Spotify Playlist Recommendation Model

(Logan DeSmet – *Team Submission: Erin Anderson*)

I: Business Problem

We are working at Spotify in their data analyst team and have been tasked with creating a playlist recommendation system for users based off the company's internal 1 million playlist dataset. Currently users are frustrated with finding new music based off what they are actively listening to. Everyone's taste in music goes through phases and periodic shifts – which is not considered with our current user experience. Whether that be one week listening exclusively to 80s throwbacks and the next shifting to modern day pop hits, the user would like to find music following their current taste. The root of the problem is that user created playlists require constant updating and deleting inorder to fit their current music requirements. Additionally, the inhouse Spotify '*For* You' playlist recommendation takes a holistic approach to recommending songs to the user – and does not consider recent trends. This has become exhausting for the user and is leading some to seek other forms of entertainment – not on *Spotify*. So, in an effort to improve user time spent on the platform, we are building a playlist recommendation system to solve this problem.

The key to the system is that rather than analyzing the user's personal history, which the '*For You'* playlist does, we instead will be autocompleting playlists for users - based off as little as one user inputted song. Simply put, we will be autogenerating song recommendations based off the current songs in a playlist – allowing the user to select up to 500 recommendations. When creating a new playlist, the user will input the title and atleast one song into the playlist. Next the user will select how many songs it would like the desired playlist to be – up to 500 recommendations in total – and the completed playlist will be autogenerated. Our model will be the engine behind the scenes, generating the song recommendations. The goal being that the recommended songs will lead to less user frustration and an overall increase in average user time spent on the platform – meaning more money from ads. Additionally, this could be made into a subscription only service, which in that case would increase subscriptions (likely decreasing from another streaming service – our competitors) and in turn creates growth in our revenue and market cap. Either way, this playlist recommendation model will bring immense value to our bottom line at *Spotify.*

II: Item-to-Item Collaborative Filtering System

The dataset being used to train the model is *Spotify's* 1-million playlist dataset – containing data from 1 million playlists. This data was loaded onto 10 json files, each containing 100,000 playlists.

Each row corresponds to a playlist, containing data on the playlist name, whether it was collaborative (created with another user), the playlist id, the number of tracks, the number of albums, the number of followers, the tracklist, the number of edits, the duration in milliseconds, the number of artists, and the description. The tracklist was a list of dictionaries with a dictionary for each track, including information on the track's position in the playlist, the artist, the track uri (connects to Spotify API), the artist uri, the track name, the album uri, the duration in milliseconds, and the album name. Using the json normalize pandas' function, here is one tracklist of a playlist containing 212 songs:

	pos	artist_name	track_uri	artist_uri	track_name	album_uri	duration_ms	album_name
0	0	Travis Scott	spotify:track:0ESJlaM8CE1jRWaNtwSNj8	spotify:artist:0Y5UX1MQlPlqiwlOH1tJY	beibs in the trap	spotify:album:42WVQWuf1teDysXiOuplZt	213863	Birds In The Trap Sing McKnight
1	1	Travis Scott	spotify:track:6gBFPUFcJLzWGx4lenP6h2	spotify:artist:0Y5UX1MQlPlqiwlOH1tJY	goosebumps	spotify:album:42WVQWuf1teDysXiOuplZt	243836	Birds In The Trap Sing McKnight
2	2	Travis Scott	spotify:track:2c2csx4OTYlbkzvbSTXlGY	spotify:artist:0Y5UX1MQlPlqiwlOH1tJY	guidance	spotify:album:42WVQWuf1teDysXiOuplZt	207107	Birds In The Trap Sing McKnight
3	3	Travis Scott	spotify:track:1yxgsra98r3qAbxqiGZPIX	spotify:artist:0Y5UX1MQlPlqiwlOH1tJY	Butterfly Effect	spotify:album:4fOw7xSDwqb58Z2Qia5j81	190677	Butterfly Effect
4	4	Travis Scott	spotify:track:1SGt65i9AnXYdDQt1AtDRH	spotify:artist:0Y5UX1MQlPlqiwlOH1tJY	3500	spotify:album:4PWBTB6NYSKQwfo79l3prg	461840	Rodeo
...	...	...	...	...	...	...	...	...
207	207	Russ	spotify:track:3pndPhlQWjuSoXhcIIdBjv	spotify:artist:1z7b1Pr1rSlvWRzsW3HOrS	What They Want	spotify:album:0lUL92det7mZ4DaHYmiUEC	165853	There's Really A Wolf
208	208	Jeremih	spotify:track:0PJlbOdMs3bd5AT8liULMQ	spotify:artist:3KV3p5EY4AvKxOlhGHORLg	oui	spotify:album:7DMyQuDPe8xzjC0UDSDa96	238320	Late Nights: The Album
209	209	Jeremih	spotify:track:08zJpaUQVi9FrKv2e32Bah	spotify:artist:3KV3p5EY4AvKxOlhGHORLg	Planez	spotify:album:7DMyQuDPe8xzjC0UDSDa96	240320	Late Nights: The Album
210	210	Rihanna	spotify:track:3DZQ6mzUkAdHqZWzqxBKlK	spotify:artist:5pKCCKE2ajJHZ9KAiaK11H	Loveeeeeee Song	spotify:album:4eddbruVtOqw8khwxSH6H2	256320	Unapologetic
211	211	Chris Brown	spotify:track:5gx2E5aIXMRBellPZW1up5	spotify:artist:7bXgB6jMjp9ATFy66eO08Z	Pills & Automobiles	spotify:album:3zak0kNLcOY5vFcB3Ipskp	293434	Heartbreak on a Full Moon

212 rows × 8 columns

When considering an approach to building this model the best solution seemed to be building an item-based collaborative filtering system – similar to what Netflix and other similar platforms use to make recommendations. The difference being that we aren't interested in making recommendations for a user – like what Netflix does – rather we want to make recommendations based on other tracks. Fortunately, we can use a similar approach with some key differences. Their approach at its simplest is to construct a cosine similarity matrix among users and - using an unsupervised model – would make recommendations based off similarities to other users and content the user liked in the past. Our approach was to instead create a similarity matrix of every track appearing in the whole dataset – calculating the cosine similarities between tracks (rather than users). This way the model can make recommendations based off what it receives for tracks – by optimizing around the similarity matrix.

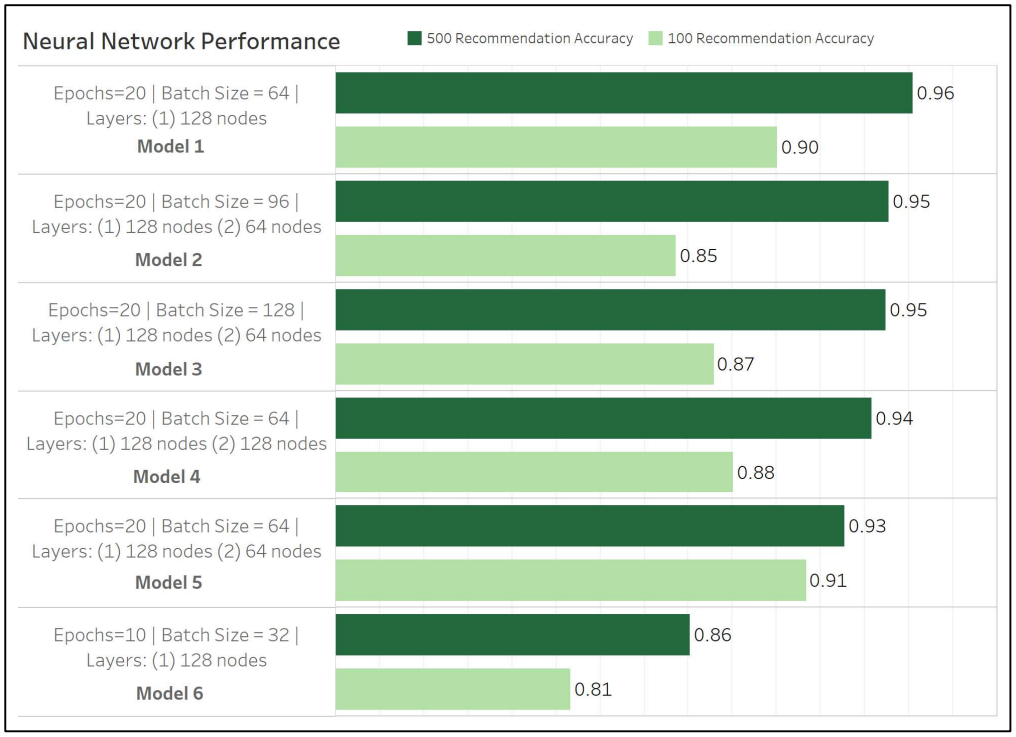
The first step was to construct a cooccurrence matrix, which documents how many times a track appeared with another track, for every song in the database. It's important to note that the colab-file used to construct this model kept crashing due to RAM limitations when creating the cooccurrence matrix (as there are hundreds of thousands of different tracks). We decided to purchase colab-pro which did allow for a much larger cooccurrence matrix, however not the full dataset. Inorder to combat this, a subset of 1000 playlists was taken at a time to construct the model – which with more computing power could easily be converted to handle the whole dataset rather

