Datasets choice reminder **2**
1- First Dataset: 2
2- Second Dataset: 2
3- Motivation: 2
Implementation and evaluation **2**
1. Clustering 2
K-means: 3
Gaussian mixtures: 4
2. Feature Transformation 5
PCA: 5
ICA: 5
Randomized projection: 6
K-best selection: 6
3. Clustering after dimensionality reduction: 7
4. Neural Network on reduced features (and runtimes of algorithms): 9
On transformed features: 10
On clusters as features: 11

Datasets choice reminder

1- First Dataset:

The first dataset is Indian patient records in and their binary labels as having or not a liver disease.
The features used are basic patient information (age, gender), levels of some chemical compounds present in the human body and results of some medical tests.
The datasets size is : 583 data points with 10 features.
The labels ratios are : 72% positive examples versus 28% positive examples.

2- Second Dataset:

The second dataset is for fall detection, it is a multi-class dataset that labels elderly people in a chinese hospital, given some of their monitored health data (Sugar level, heart rate, blood pressure etc…) in six different states : Standing, Walking, Sitting, Falling, Cramps or Running.
The datasets size is : 16382 datapoint with 6 features.
The ratios for these states are respectively : 28% 3% 15% 22% 22% 10%.

3- Motivation:

What makes these two datasets interesting is that falls are one of the major health issues that the elderly face, especially when they're not permanently monitored by nurses. and that since liver damage has been an increasing health issue in India due to the excessive consumption of alcohols and/or contaminated foods etc..

This makes the problems at hand meaningful and potentially very impactful. But what's also important is that these datasets are perfect for evaluating clustering and feature transformation methods for the following reasons:

- A lot of relevant features for both datasets (We can be fairly confident that the information of the liver's health is somewhat included in the medical tests and chemical levels, and similarly the state of the elederly person given their heart rate etc..
- Clean datasets and almost no preprocessing needed
- The datasets are of varying lengths0.
- Binary classification vs multi-class classification with the same algorithms.

Implementation and evaluation

1. Clustering

We will evaluate two clustering algorithms that we will implement using the sklearn library; the distance considered for both these algorithms and for both datasets will be the euclidean distance for the following reason:

- The comparison of distance methods when using KNNs for both datasets in assignment 1, showed that euclidean distance is the one that allows for the best discrimination of our dataset labels. And although we do not use the labels in this ; step, we ultimately want our clustering to help us separate between the different classes.

Our features space is multi-dimensional so we will also use the a two dimensional vector embedding using sklearn TSNE (converts euclidean similarity between data points to joint probability and projects in 2D in order to minimize the KL-Divergence). This allows us to have a 2D visualization of our data that represents euclidean similarity between points.

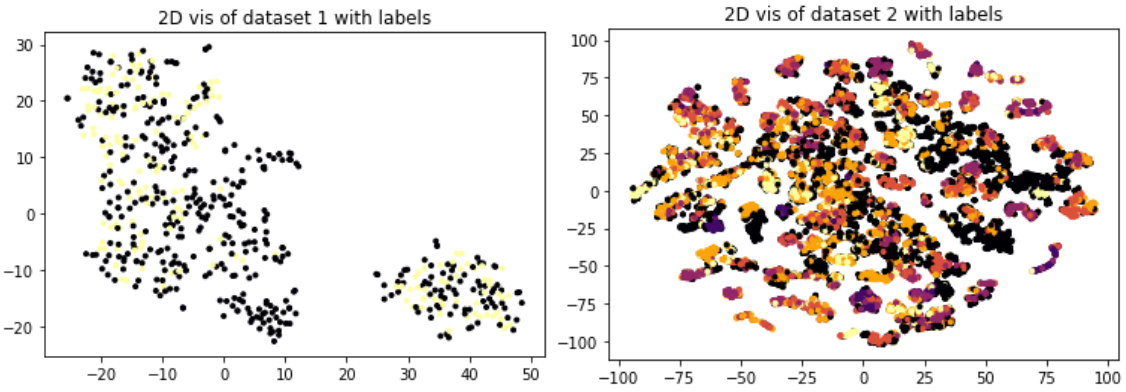


Figure 1 - 2D visualization of dataset 1 (left) and dataset 2 (right) with different colors for every class.

K-means:

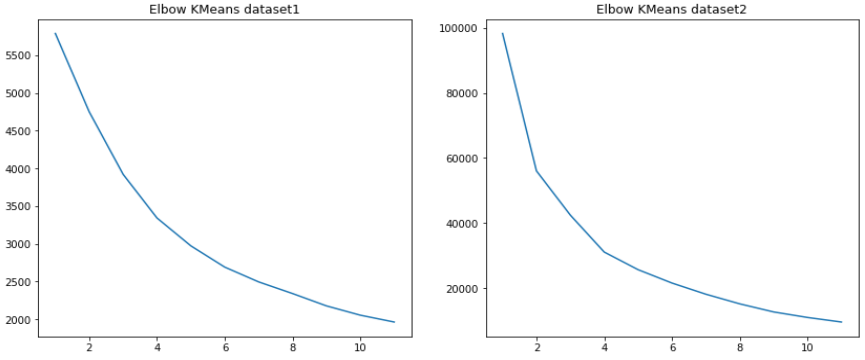
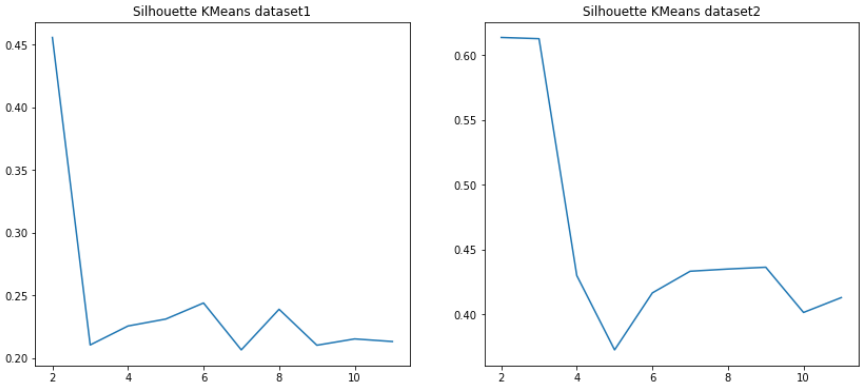


Figure 2 - Elbow method with KMeans for Dataset 1 (left) and Dataset 2 (right)

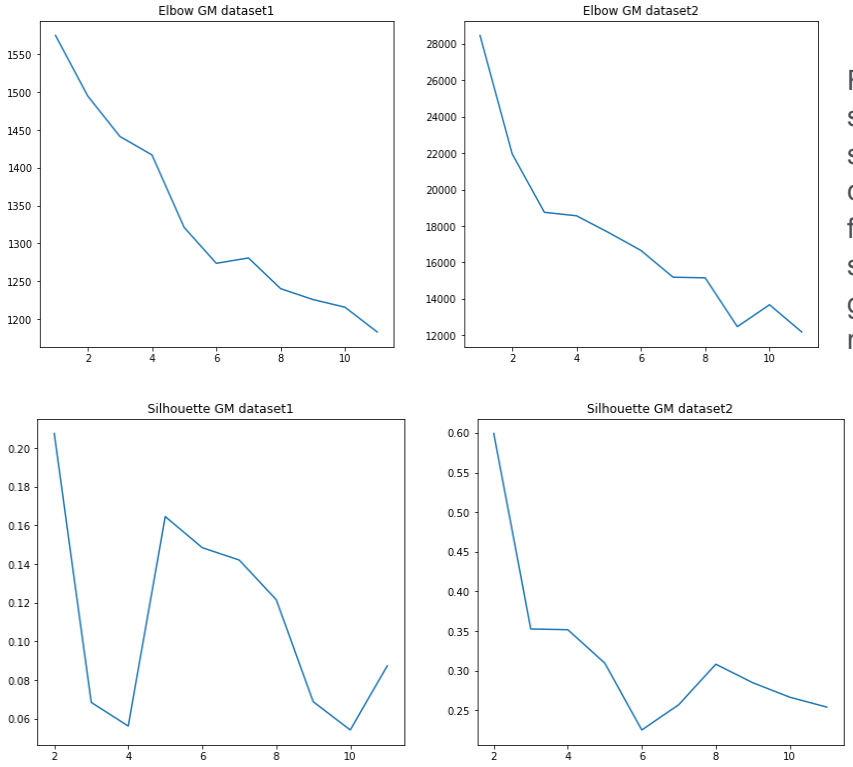
The error is calculated as the sum of squared differences, there does not seem to be a strong optimum for any of the datasets. But for the dataset 2 we have a steeper descent early on, that might be because, although there are 6 different clusters, there is a strong difference in almost all features between one class (fall) and all others, so there is a chance that the clustering algorithm actually separates accordingly.



The silhouette score is a measure of how similar an object is to its own cluster compared to other clusters. The closer it is to one, the better.
We can see it is better values for smaller values of k but also that it is overall better for dataset 2

Figure 3 - Silhouette score for different k number of clusters with KMeans (Dataset 1 on the left)

Gaussian mixtures:



For dataset 1, we have a low silhouette score right from the start which means that there is a confusion between clusters right from the start whereas for the second dataset we do have a good score for 2 clusters which means there one cluster that separates well from the others. (same as K-means)
But we see that error values are overall smaller than for K-means which makes sense since both our datasets have data that is not easily separable by distances.

Figure 4 - Elbow method (top) and silhouette score (bottom) with gaussian mixtures for Dataset 1 (left) and Dataset 2 (right)

Let's a look at the clusterings in 2D and compare them with the labels (we'll choose k equal to number of classe: 2 for dataset 1 and 6 for dataset2)

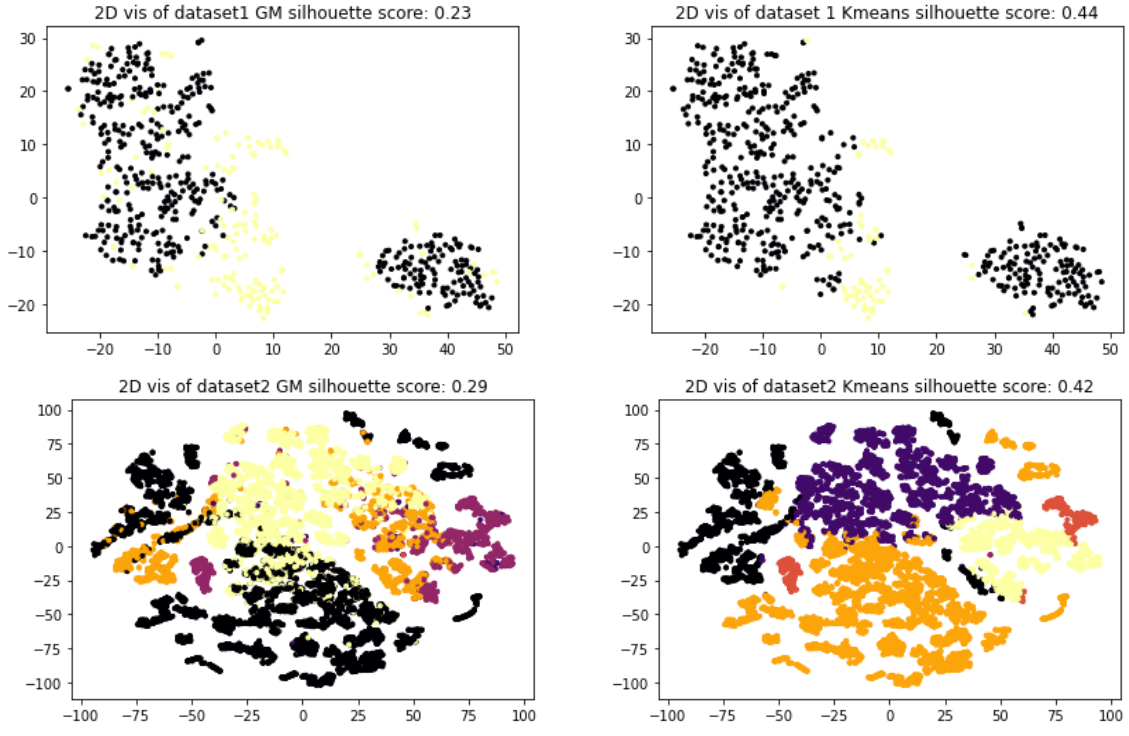


Figure 5 - 2-D visualization of clustering (K-means and Gaussian mixtures for both datasets)

Using these two algorithms alone, it is hard to have meaningful clusters that align with our labels, simply because with distance metrics using our features the way they are, close points can have different labels as we can see on figure 1, whereas k-means and gaussian mixture (especially kmeans) and any clustering method in general tends to cluster close points together.

2. Feature Transformation

PCA:

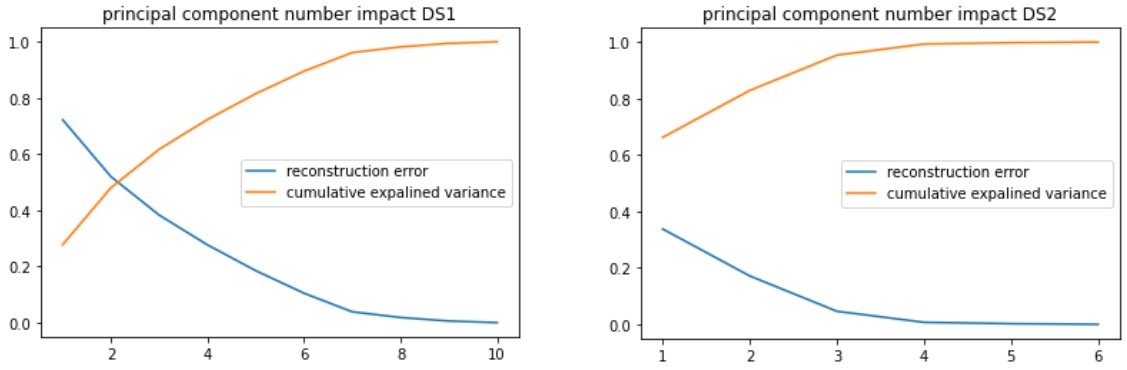


Figure 6 - Explained variance and reconstruction error evolution with number of pca components for both datasets.

We vary the components number from 1 to the total number of features for each dataset, the explained variance added by every component is the proportion of the magnitude of its associated eigenvalue over all the other ones. We can see that for both datasets the reconstruction error is symmetrical with the explained variance, as the pca perfectly explains what component bears exactly how much of the data's variance. We can threshold at for example 95% of the total data variance and keep the first 6 for components for dataset 1 and the 3 first for dataset 2.

ICA:

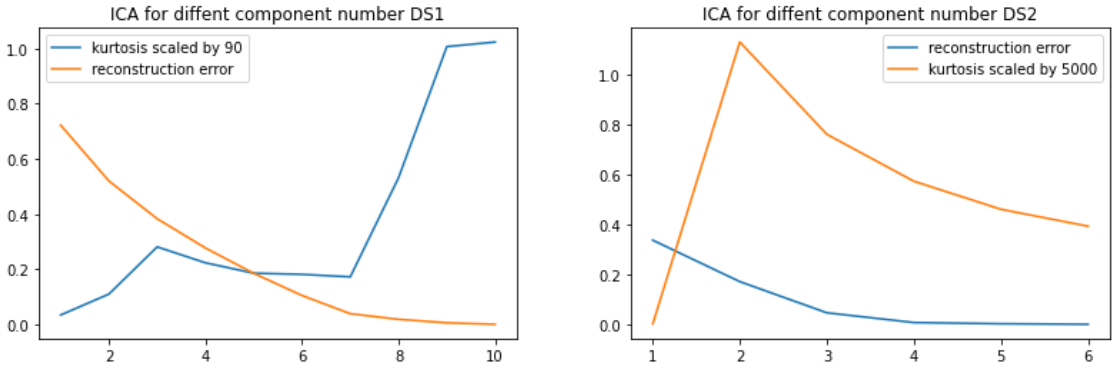


Figure 7 - Kurtosis and reconstruction error evolution with number of ica components for both datasets.

We use a fast ICA algorithm that finds the independent components by trying to make the distributions of every component as kurtotic as possible, which means as non-gaussian as possible.
So we will pick a component number that has both a high kurtosis and a low reconstruction error, because sometimes we can have high kurtosis but while still losing relatively lots of information,

like for dataset 2 with two components. The fact that ICA does not select features in an order makes it important to try for various numbers of components.

Randomized projection:

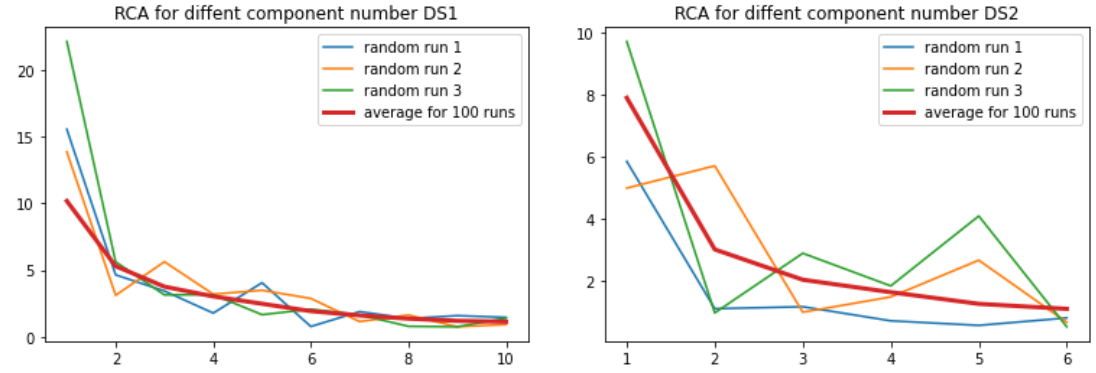


Figure 8 - Reconstruction error evolution with number of components (3 random runs and one average of 100 runs) for both datasets.

This randomized linear transformation of features, allow us to have a cheap and fast dimensionality reduction, and looking at the plots, we can see the importance of running multiple restarts: On average, higher component number implies higher quality reconstruction but sometimes for random runs we can a better representation with 2 components than with 5 (green curve for dataset 2).
It is also interesting to point out that overall the reconstruction error is the highest among all other algorithms and that even when picking as many components or more than the original number of features, the reconstruction is never perfect (because every component is created randomly).

K-best selection:

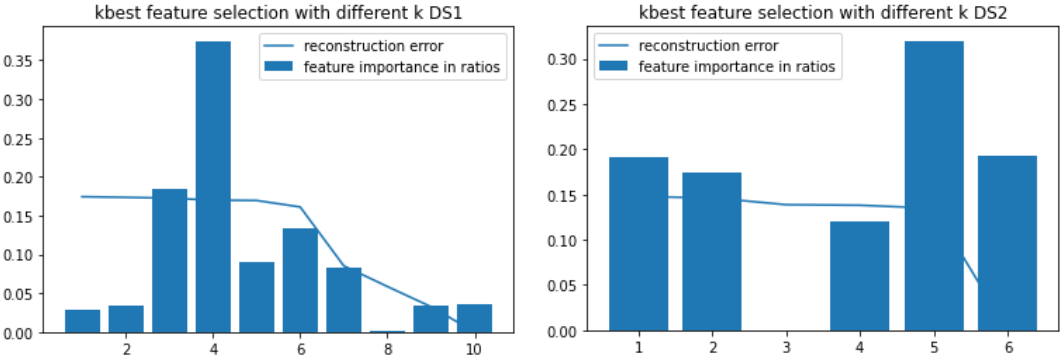


Figure 9 - Reconstruction error evolution with number of components for both datasets and feature importance in ratios (dependance between feature and label).

This algorithm has the particularity of using the label information, which can be very powerful but also extremely limiting (for unsupervised tasks).
It uses the chi2 distribution (K-best can use any other function that will output a score between labels and features) to compute dependency scores between every single feature and the labels; then we keep the k features that have the highest scores.

As the graphs show, we end up with a ratio that represents the importance of each feature, it is very interesting because we can catch features that have no relevance with respect to our

labeling problem, in this case we can set a threshold and eliminate all features that have below 5% of relevance. We'll keep 5 for dataset one and 5 for dataset two as well.

3. Clustering after dimensionality reduction:

For this step we used the tuning that was "best" for each of the feature transformation algorithms as we saw earlier, and now for every dataset and for the two clustering methods, we will plot the silhouette scores and elbow figure. We will try to analyze the elbow plots.
We also have a 2D visualization of the clustered datasets for each run (16 plot) with the number of labels as features. We will also discuss the relevant ones (the others can be found in the code).

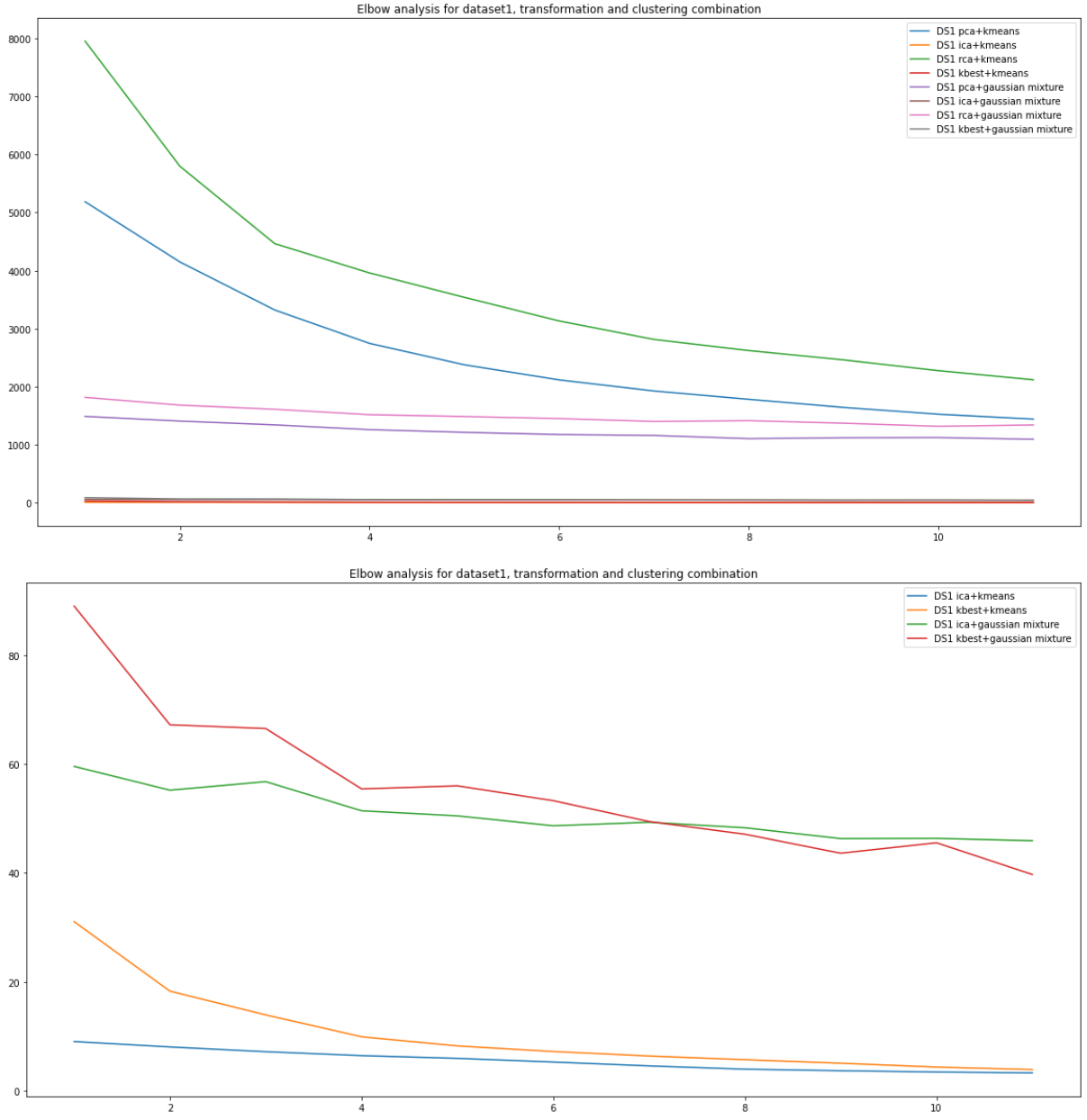


Figure 10 - Elbow analysis for dataset 1 at 2 different scales

We can see that the best curve is obtained after running the k-best or ICA algorithm. The poor performance of the randomized projection can mainly be explained by the fact that the randomness of the features has no guarantee of being useful to us. K-best has the information on the label so that could be why it performed better. PCA not working very well suggests that our features have weak linearity between them.

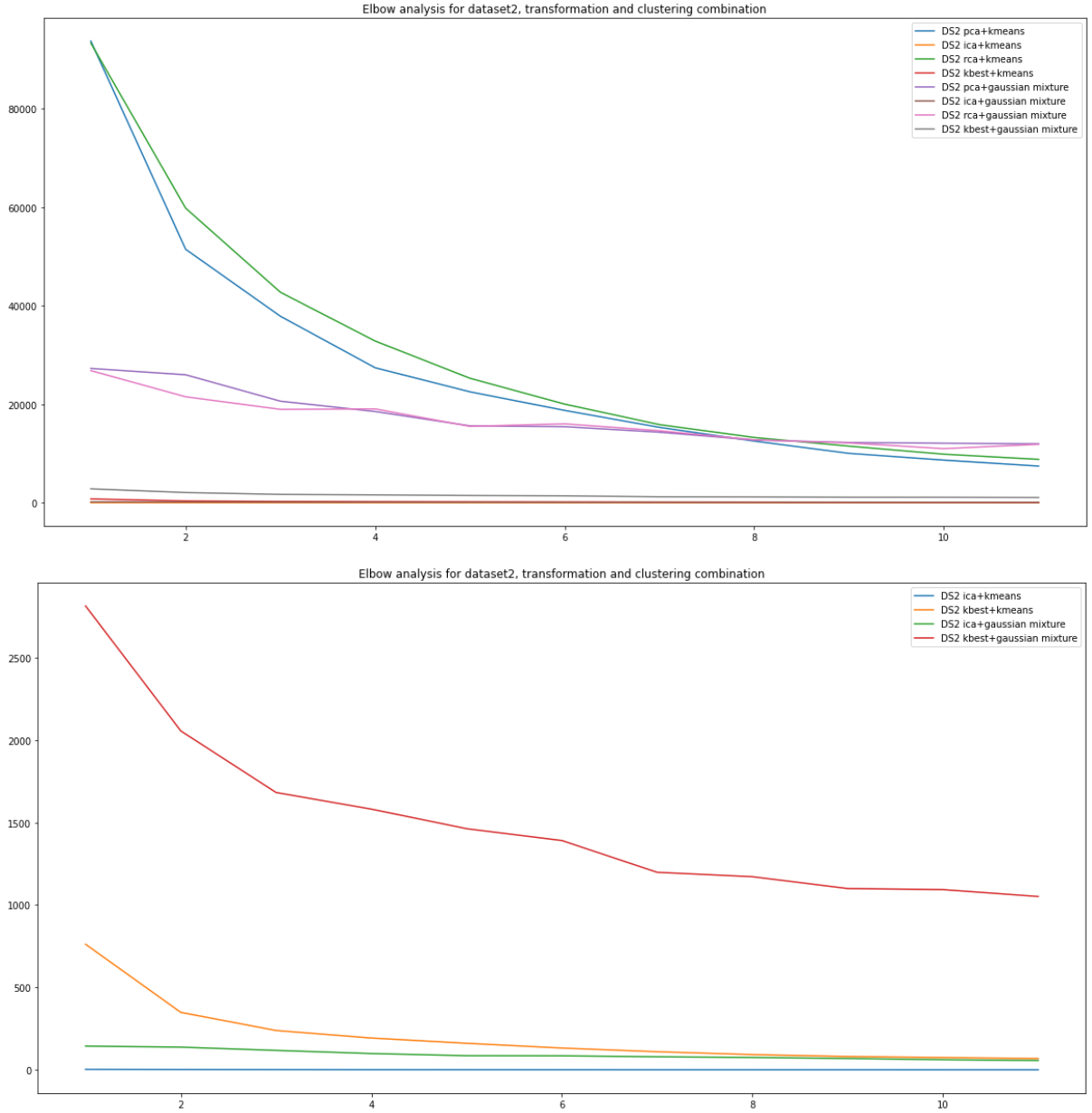


Figure 11 - Elbow analysis for dataset 2 at two different scales

We can that once again pca and randomized projection result in the “worst” clustering whereas as ICA works the best, this means that there are probably some latent underlying variables behind our observable features.

An important note: after the feature transformations, the scaling can differ a bit which makes the comparison not 100% accurate, but it can still be insightful.

We can now visualize the clusters that we obtain for ICA (since it performed the best in SSD error) :

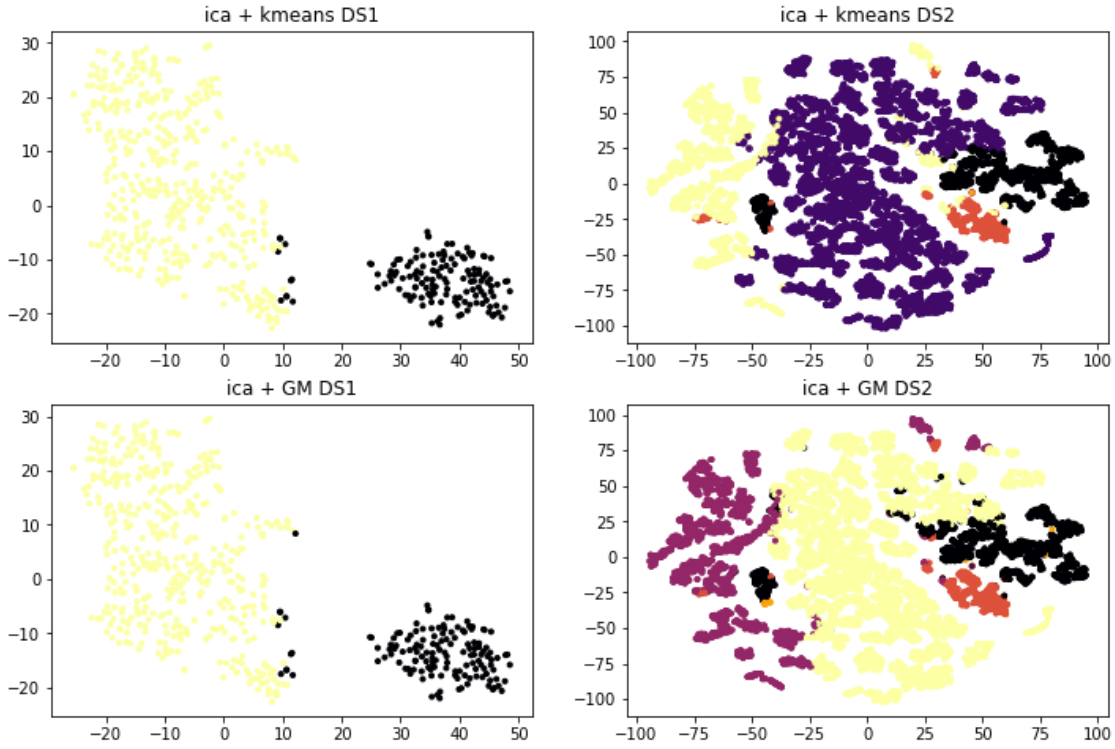


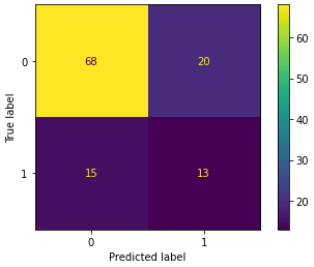
Figure 12 - Clusters 2D visualization for K-best with K-means and GM for both datasets

we get more or less similar results with the other dimensionality reduction methods, and it is closer to the clustering before the dimensionality reduction than it is to the labels clustering. Now this is an expected result for the three first algorithms, because they try to conserve the same variance that original data has, but with the K-best we would have hoped that the new features would capture better the variance in the labels.

Which is why the PLSR (partial least square regression) method is a feature transformation method that I would have liked to try for these datasets, it is a feature transformation algorithm that conserves the variance of the features that explains the best the variance in the labels. Which would have been very interesting in our case as we need to project our features some way that would make it more “separable” between the labels before running our clustering algorithms.

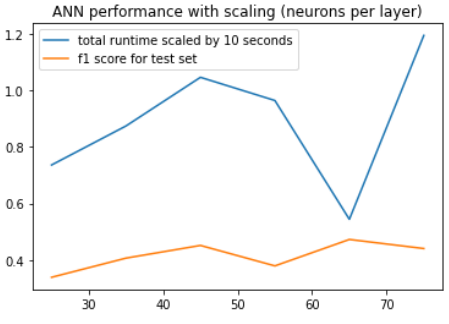
4. Neural Network on reduced features (and runtimes of algorithms):

We’ll do this using the first dataset since it has the highest number of features, so that we can really see the impact of the number of features on the training time and accuracy. The reference metrics on our neural network running on the original sized features

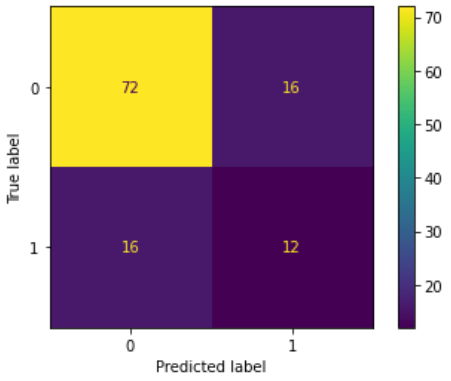


F1 train score:0.99
F1 test score:0.42
runtime:11.95 seconds

Figure 13



On transformed features:



PCA:

F1 train score:0.93
F1 test score:0.41
runtime:3.81 seconds

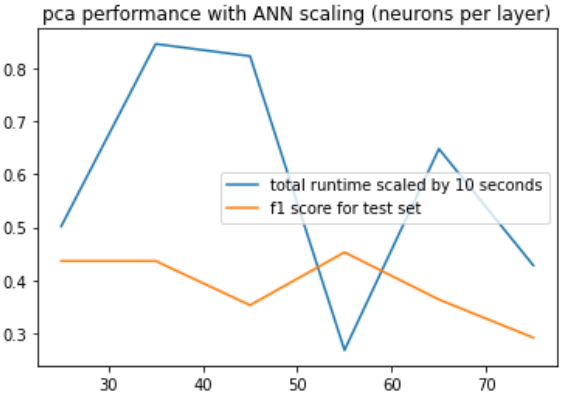
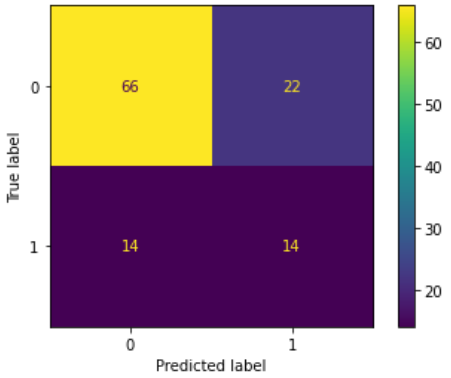
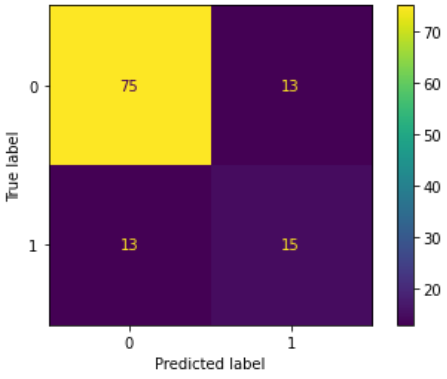


Figure 14 - NN with PCA



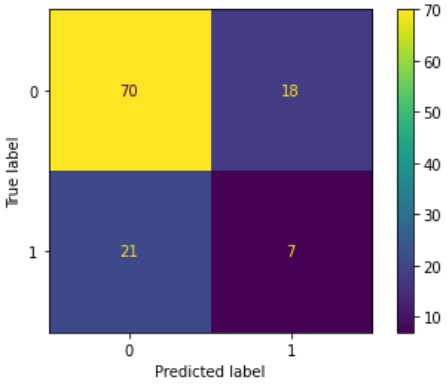
ICA:

F1 train score:0.95
F1 test score:0.43
runtime:6.59 seconds



RCA:

F1 train score:0.97
F1 test score:0.53
runtime:3.06 seconds



K-best:

F1 train score:0.44,F1 test score:0.48

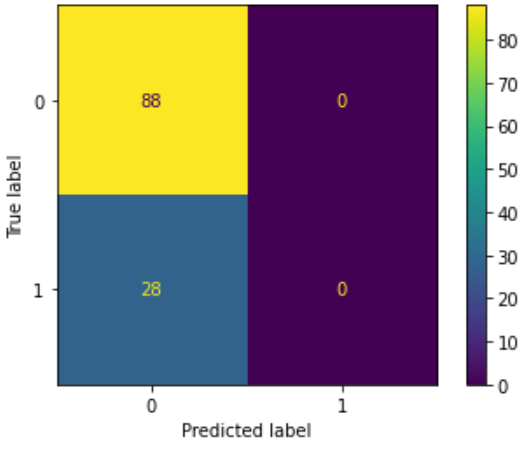
runtime:5.82 seconds

We can clearly see that all the dimensionality reduction algorithms make the neural network train faster (because we

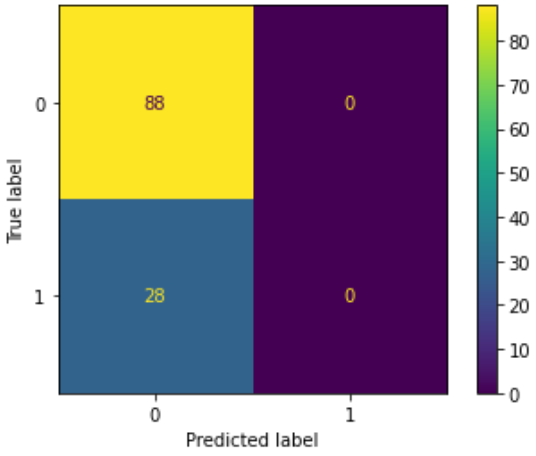
have less features to process) the fastest being RCA because even its algorithm to reduce dimensionality is fast.
Score-wise we do obtain better scores than when we use all the features, we go from around 0.4 F1 score to around 0.5, which is a consequence of discarding the least relevant future: the training has better chances of converging towards a better optimum.

On clusters as features:

K means



Gaussian Mixture



When we use the clusters as features, for this dataset, for both K-means and Gaussian mixtures and no matter the number of clusters K that we choose, the algorithm converges towards classifying everything as the predominant label. And that happens because as we saw in the earlier parts, the clusters that we obtain and the actual labels have no correlation between them. (figure 1 versus figure 5). So it is almost as if we were trying to teach the model based on random feature vectors, which is why we obtain this behaviour.

Now we also tried for dataset 2 because clusters from figure 1 and 5 didn't seem to be absolutely uncorrelated. By varying the number of clusters we obtain:

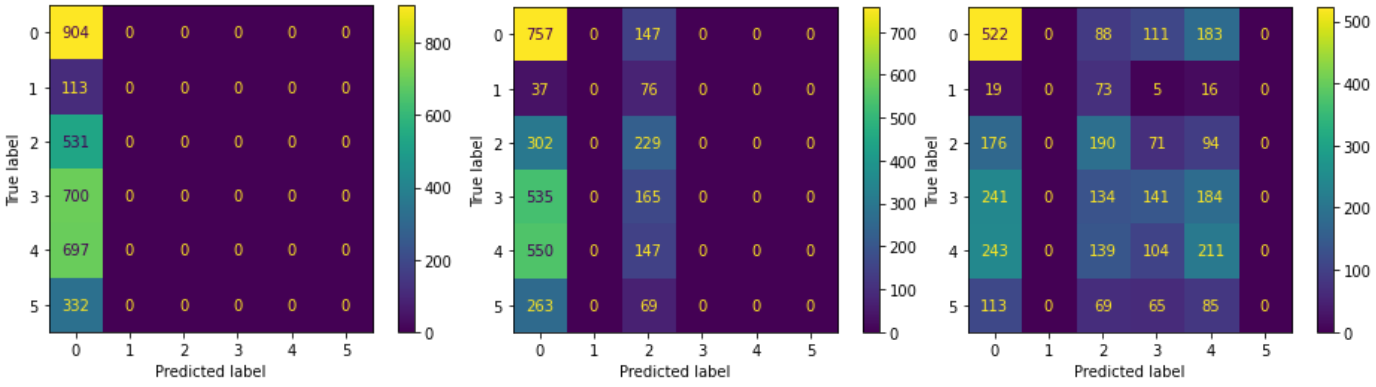


Figure 15 - NN with K-mean clusters as labels for dataset 2 (K values of 6, 15 and 20 from left to right)

Now for this dataset we have the same results as before for low K values, but as it increases, the clusters start to capture information that helps the model predict the classes correctly.