Concept**: Uses a network of neurons to represent and model the topological relationships between data points.
- **Process**:
    1. **Initialization**: Set up a network of neurons with a predefined topology.
    2. **Training**: Adjust the network based on input data to preserve the topological relationships.
- **Pros**:
    - Preserves the topological structure of the input space.
    - Can handle complex data relationships.
- **Cons**:
    - Complex to implement and requires careful design of the network topology.

5.52.3.5 Self-Organizing Maps (SOMs)

- **Concept**: An unsupervised neural network algorithm that maps high-dimensional data onto a lower-dimensional grid.
- **Process**:
    1. **Initialization**: Initialize a grid of neurons with random weights.
    2. **Training**: For each input data point, find the best-matching neuron (BMU), update the weights of the BMU and its neighbours to be more similar to the input.
- **Pros**:
    - Visualizes high-dimensional data in lower dimensions.

    - Preserves the topological and metric relationships of the data.
- **Cons**:
    - Requires careful tuning of learning rate and neighbourhood function.
    - The grid size and shape need to be determined in advance.

5.2.3.6 Neural Networks for Clustering

- **Concept**: Uses various neural network architectures to perform clustering and learn data representations.
- **Examples**:
    - **Autoencoders**: Learn compressed representations of data, which can then be clustered.
    - **Restricted Boltzmann Machines (RBMs)**: Learn features from data and perform clustering based on these features.
- **Pros**:
    - Can learn complex, non-linear relationships in the data.
    - Flexible and can be adapted to different types of data.
- **Cons**:
    - Requires extensive training and tuning of network parameters.
    - Computationally intensive.

Partitioning methods are diverse and each has specific strengths and weaknesses. The choice of method depends on factors such as the size of the data set, the desired number of clusters, and the nature of the data (e.g., linear vs. non-linear relationships). In practice, a combination of methods and careful parameter tuning is often used to achieve optimal clustering results.

5.52.4 Hybrid Approaches

Hybrid methods combine different clustering techniques to leverage their strengths and address limitations.

5.2.4.1 Two-Step Clustering:
- **Algorithm**: First, clusters are created using a hierarchical or partitioning method (e.g., K-Means) to generate initial clusters. Then, these clusters are further refined using another method. It often combines large-scale partitioning with detailed clustering.

5.2.4.2 Bagged Clustering
- **Concept**: Applies bootstrapping (resampling) to create multiple clustering solutions from different subsets of data. The final clusters are obtained by aggregating results from these multiple runs, which can improve stability and robustness of the clustering solution.

5.3 Model-Based Methods

Model-based methods for clustering, unlike distance-based methods, rely on statistical models to define and estimate the structure of market segments. Instead of using distances or similarities, these methods assume that the data is generated from a mixture of underlying probability distributions, each corresponding to a different segment. The general idea is to find the best-fit model that explains the data in terms of these underlying distributions.

5.3.1 Finite Mixtures of Distributions

- **Concept**: These models assume that the data is generated from a mixture of several distributions. Each component of the mixture corresponds to a different segment.
- **Properties**:

- **Segment Sizes**: Each segment has a certain size, which can be represented by probabilities associated with each distribution.
- **Segment-Specific Characteristics**: Each segment has specific characteristics that are modelled by the parameters of the distributions.

5.3.1.1 Normal Distributions

- **Description**: For continuous, metric data, finite mixtures of multivariate normal distributions are commonly used.
  - **Multivariate Normal Distribution**: Models the joint distribution of several variables, capturing correlations between them.
  - **Applications**: Useful for modelling data with natural correlations, such as body measurements or market prices.
- **Example**: Segmenting consumers based on physical measurements (height, arm length, etc.) where these measurements are correlated.

5.3.1.2 Binary Distributions

- **Description**: For binary data (where variables are 0 or 1), finite mixtures of binary distributions are used, often referred to as latent class models.
  - **Latent Class Models**: Identify unobserved (latent) subgroups within the data, with each group having different probabilities for binary outcomes.
  - **Applications**: Useful for segmentation based on activities or behaviours (e.g., whether a consumer engages in certain activities).
- **Example**: Segmenting tourists based on whether they participate in activities like skiing or sightseeing.

5.3.2 Finite Mixtures of Regression Models

- **Description**: This approach extends finite mixture models by including regression, where the dependent variable is modelled based on independent variables for each segment.
  Segment-Specific Relationships: Assumes that the relationship between the dependent variable and the independent variables varies across segments.
- **Applications**: Useful for market segmentation where the goal is to understand how different factors (like spending on various categories) influence a target variable differently across segments.
- **Example**: Analysing consumer spending behavior where the effect of income on spending varies by market segment.

5.3.3 Extensions and Variations

- **Mixtures of Multinomial Distributions**: Used for categorical data where variables can take on more than two values.
  Applications: Useful for segmenting based on preferences or choices among multiple categories.

- **Mixtures of Ordinal Distributions**: For ordinal data where variables have a meaningful order but not a numeric scale.
  Applications: Handling responses that are ranked but not quantified (e.g., satisfaction ratings).

- **Mixtures of Mixed-Effects Models**: These models combine fixed effects and random effects to account for both segment-specific and within-segment variations.
  Applications: Useful in scenarios where there is both variation between segments and variation within segments.

- **Dynamic Latent Change Models**: Used for time-series data to model changes in segments over time.
  Applications: Tracking changes in consumer behavior or brand preferences over time.

- **Markov Chains**: Used to model transitions between segments over time.
  Applications: Understanding how consumer behavior changes, such as brand switching.

- **Concomitant Variables**: These are additional variables included to model differences in segment sizes based on external descriptors.
  Applications: Adjusting segment sizes based on characteristics like age or income.

Model-based methods for clustering provide a robust framework for segmenting data, particularly when the data exhibits complex structures or when traditional distance-based methods fall short. They offer flexibility in modelling different types of data (continuous, binary, ordinal) and can incorporate additional complexity like temporal changes or mixed effects. The choice of method depends on the nature of the data and the specific goals of the segmentation analysis.

5.4 Algorithms with Integrated Variable Selection

In data segmentation tasks, algorithms generally assume that all variables contribute to determining the segments. However, this assumption may not always hold true, especially when variables are redundant or noisy. Pre-processing methods can help identify and remove such variables, but when dealing with binary data, pre-filtering becomes challenging. For binary segmentation variables, methods need to integrate variable selection within the segmentation process. Here, we discuss three key approaches: Biclustering, Variable Selection for Clustering Binary Data (VSBD), and Factor-Cluster Analysis.

5.4.1 Biclustering Algorithms

Biclustering algorithms aim to simultaneously cluster both observations and variables. This is particularly useful for binary data, where the goal is to find subsets of variables that simultaneously group consumers based on shared characteristics.

- **Concept**: Biclustering identifies groups of consumers and variables that exhibit similar patterns. For binary data, a Bicluster consists of consumers who share a value of 1 across a subset of variables.
- **Procedure**:
  1. **Rearrange Data Matrix**: Adjust rows and columns to create a rectangle of 1s at the top-left, maximizing its size.
  2. **Assign to Bicluster**: Assign observations within this rectangle to one bicluster, with the variables defining the rectangle being active.
  3. **Iterate**: Remove assigned rows and repeat until no more significant Biclusters can be found.
- **Advantages**:
  - **No Data Transformation**: Preserves original data.
  - **Niche Market Identification**: Effective for identifying highly specific market segments.
- **Challenges**: Not all consumers are grouped, and large numbers of variables can make the process complex.

5.4.2 Variable Selection Procedure for Clustering Binary Data (VSBD)

VSBD, proposed by Brusco (2004), integrates variable selection into the clustering process using k-means clustering. The goal is to identify and remove irrelevant variables that do not contribute to the clustering solution.

- **Procedure**:
    1. **Subset Selection**: Select a subset of observations to work with, based on data size.
    2. **Exhaustive Search**: Find the subset of variables that minimizes the within-cluster sum-of-squares (WCSS).
    3. **Incremental Addition**: Add variables one by one, selecting the one that minimally increases WCSS.
    4. **Thresholding**: Stop adding variables when the increase in WCSS exceeds a pre-set threshold.
- **Advantages**: Helps in removing irrelevant variables, simplifying the clustering process.
- **Challenges**: Computationally intensive, especially with large datasets.

5.4.3 Factor-Cluster Analysis

Factor-Cluster Analysis involves a two-step process where segmentation variables are first reduced using factor analysis, and then segments are extracted using these factor scores.

- **Procedure**:
    1. **Factor Analysis**: Reduce the number of original variables by identifying underlying factors.
    2. **Segmentation**: Perform clustering based on the factor scores instead of the raw variables.
- **Advantages**:
    - **Variable Reduction**: Useful when dealing with a high number of variables.
- **Challenges**:
    - **Information Loss**: Significant information may be lost in the factor analysis process.
    - **Interpretation**: Results are harder to interpret as factors are combinations of original variables.
    - **Empirical Evidence**: Studies have shown that factor-cluster analysis may not always outperform clustering with raw data.

In summary, these methods offer different approaches to integrating variable selection into the clustering process. Biclustering and VSBD focus on improving the segmentation by addressing variable relevance directly, while Factor-Cluster Analysis provides a way to handle large sets of variables but at the cost of potential information loss.

5.5 Data Structure Analysis

Data Structure Analysis in market segmentation is primarily exploratory, focusing on assessing the reliability or stability of segmentation solutions. Unlike traditional validation that targets a clear optimality criterion, validation in this context revolves around assessing the stability of solutions across variations in data or algorithms. This type of validation is termed **stability-based data structure analysis**.

Stability-Based Data Structure Analysis:

- **Purpose:** Determine whether natural, distinct, and well-separated market segments exist in the data. This involves evaluating whether the results are consistent when data or algorithms are slightly altered.
- **Approach:** Instead of comparing against an external criterion, the focus is on stability and reproducibility of segments across different conditions.

Cluster Indices:

- **Internal Cluster Indices:**
    - **Definition:** Calculated based on a single segmentation solution using information contained within that solution.
    - **Examples:**
        - **Compactness:** Measures how similar members within a segment are. For instance, the sum of all distances between pairs of segment members.

        - **Separation:** Evaluates how distinct different segments are from each other.

- **External Cluster Indices:**
    - **Definition:** Requires additional segmentation solutions for comparison. It measures the similarity between different segmentation outcomes.
    - **Examples:** Jaccard index, Rand index, and adjusted Rand index, which assess how similar the segments are across different segmentation solutions.

Gorge Plots:

- **Purpose:** Visualize how well segments are separated by examining the distances between consumers and segment representatives (e.g., centroids).
- **Methodology:**
    - Calculate similarity values.
    - High similarity values suggest a consumer is close to the segment representative, while low values suggest the opposite.
- **Visualization:** Gorge plots, silhouette plots, or shadow plots, displaying the distribution of similarity values.

Global Stability Analysis:

- **Purpose:** Evaluate the stability of market segmentation solutions across multiple resampling iterations.
- **Approach:** Generate several new datasets using resampling methods and compare the stability of segmentation solutions across these iterations.
- **Importance:** Helps determine whether market segments are consistent and stable or if they vary significantly across different samples.

Segment Level Stability Analysis:

- **Segment Level Stability Within Solutions (SLSW):**
    - **Definition:** Assesses stability of individual segments within a given segmentation solution. Ensures that potentially valuable segments are not discarded due to overall poor stability.
    - **Importance:** Allows the identification of stable individual segments even if the overall solution is not globally stable.

- **Segment Level Stability Across Solutions (SLSA):**
    - **Definition:** Evaluates how consistently a market segment appears across different segmentation solutions with varying numbers of segments.
    - **Purpose:** Identifies whether segments are naturally occurring or artificially created.
    - **Method:** Compares segment re-occurrence across different segment numbers, often requiring consistent labelling of segments across solutions.

Data structure analysis in market segmentation involves assessing the stability and reliability of segmentations through both internal and external indices. Gorge plots provide a way to visualize segment separation, while global and segment-level stability analyses offer insights into the consistency of segmentation solutions. Understanding these aspects helps in making informed decisions about segment extraction and ensuring that identified segments are meaningful and actionable.

5.6 Checklist for Market Segmentation

To ensure a thorough and effective market segmentation analysis, follow this checklist:

1. **Pre-Selection of Extraction Methods:**
    - **Assess Data Properties:** Determine the nature of your data (e.g., quantitative, qualitative, structured, unstructured).
    - **Choose Suitable Methods:** Select extraction methods that align with the data properties. Options may include hierarchical clustering, k-means, model-based methods, or other relevant algorithms.

2. **Group Consumers:**
    - **Apply Extraction Methods:** Use the chosen methods to perform the segmentation and group consumers into market segments based on the extracted patterns.

3. **Conduct Stability Analyses:**
    - **Global Stability Analysis:**
        - Perform resampling methods to assess the consistency of segmentation solutions across different datasets.
    - **Segment Level Stability Analysis:**
        - **Segment Level Stability Within Solutions (SLSW):** Evaluate stability of individual segments within each solution.
        - **Segment Level Stability Across Solutions (SLSA):** Assess the re-occurrence and consistency of segments across different numbers of segments and methods.

4. **Select Promising Segments:**
    - **Evaluate Stability:** From the segmentation solutions, select those segments that exhibit strong segment-level stability.
    - **Filter Segments:** Choose the most promising market segments based on their stability and potential usefulness.

5. **Apply Knock-Out Criteria:**
    - **Define Criteria:** Utilize predefined criteria (e.g., segment size, profitability, alignment with strategic goals) to assess the remaining segments.
    - **Filter Out Unpromising Segments:** Remove segments that do not meet the knock-out criteria, ensuring only viable segments proceed.

6. **Detailed Profiling:**
    - **Pass Final Segments to Profiling:** Transfer the remaining set of promising segments to Step 6 for in-depth profiling, which will involve further analysis and understanding of each segment's characteristics and needs.

This checklist guides you through a systematic process of market segmentation, ensuring that you use appropriate methods, assess stability thoroughly, and select and profile segments effectively. Each step builds upon the previous one to refine and validate the market segments for strategic decision-making.

Step 6: Profiling Segments

Introduction

Profiling market segments is a crucial step in understanding and interpreting the results of market segmentation. It involves identifying and characterizing the defining attributes of market segments that were extracted during the segmentation process. This report outlines the key aspects of profiling, including traditional approaches, the role of visualizations, and practical steps for effective profiling.

Identifying Key Characteristics of Market Segments

Profiling aims to reveal the defining characteristics of market segments, especially when data-driven segmentation is used. Unlike commonsense segmentation, where profiles are predefined (e.g., age groups), data-driven segmentation requires detailed analysis to understand segment characteristics.

Profiling involves:

- **Characterizing Segments Individually:** Each segment is examined to identify its unique features based on segmentation variables.
- **Comparing Segments:** It is essential to compare segments to determine what differentiates them from one another. For example, while alpine skiing may characterize a segment, it may not distinguish it from other segments if multiple segments have similar characteristics.

Effective profiling provides a foundation for interpreting segmentation results accurately, which is critical for making informed strategic marketing decisions. However, data-driven market segmentation can be challenging to interpret, as evidenced by studies indicating that many managers find such results difficult to understand and liken them to a "black box."

Traditional Approaches to Profiling Market Segments

Traditional approaches to presenting data-driven segmentation results often face two main challenges:

1. **Simplified Summaries:** High-level summaries may overly simplify segment characteristics, leading to misleading conclusions.
2. **Complex Tables:** Large tables with detailed percentages for each segmentation variable can be difficult to interpret and do not provide a quick overview of key insights. Additionally, using statistical significance tests to assess differences between segments is often inappropriate, as segments are designed to be maximally different based on segmentation variables.

Segment Profiling with Visualizations

Visualizations play a vital role in enhancing the interpretability of market segmentation results. Despite their importance, graphical representations are often underutilized in traditional profiling methods.
Key points include:

- **Importance of Graphics:** Graphics are integral to statistical data analysis, helping to reveal complex relationships between variables. They are particularly useful in exploratory analysis, such as cluster analysis, and can simplify monitoring of developments over time.

- **Benefits of Visualization:** According to various studies and reviews, visualizations can make segmentation results more accessible and insightful. For instance, graphical methods are recommended to represent segment profiles clearly and intuitively. Visualizations can help in:
    - **Identifying Defining Characteristics:** Graphical representations can illustrate the distinct features of each segment.
    - **Assessing Segment Separation:** Visual tools can aid in evaluating how well-segmented the market is, ensuring that segments are distinct and meaningful.

Examples of effective visualizations in market segmentation have been provided in various studies and literature, demonstrating their utility in enhancing the understanding of segmentation solutions.

Checklist for Profiling Market Segments

To conduct effective profiling, follow these steps:

1. **Use Selected Segments:** Start with the segments identified in the previous segmentation step.
2. **Visualize Segment Profiles:** Create visualizations to explore and understand the distinct characteristics of each segment.
3. **Apply Knock-out Criteria:** Assess whether any segments should be eliminated based on predefined criteria to ensure relevance and quality.
4. **Proceed to Description:** Pass the remaining, validated segments to the next step for detailed description and analysis.

