Machine Learning Content-Based Music Recommendation System

1st Isaac Olvera
Computer Science, CSUF
California State University, Fullerton
Fullerton, CA
isaac_olvera712@csu.fullerton.edu

2nd Christina Reyes
Computer Science, CSUF
California State University, Fullerton
Fullerton, CA
r.christina24@csu.fullerton.edu

3rd Wilson Tu
Computer Science, CSUF
California State University, Fullerton
Fullerton, CA
wtu4979@csu.fullerton.edu

***Abstract*—The objective of this paper is to analyze and create efficient techniques for a music recommendation system. This can be done through a wide variety of factors such as danceability, tempo, acousticness, instrumentalness, energy, and valence of an individual song. Using machine learning techniques such as decision trees and k-nearest neighbor, we can take these factors and use them to train, test, and predict how similar one song is to another, which in turn creates an effective system that will increase the enjoyability and overall happiness of a user.**

***Index Terms*—Machine learning, recommendation system, preprocessing, label encoding, correlation matrix**

I. Introduction

In order to sustain a sizable and steady consumer base, many e-commerce websites rely on recommendation systems. Websites like Amazon and retail businesses use client purchase data to predict buying trends and, as a result, recommend products to entice repeat business. Streaming services like Netflix and Spotify operate similarly in that they use various elements of a user's prior usage history to suggest films or music that will keep the user coming back. In this paper, we will focus on the use case of music streaming services, whose primary goals are not just to provide a platform for music but to also introduce users to new songs and musical artists.

In recent years, we as consumers have watched and experienced the rapid advancement of the technology that services use for their systems to determine the most suitable recommendations. Prior to this technology, the best alternative to finding new music was through social recommendations, where one could learn about music based on the suggestion of another person. Although this social aspect was something people could bond over, it was severely restricted in terms of the quantity, quality, and speed of things that one person could find that they would actually enjoy [1]. Consumers can now, however, rely on programs that direct us to new music, which is advantageous to both the client and the musical artist in the form of revenue.

Just as humans are, technology can be flawed and requires trial and error to learn the preferences of a user. Through the use of machine learning, developers can create machines that utilize a variety of factors that they will learn from in order to deliver instant results for the user. This is where streaming services differentiate themselves from retail services. Machine learning allows users to find a song or artist in an instant and can make use of these results repeatedly, continuously, and with a user's own intentions [2][3]. To accomplish this, developers must carefully select the appropriate methods and dataset.

Spotify serves as an example of a few distinct approaches that can result in various playlist styles. Spotify can generate playlists for a user based on certain criteria, such as moods or artists the user is following. However, a study found that the most effective and frequently utilized technique of generation is the "Radio" method, which makes use of particular attributes of a selected song or artist [8]. This method can be referred to as **content-based** recommendation as it uses the qualities of a song or artists and can make predictions without the use of additional outside information [6]. This method is similar to how we approached our recommendation system.

A recommendation system can be built using machine learning in a variety of ways, including unsupervised, supervised, and even deep learning [5]. Support Vector Machines, Decision Trees, Random Forest, and Naive Bayes were the supervised learning methods we employed for this project. These methods allowed us to compare the musical qualities of songs and predict their genres and similarities, which in turn allowed us to produce a list of suggested songs.

II. Background

A. Types of Recommendation Systems

As previously established, our project shares similarities with a content-based filtering system, indicating the existence of alternative filtering methodologies. Among various styles, content-based and collaborative filtering systems are most commonly adopted.

To elaborate on content-based filtering, this technique initially extracts the attributes of each data point — in our context, each song — based on the user's past interactions. Each song's features, such as "acousticness," "danceability," "energy," and "genre," play crucial roles in this extraction. These features are reflective of those found in the Free Music Archive (FMA), an open and easily accessible dataset, which is an excellent tool for tasks in Music Information Retrieval (MIR) - a field concerned with the browsing, searching, and organizing of extensive music collections.

The FMA dataset offers a comprehensive resource, including 917 GiB and 343 days of Creative Commons-licensed audio from 106,574 tracks by 16,341 artists across 14,854 albums. These tracks are categorized in a hierarchical taxonomy of 161 genres, providing full-length, high-quality audio, pre-computed features, and various levels of metadata, tags, and free-form text such as biographies. The dataset, therefore, is quite effective in overcoming the community's growing demand for large audio datasets for feature and end-to-end learning [12].

After extracting the features from a user's historical data, the system can predict other songs the user might enjoy based on similar descriptors. Importantly, content-based recommendations do not necessarily rely on the data of multiple users or a rating history. Instead, they can draw from the data of a single user [6][7]. However, this method heavily relies on the existing user's data. Therefore, if a user's history is insufficient, the accuracy of the recommendations may be less precise.

In areas where content-based filtering faces limitations, collaborative filtering shines. Collaborative recommendation systems harness the power of community data, including the listening history and ratings of other users with similar tastes, to generate personalized song recommendations.

Unlike content-based systems that focus on individual song characteristics, collaborative filters consider the broader music ecosystem. They factor in the listening history of not just the individual user, but also the musical preferences of other active users within the community. The dataset for this approach typically comprises the current user's ratings along with data points from these like-minded users.

This broad-based approach allows the system to generate recommendations even when the current user's data is sparse. It looks at what similar users listen to and uses this information to recommend songs. While this method can effectively fill gaps in data, it's important to note that the resulting recommendations might not always be as specific as desired and could lean towards the genres that the current user frequently listens to [4]. Therefore, while collaborative filtering offers a robust approach to music recommendation, it should also be applied thoughtfully to maintain the relevance and diversity of the recommendations.

III. Dataset And Data Preprocesssing

The FMA dataset used for this project contained a total of 4 csv files (echonest, features, tracks, and genres) that had well over 100+ features for analysis. The approach that we wanted to take for this project was to analyze and train the data based on features such as danceability, tempo, and other similar factors. This meant however that a majority of the features within each file, especially the features file, would not be used as they required other techniques that we decided not to cover such as audio analysis.

Once we were able to extract all of the features required for our machine-learning techniques, we began to preprocess the data. First, we needed to merge two csv files together, tracks and echonest, since they both contained important features as well as our target feature: genre. From there, we reordered our new dataset based on the track id of each song, and we removed each song that contained multiple missing values in features such as acousticness and valence. Upon further examination, some of the features within our dataset were still either missing values or were categorical instead of nominal. To combat this, we decided to do the three following things:

1) Manually convert some of the features, such as duration, from string to float using the astype() function provided by Python.
2) Fill in each missing value with either the median value of the column or by placing a string ("unknown" in this case) in each empty spot.
3) Use the LabelEncoder() function provided by scikit-learn to change each remaining string into its nominal version based on the number of distinct labels.

From there, once we finished preprocessing the data, this then allowed us to examine all of the features within our dataset using histograms, box plots, and a correlation matrix to find which features were the most similar/dissimilar from each other, and also to find if certain features contained noise or outliers. Features such as valence and danceability had a high positive correlation with one another, which meant that it would be essential to keep and use them for our training and testing. Other features, such as speechiness and instrumentalness, had a high negative correlation with one another. After fully examining each feature, we were then finally able to start training and testing our data

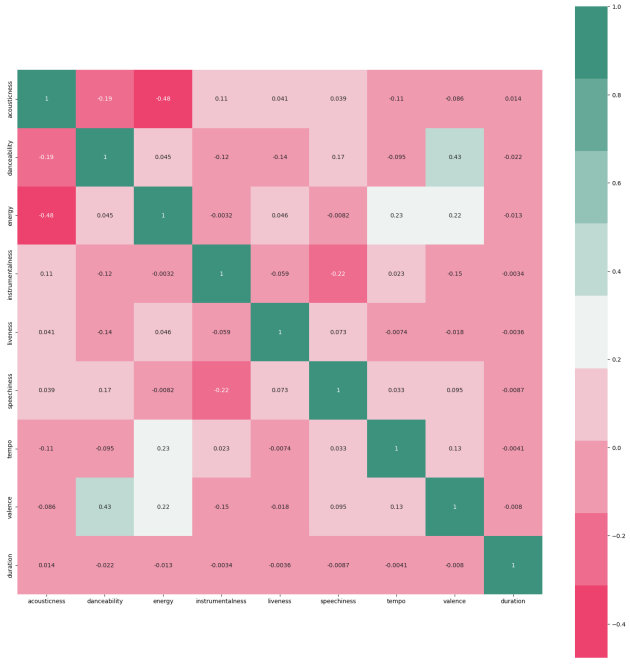

Fig. 1. Correlation matrix of all the features in our dataset.

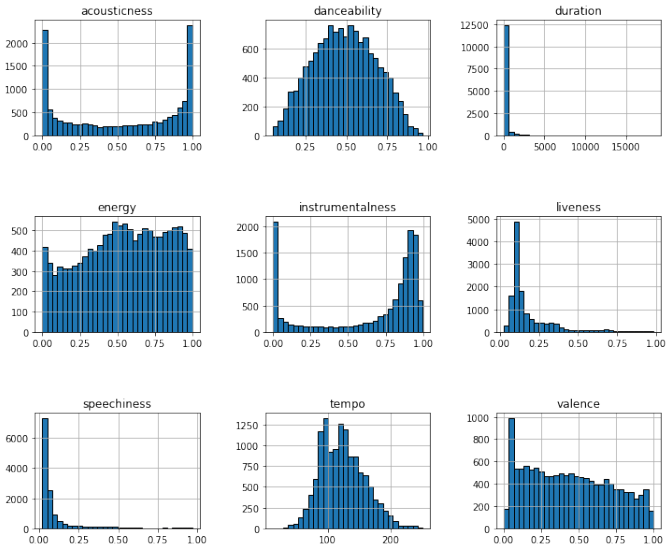

Fig. 2. Histogram distribution of all the features in our dataset.

IV. Machine Learning Techniques

A. Support Vector Machine

Support Vector Machines (SVM) create hyperplanes in an n-dimensional space (n being the number of features) that distinctly classifies data points within a dataset. A hyperplane is chosen based on the best maximum distance between data points of each class. Each hyperplane created uses support vectors, which are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane itself. SVM also heavily relies on the type of kernel (mathematical functions) it uses because this is how the machine learning technique transforms our data into its proper form.

For this project, since this was the first machine-learning technique we decided to use on our data, we created two separate SVM algorithms. One of the SVM techniques contained a full feature set, and the second technique contained a reduced feature set (removing features such as artist name and song name) so that we would be able to determine which features we should keep and train later on using our other machine learning techniques. After testing and training, we found that although both SVM techniques did not score as high as we had hoped, they both reached a similar accuracy score with the first SVM technique scoring a 45% and the second SVM technique scoring a 40.1%. Since both of the techniques had similar accuracy, precision, recall, and f-score results, then it would not matter which feature set we used to test on for our other algorithms, including the KNN which is where we create our music recommendation system.

B. Decision Tree

Decision Trees are a type of supervised machine learning technique used to categorize or make predictions based on how a previous set of questions were answered. They are a common way of visualizing decision making processes and are used in many other applications besides machine learning. Decision trees are made up of root, decision, and leaf nodes, where the root node is the base of the tree, decision nodes are the decisions to be made, and the leaf node represents possible outcomes. In order to achieve the best possible outcomes, decision trees utilize splitting and pruning, which is the processes of creating and removing nodes based on their importance.

For this project, we used the DecisionTreeClassifier provided by scikit-learn to train and test our data using the reduced feature set that was created during our training and testing for our previous SVM algorithms. Tuning its parameters and setting its random_state to 42, we were able to achieve an accuracy of 75.9% making it the most effective model in our project. A closer examination of its results also reveals that its precision, recall, and f1-score results all contained the respective value of 76%, providing further confidence in making it our top model.

C. Random Forest Classifier

A Random Forest Classifier is a versatile and powerful ensemble learning technique that combines multiple decision trees to improve the overall prediction performance of a classification task. It operates by constructing a multitude of decision trees at training time and outputs the class that is the mode of the classes predicted by individual trees. This model is known for its robustness, interpretability, and ability to handle large datasets with high dimensionality and missing values. The core principle of the Random Forest Classifier relies on the wisdom of the crowd, where the collective decision of multiple trees results in a more accurate and reliable prediction than any single tree in isolation.

For this project, we utilized the RandomForestClassifier class using the Scikit-learn library and tuned its hyperparameters, such as the number of trees in the forest (n_estimators), the maximum depth of the trees (max_depth), and the minimum number of samples required to split an internal node (min_samples_split), to optimize the model's performance.

The model underwent training on the training set, employing a 5-fold cross-validation method to minimize overfitting while evaluating its generalization capabilities. Our Random Forest Classifier achieved an accuracy score of 72.7% on the test dataset. The model's precision, recall, and F1-score were also satisfactory, with values of 75.8%, 72.7%, and 71.2%, respectively. Consequently, it emerged as our second most effective model in the study.

D. Naive Bayes

Naive Bayes is a type of supervised machine learning technique that seeks to model the distribution of inputs of a given class or category. The technique relies on the Bayes' Theorem, which is a probability classifier, in order to create predictions based on events that have occurred. Any additional information that is later added on to the algorithm will affect the initial probability that was already given to an outcome. Predictors within a Naive Bayes model are presumed to be conditionally independent and contribute equally to the outcome as well.

For this project, we used the GaussianNB classifier provided by scikit-learn to train and test our data. Tuning its parameters, we were able to achieve an accuracy of around 21.8% making it the least effective model in our project. A closer examination of its results also reveals that its precision, recall, and f1-score results contained the respective values of 21.8%, 29%, 22%, and 20%, providing further confidence in making it our worst model.

The main reasons as to why this model scored the lowest is due to the nature of the Naive Bayes algorithm. Not all of the features within out dataset are equal, and as stated early in the paper certain features have high or negative correlation with one another, which affects the decision making of the model. Naive Bayes also works best when all categorical features exist in the training set, and unfortunately this is not the case for our dataset which makes the overall probability of those events equal to zero.

E. K-Nearest Neighbor

The K-Nearest Neighbors (KNN) algorithm has been employed in this music recommendation system to provide personalized suggestions to users based on their input song. The KNN algorithm leverages the power of similarity metrics by comparing the features of a song, such as genre, tempo, and energy, to the features of other songs in the dataset in order to find the most similar songs.

To begin with, the KNN algorithm sets the number of recommended songs to be returned for each input song to 3. It then initializes two empty lists to store the recommended songs and the track IDs of the first five input songs. Additionally, a StandardScaler object is initialized to scale the features of the dataset, and a LabelEncoder object is used to encode the song names.

The algorithm proceeds by iterating through the first five rows of the dataset with the genre column included, extracting the predicted genre value for each row. The dataset is then filtered to retrieve all songs with the same genre as the input song. Subsequently, the NearestNeighbors algorithm with one nearest neighbor and the Euclidean distance metric is utilized to find the track ID of the song in the dataset that is most similar to the current song. This track ID is then stored in the list of track IDs.

Before proceeding, the same-genre songs dataset is filtered to exclude any song names that are not in the encoder's classes. The same-genre songs dataset and the current song's features are then scaled using the StandardScaler object. With the data prepared, the NearestNeighbors algorithm with k nearest neighbors and the Euclidean distance metric is used to find the most similar songs to the current song.

The Euclidean distance metric employed by the KNN algorithm can be mathematically expressed as:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

where p and q are two points in an n-dimensional space, and $d(p, q)$ is the Euclidean distance between them.

Once the most similar songs are identified, the recommended song details are extracted and stored in a variable. Finally, a list of recommended song names is created by inverse transforming the encoded song name column of the recommended songs DataFrame. These song names are then stored in a list of recommended songs for each input song, thereby providing users with a set of personalized recommendations based on their preferences.

V. Results and Analysis

In our project, we utilized a variety of machine learning models to analyze and predict our dataset. Of these, the Decision Tree model emerged as the most successful in terms of its performance. The metrics we used to evaluate the models included accuracy, precision, recall, and the f1-score. Each of these metrics offers a unique perspective on the performance of a model and enables us to understand the model's strengths and weaknesses in different areas.

The Decision Tree model showed an accuracy of 75.9%. Accuracy is a fundamental metric that reflects the proportion of total predictions that the model got correct, including both true positives and true negatives. A high accuracy means that the model was able to correctly classify a large proportion of the instances in our dataset.

In terms of precision, the Decision Tree model achieved a score of 76%. Precision is a measure of how many of the instances that the model predicted as positive were actually positive. A high precision score indicates that when the model predicts a positive outcome, it is usually correct.

The model's recall was also 76%. Recall measures the proportion of actual positive instances that the model was able to identify correctly. A high recall score means that the model was able to correctly identify a large proportion of the positive instances in the dataset.

The f1-score for the Decision Tree model was also favorable. The f1-score is the harmonic mean of precision and recall, providing a balanced measure of both metrics. It is particularly useful when the data has an uneven class distribution. An f1-score of 76% signifies that the model maintained a good balance between precision and recall.

On the other hand, the Naive Bayes model displayed the least impressive results. It achieved an accuracy of just 21.8%, suggesting it struggled to correctly classify instances in our dataset. Its precision was marginally better at 29%, but this still indicates a high rate of false positives. The recall was also low at 22%, meaning it failed to identify a large proportion of actual positive instances. Finally, the f1-score was the lowest at 20%, suggesting that the model struggled to find a balance between precision and recall.

In summary, our results show a stark contrast between the performance of the Decision Tree and Naive Bayes models. While the Decision Tree model performed admirably across all metrics, the Naive Bayes model exhibited significant weaknesses in its predictive capabilities.

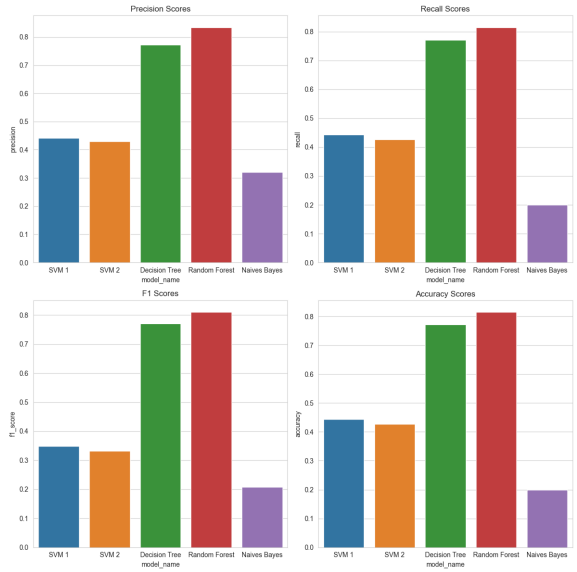

Fig. 3. Performance metric bar of all the machine learning techniques used.

In the realm of our K-Nearest Neighbors (KNN) based song recommendation algorithm, conventional accuracy or performance metrics cannot be straightforwardly applied. The primary goal of this algorithm is to generate personalized song recommendations, a task that does not lend itself to traditional model performance evaluation methods.

Nevertheless, the algorithm is adept at delivering pertinent song recommendations according to a given set of input songs and their features. This process is accomplished by identifying the songs in the dataset that are most similar to the input songs based on their various features such as danceability, tempo, liveness, etc. By adjusting the 'k' parameter, users can control the number of songs recommended per input song. This flexibility allows users to find an optimal balance between the diversity of suggestions and the relevance of the recommendations, ensuring that the output is both varied and in line with their musical tastes.

```
3 recommended songs for Ring The Bell (Instrumental) by Steve Gunn:
- Chiller by André Marchal
- I can't believe in Lo-Brow by Samita Sinha
- Silence, Amen by Horiso


3 recommended songs for Liverpool Sluts by Transmitters:
- I can't believe in Lo-Brow by Samita Sinha
- Dearth by Kraus
- Chatting With Richard by H. Benne Hendon


3 recommended songs for Qualms of Conscience by David Rovics:
- Dead City by Live Ones
- lunoion = a nu lion by Carsie Blanton
- gigngante = anti egg by Carsie Blanton
```

Fig. 4. Example of song recommendations generated by the KNN-based recommendation algorithm when k-parameter = 3.

Figure 4 offers a visual representation of the song recommendations made by the KNN-based recommendation algorithm. This graphic clearly demonstrates the algorithm's capability to generate a range of song suggestions based on the input songs.

In conclusion, while our Decision Tree model stood out as the highest performer in terms of conventional performance metrics, the KNN-based song recommendation algorithm presents a valuable and adaptable method for generating personalized song recommendations. It provides a solution that directly caters to the unique musical preferences of individual users.

However, the effectiveness of the KNN-based recommendation algorithm is not without room for further exploration and refinement. Future research could delve into understanding the impact of varying the 'k' parameter on the quality and relevance of the recommendations generated. This could shed light on how to fine-tune the 'k' parameter to optimize the performance of the algorithm, thereby enhancing its utility in delivering personalized and satisfying song recommendations.

VI. Future Work

Building upon the concept of content-based recommendation, our project extracted features from each song to make personalized suggestions. While this methodology is robust, it does exhibit certain limitations which can be addressed by employing collaborative filtering. However, to truly maximize the strengths of both content-based and collaborative filtering and to minimize their individual shortcomings, we propose the implementation of a hybrid recommendation system.

Hybrid filtering marries the best features of both systems, drawing on the current user's data and the collective data of the wider user base [9][10]. This combination allows for a more comprehensive understanding of user preferences, thereby providing incredibly precise recommendations. Thus, the hybrid recommendation system represents a promising future direction for our project, offering a more effective solution that caters to individual tastes while also considering the broader music community.

Beyond the algorithmic enhancements, user experience is a critical aspect that deserves equal attention. By designing a user-friendly interface for the recommendation system, we could significantly elevate the overall user experience. A well-crafted user interface would facilitate the input of song preferences, and present the recommended songs in an engaging, visually pleasing format.

This improvement in user interaction wouldn't just make the system more accessible and user-friendly, but it would also stimulate user engagement. An active user community can significantly enrich the system's data, enhancing the system's ability to generate highly accurate and personalized song recommendations. Consequently, by improving the interface and the underlying algorithms, our recommendation system would become even more powerful and effective in catering to diverse musical preferences.

VII. Conclusion

Our project involves the implementation and analysis of five distinct machine learning techniques, namely Support Vector Machines (SVM), Decision Trees, Random Forests, Naive Bayes, and K-Nearest Neighbors (KNN). Through our research, we have been able to explore and evaluate the various approaches that are most suitable for content-based recommendation systems in the context of music.

The Decision Tree algorithm, in particular, demonstrated the highest performance among the methods we tested, as it effectively uses feature-based information from each song to create branching structures for classification. However, we believe that the accuracy of the Decision Tree and other methods could be further improved with techniques such as pruning and additional preprocessing of the data.

Having gained insights into content-based recommendation systems, our next step would be to explore the potential of hybrid and collaborative filtering methods. Hybrid filtering combines the strengths of both content-based and collaborative filtering techniques, providing a more comprehensive approach to recommendation systems. Collaborative filtering, on the other hand, focuses on leveraging user preferences and interactions to make recommendations, adding a new dimension to the system.

By integrating these methods, we aim to create a more robust and accurate music recommendation system that can cater to a diverse range of user preferences and experiences, ultimately enhancing the overall user satisfaction and engagement with the platform.

References

[1] So Yeon Park and Blair Kaneshiro. 2021. Social Music Curation That Works: Insights from Successful Collaborative Playlists. Proc. ACM Hum.-Comput. Interact. 5, CSCW1, Article 117 (April 2021), 27 pages. https://doi-org.lib-proxy.fullerton.edu/10.1145/3449191

[2] Celma, Òscar. Music Recommendation and Discovery : The Long Tail, Long Fail, and Long Play in the Digital Music Space, Springer Berlin / Heidelberg, 2010. ProQuest Ebook Central, https://search.proquest.com/legacydocview/EBC/646113.

[3] Andres Ferraro, Dmitry Bogdanov, Jisang Yoon, KwangSeob Kim, and Xavier Serra. 2018. Automatic playlist continuation using a hybrid recommender system combining features from text and audio. In Proceedings of the ACM Recommender Systems Challenge 2018 (RecSys Challenge '18). Association for Computing Machinery, New York, NY, USA, Article 2, 1–5. https://doi-org.lib-proxy.fullerton.edu/10.1145/3267471.3267473

[4] B. Kostek, "Listening to Live Music: Life Beyond Music Recommendation Systems," 2018 Joint Conference - Acoustics, Ustka, Poland, 2018, pp. 1-5, doi: 10.1109/ACOUSTICS.2018.8502385.

[5] Elbir, A., and N. Aydin. "Music Genre Classification and Music Recommendation by Using Deep Learning." Electronics Letters, vol. 56, no. 12, 2020, pp. 627–29, https://doi.org/10.1049/el.2019.4202.

[6] Jannach, Dietmar, et al. Recommender Systems : An Introduction, Cambridge University Press, 2010. ProQuest Ebook Central, https://www.google.com/legacydocview/EBC/585299?accountid=9840.

[7] N. V. Durga Malleswari, K. Gayatri, K. Y. Sai Kumar, N. Likhita, P. K and D. Bhattacharyya, "Music Recommendation System using Hybrid Approach," 2023 Second International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, 2023, pp. 1560-1564, doi: 10.1109/ICEARS56392.2023.10085059.

[8] Muh-Chyun Tang, and Mang-Yuan Yang. "Evaluating Music Discovery Tools on Spotify: The Role of User Preference Characteristics." Tu Shu Zi Xun Xue Kan, vol. 15, no. 1, 2017, pp. 001–16, https://doi.org/10.6182/jlis.2017.15(1).001.

[9] A. Singh, U. Porwal and P. Selvaraj, "Hybrid Music Recommendation System Utilizing Item Popularity and Content-Based Filtering Innovations in music," 2022 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI), Chennai, India, 2022, pp. 1-6, doi: 10.1109/ICDSAAI55433.2022.10028966.

[10] Marko Balabanović and Yoav Shoham. 1997. Fab: content-based, collaborative recommendation. Commun. ACM 40, 3 (March 1997), 66–72. https://doi-org.lib-proxy.fullerton.edu/10.1145/245108.245124

[11] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for Music Analysis," arXiv.org, https://arxiv.org/abs/1612.01840 (accessed May 9, 2023).