Acquiring New Tastes: Using Deep Learning to Recommend Music Based on my Listening History

Presented by: Apoorv Anand
Prepared for: The University of Chicago MS in Applied Data Science, Machine Learning and Predictive Analytics Final Project
Date: May 23, 2024

Agenda

Problem Statement

Exploring the problem and key assumptions/hypotheses

Diving into the Data

EDA findings and Feature Engineering choices

Crafting a Solution

Examining a baseline model to check over/under fitting + my proposed model

Recommending my New Favorites

Exploring my results and future work potential

The Problem: which songs in the Billboard Top 100 would I enjoy most based on my listening history?

My listening history data

Scrape my listening history data from the past year

Billboard Top 100

The Billboard 100 is one of the most reputable sources for the pulse of the music industry

Spotify Investment

Spotify has invested billions into music recommender systems

Recommend the top 10 songs from the Billboard Top 100 I will enjoy the most based on my listening history

The data comes from scraping the Spotify API for the metadata of my listening history over the past year

Listening History from Previous Year (Requested from Spotify):

- **endTime** (date I listened to the song)
- **Artist Name**
- **Track Name**
- **msPlayed** (how long I listened to the song)

Initial Concerns:

- ~8100 unique track and artist names without useful metadata
- Needed to use API to get **trackID** (a unique indicator by song used by Spotify) and then use **trackID** to get **more metadata**

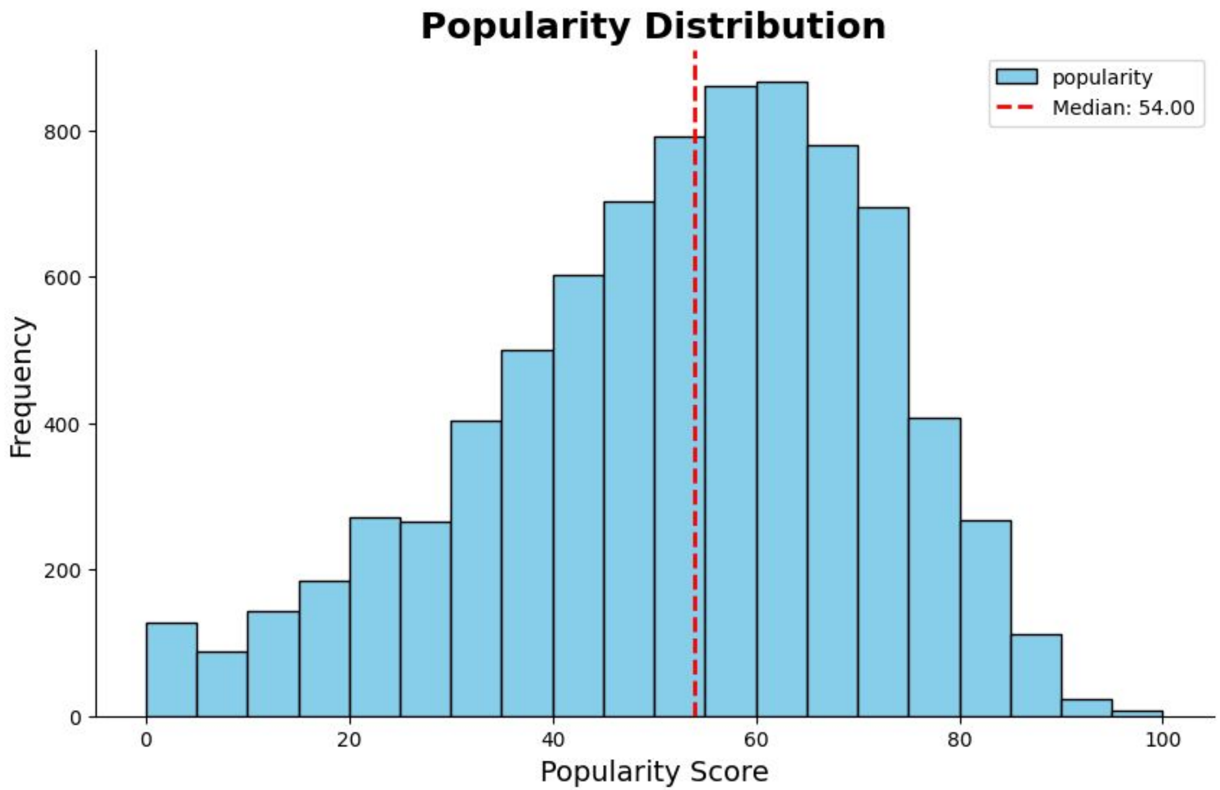
Spotify API

using "Spotipy" package in Python

Metadata

- **Popularity**
- **Danceability**
- **Energy**
- **Key**
- **Loudness**
- **Mode**
- **Acousticness**
- **Instrumental-ness**
- **Liveness**
- **Tempo**

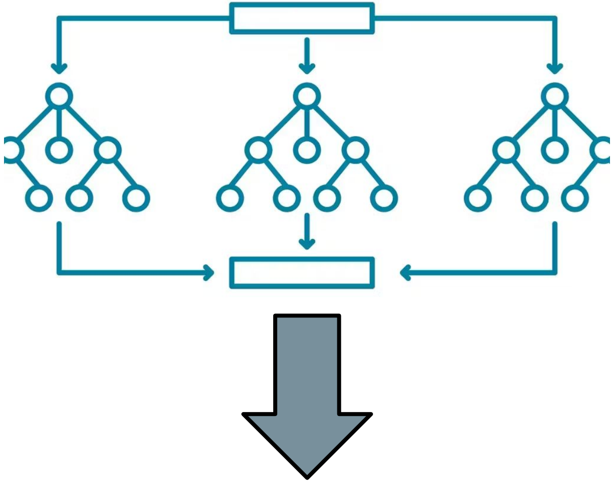
Song popularity stratifies my listening data nicely, making a great target variable when modeling



- **Popularity** is a 1-100 score given to songs based on the total number of streams and recency of streams
- Near-normal distribution makes it a great candidate to be a target variable (over/under **median)**

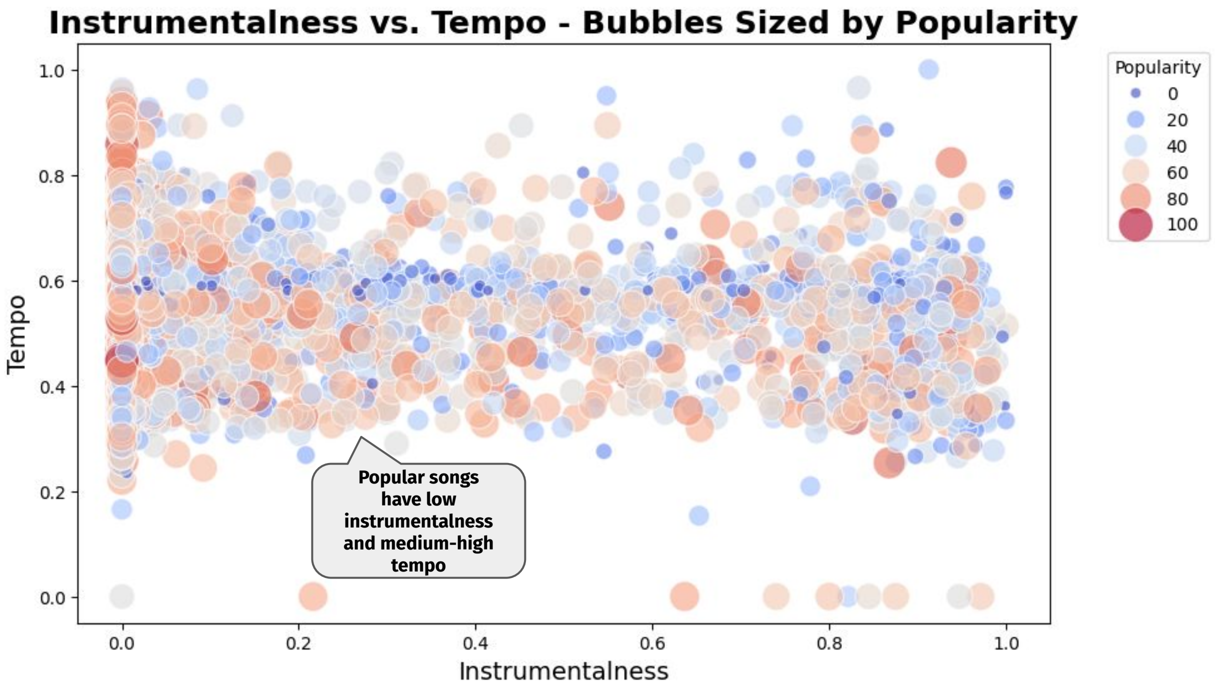
Random Forest feature importance indicates that instrumentalness and tempo impact the popularity of a song

Random Forest Model



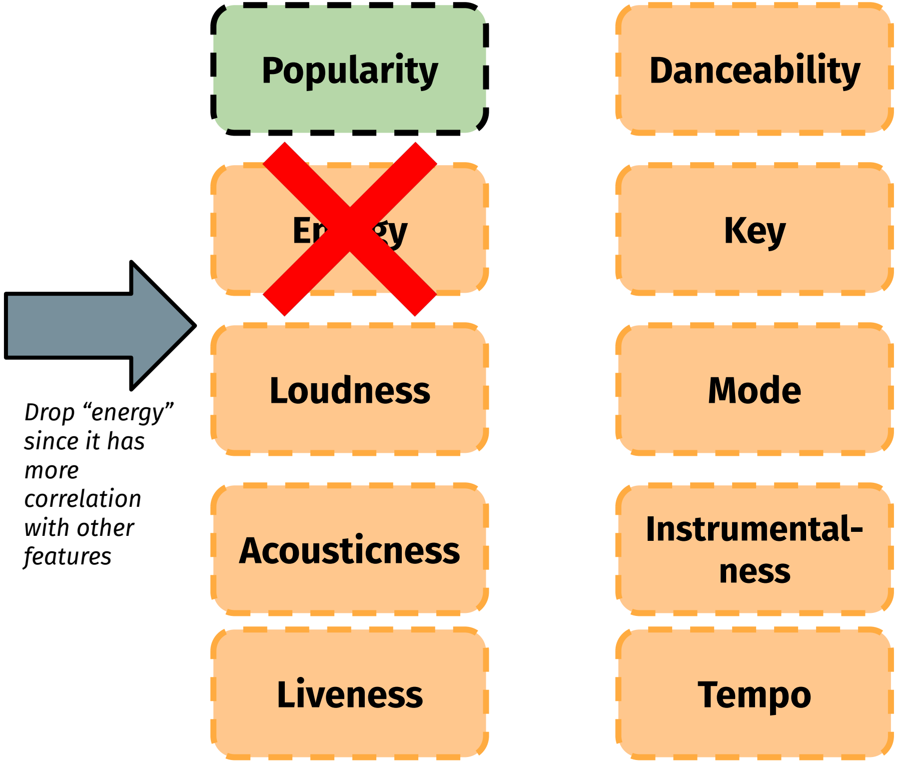
Top 3 Features Ranked by Importance

1. Instrumentalness
2. Tempo
3. Loudness



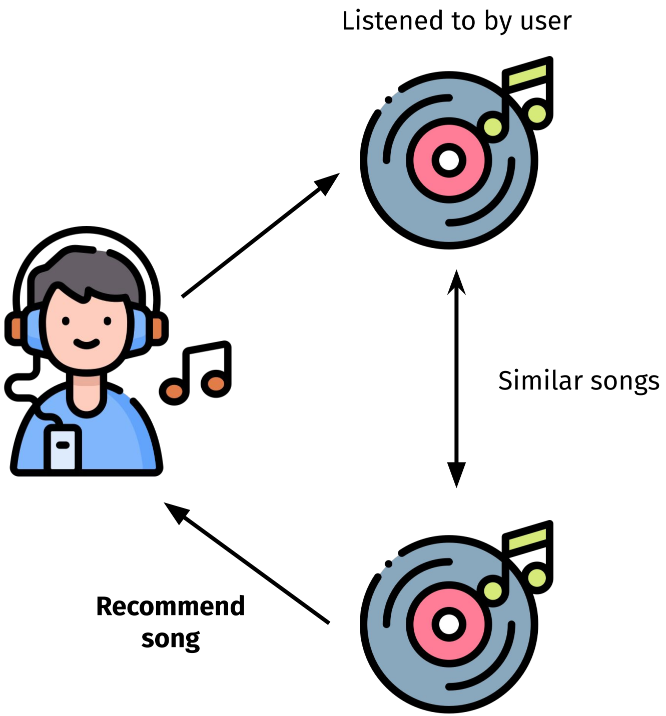
A correlation heatmap shows correlation between loudness and energy, prompting collinearity concerns

Drop "energy" since it has more correlation with other features



For a baseline model, a cosine similarity Content-Based Recommender works well due to its ubiquitous use

Content-Based Recommender Architecture (with cosine similarity)

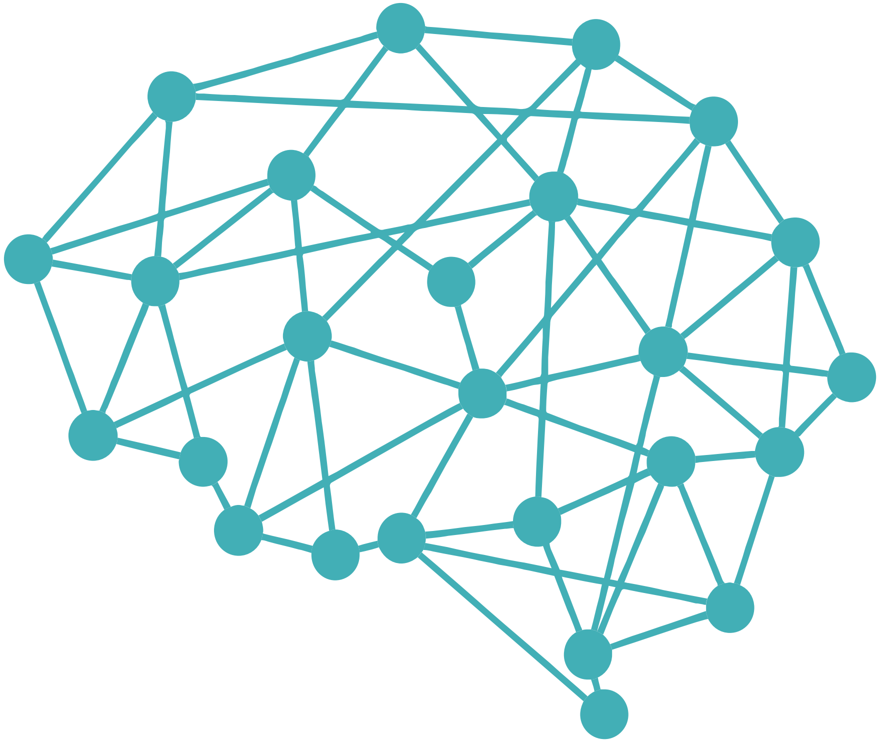


Recommended Song Name	Artist Name	Predicted Like
euphoria	Kendrick Lamar	0.596012
ADVINO	Myke Towers, Bad Bunny	0.591180
Greedy	Tate McRae	0.587361
Enough (Miami)	Cardi B	0.582127
Whatsapp (wassam)	Gunna	0.579810

49% Train Accuracy < 51% Test Accuracy

Slight underfitting if anything, but no overfitting worries

A recommendation system driven by a neural network engine provides additional flexibility and adaptability to the data

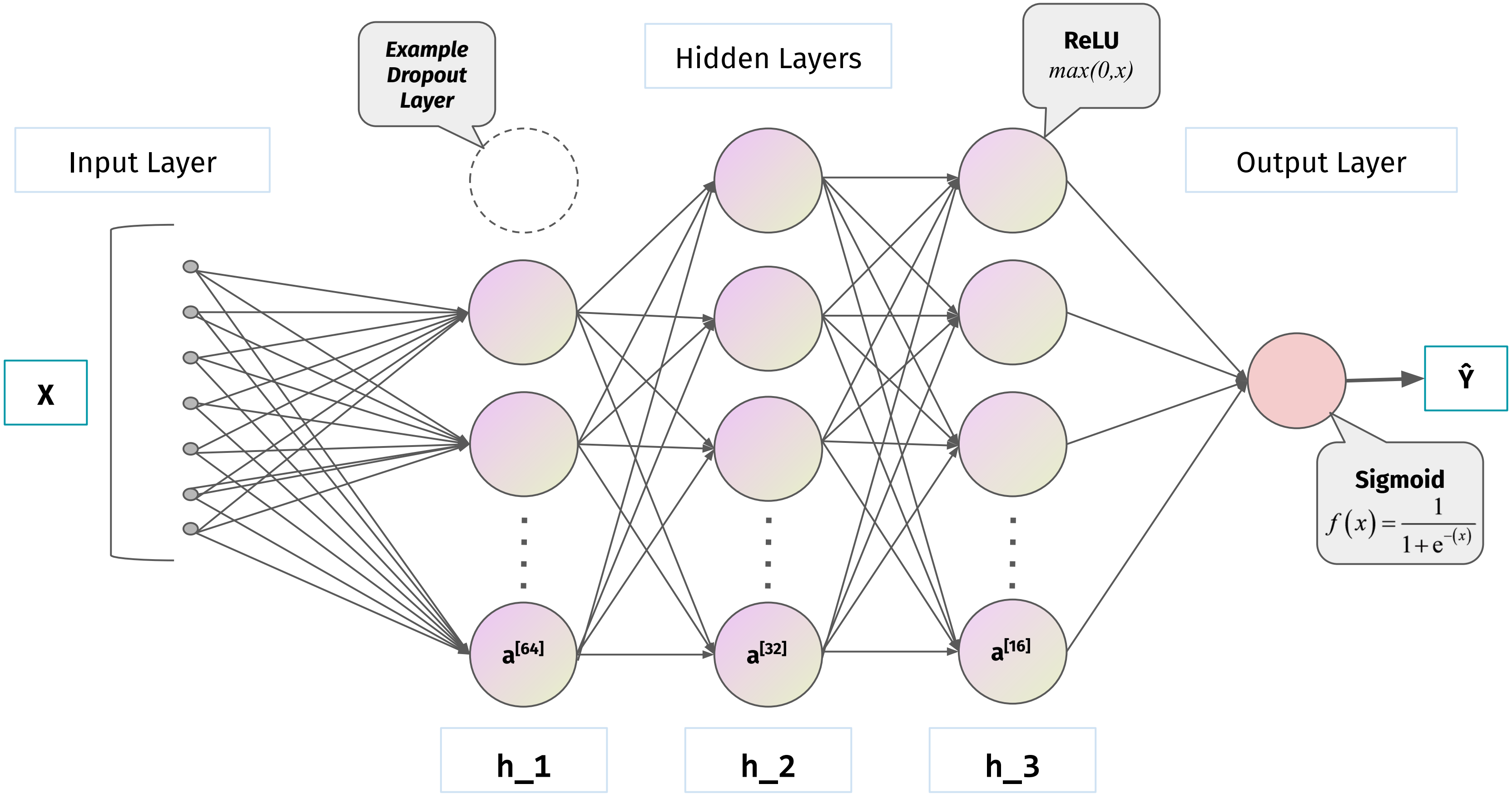


1. A simple neural network can learn non-linear relationships between features and the target variable that a content-based recommender cannot

2. A simple neural network can learn interactions **between** features through its layered structure, giving weights to feature combinations that may affect a recommendation

3. Can train the model to optimize directly to the target variable (popularity) rather than an indirect measure of similarity

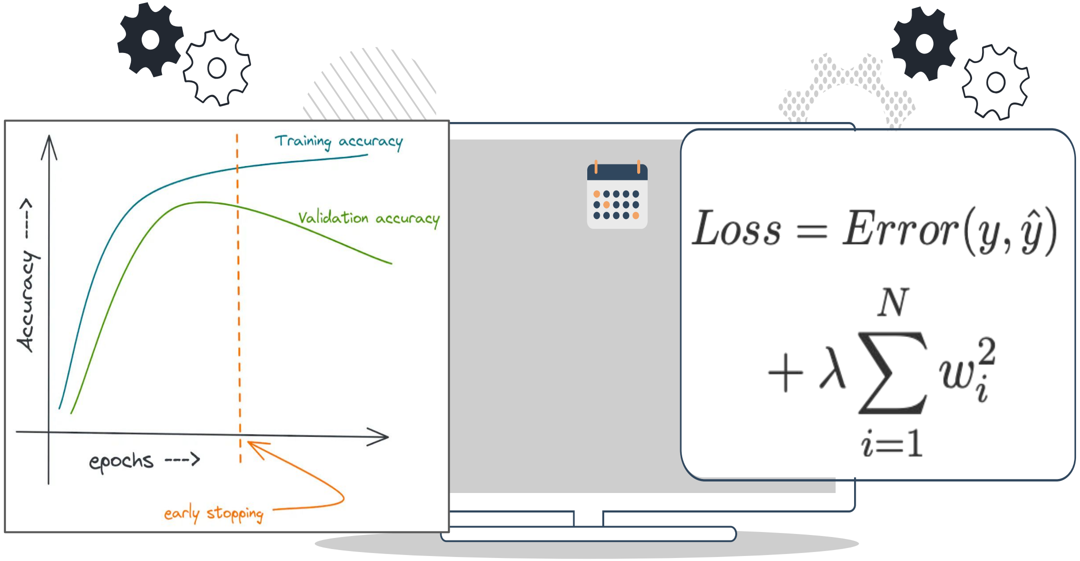
The music recommender model is a simple neural network with three hidden layers and two dropout layers



The neural network recommender employs regularization and early stopping to prevent overfitting and promote efficiency

Early Stopping

- Stops training when validation performance no longer improves
- Computationally efficient
- Stops when marginal improvement over 5 epochs (patience = 5)
- Ensures best observed training outcome

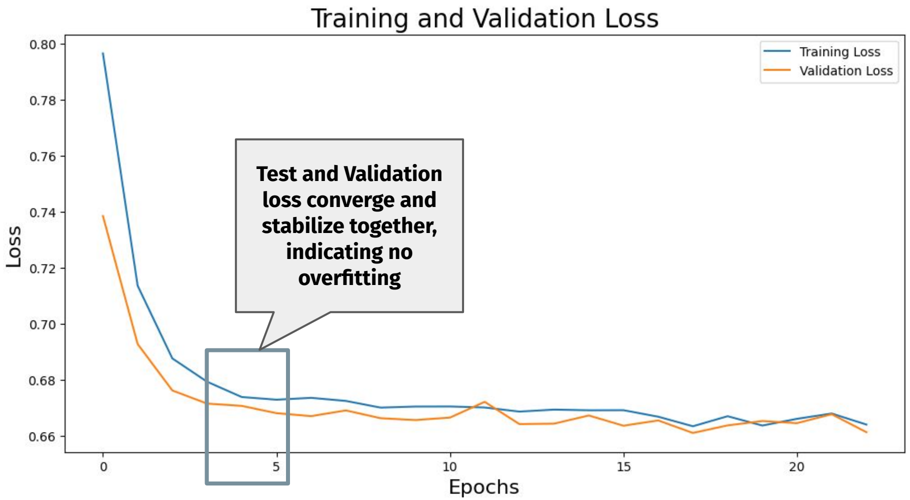


L2 Regularization

- Adds penalty function to loss term that is proportional sum of square weights
- Discourages model from relying on single group or feature
- Lambda shows strength of weights (0.01 in this model)
- Prevents overfitting through penalization

The neural network achieved ~60% accuracy in providing Billboard Top 100 song recommendations by popularity

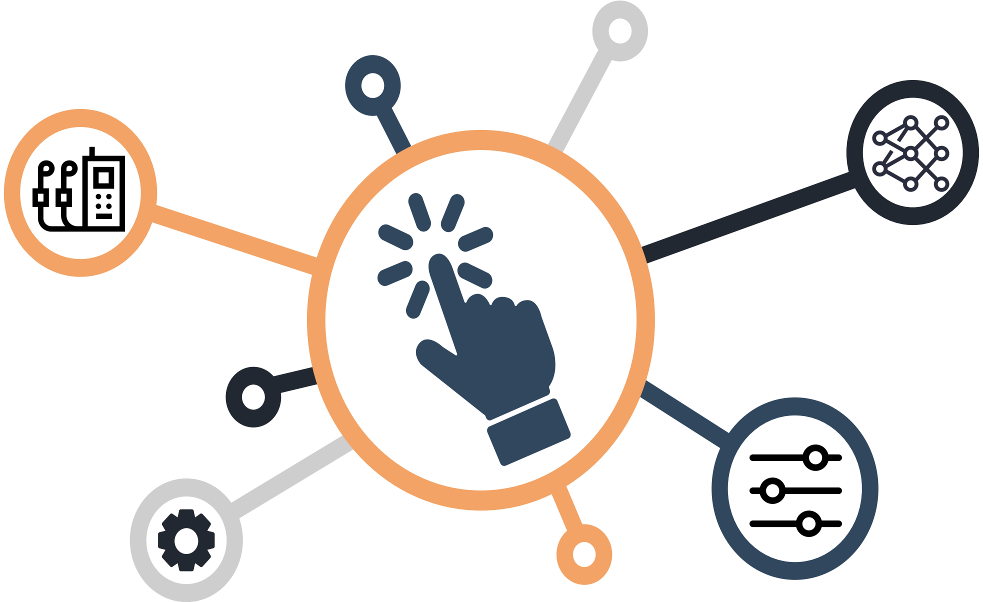
Recommended Song Name	Artist Name	Predicted Like
Never Lose Me	Flo Milli	0.679073
Type S**t	Future, Metro Boomin	0.677588
One of the Girls	The Weeknd, JENNIE, Lily-Rose Depp	0.675264
Bandit	Don Toliver	0.673142
euphoria	Kendrick Lamar	0.671725
I Remember Everything	Zach Bryan, Kacey Musgraves	0.670969
Outskirts	Sam Hunt	0.669854
Down Bad	Taylor Swift	0.669581
Beautiful Things	Benson Boone	0.668866
Push Ups	Drake	0.667559



61% Train Accuracy **>** **60%** Test Accuracy

Figures are close enough to not worry about overfitting

Future work could include gathering more listening data for better training as well as more advanced neural network methods



Additional Listening History

Scrape more of my listening history and predict on new (potentially larger) playlists

CNN or RNN implementation

RNNs (since music is consumed sequentially in many instances) or CNNs using "bag-of-audio-words" could be effective[1]

Model Architecture Adjustments

Advanced regularization techniques (i.e elastic) or learning rate schedulers along with more hidden layers may boost accuracy (but risk overfitting)

Hyperparameter Tuning

Advanced methods of choosing features such as random search could help us choose the optimal set of features

1. Schedl M (2019) Deep Learning in Music Recommendation Systems. *Front. Appl. Math. Stat.* 5:44. doi: 10.3389/fams.2019.00044

Questions?

For any follow-ups, please e-mail:

Apoorv Anand – apanand@uchicago.edu

Link to Project Github:

https://github.com/apanand/UChicago-MSADS/tree/main/Spotify%20Recommender