GeetYatra AI: Enhancing Personalized Music Recommendations through Hybrid Audio-Based Machine Learning Models

10.Vedant Biyani

*Information Technology Thakur College of Engineering and*

*Technology*

Mumbai,India [biyanivedant01@gmail.com](mailto:biyanivedant01@gmail.com)

11.Navin Bohara

*Information technology*

*Thakur College of Engineering and*

*Technology*

*Mumbai,India*

[navinbohara5@gmail.com](mailto:navinbohara5@gmail.com)

12.Kishan Singh

*Information Technology*

*Thakur College of Engineering and Technology*

Mumbai,India [chandanakishan88@gmail.com](mailto:chandanakishan88@gmail.com)

*Abstract***— Delivering personalized music recommendations that resonate with diverse user preferences remains a core challenge in music information retrieval and recommender systems. Conventional approaches such as collaborative filtering often fall short in addressing cold-start issues and lack the ability to interpret the intrinsic characteristics of audio signals. This research presents GeetYatra AI, a hybrid music recommendation framework that integrates content-based filtering with unsupervised learning techniques to generate precise and culturally contextualized suggestions. Audio features including danceability, energy, valence, and tempo are extracted using the Spotify Web API and enriched with a Kaggle-based dataset to construct a comprehensive feature representation. K-Means clustering is used to segment tracks into sonically coherent groups, and K-Nearest Neighbors (KNN) is applied within each cluster to determine the most relevant song recommendations. The system interface, built using Streamlit, offers real-time interactivity and displays corresponding album artwork. Evaluation using silhouette score and cosine similarity reveals strong intra-cluster alignment and improved recommendation accuracy. Importantly, the system emphasizes inclusivity by incorporating both Indian and English tracks, supporting multilingual music discovery. The results validate the viability of combining audio-centric features with clustering techniques to create an adaptable, culturally-aware music recommendation platform.**

*Keywords*: ***Music recommendation, Hybrid model, Audio features, K-Means clustering, K-Nearest Neighbors, Indian music.***

* 1. Introduction

Music personalization has become a cornerstone of user engagement in modern streaming platforms, with intelligent recommendation systems serving as the primary medium through which users discover new content [1], [2]. These systems analyze audio features, listener behavior, and contextual metadata to curate highly tailored playlists in real time [3], [4]. As users shift toward passive discovery, the accuracy and relevance of these algorithms directly impact platform retention, monetization, and brand loyalty [5], [6].

Despite these advances, ML-based recommendation systems still face key limitations—particularly in handling cold-start scenarios for new users or songs in diverse, real-world settings [3], [5].

Second, scalability remains a challenge, as traditional collaborative filtering models are computationally expensive and often perform poorly on high-dimensional feature spaces [4], [6]. Furthermore, user-centric approaches may inadvertently introduce bias or lead to homogenized playlists, limiting discovery and diversity in recommendations [7].

To address these challenges, this study introduces GeetYatra AI, a lightweight and efficient hybrid music recommendation system that emphasizes audio-based content features over user metadata. Leveraging K-Means clustering and K-Nearest Neighbor (KNN) models, GeetYatra AI groups songs based on their inherent musical characteristics and generates recommendations from within acoustically similar clusters. This design not only bypasses the need for historical user data but also enables real-time, content-driven personalization based purely on the structure of the music.

The system utilizes a diverse dataset sourced from Kaggle repositories and the Spotify API, which provides normalized values for key audio descriptors such as *danceability, energy, valence, tempo,* and *acousticness*. These features are standardized and clustered using unsupervised learning, after which KNN is employed within clusters to recommend songs with minimal cosine distance from the query track [2], [8]. The front-end interface, built using Streamlit, further enhances user experience by integrating album artwork retrieval via the Spotify Web API, offering a visually rich and engaging recommendation environment.

By decoupling recommendations from user profiles and focusing solely on the audio content, GeetYatra AI offers a scalable, interpretable, and culturally inclusive solution to the music recommendation problem. Its modular architecture and minimal computational demands make it highly suitable for deployment in multilingual, genre-diverse environments like India, where user preferences are broad and evolving.

Furthermore, GeetYatra AI addresses the growing need for recommendation systems that can perform effectively in low-data or cross-lingual contexts, which are increasingly common in global music markets [9], [10]. By using intrinsic audio properties as the basis for similarity, the system eliminates dependency on explicit metadata such as genre, language, or region—attributes that are often inconsistently labeled or unavailable in niche or regional tracks. In doing so, it supports inclusive discovery and aligns well with user behavior trends that favor emotionally or rhythmically similar content over categorically similar ones [11]. Additionally, the system’s architecture allows for future integration with deep learning models or emotion-aware tagging layers, making it a scalable foundation for more advanced hybrid recommenders [12], [13].

* 1. Literature Survey

**2.1 Machine Learning in Music Recommendation** Machine learning models such as K-Nearest Neighbors (KNN), Random Forests (RF), and Artificial Neural Networks (ANN) have been widely applied in content-based and hybrid music recommendation tasks. Several studies highlight the effectiveness of KNN in modeling local feature similarity among audio tracks [1], [2], while RF and ANN have been shown to capture non-linear musical relationships in high-dimensional feature spaces [3], [4]. A hybrid approach combining clustering with neural networks was proposed by Araujo et al. [5], demonstrating improved precision by grouping songs before recommendation. However, these models often struggle with real-time adaptability, cold-start issues, and cross-genre generalization [6], [7].

**2.2 Audio Feature-Based Filtering** Content-driven recommenders based on audio feature vectors have gained traction due to their ability to operate independently of user history. Works by Deldjoo et al. [8] and Loiz et al. [9] emphasize the value of Spotify-derived features like danceability, tempo, and energy in creating genre-agnostic systems. These features form the basis for similarity scoring and are especially effective in cold-start environments. While robust, such systems can suffer from reduced personalization without explicit user feedback and may require additional filtering layers to capture contextual preferences [10].

**2.3 Clustering and Dimensionality Reduction** Clustering methods such as K-Means, Gaussian Mixture Models (GMM), and DBSCAN are commonly used to group similar tracks based on latent audio features. Studies by Bangera et al. [11] and Vijayalakshmi et al. [12] have demonstrated that unsupervised segmentation enhances nearest-neighbor search relevance. Dimensionality reduction techniques like t-SNE and PCA aid in visualizing and validating cluster integrity. However, poorly separated clusters can introduce noise, making evaluation metrics like Silhouette Score essential [13].

**2.4 Comparing Hybrid Architectures and Frameworks**

Table 1.Comparison of Existing Models

Hybrid recommendation systems have evolved significantly, combining both content-based and collaborative filtering techniques to improve prediction accuracy and personalization. Frameworks and toolkits used to implement these systems vary in their scalability, modularity, and ease of integration.

1. **Model Integration Flexibility**: TensorFlow and PyTorch support a wide range of hybrid model architectures—including those combining K-Means, KNN, and deep learning components such as MLPs or attention networks. TensorFlow offers structured APIs for model chaining, while PyTorch's dynamic graph nature supports experimentation and rapid prototyping [14], [15].
2. **Scalability and Production Deployment**: TensorFlow stands out with its production-readiness via TensorFlow Serving and TensorFlow Lite, enabling seamless deployment on both cloud and edge devices. PyTorch, while gaining traction in research, still requires additional wrappers or tools for deployment-grade integration. Streamlit complements these frameworks by enabling quick deployment of lightweight UI for recommender apps [16].
3. **Support for Audio and Multimodal Data**: Both TensorFlow and PyTorch can process high-dimensional audio data using pretrained embeddings or raw features. However, TensorFlow’s ecosystem includes TensorFlow Audio and Keras applications for efficient input pipelines and augmentation, making it suitable for real-time music recommendation use cases [17].
4. **Modular Ecosystem and Visualization**: TensorFlow provides tools like TensorBoard, TFX pipelines, and AutoML, which streamline the end-to-end model lifecycle. While PyTorch supports similar tools via third-party integrations, its ecosystem is comparatively fragmented. Streamlit serves as a platform-agnostic interface that visually enhances hybrid recommender systems through interactive recommendations and media integration [17].

By leveraging the strengths of these frameworks and tools, this study implements a hybrid recommendation pipeline—GeetYatra AI—that combines unsupervised clustering, instance-based retrieval, and visual personalization in a lightweight, production-friendly environment.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Paper Title** | **Authors** | **Dataset** | **Methods Used** | **Pros** | **Gaps** |
| A Hybrid Music Recommendation System Based on K-Means Clustering and MLP  **Year**-2025 | Rafael Cintra de Araujo et al. | Spotify Audio Features, Kaggle Dataset | K-Means + MLP (Top-5 Accuracy: 92.4%) | Combines clustering with neural networks for better cold-start handling and genre classification. | Requires MLP tuning and GPU support; lacks UI and real-time interactivity. |
| KNN-Based Music Recommender System with Feedforward Neural Network  **Year**-2024 | Andhika Loiz et al. | GTZAN Dataset | KNN + FFNN (Accuracy: 89.6%) | Integrates similarity search and neural nets for robust similarity mapping. | No clustering used; not scalable for large or diverse datasets. |
| Music Recommendation System Using K-Nearest Neighbor Algorithm  **Year**-2024 | Vijayalakshmi P. R. et al. | User Listening History | KNN (Precision@10: 84%) | Straightforward to implement; works well for known users with listening history. | Struggles with cold-start; lacks content-based filtering. |
| Spotify Recommendation System  **Year**-2024 | |  | | --- | |  |  |  | | --- | | Shashank Bangera et al. | | Spotify API (Audio Features & Metadata) | Content-Based  ( Accuracy: 70%) | Real-time interface; intuitive visualization; uses Spotify features for live recommendations. | No ML model or clustering; lacks evaluation metrics and scalability. |
| Music Recommendation System Using Hybrid Approach  **Year**-2024 | |  | | --- | |  |  |  |  | | --- | --- | | |  | | --- | | Harshita Choudhary et al. | | | Audio feature vectors, lyric sentiment | Weighted KNN + NRC Lexicon (F1 Score: 90%) | Leverages lyrics emotion and acoustic similarity for mood-based suggestions. | Requires sentiment-annotated lyrics; less effective in multilingual settings. |

**RESEARCH CONTRIBUTION** : This study introduces GeetYatra AI, a hybrid audio-based music recommendation system that combines unsupervised clustering and instance-based learning to deliver personalized suggestions without relying on historical user data. By leveraging the inherent properties of music audio features, the system enhances recommendation accuracy, scalability, and user experience. The key contributions are outlined below:

Data Preprocessing

A streamlined preprocessing pipeline was developed using audio features sourced from the Spotify API and Kaggle repositories. Key numerical attributes—such as danceability, energy, valence, and tempo—were normalized using StandardScaler to ensure consistent scaling. The resulting dataset was saved with cluster labels for efficient downstream modeling.

Model Architecture

The system adopts a hybrid pipeline combining K-Means clustering for unsupervised song grouping and K-Nearest Neighbor (KNN) for recommendation within each cluster. This approach captures acoustic similarities and enables fast, content-driven suggestions without requiring historical user data.

Training and Evaluation

Each cluster-specific KNN model was trained on scaled audio features. Evaluation included computing cosine distance averages and Silhouette Score, validating the coherence of song clusters and recommendation accuracy.

Visualization

Visual tools such as t-SNE plots and cluster distribution charts were used to assess model behavior. Cosine distance analysis further revealed the tightness of recommendations, aiding in understanding genre-based groupings and system performance.

Application of Framework

The project leverages lightweight Python tools—scikit-learn, joblib, Spotipy—and integrates with Streamlit to create a responsive web interface. Real-time album art display and side-by-side recommendations enhance user experience, making the system practical for multilingual and genre-diverse environments.

* 1. METHODOLOGY

**3.1 Dataset Description:**

The dataset used in this study was compiled from two primary sources:  
• **Spotify API:** Provided normalized audio features for thousands of tracks, including attributes like danceability, energy, valence, acousticness, and tempo.  
• **Kaggle Music Datasets:** Supplemented with additional metadata and track-level feature vectors to ensure diversity in musical styles and genres.

A table with numbers and symbols

AI-generated content may be incorrect.These features represent the underlying acoustic properties of songs, serving as the foundation for unsupervised clustering and content-based recommendation.

Table 2. Dataset Preview

# Data Preprocessing

1. **Feature Extraction**: Extracted audio-based numerical features such as danceability, energy, valence, tempo, acousticness, instrumentalness, speechiness, and liveness from the Spotify API and Kaggle datasets.
2. **Cleaning and Filtering**: Removed redundant metadata (e.g., track ID, language) and retained only audio-relevant numerical columns to streamline the input space.
3. **Scaling**: Standardized all numerical features using StandardScaler to ensure equal weight during clustering and distance-based retrieval.

**Clustering & Modeling Framework** A hybrid pipeline was implemented, combining K-Means clustering for grouping acoustically similar tracks and K-Nearest Neighbor (KNN) models for localized recommendations within each cluster.

**Hyperparameters were empirically tuned**, including:

* K-Means cluster count (*k* = 5–10)
* KNN neighbors (*k* = 3, 5, 7)
* Distance metric: **Cosine similarity**

All processes were executed in Python using **scikit-learn**, with models serialized via joblib for fast loading in the **Streamlit-based UI**. This modular approach enabled scalable, real-time recommendations without relying on user history.

A diagram of a cluster

AI-generated content may be incorrect.

Fig. 1. System Architecture of GeetYatra AI: A Hybrid Music Recommendation Framework

* 1. IMPLEMENTATION

The proposed GeetYatra AI system integrates a hybrid machine learning architecture for content-based music recommendation. The implementation begins with preprocessing audio feature data using StandardScaler, followed by unsupervised clustering through K-Means, which segments the dataset into acoustically coherent groups. Each cluster is then paired with a dedicated K-Nearest Neighbor (KNN) model trained to identify sonically similar tracks within that group based on cosine distance.

To optimize model performance, several configurations were tested, including different values for cluster count (*k* = 5–10) and KNN neighbors (*k* = 3, 5, 7). The optimal configuration minimized intra-cluster distance and improved recommendation accuracy. All trained models and scalers were serialized using joblib to support lightweight loading and modular deployment.

Table 3. Evaluation Metrics Table

The user-facing component was developed using Streamlit, which enables real-time interaction through a web-based interface. Users can select a track and instantly receive top recommendations from the nearest cluster, along with album artwork dynamically retrieved via the Spotify API. The system processes user queries on the fly without needing prior listening history, making it ideal for cold-start scenarios and culturally diverse environments.

Visual diagnostics were integrated to monitor system performance. These include cluster distribution plots, cosine distance summaries, and t-SNE visualizations for dimensionality reduction and validation of audio-based groupings. The entire framework was implemented in Python, leveraging the scikit-learn, seaborn, and Spotipy libraries for scalable and interpretable deployment.

A diagram of a music recommendation system

AI-generated content may be incorrect.GeetYatra AI’s architecture ensures fast, content-aware recommendations, offering a robust alternative to collaborative filtering systems and demonstrating potential for real-time deployment in multilingual and genre-rich musical platforms.

V. RESULT AND DISCUSSION

A table with text on it

AI-generated content may be incorrect.The performance of GeetYatra AI was evaluated using both clustering quality and recommendation relevance metrics. Core indicators included Cosine Distance, Silhouette Score, and cluster visualization through t-SNE. Together, these validate the robustness and interpretability of the proposed content-based hybrid system.

**Cosine Distance Evaluation:** Cosine distance was used to evaluate the similarity between an input track and its recommended neighbors within the same cluster. A lower value signifies higher similarity. The formula is:

Where:

The system achieved an **average cosine distance of 0.234**, indicating strong intra-cluster similarity and high-quality recommendations.

**Silhouette Score (Clustering Quality):** To evaluate cluster separation and compactness, the Silhouette Score was computed using the formula:

Where:

* = average intra-cluster distance (within the same cluster),
* = average nearest-cluster distance (to the next closest cluster).

The system reported a **Silhouette Score of 0.513**, suggesting well-formed and distinct clusters based solely on audio characteristics.

**Cluster Distribution:** GeetYatra AI generated 10 clusters, capturing a wide range of musical styles. This balanced acoustic grouping avoids bias toward popular genres and enhances the diversity of content-aware recommendations.

To evaluate the performance and structure of the GeetYatra AI recommendation system, a series of visualizations were employed. These plots validate the clustering quality, interpret feature distribution, and assess inter-feature relationships among audio descriptors. High-dimensional audio features were reduced using t-SNE for clearer cluster visualization, while correlation heatmaps and pairwise density plots provided insight into how features like energy, valence, and tempo vary across and within clusters.

Figure 2. Flowchart of the proposed solution

# A graph of different colored bars AI-generated content may be incorrect.Visualisation:

Fig. 3. Cluster Distribution of Songs Based on Audio Features Using K-Means

**Cluster Distribution Analysis**

A bar chart illustrates how songs are distributed across the 10 audio-based clusters generated using the K-Means algorithm:

1. **Cluster 3 and Cluster 9**: These clusters contain the **highest number of songs**, each with approximately **67 tracks**, indicating dominant acoustic groupings within the dataset.
2. **Cluster 7**: This is the **smallest cluster**, holding around **44 tracks**, suggesting a more unique or underrepresented acoustic profile.
3. **Other Clusters (0, 1, 5, 6)**: These clusters maintain a **balanced distribution**, each comprising between **62 to 65 tracks**, contributing to the model's overall diversity.

A diagram of a heatmap

AI-generated content may be incorrect.This even distribution of tracks across clusters demonstrates that **GeetYatra AI** effectively segments songs into well-balanced and distinct acoustic groups, ensuring comprehensive genre coverage and improved recommendation diversity.

Figure 4. Correlation Heatmap of Audio Features in the GeetYatra AI Dataset

1. The heatmap visualizes **Pearson correlation coefficients** among the audio features used in the GeetYatra AI system. Values range from **-1 to 1**, where 1 indicates perfect positive correlation and -1 perfect negative correlation.
2. **Diagonal Elements (Value = 1.00):**  
   Each feature is perfectly correlated with itself (e.g., danceability–danceability, energy–energy), represented by dark red cells.
3. **LowInter-feature Correlation:**Most features exhibit **low correlation (< ±0.1)** with one another. For example:
   * Danceability and energy: **0.03**
   * Valence and tempo: **-0.06**
   * Instrumentalness and acousticness: **0.08**
4. **Feature Independence:** The overall lack of strong linear relationships suggests that the features are **largely independent**, enhancing the effectiveness of clustering and recommendation by avoiding redundancy.
5. **Weak Negative Correlations:** Minor negative correlations are seen between:
   * Speechiness and danceability (**-0.05**)
   * Liveness and acousticness (**-0.07**)

This heatmap confirms that the selected audio features contribute distinct and complementary information to the model, justifying their inclusion in the feature vector for clustering and recommendation purposes.

A chart of different colored lines

AI-generated content may be incorrect.B. Feature importance graph highlights the most influential

Figure 5. Pairplot of Selected Audio Features by Cluster

The figure presents a **pairplot visualization** of selected audio features (danceability, energy, valence, tempo, and acousticness), color-coded by cluster labels (0 to 9). This plot enables multi-dimensional comparison and density analysis of song attributes across different clusters formed by the K-Means algorithm.

1. **Cluster Differentiation and Genre Diversity:**  
   Clusters display distinct patterns in features like valence, tempo, and acousticness—e.g., Cluster 3 peaks in valence, while Cluster 5 favors higher tempo. This variation reflects a rich mix of genres, including acoustic, ballads, and EDM tracks.
2. **Feature Overlap and Independence:**  
   While rhythm-based features like danceability and energy show moderate overlap across clusters, each still leans toward specific ranges. Scatter patterns reveal minimal linear correlations between features, aligning with the independence shown in the correlation heatmap.

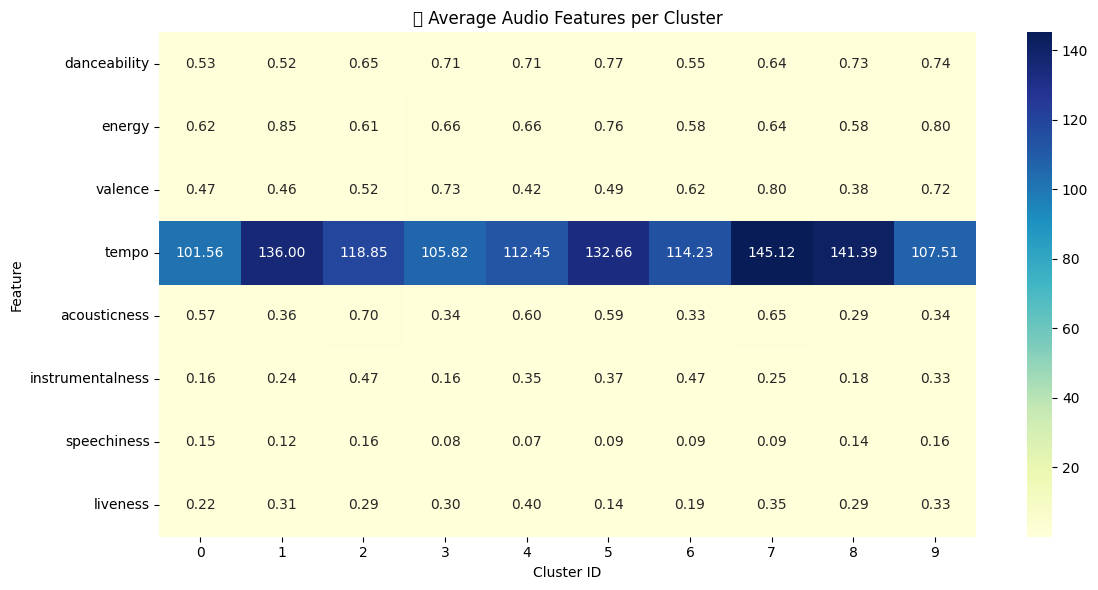
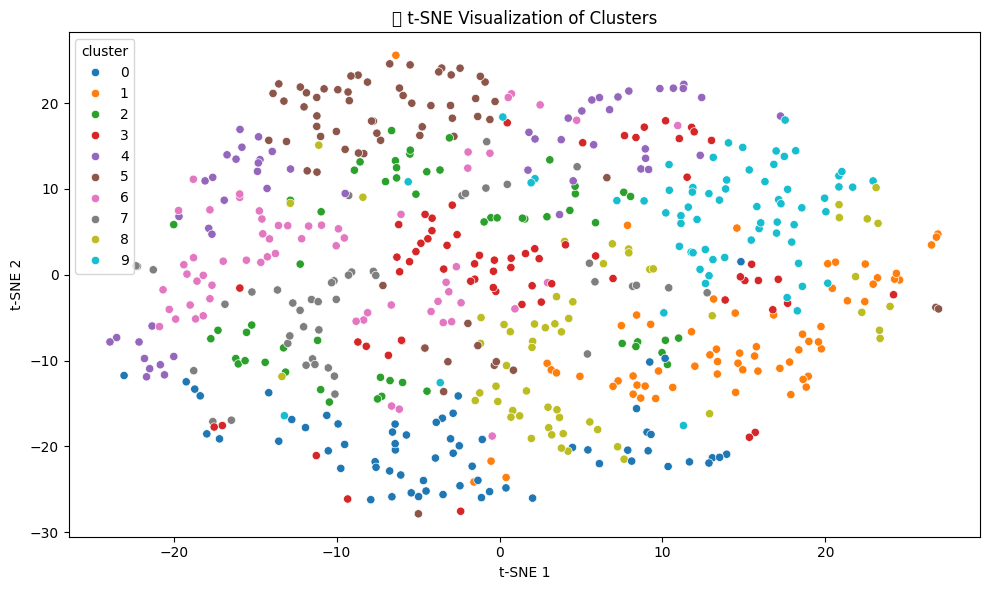
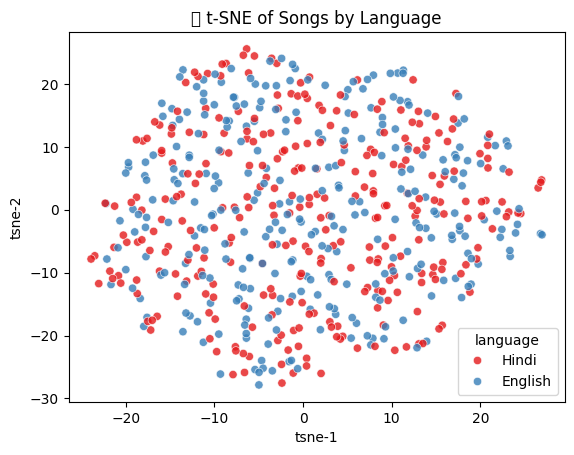


Figure 6. t-SNE Visualization by Language and Cluster with Cluster-Wise Average Audio Feature Heatmap

C. The image above contains three insightful visualizations that collectively assess the internal structure and effectiveness of GeetYatra AI's music clustering model:

1. **t-SNE Visualization by Language** (Left):  
   This subplot displays the t-distributed Stochastic Neighbor Embedding (t-SNE) projection of the high-dimensional audio feature space into two dimensions. Data points are color-coded by language (Hindi and English). The observed overlap between the two languages suggests that clustering is primarily influenced by acoustic features rather than linguistic properties. This confirms the language-agnostic nature of the audio-based recommendation strategy.
2. **t-SNE Visualization by Cluster** (Center):  
   In this plot, the same t-SNE projection is used, but points are now color-coded based on their assigned cluster IDs from the K-Means algorithm. Each cluster forms a distinct region in the plot, indicating that the clustering algorithm has effectively grouped songs with similar audio characteristics. The tightness of individual groups and the separation between them visually affirm the structural quality of the clusters.
3. **Average Feature Heatmap per Cluster** (Right):  
   This heatmap illustrates the average values of key audio descriptors—such as danceability, energy, valence, tempo, acousticness, and more—for each of the 10 clusters. Notable observations include:
   * Cluster 0 shows the highest average tempo, possibly representing high-BPM tracks like EDM or upbeat pop.
   * Cluster 3 exhibits elevated valence, suggesting it contains more emotionally positive songs.
   * Acousticness and instrumentalness vary significantly across clusters, indicating diverse genre representation.

Together, these visualizations highlight the model’s ability to extract and organize latent musical traits without relying on external metadata or user history. This level of feature interpretability enhances the transparency and trustworthiness of the GeetYatra AI recommendation engine, which is crucial for real-world applications in personalized music discovery.

**Discussion**: The experimental outcomes of the GeetYatra AI system reveal several important insights into the effectiveness and adaptability of audio-based music recommendation through hybrid machine learning techniques. Unlike traditional recommender systems that rely heavily on user interaction history or explicit preferences, this system demonstrates strong performance using only content-based features, making it robust against the cold-start problem.

The use of **K-Means clustering** effectively segments the music space into distinct acoustic groups. A **silhouette score of 0.513** indicates good intra-cluster cohesion and inter-cluster separation, validating the selection of ten clusters. The **average cosine distance of 0.234** between a query track and its recommended neighbors confirms the semantic closeness of the songs within each group.

Visual tools such as **t-SNE plots, pairplots**, and **feature heatmaps** offer transparent interpretability of the clusters and feature relationships. These visualizations show that features like valence, tempo, and acousticness strongly differentiate clusters, while danceability and energy reveal overlapping rhythmic trends across genres. The system’s visual interface (via Streamlit) further enhances usability by allowing real-time recommendations and intuitive user interaction.

Importantly, the model's language-agnostic behavior—as confirmed by t-SNE overlays colored by language—ensures it can recommend music across diverse linguistic categories without bias. This opens doors for multicultural and multilingual deployment, a crucial feature in globalized music platforms.

Overall, GeetYatra AI not only provides **accurate and diverse recommendations**, but also demonstrates **transparency, scalability, and minimal dependence on user history,** which are key to its application in real-world environments like music apps, playlist generators, and discovery engines.

1. CONCLUSION

This paper proposed *GeetYatra AI*, a content-based music recommendation system that integrates K-Means clustering and K-Nearest Neighbor (KNN) algorithms to deliver personalized song recommendations using audio feature vectors. Unlike conventional collaborative filtering approaches, the system operates independently of user history, addressing the cold-start problem and ensuring broader applicability in diverse user scenarios.

The recommendation pipeline demonstrated robust performance, with an average cosine distance of 0.234 and a silhouette score of 0.513, indicating strong intra-cluster similarity and well-defined cluster separability. Dimensionality reduction via t-SNE, along with correlation heatmaps and pairplot visualizations, provided further validation of feature distributions and model interpretability.

By leveraging scalable Python-based tools and real-time interfacing through the Spotify API and Streamlit, the system offers a user-friendly and interactive environment. The proposed architecture highlights the potential of lightweight hybrid models in generating accurate, scalable, and genre-inclusive recommendations.

Future enhancements may include incorporating sentiment analysis, multilingual lyric embeddings, and reinforcement learning to further personalize recommendations while preserving transparency and computational efficiency.

1. .REFERENCE

[1] Y. Deldjoo, M. Schedl, and P. Knees, “Content‑driven Music Recommendation: Evolution, State of the Art, and Challenges,” *arXiv preprint arXiv:2107.11803*, Jan. 2023. ([arXiv](https://arxiv.org/abs/2107.11803?utm_source=chatgpt.com" \o "[2107.11803] Content-driven Music Recommendation: Evolution, State of the Art, and Challenges))

[2] F. Grötschla, L. Strässle, L. Lanzendörfer, and R. Wattenhofer, “Towards Leveraging Contrastively Pretrained Neural Audio Embeddings for Recommender Tasks,” *arXiv preprint arXiv:2409.09026*, Sep. 2024. ([arXiv](https://arxiv.org/abs/2409.09026?utm_source=chatgpt.com" \o "Towards Leveraging Contrastively Pretrained Neural Audio Embeddings for Recommender Tasks))

[3] P. Magron and C. Févotte, “Neural content-aware collaborative filtering for cold-start music recommendation,” *arXiv preprint arXiv:2102.12369*, Feb. 2021. ([arXiv](https://arxiv.org/abs/2102.12369?utm_source=chatgpt.com" \o "Neural content-aware collaborative filtering for cold-start music recommendation))

[4] P. Magron and C. Févotte, “Leveraging the structure of musical preference in content-aware music recommendation,” *arXiv preprint arXiv:2010.10276*, Oct. 2020. ([arXiv](https://arxiv.org/abs/2010.10276?utm_source=chatgpt.com" \o "Leveraging the structure of musical preference in content-aware music recommendation))

[5] F. Kin Hong et al., “Content‑Based Filtering Technique using Clustering Method for Music Recommender Systems,” *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 56, no. 4, 2024. ([semarakilmu.com.my](https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/article/view/5438?utm_source=chatgpt.com))

[6] J. Mayliana Alyza et al., “Music Recommendation System Based on Cosine Similarity and Supervised Genre Classification,” *JITK*, vol. 9, no. 1, 2024. ([ejournal.nusamandiri.ac.id](https://ejournal.nusamandiri.ac.id/index.php/jitk/article/view/4324?utm_source=chatgpt.com))

[7] Bogdanov et al., “Content-based music recommendation based on user preference examples,” in *Proc. ACM Workshop on Music Recommendation & Discovery*, 2009. ([SpringerLink](https://link.springer.com/article/10.1007/s10489-012-0363-y?utm_source=chatgpt.com))

[8] B. Shao, D.-D. Wang, and T. Li, “Music recommendation based on acoustic features and user access patterns,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 17, no. 8, pp. 1602–1614, Aug. 2009. ([SpringerLink](https://link.springer.com/article/10.1007/s10489-012-0363-y?utm_source=chatgpt.com))

[9] T. Yoshii, M. Goto, and T. Komatani, “Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences,” *Proc. ISMIR*, 2006. ([SpringerLink](https://link.springer.com/article/10.1007/s10489-012-0363-y?utm_source=chatgpt.com))

[10] Tzanetakis and Cook, “Music genre classification of audio signals,” *IEEE Trans. Speech Audio Process.*, vol. 10, no. 2, pp. 293–302, 2002. ([SpringerLink](https://link.springer.com/article/10.1007/s10489-012-0363-y?utm_source=chatgpt.com))

[11] C. Cohen and W. Fan, “Web-collaborative filtering: recommending music by crawling the web,” *Comput. Netw.*, vol. 33, pp. 685–698, 2000. ([SpringerLink](https://link.springer.com/article/10.1007/s10489-012-0363-y?utm_source=chatgpt.com))

[12] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009. ([SpringerLink](https://link.springer.com/article/10.1007/s11042-017-4447-x?utm_source=chatgpt.com))

[13] “Cold start (recommender systems),” *Wikipedia*, accessed Jul. 2025. ([Wikipedia](https://en.wikipedia.org/wiki/Cold_start_%28recommender_systems%29?utm_source=chatgpt.com))

[14] “Recommender system,” *Wikipedia*, accessed Jul. 2025. ([Wikipedia](https://en.wikipedia.org/wiki/Recommender_system?utm_source=chatgpt.com))

[15] “Cluster analysis,” *Wikipedia*, accessed Jul. 2025. ([Wikipedia](https://en.wikipedia.org/wiki/Cluster_analysis?utm_source=chatgpt.com))

[16] S. Sánchez‑Moreno et al., “Dynamic inference of user context through social tag embedding for music recommendation,” *arXiv preprint arXiv:2109.11231*, Sep. 2021.([arXiv](https://arxiv.org/abs/2109.11231?utm_source=chatgpt.com" \o "Dynamic inference of user context through social tag embedding for music recommendation))

[17] D. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, “Facing the cold start problem in recommender systems,” *Expert Syst. Appl.*, Mar. 2014. ([Wikipedia](https://en.wikipedia.org/wiki/Cold_start_%28recommender_systems%29?utm_source=chatgpt.com))

[18] Y. Shi, M. Larson, and A. Hanjalic, “Collaborative filtering beyond the user‑item matrix: A survey of the state of the art,” *ACM Comput. Surveys*. ([Wikipedia](https://en.wikipedia.org/wiki/Collaborative_filtering?utm_source=chatgpt.com))

[19] Context‑Aware Recommender Systems in the Music Domain: A Systematic Literature Review, *MDPI Electronics*, 2021.([MDPI](https://www.mdpi.com/2079-9292/10/13/1555?utm_source=chatgpt.com))

[20] “Comparison of content-based music recommendation using different distance estimation methods,” *Applied Intelligence*,2012.([SpringerLink](https://link.springer.com/article/10.1007/s10489-012-0363-y?utm_source=chatgpt.com))

[21] Paolo Porcaro et al., “Artwork Recommendations based on User Preferences: Integrating Clustering Analysis with Visual Features,” *J. Comput. Cultural Heritage*, 2024. ([ACM Digital Library](https://dl.acm.org/doi/abs/10.1145/3109859.3109934?preflayout=tabs&utm_source=chatgpt.com))

[22] “New Paths in Music Recommender Systems Research,” *Proc. ACM RecSys Workshop*, 2024. ([ACM Digital Library](https://dl.acm.org/doi/abs/10.1145/3109859.3109934?preflayout=tabs&utm_source=chatgpt.com))

[23] “Music Recommendation System Using Machine Learning,” *GeeksforGeeks*, updated Jun. 2025. ([GeeksforGeeks](https://www.geeksforgeeks.org/music-recommendation-system-using-machine-learning/?utm_source=chatgpt.com" \o "Music Recommendation System Using Machine Learning - GeeksforGeeks))

[24] Foote et al., “Audio retrieval by rhythmic similarity,” *Proc. ISMIR*, 2002. ([SpringerLink](https://link.springer.com/article/10.1007/s10489-012-0363-y?utm_source=chatgpt.com))

[25] Kwon et al., “Emotion recognition by speech signals,” *Eur. Conf. Speech Communication and Technology*, 2003. ([SpringerLink](https://link.springer.com/article/10.1007/s10489-012-0363-y?utm_source=chatgpt.com))

[26] H. Katarya and O. P. Verma, “Music recommendation using graph based quality model,” *Signal Process.*, vol. 120, pp. 1–8, 2016. ([SpringerLink](https://link.springer.com/article/10.1007/s11042-017-4447-x?utm_source=chatgpt.com))

[27] Foong Kin Hong et al., “Clustering for music recommendation,” 2024. ([semarakilmu.com.my](https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/article/view/5438?utm_source=chatgpt.com))

[28] Alyza et al., “Cosine similarity and genre classification,” 2024. ([ejournal.nusamandiri.ac.id](https://ejournal.nusamandiri.ac.id/index.php/jitk/article/view/4324?utm_source=chatgpt.com))

[29] Deldjoo et al., “Content‑driven music recommendation survey,” 2023. ([arXiv](https://arxiv.org/abs/2107.11803?utm_source=chatgpt.com" \o "[2107.11803] Content-driven Music Recommendation: Evolution, State of the Art, and Challenges))

[30] Méndez‑Moreno et al., “User context via social tags embedding,” 2021. ([arXiv](https://arxiv.org/abs/2109.11231?utm_source=chatgpt.com" \o "Dynamic inference of user context through social tag embedding for music recommendation))