OwusuSefah630Week4

September 18, 2024

Week 4: Descriptive Modeling

Bernard Owusu Sefah

4.2 4.2 Assignment: Clustering Exercise

DSC 630

[1]:
```
# load the data and inspect its structure to understand which columns might not␣
 ↪be relevant to the ALS condition, then proceed with further steps.
import pandas as pd

# Load the data
file_path = 'als_data.csv'
als_data = pd.read_csv(file_path)

# Display the first few rows and the columns to inspect the structure
als_data.head(), als_data.columns
```

[1]:
```
(   ID  Age_mean  Albumin_max  Albumin_median  Albumin_min  Albumin_range  \
 0   1        65         57.0            40.5         38.0       0.066202
 1   2        48         45.0            41.0         39.0       0.010453
 2   3        38         50.0            47.0         45.0       0.008929
 3   4        63         47.0            44.0         41.0       0.012111
 4   5        63         47.0            45.5         42.0       0.008292

    ALSFRS_slope  ALSFRS_Total_max  ALSFRS_Total_median  ALSFRS_Total_min  …
\
 0     -0.965608                30                 28.0                22  …
 1     -0.921717                37                 33.0                21  …
 2     -0.914787                24                 14.0                10  …
 3     -0.598361                30                 29.0                24  …
 4     -0.444039                32                 27.5                20  …

    Sodium_min  Sodium_range  SubjectID  trunk_max  trunk_median  trunk_min  \
 0       143.0      0.017422        533          8           7.0          7
 1       136.0      0.010453        649          8           7.0          5
 2       140.0      0.008929       1234          5           0.0          0
 3       138.0      0.012469       2492          5           5.0          3
```

```
4       138.0      0.008292       2956          6           4.0          1

   trunk_range  Urine.Ph_max  Urine.Ph_median  Urine.Ph_min
0     0.002646           6.0              6.0           6.0
1     0.005386           7.0              5.0           5.0
2     0.008929           6.0              5.0           5.0
3     0.004988           7.0              6.0           5.0
4     0.008489           6.0              5.0           5.0

[5 rows x 101 columns],
Index(['ID', 'Age_mean', 'Albumin_max', 'Albumin_median', 'Albumin_min',
       'Albumin_range', 'ALSFRS_slope', 'ALSFRS_Total_max',
       'ALSFRS_Total_median', 'ALSFRS_Total_min',
       …
       'Sodium_min', 'Sodium_range', 'SubjectID', 'trunk_max', 'trunk_median',
       'trunk_min', 'trunk_range', 'Urine.Ph_max', 'Urine.Ph_median',
       'Urine.Ph_min'],
      dtype='object', length=101))
```

The dataset contains various columns, including patient IDs, physiological measures, ALS-related scores, and ranges of values. To ensure focus on relevant features for clustering based on ALS condition, I will remove non-ALS-related columns like the ID, SubjectID, and other columns that do not directly impact ALS.

0.1 Step 1: Remove any data that is not relevant to the patient's ALS condition.

[2]:
```
#Keeping columns that seem to be related to ALS progression and patient␣
 ↪conditions

als_data_relevant = als_data.drop(columns=['ID', 'SubjectID'])

# Display the remaining columns after dropping irrelevant ones
als_data_relevant.columns
```

[2]:
```
Index(['Age_mean', 'Albumin_max', 'Albumin_median', 'Albumin_min',
       'Albumin_range', 'ALSFRS_slope', 'ALSFRS_Total_max',
       'ALSFRS_Total_median', 'ALSFRS_Total_min', 'ALSFRS_Total_range',
       'ALT.SGPT._max', 'ALT.SGPT._median', 'ALT.SGPT._min', 'ALT.SGPT._range',
       'AST.SGOT._max', 'AST.SGOT._median', 'AST.SGOT._min', 'AST.SGOT._range',
       'Bicarbonate_max', 'Bicarbonate_median', 'Bicarbonate_min',
       'Bicarbonate_range', 'Blood.Urea.Nitrogen..BUN._max',
       'Blood.Urea.Nitrogen..BUN._median', 'Blood.Urea.Nitrogen..BUN._min',
       'Blood.Urea.Nitrogen..BUN._range', 'bp_diastolic_max',
       'bp_diastolic_median', 'bp_diastolic_min', 'bp_diastolic_range',
       'bp_systolic_max', 'bp_systolic_median', 'bp_systolic_min',
       'bp_systolic_range', 'Calcium_max', 'Calcium_median', 'Calcium_min',
       'Calcium_range', 'Chloride_max', 'Chloride_median', 'Chloride_min',
```

```
       'Chloride_range', 'Creatinine_max', 'Creatinine_median',
       'Creatinine_min', 'Creatinine_range', 'Gender_mean', 'Glucose_max',
       'Glucose_median', 'Glucose_min', 'Glucose_range', 'hands_max',
       'hands_median', 'hands_min', 'hands_range', 'Hematocrit_max',
       'Hematocrit_median', 'Hematocrit_min', 'Hematocrit_range',
       'Hemoglobin_max', 'Hemoglobin_median', 'Hemoglobin_min',
       'Hemoglobin_range', 'leg_max', 'leg_median', 'leg_min', 'leg_range',
       'mouth_max', 'mouth_median', 'mouth_min', 'mouth_range',
       'onset_delta_mean', 'onset_site_mean', 'Platelets_max',
       'Platelets_median', 'Platelets_min', 'Potassium_max',
       'Potassium_median', 'Potassium_min', 'Potassium_range', 'pulse_max',
       'pulse_median', 'pulse_min', 'pulse_range', 'respiratory_max',
       'respiratory_median', 'respiratory_min', 'respiratory_range',
       'Sodium_max', 'Sodium_median', 'Sodium_min', 'Sodium_range',
       'trunk_max', 'trunk_median', 'trunk_min', 'trunk_range', 'Urine.Ph_max',
       'Urine.Ph_median', 'Urine.Ph_min'],
      dtype='object')
```

The remaining columns include several physiological measurements and ALS-related metrics. Removing of ID and SubjectID was success

1 Step 2: Apply a standard scaler to the data.

Will apply a standard scaler to normalize the data for clustering

[3]:
```
from sklearn.preprocessing import StandardScaler

# Apply standard scaling to the relevant ALS data
scaler = StandardScaler()
scaled_als_data = scaler.fit_transform(als_data_relevant)

# Check the scaled data by displaying the first few rows
scaled_als_data[:5]
```

[3]:
```
array([[ 9.17136983e-01,  3.08941722e+00, -1.30078105e+00,
        -8.66550089e-01,  5.48092948e+00, -3.81450341e-01,
        -3.18520120e-01,  1.34959879e-01,  2.47367899e-01,
        -3.01588467e-01, -6.79085269e-01, -7.04767241e-01,
        -4.46605772e-01, -4.53086914e-01, -3.43749805e-01,
        -1.64404050e-01,  1.97133625e-01, -2.53516298e-01,
        -2.83618538e-01,  4.71373703e-01,  7.62329697e-01,
         4.90774047e-02,  2.92859599e-01,  1.16967352e+00,
         1.78065854e+00, -4.03233483e-01, -2.31901530e-01,
         2.60443579e-01, -1.05509751e-01,  2.89532342e-01,
         8.16773688e-01,  7.74853650e-01,  1.39114351e+00,
         3.27228335e-01,  1.11898292e-01, -1.41443932e+00,
        -1.33418881e-02,  8.63964983e-01,  7.06250323e-01,
```

```
  1.93837712e+00,  1.08444436e+00,  2.60029864e-01,
  3.99225881e-02,  8.27717849e-01,  1.01594400e+00,
 -7.33335170e-01, -1.32592025e+00,  1.09146338e-01,
 -7.97425559e-01, -3.79966423e-02,  7.89353067e-01,
  9.19974025e-01,  1.05490429e+00,  1.13257254e+00,
 -3.35338214e-01,  2.05758711e-01,  3.03909454e-01,
  3.23428459e-01,  1.91746096e-01,  3.03416425e-01,
  1.47494944e-01,  5.06397326e-01,  2.00821914e-01,
  1.19870045e+00,  1.07456347e+00,  7.09255006e-01,
  1.01107625e+00, -3.02736718e+00, -2.23847738e+00,
 -2.08016715e+00,  1.08267679e+00, -8.25839398e-01,
 -1.90993281e+00, -1.61491746e+00, -1.32627662e+00,
 -1.14341597e+00, -9.68724690e-02,  2.49900646e-01,
  5.42514984e-01, -5.70354132e-04, -1.04863895e+00,
 -1.06370228e+00, -5.63513096e-01, -2.17842127e-01,
  3.01453800e-01, -9.68019231e-01,  1.97809628e-01,
  5.33105891e-02,  1.95964045e+00,  2.99234197e+00,
  2.30047015e+00,  2.60968296e-01,  1.02801779e+00,
  9.81832469e-01,  1.71536537e+00, -9.97419581e-01,
 -8.80375507e-01,  4.63053547e-01,  1.86853157e+00],
[-5.74878674e-01, -6.22015605e-01, -1.11240084e+00,
 -5.53303052e-01, -3.47725290e-01, -3.10907036e-01,
  9.98994923e-01,  8.88863159e-01,  1.30839219e-01,
  1.66537451e-01, -6.56773548e-01, -1.28174892e+00,
 -1.33718125e+00, -3.74514806e-01, -3.43749805e-01,
 -1.25909443e+00, -1.02012894e+00, -2.32796373e-01,
  3.48540694e-01,  4.71373703e-01,  7.62329697e-01,
 -4.14377997e-01,  4.50297076e-01, -6.08052234e-01,
 -1.06927827e-01,  9.86556290e-02, -1.37394208e+00,
 -4.29787795e-01, -6.97803123e-01, -6.94128577e-01,
 -4.53026933e-01,  2.40235156e-01, -8.42491270e-01,
 -1.97758390e-01, -8.57275731e-01, -1.97893172e+00,
 -1.16877867e+00, -3.96840705e-02,  3.24042648e-01,
 -6.31201678e-01,  2.13634926e-01, -3.36276748e-01,
 -8.62669864e-01, -7.00244723e-01, -4.21909426e-01,
 -7.33335170e-01, -1.32592025e+00, -1.53752256e-01,
 -3.95742769e-01, -1.72185912e-01, -2.54892757e-01,
  9.19974025e-01,  4.45034176e-01,  1.13257254e+00,
 -6.93561585e-01, -3.05368252e-03,  1.09571687e-02,
  6.38254401e-02, -3.65663577e-01, -1.11187137e+00,
 -1.05937705e+00, -5.01175413e-01, -5.92730268e-01,
  1.19870045e+00,  1.51310846e+00,  2.38465553e-01,
  6.43794256e-01, -9.18857039e-01, -6.14647905e-01,
 -1.01042877e+00,  3.88825780e-01,  8.30675360e-01,
 -1.90993281e+00,  9.95097428e-03,  4.79303596e-01,
  4.38408688e-01,  2.80453538e-01,  4.55644558e-01,
  1.62273848e-01,  5.95561456e-02, -5.78592952e-02,
```

```
 -1.15368838e-01, -1.76999794e-01, -2.52168315e-01,
  3.01453800e-01,  6.65306576e-01,  1.97809628e-01,
 -2.89978833e-01, -6.12561204e-01, -1.19881209e+00,
 -2.78144366e-01, -4.89913153e-01,  1.02801779e+00,
  9.81832469e-01,  8.67032287e-01, -3.88669250e-01,
  1.92664503e-01, -1.13720768e+00, -4.19151242e-01],
[-1.45253494e+00,  9.24414737e-01,  1.14816173e+00,
  1.32617917e+00, -5.07102648e-01, -2.99768618e-01,
 -1.44781873e+00, -1.97596931e+00, -1.15097626e+00,
 -6.40998269e-02, -6.56773548e-01, -8.32985391e-01,
 -8.02835962e-01, -4.64479867e-01, -5.42158214e-01,
 -1.05058198e+00, -4.79123355e-01, -4.57607474e-01,
  1.29677954e+00,  9.26264521e-01,  3.47043370e-01,
  2.46045913e-01, -8.09202743e-01, -8.81548503e-01,
 -3.76583021e-01, -7.63966400e-01, -6.88717748e-01,
 -7.05880345e-01, -1.40855517e+00,  8.58523554e-02,
 -1.72282755e+00, -1.61036732e+00, -2.09332675e+00,
 -5.31883234e-01, -2.65551400e-02, -5.67700726e-01,
 -1.57771485e-01, -1.05199661e-01,  3.24042648e-01,
  1.08185086e+00,  1.37471418e+00, -9.17675697e-01,
  4.91218814e-01,  8.27717849e-01,  1.01594400e+00,
 -7.12886609e-01,  7.54193171e-01, -5.91916580e-01,
 -3.06479926e-01, -3.79966423e-02, -6.03234435e-01,
 -1.10337721e+00, -1.58786621e+00, -1.16880658e+00,
  5.48050964e-02,  5.53779367e-01,  5.55600853e-01,
  6.08991780e-01, -2.06569538e-01,  6.96551924e-01,
  8.37136085e-01,  1.04376945e+00, -5.80331017e-01,
 -5.83715003e-01, -4.60344015e-01, -2.32323900e-01,
 -5.92848454e-01, -3.91729504e-01, -9.75498900e-01,
 -1.01042877e+00,  6.72451948e-01, -1.20960675e+00,
 -1.90993281e+00, -7.45470314e-01, -4.90007890e-01,
 -8.39218922e-01, -3.98733275e-01, -7.78818913e-01,
  1.62273848e-01, -4.78575958e-01, -7.78426317e-01,
 -4.70993879e-01, -6.92350863e-01, -5.22019010e-01,
  3.01453800e-01,  6.65306576e-01,  1.14550532e+00,
 -1.01491356e+00,  6.73539623e-01,  1.59529062e+00,
  1.19534965e+00, -6.54168554e-01, -6.88949560e-01,
 -2.28066914e+00, -1.25380043e+00,  3.98248909e-01,
 -8.80375507e-01, -1.13720768e+00, -4.19151242e-01],
[ 7.41605729e-01, -3.44346837e-03,  1.78804432e-02,
  7.31910228e-02, -1.74360825e-01,  2.08800524e-01,
 -3.18520120e-01,  2.85740535e-01,  4.80425258e-01,
 -6.85523953e-01,  1.68760113e-01,  1.73137761e+00,
  1.60171782e+00, -1.69286832e-01,  8.14110721e-02,
  1.13879878e+00,  1.54964759e+00, -1.99468823e-01,
 -2.49617585e+00, -3.16775284e+00, -1.31410194e+00,
 -8.32373902e-01,  2.92859599e-01,  1.85341419e+00,
```

```
  1.78065854e+00, -6.88596957e-01, -2.31901530e-01,
 -1.53695245e-01,  1.29489239e-02,  8.12809947e-02,
  1.81873377e-01,  3.46126580e-02,  5.87034991e-01,
  1.09797336e-01, -2.65551400e-02, -3.20833028e-03,
  4.13619072e-02, -1.38784873e-01,  7.06250323e-01,
  1.51011399e+00,  1.95525380e+00, -8.88715386e-01,
 -4.11373638e-01, -1.90923866e-01,  5.73750490e-02,
 -3.80457338e-01,  7.54193171e-01, -8.10998742e-01,
 -5.74268453e-01,  3.19841410e-01, -8.40116621e-01,
 -9.17015909e-02,  2.41744137e-01,  3.65446166e-01,
 -3.99273319e-01,  3.37233181e-01,  2.91531188e-01,
  4.09962799e-01,  3.54894053e-03,  1.46162226e-01,
  6.12898011e-02,  5.73568842e-01, -3.78758122e-01,
 -5.83715003e-01, -2.41071518e-01, -2.32323900e-01,
 -2.68863801e-01,  6.62525566e-01,  8.28756075e-01,
  1.12904798e+00, -1.07640173e+00,  7.72381586e-01,
  4.75337211e-01, -1.46834576e-01, -1.09885739e-01,
 -8.88661986e-02, -2.47802872e-01,  4.41567343e-02,
  5.42514984e-01, -3.43636115e-01, -5.98284561e-01,
 -5.89535560e-01,  3.38351276e-01, -4.99587731e-01,
 -3.04920518e+00, -9.68019231e-01,  1.97809628e-01,
 -1.01491356e+00, -1.83860928e-01, -6.39991551e-01,
  4.58602640e-01, -2.72700588e-01, -6.88949560e-01,
  4.96891512e-02,  1.86991999e-02, -4.77180952e-01,
  1.92664503e-01,  4.63053547e-01, -4.19151242e-01],
[ 7.41605729e-01, -3.44346837e-03,  5.83021085e-01,
  3.86438060e-01, -5.73670271e-01,  4.56830683e-01,
  5.79127491e-02,  5.95695507e-02,  1.43105396e-02,
 -3.50529411e-01, -3.66721180e-01, -4.16276403e-01,
 -9.03755830e-02, -4.02321251e-01, -2.30373571e-01,
 -2.68660277e-01, -2.08620563e-01, -2.89183413e-01,
  3.48540694e-01,  4.71373703e-01, -6.82429579e-02,
 -1.72274453e-01, -9.66640220e-01, -1.01829664e+00,
 -3.76583021e-01, -9.29259818e-01,  9.10139014e-01,
 -1.53695245e-01, -2.23968425e-01,  1.98332325e-01,
  8.16773688e-01,  3.46126580e-02, -8.42491270e-01,
  6.55247324e-01, -3.03462004e-01, -9.91070023e-01,
 -3.02201082e-01, -2.63872376e-01, -5.81650277e-02,
  2.25324589e-01,  2.13634926e-01, -5.35540014e-01,
 -8.62669864e-01, -9.54905152e-01, -1.38047838e+00,
  5.93270102e-03, -1.32592025e+00,  1.31054555e-01,
  1.84465706e-01,  6.32949706e-01, -3.65802418e-01,
  9.19974025e-01,  6.48324214e-01, -1.81170196e-02,
  3.38399380e-01,  1.59355957e-01,  2.79152923e-01,
  2.19587252e-01, -3.52722005e-01, -1.10919741e-02,
  1.90597515e-01,  1.70539746e-01, -4.27735904e-01,
 -1.47492273e+00, -8.98889010e-01, -1.17390281e+00,
```

```
       -6.33078913e-01,  6.62525566e-01,  8.28756075e-01,
        1.12904798e+00, -1.07640173e+00, -2.63537532e+00,
        4.75337211e-01,  3.94788234e-01,  8.49922694e-01,
        1.20904122e+00, -2.14072677e-02, -1.80753847e+00,
       -1.35869069e+00,  2.77479379e-02,  9.32920359e-01,
        2.25546477e+00,  1.11137788e+00, -3.21612512e-01,
        3.01453800e-01,  6.65306576e-01,  1.97809628e-01,
       -3.29363976e-01, -1.83860928e-01, -8.11710090e-02,
        4.58602640e-01, -7.22773605e-01, -1.16627112e-01,
       -4.16382507e-01, -8.29633888e-01,  3.00598265e-01,
       -8.80375507e-01, -1.13720768e+00, -4.19151242e-01]])
```

With the above data being successfully scaled will proceed to creating a plot of the cluster silhouette score

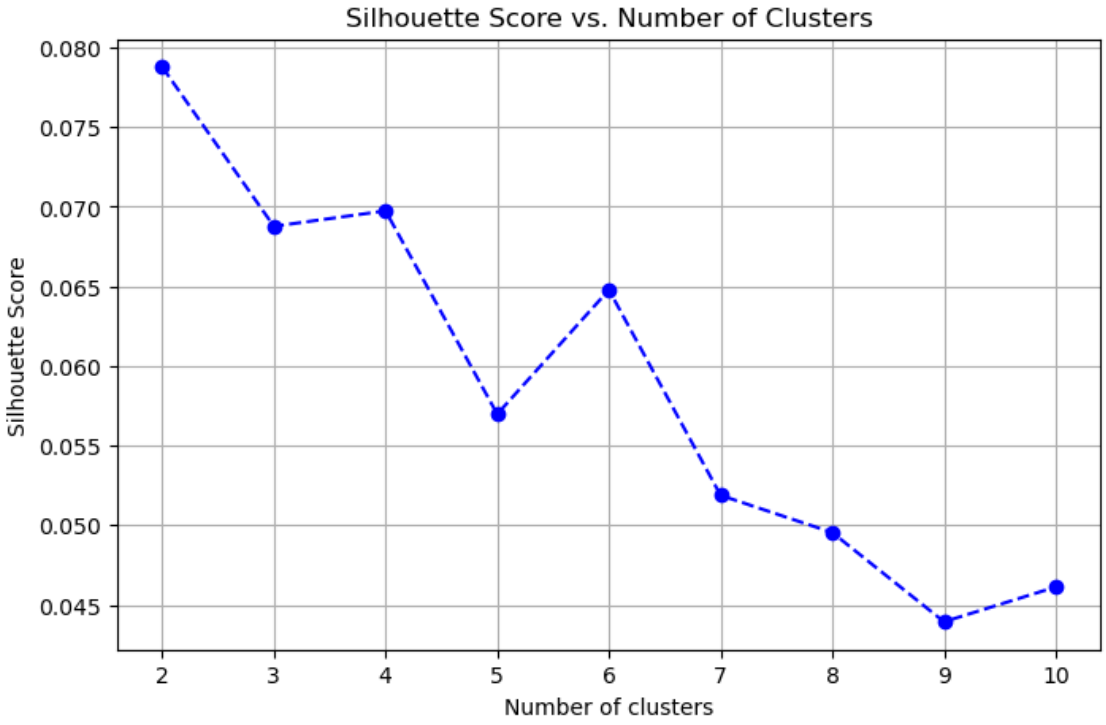
1.1 Step 3: Create a plot of the cluster silhouette score versus the number of clusters in a K-means cluster.

Will use different values for the number of clusters (e.g., from 2 to 10) and calculate the silhouette score for each, then plot the results.

[8]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import os

# Setting the environment variable to prevent memory leak in Windows with MKL
os.environ['OMP_NUM_THREADS'] = '9'

# Define range for number of clusters
cluster_range = range(2, 11)
silhouette_scores = []

# Calculate silhouette score for each number of clusters
for n_clusters in cluster_range:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=11)  #␣
 ↪n_init set to 10 to remove the warning
    cluster_labels = kmeans.fit_predict(scaled_als_data)
    silhouette_avg = silhouette_score(scaled_als_data, cluster_labels)
    silhouette_scores.append(silhouette_avg)

# Plot the silhouette scores
plt.figure(figsize=(8, 5))
plt.plot(cluster_range, silhouette_scores, marker='o', linestyle='--',␣
 ↪color='b')
plt.title('Silhouette Score vs. Number of Clusters')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
```

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

```
C:\Users\berna\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\berna\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\berna\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\berna\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\berna\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\berna\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\berna\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\berna\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\berna\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=9.
  warnings.warn(
```



Plot Interpretation: The Silhouette Score represents how well each point in a cluster is similar to its own cluster compared to other clusters. A higher silhouette score indicates better-defined clusters. * The peak silhouette score is at 2 clusters, indicating that using 2 clusters might result in the most distinct separation between clusters. * After 2 clusters, the silhouette score generally declines, showing that increasing the number of clusters does not improve the clustering significantly. Given this plot, 2 or possibly 3 clusters seem to be optimal choices, with 2 clusters being the most well-defined.

1.2 Step 4: Choosing the optimal number of clusters

From the plot, 2 clusters is the optimal choice based on the highest silhouette score. The score drops significantly after 2 clusters.

1.3 Step 5: Fit a K-means model to the data with 2 clusters

[9]:
```
# Fit K-means model with 2 clusters
optimal_clusters = 2
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42, n_init=10)
cluster_labels = kmeans.fit_predict(scaled_als_data)

# The cluster labels (0 or 1) are now assigned to each data point
print(cluster_labels[:10])  # Print first 10 cluster assignments
```

```
C:\Users\berna\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=9.
  warnings.warn(

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

Each number in this array (either 0 or 1) represents the cluster assignment of the corresponding data point in the dataset.

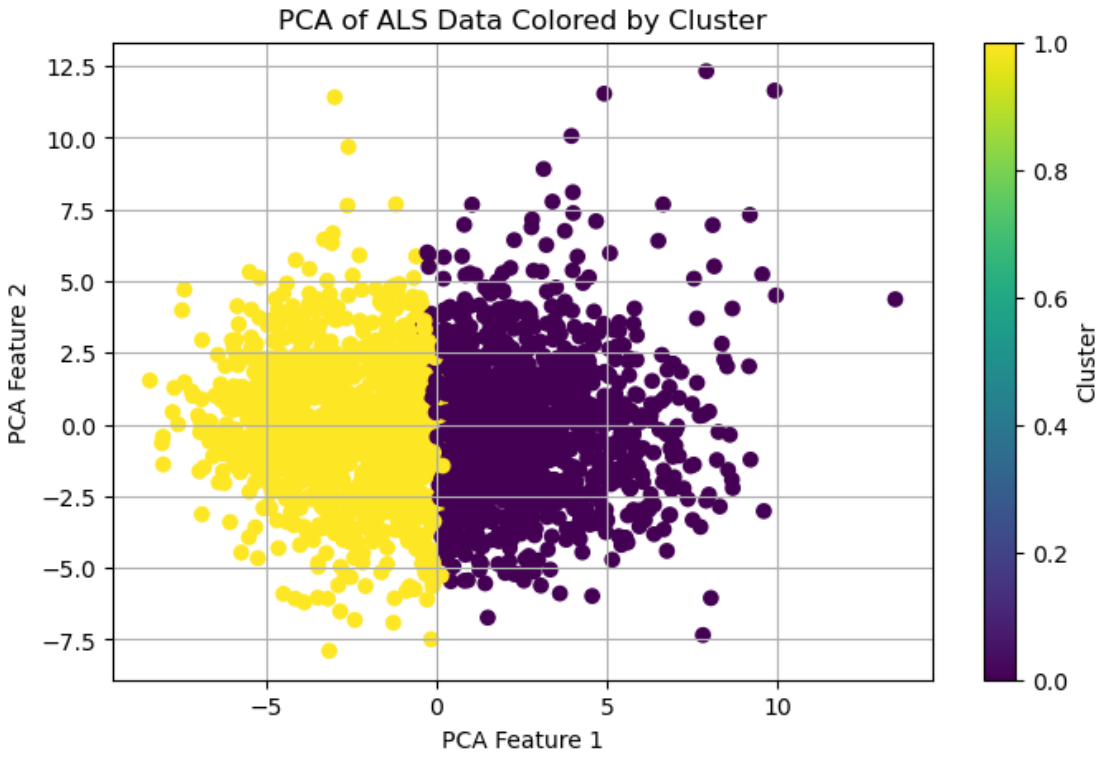
- Cluster 0: Data points assigned to cluster 0.
- Cluster 1: Data points assigned to cluster 1. The first data point (at index 0) is assigned to Cluster 1, the second data point (at index 1) is also assigned to Cluster 1, and the third data point (at index 2) is assigned to Cluster 0.

In other words, K-means has divided the dataset into two groups (clusters), and each point has been assigned to one of the two clusters based on the features in the dataset.

1.4 Step 6: Fit a PCA transformation with two features to the scaled data

Will now reduce the dimensionality of the scaled data to 2 features using PCA, which will help visualize the clustering results.

[10]:
```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Fit PCA and transform the data
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_als_data)

# Make a scatter plot of the PCA transformed data, coloring each point by its␣
 ↪cluster
plt.figure(figsize=(8, 5))
plt.scatter(pca_data[:, 0], pca_data[:, 1], c=cluster_labels, cmap='viridis')
plt.title('PCA of ALS Data Colored by Cluster')
plt.xlabel('PCA Feature 1')
plt.ylabel('PCA Feature 2')
plt.colorbar(label='Cluster')
plt.grid(True)
plt.show()
```



Plot Details:

1. PCA Transformation:

- The data has been reduced to two dimensions (PCA Feature 1 and PCA Feature 2) from its original high-dimensional space.
- PCA Feature 1 and PCA Feature 2 represent the two most significant principal components that capture the maximum variance in the data. Essentially, PCA helps visualize high-dimensional data in a lower-dimensional space (2D in this case) while preserving as much information as possible.

2. Cluster Assignments:

- The data points are colored based on their cluster assignments from the K-means model.
- Yellow points represent data assigned to Cluster 0.
- Purple points represent data assigned to Cluster 1.
- The color bar on the right shows the cluster values, where 0 corresponds to yellow and 1 corresponds to purple.

3. Interpretation of Clusters:

- The points are relatively well separated into two distinct groups, showing that the clustering was able to differentiate between two distinct patterns in the ALS dataset.
- The plot shows that most points from Cluster 0 are located on the left side (yellow region), while points from Cluster 1 are on the right side (purple region).

- There is some overlap between the two clusters, which is expected in most real-world data, but overall, the separation seems reasonably clear.

4. Separation in PCA Space:

- The horizontal axis (PCA Feature 1) appears to be the primary dimension separating the clusters, as the majority of separation happens along this axis.
- The vertical axis (PCA Feature 2) shows less variation between clusters, suggesting that most of the variance captured by PCA is in PCA Feature 1.

1.5 Conclusion

In this analysis, applied K-means clustering to anonymized ALS patient data and visualized the results using PCA. Here are the key steps and findings:

1. Data Preparation:

- Irrelevant columns such as ID and SubjectID were removed to focus on features related to ALS.
- The remaining dataset was scaled using a StandardScaler to ensure that all features were on a similar scale, improving the effectiveness of clustering algorithms.

2. Optimal Number of Clusters:

- To determine the optimal number of clusters, evaluated the silhouette scores for different cluster numbers (ranging from 2 to 10).
- The silhouette score plot showed that 2 clusters provided the highest score, indicating that two distinct groups could be meaningfully identified in the dataset.

3. K-means Clustering:

- Applied the K-means algorithm with 2 clusters based on the silhouette score results.
- Each patient in the dataset was assigned to one of the two clusters, representing distinct groups with different patterns in the dataset.

4. PCA Visualization:

- To visualize the clusters, reduced the high-dimensional dataset into two principal components using PCA.
- The PCA plot showed a clear separation between the two clusters, with one cluster (yellow) mainly occupying the left side of the plot, and the other cluster (purple) on the right.
- While some overlap between clusters was present, the general separation suggested that there are meaningful differences between the two groups.

Cluster Distinction: The K-means model successfully grouped the ALS patients into two distinct clusters, indicating that there are patterns in the data that differentiate patients into two broad categories.

Separation in PCA Space: The PCA plot highlights that most of the variation between the clusters is captured along PCA Feature 1, suggesting that one dimension in the data plays a significant role in differentiating the two groups.

Silhouette Scores: The silhouette score for two clusters was the highest, confirming that the choice of two clusters was appropriate for this dataset.

The analysis revealed that the ALS patient data can be effectively grouped into two clusters, which likely represent distinct patient profiles or stages of disease progression. Further investigation could involve analyzing the specific characteristics of each cluster to understand what differentiates the groups. This information could be useful in tailoring treatments or interventions based on the cluster a patient belongs to.

[ ]: