# Artist Recommender System: Personalizing Music Discovery

Facundo Herrera, Ehiane Oigiagbe, Justin Simm,
Adrian Casabal, Ethan Villalovoz, Chibuike Ugwu
CPTS 440: Artificial Intelligence
School of Electrical Engineering & Computer Science, Washington State University

# Introduction

Music streaming platforms give users access to millions of songs and artists at the push of a button. While variety is probably in the interest of the consumer, it also makes it harder to choose what to listen to next. Faced with so many choices, people often stick with what they already know instead of exploring new artists they might enjoy.

To tackle this problem, our project builds personalized artist recommendations to guide the user in the right direction. The idea is to analyze a user's listening patterns and find others with similar tastes. From there, we can suggest new artists that those similar users like—similar to receiving music recommendations from a friend. This was implemented with collaborative filtering, which means the system learns from patterns in user behavior instead of trying to analyze the music itself. Our data came from Last.fm, and it gave us what we needed to build a system that recommends artists based on shared listening habits.

# Problem Statement and Motivation

Even though it’s easier than ever to access music, it’s also easier to get overwhelmed by the options. Many existing recommendation systems tend to focus on what’s popular overall, which isn’t always helpful for someone with unique or niche music preferences. Because of this, users might not discover artists that actually match their preferences.

We determined the solution was building a smarter recommendation system—one that doesn’t just push top-chart artists but actually learns what each person likes based on who they listen to. By focusing on user interactions rather than general popularity, our system is better at finding lesser known artists that fit the user better and making suggestions that feel more personalized. Collaborative filtering makes this possible by connecting users with similar listening habits and offering suggestions based on that shared behavior.

# Dataset and Preprocessing

For this project, we used a dataset from Last.fm provided by the GroupLens research group. It includes user interactions such as the artist and song someone listened to, along with some extra details like date and time. Most of our work centered around using the artist and user info.

We started by cleaning up the dataset—removing unnecessary columns (like entry ID) and focusing on how often each user listened to each artist. From there, we grouped the data to calculate play counts per user-artist pair. To make these numbers comparable across different users, we normalized the play counts to fit  a 0 to 1 scale. This helped smooth out differences between casual and frequent listeners.

We also filtered out users who listened to fewer than five artists and artists who didn't have at least five listeners. This kept our data clean by removing cases where we wouldn't have enough info to make meaningful recommendations, removing sparsity in the resulting user-artist interaction matrix. After all that, we ended up with a clean, normalized interaction matrix that we used as the input for our recommendation model.

# Collaborative Filtering Approach

We employed a collaborative filtering approach to build a personalized artist recommendation system. Collaborative filtering is a widely used method in recommendation systems that predicts user preferences by learning from the behaviour of similar users. Instead of relying on the explicit features of the artists, collaborative filtering leverages user interaction data to find patterns that represent the relationship between users and artists.

# The prediction engine

This is at the heart of the system, which learns to recommend new artists to users based on shared listening preferences. The main idea is that if two users have historically liked or interacted with the same set of artists, there is a good chance they will also enjoy similar artists in the future. The person-to-person comparison allows the systems to infer missing user preferences and recommend artists that a given user may not have discovered yet.

# Singular Value Decomposition (SVD)

SVD is a popular matrix factorization technique that was used to implement this approach. Specifically, we use the implementation from the Surprise library, which simplifies decomposing large interaction datasets. In our case, the interaction data takes the form of a User-Artist Matrix, where rows represent users, columns represent artists, and each entry corresponds to some form of interaction metric, such as number of plays, likes, or ratings.

# Latent factors

The role of SVD is to break this high-dimensional User-Artist Matrix into lower-dimensional components, thereby capturing underlying structures in the data. This decomposition helps reveal latent factors, or hidden features, that drive user behavior and artist appeal. For example, one latent factor might correspond to a preference for upbeat music, while another might align with genre-specific interests like alternative rock or hip hop. These factors are not labeled or directly observed—they are inferred from interaction patterns across users and artists.

By mapping both users and artists into a shared latent factor space, we can estimate how well a particular artist fits a user's taste profile—even if the user has never interacted with that artist before. The model predicts missing values in the User-Artist Matrix (i.e., unknown preferences) and uses these predictions to generate a ranked list of artist recommendations tailored to each user.

Our collaborative filtering system builds on the idea of shared user behavior to offer personalized artist suggestions. The combination of interaction data, SVD-based matrix factorization, and latent factor modeling allows us to deliver intelligent recommendations that reflect users' implicit preferences, leading to a more engaging and discovery-rich experience.

# Model Training Process

We implemented a collaborative filtering approach using the Singular Value Decomposition (SVD) algorithm from the Surprise Python library to generate personalized artist recommendations. This algorithm enabled us to learn underlying patterns in user listening behavior without relying on explicit artist metadata. The modeling process involved several key stages: data preparation, parameter tuning, model training, and evaluation.

# Data Preparation

Before model training, we constructed an interaction matrix from the filtered Last.fm dataset, where each entry reflects a user's normalized play count for a specific artist. Normalization was necessary for two reasons:

It reduced the impact of outliers (users with extremely high or low play counts), and

It compared individual user preferences by scaling play counts between 0 and 1.

To evaluate the model's generalization ability, we split the dataset 80/20. The training set contained the majority of user-artist interactions, which were used to learn latent patterns, while the test set included unseen user-artist pairs for performance evaluation.

# Parameter Tuning

SVD introduces several tunable hyperparameters that significantly impact performance:

- n_factors: the number of latent dimensions used to represent users and items. A larger value can capture more complex interactions but risks overfitting if it is too high.
- lr_all: the global learning rate determining how quickly the model updates its parameters during training.
- reg_all: a regularization parameter to prevent overfitting by penalizing large weights in the latent matrices.

We conducted empirical testing to identify optimal values for these parameters. After several iterations, we settled on n_factors = 50, lr_all = 0.005, and reg_all = 0.02, which provided a good trade-off between accuracy and generalization.

# Model Training

Using these hyperparameters, we trained the SVD model on the training data. The SVD algorithm factorizes the user-artist interaction matrix into two smaller matrices: one representing users in a latent feature space and one representing artists in the same space. The dot product of these two matrices approximates the original play count. This allows the model to:

- Infer missing values (i.e., unobserved user-artist interactions), and
- Discover implicit relationships based on shared patterns (e.g., users who like similar genres).

Importantly, this approach doesn't require explicit artist features such as genre or popularity; recommendations are derived purely from collaborative signals.

# Validation and Evaluation

We implemented the recommendation generation phase using the trained SVD model to produce personalized top-5 artist recommendations for each user. This step involved extracting model predictions, presenting them in a meaningful format, and evaluating their real-world relevance using practical ranking-based metrics. The process consisted of three main stages: ranking generation, visualization, and recommendations quality assessment.

# Top-N Recommendation Generation

Once the model was trained and validated, we used the full dataset to generate predictions for each user-artist pair that had not yet been observed (i.e., the anti-testset). For each user, we sorted these predictions in descending order of estimated preference and selected the top 5 artists. These top-N results represent the model’s best guesses of artists the user is most likely to enjoy, based on listening behaviours from similar users.

This stage made use of a helper function to extract the top 5 recommendations per user from the Surprise predict() output. These recommendations formed the basis for both practical evaluations and user-facing visualizations.

# Recommendation Visualization

To improve the interpretability of our results, we designed a visualization interface that displayed each user’s top-5 recommendations as horizontal bar charts using matplotlib. In the absence of actual rating scores, we applied descending mock scores (5 to 1) to convey ranking. This simple visual hierarchy helped us communicate the relative importance of each recommendation.

# Recommendation Evaluation

To evaluate our model's effectiveness, we used Root Mean Squared Error (RMSE), a widely used metric in collaborative filtering. RMSE measures the average difference between the model's predicted scores and the actual user interactions. A lower RMSE indicates that the model is more accurately predicting user preferences.

Our model achieved a low RMSE of 0.0498, which suggests high prediction accuracy. To assess consistency, we performed 5-fold cross-validation and recorded the standard deviation across folds. This helped confirm that the model's performance was not only accurate but also stable across different user subsets.

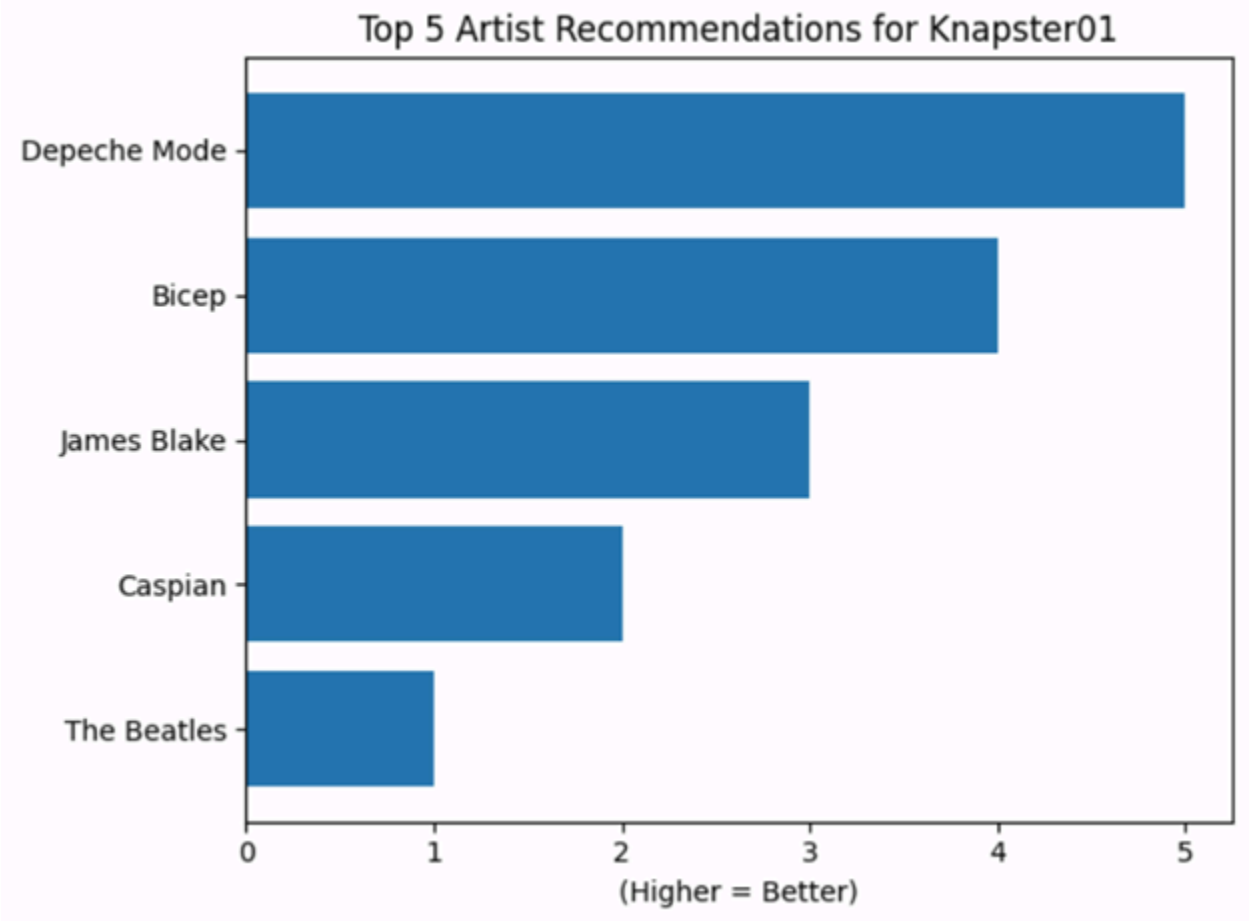
Although we received feedback to explore ranking-based evaluation methods such as Mean Reciprocal Rank (MRR), Precision@5, and NDCG@5, we focused primarily on RMSE due to time constraints. These other metrics remain promising directions for future work, as they better capture how well a model ranks relevant recommendations, something especially important in Top-N recommender systems.

# User Output

As a temporary measure, we have implemented a simple interface for users to be given their top five recommended artists. We used a pandas dataframe to organize the data using a simple 1-5 ranking associated with a given artist. We then used the matplot python library to construct a table using the data frame to display this data. Each user was then given a table display as below.

| Rank | Recommended Artist |
|---|---|
| 1 | Depeche Mode |
| 2 | Bicep |
| 3 | James Blake |
| 4 | Caspian |
| 5 | The Beatles |

As an extra measure, we implemented a bar graph as an option. This method created a bar graph for each user by implementing the ax.barh method for each users' stored top 5 artists. They were then displayed as below.




During the development of our project, we were faced with numerous essential challenges inherent in most projects using a collaborative filtering approach. Two of the most significant challenges that we faced were the cold start problem and data sparsity. These issues had direct implications on the quality and reliability of the recommendations produced by our model and influenced many of the design choices during the project.

# Cold Start & Data Sparsity Issue

The cold start problem, especially, is a significant challenge for our collaborative filtering approach. This occurs when the system attempts to make recommendations for new users with

limited or no listening history, or when new artists are introduced into the system with very few interactions by any users. Collaborative filtering depends on the availability of common behavior between artists and listeners, commonly by way of similar interactions. So then the unavailability of such information affects the accuracy of the model and severely disabled the model from making effective predictions. Without some data beforehand on a listener's taste or on an artist's fan base, the system doesn't have enough context to make relevant recommendations. This limitation is not limited only to our project; it is a well-known issue in the recommender systems domain and is still an open research question.

Aside from the cold start problem issue, we also encountered serious data sparsity issues. The Last.fm dataset utilized for this project, although full of user-artist interaction data, revealed that many users had only interacted with a small subset of artists.

# Solutions

This made the user-artist interaction matrix highly sparse, with many of the entries being empty. This sparsity puts a restriction on the level of useful information that is present for the model to detect strong patterns of similarity among users. The absence of common listening histories among users undermines the performance of collaborative filtering since collaborative filtering relies on the identification of shared preferences to make knowledgeable recommendations. To combat these issues, we employed several pragmatic approaches. First, we removed users and artists with very few interactions, thus removing data points that mostly added to the sparsity problem. This preprocessing step allowed us to focus the model on users and artists with longer listening histories, thus improving the quality of the learned patterns. Second, we employed matrix factorization through Singular Value Decomposition (SVD), a very effective technique to deal with missing data and sparse matrices. SVD enabled us to learn latent factors describing hidden user and artist attributes and hence enable the system to make precise estimates even when there is no intensive direct interaction data available. One key design decision that enhanced the user experience was our decision to show recommendations by username rather than numeric user IDs. This made the system output more readable and recognizable to users since they were able to directly observe recommendations for their own accounts.

Despite all these efforts, the cold start problem was one where further improvement is necessary. Our current approach, though helpful for users and artists with enough past data, is troubled by the lack of past data. One of the possible enhancements we have identified involves integrating Spotify's API into our recommendation engine. Spotify’s API is rich in metadata regarding artists, including genres, popularity levels, related artists, and followers.

The inclusion of this data would enable the system to provide recommendations not just based on user activity but also on artist characteristics, in this way further alleviating the cold start issue

by providing recommendations even in the absence of historical interaction data. Expanding on this concept, one of the most significant enhancements we anticipate is the creation of a hybrid recommendation system that integrates collaborative filtering with content-based filtering. Hybrid techniques borrow the strengths of each methodology and merge them for stronger recommendations while offsetting their individual weaknesses. We could improve the system's capacity for fitting new users and new artists by hybridizing collaborative scores with content-based similarity metrics, or by applying ensemble methods for merging the output of a variety of models.

# Future Work

A future direction is also aligned with enhancing accessibility and user engagement by developing an intuitive web interface. Using Streamlit, we aim to present an interactive web-based interface through which users can conveniently type in their usernames or favorite artists in order for them to obtain in real-time personalized recommendations. This interface would not only augment the ease-of-use of the system to end-users but also make qualitative evaluation of recommendation quality easier. Through it, users would be able to navigate through recommended artists, view similarities, and interact with the system in a more natural way.