**I. Pen-and-paper**

**E-step:**

Normalized posteriors:

Normalized posteriors:

Normalized posteriors:

**M-step:**



Normalized:

> => cluster 1

Normalized:

> => cluster 2

Normalized:

> => cluster 2

1. Being cluster 2 the larger cluster

**II. Programming and critical analysis**

1. Silhouette score for k-means with random\_state = 0 : 0.1136202757517943

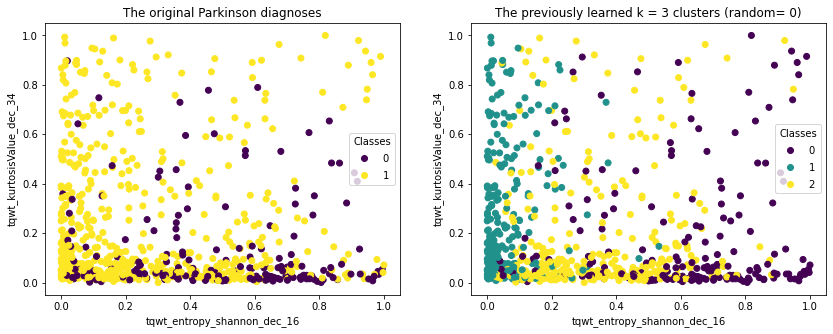
Purity score for k-means with random\_state = 0 : 0.7671957671957672

Silhouette score for k-means with random\_state = 1 : 0.11403554201377072

Purity score for k-means with random\_state = 1 : 0.7632275132275133

Silhouette score for k-means with random\_state = 2 : 0.1136202757517943

Purity score for k-means with random\_state = 2 : 0.7671957671957672

1. The non-determinism is caused by the random initialization of the centroids. The algorithm will converge to a local minimum, and because the initial centroids are different, the algorithm will converge to different local minimums.
2. 
3. 31 principal components are needed to explain more than 80% of variability.

**III. APPENDIX**

import pandas as pd, numpy as np

from scipy.io.arff import loadarff

from sklearn.preprocessing import MinMaxScaler

from sklearn.cluster import KMeans

from sklearn import metrics, cluster

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

# Load the data

data = loadarff('pd\_speech.arff')

df = pd.DataFrame(data[0])

y = pd.to\_numeric(df['class'])

df = df.drop('class', axis=1)

df\_norm = pd.DataFrame(MinMaxScaler().fit\_transform(df), columns=df.columns)

# 1)

# Purity

def purity\_score(y, y\_pred):

    confusion\_matrix = metrics.cluster.contingency\_matrix(y, y\_pred)

    return np.sum(np.amax(confusion\_matrix, axis=0)) / np.sum(confusion\_matrix)

# K-means

kmeans = []

for i in range(3):

    kmeans += [KMeans(n\_clusters=3, random\_state=i)]

    kmeans[i].fit(df\_norm)

    y\_pred = kmeans[i].labels\_

    print("Silhouette score for k-means with random\_state =", i, ":", metrics.silhouette\_score(df\_norm, y\_pred, metric='euclidean'))

    print("Purity score for k-means with random\_state =", i, ":", purity\_score(y, y\_pred))

# 3)

sorted\_by\_variance = df\_norm.var().sort\_values(ascending=False)

features = sorted\_by\_variance[:2].index

plt.figure(figsize=(14, 5))

plt.subplot(121)

scatter = plt.scatter(df\_norm[features[0]], df\_norm[features[1]], c = y)

plt.xlabel(features[0])

plt.ylabel(features[1])

plt.legend(\*scatter.legend\_elements(), loc="best", title="Classes")

plt.title('The original Parkinson diagnoses')

plt.subplot(122)

scatter = plt.scatter(df\_norm[features[0]], df\_norm[features[1]], c=kmeans[0].labels\_)

plt.xlabel(features[0])

plt.ylabel(features[1])

plt.legend(\*scatter.legend\_elements(), loc="best", title="Classes")

plt.title('The previously learned k=3 clusters (random=0)')

plt.show()

# 4) PCA

pca = PCA(n\_components=0.8)

pca.fit(df\_norm)

print(pca.n\_components\_, "principal components are needed to explain more than 80% of variability.")

**END**