In [2]:
```
pip install bioinfokit
```

```
Requirement already satisfied: bioinfokit in c:\users\an652\appdata\local\programs\p
ython\python313\lib\site-packages (2.1.4)
Requirement already satisfied: pandas in c:\users\an652\appdata\local\programs\pytho
n\python313\lib\site-packages (from bioinfokit) (2.2.3)
Requirement already satisfied: numpy in c:\users\an652\appdata\local\programs\python
\python313\lib\site-packages (from bioinfokit) (2.2.5)
Requirement already satisfied: matplotlib in c:\users\an652\appdata\local\programs\p
ython\python313\lib\site-packages (from bioinfokit) (3.10.3)
Requirement already satisfied: scipy in c:\users\an652\appdata\local\programs\python
\python313\lib\site-packages (from bioinfokit) (1.15.3)
Requirement already satisfied: scikit-learn in c:\users\an652\appdata\local\programs
\python\python313\lib\site-packages (from bioinfokit) (1.6.1)
Requirement already satisfied: seaborn in c:\users\an652\appdata\local\programs\pyth
on\python313\lib\site-packages (from bioinfokit) (0.13.2)
Requirement already satisfied: matplotlib-venn in c:\users\an652\appdata\local\progr
ams\python\python313\lib\site-packages (from bioinfokit) (1.1.2)
Requirement already satisfied: tabulate in c:\users\an652\appdata\local\programs\pyt
hon\python313\lib\site-packages (from bioinfokit) (0.9.0)
Requirement already satisfied: statsmodels in c:\users\an652\appdata\local\programs
\python\python313\lib\site-packages (from bioinfokit) (0.14.4)
Requirement already satisfied: textwrap3 in c:\users\an652\appdata\local\programs\py
thon\python313\lib\site-packages (from bioinfokit) (0.9.2)
Requirement already satisfied: adjustText in c:\users\an652\appdata\local\programs\p
ython\python313\lib\site-packages (from bioinfokit) (1.3.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\an652\appdata\local\prog
rams\python\python313\lib\site-packages (from matplotlib->bioinfokit) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\an652\appdata\local\programs
\python\python313\lib\site-packages (from matplotlib->bioinfokit) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\an652\appdata\local\pro
grams\python\python313\lib\site-packages (from matplotlib->bioinfokit) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\an652\appdata\local\pro
grams\python\python313\lib\site-packages (from matplotlib->bioinfokit) (1.4.8)
Requirement already satisfied: packaging>=20.0 in c:\users\an652\appdata\local\progr
ams\python\python313\lib\site-packages (from matplotlib->bioinfokit) (25.0)
Requirement already satisfied: pillow>=8 in c:\users\an652\appdata\local\programs\py
thon\python313\lib\site-packages (from matplotlib->bioinfokit) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\an652\appdata\local\prog
rams\python\python313\lib\site-packages (from matplotlib->bioinfokit) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\an652\appdata\local
\programs\python\python313\lib\site-packages (from matplotlib->bioinfokit) (2.9.0.po
st0)
Requirement already satisfied: pytz>=2020.1 in c:\users\an652\appdata\local\programs
\python\python313\lib\site-packages (from pandas->bioinfokit) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\an652\appdata\local\progra
ms\python\python313\lib\site-packages (from pandas->bioinfokit) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in c:\users\an652\appdata\local\program
s\python\python313\lib\site-packages (from scikit-learn->bioinfokit) (1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\an652\appdata\local
\programs\python\python313\lib\site-packages (from scikit-learn->bioinfokit) (3.6.0)
Requirement already satisfied: patsy>=0.5.6 in c:\users\an652\appdata\local\programs
\python\python313\lib\site-packages (from statsmodels->bioinfokit) (1.0.1)
Requirement already satisfied: six>=1.5 in c:\users\an652\appdata\local\programs\pyt
hon\python313\lib\site-packages (from python-dateutil>=2.7->matplotlib->bioinfokit)
(1.17.0)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [3]:
```
data=pd.read_csv("mcdonalds.csv")
data1=pd.read_csv("mcdonalds.csv")
data.columns.values.tolist()
```

Out[3]:
```
['yummy',
 'convenient',
 'spicy',
 'fattening',
 'greasy',
 'fast',
 'cheap',
 'tasty',
 'expensive',
 'healthy',
 'disgusting',
 'Like',
 'Age',
 'VisitFrequency',
 'Gender']
```

In [4]:
```
data.shape
```

Out[4]:
```
(1453, 15)
```

In [5]:
```
data.head(3)
```

Out[5]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	d
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 [6]:
```
MD=data.iloc[:,0:11].replace("Yes",1).replace("No",0)
mean=round(MD.mean(),2)
mean
```

```
C:\Users\an652\AppData\Local\Temp\ipykernel_34444\2977006428.py:1: FutureWarning: Do
wncasting behavior in `replace` is deprecated and will be removed in a future versio
n. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. T
o opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting',
True)`
  MD=data.iloc[:,0:11].replace("Yes",1).replace("No",0)
```

Out[6]:
```
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
```

In [7]:
```
pca = PCA()
MD_pca=pca.fit_transform(MD)
MD_p=pca.fit(MD)

SD=np.sqrt(pca.explained_variance_)
PV=pca.explained_variance_ratio_
index=[]
for i in range(len(SD)):
    i=i+1
    index.append("PC{}".format(i))

sum=pd.DataFrame({
    "Standard deviation":SD,"Proportion of Variance":PV,"Cumulative Proportion":PV.
},index=index)
sum
```

Out[7]:

	Standard deviation	Proportion of Variance	Cumulative Proportion
PC1	0.757050	0.299447	0.299447
PC2	0.607456	0.192797	0.492244
PC3	0.504619	0.133045	0.625290
PC4	0.398799	0.083096	0.708386
PC5	0.337405	0.059481	0.767866
PC6	0.310275	0.050300	0.818166
PC7	0.289697	0.043849	0.862015
PC8	0.275122	0.039548	0.901563
PC9	0.265251	0.036761	0.938323
PC10	0.248842	0.032353	0.970677
PC11	0.236903	0.029323	1.000000

In [8]:
```
print("Standard Deviation:\n",SD.round(1))

load = (pca.components_)
i=0
```

```
rot_matrix = MD_p.components_.T

rot_df = pd.DataFrame(rot_matrix, index=MD.columns.values, columns=index)
rot_df=round(-rot_df,3)
rot_df
```

```
Standard Deviation:
 [0.8 0.6 0.5 0.4 0.3 0.3 0.3 0.3 0.3 0.2 0.2]
```

Out[8]:

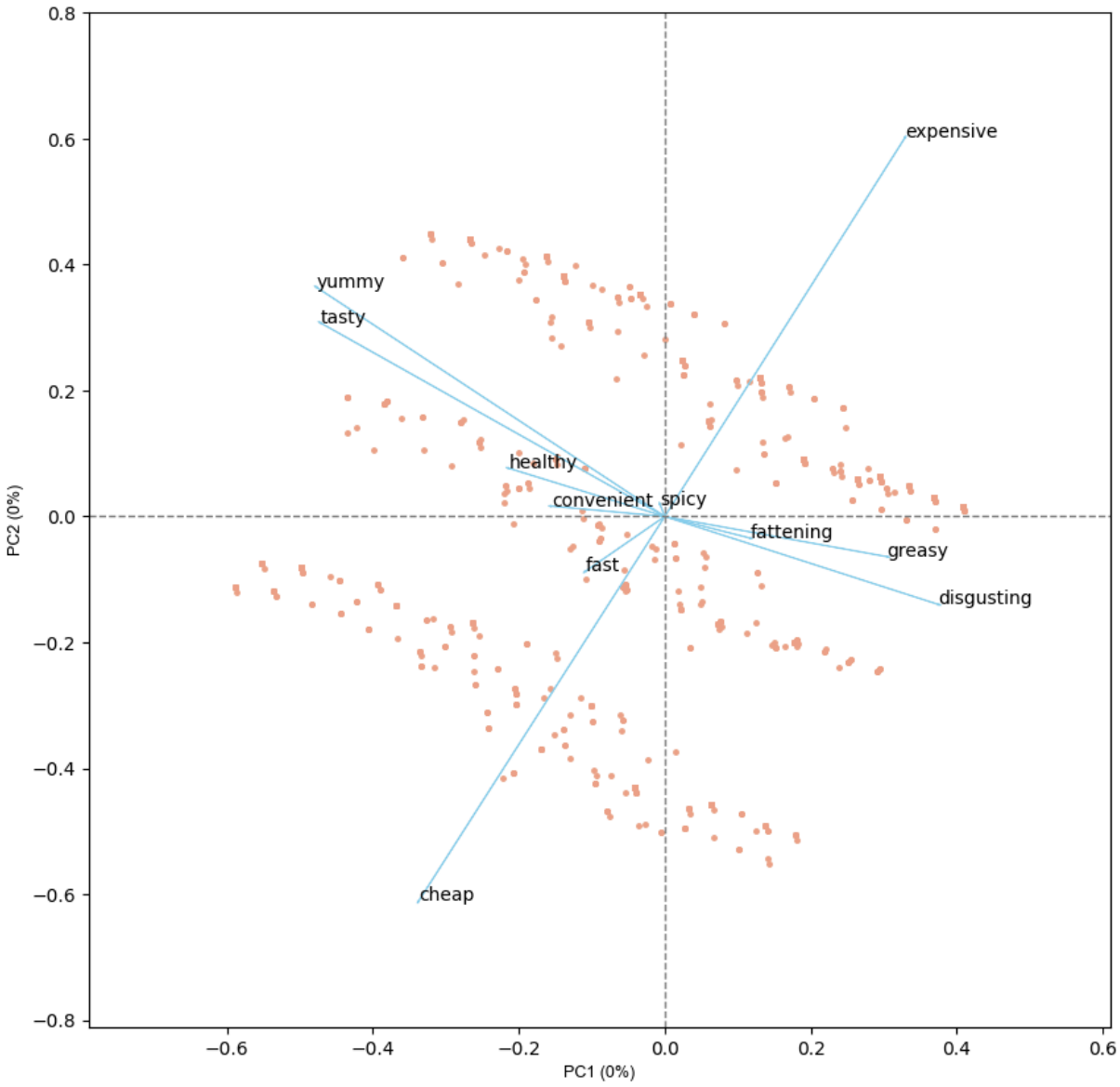
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC
yummy	-0.477	0.364	-0.304	-0.055	-0.308	0.171	0.281	0.013	0.572	-0.110	0.0
convenient	-0.155	0.016	-0.063	0.142	0.278	-0.348	0.060	-0.113	-0.018	-0.666	-0.5
spicy	-0.006	0.019	-0.037	-0.198	0.071	-0.355	-0.708	0.376	0.400	-0.076	0.1
fattening	0.116	-0.034	-0.322	0.354	-0.073	-0.407	0.386	0.590	-0.161	-0.005	0.2
greasy	0.304	-0.064	-0.802	-0.254	0.361	0.209	-0.036	-0.138	-0.003	0.009	0.0
fast	-0.108	-0.087	-0.065	0.097	0.108	-0.595	0.087	-0.628	0.166	0.240	0.3
cheap	-0.337	-0.611	-0.149	-0.119	-0.129	-0.103	0.040	0.140	0.076	0.428	-0.4
tasty	-0.472	0.307	-0.287	0.003	-0.211	-0.077	-0.360	-0.073	-0.639	0.079	0.0
expensive	0.329	0.601	0.024	-0.068	-0.003	-0.261	0.068	0.030	0.067	0.454	-0.4
healthy	-0.214	0.077	0.192	-0.763	0.288	-0.178	0.350	0.176	-0.186	-0.038	0.1
disgusting	0.375	-0.140	-0.089	-0.370	-0.729	-0.211	0.027	-0.167	-0.072	-0.290	-0.0

In [9]:
```
rot_df
```

Out[9]:

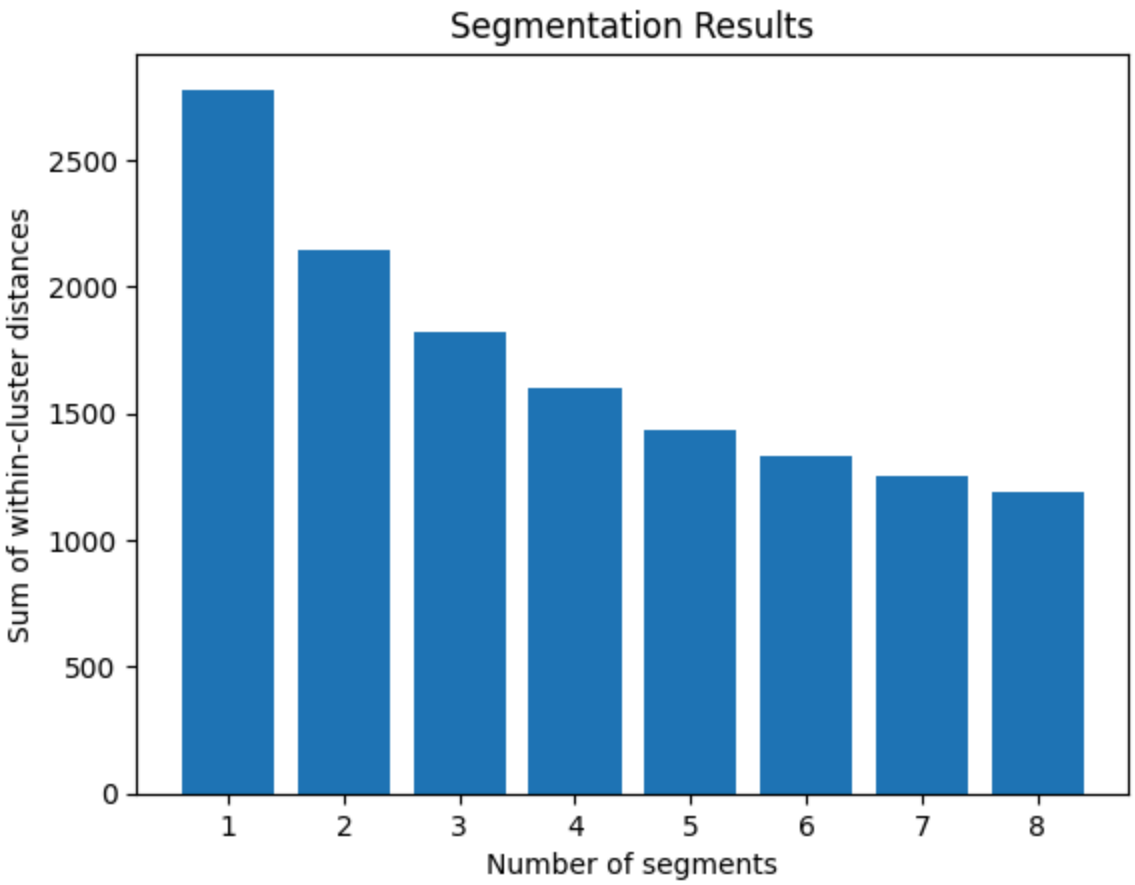
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC
yummy	-0.477	0.364	-0.304	-0.055	-0.308	0.171	0.281	0.013	0.572	-0.110	0.0
convenient	-0.155	0.016	-0.063	0.142	0.278	-0.348	0.060	-0.113	-0.018	-0.666	-0.5
spicy	-0.006	0.019	-0.037	-0.198	0.071	-0.355	-0.708	0.376	0.400	-0.076	0.1
fattening	0.116	-0.034	-0.322	0.354	-0.073	-0.407	0.386	0.590	-0.161	-0.005	0.2
greasy	0.304	-0.064	-0.802	-0.254	0.361	0.209	-0.036	-0.138	-0.003	0.009	0.0
fast	-0.108	-0.087	-0.065	0.097	0.108	-0.595	0.087	-0.628	0.166	0.240	0.3
cheap	-0.337	-0.611	-0.149	-0.119	-0.129	-0.103	0.040	0.140	0.076	0.428	-0.4
tasty	-0.472	0.307	-0.287	0.003	-0.211	-0.077	-0.360	-0.073	-0.639	0.079	0.0
expensive	0.329	0.601	0.024	-0.068	-0.003	-0.261	0.068	0.030	0.067	0.454	-0.4
healthy	-0.214	0.077	0.192	-0.763	0.288	-0.178	0.350	0.176	-0.186	-0.038	0.1
disgusting	0.375	-0.140	-0.089	-0.370	-0.729	-0.211	0.027	-0.167	-0.072	-0.290	-0.0

In [10]:
```
cluster.biplot(cscore=MD_pca, loadings=-load, labels=data.columns.values,var1=0,var
```



In [11]:
```
np.random.seed(1234)

