MUSIC RECOMMENDER

Daron Marino

About me

- I am a Data Scientist with a background in Experimental Psychology and Crisis Response.
- I have an affinity for studying human behavior.
- Before my academic and professional careers, I was a touring musician in a metal band.

Overview

- The data consists of 174,389 Spotify songs across 3232 genres.

- The overall goal of the project was to build a functioning recommender system that can provide accurate recommendations based on a preferred artist.

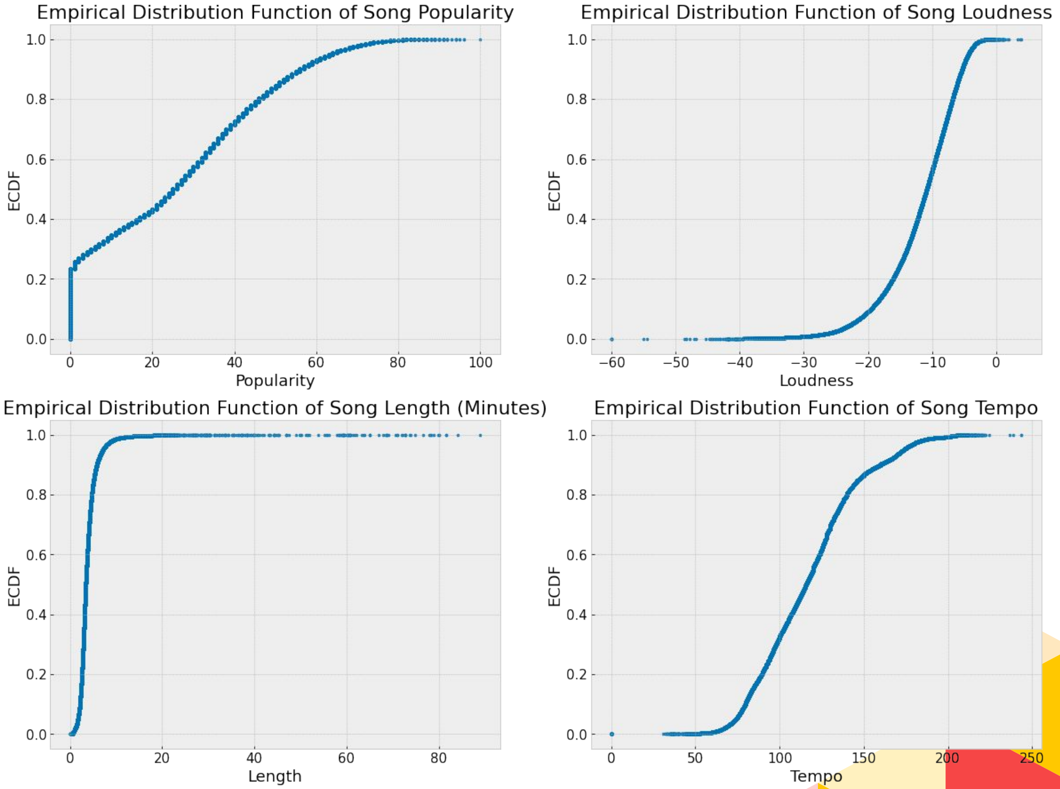
- Out of the distance measures tested, cosine similarity performed the best.

- A web app was developed to demonstrate the use of the recommender.

Data Source: https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks

Exploratory Data Analysis

- The full dataset contains 174,389 songs and 19 features (acousticness, danceability, liveness, popularity, etc.)
- Earliest song released 1920
- Most recent song released 2021



Empirical Distribution Function of Song Popularity

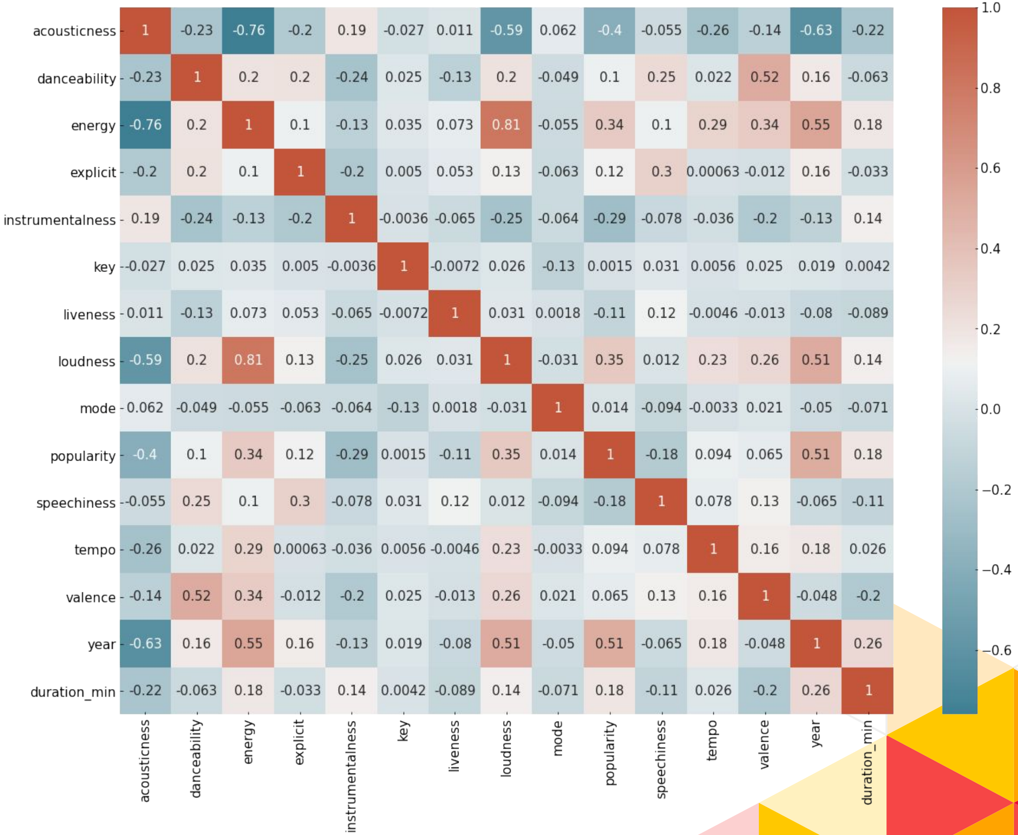
Empirical Distribution Function of Song Loudness

Empirical Distribution Function of Song Length (Minutes)

Empirical Distribution Function of Song Tempo

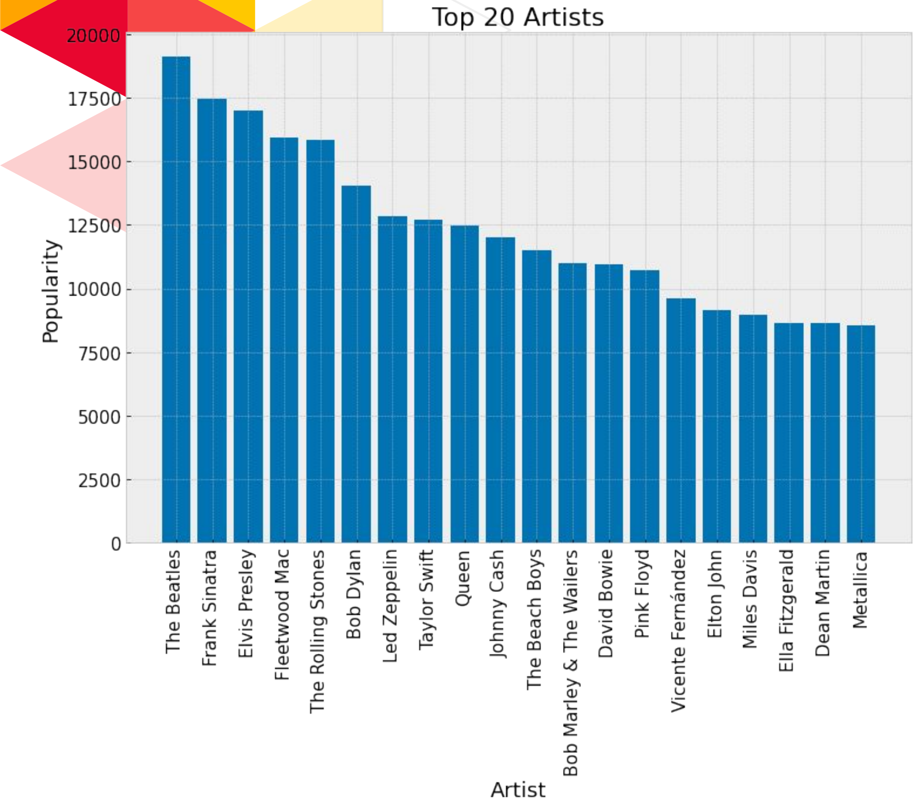
Exploratory Data Analysis

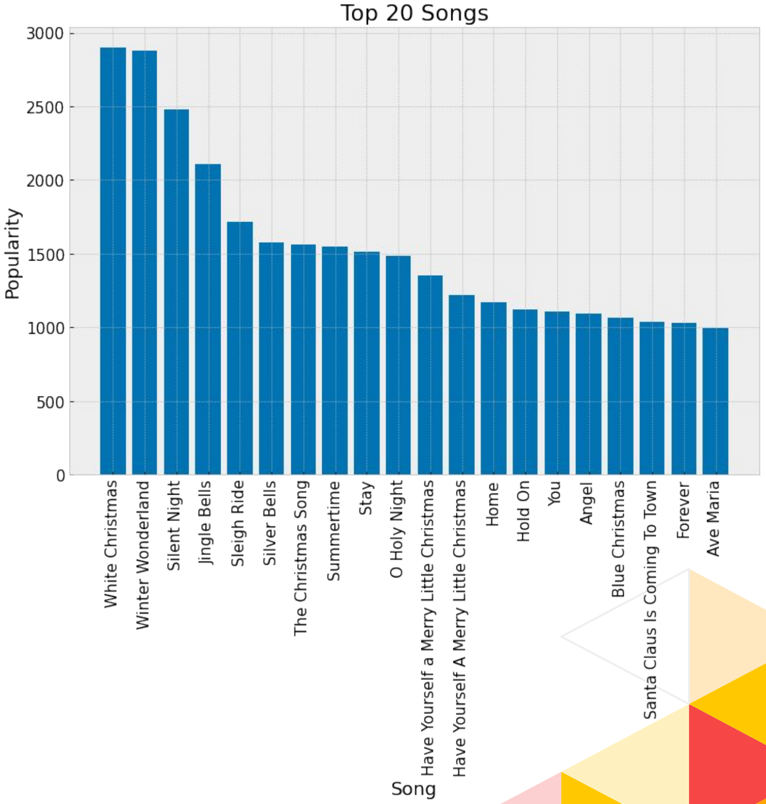
- Valence and danceability: **0.52**
- Energy and year: **0.55**
- Loudness and year: **0.51**
- Loudness and popularity: **0.35**
- Loudness and energy: **0.81**
- Year and popularity: **0.51**



	acousticness	danceability	energy	explicit	instrumentalness	key	liveness	loudness	mode	popularity	speechiness	tempo	valence	year	duration_min
acousticness	1	-0.23	-0.76	-0.2	0.19	-0.027	0.011	-0.59	0.062	-0.4	-0.055	-0.26	-0.14	-0.63	-0.22
danceability	-0.23	1	0.2	0.2	-0.24	0.025	-0.13	0.2	-0.049	0.1	0.25	0.022	0.52	0.16	-0.063
energy	-0.76	0.2	1	0.1	-0.13	0.035	0.073	0.81	-0.055	0.34	0.1	0.29	0.34	0.55	0.18
explicit	-0.2	0.2	0.1	1	-0.2	0.005	0.053	0.13	-0.063	0.12	0.3	0.00063	-0.012	0.16	-0.033
instrumentalness	0.19	-0.24	-0.13	-0.2	1	-0.0036	-0.065	-0.25	-0.064	-0.29	-0.078	-0.036	-0.2	-0.13	0.14
key	-0.027	0.025	0.035	0.005	-0.0036	1	-0.0072	0.026	-0.13	0.0015	0.031	0.0056	0.025	0.019	0.0042
liveness	0.011	-0.13	0.073	0.053	-0.065	-0.0072	1	0.031	0.0018	-0.11	0.12	-0.0046	-0.013	-0.08	-0.089
loudness	-0.59	0.2	0.81	0.13	-0.25	0.026	0.031	1	-0.031	0.35	0.012	0.23	0.26	0.51	0.14
mode	0.062	-0.049	-0.055	-0.063	-0.064	-0.13	0.0018	-0.031	1	0.014	-0.094	-0.0033	0.021	-0.05	-0.071
popularity	-0.4	0.1	0.34	0.12	-0.29	0.0015	-0.11	0.35	0.014	1	-0.18	0.094	0.065	0.51	0.18
speechiness	-0.055	0.25	0.1	0.3	-0.078	0.031	0.12	0.012	-0.094	-0.18	1	0.078	0.13	-0.065	-0.11
tempo	-0.26	0.022	0.29	0.00063	-0.036	0.0056	-0.0046	0.23	-0.0033	0.094	0.078	1	0.16	0.18	0.026
valence	-0.14	0.52	0.34	-0.012	-0.2	0.025	-0.013	0.26	0.021	0.065	0.13	0.16	1	-0.048	-0.2
year	-0.63	0.16	0.55	0.16	-0.13	0.019	-0.08	0.51	-0.05	0.51	-0.065	0.18	-0.048	1	0.26
duration_min	-0.22	-0.063	0.18	-0.033	0.14	0.0042	-0.089	0.14	-0.071	0.18	-0.11	0.026	-0.2	0.26	1

Exploratory Data Analysis





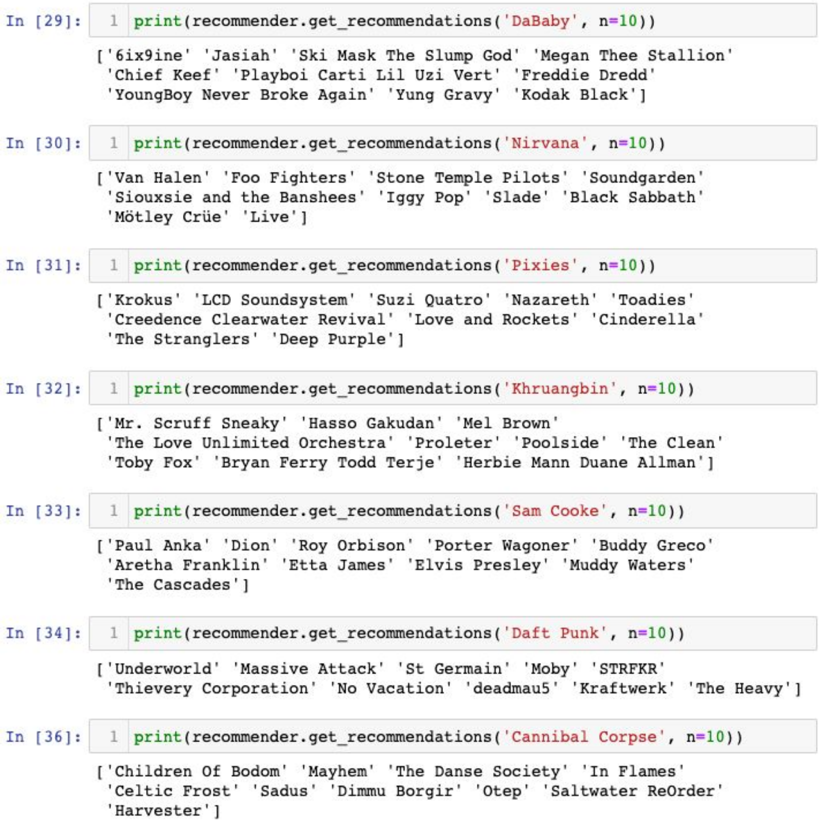
Setting Up the Recommender

- The data were organized by song, but I’m interested in creating an artist recommender.
- I needed to figure out a way to represent each artist’s characteristics via aggregation given most artists appear several times in the dataset.
  - Data were grouped by the artist feature and then aggregated via mean.
- Year and Key features dropped from the dataset.
  - The remaining features were normalized.

Setting Up the Recommender

- The recommender is based on content-filtering. Meaning the recommendations are made based on an item's associated features.
- Similarity between artists calculated via distance metric, top items are sorted by similarity, and recommendations are provided.
  - This project uses cosine similarity for the distance metric.

Examples of Model Performance

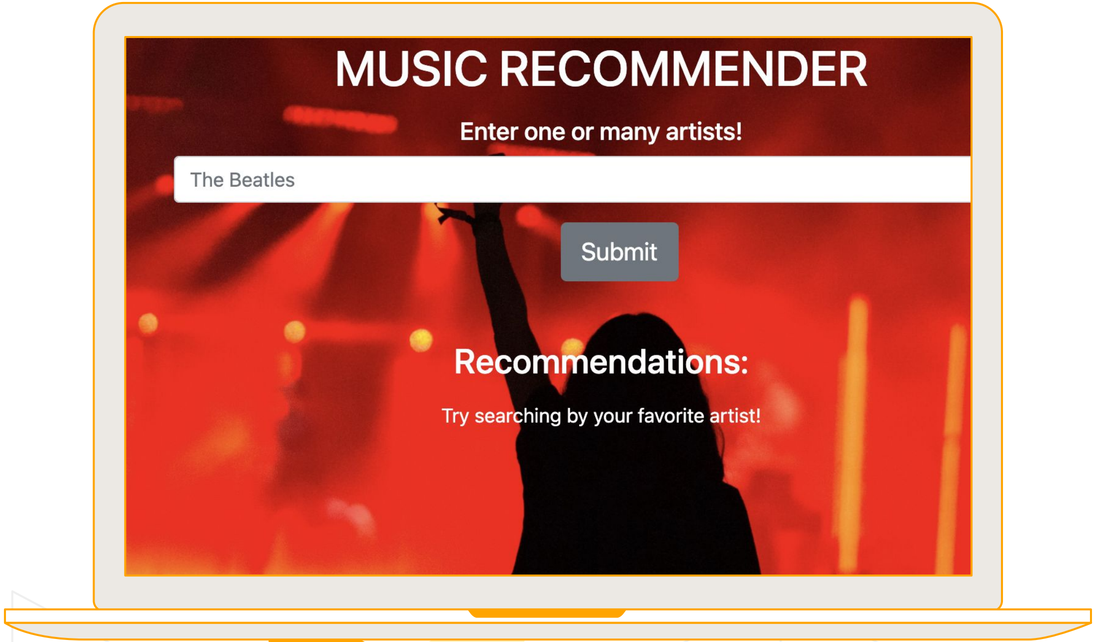


```
In [29]:  print(recommender.get_recommendations('DaBaby', n=10))

          ['6ix9ine' 'Jasiah' 'Ski Mask The Slump God' 'Megan Thee Stallion'
           'Chief Keef' 'Playboi Carti Lil Uzi Vert' 'Freddie Dredd'
           'YoungBoy Never Broke Again' 'Yung Gravy' 'Kodak Black']

In [30]:  print(recommender.get_recommendations('Nirvana', n=10))

          ['Van Halen' 'Foo Fighters' 'Stone Temple Pilots' 'Soundgarden'
           'Siouxsie and the Banshees' 'Iggy Pop' 'Slade' 'Black Sabbath'
           'Mötley Crüe' 'Live']

In [31]:  print(recommender.get_recommendations('Pixies', n=10))

          ['Krokus' 'LCD Soundsystem' 'Suzi Quatro' 'Nazareth' 'Toadies'
           'Creedence Clearwater Revival' 'Love and Rockets' 'Cinderella'
           'The Stranglers' 'Deep Purple']

In [32]:  print(recommender.get_recommendations('Khruangbin', n=10))

          ['Mr. Scruff Sneaky' 'Hasso Gakudan' 'Mel Brown'
           'The Love Unlimited Orchestra' 'Proleter' 'Poolside' 'The Clean'
           'Toby Fox' 'Bryan Ferry Todd Terje' 'Herbie Mann Duane Allman']

In [33]:  print(recommender.get_recommendations('Sam Cooke', n=10))

          ['Paul Anka' 'Dion' 'Roy Orbison' 'Porter Wagoner' 'Buddy Greco'
           'Aretha Franklin' 'Etta James' 'Elvis Presley' 'Muddy Waters'
           'The Cascades']

In [34]:  print(recommender.get_recommendations('Daft Punk', n=10))

          ['Underworld' 'Massive Attack' 'St Germain' 'Moby' 'STRFKR'
           'Thievery Corporation' 'No Vacation' 'deadmau5' 'Kraftwerk' 'The Heavy']

In [36]:  print(recommender.get_recommendations('Cannibal Corpse', n=10))

          ['Children Of Bodom' 'Mayhem' 'The Danse Society' 'In Flames'
           'Celtic Frost' 'Sadus' 'Dimmu Borgir' 'Otep' 'Saltwater ReOrder'
           'Harvester']
```

- Testing to see how the model performs when given artists from varying genres.

The Flask App

- Flask app constructed using HTML and CSS
- Type artist in search bar and review recommendations
- Spotify API integration



MUSIC RECOMMENDER

Enter one or many artists!

The Beatles

Recommendations:

Try searching by your favorite artist!

Future Directions

- Get additional distance metrics working (e.g., jaccard).
  - Potentially use to make an ensemble recommender.
- Additional tuning to ensure most important features have appropriate weights.
- Introduce NLP component.
- Make Improvements to the Flask app.

Data Science Stack

python™
NumPy
pandas
matplotlib
seaborn
Scikit learn
Flask
amazon web services

Thank You!

Any questions?

You can find me at

- linkedin.com/in/daronmarino
- github.com/Daron-Marino
- marino.daron@gmail.com