Week 4 Assignment

September 24, 2023

[1]:
```
# Ignores warnings
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

[3]:
```
df = pd.read_csv('als_data.csv')
```

[4]:
```
df.head()
```

[4]:

	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]
```

Remove any data that is not relevant to the patient's ALS condition.

[5]:
```
# Display Columns in df
x = df.columns
for x in x:
    print(x)
```

```
ID
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
SubjectID
trunk_max
trunk_median
trunk_min
trunk_range
Urine.Ph_max
Urine.Ph_median
Urine.Ph_min
```

[6]:
```
# Drop columns not needed for analysis
df = df.drop(columns=['ID', 'SubjectID'])
```

Apply a standard scalar to the data.

[7]:
```
# Create StandardScaler
scaler = StandardScaler()
```

[8]:
```
# Fit the scaler to the data and transform the DataFrame
df_scaled = scaler.fit_transform(df)
```

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

[9]:
```
# Create a range of number of clusters to try
num_clusters_range = range(2, 11)

# Initialize an empty list to store silhouette scores
silhouette_scores = []

# Iterate over the number of clusters
for num_clusters in num_clusters_range:
    # Create an instance of KMeans with the current number of clusters
    kmeans = KMeans(n_clusters=num_clusters, random_state=42)
```

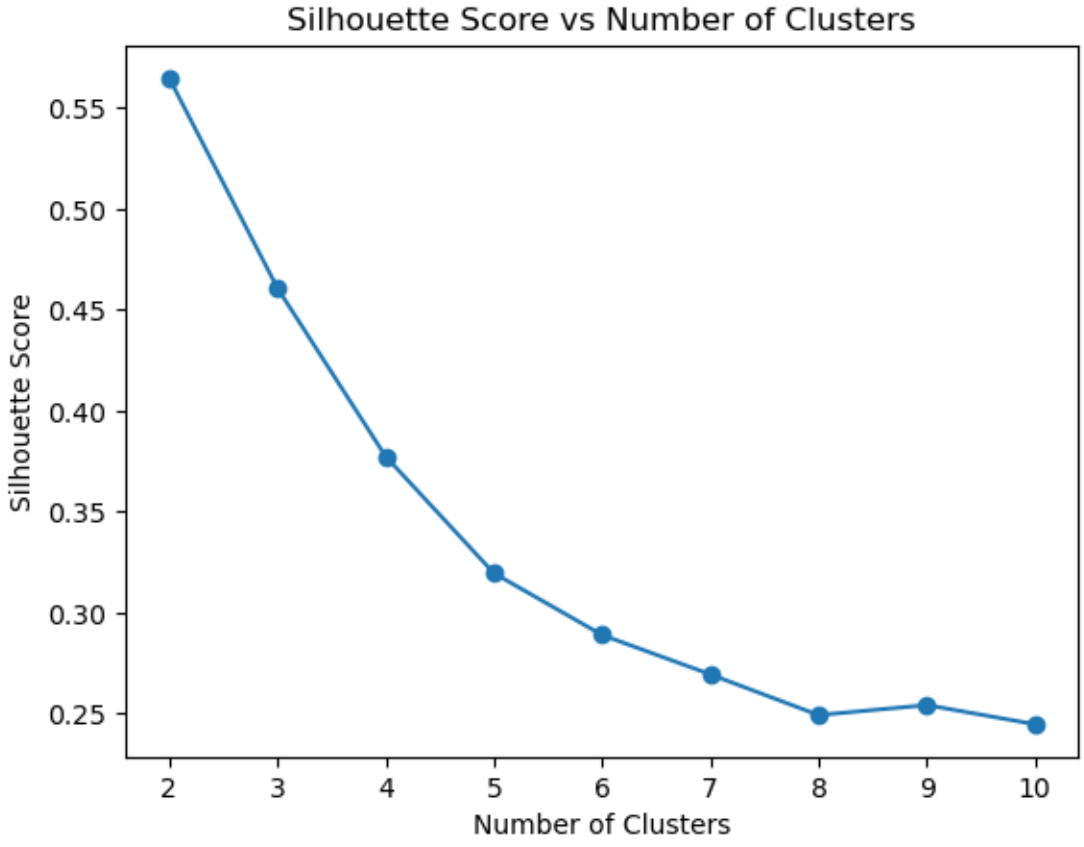
```
    # Fit the KMeans model to the data
    kmeans.fit(df)

    # Get the cluster labels for each sample
    cluster_labels = kmeans.labels_

    # Calculate the silhouette score for the current number of clusters
    silhouette_avg = silhouette_score(df, cluster_labels)

    # Append the silhouette score to the list
    silhouette_scores.append(silhouette_avg)

# Plot the silhouette scores versus the number of clusters
plt.plot(num_clusters_range, silhouette_scores, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score vs Number of Clusters')
plt.show()
```



Use the plot created in (3) to choose on optimal number of clusters for K-means.

Justify your choice.

