# CSC-691 — Data Mining
# Assignment 5

Esteban Murillo Burford

Saturday 9th November, 2019

## Summary

For this assignment, everything related to clustering was implemented successfully. As it can be seen on Table 1, all yielded times are similar, except for the customized version of the *k-Means* algorithm, which takes longer to execute. It is worth noting that all of these algorithms are using the same parameters. Also, for the first version of the *DBSCAN*, all elements were classified as noise, that is why, the first version of this algorithm was not taken into consideration. There is an output file for it, however. After some experimentation with the algorithm in question, it was discovered that changing the ***EPS*** variable to 25 modified the output of the algorithm and it actually produced 3 different clusters. Moreover, the amount of clusters found by *DBSCAN* is what is used as input for all other algorithms, and they in fact, produce the same amount of clusters (3).

Finally it is worth mentioning that the last three clustering algorithms (*k-Means (custom), k-Means (scikit-learn) and Agglomerative (scikit-learn)*) all have very similar results in term of the elements that end up together. The only exception to this is the *DBSCAN* algorithm that change things a little bit and actually groups different elements together.

## Time comparison

| **Name of algorithm** | **Time (s)** |
|---|---|
| *DBSCAN (EPS = 25)* | 0.0259 |
| *k-Means (custom)* | 0.2593 |
| *k-Means (scikit-learn)* | 0.0060 |
| *Agglomerative (scikit-learn)* | 0.0588 |

Table 1: Yielded times for all algorithms. Timing was done using 922 files and 40 features

# Matrix breakdown

Below is an attempt to generate a heat map to visualize in a better way which files ended with which. As it can be seen, there are very few cases in which red can be seen (the elements were never clustered together). This means that there is some discrepancy between the algorithms, in other words, elements end up groped within different clusters for the algorithms. In an ideal scenario, we would have more red cells, meaning that a "concensus was reached" from all algorith. As mentioned in the previous section, this is due to the *DBSCAN* algorithm that adds some noise to the results. It is important to mention that this is a reduced version, using around 210 files to generate the heat map. To see the final results run the program and check the generated matrix in the 'output' folder.

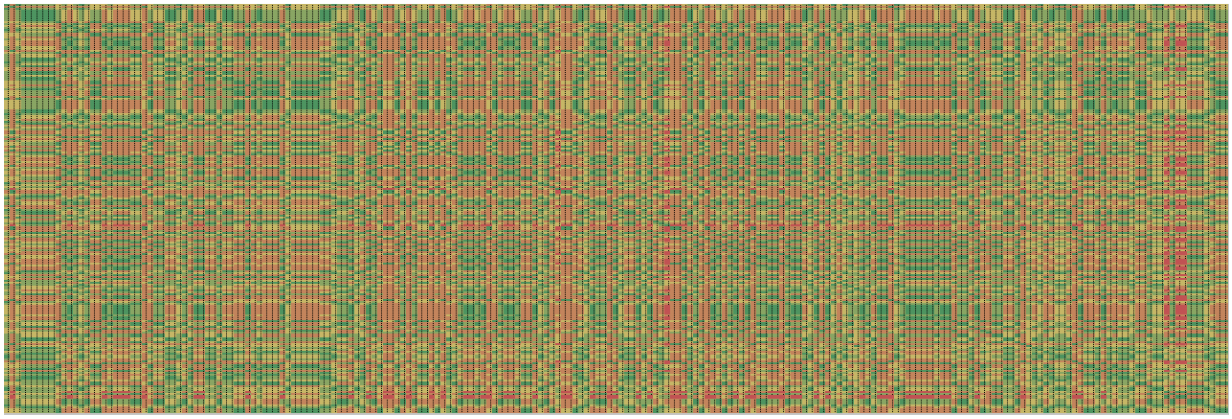

Figure 1: Heat map color code: red = never clustered together, green = always clustered together

# Notes

- For a matter of simplicity, all 40 features of all 922 files are provided with the 'sound_features.csv' file. If you wish to have the program read different values, delete the file and change the respective variable in 'config.py' accordingly
- To modify the behavior of the program, change values in *config.py* accordingly
- To see the values of the matrix, check the file in 'output/matrix.csv'
- Remember to run code using the **version 3 of the *Python* interpreter**