A User-to-User Music Similarity System Using Machine Learning

By Nick Hellmer

1 Problem Statement

Understanding musical similarity (whether between artists or between listeners) is essential to personalization in digital music platforms like Spotify. Similarity-driven models and recommendation techniques play a central role in finding patterns in user behavior to power discovery, engagement, and overall experience.

This project explores two complementary angles: identifying artist-to-artist similarity from playlist co-occurrence (for artist recommendation), and modeling user-to-user similarity based on shared artist preferences (for matching similar listeners).

At the core of this work is the hypothesis that patterns in user-created playlists reveal latent musical connections, both between artists who frequently appear in the same user-created playlists, and between users with overlapping preferences. This work aimed to capture these patterns using large-scale Spotify playlist data, co-occurrence models, and KNN models. While the dataset is massive (33.54 GB), it was partitioned strategically to balance computational efficiency and the ability to draw confident conclusions.

2 Data Source

The "Spotify Million Playlist Dataset", available on Kaggle (`https://www.kaggle.com/datasets/himanshuwagh/spotify-million/data`), contains one million real human created playlists, with each playlist including metadata on the list of tracks. For each track, there is also artist and track-level information, such as the artist's name, which was a core part of our modeling.

To enable rapid prototyping, an initial focus was placed on a more manageable set of the data (the first 1,000 playlists). This allowed for faster iteration, model testing, and evaluation. Then, the sample size was scaled up in select areas to larger subsets (10,000 and 50,000 playlists) to validate initial discoveries on more robust data samples.

While this dataset was notably large for academic work, it is relatively small compared to the scale of real-world data used by major technology organizations such as Spotify, suggesting that the model's performance could improve further with greater scale. It is ample data for this project's purposes, with the added benefit that it is neatly structured in an easily parse-able JSON format.

3 Methodology: Modeling User Similarity

3.1 Data Pre-Processing: Synthetic User Generation

Motivation and Clustering Strategy. Since the Spotify playlist data used does not have any user tags, simulation of realistic music listeners was done by clustering playlists into groups that represent "synthetic users." This was a necessary and key step, as the existence of user profiles is prerequisite to formulation of recommendation techniques at the user-level. Each synthetic user was formed by aggregating a set of playlists that exhibit similar musical preferences, captured through artist co-occurrence patterns.

To create synthetic users through clustering, both k-means and k-medoids was considered. While k-medoids has some advantages (specifically with handling non-Euclidean distances), k-means was ultimately selected due to its simplicity and general performance in high-dimensional spaces. Since k-means usually uses Euclidean distance, normalization of all playlist vectors before calculating distances was a necessary first step. This normalization results in cosine similarity being effectively equivalent to Euclidean distance.

Determining the Number of Clusters (k). Two approaches were used to determine the optimal number of clusters (i.e., synthetic users) for the k-means algorithm. The first was the elbow method, which is a commonly used technique for k-means modeling.

In parallel, this project used a rule-based method grounded in real-world Spotify usage behavior. According to statistics from `soundplate.com`, as of 2024 there were 356 million Spotify users and 8 billion user-curated playlists, implying the average user has ~23 playlists. Using this insight, a forced number of clusters was used by dividing the total number of playlists in the input dataset by 23 (e.g., $k \approx \frac{\#\text{ playlists}}{23}$). This method uses a natural baseline as motivation to determine the best number of clusters.

These two methods were repeated for each of our three sample sizes, resulting in six total unique datasets to use as inputs.

Synthetic Dataset Composition Across Scales. *Table 1* compares the synthetic user datasets generated under both the Elbow Method and Average # of Playlists strategy, across three dataset sizes: 1,000, 10,000, and 50,000 playlists. For each configuration, the number of synthetic users as well as key summary statistics were reported.

A few key patterns emerge from these summary statistics. Both strategies result in 23 playlists per user when scaled to 10k and 50k playlists. This confirms that the Average # strategy successfully targets real-world user behavior. Additionally, the diversity of user profiles diversifies as the number of playlists used increases, as reflected in the growing max values. This hints at the potential scalability of psuedo user generation and the need for robust similarity modelings.

Metric	**1k**		**10k**		**50k**	
	Elbow	Avg.	Elbow	Avg.	Elbow	Avg.
Total synthetic users	19	43	433	434	1761	2173
Playlists per user						
Mean	52.63	23.26	23.09	23.04	28.39	23.01
Min	12	8	1	1	1	1
Max	352	198	868	859	2215	1864
Unique tracks per user						
Mean	2583.11	1240.58	1082.88	1080.34	1195.84	992.82
Min	439	335	22	22	9	9
Max	15827	8970	38100	37605	88736	75856
Unique artists per user						
Mean	1014.32	549.81	468.83	467.79	506.01	429.54
Min	194	127	11	11	7	3
Max	5727	3700	13279	13130	27874	25195

Table 1: Synthetic user comparison across three playlist volume inputs (1k, 10k, 50k) and two clustering strategies (Elbow vs. Avg. # of Playlists)

3.2 User Similarity Modeling

Once the psuedo-users were constructed by clustering playlists, the next step was to model user-to-user similarity based on their aggregated preferences. The goal was to identify which pseudo-users are musically similar, which in turn powers our downstream evaluation strategy discussed in a later section of this paper. Because user vectors can vary in both direction and magnitude, different cosine similarity measures were explored to better quantify these relationships.

Similarity Metric Variants. Three different cosine similarity formulas were used here and then evaluated to determine which performs best: Standard Cosine Similarity, Scaled Cosine Similarity, and Soft Cosine Similarity (with tunable exponent α).

$$\text{cosine}(u, v) = \frac{u \cdot v}{\|u\| \cdot \|v\|}, \quad \text{scaled\_cosine}(u, v) = \text{cosine}(u, v) \cdot \min\left(\frac{\|u\|}{\|v\|}, \frac{\|v\|}{\|u\|}\right),$$

$$\text{soft\_cosine}_\alpha(u,v) = \frac{u \cdot v}{\|u\|^\alpha \cdot \|v\|^\alpha}, \quad \alpha \in [0,1]$$

Each variation attempts to balance vector directionality with magnitude in different ways. Standard Cosine Similarity only considers the direction of user vectors, while Scaled Cosine and Soft Cosine also consider vector magnitude. The soft cosine introduces a tunable parameter α that creates a trade-off between pure dot product ($\alpha = 0$) and standard cosine similarity ($\alpha = 1$).

4 Evaluation and Final Results

4.1 Model Output: User Similarity Scores

The user similarity model aims to identify, for each synthetic user, a ranked list of other users with similar musical preferences. These user-to-user similarity scores are also essential to the evaluation of the model's performance.

Each synthetic user is represented by a normalized vector of artist counts, capturing the user's music taste through frequency of artist appearance. To compare users, a pairwise similarity matrix was computed where each entry (i, j) reflects the similarity between user i and user j. To enhance end user interpret-ability, three different scoring functions were explored to enhance user experience of these similarity values. First, the raw similarity score is recorded directly from the similarity matrix entries. Second, a scaled version of this score is derived by applying min-max normalization across all pairs, mapping the scores to a standardized range between 0 and 1. Finally, a percentile-based score is computed by ranking the raw scores across all user pairs and converting each into a percentile. This percentile allows easier interpretation of a given similarity value relative to the overall population. The latter two scores are simpler to understand for the end-user. Once the full matrix is created, the model can generate a similarity score between two users, or generate the top-K most similar users for a given user. Transforming the raw scores into our more interpretable scores produces an output such as:

62.5% match, placing you in the top 4.1% of user similarities

where the first score is the scaled score, and the second is the percentile.

4.2 Model Testing

Evaluation Strategy. The goal of the user-to-user similarity component of this project was to model musical similarity between users. Since direct evaluation of the model is difficult due to the lack of labels (unsupervised nature), a process was created to assess the accuracy of the outputted similarity relationships. Specifically, the evaluation tested whether the top-K most similar users could predict future listening behavior. This evaluation strategy is inspired by recommendation systems, but was used strictly for testing purposes. Consider the following example:

- First, split each pseudo user's playlists into training and test sets.
- Then, after constructing the similarity matrix from the training data, extract the top K most similar users to a given user A.
- If user A is musically similar to users B, C, and D, then B/C/D's artist habits should be a good indicator for A's future behavior.

The evaluation then combines the artist vectors for users B, C, and D into a single vector, which acts as a "predicted" artist profile for user A. This vector contains the prediction of what the model thinks user A will listen to in the future.

