

Tarefa 2 (mo433)

Imports

```
In [ ]: import pandas as pd
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
from jncvi import bias
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.cm as cm
import numpy as np
from sklearn.metrics.cluster import adjusted_rand_score, fowlkes_mallows_score, adjusted_mutual_info_score

import warnings
warnings.filterwarnings("ignore")
```

Leitura do Dataset

```
In [ ]: df = pd.read_csv('ex2-data.csv', sep = '\\s+', header = None)
print('Shape do dataset: ' + str(df.shape))
df.head()
```

```
Out[ ]: Shape do dataset: (10000, 13)
      0      1      2      3      4      5      6      7      8      9     10     11     12
0 -10.63 -3.91 27.69 2.32 -8.17 -6.15 -2.45 -10.30 -5.62 7.86 5.32 1.35 -4.56
1 12.66 -19.50 4.39 1.53 -3.55 -15.97 -9.16 4.88 5.72 1.88 -3.41 -1.85 2.55
2 4.19 -12.30 -22.25 -6.14 7.47 12.42 6.47 -3.35 -10.22 8.19 5.50 11.08 -2.19
3 -6.38 -18.36 -6.67 -3.42 -3.67 13.01 3.23 -7.45 -3.01 -4.39 -3.40 -1.06 8.25
4 -16.33 0.41 1.56 -10.51 9.37 -3.81 -9.21 -2.16 -6.41 -6.93 -8.05 3.81 2.76
```

1. Kmeans

Nessa etapa iremos rodar o algoritmo do kmeans sobre o dataset (os dados não precisam ser normalizados)

- rodar com k de 2 a 15
- usar `silhouette`
- usar outra medida interna de qualidade (escolhida: Davies-Bouldin score)

Primeiramente iremos iniciar computando o `a` silhouette e plotando os gráfico com k de 2 até 15. Os valores do coeficientes de silhueta que estão próximos a +1 mostram que a amostra está longe dos clusters vizinhos. Já o valor 0 indica que a amostra está no limite de decisão entre dois clusters vizinhos. Por fim, valores negativos indicam que essas amostras podem ter sido atribuídas ao cluster errado.

```
In [ ]: x = df.values
ny =
db_scores = []
for n_clusters in range(2, 16):
    fig, (ax1) = plt.subplots(1, 1)
    fig.set_size_inches(16, 7)

    ax1.set_xlim([-1, 1])
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(x)

    silhouette_avg = silhouette_score(x, cluster_labels)
    print('    n_clusters:',
          'The average silhouette score is :',
          silhouette_avg)

    sample_silhouette_values = silhouette_samples(x, cluster_labels)
    db_scores.append(davies_bouldin_score(x, cluster_labels))
    y_lower = 10
    for i in range(n_clusters):
        ith_cluster_silhouette_values = sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        sith_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_between(
            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(-0.85, 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(silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([]) # Clear the y-axis labels / ticks
    ax1.set_xticks([0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

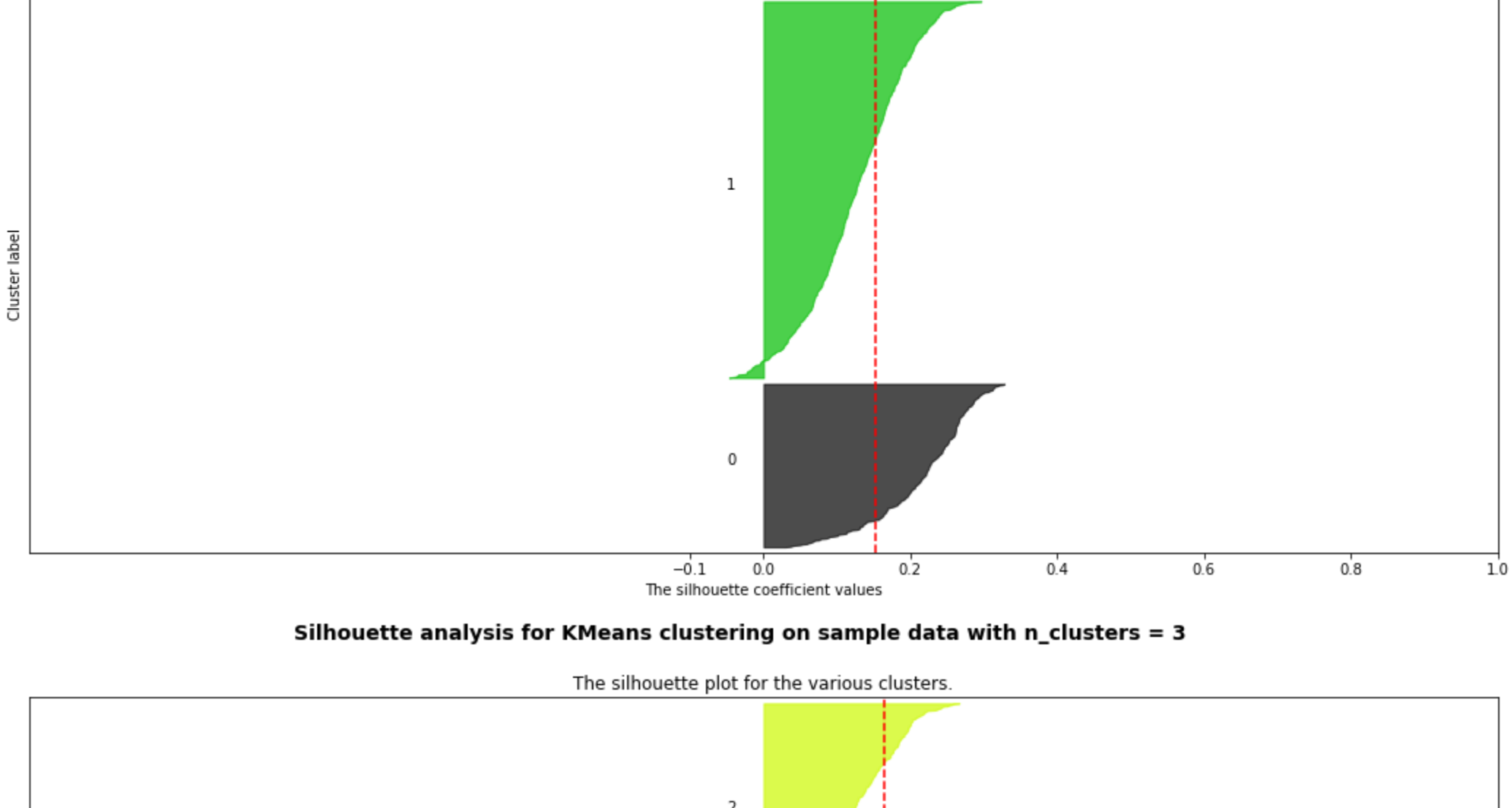
    # 2nd Plot showing the actual clusters formed
    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    plt.subplot(2, 1, 1)

    # Silhouette analysis for KMeans clustering on sample data with n_clusters = nd"
    % n_clusters,
    fontsize=16,
    fontweight="bold",
)

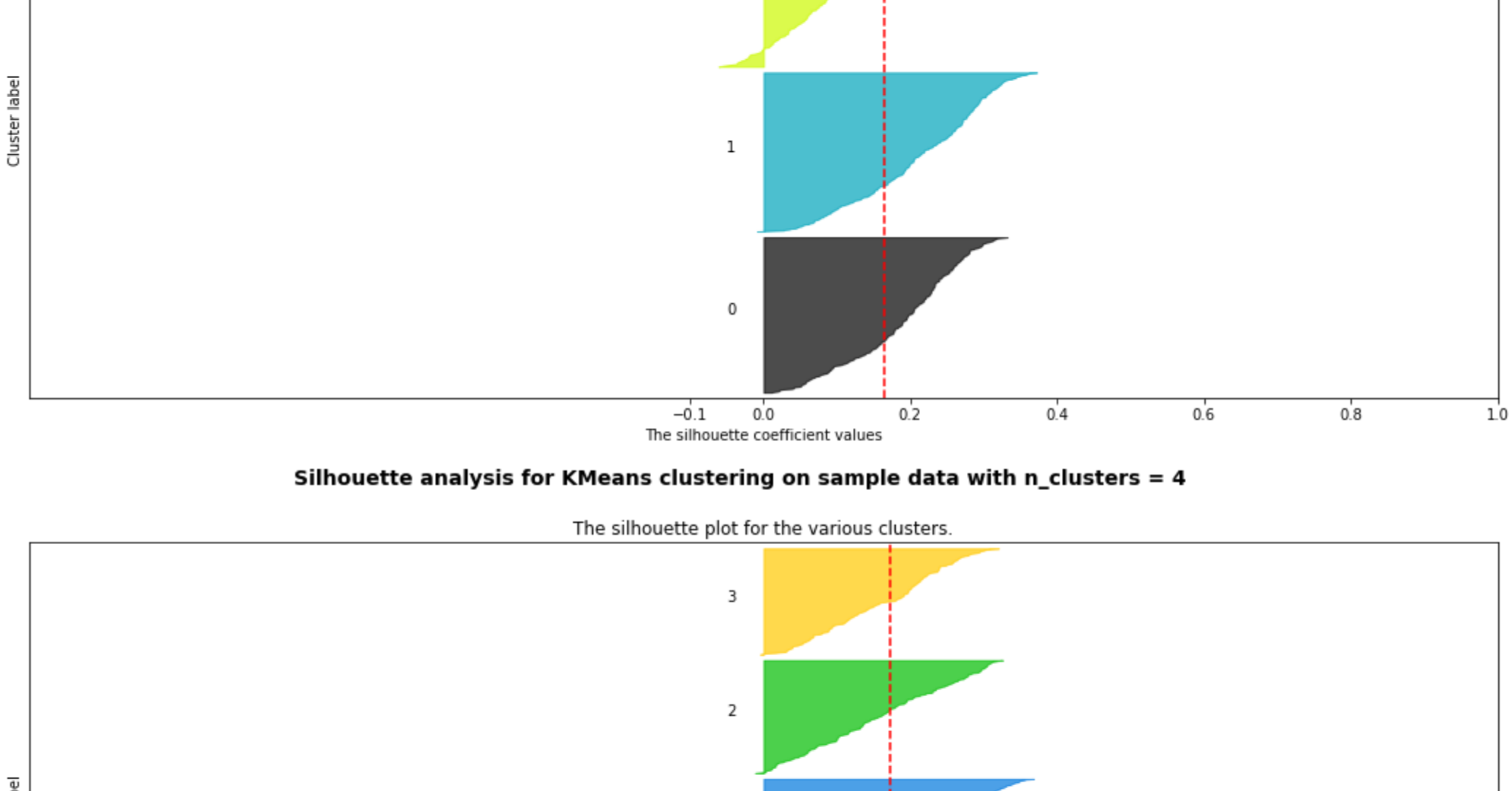
plt.show()
```

For n_clusters = 2 The average silhouette score is : 0.16241245372781554
For n_clusters = 3 The average silhouette score is : 0.16462978487411052
For n_clusters = 4 The average silhouette score is : 0.17238534298817385
For n_clusters = 5 The average silhouette score is : 0.1824323719802328
For n_clusters = 6 The average silhouette score is : 0.26342399851426513
For n_clusters = 7 The average silhouette score is : 0.2680752817489335
For n_clusters = 8 The average silhouette score is : 0.23662932262153382
For n_clusters = 9 The average silhouette score is : 0.2698184142698335
For n_clusters = 10 The average silhouette score is : 0.238659367488613
For n_clusters = 11 The average silhouette score is : 0.242658584772817
For n_clusters = 12 The average silhouette score is : 0.246867939917338
For n_clusters = 13 The average silhouette score is : 0.24453861622776219
For n_clusters = 14 The average silhouette score is : 0.2478641221287564
For n_clusters = 15 The average silhouette score is : 0.25185120611386635

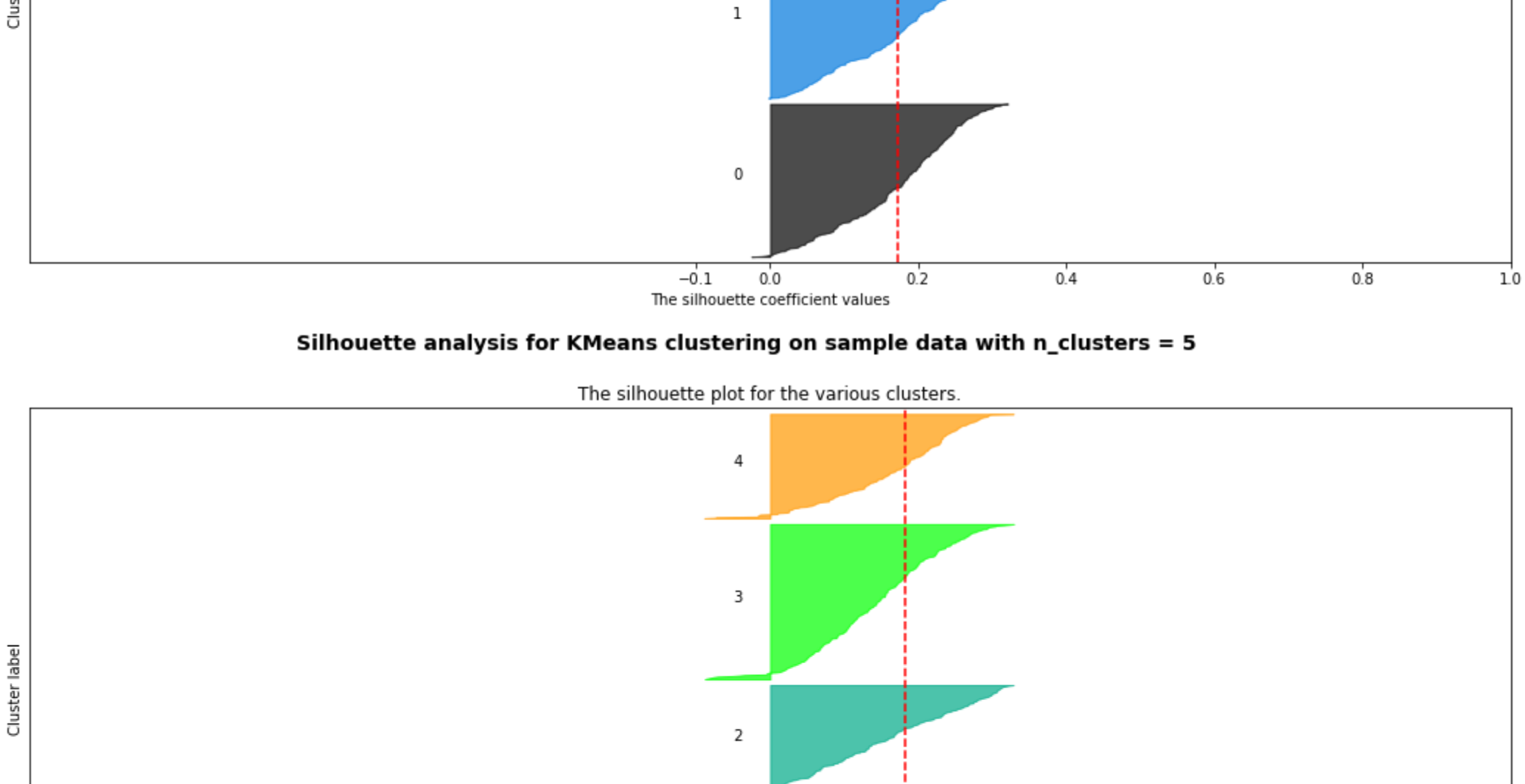
Silhouette analysis for KMeans clustering on sample data with n_clusters = 2



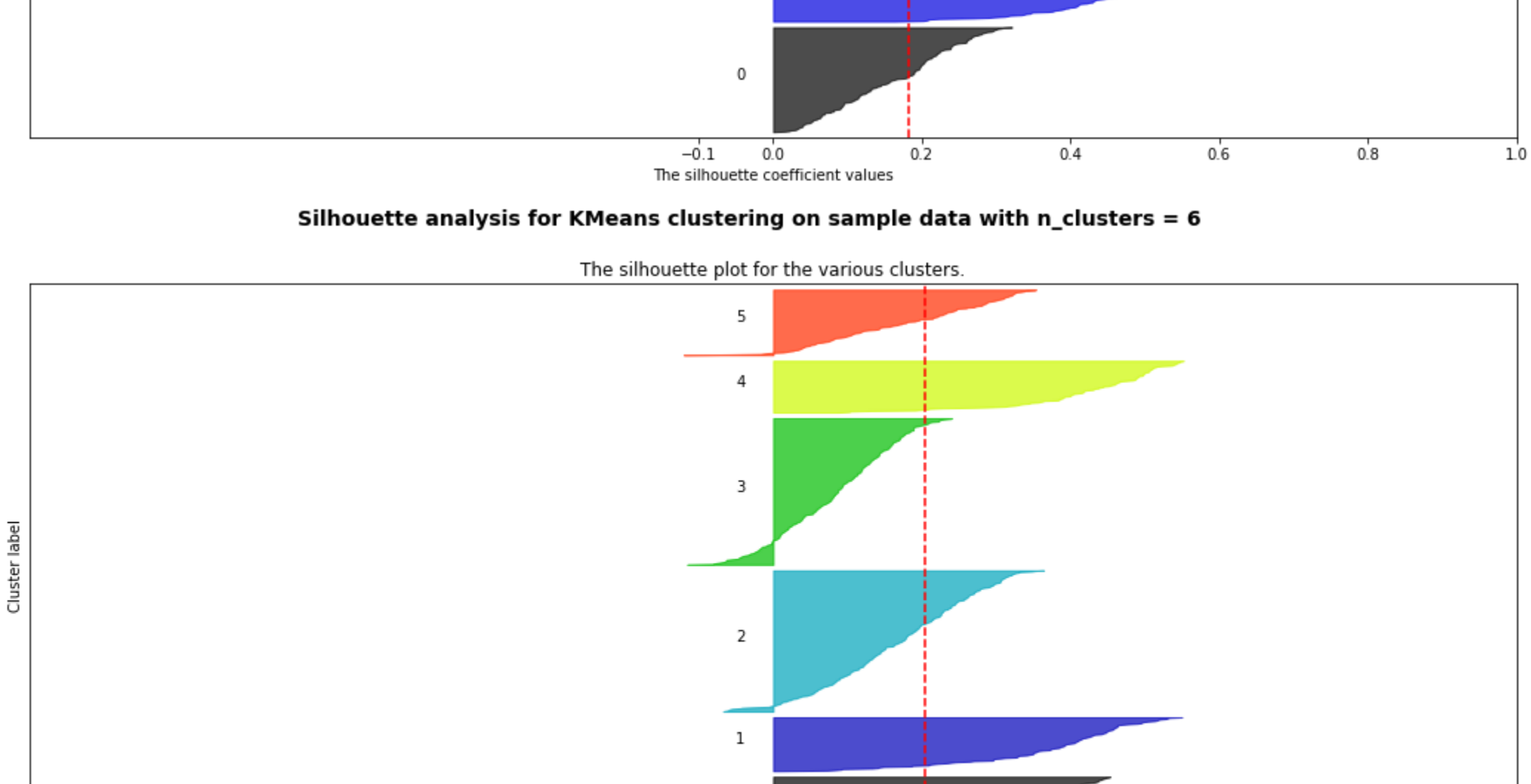
Silhouette analysis for KMeans clustering on sample data with n_clusters = 3



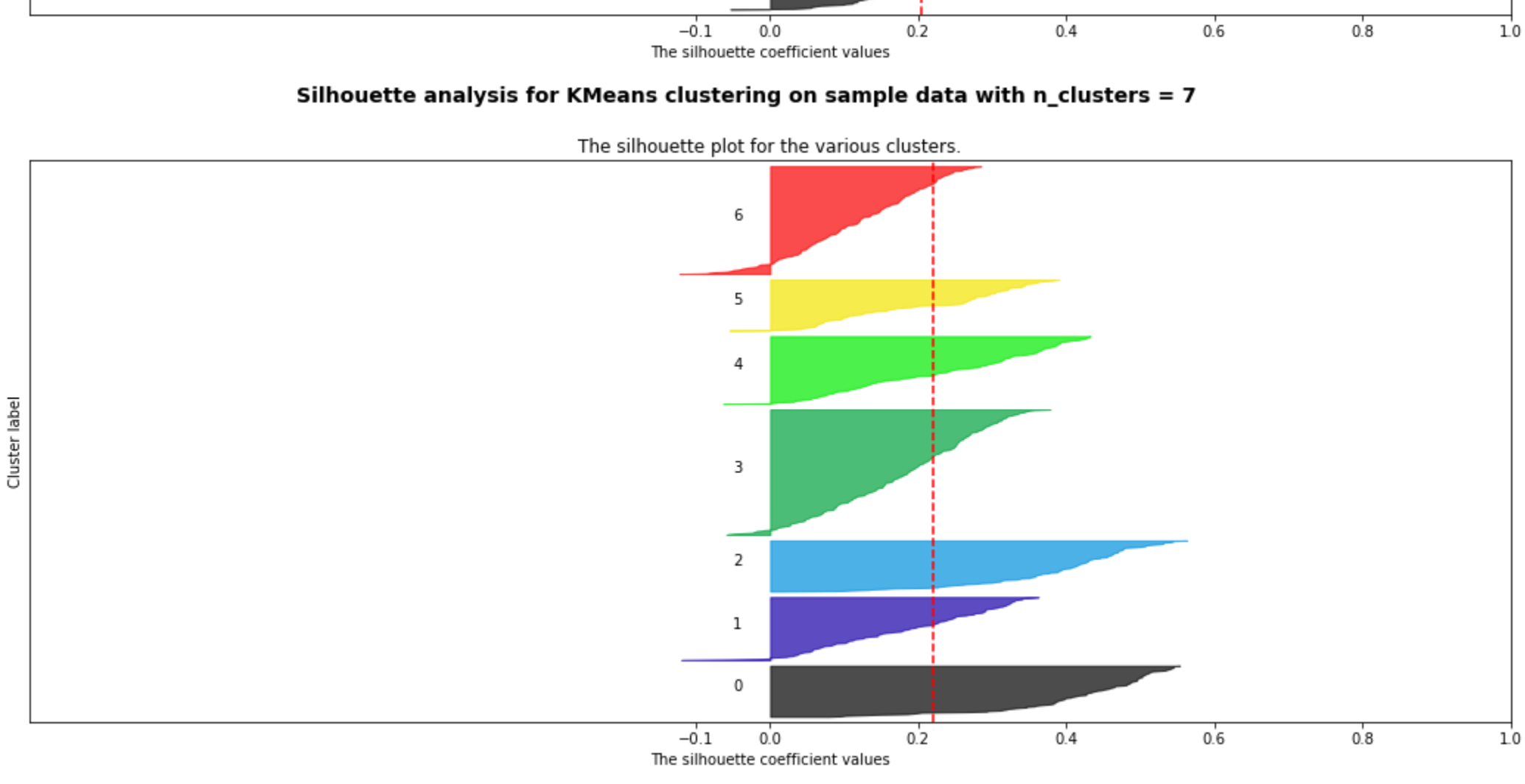
Silhouette analysis for KMeans clustering on sample data with n_clusters = 4



