ANIME RECOMMENDATION SYSTEM

[1]Aditya Gupta (2022031)
Dept. of Computer Science and Engineering
IIIT - Delhi
India
Email: aditya22031@iiitd.ac.in

[2]Sahil Gupta (2022430)
Dept. of Computer Science and Artificial Intelligence
IIIT - Delhi
India
Email: sahil22430@iiitd.ac.in

***Abstract*—This paper outlines the overview of the methods and techniques used for anime recommendation systems . We will implement 4 types of recommendor: 1.Popularity based recommendation , 2.Content-based recommendation, 3.Collaborative filtering , 4.Memory Based Approaches , 5.Model-based approach , 6.Deep Learnig Mf - Generalized Matrix Factorization (Mf), 7.Clustering and Collaborative Recommender**

1. Problem statement

Recommendation systems are straightforward algorithm designed to offer the users the most precise and dependable items by sifting through a vast pool of information . These system operate by analyzing data patterns within the dataset , learning from user choices and preferences, and subsequently predicting outcomes that closely align with the user's interest . We use datasets in this project to do the same . Recommendation systems are used in various applications, from web searches to e-commerce websites. In our case, we are focusing on a recommendation system for anime searching services. The Anime Recommendation system allows users to explore their favorite animes across different genres and with varying episode lengths, ranging from short movies to over a thousand episodes. This reduces the time needed to find the desired anime, making it essential for the recommendation system to be highly reliable in providing suggestions that align with users' interests.

2. Motivation

In the recent years , anime has surged in popularity , captivating the interests of vast audience , particularly teenagers .World of anime offers a diverse range of genres, captivating storylines, and visually stunning animations .As anime libraries continue to grow, users face the challenge of navigating through an extensive pool of options to discover content that resonates with their specific interests.Thus the need of an anime recommendation system is of great importance which derives our motivation for this project .

3. Dataset Details

The dataset is being taken from kaggle.com which contains Recommendation data from 76,000 users at myanimelist.net . The data set contains information on user preference data from 73,516 users on 12,294 anime . Each user is able to add anime to their completed list and give it a rating and this data set is a compilation of those ratings. The scores/ratings vary from 1 - 10 with 10 being the most effective. A rating of -1 indicates that the user did not rate the item. The dataset contains 2 comma-seperated files :

1) `anime.csv` - contains information about anime_id, name, genre, type, episodes, rating, members.
2) `rating.csv` - contains information about user_id , anime_id , rating .

4. Methodology

4.1. Popularity based recommendation

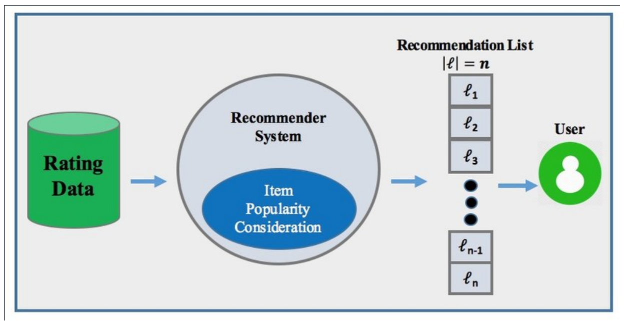


Figure: Popularity Based Approach

4.1.1. Introduction

For any machine learning problems, we need a baseline model or method to use as a reference whether our approach is good or not. Our machine learning prediction or sophisticated analysis should, at least, beat those baseline performance. For recommendation systems, a simple baseline could be recommending popular items to users. This baseline assumes that if an item is popular among many users, it's likely to be a good recommendation for new users as well. Thus, this is the simplest model for any recommendation system.

4.1.2. To define the popularity of the item

Regarding the IMDB system, we use a metric called the weighted rating system which is used to score the rating of each movie.

