Spotify Song Mood Recommendation Report

Yumo Bai, baiym104@gmail.com

July 3, 2022

1. Problem Statement

In this project, we tried to find a way to classify Spotify songs into four different moods and leverage such a classification method to generate balanced recommendations for a Spotify user.

2. Background

Spotify offers a great music streaming experience: not only does it have a visually aesthetic UI, and a thorough collection of music but also it has millions of playlists that are made by the users. However, it does not utilize the emotional aspect very well. Under the search section of Spotify, there is a section where users could browse playlists by Mood, but there are only these playlists with `Happy` tracks are included. With the model that we come up with, we should be able to not only fit other emotions for the Mood section but also theoretically create any new section with multiple subcategories. To utilize such a new function, we also want to see if we can incorporate it into a recommendation system so we can generate recommendations to users based on such new categories.

3. Song Mood Classification

As we mentioned above, the initial motivation of this project is to develop a way to classify music by the emotion it would evoke in the listener. So the very first thing we need to do is to gather tracks and label them with how people felt while listening to them. We gathered the data with Spotify Web API in the following way:

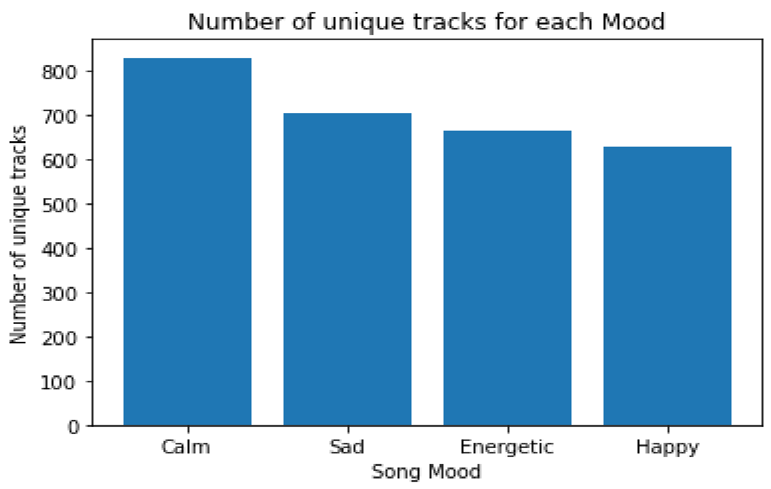
1. Search for playlists with the name of the mood
2. Extract tracks from found playlists and label them with the searched mood's name
3. Combine the tracks and get their audio features using Spotify Web API

We developed a Python script to automate this process. In the end, we gathered 4179 data points with the following features:

track_name: The name of the track.
track_id: The Spotify ID for the track.
artist_name: The name of the artist.
artist_id: The Spotify ID for the artist.
album_name: The name of the album. In the case of an album takedown, the value may be an empty string.
album_id: The Spotify ID for the album.

popularity: The popularity of the track. The value will be between 0 and 100, with 100 being the most popular.
danceability: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity.
energy: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
key: The key the track is in.
loudness: The overall loudness of a track in decibels (dB).
mode: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
speechiness: Speechiness detects the presence of spoken words in a track..
acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
Instrumentalness: Predicts whether a track contains no vocals.
liveness: Detects the presence of an audience in the recording.
valence: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.
tempo: The overall estimated tempo of a track in beats per minute (BPM).
type: The object type. Allowed value:"audio_features"
duration_ms: The track length in milliseconds.
time_signature: An estimated time signature.
mood: The keyword that we searched the playlists with. It is used as the label of the tracks and the possible values are ‘Happy’, ‘Sad’, ‘Energetic’ and ‘Calm’.

During the data cleaning, we removed 1360 duplicated rows on a track_name-artist basis and left with 2819 unique songs. This was necessary since we need to make sure the model will be training on the unique data and there is no leakage between the training and testing set. After cleaning, the distribution of the moods looked like this:



Then we filtered out the features that are not statistically significant in separating the moods with a designed hypothesis testing implementing Holm-Bonferroni correction. We chose to do this since this way we would drastically reduce the noise in the data that are not beneficial for classification. After filtering, The features that were left with are `'popularity', 'danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'duration_ms', 'time_signature'`. Then we split the data into training and testing sets and normalized it with StandardScaler. This was necessary since the range of the features varies drastically as some of them are from 0 to 1 and hundreds of thousands for duration_ms

Finally, we tried fitting different models to the data, including Logistic Regression, Decision Tree, Random Forest and SVM. By comparing the accuracies and fine-tuning with GridSearchCV, we finalized our model to be a RandomForestClassifier. The model scored 97% accuracy on the training set and 75.7% on the testing set. We analysed its performance further with the confusion matrix and found that most misclassification happened between ‘Happy’ and ‘Sad’. We also tackled the multicollinearity issue by training a separate model on data with highly correlated features removed. The performance was similar to the original model.

4. Recommendation

After successfully developing the classification model. We moved on to developing a content-based recommender engine that leverages the song mood and genres as criteria. We used the 600k+ Spotify Tracks as the candidates dataset and generated a user top track dataset with Spotify Web API. The cleaning process is similar to what we did in the Mood classification. We removed about 140,197 duplicated tracks from the candidates dataset and left with 446475 unique tracks. We then added the genres from the artist data. For the user dataset, we engineered a new feature called ‘preference_weight’ which measures how much the user likes a certain track. Then lastly we generated moods for all the tracks in both datasets.

To make the recommendation we leveraged cosine similarity from sklearn, for which we needed to generate the user vector and a candidates data matrix consisting of candidate track vectors. To do that, We firstly one-hot-encoded the moods and performed TF-IDF transformation for the genres to generate the user and candidates data matrix, then for the user matrix, we multiplied it with the ‘preference_weight’ and thereby generated the user vector. Lastly, we compared the cosine similarity between every candidate track to the user vector and were able to generate some results. However, the results appeared to be heavily biased towards ‘Happy’ tracks since it was the most preferred mood of the user and failed to produce a balanced recommendation result that we imagined to get. We also could not get a perfect evaluation for the model since Spotify does not provide access to the other user’s data nor does it have a ground truth criterion for making recommendations.

d5. Summary

In summary, we developed the fully functional mood classification model that we were hoping for. It could be particularly useful in the sense that we can replace the moods with any subject and the same method would work just fine. So it really expanded the space for adding new subjects for Spotify users to explore. It also provides a layer of interpretability as to what a piece of music could be associated with.

The recommender system, on the other hand, did not meet our goal of generating “balanced” recommendations. Despite not being able to properly evaluate the recommender system, I did recognize a couple of songs that I liked from the recommendation result. We will be working on finding a way to properly simulate Spotify users and the ground truth so we can properly evaluate and improve the recommender system.