Project Title: Music Recommendation using Machine Learning

Team Members: Jüri Kivimäe

Business Goals

Background

Music has become an indispensable part of human life, offering emotional depth, entertainment, and therapeutic value. With the advent of digital media, individual music collections can easily span thousands of tracks. However, navigating such vast collections can be overwhelming. This necessitates a smart way of recommending music tailored to user preferences.

Business Goals

The primary objective is to build a personalized Music Recommender System to assist users in navigating their music collections more efficiently. The system aims to provide accurate recommendations based on various parameters like genres, BPM, key, loudness, and pitch.

Business Success Criteria

1. At least 80% accuracy in song recommendations.
2. User-friendly interface for easy navigation.
3. Scalability for large music libraries.

Assessing Your Situation

Inventory of Resources

1. Existing music collection with rich metadata.
2. Computational resources for data mining and machine learning.
3. Programming skills in Python and familiarity with machine learning libraries like TensorFlow or scikit-learn.

Requirements, Assumptions, and Constraints

- **Requirements**: High-quality metadata for each track, sufficient computational power.
- **Assumptions**: Songs with similar features are likely to be enjoyed together.
- **Constraints**: Limited computational resources for extremely large music collections.

Risks and Contingencies

- Inaccuracy in metadata can lead to poor recommendations.
- Computational limitations may require downsampling or feature reduction.

Terminology

- BPM: Beats Per Minute
- Key: Musical key the song is in
- Loudness: Overall loudness of a song

- Pitch: Frequency of the musical notes

Costs and Benefits

- **Costs**: Time and computational resources for model training.
- **Benefits**: Highly personalized music experience, effective navigation through large music libraries.

Defining Data-Mining Goals

Data-Mining Goals

1. To classify songs into multiple clusters based on their features.
2. To identify patterns and correlations between different features of the songs.
3. To develop a model that can accurately recommend songs from the collection based on user input or listening history.

Data-Mining Success Criteria

1. Achieve a model accuracy of at least 80%.
2. Efficiently handle a large number of songs without significant compromise on speed or accuracy.
3. Validate the model using techniques like cross-validation to ensure robustness.

Data Understanding

Gathering Data

Outline Data Requirements

For the Music Recommender System, the data should include at least the following attributes for each song in the music collection:

- Song title and artist
- Genre(s)
- Beats Per Minute (BPM)
- Key
- Loudness
- Pitch
- File format (MP3, FLAC, etc.)

Additional features like sentiment of lyrics, instrumentals vs. vocals, and other musical elements can also be beneficial.

Verify Data Availability

- Most of the metadata (Song title, artist, genre, etc.) is likely readily available in the music files or can be scraped from online databases like MusicBrainz.
- Audio features like BPM, key, loudness, and pitch might require computational analysis of the audio files, which can be done using libraries like Librosa for Python.

Define Selection Criteria

- Only include songs with complete metadata.
- Exclude any songs with corrupt or unreadable files.

Describing Data

The dataset will be a multi-dimensional table where each row represents a song and each column represents a feature (e.g., Genre, BPM, etc.). The primary key could be a unique song identifier, and the values would be the attributes of each song.

Exploring Data

- Perform summary statistics to understand the range, mean, or mode of each feature.
- Visualize data through histograms or scatter plots to understand the distribution of features like genres, BPM, etc.
- Conduct pairwise correlation analyses to identify any strongly correlated features, as these could influence the recommendation system.

Verifying Data Quality

- Check for missing or inconsistent data and decide on an imputation strategy or the removal of such records.
- Validate the accuracy of metadata (e.g., Genre tags can sometimes be inaccurate).
- Test a sample set of data for any inconsistencies in the audio feature extraction process.

Planning

Detailed Plan with List of Tasks

1. **Data Collection (10 hours)**
   - Collect metadata and tags for each song in your music collection.
   - Extract audio features like BPM, key, loudness, and pitch.
2. **Data Preprocessing (5 hours)**
   - Cleanse the data by handling missing values and removing outliers.
   - Normalize and standardize numerical features for model training.
3. **Exploratory Data Analysis (EDA) (5 hours)**
   - Utilize summary statistics and visualizations to understand the underlying patterns in the data.
4. **Model Development (15 hours)**
   - Experiment with different machine learning models including neural networks.
   - Optimize hyperparameters and evaluate models using techniques like cross-validation.
5. **Deployment and User Interface (UI) Development (5 hours)**
   - Deploy the finalized model into a usable application.
   - Create a simple UI for users to interact with the system.

Methods and Tools

- **Data Collection**: Python libraries like pydub for audio feature extraction and web scraping tools for collecting metadata.
- **Data Preprocessing**: Python libraries such as pandas for data manipulation and scikit-learn for feature scaling.
- **EDA**: Data visualization libraries like matplotlib and seaborn.
- **Model Development**: Machine learning libraries like TensorFlow for neural networks and scikit-learn for traditional algorithms.
- **Deployment**: Web frameworks like Flask or Django for exposing the model as a service.

Additional Comments

- The Data Collection phase may require external APIs or scraping techniques, adhering to terms of service.
- For Model Development, it might be beneficial to consider simpler models like k-NN or Decision Trees before diving into neural networks for easier interpretability and quicker iteration.