Spotify Top Track Prediction

Rachit Agarwal
ragarwa7@ncsu.edu

Tsung-Hsun Hsieh
thsieh4@ncsu.edu

Yu-Lun Hong
yhong3@ncsu.edu

1 Background

Music streaming services, such as Spotify, Google Music, Apple Music, and Pandora, have become a trend and changed the way of how people listen to music. Thanks to their availability and convenience, nowadays people can play new songs anytime and anywhere. In fact, the user habits have changed and listeners are more sensitive to the popular songs than before, and users now expect the service to keep them up with the trends. In order to engage the audience and advertise new songs, those streaming services have started to analyze tremendous amount of data collected from their users, try to predict the songs that people will like based on music features, and recommend the predicted songs to the user. This new habit of listening music has inspired us to study the possible techniques for song prediction as well as recommendation.

In digital market, music is a big industry. According to Global Music Report 2017 from IFPI [4], in 2016, the total revenues in music industry was US$15.7 billion, and there were 112 million users of paid music streaming subscriptions. Hence, before releasing a new album or EP, a music company would eager to know whether the songs in an album will be a hit or not. If there is a great chance that songs in an album will be a great hit, the music company will be more willing to invest in this album since it will be profitable. For this reason, a model that is capable of predicting whether a song will be a hit is crucial in music industry. In our project, we are trying to build such model based on musical features of past top and non-top songs using three different supervised machine learning techniques. To be specific, we are interested in answering the question: whether a given song will be hit or not. In practice, this is a binary classification problem.

In addition to top song prediction, knowing what musical elements make a song popular will help the music companies knowing the trending music style to advertise. Therefore, we also want to figure out those significant musical elements, or musical features, in this project. Finally, from the concept of Occam's razor, a simple model is more favorable than a complex one, so we will try to simplify the models at the end.

In previous work, Herremans et al. [3] used Naive Bayes, Support Vector Machine (SVM) with Radial Basis Function (RBF) Kernel, Logistic Regression and C4.5 tree as the binary classifiers in top dance song prediction. One of the advantages of their work is that they provides details about the research such as data collection (from Echo Nest), preprocessing, description of the techniques used. However, there are also several weaknesses. First, their dataset is relatively small (400 songs, 21 features), therefore the sample data has a higher chance to be non-representative. Such dataset will lead to less robust models. Second, not all of the features are useful in prediction, and hence a feature selection is needed. Finally, they focus on dance songs only, but ours concentrate on at least 35 different genres of songs. To demonstrate the performance difference, we will compare our results with the previous works in the conclusion section.

Software used: Python with libraries (NumPy, pandas, SciPy, Matplotlib, scikit-learn, spotipy). Jupyter Notebook.

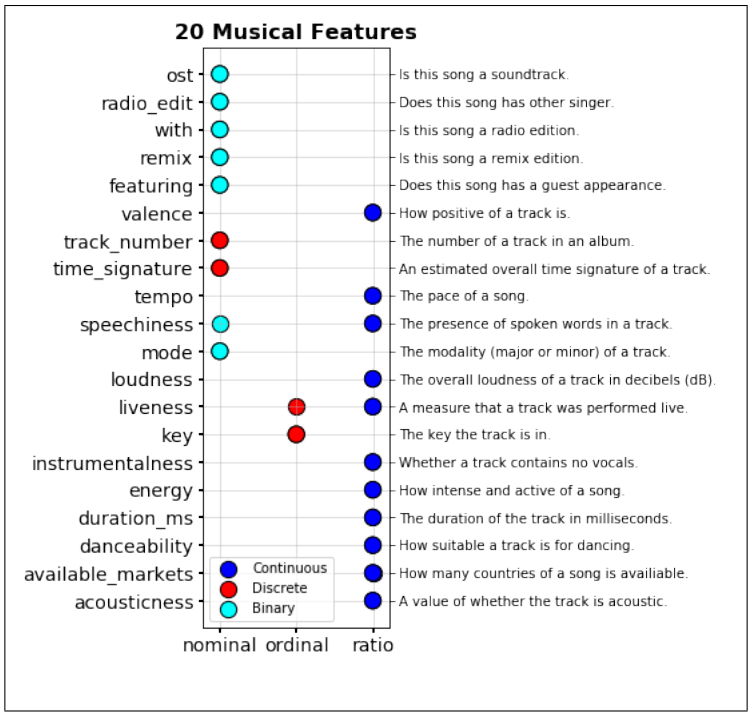

Figure 1: Musical Features and Description

2 Method

2.1 Data Collection

In this project, the data of top songs (y=1) is collected from the charts of Shazam [6] Global Top 100 and Spotify [8] Top 200 in 2017. On the other side, non-top songs (y=0) are randomly chosen across different genres from Spotify. Each song or track contains 20 different types of musical features (See Figure 1). Here, 15 out of 20 features are provided by Spotify [7] Web API, and we used spotipy, the Python package written by Lamere [5], to query these audio features automatically. The remaining 5 musical features were generated by ourselves using simple text mining. The description of all musical features are also summarized in Figure 1. At this point, our self-collected dataset contains 10809 observations (top songs: 7700; non-top songs: 3109) and 20 features. One noteworthy fact is that our data is imbalanced (top songs: 20%; non-top songs: 80%). Therefore, a suitable evaluation scoring metric is needed to quantify the binary classification performance. According to Fawcett [2], Receiver Operating Characteristic (ROC) Curve and Area Under ROC Curve, or ROC-AUC for short, are much proper for the evaluation of imbalanced data when comparing with accuracy. However, we still list accuracy in our final result for reference.

2.2 Classification Techniques

Although the supervised learning algorithms we selected are similar to the reference works, there are no specific reasons why the authors chosen those classifiers in their works. For us, one of the reasons for selecting those algorithms is to compare our results with theirs. The other reasons are briefly described below for each models.

2.2.1 Logistic Regression

The first classification model we utilized is Logistic Regression, which is an adaption of Linear Regression, and there are several reasons to choose Logistic Regression to start with. For example, Logistic Regression is relatively fast in the process of training and prediction. More importantly, it can provide us with the information about how significant our features are, and this would be helpful for future feature selection. Practically, Logistic Regression has different variations, and we chose the

one with L2 regularization in this project. Regularized Logistic Regression has one hyperparameter, C, to control the model complexity: the lower the value of C, the higher the regularization of the model is.

2.2.2 SVM with RBF Kernel

Secondly, we used SVM with RBF kernel as our nonlinear classification model since RBF is very common in SVM. On the other hand, Polynomial kernel (polynomial features) is often used in Logistic Regression . The complexity of SVM is determined by two hyperparameters, C and σ. For the purpose of model comparison, we choose kernel SVM because it is a nonlinear classifier while our first model is linear.

2.2.3 Random Forest

Random Forest is an ensemble learning method, which is used to build a strong classifier by combing a series of weak learners based on bootstrapping. It has lots of hyperparamters for tuning, such as *number of decision estimators (decision trees)*, *maximum depth of each tree*, *maximum number of features*, etc. To simplify the experiment, only the *number of decision trees* and the *maximum tree depth* were considered when tuning a model. Conceptually, ensemble learning methods should have the best performance due to their natural mechanism.

2.3 Experiment

2.3.1 Data Preprocessing

The first reasonable procedure of data cleaning is to remove the duplicates in our data since most of the top songs will remain on the top chart for several weeks. After dropping duplicates, we have 764 songs in the top set and 3015 songs in the non-top set. Secondly, instead of imputation, we dropped the observations whose features contain missing values, because only 11 observations containing missing values. Up to this point, we have 763 songs as top songs and 3005 as non-top songs (See Table 1).

