Spotify Song Genre Clustering

Introduction

With Spotify having more than 551 million users and 220 million subscribers in more than 180 markets, it's clear that music is a universal common interest among many people [1]. People having varying opinions on how to keep playlists. Some people like shorter playlists, others like longer playlists, and some don't keep playlists at all and stick to their liked songs or manually queuing up music. If someone is inclined to keep long playlists or listen to their liked songs, a lot of different themes and moods of music can get jumbled up into one thing. Even making playlists themselves deters people from making them in the first place just because of how time consuming it can be. It is very easy to get caught up in how specific a playlist can be, or even if it is as simple as making a playlist by genre, it takes a while to make sure the songs that you are putting on fit what you are going for. The Spotify Playlist Maker aims to streamline this playlist making process, something that can be very tedious for anyone. This project would be an app where a user can provide their Spotify account/ a link to a playlist so that this app can access the playlist/liked songs of the user and make a variety of playlists based on genre and/or different mixes of different genres. The app would theoretically run on three different models. Here, we are focusing on the first model where we are focused on assessing which clustering algorithm would be best for this overall project. There are four possible algorithms to consider: K-Means, Bisecting K-Means, Fuzzy K-Means, and Affinity Propagation (Hierarchal Clustering). K-Means clustering is a distance-based algorithm that is scalable to large data. Clusters are specified and data points are assigned to only one cluster. Bisecting K-Means is a hierarchal variation of K-Means, Fuzzy K-Means is another variation, allowing for one data point to be assigned to multiple clusters. Affinity Propagation is a hierarchical clustering algorithm based on similarity.

Data Exploration

Data was sourced off Kaggle [3] which was sourced through the Spotify API for a total of 114,000 song records, with 1 NA in artists and album_name, which were almost immediately dropped. In terms of similar projects, all the projects that were made with this dataset and similar datasets that were not chosen for varying reasons, were focused on song recommendation and classification. When looking around at other sources, Alejandra Vlerick from Towards Data Science looks like they are doing something similar, although the details of the project cannot be seen at this time due to the site's paywall. From what can be gathered, the goal is the same, but the methodologies seem completely different [4]. Analysis began with looking at the distribution for genre. Without cleaning, all of the genres have exactly 1000 records. We dropped 450 exact duplicates and 6601 inexact duplicates. An inexact duplicate can be defined as a duplicate that does not take into account the album name and track ID attributes. After dropping those duplicates, we are left with a minimum genre count of 540 and a standard deviation of 87 songs (median = 979). In terms of other transformations, we converted duration_ms to duration_sec, converting from milliseconds to seconds as well as creating an algorithm to convert all 114 genres from string labels to numbers. At this point, N = 106,949. After this, each notable variable's distribution (popularity, duration, danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo, explicit, mode, time signature, and key) was assessed, dropping outliers as needed. Outliers were dropped using IQR, and then again with brute force if there were outliers that remained. After dropping outliers, we are left with N = 45,671. After our distribution checks, we checked correlation for each variable. Notably, energy and loudness are heavily correlated at .72 and danceability and valence are decently correlated at .42. Speechiness and energy are also decently correlated at .32. Acousticness is decently negatively correlated with energy and loudness, -.73 and -.51 respectively. To see some of these correlations better, scatterplots were made.

Data Preprocessing

We have varying methods in how we can prepare our data for our models. Four different datasets will be prepared and plugged into each model. The first dataset will be the original data with nothing changed from it. This will serve as a benchmark for the other datasets with each model. The second dataset will have the necessary attributes normalized and standardized, as well as popularity and explicit dropped. When going through the distributions, we concluded that these variables should be dropped since they are not really telling of genre. Popularity may actually impact since popular songs may belong to one genre more than others, allowing for a bit of bias. Explicitness is not really telling either since artists release both versions either way and also 91% of the data is not explicit. Therefore, it should be assumed these two variables are dropped for each dataset except the first. The third dataset will take the dataset and will run PCA on it with n_components = 7 or an optimal number if 7 is not producing good results. 7 is used because of the fourth dataset. The fourth dataset focuses on using 3 PCAs with n_components = 1, making variables to plug back into a dataset with the variables that weren't used in the 3 PCAs. This leaves us with 7 variables. The variables will be separated as follows:

- Energetic_loud: energy, loudness, speechiness
- Upbeat_happy: danceability, valence, instrumentalness
- Musical_measures: key, mode, tempo, time_signature

Our final 3 PCA dataset would be energetic_loud, upbeat_happy, musical_measures, track_genre, liveness, acousticness, duration_sec.

Clustering

First thing that should be noted is that we opted out of normalizing any variables due to most of the measures already being in between 0 and 1 anyways. Instead, all variables that were between 0 and 1 were left alone, leaving tempo, duration_sec, and loudness to be only standardized. When running our PCA datasets, our third dataset (1 PCA) showed an optimal cluster size of 2, leaving that dataset with n_components = 2 instead of the original 7. After having our datasets prepared, the data was ready to be plugged into our algorithms. There was an issue that came up in terms of interpretation though. Our original algorithms consisted of different KMeans algorithms and Hierarchal clustering. The goal for clustering here is to eventually implement it into an app, meaning that these algorithms cannot be terribly slow. Both types of methods also yield their own results, so it is detrimental that we properly assess how to proceed here. According to a graph found in the HDBSCAN documentation on benchmarks for different Python machine learning algorithms [5], KMeans is by far the fastest clustering algorithm. The issue with KMeans in this context though is that it does not work well with large cluster sizes, meaning that if we were to try to assess this through the lens of genres, we would be passing a K-size of 114 for the 114 different genres in the dataset. Deviating from this number means that we must redefine how genres get interpreted, which moves into very subjective territory. Instead, it may work better to view these clusters as playlists instead of genres and run the algorithm over a series of different K-sizes to represent how the clusters would look with different K-sizes, or the determined number of playlists to derive from the original dataset. Upon coming across a reddit post from the subreddit r/dataisbeautiful, user trevorData clustered music into genres using SciPy's Agglomerative clustering in R [6]. According to the graph, this is one of the faster hierarchal clustering algorithms which makes it viable for implementation. With this clustering, it makes more sense to try to interpret these as genres instead of playlists since the agglomerative clustering algorithm is going to go from a single data point to the entire thing. Theoretically, it should be able to find those different genres as it clusters up. With KMeans, all our runs for each dataset gave us a high silhouette score for K = 2 clusters, with none of the runs breaking too far past a silhouette score of .6. Looking closer into the different samples from each cluster group, we get odd mixes of genres. For example, death metal and Disney music are often grouped together. Given our relatively decent silhouette score, I can only assume that these songs are most likely being grouped by mood together. In our K = 2 clustering, it could be argued that one cluster is representative of slower, more moody music versus the other cluster that has genres that tend to have more upbeat music. It is hard to truly say though due to working with music data, something that is inherently best interpreted by listening and feeling the music. Numbers on music can only do so much justice to the actual content matter. Our hierarchal clustering produced more unexpected results. Just like our KMeans clustering, our agglomerative clustering using Ward's linkage also gave us two distinct clusters, excluding the cluster that holds all the datapoints. As mentioned previously, more clusters were expected to represent genres. This result in clustering is most likely attributed to the many unique attributes we have that gives a detailed view of the mood of a song. Attributes like danceability, valence, or energy gives a quantifiable amount to moods, which ultimately provides a spectrum of moods for the algorithm to view. Ultimately, clustering into two distinct clusters makes sense in the eyes of the algorithm given the data that was inputted. Our Cophenetic Correlation Coefficient uses distance and dissimilarity to assess how good and concise our clusters are in relation to the distances between the original datapoints. The higher the coefficient, the better the clusters are representing our data. Here, our runs produced high results, with the scaled and both PCA datasets scoring about

.74. Our songs3PCA dataset scored the highest by a very slim margin. Looking into our samples more, we can see that although it is hard to assess what type of mood is being portrayed without listening to the music and with the confusion of the variety of genres that come through, we can still see that our clusters are clustering on mood more than anything else. It may have been expected that agglomerative clustering would help on honing in on different moods within genres for distinction, but either way the clustering is giving us viable results. This can be supported with our Cophenetic Correlation Coefficient being relatively high with our clusters, showing that the two general moods that we found through clustering are an accurate representation of what existed in our original dataset.

Conclusions

These results were unexpected yet make a lot of sense with the context of our data. In the future, it may help to look more into different ways to house data for music to get results that yield distinct clusters for genres. Sticking with mood as well, it may also be viable to investigate seeing if Spotify has more attributes for music that can give more details about the songs, which may help in drawing better lines between the genres. Another thing that may help here is querying the data directly from the Spotify API as we may be able to source different sections of music. Even with our final N = 45,671 and 114 genres, there is still a lot of music and different genres that are not actively being represented in our sample. Porting our models directly into the API would allow us to change what exactly is going into the model and seeing how that affects our clusters. Outside of cluster cohesion, it may also be incredibly valuable to run a survey with Spotify users to understand more about listening habits, specifically with how users interact and use playlists/their queue. Not only would it help us identify target groups, but it would also maybe help us with identifying even more vulnerabilities and problems with music listening and playlist making.

References

1. https://newsroom.spotify.com/company-
2. info/#:~:text=We%20are%20the%20world's%20most,in%20more%20than%20180%20markets.
3. https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset/data
4. https://towardsdatascience.com/k-means-clustering-using-spotify-song-features-9eb7d53d105c
5. https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html
6.
    a. Reddit
       https://www.reddit.com/r/dataisbeautiful/comments/blwed1/using_machine_learning_to_create_a_music_genre/
    b. Github Code
       https://github.com/trevorData/RYMGenres/blob/master/RYMDescriptors.R