Week 4

- Joshua Burden
- DSC630 Predictive Analytics
- Bellevue University
- Andrew Hua

- 09/25/2022

In [ ]:
```
# Use for import/install modules that don't exist
```

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.style as style
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.metrics import silhouette_samples, silhouette_score
from yellowbrick.cluster import SilhouetteVisualizer
from sklearn.preprocessing import StandardScaler
from kneed import KneeLocator
from sklearn.decomposition import PCA
import seaborn as sns
```

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

df.head()
```

Out[ ]:

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

5 rows × 101 columns

In [ ]:
```
print(df)
```

	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	
...	...	...	...	...	...	...	
2218	2419	33	50.0	49.0	45.0	0.008772	
2219	2420	61	47.0	45.0	42.0	0.009074	
2220	2421	47	46.0	44.0	41.0	0.012111	
2221	2422	37	49.0	44.0	39.0	0.017857	
2222	2424	48	48.0	45.0	40.0	0.018476	

	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	
...	...	...	...	...	
2218	-0.239501	35	32.5	30	
2219	-0.388711	31	26.0	17	
2220	-0.108631	26	23.0	20	
2221	-0.855880	34	29.5	21	
2222	-2.050562	37	34.0	11	

	...	Sodium_min	Sodium_range	SubjectID	trunk_max	trunk_median	\
0	...	143.0	0.017422	533	8	7.0	
1	...	136.0	0.010453	649	8	7.0	
2	...	140.0	0.008929	1234	5	0.0	
3	...	138.0	0.012469	2492	5	5.0	
4	...	138.0	0.008292	2956	6	4.0	
...	...	...	...	...	...	...	
2218	...	136.0	0.014035	997136	7	5.0	
2219	...	141.0	0.009074	998047	5	4.0	
2220	...	135.0	0.013123	998773	5	4.0	
2221	...	136.0	0.007143	998908	8	4.5	
2222	...	137.0	0.018476	999482	8	8.0	

	trunk_min	trunk_range	Urine.Ph_max	Urine.Ph_median	Urine.Ph_min
0	7	0.002646	6.00	6.0	6.0
1	5	0.005386	7.00	5.0	5.0
2	0	0.008929	6.00	5.0	5.0
3	3	0.004988	7.00	6.0	5.0
4	1	0.008489	6.00	5.0	5.0
...	...	...	...	...	...
2218	5	0.003643	7.00	6.0	5.0
2219	3	0.003630	7.41	5.5	5.0
2220	4	0.001825	9.00	6.0	5.0
2221	2	0.010714	6.00	5.0	5.0
2222	1	0.016129	5.00	5.0	5.0

[2223 rows x 101 columns]

In [ ]:
```
df.shape
```

Out[ ]:
```
(2223, 101)
```

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

In [ ]:
```
df.drop(["SubjectID", "ID"], axis=1, inplace=True)
```

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

Out[ ]:

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

5 rows × 99 columns

Apply a standard scalar to the data.

In [ ]:
```
scaler = StandardScaler()
X = scaler.fit_transform(df)
print(X)
```

```
[[ 0.91713698  3.08941722 -1.30078105 ... -0.88037551  0.46305355
   1.86853157]
 [-0.57487867 -0.62201561 -1.11240084 ...  0.1926645  -1.13720768
  -0.41915124]
 [-1.45253494  0.92441474  1.14816173 ... -0.88037551 -1.13720768
  -0.41915124]
 ...
 [-0.6626443  -0.31272954  0.01788044 ...  2.33874452  0.46305355
  -0.41915124]
 [-1.54030057  0.61512867  0.01788044 ... -0.88037551 -1.13720768
  -0.41915124]
 [-0.57487867  0.3058426   0.39464087 ... -1.95341552 -1.13720768
  -0.41915124]]
```

In [ ]:
```
np.mean(X),np.std(X)
```

Out[ ]:
```
(-6.941510843405787e-19, 1.0)
```

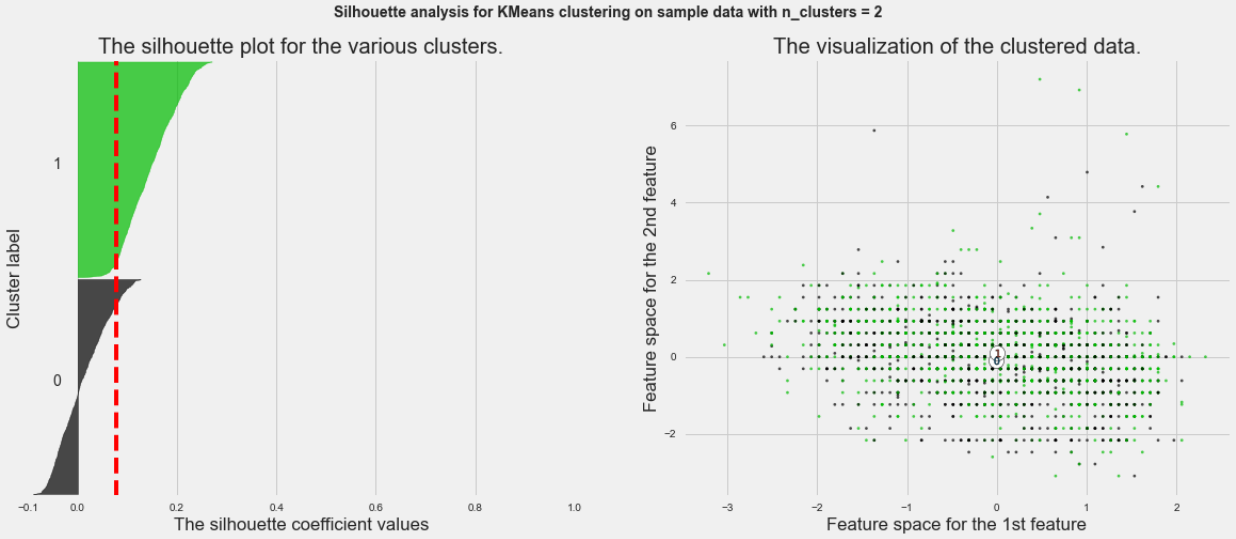
Create a plot of the cluster silhouette score versus the number of clusters in a K-means cluster.

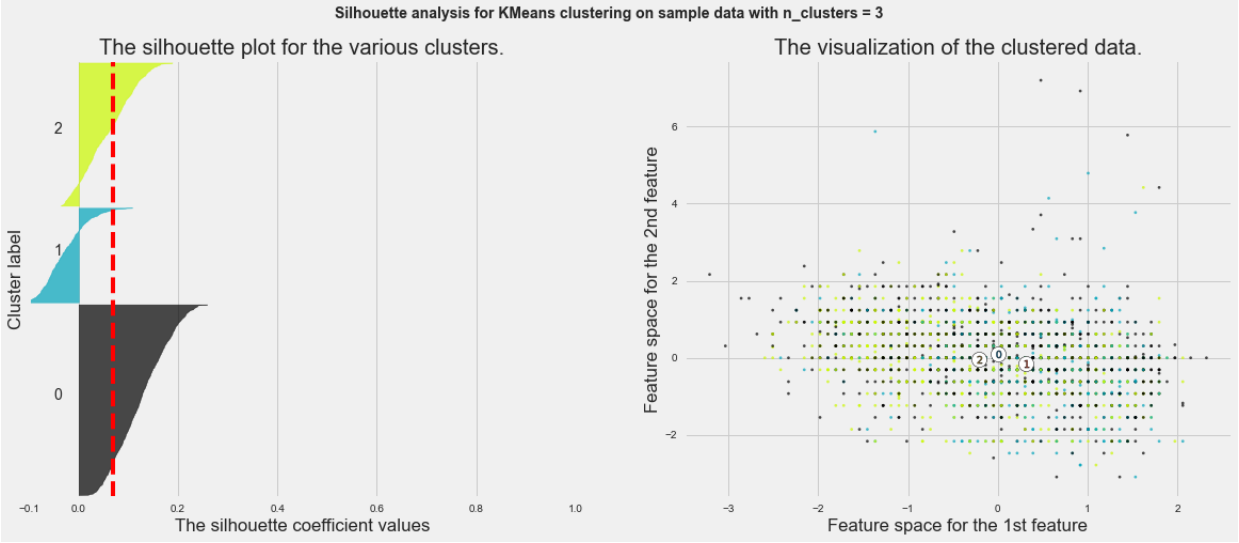
In [ ]:
```
# https://towardsdatascience.com/silhouette-method-better-than-elbow-method-to-find-op

range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10]
silhouette_avg_n_clusters = []
```

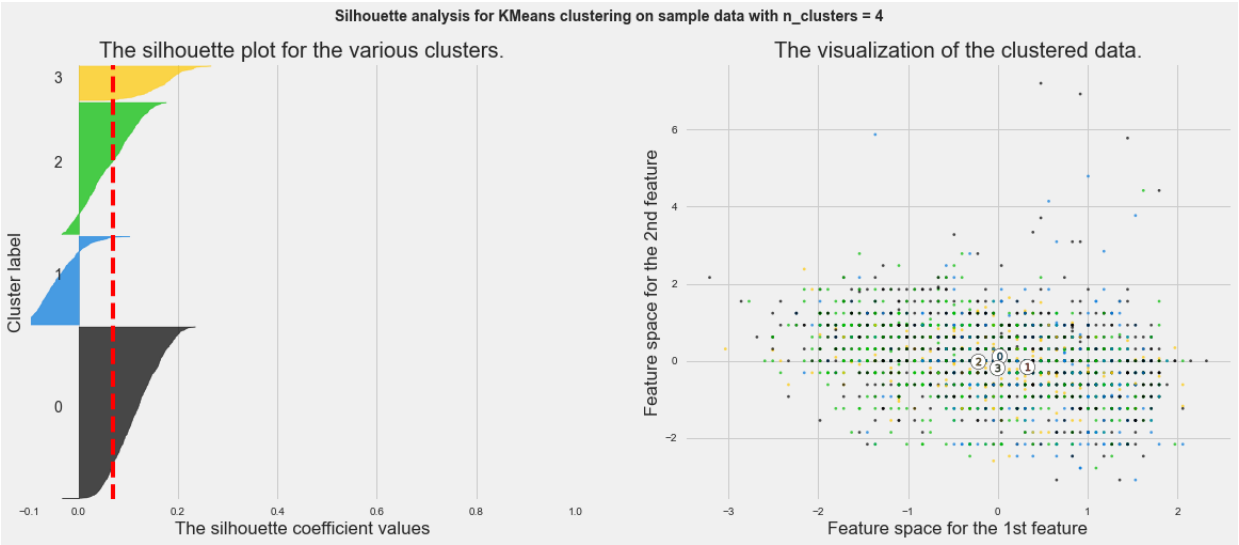
```
for n_clusters in range_n_clusters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # The 1st subplot is the silhouette plot
    # The silhouette coefficient can range from -1, 1 but in this example all
    # lie within [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
    # The (n_clusters+1)*10 is for inserting blank space between silhouette
    # plots of individual clusters, to demarcate them clearly.
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    # Initialize the clusterer with n_clusters value and a random generator
    # seed of 10 for reproducibility.
    clusterer = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = clusterer.fit_predict(X)

    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the formed
    # clusters
    silhouette_avg = silhouette_score(X, cluster_labels)
    print("For n_clusters =", n_clusters,
          "The average silhouette_score is :", silhouette_avg)

    silhouette_avg_n_clusters.append(silhouette_avg)
    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):
        # Aggregate the silhouette scores for samples belonging to
        # cluster i, and sort them
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)

        # Label the silhouette plots with their cluster numbers at the middle
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        # Compute the new y_lower for next plot
        y_lower = y_upper + 10  # 10 for the 0 samples

    ax1.set_title("The silhouette plot for the various clusters.")
    ax1.set_xlabel("The silhouette coefficient values")
    ax1.set_ylabel("Cluster label")

    # The vertical line for average silhouette score of all the values
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])  # Clear the yaxis labels / ticks
```

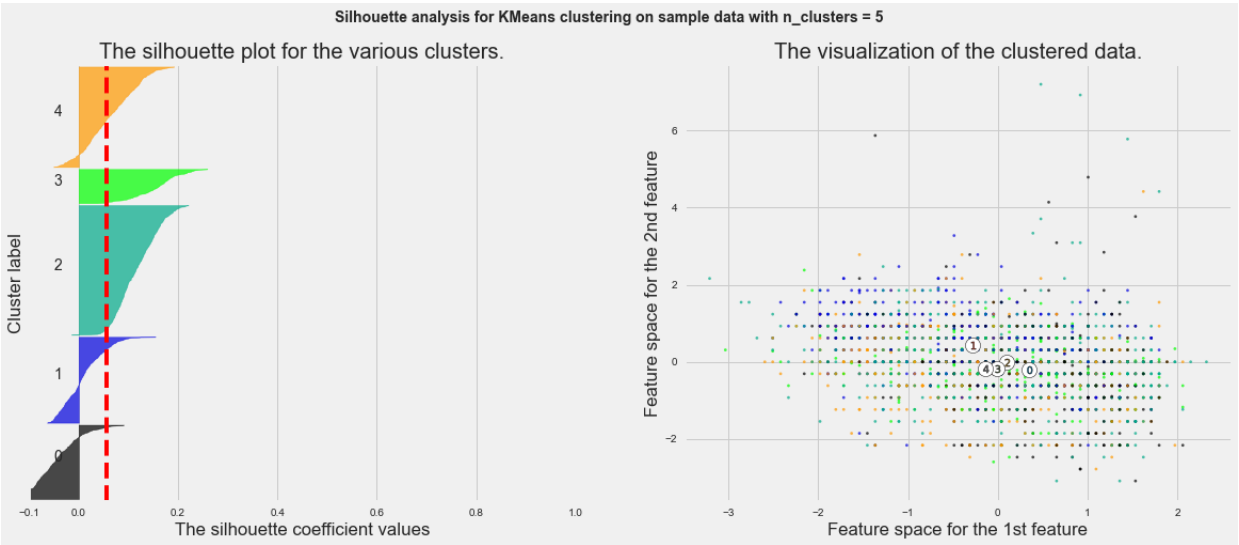
```
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    # 2nd Plot showing the actual clusters formed
    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax2.scatter(X[:, 0], X[:, 1], marker='.', s=30, lw=0, alpha=0.7,
                c=colors, edgecolor='k')

    # Labeling the clusters
    centers = clusterer.cluster_centers_
    # Draw white circles at cluster centers
    ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
                c="white", alpha=1, s=200, edgecolor='k')

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                    s=50, edgecolor='k')

    ax2.set_title("The visualization of the clustered data.")
    ax2.set_xlabel("Feature space for the 1st feature")
    ax2.set_ylabel("Feature space for the 2nd feature")

    plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "
                  "with n_clusters = %d" % n_clusters),
                 fontsize=14, fontweight='bold')

plt.show()


style.use("fivethirtyeight")
plt.plot(range_n_clusters, silhouette_avg_n_clusters)
plt.xlabel("Number of Clusters (k)")
plt.ylabel("silhouette score")
plt.show()
```

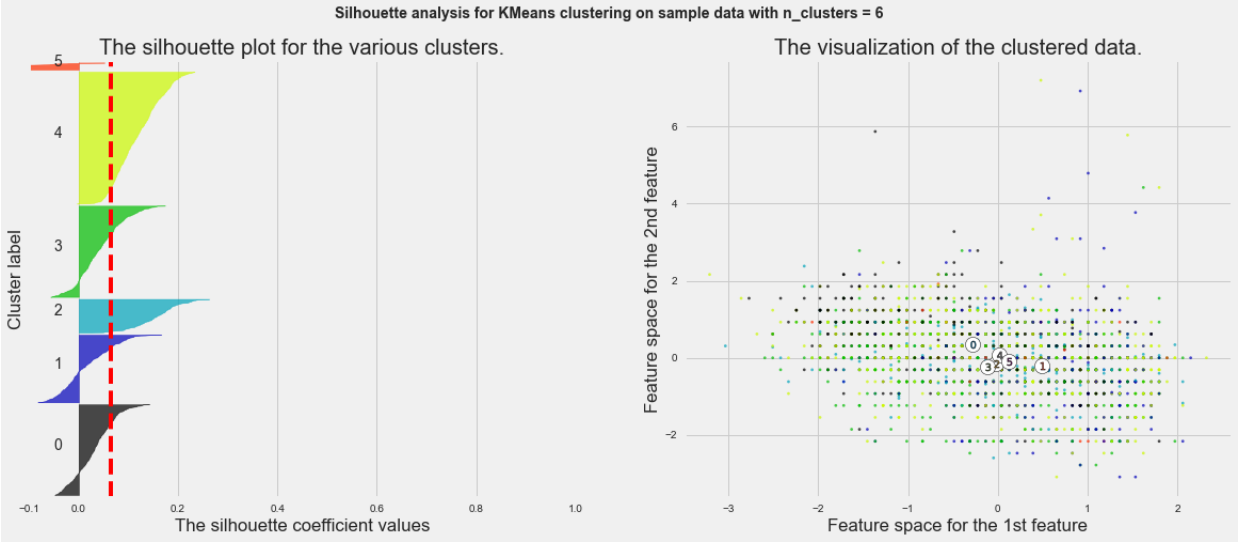
```
For n_clusters = 2 The average silhouette_score is : 0.07878005888570402
For n_clusters = 3 The average silhouette_score is : 0.0687707291658565
For n_clusters = 4 The average silhouette_score is : 0.06973816142698218
For n_clusters = 5 The average silhouette_score is : 0.05697679932842005
For n_clusters = 6 The average silhouette_score is : 0.06477886829610223
For n_clusters = 7 The average silhouette_score is : 0.05187647631845004
For n_clusters = 8 The average silhouette_score is : 0.04954004349267962
For n_clusters = 9 The average silhouette_score is : 0.0439371958229717
For n_clusters = 10 The average silhouette_score is : 0.04612161184531545
```

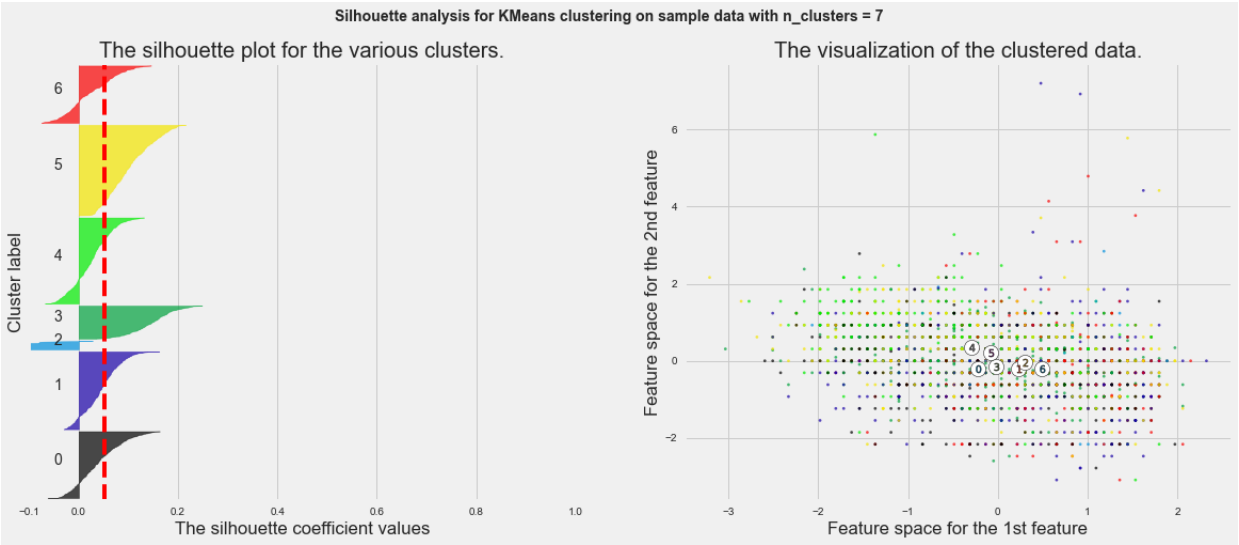


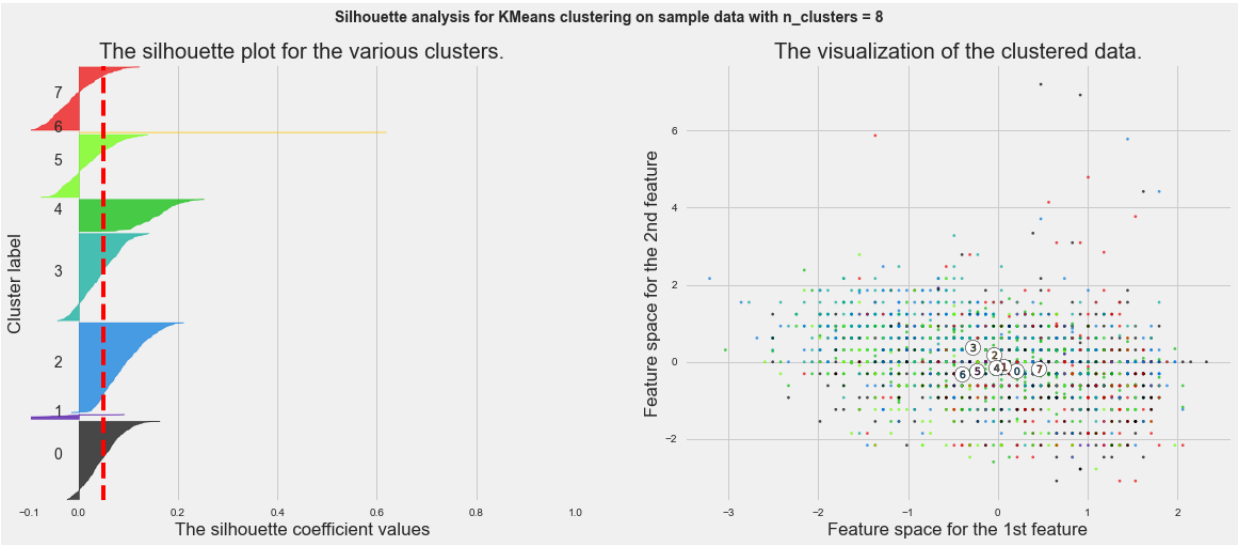

Silhouette analysis for KMeans clustering on sample data with n_clusters = 3
The silhouette plot for the various clusters.
Cluster label
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature

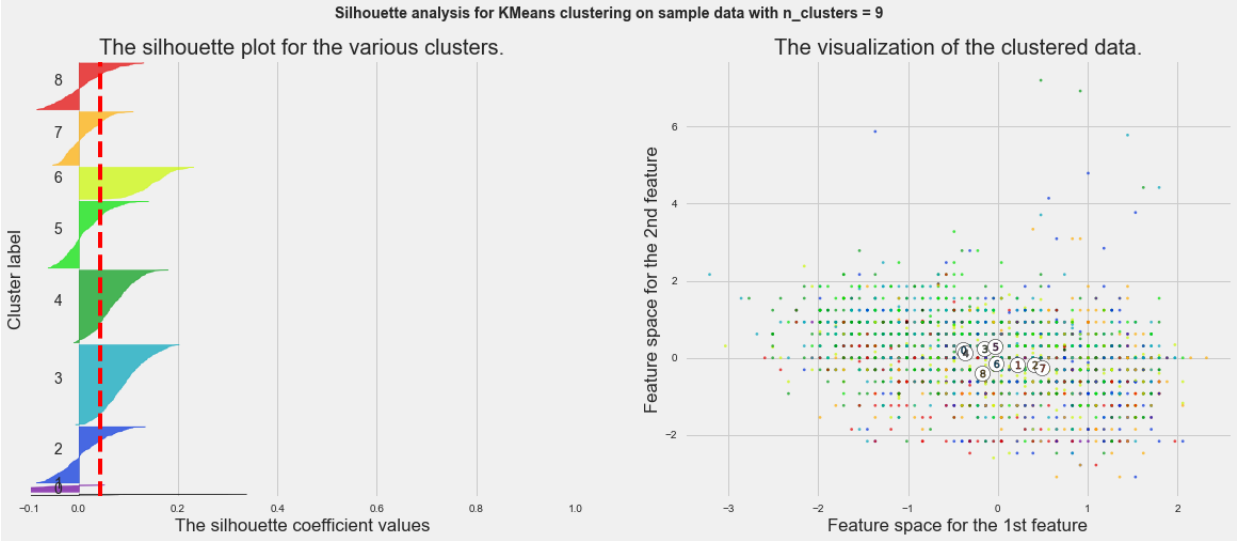

Silhouette analysis for KMeans clustering on sample data with n_clusters = 4
The silhouette plot for the various clusters.
Cluster label
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature

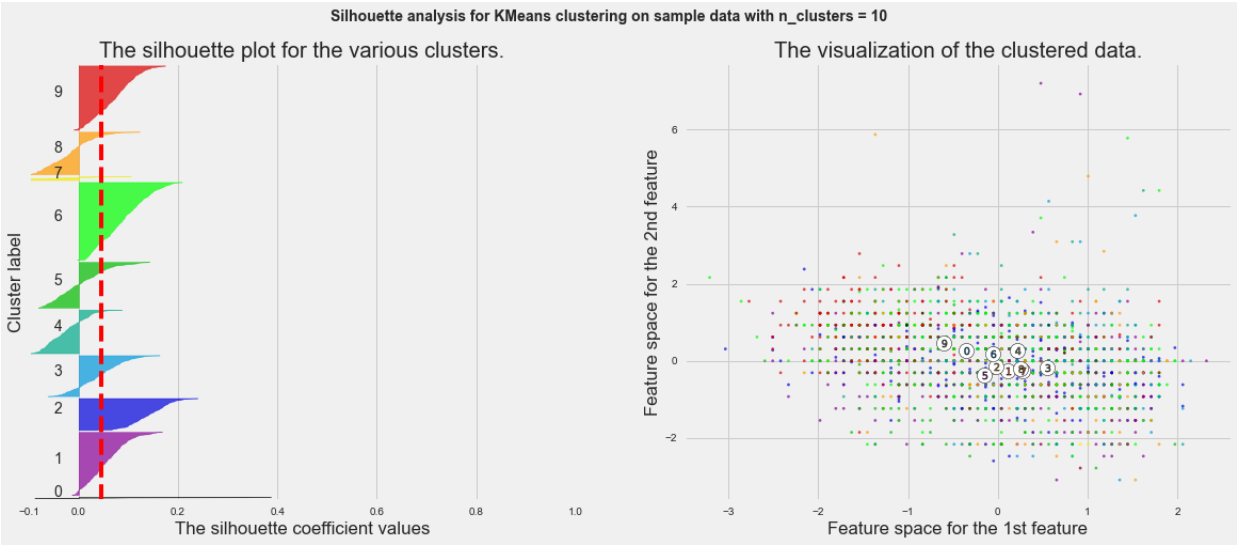

Silhouette analysis for KMeans clustering on sample data with n_clusters = 5
The silhouette plot for the various clusters.
Cluster label
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 6
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 7
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 8
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature





Use the plot created in (3) to choose on optimal

number of clusters for K-means. Justify your choice.

- ### The optimal number of clusters is 2 as it has the highest silhouette score (0.07878005888570402) and is validated with the above graph.

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

In [ ]:
```
k_means_model = KMeans(n_clusters=2, random_state=7)
k_means_model.fit(X)
```

Out[ ]:
```
KMeans(n_clusters=2, random_state=7)
```

In [ ]:
```
label = k_means_model.fit_predict(X)
```

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

In [ ]:
```
# Create a pca with 2 components
pca = PCA(n_components=2)

# Fit the PCA
als_pca = pca.fit_transform(X)
# Create a dataframe
als_pca_df = pd.DataFrame(data=als_pca, columns = ['PC 1', 'PC 2'])
# Add the cluster label to the dataframe
als_pca_df['Cluster'] = label
# View the dataframe
als_pca_df
```

Out[ ]:

	PC 1	PC 2	Cluster
0	-1.426717	-2.318425	0
1	-1.440247	-4.871998	0
2	1.617842	-0.428095	1
3	-1.919958	2.095952	0
4	0.297689	0.167373	1
...	...	...	...
2218	-4.477605	1.200488	0
2219	-0.398966	-1.877314	0
2220	-0.432883	4.245829	0
2221	-0.330783	3.317258	0
2222	1.468006	0.583028	1

2223 rows × 3 columns

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

In [ ]:
```
# Filter out clusters
als_cluster0 = als_pca_df.loc[als_pca_df['Cluster'] == 0]
als_cluster1 = als_pca_df.loc[als_pca_df['Cluster'] == 1]
als_cluster0
```