Conclusion

Profiling market segments is essential for understanding the results of data-driven segmentation. By identifying defining characteristics, comparing segments, and utilizing visualizations, businesses can gain valuable insights into their market segments. Traditional approaches, while useful, may not fully capture the complexity of the segments, highlighting the need for effective graphical representation. Following a structured profiling process ensures that segmentation results are accurately interpreted and strategically valuable.

Case Study: Fast food (McDonalds)

Step 1: Deciding (not) to Segment

There is value in investigating systematic heterogeneity among consumers and harvest these differences using a differentiated marketing strategy.

Step 2: Specifying the Ideal Target Segment

In terms of knock-out criteria, the target segment or target segments must be homogeneous, distinct, large enough to justify the development and implementation of a customised marketing mix, matching the strengths of McDonald's, identifiable and reachable. In terms of segment attractiveness criteria, the obvious choice would be a segment that has a positive perception of McDonald's, frequently eats out and likes fast food. But McDonald's management could also decide that they not only wish to solidify their position in market segments in which they already hold high market shares, but rather wish to learn more about market segments which are currently not fond of McDonald's.

In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
```

Step 3: Collecting Data

In [8]:
```
df = pd.read_csv(r"C:\Users\gyanp\Downloads\mcdonalds.csv")
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['yummy', 'convenient', 'spicy', 'fattening', 'greasy', 'fast', 'cheap',
       'tasty', 'expensive', 'healthy', 'disgusting', 'Like', 'Age',
       'VisitFrequency', 'Gender'],
      dtype='object')
```

The data set contains responses from 1453 adult Australian consumers relating to their perceptions of McDonald's with respect to the following attributes: YUMMY, CONVENIENT, SPICY, FATTENING, GREASY, FAST, CHEAP, TASTY, EXPENSIVE, HEALTHY,

and DISGUSTING. For each of those attributes, respondents provided either a YES, or a NO response .

Step 4: Exploring Data

Explore the key characteristics of the data set by loading the data set and inspecting basic features such as the variable names, the sample size, and the first three rows of the data

In [13]:
```
df.shape
```

Out[13]:
```
(1453, 15)
```

In [14]:
```
df.head(3)
```

Out[14]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy
0	No	Yes	No	Yes	No	Yes	Yes	No	Yes	No
1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No
2	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes

In [15]:
```
# Select columns 1 to 11 (0-indexed in Python, so columns 0 to 10)
MD_x = df.iloc[:, 0:11]

# Convert "Yes"/"No" to binary (1/0)
MD_x = (MD_x == "Yes").astype(int)

# Calculate column means and round to 2 decimal places
col_means = MD_x.mean().round(2)

print(col_means)
```

```
yummy         0.55
convenient    0.91
spicy         0.09
fattening     0.87
greasy        0.53
fast          0.90
cheap         0.60
tasty         0.64
expensive     0.36
healthy       0.20
disgusting    0.24
dtype: float64
```

Step 5: Extracting Segments

In [17]:

```python
from sklearn.decomposition import PCA

# Perform PCA
pca = PCA()
pca.fit(MD_x)

# Print explained variance ratio with 1 decimal place
explained_variance_ratio = pca.explained_variance_ratio_
components = pca.components_
formatted_explained_variance = [f"{x:.1f}" for x in explained_variance_ratio]

print("\nPCA Components:")
print(components)

print("\nExplained Variance Ratio:")
print(formatted_explained_variance)

# Print PCA components with 1 decimal place
components = pca.components_
formatted_components = pd.DataFrame(components).applymap(lambda x: f"{x:.1f}")

print("\nPCA Components:")
print(formatted_components)
```

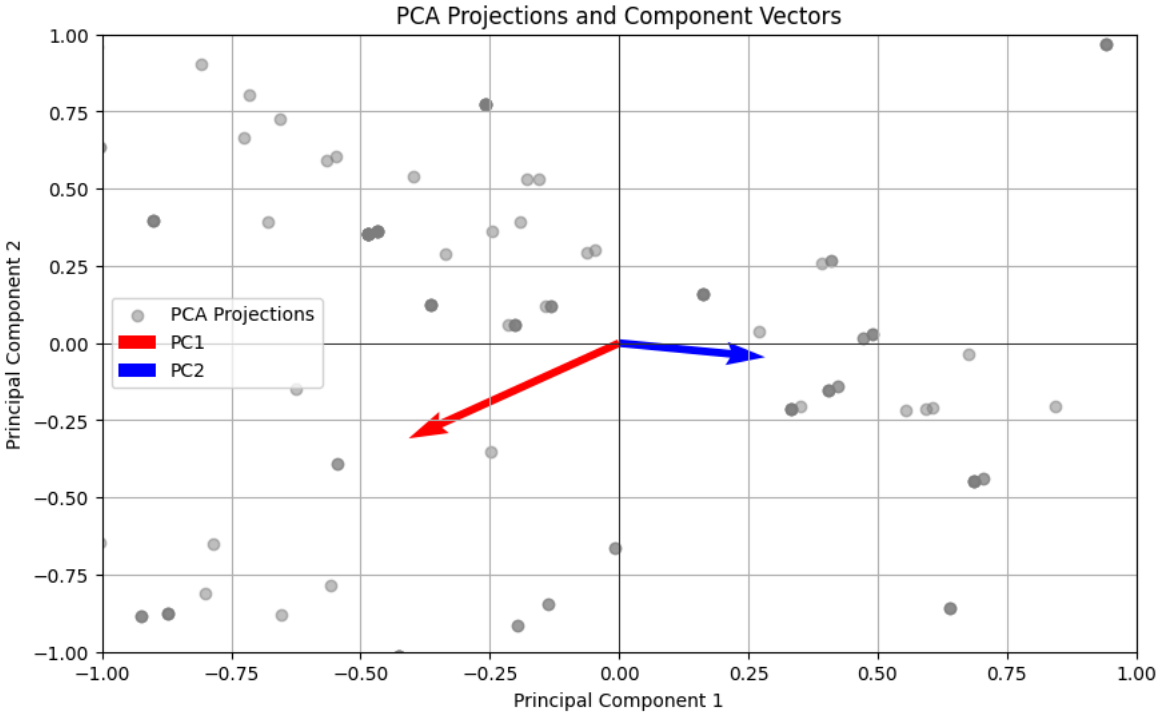
```
PCA Components:
[[-0.47693349 -0.15533159 -0.00635636  0.11623168  0.3044427  -0.10849325
  -0.33718593 -0.47151394  0.32904173 -0.21371062  0.37475293]
 [ 0.36378978  0.016414    0.01880869 -0.03409395 -0.06383884 -0.0869722
  -0.61063276  0.3073178   0.60128596  0.07659344 -0.13965633]
 [-0.30444402 -0.0625153  -0.03701866 -0.32235949 -0.80237317 -0.06464172
  -0.14931026 -0.28726479  0.02439661  0.19205128 -0.08857138]
 [ 0.0551622  -0.14242496  0.19761927 -0.35413876  0.2539601  -0.09736269
   0.11895823 -0.00254696  0.06781599  0.76348804  0.36953871]
 [-0.30753507  0.27760805  0.07062017 -0.07340475  0.36139895  0.10793025
  -0.12897259 -0.21089912 -0.00312457  0.28784553 -0.72920859]
 [ 0.17073819 -0.34783006 -0.3550866  -0.40651542  0.20934711 -0.59463206
  -0.10324067 -0.07691443 -0.26134151 -0.17822612 -0.21087805]
 [-0.28051863 -0.05973793  0.70763705 -0.38594277  0.03616962 -0.08684577
  -0.04044934  0.36045348 -0.06838452 -0.34961569 -0.02679159]
 [ 0.01304117 -0.11307868  0.37593402  0.58962241 -0.13824084 -0.62779877
   0.14006047 -0.07279193  0.02953939  0.17630281 -0.16718101]
 [ 0.57240278 -0.01846534  0.40027977 -0.16051227 -0.00284738  0.16619659
   0.07606907 -0.63908592  0.06699639 -0.1855722  -0.07248255]
 [-0.11028437 -0.66581756 -0.07563413 -0.00533813  0.00870725  0.23953197
   0.42808739  0.0791838   0.45439925 -0.03811713 -0.28959188]
 [ 0.04543901 -0.54161635  0.14172992  0.25090987  0.00164229  0.33926454
  -0.48928285  0.01955226 -0.49006853  0.15760765 -0.04066227]]

Explained Variance Ratio:
['0.3', '0.2', '0.1', '0.1', '0.1', '0.1', '0.0', '0.0', '0.0', '0.0', '0.0']

PCA Components:
```

	0	1	2	3	4	5	6	7	8	9	10
0	-0.5	-0.2	-0.0	0.1	0.3	-0.1	-0.3	-0.5	0.3	-0.2	0.4
1	0.4	0.0	0.0	-0.0	-0.1	-0.1	-0.6	0.3	0.6	0.1	-0.1
2	-0.3	-0.1	-0.0	-0.3	-0.8	-0.1	-0.1	-0.3	0.0	0.2	-0.1
3	0.1	-0.1	0.2	-0.4	0.3	-0.1	0.1	-0.0	0.1	0.8	0.4
4	-0.3	0.3	0.1	-0.1	0.4	0.1	-0.1	-0.2	-0.0	0.3	-0.7
5	0.2	-0.3	-0.4	-0.4	0.2	-0.6	-0.1	-0.1	-0.3	-0.2	-0.2
6	-0.3	-0.1	0.7	-0.4	0.0	-0.1	-0.0	0.4	-0.1	-0.3	-0.0
7	0.0	-0.1	0.4	0.6	-0.1	-0.6	0.1	-0.1	0.0	0.2	-0.2
8	0.6	-0.0	0.4	-0.2	-0.0	0.2	0.1	-0.6	0.1	-0.2	-0.1
9	-0.1	-0.7	-0.1	-0.0	0.0	0.2	0.4	0.1	0.5	-0.0	-0.3
10	0.0	-0.5	0.1	0.3	0.0	0.3	-0.5	0.0	-0.5	0.2	-0.0

```
C:\Users\gyanp\AppData\Local\Temp\ipykernel_15316\626742030.py:20: FutureWarning:
DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  formatted_components = pd.DataFrame(components).applymap(lambda x: f"{x:.1f}")
```

In [18]:
```python
# Standardize the data (important for PCA)
scaler = StandardScaler()
MD_x_scaled = scaler.fit_transform(MD_x)

# Perform PCA
pca = PCA()
pca.fit(MD_x_scaled)

# Project data onto principal components
MD_pca = pca.transform(MD_x_scaled)

# Create the plot
plt.figure(figsize=(10, 6))

# Plot PCA projections
plt.scatter(MD_pca[:, 0], MD_pca[:, 1], color="grey", alpha=0.5, label='PCA Proj
```

```
# Plot PCA component vectors
plt.quiver(0, 0, pca.components_[0, 0], pca.components_[0, 1], angles='xy', scal
plt.quiver(0, 0, pca.components_[1, 0], pca.components_[1, 1], angles='xy', scal

# Set limits and labels
plt.xlim(-1, 1)
plt.ylim(-1, 1)
plt.title("PCA Projections and Component Vectors")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.grid(True)
plt.legend()

# Show the plot
plt.show()
```

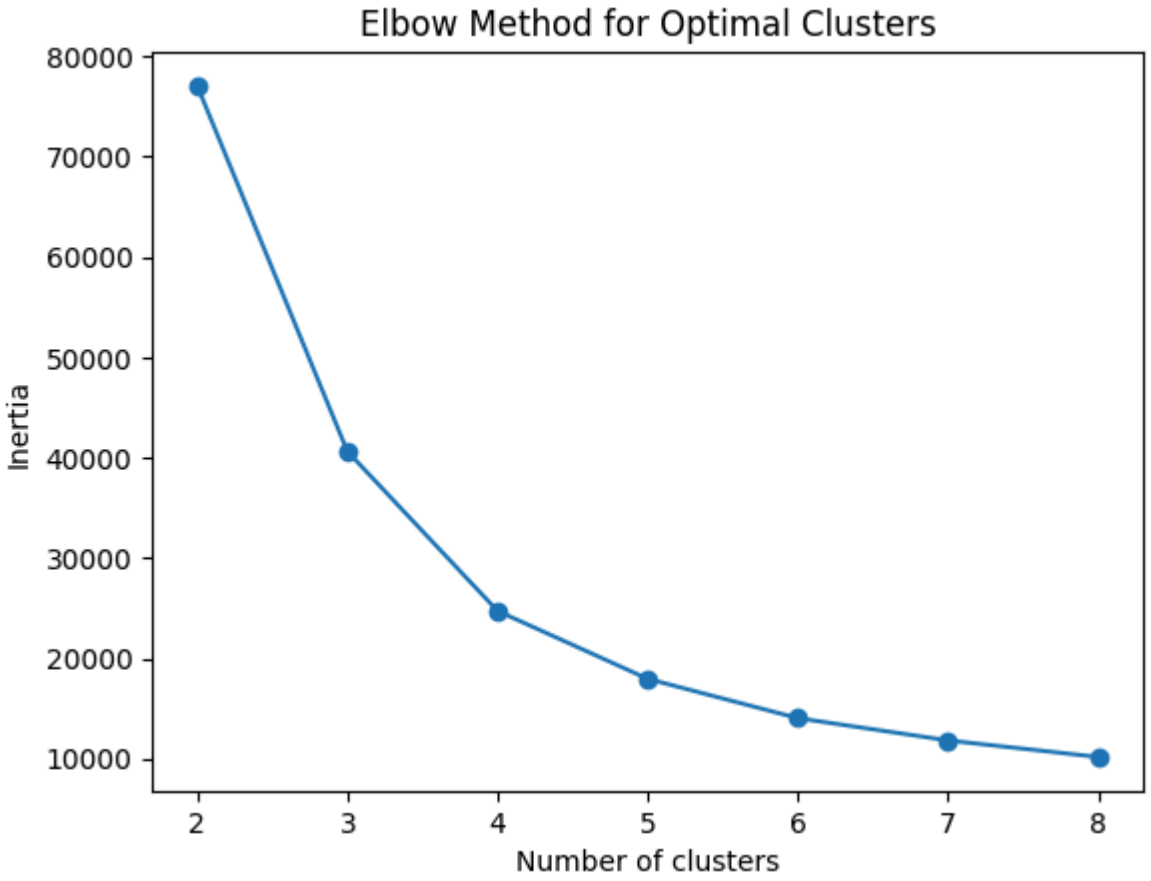


Using k-Means

In [20]:
```
from sklearn.cluster import KMeans

np.random.seed(1234)

# Assuming you have already encoded categorical variables
df_encoded = pd.get_dummies(df, drop_first=True)

kmeans_models = {}
for n_clusters in range(2, 9):
    kmeans = KMeans(n_clusters=n_clusters, n_init=10, random_state=1234)
    kmeans.fit(df_encoded)
    kmeans_models[n_clusters] = kmeans
```

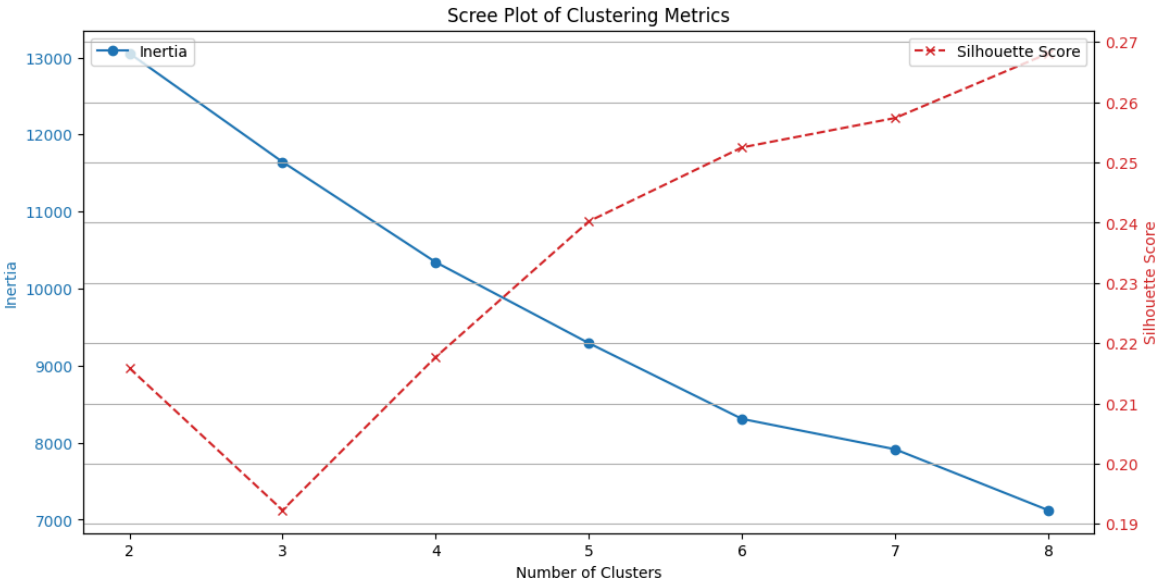
In [21]:
```
import matplotlib.pyplot as plt

inertia = [kmeans_models[n].inertia_ for n in range(2, 9)]
```

```
plt.plot(range(2, 9), inertia, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal Clusters')
plt.show()
```



In [22]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Set random seed
np.random.seed(1234)

# Standardize the data
scaler = StandardScaler()
MD_x_scaled = scaler.fit_transform(MD_x)

# Initialize lists to store metrics
n_clusters_list = list(range(2, 9))
inertia_list = []
silhouette_scores = []

# Perform k-means clustering and collect metrics
for n_clusters in n_clusters_list:
    kmeans = KMeans(n_clusters=n_clusters, n_init=10, random_state=1234)
    kmeans.fit(MD_x_scaled)

    # Store inertia (sum of squared distances of samples to their closest cluste
    inertia_list.append(kmeans.inertia_)

    # Calculate silhouette score
    labels = kmeans.labels_
```

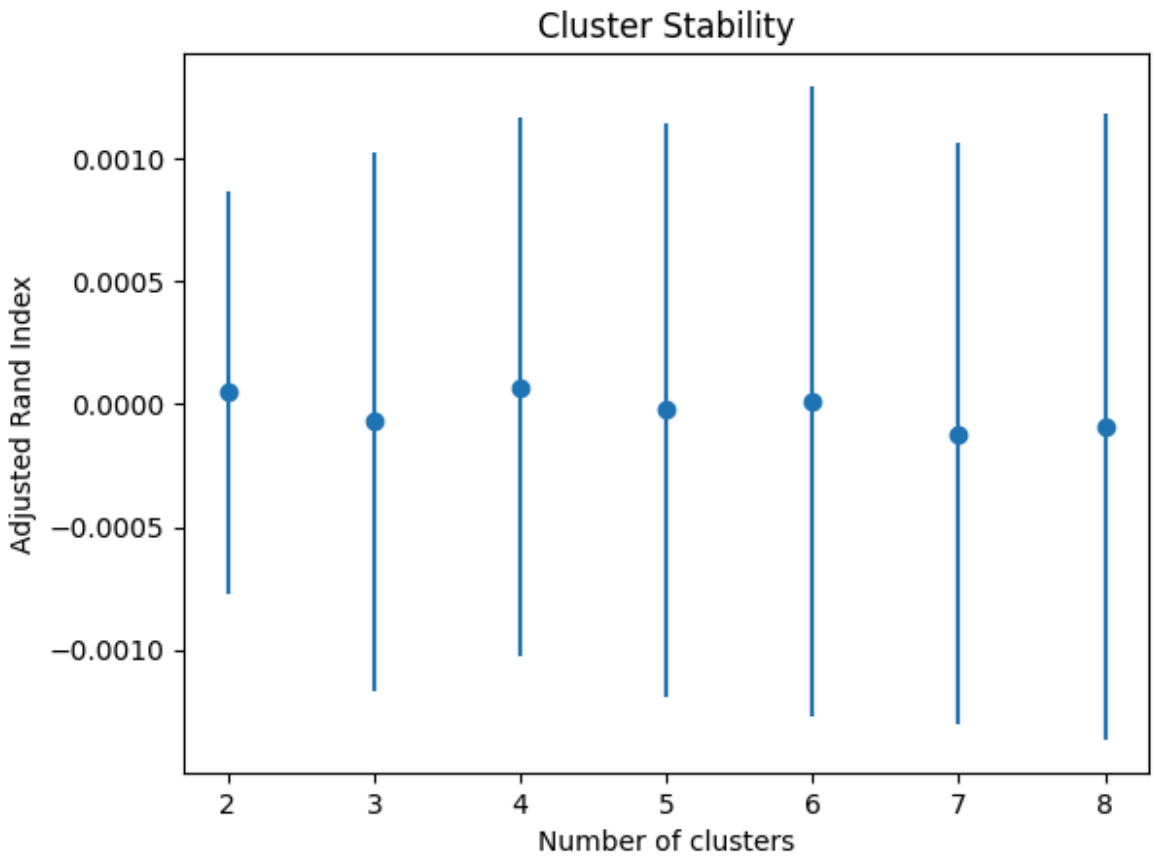
```
    silhouette_avg = silhouette_score(MD_x_scaled, labels) if len(set(labels)) >
    silhouette_scores.append(silhouette_avg)

# Create the scree plot
fig, ax1 = plt.subplots(figsize=(12, 6))

# Plot inertia
color = 'tab:blue'
ax1.set_xlabel('Number of Clusters')
ax1.set_ylabel('Inertia', color=color)
ax1.plot(n_clusters_list, inertia_list, color=color, marker='o', label='Inertia'
ax1.tick_params(axis='y', labelcolor=color)
ax1.legend(loc='upper left')

# Create a second y-axis for silhouette scores
ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Silhouette Score', color=color)
ax2.plot(n_clusters_list, silhouette_scores, color=color, marker='x', linestyle=
ax2.tick_params(axis='y', labelcolor=color)
ax2.legend(loc='upper right')

# Add title and grid
plt.title('Scree Plot of Clustering Metrics')
plt.grid(True)
plt.show()
```



In [23]:

```
from sklearn.utils import resample
from sklearn.metrics import adjusted_rand_score

def bootstrap_kmeans(X, n_clusters, nboot=100):
    rand_indices = []
    for _ in range(nboot):
        X_boot = resample(X)
        kmeans_boot = KMeans(n_clusters=n_clusters, random_state=1234).fit(X_boo
        rand_indices.append(adjusted_rand_score(kmeans.labels_, kmeans_boot.labe
    return np.mean(rand_indices), np.std(rand_indices)

stability_scores = {}
for n_clusters in range(2, 9):
    mean_ari, std_ari = bootstrap_kmeans(df_encoded, n_clusters, nboot=100)
    stability_scores[n_clusters] = (mean_ari, std_ari)
```

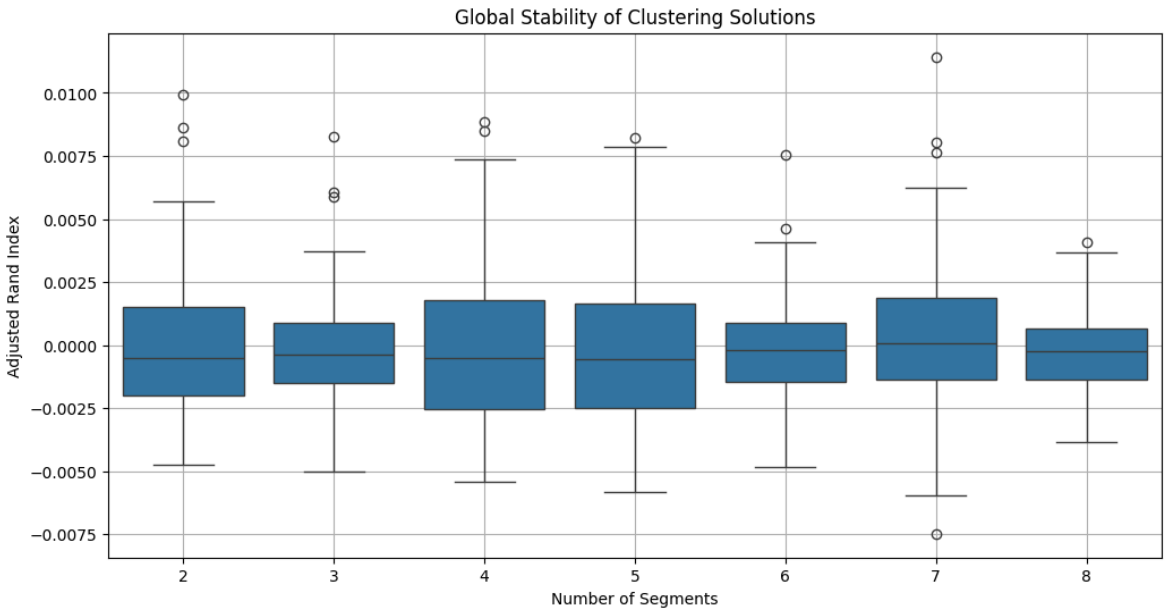
```
# Plot stability scores (adjusted Rand index)
means = [stability_scores[n][0] for n in range(2, 9)]
stds = [stability_scores[n][1] for n in range(2, 9)]

plt.errorbar(range(2, 9), means, yerr=stds, fmt='o')
plt.xlabel('Number of clusters')
plt.ylabel('Adjusted Rand Index')
plt.title('Cluster Stability')
plt.show()
```



In [24]:
```
# Set random seed
np.random.seed(1234)

# Number of bootstrap samples and initializations
n_boot = 100
n_init = 10
n_clusters_list = list(range(2, 9))

# Function to perform k-means clustering and calculate ARI
def compute_ari_for_bootstrap(X, n_clusters, n_boot, n_init):
    ari_scores = []
    for _ in range(n_boot):
        # Create a bootstrap sample
        X_boot = resample(X, random_state=np.random.randint(0, 10000))

        # Perform k-means clustering
        kmeans = KMeans(n_clusters=n_clusters, n_init=n_init, random_state=1234)
        kmeans.fit(X_boot)
        labels_boot = kmeans.labels_

        # Compare with original data clustering
        kmeans_orig = KMeans(n_clusters=n_clusters, n_init=n_init, random_state=
```

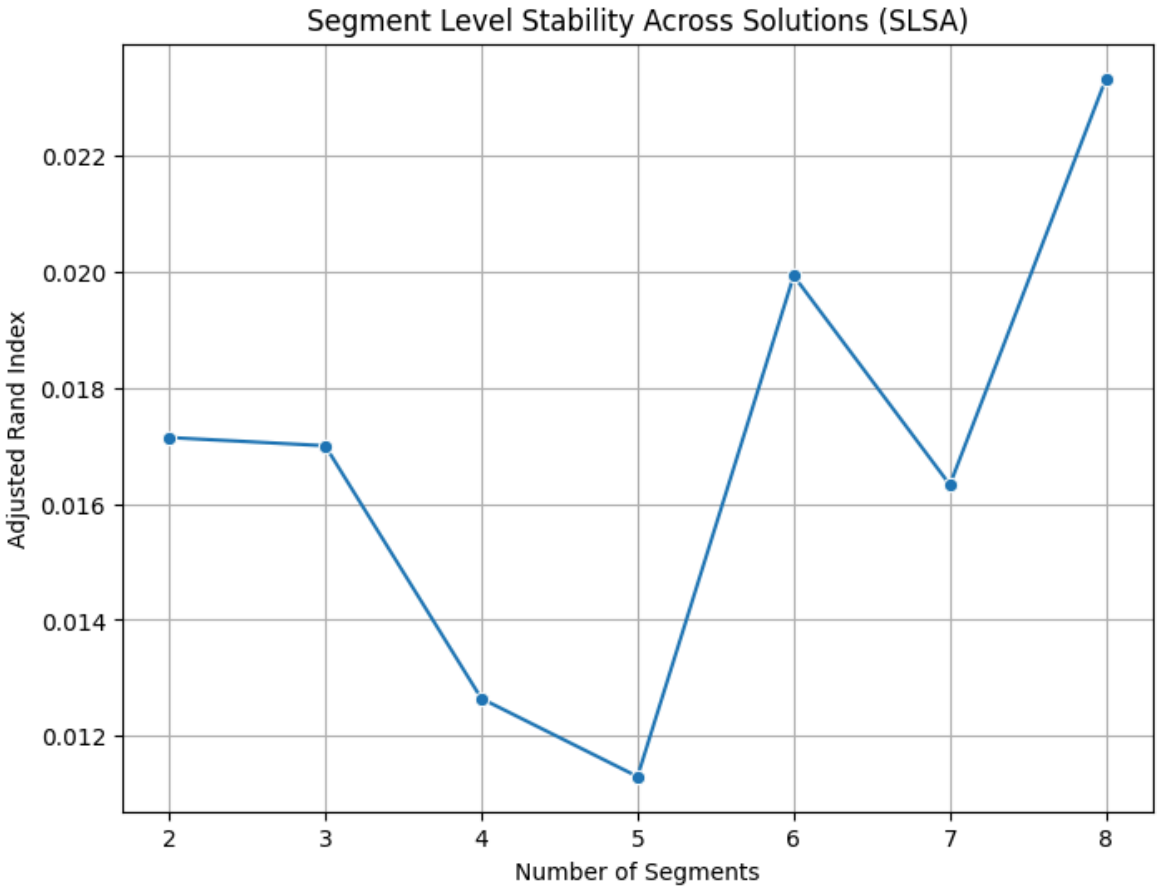
```
        kmeans_orig.fit(X)
        labels_orig = kmeans_orig.labels_

        # Compute Adjusted Rand Index
        ari = adjusted_rand_score(labels_boot, labels_orig)
        ari_scores.append(ari)

    return ari_scores

# Compute ARI for each number of clusters
results = {n_clusters: compute_ari_for_bootstrap(MD_x_scaled, n_clusters, n_boot
```

In [25]:
```
# Prepare data for plotting
ari_df = pd.DataFrame([(n_clusters, ari) for n_clusters, aris in results.items()
                       columns=['Number of Clusters', 'Adjusted Rand Index'])

# Plot boxplot of ARI values
plt.figure(figsize=(12, 6))
sns.boxplot(x='Number of Clusters', y='Adjusted Rand Index', data=ari_df)
plt.xlabel('Number of Segments')
plt.ylabel('Adjusted Rand Index')
plt.title('Global Stability of Clustering Solutions')
plt.grid(True)
plt.show()
```

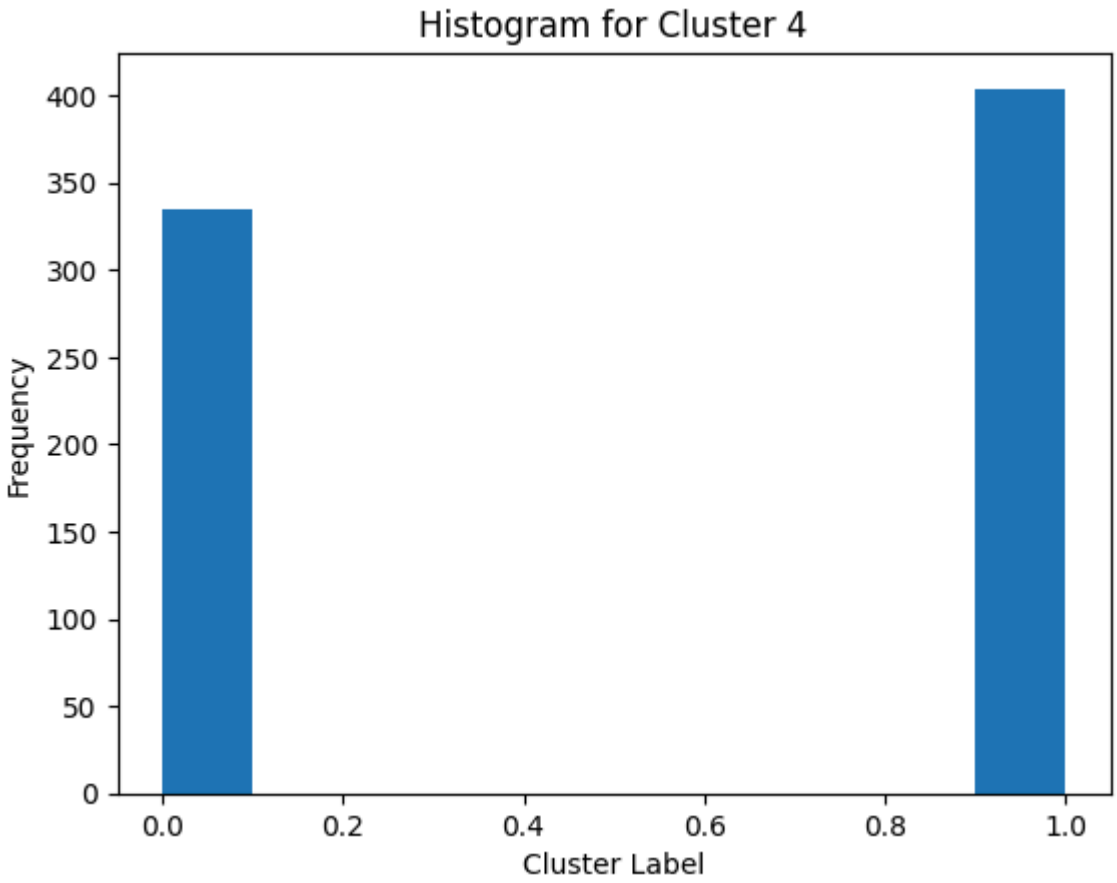


In [37]:
```
# Function to calculate global stability with Adjusted Rand Index
def calculate_stability(data, cluster_range, nrep=10, random_state=1234):
    stability_scores = []
    np.random.seed(random_state)

    for k in cluster_range:
        km = KMeans(n_clusters=k, n_init=nrep, random_state=random_state)
        km.fit(data)
        labels = km.labels_
        score = adjusted_rand_score(labels, kmeans_models[4].labels_)  # Compare
        stability_scores.append(score)

    return stability_scores

# Compare 2 to 8 segment solutions for stability
cluster_range = range(2, 9)
```

```
stability_scores = calculate_stability(MD_x_scaled, cluster_range)

# Plot stability scores
plt.figure(figsize=(8, 6))
sns.lineplot(x=cluster_range, y=stability_scores, marker="o")
plt.xlabel('Number of Segments')
plt.ylabel('Adjusted Rand Index')
plt.title('Segment Level Stability Across Solutions (SLSA)')
plt.grid(True)
plt.show()
```



In [39]:
```
cluster_4_labels = kmeans_models[4].labels_

plt.hist(cluster_4_labels, bins=10, range=(0, 1))
plt.xlabel('Cluster Label')
plt.ylabel('Frequency')
plt.title('Histogram for Cluster 4')
plt.show()
```



In [41]:
```
kmeans_4 = kmeans_models[4]

# Try for other number of clusters also
kmeans_3 = kmeans_models[3]
kmeans_5 = kmeans_models[5]
kmeans_2 = kmeans_models[2]
```

In [42]:
```
from sklearn.metrics import silhouette_score

silhouette_avg = silhouette_score(df_encoded, kmeans_4.labels_)
print(f"Silhouette Score for 4 clusters: {silhouette_avg}")
print("\n")

silhouette_avg = silhouette_score(df_encoded, kmeans_3.labels_)
print(f"Silhouette Score for 3 clusters: {silhouette_avg}")
print("\n")

silhouette_avg = silhouette_score(df_encoded, kmeans_5.labels_)
print(f"Silhouette Score for 5 clusters: {silhouette_avg}")
print("\n")

silhouette_avg = silhouette_score(df_encoded, kmeans_2.labels_)
print(f"Silhouette Score for 2 clusters: {silhouette_avg}")

# Silhoutte score is high for two clusters.
```

```
Silhouette Score for 4 clusters: 0.4864095925289268

Silhouette Score for 3 clusters: 0.5248353793529995

Silhouette Score for 5 clusters: 0.4520301634961908

Silhouette Score for 2 clusters: 0.6107651701790174
```

plot(MD.r4, ylim = 0:1, xlab = "segment number", ylab = "segment stability")

In [44]:
```
kmeans_4.labels_
```

Out[44]:
```
array([0, 2, 0, ..., 2, 1, 3])
```

In [45]:
```
df_encoded.head(3)
```

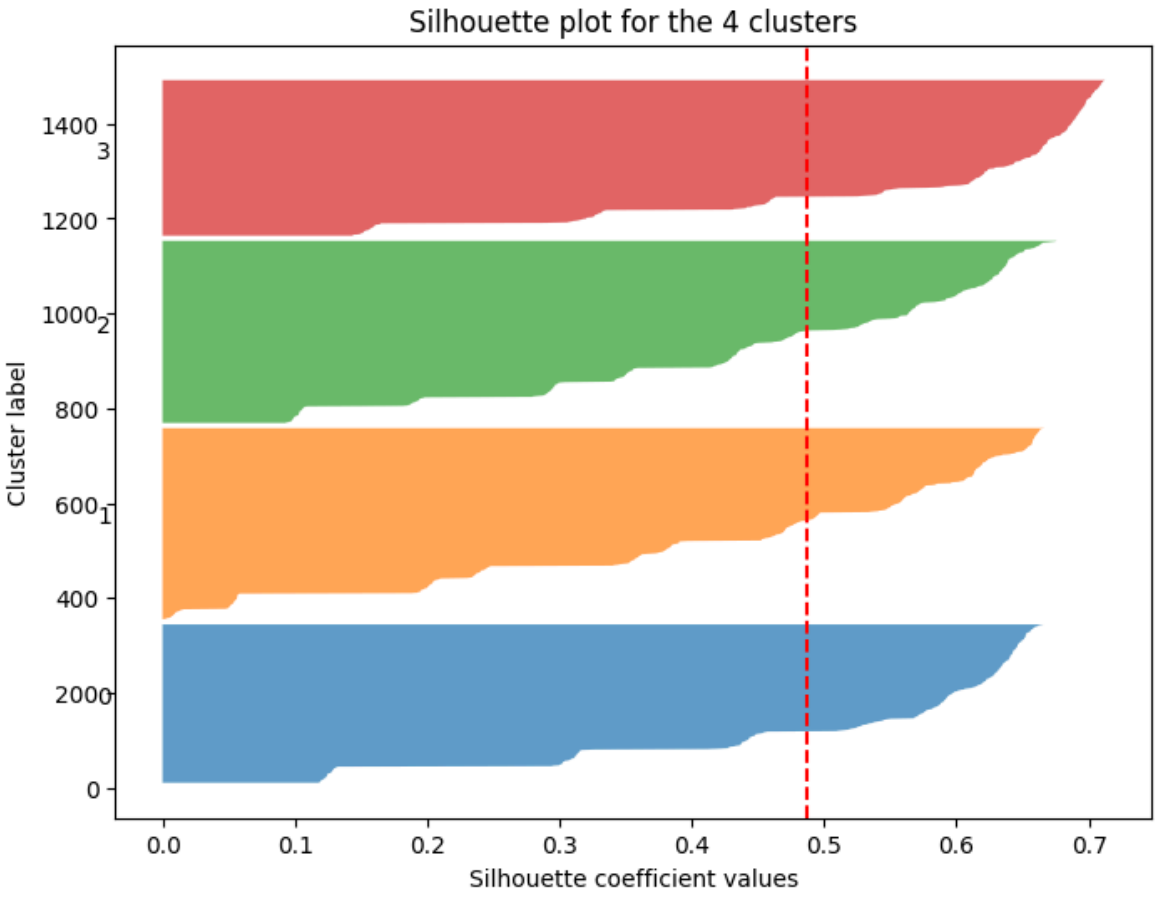
Out[45]:

	Age	yummy_Yes	convenient_Yes	spicy_Yes	fattening_Yes	greasy_Yes	fast_Yes	che
0	61	False	True	False	True	False	True	
1	51	True	True	False	True	True	True	
2	62	False	True	True	True	True	True	

3 rows × 28 columns

In [46]:
```
from sklearn.metrics import silhouette_samples

# Get silhouette scores for each sample
sample_silhouette_values = silhouette_samples(df_encoded, kmeans_4.labels_)

# Create a plot
plt.figure(figsize=(8, 6))
y_lower = 10
n_clusters = 4  # number of clusters

for i in range(n_clusters):
    ith_cluster_silhouette_values = sample_silhouette_values[kmeans_4.labels_ ==
    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    plt.fill_betweenx(np.arange(y_lower, y_upper),
                      0, ith_cluster_silhouette_values, alpha=0.7)

    plt.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

    y_lower = y_upper + 10  # 10 for spacing between clusters

plt.xlabel("Silhouette coefficient values")
plt.ylabel("Cluster label")
```

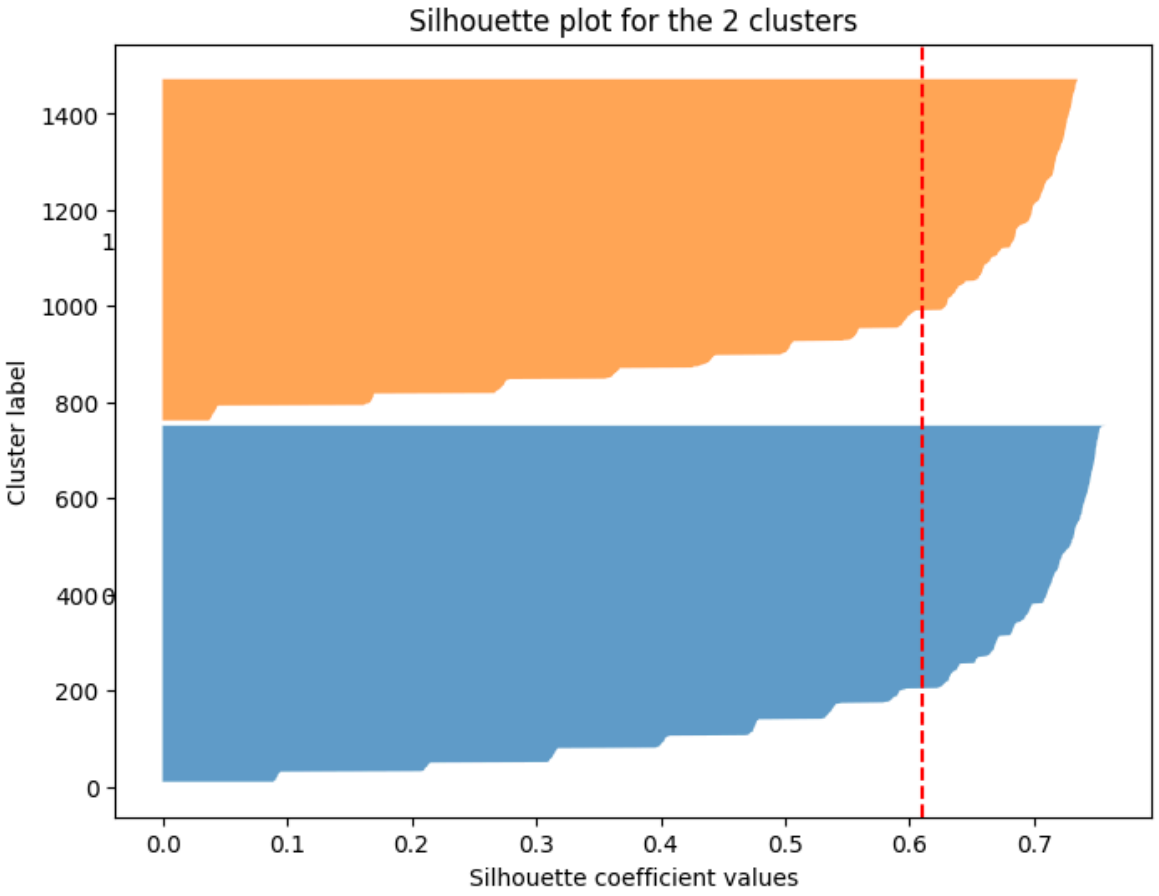
```
plt.axvline(x=np.mean(sample_silhouette_values), color="red", linestyle="--")
plt.title("Silhouette plot for the 4 clusters")
plt.show()
```



In [47]:

```
# Try for two clusters also

# Get silhouette scores for each sample
sample_silhouette_values = silhouette_samples(df_encoded, kmeans_2.labels_)

# Create a plot
plt.figure(figsize=(8, 6))
y_lower = 10
n_clusters = 2  # number of clusters

for i in range(n_clusters):
    ith_cluster_silhouette_values = sample_silhouette_values[kmeans_2.labels_ ==
    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    plt.fill_betweenx(np.arange(y_lower, y_upper),
                      0, ith_cluster_silhouette_values, alpha=0.7)

    plt.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

    y_lower = y_upper + 10  # 10 for spacing between clusters

plt.xlabel("Silhouette coefficient values")
plt.ylabel("Cluster label")
```

```
plt.axvline(x=np.mean(sample_silhouette_values), color="red", linestyle="--")
plt.title("Silhouette plot for the 2 clusters")
plt.show()
```



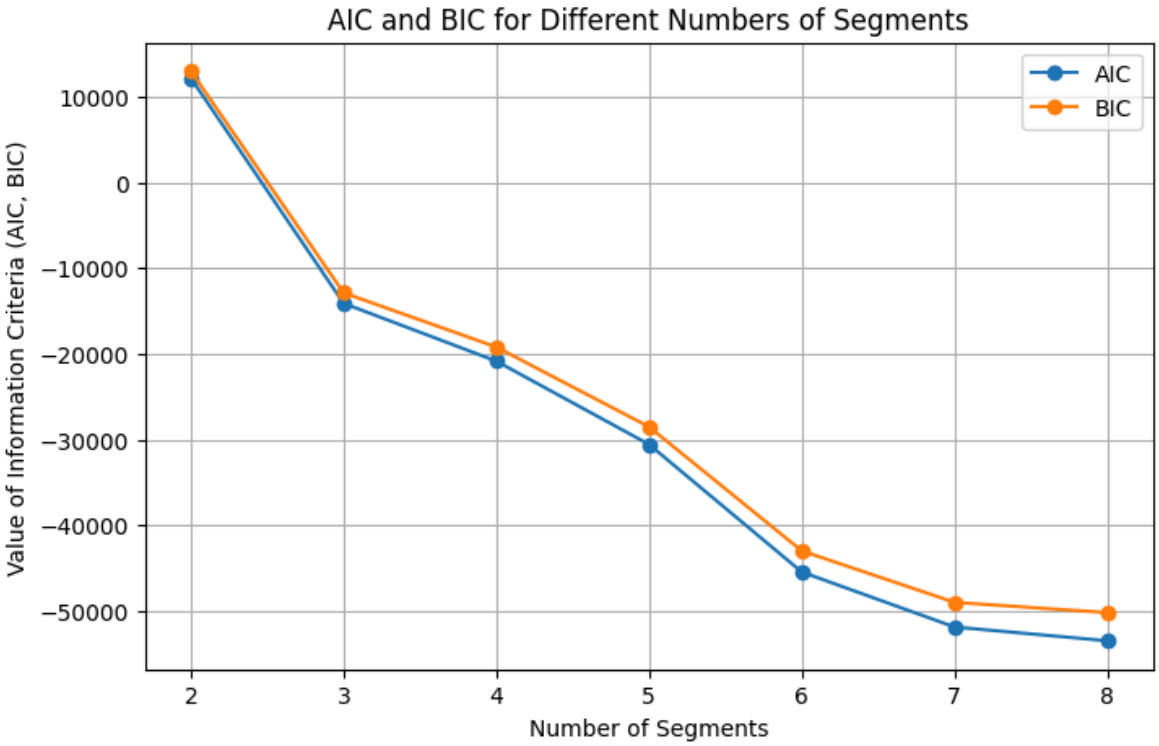
Using Mixtures of Distributions

In [49]:
```
from sklearn.mixture import GaussianMixture

# Set random seed for reproducibility
np.random.seed(1234)

# Function to fit a Gaussian Mixture Model for a range of cluster sizes
def fit_mixture_model(X, n_clusters_range, n_init=10):
    mixture_models = []

    # Iterate over the range of cluster sizes
    for n_clusters in n_clusters_range:
        # Fit the Gaussian Mixture Model with n_clusters
        gmm = GaussianMixture(n_components=n_clusters, n_init=n_init, random_sta
        gmm.fit(X)

        # Append the fitted model to the list
        mixture_models.append(gmm)

        # Print the model details
        print(f"Fitted Gaussian Mixture Model for {n_clusters} segments.")

    return mixture_models

# Fit mixture models for 2 to 8 clusters (analogous to R's stepFlexmix)
n_clusters_range = range(2, 9)  # Clusters 2 to 8
mixture_models = fit_mixture_model(MD_x_scaled, n_clusters_range)
```

```
# Display information about the fitted models
for n_clusters, model in zip(n_clusters_range, mixture_models):
    print(f"Mixture Model with {n_clusters} segments:")
    print(f"Converged: {model.converged_}, Log Likelihood: {model.lower_bound_:.
```

