UNIVERSITÀ
di VERONA

Department of Computer Science

Master Degree in Artificial Intelligence

Music Genre Classification and Analysis

A Study on the 30,000 Spotify Songs Dataset

Course: Machine Learning
Academic Year: 2025/2026

Author:
Alessio Brighenti
ID: VR541999

Professor:
Prof. Cigdem Beyan

Date: February 17, 2026

Abstract

The aim of this project is to explore Supervised and Unsupervised Learning techniques studied during the course applying them to music audio features. The chosen dataset is "30000 Spotify Songs" dataset, the main task of the project is the prediction of music genres based on technical attributes such as tempo, energy, danceability and so on. I compare different classifiers such as KNN, Random Forest, AdaBoost and analyze the intrinsic structure of the data using K-Means clustering and PCA. I have also implemented a 'Recommender System' as a real application machine learning techniques. The results highlight the complexity of genre classification task due to class overlap, but demonstrate the effectiveness of tree-based models over distance-based ones.

Contents

1 Motivation and Rationale

Music streaming platforms like Spotify have to manage millions of songs, quantities that are impossible to manage manually, which is why the use of machine learning techniques becomes useful.
This project focuses on topics covered by Music Information Retrieval (MIR) [6], the interdisciplinary science of retrieving information from music, in particular I focused on the following two :

1. **Automatic Genre Classification:** The task of assigning a genre label (e.g. Pop, Rock, R&B) to a song based only on its audio features without considering metadata such as artist, album, membership, etc. (Supervised Learning)

2. **Content-Based Recommendation:** A system capable of predicting what music a user might like based on their tastes and the similarity between songs. This is very useful for solving the so-called "Cold Start Problem," which focuses on recommending a song that has just been released and has no information about it.

The rationale behind this project is to investigate the extent to which mathematical features extracted from audio signals can capture the complex, subjective nature of musical genres. While metadata (Artist, Album) is useful, it is often incomplete or misleading. A robust classification model based on audio features ensures that tracks are organized by how they *sound*, rather than who performed them.

2 State of the Art (SOTA)

2.1 Music Genre Classification

As just mentioned, classification of music genres is a central task of MIR, which has moved from manual feature extraction to deep learning models and techniques.

1. **Feature-Based Approaches (Tabular Data):** Traditional methods rely on extracting hand-crafted features from the audio signal [2], such as Mel-Frequency Cepstral Coefficients (MFCCs), Spectral Centroid, and Zero-Crossing Rate. These features are then fed into classic Machine Learning algorithms like Support Vector Machines (SVM), k-Nearest Neighbors (KNN), and Random Forests. This project follows this established methodology, utilizing high-level audio features provided by the Spotify API (e.g., `danceability`, `energy`), which are essentially engineered summaries of the raw audio signal.

2. **Deep Learning Approaches (Raw Audio):** The current state-of-the-art involves Convolutional Neural Networks (CNNs) applied directly to visual representations of audio, such as Spectrograms or Mel-spectrograms [1]. These models treat audio classification as an image recognition problem.

3. **Music Recommender Systems:** Recommender systems in the music industry [4] is typically divided into two categories: Collaborative Filtering (CF), that is the industry standard (e.g., "Users who listened to X also listened to Y") and Content-Based Filtering (CBF) that recommends tracks based on their intrinsic properties such as audio features (the method implemented in this project).

2.2 Limitations of Current Methods

The main limitation of these methods is the ambiguity of the ground truth labels. Genre boundaries could be subjective and they often overlap with each other, so Supervised Learning models struggle with this label noise.

3 Objectives

The general objective of the project is to build a robust Machine Learning pipeline for music analysis. Specific objectives include:

- **Data Pre-processing:** Fundamental phase of cleaning and transforming raw data into a structured, consistent format suitable for analysis or machine learning.
- **Classification:** Compare the accuracy of KNN, Random Forest, and AdaBoost in predicting the 6 main genres.
- **Clustering:** Investigate if genres form natural clusters using K-Means.
- **Real-Life Application:** Develop a simple Recommender System based on Cosine Similarity (Nearest Neighbors).

