Final Report

Hao Ding, Jiachuan Deng and Ruijiu Mao*

December 7, 2017

The glory days of Radio DJs have passed, and musical gatekeepers have been replaced with personalizing algorithms and unlimited streaming services. Without enough historical data, how would an algorithm know if listeners will like a new song or a new artist? And, how would it know what songs to recommend brand new users? Our group aims at help solving these problems and build a better music recommendation system. The dataset is from KKBOX, Asia's leading music streaming service, holding the world's most comprehensive Asia-Pop music library with over 30 million tracks.

Data (https://www.kaggle.com/c/kkbox-music-recommendation-challenge/data) includes:
1. train.csv: user id, song id, source user's event triggered, label (1: recurring listening event within a month, 0: otherwise)
2. songs.csv: song id, song length, genre category, artist name, composer, lyricist, language
3. members.csv: user id, city, gender, birthday, registered source, registration init time, expiration date

We plan to construct and train a random forest model to build a binary classifier (may use xgboost package in our implementation), we may also further modify the model based on the performance. We will also compare the performance of xgboost with SVM and KNN using the same set of features. The dataset will be split into train set, test set and validation set in the ratio of 7:2:1, and k-fold cross validation (k may be 5 or 10) technique will be applied in parameter tuning process. The parameters we will tune may include: learning rate, max depth of tree, subsample ratio, max number of trees.

The metrics we adopt to evaluate our model are listed as follows:
1. Number of samples versus accuracy or ROC(AUC): We will train 20%, 40%, 60%, 80%, 100% of training set respectively, and observe the changes of accuracy and ROC_AUC. Accuracy should be higher if we train model with more samples, which suggests superb scalability of model. When the model fails, the accuracy may remain stable at certain point or marginally increase, indicating a limited representation capability.
2. ROC curves (AUC): If the model performs well, the AUC will be large (in the best condition, it will be almost 1.0); if the model fails, AUC will be small.

Language to use: Python

1 Data Preprocessing

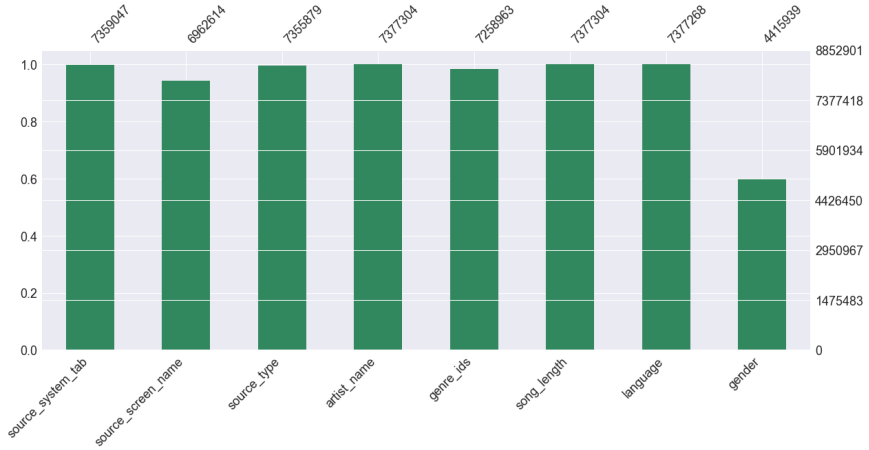
We first merged train data with songs' and members' information on song id and msno (user id) respectively, the merged dataframe as following:

msno	song id	src.sys.tab	src.scrn.	src.typ.	tgt.	artist	genre id	length	language
9176	86884	2	8	7	1	3785	308	206471.0	52.0
19273	260594	4	9	5	1	36868	98	284584.0	52.0
19273	140755	4	9	5	1	24602	98	225396.0	52.0
19273	27577	4	9	5	1	31652	7	255512.0	-1.0
9176	38706	2	8	7	1	5191	3	187802.0	52.0

*H. Ding, J. Deng and R. Mao are with Purdue University, West Lafayette IN 47907 (email: mao95@purdue.edu).

city	bd	gender	reg.via	reg.yr.	reg.mth.	reg.date	exp.yr.	exp.mth	exp.date
1	0	0	7	2012	1	2	2017	10	5
13	24	1	9	2011	5	25	2017	9	11
13	24	1	9	2011	5	25	2017	9	11
13	24	1	9	2011	5	25	2017	9	11
1	0	0	7	2012	1	2	2017	10	5

We explored missing values in dataframe and fill in -1 as replacement. The details of missing values are as following:



2 Model Selection

Before running models for binary classification, we discussed several types of machine learning models. There are generally 6 types of models: decision tree based methods, linear regression based methods, neural network, bayesian network, SVM and nearest neighbor. We pick SVM, KNN from nearest neighbor, and Lightgbm and XGBoost from decision tree based methods.

- SVM is based on finding a linear plane with maximum margin to separate two class of output.
- KNN finds the top k closest images, and have them vote on the label of the test image.
- Decision tree based methods recursively divide the training data into buckets of homogeneous members through the most discriminative dividing criteria. During the learning, various dividing criteria based on the input will be tried ; input will first be turned into binary and then use the true/false as a decision boundary to evaluate the homogeneity.The members of the bucket represented at leaf node will vote for the prediction.

We want to compare the results based on different types of learning models.

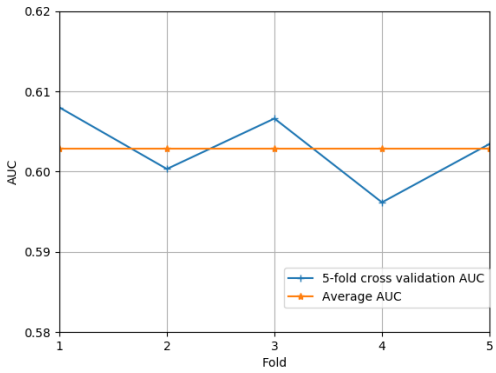
3 Cross-Validation

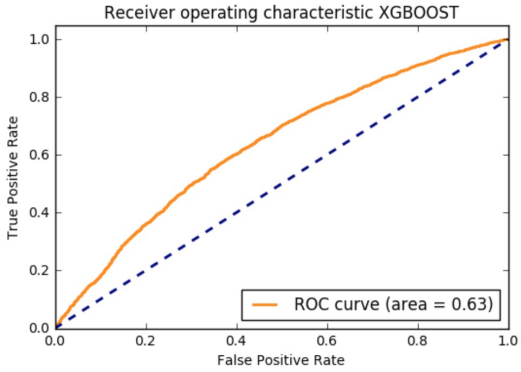
3.1 Implementation on XGBoost Model

XGBoost parameters setup:

$$
\begin{aligned}
&params =' binary : logistic'\\
&params['eta'] = 0.7\\
&params['max\_depth'] = 8\\
&params['silent'] = 1\\
&params['eval\_metric'] =' auc'\\
&XGBmodel = xgb.XGBClassifier(max\_depth = 8,\\
&\qquad\qquad\qquad\qquad objective =' binary : logistic',\\
&\qquad\qquad\qquad\qquad learning\_rate = 0.7\\
&\qquad\qquad\qquad\qquad).
\end{aligned}
$$

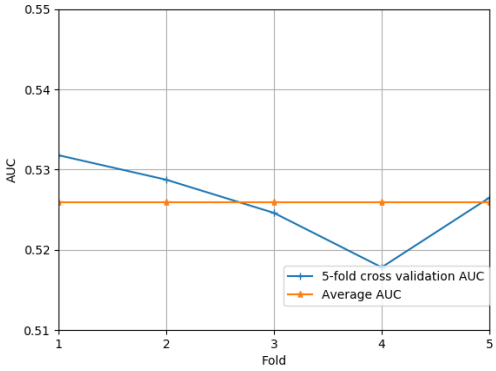
Plot of AUC for k-fold cross validation, and plot of ROC curve:

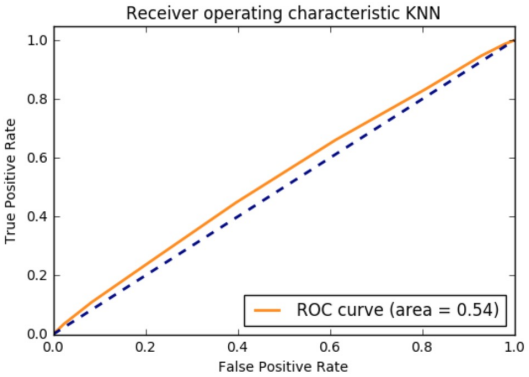




3.2 Implementation on KNN Model

When using KNN model (k=10), we get plot of AUC for kfold cross validation, and plot of ROC curve:





Note that the average AUC is less than 0.53. Since the problem is binary classification, random selection will achieve similar results. We don't consider KNN as a appropriate model.

3.3 Implementation on SVM Model

We implemented and trained SVM Classifier with same features as above. Unfortunately, it takes extremely long time to run. We attribute the issue to categorical features of the data.

3.4 Analysis

From the above comparison, the advantage of XGBoost model is obvious.

We conclude the drawbacks of SVM and KNN as follows. Given that one-hot encoder is not a good choice, we applied label encoder to handle the categorical attributes in our dataset. Label encoder will convert categories into a sequence of integers. In this way, we can transfer our raw data into a feature vector. The drawback of label encoder is obvious: the distance between feature vectors will prone to be less informative or even misleading.For example, the label encoder will encode categories 'red', 'yellow', 'blue' into 1, 2 and 3, while the distances of ('red','blue') and ('red', 'yellow') are different. The differences between those categories should be the same (not belonging to the same category). That is exactly the reason why SVM and KNN are not appropriate models for our dataset.

4 Feature Engineering

Our previous model simply used every feature from the raw dataset, we improve our model with feature engineering by constructing and refining features.

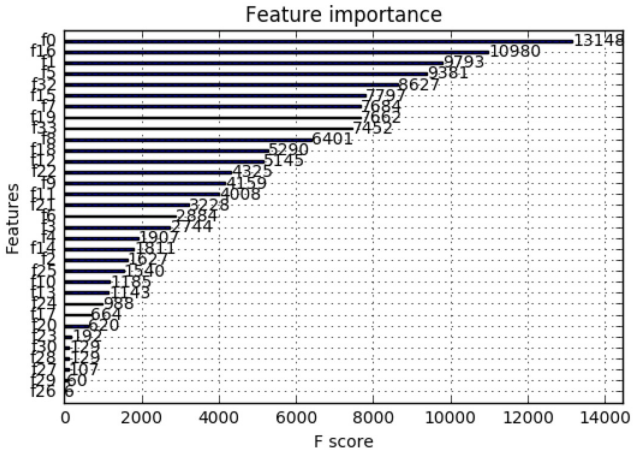
4.1 Adding New Features

Based on our understanding of the data, we added several new features, the followings are the most representative ones:

Feature Added	Description
genre_ids_count	Count Each Genre's Number
lyricists_count	Count Number of Lyricists for Each Song
composer_count	Count Number of Composers for Each Song
song_lang_boolean	Check Whether Song's Language Belongs to 17 or 45 (Songs have the highest percentage of 1s with values 17 and 45 in response variable)
count_song_played	Count Number of Times having been played
count_artist_played	Count Number of Times Each Artist's Songs having been played

4.2 Feature Analysis

In total, we have 34 features. We evaluate the importance of each feature by computing their F-score, which is the sum of times each feature's presence in all the trees built by XGBoost model. The distribution of their scores varys a lot. As a result, we believe that feature selection may provide improvement.



4.3 Feature Selection

We use the following greedy algorithm for feature selection.

```
Algorithm: Greedy Algorithm for Feature Selection
previousAUC ← model.cal_auc(X,y)
deletedFeatures ← ∅
for f in all features:
    X_temp ← X delect feature f
    auc_temp ← model.cal_auc(X_temp,y)
    if auc_temp>previousAUC:
        X← X_temp
        previousAUC ← auc_temp
        deletedFeatures.add(f)
Return deletedFeatures
```

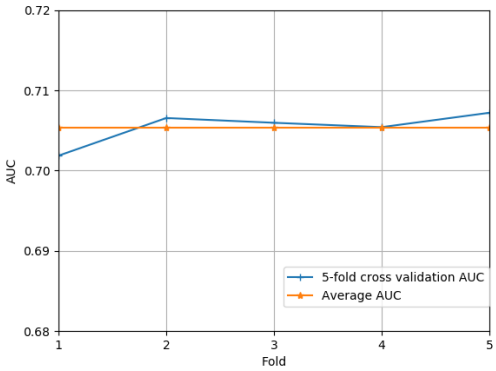
5 Baseline Change

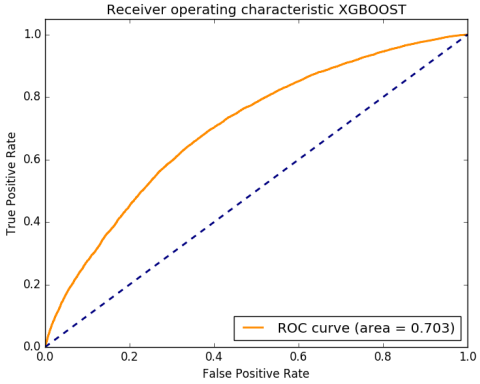
From the performance above, we realized that SVM and KNN are not suitable to be the baselines because of our categorical data structure. We considered about one-hot-encoder to resolve categorical data structure issue, but the size of features will explode. Decision tree based model should be more appropriate. As a result, we decide to add LightGBM and Random Forest as our new baselines.

5.1 Model Performance

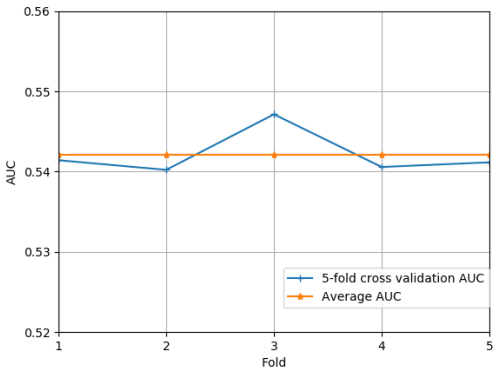
Plots of AUC for k-fold cross validation, and plots of ROC curve are the following:

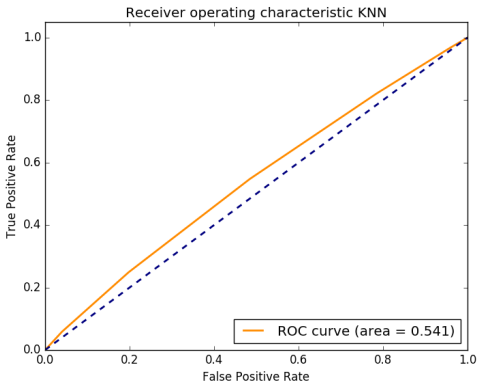
XGBoost:



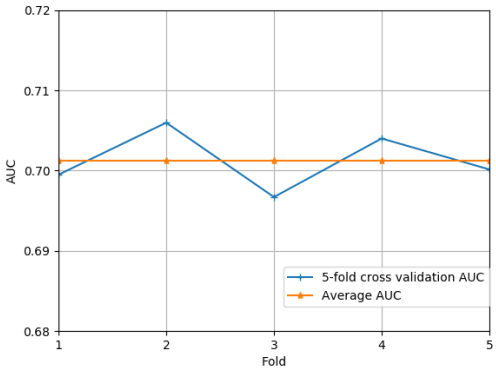


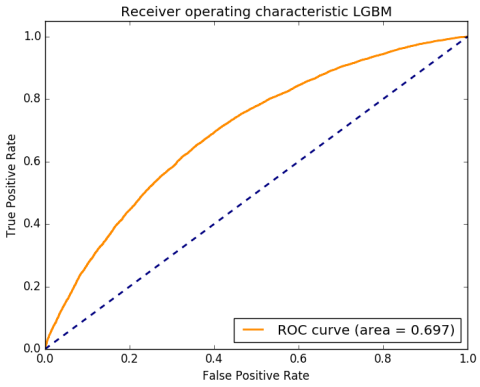
KNN:



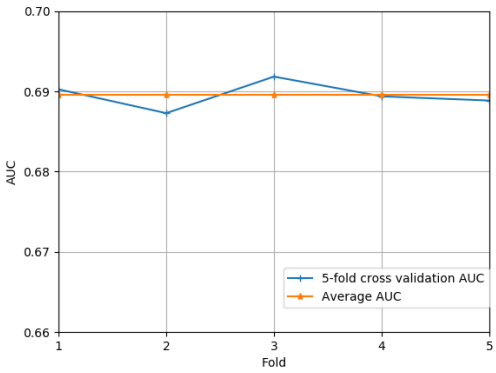


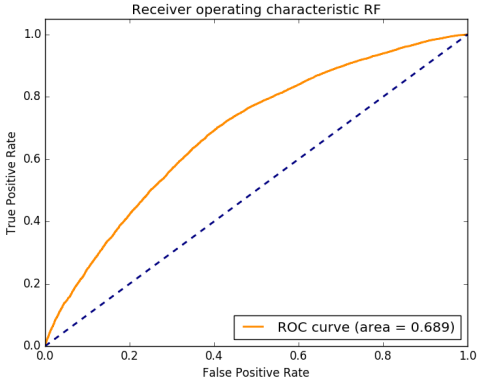
LightGBM:





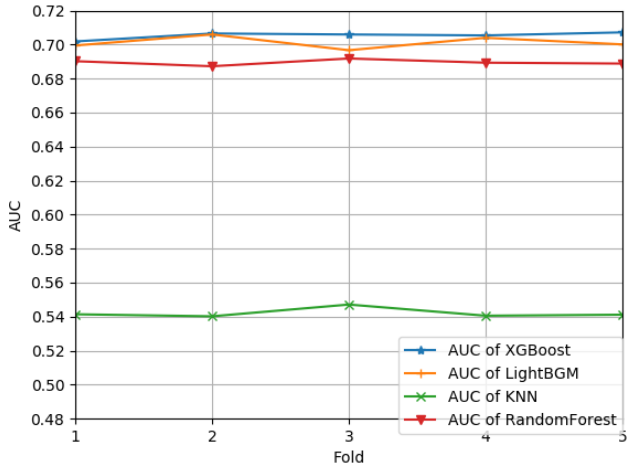
Random Forest:





5.2 Comparison

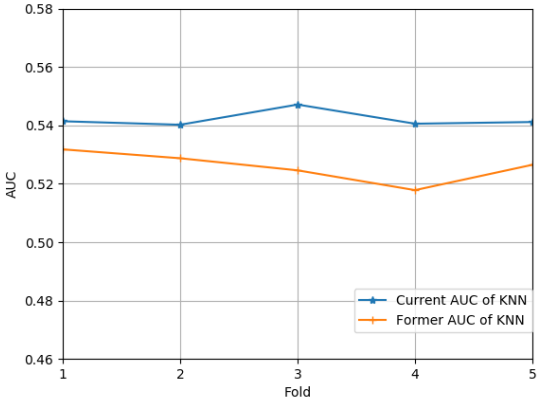
We compare the three models by their AUC curves:

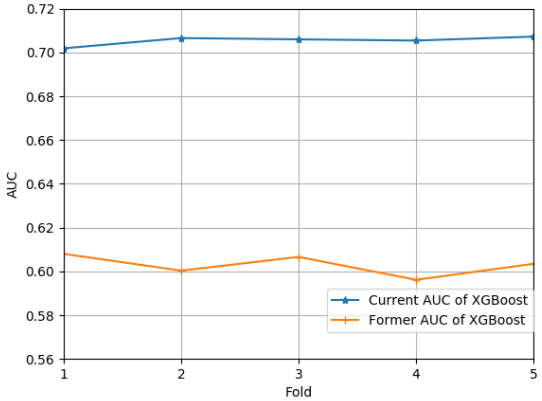


From the comparison of AUC curves, the advantage of decision tree models is obvious.

6 Improvement after Feature Engineering

We compare the performances of KNN and XGBoost models before and after feature engineering. The comparison of AUC curves for KNN and XGBoost are the following:





From the above comparison, the improvement after the feature engineering is obvious, especially for XGBoost model.

7 Conclusion

In this report, we explore the performances of extreme gradient boosting algorithm (XGBoost), KNN, SVM and light gradient boosting machine on KKBox's Music dataset.

The first advantage of decision tree model compared to others is interpretability. Decision trees are "white boxes" in the sense that the acquired knowledge can be expressed in a readable form, while KNN, SVM are generally black boxes. Secondly, note that KNN and SVM are used for continuous value inputs, unlike Decision Trees that is applicable for continuous and categorical inputs. Since we are dealing with a problem where inputs are categorical values, decision tree model is more appropriate.

During data preprocessing, we use label encoder to deal with our categorical values. Label encoding simply converts each value in a column to a number. The nice aspect of this approach is that we can get compact data size, and can order the categories with ease.

Feature Engineering plays an extremely important role in our project. When we prepare a data matrix for modeling, not all columns are useful in their raw form, such as "artist name" and "isrc" in our raw data set. In this project, we applied feature generation and feature selection during the feature engineering step. Feature generation is the process of manually constructing new attributes from raw data. For example, "artist name" in raw data is meaningless for our model, but we generated a new feature 'count_artist_played' by counting number of songs belonging to each artist (since artists with more songs tend to be more popular). Adding new features enables our model to learn better. During feature selection, we rank the attributes by their predictive ability. XGBoost library can automatically rank the importance of features. However, instead of simply preserve the most importance ones, we used a greedy method to select features which has guarantee to increase the performance of model. Feature engineering improves XGBoost model performance (AUC) from 0.61 to 0.70.