# KKBOX Music Recommendation System

Zack Liu, Xiaotong Ding, Yaqi Lin, Jixian Fu, Haoran Shi
Department of Statistics
University of California, Davis
1 Shields Ave, Davis, CA 95616
{zheliu, xtding, yqilin, mrfu, hrshi}@ucdavis.edu

## Abstract

*In this report, we presented the works we have done in performing feature selection using correlation matrix and feature importance based on lightGBM on the KKBox Music recommendation competition. Besides, we built a logistic regression model for content-based filtering. We compared the RMSE and AUC-ROC for these models; analyzed the intrinsic differences among these music recommendation algorithms and the significance of parameters involved. Moreover, we applied collaborative filtering methods: Matrix factorization based on Funk SVD and Non-negative matrix factorization to create our music recommendation model and make predictions based on the existing features from the KKBox datasets*.

## I. Introduction

CDs have been replaced by digital music records and musical gatekeepers have been replaced with personalizing algorithms and unlimited streaming services. All our online transactions leave digital traces that can be transformed into individual and group behavior. The transformed data enables us to measure the previously unmeasurable. Work, transacted through online sources, is recorded at incredibly fine levels of detail.

Our data come from KKBOX Kaggle competition, where KKBOX provides a training data set consisting of information of the first observable listening event for unique user-sing pair within a specific time duration. Metadata of each unique user and song pair is also provided. There are five datasets included in the source with total size 345MB. The datasets provide information about the user, such as age, gender, and information about the songs, such as genre, song length, and artist. The original competition asks to predict a binary variable *target*. If *target = 1*, then it indicates that the user repeats the same songs within one month period, otherwise *target = 0*. Different from this traditional definition, we treated the variable ***target*** as the **user preference variable**, where **1** indicates the **user likes the song** and **0** indicates the **user dislikes the song** in our Music Recommendation System.

We would go through different models and predictors to explore their performances and differences. We aim to detect user's recurring listening events on different songs that the user listened and to predict the users' preferences based on the given dataset using content-based and collaborative filtering methods.

# II. Exploration

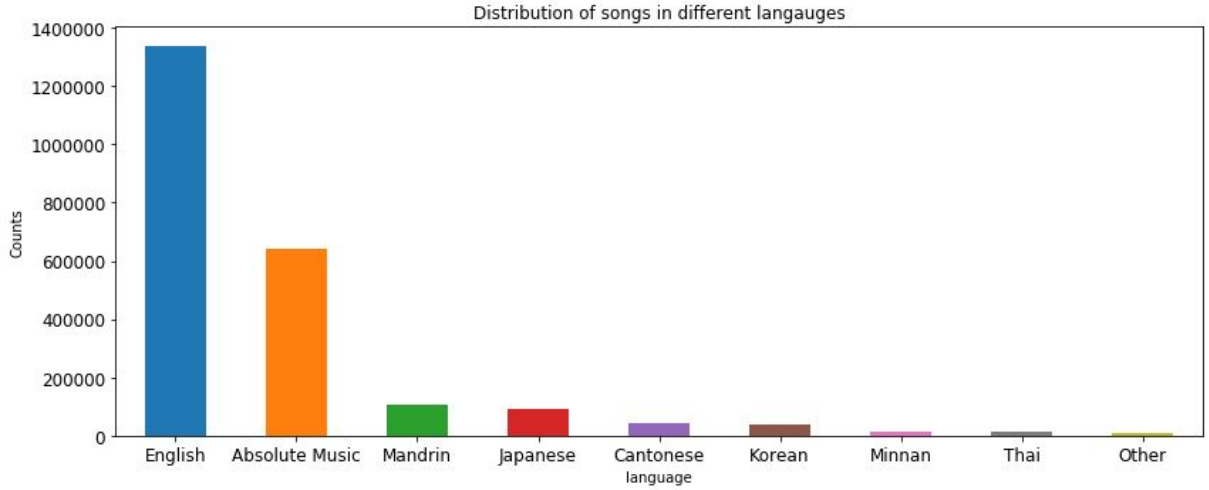
## 2.1 User Features:

Among all users, 7096 ( 20.6%) are female, 7405 ( 21.5%) are male, and 19902 ( 57.8%) are others. The percentage/numbers of female and male users are similar; while over half of the users preferred not to reveal their gender status to KKBOX.

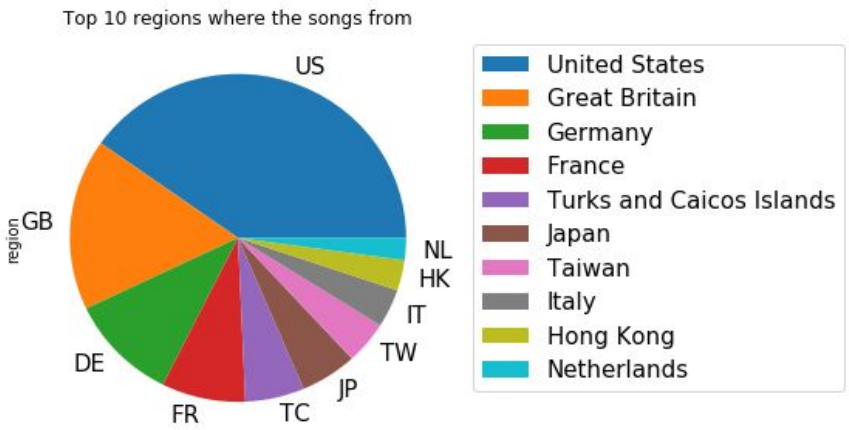
We classified users with less than a month membership length to be 'trial user' of KKBOX; otherwise, be 'paid members'. We found that 26066 (75.8%) of the users are 'members' while 8337 (24.2%) are on their trial.

## 2.2 Song Features:

There are 10 different songs' languages provided by KKBOX.




From the bar chart of songs for each language above, we classified languages based on the least and second least counts to 'others' due to small numbers and unclear identifications. The chart demonstrates that English songs have most records in their stock ( 58%) and they have 28% of absolute music. It is significantly noticeable that the rest languages of music records come from Asia. This result matches reality. KKBOX is an Asia company headquartered in Taiwan, so they may be more accessible to Asian songs' resources.



Other than songs' languages, we can find songs' production regions from the International Standard Recording Code(ISRC). ISRC is an international code for identifying sound recordings, and the first two values of the ISRC represent the region where the ISRC code was issued. Based on this information, we plotted the pie chart above to show the top 10 regions where song records provided by KKBOX were produced. We can see that most of the songs were produced by the United States and Great Britain.

# III. Music Recommendation Modeling

## 3.1 Target:

By checking the distributions of 0 and 1 of the *target* variable, we see that 0 and 1 have about the same number of counts. Therefore, there doesn't exist imbalanced classification problem in our dataset.

## 3.2 Content-based Algorithm

### 3.2.1 LightGBM

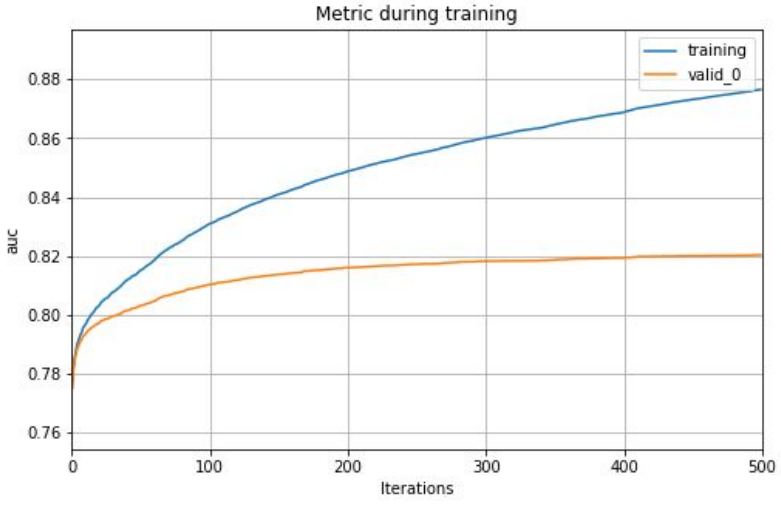
Since the variable *target* that we need to estimate is a binary variable. We can first view our problem as a traditional binary classification problem. LightBGM is a popular gradient boosting framework that uses tree-based learning algorithms. Comparing to other gradient boosting framework like XGBoost, Light GBM is memory and CPU efficient.

Before putting our data into the algorithm, we generated two new features based on the original data. (*song_repeat_rate* and *user_repeat_rate)*

*song_repeat_rate*: represents the proportion that the song had been like in the past
*user_repeat_rate*: represents the proportion of songs that the user like in the past

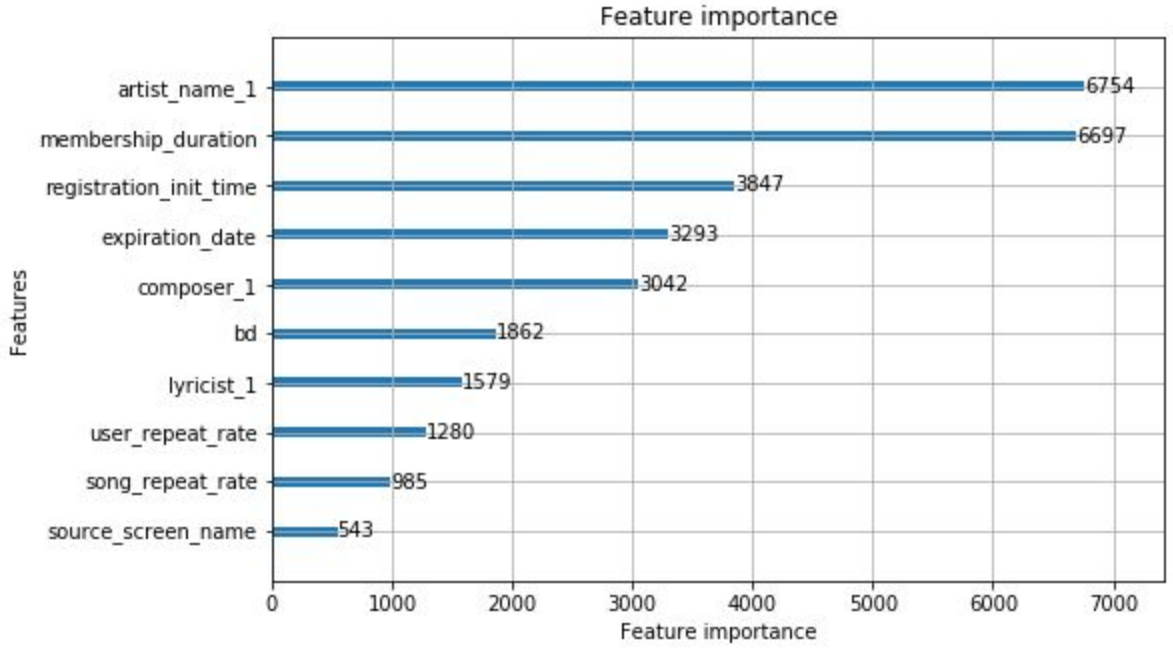
**Training:**

By using 80% of the data as a training set and 20% of the data as a validation set, We trained our LightGBM model in 500 boosts.



From the plot above, we can see the AUC score of the model on the validation set is improving very slowly after about 350 boosts. The AUC score we achieved on the validation set is about 80%

**Result:**
For further analysis, we selected features which contribute most to the model. We plotted the feature importance for the top 10 features in the following graph.