The formula for the same is:

$$\mathrm{Wr} = \left(\frac{v}{v+m}\right) \cdot R + \left(\frac{m}{v+m}\right) \cdot C$$

where:

- R = average rating of the show/movie
- v = total number of people's votes for the movie
- m = minimum votes required to be listed in the top 250 (This number may vary as it's defined by >80 percentile of total votes)
- C = the average rating across the whole dataset

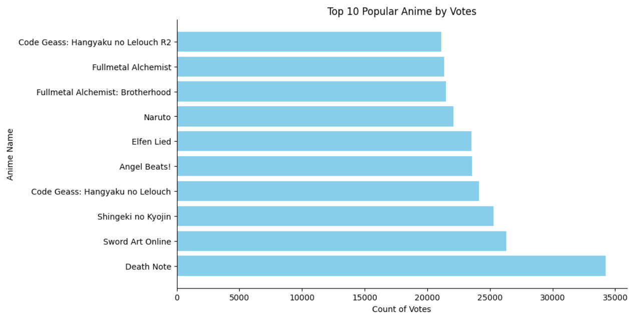


Figure: Top 10 Popular Anime by Votes

4.1.3. Disadvantages (Drawbacks)

This method is not personalized for each user; each user gets the exact same result.

4.1.4. Use case

For new users, if we don't know their liked shows and their favorite genres, we can provide them a list based on ranking based on weighted rating as this is the simplest model.

This is how the streaming platforms work when we search by "popular" on them.

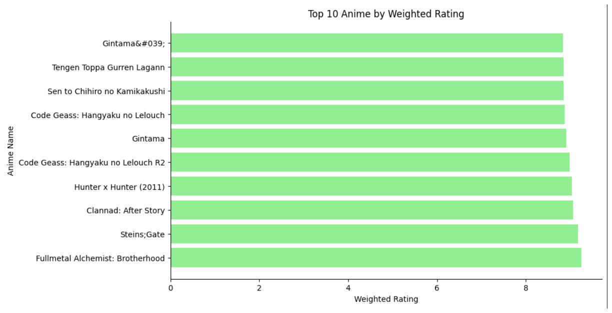


Figure: Top 10 Anime by Weighted Rating

4.2. Content-based recommendation

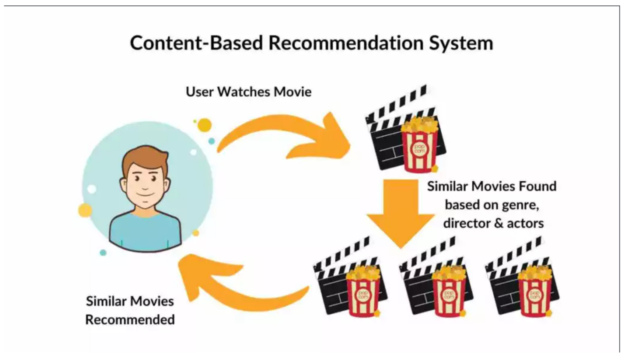

Content-Based Recommendation System
User Watches Movie
Similar Movies Found based on genre, director & actors
Similar Movies Recommended

Figure: Content Based Approach

4.2.1. Introduction

This recommendation system acts in a similar way to recommending animes of the same genre that we like and have watched before. For example, if someone likes "Bleach," they will be recommended animes like "Naruto" and "Jujutsu Kaisen." Here, we create a better way of recommendation by introducing other features of the content into our algorithm engine. It's a further improved system compared to the popularity-based recommendation system mentioned before.

4.2.2. Disadvantages (Drawbacks)

The main drawback of this algorithm is that it is limited to recommending animes of the same genre. Nowadays, viewers want variety as well. It will not recommend animes that the viewer has not seen or liked in the past.

4.2.3. Use case

This method is beneficial if someone wants to watch animes that are of the same type they have recently watched. It helps them to re-experience those feelings they got from watching the previous anime.

4.3. Collaborative filtering

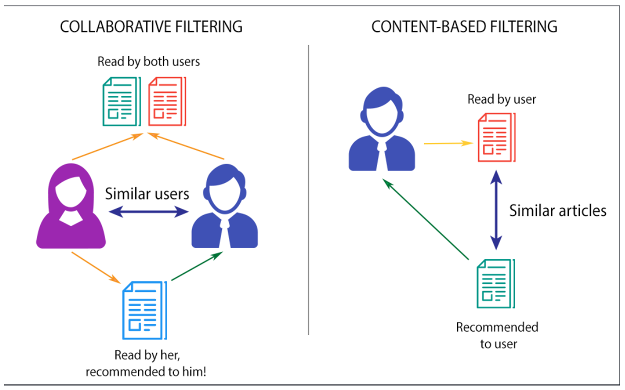


Figure: Collaborative Filtering vs Content-Based Filtering

4.3.1. Introduction

The collaborative filtering algorithm uses ”User behavior” for recommending systems. This is widely used in the industry nowadays as it is not dependent on additional information. There are different types of collaborative filtering techniques. Some of the memory-based collaborative filtering methods are as follows:

label=)

1) User-based: The user-similarity matrix consists of a distance metric that measures the similarity between any two pairs of users. This is useful when the number of users is small. It is not very effective when there are many users, as it takes a lot of time to compute the similarity between all users. This leads to product-product collaborative filtering, which is effective when the number of users is more than the number of products. In this case, the number of users should be more than the number of recommended animes.
2) Product-based: Similarly, the item-similarity matrix assesses the resemblance between any two sets of items.

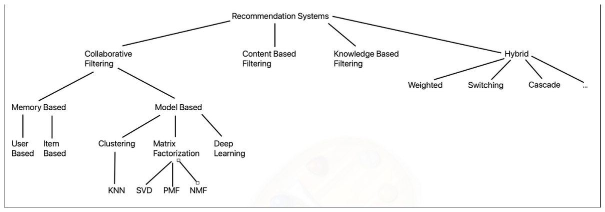


Figure: Division of approaches for recommended systems

4.3.2. Disadvantages (Drawbacks)

Some of the disadvantages are also mentioned in the user-based collaborative filtering section.

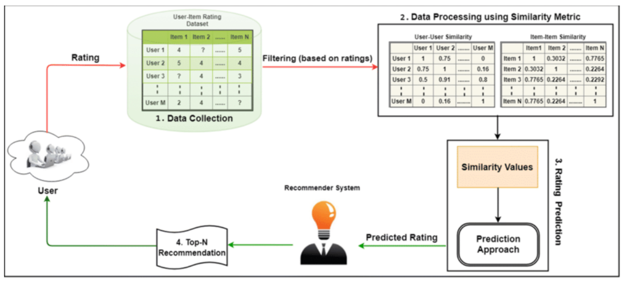
When a new user or item is introduced to the dataset, it triggers what's known as a Cold Start situation. This can manifest in two ways:

1) New User (Visitor): Since this user lacks any historical data, our system lacks insights into their preferences. One approach to address this is by leveraging recent overall or regional trends to gauge popular choices.
2) New Item (Product): The more interactions a product garners, the more effective our model becomes in recommending it accurately to the appropriate users. Content-based filtering is a technique that can be employed to mitigate this challenge.

4.3.3. Use case

Collaborative filtering is a technique used in recommendation systems to make predictions about an individual's preferences based on the preferences of similar users.

4.4. Memory Based Approaches - Subsection - 1 of Collaborative filtering



	Item 1	Item 2	......	Item N
User 1	4	?	......	5
User 2	5	4	......	4
User 3	?	4	......	3
⋮	⋮	⋮	⋮	⋮
User M	2	4	......	?

	User 1	User 2	....	User M
User 1	1	0.75	....	0
User 2	0.75	1	....	0.16
User 3	0.5	0.91	....	0.8
⋮	⋮	⋮	⋮	⋮
User M	0	0.16	....	1

	Item1	Item 2	....	Item N
Item 1	1	0.3032	....	0.7765
Item 2	0.3032	1	....	0.2264
Item 3	0.7765	0.2264	....	0.2292
⋮	⋮	⋮	⋮	⋮
Item N	0.7765	0.2264	....	1

Figure: Memory based approach

4.4.1. Introduction

The fundamental distinction between memory-based approaches and model-based techniques lies in their parameter learning process. Memory-based methods do not involve learning parameters through algorithms like gradient descent; instead, they rely solely on arithmetic operations such as Cosine similarity or Pearson correlation coefficients to compute the proximity between users or items. These methods leverage historical user rating data to gauge similarity between users or items, aiming to recommend unseen items by identifying the most similar ones. This process forms the basis of memory-based collaborative filtering recommender systems.

4.4.2. How to proceed (implementation)

The size of the user-product matrix is really huge. It's computably impossible to compute the user_features with unique_users X unique_animes shape or the anime features for the same reason.

Thus, we drop some of the rows for demonstrating this method.

Also, we can create a matrix of lower dimension using ”TruncatedSVD” or ”PCA”.

