Name – Yashika Tirkey (Solo)

Github link --- https://github.com/yashikart/McDonalds-Case-Study

Marketing involves matching consumer needs with supplier offerings through strategic and tactical planning. Strategic decisions set long-term objectives, determining target consumers and organizational image. Tactical planning addresses short-term actions like product, price, place, and promotion. The success asymmetry highlights the foundational importance of good strategic marketing. Tactical excellence cannot compensate for a flawed strategy.

Market segmentation is a crucial tool for marketing managers to choose a target market and design an effective marketing mix. It involves dividing a diverse market into smaller, more homogenous segments based on consumer characteristics. Successful firms prioritize segmentation for marketing success. Segmentation criteria can include factors like age, gender, or consumer behaviours. Implementing a well-thought-out segmentation strategy can lead to a better understanding of consumer differences, improved alignment of organizational strengths with consumer needs, and potentially a long-term competitive advantage. However, it requires a significant investment of time and resources, and a poorly executed strategy can result in wasted resources and expenses. Therefore, organizations must carefully decide whether to undertake market segmentation.

Market segmentation analysis involves grouping consumers based on similar product preferences or characteristics. It has three layers:

- Core Layer (Technical): Involves extracting market segments through statistical methods. It requires competent data analysts and user involvement.
- Enabling Layer (Technical): Tasks include collecting good data, exploring data, and profiling and describing segments. These technical steps ensure the quality of market segmentation.
- Implementation Layer (Non-Technical): Focuses on organizational decisions, including assessing if segmentation brings opportunities, committing to a long-term strategy, and making user-driven choices on target segments and marketing plans.

Two approaches to market segmentation are based on organizational constraints and the nature of segmentation variables. Organizational constraints include quantitative survey-based, creation of segments from existing classifications, and emergence from qualitative research. Segmentation based on variables can be unidimensional (using one variable) or multidimensional (using multiple variables).

Data-driven segmentation can be based on natural segments, reproducible segments, or constructive segments. Conducting data structure analysis before segmentation helps understand whether natural or constructive segmentation is needed.

A step-by-step approach involves deciding to segment, specifying the ideal target segment, collecting and exploring data, extracting segments, profiling and describing segments, selecting target segments, developing a marketing mix, and evaluating and monitoring success. The steps are the same for commonsense and data-driven segmentation, with data-driven segmentation requiring additional decisions.

Deciding to pursue market segmentation is like committing to a long-term relationship. It requires a willingness to make significant changes and investments. There are costs involved, including research and designing, and the expected increase in sales should justify these expenses. Changes may be needed in products, pricing, and communication. The decision needs top-level executive approval and continuous communication across the organization.

Implementing market segmentation faces barriers. Senior management must actively support it and allocate resources. Organizational culture, resistance to change, and lack of training can hinder success. A qualified marketing team is crucial. Financial constraints or inability to make necessary changes are obstacles. The process needs clear objectives, planning, and a structured approach. Overcoming these barriers requires dedication, patience, and a clear sense of purpose.

In the second step of market segmentation analysis, user input is crucial throughout various stages. The organization must contribute conceptually in Step 2, determining knock-out criteria and attractiveness criteria for potential market segments. Knock-out criteria are non-negotiable essentials like homogeneity and reachability, automatically eliminating unsuitable segments. Attractiveness criteria, a diverse set, are evaluated for each segment. The organization's strengths and segment size are factors. A structured process involves creating a segment evaluation plot, considering attractiveness against organizational competitiveness. The segmentation team collaborates, and criteria are weighted based on importance. This groundwork helps streamline data collection and target segment selection in later steps.

Segmentation Variables:

In market segmentation, empirical data is fundamental for both commonsense and data-driven segmentation approaches. The term "segmentation variable" is used to refer to the variable in empirical data that is utilized in commonsense segmentation to divide the sample into market segments.

Commonsense Segmentation:

In commonsense segmentation, a single characteristic of consumers is typically used as the segmentation variable. For example, gender might be used to split the sample into segments of men and women. Other personal characteristics, such as age, number of vacations taken, and benefits sought, serve as descriptor variables to describe these segments in detail.

Data-Driven Segmentation:

Data-driven segmentation, on the other hand, is based on multiple segmentation variables. This approach uses these variables to identify naturally existing or artificially created market segments. For instance, instead of gender, benefits sought could be used as segmentation variables, revealing segments based on preferences rather than demographics.

hese examples underscore the importance of data quality for developing valid segmentation solutions. Whether through commonsense or data-driven approaches, the correct assignment of individuals to segments and the accurate description of these segments rely on the quality of the empirical data.

Sources of Empirical Data:

Empirical data for segmentation studies can be obtained from various sources, including:

1. **Survey Studies:** Common but can be unreliable in reflecting behavior, especially for socially desirable actions.
2. **Observations:** Scanner data, linked to individual customer purchase history via loyalty programs.

3. **Experimental Studies:** Controlled experiments designed to gather specific behavioral data.

The choice of the data source should reflect consumer behavior as closely as possible.

Segmentation Criteria:

Before data is collected, organizations must decide on the segmentation criterion. The common criteria include:

1. **Geographic Segmentation:** Based on the consumer's location.
2. **Socio-Demographic Segmentation:** Utilizes criteria like age, gender, income.
3. **Psychographic Segmentation:** Groups based on beliefs, interests, preferences.
4. **Behavioral Segmentation:** Focuses on similarities in behavior or reported behavior.

The choice of criterion depends on the nature of the market, and the recommendation is to use the simplest possible approach that works effectively for the product or service. Bock and Uncles (2002) highlight relevant differences between consumers, including profitability, bargaining power, preferences, barriers to choice, and interaction effects, as key considerations for segmentation

Geographic Segmentation:

- Involves using the consumer's location of residence as the segmentation criterion.
- Advantages include easy assignment to geographic units for targeted communication.
- Disadvantages include the assumption that living in the same area doesn't necessarily imply shared preferences or behaviors.

Socio-Demographic Segmentation:

- Utilizes criteria like age, gender, income, and education.
- Common in industries like luxury goods, cosmetics, baby products, etc.
- often provides an explanation for specific product preferences but may not capture the underlying reasons for consumer behaviour.

Psychographic Segmentation:

- Involves grouping people based on psychological criteria like beliefs, interests, preferences, or benefits sought.
- More complex than other criteria and often uses multiple variables. Offers insights into the underlying reasons for consumer behaviour, such as travel motives.
- Requires careful consideration of the reliability and validity of measures used.

Behavioral Segmentation:

- Focuses on similarities in actual behavior or reported behavior.
- Uses variables like prior experience, purchase frequency, amount spent, or brand choice behavior.
- Directly aligns with the behavior of interest, but data may not always be readily available.

Data from Survey Studies:

- Most market segmentation analyses use survey data, which is cost-effective and easily collected.
- Data quality is crucial for valid segmentation solutions.
- Empirical data can come from various sources, including surveys, observations, and experimental studies.
- Careful selection of variables is essential to avoid noise and irrelevant information.

Choice of Variables:

- Selecting segmentation variables is critical for both commonsense and data-driven segmentation.
- In data-driven segmentation, all relevant variables must be included, avoiding unnecessary ones to prevent respondent fatigue and data noise.

Response Options:

- Different response options (binary, ordinal, metric) impact the type of data available for analysis.
- Binary or metric options are preferable for segmentation analysis.

Response Styles:

- Biases in survey responses, such as extreme or agreeable responses, can impact segmentation results.
- Minimizing the risk of capturing response styles is essential.

Sample Size:

- Sufficient sample size is crucial for accurate segmentation results.
- Sample size recommendations vary, but a common guideline is at least 100 respondents per segmentation variable.

Considerations for Quality Data:

- Optimal data for segmentation should include all necessary items, be free of unnecessary or correlated items, have high-quality responses, and be free of response styles.
- A suitable sample and sufficient sample size (at least 100 respondents per segmentation variable) are critical.

