**Spotify Recommendation System**

# 

**ANLY 502 - Final Project**

**Group Mountain Dew**

**Group Numbers:**

**Yanou Yang**

**Xinya Xu**

**Wenhao Jiang**

**Leilin Wang**

# **Executive Summary**

The recommendation system is widely used in online media service providers. Especially for Spotify, better the recommended songs, more likely the user will keep using Spotify. Since users cannot search for all songs manually, if Spotify can provide the recommendations of the songs that users would highly like, then there are higher chances of Spotify staying relevant and becoming successful. So our goal is to develop a recommender system for Spotify playlist. Given a set of user’s behavior features, a recommendation system should be able to generate a list of recommended tracks that meets a certain user’s preference.

Spotify has released a dataset called “Million Playlist Dataset” which would be used as a dataset to train our model. The main problem here is to identify the various ways to recommend the next song to the user or the playlist. There are two main methods in collaborative filtering. The first one is user rating various songs and then the recommended songs were provided to them resulting in highly favorable outputs from the model. But since, we cannot always ask a user to rate the songs but rather songs are liked or disliked in Spotify. Therefore, the second method to model the system was based on this idea. We got implicit feedback like views, clicks, likes, or shares in our dataset, so we transformed it into numbers to represent the strength in observations of user actions’ preference. The generated results were in accordance with the user’s choices as well as on the statistical viewpoints. More information about the project and the methods used are detailed in the given report.

# **Introduction**

Music recommender systems have recently exploded in popularity thanks to music streaming services like Spotify and Apple Music. By some accounts, almost half of all current music consumption is by the way of these services. While recommender systems have been around for quite some time and are very well researched, music recommender systems differ from their more common siblings in some important ways: the duration of the items is less (3-5 min for a song vs 90 minutes for a movie ), the items are consumed in sequence with multiple items consumed in a session, repeated recommendations have a different significance. Music Recommender Systems then require different approaches from traditional recommender systems.

This project’s goal is to develop a recommender system for a playlist, which would enable Spotify to seamlessly support their users in creating and expanding the playlists by making recommendations based on their choices and preferences. Furthermore, the recommender system does not require any rich and varied supply of music data, instead requiring only basic information as input such as the title of the playlist, the tracks currently in the playlist, and the user behavior with those tracks.

First, we need to decide on the plan to approach this dataset to train our model. There are two basic methods of music recommendation-metadata information retrieval and collaborative filtering.

Metadata information retrieval uses editorial information supplied by the creators, such as the title of the song, artist name, and lyrics to find some target songs. Though it is fast and accurate, the drawbacks are obvious. First of all, the user has to know about the editorial information for a particular music item. Secondly, it is also time-consuming to maintain increasing metadata. Moreover, the recommendation results are relatively poor, since it can only recommend music based on editorial metadata and none of the users’ information has been considered.

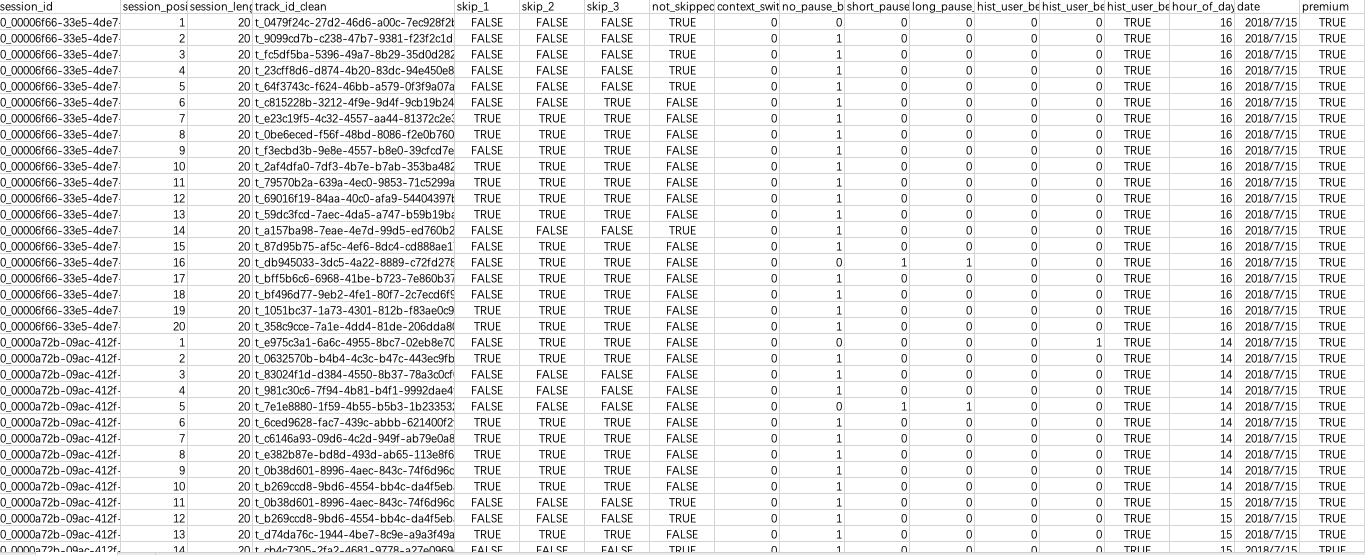
Collaborative filtering is able to recommend items via the choice of other similar users. As one of the most successful approaches in recommendation systems, it assumes that if user X and Y rate n items similarly or have similar behavior, they will rate or act on other items similarly. Instead of calculating the similarity between items, a set of ‘nearest neighbor’ users for each user whose past ratings have the strongest correlation are found. Therefore, scores for the unseen items are predicted based on a combination of the scores known from the nearest neighbors.

