Anime Recommendation System

Gilberto Juárez Rangel
(Dated: June 3, 2024)

In this report, I leverage the 'Anime Recommendation Database 2020' from Kaggle to develop a recommendation model using unsupervised clustering algorithms. I employ three distinct models and compare their performance. Prior to modeling, the data undergoes cleaning and transformation to enhance the system's effectiveness.

I. INTRODUCTION

With the ever-growing popularity of anime, the need for effective recommendation systems has become increasingly important to help fans discover new content that aligns with their preferences. The 'Anime Recommendation Database 2020' from Kaggle provides a rich dataset that can be harnessed to build such systems. In this report, I aim to create a robust recommendation system using unsupervised clustering algorithms.

Recommendation systems are pivotal in a variety of applications, from streaming services to e-commerce, where they enhance user experience by suggesting relevant items based on past behavior and preferences. By clustering similar items, these systems can provide personalized recommendations without explicit user ratings.

I explore three different clustering models to determine which one offers the best performance in the context of anime recommendations. Our approach involves a comprehensive data preprocessing phase, where I clean and transform the raw data to ensure that the algorithms operate on high-quality inputs. This step is crucial as it directly impacts the performance and accuracy of the recommendation system.

For this I will be using the 'Anime Recommendation Database 202' dataset which includes 35 features (22 numerical and 13 categorical). Some of the numerical features are written as strings so I should be aware of it. The type of data (which should have) of each feature are:

- MAL_ID (Identifier)
- Name (Categorical)
- English name (Categorical)
- Japanese name (Categorical)
- Score (Numerical)
- Genres (Categorical)
- Type (Categorical)
- Episodes (Numerical)
- Aired (years: Numerical)
- Premiered (Categorical)
- Producers (Categorical)
- Licensors (Categorical)
- Studios (Categorical)
- Source (Categorical)
- Duration (Numerical)
- Rating (Categorical)
- Ranked (Numerical)
- Popularity (Numerical)
- Members (Numerical)
- Favorites (Numerical)
- Watching (Numerical)
- Completed (Numerical)
- On-Hold (Boolean)
- Dropped (Boolean)
- Plan to Watch (Bool)
- Scores 1-10 (Numerical)

II. DATA EXPLORATION, CLEANING, FEATURING AND PREPROCESSING

A. Exploration

Before cleaning the data I looked up how many null data is in the dataset and how are they distributed over the dataset. In this case I found no null data directly, nevertheless many features including numerical ones are written as strings and the null data was written as 'Unknown'. When searching for 'Unknown' values I found out the percentages shown in Figure 1, where the two main unknown features are the 'Licensors' and the 'Premiered'.

After that I searched for the data types. In this case I find out that all data was written either on int64 or strings. Many features (such as the scores) can be written as numbers, either int or float instead of a string.

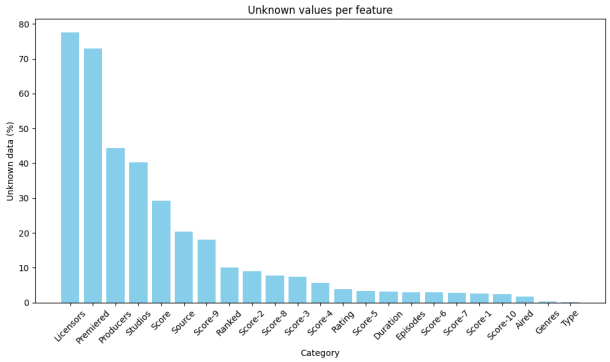

FIG. 1: Percentage of unknown values in the dataset per feature.

Finally I searched for any ordinal categorical features. After showing the unique value of the categorical features I find out that none of them where ordinal features, i.e., I can eventually use OHE or other encoding techniques to describe the categories with numerical values.

B. Cleaning

Since I'm creating a recommendation system, I won't remove any row. I can still remove, transform or add features from the dataset but removing a row means that anime will not be recommended ever.

The first step taken to clean the dataset was to change the 'Score-N' datatype to float and impute the unknown values by linear interpolation (since most of the time I see a monotonically increase or decrease of the scores). Once the interpolation was done, I recalculated the 'Score' feature, which was just a weighted average. For the cases where no information was available (i.e., all the 'Score-N' values were unknown) I set the 'Score-N' values to 0 and a 'Score' of 5. This decision was made since a 'Score of 5' establishes a not good neither bad anime.

Then I removed the 'Premiered' and 'Licensors' features since more than 70% of the data was unknown. For the 'Aired' feature I considered that the most valuable information was in the year (The difference of animation, drawing, style etc... are more distinguishable by year than month or day), so I split the 'Aired' feature into open and closing year of the anime. For the cases where only one year was given I took that year as both open and closing year. For the unknown years I took the average year of anime to avoid setting the year to old or to new.

For the 'Duration', 'Rating' and 'Rank' features I just replaced the 'Unknown' values into the average values of each feature.

C. Feature engineering

After doing all the cleaning steps I created new features from the categorical features applying OHE, the categorical 'Unknown' values can be still processed as a category so this won't be a problem. After the OHE I finished with a dataframe with 15460 features with 15393 belonging to boolean features (from the OHE). To reduce the number of features, I applied PCA ending with a total of a 100 features.

III. CLUSTERING RESULTS

In this section I will show the results of Kmeans++, Hierarchical Clustering Algorithm (HAC) and DBSCAN. Since I'm creating a recommendation system, I would prefer to have at least 10 different clusters to recommend for a user depending on the animes they have seen. For all the models I used took an average Silhouette score from 5 random initial states for each number of clusters to evaluate their performance (Just to recap, -1 score means clusters are not distinguishable at all, 0 means many datapoints are just in the boundary between clusters and 1 means the clusters are distinguishable).

A. Kmeans++

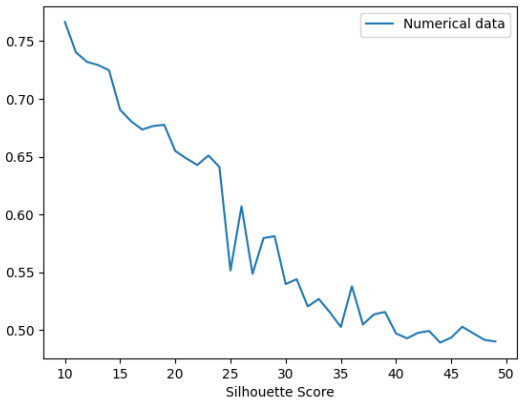

FIG. 2: Kmenas++ average silhouette scores for different number of clusters

B. HAC

C. DBSCAN

Since in DBSCAN I can't assign the number of clusters I did a search for the best parameters. When applying

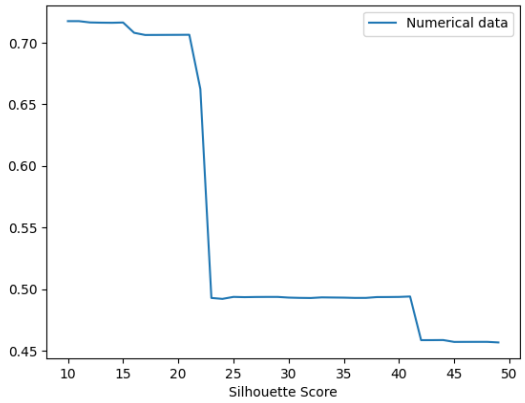

FIG. 3: HAC average silhouette scores for different number of clusters

DBSCAN, increasing the minimum samples parameter keeps or reduces the number of clusters. In the process of searching for the best parameters I stopped increasing the minimum samples parameters when the number of cluster was less than 10. In Figure 4 is shown the heat map of parameters and silhouette scores.

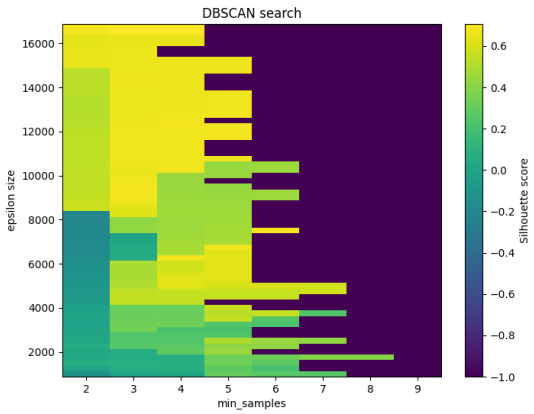

FIG. 4: DBSCAN average silhouette scores for different parameters

IV. FINDING AND INSIGHTS

A. Findings

For the Kmeans++ results we can see that the Silhouette score decreases almost monotonically with a maximum value of 0.77 at 10 clusters. Commonly the Silhouette score is considered good above 0.7, i.e., that for the Kmeans++ algorithm we can use up to 14 clusters.

In the HAC algorithm we see that the Silhouette score

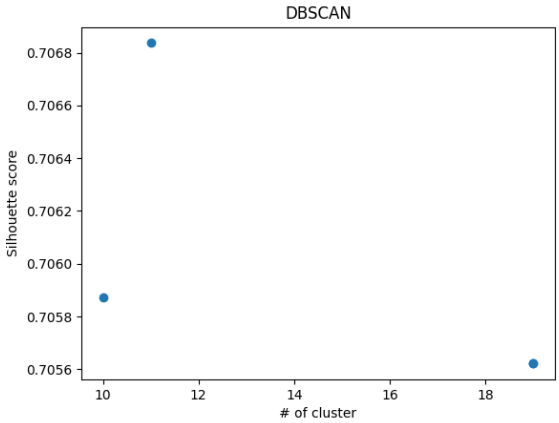

FIG. 5: DBSCAN cases with Silhouette score greater than 0.7

moves as a stair. By using the same logic as before, we can use up to 22 clusters and still consider this a good clustering model.
Finally for the DBSCAN, the heatmap behaves quite different than the other models, the continuity seems to be only on the x-axis. For our dataset we find out that the best parameters are $\epsilon = 16500$ and $\_samples = 4$ with a Silhouette score of 0.707 and 11 clusters. Nevertheless we also find other cases with Silhouette score greater than 0.7, this is shown in Figure 5.

From all models we consider the HAC algorithm the best, we needed at least 10 classes and we have seen it can get up to 22 classes keeping a good cluster separation with Silhouette score greater than 0.7. In Figure 3 is shown the wide variety of number of clusters available to keep a well separated and identifiable clusters. The DBSCAN algorithm is usually the best choice for clustering but in this case we can't take some rows as noise since we need all animes to have a chance of being recommended in the system. The Kmeans++ algorithm can be another good option for the system, but we might be limited up to 15 clusters, nevertheless this model has up to 0.76 of Silhouette score with 10 clusters with is quite better than HAC model.

V. NEXT STEPS

For this project we might include other models such as the Gaussian Mixture Model, this can be a good option since it works with probabilistic values of belonging to each cluster. This might enable the analysis of different animes seen by a user and get the probability of being a good recommendation considering multiple history.