Netflix catalogue study using unsupervised learning

Iñigo Maiza Razkin

Universidad Politécnica de Madrid (UPM)

Code of the project available at:
https://github.com/imaiza

Abstract- *In this project, I used a dataset containing all the movies and TV shows available on Netflix as of 2019. This data was mixed with IMDb ratings data to create the final dataset. I applied four different unsupervised learning algorithms, and I was able to separate the data on four clusters. This clusters can be successfully interpreted, and they appear to be really useful. Not only do we gain new knowledge about the data using them, but they also show that a user recommendation model could be built using this clustering.*

I. INTRODUCTION

Netflix, Inc. is an American over-the-top content platform and production company headquartered in Los Gatos, California. The company's primary business is a subscription-based streaming service offering online streaming from a library of films and television series, including those produced in-house. The streaming platform has increased his catalogue substantially in his last 10 years of existence. Netflix has way more films than all his competitors, such as HBO or Amazon video, which are following Netflix`s steps in order to obtain the same success.

An element that is closely related to Netflix is IMDb. IMDb (an acronym for Internet Movie Database)[1] is an online database of information related to films, television programs, home videos, video games, and streaming content online – including cast, production crew and personal biographies, plot summaries, trivia, ratings, and fan and critical reviews. These ratings are created by the users, and will be used during this project study along with the Netflix catalogue.

II. PROBLEM DESCRIPTION

In this project, I will try to use four unsupervised learning algorithms (partitional clustering, hierarchical clustering, probabilistic clustering and affinity propagation) in a dataset taken from Kaggle [2]. This dataset was collected from "Flixable" which is a third-party Netflix search engine. The dataset consists of tv shows and movies available on Netflix as of 2019. Moreover, a IMDb dataset containing more information about the films was added to form the final dataset.

The final dataset was formed by 12 features ('id', 'type', 'title', 'director', 'cast', 'country', 'date_added', 'release_year', 'rating', 'duration', 'listed_in' and 'description') and 6234 instances. Some of these features are categorical, so a lot of feature engineering was needed. I deleted some of the features that could not be helpful in the future analysis ('id', 'title', 'description', 'cast') and the ones that provide redundant information ('date_added', given that we already have the 'release_year' one). A lot of instances also had to be removed, and a lot of missing

values filled. The 'duration' feature also was removed, because we have both TV shows, series and films in the dataset, so it was providing misleading and unhelpful information. To check that the cleaned dataset was clean and didn't have correlated values, the correlation matrix was plotted:

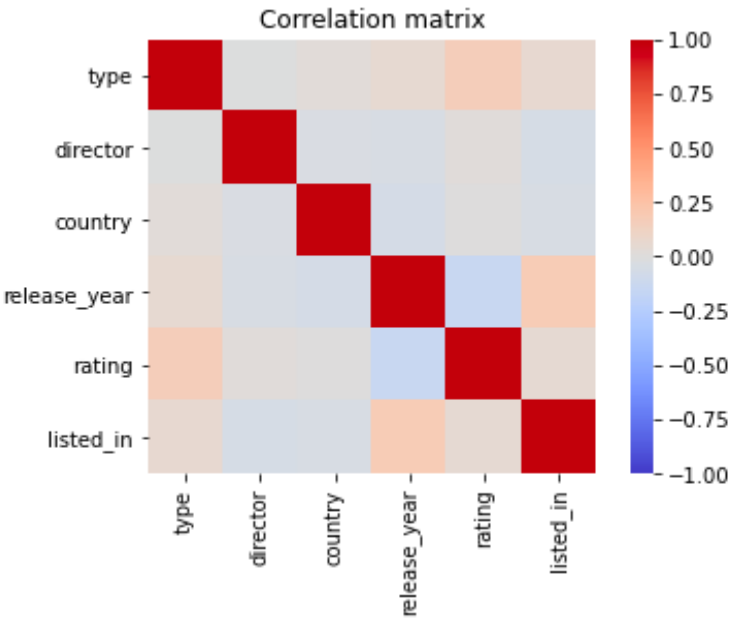

Figure 1: *Correlation matrix of all variables.*

Here, 'type' refers to the type of content: series or movie. 'rating' is the rating of the film/series on IMDb, and 'listed_in' the category where the film is listed (comedia, drama, thriller …).

Some of the variables are categorical (type, director, contry, listed_in), so I had to codify them, giving them numerical values using programming functions, in order to be able to use them in the future clustering. After having all variables as numerical variables, I did a preliminary study of the dataset, plotting all features against each other and the distribution of all them, obtaining figure A1 (*Appendix, figure A1*).

III. METHODOLOGY

This whole project was developed using Python. The clustering algorithms were taken from "sklearn" library [3]. For the visualization tasks, "matplotlib" and "seaborn" were used. "ScyPy" library was also used for some statistics and for the dendrogram visualization.

This work is divided in 5 analysis. The first four parts refer to the different clustering techniques that were applied (partitional, hierarchical, probabilistic and affinity). The goal of this parts will be to perform clustering with each method, to compare the results and to conclude the best clustering method and parameters for our dataset. The last analysis will consist on the model interpretation. Here, we will see how the model works in our data, interpreting the clusters and the hypothetical practical uses of them.

IV. RESULTS AND DISCUSSION

Before getting into the clustering methods, it is worth pointing out that in order to obtain the best possible results, a transformation was applied to the dataset. As we can see in *Figure A1,* the scale of the features differ from each other significantly, so the transformation "sklearn.preprocessing.StandardScaler" was applied to the dataset. This transformation standardizes features by removing the mean and scaling to unit variance.

