KKBox's Music Recommendation Project Report

Group 7 MSIS2508

Chia-Ling Ni, Qian Gao,Yi-Hsuan Tseng, Ziqi Liu

I. Project idea & Motivation

KKBox is a music streaming service developed in 2005 by KKBox Inc, a software company in Taipei, Taiwan. KKBOX is an Asian leading music streaming service and mainly targets the music markets of East and Southeast Asia, focusing on regions including: Taiwan, Hong Kong, Malaysia, Japan and Singapore. It holds the world's most comprehensive Asian-Pop music library with over 90 million tracks and a diverse music recommendation system.
To successfully build personalized recommendation algorithms to the users with unlimited streaming services, it is essential to analyze the user's decision and clearly state the factors that influence the listener's choice to replay a given song. Based on this idea, we decide to predict the chance of replaying the certain song within a month based on users' behaviors and song features.

II. Data Description & Data preprocessing

We pull more than 9.9 millions data from Kaggle which contains 5 (.csv)files. The Members file contains member information such as id, age, gender, registration date and so on. The Songs file contains song id, length of song, genre id, artist name, composer, lyricist and language. The Song_extrainfo file provides extra information for the song such as the singer name. Since the dataset is for a competition challenge, they already split the original data into training and testing sets and separate into two files. Due to the missing target values in the Test file, we decide to use the Train file only as our entry data which still contains 7.3 millions rows. The Train file contains the user id, song id, source system tab, source screen name, source type, and target, representing whether the user listens to the recommended song again within a month.

To deal with this large amount of data, we first combine all Train, Members, Songs, and Song_extrainfo files into one huge dataset having 20 columns. Secondly, we examine the data as a whole and find some useful observations, for example, we surprisingly find some of the features contain nearly 30% - 40% of missing values. Based on this understanding, we handle those 'NA' values whether drop or fill in with 'Others' depending on the proportion and the correlations with target values. After scanning and picking useful features, we use label encoder to deal with the categorical values and standard scaler for numerical data. Compared with OneHotEncoder, LabelEncoder helps us to assign each distinct category as one index in the same columns instead of expanding lots of columns with binary values.

III. Exploratory Data Analysis

In machine learning classification problems, models will not work as well and be incomplete without performing data balancing on train data. Therefore, balancing data gives us the same amount of information to help predict each class and gives a better idea of how to respond to the test data. As we can see in Fig. 3-1, the 'target' variable in our train data is around balanced.

The following insights have been arrived at while exploring the data initially:

- Our observation across "Gender" shows in Fig. 3-2, the proportion of male and female is almost balanced. However, 40% of the data is unknown. This result indicates that there might be some missing values or some users refusing to share their gender. As a result, "Gender" column will not be our focusing feature in the models.
- In Fig. 3-3, user registration rate has increased a lot since 2010 and reached its peak point in 2016. In 2017, the registration rate has sharply decreased. One of the reasons might be Spotify-another well-known streaming platform expanded its student discount program to 33 more countries including Asia in 2017.
- On further observation across ages shown in Fig. 3-4. "bd" feature(Age) contains many zero values and some negative values like -43, but we know any user cannot have age zero or negative. Therefore, we will not regard the feature that contains lots of outliers as an important one.
- As seen in Fig.3-5, Genre_id 465 has the highest count - 3717210 which indicates this genre could be the mainstream music in the market so far.
- By comparing different sources of entry in the app from Fig. 3-6, 3-7, 3-8, we can conclude there are more people using the local library as an entry point to first play music on mobile apps. Users listening from "my library" have the highest chance to listen to the same song within a month. On the contrary, songs played by the radio have the lowest chance to be listened again by the same listener within a month.

IV. Model Assessments & Evaluation

Based on the Exploratory Data Analysis, we selected 15 features for our classification models. In addition to that, we randomly sample 20% of the data from the whole dataset to perform the models due to limitations of computer power. After that, we split the dataset into train, validation and test sets by a fraction of 64%, 16% and 20%.

1. K-nearest neighbors (KNN)

We first choose the KNN model as the baseline model to examine how a simple classification model performs with our dataset. We use GridSearchCV in Scikit-Learn to pick the best n-value parameter, which would be 13 in this case. The result shows that KNN model cannot perform well and only achieves an accuracy rate of 54.72% on the test set. Furthermore, the accuracy rate on the train-validation set is 63.65%, which is much higher than the test set score, indicating there might be an overfitting problem with this model.

2. Random Forest

Since there is a huge amount of data and plenty of features within our dataset, we decide to use Random Forest instead of a simple decision tree as the second model. We still use GridSearchCV in Scikit-Learn to pick the best parameters like max_features, max_depth and n_estimators, which shows us the best model having max_depth=10, max_features=10, and n_estimators=150. With the best Random Forest model, the performance accuracy rate goes up to 63.63% on the test set, which is much better compared to the KNN model. Furthermore, the accuracy rate on the train-validation set is 63.92%, similar to the one we had on the test set, indicating less overfitting problems with this model.

3. XGBoost

To further improve the performance, we also use Gradient Tree Boosting classification models such as XGBoost, CatBoost and LightGBM. Using GridSearchCV in Scikit-Learn to pick the best parameters and the result shows that the best XGBoost model is having max_depth=10, max_features=10, n_estimators=150 and other default parameters, giving 67.04% accuracy rate on test set. However, the train-validation set also shows a slightly higher accuracy rate of 71.37%. Thus, even though XGBoost has the best performance so far, there is a slight overfitting problem from this model, and we could actually try regularization methods or adjusting other parameters for future performance improvement.

4. LightGBM and CatBoost

We decided to use the default setting of parameters in Scikit-Learn for these two Gradient Tree Boosting classification models. The test set result shows 64.22% of accuracy by using LightGBM and 65.44% accuracy by using CatBoost classifiers. There is no distinct difference on the train-validation set, 64.29% and 66.30% individually. The performances of these two models are better than KNN and Random Forest but not as good as the XGBoost Classifier model.

5. Best Model's Performance Metrics and Feature Importance

Comparing all performances among our models, the XGBoost gives the best performance with an accuracy rate of 67.04%. Examining other performance metrics, Precision rate is 66.3%, and Recall rate is a little bit higher at 70.3%. Looking into the feature percentages within the model from Fig. 4-1, the 3 features related to sources are all within top 10, indicating that source is an important factor to our target variable. We can see that 'msno' feature is also within top 10, but here it only indicates the user_id label. We would further aggregate this user_id to show user behavior as a new feature for our model improvement.

V. Model Improvement

This is one of the competitions on Kaggle, when comparing with some of the best prize winning models, one of the biggest differences is the features selection. We believe that further digging to feature engineering, there is still room for improvement. To improve our model performance, we decided to change feature selection as follows:

- Removed the data that's unique to each single song: their name and id
- Combined the initial registration date and the expiration date into one membership duration
- Added the user count feature which counts how many times a user repeatedly used the app to listen to songs

With the hyperparameters staying the same, each model performance increased slightly. We are confident to say that the new set of features helps with our model performance, although which one has the highest contribution requires further examination. More feature engineering could further help with our model performance.

VI. Business Insights

- Recommended song list from KKbox leans heavily from the same genres, artists and languages. Alternative algorithms could be utilized, for example they can launch “Discover Weekly”, the recommendation will be more diversified.

- Based on the accuracy rate and feature importance from the models, KKBOX could get deeper insights on which features they can look into more, such as source types.
- User behaviors are important features for music recommendation, but some users' information, like genders and ages, are not correct or missing from the data. KKBOX could also include other kinds of user behavior features, such as if the user will listen to the song completely or not, for better music recommendation.
- The input contains text data only, and no audio features. KKBOX currently uses a collaborative filtering based algorithm with matrix factorization and word embedding in their recommendation system. We believe new techniques could increase the accuracy of recommending music to users and lead to better results.

VII. Conclusion & Future work

We could introduce new forms of features, such as adding polynomial or logistic forms of the song length, integrating users’ biological properties such as age, region, education background, etc. These could further improve the prediction accuracy. The way to handle N/A values in the dataset may also influence the accuracy, which requires further examination to find the most practical and effective method.

The bigger the sample size the better the model performance, but there is a tradeoff between accuracy and training speed. With our current computer setup we are not able to include all the available samples in our model, but for sure that the model performance could be further improved. We need to keep in mind that increasing samples to a certain size will result in a diminishing improvement, so we have to value our computing power as well to balance the benefit of a higher accuracy v.s. a shorter training time.

VIII. Appendix

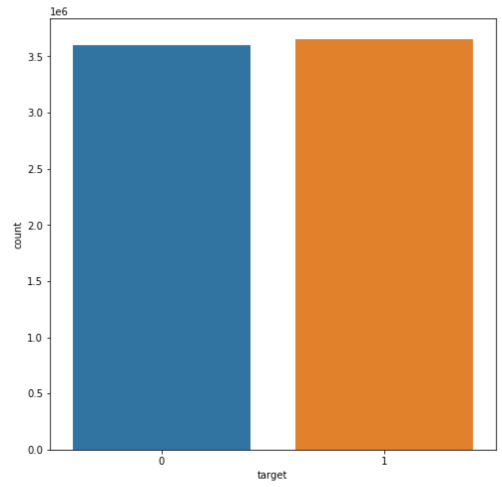

Fig. 3-1 Balanced Data Bar Plot

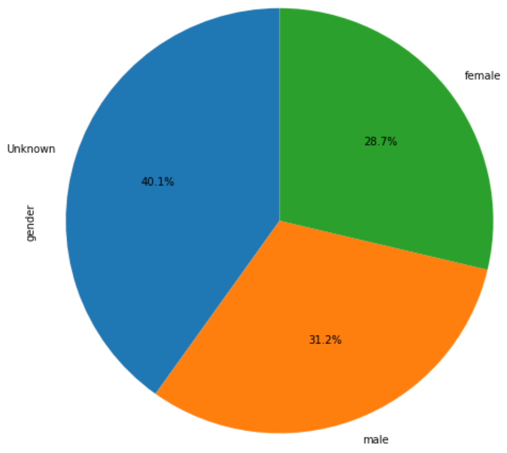

Fig. 3-2 Gender Pie Chart

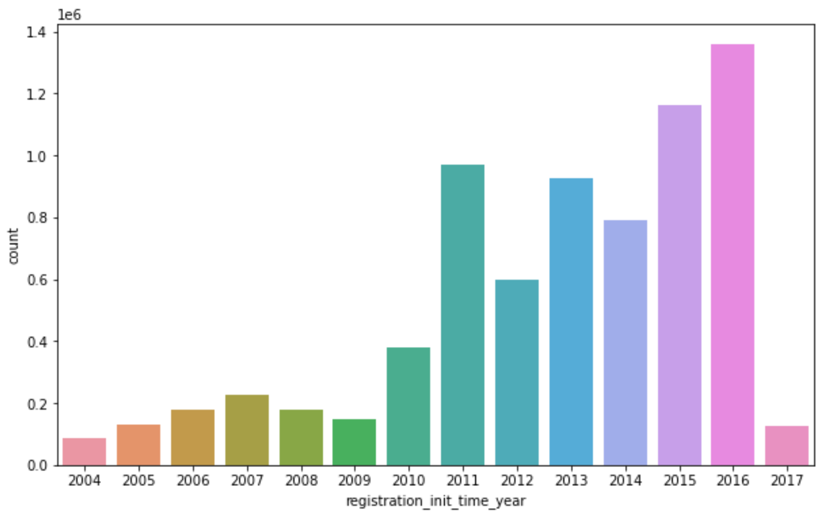

Fig. 3-3 User Registration Rate Bar Plot

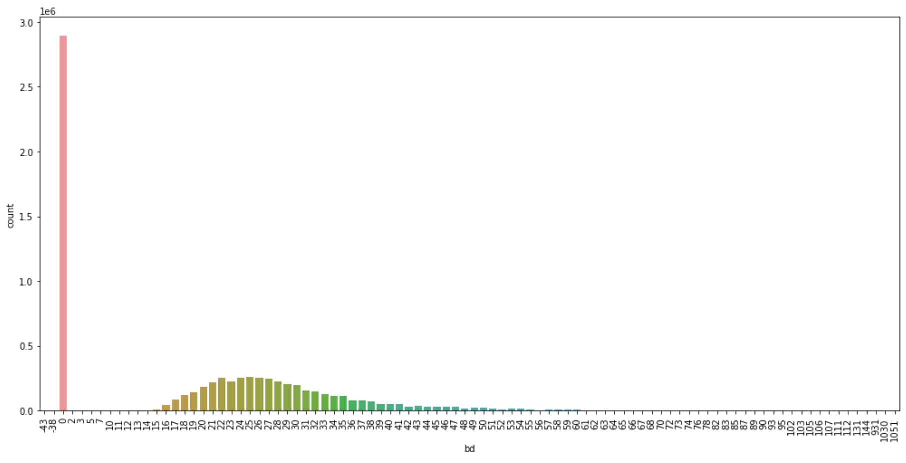
Fig. 3-4 Age Distribution

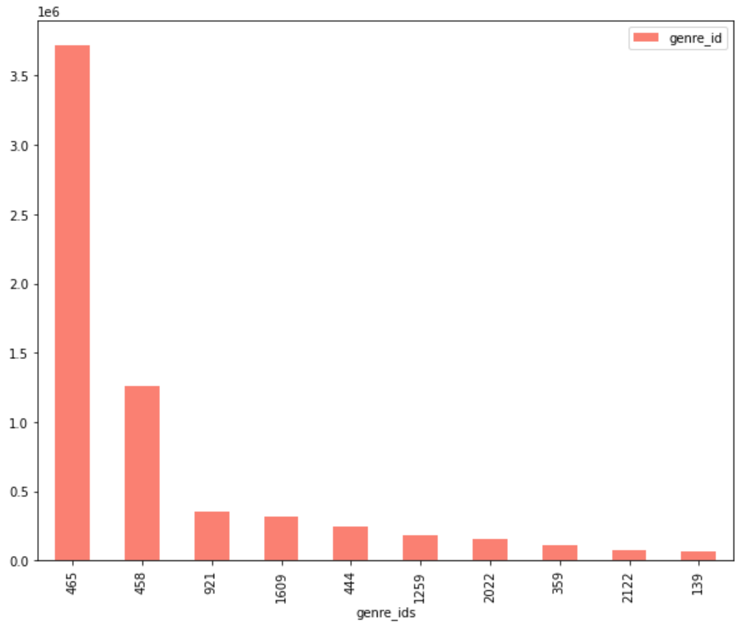

Fig. 3-5 Top 10 Music Genre

(Data has been encrypted)

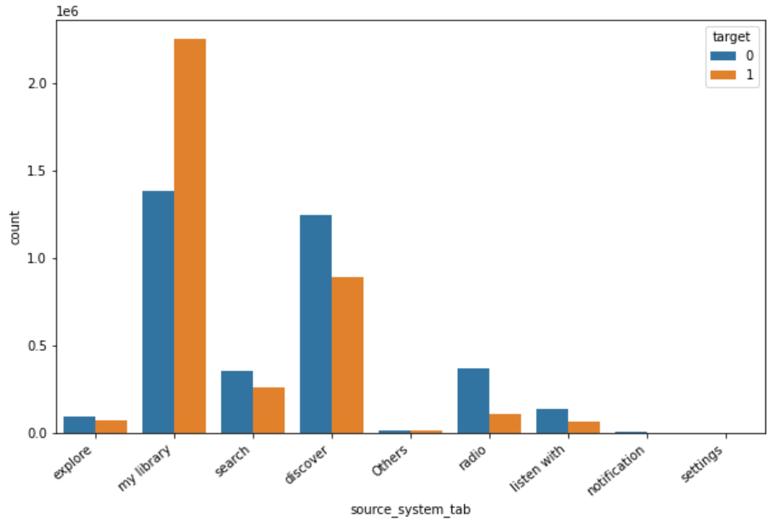

Fig. 3-6 Source System Type

(The name of the tab where the event was triggered)

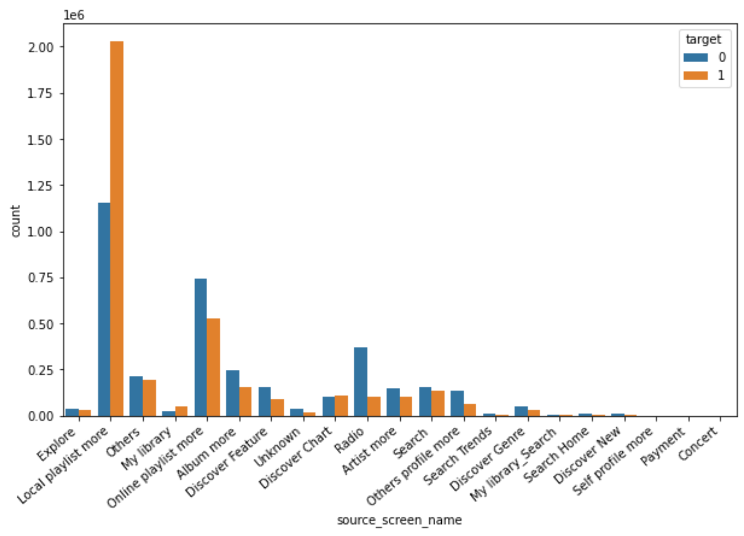

Fig. 3-7 Different Source Screen
(Name of the layout a user sees)

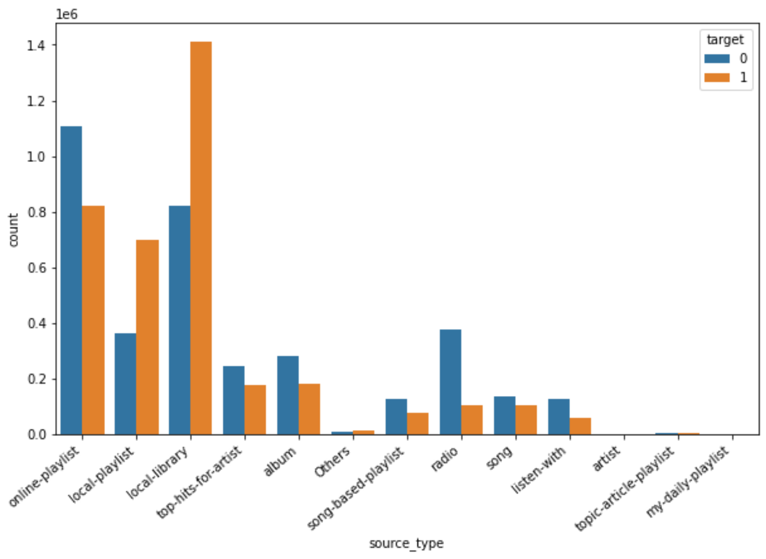

Fig. 3-8 Different Source Type
(An entry point of a user first plays music on mobile apps)

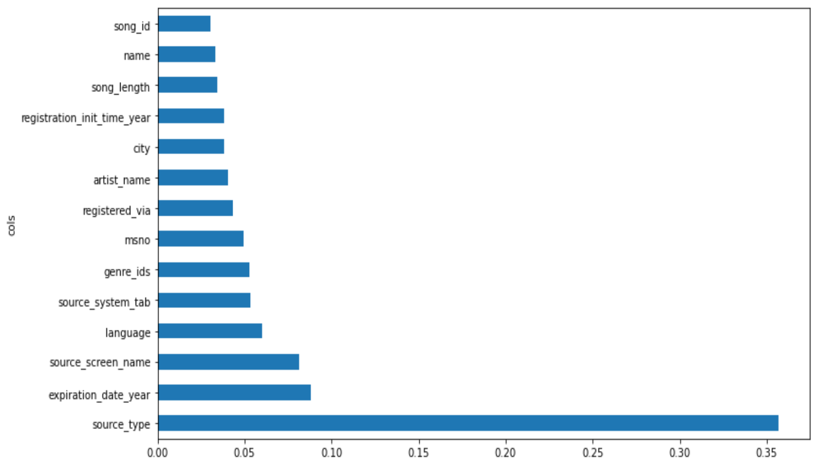

Fig. 4-1 Feature Importance Percentages from XGBoost Model