# Automated Playlist Generation

Kade Keith, Demetrios Fassois
{kade, dimifass}@stanford.edu
CS 229, Fall 2017

## MOTIVATION

With the growth of music streaming services, there are now more songs than ever at music listeners fingertips. Because of this growth, the art of constructing playlists has become increasingly challenging, and discovering new music the in the expanse of choices is a daunting task. We explore multiple approaches to automatic playlist generation that, given a few songs as a seed, can generate or help a user complete their own playlists based on lyrical and audio features. These models are trained on and evaluated against human-generated playlists.

## DATA & FEATURES

Our data comes from the Million Song Dataset and musiXmatch lyrics collection. We augment those songs with info from Spotify. Our playlists are from Spotify.

**Raw features:**

- Tempo, Timbre, Danceability, Energy, Loudness, Speechiness, Acousticness, Instrumentalness, Liveness, Valence (Audio Features)
- Popularity, Year (Song Metadata)
- Lyrics in bag-of-words format

**Derived features:**

- TODO (Stuff from Timbre)
- TODO (LDA)
- TODO (Sentiment)

We chose our features based on what we found important while making our own playlists, and thus what we believe is important to others.

## MODELS

### Classification-based

Our primary approach was to think of selecting songs for a playlist as a classificaion problem.

**Logistic Regression**

Lorem ipsum dolor sit amet, mutat quodsi usu ei, ne quis nullam eruditi pro. Doming menandri ex sea, mel et sonet corpora phaedrum, per at veri porro neglegentur.

**TODO**

Lorem ipsum dolor sit amet, mutat quodsi usu ei, ne quis nullam eruditi pro. Doming menandri ex sea, mel et sonet corpora phaedrum, per at veri porro neglegentur.

### Graph-based

An interesting alternative is to think of songs as nodes on a completely connected graph.

**k-Nearest Neighbors**

We represent each song as a point in n-dimensional space according to our normalized features. Then we select the next songs for that playlist based on proximity to the seed.

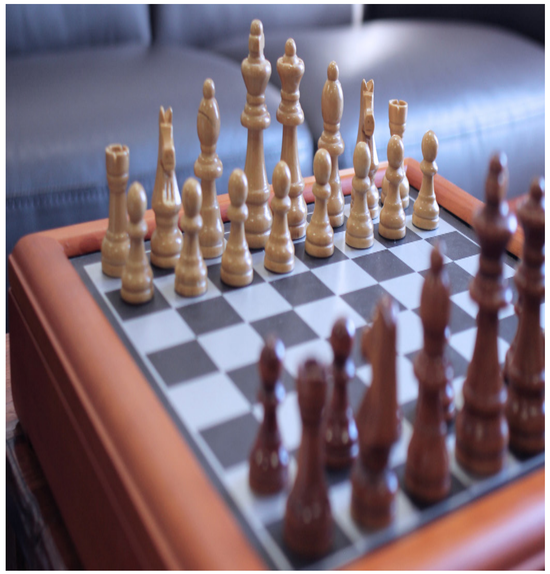
**Shortest path (of length k)**

Given two songs, we construct a playlist as the shortest path of length k between them. This is particularly intriguing if the two seed songs are very different. Via dynamic programming, the shortest path to node v of length k is:

$$path[v][k] = \min_u \left( path[u][k-1] + dist[u][v] \right)$$

## RESULTS

Lorem ipsum dolor sit amet, mutat quodsi usu ei, ne quis nullam eruditi pro. Doming menandri ex sea, mel et sonet corpora phaedrum, per at veri porro neglegentur. Imperdiet euripidis aliquando vis ea, ut nonumy quaerendum sed, at noluisse corrumpit vix. Sed reque tamquam deleniti ut, quo te simul decore convenire. Vix facilisis cotidieque ad, cu simul accusamus pro. Ne invenire contentiones vim.



## DISCUSSION

Lorem ipsum dolor sit amet, mutat quodsi usu ei, ne quis nullam eruditi pro. Doming menandri ex sea, mel et sonet corpora phaedrum, per at veri porro neglegentur. Imperdiet euripidis aliquando vis ea, ut nonumy quaerendum sed, at noluisse corrumpit vix. Sed reque tamquam deleniti ut, quo te simul decore convenire. Vix facilisis cotidieque ad, cu simul accusamus pro. Ne invenire contentiones vim.

## FUTURE

Lorem ipsum dolor sit amet, mutat quodsi usu ei, ne quis nullam eruditi pro. Doming menandri ex sea, mel et sonet corpora phaedrum, per at veri porro neglegentur.

## REFERENCES

Lorem ipsum dolor sit amet, mutat quodsi usu ei, ne quis nullam eruditi pro. Doming menandri ex sea, mel et sonet corpora phaedrum, per at veri porro neglegentur.