IV.I. Partitional clustering

For the partitional clustering method, the well-known k-means algorithm was used. Specifically, the algorithm "sklearn.cluster.KMeans". This method depends mainly in the number of clusters that we impose, so in order to find the optimal number, the elbow method was performed. Here we represent the sum of squared distances of samples to their closest cluster center in the figure 3:

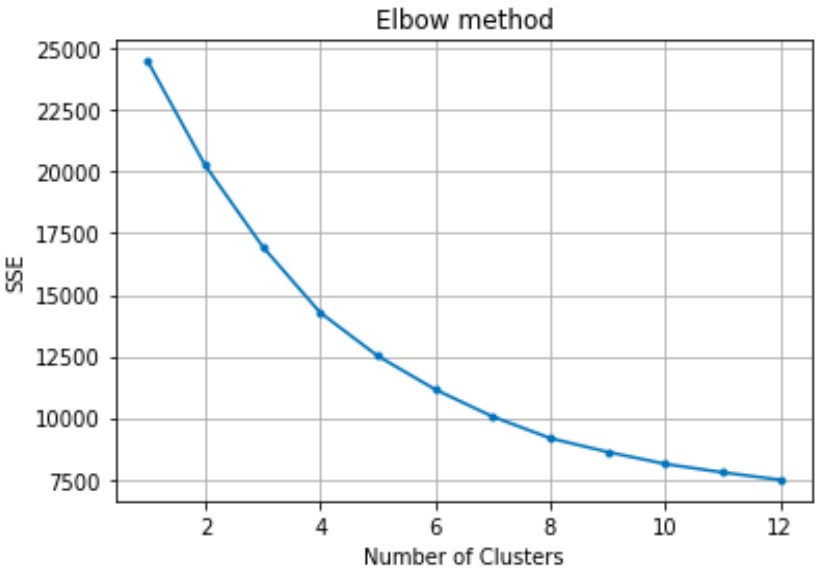

Figure 3: *Elbow method using k-means*

Using this method we see that the sum of distances descends with the number of clusters as expected, but we can't conclude the optimal number of clusters using just this figure. Even though we can't infer an elbow point, we see that for less than 4 clusters the distances are considerably big.

Another method that can be helpful in this situation is the silhouette analysis. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to assess parameters like number of clusters visually. This analysis was performed in order to conclude the optimal number of clusters. The results are plotted in the figure 4. The figure 4 is formed by four pairs of plots. In the left side we find the silhouette analysis and in the right side the datasets' first components (executed using sklearn.decomposition.PCA) where a colour is assigned to each cluster.

All the silhouette analysis have a assigned coeeficient. Silhouette coefficients near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster. The values of our analysis are presented in the Table 1.

After this analysis, given the silhouette score and the way clusters are located, we can think of 4 as the optimal number of cluster by now. N=2 and N=3 seem like really simple models, and N>4 shows a similar results than N=4

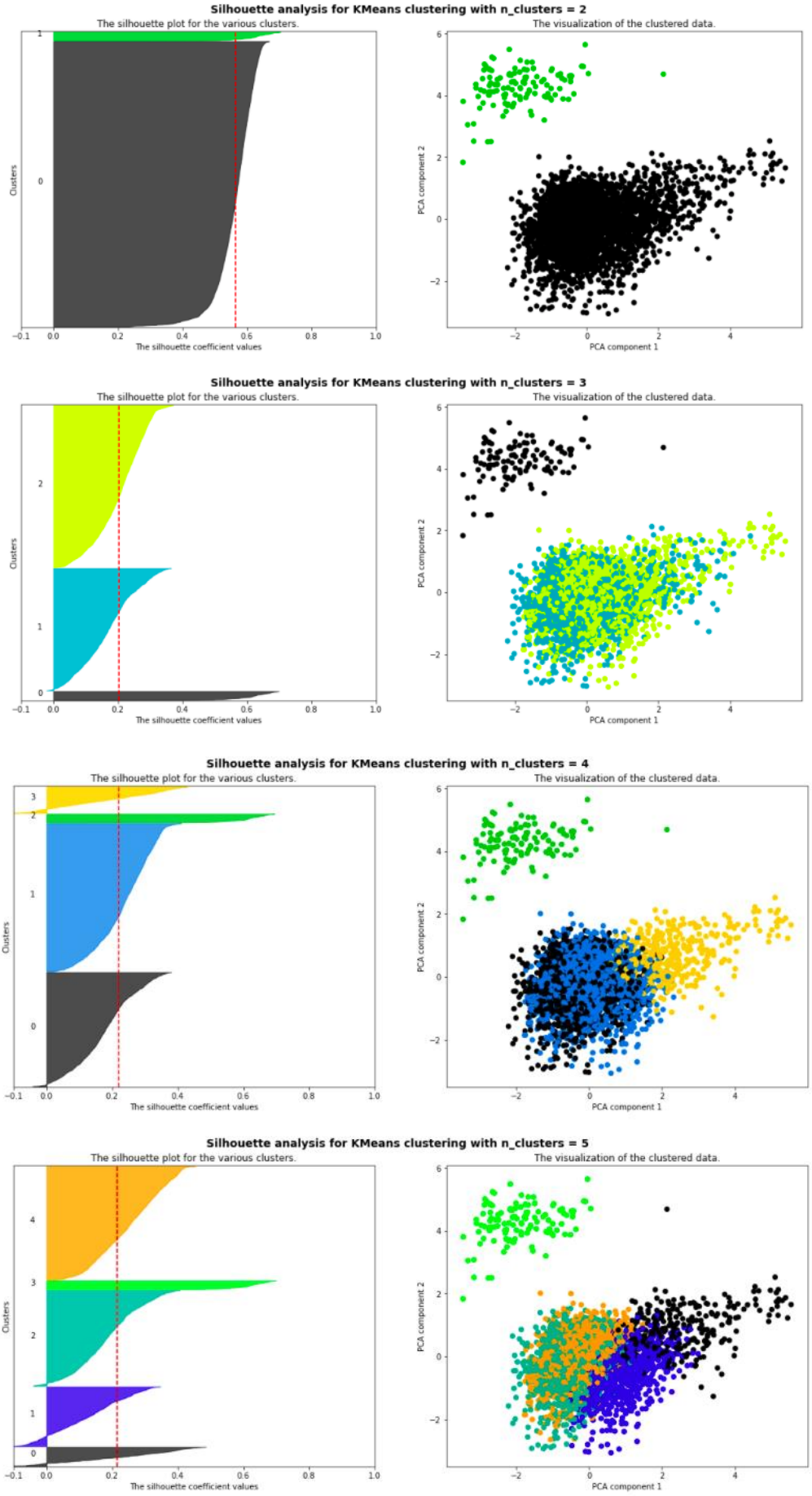

Figure 4: *Silhouette analysis for KMeans with 2,3,4 and 5 clusters.*

Nº clusters	Silhouette score
2	0.564
3	0.203
4	0.221
5	0.214

Table 1: *Table containing the number of clusters and the correspondent silhouette score.*

but with a worst score. Also, this analysis was performed for a bigger number of clusters too, but the results weren't worth to show. In fact, having too much clusters won't improve the model but will make the interpretation more difficult.

IV.II. Hierarchical clustering

Here, instead of using the elbow method I decided to plot a dendrogram (scipy.hierarchy.dendrogram) using the "scipy.cluster.hierarchy.linkage" function to see how are the clusters formed in a visual way:

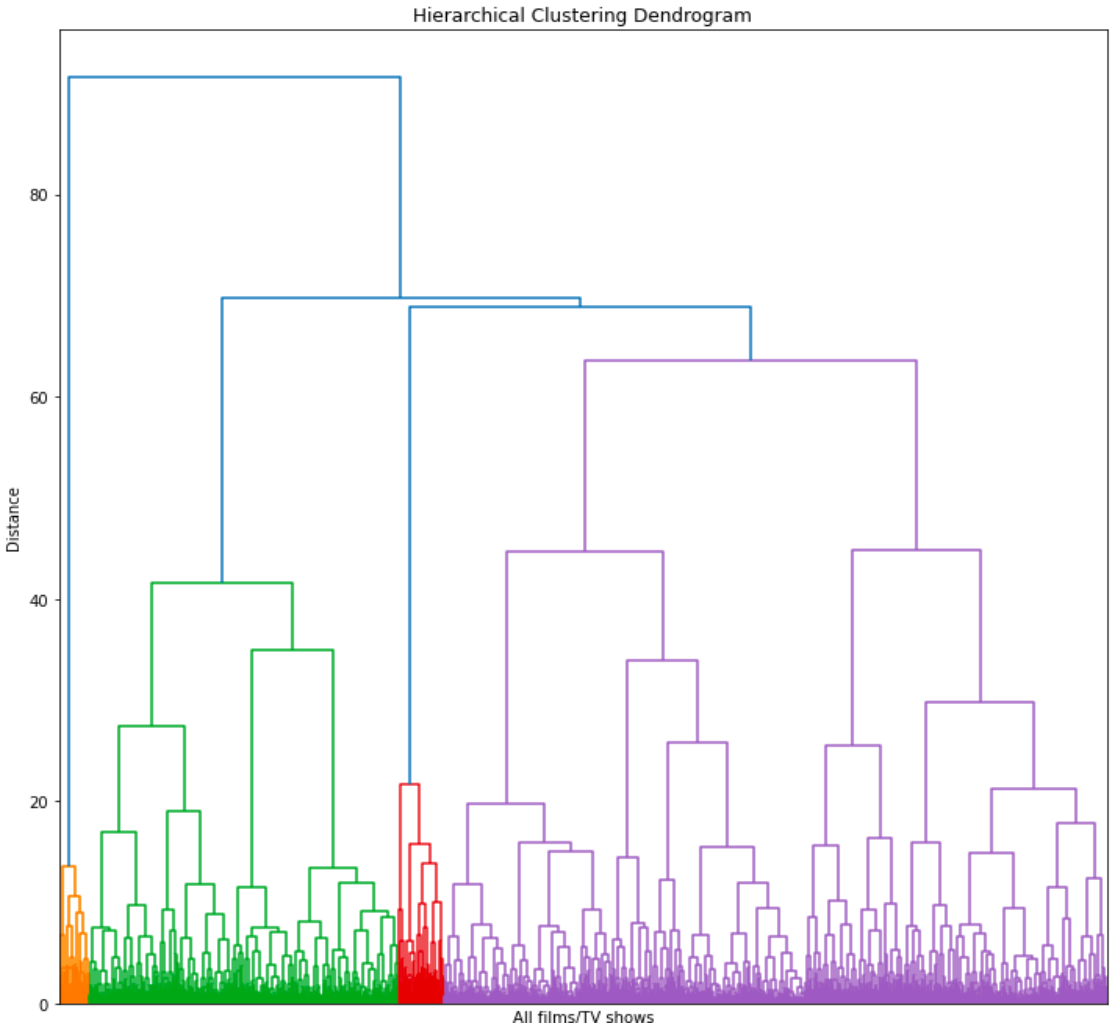

Figure 5: *Dendrogram performed using 'ward' method and 'euclidean' distance as metric. Colors for 4 clusters.*

