Note: This project is live at http://recommendsongs.live
Github Repo: https://github.com/gilpasternak35/SongRecommender

Section 1: Exploratory Data Analysis

For this assignment, we utilized the Spotify Million Playlist Challenge Dataset. The Dataset contains a sample of a million playlists with corresponding metadata, which includes the playlist name, duration, and complete information about the individual tracks.

Since the entire dataset contained over 2 million unique songs (this does not include extensive repetition) we realized early on that we wouldn't need the entire dataset regardless of predictive task, and resultantly decided to speed up our feedback loop by conducting analysis on the first 48,000 playlists. While the playlist number of 48,000 was chosen somewhat arbitrarily, this included more than enough tracks (3,213,553) for us to build an adequate recommender system.

Below are some of the basic properties of our selected dataset:

1.1 Basic Properties

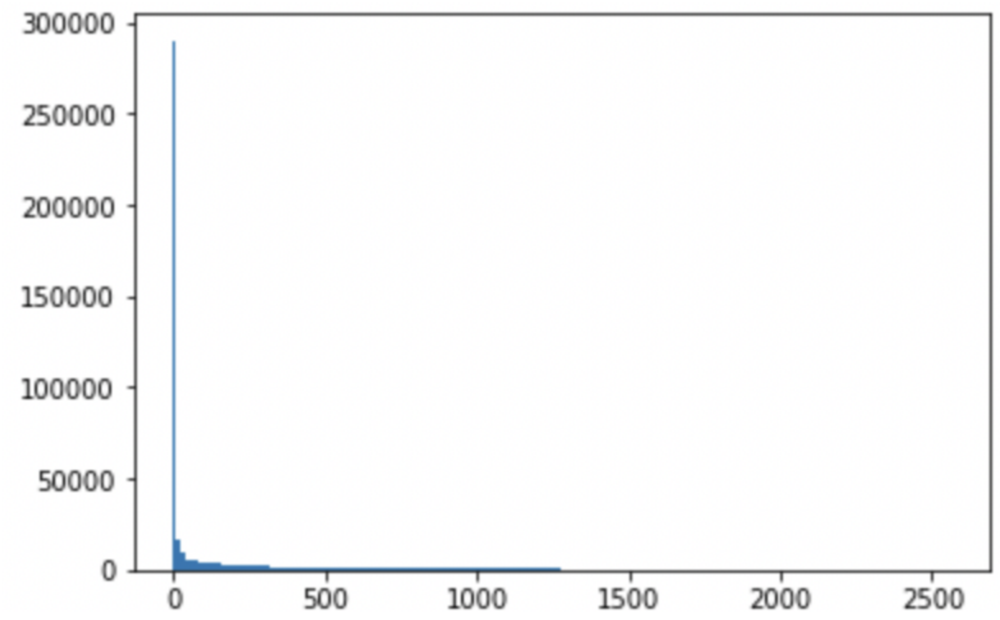
Total Users	**Total Unique Tracks**	**Longest Playlist Length**	**Average Tracks Per Playlist**	**Average Users Listening to a Track**	**Median Users Listening to a Track**	**Deviation of Users Listening to a Track**
48,000	329,741	250	66.95	9.75	1	50.99

Top 10 Songs, by number of listeners:

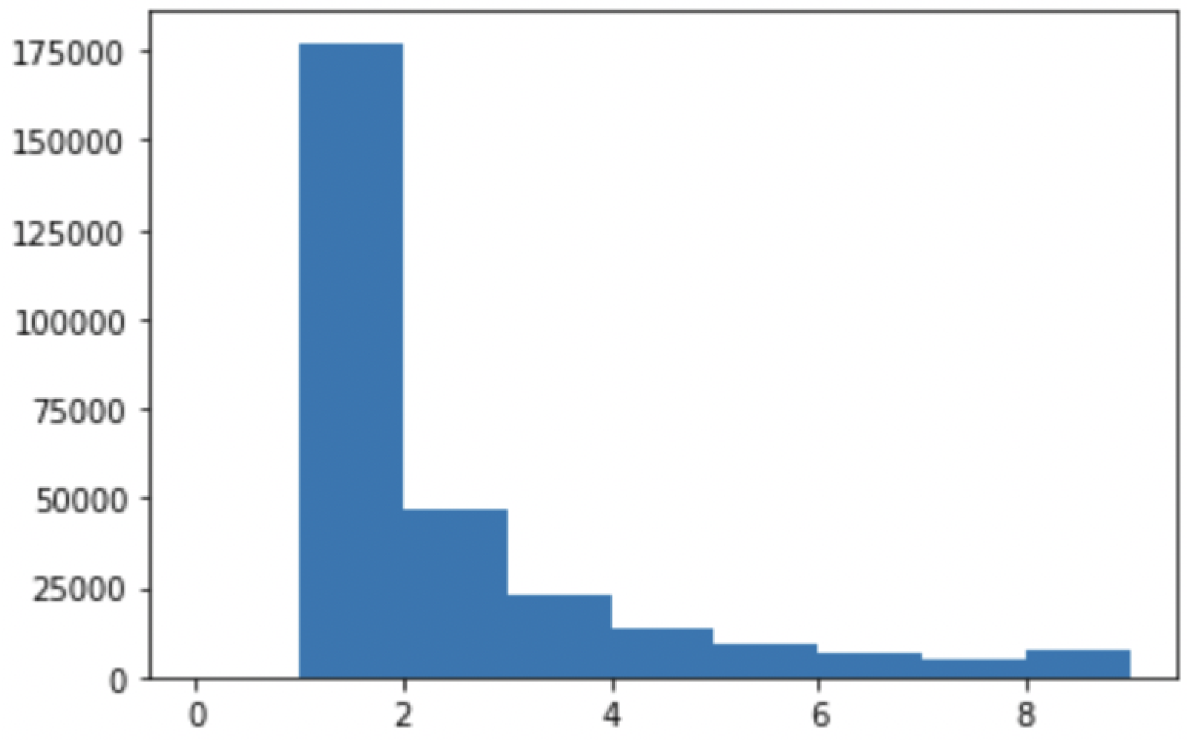
1. Closer, 3288 Playlists
2. Home, 2439 Playlists
3. Roses, 2149 Playlists
4. HUMBLE, 2139 Playlists
5. One Dance, 2049 Playlists
6. Congratulations, 1931 Playlists
7. Ride, 1906 Playlists
8. Broccoli (feat. Lil Yachty), 1887 Playlists
9. Let Me Love You, 1841 Playlists
10. Caroline, 1791 Playlists

1.2 Interesting Findings

Right-Tailed Skew
Among our first findings was that our Listeners per Song distribution was massively skewed to the right. Assuming that each playlist came from a different user (likely not too wrong of an assumption, given the playlists were randomly sampled from the over 4 billion playlists on Spotify), we noticed that the majority of unique songs (53%) had only been listened to by 1 user, whereas the 5 most popular songs had amassed over 2000 users, leading to a distribution that was heavily skewed to the right. We convey this distribution in the histogram below:



Zooming in on the songs that had been listened to by 10 people or less, we see that the skew is only slightly less pronounced:



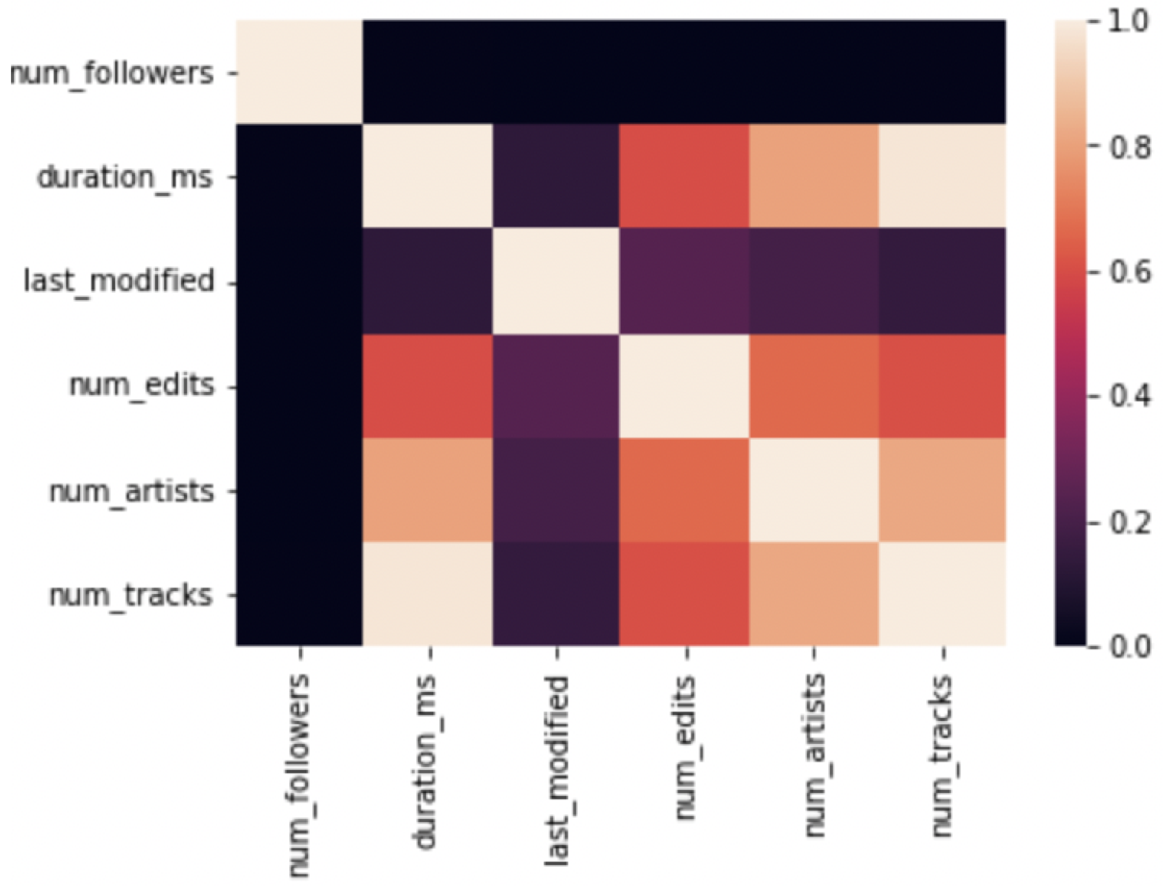
Fortunately, having so many unique songs, we aren't forced to utilize songs that had only been listened to once and could cast them out of our training set for modeling purposes. There were over 2.8 million songs we considered non-obscure (appeared in at least 5 playlists) which were more than sufficient in formally building out our Recommender System.

Data Quality

A Quick Nullity Check showed that our data contained no null values. The Histogram above also evinces that all playlists were nonempty.

Correlational Investigation of Playlist Followers as a Popularity Proxy

We shortly considered using the number of Playlist followers as a secondary proxy for the popularity of the songs in the playlist. Before this, we set out to see if any of the playlist metadata factors correlated with this secondary proxy. We plot correlation with every Playlist Metadata factor below:



This was interesting only in exclusion: neither the number of artists/songs, nor effort (num_edits) put into the playlist, correlated at all with the popularity proxy, which to some extent demotivated its use.

We then proceeded to compute the correlation of the popularity proxy with the number of popular songs (within the top 50 most popular songs) the playlist had, for which the correlation was also quite marginal (0.0005). This showed that playlist popularity was not particularly correlated with the popularity of the songs within it.

Finally, we elected to investigate the correlation between the number of popular artists and our playlist popularity metric. This was the most surprising result of all, as we found a reasonable negative correlation between playlist popularity and the number of top 50 artists on it (-0.002762). This was an interesting insight, as it seems in recommendation scenarios people tend to enjoy playlists with less cliche artists, emphasizing the impact of novelty in recommendation scenarios.

The previous 2 metrics invalidated the use of playlist followers as a proxy for song popularity; the two are, as observed, inherently different in nature. Rather, it seems to be the case that Spotify listeners have a slight preference towards novelty in their playlists, instead of the anticipated preference for popular musical hits.

Section 2: Predictive Task and Baselines

The predictive task we selected was: **given a user, could we predict whether they will have a particular song on their playlist?** We planned to evaluate this by constructing a test set from the Spotify Million Playlist Dataset, sampling half songs which had been on the user's playlist and half which had not. We wanted to see if we could accurately predict which songs had been listened to, beating several naive and some not-so-naive baselines. We elected to measure our success by the accuracy metric, defined below as follows: $\frac{\Sigma_i |y_i^* - y_i|)}{|y|}$.

where y represents the label vector and y_i^* represents our prediction for y_i. Stated simply, we measured our success by the proportion of the time we had correctly predicted whether a song had been listened to by a user.

For the purpose of this challenge we opted to implement three relevant baselines, seeking to use them as a comparison point and to justify our choice of model, the justifications and results from these baselines are listed below:

Baseline #1: Unpersonalized Popularity Prediction - This method was used to show that a primitive model without personalization could not learn a meaningful pattern in our dataset, where negative examples were created randomly and not systematically. All meaningful models would have to proceed by identifying meaningful personalized patterns. Resultantly, this did not outperform random chance.

Baseline #2: Personalized Collaborative Filtering - This method performed quite well, utilizing Jaccard similarity with the other tracks the user had listened to in order to make sense of whether the song was similar to those songs in some sense (in terms of interactions). This learned a meaningful pattern, significantly outperforming random chance and building a reasonable predictive algorithm of which songs belonged with which users.

Baseline #3: Personalized Logistic Regression- This method leveraged Jaccard similarity in order to predict interaction in a linear fashion. Unfortunately, the sheer scale of the training data made the local construction of the feature vector a near impossibility, so we had to feed the model a very small amount of data. Resultantly, it barely managed to learn a pattern, failing to outperform random chance on the test set.

Model	Baseline #1: Unpersonalized Popularity Prediction	Baseline #2: Personalized Collaborative Filtering	Baseline #3: Personalized Logistic Regression	Final Model
Train Accuracy	N/A	N/A	0.50202	N/A
Val Accuracy	0.49975	0.663	0.5	0.7629075
Test Accuracy	0.50026	0.698	0.498675	0.7641325

Since our Decision Boundary is entirely dependent on whether or not the user interacted with the item (non interacted with items are sampled from randomly), we utilize mostly personalized features that relate users and items when constructing our model. Among these are the user's other tracks, the track's other users, and the user's other favored artists . In the case that the user had never been seen before and no personalized information was available, we decided to rely on content-based features: among these the playlist name, song name, and artist name. These were used to match unseen users with items based on contextual items from the user profile.

The personalized information can be obtained from our training data by building dictionaries accounting for these feature interactions as we read through the data. We utilize default dictionaries, maintaining lists of the other attributes that have interacted with our desired entity (user or item) in order to get a sense of that entity's relative properties.

Section 3: The Model

Our Model: (CF+CBM)++

We propose a model we call (CF+CBM)++, which utilizes an ensemble of collaborative filters weighted by their validation accuracies for users who have some interaction data already available, and a content based recommender (based on the name of the user's playlist) to personalize recommendations for users unseen in the training set. In ideation, we were inspired by the AdaBoost model and the idea to build a "content-based recommender for unseen users/items" (Schedl at al).

We justify our model as follows: several review papers covering the Spotify Million Dataset Challenges spoke to the success of interaction models, among these Collaborative Filtering. Lacking the computational resources to train the large Latent Factor Models that many GPU-equipped challenge participants had trained, we opted to see if we could find an efficient way to significantly improve upon the Collaborative Filtering baseline. (CF+CBM)++ does exactly this, as it takes 10 minutes to train and predict in a standard Google Colab environment.

(CF+CBM)++ has two separate components: a collaborative recommender and a content-based recommender. The collaborative component was an ensemble algorithm: we used two separate collaborative filtering algorithms, each weighted by their validation accuracy, to make predictions on the test set. One was the standard collaborative filtering algorithm (using Jaccard similarity with the user's other tracks in order to get a sense of whether the user would normally listen to a song like this track). The other instead utilized interaction of the user with a specific song feature; the artist. The interaction between users and artists was a significantly denser problem-space as more users had artists in common than songs, and resultantly it was much easier to represent similarity between various artists (in terms of users that had listened to them). The joint coverage of the two models allowed (CF+CBM)++ to gain an understanding of whether the song and artist made sense in the context of the user's current history. The ensembling of these two CFs , weighting each by its performance, significantly outperformed either of the CFs individually and alone resulted in over a 6% increase in validation and test accuracy.

We additionally observed that while the collaborative filtering baseline was relatively strong (~0.7 accuracy) on seen users, it was much weaker on unseen users because it essentially ended up guessing that they hadn't listened to the song every time. This was not a particularly educated guess. We decided to improve this by adding a content-based recommender, utilizing a similarity score between track / album name and the unseen user's playlist title to see if the two matched. These similarity scores improved our accuracy on previously unseen examples and resultantly slightly increased our overall accuracy. Our final model significantly outperformed all baselines, and generated reasonable recommendations. All hyperparameters, including model weighting and accepted similarity threshold, were searched for and selected from a range of values by a validation pipeline.

Issues Along the Way

We ran into a multitude of issues in the creation of this model.

For one, we initially hoped to add a latent factor component to *(CF+CBM)++*. Unfortunately, the sheer scale in the number of available songs and users made this impossible even when downsampling our data, resulting in constant compute crashes within library implementations. We also attempted a from-scratch implementation, for which the algorithm was simply too slow to converge in reasonable time. Finally, we attempted to construct our latent-factor model with stochastic gradient descent, which would find a way to diverge midway through our training process regardless of our learning rate initialization because it would overshoot minima once it had gotten close enough. A potential solution for this could have been the adoption of an adaptive learning rate, but after numerous attempts time did not permit this.

Scalability also presented issues with our logistic regression algorithm, as our inability to properly utilize all the data (the model would take far too long to train if we did) resulted in drastic underfitting.

Ultimately, after implementing baselines and seeing that only collaborative filtering achieved a meaningful personalized pattern which was interpretable, we attempted to construct a model which significantly improved among CF's accuracy while doing better on unseen users and maintaining interpretability. Our model ended up being interpretable, as we used its similarity scores among tracks by similar artists to generate "reasonable" playlist predictions. We hosted these predictions live as a Flask Application.

Section 4: Relevant Literature

The dataset we are using is a subset of the Spotify Million Playlist Challenge (MPC) Dataset, which is a dataset consisting of a Million Playlists with titles and relevant metadata. This dataset was part of a challenge (extension of 2018 RecSys challenge) that measured automatic playlist continuation, measuring the quality of 500 recommended tracks as a continuation of the playlist. Instead of trying to predict whether a track will be in the playlist, however, we have taken on the task of predicting whether a user (as defined by their playlist) will listen to a given track, so our use case is slightly different. However, many techniques that emanated from the Spotify Million Playlist Challenge are still quite relevant to our prediction task.

In particular, a review paper written by the challenge organizers extensively covered some of the primary challenges of the dataset and the prediction challenge itself. Cardinal among such challenges were cold-start recommendation (recommending previously unseen items), scalability (there were millions of songs from which one could recommend), and difficulty in capturing emotional and sociological song context (Schedl et al).

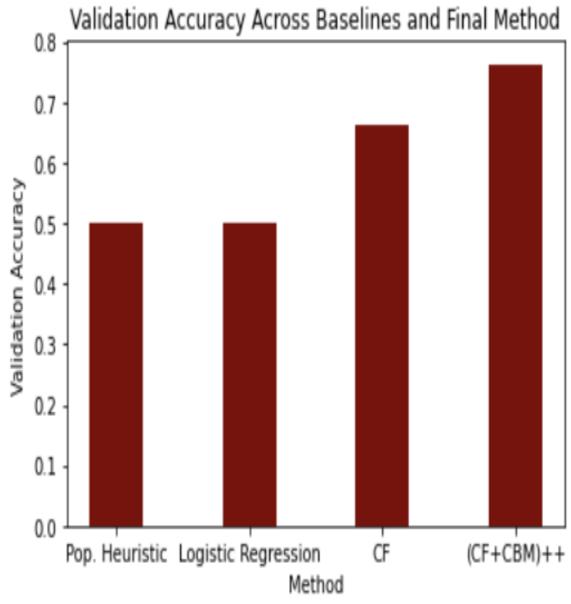
For the first of these challenges, the review paper spoke of content based recommendation (using user and item features independently when no other information was available). This could then be ensembled with a Collaborative Filtering mechanism to make history aware recommendations when history was available (generally better , per van den Oord et. al) and item-feature based recommendations when it was not. Other papers focused on the MPC dataset and the largely similar Yahoo! dataset proposed other approaches to combat this phenomena. One such approach utilized bias modeling for user, song, and genre to account for most of the variance of a personalized model when there was no available data to personalize (Dror et al). Another proposed learning a deep convolutional neural network to learn a latent policy for such scenarios (van den Oord et. al). These are both state-of-the-art methodologies that require a significant number of computational resources we didn't have.

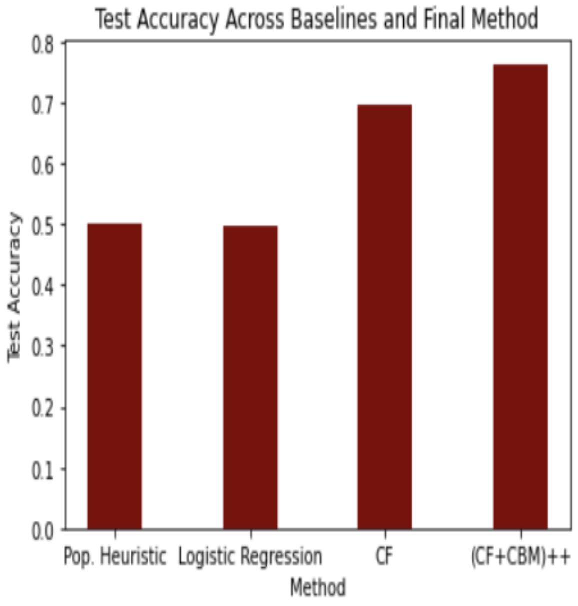
The scalability issue that frequently appeared in the literature was often simply solved by parallelization (van den Oord et. al). Since we did not have access to parallel compute, we elected to use a smaller subset of the data, as it was more than sufficient for our predictive task. Despite lacking such resources, we managed to significantly beat a relatively strong Collaborative Filtering baseline.

The challenge's organizers hoped that using song titles, which were provided as a portion of the metadata, could ease the 3rd issue. This was a complex linguistic task, however, and the several works of literature we read that focused on the MPC dataset failed to use the titles altogether. One of these papers mentioned that user personality alignment may play a big role in taste similarity (Song et al). Given this assumption, collaborative filtering tended to be the obvious approach as to the method required to recommend songs, as this filter places more emphasis on user similarity. However, collaborative filtering is dependent on a massive amount of interaction data, and in practice did not do such a good job of capturing such psychological context. There were also several models attempting to utilize external data utilizing emotional context, yet such models are at a nascent stage and proved generally unreliable (Song et al). We did manage to significantly exceed Collaborative Filtering, capturing additional context through song artists, ensembling, and playlist titles.

Section 5: Results and Conclusions

As seen in the figures below, our model significantly outperforms relevant baselines both on the Validation and Test Set.





Critically, it is visible that ensembling collaborating filtering methods (weighted by accuracy), leveraging denser artists information, using 2 models to represent mutual information, and using a content based model when no collaborative data is available yield a model which more accurately predicts whether a given user has listened to a song.

Our learned model parameters are similarities, and represent:

1) The relative closeness of a user attribute (the artists they listen to, the song they listen to) to the current track for collaborative models

Or

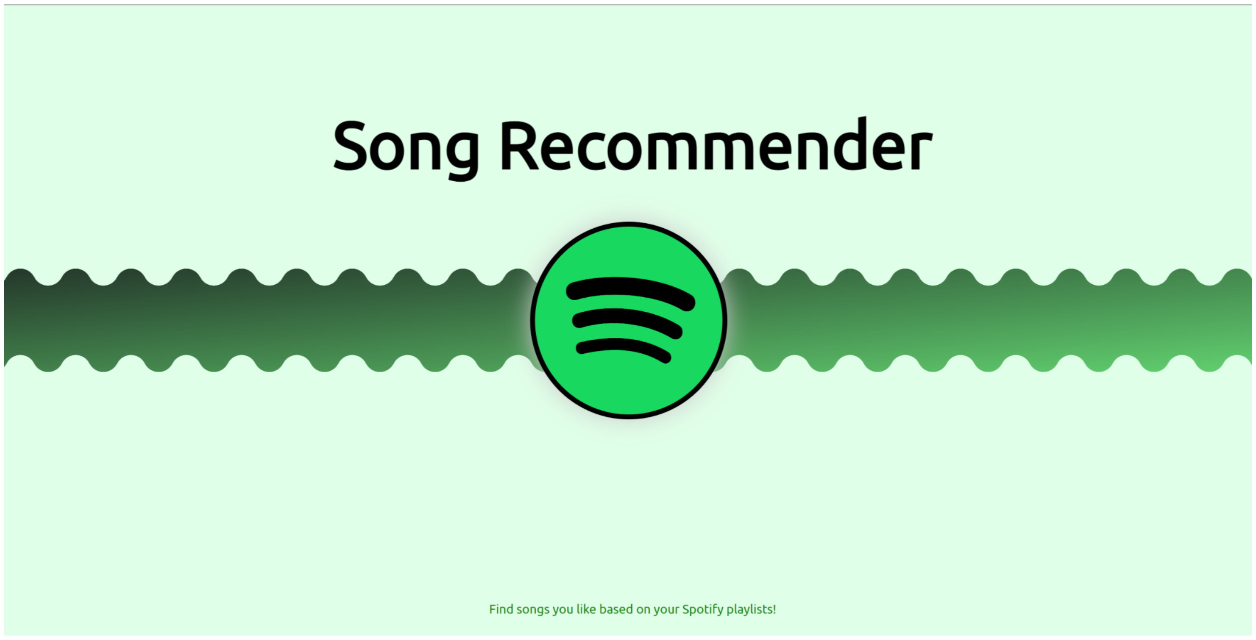
2) The closeness of the user playlist name to the track and album name (the likelihood that a playlist would contain a certain track from a certain album, based only on naming conventions.

Our positive example decision threshold, computed by a search method, was 0.015. This can be interpreted as: *if a track by a certain artist from a specifically named album has at least 1.5% overlap with the listening habits and playlist nomenclature of a specific user, then it is more likely than not that the user will listen to it.*

