# **Modelling Music Recommendation**

Wong Jing Hui Lim Zi Yi Lim Jia Le

U1540360A U1540659D U1540949A

November 20, 2018

# **1 Introduction**

The advent of technology has transformed how we consume music. As people increasingly prefer personalization in their consumption of music, radio channels and DJs have been replaced by online music stores and streaming services like iTunes (<https://www.apple.com/itunes/download/>), Spotify (https://www.spotify.com/), KKBox (<https://www.kkbox.com/tw/tc/index.html>) and Grooveshark (<http://groovesharks.org/>) [1]. However, there are many challenges in the algorithms for personal music recommendation service. Without enough historical data, how would an algorithm know if the user will like a new song or a new artist? How would the recommendation engine identify what songs to recommend new users? This project aims to predict whether a user will replay a song after listening to it for the first time, within a month, and therefore, recommend songs that users will enjoy the most. This can also be extended to new users by recommending songs to users of similar interests.

# **2 Data Acquisition and Processing**

The dataset is obtained from [KKBOX](https://www.kkbox.com/), Asia’s leading music streaming service in Kaggle [2]. There are two main sources of data used for our prediction: song data and members information. The song data consists of song\_id, song\_length (in ms), genre\_ids, artist\_name, composer, language. The members information consists of msno, source\_system\_tab, source\_screen\_name, city, bd, gender. For data processing, users with negative, extremely small (<6), extremely large (>90) values in age are replaced with a default value as the outlier values would cause the predictions to be inaccurate. We have also removed users with an expiry id before 2018. Our dataset includes songs that were listed after 2012. The qualitative features in table 1 are converted into levels. Repeat percentages for the following features are used: artist\_name, msno, song\_id, genre\_ids, composer. These features were excluded from the dataset: “Registration Time”, “Registration Method”, “Expiration Date Reg” and “Lyricist”. The data was split into training and test data randomly, with 70% training data and 30% test data. Training data was then normalized to reduce the bias of predictions arising from large values. Methods that required cross validation had normalization done on the pseudo training set instead. For normalization, the features were normalized by the mean and divided by the standard deviation. It was not feasible to process a few categorical features, namely msno, song\_id, artist\_name, genre\_id and composer, which had too many levels. Therefore, in order to obtain better information from these features, we decided to use their repeat percentage instead. This is calculated by grouping according to the feature, and calculating each group’s average repeating percentage.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Categories** | **Features** | **Details** | **Examples** |
|  | msno | user id (anonymized) | “//h43Z3YcFMRGM…” |
| song\_id | song id (anonymized) | “sgHKhG2d57U2o88…” |
| Source\_system\_tab | Name of the tab which users accessed, to listen to the song | My Library, Radio, Discover |
| Source\_screen\_name | Name of the layout a user sees | My Library\_search, Discover chart, Discover Feature, Discover Genre |
| **Qualitative** | Source\_type | An entry point which a user first plays music | Album, online-playlist, local-playlist |
| Language | Language (index as integers) | 60 |
| City | The city which the user resides in | 1, 13, 22 |
| Gender | User’s gender | Male, Female |
| **Quantitative** | Song\_length | Song length in milliseconds | 247640 |
| BD | User’s age | 15, 28, 59 |
| Artist\_repeat\_percentage | For any given artist, the ratio total number of songs replayed against total number of songs listened | 0.26 |
| Msno\_repeat\_percentage | For any given user id, the ratio total number of songs replayed against total number of songs listened | 0.16 |
| Song\_repeat\_percentage | For any given song id, the ratio total number of times the song is 0.26replayed against total number of times the song was listened | 0.48 |
|  | Genre\_repeat\_percentage | For any given genre, the ratio total number of songs replayed against total number of songs listened | 0.63 |
| Composer\_repeat\_percentage | For any given composer, the ratio total number of songs replayed against total number of songs listened | 0.78 |

Table 1: Features utilized in statistical learning methods

# **3 Methods and Results**

The statistical learning models used for our prediction are: Collaborative filtering (SVD), Logistic regression, Classification Trees, with Random Forests, xgBoosting and Light Gradient Boosting Machine (LGBM). The model performance is assessed by comparing the mean square errors of the test set and the interpretability of the models. Subsequently, we will discuss the advantages and disadvantages of each model with respect to the context of music recommendation engine.

# **3.1 Collaborative Filtering (SVD)**

Collaborative filtering is a technique used in recommender systems to consider a variety of criteria and recommend songs based on similarities to other users [3]. We apply Singular Value Decomposition on Collaborative filtering to minimize the mean square error and predict the probability (rating) of the user replaying a song. The features used are msno and song\_id to form a user-item pair. We apply a classifier on the rating; for probability more than 0.5, we classify the output as 1 (the user will replay the song) and 0 (the user will not replay the song), otherwise.

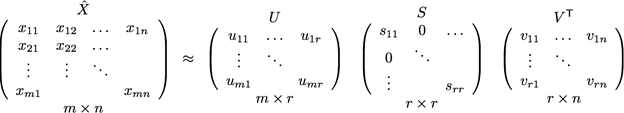


Figure 1: Matrix Factorization with Singular Value Decomposition (SVD)

Matrix X denotes the utility matrix of m users and n songs and each entry represent the ratings. U is a left singular matrix, representing the relationship between users and latent factors. S is a diagonal matrix describing the strength of each latent factor, while V transpose is a right singular matrix, indicating the similarity between items and latent factors. SVD decreases the dimension of the utility matrix by extracting the latent factors by mapping each user and song into a latent space with smaller dimension *r, to* understand the relationship between users and songs and predict the ratings. The results are summarized below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | MAE (Pseudo Test set) | MSE (Pseudo Test set) | Misclassification rate  (Test error) |
| **CF (SVD)** | 42.58% | 22.01% | 35.20% |

Table 2. Results of Collaborative filtering (SVD)

**3.2 Logistic Regression**

Logistic Regression was performed with the 13 features (except msno and song\_id) from Table 1 as inputs. As expected, logistic Regression yielded a higher misclassification rate than Collaborative Filtering. This suggests music consumption is highly personalized that is less influenced by factors such as genres and song length. However, the performance of Logistic Regression can be improved with regularization methods, namely Ridge and Lasso regression. 10-fold Cross-validation was carried out to determine the best penalty tuning parameter, lambda, from a range of 5e-07 to 5e+06. Results in Table 3 have shown that regularization has improved the model performance and is close to the performance of Collaborative Filtering. Both the Lasso and Ridge regularization methods have similar performance, indicating both L1 and L2 penalty tuning parameters are equally effective in improving test error. However, Lasso regression helps in the feature selection by tuning coefficients of certain features to 0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | MAE (Test set) | MSE (Test set) | Misclassification rate  (Test error) | Lambda |
| **Logistic Regression** | 44.08% | 44.08% | 44.08% | - |
| **Logistic Regression (Lasso)** | 42.73% | 23.07% | 35.09% | 5.0665e-07 |
| **Logistic Regression (Ridge)** | 42.73% | 23.07% | 35.08% | 0.00017 |

Table 3. Results of Logistic Regression

# **3.3 Classification Trees**

Classification Trees was another technique chosen, due to its high interpretability. In addition, with the aid of Random Forest and Boosting, this method has performed considerably well in similar prediction problems. Furthermore, classification trees are able to handle the 8 categorical features very well, without the need for additional encoding through dummy variables. Performed on the 13 features similar to Logistic Regression, Random Forest, xgBoost and LightGBM was applied. Both boosting techniques make use of leaf-wise growth instead of traditional level-wise growth. This enables better performance in both accuracy and efficiency. For Pruned Tree, Cross-Validation approach was used to obtain the optimal tree complexity. However, this did not improve the misclassification rate. For Random Forest and xgBoost, the parameters of max\_depth and number of estimators were optimized using gridSearch. Gridsearch allows inputs of a range of parameters, and would find the combination with returns the best classification rate. From the classification trees, we are also able to obtain the importance of each feature. From the Random Forest algorithm, the three most important features are the song repeat rate, user repeat rate and composer repeat rate, with importance of 0.362095, 0.268958 and 0.101797 respectively. The importance can also be seen in the graph plotted below, Figure 3.

|  |  |  |
| --- | --- | --- |
|  | Misclassification rate  (Test error) | Parameters |
| **Classification Tree** | 37.47% | - |
| **Pruned Tree** | 37.47% | Optimal Tree Complexity is 6 |
| **Random Forest** | 33.28% | N\_estimators : 200  Max\_depth : 10 |
| **xgBoost** | 31.15% | N\_estimators : 250  Max\_depth : 10 |
| **LightGBM** | 33.74% | N\_estimators : 250 |

Table 4. Results of Classification Trees

# 

Figure 2: Plot of Classification Tree (Pruned)

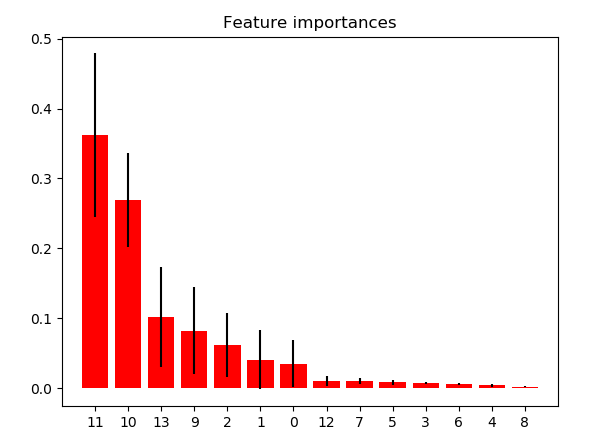


Figure 3: Plot of Feature Importance

# 

# **4 Discussion and Conclusion**

We develop a statistical learning pipeline for recommendation of songs to users using Collaborative Filtering, Logistic Regression with regularization and Regression Trees with boosting and random forest. The performance across all the models are similar, between 30% to 35% misclassification rate. This shows that consumption of music is highly personalized and differs broadly between users.

Regression Trees with xgBoost has the best performance and outperforms Collaborative Filtering, which is a common approach in song recommendation. This might be due to the fact that the categorical features were well-handled by the classification trees model. Furthermore, classification trees can be easily understood and interpreted, especially in the normal and pruned tree. Even in the case of Random Forest, a certain level of interpretability is maintained, as it is able to provide us with the importance of each feature. This research is significant as current music streaming companies should model regression trees in song recommendation engines. Regression Trees serve as an alternative to Collaborative Filtering, which performs poorly with small user dataset.

The performance of Collaborative Filtering can be improved by adding features like genre\_id to provide contextual information to predict the ratings matrix more accurately. The performance of Classification Trees can also be improved further, through the use of larger and more comprehensive datasets. Other forms of boosting can also be used for prediction. Future area of research can involve lyric-based text mining to obtain tag categories to predict songs that users will listen to. Similarly, Natural Language Processing (NLP), Neural Networks (NN) or Support Vector Machine (SVM) can be used to predict and recommend songs to users.

There are many classification algorithms applied to classify songs, such as Support Vector Machine, Artificial Neural Network and Gaussian naive Bayes. Many existing methods employ only a single classification algorithm. The single classifier has its own inherent inadequacy. For instance, SVM has high computational complexity [4], and its performance is heavily dependent on its defined parameters. Logistic regression requires huge samples to achieve stable performance. Rather than applying a single classifier, the ensemble learning attempts to train multiple models using the base classification algorithms, then stacking them to make the final predictions. The prediction accuracy might be enhanced if we apply the stacking ensemble method. The further work on this project could also begin with text mining on the song lyrics to identify different genres that the particular user would prefer to listen.

# 

# **References**

[1] YouGov. (2018, March 1). Rise of music streaming services sees radio and CDs suffer. Retrieved from <https://sg.yougov.com/en-sg/news/2018/03/01/music-streaming-services/>

[2] Kaggle Inc. (2017, September 28) WSDM - KKBox's Music Recommendation Challenge*.* Retrieved from <https://www.kaggle.com/c/kkbox-music-recommendation-challenge>

[3] A new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques. (2018). 2018 9th International Conference on Information and Communication Systems (ICICS), Information and Communication Systems (ICICS), 2018 9th International Conference On, 102.<https://doi-org.ezlibproxy1.ntu.edu.sg/10.1109/IACS.2018.8355449>

[4] Time Complexity Analysis of Support Vector Machines (SVM) in LibSVM. (2015). International Journal of Computer Applications (0975 – 8887) Volume 128 – No.3, October 2015. Retrieved from <https://pdfs.semanticscholar.org/a8b4/786c9128d4a94caeb67c858ab4f4288c49ff.pdf>