Machine Learning for Music Genre Classification

Thesis · September 2019

1 author:

Bryn Lansdown
University of Birmingham
3 PUBLICATIONS **0** CITATIONS

SEE PROFILE

UNIVERSITY OF BIRMINGHAM

MSC COMPUTER SCIENCE

Machine Learning for Music Genre Classification

Author:
Bryn Lansdown

Supervisor:
Dr. Shan He

School of Computer Science

September 2019

Abstract

This study compares machine learning algorithms in their ability to automatically classify song excerpts into their musical genres. First, a review of existing techniques and approaches is carried out, both in terms of feature engineering and algorithms on offer. Two unique datasets are created from an existing online music archive, using undersampling and oversampling to balance the classes. Fifty-four features are manually extracted from each sample, using audio analysis libraries LibROSA and Aubio. Two sets of analyses are then performed; the first comparing, for each feature, the average values for each genre; the second comparing each feature by f-value (the scores having been calculated using Scikit-learn's SelectKBest function). Four classifiers are created - a neural network, a support-vector machine, a random forest and a gradient boosting machine - and training and testing are performed on each, with each of the two unique datasets created. Results indicate that the support-vector machine is the algorithm most suited to the task, with Hip-Hop music the most accurately classified genre on average. Finally, an evaluation of the whole process is conducted, including an assessment of the choice of dataset, choice of features, and various aspects of the project management.

Acknowledgements

I wish to thank Dr. Shan He for supervising this project, and for his excellent feedback and advice. I am also grateful to Tom Goodman for volunteering his time to help guide the project. Finally, I would like to thank my family and friends, for their continued encouragement and support.

Contents

Chapter 1

Introduction

The aim of this project was to compare machine learning algorithms in their ability to automatically classify song excerpts into the correct musical genre. For humans who are familiar with the genres in question, music genre recognition (MGR) is not an especially difficult task, as exemplified in Gjerdingen (2008). Most people well-acquainted with Western popular music are able to identify that a given Metallica song is an example of the heavy metal genre, and that a given Status Quo song is an example of rock music, just by listening to the audio. Importantly, they are usually able to perform this type of broad genre classification even if they are unfamiliar with the song or artist in question: 'heard' qualities of the sound are normally enough. As Gjerdingen (ibid.) describes, only a short sample of audio is usually needed, sometimes less than a second. A more fine-grained classification into musical subgenre usually requires some level of expertise on the part of the listener, and the greater the level of expertise, the more accurate the classification that can be made. The question is raised as to whether software can be written to perform genre classifications as well as humans, and what type of approaches work best.

Despite being an interesting endeavour in and of itself, there are practical, real-world applications of automated music genre recognition, with music streaming services representing the most pertinent example. Pandora is a US-based streaming and recommendation platform that employs musicologists to annotate songs with genre and rhythm features, with the aim of creating better recommendation playlists (Lawton, 2017, 'Deconstructing the recommendation engine'). Users may want to discover tracks that are similar to those that they are already a fan of, and it makes sense for providers to have such metadata for their songs available, in order to facilitate this kind of discovery. Given the vast and ever-growing quantity of music available on the internet, and the need for services to manage increasingly large song databases, manual analysis and annotation of individual songs does not seem like a sustainable long-term solution. From an engineering perspective, despite the cost of developing and maintaining the system, long-term expenditures could be significantly reduced with the use of accurate, automated genre-recognition systems. Large streaming services are surely already aware of the prospects of automated genre recognition techniques, but questions remain as to the best way a music genre recognition system can be architected.

Various machine learning techniques have been applied to problems - including genre recognition - in the field of Music Information Retrieval. A range of possible approaches toward the feature engineering of such projects exist. In general, machine learning classifiers will perform well when they are able to 'understand' what makes a data sample a member of a particular class, and, in many cases, when classes tend to be easily 'separable' from one another (i.e. with different classes varying significantly in terms of their average feature values). When humans are able to recognize the genre of a given audio track, they typically do so based on 'high-level'

aspects of the sound in question, e.g., the instruments in use, the feel with which they are played, the 'overall sound' of the song, etc. These types of emergent musical property may be difficult for software to recognize; however, given the direct availability of audio files, it is possible for software to detect aspects of audio files that humans cannot perceive through mere listening alone. Automatic genre classification based on these types of features has already been performed, with promising results. The general motivation behind such approaches is that different musical genres tend to use different types of sounds, perceptible by humans and potentially detectable by the right software as well.

This study compares several machine learning algorithms in their ability to classify songs accurately. First, a survey of existing literature is carried out, looking at how machine learning has been applied to problems of music classification and audio classification in general. Tools for extracting features from audio files are discussed, as well as techniques for classifying music based on other modalities such as song lyrics and album artwork. This is followed by a discussion of machine learning techniques fit for present purposes. Various candidate datasets are evaluated, with a justification of the final dataset chosen and an explanation of how it was preprocessed and reduced in size. An explanation of the features extracted, and an analysis of the features chosen (in terms of usefulness), are then presented. After this, a description of the machine learning models follows, as well as a presentation of the results of each classifier. Finally, an interpretation of the results is carried out, in addition to an evaluation of design choices and an assessment of the project management. The study is concluded with the main lessons learned, and recommendations for future research.

Chapter 2

Related Work

Automatic genre classification of music can be attempted in a variety of ways. A typical approach involves processing a dataset of audio files, extracting features from them, and then using a dataset of these extracted features to train a machine learning classifier. This is the strategy that the majority of this literature review will focus on, as it encompasses the techniques that have proven to be most effective when classifying via one modality; as such, it is the approach that has been studied most extensively. However, there exist other modalities to make use of, including song lyrics, album reviews, and album artwork. Each genre of music tends to produce content with distinct approaches to *all* aspects of the finished product, and, as a result, genre classification via analysis of related text and imagery has also proven fairly successful. 'Multimodal' classification has shown some promise, though more research is needed.

After exploring these different approaches, surveying the existing literature in this area, and justifying the choice of modality that this particular study focuses on, this section will look more carefully at the different A.I. techniques that classification systems (including those in the field of music genre recognition) make use of. The majority of approaches involve some kind of machine learning classifier, such as an artificial neural network or a support-vector machine, although different strategies are sometimes used in text classification. This section will also justify the types of classifier chosen for this particular project.

2.1 Modalities

2.1.1 Audio

Machine learning typically requires each data example to be represented numerically, often as a vector of relevant information or 'feature vector'. To classify based on audio alone, audio files are often processed such that the relevant characteristics are extracted and stored in vector format. The rest of this section details the types of characteristics that can be derived from audio files for use in machine learning, and examples of studies that have utilized these techniques for classification purposes.

Audio Features

Panagakis et al. (2008, Introduction) state that there are three main types of audio feature employed for music classification: timbral texture features, rhythmic features, and pitch content features. Detailing every possible candidate feature is outside the scope of this project; this section will instead outline a selection, which are among the most commonly used for classification (and all of which were used in this project):

1. MFCCs (timbral texture feature)
2. Zero Crossing Rate (timbral texture feature)
3. Spectral Spread (timbral texture feature)
4. Chroma Features (pitch content feature)
5. Average Tempo (rhythmic feature)

Mel-Frequency Cepstral Coefficients (MFCCs) (timbral texture feature)

MFCCs are one of the most common feature types used in audio classification of all kinds, having been used in speech and ambient noise recognition as well as music genre classification. Unlike the estimated tempo of a given audio signal, which consists of a single value, ordinarily between 13 and 30 MFCCs are extracted, with the exact number of coefficients a decision made by the researcher. In this project, of the 54 features extracted from the audio file dataset, 30 were MFCCs; thus, it is worth examining the feature type in some detail.

MFCCs can be better understood by understanding the process by which they are derived from an audio file, which ordinarily runs as follows (Lyons, 2012):

1. An audio file is divided into small frames (a process known as 'windowing'), each lasting 20-40ms. The resulting vector is a time series that represents the overall audio file.
2. For each frame, an estimation of the power spectrum is calculated, by use of a Fourier Transform. The power spectrum of an audio frame is roughly equivalent to the power present for each of the frequency bands within that frame.
3. A 'Mel-spaced Filterbank' is used on each of the output sets from step 2. The Filterbank is spaced according to the mel scale, to give emphasis to the frequency bands that are most relevant in human perception (humans find it easier to perceive differences in the lower registers, i.e., the bassier frequencies, so the filters are more narrowly spaced in these lower registers than in the higher registers). Each of the filters corresponds to a particular frequency band, and removes all of the values that fall outside this range. The result is a series of 'filterbank energies'; one value per filter.
4. After computing the filterbank energies, their logarithms are calculated.
5. A Discrete Cosine Transform is performed on the logarithms, and approximately half of the resulting values are dropped. The result is a time-series vector of MFCCs: one set of MFCCs for each frame of the audio file.

It is further possible to find the mean values for each MFCC of an audio file. For example, to find a signal's mean MFCC5 value, the values of MFCC5 across all windows of the signal would be added up, and this value would then be divided by the number of windows in the signal. This process can be applied for each MFCC to find the mean MFCC values.

MFCCs are used in music genre classification to detect timbre (Jensen et al., 2006), which can be defined as the 'quality' or 'colour' of a sound. Different genres of music use different sets of instruments, resulting in striking timbral differences - compare the soft sound of Chopin's piano compositions with the harshness of punk

rock. Even when two genres use the same set of instruments, they are often used to totally sonic different effect; e.g. Electronic Dance Music often features a bass-heavy and punchy kick drum, in comparison to jazz music, which typically uses the kick drum in a much more subdued way. The timbral qualities of a song are detectable in its spectral information, and use of MFCCs has proven to be a powerful way of representing this content for use in machine learning.

Zero Crossing Rate (timbral texture feature)

A second feature commonly used in music genre recognition is the zero-crossing rate (ZCR). The ZCR of a signal is defined as the rate at which a signal changes from positive to negative or negative to positive (Peeters, 2004, section 4.2). In the context of audio, this refers to the number of times the amplitude of the signal passes through a value of zero, within a given time interval. It is commonly used as a measure of the *smoothness* of an audio signal (Bäckström, 2019), and has been used as a feature in various forms of audio classification. For example, it has been used to separate speech recordings into periods of 'voiced' speech (sounds made when vowels are spoken) and 'unvoiced' speech (sounds made when consonants are spoken) (Bachu et al., 2008). In the context of music information retrieval, ZCR has been used to identify the presence of percussive sounds (Gouyon, 2000); a factor relevant to genre recognition because percussive sounds are utilized by some genres more than others. Like MFCCs, ZCR is another feature that can be used to understand the timbre of an audio file.

Spectral Spread (timbral texture feature)

'Spectral spread' is understood in the context of 'spectral centroid', which refers to the 'centre of gravity' of the spectrum of a signal (Giannakopoulos & Pikrakis., 2014, 4.4.1 'Spectral Centroid and Spread'). Perceptually, spectral centroid relates to the *brightness* of a signal's sound. The spectral spread of a signal describes the 'average deviation of the rate map around its centroid' (Two!Ears team, 2015, 'Spectral features'); i.e., the variance. Noisier sounds imply a higher spectral spread, whereas signals with a spectrum 'tightly concentrated' around the centroid will have a lower spectral spread (Giannakopoulos & Pikrakis, 2014, 4.4.1 'Spectral Centroid and Spread'). Like MFCCs and ZCR, spectral spread is used to understand the timbre of an audio file, and is another feature that has been used in music genre recognition. For example, Giannakopoulos et al found that spectrograms of electronic music excerpts are typically spread more widely around their centroid than are excerpts of jazz (ibid.).

Chroma Features (pitch content feature)

Chroma-based features concern the musical notes that are played throughout a song. Various techniques for detecting musical pitch exist, with newer approaches focusing on pitch detection of polyphonic sequences (Goodman and Batten, 2018). Audio analysis software packages are able to estimate the intensity with which specific notes are present within an audio file, as well as how these changes take place over time. The chromagram in figure 2.1 (Meinard.mueller, 2016) represents a C-major scale played over the course of several seconds.

Information on the notes played most frequently throughout a song may be useful in classification for a number of reasons - for example, because different genres

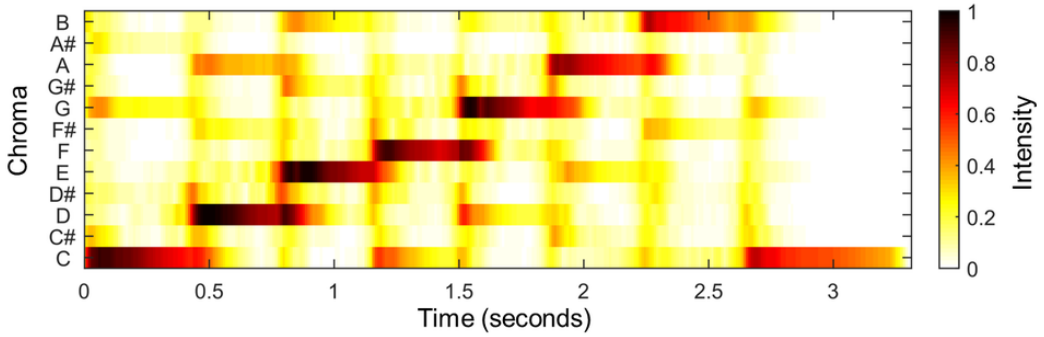

FIGURE 2.1: Chromagram

of music may tend towards different key signatures. Typically, the most frequently played note of a song will be its root note; i.e., the note of its key signature (e.g. 'A' is the root of A major). Guitar-based music is more likely to be played in E or A than F or A#, because E and A chords (and the chords of these key signatures) are easier to play on standard-tuned guitars than are F and A# chords. Other styles of music may have no special leaning towards these particular keys, or may lean toward other key signatures entirely. Therefore, all else being equal, if a song is detected to be in E or A, we might have additional reason to believe that it is in a guitar-based style, unlike if it is detected to be in F or A#.

Tempo (rhythmic feature)

Finally, an estimation of musical tempo is sometimes used as a feature for classification. Although a song may include several tempo changes throughout its duration, most popular music is set to a fixed number of beats per minute. Given the rhythmic emphasis placed on each beat of a bar (e.g. a typical 4/4 drumbeat will involve a high-hat played on each beat, with a snare drum on the 2nd and 4th of each bar), it is relatively straightforward to estimate the number of beats per minute, as long as these rhythmic emphases can be detected. Tempo may act as a suitable feature for genre classification because different genres may typically be played at different speeds. Within certain subgenres, this is trivially true (e.g. speed metal, as the name suggests, is played at high tempos, whereas doom metal tends to be played much more slowly); it is plausible that this applies to broader genre categories as well.

2.1.2 Other modalities

Analysis of audio features does not represent the only avenue for automatic music genre classification; other modalities such as text and imagery have been used as well. This section will explore these alternate approaches and justify the final choice of modality used in this project.

Textual

Songs and albums often have several text sources associated with them. These include primary sources, such as song titles and lyrics, and secondary resources, such as album reviews. Lyrically, different genres clearly tend to use different types of words. For example, rap lyrics are more likely to include topical words such as 'game' and 'rhyme' than metal lyrics, which tend to favour darker terms such as 'death' and 'shadow' (Fell & Sporleder, 2014, Experiment 1). Emergent properties

such as rhyme scheme and song structure have also been used as basis for classification. Fell and Sporleder (ibid.) used an n-gram model to classify lyrics into different genres, with some success, although some genres (e.g. rap) were more easily detectable than others (e.g. folk was often confused with blues or country). Mayer, Neumayer and Rauber (2008, 'Experiments') used a range of different feature combinations, and categorized lyrics with several classifiers, including k-Nearest Neighbours and Naïve Bayes.

Visual

There has been comparatively little work done on classification of music based on its album artwork. Automatic classification of images by genre has been attempted for other art forms, with some success. For example, Cujak et al. (2011) investigated whether paintings could be automatically classified into their art genres. Colour- and texture-based features were used as input to a range of classifiers including a neural network and a decision table, with promising final results. Album covers of disparate genres tend to differ in content and style (e.g. violent imagery in death metal album covers vs. its relative absence in k-pop artwork) and recently automatic classification of music by album artwork was successfully implemented by Oramas et. al (2018), as part of their multimodal experiments (detailed below).

Multimodal

The prospect of multimodal music classification - classification based on several modalities used in conjunction with one another - has received relatively little attention over the years, although interest is growing. In their 2018 paper (ibid.), Oramas et. al detail two experiments. The first employs separate neural networks to classify albums into distinct genres: first using audio features (of the songs), then using visual features (of the album art), and finally according to a 'multimodal feature space' made by fusing audio and visual representations. Their findings suggest that combining the representations yielded superior results to using either modality by itself. Their second experiment introduced text representations learned from album reviews. Neural networks were trained according to each modality, as well as for different modality combinations, with the most accurate approach taking all three modalities into account.

Schindler and Rauber (2015) have investigated whether analysis of the colours used in music videos could be used to predict the musical genre of the songs they accompany. When using visual features only, performance of the classifiers was still noticeably lower than that achieved using audio features such as Temporal Variants. The addition of visual features did appear to slightly increase the accuracies of the classifiers in some cases, but whether there was improvement varied with the type of classifier used (some accuracies decreased when both sets of features were considered). Thus, as regards automated music video classification, analysis of video content may represent an unnecessary addition to the process, if the audio for the music videos is readily available to be used and analysed.

Choice of Final Modality

Originally, this project was intended to further explore the possibility of multimodal genre classification, making use of textual and image sources as well as audio features. However, for this study, only audio features were used. There were several

motivations behind this choice. First, after consideration of the time and resources available, it was determined early on that multimodal classification was outside the scope of this project. Selecting one modality allowed for deeper, more thorough exploration of that particular modality, and probably yielded better results than would have been achieved through a multimodal approach. Secondly, text-based approaches typically make use of natural language processing techniques that were also determined to be outside the scope of this project's aims. Although genre classification of music via album artwork alone represents an interesting and novel possibility, it is not immediately clear that these types of classifier perform especially well, and there is relatively little existing research in this area to make use of. Finally, as discussed above, the majority of existing literature and resources in this area concern music genre classification via audio features. There also exist a number of useful and relevant audio analysis libraries which could be made use of in the process of feature extraction. Therefore, after these assessments had been made, an audio-based approach was chosen.

2.2 Classification Algorithms

The remainder of this chapter will look more closely at four machine learning algorithms that are frequently used for classification problems, and will include discussion of how such methods can be applied to the task of music genre recognition.

2.2.1 Artificial Neural Network

Artificial neural networks are a popular form of machine learning classifier, consisting of layers of connected nodes that transform input data to some kind of output (dependent upon the specific implementation). In the context of classification, neural networks learn models from training data, in order to predict the class of unseen data samples. Various kinds of neural network exist, including fully-connected, convolutional, and recurrent, and networks can be structured with varying numbers of nodes and layers according to design choices made by the engineer. A number of strategies exist for combating overfitting and reducing model complexity, and, when architected correctly, neural networks have proven to be especially effective for classification problems. The exact structure of the network used for this project will be provided in the 'models' section of this report.

A range of approaches can be taken toward neural network input. The shape of the first layer of the network will depend upon choices made in the feature engineering stage of the project. For example, in image recognition, the number of 'input nodes' will usually equal the number of pixels of each image of the dataset. Each pixel is represented by a number, and, if structured correctly, the network can be trained to learn 'features' of the images, e.g. the curve of a '2' in handwriting recognition. However, when features have been manually selected from the data, as is the case for this project, the number of input nodes is determined by the number of features extracted in this way.

Audio data can be described as 'time-series data', in that it describes sonic events that take place over time. Because of this, there exists the possibility of representing audio files as time-series vectors, in terms of features such as MFCCs. Recurrent Neural Networks make use of loops and are able to 'remember' information between stages of the network's processing. This makes them especially suited to sequential and time-based data (Karpathy, 2015, 'Recurrent Neural Networks'). Consequently,

RNNs have been used for audio classification problems such as speech recognition (Olah, 2015). The use of RNNs for music genre classification presents an interesting and potentially fruitful opportunity, especially the use of LSTMs (Irvin, Chartock & Hollander, 2016). However, due to time and resource constraints, use of RNNs was ultimately deemed to be out of scope for this particular project, and other architectures were experimented with instead.

2.2.2 Support-Vector Machine

SVMs are another type of supervised learning machine learning algorithm. In classification problems, data points of the same class will oftentimes cluster together on graphs that plot their feature values. In order to make predictions for new data points, a 'n-1 dimensional hyperplane' can be used to carve up the graph, where n = the number of features. A new data point falling on one side of the hyperplane will be predicted to be of one class; a data point falling to the other side will be predicted another. SVMs attempt to find the optimal hyperplane for a given dataset, ordinarily defined as the hyperplane that maximizes the margin between the classes.

SVMs can be initialized with a range of different kernel functions, which create different types of data divisors. A linear kernel function attempts to divide data points with a linear hyperplane, whereas a 'kernel trick' can be used to create polynomial decision boundaries, by effectively mapping the data points into higher-dimensional space so that a dividing hyperplane can be drawn. Depending on how disparate two given classes are, the clusters may be 'far apart' on the graph, or there may be some degree of overlap, and the success of a support vector machine implementation will partly depend on the ease with which the classes can be separated. If, as has been argued above, different musical genres produce distinct average feature values, then support vector machines may prove promising for music genre classification, as the clusters of each genre could be separated with hyperplane divisors relatively easily.

2.2.3 Random Forest

Random forest is a form of ensemble learning, meaning that it combines models with the aim of producing a model that is better than any of the individual models by itself. In applying the random forest technique to classification problems, a data point is put through a number of decision trees, and the predicted class of the data point is equal to the mode average of the outputs of the individual trees considered together. In order to predict the class of a sample, a decision tree passes the values of the features through a series of conditional statements. This continues until a 'leaf node' - which designates the class predicted - has been reached. In random forests, an element of randomness is added through 'feature bagging'.

Various hyperparameters can be tuned, such as the number of decision trees in the forest and the maximum number of features used per tree. A more detailed description of how the hyperparameters of the random forest implemented in this project were tuned can be found in the 'Models' selection of this paper. Random forests use bagging in order to reduce overfitting (Li, Xue & Zhang, 2018, 'Models'), and given the ubiquity of overfitting as a problem in machine learning, this is a quality that makes them attractive for classification problems. As such, random forest was deemed suitable to be one of the classification algorithms for this project.

2.2.4 Gradient Boosting Machine

Like random forest, boosting is another ensemble technique. Gradient boosting machines use decision trees in the context of an additive model, in which gradient descent is used to minimize the level of loss as more trees are added (Brownlee, 2019a). Gradient boosting has proven to be among the most effective machine learning algorithms, often outclassing the more well-known alternatives. For example, the XGBoost gradient boosting machine implementation has been used to win a number of machine learning competitions ('Awesome XGBoost' readme), and it has shown to be effective for both classification and regression problems. Due to its standout reputation, XGBoost was used to implement gradient boosting for this project. Hyperparameters include the minimum child weight, gamma, and the maximum depth of the trees; tuning of such hyperparameters will be discussed later in this report.

Chapter 3

Method

This chapter details the methods used to select a dataset, preprocess the data, extract and analyse features, and create and test the machine learning models. Python 3 was used for the development of this project, due to the extensive machine learning libraries and online resources available for it.

3.1 Dataset Selection

The choice of dataset plays an important role in any machine learning project. Oftentimes, there exists a number of potentially suitable datasets available, each with their own advantages and disadvantages, and the decision to choose one dataset over another can significantly influence the course of the project. This section will outline the primary options considered for this study, and a justification of the final choice made.

3.1.1 GTZAN Dataset

The GTZAN dataset, detailed in Tzanetakis (2002), is a relatively well-known resource in the field of MIR. It consists of 10 groups of 100 30-second song excerpts, with each group representing one genre, totalling 1000 audio files overall. The dataset comes pre-organised into folders - one per genre - making it fairly straightforward to navigate, and the balanced nature of the dataset makes it appealing for machine learning projects.

However, despite these attractive features, GTZAN bears a number of disadvantages. First, with only 100 items per class, it is a relatively small dataset, in the context of other machine learning projects. Deep learning is often considered to thrive on large amounts of data (Ng, 2015), and having fewer training examples means that the classifiers have less information from which to build their models. It therefore made sense to see whether a larger dataset would be available.

Further, Sturm (2013) has identified a number of issues with the dataset, including repetitions of songs (i.e. different excerpts of the same song appearing as different tracks in the dataset) and mislabeling of tracks (i.e. tracks being labelled as the wrong genre). For these reasons, GTZAN was considered to be unsuitable for this project.

3.1.2 The Million Song Dataset

The second dataset considered was the Million Song Dataset (Bertin-Mahieux et al., 2011). As implied by the name, the MSD consists of one million songs, spreading over a variety of genres and subgenres. One million data points is a large dataset,

and many good machine learning projects have been undertaken using far fewer training examples.

However, the MSD differs from other datasets such as GTZAN in a significant way: while GTZAN is able to provide access to the audio files of the dataset themselves (due to the songs' relative obscurity), the MSD deals with contemporary popular music, meaning that access to audio files cannot be provided, on grounds of copyright. A dataset of a million audio files would also take up several terabytes of data, making it difficult to download and store for many researchers. Instead, the dataset consists of pre-extracted features and metadata, such danceability and popularity, for each of the songs. While these may be interesting to analyse in another context, it was deemed necessary to be able to have access to the audio files themselves, in order to allow for extraction and analysis of audio features such as the timbre and pitch-content features detailed above. Therefore, another dataset was required.

3.1.3 The Free Music Archive

The final dataset considered for this project was the Free Music Archive (Defferrard et al., 2017a). This dataset consists of over 100,000 songs, with genre labels available for each track. Several different versions of the dataset exist, ranging from the 'small' version (8,000 30-second samples) to the 'full' version (all 106,574 songs in their entirety). The size of this dataset makes it ideal for classification purposes, and the fact that the audio files are available to download means that features from the audio can be extracted directly. Therefore, the Free Music Archive avoids the main flaws of both the GTZAN and the MSD, with respect to this particular project.

The final dataset downloaded for this study was the 'large' version, which consists of 30s samples of all 106,574 songs. 30-second samples are usually enough for classification purposes, and the use of smaller song excerpts makes the dataset much easier to acquire than if full tracks had to be downloaded. The large dataset is unbalanced - i.e. different genres were represented to different extents - but, as will be detailed in the next section, two smaller versions of this dataset were created, both genre-balanced, for purposes of classification.

3.2 Data Preprocessing

3.2.1 Preliminaries

Several stages of preprocessing were required in order to ready the dataset for classification (note: the changes described in this and the next paragraph were made with Microsoft Excel, not with code). First, 'tracks.csv' was downloaded from the FMA webpage (Defferrard et al., 2017b). This provided information on the genres associated with each song. Lines which contained no information in the 'genre_top' field were deleted, as some tracks were not given a 'top genre', but instead were given a list of genres in another column. This was done as part of clarifying the problem, in order to discount songs that were not given a single genre and had apparently been marked to suggest that they may have been classifiable into more than one.

After this, several genres were removed from the dataset. 'Spoken' tracks were deleted because spoken-word poetry does not really constitute music, at least with regard to the particular project at hand. 'Experimental' tracks were deleted because experimental music is broad and diverse, and songs under the label of 'experimental' may bear little sonic similarity to one another, making the genre hard to define

and potentially making the tracks very difficult to classify. 'Instrumental' tracks were deleted because a song from any genre can be made instrumental simply by removing the vocals, meaning that many tracks under the label may bear little similarity to one another (arguably, 'instrumental' is not even a genre at all). Finally, 'international' music was discounted because again, there may be too much diversity under such a broad label.

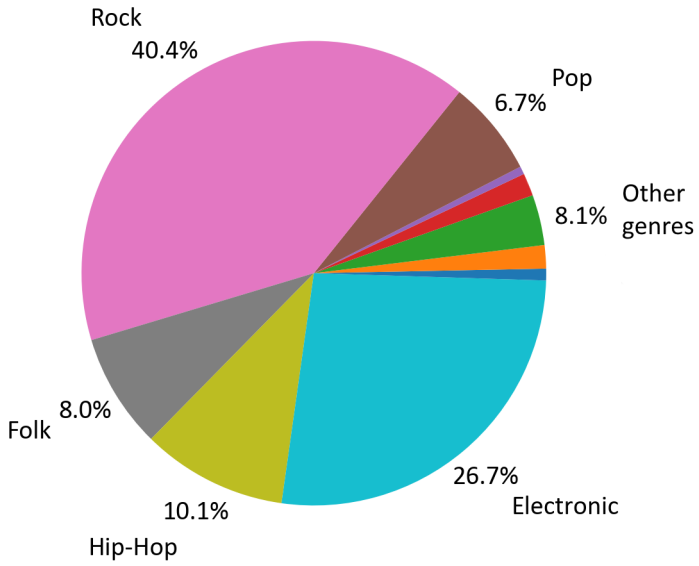

FIGURE 3.1: Genre distribution

As can be seen from the pie chart in figure 3.1 (made with Matplotlib and Paint.NET), which visualises the dataset after these changes were made, the overall distribution of genres was still heavily imbalanced. Many classifiers, such as random forests, are known to cope poorly with imbalanced data; it therefore made sense to find ways to balance the dataset. To do this, all but the five most populous genres - Rock, Pop, Folk, Hip-Hop, and Electronic - were set aside. Then, two groups of songs were created. The first was made with undersampling; the second, supplementary version of the dataset was made with a combination of undersampling and oversampling (via SMOTE) - each with an equal number of tracks per genre. These two groups were titled the 'standard dataset' and the 'augmented dataset' and are described below, after a description of the features that were chosen to be extracted.

3.2.2 Feature Extraction

In order to represent the tracks numerically, 54 audio features were extracted from each track. These values represent the average over the entire track (i.e., a track's MFCC1 value was defined as the average value of MFCC1 across all windows of that track). The first 41 were extracted using LibROSA audio analysis library (librosa.github.io/librosa); the final 13 were extracted using Aubio (aubio.org), which required conversion from mp3 to WAV format first. In total, this consisted of: 30 MFCCs, spectral centroid, spectral bandwidth, spectral contrast, spectral flatness, spectral rolloff, chroma stft, chroma cqt, chroma cens, tempo, zero crossing rate, root mean square, energy, high-frequency content, spectral complex, spectral phase,

spectral difference, Kullback-Liebler, modified Kullback-Liebler, spectral flux, spectral slope, spectral spread, spectral skewness, spectral kurtosis, and spectral decrease. Several of these features were discussed earlier in this paper in terms of their usefulness in music genre recognition; in general, the features chosen were selected because they represent sonic information relevant to this type of classification. 'Tempo' was the only rhythmic feature extracted, and 'chroma cqt', 'chroma_stft' and 'chroma_cens' were the only pitch-content features extracted; the rest all related to the timbre and texture of the tracks. All of the Aubio features used are described in their documentation as 'spectral features' (Brossier, 2018).

Feature scaling was used in the creation of the following two datasets, in order that each feature played an equivalent role when put through the machine learning classifiers. However, not each feature was equally informative, as shall be discussed in the analysis section of this report.

Here is an example of a song from the Electronic class, in terms of its first five features. MFCC1: 0.684962572 MFCC2: 0.666046288 MFCC3: 0.555326254 MFCC4: 0.45139384 MFCC5: 0.543886787.

3.2.3 The Standard Dataset

The creation of the standard dataset was relatively straightforward. From the five genres remaining after the preliminary trimming, 'Pop' was the least populous, with only 2331 song samples. Therefore, to create a balanced genre distribution, undersampling was used. As part of the preparation for the augmented dataset (described below), features from 3000 Rock songs, 3000 Hip-Hop songs and 3000 Electronic songs were extracted, along with features from 2801 Folk songs and 2331 Pop songs (for each of the latter two classes, these were the maximum number of samples available). Feature vectors from 2331 songs from each genre were then set aside, creating a balanced dataset with 11655 songs in total. This version of the dataset will be henceforth referred to as the 'standard dataset'.

3.2.4 The Augmented Dataset

In order to take advantage of the larger number of samples in the Electronic, Rock and Hip-Hop genres, an additional, supplementary version of the dataset was made. As previously discussed, certain forms of machine learning do especially well when more data is available, and this additional version of the dataset was created to see whether classification accuracy could be improved with more training data. In order to bring the size of the Folk set and the Pop set up to the size of the others - which each contained 3000 tracks - some method of generating synthetic data was required.

Focus initially turned to a form of data augmentation that involved manipulation of existing data via noise injection and alteration of pitch and speed. However, these methods were hypothesized to cause potential problems in the context of classification. Take alteration of speed, for example. The motivation behind using estimated tempo as a basis for genre classification is that different genres may have different average tempos. Creating new samples by altering the tempo of existing samples could potentially change the average tempo of the genre, potentially muddying the classification process.

Instead, SMOTE (Synthetic Minority Oversampling Technique) was chosen to balance the additional dataset; a method of oversampling that is much faster to process than data augmentation in the manner described above. It involves 'adding in'

data points between some of the samples and their nearest neighbours in the under-represented classes (Chawla et al., 2002). It is important to note that SMOTE should only be performed on training data, and, in order to prevent issues later on, five lists of files (one list per genre) that would be used as test data for each genre were created before the SMOTE process took place. This meant that when it came time for classification, rather than performing a typical train-test split, a manual train-test split had already been performed, in which all the songs not in the previously created 'test lists' would be used for training, and only the songs present in the 'test lists' would be used for testing. SMOTE was applied to Folk and Pop's training data, and after the new data points were created, the resultant csv was saved. This version of the dataset, which contained 3000 samples per class, shall henceforth be referred to as the 'augmented dataset'.

3.3 Genre and Feature Analysis

At this point, prior to the classification phase of the project, two stages of analysis were performed. The first involved plotting the average value for each class, for each feature extracted. The second involved comparing and ranking each of the features by their F-value, to see which would be most relevant in classification.

3.3.1 Genre Analysis

In order to compare classes by their average value for each feature, the standard dataset csv was used, as it made sense to inspect only non-synthetic data. The average values were computed in the following way (rough pseudocode representation):

1. For each feature f:
    (a) For each genre g:
        i. Add up all g's instances, in terms of f
        ii. Divide by the number of instances in g (in this case, 2331)
    (b) Plot the results.

The final graphs are viewable at a link given in the appendix, with an example graph viewable in figure 3.2. This graph plots the average values for the MFCC1 feature, which is the MFCC that deals with the lowest frequency range. For this feature, the value for Folk is the lowest, whereas the value for Hip-Hop is the highest.

It might be supposed that Folk's position as the lowest value may be due to it being generally quieter than the other genres listed here, meaning that lower values would be expected for each of the MFCCs. However, for some of the other MFCCs, such as MFCC6, Folk's average value was actually the highest of all genres. This seems to suggest that the different genres activated different frequency ranges to different degrees: genres like Pop may have had more activation in the lowest frequencies, suggesting a bassier mix, whereas Folk seems to have had more activation in the middle frequencies (e.g. those dealt with by MFCCs 6 to 9), suggesting a mix with greater emphasis on the mids. This chimes with what we know about the instruments often used for each of these genres: Folk music that consists of just acoustic guitar and vocals will not be as bassy as Pop music played with a loud, thudding kick drum and synthesized bassline.

A visual scan of the graphs seems to suggest that Folk often has quite different values from the other classes - for many features, it has the lowest value, but for

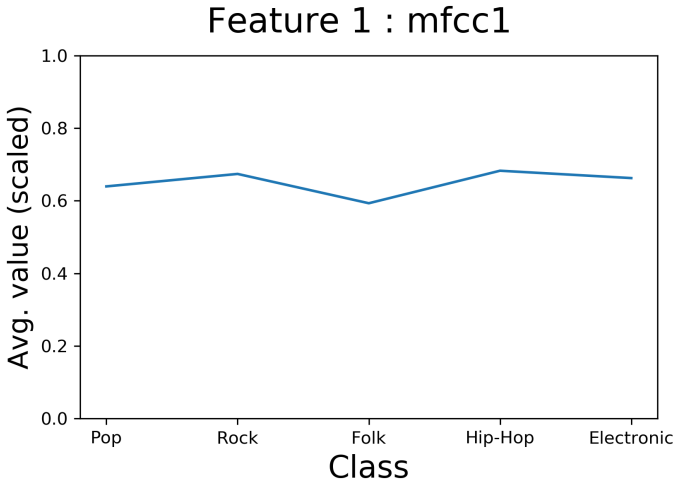

FIGURE 3.2: Average class values for MFCC1

several, it has the highest. This may help to explain why, as discussed later, Folk was (on average) one of the best-classified genres: it seems to be quite distinct in terms of its feature values.

3.3.2 Feature Analysis

The second stage of analysis involved ranking each of the features by their F-value, again using the standard dataset. This analysis made use of the SelectKBest function, which is part of the sklearn.feature_selection module. Given a set of data points, SelectKBest allows a program to filter out those features which are least relevant to classification, by ranking each feature according to a score function, and setting aside those features that are not in the top 'K' (given as an argument). For classification problems, one of chi squared, f_classif or mutual_info_classif is usually chosen as the score function (Scikit-learn.org, 2019a). F_classif was deemed appropriate for the variables in this instance (because they are continuous, not categorical), which leads to computation of the F-values of the features.

Ordinarily, SelectKBest is used as a form of filter-based feature selection; i.e. as a means of setting aside any features that may be hindering performance of the classifier. However, it is also possible to use the function to retrieve the scores that each feature was given. In this case, the scores for each of the 54 features were retrieved, and plotted in the form of a bar chart, viewable at a link in the appendix. A table is presented in figure 3.3 for legibility.

As can be seen, there was a wide distribution in terms of how informative and useful each feature was calculated to be. Two of the most informative features were chroma_cens and chroma_cqt; a noteworthy result given how they were calculated. In this study, the chroma features extracted for each track were averaged - rather than one feature per note, the average score across *all* notes was calculated, resulting in one chroma_cens and one chroma_cqt value per track. This is a novel approach to calculating chroma features that seems to have been successful in this instance, given the features' F-values. The results seem to suggest that some genres simply contained 'more notes' than others (i.e. more notes were played in the songs) - something that the genre averages graphs for chroma_cens and chroma_cqt appear to indicate.

It makes sense that, in general, the lower MFCCs were calculated to be more useful than the higher MFCCs, because more of the spectral information is contained within the lower frequencies (after a certain point, there is no more spectral information left to analyse, because the frequency bands eventually become higher than the frequencies instruments typically produce). Tempo, a rhythmic feature, achieved a fairly low F-value, which is not especially surprising. Although, as earlier discussed, some genres are typically faster than others, it may be that in this particular dataset, there is significant overlap between the genres. The genre averages graph for this particular feature seems to imply little difference between the average scores for each genre, which may help to explain why it is one of the less-informative features on offer in this instance.

SelectKBest was used, with varying values of K, in an attempt to improve the accuracy of the classifiers (detailed in the next section). However, filtering out any of the features actually resulted in performance that was either no different or slightly worse for all four of the classification algorithms, compared to keeping all features in. Therefore, all available features were used as part of the classification process.

Feature	F-Value	Feature	F-Value	Feature	F-Value
1. decrease	719.19	**20.** mfcc9	238.4	**39.** mfcc5	68.0
2. chroma_cens	709.5	**21.** spectral_contrast	229.72	**40.** spectral_centroid	59.09
3. chroma_cqt	585.41	**22.** hfc	227.58	**41.** zcr	52.38
4. phase	552.09	**23.** kl	220.99	**42.** mfcc17	47.17
5. mfcc2	548.09	**24.** rms	198.66	**43.** mfcc27	44.83
6. spread	477.96	**25.** skewness	187.77	**44.** mfcc21	43.72
7. mkl	462.44	**26.** mfcc24	175.0	**45.** kurtosis	41.19
8. mfcc3	435.72	**27.** mfcc22	174.75	**46.** mfcc19	32.59
9. slope	409.12	**28.** mfcc13	130.26	**47.** mfcc29	25.36
10. mfcc1	394.48	**29.** mfcc28	125.69	**48.** mfcc15	23.36
11. specdiff	349.82	**30.** mfcc30	123.59	**49.** spectral_bandwidth	17.7
12. mfcc4	321.91	**31.** mfcc20	114.79	**50.** spectral_flatness	16.81
13. mfcc10	308.41	**32.** mfcc8	108.08	**51.** mfcc25	16.42
14. mfcc12	295.71	**33.** chroma_stft	103.19	**52.** mfcc23	15.74
15. specflux	278.41	**34.** mfcc14	100.52	**53.** spectral_rolloff	15.44
16. mfcc6	269.76	**35.** mfcc26	88.3	**54.** mfcc7	7.74
17. complex	263.85	**36.** mfcc16	87.65		
18. mfcc11	262.36	**37.** mfcc18	85.08		
19. energy	256.95	**38.** tempo	78.57		

FIGURE 3.3: Table of f-values.

3.4 Models

This section will first detail the cross-validation procedure used to train and validate each machine learning model. It will then detail how each of the four classifiers was implemented, and how hyperparameters were selected, as well as details of the training process.

3.4.1 Train-Test Split

A test size of 10% was used in all experiments. When using the standard dataset, the split was performed using Scikit-learn's train-test split function. For the augmented dataset, the train set and test set had to be imported manually and separately, as they had been exported separately as different csvs during the earlier SMOTE process.

3.4.2 Cross Validation & Grid Search

Five-fold cross validation was used on the training data for all experiments. This was used in combination with grid search through the Scikit-learn GridSearchCV function, in order to help find the best-performing hyperparameters. Cross-validation has a number of advantages over splitting a dataset into training, validation and test sets. For example, it allows the classifier to train on more of the data than it would if both a validation set and a test set had to be used, because there is no need for a validation set as the training set can also be used for validation. The diagram in figure 3.4 (Scikit-learn.org, 2019b) explains how five-fold cross-validation is structured with respect to a given dataset, and because five-fold cross validation was used for this project, it also applies to this project's structure as well.

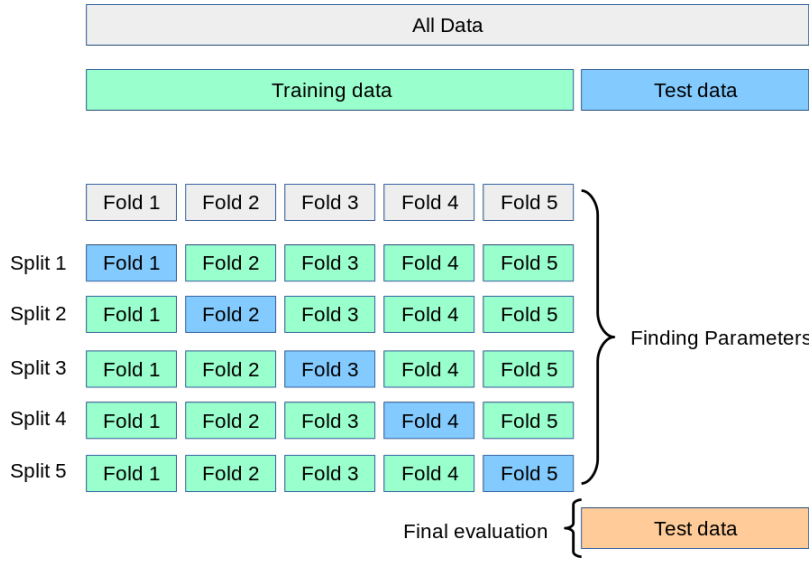

FIGURE 3.4: Cross validation

After each classifier had been trained and validated on the training data, final accuracy scores were produced by testing it on the testing data.

3.4.3 Neural Network

The neural network was the first classifier to be implemented. A Sequential Keras model was used because it is a straightforward model that is relatively easy to tune, and this allowed more time to be spent tuning the other classifiers.

Structure of the Network

The input layer to the network was given the same number of nodes as features of the data (fifty-four). The output layer, which was activated with Softmax, was given five units, one for each genre. Several designs for the internal structure of the network were tested. In general, the more layers and units used, the more a network is able to learn complex features from the data. However, care must be taken not to overfit the training data; hence, regularisation was used to prevent this from occurring (see below). More complex architectures were initially experimented with, but the best result was achieved with a five-layer network in which the number of nodes decreased as the layers went on, as is common for neural networks of this kind (Smith, 1997, ch. 26). The three hidden Dense layers were, like the input layer, activated with Relu. Relu is considered a good all-round activation function for a variety of reasons, e.g. it has been shown to be faster than alternatives like tanh in many cases (e.g., see (Krizhevsky, Sutskever & Hinton, 2012, section 3.1)).

Optimizer

Adam was used as the optimizer because, like Relu, it has been shown to perform well in a wide variety of systems (e.g., see Ruder (2017), 'Which Optimizer to Use?').

Regularisation

Dropout layers were added in between the layers of the network for regularisation purposes. Several values were tested; it was revealed that a value of 0.2 had the best effect on reducing overfitting without causing underfitting.

Training & Testing

A number of different epochs and batch sizes were experimented with. Grid search was performed to help find the best values. For the standard dataset, 150 epochs and a batch size of 100 were found to be best. For the additional, augmented dataset, 250 epochs and a batch size of 150 were selected. Testing was performed by comparing the model's predicted labels for the test data with the data's true labels, and returning the accuracy score as a percentage.

3.4.4 Support-Vector Machine

The support-vector machine was implemented with SVC from sklearn.svm. Grid search is a time-consuming technique, meaning not every possible parameter could be tested; therefore only a select few were used in the tuning process (this approach was also adopted for the random forest and gradient boosting machine).

Kernel Type

The kernel type determines the way in which the decision boundary of a SVM is drawn. Linear kernels are used when the data is considered to be linear separable,

but given the high-dimensional nature of the data in question, this was deemed to be unlikely. Several kernel types were experimented with; initial testing revealed that the 'rbf' (radial basis function) kernel proved the most successful, and this was selected for the final model.

C

'C' is another relevant hyperparameter, used to determine the level of regularization and model complexity, by specifying the degree to which misclassified points should negatively impact the score of a model. Several values of C were tested.

Gamma

'Gamma' is used to determine the length of the influence of a single training example in the calculation of the dividing hyperplane. Several values were tested.

Shrinking

When the shrinking hyperparameter is set to 'True', the shrinking technique 'tries to identify and remove some bounded elements, so a smaller optimization problem is solved' (Chang & Lin, 2011, section 5.1). Testing revealed that setting the shrinking hyperparameter to 'True' yielded better results.

Training & Testing

Grid search was again used to help identify the best hyperparameters, and after a number of tests, the following hyperparameters were chosen: kernel=rbf, C=5000, gamma=0.1, shrinking=True. These proved to be the best found for both the standard dataset experiments and the augmented dataset experiments.

3.4.5 Random Forest

The RandomForestClassifier from sklearn.ensemble was used for the random forest, because, like sklearn's SVC, it is relatively straightforward to implement and tune.

Number of Estimators

One of the primary hyperparameters to set is the number of decision trees (also known as estimators) in the forest. More trees usually leads to better accuracy; the main downside being increased running time of the algorithm. A range of values were tested.

Criterion

This hyperparameter is used to 'measure the quality of a split' during the creation of decision trees in the forest (Scikit-learn.org, 2019c). Sklearn's implementation supports Gini Impurity and Information Gain. Both values were tested during the grid search process.

Max Features

Another important hyperparameter is the maximum number of features that is used per tree. By default, there is ordinarily no limit on this number. Several values were tried in the initial testing of the algorithm; it was found that the 'automatic' (i.e. default) value yielded the best result.

Max Depth

Increasing the maximum depth of each tree in the forest may allow for a more detailed learning process. Choosing to place no limits on the depth of the trees can improve accuracy, although this does increase model complexity and thus the likelihood of overfitting. This hyperparameter was tested during the grid search process.

Training & Testing

After grid search was used to identify the best hyperparameters, the following were chosen: n_estimators=1000, criterion=gini, max_features=auto, max_depth=100. These applied for both the standard and augmented datasets.

3.4.6 Gradient Boosting Machine

XGBoost's classifier 'XGBClassifier' was used to implement the gradient boosting machine, due to its strong reputation (see Chapter 2's section on gradient boosting machines for more).

Min Child Weight

The min child weight refers to the 'Minimum sum of instance weight (hessian) needed in a child' (XGBoost, 2019a). Higher values of 'min child weight' will result in models that are less complex, so as a parameter it can be used to prevent overfitting. Several values were tested.

Learning Rate

A faster learning rate will result in shorter training times, but means that the model is more likely to overfit the data. A number of values between 0 and 1 were experimented with.

Max Depth

The 'max depth' hyperparameter refers to the maximum tree depth for the base learners (XGBoost, 2019b). Again, increasing this value leads to greater model complexity, which can be advantageous but can lead to overfitting. Several values were tested.

Training & Testing

The final hyperparameters chosen (for both the standard and the augmented dataset) were as follows: min_child_weight=5, learning_rate=0.3, max_depth=10.

Chapter 4

Results

This chapter displays the results in the form of highest accuracies achieved for each experiment, as well as confusion matrices to help visualise classification accuracies for individual genres. All confusion matrix values have been normalised and rounded. Confusion matrices for the standard dataset are to the left; confusion matrices for the augmented dataset are to the right. A discussion of the results also follows.

4.1 Discussion

4.1.1 Genres

The average genre scores across the standard dataset were as follows: Electronic: 60%, Folk: 73%, Hip-Hop: 74%, Pop: 41%, Rock: 73%.

Although performance varied with each classification algorithm, similar trends were identified in the results of each. For example, across most experiments, Folk and Hip-Hop were classified the most accurately. Folk may have been one of the most successfully classified genres because, as discussed earlier, its average scores for a number of the features were significantly different to those of other genres (refer to the MFCC1 diagram in the 'Feature and Genre Analysis' section of this report for an example). This implies the presence of sonic differences that were detected successfully in the training of the algorithms. Hip Hop's place as the most accurately classified genre is harder to explain; it may be that the presence of rapped vocal passages (i.e. vocal passages that are not sung) helped differentiate it, if such passages led to distinctive feature trends. Further investigation of the dataset would be required to understand why Hip-Hop was classified the best.

Electronic tracks were misclassified as Hip-Hop more than they were misclassified as any other genre, and vice-versa. Given the nature of the two genres, this makes sense: Hip-Hop beats tend to be made using electronic samples such as those of the Roland R-808 drum machine, and the two genres bear much similarity in terms of production. Instrumentally, Hip-Hop typically bears very little in common with Folk music - the latter tends to be based around acoustic and electric guitars - and this explains why there was very little confusion between the genres, less than 5% in all but one experiment.

Across all experiments, Pop was misclassified more than any other genre by a significant margin. Although accuracy was notably higher than random chance (20%), its final average accuracy was nearly half of that of the best-classified genre (Folk). It is unclear exactly why this is the case. A potential explanation may lie in the fact that as a concept, 'Pop' music may be more broadly and ambiguously defined than the other genres in the dataset. To take two contemporary examples, Ed Sheeran's music is often thought of as Pop, despite being mostly guitar-based, as is

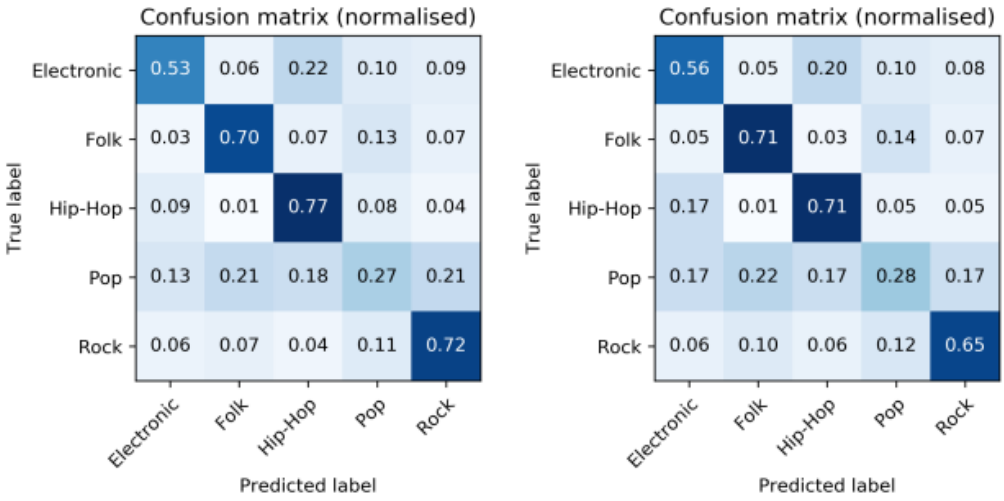

FIGURE 4.1: Neural network confusion matrices. Standard dataset accuracy: 60%. Augmented dataset accuracy: 58%.

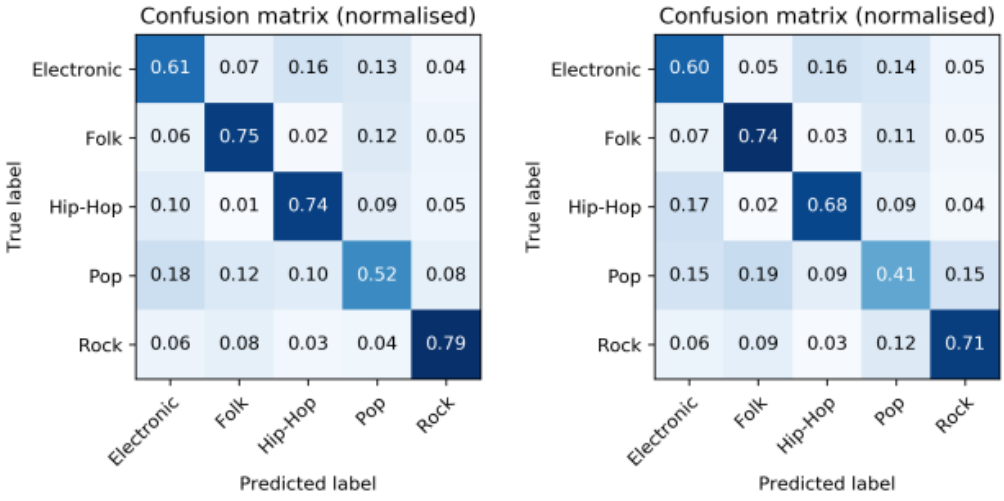

FIGURE 4.2: SVM confusion matrices. Standard dataset accuracy: 68%. Augmented dataset accuracy: 63%.

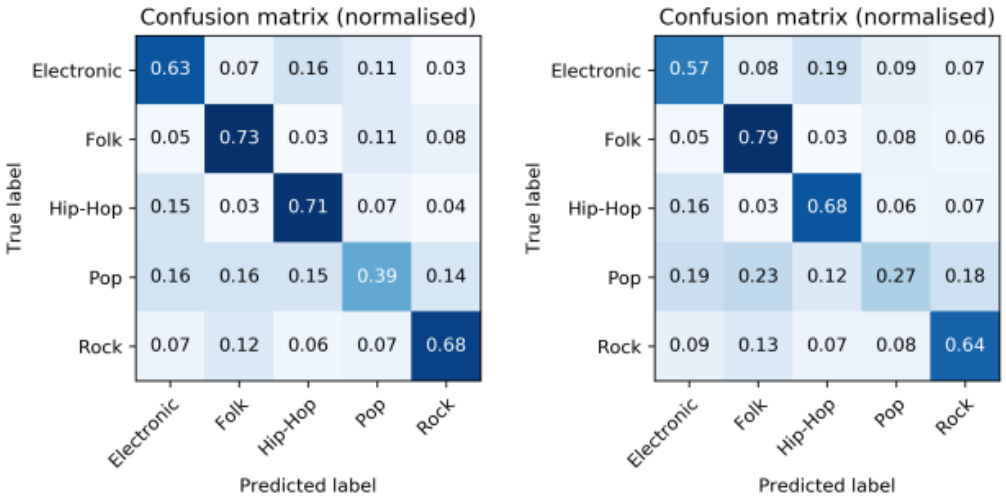

FIGURE 4.3: Random forest confusion matrices. Standard dataset accuracy: 63%. Augmented dataset accuracy: 59%.

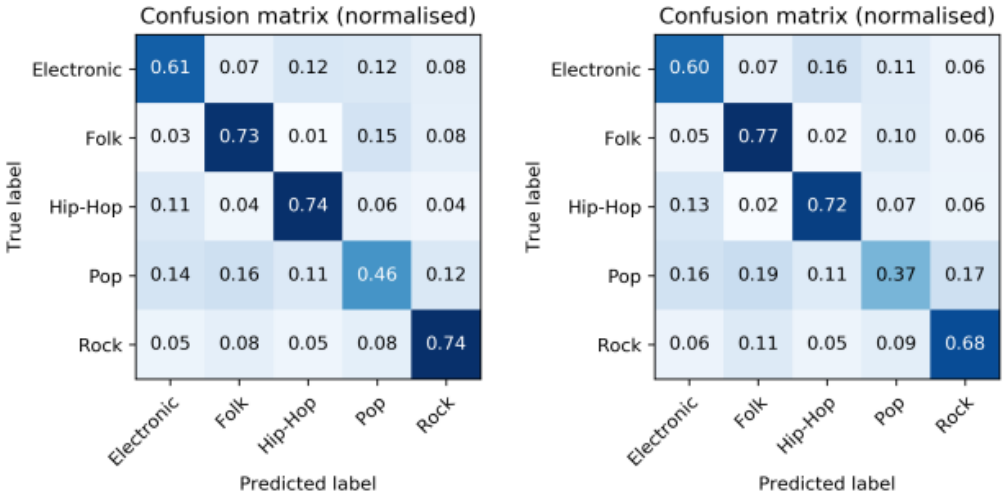

FIGURE 4.4: Gradient boosting machine confusion matrices. Standard dataset accuracy: 66%. Augmented dataset accuracy: 63%.

Katy Perry's, despite being mostly synthesizer-based. Perhaps the curators of the original dataset were working with a definition of 'Pop' that was looser than their definitions of the other genres. However, it is worth noting that Hip-Hop also encompasses a reasonably broad range of production styles, and yet it achieved the best score of all the genres. Further investigation into the dataset is needed; more discussion of Pop's classification results takes place in the next chapter.

4.1.2 Standard vs Augmented

Across all experiments, results were slightly better for the standard dataset than for the augmented dataset. It appears that in this instance, a larger version of the dataset did not lead to improved results. The exact cause of this reduction in accuracy is unclear. A potential explanation may lie in the order in which certain parts of the process were carried out. After further research, it was revealed that performing SMOTE before cross validation (i.e., not during it) can sometimes lead to impaired results (Altini, 2015). This may help to explain why the results were worse for the augmented dataset.

A notable trend between the scores of the standard dataset and those of the augmented dataset is that, in most cases, Pop was classified significantly worse on the augmented dataset than on the standard version. The majority of other scores remained approximately the same, and most of the reduction in average accuracy between the datasets appears to be attributable to the reduction in accuracy for Pop. This might relate to the fact that Pop was the class which was augmented the most, having had approximately 700 new data points created for it. Having 2331 samples per class may well have been enough in this instance, although alternate approaches to data augmentation may yield more promising results in the future.

4.1.3 Comparing the Algorithms

This project initially set out to compare machine learning algorithms in terms of how suited they were to the problem of music genre classification. In terms of accuracy, on the standard dataset, the final ranking from best to worst is: SVM (average score: 68%), Gradient Boosting Machine (66%), Random Forest (63%), Neural Network (60%).

To better understand the difference in accuracy between the classifiers, once again it is informative to look at the individual genre scores. When trained on the standard dataset, the neural network achieved an accuracy of 27% for Pop, whereas the SVM was able to achieve the significantly higher score of 52%. However, for the other genres, the difference in scores between the classifiers was not nearly as pronounced, suggesting that the primary advantage of the SVM over the neural network was the ability to classify Pop songs correctly much more of the time. It was also recognized that the SVM was the fastest of the four classifiers, adding to its advantage.

The gradient boosting machine outperformed the random forest, on both the standard dataset and the augmented dataset. This is in line with previous findings that suggest that gradient boosting machines may be more accurate than random forests under a range of conditions (e.g. see Kopczyk (2019)), although, of course, this will depend on the classification problem in question, and the random forest algorithm is still worth testing as it may outperform a gradient boosting machine in certain circumstances.

The neural network did not perform especially well in comparison to the other classifiers. The exact cause of this is unclear; it is true that, in general, neural networks offer more 'customizability' than the other classifiers discussed (i.e. it is possible to construct them with varying numbers of nodes, layers, etc.). This means that there are many more ways a neural network can be parameterized and structured by researchers; the cost of this may be that it is harder to structure one optimally, as there are more factors to take into account. Deep learning is typically performed on datasets with more features than were present for this experiment, which may explain why it failed to outperform the other algorithms in this instance.

As discussed previously, the 'order' of the rankings - i.e., which genre was classified most accurately, which was classified second-most accurately, etc. - was largely the same for each algorithm. This is significant, because it highlights trends in the underlying data, and it exemplifies how such trends can, in some ways, be just as important as the algorithm used in the context of a classification experiment. Nevertheless, notable differences were observed between the results of each algorithm, and overall, the results indicate that the support-vector machine was the most promising algorithm of the four considered in this study.

4.1.4 Comparison with Other Studies

The original curators of the FMA ran several initial genre classification experiments on the 'medium' version of the dataset (Defferrard et al., 2017a). The best accuracy achieved was 63%, using a support-vector machine on features from LibROSA. It is noteworthy that the best performing classifier in their experiment was the same as the best performing classifier in the present study, and further supports the notion that SVMs might be one of the best classifiers for the purpose at hand. Although it is interesting to compare the results, a direct comparison between the two is not quite valid; a significant difference being that the original experiments were performed on a different subset of the data than was curated for this version.

As previously discussed, for this study, experimentation of different values of 'K' in SelectKBest indicated that the best accuracy was attained when none of the extracted features were dropped. In line with this, initial testing revealed that classification accuracy was significantly better when both the features extracted with LibROSA and the features extracted with Aubio were used, instead of using the features extracted with either library by themselves. Other LibROSA features were extracted for this experiment than were extracted for the original FMA experiment (such as 10 additional MFCCs) and, as the F-value analysis indicated, these extra values were useful in the context of classification. These considerations hint at the tentative conclusion: extraction of more (relevant) features may generally lead to better results in the context of music genre classification.

Chapter 5

Evaluation

Following on from the evaluation and interpretation of the results in the previous chapter, this section will evaluate the core aspects of the project as a whole, with the aim of better understanding the results achieved.

5.1 Evaluation of the Dataset

In general, understanding a dataset in greater detail will often lead to a better understanding of the results of the classifiers trained and tested on it. Ideally, a classifier should be able to generalise to real-world data, i.e., to examples outside the bounds of the particular dataset it was trained on. However, if the dataset is not especially reflective of the real-world phenomena it is supposed to describe, classification accuracy may be harmed as a result (both on testing data and on other examples). After the classification phase of this project, it made sense to manually inspect the dataset, to see whether this may have been taking place.

It is important to note that in terms of music genre classification, humans can and often do disagree as to what genre label is appropriate for a given song. Despite the subjectivity involved in this process, there must also be a level of objectivity involved too (if genre labels are to have any universal meaning at all). Given that, in this project, over 10,000 30-second audio samples were used - a total of more than 83 hours of audio - manually verifying the genre label of every sample was clearly out of scope. However, listening to a random selection of tracks seemed to indicate that although most tracks were fairly categorized, the genre label of a significant number of them seemed questionable. It may be that the confusion exhibited by some of the classifiers in this project is reflective of the mislabelling of some of the tracks in the original dataset (there may be a number of 'Electronic' tracks that should, in fact, have been labelled as 'Hip-Hop' tracks, for instance). As discussed in the previous section, perhaps a tighter definition of 'Pop' music would have led to improved results as well. The dataset could therefore be described as suffering from a degree of 'label noise' (for more on label noise, see Frénay et al. (2014)).

Future work on the FMA could involve creating a smaller version of the dataset, constructed by manually listening to tracks from the given genres, and marking those which seem to best represent the genre. Tracks that did not seem to reflect the genre label could be set aside. The end result would be a smaller, class-balanced version of the dataset, in which each track had been manually verified to be representative of its genre label. Although potentially quite a time-consuming task, depending upon the size of the final dataset chosen, training classifiers on this dataset could yield better results, and models that could be better applied to real-world data.

5.2 Evaluation of Method

The machine learning algorithms selected for experimentation were, in retrospect, strong choices to have made. Initially, the neural network was chosen to be the primary algorithm evaluated by the project. The addition of the SVM and the random forest helped push the project forward and deepen understanding; ultimately, they both outperformed the neural network in terms of accuracy. Choosing to implement the gradient boosting machine further enhanced the project, and was a worthwhile decision as the algorithm achieved the second-highest scores overall. The decision to compare algorithms, rather than merely focusing on the optimisation of one, allowed for deeper insights and a more worthwhile project overall. Use of grid search and hyperparameter tuning also allowed for measurably improved results.

As was earlier discussed, other approaches to the neural network could have resulted in better performance. The use of a recurrent neural network, and the extraction of features as time-series data, could have led to improved accuracy and the learning of complex features. Unfortunately, resource constraints prevented this option from being fully explored; thus it may represent an opportunity for future research in this area. It is worth noting that the neural network did achieve a score significantly higher than random chance, and its overall accuracy only trailed those of the other classifiers by several percent. Therefore, despite the potential improvements that could have been made, its performance was still fairly successful.

The use of SMOTE for data augmentation ultimately did not lead to improved results. However, this was likely due to an error not in how the method was used, but of when. Use of SMOTE within the cross-validation process, i.e. in each training fold, may have led to higher accuracy on the test data. As such, the technique represents a possible avenue for future research.

In line with the findings of the analysis, every feature extracted contributed to the overall success of the classifiers, and discounting any of them decreased accuracy scores (with the largest improvements obtained from those features with the highest F-values). As previously mentioned, it may be that the more features available for analysis, the better the classifiers would have performed. Extracting time-series data from the audio may have improved performance, although it may also have substantially increased the time taken to train the classifiers, given that it would have resulted in a far higher number of features than was ultimately used.

In evaluating the method, another factor to consider is the choice of 'accuracy' as the performance metric for the classifiers. Accuracy is an intuitive measure of model quality in classification problems; as such, it was a sound choice for this particular study. However, it does not constitute the only option available to data scientists. For example, logarithmic loss can be used to evaluate the confidence scores given to different classes in the classification process (Brownlee (2019b)). This can be useful as it allows researchers to learn more about how their classifiers are making predictions. If more time had been available, other performance metrics (in addition to accuracy) would have been used to evaluate the models in this study, as this would have offered an alternative perspective and may have revealed comparative advantages and disadvantages of the algorithm implementations that are currently unknown.

5.3 Evaluation of Project Management

The management of the project was relatively smooth, though there may have been room for improvement in a number of areas. For example, Spyder was used as the IDE for developing the project, as it was specifically designed with scientists in mind. Though it proved fairly intuitive and extensive in its capabilities, alternative options such as PyCharm could have been tested and compared, to see whether they were more suited to the project at hand.

Various productivity strategies helped move the project forward successfully. The use of an online project log proved useful in terms of recording results, tracking progress and keeping note of useful links and resources. Use of a University computer could have allowed for better performance in some areas, e.g. in time spent training the neural network. However, use of a personal laptop had many logistical advantages and was overall worthwhile in terms of productivity.

5.4 Personal Achievements & Experience

In terms of personal achievement, I feel that I have learned a lot about machine learning and Python-based data science through conducting this study. Although I took the introductory A.I. module earlier this year, which introduced me to many relevant topics such as regression and neural networks, the module was mostly theoretical and did not require us to implement these algorithms in code. As such, learning how to develop neural networks as part of software systems represented a central challenge of the project. SVMs, random forest and gradient boosting machines were also not covered during the module's content, and so I had to learn about these algorithms independently as well.

Teaching myself a new programming language represented a difficult but rewarding task. Having focused primarily on Java during the taught component of the degree course, I needed to learn the basics of Python and a number of its libraries, as well as how to use a Python-based IDE. Python was chosen as the language for development because of its wide range of machine learning and graph-plotting libraries, and the substantive instructional content available to help navigate them. Scikit-learn was chosen to implement the SVM and random forest because it is a popular machine learning library that is accessible for beginners, and there are ample resources to learn about it online. Keras and Tensorflow were selected for similar reasons. Learning about these libraries will come in useful later on, as I plan to use them in personal research projects.

Navigating the relevant scientific literature represented an interesting and exciting challenge. Past experience in essay subjects helped the project write-up significantly, and my long-term interest in mixing and music tech helped me to understand some of the features relevant to this subfield, e.g. MFCCs. Overall, my passion and enthusiasm for music helped substantially, particularly in the discussion of the results, and I am glad that I chose this topic to study.

Chapter 6

Conclusion

This study compared machine learning algorithms in their suitedness to the task of music genre classification. Of the approaches surveyed, the support-vector machine was shown to be the most promising. Classification was based upon 54 features extracted from each song of the dataset, most of which pertained to spectral information. Analysis of F-scores indicated that some features were significantly more informative than others; an observation that could prove useful for future research.

Several avenues for further investigation present themselves. First, as has been noted, the use of time-based features could allow for classification based on more complex musical properties, as could be performed by algorithms such as recurrent neural networks. Second, other types of feature could be investigated in terms of their relevance for this type of classification problem. Only one explicitly rhythmic feature was extracted for this study - average tempo - thus greater emphasis on rhythm may represent an interesting opportunity for researchers in this field. Key signature and melody could also be worthy of greater focus. Finally, more work on curating high-quality datasets could be performed. This could involve working with and reducing the size of pre-existing datasets, combining existing datasets together, or the collection of new tracks that are not currently in the public domain. Overall, music genre classification remains an interesting and worthwhile challenge for both academic institutions and businesses alike, and there is plenty of room for further study and analysis.

Chapter 7

Bibliography

Altini, M. (2015). Dealing with Imbalanced Data: Undersampling, Oversampling and Proper Cross-Validation, marcoaltini.com. [online]. Available at: https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampl ing-oversampling-and-proper-cross-validation [Accessed Sep 2019].

Awesome XGBoost readme. [online]. Available at: https://github.com/dmlc/xgboost/tree/master/demomachine-learning-challenge-winning-solutions [Accessed Sep 2019].

Bachu, R. G., Kopparthi, S., Adapa, B. and Barkana, B.D. (2008). 'Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal', ASEE Regional Conference, Pittsburgh, Pennsylvania, 22-25 June.

Bäckström, T. (2019). Zero-crossing rate, Introduction to Speech Processing. [online]. Available at: https://wiki.aalto.fi/display/ITSP/Zero-crossing+rate [Accessed Sep 2019].

Bertin-Mahieux, T., Ellis, D., Whitman, B. and Lamere, P. (2011) 'The Million Song Dataset', ISMIR 2011 - 12th International Conference on Music Information Retrieval, Miami, Florida, 24-28 October.

Brossier, P. (2018). Spectral Features, Aubio 0.4.9 documentation. [online]. Available at: https://aubio.org/manual/latest/py_spectral.html

Brownlee, J. (2019a). A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning, Machine Learning Mastery. [online]. Available at: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/ [Accessed Aug 2019].

Brownlee, J. (2019b). Metrics To Evaluate Machine Learning Algorithms in Python, Machine Learning Mastery. [online]. Available at: https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/ [Accessed Sep 2019].

Chang, C., Lin, C. (2013). LIBSVM: A Library for Support Vector Machines, ACM Transactions on Intelligent Systems and Technology 2.

Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research 16 (2002), AI Access Foundation , USA, pp. 321–357.

Čuljak, M., Mikuš, B., Jež, K. and Hadjić, S. (2011). 'Classification of art paintings by genre', Proceedings of the 34th International Convention MIPRO, Opatija, Croatia, 23-27 May.

Defferrard, M., Benzi, K., Vandergheynst, P. and Bresson, X. (2017a). 'FMA: A Dataset For Music Analysis', ISMIR 2017 - 18th International Society for Music Information Retrieval Conference, Suzhou, China, 23-27 October.

Defferrard, M., Benzi, K., Vandergheynst, P. and Bresson, X. (2017b). FMA GitHub. [online]. Available at: https://github.com/mdeff/fma [Accessed Aug 2019].

Fell, M. and Sporleder, C. (2014). 'Lyrics-based Analysis and Classification of Music', COLING 2014 : 25th International Conference on Computational Linguistics, Dublin, Ireland, 23-29 August.

Frénay, B,. and Kabán, A. (2014). 'A Comprehensive Introduction to Label Noise', ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges, Belgium, 23-25 April.

Giannakopoulos, T. and Pikrakis, A. (2014). Introduction to Audio Analysis: A MATLAB Approach. Academic Press.

Gjerdingen, R. O. (2008). Scanning the Dial: The Rapid Recognition of Music Genres, Journal of New Music Research, Volume 37. Routledge.

Gouyon, F., Pachet, F. and Delerue, O. (2000). 'On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds', International Conference on Digital Audio Effects, Verona, Italy, 7-9 December.

Goodman, T., and Batten, I. (2018). 'Real-Time Polyphonic Pitch Detection on Acoustic Musical Signals', 2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Louisville, Kentucky, USA, 6-8 December.

Irvin, J., Chartock, E. and Hollander, N. (2016). Recurrent Neural Networks with Attention for Genre Classification. [online]. (Published 2016). Available at https://www.semanticscholar.org/paper/Recurrent-Neural-Networks- with-Attention-for-Genre-Irvin-Chartock/6da301817851f19107447e4c72e682e3f183ae8a [Accessed Sep 2019].

Jensen, J. H., Christensen, M. G., Murthi, M. H. and Jensen, S. H. (2006). 'Evaluation of MFCC estimation techniques for music similarity', 14th European Signal Processing Conference, Florence, Italy, 4-8 September.

Karpathy, A. (2015). The Unreasonable Effectiveness of Recurrent Neural Networks. [online]. (Published 21 May 2015). Available at http://karpathy.github.io/2015/05/21/rnn-effectiveness/ [Accessed Aug 2019].

Kopczyk, D. (2019). From Decision Trees to Gradient Boosting. [online]. (Published 18 April 2019). Available at http://dkopczyk.quantee.co.uk/tree-based/ [Accessed Sep 2019].

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). 'ImageNet Classification with Deep Convolutional Neural Networks', 26th Annual Conference on Neural Information Processing Systems, Lake Tahoe, Nevada, 3-8 December.

Lawton, G. (2017). How Pandora built a better recommendation engine. [online]. (Published 09 Aug 2017). Available at: https://www.theserverside.com/feature/How-Pandora-built-a-better-recommendation-engine [Accessed Aug 2019].

Li, H., Xue, S. and Zhang, J. (2018). Combining CNN and Classical Algorithms for Music Genre Classification. Institute for Computational Mathematical Engineering, Stanford University.

Lyons, J. (2012). Mel Frequency Cepstral Coefficients. [online]. Available at: http://practicalcryptography.com/miscellaneous/machine-learning/guide -mel-frequency-cepstral-coefficients-mfccs/ [Accessed Aug 2019].

Mayer, R., Neumayer, R. and Rauber, A. (2008). Rhyme and Style Features for Music Genre Classification by Song Lyrics, ISMIR 2008 - 9th International Conference on Music Information Retrieval, Drexel University, 14-18 September.

Meinard.mueller. (2016). ChromaFeatureCmajorScaleScoreAudioColor.png. [Image]. Available at: https://en.wikipedia.org/wiki/File:ChromaFeatureCmajorScaleScoreAudioColor.png [note: original image has been cropped]

Ng, A. (2015). What Data Scientists Should Know About Deep Learning, Extract Data Conference. [online]. Available at: https://www.slideshare.net/ExtractConf/andrew-ng-chief-scientist-at-baidu [Accessed Aug 2019].

Olah, C. (2015). Understanding LSTM Networks. [online]. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ [Accessed Aug 2019].

Oramas, S., Barbieri, F., Nieto, O. and Serra, X. (2018). 'Multimodal Deep Learning for Music Genre Classification', Transactions of the International Society for Music Information Retrieval, CIUP, Paris, 24-26 September.

Panagakis, Y., Benetos, E. and Kotropoulos, C. (2008). 'Music Genre Classification: A Multilinear Approach', ISMIR 2008 - 9th International Conference on Music Information Retrieval, Drexel University, 14-18 September.

Peeters, G. (2004). 'A large set of audio features for sound description (similarity nd classification) in the CUIDADO project', IRCAM, Analysis/Synthesis Team.

Ruder, S. (2017). An Overview of Gradient Descent Optimization Algorithms. [online]. Available at: https://arxiv.org/abs/1609.04747

Schindler, A. and Rauber, A. (2015). 'An Audio-Visual Approach to Music Genre Classification through Affective Color Features', ECIR 2015 - 37th European Conference on Information Retrieval, Vienna, Austria, 29 March - 2 April.

Scikit-learn.org (2019a). Feature selection — scikit-learn 0.21.3 documentation. [online]. Available at: https://scikit-learn.org/stable/modules/feature_selection.htmlunivariate-feature-selection [Accessed Sep. 2019].

Scikit-learn.org (2019b). Cross-validation: evaluating estimator performance — scikit-learn 0.21.3 documentation. [image]. Available at: https://scikit-learn.org/stable/modules/cross_validation.html [Accessed Sep 2019].

Scikit-learn.org (2019c). sklearn.ensemble.RandomForestClassifier — scikit-learn 0.21.3 documentation. [online]. Available at: https://scikit-learn.org/stable/ modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accessed Sep 2019].

Smith, S. (1997). The Scientist Engineer's Guide to Digital Signal Processing, California Technical Pub.

Sturm, B. (2013). The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. [online]. Available at: https://arxiv.org/abs/1306.1461.

Two!Ears team. (2015). Spectral Features, Two!Ears Documentation. [online]. Available at http://docs.twoears.eu/en/latest/afe/available-processors/ spectral-features/ [Accessed Sep 2019].

Tzanetakis, G. (2002). Musical Genre Classification of Audio Signals. In: IEEE Transactions on Speech and Audio Processing (Vol: 10, Issue 5), IEEE, pp 293-302.

XGBoost (2019a). XGBoost Parameters - xgboost 1.0.0-SNAPSHOT documentation. [online]. Available at: https://xgboost.readthedocs.io/en/latest/parameter.html.

XGBoost (2019b). Python API Reference - xgboost 1.0.0-SNAPSHOT documentation. [online]. Available at: https://xgboost.readthedocs.io/en/latest/python/python_api.html.

Appendix A

Appendix

A.1 Code Authors

All code is original, except in the case of methods from imported libraries, and in the case of methods noted to have been re-used (please see code comments for original sources of these methods).

A.2 Code Instructions

Each script in the repository is labelled from 1-15. Files 1-9 relate to feature extraction and the creation of the final csvs. Files 10 and 11 are used to analyse the data, and files 12-15 are the implementations of the classification algorithms. Each script is self-contained (i.e. it does not use code from any of the other scripts).

Files 1, 3, 4, 5, 6, 7, 8 and 9 are not suitable to be run; most require access to files such as the tracks of the 93GB fma large dataset (and the feature extraction files take several hours to run each). Files 2, 10, 11, 12, 13, 14 and 15 can all be run, so long as the relevant libraries, which are listed at the top of each file, are installed. The comments in each of the files specify what the file does and whether a specific directory structure is required for the file to be run. The csvs must also be present in the same folder as the Python scripts. All code is written in Python 3. Note that the 'standard dataset' and the 'augmented dataset' may be referred to as the '2331 dataset' and the '3000 dataset' in the code, as these were the datasets' working titles.

A.3 External Links

Feature averages: https://drive.google.com/open?id=1cOouKJvZTHlj7fqoTu3dtKHfwBR70g1K

F-values bar chart: https://drive.google.com/open?id=1VGJoIa63jWh18njNDkj-Baum7kOMvisO