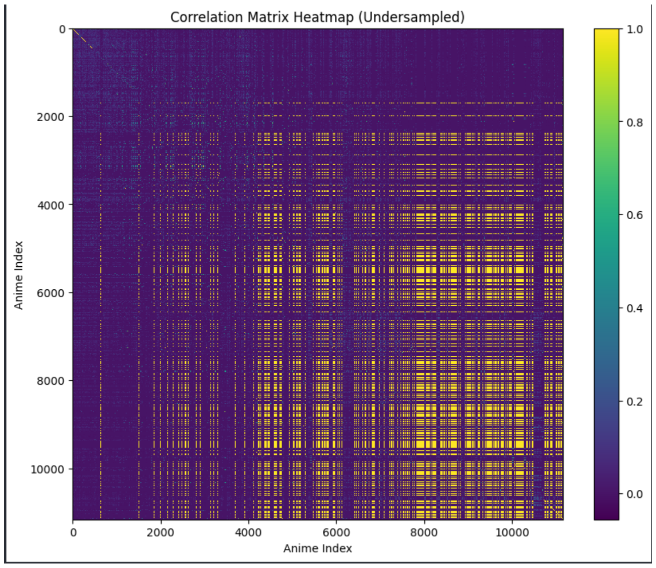


Figure: Correlation Matrix Heatmap

4.4.3. User-based

We find the similar users based on different strategies like Pearson correlation, cosine similarity, or KNN nearest neighbor. We average the rating of each item based on the group of similar users. Rank the items based on the average rating descendingly and recommend the target user with the anime that they never rated before ranking from the highest top k average rating.

4.4.4. Product-based

We identify clusters of comparable products using Pearson correlation, cosine similarity, or KNN (k-nearest neighbors) algorithms. Then, we select the top k most similar items for recommendations. It's important to clarify that item similarity in this context is not based on content (unlike content-based recommendations) but rather on the explicit ratings derived from user behavior as captured in the user-product matrix.

4.4.5. Disadvantages (Drawbacks)

1) It's not scalable due to the sparsity of the data.

2) Constructing the similarity matrix for every new user introduction is challenging to maintain and operationalize.

4.4.6. Use case

The list result can be changed into a front-end website which we are not doing in the project as of now due to lack of time (it can be a good summer project).

It can be showcased like "because you like Death Note, here are some suggestions: ".

4.5. Model-based approach - Subsection - 2 of Collaborative filtering

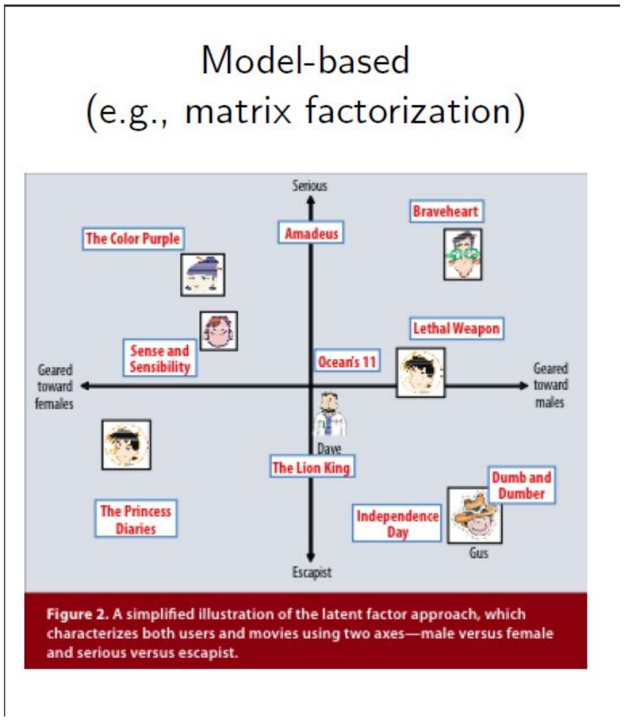


Figure: Model Based approach

4.5.1. Introduction

Model-based collaborative filtering employs machine learning techniques to forecast users' ratings for items they haven't rated. Various algorithms fall under model-based CF, with matrix factorization models like SVD being particularly prevalent. Other approaches include applying latent Dirichlet allocation or utilizing Markov decision processes.

4.5.2. How to proceed (implementation)

The size of the user-product matrix is really huge. It's computably impossible to compute the user_features with unique_users X unique_animes shape or the anime features for the same reason.

Thus, we drop some of the rows for demonstrating this method.

Also, we can create a matrix of lower dimension using "TruncatedSVD" or "PCA".

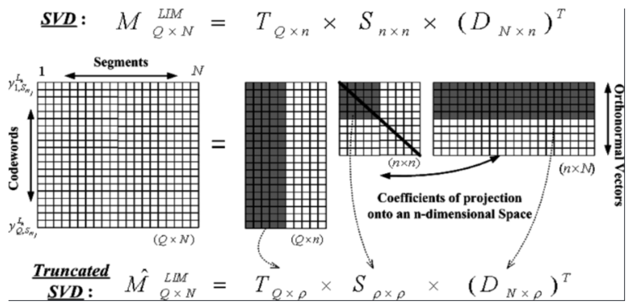

SVD: $M^{LIM}_{Q\times N} = T_{Q\times n} \times S_{n\times n} \times (D_{N\times n})^T$
Segments
1
N
Codewords
=
$(Q\times N)$
$(Q\times n)$
$(n\times n)$
$(n\times N)$
Coefficients of projection onto an n-dimensional Space
Orthonormal Vectors
Truncated SVD: $\hat{M}^{LIM}_{Q\times N} = T_{Q\times\rho} \times S_{\rho\times\rho} \times (D_{N\times\rho})^T$

Figure: Truncated SVD and PCA

4.5.3. User-based

We extract user_features and anime_features using matrix factorization techniques to capture the latent factors within the user-item interaction matrix. This helps uncover hidden patterns in user preferences.

Next, we identify clusters of similar users based on Pearson correlation, cosine similarity, or KNN (k-nearest neighbors). These clusters can vary in size and are determined based on similarity metrics.

We then calculate the average rating for each item within the group of similar users. Items are ranked in descending order based on their average ratings. For a target user, we recommend movies they haven't rated before, starting with the highest-ranked items in the top-k list of average ratings.

4.5.4. Product-based

We leverage matrix factorization techniques to derive user_features and anime_features, aiming to capture the latent structure within the user-item interaction matrix.

Subsequently, we focus on anime_features as an example. Using methods like Pearson correlation, cosine similarity, or KNN (k-nearest neighbors), we compute similarities between each item. This analysis helps identify the top_k most similar items for recommendation.

It's important to note that this similarity isn't based on content (as in content-based recommendation) or explicit user behavior (as in memory-based CF). Instead, it's rooted in the latent factors uncovered through matrix decomposition, representing underlying factors that aren't explicitly interpretable.

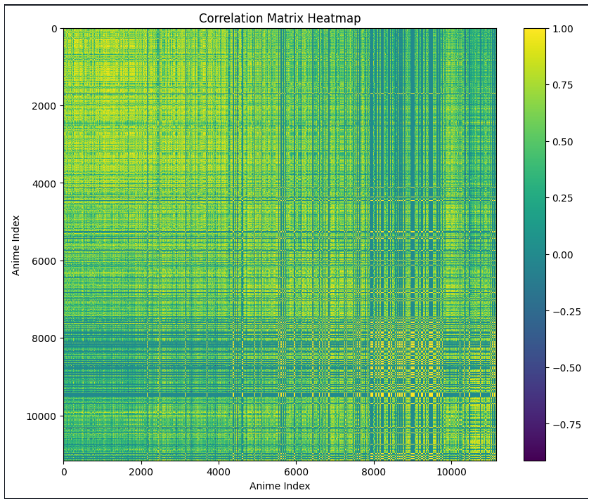


Figure: Correlation Matrix Heatmap

4.5.5. Disadvantages (Drawbacks)

1) It's not scalable due to the sparsity of the data.

2) Constructing the similarity matrix for every new user introduction is challenging to maintain and operationalize.

4.5.6. Use case

The list result can be changed into a front-end website which we are not doing in the project as of now due to lack of time (it can be a good summer project).

It can be showcased like "because you like Death Note, here are some suggestions: ".

