Business Understanding Report: Spotify Song Recommendations Project

Group D11

Ron-Aaron Vahtra, Reno Feliks Lindvere, Martin Koltsov

Repository: https://github.com/ronv3/spotify-song-recommendations

---

Background

The rapid growth of music streaming services like Spotify has made personalized recommendations a key differentiator in user experience. This project aims to leverage data from Spotify to build a recommendation system based on user input (song(s) or playlist) and predict the potential success of songs using their audio features.

Business Goals

1. Enhance user engagement by providing personalized song recommendations based on preferences.
2. Support music curators, marketers, and artists by identifying potential hits based on audio characteristics.
3. Deliver actionable insights for improving playlist design and music promotion strategies.

Business Success Criteria

- **Quantitative**: Achieve at least 80% user satisfaction in a simulated test environment where users rate recommendations.
- **Qualitative**: Develop a model that successfully identifies songs with a popularity score above a threshold of 75 (on Spotify's scale) at least 85% of the time.

Inventory of Resources

- **People**: Project lead, data analyst, ML engineer, domain expert (music industry).
- **Data**:
    - Dataset 1: Spotify tracks dataset from Kaggle (~20.12 MB).
    - Dataset 2: Spotify API datasets, including audio features and analysis.

- **Tools**: Python libraries (Pandas, NumPy, Scikit-learn, TensorFlow), data visualization tools (Matplotlib, Seaborn), Spotify API for real-time data fetching.
- **Hardware**: Personal computers with sufficient computational power or cloud platforms (e.g., Google Colab).
- **Other**: Spotify Developer account for API access.

Requirements, Assumptions, and Constraints

- **Requirements**:
    - Access to Spotify API data.
    - Adherence to Spotify's API usage limits and terms of service.
- **Assumptions**:
    - Audio features correlate strongly with user preferences and song popularity.
- **Constraints**:
    - Limited labeled data for predicting hits.
    - API rate limits affecting data-fetching speeds.

Risks and Contingencies

- **Risk**: Inadequate API quota may delay data fetching.
    - **Contingency**: Pre-fetch and cache data for commonly queried songs/playlists.
- **Risk**: Model underperformance due to lack of labeled data for hit prediction.
    - **Contingency**: Use transfer learning with existing datasets or augment data with synthetic samples.

Terminology

- **Danceability**: A measure describing how suitable a track is for dancing.
- **Popularity**: A score (0-100) provided by Spotify, reflecting song popularity.
- **Audio Features**: Characteristics like tempo, energy, and loudness.

Costs and Benefits

- **Costs**: Developer time (200 hours), Spotify API usage (free tier), computational resources ($50 for cloud computing).
- **Benefits**: Increased user engagement for music streaming platforms, better playlist curation tools for users, and actionable insights for artists and marketers.

Data-Mining Goals

1. Develop a recommendation model that predicts user preferences based on input tracks or playlists.
2. Build a classification model to predict whether a song will achieve a popularity score above 75.

Data-Mining Success Criteria

- Recommendation system with a Top-10 recommendation accuracy of at least 80%.
- Classification model with a prediction accuracy of at least 85% on the test set.

Project Plan

1. **Data Understanding**
    - Explore datasets from Kaggle and Spotify API to assess feature quality.
    - Visualize relationships between audio features and popularity.
2. **Data Preparation**
    - Preprocess datasets (handle missing values, normalize features).
    - Merge user-provided playlist data with Spotify datasets for model training.

3. **Modeling**
    - Recommendation: Use collaborative filtering and content-based filtering techniques.
    - Hit prediction: Train classifiers like Logistic Regression, Random Forest, and Neural Networks.
4. **Evaluation**
    - Validate recommendation model using user feedback simulations.
    - Evaluate hit prediction models using F1 score and ROC-AUC metrics.
5. **Deployment**
    - Build a user-friendly interface for inputting songs or playlists and displaying recommendations.

Initial Assessment of Tools and Techniques

- Use Spotify's API for live data fetching and Python-based ML frameworks for modeling. Scikit-learn will suffice for initial prototyping, with TensorFlow/Keras for deep learning tasks. Data visualization tools like Seaborn and Matplotlib will be essential for insights.

Data Understanding Report: Spotify Song Recommendations Project

Group D11

Ron-Aaron Vahtra, Reno Feliks Lindvere, Martin Koltsov

Gathering data

1. **Outline data requirements**
    - Features required:
        - ➢ *popularity* for predicting whether a track will become a hit.
        - ➢ *danceability*, *energy*, *acousticness*, *speechiness*, *tempo*, *valence* to understand audio characteristics(musical positiveness).
        - ➢ *track_genre* for track recommendations based on user prefrences.
    - Data format:
        - ➢ Kaggle Spotify dataset: CSV
        - ➢ Spotify API: JSON
    - Time range:
        - ➢ Kaggle dataset updated 2 years ago.
        - ➢ Spotify API has all-time data.
2. **Verify data availability**
    - Kaggle dataset:
        - ➢ Kaggle dataset contains 114,000 rows and 21 columns, covering a variety of audio features and metadata.
        - ➢ All required data exists and is accessible.
    - Spotify API:
        - ➢ Ability to fetch track metadata, audio features, playlists and tracks.
        - ➢ Spotify Developer account required.
        - ➢ Client ID, Client Secret and Redirect URI for OAUTH required.
3. **Define selection criteria**
    - Include all rows where *popularity* > 0 and duration_ms is within typical song length.
    - Exclude tracks with insufficient metadata (e.g. missing danceablity or energy values).

Describing data

- **Kaggle dataset:**
  - Self-explanatory data: *track_id, artists, album_name, track_name, duration_ms, loudness (dB), acousticness (0.0-1.0), tempo (BPM), track_genre*.
  - *popularity*: value between 0-100 (songs that are being played a lot now will have a higher popularity than songs that were played a lot in the past)
  - *explicit*: true = has explicit lyrics, false = does not have explicit lyrics
  - *danceability*: value 0.0-1.0 (track's suitablity for dancing based on combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity).
  - *energy*: value 0.0-1.0 (represents a perceptual measure of intensity and activity)
  - *key*: the key the track is in (e.g. 0 = C, 1 = C♯/D♭, 2 = D, if no key was detected, the value is -1)
  - *mode*: indicates the modality (major or minor) of a track, major is represented by 1 and minor is 0.
  - *instrumentalness*: value 0.0-1.0 (predicts whether a track contains no vocals).
  - *liveness*: value 0.0-1.0 (detects the presence of an audience in the recording).
  - *valence*: value 0.0-1.0 (describing the musical positiveness conveyed by a track).
  - *time_signature*: notational convention to specify how many beats are in each bar, or measure (the time signature ranges from 3 to 7 indicating time signatures of 3/4, to 7/4).
- **Spotify API:**
  - In addition to the features above, the Spotify API offers additional metadata and information, such as playlist-specific data, user-related insights, and release dates, enabling more in-depth analysis and customization for tasks like recommendation systems or trend predictions.

Exploring data

- **Kaggle dataset key findings:**
  - ➢ There is 1000 tracks per genre, 114 genres total.
  - ➢ 14.05% of tracks have popularity score 0.
  - ➢ 91.45% of tracks have explicit value false.
  - ➢ Most tracks have durations clustered around 162,000-192,000 ms.
- **Kaggle dataset hyphotheses:**
  - ➢ Tracks with explicit content might have distinct characteristics.
  - ➢ Genres with consistently low popularity scores could represent niche or less mainstream music.
  - ➢ Same logic for popular genres like “Pop-Film and “K-Pop” with higher average popularity, potentially due to broader appeal.
- **Spotify API:**
  - ➢ Able to fetch same data as in Kaggle dataset for any track in Spotify and even more (e.g., track structure, segments, key changes).
  - ➢ Hard to make hyphotheses and explore data as there is a large amount of tracks and playlists with data in Spotify API.

Verifying data quality

- **Kaggle dataset:**
  - ➢ 1 track with *artists*, *album_name*, *track_name*, *popularity*, *duration_ms* columns as 0 or null values.
  - ➢ 157 tracks with tempo = 0 (missing data).
  - ➢ 24,259 tracks have duplicated values, indicating that some tracks appear multiple times. Duplicates have every column with the same value except for *track_genre*.
- **Spotify API:**
  - ➢ **NB:** Spotify deprecated several API endpoints on November 27th 2024. The get-audio-features was one of those endpoints. This in turn affected spotipy and as of December 2nd, it is not possible to fetch audio-features and analysis data from Spotify API. https://github.com/spotipy-dev/spotipy/issues/1173

Project plan: Spotify Song Recommendations Project

Group D11

Ron-Aaron Vahtra, Reno Feliks Lindvere, Martin Koltsov

---

1. Exploring the dataset and understanding of patterns (everyone ~3 hours)

- After gathering the data we should familiarize ourselves with it and identify parameters that correlate with track popularity. Each team member can conduct their own research and share their findings with the rest of the team. This way we will have better intuition of parameters that affect track popularity and understand the patterns we should look for.

2. Model building (everyone ~10-15 hours)

- Decide whether we remove or keep rows with missing values.
- Figure out the way to get more testing data (Spotify deprecated several API endpoints on November 27th 2024).
- We will need to split the data into two separate parts for training and testing purposes. Try different ratios (20/80, 10/90, etc). We will use *sklearn* for these purposes.
- Try out different algorithms, also try more than one model. Definitely should experiment with Linear regression, random forest, gradient boosting.

3. Testing and tweaking (everyone ~10-15 hours)

- Define popularity more accurately (what the numerical value exactly represents).
- Decide how model should be tested, which output values should be.
- Test the model and analyze the results
- Last step is to save trained model using *joblib*

4. Simple UI (optional ~5 hours)

- If we will have enough time left we can make simple web/ui application for demonstrating purposes.

5. Project Presentation materials (everyone ~5 hours)

- Make poster and notebook which will describe research methods and the results.