

Song Recommendation Engine

Egert Heliste
Kristjan Sild
Group D7

Github repo: <https://github.com/EgertHel/SpotifyRecommended>

Task 2. Business understanding

1. Identifying business goals

There are over 100 million different songs available on the Spotify platform, which can feel overwhelming and it becomes difficult to find new songs to listen to that match your taste. When searching for a song, most of the time you will be recommended only the most recent and popular songs, which means that there are many songs that are left undiscovered. Nearly 80% of the artists on Spotify have less than 50 monthly listeners.

Spotify assigns many attributes for every song, for example energy, danceability, liveness, etc. These attributes can be gathered with the Spotify API but for our project it is enough to use an existing dataset from Kaggle with 600 000 different songs. Our goal is to create a song recommendation algorithm based on these attributes. We aim to create an algorithm that is able to give more accurate results because it prioritises song attributes over popularity. This benefits both the listeners and artists because you can find songs with high quality that are yet to be discovered and small artists get more exposure.

To consider our project a success, the final algorithm should recommend multiple songs and at least 50% of the recommendations should be songs that the user would rate as enjoyable. There should also be visualisations with each recommendation so that the user can better understand their listening habits and why the algorithm suggested such songs.

2. Assessing our situation

We have access to two Kaggle datasets: one containing over 600,000 tracks from 1921–2020 and another featuring top hits from 2010–2022. These datasets include essential audio features and popularity scores, making them suitable for building a recommendation system. Technical resources include Python and libraries such as pandas, numpy, scikit-learn for modeling and matplotlib/seaborn for visualization.

Constraints include the absence of the latest Spotify song data, limiting recommendations to historical tracks. Risks involve data quality issues such as missing values and outliers, which will be addressed during preprocessing. Terminology like “danceability,” “valence,” and “energy” will be standardized for clarity. The primary cost is time and computational resources, while benefits include actionable insights and a prototype recommendation system that could enhance playlist curation and user engagement.

3. Defining data-mining goals

Our data-mining objectives are to clean and merge the datasets, explore feature distributions and implement a KNN algorithm for song similarity. We will also produce visualizations to illustrate trends and correlations among audio features. Success will be defined by the ability of the KNN model to return meaningful recommendations and by the quality of visualizations that provide clear insights into the dataset.

Task 3. Data understanding

1. Gathering Data

Since we got our two datasets from Kaggle, we didn't need to gather anything. We combined the Kaggle datasets: Spotify Dataset 1921–2020 (600k tracks), Spotify Top Hit Playlist 2010–2023 (50k tracks). Our requirements included audio features, metadata and popularity metrics. Both datasets met these needs.

Selection Criteria: From these both datasets we only use the songs/tracks dataset, there is also data about artists, but the metrics there give no value to us. In the songs data we excluded unnecessary columns like playlist url, album, artist popularity, and artist genres. We removed duplicates, and kept tracks with a valid song name. We extracted release_year

from release_date due to some songs only having a release year and us not needing an exact release date.

2. Describing data

The merged dataset was created by combining two Kaggle datasets in CSV format: Spotify Dataset 1921–2020 (600k tracks) and Spotify Top Hit Playlist 2010–2022 (50k tracks). After cleaning, it contains approximately 650,000 rows and 18 columns.

Key fields include:

- song_id, song_name and artists,
- duration (milliseconds),
- explicit (binary),
- song_popularity (0–100),
- audio features such as danceability (suitability for dancing), energy (intensity), key, loudness (dB), mode, speechiness, acousticness, instrumentalness, liveness, valence (positivity), tempo (BPM), and time_signature.

The dataset includes all required fields for analyzing song characteristics and popularity trends. Audio features are normalized between 0 and 1, and popularity scores range from 0 to 100.

3. Exploring Data

To better understand the dataset, we examined the distribution and summary statistics of key variables. The merged dataset contains 588,903 tracks after further cleaning. Below are the main observations:

- Popularity: Ranges from 0 to 100, with a mean of 27.7 and median of 27, indicating most tracks have low to moderate popularity.
- Duration: Average duration is 230,033 ms (~3.8 minutes), with a few extreme outliers up to 5.6 million ms (over 90 minutes), suggesting possible anomalies.
- Explicit: Only 4.4% of tracks are marked explicit, showing most songs are clean.
- Danceability & Energy: Danceability averages 0.56 and energy 0.54, both normalized between 0 and 1. Most tracks fall between 0.45–0.68 for danceability and 0.34–0.75 for energy, indicating a balanced mix of calm and energetic songs.
- Tempo: Mean tempo is 118 BPM, with most tracks between 95–136 BPM. A few outliers exist above 240 BPM.

- Loudness: Average loudness is -10.18 dB which is typical for music tracks, but some values reach -60 dB or +5 dB.
- Release Year: Median year is 1992, with tracks spanning 1900–2021.
- Other Audio Features:
 - *Speechiness* is low (mean 0.10), suggesting most tracks are music rather than spoken word.
 - *Acousticness* varies widely (mean 0.44, max near 1.0), showing diversity in acoustic vs. electronic tracks.
 - *Instrumentalness* is near zero for most tracks, confirming that vocals dominate.
 - *Valence* averages 0.55, indicating generally positive or upbeat tracks.

4. Verifying Data Quality

Since we merged slightly different datasets we had some nan values after the merge which were a non issue. The dataset includes all required audio features and popularity metrics, making it suitable for the recommendation algorithm. Around 1000 songs were missing the song name, which we dropped. Some explicit tags were also missing. The time period was promised to be from 1921 to 2023, but after verifying the data the songs are from 1900 to 2021. No severe quality problems were found, so we can proceed with modeling without changing project goals.

Task 4. Planning the project

Task	Egert	Kristjan	Total time estimate
Cleaning and merging datasets	4h	4h	8h
Verifying data quality	0h	2h	2h
Creating general visualisations based on the entire dataset	0h	4h	4h
Making the KNN model for the recommendation algorithm	2h	0h	2h
Improving the KNN model by adding K-means clustering	4h	2h	6h

Tuning the final recommendation algorithm (hyperparameter tuning, adding weights to features, etc)	5h	3h	8h
Adding a GUI to improve usability	5h	0h	5h
Song similarity visualisations using data from K-means clustering	2h	4h	6h
Creating visualisations based on the user's chosen song and given recommendations	3h	5h	8h
TOTAL:	25h	24h	49h

The first step of the project is cleaning the datasets, merging them and verifying that there are no errors in the data. Both of the datasets contain mostly the same song attributes but the column names differ and some data is in a different format, for example the release date. When the datasets are combined then we can start with general visualisations, for example distributions of different song attributes and using dimensionality reduction techniques like UMAP to better understand the data and relations between different song attributes.

For the song recommendation algorithm, we will start off by making a simple KNN model using the song attributes, duration and popularity. This model should already be able to give some recommendations but to improve the results we will also implement K-means clustering. The idea is to cluster the songs before and then pick all of the songs from a single cluster using the KNN model. This should make the recommendations feel less random for the user because two songs may have similar attributes but be from totally different genres.

After implementing the recommendation model we can add a simple GUI using the Tkinter library to make it easier for the user to use our model. Additionally, we can use the GUI to display visualisations based on the specific recommendations that were made for the user. For example we can visualise all of the clusters and show both the input song and recommended songs on a graph.