Syncify: A Predictive Analytics Framework to Generate Song Recommendations

Syed Ali Haider* Jasmeen Kaur†
Courant Institute of Mathematical Sciences
Department of Computer Science
New York University, USA
Email: sh6070@nyu.edu* jk7297@nyu.edu†

Abstract—**The Syncify project aims to develop a robust music recommendation system that synchronizes with users' moods and preferences by leveraging advanced data science techniques. Recommendation systems are crucial in the digital age, enhancing user engagement by providing personalized content. Utilizing datasets from Spotify and Musixmatch, Syncify integrates song metadata, audio features, and lyrics to build an item-based recommendation framework. Our methodology includes data preprocessing, feature engineering, and dimensionality reduction using Doc2Vec, PCA, and SVD. By implementing cosine similarity and hybrid models that combine audio features with lyric embeddings, we generate accurate music recommendations. User feedback, incorporated through an A/B/n testing framework, refines and optimizes the algorithms. This paper presents Syncify's methodologies and evaluations, highlighting its potential to improve user engagement and satisfaction in music streaming services.**

I. Introduction

In today's digital age, the growth of music streaming services has transformed how individuals discover and consume music. Recommendation systems have become an integral component of these platforms, providing personalized music suggestions to users. These systems not only enhance the user experience by simplifying the discovery of new tracks and artists but also play a crucial role in driving user engagement and retention. As users receive recommendations that align with their tastes, they are more likely to spend more time on the platform, explore new music, and maintain subscriptions.

The importance of recommendation systems in the music industry is underscored by their significant impact on revenue. Companies like Spotify, Apple Music, and Amazon Music have leveraged advanced recommendation algorithms to curate personalized playlists, which in turn increase user satisfaction and loyalty. According to a report by McKinsey [1], effective recommendation systems can increase sales by $10 - 30\%$ for e-commerce platforms, a metric that is also relevant for streaming services as 75 percent of what consumers watch on Netflix comes from product recommendations based on recommendation algorithms. Spotify's "Discover Weekly" playlist, for example, has been a key feature in maintaining high levels of user engagement, showcasing the power of personalized recommendations.

Beyond their technical and commercial aspects, music recommendation systems also influence social dynamics. They shape musical tastes, promote emerging artists, and facilitate the discovery of niche genres.

II. Related Work

Song recommendation systems have become a cornerstone of music streaming services, employing various techniques to provide users with personalized music suggestions. The underlying methodologies of these systems typically involve collaborative filtering, content-based filtering, or a hybrid approach. Collaborative filtering is a widely used method in recommendation systems, including those for music. This technique predicts a user's preferences by analyzing the listening habits of similar users. Key advancements in collaborative filtering include matrix factorization techniques and deep learning approaches. Matrix factorization techniques, such as Singular Value Decomposition (SVD), have been fundamental in the development of recommendation systems. Golub and Reinsch [2]'s work on SVD laid the foundation for its application in recommendation systems, allowing for the decomposition of user-item interaction matrices to uncover latent factors that represent user preferences and item attributes . Koren, Bell, and Volinsky [3] further advanced this field by demonstrating the effectiveness of matrix factorization in the context of the Netflix Prize competition, highlighting its ability to capture complex patterns in user ratings Koren [4]. Recent advancements have seen the integration of deep learning techniques into recommendation systems. Zhang, Yao, and Sun [5] provide a comprehensive survey of deep learning-based recommender systems, discussing various architectures such as autoencoders and neural collaborative filtering that enhance recommendation accuracy by learning intricate representations of user-item interactions . Salakhutdinov, Mnih, and Hinton [6] introduced Restricted Boltzmann Machines for collaborative filtering, demonstrating their potential in capturing complex user-item relationships.

