FTML practical session 12: 2023/06/08

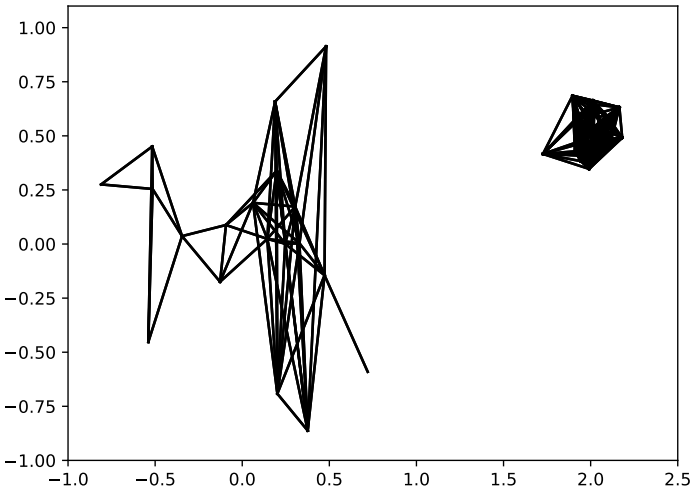

TABLE DES MATIÈRES

INTRODUCTION

The goal of this session is to manipulate different distance and similarities, and to apply them to a clustering problem, and a graph problem. The message to take away is that the choice of the metric used has important consequences on the processing made afterwards.

1 BUILDING COMPATIBILITY GRAPHS

1.1 Simple geometric data

Even for geometric (and thus numerical data), the classical euclidean distance is not the only available metric. If we take a look at the documention of **cdist** from scipy or **numpy.linalg.norm**, we see that many metrics exist.

https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cdist.html

https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html

Use the notebook **similarities/build _graphs_geometric_data.ipynb**. in order to build compabtility graphs for the data contained in **data/data.npy** (displayed in figure 1), in order to obtained the graphs shown in figures 2 to 6. You will need to choose the right **metric** for each graph. Try to think of the metric only mentally **before** implementing it !

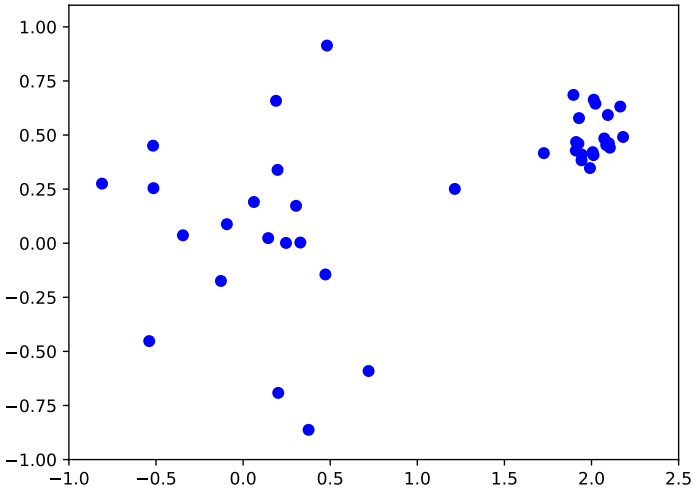

FIGURE 1 – The data to build compabtility graphs from.

