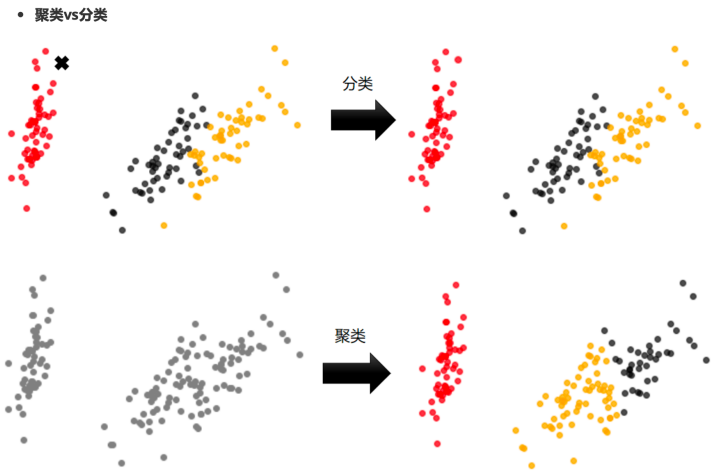
2023年7月3日 星期一

Cluster

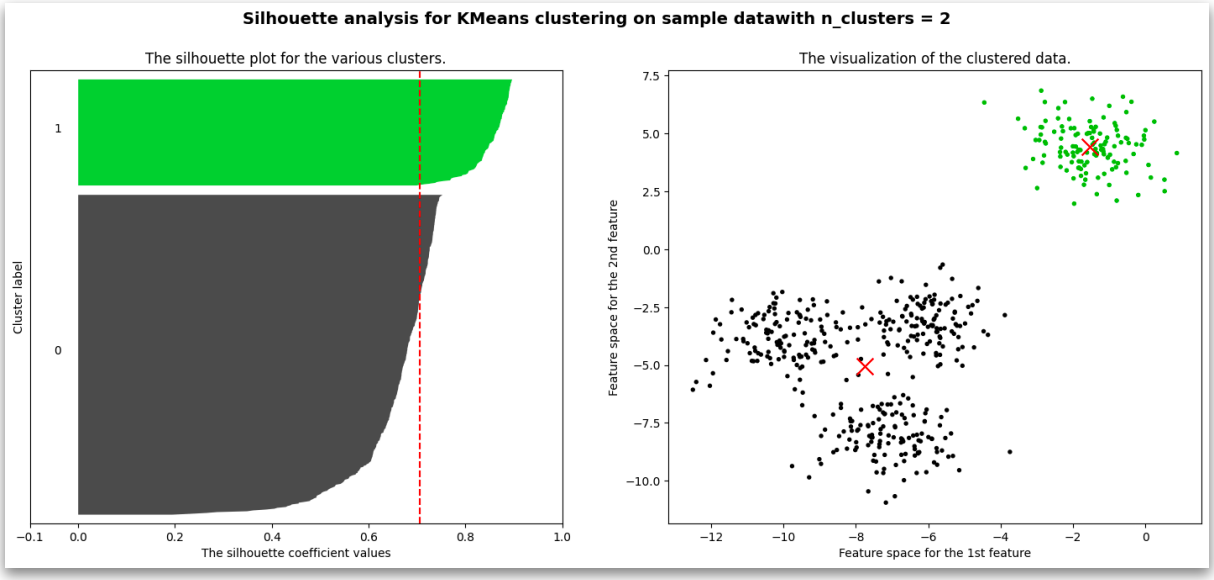
KMeans

- 前言：
  - 聚類：
    - 定義：探索每個組是否有關係
    - 學習類型：無監督（無需進行標籤的訓練）
    - 算法：Kmeans
    - 特性：
      - 結果是不確定的
      - 不一定能反映數據的真實分類，會受到需求產生好結果或壞結果
  - 分類：
    - 定義：從已分組的數據去學習，將結果放到分好的組中
    - 學習類型：監督式（需要進行標籤的訓練）
    - 算法：Decision Tree、Logistic Regression、貝葉斯
    - 特性：
      - 結果是確定的
      - 分類的結果是客觀的

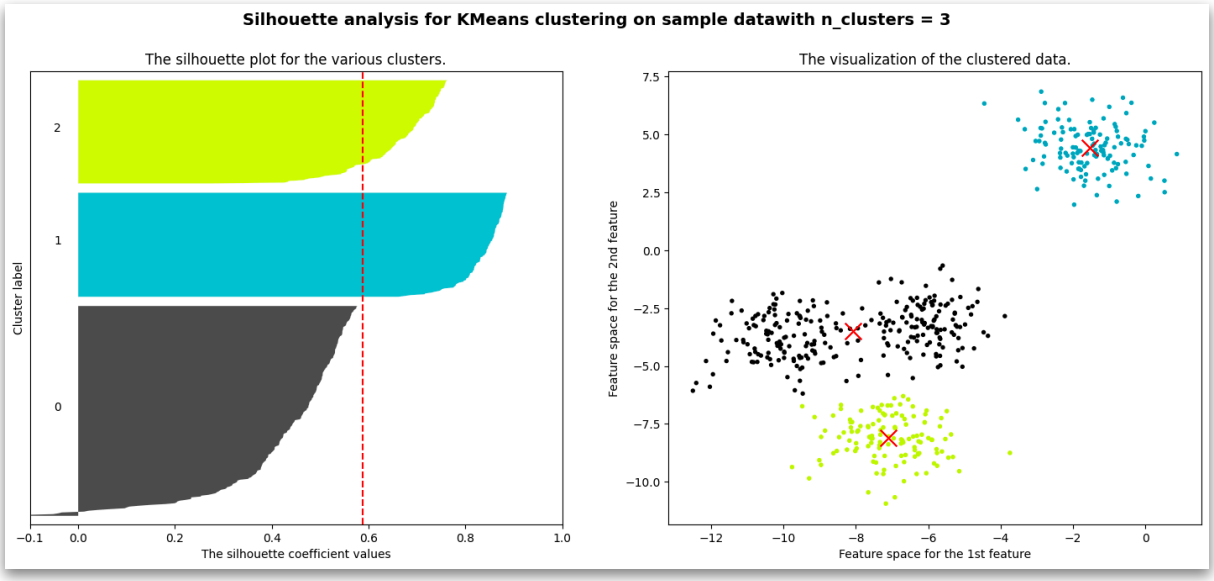


- 如何衡量聚類的好壞？
  - 以 KMeans 的目標“簇內差距要小，簇外差距要大”為基礎，因此透過衡量簇內差異來衡量好壞。
- 是否能用 Inertia 作為衡量指標？
  - 可以，但不好，因為 Inertia 會受到 n_clusters 影響，若 n_clusters 越大 Inertia 勢必越小。
- 常用的指標：輪廓係數
  - 評價 ”簇內稠密程度“ 和 ”簇間離散程度“ 來評估聚類好壞的指標
  - Score 越高越好

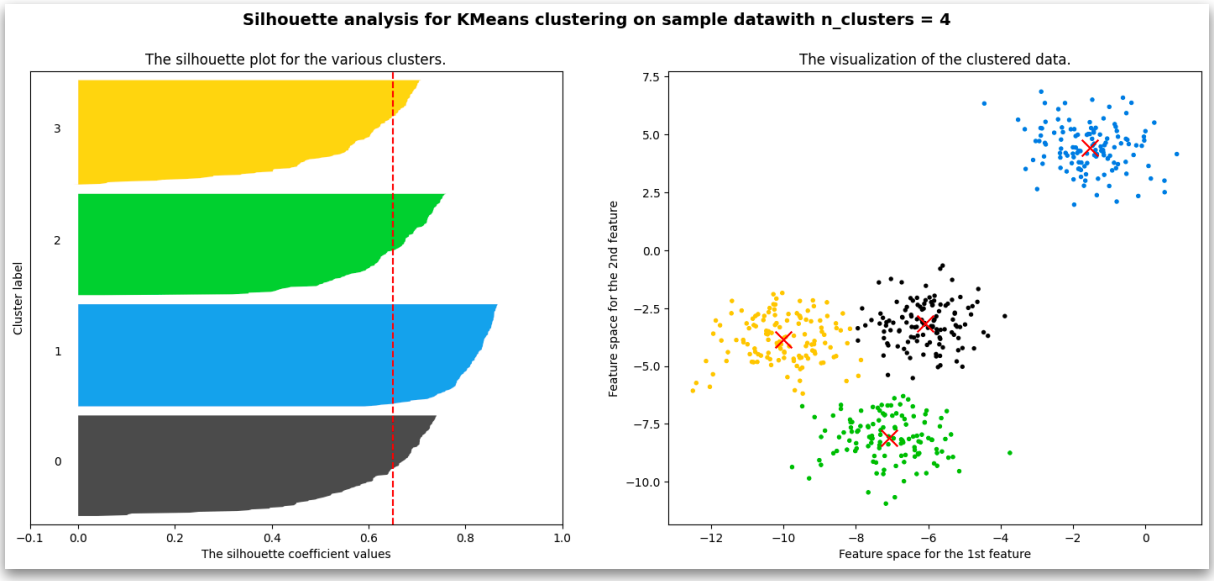
- 實作一：如何透過輪廓係數找到最佳 n_cluster 參數？
  - 目標：
    - 比較整體的平均輪廓係數和在不同n_cluster下各個樣本的輪廓係數 （子圖1）
    - 畫出聚類完之後各個簇的分布狀況（子圖2）
  - 結果：
    - n_cluster = 2
      - For n_clusters = 2 The average silhouette_score is : 0.7049787496083262



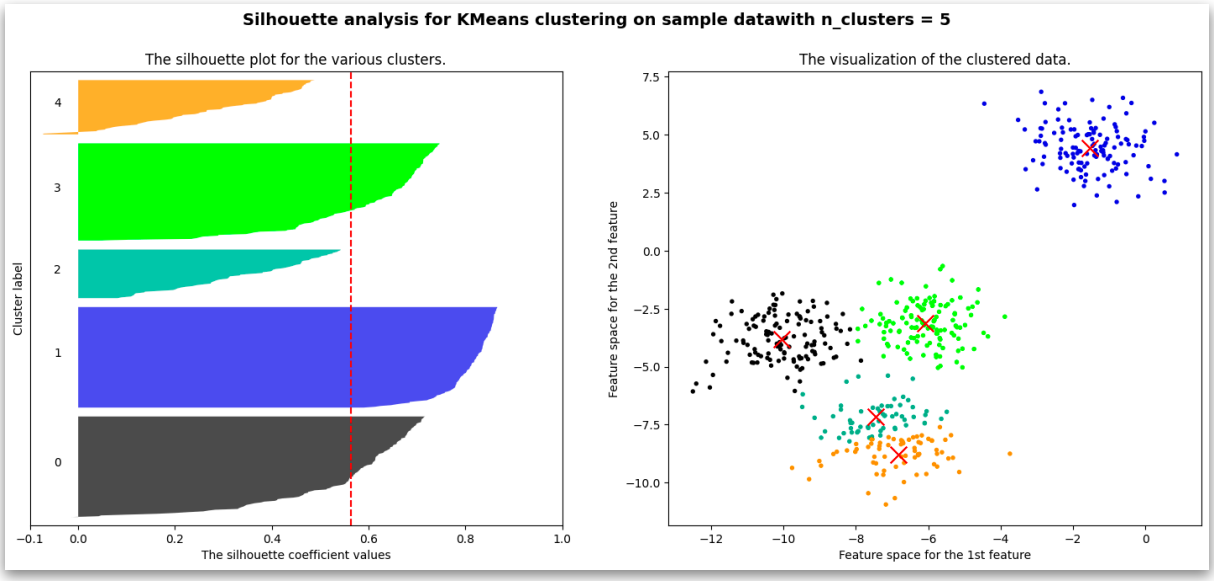
    - n_cluster = 3
      - For n_clusters = 3 The average silhouette_score is : 0.5882004012129721



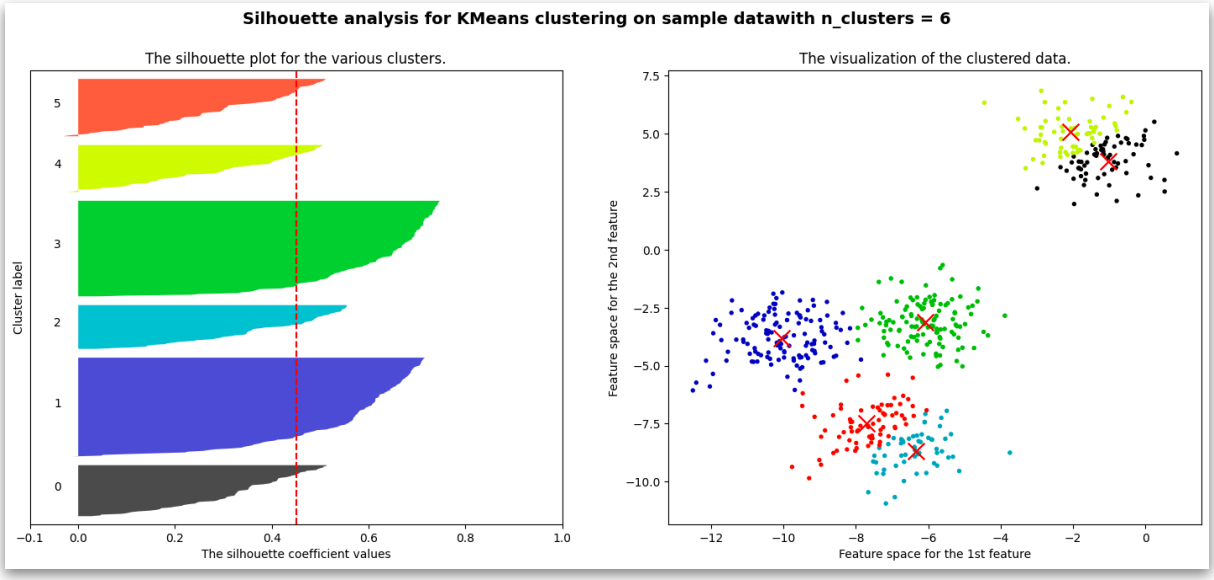
- n_cluster = 4
  - For n_clusters = 4 The average silhouette_score is : 0.6505186632729437



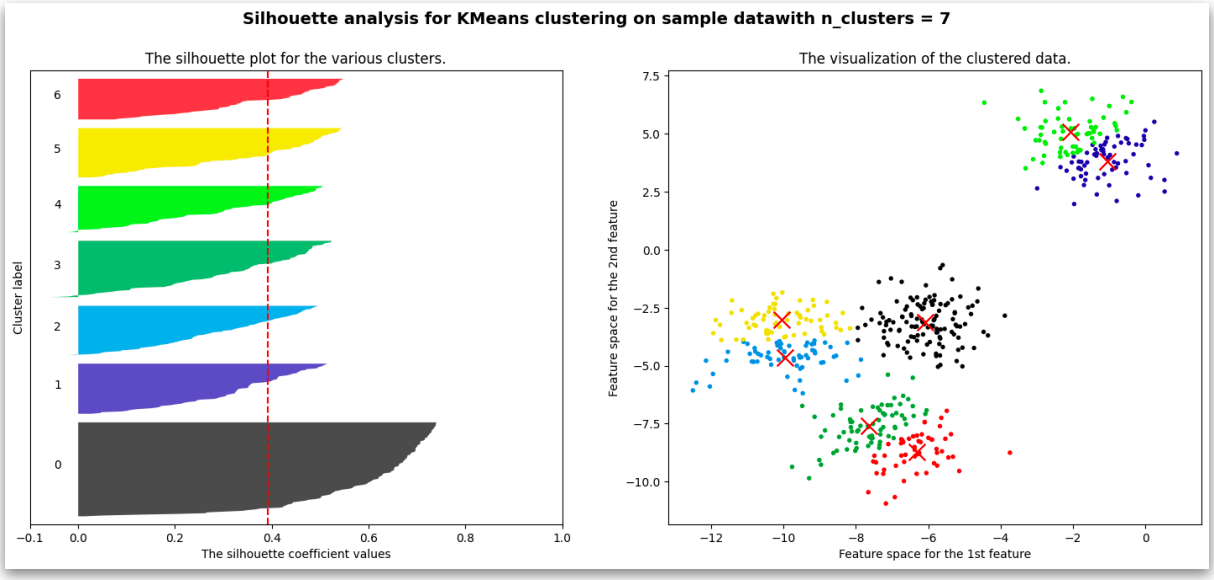
- n_cluster = 5
  - For n_clusters = 5 The average silhouette_score is : 0.56376469026194



- n_cluster = 6
  - For n_clusters = 6 The average silhouette_score is : 0.4504666294372765



- n_cluster = 7
  - For n_clusters = 7 The average silhouette_score is : 0.39092211029930857



- 結果分析＆心得：

  1. 由子圖1我們可以更傾向選取所有樣本都超過紅色虛線的
  2. 由子圖2我們可以更傾向選取所有樣本更為密集集中質心的
  3. 我們更傾向於選取average silhouette_score較高的

  綜合以上三點，n_cluster = 4 為最佳選擇。

- 實作二：利用KMeans做矢量量化的降維應用
  - 核心概念：利用質心代替與之對應的樣本來達成降維的效果
  - Code：
    - 資料預處理

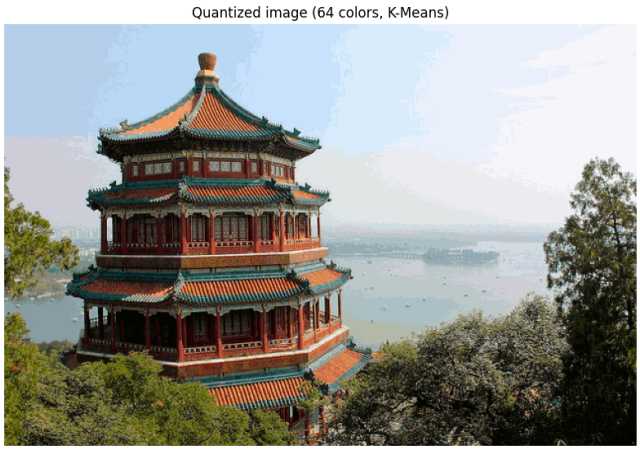
```
# KMeans 不接受三維數組做為特徵矩陣 => 要將圖片降維
n_clusters = 64
#plt.imshow 在浮點數表現優秀 => 做歸一化將數據壓縮到 [0,1]
china = np.array(china, dtype=np.float64) / china.max()
#將圖片格式轉為矩陣格式
w, h, d = original_shape = tuple(china.shape)
#assert d == 3 代表若d不為3則python中斷執行
assert d == 3
image_array = np.reshape(china, (w * h, d))
image_array.shape
```

    - KMeans 的矢量量化

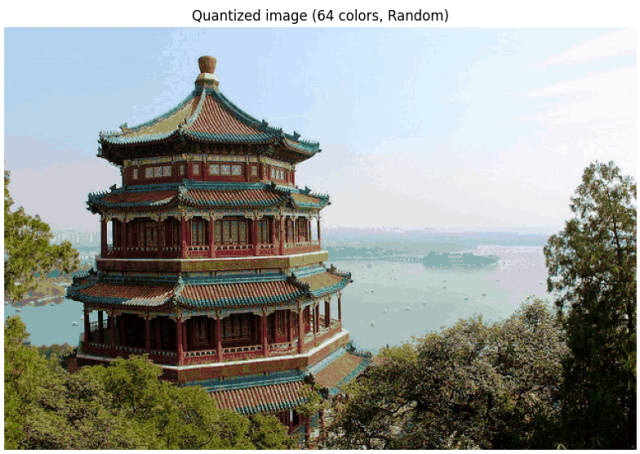
```
#使用隨機1000個數據找出64個質心
image_array_sample = shuffle(image_array, random_state=0)[:1000]
kmeans = KMeans(n_clusters=n_clusters, random_state=0).fit(image_array_sample)
kmeans.cluster_centers_.shape

#之後再對所有數據進行聚類
labels = kmeans.predict(image_array)
labels.shape

#用質心替換所有數據 (Key Point)
#kmeans.cluster_centers_ : 質心的位置
#labels : 預測後對應的質心編號
image_kmeans = image_array.copy()
for i in range(w*h):
    image_kmeans[i] = kmeans.cluster_centers_[labels[i]]

image_kmeans
pd.DataFrame(image_kmeans).drop_duplicates().shape

#恢復圖片結構
image_kmeans = image_kmeans.reshape(w,h,d)
image_kmeans.shape
```

- 結果
  - 原圖

  - Kmeans



  - Random



- 分析 & 心得

  由結果可明顯看出 Kmeans 比 Random 好，與原圖的落差較小。

  KMeans 從 colors = 96615 降為 colors = 64 大幅降低了特徵的數量。