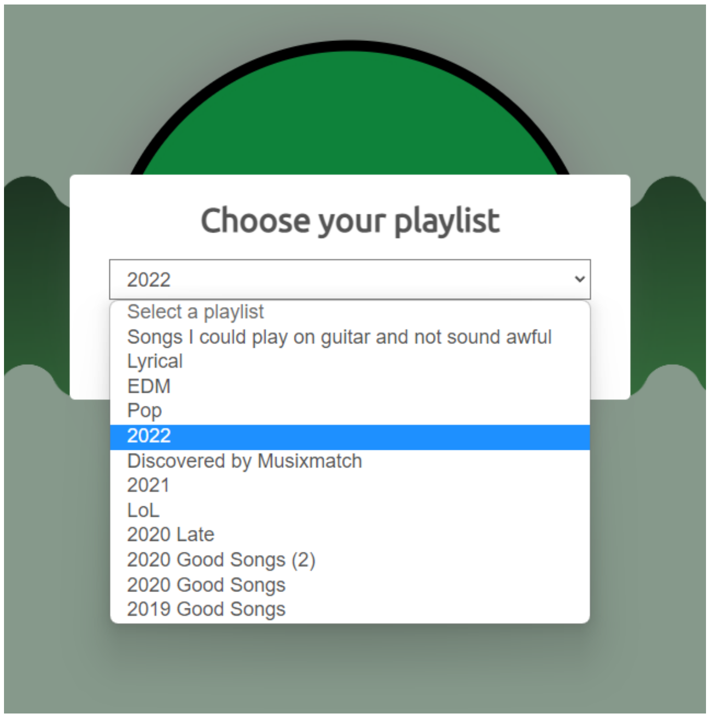
Ultimately we believe our model succeeded due to the known empirical success of boosting algorithms (we picked models that learned different features, then combined their information) and our ability to make better predictions on previously unseen users than complete guesswork. We believe that with more computational capacity, we could have extended our number of models in the ensemble (potentially also including a latent factor model), improving our accuracy further. We call for the continuation of our efforts in future Million Playlist Challenge competitions, granting more compute is available.

Section 6 (Extra): Some Generative Results and an API

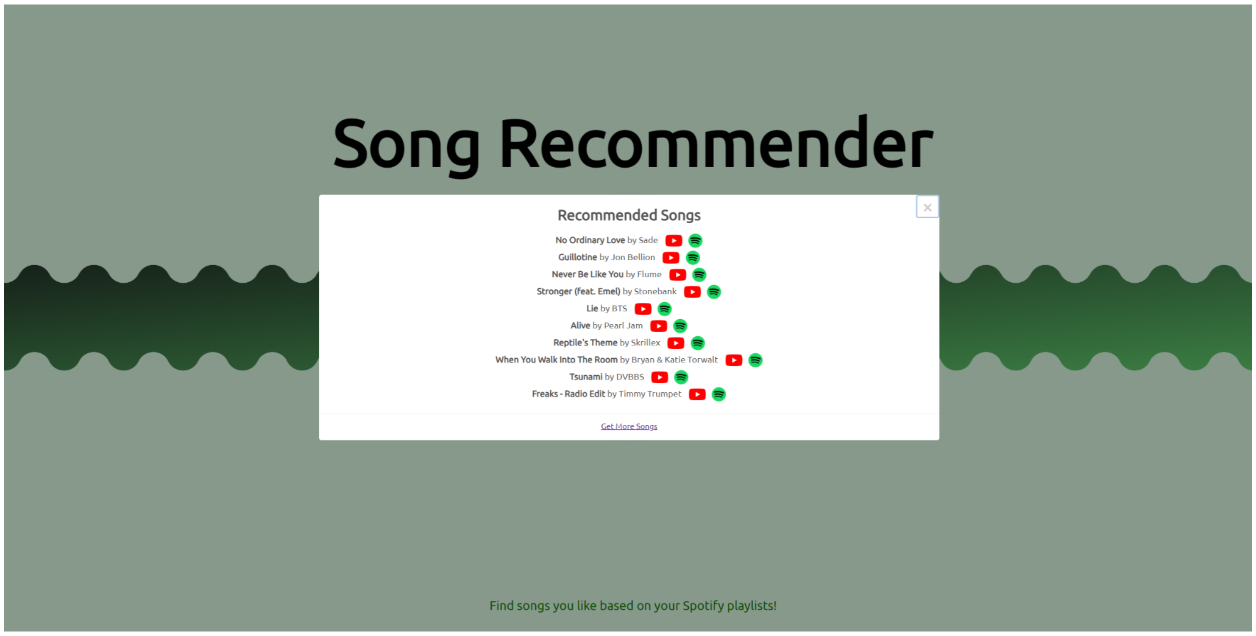
Here are some screenshots of our project in action, you can now go to http://recommendsongs.live and input your own playlists to get personal recommendations.



Screenshot of dropdown of playlists pulled using Spotify API



Example results, with YouTube and Spotify links going straight to previews.