Hierarchical clustering is a method of grouping data in a tree-like structure, known as a dendrogram. There are two main approaches to hierarchical clustering: divisive and agglomerative.

Divisive Hierarchical Clustering:

Process: Start with the entire dataset and divide it into two segments. This process continues recursively, splitting each segment into two until each observation forms its own segment.

Result: Produces a sequence of nested partitions, from one large segment containing all observations to n segments, where each segment contains a single observation.

Agglomerative Hierarchical Clustering:

Process: Start with each observation as its own segment. Merge the two closest segments iteratively until the entire dataset forms one large segment.

Result: Similar to divisive clustering, it generates a sequence of nested partitions.

Linkage Methods:

Single Linkage: Measures the distance between the closest observations of two sets.

Complete Linkage: Measures the distance between the farthest observations of two sets.

Average Linkage: Computes the mean distance between observations of the two sets.

Ward's Method: Based on squared Euclidean distances, joining sets with the minimal weighted squared Euclidean distance between cluster centers. In the context of clustering, the term "distance" refers to the dissimilarity or similarity between observations. The choice of distance measure and linkage method depends on the characteristics of the data. Different combinations can reveal different features of the data.

Single Linkage : Suitable for revealing non-convex, non-linear structures. Capable of identifying chained effects in less well-separated clusters.

Average and Complete Linkage: Tend to produce more compact clusters.

Ward's Method: Uses squared Euclidean distances and minimizes the variance within clusters.

Dendrogram:

The result of hierarchical clustering is often presented as a dendrogram, a tree diagram.

The root represents one large segment, and leaves represent individual observations.

The height of branches corresponds to the distance between clusters, with higher branches indicating more distinct segments.

```
import pandas as pd
```

```
data = pd.read_csv('/content/vacation_complete_dataset.csv')
```

```
data2 = data
```

```
data
```

	Gender	Age	Education	Income	Income2	Occupation	State	Relationship.Status	Obligation	Obligation2	...	entertainment facilities	ca abo pri
0	Female	25	6.0	$30,001 to $60,000	30-60k	Clerical or service worker	VIC	single	4.800000	Q4	...	no	
1	Female	31	8.0	$120,001 to $150,000	>120k	professional	WA	married	3.300000	Q1	...	no	
2	Male	21	3.0	$90,001 to $120,000	90-120k	NaN	NSW	single	3.400000	Q2	...	no	
3	Female	18	2.0	$30,001 to $60,000	30-60k	unemployed	NSW	single	2.633333	Q1	...	yes	
4	Male	61	3.0	Less than $30,000	<30k	retired	WA	married	3.400000	Q2	...	no	
...	...	...	...	...	...	...	...	...	...	...	...	...	
995	Male	51	3.0	$30,001 to $60,000	30-60k	manager or administrator	VIC	separated or divorced	2.366667	Q1	...	yes	
996	Male	58	4.0	$60,001 to $90,000	60-90k	small business owner	WA	living with a partner	4.400000	Q4	...	no	
997	Male	41	8.0	$60,001 to $90,000	60-90k	professional	QLD	married	3.866667	Q3	...	no	
998	Female	42	3.0	NaN	NaN	professional	VIC	living with a partner	4.000000	Q3	...	no	
999	Female	32	6.0	$120,001 to $150,000	>120k	manager or administrator	WA	living with a partner	2.333333	Q1	...	no	

1000 rows × 32 columns

```
data.columns
```

```
Index(['Gender', 'Age', 'Education', 'Income', 'Income2', 'Occupation',
       'State', 'Relationship.Status', 'Obligation', 'Obligation2', 'NEP',
       'Vacation.Behaviour', 'rest and relax', 'luxury / be spoilt',
       'do sports', 'excitement, a challenge', 'not exceed planned budget',
       'realise creativity', 'fun and entertainment', 'good company',
       'health and beauty', 'free-and-easy-going', 'entertainment facilities',
       'not care about prices', 'life style of the local people',
       'intense experience of nature', 'cosiness/familiar atmosphere',
       'maintain unspoilt surroundings', 'everything organised',
       'unspoilt nature/natural landscape', 'cultural offers',
       'change of surroundings'],
      dtype='object')
```

```
data.shape
```

```
(1000, 32)
```

```
data['Gender'].value_counts()
```

```
Male      512
Female    488
Name: Gender, dtype: int64
```

```
data['Age'].describe()
```

```
count    1000.000000
mean       44.168000
std        14.539228
min        18.000000
25%        32.000000
50%        42.000000
75%        57.000000
max       105.000000
Name: Age, dtype: float64
```

```
data['Income'].value_counts()
```

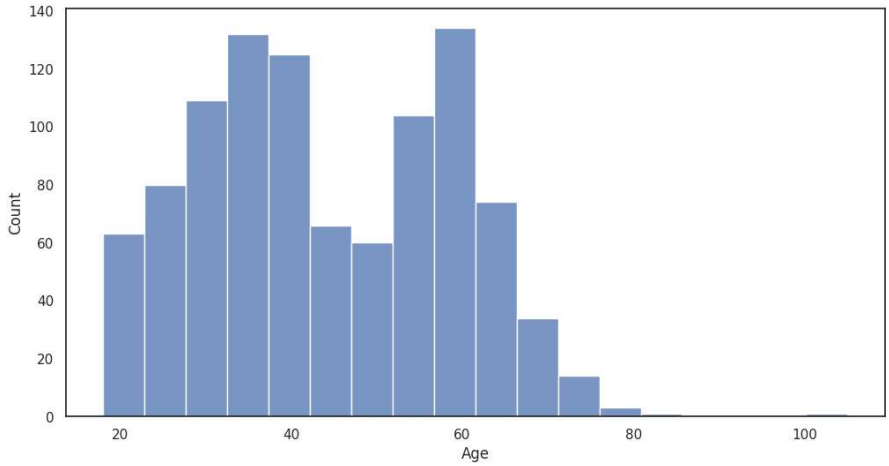
```
$30,001 to $60,000      265
$60,001 to $90,000      233
Less than $30,000       150
$90,001 to $120,000     146
$120,001 to $150,000     72
$150,001 to $180,000     32
$180,001 to $210,000     15
more than $240,001       11
$210,001 to $240,000     10
Name: Income, dtype: int64
```

```
data['Income2'].value_counts()
```

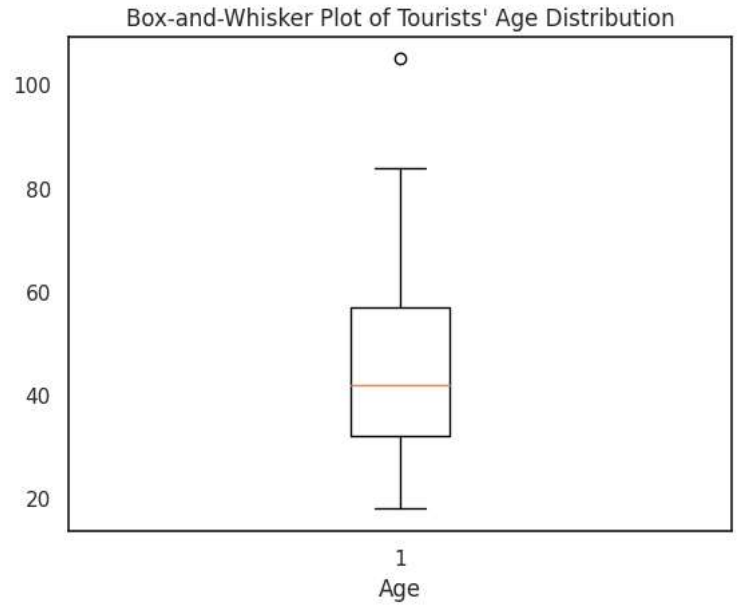
```
30-60k     265
60-90k     233
<30k       150
90-120k    146
>120k      140
Name: Income2, dtype: int64
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

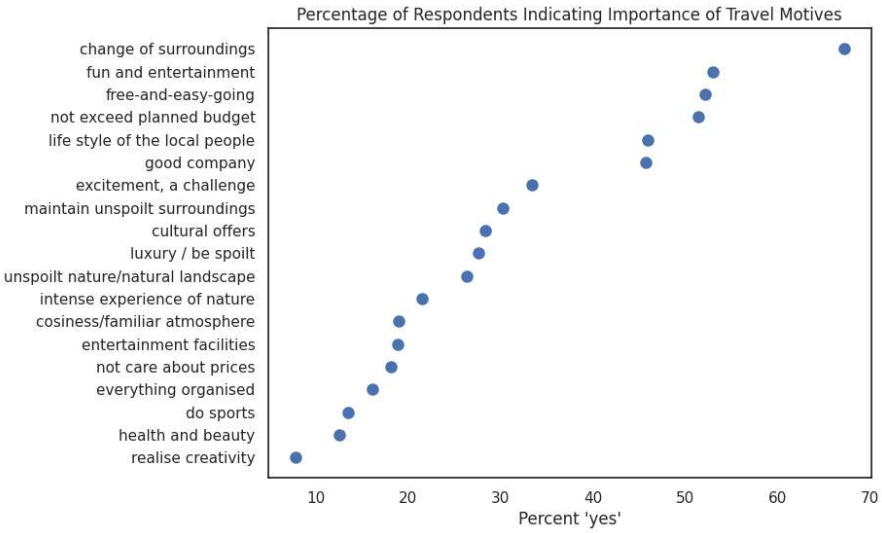
```
fig, axes = plt.subplots(figsize=(12,6))
sns.histplot(data['Age'])
fig, axes = plt.subplots(1, figsize=(12,6))
sns.histplot(data['Age'],bins=50, kde=True, stat='density')
plt.title('Histograms of tourists’ age in the Australian travel motives data set')
plt.show()
```




```
plt.boxplot(data['Age'])
plt.xlabel('Age')
plt.title('Box-and-Whisker Plot of Tourists\' Age Distribution')
plt.show()
```

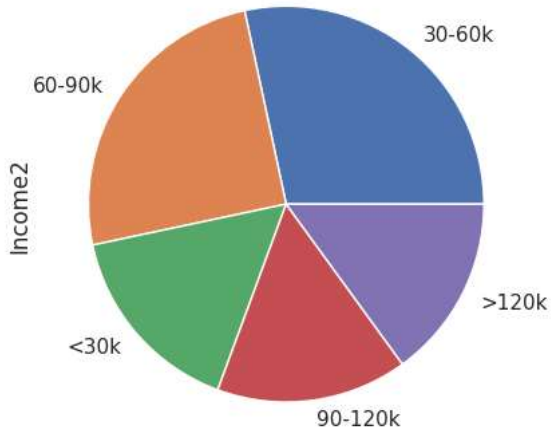



```
travel_motives = data.iloc[:, 13:32]
percentage_yes = (travel_motives == 'yes').mean() * 100
sorted_percentages = percentage_yes.sort_values()
plt.figure(figsize=(8, 6))
plt.plot(sorted_percentages, range(len(sorted_percentages)), 'o', markersize=8)
plt.xlabel("Percent 'yes'")
plt.yticks(range(len(sorted_percentages)), sorted_percentages.index)
plt.title('Percentage of Respondents Indicating Importance of Travel Motives')

plt.show()
```




```
data['Income2'].value_counts().plot(kind='pie')
```

```
<Axes: ylabel='Income2'>
```




```
for i in data.iloc[:, 12:]:
  data[i] = data[i].apply(lambda x: 0 if x=='no' else 1)

data['Gender'] = data['Gender'].apply(lambda x: 0 if x=='Female' else 1)


data['Occupation'] = pd.Categorical(data['Occupation'])
data['Occupation'] = data['Occupation'].cat.codes

data['State'] = pd.Categorical(data['State'])
data['State'] = data['State'].cat.codes

data['Relationship.Status'] = pd.Categorical(data['Relationship.Status'])
data['Relationship.Status'] = data['Relationship.Status'].cat.codes


data.isnull().sum()
```

```
Gender                            0
Age                               0
Education                         8
Income                           66
Income2                          66
Occupation                        0
State                             0
```

```
Relationship.Status                  0
Obligation                           0
Obligation2                          0
NEP                                  0
Vacation.Behaviour                  25
rest and relax                       0
luxury / be spoilt                   0
do sports                            0
excitement, a challenge              0
not exceed planned budget            0
realise creativity                   0
fun and entertainment                0
good company                         0
health and beauty                    0
free-and-easy-going                  0
entertainment facilities             0
not care about prices                0
life style of the local people       0
intense experience of nature         0
cosiness/familiar atmosphere         0
maintain unspoilt surroundings       0
everything organised                 0
unspoilt nature/natural landscape    0
cultural offers                      0
change of surroundings               0
dtype: int64
```

```
data['Vacation.Behaviour'] = data['Vacation.Behaviour'].fillna(data['Vacation.Behaviour'].mean())
```

```
data['Education'] = data['Education'].fillna(0)
```

```
data['Income2'] = data['Income2'].fillna(0)
```

```
data['Income2'].unique()
```

```
array(['30-60k', '>120k', '90-120k', '<30k', '60-90k', 0], dtype=object)
```

```
category_mapping = { 0: 0, '<30k': 1, '30-60k': 2, '60-90k':3, '90-120k':4, '>120k':5}
data['Income2'] = data['Income2'].map(category_mapping)
```

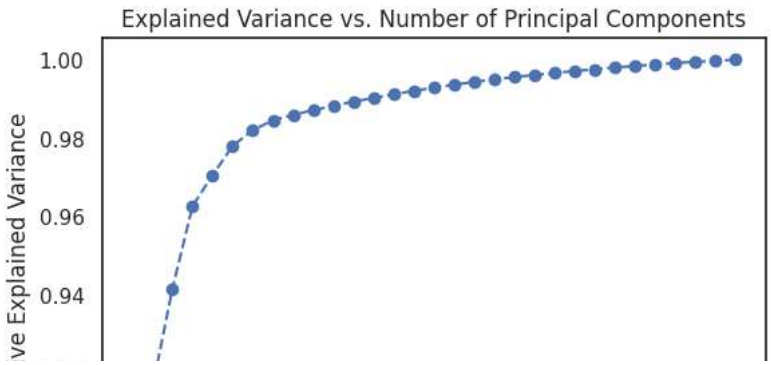
```
data['Obligation2'] = pd.Categorical(data['Obligation2'])
data['Obligation2'] = data['Obligation2'].cat.codes
```

```
data = data.drop('Income', axis=1)
```

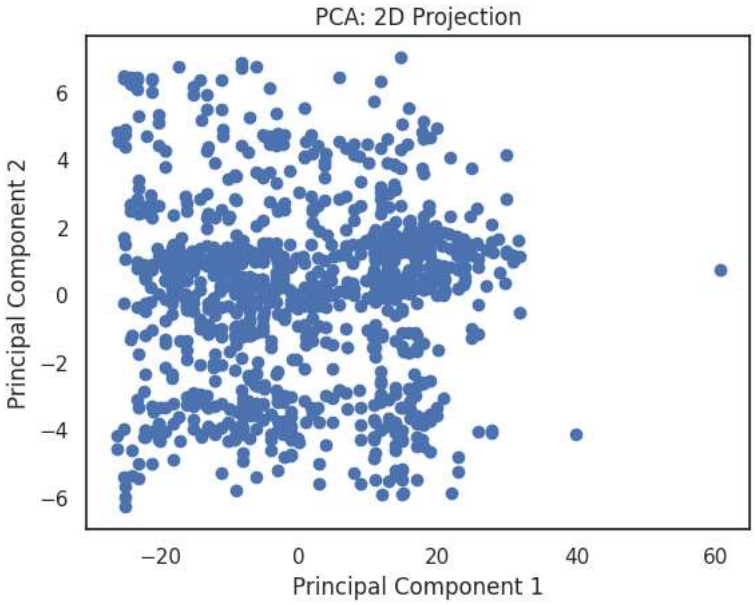
```
from sklearn.decomposition import PCA
pca = PCA()
pca.fit(data)
```