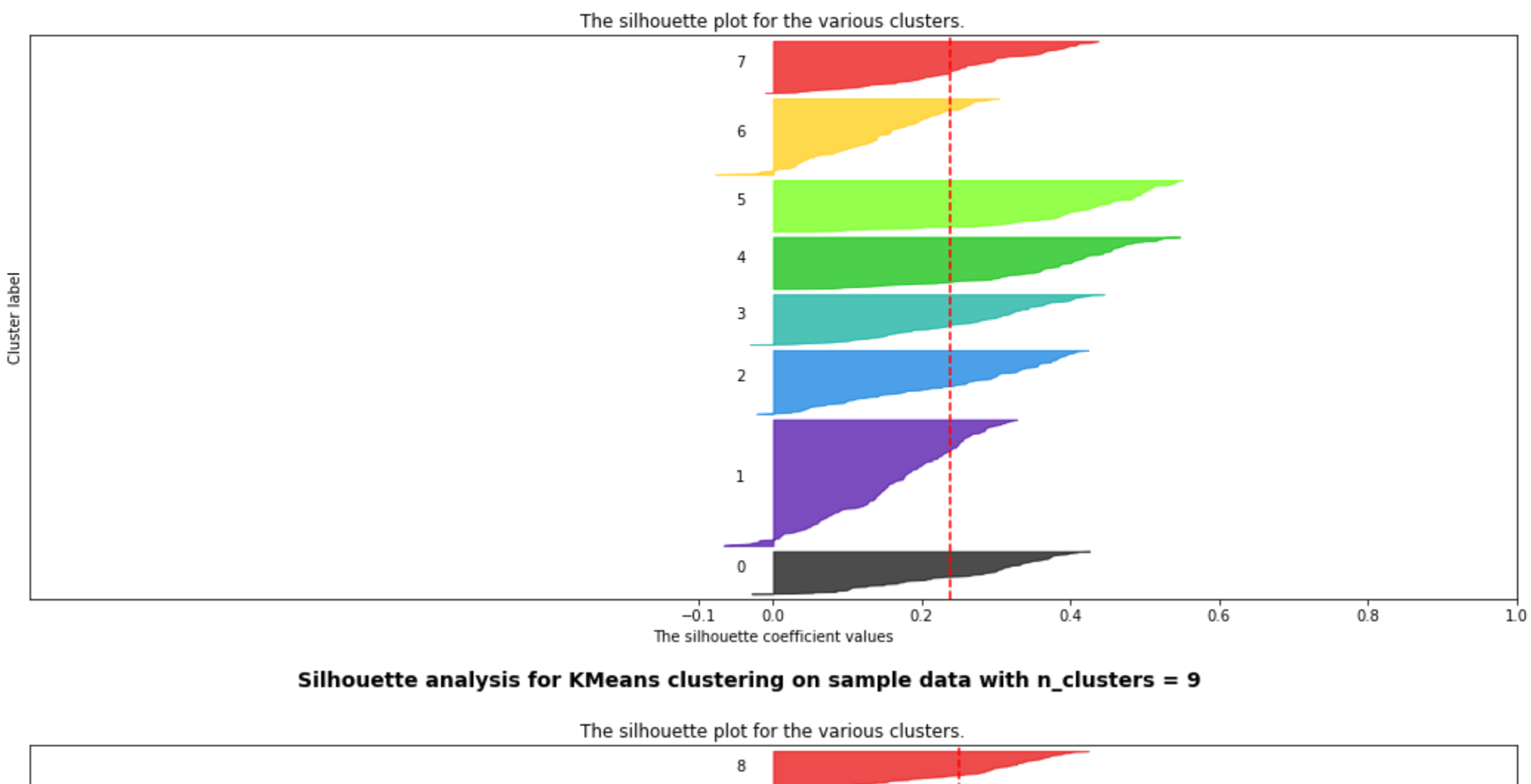
Silhouette analysis for KMeans clustering on sample data with n_clusters = 5



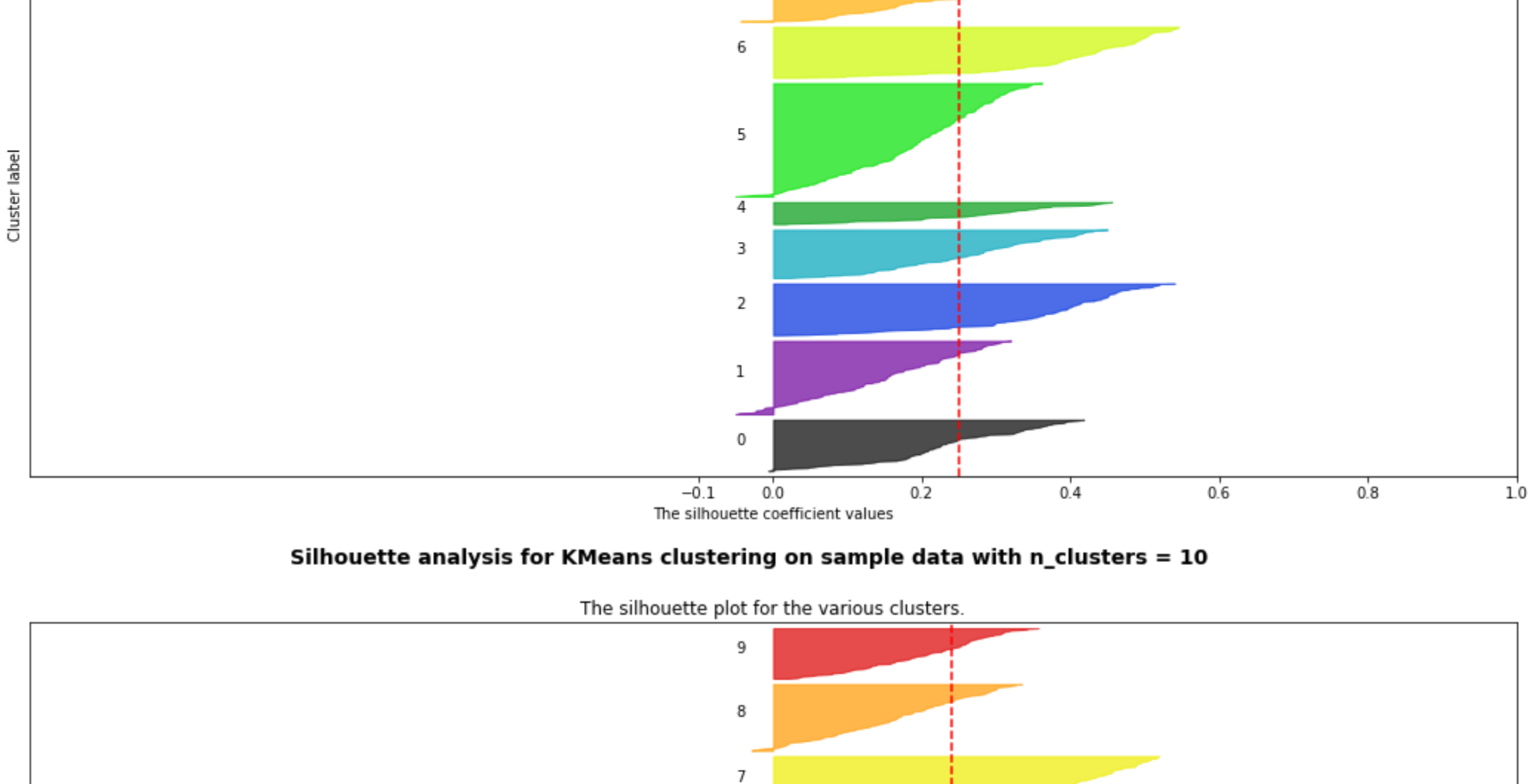
Silhouette analysis for KMeans clustering on sample data with n_clusters = 6



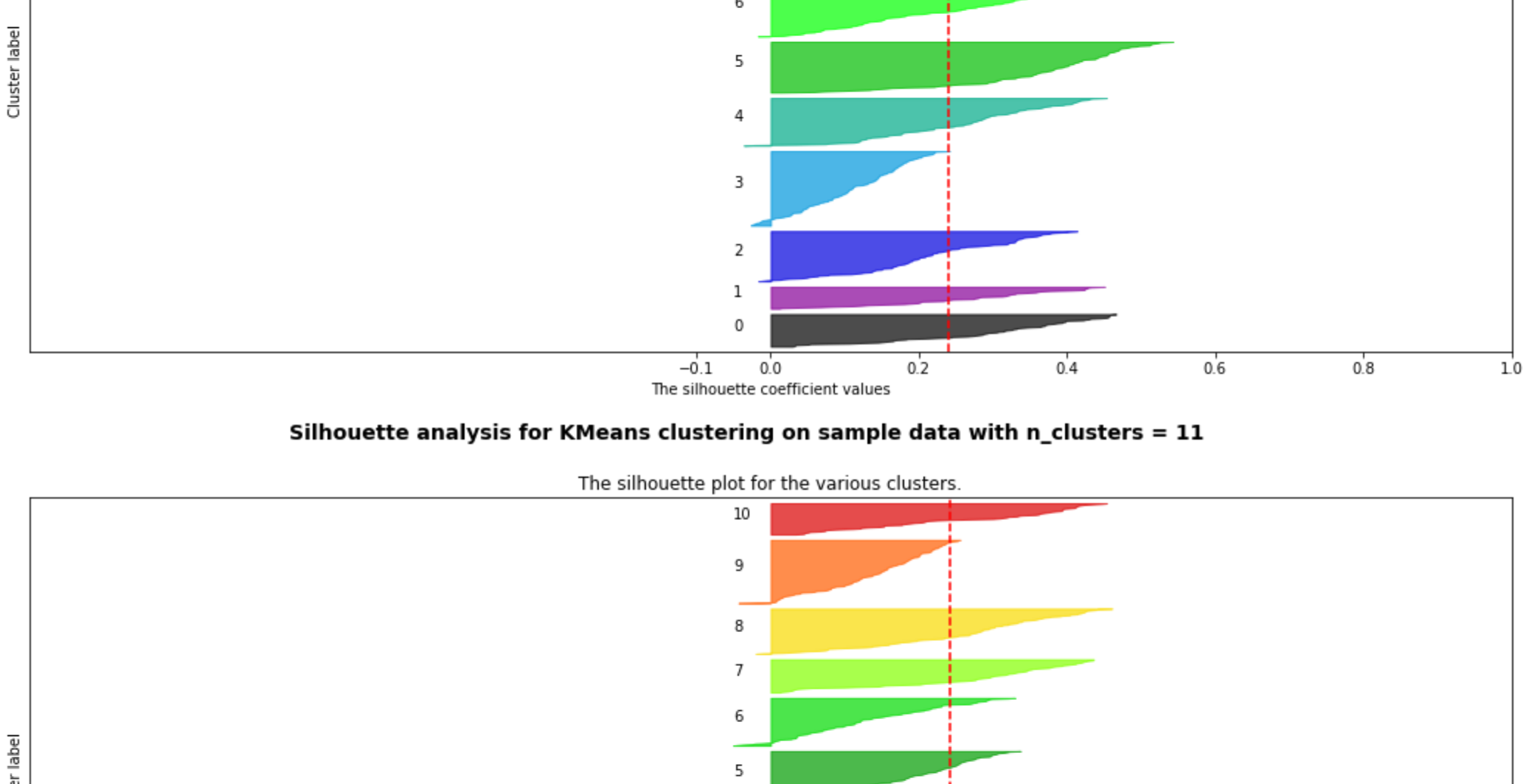
Silhouette analysis for KMeans clustering on sample data with n_clusters = 7



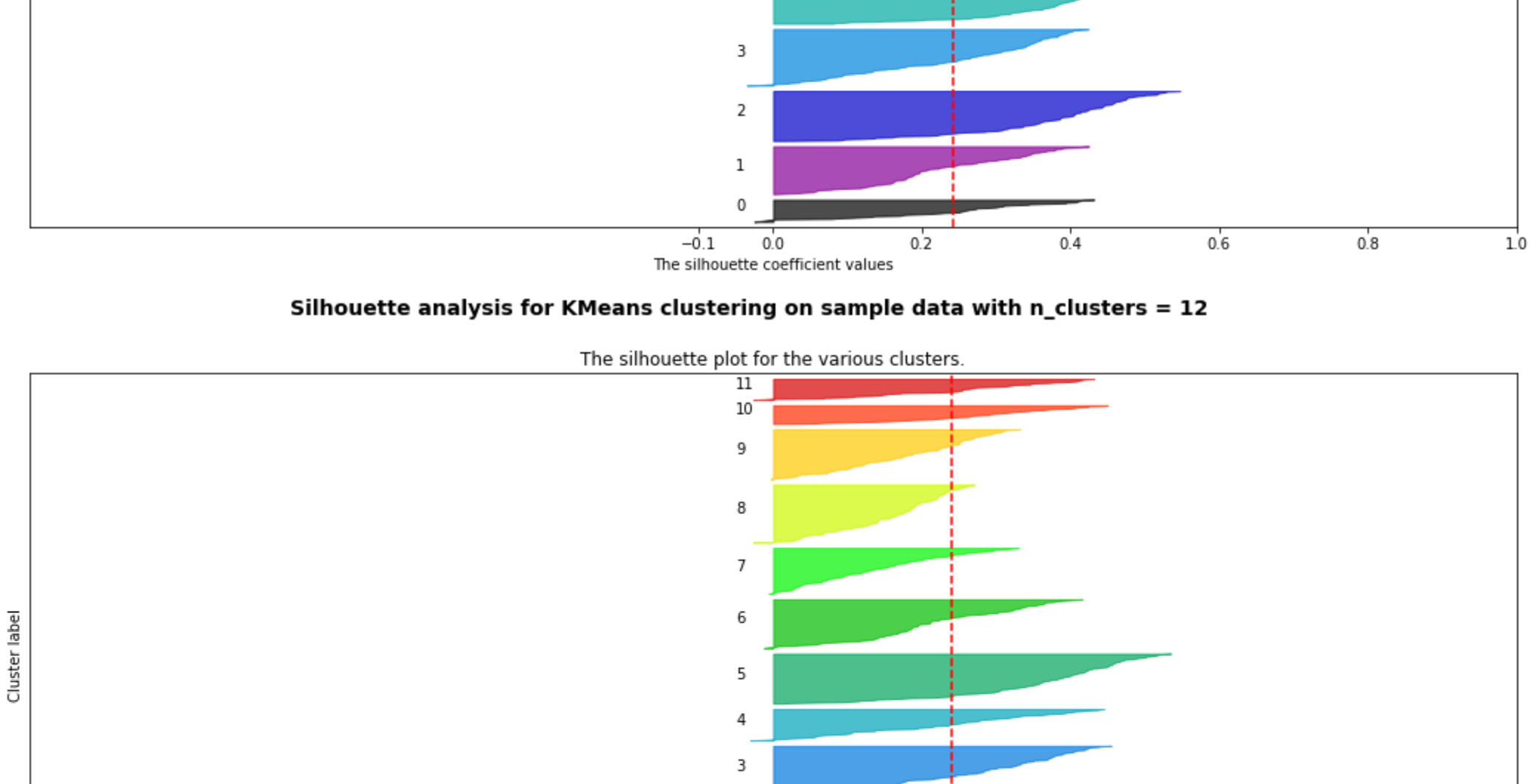
Silhouette analysis for KMeans clustering on sample data with n_clusters = 8



