Spotify Track Recommendations

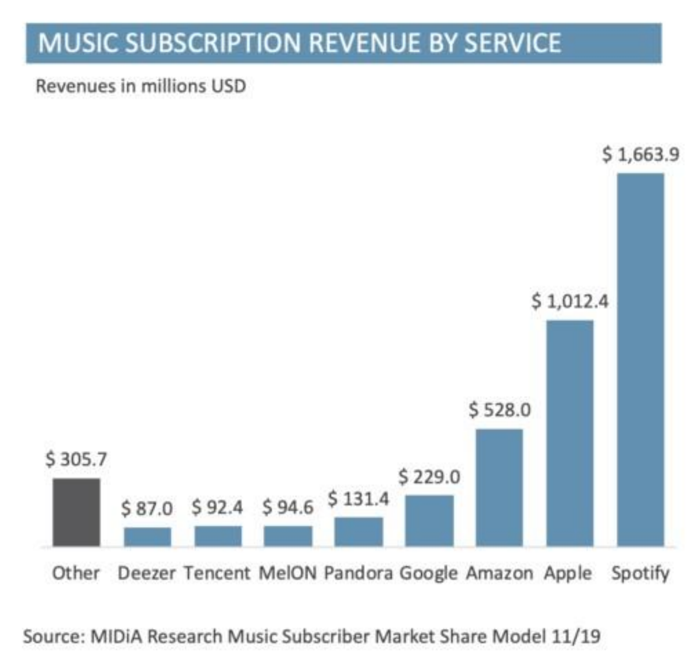
Improve shuffling with Unsupervised Learning as your DJ!

Background

As an industry leader, Spotify must prioritize customer experience in order to avoid losing paid subscription members to competitors

Retention is critical especially given the thin margins of this space with royalties due to the artists and the record labels.

How can we improve user experience? How do we keep the user engaged and listening?



Who might care?

Spotify

Spotify Paid Users

All Global Listeners (free version!)

Track + Audio Features

Tracks

- Artist(s)
- Popularity
- Genres
- Duration
- Track Name
- Date Saved

Audio Features

- Danceability
- Energy
- Key
- Mode
- Loudness
- Speechiness
- Acousticness
- Instrumentalness
- Liveness
- Valence

Data Summary

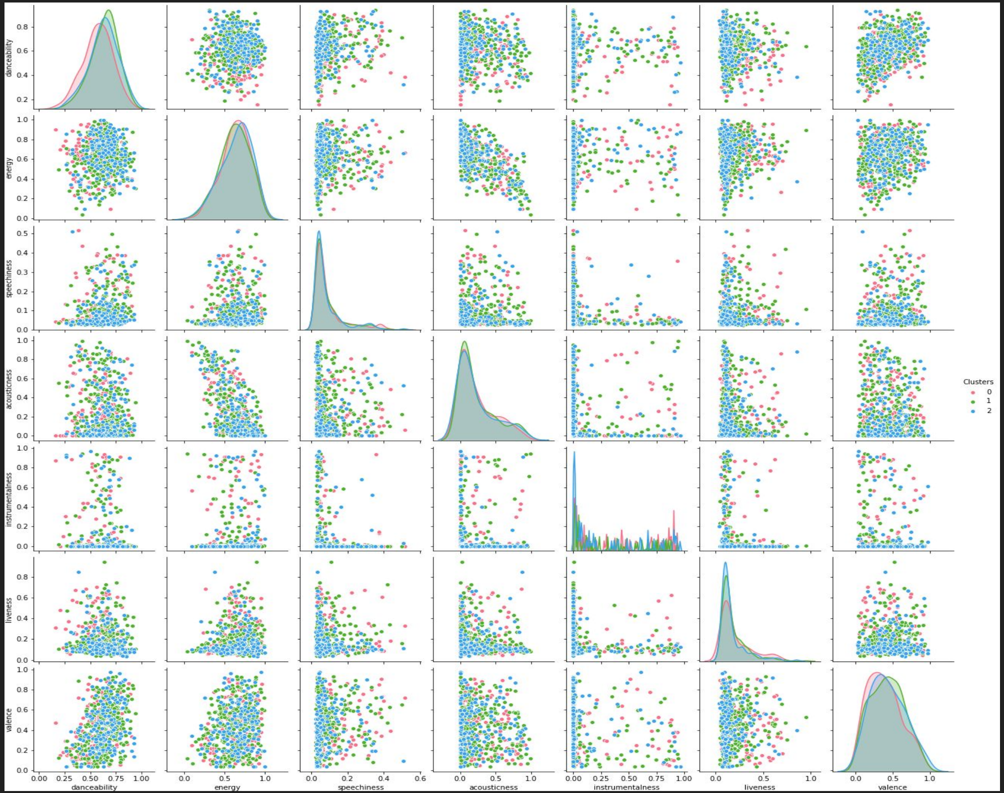
Saved tracks from between 09/2014 through 08/2020, collected through Spotify API via the Spotipy python library

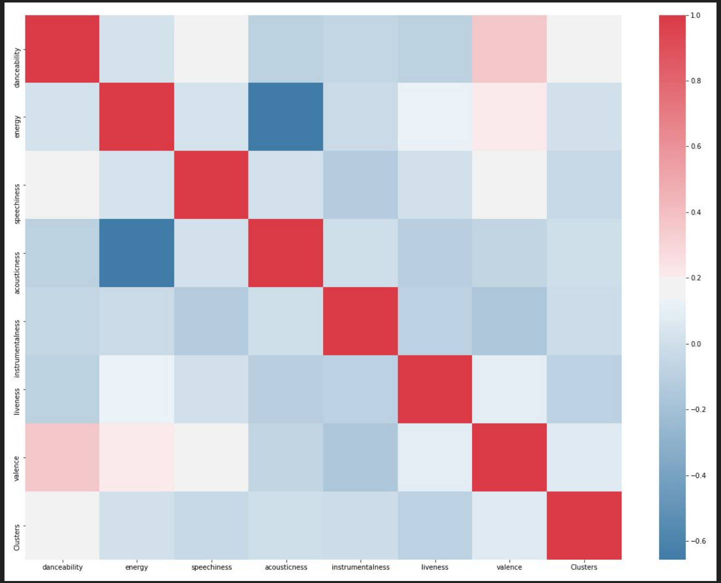
665 tracks saved

27 total columns used

Track and audio features merged via track_id

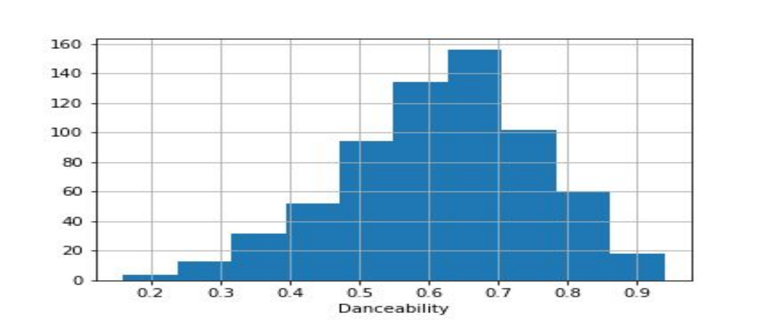
General Data Exploration

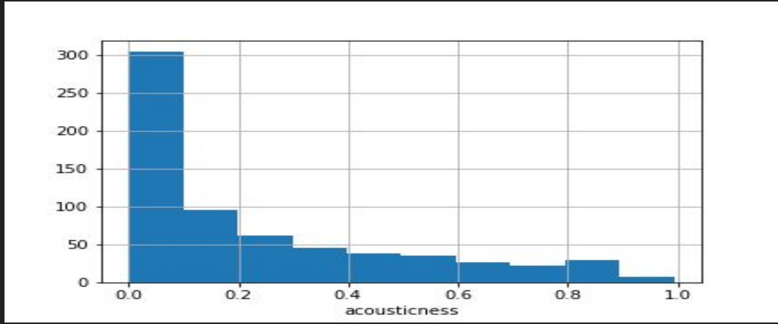


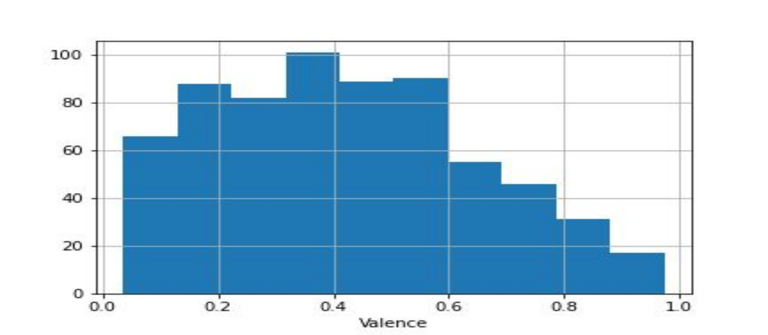


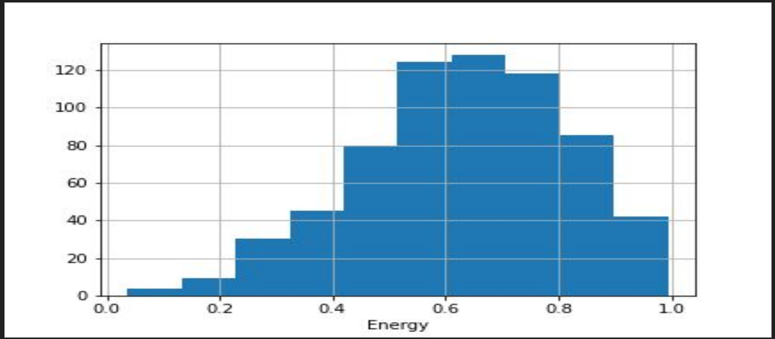
Audio Data Exploration

Audio Features scaled from 0-1



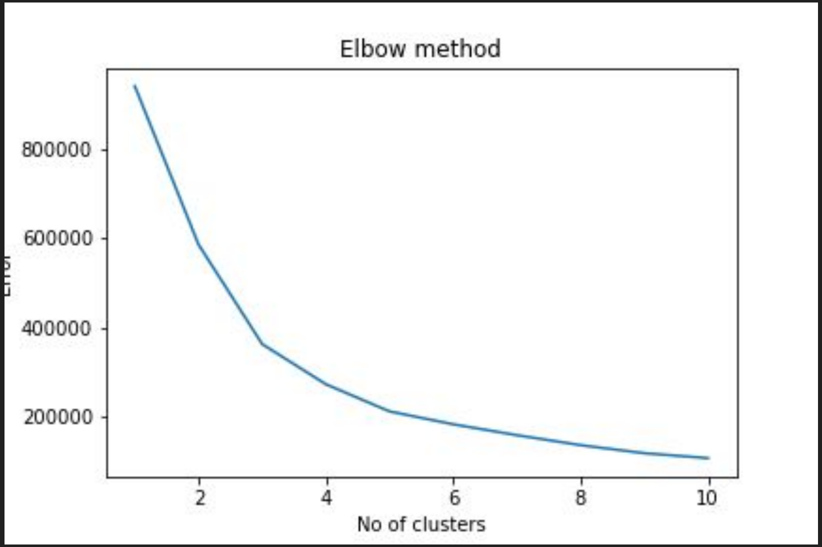






Cluster Analysis

After breaking library into **three** clusters via K-Means and elbow plot heuristic, we can see a clearer picture of classes of tracks



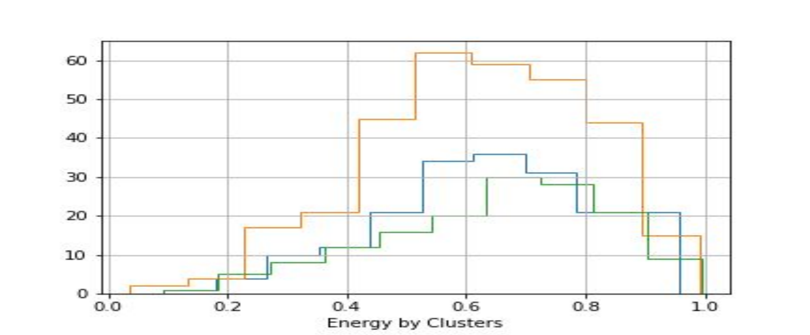
Spotify API codes mode 1 as major key (happy) and 0 as minor key (sad). From the clustering, we can see that each cluster is generally major with cluster 0 being the least minor.

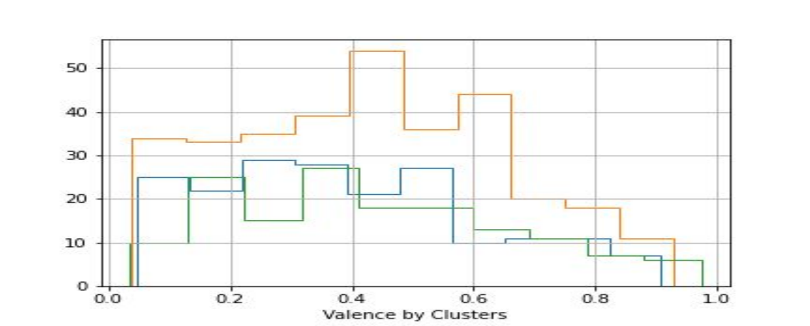
```
Clusters  mode
0         1       113
          0        78
1         1       200
          0       124
2         1        97
          0        53
Name: mode, dtype: int64
```

Cluster Analysis cont

Clusters suggest two ‘happy’ high-energy clusters with one tending to be more popular and one ‘sadder’ clusters that is slower and less upbeat.

Clusters	popularity	valence	loudness	energy	acousticness	danceability
0	49.984293	0.392729	-6.873031	0.638115	0.230026	0.572927
1	54.126543	0.430740	-6.964978	0.621410	0.241708	0.637781
2	5.620000	0.437026	-6.916380	0.644057	0.230724	0.631260





Modeling - Summary

Unsupervised learning - Choosing next track based on how closely it relates to the current one

Scaling required as some columns have larger ranges than others

Recommendations - Attempted TF-IDF but eventually settled on K-Means with size of recommendation playlist = number of neighbors

Tools: Python - Scikit-Learn NearestNeighbors and preprocessing modules

Modeling - Simple, genre-based

Spotify associates genres to artist level very generously (not unusual for 5+ genres for an artist)

Simple model for measuring similarity applied here to artist genres can be effective in producing related artists

However this model is weak for tracks since similar artists will produce similar recommendations as they would have the same genres

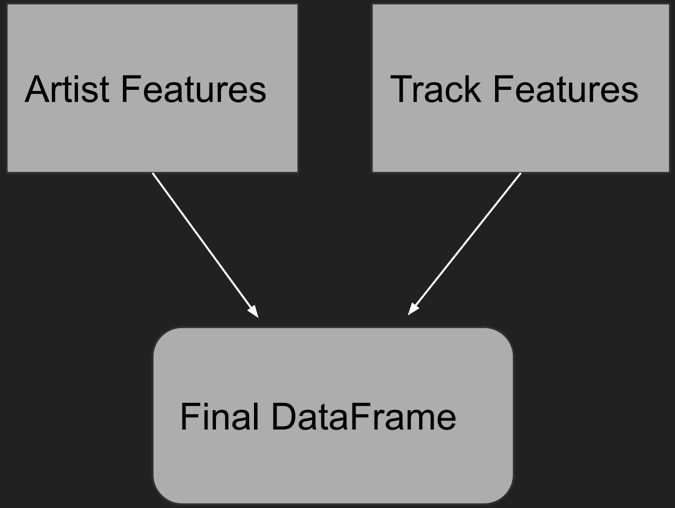
Example to the right seeded with rap artist, the tracks are fairly different in sound despite all artists being rappers

```
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
```

Modeling - Track Features recommendations

KNN to neighbor similar tracks based on euclidean distance between each track's audio features so track features are fully factored into recommendation decision-making

Some artist dataframe columns included as part of the final frame for model fitting



Modeling - Tuning

Different scaling methods used as these influenced recommendations

- MinMax Scaling
- Standard Scaling
- Absolute Max Scaling

Clustering into 3 groups generated previous in the EDA step also leveraged as a column

Scaling shapes recommendations

MinMax

```
#Rock Bottom (Generic Pop)
print_similar_tracks_minmax(id=580)
```

```
The Lumineers - Cleopatra
DEAN - D (Half Moon)
Twin Shadow - Saturdays (feat. HAIM)
Aries - CAROUSEL
Calvin Harris - I Need Your Love (feat. Ellie Goulding)
Fall Out Boy - Wilson (Expensive Mistakes)
NOTD - So Close
Lola Marsh - You're Mine
Just A Gent - Limelight (feat. R O Z E S)
```

AbsMax

```
#Rock Bottom (Generic Pop)
print_similar_tracks_max(id=580)
```

```
The Lumineers - Cleopatra
DEAN - D (Half Moon)
Twin Shadow - Saturdays (feat. HAIM)
Calvin Harris - I Need Your Love (feat. Ellie Goulding)
Aries - CAROUSEL
NOTD - So Close
Fall Out Boy - Wilson (Expensive Mistakes)
Mura Masa - 1 Night (feat. Charli XCX)
Just A Gent - Limelight (feat. R O Z E S)
```

Standard Scaling

```
#Rock Bottom (Generic Pop)
print_similar_tracks_ss(id=580)
```

```
The Lumineers - Cleopatra
Twin Shadow - Saturdays (feat. HAIM)
DEAN - D (Half Moon)
Fall Out Boy - Wilson (Expensive Mistakes)
NOTD - So Close
Aries - CAROUSEL
Lola Marsh - You're Mine
Mr. Probz - Nothing Really Matters - Afrojack Remix
Gryffin - Winnebago (feat. Quinn XCII & Daniel Wilson)
```

Takeaway / limitation discussion

Future project ideas

- Lyrical content for tracks which are more ‘speech-y’
- Designing some looping so larger recommendation requests don’t cycle through entire library

Limited Data was biggest hindrance

- I could only use what was open to me via API endpoints
- Possibilities can be expanded in the future with access to usage metrics