MyVibes: A Musical Recommender

Robert Dowd

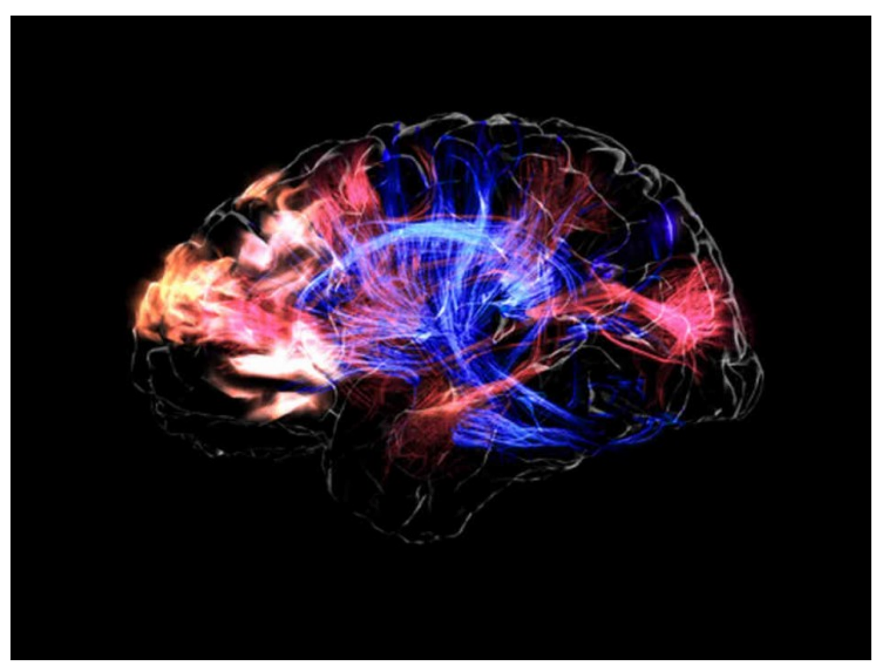
github.com/rdowd003

/in/rdowd003

rdowd003@hotmail.com

Introduction

Advanced cognition in humans is an astounding evolutionary feat and its driving forces continue to persist as a mystery. Despite our ignorance of the underlying mechanisms, the ability to transform electrical (neural) signals into something with greater meaning is what makes our lives so interesting, and of course, so unique. Enhancing these subjective cognitive experiences, in the context of music, is the primary motivation for this project.



Mickey Hart. *This is your brain on music. http://libguides.humboldt.edu/c.php?g=303882&p=2030458. 09/01/2018.*

For musical recommender systems, there are many ways to create a starting point. Favorite song, favorite artist, era, existing playlists, etc. While this project does eventually utilize similar categorizers for filtering, the basis of its recommendations are driven by connections to the deeper, more complex elements of music. This unique recommender allows users to choose a cluster, or group "vibe", from which new songs are recommended.

Objectives

1. Develop a versatile recommender algorithm that can easily be used by any kind of listener.
2. Launch a simple, interactive web-based environment for users to experiment with the recommender system.

Methods

Data processing and clustering

Data consisted of 14, proprietary audio-features, sourced through the Spotify API for a public playlist of 10,000 tracks. The playlist includes a diverse set of musical styles, artists, genres, and thus, a wide range of values for each feature (figure 1). To remove colinearity among features, the data set was first standardized, and then transformed using Principle Component Analysis.

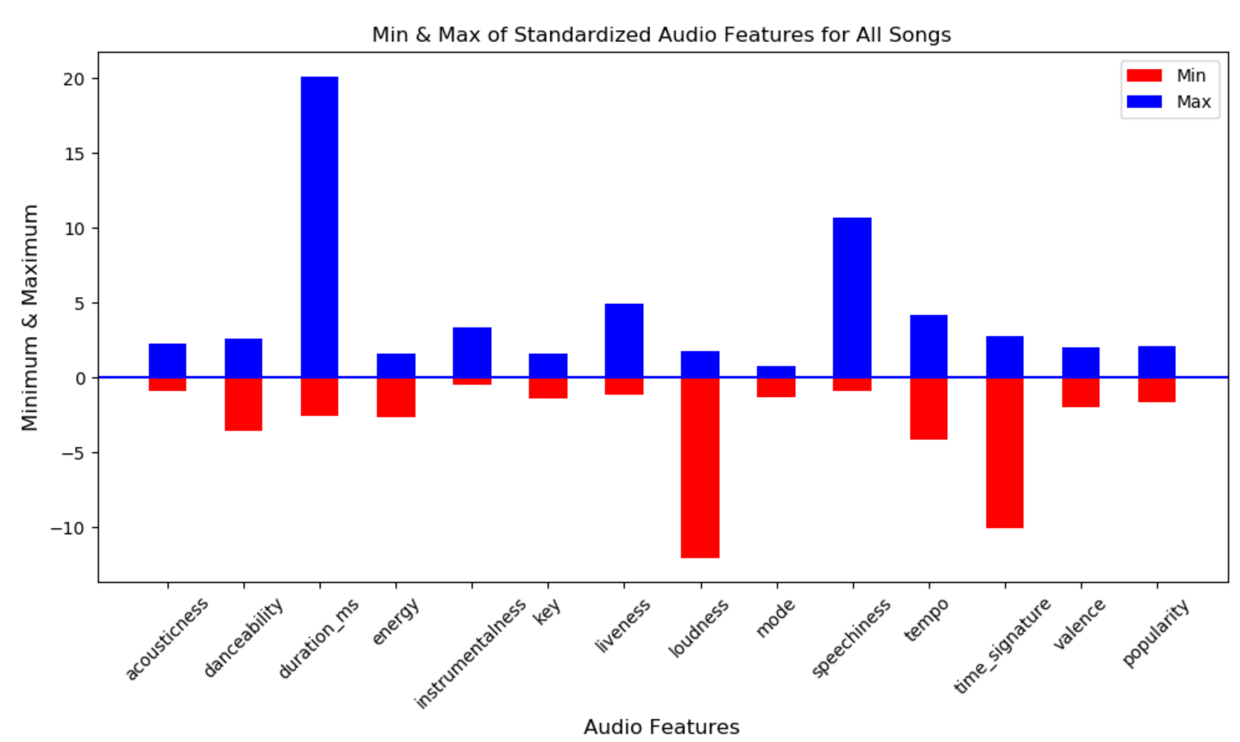


Figure 1. Max-min range of audio-feature values for entire, 10,000 track dataset.

A Kmeans clustering algorithm was used to group the tracks by audio features, setting the clusters up for the recommendation process.

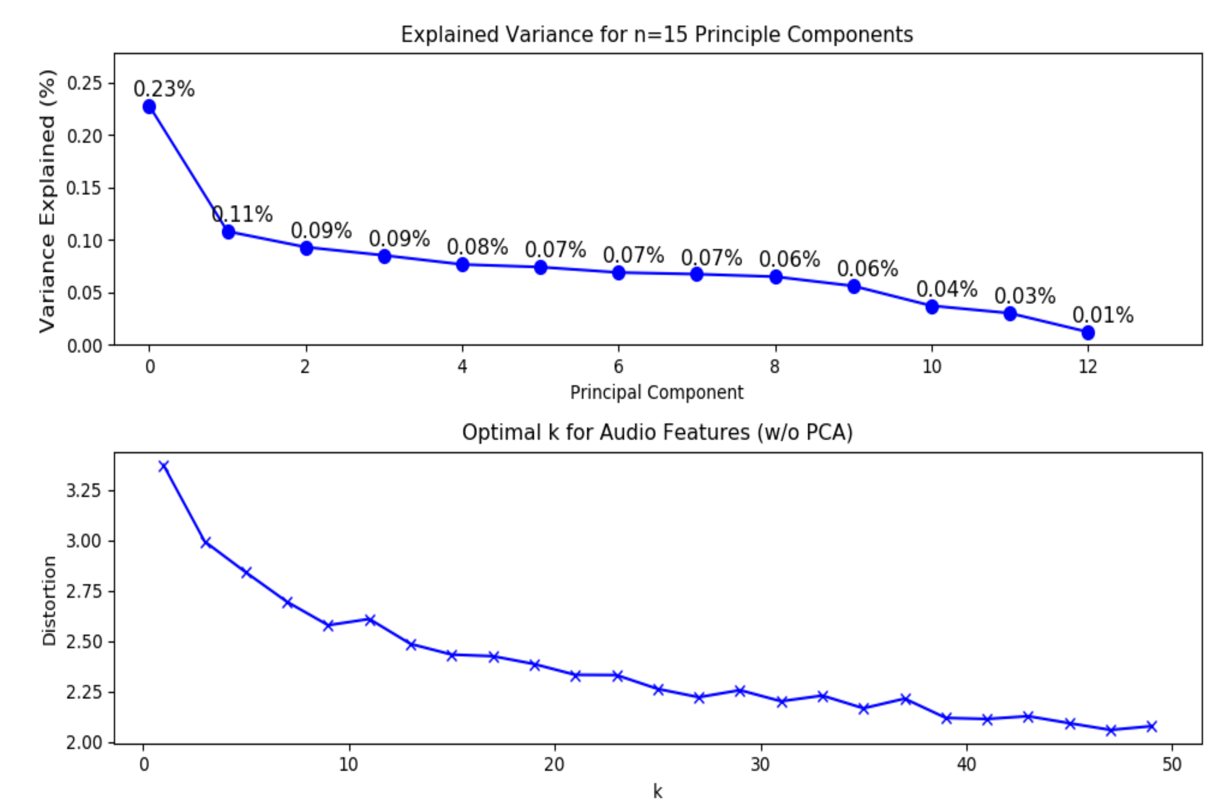
Recommending

The MyVibes recommender was deployed using a Flask App, and proceeds as follows:

1. *Preview a random selection of songs, 1 from each of the 10 clusters.*
2. *Select the cluster, or the "vibe" to which the strongest connection was felt.*
3. *Select an audio feature to prioritize, and whether more, or less popular (according to Spotify counts) songs should be recommended.*

Results

Ten principle components were sufficient to explain 90% of the variance, and were used for Kmeans analysis (figure 2). Additionally, 10 clusters were set, as this number gave the largest decrease in distortion while satisfying the "reasonable time" requirement of using a recommender interactively.



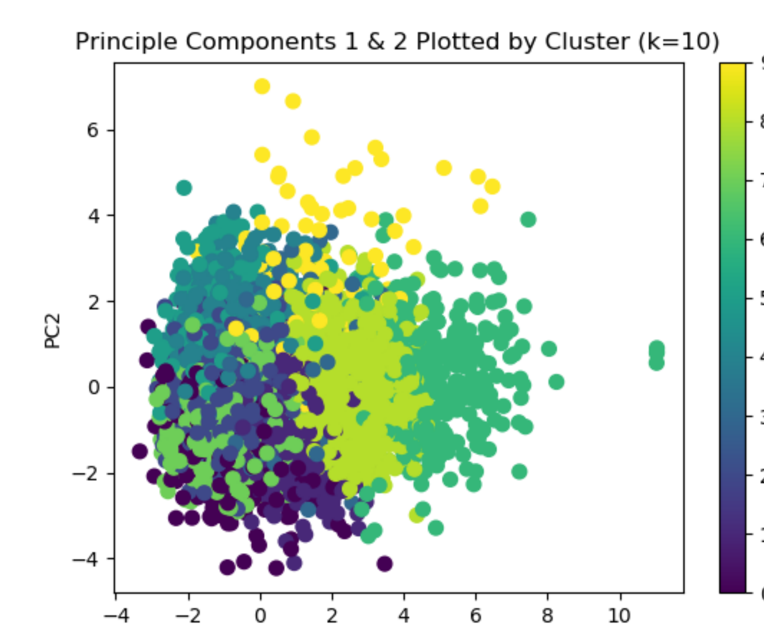


Figure 2. PCA transformations and clustering.

Top: Scree plot (to determine optimal principle components) and elbow plot (optimal k, clusters).

Bottom: First two components plotted against one another, together explaining roughly 34% of the data's variance.

Average audio features varied considerably between clusters, as expected. Cluster 6 and 8 were the furthest apart in Euclidean Distance (figure 3).

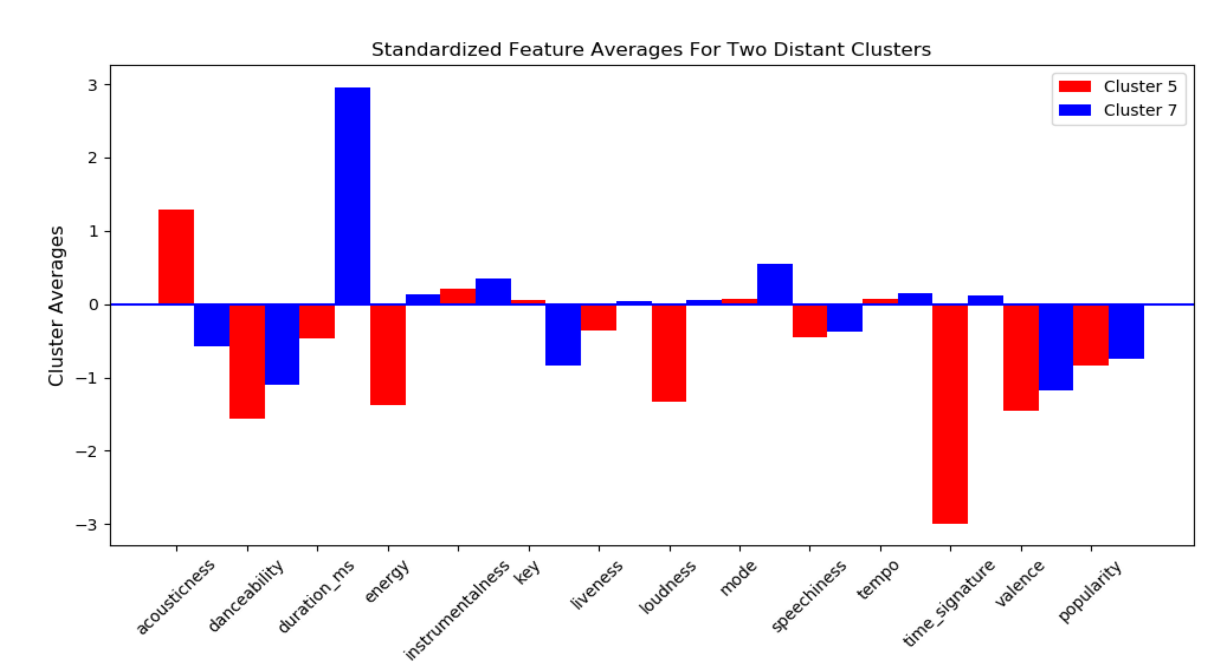


Figure 3. Cluster 6 and cluster 7 audio features (average)

Discussion

Recommendations and Limitations

Ultimately, this musical recommender achieves the goal of delivering individualized song recommendations in a fast and efficient manner. Limitations included a lack of access to the raw wave-derived audio features, having to cold-start all users, and a lack of metadata for tracks that could potentially be used to filter features or conduct an NLP analysis.

Future Directions

- Allow users to sign in with spotify credentials and gather information about top artists, top played songs, etc. , to enhance recommendations.
- With access to raw audio features, conduct a more thorough feature-engineering analysis to improve the clustering
- Store information for each user and make response-based updates.

References

1. *T. Bertin-Mahieux. Large-Scale Pattern Discovery in Music. PhD thesis, Columbia University, February 2013.*
2. *Spotify API (code available on github.com/rdowd003/Capstone-3/src*
3. *Flask, Pandas, Spark,sklearn documentation*