Spotify Music Analysis

Genre Classification, Popularity Prediction & Recommendation

Faycal Raghibi, Guerouaoui Ilyas

February 2026

1 Introduction

Music streaming platforms such as Spotify host millions of tracks spanning a wide variety of genres. Automatic genre classification, popularity prediction, and content-based recommendation are fundamental tasks that improve user experience and platform curation. This report presents the methodology and results of a machine-learning pipeline built around a Spotify dataset that contains numerical audio descriptors extracted from the Spotify Web API.

The project addresses three interconnected objectives:

1. **Genre classification** — assigning one of several genre labels to each track based on its audio features.
2. **Popularity prediction** — estimating the continuous popularity score of a track using regression.
3. **Content-based recommendation** — suggesting similar tracks by measuring feature-space proximity.

All experiments were implemented in Python using Scikit-Learn, NumPy, Pandas, Matplotlib and Seaborn. The remainder of this report is organised as follows: Section 2 describes the datasets; Section 3 details the preprocessing pipeline; Sections 4 and 5 present the classification and regression tasks respectively; Section 6 introduces the recommendation approach; Section 7 describes the bonus conformal-prediction experiment; and Section 8 concludes with a discussion.

2 Dataset Description & Exploratory Analysis

2.1 Available Datasets

Four CSV files were provided:

File	**Rows**	**Columns**	**Purpose**
`spotify_dataset_train.csv`	25 493	17	Training (incl. `genre`)
`spotify_dataset_test.csv`	2 834	16	Test (no `genre` label)
`spotify_dataset_subset.csv`	—	—	Popularity regression subset
`recommendation_spotify.csv`	—	—	Recommendation system pool

Table 1: Overview of the project datasets.

2.2 Feature Dictionary

Each track is described by the features listed in Table 2.

Feature	Type	Description
`track_id`	string	Unique Spotify identifier
`track_name`	string	Name of the track
`artists`	string	Performing artist(s)
`release_date`	string	Release date (year extracted)
`acousticness`	float	Confidence measure of acoustic quality
`danceability`	float	Suitability for dancing
`duration_ms`	int	Track duration in milliseconds
`energy`	float	Perceptual intensity measure
`instrumentalness`	float	Likelihood that the track is instrumental
`key`	int	Musical key (0–11, Pitch Class)
`liveness`	float	Probability of live performance
`loudness`	float	Overall loudness in dB
`mode`	int	Modality (0 = minor, 1 = major)
`speechiness`	float	Presence of spoken words
`tempo`	float	Estimated BPM
`valence`	float	Musical positiveness
`popularity`	int	Popularity score (0–100)
`explicit`	int	Explicit content flag
`genre`	string	Genre label (train set only)

Table 2: Feature dictionary for the Spotify datasets.

2.3 Exploratory Observations

An initial exploration of the training set revealed several characteristics that influenced the pipeline design:

- **Class imbalance** — The genre distribution is highly uneven. Some genres (e.g. *pop*, *rock*) dominate the dataset while others are significantly under-represented. This motivated the use of class-balanced weights in the classifier.
- **Missing values** — A small fraction of entries contain missing numerical values, requiring imputation.
- **Feature scales** — Features span very different ranges (e.g. `duration_ms` $\sim 10^5$ vs. `acousticness` $\in [0, 1]$), necessitating standardisation.
- **Dimensionality** — A PCA visualisation of the standardised features showed substantial overlap among genres in the first two principal components, hinting at a challenging classification task.

3 Preprocessing Pipeline

The following preprocessing steps were applied consistently to both training and test data:

1. **Year extraction.** The `release_date` column was parsed to extract a numerical `year` feature, providing a compact temporal signal.
2. **Column selection.** Non-predictive identifiers (`track_id`, `track_name`, `artists`) and the raw `release_date` string were dropped.
3. **Missing-value imputation.** Remaining missing values in numerical columns were filled using `SimpleImputer` with a *mean* strategy.
4. **Feature standardisation.** All numerical features were centred and scaled to unit variance via `StandardScaler`, fitted on the training set and applied to both splits.

5. **Categorical encoding.** The binary categorical feature `explicit` was retained as-is (already encoded as 0/1). One-hot encoding was applied where needed for any additional categorical signals.

After preprocessing, the feature matrix contained the following columns used for modelling: `acousticness`, `danceability`, `duration_ms`, `energy`, `instrumentalness`, `key`, `liveness`, `loudness`, `mode`, `speechiness`, `tempo`, `valence`, `popularity`, `explicit`, and `year`.

4 Genre Classification

4.1 Model Selection

A **Random Forest Classifier** was chosen for multi-class genre prediction. Random forests aggregate many decorrelated decision trees via bootstrap aggregation, providing robustness against overfitting and the ability to handle mixed feature types. The following hyperparameters were set:

- `n_estimators` = 100 — number of trees in the ensemble.
- `class_weight` = `balanced` — automatically adjusts sample weights inversely proportional to class frequencies, mitigating the effect of class imbalance.
- `random_state` = 42 — fixed seed for reproducibility.

4.2 Evaluation Protocol

The model was evaluated using **stratified 5-fold cross-validation** on the training set. The primary metric is the **micro-averaged F1 score**, which computes the global precision and recall across all classes and is equivalent to accuracy when all samples are assigned exactly one label.

4.3 Results

Metric	**Value**
CV F1 (micro) mean	0.4396
CV F1 (micro) std	± 0.0132

Table 3: Cross-validation results for genre classification.

A micro-averaged F1 of approximately **0.44** reflects the inherent difficulty of the task: many genres share similar audio profiles, and the feature space (purely numerical audio descriptors) may not capture high-level stylistic differences. The balanced class weighting helps ensure that minority genres are not systematically ignored, but the overall separability remains limited.

4.4 Prediction on the Test Set

After cross-validation, the classifier was retrained on the full training set and used to predict genre labels for the 2 834 unlabelled test tracks. The predictions were exported to `submission.csv` in the required format (`track_id`, `genre`).

5 Popularity Prediction

5.1 Task Definition

The `popularity` column (integer in $[0, 100]$) serves as the regression target. This task uses the dedicated `spotify_dataset_subset.csv` file, which focuses on a curated selection of tracks.

5.2 Model

A **Random Forest Regressor** was employed, mirroring the ensemble philosophy used for classification. Decision-tree-based regressors are well-suited for capturing non-linear relationships between audio features and popularity without requiring explicit feature engineering.

5.3 Results

Metric	**Value**
Mean Squared Error (MSE)	517.99
Coefficient of Determination (R^2)	0.2126

Table 4: Popularity regression performance.

An R^2 of approximately **0.21** indicates that the audio features alone explain roughly 21% of the variance in popularity. This is expected because popularity is heavily influenced by external factors—artist recognition, marketing, playlist placement, temporal trends—that are not captured by the audio descriptors. The MSE of 517.99 corresponds to a root mean squared error of approximately 22.8 popularity points on the 0–100 scale.

6 Content-Based Recommendation

6.1 Approach

A **content-based filtering** strategy was implemented using **cosine similarity** over the standardised audio-feature vectors. Given a query track, the system retrieves the k most similar tracks from the `recommendation_spotify.csv` pool by ranking pairwise cosine similarities.

6.2 Pipeline

1. Load and preprocess the recommendation dataset (same pipeline as Section 3).
2. Compute the cosine similarity matrix between all track pairs.
3. For a given query track, sort the similarity scores in descending order and return the top-k nearest neighbours (excluding the query itself).

Cosine similarity is a natural choice for this setting because it measures the angular proximity between feature vectors, making it invariant to uniform scaling—an important property when features have been standardised but may still exhibit different dynamic ranges depending on the subset.

7 Bonus: Conformal Prediction

As an additional experiment, a **conformal prediction** framework was applied to quantify prediction uncertainty on the genre classification task.

7.1 Method

1. A **K-Nearest Neighbours (KNN)** classifier was trained on the training set after standard preprocessing.
2. A **calibration set** of 1 500 samples was held out from the training data to compute non-conformity scores.
3. For each test instance the conformal predictor produces a **prediction set** — a subset of genre labels guaranteed to contain the true label with probability at least $1 - \alpha$, where α is the user-specified significance level.

7.2 Results

α	**Coverage (%)**	**Avg. Set Size**	**Empty Sets (%)**
0.01	99.88	16.74	0.00
0.05	98.32	13.96	0.00
0.10	96.93	12.36	0.00
0.20	93.82	10.19	0.00

Table 5: Conformal prediction results at various significance levels.

The conformal predictor achieves the desired coverage guarantees at each significance level. However, the average prediction-set sizes are large (e.g. ≈ 14 genres at $\alpha = 0.05$), reflecting the limited discriminative power of the audio features for fine-grained genre separation. This corroborates the modest F1 score observed in Section 4.

8 Discussion & Conclusion

This project demonstrated an end-to-end machine-learning pipeline for Spotify track analysis, covering classification, regression, and recommendation. Several observations merit discussion:

- **Feature limitations.** The purely numerical audio features provided by the Spotify API capture low-level timbral, rhythmic, and harmonic properties but cannot encode higher-level semantic or cultural attributes. Incorporating textual metadata (lyrics, artist biography) or spectral representations (e.g. Mel-spectrograms) could significantly improve genre classification.
- **Class imbalance.** Despite the use of balanced class weights, under-represented genres remain difficult to classify. Techniques such as oversampling (SMOTE), or hierarchical classification (grouping related genres) may yield improvements.
- **Popularity modelling.** An R^2 of 0.21 confirms that popularity is only weakly related to intrinsic audio properties. A richer feature set including social-network metrics, release timing, and playlist exposure would be needed for practical popularity forecasting.
- **Recommendation.** The cosine-similarity-based recommender provides a simple yet effective baseline. Hybrid approaches combining content-based and collaborative filtering could offer more diverse and accurate recommendations.
- **Uncertainty quantification.** The conformal prediction experiment illustrates that statistical coverage guarantees come at the cost of large prediction sets when the underlying classifier has moderate accuracy. Improving the base classifier would directly tighten the conformal intervals.

Overall, the Random Forest family proved to be a reliable and interpretable choice for both classification and regression on tabular audio features. Future work could explore gradient-boosted trees (XGBoost, LightGBM), neural embeddings, or multi-modal fusion to push performance further.