kernel-k-means-Copy1

#importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

#importing the Iris dataset with pandas

dataset = pd.read\_csv('Iris.csv')

x = dataset.iloc[:, [1, 2, 3, 4]].values

x\_sepal = dataset.iloc[:, [1, 2]].values

x\_graf = dataset.iloc[:, [0, 1, 2]].values

x\_full = dataset.iloc[:, [1, 2, 3, 4, 5]].values

from tslearn.clustering import GlobalAlignmentKernelKMeans

from tslearn.metrics import sigma\_gak

from tslearn.datasets import CachedDatasets

from tslearn.preprocessing import TimeSeriesScalerMeanVariance

seed = 0

sz = x\_sepal.shape[1]

gak\_km = GlobalAlignmentKernelKMeans(n\_clusters=3,

sigma=sigma\_gak(x\_sepal),

n\_init=20,

verbose=False,

random\_state=seed)

y\_pred = gak\_km.fit\_predict(x\_sepal)

print(gak\_km)

#plt.figure()

#for yi in range(3):

# plt.subplot(3, 1, 1 + yi)

# for xx in x\_sepal[y\_pred == yi]:

# plt.plot(xx.ravel(), "k-")

# plt.xlim(0, 1)

# plt.ylim(2, 8)

# plt.title("Cluster %d" % (yi + 1))

k = []

f = []

u = []

matriz =[]

for y in range(len(y\_pred)):

if y\_pred[y] == 0:

k.append(x\_graf[y,0])

elif y\_pred[y]==1:

f.append(x\_graf[y,0])

elif y\_pred[y]==2:

u.append(x\_graf[y,0])

print(y\_pred)

matriz.append(k)

matriz.append(f)

matriz.append(u)

print(matriz)

#Visualising the clusters

plt.scatter(x\_sepal[y\_pred == 0, 0], x\_sepal[y\_pred == 0, 1], s = 60, c = 'red', label = 'Iris-versicolour')

plt.scatter(x\_sepal[y\_pred == 1, 0], x\_sepal[y\_pred == 1, 1], s = 60, c = 'blue', label = 'Iris-setosa')

plt.scatter(x\_sepal[y\_pred == 2, 0], x\_sepal[y\_pred == 2, 1], s = 60, c = 'green', label = 'Iris-virginica')

plt.grid(b=True, which='major', axis='both', color='0.75', linestyle='-', linewidth=1)

plt.title('Kernel K-means', fontdict=None, loc='center', pad=None)

#plt.legend()GlobalAlignmentKernelKMeans(max\_iter=50, n\_clusters=3, n\_init=20, n\_jobs=None,

random\_state=0, sigma=2.1213203435596433, tol=1e-06,

verbose=False)

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 1 2 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1

2 2 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 2 1 2 2 2 2

2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 1 2 2 2 1 2 2 2 1 2 2 2 1 2

2 1]

[[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 41.0, 42.0, 43.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0], [54.0, 56.0, 58.0, 60.0, 61.0, 62.0, 63.0, 64.0, 65.0, 67.0, 68.0, 69.0, 70.0, 71.0, 72.0, 73.0, 74.0, 79.0, 80.0, 81.0, 82.0, 83.0, 84.0, 85.0, 86.0, 88.0, 89.0, 90.0, 91.0, 92.0, 93.0, 94.0, 95.0, 96.0, 97.0, 98.0, 99.0, 100.0, 102.0, 104.0, 107.0, 114.0, 115.0, 120.0, 122.0, 124.0, 127.0, 128.0, 134.0, 135.0, 139.0, 143.0, 147.0, 150.0], [51.0, 52.0, 53.0, 55.0, 57.0, 59.0, 66.0, 75.0, 76.0, 77.0, 78.0, 87.0, 101.0, 103.0, 105.0, 106.0, 108.0, 109.0, 110.0, 111.0, 112.0, 113.0, 116.0, 117.0, 118.0, 119.0, 121.0, 123.0, 125.0, 126.0, 129.0, 130.0, 131.0, 132.0, 133.0, 136.0, 137.0, 138.0, 140.0, 141.0, 142.0, 144.0, 145.0, 146.0, 148.0, 149.0]]

Out[53]:

Text(0.5, 1.0, 'Kernel K-means')