```
PCA
PCA()
```

```
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_explained_variance = explained_variance_ratio.cumsum()

plt.plot(range(1, len(explained_variance_ratio) + 1), cumulative_explained_variance, marker='o', linestyle='--')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance vs. Number of Principal Components')
plt.show()
```

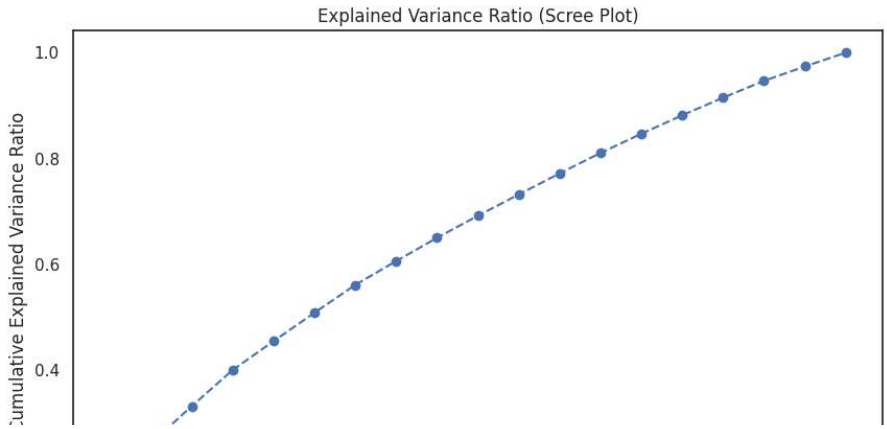



```
n_components = (cumulative_explained_variance <= 0.95).sum() + 1
pca = PCA(n_components=n_components)
df_pca = pca.fit_transform(data)
plt.scatter(df_pca[:, 0], df_pca[:, 1])
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA: 2D Projection')
plt.show()
```



```
data_for_pca = data.iloc[:, 12:32]

# Standardize the data (optional but often recommended for PCA)
data_for_pca_standardized = (data_for_pca - data_for_pca.mean()) / data_for_pca.std()

# Fit PCA
pca = PCA()
principal_components = pca.fit_transform(data_for_pca_standardized)

# Print the explained variance ratio
print("Explained Variance Ratio:")
print(pca.explained_variance_ratio_)

# Plotting the Scree Plot (explained variance)
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), pca.explained_variance_ratio_.cumsum(), marker='o', linestyle='--')
plt.title('Explained Variance Ratio (Scree Plot)')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.show()

# Plotting the data in 2D using the first two principal components
plt.figure(figsize=(10, 6))
plt.scatter(principal_components[:, 0], principal_components[:, 1], alpha=0.8)
plt.title('PCA: Principal Components 1 and 2')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')

# Display variable loadings for PC1 and PC2
loadings = pca.components_[:2, :]
for i, variable in enumerate(data_for_pca.columns):
    plt.text(loadings[0, i], loadings[1, i], variable, color='r')

plt.show()
```

```
Explained Variance Ratio:
[0.17078897 0.09081401 0.07046054 0.06819256 0.05450082 0.05361201
 0.05278774 0.04489111 0.04406192 0.04169233 0.04021179 0.03982593
 0.03864189 0.03618819 0.03503456 0.03321271 0.03180739 0.02717633
 0.02609922]
```




```
af_ds = {
    'Name': ['Anna','Bill','Frank','Julia','Maria','Michael','Tom'],
    'beach': [100,100,60,70,80,0,50],
    'action': [0,0,40,0,0,90,20],
    'culture': [0,0,0,30,20,10,30]
}

af_ds = pd.DataFrame(af_ds)
af_ds
```

	Name	beach	action	culture
0	Anna	100	0	0
1	Bill	100	0	0
2	Frank	60	40	0
3	Julia	70	0	30
4	Maria	80	0	20
5	Michael	0	90	10
6	Tom	50	20	30

euclidean distances

Definition: The Euclidean distance between two points is the straight-line distance between them. It is the length of the shortest path between the two points.

Formula (for two points (x1,y1) and (x2,y2) in 2D space): Euclidean Distance=(x1−x2)2+(y1−y2)2 In general, for n-dimensional space: Euclidean Distance=∑i=1n(xi−yi)2

```
from scipy.spatial.distance import pdist, squareform
numerical_columns = af_ds.drop('Name', axis=1)
euclidean_distances = pdist(numerical_columns, metric='euclidean')
euclidean_distances_matrix = squareform(euclidean_distances)
euclidean_distances_df = pd.DataFrame(euclidean_distances_matrix, columns=af_ds['Name'], index=af_ds['Name']).round(2)

print("Euclidean Distances:")
print(euclidean_distances_df)
```

```
Euclidean Distances:
Name       Anna    Bill  Frank   Julia   Maria  Michael    Tom
Name
Anna       0.00    0.00  56.57   42.43   28.28   134.91  61.64
Bill       0.00    0.00  56.57   42.43   28.28   134.91  61.64
Frank     56.57   56.57   0.00   50.99   48.99    78.74  37.42
Julia     42.43   42.43  50.99    0.00   14.14   115.76  28.28
Maria     28.28   28.28  48.99   14.14    0.00   120.83  37.42
Michael  134.91  134.91  78.74  115.76  120.83     0.00  88.32
Tom       61.64   61.64  37.42   28.28   37.42    88.32   0.00
```

manhattan distances

Definition: The Manhattan distance between two points is the sum of the absolute differences of their coordinates. It is the distance a car would travel along the grid-like streets of a city to reach the destination.

Formula (for two points (x1,y1) and (x2,y2) in 2D space): Manhattan Distance=|x1−x2|+|y1−y2| In general, for n-dimensional space: Manhattan Distance=∑i=1n|xi−yi|

```
manhattan_distances = pdist(numerical_columns, metric='cityblock')  # 'cityblock' is the Manhattan distance
manhattan_distances_matrix = squareform(manhattan_distances)
manhattan_distances_df = pd.DataFrame(manhattan_distances_matrix, columns=af_ds['Name'], index=af_ds['Name'])

print("Manhattan Distances:")
print(manhattan_distances_df)
```

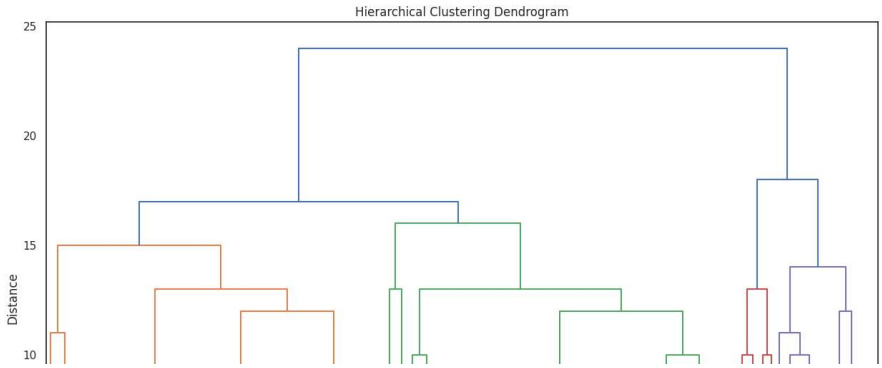
```
Manhattan Distances:
Name      Anna   Bill  Frank  Julia  Maria  Michael    Tom
Name
Anna       0.0    0.0   80.0   60.0   40.0    200.0  100.0
Bill       0.0    0.0   80.0   60.0   40.0    200.0  100.0
Frank     80.0   80.0    0.0   80.0   80.0    120.0   60.0
Julia     60.0   60.0   80.0    0.0   20.0    180.0   40.0
Maria     40.0   40.0   80.0   20.0    0.0    180.0   60.0
Michael  200.0  200.0  120.0  180.0  180.0      0.0  140.0
Tom      100.0  100.0   60.0   40.0   60.0    140.0    0.0
```

Hierarchical clustering is a method of grouping data in a tree-like structure, known as a dendrogram. There are two main approaches to hierarchical clustering: divisive and agglomerative.

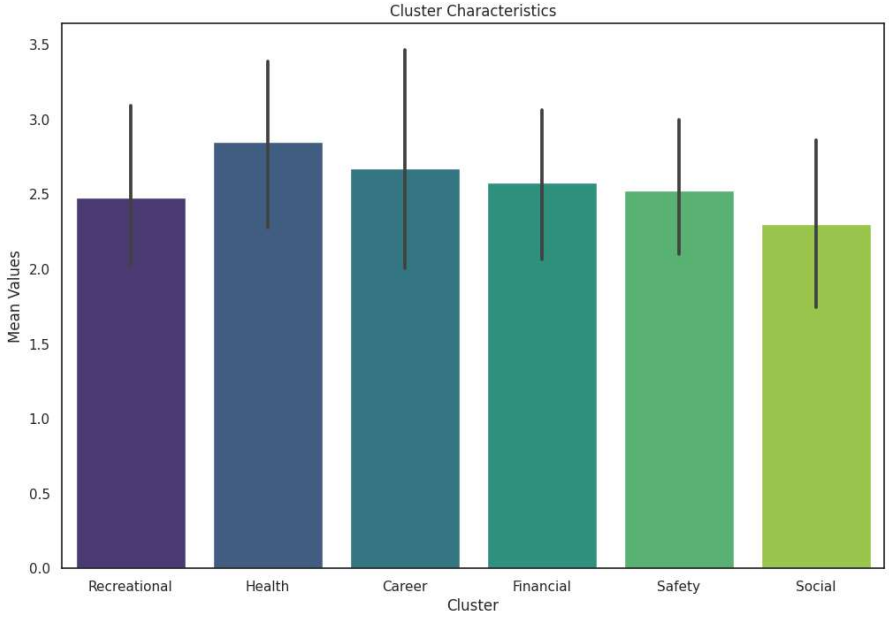
Divisive Hierarchical Clustering: 1. Process: Start with the entire dataset and divide it into two segments. This process continues recursively, splitting each segment into two until each observation forms its own segment. 2. Result: Produces a sequence of nested partitions, from one large segment containing all observations to n segments, where each segment contains a single observation. **Agglomerative Hierarchical Clustering:** 1. Process: Start with each observation as its own segment. Merge the two closest segments iteratively until the entire dataset forms one large segment. 2. Result: Similar to divisive clustering, it generates a sequence of nested partitions.

```
risk = pd.read_csv('/content/risk.csv')
```

```
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
link_matrix = linkage(risk, method='complete', metric='cityblock')  # 'cityblock' corresponds to Manhattan distance
plt.figure(figsize=(15, 10))
dendrogram(link_matrix, labels=risk.index, orientation='top', leaf_rotation=0, leaf_font_size=10)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Sample Index')
plt.ylabel('Distance')
plt.show()
```

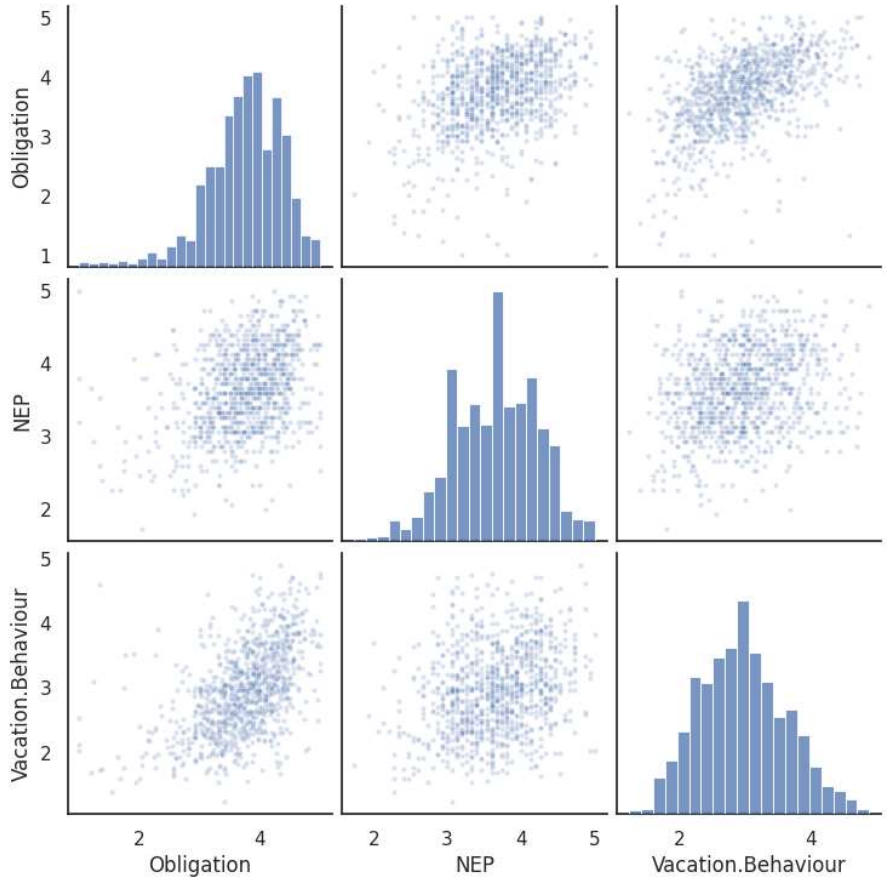



```
num_clusters = 6
clusters = fcluster(link_matrix, t=num_clusters, criterion='maxclust')
cluster_means = risk.groupby(clusters).mean()
total_means = risk.mean().to_frame().T
total_means.index = [0]
cluster_means = pd.concat([total_means, cluster_means])
plt.figure(figsize=(12, 8))
sns.barplot(data=cluster_means, palette='viridis')
plt.title('Cluster Characteristics')
plt.xlabel('Cluster')
plt.ylabel('Mean Values')
plt.show()
```

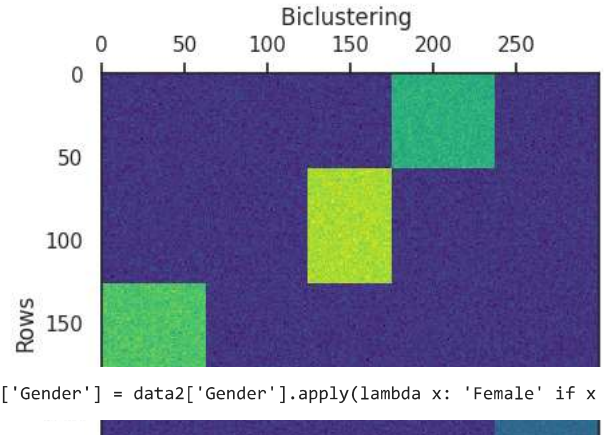


Bagged clustering also combines hierarchical clustering algo- rithms and partitioning clustering algorithms, but adds bootstrapping (Efron and Tibshirani 1993). Bootstrapping can be implemented by random drawing from the data set with replacement. That means that the process of extracting segments is repeated many times with randomly drawn (bootstrapped) samples of the data. Bootstrapping has the advantage of making the final segmentation solution less dependent on the exact people contained in consumer data.

```
v_data = data[['Obligation', 'NEP', 'Vacation.Behaviour']]
v_data = v_data.dropna()
sns.set(style="white")
sns.pairplot(v_data, markers='.', plot_kws={'alpha':0.2})
plt.show()
```



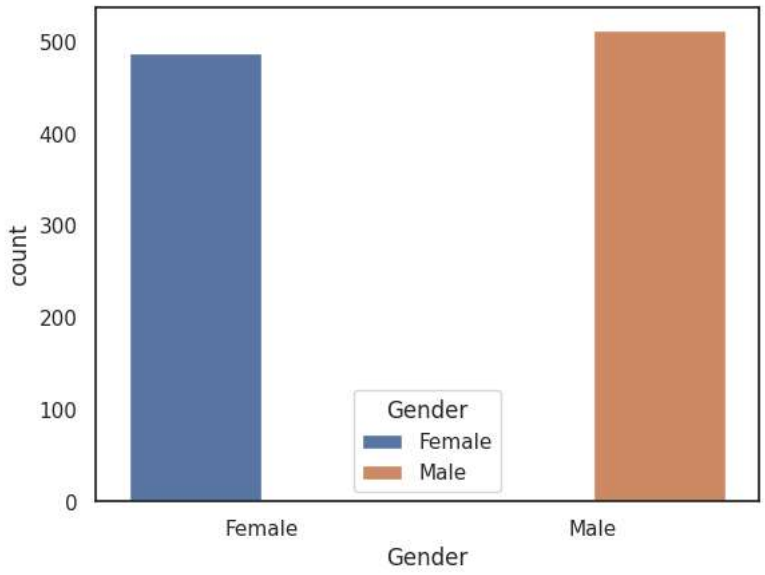

```
winterActiv_df = pd.read_csv('/content/1997_98_27_activities.csv')
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_biclusters
from sklearn.cluster import SpectralBiclustering

winterActiv_df, rows, columns = make_biclusters(shape=(300, 300), n_clusters=5, noise=5, random_state=42)

model = SpectralBiclustering(n_clusters=5, method='log', random_state=0)
model.fit(winterActiv_df)

fit_data = winterActiv_df[np.argsort(model.row_labels_), :]
fit_data = fit_data[:, np.argsort(model.column_labels_)]

plt.matshow(fit_data, cmap='viridis')
plt.title('Biclustering ')
plt.xlabel('Columns')
plt.ylabel('Rows')
plt.show()
```




```
data2['Gender'] = data2['Gender'].apply(lambda x: 'Female' if x == 0 else 'Male')
```

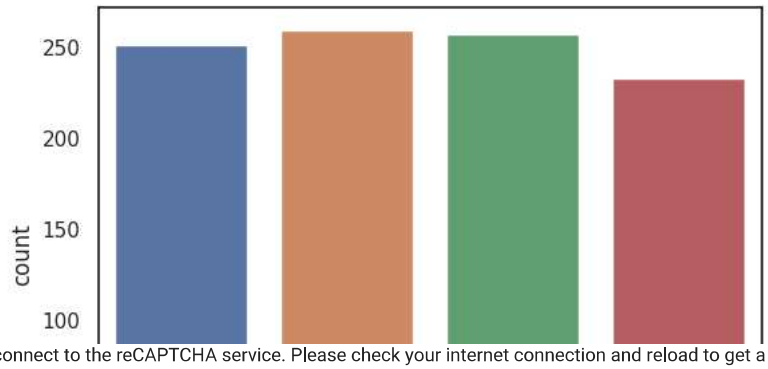
```
sns.countplot(x=data2['Gender'], hue= data2['Gender'])
```

```
<Axes: xlabel='Gender', ylabel='count'>
```




```
sns.countplot(x=data2['Obligation2'])
```

```
<Axes: xlabel='Obligation2', ylabel='count'>
```



Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.

McDonalds Case Study in Python

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Exploring Data

```
data = pd.read_csv('/content/mcdonalds.csv')
```

```
data.shape
```

```
(1453, 15)
```

```
data.head()
```

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

```
data.isnull().sum()
```

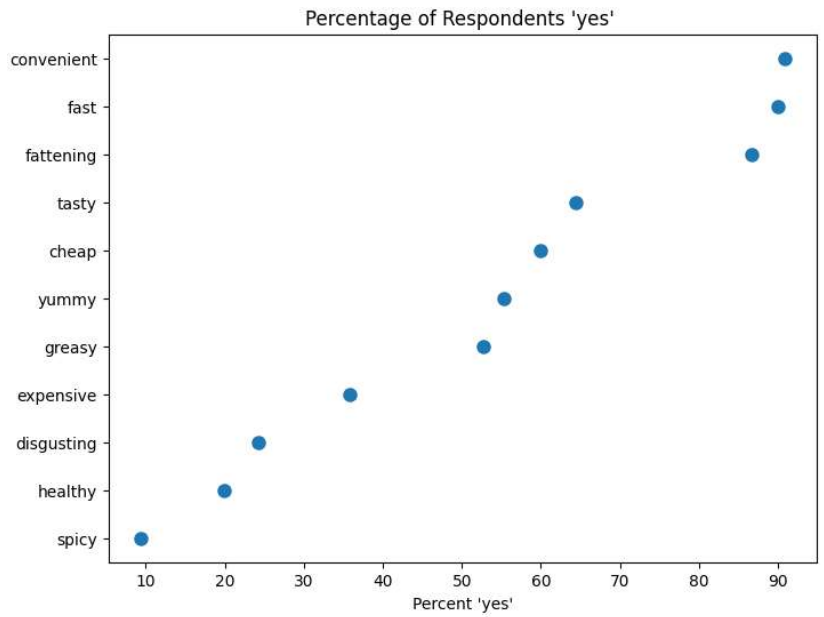
```
yummy             0
convenient        0
spicy             0
fattening         0
greasy            0
fast              0
cheap             0
tasty             0
expensive         0
healthy           0
disgusting        0
Like              0
Age               0
VisitFrequency    0
Gender            0
dtype: int64
```

```
data.columns
```

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

Double-click (or enter) to edit

```
data_check = data.iloc[:, 0:11]
percentage_yes = (data_check == 'Yes').mean() * 100

sorted_percentages = percentage_yes.sort_values()

plt.figure(figsize=(8, 6))
plt.plot(sorted_percentages, range(len(sorted_percentages)), 'o', markersize=8)


plt.xlabel("Percent 'yes'")
plt.yticks(range(len(sorted_percentages)), sorted_percentages.index)
plt.title("Percentage of Respondents 'yes' " )

plt.show()
```



Double-click (or enter) to edit

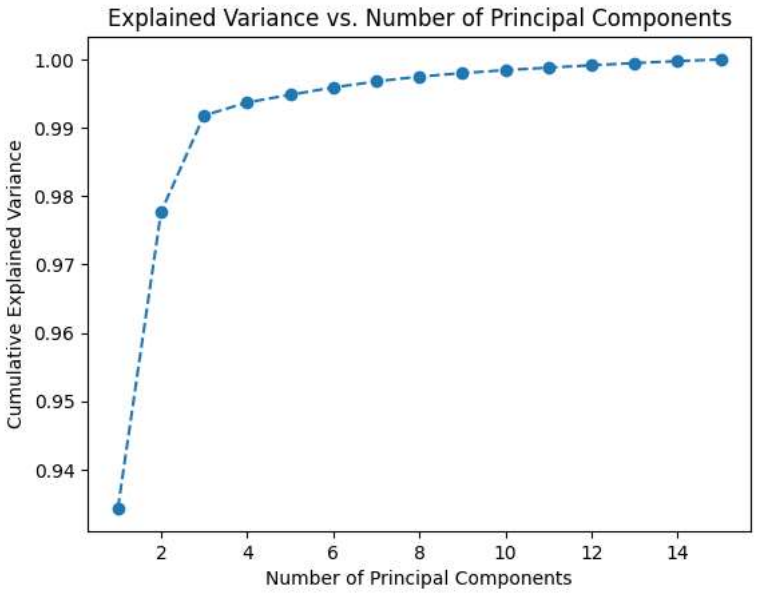
```
for i in data_check.columns:
  data[i] = data[i].apply(lambda x: 1 if x=='Yes' else 0)

data['Gender'] = data['Gender'].apply(lambda x: 1 if x=='Female' else 0)
```

```
data['VisitFrequency'] = pd.Categorical(data['VisitFrequency'])
data['VisitFrequency'] = data['VisitFrequency'].cat.codes
```

```
sentiment_mapping = {'I hate it!-5': -5, '-4': -4, '-3': -3, '-2': -2, '-1': -1, '0': 0, '+1': 1, '+2': 2, '+3': 3, '+4': 4, 'I l
data['Like'] = data['Like'].map(sentiment_mapping)
```

Principal components analysis of the fast food data set

```
from sklearn.decomposition import PCA
pca = PCA()
pca.fit(data)
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_explained_variance = explained_variance_ratio.cumsum()

plt.plot(range(1, len(explained_variance_ratio) + 1), cumulative_explained_variance, marker='o', linestyle='--')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance vs. Number of Principal Components')
plt.show()
```



Double-click (or enter) to edit

```
print("Standard Deviation:", pca.explained_variance_)
print("Proportion of Variance:", pca.explained_variance_ratio_)
print("Cumulative Proportion:", pca.explained_variance_ratio_.cumsum())
```

```
Standard Deviation: [2.02938126e+02 9.40928005e+00 3.07233920e+00 4.12969463e-01
 2.44117821e-01 2.32443346e-01 1.90784372e-01 1.55751188e-01
 1.09391425e-01 9.57335597e-02 7.94100817e-02 7.49976851e-02
 6.99665992e-02 6.16968855e-02 5.56552803e-02]
Proportion of Variance: [9.34326142e-01 4.33202794e-02 1.41450347e-02 1.90130939e-03
 1.12391725e-03 1.07016803e-03 8.78370318e-04 7.17077706e-04
 5.03637587e-04 4.40756841e-04 3.65603628e-04 3.45288976e-04
 3.22125881e-04 2.84052159e-04 2.56236638e-04]
Cumulative Proportion: [0.93432614 0.97764642 0.99179146 0.99369276 0.99481668 0.99588685
 0.99676522 0.9974823  0.99798594 0.99842669 0.9987923  0.99913759
 0.99945971 0.99974376 1.        ]
```

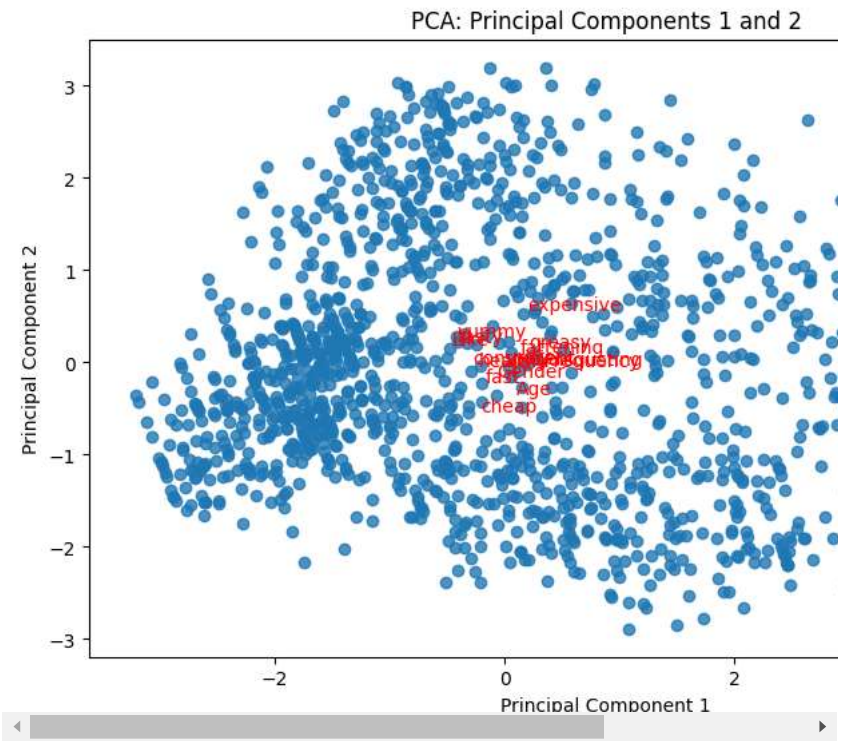
```
comp = pca.components_
num_pc = pca.n_features_
pc_list = ["PC"+str(i) for i in list(range(1, num_pc+1))]
loadings_df = pd.DataFrame.from_dict(dict(zip(pc_list, comp)))
loadings_df['variable'] = data.columns.values
loadings_df = loadings_df.set_index('variable')
loadings_df
```

```
usr/local/lib/python3.10/dist-packages/sklearn/utils/deprecation.py:101: FutureWa
 warnings.warn(msg, category=FutureWarning)
```

variable	PC1	PC2	PC3	PC4	PC5	PC6	P
yummy	-0.409077	0.271809	-0.063991	-0.016767	0.134743	0.087489	0.0365
convenient	-0.273774	-0.019944	-0.170822	0.247276	-0.349505	-0.195160	0.0269
spicy	-0.007178	-0.020842	0.154058	0.777782	0.272267	-0.028348	0.2690
fattening	0.134744	0.102769	-0.543648	0.128494	-0.209129	-0.030277	0.1780
greasy	0.211487	0.160652	-0.433328	0.219149	0.319349	0.153647	0.0002
fast	-0.166572	-0.213546	-0.242357	0.272391	-0.197269	-0.294995	-0.0502
cheap	-0.201997	-0.540426	-0.234453	0.013200	0.252451	0.109246	-0.1654
tasty	-0.412073	0.211947	-0.053204	0.076139	0.085718	0.039078	0.0444
expensive	0.200417	0.560809	0.177905	0.091424	0.118757	0.116389	0.1623

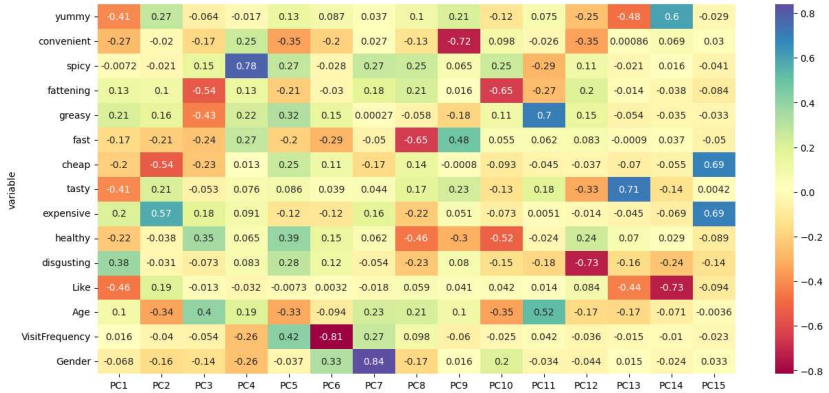
Double-click (or enter) to edit

```
data_for_pca_standardized = (data - data.mean()) / data.std()
principal_components = pca.fit_transform(data_for_pca_standardized)
plt.figure(figsize=(10, 6))
plt.scatter(principal_components[:, 0], principal_components[:, 1], alpha=0.8)
plt.title('PCA: Principal Components 1 and 2')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')

# Display variable loadings for PC1 and PC2
loadings = pca.components_[:2, :]
for i, variable in enumerate(data.columns):
    plt.text(loadings[0, i], loadings[1, i], variable, color='r')

plt.show()
```



Double-click (or enter) to edit

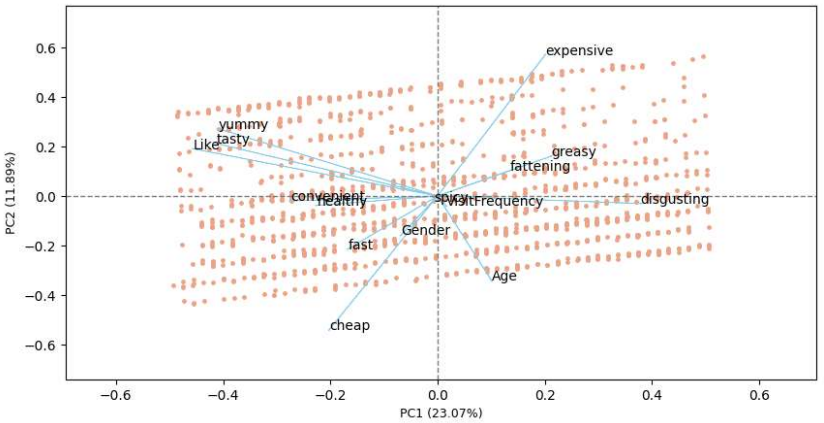
```
plt.figure(figsize=(15, 7))
sns.heatmap(loadings_df, annot=True, cmap='Spectral')
plt.show()
```



Double-click (or enter) to edit

```
from bioinfokit.visuz import cluster
# get PC scores
pca_scores = PCA().fit_transform(data)

# get 2D biplot
cluster.biplot(cscore=pca_scores, loadings=loadings, labels=data.columns.values, var1=round(pca.explained_variance_ratio_[0]*100,
    var2=round(pca.explained_variance_ratio_[1]*100, 2),show=True,dim=(10,5))
```

```
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
```



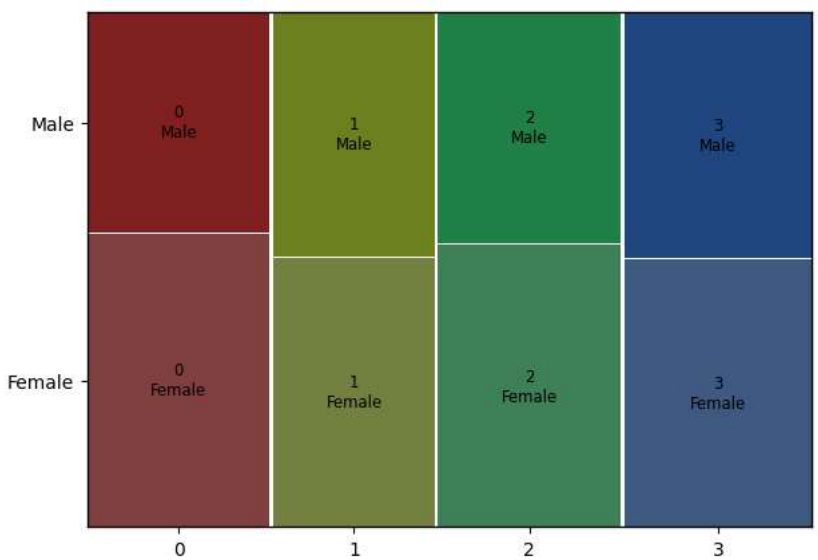
Using k-Means

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0).fit(data)
data['cluster_num'] = kmeans.labels_ #adding to df
print (kmeans.labels_) #Label assigned for each data point
print (kmeans.inertia_) #gives within-cluster sum of squares.
print(kmeans.n_iter_) #number of iterations that k-means algorithm runs to get a minimum within-cluster sum of squares
print(kmeans.cluster_centers_) #Location of the centroids on each cluster.
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will char
  warnings.warn(
[2 0 2 ... 0 3 1]
40236.0792976448
6
[[ 4.15094340e-01  9.13746631e-01  9.43396226e-02  8.49056604e-01
   4.44743935e-01  9.08355795e-01  5.82210243e-01  5.66037736e-01
   3.85444744e-01  1.88679245e-01  2.74932615e-01  2.74932615e-01
   5.10404313e+01  2.59299191e+00  5.71428571e-01]
 [ 7.80487805e-01  9.35975610e-01  5.18292683e-02  9.32926829e-01
   7.34756098e-01  9.32926829e-01  6.28048780e-01  8.04878049e-01
   3.50609756e-01  1.64634146e-01  1.95121951e-01  2.14329268e+00
   2.51310976e+01  2.68292683e+00  5.24390244e-01]
 [ 4.23180593e-01  8.76010782e-01  1.72506739e-01  8.16711590e-01
   4.31266846e-01  8.97574124e-01  6.49595687e-01  5.57951482e-01
   2.66846361e-01  2.07547170e-01  2.31805930e-01 -8.35579515e-02
   6.23530997e+01  2.53369272e+00  5.49865229e-01]
 [ 6.16187990e-01  9.08616188e-01  5.22193211e-02  8.77284595e-01
   5.19582245e-01  8.66840731e-01  5.40469974e-01  6.65796345e-01
   4.25587467e-01  2.29765013e-01  2.63707572e-01  9.26892950e-01
   3.78563969e+01  2.74412533e+00  5.22193211e-01]]
```

```
from statsmodels.graphics.mosaicplot import mosaic
from itertools import product
```

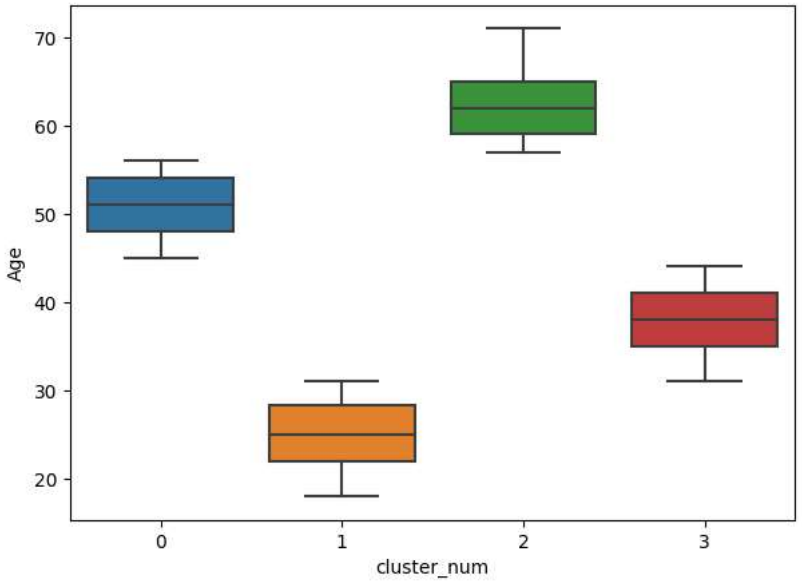
```
data['Gender'] = data['Gender'].apply(lambda x: 'Female' if x==1 else 'Male')
gender =pd.crosstab(data['cluster_num'],data['Gender'])
plt.rcParams['figure.figsize'] = (7,5)
mosaic(gender.stack())
plt.show()
```



Double-click (or enter) to edit

```
sns.boxplot(x="cluster_num", y="Age", data=data)
```

```
<Axes: xlabel='cluster_num', ylabel='Age'>
```



Double-click (or enter) to edit

```
visit = data.groupby('cluster_num')['VisitFrequency'].mean()
visit = visit.to_frame().reset_index()
Like = data.groupby('cluster_num')['Like'].mean()
Like = Like.to_frame().reset_index()
data['Gender'] = data['Gender'].apply(lambda x: 1 if x=='Female' else 0)
Gender = data.groupby('cluster_num')['Gender'].mean()
Gender = Gender.to_frame().reset_index()
new = Gender.merge(Like, on='cluster_num', how='left').merge(visit, on='cluster_num', how='left')
new
```

	cluster_num	Gender	Like	VisitFrequency
0	0	0.571429	0.274933	2.592992
1	1	0.524390	2.143293	2.682927
2	2	0.549865	-0.083558	2.533693