nrep = 10

num_segments = range(1, 9)
within_cluster_distances = []
MD_km28 = {}

for k in num_segments:
    kmeans = KMeans(n_clusters=k, n_init=nrep, random_state=1234)
    kmeans.fit(MD)
    within_cluster_distances.append((kmeans.inertia_))
    MD_km28[str(k)] = kmeans

plt.bar(num_segments, within_cluster_distances)
plt.xlabel("Number of segments")
plt.ylabel("Sum of within-cluster distances")
```

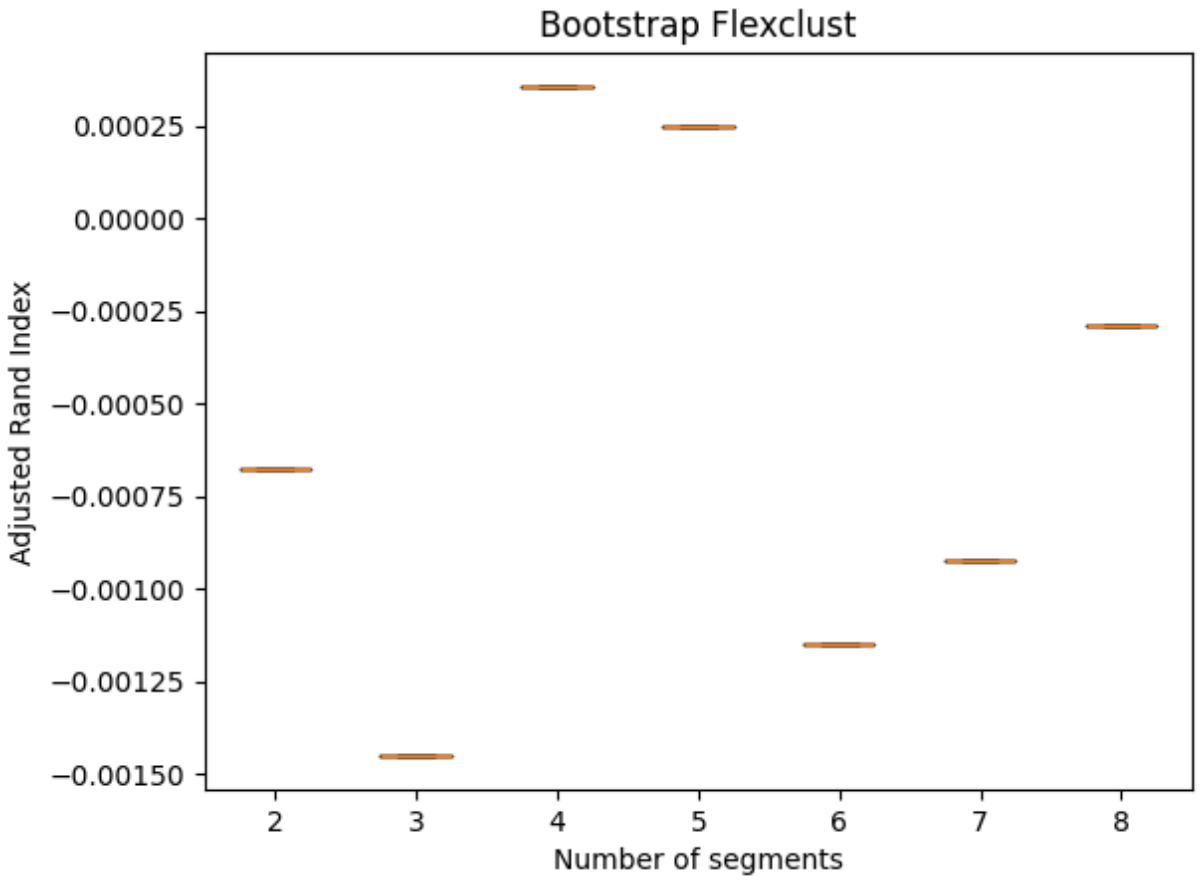
```
plt.title("Segmentation Results")
plt.show()
```



In [12]:
```
np.random.seed(1234)
nboot = 100
nrep = 10

bootstrap_samples = []
for _ in range(nboot):
    bootstrap_sample = resample(MD.values, random_state=1234)
    bootstrap_samples.append(bootstrap_sample)

adjusted_rand_index = []
num_segments = range(2, 9)
for k in num_segments:
    stability_scores = []
    for bootstrap_sample in bootstrap_samples:
        kmeans = KMeans(n_clusters=k, n_init=nrep, random_state=1234)
        kmeans.fit(bootstrap_sample)
        cluster_labels = kmeans.predict(bootstrap_sample)
        true_labels = kmeans.predict(MD.values)
        stability_score = adjusted_rand_score(true_labels, cluster_labels)
        stability_scores.append(stability_score)
    adjusted_rand_index.append(stability_scores)

# Transpose the adjusted_rand_index list
adjusted_rand_index = np.array(adjusted_rand_index).T
```

```
# Create boxplot of adjusted Rand index
plt.boxplot(adjusted_rand_index, labels=num_segments, whis=10)
plt.xlabel("Number of segments")
plt.ylabel("Adjusted Rand Index")
plt.title("Bootstrap Flexclust")
plt.show()
```

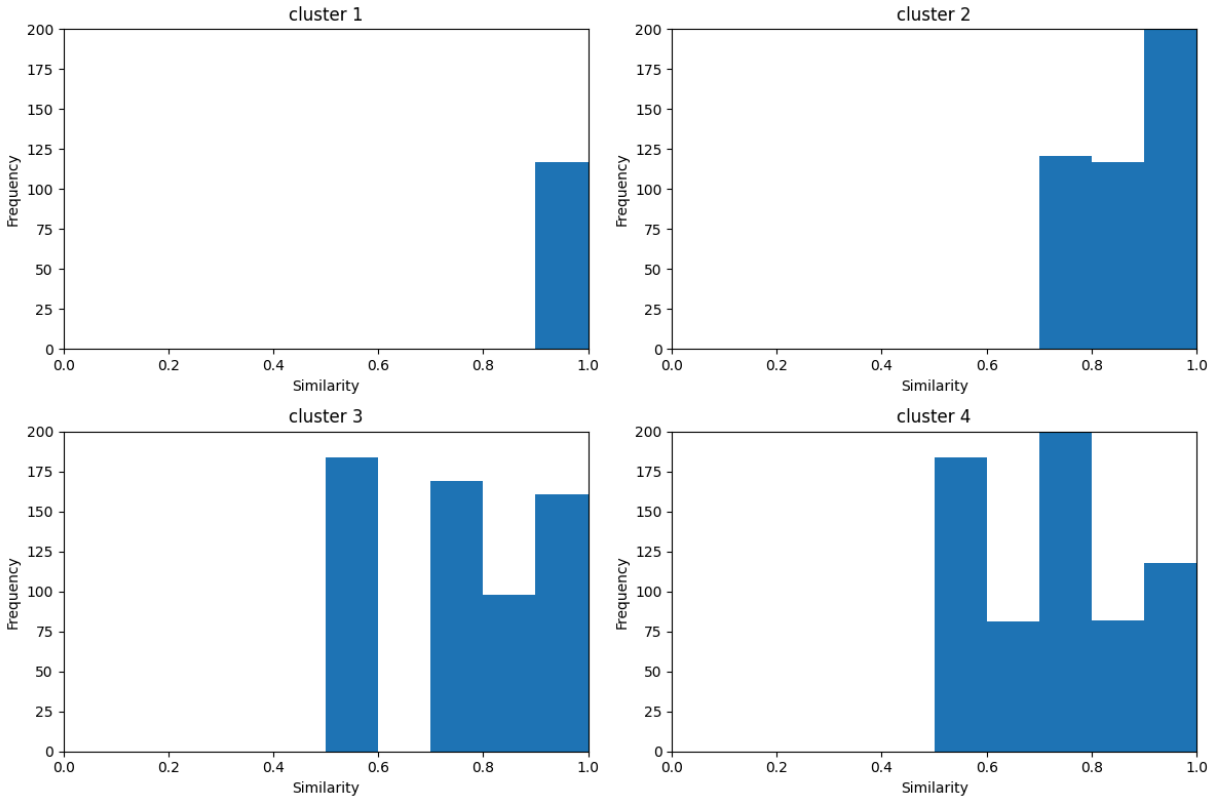
```
C:\Users\an652\AppData\Local\Temp\ipykernel_34444\205992536.py:27: MatplotlibDeprecationWarning: The 'labels' parameter of boxplot() has been renamed 'tick_labels' since Matplotlib 3.9; support for the old name will be dropped in 3.11.
  plt.boxplot(adjusted_rand_index, labels=num_segments, whis=10)
```



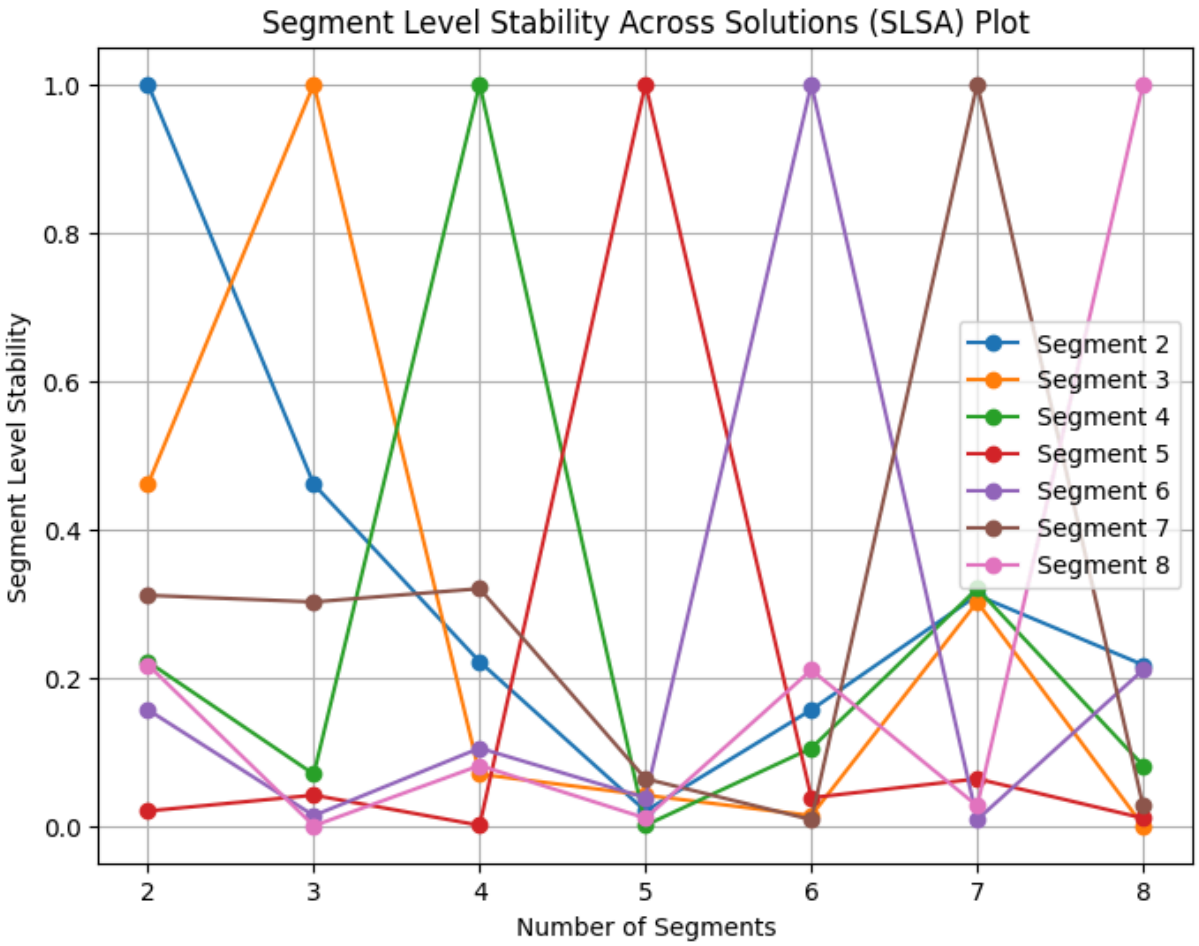
In [13]:

```
range_values = (0, 1)
num_bins = 10
max_frequency = 200


fig, axs = plt.subplots(2, 2, figsize=(12, 8))


for i in range(1, 5):
    labels = MD_km28[str(i)].predict(MD)
    similarities = MD_km28[str(i)].transform(MD).min(axis=1)
    row = (i - 1) // 2
    col = (i - 1) % 2

    axs[row, col].hist(similarities, bins=num_bins, range=range_values)
    axs[row, col].set_xlabel('Similarity')
    axs[row, col].set_ylabel('Frequency')
```

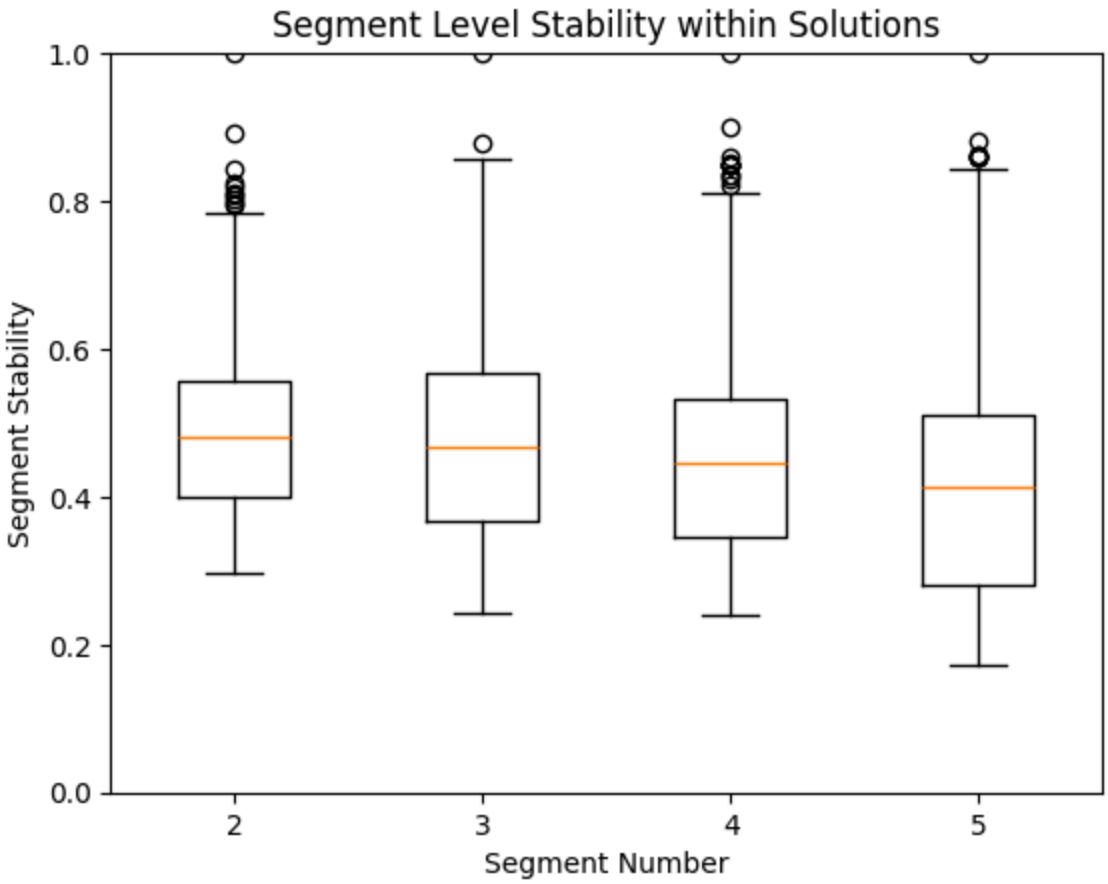
```
    axs[row, col].set_title('cluster {}'.format(i))

    axs[row, col].set_xlim(range_values)
    axs[row, col].set_ylim(0, max_frequency)


    axs[row, col].set_xticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])

