Session-based Sequential Skip Prediction: Spotify

Bhairavi Muralidharan Parth Sheth Purva Sheth

1 Abstract

Spotify is a leading music streaming service with over 365 million monthly active users with around 70 million tracks that the users are engaging with, valued at $ 41.06 billion. This service is fuelled by its customisation and music knowledge driven by algorithms and not community-created playlists. Our motive is to understand the way users sequentially interact with music. This project will focus on the task of session-based sequential skip prediction, i.e. predicting whether users will skip tracks, given their immediately preceding interactions in their listening session. We describe two types of solution approaches in this paper and analyse them in detail.

2 Motivation

The main challenge for Spotify is to recommend the right music to each user and hence most of the work focuses on Recommender Systems and a little on describing how users sequentially interact with the streamed content they are presented with. In particular within music, the question of if, and when, a user skips a track is an important implicit feedback signal. This is an exciting Supervised Machine Learning problem as it helps us understand intricate behavioural patterns of how users engage with tracks and which of the track features play an important role in this prediction.

3 Related Work

Automatic recommendations play an important role in music consumption on streaming services and this topic is widely studied in the literature [4]. One common approach is to use the implicit feedback information, such as the interactions between the users and the tracks, to generate recommendations. Existing research on music recommender systems has considered a number of related tasks, including Automatic Playlist Generation and Automatic Playlist Continuation. And the methods developed for these tasks can be leveraged in this case. This is because the current task shares many aspects of session-based music recommender systems. Hence, understanding the approaches that are employed to such systems is useful. Recurrent Neural Networks (RNNs) have been shown to work exceedingly well with sequential modeling tasks.

4 Dataset

The Dataset[1] is split into two main parts:

Session features dataset The dataset obtained contains roughly 130 million listening sessions with 21 associated user interactions on Spotify. Each session consists of between 10 to 20 most recently listened tracks per user. While testing, all the user interactions are provided for the first half of the session (first 10 tracks), but only the track ID's are provided for the second half. We processed this data by using a One Hot Encoder on the categorical features and standardised the numerical features.

Track features dataset Each track was characterized by the following 30 features available via the Spotify API: popularity, acousticness (8 different features drawn using Deep Learning methods from the physical wave forms of the tracks), beat strength, bounciness, danceability, mean dynamic range, energy, flatness, instrumentalness, liveness, loudness, mechanism, tempo, organism, speechiness and valence. The users interacted with almost 4 million tracks during these sessions.

5 Problem Formulation

Since the aim of this project is session-based sequential skip prediction, we formulate the task as a supervised classification problem for sequential data. For each user, we are given a sequence of tracks and their features, and a list of next tracks that will be recommended to them. Our task is to then classify each track from the list of next tracks as 1 or 0 depending on whether the user will skip the track. Thus, for each track we need to classify, we use the acoustic

data from the tracks in the session, the acoustic features of the current track, as well as the session data as input to our model, to generate our prediction.

6 Methods

Pre-processing
In order to generate input features to pass through our models, we extracted and combined the information from both our datasets mentioned above. We divide it into 3 main types of feature vectors:

1] Track Features

In the dataset provided, we have a total of 30 features for each track. We separate the track-ids in a list and use the remaining to generate our vectors. The 29 features mentioned above include a set of 8 acoustic vectors that have been generated using deep learning techniques from the actual track wave-forms. We use them as it is in the final vector. For the remaining 22 features, we plotted their correlation plot as a heat-map to get a better understanding of redundant and correlated variables. We have plot the correlation map below.

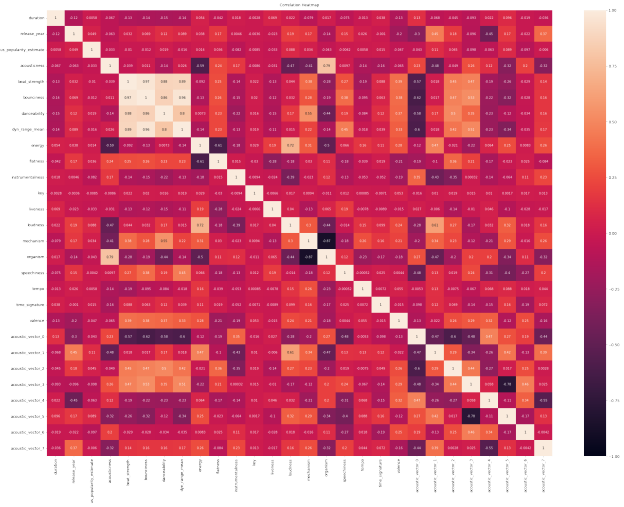

Figure 1: Heat map of feature correlation

We then ran PCA on same 22 features and plot the graph of the variance explained vs number of principle components to select an ideal number of components to reduce the dimensionality of the data. We saw that selecting 6 out of the 22 features gave a reconstruction accuracy of 99.9%, as seen in the graph below, and hence chose the number of principle components to be 6. Thus we generated a track features vector of size 13.

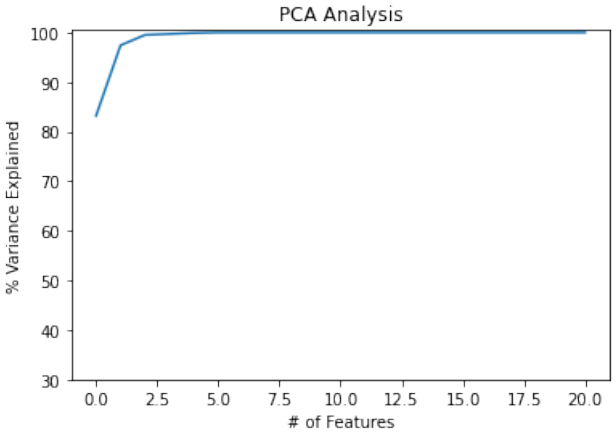

Figure 2: PCA on Track Features

We also applied PCA to the 8 acoustic features, whose graph we have plotted below. We chose to retain all of the features because they each were calculated to capture a specific aspect of the track waveform using deep learning techniques.

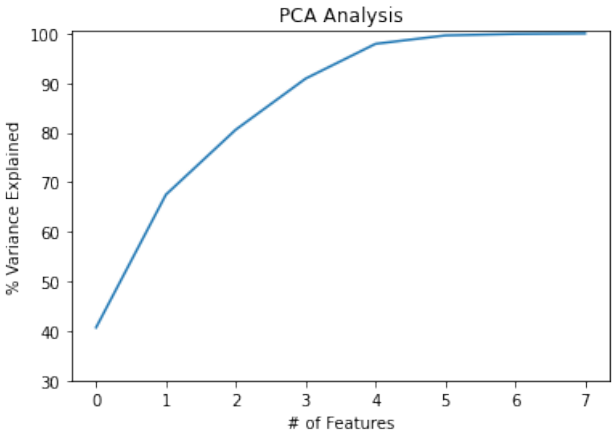

Figure 3: PCA on Acoustic Vectors

2] Session Features

For the session features, we were given 10 features per song in the first half of the session and none for the tracks to be predicted. So in order for our model to be consistent, we chose only those features that would stay constant throughout a session and those whose values we know for the

second half track. These included the session length, track position, whether the user is a premium user or not, and hour of the day. We converted the premium feature into a numerical value of 0 or 1, and thus generated a session features vector of length 4. We could not use many seemingly important session features as part of our input since they are not given for the tracks to be predicted.

3] Track-Session features
In order to incorporate the track features of tracks within a particular session, we generate a third type of feature set called track-session features. For the first half of the session, we divide the tracks into 2 categories depending upon whether they were skipped or not. For each category we calculate the average vector of all the track feature vectors in it. Thus we generate 2 track session feature vectors each of size 13, 1 for each category.

Baselines
Our simple baseline is to label all tracks as "not skipped" to get an initial idea of how well the recommendation system works. A slightly more complex baseline would entail initially creating embeddings of the track features. For each session, the average embedding of skipped and not skipped tracks are calculated. For each new track, we compare it's embedding with the average embeddings of skipped and not skipped tracks for the same session by using the cosine similarity metric which would result in the prediction.

We plan to focus on two main approaches where the first approach does not assume that the order of the track matters, and the second focuses on the importance of order of the track.

Feature-based classification using boosting trees
In this method, the 3 types of features mentioned above were calculated. Using these features,we build and train a classifier model (xgboost) to predict whether each of the new tracks is skipped by the user. This is the single classifier approach.

We see that the single classifier model too does not consider the sequence of the tracks to be predicted in the decision making process. In order to incorporate the positional and sequential data, we trained 10 positional classifiers, following [3]. The data was divided on the basis of the position of the song to be predicted in the session, and multiple position-dependent models that predict a skip at a particular position in the playlist were trained. We split the training examples into 10 subsets according by their position index in the skipping behaviour set (from 1 to 10). In total 10 models were trained. We choose all the models to have the same optimized hyper-parameters and the same classifier (XGBoost), selected after extensive cross validation.

Even though this considered the positional information, previous predictions were not influencing the current prediction yet. So we used the 10 trained models, and their outputs to generate the predictions based on different combinations and methods. For the first method, we took a weighted sum of the previous track prediction and the current model output and experimented with different weights that add to 1.

$$S(t_i) = 0.5 * S(t_{i-1}) + 0.5 * M(t_i)$$

Where,
$S(t_i)$ = Prediction for track t_i (0,1)
$M(t_i)$ = Model output for track t_i (0,1)
Since, the previous models prediction is not a 100% accurate, the error in the prediction propagates forward, and each consecutive predictions get more erroneous. To avoid this, we consider the prediction of the last training sample from each session as the previous track prediction, since it is the last stable prediction. Thus we get our current prediction as:

$$S(t_i) = 0.5 * S(t_0) + 0.5 * M(t_i)$$

Where,
$S(t_i)$ = Prediction for track t_i (0,1)
$M(t_i)$ = Model output for track t_i (0,1)
$S(t_0)$ = Prediction for the last training track for the current session (0,1)

Sequential modelling using RNNs
After experimenting with the extensions of the 10 classifiers method, even though we incorporated the sequence and position of the tracks to be predicted, the sequence of the tracks were still not taken into consideration as we took averages of the skipped and not skipped tracks. We still were treating the input tracks as a bag of tracks model to extract features. Since sequence plays an important role in the prediction, we tried a Seq2Seq model based on Causal Convolution layers[2].

7 Experiments and Results

For our classification model, we tested Random Forrest Classifier, XGBoost Classifier and Logistic

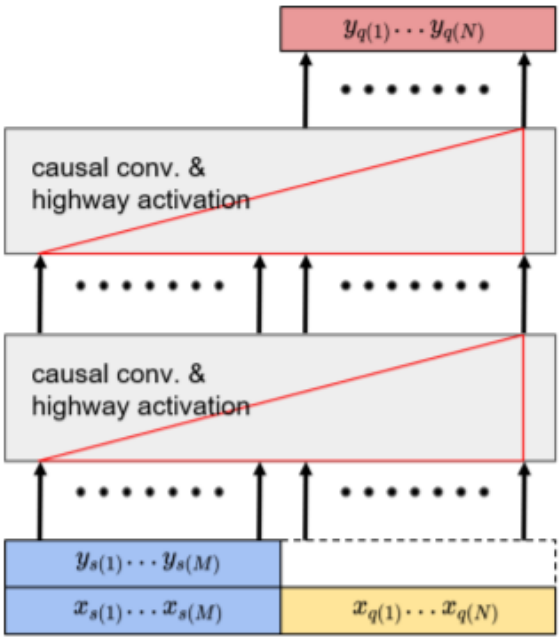

Figure 4: Block Diagram

Depth	Estimators	Train Acc	Val Acc
2	100	56.4	56.15
4	100	59.63	57.4
9	100	68.84	56.0
4	50	62.2	59.4

Table 1: Hyper Parameter Tuning for Random Forest

Classifier for all our approaches. We found out that the XGBoost classifier achieved the best validation scores across all evaluation metrics. The chosen hyper parameters were a max depth of 2 and number of estimators equal to 100. We did our hyper parameter tuning for our baseline as well as the single classifier approach. The same classifier was then used for all 10 models in the positional classifier approach.

We evaluate the effectiveness of our approaches and classifiers by using three different metrics. The

Depth	Estimators	Train Acc	Val Acc
3	100	63.39	59.75
6	100	71.96	57.4
2	100	61.3	59.80
2	50	60.60	59.4

Table 2: Hyper Parameter Tuning for XGBoost

first is the overall accuracy. This includes all of the tracks spanning all the sessions. We use the sklearn accuracy-score package for the same. This does not take into account the order of the track predictions. It also does not consider the session information such as length to evaluate the methods. For this reason, we use 2 other evaluation metrics.
1] Mean Average Accuracy (MAA) is selected as the main metric for evaluation. Since we base our approaches on the fact that the sequence of the tracks affects its predictions, we need to evaluate them based on each individual sessions too. This is because shorter sessions will have higher accuracy scores since the error in predictions will propagate. consider each session. It is defined as;

$$MAA = \sum_{i=1}^{T} \frac{A(i)L(i)}{T}$$

Where,
A(i) = Accuracy at position i of the sequence
L(i) = Boolean indicator for if the i'th prediction was correct
T = Number of tracks to be predicted for the given session

2] First Prediction Accuracy (FPA): Since we know that the error in prediction propagates due to the sequential nature of the predictions, we give more importance to the accuracy of the first prediction of each session as an evaluation metric. FPA is defined as;

$$FPA = P/N$$

Where,
P: Number of times the First Prediction was correct
N: Number of sessions

The final results(in %) across all approaches for the three evaluation metrics described above are given in the table below:

8 Conclusion and Discussion

As per our initial findings from the simple baseline (All Tracks Not-Skipped) where we labelled all tracks as not skipped, we obtained a MAA of 37% which we computed to get an overall understanding of the recommendation system. We built on this by introducing the Cosine Similarity method which instantly shows a 4% increase as introducing the average of track features of skipped and not skipped song makes it more relevant to whether a user would

	MAA	FPR	Acc
All Not-Skipped	37.00	45.98	47.95
Cosine Similarity	41.33	53.57	50.95
Single Classifier	48.11	62.50	59.80
Positional Classifiers 1	47.50	60.90	60.05
Positional Classifiers 2	52.50	71.34	62.20
Positional Classifiers 3	54.29	74.78	63.67
Seq2Seq RNN Model	63.33	82.91	78.73

Table 3: Comparison of evaluation metrics for different solution approaches

skip the new track they are presented with. But we noticed the shortcoming of this method is that when the user skips no tracks or skips all tracks the averages wouldn't hold any significance. It also doesn't take into consideration any session features.

We then tried the XGBoost Classifier (Single Classifier) which did increase the MAA, and based on [3] we extended this by building 10 classifiers that focus on the position of the track in the session and the above mentioned extensions to the same. This approach focuses only on the positions of the tracks that are new to the session and not the initial tracks which are provided as input. Hence, even though this approach takes the position into account the sequence of the tracks isn't captured. We then tried an RNN architecture (Seq2Seq RNN model) designed for music recommender systems [2] which included the sequential information of the input tracks in each session, which worked better than the other models.

For future work, our models can be improved by finding a better way to generate the track-session features, instead of a simple average for skipped and non-skipped tracks respectively. The session dataset had many more useful features that we have not used in any of approaches. This is because the feature data mentioned is only available for the tracks from the first half of each session. Other ways to utilize the session features information could possibly lead to better models.

References

[1] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. The music streaming sessions dataset. In *Proceedings of the 2019 Web Conference*. ACM, 2019.

[2] Sungkyun Chang, Seungjin Lee, and Kyogu Lee. Sequential skip prediction with few-shot in streamed music contents. *CoRR*, abs/1901.08203, 2019.

[3] Andrés Ferraro, Dmitry Bogdanov, and Xavier Serra. Skip prediction using boosting trees based on acoustic features of tracks in sessions, 2019.

[4] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2):95–116, Apr 2018.