Project Report: Spotify Recommendations

Background and Problem

With 35% market share in 2019, Spotify is by some distance a leader in the music streaming services market segment. In order to maintain that distance between the closest competition, a subscription service must at the core of its business be able to provide a positive experience in order to retain its paying customers.

At times a user may just want a simple means to listen to a series of tracks without having to manually select them could be useful such as when hosting a party, making dinner etc. One of the biggest weaknesses I've seen in the Spotify platform is the lack of ability to group music by genre even for saved tracks so creating a system to automatically recommend similar tracks within the user library may greatly enrich the user experience.

By leveraging my saved tracks data in Spotify, I created a K-Means Clustering model which can recommend similar tracks to any given track in a user's library which can serve both as a means to automate track selection and a tool for categorizing the user's music.

Data Wrangling

This is a consolidated dataframe with both tracks, artist and features data. The data comes from my own personal music library on spotify which is 656 records with 30+ columns. Since some tracks can have multiple artists and often have more than one genre as well, I created additional columns to capture the data. I also added an extra column produced via K-Means clustering to supplement the grouping model implemented.

The data is very clean for the audio features where data is factually present so cleaning was limited. I only ended up dropping the additional artist columns and additional genres with more than 25% of the data missing. In the process of cleaning up the data for model fitting, I ended up dropping more columns that did not contribute to meaningful improvements to the mode while keeping all the rows.

In the process of testing different modeling possibilities, I created two main dataframes, one for artist and associated data and another for track specific data. The reason artists warranted its own dataframe was due to the fact that genres in Spotify are tied to the artist instead of the track itself.

The final dimension of my dataset was 656 rows by 18 columns.

Exploratory Data Analysis

Audio features native in the Spotify platform all introduce some aspect of the audio track with the following limits

- Danceability: Scaled between 0 and 1
- Energy: Scaled between 0 and 1
- Loudness: Generally between -60 and 0 decibels
- Speechiness: Scaled between 0 and 1
- Acousticness: Scaled between 0 and 1
- Instrumentalness: How much vocals are in a given track. Values above 0.5 are generally 100% instrumental. Scaled between 0 and 1
- Liveness: Detects live audience, 0.8 or higher confidently indicates a live recording. Scaled between 0 and 1
- Valence: Indicates how generally ‘happy’ a track sounds. Scaled between 0 and 1
- Tempo: Generally between 50 and 200 Beats per Minute(bpm)
- Duration: Measured in milliseconds
- Popularity: Between 0 and 100

I am mostly interested in the relationships between these features to confirm that they are generally independent of each other and individually to see how they generally correspond to the clustering made in the previous step for some making inferences about the tracks in each cluster.

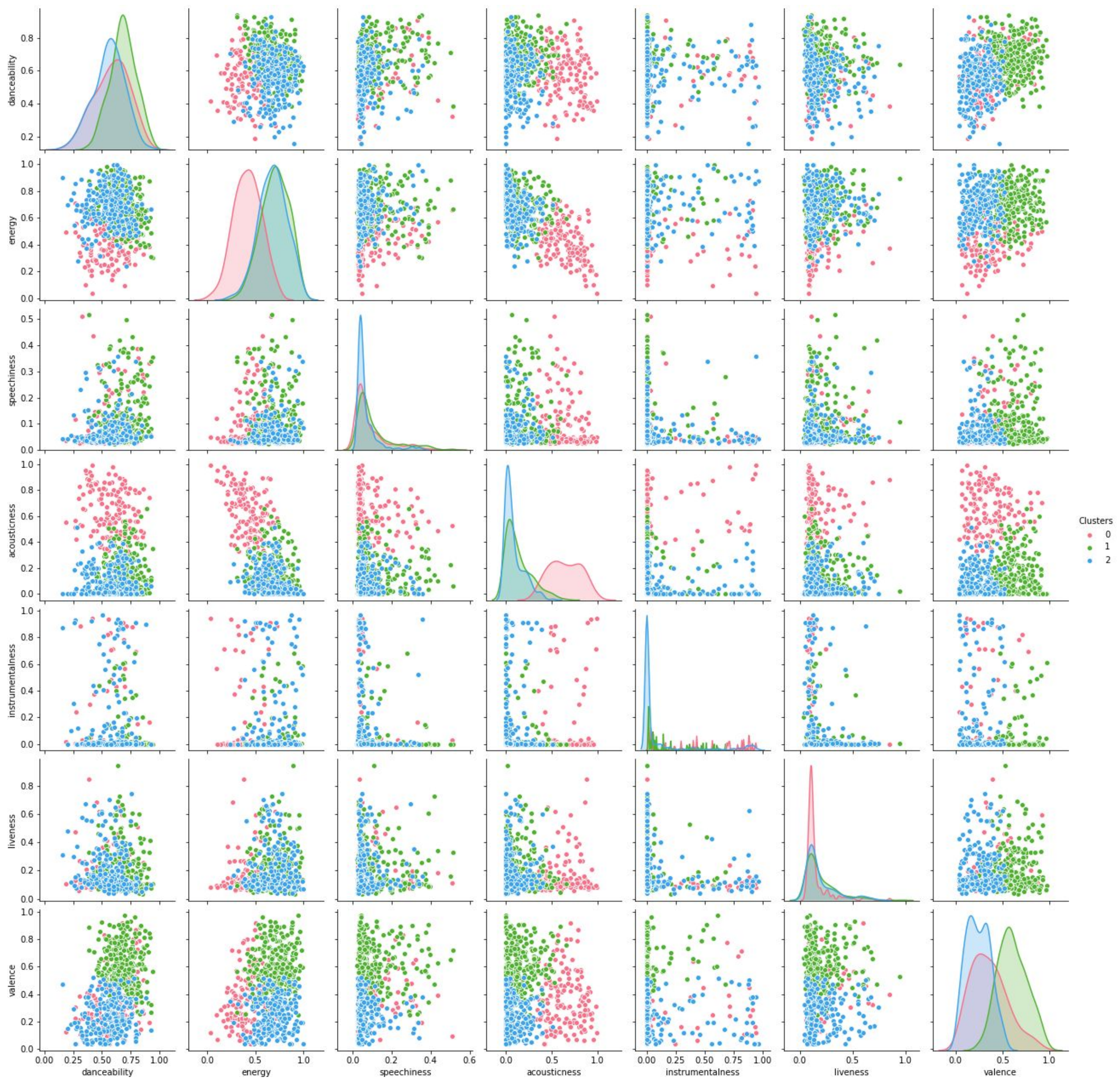
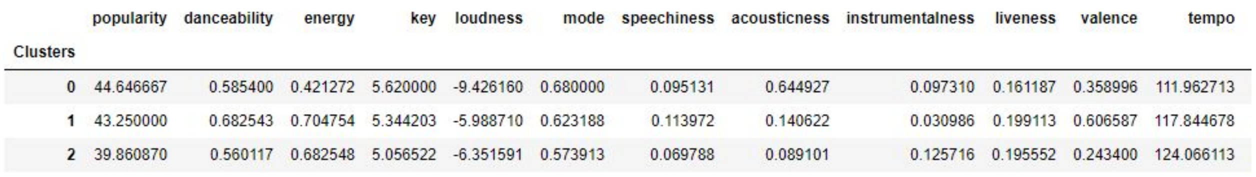


Fig 1: Pairplot with coloring for clusters

There does not appear to be any correlation between these features and thus passes an eye test for independence.

Cluster Analysis

Three clusters were chosen based on the elbow plot heuristic.Although they were all fairly close in duration, the audio features made each cluster fairly distinct.



Clusters	popularity	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo
0	44.646667	0.585400	0.421272	5.620000	-9.426160	0.680000	0.095131	0.644927	0.097310	0.161187	0.358996	111.962713
1	43.250000	0.682543	0.704754	5.344203	-5.988710	0.623188	0.113972	0.140622	0.030986	0.199113	0.606587	117.844678
2	39.860870	0.560117	0.682548	5.056522	-6.351591	0.573913	0.069788	0.089101	0.125716	0.195552	0.243400	124.066113

Fig 2: Three clusters set in order to supplement recommendations

The difference comes down to the aspects of valence, loudness, energy and danceability. Cluster 1 is the upbeat, happy playlist with the highest valence, loudness, energy and danceability. And while both clusters 0 and 2 tend to be sad sounding, cluster 0 is also much more likely to be an acoustic track, with less energy and also much quieter by comparison to the others. This suggests a quieter, live performance compared to a lively sounding studio recording.

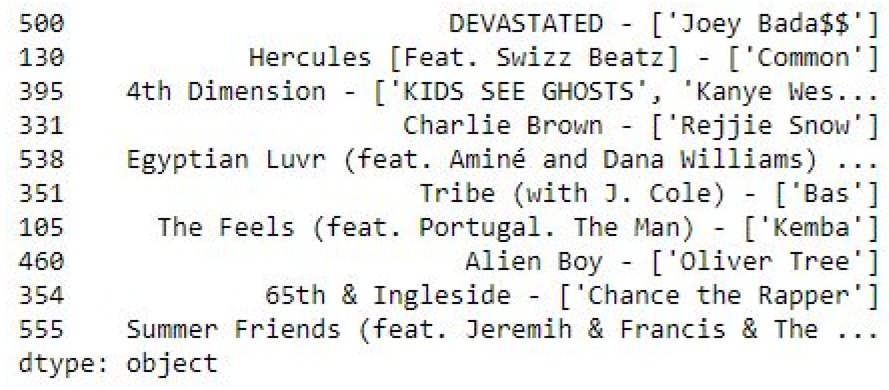
Model Selection

I removed any categorical features (artists and genres) for the final KMeans model since they individually did not contribute much as features (very few artists show up repeatedly in the saved tracks and far too many different genres overall). However, before I made this decision I attempted to leverage the artist dataframe with the genres associated with students in a simple recommendations system based purely on genres.

TF-IDF Model (Simple recommendations - Genres)

For this model, I concatenated all the genres associated with each value and vectorized these values (replacing spaces with dashes ('-') in order to avoid any loss of information for genres with space in them.

This array is then fitted and transformed in order to generate track recommendations which runs fairly quickly through a single function. Below is an example set of recommendations with a rap track set as the seed.



```
500                        DEVASTATED - ['Joey Bada$$']
130             Hercules [Feat. Swizz Beatz] - ['Common']
395    4th Dimension - ['KIDS SEE GHOSTS', 'Kanye Wes...
331                      Charlie Brown - ['Rejjie Snow']
538    Egyptian Luvr (feat. Aminé and Dana Williams) ...
351                       Tribe (with J. Cole) - ['Bas']
105      The Feels (feat. Portugal. The Man) - ['Kemba']
460                          Alien Boy - ['Oliver Tree']
354              65th & Ingleside - ['Chance the Rapper']
555    Summer Friends (feat. Jeremih & Francis & The ...
dtype: object
```

However as mentioned above, this model does have a huge weakness in that it's based purely off of genres which means any two tracks for the same artist would generate the same recommendations, regardless of the track's individual content. To this end, this model may be a better fit as a means to recommend related artists instead. However to improve on tracks for this instance, I decided to leverage the audio features instead.

K-Means Feature Selection

For the audio features data, I attempted different scaling methods feeding into the K-Means model in order to do my unsupervised learning. I gauged results purely based on logical order and choice of the recommendations.

I attempted absolute scaling, standard scaling and MinMax scaling. Since these recommendations are somewhat contingent on personal taste, this may be an aspect that is ultimately left to the user to choose. Personally I liked the recommendations from the standard scaling. Here are some of the results from this model.

Generic Pop Example - Rock Bottom by Hailee Steinfeld

1. *The Lumineers - Cleopatra*
2. *Twin Shadow - Saturdays (feat. HAIM)*
3. *DEAN - D (Half Moon)*
4. *Fall Out Boy - Wilson (Expensive Mistakes)*
5. *NOTD - So Close*
6. *Aries - CAROUSEL*
7. *Lola Marsh - You're Mine*
8. *Mr. Probz - Nothing Really Matters - Afrojack Remix*

9. *Gryffin - Winnebago (feat. Quinn XCII & Daniel Wilson)*

Hip Hop / Rap Example - 65th & Ingleside by Chance The Rapper

1. *KAYTRANADA - YOU'RE THE ONE*
2. *Kanye West - Gone*
3. *SZA - Quicksand*
4. *Snakehips - All My Friends (feat. Tinashe & Chance the Rapper)*
5. *Kendrick Lamar - X (with 2 Chainz & Saudi)*
6. *Kygo - Remind Me to Forget*
7. *Louis The Child - Better Not (feat. Wafia) - Hotel Garuda Remix*
8. *Sabrina Claudio vs. - Don't Let Me Down - Sabrina Claudio vs. Devault Remix*
9. *SBTRKT - Wildfire*

Standard Scaling

This means of scaling normalizes each data point with the respective mean and standard deviation.

Standardization:

$$z = \frac{x-\mu}{\sigma}$$

with mean:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

and standard deviation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$$

K-Means Clustering

Can be summarized down to alternating between two steps, assignment step and update step.
On each iteration, observations are assigned to the cluster with the nearest mean measured with squared euclidean distance.

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \le \|x_p - m_j^{(t)}\|^2 \ \forall j, 1 \le j \le k\},$$

where each x_p is assigned to exactly one $S^{(t)}$, even if it could be assigned to two or more of them.

Cluster means are then recalculated and then reassigned to the new clustering

$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

I also tried different neighboring binary tree algorithms (KD Tree, Ball Tree, Brute) to further tune the results produced, but this did not appear to make any differences in my results.

Limitations + Discussion on Future Improvements

Since the model uses a user's saved tracks, recommendations will only work well if run on small batches otherwise the entire saved tracklist will cycle through. To address this in future improvements, a possible solution would be to re-run K-Means each of the recommendations in batches of 20 tracks since it will at least keep the playlist to similar music.

I can also leverage lyrics in the future as categorical variables to improve track recommendations for more 'speech-y' music.

Ultimately the biggest hindrance for making stronger recommendations in this case is the lack of availability for individual usage data ie. how many skipped tracks, playback count, frequency of playback etc. Having these extra data points would not only give us more features to make recommendations but also provide a whole dimension of feature engineering possibilities. Hopefully in the future, Spotify will be more liberal in allowing premium user access to his/her own data.