Scalable Pipeline Architecture for a Spotify Playlist Recommender System

Pablo Sánchez Orozco, Nicolás Every Alvarez, Alberto Arath Figueroa Salomon

March 29, 2025

Abstract

We propose a scalable machine learning pipeline designed for a Spotify playlist recommender system. Leveraging over one million playlists, our end-to-end pipeline is structured to handle large-scale data ingestion, rigorous validation, efficient processing, iterative model training, and comprehensive analysis, followed by seamless deployment. The architecture emphasizes a model lifecycle approach, ensuring that even small baseline models can evolve by integrating advanced algorithms over time, thus enabling a robust and future-proof recommendation system.

1 Introduction

The growth in digital music consumption has underscored the need for sophisticated recommender systems that can efficiently process large volumes of data and provide personalized recommendations. This paper details the architecture of a scalable pipeline that serves as the backbone of a Spotify playlist recommender system. With a focus on modularity and lifecycle management, our approach not only meets current data processing demands but is also designed to seamlessly evolve with new algorithmic advances.

2 Pipeline Architecture

The proposed pipeline consists of the following key components:

2.1 Data Ingestion

- **Objective:** Efficiently ingest and partition of our tracks and playlist data, we might need to use a distributed file storaged
- **Implementation:** The system could use distributed file storage (e.g., Google Cloud Storage or AWS S3) to manage over one million playlists. Data is ingested using parallel processing frameworks such as Apache Beam, and stored in optimized formats like TFRecords to enhance throughput and scalability.

2.2 Data Validation

- **Objective:** Ensure the integrity and quality of ingested data.
- **Implementation:** Automated routines, powered by tools like TensorFlow Data Validation (TFDV), check for missing values, inconsistencies, and anomalies. Establishing a reliable data schema early in the pipeline is critical for downstream processes.

2.3 Data Processing

- **Objective:** Apply standard techniques to enhance and clean our playlist data set
- **Implementation:** Usage of scalers and encoders to handle categorical variables.

2.4 Model Training

- **Objective:** Train robust baseline models while allowing for the integration of more advanced algorithms.
- **Implementation:** The training component is modular, initially employing collaborative filtering or other baseline techniques. The architecture is designed to facilitate easy upgrades and experimentation with deep learning models, leveraging scalable cloud-based compute resources.

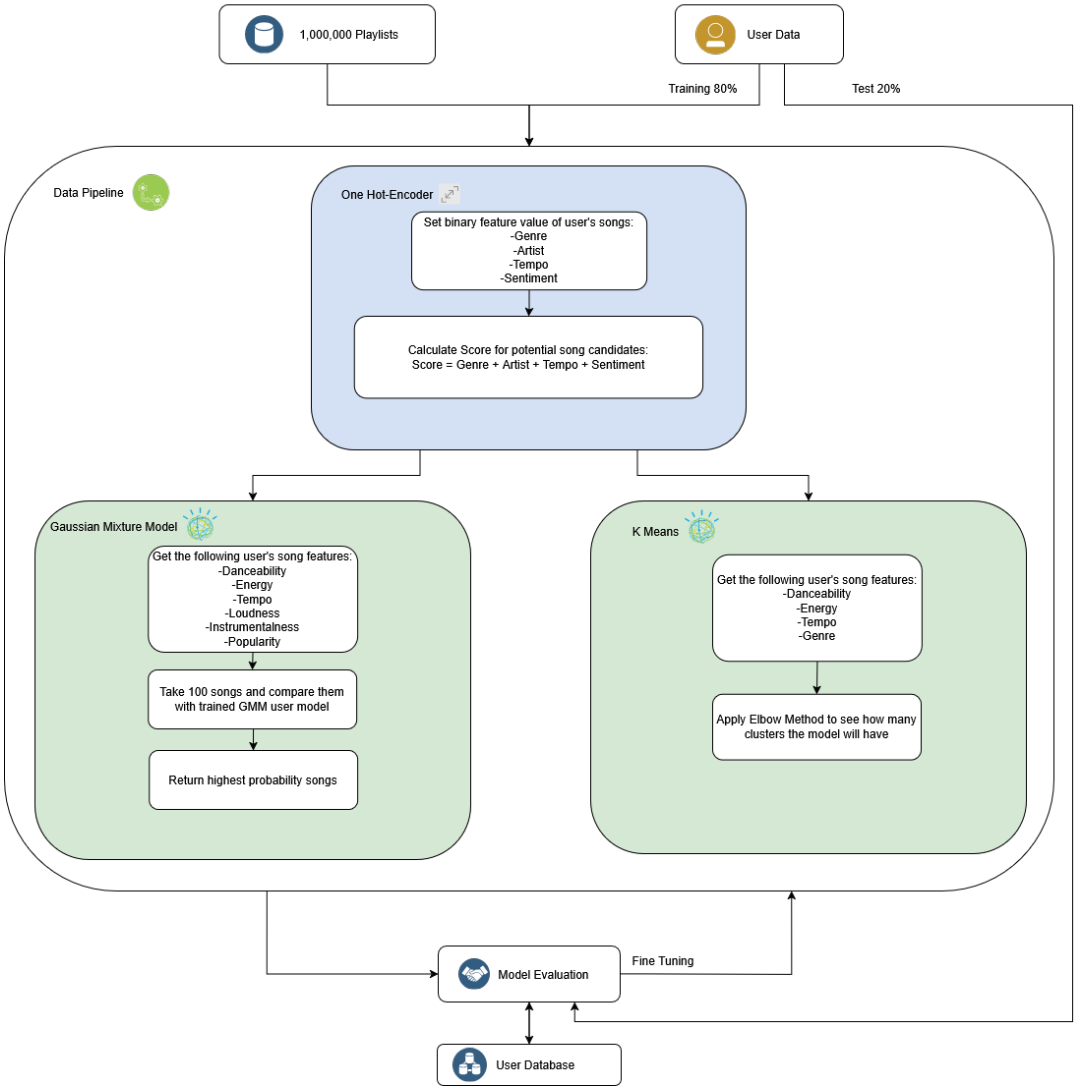

Figure 1: Project Terminal: Overview of the Model Training Workflow

2.5 Model Analysis and Validation

- **Objective:** Evaluate model performance and ensure high-quality recommendations.
- **Implementation:**
  - **User Content Splitting:** Since our training is unsupervised, we perform validation by splitting each user's content. A portion of the user's playlists or tracks is withheld during training.
  - **Matching Recommendations:** The system then generates recommendations based on the trained model and compares these suggestions to the held-out selections from the same user.

- **Statistical Test Formulation:** We introduce a statistical test. The null hypothesis (H_0) is that the recommendation system has no effect on user selections, while the alternative hypothesis (H_1) is that the system improves alignment with user preferences. Based on the computed test statistic, we can use a t-test, chi-square test, or non-parametric tests (depending on the data distribution) to determine if the observed effect is statistically significant.

2.6 Model Deployment

- **Objective:** Deploy the trained model into production for real-time recommendations.
- **Implementation:** Deployment is managed via an automated CI/CD pipeline, which facilitates continuous integration and incremental updates with minimal downtime. This ensures the model lifecycle is maintained, allowing for regular updates and improvements.

3 Scalability and Model Lifecycle Management

- **Modularity:** Each component (data ingestion, validation, processing, training, analysis, and deployment) design is decoupled as we are using a pipeline design, allowing independent updates and maintenance.

4 Summary

We have presented a comprehensive and scalable pipeline architecture for a Spotify playlist recommender system. By covering all critical stages—from data ingestion to model deployment—our design supports efficient handling of over one million playlists and is built with a model lifecycle approach to continuously evolve with advancements in recommendation algorithms. Future work will focus on integrating more sophisticated models and further optimizing each component of the pipeline for enhanced performance.