UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

Algorithms For Massive Datasets

Project 6: Clustering

Armenakis Nikolaos
Student ID: V11249 - ERASMUS

Milan, June 2024

Link for the Google Collab Notebook:

https://colab.research.google.com/drive/1QoSx1OdviUFiWd8usONSZ8Zdb5mQrnlY?usp=sharing

The notebook contains sufficient comments on it to explain each of its functionalities.

Version of the Dataset

The dataset version is assumed to be the current one, as the dataset is estimated to be updated annually and has already been updated in February 2024.

Data Organization

The CSVs are read as DataFrames using the read_csv function from pandas library (Cell 3). Only movies.csv, genres.csv and countries.csv are used for the clustering algorithm as they present useful parameters for the implementation.

More specifically, from the movies.csv, only the columns: 'id', 'name', 'date', 'minute', 'rating' are kept. From them, only columns 'date', 'minute' and 'rating' are used as actual parameters for the clustering algorithm. From the other datasets, only the first corresponding genre and country from each movie are kept as one entry of each parameter per movie is thought to be sufficient for the clustering implementation (Cell 4).

After the pre-processing, left join is performed on column 'id' to merge all the used DataFrames together and then features are chosen from the respective names of their columns and with that, the first input X can be created (Cell 5).

Data Pre-Processing Techniques

First, a check for NaN values if performed to every DataFrame by calculating the sum of every NaN instance respectively. Only movies_df contains NaN values, so it gets "cleaned-up" by removing this entries completely.

Then, One-Hot-Encoding is applied to the columns 'genre' and 'country' from DataFrames genres_df and countries_df, as they contain categorical data, using OneHotEncoder from the sklearn library. This will create a lot of columns that only have 1 in the corresponding genre/country of each movie and 0 elsewhere. These "empty" columns will be dealt with after using the PCA technique later in the code.

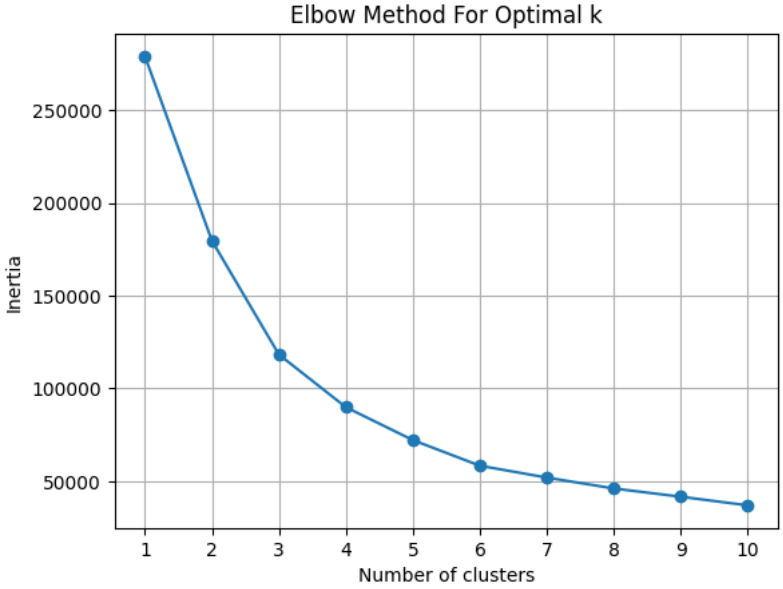
After that, standardization is performed to the data using the StandardScaler from sklearn library and PCA technique, in order to keep 2 components for the KMeans Clustering implementation. The PCA is implemented using the PCA function, again from the sklearn library (Cell 5).

Considered Algorithms, their Implementation and Experiment Analysis

KMeans Algorithm was chosen for the implementation of the solution. It begins by random initialization of a preset number of centroids (clust_num). After that, each data point gets a label assigned based on the distance from each centroid. This happens in the label_assign() function and inside the iteration part of the fit() function. Then the centroids get updated based on the assigned labels. Each new centroid is created by computing the mean of the data points assigned to the current cluster. This procedure can be found inside of the function centroid_upd().

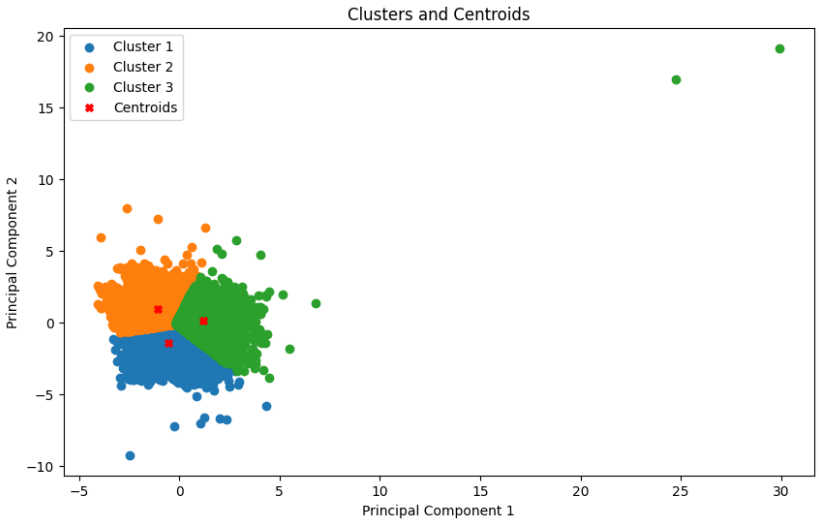
The algorithm iterates until convergence (np.allclose) or reach of maximum iterations (max_iters). The function np.allclose checks whether or not the new_centroids have changed significantly from the pre-existing centroids between iterations. If np.allclose returns True, it means that the centroids have stabilized and the algorithm has converged (in this case the code breaks) (Cell 6).