Content-based filtering recommends items based on the intrinsic properties of the items themselves. In music recommendation systems, this involves analyzing audio features such as tempo, rhythm, and genre to suggest songs that are similar to those the user has previously enjoyed. Systems like Spotify extensively utilize content-based filtering by analyzing detailed song features and track metadata. This allows for

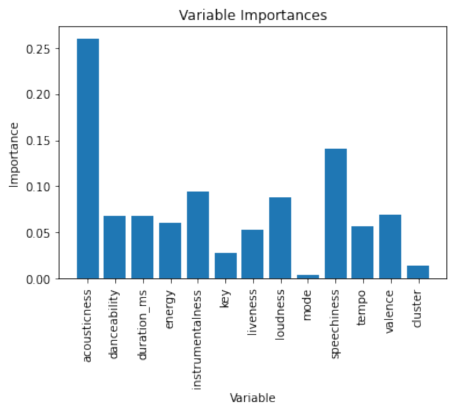

Fig. 1: Feature Importance

precise recommendations based on the specific characteristics of each track, enhancing user satisfaction by suggesting songs that align closely with their preferences.

Current content-based music recommendation systems typically focus on song metadata and audio features but often do not include lyrics as a feature. Syncify aims to fill this gap by developing a music recommendation framework that aligns with users' moods and preferences while incorporating lyrics as a significant aspect of the recommendation process. Incorporating lyrics into music recommendation systems adds another layer of personalizing by considering the textual content of songs. This approach can capture the emotional and thematic elements of music that are not evident from audio features alone.

III. IMPLEMENTATION

A. Dataset Exploration

We conducted an extensive exploration of various datasets, each possessing unique features crucial for music recommendation system development. The primary challenge encountered was the availability of rich content metadata and user ratings, which is essential for building collaborative filtering and user centric recommendation models. Additionally, except for Yahoo Music, none of the datasets included user ratings, posing a significant limitation.

Here is a concise overview of the datasets we explored:

- **Last.fm**: The Million Song Dataset by Last.fm [7] offers a vast repository of one billion listening events, providing valuable insights into user behavior. However, it lacks rich metadata for effective recommendation generation, and there is no direct mapping between users and music tracks.
- **Yahoo Music**: With over 717 million ratings from 1.8 million users for 136 thousand songs, Yahoo Music [8] provides a rich source of user interaction data. However, it lacks metadata such as real song and artist names, hindering its suitability for robust model evaluation and A/B testing.
- **Spotify**: Spotify [9] stands out for its detailed song features, 3.7 million unqiue track with metadata, and extensive user data but that is only on the playlist level and engagement no ratings are provided. But leveraging the Spotify API enables seamless A/B testing and evaluation of content based recommendation models. Despite the absence of a user rating feature, Spotify emerged as a key dataset due to its various features, access to API for extracting additional metadata, and the ability to map it to the MusixMatch Dataset.
- **Musixmatch**: This dataset contains song lyrics, which can complement the Million Song Dataset [10] by enriching its features. Mapping Musixmatch data to Spotify enhances the richness of song metadata, facilitating more accurate recommendations.

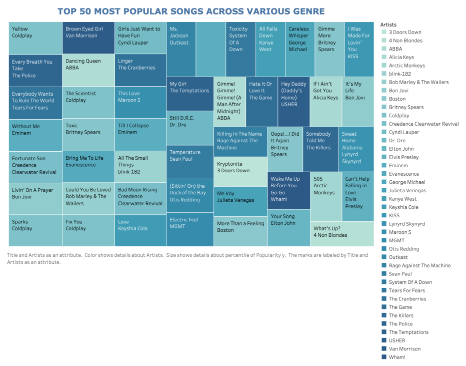

Fig. 2: Top 50 songs in various genre

Our exploration revealed the importance of rich metadata and user interaction data in building effective music recommendation systems. Spotify emerged as a key dataset due to its comprehensive metadata, various features, and the ability to map it to other datasets such as Musixmatch.

B. Dataset Understanding

To get a better understanding of our datasets, we did some preliminary Exploratory Data Analysis using RapidMiner, Jupyter Notebook and Tableau. We explored some statistics to understand how our data is formatted, look at some trends through the years and explored the correlation between various features to predict the popularity of a song. These insights allowed us to understand that the dataset we are using is not as trendy as we would like it to be since we are looking at songs from 2018 to 1920.

Initially we explored the top 50 genre's and their top songs as seen in Figure 1, we can see that most of these are making sense as these are popular artists and songs in their genres.

TABLE I: Predicting Popularity of a Song Using Audio Features

Model	MAE	MSE	RMSE	R-squared
Random Forest	8	159	12.63	0.67
Gradient Boosting	9	183	13.56	0.62
Decision Tree	10	320	17.89	0.34
Linear Regression	13	300	17.33	0.37

We can see the feature importance from the Figure 1. We saw that acousticness had the highest importance of 0.26 followed by speechiness with importance of 0.14. This gives

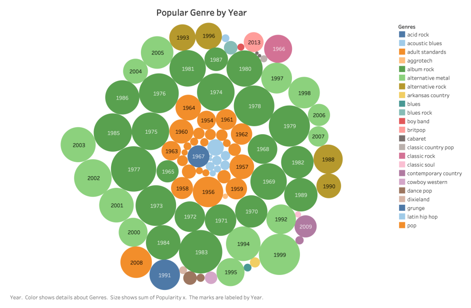

Fig. 3: Popular genre by year

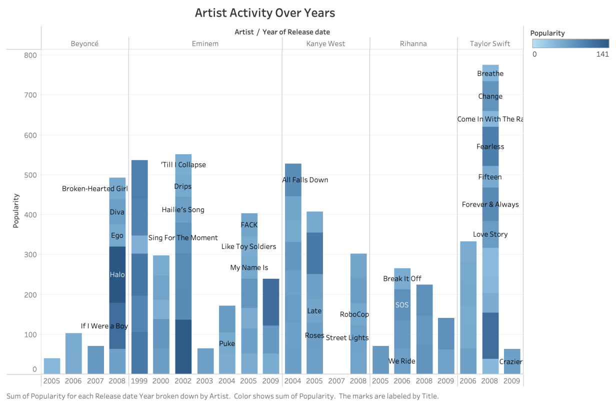

Fig. 4: Artist's activity over years

us a fair idea that there is no one feature that is dominating in the importance since 0.14 and 0.26 are both not very high.

Further we explored which are some of the most popular genre in our dataset through the years and saw alot of rock and metal songs, which can be explained due to the skew in the data that it has more years in the early 90's and 2000's when this type was music was more popular as seen in Figure 2.

We wanted to see how the trends in various artists' popularity is going in our dataset, so we looked at the top 5 artists and saw how popular their songs by the year were, to see if we can see trends in music taste. This can be seen in the Figure 3. This allows us to know that the dataset has missing values from the actual data because for example Taylor Swift's popularity has increased further in the recent years.

C. *Lyrics & Audio Feature Model*

The integration of lyric data with audio features presents a new approach to enhancing the accuracy and personalization of music recommendation systems. This model leverages both textual content from song lyrics and numerical metadata from audio tracks to create a comprehensive feature set for song recommendation. The innovation lies in the dual-modality analysis, which combines semantic understanding from lyrics with acoustic properties from audio features. The framework for this model is shown in Figure 1.

1) *Data Preprocessing:*

a) Lyric Data Processing: The lyric data is sourced from the MusixMatch dataset, which provides a bag-of-words representation for each song. This raw data undergoes significant preprocessing to clean and normalize the text. Stopwords are removed to focus on the most relevant terms, and special characters are stripped away. This process ensures that the textual data is in a consistent format suitable for embedding generation.

b) Audio Metadata Processing: Audio features are derived from Spotify's metadata, encompassing various acoustic attributes such as danceability, energy, key, loudness, and tempo. This metadata is also cleaned and standardized. Artist and song names are normalized by converting to lowercase and removing special characters, ensuring compatibility between different datasets.

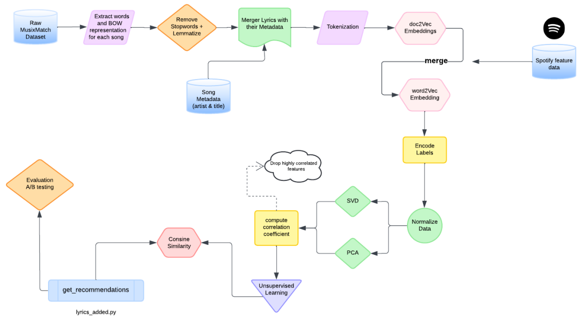

Fig. 5: Process Chart with Lyrics

2) *Embedding Generation:*

To capture the semantic content of lyrics, a Doc2Vec model is employed. This model converts the cleaned lyrics into dense vector representations, effectively embedding each song into a high-dimensional space where similar songs are located closer together. This embedding process transforms textual data into a format that can be easily integrated with numerical audio features.

a) Sampling Text Windows: At each iteration of training, a contiguous window of text is sampled from the lyrics of a song, forming the basis for constructing a paragraph matrix.

b) Word Prediction: Within the sampled window, a random word is selected. The model aims to predict this target word using the paragraph matrix as input.

c) Parameter Updates: As training progresses, the model undergoes parameter updates to optimize the prediction task. This involves fine-tuning the weights associated with the paragraph vectors.

Mathematically, the Doc2Vec model can be represented as:

$$\text{Doc2Vec}(w, t) = \text{Softmax}(w \cdot t)$$

where: w represents the weights associated with the paragraph matrix. t represents the target word vector. Softmax is the softmax function used for prediction. Through iterative training, the model learns to embed songs in a high-dimensional space, facilitating subsequent analysis and recommendation tasks.

3) *Integration of Lyrics and Audio Features:*

The integration process involves merging the lyric embeddings with the audio features to form a unified dataset. This is achieved by joining datasets on common attributes such as artist and song titles. The resulting dataset combines the semantic richness of lyrics with the quantitative details of audio features, providing a holistic representation of each song.

4) Dimensionality Reduction: Given the high dimensionality of the combined dataset, dimensionality reduction techniques such as Principal Component Analysis (PCA) and Truncated Singular Value Decomposition (SVD) were applied. These methods reduce the number of features while preserving the most significant variance within the data. This step not only enhances computational efficiency but also mitigates the risk of overfitting in subsequent modeling stages.

a) Normalization: Normalization is performed to ensure that all features contribute equally to the analysis. The min-max normalization formula is given by:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

where X is the original value, X' is the normalized value, $X_{\min}$ is the minimum value in the feature, and $X_{\max}$ is the maximum value in the feature.

b) Principal Component Analysis (PCA): Then we applied PCA. PCA reduces the dimensionality of the dataset by transforming the original features into a new set of orthogonal features (principal components) that capture the maximum variance. The transformation is defined as:

$$Z = XW$$

where X is the normalized data matrix, W is the matrix of eigenvectors, and Z is the matrix of principal components. We computed for 20 Principal Components.

c) Singular Value Decomposition (SVD): SVD decomposes the data matrix into three matrices: U, Σ, and V. The decomposition is given by:

$$X = U\Sigma V^T$$

where X is the original data matrix, U and V are orthogonal matrices, and Σ is a diagonal matrix of singular values. Truncated SVD selects the top k components to reduce dimensionality. Where top k singular values and corresponding vectors are retained, the multiplication involves only the first k columns of U, the first $k \times k$ submatrix of Σ, and the first k rows of V^T. This effectively reduces the dimensionality of the data while retaining the most important information. We choose k to be 20 in our case.

5) Correlation Analysis: A critical step in the model is the analysis of feature correlations. By identifying highly correlated features, we can eliminate redundant data, thereby simplifying the model and improving its interpretability. This process ensures that the final feature set is both comprehensive and non-redundant, optimizing the model's performance. We used the Pearson correlation with 0.5 as a threshold. The Pearson correlation coefficient, often denoted as r, is a measure of the linear correlation between two variables X and Y. It is calculated using the following mathematical formula:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X}) \cdot (Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2} \cdot \sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

This formula calculates the Pearson correlation coefficient r as the ratio of the covariance of X and Y to the product of their standard deviations. It measures the strength and direction of the linear relationship between the two variables. In our analysis we found that no two variables were lying in the threshold values, so we didn't drop any features.

6) Similarity Calculation: In our model, we utilized cosine similarity as a measure to quantify the similarity between two vectors. The cosine similarity between two vectors $\mathbf{t}$ and $\mathbf{e}$ is calculated using the formula:

$$\cos(t, e) = \frac{\mathbf{t} \cdot \mathbf{e}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} t_i e_i}{\sqrt{\sum_{i=1}^{n}(t_i)^2}\sqrt{\sum_{i=1}^{n}(e_i)^2}}$$

where t_i and e_i are the components of vectors $\mathbf{t}$ and $\mathbf{e}$ respectively. $\sum_{i=1}^{n} t_i e_i$ denotes the dot product of vectors $\mathbf{t}$ and $\mathbf{e}$. $\sqrt{\sum_{i=1}^{n}(t_i)^2}$ and $\sqrt{\sum_{i=1}^{n}(e_i)^2}$ represent the magnitudes (Euclidean norms) of vectors $\mathbf{t}$ and $\mathbf{e}$ respectively.

This formula computes the cosine of the angle between the two vectors, providing a value between -1 and 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates perfect dissimilarity. By comparing the cosine similarity values between vectors, we can effectively identify the most similar songs in our dataset, facilitating accurate song recommendations for users.

7) Example of Recommendations: To illustrate the effectiveness of the model, we provide recommendations for the song "Strawberry Swing" by Coldplay. Using the SVD-based similarity matrix, the following recommendations are generated:

Since you listened to Strawberry Swing, you might also like:

1) Index: 10633 Name: The Spangle Maker By: Cocteau Twins
2) Index: 699 Name: Purple Haze By: Jimi Hendrix
3) Index: 1931 Name: Only Shallow By: My Bloody Valentine

Similarly, using the PCA-based similarity matrix, the recommendations are:

Since you listened to Strawberry Swing, you might also like:

1) Index: 10633 Name: The Spangle Maker By: ['Cocteau Twins']
2) Index: 699 Name: Purple Haze By: ['Jimi Hendrix']
3) Index: 1585 Name: I Ran (So Far Away) By: ['A Flock Of Seagulls']

These recommendations demonstrate the model's ability to suggest contextually relevant songs based on both lyrical content and audio features, thus providing a more personalized listening experience.

In summary, the Lyrics + Audio Feature Model explores how the fusion of semantic and acoustic data can lead to more sophisticated and personalized music recommendation systems.

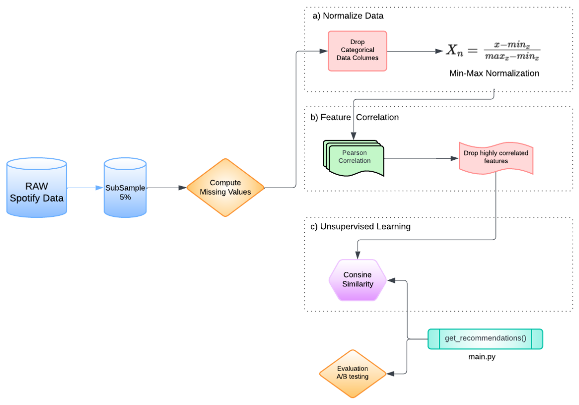

Fig. 6: Naive Audio Feature Model Process Chart

D. *Naive Audio Feature Model*

In this section, we present a naive model that focuses solely on leveraging audio features to recommend songs to users. By analyzing numerical metadata extracted from audio tracks, our model aims to provide personalized recommendations based on the acoustic properties of songs, this is used as our base model to test against the Lyrics & Audio Model in the A/B testing. You can see the structure of this in Figure 2.

1) Data Preprocessing: The data preprocessing stage involves several steps to ensure the quality and compatibility of the audio feature data. In order to compute the various matrix we had to reduce our dataset to just 5% for computational ease. Then, we perform min-max normalization on numerical features to scale them within a consistent range, enabling fair comparison and analysis across different features.

Additionally, we conduct a Pearson correlation analysis to identify highly correlated features. By eliminating redundant data, we enhance the efficiency and interpretability of the model. Features with high correlation coefficients are either removed or combined to reduce dimensionality and enhance the model's performance.

We did dimension reduction using both SVD and PCA to see which one performs better and saw similar matrix for both.

2) Recommendation Generation: The recommendation generation process relies on cosine similarity to identify songs with similar audio features. By computing the cosine similarity between feature vectors of songs, our model identifies songs that share similar acoustic properties.

IV. Evaluation

In evaluating the two recommendation models—Model 1, which uses Spotify song features, and Model 2, which combines lyrics with Spotify song features—we utilized a comprehensive A/B testing approach. Given the subjective nature of music recommendation, the effectiveness of these models is primarily assessed through user interactions and feedback. Users are randomly assigned to either model and presented with song recommendations based on their previous ratings and interactions. The evaluation process involves tracking several key metrics:

- **User Likes and Dislikes**: Users can express their preferences by liking or disliking the recommended songs. A higher number of likes indicate a better performance of the recommendation model in capturing user preferences.
- **Playlist Additions**: One of the most significant indicators of a recommendation's success is if the user adds the recommended song to their Spotify playlist. This action demonstrates a high level of satisfaction and trust in the recommendation, thereby providing a strong endorsement for the model's effectiveness.
- **User Engagement**: The frequency of user interactions with the recommendations, including likes, dislikes, and playlist additions, is monitored. Higher engagement rates suggest that the model is effectively capturing the users' music tastes and keeping them engaged.

Metrics such as novelty, diversity, and serendipity are also crucial in assessing the performance of recommendation models, as they ensure users are exposed to a wide range of options and discover new and relevant content as referenced in Unknown [11] Amazon Web Services [12]. Hence the use of A/B testing in a live environment allows us to gather empirical evidence on how user interactions influence the effectiveness of our recommendations, thus providing a robust evaluation framework as also mentioned in Amazon Web Services [12].

Preliminary results from the A/B testing are still being produced since a larger sample size of user interactions is required to draw more definitive conclusions. The evaluation process remains dynamic, with continuous feedback loops to refine and enhance the recommendation algorithms based on user interactions.

V. Deployment

A. *Web Application Architecture and Security*

The web application is developed using the Flask web framework, providing a robust foundation for handling HTTP requests and rendering responses. User authentication is managed through Flask-Login, which facilitates session management and ensures that features such as song recommendations are accessible only to authenticated users. For database interactions, SQLAlchemy is employed, allowing for efficient querying and storage of user data and song metadata.

Security is a priority in the application design. The werkzeug.security library is utilized to hash user passwords during registration and validate users by comparing these hashes during the login process. As also mentioned in MarketSplash [13] SecureCoding [14], this approach safeguards user credentials against potential data breaches. Environment variables are strategically used to securely integrate the Spotify API, ensuring that sensitive information remains decoupled from the source code.

B. *User Interaction and Session Management*

As displayed in Figure 7, Upon accessing the web application, users are prompted to log in or register, initiating a secure session that ties their actions to a personalized database entry. The homepage presents users with two primary actions:

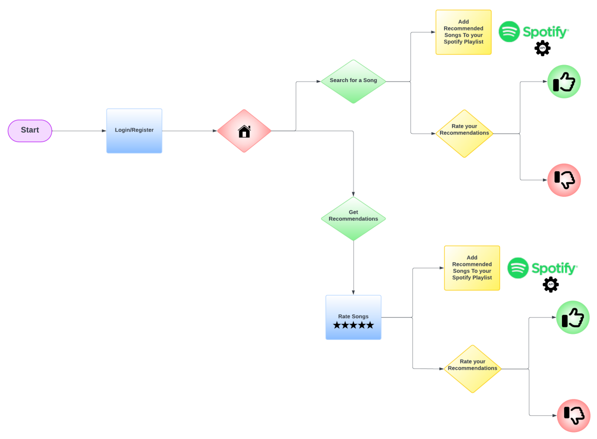
Start
Login/Register
Search for a Song
Add Recommended Songs To your Spotify Playlist
Spotify
Rate your Recommendations
Get Recommendations
Rate Songs
★★★★★
Add Recommended Songs To your Spotify Playlist
Spotify
Rate your Recommendations

Fig. 7: Flow Chart of User on Syncify App

searching for songs or obtaining recommendations. When the "Get Recommendation" button is clicked, the system displays five randomly selected songs. Users rate these songs, and based on the highest-rated song, the system recommends five similar tracks. Users can upvote or downvote these suggestions, influencing future recommendations and enabling the system to adapt to their preferences for our collaborative-filtering based recommendation system.

C. *Spotify API Integration for Playlist Management*

A core feature of Syncify is its seamless integration with Spotify's Web API, allowing users to add recommended songs directly to their Spotify playlists. The integration process begins with user authentication through the Spotify OAuth flow, which securely grants the application access to the user's Spotify account. Once authenticated, the application can interact with Spotify's API endpoints to manage playlists and tracks.

When a user selects a song to add to their playlist, Syncify first searches for the song on Spotify using the sp.search method, querying by song name and artist. If the song is found, the application retrieves the song's Spotify ID. Next, the web application checks the user's existing playlists to see if a playlist named "My Recommended Songs" exists. If the playlist does not exist, the application creates a new one. The song is then added to the user's Spotify playlist, passing the playlist ID and the song ID. Throughout this process, appropriate flash messages are displayed to inform the user of the success or failure of each action, enhancing user experience. This integration not only personalizes the music recommendation experience but also leverages Spotify's extensive music catalog, making it convenient for users to manage and enjoy their music seamlessly.

Therefore it allows us to enhances user satisfaction along with providing additional data points for improving the accuracy of the recommendation algorithms. Hence the web application ensures a robust, secure, and user-friendly music recommendation system that enhances the user experience and satisfaction.

VI. Discussion

A. *Data Integration Challenges*

A significant portion of our effort was dedicated to data preprocessing and integration, a common challenge in data science projects. While the Spotify dataset provided clean and comprehensive audio feature metadata, integrating it with the MusixMatch lyric data posed considerable challenges. Merging the Spotify and MusixMatch datasets required meticulous string manipulation to ensure accurate matches between song titles and artists. Despite our best efforts, a significant amount of data was lost due to mismatches. This issue was particularly pronounced for newer songs, leading to a dataset biased towards older tracks. This skew impacted the diversity of our recommendations and highlighted the difficulty of achieving a perfect dataset.

B. *Lack of User Ratings*

Another major challenge was the absence of user ratings, which are crucial for validating and refining recommendation systems. Without ground truth data, it was mathematically challenging to evaluate our models against standard metrics. Most existing recommendation systems benefit from extensive rating datasets available for movies and books, but such datasets are scarce for music. This lack of user feedback necessitated innovative approaches to enhance our content-based models, which have been the industry standard for a long time.

C. *Addressing Evaluation Issues*

The lack of user rating history complicated the testing and evaluation of our models. To overcome this, we implemented an online live recommendation system for Synifcy. Although our current user base for real-time A/B testing is small, we plan to expand this to achieve robust statistical rigor in our evaluations. This system includes a rating feature, allowing users to provide feedback on recommendations. Collecting and analyzing this feedback will enable us to continuously improve our models.

D. *Innovative Approaches*

Given the limitations of existing datasets and the absence of user ratings, we focused on enhancing our content-based models. We integrated lyrical content with audio features, leveraging techniques like PCA and SVD for dimensionality reduction. This dual-modality approach aimed to provide more personalized and contextually relevant recommendations. However, the complexity of merging high-dimensional data required careful preprocessing and sophisticated analysis techniques.

E. *Future Directions*

To mitigate the issues we encountered, we plan to expand our live recommendation system and gather a larger user base for real-time feedback. This expansion will facilitate A/B testing, providing valuable data for refining our models. Additionally, we intend to make our rating data publicly available, supporting future research and addressing a significant gap in the current landscape of music recommendation systems.

F. Conclusion

In summary, our project underscored the complexities and trade-offs inherent in developing music recommendation systems. Data integration challenges and the lack of user ratings were significant hurdles. However, our innovative approaches to enhancing content-based models and implementing a live recommendation system demonstrate the potential for overcoming these challenges. Future work will focus on expanding our user base, refining our algorithms, and contributing to the broader research community by sharing our data and insights.

G. Future Work

For future work on our music recommendation system, several enhancements are planned to improve its functionality and user experience. Firstly, we aim to increase the dataset, leveraging the Genius API and MusixMatch to enrich our song database with comprehensive lyrics and metadata. Currently, the MusixMatch API limits us to 2,000 tokens per day, and with the aim to match up to 3.7 million songs, scraping all the data will take time. Once complete, we expect to have richer metadata, resulting in more accurate and comprehensive song recommendations. Additionally, we will implement intensive mood sentiment analysis, as mentioned by Raschka [15] and we can associating specific moods with acoustic features as mentioned by Spotify [16]: angry with speechiness, sad with acousticness, energetic with energy, happy with valence, and relaxing with instrumentalness. To further refine recommendations, we will develop new recommendation system models that suggest songs based on the user's mood and predict the mood trends of different regions or countries.

Moreover, integrating the Spotify API will allow for real-time data processing and seamless retrieval of recommendations, enhancing the system's responsiveness and accuracy. We also plan to continuously improve the application's functionality, ensuring a smooth and engaging user experience. Additionally, once we collect enough user data from our application, we will implement collaborative filtering and a hybrid model to tailor song recommendations more effectively based on users' liked songs and preferences stored in their database. These advancements will collectively enhance the system's capability to deliver personalized and contextually relevant music recommendations.

References

[1] J. Boudet, B. Gregg, J. Wong, and G. Schuler, "What shoppers really want from personalized marketing," *McKinsey & Company*, 2017. [Online]. Available: https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/what-shoppers-really-want-from-personalized-marketing.

[2] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[4] Y. Koren, "The bellkor solution to the netflix grand prize," *Netflix prize documentation*, vol. 81, pp. 1–10, 2009.

[5] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: A survey and new perspectives," *arXiv preprint arXiv:1707.07435*, 2017.

[6] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning*, ACM, 2007, pp. 791–798.

[7] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere, "The million song dataset.," Jan. 2011, pp. 591–596.

[8] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer, "The yahoo! music dataset and kdd-cup'11," in *Proceedings of KDD Cup 2011*, G. Dror, Y. Koren, and M. Weimer, Eds., ser. Proceedings of Machine Learning Research, vol. 18, PMLR, 21 Aug 2012, pp. 3–18. [Online]. Available: https://proceedings.mlr.press/v18/dror12a.html.

[9] *Spotify api documentation*, https://developer.spotify.com/documentation/web-api/.

[10] *Musixmatch dataset: The official lyrics collection for the million song dataset*, http://millionsongdataset.com/musixmatch, Accessed: 2024-05-17.

[11] A. Unknown, "A comprehensive survey of evaluation techniques for recommendation systems," 2023, Accessed: 2024-05-17. [Online]. Available: https://arxiv.org/abs/2312.16015.

[12] Amazon Web Services. "Using a/b testing to measure the efficacy of recommendations generated by amazon personalize." Accessed: 2024-05-17. (2023), [Online]. Available: https://aws.amazon.com/blogs/machine-learning/using-ab-testing-to-measure-the-efficacy-of-recommendations-generated-by-amazon-personalize/.

[13] MarketSplash. "How to secure flask applications: Essential strategies and best practices." Accessed: 2024-05-17. (2024), [Online]. Available: https://marketsplash.com/how-to-secure-flask-applications/.

[14] SecureCoding. "Best practices for flask security." Accessed: 2024-05-17. (2024), [Online]. Available: https://www.securecoding.com/blog/flask-security-best-practices/.

[15] S. Raschka, "Musicmood: Predicting the mood of music from song lyrics using machine learning," Nov. 2016.

[16] Spotify. "Get audio features." (Accessed 2024), [Online]. Available: https://developer.spotify.com/documentation/web-api/reference/get-audio-features.