Music Genre Recognition with Extreme Gradient Boosting

Ritwik Awasthi, Junichi Takano, Kyrus Wankadiya

Abstract
The purpose of this paper is to showcase a model that classifies a given set of musical audio files into ten genres as accurately as possible. MGR (Music Genre Recognition) is widely used in industry, from recommender systems to real-time analysis, improvements to this technology can result in better user experiences for products and services that depend on MGR. Our XGBoost model achieves superior performance compared to state-of-the-art models on the GTZAN leaderboard without using deep learning architectures.

Introduction

Automatic music genre classification has emerged as a cornerstone of music information retrieval, enabling efficient cataloging, recommendation, and organization of vast digital audio collections. Traditional approaches have relied on feature-based methods, extracting timbral, rhythmic, and harmonic descriptors—such as Mel-frequency cepstral coefficients, chroma features, and spectral contrast—to characterize the stylistic nuances of each track. In this study, we undertake a comparative evaluation of three supervised learning algorithms—extreme gradient boosting (XGBoost), k-nearest neighbors (KNN), and support vector machines (SVM)—for multiclass genre recognition on the GTZAN dataset[1]. By systematically tuning model hyperparameters and investigating the impact of feature selection and data augmentation strategies, we demonstrate how each classifier balances robustness to noise and predictive accuracy. Our results provide clear guidance on algorithmic trade-offs and pave the way for more adaptive, real-time music classification systems.

Data

For this project, we used the GTZAN Genre Collection from Kaggle[1], a dataset comprising of 1000 30-second audio clips in .au format, and sorted into ten genres: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock.

The dataset was largely clean, however differences in recorded audio exist, including the volume, sound quality, and recording techniques varying from song to song. There is also an inherent question about the "official" classification of songs in the dataset. While some songs may clearly belong to one genre, there are plenty of songs which may be a fusion of multiple styles or take influences from multiple genres. Classification for a genre like rock for instance, is inherently difficult as different eras of the genre share some similarities with blues, metal, pop, or country. Due to these overlaps, some amount of misclassifications are to be expected.

Despite being one of the most popular datasets for music genre classification, there is some inherent data leakage in the dataset as artist and song metadata is not available[2]. Therefore, we are utilizing a carefully split training and test dataset, restricting data leakage wherever possible, given the inherent faults within the original dataset.

Features

For all features that output several values across a n-second split of the audio signal, we use the mean, standard deviation, minimum, and maximum values to capture as much information from the audio signature.

Majority of the features were extracted using python along with the librosa package.

MFCCs or Mel-Frequency Cepstral Coefficients

- MFCC's are derived from the Spectrogram of an audio file. A spectrogram is a visual representation of the power of frequencies of a signal across time. This data contains several genre "identifying" features as songs composed in different time signatures, scales, and octaves will have a vastly different spectrogram signature. We can calculate the spectrogram of a signal by using a discrete fourier transform to separate an audio signal into its constituent frequencies.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi \frac{k}{N} n}$$

Figure 1: Discrete Fourier Transform

- In order to convert our spectrogram to MFCC, we apply a triangular bank filter to shift the frequencies into a mel scale, which is a better representation of how human ears interpret audio signals.

- In order to reduce dimension and extract the major identifying coefficients of our mel-spectrogram, we apply a log transformation and a discrete cosine transform.

- A series of variable amounts of coefficients that summarize textures of the sound such as the volume, energy, frequency patterns, and finer details. In general, MFCC0 would be the broadest snapshot of the frequencies, while high numbers capture finer details.

Chroma Features

- A series of 12 variables, one for each note in an octave, that reveals what notes are being played. Given the constituent frequencies of an audio signal, we can utilize the formula to map each frequency to one of 12 keys on an idealized piano (88 keys with A4 tuned to 440 hz).
- The result of this transformation is a 12-variable vector, representative of the key signature of the audio signal.
- The librosa package splits the file into 22k samples per second, where one pitch can be determined for each sample, mapping out a spectrum, which can be transformed and then mapped to the 12-variable chroma vector.
- C, C#, D, D#, E, F, F#, G, G#, A, A#, B are used as data points in the model. This would reveal common notes or patterns of notes, however the same note in a different octave would be treated the same.

Spectral Centroid

- Spectral centroids are representative of the center of mass of the audio signal. Brighter audio would lean towards higher frequencies while darker audio would have a center of mass in the lower frequencies.

Spectral Bandwidth

- Variance of the power spectral density around the spectral centroid.
- This is representative of how tightly packed the frequencies are in an audio signal. Tightly packed frequencies are seen when clean notes are played, whereas a wider bandwidth is seen when distortion or noise is added.
  - Pure sine wave: tight bandwidth.
  - White noise: wide bandwidth.
- Ambient audio, classical or jazz, would generally have lower bandwidth, while noisier audio with more dynamic energy (or many different frequencies at the same time) like would be found in metal would have higher bandwidth.

ZCR or Zero Crossing Rate

- The rate of signal changes. A high ZCR would indicate dynamic and abrupt sounds. Similar in a lot of ways to Spectral Bandwidth, in that it indicates noise, but instead of determining how many different frequencies at once, ZCR is focused on how quickly the signals change

RMSE or Root Mean Square Energy

- Measures how loud the overall sound is, or the average power of the audio signal.

Spectral Rolloff

- The frequency below a certain percentage of the total energy of the audio. Default for Librosa is 85
- Used to discretize spectral bandwidth into bins.

Onset Envelope

- Tracks sudden changes in the audio signal to detect the beat and rhythm.

Tempo

- Beats per minute of the soundtrack.

Spectral Contrast

- A series of 7 variables representing frequencies split into bands such as:
  - Lowest band: 20-60 Hz
  - Low: 60-250 Hz
  - Low-mid: 250-500 Hz
  - Mid: 500Hz - 2 kHz
  - Upper-Mid: 2-4 kHz
  - High: 4-6 kHz
  - Highest band: 5-20 kHz
- Within each band, Spectral Contrast represents how dynamic the audio is. If there are peaks and valleys in the frequencies in an audio file, it will have higher spectral contrast than one that is more monotone.

Tonnetz

- A 2d-graph representation of the audio signal that captures harmonic relationships between notes to help determine chord progressions, tonality, complexity, use of major or minor chords, and more defining characteristics of a genre.
- The six Tonnetz values are as follows
  - Movement by fifths (chords that are five notes apart)
  - Minor Thirds (darker steps)
  - Major Thirds (brighter steps)
  - Relative Major/Minor (changes between major and minor a lot)

  - Parallel Major/Minor (changes between major and minor a lot, but on the same note- i.e. C major to C minor)
  - Chromatic Motion (more complex and unpredictable movements)

- Tonnetz captures the consonance (in key) and dissonance (out of key) of an audio signal.

Augmentation

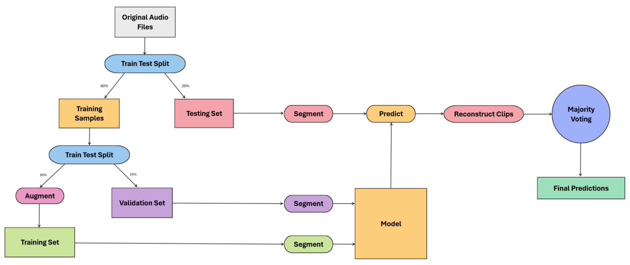


Figure 2: Model Pipeline

A few different methods of data augmentation were used to improve the model. First, in order to increase the sample size of the model, each audio file was split into six 5-second clips. Then copies were made of each 5-second clip each using one of four shifts:

- Time stretch of 0.9 (slower)
- Time stretch of 1.1 (faster)
- Pitch step of -2 semitones (lower pitch)
- Pitch step of +2 semitones (higher pitch)

By generating augmented tracks, our model is better equipped to prevent overfitting. Introducing pitch and time shifted tracks allows our model to generalize better, therefore, we are able to focus on capturing the key features when classifying, while ignoring noise in the data. A number of different values were tried for time stretches, pitch shifts, length of clips, and the number of MFCCs, and these proved to yield the most accurate results.

In experimenting with different models, we found that certain models excel with 5 second splits (KNN) and others perform better on 3 second splits (XGB and SVM) when classifying genre. This difference is further exacerbated when we combine the constituent splits back into the original audio file for majority voting classification.

Introducing augmented data into the training pipeline can often lead to data leakage, thus, we ensure that any audio file in the test dataset is unseen by the model when training, this includes any augmented versions of that specific track. The testing and validation data sets do not include any augmented tracks.

KNN and SVM Model

A KNN (K Nearest Neighbors) model was constructed using multiple values for seconds per split of the audio file, as well as grid searches for parameters n-neighbors, weights for uniform and distance, euclidean or manhattan, and hyperparameters for the SVMs. The first pass through, without any augmented audio, no tonnetz features, 13 MFCCs, and 3-second clips yielded models of accuracy rates in the high 50s to mid 60s percent.

As we added features, boosted the number of MFCCs to various numbers up to 64, added majority voting, added augmented audio to the inputs, continued to tune the hyperparameters using grid searches on each model, and finalized the use of 5-second clips per input, accuracy of the model continued to jump steadily. In addition, feature selection was performed based on the results of permutation feature importance. The majority voting technique, taking the popular vote of 10 3-second clips belonging to the 30-second original clip, yielded an accuracy of 82% using the SVM model on our test sample of 200 songs across the ten genres.

Extreme Gradient Boosting

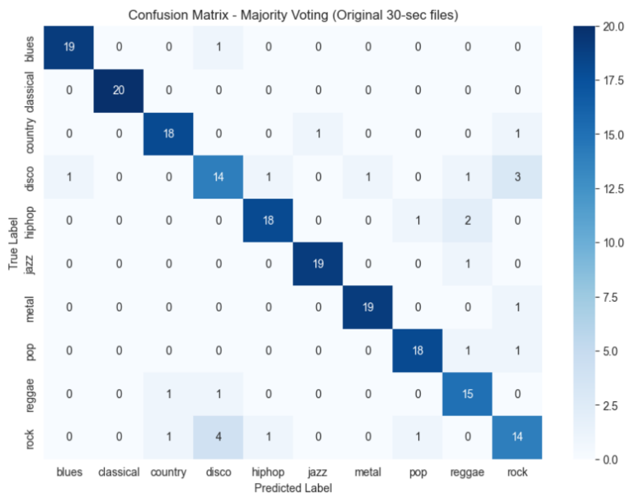
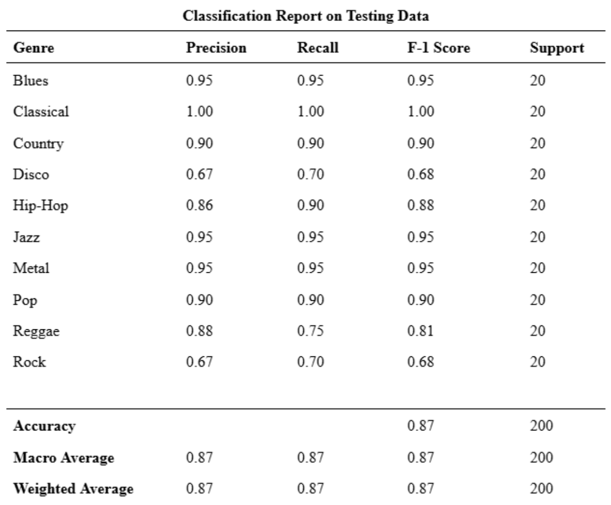


Figure 3: XGboost Confusion Matrix

We constructed an XGBoost classifier using 3 second splits of our audio files and 64 MFCCs. We utilized GridSearchCV for hyperparameter tuning, while limiting the max depth of our boosted trees to 3 in order to ensure we create "weak" learners. Early stopping conditions were implemented to avoid unnecessary computation via a validation set consisting of 10% of the training samples for evaluation. 1000 estimators were used with a learning rate of 0.18.

In order to classify the original 30 second track as a whole, we use a majority voting classifier across each audio clip's constituent splits. The majority vote is classified by taking the most common predicted label and assigning it to the audio file. Our model showcases significant improvement over state of the art neural network and deep learning models[3] over the same dataset for classification of the full 30 second clips.



Classification Report on Testing Data

Genre	Precision	Recall	F-1 Score	Support
Blues	0.95	0.95	0.95	20
Classical	1.00	1.00	1.00	20
Country	0.90	0.90	0.90	20
Disco	0.67	0.70	0.68	20
Hip-Hop	0.86	0.90	0.88	20
Jazz	0.95	0.95	0.95	20
Metal	0.95	0.95	0.95	20
Pop	0.90	0.90	0.90	20
Reggae	0.88	0.75	0.81	20
Rock	0.67	0.70	0.68	20
Accuracy			0.87	200
Macro Average	0.87	0.87	0.87	200
Weighted Average	0.87	0.87	0.87	200

Figure 4: Classification Report

Furthermore, we were interested in determining where this model struggles and how future work could benefit the classification task at hand. The following visualizations may provide insight into where and how this model misclassified data by genre, we will only look at the 3 worst performing genres in regards to precision: Disco, Hip-hop, and Rock.

Criticism

Issues with misclassification can be attributed to inherent issues within the dataset, as well as the dynamic nature of music and genres. Genres that share similarities are more likely to be misclassified, for example Rock and Country (similar time signatures, instruments, and musical scales.). More work is needed to be able to generate features capable of differentiating these misclassifications.

Conclusion

In this paper we explored several models tasked with Music Genre Classification on the GTZAN dataset. Our best performing model showcases state of the art performance without utilizing any deep learning architectures. Future work involves feature engineering to better distinguish between similar genres, and we would propose using a dataset better curated for MGR tasks, despite GTZAN being a popular choice.

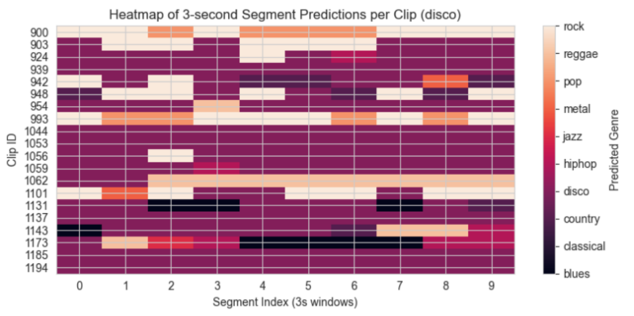


Figure 5: Disco

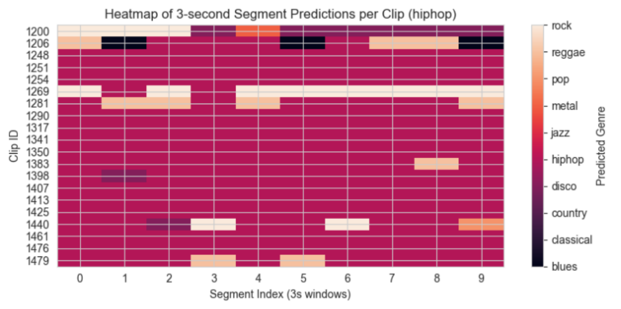


Figure 6: Hip-Hop

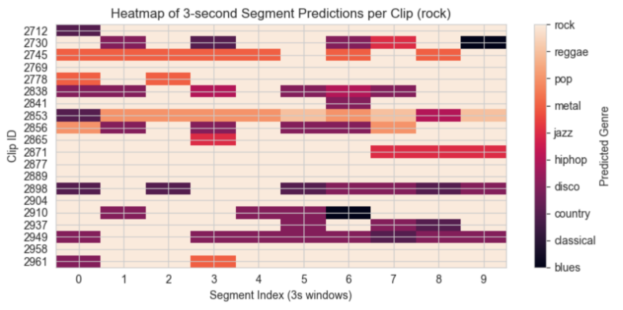


Figure 7: Rock

References

1. GTZAN Genre Collection dataset: Kaggle. Available from: https://www.kaggle.com/datasets/carlthome/gtzan-genre-collection
2. Sturm, B. L. et. al. The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. arXiv.org (e-prints) 2013 :1–29. Available from: http://arxiv.org/abs/1306.1461
3. Papers With Code: GTZAN Benchmark. 2025. Available from: https://paperswithcode.com/sota/music-genre-classification-on-gtzan