Neural and Evolutionary Computation (NEC)

A3: Unsupervised learning with PCA, t-SNE, k-means, AHC and SOM

Report

Sergi Salido Cubero
sergi.salido@estudiants.urv.cat

Yanbing Zhu
yanbing.zhu@estudiants.urv.cat

Contents

Description of the implementation (languages, tools used, etc.)

To code all the implementations we have used **Jupyter Notebooks** (IPYNB) and **Python**.

We have used multiple libraries (sklearn, Somoclu, etc.) that implement the unsupervised learning algorithms to solve the exercises. We have also used multiple data visualization libraries like matplotlib, seaborn, plotly.

We have used VS Code as IDE in Windows and Linux.

Execution instructions

PCA execution

Open the folder in VS Code on Windows/Linux.

Run **PCA_d1.ipynb** file.

Run **PCA_d2.ipynb** file.

t-SNE execution

Open the folder in VS Code on Windows/Linux.

Run **t-SNE_d1.ipynb** file.

Run **t-SNE_d2.ipynb** file.

k-means execution

Open the folder in VS Code on Windows/Linux.

Run **k-means_d1.ipynb** file.

Run **k-means_d2.ipynb** file.

AHC execution

Open the folder in VS Code on Windows/Linux.

Run **AHC_d1.ipynb** file.

Run **AHC_d2.ipynb** file.

SOM execution

Open the folder in VS Code on Linux since Somoclu has problems to be executed on Windows.

Run **SOM_d1.py** file.

Run **SOM_d2.py** file.

Implementation decisions

We followed all the recommendations from the Dr.

To implement SOM we have used Somoclu library for Python. We had problems with this library on Windows so we executed the code on a Linux virtual machine.

More explanations, details and the references can be found on the notebooks.

Description and link to the selected dataset

We have selected the **Wine Customer Segmentation** dataset from Kaggle.

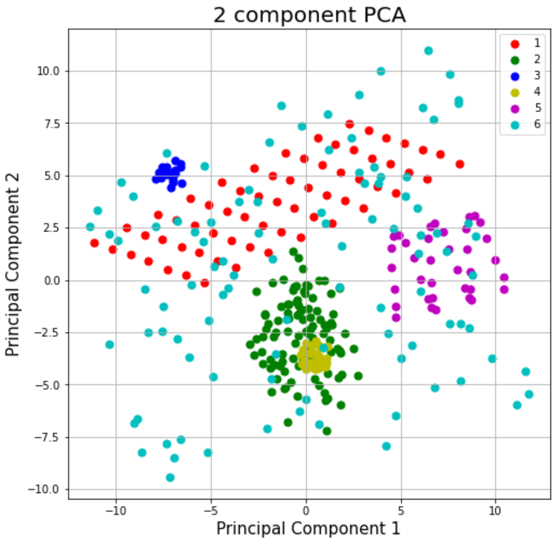
Description: These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

Link: https://www.kaggle.com/sadeghjalalian/wine-customer-segmentation

Unsupervised learning results, including plots

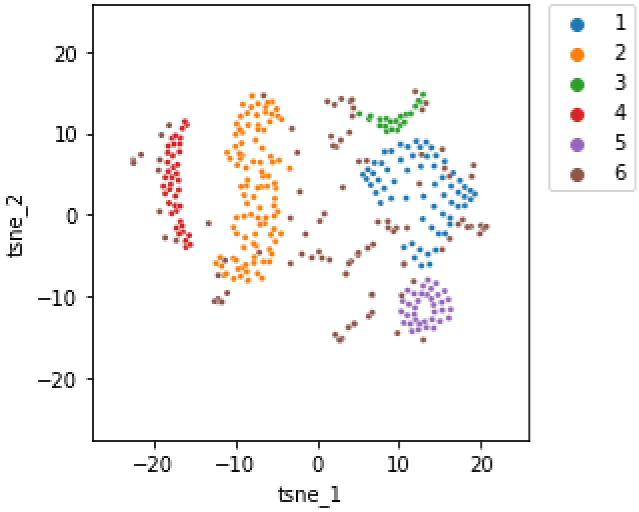
PCA results, Database 1

2D projection (2 component PCA)



