NEW YORK INSTITUTE OF TECHNOLOGY

DTSC 710 - M01

Machine Learning Project Report

Spotify Music Recommendation System

Professor Wu

Team 2 Members:

Selina Narain	1261565
Neelam Boywah	1226855
Zoya Haq	1222440

Table of Contents

Introduction & Motivation

A music recommendation system can be designed to create personalized music suggestions for the listeners based on various different factors like preferences, listening history, and more. A music recommendation system allows for streaming services such as Spotify and Apple Music to offer tailored experiences to each listener. Through the analysis of user data, a music recommendation system can suggest music that aligns with their preferences which would ultimately lead to engagement and user satisfaction.

Music recommendation systems can aid users to finding new artists, genres and songs that they may not have been exposed to before. By doing so, the user's listening experience can broaden their playlists with their type of music. Recommendation systems can also be used for revenue and gain for artists who are looking to promote their music. To this point, it can also recommend lesser known artists where the user may enjoy the artist's style or genre. Sometimes streaming platforms can have a lot going on as there are so many options to choose from. This type of system can streamline aspects that will aid in categorizing content without the user constantly searching for something they may like. The tools and technologies that were used to build the Spotify Music Recommendation System include: Google Collab, Google Drive, Python, Spotipy, Spotify Web API and Discord.

Methodology

In a music recommendation system, a clustering technique such as K-means clustering can be utilized to group similar songs together based on their characteristics. Through clustering, the system can create clusters of songs based on features such as genres and listening habits and preferences. Using a sklearn's Pipeline, we can combine multiple methods to work parallel with each other and cross validate them together. K-means clustering can also be used to identify genres that have similar and overlapping characteristics. Through the process of clustering songs into genres, the system can offer playlists for specified genres, allowing the user to easily access their music. For the K-means Clustering for Genres model, we use a pipeline to have Standard Scaler and K-means work together in order to perform a Grid Search and predict the outcome. The Standard Scaler function from the sklearn library is used to standardize our data values into

a standard format which is used in both baseline models. We then create another pipeline using K-means with the best parameter result from Grid Search.

In addition to K-means clustering, the use of principal component analysis in a music recommendation system allows for dimensionality reduction. There are various attributes which can lead to overfitting and reduced performance of the algorithms. We used PCA for our song clusters in order to reduce the dimensionality to 2 dimensional data as well as K-means clustering to hone our data further. PCA also helps eliminate redundancy which can become helpful because the redundant features do not provide extra information and it may introduce noise or bias in the system itself.

Datasets

In our Music Recommendation System, four spotify datasets were utilized. The Spotify datasets that are used for our machine learning methods are ‘data.csv’, ‘data_by_genres.csv’, and ‘data_by_artists.csv’. These datasets are similar to each other but consist of different data types like artists and genres. Some of the features that are being utilized from these datasets include danceability, instrumentalness, energy, liveness and acousticness. For our visualizations we used all of our datasets to identify trends in specific audio features as well as throughout time. This helped us visualize different graphs portraying music consumption over the decades, sound features over the years, and using sklearn T-distributed Stochastic Neighbor Embedding (TSNE), visualization of music genres clustered by audio features, and plotly to visualize Spotify song clusters in order to help us understand the datasets better.

Data Preprocessing

In order to prepare and process the raw data from the datasets to use in our system, we used visualizations, feature selection, standard scaler and principal component analysis. In our visualizations we are looking through all the features and its attributes to see what is most needed for our recommendation system. In our baseline models, K-means for clustering genres and K-means and PCA for clustering songs, we used features that we thought would be best suited for our models. These features include 'acousticness', 'danceability', 'duration_ms', 'energy', 'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo', 'valence', 'popularity', and 'key'.

StandardScaler is a preprocessing technique commonly used in machine learning to standardize and normalize the features of the data. Through the application of StandardScaler, the features can have a zero mean and variance which will prevent features with larger values to dominate the algorithm as a whole. In order to decrease the noise in our dataset, we also used PCA to reduce the number of dimensions in our training set.

Baseline Models

The purpose of establishing a baseline model in the music recommendation system is to evaluate the performance and effectiveness of pipelines we created and how effective it was to later on build the recommendation system. Two baseline models were created for us to reference in the recommendation system. The first model is a K-means for clustering genres Model. In this model we use a Pipeline to scale the data and perform K-means clustering. For the grid search of genre clustering, the hyperparameter tuning had a range of 1 to 20 clusters. When performing the grid search, the best parameters displayed 19 clusters to use. We then use Pipeline again to scale the data and perform K-means clustering based on the parameters provided from grid search. After fitting the K-means clustering model, we were then able to visualize the clusters with centroids using TSNE. The second model is a K-means and PCA for Clustering Songs. In this model we also utilize Pipeline to scale the data and perform K-means Clustering. For the grid search, the hyperparameter tuning for clusters had a range from 1 to 25 and the best parameters were 24 clusters to use. Then we used Pipeline two more times, the first to scale the data and perform K-means Clustering based on the Grid Search and the second, to scale the data and perform PCA. After fitting the K-means clustering PCA model, we were then able to visualize the clusters using plotly.

Model Description

The first baseline model, K-means for clustering genres, scales the data and performs K-means Clustering on the genres data using Pipeline. The Grid Search displays the best number of clusters to use. We use Pipeline again based on the Grid Search and then we fit the K-means Clustering Model. Lastly, we are able to visualize the clusters with centroids. The second baseline model, K-means and PCA for clustering songs is similar to the first baseline model by

scaling the data, performing K-means clustering on the songs data, and running the grid search to find the best number of clusters to use. After doing this we utilize the Pipeline twice more, once based on the Grid Search and another time performing PCA. After we fit the K-means Clustering PCA Model, we are able to visualize the clusters.

Using the baseline models we created, we can then build the recommendation system. The recommendation system uses Spotify Web API to pull metadata from Spotify regarding song names, release, year, audio features, etc. There are specific features used to recommend these songs which can be cross referenced from our Kaggle datasets as well as the API.

Implementation

In the Music Recommendation System, we have created a Spotify application through Spotify for developers and then utilized our Spotify Web API client credentials. We then used Spotipy, a python client used in conjunction with the API. This helps us pull metadata and identify Spotify's catalog for songs. We then define a method to find song details based on its name and year of release. After getting those details, we extract relevant song details and store them into an empty list created. The recommendation system then uses euclidean distance. There are different features that are used to recommend songs like year, loudness, mode, etc. Then the model tries to find the song in the dataframe based on the name and year of the release and if that song is not found in the data frame itself, it goes to the Spotify API for additional song data. There is an empty list that is created to hold feature vectors for each song in the input list. Through metadata, allows us to get the feature data for the songs from the Spotify database. If the song cannot be found, there is a warning message that appears stating it does not exist. The audio features of the song are extracted in a numpy array and appended into another list. The mean feature vectors are then calculated across all the songs in the input list and returned.

Furthermore, there is an empty default dictionary in which a function iterates over each key-value pair in that dictionary. For each of these pairs, the value is appended to its corresponding list and later a fully flattened dictionary is returned. There is a list of metadata columns that we want to return for the recommended songs and the flattened version of the input song list represented as a dictionary. The mean vector of the input song list is calculated and then

a scaler is used. There is a scaled data of all the songs that are in the dataframe. There is also a matrix of cosine distances between the scaled songs and the data. The top *n_songs* in the data that are closest to the center are ultimately based on the cosine distance. There is a data frame of recommended songs and a dictionary is returned with the recommended song and its respective metadata. The system then returns recommended music for the users displaying the songs name, year, artists and popularity.

Computational Requirements

There are many computational requirements in a music recommendation system and all this depends on the size of the data set, number of users, complexity of algorithms and the scalability goals of the system. Some of the key computational requirements for this specific recommendation system include data preprocessing which involves cleaning the data. Another computational requirement is feature extraction. Extracting the relevant features from music audio signals or metadata can be computationally intensive. We also had to complete various mathematical computations such as cosine similarity to find the similarity between data points based on song vectors.

Analysis & Results

Through all the aforementioned machine learning techniques, the Spotify Music Recommendation System was successfully developed. Although we were not able to find the exact accuracy metric, we were able to successfully identify clusters throughout our dataset based on the genres and songs. Therefore, using those clusters with our recommendation system, we were able to successfully output 5 song recommendations based on our initial models. This output gives us the name of the recommended songs, the year it was released, the artist who sang the song, and its popularity score. Overall, we were able to recommend these songs based on our results. For future improvements, we also believe it would be beneficial to include playlists as well in order to improve the predictions of the recommended songs.

References

Maharshi Pandya. “Spotify Tracks Dataset.” Kaggle, 22 Oct. 2022, https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset.

Mavani, Vatsal. “Spotify Dataset.” Kaggle, 17 Dec. 2021, https://www.kaggle.com/datasets/vatsalmavani/spotify-dataset.

Samoshyn, Andrii. “Dataset of Songs in Spotify.” Kaggle, 6 Dec. 2020, https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify.

Schedl, Markus, et al. “Current Challenges and Visions in Music Recommender Systems Research - International Journal of Multimedia Information Retrieval.” SpringerLink, Springer London, 5 Apr. 2018, https://link.springer.com/article/10.1007/s13735-018-0154-2#:~:text=A%20number%20of%20approaches%20have,domain%20recommendation%2C%20and%20active%20learning.

“User Guide: Contents.” *Scikit*, scikit-learn.org/stable/user_guide.html.