For this project, We will use neural collaborative filtering to build the playlist recommendation system. We also plan to do some exploratory analyses, like looking at numbers of unique tracks and users’ behavior distribution that helps further understanding in methodology. We may also do some playlist clustering and PCA, to find interesting patterns in these user-generated playlists.

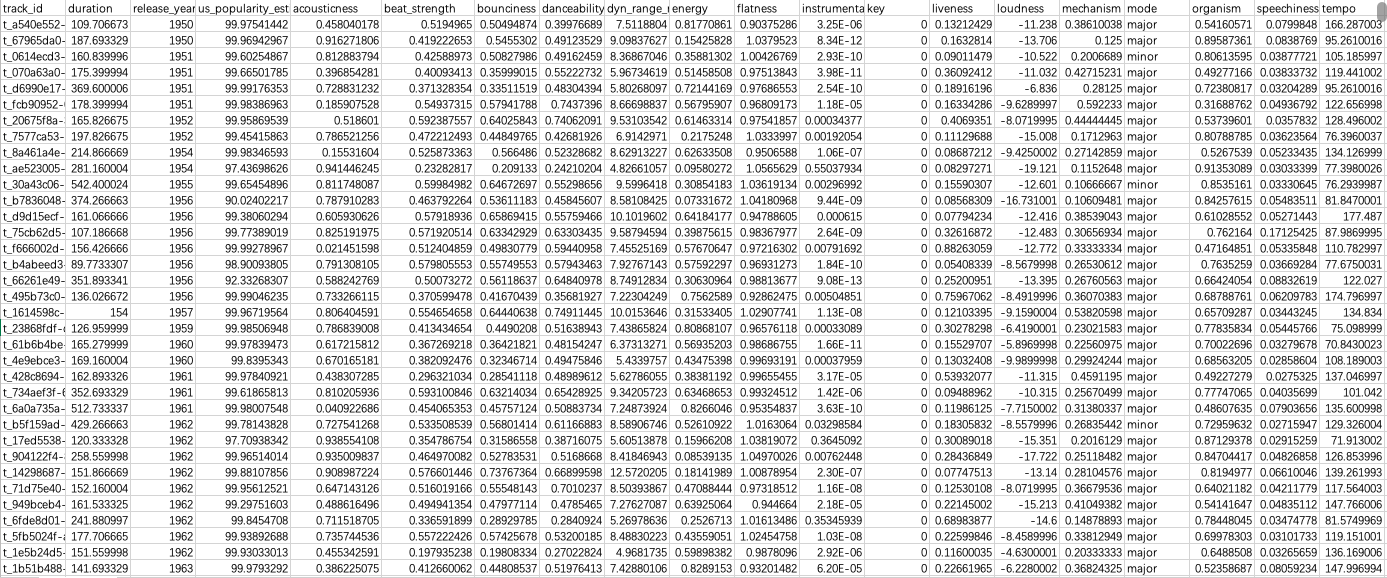
# **Data Exploration**

Before we build the machine learning model for this project, first we need to know what our data looks like, to understand what people like before we recommend anything to them.

The dataset we use consists of two parts. The first part is as what we introduced before, containing all the user action upon a single track for a session, including information of skipping, seeking back, etc. The other part of the data contains specific features of these tracks. The columns include the same ‘track\_id’, loudness, danceability, tempo, time signature, etc. All these variables are vectorized to numbers beforehand. The rows do not contain any missing value or NA. So the dataset is very clean, and ready to use. The pictures below show a short glance of the two parts of data.

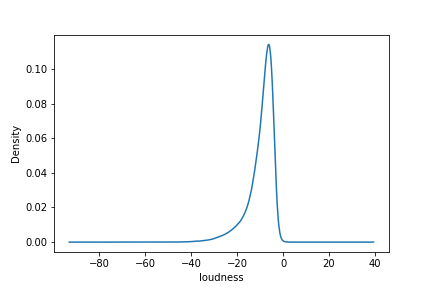


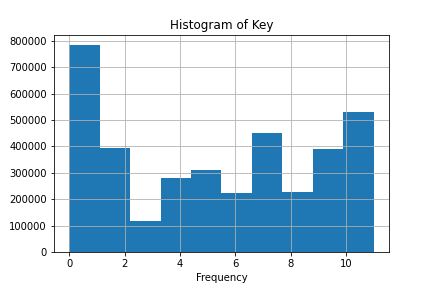
The Log sample: user action upon a single track

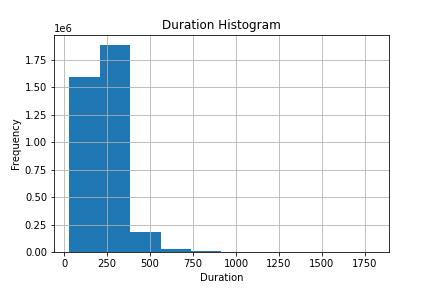


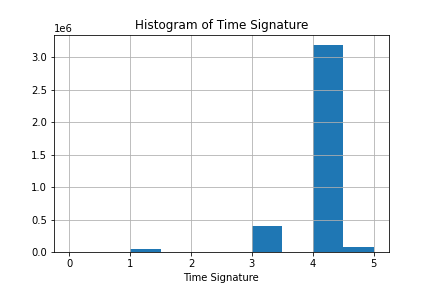
The Feature sample: the musical information of a single track

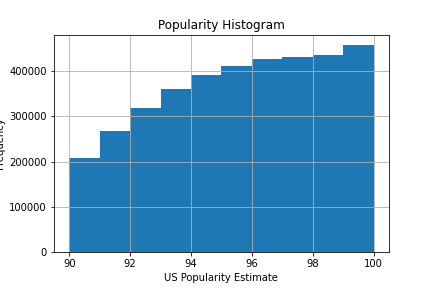
Since there are a total of 29 unique columns in the feature dataset, only a portion of them are chosen to be visualized. The variables we choose are: loudness, a vector of measurement of the audio magnitude; key, an integer representing the specific key for a certain track; duration, measurement of length of a certain track, in seconds; time signature, a integer representing the number of beats (pulse) in each measure; and popularity, a relative score measuring the popularity estimate of the US for a certain track.





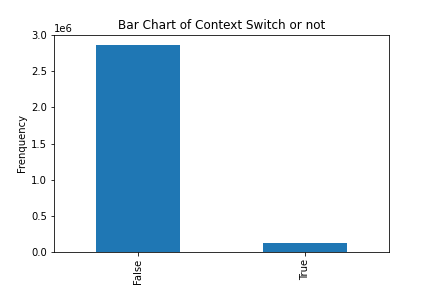


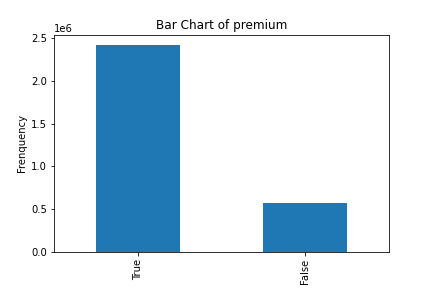


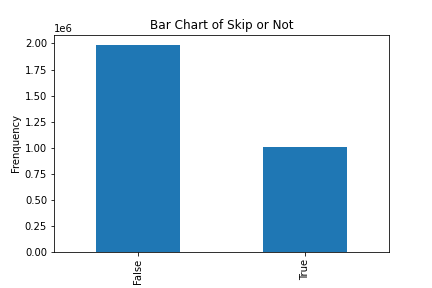


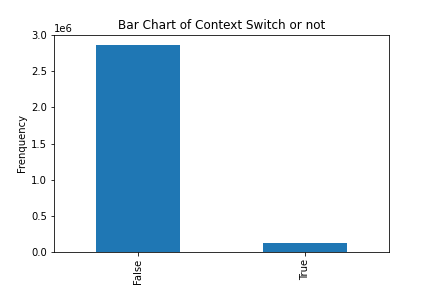
With a glance into the features of tracks as shown above in the graphs, we now have some idea of what this part of data looks like. All the five graphs show that the data is not balanced, mostly clustered in a small interval. And many other variables in the dataset show similar imbalanced patterns as these. Then we can look into details of the first parts of data: user implicit feedback.

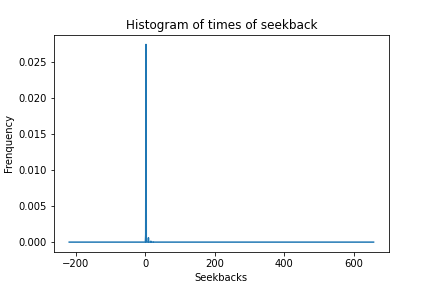
Having a look at the user action dataset, it appears to be a large data set, which is 2,072,002,577 rows with 21 columns. There are 3,604,454 unique tracks being played in the time slot between July 15, 2018 to September 18, 2018. Because of the extremely enormous size of the dataset, we planned to output some simple summary and diagrams for the columns we are interested in.





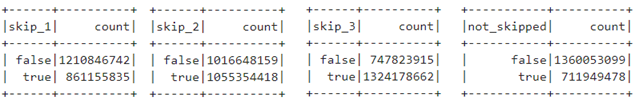






The five graphs above show how the user implicit feedbacks distribute. They give us some basic idea of how people respond to a single track in general. These variables are important to our estimate to a certain track.

Let’s dive deeper into the numeric summaries of some variables. Since user skip behavior is one of the most direct behaviours that are related to whether the user likes the song or not, we check the output for all four columns that correspond to skip behaviour, which are skip\_1, skip\_2, skip\_3 and not\_skipped.



For those four variables, skip\_1 is the boolean indicating if the track was only played very briefly, skip\_2 is the boolean indicating if the track was only played briefly skip\_3 is the boolean indicating if most of the track was played and not\_skipped is the boolean indicating that the track was played in its entirety. From the numeric value of each boolean, we explored that since adding all the trues for all four variables results in a number much larger than the total rows, those variables are not displayed in the pattern that as one of four booleans is true, other boolean variables are false. We did not understand what is the actual skipped behaviour when there are more than one true values in those four variables. So we made an assumption that we just consider each track to have the longest play time, meaning that if one track has skip\_1 and skip\_3 are both true and others are false, we just consider it as skip\_3 when we are rating the user's skipping behaviour.

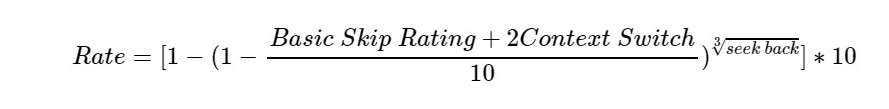
With such knowledge in mind, we could then consider how we can approach the data with better accuracy. The next section would illustrate in detail how we use these variables for our model.

# **Methdology**

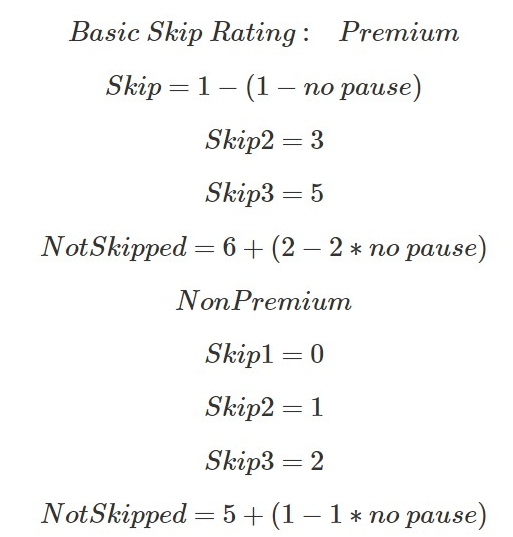
Recommendation systems predict what users would like in the future based on their behavior patterns. The most basic models for recommendation systems are collaborative filtering models.

Collaborative filtering models which are based on the assumption that people tend to like things similar to the historical things they like, and things liked by other people who have the same favour with them.

In this project, we used matrix factorization-based collaborative filtering, that treats the entries in the user-item matrix as explicit preferences given by the user to the item, for example, users giving rating to movies. So to the spotify dataset, firstly, an explicit rating to each movie should be defined. The rating formula is defined as follows.



The parameters are defined as belows:



The above formula is defined by analyzing user behaviour. For example, non premium users’ skip should be paid more attention, because non premium users only have three chances to skip songs in the Spotify system. Besides, if one user didn’t skip this song and didn’t pause during the track, it means this song might satisfy the user’s taste. The other user behaviour index is explained in the following table:

|  |  |
| --- | --- |
| Skip | Played briefly or entirely |
| Pause | Pause between previous track and this track |
| Seek Forward | Seek forward within track |
| Seek Backward | Seek backward within track |
| Context Switch | Change context |
| Premium User | Potential implications for skipping behavior |

After assigning each song a rate based on users’ behaviour, we have a user-track rating matrix. The next step is matrix factorization. Matrix factorization can be done by various methods and in this project we use a pyspark package, which uses orthogonal factorization(SVD). The idea behind matrix factorization is that attitudes or preferences of a user can be determined by a small number of hidden factors, which can be called embeddings. Intuitively, embeddings are low dimensional hidden factors for items and users.

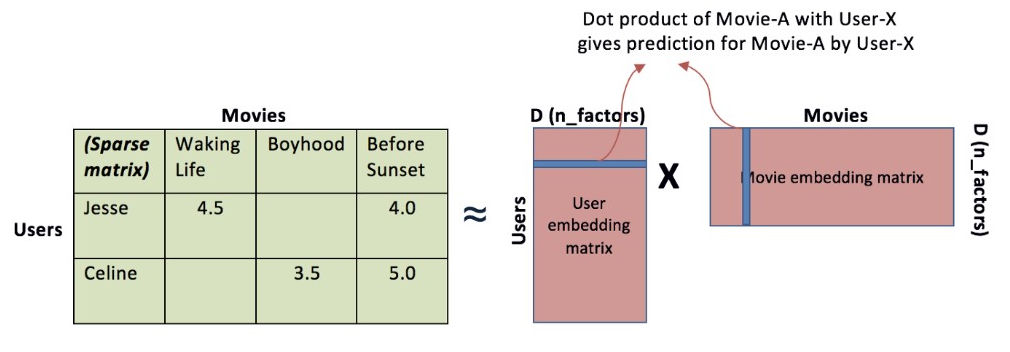


Fig: Visualization of matrix factorization

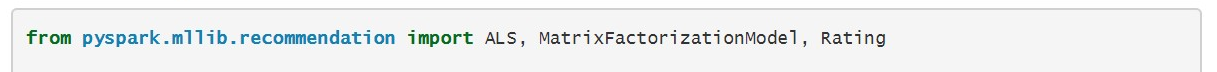
(Source: <https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0>)

Matrix factorization eventually measures how much a user is aligned with a set of latent features, and how much a track fits into this set of latent features. The advantage of it over standard nearest neighborhood is that even though two users haven’t rated any same tracks, it’s still possible to find the similarity between them if they share the similar underlying tastes, again laten features.

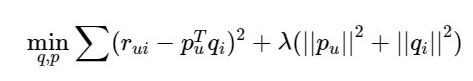
The predicted rating for specific users and items is the below formula. Item i is noted as a vector and user is noted as a vector such that the dot product of these two vectors is the predicted rating for the user on the item.



In our project, the way to find optimal and is using Alternative Least Square to minimize the cost of the errors, which together with the whole collaborative filtering algorithm in pyspark package.



The loss function in the algorithm is as belows,

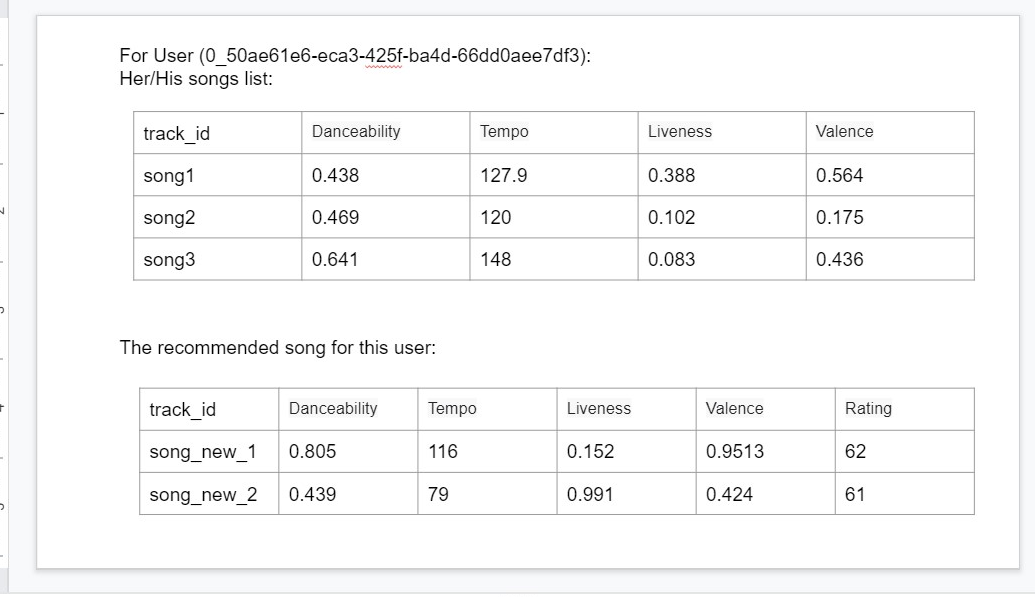


is the true ratings from the original user-item matrix.

The project hypothesis is that we assume the defined rating can truly represent users’ preference. If the predicted outcome coherents with user historical preference which can measure by similarity matrix and MSE, then we can conclude our model is a good recommendation system model.

# **Result and Conclusion**

The final result is somewhat within our expectation. The model would recommend new tracks to the user as expected. Below is one example of the results from our recommendation system. For this user, we see that the two recommended songs have some similarity to the user’s preference. The first new song has similar tempo and liveness, while the second new song has similar danceability and valence. Both have very close ratings.



Overall, our model would recommend new tracks to the user well within his/her preference. But admittedly, the other features of the two recommended songs do not meet the user’s preference very well. This is one of the drawbacks of our model. Or maybe this is due to that each user only has a small number of tracks for our model to learn from. The final evaluation of our recommendation is always done by the user himself/herself, for music preference is ultimately subjective, rather than some measurement of 29 vectors.

# **Future Work**

One thing we can do to improve the performance of our model is to optimize or redo the formula we used for calculating the rate of each track. For simplicity reasons, only 6 variables are used in the formula. However, due to data imbalance and low inclusiveness, this rate we calculated could be biased or inaccurate. By introducing more variables and reformulating the formula, more appropriate ratings would be calculated, thus improving recommendation accuracy. But this is rather a mathematical problem which could take more effort and creativity to achieve.

Another way to improve our project is to introduce new models for commendation. For now, collaborative filtering is the best and most approachable model among our research. There are some other possible models that could do a better job on a recommendation system. Besides, within collaborative filtering, there are different approaches, including KNN, SVD, Matrix Factorization, Deep Learning with neural networks, etc. Thus, by using more models, we could do horizontal comparison to find the best for this recommendation system.

# **References**

The file “Small\_version.ipynb” is the final code we used to apply the model to the data.

[1]<https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26>

[2]<https://spark.apache.org/docs/latest/mllib-collaborative-filtering.html>

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

# **Division of labor**

Data Storage, Data Visualization and Model Pipeline Build and Others:

Xinya Xu, Wenhao Jiang, Leilin Wang and Yanou Yang