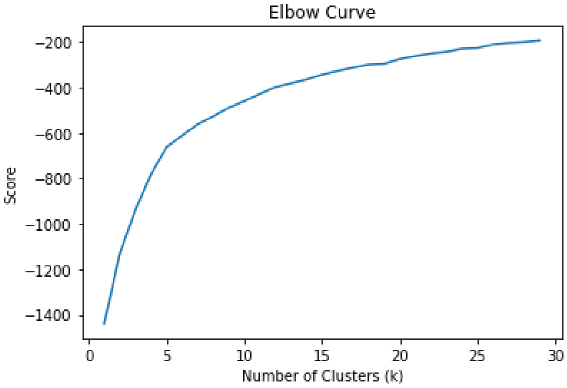
t-SNE results, Database 1

t-SNE projection to 2D

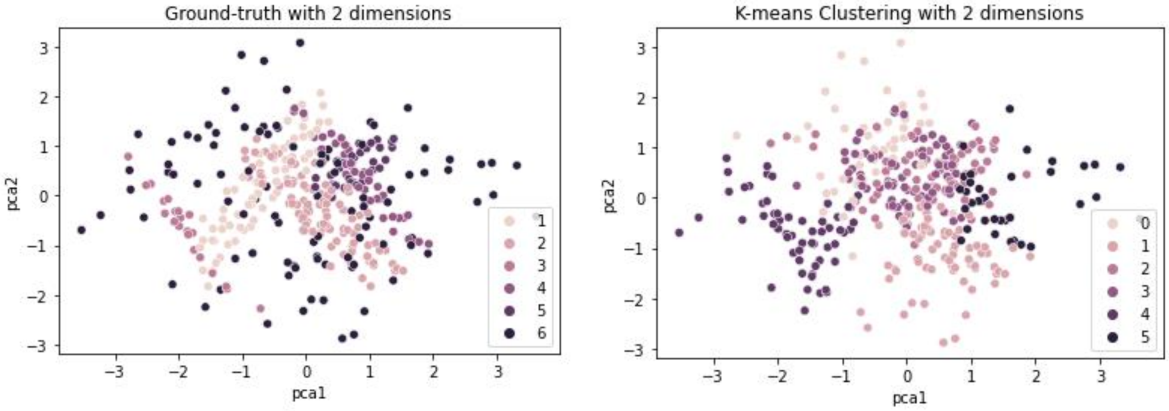


k-means results, Database 1

Elbow curve:

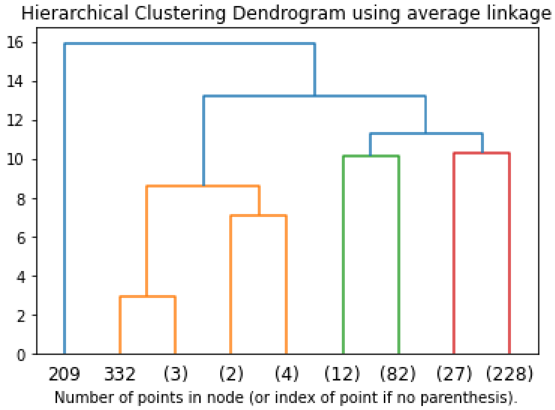


2D projection: Ground-truth and K-means clustering with 2 dimensions.

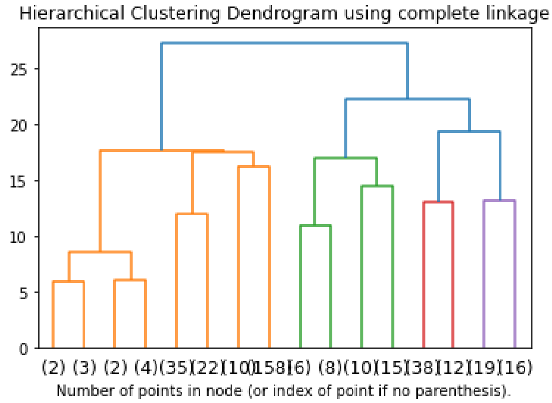


AHC results, Database 1

Hierarchical Clustering Dendrogram using average linkage

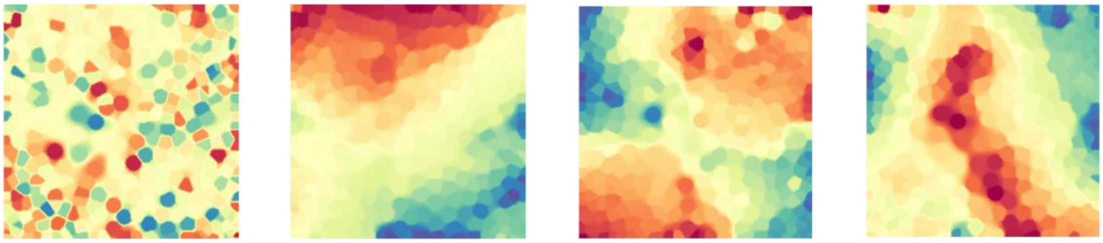


Hierarchical Clustering Dendrogram using complete linkage

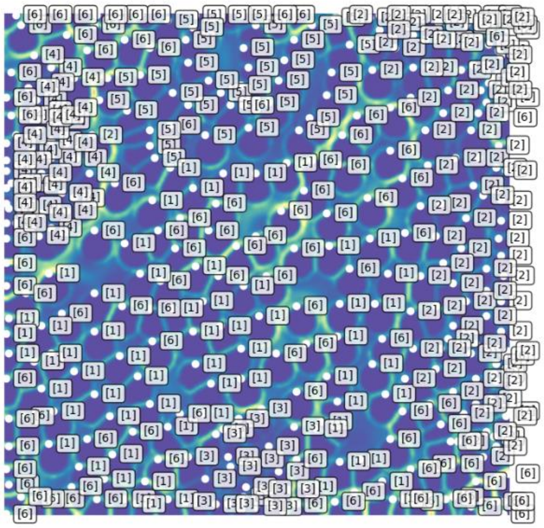


SOM results, Database 1

Planar map – Component planes

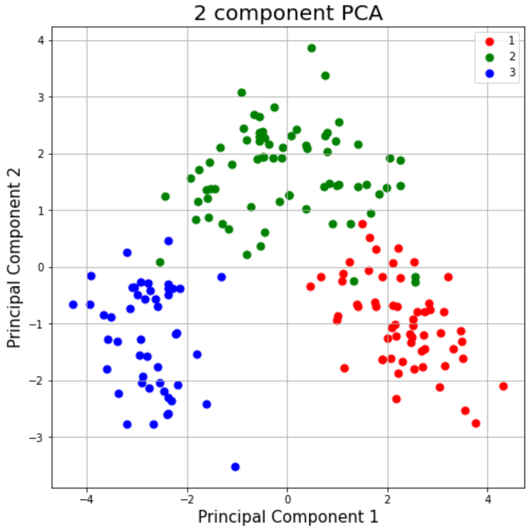


Planar map – Umatrix



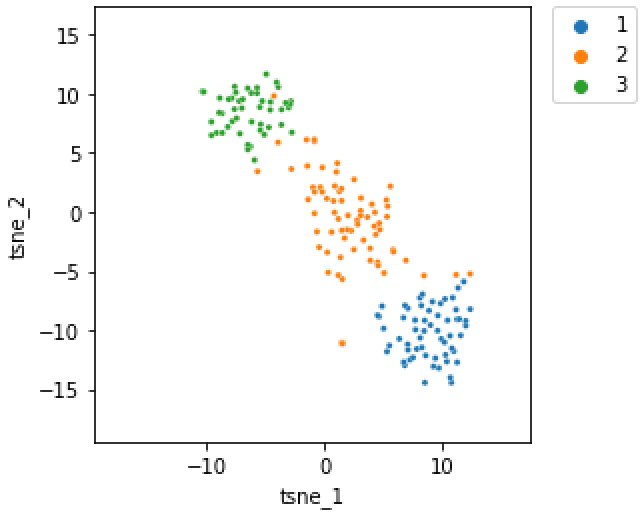
PCA results, Database 2

2D projection (2 component PCA)



