Music recommender: extra notes

How did you validate a pattern exists in the dataset in the first place?

I trained a logistic regression model on my entire dataset (unreduced, but L2 normalized), which scored ~86% accuracy. Using PCA I reduced dimensions to 250 while maintaining 77% accuracy. The feature weights were well-distributed, confirming no overly dominant features and showing my taste follows a linear but highly distributed pattern.

Why use a regression model with binary labels?

I preferred minimal discretization from early on, leading to my ridge-based approach. Binary liked/not classification was the quickest way to get a working reward function with full dataset label coverage. While I could have used my additional 'loved' and 'near-miss' categories for more granular scoring, finding proper reward values would require significant time investment in a system I plan to rework anyway.

Why did you not use songs or clusters of any kind as bandit arms?

Songs can't be arms because:

1. they are non-repeatable for a single user and focussed on capturing group taste - while I'm focussing on individual taste;
2. cross-learning from one song to another would be very limited.

And clusters or genres as arms would add major complications like:

1. Depending on where your taste lies you may need a single 'root genre' arm or multiple subgenre arms to group properly, and the proper granularity needed for all these groups could evolve over time. I don't believe this is impossible to implement, but it would be a very complex system for little added benefit from my perspective;
2. It would add a layer of indirection I don't want at this point in the project. If the system picked a song cluster, I'd need another model to select the right song from that cluster - which brings me back to the parameterized value function I already have running over all songs directly.
   I'm aware of recommendation system funneling (lean model for initial selection, then complex models as selection narrows). I'll extend to multiple steps eventually, but want to experiment with other aspects first.

Did you find any new music with this tool?

Yes! In the ~1400 songs labeled as 'not' based on the playlist it came from, I knew there must be some false negatives: songs I enjoyed but forgot to like, add to the right playlist, etc. After training the LinUCB model, I relistened to songs labeled 'not' with high scores and found ~10 songs that I really enjoy. While that doesn't sound like much, this music came from albums I'd already cherry-picked from, where my usual like-rate is only ~10%. This gave me confidence to try the model on completely new music.

While it is very hard to determine how well predicted value actually equates to listening enjoyment, I sampled numerous songs with scores across the entire score range and definitely noticed adherence to my taste in the predicted value.

What struggles did you face?

I had a hard time figuring out the right way to pick a single song based on the audio embeddings + labels, before even getting to the exploration method. From the beginning of the project I was pointed towards bandits in RL, but struggled finding proper arms for my model due to the reasons above. It took a while before I realized I could simplify one of these models to do what I wanted. I also looked into basing the recommendations on cosine similarity, but the pattern in the dataset was too spread out for that (as discussed in the first paragraph).

To understand model performance better I wanted a system of primitives to easily store any metric for any model trajectory and be able to flexibly plot these. That means the same model with different hyperparameters, different models, different data preprocessing, etc. Never before have I been so constrained when making the first abstractions/decouplings. I know that making good primitives is key to a scalable codebase so I was very happy to rapidly run into walls due to my own implementation to learn from. There are still minor tweaks to be made, but overall the primitives for defining, tracking, and plotting metrics are very nice to use and have massively sped up my plot creations.

What did you learn?

I have done a lot of research into different data reduction techniques, trying UMAP and PCA with a lot of different parameters and learning how they work under the hood as well. This also led to me learning more about L normalizations.

While the value prediction method MVP might not be complex, the road to getting there required a lot of thinking about model and environment design. I believe that during this process I got better at system design.

I also explored new tools like Marimo (a Jupyter replacement with better interactivity and code cell isolation) when my existing toolkit became limiting.

What are your next steps?

I have big plans for further in the future which make this entire system distributed across multiple hosts. That will require an amount of network facing activity I don't feel comfortable with in Python. I want to prepare for this while the codebase is still small by rewriting it in Rust. After that, I will experiment more with more steps in the recommendation process, building up the funnel.