Music Recommender with SVD and KNN

By Anthony Ayala, MSBA Class of 2024

Due Date: February 9th, 2024

Business Problem

Music has become more accessible and offered so much variety thanks to streaming, which has transformed the music industry. Spotify has become the world's most popular audio streaming subscription service and has continued to reign thanks to its features but more importantly their complex and intricate recommender system. The Spotify algorithm took a lot of experimenting and is still seeking improvements to this day. So, as a new analyst at Spotify, I have been challenged with an experiment to build and compare two types of recommender systems (Singular Value Decomposition and K-Nearest Neighbors) that make recommendations based on play count using a subset of the Million Song Dataset. The goal is to find which model performs better, share insights, and see whether our model of choice is valid to implement in the Spotify algorithm of the recommender system.

Methodology:

To achieve personalized recommendations, I have created a dataset of my peer and I's favorite songs from the large data set and added our own play counts to each of our favorite songs. A play count of 5 or 10 where a play count of greater than 5 for a specific song means the user liked the song, and then preprocessed the main data where we created a rating scale of play counts from 1 to 10. Then this sampled data set was joined back to the original data frame where the modeling process began with training 70% of the data and testing 30% of the data. The experiment consisted of two versions of each model, one is the baseline recommendation without tuning and the other is a tuned model. For the SVD model, I performed a SVD K-Fold to find the best parameters and then fitted the SVD model with the best parameters to make a comparison with the baseline recommendation. For the KNN model, I performed tuning with user-based and item-based parameters and compared those results to the baseline recommendation. Once all models were fitted and compared, then I had each model spit out its' top five song recommendations for a specified user. Finally, the results are addressed by the songs that a user liked a lot and see if the recommended songs are relevant in regard to sharing the same artist or genre, or just being similar enough to dictate that it was a solid match.

Recommendations:

To be quite frank, they were poor for all models and even accounting for the tuning. In my original table, I had shown a strong interest in 80s pop, 2000s R&B, and Rap, and for all the model recommendations I was lucky to be given Coldplay and Davie Bowie but even then these aren't necessarily the best recommendations as the list I had didn't show much interest in glam rock nor alternative pop rock. To be clear, the SVD model with hyper parameters did perform the best out of all

models used, hence how my best recommendation included Davie Bowie and Coldplay but the problem that lies in the popularity and frequency of these artists. Additionally, the SVD model with the best parameters metric improvements were not significant across all models. What became a growing concern throughout the experiment was the non-personalized music recommendations. For example, for users that were random and expert music reviewers there was a common trend that in the results some users shared the same song recommendation or more.

Understanding Relevant Information and Model Functionality:

Personalized recommendations are broken down into the following types: content-based, collaborative, collaborative filtering-based recommendations and hybrid which is simply a combination of different types. The scope of the experiment focused on collaborative filtering-based recommendations. Both KNN and SVD are good models for collaborative filtering, but I must highlight that the problems weren't necessarily the models but rather the data itself and facing common challenges. The data itself was quite large, yet it had too few features. Beyond that, the data itself was a solid example of data sparsity as there were over one hundred thousand unique users and songs but less than 300 unique play counts, which is a small number of interactions between item and user. Additionally, there may have been some issues with scalability and privacy. Using Apache Spark to handle large datasets, and lack of access to more of the user data and trying to find more interactions would have been very useful.

My suggestions and naïve recommendations:

With underwhelming performance from the machine learning models, and conducting this experiment to improve Spotify's recommendation engine, my approach has come to making naïve non-personal recommendations. Recommending what is popular, most played song, and an artist that pops up frequently and has a high play count is sufficient in this experiment. The thought process starts with naïve recommendations and to then see the interaction between the user and the naïve recommendation, which the interaction is either a hit or a miss. If that doesn't work well, then it comes down to adding more features based on songs and user data or adding more interactions like adding the MSBA's class top ten songs, which address the main issue of data sparsity in this dataset.