4 Methodology

4.1 Dataset Description

The concerned dataset is "30000 Spotify Songs" [5] from `Kaggle` containing approximately 32,000 rows and 23 columns (structural details of the dataset in Table Table 1).

4.2 Data Preprocessing

The following preprocessing steps were applied to the previously mentioned dataset:

1. **Stratified Sampling:** From the total 32,000 rows, 10,000 rows were extracted to perform the previously mentioned analyses. The rows were extracted through stratification to obtain a balanced subset. Stratification have been applied with `train_test_split` function from `scikit-learn` library for Python, in particular `stratify=dataset['playlist_genre']` parameter allowed to obtain a subset with the same genres distribution of the original dataset.
2. **Handling missing values:** Any missing values have been removed using `dropna` function on the previously extracted subset.
3. **Featrue Selection:** Any irrelevant columns such as track ID, track name, track artist, etc. have been removed. The main reasons are: unique categorical columns can cause 'Curse of Dimensionality' if encoded, the models should learn only from audio features not from metadata.
4. **Standardization**: Applied using `StandardScaler` for distance-based models (KNN Classifier, Clustering with K-MEANS).

Feature Name	Data Type	Description
	Metadata (Identifier & Context)	
track_id	String	Unique ID for the track on Spotify
track_name	String	Title of the song
track_artist	String	Name of the artist
track_popularity	Double	Popularity score (0-100)
track_album_id	String	Unique ID for the album
track_album_name	String	Name of the album
track_album_release_date	String	Date of release (YYYY-MM-DD)
playlist_name	String	Name of the playlist containing the track
playlist_id	String	Unique ID for the playlist
playlist_genre	String	**Target Variable**: Main genre (e.g., Pop, Rock)
playlist_subgenre	String	Specific sub-genre (e.g., Electropop)
	Audio Features (Numerical Input for ML)	
danceability	Double	How suitable a track is for dancing (0.0 to 1.0)
energy	Double	Intensity and activity measure (0.0 to 1.0)
key	Double	The estimated overall key of the track (0 to 11)
loudness	Double	Overall loudness of a track in decibels (dB)
mode	Double	Modality (0 = Minor, 1 = Major)
speechiness	Double	Presence of spoken words (0.0 to 1.0)
acousticness	Double	Confidence measure of acoustic sound (0.0 to 1.0)
instrumentalness	Double	Likelihood the track contains no vocals (0.0 to 1.0)
liveness	Double	Probability that the track was performed live
valence	Double	Musical positiveness (0.0 = sad, 1.0 = happy)
tempo	Double	Estimated tempo in Beats Per Minute (BPM)
duration_ms	Double	Duration of the track in milliseconds

Table 1: Complete list of features in the dataset with data types and descriptions.

4.3 Algorithms and Models

As previously mentioned, the Python language was used to develop the code, in particular the use of the `scikit-learn` library [3] was fundamental.
As already mentioned, different techniques studied during the course were compared.

4.3.1 Supervised Learning: Classification with Tree-Based Ensemble Methods

To tackle the multi-class classification problem of assigning genres to songs, we selected two ensemble algorithms based on Decision Trees: `Random Forest` and `AdaBoost`. Tree-based models were chosen over linear models (like Logistic Regression) because music data often exhibits non-linear relationships. For instance, a high `tempo` might indicate "EDM" only if `energy` is also high; otherwise, it might be a fast "Pop" song. Decision trees can naturally capture these "if-then" interactions. Let's see some implementation and configuration details:

- **Random Forest (Bagging):** Random Forest applies the Bagging technique. It trains multiple decision trees in parallel on different random subsets of the data and averages their predictions. The model have been instantiated with `n_estimators=50` (number of trees) and utilized Gini Impurity as the splitting criterion (default criterion).

- **AdaBoost (Boosting):** Adaptive Boosting (AdaBoost) applies the Boosting technique. Unlike Random Forest, it trains trees sequentially. Each new tree focuses on correcting the errors made by the previous ones by assigning higher weights to misclassified instances. The model have been instantiated with `n_estimators=50` and standard Decision Tree as the base estimator.

4.3.2 Supervised Learning: Classification with Distance-based Method

To establish a baseline for the classification task, I also used the K-Nearest Neighbors (KNN) model. KNN is a non-parametric learning algorithm that classifies a new data point based on the majority class of its k closest neighbors in the feature space.
Unlike tree-based models, KNN relies on Euclidean distance to measure similarity between data points. In the Spotify dataset, features have vastly different ranges:

- `duration_ms`: ranges in the hundreds of thousands (10^5).
- `tempo`: ranges around 60-200 (10^2).
- `acousticness`: ranges from 0 to 1 (10^0).

So as mentioned in the data preprocessing section we need to normalize the features. Without normalization, features with larger magnitudes (like duration) would dominate the distance calculation, rendering smaller features (like acousticness) irrelevant, furthermore, the dataset appears sparse, making the distance calculation meaningless. To address this, we applied Standard Standardization (`StandardScaler`) to all numerical features before training, ensuring each feature contributes equally to the distance metric with a mean of 0 and a standard deviation of 1.
The performance of KNN is highly sensitive to the choice of k (the number of neighbors):

- A small k leads to a complex model prone to overfitting (capturing noise).
- A large k leads to a simple model prone to underfitting (smoothing out decision boundaries too much).

In order to address this problem, i performed an iterative grid search, testing values of k ranging from 20 to 70. As shown in Figure 1 (Left), the accuracy initially increases as the model generalizes better, reaching a plateau. The algorithm automatically selected the optimal k (e.g., $k = 57$) which maximized validation accuracy.

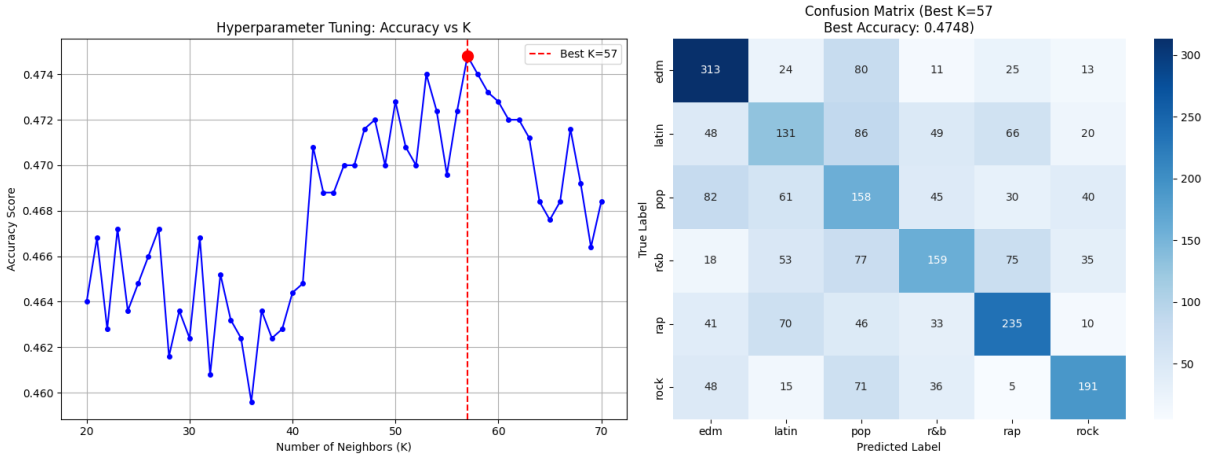


Figure 1: KNN Analysis: Hyperparameter tuning for k (Left) and Confusion Matrix of the best model (Right).

4.3.3 Unsupervised Learning: Clustering with K-MEANS

To investigate whether the 6 interested musical genres form natural, separable groups in the feature space, we applied K-Means Clustering. Unlike the previous supervised tasks, here the model does not see the ground truth labels (y), forcing it to group songs solely based on their audio similarities.
Since K-Means is a distance-based algorithm, feature scaling is mandatory.

- **Standardization:** I applied `StandardScaler` to the full set of 12 numerical features to ensure that features with larger ranges (e.g., `loudness`, `tempo`) do not dominate the clustering process.
- **High-Dimensional Clustering:** We chose to fit the K-Means algorithm on the **original 12-dimensional scaled space**, rather than on reduced PCA components. This preserves 100% of the variance and prevents information loss during the grouping phase (if clustering is applied on the reduced 3-dimensional space, performances significantly worsen.).
- **Visualization (PCA):** To interpret the results visually, we employed **Principal Component Analysis (PCA)** to project the high-dimensional clusters into a 3D space. The first 3 Principal Components (PC1, PC2, PC3) captured the most significant variance directions, allowing us to inspect the separation between groups.

4.3.4 Application: Content-Based Recommender System (EXTRA)

As a final practical application of my analysis, I wanted to try to implement a Content-Based Recommender System. Unlike Collaborative Filtering methods (which rely on user listening history matrices), the implemented system suggests songs solely based on their audio feature vector similarity. This approach addresses the well-known "Cold Start Problem": it allows the system to recommend new or unpopular tracks immediately, provided their audio analysis is available.
For the recommendation engine, i utilized the `NearestNeighbors` algorithm in an unsupervised setting. The main design choice was the selection of the distance metric. While we used Euclidean distance for the KNN classifier, for the recommender system we opted for `Cosine Similarity`:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} = \cos(\theta) \tag{1}$$

- **Why Cosine?** In high-dimensional feature spaces, the magnitude of the vectors is often less important than their direction. Cosine similarity measures the angle between two feature vectors. Two songs are considered "similar" if they point in the same direction in the feature space, representing a similar "balance" of attributes (e.g., high energy relative to low acousticness), regardless of the absolute scale.
- **Implementation:** We sampled 10,000 tracks to create a diverse search space and applied `StandardScaler` to normalize the features before computing the similarity matrix.

5 Experiments and Results

In this section, i analyze the quantitative and qualitative results obtained from the models described in the Methodology. All experiments were conducted using a stratified train-test split (75% training, 25% validation) to ensure class balance. The primary metric for classification is `Accuracy`, supplemented by the `Confusion Matrix` to analyze class-specific errors.

5.1 Tree-Based Classification (Random Forest vs. AdaBoost)

We compared two ensemble methods: Random Forest (Bagging) and AdaBoost (Boosting).

5.1.1 Quantitative Comparison

As summarized in Table 2, Random Forest outperformed AdaBoost in terms of overall accuracy.

Table 2: Performance comparison of Tree-based models.

Model	Accuracy	Key Observation
Random Forest (n=50)	**52.6%**	Robust to noise, stable predictions
AdaBoost (n=50)	43.7%	Slight overfitting on outliers

5.1.2 Error Analysis

The Confusion Matrix (Figure 2) for the best Random Forest model reveals distinct patterns:

- **High Performance:** The model performs best on **Rock**, **Rap** and **EDM**. These genres have distinct acoustic footprints from each other.
- **Main Confusion:** The most significant errors occur between **Pop** and **Latin** or **Rap** and **R&B**. These genres share very similar values in the analyzed features, making them hard to distinguish without lyrics analysis.

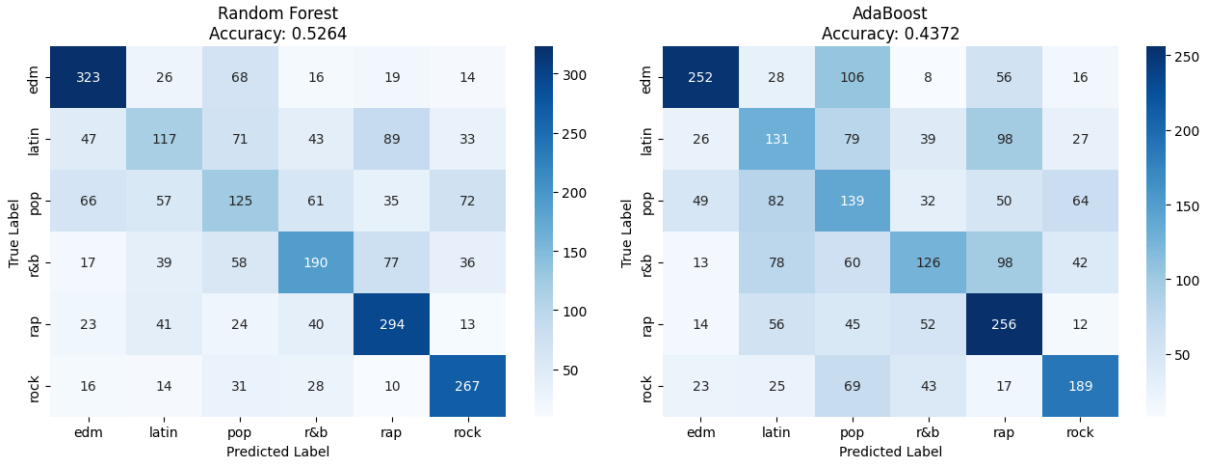


Figure 2: KNN Analysis: Hyperparameter tuning for k (Left) and Confusion Matrix of the best model (Right).

5.1.3 Feature Importance

Furthermore the Random Forest analysis highlighted that `popularity`, `acousticness`, and `danceability` are the top 3 drivers for classification. Conversely, `key` and `mode` were found to be statistically irrelevant, suggesting that musical tonality does not define a genre in this dataset.

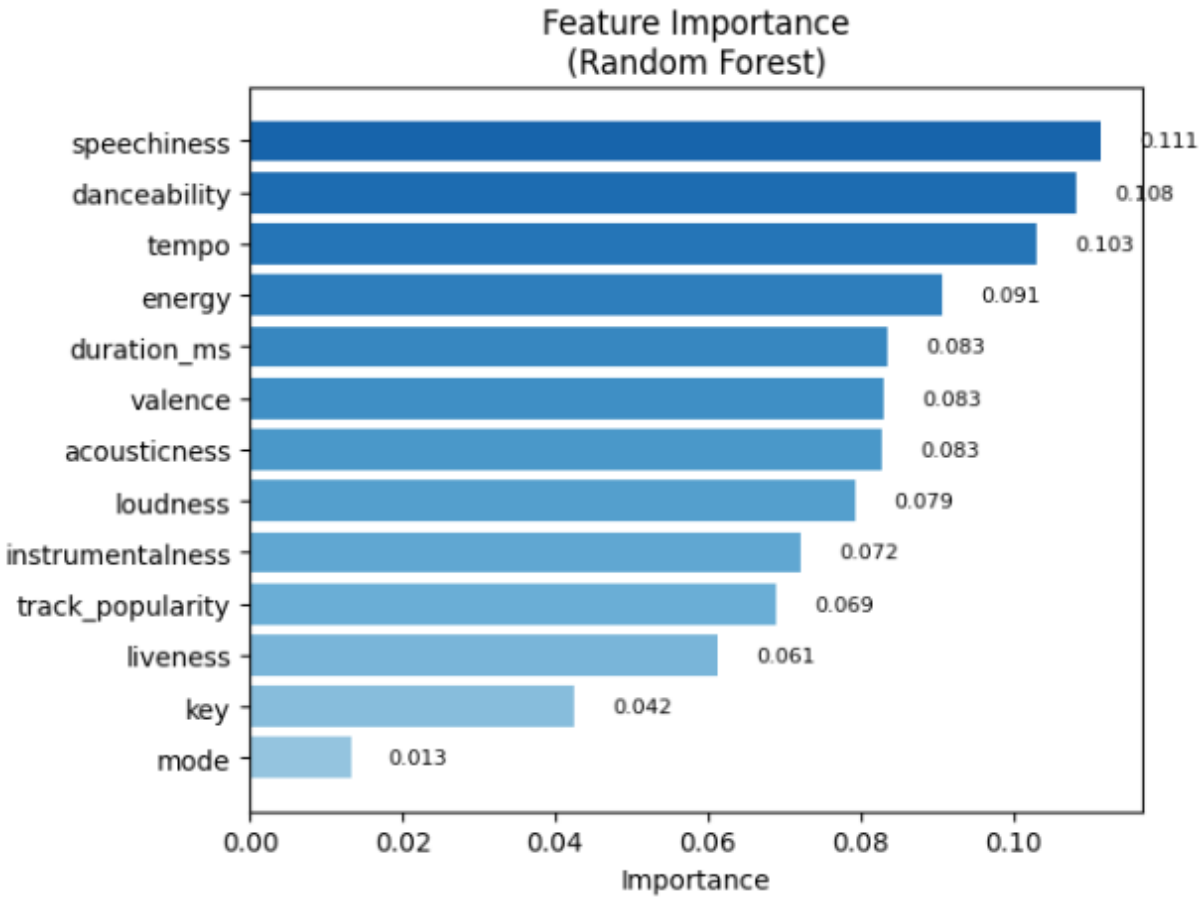


Figure 3: Feature importance obtained by Random Forest model.

5.2 Distance-Based Classification (KNN)

We evaluated the K-Nearest Neighbors classifier after applying Standard Scaling to the features.

5.2.1 Hyperparameter Tuning

As previously said a grid search for $k \in [20, 70]$ have been performed.

- The accuracy peaked at approximately **48%** with $k = 57$ (Figure 1, left).
- Values of $k < 10$ resulted in lower validation accuracy due to overfitting (capturing local noise).
- Values of $k > 60$ caused the model to over-smooth decision boundaries (underfitting).

5.2.2 Comparison with Trees

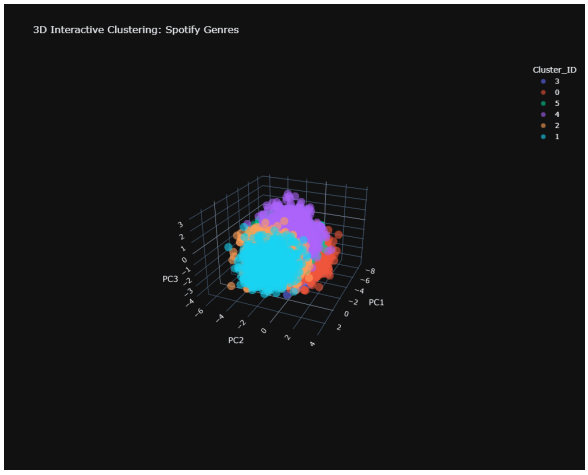
KNN achieved slightly lower accuracy compared to Random Forest (-4/5%). This performance gap could be caused from the "Curse of Dimensionality": in a 12-dimensional space, data points become sparse, and the concept of Euclidean distance becomes less discriminative than the hierarchical decision boundaries drawn by trees.

5.3 Unsupervised Clustering (K-Means)

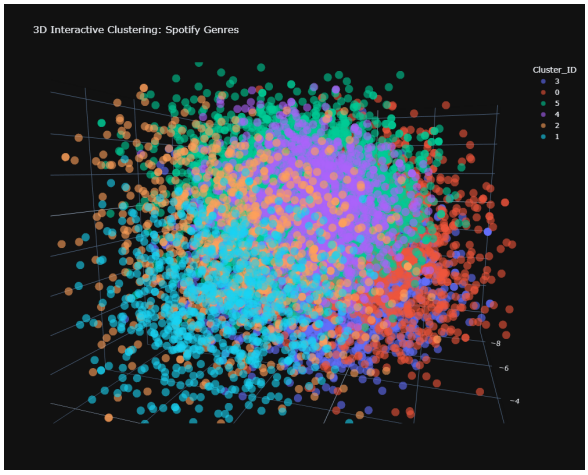
To explore the intrinsic structure of the data, we applied K-Means with $K = 6$ on the full scaled dataset.

5.3.1 Cluster Separation

The clustering results were evaluated using the Silhouette Score which was approximately 0.10. Visualizing the data in 3D using PCA (Principal Component Analysis) confirmed that the genres do not form separate clusters. More precisely, if the graph is viewed from afar it appears to show separate point clouds, but viewing the points closer reveals that the clusters are mixed together and difficult to distinguish (Figure 4).



(a) Global View: From a distance, the 6 clusters appear to occupy distinct regions of the PCA space.



(b) Local View (Zoom): Upon closer inspection, boundaries blur significantly, showing high overlap between genres.

Figure 4: 3D Visualization of K-Means Clustering ($K = 6$). The comparison highlights the "Genre Continuum" hypothesis: while macro-structures exist, local boundaries are ambiguous.

Key Finding: This negative result is significant. It mathematically demonstrates that "Genre" is a subjective, cultural label that often overlaps in the audio feature space.

5.4 Recommender System Application

Finally, we qualitatively evaluated the Content-Based Recommender System using Cosine Similarity.
I evaluated the recommendation system through some manual testing by myself. During these tests, I encountered some performance issues related to the dataset's content, such as duplicate songs (an artist may release a song first as a single and then as part of an album, so I assume this is the reason). Consequently, the system recommends the song itself (correctly, based on similarity). Aside from this duplication issue, the system, in my opinion, responds to requests with an accuracy of about 50%, confirming the results obtained by previously tested models.

6 Conclusions

This project set out to investigate the feasibility of automatic music genre classification and recommendation using tabular audio features from Spotify. Through a comparative analysis of Supervised and Unsupervised Learning techniques, several key insights were derived regarding the relationship between mathematical signal attributes and human-defined musical genres.

6.1 Summary of Findings

1. **Tree-Based Models are Superior:** Random Forest proved to be the most effective classifier (Accuracy $\approx 50\%$), outperforming both AdaBoost and the distance-based KNN.

Its ability to handle non-linear decision boundaries and ignore irrelevant features makes it the robust choice for this dataset.

2. **The "Genre Continuum":** The most significant finding—supported by both the Confusion Matrices and the low Silhouette Score in clustering—is that musical genres do not form distinct, separable clusters in the audio feature space. Genres like *Pop*, *Latin*, and *EDM* share a vast amount of structural similarity (high energy, high danceability), making the distinction really difficult.

3. **Feature Relevance:** Our analysis confirmed that rhythmic and intensity features (`energy`, `danceability`, `loudness`) are strong predictors of genre, whereas tonal features (`key`, `mode`) are statistically irrelevant.

4. **Cold Start Solution:** The Content-Based Recommender System demonstrated that even without historical user data, cosine similarity on audio features can successfully retrieve semantically related tracks, offering a viable solution for new music discovery (Still considering the limitations of the case).

6.2 Limitations and Future Work

The primary limitation of this study is the reliance on high-level tabular features, which abstract away the the detailed qualities of the sound.
Future developments to improve classification performance could include:

- **Deep Learning on Raw Audio:** Utilizing Convolutional Neural Networks (CNNs) on Mel-Spectrograms to capture timbral textures (e.g., distorted guitars vs. synthesizers) that tabular data misses.

- **Natural Language Processing (NLP):** Integrating lyrics analysis to better distinguish linguistically distinct genres like *Rap* and *R&B*, which often sound instrumentally similar.

- **Hierarchical Classification:** Implementing a two-step classification (e.g., first classify "Electronic vs Acoustic", then sub-genres) to reduce the confusion between macro-categories.

References

[1] Yandre M.G. Costa, Luiz S. Oliveira, and Carlos N. Silla Jr. "An Evaluation of Convolutional Neural Networks for Music Classification Using Spectrograms". In: *Applied Soft Computing* 52 (2017), pp. 28–38. DOI: 10.1016/j.asoc.2016.12.024.

[2] Zhouyu Fu et al. "A Survey of Audio-Based Music Classification and Annotation". In: *IEEE Transactions on Multimedia* 13.2 (2011), pp. 303–319. DOI: 10.1109/TMM.2010.2098858.

[3] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] Markus Schedl et al. "Music Recommender Systems". In: *Recommender Systems Handbook*. Springer, 2015, pp. 453–492. DOI: 10.1007/978-1-4899-7637-6_13.

[5] Joakim Arvidsson Thompson. *30000 Spotify Songs Dataset*. https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs. Kaggle Dataset. 2020.

[6] George Tzanetakis and Perry Cook. "Musical Genre Classification of Audio Signals". In: *IEEE Transactions on Speech and Audio Processing* 10.5 (2002), pp. 293–302. DOI: 10.1109/TSA.2002.800560.