Song Recommendation Algorithm

Anthony Pagas Jack Sullivan Dylan Mahant Seth Ray Josh Richards

Introduction

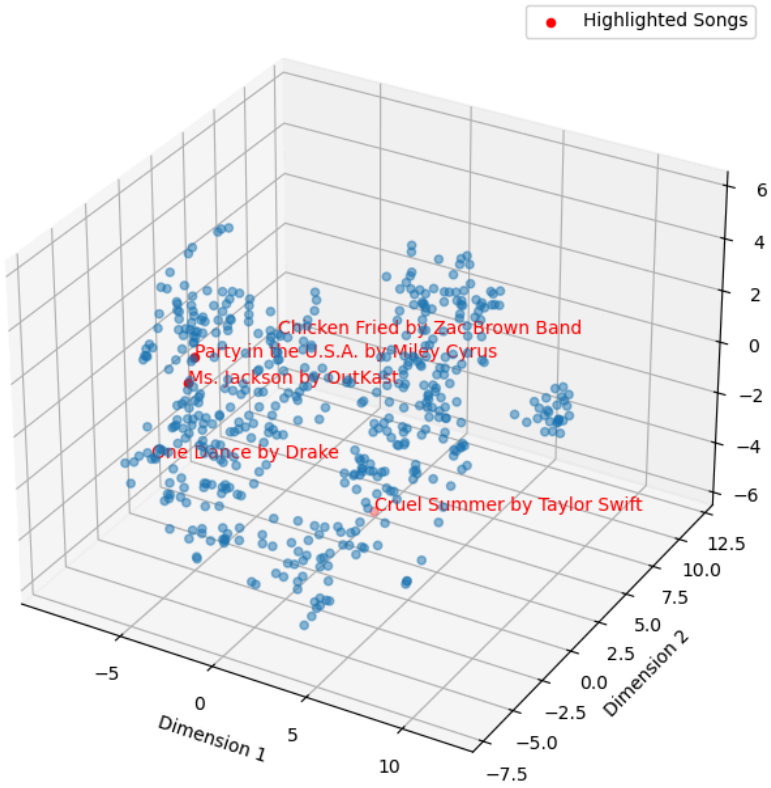
In our project, our group analyzed the Spotify 1 million tracks data from kaggle. We wanted to test if we could produce an algorithm that outputs 10 potential songs that a user would like based on 10 songs inputted into the algorithm. In order to do so, we utilized the process of vectorizing song features and comparing similarity. Although results are subjective, we concluded that our algorithm performed well based on personal feedback of our group. However, further improvements could be explored.

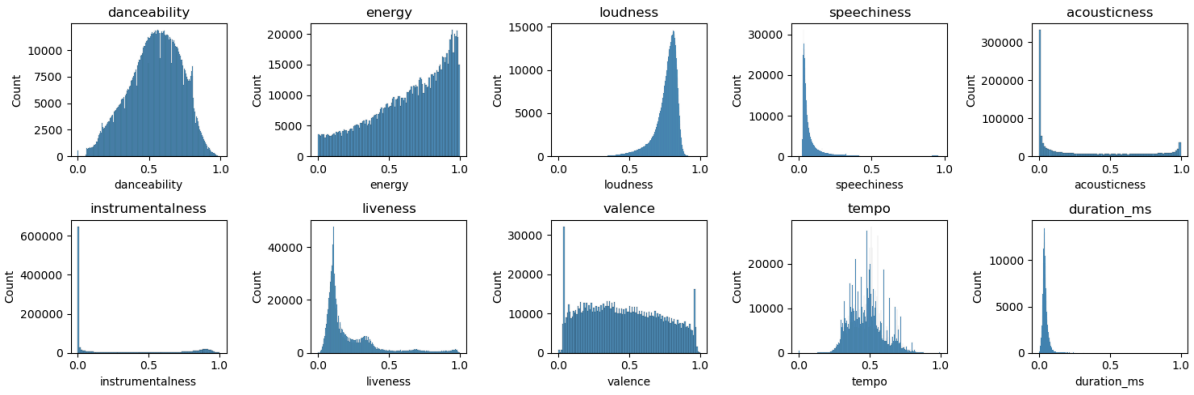
Data

The source of data for our project was the 1 million song Spotify dataset from Kaggle [1]. The dataset contained artists, track names, genres, and a number of numeric features describing the songs. We did a number of preprocessing steps such as binning genres, one hot encoding genres, min-max scaling features and excluding any ambient sleep music as it wasn't relevant for our project. Below you can see an example of our final processed dataframe.

	master_idx	artist_name	track_name	track_id	year	genre	popularity	danceability	energy	key	...	Instrumental	Jazz and Blues	Latin	Misc
0	0	Jason Mraz	I Won't Give Up	53QF56cjZA9RTuuMZDrSA6	2012	Acoustic	0.68	0.483	0.303	4	...	0	0	0	
1	1	Jason Mraz	93 Million Miles	1s8tP3jP4GZcyHDsjvw218	2012	Acoustic	0.50	0.572	0.454	3	...	0	0	0	
2	2	Joshua Hyslop	Do Not Let Me Go	7BRCa8MPiyuvr2VU3O9W0F	2012	Acoustic	0.57	0.409	0.234	3	...	0	0	0	
3	3	Boyce Avenue	Fast Car	63wsZUhUZLIh1OsyrZq7sz	2012	Acoustic	0.58	0.392	0.251	10	...	0	0	0	
4	4	Andrew Belle	Sky's Still Blue	6nXIYClvJAfi6ujLiKqEq8	2012	Acoustic	0.54	0.430	0.791	6	...	0	0	0	

In exploring our data, we wanted to formulate models that would give us a good idea of the layout of the data we are working with and how to generalize patterns of songs. Therefore, we constructed a TSNE dimensionality reduction 3D model with highlighted songs. Secondarily, we also plotted the distributions of each numeric feature.

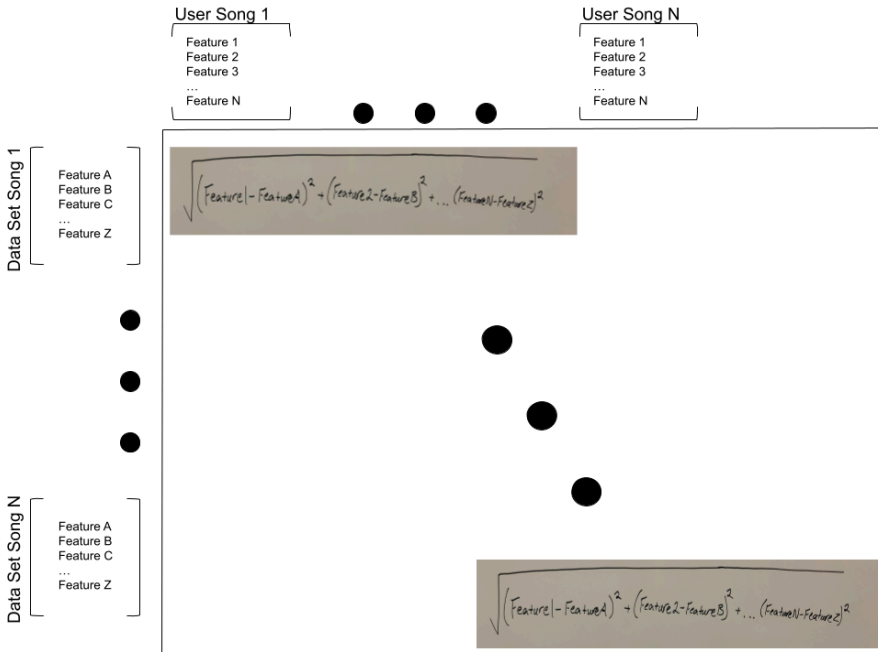




Machine Learning Approaches

We each decided to attempt to answer the question in our own way, and reconvene with finished algorithms. Below is a summary of our methods;

Method	**Description**
"Traditional Sklearn"	We used more conventional "out of the box" Sklearn algorithms to 1. Train a K-Means and HAC model to assign clusters. 2. Accept user input songs. 3. Only keep clusters of user songs. 4. Gaussian mixture to determine which songs are most similar to the inputted songs 5. Report the most similar songs
"TFIDF Text Analysis"	In this method we attempted to use an open source TFIDF text analysis algorithm to predict songs based on the similarity of lyrics.
"Vectorization"	In this approach we treated all numeric features of songs as vectors, and found song recommendations based on nearest euclidean distance pairs of user input songs. This method tended to perform the best in our evaluation trials conducted among group members and friends. Below is an illustration of the matrix used to compute distances as well as our evaluation of the model.



Seth	8/10 Liked (Cross genre input)
Jack	7/10 Liked (Specific genre input)
Dylan	5/10 (Cross Genre Input)
Anthony	7/10 (Cross genre input)

An aspect of the solution we were particularly proud of developing was our dynamic feature importance code. We figured, because people like songs for a variety of reasons, it would be hard to generalize a universal set of features to evaluate distances on. As a solution, we ran a Monte Carlo simulation to find statistically significant values of variance for each feature that we treated as thresholds. If the user input variance for any given feature was lower than the corresponding threshold, that feature was deemed important and factored into distance calculations for song prediction.

Hyperparameter Selection

The only methods that required hyperparameter analysis was the traditional SKlearn model as it regarded the K Means and HAC clustering algorithms. We spent some time choosing the number of clusters for this by investigating values that optimized metrics like inertia of clusters, and briefly looking into silhouette scores (we choose 15 clusters). However this process was less than exhaustive because we had already determined the vectorization method was more fruitful, so we spent considerably more time on that.

Weaknesses

We found one of the greatest weaknesses of our algorithm was the sheer volume of data we were using. We believe with a datasource that included something like a language column to partition songs more tailored to the user, we could have yielded even better results.

Conclusion

To recap, we wanted to test if we can produce 10 potential songs a user likes based on 10 songs a user inputs into our algorithm. Despite facing challenges with unstructured data that was at times too broad, we were still able to produce insightful visualizations, a variety of methods to address our driving question, and most importantly an algorithm that produces songs that are objectively similar to user inputs (Although the human subjectivity of enjoyment makes evaluation difficult). We are content with the results, but the algorithm could always be improved by methods like incorporating feedback based learning: where a user can grade the algorithm outputs, and begin to build more of a labeled data problem by providing their own labels to the songs.

Member	Proposal	Coding	Presentation	Report
Anthony Pagas	0	0.5	1	1
Jack Sullivan	1	1	0.5	0.75
Seth Ray	1	0.8	0.75	0.5
Josh Richards	0	0.75	0.5	0.5

1. https://www.kaggle.com/datasets/amitanshjoshi/spotify-1million-tracks?select=spotify_data.csv