Final Year Project Proposal

TU856

Automating Spotify Playlist Creation using Machine Learning

Alexandros Tsiogas
C20336236

School of Computer Science
TU Dublin – City Campus

30/09/2023

Table of Contents

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

*Alexandros Tsiogas*_____

Alexandros Tsiogas

27/09/2023

Summary

My project will be a web application that will be able to create new Spotify playlists for users using machine learning in ways not currently available in the Spotify app itself, or in any other major application with Spotify playlist creation functionality.

My app will allow a user to:

- View their playlists and the contents of each playlist in the web browser.
- Select one of their playlists and have a new playlist dynamically generated that will contain songs of a similar mood and tempo.
- Alter some variables such as “happiness”, “energy” and “danceability” before the creation of the playlist, which will change the songs that are selected as candidates for the playlist.
- Have the created playlist automatically added to their Spotify library, ready for listening.

Currently, Spotify’s recommendation system grants no way for the user to fine tune their recommendations - once completed this application will serve as an alternative way for Spotify users to create playlists that grants them more control over what they get to hear. Furthermore, it will serve as a way for Spotify users to find new music that they enjoy that is different, and hopefully superior, to the current Spotify recommendation algorithm, and other available solutions.

This application will also automate the playlist creation process for users, as currently there exists no way in Spotify’s core functionality to have a playlist automatically created based on user input.

Upon delivery, this app will be a useful companion to the Spotify application, that can enhance a user’s listening experience by curating music to their tastes.

Background (and References)

The aim of this project is to address some of the shortcomings of the current Spotify music recommendation system. A 2015 study [1] found that only 3% of Spotify users find that Spotify generated recommendations always match their taste, while 68.75% of users stated they match their taste only sometimes. The fact that users’ moods aren’t considered when they are being recommended music was highlighted as “one of the most important drawbacks” of the recommendation system. 66.7% of users interviewed chose “mood” as the main influence factor on the music they want to listen to. By granting a user the ability to take some control over the mood of the music they will be recommended, I hope to address this issue and create a system that better considers the mood of a user.

The shortcomings of the Spotify recommendation system are also an issue that developers have been trying to address a great deal in recent times, as is evident in the sheer volume of web and mobile applications that have been developed that utilise the Spotify API to build on the application's core functionality.

One such application is the mobile app "Discz" - a massively successful app that allows users to swipe through songs and give their sentiments on them, in a binary "like" or "dislike". The app uses the users swiping data to improve its recommendations. The Discz app uses a machine learning algorithm in its recommendations, which proves the feasibility of my project, and means the app can serve as a comparison to my finished product, and a source of research and comparison for my initial design. The popularity of the Discz app, as is highlighted in the 2022 Rolling Stone Article *Is Discz the Music Industry's Hottest New Recommendation Tool?* [2], also conveys how many Spotify users are eager to find alternatives to the app's built in recommendation system, which I believe makes this project all the more worthwhile.

In terms of the finer aspects of implementation, there are many resources available to aid me in my research. A primary resource is the *Spotify Web API Documentation* [3] itself, which offers extensive information about how the API can be implemented and interacted with. There are various online tutorials on how to implement the API itself, along with "spotipy", a python library that is used very often when interacting with the Web API, and one that I plan on using in mine too. The article *Getting Started with Spotify's API & Spotipy* [4] from Medium is a viable source on how to establish the foundations of a web app that interacts with Spotify.

Proposed Approach

I plan on completing my project in three distinct phases – Design & Research, Implementation, and Testing.

The research phase will constitute all my preliminary reading to assure the feasibility of my project. Beyond this, I will research the similar solutions highlighted in *Background* and others to identify possible ways to present my web application, and ways to implement my machine learning algorithm. Finally, it will be necessary to have a good understanding of the Spotify Web API and other technologies like Spotipy, which I will attain through reading documentation.

Design can be split into two main areas – the front and back ends. I plan on deciding upon my final design for the front-end presentation of my web application by drafting and refining wireframes of each page. The main design concerns in the back end relate to how I'm going to decide how to design my machine learning model. This may involve comparing the performance of multiple types of models before settling on the best option.

I plan on implementing my web application through HTML pages, styled with CSS.
I plan on using the Python web framework Flask to allow the Python elements to be implemented and into the web page – this is suitable, as I also plan on utilizing Spotipy, the previously Python library that simplifies interaction with Spotify data. Python will also be used to implement the machine learning algorithm itself. The final technology to be utilized is the Spotify Web API – this will allow us access to the user's listening data and grants the ability to create playlists on their behalf. Once fully implemented, the project will be presented as a web app, that prompts a user to log in, and then presents them with various options of playlist creation automation processes, which they can select. Upon selecting one of these options and, a playlist will be made for them and added directly to their Spotify library.

The basic functionality of the application, such as assuring a playlist is added when the correct button is pressed, will have to be rigorously tested by multiple users, to assure no differences in Spotify settings can interfere with the process, and that the web app itself is bug free.

The accuracy of the machine learning model will be assessed through a group of users, who will be asked to compare the recommendations of my web app to those of other similar apps, and Spotify itself, allowing me to compare the performance of my model to other similar models. I can then refine my recommendation system based on my findings.

Deliverables

My deliverables include:

- An aesthetically pleasing front end, with a visually pleasing UI that is easy to navigate and offers a good user experience.
- Functionality that prompts a user to log into Spotify upon opening the URL and grants us access to their data.
- Functionality that allows a user to have a playlist created and added to their library, after interaction with elements of the web app.

- A page build on the previous functionality and allows a user to select a playlist that exists in their library and have a playlist dynamically generated based on this playlist's contents.
- A machine learning algorithm that can select songs from Spotify's database to recommend to a user, based on their selected playlist.
- Elements on the web page that allow a user to change variables about the mood of their desired recommendations, which will be processed by the machine learning algorithm.
- Functionality to add these recommended songs to a playlist and to add this playlist to the user's library.

Technical Requirements

This project will be built primarily using Python and its libraries, created and ran in VS Code – The Flask web framework will be used to present the app on the web browser. HTML and CSS will be used for the front end of the web app.

No specialised hardware or infrastructure will be required.

Conclusion

In conclusion, the concept for this project is to create a web application that allows a user to have Spotify playlists automatically created and saved on their behalf, that are created using a machine learning model. The user will be granted a level of control over the mood, tempo, and energy of the music they are recommended, to make the recommendations conform better to the user's specific desires.

This project aims to build on Spotify's current functionality for playlist creation and to address some of its shortcomings by creating new and unique ways of curating playlists, powered by machine learning.

References

[1] Ding, Yiwen, and Chang Liu. (2015)."Exploring drawbacks in music recommender systems: the Spotify case."

[2] Leight, E (2022) “Is Discz the Music Industry's Hottest New Recommendation Tool?” (Online) Available at: https://www.rollingstone.com/music/music-features/discz-tiktok-recommendation-app-1324785/ (Accessed 27th September 2023)

[3] Spotify Web API Documentation (Online) Available at https://developer.spotify.com/documentation/web-api (Accessed 27th September 2023)

[4] Tingle, M (2019) “Getting Started with Spotify’s API & Spotipy (Online)” Available at: https://medium.com/@maxtingle/getting-started-with-spotifys-api-spotipy-197c3dc6353b (Accessed 27th September 2023)

Appendix A: First Project Review

Title: Football Data Mining, Result Prediction, and Visualization

Student: Yahia Ragab

This project is a mobile and web application that allows users to view in depth stats on the English Premier League, and to view predictions of the outcome of matches, created with machine learning.

A huge amount of historical data relating to the English Premier League was mined and stored. A sophisticated machine learning model was developed to allow for predictions to be made, and a detailed front-end was designed that visualises statistics in digestible ways.

Python was used for creating the machine learning model, with use of libraries like NumPy, Pandas and Matplotlib for data visualisation. The web app was presented as a Django application, and Docker was used for deploying cloud components. MYSQL was used as the database management system.

This project contains a machine learning algorithm written in Python, similar in nature to the one I plan on creating in my app. The model is clearly highly sophisticated, and after rigorous testing seemed to yield very accurate results, which I believe is a huge strength of the project, and one that pertains to my own project. The amount of data made available for viewing on the app is similarly impressive, and the way in which it is visualised is understandable and useful.

The front-end design is quite cluttered and perhaps not very user friendly and lacks some aesthetic appeal – it could have done a more thorough design phase that focused more on user experience. The project is, however, of a very high standard and displays many strengths which I could draw inspiration from

Appendix B: Second Project Review

Title: Moodify: Mood Playlist Generation using Song Emotion Classification by Lyrics and Audio Features

Student: Louis Miguel Chavez

This project is an Android application that classifies songs by mood, based on their lyrics and audio features, using AI and the Spotify API.

The project contains a complex machine learning algorithm for sentiment classification, used to determine the mood of songs through sound and lyrics – something which pertains very closely to my own proposal. It is also presented in an Android application with a very appealing and user-friendly design.

The sentiment and music mood classifiers were made through Python, in Jupyter Notebook – other components were made in PyCharm. The Android application was made in Android Studio, using Java for functionality and XML for the layouts and visual components. A DigitalOcean Managed Cluster was used as the final database functionality.

The main strengths of this project, in my view, are the originality and accuracy of the mood classifier - it offers interesting insight into the listening habits of users. The further functionality allowing users to construct playlists based on a chosen mood, is another unique strength. I also find the front-end to be aesthetically pleasing and accessible for users of all technical skill levels. The only evident weakness of this project is that it requires users to manually upload playlist files for analysis, as opposed to having the client automatically extract them from the device – this is something I aim to avoid in my project.