In this figure we see how cutting the dendrogram to get 3 clusters is really similar to doing it to get 4 clusters. We can repeat the figure 4 performing the silhouette analysis and plotting the clustered data in the principal components. In this case, "sklearn.cluster.AgglomerativeClustering" was used (the results are plotted in *Appendix, figure A2*). The table of the results:

Nº clusters	**Silhouette score**
2	0.563
3	0.147
4	0.173
5	0.172

Table 2: *Table containing the number of clusters and the correspondent silhouette score for hierarchical analysis.*

We see that the results obtained with this method are really similar to the ones obtained using partitional clustering. Even though the silhouette scores are lower in this case, we see again that scores for 4 clusters are better than for 3 or 5. We conclude the same thing with the dendrogram analysis and with the k-means algorithm.

IV.III. Probabilistic clustering

For the probabilistic clustering, "sklearn.mixture.GaussianMixture" was used. The results obtained with this model were really similar to the ones obtained with the two previous clustering algorithms. In this algorithm, the only parameter is the number of clusters. Following previous results we plot 4 clusters:

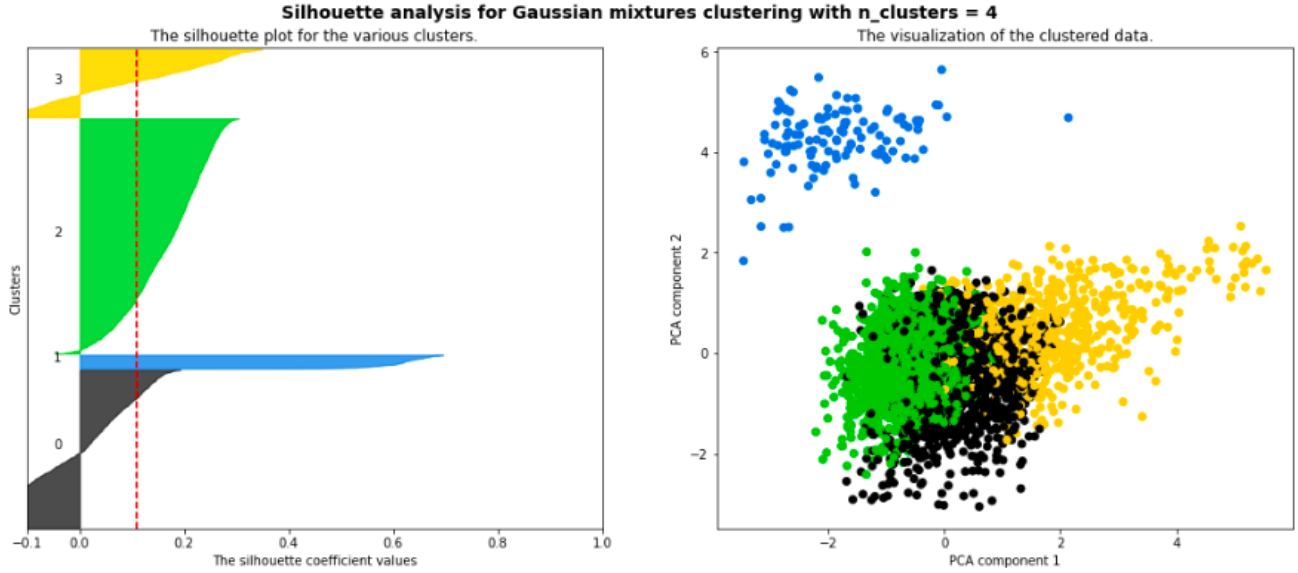

Figure 6*: Silhouette analysis for gaussian mixtures using n_clusters=4.*

The silhouette coefficient in this case is 0.110. The results are worse than the ones obtained with other clustering algorithms, even looking at the clustered scatterplot (the yellow points don't look like properly clustered). The analysis was executed with a bigger number of clusters too, getting similar results. Even though the results are poorer with this method, we can also conclude here that four is the optimal number of clusters for our dataset.

IV.IV. Affinity propagation

The last studied clustering algorithm was affinity propagation, "sklearn.cluster.AffinityPropagation". In this algorithm the number of clusters doesn't have to be defined beforehand, but there are a lot of hyperparameters that have to be defined.

Affinity Propagation can be interesting as it chooses the number of clusters based on the data provided. For this purpose, the two important parameters are the preference, which controls how many exemplars are used, and the damping factor which damps the responsibility and availability messages to avoid numerical oscillations when updating these messages. Using the default values we obtain:

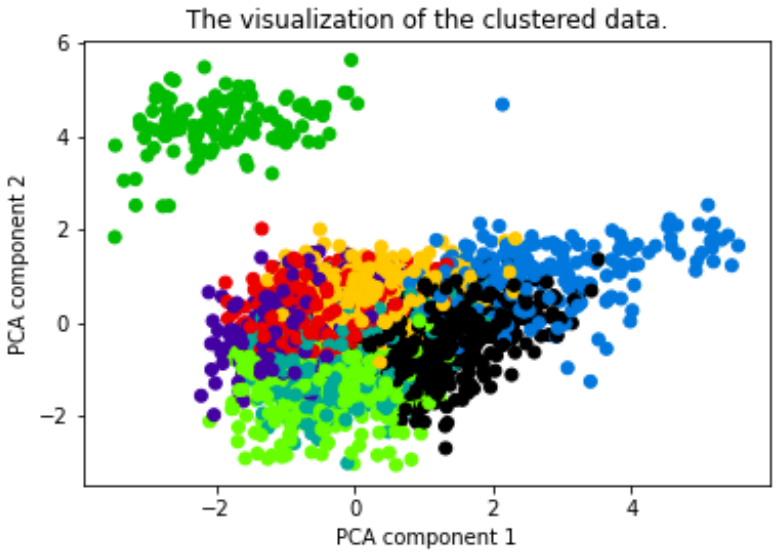

***Figure 7**: Clustered data scatterplot for the affinity propagation model.*

After 93 iterations, the algorithms converges estimating a number of 8 clusters. The results obtained here don't agree with all the previous analysis, and creates a more complex model using 8 clusters. This result varies a lot when tunning the parameters and building a complex model isn't something that we are looking for. Therefore, we discard this algorithm and we keep the models and conclusions obtained in previous sections.

IV.V. Model interpretation

In the above sections, following the results obtained with the partitional, hierarchical and probabilistic clustering we concluded that using four clusters would be the optimal way of clustering our dataset. In order to interpret this clusters and how do they work on the data, we use the previously commented KMeans model and we represent all variables against each other, but colouring each point depending in the cluster they are located in. The results can be found in *Appendix, figure A3*.

In this figure, checking the plot regarding the 'type' feature, we clearly see how the cluster 2 (green) is formed by the series whereas the movies are grouped on the other clusters. We also see that clusters are differentiating between 'country' or 'release_year'. This can be seen also in the distributions of each feature, located at the diagonal of the plot. To visualize these relations between clusters and the features better, we plot boxplots for each feature. We obtain the following:

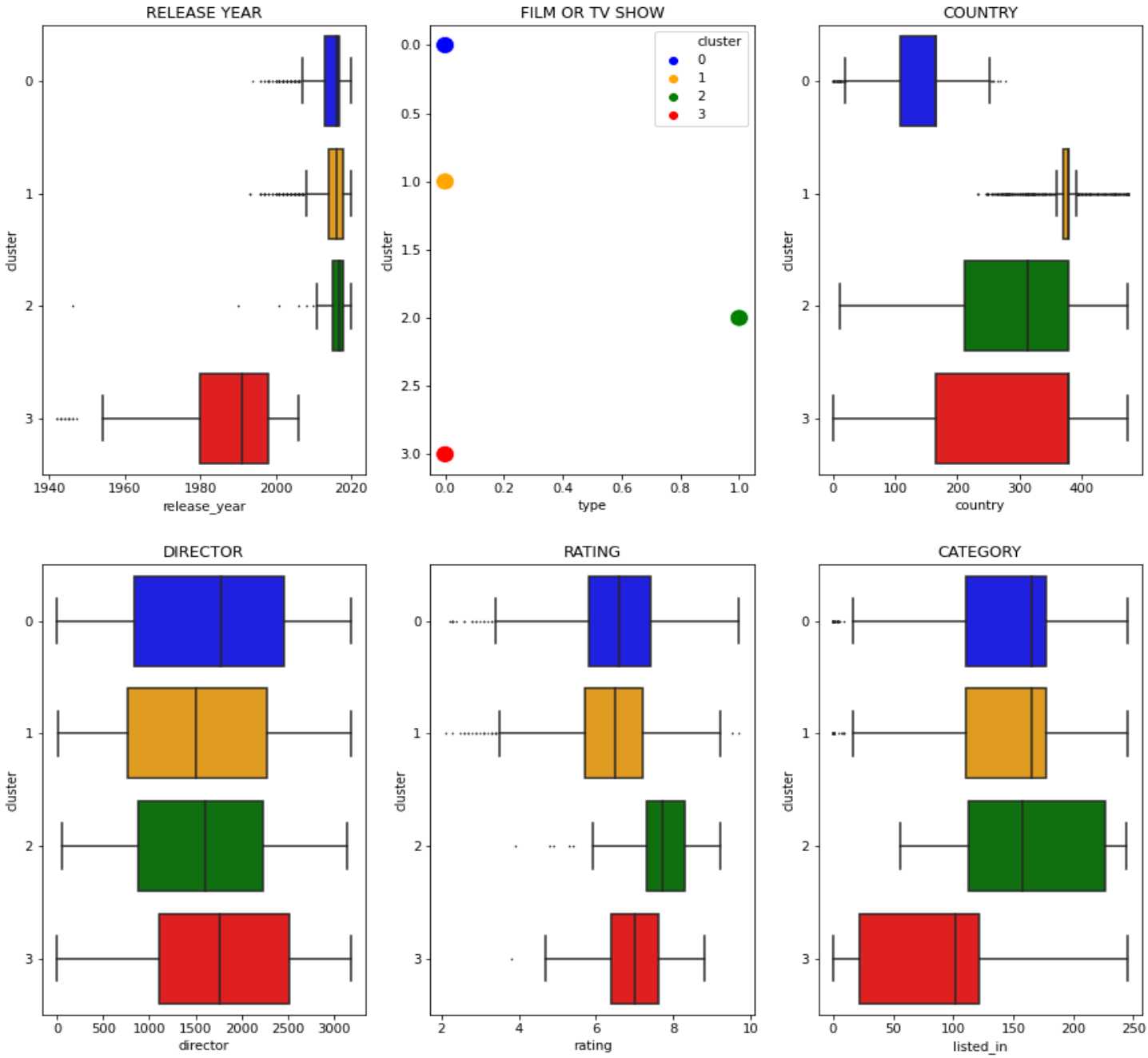

Figure 8: *Boxplots of all the features showing their data distribution in the four clusters.*

First of all, for the 'release_year', we see that the 3rd cluster (red) groups the old movies (majority of movies released before 2005), whereas the other three clusters are formed by newer movies and tv shows. In our dataset, the films released after 2005 represent the 90.8% of the cases. This 9% of old movies, although some of them are part of the clusters 0 (blue), 1 (yellow) and 2 (green) (this clusters present some outliers), are mainly grouped in the 3rd cluster(red).

In the 'type' feature analysis, we have to point out that we codified the variable (so that we could feed the clustering algorithms) so that ***0****: Movies* and ***1****:TV Shows/Series*. With this information, we clearly see in figure 8 how the cluster 2 (green) collects the series while the other three clusters are formed by movies. In our dataset films represent the 68.32% of the instances.

With regard to the 'country' variable, is striking how the 0 (blue) and the 1 (yellow) clusters group a little subgroup of the countries, while red and green clusters group films (red) and series (green) from all countries. To understand what countries are yellow and blue clusters grouping, we need to check how was this variable codified. We see that 'United States' is codified as 379, which is exactly the point where the yellow cluster is centered. In fact, 32% of the films of our dataset are from the US. Checking the values around 100-200 (where blue cluster is

centered) we find out that India, the second biggest film provider (17%) country, is codified as 167. Therefore, we conclude that green and red clusters group series (green) and movies (red) from all the countries, whereas blue and yellow clusters are grouping films around the two biggest film creator countries, India and the US respectively.

In contrast of the rest of the features, in the 'director' variable we don't find evident differences between the directors located at each cluster. In fact, we have more than 3000 different directors in our dataset, and although some of them participate on more than one movie or TV show, it is not enough to create a clustering model that differentiate them.

With the ratings of the films and series something similar happens. We see that the clusters are not formed depending on the rating, but we can get some insights on the data using this representation. For example, we see that series have on average better rating than films. Also, old movies have on average better ranking than new films.

In the 'listed_in' feature, we again have to check the codified values to understand the results. We find out that the films between 50-110 are 'Action & Adventure', 'Classical Movies' and 'Comedies'. These categories are represented mainly by the 3rd (red) cluster. We also see that the other three clusters are centered and get films from the categories between 150-200. In fact, that's the range where the most popular categories are grouped:165-174 categories refer to Documentaries; and 177-190 refer to Dramas (mainly located in green cluster).

V. CONCLUSIONS

First of all, a dataset containing the Netflix catalogue mixed with IMDb data was chosen, and after some feature engineering work and a preliminary analysis we concluded that the features were not correlated and that it was possible to use our dataset in an unsupervised learning context. Then, we performed four analysis using different algorithms, in order to decide what was the optimal way to perform clustering in our dataset.

The first analysis was performed using a partitional clustering algorithm (K-means) and after applying the elbow method and doing an exhaustive silhouette analysis, we concluded that the optimal number of clusters for our dataset was four. In the hierarchical clustering analysis (agglomerative clustering) we obtained the same results. Both the dendrogram visualization and the silhouette analysis showed that four was a coherent number of clusters. In fact, we wanted our model to be as simple and as interpretable as possible. For the probabilistic clustering analysis (gaussian mixtures) same results were obtained, but the silhouette scores or even the clustered data scatterplots were poorer than in the two previous cases.

The affinity propagation clustering estimated a number of 8 clusters. Nevertheless, this algorithm needed a lot of hyperparameter tunning and his results variate a lot in function of these parameters, so we finally rejected the complex model built using this algorithm.

Finally, once we concluded that the three main clustering methods obtain the same result, we built a KMeans model to analyze how does the clustering affect our dataset. In this last analysis, we concluded that the first cluster 2 (green) is formed by series/TV shows while the other three clusters are formed by films. This green cluster is formed mainly by Dramas, and their rating is higher than average. The cluster 3 (red) is mainly formed by movies whose release year is previous to 2005 and the majority of this films are 'Action & Adventure', 'Classical Movies' and 'Comedies'. The clusters 0 (blue) and 1 (yellow) differentiate the films by the country, being the US and India respectively the biggest film producers and the heart of this clusters. We also concluded that the films of these two clusters are mainly Dramas and Documentaries.

We conclude that even if our models are pretty simple, the performed clustering could be really useful to build a user recommendation algorithm.

VI. REFERENCES

[1] https://web.archive.org/web/20050205152519/http://www.terra.es/personal3/dalton22/historia.htm