Table 1: Summary of data preprocessing

	Top Songs	Non-Top Songs
Without Processing	7700	3109
Removing Duplicates	764	3015
Dropping missing values	763	3005

2.3.2 Feature Generation

In addition to the 15 audio features extracted from Spotify [7] Web API, we further generate 5 features for each song based on their song title using simple text mining. Basically, if there are some keywords appear in one song title, then we say this song has certain features. For example, *Rockabye (feat. Sean Paul Anne-Marie)*, a song having two guests *Sean Paul* and *Anne-Marie*, will have musical feature *featuring* being 1. With the similar approach, we generated 5 additional features, *featuring*, *remix*, *radio_edit*, *with*, and *ost* (Figure 1).

2.3.3 Exploratory Data Analysis (EDA)

As mentioned above, there are 20 features contained in our dataset. To get a high-level understanding of our data, we conduct simple EDA in this section. Since the features like *acousticness*, *danceability*, *energy*, *instrumentalness*, *liveness*, *speechiness*, and *valence* have values ranged within 0 and 1, it is easier to analyze them together as a group. Shown in the boxplot below (Figure 2), we can see that most of the features, except for *liveness*, have quite different distribution between the top and the non-top. For example, *instrumentalness* for top songs has very dense distribution near value 0, this gives us a clue that top songs should have more vocal content.

The next set of features are *available_markets* and *track_number*, shown in Figure 3, due to the similar range of values. It seems that *track_number* is more distinguishable between the top and

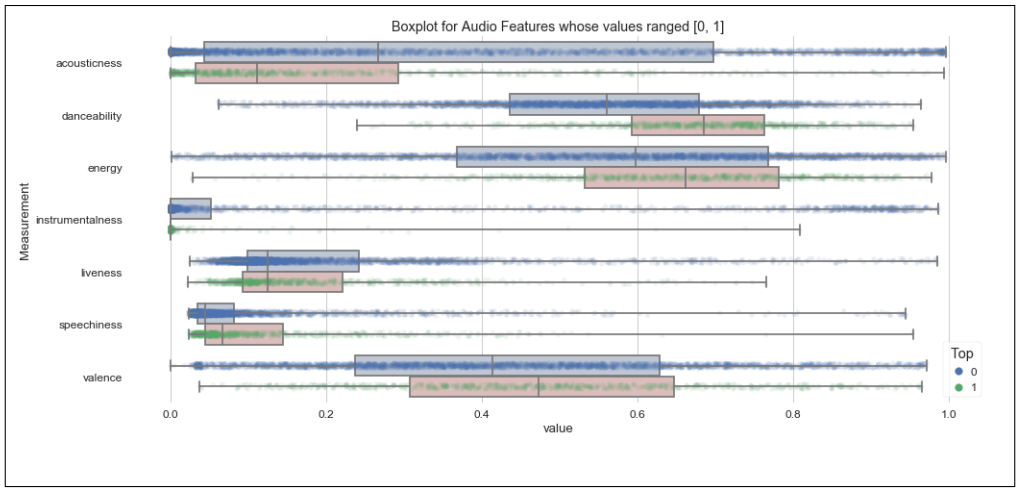

Figure 2: Boxplot of Six Musical Features

non-top songs. Alternatively, *available_markets* for both top and non-top songs have quite similar distribution.

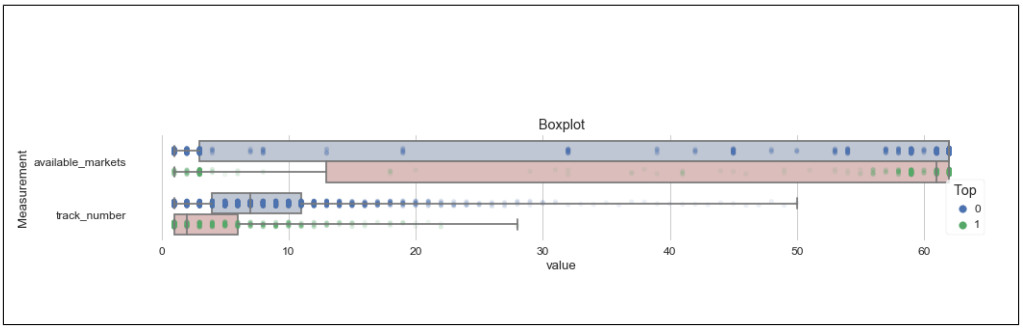

Figure 3: Boxplot of Two Musical Features

Here, we only show parts of EDA because we have as many as 20 features. After simple graphical analysis, we know that features such as *acousticness*, *danceability*, *energy*, *instrumentalness*, *speechiness*, *valence*, *track_number*, *track_numbers* and *loudness* could be representative and thus they could be a good feature for our binary classification problem. In contrast, features including *liveness*, *available_markets*, *key* and *tempo* are not that distinctive, and could be dropped once we want to reduce the model complexity.

2.3.4 Cross Validation and Grid Search

Before building classifiers, the dataset is split into training set and test set. In detail, we randomly pick 30% of the data as the unseen test data using stratified sampling due to the imbalanced distribution of classes. The rest 70% of the data is used for model building and model selection. Both 30% test set and 70% training set are standardized based on the 70% training set, because feature scaling can improve the performance of our Logistic Regression and SVM models. We use the method of Grid Search with five-fold cross validation to train and validate the models, since Grid Search allows us to select the best model with the corresponding hyperparameters by trying all of the specified values for a classifier using training set. In addition, our models are optimized according to AUC or accuracy. For instance, we select 2 models for each of the three machine learning techniques, one outputs the best AUC score, and the other ouputs the best accuracy (with their own best hyperparameters). After that, the remaining 30% unseen data is fed into those trained classifiers for performance evaluation.

Table 2: Model Performance and Hyperparameters (20 Features)

Model	Accuracy	Best Hyperparameter	AUC	Best Hyperparameter
Baseline	0.7975	NA	0.5000	NA
Logistic Regression	0.8453	C: 1.0	0.8518	C:1.0
SVM - RBF kernel	0.8621	C: 0.1, γ: 0.1	0.8475	C: 10.0, γ: 0.001
Random Forest	0.8621	Max. tree depth: 13, Num. of tree: 140	0.8654	Max. tree depth: 9, Num. of tree: 230

Table 3: Model Performance and Hyperparameters (14 Features)

Model	Accuracy	Best Hyperparameter	AUC	Best Hyperparameter
Baseline	0.7975	NA	0.5000	NA
Logistic Regression	0.8479	C: 10.0	0.8537	C: 1.0
SVM - RBF kernel	0.8629	C: 10.0, γ: 0.1	0.8391	C: 100.0, γ: 0.001
Random Forest	0.8541	Max. tree depth: 15, Num. of tree: 150	0.8567	Max. tree depth: 9, Num. of tree: 230

2.3.5 Feature Selection

Although we can use all the features we have collected to build the classifiers, we want to make our classifiers as simple as possible in the end. In practice, feature selection does not only simplify the models, but also save us a significant amount of work when collecting data. Therefore, in the result section, we also have the result using only certain representative features for building the classifiers. The way we select the features is described in the result section based on the trained coefficients of Logistic Regression.

3 Results

The experimental results are summarized in Table 2, Table 3, Figure 4 and Figure 5. Noted that the Table 2 is the results using all of 20 features while Table 3 using only 14. We will explain the reason why we use these 14 features later. Clearly, for all of the three classifiers, their accuracy are higher than the baseline model 0.7975, which is calculated as the ratio of the majority labels (non-top songs) to the total number of data (non-top songs plus top songs). This shows that our models beat the baseline model and are great classifiers in general.

Looking into each model respectively, the AUC scores of the three models using 20 features (Table 2) have the similar performance, and are much better than the random guessing (0.5). For Logistic Regression model, although its AUC score is slightly lower than the Random Forest one, it does provide several advantages over Random Forest. Firstly, as shown in Figure 6, it outputs the weight of each feature, which gives us an idea about the relationship between features and class prediction. To be specific, as a weight gets close to zero, this feature becomes less important in our prediction. Using the feature *danceability* as an example, if a given song has a higher value of *danceability*, then this song has a higher probability to be a hit song. This result is in agreement with what we have found earlier in our EDA (Figure 2). Based on the value of weights, we can remove those less significant features (e.g. *key*, *mode* etc.) that contributes little to the prediction.

SVM with RBF kernel has the same accuracy as the Random Forest model, and it has higher accuracy than the Logistic Regression model since it is nonlinear while Logistic Regression is linear. However, kernel SVM has lower AUC score than Logistic Regression. To interpret this result, generally SVM is not good at probability prediction, and AUC is basically constructed and related to the output probabilities of a model. If accuracy is what we concern, then SVM with RBF kernel will be a better choice than the Random Forest model due to SVM's relatively low complexity and training speed.

Finally, Random Forest produces the highest test AUC score. Similar to Logistic Regression, Random Forest can also show us the importance of each feature (shown in Figure 7). However, unlike the weight of Logistic Regression, feature importance of Random Forest does not point out whether a feature will increase or decrease the possibility of a song to be hit, but only the importance when

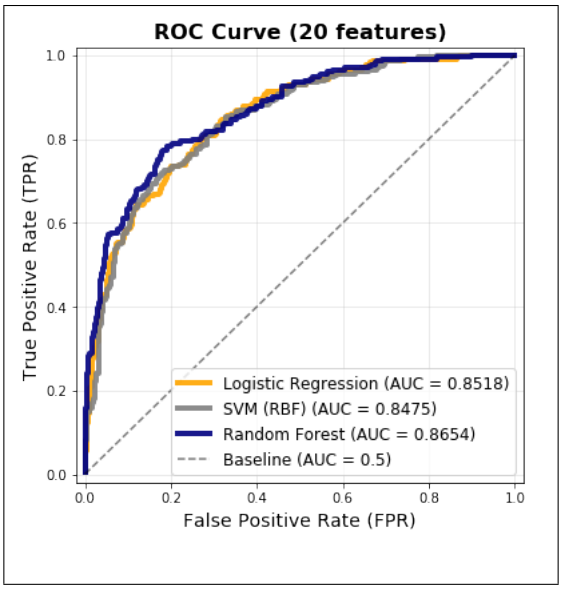

Figure 4: ROC curve & AUC for different classifiers using 20 features

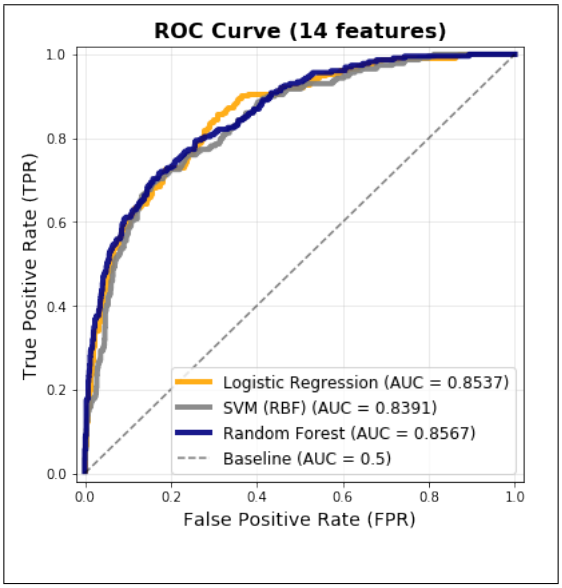

Figure 5: ROC curve & AUC for different classifiers using 14 features

building a model. Perhaps the most significant point to note is that, in order to generate such a high AUC score, Random Forest must used as many as 230 decision trees with max depth of 9 to build a classifier. Compared to Logistic Regression and SVM with RBF, Random Forest not only takes longer time in model training, but also increases the complexity of a model. In fact, our Random Forest model has simply 0.0136 higher AUC than the Logistic Regression model, but has much higher model complexity and time complexity. Therefore, if time, memory, and interpretation are key concerns in an application, it might be more reasonable to adopt Logistic Regression instead.

As mentioned in background section, other than building a prediction model, we also want to simplify the model while keep the similar model performance at the same time. Moreover, we aim to know which musical features will more likely make a song be a hit. According to Figure 6, we can see that *available_markets*, *key*, *mode*, *tempo*, *remix*, and *radio_edit* have weight (absolute value) less than 0.1 and are close to zero. Since the value of coefficients has maximum value 1 and minimum value -1, and we use 0.1 (10%) as an threshold to drop features. Basically, such low value represents

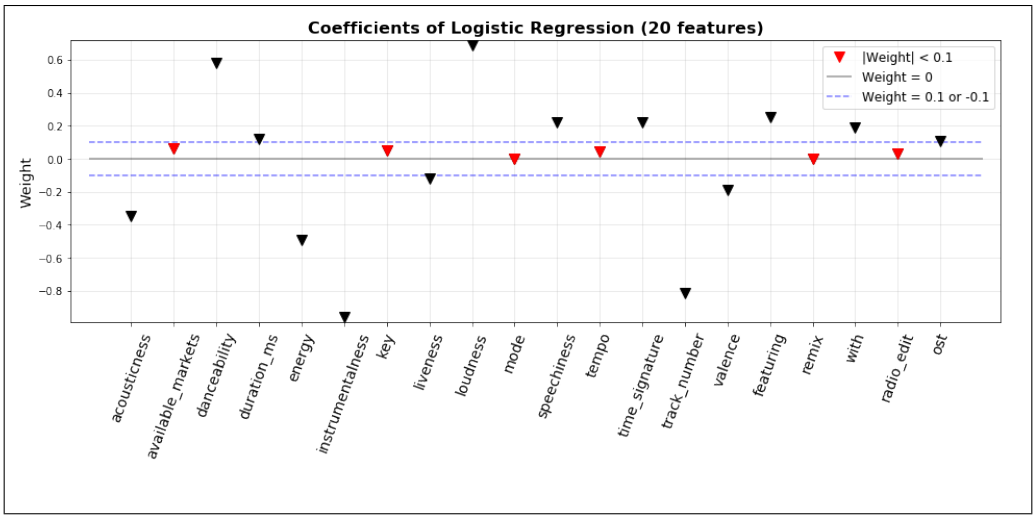

Figure 6: Logistic-Regression - weight of 20 features.

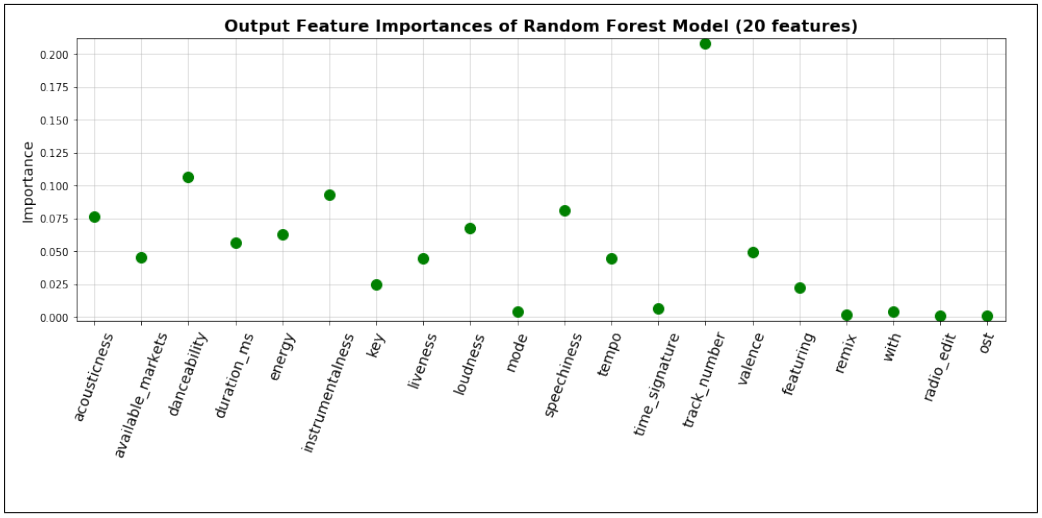

Figure 7: Random Forest - importance of 20 features.

that the feature contribute little to the top song prediction. Therefore, we removed those 6 musical features to simplify the models. The results are shown in Table 3 and Figure 5. Compared to Table 2, all of the three classifiers produced similar results, which proves that those 6 dropped features are not significant in our problem. More importantly, we succeed in reducing the model complexity from the aspect of feature selection by 30%.

Table 4: Comparison of Model Performance in AUC

Model	Reference Results	Our Results	
	400 observations 21 features	3768 observations 20 features	14 features
Logistic Regression	0.81	0.8518	0.8537
SVM - RBF kernel	0.57	0.8475	0.8391
C4.5 / Random Forest	0.62	0.8654	0.8567

Table 4 shows the difference of AUC score between our results and the reference work. Clearly, our classifiers are much better than the previous works in general, even with the simplified models with

only 14 features. The main reason is that our dataset covers at least 35 kinds of genres, including *Hip-Hop*, *Latin*, *R&B*, and even *K-Pop*. This makes our dataset more representative over the whole population, due to the fact that nearly all types of songs have been listed in Top song charts in reality.

4 Conclusion

In this project, three different machine learning techniques have been utilized in the prediction of Top Songs, including Logistic Regression, SVM with RBF kernel, and Random Forest. The highest AUC achieved is 0.8654 using Random Forest, with 230 decision trees based on 20 features. Although Logistic Regression model did not produce the best result, its high interpretability is suggestive of the relation between data and features. Most importantly, the Logistic Regression model provides us with a feature selection tool to simplify the models by reducing the number of features down to 14 while maintaining the similar performance.

Another interpretation of the results is that, in order to make a hit song, the *danceability* and *loudness* should be high while the *instrumentalness* and *track_number* should be as low as possible. The other features such as *mode* and *remix* are not very important.

To further improve our models, we could try to generate more representative features. This can be achieved by analyzing the lyrics, album image, or album profile of a song. For Logistic Regression specifically, multicollinearity could be an issue when building a classifier. However, this is solvable by decorrelating the data using PCA or manually selecting features. As to SVM, several different format of kernels, such as polynomial kernel and sigmoid kernel, is also worth trying. Finally, for Random Forest, we could improve the performance by fine tuning more hyperparameters, such as *minimum number of samples at a leaf*, when building a model. Furthermore, we have evaluated our models mainly based on AUC. However, Davis and Goadrich [1] mentioned that Precision-Recall (PR) curve gives us a better intuition about models. Therefore, we can also take PR curve into account when doing model evaluation in the future work.

GitHub Repository: https://github.com/thsieh4/CSC522_project

References

[1] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143874. URL `http://doi.acm.org/10.1145/1143844.1143874`.

[2] T. Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. *Intelligent Enterprise Technologies Laboratory*, 31:1–38, 2003. URL `http://www.hpl.hp.com/techreports/2003/HPL-2003-4.pdf`.

[3] D. Herremans, D. Martens, and K. Sörensen. Dance hit song prediction. *Journal of New Music Research*, 43(3):291–302, 2014. doi: 10.1080/09298215.2014.881888. URL `http://dx.doi.org/10.1080/09298215.2014.881888`.

[4] IFPI. Ifpi global music report 2017, 2017. URL `http://www.ifpi.org/news/IFPI-GLOBAL-MUSIC-REPORT-2017`.

[5] P. Lamere. Welcome to spotipy!, 2017. URL `https://spotipy.readthedocs.io/en/latest/`.

[6] Shazam. Shazam global top 100, 2017. URL `https://www.shazam.com/charts/top-100/world`.

[7] Spotify. Get audio features for a track, 2016. URL `https://developer.spotify.com/web-api/get-audio-features/`.

[8] Spotify. Spotify top 200, 2017. URL `https://spotifycharts.com/regional`.