Regional Music Trend Using Spotify Data

Arpan Das
arda3761@colorado.edu
CSCI 6502-002
University of Colorado Boulder
Boulder, Colorado, USA

Debrup Basu
deba8184@colorado.edu
CSCI 6502-001
University of Colorado Boulder
Boulder, Colorado, USA

Shaily Goyal
shgo4037@colorado.edu
CSCI 6502-001
University of Colorado Boulder
Boulder, Colorado, USA

ACM Reference Format:
Arpan Das, Debrup Basu, and Shaily Goyal. 2022. Regional Music Trend Using Spotify Data. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

Spotify is a music streaming service with hundreds of millions of listeners from all over the world, all with different listening preferences. There are potentially trends to unearth with all the listening data available. The analysis includes finding the trending artists and songs based on a region, popularity of a song over time based on certain features, and sentiment analysis of the song lyrics.

The project aims to improve music discovery on Spotify by using "Jammin' App". It's a Streamlit app that searches and explores music, using the Spotify API to retrieve search results. The app offers features like polar plots for song features and recommendations for similar songs. It also utilizes the Spotify API to retrieve search results and enables users to select and explore their desired songs, artists, and albums. The project also conducts a lyrical sentiment analysis on popular songs to gain insight into their emotional perception. This analysis will provide insight into the audience's perception of the music, allowing for a better understanding of its reception and potentially informing decisions about future music production.

2 INTRODUCTION

Spotify is a music streaming service with hundreds of millions of listeners from all over the world, all with different listening preferences. There are potentially trends to unearth with all the listening data available. The analysis includes finding the trending artists and songs based on a region, popularity of a song over time based on certain features, and sentiment analysis of the song lyrics.

Our goal is to analyze trending music across across the US, and use sentiment analysis to know audience reactions to the trending music. We opted for US only so our lyric analysis is consistent to the English language. For some measurable output, we will compare different language models and compare their accuracies.

3 RELATED WORK

The paper "Clustering Pop Songs Based On Spotify Data Using K-Means And K-Medoids Algorithm" provides a technique for grouping pop songs on Spotify based on their audio and textual features [4]. The study extracts auditory characteristics including pace, energy, and valence, as well as textual characteristics like song title, artist, and lyrics. To group songs based on these attributes, the K-means and K-medoids clustering methods are used.

Another paper describes a unique statistical model that may be used to evaluate music consumption data from Spotify, a prominent streaming platform [5]. The authors offer a Beta Generalized Linear Mixed Model (GLMM) that can be used to estimate the popularity distribution of different songs as well as determine the elements that impact a song's popularity.

The authors in "Catching the Earworm: Understanding Streaming Music Popularity Using Machine Learning Models" focused on understanding music popularity in the context of music streaming platforms, specifically on Spotify [2]. It proposes machine learning models to predict song popularity on Spotify based on various audio and metadata features. The authors conducted experiments using a large dataset of over 170,000 songs and found that certain features were more important than others in predicting popularity, and that the importance of these features varied across different regions and genres.

The paper "Evolution of global music trends: An exploratory and predictive approach based on Spotify data" aims to analyze the sentiment of audience reactions to top trending music in different regions worldwide using natural language processing and sentiment analysis techniques [1]. The study collected data on top trending music from Spotify's Global Top 50 charts and found that the sentiment of tweets varied significantly across regions, indicating the influence of cultural, social, and historical contexts. It found that the sentiment of tweets strongly correlated with the popularity of songs in most regions, highlighting the importance of audience reactions in shaping music trends.

4 TOOLS AND TECHNOLOGIES

- Python as a programming language and libraries Tweepy to collect tweets, Pyspark for tweet processing, textblob for classification.
- MongoDB, a NoSQL document database for storage of Spotify data.
- Plotly/Tableau, Word Cloud for data visualization.
- Jupyter Notebook.
- Github for source code version control.
- Streamlit for the Jammin' App.
- Docker for containerization.
- Apache JMeter for load testing.
- Render for cloud deployment of app.

Conference'17, July 2017, Washington, DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

5 DESIGN AND IMPLEMENTATION

5.1 Dataset

The Spotify Hit Predictor Dataset (1960-2019) is a collection of features for tracks obtained using Spotify's Web API. This dataset is labelled with '1' or '0' ('Hit' or 'Flop'), which means that it can be used to train a classification model that predicts whether a track would be a 'Hit' or not. The dataset includes a variety of attributes such as the track name, artist name, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, durationMS, timeSignature, chorusHit, and sections. The values of these attributes can be used to make predictions on whether a track will be successful or not. The dataset is split into a 70:30 train-test split.

The dataset is a valuable resource for anyone interested in analyzing music and predicting whether a track will be successful or not. It is thanks to Spotify's API and their database of in-depth details of every track in their library that this dataset is available. The dataset includes attributes such as danceability, energy, and valence, which provide valuable insights into the musical features that are important for predicting whether a song will be a 'Hit' or not. The train-test split of 70:30 allows for accurate training and testing of models, ensuring that the predictions made are as accurate as possible. This dataset is a great starting point for anyone interested in music analysis and prediction.

Similarly for the Jamming' App we used Spotify's Web API to retrieve data about songs and artists, including track names, artists' names, genres, popularity, acousticness, danceability, energy, instrumentalness, key, liveness, loudness, mode, speechiness, tempo, time signature, and valence. The dataset is not pre-existing, but is created by the code by making API requests to Spotify.

The Billboard Hot 100 was scraped using RegeX and BeautifulSoup4 to grab songs from the last three years, from April 2020 to present. Then those songs' Spotify track features were scraped using the Spotify API and requests library. Those results were then stored in a MondoDB Atlas database for scalability. To determine the emotion in trending songs, we apply sentiment analysis on the song lyrics. For this task, we extracted data from Genius API for those songs on the MongoDB database. [3].

5.2 Spotify Data Visualisation using Tableau

1) We performed a comparative analysis of the Genre of the Song and different features like Popularity, Acousticness, Danceability, Energy, Loudness, Duration and Speechiness.

Use Case: The use case for conducting a comparative study of the song's genre and many aspects such as popularity, acousticness, danceability, energy, loudness, length, and speechiness of the Spotify API is to acquire insights into the music industry's patterns and trends. By examining the relationship between the genre and the various traits, one may determine which features are more crucial in establishing a genre's appeal. This study can assist music streaming companies like Spotify improve their recommendation algorithms by making more accurate predictions about the user's music choices using this data. It may also assist musicians and music producers identify which characteristics of a song are more likely to make it popular and utilize this information to create more successful music.

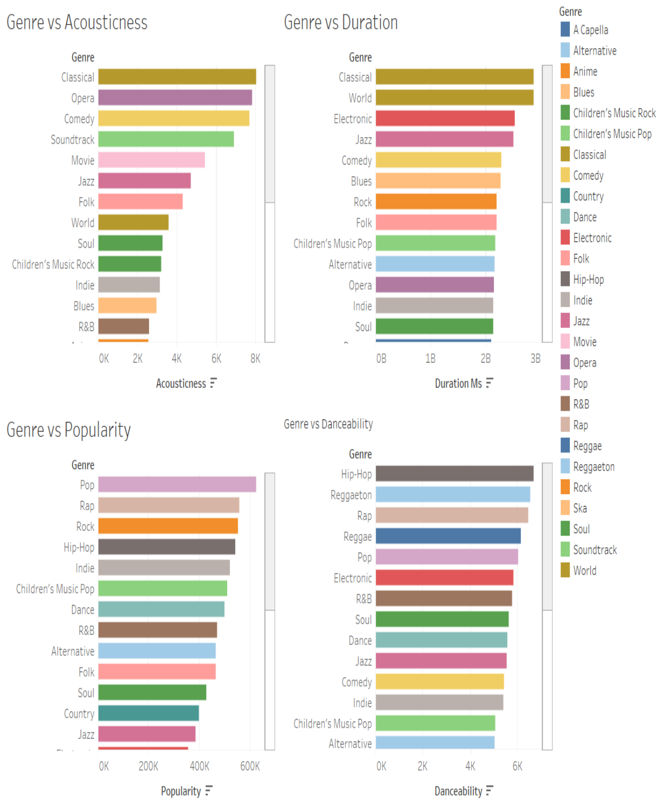


Figure 1

2) We also performed a comparison between the popularity of songs and explicitness.

Use Case: The use case of comparing the popularity of songs and explicitness can be to understand the audience's preferences in terms of the level of explicit content in the songs they listen to. This analysis can help music companies and artists to make informed decisions about the content they create and promote, based on the target audience's preferences.

3) We did a comparison of Year of Release of Songs between Average Duration, Explicitness, Acousticness, and Speechiness of Songs.

Use Case: The use case of comparing the Year of Release of Songs with different features like average duration, explicitness, acousticness, and speechiness of the songs is to gain insights into the relationship between the year a song was released and its characteristics. This comparison can be useful for music industry professionals to understand the trends and changes in the music industry over time, and to make informed decisions about the type of music

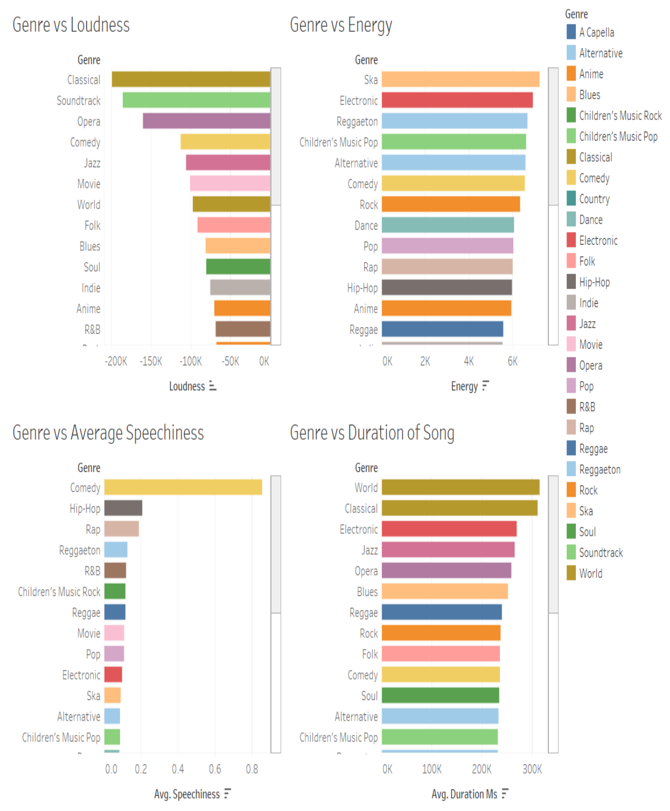


Figure 2

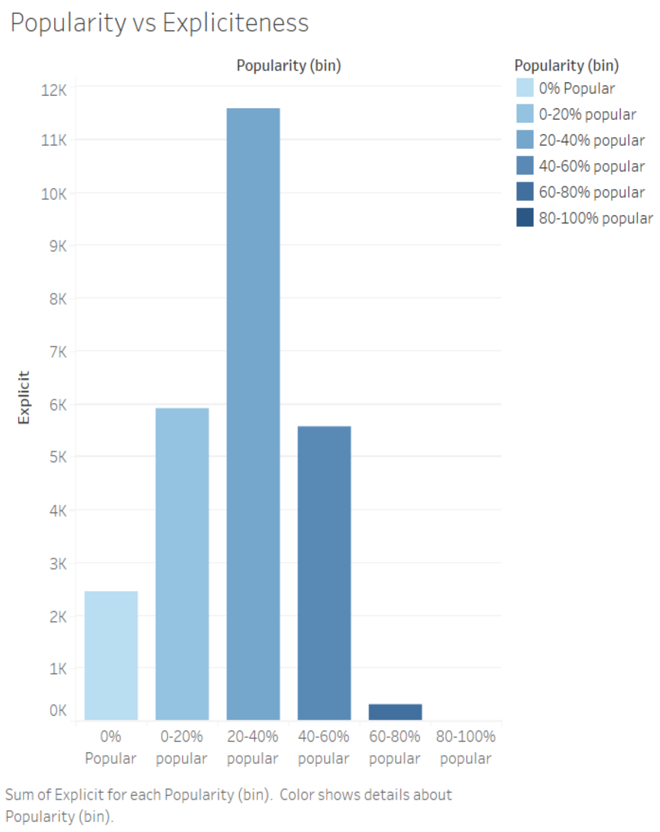


Figure 3

they produce, promote, or license.

4) Comparison of Energy and Average Danceability

Use Case: The use case of comparing Energy and Average Danceability is to analyze the relationship between the energy level and danceability of music. This type of analysis can be useful in various industries, such as music production, radio stations, and event planning.

5) Comparison of Average Popularity and Speechiness

Use Case: The use case of comparing the average popularity and speechiness of songs could be to identify any potential correlation or relationship between these two audio features. Popularity is a measure of how well-known and frequently a song is played, whereas speechiness is a measure of how much spoken word content (e.g., lyrics) a song contains. By comparing the average values of these two features, one can gain insights into whether there is a relationship between the popularity of a song and the amount of spoken word content it contains.

5.3 Audio Features of the tracks

We did an implementation of data analysis on music tracks and their respective albums. It begins by importing necessary libraries, including pandas, matplotlib, seaborn, and spotipy. Spotipy is used to extract data from the Spotify API for the music tracks and albums. After importing the libraries, it proceeds to extract data for various music tracks from the Spotify API, such as the album name, artist name, track name, release date, and more. The data is then stored in a pandas DataFrame to perform data analysis.

Further, we performed data cleaning to handle null values and unwanted columns. The DataFrame is then split into two separate tables: one for album data and the other for track data. The album data table contains information such as the album name, artist name, release date, and number of tracks, while the track data table contains information such as the track name, track duration, popularity score, danceability, energy, and more.

We then proceeded to visualize the data using various plotting techniques, such as scatter plots, bar charts, and heatmaps. The visualizations provide insights into the relationships between album popularity and track features such as danceability, energy, and tempo. Finally, we exported the cleaned and analyzed data to separate CSV files for further use.

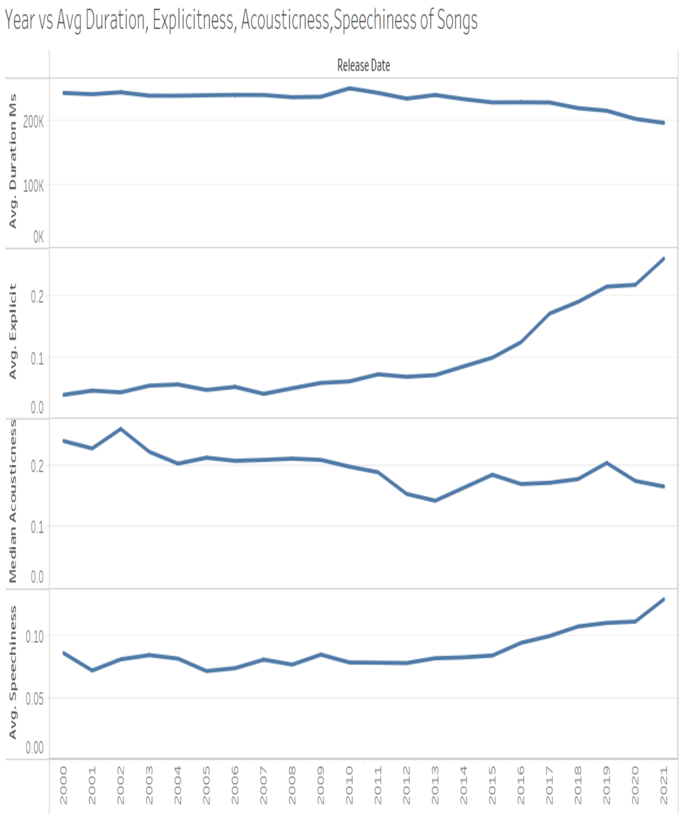


Figure 4

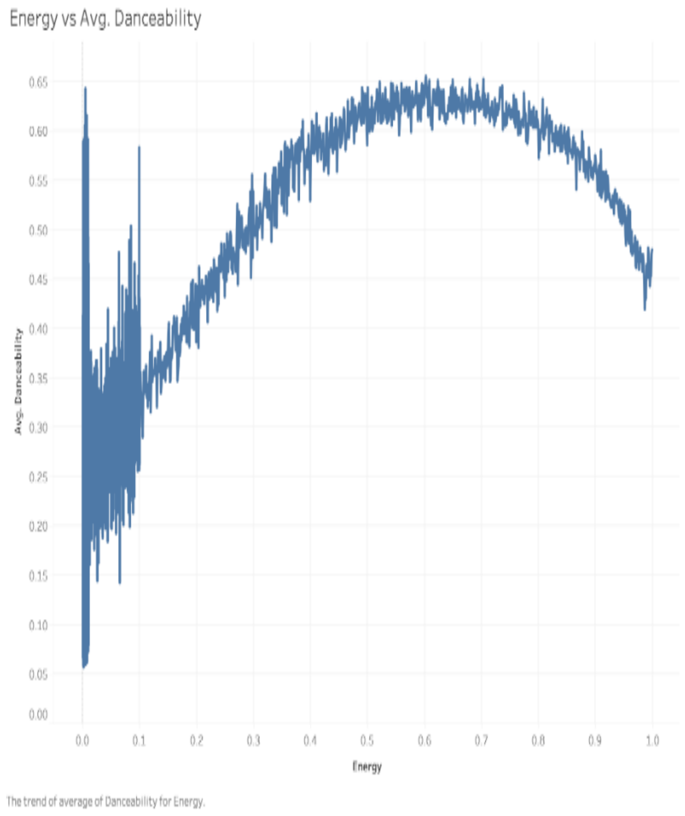


Figure 5

5.4 Song Recommendation

We implemented a song recommendation system using collaborative filtering. Collaborative filtering is a technique used in recommendation systems that relies on past user behavior, preferences, and activities to predict future user preferences. We used the Million Song Dataset, which is a freely available collection of audio features and metadata for a million contemporary popular music tracks.

First we loaded the dataset and prepares it for use. It then calculates the user-item matrix, which represents the user preferences for each song. The matrix is then split into a training and a test set. Then we build the collaborative filtering model using matrix factorization, which is a technique for reducing the dimensionality of the user-item matrix. The model is trained on the training set and evaluated on the test set using the mean absolute error metric.

After building the model, the users are allowed to input a song and get a list of recommended songs based on the user-item matrix and the collaborative filtering model. We first retrieve the song features from the Million Song Dataset and use them to predict the user preferences for the input song. The model is then used to predict the preferences for all other songs in the dataset, and a list of recommended songs is returned based on their predicted preferences.

5.5 Songs Audio Features Comparison

We then implement of the audio features comparison of different songs. Different libraries such as Pandas, Matplotlib, and Seaborn, were used to extract and visualize the audio features of these songs. The audio features used for comparison include danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, and tempo.

Firstly, we loaded the audio files of the three songs and extracts the audio features using the Librosa library. These features are then stored in a Pandas dataframe for easier analysis. The code then visualizes the distribution of each audio feature for the three songs using Matplotlib and Seaborn. The visualization helps to identify the differences and similarities between the songs in terms of their audio features.

Then we computed the pairwise correlation between the audio features for each song using Pandas. The correlation matrix is then visualized using a heatmap to determine how the audio features are related to each other within a song. This analysis provides insights into which audio features have the most impact on the overall

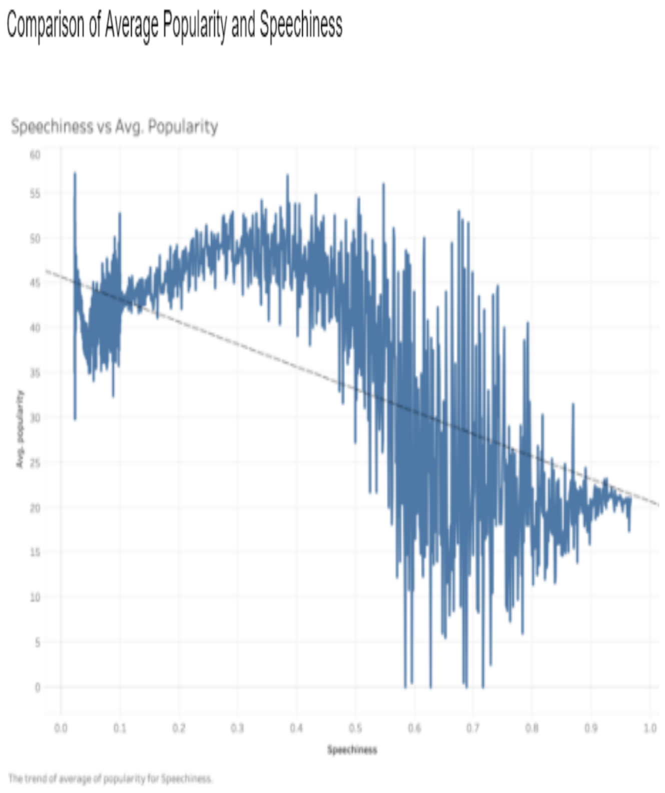


Figure 6

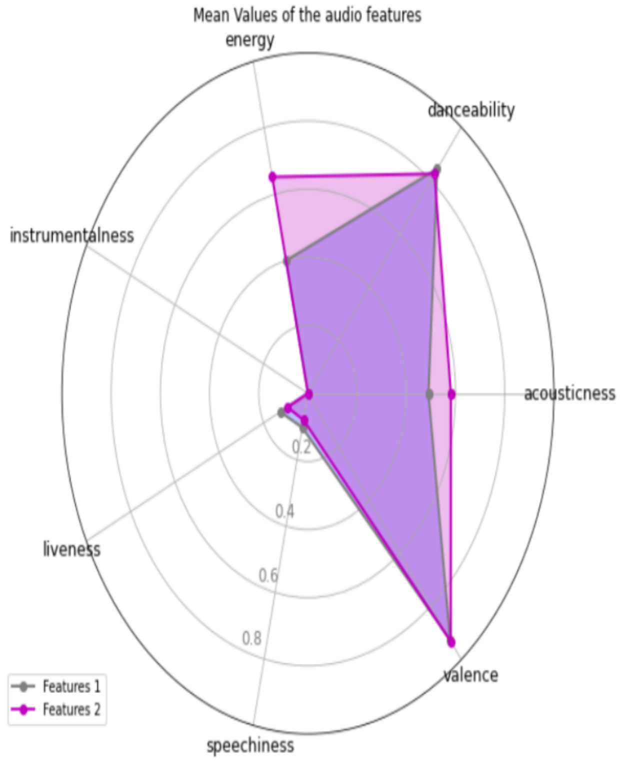


Figure 7

sound of a song.

5.6 Polar Plots

The main functionality of this implementation is to showcase the audio features of a particular track and visualize them using a polar plot. The audio features are extracted using the Spotify API, which provides several audio features of a song, such as danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, and tempo.

The polar plot is a graphical representation that shows the audio features of a track in a circular plot. The circle represents the maximum value of each audio feature, and the radial lines represent the value of each audio feature. The audio features are plotted in a clockwise direction, starting from the top, with the danceability feature at the top, followed by energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, and tempo. The polar plot makes it easy to compare the audio features of different tracks.

5.7 Jammin' App

We implemented a music search and recommendation application using the Spotify Web API. First, we exported the necessary libraries such as Streamlit for building the app, Pandas for data manipulation, and Spotipy, which is a lightweight Python library for the Spotify Web API.

The Spotify API requires an authentication token to access its services, we used a Spotify client ID and secret key to generate an authentication manager that handles this. The app interface includes a search box that allows users to search for tracks, artists, or albums using keywords.

Once the search parameters are entered, the app makes a request to the Spotify API and retrieves a list of search results that match the keyword query. The app then presents the user with a selection box that allows them to choose a specific track, artist, or album from the search results.

For a selected track, the app retrieves information such as song features (e.g., acousticness, danceability, energy, instrumentalness, liveness, speechiness, and valence) and generates a polar plot of these features.

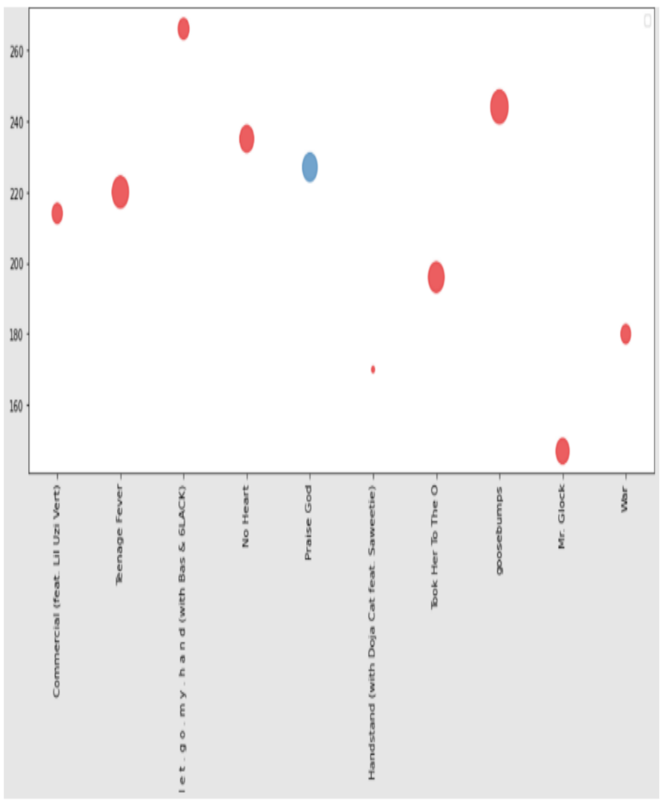


Figure 8

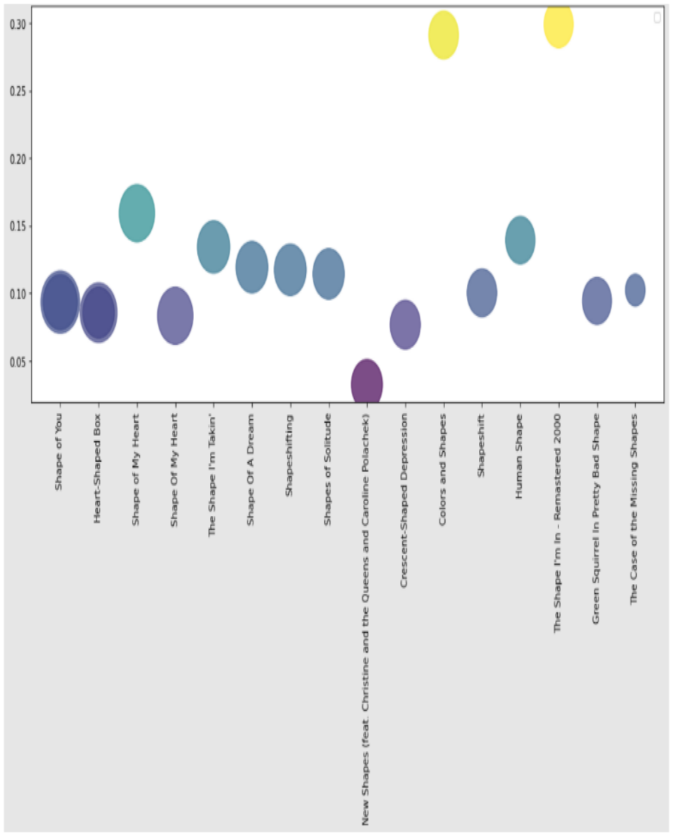


Figure 9

5.8 Spotify Song Attributes

- Danceability: How suitable a track is for dancing based on a combination of musical elements (tempo, rhythm stability, beat strength). A value of 0.0 is least danceable and 1.0 is most danceable.
- Energy: Represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. (Scale 0-1)
- Speechiness: Detects the presence of spoken words in a track. A track with a score closer to 1.0 would be exclusively speech-like, such as an audio book or talk show. A score in the range 0.33-0.66 consists of tracks that contain both music and speech. This would be, for example, rap music. Values below 0.33 represent music and other non-speech-like tracks.
- Acousticness: Confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
- Popularity: Song Popularity (0-100) where higher is better
- Explicit: boolean, does the track have explicit lyrics? (true = yes it does; false = no it does not OR unknown).

5.9 Sentiment Analysis Prediction

One goal was to run a sentiment analysis model to obtain emotions of songs based on song lyrics. The model used was the "cardiffnlp/twitter-roberta-base-emotion" emotion analysis model from HuggingFace. This model was chosen for its ability to output values for different levels of emotion based on the text. The emotions this model outputs are joy, anger, optimism, and sadness.

The Genius API was used to grab the lyrics for the previously scraped Hot 100 songs from April 2020 to present. This dataset was scraped and chosed for two reasons. One. the more popular songs were more likely to have lyrics. Second, these songs already had the values for the different Spotfiy song attributes grabbed from the Spotify API and stored in MongoDB. When obtaining lyrics, only songs with lyrics available in english were chosen.

After obtaining the sentiment values for the dataset, three different machine learning models were chosen: Random Forests, XG-Boost, and neural networks in Keras. These were chosen because of their different complexities and overall different approach. For the sentiment analysis prediction, the data was split into train and test at an 80/20 split.

5.10 Deployment onto Render Cloud

The StreamLit app was deployed on Render to provide it with an free, accessible platform. Render was chosen specifically because

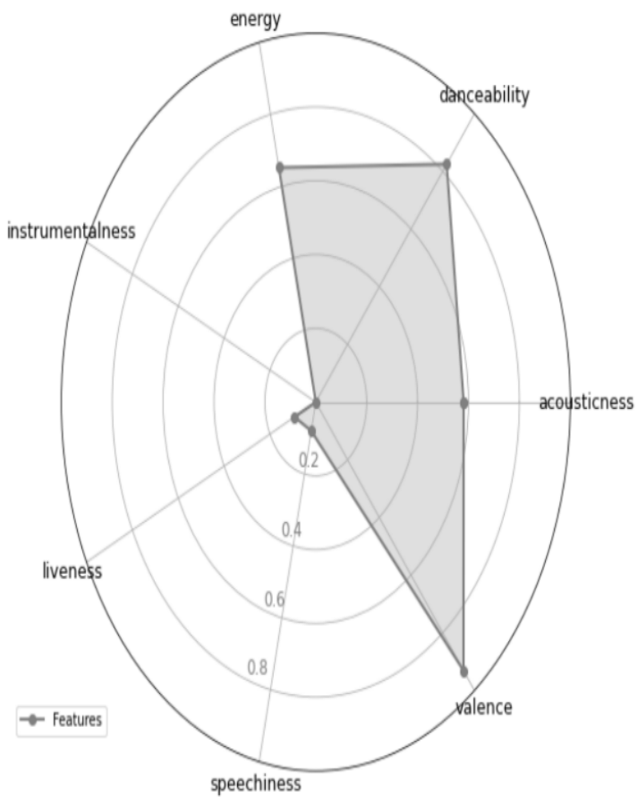


Figure 10

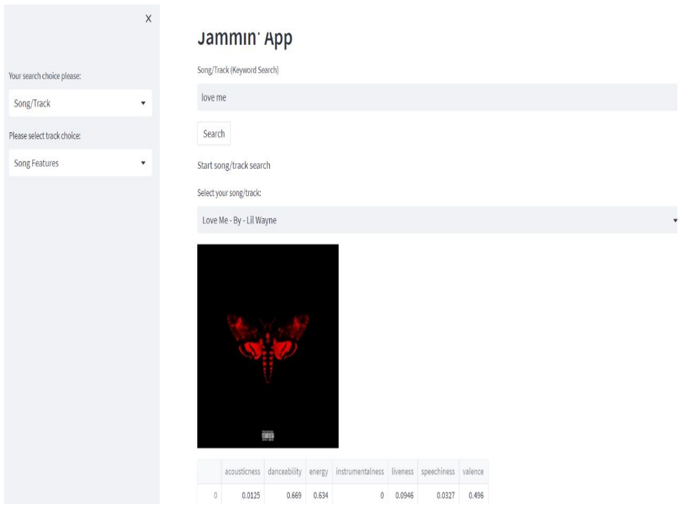
it was free and required little work other than to create and account and create a Dockerfile in the root directory so the app is containerized. In Render, the latest commit is run and deployed on the cloud. There are certain features locked behind payment, such as scalability and jobs. The app is run on the domain: https://bda-project-cloud-6502.onrender.com/ when it is deployed. Because we are on the free version, we do not deploy it when not being used.

6 EVALUATION

Reading multiple CSV files and concatenates them into a single dataframe. Then, the target variable is separated from the input variables, and the input variables are standardized using StandardScaler. Then, several machine learning models are trained and evaluated using cross-validation techniques. Finally, the confusion matrices, accuracy, precision, recall, and F1 scores are calculated for each model.

6.1 Sentiment Analysis Results

The three metrics used for the model metrics were r-squared, Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC). R-squared was chosen to see how the accurate the regressions were. AIC and BIC were chosen because we have a smaller dataset, and both metrics penalize overfitting on a smaller dataset. The AIC and BIC values themselves are arbitrary, we just use them to compare the models themselves relatively, with lower values representing a better fit.



Your search choice please: Song/Track

Please select track choice: Song Features

Song/Track (Keyword Search): love me

Start song/track search

Select your song/track: Love Me - By - Lil Wayne

	acousticness	danceability	energy	instrumentalness	liveness	speechiness	valence
0	0.0125	0.669	0.634	0	0.0946	0.0327	0.496

Figure 11

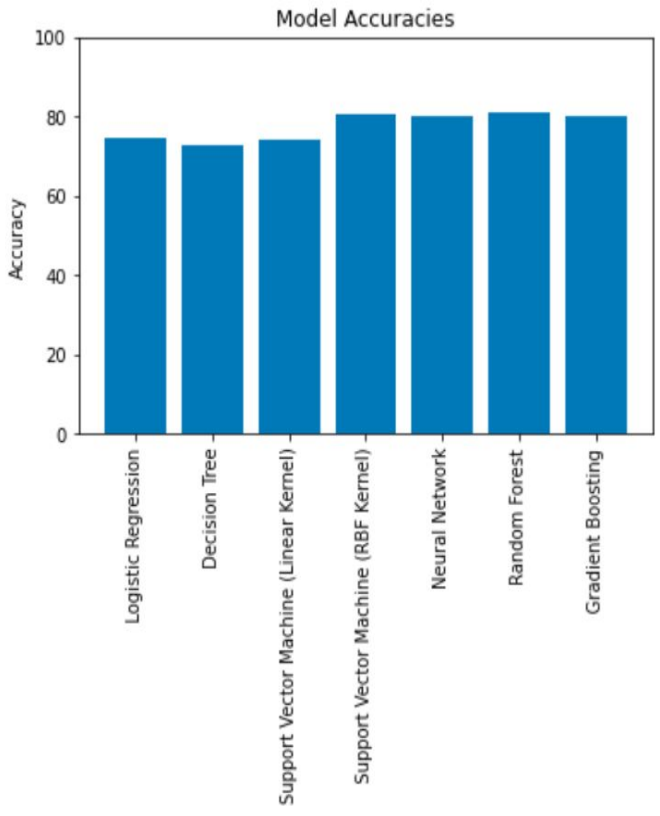
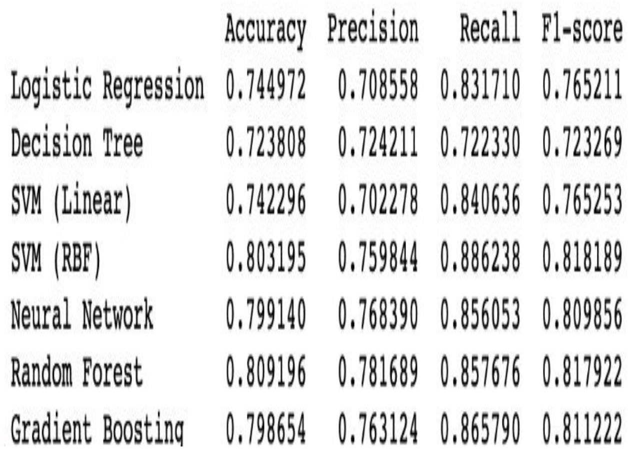


Figure 12



	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.744972	0.708558	0.831710	0.765211
Decision Tree	0.723808	0.724211	0.722330	0.723269
SVM (Linear)	0.742296	0.702278	0.840636	0.765253
SVM (RBF)	0.803195	0.759844	0.886238	0.818189
Neural Network	0.799140	0.768390	0.856053	0.809856
Random Forest	0.809196	0.781689	0.857676	0.817922
Gradient Boosting	0.798654	0.763124	0.865790	0.811222

Figure 13

Table 1: ML Model Metrics

Models/Metrics	Random Forest	XGBoost	Neural Network
R-squared	-0.003517	-0.163394	-0.386671
AIC joy	3.698	3.463	3.454
AIC optimism	2.438	2.243	1.781
AIC anger	5.792	5.489	4.488
AIC sadness	3.900	3.456	3.670
BIC joy	23.087	22.852	22.843
BIC optimism	21.826	21.632	21.169
BIC anger	25.180	24.877	23.876
BIC sadness	23.288	22.845	23.059

As we can see, none of the models performed particularly well in this problem. And there are a few reasons for that, namely a smaller dataset.

6.2 ML Model Scalability

The model scalability was tested for inference time versus number of samples. A scalable model should have an inference time that grows linearly or sublinearly with the dataset size.

Table 2: ML Model Scalability Inference Time (s)

Models/n	Random Forest	XGBoost	Neural Network
1000	0.08	0.01	0.13
10000	0.48	0.02	0.30
100000	3.48	0.17	2.68
1000000	31.63	1.70	26.54
10000000	336.05	16.00	243.24

All three models scale linearly, but in terms of making predictions, XGBoost is by far the fastest, and would be best suited for a big data setting.

6.3 App Scalability

Lastly, the scalability for the app was also tested regarding response time and throughput. Apache Jmeter was used to set up HTTP responses and thread count. It also produced a summary report with the following values. Apache JMeter ran three separate times. Once for 100 users with a ramp up period of 10 seconds, once for 1000 users with a ramp up period of 10 seconds, and lastly an infinite loop of 100 users with a ramp up period of 10 seconds to see the overall pattern.

Table 3: App Scalability Response Time (ms)

Samples	Average	Error Percentage	Throughput
100	164	0	9.95619
1000	1463	0	78.59781
156287	1164	0.065	85.61893

Overall, the app at its with a constant inflow of concurrent users has a throughput of 85.619 requests per ms. The throughput can indicate how efficiently the system is at processing and analyzing large volumes of data. The average response time with the high load is 1164 ms. This isn't ideal, but considering the high load is not too bad. Overall, the error rate is minimal, which shows the scalability of the app, even on the free version of render.

7 DISCUSSION

The app does a good job in providing a platform that lets us look at the Spotify track features. It provides polar plots of the song features, something not available on normal Spotify.

The modeling and analysis had its negatives. There were many possibilities why sentiment analysis prediction may have been poor, such as not a lot of data with only 2000 entries, not a lot of song features available on Spotify, no strong correlations between Spotify song features and emotion analysis results, and sentiment analysis not being perfect as a whole. For the latter, it is hard to account for tone and key with just these features, and some songs may have less obvious meanings than what the emotion analysis makes of it. For example, the song "Glimpse of Us" by Joji is a popular song that is notably sad. It has an anger value of 0.233465, joy value of 0.0521617, optimism value of 0.4967704, and sadness value of 0.2176025. This means the emotion analysis reads the song lyrics as having optimism. I think this is something tough to pick up on with lyrical analysis alone because the song is about not having the same spark in love as previously, and the lyrics that say "hoping" may mislead the model. Also in the context of the models, more computationally expensive does not necessarily mean more accurate.

The inference speeds can be justified. The XGBoost model is computationally efficient due to its optimized implementation. It is designed for parallel processing wheras the others were not using the same built in parallelization. In addition, the model size for XGBoost is typically smaller compared to the others. Lastly, the hardware was not using a GPU and was not GPU optimized in environment.

8 CONCLUSION

In conclusion, the project successfully implemented a recommendation algorithm based on audio features of songs. Further work can be done to improve the algorithm by incorporating user reactions through sentiment analysis. Additionally, allowing users to save playlists within the app and making it mobile-friendly can improve user experience and accessibility.

Another potential improvement is implementing clustering for the recommendation system. However, this would require extensive frontend and user survey work, making it harder to evaluate the success of the system without going too far out of scope. Overall, the project lays the groundwork for future development and improvements in the field of music recommendation systems.

ACKNOWLEDGMENTS

The authors thank Professor Qin Lv and the Teaching Assistants, Cara Spencer and Rahul Aedula for their valuable feedback.

REFERENCES

[1] Jesus Soto Fernando Terroso-Saenz and Andres Muñoz. 2023. Evolution of global music trends: An exploratory and predictive approach based on Spotify data. https://www.sciencedirect.com/science/article/pii/S1875952122000593. *Entertainment Computing* (2023).
[2] Andrea Gao. 2021. Catching the Earworm: Understanding Streaming Music Popularity Using Machine Learning Models. https://www.researchgate.net/publication/351387151_Catching_the_Earworm_Understanding_Streaming_Music_Popularity_Using_Machine_Learning_Models/. *E3S Web of Conferences* (2021).
[3] Genius. 2021. Genius API Documentation. *Genius* (2021). https://docs.genius.com/
[4] Novia Ayu Privandhani and Sulastri. 2022. Clustering Pop Songs Based On Spotify Data Using K-Means And K-Medoids Algorithm. https://www.iocscience.org/ejournal/index.php/mantik/article/view/2517. *Jurnal Mantik* (2022).
[5] Mariangela Sciandraa and Irene Carola Sperab. 2022. A model-based approach to Spotify data analysis: a Beta GLMM. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9042099/. *Journal of Applied Statistics* (2022).