使用python+sklearn实现在KMeans聚类结果上通过轮廓分析选择聚类的数量

原创 魔图哥 机器学习算法与知识图谱 8月19日

收录于话题

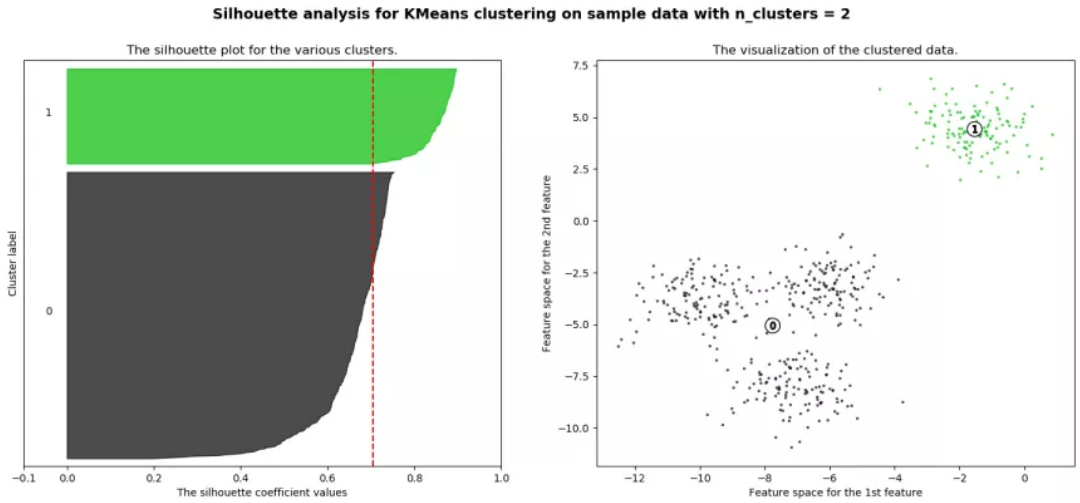
#sklearn 45 #scikit-learn学习路线 164

轮廓分析(silhouette analysis)可用于研究聚类结果之间的分离距离。轮廓图是一个聚类中的每个点与相邻聚类中的点之间接近程度的度量指标，从而提供了一种直观地评估参数(如聚类的数量)的方法。此度量指标的范围为[-1,1]。

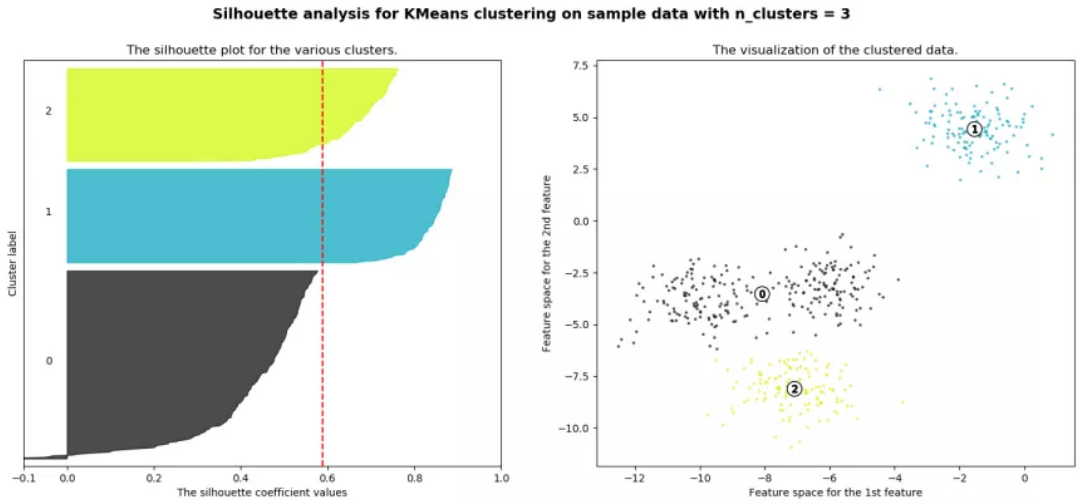
接近+1的(被称为)轮廓系数的值表示相邻聚类的样本距离很远；值为0表示样本在两个相邻聚类之间的决策边界上或非常接近决策边界；而负值表示这些样本可能已分配给错误的聚类。

在本例中，轮廓分析用于选择 `n_clusters` 的最佳值。轮廓图显示，对于本示例给定的数据，`n_clusters` 的取值3、5和6是一个不好的选择，这是由于存在有轮廓分数低于平均水平的聚类，并且轮廓图的大小存在较大波动。在确定2到4之间时，轮廓分析更具矛盾性。

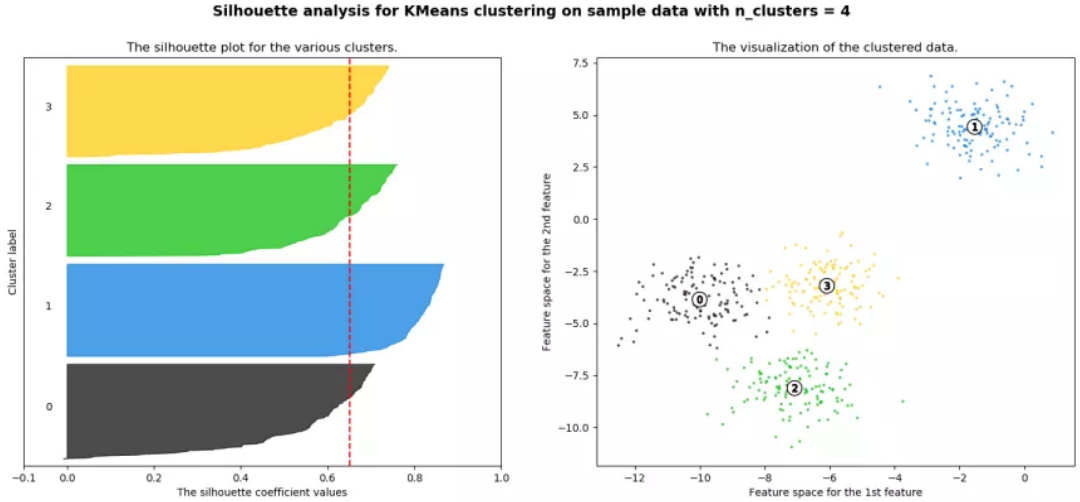
同样从轮廓图的厚度可以看到聚类的大小。当 `n_clusters` 等于2时，由于将3个子聚类分组为一个大聚类，因此聚类0的轮廓图的大小较大；但是，当 `n_clusters` 等于4时，所有图或多或少都具有相似的厚度，因此具有相似的大小，这也可以从右侧的标记散点图进行验证。



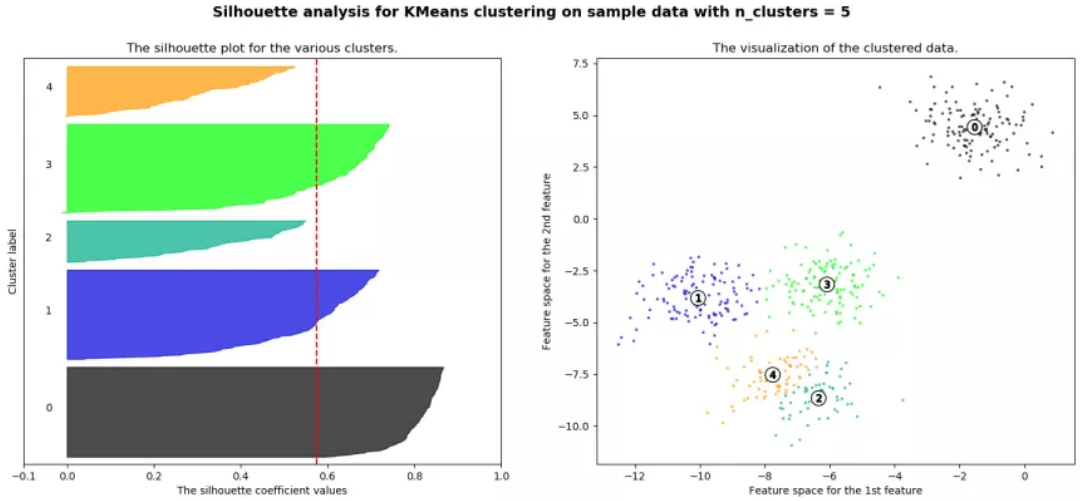
sphx_glr_plot_kmeans_silhouette_analysis_001



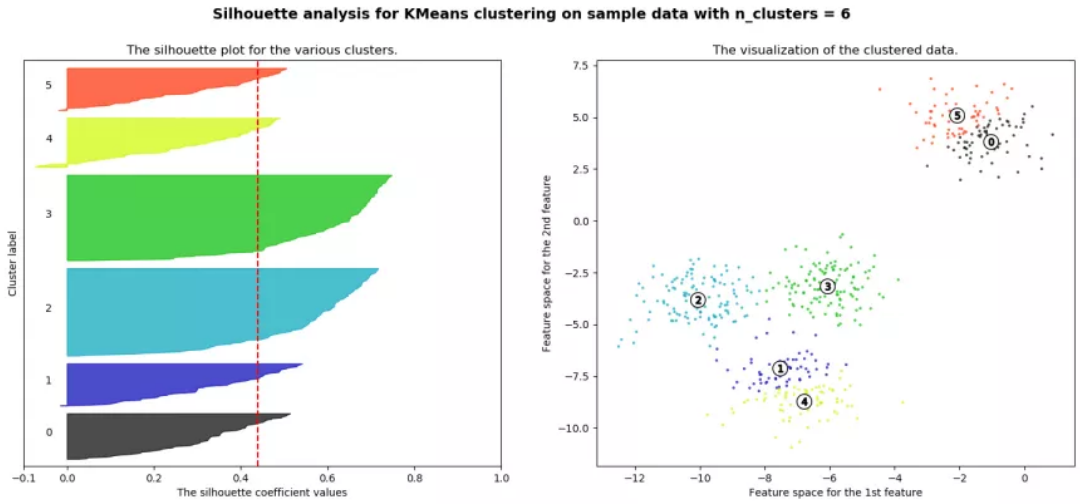
sphx_glr_plot_kmeans_silhouette_analysis_002



sphx_glr_plot_kmeans_silhouette_analysis_003



sphx_glr_plot_kmeans_silhouette_analysis_004



sphx_glr_plot_kmeans_silhouette_analysis_005

输出：

```
For n_clusters = 2 The average silhouette_score is : 0.7049787496083262
For n_clusters = 3 The average silhouette_score is : 0.5882004012129721
For n_clusters = 4 The average silhouette_score is : 0.6505186632729437
For n_clusters = 5 The average silhouette_score is : 0.5745566973301872
For n_clusters = 6 The average silhouette_score is : 0.43902711183132426
```

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

import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np

print(__doc__)

# 从make_blobs生成样本数据
# 特定设置具有一个不同的聚类，并且3个聚类被放在了一起
X, y = make_blobs(n_samples=500,
                  n_features=2,
                  centers=4,
                  cluster_std=1,
                  center_box=(-10.0, 10.0),
                  shuffle=True,
                  random_state=1)  # 为了可重现性

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

for n_clusters in range_n_clusters:
    # 创建一个具有1行2列的子图
    fig, (ax1, ax2) = plt.subplots(1, 2)
```

```
    fig.set_size_inches(18, 7)

    # 第一个子图是轮廓图
    # 轮廓系数的范围可以为-1、1，但在此示例中轮廓系数
    # 在[-0.1，1]之内
    ax1.set_xlim([-0.1, 1])
    # （n_clusters + 1）* 10用于在轮廓之间插入空格
    # 来区分各个聚类的图，以明确划分它们。
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    # 用n_clusters值和一个随机数生成器初始化聚类器
    # 随机数的种子seed=10
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(X)

    # silhouette_score给出所有样本的平均值。
    # 这提供了形成聚类的密度和分离的透视图
    silhouette_avg = silhouette_score(X, cluster_labels)
    print("For n_clusters =", n_clusters,
          "The average silhouette_score is :", silhouette_avg)

    # 计算每个样本的轮廓分数
    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):
        # 汇总属于聚类i的样本的轮廓分数，并对它们进行排序
        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)

        # 在轮廓图的中间标注聚类编号
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        # 为下一个图计算新的y_lower
        y_lower = y_upper + 10  # 0样本的10

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

```
    # 所有值平均轮廓得分的垂直线
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])  # 清除y轴标签/刻度
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    # 第二图显示了实际形成的聚类
    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')

    # 标记聚类
    centers = clusterer.cluster_centers_
    # 在聚类中心绘制白色圆圈
    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()
```

脚本的总运行时间： （0分钟1.106秒）

估计的内存使用量： 8 MB

https://mybinder.org/badge_logo.svg

下载Python源代码:plot_kmeans_silhouette_analysis.py

下载Jupyter notebook源代码:plot_kmeans_silhouette_analysis.ipynb

由Sphinx-Gallery生成的画廊