Analysis over Wikipedia movie plots dataset

1. **The task**

This project consists of comparing two unsupervised machine learning methods on a specific dataset considering into account the following aspects:

* dataset description
* description of the implemented machine learning methods
* figures and/or tables with results
* comments on the results
* conclusion

In this report, I have decided to study the performances of the **K-means** and **DBSCAN** in the context of movies plots clustering mainly by genre but not only. The analyzed dataset is Wikipedia movie plots.

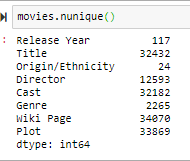
1. **Dataset description**

The dataset used contains descriptions of 34.886 movies from around the world. Column descriptions are listed below:

* Release Year
* Title
* Origin/Ethnicity – Origin of the movie (i.e American, Bollywood, etc.)
* Director
* Cast – Main actor and actresses
* Wiki Page – URL of the Wikipedia page from which the plot description was scrapped
* Plot – Long form description of movie plot

**Data cleaning and feature engineering**

Firstly, it is obviously that Wiki Page URL does not bring any information in terms of clustering the data, so it was dropped. Secondly, since the purpose of this task was to cluster the data mainly based on plots, the title column was dropped due to large number of unique values as it can be observed below.



Secondly, the number of samples of the dataset exceeded the computational power of the environment used, so it was decided to reduce the number of used samples. First step was to eliminate those rows that contained NaN values on any column. In this way the dataset was reduced to only 27832.

By simply displaying some random values from the column “Genre” it was observed that multiple terms such as: “war-time”, “wartime”, “ww1”,” wwii”, etc. were referring to the same generic category: “war”. To correct this kind of issues multiple filters were used. Also, by performing exploratory data analysis it was noticed that the same composed genres such as “comedy-drama-thriller” appeared in multiple forms: “drama-thriller-comedy”, “comedy-thriller-drama”. This aspect was corrected by splitting these composed genres and sorting them.

Even with these slight modifications the number of the total genres was more than 1000. It is less probable to have a good clustering with such high number of clusters. In order to reduce the number of genres it was decided to eliminate those genres whose cumulative frequency is bigger than 0.957 (when genres are sorted descending by frequency). The remaining genres which had a frequency lower than 0.03 were eliminated which left us to a total of 5 genres. Only 15% of the samples from each genre were used in the clustering process, which results in a total of 2500 samples.

1. Description of the implemented machine learning methods

**K-means**

The KMeans algorithm clusters data by trying to separate samples in n groups of equal variances, minimizing a criterion known as the inertia or within-cluster sum-of-squares. Unfortunately, this algorithm needs as input parameter **K** the number of total clusters which in most of the cases it is not known. The algorithm splits the sample into K disjoint clusters. Initially, K points are selected to represent the center of the clusters, also known as **centroids**. Afterwards, at every iteration, each point is assigned to the cluster with the nearest centroid. After each iteration is complete, new centroids are created by taking the mean value of all the samples assigned to each previous centroid. The algorithm stops when the distance between the old and the new centroids is below a certain threshold or when a fixed number of iterations is implied.

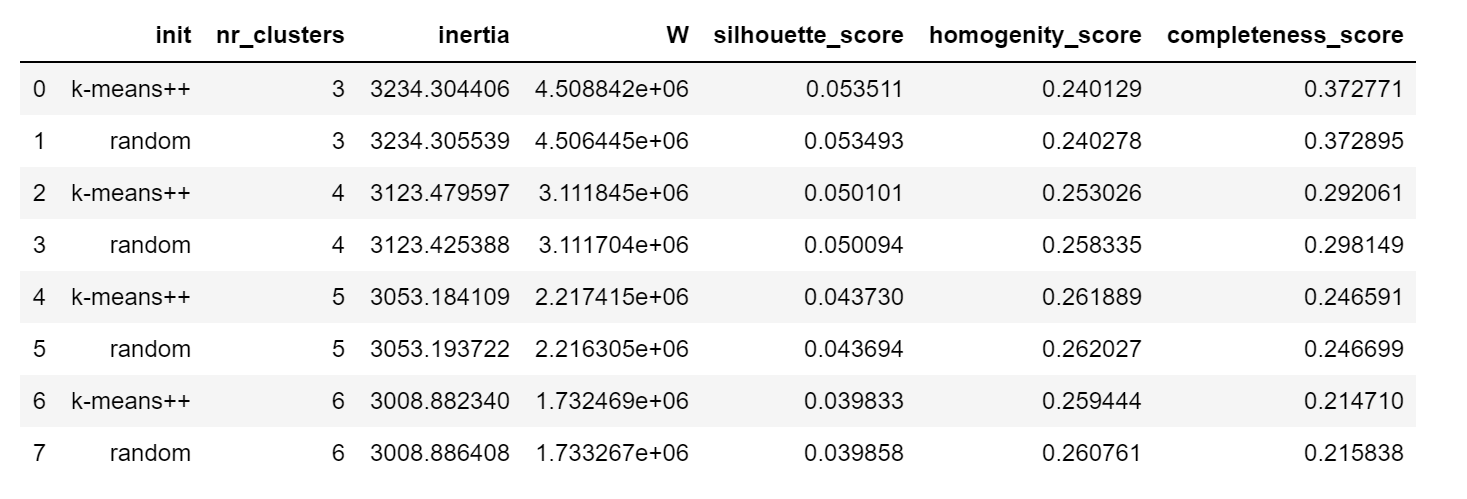
It can not handle properly not convex clusters.

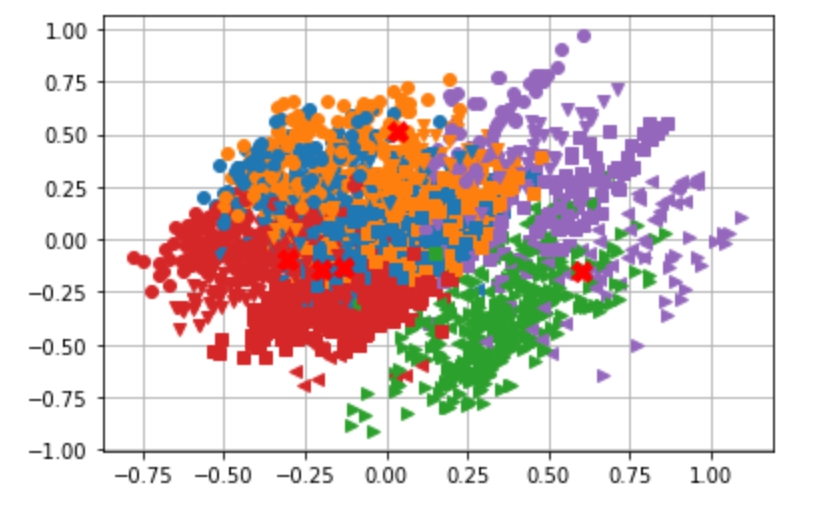
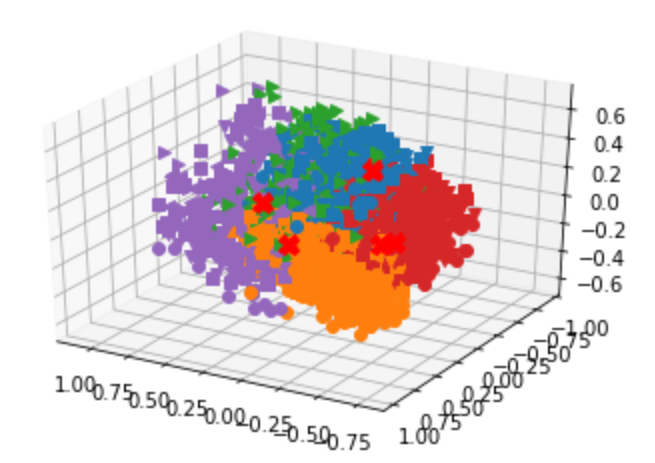
**DBSCAN**

The DBSCAN algorithm represents clusters as high-density areas which are separated by less density areas. Due to this construction, this algorithm does not require the number of clusters before training and can handle data in any form, not only convex clusters. For this algorithm to work it needs two parameters: **min\_samples** and **eps.** These 2 paramers are used to describe a cluster which can be seen as a region where every sample has at least min\_samples neighbors at a maximum eps distance from them. Min\_samples and eps are the noise tolerance of the algorithm since every point that has not min\_samples neighbors at eps radius is an outlier. However, setting min\_samples too low could case too many clusters while setting a too high eps could lead into one single huge cluster. Also, the distance function used it is highly important for this algorithm. Frequently, the Euclidian distance is used but it can cause problems for large features spaces.

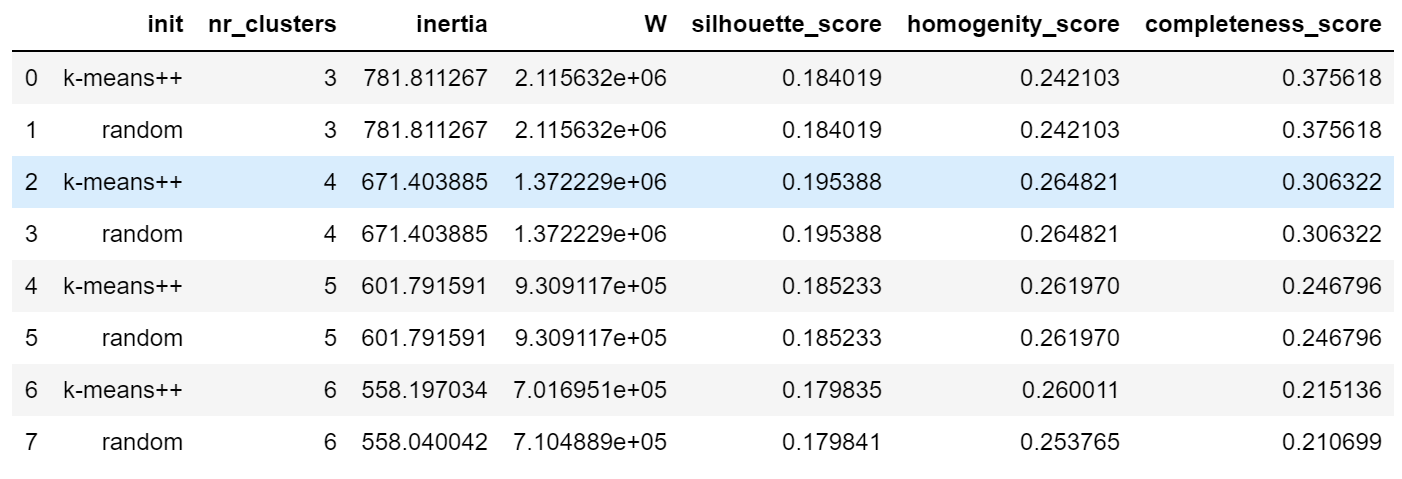
1. **Figures and table with results**

**K-means**

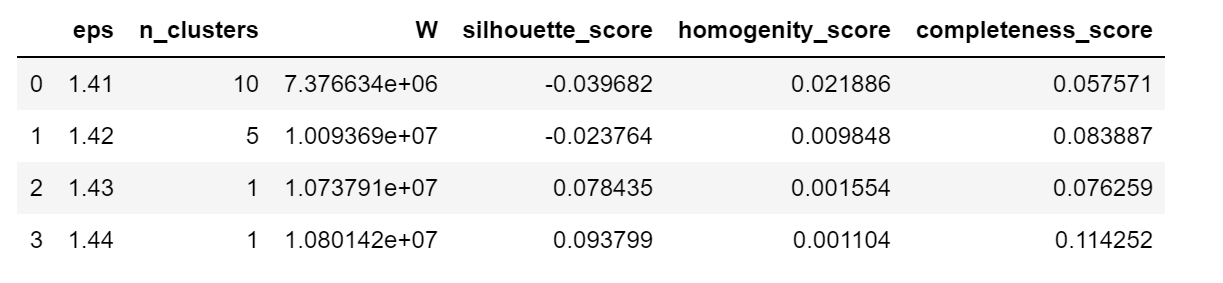


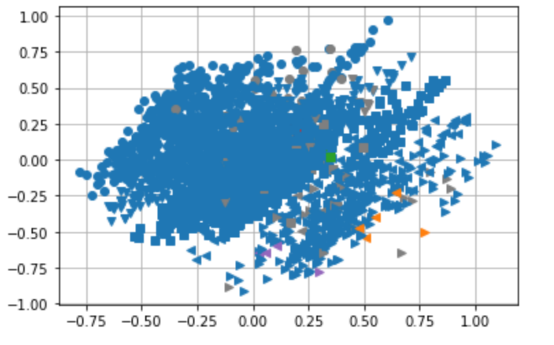
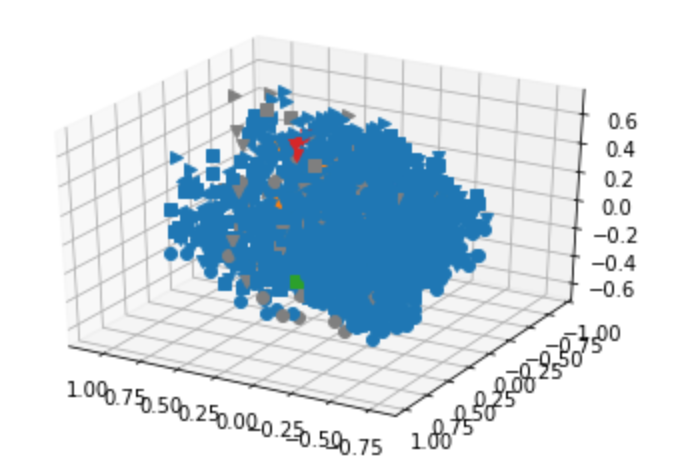
 

**K-means PCA**

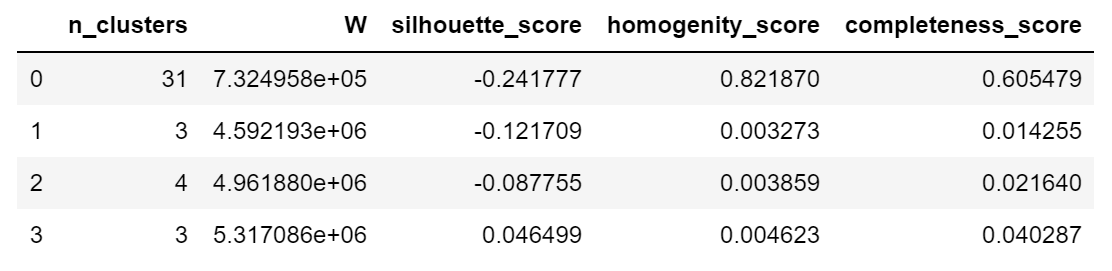


**DBSCAN**





**DBSCAN – PCA**



1. **Comments on the results**

From the **K-means** section tables, it can be observed that K-means does not perform an optimum clustering in both PCA and non PCA versions. The homogeneity score is 25% on average which means that the clusters are overlapping. Same thing happens to completeness score which measure whether all members of a given class are assigned to the same cluster. The silhouette coefficient is better with more than 15% in some cases between PCA and non PCA versions. Since both uses Euclidian distance to measure silhouette score which suffers from curse of dimensionality for large feature spaces the high differences between their score is natural. Even though inertia(within-cluster-sum-of-squares) is decreasing as the number of clusters increase, it can be observed that the completeness score is decreasing which suggest that the clustering is still not capturing the right structure. From the plots it can be seen that labels are mixed up.

Also, when it comes to comparison between the genres provided by the K-means and the ground truth, the K-means with 5 clusters obtains 70% accuracy.

Compared to K-means, **DBSCAN** does provide poor results for the task analyzed. From the DBSCAN section plots, over 90% of the samples are composing one single cluster which is not the outcome desired. The results are not improving when using the PCA version. The negative silhouette coefficient points out that the realized clustering is quite incorrect. Also, both homogeneity score and completeness score relatively close to 0 suggest the poor results of DBSCAN considering that one cluster holds over 90% of the total samples. Even slight modifications of the **eps** parameter changes the number of clusters dramatically as we can see multiple leaps between 10, 5 and 1. This could be also due to high feature space and normalization.

1. **Conclusion**

Considering the analysis did within this report, one can conclude that K-means performs much better than DBSCAN for our task. Even though, the silhouette, homogeneity and completeness score could be improved, the K-means gives about 70% accuracy which is promising. On the other hand, DBSCAN groups more than 90% of the samples into one cluster, being incapable of structuring the data. Both algorithms could be improved with a better preprocessing of our plots, since only same basic corrections are applied.