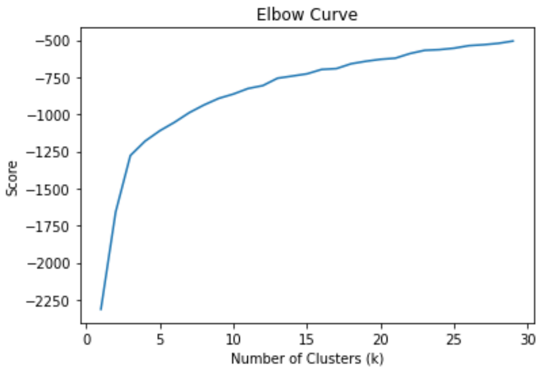
t-SNE results, Database 2

t-SNE projection to 2D

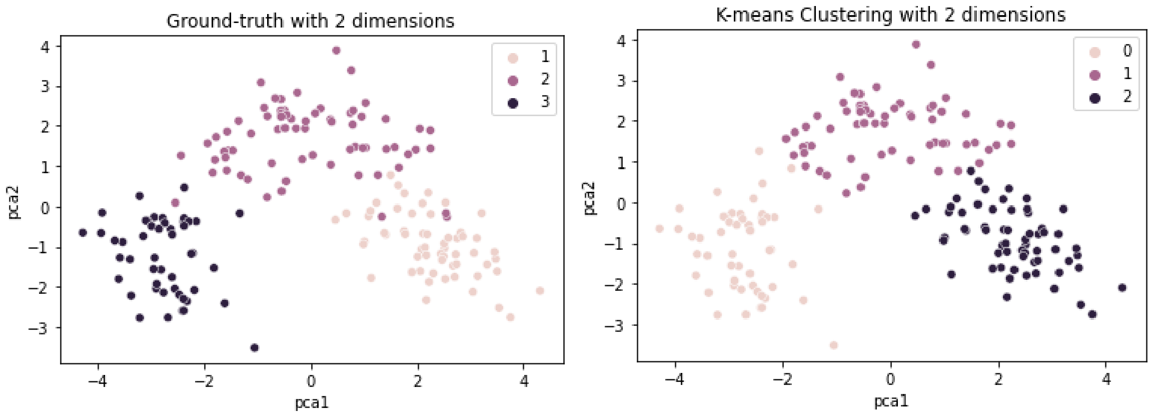


k-means results, Database 2

Elbow curve:

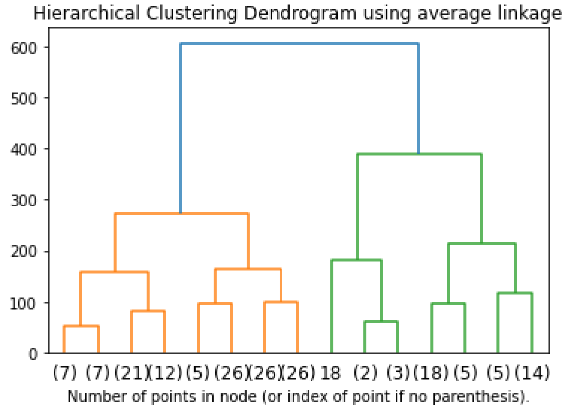


2D projection: Ground-truth and K-means clustering with 2 dimensions.

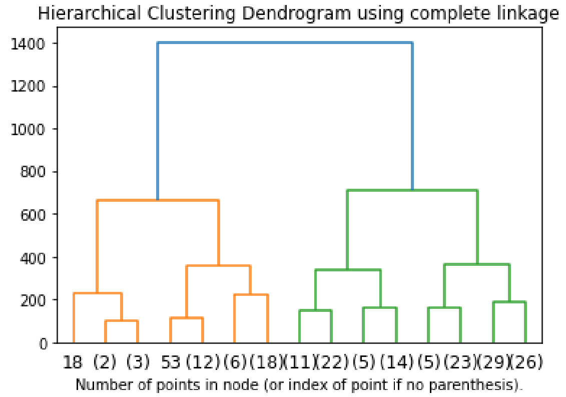


AHC results, Database 2

Hierarchical Clustering Dendrogram using average linkage

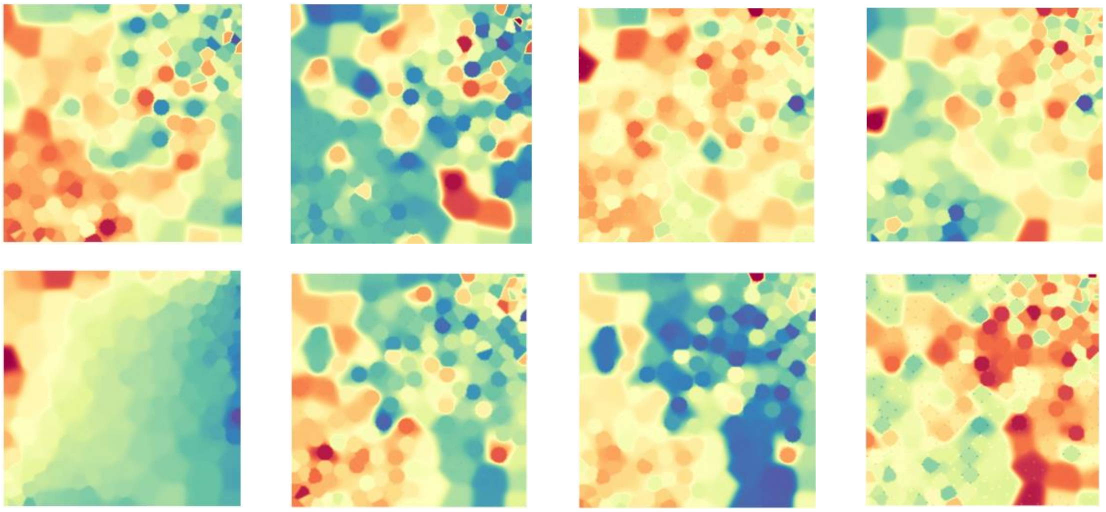


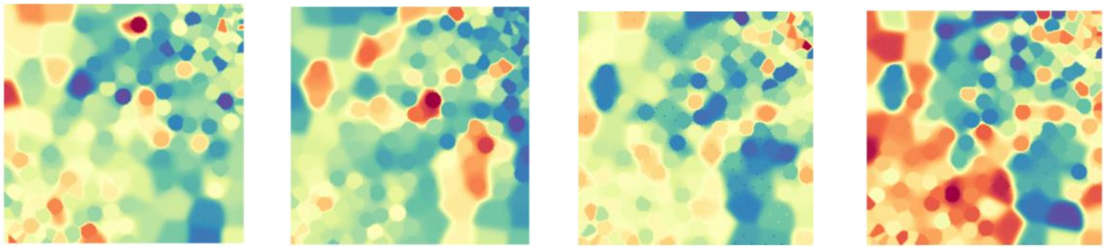
Hierarchical Clustering Dendrogram using complete linkage



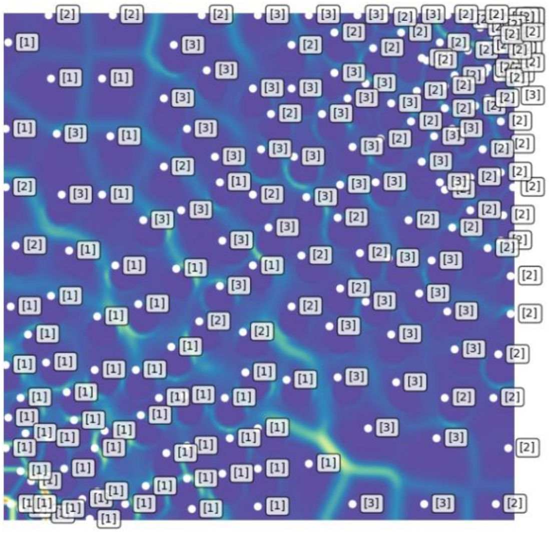
SOM results, Database 2

Planar map – Component planes

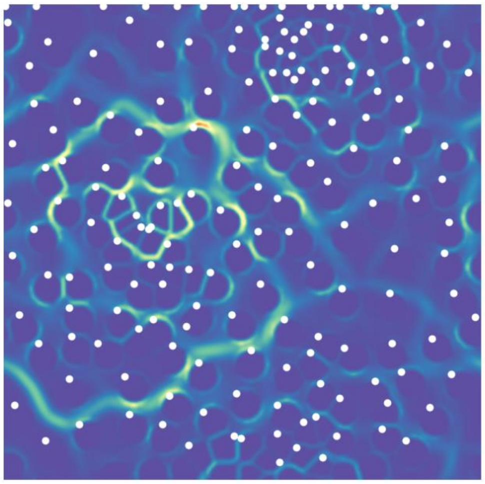




Planar map – Umatrix



Toroid – Umatrix



Discussion and interpretation of the results

Note: In the Python Notebooks there are comments and explanations of each step of the code.

All these techniques are very useful to have a good idea in the exploratory data analysis step in Data Science about which attributes have the most relevance, once we know that is much easier to analyze our data and understand the results.

The main difference between PCA and T-SNE is that PCA is a linear algorithm, so it will only find linear dependencies or relationships in the data. In this way, it will ignore all kinds of complex polynomial relationships, while T-SNE will be able to highlight them.

The K-Means algorithm clusters data by trying to separate samples in n groups of equal variances.

The agglomerative clustering groups objects in clusters based on their similarity. The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects.

Finally, SOM produces a low-dimensional representation of a higher dimensional data set while preserving the topological structure of the data.