Do Machines Think Equally About Music?*

Testing an SVM's Classification Ability on Machine-Generated Music

Luca Carnegie

May 1, 2025

This paper demonstrates a systematic workflow for measuring how well a Music Generation Model, such as Meta's MuseGen, has learned the patterns of a particular genre or style of music. First, several classifier models are trained in Python, including an SVM, K-nearest-neighbors, Random Forest, and XGBoost, with the SVM being the most balanced at classification across genres. Then, the smallest version of Facebook's MuseGen model is systematically prompted through the MusicGPT interface to generate music 'quintessentially' describing one of ten genres of music. Finally, the features of this music are extracted using Python and then individually inputted into the SVM model, with the trained model in this workflow setup unable to classify any of the generated music clips. This paper highlights the need for further work on how machine learning algorithms embed understanding about musical features like timbre and how machine-generated music can impact existing music information retrieval systems, providing directions for future work in this important area. These points are particularly pertinent in light of extensive developments in machine-generated music.

Table of contents

*Code, workflow and prompts are available at: https://github.com/lcarnegie/song-classification. Many thanks to Periklis Andritsos for your supervision, guidance and excellent conversation.

1 Introduction

The advent of machine learning was a transformative force in the field of music information systems. Through advanced algorithms, previously un-automatable tasks suddenly became automatable, opening up deeper and more useful insights about the nature of music and how it fits within the distinctly human concept of musical genre. While genre classification quickly became a canonical problem within the context of machine learning, the concept of a music's genre still carries social and cultural weight within discussions about music and the artists that make it.

Recently, AI-powered tools like Suno ("Suno: AI-Powered Music Generation Platform" 2025) have emerged, targeting consumers interested in "creating" their own music. This raises questions about how machines understand music, especially in the case of generative models such as Suno's. While more research-based proofs of concept like Meta's MuseGen (Shi et al. 2023) have existed for years, tools like Suno adopt a more consumer-focused approach that inherently makes the algorithms that drive the product even more of a "black box" than before. But how does a word entered into a prompt box translate into the auditory representation humans understand as music? Do different artificial intelligence systems have different conceptualizations of commonly accepted human concepts of genre?

To understand if there is a different between conceptualizations of genre between different machine learning systems, we train several different machine learning algorithms in the task of predicting genre, based on a dataset of the timbral audio features of songs across 10 different genres extracted from the Free Music Archive (Defferrard et al. 2017). Then, we ask the 'small' variant of Meta's MuseGen Music Generation Model to create 'quintessential' representations of different genres of music using systematic prompts, and test if the most balanced algorithm can accurately predict the genre based on the extracted audio features.

The most balanced classifier, a Support Vector Machine (SVM) model, was chosen out of four test, but it was unable to guess the genre of any of the machine-generated tracks. It is unclear why this this. It could either be because of the perhaps due to the music generation model's inability to create music that follows patterns of any genre, or it may be because the SVM performs very poorly. In this case, the poor performance of the SVM inhibits deeper insight, but does signal that diffrent machine learning models "learn" the conceptualization of particular music genres and the features that make it up

The prospect of different conceptions of music across models presents important implications for music information systems in the realms of copyright, recommendation systems, and the future of musicology more broadly.

The remainder of this paper is structure as follows. Section 2 details how music is "measured" and the dataset used to train the models in genre classification. Section 3 describes each of the models trained and their performance in training. Section 4 describes how the machine

generated music was generated and how it's featured were extracted, and Section 5 then compares the predicting genre to the prompted genre. Lastly, Section 6 discusses the implications for the results, along with weaknesses of this approach and directions for future work.

2 Data

To investigate the human-likeness of model-based music generation, the nature of audio sampling, measurement and classification is understood. Then, dataset of human-created music from the Free Music Archive (Defferrard et al. 2017) is downloaded, cleaned and transformed. Finally, the data is prepared to train various classifier models, of which the one with the best performance is used to classify the music.

The Python programming language (Python Core Team 2019), particularly the `pandas` module (team 2020) was used to clean and transform the data. The `matplotlib` (Hunter 2007) and `seaborn` (Waskom and team 2023) modules were used to visualize aspects of the dataset during the exploration and model diagnostics phases.

2.1 Measurement and Classification of Sound

Many metrics can be calculated to understand the rhythmic and timbral features of a piece of audio. They can be divided into two main areas: rhythmic metrics, which attempt to quantify aspects of repeated patterns like the beat or rhythm, or timbral metrics, which measure what the audio appears to "sound like" to a human listener. Timbre in music is a complex interplay of auditory attributes that distinguish different types of sound production, even when pitch, loudness, and duration remain constant. That said those aspects

To analyze sound, audio signals are typically sampled digitally at regular intervals, converting continuous acoustic waveforms into discrete values. These discrete samples allow computational methods to analyze sound by capturing amplitude changes over time, enabling the extraction of various metrics that quantify specific auditory characteristics. Sound itself can be quantified through various audio features, broadly grouped into timbral and non-timbral features.

Non-timbral features capture features of rhythm, pitch, and harmony inherent to an audio track. Rhythm in a music is measured by inferring various aspects of a piece through a metric called the onset signal strength. This metric is obtained by calculating a signal that has "high values at the onset of musical events" (Tzanetakis 2011), such as the 'beat drop' of an intense electronic song, or the cadence of a Mozart Symphony. Measuring these events allows for the detection of reoccuring patterns, which can then be inferred into metrics like the tempo or "beat" of a piece of music. Within the realm of pitch and harmony, the most common representation of pitch is through the Pitch Class Profile, otherwise known as Chroma. This metric is composed of vectors that contain the individual occurences of specific musical pitches

in region of the audio being measured. These representations can then be used to accomplish tasks like key and chord detection, though are less useful for genre classificatio, as is the focus of this work.

Timbral features, on the other hand, capture the "textural" qualities of sound. Timbral audio features are typically derived from short-time spectral analysis. They tend to be the most useful for genre classification tasks, as Tzanetakis and Cook (2002) found early on. Key timbral features include Mel-Frequency Cepstral Coefficients (MFCCs), spectral centroid, spectral rolloff, spectral flux, and zero-crossing rate.

MFCCs are psychoacoustically motivated coefficients representing the short-term power spectrum of sound. The computation involves applying the Fourier transform, converting the frequencies to the Mel scale, logarithmically scaling the amplitudes, and applying a discrete cosine transform (DCT) to decorrelate the coefficients. MFCCs provide a compact representation of the timbral qualities and are widely used due to their effectiveness in differentiating between speech, music, and other audio signals. Typically, after transformation a sound has around 40 transformed coefficients, but "in the 'classic' MFCC implementation the lower 13 coefficients are retained" (Tzanetakis 2011), as the higher coefficients tend to add more noise than meaningful insight into the general characteristics of the sound being analyzed.

Spectral centroid measures the "center of gravity" of the sound spectrum, indicating the perceived brightness of a sound. It is calculated by weighting frequencies by their magnitudes and taking the average frequency. Higher spectral centroid values correlate with brighter, higher-frequency sounds, while lower values suggest duller, lower-frequency textures.

Spectral rolloff quantifies the frequency below which a specific percentage (commonly 85%) of the total spectral energy resides. It characterizes the point in the frequency spectrum that separates the lower frequencies containing most energy from the higher frequencies, capturing perceived brightness and spectral distribution.

Spectral flux represents the rate of change in the spectral content between successive frames. It calculates the squared difference between normalized magnitudes of consecutive spectra, thus effectively quantifying how rapidly timbre changes over time. High spectral flux indicates a dynamic timbral evolution, typical of rapidly changing audio signals.

The zero-crossing rate (ZCR) measures the frequency at which the audio waveform crosses the zero amplitude axis. This feature is closely related to the noisiness of the sound, with higher zero-crossing rates generally indicative of noisier signals, such as percussion or certain speech phonemes.

As Tzanetakis and Cook (2002), S.-H. Chen and Chen (2009), and Bhalke, Rajesh, and Bormane (2017) found, timbral metrics allow for sophisticated and parsimonious audio analyses, allowing for efficient classification of genre across various genres of music, so they are made central to the modelling strategy described in Section 3

2.2 Dataset

After understanding timbral features' centrality in genre classification, finding extensive audio feature data to train the classifier was then made of interest. A key requirement of this data is that all music sourced within it must be human generated, and so we use the Free Music Archive (Defferrard et al. 2017). The full archive contains 106,574 tracks across 163 subgenres, and was published in 2017. Vaswani et al. (2017), the paper describing the general Transformer architecture, and Huang et al. (2019), the first paper describing the archictecture of a "music transformer" were published in 2017 and 2018, respectively. Given the close publication time of the dataset and these two publications, it is reasonably unlikely that any artificially-generated music was added to the dataset, since the technology to create sophisticated human-like music did not exist before the dataset was published.

While the FMA is primarily made of audio tracks, which require large amounts of memory to store, Defferard et al. also extracted audio features from each of the tracks using Python's `librosa` module (McFee et al. 2015), and provided associated metadata for each of the songs, particularly a unique ID, title, artist names, and primary genre of the music.

The data cleaning process began by loading raw audio feature data. From this, the key timbral features of each track were extractedd and consolidated systematically into a single, new dataset. The feature groups that were extracted were the first 16 mean values for the Mel-Frequency Cepstral Coefficients (MFCCs), and the means and variances for the spectral centroid, spectral rolloff, and zero crossing rates for each track. Given that Spectral Flux was not one of the coefficients recorded by Defferard et al, and it is omitted from the feature set as a key predictor.

Following feature extraction, metadata associated with the tracks was loaded. The track ID, song title, artist name, and primary genre of the music was extracted, then merged with the features by cross-referencing song ID common to both datasets. Some rows did not have a primary genre, so they were dropped, leading to a dataset, leading to a dataset of 49,598 entries. A sample of the dataset is provided in Table 1.

Table 1: Cleaned FMA Dataset

	Track ID	Title	Artist	Genre	MFCC 1	...	MFCC 16	...
0	2	AWOL - A Way Of Life	AWOL	Hip-Hop	-163.772964	...	0.201078	...
1	3	AWOL - A Way Of Life	AWOL	Hip-Hop	-159.004166	...	-2.270682	...
2	5	AWOL - A Way Of Life	AWOL	Hip-Hop	-205.440491	...	1.947641	...
3	10	Constant Hitmaker	Kurt Vile	Pop	-135.864822	...	0.779950	...
4	134	AWOL - A Way Of Life	AWOL	Hip-Hop	-207.661484	...	-0.493938	...

	...	Spectral Centroid (Mean)	Spectral Centroid (Var)	Spectral Rolloff (Mean)	...
0	...	1639.583252	518069.583913	3267.804688	...
1	...	1763.012451	946258.949226	3514.619629	...
2	...	1292.958130	442649.737553	2773.931885	...
3	...	1360.028687	447160.767490	2603.491943	...
4	...	1257.696289	515674.499991	2462.616943	...

	...	Spectral Rolloff (Var)	Zero Crossing Rate (Mean)	Zero Crossing Rate (Var)
0	...	1.691898e+06	0.085629	0.003776
1	...	2.723681e+06	0.084578	0.004807
2	...	1.751560e+06	0.053114	0.002012
3	...	2.323799e+06	0.077515	0.001665
4	...	1.978261e+06	0.064370	0.002984

Figure 1 illustrates the distribution of genre, which is the outcome variable of interest, in the final dataset.

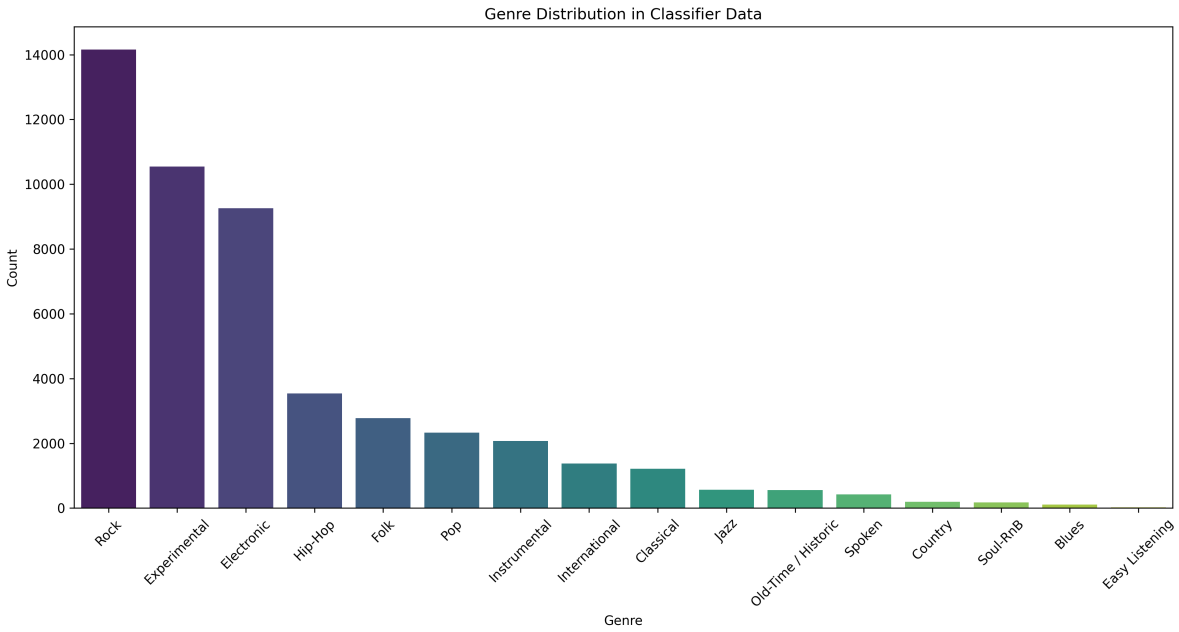

Figure 1: Song Counts per Genre, full cleaned FMA dataset

As Figure 1 shows, the dataset exhibits significant class imbalance. Rock dominates with over 14,000 songs, followed by Experimental and Electronic music with approximately 8,000 and 7,000 songs respectively. The remaining 13 genres have substantially fewer observations, with

some containing less than 1,000 songs. To mitigate this imbalance and simplify analysis, the dataset was filtered to include only the 10 most frequent genres, as detailed in Table 4.

Table 4: Top 10 Song Counts, by Genre

	Genre	Count
0	Rock	14155
1	Experimental	10544
2	Electronic	9260
3	Hip-Hop	3536
4	Folk	2773
5	Pop	2325
6	Instrumental	2070
7	International	1378
8	Classical	1212
9	Jazz	564

By reducing the number of classes to 10, the classification task is simplified and more targeted comparisons and analysis can be made between classifying human and machine-generated music. Though the training data is still unbalanced, as seen in Table 4, class-weighting is utilized in the modelling process to account for the overrepresentation of certain genres.

With respect to the predictor variables of the classifier, while it would be possible to visualize each coefficient of the MFCC spectrum, centroids, and other features, each individual coefficient of a song's MFCCs and spectral features are not particularly useful in gleaning special insight on the dataset as a whole. Thus, visualizing the feature set is of adds little to understanding the general aspects of the auditory structure before modelling it and so the features remain unvisualized.

3 Modelling

The goal of the modelling strategy was to predict a sound's genre based on features in the feature set. The key assumption of the model is that the audio features were extracted from a recording of a piece of music and thus have an inherent genre to them to be classified into.

Python's `scikit-learn` (Pedregosa et al. 2011) and `xgboost` (T. Chen and Guestrin 2016) modules were used to implement several classification models, which were iteratively trained and tested before settling on the one with the best performance. Before training the models, the cleaned dataset was split into 80% training data to 20% testing data. Due to compute limitations and marginal (1-2%) gains in accuracy (Molina 2021), cross-validation was not used in the training process, this particular data split allowed for a considerable amount of

data to be devoted to training the models, while also leaving some human-made music data to test the performance of the classifiers on. This setup allowed for objective diagnostics of model performance before attempting to classify the artificially-generated music, which can be seen in Section A. Wherever possible (all models except KNN), the class-weighting parameter was used to address the imbalanced nature of the genre distribution in the training dataset. This parameter assigns higher penalties to misclassifications of underrepresented classes during model training, helping to mitigate bias toward majority classes. For SVM, Random Forest, and XGBoost classifiers, we applied the 'balanced' weighting option, which automatically adjusts weights inversely proportional to class frequencies. The KNN algorithm does not support class weighting as it makes predictions based solely on distance metrics between data points, rather than through an optimization process.

In order of least to best overall performance, the Support Vector Machine (SVM), K-nearest neighbours, Random Forest, and Extreme Gradient Boosting were evaluated. Due to restrictions in computing power, Deep Learning models were not able to be tested, but were initially considered for this problem. The Support Vector Machine (SVM) model, while the least accurate overall, was the most balanced in prediction across genres. So, it was then used to classify the artificially generated music described in Section 5.

3.1 Support Vector Machine

Support Vector Machines (SVMs) are a widely used supervised learning method for classification tasks, particularly music. SVMs work by finding the best possible boundary, or hyperplane, that separates songs from different genres based on the extracted timbral audio features. When the relationship between genres is too complex for a simple linear boundary, SVMs can apply kernel functions to capture more intricate, non-linear patterns in the data and find higher-dimensional boundaries between the features, effectively classifying them into genres. By focusing mainly on the most informative examples in the dataset, known as support vectors, SVMs achieve strong performance while avoiding overfitting the training data (James et al. 2023). This makes them particularly effective for classifying music, where genre characteristics often overlap and vary in subtle ways.

Both S.-H. Chen and Chen (2009) and Bhalke, Rajesh, and Bormane (2017) demonstrated particular success in genre classification with SVM, each implementing them to predict genre from audio features to a high degree of accuracy. Bhalke et al. demonstrated nearly 96% accuracy on their music classification task, so it was expected that SVMs would exhibit similar good performance using the FMA data. The setup for our SVM training used the default parameters in `scikit-learn`. That meant our SVM used an RBF kernel with $C = 1.0$ for regularization and $gamma =' scale'$ to automatically adjust the kernel coefficient based on the input data. The class-weighting parameter was also enabled in order to account for the unbalancedness of the dataset. Results of testing the SVM on the test set are presented in Table 5

Table 5: SVM Training Set Results

	precision	recall	f1-score	support
Classical	0.485	0.789	0.601	242.000
Electronic	0.619	0.447	0.519	1852.000
Experimental	0.591	0.356	0.445	2109.000
Folk	0.363	0.546	0.436	555.000
Hip-Hop	0.407	0.620	0.491	707.000
Instrumental	0.206	0.384	0.269	414.000
International	0.222	0.453	0.298	276.000
Jazz	0.088	0.442	0.147	113.000
Pop	0.149	0.178	0.163	465.000
Rock	0.797	0.619	0.697	2831.000
accuracy	0.489	0.489	0.489	0.489
macro avg	0.393	0.483	0.406	9564.000
weighted avg	0.573	0.489	0.511	9564.000

Surprisingly, this classifier had an accuracy of about 49%, which prompted the investigation of other classification methods for this task described later on. More surprisingly, were the results of testing the SVM on the test data from the FMA. The SVM learned the patterns of Classical music extremely well, despite it having the 2nd least amount of training examples in the dataset, achieving recall of nearly 80% and an F1 score of 0.6. Upon further reflection, this makes sense, as many subgenres of 'Classical' music have particular blends of instruments whose sounds combine together in predictable ways. An example of this within the context of the dataset is that of string quartets, whose timbre of the 2 Violins, Viola, and Cello come together in predictable ways across recordings, interpretations, and composers, which the classifier then learns.

Jazz and Pop genres showed very poor performance, with F1 scores below 20%, despite enabling class-weighting as a parameter. This likely stems from the diverse range of instruments and arrangements in these genres, requiring more training data for effective pattern recognition. The dataset's imbalance is evident: Jazz has only 564 songs and Pop 2,325, compared to Rock's 14,000. Attempts to improve accuracy by selecting the top 10 most important features did not yield significant improvements. Nevertheless, the SVM maintains relatively balanced classification across genres compared to other models tested.

3.2 K-nearest neighbours

Next, the K-nearest neighbours algorithm (KNN) was tested. Tzanetakis and Cook (2002) had some modest success using the KNN classifier, with genre classification accuracy above 50% based on only timbral features. KNN classifies a new song based on the genres of its K closest

neighbors in the feature space. The number of neighbors, K, is a key parameter that balances flexibility and stability: smaller values of K make the model sensitive to local patterns, while larger values smooth out noise but may overlook finer distinctions. KNN does not build an explicit model but instead relies directly on the training data, making it highly adaptable but potentially sensitive to the choice and scaling of features (James et al. 2023). In music genre classification, KNN is particularly useful when songs of the same genre tend to cluster closely together based on their properties. Given that music of a certain genre tends to "sound" similar in terms of timbre, it would make sense that KNN could work well at classifying genre based on how those features cluster together.

The setup for our KNN classifier was such that based on a $K = 25$, basing classification on the 25 nearest neighbours. In repeated trainings, this was found to balance noise control with precision. The results of testing on the test set are presented in Table 6

Table 6: K-Nearest Neighbours Training Set Results

	precision	recall	f1-score	support
Classical	0.488	0.727	0.584	242.000
Electronic	0.481	0.650	0.553	1852.000
Experimental	0.497	0.493	0.495	2109.000
Folk	0.460	0.366	0.408	555.000
Hip-Hop	0.497	0.307	0.379	707.000
Instrumental	0.424	0.101	0.164	414.000
International	0.701	0.196	0.306	276.000
Jazz	0.667	0.018	0.034	113.000
Pop	0.350	0.015	0.029	465.000
Rock	0.634	0.791	0.704	2831.000
accuracy	0.542	0.542	0.542	0.542
macro avg	0.520	0.366	0.366	9564.000
weighted avg	0.530	0.542	0.509	9564.000

This classifier had an overall accuracy of about 54%, improving it slightly beyond what the SVM could predict overall. Like the SVM, classical music exhibited high performance in the unseen test data, despite low training examples. That said, KNN relies exactly on the training data, there was no class-weighting parameter, so the lesser-represented classes suffered greatly in learning quality. So, from the SVM to KNN, we observed correct classification rates for Pop music go from 16.3% (SVM) to 6.8% (KNN) and Jazz from 14.7% to 3.4%, making this classifier much more biased than the SVM. More represented genres tended to perform even better than in the SVM, with correct Electronic music classification going from 51.9% to 55.3%.

3.3 Random Forest

Tp test if classification could be improved further, tree-based classification methods were then attempted. A Random Forest classifier was attempted first.

A Random Forest classifier builds a large collection of decision trees, each trained on a different subset of the data, and makes predictions by aggregating the results of all the trees, typically through majority voting by each tree. This approach reduces the risk of overfitting that is common with a single decision tree and improves the model's overall accuracy and robustness (James et al. 2023). In the context of music genre classification, Random Forests are particularly valuable because they can capture complex interactions between features such as the ones collected, while maintaining strong predictive performance.

The Random Forest classifier was trained using class-weighting enabled ('balanced') to address the biased data distribution across genre classes. The model utilized scikit-learn's default hyperparameters, including 100 decision trees (n_estimators=100), considering all features at each split (max_features='sqrt'), no maximum depth limitation (max_depth=None), minimum samples required for a node split of 2 (min_samples_split=2), and minimum samples required at a leaf node of 1 (min_samples_leaf=1). The resulting forest contained trees with a mean of 22,023 nodes and an average depth of 32, indicating relatively complex decision boundaries. Bootstrap samples were used during training (bootstrap=True) with Gini impurity as the splitting criterion (criterion='gini'). The results of testing on the test set are presented in Table 7.

Table 7: Random Forest Training Set Results

	precision	recall	f1-score	support
Classical	0.778	0.607	0.682	242.000
Electronic	0.522	0.604	0.560	1852.000
Experimental	0.470	0.619	0.534	2109.000
Folk	0.542	0.373	0.442	555.000
Hip-Hop	0.576	0.339	0.427	707.000
Instrumental	0.588	0.121	0.200	414.000
International	0.866	0.210	0.338	276.000
Jazz	1.000	0.044	0.085	113.000
Pop	0.588	0.022	0.041	465.000
Rock	0.638	0.785	0.704	2831.000
accuracy	0.561	0.561	0.561	0.561
macro avg	0.657	0.372	0.401	9564.000
weighted avg	0.578	0.561	0.531	9564.000

While more accurate overall than both of the other models, at 56.4%, the less-represented genres in the dataset still suffer greatly, even with class-weighting in the random forest ensem-

ble. While Jazz sees a modest increase in correct classification rates, from 3.4 to 8.5 percent, in absolute terms it compares much worse than the SVM at 16.3%. Similarly, this classifier performs the worst on Pop music out of all models tested, with a correct classification rate of 5.6%.

3.4 Extreme Gradient Boosting

Following this, Extreme Gradient Boosted Trees (XGBoost) was explored as a more advanced tree-based method. XGBoost models build an ensemble of decision trees sequentially, where each new tree focuses on correcting the errors made by the previous ones. By using a carefully designed error function and combining it with the use of regularization techniques, XGBoost can achieve both high predictive accuracy and strong control over overfitting. Overfitting is a key concern when using tree-based methods (T. Chen and Guestrin 2016), since each individual tree could just learn the individual features of the training data, without actually learning the overall patterns. In the task of music genre classification, XGBoost can be particularly effective at capturing subtle patterns, meaning it could capture the subtle relationships between audio features more effectively than other models.

Similar to the Random Forest model, the XGBoost classifier was trained with class weighting enabled to address the uneven distribution of genre classes in the dataset. To enhance performance, the model was evaluated using Mean Average Precision (MAP), a metric suited for multiclass classification tasks. Apart from these adjustments, the model relied on XGBoost's default hyperparameters: 100 estimators, a learning rate of 0.3, a maximum tree depth of 6, and a minimum child weight of 1. The model was trained on the same set of features used for the Random Forest. Ultimately, the XGBoost ensemble was composed of 100 decision trees, collaboratively enhancing prediction accuracy. The results of the model's performance on the test set are shown in Table Table 8.

Table 8: Extreme Gradient Boosted Tree Training Set Results

	precision	recall	f1-score	support
Classical	0.714	0.640	0.675	242.000
Electronic	0.536	0.593	0.563	1852.000
Experimental	0.492	0.602	0.542	2109.000
Folk	0.474	0.382	0.423	555.000
Hip-Hop	0.510	0.426	0.464	707.000
Instrumental	0.405	0.159	0.229	414.000
International	0.681	0.225	0.338	276.000
Jazz	0.583	0.062	0.112	113.000
Pop	0.277	0.039	0.068	465.000
Rock	0.659	0.781	0.715	2831.000
accuracy	0.564	0.564	0.564	0.564

Table 8: Extreme Gradient Boosted Tree Training Set Results

	precision	recall	f1-score	support
macro avg	0.533	0.391	0.413	9564.000
weighted avg	0.548	0.564	0.540	9564.000

The XGBoost tree further improved upon the Random Forest in overall accuracy, with a accuracy of 56.4% overall. XGBoost did marginally improve classification rates of the less represented genres of Jazz and Pop. Correct classification of jazz increased from 8.5% to 8.6% while Pop increased from 5.6% to 6.8%. That said, in absolute terms, this model is very biased toward better represented genres, despite it's higher overall accuracy.

From this experimentation process, it was found that the Extreme Gradient-Boosted Model was the most effective at classifying music in the FMA, overall. However, like the Random Forest and KNN classifiers, their classification accuracy tended to be very biased toward data that was more represented in the dataset, even with the class-weighting parameter being enabled. On the other hand, while less predictive overall, the Support Vector Machine was able to be more balanced across different genres, particularly Jazz and Pop where far fewer training examples were provided to the model.

A classification system should have a fair understanding of all genres to be classified, so for the sake of attempting to classify artificially generated music, the Support Vector Machine was used, given the model's balanced learning of all the genres in the dataset.

4 Machine-Generated Music

Next, machine generated music to test the classifieron was sourced. Given limitations in compute power, having only a 4-core CPU, options were limited in terms of music models that could generate music of a decent quality to be tested. MusicGPT (Musat 2024) is a PC-based application that allows smaller models, mainly Meta's MuseGen models (Shi et al. 2023), to be run and generate small clips of music in resource constrained environments. For this setup, the default 'small' model was selected for the experiment.

Nine prompts were systematically devised to prompt the model to generate music that would have at least some of the general characteristics of the genre described to it. Notably, no track falling under an 'international' genre was generated as the category is too broad and the model could not effectively generate an intelligible piece. The prompts were written out as follows:

- Classical Music: "A quintessential classical music song"
- Electronic Music: "A quintessential electronic music song"
- Experimental Music: "A quintessential experimental music song"
- Folk Music: "A quintessential folk music song"

- Hip Hop Music: "A quintessential hip hop music song"
- Instrumental Music: "An instrumental music song"
- Jazz Music: "A quintessential jazz music song"
- Pop Music: "A quintessential pop music song"
- Rock Music: "A quintessential rock music music song"

These prompts were then fed one by one into the MusicGPT interface, which passed the prompt into the MuseGen model. Each prompt was then set to generate a clip lasting 30 seconds, which mimics the feature extraction setup of Defferrard et al. (2017). A new chat was created for each genre, to avoid the model confusing the context from the previous prompt with the prompt of the current song being generated and a total of nine .wav files were saved. Notably, despite the classifiers being trained to classify into an 'International' genre, no such song was able to be generated with MusicGPT, since it is too broad a category, genre-wise.

Next, like Defferrard et al. (2017), the `librosa` Python module was then used to extract the timbral features of each of the songs: MFCCs 1-16, and the means and variances of the spectral centroid, rolloff, and zero-crossing rate. The features of each machine-generated track were then assembled into a dataset identical to that of the cleaned FMA dataset, except for the columns of "track_id" and "title", given that these songs were machine generated and there were only 10 generated. Finally, each song's features was then passed into the SVM classifier and a predicted genre was obtained.

5 Results

The following genres were predicted by the SVM model and subsequently compared against their actual genres.

Table 9: Results of SVM Classification of Machine-Generated MusicGPT Songs

	MusicGPT Genre	Predicted
0	classical	International
1	electronic	Experimental
2	experimental	Classical
3	folk	International
4	hiphop	Pop
5	instrumental	Pop
6	jazz	Hip-Hop
7	pop	Hip-Hop
8	rock	Rock

As Table 9 shows, the classifier could not classify any of the machine-generated songs into their respective genres.

The classification poor results reveal several key insights about both the classifier's performance and the characteristics of the generated music. The SVM classifier misclassified all generated songs, with a striking tendency to categorize multiple tracks as "International". This complete misclassification can be attributed to multiple factors: the classifier's inherent limitations (demonstrated by its modest 48.9% accuracy on human music test data), the "International" genre potentially serving as a catch-all category for songs with ambiguous features due to its broad stylistic range in the training data, and the generated music possibly lacking the genre-specific timbral patterns that the classifier was trained to recognize.

A particularly intriguing observation emerged in the symmetric confusion between Pop and Hip-Hop genres, where the prompted Pop song was classified as Hip-Hop and vice versa. This symmetric misclassification suggests that the small MuseGen model may encode these genres' characteristics differently than they manifest in human-made music, or alternatively, that these genres share similar timbral features that the classifier struggles to distinguish between.

The complete failure to correctly classify any generated songs likely stems from the limitations of using MuseGen's "small" model variant. This model, because it was designed for resource-constrained environments, may lack the capacity to capture and reproduce finer genre-specific timbral patterns effectively. This limitation is apparent even to human listeners, as the generated samples often lack clear genre-specific characteristics despite being prompted to be "quintessential" examples.

From these results, there are two possible explanations: either the classifier's poor performance (48.9% accuracy) makes it unreliable, or the generated music fundamentally differs from human-made music in its timbral features.

This setup demonstrates a workflow for assessing AI music generation, but the SVM's limitations make it difficult to determine if misclassification stems from the classifier or fundamental differences in the generated music. Future work should use higher-performing classifiers, test larger music generation models, and incorporate human validation.

Despite limitations, this experiment provides a systematic approach for measuring how well a music generation model learns genre-specific characteristics. The methodology could be improved through better, more balanced datasets, classifiers with stronger performance (above 80% at least), larger models, or more sophisticated prompting.

6 Discussion

6.1 Human-Like vs. Machine-Generated Music: Are They Really Comparable?

As observed in the experiment, none of the songs generated were able to be classified by the classifier into their correct genres, despite being classified on the exact same timbral features as songs that were extracted from the Free Music Archive. The support vector classifier cannot discern any pattern within the music that was generated. This indicates that the small

MuseGen model does not appear to embed genre-specific features into its knowledge base in the same way that human music does. Otherwise, it would have been able to generate music that would have been correctly classified.

Given that only the "small" model variant of Meta MuseGen was used to generate the investigated music, there is clear evidence that the small model, at least, generated "surface-level" music patterns that miss the deeper structural patterns in human-made music that the classifier was trained on. Given that the small model was designed particularly to operate in resource-constrained environments, it likely lacked the ability to learn rich style representations of each individual genre. Even if it was prompted to capture the general idea of each genre, the breadth of music that it was expected to produce was likely too much to expect for the model.

At least at the size of a small music generation model, it is clear to see that the music it is able to generate is incomparable to what is generated by human effort. Given the advent of sophisticated music generation tools such as Suno ("Suno: AI-Powered Music Generation Platform" 2025), it does not appear that this trend continues as Music generation models scale larger.

6.2 The Failure of the SVM Classifier

Interestingly, the SVM classifier completely failed at the classification task it was presented with; not a single song was classified correctly. The total misclassification of AI-generated music suggests that it is measurably different from human-made music, not just on a perceptual level as described by the extracted timbral features, but also in how those features present themselves statistically.

While this could be attributed to the fact that it was the worst-performing classifier overall, even a mediocre classifier should have identified some genres correctly, especially genres such as Rock, which constituted a large majority of the training examples. Given that a classifier quantitatively learns the various timbral patterns inherent in a piece of music through statistical means, it can offer a more objective way of measuring "authenticity" to a particular genre by learning the patterns inherent in the vast majority of songs within that genre—something a human would be incapable of or uninterested in doing.

Automatic classifiers have shown similar performance to humans in music classification tasks. An investigation by Dong (2018) showed that a Convolutional Neural Network model was able to exactly match human performance in music classification, which was around 70% accuracy. Future work with model designs such as Transformers could result in classifiers being better at humans at classifying music, leading to them being more objective measures of a music's fidelity to a particular genre.

Although the SVM may have failed in this iteration of the setup, future work to improve a classifier's accuracy would allow for a more objective and efficient assessment of fidelity to a genre compared to human evaluation.

6.3 Impact of Model Size and Prompting on Music Quality

Given this investigation, it is clear that the Small MuseGen model was not able to encode all of the patterns of each genre tested strongly enough into it's learning. In the field of LLMs, work by Kaplan et al. (2020) have shown a power-law relationship between the number of parameters a Transformer-based Large Language Model has and the perceived intelligence of responses. Given the Transformer based nature of some music generation models like Huang et al. (2019), perhaps a similar law exists, wherin a larger model size yields music whose prompted genre has higher fidelity to the genre described.

In a similar vein, while the each of the prompts were made to be simple and systematic for straightforward testing, perhaps more sophisticated prompting methods that described various aspects of the genre of music to be generated could have yielded results with higher genre fidelity. This could have included things like instruments to be used and comparisons to major artists, grounding the model to generate music that is most like the genre that was desired during prompting.

6.4 Broader Implications for Music Information Retrieval (MIR)

This experiment showed the potential for automatic genre classification systems to fail on AI-generated content. Given this, the results present some important implications for Music Information Systems more broadly, such as on services like Spotify or Apple Music.

Copyright detection systems on services such as YouTube and Spotify currently use machine learning methods to discern sounds that may appear like copyrighted material or not. If artificially intelligent music generation software can create music that can foos these copyright systems, the legal system of copyright and the protection it affords artists is put at risk. If copyright laws can be subverted through widely available tools, such as improved versions of MuseGen or similar software, then artists lose the exclusive right to sell or license their material, creating issues of ownership and market structuring. Who "owns" the patterns embedded in a music generation model: the owner of the music used for training, or the maker of the model?

Machine learning methods are also used to recommend music to users within a particular genre or genres. But what makes one song be recommended more than another? While some of it has to do with popularity through the number of clicks, the clicks are ultimately driven through human enjoyment of the particular features of a piece of music. Given that features and measurements of popularity can be mined from songs on services like Spotify, it is perfectly reasonable that a savvy cultural analyst could attempt to correlate certain features with high

popularity. If these results are then fed into a music generation model, it is conceivable that models could learn to optimize to "please" the recommendation algorithms, creating music that is more popular than any individual human or team of humans could write.

The previous two observations have more general implications for musicology in general. If music can be "optimized" for, like a financial portfolio or human-like language generation, when does it cease to become music? As tools emerge that can generate music nearly indistinguishable from that created by humans - and in a fraction of the time - we are prompted to reconsider the question: what is music? What are aesthetics and beauty in a world largely shaped by algorithms and optimization? Is there a limit to how much machines can mimic human creation before it ceases to be human altogether? These are some of the important questions about auditory aesthetics that need to be considered, experimented on, and addressed by humans, as algorithms continue to take over more and more facets of our lives, individual decision making, and agency.

6.5 Weaknesses and next steps

This investigation has several significant weaknesses that should be considered when interpreting the results, suggesting that this work primarily serves as scaffolding for future research in this domain.

The small MuseGen model employed in this study demonstrated limited capability in capturing genre-specific patterns. Trained on a restricted subset of music data to accommodate local computational constraints, the model produced outputs with poor stylistic fidelity to the intended genres. Consequently, these generated pieces lacked the distinctive characteristics necessary for accurate genre classification.

The Support Vector Machine (SVM) classifier exhibited inadequate performance (49% accuracy), likely attributable to the imbalanced training dataset. This poor classification rate introduces ambiguity regarding whether misclassifications stemmed from the MuseGen model's failure to generate stylistically faithful music or from the classifier's inherent limitations in distinguishing musical genres.

A methodological limitation was the exclusive reliance on timbral features for classification. While Tzanetakis and Cook (2002) demonstrated the efficacy of timbral features in genre classification, their research also highlighted that incorporating rhythmic and harmonic features significantly improved classification performance. The absence of these additional feature types in our approach, along with the omission of important timbral aspects like Spectral Flux (due to limitations in the FMA dataset), likely constrained classification accuracy.

The absence of human validation represents another critical shortcoming in our experimental design. Without human listeners evaluating the genre categorization of the generated pieces, we lacked a baseline against which to measure both the classifier's performance and

the stylistic authenticity of the AI-generated music. Such validation would have strengthened the experimental foundation and added weight to our findings.

Finally, the limited scope of generating only one sample per genre, combined with the aforementioned classifier limitations, undermines the statistical power of our results. This approach precludes generalization to the broader set of potential outputs from the MuseGen model, restricting the validity and applicability of our conclusions regarding machine-generated music.

Building upon the limitations identified, several concrete directions for future research emerge that could substantially enhance our understanding of machine-generated music classification and evaluation.

Future studies should employ larger, more sophisticated music generation models. Specifically, experiments could compare outputs from the full-scale MuseGen model against those from alternative frameworks such as Suno, MusicLM, and MusicGen. A comparative analysis of these models' outputs—using the same genre prompts—would provide insights into which architectures produce the most stylistically faithful music across different genres. This approach would help identify whether certain models excel at specific genres while struggling with others.

A more robust experimental design should incorporate multiple generated samples per genre (e.g., 25-50 samples) to enable statistical analysis of classification performance. Each sample could be analyzed individually and collectively to determine if certain genres consistently prove more challenging for both generation and classification. This expanded dataset would allow for meaningful statistical testing and more defensible conclusions about model performance.

Future work should explore alternative classification approaches beyond traditional machine learning algorithms. Specifically, implementing Convolutional Neural Networks (CNNs) trained directly on audio spectrograms could capitalize on spatial patterns within the frequency domain, which were shown by Dong (2018) to have at least a human-level (70%) accuracy, if not higher with current developments in model tuning. Additionally, transformer-based classifiers could potentially capture longer-term dependencies in the musical structure. A direct comparison between these approaches and traditional methods (SVM, Random Forest, KNN) using standardized evaluation metrics would quantify the performance improvements offered by more advanced techniques.

The feature extraction process should be expanded to include rhythmic and harmonic features alongside timbral characteristics. Concrete implementations could include beat histogram analysis for rhythmic features and chord progression detection for harmonic content. Following the methodology Tzanetakis and Cook (2002) more comprehensively by implementing their full feature set could likely improve classification performance and provide a more nuanced understanding of genre distinctions in AI-generated music.

A critical enhancement would be through incorporating human evaluation through structured listening tests. A formal experiment could involve music experts rating generated samples on genre adherence using Likert scales and providing qualitative feedback. This human-centered

evaluation could serve as a gold standard against which to measure classifier performance. Additionally, A/B testing that asks participants to distinguish between human-composed and AI-generated music within specific genres would provide insights into which genres machines replicate most convincingly.

Finally, future research could explore embedding similarity between classifier models and music generation models. This novel approach would involve extracting and comparing the latent representations from both model types when processing the same musical inputs. If publicly available, comparing the weight distributions across model layers could reveal whether certain musical features are encoded similarly in both classification and generation systems. This analysis might uncover fundamental commonalities in how machines represent musical concepts, regardless of whether the task is generative or discriminative.

Collectively, these proposed directions address the limitations of the current study while advancing our understanding of how machines conceptualize and reproduce musical genres. With the rapid proliferation of AI-generated music through platforms like Suno, developing reliable methods to analyze and evaluate such content becomes increasingly important for both academic research and practical applications in the music industry.

7 References

Bhalke, Daulappa Guranna, Betsy Rajesh, and Dattatraya Shankar Bormane. 2017. "Automatic Genre Classification Using Fractional Fourier Transform Based Mel Frequency Cepstral Coefficient and Timbral Features." *Archives of Acoustics* 42 (2): 213–22. https://doi.org/10.1515/aoa-2017-0024.

Chen, Shi-Huang, and Shih-Hao Chen. 2009. "Content-Based Music Genre Classification Using Timbral Feature Vectors and Support Vector Machine." In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, 1095–1101. ICIS '09. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1655925.1656124.

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." https://xgboost.readthedocs.io/en/latest/.

Defferrard, Michaël, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. 2017. "FMA: A Dataset for Music Analysis." In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*. https://arxiv.org/abs/1612.01840.

Dong, Mingwen. 2018. "Convolutional Neural Network Achieves Human-Level Accuracy in Music Genre Classification." *arXiv Preprint arXiv:1802.09697*. https://arxiv.org/abs/1802.09697.

Huang, Cheng-Zhi Anna, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew Dai, Matt Hoffman, Monica Dinculescu, and Douglas Eck. 2019. "Music Transformer: Generating Music with Long-Term Structure." In *Proceedings of the 2019 International Society for Music Information Retrieval Conference (ISMIR)*. https://arxiv.org/abs/1809.04281.

Hunter, J. D. 2007. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering* 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.

James, Gareth, Daniela Witten, Trevor Hastie, Rob Tibshirani, and Jonathan Taylor. 2023. *An Introduction to Statistical Learning with Applications in Python*. Springer. https://www.statlearning.com/.

Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. "Scaling Laws for Neural Language Models." *arXiv Preprint arXiv:2001.08361*. https://arxiv.org/abs/2001.08361.

McFee, Brian, Colin Raffel, Dawen Liang, Daniel P. W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. "Librosa: Audio and Music Signal Analysis in Python." Edited by Katharina Huff and James Bergstra. SciPy. https://doi.org/10.25080/Majora-7b98e3ed-003.

Molina, Eden. 2021. "A Practical Guide to Implementing a Random Forest Classifier in Python." 2021. https://towardsdatascience.com/a-practical-guide-to-implementing-a-random-forest-classifier-in-python-979988d8a263.

Musat, Gabriel. 2024. "MusicGPT: Generate Music Based on Natural Language Prompts Using LLMs Running Locally." *GitHub Repository*. https://github.com/gabotechs/

MusicGPT; GitHub.

Pedregosa, Fabian, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research*. http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

Python Core Team. 2019. "Python: A Dynamic, Open Source Programming Language." Python Software Foundation. https://www.python.org/.

Shi, Yu, Wenjie Hu, Haotian Zhang, and Shuang Yang. 2023. "MuseGen: A Generative Model for Music." https://research.facebook.com/publications/musegen-a-generative-model-for-music/.

"Suno: AI-Powered Music Generation Platform." 2025. https://suno.ai/.

team, The pandas development. 2020. "Pandas-Dev/Pandas: Pandas." Zenodo. https://doi.org/10.5281/zenodo.3509134.

Tzanetakis, George. 2011. "Audio Feature Extraction, Chapter 2: Audio Feature Extraction." In *Music Data Mining*, edited by Tao Li, Mitsunori Ogihara, and George Tzanetakis, 23–51. CRC Press. https://www.taylorfrancis.com/chapters/edit/10.1201/b11041-8/audio-feature-extraction-george-tzanetakis.

Tzanetakis, George, and Perry Cook. 2002. "Musical Genre Classification of Audio Signals." *IEEE Transactions on Speech and Audio Processing* 10 (5): 293–302. https://doi.org/10.1109/TSA.2002.800560.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." *arXiv Preprint arXiv:1706.03762*. https://doi.org/10.48550/arXiv.1706.03762.

Waskom, Michael, and the seaborn development team. 2023. "Seaborn: Statistical Data Visualization." https://seaborn.pydata.org/.

Appendix

A Full Model Diagnostics

A.1 Support Vector Machine

Table 10: SVM Training Set Results

	precision	recall	f1-score	support
Classical	0.485	0.789	0.601	242.000
Electronic	0.619	0.447	0.519	1852.000
Experimental	0.591	0.356	0.445	2109.000
Folk	0.363	0.546	0.436	555.000
Hip-Hop	0.407	0.620	0.491	707.000
Instrumental	0.206	0.384	0.269	414.000
International	0.222	0.453	0.298	276.000
Jazz	0.088	0.442	0.147	113.000
Pop	0.149	0.178	0.163	465.000
Rock	0.797	0.619	0.697	2831.000
accuracy	0.489	0.489	0.489	0.489
macro avg	0.393	0.483	0.406	9564.000
weighted avg	0.573	0.489	0.511	9564.000

A.2 K-Nearest Neighbours

Table 12: K-Nearest Neighbours Training Set Results

	precision	recall	f1-score	support
Classical	0.488	0.727	0.584	242.000
Electronic	0.481	0.650	0.553	1852.000
Experimental	0.497	0.493	0.495	2109.000
Folk	0.460	0.366	0.408	555.000
Hip-Hop	0.497	0.307	0.379	707.000
Instrumental	0.424	0.101	0.164	414.000
International	0.701	0.196	0.306	276.000
Jazz	0.667	0.018	0.034	113.000
Pop	0.350	0.015	0.029	465.000
Rock	0.634	0.791	0.704	2831.000

Table 12: K-Nearest Neighbours Training Set Results

	precision	recall	f1-score	support
accuracy	0.542	0.542	0.542	0.542
macro avg	0.520	0.366	0.366	9564.000
weighted avg	0.530	0.542	0.509	9564.000

A.3 Random Forest Classifier

Table 14: Random Forest Training Set Results

	precision	recall	f1-score	support
Classical	0.778	0.607	0.682	242.000
Electronic	0.522	0.604	0.560	1852.000
Experimental	0.470	0.619	0.534	2109.000
Folk	0.542	0.373	0.442	555.000
Hip-Hop	0.576	0.339	0.427	707.000
Instrumental	0.588	0.121	0.200	414.000
International	0.866	0.210	0.338	276.000
Jazz	1.000	0.044	0.085	113.000
Pop	0.588	0.022	0.041	465.000
Rock	0.638	0.785	0.704	2831.000
accuracy	0.561	0.561	0.561	0.561
macro avg	0.657	0.372	0.401	9564.000
weighted avg	0.578	0.561	0.531	9564.000

A.4 Extreme Gradient Boosting Classifier

Table 16: Extreme Gradient Boosted Tree Training Set Results

	precision	recall	f1-score	support
Classical	0.714	0.640	0.675	242.000
Electronic	0.536	0.593	0.563	1852.000
Experimental	0.492	0.602	0.542	2109.000
Folk	0.474	0.382	0.423	555.000
Hip-Hop	0.510	0.426	0.464	707.000
Instrumental	0.405	0.159	0.229	414.000

Table 16: Extreme Gradient Boosted Tree Training Set Results

	precision	recall	f1-score	support
International	0.681	0.225	0.338	276.000
Jazz	0.583	0.062	0.112	113.000
Pop	0.277	0.039	0.068	465.000
Rock	0.659	0.781	0.715	2831.000
accuracy	0.564	0.564	0.564	0.564
macro avg	0.533	0.391	0.413	9564.000
weighted avg	0.548	0.564	0.540	9564.000

Table 11: SVM Confusion Matrix

(a)

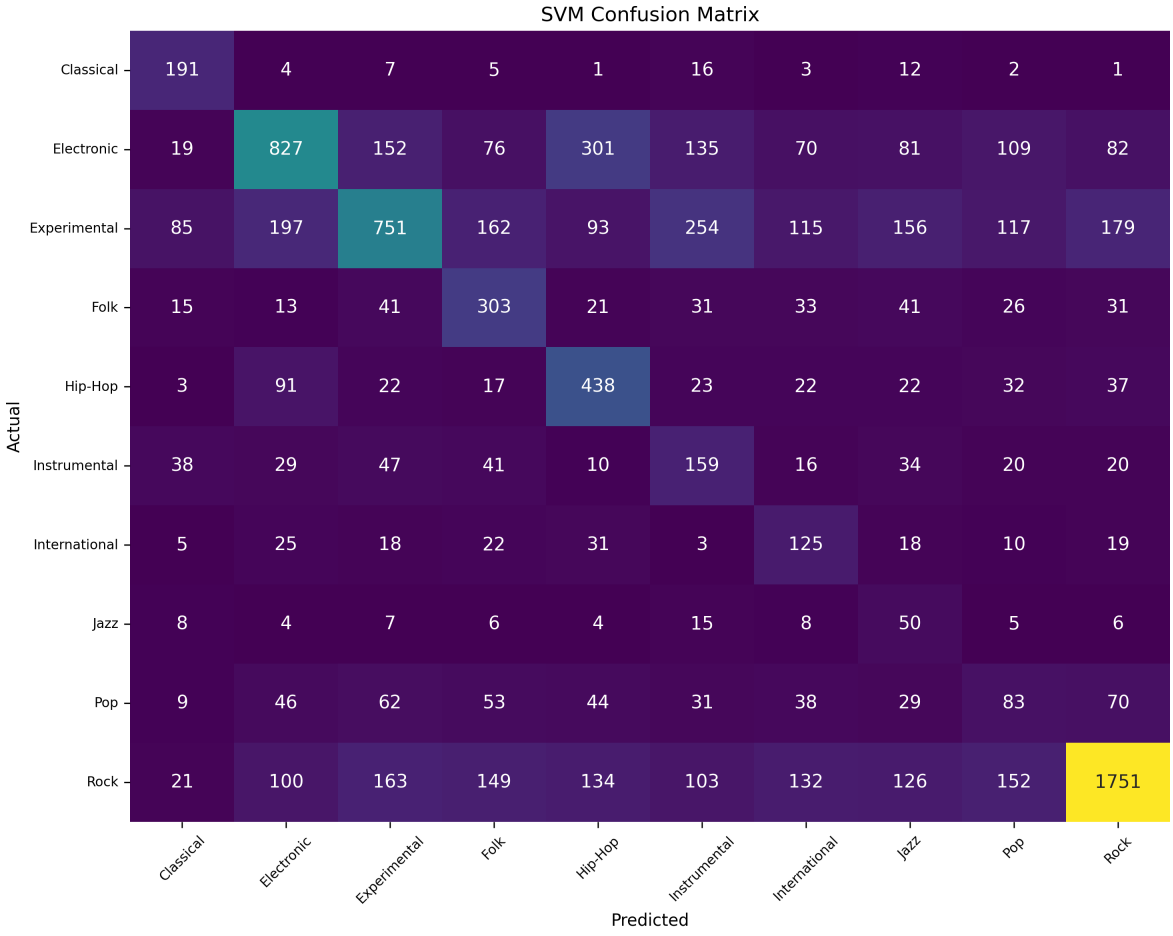

Table 13: K-Nearest Neighbours Confusion Matrix

(a)

Table 15: Random Forest Confusion Matrix

(a)

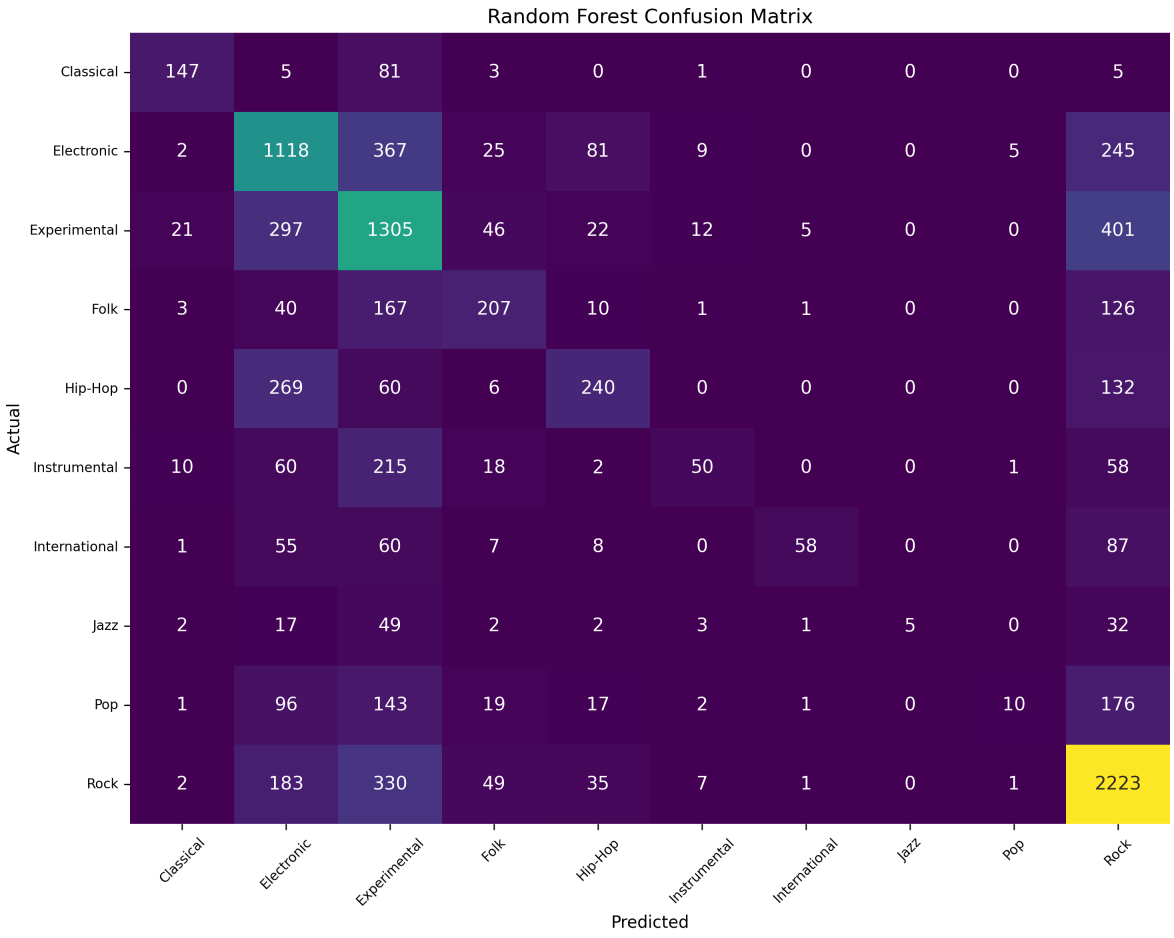

Table 17: Extreme Gradient Boosting Confusion Matrix

(a)