Christian Campbell

Clustering Exercise

In [11]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
```

In [3]:
```
path = r"C:\Users\chris\Documents\Bellevue University\7 - Predictive Analytics\als_data.csv"
als_df = pd.read_csv(path)
als_df.head()
```

Out[3]:

	ID	Age_mean	Albumin_max	Albumin_median	Albumin_min	Albumin_range	ALSFRS_slope	ALSFRS_Total_max	ALSFRS_Tot
0	1	65	57.0	40.5	38.0	0.066202	-0.965608	30	
1	2	48	45.0	41.0	39.0	0.010453	-0.921717	37	
2	3	38	50.0	47.0	45.0	0.008929	-0.914787	24	
3	4	63	47.0	44.0	41.0	0.012111	-0.598361	30	
4	5	63	47.0	45.5	42.0	0.008292	-0.444039	32	

5 rows × 101 columns

1.

In [4]:
```
als_df1 = als_df.drop(columns=['ID'])
als_df1.head()
```

Out[4]:

	Age_mean	Albumin_max	Albumin_median	Albumin_min	Albumin_range	ALSFRS_slope	ALSFRS_Total_max	ALSFRS_Total_m
0	65	57.0	40.5	38.0	0.066202	-0.965608	30	
1	48	45.0	41.0	39.0	0.010453	-0.921717	37	
2	38	50.0	47.0	45.0	0.008929	-0.914787	24	
3	63	47.0	44.0	41.0	0.012111	-0.598361	30	
4	63	47.0	45.5	42.0	0.008292	-0.444039	32	

5 rows × 100 columns

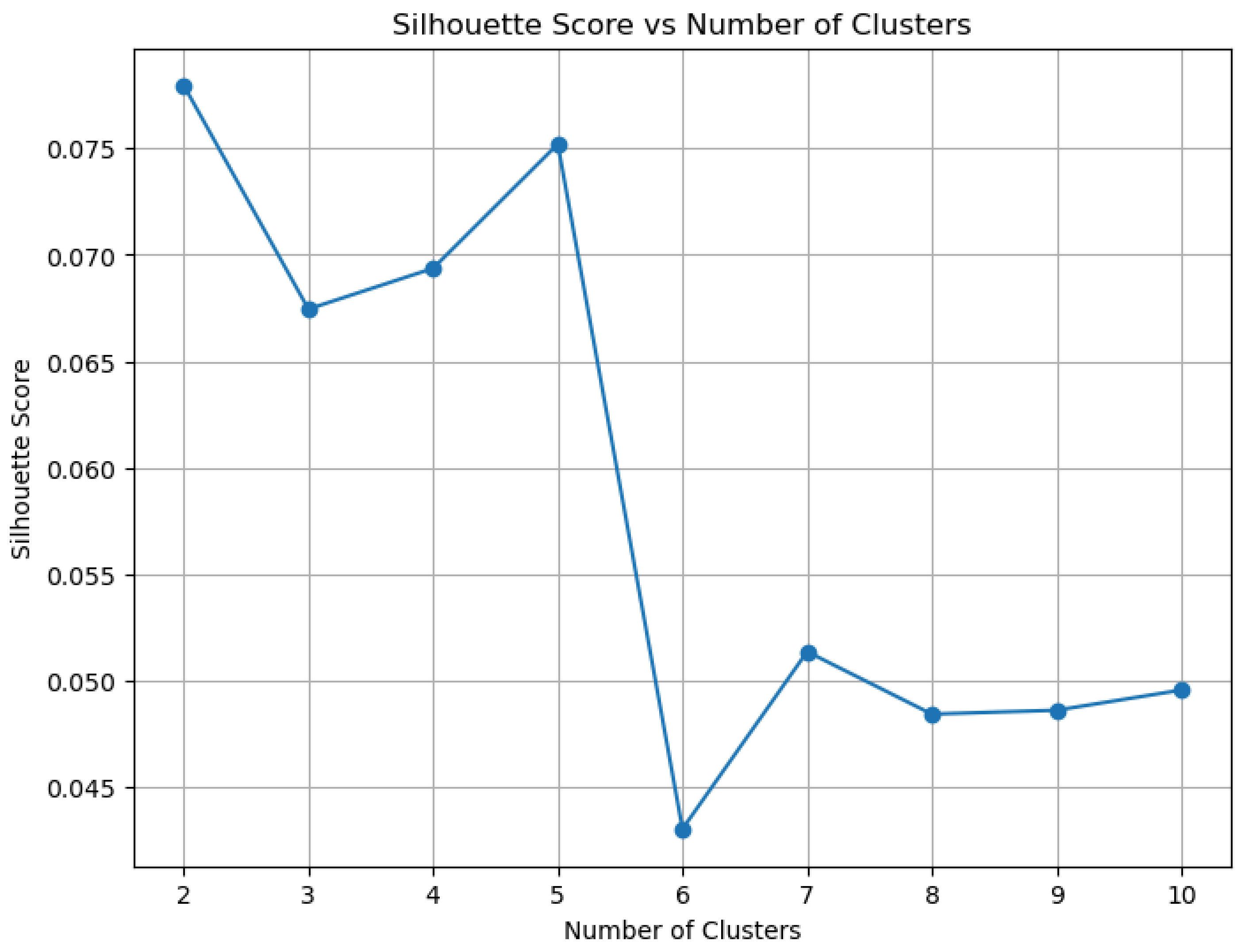
2.

In [6]:
```
scaler = StandardScaler()

# Fits the scaler and transform the data
scaled_data = scaler.fit_transform(als_df1)

# Converts the scaled data back to a DataFrame
scaled_df = pd.DataFrame(scaled_data, columns=als_df1.columns)
```

3.

In [8]:
```python
X = scaled_df.values

# Defines the range of clusters to test
range_n_clusters = list(range(2, 11))  # Example: testing from 2 to 10 clusters

# creates a list to hold silhouette scores
silhouette_scores = []

for n_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    silhouette_scores.append(silhouette_avg)

# Plots the results
plt.figure(figsize=(8, 6))
plt.plot(range_n_clusters, silhouette_scores, marker='o')
plt.title('Silhouette Score vs Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.xticks(range_n_clusters)
plt.grid(True)
plt.show()
```

```
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default v
alue 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(
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default v
alue 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(
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default v
alue 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(
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default v
alue 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(
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default v
alue 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(
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default v
alue 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(
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default v
alue 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(
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default v
alue 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(
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default v
alue 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(
```



4.

I choose 2 as the number of clusters. The reason being that the silhouette score is highest at 2 clusters, indicating well-separated and cohesive clusters. Also the silhouette score decreases "overall"as the number of clusters increases beyond 2, suggesting that more clusters do not provide better separation or might lead to overfitting.

5.

In [9]:
```
X = scaled_df.values

# Creates the KMeans model with 2 clusters
kmeans = KMeans(n_clusters=2, random_state=42)

# Fits the model to the data
kmeans.fit(X)

# Adds cluster labels to the scaled_df DataFrame
scaled_df['Cluster'] = kmeans.labels_

# Gets cluster centroids
centroids = kmeans.cluster_centers_

# Prints cluster labels and centroids
print("Cluster labels:\n", kmeans.labels_)
print("Cluster centroids:\n", centroids)
```

```
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default v
alue 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(
Cluster labels:
 [1 1 0 ... 1 1 0]
Cluster centroids:
 [[-0.00642016 -0.08901726 -0.13820845 -0.18534935  0.18913127 -0.48351683
  -0.58209102 -0.70816775 -0.78709996  0.57458728  0.00840946 -0.08653956
  -0.1407987   0.10635349 -0.04735121 -0.15086672 -0.1864486   0.06048401
  -0.01291302 -0.06128745 -0.04435182  0.15511697 -0.05915251 -0.12794127
  -0.16205236  0.16696102  0.0376983  -0.00911387 -0.03501135  0.15326337
   0.00296774 -0.04040525 -0.08390136  0.17543205  0.05469435  0.09990415
   0.05757728  0.07542603 -0.0783955  -0.11871475 -0.1591092   0.19785201
  -0.25278736 -0.28566273 -0.34302456  0.17433355 -0.13487244  0.07915907
   0.0533418   0.01983894  0.13628384 -0.4342482  -0.54116998 -0.64828124
   0.45648136  0.04729247  0.03015676  0.00536755  0.21901625 -0.01938717
  -0.08531335 -0.14564785  0.22357562 -0.32816581 -0.44175825 -0.56811692
   0.31090386 -0.24057842 -0.32261565 -0.43914604  0.47145773  0.04726187
  -0.04301272  0.17295705  0.12720217  0.07582623 -0.01047876 -0.06583927
  -0.06631656  0.07525728  0.23785582  0.19744652  0.14712768  0.22149343
  -0.16383158 -0.32220014 -0.4345572   0.41410333 -0.04504554 -0.04821622
  -0.08369365  0.17051115 -0.02254309 -0.5083248  -0.62252151 -0.72598273
   0.47526602  0.10001925  0.06948164  0.0125999 ]
 [ 0.00639134  0.08861772  0.13758812  0.18451745 -0.18828239  0.48134664
   0.5794784   0.70498926  0.78356719 -0.57200835 -0.00837171  0.08615114
   0.14016675 -0.10587614  0.04713868  0.15018958  0.18561176 -0.06021254
   0.01285506  0.06101237  0.04415276 -0.15442075  0.05888702  0.12736703
   0.16132502 -0.16621164 -0.0375291   0.00907296  0.03485421 -0.15257547
  -0.00295442  0.0402239   0.08352479 -0.17464466 -0.05444887 -0.09945574
  -0.05731886 -0.0750875   0.07804363  0.11818192  0.15839507 -0.19696399
   0.25165277  0.28438058  0.34148495 -0.17355108  0.13426709 -0.07880378
  -0.05310239 -0.0197499  -0.13567215  0.43229915  0.53874103  0.64537154
  -0.45443253 -0.0470802  -0.03002141 -0.00534346 -0.21803324  0.01930015
   0.08493044  0.14499413 -0.22257213  0.32669289  0.43977549  0.56556702
  -0.30950842  0.23949862  0.32116765  0.437175   -0.46934167 -0.04704975
   0.04281967 -0.17218076 -0.12663125 -0.0754859   0.01043173  0.06554376
   0.06601891 -0.0749195  -0.23678825 -0.19656031 -0.14646732 -0.2204993
   0.16309625  0.320754    0.43260677 -0.4122447   0.04484336  0.04799981
   0.083318   -0.16974584  0.02244191  0.50604327  0.61972742  0.72272428
  -0.47313287 -0.09957033 -0.06916978 -0.01254334]]
```

6.

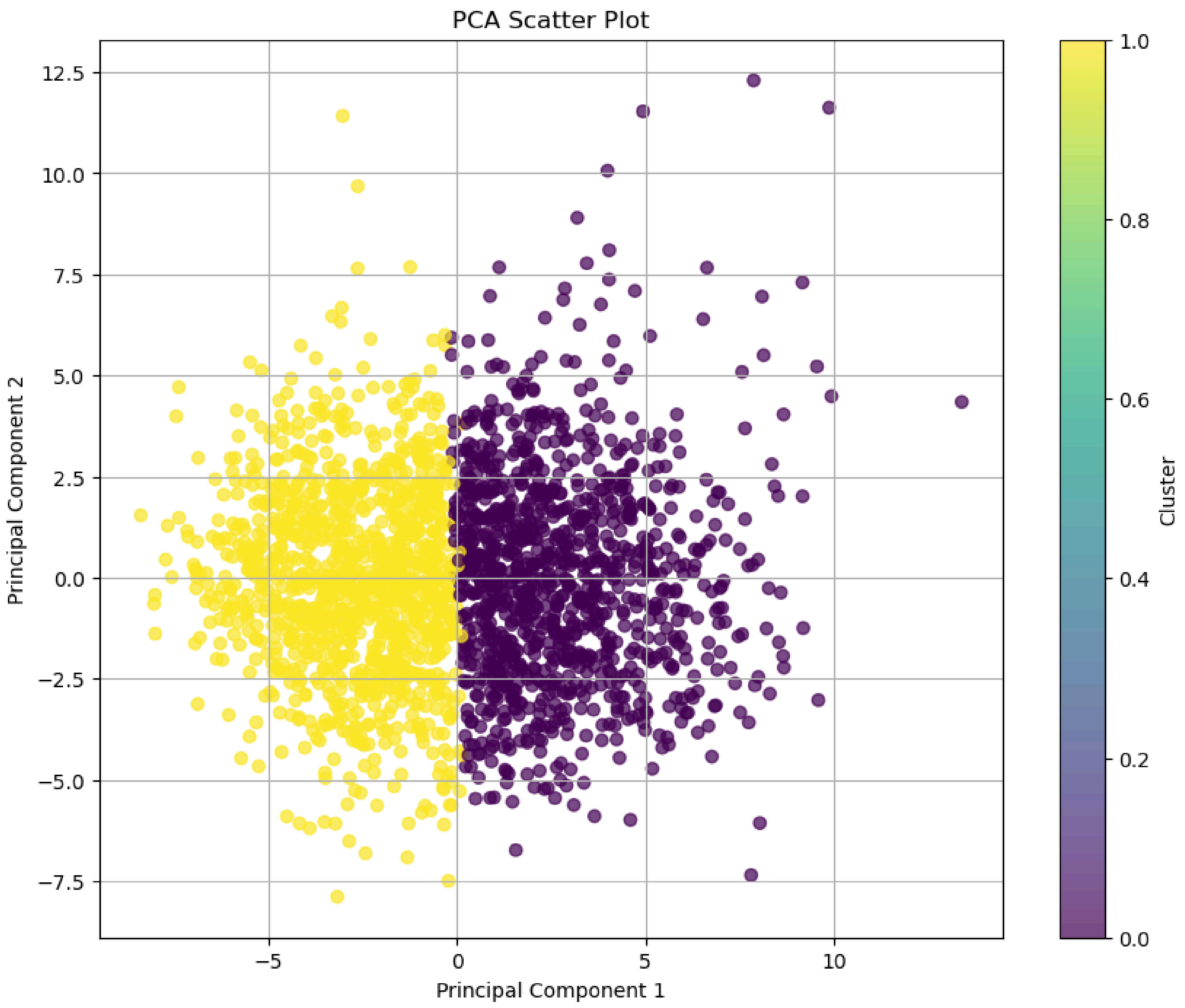
In [12]:

```python
X = scaled_df.values

# Creates the PCA model with 2 components
pca = PCA(n_components=2)

# Fits the PCA model and transforms the data
pca_transformed = pca.fit_transform(X)

# Converts the transformed data to a DataFrame
pca_df = pd.DataFrame(pca_transformed, columns=['PC1', 'PC2'])

# Adds the cluster labels
if 'Cluster' in scaled_df.columns:
    pca_df['Cluster'] = scaled_df['Cluster']

# Prints the explained variance ratio for each principal component
print("Explained variance ratio:", pca.explained_variance_ratio_)

# Prints the first few rows of the PCA-transformed DataFrame
print(pca_df.head())
```

```
Explained variance ratio: [0.11356381 0.06366391]
        PC1       PC2  Cluster
0 -1.474243 -2.319633        1
1 -1.485195 -4.870825        1
2  1.684187 -0.422816        0
3 -1.954833  2.100639        1
4  0.366860  0.173137        0
```

7.

In [14]:
```
# Creates a scatter plot
plt.figure(figsize=(10, 8))
scatter = plt.scatter(pca_df['PC1'], pca_df['PC2'], c=pca_df['Cluster'], cmap='viridis', alpha=0.7)
plt.colorbar(scatter, label='Cluster')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Scatter Plot')
plt.grid(True)
plt.show()
```


PCA Scatter Plot
Principal Component 1
Principal Component 2
Cluster
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
1.0
0.8
0.6
0.4
0.2
0.0

8.

The PCA Scatter plot sh