Recommendation System for Music

Betzabeth Narvaeza[a.1], Jiho Kil[b.2], Le Minh Huyen[c.3], Pho Vu[d.4]

[a]*Major, Underwood International College, Yonsei University*

[1]*betzynarv@yonsei.ac.kr,* [2]*robin311@naver.com*

[3]*leminhhuyen@yonsei.ac.kr,* [4]*pvu23@yonsei.ac.kr*

Abstract

Over decades, Hallyu (Korean Wave) has been in an all-time high demand. However, there is still a lack of a recommender system of Korean-only entertainment sources for foreign fans. We aim to provide a recommender system for Korean pop songs based on the idea of user-based collaborative filtering from Netflix.

Keywords: Recommendation system, collaborative filtering, Korean entertainment, Korean pop songs

Introduction

In recent years, South Korea has been the subject of growing attention in the entertainment sector, due to the explosion of popularity for certain agencies and items such as Bangtan Boys (BTS) and Squid Game. However, the concentration of international preferences towards only a few Korean entertainment samples is of possible concern with regards to the sustainability of Korean culture. Recommender systems are able to play an active role to mitigate the issue by expanding the scope of global fans and introducing them to other Korean entertainment items according to their preferences.

Recommender systems have steadily evolved over decades to be applied to many areas involved in online content services such as movies, music, news and books. The democratization of the internet has allowed technology companies to develop different filtering algorithms that provide custom recommendations of their services to the customers. We plan to use samples of

ratings for K-pop songs given by multiple nationalities to create a recommender system for the pop songs

Research question: How effectively can the collaborative filtering algorithm be used to provide users with personalized recommendations of Korean pop songs?

Literature Review

Recommender systems are based on two different categories (or combinations thereof): the content-based approach and collaborative filtering. [2] Collaborative filtering is the most common method to build a recommender system because more information about users is included. Most websites like Amazon, YouTube, and Netflix, Spotify use collaborative filtering as a part of their sophisticated recommendation systems. In this study, we will apply the collaborative filtering approach to create our own recommender system for Korean pop songs.

"Automated collaborative filtering is quickly becoming a popular technique for reducing information overload, often as a technique to complement content-based information filtering systems. In this paper we present algorithmic elements that increase the accuracy of collaborative prediction algorithms" (Herlocker, Konstan, Borchers, & Riedl, 1999, p.1).

"Automated collaborative filtering systems work by collecting human judgments (known as ratings) for items in a given domain and matching together people who share the same information needs or the same tastes" (Herlocker, Konstan, Borchers, & Riedl, 1999, p.1).

"CF analyzes relationships between users and interdependencies among products, in order to identify new user-item associations. For example, some CF systems identify pairs of items that tend to be rated similarly or like-minded users with similar history of rating or purchasing to deduce unknown relationships between users and items. The only required information is the past behavior of users, which might be their previous transactions or the way they rate products" (Yifan, Koren, & Volinsky, 2008, p.1).

Study Design and Methodology

1. Collaborative Filtering

Collaborative filtering is a method for making predictions about the preferences of a user by collecting ratings of items from many users. It is currently the most widely employed and successful kind of recommendation system, with different variations being deployed by online marketing firms such as Netflix. The main intuition of collaborative filtering is to search for users with similar interests, and create a recommendation list based on the preexisting ratings. There are a few foundational assumptions underlying collaborative filtering. First, it is possible to access information regarding explicit or implicit ratings of the given item. Second, we can pinpoint similar preferences based on existing user-item ratings data. Third, users with similar interests will have consistent ratings patterns.

The similarity metric employed in the recommendation algorithm for the research is the cosine similarity. The value for every user vector based on the user of interest is extracted from the user-item matrix to derive the similar users. The predicted rating of the user is computed by

summing the weighted average of all the neighboring user ratings, with the weights being the cosine similarity.

$$r_{ij} = \frac{\sum_k Similaries(u_i, u_k) r_{kj}}{number\ of\ ratings} \quad Cosine\ Similarity : Sim(u_i, u_k) = \frac{r_i \cdot r_k}{|r_i||r_k|} = \frac{\sum_{j=1}^{m} r_{ij} r_{kj}}{\sqrt{\sum_{j=1}^{m} r_{ij}^2 \sum_{j=1}^{m} r_{kj}^2}}$$

The amount of users to consider for the predicted ratings is an important variable in the research which impacts model accuracy. The root mean squared error (RMSE) is employed as the metric for comparing and deriving the optimal value of *k* in the application of the k nearest neighbors in the pop song data. In the data analysis, we computed the RMSE for a wide range of values for *k*, and chose the optimal *k* which produces the lowest error.

Consideration of bias plays a major role in the accuracy of recommender algorithms. Some users may be inclined to provide high ratings for all movies on average, while others tend to be extremely harsh and grade low points overall. Therefore, ignoring the unique disposition of the consumers will inevitably lead to bias in the recommender system. In order to mitigate the bias, the predicted ratings formula can be improved by taking into account the deviation of all users from their usual tendency instead of the absolute ratings values. The weighted average of the deviations for all neighboring users leads to the predicted deviation, to which we add the average rating of the user in question to derive the predicted ratings.

$$s(i,j) = \bar{r}_i + \frac{\sum_{i'} w_{ii'} \left\{ r_{i'j} - \bar{r}_{i'} \right\}}{\sum_{i'} w_{ii'}}$$

We combine the bias-from-mean formula and k nearest neighbors in our user-based collaborative filtering to create a recommendation system for the referenced Kpop songs.

Data Analysis

1. Data Pre-Processing

In order to make a recommender system of Korean pop songs, we have two raw datasets to handle with. The first dataset includes information on users such as nationality, participant_id, stimID; a few information about the songs and mainly participants' mood ratings. For the first raw dataset, our target is the ratings for Korean pop songs only. Hence, we filter the rows that 'song_country' = KR. When it comes to the second dataset, it shows information about 360 songs, each song has 24 features.

```
participant = pd.read_csv('/content/gdrive/MyDrive/Group 3/Data/K-pop songs Dataset/Data 1 - raw-participant.csv',  encoding='cp949')
```

```
participant = participant[participant['song_country'] == 'KR']
participant
```

	rater_nationality	participant_id	stimID	repeated_trial	song_country	song_year	singer_gender	preference_5pt	familiarity_4pt	calm	tense	cheerful	sad	danceable	love	dreamy	electronic	energy
0	American	participant_US1	KR-2014-block11	f	KR	2014	m	4	1	1	1	4	1	4	4	4	4	4
1	American	participant_US1	KR-2013-block11	f	KR	2013	f	1	1	2	1	4	1	4	4	2	3	4
5	American	participant_US1	KR-2012-block11	f	KR	2012	f	5	1	2	1	2	2	3	4	4	2	3
7	American	participant_US3	KR-2018-block11	f	KR	2018	m	2	1	2	2	1	3	1	2	2	1	2
10	American	participant_US3	KR-2015-block11	f	KR	2015	m	3	1	1	1	3	1	2	2	2	3	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11490	Brazilian	participant_BR6	KR-2013-block5	f	KR	2013	m	2	1	2	1	2	1	2	1	1	2	2
11491	Brazilian	participant_BR6	KR-2015-block5	f	KR	2015	f	3	1	2	1	2	1	3	3	2	2	2
11492	Brazilian	participant_BR6	KR-2011-block5	f	KR	2011	f	1	1	2	3	1	3	1	1	1	1	1
11494	Brazilian	participant_BR6	KR-2017-block5	f	KR	2017	f	3	1	2	1	2	1	2	1	2	2	2
11499	Brazilian	participant_BR6	KR-2019-block5	f	KR	2019	f	2	1	2	1	1	1	2	1	1	2	2

3845 rows × 18 columns

```
KR_song = pd.read_csv('/content/gdrive/MyDrive/Group 3/Data/K-pop songs Dataset/Data 2 - song_chart.csv',  encoding = 'utf-8')
```

```
KR_song
```

