ANIME RATING PREDICTION USING COLLABORATIVE FILTERING TECHNIQUE

Sirapatsorn Pongpiriyakarn, Jiravit Moontep, Surachan Liaowongphuthorn
Department of Mathematics
Harvey Mudd College
Claremotn, CA 91711
apongpiriyakarn@hmc.edu, {jmoontep, sliaowon}@students.pitzer.edu

June 6, 2019

ABSTRACT

The goal of this project is to predict Anime ratings and create a recommendation system for users. In our midterm project, we learned multiple collaborative filtering techniques from the Netflix Prize competition. In this project, we applied the techniques like Matrix Factorization or Neighborhood Models on the Anime dataset on Kaggle. We visualized and cleaned data before implementing 9 different models in Python with Surprise and R with RecommenderLab. The results suggested that the SVD model gave the lowest RMSE, thus was the most accurate to predict the Anime rating. We explored the cross validation technique and included an example of our recommender for a user.

1 Introduction

The recommendation system is one of the most popular application in the field of data analytics. Many world-leading tech companies developed a system to help their customers gain overall better experiences. Customers can buy products that they would likely be satisfied, watch movies that they would probably give high ratings, or see online posts about what they would tentatively like. Additionally, having this kind of system can help consumers decide. In particular, without the recommendation system they will likely spend more time looking through an enormous database and choosing what they guess would fit them. With the recommendation system they can efficiently decide their likely-preferred items, thus save both time and money.

This project was inspired by our earlier project in which we investigated the papers of the winning team in the Netflix Prize competition. We learned the specific techniques they used in their models and got more comfortable with a recommendation system. In this paper, we used multiple basic collaborative filtering techniques to build a recommendation system that predicts customer ratings of Anime movies and TV shows and gives suggestions of what movies and TV shows they choose watch. We compared different models and picked the one with the most accurate prediction.

1.1 Overview

Before we built our recommender, we filtered data from the database so that we could look at an effective set of data and it was not too large to reasonably train in the short period of time. We visualized data to initially get more comfortable with the dataset and look into the characteristics of it. We then used the public machine learning libraries to train the filtered data with 9 different models, 7 of which in Python and the rest in R. All models have an associated error value that we used to compared for the better model. The model was then used for prediction by given a user, we listed the movies with the highest associated ratings.

Note that we faced a big time constraint in this project. Specifically, after we decided on what data set we would use we had three days to write codes to filter useful parts of database, divide and train multiple data sets, and predict data, try to understand various models, make slides and give final presentations. As a result, our recommendation system might be prone to errors.

1.2 Recommendation System

A recommendation system is a system that predicts user responses. It is typically trained on a large source of historical information. For example, Amazon gives its customers product suggestions after they make a purchase. Facebook shows users' favorite posts or pages on their timeline. Google displays advertisements when users browse through the website. YouTube has related videos right next to the video users are watching. Netflix recommends the next movies to watch. In general, the system learns customer choices and provides the outcomes of their interest.

1.3 Netflix Prize Competition

Netflix has its recommendation system Cinematch that the company is always eager to improve thus in 2006, it held the Netflix Prize Competition that promised to give a million dollar to anyone that could predict the user movie ratings 10 percent better than the Cinematch. The BellKor's pragmatic Chaos team won with the best system with collaborative filtering techniques that gave rating prediction based on historical ratings. The team was conditioned to share the method to the world in order to keep the prize.

As we mentioned earlier, our team studied the algorithm of the winning team in our midterm project in order to get more familiar with data analytics for rating predictions using collaborative filtering.

2 Goals

In this project, we aimed to predict user's ratings of Anime movies and TV shows using the collaborative filtering technique. We compared the accuracy and performance of various algorithms to see which one produced the best result. We later chose one of the model to give a recommendation on a list of movies and TV shows to watch. The ultimate goal is for us to learn and get more experience on working with a big data set. Throughout the project, we used open source libraries and did not pursue any new algorithm.

3 Dataset

We spent a few days finding a data set that would fit our project. We then chose to use the Anime data set available on Kaggle. In this section, we will go over what the data set looks like.

3.1 Kaggle

Kaggle is an online platform that data scientists use to release and retrieve data sets for predictive modeling competition. It is open for everyone to explore and build models, ask questions, give comments on the work. It has a built-in kernel that works like a jupyter notebook. In particulatar, it is an interactive notebook that you can read an article, run a programming code and see the results along with it. It also has tutorials that we followed and a GPU for training bigger data set.

3.2 Dataset Summary

The Anime dataset contains 73,516 users and 12,294 anime from `myanimelist.net`. It has around four million entries of user/item pair with the rating between -1 and 10. The dataset comes with two csv files, $Anime.csv$ and $Rating.csv$.

3.2.1 Anime.csv

This file consists of seven fields including Anime id, Name, Genre, Type, Episode, Rating, and Members. Anime id is a number randomly assigned to each Anime. Name is the title of the Anime. Genre acts like a category of the Dataset such as drama, action, comedy. Type indicates the type of Anime like movie, TV, or others. Episode shows the number of episodes of the Anime. The rating shows overall user rating of the Anime round up to two decimal points. Members is the number of group members in the community that watch the Anime. The total number of entries is 12,294 corresponding to the number of Anime. The first four rows of the data is shown in figure 1

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

Figure 1: $Anime.csv$

3.2.2 Rating.csv

There are three columns in this files. The first one is the user id which is a number randomly assigned to a user. The second one is the Anime id which is a number associated with the Anime and is in the same set as the ones in the $Anime.csv$ file. The last one is the rating that the user gives the specific Anime, from -1 which means there is no rating to 10 which is the best score an Anime can get. The total number of entries is 7,813,737 which indicates the number of user/item rating pairs. The head of Rating.csv is shown in figure 2.

	user_id	anime_id	rating
0	1	20	-1
1	1	24	-1
2	1	79	-1
3	1	226	-1
4	1	241	-1
5	1	355	-1
6	1	356	-1
7	1	442	-1
8	1	487	-1
9	1	846	-1

Figure 2: $Rating.csv$

4 Data Visualization

Visualizing data is a good way to initially look at patterns or characteristics of a dataset. We first used Tableau to display our data then we used Python to do the Principal Component Analysis (PCA) and K Mean Clustering.

4.1 Tableau

Tableau is an interactive data visualization software that is commonly and professionally used in the industry.

We used Tableau to visualize and produce five plots included in this section. The first one below shows different genre of our dataset. The bigger texts indicate more densed genre or the genre with a larger amount of Anime. Similarly, the small genre texts mean there are less Anime that falls into the genre. Note that one Anime can be in multiple genres. There are 38 genres in total as shown in the figure 3.

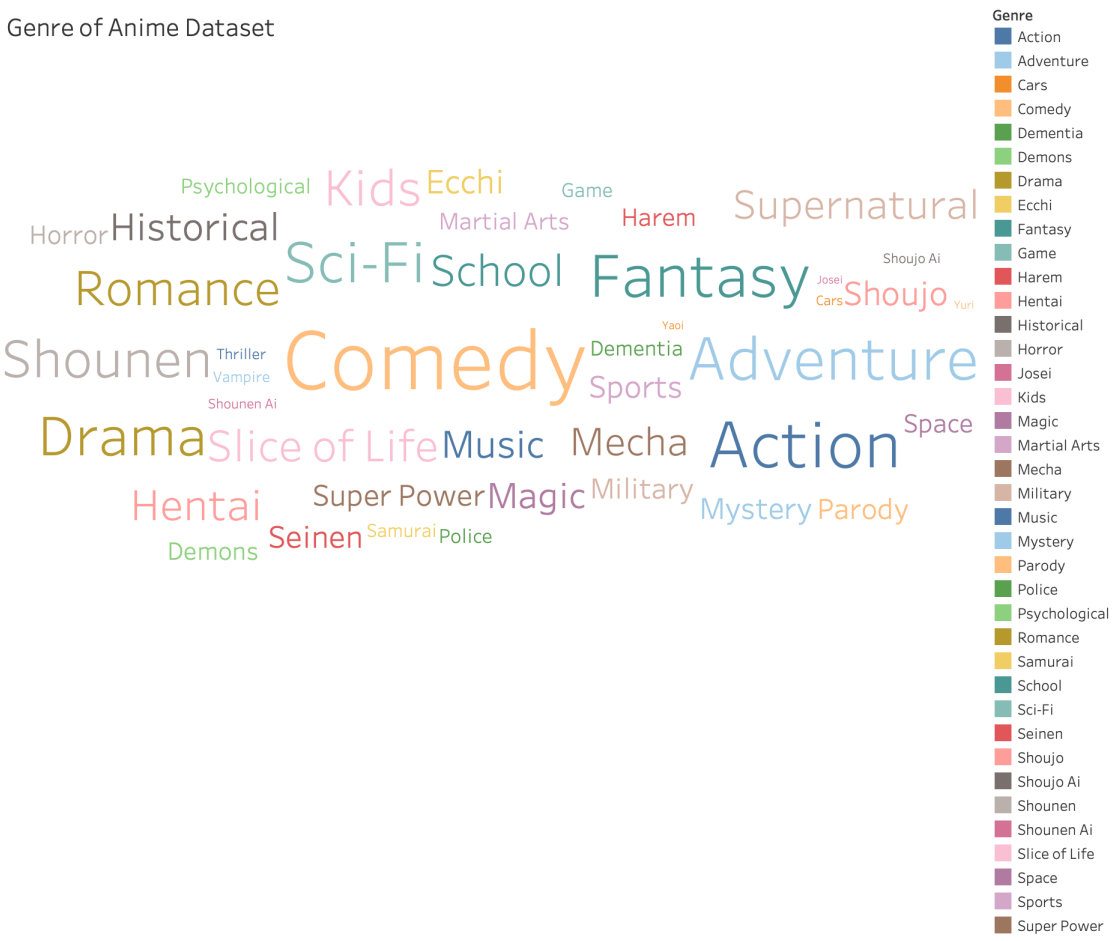

Figure 3: Genre of Anime Dataset

The second plot shows the user frequency of the dataset. For example, the first left most bar says there are over 8,000 users that rate between 0 to 100 Anime in this dataset. The next bar says there are around 5,000 users that rate between 100 to 150 Anime in this dataset, and so on. On a side note, we later cut the data with the frequency of 130 forward so that we only considered users that rate less than 130 Anime to keep the dataset within a reasonable size. See figure 4 for more detail.

The third plot looks similar to the second plot. It shows the member frequency of the dataset. For example, the first left most bar says there are almost 750 items that have between 0 to 20,000 members in this dataset. The next bar says there are around 400 items that have between 20,000 to 40,000 members in this dataset, and so on. On a side note, we later cut the data with the frequency of 50,000 backward so that we only considered items that have at least 50,000 members keep the dataset within a reasonable size. See figure 5 for more detail.

The fourth plot is another bar graph showing various items in each type. The Anime type TV has the largest amount of Anime which is over $3,500$, followed by OVA with $3,300$, and movie with $2,400$. Note that OVA stands for Original Video Animation that is made without any other showings than in home video formats. ONA or Original Net Animation is known as Web Anime and is released on the internet. See figure 6.

The last plot shows the data distribution of the Anime type TV in pink and the type Movie in blue. Notice that the average rating of TVs is around 7 which is similar to the average rating of the movies but the movie data spreads out more thus appears shorter. The TV data is more dense or has more numbers of items with closer ratings thus the shape looks tall. This is partly because TV data consists of more items as we talked about in figure 7.

After we filtered the data, we had $11,107$ users and $1,112$ Anime.

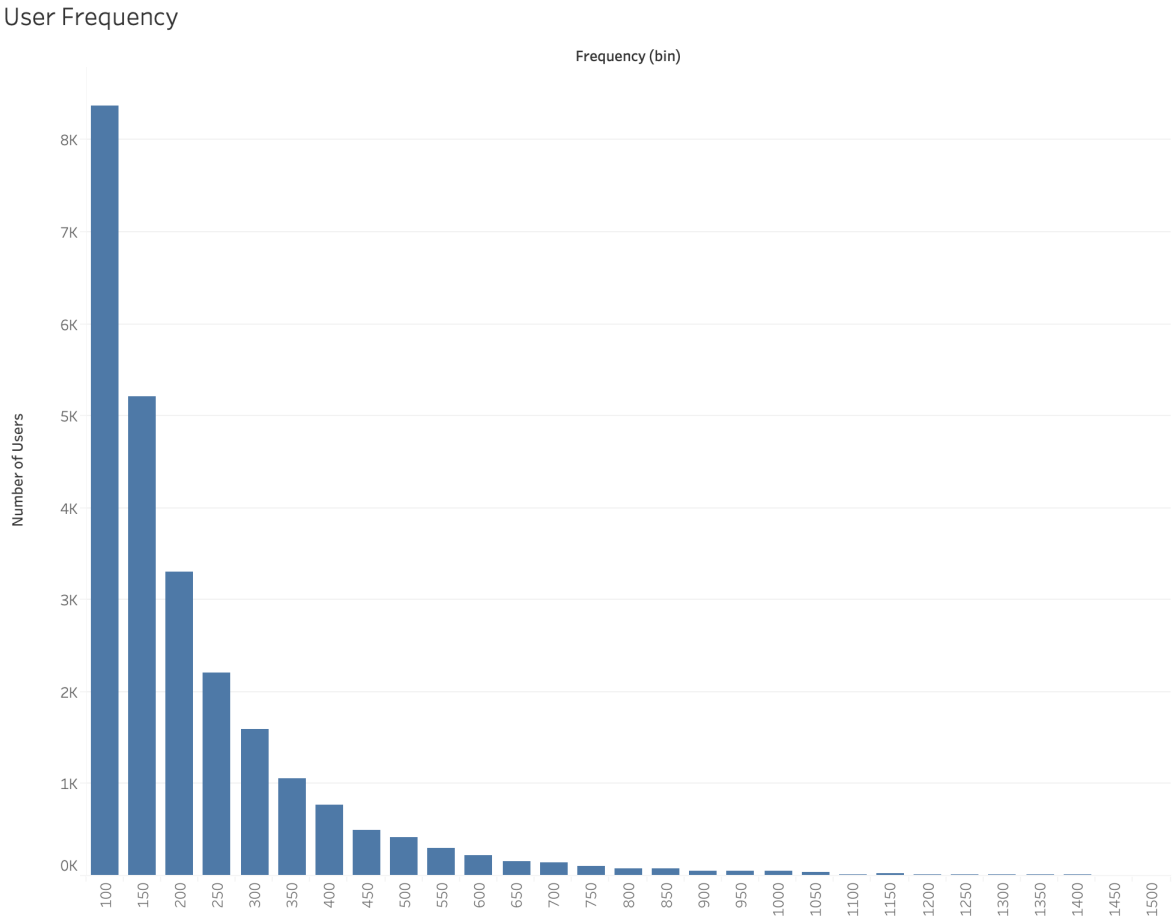

Figure 4: User Frequency

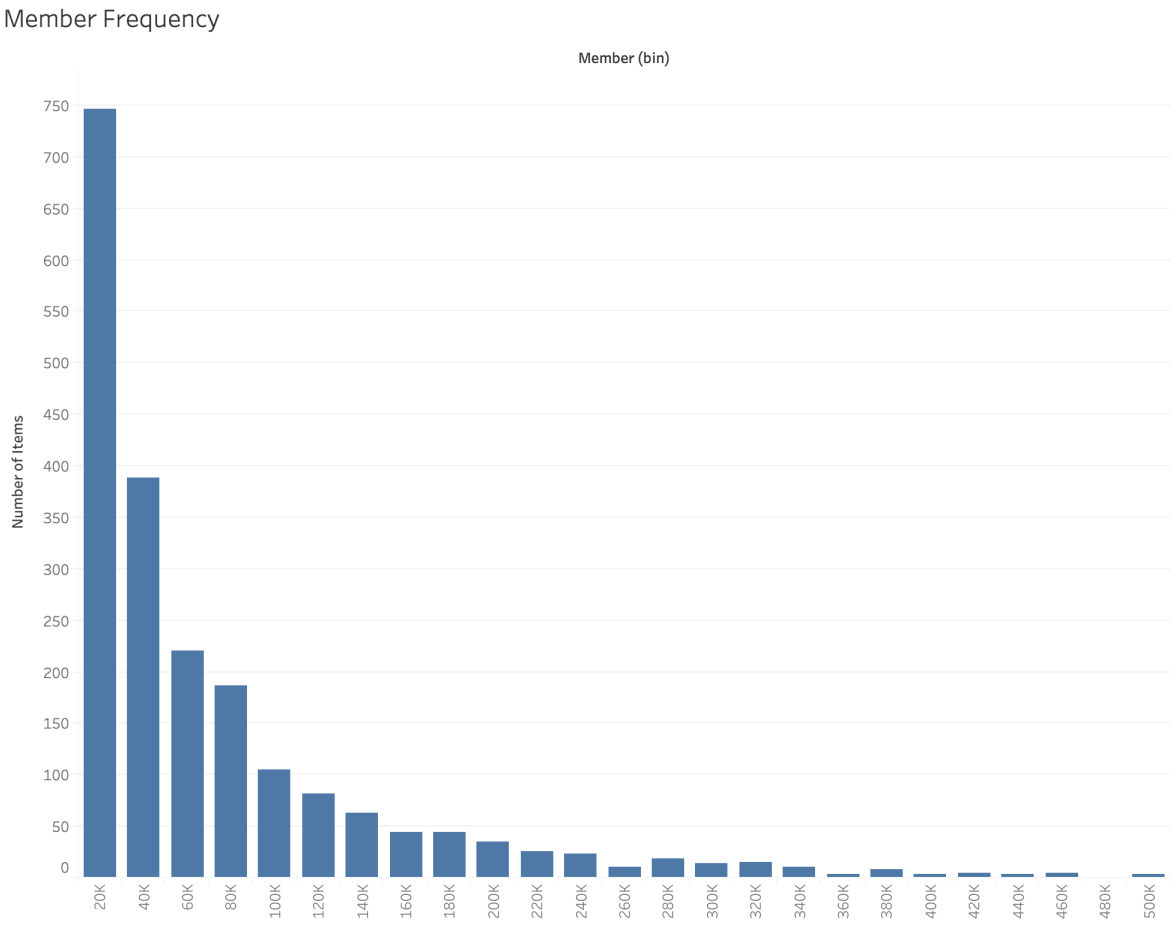

Figure 5: Member Frequency

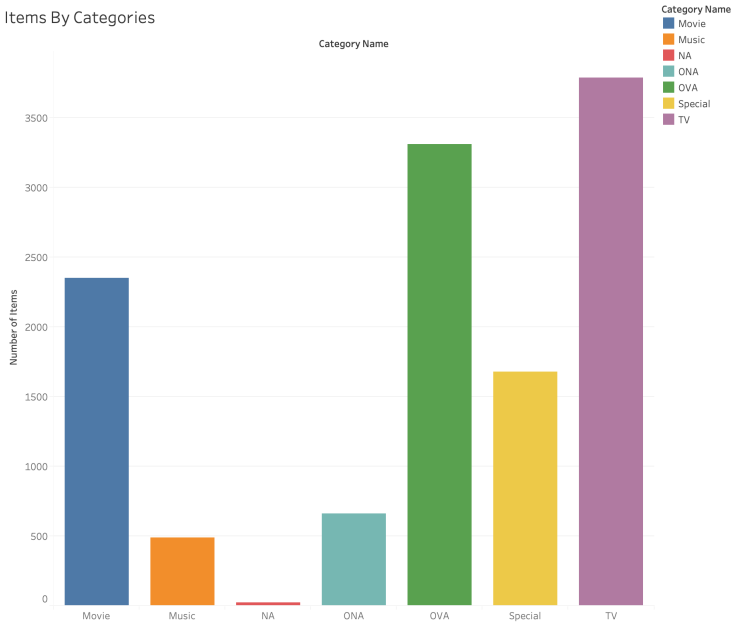

Figure 6: Items by Categories

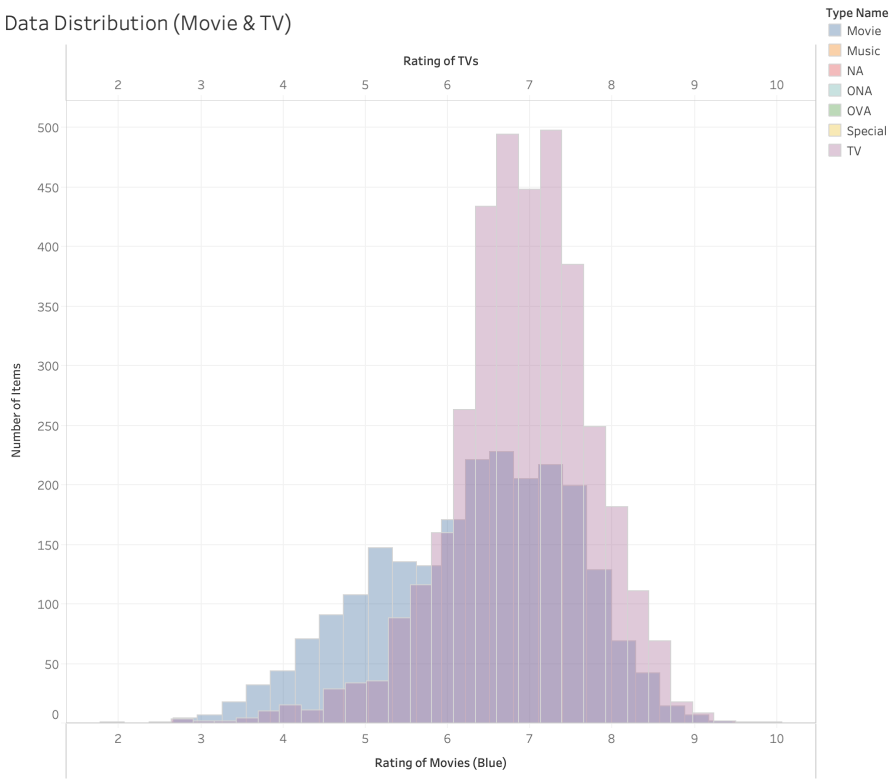

Figure 7: Data Distribution between two Anime Types

4.2 Data Visualization in Python

In addition to Tableau, we also followed a tutorial in Kaggle to visualize our filtered data. There are two main methods that we implemented. The first one is PCA and the second one is K Mean clustering.

4.2.1 PCA

Principal component analysis or PCA is a method that converts original variables to a new set of variables which are a linear combination of the old ones. It reduces the dimension of data for clustering and visualizing.

To summarize, it creates a new dimension data set then computes mean, covariance, eigenvectors, and eigenvalues. It then sorts and chooses eigenventors with the largest eigenvalues and uses a result to transform samples into a new subspace.

$$cov(X, Y) = E[(X - E[X])(Y - E[Y])] \tag{1}$$

We implemented PCA in python with the $sklean$ library. We set the desired dimension of PCA to 3 then fitted an transformed the data set and plotted it. The result was shown in figure 8.

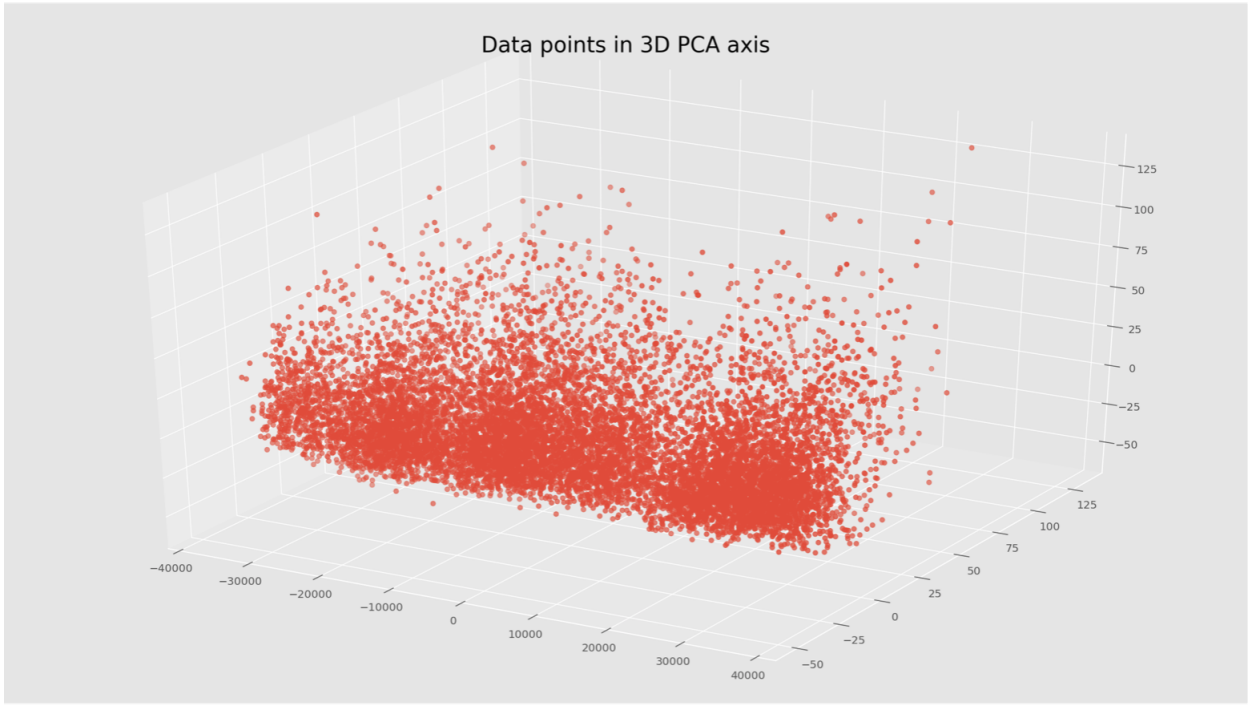

Figure 8: 3D Data Points in PCA axis

4.2.2 K-Mean Clustering

We selected the number of k for clustering by plotting the inertia against the number of cluster. The result in figure 9 showed the $k = 2$ has the most inertia of 1.2.

We also plotted the silhouette score againt the number of cluster. The plot was shown in figure 10.

From two plots, we chose $k = 4$ as our K Means value. Now we re-plotted our filtered dataset in PCA axis with cluster coloring. See figure 11.

Figure 12 represents our data divided into 4 clusters in two dimensions on PCA axis. The red dots represent the center weight of each cluster.

Lastly, we looked at the characteristic of each cluster. In particular, we found out the favorite genre of users in each cluster. The results were shown in figure 13.

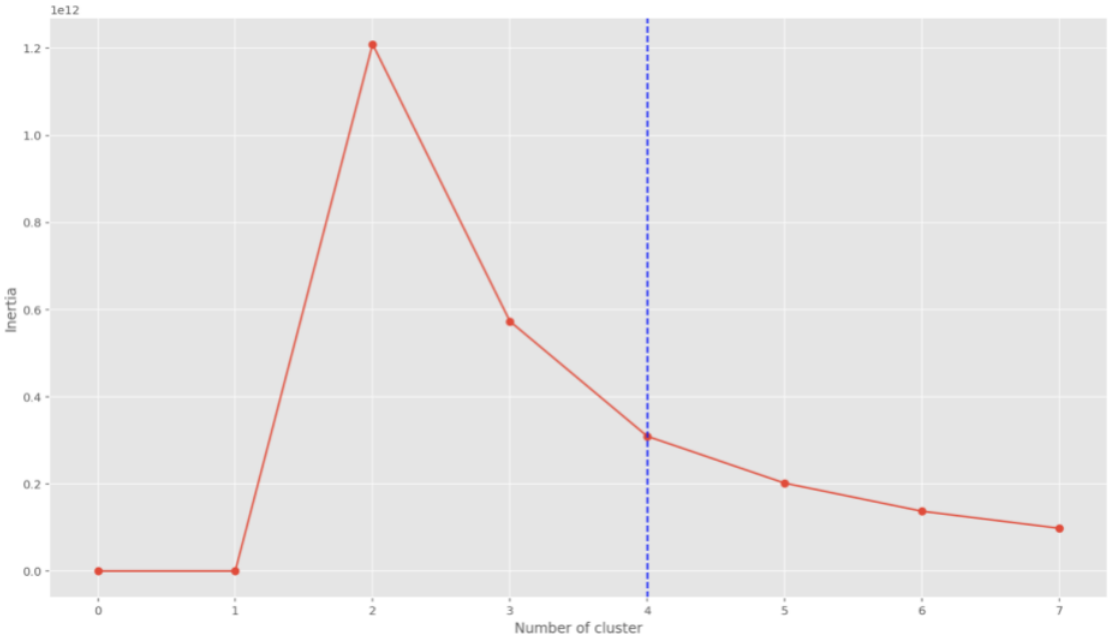

Figure 9: Inertia VS Clusters

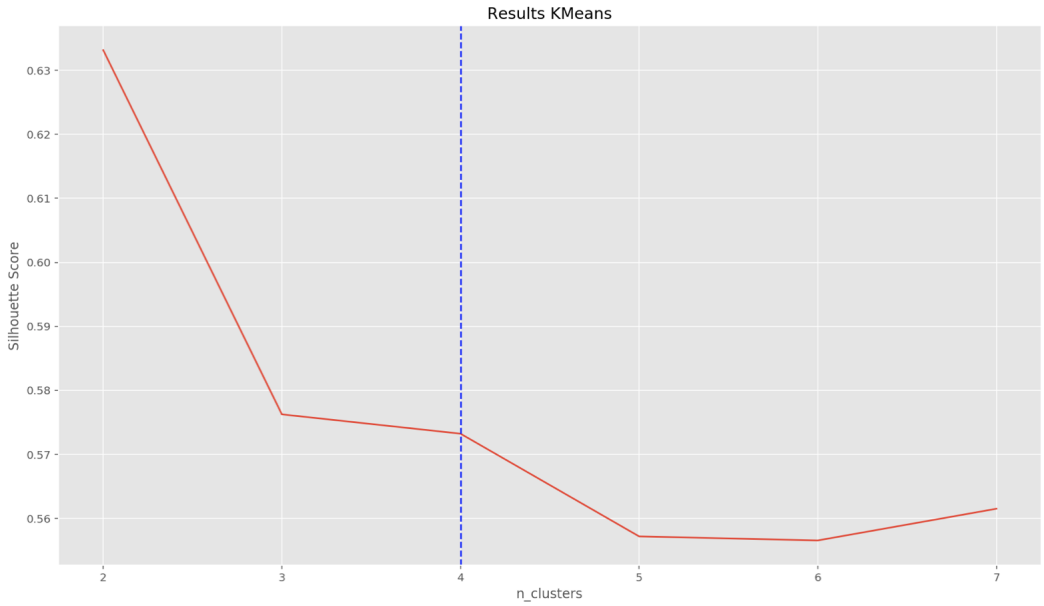

Figure 10: K-Means Result

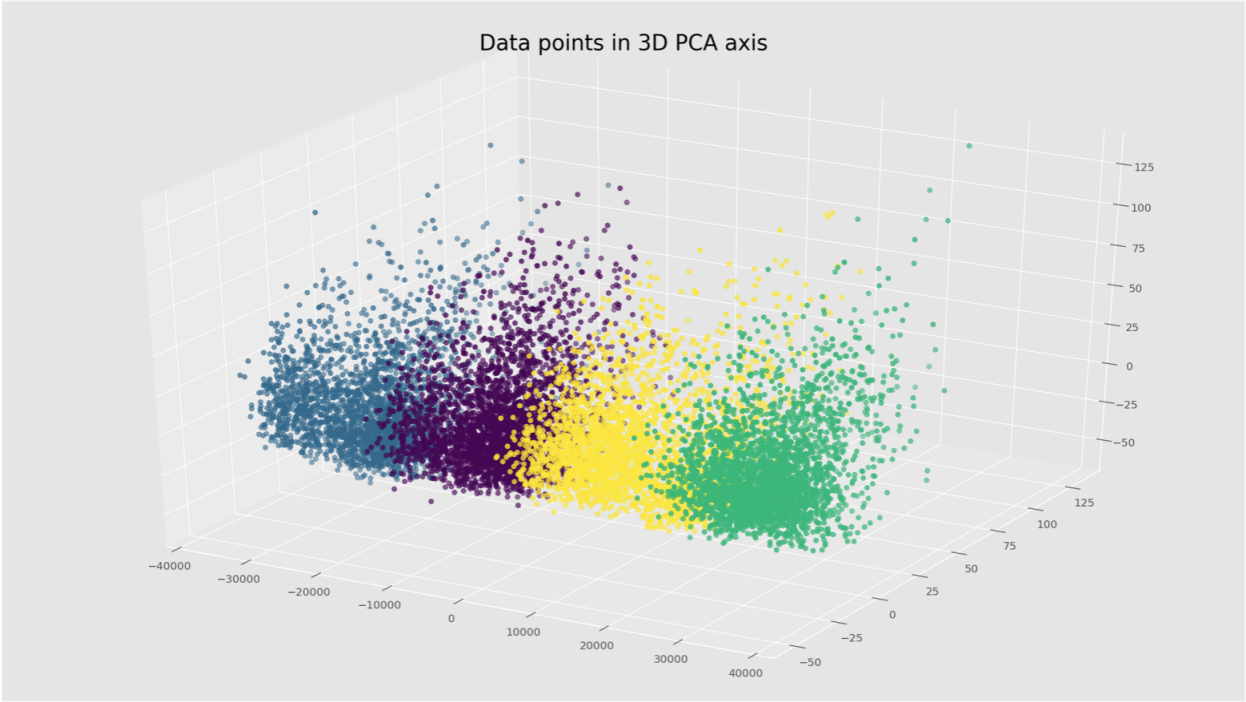

Figure 11: 3D Data Points in PCA axis with Clusters

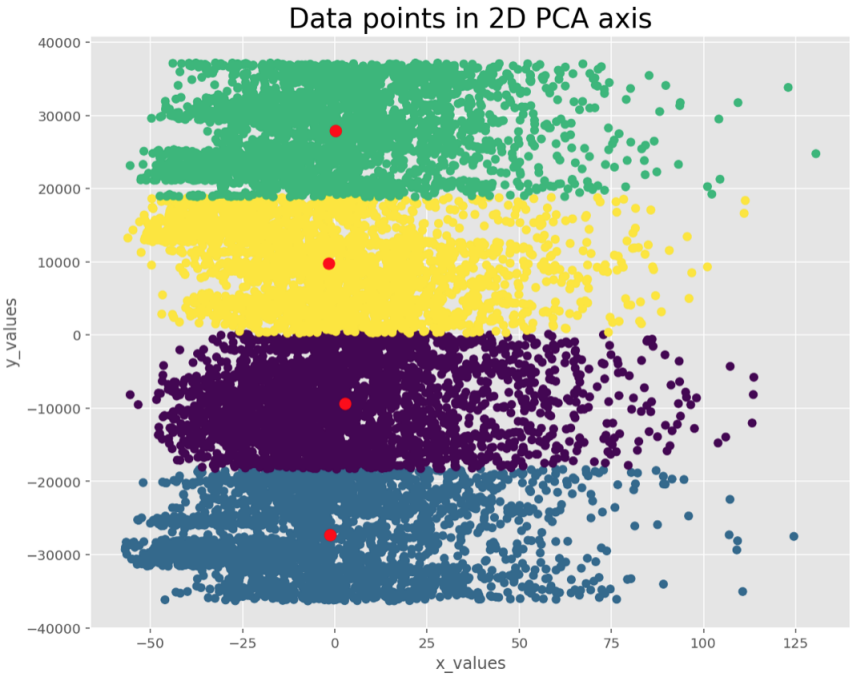

Figure 12: 2D Data Points in PCA axis with Clusters

Figure 13: Anime Genre for indivial Cluster

5 Data Cleaning

Since the data is large, it is not practical for our processing and time resources. Thus, we eliminate some data the does not represent the majority of data. Then, The processed data from "anime.csv" and "rating.csv" are shaped into a required format for the models in library "Surprise" which we discuss later in 6.3.

5.1 Filtering Anime.csv

The popular anime represent the majority of the anime dataset. There are several ways to determine the popularity of a anime. "myanimelist.net", the collector of this dataset, create a feature called "member" which represents number of community members that are in this anime's "group"(this is explained in more detail in 3.2.1). The higher value in member attribute of each anime means the more people that are into the anime. Therefore, we use "member" to determine the popularity of anime. We defines unpopular anime as anime with "member" less than 50,000. Those anime are discarded from our perspective in this predictor. We use only anime with "member" more than or equal to 50,000. The rating from use on unpopular anime are eliminated from "rating.csv" which is discuss next in subsection "Rating". All other features of "Anime.csv" are eliminated also to prevent overfitting.

5.2 Filtering Rating.csv

The ratings, from scale -1 and 1-10, of all user are recorded in "Rating.csv"(more detail discussion is in 3.2.2). We decide that ratings -1 are implicit information. It may cause over-fitting on models we use. Moreover, our processing power and time constrains the viable size of data set. Therefore, we eliminate all -1 rating from the "Rating.csv". In continuing, we determine the adequate information by user frequency. The incomplete data may lead to an imprecise prediction. We consider the users with user frequency below threshold as incomplete data. Thus, we don't try to recommend those user base on their regimental information. Therefore, we eliminate all the ratings rated by users with user frequency below 130 to facilitate the model with the complete data set. To recap, all filtering process on "Rating,csv" are eliminating all ratings containing -1 rating, removing user's data with user frequency below 130, and, finally, we delete all data involving anime with "member" less than 50,000

5.3 Formatting for python model

The required format for dataset is the same format as in "Rating.csv". Therefore, after we filter "Rating.csv", the dataframe is ready to be used.

5.4 Formatting for R model

In R, We merge the "Rating.csv" and "Anime.csv" to "user_data_filtered" which contains user_ID, anime_ID which the user rated, and the rate that the user rated the anime as below figure, the figure is "user_data_filtered" row 1 to 10.
14

	user_id	anime_id	mean_rating
1	1	8074	10
2	1	11617	10
3	1	11757	10
4	1	15451	10
5	2	11771	10
6	3	20	8
7	3	154	6
8	3	170	9
9	3	199	10
10	3	225	9

Figure 14: user_data_filtered

6 Anime rating Predictor

6.1 Models

Several Collaborative Filtering algorithms are tested. Each model algorithm comes from surprise library. The following part is the equations behind each built in function from the Surprise library documentation.

6.2 Notation

- $\mathbf{R}$: the set of all ratings.
- $\mathbf{R}_{train}$, $\mathbf{R}_{test}$ and $\widehat{\mathbf{R}}$: the training set, the test set, and the set of predicted ratings.
- $\mathbf{U}$: the set of all users. u and v denotes users.
- $\mathbf{I}$: the set of all items. i and j denotes items.
- $\mathbf{U}_i$: the set of all users that have rated item i.
- $\mathbf{U}_{ij}$: the set of all users that have rated both items i and j.
- $\mathbf{I}_u$: the set of all items rated by user u.
- $\mathbf{I}_{uv}$: the set of all items rated by both users u and v.
- $\mathbf{r}_{ui}$: the true rating of user u for item i.
- $\widehat{\mathbf{r}}_{ui}$: the estimated rating of user u for item i.
- $\mathbf{b}_{ui}$: the baseline rating of user u for item i.
- μ : the mean of all ratings.
- μ_n: the mean of all ratings given by user u.
- μ_i : the mean of all ratings given to item i.
- σ_u : the standard deviation of all ratings given by user u.
- σ_i : the standard deviation of all ratings given to item i.
- $N_k^i(u)$: the k nearest neighbors of user u that have rated item i. This set is computed using a similarity metric.
- $N_k^u(i)$: the k nearest neighbors of item i that are rated by user u. This set is computed using a similarity metric.

6.2.1 SVD

$$\widehat{\mathbf{r}}_{ui} = \mu + b_u + b_i + q_i^\intercal p_u$$

b_u and b_i are baseline terms which can be calculated from minimizing regularized squared error:

$$\sum_{r_{ui} \in \mathbf{R}_{train}} (r_{ui} - (\mu + b_u + b_i))^2 + \lambda(b_u^2 + b_i^2)$$

q is user factor.

p is item factor.

The terms apply similarly to SVD++ and other algorithms involving the terms.

6.2.2 SVD++

$$\widehat{\mathbf{r}}_{ui} = \mu + b_u + b_i + q_i^\intercal (p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_i)$$

y_i is the new set of item factor that capture the implicit ratings.

6.2.3 NMF

$$\widehat{\mathbf{r}_{ui}} = q_i^\intercal p_u$$

To estimate all the unknown, we minimize the following regularized squared error:

$$\sum_{r_{ui} \in \mathbf{R}_{train}} (r_{ui} - \widehat{r}_{ui})^2 + \lambda(b_u^2 + b_i^2 + ||q_i||^2 + ||p_u||^2)$$

6.2.4 Co Clustering

$$\widehat{\mathbf{r}}_{ui} = \overline{C}_{ui} + (\mu_u - \overline{C}_u) + (\mu_i - \overline{C}_i)$$

$\overline{C}_{ui}$ is the average rating of co-cluster C_{ui}

$\overline{C}_u$ is the average rating of u's cluster.

$\overline{C}_i$ is the average rating of i's cluster.

6.2.5 K-nearest neighbor

Similarity between users

$$\widehat{\mathbf{r}}_{ui} = \frac{\sum_{v \in N_i^k(u)} \text{sim}(u, v) \cdot \mathbf{r}_{vi}}{\sum_{v \in N_i^k(u)} \text{sim}(u, v)}$$

Similarity between items

$$\widehat{\mathbf{r}}_{ui} = \frac{\sum_{j \in N_u^k(j)} \text{sim}(i, j) \cdot \mathbf{r}_{uj}}{\sum_{j \in N_u^k(j)} \text{sim}(i, j)}$$

where $\text{sim}(i, j)$ is similarity function which is similarity function of Pearson correlation.

6.2.6 Slope One

$$\widehat{\mathbf{r}}_{ui} = \mu_u + \frac{1}{|\mathbf{R}_i(u)|} \sum_{j \in \mathbf{R}_i(u)} \text{dev}(i, j)$$

where $\mathbf{R}_i(u)$ is the set of relevant items, i.e. the set of items j rated by u that also have at least one common user with i. $\text{dev}(i, j)$ is defined as the average difference between the ratings of i and those of j:

$$\text{dev}(i, j) = \frac{1}{|\mathbf{U}_{ij}|} \sum_{u \in \mathbf{U}_{ij}} \mathbf{r}_{ui} - \mathbf{r}_{uj}$$

6.2.7 Normal Predictor

We assume that $\mathbf{R}_{train}$ to be normal. where

$$\widehat{\mu} = \frac{1}{|\mathbf{R}_{train}|} \sum_{\mathbf{r}_{ui} \in \mathbf{R}_{train}} \mathbf{r}_{ui}$$

and

$$\widehat{\sigma} = \sqrt{\sum_{\mathbf{r}_{ui} \in \mathbf{R}_{train}} \frac{(\mathbf{r}_{ui} - \widehat{\mu})^2}{|\mathbf{R}_{train}|}}$$

then prediction $\widehat{r}_{ui}$ generate from a normal distribution $\mathcal{N}(\widehat{\mu}, \widehat{\sigma}^2)$

6.2.8 Baseline

$$\widehat{\mathbf{r}}_{ui} = b_{ui} = \mu + b_u + b_i$$

6.2.9 Pearson correlation

The model use Pearson correlation which used to measure the degree of relationship between the two by calculate Pearson Correlation Coefficient to create the similarity function by following formula

$$\text{sim}(u, v) = \frac{\sum_{i \in I_u \cap I_v} (r_{ui} - \mu_u)(r_{vi} - \mu_v)}{\sqrt{\sum_{I_u \cap I_v} (r_{ui} - \mu_u)^2} \sqrt{\sum_{I_u \cap I_v} (r_{vi} - \mu_v)^2}}$$

and we used it to predict the preference of the anime between users by

$$\widehat{\mathbf{r}}_{ui} = \frac{\sum_{i \in I_u \cap I_v} r_{vi} \cdot \text{sim}\ (u, v)}{N}$$

where N is the number of user.

6.2.10 Euclidean

The model use Euclidean distant to measure the similarity between user by the similarity function that

$$\text{sim}(u, v) = \frac{1}{1 + E(u, v)}$$

where E is The Euclidean distance,

$$E(u, v) = \sqrt{\sum_{I_u \cap I_v} (r_{ui} - r_{vi})^2}$$

the recommender formula:

$$\widehat{\mathbf{r}}_{ui} = \frac{1}{\sum_{n=1}^{N} \text{sim}(I_u, I_n)} \sum_{n=1}^{N} \text{sim}(I_u, I_n) \cdot r_{ni}$$

where N is the number of user.

6.3 Surprise Library

Surprise stands for Simple Python Recommendation System Engine. It is created from scikit, a popular Python machine learning tool. Surprise is designed for user to easily interact with recommendation system algorithm. It provide the several models and variety of testing method. Not only build-in function is available, but also the option of creating new model is encouraged. Moreover, the library provide a number of dataset which is in a required format and ready to be used. However, only build-in functions are used in this project. The anime dataset is not in the required format, therefore, we have to reformat the dataset (more detail in 5).

6.4 Recommederlab

Recommenderlab is package in R which provides the infrastructure to develop and test recommender algorithms for rating data. It provide build-in function for simple recommender system such as eveluationscheme function which allow us to create training set and test data. In this project, we use evalutionscheme function in recommenderlab to randomly assigns the proportion of the data specified by train to the training set and the rest is used for the test set.

6.5 K-Cross Validation

Function "cross_validate" from Surprise is used to perform K-Cross Validation. In K-Cross Validation, the dataset is randomly divided into K sets. In other words, the function does not group data base on position. Then, set 1 becomes a validation set while other sets become training set. The process is repeated until every set get a chance to be a validation set. The exaple visualization of the process is shown in figure 15.

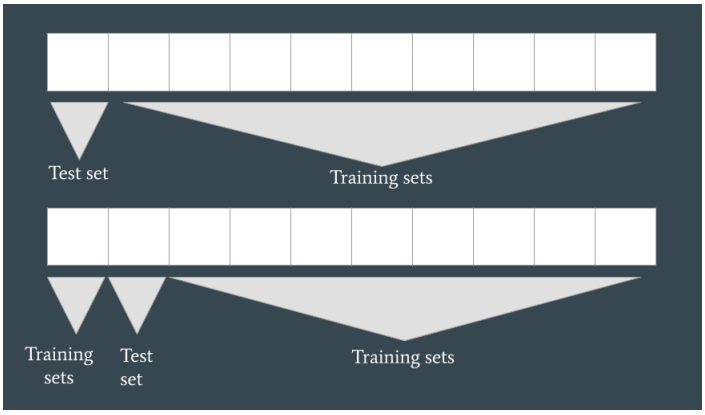

Figure 15: The first two steps of the K-Cross Validation on a dataset

6.6 Results

6.6.1 Benchmark

The benchmark of each algorithm show the accuracy in term of Root Mean Square Error (RMSE) and mean average error (MAE). Both RMSE and MAE are calculated from K-cross validation test (explained in more detail in 6.5). The RMSE and MAE shown in Table 1 are average of RMSE and MAE from K validation set. Runtime column show the performance in term of speed of each algorithm. SVD++ take more than four hours to run. Due to limited time, we skip the algorithm.

(Note: The benchmark is constructed on a PC with Intel Core i5 7th gen (2.5 GHz), 8G RAM, and GPU: GTX 1070ti)

6.6.2 Recommender Example

All of the model we use predicts the missing rating of every anime for each user. Thus, the ranking of anime list can be made for each user. The ranking from high to low score for an individual user allow us to conveniently create a list of recommending anime by excluding the watched ones as shown in figure 6.6.2.

Table 1: Benchmark of collaborative filtering algorithms

Algorithm	**RMSE**	**MAE**	**Runtime(hour)**
SVD	1.065	0.798	0 : 09 : 41
SVD + +	-	-	>4:00:00
NMF	2.884	2.672	0 : 10 : 07
SlopeOne	1.148	0.868	0 : 07 : 26
k − NNBasic	1.135	0.849	2 : 05 : 32
Centeredk − NN	1.128	0.856	2 : 06 : 43
k − NNBaseline	1.113	0.840	2 : 08 : 23
Co − Clustering	1.138	0.863	0 : 04 : 55
Baseline	1.153	0.875	0 : 01 : 23
Random	2.045	1.621	0 : 00 : 56

name	Estimate_Score
Kimi no Na wa.	9.210568
Fullmetal Alchemist: Brotherhood	9.206003
Gintama°	9.180509
Ginga Eiyuu Densetsu	9.151933
Steins;Gate	9.082832
Gintama'	9.076712
Gintama	9.064117
Koe no Katachi	9.035095
Hunter x Hunter (2011)	9.010821
Gintama': Enchousen	9.008594

Figure 16: Top ten recommended anime for user _ id = 1

7 Conclusion

SVD algorithm produces the most accurate result with RMSE = 1.065 and MAE = 0.798. Even though SVD does not use the lowest runtime, its runtime is fairly fast considering the amount of the data. Other algorithms except SVD++, NMF, and Random provide the nearly similar RMSE and MAE. k-NN Baseline used the most runtime of 2:08:23.

8 Future Work

We would incorporate more available information, for example, the genre of anime, number of episode.

9 Acknowledgement

We would like to thank Professor Weiqing Gu for providing us with her large source of knowledge in this class. Without her, we simply would not be able to complete this project and learn so much in such a small period of time.

We are thankful for Mek and Yai, our Thai friends who gave us advices throughout the project. Without them, we would not be able to accomplish this much in the project.

We also want to thank Chelsea and Alireza for listening to our project and taking the class with us. Without them, we would be lonely and have less fun in the class.

References

[1] "Collaborative Filtering." *Recommender Systems,*23 Jan. 2012, recommender-systems.org/collaborative-filtering/.

[2] CooperUnion. "Anime Recommendations Database." Kaggle, 21 Dec. 2016, https://www.kaggle.com/CooperUnion/anime-recommendations-database.

[3] "Welcome to Surprise' Documentation!¶." *Welcome to Surprise' Documentation! - Surprise 1 Documentation,* surprise.readthedocs.io/en/stable/index.html.

[4] Shimodaira, Hiroshi. *Similarity and Recommender Systems.* 20 Jan. 2015, www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn-note02-2up.pdf.