Spotify + Last.fm Tag-Based Music Recommender

Logistic regression baseline + Streamlit demo

By Hill Zhang

01 BACKGROUND

02 SETTING UP CODE

03 DEMO

04 LIMITATIONS

05 FUTURE DIRECTION

01

Background

Overview of the Project

Goal	● Train a model on Nathan's Spotify listening history (likes and dislikes) ● Get features for tracks using LastFM tags ● Outputs song recommendations not in user's library
Motivation	● Streaming services' recommendation algorithm is often repetitive ● Wanted to see how a basic model competes when the features are chosen well
Why LastFM?	● Originally wanted to use Spotify API to train and predict ● Tags signify genre, mood, location, era, etc. ● Act as a proxy for musical features + users make their own
Broader Question	● Can simple tag patterns understand individual taste well enough to recommend new songs?

02

Setting up Code

Making our Dictionary

Positive examples

- Pull Nathan's top tracks and saved tracks from Spotify
- Assign weights: 2 for top tracks, 1 for saved-only tracks
- Fetch top tags using Last.fm (for the tracks + artists)
- Aggregate tags into a weighted user_tag_counts profile that reflects Nathan's listening habits.

Negative examples

- Using the positive examples, expand dictionary
- For tags that never appear in listening history, find top tracks, artists and similar tags
- Remove tracks that overlap: same artist or tag as positive example
- Remaining tracks become negative examples (weight = 0)

Preprocessing and Training the Model

Turning tags into feature vectors

- Clean each tag string into a list
- Use CountVectorizer with our custom tag splitter
- Build a binary bag-of-tags matrix:
  - One column per tag in the dictionary
  - 1 = track has that tag; 0 = it doesn't
- Attach the label, giving us a full feature matrix ready for modeling.

Training + Tuning

- Fit a logistic regression model on the tag vectors
- Split the data into train/dev/test sets to avoid overfitting
- Tune regularization parameter C → pick value that gives the best F1 on the dev set
- Refit on train+dev with the chosen C, then evaluate once on the test set
- Save the final model and vectorizer so we can score new tracks later.

03

Demo

04

Limitations

Model is trained on one user’s listening history only

Tags from Last.fm are noisy, inconsistent, and sometimes non-musical

Positive labels are implicit (“liked” = saved or top tracks); no explicit user feedback

Negative labels rely on tag-based filtering, which may not reflect true dislikes

05

Future Direction

How can we improve our model?

01

More Data & More Users

More users and more listening history would improve the model's generalization

02

Better Negative Sampling

Use skips or explicit dislike signals instead of relying only on tag differences.

03

Add Audio Features

Combine tags with Spotify audio metrics (tempo, energy, bass) for more nuanced output

How can we improve our model?

04

Interactive Feedback Loop

Let users to rate the recommendations so the model adapts over time

05

Stronger models

Experiment with linear SVMs, tree-based models, or neural net to better capture tag interactions

06

Improve Tag Normalization

Clean and merge redundant tags (hip hop vs. hip-hop) so the model has less noisy data

References

Kauchak, D. (2025). CS 158: Machine Learning, Fall 2025. Department of Computer Science, Pomona College. Course website:

Daumé III, H. (2015). A Course in Machine Learning. Online textbook. Retrieved from http://ciml.info

Streamlit Inc. (2025). Streamlit Documentation. Retrieved from https://docs.streamlit.io

Last.fm Ltd. (2025). Last.fm API Documentation. Retrieved from https://www.last.fm/api

Spotify AB. (2025). Spotify Web API Documentation. Retrieved from https://developer.spotify.com/documentation/web-api

NumPy Developers. (2025). NumPy Reference Documentation. Retrieved from https://numpy.org/doc

pandas Development Team. (2025). pandas Documentation. Retrieved from https://pandas.pydata.org/docs

THANK YOU!

Questions?