Machine Learning - Report 2

António Ferreira & Ricardo Esteves

December 9, 2017

Abstract

The goal of this assignment is to examine the performance of three clustering algorithms in the problem of clustering seismic events. The algorithms that we are going to use, are the following: K-Means, Gaussian Mixture Models, and DBSCAN, which were lectured on the Machine Learning classes. We are provided with the seismic events data in a .csv file. Based on this, we will analyze and examine the performance of each algorithm resorting to five internal indexes and one external index and retrieve conclusions from this analysis.

1 Introduction

As stated above, we were asked to use three clustering algorithms: K-Means, Gaussian Mixture Models, and DBSCAN and examine their performance in the problem of clustering seismic events. One of the most challenging parts of this assignment is how we will implement the discovery of the best epsilon value based on the paper, "A density-based algorithm for discovering clusters in large spatial databases with noise (1996)" published by "Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu" as well as comparing and deciding which indexes are adequate or inadequate for this type of problem. Lastly, we will use the Silhouette as our internal index and the Adjusted Rand Score (ARI), Rand, Recall, Precision and F1 as our external indexes to measure the performance of our three clustering algorithms while comparing each other and draw conclusions about them.

2 Should we standardize and shuffle the data?

Shuffling the data works well when we want to reduce the variance and make sure that our models remain general. To understand better our point of view, we give an example of when we should shuffle our data. For instance, when we are given sorted data, we will shuffle the data to guarantee that we are not training a close range of values of the data and thus obtaining a homogeneous distribution of the data. As our seismic data is related to real-world events and is not sorted, we decided not to shuffle our data.

Regarding data standardization, we also decided not to standardize our data. To explain our point of view we will step down to the purpose of standardization. Standardize will rescale our data to have a mean of zero and a standard deviation of one, which means we will have a homogeneous distribution of our data. In our problem, which is a problem of clustering seismic events, standardizing our data, will mean changing the latitude and longitude variables of our data points, therefore changing the distances between each point. These modified distances will then be used by our algorithms to group each event and these changes (standardizing the data) will give us a result that tends to divert from the original cluster of the data set as we are changing the location of each seismic event around the globe.

3 Explanation of the three clustering algorithms

3.1 K-Means

K-Means [3] is a partitional and prototype-based clustering method. In a simple approach, K-Means consists in dividing the data set into k clusters, each one defined by the mean vector of the points of the cluster, which is the prototype of that cluster. Each point belongs to the cluster represented by the closest prototype. One particular thing about this algorithm is how we discover the best k value. To do that, we need to choose a range of k values, in our case we chose the values 2 and 110 for the lower and upper bounds respectively. The lower bound choice of 2, was made because the Silhouette and ARI indexes need at least 2 clusters in order to compute. To the upper bound we choose a value of 110 as a value that is substantially higher than the number of faults. Now that we have set a range of values for the variation of the k value, we will plot all our indexes values that were obtained by computing the K-Means algorithm in all of the k values ranges defined above. To find the best k value, we want to find the k value where the indexes values reach their maximum. In conclusion, K-Means uses the distance between the cluster center and the respective points to associate a point to a cluster. This leads to a poor performance in real-world problems, but if we have a simple and well-distributed data K-Means finds a proper result and is a strong tool.

3.2 Gaussian Mixture Model

Gaussian Mixture Model (GMM) [5] has the same objective has the K-Means with a slight difference: GMM uses a probabilistic model that measures the probability of a point belonging to a given cluster, in others words, the probability of assigning a point to a cluster. This difference makes GMM better than K-Means in multiple ways. GMM supports clusters that have different sizes and correlation structures within them. This does not happen in K-Means. Similar to K-Means, GMM divides the data into C components. We chose the best C value like we did in K-Means, as is stated above. One important thing to mention is that when we have low values o C, we could be having a lot of different points inside the same cluster and therefore, having a lot of "miss grouped" points. A similar effect happens if the C value is too high as we could be having different clusters grouping the same points. This also applies to K-Means k parameter.

3.3 DBSCAN

DBSCAN [6] is a Density-Based Spatial Clustering of Applications with Noise, which means: given a set of points in a data set, it groups together points that are close to each other and marks as "noise" points that lie alone in low-density regions of the data set. This algorithm has two main parameters: Epsilon and MinPts. Epsilon is the maximum radius of the neighborhood from a given point p and the MinPts is the minimum number of points in the neighborhood of a given point p, for p to be considered a core point, within distance epsilon. We should also note that we have two type of points: the core points and the noise points, the core points is the points which have at least MinPts points within distance epsilon from itself, and the noise points are points that have less than MinPts points within distance epsilon from itself. Since the algorithm depends on these two parameters is important to choose them wisely. Therefore we will extrapolate them via the method proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in the paper "A density-based algorithm for discovering clusters in large spatial databases with noise (1996)". Regarding the value of Epsilon it is important to mention that if we choose very low values of Eps, a large part of the data will not be clustered and if we choose high values for Eps, clusters will merge and the majority of objects will be in the same cluster, in other words, we would have very few clusters covering a large amount of (different)data.

If we look to DBSCAN in a comparative perspective we see that DBSCAN doesn't have a pre-defined number of clusters, unlike K-Means and GMM that we have defined a K and a C respectively. Another

difference comparing this three clustering algorithms is that DBSCAN can detect noise, in other words, has points in the data that do not belong to any cluster, unlike K-Means and GMM were we associate all points in the data set to certain cluster. In conclusion, DBSCAN is a more natural cluster algorithm, because it leaves out points that cannot be assigned to any cluster(Noise), which in the context of our problem is a very strong suit.

4 How adequate or inadequate each score is?

Starting with our one and only internal index that will be approached in this assignment, the silhouette [2] index is a measure of how similar a point is to its own cluster compared to other clusters, and this measure varies from minus one and to one, where higher values mean that the clustering configuration (amount of clusters) is appropriate and lower values mean that the cluster configuration has too many or too few clusters. In our specific problem, this score will be very useful as it will help us determine if the groupings made by our cluster algorithm are getting closer or away from the real associated fault lines.

Regarding the external indexes that we approach on our assignment, the Rand Index is the measure of the similarity between two data clusterings. Its improvement, the Adjusted Rand Index (ARI) [1] is a corrected-for-chance version of the Rand index. The Rand Index has values between zero and one, ARI support values of one and minus one, this happens if the true labels are less than the predicted label. These scores are also relevant for our specific problem because they measure the accuracy of our groupings related to the real associated fault lines. Which means we want higher values to achieve higher accuracy.

$$Rand = \frac{TP + TN}{N(N-1)/2}$$

Recall and Precision are other external indexes, these two indexes are alike. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations, while Recall is the ratio of correctly predicted positive observations to all observations in actual class. Recall and Precision uses a scale from zero to one. Both of these indexes are here because they are used to calculate the F1 index described below.

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

Lastly, the F1 index is the weighted average of Precision and Recall. The F1 has a scale from zero to one when F1 reaches 1 means that Precision and Recall are perfect and very bad when it's 0. This index is also useful for our problem as we intend to maximize the precision and recall results (we want to have had correctly grouped all faults) and this index gives us their weighted average, and therefore it is intended to be maximized as well.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

5 How do we choose the best parameter for each algorithm?

After explaining the goal of each index and their relevance on this problem of clustering seismic faults, we can now analyze and discuss what is the best indexes to evaluate our three clusters algorithms. In all

the presented graphics we did not consider the Precision and Recall indexes in the decision of the best parameter, as they only appear because they are needed for the calculation of the F1 index. Considering that, in K-Means and GMM we considered the following indexes: Silhouette, Rand Index, ARI, and F1. After analyzing the graphic in Figure 1 and considering these indexes we decided that the best value for K is 19. Making the same analysis for Figure 2, that represents the graphic of the indexes for GMM, we considered that the best value for C is to 18.

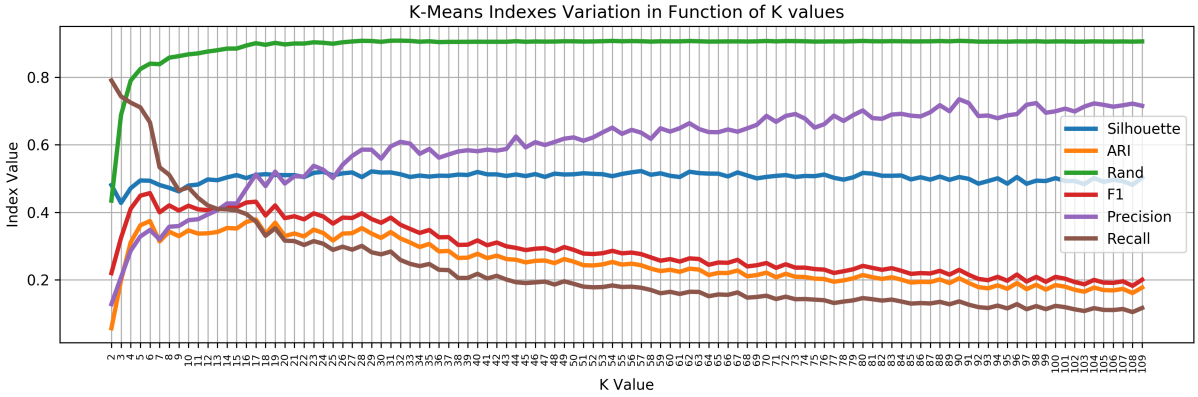

Figure 1: Indexes Values in function of the K parameter in the K-Means Algorithm.

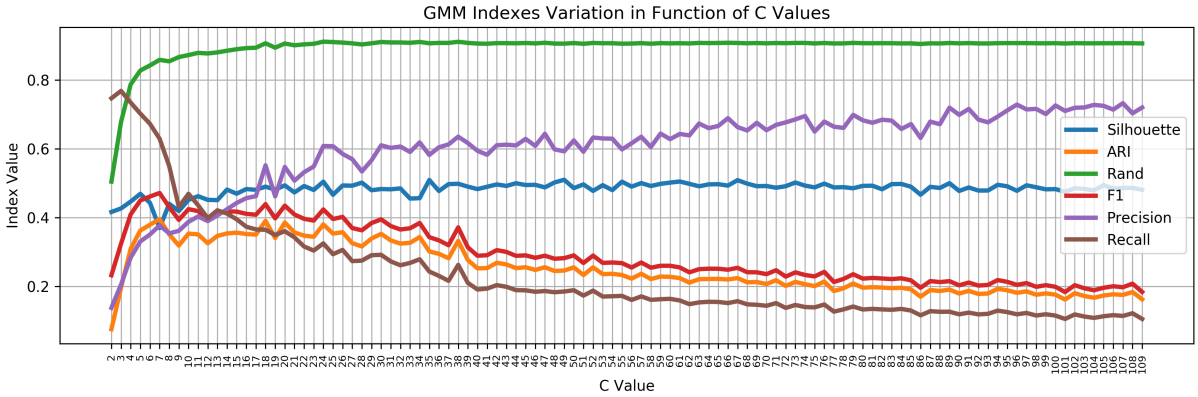

Figure 2: Indexes Values in function of the C parameter in the Gaussian Mixture Model Algorithm.

For the DBSCAN, we considered the same indexes as we did for K-Means and GMM with the exception of the silhouette index, because this index measures the similarity of an object to its own cluster and then compares it to other clusters. As we know the DBSCAN algorithm does not cluster "noise" points which means these points do not belong to any cluster. This is a problem when using this index because it can't be used to its full potential, as there is "missing" data (points that are not clustered). With that being said, after analyzing the graphic in Figure 3, we chose considered that the best value of Epsilon is 130. The choice for all these values is based in the "zone" of the graphics, where we can maximize each considered indexes values.

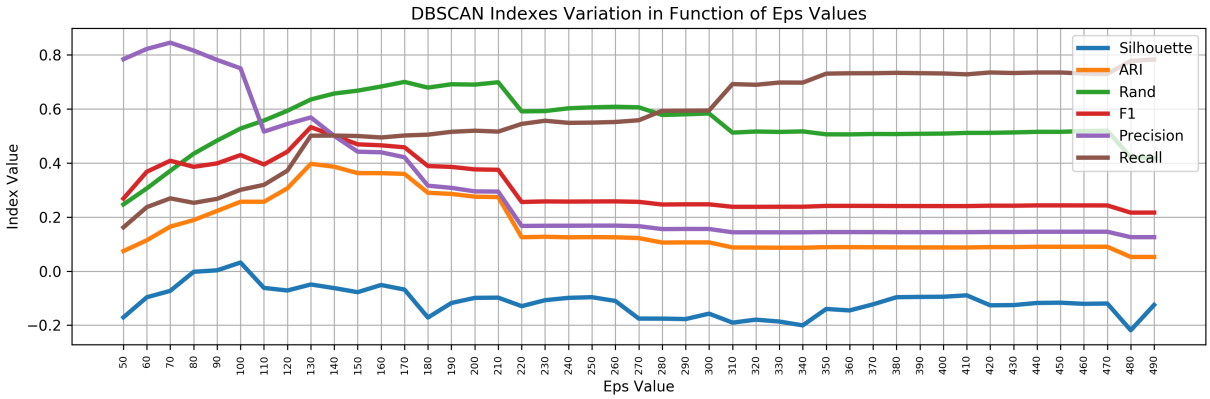

Figure 3: Indexes Values in function of the Epsilon parameter in the DBSCAN Algorithm.

6 What is the best algorithm?

After a deep analysis of our problem of clustering seismic events, we concluded that the best algorithm for this specific problem is DBSCAN. This decision is based in the fact that DBSCAN can detect noise and therefore does not create clusters with those points and because DBSCAN can handle clusters of different shapes and sizes, which in this case of clustering seismic events is very useful as the data is spread around the globe and therefore, we can and will obtain clusters with a lot of different shapes.

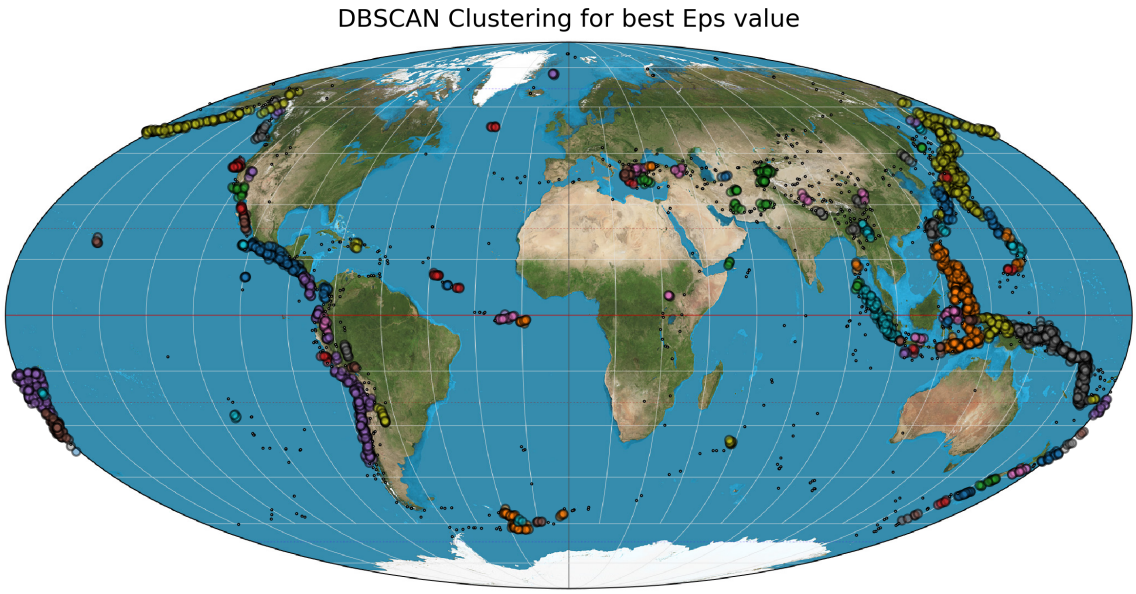

Figure 4: Shows globe vision of the seismic events coloured according to DBSCAN clustering algorithm.

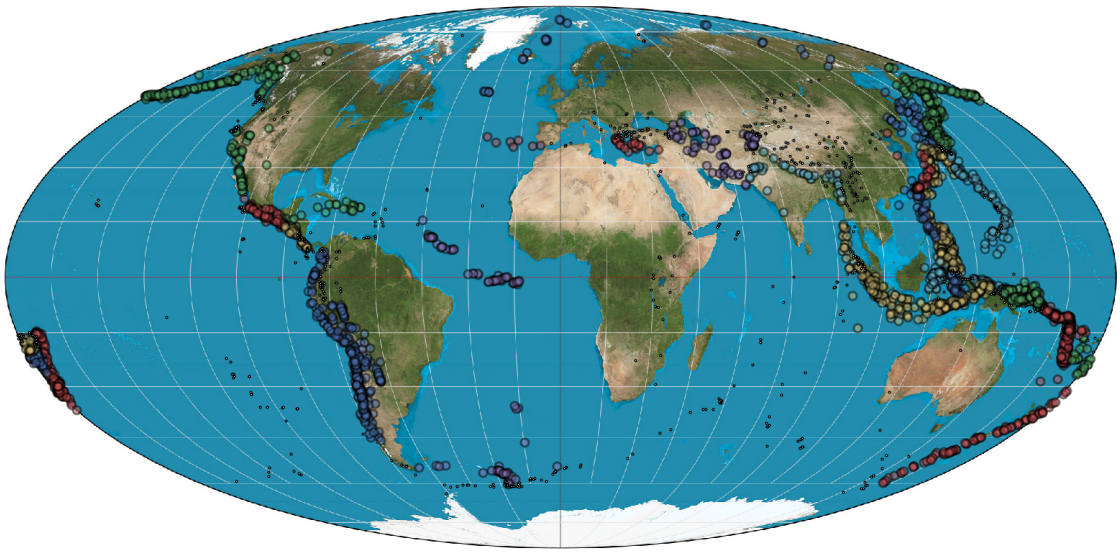

Figure 5: Shows globe vision of the seismic events coloured according to the associated fault line.

As we can see we comparing the two figures above we can see that the one produced by DBSCAN algorithm (Figure 4) is very similar with Figure 5, which represents the distribuition the the faults along the globe. We can easly see that the majority of the noise points are correctly classified as well as the majority of the faults is well grouped in their respective cluster.

7 Explanation of how epsilon is calculated

According to 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" an effective heuristic to determine the parameter *Epsilon* is the following: considering d as the distance of a given point p to its k-nearest neighbour, then for a given k value we define a function k-dist mapping each point to the distance from its k-th nearest neighbours. Then the points are sorted in descending order of their k-dist values and we plot the graph for this function. As the authors named it, this graph is called *sorted k-dist graph*. The optimal value of *Epsilon* will be the **"elbow"** of this graph. As the authors mentioned in the paper, the optimal k value and the parameter Minimum Points(MinPts) is 4 as, according to their experiments, the k-dist graphs for k values higher than 4 "do not significantly differ from the 4-dist graph and, furthermore, they need considerably more computation"(Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu).

With that being said, we now will move on to our implementation (of the method described by the authors) of the *Epsilon* parameter selection. We start by instantiating the KNN classifier provided by the *sklearn* library with the `n_neighbours` parameter equal to 4 (as discussed above). We then proceed to fit the data points and we use a function called *kneighbors* to obtain the distances to the fourth nearest points for each point. After this step, for each point, we save the distance to its closest neighbor from the previously calculated set of four neighbors. Finally, we just need to sort the array of these distances in descending order (the array indexes correspond the point p and each position to the distance between p and his closest neighbor). Finally, as stated above, the value of *Epsilon* will be the **"elbow"** of this graph.

After our implementation, we obtained the following sorted 4-dist graph:

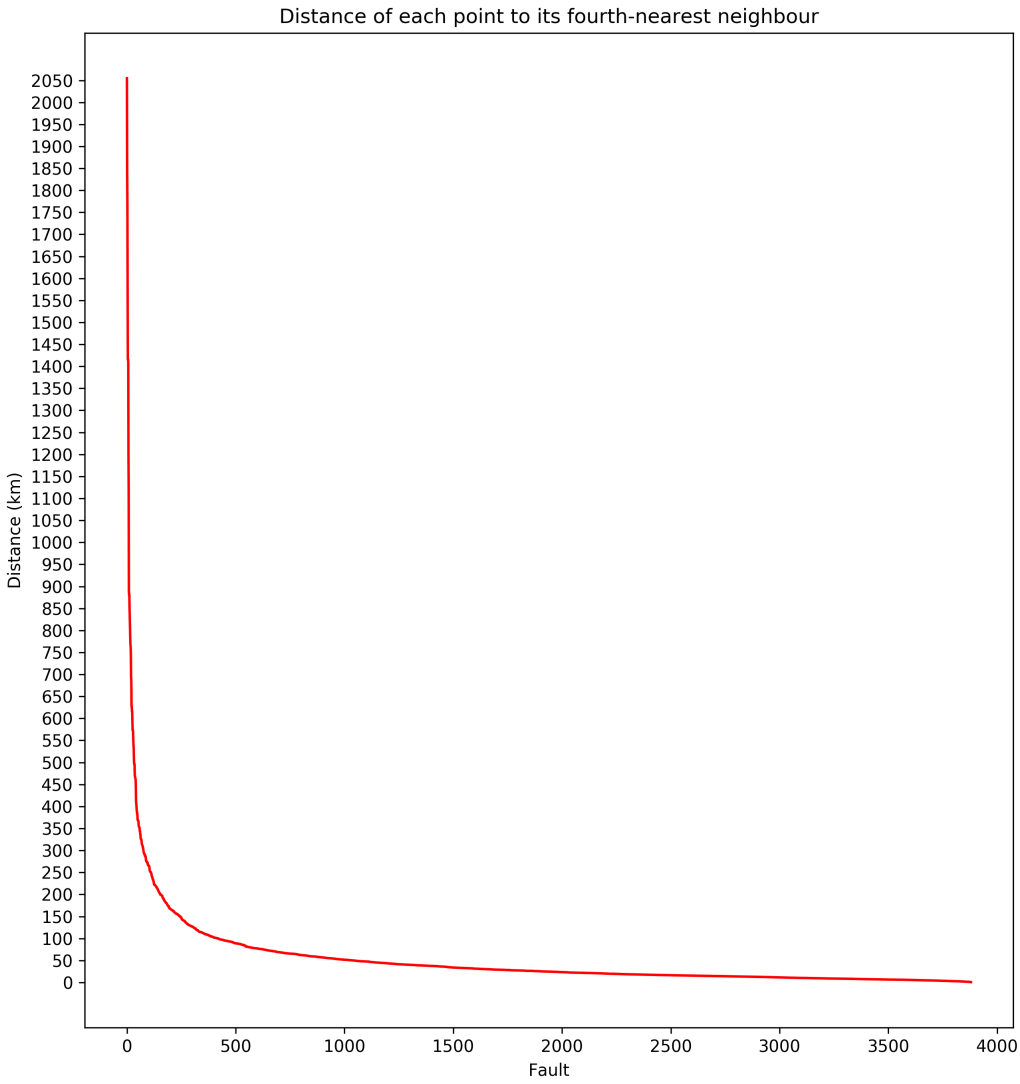

Figure 6: Distance values in function of the points (Faults)

As we can see in the graph above we can visually extrapolate from the "elbow" of the graph an approximate distance value is 130. Although we have determined this value, in the DBSCAN, in order to calculate the indexes, we made Epsilon value vary from 50 to 500. These values we also visually extrapolated from the graph above as the values where the graph starts and stops forming the "elbow".

Finally regarding the method proposed by the authors to obtain the Epsilon value and its relevance in this specific problem of clustering seismic events, we think its a good method to obtain this parameter, because the distance value obtained (Epsilon value, in our case 130), in a reasonable distance value, in other words, not too small and not too high. This reflects in the clustering of each fault, as we can group faults that are closer together and label as noise fault that is not that far away, but in reality, don't belong to the safe fault group (cluster).

8 Conclusions

Due to the nature of this assignment, we can say that we have achieved the main goal of this assignment, which was examining the performance of the three in the problem of clustering seismic events. We approached the problem with a realistic expectation, and so we think that we have reached the final proposed objectives.

By developing this assignment, we struggled while trying to understand what were the most important indexes to be considered to evaluate our three clustering algorithms, but in the end and after a lot of study regarding these indexes and their influence in these algorithms, we bypassed this difficulty and we were able to draw strong conclusions about it.

As stated in the third and fifth sections, we did multiple tests to our algorithms to get the best parameters for each one of them with the help of the various internal and external indexes. We also compared our "Globe Plots" with the ones given in the assignment, leading us to consider that our analysis is correct. After all, we considered DBSCAN to be the best clustering algorithm that best suits our problem of clustering seismic events.

9 Bibliography

[1] Adjusted Rand Score function,
`http://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html`

[2] Silhouette Score function,
`http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html`

[3] K-Means function algorithm,
`http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

[4] Gaussian Mixture Model function algorithm,
`http://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html`

[5] Information about the calculation of the true positives and so on,
`http://dmml.asu.edu/smm/SMM.pdf`

[6] DBSCAN function algorithm,
`http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html`

[7] Video Lectures about this part of the subject,
`https://www.youtube.com/playlist?list=PL2TJmrWpyg7Su-Td5pojMSfjJCmkLbqwv`

[8] Lectures Notes,
`http://aa.ssdi.di.fct.unl.pt/files/LectureNotes.pdf`