[2] https://www.kaggle.com/shivamb/netflix-shows

[3] Pedregosa et al., *JMLR 12, pp. 2825-2830, 2011.*

APPENDIX

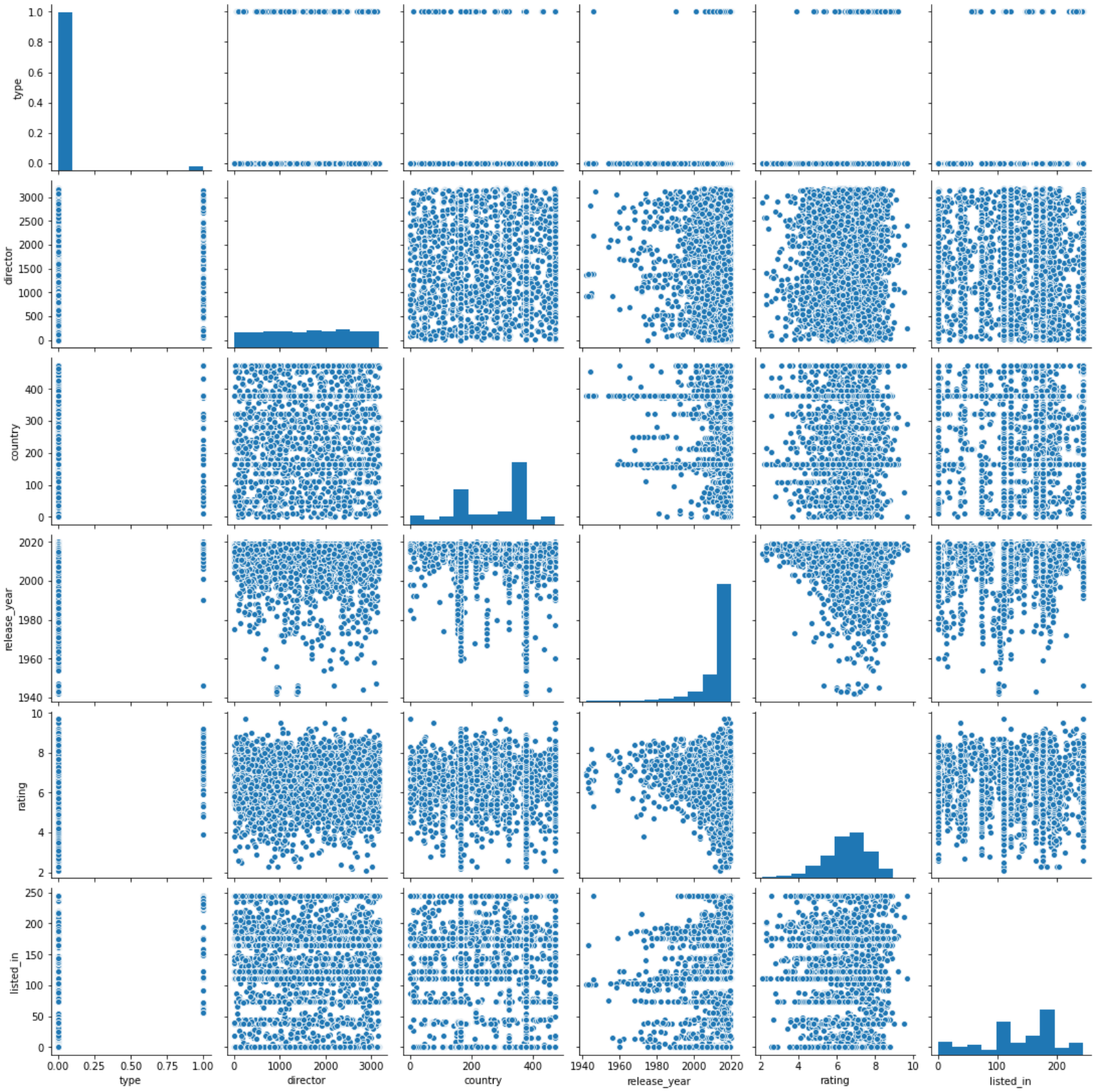

Figure A1: *All variables and their distribution (diagonal) of our dataset represented against each other.*

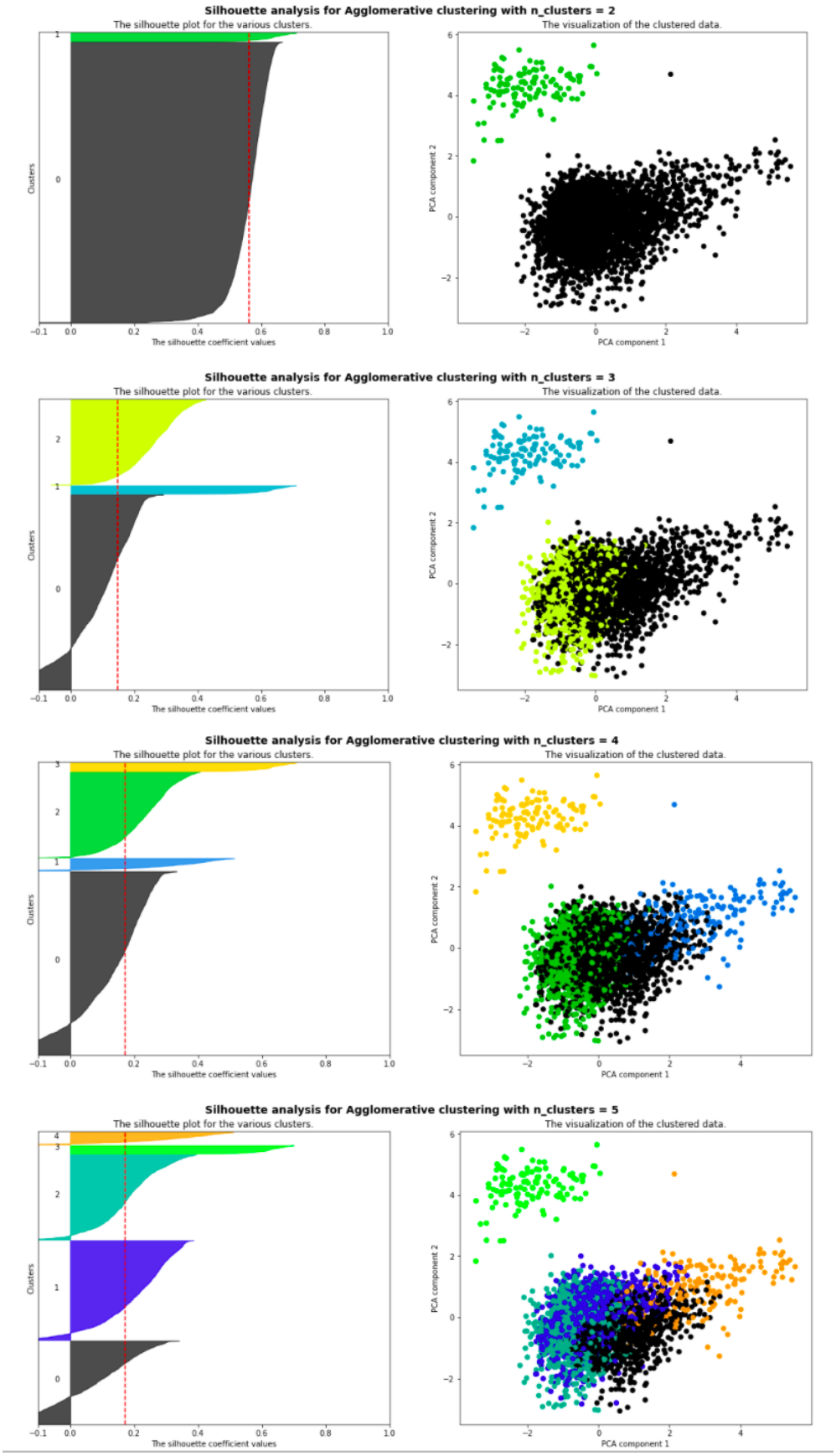

Figure A2: *Silhouette analysis for agglomerative clustering with 2,3,4 and 5 clusters.*

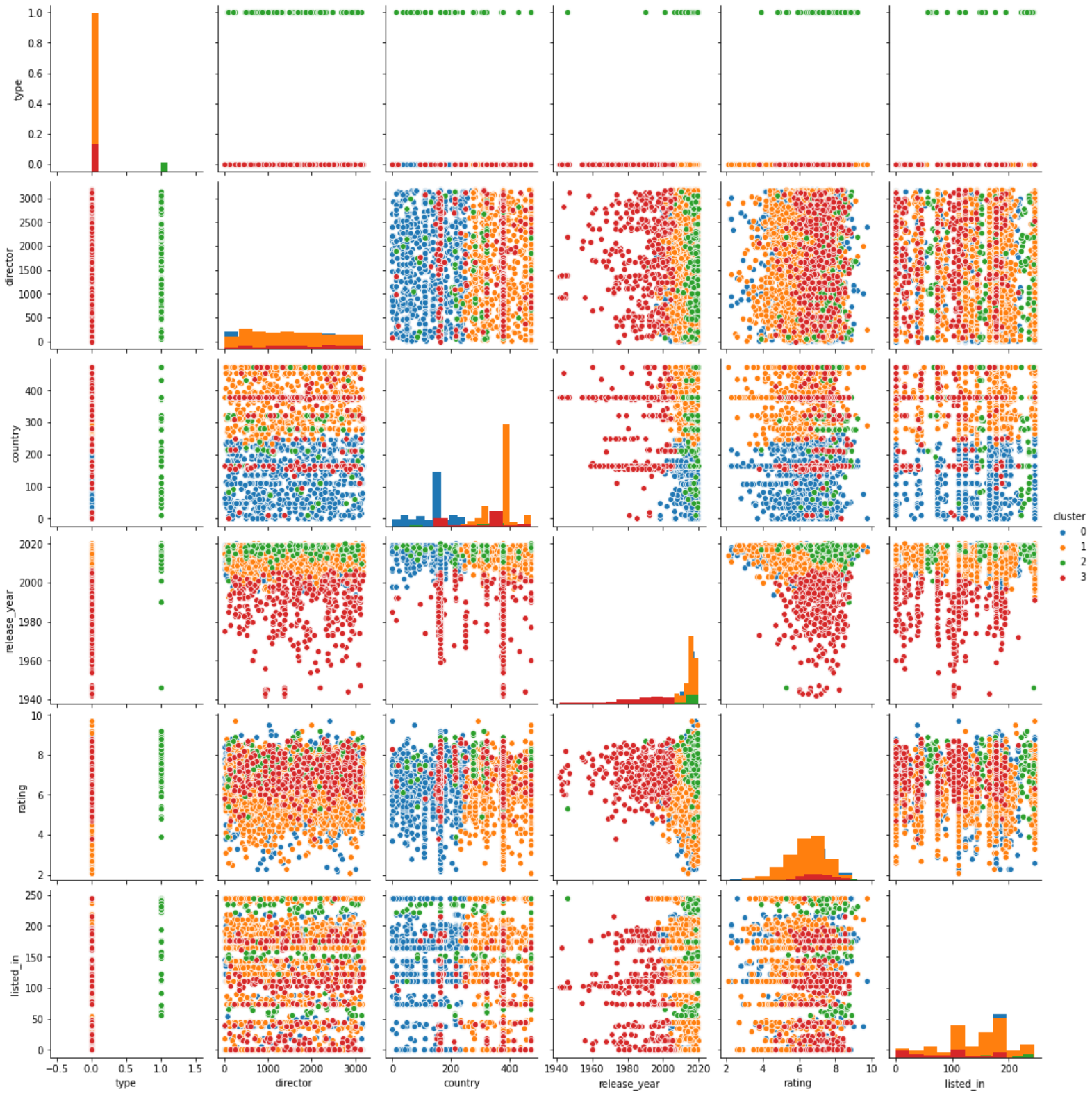

Figure A3: *All variables and their distribution (diagonal) of our dataset represented against each other, differentiated by the cluster they belong to.*