than just a subset. Once the cooccurrence matrix was constructed then it was transposed and used to construct the similarity matrix – using the cosine similarities. Note, this was the similarity metric used as it seemed the most applicable to our large number of tracks, and furthermore appeared to be the industry standard. Next, a function was constructed to intake a track name and the similarity matrix (which was 30752 x 30752), and then output the desired number of recommendations (which was specified when calling the function). This function would output the corresponding track indexes, which then using another function could be converted to the proper corresponding track names.

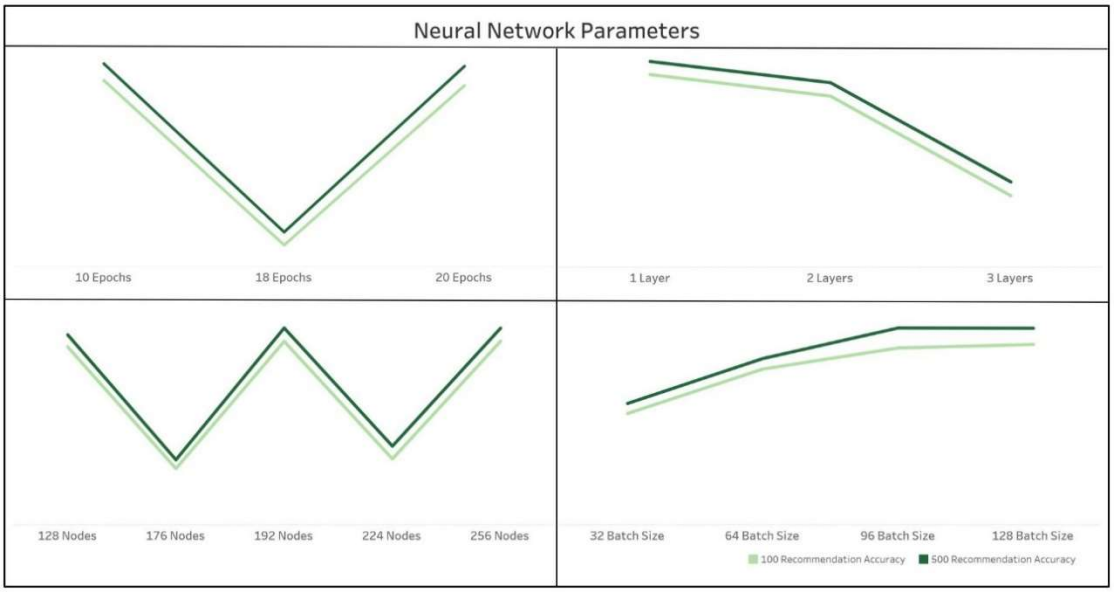
Now that the item-to-item collaborative filtering function was constructed, we constructed a multilabel classification neural network to intake a list of tracks and output a seed track. When thinking about the similarity matrix, the list of tracks would refer to the values in the columns and the seed track would be the row index (its corresponding track). The important thing to keep in mind is that the neural network model accuracy is rather *useless* for our business case. This accuracy score will be calculating the model's accuracy at predicting the seed track – the most likely out of all 30752 tracks in the similarity matrix. The model will likely never get this track right due to there being 30752 classes for the model to predict between. However, we only care if the model is predicting a track in the top 500 recommendations – which we can determine using our recommendation function (item-to-item collaborative filtering function). So instead of tuning the model's parameters based off the model accuracy – rather we will be tuning the parameters using its accuracy score at predicting within the top 500 songs and within the top 100 songs.

III: Model Performance

The similarity matrix was considered the X dataset, and the corresponding list of seed tracks was considered the Y dataset. The data was split into test and training sets using a 0.2 split (or 20% test size), with a random state of 42. Additionally, a label encoder was fit to the y datasets. A sequential Neural Network was built, using the sparse categorical crossentropy loss function. After the model was constructed the top 500 and top 100 accuracy scores were calculated, and the parameters were tuned using the results. Eight models were ran and documented - the top six performances and parameter averages for the eight models are documented as so:



Top Six Performing Models



Parameter Averages Across Eight Models

Again, we only care about the model correctly predicting the top 500 – the order is ambiguous. So in this case our models performed relatively well, however the worst performing model had a top 500 accuracy score of 31.2% and top 100 accuracy score of 27.0%. The best performing model was a single layer network with 128 nodes, trained on 20 epochs, with a batch size of 64. This model has a top 500 accuracy score of 96% and a top 100 score of 90%, which is really good considering the massive volume of tracks (or categories for the model to predict). Furthermore,

when comparing this approach to other relevant benchmark methods, this approach worked extremely well. Many typical approaches to classification, such as a random forest or logistic model, are either impossible or return extremely low accuracy scores with the dataset. Furthermore, the method of using the cosine similarities is the industry standard approach and for an evident reason, as it returned impressive performance results. Furthermore, as far as tuning the parameters in the model, we tried to make sense of the noise through the second graph. Clearly, as the batch size increased and the number of layers decreased the models performed better. Additionally, there is seemingly no relationship between the number of nodes and epochs with the accuracy of the model. Ultimately, our final single layer model has proven to be the most effective at making playlist song predictions among all relevant possibilities.

IV: Conclusions

As of now the model performs exceptionally well within the 30752 songs it was trained on. However, with access to more computing power the model can easily be converted to the full dataset and its corresponding songs. Considering the performance of the model currently, with more songs, the number of epochs should be increased as well as the batch size. Lastly, it is worth trying a multi-layered network with the full dataset – as it might perform the best with additional training time and resources (so it doesn't crash). For now, however, the model performs exceptionally well with the current dataset and will certainly bring added value to the user experience on the Spotify platform. This should perform exactly as expected, and both increase subscription counts and also further increase our bottom line and project revenue.

When considering the next steps, the first thing should be to construct the full model when granted more computing power. Additionally, considering the volume of new music that gets uploaded everyday – the model should be continuously updated using the cloud inorder to keep recommendations up to date. Another change that should be considered is how to handle circumstances where the user only inputs the playlist title but makes no track inputs. In this case, the model should be conditioned to perform sentiment analysis on the title and return track suggestions using said analysis. This would further improve the user experience, and increase the perceived value of our playlist recommendation model. Ultimately, this model already brings immense value to the platform, however with some additional computing resources and some additional time to train the model for circumstances with no tracks – the model can improve the user experience even more and in turn generate additional revenue for the platform.