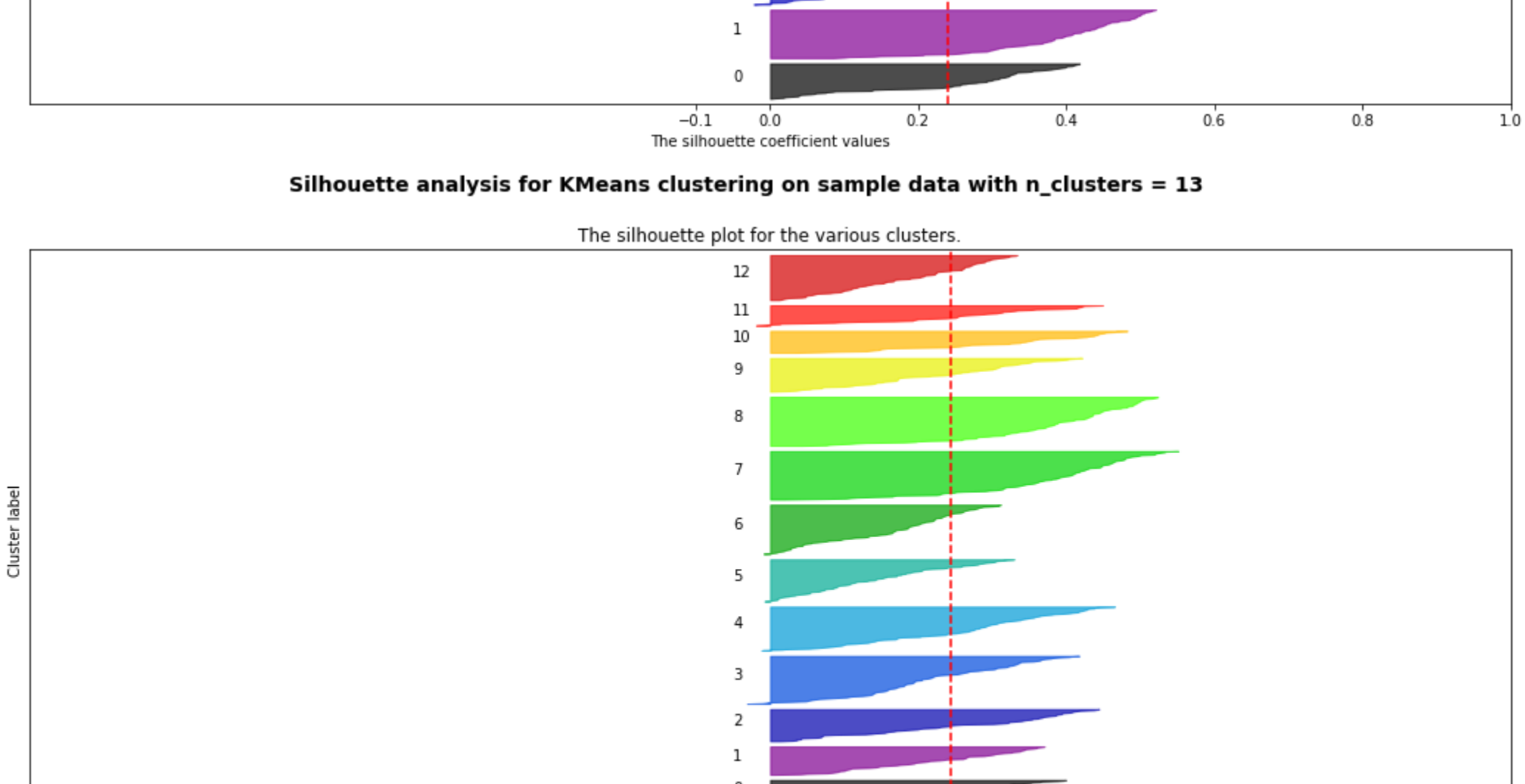
Silhouette analysis for KMeans clustering on sample data with n_clusters = 9



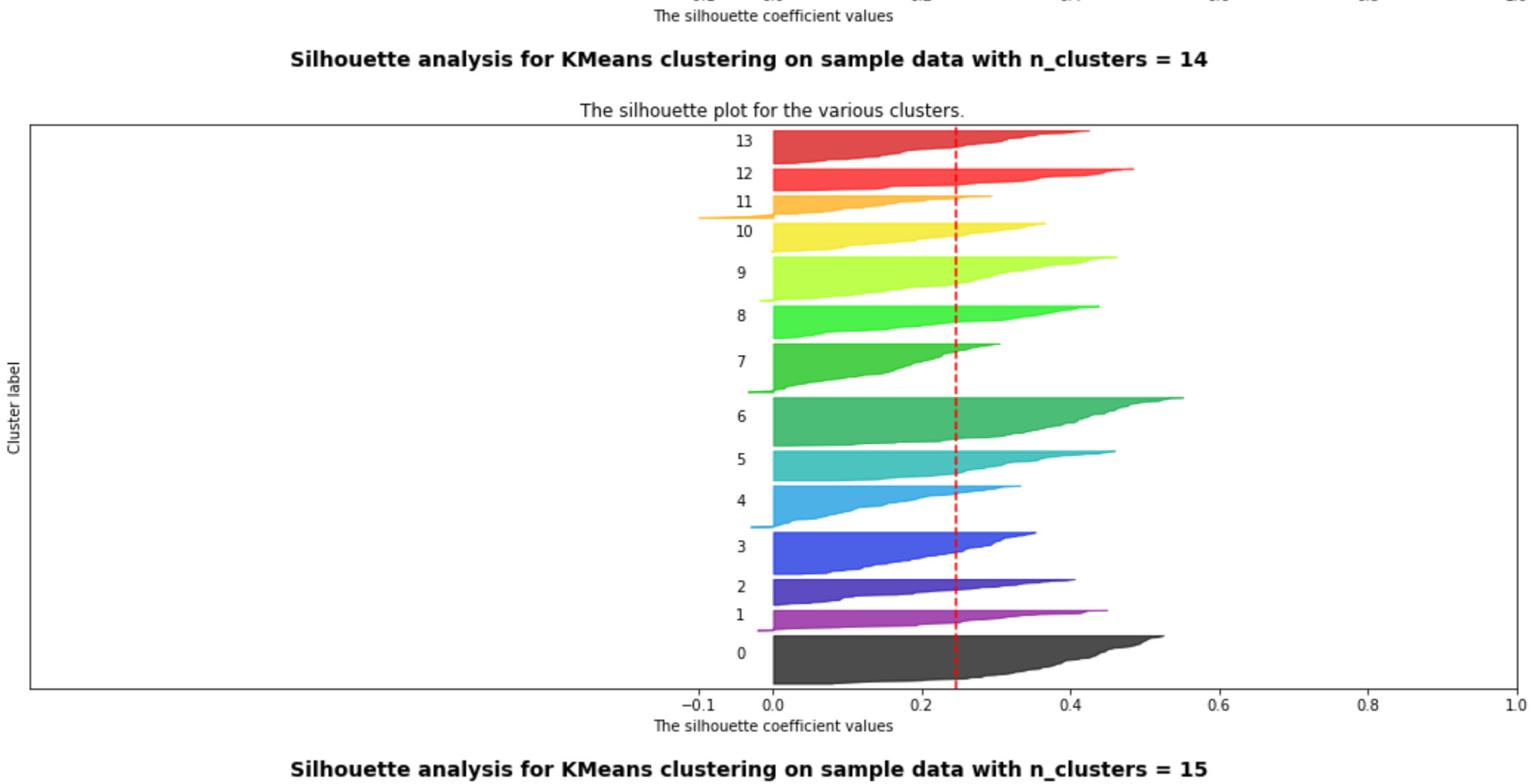
Silhouette analysis for KMeans clustering on sample data with n_clusters = 10



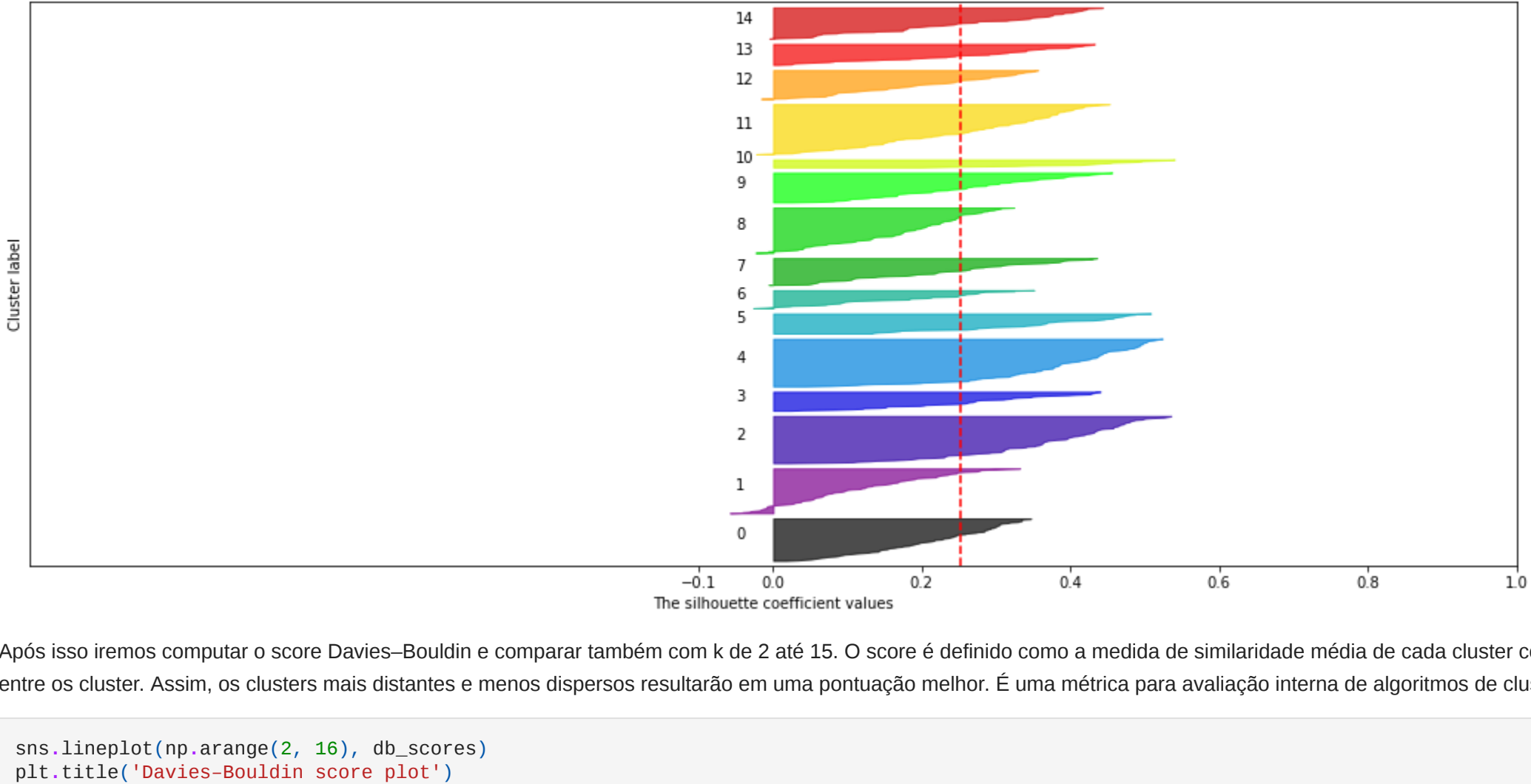
Silhouette analysis for KMeans clustering on sample data with n_clusters = 11



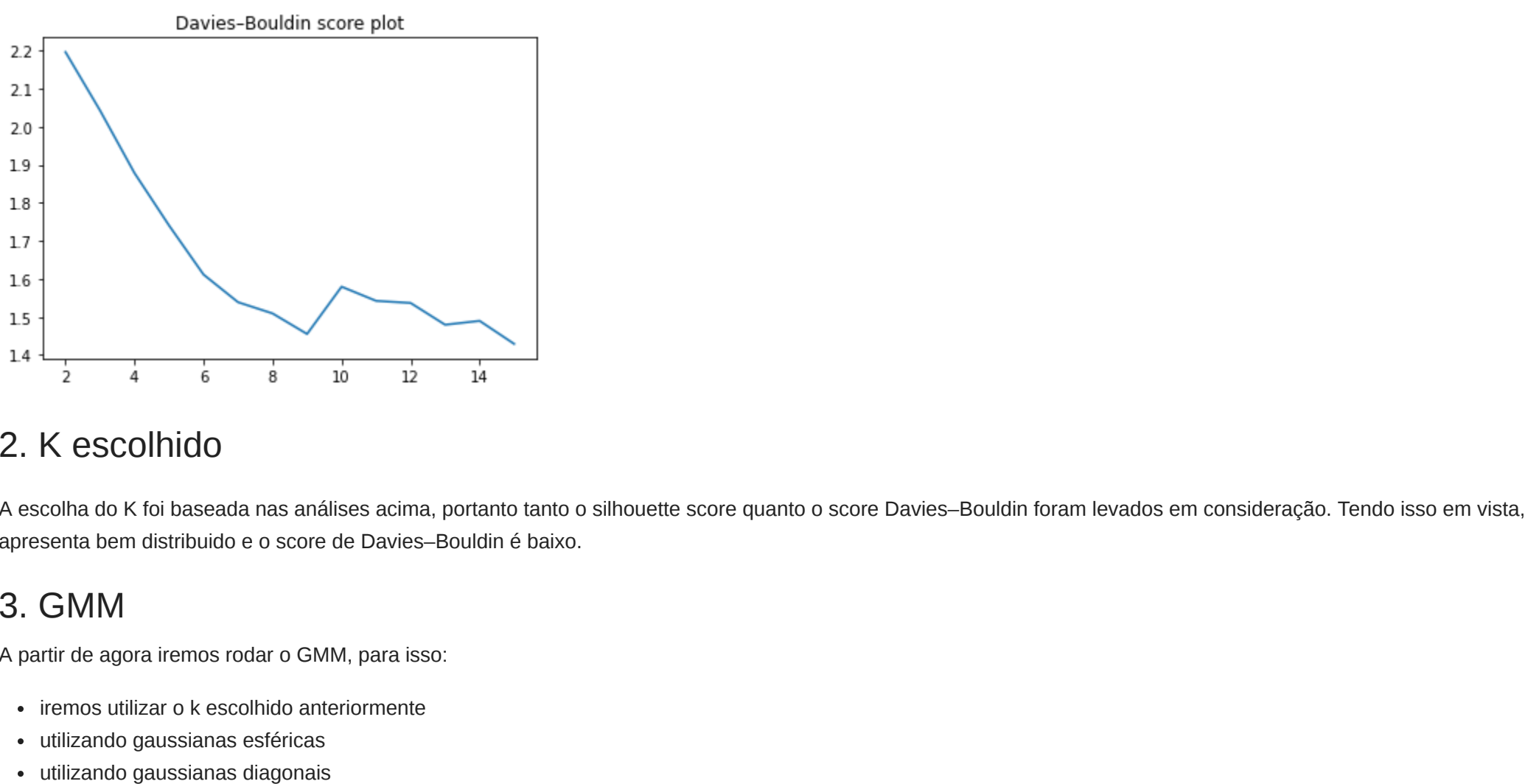
Silhouette analysis for KMeans clustering on sample data with n_clusters = 12



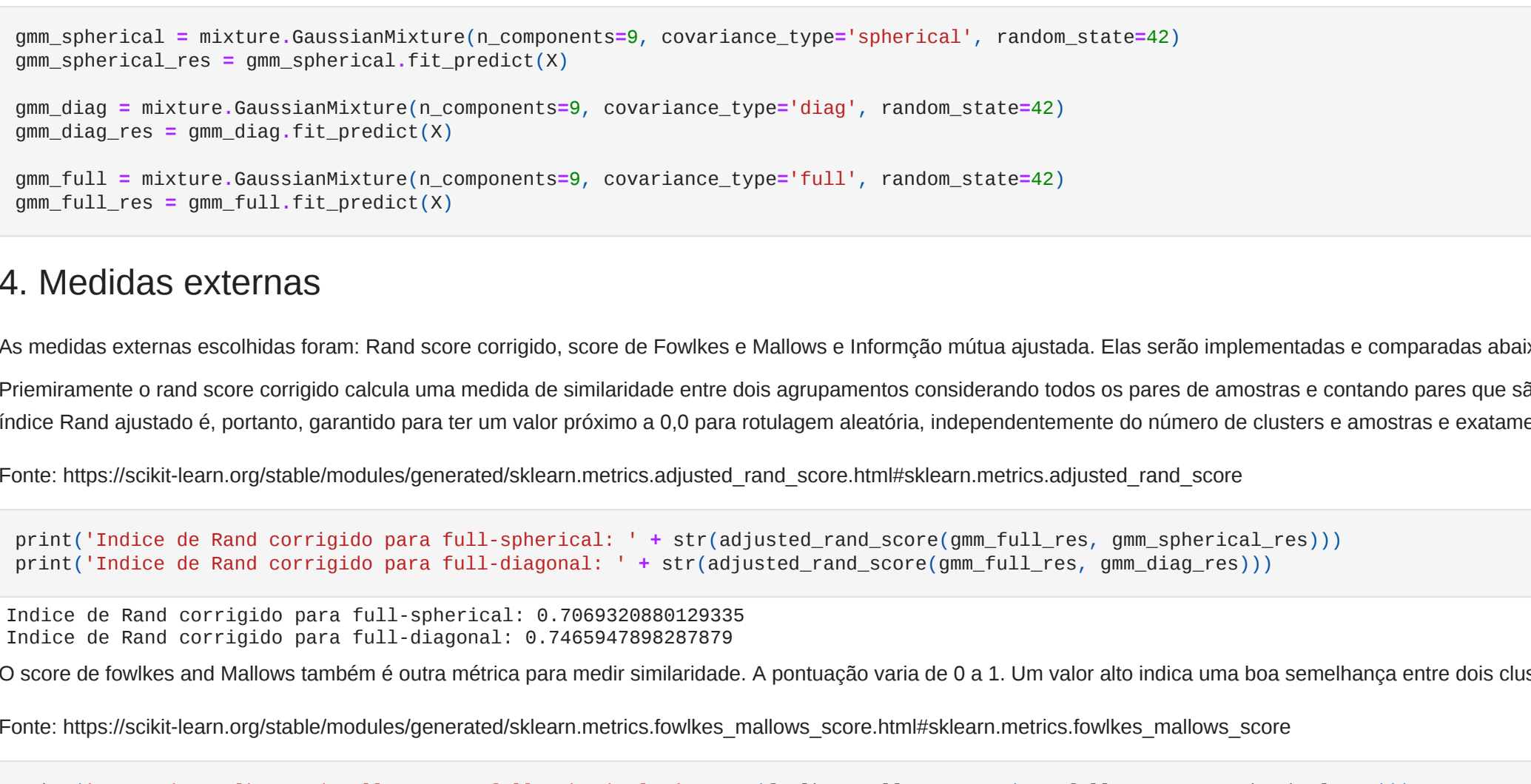
Silhouette analysis for KMeans clustering on sample data with n_clusters = 13



Silhouette analysis for KMeans clustering on sample data with n_clusters = 14

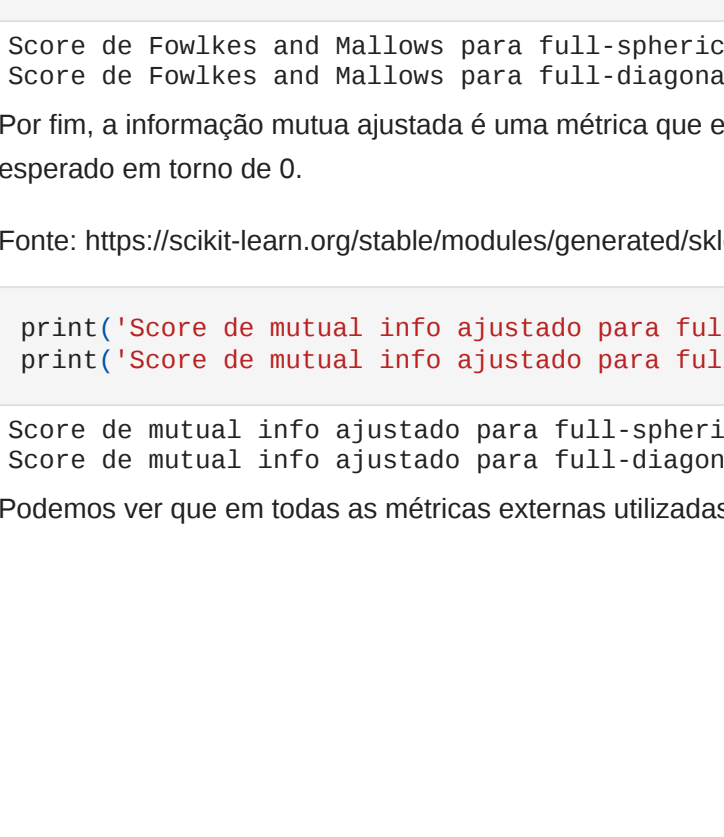


Silhouette analysis for KMeans clustering on sample data with n_clusters = 15



Após isso iremos computar o score Davies-Bouldin e comparar também com k de 2 até 15. O score é definido como a medida de similaridade média de cada cluster com seu cluster mais semelhante. A similaridade é a razão entre as distâncias dentro do cluster e entre as distâncias entre os cluster. Assim, os clusters mais distantes e menos dispersos resultam em uma pontuação melhor. É uma métrica para avaliação interna de algoritmos de clustering. Quanto mais baixo o valor do índice, melhor o clustering.

```
In [ ]: sns.lineplot(x=range(2, 16), db_scores)
plt.title('Davies-Bouldin score plot')
plt.show()
```



2. K escolhido

A escolha do K foi baseada nas análises acima, portanto tanto o silhouette score quanto o score Davies-Bouldin foram levados em consideração. Tendo isso em vista, o K escolhido foi 10, pois apresenta bom valor médio de silhouette, o grafico de silhouette para este valor se apresenta bem distribuído e o score de Davies-Bouldin é baixo.

3. GMM

A partir de agora iremos rodar o GMM, para isso:

- iremos utilizar o `K` escolhido anteriormente
- utilizando gaussianas esféricas
- utilizando gaussianas diagonais
- utilizando gaussianas sem restrição

```
In [ ]: gmm_spherical = mixture.GaussianMixture(n_components=9, covariance_type='spherical', random_state=42)
gmm_spherical_res = gmm_spherical.fit_predict(x)

gmm_diag = mixture.GaussianMixture(n_components=9, covariance_type='diag', random_state=42)
gmm_diag_res = gmm_diag.fit_predict(x)

gmm_full = mixture.GaussianMixture(n_components=9, covariance_type='full', random_state=42)
gmm_full_res = gmm_full.fit_predict(x)
```

4. Medidas externas

As medidas externas escolhidas foram: Rand score corrigido, score de Fowlkes e Mallows e Informação mútua ajustada. Elas serão implementadas e comparadas abaixo:

Primeiramente o Rand score corrigido calcula uma medida de similaridade entre dois agrupamentos considerando todos os pares de amostras e contando pares que são atribuídos no mesmo agrupamento ou em verdadeiros. O Índice Rand ajustado é, portanto, garantido para ter um valor próximo a 0.0 para agrupamento aleatório, independentemente do número de clusters e amostras e exatamente 1.0 quando os agrupamentos são idênticos.

Fonte: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html#sklearn.metrics.adjusted_rand_score

```
In [ ]: print('Índice de Rand corrigido para full-spherical: ' + str(adjusted_rand_score(gmm_full_res, gmm_spherical_res)))
print('Índice de Rand corrigido para full-diagonal: ' + str(adjusted_rand_score(gmm_full_res, gmm_diag_res)))

Índice de Rand corrigido para full-spherical: 0.7869328886129335
Índice de Rand ajustado para full-spherical: 0.7465947892827879
```

O score de Fowlkes and Mallows também é outra métrica para medir similaridade. A pontuação varia de 0 a 1. Um valor alto indica uma boa semelhança entre dois clusters.

```
In [ ]: print('Score de Fowlkes and Mallows para full-spherical: ' + str(fowlkes_mallows_score(gmm_full_res, gmm_spherical_res)))
print('Score de Fowlkes and Mallows para full-diagonal: ' + str(fowlkes_mallows_score(gmm_full_res, gmm_diag_res)))

Score de Fowlkes and Mallows para full-spherical: 0.74892643488388
Score de Fowlkes and Mallows para full-diagonal: 0.7169945952520921
```

Por fim, a informação mútua ajustada é uma métrica que está intimamente relacionado à variação da informação. Retorna um valor de 1 quando as duas partições são idênticas (ou seja, perfeitamente combinadas). As partições aleatórias (rotulagens independentes) tem um retorno espremido em torno de 0.

Fonte: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_mutual_info_score.html#sklearn.metrics.adjusted_mutual_info_score

```
In [ ]: print('Score de mutual info ajustado para full-spherical: ' + str(adjusted_mutual_info_score(gmm_full_res, gmm_spherical_res)))
print('Score de mutual info ajustado para full-diagonal: ' + str(adjusted_mutual_info_score(gmm_full_res, gmm_diag_res)))

Score de mutual info ajustado para full-spherical: 0.7873659759388855
Score de mutual info ajustado para full-diagonal: 0.7465947892828851
```

Podemos ver que em todas as métricas externas utilizadas os valores são relativamente altos, o que indica que a similaridade dos dois agrupamento também é alta. Portanto, os resultados das gaussianas livres acabaram ficando parecidos com as gaussianas que possuem restrições.