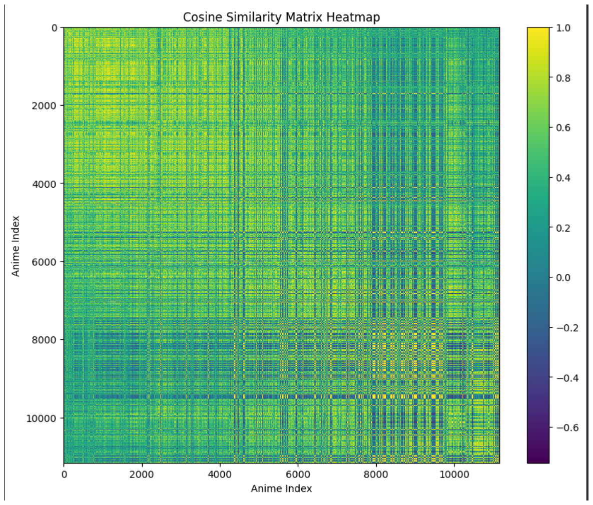


Figure: Cosine Similarity Matrix Heatmap

This method demonstrates how we can calculate similarity using a specified number of latent factors. Recommendations are then made based on these latent factors, which aren't directly interpretable but are mathematically the ones that minimize the loss between actual ratings and reconstructed ratings. Although this approach reduces the size of the matrix compared to memory-based methods, it still has limitations. As the number of users or items grows over time, computational resources become strained, potentially reaching their limits. This scalability issue highlights the need for more efficient techniques in large-scale recommender systems. The difference in the correlation matrix clearly shows that how the models are progressing towards better results .

4.6. DEEP LEARNING MF - GENERALIZED MATRIX FACTORIZATION (MF)

4.6.1. Introduction

In the context of the paper, a generalized matrix factorization can be described by the equation:

$$\hat{y}_{ui} = a\left(h^T(p_u \cdot q_i)\right)$$

where a is an activation function and h is the edge weight matrix of the output layer. The edge weight matrix can be seen as an additional weight to the layer.

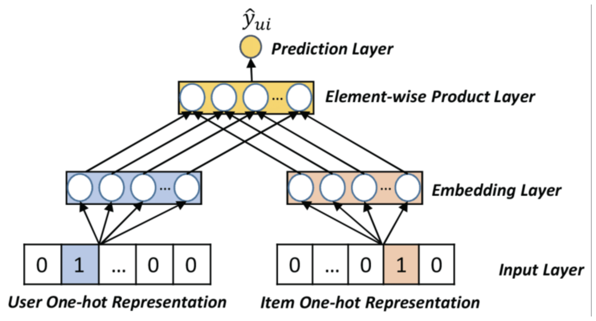

$\hat{y}_{ui}$
Prediction Layer
Element-wise Product Layer
Embedding Layer
Input Layer
0 1 ... 0 0
0 ... 0 1 0
User One-hot Representation
Item One-hot Representation

Figure: Multiclass autoencoder

In the deep learning approach, we leverage a distinct internal engine known as Deep Neural Networks (DNNs) to estimate the latent features of users and items. We estimate the shared latent features by minimizing the target loss function, which in this case is binary cross-entropy(BCE).

After that, we used the fitted model to predict the pair of anime that each customer have never watched to see what to recommend next.

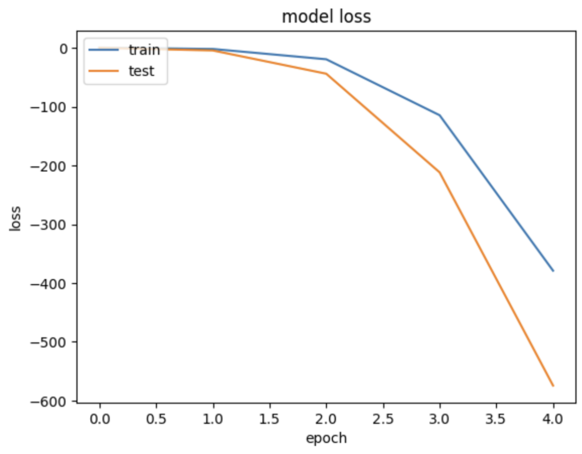


Figure: Model Loss

4.6.2. Use Case

In general, DL models achieve higher accuracy. First, DL can leverage additional data. Many traditional machine-learning techniques plateau with more data. However, when you increase the capacity of neural networks, the model can improve performance with more data.

4.6.3. Disadvantages

Takes more time and power to be computed .

4.6.4. Loss reduces as epochs increase

We see how the loss reduces as the number of epochs increases in the neural network .

Prediction - GMF Now we predict based on this neural network .

Item-based recommendation In item-based recommendation systems, we can derive the latent factors matrix from the Keras model. After that we can find the similarity between each item with respect to the latent factors .

4.7. Clustering and Collaborative Recommender

4.7.1. Introduction

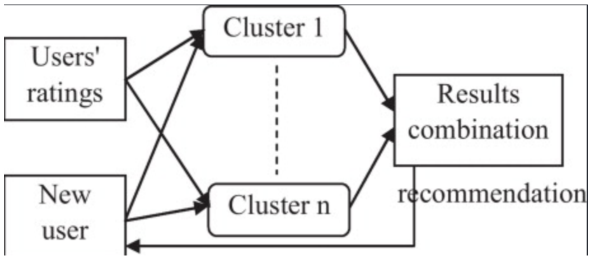


Figure: clustering

Clustering is an unsupervised learning technique that divides a set of data into multiple groups (also known as clusters) based on their similarities. The aim is to have similar items grouped together and dissimilar items placed in different clusters . In recommendation systems, clustering algorithms can be used to group similar users together based on their preferences and behaviors. This information can then be used to make personalized recommendations to each user based on the preferences of the cluster they belong to.

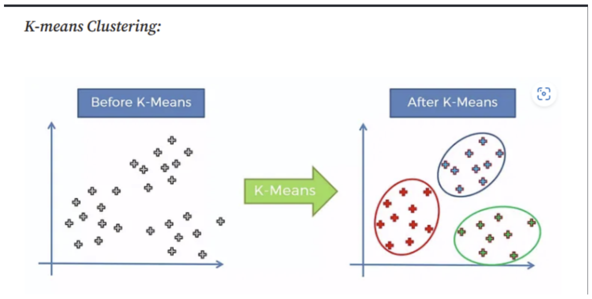


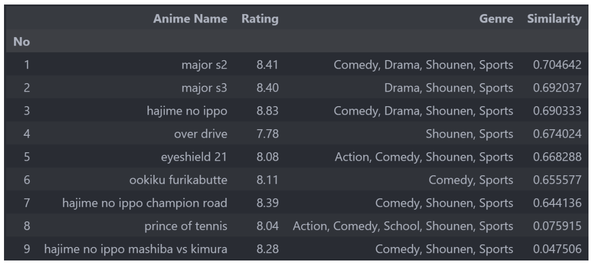
Figure: K means clustering

4.7.2. Use Case

Personalization: Clustering algorithms allow for personalization by grouping similar users together and making recommendations based on the preferences of the cluster.

Scalability: Clustering algorithms are scalable and can handle large amounts of data, making them ideal for use in recommendation systems that have a large user base.

Robustness: Clustering algorithms are robust to noisy data and can handle missing values, making them suitable for use in real-world recommendation systems.



No	Anime Name	Rating	Genre	Similarity
1	major s2	8.41	Comedy, Drama, Shounen, Sports	0.704642
2	major s3	8.40	Drama, Shounen, Sports	0.692037
3	hajime no ippo	8.83	Comedy, Drama, Shounen, Sports	0.690333
4	over drive	7.78	Shounen, Sports	0.674024
5	eyeshield 21	8.08	Action, Comedy, Shounen, Sports	0.668288
6	ookiku furikabutte	8.11	Comedy, Sports	0.655577
7	hajime no ippo champion road	8.39	Comedy, Shounen, Sports	0.644136
8	prince of tennis	8.04	Action, Comedy, School, Shounen, Sports	0.075915
9	hajime no ippo mashiba vs kimura	8.28	Comedy, Shounen, Sports	0.047506

Figure: Similarity between recommended anime and selected anime

4.7.3. Disadvantages

Determining the optimal number of clusters can be subjective and challenging.

The choice of clustering algorithm can significantly impact the results.

Clustering can be sensitive to outliers and noise in the data.

5. Visualizations

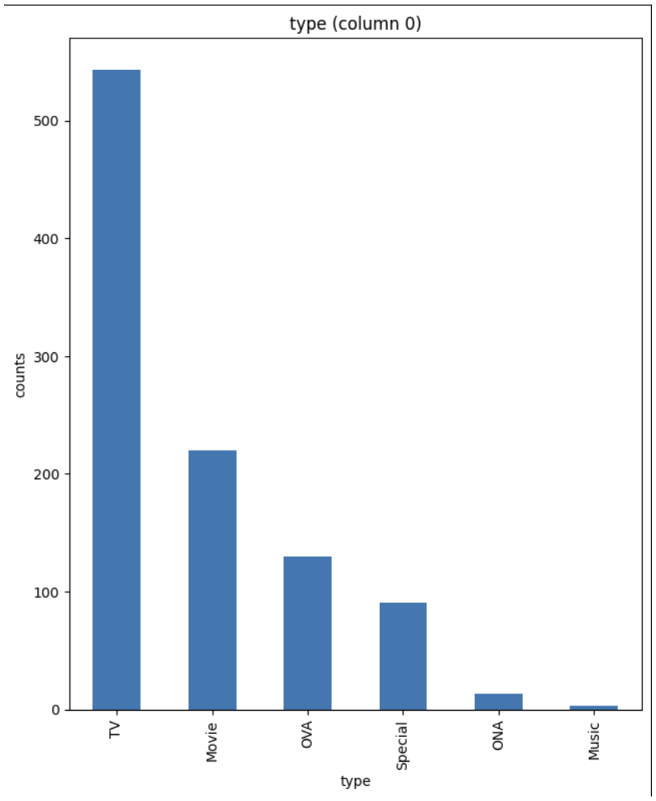


Figure: plot of per column distribution

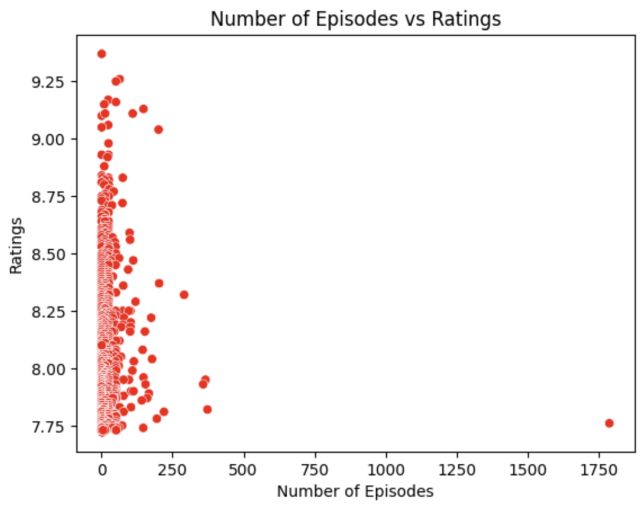


Figure: Number of Episodes vs Ratings

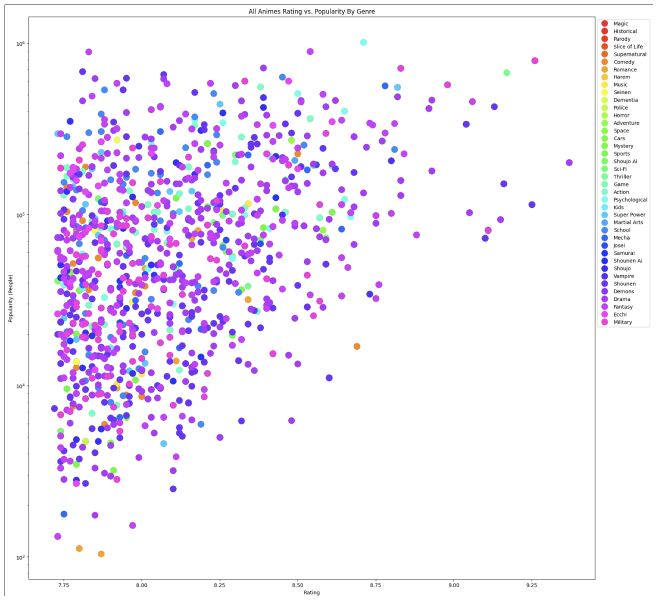


Figure: All Anime Rating vs Popularity By Genre

References

[1] https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database/data

[2] https://www.kaggle.com/code/yamanizm/recommendation-systems-svd-hybrid-k-nn-kmeans

[3] https://www.kaggle.com/code/andidewa21/starter-anime-recommendations-database-82179a4e-e

[4] https://thingsolver.com/blog/introduction-to-recommender-systems/

[5] https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV?ref_=helpms_helpart_inline

[6] https://towardsdatascience.com/content-based-recommender-systems-28a1dbd858f5

[7] https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/

[8] https://www.ethanrosenthal.com/2015/11/02/intro-to-collaborative-filtering/

[9] https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/

[10] https://towardsdatascience.com/how-does-collaborative-filtering-work-da56ea94e331

[11] https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

[12] https://towardsdatascience.com/how-does-collaborative-filtering-work-da56ea94e331

[13] https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_systems)

[14] https://keras.io/examples/structured_data/collaborative_filtering_movielens/

[15] https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401

[16] https://medium.com/@Karthickk_Rajah/clustering-based-algorithms-in-recommendation-system-205fcb15bc9b