# Assignment 5 Report

## CAP4770 Spring 2021

## Professor Dobra

---


## Miguel Caputo

## Preprocessing

First, I read the CSV file into a data frame, then I encoded all the 5 categorical columns using the function LabelEncoder() from ScikitLearn to make the values numbers. After encoding them I normalized the whole data frame by dividing all the columns by their max value. I finalized the preprocessing step by converting the data frame into an array.

## K-means

### How you set the parameters

For the parameters, I had to pass the data in the form of a k x n matrix where k was the number of dimensions and n the number of different data points. I also decided to use 10 clusters after looping through a lot of values that was the number of clusters that gave me better results.

### Best score (according to the scoring method of each algorithm)

A smaller Total Cost is always better

The Silhouette Coefficient has a range of -1 to 1, -1 being the worst possible accuracy and 1 being the best

**Total Cost:** 8980.980811368008

**Silhouettes Coefficient:** 0.14767512823988224

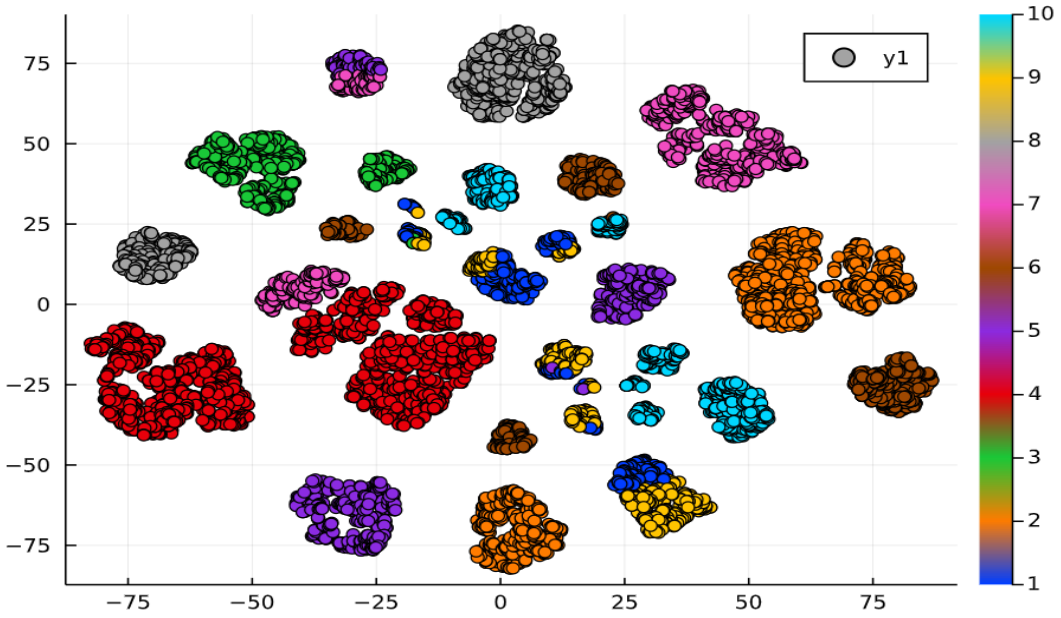
### Running time

3.375848 seconds

### Some interpretation of the clusters found

The K-means clustering algorithm essentially focuses on finding centroids (by averaging the data) for each cluster and then grouping all the data points that are close to that centroid into that specific cluster. The problem with K-means is that it is not fitting for our data for the reasons in the following paragraph.

In the bellow graph (graph of our Kmeans clusters), we can observe that most of our clusters are spread out like cluster 8 (in grey) and cluster 10 (in light blue), this makes sense because our clusters are of varying sizes and density which greatly affects the performance of the algorithm. Also, the outliers drag the centroids away from where they should be located which largely impacts how the clusters spread out. Lastly, our data contains a lot of dimensions (27) which affects how the algorithm predicts the centroids.




Due to all these mentioned reasons, it is not a surprise that Kmeans is one of the algorithms with the worse performance that I used for this assignment.

**Identify 3 outliers (based on the model provided by each clustering algorithm).**

1. Outlier was detected in cluster number 9, at point number 1318 with a silhouette coefficient of -0.08993843282911018

2. Outlier was detected in cluster number 9, at point number 4543 with a silhouette coefficient of -0.07422749124889305

3. Outlier was detected in cluster number 9, at point number 6509 with a silhouette coefficient of -0.0696821051093548

**Explain how you found the outliers**

I first found all the silhouette coefficients using the silhouette function **silhouettes(Kmeans.assignments, Kmeans.counts, dist),** then I found the three most negative values. Because the silhouette coefficients determine how similar a point is to its cluster, the most negative values must be the outliers.

## K-medoids

**How you set the parameters**

For the parameters, I had to pass the data in the form of a n x n distance matrix where n is the number of different data points. I also decided to use 10 clusters after looping through a lot of values that was the number of clusters that gave me better results.

**Best score (according to the scoring method of each algorithm)**

A smaller Total Cost is always better

The Silhouette Coefficient has a range of -1 to 1, -1 being the worst possible accuracy and 1 being the best

**Total Cost:** 9674.043543643125

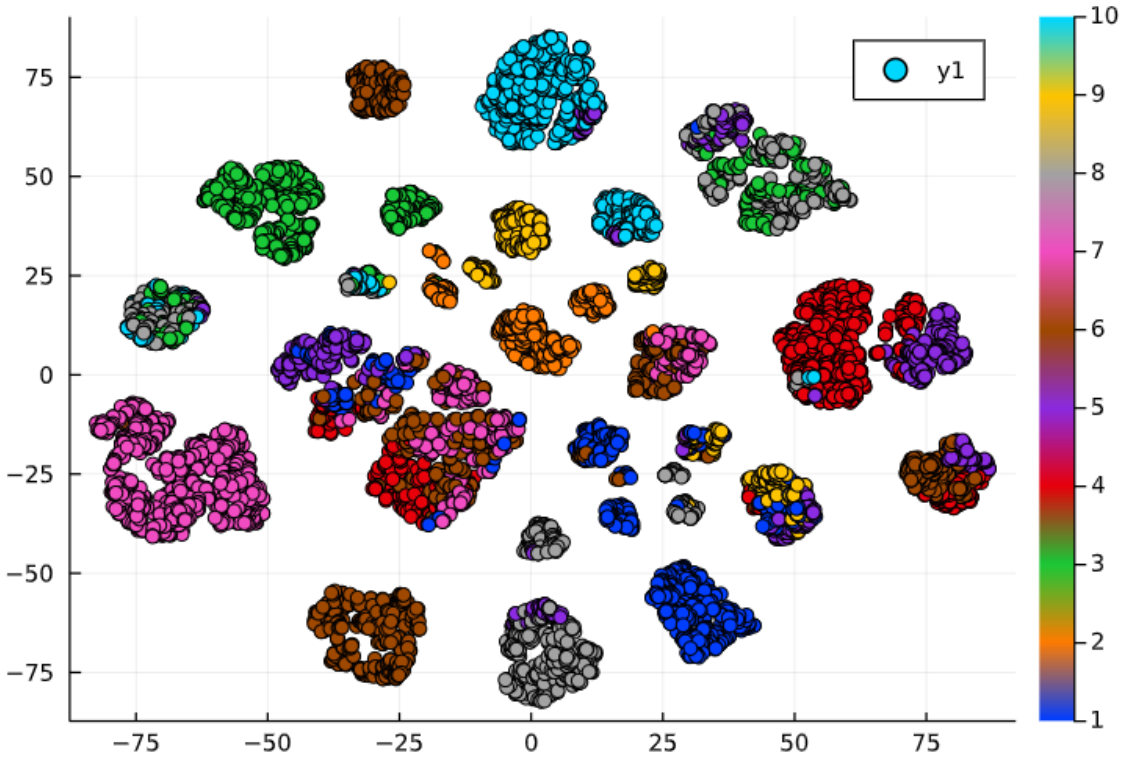
**Silhouettes Coefficient:** 0.11029004019660318

**Running time**

0.755929 seconds

**Some interpretation of the clusters found**

Kmedoids is an algorithm kind of similar to Kmeans in the sense that both focus on “centers” this time called medoids, each medoid is an actual point in our data that was chosen to the algorithm to group points to each cluster, the difference is that k medoids focus more on distances than on points.




In this graph  (graph of our K Medoids clusters), we can observe that most of our clusters are mixed between them and there is not much order, again this is due to our data set not being perfect and having a lot of outliers, Kmedoids is an algorithm that is affected by outliers and we can see that in the results. Some interesting things I could notice while inspecting the chosen medoids is that none of them contain

current smokers, also 9/10 medoids have no “FamilyHx" and only some of them had had remissions, this is logical because this is the most common data in our dataset which is going to be the more centric ones, therefore, being the best ones as medoids.