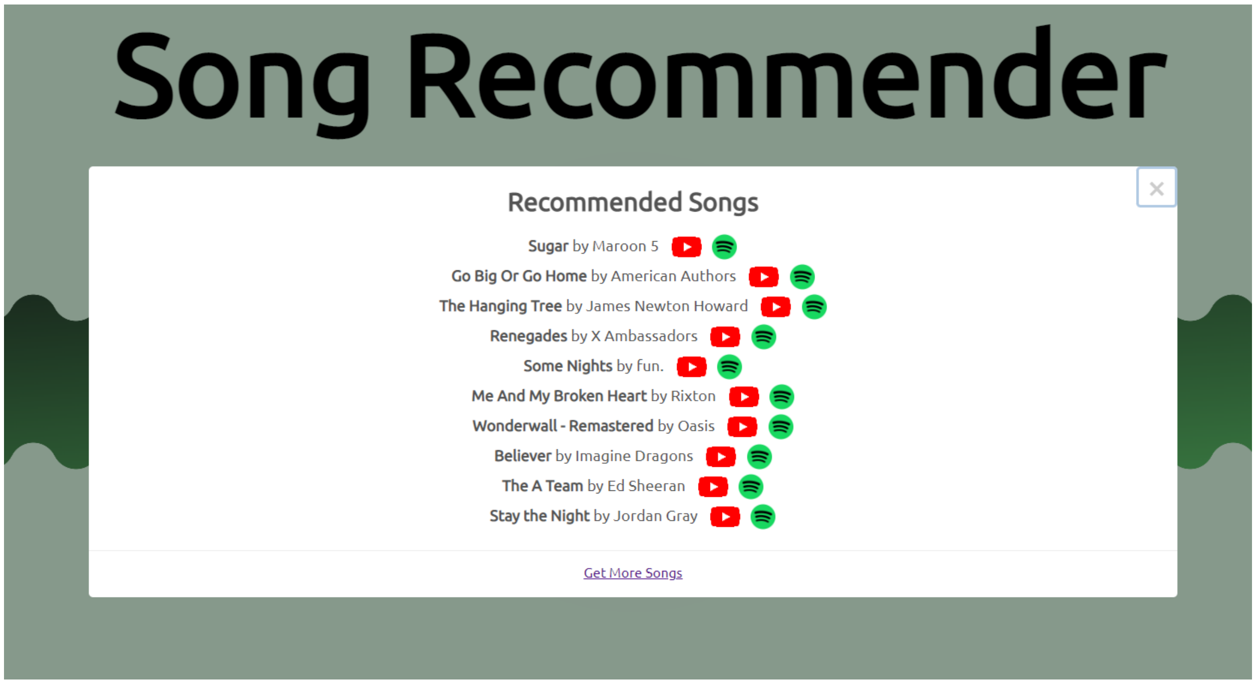
Song Recommender

Recommended Songs

No Ordinary Love by Sade
Guillotine by Jon Bellion
Never Be Like You by Flume
Stronger (feat. Emel) by Stonebank
Lie by BTS
Alive by Pearl Jam
Reptile's Theme by Skrillex
When You Walk Into The Room by Bryan & Katie Torwalt
Tsunami by DVBBS
Freaks - Radio Edit by Timmy Trumpet

Get More Songs

Find songs you like based on your Spotify playlists!



Song Recommender

Recommended Songs

Sugar by Maroon 5
Go Big Or Go Home by American Authors
The Hanging Tree by James Newton Howard
Renegades by X Ambassadors
Some Nights by fun.
Me And My Broken Heart by Rixton
Wonderwall - Remastered by Oasis
Believer by Imagine Dragons
The A Team by Ed Sheeran
Stay the Night by Jordan Gray

Get More Songs

References Cited

Schedl, M., Zamani, H., Chen, CW. *et al.* Current challenges and visions in music recommender systems research. *Int J Multimed Info Retr* 7, 95–116 (2018). https://doi.org/10.1007/s13735-018-0154-2

Song, Yading, Simon Dixon, and Marcus Pearce. "A survey of music recommendation systems and future perspectives." *9th international symposium on computer music modeling and retrieval*. Vol. 4. 2012**.**

Sarwar, Badrul & Karypis, George & Konstan, Joseph & Riedl, John. (2001). Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of ACM World Wide Web Conference. 1. 10.1145/371920.372071.

Koenigstein, Noam & Dror, Gideon & Koren, Yehuda. (2011). Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. RecSys. 165-172. 10.1145/2043932.2043964.

McAuley, Julian. *Personalized Machine Learning*. Cambridge University Press, 2022.

van den Oord, Aaron, Sander Dieleman, και Benjamin Schrauwen. 'Deep content-based music recommendation'. *Advances in Neural Information Processing Systems*. Επιμέλ. C. J. Burges κ.ά. τ. 26. Curran Associates, Inc., 2013. Web.

"Spotify Million Playlist Dataset Challenge: Challenges." *AIcrowd*, https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge.