# Skip prediction using decision trees

Jasmijn Bookelmann
*Radboud University*
Nijmegen, Netherlands
jasmijn.bookelmann@ru.nl

***Abstract*—NOTE THAT THIS IS AN ASSIGNMENT PAPER NOT BASED ON ACTUAL RESEARCH - Skip prediction is predicting whether a user will skip a song or not. This is a good indication of how much the user likes this song. Therefore, it is an important component of recommender systems. Such as the spotify algorithm. We have created a skip prediction model using the spotify sessions database, which contains listening sessions and metadata of the songs in it. Our model predicts based on this initial session whether a user will skip the next song or not. For creating this model, we have first preprocessed the data by summarizing the session into unary variables. After this we apply decision trees to classify this data.**

## I. Introduction

Automatic recommendations are getting increasingly popular with the digitalisation of music. They have an important role in the consumption of music nowadays.

Predicting whether or not a user will skip a song is known as skip prediction. Accurate skip prediction is a tool in automatic recommendation systems, as it is a measure for whether a user will enjoy a song [1]. This can be used in systems such as the playlist creator or autoplay from Spotify.

We evaluate the performance of skip prediction using decision trees.

## II. Background

In 2019 spotify released a session dataset of around 10GB filled with anonimized user listening sessions [2]. We will use this dataset in order to create, train and test our model.

Every record in this dataset has two main parts: The listening session and the track. The goal of our model is to predict whether the track will be skipped based on the listening session and its metadata.

### A. Listening Sessions

The listening sessions contain the tracks in the order the user listened to. These sessions contain up to 20 tracks.

The session records have the following properties: Whether the user has Spotify Premium or not, the action causing the listening session to start, the date, etc.

### B. Tracks

The track entries contain data about their audio features, provided by the spotify API. This is information such as the bounciness, dancability and key.

In addition, the duration, popularity and release year. A track is skipped if a user did not listen to the entire track. There are three metrics which measure this:

- `skip_1`: The track was only played very briefly
- `skip_2`: The track was only played briefly
- `skip_3`: Most of the track was played
- `not_skipped`: The track was played in its entirety

We use `skip_2` as ground-truth.

## III. Methods

We reduce our research question to a classification problem for our model. This question is formulated as follows: Assuming we have session data containing 9 songs and the track information of the 10th song, classify this track as 'skipped' or 'not skipped'.

We first preprocess the data such that it can be used to train our model. After this we find the optimal parameters for our model using k-fold cross validation. We use smaller subsets of the data with roughly 100.000 entries to train and test our model as we have limited resources.

### A. Preprocessing

In this section we discuss the features of every record which are used by our data mining model. In section A we discuss the features of the current track, in B and C we discuss how we summarize the data from the session.

*1) Track Features:* As mentioned in the Background section, every track has 21 audio features. As seen in table 3.1 the ranges of track features differ quite a lot. We have standardized this data, which avoids certain track features with larger ranges having larger impact on our model.

In addition to the audio features, there is also the `duration`, `release_year` and `popularity`. We do not change this data.

*2) Session Skips:* Our models do not take sequential data into account, thus we summarize the session skips into single features. These are the features of every track in the session indicating whether the user skipped this track or not. In order to use this for calculations we need to convert the skip to a number. We use `skip_2` as indicator of whether a user skipped or not in a track, just like the ground-truth. If `skip_2` is true, then our skip number equals 1, otherwise is 0.

In order to summarize this we create two features:

- `%_skip`: The percentage of skips. E.g., if a user has skipped 3 out of 10 tracks `%_skip` is equal to 30%
- `sd_skip`: The standard deviation of the skips. So if a user has skipped none of the tracks `sd_skip` is equal to 0. Or if a user has skipped 3/10 tracks the `sd_skip` is equal to 0.46. This is a measure of how irregular the user's skipping behavior is.

*3) Session Accoustic Features:* In order to summarize the accoustic data of the session tracks we create two features. One feature is the average accoustic data of all the skipped tracks. The other feature is the average accoustic data of all the unskipped tracks. This allows us to see the average information of the preferenced and unpreferenced tracks.

*4) Transforming other features:* In order to make the feature more representative of the data we also adjust the following features:

We reformat the date to represent the 7 days of the week instead of an absolute date. For example July 31st 2020 is rewritten to Friday. This will reduce the entropy of this feature, which is known to improve the performance of decision trees [1]. In addition, we expect this information to be more indicative of skipping behavior than the absolute date.

Popularity, year of release and duration wil be kept the same.

### B. Sample selection

As the data itself contains millions of tracks, and realistically we are not able to process this much using our limited resources. We select sample of our data to train and test our model. These subsets each contain 100.000 track sessions.

In order to select representative samples we apply stratified data sampling. As there is an uneven distribution of skipped tracks compared to non skipped, using this method we create an evenly distributed dataset. This will avoid pitfalls such as underrepresentation.

We select only one sample to train and select our model on. Then we test this model on multiple samples to ensure it performs the same accross different subsets of the data.

### C. Model Training

*1) Performance Metric:* In order to measure the performance of our model we use the F-measure metric. Here the accuracy is defined as:

$$\text{F-measure(F)} = \frac{2TP}{2TP + FP + FN}$$

Where:

- $TP$ = Cost of a true positive
- $FP$ = Cost of a false positive
- $FN$ = Cost of a false negative

In our accuracy metric we make the assumption that the cost of a false positive (recommending a song a user actually does not like) is worse than the cost of a false negative (not recommending a song a user might like).

From this we create the following cost matrix:
`table of cost matrix`

In this matrix the cost of a false positive is 1.5 times this of a false negative.

*2) Model:* In order to train and apply our model, we use the `DecisionTreeClassifier` from the python library scikit-learn. The advantages of this model are that it is easy to use and gives us a lot of insights in the inner workings of the decision tree. In addition, it has good compatability with plotting libraries such as matplotlib.

In order to optimize the parameters for the decision trees we apply nested k-fold cross validation.

K-fold cross validation is a method which given a $k$, splits the dataset into $k$ groups. Then using one of the groups as training set and the others as test set, it measures the performance of the model. We set $k = 10$, which balances a decent amount of testing with a doable running time.

We apply cross validation for the following parameter ranges:

- max depth: 10, 15, 20, 25
- min samples split: 15, 20, 30
- min weight fraction leaf: 0 1 3 5

In total we trained 48 models to check which parameters are optimal. This resulted in max depth = 15, min samples split = 30, min weight fraction leaf = 0. With a model of 85% accuracy.

## IV. Results

Our model achieved an 85% accuracy using F-measure. And an AUC score of 79%. This indicates that it was able to accurately predict whether a user will skip a song or not in most cases, with a bias towards true positives. When applying our model to other samples the accuracy stays the same, which shows our model performs well on other samples too.

After mapping out the most influential features of our decision trees we see that US Popularity, Dancability and Average Session Skip have been the most influential in the classification.

Overall, our study has demonstrated the effectiveness of using decision tree models for predicting user behavior on Spotify, and has provided valuable insights for improving the user experience on the platform.

`[show plot with most influential features and pe`
`[show plot of ROC curve]`

## V. Discussion

We have found that decision trees are effective when applying to skip prediction. Their flexibility allowed for a good prediction model.

We do have a few limitations: We had to train and test the model on relatively small sets of data due to limited resources. Thus further research could be conducted on larger sets of data to improve accuracy given stronger resources.

In addition, by averaging the data of the sessions we lose a lot of sequential information. We believe sequence-based data training models, such as sequential neural networking, might result in better accuracy.

We believe further research can also delve more into machine learning algorithms using decision trees, such as random forests or boosting trees.

## VI. Conclusion

In conclusion, our study has demonstrated the effectiveness of using decision tree models for predicting user behavior on Spotify. The model was able to accurately predict whether a user would skip a song or not based on the acoustic information of the song and the songs in the listening session, with an average accuracy of 85

## Acknowledgment

I would like to thank Tim Kersten, Ivo Melse, Daniël Mol and Rulian Wang for giving feedback on this paper.

## References

[1] `some relevant article`
[2] `source of dataset`