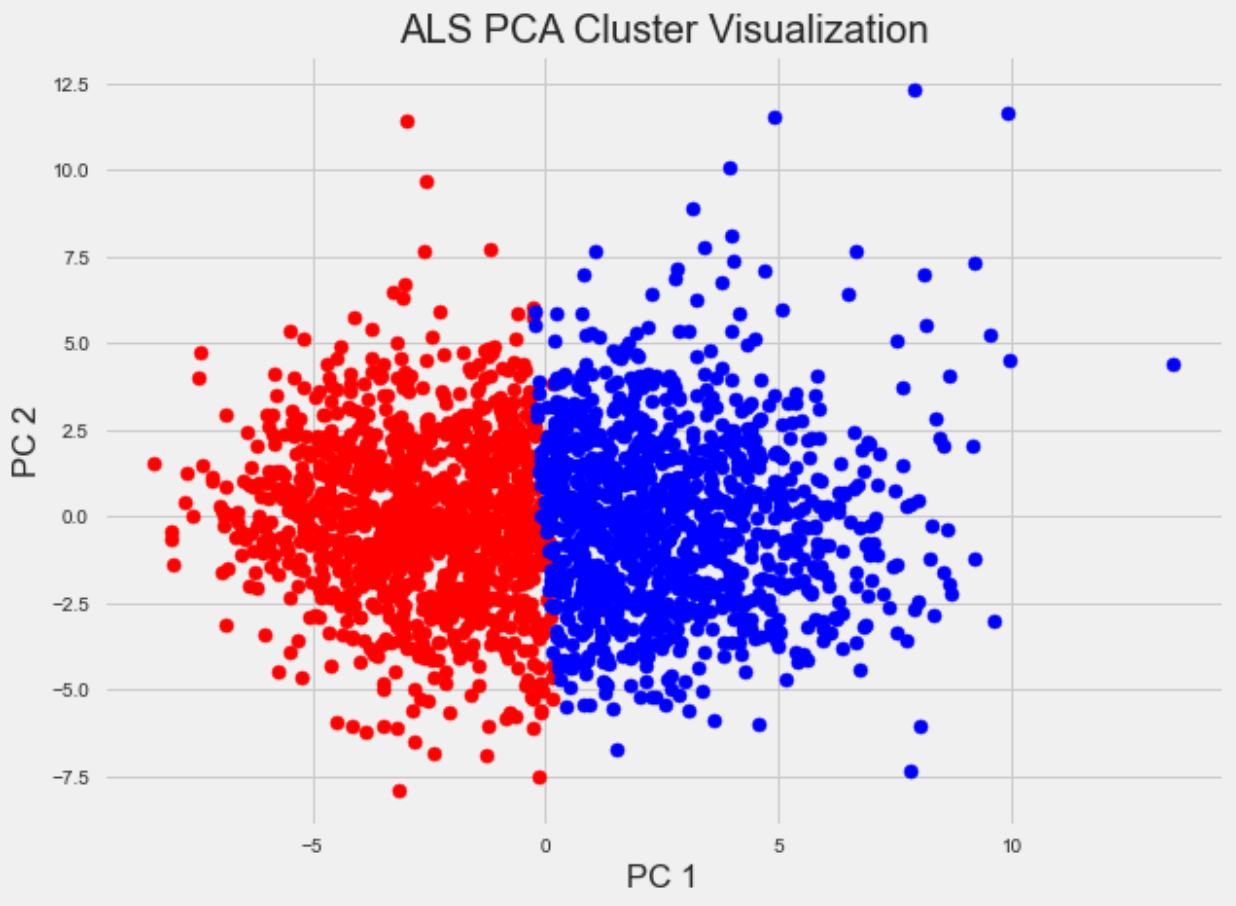
Out[ ]:

	PC 1	PC 2	Cluster
0	-1.426750	-2.318491	0
1	-1.440244	-4.871415	0
3	-1.919988	2.096074	0
5	-4.528940	-1.298245	0
8	-2.534987	2.298744	0
...	...	...	...
2217	-2.865582	2.049019	0
2218	-4.477612	1.200724	0
2219	-0.398950	-1.876134	0
2220	-0.432879	4.244189	0
2221	-0.330774	3.316133	0

1115 rows × 3 columns

In [ ]:
```
# Create a scatter plot of Clusters vs Silhouette Score
```

```
fig=plt.figure()
ax=fig.add_axes([0, 0, 1, 1])
ax.scatter(als_cluster0['PC 1'] , als_cluster0['PC 2'] , color = 'red')
ax.scatter(als_cluster1['PC 1'] , als_cluster1['PC 2'] , color = 'blue')
ax.set_xlabel('PC 1')
ax.set_ylabel('PC 2')
ax.set_title('ALS PCA Cluster Visualization')
plt.show()
```



Summarize your results and make a conclusion.

- ### Applying kmeans clustering and PCA we can see after that two different and distinct groups are identified. If we use the visualization from above we can see that there is not a datapoints that is out of place from the groupings.