By looking at the graph we can observe that our k medoids are the worst algorithm that I used for the assignment, but this is mostly because of how our data is presented and how many outliers we have.

**Identify 3 outliers (based on the model provided by each clustering algorithm).**

1. Outlier was detected in medoid number 1, at point number 5328 with a silhouette coefficient of -0.25099556757009145

2. Outlier was detected in medoid number 1, at point number 4724 with a silhouette coefficient of -0.24166391205965643

3. Outlier was detected in medoid number 1, at point number 5343 with a silhouette coefficient of -0.23581903246817082

**Explain how you found the outliers**

I first found all the silhouette coefficients using the silhouette function **silhouettes(Kmedoids.assignments, Kmedoids.counts, dist),** then I found the three most negative values. Because the silhouette coefficients determine how similar a point is to its cluster, the most negative values must be the outliers.

## Dbscan

**How you set the parameters**

For the parameters, I had to pass the data in the form of a n x n distance matrix where n is the number of different data points. I am also passing a radius of 0.99, and a minimum number of neighbors of 4 which I found to be the best parameters.

**Best score (according to the scoring method of each algorithm)**

The Silhouette Coefficient has a range of -1 to 1, -1 being the worst possible accuracy and 1 being the best

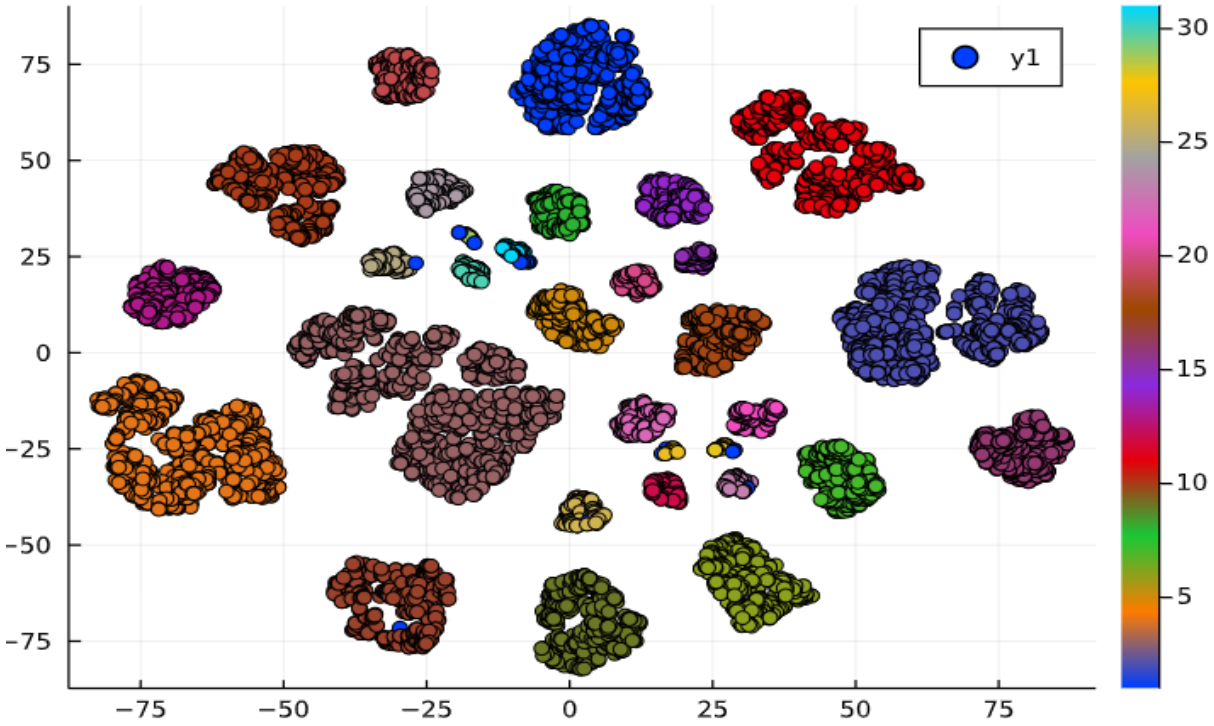
**Silhouettes Coefficient:** 0.21502827685357004

**Running time**

0.178824  seconds

**Some interpretation of the clusters found**

In contrast to Kmeans and Kmedoids, the Dbscan algorithm does not care about outliers or choosing the numbers of clusters, it just cares about the number of “neighbors” that a point “seed” has, therefore outliers that do not have neighbors are ignored this means that even with our messy data that contains a lot of “sound” we would have relatively decent results.




In this graph (graph of our Dbscan clusters), we can observe that almost all the clusters are condensed and are not that spread out, and this is thanks to the idea of the neighborhoods and neighbors. Understandably, we have results like this because by not being impacted by outliers the data is more focused on grouping by their similarities. Some interesting things I found by inspecting the seeds (or starting points for the neighborhoods) is that the clusters that are small like 20 and 25 contain data that are not that common in other clusters like “yes” in “FamilyHx” and “top” in “School”

There is no surprise that Dbscan is the algorithm from Clustering.jl that gave the better results.

**Identify 3 outliers (based on the model provided by each clustering algorithm).**

1. Outlier was detected in cluster number 1, at point number 3717 with a silhouette coefficient of -0.5510410953549703

2. Outlier was detected in cluster number 1, at point number 2118 with a silhouette coefficient of -0.4646722344363998

3. Outlier was detected in cluster number 1, at point number 3211 with a silhouette coefficient of -0.4424075705688798

### Explain how you found the outliers

For Dbscans finding the outliers, it is pretty easy, when initializing the algorithm we must choose the number of neighbors that each point must have and if we choose a number > 1 we would find them. Because of the way the library is written, I could not just skip those outliers and not having them on the model so I add them to the first cluster, after that, I just had to find the ones with a more negative silhouette coefficient using the function **silhouettes(Dbscan.assignments, Dbscan.counts, dist)** and those were the outliers.

## Fuzzy C-Means

### How you set the parameters

For the parameters, I had to pass the data in the form of a k x n matrix where k was the number of dimensions and n the number of different data points. I also decided to use 6 fuzzy clusters after looping through a lot of values that was the number of clusters that gave me better results with a cluster fuzziness value of 2.

### Best score (according to the scoring method of each algorithm)

The Fuzzy C-Means algorithm does not have a way to find the best score, so I had to create one using the source code of the other algorithms.

A smaller Total Cost is always better

The Silhouette Coefficient has a range of -1 to 1, -1 being the worst possible accuracy and 1 being the best

**Total Cost:** 11553.558663701373

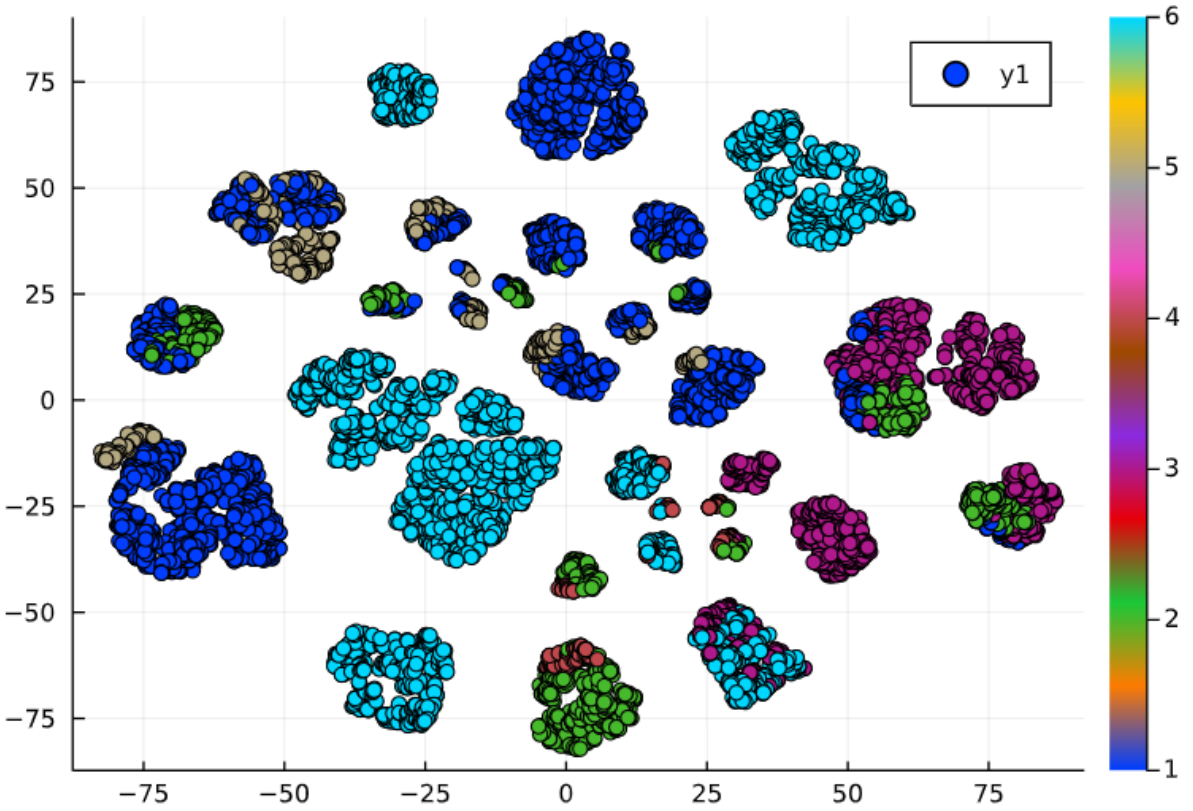
**Silhouettes Coefficient:** 0.09457012354440009

### Running time

1.288878 seconds

### Some interpretation of the clusters found

The Fuzzy C-means algorithm is the weirdest of them all, similar to Kmeans, it focuses on finding average centers between the data for each fuzzy cluster. The difference with the other algorithms is that in Fuzzy C-means points can be presented in more than one cluster.




In this graph (graph of our Dbscan clusters), we can observe that our points are all over the place, some of them are inside multiple clusters at the same time and this is what Fuzzy C-means is all about, the main problem is that again we are being affected by outliers dragging the centers around and lowering our effectivity. This result is understandable because that is like the algorithm works but I do not think it was the best choice for the type of data we are using.

It is not surprising that this was also one of the worst algorithms that I used for this assignment.

**Identify 3 outliers (based on the model provided by each clustering algorithm).**

1. Outlier was detected in fuzzy cluster number 2, at point number 5651 with a silhouette coefficient of -0.16029527744161487

2. Outlier was detected in fuzzy cluster number 2, at point number 4228 with a silhouette coefficient of -0.15942543401435727

3. Outlier was detected in fuzzy cluster number 2, at point number 5720 with a silhouette coefficient of -0.15700598284214995

**Explain how you found the outliers**

I first found all the silhouette coefficients using the silhouette function **silhouettes(Fuzzy_assignments, Fuzzy_counts, dist)** (The Fuzzy_assignments

and Fuzzy_counts variables were calculated by my as stated above using the source code of other algorithms), then I found the three most negative values. Because the silhouette coefficients determine how similar a point is to its cluster, the most negative values must be the outliers.

## Gaussian Mixtures

### How you set the parameters

For the parameters, first I initialized the Gaussian mixture with 50 Gaussians and the size of the data set. Then I added the data to the model, using the em!() function where I passed the model, the dataset as an n x k array where n is the number of data points and k the number of dimensions, I also specified 20 iterations (nIter= 20) with a varfloor of '1e-10'

### Best score (according to the scoring method of each algorithm)

A bigger Average Log Likelihood is always better

**Average Log-Likelihood:** 0.27173994022561465

### Running time

4.596819 seconds

### Some interpretation of the clusters found

Because Gaussian Mixtures is better to model non-circular and scalable data it makes it the best type of model that we can use to test our dataset, sadly the GaussianMixtures library is really buggy and do not properly work in Julia so I was not able to plot the data, but by analyzing the log-likelihoods and the average log-likelihood we can interpret that we created an appropriate model where to fit the data. Moreover, it makes sense we have a lot of high values for our outliers mostly because as I stated before the data contain a lot of them that skewed the final results.

### Identify 3 outliers (based on the model provided by each clustering algorithm).

1. Outlier was detected in gaussian number 1, at row number 8051 with a log-likelihood of -27.603600552854303

2. Outlier was detected in gaussian number 1, at row number 3101 with a log-likelihood of -25.05602711263232

3. Outlier was detected in gaussian number 1, at row number 913 with a log-likelihood of -21.833534603946944

### Explain how you found the outliers

I found the log-likelihoods array for the model, then I looked for the three most negative values to determine the outliers. The three most negative values of the log-likelihoods are the outliers because the log-likelihood function tells us in some sense how 'plausible' the model at hand is, given the data that we have.

## Notes:

- If you run again the code might have different results due to the library not providing a way for us to change the random seed.
- My code assumes that the Data Set is  in the same main folder as the code
- Necessary Packages:
    - CSV
    - DataFrames
    - Clustering
    - GaussianMixtures
    - Distances
    - Statistics
    - ScikitLearn
    - Plots (only for my interpretation, is not needed)
- All of the necessary Packages can be downloaded inside the program, just uncomment the necessary packages at the beginning of the code.