# Emotion-Driven Song Recommendation System Using Deep Learning

Sumanth Sai Prasad - CSCI 5502
University of Colorado Boulder
Boulder, CO, United States
sumanth.saiprasad@colorado.edu

Ramyashree Mummuderlapalli Krishnaiah - CSCI 5502
University of Colorado Boulder
Boulder, CO, United States
Ramu2567@colorado.edu

*Abstract*—**Recommendation systems are designed to identify patterns in data and present users with items that best match their potential interests. This project aims to design a music recommendation model that integrates both audio properties and lyrical analysis. The audio tracks are examined through features such as tempo, beats per minute, rhythmic consistency, and genre, which are used in a content-based filtering approach. Beyond this, an additional emotional layer is introduced by applying natural language processing (NLP) to classify song lyrics into thirteen emotion categories. The outcomes of this classification are incorporated into the system pipeline, and generative adversarial networks (GANs) are then utilized to generate more personalized music recommendations.**

## I. Introduction

Recommendation systems are widely used algorithms that analyze user behavior and deliver suggestions aligned with their preferences [1]. They shape everyday choices such as online shopping, movie streaming, and music listening, often working in the background without users realizing it. Designing these systems requires selecting suitable approaches, as recommendation tasks vary across domains.

Two primary strategies are commonly employed. The first, content-based filtering, suggests items by comparing their characteristics [2]. For example, if a listener enjoys song A and song B shares similar features, the system may recommend song B. This project extends beyond such traditional item-attribute comparisons by incorporating lyric-based features into the recommendation process. Specifically, the lyrics of each track are analyzed and classified into one of thirteen emotional categories identified in a UC Berkeley study [3].

To achieve this, natural language processing (NLP) techniques [4] are applied for lyrical emotion classification. The resulting emotion labels are then merged with audio-based attributes (such as tempo, rhythm, and genre) and integrated into the dataset. These combined features are further enhanced using a generative approach, Generative Adversarial Networks (GANs), introduced by Ian Goodfellow in 2014 [5]. By leveraging GANs, the system can improve the quality of its recommendations by modeling complex patterns across variables like mood, tempo, and genre.

The remainder of this paper is organized as follows: Section II outlines the motivation for developing this system, while Section III presents an overview of the algorithms commonly applied in recommendation systems.

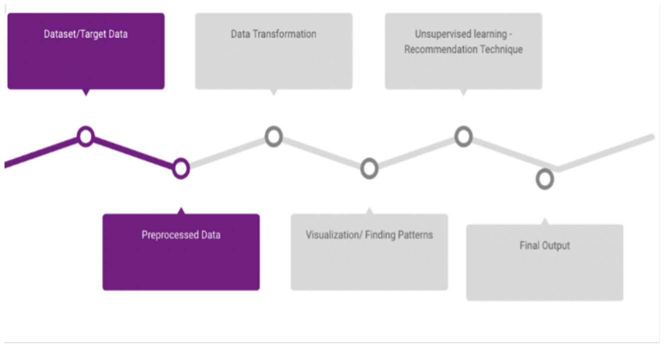
Dataset/Target Data
Data Transformation
Unsupervised learning - Recommendation Technique
Preprocessed Data
Visualization/ Finding Patterns
Final Output

Fig. 1. Proposed system pipeline for emotion-aware music recommendation.

## II. Motivation

Our initial exploration into recommendation systems began with approaches that did not incorporate emotional aspects. A key limitation we noticed was that many existing methods relied on a narrow set of variables when generating suggestions. While a few systems considered multiple attributes, such as tempo, frequency, or emotional factors, they often lacked the ability to update recommendations in real time. In most cases, the outputs were generated from previously stored data, sometimes as outdated as the previous day's records.

This limitation encouraged us to incorporate sentiment analysis into the design of our system. By classifying songs based on their emotional content in addition to audio features, we aimed to achieve a more dynamic and accurate recommendation process. Narrowing the scope from general recommendation systems to emotion-aware music recommendations allowed us to enrich the characterization of tracks and ultimately provide more personalized suggestions.

## III. Related Work

Several studies have identified similar challenges arising from the absence of real-time updates or the difficulty of handling multiple variable types in recommendation systems, and have put forward potential remedies. Fang et al. propose a method that begins with a questionnaire to establish a user profile [11]. According to their approach, "the information obtained from the questionnaire is compared with musical features to prune the initial 384,500 songs to 1000 songs the user is more likely to enjoy" [11]. The system then applies clustering—a machine learning technique that organizes data into groups of similar items—to categorize users with comparable musical preferences [11][12]. Collaborative filtering is subsequently applied, merging individual user profiles into group profiles in order to generate recommendations suitable for collective activities, such as physical therapy sessions, Zumba, yoga, or pilates [11]. Finally, Reinforcement Learning (RL) is employed to fine-tune the initial results produced by the User Profiling (UP) recommendation model [11]. The questionnaire [11] captures user ratings on features such as tempo, loudness, energy, positivity, familiarity, and rhythmic strength using a five-point scale. This design allows participants to define precise preferences, which can be essential for structured choice-based activities. While the paper integrates collaborative and content-based filtering as well as multiple input variables, its application is restricted to exercise-related music, making it unsuitable for broader everyday use cases.

Jiang et al. (2017) employed recurrent neural networks (RNNs) to evaluate song similarities, which enabled their system to assign ranking scores based on a combination of musical factors [13]. Although RNNs have been referenced in other works [14], this particular study was distinctive in that it explicitly focused on measuring similarity between songs. However, the system still lacked real-time updating capabilities.

Hayashi et al. introduced a fast content-based music information retrieval (CBMIR) framework termed indirect matching [15]. Their method utilized representative queries from offline searches, which were then leveraged as the foundation for generating efficient similarity approximations in online scenarios. Like [13] and [16], this approach incorporated aspects of content-based filtering but excluded collaborative filtering and real-time adaptation.

In the same year, Sunny et al. developed a framework for handling real-time data streams to produce instant and accurate recommendations [18]. Their implementation applied Spark and machine learning libraries to the domain of TV channel recommendations. Similarly, Mwinyi et al. proposed a predictive, self-learning recommendation system that adjusts to user preferences both before and after content choices are made [17]. While both approaches introduced real-time updates and accounted for multiple variable types, they did not make use of collaborative filtering methods.

## IV. Proposed Work

### Generative Adversarial Networks (GANs)

A GAN consists of two competing neural networks: a generator and a discriminator. The generator's role is to produce synthetic samples that resemble the training data, while the discriminator attempts to distinguish between real and generated inputs. Through this adversarial process, the generator progressively improves, ultimately learning to create samples with statistical properties that closely mirror the original dataset. Once trained, the GAN serves as a powerful generative model capable of producing novel examples that maintain the characteristics of the input data.

### Content-Based Filtering

When making suggestions, content-based filtering compares user profiles to the keywords and characteristics that have been assigned to objects in a database. For example, if a customer recently purchased a smartphone and has a history of buying accessories, the system could recommend compatible phone holders or cases. Features such as brand, model, and material (e.g., RFID-blocking fabric) guide the recommendation, ensuring that the suggested items align with the user's established interests.

### Collaborative Filtering

Collaborative filtering generates recommendations by analyzing the relationships between users and items simultaneously. Unlike content-based methods, which rely heavily on item attributes, collaborative filtering leverages the preferences of similar users. This enables the system to provide serendipitous suggestions. For example, the system might recommend an item to one user because another user with similar tastes enjoyed it. Modern implementations further enhance this process by using embeddings, which allow the system to automatically learn feature representations rather than depending solely on manual feature engineering.

## V. Data Collection, Modeling, and Cleaning

### Data Collection

To construct a reliable and diverse dataset, the Spotify Web API was selected as the primary data source. The API provides access to detailed song-level metadata such as artist name, track ID, tempo, key, loudness, energy, and valence. These features were retrieved using the Python package *Spotipy*, which offers convenient wrapper functions for the official Spotify endpoints. Because the Spotify API enforces rate limits, the data retrieval process was automated through batched queries with scheduled delays to prevent throttling. The final dataset integrates both the audio features obtained from Spotify and the lyrical data scraped from publicly available lyric databases, ensuring alignment between song metadata and text content for downstream emotion analysis.

### Data Modeling

The collected data was organized into three interconnected entities: *Tracks*, *Artists*, and *Audio Features*. Each entity was stored in a PostgreSQL relational schema to maintain referential integrity. The use of structured storage allowed flexible joins between track attributes and emotion labels generated through natural language processing (NLP). This design simplified the integration of emotion-based features into the overall recommendation pipeline. PostgreSQL was selected for its strong support for analytical queries and smooth compatibility with visualization tools such as Tableau, which were used for exploratory data inspection.

### Data Cleaning

The raw data returned by Spotify included inconsistent formats and occasional missing entries. Numerical attributes (for instance, energy and tempo) were converted to the appropriate data types, while categorical fields such as genre were standardized to lowercase and stripped of special characters. Missing values were handled using imputation strategies tailored to the attribute type-mean imputation for continuous variables and mode substitution for categorical features. For the lyric dataset, text preprocessing steps included tokenization, lowercasing, removal of punctuation and stop words, and lemmatization. These operations ensured that the NLP classifier for emotion detection received clean, normalized input, improving both classification accuracy and feature quality for the recommendation model.

## VI. Analysis of Audio Features

Building a music recommendation system requires identifying which song characteristics most accurately represent listener preferences. Because such systems often predict on unseen data, it is crucial to verify that the underlying features capture meaningful musical patterns. To validate our feature selection, exploratory data analysis (EDA) was conducted using both univariate and multivariate approaches. Visualizations were produced through Seaborn, Matplotlib, and Tableau to study how numerical attributes correlate with each other and to ensure that the data distribution aligns with real-world trends in music preferences.

### Feature Relationships

To understand the relationships between song-level properties, we computed the correlation matrix for all numerical features. Figure 2 presents the resulting heatmap, which highlights strong correlations among *danceability*, *tempo*, and *loudness*. As these three features increase together, they often characterize energetic and rhythmically engaging tracks. Gaussian kernel density estimation revealed that most audio features are non-normally distributed, exhibiting either left or right skew. Consequently, direct visualization of raw values may obscure trends, so stratified sampling was applied to ensure balanced representation across feature intervals.

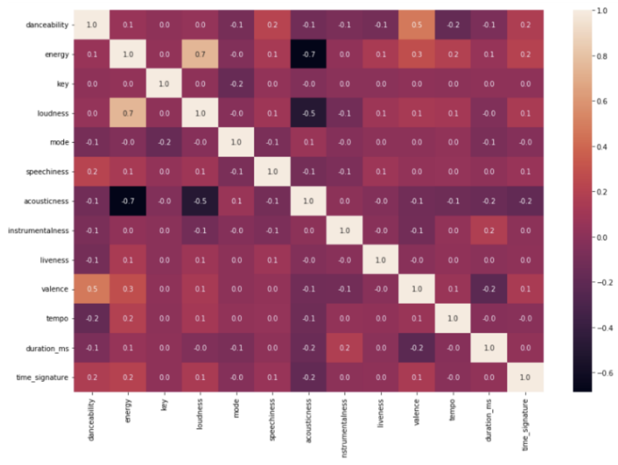

Fig. 2. Correlation heatmap of audio features.

We further examined several pairwise relationships among features. Figures 3–5 demonstrate how *tempo*, *loudness*, and *energy* evolve with respect to *danceability* and *instrumentalness*. A clear positive trend between danceability and tempo/loudness suggests that upbeat and louder tracks tend to be more danceable. Meanwhile, acousticness exhibits an inverse relation with energy, showing that more acoustic tracks are typically softer and calmer in tone.

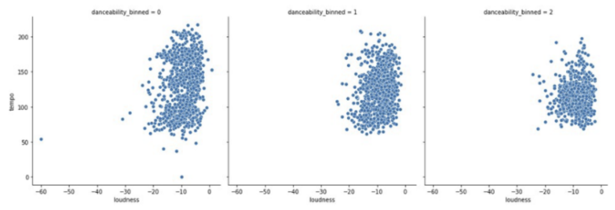

Fig. 3. Tempo vs. Loudness with respect to Danceability.

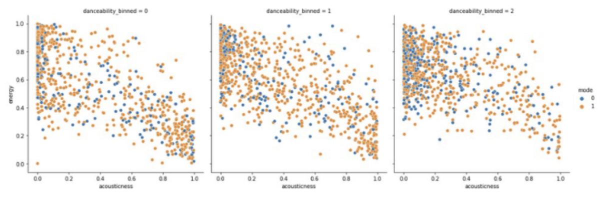

Fig. 4. Energy vs. Acousticness with respect to Danceability.

, ,

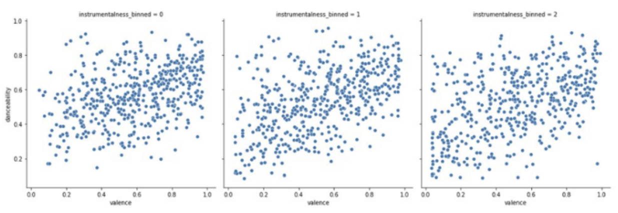

Fig. 5. Danceability vs. Valence with respect to Instrumentalness.

### Clustering

To uncover latent groupings within the feature space, unsupervised learning techniques were applied. Clustering provides insight into natural divisions in the dataset without relying on labeled data. Algorithms such as K-Means, K-Medoids, and DBSCAN (Density-Based Spatial Clustering of Applications with Noise) were implemented to test how audio attributes cluster under different distance measures.

To facilitate visualization, dimensionality reduction using Principal Component Analysis (PCA) was performed to project high-dimensional audio features into two principal components. The resulting plot (Figure 6) illustrates the output of the K-Medoids algorithm on the transformed data. Although clear clusters were not strongly separable, the experiment confirmed that most tracks share overlapping audio traits, which supports the need for emotion-based augmentation in the recommendation model.

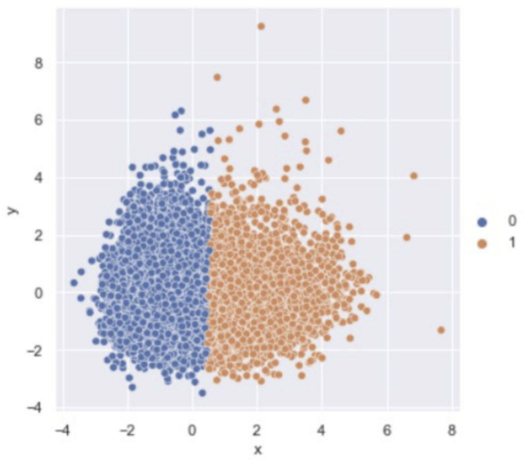

Fig. 6. K-Medoids clustering on PCA-transformed feature space.

## VII. Content-Based Filtering

Content-based filtering recommends items by analyzing the characteristics of objects and comparing them with a user's established preferences. In a music recommendation context, this method evaluates song-level metadata-such as duration, tempo, artist, and genre-along with user interaction history, including frequently played tracks and liked songs. These patterns are encoded into a user profile that serves as the foundation for generating personalized suggestions.

Our system integrates all available song attributes, including acoustic properties and lyric-based emotion categories, to identify songs that best align with the listener's profile. For each new input track, the model extracts key features, applies vectorization techniques such as TF–IDF, and calculates similarity scores to retrieve the most relevant recommendations.

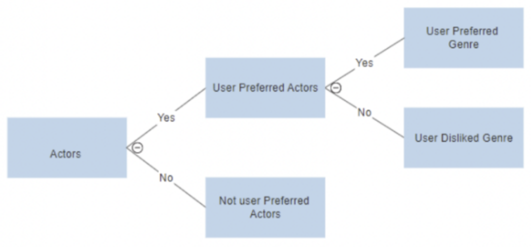

Fig. 7. Workflow of the content-based recommendation process.

### Cosine Similarity

Cosine similarity measures how closely two items resemble each other by comparing the angle between their feature vectors. A smaller angle corresponds to greater similarity. In this project, cosine similarity was applied to assess how closely the feature representations of a user's preferred tracks align with those in the larger corpus. The resulting similarity scores guide which songs are recommended next.

Mathematically, cosine similarity between vectors $A$ and $B$ is defined as:

$$S_C(A,B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

This metric enables the system to identify the songs whose combined lyrical and audio features share the most overlap with the user's historical preferences.

### Sentiment Analysis using TextBlob

Natural Language Processing (NLP) provides an avenue for analyzing song lyrics to extract emotional and contextual

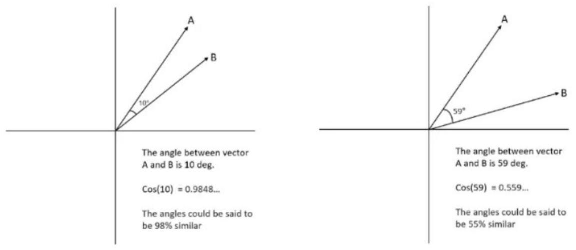
A
B
10°
The angle between vector A and B is 10 deg.
Cos(10) = 0.9848...
The angles could be said to be 98% similar
A
B
59°
The angle between vector A and B is 59 deg.
Cos(59) = 0.559...
The angles could be said to be 55% similar

Fig. 8. Geometric intuition of cosine similarity.

meaning. In this project, the TextBlob library was used to evaluate the sentiment polarity and subjectivity of each track title or lyric segment. Polarity ranges from –1 (negative) to +1 (positive), while subjectivity measures the extent to which a statement expresses opinion rather than fact.

For instance, a song like "Good Vibrations" yielded a strongly positive polarity and high subjectivity, indicating a cheerful tone, whereas "Smells Like Teen Spirit" produced near-neutral polarity with low subjectivity, reflecting a more ambiguous emotional stance. These sentiment scores supplement the audio features, enhancing emotion-driven personalization within the recommendation model.

### Term Frequency–Inverse Document Frequency (TF-IDF)

To further refine the lyrical and textual feature representation, the TF–IDF (Term Frequency–Inverse Document Frequency) technique was applied. This method assigns higher weights to words that are distinctive within a document but rare across the entire corpus, making it especially useful for song lyric analysis. By using TF–IDF, the system emphasizes terms that uniquely characterize a song's emotional or thematic content while down-weighting frequently used but uninformative words such as "the" or "and."

Mathematically, the TF-IDF value for a term $i$ in document $j$ is given as:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \tag{2}$$

where $tf_{i,j}$ denotes the number of occurrences of word $i$ in document $j$, $df_i$ represents the number of documents containing the word $i$, and $N$ is the total number of documents in the dataset.

The *term frequency* captures how frequently a word appears in a song's lyrics, while the *inverse document frequency* measures how unique that word is across all songs in the dataset. Words that occur often in only a few songs, such as "heartbeat" or "rainfall," are considered more characteristic and therefore receive higher weights.

By constructing a TF–IDF vector for each track, we created a rich text-based feature representation that complements the numerical audio attributes. These vectors were integrated with genre tags, emotional labels, and sentiment polarity to form the final feature matrix used for generating recommendations. This hybrid representation helped the model discern subtle lyrical patterns-such as tone, mood, and emotional emphasis-leading to more context-aware and personalized music suggestions.

## VIII. Collaborative Filtering

Collaborative Filtering (CF) is a recommendation strategy that derives suggestions by learning from the collective behavior of users. Instead of depending solely on song attributes or lyrics, CF assumes that users who share similar preferences will likely enjoy the same set of tracks. For example, if User A and User B both enjoy a group of songs, and User A later listens to a new track, that track becomes a strong candidate for recommendation to User B.

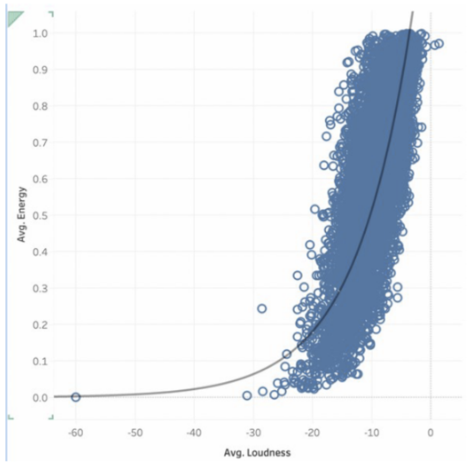

Fig. 9. Collaborative filtering framework showing user–item relationships.

### Principle

The main intuition behind collaborative filtering is that similarity between users or items can be exploited to predict missing preferences. If a group of users consistently agrees on the relevance or appeal of certain tracks, the system infers that they will likely agree on others as well. Hence, user feedback-such as play counts, likes, or skips-is represented in a matrix format where rows correspond to users and columns correspond to tracks. The missing entries in this matrix are then predicted using CF algorithms.

### Similarity Computation

To compute similarity between users or items, several mathematical functions can be applied. The most common measures include:

- Pearson Correlation Coefficient
- Constrained Pearson Correlation
- Spearman Rank Correlation

These metrics evaluate the degree of agreement between user ratings or listening patterns. In this project, cosine similarity was also tested for cases with sparse user data, as it normalizes vectors and remains stable even when the number of common ratings is small.

### Model Representation

The user–item interaction data can be modeled as a sparse matrix $R$ of dimension $m \times n$, where $m$ is the number of users and $n$ is the number of songs. The goal is to estimate the unknown preference values $\hat{r}_{u,i}$ for each user $u$ and song $i$. To capture hidden features underlying these interactions, matrix factorization techniques such as Singular Value Decomposition (SVD) can be used, decomposing $R$ into two lower-dimensional matrices:

$$R \approx P \times Q^T \tag{3}$$

where $P$ encodes latent user preferences and $Q$ encodes latent song characteristics. The reconstructed matrix approximates how likely each user is to enjoy a specific song.

### Hybrid Integration

Although CF performs well in discovering new and relevant tracks, it faces challenges such as the cold-start problem when new users or songs have little to no history. To overcome this limitation, our system combines collaborative filtering with content-based and emotion-aware components. The CF module identifies user-behavior patterns, while the content-based and lyrical emotion layers refine recommendations based on mood and song attributes. This hybrid design ensures both accuracy and diversity in the final recommendation output.

## IX. Milestones

### Milestone 1: Project Planning and Setup

- **Tasks:**
  - Define the overall project scope and objectives.
  - Set up the development environment and dependencies.
  - Gather and preprocess the dataset for analysis.
- **Timeline:** Weeks 1–2
- **Deliverables:** Project plan, initialized environment setup, and preprocessed dataset.

### Milestone 2: Feature Engineering and Integration

- **Tasks:**
  - Analyze the dataset to understand its structure and key characteristics.
  - Perform feature engineering on the dataset to refine predictive variables.
  - Integrate emotion classification using NLP-based lyric analysis.
- **Timeline:** Weeks 3–4
- **Deliverables:** Detailed data analysis report, engineered features, and emotion classification module.

### Milestone 3: Model Development and Training

- **Tasks:**
  - Develop and implement the GAN-based model for emotion-aware music recommendation.
  - Train the model using the preprocessed and feature-engineered dataset.
- **Timeline:** Weeks 5–7
- **Deliverables:** Trained GAN model, training logs, and performance metrics.

### Milestone 4: Model Evaluation and Optimization

- **Tasks:**
  - Evaluate the model using the defined performance metrics.
  - Optimize the model based on quantitative and qualitative evaluation results.
- **Timeline:** Weeks 8–9
- **Deliverables:** Evaluation report, optimized model, and updated performance metrics.

## References

[1] P. Jomsri, S. Sanguansintukul, and W. Choochaiwattana, “A framework for tag-based research paper recommender system: An IR approach,” in *Proc. IEEE Int. Conf. Advanced Information Networking and Applications Workshops*, pp. 103–108, 2010.

[2] M. Hassan and M. Hamada, “Improving prediction accuracy of multicriteria recommender systems using adaptive genetic algorithms,” in *Proc. Intelligent Systems Conference (IntelliSys)*, pp. 326–330, 2017.

[3] A. S. Cowen, X. Fang, D. Sauter, and D. Keltner, “What music makes us feel: At least 13 dimensions organize subjective experiences associated with music across different cultures,” in *Proc. Int. Conf. Infocom Technologies and Unmanned Systems*, 2017.

[4] Y. Agrawal, R. G. R. Shanker, and V. Alluri, “Transformer based approach towards music emotion recognition from lyrics,” 2020.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Networks,” in *Advances in Neural Information Processing Systems (NIPS)*, pp. 2672–2680, 2014.

[6] “Music Recommendation System Using Natural Language Processing and Audio Feature Analysis,” 2021.

[7] “Music Recommender System Based on Sentiment Analysis Enhanced with Natural Language Processing Techniques,” 2021.

[8] “Lyric-based Song Recommendation with Doc2Vec Embeddings and Spotify’s API,” 2021.

[9] "MusicMood: Predicting the Mood of Music from Song Lyrics Using Machine Learning," 2021.

[10] J. Zhan and B. Dahal, "Using deep learning for short text understanding," *Journal of Big Data*, vol. 4, no. 34, pp. 1–15, 2017.

[11] J. Fang, D. Grunberg, S. Luit, and Y. Wang, "Development of a music recommendation system for motivating exercise," in *2017 International Conference on Orange Technologies (ICOT)*, Dec 2017, pp. 83–86.

[12] M. Ahmed, M. T. Imtiaz, and R. Khan, "Movie recommendation system using clustering and pattern recognition network," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2018, pp. 143–147.

[13] M. Jiang, Z. Yang, and C. Zhao, "What to play next? a RNN-based music recommendation system," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct 2017, pp. 356–358.

[14] M. Kataoka, M. Kinouchi, and M. Hagiwara, "Music information retrieval system using complex-valued recurrent neural networks," in *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, vol. 5, Oct 1998, pp. 4290–4295 vol.5.

[15] T. Hayashi, N. Ishii, M. Ishimori, and K. Abe, "Stability improvement of indirect matching for music information retrieval," in *2015 IEEE International Symposium on Multimedia (ISM)*, Dec 2015, pp. 229–232.

[16] X. Wang, Y. Bai, and Y. Li, "An information retrieval method based on sequential access patterns," in *2010 Asia-Pacific Conference on Wearable Computing Systems*, Apr 2010, pp. 247–250.

[17] I. H. Mwinyi, H. S. Narman, K. C. Fang, and W. S. Yoo, "Predictive self-learning content recommendation system for multimedia contents," in *2018 Wireless Telecommunications Symposium (WTS)*, Apr 2018, pp. 1–6.

[18] B. K. Sunny, P. S. Janardhanan, A. B. Francis, and R. Murali, "Implementation of a self-adaptive real time recommendation system using Spark machine learning libraries," in *2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, Aug 2017, pp. 1–7.

[19] Spotify Developers, "Spotify Web API Reference," [Online]. Available:https://developer.spotify.com/documentation/web-api/. Accessed: Oct. 2025.

[20] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1–38, 2020.

[21] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[22] PostgreSQL Global Development Group, "PostgreSQL Documentation," [Online]. Available:https://www.postgresql.org/docs/. Accessed: Oct. 2025.

[23] A. Grinberg, "Flask Web Development: Developing Web Applications with Python," 2nd ed. O'Reilly Media, 2023.