The silhouette score "is a measure of how well a data point belongs to its assigned cluster and how far it is from the neighboring clusters" (Cloud, 2023). When observing these scores, they can rang between 1 and -1. The goal is to have a sihouette score closer towards 1 for a given number of clsuter. In our graph above, the highest silhouette score is aroun 0.56 with a number of clusters of 2. This is our ideal number of clusters since it produces the highest silhouette score. https://saturncloud.io/blog/how-to-use-silhouette-score-in-kmeans-clustering-from-scikitlearn-library/#:~:text=The%20silhouette%20score%20is%20a%20useful%20metric%20for%20evaluating%20the,is%20fr

Fit a K-means model to the data with the optimal number of clusters chosen in part (4).

[10]:
```
# Create KMeans with 2 clusters
kmeans = KMeans(n_clusters=2, random_state=42)
```

[11]:
```
# Fit the KMeans model to the data
k_means_model = kmeans.fit(df)
```

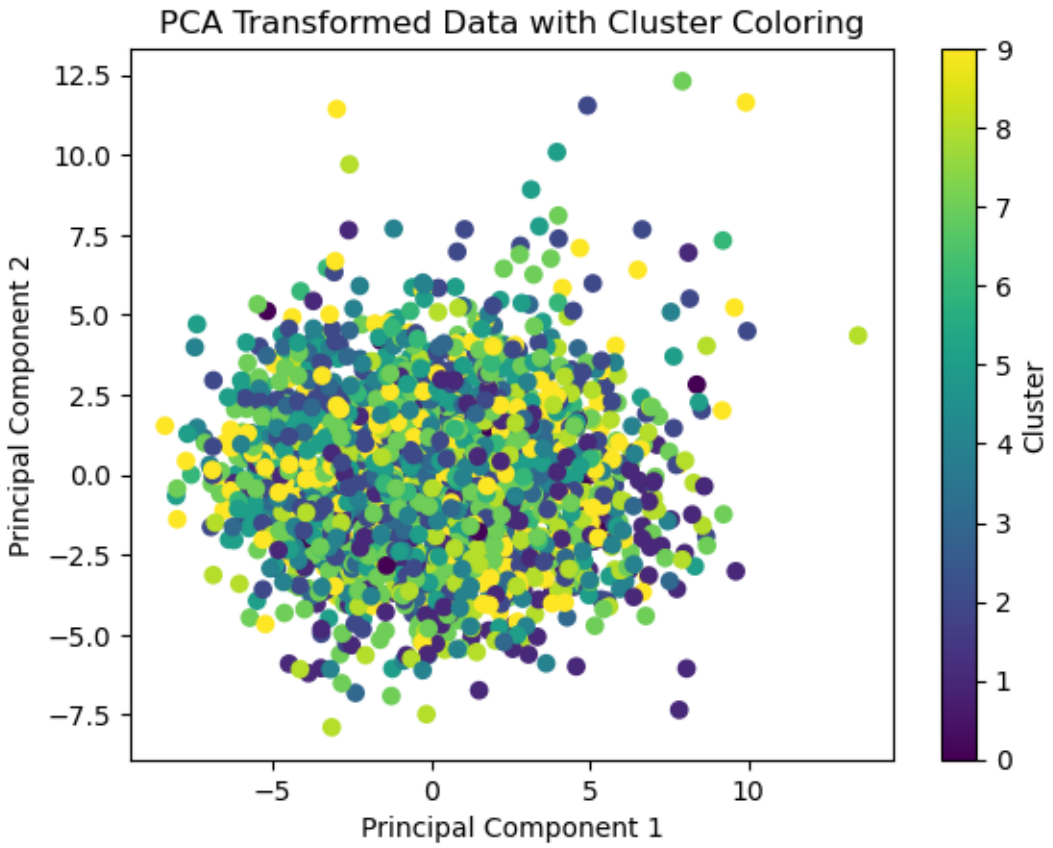
Fit a PCA transformation with two features to the scaled data.

[12]:
```
# Create an instance of PCA with 2 components
pca = PCA(n_components=2)
```

[13]:
```
# Fit and transform the PCA model to the scaled data
df_pca = pca.fit_transform(df_scaled)
```

Make a scatterplot the PCA transformed data coloring each point by its cluster value.

[14]:
```
# Create a scatter plot of the PCA transformed data
plt.scatter(df_pca[:, 0], df_pca[:, 1], c=cluster_labels)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Transformed Data with Cluster Coloring')
plt.colorbar(label='Cluster')
plt.show()
```



Summary and Findings

In this analysis, we removed the fields not needed, we discovered the number of optimal clusters using silhouette scores as well as created a scatterplot of the PCA transformed data with each cluster. The fields that were not needed were ID fields included in the data. These do not contain information needed for the model. The silhouette score analysis determined that the optimal number of clusters was 2. The goal of the PCA plot is to display samples of the two components based on their similarity. When plotting the data with the number of clusters used in step 2, there are no apparent clusters forming between the components in the PCA. This indicates no similarity between any points in the data.