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



In [14]:
```
num_segments = range(2, 9)

segment_stability = []
for segment in range(2, 9):
    labels_segment = MD_km28[str(segment)].predict(MD)
    segment_stability.append(labels_segment)

plt.figure(figsize=(8, 6))
for i, segment in enumerate(range(2, 9)):
    plt.plot(num_segments, [np.mean(segment_stability[i] == labels) for labels in s

plt.xlabel('Number of Segments')
plt.ylabel('Segment Level Stability')
plt.title('Segment Level Stability Across Solutions (SLSA) Plot')
plt.xticks(num_segments)
plt.legend()
plt.grid(True)

plt.show()
```



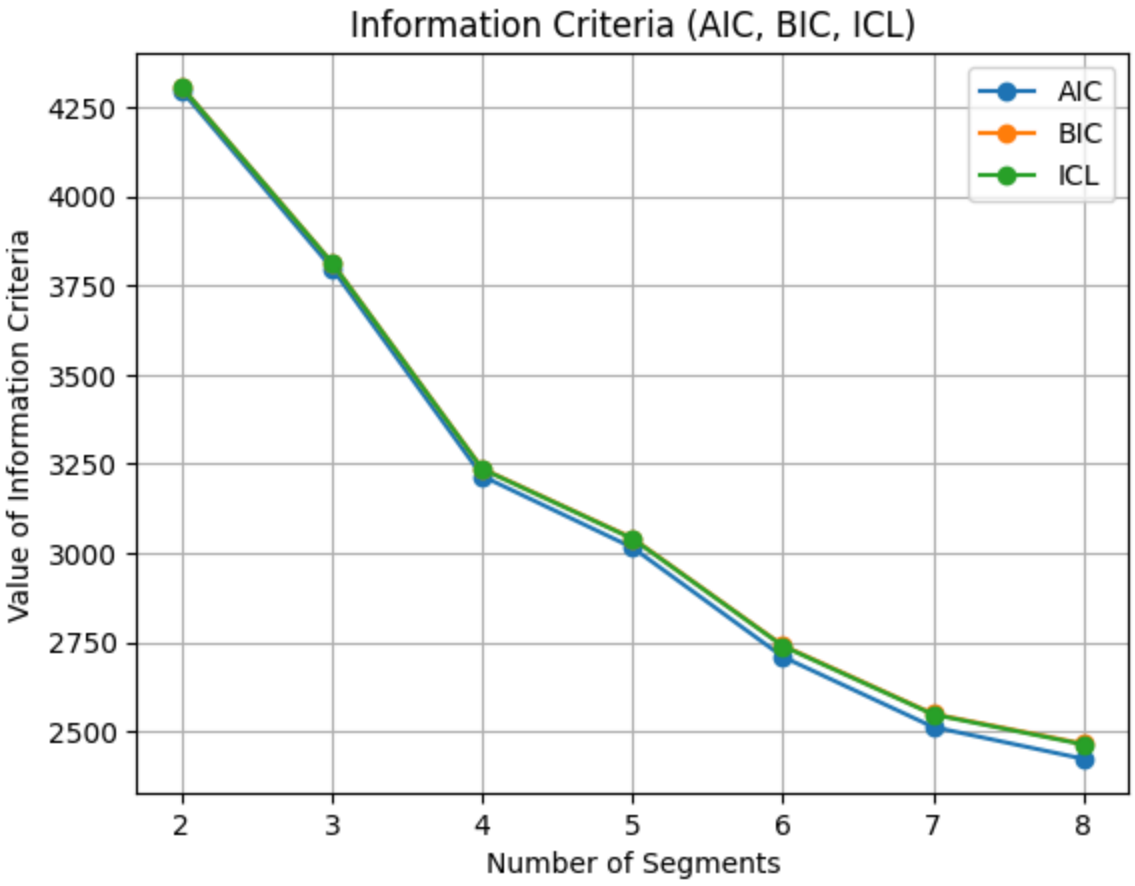
In [15]:
```
segment_solutions = ["2", "3", "4", "5"]
segment_labels = {}
segment_similarities = {}

for segment in segment_solutions:
    segment_labels[segment] = MD_km28[segment].predict(MD)
    segment_similarities[segment] = MD_km28[segment].transform(MD).min(axis=1)

segment_stability_values = []
for segment in segment_solutions:
    similarities = segment_similarities[segment]
    normalized_similarities = similarities / np.max(similarities)
    segment_stability_values.append(normalized_similarities)

plt.boxplot(segment_stability_values, whis=1.5)
plt.xlabel("Segment Number")
plt.ylabel("Segment Stability")
plt.xticks(range(1, len(segment_solutions) + 1), segment_solutions)
plt.ylim(0, 1)
plt.title("Segment Level Stability within Solutions")

plt.show()
```



In [16]:
```
from scipy.stats import entropy

np.random.seed(1234)
k_values = range(2, 9)
MD_m28 = []

for k in k_values:
    model = KMeans(n_clusters=k, random_state=1234)
    model.fit(MD.values)
    iter_val = model.n_iter_
    converged = True
    k_val = k
    k0_val = k
    log_likelihood = -model.inertia_
    n_samples, _ = MD.shape
    aic = -2 * log_likelihood + 2 * k
    bic = -2 * log_likelihood + np.log(n_samples) * k
    labels = model.labels_
    counts = np.bincount(labels)
    probs = counts / float(counts.sum())
    class_entropy = entropy(probs)
    icl = bic - class_entropy

    MD_m28.append((iter_val, converged, k_val, k0_val, log_likelihood, aic, bic, ic
MD_m28 = pd.DataFrame(MD_m28, columns=['iter', 'converged', 'k', 'k0', 'logLik', 'A

print(MD_m28)
```

	iter	converged	k	k0	logLik	AIC	BIC	ICL
0	10	True	2	2	-2146.062044	4296.124088	4306.686859	4306.015908
1	5	True	3	3	-1896.330266	3798.660532	3814.504689	3813.529671
2	9	True	4	4	-1603.913802	3215.827604	3236.953147	3235.627738
3	9	True	5	5	-1502.697153	3015.394306	3041.801234	3040.267284
4	7	True	6	6	-1348.665399	2709.330799	2741.019113	2739.277954
5	10	True	7	7	-1249.233890	2512.467780	2549.437480	2547.530062
6	9	True	8	8	-1203.646165	2423.292330	2465.543415	2463.533662

In [17]:
```
num_segments = MD_m28["k"]
AIC_values = MD_m28["AIC"]
BIC_values = MD_m28["BIC"]
ICL_values = MD_m28["ICL"]

plt.plot(num_segments, AIC_values, marker='o', label='AIC')
plt.plot(num_segments, BIC_values, marker='o', label='BIC')
plt.plot(num_segments, ICL_values, marker='o', label='ICL')

plt.xlabel('Number of Segments')
plt.ylabel('Value of Information Criteria')
plt.title('Information Criteria (AIC, BIC, ICL)')
plt.legend()
plt.grid(True)

plt.show()
```



In [18]:
```
from sklearn.mixture import GaussianMixture
k = 4
```

```
kmeans = KMeans(n_clusters=k, random_state=1234)
kmeans.fit(MD)
kmeans_clusters = kmeans.predict(MD)

gmm = GaussianMixture(n_components=k, random_state=1234)
gmm.fit(MD)
gmm_clusters = gmm.predict(MD)

results = pd.DataFrame({'kmeans': kmeans_clusters, 'mixture': gmm_clusters})

MD_m4 = MD[results['mixture'] == 3]

k4_m4 = KMeans(n_clusters=k, random_state=1234)
k4_m4.fit(MD_m4)
k4_m4_clusters = k4_m4.predict(MD_m4)

results_m4 = pd.DataFrame({'kmeans': k4_m4_clusters, 'mixture': 3})

print(pd.crosstab(results['kmeans'], results['mixture']))
print(pd.crosstab(results['kmeans'], results_m4['kmeans']))
```

```
mixture    0    1    2    3
kmeans
0        546    0    1   33
1          0  213   11    4
2         46    3  265    8
3         29   38    0  256
kmeans   0   1   2   3
kmeans
0       51  15  35  20
1       18   8  11  11
2       23   9  14  22
3       29   5  15  15
```

In [19]:

```
from sklearn.mixture import GaussianMixture
import numpy as np

gmm_m4a = GaussianMixture(n_components=4)
gmm_m4a.fit(MD)

log_likelihood_m4a = gmm_m4a.score(MD)

gmm_m4 = GaussianMixture(n_components=4)
gmm_m4.fit(MD)

log_likelihood_m4 = gmm_m4.score(MD)

print("Log-likelihood for MD.m4a:", log_likelihood_m4a)
print("Log-likelihood for MD.m4:", log_likelihood_m4)
```

```
Log-likelihood for MD.m4a: 9.456781748071887
Log-likelihood for MD.m4: 3.642746401176581
```

In [20]:
```
like_counts = pd.value_counts(data['Like'])
reversed_counts = like_counts.iloc[::-1]

print(reversed_counts)
```

```
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
```

```
C:\Users\an652\AppData\Local\Temp\ipykernel_34444\1383621504.py:1: FutureWarning: pa
ndas.value_counts is deprecated and will be removed in a future version. Use pd.Seri
es(obj).value_counts() instead.
  like_counts = pd.value_counts(data['Like'])
```

In [21]:
```
# Define a mapping of string values to numeric codes
like_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 LOVE IT!+5': 5
}

data['Like.n'] = data['Like'].map(like_mapping)


like_n_counts = data['Like.n'].value_counts()


print(like_n_counts)
```

```
Like.n
 3.0    229
 2.0    187
 0.0    169
 4.0    160
 1.0    152
-3.0     73
-4.0     71
-2.0     59
-1.0     58
Name: count, dtype: int64
```

In [22]:
```python
from patsy import dmatrices

independent_vars = data.columns[0:11]

formula_str = ' + '.join(independent_vars)

formula_str = 'Like ~ ' + formula_str


f = dmatrices(formula_str, data=data)[1]

print(f)
```

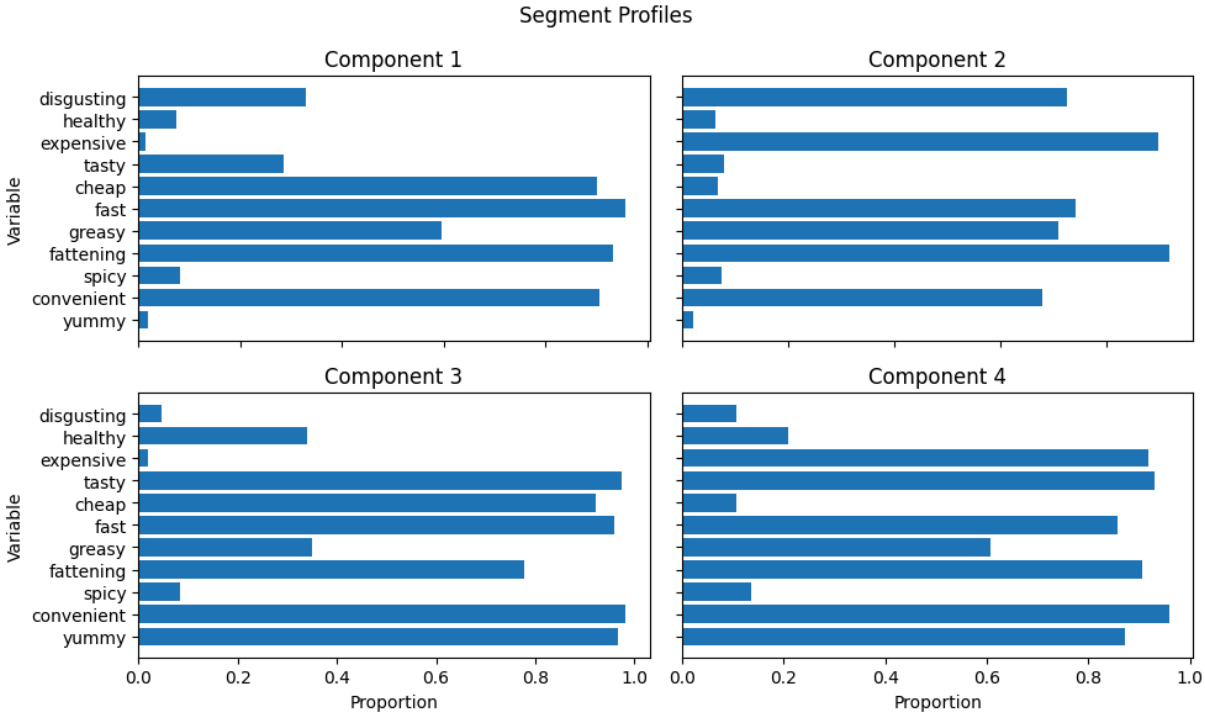
```
[[1. 0. 1. ... 1. 0. 0.]
 [1. 1. 1. ... 1. 0. 0.]
 [1. 0. 1. ... 1. 1. 0.]
 ...
 [1. 1. 1. ... 1. 0. 0.]
 [1. 1. 1. ... 0. 1. 0.]
 [1. 0. 1. ... 1. 0. 1.]]
```

In [23]:
```python
from sklearn.mixture import GaussianMixture
from patsy import dmatrix
np.random.seed(1234)

X = dmatrix(f.design_info, data=data)
y = dmatrix('Like', data=data)

n_components = 2
n_init = 10
verbose = False
n_rep=10

model = GaussianMixture(n_components=n_components, n_init=n_init, verbose=verbose)
MD_reg2 = model.fit(X, y)

print(MD_reg2)
cluster_sizes = np.bincount(model.predict(X))

print("Cluster sizes:")
for i, size in enumerate(cluster_sizes):
    print(f"{i+1}: {size}")
```

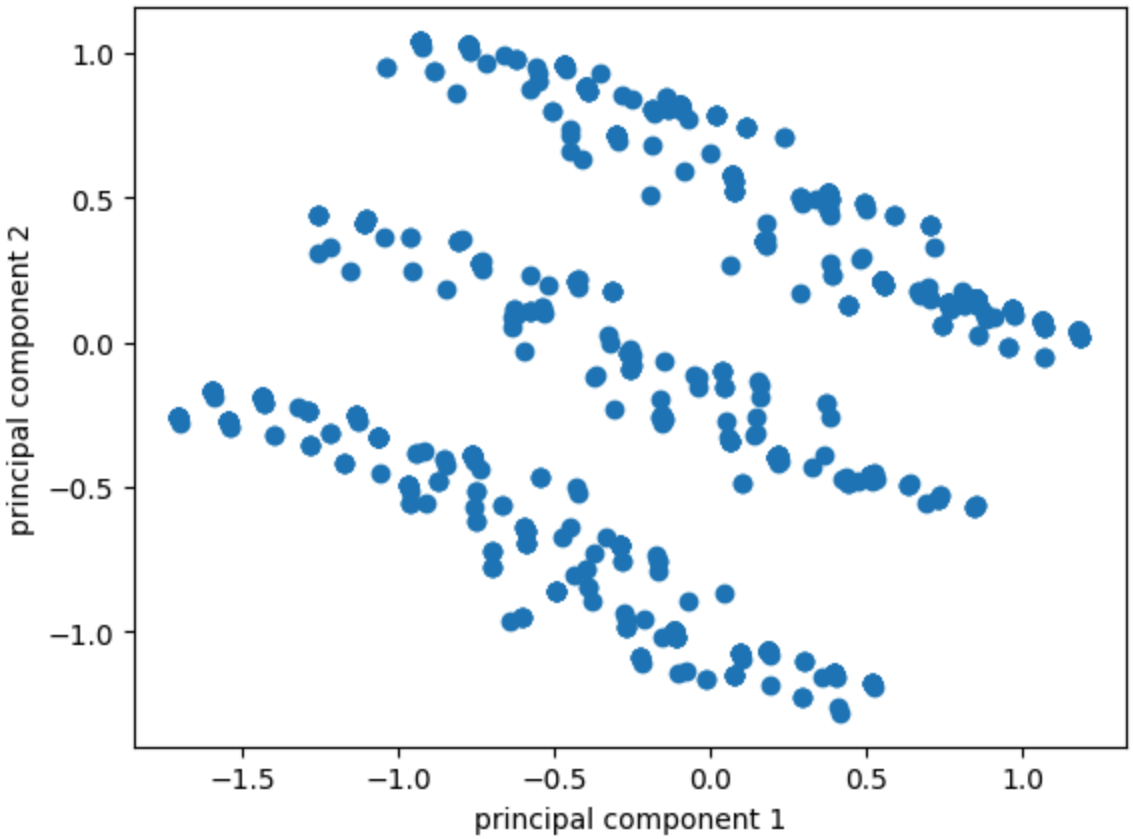
```
GaussianMixture(n_components=2, n_init=10, verbose=False)
Cluster sizes:
1: 985
2: 468
```

In [24]:

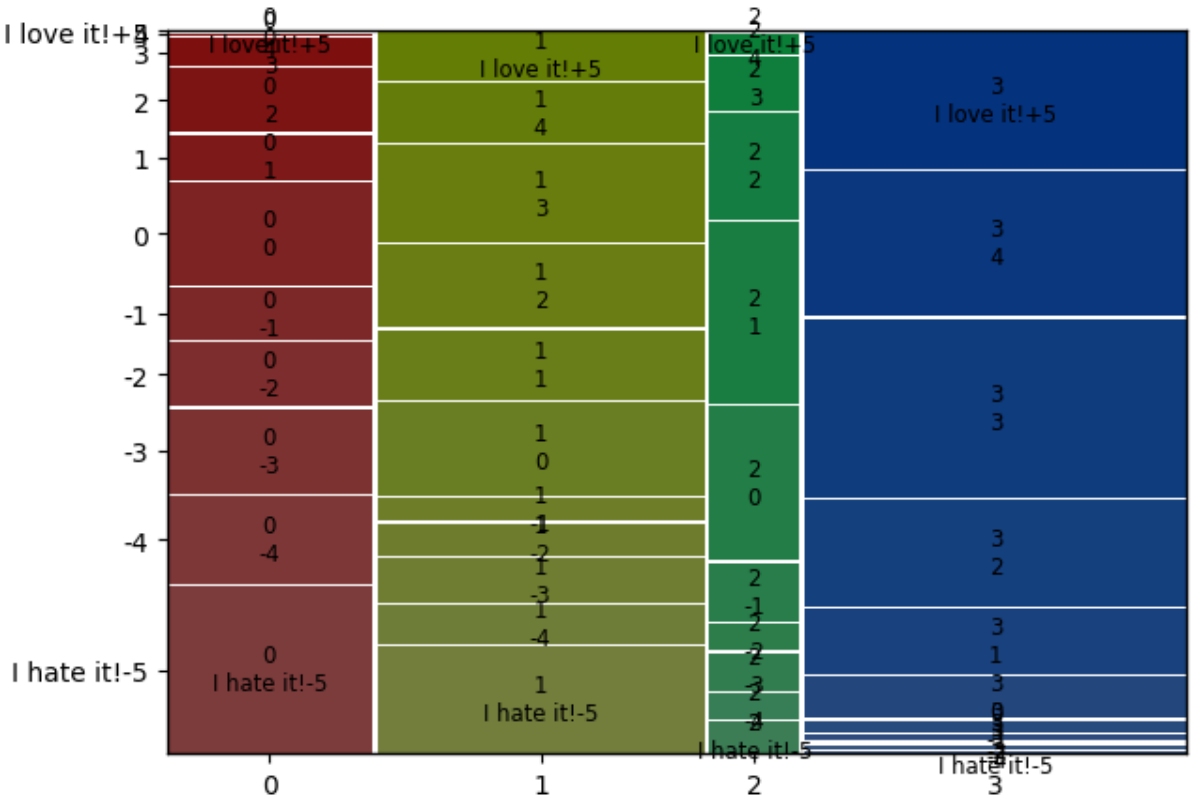
```python
import pandas as pd
import matplotlib.pyplot as plt

kmeans = MD_km28['4']

labels = kmeans.labels_

MD_mean = MD.groupby(labels).mean()

fig, axs = plt.subplots(2, 2, figsize=(10, 6))
axs[0, 0].barh(range(MD_mean.shape[1]), MD_mean.iloc[0])
axs[0, 0].set_title('Component 1')
axs[0, 1].barh(range(MD_mean.shape[1]), MD_mean.iloc[1])
axs[0, 1].set_title('Component 2')
axs[1, 0].barh(range(MD_mean.shape[1]), MD_mean.iloc[2])
axs[1, 0].set_title('Component 3')
axs[1, 1].barh(range(MD_mean.shape[1]), MD_mean.iloc[3])
axs[1, 1].set_title('Component 4')

for ax in axs.flat:
    ax.set(ylabel='Variable', xlabel='Proportion')
    ax.set_yticks(range(MD_mean.shape[1]))
    ax.set_yticklabels(MD.columns)

for ax in axs.flat:
    ax.label_outer()

fig.suptitle('Segment Profiles')

fig.tight_layout()

plt.show()
```



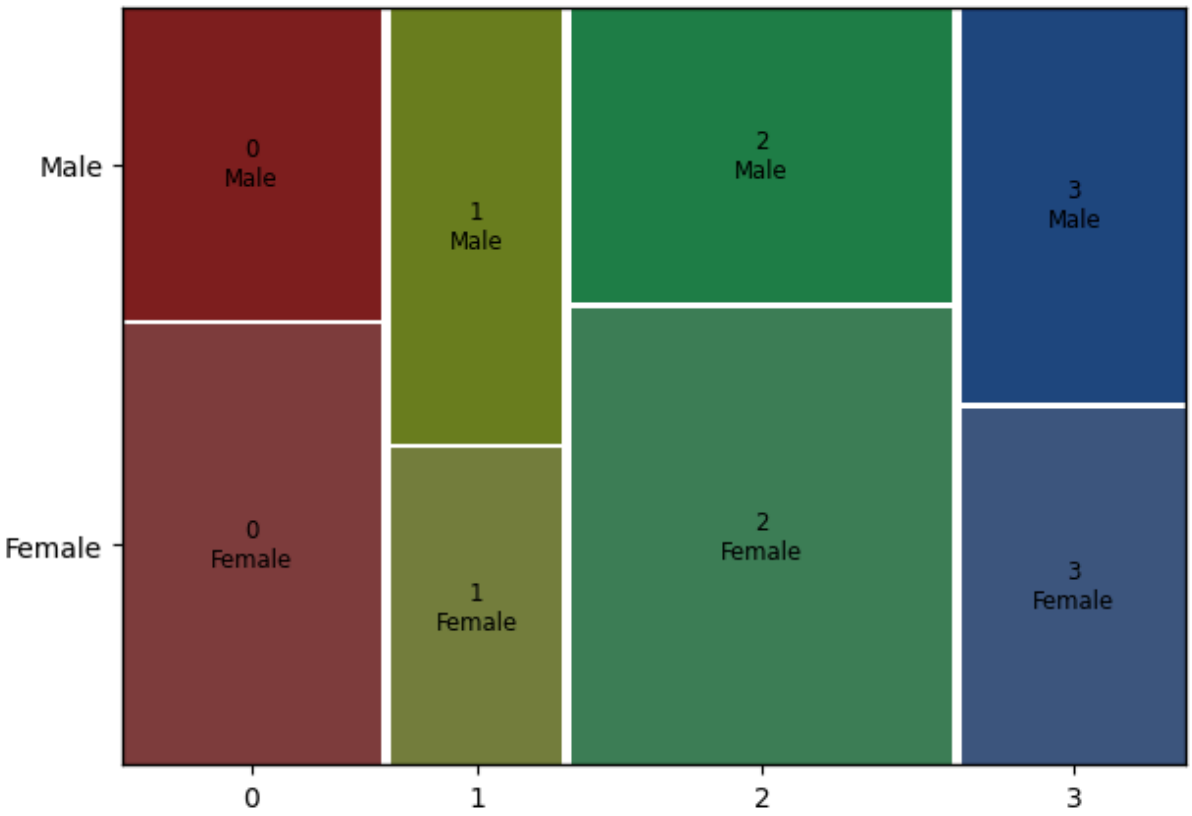
In [25]:
```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

kmeans = KMeans(n_clusters=4)
kmeans.fit(MD)

pca = PCA(n_components=2)
MD_pca = pca.fit_transform(MD)

fig, ax = plt.subplots()

ax.scatter(MD_pca[:, 0], MD_pca[:, 1])
ax.set_xlabel('principal component 1')
ax.set_ylabel('principal component 2')
plt.show()
```



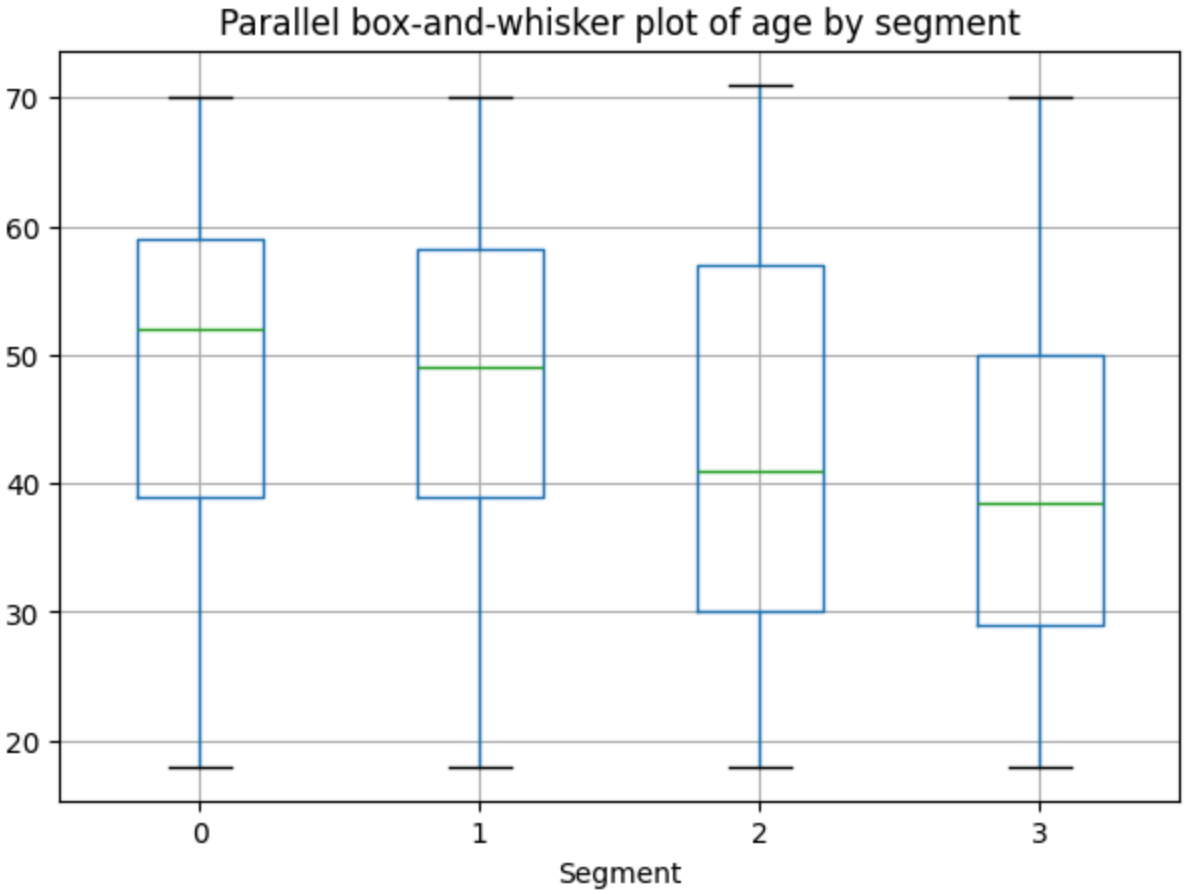
In [26]:
```
from statsmodels.graphics.mosaicplot import mosaic
from itertools import product
#Label encoding for categorical - Converting 11 cols with yes/no

from sklearn.preprocessing import LabelEncoder
def labelling(x):
    data1[x] = LabelEncoder().fit_transform(data1[x])
    return data1

cat = ['yummy', 'convenient', 'spicy', 'fattening', 'greasy', 'fast', 'cheap',
       'tasty', 'expensive', 'healthy', 'disgusting']

for i in cat:
    labelling(i)
data1
df_eleven = data1.loc[:,cat]
df_eleven
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0).fit(df_eleven)
data1['cluster_num'] = kmeans.labels_
crosstab =pd.crosstab(data1['cluster_num'],data1['Like'])
#Reordering cols
data1
crosstab = crosstab[['I hate it!-5','-4','-3','-2','-1','0','1','2','3','4','I love
crosstab
plt.rcParams['figure.figsize'] = (7,5)
mosaic(crosstab.stack())
plt.show()
```



In [27]:
```
from statsmodels.graphics.mosaicplot import mosaic

MD_k4=MD_km28['4']
k4 = MD_k4.labels_

ct = pd.crosstab(k4, data['Gender'])
ct
mosaic(ct.stack(),gap=0.01)

plt.show()
```



In [28]:
```
df = pd.DataFrame({'Segment': k4, 'Age': data['Age']})

df.boxplot(by='Segment', column='Age')
plt.title('Parallel box-and-whisker plot of age by segment')
plt.suptitle('')
plt.show()
```



In [29]:
```
data1['VisitFrequency'] = LabelEncoder().fit_transform(data1['VisitFrequency'])
visit = data1.groupby('cluster_num')['VisitFrequency'].mean()
visit = visit.to_frame().reset_index()
visit
```

Out[29]:

	cluster_num	VisitFrequency
0	0	2.787162
1	1	2.596195
2	2	2.618321
3	3	2.598553

In [30]:
```
#Like
data1['Like'] = LabelEncoder().fit_transform(data1['Like'])
Like = data1.groupby('cluster_num')['Like'].mean()
Like = Like.to_frame().reset_index()
Like
```

Out[30]:

	cluster_num	Like
0	0	4.641892
1	1	5.839323
2	2	4.549618
3	3	7.097649

In [31]:
```
data1['Gender'] = LabelEncoder().fit_transform(data1['Gender'])
Gender = data1.groupby('cluster_num')['Gender'].mean()
Gender = Gender.to_frame().reset_index()
Gender
```

Out[31]:

	cluster_num	Gender
0	0	0.432432
1	1	0.547569
2	2	0.427481
3	3	0.401447

In [32]:
```
import seaborn as sns
segment = Gender.merge(Like, on='cluster_num', how='left').merge(visit, on='cluster
segment
plt.figure(figsize = (9,4))
sns.scatterplot(x = "VisitFrequency", y = "Like",data=segment,s=400, color="r")
plt.title("Simple segment evaluation plot for the fast food data set",fontsize = 15
plt.xlabel("Visit", fontsize = 12)
plt.ylabel("Like", fontsize = 12)
plt.show()
```