Machine Learning - Report of the second practical work

João Albuquerque (45040) & Miguel Almeida (45526)

January 16, 2018

Abstract

Our work consists in implementing and examining the performance and behaviour of three different clustering algorithms in the problem of clustering seismic events, the K-Means, the Gaussian Mixture and the DBSCAN models. We implemented the algorithms and internal indexes, and using an internal index, the silhouette score, and external indexes computed from the fault line information: the Rand index, Precision, Recall, the F1 measure and the adjusted Rand index, we computed the best main parameter for each. We also used the elbow graph to compute the best parameter for the DBSCAN method according to the article "A density-based algorithm for discovering clusters in large spatial databases with noise (1996). Martin Ester , Hans-Peter Kriegel , Jörg Sander , Xiaowei Xu."

1 Introduction

The goal of clustering is to create groups of aggregated examples in a way that maximizes some measure of similarity between examples aggregated in the same group (cluster) and minimizes the same measure of similarity in examples that belong to different groups. This can help us understand the structure of our data and the connections between similar examples and features. Also it can be used to replace a large data-set with a smaller number of data points allowing us to effectively summarize our data.

To achieve a better understating and comprehension of clustering algorithms we implemented three different types of clustering algorithms using real and not synthetic data of seismic events. Our goal is to examine the performance of each of our algorithms in clustering the seismic events, and detail the pros and cons of each of our implemented clustering algorithms as well as the importance of certain index and external indexes to select the value of parameter of each algorithm that maximizes its performance.

The clustering algorithms implemented in this project were the K-Means, Gaussian Mixture and the DBSCAN models.

The most interesting aspect of this project was the process of selecting the best value of the main parameter of each of our models (number of clusters, number of Gaus-

sian components and neighbourhood distance, respectively), through the behavioural observation of the different external and internal indexes computed.

We also used the method of plotting an elbow graph of every distance of the 4th neighbor of each point to determine the optimal neighbourhood distance, to use in the DBSCAN model and compared the result we obtained with the one using the cycle methods.

2 System Implementation

The data was pre-processed in the sense that, by considering the Earth a three dimensional axis, we calculated the x, y and z coordinates of each seismic event using the longitude latitude and Earth radius of each point, making possible the calculations of distances between seismic events, which is pretty useful for the algorithms we implemented. There was no need to standardize and shuffle the data, because we will use parameters (x, y, z) of the same unit and in the same scale standardizing the data wouldn't prove useful, shuffling it's also not required because we are going to use the the whole data-set and not parts of it, so by shuffling the data the results would be the exact same.

2.1 Implementation of the K-Means Clustering Algorithm

The K-Means clustering algorithm is an example of prototype-based clustering. In prototype-based clustering each example is assigned to the cluster represented by the closest prototype, usually the result become a Voronoy partition of the feature space because the most employed distance measured is the euclidean distance. The K-Means clustering algorithm consists in dividing the data-set into k clusters, the prototype of each cluster being the mean vector of the members of the cluster. Each example is assigned to the cluster represented by the closest prototype. This algorithm starts assigning a random set of prototypes and attributing each seismic event to the closest prototype, it then recalculates the value of each prototype by recomputing the mean point of all examples in that prototype and assigning the prototype to it. It then repeats this process over and over again till it reaches convergence or a stop condition.

The parameter to optimize in this clustering algorithm is the number of clusters that initially divide the data-set, by assigning to few we have the risk of having a data-set too summarized and not useful at all, with examples not that similar being represented by the same prototype, as well as by assigning to much clusters we can be left with an exact replica of the same data-set or with even more data-point than the original data-set this is not useful at all because it makes the process of analyzing data too complex.

2.2 Implementation of the Gaussian Mixture Clustering Algorithm

The Gaussian Mixture Clustering Algorithm utilizes the Expectation-Maximization statistical algorithm to fit mixtures of Gaussian models, it is very similar to the K-Means algorithm in the way it works. First we assume a number of components normally distributed around the origin, and compute the probability of each point being generated by each component of the model, then the components are tweaked to maximize the likelihood of the data given these assignments, this process is repeated till we achieve convergence or a stop clause.

In a way this algorithm it's very similar to the K-Means algorithm using components to determine the probability of each data-point belonging to each component instead of evaluating the euclidean distance of each point to the prototype.

The parameter to optimize in this clustering algorithm is the number of components used initially, having the same problem of the number of cluster in the K-means algorithm, if we have too few components the risks of the result being a data-set too summarized in which we can't infer any conclusion increase drastically, in the other way around if we have too many components the data-set that we are left out with can be an exact replica or a more complex one in relation to the original which is not useful at all.

2.3 Implementation of the DBSCAN Clustering Algorithm

The DBSCAN Clustering Algorithm (Density Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm, which means that it assigns examples to cluster based on high-density regions, while some examples are left out, unassigned and discarded as noise.

To understand this algorithm first we are going to introduce a few concepts:

- Density : number of points within a specified radius(Eps).
- Eps : maximum radius of the neighborhood.
- MinPts : minimum number of points in an Eps neighbourhood of that point.
- Core point : if it has more than a specified number of points (MinPts) within Eps.
- Border point : has fewer than MinPts within Eps, but is in the neighborhood of a core point.
- Noise point : any point that is not a core point or a border point.
- Density-Reachable : A point p is density-reachable from a point q w.r.t. Eps, MinPts if there is a chain of points p1,..., pn, p1=q, pn= p such that pi+1 is directly density reachable from pi

How this algorithm works is it arbitrarily selects a point p, it then retrieves all points density reachable from p w.r.t. Eps, MinPts, it then evaluates if p is a core point or not (if it has a number of density reachable point higher than MinPts), if it is a cluster is formed, if it isn't and p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database. This algorithm may merge two clusters if there are core points close to each other, merging the two cluster formed by these core points.

The two parameters of this cluster algorithm to optimize are the MinPts and the Eps, however as it was suggested by us in the paper "A density-based algorithm for discovering clusters in large spatial databases with noise (1996). Martin Ester , Hans-Peter Kriegel , Jörg Sander , Xiaowei Xu.", we are going to set the MinPts parameter always at 4 because it considerably takes less computations to execute and behaviours too similarly as with parameters of higher value. SO that would only leave us to optimize the Eps parameter. This Eps parameter is very important to the DBSCAN algorithm, because it defines the radius of search for each point, if the radius of search is too small the number of cluster would be considerably higher and the data-set too complex to evaluate, however if the radius of search is too high the possibilities of having merged cluster that shouldn't have been merged increase having a data-set that's too summarized.

This algorithm works well because it ignores noise and is able to handle clusters of different shapes and sizes, however it is very susceptible to varying densities and higher dimensional data decreasing its performance.

2.4 Clustering Validation

To validate the parametrization and behaviour of our clustering algorithms we used several indexes that would allow us to grade and compare the performance of our algorithms with different parameters

2.5 External Indexes

For external indexes we used different types of indexes, like precision, recall, F1 score, Rand Index and ARI (adjusted Rand index).

- Precision : fraction of correctly classified positive examples in all examples classified as positive (correctly or not).

$$\frac{TruePositive}{TruePositives + FalsePositives}$$

- Recall : fraction of correctly classified positive examples from the set of all positive examples.

$$\frac{TruePositive}{TruePositive + FalseNegative}$$

- F1 : harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$2 * \frac{Precision * Recall}{Precision + Recall}$$

- Rand Index and Adjusted Rand Index : The Rand Index computes a similarity measure between two clusters by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. The raw Rand Index score is then adjusted for chance into the ARI score using the following formula (Where N is the total number of examples):

$$RandIndex : \frac{TruePositives + TrueNegatives}{N(N-1)/2}$$

$$ARI : \frac{RI - ExpectedRI}{max(RI) - ExpectedRI}$$

We expect to maximize the values of all of this indexes.

2.6 Internal Indexes

For Internal Indexes we calculated the Silhouette Score.

The Silhouette Coefficient is a measure of how well the cluster mechanism is working, to know if the points are being clustered with points that are similar to them. Clustering models with a high Silhouette values means that points in the same cluster are similar to each other, and well separated, where points that are in different clusters are different from those.

The best value for the silhouette coefficient is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

2.7 Finding Optimal Values

To find optimal the optimal values of each parameter we ran the algorithms sequentially increasing the value of the parameters and calculating the values different indexes and storing them in an array, then we plotted the graphs of each index in relation to the parameter being evaluated.

Trough observation we decided which value of each parameter fared better and gave the algorithm the best perforamnce.

For the DSBCAN algorithm to find the optimal Eps possible we calculated the distance of 4th nearest neighbor of every data-point in the data-set using the KNNeihbors method. Then we matched that distance to the particular point and plotted the results,

this graph is called elbow graph and it gives us an idea of the density distribution for the points in the data-base, the elbow of the graph should be the threes-hold of the Eps because it tells us when do we start having thinner density between points, and that should be the deciding fact between assigning an example to a specific cluster or other.

2.8 Evaluation

As the graphs between the K-Means, and Gaussian Mixture clustering algorithms are very similar, meaning that they have a very similar behaviour we decided to analyze them together, to also easily point out the small differences.

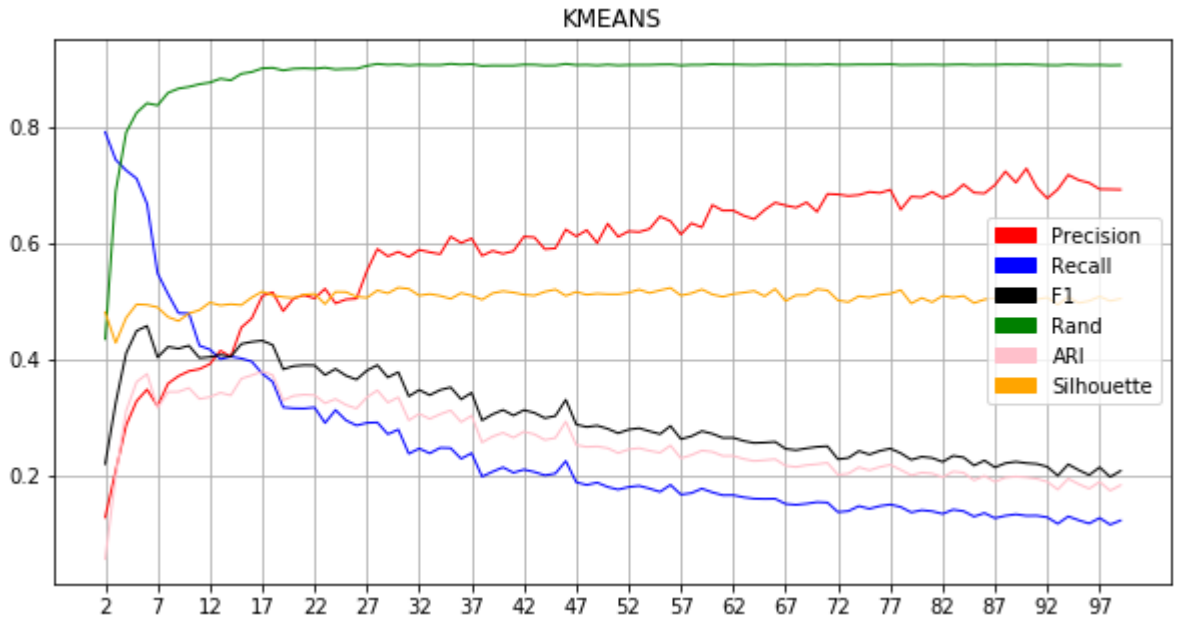

Figure 1: K-Means Clustering with 2 to 100 clusters

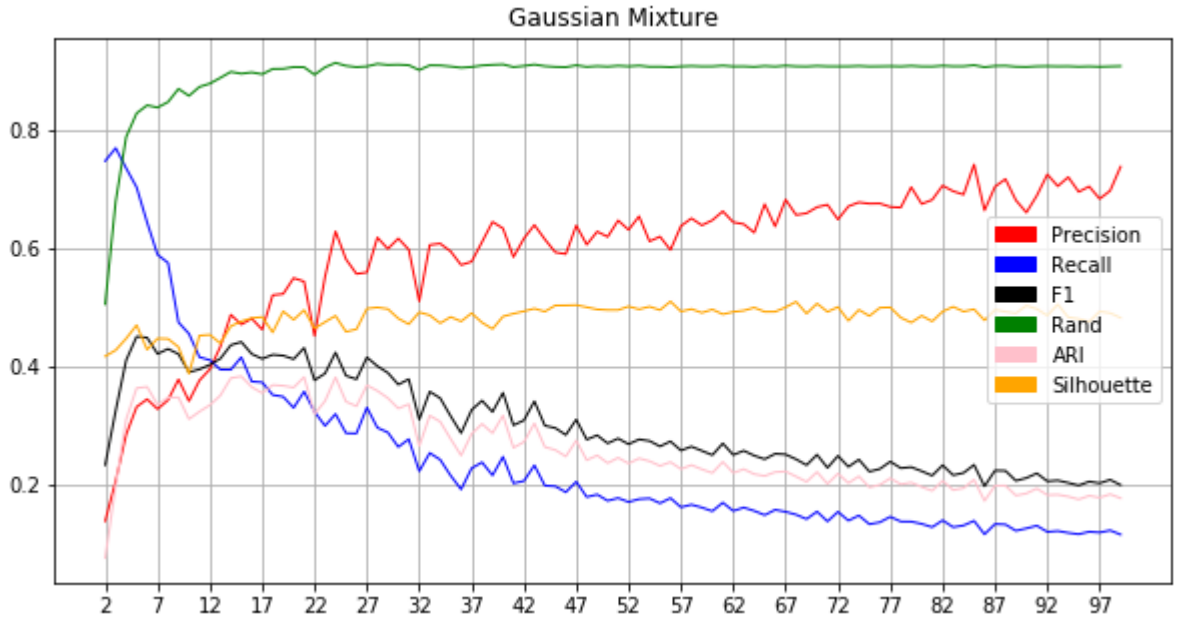

Figure 2: Gaussian Mixture Clustering with 2 to 100 components

As we can see in the figures 1 and 2, we executed our program to test the K-Means and Gaussian Mixture clustering algorithms using 2 to 100 clusters/components, as we can see after the use of 37 clusters we observed that the results of the different indexes would not improve and would have small indifferent oscillations, because of this we reduced the window of results setting the interval of the number of clusters/components between 2 and 30, allowing us to obtain more precise and visible results, as seen in the picture below.

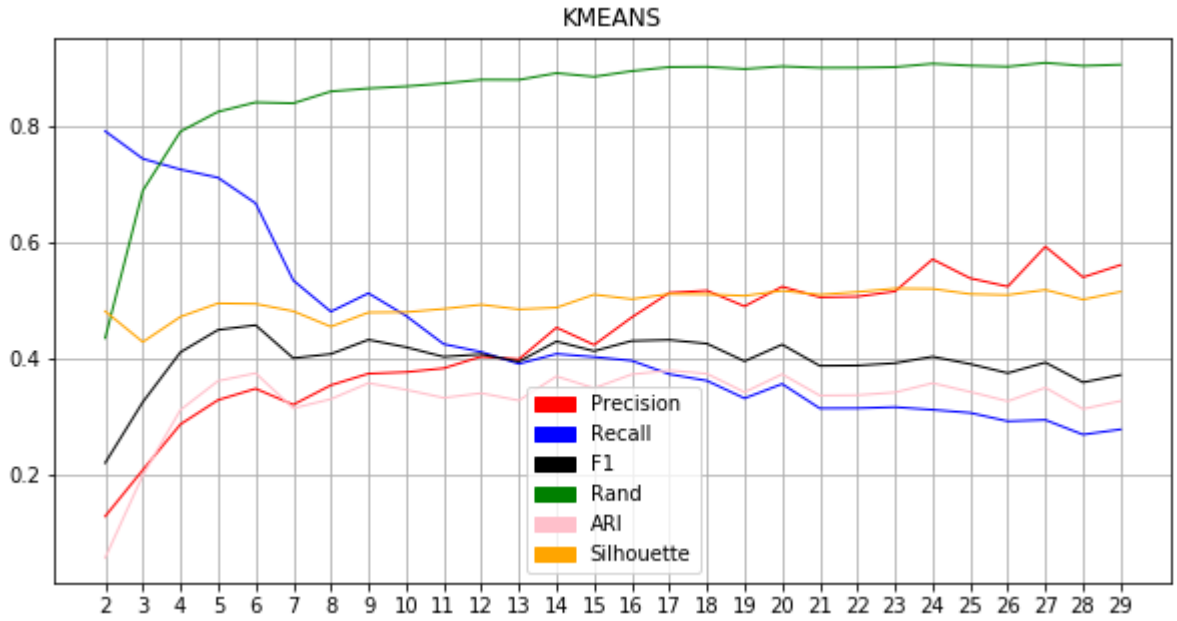

Figure 3: K-Means Clustering with 2 to 30 clusters

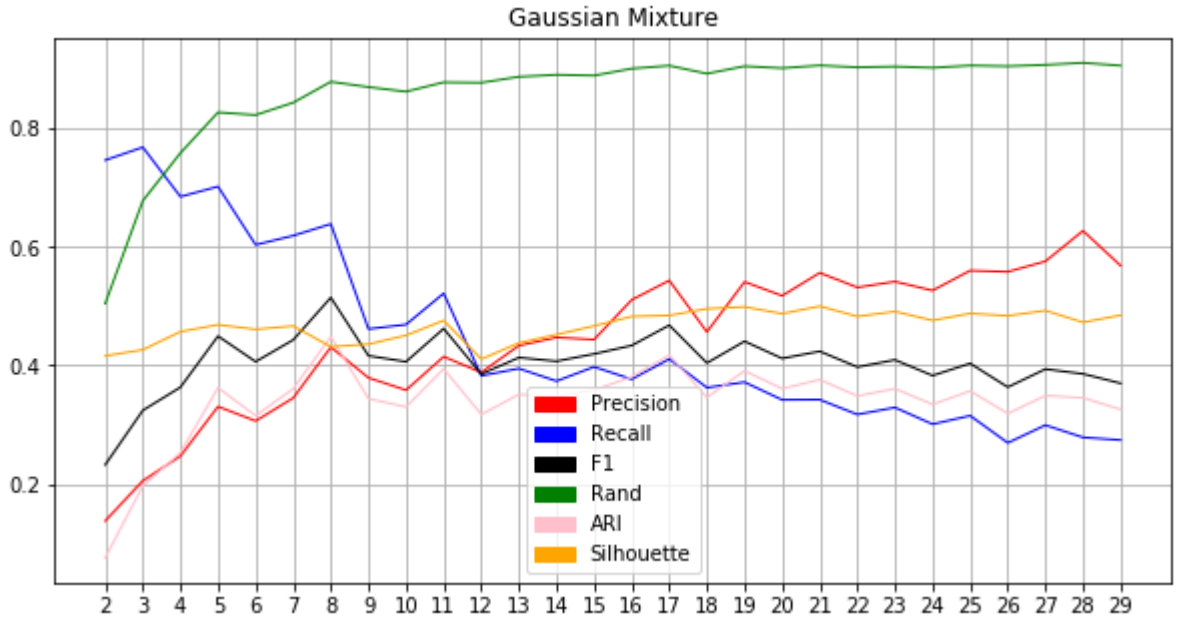

Figure 4: Gaussian Mixture Clustering with 2 to 30 components

Using a smaller window as we can see in figures 3 and 4,it's possible to easily identify and detail the behaviour of the multiple indexes.

The precision score increases with the number of clusters/components used, this is due to the fact that as we increase the number of clusters we decrease the probability of having false positives because we tend to approximate our number of clusters to the real number of faults, so having two points in the same cluster that are represented by different faults is highly unlikely to happen.

The recall score decreases with the number of of clusters/components used having its maximum value when we use only two components, because having only a really small number of clusters/components the number of false Negatives decreases drastically because it's hard to not classify similar examples in the same cluster/component having so few of them.

The F1 score although it suffers a small but drastic increase in value using small numbers of cluster/components, it then decreases very slowly but surely as we increase the number of cluster/components used, this is due to it being a harmonic mean between the recall and precision and it always stays as an intermediate value of both with a slight edge to the lower value, because it gives more prevalence to the decrease of precision/recall than its increase.

The Rand value increases drastically in the start and the it stabilizes and shows a similar value for the rest of upcoming values of number of clusters/components, if we look at the formula of Rand Score we see that the denominator of the fraction stays the same between all the iterations of the algorithm, we evaluate always the same number of points. That leaves us with evaluating the number of true positives and true negatives, which by the behaviour of precision we know that the number of true positives increases with the increase of clusters/components drastically in the beginning, and by the behaviour of recall we know that the number of true negatives decreases because the number of false negatives increases, this explains the drastic increase in the beginning as the increase of true positives heavily outweighs the decrease of true negatives, but it then stabilizes as both the increase of true negatives and the decrease of true positive are proportional. It always maintains really high values.

The ARI score has a very similar behaviour to the Rand Score although in lower values, it also has an increase in the beginning and it stabilizes throughput the rest of the runs. This is due to the fact that the Rand Value doesn't have much variations and the expected index pretty much stays the same. The low values of this index state that there is a low correlation between the clustering and the group it is being compared to.

The silhouette score has really low variations throughout the increase of clusters/components and it always stays in values close to 0.5 this indicate that most of the points are being assigned to the right cluster, representing the cluster with points that have a higher degree of similarity to this point than points in other clusters.

In concordance with the graphs we selected the value of 6 clusters as the optimal value for the K-Means clustering algorithm and the number of components 8 as the optimal value of the Gaussian Mixture clustering algorithm, which we used to cluster seismic events and represent them in a Mollweide Projection.

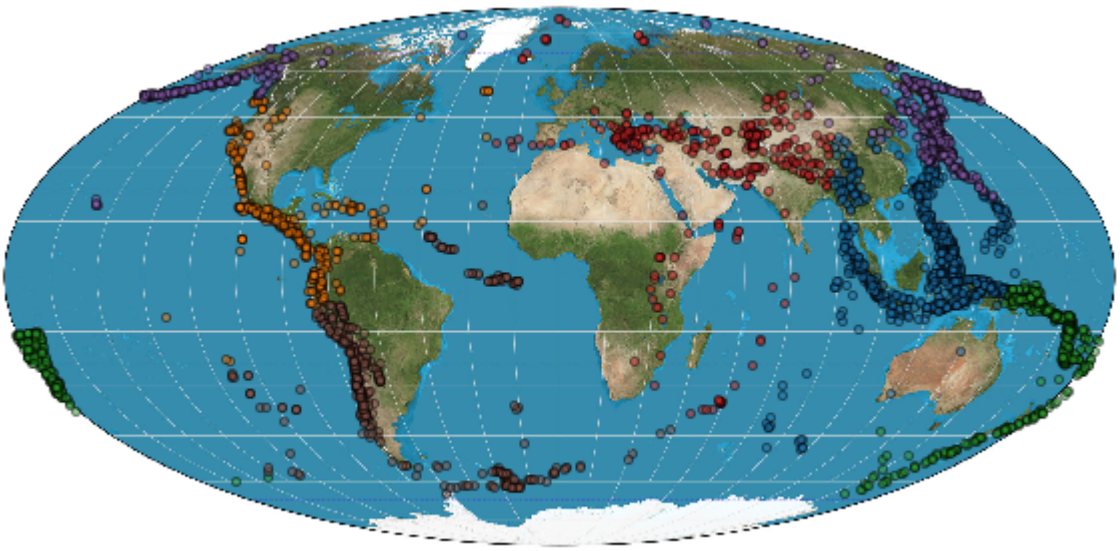

Figure 5: World Map Representation Using K-Means Clusters Classification with K=6

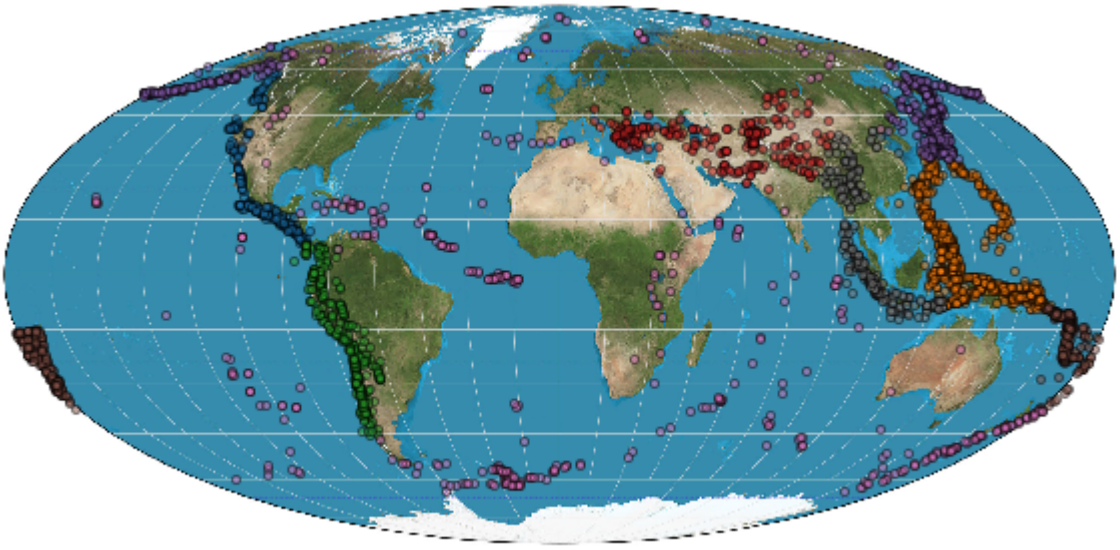

Figure 6: World Map Representation Using Gaussian Mixture Clusters Classification with C=8

We then evaluated the DBSCAN clustering algorithm using the same method as the K-Means, and Gaussian Mixture to fin the optimal value for the parameter Eps.

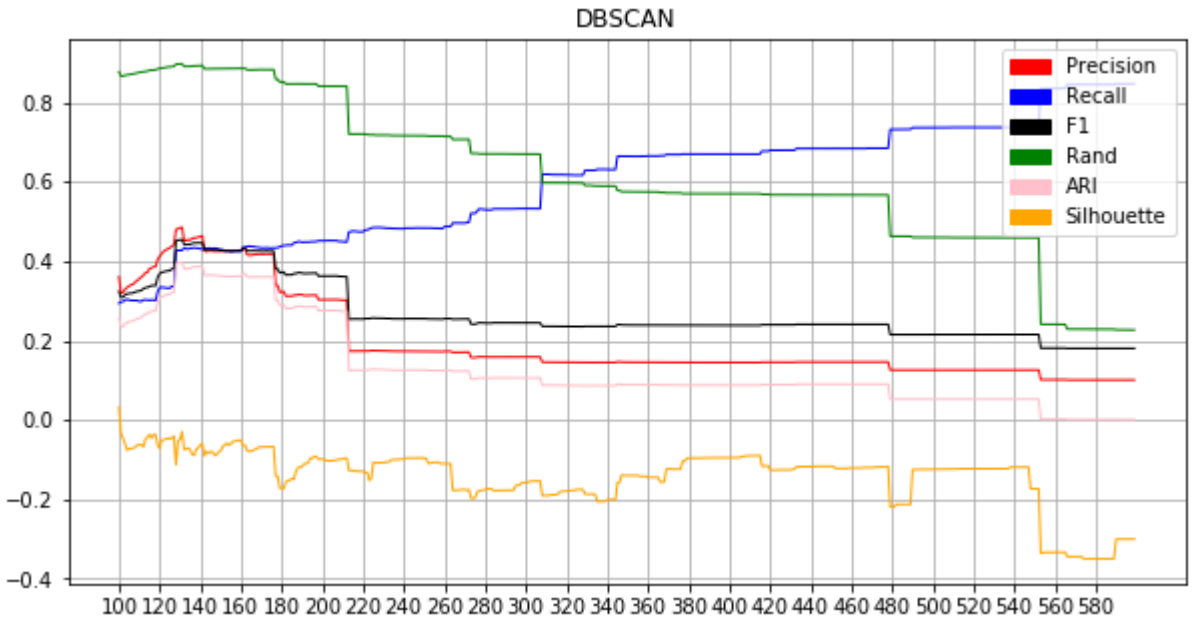

Figure 7: DBSCAN Clustering with 100 to 600 neighborhood radius

As we executed the DBSCAN algorithm for values of Eps between 100 and 600, we verified that between the values of 200 and 220 the values of indexes suffer a drastic decrease and the stabilize for the rest of the iterations of this algorithm so we also decreased the window of results setting the interval of the number of radius of the neighbourhood between 100 and 220, allowing us to evaluate and select the optimal Eps much more efficiently.

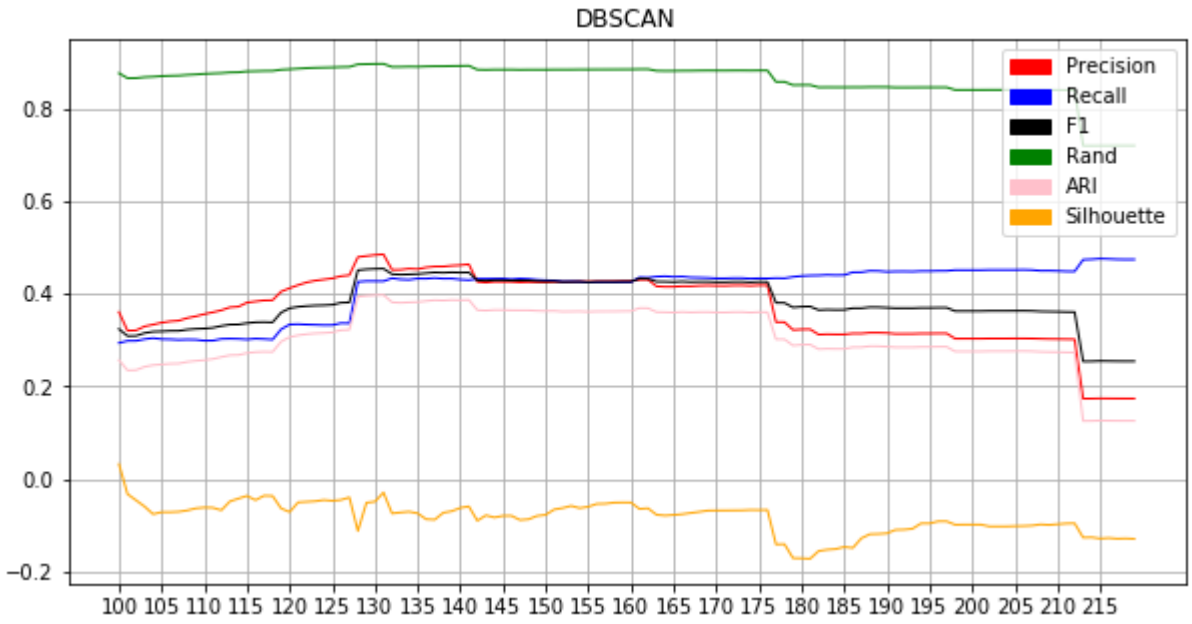

Figure 8: DBSCAN Clustering with 100 to 220 neighborhood radius

The precision, f1 and ARI scores have all very similar behaviours and values, their values vary throughout the iterations in the same way. Their values decrease a little as we increase the neighborhood radius, this means that the value of the false positives increases meaning that the value of true positives is decreasing so it explains why the ARI (being calculated through Rand which is heavily influenced by true positives) would also decrease as well as the F1 score. They all have similar values between 0.5 and 0.1.

The Rand score starts really high and starts dropping as the value of radius starts increasing, which means that many clusters may be wrongly merged increasing the number of false positives while the number of true positives decreases, these weigh heavily on the value of rand decreasing it significantly throughout the increase of radius.

The recall score increases significantly as we increase the radius of the neighborhood in a contrary fashion to rand, as we increase the radius the number of false negatives decreases because more clusters are being merged and it increases the chances of similar points belonging in the same cluster.

The silhouette score really surprised us because of its negative value all throughout the iterations, this means that the samples have been assigned to the wrong cluster,

meaning that a cluster have more similar examples, these can be explained in that high density areas in the map can be from many different faults and it's hard to have a radius that only catches similar seismic events. The silhouette score only has the distances between each seismic event to confer similarity too, which means that although the distance between seismic events weighs on the fault it is associated too, in highly geological unstable areas many faults are present and it can be hard to evaluate any degree of similarity between seismic events, which the silhouette score tries to do not using the best features.

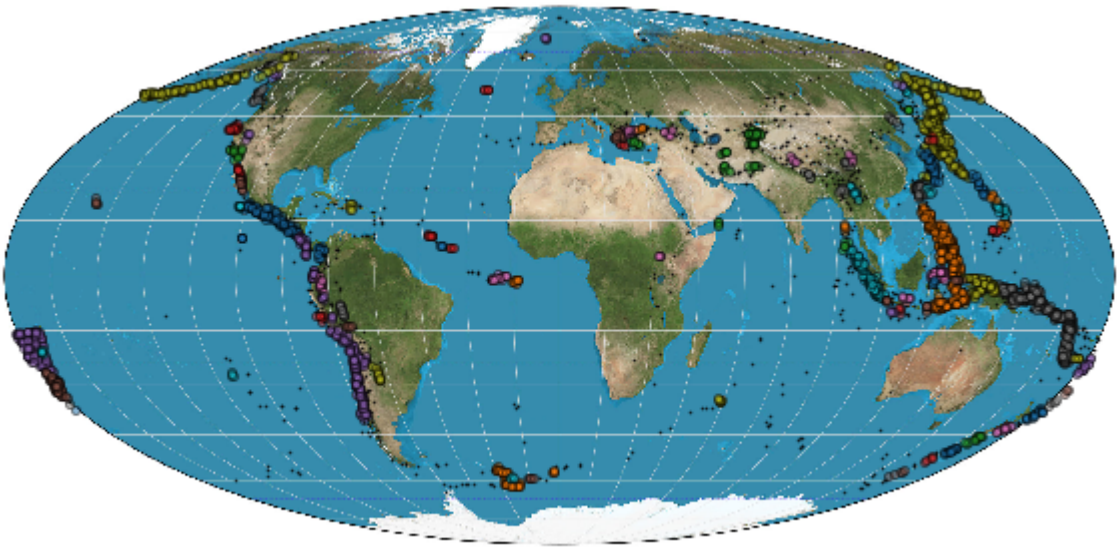

Figure 9: World Map Representation Using DBSCAN Cluster Classification with Eps=130

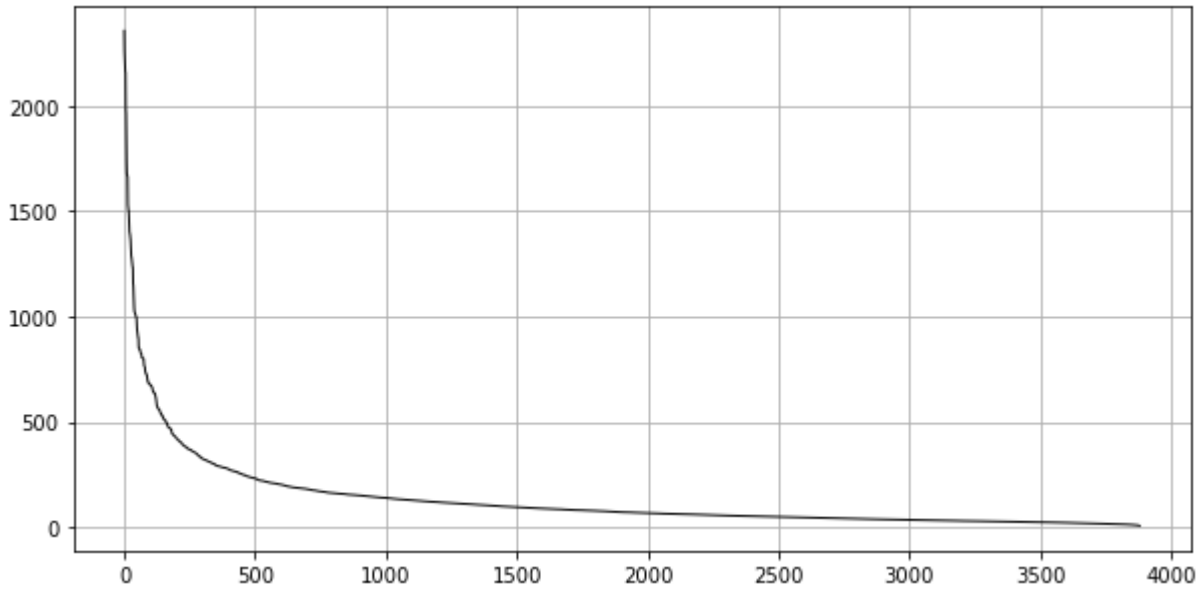

Figure 10: Graph of the elbow in the DBSCAN algorithm

This graph represents the distance of every point to its forth nearest neighbour, the interest to plotting this graph is to find the threshold that represents the maximum value possible of the radius of the neighbourhood, but maintaining that value to only affect the high density region of its own points and not crossing other high density regions avoiding merging unwanted clusters.

After we plotted this graph as you can see the value which corresponds to the elbow of the graph is approximately Eps = 470, we then plot the clusters in the Mollweide Projection. We can verify that the results obtained for the optimal value of Eps using this method are very different than the results obtained by the observation of the value of the indexes.

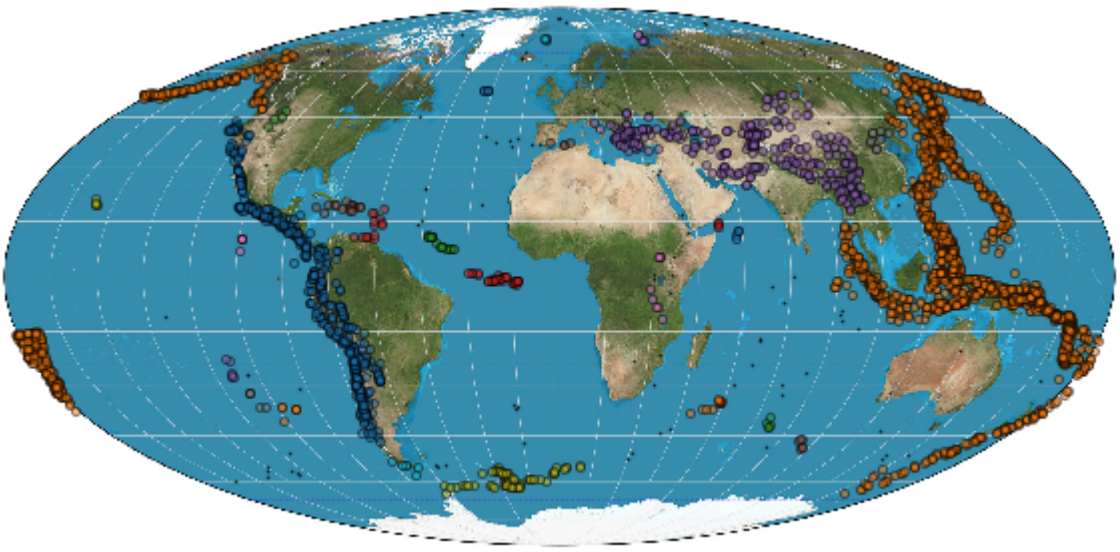

Figure 11: World Map Representation Using DBSCAN Cluster Classification with Eps=470

3 Conclusion

In this project we implemented and examined the performance of three different clustering algorithms in the problem of clustering seismic events. Throughout the practical work we observed and discussed different index values for each clustering algorithm that we used, we used the precision, recall, F1 score, Rand Index, ARI and Silhouette as base values for our conclusions in all the different tests that we performed. We tested the different clustering algorithms with multiple parameters, varying the K in the particular case of the K-Means algorithm, the C in the Gaussian Mixture algorithm and varying the Eps in the DBSCAN clustering algorithm.

In the end of the practical work we conclude that the K-Means clustering algorithm is the best to solve this problem, although the graphical resolves of the K-Means and the Gaussian Mixture are very similar to each other, the K-Means have slightly better results, this and the fact that the K-Means clustering algorithm is more robust to real life values.