Finally, this predicted vector is compared against the actual artist vector from user A's reserved test. The test playlists reflect what user A actually listened to in the future. The key performance metrics used in this evaluation are **Recall@N**, **Precision@N**, and **F1@N**. These metrics quantify how well the aggregated predictions from a user's K nearest neighbors align with the artists in their reserved test playlists.

Recall@N is defined as the ratio of a user's true test artists that appear in the top-N predicted artists. **Precision@N** is the fraction of the top-N predicted artists that appear in the user's test data (i.e., correct predictions). **F1@N** balances the two by computing their harmonic mean.

$$\text{F1@N} = 2 \cdot \frac{\text{Precision@N} \cdot \text{Recall@N}}{\text{Precision@N} + \text{Recall@N}}, \quad \text{Recall} = \frac{\text{all hits}}{\text{all actual test artists}}, \quad \text{Precision} = \frac{\text{all hits}}{\text{N}}$$

Example: Let's say for `user_0009`:

- Actual test artists: [Taylor Swift, Drake, Beyoncé]
- Predicted top-N artists (based on K similar users): [Drake, Beyoncé, Kanye, ...]

Then:

$$\text{Matches} = 2, \quad N = 50, \quad \text{Total actual test artists} = 3$$

$$\text{Recall@50} = \frac{2}{3} = 0.667, \quad \text{Precision@50} = \frac{2}{50} = 0.04$$

$$\text{F1@50} = 2 \cdot \frac{0.667 \cdot 0.04}{0.667 + 0.04} \approx 0.074$$

These scores are calculated for all users, and then averaged to evaluate model performance.

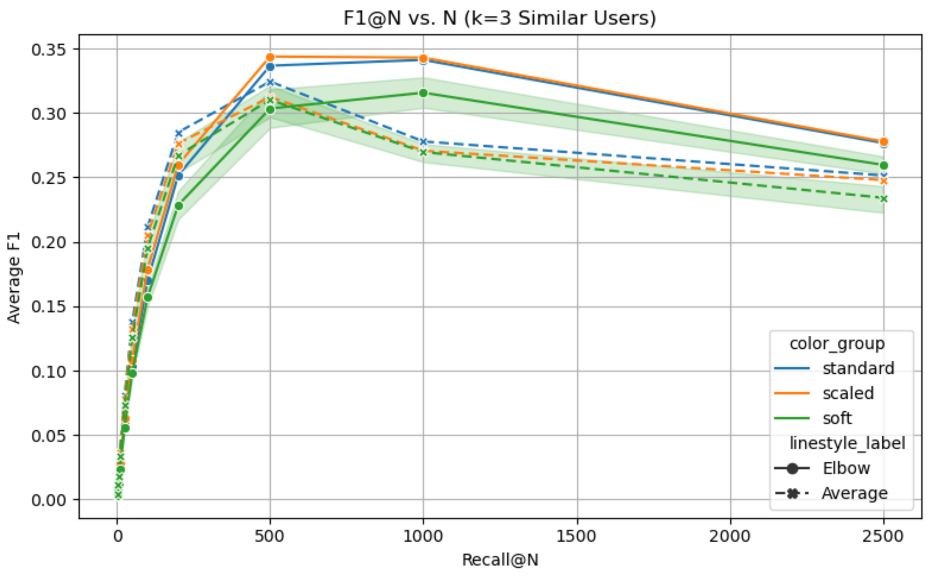
Evaluation Results. This project began by testing a wide range of model configurations on the 1k playlist dataset to evaluate the quality of the user-to-user similarity scores more efficiently. The smaller dataset allowed for fast iteration and low-level inspection of parameter combination before scaling up to the larger 10k and 50k datasets. The grid search on the 1k dataset included the three cosine similarity methods discussed prior, a set of nine tunable α values for soft cosine, eight values for the number of neighbors k (from 1 to 20), and eleven values of total recommended songs to return N (i.e., Recall@N), from very short (N=1) to very large (N=2500). In total, this resulted in over 500 combinations to test, each evaluated on precision, recall, and F1. Through this exhaustive exploration of model configurations on the 1k dataset, this strategy was able to identify the most promising parameter ranges to narrow the search space for later larger scaled evaluations.

Figure 1 contains the resulting plots from the evaluation on the 1k dataset, from which a few key insights were found regarding the optimal N. We see that the N values with the highest scores depends greatly on the associated metric score. Precision is optimized with a smaller N, but likely too small such that it is a very skewed or biased score. Recall benefits from large N and appears to continue to trend upward past $N = 2500$. This is too large of list of a user's predicted future artist preferences to be realistic. F1 finds a better balance between the two, but is still optimized at too large of an N to be realistic. As a result, a fixed $N = 50$ was chosen for all following evaluations, which balances higher scores while remaining realistic.

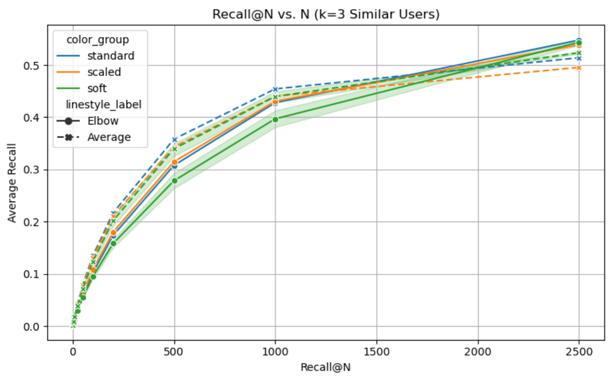
Next, we focus on the highlights from the plots in *Figure 2*. First, they help narrow the k range. It is clear that models generally have higher scores with a lower number of neighbors, particularly for $k \in [1, 3]$, but there is still strong performance through $k = 10$. Second, all three plots clearly show soft cosine lagging behind standard and scaled. Scaled potentially has a slight edge on standard, specifically at higher k values, but not enough to be truly significant. The average dataset consistently outperforms the elbow dataset for F1 and recall, but under-performs for precision. Based on these three graphs, we were able to narrow the k-range to smaller values and eliminate soft cosine as a contender.

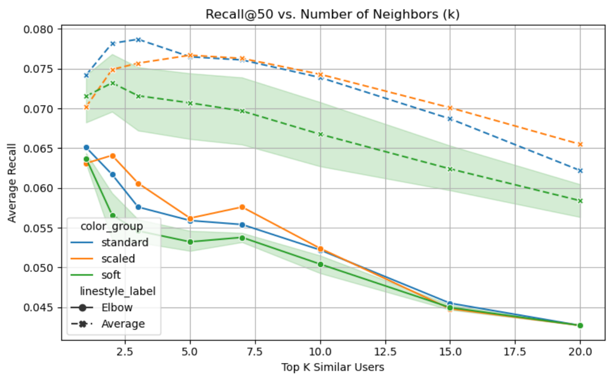
With the narrowed parameter ranges, next was to execute the same methodology on the 10k dataset (see *Figure 3* for the resulting plots). For average versus elbow datasets, average outperforms for all scores, but not by a large margin. Standard cosine consistently outperforms scaled for all three scores, and by a good margin. The k range peaks in the three to five range. This is a small increase from the 1k dataset's optimal k, but still prefers small k. We also see that all three scores across all number of neighbors generally increase from the 1k dataset to 10k, hinting that the model's power may scale with dataset size.

Finally, in *Figure 4*, application of the narrowed parameters was done on the 50k dataset. We see similar patterns to the 10k results (the graphs generally having the similar peaks and valleys locations) the key



(a) 1,000 playlists: F1@N vs N

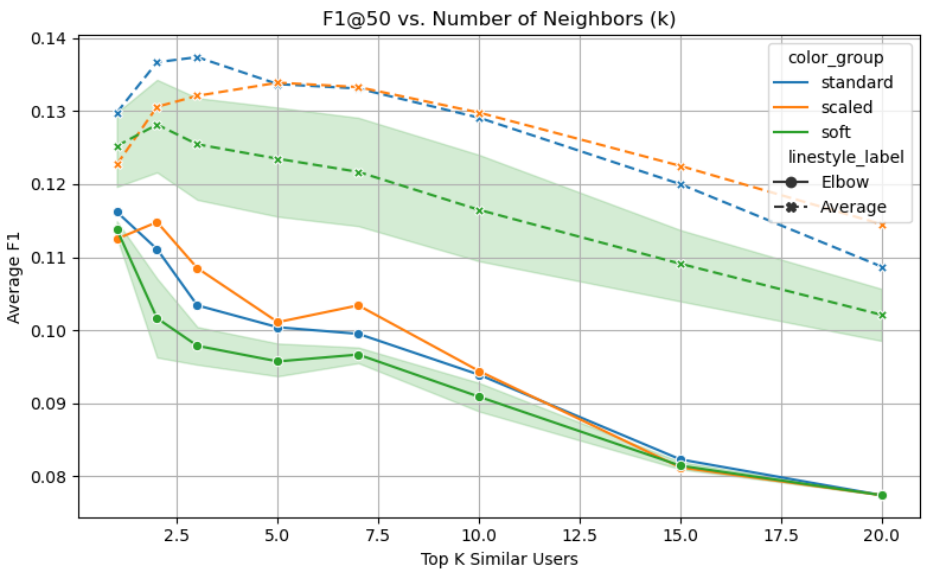




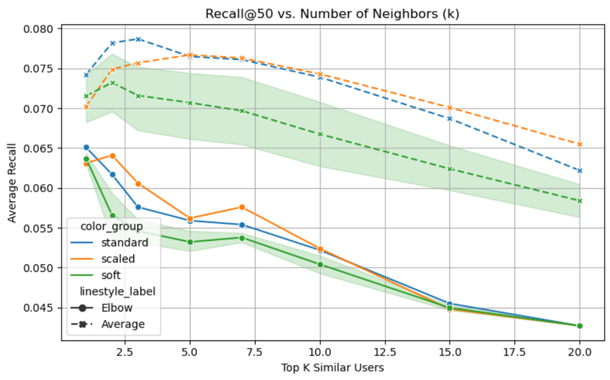
(b) 1,000 playlists: Recall@N vs N

(c) 1,000 playlists: Precision@N vs N

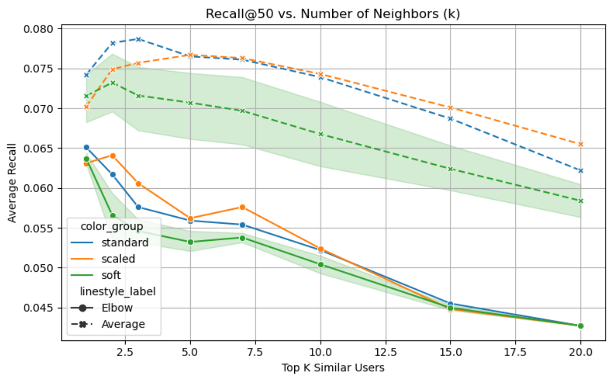
Figure 1: Varying Number of Neighbors k on 1,000 Playlist Dataset



(a) 1,000 playlists: F1@N vs k

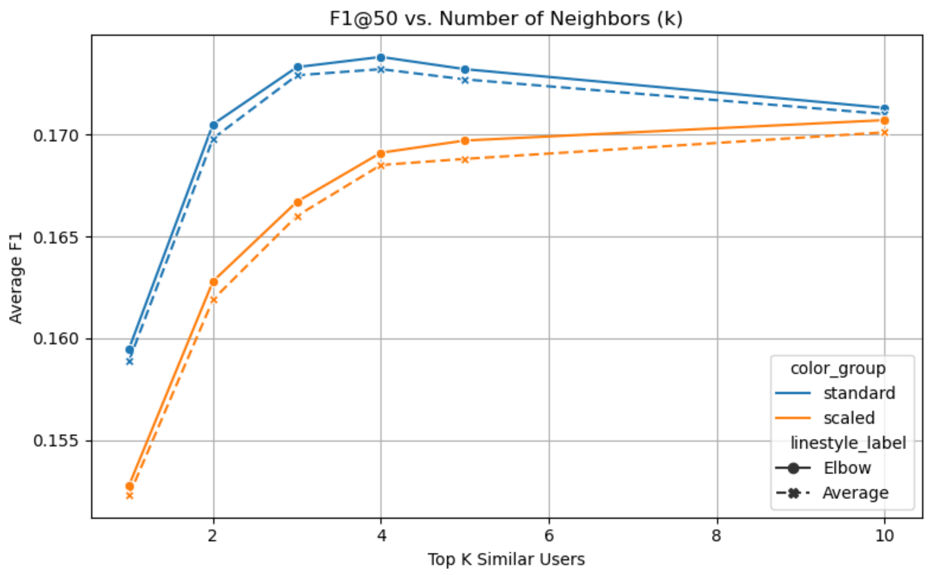


(b) 1,000 playlists: Recall@N vs k

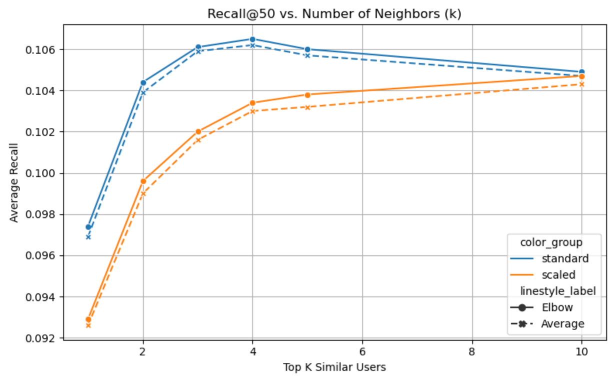


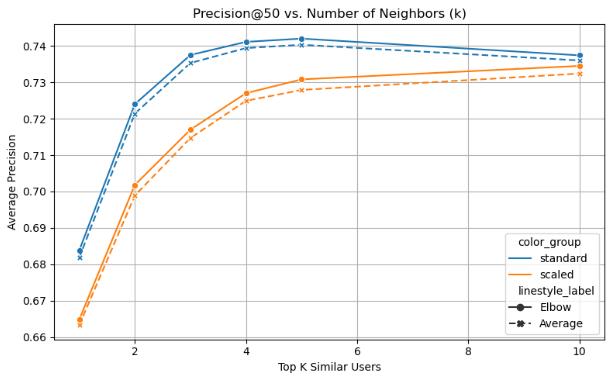
(c) 1,000 playlists: Precision@N vs k

Figure 2: Varying Number of Neighbors k on 1,000 Playlist Dataset



(a) 10,000 playlists: F1@N vs k





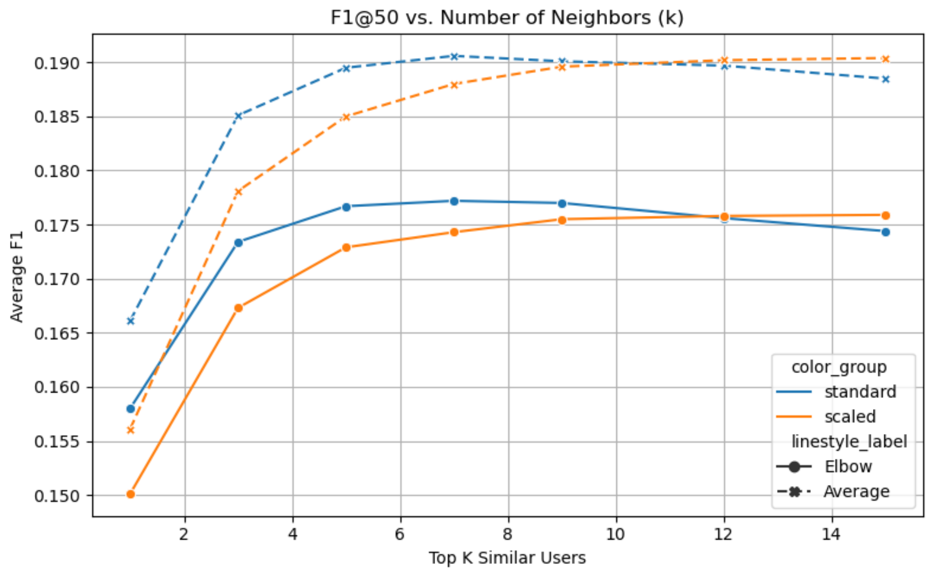
(b) 10,000 playlists: Recall@N vs k

(c) 10,000 playlists: Precision@N vs k

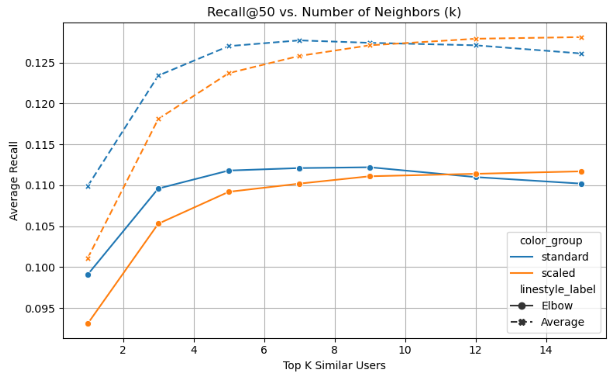
Figure 3: Varying Number of Neighbors k on 10,000 Playlist Dataset

difference being the value of metric scores. We also see a similar pattern as in the 1k dataset for average versus elbow datasets. The average outperforms for F1 and recall, but elbow outperforms for precision. The optimal k again increases slightly from the 10k to 50k dataset, suggesting that higher sample size can more consistently produce similar users at a higher volume. Standard cosine more consistently outperforms scaled cosine at lower k than in previous iterations. However, scaled cosine does begin to outperform standard around $k = 15$, but only by a small margin. These results help us to further refine the parameter ranges. A final set of eight model configurations was executed, with the results stored in *Table 2*.

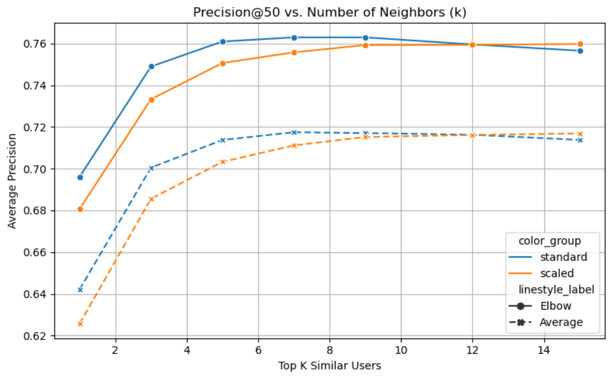
Among the final eight models in *Table 2*, the strongest overall performer used standard cosine similarity, the average-based 50k dataset, and $k = 7$ nearest neighbors. This combination achieved the highest F1 score (with N fixed at 50), while maintaining a competitive recall and precision scores, making it a compelling choice for general-purpose user similarity modeling.



(a) 50,000 playlists: F1@N vs k



(b) 50,000 playlists: Recall@N vs k



(c) 50,000 playlists: Precision@N vs k

Figure 4: Varying Number of Neighbors k on 50,000 Playlist Dataset

User Similarity Conclusion This component of the project clustered playlists into synthetic users and compared their artist profiles using variations of cosine similarity. Several strategies were explored at various steps, ultimately to apply a final set of parameters on the 50k dataset for model selection. First, it was found that standard cosine outperformed the others. Second, scores generally peaked when using fewer than 10 similar users (k), indicating that smaller, more curated neighborhoods yield stronger recommendations.

Dataset	Method	Alpha	Train	Test	k	Recall@	Precision	Recall	F1
avg_50k	standard	None	2173	2173	7	50	0.7175	0.1277	0.1906
avg_50k	standard	None	2173	2173	8	50	0.7178	0.1279	0.1905
avg_50k	standard	None	2173	2173	9	50	0.7171	0.1274	0.1901
avg_50k	standard	None	2173	2173	15	50	0.7138	0.1261	0.1885
elbow_50k	standard	None	1761	1761	8	50	0.7635	0.1123	0.1773
elbow_50k	standard	None	1761	1761	7	50	0.7630	0.1121	0.1772
elbow_50k	standard	None	1761	1761	9	50	0.7630	0.1122	0.1770
elbow_50k	standard	None	1761	1761	15	50	0.7566	0.1102	0.1744

Table 2: Final Model results (50,000 playlists).

Third, performance improved steadily with sample size which demonstrates the scalability and robustness of this model. Lastly, while the average-based dataset had a slight edge overall, the choice ultimately depends on the model's intended use: broader versus more conservative similarity matching. If the goal is high volume similarity suggestions, the average dataset is preferred due to its higher F1 and recall scores, which reflect stronger coverage and general matching ability. Conversely, the higher precision score seen in the elbow dataset suggests its better suited for more curated or conservative user similarity matching.

5 References

- Soundplate. "How Many Playlists Are There On Spotify? (and other Spotify stats) – 2024 Update!" `https://soundplate.com/how-many-playlists-are-there-on-spotify-and-other-spotify-stats/`
- Kaggle. "Spotify Million" `https://www.kaggle.com/datasets/himanshuwagh/spotify-million/data`