_source	year	stimID	song_query	artist_query	song_result_spotify	artist_result_spotify	spotify_trackid	preview_url	...	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms	time_signature
on Chart	2010	KR-2010-block1	Here I Am	포맨 (4men), 美	Here I am	4MEN	2HeHlpLSY7SWKLjsZdUbjQ	https://p.scdn.co/mp3-preview/8e5d94375aa154f3...	...	-7.834	1	0.0335	0.8540	0.000000	0.1030	0.101	129.552	235563	4
on Chart	2010	KR-2010-block2	반하늘의 별을..	양정승 (kiroy Y), Kcm & No Noo	반하늘의 별을..	Kiroy Y	3zNyjuCpWg3iYWs9PgbDM	https://p.scdn.co/mp3-preview/0a34821c28bbafbf2...	...	-4.578	1	0.0609	0.2410	0.000001	0.1160	0.406	153.807	225787	4
on Chart	2010	KR-2010-block3	Y	엠블랙 (mblaq)	Y	MBLAQ	3xVGwwax3gAPAkJewnB2eZ	https://p.scdn.co/mp3-preview/c9e9462cf7b31162...	...	-4.828	0	0.1420	0.0298	0.000004	0.0967	0.875	122.952	208523	4
on Chart	2010	KR-2010-block4	사랑한다는 말	남규리	사랑한다는 말	Nam Gyu ri	1o68HlqheZWvKk4m9DnCK0	https://p.scdn.co/mp3-preview/e0a64b47cc65954b...	...	-7.516	1	0.0372	0.8130	0.000000	0.1110	0.172	134.063	248099	4
on Chart	2010	KR-2010-block5	사랑, 우정 그리고... (with No Noo & 양정승)	제이비 (jb)	사랑, 우정 그리고...	JB	2ovX9QgISOm2ouAIhtV3gB	https://p.scdn.co/mp3-preview/836553176e549f53...	...	-6.121	1	0.0336	0.8170	0.000000	0.1090	0.499	171.757	218906	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Crowley	2019	BR-2019-block8	Saudade Infinita	Roberta Miranda	Saudade Infinita	Roberta Miranda	6J2ySKkqPkB0oB2laqYO1e	https://p.scdn.co/mp3-preview/e98bd3f00838fbf2...	...	-6.369	1	0.0274	0.2240	0.000000	0.1250	0.582	105.960	194893	4
Crowley	2019	BR-2019-block9	Combate	Um44k	Combate	UM44K	2gVByu7sEnewmsVa4CnbOG	https://p.scdn.co/mp3-preview/76335d9d17acdd5e...	...	-6.116	1	0.1450	0.3150	0.000000	0.0974	0.831	88.040	179253	4
Crowley	2019	BR-2019-block10	Salvou Meu Dia	Mc Kevinho Part.Gusttavo Lima	Salvou meu dia (Participação especial de Gustt...	MC Kevinho, Gusttavo Lima	7CBImG41isOs2r7lQU7CCy	https://p.scdn.co/mp3-preview/cbcc7ed512a1508...	...	-2.688	0	0.0807	0.4090	0.000000	0.3260	0.830	162.077	163666	4
Crowley	2019	BR-2019-block11	500 Metros	Paula Mattos Part.Lucas Lucco	500 metros (Participação especial de Lucas Lucco)	Paula Mattos, Lucas Lucco	2mo6GXEG5IwAA8CfBVIGAf	https://p.scdn.co/mp3-preview/55f5b6b3309ab5ec...	...	-3.862	1	0.0510	0.6190	0.000000	0.1060	0.686	123.872	175326	4
Crowley	2019	BR-2019-block12	Toquin Do Violao	Sandro Cesar	Toquin do Violão	Sandro César	1qrrjgSAnpuw8Pf2NTr5eN	https://p.scdn.co/mp3-preview/f8c44e7fd94077a1...	...	-3.283	1	0.0699	0.3630	0.000000	0.0928	0.735	147.880	167211	4

We then merged two dataframes into one, using the primary key 'stimID'. Before jumping to the recommender algorithms, we selected only three necessary variables which are 'participant_id', 'song_query', and 'preference_5pt'. The 'preference_5pt' variable is very similar to the ratings variable that we used to handle in the Netflix Prize Dataset.

After that, we got the final data frame for applying recommendation algorithm 3845 samples and saved as a new file named 'Ratings'

```
ratings = song_pref[["participant_id", "song_query", "preference_5pt"]]
ratings
```

	participant_id	song_query	preference_5pt
0	participant_US1	Thunder	4
1	participant_US1	그녀의 사무실	1
2	participant_US1	사랑아	5
3	participant_US3	영원히	2
4	participant_US3	너를 그리다	3
...	...	...	...
3840	participant_BR6	만약에 말야 (전우성 Solo)	2
3841	participant_BR6	Wifey (feat. Mc몽)	3
3842	participant_BR6	Break Up	1
3843	participant_BR6	낡은 그리움	3
3844	participant_BR6	Flash	2

3845 rows × 3 columns

```
ratings.to_csv('/content/gdrive/MyDrive/Group 3/Data/K-pop songs Dataset/Ratings.csv', index = None, header=True )
```

2. Data Analysis

We first divided the pre-processed dataset into train and test sets in order to assess the accuracy of the recommendation model using RMSE. We then converted the train dataset into a participant_id - song_query matrix for the computation of the cosine similarity. The rows represent the users in the aforementioned mood ratings research, and the columns describe the song titles that were evaluated by the research participants. There are many unrated songs shown by the NaN values, which is normal in the case of user-item matrices. A dense matrix implies less necessity for recommendation since almost all songs have already been rated.

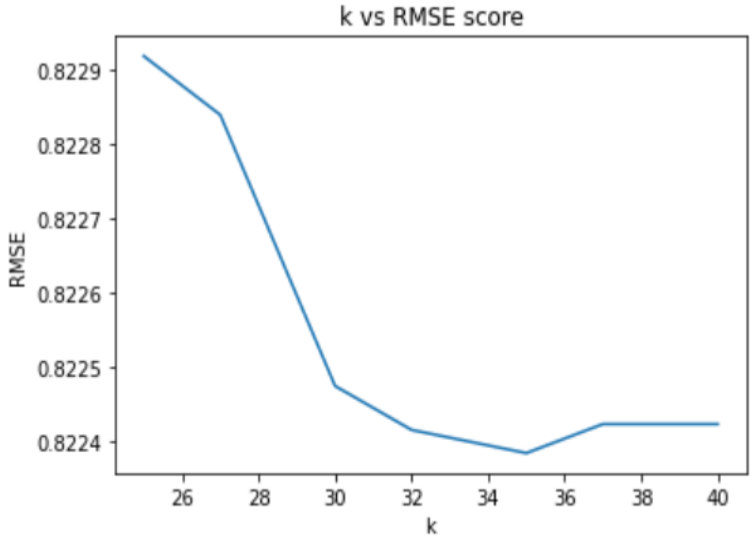
song_query	..더 라면	119 (feat. Gray)	Adios (prod. Boycold)	Apple Pie	Bad Girl Good Girl	Band	Beep	Break Up	Cafe	Cake Love (prod. By 검정치마)	...
participant_id											
participant_BR1	NaN	NaN	5.0	4.0	5.0	4.0	NaN	NaN	NaN	NaN	...
participant_BR10	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
participant_BR11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	...
participant_BR12	NaN	NaN	2.0	1.0	3.0	NaN	NaN	NaN	NaN	NaN	...
participant_BR13	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
...	...	...	...	...	...	...	...	...	...	...	...
participant_US54	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
participant_US6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
participant_US7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	...
participant_US8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	NaN	...
participant_US9	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.0	...

166 rows × 120 columns

We defined an accuracy score which takes in the recommender model and the neighbor size, and returns the RMSE of the combination. The cosine similarity was calculated after replacing the NaN values with zeros. The resulting similarity matrix, which is a diagonally symmetric matrix displaying the participant_ids for both indexes, contains the similarity values of the row-column intersection. The diagonal values are all equal as 1, since they represent the similarity of the same user.

participant_id	participant_BR1	participant_BR10	participant_BR11	participant_BR12	participant_BR13	participant_BR14
participant_id						
participant_BR1	1.000000	0.000000	0.319361	0.416791	0.000000	0.000000
participant_BR10	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
participant_BR11	0.319361	0.000000	1.000000	0.000000	0.000000	0.000000
participant_BR12	0.416791	0.000000	0.000000	1.000000	0.000000	0.000000
participant_BR13	0.000000	0.000000	0.000000	0.000000	1.000000	0.325214
...	...	...	...	...	...	...
participant_US54	0.000000	0.877580	0.000000	0.000000	0.000000	0.000000
participant_US6	0.319899	0.000000	0.385068	0.092903	0.000000	0.000000
participant_US7	0.333351	0.000000	0.000000	0.000000	0.252012	0.621658
participant_US8	0.000000	0.000000	0.000000	0.000000	0.461211	0.808654
participant_US9	0.000000	0.485022	0.109841	0.000000	0.140004	0.298426

We created the recommendation algorithm which designates the user and song of interest. In the process, we removed users who didn't participate in the song rating, due to their irrelevance in the prediction. For the comparison of the accuracy scores, we experimented a range k nearest neighbor values from 25 to 40 and determined the k with the lowest RMSE value. The RMSE corresponding to k=35 was the lowest with the value of 0.82, and was employed for the recommendation model.



After determining the optimal k, we converted the whole dataset into the user-item matrix to create the actual recommender algorithm. We mapped the computed predicted ratings to their corresponding song, and sorted the values from highest. Our sample recommender shows the top 10 songs to be recommended to a random user entered into the model.

```
# Recommendation
recommender('participant_BR11')
```

```
['밤하늘의 별을..',
 'Rumor',
 'Think About` Chu (prod. By 박근태)',
 '내 눈에만 보여',
 'Fingertips',
 'So Hot',
 '살자 (the Cure)',
 'Moya (모야)',
 'White Day',
 'I`m In Love']
```

```
# Recommendation
recommender('participant_US6')
```

```
['Rumor',
 '내 눈에만 보여',
 '잘가라',
 'Only U',
 '커플',
 '같은 꿈',
 '내 모습을',
 'White Day',
 '실례해도 될까요',
 'Fingertips']
```

3. Validity of Analysis

In the previous demonstration of the simple recommender system on the Netflix prize dataset, we did not take into account the bias-from-mean factor as well as the k nearest neighbor. We considered the similarity of all existing users, and computed the weighted mean of the ratings instead of the deviation. In order to compare the accuracy of the two models, we computed the

RMSE for the initial recommender model. The score showed a value of approximately 1.05, which was higher than the model score with the bias-from-mean adjustment and a nearest neighbor value of 35 (0.82). Therefore, our modified recommender model has shown better accuracy in terms of error and higher improvement in terms of performance.

Conclusion, Suggestion and Further Developments

The purpose of the research was to provide an insight into a recommender system for Kpop songs for international users. Previous existing recommendations were mainly based on Korean fans and users whose perception of items outside the mainstream hits such as BTS songs could be different from users of other nationalities. Therefore, we devised a user-based sample recommendation system which would consider users of different nationalities and their unique sentiments by gathering data from the mood ratings research of three countries.

Although the adjusted recommender algorithm showed better accuracy, one aspect which could improve the performance is the availability of more users. The increase in users can potentially draw more within the range of the neighbor size who have higher similarity values, allowing for lower RMSE and enhanced recommendation. Nevertheless, there is considerable significance in the research which enables application of the recommender system irrespective of sample size.

References

Adiyansjah, Gunawan, A. A., & Suhartono, D. (2019). Music Recommender System Based on Genre using Convolutional Recurrent Neural Networks. *Procedia Computer Science*, *157*, 99–109. https://doi.org/10.1016/j.procs.2019.08.146

Chen, Hung-Chen & Chen, Arbee. (2001). A Music Recommendation System Based on Music Data Grouping and User Interests.. 231-238. 10.1145/502585.502625

Chen, R., Lin, C., Liou, M., & Dewi, C. (2022). Mobile applications for music recommendation system combined with brainwave. 2022 6th International Conference on Medical and Health Informatics. https://doi.org/10.1145/3545729.3545741

Chung, J. and Kim, M. J. (2018). Music recommendation model by analysis of listener's musical preference factor of k-pop. Proceedings of the 2018 International Conference on Information Science and System. https://doi.org/10.1145/3209914.3209932

Ferraro, A., Serra, X., & Bauer, C. (2021). What is fair? exploring the artists' perspective on the fairness of music streaming platforms. Human-Computer Interaction – INTERACT 2021, 562-584. https://doi.org/10.1007/978-3-030-85616-8_33

Hu, B., Guo, M., & Zhang, H. (2009). A hybrid music recommendation system by m-lsa. 2009 International Conference on Computational Intelligence and Natural Computing. https://doi.org/10.1109/cinc.2009.74

Hu, Y., Koren Y., & Volinsky C. [2] (2008). Collaborative Filtering for Implicit Feedback Datasets. 2008 Eighth IEEE International Conference on Data Mining. https://ieeexplore.ieee.org/document/4781121

McKee, K. B. and Pardun, C. J. (1999). Reading the video: a qualitative study of religious images in music videos. Journal of Broadcasting &Amp; Electronic Media, 43(1), 110-122. https://doi.org/10.1080/08838159909364478

Zhang, T. and Liu, S. (2022). Hybrid music recommendation algorithm based on music gene and improved knowledge graph. Security and Communication Networks, 2022, 1-11. https://doi.org/10.1155/2022/5889724

Appendix

A. Data Preprocessing

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
import numpy as np
import pandas as pd
```

```
participant = pd.read_csv('/content/gdrive/MyDrive/Group 3/Data/K-pop songs Dataset/Data 1 - raw-participant.csv',  encoding='cp949')
```

```
participant = participant[participant['song_country'] == 'KR']
participant
```

	rater_nationality	participant_id	stimID	repeated_trial	song_country	song_year	singer_gender	preference_5pt	familiarity_4pt	calm	tense	cheerful	sad	danceable	love	dreamy	electronic	energy
0	American	participant_US1	KR-2014-block11	f	KR	2014	m	4	1	1	1	4	1	4	4	4	4	4
1	American	participant_US1	KR-2013-block11	f	KR	2013	f	1	1	2	1	4	1	4	4	2	3	4
5	American	participant_US1	KR-2012-block11	f	KR	2012	f	5	1	2	1	2	2	3	4	4	2	3
7	American	participant_US3	KR-2018-block11	f	KR	2018	m	2	1	2	2	1	3	1	2	2	1	2
10	American	participant_US3	KR-2015-block11	f	KR	2015	m	3	1	1	1	3	1	2	2	2	3	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11490	Brazilian	participant_BR6	KR-2013-block5	f	KR	2013	m	2	1	2	1	2	1	2	1	1	2	2
11491	Brazilian	participant_BR6	KR-2015-block5	f	KR	2015	f	3	1	2	1	2	1	3	3	2	2	2
11492	Brazilian	participant_BR6	KR-2011-block5	f	KR	2011	f	1	1	2	3	1	3	1	1	1	1	1
11494	Brazilian	participant_BR6	KR-2017-block5	f	KR	2017	f	3	1	2	1	2	1	2	1	2	2	2
11499	Brazilian	participant_BR6	KR-2019-block5	f	KR	2019	f	2	1	2	1	1	1	2	1	1	2	2

3845 rows × 18 columns

```
KR_song = pd.read_csv('/content/gdrive/MyDrive/Group 3/Data/K-pop songs Dataset/Data 2 - song_chart.csv',  encoding = 'utf-8')
```

