Radhika

December 20, 2024

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
file_path = 'mcdonalds (1).csv'
mcdonalds = pd.read_csv(file_path)
print(mcdonalds.columns)
```

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

[3]:
```python
print(mcdonalds.shape)
```

```
(1453, 15)
```

[5]:
```python
mcdonalds.head(4)
```

[5]:
```
  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
3   Yes        Yes    No       Yes    Yes  Yes   Yes   Yes        No      No

  disgusting Like  Age      VisitFrequency  Gender
0         No   -3   61  Every three months  Female
1         No   +2   51  Every three months  Female
2         No   +1   62  Every three months  Female
3        Yes   +4   69         Once a week  Female
```

[6]:
```python
print(mcdonalds. info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1453 entries, 0 to 1452
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   yummy           1453 non-null   object
 1   convenient      1453 non-null   object
 2   spicy           1453 non-null   object
 3   fattening       1453 non-null   object
 4   greasy          1453 non-null   object
 5   fast            1453 non-null   object
 6   cheap           1453 non-null   object
 7   tasty           1453 non-null   object
 8   expensive       1453 non-null   object
 9   healthy         1453 non-null   object
 10  disgusting      1453 non-null   object
 11  Like            1453 non-null   object
 12  Age             1453 non-null   int64
 13  VisitFrequency  1453 non-null   object
 14  Gender          1453 non-null   object
dtypes: int64(1), object(14)
memory usage: 170.4+ KB
None
```

[7]:

```python
MD_x = mcdonalds.iloc[:, :11].applymap(lambda x: 1 if x == "Yes" else 0)  # Convert "Yes" to 1, others to 0
col_means = MD_x.mean(axis=0).round(2)  # Compute column means and round to 2 decimal places

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
```

```
/tmp/ipykernel_1315/3819941208.py:1: FutureWarning: DataFrame.applymap has been
deprecated. Use DataFrame.map instead.
  MD_x = mcdonalds.iloc[:, :11].applymap(lambda x: 1 if x == "Yes" else 0)  #
Convert "Yes" to 1, others to 0
```

[8]:

```python
print(mcdonalds.describe())
```

```
               Age
count  1453.000000
mean     44.604955
```

```
std      14.221178
min      18.000000
25%      33.000000
50%      45.000000
75%      57.000000
max      71.000000
```

[9]:
```
MD_x = mcdonalds.iloc[:, :12].values
MD_x = (MD_x == "Yes").astype(int)
column_means = np.round(MD_x.mean(axis=0),2)
print(column_means)
```

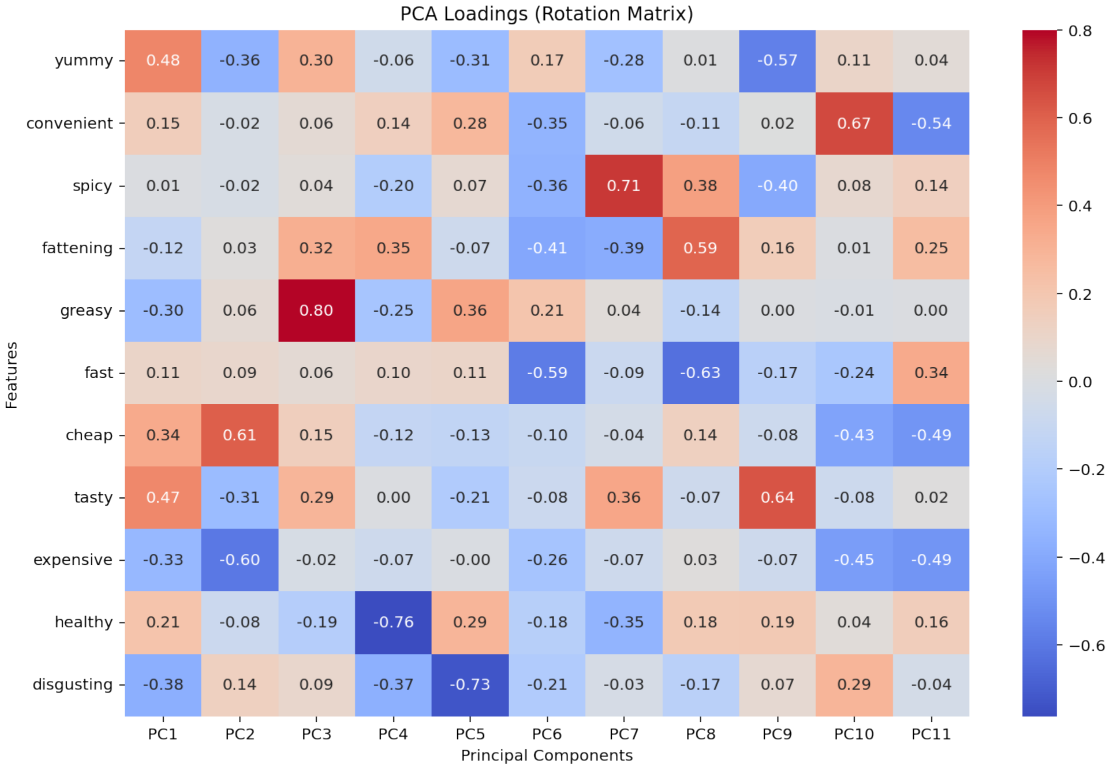
```
[0.55 0.91 0.09 0.87 0.53 0.9  0.6  0.64 0.36 0.2  0.24 0.  ]
```

[10]:
```
from sklearn.cluster import KMeans
```

[11]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Define the rotation matrix (PCA loadings)
rotation_matrix = {
    "PC1": [0.477, 0.155, 0.006, -0.116, -0.304, 0.108, 0.337, 0.472, -0.329, 0.
 ↪214, -0.375],
    "PC2": [-0.36, -0.02, -0.02, 0.03, 0.06, 0.09, 0.61, -0.31, -0.60, -0.08, 0.
 ↪14],
    "PC3": [0.30, 0.06, 0.04, 0.32, 0.80, 0.06, 0.15, 0.29, -0.02, -0.19, 0.09],
    "PC4": [-0.055, 0.142, -0.198, 0.354, -0.254, 0.097, -0.119, 0.003, -0.068,␣
 ↪-0.763, -0.370],
    "PC5": [-0.308, 0.278, 0.071, -0.073, 0.361, 0.108, -0.129, -0.211, -0.003,␣
 ↪0.288, -0.729],
    "PC6": [0.17, -0.35, -0.36, -0.41, 0.21, -0.59, -0.10, -0.08, -0.26, -0.18,␣
 ↪-0.21],
    "PC7": [-0.28, -0.06, 0.71, -0.39, 0.04, -0.09, -0.04, 0.36, -0.07, -0.35,␣
 ↪-0.03],
    "PC8": [0.01, -0.11, 0.38, 0.59, -0.14, -0.63, 0.14, -0.07, 0.03, 0.18, -0.
 ↪17],
    "PC9": [-0.572, 0.018, -0.400, 0.161, 0.003, -0.166, -0.076, 0.639, -0.067,␣
 ↪0.186, 0.072],
    "PC10": [0.110, 0.666, 0.076, 0.005, -0.009, -0.240, -0.428, -0.079, -0.
 ↪454, 0.038, 0.290],
    "PC11": [0.045, -0.542, 0.142, 0.251, 0.002, 0.339, -0.489, 0.020, -0.490,␣
 ↪0.158, -0.041]
}

# Corresponding feature names
features = [
    "yummy", "convenient", "spicy", "fattening", "greasy",
    "fast", "cheap", "tasty", "expensive", "healthy", "disgusting"
]

# Create DataFrame for rotation matrix
rotation_df = pd.DataFrame(rotation_matrix, index=features)

# Display the rotation matrix
print("Rotation Matrix (PCA Loadings):")
print(rotation_df)

# Heatmap for better visualization
plt.figure(figsize=(12, 8))
sns.heatmap(rotation_df, annot=True, cmap="coolwarm", fmt=".2f", cbar=True)
plt.title("PCA Loadings (Rotation Matrix)")
plt.xlabel("Principal Components")
plt.ylabel("Features")
plt.show()

# Analyze top contributors to each principal component
top_contributors = rotation_df.apply(lambda x: x.nlargest(3).index.tolist(),␣
  ↪axis=0)
print("\nTop 3 Contributors to Each Principal Component:")
print(top_contributors)
```

```
Rotation Matrix (PCA Loadings):
              PC1   PC2   PC3    PC4    PC5   PC6   PC7   PC8    PC9   PC10  \
yummy       0.477 -0.36  0.30 -0.055 -0.308  0.17 -0.28  0.01 -0.572  0.110
convenient  0.155 -0.02  0.06  0.142  0.278 -0.35 -0.06 -0.11  0.018  0.666
spicy       0.006 -0.02  0.04 -0.198  0.071 -0.36  0.71  0.38 -0.400  0.076
fattening  -0.116  0.03  0.32  0.354 -0.073 -0.41 -0.39  0.59  0.161  0.005
greasy     -0.304  0.06  0.80 -0.254  0.361  0.21  0.04 -0.14  0.003 -0.009
fast        0.108  0.09  0.06  0.097  0.108 -0.59 -0.09 -0.63 -0.166 -0.240
cheap       0.337  0.61  0.15 -0.119 -0.129 -0.10 -0.04  0.14 -0.076 -0.428
tasty       0.472 -0.31  0.29  0.003 -0.211 -0.08  0.36 -0.07  0.639 -0.079
expensive  -0.329 -0.60 -0.02 -0.068 -0.003 -0.26 -0.07  0.03 -0.067 -0.454
healthy     0.214 -0.08 -0.19 -0.763  0.288 -0.18 -0.35  0.18  0.186  0.038
disgusting -0.375  0.14  0.09 -0.370 -0.729 -0.21 -0.03 -0.17  0.072  0.290

             PC11
yummy       0.045
convenient -0.542
spicy       0.142
fattening   0.251
greasy      0.002
fast        0.339
cheap      -0.489
tasty       0.020
```

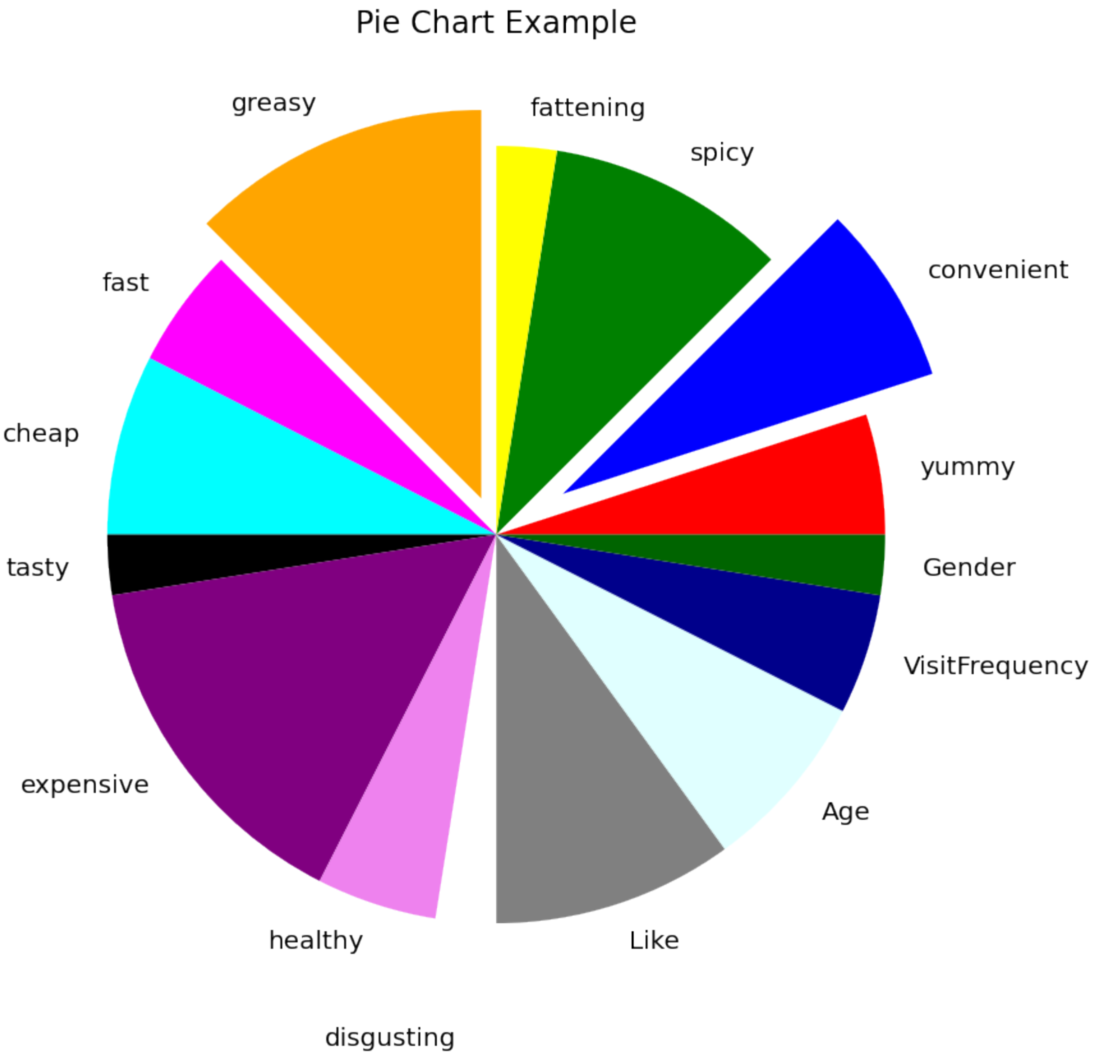
```
expensive  -0.490
healthy     0.158
disgusting -0.041
```



```
Top 3 Contributors to Each Principal Component:
     PC1         PC2        PC3         PC4         PC5      PC6     PC7  \
0  yummy       cheap     greasy   fattening      greasy   greasy   spicy
1  tasty  disgusting  fattening  convenient     healthy    yummy   tasty
2  cheap        fast      yummy        fast  convenient    tasty  greasy

         PC8        PC9        PC10       PC11
0  fattening      tasty  convenient       fast
1      spicy    healthy  disgusting  fattening
2    healthy  fattening       yummy    healthy
```

[12]:

```python
import matplotlib.pyplot as plt

# Define the data
sizes = [10, 15, 20, 5, 25, 10, 15, 5, 30, 10, 5, 20, 15, 10, 5]  # Example sizes
labels = ['yummy', 'convenient', 'spicy', 'fattening', 'greasy', 'fast', 'cheap',
          'tasty', 'expensive', 'healthy', 'disgusting', 'Like', 'Age',
          'VisitFrequency', 'Gender']
colors = ['red', 'blue', 'green', 'yellow', 'orange', 'magenta', 'cyan',
          'black', 'purple', 'violet', 'white', 'gray','lightcyan',
          'darkblue', 'darkgreen']
explode = (0, 0.2, 0, 0, 0.1, 0, 0, 0, 0, 0, 0.2, 0, 0, 0, 0)

# Plot the pie chart
plt.pie( sizes,explode=explode, labels=labels, colors=colors)

# Add a title
plt.title('Pie Chart Example')

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



[13]:

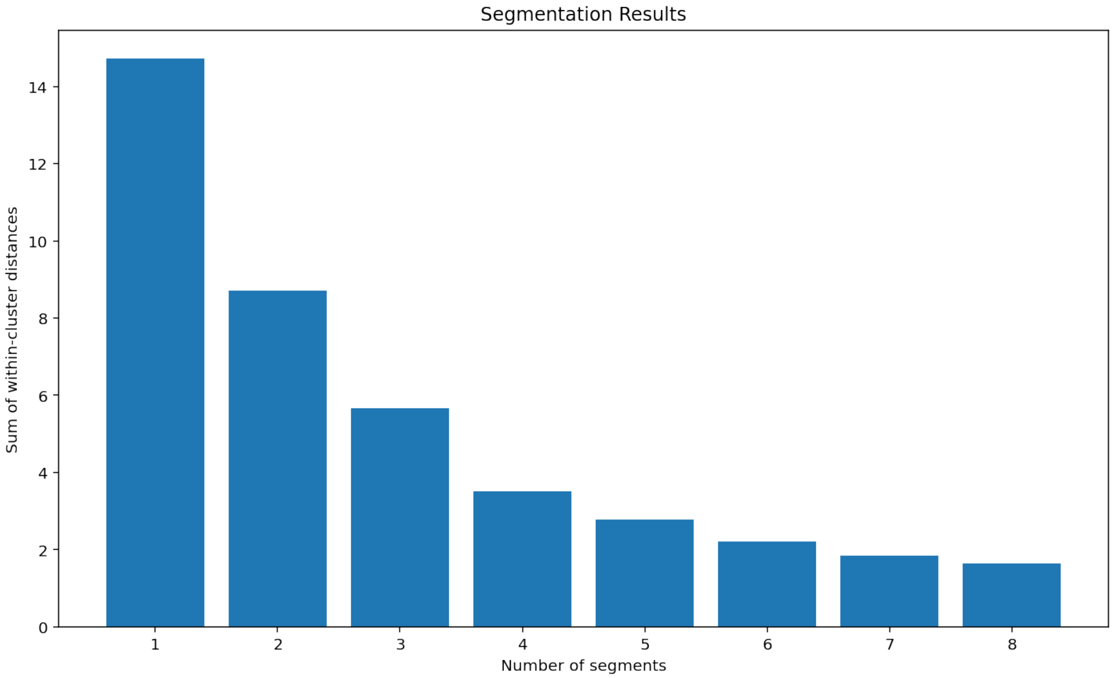
```
import pandas as pd
mcdonalds =pd.DataFrame({'Yammy': [1, 2], 'Convenient': [3, 4],'Spicy':
 ↪[12,12],'fast':[12,12],'cheap':[12,12],'tasty':[12,12],'Gender':[12,12]})
mcdonalds.corr()
```

[13]:

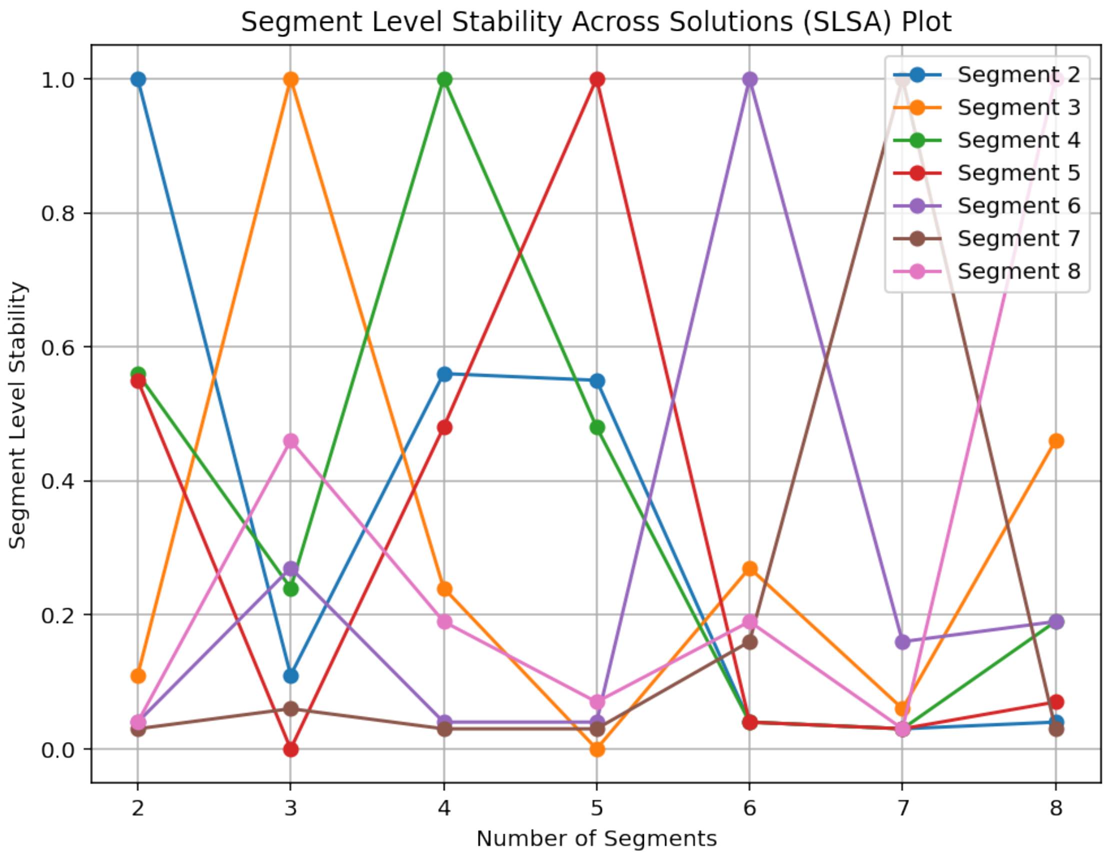
```
            Yammy  Convenient  Spicy  fast  cheap  tasty  Gender
Yammy         1.0         1.0    NaN   NaN    NaN    NaN     NaN
Convenient    1.0         1.0    NaN   NaN    NaN    NaN     NaN
Spicy         NaN         NaN    NaN   NaN    NaN    NaN     NaN
fast          NaN         NaN    NaN   NaN    NaN    NaN     NaN
cheap         NaN         NaN    NaN   NaN    NaN    NaN     NaN
tasty         NaN         NaN    NaN   NaN    NaN    NaN     NaN
Gender        NaN         NaN    NaN   NaN    NaN    NaN     NaN
```

[14]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
MD = np.random.rand(100, 2)

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()
```

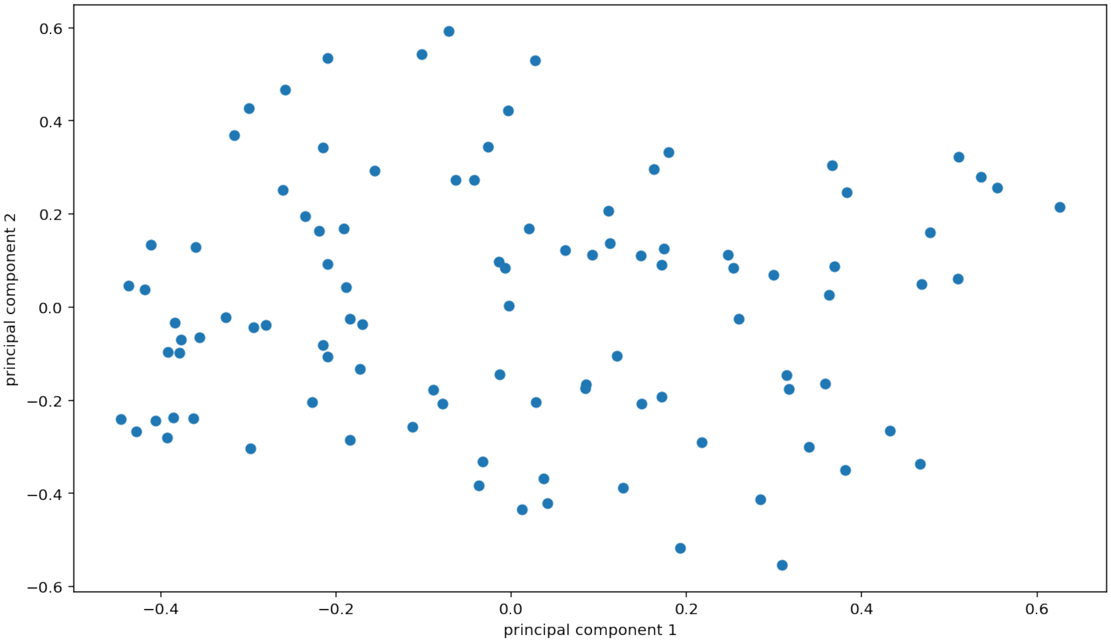


[15]:
```
num_segments = range(2, 9)

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 segment_stability], marker='o', label=f'Segment {segment}')

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()
```



[16]:

```
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()
```


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
principal component 2
−0.4
−0.2
0.0
0.2
0.4
0.6
principal component 1

[ ]: