Understanding Spotify's Hybrid Music Recommendation System and Its Limitations in Closed Music Libraries

Nathaniel Christian - 13524122[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524122@itb.ac.id, [2]nathaniel.christian@gmail.com

***Abstract*—Modern music platforms like Spotify rely on recommendation algorithms to personalize listening experiences for millions of users. These systems integrate mathematical models—such as matrix factorization and vector-space similarity—with large-scale machine learning architectures. This paper presents a structured study of music recommendation systems by examining foundational recommendation approaches and their application in Spotify's recommendation pipeline. We then contrast these techniques with the constraints of closed music environments where global user data is unavailable, and highlight algorithmic adaptations suitable for such systems.**

***Keywords*—Music Recommendation System, Spotify, Closed Music Systems, Matrix Decomposition.**

I. Introduction

Digital music platforms have changed how people perceive and consume music since their emergence in the late 1900s with the introduction of peer-to-peer file-sharing systems, most notably Napster. Widespread adoption followed in the mid-2000s with the rise of legal, subscription-based music streaming services such as Spotify. As music catalogs expanded to tens of millions of tracks and user bases grew exponentially, many platforms delved into personalization battle to reach and open up to many more audiences [7].

Although Spotify appears to be a simple music player on the surface, it operates on top of complex algorithms designed to model user behavior and deliver personalized recommendations. These algorithms, commonly referred to as recommender systems (RSs), form an algorithmic pipeline that predicts a user's musical preferences by identifying patterns within large-scale interaction data. At its core, such systems rely heavily on mathematical structures and computations from linear algebra, including vector spaces, matrix decompositions, and eigen-based techniques [3].

Modern music streaming platforms operate at a scale defined by large user bases and continuous streams of interaction data. Personalization in such systems is therefore driven by the analysis of user's listening behavior, from which recurring patterns across the user population can be inferred.

Despite the effectiveness of such large-scale recommendation systems, their underlying assumptions do not generalize to all settings. In closed music environments, such as self-hosted music servers or personal digital libraries, global user interaction data is unavailable. As a result, personalization strategies designed for data-rich platforms cannot be directly applied and must be reconsidered under different modeling constraints.

This paper presents a structured study of music recommendation systems by first examining the algorithmic foundations underlying Spotify's large-scale recommendation architecture and then contrasting them with the constraints of closed music environments. The goal is to identify algorithmic insights that remain applicable when operating under limited data availability.

II. Spotify Recommendation

Spotify's recommendation system is designed to operate at a global scale, serving millions of users with different musical preferences and interaction behaviors. To achieve this, Spotify uses a hybrid recommendation system that combines two primary approaches, namely collaborative filtering and content-based filtering [1].

A. Collaborative Filtering

Collaborative filtering forms the foundation of Spotify's recommendation hybrid system. It leverages collective listening behavior to identify similarities between users and tracks, under the assumption that users who have similar listening and interaction patterns are likely to share musical tastes.

Intuitively, collaborative filtering assumes that if two users show similar interaction patterns over a subset of tracks, then tracks favored by one user but not yet consumed by the other are likely to be relevant

recommendations. For example, if user A has enjoyed songs X, Y, and Z, and user B has enjoyed songs X and Y but has not yet listened to song Z, then song Z is a strong recommendation candidate for user B.

User behavior is usually represented as a user-item interaction matrix R, where rows correspond to users and columns correspond to tracks [5]. Each entry $r(u,i)$ corresponds to the strength of a specific interaction between $user\ u$ and $track\ i$, such as play counts, skips, playlist additions, or anything else—which differs depending on the algorithm and its objective. Formally, this matrix can be written as:

$$R_{u,i} = r(u,i) \tag{1}$$

with $u \in \{1, \dots, m\}$ users and $i \in \{1, \dots, n\}$ tracks.

In practice, this matrix is extremely sparse, as each user interacts with only a small fraction of the available music collection. This approach was originally popularized in RSs for movies, but it has since been adapted and improved to fit the music RSs [9].

To better understand this approach, it is necessary to examine how Spotify represents user behavior. Essentially, Spotify's algorithm assigns every user something called a user profile—often called user taste profile—to simplify the data collection process. The user profile is generated from the user feedback, which can be split into two categories [8]:

- Explicit feedback, such as saving tracks, adding songs to playlists, or following artists, provides strong positive signals.
- Implicit feedback, including listening duration, repeat plays, and session-level behavior, offers additional but noisier information.

Spotify interprets this feedback in a context-aware manner; for example, a skipped track during exploratory browsing does not carry the same negative weight as a skip within a focused playlist. These signals are aggregated and weighted to construct the user taste profile, which is basically a structured representation of a listener's musical preferences across genres, moods, popularity levels, and listening contexts.

Early collaborative filtering systems relied on nearest-neighbor methods, where similarity between users was computed directly from their interaction vectors. Recommendations were generated by averaging preferences from similar users. While intuitive, this approach scales poorly for large user bases and high-dimensional interaction data.

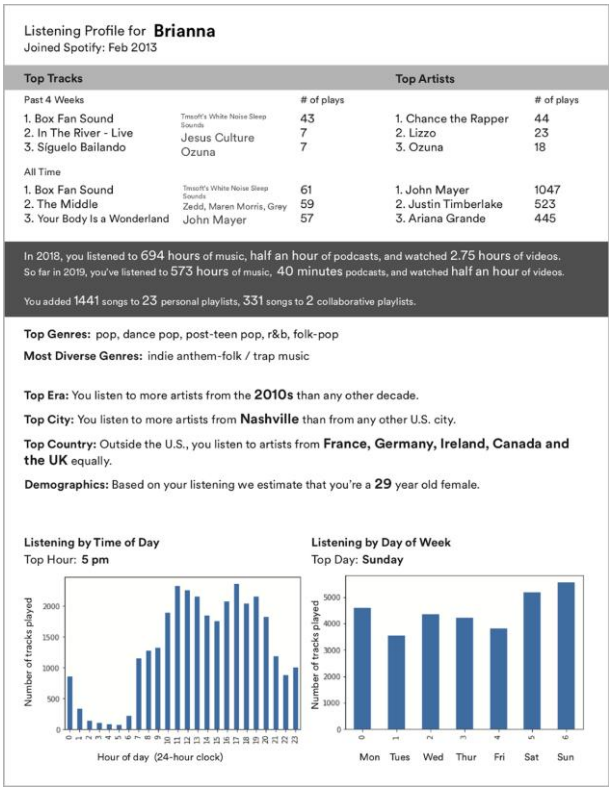

Fig. 1. Example of a context-aware Spotify user profile illustrating listening preferences across tracks, artists, genres, and temporal patterns. The figure shows a subset of user profile information; underlying model representations are not shown. *Note: Image taken from [8].*

To address the scalability and sparsity problem, Spotify uses models based on matrix factorization. In this formulation, the interaction matrix R is approximated by the product of two low-rank matrices:

$$R \approx UV^T \tag{2}$$

Here, U is a user matrix where each row represents a vector describing a single user, and V is a track matrix where each row represents a vector describing a single track. The dimension k is selected to be significantly smaller than the number of users and tracks, producing a low-dimensional representation of the original interaction matrix. This dimensionality reduction reduces computational complexity while preserving the most relevant preference patterns learned from listening behavior, rather than relying on explicitly defined musical attributes.

With this information, now we can compute the predicted preference score between a user and a track is computed using an inner product:

$$\hat{r}(u,i) = u \cdot v \tag{3}$$

Tracks which the vectors are most aligned with a user's vector receive higher predicted scores and are selected as recommendation candidates.

To efficiently learn these low-dimensional representations from large-scale implicit feedback data, Spotify uses optimization techniques such as Alternating Least Squares (ALS). ALS iteratively fixes one factor

matrix while solving for the other by minimizing a regularized least-squares objective. By alternating between updating the user matrix U and the track matrix V, this method enables scalable training on sparse interaction matrices while maintaining numerical stability.

Through matrix factorization and vector-space similarity, collaborative filtering enables Spotify to recommend tracks a user has never previously encountered by exploiting patterns observed across the broader user population.

While matrix factorization provides a useful abstraction for understanding Spotify's recommender system, production systems rely on more complex neural embedding and ranking architectures that extend beyond classical linear models.

In hindsight, the core idea mirrors the long-standing human practice of sharing opinions and preferences within a community. Collaborative filtering allows us to do that much faster and not limited to a small number of people [6].

B. Content-based Filtering

Content-based filtering, on the other hand, focuses on the characteristics of the music audio itself [6]. In this approach, each song is represented as a feature vector composed of multiple audio and metadata attributes. These features, however, are not fixed and may vary depending on the extraction algorithm and modeling objective. Based on [1, 2], features like artist, genres, relevance, temporal context (e.g., time of day), repeated tracks, language, song length, habits, tempo, and many others curates how Spotify gives recommendation.

Once tracks are represented in this feature space, recommendations are produced by computing similarity between tracks or between a user's listening profile and possible similar tracks that, in theory, are likely to match the user's preferences. Unlike collaborative filtering, this approach does not depend on collective user behavior and can operate solely on the properties of the music itself.

Formally, each track i is mapped to a vector:

$$x_i \in \mathbb{R}^d \tag{4}$$

where d represents the dimension of the feature space. Recommendation then becomes a problem of vector similarity within this space, under the assumption that musically similar tracks are located near each other.

A common similarity measure used in content-based filtering is cosine similarity, defined as:

$$sim(i,j) = \frac{x_i \cdot x_j}{\| x_i \| \cdot \| x_j \|} \tag{5}$$

Cosine similarity captures the similarity between feature vectors in the feature space. Tracks with higher cosine similarity scores are considered stronger recommendation candidates.

To create personalized recommendations, a user's preferences can be modeled by aggregating—combining multiple track feature vectors from a user's listening history into a single vector representation—the feature vectors of tracks the user has previously interacted with. For example, a user profile vector may be constructed as a weighted combination of feature vectors corresponding to the user's listening history.

This can be written as:

$$p(u) = \frac{\sum_{i \in H(u)} w(u,i) \cdot x_i}{\sum_{i \in H(u)} w(u,i)} \tag{6}$$

where:

- $H(u)$ denotes the set of tracks previously interacted with by user u,
- $w(u,i)$ represents the interaction weight between user u and track i, such as play frequency, recency, or other engagement-based signals,
- x_i denotes the feature vector of track i.

Candidate tracks are then ranked based on their similarity to this profile.

One advantage of content-based filtering is its ability to recommend new or less popular tracks, as it does not depend on collective listening behavior. This property makes it particularly effective in cold-start scenarios, where interaction data is sparse. However, content-based approaches are limited by the expressiveness of the chosen feature space and may struggle to capture higher-level preference patterns that emerge from collective user behavior.

In large-scale music platforms such as Spotify, content-based filtering serves as a complementary component to collaborative filtering. By grounding recommendations in audio-level similarity, it provides robustness in situations where behavioral data alone is insufficient, and plays a crucial role in the system's hybrid recommendation pipeline.

C. Hybrid Recommendation Pipeline

In practice, large-scale music platforms such as Spotify do not rely on a single recommendation approach. Instead, collaborative filtering and content-based filtering are integrated within a hybrid recommendation pipeline, allowing the system to exploit the strengths of each method while mitigating their individual limitations.

At a high level, Spotify's recommendation process can be viewed as a multi-stage pipeline consisting of candidate generation and ranking. In the candidate generation stage, multiple models independently propose a set of potentially relevant tracks for a user. Collaborative filtering models contribute candidates based on latent user–track similarity learned from collective listening behavior, while content-based models contribute candidates based on audio-level and metadata similarity. Additional heuristic or context-driven sources, such as recent listening history or editorial playlists, may also be included.

Each candidate source operates within its own vector space. Collaborative filtering produces latent embeddings for users and tracks, while content-based filtering produces feature vectors derived from audio and metadata. As a result, a single user may be associated with multiple representations simultaneously. Rather than collapsing these representations into a single model, Spotify combines their outputs at the pipeline level.

Once a candidate set is generated, tracks are passed to a ranking stage, where they are scored and ordered according to their estimated relevance. While the exact ranking models used by Spotify are proprietary, the core idea remains mathematically grounded: relevance scores are computed using vector similarity measures, weighted interaction signals, and contextual features such as time of day, listening device, and session intent.

To support real-time recommendation at scale, Spotify relies on efficient vector retrieval mechanisms. Rather than comparing a user vector against the entire track catalog, approximate nearest-neighbor search is used to retrieve the most relevant candidates from high-dimensional embedding spaces. This design allows the system to maintain low latency while operating over millions of tracks and users.

The hybrid pipeline plays a crucial role in addressing common recommendation challenges. Collaborative filtering excels at capturing collective trends but struggles in cold-start scenarios, while content-based filtering can recommend new or unpopular tracks but may lack personalization depth. By combining both approaches, Spotify achieves a balance between discovery, personalization, and robustness.

This hybrid design, however, implicitly assumes access to large-scale user interaction data and a rich ecosystem of behavioral signals. In environments where such assumptions do not hold—such as closed music systems with limited users and interaction history—the effectiveness of this pipeline must be reconsidered. This limitation motivates the exploration of alternative recommendation strategies tailored to closed and data-constrained settings.

III. Closed Music Library

A. Definition and Characteristics

A closed music library refers to a music environment in which the available content and user interaction data are restricted to a limited, self-contained ecosystem. Unlike commercial streaming platforms such as Spotify, which operate on globally shared catalogs and aggregate behavior from millions of users, closed music systems are typically personal or community-scale. Common examples include self-hosted music servers, offline digital libraries, and local media platforms such as Navidrome.

In a closed music library, the music catalog is fixed to what the user has explicitly collected or imported, often ranging from hundreds to a few thousand tracks rather than tens of millions. User interaction data is similarly limited, frequently originating from a single user or a very small group of listeners. As a result, there is no access to large-scale collective listening behavior, cross-user similarity signals, or external popularity metrics.

From a data perspective, closed systems lack the fundamental structures that large-scale recommender systems rely on. There is no meaningful global user-item interaction matrix, no dense behavioral signals, and no long-term historical patterns spanning diverse user populations. Instead, the available data consists primarily of localized listening histories, simple interaction logs, and static metadata embedded within audio files.

B. Fundamental Constraints

The defining characteristics of closed music libraries impose several fundamental constraints that significantly alter the feasibility of conventional recommendation algorithms. These constraints arise not from implementation limitations, but from structural deficiencies in the available data.

The most critical limitation is the absence of large-scale user interaction data. Collaborative filtering relies on discovering correlations across users by analyzing shared interaction patterns. In closed systems, the number of users is often extremely small—usually limited to a single individual—making the construction of a meaningful user-item interaction matrix infeasible. Even if such a matrix is defined, it is degenerate in the sense that most similarity computations collapse due to the lack of comparable user vectors.

Formally, consider a user-item matrix R with m users and n tracks. In commercial platforms, m is on the order of millions, enabling the extraction of latent preference factors through matrix factorization. In closed environments, however, m is typically very small, and usually $m = 1$. Under these conditions, factorizing R into meaningful latent representations becomes ill-posed, as there is insufficient data to infer shared preference structure. As a result, collaborative filtering loses its predictive power.

A second major constraint is extreme sparsity. Even within a small music library, a user interacts with only a subset of tracks, resulting in interaction vectors with very few non-zero entries. This sparsity is not mitigated over time by new users or diverse interaction patterns, as is the case in large-scale systems. Consequently, similarity measures become unstable and highly sensitive to noise, leading to unreliable recommendations.

Closed music systems also lack access to external signals commonly used in large-scale recommendation pipelines. Popularity statistics, global trends, demographic priors, and cross-domain behavior—such as podcast or social activity—are unavailable. This absence eliminates many of the additional features that help stabilize and

regularize recommendation models in commercial platforms.

Finally, the cold-start problem becomes the default operating condition rather than an edge case. New tracks added to a closed library have no associated interaction history, and users may explore their collection irregularly or non-uniformly. Without collective feedback, the system cannot infer relevance through behavioral propagation, forcing recommendations to rely primarily on intrinsic properties of the music or short-term listening context.

Taken together, these constraints demonstrate that recommendation in closed music environments is not merely a scaled-down version of large-scale systems. Instead, it constitutes a distinct problem setting in which assumptions underlying collaborative filtering and large-scale latent modeling no longer hold. This motivates the need for alternative algorithmic approaches specifically designed to operate under limited data availability and localized interaction histories.

C. Implications for Recommendation Algorithms

The structural constraints of closed music environments directly affect the applicability and effectiveness of standard recommendation algorithms. Since the assumptions underlying large-scale recommender systems no longer hold, many widely used techniques either degrade significantly in performance or become infeasible altogether.

As a direct consequence of these constraints, collaborative filtering—both in its user-based and latent factor formulations—ceases to function as a general-purpose recommendation mechanism in closed systems.

Hybrid recommendation pipelines, such as those used by Spotify, also become difficult to replicate. These systems rely on a multi-stage architecture involving candidate generation, ranking, and re-ranking, often supported by auxiliary signals such as popularity priors, global trends, and cross-domain embeddings. In closed environments, the absence of these signals removes much of the pipeline's stabilizing structure, forcing the system to operate with fewer stages and simpler decision rules.

In contrast, content-based approaches remain viable under these constraints, as they do not depend on collective user behavior. Algorithms that operate directly on audio features, metadata, and listening history can still produce recommendations by exploiting similarity relationships within the music collection itself. However, even content-based methods face limitations in closed settings. The expressiveness of the feature space becomes critical, as poorly chosen or overly shallow features may lead to repetitive or overly narrow recommendations.

Another implication is the increased importance of temporal and contextual modeling. Since long-term behavioral data is limited, short-term listening context—such as recent plays, time of day, or session-level patterns—becomes a dominant signal. Recommendation algorithms in closed systems must therefore adapt more rapidly to recent activity rather than relying on long-term preference aggregation.

Lastly, evaluation criteria must also shift. In large-scale systems, recommendation quality is often measured using metrics derived from massive interaction logs, such as click-through rates or engagement uplift. In closed environments, such metrics are either unavailable or unreliable. Instead, recommendation success must be assessed through qualitative measures such as diversity, novelty, and perceived relevance over time.

These implications indicate that closed music libraries require a fundamentally different design philosophy. Rather than optimizing for scale and global accuracy, recommendation algorithms must prioritize robustness, interpretability, and adaptability under limited data conditions. This observation motivates the exploration of algorithmic approaches specifically tailored to closed music systems, which are discussed in the following section.

IV. Approaches For Closed Systems

A. Content-based Filtering as the Core Mechanism

In closed music environments, content-based recommendation techniques transition from being a complementary component—as seen in large-scale platforms—to serving as the primary mechanism for personalization. As we've discussed, this shift is driven by the absence of large-scale behavioral data required by collaborative filtering and hybrid pipelines.

Because closed systems lack access to user-collective listening data, recommendations must be inferred directly from the properties of the music collection and the user's local listening history. Content-based methods remain effective under these conditions, as they do not rely on cross-user similarity or global popularity signals. Instead, they enable recommendation through direct comparison between tracks or between recently consumed content and the rest of the library.

However, the role of content-based methods in closed systems is fundamentally constrained. Without variation introduced by multiple users, similarity-based recommendation tends to converge toward a narrow region of the musical feature space. This results in repetitive recommendations that emphasize small stylistic differences rather than meaningful musical progression. Consequently, content-based similarity in closed environments should be treated as a foundational mechanism rather than a complete solution, requiring additional constraints to prevent recommendation stagnation.

Despite this limitation, content-based approaches remain well-suited for closed systems due to their deterministic behavior, low data requirements, and

compatibility with small music libraries. These properties make them an effective foundation upon which more adaptive mechanisms can be layered.

B. User-Centric Preference Modelling

In closed music systems, personalization must be derived from the behavior of a single user or a very small group of users. Unlike large-scale platforms that infer preferences through cross-user comparison, closed systems must model taste solely from individual listening history.

Rather than constructing a static user profile, effective closed-system recommendation represents preferences as an evolving aggregation of past interactions. Tracks can be weighted by interaction strength and historical relevance to capture stable musical tendencies while reducing sensitivity to short-term noise. Given limited data, simple aggregation methods—such as weighted combinations of track embeddings—are often more robust than complex models that require large training sets.

Listeners frequently exhibit multiple musical interests that vary across time and context. To address this, closed systems may maintain multiple lightweight preference representations derived from different segments of listening history, such as long-term trends and medium-term patterns. This approach preserves adaptability without introducing unnecessary model complexity.

C. Time-based and Context-Aware Systems

Temporal and contextual signals play a central role in closed music recommendation. With limited long-term interaction data, recent listening behavior often provides a more accurate indication of user intent than historical preference aggregation.

Closed systems benefit from explicitly modeling session-level context, including recently played tracks, repetition patterns, and listening duration [11]. These signals allow the system to infer short-term intent—such as focused listening or exploration—and generate recommendations that align with the current session rather than long-term averages.

Time-dependent listening patterns may also emerge, such as preferences tied to time of day or activity. By emphasizing recency or maintaining separate context-sensitive representations, closed systems can adapt recommendations dynamically. This temporal focus also reduces repetitive recommendations by shifting attention across different listening contexts over time.

D. Hybrid Strategies

Closed music systems can extend baseline recommendation using simple, low-cost mechanisms:

- Bounded random selection
  Introduce a small probability of selecting tracks outside the top similarity scores to avoid repetitive recommendations, while constraining selection within reasonable musical bounds (e.g., tempo or genre proximity).
- Redundancy-aware ranking
  Penalize repeated artists, albums, or highly similar tracks within short recommendation windows to prevent over-concentration.
- Context-driven heuristics
  Apply simple rules based on listening context, such as favoring smoother transitions for background listening or higher-energy tracks during active sessions.
- Hybrid models
  Combine content similarity with user-centric and temporal weighting to balance stability with responsiveness, without constructing full hybrid pipelines.

These strategies prioritize interpretability and robustness while remaining compatible with the data constraints of closed music environments.

V. CONCLUSION

This paper presented a structured study of music recommendation systems, beginning with general algorithmic principles and progressing to Spotify's large-scale implementation. By examining collaborative filtering and content-based methods through a mathematical lens, the discussion highlighted how modern recommender systems rely on large-scale interaction data, latent factor modeling, and hybrid pipelines to deliver personalized music experiences.

The analysis then shifted to closed music environments, where the assumptions underlying large-scale recommendation systems no longer hold. In such settings, the absence of global user data, extreme interaction sparsity, and limited contextual signals fundamentally constrain the applicability of collaborative filtering and complex hybrid architectures. These limitations demonstrate that recommendation in closed systems is not merely a scaled-down version of commercial platforms, but a distinct problem with different objectives and design requirements.

To address these constraints, the paper explored algorithmic approaches suitable for closed music libraries. Content-based similarity was identified as a stable foundation, while user-centric preference modeling and temporal context were shown to play a critical role in maintaining personalization and adaptability. Lightweight hybrid strategies and heuristic extensions further provide practical mechanisms for improving recommendation quality without violating closed-system constraints.

Future work may explore the implementation and evaluation of such approaches within real-world closed music platforms. In particular, designing and experimenting with adaptive, context-aware recommendation algorithms for self-hosted systems presents an opportunity to bridge theoretical models with

practical system design. As personal and decentralized music libraries continue to gain relevance, effective recommendation under limited data conditions remains an open and meaningful area for further research.

VIDEO LINK AT YOUTUBE

https://youtu.be/qINoBc0Rh28

ACKNOWLEDGMENT

The author would like to thank Dr. Ir. Rinaldi Munir and Ir. Rila Mandala for their guidance and support throughout the IF2123 Geometric Algebra course. Appreciation is also extended to fellow students for their discussions and feedback during the preparation of this work.

REFERENCES

[1] P., Dmitry. "Inside Spotify's Recommender System: A Complete Guide to Spotify Recommendation Algorithms," Music Tomorrow, 2025. [Online]. Available: https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-complete-guide.

[2] Spotify. "Made to be Found". [Online]. Available: https://found.byspotify.com/made-for-you.

[3] F. Ricci, L. Rokach, and B. Shapira, "Introduction to Recommender Systems Handbook," in Recommender Systems Handbook, F. Ricci, L. Rokach, B. Shapira, and P. Kantor, Eds. Boston, MA: Springer, 2011, pp. 1–35, doi: 10.1007/978-0-387-85820-3_1.

[4] O. Baumann, D. Nandini, A. Rossanez, M. Schoenfeld, and J. C. dos Reis, "How to surprisingly consider recommendations? A knowledge-graph-based approach relying on complex network metrics," arXiv preprint arXiv:2405.08465, May 2024.

[5] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," Proc. 10th Int. Conf. World Wide Web, pp. 285–295, 2001.

[6] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in The Adaptive Web: Methods and Strategies of Web Personalization. Springer, 2007, pp. 291–324.

[7] Castr, "History of music streaming," Castr Blog, Nov. 30, 2024. [Online]. Available: https://castr.com/blog/history-of-music-streaming/

[8] J. Wirfs-Brock, S. Mennicken, and J. Thom, "Giving voice to silent data: Designing with personal music listening history," in Proc. 2020 CHI Conf. Human Factors Comput. Syst., 2020, pp. 1-11, doi: 10.1145/3313831.3376493.

[9] Y. Deldjoo, M. Schedl, and P. Knees, "Content-driven music recommendation: Evolution, state of the art, and challenges," Comput. Sci. Rev., vol. 51, art. no. 100618, Feb. 2024, doi: 10.1016/j.cosrev.2024.100618.

[10] https://github.com/DataSorcerer/Music-Recommendation-System

[11] D. Jannach, M. Quadrana, and P. Cremonesi, "Session-based recommender systems," in Recommender Systems Handbook, 3rd ed., F. Ricci, L. Rokach, B. Shapira, and P. Kantor, Eds. Springer, 2022, pp. 301–335.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Desember 2025

Nathaniel Christian – 13524122