Can Spotify predict if you'll skip a song?

Spotify Sequential Skip Prediction

Sheena Tan

2024-03-13

Table of contents

Github Repo Link

https://github.com/stat301-2-2024-winter/final-project-2-sheena-tan

1 Introduction

Spotify, an online music streaming platform with over 190 million active users and over 40 million tracks, faces a key challenge: how do you recommend the right music to right user?

Out of a robust body of literature on recommendation systems, little work describes how users sequentially interact with recommended content. This gap is particularly evident in the music domain, where understanding when and why users skip tracks serves as crucial implicit feedback. The Music Streaming Sessions Dataset was released by Spotify in 2018 to encourage research on this overlooked aspect of streaming.

This project hopes to provide a solution to Spotify's challenge: **Predict whether individual tracks encountered in a listening session will be skipped by a particular user.**

The data set provides information about each user's listening session, including metadata and acoustic descriptors for all tracks encountered. This project uses an original target variable, `skipped`, which is a modified version of the original challenge's `skip_2` field: `skip_2` is a boolean variable (`TRUE`/`FALSE`) and `skipped` is a character variable (`yes`/`no`).

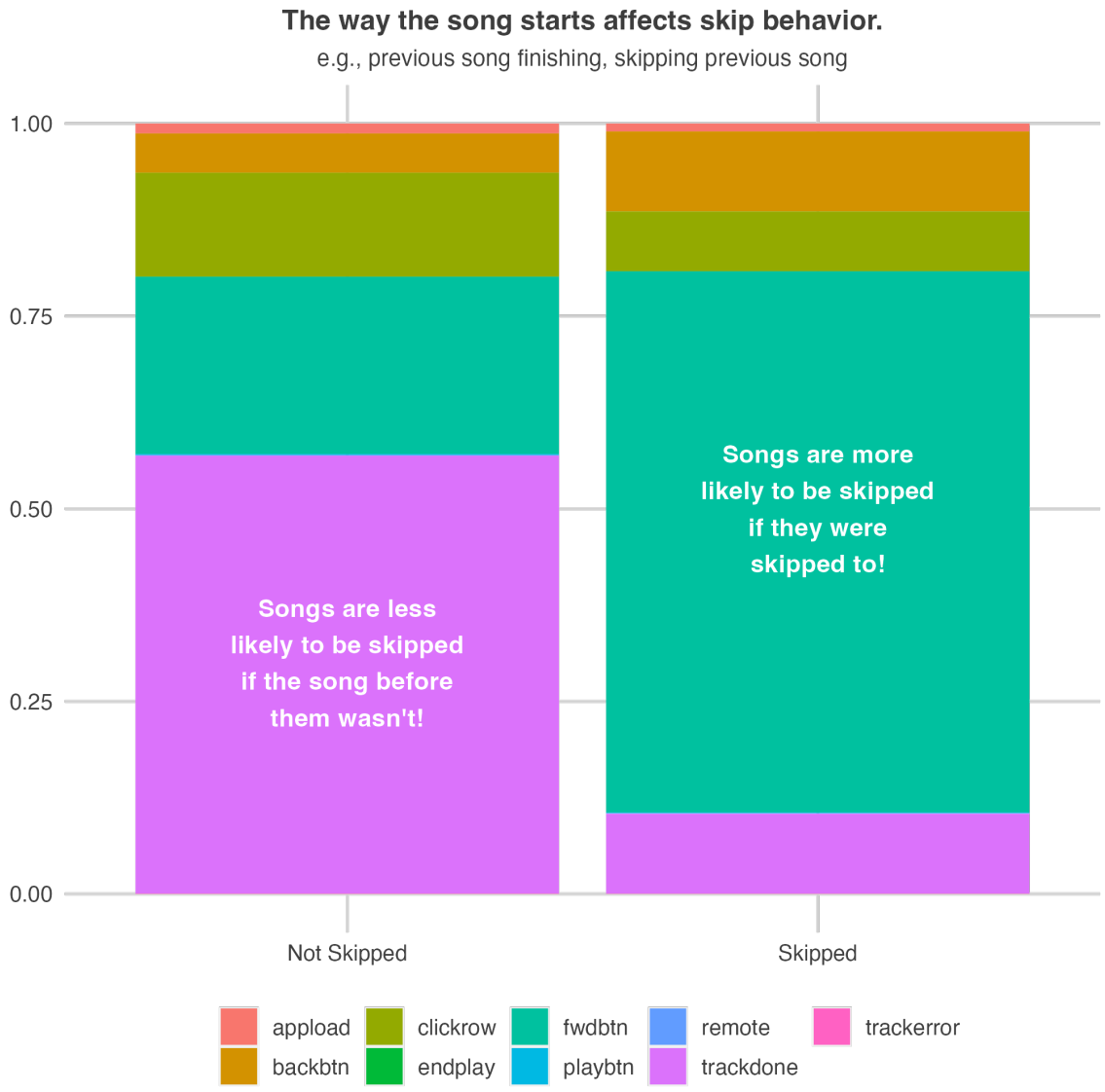
2 Data Overview

Given laptop constraints, this project uses the miniature version of the Music Streaming Sessions Dataset, which is a minimally sized version of the original training set and track features provided by Spotify for challenge users to familiarize themselves with the data set.

The complete data set contains 167,880 observations of 48 variables. Each observation corresponds to the playback of one track, for a total of 167,880 tracks listened to during 10,000 unique listening sessions, representing 50,704 unique tracks.

Each listening session contained information on the order that tracks were listened to and whether or not the track was skipped. Each track contained information on over 25 of its different properties, such as its popularity; musical classifications like key, time signature, and tempo; or Spotify-specific classifications like danceability, bounciness, speechiness, or liveness. For more information on the data, please refer to the `README.me` codebook.

Figure 1: The way a song was started differs by whether or not it was skipped



Exploratory data analysis demonstrated correlations between skip behavior and several other listening variables, like the way a track was started (*above*). Analysis of the target variable revealed no major issues and proportions of songs skipped or not skipped were approximately even. There was no missingness in any of the variables. See Appendix A for full EDA.

Overall, initial explorations support the viability of a predictive model.

3 Methods

3.1 Assessment metric

As the prediction problem is a *classification* problem, models will be compared using **accuracy**, and a confusion matrix will be produced for the selected final model.

3.2 Model overview

An 80-20 training-test split ($N\_train = 134303$; $N\_test = 33577$) was implemented using stratified sampling by target variable (2 strata). Resamples were constructed by taking the training dataset and applying repeated V-fold cross-validation (5 folds, 3 repeats) with stratification on the target variable with 2 strata.

The following model types were created with hyperparameters tuned using a regular grid:

- Naive bayes baseline model (`klaR` engine)
- Logistic model (`glm` engine)
- Neural network model (`nnet` engine)
  - Number of hidden units was explored over $[1, 10]$ with 5 levels
  - Penalty was explored over $[-10, 0]$ with 5 levels (on log-10 scale)
  - Number of epochs over $[10, 1000]$ with 5 levels
- K-nearest neighbors model (`kknn` engine)
  - Neighbors were explored over $[1, 15]$ with 5 levels
- Random forest model (`ranger` engine)
  - Number of randomly selected predictors to split on were explored over $[1, 47]$ with 5 levels
  - Minimum number of data points in a node for splitting were explored over $[2, 40]$ with 5 levels
- Boosted tree model (`xgboost` engine)
  - Number of randomly selected predictors to split on were explored over $[1, 47]$ with 5 levels
  - Minimum number of data points in a node for splitting were explored over $[2, 40]$ with 5 levels
  - Learning rate was explored over $[-3, -0.5]$ with 5 levels (on log-10 scale)

The naive bayes model used the preprocessing stored in `spotify_recipe_naive_bayes`. Variables used to define the target variable and identifier variables (`skip_1`, `skip_2`, `skip_3`, `not_skipped`, `session_id`, `track_id`, and variables with zero variance) were removed and all numerical variables were standardized.

The logistic model and neural network models used the preprocessing stored in `spotify_recipe_lm`. Variables used to define the target variable and identifier variables (`skip_1`, `skip_2`, `skip_3`, `not_skipped`, `session_id`, `track_id`, and variables with zero variance) were removed, nominal predictor variables were dummy coded, and all numerical variables were standardized.

The k-nearest neighbors, random forest, and boosted tree models used the preprocessing stored in `spotify_recipe_tree`. Variables used to define the target variable and identifier variables (`skip_1`, `skip_2`, `skip_3`, `not_skipped`, `session_id`, `track_id`, and variables with zero variance) were removed, nominal predictor variables were one-hot dummy coded, and all numerical variables were standardized.

4 Model Building & Selection Results

The naive bayes model was defined and fitted as a baseline model. The logistic model was also fitted. The logistic model had a slightly higher accuracy than the baseline model with a lower standard error than the baseline model, suggesting that more complex models would be worthwhile to explore.

The neural network model, k-nearest neighbors model, random forest model, and boosted tree model were then fitted. All trained models were compared using accuracy to determine the final model. Tuning parameters were compared for the support vector machine, k-nearest neighbors, random forest, and boosted tree models. A table containing the best performing parameters for each model can be seen below.

Figure 2: Performance and best parameters across model types

Best performing model results

Neural network and **boosted tree** models performed best.

model	accuracy	std_err	parameter	value
nnet	0.87773	4.2538×10^{-4}	hidden_units	5
nnet	0.87773	4.2538×10^{-4}	penalty	1
nnet	0.87773	4.2538×10^{-4}	epochs	1,000
bt	0.87767	3.0043×10^{-4}	mtry	47
bt	0.87767	3.0043×10^{-4}	min_n	11
bt	0.87767	3.0043×10^{-4}	learn_rate	0.3162278
rf	0.87728	2.7327×10^{-4}	mtry	12
rf	0.87728	2.7327×10^{-4}	min_n	40
lm	0.87427	3.7033×10^{-4}	NA[1]	[1] NA
baseline	0.85954	7.7444×10^{-4}	NA[1]	[1] NA
knn	0.84917	2.9637×10^{-4}	neighbors	15

[1] No parameters tuned

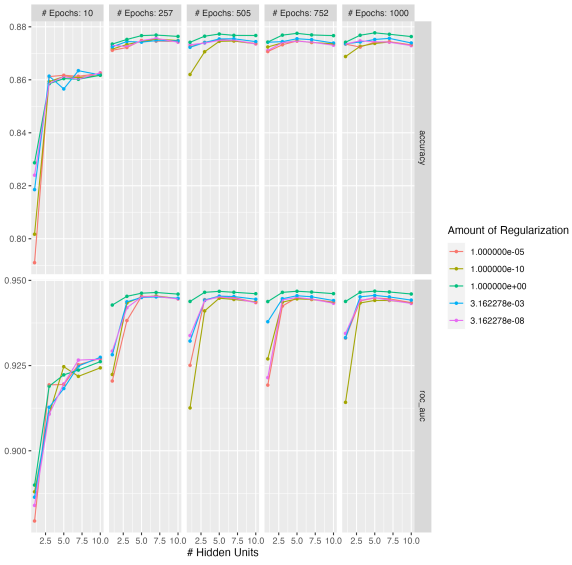
The neural network model (hidden_units = 5, epochs = 1000, penalty = 1) performed the best. As shown in Figure 3, other than when the epoch hyperparameter was low (epochs = 10), the neural network model had consistently high accuracy (above 0.86) across variation in hidden units and regularization. Further tuning of the number of hidden units hyperparameter may be worthwhile to explore. Accuracy appears to be highest within a range of values between 5 and 7.5, but increasing the number of levels could help to narrow in on where the peak performance actually is.

Notably, the boosted tree model (mtry = 47, min_n = 11, learn_rate = 0.316) performed almost as well as the neural network model - within one standard deviation. However, as shown in Figure 3, accuracy improved as learning rate and number of randomly selected predictors increased, reaching the highest accuracy at the highest learning rate (learn_rate = 0.316) and number of randomly selected predictors (mtry = 47) possible. These results suggest that, unlike the neural network model, there is not much room for this model to be improved upon by further tuning hyperparameters.

Calculating the mean model accuracy across all explored parameters reveals that the logistic and random forest models are also strong choices.

Figure 3: Parameters explored across best performing models

(a) Neural network model performance across hyperparameters



(b) Boosted tree model performance across hyperparameters

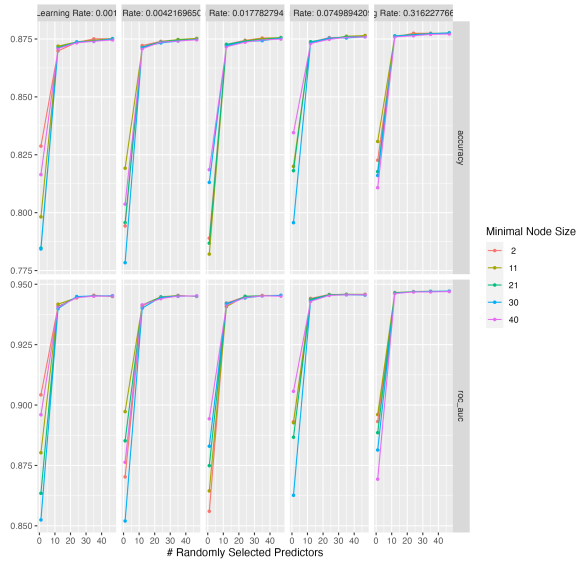


Figure 4: Performance averaged across all parameters by model

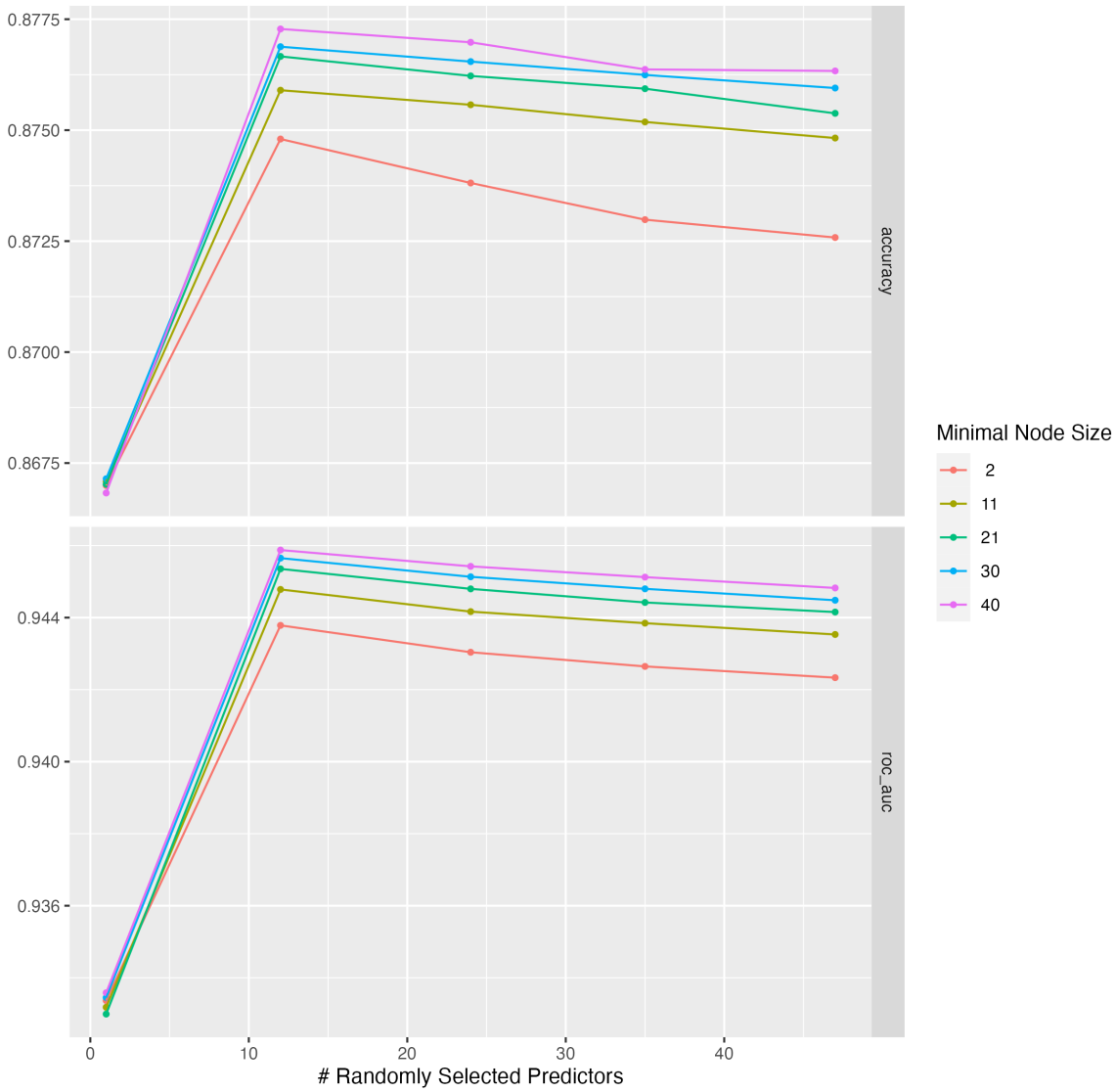
Mean model accuracy across parameters

Logistic and **random forest** models performed best.

model	mean	std_err
lm	0.87427	3.7033×10^{-4}
rf	0.87390	2.9749×10^{-4}
nnet	0.86958	1.9090×10^{-3}
bt	0.86118	3.3325×10^{-3}
baseline	0.85954	7.7444×10^{-4}
knn	0.82837	3.9884×10^{-4}

As shown in figure 5, further tuning of the number of randomly selected predictors hyperparameter may be worthwhile to explore for the random forest model. After 10 (mtry = 10), accuracy decreases as the number of randomly selected predictors increases, so the best parameter value appears to be within a range between 0 and 10, but increasing the number of levels could help to narrow in on what that value actually is. For example, model performance could be higher at 8 or 9 randomly selected predictors.

Figure 5: Random forest model performance across hyperparameters



Despite this possibility, this project ultimately selects the **neural network model** as the final model. Even though the random forest and logistic models had higher averaged accuracy values, their best performances were still more than one standard deviation away from that of both the neural network and boosted tree models. For behavioral data that is nonlinear and complex, like what makes a song worthwhile for each user to listen to, a neural network model makes for a proper theoretical fit. Neural network models have likewise been shown to be well-suited for developing personalized recommendations from user activity.

5 Final Model Analysis

The final neural network model (hidden_units = 5, epochs = 1000, penalty = 1) was fitted to the testing data. The model correctly predicted **88.1%** of user skip behavior for each track, suggesting that the model was a very good fit.

Table 1: Final model performance confusion matrix

	no (actual)	yes (actual)
no (predicted)	13643	1416
yes (predicted)	2569	15949

The model predicted more false positives ($N = 2569$) than false negatives ($N = 1416$), suggesting that the model tends to predict that people skip more than they actually do. However, the difference is only by about 5.5%, which is not a significant skew. Overall, this predictive model performs significantly better than mere chance (i.e., 50% accuracy). The neural network model's capacity to process complex, nonlinear data in a way that is inspired by the human brain may have ultimately facilitated the prediction of human behavioral activity like musical preference.

6 Conclusion

Given the properties of the songs that a user listens to and information about their listening session, a fairly robust model can be created to predict whether a user will skip a certain song. This model can help to inform music recommendation systems, such that users can be recommended music that they will be less likely to skip.

Interestingly, an additional exploration modifying the model preprocessing to only include either listening session information (e.g., what context they listened to the song in, how they started a song, what position the song was played in that session) or song properties (e.g., danceability, key, tempo) revealed that the model remains ***equally robust*** without information on song properties, but becomes *little better than mere chance* without information on listening sessions.

Table 2: Accuracy of final neural network model modified to include either song properties or session information

song properties only	session information only
0.526	0.879

These findings highlight the possibility of certain key predictors in the session information data that could have been driving model performance, while simultaneously suggesting the difficulty of predicting skip behavior from song properties alone. Future explorations should investigate specific song properties and specific listening session characteristics that are the most influential for skip behavior.

7 References

Spotify. (2018). *Spotify sequential Skip Prediction Challenge: Challenges*. AIcrowd. https://www.aicrowd.com/challenges/spotify-sequential-skip-prediction-challenge

Brost, B., Mehrotra, R., & Jehan, T. (2019, May). The music streaming sessions dataset. In *The World Wide Web Conference* (pp. 2594-2600).

8 Appendix

8.1 Appendix A: EDA

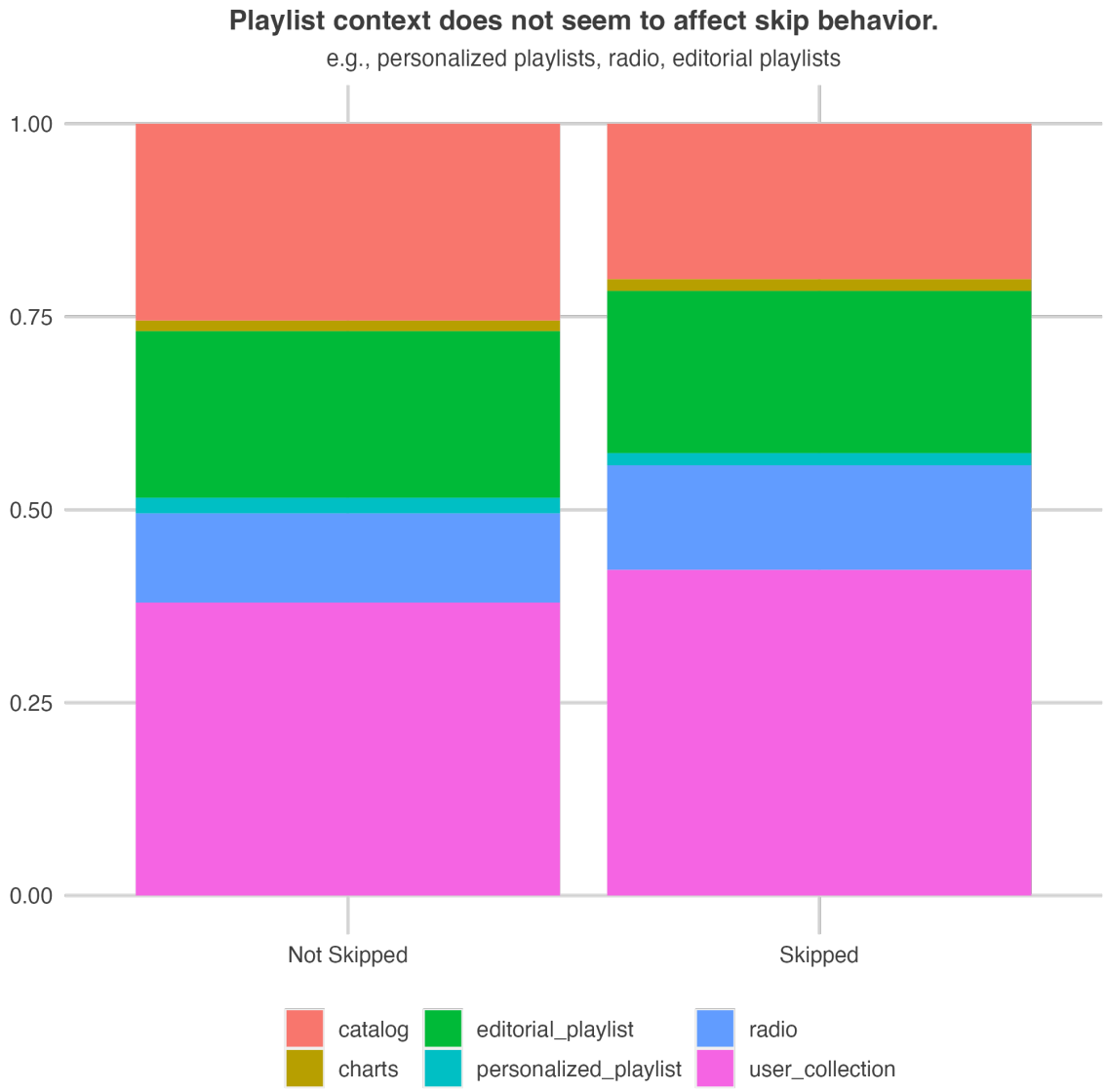
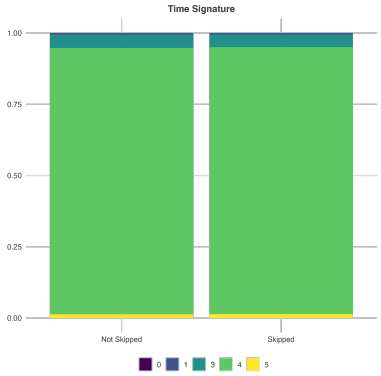
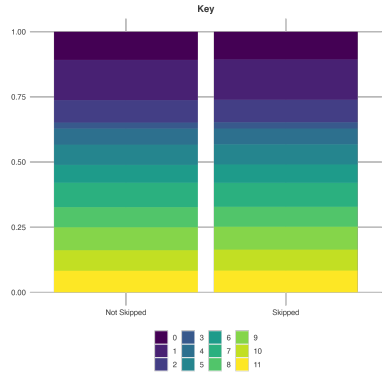


Figure 6: Song properties did not demonstrate differences by skip behavior

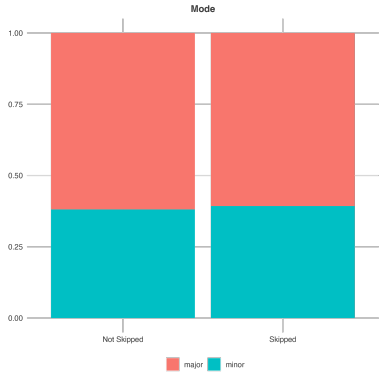
(a) Time signature



(b) Key



(c) Major/Minor



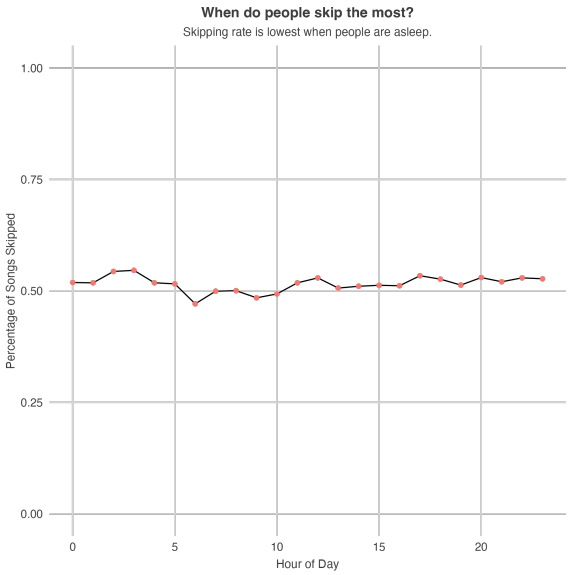


Figure 7: Skip behavior by time of day

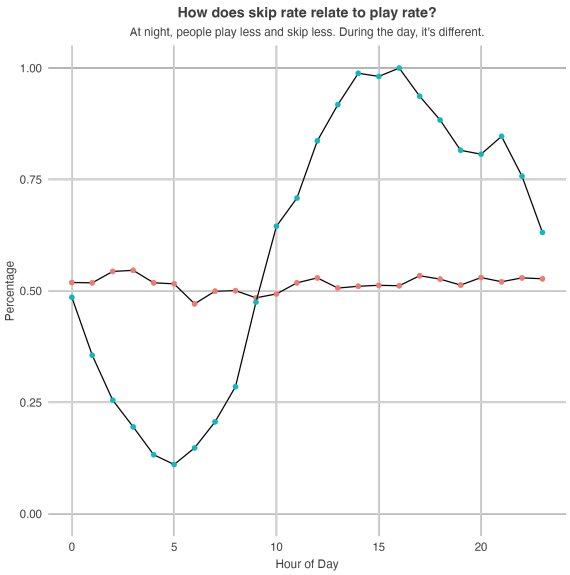


Figure 8: Play behavior by time of day