Song Recommendation Algorithms

Using k-Means, KNN, and the Spotify Echo Nest

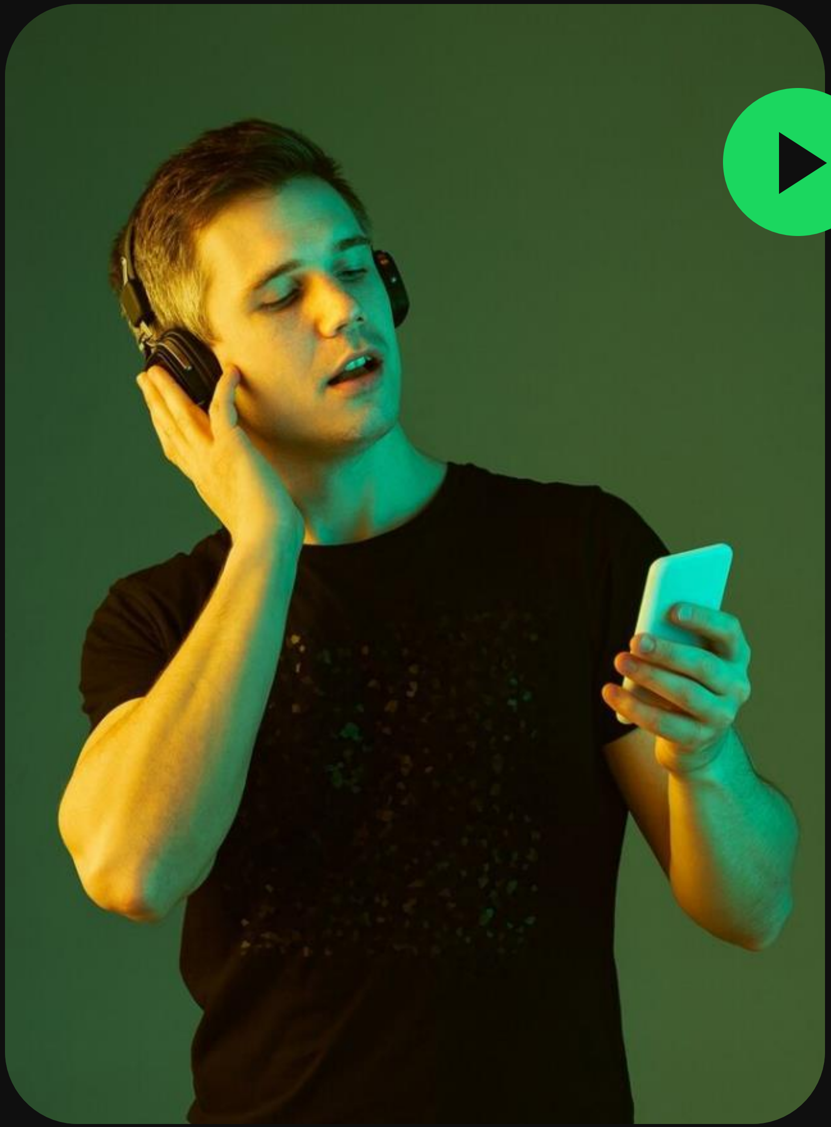
Table of Contents

Project Goal

Our goal was to build a music recommendation model using Spotify's Echo Nest audio features and Predictive Modeling. More specifically, we set out to build a model that would "predict" songs a user might like, given a song they are known to enjoy

Play

Shuffle



Our Dataset

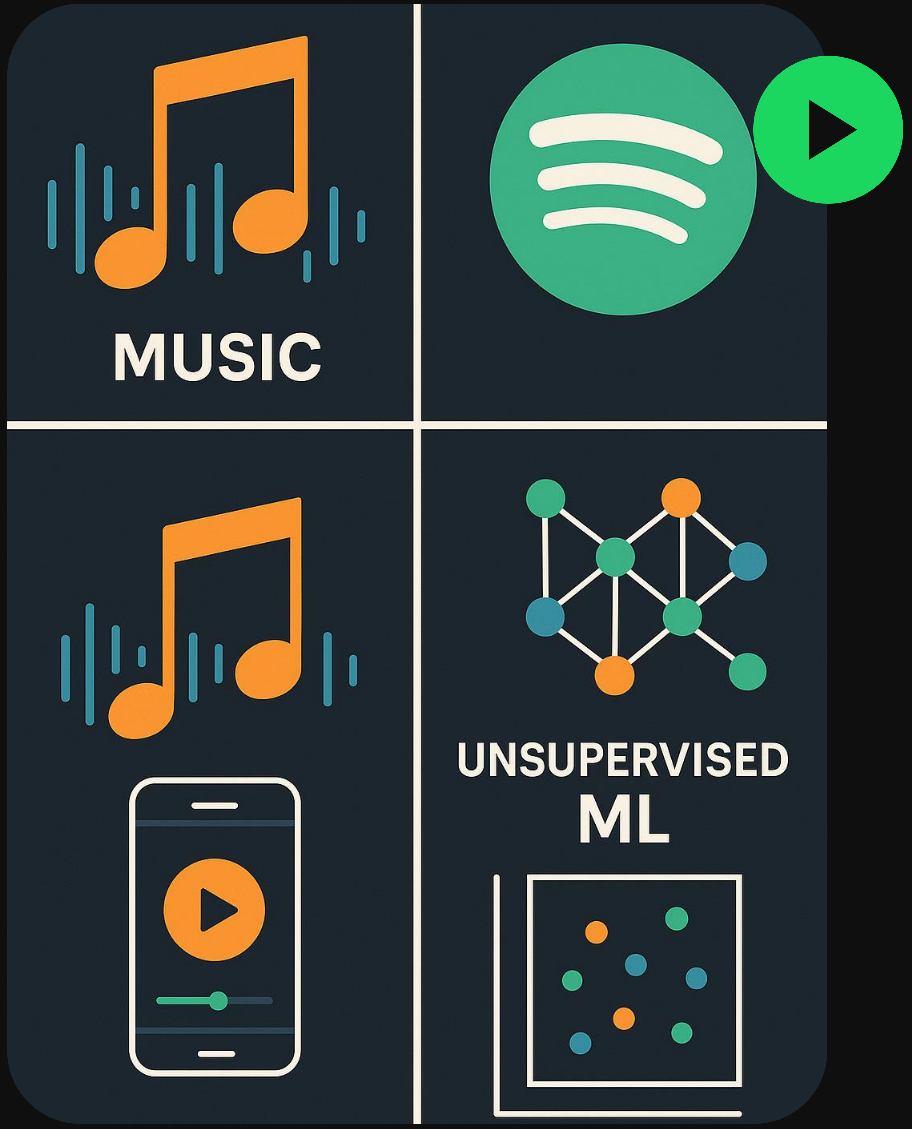
#		Echo Nest Variables		
1		Danceability	Double/Numeric	
2		Energy	Double/Numeric	
3		Speechiness	Double/Numeric	
4		Acousticness	Double/Numeric	
5		Instrumentalness	Double/Numeric	
6		Liveness	Double/Numeric	
7		Valence	Double/Numeric	
8 - 12		Key, Loudness, Mode, Tempo	Double/Numeric	
13 - 14		Popularity (50 – 100), Duration (milliseconds)	Double/Numeric	

Running Predictive Models With No Outcome Variables

Our dataset does not have an outcome variable, there isn't anything we are trying to predict other than “Will the user like the songs our algorithm suggests” based on a song given that the user enjoys.

Unsupervised Learning

In class we primarily learned how to run supervised models -- but K-Means and KNN can also be used to “predict” (cluster) groups or relationships within a dataset



Sometimes, Simple Models == Simple Solutions

Initially, our plan was straightforward: use K-Nearest Neighbors (KNN) to identify songs statistically similar based on Spotify's Echo Nest features. We were fortunate that our variables showed no significant correlations (none exceeded a correlation of 0.8), meaning minimal preprocessing was needed. This allowed us to rapidly prototype a simple baseline model -- just three lines of code -- to quickly gauge effectiveness.

So, What now?

Our initial, simple KNN model performed surprisingly well, meeting expectations despite its simplicity. Curious if complexity would yield better results, we incrementally tested more advanced methods available to us (excluding algorithms outside of our class toolkit like neural networks, DBSCAN, or topic modeling).

Model 1

K – Nearest Neighbors (kNN)

Model 2

K-Means portioned Data, kNN

Model 3

K-Means portioned Data, Principal Component Analysis (PCA), kNN

PLAY

FOLLOWING

We built the Models, so do they work?

Due to the unsupervised nature of our project, traditional statistical evaluation methods were not applicable. Instead, we conducted a manual evaluation based on subjective user feedback. Each team member selected 10 songs they personally enjoyed, and each model generated 5 recommended tracks per input song, resulting in 50 recommendations per model for annotation.

During evaluation, each team member manually annotated the recommendations, simply marking songs as either liked (1) or disliked (0). Importantly, we intentionally provided no strict criteria beyond personal preference. Even if a recommended track was statistically or acoustically similar to the input song, annotators marked it negatively if they did not personally enjoy it.

This subjective approach aligned with our original project goal: building a recommendation model focused on user enjoyment rather than purely statistical or acoustic similarity.

Evaluation Results

Model 1 — **Simple, Efficient**
Average 59% Accuracy

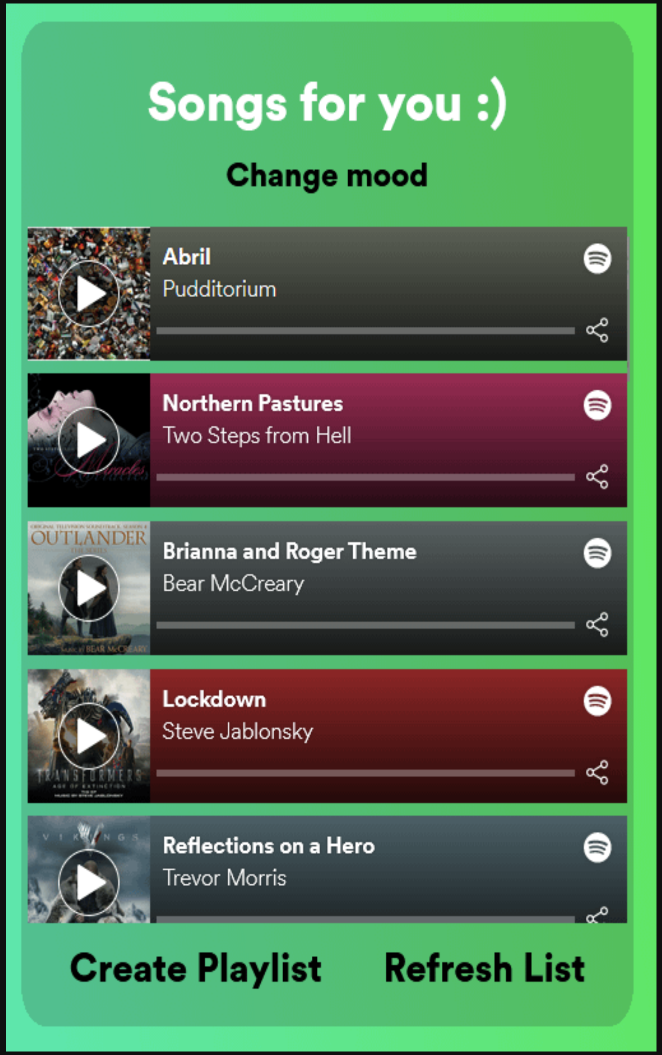
Model 2 — **The Fan Favorite**
Average 61.5% Accuracy

Model 3 — **The Let Down**
Average 59% Accuracy

Final Approach

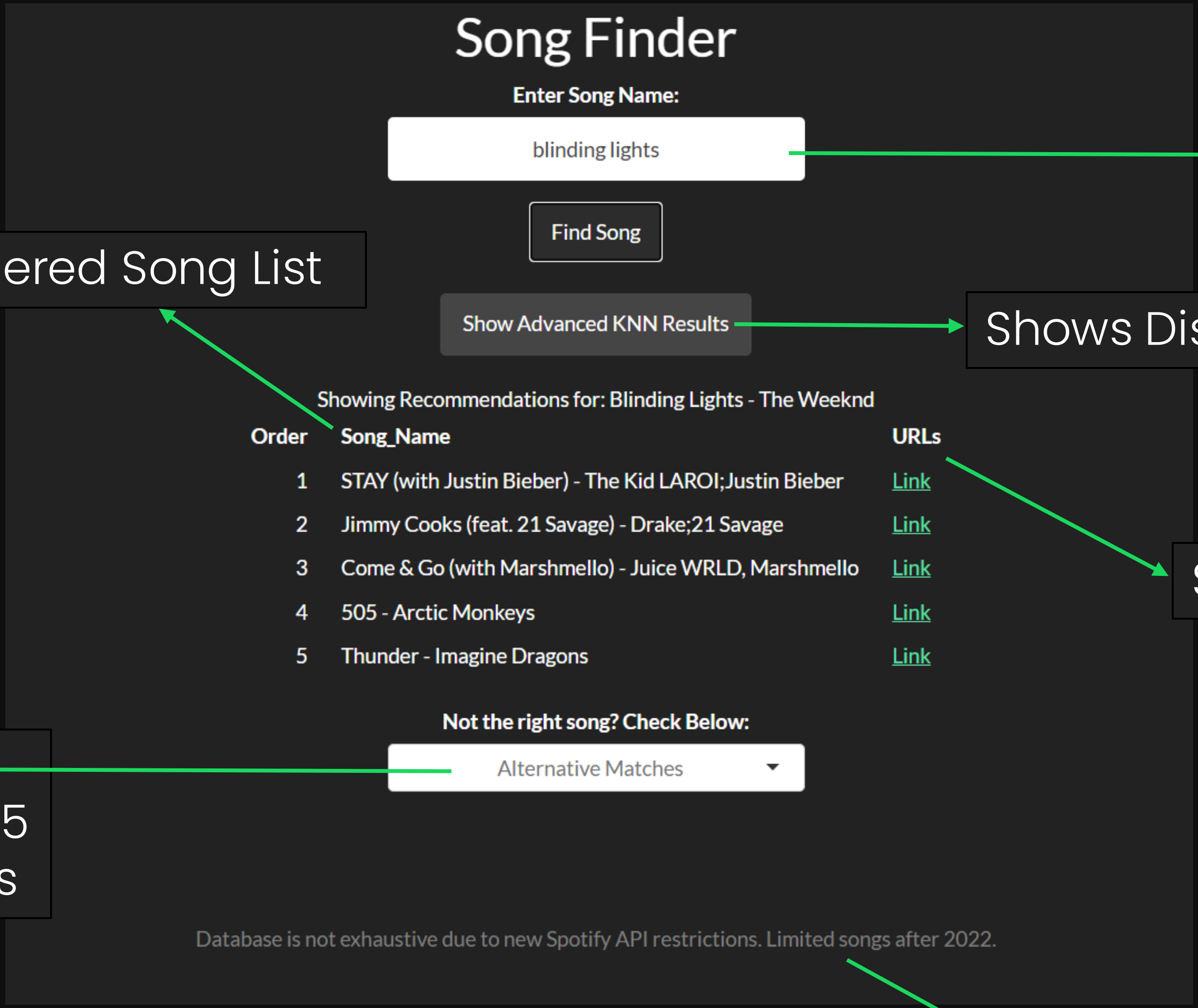
The final approach we took towards the development of our app was to utilize our best performing model in terms of annotator satisfaction. This ended up being Model #2.

Surprisingly, the simplest (Model 1) and most complex model (Model 3) graded out equally to our annotators, while the model in-between them (Model 2) performed slightly better. Despite Model 1 and 3 reporting the same accuracy percentage, it is important to note that Model 3 reported 3 out of 4 annotators worst playlists.



https://daniel-tafmizi.shinyapps.io/final_proj_shinyapp/

ShinyApp.io Interactive Song Finder



User's Song Input

Ordered Song List

Shows Distance Metric

Spotify Link

Dropdown With the Next 5 Fuzzy Matches

Sad Disclaimer :(

Limitations

Sometimes, songs were flagged as similar because they shared statistical traits, like tempo or energy, but ended up sounding completely different or clashing thematically.

How can we fix this

Additional metadata -- like genre tags, lyrics, or mood descriptors – could help capture what a song is about, not just how it sounds.

Also: a working Spotify API would have helped.

≠

Noah Kahan

Gobsmack

PLAY FOLLOWING

Conclusions

Music is deeply subjective, and no algorithm can fully capture the personal or cultural context behind a listener's preferences. While our model uses audio features like tempo and energy, it lacks metadata such as lyrics, artist reputation, or cultural relevance.

Some factors are simply unpredictable—an artist may be cancelled overnight or go viral unexpectedly. A song might be statistically perfect based on a user's favorites and still be a poor fit due to vibe alone. These are limitations we can't address through audio data alone, or even publishing and contextual data.

Handling those complexities would require broader models involving trend analysis or artist risk assessment, which is, unfortunately, far beyond our current project scope, but an important reminder that data-driven recommendations have natural limits.

The Team

Alina Hagen
Model Design / Evaluation

Daniel Tafmizi
Model / App Design

Alexander Crowell
Model Evaluation / Visualization

Anthony Venson
Annotation / Debugging

Thanks!

For more a more detailed project description, visit our GitHub Repository

ATTRIBUTION

This presentation template was created by SlideKit

Please keep this slide for attribution.