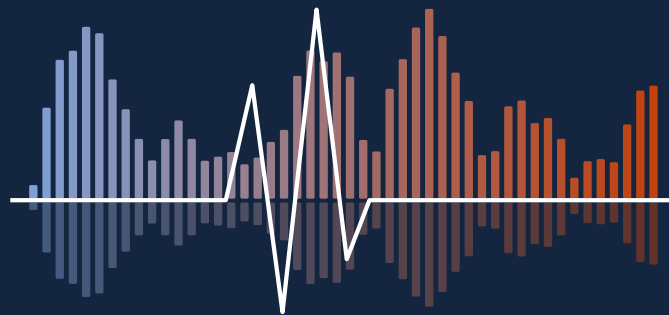




UNIVERSITÀ DI PISA

END-TO-END MACHINE LEARNING

clustering • deep learning • explainability



SoundPulse

114,000 tracks • 114 → 10 genres • 6 notebooks

SoundPulse: End-to-End Machine Learning on 114,000 Spotify Tracks

Project Report

Giuseppe Muschetta

M.Sc. in Data Science & Business Informatics

University of Pisa

2025

Abstract

SoundPulse takes a public catalog of 114,000 Spotify tracks, described by audio descriptors and a few metadata fields, through the full data-science lifecycle in six notebooks. After cleaning and feature engineering, the 114 original genre labels are mapped by hand onto ten macro-genres. A Local Outlier Factor screen removes the 5% least typical tracks and exposes tracks that are listed several times under different genres. K-Means finds nine segments; a neural network and a decision tree recover them from the features with 97.5% and 92.6% test accuracy, which shows that the segments are not artifacts of the algorithm. For ten-class genre prediction from 13 features, a tuned Random Forest reaches 58.5% accuracy and a macro ROC-AUC of 0.898, while a residual deep network with 4.6 million parameters places the correct genre among its top three suggestions for 84.1% of the tracks. Permutation importance, per-genre gradient saliency and SHAP values explain both models and agree that acousticness, popularity, danceability and energy carry most of the signal. Popularity is almost uncorrelated with every single feature, yet tree ensembles explain more than half of its variance ($R^2 = 0.529$, eighteen times the linear baseline). Finally, for the rare explicit-content class, SMOTE raises minority recall from 39.5% to 62.8% and macro-F1 from 0.760 to 0.795 at an almost unchanged ROC-AUC.

Contents

1	Introduction	3
2	Data and preparation	3
2.1	The dataset	3
2.2	Cleaning and feature engineering	3
2.3	A ten-class genre taxonomy	4
2.4	Exploratory analysis	4
3	Outlier detection with Local Outlier Factor	5
3.1	Feature space	5
3.2	The Local Outlier Factor	6
3.3	Choosing the number of neighbors	6
3.4	Results	6
4	Clustering: latent sound families	7
4.1	K-Means	7
4.2	Choosing the number of clusters	8
4.3	The nine segments	8
4.4	Are the clusters real?	8
5	Genre classification	10
5.1	Setup	10
5.2	Model A: Random Forest	10
5.3	Model B: a residual deep network	11
5.4	Comparing the two models	14
6	Explainability	15
6.1	Deep network: permutation importance	16
6.2	Deep network: gradient saliency per genre	16
6.3	Random Forest: SHAP values	16
6.4	Local explanations	17
6.5	Do the explanations agree?	17
7	Popularity regression	17
7.1	Setup	17
7.2	Models	18
7.3	Results	19
8	Imbalanced learning: detecting explicit content	20
8.1	Task	20
8.2	Resampling strategies	20
8.3	Results	21
9	Implementation and reproducibility	23
10	Conclusions	23

1 Introduction

Streaming platforms describe every track with a small set of audio descriptors computed from the signal, such as energy, danceability or acousticness, together with metadata such as popularity and an explicit-content flag. These numbers drive search, recommendation and playlist curation, so it is natural to ask how much they actually reveal about a track. This project studies that question on a public catalog of 114,000 Spotify tracks [24] and turns it into four concrete questions:

- Q1. Which tracks are anomalous?** Local Outlier Factor [7] flags and removes the 5% least typical tracks.
- Q2. Which natural “sound families” exist?** K-Means [18, 2] finds nine segments, and two supervised models check that they are real and separable.
- Q3. Can the features alone tell the genre?** A Random Forest [5] and a residual deep network classify tracks into ten macro-genres; permutation importance, gradient saliency and SHAP [21] explain why.
- Q4. Can popularity be predicted, and can explicit content be detected?** Five regressors are compared on popularity, and three resampling strategies are compared on the rare explicit class.

The work is organized as a pipeline of six notebooks (Figure 1). Each notebook reads a dataset written by an earlier one, so every stage can be rerun and inspected on its own. Section 2 describes the data and its preparation, Section 3 the outlier screen, Section 4 the clustering, Section 5 the genre classifiers and Section 6 their explanations. Section 7 covers popularity regression, Section 8 imbalanced learning, Section 9 the implementation, and Section 10 concludes.

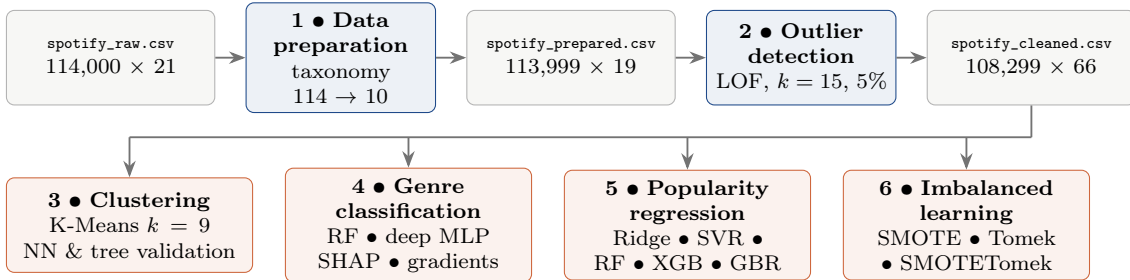


Figure 1: The SoundPulse pipeline. Blue boxes are the two preparation notebooks, gray boxes the datasets they write (rows × columns), orange boxes the four modeling notebooks, which all start from the cleaned dataset.

2 Data and preparation

2.1 The dataset

The data is the *Spotify Tracks Dataset* [24]: 114,000 tracks and 21 columns, with exactly 1,000 tracks for each of 114 genre labels. Besides identifiers and names (`track_id`, `artists`, `album_name`, `track_name`), every track carries the fields in Table 1. Most of them are Spotify’s audio descriptors, which are computed from the signal and scaled to $[0, 1]$; popularity and the explicit flag are metadata.

2.2 Cleaning and feature engineering

The dataset is almost complete: a single row has no artist, album or track name and is dropped, leaving 113,999 tracks. The technical columns `Unnamed: 0` and `track_id` are removed. Three transformations make the remaining fields easier to use:

Field	Range	Meaning
popularity	0–100	Popularity score, driven mainly by recent play counts
explicit	0/1	Whether the lyrics contain explicit content
danceability	0–1	Suitability for dancing (tempo, rhythm stability, beat strength)
energy	0–1	Perceived intensity and activity
loudness	dB	Average loudness of the track
speechiness	0–1	Presence of spoken words
acousticness	0–1	Confidence that the track is acoustic
instrumentalness	0–1	Likelihood that the track contains no vocals
liveness	0–1	Likelihood that the track was recorded live
valence	0–1	Musical positiveness (happy, cheerful versus sad, tense)
tempo	BPM	Estimated tempo
time_signature	integer	Estimated meter (beats per bar)
key, mode	0–11, 0/1	Pitch class of the key and major/minor mode
duration_ms	ms	Track length
track_genre	114 labels	Genre label under which the track was collected

Table 1: Fields used in the project, besides identifiers and names.

- **keymode.** The integer **key** (0–11) and the binary **mode** are merged into one readable categorical feature such as “C# minor” or “A major”, with 24 possible values. Musically, key and mode only make sense together.
- **duration_mins.** **duration_ms** is converted to minutes and the original column is dropped, so the same information never appears twice.
- **Types and names.** **explicit** is cast from Boolean to a 0/1 integer and **track_genre** is renamed to **genres**.

2.3 A ten-class genre taxonomy

With 114 labels of 1,000 tracks each, the raw genres are too fine to learn from 13 features, and many of them overlap by construction: *songwriter* and *singer-songwriter*, *latin* and *latino*, *house* and *deep-house*. A semantic mapping, written by hand, compresses them into ten macro-genres (Table 2). The grouping follows musical families (all metal sub-genres together, all electronic dance styles together) and, where sound is not the defining trait, geography and culture (*k-pop*, *j-rock*, *cantopop* or *afrobeat* in World & Regional).

The resulting target is realistically imbalanced: the two largest classes hold more than 40% of the tracks, while Hip-Hop & R&B, which groups only two original labels, holds less than 2%. Every classifier in Section 5 has to deal with this.

2.4 Exploratory analysis

Figure 2 shows the distributions of the numeric features and their Spearman rank correlations. The main observations are:

- **Popularity has a large spike at zero:** a sizable part of the catalog has essentially no recent plays.
- **acousticness** is bimodal, since tracks tend to be clearly acoustic or clearly produced, while **instrumentalness**, **speechiness** and **liveness** are strongly right-skewed.
- The strongest monotonic relations are physically meaningful: **energy and loudness** ($\rho = 0.75$), **energy and acousticness** ($\rho = -0.71$), loudness and acousticness ($\rho = -0.53$),

Macro-genre	Labels	Examples of original labels	Share
World & Regional	24	k-pop, j-pop, j-rock, cantopop, afrobeat, reggae, turkish	21.3%
Electronic & Dance	23	house, techno, trance, drum-and-bass, dubstep, edm	20.0%
Classical/Ambient & Stage	13	classical, opera, piano, ambient, sleep, show-tunes	11.5%
Rock & Alternative	12	rock, alt-rock, punk, grunge, emo, psych-rock	10.5%
Latin & Brazilian	11	latin, reggaeton, salsa, samba, sertanejo, tango	9.6%
Country/Folk & Songwriter	8	country, folk, bluegrass, acoustic, singer-songwriter	7.1%
Pop & Indie	8	pop, indie, indie-pop, synth-pop, power-pop	7.0%
Metal & Hardcore	7	metal, black-metal, death-metal, metalcore, grindcore	6.2%
Jazz/Soul/Funk/Blues	6	jazz, soul, funk, blues, gospel, groove	5.1%
Hip-Hop & R&B	2	hip-hop, r-n-b	1.7%

Table 2: The ten macro-genres, the number of original labels each one groups, and their share of the 108,299 tracks that remain after outlier removal (Section 3).

and danceability and valence ($\rho = 0.46$). Loud tracks are energetic, acoustic tracks are quiet, and danceable tracks tend to sound positive.

- **Popularity is almost uncorrelated with every single feature** ($|\rho| \leq 0.08$). If popularity can be predicted at all, it must be through interactions between features, which Section 7 confirms.

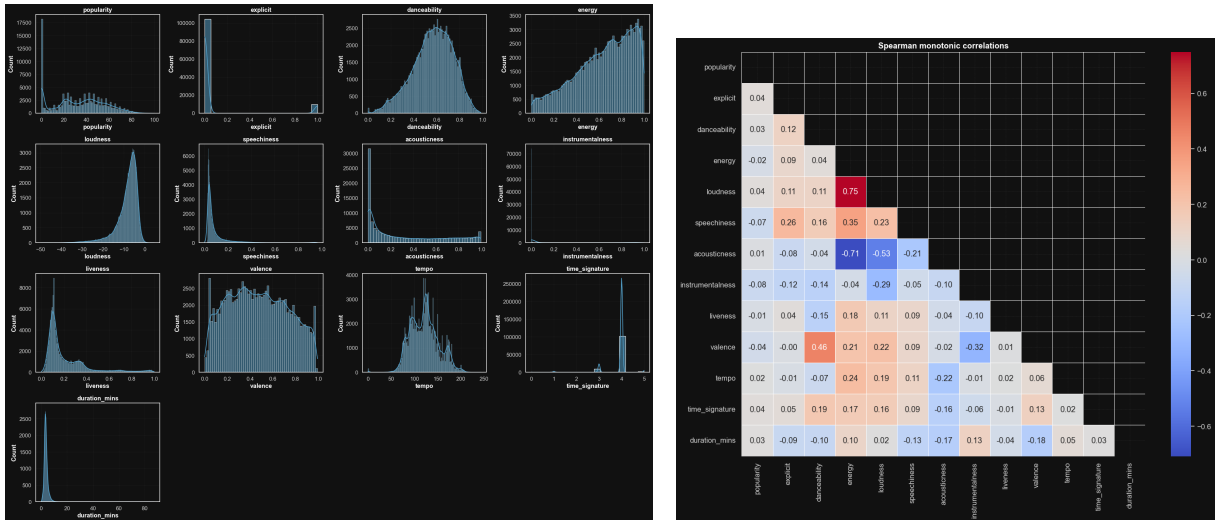


Figure 2: Left: distributions of the numeric features. Right: Spearman correlation matrix (lower triangle).

3 Outlier detection with Local Outlier Factor

3.1 Feature space

Outliers are searched in the space that the later models will also use. The 13 numeric fields of Table 1 (with `duration_mins` in place of `duration_ms`) are standardized to zero mean and unit variance, and the two categorical features, `keymode` (24 values) and the macro-genre (10 values),

are one-hot encoded. The result has $13 + 24 + 10 = 47$ dimensions, and distances are Euclidean. The standardized and encoded columns are kept next to the original ones, which gives the 66 columns of the cleaned dataset.

3.2 The Local Outlier Factor

A global rule, such as a z -score threshold, only finds points that are far from the bulk of the data. Many anomalies are instead *local*: a point can sit in a sparse pocket next to a dense region without being extreme in any coordinate. The Local Outlier Factor of Breunig et al. [7] compares the density around a point with the density around its neighbors. Let $N_k(p)$ be the k nearest neighbors of p and $d_k(o)$ the distance from o to its own k -th neighbor. The reachability distance and the local reachability density are

$$\text{reach}_k(p, o) = \max\{d_k(o), d(p, o)\}, \quad \text{lrd}_k(p) = \left(\frac{1}{|N_k(p)|} \sum_{o \in N_k(p)} \text{reach}_k(p, o) \right)^{-1}, \quad (1)$$

and the outlier factor is the average ratio between the densities of the neighbors and that of the point:

$$\text{LOF}_k(p) = \frac{1}{|N_k(p)|} \sum_{o \in N_k(p)} \frac{\text{lrd}_k(o)}{\text{lrd}_k(p)}. \quad (2)$$

Values close to 1 mean that the point is as dense as its neighborhood; values well above 1 mean that it is much sparser. scikit-learn [25] reports the negated value, so the most anomalous tracks have the most negative scores.

3.3 Choosing the number of neighbors

Instead of guessing k , a single nearest-neighbor search with $k = 100$ gives, for every $k \leq 100$, the average distance from each track to its k -th neighbor. The curve rises steeply for small k and then flattens; its knee is at $k = 15$, which is the value used. The contamination is set to 5%, so the 5% of tracks with the highest factor are labeled as outliers.

3.4 Results

LOF flags **5,700 tracks (5.00%)** and keeps 108,299. The median score is -1.042 , so a typical track is as dense as its neighborhood, and the 5% threshold corresponds to a factor of 1.218. Table 3 lists the ten most extreme tracks, and two patterns stand out.

#	Track	Artist(s)	Label	Pop.	Score
1	The Prayer	Kid Cudi	hip-hop	0	-2.33×10^{10}
2	Autonom III, Op. 10	Alireza Mashayekhi	iranian	0	-2.09×10^{10}
3	Rain	Wiz Khalifa; PARTYNEXTDOOR	hip-hop	0	-1.92×10^{10}
4–6	Face	Prateek Kuhad	folk, songwriter, singer-songwriter	38	-1.86×10^{10}
7	Rubai 40	K. Saheb Nassagh; A. Allah Yari	iranian	0	-1.76×10^{10}
8	Oh Meu Deus	Projota	r-n-b	0	-1.74×10^{10}
9	Little Girl – Mono	John Mayall & The Bluesbreakers	blues	0	-1.74×10^{10}
10	Christmas Is The Time...	Billy Squier	metal	0	-1.72×10^{10}

Table 3: The ten tracks with the most negative LOF score. Rows 4–6 are the same recording listed under three different genre labels.

- **Zero-popularity re-releases.** Seven of the ten tracks have popularity 0, and several are known songs re-released on compilation-style albums such as *Music for Rainy Days* or *Fiesta temática 60s*. They have the audio profile of mainstream tracks but no plays, so they sit apart from their look-alikes along the popularity dimension.
- **Exact duplicates.** Prateek Kuhad’s *Face* appears three times with identical features, once for each of the labels *folk*, *songwriter* and *singer-songwriter*, which all map to the same macro-genre. Cross-listing of this kind is a classic data-quality issue of playlist-based datasets.

Duplicates also explain the size of the scores. When more than k identical copies of a track exist, the copies have a zero reachability distance to each other, and their local density becomes numerically infinite; scikit-learn caps it at 10^{10} by adding 10^{-10} to the denominator. Any track whose neighbors are such copies then receives a factor of the order of 10^{10} . The ranking is still meaningful, but the raw values should be read as “extremely isolated” rather than as exact ratios.

Figure 3 projects all tracks onto the first two principal components. Many flagged tracks lie *inside* the main cloud: they are local anomalies, sparse with respect to their own neighborhood in 47 dimensions, which is exactly what LOF is designed to catch and what a global rule would miss. The 108,299 inliers form the cleaned dataset used by every later notebook.

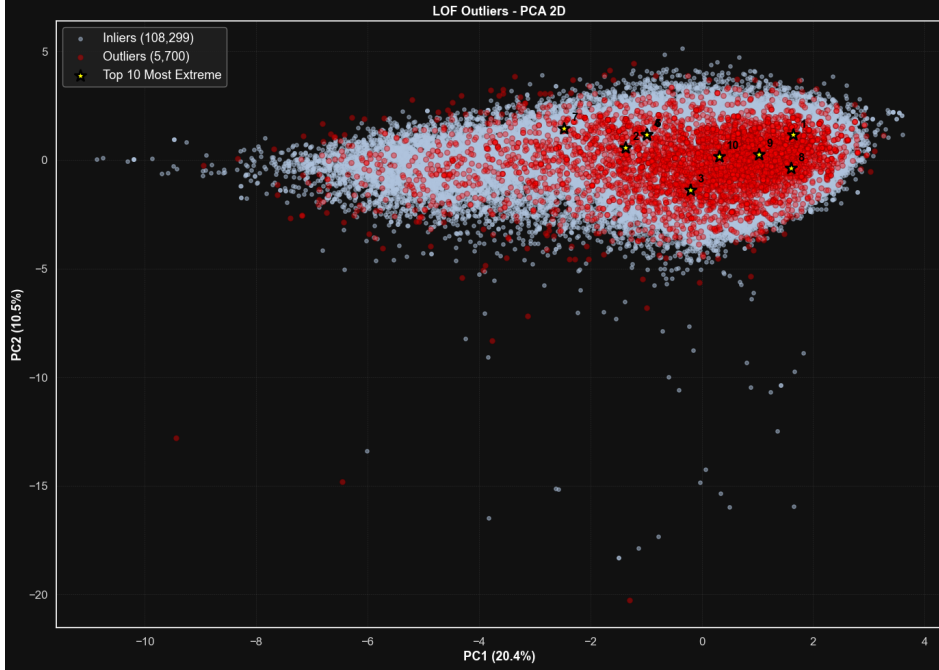


Figure 3: LOF outliers (red) and inliers (light blue) on the first two principal components; stars mark the ten most extreme tracks.

4 Clustering: latent sound families

4.1 K-Means

K-Means [18] partitions the tracks into k clusters $C_1, \dots, C_k$ with centroids $\mu_1, \dots, \mu_k$ by minimizing the within-cluster sum of squares, or inertia,

$$W(k) = \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mu_j\|^2. \quad (3)$$

The centroids are initialized with k -means++ [2], which spreads the initial centers and makes the result much less dependent on chance, and the iterations use Elkan’s algorithm [11], which skips distance computations through the triangle inequality. The input is the 47-dimensional space of Section 3, and the random seed is fixed for reproducibility.

4.2 Choosing the number of clusters

Inertia always decreases with k , so it was computed for $k = 2, \dots, 50$ and inspected for an elbow, which appears around $k \approx 9$ –10 (Figure 4, left). The decision was then made with the silhouette coefficient [26]. For a track i , let $a(i)$ be its mean distance to the other members of its cluster and $b(i)$ its mean distance to the members of the nearest other cluster:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \in [-1, 1]. \quad (4)$$

The mean silhouette was estimated on a random sample of 20,000 tracks for $k = 2, \dots, 30$, and its local maxima were found automatically: $k = 4, 7, 9, 14, 20, 24$ and 29. The value at $k = 2$ is higher (0.186), but a two-way split is too coarse to be useful. Among the local maxima, $k = 9$ **has the highest silhouette, 0.123**, and it agrees with the elbow region of the inertia curve.

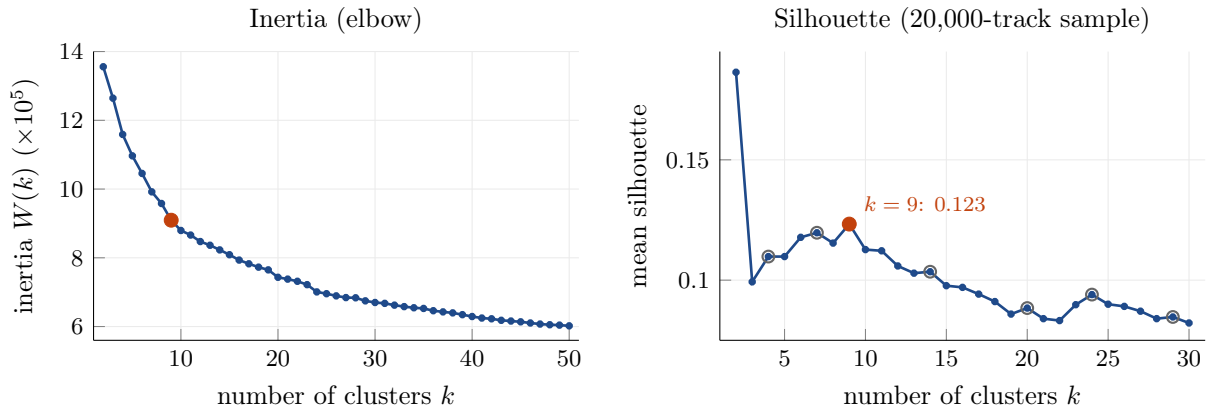


Figure 4: Choice of k . Left: inertia for $k = 2, \dots, 50$. Right: mean silhouette for $k = 2, \dots, 30$; open circles are the other local maxima, and the orange point is the selected $k = 9$.

A silhouette of 0.12 is low in absolute terms, and this is expected. In 47 dimensions, with 34 binary one-hot columns, tracks do not form well-separated blobs; they fill a continuous space, and the clusters are contiguous regions of it. Whether such regions are still meaningful is tested directly in Section 4.4.

4.3 The nine segments

Table 4 reports the size of each segment. Three large clusters (2, 6 and 8) hold more than half of the tracks, and cluster 7 is a small niche of 879 tracks. In the principal-component projections of Figure 5, the clusters appear as ordered, contiguous bands that run from acoustic and instrumental tracks on one side to loud, high-energy productions on the other.

4.4 Are the clusters real?

A clustering algorithm always returns clusters, even on structureless data. A practical test is whether a supervised model can learn to *recover* the cluster labels from the features: if it can, with high accuracy on unseen tracks, the segments correspond to consistent regions of the feature space rather than to arbitrary cuts. Two very different validators were trained on a stratified 70/30 split (75,809 training and 32,490 test tracks):

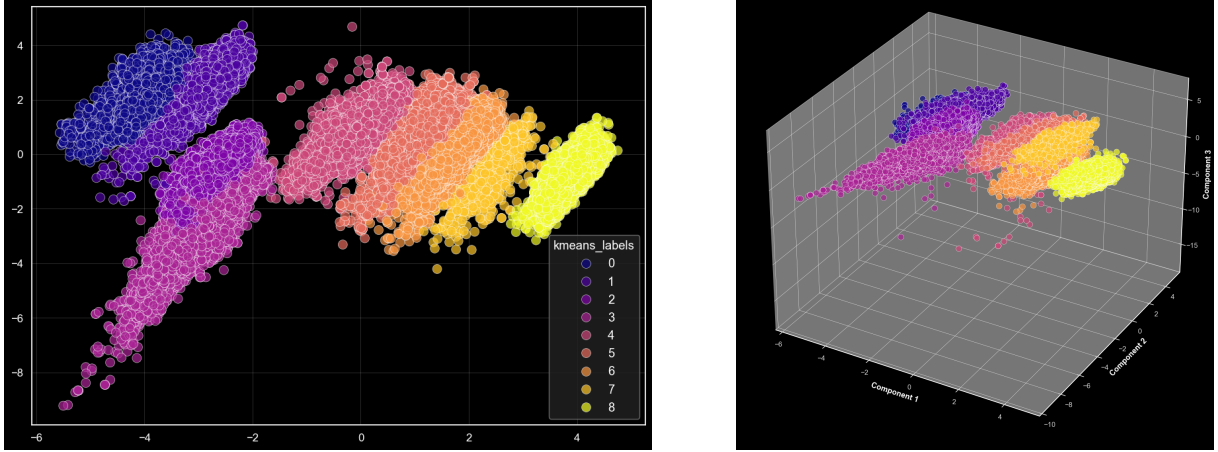


Figure 5: The nine K-Means clusters projected on the first two (left) and three (right) principal components.

- **A feed-forward neural network** with layers of 512, 256, 128 and 64 units (Swish activation, batch normalization, dropout from 0.4 to 0.2), then 32 ReLU units and a 9-way softmax. It is trained with AdamW [20] (learning rate 10^{-3} , weight decay 10^{-4}), batches of 512, balanced class weights and early stopping on a 20% validation split.
- **A decision tree** [6], tuned by a grid search over 2,940 configurations (depth, split and leaf sizes, criterion, splitter and feature subsampling) with 5-fold stratified cross-validation, for 14,700 fits in total.

Cluster	Tracks	Share	F1, neural network	F1, decision tree
0	6,335	5.8%	0.95	0.882
1	5,026	4.6%	0.97	0.932
2	21,824	20.2%	0.98	0.929
3	8,419	7.8%	0.96	0.926
4	17,961	16.6%	0.97	0.917
5	8,485	7.8%	1.00	0.995
6	20,334	18.8%	0.98	0.918
7	879	0.8%	0.97	0.922
8	19,036	17.6%	0.98	0.922
Test accuracy			97.47%	92.58%
Balanced accuracy / macro-F1			98.04% / 0.97	92.65% / 0.927
Cohen's κ / MCC			0.970 / 0.970	—

Table 4: Cluster sizes and how well each cluster is recovered by the two validators on the 32,490 test tracks.

Both validators recover the segments with high accuracy (Table 4). The network misclassifies only 821 of 32,490 test tracks (2.5%), with Cohen's κ [10] and the Matthews correlation coefficient [23] both at 0.970, and even the smallest cluster (879 tracks) has an F1 of 0.97. Its mean confidence is 0.97 overall but only 0.71 on its errors, so the mistakes concentrate on genuinely ambiguous tracks near cluster borders. Cluster 5 is recovered almost perfectly by both models, which suggests that it is defined by a sharp threshold on very few features.

The best tree uses entropy, a maximum depth of 20 and no feature subsampling; it has 3,483 leaves. Its cross-validated accuracy is $92.10\% \pm 0.14\%$ across the five folds, and its test accuracy, 92.58%, matches it. The training accuracy is 99.97%, so the tree memorizes its training data,

but the stable cross-validation shows that this costs little at test time, and the learning curve in Figure 6 is still rising: more data would help further.

The tree is also the more interpretable of the two. Its feature importances show what *defines* the segments: **energy** (0.176), **instrumentalness** (0.139), **explicit** (0.135), **popularity** (0.112) and **danceability** (0.102), followed by liveness, acousticness and valence. The 34 one-hot columns for key and macro-genre are almost irrelevant: the segments are shaped by how tracks *sound* and how they are consumed, not by their key or by their genre label. Each segment can therefore be read in business terms, for example “quiet instrumental tracks” versus “high-energy explicit tracks”.

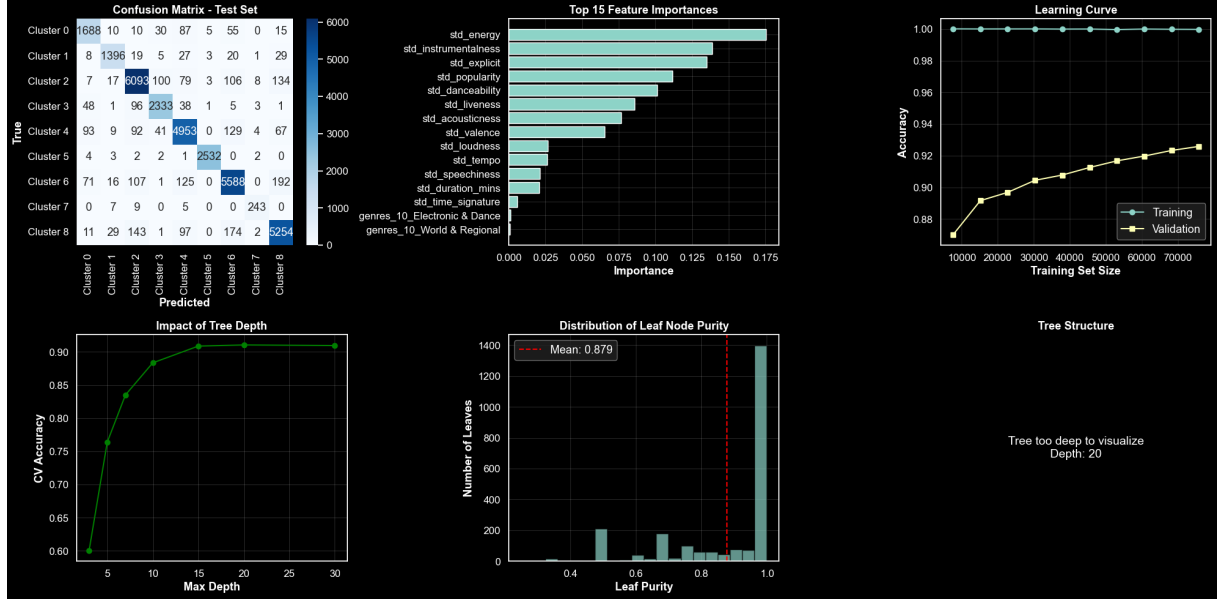


Figure 6: Decision-tree validation of the clusters: test confusion matrix, feature importances, learning curve, accuracy versus maximum depth and distribution of leaf purity.

5 Genre classification

5.1 Setup

The task is to predict one of the ten macro-genres from the 13 standardized numeric features alone: popularity, explicit, danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo, time signature and duration. Key and mode are not used. The data is split 70/30 with stratification, giving 75,809 training and 32,490 test tracks, and the test set is used only for the final evaluation. Two reference points help to read the results: guessing uniformly at random gives 10% accuracy, and always predicting the largest class, World & Regional, gives 21.3%. Because the classes are imbalanced, the main selection metric is the macro-averaged F1 score, which weighs every genre equally.

5.2 Model A: Random Forest

A Random Forest [5] with balanced class weights was tuned by a grid search over 160 configurations (number of trees, maximum depth, minimum samples per split and per leaf) with 5-fold stratified cross-validation and macro-F1 as the score, for 800 fits. The best forest has 300 trees, a maximum depth of 30, at least 7 samples per split and $\sqrt{13}$ candidate features per split; its out-of-bag accuracy is 0.579.

Metric	Score	Metric	Score
Accuracy	0.585	ROC-AUC, one-vs-rest, macro	0.898
Balanced accuracy	0.540	ROC-AUC, one-vs-rest, weighted	0.892
Macro-F1	0.535	ROC-AUC, one-vs-rest, micro	0.917
Weighted F1	0.581	Matthews correlation	0.520
Macro precision / recall	0.535 / 0.540	Cohen’s κ	0.520

Table 5: Test performance of the tuned Random Forest on 32,490 tracks.

The forest reaches **58.5% accuracy**, 5.8 times random guessing and almost three times the majority baseline, and a **macro ROC-AUC of 0.898** (Table 5). The AUC is much higher than the accuracy suggests: even when the top prediction is wrong, the true genre is usually ranked high. Table 6 and Figures 7 and 8 break the result down by genre.

Genre	Test tracks	Precision	Recall	F1	AUC
Metal & Hardcore	2,023	0.58	0.67	0.63	0.954
Latin & Brazilian	3,117	0.64	0.71	0.67	0.949
Classical/Ambient & Stage	3,739	0.65	0.69	0.67	0.935
Country/Folk & Songwriter	2,298	0.57	0.64	0.60	0.933
Electronic & Dance	6,512	0.70	0.70	0.70	0.917
Rock & Alternative	3,426	0.51	0.47	0.49	0.878
Hip-Hop & R&B	542	0.23	0.26	0.24	0.876
Pop & Indie	2,268	0.47	0.38	0.42	0.874
Jazz/Soul/Funk/Blues	1,648	0.47	0.35	0.40	0.855
World & Regional	6,917	0.53	0.52	0.53	0.811

Table 6: Random Forest results per genre, sorted by one-vs-rest AUC.

The per-genre view tells a clear story:

- **Genres defined by sound are easy.** Metal (AUC 0.954), Latin (0.949), Classical/Ambient (0.935) and Country/Folk (0.933) have distinctive acoustic signatures: distortion and loudness, specific rhythms, acoustic and instrumental textures.
- **Genres defined by culture or geography are hard.** World & Regional has the lowest AUC (0.811). It groups 24 labels, from k-pop to reggae and turkish music, that share a region or a market rather than a sound. Being both the largest and the most heterogeneous class, it also acts as a sink: for seven of the ten genres, it is the most frequent wrong prediction.
- **The remaining errors are musically plausible.** 30% of Hip-Hop & R&B tracks are predicted as Latin & Brazilian, plausibly because reggaeton, which belongs to the latter in this taxonomy, shares rhythmic and production traits with hip-hop. 15% of Metal tracks are predicted as Rock, 17% of Pop & Indie as Electronic, and 21% of Jazz/Soul/Funk/Blues as World & Regional.
- **The forest knows when it is unsure.** Its mean top-class probability is 0.51, and 55.7% of its predictions have a top probability below 0.5, which is consistent with classes that genuinely overlap in feature space.

5.3 Model B: a residual deep network

The second model is a deep network designed from scratch for tabular, imbalanced, multi-class data and implemented in TensorFlow/Keras [1]. Its architecture is shown in Figure 9. The main design choices are:

True genre	Classical/Ambient & Stage	0.69	0.05	0.04	0.01	0.01	0.02	0.00	0.05	0.03	0.10
	Country/Folk & Songwriter	0.09	0.64	0.02	0.00	0.01	0.03	0.00	0.04	0.04	0.12
	Electronic & Dance	0.02	0.01	0.70	0.01	0.02	0.01	0.04	0.03	0.04	0.12
	Hip-Hop & R&B	0.04	0.02	0.04	0.26	0.03	0.30	0.01	0.08	0.08	0.15
	Jazz/Soul/Funk/Blues	0.04	0.03	0.10	0.02	0.35	0.11	0.04	0.03	0.07	0.21
	Latin & Brazilian	0.03	0.03	0.01	0.04	0.04	0.71	0.00	0.00	0.02	0.12
	Metal & Hardcore	0.01	0.00	0.07	0.01	0.02	0.00	0.67	0.02	0.15	0.05
	Pop & Indie	0.10	0.07	0.17	0.01	0.01	0.02	0.04	0.38	0.08	0.12
	Rock & Alternative	0.03	0.05	0.06	0.03	0.03	0.03	0.11	0.04	0.47	0.15
	World & Regional	0.08	0.06	0.12	0.01	0.02	0.07	0.03	0.03	0.06	0.52
		Classical	Country/Folk	Electronic	Hip-Hop/R&B	Jazz/Soul	Latin	Metal	Pop/Indie	Rock	World
		Predicted genre									

Figure 7: Row-normalized confusion matrix of the Random Forest: each row shows how the test tracks of one genre are distributed over the predicted genres, and the diagonal is the recall.

- **A wide first layer** (1,536 units) that projects the 13 inputs into a rich representation, followed by blocks of decreasing width (768, 512, 384, 256, 128, 64).
- **Residual connections** [14] around two blocks, so that gradients flow easily through the depth of the network.
- **Layer normalization** [3] and **GELU** activations [15] in every block, which are more stable than batch statistics with very large batches.
- **A sigmoid gate** after the first residual block: a dense layer with sigmoid activation produces a weight in $(0, 1)$ for each of the 768 units, and the representation is multiplied element-wise by these weights. This attention-style mechanism lets the network suppress or amplify features depending on the input.
- **Strong regularization:** Gaussian input noise ($\sigma = 0.1$), dropout decaying from 0.45 to 0.10 [28], L2 penalties of 3×10^{-4} on the hidden layers and an L1/L2 penalty on the output layer.

Training. The network is trained with AdamW [20] (weight decay 5×10^{-5}) and gradient clipping at norm 1. The learning rate follows a linear warm-up [13] over the first $E_w = 10$ epochs

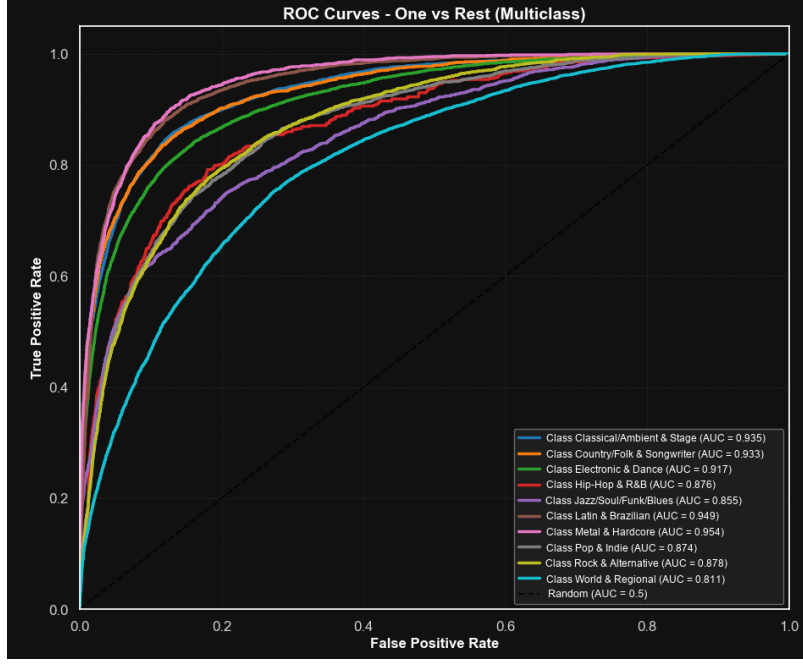


Figure 8: One-vs-rest ROC curves of the Random Forest for the ten genres.

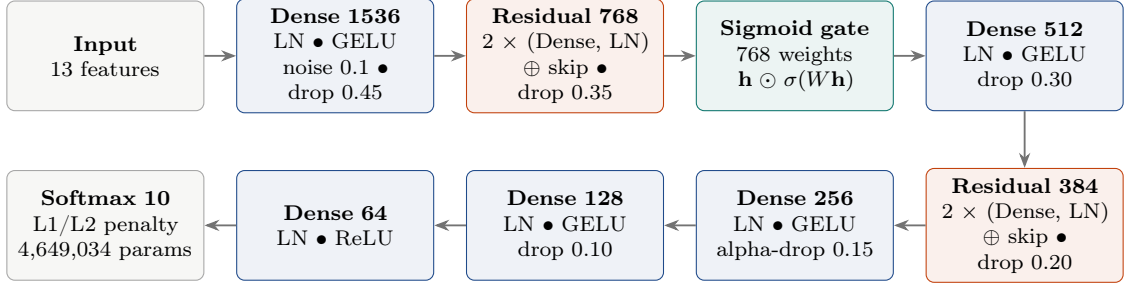


Figure 9: Architecture of the residual deep network (LN: layer normalization). Blue blocks are dense layers, orange blocks residual blocks and the green block the sigmoid gate.

and then a cosine decay [19] over the remaining ones:

$$\eta(e) = \begin{cases} \eta_0 \frac{e+1}{E_w}, & e < E_w, \\ \frac{\eta_0}{2} \left(1 + \cos \frac{\pi(e - E_w)}{E - E_w} \right), & e \geq E_w, \end{cases} \quad \eta_0 = 5 \times 10^{-4}, E = 350, \quad (5)$$

with a plateau-based reduction as a backup. Class imbalance is handled with class weights. Fully balanced weights $n/(Kn_c)$ would give Hip-Hop & R&B a weight more than twelve times that of World & Regional, which makes training unstable, so the weights are smoothed with a power of 0.75 and rescaled to mean one:

$$w_c \propto \left(\frac{n}{Kn_c} \right)^{0.75}, \quad \frac{1}{K} \sum_{c=1}^K w_c = 1. \quad (6)$$

The resulting weights range from 0.42 (World & Regional) to 2.82 (Hip-Hop & R&B). Training uses batches of 4,096 tracks and holds out 20% of the training set for validation; the test set plays no part in model selection. Early stopping with a patience of 35 epochs restores the weights of the best epoch, **epoch 197**. The network was trained on the GPU of an Apple M3 Ultra through TensorFlow-Metal.

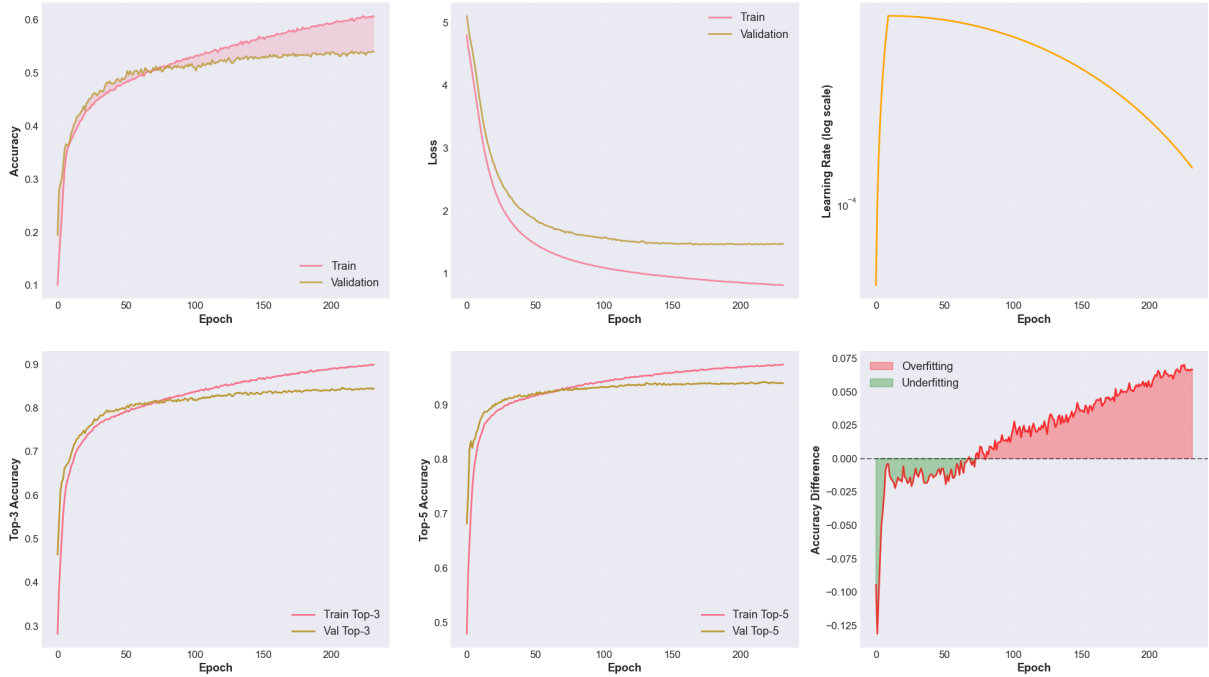


Figure 10: Training dashboard of the deep network: accuracy, loss, learning-rate schedule, top-3 and top-5 accuracy, and the gap between training and validation accuracy.

Figure 10 shows the training dynamics. Validation accuracy is slightly *above* training accuracy during the first ~ 80 epochs, because dropout and input noise are active only during training. After that point the gap reverses and grows slowly, while the validation loss flattens: the network has extracted most of what the features can tell, and early stopping prevents it from memorizing further.

Results. On the test set the network reaches a top-1 accuracy of 53.6% and a macro-F1 of 0.502 (Table 7). Its strength appears when it is allowed more than one guess (Figure 11): **the correct genre is among the top three suggestions for 84.1% of the tracks and among the top five for 94.1%**. Genre boundaries are fuzzy, so for a recommender or a playlist-tagging assistant, a short ranked list that almost always contains the right genre is often more useful than a single hard label. The network is also aware of its own uncertainty: its mean confidence is 0.74 on correct predictions and 0.55 on wrong ones.

Metric	Score	Metric	Score
Top-1 accuracy	0.536	Macro-F1	0.502
Top-3 accuracy	0.841	Weighted F1	0.537
Top-5 accuracy	0.941	Macro precision	0.489
Mean confidence, correct / wrong	0.74 / 0.55	Macro recall	0.542

Table 7: Test performance of the residual deep network.

5.4 Comparing the two models

The Random Forest is the better single-label classifier: it has higher accuracy (58.5% against 53.6%) and higher macro-F1 (0.535 against 0.502). The two models, however, make different trade-offs, which Figure 12 shows genre by genre. The smoothed class weights push the network toward the

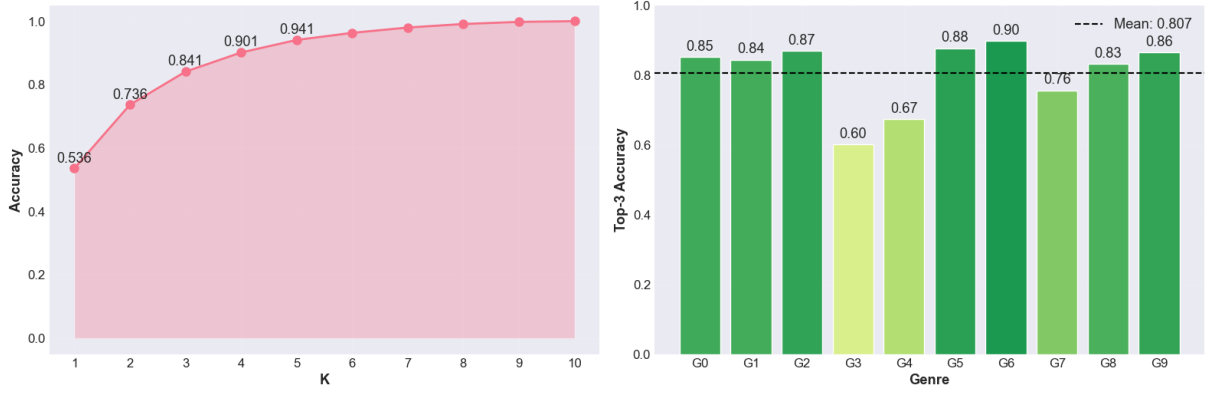


Figure 11: Left: top- K accuracy of the deep network for $K = 1, \dots, 10$. Right: top-3 accuracy per genre (G0–G9 are the genres in alphabetical order: Classical, Country/Folk, Electronic, Hip-Hop, Jazz, Latin, Metal, Pop, Rock, World).

minority genres: its recall is higher on Hip-Hop & R&B (0.41 against 0.26), Jazz/Soul/Funk/Blues (0.46 against 0.35), Pop & Indie (0.44 against 0.38) and Metal (0.72 against 0.67), and lower on the two largest classes, World & Regional (0.35 against 0.52) and Electronic & Dance (0.60 against 0.70). Its macro recall, 0.542, is the highest of the two models. The forest is the right choice when one label must be assigned; the network is the right choice when a ranked list of candidates is acceptable and rare genres must not be overlooked.

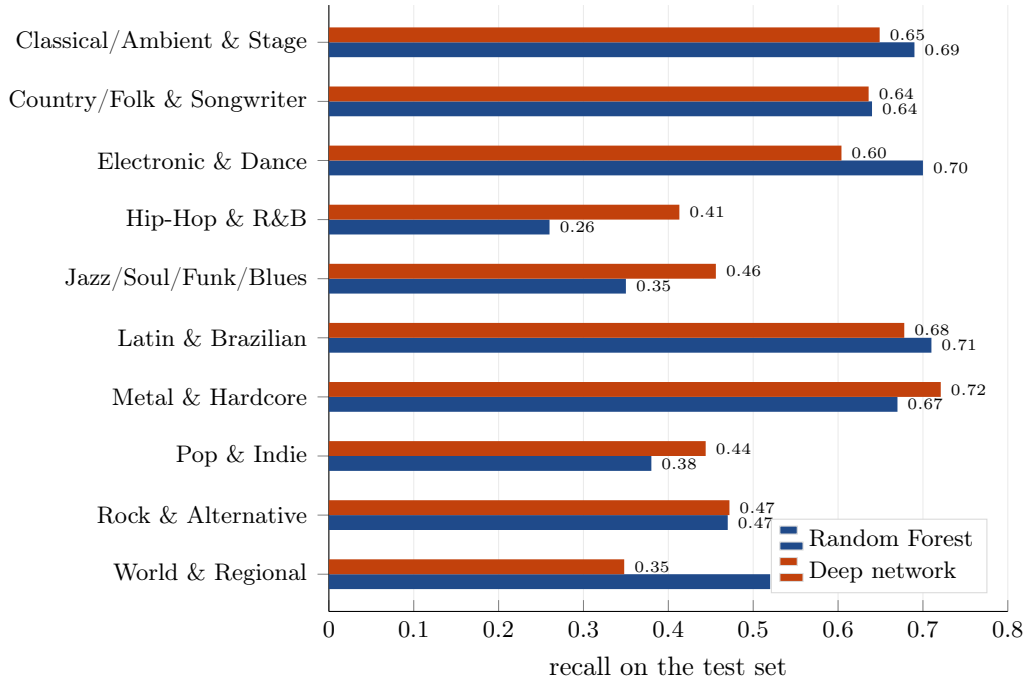


Figure 12: Recall per genre of the two classifiers. The deep network trades recall on the two largest classes for recall on the minority genres.

6 Explainability

A model that assigns genres is only useful if its decisions can be understood and checked. Both classifiers were therefore opened at three levels: *global* (which features matter overall), *per class* (which features matter for each genre) and *local* (why a given track received its label).

6.1 Deep network: permutation importance

Permutation importance [5] measures how much the accuracy drops when the values of one feature are randomly shuffled, which breaks its link with the target while keeping its distribution. It was computed for the network on 2,000 test tracks, with five shuffles per feature (Figure 13, left). The five most important features are **popularity** (0.160 ± 0.008), energy (0.110), danceability (0.109), acousticness (0.105) and loudness (0.094). Every feature has a positive importance, and the top eight account for 80% of the total.

6.2 Deep network: gradient saliency per genre

To see which features each genre reacts to, the gradient of every class probability with respect to the inputs was computed with `tf.GradientTape` on 500 random training tracks, and averaged in absolute value [27]:

$$S_{c,j} = \frac{1}{N} \sum_{i=1}^N \left| \frac{\partial p_c(\mathbf{x}_i)}{\partial x_{ij}} \right|. \quad (7)$$

Figure 13 (right) normalizes each genre’s column by its maximum. Popularity is the most sensitive input for every genre, but the second tier is genre-specific: Electronic & Dance reacts strongly to duration, danceability, speechiness and loudness, and Country/Folk & Songwriter reacts most to speechiness and duration.

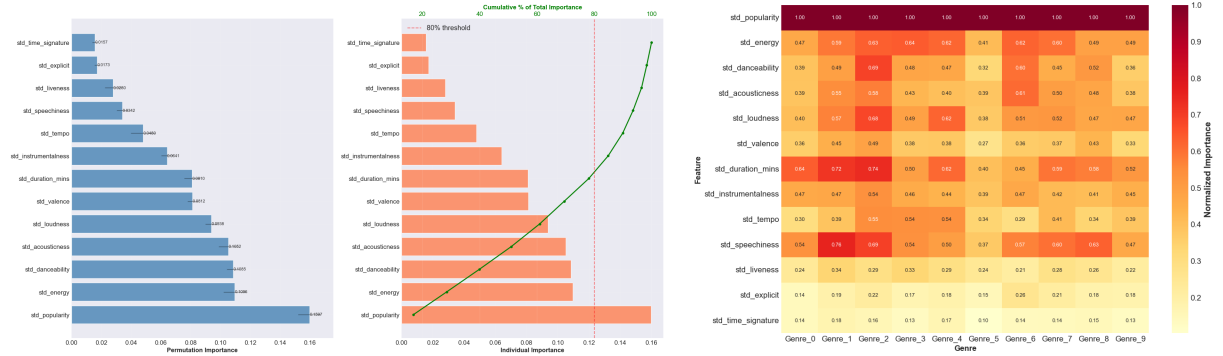


Figure 13: Deep network. Left: permutation importance with standard deviations, and cumulative share of the total. Right: gradient saliency per genre, normalized by the largest value in each column (Genre_0 to Genre_9 in the order of Figure 11).

6.3 Random Forest: SHAP values

SHAP [21] explains a prediction $f(\mathbf{x})$ by distributing the difference between it and the average prediction among the features, according to their Shapley values from cooperative game theory:

$$f(\mathbf{x}) = \phi_0 + \sum_{j=1}^M \phi_j, \quad \phi_j = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!(M - |S| - 1)!}{M!} [f_{S \cup \{j\}}(\mathbf{x}) - f_S(\mathbf{x})], \quad (8)$$

where F is the set of M features, f_S the model restricted to the features in S , and ϕ_0 the average prediction. The Shapley values are the only attribution that is locally accurate, consistent and gives zero to features that play no role. For tree ensembles, TreeExplainer [22] computes them exactly and in polynomial time. Values were computed for the first 500 test tracks and all ten classes, giving a $500 \times 13 \times 10$ tensor.

Summed over the ten classes, the mean absolute SHAP values rank **acousticness** first (0.299), followed by popularity (0.237), danceability (0.200), energy (0.176) and instrumentality

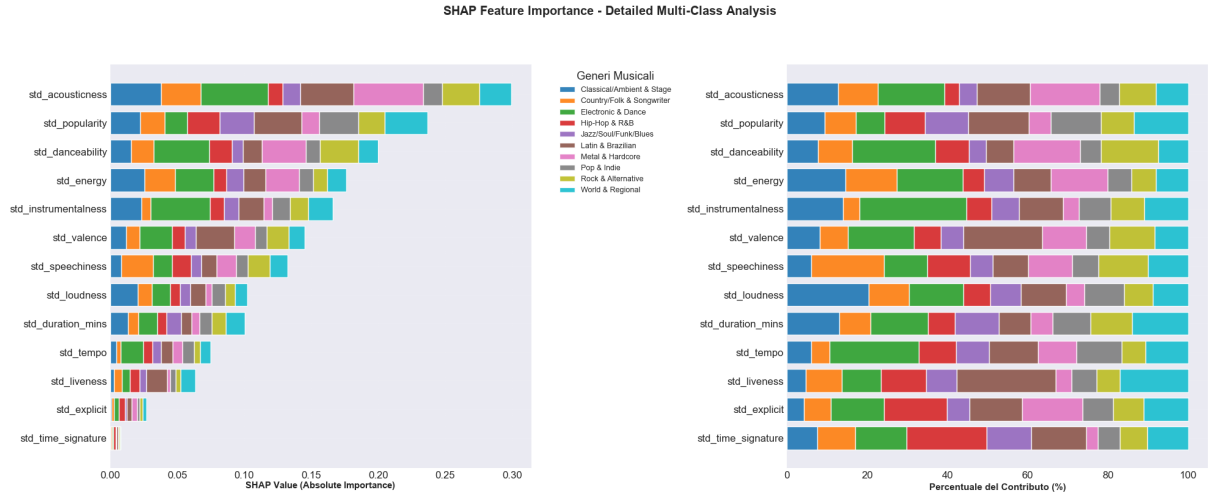


Figure 14: Random Forest SHAP importance: mean absolute SHAP value of each feature, stacked by genre (left), and the same contributions as a share of each feature’s total (right).

(0.166) (Figure 14). Acousticness is also the single most important feature for five genres: Classical/Ambient, Country/Folk, Electronic, Latin and Metal. Popularity leads for Hip-Hop, Jazz, Pop and World, and danceability for Rock. The split is meaningful: acousticness separates genres that *sound* different, while popularity separates genres whose playlists have different audiences.

6.4 Local explanations

Figure 15 explains a single test track, an Electronic & Dance track that the forest classifies correctly. Starting from the base rate of 0.10, its low acousticness (-0.94 standard deviations) adds $+0.12$ to the probability of Electronic & Dance, its high danceability ($+1.13$) adds $+0.09$, and tempo, valence and speechiness add smaller amounts, for a final probability of 0.446. The same analysis was run on a *misclassified* track: a Hip-Hop & R&B track predicted as Latin & Brazilian with a low probability of 0.205, mostly because of its popularity ($+0.071$) and acousticness ($+0.046$), with its energy pointing the other way (-0.042). Local explanations of this kind show where the model fails and whether its reasons are sensible, which aggregate metrics cannot do.

6.5 Do the explanations agree?

Table 8 compares the top five features found by the different methods. Despite different models and different definitions of importance, the same small group recurs: **acousticness**, **popularity**, **danceability** and **energy** appear in every list for genre classification. This agreement makes the explanations more trustworthy than any single method would be. It also qualifies the phrase “from audio alone”: popularity is metadata, not audio, and its weight shows that the genre playlists in this catalog have very different popularity profiles. A model meant for new, unreleased tracks would have to do without it.

7 Popularity regression

7.1 Setup

The goal is to predict a track’s popularity from its other features. The target is the standardized popularity score, and the inputs are the remaining 12 standardized features; popularity itself is,

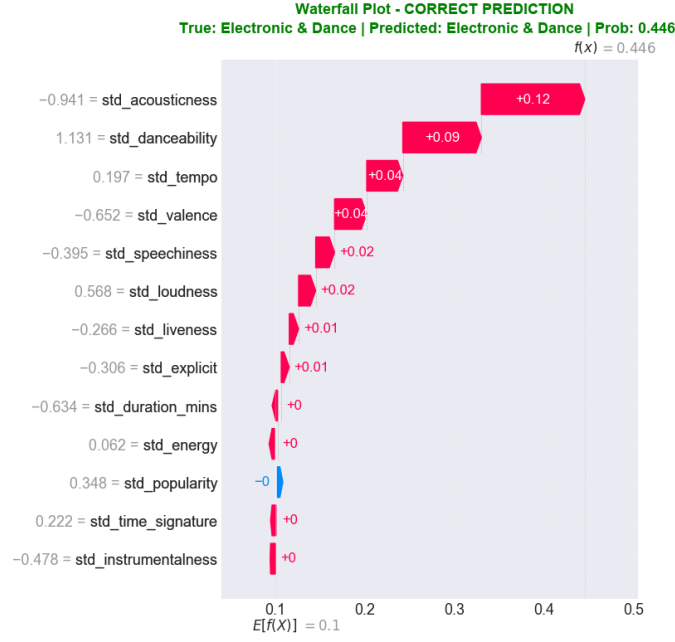


Figure 15: SHAP waterfall plot for one correctly classified Electronic & Dance track: each bar is the contribution of one feature (the standardized value is shown on the left), from the base rate $E[f(X)] = 0.10$ to the predicted probability $f(\mathbf{x}) = 0.446$.

Rank	RF, impurity	RF, SHAP	Network, permutation	Clusters, tree
1	acousticness	acousticness	popularity	energy
2	popularity	popularity	energy	instrumentalness
3	danceability	danceability	danceability	explicit
4	energy	energy	acousticness	popularity
5	speechiness	instrumentalness	loudness	danceability

Table 8: Top five features by method. The first three columns explain genre classification; the last one explains the cluster segments of Section 4.

of course, excluded. The data is split 70/30 at random (75,809 and 32,490 tracks). Models are compared on the coefficient of determination R^2 , the root mean squared error (RMSE) and the mean absolute error (MAE). Since the target is standardized, RMSE and MAE are expressed in standard deviations of popularity, and a model that always predicts the mean has $R^2 = 0$ and an RMSE close to 1.

7.2 Models

- **Ridge regression** [16] with $\alpha = 1$, which minimizes $\|\mathbf{y} - X\mathbf{w}\|_2^2 + \alpha\|\mathbf{w}\|_2^2$.
- **Linear support vector regression** with $\varepsilon = 0.1$ and $C = 1$, which ignores errors smaller than ε and penalizes larger ones linearly.
- **Random Forest** with 800 trees, unlimited depth and $\sqrt{12}$ candidate features per split.
- **XGBoost** [9] with learning rate 0.1, tuned over the number of trees (300, 500) and the maximum depth (15, 18).
- **Gradient Boosting** [12] with learning rate 0.1 and 300 trees, tuned over 16 combinations of depth and minimum split and leaf sizes.

The three ensembles were evaluated with repeated k -fold cross-validation (5 folds, 2 repeats) and

three scores at once (R^2 , RMSE, MAE), refitting the best configuration on R^2 .

7.3 Results

Model	Train R^2	CV R^2	Test R^2	Test RMSE	Test MAE
Ridge	–	–	0.029	0.975	0.809
Linear SVR	–	–	0.014	0.982	0.802
XGBoost (300 trees, depth 15)	0.959	0.468	0.508	0.694	0.454
Gradient Boosting (300 trees, depth 20)	0.967	0.483	0.522	0.684	0.440
Random Forest (800 trees)	0.911	0.492	0.529	0.679	0.488

Table 9: Popularity regression on 32,490 test tracks. RMSE and MAE are in standard deviations of popularity. The Random Forest’s out-of-bag R^2 is 0.531.

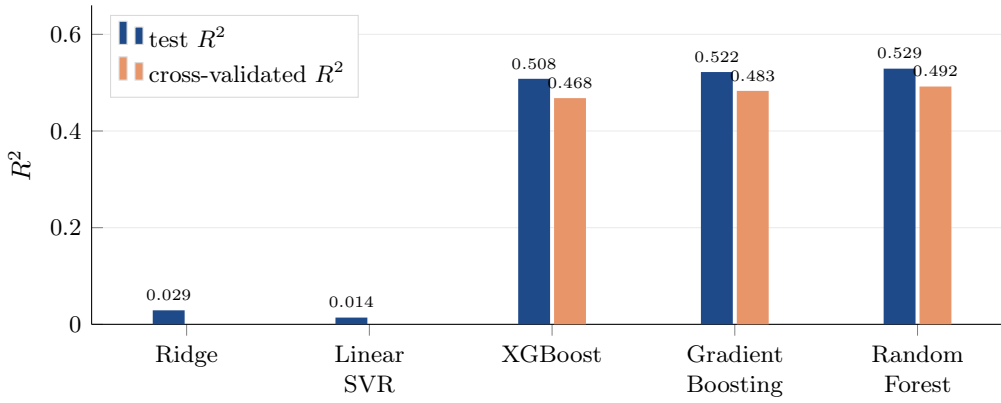


Figure 16: Test and cross-validated R^2 of the five regressors. The linear models explain almost nothing; the tree ensembles explain about half of the variance.

The contrast is stark (Table 9 and Figure 16). The two linear models explain essentially none of the variance ($R^2 = 0.029$ and 0.014), while all three tree ensembles explain about half of it. The **Random Forest is the best model, with $R^2 = 0.529$** , eighteen times the Ridge baseline, the lowest RMSE and the highest cross-validated score; Gradient Boosting has the lowest MAE. The out-of-bag estimate of the forest, 0.531, confirms the test score. All ensembles fit their training data far better than the test data (training R^2 from 0.91 to 0.97), but the cross-validated and test scores agree, so the gap reflects the noise in popularity rather than an unstable model.

This confirms the hint from the correlation analysis: no single feature is related to popularity, yet combinations of features are. Popularity depends on *interactions*, such as the combination of mood, acoustic texture and length typical of a given market, and only non-linear models can capture them.

The parity plots in Figure 17 show the difference. Ridge collapses into a flat horizontal band: it predicts nearly the mean for every track. The forest follows the diagonal, although it shrinks the extremes toward the center, as averaging ensembles do. The vertical stripe at the far left corresponds to the tracks with popularity zero: their features look like those of popular tracks, so no model can single them out, and they produce some of the largest errors.

According to permutation importance (Figure 18, left), the strongest drivers of popularity are **valence**, acousticness, duration, instrumentality and danceability, while the explicit flag and the time signature contribute almost nothing. The residuals (right) are centered on zero and roughly symmetric, so the model has no systematic bias toward over- or under-prediction.

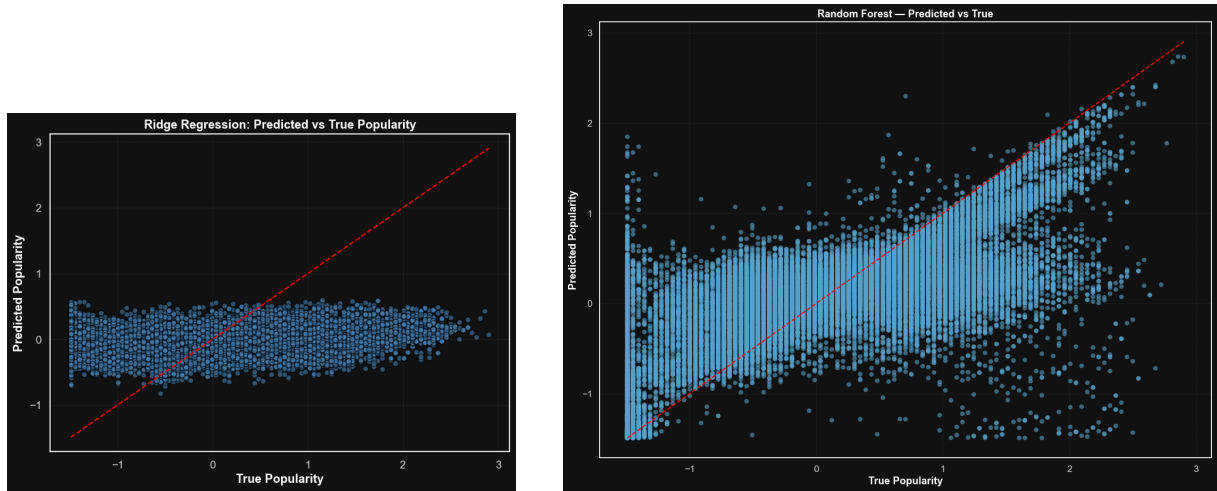


Figure 17: Predicted versus true standardized popularity for Ridge regression (left) and the Random Forest (right). The dashed line is the perfect prediction.

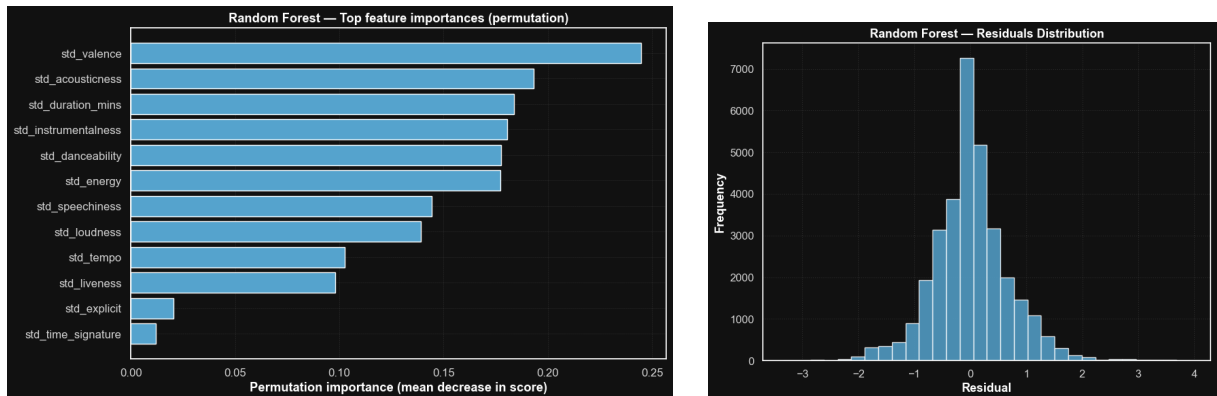


Figure 18: Random Forest. Left: permutation importance on the test set (10 repeats). Right: distribution of the test residuals.

8 Imbalanced learning: detecting explicit content

8.1 Task

The last task is to predict the **explicit** flag, for example to support a parental-control filter. The classes are strongly imbalanced: **99,181 clean tracks** and **9,118 explicit ones**, so only 8.4% of the tracks are explicit. The inputs are the other 12 standardized features plus the one-hot macro-genre and key, 46 columns in total, and the explicit flag is of course excluded from its own predictors. The data is split 80/20 with stratification, so the test set has 19,836 clean and 1,824 explicit tracks. On such data accuracy is misleading: a model that always answers “clean” scores 91.6%. All models are therefore tuned on **macro-F1** and compared on macro-F1, ROC-AUC and, above all, the precision and recall on the explicit class.

8.2 Resampling strategies

Four pipelines, all based on a Random Forest, were compared:

- **Baseline:** the forest on the original data, tuned over 24 configurations.
- **SMOTE** [8]: the minority class is oversampled with synthetic tracks. Each new track is generated on the segment between an explicit track $\mathbf{x}_i$ and one of its five nearest explicit

neighbors $\mathbf{x}_z$:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_i + \delta (\mathbf{x}_z - \mathbf{x}_i), \quad \delta \sim \mathcal{U}(0, 1), \quad (9)$$

until the two classes have the same size. Tuned over 12 configurations.

- **Tomek links and random under-sampling:** pairs of tracks of opposite classes that are each other’s nearest neighbor (Tomek links [29]) are cleaned by removing the majority member, then the majority class is randomly under-sampled to the size of the minority.
- **SMOTETomek** [4]: SMOTE followed by Tomek-link cleaning. Tuned over 4 configurations.

A subtle but important point is *where* resampling happens. All four pipelines are built with `imblearn.Pipeline` [17], so that during cross-validation the resampling is fitted only on the training folds:

```
pipeline = Pipeline(steps=[
    ("smote", SMOTE(random_state=42)),
    ("rf", RandomForestClassifier(random_state=42, n_jobs=-1)),
])
grid = GridSearchCV(pipeline, param_grid, scoring="f1_macro",
                    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42))
```

If the data were resampled *before* the split, synthetic tracks built from validation tracks would leak into training and the scores would be optimistic.

8.3 Results

Pipeline	Macro-F1	Macro P	Macro R	ROC-AUC	Explicit class	
					Recall	Precision
Baseline RF	0.760	0.924	0.695	0.945	0.395	0.901
SMOTE + RF	0.795	0.794	0.796	0.941	0.628	0.622
Tomek + under-sampling + RF	0.669	0.643	0.842	0.924	0.870	0.300
SMOTETomek + RF	0.795	0.794	0.795	0.941	0.625	0.623

Table 10: Explicit-content detection on 21,660 test tracks (macro P and R: macro-averaged precision and recall).

	Baseline RF		SMOTE + RF		Tomek + under + RF		SMOTETomek + RF	
clean	19,757 (99.6%)	79 (0.4%)	19,140 (96.5%)	696 (3.5%)	16,138 (81.4%)	3,698 (18.6%)	19,146 (96.5%)	690 (3.5%)
explicit	1,104 (60.5%)	720 (39.5%)	679 (37.2%)	1,145 (62.8%)	238 (13.0%)	1,586 (87.0%)	684 (37.5%)	1,140 (62.5%)
	clean	explicit	clean	explicit	clean	explicit	clean	explicit

Figure 19: Test confusion matrices of the four pipelines (rows: true class, columns: predicted class; percentages are per row).

Table 10 and Figures 19 and 20 summarize the results.

- **The baseline is precise but blind.** When it says “explicit” it is right 90% of the time, but it finds only 39.5% of the explicit tracks and misses 1,104 of them.
- **SMOTE gives the best balance.** It raises the explicit recall from 39.5% to **62.8%**, a **relative gain of 59%**, and the macro-F1 from 0.760 to **0.795**. The price is lower precision on the explicit class (0.901 to 0.622).

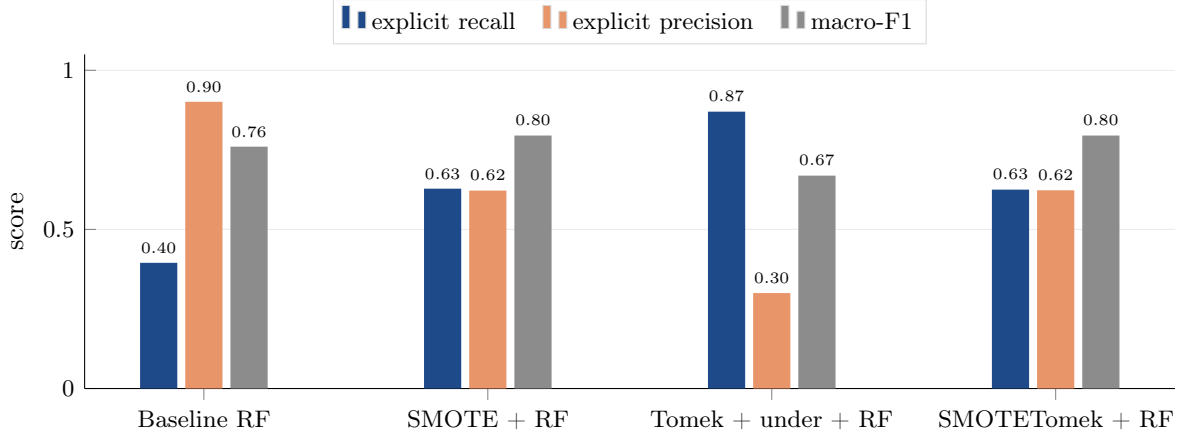


Figure 20: The precision–recall trade-off on the explicit class: each strategy moves the operating point along the same curve.

- **Tomek links with under-sampling maximize recall.** They catch 87% of the explicit tracks, but they raise 3,698 false alarms, 18.6% of all clean tracks, and their precision drops to 0.300.
- **SMOTETomek behaves like SMOTE:** the additional Tomek cleaning after oversampling has almost no effect on the result.

The ROC-AUC barely moves across the three best pipelines (0.941 to 0.945). This shows that resampling does not make the forest better at *ranking* explicit tracks above clean ones; it moves the operating point along essentially the same ROC curve (Figure 21). The right strategy therefore depends on the cost of each error:

- **SMOTE** is the best general-purpose trade-off;
- **Tomek links with under-sampling** fit applications where a missed explicit track is very costly, such as a parental-control filter, provided that false alarms can be reviewed;
- **the baseline** fits applications where false alarms must be minimal.

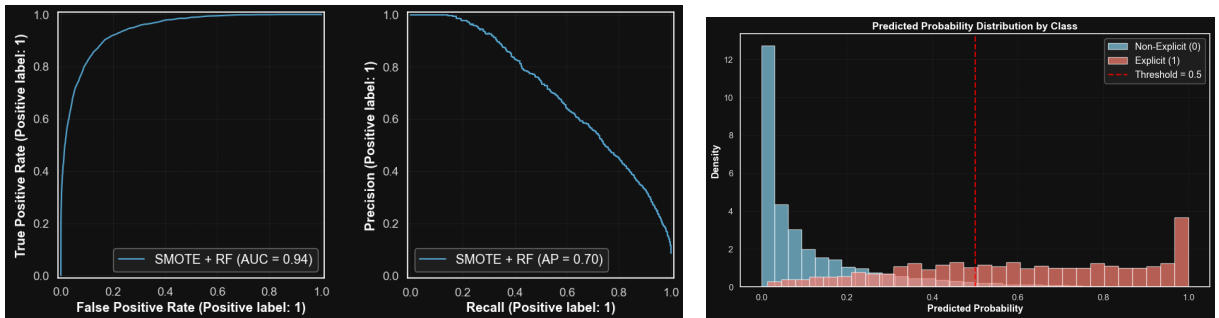


Figure 21: Left: ROC curve (AUC 0.94) and precision–recall curve (average precision 0.70) of SMOTE + RF on the explicit class. Right: distribution of the predicted probability of “explicit” for SMOTETomek + RF, by true class, with the default threshold of 0.5.

The probability distributions on the right of Figure 21 make the same point. Clean tracks concentrate near zero, while explicit tracks spread over the whole range with a peak at one. Moving the threshold away from 0.5 would trade precision for recall without retraining, which is a natural next step.

9 Implementation and reproducibility

The project is written in Python with pandas, scikit-learn [25], imbalanced-learn [17], XGBoost [9], SHAP [21] and TensorFlow/Keras [1]. It is organized as six notebooks that communicate only through CSV files, as shown in Figure 1:

1. `1_data_understanding_and_preparation.ipynb`: exploration, feature engineering and the genre taxonomy;
2. `2_outlier_detection.ipynb`: encoding, scaling, the k -distance curve, LOF and the PCA view;
3. `3_clustering.ipynb`: K-Means, inertia and silhouette, projections and the two validators;
4. `4_classification_explainability.ipynb`: Random Forest, the residual network, permutation importance, gradients and SHAP;
5. `5_regression.ipynb`: the five regressors with diagnostics and permutation importance;
6. `6_imbalanced_learning.ipynb`: the four resampling pipelines.

Several choices protect the results from data leakage: `duration_ms` is dropped after its conversion to minutes; the explicit flag is excluded from the predictors of its own task, and popularity from the regression inputs; resampling happens only inside cross-validation folds; and the deep network selects its best epoch on a validation split carved from the training data, never on the test set. Validation is consistently careful: stratified splits and stratified k -fold cross-validation, repeated k -fold for regression, out-of-bag estimates, and a check of the gap between training and test scores. Hyperparameters are chosen from the data: the k -distance curve sets LOF's k , the silhouette local maxima set the number of clusters, and grid searches tune the models, up to 14,700 fits for the decision tree. Random seeds are fixed to 42 for all splits and models.

The scikit-learn grid searches use all CPU cores, and the deep network was trained on an Apple M3 Ultra GPU (96 GB of unified memory) through TensorFlow-Metal. The following excerpt shows the learning-rate schedule of the network:

```
def lr_schedule(epoch):
    if epoch < warmup_epochs: # linear warm-up
        return initial_learning_rate * (epoch + 1) / warmup_epochs
    progress = (epoch - warmup_epochs) / (epochs_total - warmup_epochs)
    return initial_learning_rate * 0.5 * (1 + np.cos(np.pi * progress)) # cosine decay
```

10 Conclusions

SoundPulse followed 114,000 Spotify tracks from raw data to explained predictions, and each stage answered its question:

- **Anomalies.** LOF removed 5,700 atypical tracks (5%), many of them zero-popularity copies of known songs or exact duplicates listed under several genres.
- **Segments.** K-Means found nine sound families, and their reality was confirmed by a neural network (97.5% recovery accuracy) and an interpretable decision tree (92.6%), which showed that energy, instrumentality, explicitness, popularity and danceability define them.
- **Genres.** With 13 features, the Random Forest reaches 58.5% accuracy and a macro ROC-AUC of 0.898, and the deep network places the right genre in its top three for 84.1% of the tracks. Genres defined by sound (Metal, Latin, Classical) are recognized well; genres defined by geography (World & Regional) are not.

- **Explanations.** Three methods agree that acousticness, popularity, danceability and energy carry most of the signal, and local SHAP explanations make individual decisions, including mistakes, readable.
- **Popularity.** Uncorrelated with any single feature, it is still half predictable by tree ensembles ($R^2 = 0.529$, eighteen times the linear baseline): popularity lives in feature interactions.
- **Explicit content.** SMOTE raises the recall on the rare class by 59% at the cost of precision, and the choice among the strategies depends on the cost of each type of error.

Future work. Several extensions follow naturally from these results:

- removing cross-listed duplicates *before* the train/test split, so that the same recording can never appear on both sides;
- threshold tuning and probability calibration for the explicit-content detector;
- a focal loss or ordinal variants of the deep classifier to further lift the minority genres;
- hierarchical classification, from macro-genre to the 114 original labels;
- packaging the pipeline as a command-line tool with a dashboard for non-technical users.

References

- [1] M. Abadi et al. TensorFlow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016.
- [2] D. Arthur and S. Vassilvitskii. k -means++: the advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.
- [3] J. L. Ba, J. R. Kiros and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [4] G. E. A. P. A. Batista, R. C. Prati and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.
- [5] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [6] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.
- [7] M. M. Breunig, H.-P. Kriegel, R. T. Ng and J. Sander. LOF: identifying density-based local outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 93–104, 2000.
- [8] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [9] T. Chen and C. Guestrin. XGBoost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.
- [10] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.
- [11] C. Elkan. Using the triangle inequality to accelerate k -means. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 147–153, 2003.
- [12] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

- [13] P. Goyal et al. Accurate, large minibatch SGD: training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.
- [14] K. He, X. Zhang, S. Ren and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [15] D. Hendrycks and K. Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.
- [16] A. E. Hoerl and R. W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
- [17] G. Lemaître, F. Nogueira and C. K. Aridas. Imbalanced-learn: a Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.
- [18] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
- [19] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.
- [20] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.
- [21] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 4765–4774, 2017.
- [22] S. M. Lundberg et al. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1):56–67, 2020.
- [23] B. W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta – Protein Structure*, 405(2):442–451, 1975.
- [24] M. Pandya. Spotify Tracks Dataset. Kaggle, 2022. <https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset>.
- [25] F. Pedregosa et al. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [26] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.
- [27] K. Simonyan, A. Vedaldi and A. Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. In *ICLR Workshop Track*, 2014.
- [28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.
- [29] I. Tomek. Two modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(11):769–772, 1976.