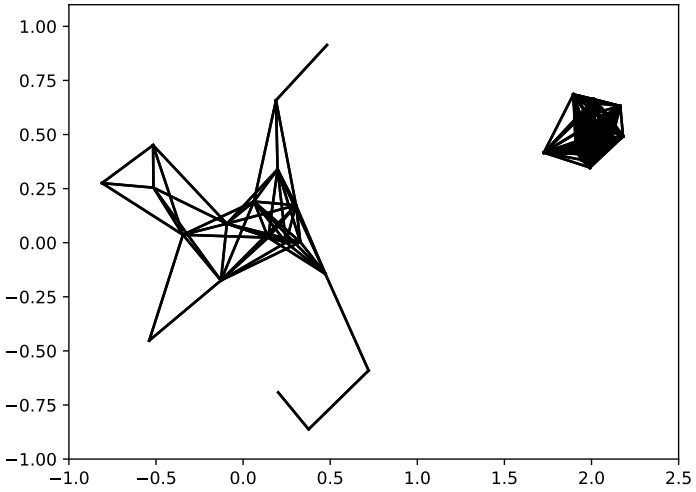

FIGURE 2 – Graph 1

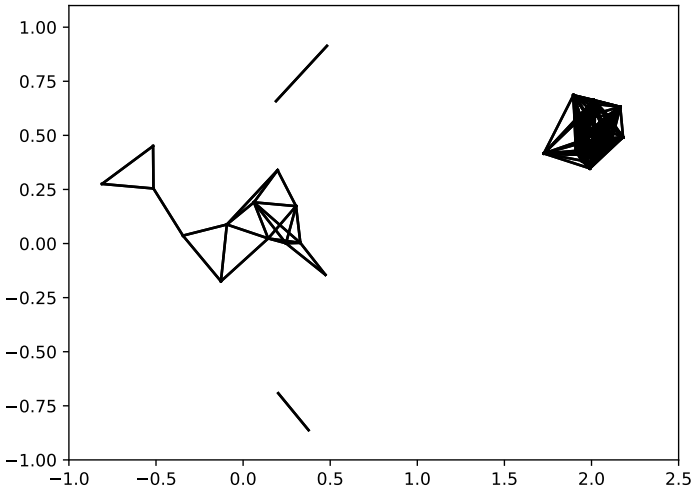

FIGURE 3 – Graph 2

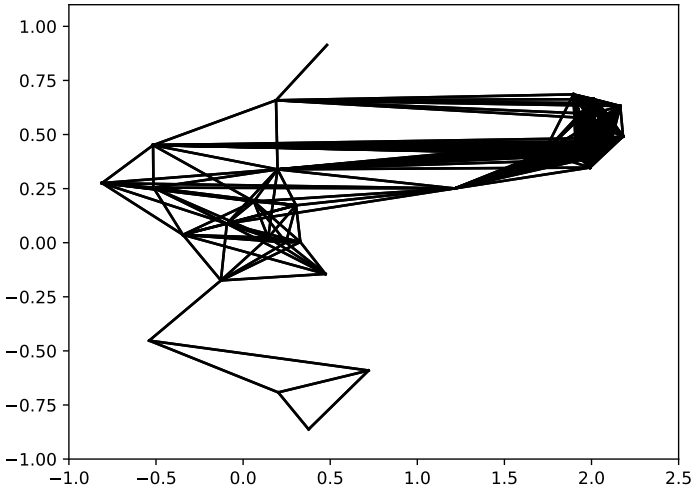

FIGURE 4 – Graph 3

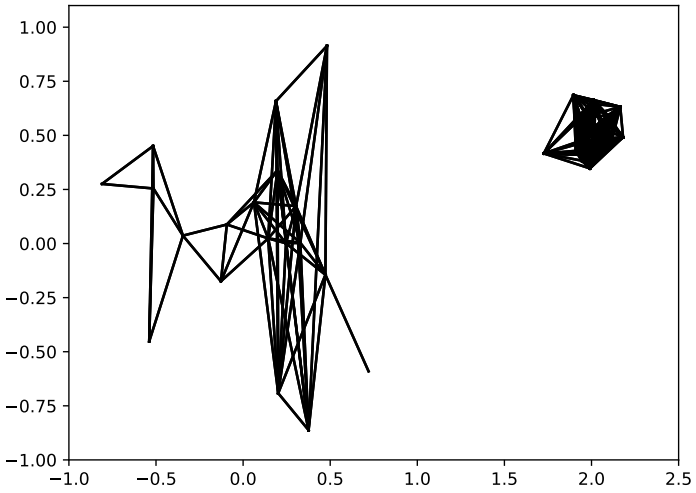

FIGURE 5 – Graph 4

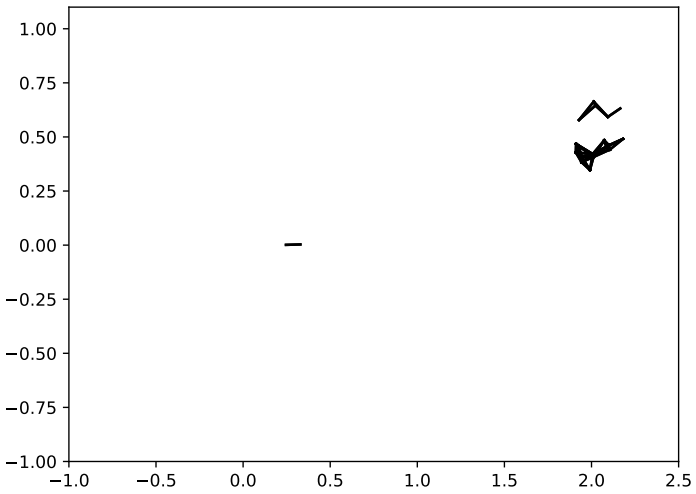

FIGURE 6 – Graph 5

1.2 Hybrid data

Same instructions, but this time the data are not only numerical and also contain categorical attributes.

Figures : 7, 8, 9, 10, 11, 12,
Folder : **similarities/hybrid _data/**
Data : **similarities/hybrid _data/hybrid_data.csv**
Notebook : **similarities/hybrid _data/build_graphs_hybrid_data.ipynb**
To edit : function **compute_dissimilarity()** and **THRESHOLD** in the last cell.

FIGURE 7 – Graph 1

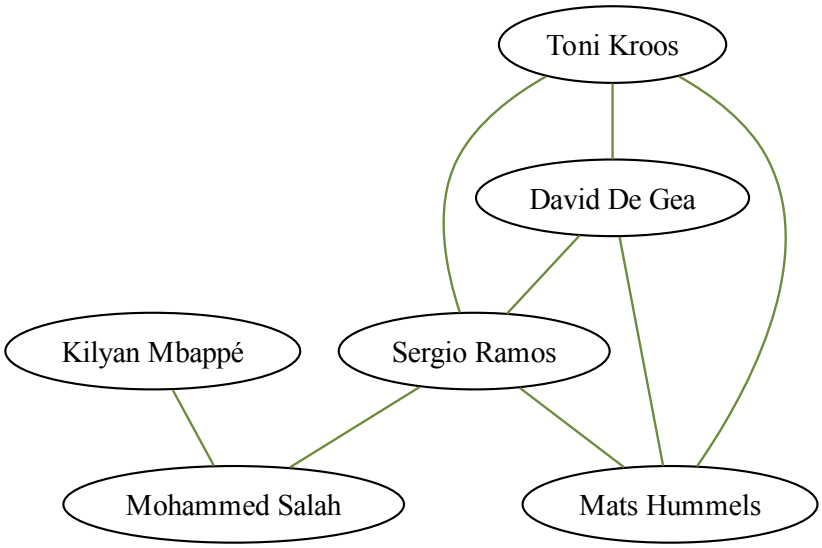

FIGURE 8 – Graph 2

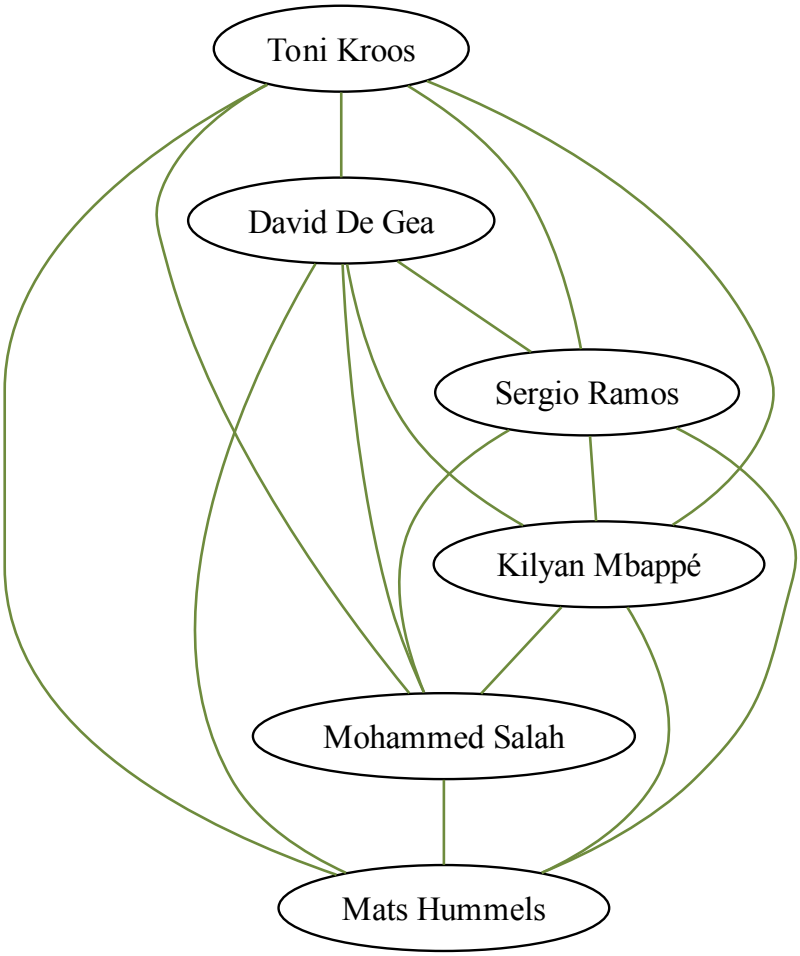

FIGURE 9 – Graph 3

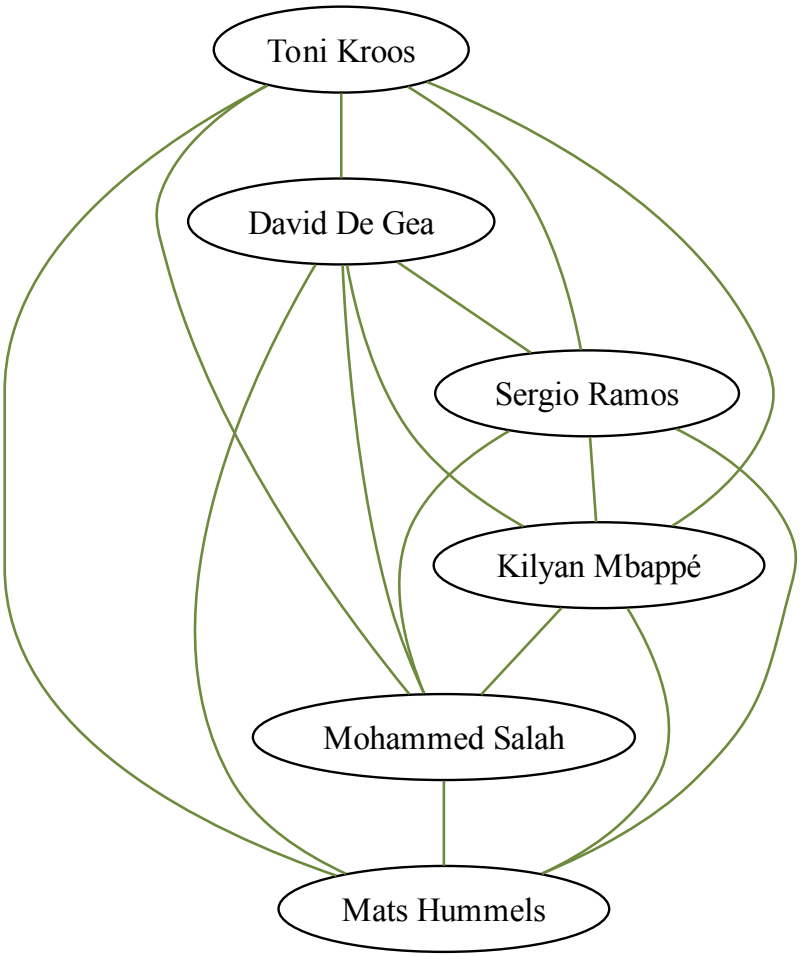

Figure 10 – Graph 4

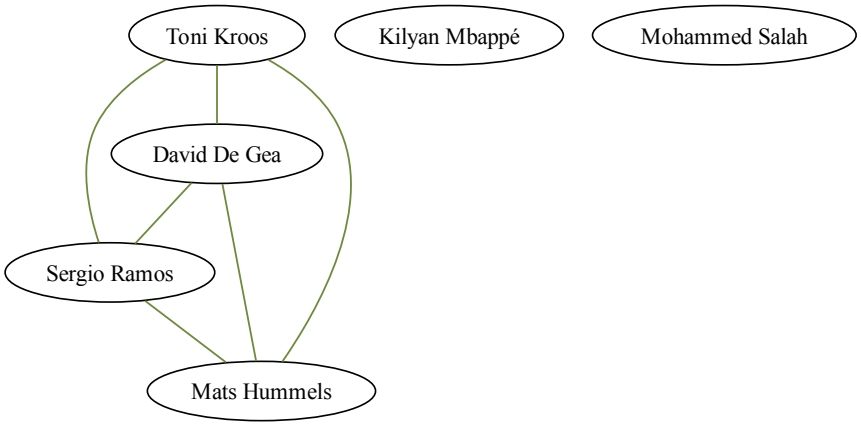

Figure 11 – Graph 5

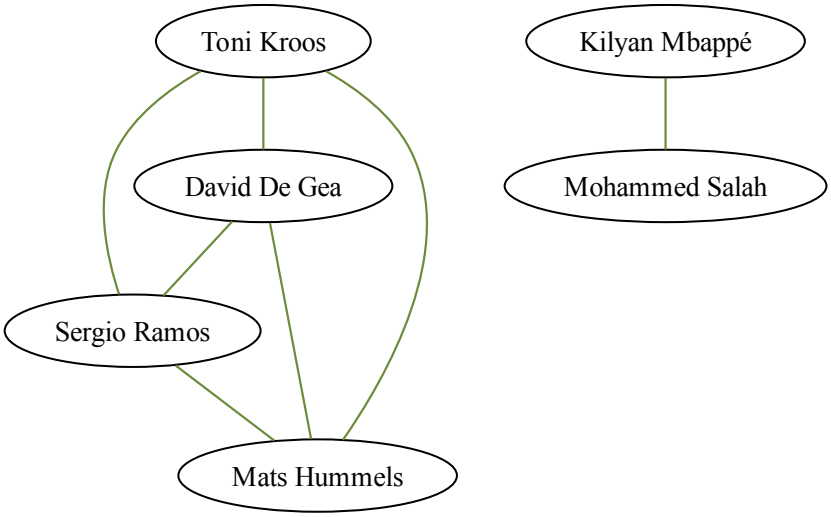

FIGURE 12 – Graph 6

2 SPECTRAL CLUSTERING

Perform a spectral clustering of the data contained in **spectral _clustering/ data.npy**. You will need to :

— define a similarity matrix for the data (you can try several similarities, or use a similarity suggested during the class).

— apply a Spectral clustering to this similarity matrix

— evaluate the relevant number of clusters by using the knee heuristic applied to the normalized cut score.