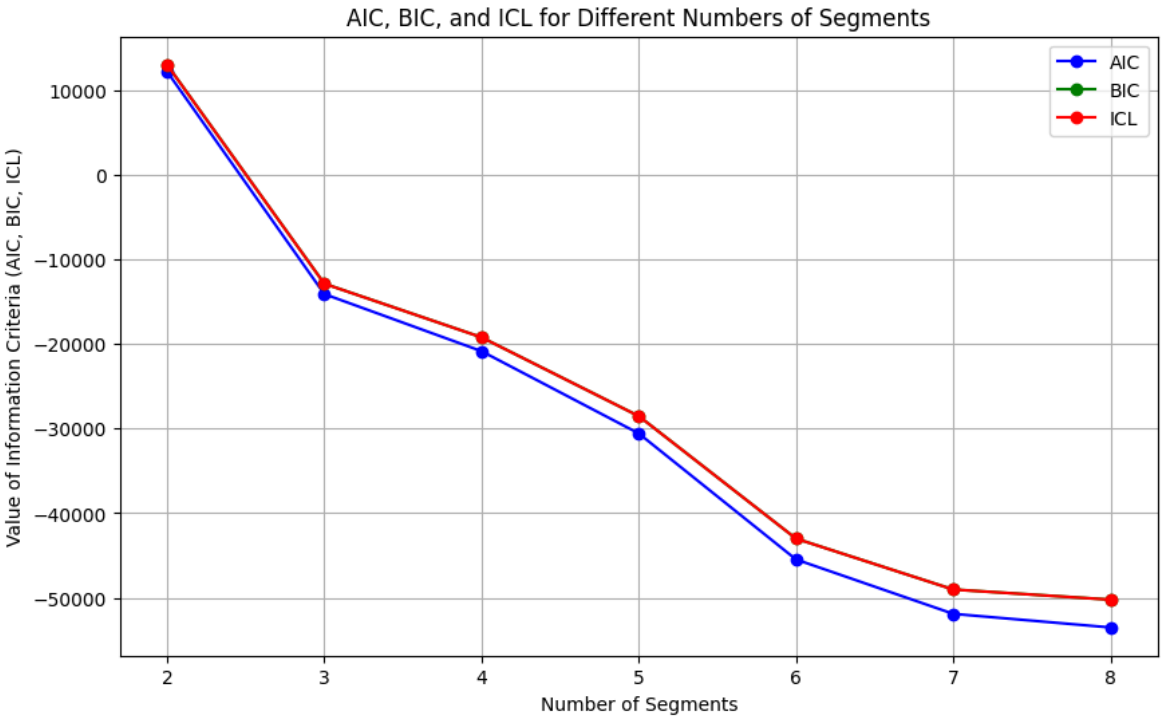
```
Fitted Gaussian Mixture Model for 2 segments.
Fitted Gaussian Mixture Model for 3 segments.
Fitted Gaussian Mixture Model for 4 segments.
Fitted Gaussian Mixture Model for 5 segments.
Fitted Gaussian Mixture Model for 6 segments.
Fitted Gaussian Mixture Model for 7 segments.
Fitted Gaussian Mixture Model for 8 segments.
Mixture Model with 2 segments:
Converged: True, Log Likelihood: -4.08
Mixture Model with 3 segments:
Converged: True, Log Likelihood: 5.02
Mixture Model with 4 segments:
Converged: True, Log Likelihood: 7.39
Mixture Model with 5 segments:
Converged: True, Log Likelihood: 10.79
Mixture Model with 6 segments:
Converged: True, Log Likelihood: 15.96
Mixture Model with 7 segments:
Converged: True, Log Likelihood: 18.23
Mixture Model with 8 segments:
Converged: True, Log Likelihood: 18.84
```

In [50]:
```
n_clusters_range = range(2, 9)  # Segments 2 to 8
aic_values = []
bic_values = []

# Fit GMMs and calculate AIC and BIC
for n_clusters in n_clusters_range:
    gmm = GaussianMixture(n_components=n_clusters, n_init=10, random_state=1234)
    gmm.fit(MD_x_scaled)

    # Append AIC and BIC values
    aic_values.append(gmm.aic(MD_x_scaled))
    bic_values.append(gmm.bic(MD_x_scaled))

# Plotting AIC and BIC values
plt.figure(figsize=(8, 5))
plt.plot(n_clusters_range, aic_values, label='AIC', marker='o')
plt.plot(n_clusters_range, bic_values, label='BIC', marker='o')
plt.xlabel('Number of Segments')
plt.ylabel('Value of Information Criteria (AIC, BIC)')
plt.title('AIC and BIC for Different Numbers of Segments')
plt.legend()
plt.grid(True)
plt.show()
```



In [51]:
```python
from scipy.stats import entropy

n_clusters_range = range(2, 9)  # Clusters 2 to 8
aic_values = []
bic_values = []
icl_values = []

def calculate_entropy(probabilities):
    """Calculate entropy of cluster assignments"""
    return entropy(probabilities, base=2)

# Fit GMMs and calculate AIC, BIC, and ICL
for n_clusters in n_clusters_range:
    gmm = GaussianMixture(n_components=n_clusters, n_init=10, random_state=1234)
    gmm.fit(MD_x_scaled)

    # Calculate AIC and BIC
    aic_values.append(gmm.aic(MD_x_scaled))
    bic_values.append(gmm.bic(MD_x_scaled))

    # Calculate Entropy and ICL
    labels = gmm.predict(MD_x_scaled)
    cluster_probs = gmm.predict_proba(MD_x_scaled)
    # Entropy for each cluster
    entropy_value = np.mean([calculate_entropy(cluster_probs[:, i]) for i in ran
    icl_value = gmm.bic(MD_x_scaled) - entropy_value
    icl_values.append(icl_value)

# Plotting AIC, BIC, and ICL values
plt.figure(figsize=(10, 6))
plt.plot(n_clusters_range, aic_values, label='AIC', marker='o', color='blue')
plt.plot(n_clusters_range, bic_values, label='BIC', marker='o', color='green')
plt.plot(n_clusters_range, icl_values, label='ICL', marker='o', color='red')
plt.xlabel('Number of Segments')
plt.ylabel('Value of Information Criteria (AIC, BIC, ICL)')
plt.title('AIC, BIC, and ICL for Different Numbers of Segments')
```

```
plt.legend()
plt.grid(True)
plt.show()
```



In [52]:
```
print("BIC Values:", bic_values)
```

```
BIC Values: [12990.960915652473, -12877.852181663086, -19213.66960039565, -28521.
164881337143, -42992.02590051668, -49004.57590504398, -50202.49927802214]
```

In [53]:
```
from sklearn.metrics import confusion_matrix

# Fit k-means with 4 clusters
kmeans_4 = KMeans(n_clusters=4, n_init=10, random_state=1234)
kmeans_4.fit(MD_x_scaled)
kmeans_labels = kmeans_4.labels_

# Fit Gaussian Mixture Model with 4 clusters
gmm_4 = GaussianMixture(n_components=4, n_init=10, random_state=1234)
gmm_4.fit(MD_x_scaled)
gmm_labels = gmm_4.predict(MD_x_scaled)

# Create a DataFrame to compare cluster assignments
comparison_df = pd.DataFrame({
    'kmeans_cluster': kmeans_labels,
    'gmm_cluster': gmm_labels
})

# Compute the contingency table
contingency_table = pd.crosstab(comparison_df['kmeans_cluster'], comparison_df['

# Display the contingency table
print("Contingency Table:")
print(contingency_table)

# Optionally, compute confusion matrix if labels are known
# confusion = confusion_matrix(kmeans_labels, gmm_labels)
# print("Confusion Matrix:")
# print(confusion)
```

```
Contingency Table:
gmm_cluster      0    1    2    3
kmeans_cluster
0               21   14   27  113
1               28  134  198    0
2               41   14   36  227
3               46    9  147  398
```

In [54]:
```python
# Perform k-means clustering
kmeans = KMeans(n_clusters=4, n_init=10, random_state=1234)
kmeans_labels = kmeans.fit_predict(MD_x_scaled)

# Initialize Gaussian Mixture Model with k-means labels
gmm = GaussianMixture(n_components=4, init_params='kmeans', n_init=10, random_st
gmm.fit(MD_x_scaled)

# Reassign the GMM to use the k-means cluster labels as the initialization
gmm.means_init = kmeans.cluster_centers_

# Fit the GMM
gmm.fit(MD_x_scaled)

# Get the cluster assignments from the GMM
gmm_labels = gmm.predict(MD_x_scaled)

# Compare k-means and GMM cluster assignments
comparison_df = pd.DataFrame({
    'kmeans': kmeans_labels,
    'gmm': gmm_labels
})

# Create a contingency table
contingency_table = pd.crosstab(comparison_df['kmeans'], comparison_df['gmm'])
print(contingency_table)
```

```
gmm       0    1    2    3
kmeans
0       103   47   25    0
1         0  208   14  138
2        35   29  247    7
3       416   21    1  162
```

In [55]:
```python
# Perform k-means clustering
kmeans = KMeans(n_clusters=4, n_init=10, random_state=1234)
kmeans_labels = kmeans.fit_predict(MD_x_scaled)

# Initialize GMM with k-means labels
gmm_initial = GaussianMixture(n_components=4, init_params='kmeans', n_init=10, r
gmm_initial.fit(MD_x_scaled)

# Reassign the GMM to use the k-means cluster labels as the initialization
gmm = GaussianMixture(n_components=4, n_init=10, random_state=1234)
gmm.means_init = kmeans.cluster_centers_
gmm.fit(MD_x_scaled)

# Compute log-likelihood values
log_likelihood_initial = gmm_initial.score(MD_x_scaled) * -1  # Negative log-lik
log_likelihood_custom = gmm.score(MD_x_scaled) * -1  # Negative log-likelihood

# Print log-likelihood values
```

```
print(f'Log-likelihood with k-means initialization: {log_likelihood_initial:.3f}
print(f'Log-likelihood with custom initialization: {log_likelihood_custom:.3f}')
```

```
Log-likelihood with k-means initialization: -7.391
Log-likelihood with custom initialization: -7.174
```

Using Mixtures of Regression Models

In [57]:
```
# Create a frequency table
frequency_table = df['Like'].value_counts()

# Reverse the order of the frequency counts
reversed_frequency_table = frequency_table[::-1]

print(reversed_frequency_table)
```

```
Like
-1               58
-2               59
-4               71
-3               73
I love it!+5    143
I hate it!-5    152
+1              152
+4              160
0               169
+2              187
+3              229
Name: count, dtype: int64
```

In [58]:
```
# Convert 'Like' column to numeric (if it's not already numeric)
df['Like'] = pd.to_numeric(df['Like'], errors='coerce')

# Create the new column 'Like.n' by subtracting from 6
df['Like.n'] = 6 - df['Like']

# Create a frequency table for the new column 'Like.n'
frequency_table = df['Like.n'].value_counts()

print(frequency_table)
```

```
Like.n
3.0     229
4.0     187
6.0     169
2.0     160
5.0     152
9.0      73
10.0     71
8.0      59
7.0      58
Name: count, dtype: int64
```

In [59]:
```
from sklearn.preprocessing import LabelEncoder

# Example DataFrame creation for demonstration
# Replace this with loading your actual dataset
mcdonalds = df

# Convert categorical variables to numeric using Label Encoding
label_encoders = {}
```

```
for column in mcdonalds.columns:
    le = LabelEncoder()
    mcdonalds[column] = le.fit_transform(mcdonalds[column])
    label_encoders[column] = le

# Define the feature matrix (assuming all columns are features)
X = mcdonalds

# Set random seed for reproducibility
np.random.seed(1234)

# Fit Gaussian Mixture Model with 2 components
gmm = GaussianMixture(n_components=2, n_init=10, random_state=1234)
gmm.fit(X)

# Display cluster sizes
labels = gmm.predict(X)
cluster_sizes = pd.Series(labels).value_counts().sort_index()

print("Cluster sizes:")
print(cluster_sizes)

# Display model parameters
print("GMM model details:")
print(gmm)
```

```
Cluster sizes:
0     408
1    1045
Name: count, dtype: int64
GMM model details:
GaussianMixture(n_components=2, n_init=10, random_state=1234)
```

In [60]:

```
# Convert categorical variables to numeric using Label Encoding
from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for column in mcdonalds.columns:
    le = LabelEncoder()
    mcdonalds[column] = le.fit_transform(mcdonalds[column])
    label_encoders[column] = le

# Define the feature matrix
X = mcdonalds

# Set random seed for reproducibility
np.random.seed(1234)

# Fit initial Gaussian Mixture Model
gmm_initial = GaussianMixture(n_components=2, n_init=10, random_state=1234)
gmm_initial.fit(X)

# Refitting the model (e.g., reinitialize with different parameters or same para
gmm_refit = GaussianMixture(n_components=2, n_init=10, random_state=1234)
gmm_refit.fit(X)

# Summarize the refitted model
print("Refitted GMM model details:")
print("Means of components:\n", gmm_refit.means_)
print("Covariances of components:\n", gmm_refit.covariances_)
print("Weights of components:\n", gmm_refit.weights_)
```

```
print("Converged:\n", gmm_refit.converged_)

# Display cluster sizes for the refitted model
labels = gmm_refit.predict(X)
cluster_sizes = pd.Series(labels).value_counts().sort_index()

print("Cluster sizes for refitted model:")
print(cluster_sizes)
```

```
Refitted GMM model details:
Means of components:
 [[ 0.4877451   0.67156863  0.33333333  0.52696078  0.45098039  0.84803922
   0.56617647  0.59558824  0.40931373  0.34558824  0.33088235  6.16176471
  31.31372549  2.63480392  0.53676471  5.0245098 ]
 [ 0.57799043  1.          0.          1.          0.55598086  0.92057416
   0.61148325  0.66315789  0.33779904  0.14162679  0.20861244  5.69090909
  24.76650718  2.63923445  0.42679426  3.88803828]]
Covariances of components:
 [[[ 2.49850817e-01  1.08720204e-01  2.61437908e-02 -8.30029316e-02
    -8.27085736e-02  3.98044022e-02  7.43403979e-02  1.75191032e-01
    -5.01309592e-02  1.05950836e-01 -1.12366638e-01  4.33354960e-01
    -9.66743561e-01 -5.47205402e-02 -2.16082757e-02 -6.63915321e-01]
  [ 1.08720204e-01  2.20565206e-01  8.74183006e-02 -9.89883698e-02
    -7.98250673e-02  5.79344483e-02  6.58520761e-02  1.26982411e-01
    -6.40979431e-02  8.89922145e-02 -1.21720012e-01 -1.30478662e-02
     5.12351019e-01 -6.60202807e-02  4.72174164e-03 -7.54205113e-01]
  [ 2.61437908e-02  8.74183006e-02  2.22223222e-01  9.88562091e-02
     5.31045751e-02  2.36928105e-02  1.96078431e-02  4.65686274e-02
     5.71895424e-03 -4.90196079e-03 -1.47058824e-02 -1.98529412e-01
     5.86601307e-01 -3.02287582e-02 -2.33531578e-13 -2.26307190e-01]
  [-8.30029316e-02 -9.89883698e-02  9.88562091e-02  2.49274116e-01
     1.56958862e-01 -8.15792003e-03 -3.11959342e-02 -7.36555652e-02
     7.59743849e-02 -9.38761534e-02  1.17304642e-01 -2.17596597e-01
    -1.65321031e-01  1.10714629e-02  1.45977509e-03  4.99339196e-01]
  [-8.27085736e-02 -7.98250673e-02  5.31045751e-02  1.56958862e-01
     2.47598078e-01 -1.72529796e-02 -4.20991926e-02 -7.00692041e-02
     8.99173395e-02 -7.74221453e-02  1.13033449e-01 -2.34717416e-01
    -1.15128797e+00  6.66570550e-02  3.48904268e-02  4.64436755e-01]
  [ 3.98044022e-02  5.79344483e-02  2.36928105e-02 -8.15792003e-03
    -1.72529796e-02  1.28869704e-01  6.15268166e-02  5.12903691e-02
    -4.07415417e-02  2.80060554e-02 -4.28561130e-02  4.97404845e-03
     1.06497501e-01  5.77902729e-03 -6.66810842e-03 -2.56079393e-01]
  [ 7.43403979e-02  6.58520761e-02  1.96078431e-02 -3.11959342e-02
    -4.20991926e-02  6.15268166e-02  2.45621675e-01  7.21056805e-02
    -1.58214389e-01  6.65909746e-02 -5.25338812e-02  1.02040081e-01
     3.68944636e-01  2.04909170e-02 -3.67466840e-02 -2.63876874e-01]
  [ 1.75191032e-01  1.26982411e-01  4.65686274e-02 -7.36555652e-02
    -7.00692041e-02  5.12903691e-02  7.21056805e-02  2.40863889e-01
    -5.50569492e-02  9.07403403e-02 -1.25991205e-01  2.95811707e-01
    -5.71655133e-01  6.72217416e-03 -1.57691753e-02 -7.71950692e-01]
  [-5.01309592e-02 -6.40979431e-02  5.71895424e-03  7.59743849e-02
     8.99173395e-02 -4.07415417e-02 -1.58214389e-01 -5.50569492e-02
     2.41777000e-01 -3.85128316e-02  8.02515859e-02 -3.18987889e-02
    -1.33919646e+00  4.87192425e-03  4.74517013e-02  3.45359958e-01]
  [ 1.05950836e-01  8.89922145e-02 -4.90196079e-03 -9.38761534e-02
    -7.74221453e-02  2.80060554e-02  6.65909746e-02  9.07403403e-02
    -3.85128316e-02  2.26158007e-01 -6.53294406e-02  2.72527393e-01
    -2.48125721e-01 -1.24351212e-03 -1.14799596e-02 -3.44254614e-01]
  [-1.12366638e-01 -1.21720012e-01 -1.47058824e-02  1.17304642e-01
     1.13033449e-01 -4.28561130e-02 -5.25338812e-02 -1.25991205e-01
     8.02515859e-02 -6.53294406e-02  2.21400221e-01 -7.31329296e-02
    -4.05276817e-01  6.20134083e-02  1.60214821e-02  7.59047001e-01]
  [ 4.33354960e-01 -1.30478662e-02 -1.98529412e-01 -2.17596597e-01
    -2.34717416e-01  4.97404845e-03  1.02040081e-01  2.95811707e-01
    -3.18987889e-02  2.72527393e-01 -7.31329296e-02  8.23363710e+00
    -3.51888697e+00  7.62326990e-02 -1.33001730e-02  8.09760669e-01]
  [-9.66743561e-01  5.12351019e-01  5.86601307e-01 -1.65321031e-01
    -1.15128797e+00  1.06497501e-01  3.68944636e-01 -5.71655133e-01
    -1.33919646e+00 -2.48125721e-01 -4.05276817e-01 -3.51888697e+00
```

```
  1.86582950e+02 -2.42219339e+00 -1.19780854e+00 -1.18170896e+00]
 [-5.47205402e-02 -6.60202807e-02 -3.02287582e-02  1.10714629e-02
  6.66570550e-02  5.77902729e-03  2.04909170e-02  6.72217416e-03
  4.87192425e-03 -1.24351212e-03  6.20134083e-02  7.62326990e-02
 -2.42219339e+00  2.97692694e+00  4.84789503e-03  4.50127355e-01]
 [-2.16082757e-02  4.72174164e-03 -2.33513526e-13  1.45977509e-03
  3.48904268e-02 -6.66810842e-03 -3.67466840e-02 -1.57691753e-02
  4.74517013e-02 -1.14799596e-02  1.60214821e-02 -1.33001730e-02
 -1.19780854e+00  4.84789503e-03  2.48649356e-01  1.87067474e-02]
 [-6.63915321e-01 -7.54205113e-01 -2.26307190e-01  4.99339196e-01
  4.64436755e-01 -2.56079393e-01 -2.63876874e-01 -7.71950692e-01
  3.45359958e-01 -3.44254614e-01  7.59047001e-01  8.09760669e-01
 -1.18170896e+00  4.50127355e-01  1.87067474e-02  1.18572434e+01]]

[[ 2.43918493e-01  7.99240654e-31  0.00000000e+00  7.99240654e-31
  -2.27870241e-02  4.75904856e-03  5.42020558e-03  1.57370939e-01
   9.27634440e-04  3.20166663e-02 -7.75137932e-02  5.57598956e-01
  -2.22485200e+00 -7.74890685e-03 -1.03193608e-02 -5.70693894e-01]
 [ 7.81948440e-31  1.00000000e-06  0.00000000e+00  1.23259516e-30
   5.13909805e-31  1.26188170e-30  6.18153273e-31  1.03723762e-30
   3.06956488e-31  1.80964527e-31  2.56444608e-31  6.10091436e-30
   2.47241841e-29  3.18865882e-30  5.70697459e-31  4.49311015e-30]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e-06  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 7.81948440e-31  1.23259516e-30  0.00000000e+00  1.00000000e-06
   5.13909805e-31  1.26188170e-30  6.18153273e-31  1.03723762e-30
   3.06956488e-31  1.80964527e-31  2.56444608e-31  6.10091436e-30
   2.47241841e-29  3.18865882e-30  5.70697459e-31  4.49311015e-30]
 [-2.27870241e-02  5.19140775e-31  0.00000000e+00  5.19140775e-31
   2.46867143e-01 -7.51539571e-03 -9.82944530e-03 -2.89901788e-02
   1.60179483e-02 -2.22824569e-02  5.53073419e-02 -1.17146585e-01
  -1.66252696e+00 -1.66461391e-02 -2.83967858e-03  2.24928001e-01]
 [ 4.75904856e-03  1.26397031e-30  0.00000000e+00  1.26397031e-30
  -7.51539571e-03  7.31183737e-02  2.65580000e-02  7.69579451e-03
  -2.29307937e-02 -1.19136467e-03 -5.44035164e-03  7.98608090e-03
  -3.00286165e-02  5.79565486e-03 -5.33595843e-03 -6.97877795e-03]
 [ 5.42020558e-03  6.22997238e-31  0.00000000e+00  6.22997238e-31
  -9.82944530e-03  2.65580000e-02  2.37572484e-01  1.64996223e-02
  -1.73065635e-01  1.19621803e-02 -1.46443534e-02  3.78077425e-02
   1.03542501e-01  3.11284082e-02 -2.36569676e-02 -2.72310616e-02]
 [ 1.57370939e-01  1.03094495e-30  0.00000000e+00  1.03094495e-30
  -2.89901788e-02  7.69579451e-03  1.64996223e-02  2.23380501e-01
  -1.06169731e-02  2.85671116e-02 -7.32712163e-02  4.56650718e-01
  -1.49396122e+00 -3.06119365e-02 -6.47695794e-03 -5.46804331e-01]
 [ 9.27634440e-04  3.08254948e-31  0.00000000e+00  3.08254948e-31
   1.60179483e-02 -2.29307937e-02 -1.73065635e-01 -1.06169731e-02
   2.23691850e-01 -7.65000802e-03  2.33108216e-02 -1.32927360e-02
  -2.79978022e-01 -1.30619720e-02  2.71211740e-02  6.36578833e-02]
 [ 3.20166663e-02  1.79858635e-31  0.00000000e+00  1.79858635e-31
  -2.22824569e-02 -1.19136467e-03  1.19621803e-02  2.85671116e-02
  -7.65000802e-03  1.21569645e-01 -2.28465466e-02  1.10761201e-01
  -1.09514892e-01 -7.27913738e-03 -1.35555505e-02 -5.68704929e-02]
 [-7.75137932e-02  2.64164606e-31  0.00000000e+00  2.64164606e-31
   5.53073419e-02 -5.44035164e-03 -1.46443534e-02 -7.32712163e-02
   2.33108216e-02 -2.28465466e-02  1.65094290e-01 -1.78581992e-01
   1.16652091e-01 -9.90728234e-03  1.04869394e-02  4.90342254e-01]
 [ 5.57598956e-01  6.10385844e-30  0.00000000e+00  6.10385844e-30
  -1.17146585e-01  7.98608090e-03  3.78077425e-02  4.56650718e-01
```

```
  -1.32927360e-02  1.10761201e-01 -1.78581992e-01  6.43939204e+00
  -7.62145281e+00  4.63853849e-02 -3.45889517e-02 -1.21451066e+00]
 [-2.22485200e+00  2.47004050e-29  0.00000000e+00  2.47004050e-29
  -1.66252696e+00 -3.00286165e-02  1.03542501e-01 -1.49396122e+00
  -2.79978022e-01 -1.09514892e-01  1.16652091e-01 -7.62145281e+00
   1.96125386e+02 -2.63183535e-01  6.99883244e-02  3.06189602e+00]
 [-7.74890685e-03  3.17384408e-30  0.00000000e+00  3.17384408e-30
  -1.66461391e-02  5.79565486e-03  3.11284082e-02 -3.06119365e-02
  -1.30619720e-02 -7.27913738e-03 -9.90728234e-03  4.63853849e-02
  -2.63183535e-01  3.12248080e+00 -2.11469518e-02  3.79703761e-01]
 [-1.03193608e-02  5.59845903e-31  0.00000000e+00  5.59845903e-31
  -2.83967858e-03 -5.33595843e-03 -2.36569676e-02 -6.47695794e-03
   2.71211740e-02 -1.35555505e-02  1.04869394e-02 -3.45889517e-02
   6.99883244e-02 -2.11469518e-02  2.44641919e-01 -3.64259060e-02]
 [-5.70693894e-01  4.46197090e-30  0.00000000e+00  4.46197090e-30
   2.24928001e-01 -6.97877795e-03 -2.72310616e-02 -5.46804331e-01
   6.36578833e-02 -5.68704929e-02  4.90342254e-01 -1.21451066e+00
   3.06189602e+00  3.79703761e-01 -3.64259060e-02  9.28603017e+00]]]
Weights of components:
 [0.28079835 0.71920165]
Converged:
 True
Cluster sizes for refitted model:
0     408
1    1045
Name: count, dtype: int64
```

In [61]:
```
# Assign cluster labels to a new column
df['cluster'] = kmeans_4.labels_
# Display the first few rows to verify
df_encoded.head()
```

Out[61]:

	Age	yummy_Yes	convenient_Yes	spicy_Yes	fattening_Yes	greasy_Yes	fast_Yes	che
0	61	False	True	False	True	False	True	
1	51	True	True	False	True	True	True	
2	62	False	True	True	True	True	True	
3	69	True	True	False	True	True	True	
4	49	False	True	False	True	True	True	

5 rows × 28 columns

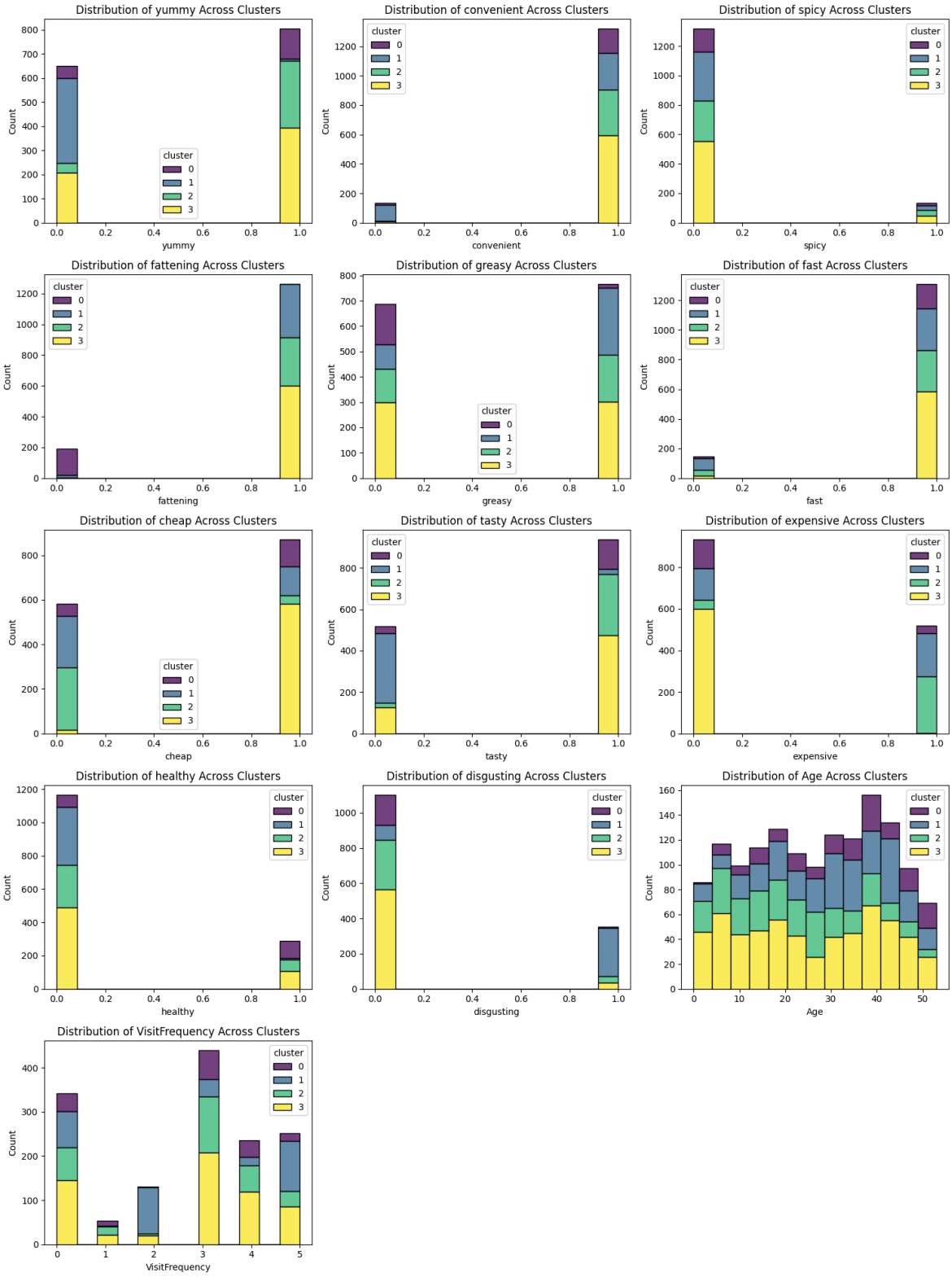
In [62]:
```
df.head()
```

Out[62]:

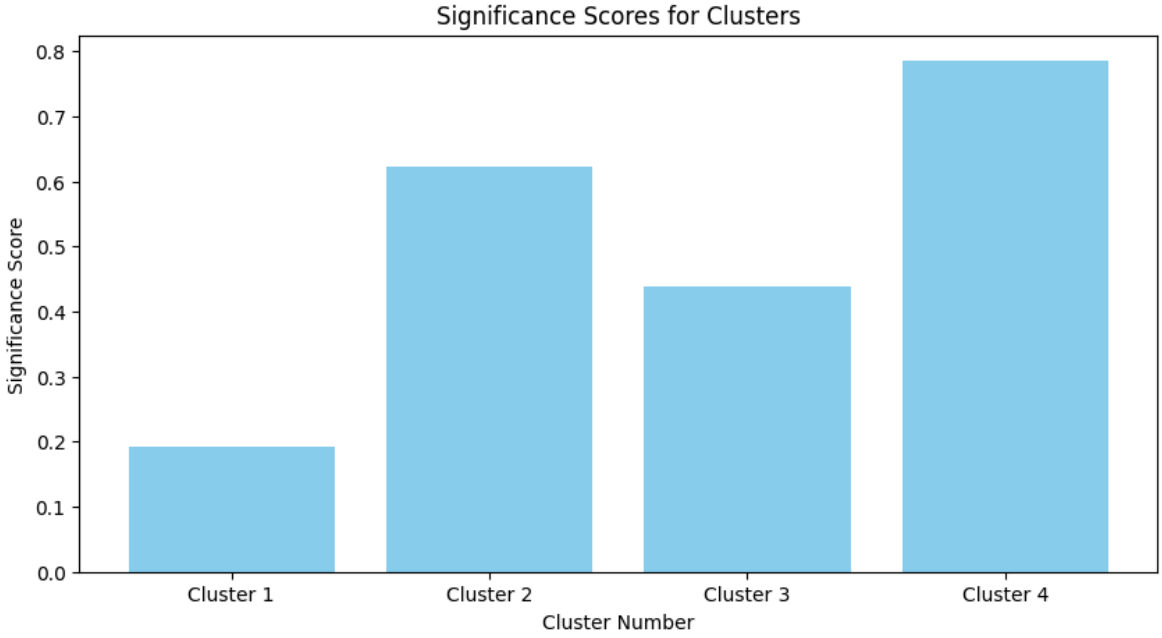
	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy
0	0	1	0	1	0	1	1	0	1	0
1	1	1	0	1	1	1	1	1	1	0
2	0	1	1	1	1	1	0	1	1	1
3	1	1	0	1	1	1	1	1	0	0
4	0	1	0	1	1	1	1	0	0	1

In [63]:

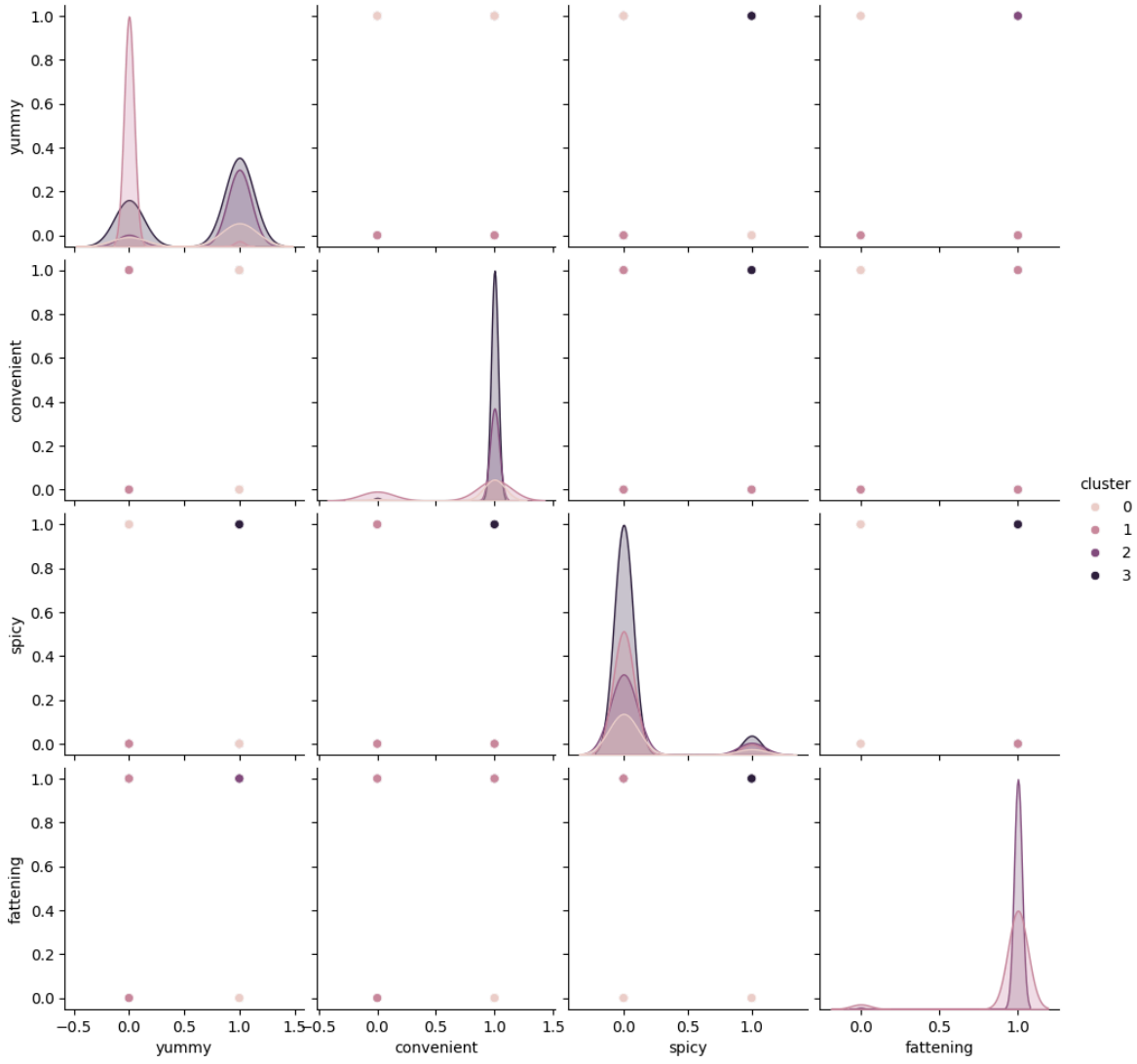
```
# List of features to visualize (excluding 'Cluster' column)
features = ['yummy', 'convenient', 'spicy', 'fattening', 'greasy', 'fast', 'chea

# Set up the plotting area
num_features = len(features)
num_cols = 3  # Number of columns in the plot grid
num_rows = (num_features + num_cols - 1) // num_cols  # Calculate number of rows

plt.figure(figsize=(15, 4 * num_rows))

for i, feature in enumerate(features):
    plt.subplot(num_rows, num_cols, i + 1)
    sns.histplot(data=df, x=feature, hue='cluster', multiple='stack', palette='v
    plt.title(f'Distribution of {feature} Across Clusters')
    plt.xlabel(feature)
    plt.ylabel('Count')

plt.tight_layout()
plt.show()
```



In [64]:
```
# Mock-up significance scores for clusters (replace with actual data if availabl
n_clusters = len(df['cluster'].unique())
significance_scores = np.random.rand(n_clusters)  # Replace with actual signific

plt.figure(figsize=(10, 5))
plt.bar(range(n_clusters), significance_scores, color='skyblue')
plt.title('Significance Scores for Clusters')
plt.xlabel('Cluster Number')
plt.ylabel('Significance Score')
plt.xticks(range(n_clusters), [f'Cluster {i+1}' for i in range(n_clusters)])
plt.show()
```

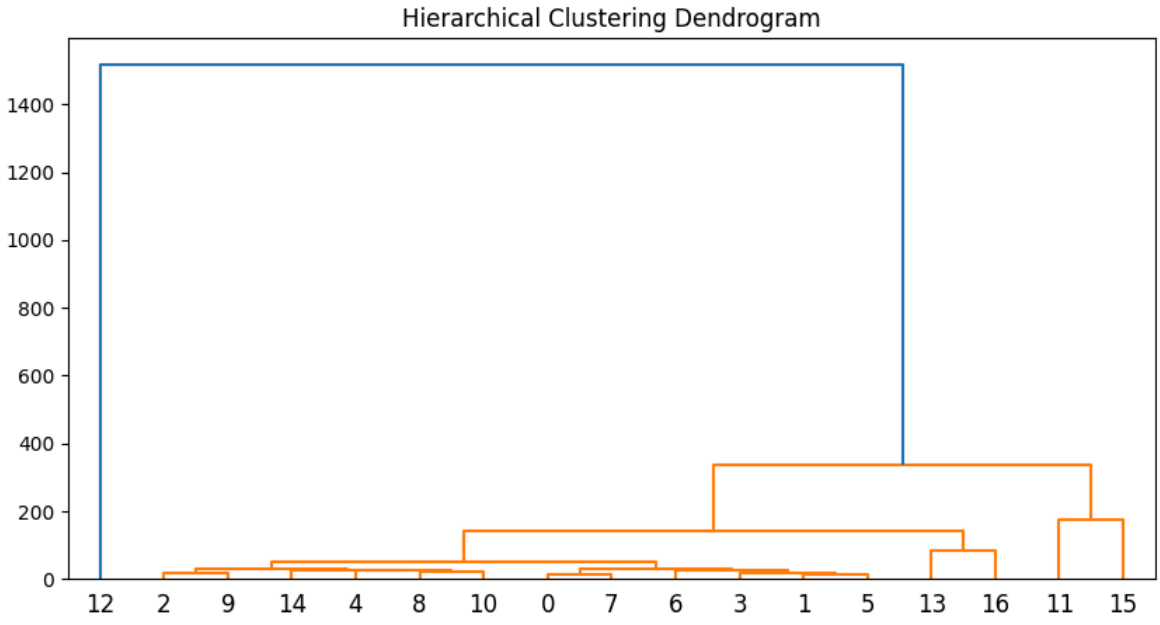


In [65]:
```
# Pairplot to visualize relationships between features across clusters
sns.pairplot(df, hue='cluster', vars=['yummy', 'convenient', 'spicy', 'fattening
plt.show()
```



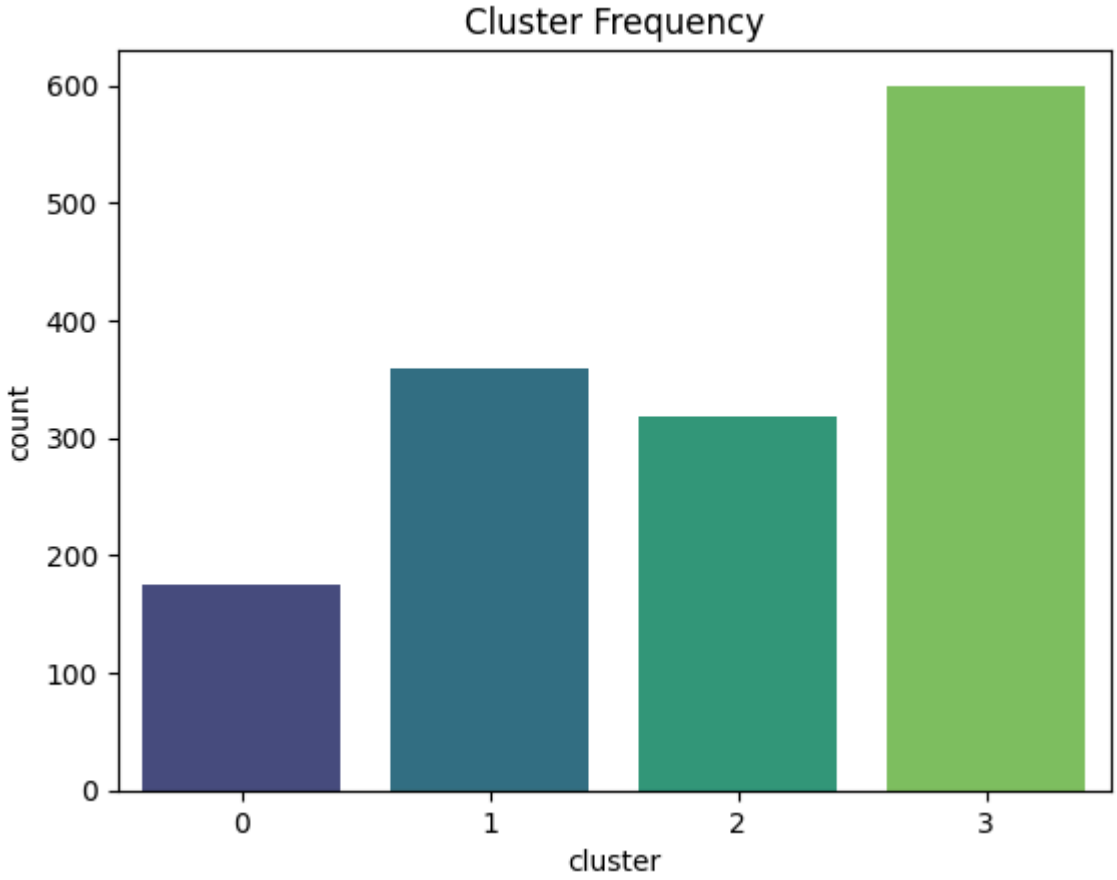
Step 6: Profiling Segments

In [67]:
```
from scipy.cluster.hierarchy import linkage
from scipy.spatial.distance import pdist

# Transpose the dataset (similar to t(MD.x) in R)
df_transposed = df.T

# Calculate the distance matrix and perform hierarchical clustering
dist_matrix = pdist(df_transposed)
hier_clust = linkage(dist_matrix, method='ward')

# Now you can plot a dendrogram if needed
from scipy.cluster.hierarchy import dendrogram
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
dendrogram(hier_clust)
plt.title('Hierarchical Clustering Dendrogram')
plt.show()
```

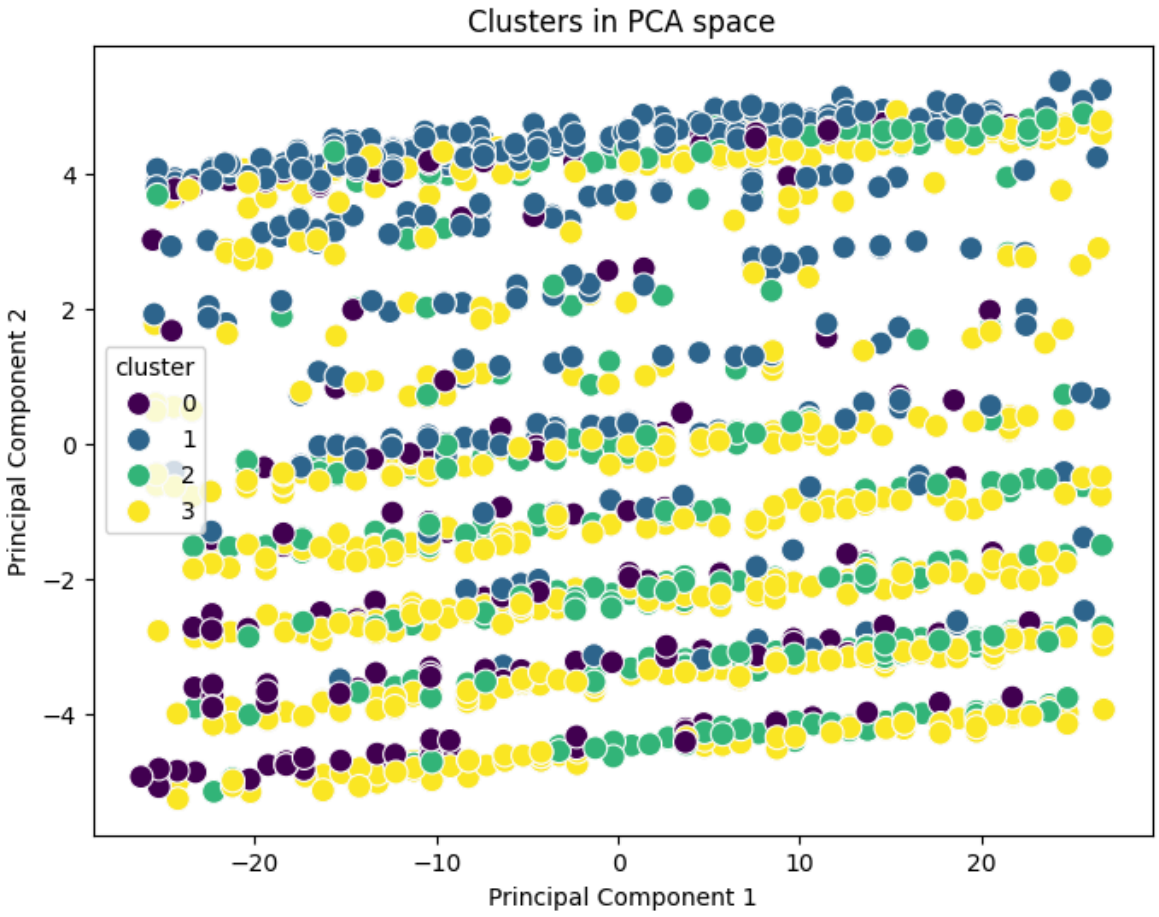


In [68]:
```
import seaborn as sns

# Create a bar chart showing cluster frequencies
sns.countplot(x='cluster', data=df, palette='viridis')
plt.title('Cluster Frequency')
plt.show()
```

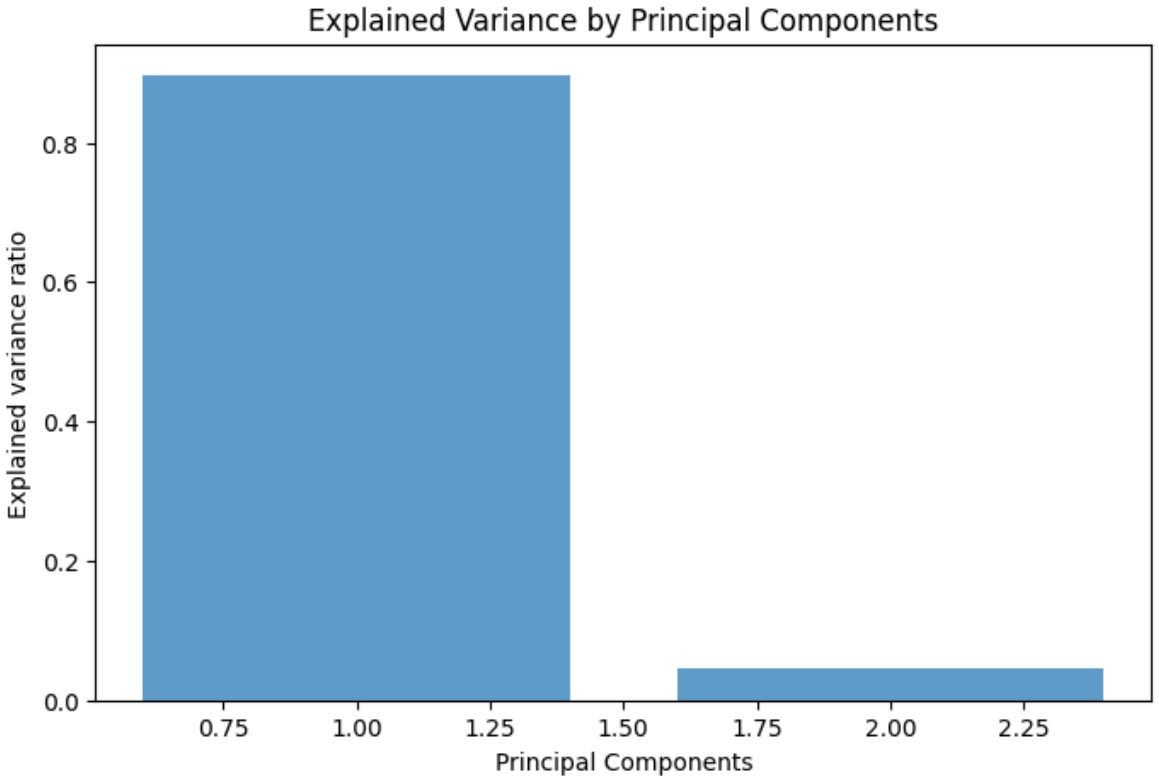
```
C:\Users\gyanp\AppData\Local\Temp\ipykernel_15316\3552391455.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='cluster', data=df, palette='viridis')
```



In [69]:
```
# Perform PCA to reduce the data to 2 components
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df)

# Add PCA components to the DataFrame
df['PCA1'] = df_pca[:, 0]
df['PCA2'] = df_pca[:, 1]

# Plot the clusters in the PCA space
plt.figure(figsize=(8, 6))
sns.scatterplot(x='PCA1', y='PCA2', hue='cluster', data=df, palette='viridis', s
plt.title('Clusters in PCA space')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

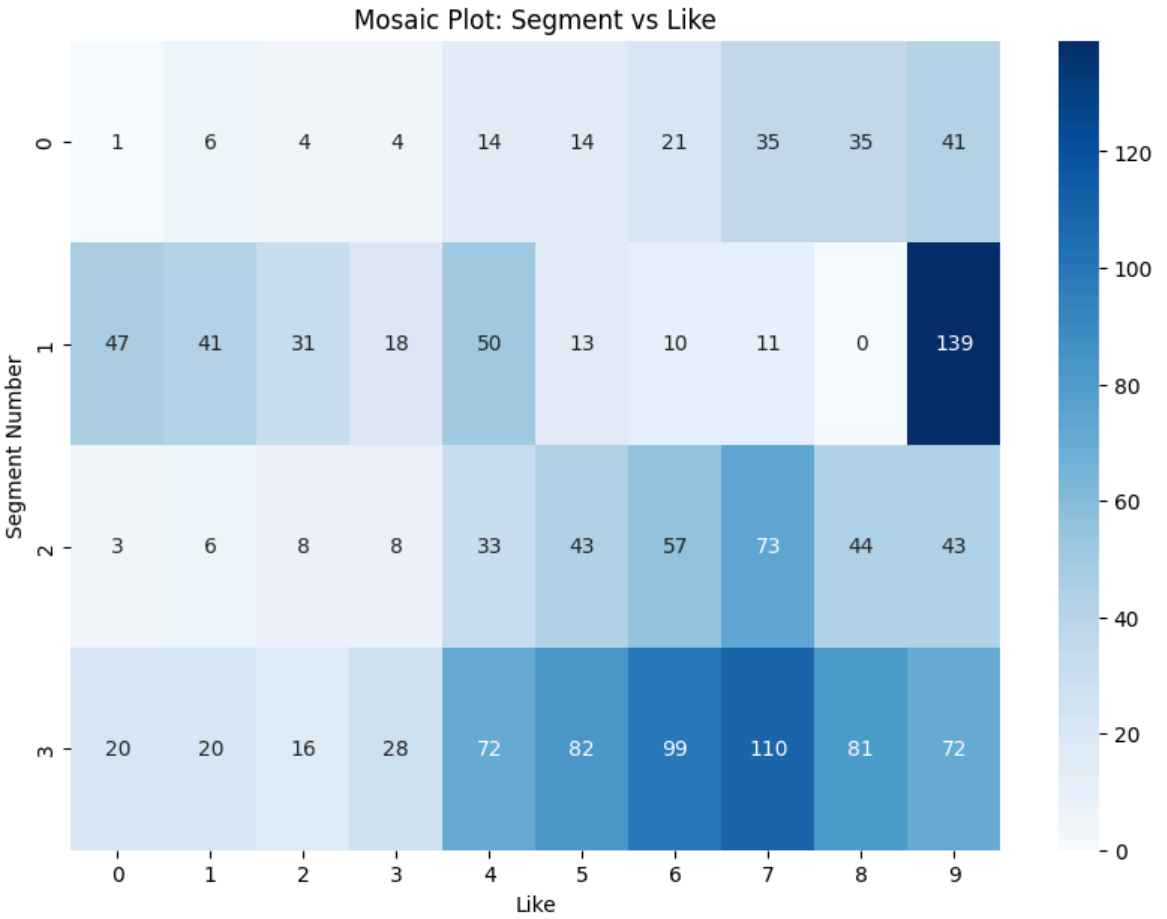


In [70]:
```
# Plot the explained variance ratio of the PCA components
plt.figure(figsize=(8, 5))
plt.bar(range(1, 3), pca.explained_variance_ratio_, alpha=0.7, align='center', l
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal Components')
plt.title('Explained Variance by Principal Components')
plt.show()
```

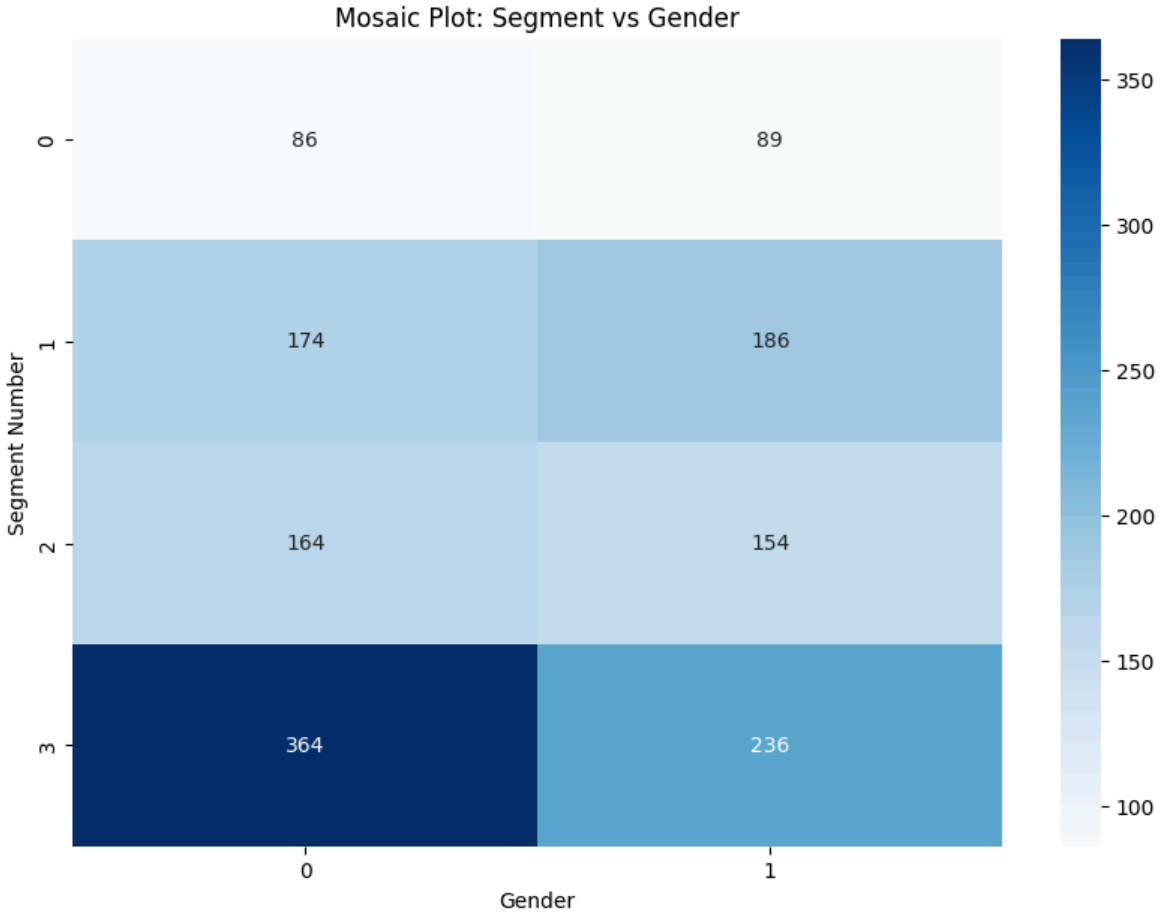


Step 7: Describing Segments

In [72]:
```
k4 = kmeans_4.labels_

# Create a contingency table
contingency_table = pd.crosstab(k4, df['Like'])

# Plot a heatmap
plt.figure(figsize=(10, 7))
sns.heatmap(contingency_table, annot=True, cmap='Blues', fmt='d')
plt.xlabel('Like')
plt.ylabel('Segment Number')
plt.title('Mosaic Plot: Segment vs Like')
plt.show()
```

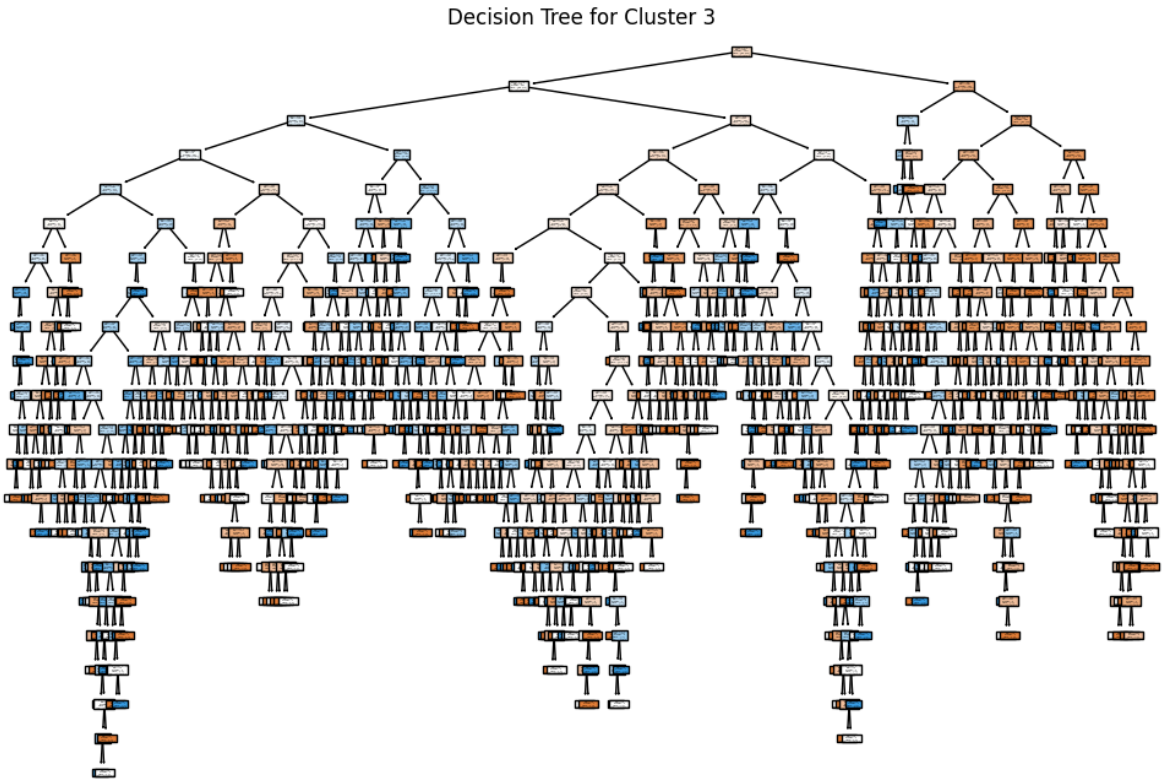


In [73]:
```
# Create a contingency table
contingency_table_gender = pd.crosstab(k4, df['Gender'])

# Plot a heatmap
plt.figure(figsize=(10, 7))
sns.heatmap(contingency_table_gender, annot=True, cmap='Blues', fmt='d')
plt.xlabel('Gender')
plt.ylabel('Segment Number')
plt.title('Mosaic Plot: Segment vs Gender')
plt.show()
```



In [74]:

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.preprocessing import LabelEncoder

# Encode categorical variables
le = LabelEncoder()
df['Gender'] = le.fit_transform(df['Gender'])
df['VisitFrequency'] = le.fit_transform(df['VisitFrequency'])

# Define the feature set and target variable
X = df[['Like.n', 'Age', 'VisitFrequency', 'Gender']]
y = (k4 == 3).astype(int)  # Target variable

# Create and fit the decision tree
tree = DecisionTreeClassifier(random_state=1234)
tree.fit(X, y)

# Plot the decision tree
plt.figure(figsize=(12, 8))
plot_tree(tree, feature_names=['Like.n', 'Age', 'VisitFrequency', 'Gender'], cla
plt.title('Decision Tree for Cluster 3')
plt.show()
```



Step 8: Selecting (the) Target Segment(s)

In [76]:
```
visit = df.groupby('cluster')['VisitFrequency'].mean()
print(visit)
```

```
cluster
0    2.565714
1    2.711111
2    2.569182
3    2.651667
Name: VisitFrequency, dtype: float64
```

In [77]:
```
like = df.groupby('cluster')['Like.n'].mean()
print(like)
```

```
cluster
0    3.645714
1    6.833333
2    3.110063
3    3.376667
Name: Like.n, dtype: float64
```

In [78]:
```
df['Gender_numeric'] = (df['Gender'] == 0).astype(int)
female = df.groupby('cluster')['Gender_numeric'].mean()
print(female)
```

```
cluster
0    0.491429
1    0.483333
2    0.515723
3    0.606667
Name: Gender_numeric, dtype: float64
```

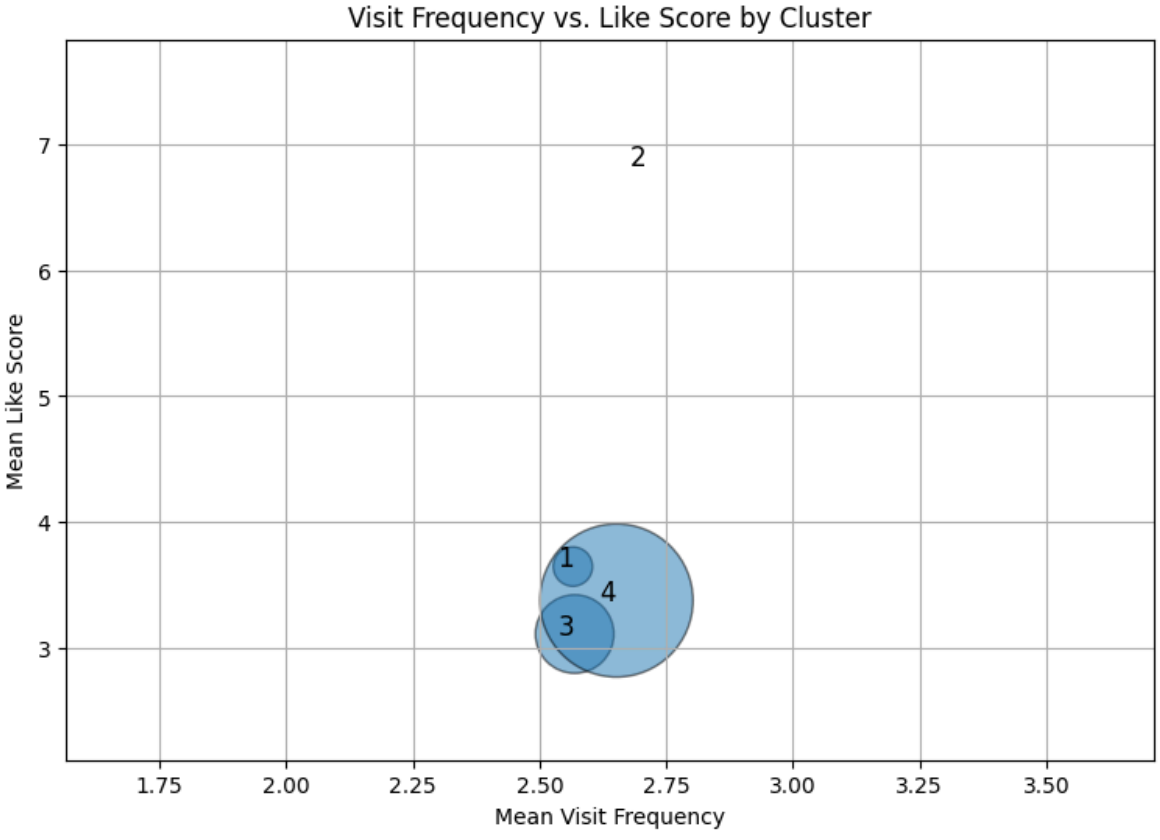
In [79]:
```
df.head()
```

Out[79]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy
0	0	1	0	1	0	1	1	0	1	0
1	1	1	0	1	1	1	1	1	1	0
2	0	1	1	1	1	1	0	1	1	1
3	1	1	0	1	1	1	1	1	0	0
4	0	1	0	1	1	1	1	0	0	1

In [80]:

```
# Normalize bubble sizes
bubble_size = (female - female.min()) / (female.max() - female.min()) * 5000

# Plotting
plt.figure(figsize=(9, 6))
plt.scatter(visit, like, s=bubble_size, alpha=0.5, edgecolor='k')

# Annotate each point with its cluster number
for i in range(len(visit)):
    plt.text(visit.iloc[i], like.iloc[i], str(i + 1), fontsize=12, ha='right')

plt.xlim(visit.min() - 1, visit.max() + 1)
plt.ylim(like.min() - 1, like.max() + 1)
plt.xlabel('Mean Visit Frequency')
plt.ylabel('Mean Like Score')
plt.title('Visit Frequency vs. Like Score by Cluster')
plt.grid(True)
plt.show()
```



Step 9: Customising the Marketing Mix

McDonald's can target segment 3—young, price-sensitive customers who find McDonald's food tasty but expensive—by introducing the MCSUPERBUDGET line. This strategy focuses on adjusting the Price to meet their expectations and could foster loyalty as these customers' incomes rise. To differentiate from the main product range and avoid cannibalization, the Product features will be unique. Effective Promotion channels will need to be identified based on segment 3's media use, while Place will remain consistent with McDonald's outlets, possibly incorporating a separate lane for MCSUPERBUDGET to manage queue times.

Step 10: Evaluation and Monitoring

After the market segmentation analysis is completed, and all strategic and tactical marketing activities have been undertaken, the success of the market segmentation strategy has to be evaluated, and the market must be carefully monitored on a continuous basis. It is possible, for example, that members of segment 3 start earning more money and the MCSUPERBUDGET line is no longer suitable for them. Changes can occur within existing market segments. But changes can also occur in the larger marketplace, for example, if new competitors enter the market. All potential sources of change have to be monitored in order to detect changes which require McDonald's management to adjust their strategic or tactical marketing in view of new market circumstances