```
KR_song
```

	song_origin	chart_source	year	stimID	song_query	artist_query	song_result_spotify	artist_result_spotify	spotify_trackid	preview.url	...	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	dura
0	KOR	Gaon Chart	2010	KR-2010-block1	Here I Am	포맨 (4men), 美	Here I am	4MEN	2HeHipLSY7SWKLjsZdUbjQ	https://p.scdn.co/mp3-preview/8e5d94375aa154f3...	...	-7.834	1	0.0335	0.8540	0.000000	0.1030	0.101	129.552	
1	KOR	Gaon Chart	2010	KR-2010-block2	밤하늘의 별을..	양정승 (kiroy Y), Kcm & No Noo	밤하늘의 별을..	Kiroy Y	3zNyjuCpWig3iYWs9PgbDM	https://p.scdn.co/mp3-preview/0a34821c26bbafb2...	...	-4.578	1	0.0609	0.2410	0.000001	0.1160	0.406	153.807	
2	KOR	Gaon Chart	2010	KR-2010-block3	Y	엠블랙 (mblaq)	Y	MBLAQ	3xVGwwax3gAPAkJewmB2eZ	https://p.scdn.co/mp3-preview/c9e9462cf7b3116...	...	-4.828	0	0.1420	0.0298	0.000004	0.0967	0.875	122.952	
3	KOR	Gaon Chart	2010	KR-2010-block4	사랑한다는 말	남규리	사랑한다는 말	Nam Gyu ri	1o68HIqheZWvKk4m9DnCK0	https://p.scdn.co/mp3-preview/e0a64b47cc65954b...	...	-7.516	1	0.0372	0.8130	0.000000	0.1110	0.172	134.063	
4	KOR	Gaon Chart	2010	KR-2010-block5	사랑, 우정 그리고... (with No Noo & 양정승)	제이비 (jb)	사랑, 우정 그리고...	JB	2ovX90giSOm2ouAIhtV3gB	https://p.scdn.co/mp3-preview/8385531176e549f53...	...	-6.121	1	0.0336	0.8170	0.000000	0.1090	0.499	171.757	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
355	BRA	Crowley	2019	BR-2019-block8	Saudade Infinita	Roberta Miranda	Saudade Infinita	Roberta Miranda	6J2ySKkqPkB0oB2laqYO1e	https://p.scdn.co/mp3-preview/e96bd3f00838fbf2...	...	-6.369	1	0.0274	0.2240	0.000000	0.1250	0.582	105.960	
356	BRA	Crowley	2019	BR-2019-block9	Combate	Um44k	Combate	UM44K	2gV8yu7sEnewmsVa4CnbOG	https://p.scdn.co/mp3-preview/76335d9d17acdd5e...	...	-6.116	1	0.1450	0.3150	0.000000	0.0974	0.831	88.040	
357	BRA	Crowley	2019	BR-2019-block10	Salvou Meu Dia	Mc Kevinho Part.Gusttavo Lima	Salvou meu dia (Participação especial de Gustt...	MC Kevinho, Gusttavo Lima	7CBlmG41isOs2r7lQU7CCy	https://p.scdn.co/mp3-preview/cbcc7edf512a1508...	...	-2.688	0	0.0807	0.4090	0.000000	0.3260	0.830	162.077	
358	BRA	Crowley	2019	BR-2019-block11	500 Metros	Paula Mattos Part.Lucas Lucco	500 metros (Participação especial de Lucas Lucco)	Paula Mattos, Lucas Lucco	2mo6GXEG5iwAA8CfBViGAf	https://p.scdn.co/mp3-preview/55f5b6b3309ab5ec...	...	-3.862	1	0.0510	0.6190	0.000000	0.1060	0.686	123.872	
359	BRA	Crowley	2019	BR-2019-block12	Toquin Do Violao	Sandro Cesar	Toquin do Violão	Sandro César	1qrjjgSAnpuw8Pf2NTr5eN	https://p.scdn.co/mp3-preview/f9c44e7fd9407fa1...	...	-3.283	1	0.0699	0.3630	0.000000	0.0928	0.735	147.880	

360 rows × 24 columns

```
song_pref = pd.merge(participant, KR_song, how = 'left', on = 'stimID')
```

```
song_pref.columns
```

```
Index(['rater_nationality', 'participant_id', 'stimID', 'repeated_trial',
       'song_country', 'song_year', 'singer_gender', 'preference_5pt',
       'familiarity_4pt', 'calm', 'tense', 'cheerful', 'sad', 'danceable',
       'love', 'dreamy', 'electronic', 'energy_x', 'song_origin',
       'chart_source', 'year', 'song_query', 'artist_query',
       'song_result_spotify', 'artist_result_spotify', 'spotify_trackid',
       'preview.url', 'ISRC', 'danceability', 'energy_y', 'key', 'loudness',
       'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness',
       'valence', 'tempo', 'duration_ms', 'time_signature'],
      dtype='object')
```

```
ratings = song_pref[["participant_id", "song_query", "preference_5pt"]]
ratings
```

	participant_id	song_query	preference_5pt
0	participant_US1	Thunder	4
1	participant_US1	그녀의 사무실	1
2	participant_US1	사랑아	5
3	participant_US3	영원히	2
4	participant_US3	너를 그리다	3
...	...	...	...
3840	participant_BR6	만약에 말야 (전우성 Solo)	2
3841	participant_BR6	Wifey (feat. Mc몽)	3
3842	participant_BR6	Break Up	1
3843	participant_BR6	낡은 그리움	3
3844	participant_BR6	Flash	2

3845 rows × 3 columns

```
ratings.to_csv('/content/gdrive/MyDrive/Group 3/Data/K-pop songs Dataset/Ratings.csv', index = None, header=True )
```

B. Data Analysis

```
# Divide dataset into train and test, convert x_train into pivot table (user-item matrix)

from sklearn.model_selection import train_test_split
x = ratings.copy()
y = ratings['participant_id']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=12)
rating_matrix = x_train.pivot_table(index='participant_id', columns='song_query', values='preference_5pt', aggfunc='mean')
```

```
# RMSE Formula
def RMSE(y_true, y_pred):
    return np.sqrt(np.mean((np.array(y_true) - np.array(y_pred))**2))

# Accuracy
def score(model, neighbor_size=0):
    id_pairs = zip(x_test['participant_id'], x_test['song_query'])
    y_pred = np.array([model(user, song, neighbor_size) for (user, song) in id_pairs])
    y_true = np.array(x_test['preference_5pt'])
    return RMSE(y_true, y_pred)
```

```
# Average ratings of all users
rating_mean = rating_matrix.mean(axis=1)

# KNN + User Bias

def ubcf_bias_knn(user_id, song_id, neighbor_size=0):
    # Average rating of user of interest
    user_mean = rating_mean[user_id]
    if song_id in rating_matrix:
        # Similarity between user of interest and other users
        sim_scores = user_similarity[user_id]
        # Ratings for the song in question
        song_ratings = rating_matrix[song_id]
        # Average ratings of all users
        others_mean = rating_mean
        # Erase users who did not rate the song in question
        none_rating_idx = song_ratings[song_ratings.isnull()].index  # Users who didn't rate
        song_ratings = song_ratings.drop(none_rating_idx) # drop them from song ratings
        sim_scores = sim_scores.drop(none_rating_idx)  # drop from similarity
        others_mean = others_mean.drop(none_rating_idx) # drop from average ratings

        if neighbor_size == 0:                # If neighbor size = 0
            if len(song_ratings) >= 2:       # Only if at least 2 people rated the song
                # Predicted deviation
                song_ratings = song_ratings - others_mean  # deviation
                prediction = np.dot(sim_scores, song_ratings) / sim_scores.sum()
                # Predicted rating
                prediction = prediction + user_mean
            else:
                prediction = user_mean
```

```
        else:                                   # If there is neighbor size
            # smaller value between designated neighbor size and number of users who rated
            neighbor_size = min(neighbor_size, len(sim_scores))
            # convert to array
            sim_scores = np.array(sim_scores)
            song_ratings = np.array(song_ratings)
            others_mean = np.array(others_mean)
            # Sort the similarity in order and get their indices
            user_idx = np.argsort(sim_scores)
            # Similarity and Ratings by neighbor size
            sim_scores = sim_scores[user_idx][-neighbor_size:]
            song_ratings = song_ratings[user_idx][-neighbor_size:]
            # Mean values of neighbor size
            others_mean = others_mean[user_idx][-neighbor_size:]
            # Predicted ratings
            if len(song_ratings) >= 2:
                song_ratings = song_ratings - others_mean
                prediction = np.dot(sim_scores, song_ratings) / sim_scores.sum()

                prediction = prediction + user_mean
            else:
                prediction = user_mean
    else:
        prediction = user_mean
    return prediction
```

```
        item_ids.append(item)
    predictions = np.array(predictions)                    # Numpy array conversion
    item_ids = np.array(item_ids)
    prediction_item = dict(zip(predictions,item_ids))      # dictionary mapping ratings to the song
    recommendations = np.sort(predictions)[::-1][:n_items] # choose the highest predicted ratings
    recommend_item = []
    for rating in recommendations:
      recommend_item.append(prediction_item[rating])  # song names corresponding to the predictions
```

```
# Accuracy
score(ubcf_bias_knn, 35)
```

```
0.8223833672051327
```