Silhouette-Coefficient

October 22, 2024

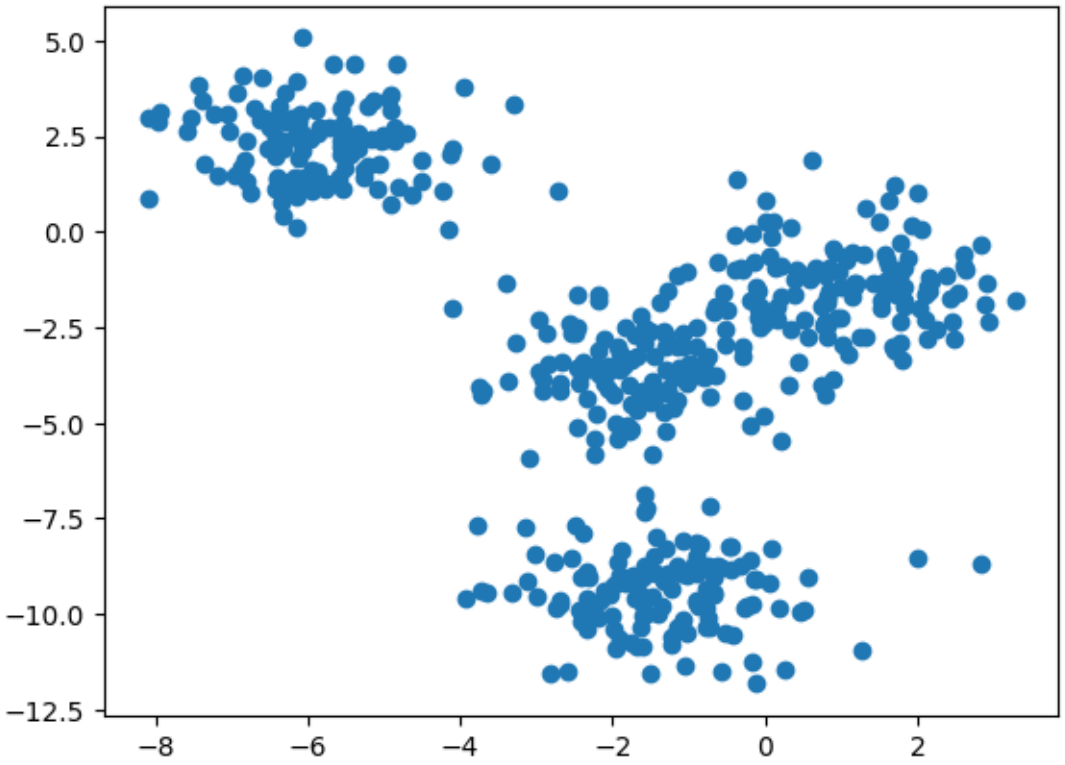
1 Computing Silhouette Coefficient to evaluate Clustering results

- Reference: https://medium.com/@MrBam44/how-to-evaluate-the-performance-of-clustering-algorithms-3ba29cad8c03

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import warnings
warnings.filterwarnings('ignore')

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples , silhouette_score
from sklearn.datasets import make_blobs
```

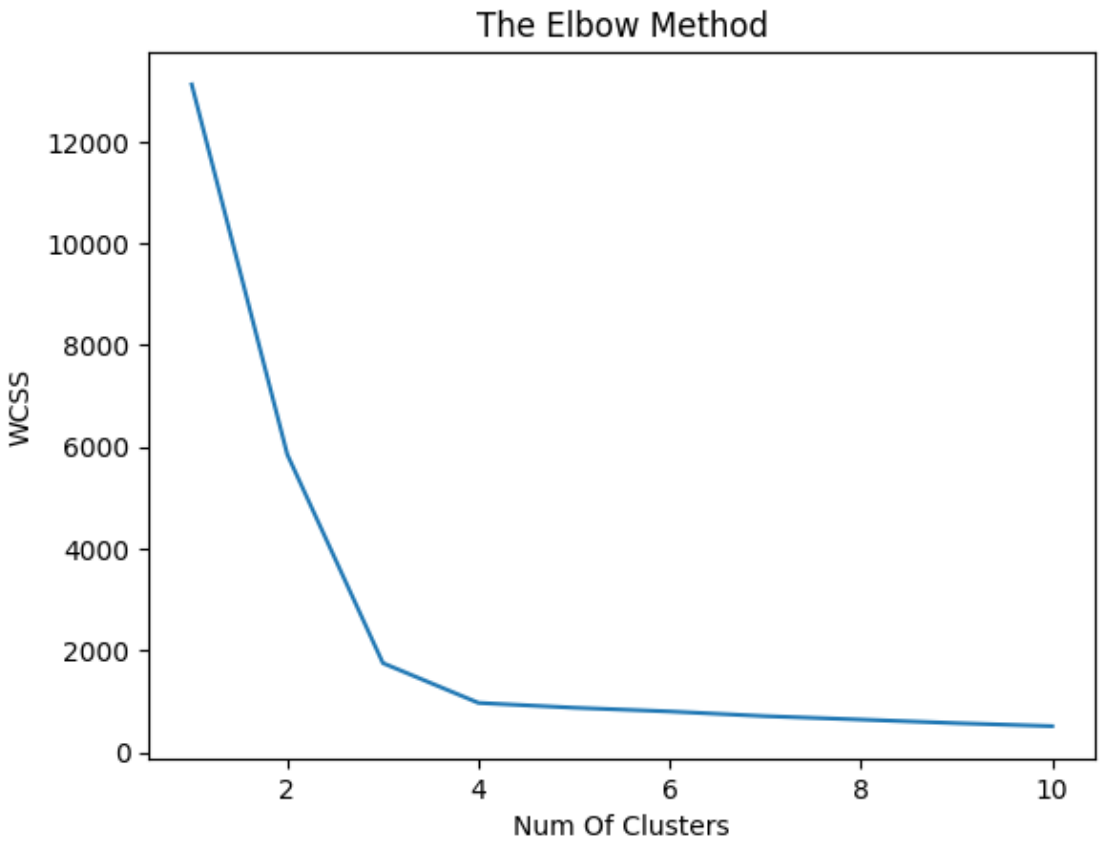
[2]:
```
# Generating the sample data from make_blobs
X, y = make_blobs(n_samples=500,cluster_std=1,centers=4,
                  shuffle=True,center_box=(-10.0,10.0),␣
 ↪n_features=2,random_state=2)
range_n_clusters = [2, 3, 4, 5, 6]
```

[3]:
```
plt.scatter(X[:,0],X[:,1])
plt.show()
X.shape
```

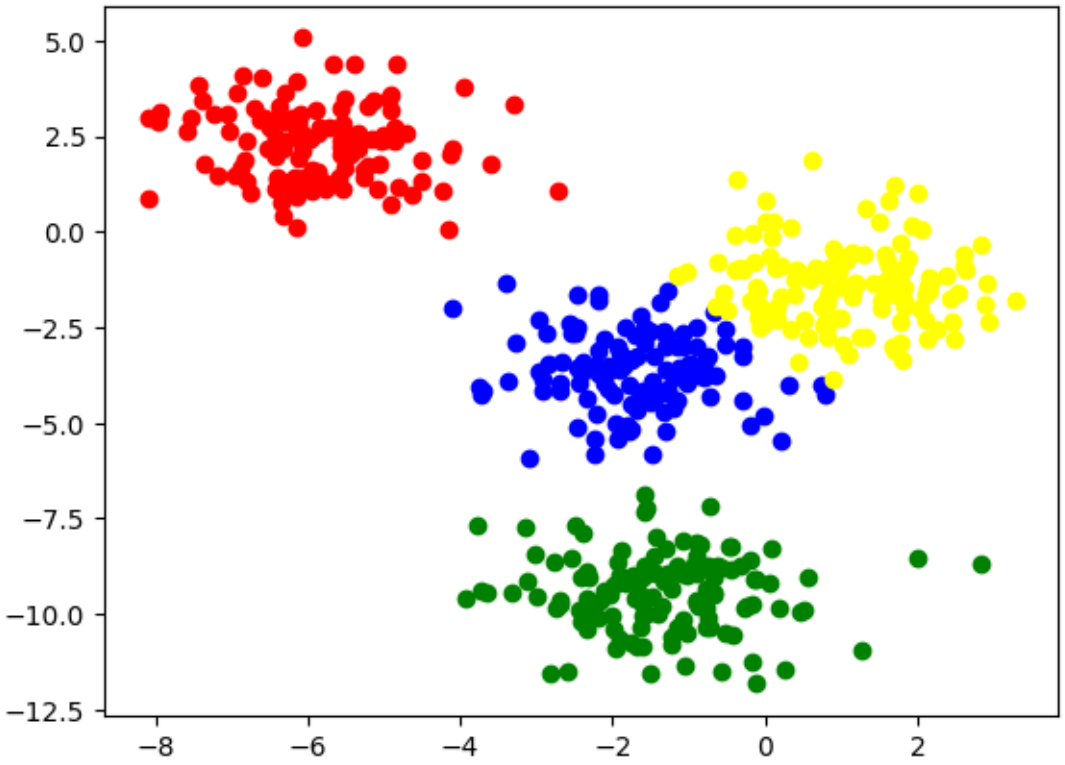


[3]: (500, 2)

[4]:
```
WCSS =  [ ] #within-cluster sum of squares
for i in range(1, 11):
    km = KMeans(n_clusters= i, init='k-means++', random_state=0)
    km.fit(X)
    WCSS.append(km.inertia_)

plt.plot(range(1,11),WCSS)
plt.title('The Elbow Method')
plt.xlabel('Num Of Clusters')
plt.ylabel('WCSS')
plt.show()
```



[5]:
```
clusterer = KMeans(n_clusters=4, random_state=10)
y_mean = clusterer.fit_predict(X)
plt.scatter(X[y_mean == 0,0],X[y_mean ==0,1], color='red')
plt.scatter(X[y_mean ==1,0], X[y_mean==1,1], color= 'blue')
plt.scatter(X[y_mean ==2,0], X[y_mean==2,1], color= 'green')
plt.scatter(X[y_mean ==3,0], X[y_mean==3,1], color= 'yellow')
plt.show()
```



[6]:
```
print(f'Silhouette Score(n=4): {silhouette_score(X, y_mean)}')
```

```
Silhouette Score(n=4): 0.6250462156493074
```

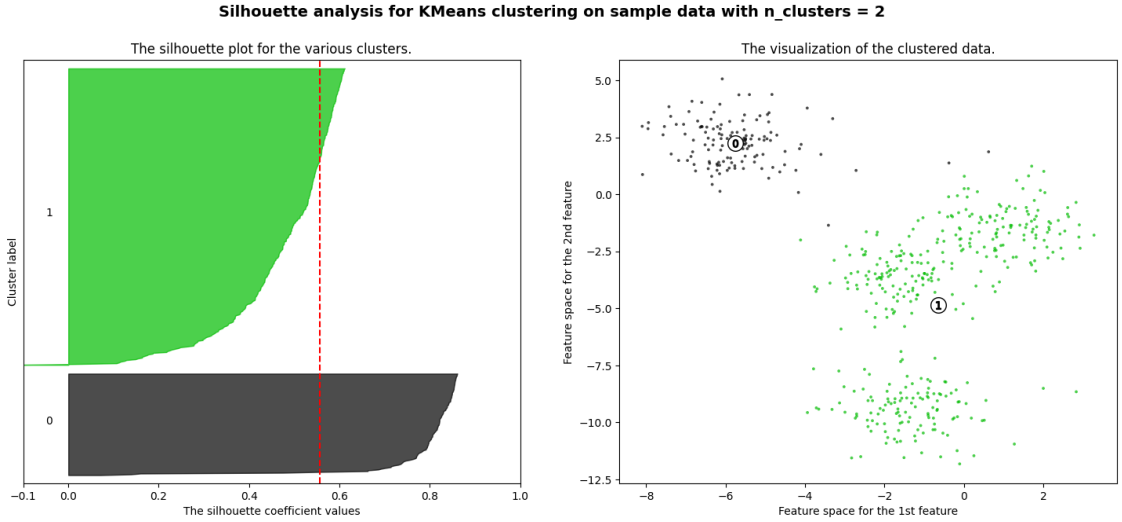
[7]:
```
range_n_clusters = [2, 3, 4, 5, 6]

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(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=10)
```

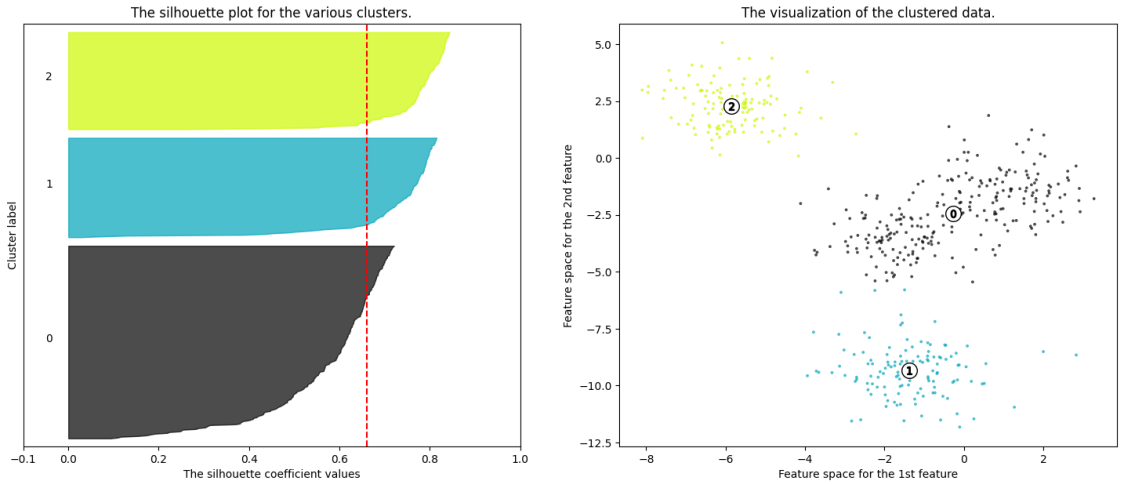
```
    cluster_labels = clusterer.fit_predict(X)

    # 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,
    )

    # 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()
```

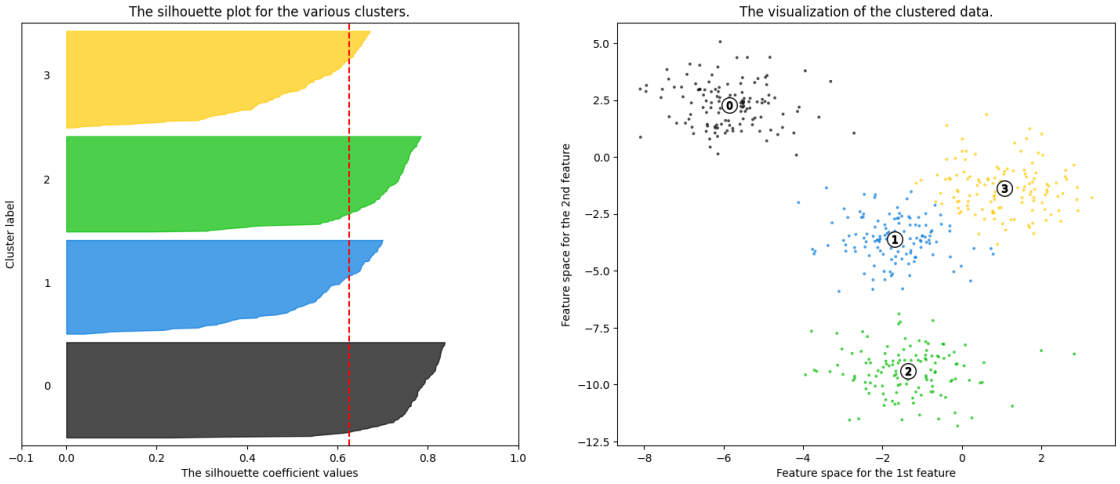
```
For n_clusters = 2 The average silhouette_score is : 0.5563935385599693
For n_clusters = 3 The average silhouette_score is : 0.6602750377214402
For n_clusters = 4 The average silhouette_score is : 0.6250462156493074
```

```
For n_clusters = 5 The average silhouette_score is : 0.5682151387843899
For n_clusters = 6 The average silhouette_score is : 0.45568570243810663
```

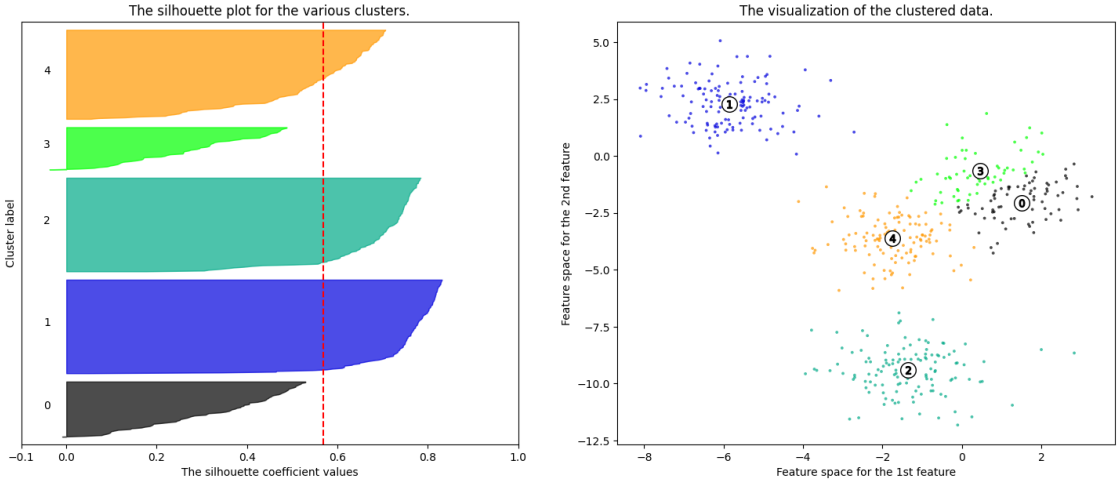


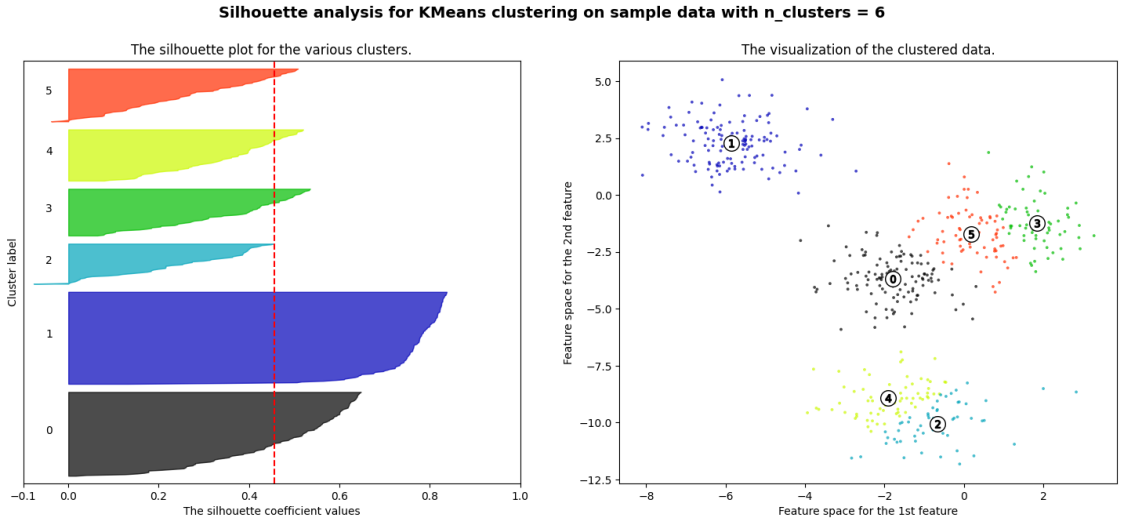


Silhouette analysis for KMeans clustering on sample data with n_clusters = 4



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




Silhouette analysis for KMeans clustering on sample data with n_clusters = 6
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

[ ]:

[ ]: