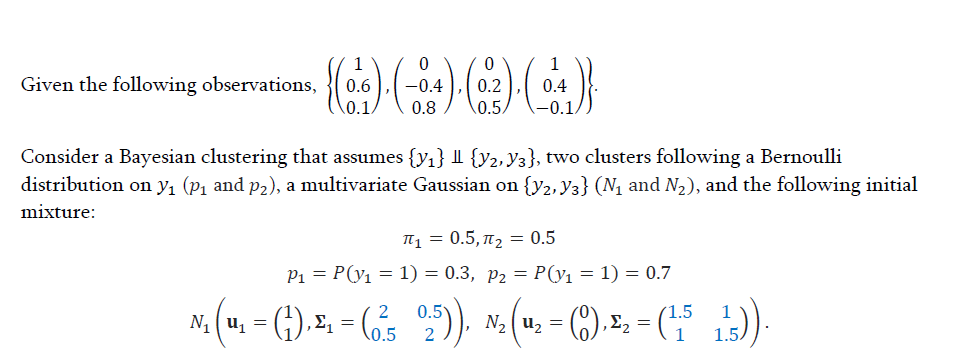
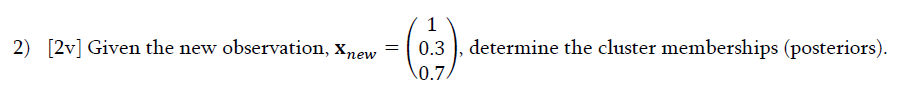
* 1. **Pen-and-paper** [12v]



Note: you can solve this exercise by neglecting 𝑦1 and still scoring up to 70% of its grade.

1) [6v] Perform one epoch of the EM clustering algorithm and determine the new parameters. Hint: we suggest you to use numpy and scipy, however disclose the intermediary results step by step.



3) [2.5v] Performing a hard assignment of observations to clusters under a ML assumption, identify the silhouette of the larger cluster under a Manhattan distance.

4) [0.5v] Knowing the purity of the clustering solution is 0.75, identify the number of possible classes (ground truth).

# Programming and critical analysis [8v]

Recall the column\_diagnosis.arff dataset from previous homeworks. For the following exercises, normalize the data using sklearn’s MinMaxScaler.

1) [4v] Using sklearn, apply k-means clustering fully unsupervisedly on the normalized data with 𝑘 ∈ {2,3,4,5} (random=0 and remaining parameters as default). Assess the silhouette and purity of the produced solutions.

import pandas as pd

import numpy as np

from scipy.io.arff import loadarff

from sklearn.preprocessing import MinMaxScaler

from sklearn.cluster import KMeans

from sklearn.metrics.cluster import contingency\_matrix

from sklearn.metrics import silhouette\_score

# Read ARFF file

data = loadarff('column\_diagnosis.arff')

df = pd.DataFrame(data[0])

df['class'] = df['class'].str.decode('utf-8')

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

y = df['class']

X\_scaled = MinMaxScaler().fit\_transform(X)

k\_values = [2, 3, 4, 5]

for k in k\_values:

    # Perform K-means clustering

    kmeans = KMeans(n\_clusters=k, random\_state=0).fit(X\_scaled)

    if (k==3):

        kmeans3 = kmeans

    # Calculate silhouette scores and purity scores

    s\_score = silhouette\_score(X\_scaled, kmeans.labels\_)

    confusion\_matrix = contingency\_matrix(y, kmeans.labels\_)

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

    print('For K =', k)

    print('Silhouete score:', s\_score)

    print('Purity score:', p\_score, '\n')

Output:

For K = 2

Silhouete score: 0.36044124340441114

Purity score: 0.632258064516129

For K = 3

Silhouete score: 0.29579055730002257

Purity score: 0.667741935483871

For K = 4

Silhouete score: 0.27442402122340176

Purity score: 0.6612903225806451

For K = 5

Silhouete score: 0.23823928397844843

Purity score: 0.6774193548387096

2) [2v] Consider the application of PCA after the data normalization:

i. Identify the variability explained by the top two principal components.

from sklearn.decomposition import PCA

pca = PCA(n\_components=2)

principal\_components = pca.fit(X\_scaled)

# Compute the explained variance for the top two components

explained\_variance = pca.explained\_variance\_ratio\_

print('Explained variance of the top two components:', explained\_variance)

Output:

Explained variance of the top two components: [0.56181445 0.20955953]

ii. For each one of these two components, sort the input variables by relevance by inspecting the absolute weights of the linear projection.

# Get the absolute weights

component\_weights = abs(pca.components\_)

# Sort the input variables by relevance

relevance\_df = pd.DataFrame(component\_weights, columns=X.columns, index=['Component 1', 'Component 2']).T

relevance\_df = relevance\_df.sort\_values(by=['Component 1', 'Component 2'], ascending=False)

print('Sorted relevance of input variables in the top two components:')

print(relevance\_df)

Output:

Sorted relevance of input variables in the top two components:

Component 1 Component 2

pelvic\_incidence 0.591621 0.100037

lumbar\_lordosis\_angle 0.515085 0.080047

pelvic\_tilt 0.467039 0.670373

sacral\_slope 0.325689 0.443303

degree\_spondylolisthesis 0.216930 0.004583

pelvic\_radius 0.115824 0.581074

3) [2v] Visualize side-by-side the data using: i) the ground diagnoses, and ii) the *previously* learned 𝑘 = 3 clustering solution. To this end, projected the normalized data onto a 2-dimensional data space using PCA and then color observations using the reference and cluster annotations.

import matplotlib.pyplot as plt

# Reduce the dimensionality of data

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled)

# Real labels as inetegers to compare with kmeans labels

codes = {'Hernia':0, 'Normal':1, 'Spondylolisthesis':2}

y\_pred = y.map(codes).tolist()

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

plt.subplot(121)

plt.scatter(X\_pca[:,0], X\_pca[:,1], c=y\_pred)

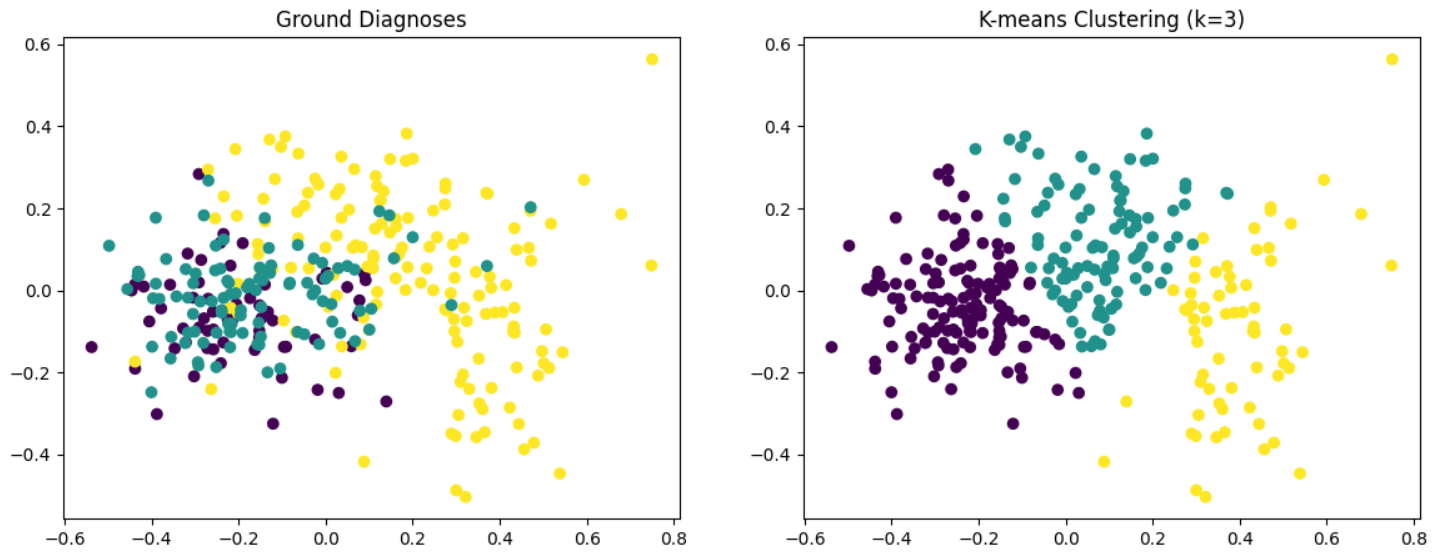
plt.title('Ground Diagnoses')

plt.subplot(122)

plt.scatter(X\_pca[:,0], X\_pca[:,1], c=kmeans3.labels\_)

plt.title('K-means Clustering (k=3)')

plt.show()



4) [1v] Considering the results from questions (1) and (3), identify two ways on how clustering can be used to characterize the population of ill and healthy individuals.

Answer: Clustering can be used to identify groups of patients with similar medical characteristics to discover potential causes of the disease or define groups at higher risk of developing the disease.

# END