In order to find the optimal number of clusters, Elbow Method is performed (Cell 7). The Elbow Method helps to find the balance between the number of clusters and the variance within each cluster, known as the inertia. Elbow Method performs KMeans with a different range of values of clusters (clust_num between 1 and max_clusters) and for each value, the inertia value is computed. In this case KMeans is being tested for range of clust_num values [1,10]. After each iteration, the value of inertia (Cell 6) is appended in an array "inertias". Inside the inertia() function, is computed the sum of squared distances to the nearest centroid. The inertia values are plotted and the point at which the curve bends and forms an elbow is considered to be the optimal number of clusters. This point indicates diminishing returns in variance reduction, meaning adding more clusters beyond this point doesn't significantly improve the clustering.

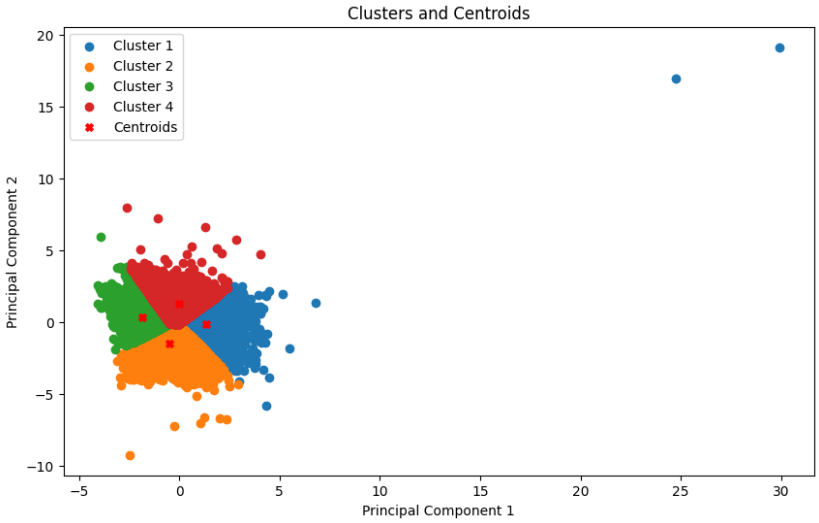


As it seems, in the current plot, the Elbow Point is ambiguous between 3 and 4 clusters. For this experiment, results will be presented for both of these values.

When using the KMeans algorithms, the optimal_number of clusters is set, KMeans is initialized and then the data after the PCA implementation (X_pca) are fitted into the algorithm. Each movie gets its label (cluster) assigned and then the result of the clustering is plotted. The plots of the KMeans results for 3 and 4 clusters respectively can be seen below.



Results of KMeans for optimal_clusters=3



Results of KMeans for optimal_clusters=4

Comments on Results

The data points are all distributed on a similar place on the graph. This may have to do with the fact that the range of values for the different parameters taken from movies.csv is not that variant. For example, dates follow an exponential line in the Kaggle Data Card of the dataset, with most of them being after 2000 [Im. 1]. Also a lot of movies that are older might have been excluded after the NaN check because of missing values in the dataset.

It is also visible that there are 2 extreme outliers that could not be handled with the data preprocessing.

None the less, the clusters seem to have similar number of elements as it is seen below [Im. 2].

Fig. 1

```
Cluster 0: 15195 entries
Cluster 1: 19794 entries
Cluster 2: 27908 entries
Cluster 3: 22202 entries
```

Fig. 2

Scalability of Proposed Solution

Although Kmeans is not capable of handling large datasets by itself (it can be used based on MapReduce), in this particular experiment, the KMeans algorithm seems to be able to cope successfully with the size of the Dataset. There was no subsampling taking place in order for the methodology to be implemented, the only thing that reduced the size of the dataset was the clean up of NaN values. PCA also helped KMeans regarding the dataset size, with the dimensionality reduction of the dataset, but that is also needed for the KMeans to function properly when having more than two parameters.

In case of a larger dataset, Algorithm of Bradley, Fayyad, and Reina (BFR) might have been a more suitable solution to the problem. This is because BFR is designed to cluster large-scale datasets in a high-dimensional Euclidean space, but for this experiment KMeans was sufficient.