Music Recommendation System with Classification and Collaborative Filtering Approaches

Hyunseok Choi
University of Pittsburgh
4200 Fifth Ave
Pittsburgh, PA 15260 USA
hyc23@pitt.edu

Meng Li
University of Pittsburgh
4200 Fifth Ave
Pittsburgh, PA 15260 USA
mel165@pitt.edu

Junchao Mei
University of Pittsburgh
4200 Fifth Ave
Pittsburgh, PA 15260 USA
jum68@pitt.edu

ABSTRACT

For music streaming companies, it is important to secure customers by recommending new songs based on their preferences. However, most of the music recommendation systems are still struggling to satisfy customers with diverse tastes of music. We focus on a subproblem: predicting which songs will be played repetitively after the first observable listening event within a given dataset of songs and users. Before running the prediction model, we conduct data preprocessing for imputation of missing values and handling categorical variables. We use several algorithms within two different approaches: Collaborative Filtering and Classification for music recommendation. In the end, we discuss limitation and future works of our models.

Keywords

Recommender system, Collaborative Filtering, Classification, Music recommendation

1 INTRODUCTION

Nowadays, people has considered music as a very important aspect of lives and a lot of people listen to music every day, leading to a giant profitable music market all over the world. For music streaming companies, it is important to secure customers by recommending new songs based on their own preferences no matter whom they are paid users or unpaid users. However, most of the music recommendation systems are still struggling on improving users' satisfaction. In this project, we focus on a subproblem: predicting which songs will be played repetitively after the first observable listening event within a given dataset of songs and users.

The existing algorithms for music recommendation can be divided into two mainstreams of data mining techniques: collaborative filtering and classification. Collaborative filtering (CF), a classic recommendation algorithm, has been a de facto standard for music recommendation systems in terms of its capability of personalized recommendation, while popularity-based recommendation systems ignore personalization. We try this algorithm as a baseline for comparison. We try the other approach, classification which includes LightGBM, a high performance gradient boosting framework based on decision tree[1], as well as other conventional classification algorithms such as support vector machine (SVM), naïve bayesian, logistic regression, k-nearest neighbors (KNN) and classic decision tree. Lastly, we compare different approaches for their strengths and weaknesses and suggest best approach for music recommendation.

2 RELATED WORK

Music Recommendation is always a challenging problem, considering the low performances given even by those "best" music recommendation systems. There are lots of researches on improving music recommendation with only one goal: increase listeners’ satisfaction.

The traditional way to recommend music is using query-search based method, which requires prior knowledge about the music to get popularity of songs[4]. Collaborative Filtering (CF) Network has better performance than query-based search model[5-7], by requiring collaboration from customers to explicitly reveal the preference of songs. If the training data is reliable, although this method is a simple model, it results in higher performance compared to the later-developed method, Context-Based Recommendation[8-

[10]. The performance of Context-Based Recommendation tends to be more unstable because it is hard to make predictions based on ever-changing users' context. In spite of its drawbacks, Context-Based Method has some advantages since it enhances the automation and obtains users' preferences implicitly to make better recommendations[11].

The word "context" means the interaction between a user and a song at this moment. On the user's side, it means his/her current situation and interests in music[12-14]. On the music's side, it represents the genre, tune, lyrics and melody[15]. And how well they interact with each other really depends on how similar between their "context". In order to measure such similarities, some techniques are used: Graph Theory[16], AdaBoost[17], k-means, Naive Bayesian[15], multi-tag indicators[18], and so on. For a user, his/her context, or the emotion, can be either revealed by social media[19], heartbeat frequency, location[20], etc., or evaluated by explicit survey[21]. During this procedure, a technique called eSM (enhanced Sentiment Metrics)[22] is developed and used for recommendation. Taken in this sense, lyric-based method[23], emotion-aware method[14,18] and location-aware[11,16] method were developed. For example, the essence of lyric-based method is using text mining with analysis of genre, mood and vocalist of a song, while EGG (electroencephalography)[24] is used to directly test the brain activity of a user to obtain his/her emotion.

We try two approaches: Collaborative Filtering and Supervised Classification, and compare the suitability of different models when applied to different situations. When comparing performances among different methods, conventional indicators can be used: Precision, Recall, F1 score, AUC(Area Under the ROC curve), etc.[19]

3 DATASET

3.1 Data Collection

The music streaming dataset is available from the Kaggle competition, WSDM – KKBox's Music Recommendation Challenge[2]. The dataset is provided by KKBox, which is one of the largest music streaming service in Asia, supported by advertising and paid subscriptions. KKBox provides a dataset of 9,934,208 records which includes information of the first music play for each unique user. If the user plays the song within a month after the first play, its target is marked 1, and 0 otherwise in the training dataset. Since the target variable is not provided in the test dataset, we use training dataset for testing and evaluation. Details of the dataset files are as in the Table 1.

Table 1: Data File Description

File Name	Description
train.csv	Streaming history data with user ID, song ID, entry point, and the target variable
test.csv	Streaming history data with user ID, song ID, and entry point.
songs.csv	Song information with song ID, length of the song, genre IDs (which can be multiple value), artist name, composer, lyricist, and language.
members.csv	User information with user ID, city, age, gender, registration method, registration date, and membership expiration date.
song_extra_info.csv	Additional song information with ISRC(International Standard Recording Code), which includes information about country code and reference year.

3.2 Exploratory Data Analysis

In the training dataset of 7,377,418 records, the distribution of user ID to the target variable and the distribution of song ID to the target variable are nearly even with 3,662,762 and 3,714,656.

Figure 1 plots some categorical predictor variables such as city, gender, language, and registered_via against the target variable. It is observed that there are no big differences from different predictor variables.

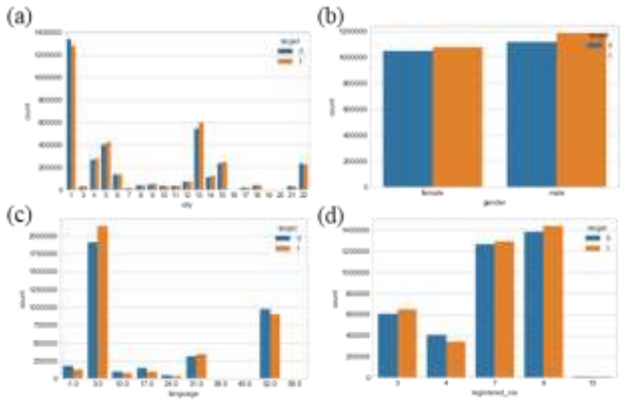

Figure 1: The histogram of categorical predictor variable with (a) city (b) gender (c) language (d) registered_via.

Figure 2 shows the relationship between numerical predictor variables released song_year and song_length, and the target variable. It seems that the relationship between those variables is vague.

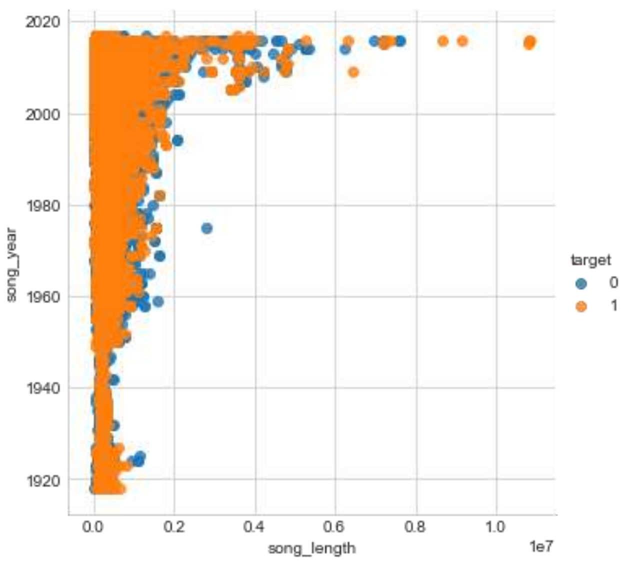

Figure 2: The plot of song_length and song_year with respect to the target variable

4 METHOD

4.1 Baseline method

Collaborative filtering based algorithm with matrix factorization is widely used for recommendation system by many companies which include the host company, KKBox. It doesn't require any additional feature information about the users or the items. For example, the model doesn't require demographic information about the users neither genre of the songs. It uses item ID (song_id), user ID (msno) and the corresponding rating (target) information for the model. In this project, the algorithm is used as a baseline method to compare how our suggested approach works better than the existing recommender system. The implementation of the model is powered by a Python library, Surprise.[28]

4.2 Data Pre-processing

Let train.csv left join with songs.csv and song_extra_info.csv on ‘song_id’ column first. Then making the result left join with members.csv on ‘msno’ column to get the initial music recommendation dataset. The dataset contains 7,377,418 records with missing values for some attributes. We split that dataset into train dataset, validation dataset and test dataset with the ratio of 6: 2: 2.

Figure 3 shows the columns with missing values, e.g., the source_system_tab column has 7,359,047 values which are not null out of 7,377,418 rows. Since ‘composer’, ‘lyricist’ and ‘gender’ columns have many null values, we simply discarded these columns. For other columns with null value, we use mode to fill in the categorical variables and mean to fill in the numerical values.

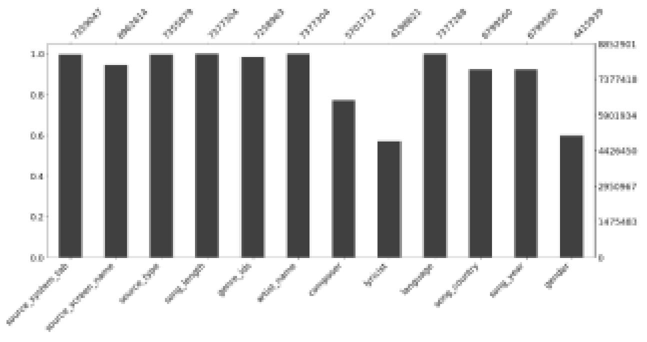

Figure 3: The columns with missing values in the initial music recommendation dataset

Since ‘bd' column means ages of members and has 2,940,499 values equal to zero. It has many outliers, so we delete ‘bd' column directly.

4.3 Feature Engineering

4.3.1 Duration. Use duration to denote the length of a user to be a member. The formula is as following.

duration = expiration_date - registration_init_time

4.3.2 Genre_ids_count. One song can belong to many genres. Use genre_ids_count to denote the number of genres a song belongs to.

4.4 Model Selection

Many algorithms are used for classification, such as logistic regression, KNN, decision tree, SVM and naïve bayesian. Some of them need lots of computing power to run the model. We sample random 50,000 tuples for model selection.

After tuning the parameters respectively, we get the following result. According to table 2, decision tree performs the best.

Table 2: Model performance for the sample dataset

	Accuracy	Precision	Recall	F1-score	AUC	RMSE
CF (Baseline)	0.559	0.552	0.647	0.596	0.558	0.663
Logistic Regression	0.617	0.616	0.630	0.623	0.617	0.619
KNN	0.570	0.570	0.592	0.580	0.570	0.655
Decision Tree	**0.624**	**0.624**	**0.636**	**0.630**	**0.624**	**0.613**
SVM	0.590	0.592	0.592	0.592	0.590	0.640
Naïve Bayesian	0.510	0.506	0.966	0.664	0.508	0.700

4.5 LightGBM

Since decision tree performs best, we implement an advanced decision tree, LightGBM, on the whole dataset. Ke et al. proposed a novel gradient boosting decision tree (GBDT) algorithm called LightGBM. [1] While model like XGBoost[3] grows tree level-wise, LightGBM is leaf-wise. Ke et al. claims that LightGBM can perform almost the same with less than one in 20 times XGBoost costs.[1] The objective function was set as binary classification and tuning parameters contain maximum depth of a tree, feature fraction, bagging fraction, types of algorithm, learning rate, etc. Most of them are used to solve overfitting problems.

4.6 Hybrid Collaborative Filtering

The pure collaborative filtering (baseline method) didn't perform well in the sample dataset. Its poor performance is a typical challenge when it faced cold-start scenarios where sufficient data is not provided so that its ratings matrix becomes sparse. Kula suggests a new approach where a model infers embeddings for users and items while merging user preferences over items. [29] That is, the approach is a hybrid model that consists of both collaborative filtering and content-based methods. User vectors and item vectors are defined by linear combinations of embeddings of the each user features and item features. To describe each feature as formulas,

A user *u* is given,

$$q_u = \sum_{j \in f_u} e_j{}^U \tag{1}$$

and an item *i* is given,

$$p_i = \sum_{j \in f_i} e_j{}^I \tag{2}$$

where F^U is the set of user features, F^I is the set of item features, $e_j{}^U$ and $e_j{}^I$ are feature embeddings for each item and user feature *f*. The prediction model is given,

$$\hat{r}_{ui} = f(q_u \cdot p_i + b_u + b_i) \tag{3}$$

where b_u and b_i are bia terms for user features and item features.

The hybrid collaborative filtering algorithm has been implemented and served by a Python library, LightFM . Since it takes input as a matrix form, it requires additional data preprocessing to transform the original dataset into rating matrix form. In the rating matrix, no interaction between user and item is represented as 0, negative interaction (no repeat) is represented as 1, and positive interaction (repeat) is represented as 2 as shown in the table 3.

Table 3: Example of the rating matrix

Iem / User	0	1	2	…	359965
0	2	0	0	0	0
1	0	2	2	0	0
2	0	0	0	0	0
…	0	0	0	1	0
30754	0	0	0	0	0

At a result of the prediction, it returns recommendation scores for pairs of each item and user. We use this core to predict songs.

5 EVALUATION RESULTS

5.1 LightGBM

After tuning the parameters by using training dataset and validation dataset, we plot the receiver operating characteristic curve (ROC) as Figure 4 and achieve the area under ROC curve (AUC) value of 0.673. Confusion matrix is shown in Figure 5 with precision value of 0.674, recall value of 0.680, F1-score of 0.677 and accuracy of 0.673.

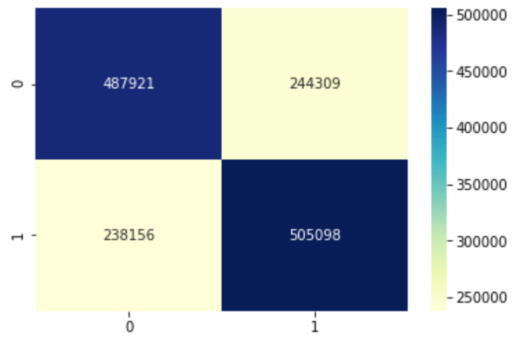

Figure 4: ROC Curve

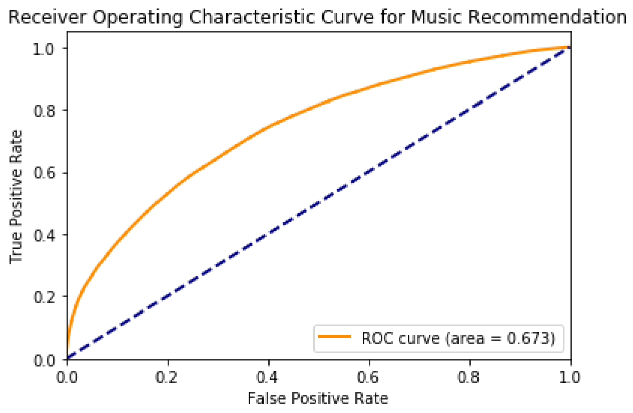

Figure 5: Confusion matrix after tuning parameters

5.2 Hybrid Collaborative Filtering

LightFM provides built-in evaluation methods such as auc_score and precision at k. AUC score of the model is calculated in a different way the classification model above because it was built for recommendation system. According to the reference guide, AUC score measures "the probability that a randomly chosen positive example has a higher score than a randomly chosen negative example." [30] Based on the definition, our model performed well with test AUC score of 0.8323. This can be interpreted that 83% of randomly chosen positive example has higher score than a randomly chosen negative example. In other words, most of the recommendation score ranks are well-ordered regardless of their magnitude. The model seems perform well, but AUC score cannot be compared directly to the AUC score of the classification models because of different definitions.

Applying the same condition to get the evaluation metrics, the structure of the evaluation of the recommendation system must be changed just like classification methods. In other words, evaluation can be done by comparing predicted values and actual values given user IDs and item IDs. The detail of the performance of the model is shown in the Table 4.

Table 4: Performance of Hybrid Collaborative Filtering

Threshold for recommendation score	Accuracy	Precision	Recall	F1	AUC
Median	0.5458	0.5487	0.5455	0.5471	0.5458
80% Median	0.5446	0.5522	0.4974	0.5234	0.5447
90% Median	0.5453	0.5505	0.5232	0.5365	0.5455
120% Median	0.5461	0.5459	0.5800	0.5624	0.5459
140% Median	0.5458	0.5439	0.5996	0.5704	0.5455

6 DISCUSSION

So far, we explored many methods to predict whether a user is going to listen to the music repetitively. Except for the pure collaborative filtering method, the models contain some degree of content-based recommendation. If we include features from text-mined lyrics, it's assumed that we can supplement the model's performance. To improve the performance of our models, further feature engineering can be conducted to have a better accuracy. To be specific for the 'artist name' column, we can use the artist popularity ranking from the test data to represent their name instead of a dummy variable which adds more dimensions to the dataset.

The reason of the poor performance of Hybrid Collaborative Filtering is assumed that it doesn't use the context of each music streaming such as menu the users entered and browsing history. The behavior of playing songs repetitively is assumed to be correlated more to music streaming context than to collaborative effects. Or it's because we applied a method which works better with explicit ratings while our data is about implicit ratings (0: not listen repetitively, 1: listen repetitively). If 1-5 explicit ratings were provided, the performance might be improved. What is more, LightFM model is designed for recommendation systems so it doesn't really fit to classification evaluation situation.

7 CONCLUSION

To summary, we've tried different models to solve music recommendation problem suggested by KKBOX. We used the pure Collaborative Filtering method as the baseline method, and tried other traditional classification algorithms: Logistic Regression, KNN, Decision Tree, SVM, Naive Bayesian, among which Decision Tree performed best. Hence, we moved onto the more advanced LightGBM algorithm, which is a boosting method based on Decision Tree model. And the result of LightGBM is pretty good (AUC=0.673). Later, we tried to Hybrid Collaborative Filtering method, but it didn't go well (AUC=0.5459).

APPENDIX

The code used in this paper is on the Github repositories:

Classification Method Implementation:
https://github.com/Meng6/Music_Recommendation_System_classification

Collaborative Filtering Implementation:
https://github.com/hyunseok-choi/MusicRecommender

REFERENCES

[1] Ke, G., Meng, Q., Wang, T., Chen, W., Ma, W. and Liu, T.Y., 2017. A Highly Efficient Gradient Boosting Decision Tree. In Advances in Neural Information Processing Systems (pp. 3148-3156).

[2] WSDM - KKBox's Music Recommendation Challenge | Kaggle. Kaggle, n.d. Web. 25 Nov. 2017. <https://www.kaggle.com/c/kkbox-music-recommendation-challenge>.

[3] Chen, T. and Guestrin, C., 2016, August. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794). ACM.

[4] Bo, S., et al. (2009). "Music Recommendation Based on Acoustic Features and User Access Patterns." IEEE Transactions on Audio, Speech, and Language Processing 17(8): 1602-1611.

[5] Braunhofer, M., et al. (2013). "Location-aware music recommendation." International Journal of Multimedia Information Retrieval 2(1): 31-44.

[6] Cai, Z.-q. and H. Hu (2016). "Session-aware music recommendation via a generative model approach." Soft Computing.

[7] Celma, O. (2010). Music Recommendation and Discovery. Spain.

[8] Chang, H.-Y., et al. (2016). "A personalized music recommendation system based on electroencephalography feedback." Multimedia Tools and Applications 76(19): 19523-19542.

[9] Chen, Y.-S., et al. (2016). "A mood- and situation-based model for developing intuitive Pop music recommendation systems." Expert Systems 33(1): 77-91.

[10] Cheng, Z. and J. Shen (2016). "On Effective Location-Aware Music Recommendation." ACM Transactions on Information Systems 34(2): 1-32.

[11] Deng, S., et al. (2015). "Exploring user emotion in microblogs for music recommendation." Expert Systems with Applications 42(23): 9284-9293.

[12] Harmon, J. (October 13, 2010). Music Recommendations Include Festival. National Mortgage News: 6-7.

[13] Hocine Cherifi, B. G., Ronaldo Menezes (2016). Complex Networks VII. 7th Workshop on Complex Networks CompleNet, Warsaw, Poland.

[14] Ja-Hwung Su, H.-H. Y., Philip S. Yu, Vincent S. Tseng (January/February 2010). "Music Recommendation Using Content and Context Information Mining." Mobile Information Retrieval: 16-26.

[15] Jiwei Qin, Q. Z., Feng Tian, Deli Zheng (2014). "An Emotion-oriented Music Recommendation Algorithm Fusing Rating and Trust." International Journal of Computational Intelligence Systems 7(2): 371-381.

[16] Kyoungro Yoon, J. L., Min-Uk Kim (2012). "Music Recommendation System Using Emotion Triggering Low-level Features." IEEE Transactions on Consumer Electronics 58(2): 612-618.

[17] Lee, K. L. a. K. (2014). "Using Dynamically Promoted Experts for Music Recommendation." IEEE TRANSACTIONS ON MULTIMEDIA 16(5): 1201-1210.

[18] Mao, K., et al. (2016). "Music recommendation using graph based quality model." Signal Processing 120: 806-813.

[19] Marcus, A. (2016). Design, User Experience, and Usability. 5th International Conference, DUXU, Toronto, Canada.

[20] McFee, B., et al. (2012). "Learning Content Similarity for Music Recommendation." IEEE Transactions on Audio, Speech, and Language Processing 20(8): 2207-2218.

[21] Mitsuko Aramaki, M. B., Richard Kronland-Martinet, Sølvi Ystad (2012). From Sounds to Music and Emotions.

[22] Park, T. and O.-R. Jeong (2015). "Social Network Based Music Recommendation System." Journal of Internet Computing and Services 16(6): 133-141.

[23] Renata L. Rosa, D. Z. R., and Graça Bressan (2015). "Music Recommendation System Based on User's Sentiments Extracted from Social Networks." IEEE Transactions on Consumer Electronics 61(3): 359-367.

[24] Taesoo Park, O.-R. J. (2015). "Social Network Based Music Recommendation System." Journal of Internet Computing and Services 16(6): 133-141.

[25] Weisi Lin, D. X., Anthony Ho, Jianxin Wu, Ying He, Jianfei Cai, Mohan Kankanhalli, Ming-Ting Sun (2012). Advances in Multimedia Information Processing. 13th Pacific-Rim Conference on Multimedia. Singapore.

[26] Xuan Zhu, Y.-Y. S., Hyoung-Gook Kim, Ki-Wan Eom (2006). "An Integrated Music Recommendation System." IEEE Transactions on Consumer Electronics 52(3): 917-925.

[27] Yuri SAITO, T. I. (2014). "MusiCube : A Visual Music Recommendation System Featuring Interactive Evolutionary Computing." Ournal of the Visualization Society of Japan 34(9): 17-27.

[28] Nicolas Hug (2017). “Surprise, a Python library for recommender systems” <http://surpriselib.com>

[29] Maciej Kula (2015). “Metadata Embeddings for User and Item Cold-start Recommendations”

[30] Model evaluation – LightFM 1.14 documentation | LightFM. 10 Dec. 2017. <https://lyst.github.io/lightfm/docs/lightfm.evaluation.html#lightfm.evaluation.auc_score>