Anime Clustering and Association Rule Mining

Zi Li Pan II
Georgia State University
CSC 4740 Data Mining
Atlanta, GA, USA
zpan3@student.gsu.edu

***Abstract* – In this study, I have clustered user information based on their anime genre preferences and the applied association rule mining within those clusters. This project was conducted on random sample of 50% of the entire dataset and only takes into consideration the user's genre preferences and their anime watch history. The clustering was applied using K-means on each user's overall genre preferences. Then, I applied Apriori to mine frequent itemsets and association rules within each cluster based on each user's watch history. The results showed that the clusters may not be distinct nor meaningful, therefore, the association rules curated from cluster to cluster may not be unique or interesting. In conclusion, it may be in one's interest to cluster based on different distance metrics, using different clustering algorithms, or applying clustering to a different set of attributes.**

I. INTRODUCTION

Providing recommendations for tv shows and movies have become widely popular due to the increase in popularity of online streaming services. This includes online anime streaming services, such as Netflix, Crunchyroll, etc. However, the recommendations provided by these streaming services are not always personalized.

Often times, many services may recommend anime based on a user's watch history. For example, 'a user who've watched anime A have also watched anime B.' This is not a bad approach to recommend anime, however, I believe it would be more personalized if we recommend anime to users based on how similar users are to each other. In other words, a method in attempting to create recommendations for a user based on the interests of another similar user. One method of achieving this is by using collaborative filtering. However, I wanted to find other methods that may successfully achieve this using more common data mining techniques.

In this project, I investigate the usefulness in clustering users by their genre preferences to help identify groups of users who have similar tastes, and then applying association rule mining on each user's anime watch history within these clusters to find anime that are commonly watched among them. The intention is to be able to use these association rules to recommend anime based on what other users have watched in the same cluster. In this report, I will describe the methods I've used, the fallbacks I have faced, what I've done to try to resolve those issues, their results, and my overall conclusion about my method of approaching this problem.

II. MATERIALS AND METHODS

A. Data Explanation and Characterization

For this project, I will use two datasets, Anime.csv and Rating.csv, both gathered from Kaggle.com [1], which are pulled from myanimelist.net.

Anime.csv has 12, 294 rows, one for each anime with a unique id. It has 7 attributes:

- anime_id: unique numerical id identifying each anime,
- name: string name of anime,

- genre: a comma separated list of string genres for the anime,
- type: a string of the type of media (OVA, Movie, TV, etc.),
- episodes: an integer of the number of episodes in the anime (1 if Movie),
- rating: the average rating of the anime, out of 10,
- members: the number of community members that are in this anime's 'group'.

Rating.csv has 7,813,737 rows with 73,516 unique users, and provides each user's ratings for the anime they watched. It has 3 attributes:

- user_id: a unique numerical id identifying each user,
- anime_id: the anime that the user has rated,
- rating: rating the user has given the anime, out of 10 (-1 if watched but unrated).

B. Data Preprocessing

First, I want to analyze the null values within the dataset. In Anime.csv, there is 62 null genres, 25 null types, and 230 null ratings. In Rating.csv, there are no null values. For Anime.csv, I looked into each of these columns with null values. I first looked into the type attribute and found that there are 7 unique types: ['Movie', 'TV', 'OVA', 'Special', 'Music', 'ONA', nan]. I looked into each of these types and noticed that 'Music' is actually music, not anime. Therefore, 'Music' type will need to be removed from analysis. While looking into the 'NaN' (null) types, I noticed that all of them had 'Unknown' episodes and null ratings. This indicates that the anime may not exist or has not been released and will need to be removed from analysis. Once these two types are removed, only actual, existing, anime remain in the dataset. Additionally, after removing music and null types, there are only 205 null ratings left in the dataset. It is important to note that if there is no ratings (ie. Null), then there are no views, making it irrelevant for analysis. So those with null ratings are also removed.

As previously discovered, there is 'Unknown' episodes for anime. Looking into this data, there are 314 anime with 'Unknown' episodes. Note that at this point, null ratings and types have been removed, so these 314 anime with 'Unknown' episodes all have ratings and actually exists. I also noticed that these anime have a large amount of 'members'. For example, *One Piece* has 504,862 members and *Naruto: Shippuuden* has 533,578 members. As a result, I decided to keep these anime with 'Unknown' episodes. Finally, all anime that were removed from Anime.csv has to also be removed from Rating.csv so that the data matches up.

Then, I look into 'members' attribute. Note that I won't use this attribute for analysis, but I think it'll give us a good insight on what we should expect. From Table 1, the anime with the most members is *Death Note*, which may indicate that it'll be in most user's watch history.

Table 1: Top 5 Anime 'Members'

Name	Members
Death Note	1013917
Shingeki no Kyojin	896229
Sword Art Online	893100
Fullmetal Alchemist: Brotherhood	793665
Angel Beats!	717796

Although I won't be using 'name' as attribute for analysis, but rather 'anime_id', I looked into the dataset for any duplicated names. I notice that there are two 'Gintama' with different spelling: *Gintama°* and *Gintama'*. Considering that the two animes' attributes are similar, aside from one with 9.25 rating and 114,262 members and the other with 9.16 rating and 151,266 members, the two may be the same anime but with errors in the spelling. However, since they have two different anime_id, I don't want to exclude this from the dataset as the users might have claimed to watch one but not the other. On the other hand, there also has been anime whose names have been duplicated. For an example, *'Saru Kani Gassen'* is in the dataset twice, with different anime_id. One is a type 'Movie' and the other, 'OVA'. These duplicates are not exact

copies and seem to be just a series of the anime, so they will be kept in the dataset.

Then, I analyzed and cleaned the genre attribute. This is one of the key attributes for my analysis. First, there are 47 anime with null genre. While this isn't a lot compared to the 12,000 anime list, in case they provide significant information, I don't want to drop them from the dataset. Instead, considering that an anime can have one or more genres, I decided to one-hot encode the genre column. There are a total of 44 genres and each genre will have its own column, including 'NaN'. Therefore, those with null genre will have NaN column set to 1, and all other columns set to 0.

Next, I needed to choose the necessary features of my analysis. First, I merge the two datasets on anime_id. So, the resulting dataset is the user_id, each anime_id they watched, and the corresponding attributes of the anime (name, type, episodes, etc.). Since I want to focus on the user's genre preferences for clustering and their anime watch history for association rule mining, I decided to only keep user_id, anime_id, and the one-hot encoded genres. The resulting dataframe has 46 columns and 7,787,221 rows. The result can be seen in Image 1 on the last page.

Since there are almost 8 million data points in my dataset, I used a random sample of 50% of the entire dataset. This reduced the dataset to 3,893,610 rows. Although this is still a lot of data, this allowed my algorithms to run significantly faster.

For clustering, I only used user_id and the one-hot encoded genres. To prepare the data for clustering, I dropped the anime_id column, grouped the users by user_id, and got the sum of their anime genre preferences. This result can be seen in Image 2 on the last page. From this, each row has information on a unique user and their total genre preferences. I then normalized this dataset using z-score normalization.

I've also attempted to apply PCA to reduce the dimensionality of the clustering dataframe. There are 44 genre attributes of the original clustering dataframe. After applying PCA, to keep at least 85% of total variance I used 9 Principal Components (PC). The results can be seen in Image 3 and Image 4 below.

Image 3: PC Variances

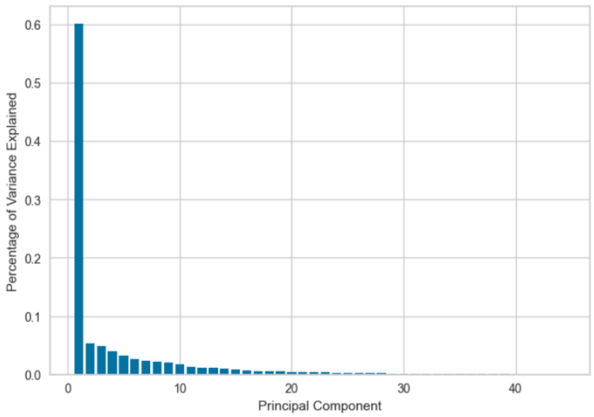
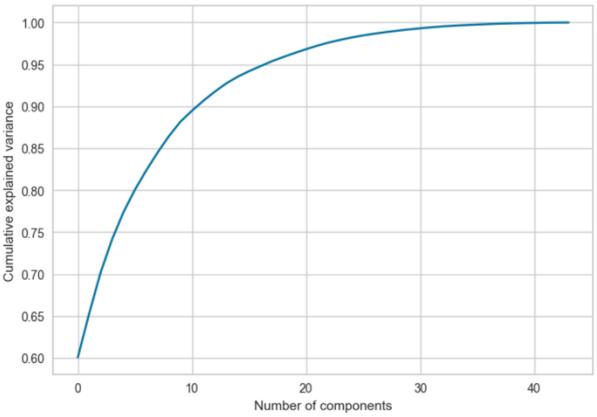


Image 4: PCA Total Variance



This was done as a result of the indistinct clustering, which will be explained later.

For association mining, I dropped the one-hot encoded genre columns and only used user_id, anime_id, and the cluster label of each user, which resulted from the clustering.

C. Data Analysis/Mining

For my study, I wanted to cluster users by their genre preferences and then perform association rule mining within clusters based on anime watch history.

The clustering algorithm I've decided to use for this project is K-means. In this way, it should cluster based on the similarity/distance from each user. Users who are most similar in their genre preferences would be clustered together.

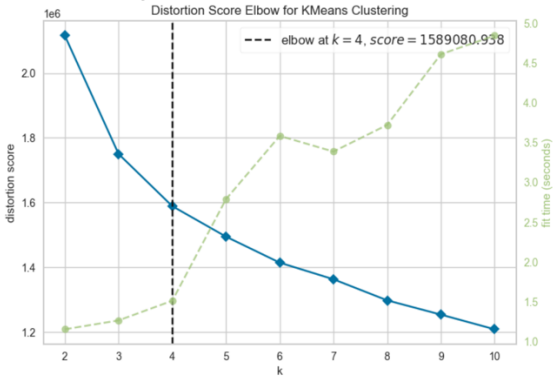
For K-means the hyperparameters include: the number of clusters (n_clusters), initialization

method (init) – the default is set to k-means++, max number of iterations (max_iter) – the default is set to 300, distance metric (metric) – the default is Euclidean distance, convergence tolerance (tol) – the default is 1e-4, seed value (random_state) – default values is none. These hyperparameters can be optimized using cross validation techniques such as grid search.

I decided to keep most of these at their default values, set random_state = 42, and test for the best number of clusters. To find the best k-clusters, I used both the elbow and the silhouette method.

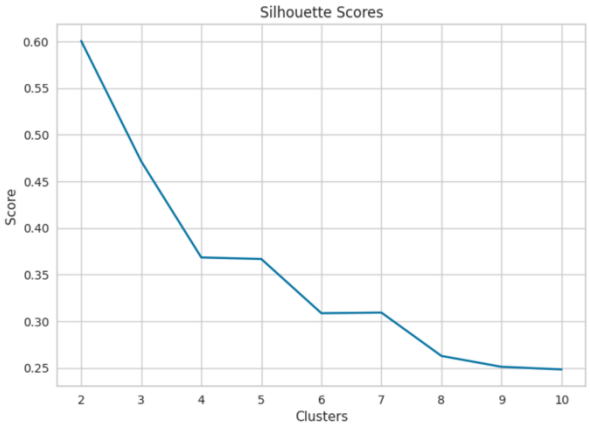
Ideally, for the elbow method, we want low distortion scores while keeping the number of clusters as small as possible. The results of the elbow method can be seen below in Image 5.

Image 5: Elbow Method



Ideally, for the silhouette method, we want high silhouette scores while keeping the number of clusters as low as possible. The results of the silhouette method can be seen below in Image 6.

Image 6: Silhouette Scores



After carefully analyzing these results, I decide that 4 clusters may be a good balance between the two methods.

I applied K-means clustering to the dataset using 4 clusters and added the resulting labels of the clustering to the dataset. Then for each cluster label, I had to encode the dataset as a transaction list for association rule mining. I then applied Apriori to the dataset, using min_support = 0.1. I tested this value at different thresholds. Naturally, I found that if I increased the value of the min_support past 10%, I got very little to no frequent itemsets returned. For example at 50%, no frequent itemsets were returned and therefore no association rules can be generated. At 20%, few frequent itemsets were returned with no association rules. And testing at 5% gave me more frequent itemsets, but still, for cluster 0, there were no association rules generated. Therefore, I felt that 10% was a good threshold for this value, although no association rules were generated for clutser 0.

Using these frequent itemsets, I generated association rules. The default metric used for association rules is confidence with the minimum threshold set of 0.5. To find meaningful, nontrivial rules, I set metric = 'lift' and min_threshold = 1.

D. Evaluation and Interpretations

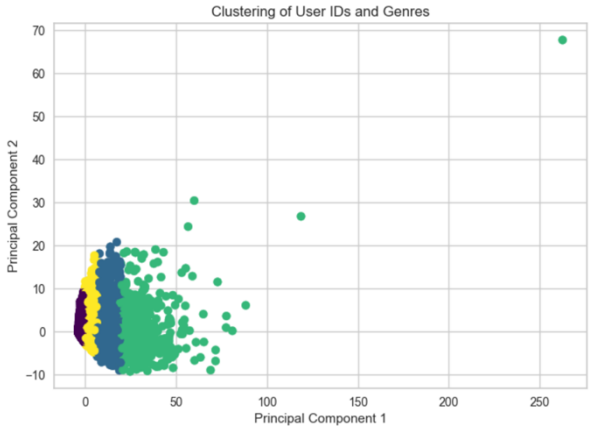
For clustering, I used both elbow method and silhouette method. As mentioned in the previous section, ideally we want low distortion scores, high silhouette scores, both while maintaining low clusters. However, the 'elbow' of the elbow method is not well distinguished and the silhouette score is still fairly low. This may result in a very bad clustering as the clusters may be too similar to each other.

To evaluate the quality the clustering, I used cross validation at 5 folds. The resulting cross validation scores: [-253960, -227897, -253815, -242939, -266776], with a mean score of: [-249077]. These scores indicate that the k-means clustering algorithm may not be performing well on my dataset, as the values of negative inertia are quite high. This suggests that the clusters may not be well separated or compact.

III. RESULTS

In order to visualize the result of the clustering, I applied PCA to reduce the dimensions of the dataset to 2 dimensions. The result can be seen in Image 7 below.

Image 7: Clustering



And the results of Apriori and Association rule mining can be seen in Image 8 below.

Image 8: Apriori and Association Rule Mining

```
Frequent itemsets for cluster 0:
    support itemsets
0  0.132229     (20)
1  0.116736    (121)
2  0.110306    (199)
3  0.126100    (226)
4  0.218173   (1535)
5  0.117534   (1575)
6  0.107285   (5114)
7  0.119670   (6547)
8  0.150462  (11757)
9  0.144226  (16498)
Association rules for Cluster 0
Empty DataFrame
Columns: [antecedents, consequents, support, confidence, lift]
Index: []
Frequent itemsets for cluster 1:
       support        itemsets
0     0.251863             (1)
1     0.139366             (5)
2     0.207835             (6)
3     0.107298            (19)
4     0.306292            (20)
...        ...             ...
1529  0.103484  (28121, 19815)
1530  0.110591  (30276, 19815)
1531  0.103657  (20507, 22319)
1532  0.104871  (27899, 22319)
1533  0.100711  (30276, 22319)

[1534 rows x 2 columns]
Association rules for Cluster 1
     antecedents consequents   support  confidence      lift
0            (1)      (1535)  0.111978    0.444597  1.094233
1         (1535)         (1)  0.111978    0.275597  1.094233
2            (1)      (1575)  0.105044    0.417068  1.042942
3         (1575)         (1)  0.105044    0.262679  1.042942
4         (2904)         (1)  0.103137    0.273689  1.086657
...          ...         ...       ...         ...       ...
1771     (22319)     (20507)  0.103657    0.374922  1.389160
1772     (27899)     (22319)  0.104871    0.516652  1.868695
1773     (22319)     (27899)  0.104871    0.379310  1.868695
1774     (30276)     (22319)  0.100711    0.374597  1.354890
1775     (22319)     (30276)  0.100711    0.364263  1.354890
```

IV. DISCUSSION AND CONCLUSION

In the beginning I had believed that I would get good, well distinguished clusters using k-means, however while evaluating the results of the elbow and silhouette method, I noticed that it may not be as I expected. The results of both were not very well distinguished and may indicate that the clustering may not perform very well. I used 4 clusters, but, in consideration to the number of data points and users, I may have underfitted. Each cluster didn't have a good representation of itself, as it was no different from other clusters.

As discussed previously, I've attempted to use PCA to reduce the dimensions of my clustering dataset. The original 44 attributes were reduced to 9 Principal Components. I had hoped that this would improve the performance of my clustering, however, the elbow and silhouette results showed that they were going to be approximately the same.

Image 9: PCA Elbow Method

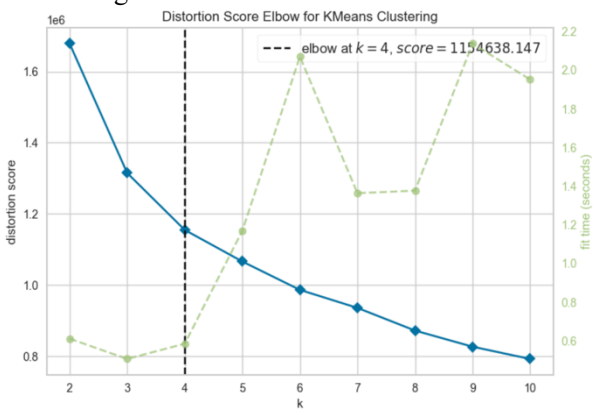
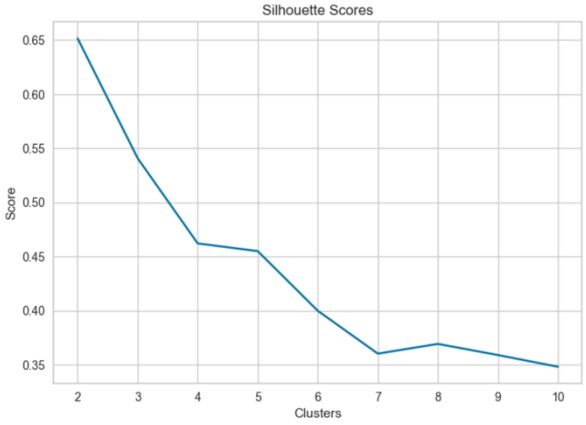
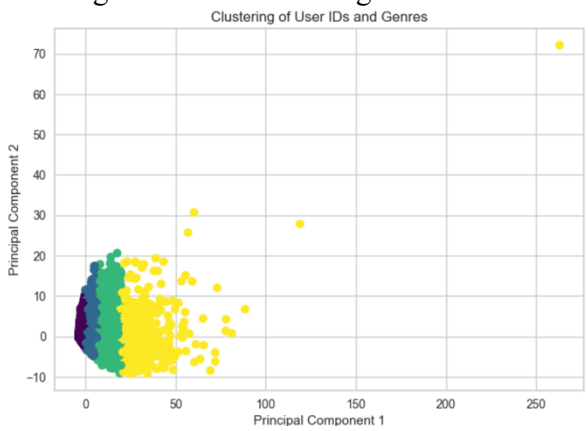


Image 10: PCA Silhouette Scores



And, as expected, the resulting cluster was similar to before, as shown in Image 11 below.

Image 11: PCA Clustering



It may be that the attributes chosen for clustering (user's genre preferences) were simply too similar to each other to begin with, especially since a single anime may have many genres. This can result in indistinct clustering. It may be that the chosen distance metric, Euclidean, was not a good option for the project. In the future, I would look into applying clustering with cosine similarity and other metrics to compare. I would even perhaps look into different clustering methods such as Fuzzy K, as users may belong in more than one cluster.

The results of frequent itemset and association rule mining is highly dependent on the clustering. Because the clustering was not clear and well separated, Apriori may generate sparse, inconsistent, and overlapping itemsets. This would explain why decreasing the minimum support threshold for Apriori still couldn't generate many frequent itemsets.

Since these frequent itemsets are used to generate association rules for anime recommendations, increasing min_support would give me sparse recommendations, and decreasing could potentially give me irrelevant recommendations. However, because of the bad clustering, I was unable to choose the optimal min_support threshold. I believe that the chosen threshold, 10%, would not suffice for a recommendation system as it could recommend items no different from user to user. Meaning that the resulting recommendation system is not personalized, not living up to my intentions.

REFERENCES

[1] CooperUnion. (2017, November). Anime Recommendations Database, Version 1. Retrieved March 20, 2023 from https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database

Image 1: Merged Dataframe

```
# merge these two datasets
merged_df = pd.merge(users, anime_genre, on='anime_id')

# result: user_id, anime_id, and genre columns - 7,787,221 rows
display(merged_df.head())
```

	user_id	anime_id	ACTION	ADVENTURE	CARS	COMEDY	DEMENTIA	DEMONS	DRAMA	ECCHI	...
0	1	20	1	0	0	1	0	0	0	0	...
1	3	20	1	0	0	1	0	0	0	0	...
2	5	20	1	0	0	1	0	0	0	0	...
3	6	20	1	0	0	1	0	0	0	0	...
4	10	20	1	0	0	1	0	0	0	0	...

5 rows × 46 columns

Image 2: User Genre Preferences

```
# get the sum of the genre preferences of each user
clustering_df = clustering_df.groupby('user_id').sum()

# result: the data is now condensed
# instead of a row per anime the user watched, we return it as one row
# each column is the sum of anime they watched with that genre
# this is so that every row has info for a unique user and their preferences
# 73,513 rows
display(clustering_df.head())
```

	ACTION	ADVENTURE	CARS	COMEDY	DEMENTIA	DEMONS	DRAMA	ECCHI	FANTASY	GAME	...
user_id											
1	75	15	0	97	0	14	29	65	46	6	...
2	0	0	0	2	0	0	0	0	0	0	...
3	64	48	0	39	0	4	31	5	52	8	...
4	31	13	0	27	0	2	5	6	25	2	...
5	157	93	4	326	1	20	78	100	96	10	...