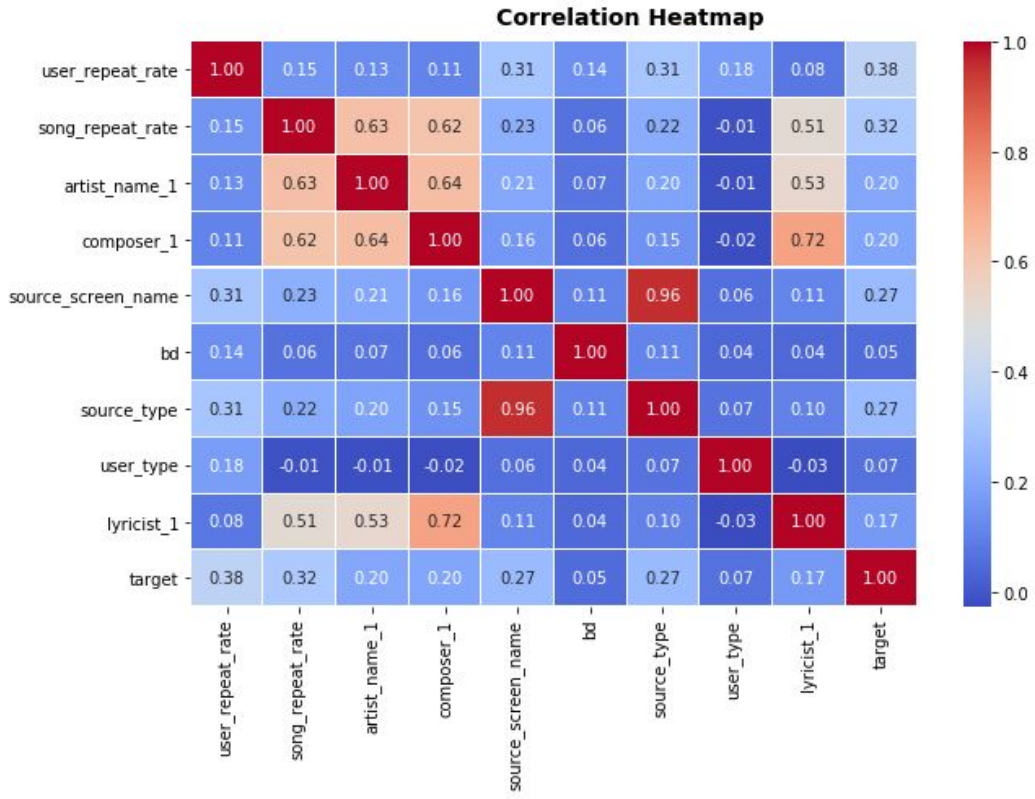

From the plot above, we can see that artist_name_1 is the most important feature. Contradicting our previous assumption, *song_repeat_rate and song_repeat_rate* don't contribute a lot of importance. One explanation is that the rates are highly correlated with other variables. Next, we selected a few features from the top 10 important features to build a logistic binary classifier.

## 3.2.2 Logistic Binary Classifier

Logistic regression is popular for classification because it is easy to implement and to understand. However, it is troublesome when the data is mostly categorial. The important features we generated from the LightGBM model are mostly categorical. Each of the features has many factors. For example, the variable artist_name_1 is the name of the artist of a song. In our data, we have 39388 unique artists. Therefore, converting the categorical variable into a dummy variable is not suitable in our case. Instead, we converted each categorical variable into a numerical variable by using a mapping we defined. For each factor of the categorical features, we calculated a *repeat rate*, which is similar to *user_repeat_rate* and *song_repeat_rate*. The

*repeat rate* of a factor is the proportion of the observation that contains the factor and has *target = 1*. We used the *repeat rate* to map each categorical variable into a numerical variable.

After the mapping, we calculated the correlation matrix for the variable and is visualized as the following:




From the correlation heatmap, we observed that *lyricist_1*, *compser_1,* and *artist_name_1* are highly correlated. One explanation is that the lyricist, composer, and artist of a song are the same person in many cases.

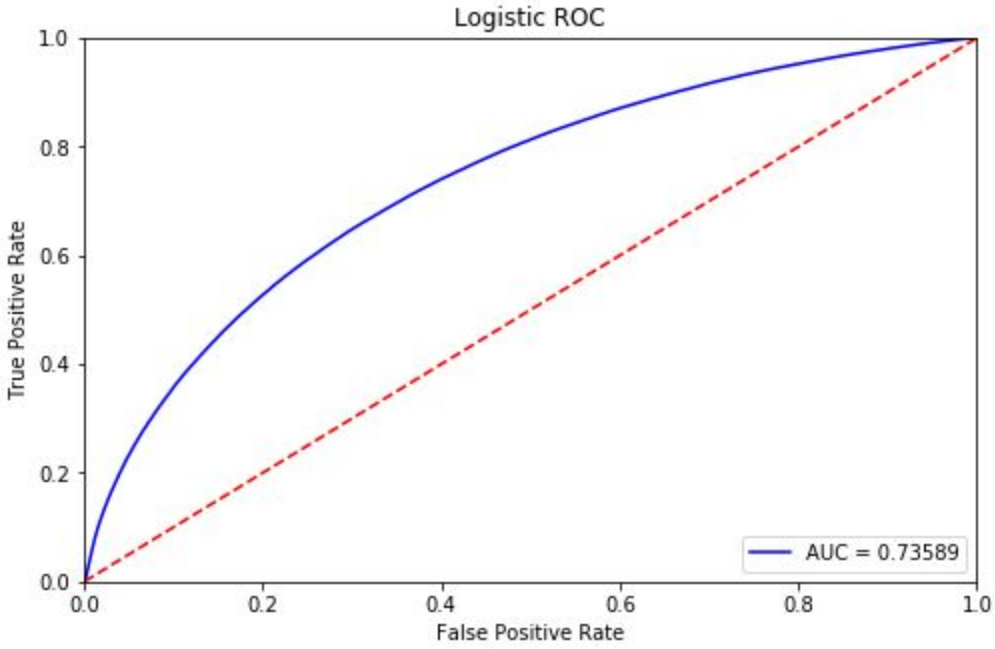
The feature vector we fed into the logical regression model is as the following:

$$[rr(artist\_name\_1), rr(compser\_1), ..., user\_repeat\_rate]$$

where *rr* represents the repeat rate.

**Training and Result:**

We trained our logistic classifier using a SAG solver and obtained the following result:




Comparing to the LightGBM, our logistic classifier didn't show an improvement in the performance. However, the logistic classifier requires less computing power and requires fewer features.

## 3.3 Collaborative Filtering

Collaborative Filtering is another type of methods for building a recommendation system. Collaborative Filtering recommends items to users based on similar users' preferences. It only requires users' historical preference to make a decent recommendation. Because it is based on past user preferences, the core assumption here is that the users who had similar preferences in the past tend to have similar preferences in the future. One of the main drawbacks of this method is that it performs poorly on cold-start problems, which is when a new item or new user with no historical preference is added to the recommendation system. Generally, for this type of recommendation system, only three variables are needed, user_id, item_id, and rating. In our dataset, the corresponding variables are *msno, song_id*, and *target*. The core of CF is the user-item matrix. In the user-item matrix, each row represents a user and each column represents

an item, and each entry is a rating. There are 30755 unique user IDs and 359914 unique song IDs in the cleaned training dataset.

### 3.3.1 Matrix Factorization

Matrix Factorization is a set of collaborative filtering algorithms. The basic idea of Matrix Factorization is to decompose the user-item matrix into a product of two lower dimensionality matrices. One matrix represents the user matrix where rows are users, and columns are latent features. The other matrix is the item matrix where columns are items and rows are latent features. The latent features can be seen as implicit information about the users and the items, and they are not explicitly observable.

#### 3.3.1.1 Funk SVD

Funk SVD introduced by Simon Funk is a famous Matrix Factorization algorithm. Despite its name, Funk SVD, it isn't a traditional mathematical Singular Value Decomposition of a matrix. A user-item matrix, *R*, is a sparse matrix where most values are missing, so a traditional Single Value Decomposition on *R* is undefined. In Funk SVD, we decompose *R* into *P* and *Q* by solving the following minimization problem.

$$\min_{p_u, q_i} \sum_{r_{ui} \in R} (r_{ui} - p_u \cdot q_i)^2$$

In the Funk SVD, the main idea is that we only use known value $r_{ui}$ and simply ignore missing values. In practice, a more robust version of the Funk SVD is used and the formulation is shown below.

$$\min_{p_u, q_i, b_u, b_i} \sum_{r_{ui} \in R} (r_{ui} - \mu - b_u - b_i - p_u \cdot q_i)^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2)$$

$u$ : the global mean of the ratings

$b_u, b_i$: mean ratings of users and mean ratings of songs

$\lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2)$: added regularization term for preventing overfitting

$b_u$ and $b_i$ are bias terms. The bias of users corresponds to the tendency of the users to give a good rating or bad rating. The bias of items corresponds to how well the items are rated compared to the average.

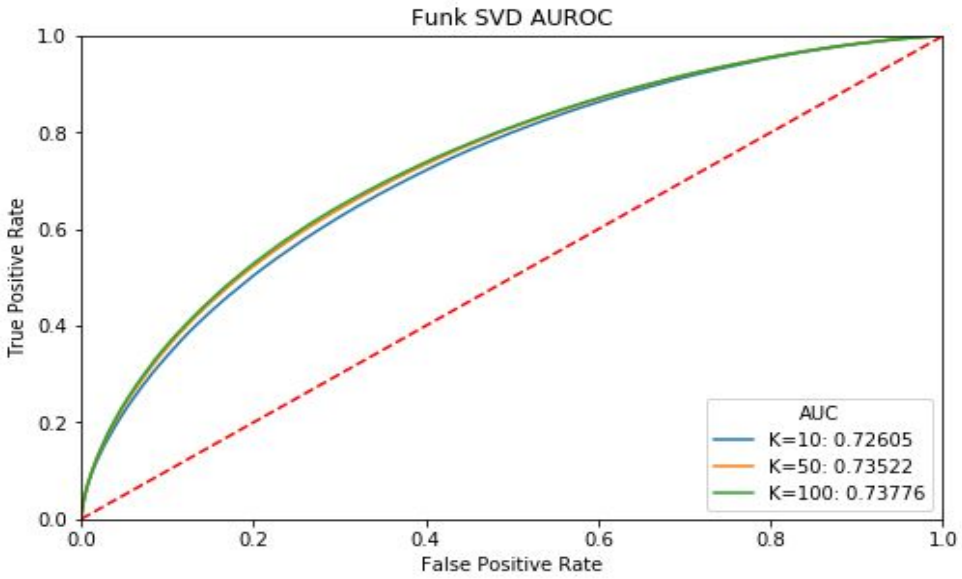
The general approach to solve the minimization problem above is using Stochastic Gradient Descent. In SGD, we take derivatives of the loss function with respect to each variable in the model and update the variables on each sample at a time. The variables above are updated by the following:

$$b_u \leftarrow b_u + \gamma(e_u i - \lambda b_u)$$
$$b_i \leftarrow b_i + \gamma(e_u i - \lambda b_i)$$
$$p_u \leftarrow p_u + \gamma(e_u i - q_i - \lambda p_u)$$
$$q_i \leftarrow q_i + \gamma(e_u i - p_u - \lambda q_i)$$

where $e_{ui} = r_{ui} - \tilde{r_{ui}}$

**Performance of Funk SVD on our data with different numbers of latent features:**

| **#'s latent features** | **K =1** | **K = 30** | **K = 100** |
|---|---|---|---|
| **RMSE** | **0.9123** | **0.8923** | **0.8951** |




As shown in the table and the AUC-ROC plot above, the best number of latent features is 30. As the number of latent features increases, the more computing power requires. However, the performance of the model does not necessarily increase. The latent features in the dataset are a representation of the users' personal preferences, songs' genre, etc. By Increasing the number of latent features, the model includes more information about the users and the songs. When the number is too high, the model will start overfitting and downgrade the performance. Therefore, the performance goes down.

### 3.3.1.2 Non-Negative Matrix Factorization:

Non-Negative Matrix Factorization factorizes the user-item matrix *R* into to matrices *W* and *H*, written as:

$$R_{m\cdot n} \approx W_{m\cdot k} \cdot H_{k\cdot n}$$

Where $W_{m\cdot k}$ and $H_{k\cdot n}$ have no negative elements.

Here, *k* is called the rank of approximation of *R*. We achieved dimension reduction with the help of *k*. Then, *W* is the basic matrix representing the features in the model. *H* is the weight matrix represents the associated importance of each feature. It is worthy to mention that column vectors of *R* can be written as a linear combination of columns of *W*, written as: $r_i = W \cdot h_i$

Here, we can see each column of R is equal to the sum of each column of W after being weighted by its corresponding row in Hi. Consequently, there is an intuition for NMF: it can combine parts to form a whole or learn part-based representation. Indeed, NMF achieved that by constructing sparse bases and sparse weightings. The idea behind the method is to best proximate R by W and H. Mathematically, we have to find the best W and H to minimize the cost function:

$$||R - WH||^2$$

There are mainly two algorithms to solve the problem above. The first one is the Coordinate Descent algorithm, in which we fix *W* and optimize *H* and then fix *H* to optimize *W* until the tolerance level is satisfied. However, in practice, that is not quite practical that the computation time is too long.

Another Algorithm is Multiplicative Update Algorithm. The basic idea behind this algorithm is that we can approach a local minimum step by step, updating *W* and *H* at the same time. To achieve that, we need three steps. First, we assign the *W* and *H* with random non-negative value. we constantly update the values in *H* and *W* to achieve a lower cost. The formulation is shown below:

$$w_{uk} \leftarrow w_{uk} \cdot \frac{\sum_{i \in I_u} h_{ik} \cdot r_{ui}}{\sum_{i \in I_u} h_{ik} \cdot \tilde{r_{ui}} + \lambda_i |I_u| w_{uk}}$$

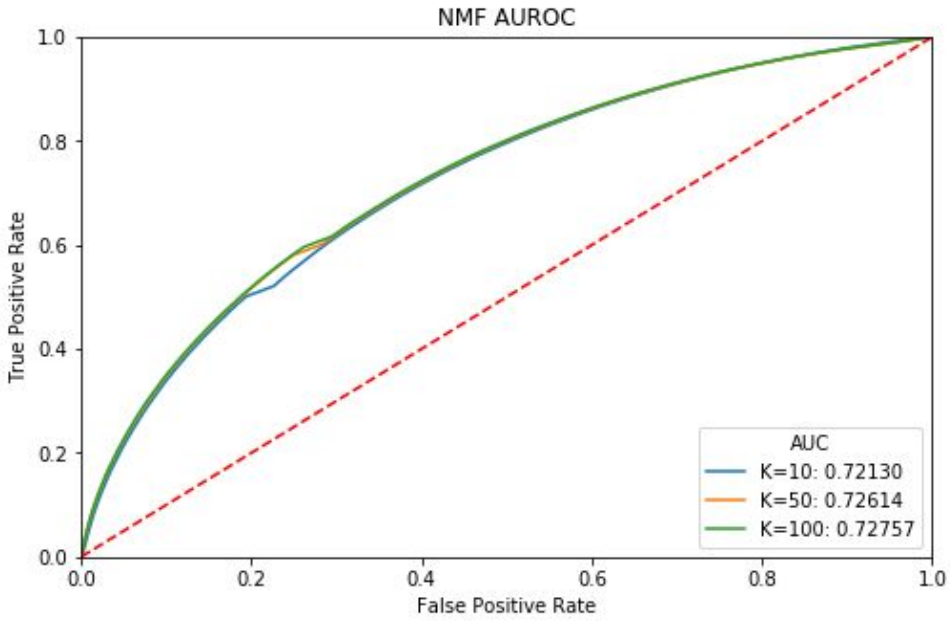
$$h_{uk} \leftarrow h_{uk} \cdot \frac{\sum_{i \in U_i} w_{ik} \cdot r_{ui}}{\sum_{i \in I_u} w_{ik} \cdot \tilde{r_{ui}} + \lambda_i |U_u| h_{uk}}$$

Where $I_u$ are set of all items rated by the user, and $U_i$ are set of all users that have rated item *i*.

For our data, we applied the second algorithm to estimate $\tilde{r_i}$.

**Performance of NNF on our data with different numbers of rank:**

| **#'s rank** | **K = 1** | **K = 30** | **K = 100** |
|---|---|---|---|
| **RMSE** | **0.9383** | **0.9204** | **0.9146** |




The training result is not as good as the FunK SVD, we obtained generally higher RMSEs and lower AUCs by NNF model. Besides, when comparing the number of latent features used, the SVD method has a fewer amount of features used, which therefore leads to a less complicated model and a faster computing speed. We think the reason why the NNF method performs not that well is its non-negative property. As we are aiming at getting close to the actual rating, then it is intuitively that some users may like one genre of songs while not like other genres. Therefore, these users in fact negatively rated other genres of songs by not listening to it again. However, because of the non-negative property, the model can only present how positive may a factor contributes to the rating. In other words, the negative attitude of users towards a song has been ignored. Therefore, the NNF model is not that desirable for our data.

## IV. Conclusion

From all the algorithms we had explored, content-based algorithms outperformed collaborative algorithms. Overall, LightGBM performed the best on the KKBOX music Dataset. Collaborative filtering relies heavily on the user's historical preferences. Without knowing a new user's past preferences or past rating for a new song, collaborative filtering cannot produce an accurate result. On the other hand, content-based algorithms use explicit features about the users and songs. When predicting a new user or a new song, these features will still be available. Therefore, content-based algorithms are more suitable for cold-start problems.

In the end, we used Matrix Factorization based on Funk SVD to make recommendations for a randomly selected user in the music dataset. The following is the **songs' recommendations for**

**the selected user** who used to listen to music in different languages and like music has genre_id: 465 and 458:

| | song_length | genre_ids | artist_name | composer | lyricist | language | name |
|---|---|---|---|---|---|---|---|
| OjCMT6Asis1m16Ez958bmrAsVjzfkO838ja+kU5Cvgo= | 301662 | 465 | 孫楠 | Bao Xiao-Song | Yi Jia Yang | Mandrin | 燃燒 (Flame Up) |
| b+OlR6qF0gzG++m+sYrOnf7bJF6WmXv51xGLBpmugxg= | 288391 | 458 | 陳奕迅 (Eason Chan) | Jun Jie Lin | NaN | Mandrin | 你給我聽好 |
| W+akwEPQhC1OugUaQlo0DR9xVs49UbJDZbiGvgN6ZyY= | 192470 | 921 | EXO CHANYEOL & PUNCH | Lee Seungju\| Rocoberry | Ji Hoon | Korean | Stay With Me |
| SVu/hfWmVp1NFP6aOXb9byfcxW15d7Nrmk1ablZg/Ww= | 288368 | 465 | 盧廣仲 (Crowd Lu) | NaN | NaN | Mandrin | 寂寞考 |
| QTBzHADX6rAiQoJRCoO7rxPmGOVvKH3xYOgmjrgYy+8= | 262164 | 458 | AMIT | Zheng Qun Hua\| Evan Lai | NaN | Mandrin | 怪胎秀 |
| Y87wsslvAzfpJA1z7WGawIIQl7mNFDWzmKS/FILtLA8= | 193410 | 465 | 蔡依林 (Jolin Tsai) | Ni Zi Gang | Lee Ge Di | Mandrin | Play我呸 |
| aDiegcdOzSszNClhMEEb3cLoZPYKBhWeyzaoB1+W1ng= | 238027 | 451 | 陳奕迅 (Eason Chan) | Da Yin Zhong | NaN | Cantonese | 於心有愧 |
| gG8cycpwQak+/chDrTlk8CXXlw4ztVIO1OhbdJBTav8= | 227631 | 921 | Crush | Hwang Chanhui\| Lee Seungju | Ji Hoon | Korean | Beautiful |
| zPwvlJr/cRKm9uUIP8uAS8bB4I4Rc4nuGBqMreD7ngY= | 293616 | 465 | 伍思凱 (Sky Wu) | NaN | NaN | Mandrin | 愛與愁 |
| 3O40mBcfgus9lS8x2xvdv05M94YGlUtBZl9M6hqLwG0= | 196754 | 465 | AMIT | 阿弟仔 | 陳鎮川 | Mandrin | 黑吃黑 (Double Cross) |

## User's listening histories:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Zag1Y0/unZXB0OZ/CPE2Jfg5zN3FbBc7FMEdBGkIamM= | 401705 | 465 | 陳奕迅 (Eason Chan) | Vincent Chow | NaN | Cantonese | 葡萄成熟時 |
| +mq4vHdgRaUU3hqCN8woazIUi9zqf/97VkC32Yh3N1s= | 255059 | 465 | 陶喆 (David Tao) | NaN | NaN | Mandrin | 勿忘我 |
| TljavjYae3k7lr2p7v9aDgZlGzqWstL5KoyCVj7RxEo= | 233570 | 458 | 王心凌 (Cyndi Wang) | 吳青峰 | 吳青峰 | Mandrin | 從未到過的地方 |
| 8Kn45Wt6j2eOQSYFai72rdxvmy6rXFGXCcbi92hzOAY= | 260284 | 458 | 周杰倫 (Jay Chou) | 周杰倫 | 方文山 | Mandrin | 明明就 |
| P0S2lUYcGYpi/DFZD1wv0Qw5DeqRviRT8up8pbQ9UxQ= | 261224 | 465 | 張清芳 (Stella Chang) | NaN | NaN | Mandrin | 別來無恙 |
| W+akwEPQhC1OugUaQlo0DR9xVs49UbJDZbiGvgN6ZyY= | 192470 | 921 | EXO CHANYEOL & PUNCH | Lee Seungju\| Rocoberry | Ji Hoon | Korean | Stay With Me |
| PxZo/GL0467CPE0sXAkq5HlhJCmrfDXhxYD36VkHS0c= | 252551 | 465 | 周杰倫 (Jay Chou) | 周杰倫 | 周杰倫 | Mandrin | 分裂 |
| a7l0gyrl1O9xRNL8lcJwoWthVtQgPxNzPqmBqdvToqA= | 273658 | 465 | 梁一貞 | 張揚 | 張欣瑜 | Mandrin | 還是想你 |
| +Qe3cSY/E3WzoKxnlaljQDDeTyBQyg7wfhNzfmAJLSE= | 213995 | 465 | Adele | Bob Dylan | NaN | English | Make You Feel My Love |
| 91jElb9HDytowpOj2G1AZ+UYE4nqf6Pc7VhUYiVQtDA= | 266379 | 458 | 王心凌 (Cyndi Wang) | Jia Ying Xu | NaN | Mandrin | 變成陌生人 |
| ceWym8LoRVhQsd8pj8qHCbEXTKz2pmwLZd9nGKnlUl0= | 181951 | 465 | Lady Gaga | Lady Gaga\| Mark Ronson\| Kevin Parker\| Michael ... | NaN | English | Perfect Illusion |
| SRqaf11RERRSmUzpsMrauu2QgdP31uG+PmmMBo0/7y8= | 192888 | 465 | 施孝榮 (Samson Shih) | NaN | NaN | Mandrin | 歸人沙城 |
| yZD8YUFiUHVPcR1S/WVxopCjlh+8JyU+/CHY8qC0mMg= | 174242 | 465 | Lady Gaga | Lady Gaga\| Mark Ronson\| Josh Homme\| Michael Tu... | NaN | English | John Wayne |

By comparing the recommendation and user's past preference, we can see that the recommendations accurately match what the user 's listening history. By and large, the matrix factorization methods are advanced tools that generate very accurate results.

**Future Goals**

For this project, we also tried the TF-IDF content-based methods. However, we didn't find an appropriate way to construct the user profile based on the given dataset due to the user's personal information or features like user type, age, gender are very limited. We need to find or create more features ourselves to build the user profile. And we need to define a rating score function also in order to build the TF-IDF content-based model.

## References


- Microsoft Corporation Revision. LightGBM. 2019. https://lightgbm.readthedocs.io/en/latest/

- Nicolas Hug. Surprise. 2005. https://surprise.readthedocs.io/en/stable/index.html

- Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering.2005. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.6458&rep=rep1&type=pdf
- Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. 2008. http://www.cs.rochester.edu/twiki/pub/Main/HarpSeminar/FactorizationMeetstheNeighborhoodaMultifacetedCollaborativeFilteringModel.pdf.

- Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. 1996. http://www.siam.org/meetings/sdm06/proceedings/059zhangs2.pdf.

- Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. 2001.http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf.

- Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems.2009.