KMeans

May 15, 2023

0.1 Clustering dengan K-MeansClustering dengan K-Means

0.1.1 1. Contoh Clustrering code dengan menentukan jumlah k terlebih dahulu

Berikut adalah cara membangkitkan dataset

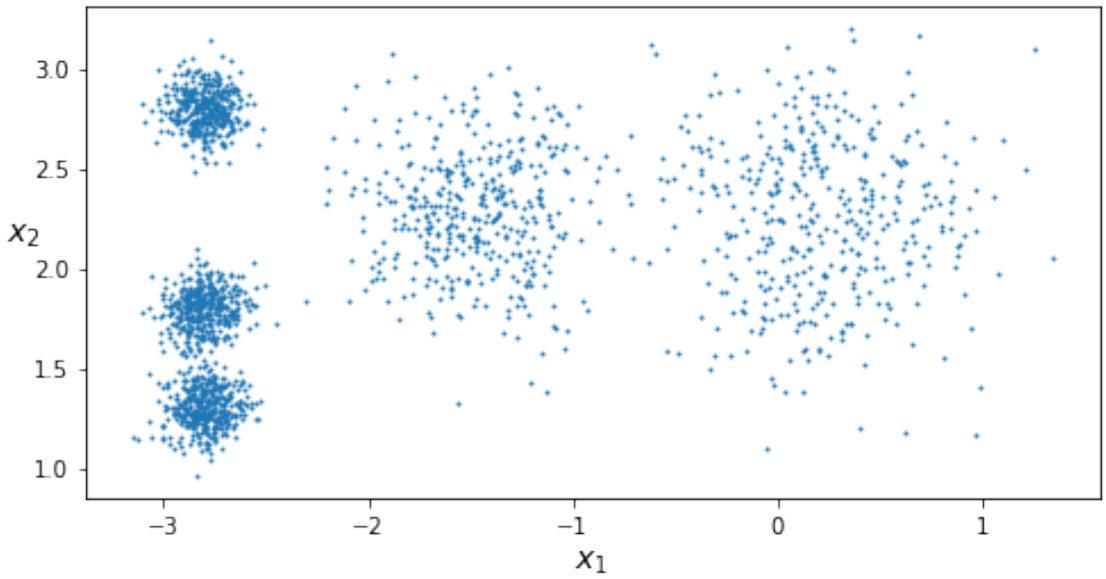
[4]:
```
from sklearn.datasets import make_blobs
import numpy as np
blob_centers = np.array(
    [[ 0.2,  2.3],
     [-1.5 ,  2.3],
     [-2.8,  1.8],
     [-2.8,  2.8],
     [-2.8,  1.3]])
blob_std = np.array([0.4, 0.3, 0.1, 0.1, 0.1])
X, y = make_blobs(n_samples=2000, centers=blob_centers,
                  cluster_std=blob_std, random_state=7)
```

Kemudian untuk melakukan plotting dataset

[5]:
```
import matplotlib.pyplot as plt
def plot_clusters(X, y=None):
    plt.scatter(X[:, 0], X[:, 1], c=y, s=1)
    plt.xlabel("$x_1$", fontsize=14)
    plt.ylabel("$x_2$", fontsize=14, rotation=0)
```

Ploting dataset x di plot clusters

[6]:
```
plt.figure(figsize=(8, 4))
plot_clusters(X)
plt.show()
```



Melakukan training algoritma clustering dari K-Means pada dataset di atas. Kita harus menspesifikasikan jumlah kluster k yang harus ditemukan oleh algoritma. Pada gambar ini sudah sangat jelas bahwa jumlah cluster yang harus ditemukan adalah k=5.

[9]:
```
from sklearn.cluster import KMeans

k=5
kmeans = KMeans(n_clusters=k, random_state=42)
y_pred = kmeans.fit_predict(X)
```

Instance dari KMeans menyimpan copy dari label-label instances hasil training pada variabel labels_instance.

[11]:
```
y_pred #### array dibawah hasil label clastering  dari kmeans training diatas
```

[11]:
```
array([4, 0, 1, …, 2, 1, 0], dtype=int32)
```

[12]:
```
y_pred is kmeans.labels_
```

[12]:
```
True
```

Kita juga dapat melihat hasil estimasi 5 lokasi titik pusat (centroids) dari setiap cluster yang ditemukan:

[13]:
```
kmeans.cluster_centers_
```

[13]:
```
array([[-2.80389616,  1.80117999],
       [ 0.20876306,  2.25551336],
       [-2.79290307,  2.79641063],
```

```
       [-1.46679593,  2.28585348],
       [-2.80037642,  1.30082566]])
```

Menentukan termasuk cluster mana sebuah data instance baru, yaitu menggunakan jarak centroid terdekat dari data tersebut.

[15]:
```
X_new = np.array([[0, 2], [3, 2], [-3, 3], [-3, 2.5]])
#Data baru yang akan dimasukan ke clastering training diatas
kmeans.predict(X_new)
```

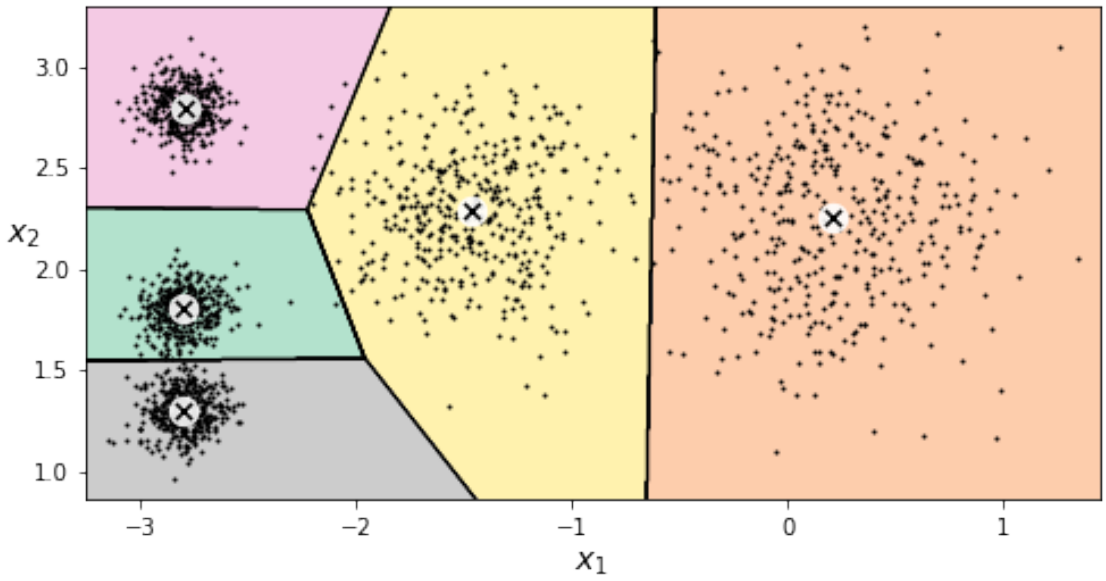
[15]:
```
array([1, 1, 2, 2], dtype=int32)
```

Function Memplot batas keputusan (decision boundary) dari cluster-cluster tersebut dengan menggunakan diagram Voronoi (Voronoi tesselation)

[17]:
```
 def plot_data(X):
    plt.plot(X[:, 0], X[:, 1], 'k.', markersize=2)
def plot_centroids(centroids, weights=None, circle_color='w', cross_color='k'):
    if weights is not None:
        centroids = centroids[weights > weights.max() / 10]
    plt.scatter(centroids[:, 0], centroids[:, 1],
                marker='o', s=30, linewidths=8,
                color=circle_color, zorder=10, alpha=0.9)
    plt.scatter(centroids[:, 0], centroids[:, 1],
                marker='x', s=50, linewidths=50,
                color=cross_color, zorder=11, alpha=1)
def plot_decision_boundaries(clusterer, X, resolution=1000, show_centroids=True,
                             show_xlabels=True, show_ylabels=True):
    mins = X.min(axis=0) - 0.1
    maxs = X.max(axis=0) + 0.1
    xx, yy = np.meshgrid(np.linspace(mins[0], maxs[0], resolution),
                         np.linspace(mins[1], maxs[1], resolution))
    Z = clusterer.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]),
                cmap="Pastel2")
    plt.contour(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]),
            linewidths=1, colors='k')
    plot_data(X)
    if show_centroids:
        plot_centroids(clusterer.cluster_centers_)
    if show_xlabels:
        plt.xlabel("$x_1$", fontsize=14)
    else:
        plt.tick_params(labelbottom=False)
    if show_ylabels:
        plt.ylabel("$x_2$", fontsize=14, rotation=0)
    else:
        plt.tick_params(labelleft=False)
```

Floting hasil clustering dengan bounderies

[18]:
```
plt.figure(figsize=(8, 4))
plot_decision_boundaries(kmeans, X)
plt.show()
```



Mencari inertia

[20]:
```
kmeans.inertia_
```

[20]: 211.5985372581684

0.2 2. Membandingkan hasil clustering dari jumlah K berbeda

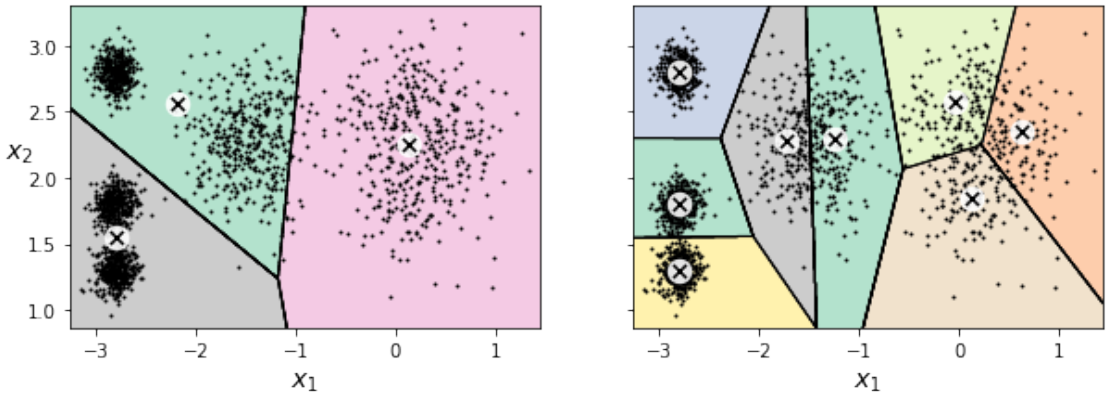
Function untuk membandingkan 2 cluster

[25]:
```
def plot_clusterer_comparison(clusterer1, clusterer2, X,␣
 ↪title1=None,title2=None):
    clusterer1.fit(X)
    clusterer2.fit(X)
    plt.figure(figsize=(10, 3.2))
    plt.subplot(121)
    plot_decision_boundaries(clusterer1, X)
    if title1:
        plt.title(title1, fontsize=14)
    plt.subplot(122)
    plot_decision_boundaries(clusterer2, X, show_ylabels=False)
    if title2:
        plt.title(title2, fontsize=14)
```

Mengbandingkan hasil proses clustering dari jumlah K (cluster berbeda)

[26]:
```
kmeans_k3 = KMeans(n_clusters=3, random_state=42)
kmeans_k8 = KMeans(n_clusters=8, random_state=42)
```

[28]:
```
plot_clusterer_comparison(kmeans_k3,kmeans_k8,X)
```



0.3 3. Menentukan jumlah K dari dari model inertia

Fitting 10 model kmeans dengan jumlah cluster dari 1 - 10

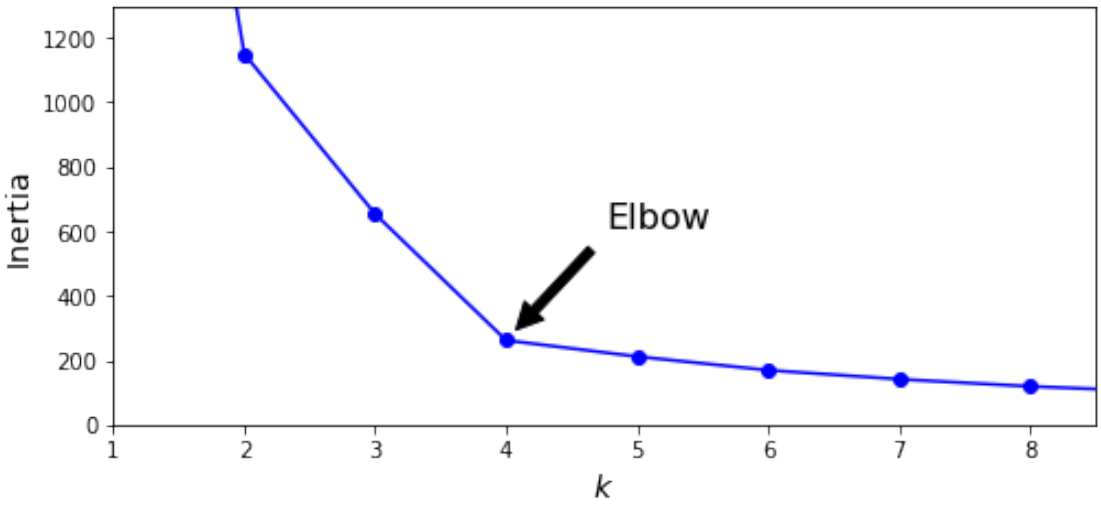
[29]:
```
kmeans_per_k = [KMeans(n_clusters=k, random_state=42).fit(X)
                for k in range(1, 10)]
```

Mendapatkan nilai inersia dari 10 Kmeant training

[31]:
```
inertias = [model.inertia_ for model in kmeans_per_k]
```

Ploting grafik jumlah k dengan nilai inersianya kmeans

[33]:
```
plt.figure(figsize=(8, 3.5))
plt.plot(range(1, 10), inertias, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Inertia", fontsize=14)
plt.annotate('Elbow',
             xy=(4, inertias[3]),
             xytext=(0.55, 0.55),
             textcoords='figure fraction',
             fontsize=16,
             arrowprops=dict(facecolor='black', shrink=0.1))
plt.axis([1, 8.5, 0, 1300])
plt.show()
```



Menentukan k yang tempat dari bentuk grafik yang seperti Elbow

0.4 4. Menentukan jumlah K dengan silhouette

Untuk menentukan SC, kita bisa gunakan fungsi silhoutte_score() pada Scikit-Learn.

[35]:
```
from sklearn.metrics import silhouette_score
silhouette_score(X, kmeans.labels_)
```

[35]:
```
0.655517642572828
```

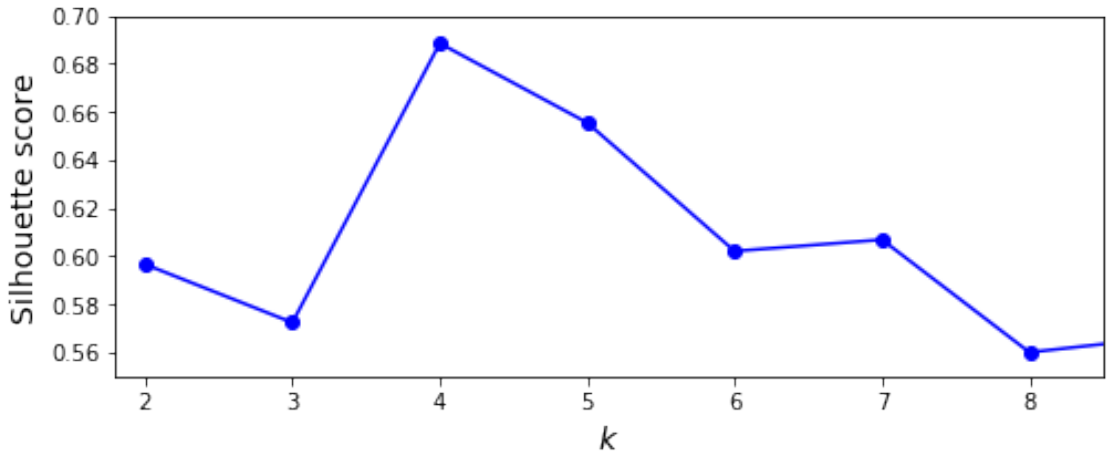
Proses awalnya sama Fitting 10 model kmeans dengan jumlah cluster dari 1 - 10

Mendapatkan nilai inersia dari 10 Kmeant training

[38]:
```
silhouette_scores = [silhouette_score(X, model.labels_)
                     for model in kmeans_per_k[1:]]
```

Ploting grafik jumlah k silhouette_scores

[41]:
```
plt.figure(figsize=(8, 3))
plt.plot(range(2, 10), silhouette_scores, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Silhouette score", fontsize=14)
plt.axis([1.8, 8.5, 0.55, 0.7])
plt.show()
```

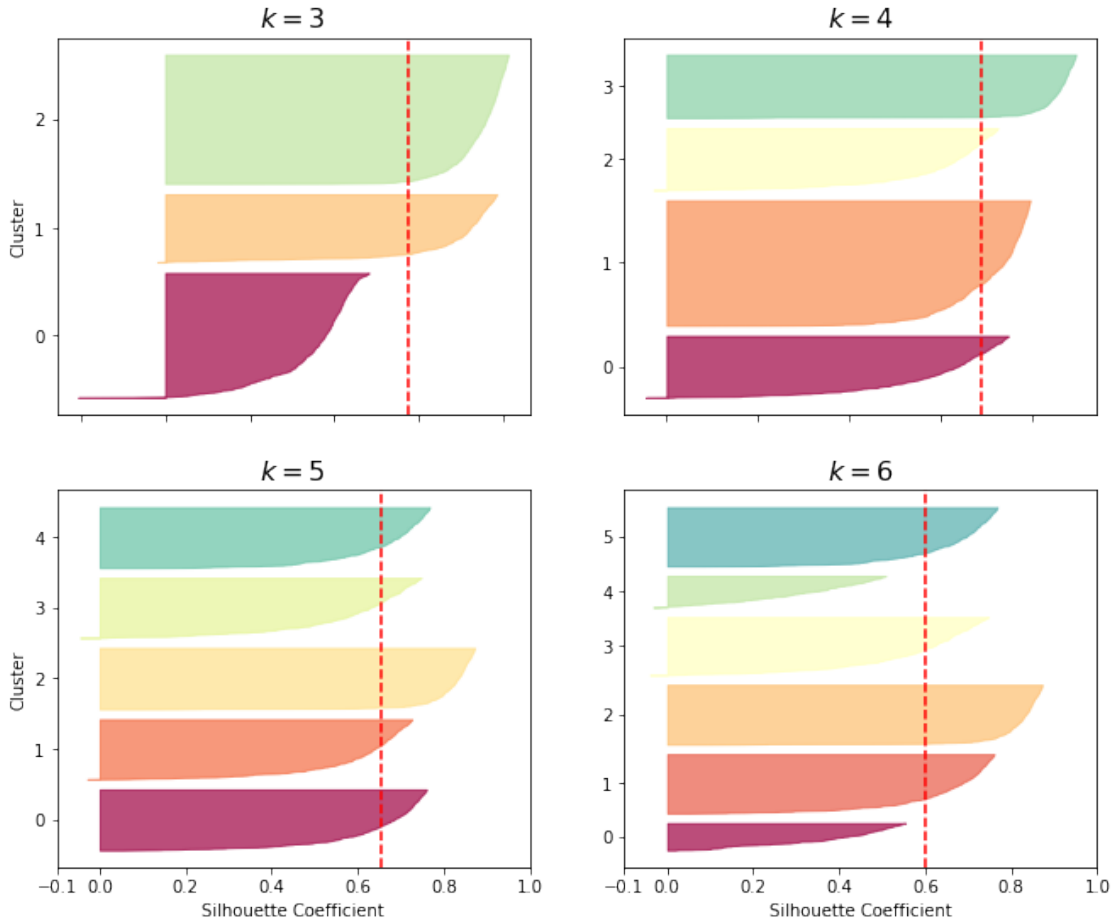


Pilih nilai K dimana nilai silhouette score mendekati 1

Pemilihan jumlah cluster dapat dilakukan berdasarkan analisis menggunakan diagram Silhoutte yang dapat dilihat pada link berikut Silhoutte Analysis. Kode program berikut adalah contoh untuk menampilkan diagram silhouette dengan jumlah cluster k bervariasi.

[43]:
```
from sklearn.metrics import silhouette_samples
from matplotlib.ticker import FixedLocator, FixedFormatter
import matplotlib as mpl
plt.figure(figsize=(11, 9))
for k in (3, 4, 5, 6):
    plt.subplot(2, 2, k - 2)
    y_pred = kmeans_per_k[k - 1].labels_
    silhouette_coefficients = silhouette_samples(X, y_pred)
    padding = len(X) // 30
    pos = padding
    ticks = []
    for i in range(k):
        coeffs = silhouette_coefficients[y_pred == i]
        coeffs.sort()
        color = mpl.cm.Spectral(i / k)
        plt.fill_betweenx(np.arange(pos, pos + len(coeffs)), 0, coeffs,
                          facecolor=color, edgecolor=color, alpha=0.7)
        ticks.append(pos + len(coeffs) // 2)
        pos += len(coeffs) + padding
    plt.gca().yaxis.set_major_locator(FixedLocator(ticks))
    plt.gca().yaxis.set_major_formatter(FixedFormatter(range(k)))
    if k in (3, 5):
        plt.ylabel("Cluster")
    if k in (5, 6):
```

```
        plt.gca().set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])
        plt.xlabel("Silhouette Coefficient")
    else:
        plt.tick_params(labelbottom=False)
    plt.axvline(x=silhouette_scores[k - 2], color="red", linestyle="--")
    plt.title("$k={}$".format(k), fontsize=16)
plt.show()
```



[ ]: