Predictive Analytics: Assignment 4.2

Joshua Greenert

DSC630-T301 Predictive Analytics

1/2/2023

In [79]:
```python
# Import the required libaries.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Pull in the data to begin preparation.
df_als = pd.read_csv('als_data.csv')
df_als.head(5)
```

Out[79]:

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

5 rows × 101 columns

In [80]:
```python
# Start by creating a correlation matrix.
corr_matrix = df_als.corr()

# Make a heatmap from data so it's easier to see the correlations.
corr_matrix.style.background_gradient(cmap='coolwarm')
```

Out[80]:

	ID	Age_mean	Albumin_max	Albumin_median	Albumin_min	Albumin_
ID	1.000000	0.007008	-0.014141	-0.011243	-0.009921	-0.0
Age_mean	0.007008	1.000000	-0.276195	-0.349024	-0.297121	0.0
Albumin_max	-0.014141	-0.276195	1.000000	0.780141	0.596662	0.2
Albumin_median	-0.011243	-0.349024	0.780141	1.000000	0.761269	-0.0
Albumin_min	-0.009921	-0.297121	0.596662	0.761269	1.000000	-0.3
Albumin_range	-0.002910	0.053197	0.223350	-0.091822	-0.369015	1.0
ALSFRS_slope	0.018619	-0.015301	0.037438	0.059234	0.112154	-0.2
ALSFRS_Total_max	0.007969	0.049054	0.113349	0.153280	0.158924	-0.1
ALSFRS_Total_median	0.001242	0.057733	0.090439	0.128122	0.172940	-0.1
ALSFRS_Total_min	0.023207	0.041025	0.058077	0.099099	0.188007	-0.1
ALSFRS_Total_range	-0.007867	0.038163	-0.072609	-0.099966	-0.151358	0.2
ALT.SGPT._max	0.001653	-0.130050	0.091963	0.101377	0.034264	0.0
ALT.SGPT._median	0.004731	-0.189788	0.137417	0.187458	0.162226	-0.0

ALT.SGPT._min	0.011468	-0.142516	0.085657	0.147607	0.173052	-0.0
ALT.SGPT._range	0.014890	-0.058298	0.008734	0.000620	-0.033994	0.1
AST.SGOT._max	-0.014878	-0.030284	0.060325	0.048049	0.007431	0.0
AST.SGOT._median	0.011718	-0.024973	0.096926	0.124172	0.109144	-0.0
AST.SGOT._min	0.020625	-0.002730	0.032387	0.080313	0.114516	-0.0
AST.SGOT._range	0.000518	-0.010642	0.004965	-0.014476	-0.036657	0.1
Bicarbonate_max	-0.019779	0.165844	0.125698	0.107236	0.025970	-0.0
Bicarbonate_median	0.009830	0.191592	0.066279	0.046939	0.007117	0.0
Bicarbonate_min	-0.009338	0.169390	-0.028314	-0.040693	-0.029353	0.0
Bicarbonate_range	0.002626	0.062786	0.008282	-0.007052	-0.016733	0.1
Blood.Urea.Nitrogen..BUN._max	0.002857	0.218799	0.064980	-0.002630	-0.033055	0.0
Blood.Urea.Nitrogen..BUN._median	0.019551	0.286131	-0.027990	-0.054270	-0.066050	0.0
Blood.Urea.Nitrogen..BUN._min	-0.006245	0.183931	-0.059244	-0.028821	0.026912	-0.0
Blood.Urea.Nitrogen..BUN._range	0.024652	0.142506	0.010039	-0.073901	-0.083650	0.1
bp_diastolic_max	-0.032920	0.005481	0.084957	0.114683	0.075712	-0.0
bp_diastolic_median	-0.029801	0.013110	0.084578	0.156109	0.137294	-0.0
bp_diastolic_min	-0.033371	0.017555	0.071161	0.136431	0.161045	-0.0
bp_diastolic_range	-0.008405	0.046004	-0.074635	-0.086270	-0.095153	0.1
bp_systolic_max	-0.009617	0.327372	-0.009723	-0.029938	-0.052392	0.0
bp_systolic_median	-0.006169	0.317989	-0.005592	-0.001736	-0.005762	0.0
bp_systolic_min	-0.018884	0.258714	0.045570	0.049809	0.049491	0.0
bp_systolic_range	0.003696	0.196450	-0.123900	-0.144217	-0.138054	0.1
Calcium_max	-0.016143	0.008545	0.155105	0.150450	0.131857	-0.0
Calcium_median	0.042489	-0.010566	0.286746	0.336714	0.293908	0.0
Calcium_min	-0.000898	0.006702	0.094279	0.128773	0.169644	-0.0
Calcium_range	0.000854	0.024142	-0.012157	-0.053379	-0.057257	0.1
Chloride_max	0.001489	-0.091505	0.035649	0.037558	0.017007	-0.1
Chloride_median	-0.002559	-0.141916	-0.009322	0.029715	0.031809	-0.1
Chloride_min	0.004836	-0.160119	-0.072399	-0.006120	0.059682	-0.0
Chloride_range	-0.010770	0.101587	-0.025731	-0.082791	-0.068252	0.2
Creatinine_max	-0.006014	0.053231	0.098292	0.110474	0.099011	-0.0
Creatinine_median	-0.000674	0.041418	0.050303	0.090986	0.117553	-0.0
Creatinine_min	0.011775	0.010950	0.009317	0.040177	0.139861	-0.0
Creatinine_range	-0.030886	0.086564	0.026042	0.010782	-0.067427	0.1
Gender_mean	0.005891	-0.168238	0.161496	0.241774	0.225021	-0.0
Glucose_max	-0.017000	0.120487	0.010767	0.009631	-0.012418	-0.0
Glucose_median	-0.005123	0.133152	0.026261	0.026086	0.009595	-0.0

Glucose_min	0.002738	-0.007844	-0.018847	0.056898	0.090714	-0.0
Glucose_range	-0.019123	0.132036	-0.028908	-0.056251	-0.062965	0.1
hands_max	-0.004176	0.173512	-0.001439	-0.007101	0.015838	-0.0
hands_median	-0.006981	0.185773	-0.023482	-0.028485	0.015642	-0.0
hands_min	0.019189	0.179134	-0.033976	-0.028484	0.036182	-0.1
hands_range	-0.014141	-0.013099	-0.041942	-0.055227	-0.071624	0.1
Hematocrit_max	-0.013430	-0.034760	0.091769	0.036835	-0.019991	0.0
Hematocrit_median	-0.013249	-0.041498	0.090150	0.042328	-0.012497	0.0
Hematocrit_min	-0.010771	-0.057266	0.094050	0.054505	0.028419	0.0
Hematocrit_range	-0.010518	0.070984	-0.059026	-0.117044	-0.146325	0.2
Hemoglobin_max	0.019765	-0.181186	0.159079	0.205137	0.189739	-0.0
Hemoglobin_median	0.020427	-0.206245	0.152803	0.238150	0.239787	-0.0
Hemoglobin_min	0.032267	-0.189124	0.102448	0.202851	0.298236	-0.1
Hemoglobin_range	-0.010617	0.056863	-0.044372	-0.107180	-0.146709	0.2
leg_max	0.003905	-0.047906	0.139818	0.179596	0.167852	-0.0
leg_median	0.000075	-0.029621	0.134157	0.166601	0.175830	-0.0
leg_min	0.009718	-0.041440	0.091300	0.130834	0.174243	-0.1
leg_range	0.001153	0.039065	-0.008516	-0.015230	-0.037524	0.1
mouth_max	0.013957	-0.055647	0.050458	0.099309	0.115960	-0.1
mouth_median	-0.000904	-0.054257	0.064704	0.109045	0.144995	-0.1
mouth_min	0.020914	-0.034408	0.062803	0.108634	0.176298	-0.1
mouth_range	-0.002034	0.038895	-0.116434	-0.153355	-0.196108	0.2
onset_delta_mean	-0.011805	-0.039550	-0.003759	0.048405	0.025833	0.0
onset_site_mean	0.006690	-0.090055	0.006648	0.027399	0.054884	-0.0
Platelets_max	-0.011493	0.037074	-0.104745	-0.141115	-0.189825	0.0
Platelets_median	-0.000375	0.002051	-0.115375	-0.137441	-0.104450	0.0
Platelets_min	0.001725	0.006589	-0.109240	-0.122950	-0.076696	0.0
Potassium_max	-0.031897	0.040688	-0.004157	-0.009839	-0.002625	-0.0
Potassium_median	-0.020533	0.144203	-0.004223	0.012496	0.018269	-0.0
Potassium_min	0.000478	0.034911	-0.028865	0.006738	0.071035	-0.0
Potassium_range	-0.023999	0.053625	-0.041060	-0.062979	-0.047411	0.1
pulse_max	-0.033012	-0.077080	0.014157	0.000910	-0.037239	0.0
pulse_median	-0.026571	-0.066583	-0.009456	-0.001371	-0.002951	0.0
pulse_min	-0.023240	-0.033929	-0.002595	-0.001306	0.007479	0.0
pulse_range	-0.026836	0.012406	-0.081420	-0.089154	-0.100420	0.2
respiratory_max	0.031774	-0.071695	0.056793	0.058894	0.061554	-0.0
respiratory_median	0.022024	-0.055059	0.090280	0.104120	0.107924	-0.1
respiratory_min	0.010647	-0.055515	0.081607	0.108920	0.158138	-0.1

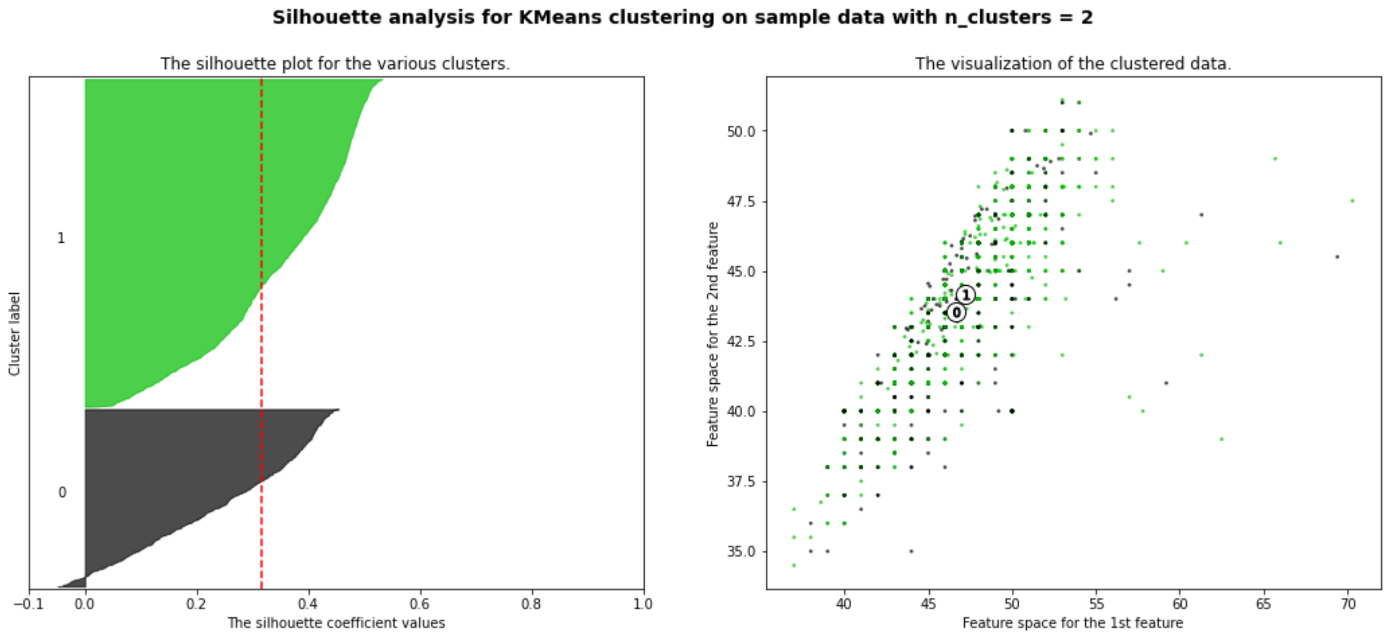
respiratory_range	0.013775	0.064244	-0.127569	-0.156803	-0.161566	0.1
Sodium_max	-0.012175	0.028854	0.051798	0.049189	0.014051	-0.0
Sodium_median	0.006870	0.005432	0.013677	0.007380	0.046761	0.0
Sodium_min	0.018663	-0.039712	-0.040877	-0.017818	0.074240	0.0
Sodium_range	-0.006561	0.080449	-0.048029	-0.069380	-0.086457	0.1
SubjectID	0.999917	0.007046	-0.014803	-0.011740	-0.010308	-0.0
trunk_max	0.011977	0.071736	0.080261	0.097823	0.095974	-0.0
trunk_median	0.000203	0.089325	0.057346	0.084183	0.113639	-0.1
trunk_min	0.015915	0.065092	0.042903	0.072317	0.134409	-0.1
trunk_range	-0.000649	0.036559	-0.064183	-0.088370	-0.114685	0.2
Urine.Ph_max	-0.022561	-0.001532	0.075413	0.056894	0.034555	-0.0
Urine.Ph_median	-0.016045	0.002561	-0.046885	-0.042497	-0.001877	0.0
Urine.Ph_min	0.002042	-0.008615	-0.138430	-0.118506	-0.049783	0.0

In [81]:
```
# Remove any data that is not relevant to the patient's ALS condition.
# Items to drop: items with < 60%
df_als_new = df_als.drop(["ID", "Age_mean", "Albumin_range", "Bicarbonate_range", "Blood
                          "bp_diastolic_range","bp_systolic_range", "Calcium_max", "Calc
                          "Creatinine_range", "Glucose_min", "onset_delta_mean", "pulse_
                          "respiratory_median", "respiratory_min", "respiratory_range",
                          "Urine.Ph_median", "Urine.Ph_min"], axis=1)
```

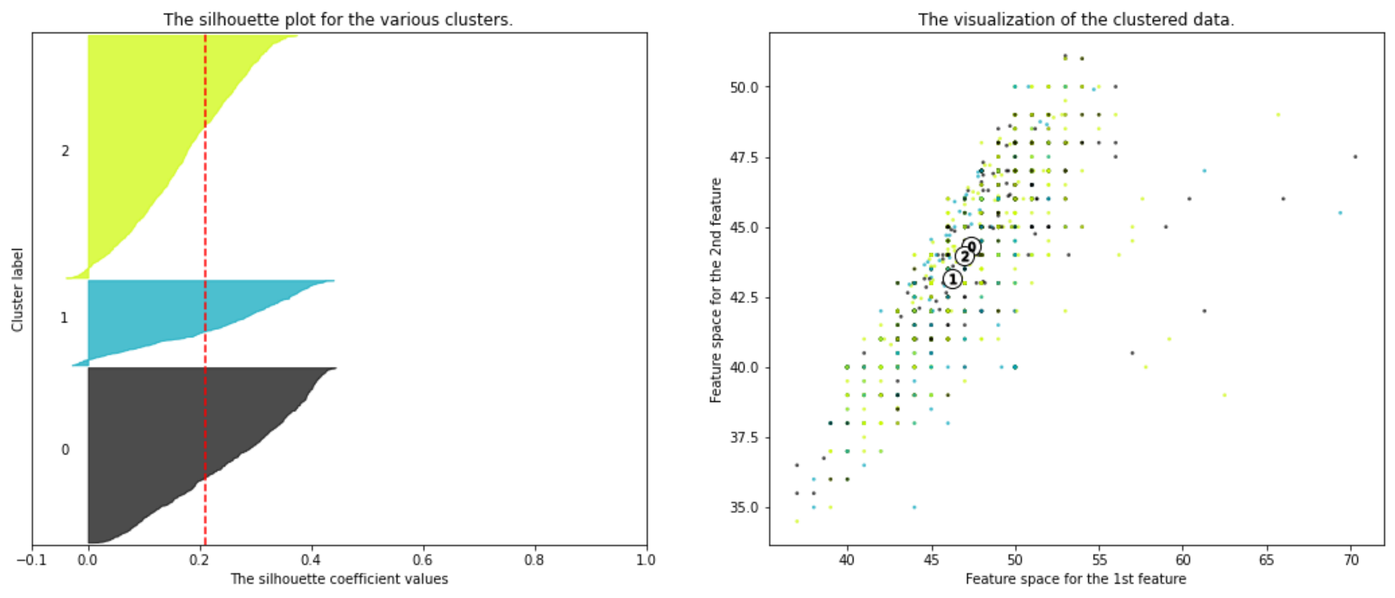
In [82]:
```
# Apply a standard scalar to the data.
from sklearn.preprocessing import MinMaxScaler

# define min max scaler
scaler = MinMaxScaler()

# transform data
scaled = scaler.fit_transform(df_als_new)
```

In [83]:
```
# Create a plot of the cluster silhouette score versus the number of clusters in a K-mea
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.cm as cm

range_n_clusters = [2, 3, 4, 5, 6]

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(df_als_new) + (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=10)
    cluster_labels = clusterer.fit_predict(df_als_new)

    # 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(df_als_new, cluster_labels)
    print(
        "For n_clusters =",
        n_clusters,
        "The average silhouette_score is :",
        silhouette_avg,
    )

    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(df_als_new, 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(
        df_als_new.iloc[:, 0], df_als_new.iloc[:, 1], marker=".", s=30, lw=0, alpha=0.7,
    )

    # 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()
```

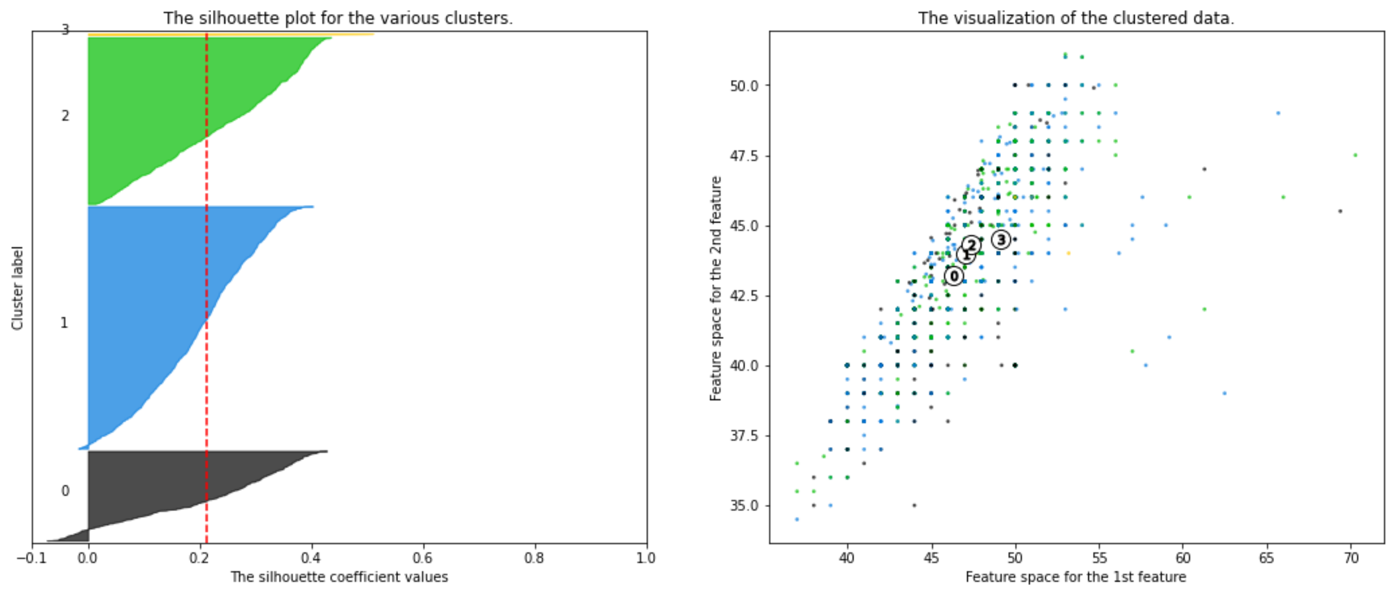
```
For n_clusters = 2 The average silhouette_score is : 0.3151676795036065
For n_clusters = 3 The average silhouette_score is : 0.210571030781278
For n_clusters = 4 The average silhouette_score is : 0.2137759446325614
For n_clusters = 5 The average silhouette_score is : 0.16901065903122586
For n_clusters = 6 The average silhouette_score is : 0.17222830018452961
```



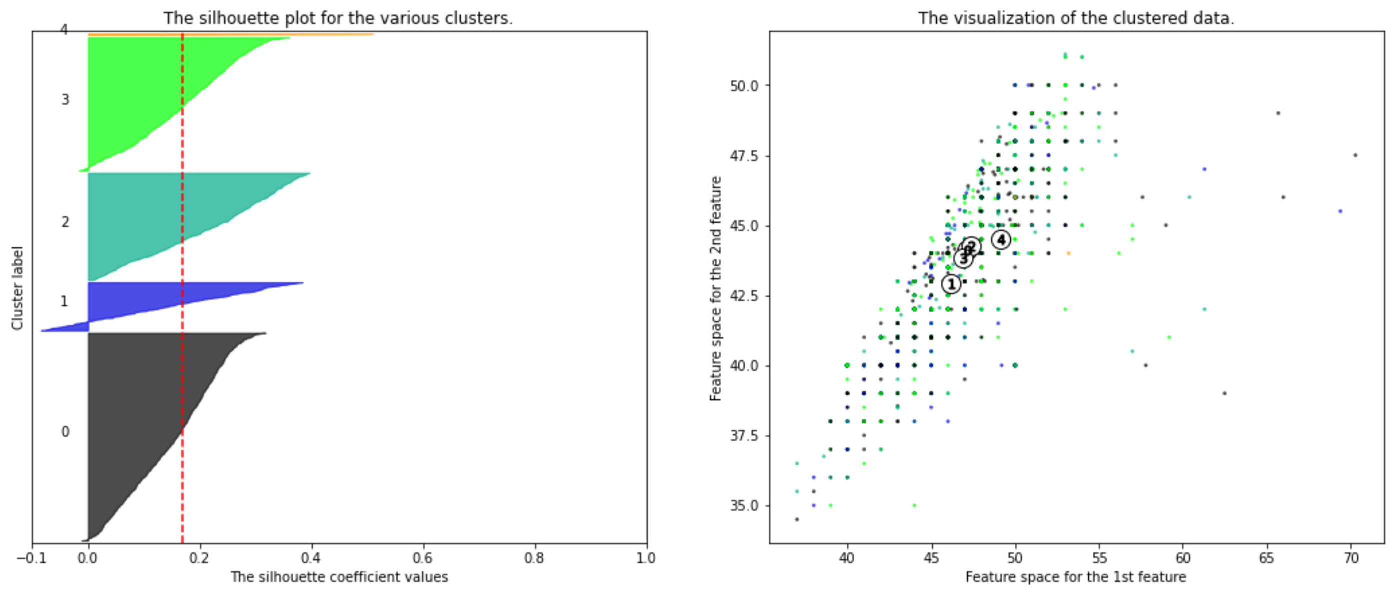
Silhouette analysis for KMeans clustering on sample data with n_clusters = 3

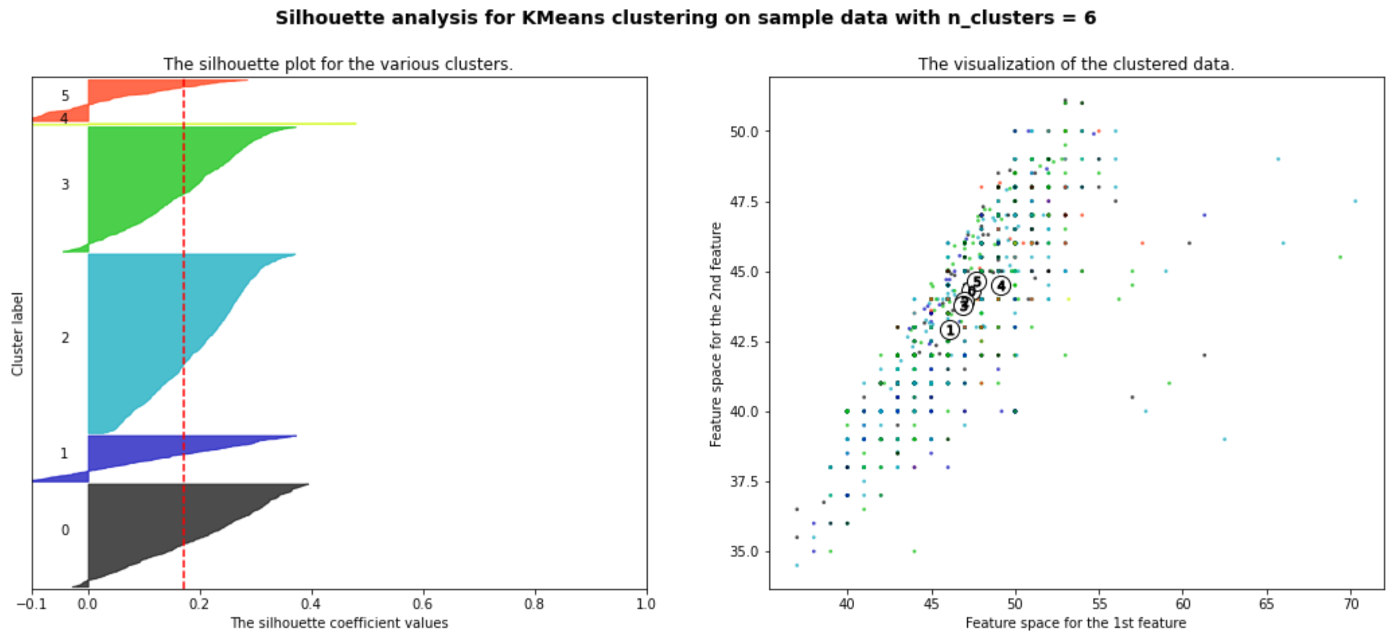


Silhouette analysis for KMeans clustering on sample data with n_clusters = 4



Silhouette analysis for KMeans clustering on sample data with n_clusters = 5



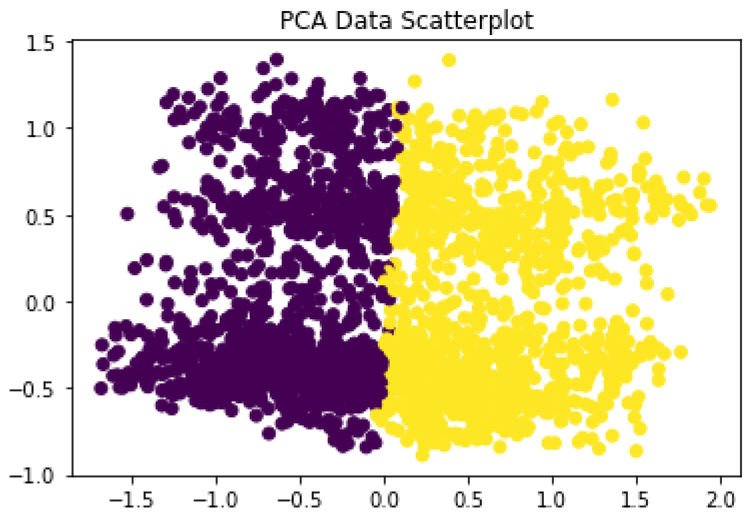


```
In [84]: # Use the plot created in (3) to choose on optimal number of clusters for K-means. Justi
         n_clusters = 2
```

From the silhouette analysis, we can determine that the silhouette_score continues to reduce closer to 0 as the number of clusters increases from 2. Even at 2, the clusters provide a silhouette score of 31.52% where a typical optimal score would be around 60%. Since the numbers are continuously dwindling, it makes sense to cap the number of clusters to reduce the impact on the overall score.

```
In [118…: # Fit a PCA transformation with two features to the scaled data.
          from sklearn.decomposition import PCA

          pca_two = PCA(n_components = 2)
          X_pca = pca_two.fit_transform(scaled)

          # Fit a K-means model to the data with the optimal number of clusters chosen in part (4)
          kmeans = KMeans(n_clusters=3, random_state=10)
          y_pred = clusterer.fit_predict(scaled)
```

```
In [120…: # Make a scatterplot the PCA transformed data coloring each point by its cluster value.
          # plot the data
          plt.scatter(X_pca[:,0], X_pca[:,1], c=y_pred)
          plt.title('PCA Data Scatterplot')
          plt.show()
```



In [123…
```python
# Show the variance between the two features within the PCA.
explained_variance = pca_two.explained_variance_ratio_
explained_variance
```

Out[123]:
```
array([0.27234005, 0.14949741])
```

In [126…
```python
from scipy.stats import ttest_ind

# Split the data into two groups based on the cluster labels
group1 = X_pca[y_pred == 0,:]
group2 = X_pca[y_pred == 1,:]

# Perform a t-test to compare the means of the two groups
t, p = ttest_ind(group1, group2, axis=0)

# Get the number of features
num_features = X_pca.shape[1]

# Print the t-statistic and p-value for each feature
for i in range(num_features):
    print(f"Feature {i}: t = {t[i]:.3f}, p = {p[i]:.3f}")
```

```
Feature 0: t = -70.631, p = 0.000
Feature 1: t = 3.618, p = 0.000
```

Summarize your results and make a conclusion.

With the assistance of some additional testing, we can see that the t-values for the first group is large which may indicate a large difference between the means. This was confirmed with the scatterplot that was performed prior. However, both p-values result in 0.000 which indicate a statistical signficance from the difference of the two means. This may mean more, but additional testing would be required to find the meaning and insight that may be present.