In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.mixture import GaussianMixture
from sklearn.metrics import confusion_matrix
from sklearn.metrics import silhouette_score
from sklearn.metrics import adjusted_rand_score
from sklearn.preprocessing import StandardScaler
from statsmodels.graphics.mosaicplot import mosaic
```

In [2]:

```
data = pd.read_csv(r"D:\market segmentation\McDonalds Case Study-20231208T041132Z-001\McDonalds Case Study\mcdo
data.columns
```

Out[2]:

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

In [3]:

```
data.shape
```

Out[3]:

```
(1453, 15)
```

In [4]:

```
data = data.replace({"Yes": 1, "No": 0})
data
```

Out[4]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting	Like	Age	VisitFrequency
0	0	1	0	1	0	1	1	0	1	0	0	-3	61	Every three months
1	1	1	0	1	1	1	1	1	1	0	0	+2	51	Every three months
2	0	1	1	1	1	1	0	1	1	1	0	+1	62	Every three months
3	1	1	0	1	1	1	1	1	0	0	1	+4	69	Once a week
4	0	1	0	1	1	1	1	0	0	1	0	+2	49	Once a month
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1448	0	1	0	1	1	0	0	0	1	0	1	I hate it!-5	47	Once a year
1449	1	1	0	1	0	0	1	1	0	1	0	+2	36	Once a week
1450	1	1	0	1	0	1	0	1	1	0	0	+3	52	Once a month
1451	1	1	0	0	0	1	1	1	0	1	0	+4	41	Every three months
1452	0	1	0	1	1	0	0	0	1	0	1	-3	30	Every three months

1453 rows × 15 columns

In [5]:

```
MD_x=data.iloc[:,:11]
temp=MD_x.mean().round(2)
temp
```

Out[5]:

```
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 [6]:

```
scaler = StandardScaler()
MD_p = scaler.fit_transform(MD_x)

pca = PCA()
MD_p = pca.fit_transform(MD_p)
```

```
pca_df= pd.DataFrame(MD_p,columns=MD_x.columns)
pca_df
```

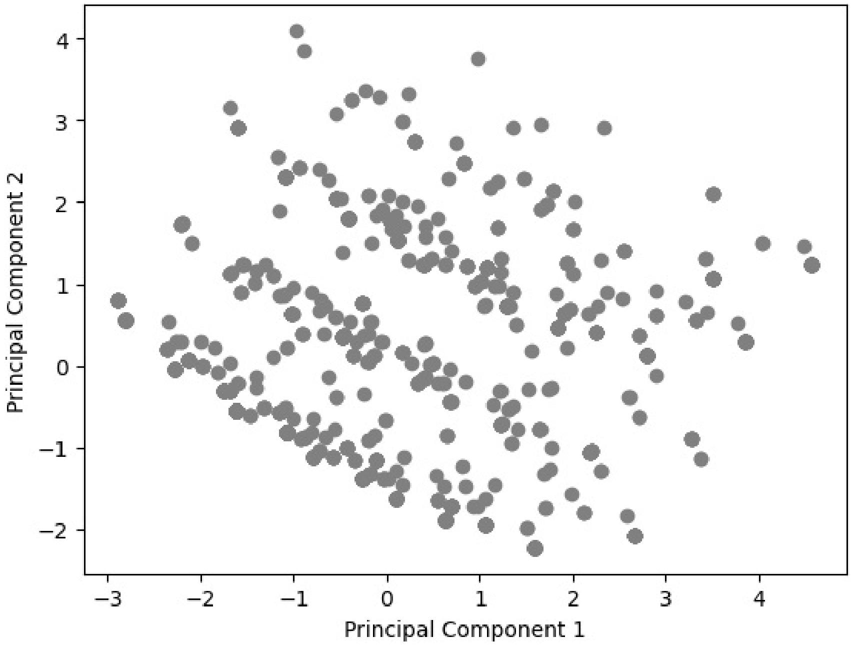
Out[6]:

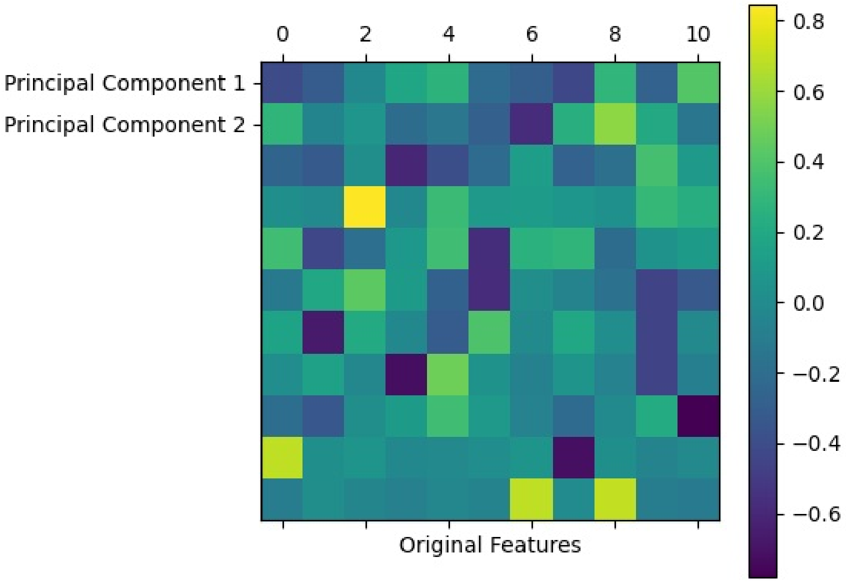
	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting
0	0.704334	-0.437016	0.268698	-0.872074	-1.521184	0.470160	-0.030969	-0.687116	0.367598	0.321161	1.701170
1	-0.467820	0.364277	-1.596835	-0.004835	0.462385	-0.449321	0.087351	0.446003	0.221855	0.191268	1.467681
2	0.191986	1.712949	-0.339413	3.368168	-1.266802	0.148058	-0.606634	-0.668576	1.377226	-1.259300	-0.128530
3	-0.116991	-1.155122	-1.003913	0.469589	1.141750	-0.857182	0.015843	0.390275	-1.578539	0.092189	-0.233201
4	-0.034724	-1.390267	0.792275	0.473031	-0.270488	-0.847963	-1.804085	-0.700019	1.630339	0.092449	-0.033144
...	...	...	...	...	...	...	...	...	...	...	...
1448	3.499105	1.069999	0.148971	-0.195977	0.787923	1.016536	-1.977414	0.049487	-0.964269	0.070579	0.155016
1449	-1.568786	0.899328	1.192503	-0.286497	2.234500	1.258306	-1.764159	-1.705500	-0.232987	-0.036497	0.039392
1450	-0.414275	1.810438	-1.071948	-0.901031	-0.750299	0.065975	0.720962	-0.397984	-0.344847	0.098558	0.118205
1451	-2.803630	0.562759	2.278887	0.083924	0.080147	-0.969368	-0.384558	0.604123	-0.211434	0.083127	0.077614
1452	3.499105	1.069999	0.148971	-0.195977	0.787923	1.016536	-1.977414	0.049487	-0.964269	0.070579	0.155016

1453 rows × 11 columns

In [18]:

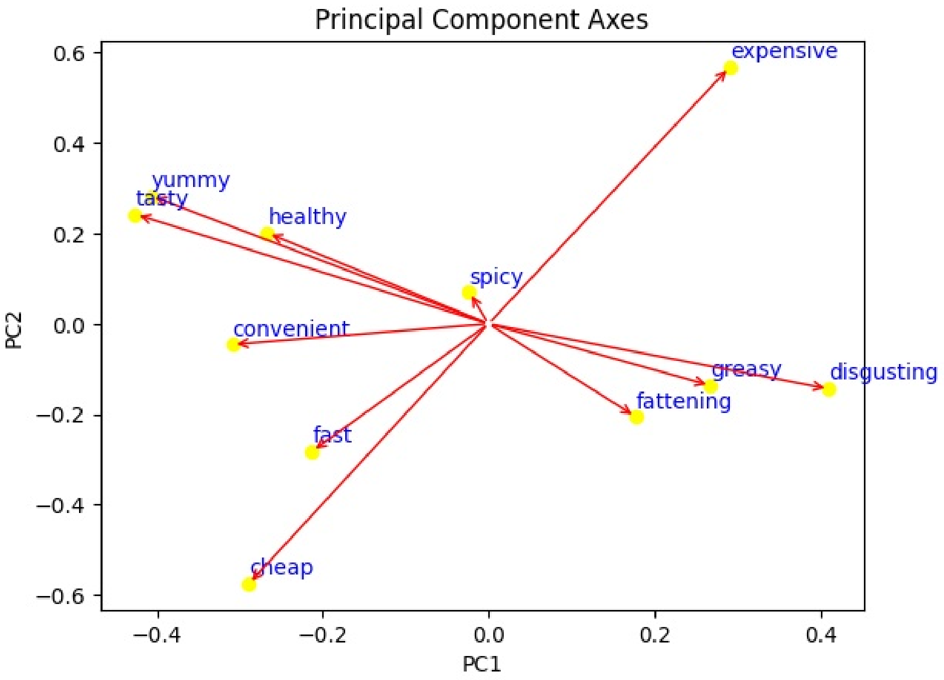
```
# Plot the PCA results
plt.scatter(MD_p[:, 0], MD_p[:, 1], c='grey')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()

# Plot the projection of the original features onto the first two principal components
plt.matshow(pca.components_, cmap='viridis')
plt.yticks([0, 1], ['Principal Component 1', 'Principal Component 2'])
plt.colorbar()
plt.xlabel('Original Features')
plt.show()
```





In [19]:

```
scale = 1

for i, j in enumerate(pca_df.columns):
    plt.text(
        pca.components_[0, :][i] * scale,
        (pca.components_[1, :][i] + 0.02) * scale,
        j, color='blue'
    )

    plt.annotate(
        '', xy=(pca.components_[0, :][i] * scale, pca.components_[1, :][i] * scale),
        xytext=(0, 0),
        arrowprops=dict(
            arrowstyle="->",
            color="red"
        )
    )

plt.scatter(
    pca.components_[0, :] * scale,
    pca.components_[1, :] * scale,
    color="yellow"
)

plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Principal Component Axes')
plt.show()
```



In [20]:

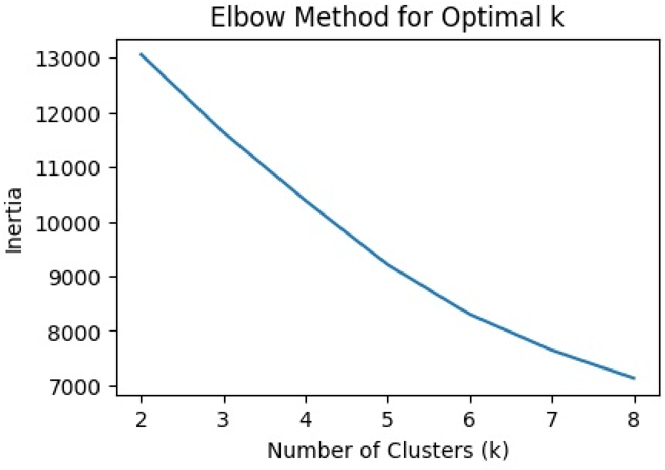
```python
np.random.seed(0)
```

In [21]:

```python
range1 = range(2, 9)
var1 = []
result_range1 = []
scores_s = []

for k in range1:
    model = KMeans(n_clusters=k, n_init=10)
    model.fit(pca_df)
    labels = model.predict(pca_df)
    silhouette_score_ = silhouette_score(pca_df, labels)
    scores_s.append(silhouette_score_)
    var1.append(model.inertia_)
    result_range1.append(labels)
plt.figure(figsize=(10, 3))
plt.subplot(1, 2, 1)
plt.plot(range1, var1)
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')
```

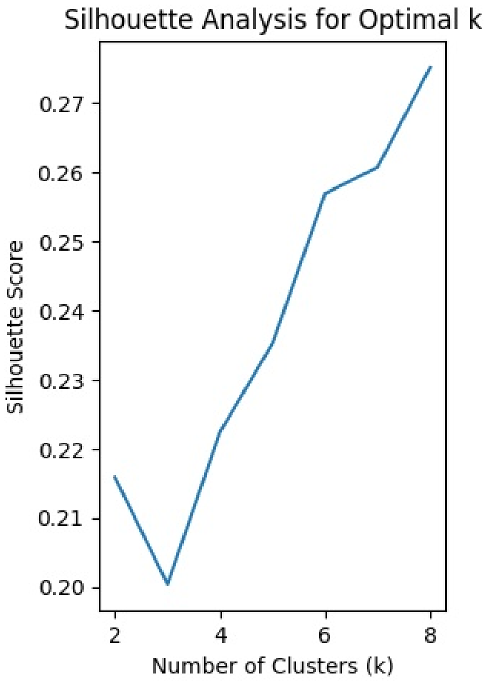
Out[21]:

```
Text(0.5, 1.0, 'Elbow Method for Optimal k')
```



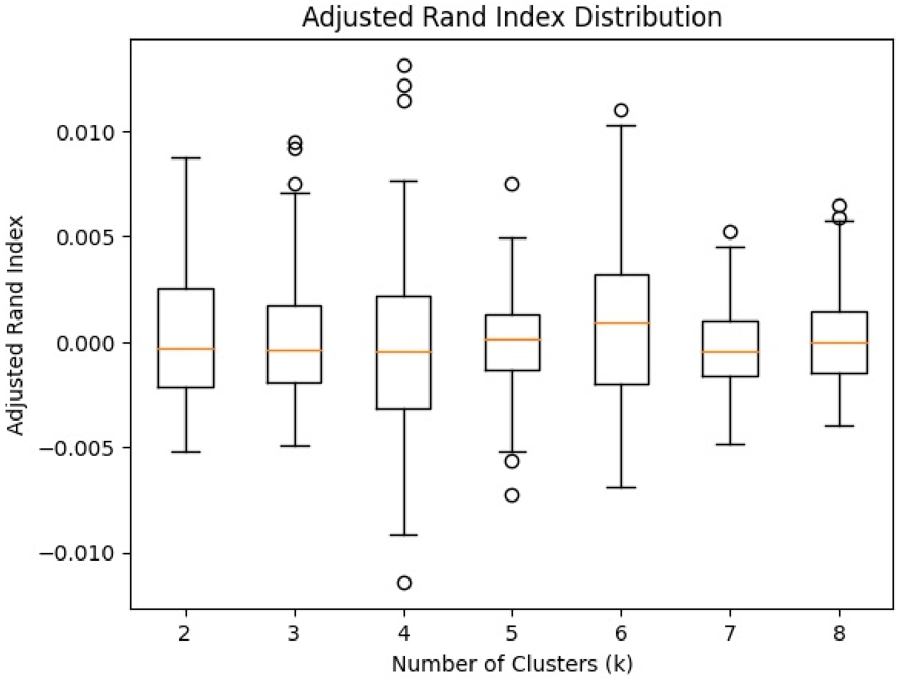
In [22]:

```python
plt.subplot(1, 2, 2)
plt.plot(range1, scores_s)
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Analysis for Optimal k')

plt.show()
```



In [23]:

```
num_boot = 100
adjusted_ri_score = []

for labels in result_range1:
    ari_boot = []
    for i in range(num_boot):
        random_s = np.random.choice(labels, size=len(labels), replace=True)
        ari = adjusted_rand_score(labels, random_s)
        ari_boot.append(ari)
    adjusted_ri_score.append(ari_boot)

plt.boxplot(adjusted_ri_score, labels=range(2, 9))
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Adjusted Rand Index')
plt.title('Adjusted Rand Index Distribution')
plt.show()
```



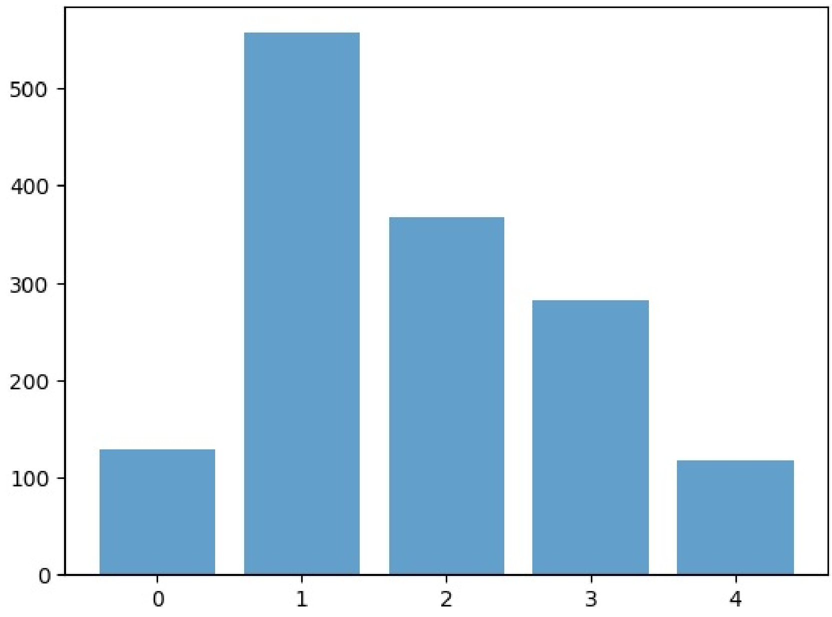
In [24]:

```
gaussian_mixture = GaussianMixture(n_components=4, covariance_type='full', random_state=1234)
gaussian_mixture.fit(pca_df)
clusters = gaussian_mixture.predict(pca_df)

model = KMeans(n_clusters=4, n_init=10, random_state=0)
model.fit(pca_df)
labels = model.predict(pca_df)

cm = confusion_matrix(labels, clusters)
print("Confusion Matrix:\n", cm)
```

```
kmeans = KMeans(n_clusters=5)
kmeans.fit(pca_df)
km = kmeans.predict(pca_df)

x = range(5)
height = [sum(km == i) for i in range(5)]
plt.bar(x, height, alpha=0.7)
plt.show()
```

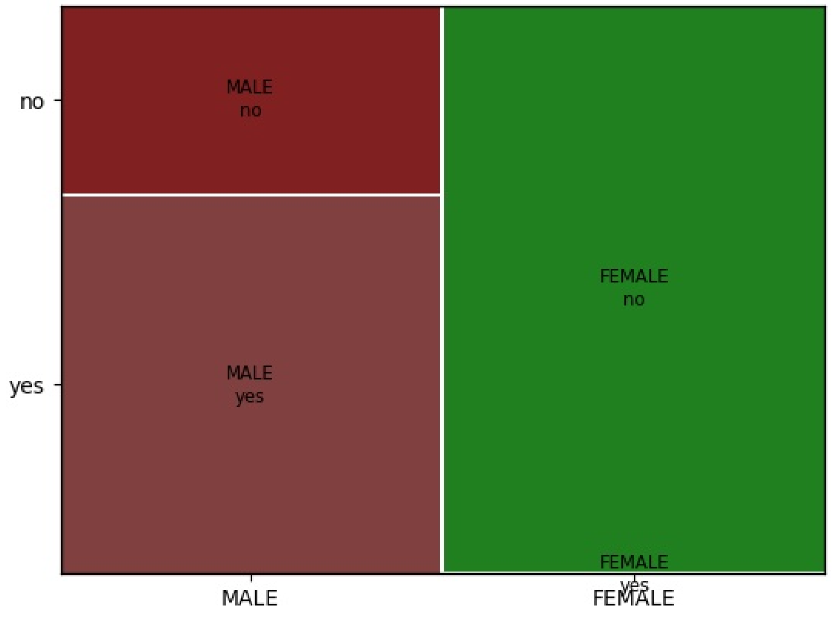
```
Confusion Matrix:
 [[116   4   0  96]
 [  7 163  97   3]
 [  6  71 268 269]
 [  5  55 222  71]]
```

```
C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
```



In [25]:

```
data={
    'Gender':['MALE','FEMALE','MALE','FEMALE','MALE','FEMALE'],
    'Preference':['yes','no','yes','no','no','no']

}
df=pd.DataFrame(data)
mosaic(df,['Gender','Preference'])
plt.show()
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js