Chapter 14

Composer Classification Models for Music-Theory Building

Dorien Herremans, David Martens, and Kenneth Sörensen

Abstract The task of recognizing a composer by listening to a musical piece used to be reserved for experts in music theory. The problems we address here are, first, that of constructing an automatic system that is able to distinguish between music written by different composers; and, second, identifying the musical properties that are important for this task. We take a data-driven approach by scanning a large database of existing music and develop five types of classification model that can accurately discriminate between three composers (Bach, Haydn and Beethoven). More comprehensible models, such as decision trees and rulesets, are built, as well as black-box models such as support vector machines. Models of the first type offer important insights into the differences between composer styles, while those of the second type provide a performance benchmark.

14.1 Introduction

Automatic composer-identification is a complex task that remains a challenge in the field of music information retrieval (MIR). The problems we address here are, first, that of constructing an automatic system that is able to distinguish between music written by different composers; and, second, that of identifying the musical properties that are important for this task. The latter can offer interesting insights for music theorists.

A data-driven approach is taken in this research by extracting global features from a large database of existing music. Based on this data, five types of classification

Dorien Herremans · Kenneth Sörensen
ANT/OR, University of Antwerp Operations Research Group, Antwerp, Belgium
e-mail: {dorien.herremans, kenneth.sorensen}@uantwerpen.be

David Martens
Applied Data Mining Research Group, University of Antwerp, Antwerp, Belgium
e-mail: david.martens@uantwerpen.be

Preprint accepted for publication in the Springer book on Computational Music Analysis, ISBN: 978-3-319-25929-1, pages 369-392, Editor: David Meredith

model (decision tree, ruleset, logistic regression, support vector machines and Naive Bayes) are developed that can accurately classify a musical piece between Bach, Haydn or Beethoven. Most of the developed models are comprehensible and offer insight into the styles of the composers. Yet, a few black-box models were also built as a performance benchmark.

14.2 Prior Work in Music Information Retrieval

The task of composer classification belongs to the domain of Music Information Retrieval (MIR), a multidisciplinary field concerned with retrieving and analysing multifaceted information from large music databases (Downie, 2003). The field of MIR has grown rapidly in recent years due to the digitization of the music industry. In 2011 alone, the European consumer expenditure on digital media exceeded 33 billion euros (Stenzel and Downes, 2012).

The first mention of the term MIR is due to Kassler (1966), who named the programming language he developed to extract information from music files "MIR". The early work done on the topic of computational music analysis is described in more detail by Mendel (1969).

Recently, many topics have been explored in the field of MIR. Examples of these are the content-based music search engine "query-by-humming", which allows a user to hum a tune in order to search for the original song in a large database (Ghias et al., 1995; Tseng, 1999). Pfeiffer et al. (1997) developed a system that can detect violence in video soundtracks. The techniques developed by Wold et al. (1996) are used to, for instance, identify different types of human speaker (e.g., female versus male). Music similarity research is another topic that has been explored by MIR scientists, in which the similarity of two musical pieces is measured (Berenzweig et al., 2004). For more detailed surveys of research done in the field of MIR, see Typke et al. (2005), Weihs et al. (2007) and Casey et al. (2008). In the current chapter, however, the focus will be on composer classification.

14.2.1 Classification Systems

The task of music classification can be seen as building models that assign one or more class labels to musical pieces based on their content. These models are often evaluated based on accuracy, i.e., the number of correctly classified instances versus the total number of instances. It should be noted however that accuracy is not always the best performance measure, for instance in the case of an unbalanced dataset. In this research, the area under the curve (AUC) of the receiver operating characteristic (ROC) is used to evaluate the performance of the models. This metric, which takes into account the true positives versus the false positives, is more suitable since the dataset is slightly skewed (see Fig. 14.1) (Fawcett, 2004). Most existing studies on music classification only evaluate their model based on the accuracy rate. When comparing the performance of previous studies, one should take into account that accuracy is not always comparable.

While the specific task of composer classification has not received a great deal of attention in the literature (Backer and van Kranenburg, 2005; Geertzen and van Zaanen, 2008; van Kranenburg, 2008), music classification has been applied to a plethora of other topics. Machine learning tools have been applied to classify pieces by, for example, cultural origin (Whitman and Smaragdis, 2002), geographic region (Hillewaere et al., 2009), timbre (Cosi et al., 1994), mood (Laurier et al., 2008), artist (Mandel and Ellis, 2005), hit ranking (Herremans et al., 2014) and genre (Chew et al., 2005; Tzanetakis and Cook, 2002).

There is an important difference between the data representation in classification models that work with *audio* data (e.g., WAV files) and *symbolic* data (e.g., MIDI files). The types of feature that can be extracted from a dataset and used to build models are vastly different for the two categories. Encouraging results have been obtained on classifying music audio data. For example, Whitman et al. (2001) built neural networks and support vector machine models for artist identification based on audio features. These models achieved classification accuracies of 91% in a one-in-five artist space over a small corpus (35 songs) and 70% correct over a larger corpus (50 songs) with ten artists. However, in this chapter, the focus will be on building *comprehensible* models, therefore we chose to work with symbolic features since they are generally more meaningful for music theorists.

In symbolic music classification, there are two main approaches. On the one hand, there are systems that use a language modelling approach, including n-grams and hidden Markov models. They take into account *local features* that change over the course of the musical fragment (Pérez-Sancho et al., 2008). On the other hand are systems that extract a finite vector of *global features* from each song (Steinbeck, 1982).

A study by Volk and van Kranenburg (2012) showed that recurring motifs are important when classifying songs into tune families. This suggests that locality, and thus local features, are important factors in classifying songs. This first approach, based on local features, was confronted by the challenge of efficiently representing data for machine learning. Techniques such as multiple viewpoints offer a possible solution to this problem (Conklin and Witten, 1995; Pearce et al., 2005). The problem of classifying folk songs by region was tackled with a language model by Li et al. (2006). They achieved an accuracy rate of 72.5% on a data set of European folk songs from six different regions. Pérez-Sancho et al. (2008) modelled chord progressions and melodies as n-grams and strings, and constructed genre classification models based on this representation. They were able to achieve an 86% accuracy rate for three broad genres. A success rate of 96.6% was obtained by Hontanilla et al. (2013) with an n-gram approach on discriminating between Shostakovich and Bach.

The second approach, based on global features, is the one that we focus on in this chapter. Other studies that have used this approach include that of Steinbeck (1982), who used global features to cluster melodies into meaningful groups such as hymns, children's songs and hunting songs. Eerola et al. (2001) used global features for assessing similarity. Ponce de León and Iñesta (2003) used global melodic, harmonic and rhythmic descriptors for style classification. The ensemble method based on a neural network and a k-nearest neighbour developed by McKay and Fujinaga (2004) for genre classification used 109 global features and achieved an accuracy of 98% for root genres and 90% for leaf genres. Moreno-Seco et al. (2006) also applied ensemble methods based on global features to a style classification problem. Bohak and Marolt (2009) used the same type of features to assess the similarity between Slovenian folk song melodies. Herlands et al. (2014) combined typical global features with a novel class of local features to detect nuanced stylistic elements. Their classification models obtained an accuracy of 80% when discriminating between Haydn's and Mozart's string quartets.

A small number of studies have compared both types of feature and compared their performance on the folk music classification problem. Jesser (1991) created classification models based on both features using the dataset from Steinbeck (1982). Her conclusion was that global features do not contribute much to the classification. A study by Hillewaere et al. (2009) found that event models outperform a global feature approach for classifying folk songs by their geographical origin (England, France, Ireland, Scotland, South East Europe and Scandinavia). In another study with European folk tunes, Hillewaere et al. (2012) compared string methods with n-gram models and global features for genre classification of 9 dance types. They concluded that features based on duration lead to better classification, no matter which method is used, although the n-gram method performed best overall. Van Kranenburg et al. (2013) obtained similar results for the classification of Dutch folk tunes into tune families. Their study showed that local features always obtain the best results, yet global features can be successful on a small corpus when the optimal subset of features is used. Similar results were obtained in a study to detect similarity between folk tunes (van Kranenburg, 2010).

Since the dataset used in our research was relatively small and global features were a relatively simple way of representing melodies that can be easily processed by different types of classifier, we opted to use a carefully selected set of global features in the research reported in this chapter.

14.2.2 Composer Classification

While a lot of research has been done on the task of automatic music classification (see Sect. 14.2.1), the subtask of composer classification remains relatively unexplored.

Wołkowicz et al. (2007) used n-grams to classify piano files into groups of five composers. Hillewaere et al. (2010) also used n-grams and compared them with global feature models to distinguish between string quartets by Haydn and Mozart. The classification accuracy of their trigram approach for recognizing composers for string quartets was 61% for violin and viola, and 75% for cello. Another system that classifies string quartet pieces by composer has been implemented by Kaliakatsos-Papakostas et al. (2011). They used a Markov chain to represent the four voices of the quartets as a monophonic melody. A classification success rate of 59 to 88% was achieved when classifying between two composers with their weighted Markov chain model. The study performed by Pollastri and Simoncelli (2001) had a lower accuracy rate than the previously discussed research. The Hidden Markov Models they designed for the classification of 605 monophonic themes by five composers had an accuracy of 42% on average. Backer and van Kranenburg (2005) have applied statistical pattern recognition to the problem of distinguishing Bach from Handel, Telemann, Haydn and Mozart using 20 features in different time slices throughout the pieces. They concluded that it is "very possible to isolate the style of J.S. Bach from Telemann, Handel, Haydn or Mozart".

Van Kranenburg and Backer (2004) used 20 global style markers based on properties of counterpoint. A decision tree (C4.5) (Quinlan, 1993) and nearest neighbour classification algorithm are built on a database of 320 pieces from the eighteenth and early nineteenth century. They are able to achieve a fairly low error rate. Although the features are described in the paper, a detailed description of the models is missing. Similar features based on counterpoint are used by Mearns et al. (2010) to develop a C4.5 decision tree and naive Bayes models. Their models correctly classified 44 out of 66 pieces with 7 groups of composers. The resulting decision tree could give music theorists insight into the differences between styles of composers; however, the tree is not shown in the paper.

In the following sections, the dataset used in this research is discussed together with the chosen features. These are then used to build accurate and comprehensible classification models (Sect. 14.4). The resulting models are described in detail in this chapter and give insight into the differences between the styles of Haydn, Beethoven and Bach.

14.3 Data Sources

The range of features that can be extracted from music depends heavily on the type of file they have to be extracted from. Computational music analysis is typically performed on two types of music file, those based on *audio signals* and *structured files*. The first category includes files representing audio signals (e.g., WAV and MP3) and files containing the values of features derived from audio signals (e.g., short-term Fourier spectra, MFCCs, etc.). While there is a large quantity of music available in such formats, the types of feature that result from analysing these files (e.g., spectral flux and zero-crossing rate (Tzanetakis et al., 2003)) are not the most comprehensible, especially for music theorists. For the purpose of our research, it was therefore more appropriate to work with files of the second type, i.e., structured files.

Structured, symbolic files, such as MIDI files, contain high-level structured information about music. MIDI files, in particular, describe a specific way of performing a piece and contain information such as the start, duration, velocity and instrument of each note. These files allow us to extract musically meaningful features.

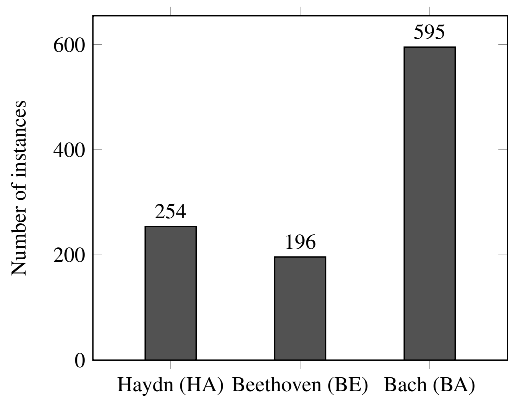


Fig. 14.1 Class distribution of the dataset for the different composers

14.3.1 Dataset

We used MIDI files from the KernScores database, since they could be loaded into jSymbolic, the software used to extract the features (see Sect. 14.3.2). This is a large collection (7,866,496 notes in 108,703 files) of virtual music scores made available by the Center for Computer Assisted Research in the Humanities at Stanford University (CCARH) (CCARH, 2012). It must be pointed out that MIDI files are a performance representation, often recorded by a human playing the score, and they therefore do not guarantee an accurate representation of the score (van Kranenburg and Backer, 2004). However, the KernScore database contains MIDI files with correct timing and pitch information, since they were derived from MuseData files that were produced by manual encoding of printed scores (Sapp, 2005).

Three composers, Johann Sebastian Bach (BA), Ludwig van Beethoven (BE) and Franz Joseph Haydn (HA), were chosen for this research since there are many pieces by these composers in the KernScores database, allowing us to create more accurate models. An overview of the composer class distribution of the selected 1045 movements is given in Fig. 14.1. Almost all available pieces by these composers were included in our dataset, except for a few very short fragments.

14.3.2 Feature Extraction

There are a number of tools available for extracting musical features from symbolic files, such as Humdrum (Huron, 2002) and MIDI toolbox for Matlab (Eerola and Toiviainen, 2004). Due to its ease of use, compatibility with polyphonic music, good support and the quality of the resulting features, the software package jSymbolic was used to extract the features from the dataset (McKay and Fujinaga, 2007). jSymbolic is contained within jMIR, the Java-based Open Source software toolbox designed for automatic music classification (McKay and Fujinaga, 2009).

Table 14.1 Features extracted with jSymbolic

Variable	Feature description
x_1	Chromatic Motion Frequency - Fraction of chromatic intervals
x_2	Melodic Fifth Frequency
x_3	Melodic Octaves Frequency
x_4	Melodic Thirds Frequency
x_5	Most Common Melodic Interval Prevalence
x_6	Most Common Pitch Prevalence[a]
x_7	Most Common Pitch Class Prevalence[b]
x_8	Relative Strength of Most Common Intervals - fraction of intervals belonging to the second most common / most common melodic intervals
x_9	Relative Strength of Top Pitch Classes[c]
x_{10}	Relative Strength of Top Pitches[c]
x_{11}	Repeated Notes - fraction of notes that are repeated melodically
x_{12}	Stepwise Motion Frequency

[a] Pitch refers to MIDI pitch number.
[b] Pitch class refers to MIDI pitch number mod 12.
[c] Top pitch or top pitch class refers to the most common pitch or pitch class in the piece.

jSymbolic is able to extract 111 different features. However, for this research, not all of them are meaningful or computationally easy to implement in classification models. Multidimensional features, nominal features, features related to instrumentation or that depend upon the key were excluded from the dataset. This resulted in a selection of twelve one-dimensional features that output normalized frequency information related to intervals or pitches. All of these features are represented in Table 14.1. They are measured as normalized frequencies and offer information regarding *melodic* intervals and pitches.

A second reason, other than musical meaningfulness, for keeping the feature set small is to avoid overfitting the models (Gheyas and Smith, 2010). Having a limited number of features allows a thorough testing of a model, even with limited instances, and can thus enhance the quality of a classification model (McKay and Fujinaga, 2006). This way we avoid the “curse of dimensionality”, where the number of labelled training and testing samples needed increases exponentially with the number of features.

In the following sections, five types of classification model are developed, based on the extracted features.

14.4 Classification Models

Predictive models can be used not only for classification, but also in theory-building and testing (Shmueli and Koppius, 2011). Using powerful models, in combination with high-level musical features enables us to construct models that give useful insights into the characteristics of the style of a composer. The first models in this section (i.e., rulesets and trees) are of a more linguistic nature and therefore fairly easy to understand (Martens, 2008). Support vector machines and naive Bayes classifiers are more black-box, as they provide a complex non-linear output score. Using pedagogical *rule extraction* techniques like Trepan and G-REX (Martens et al., 2007), comprehensible rulesets can still be extracted from black-box models. However, this falls outside the scope of this chapter. The ruleset described in Sect. 14.4.2 simply *induces* the rules directly from the data. While this research focuses on building comprehensible models, some black-box models were included to provide performance benchmarks.

The open source software, Weka, is used to create five different types of classification model, each with varying levels of comprehensibility, using supervised learning techniques (Witten and Frank, 2005). This toolbox and framework for data mining and machine learning is a landmark system in this field (Hall et al., 2009). jSymbolic, used to extract features as described in Sect. 14.3.2 above, outputs the features of all instances in ACE XML files. The jMIR toolbox offers a tool to convert these features into Weka ARFF format (McKay and Fujinaga, 2008).

Table 14.2 shows the results of using each of these types of model for composer classification on the dataset described in Sect. 14.3.1 above. For some types of model, such as decision trees and rulesets, multiple models were built with different levels of comprehensibility. In these cases, Table 14.2 shows the results for the best performing model of that type. The results are based on a run with stratified 10-fold cross validation (10CV), where the dataset is divided into 10 folds, with 9 used for model building and 1 for testing. This procedure is repeated 10 times, so that each fold is used once for testing. The AUC and accuracy values shown in Table 14.2 are the average results over the 10 test sets. The entire dataset is used to build the resulting final model, which can be expected to have a performance at least as good as the 10CV performance. In order to compare the performance of the different models,

Table 14.2 Performance of the models with 10-fold cross-validation

Method	Accuracy	AUC
C4.5 Decision tree	*80%*	*79%*
RIPPER ruleset	*81%*	*85%*
Logistic regression	*83%*	92%
Naive Bayes	*80%*	90%
Support vector machines	**86%**	**93%**

$p < 0.01$: italic, $p > 0.05$: bold, best: bold.

a Wilcoxon signed-rank test was conducted, the null hypothesis of this test being that there is no difference between the performance of a model and that of the best model.

14.4.1 C4.5 Tree

In this section we describe a decision tree classifier, which is a simple, yet widely used and comprehensible classification technique. Even though decision trees are not always the most competitive models in terms of accuracy, they are computationally efficient and offer a visual understanding of the classification process. A decision tree is a tree data structure that consists of decision nodes and leaves, where the leaves specify the class value (i.e., composer) and the nodes specify a test of one of the features. A predictive rule is found by following a path from the root to a leaf based on the feature values of a particular piece. The resulting leaf indicates the predicted composer for that particular piece (Ruggieri, 2002).

J48 (Witten and Frank, 2005), Weka's implementation of the C4.5 algorithm, is used to build decision trees (Quinlan, 1993). A "divide and conquer" approach is used by C4.5 to build trees recursively. This is a top-down approach, which repeatedly seeks a feature that best separates the classes, based on normalized information gain (i.e., difference in entropy). After this step, subtree raising is done by pruning the tree from the leaves to the root (Wu et al., 2008).

Three decision trees were built (T1, T2 and T3), each with a different setting for the confidence factor (confFactor) and the minimum number of instances per leaf (minNumObj). A low confidence factor will result in more pruning but a less accurate model. Requiring a greater minimum number of instances per leaf will also reduce the size of the tree, so that it becomes more easy to understand; on the other hand, it will also reduce accuracy. The settings for the three trees are shown in Table 14.3, together with their performance results and the sizes of the resulting trees (sizeTree), including the numbers of leaves (numLeaves).

The first model (T1) was heavily pruned, so that the resulting tree was compact. As expected, the accuracy and AUC values, respectively 73% and 72%, were lower than the less pruned models. Figure 14.2 shows the resulting classifier.

A second, slightly less pruned, model (T2) was built (see Fig. 14.3). As shown in Table 14.3, the accuracy and AUC values were slightly higher. The tree itself was slightly bigger (8 leaves), but still comprehensible.

Table 14.3 Performance and settings of the C4.5 decision trees (10CV)

ID	confFactor	minNumObj	numLeaves	sizeTree	Accuracy	AUC
T1	0.01	100	3	5	73%	72%
T2	0.01	50	8	15	76%	78%
T3	0.25	2	54	107	80%	79%

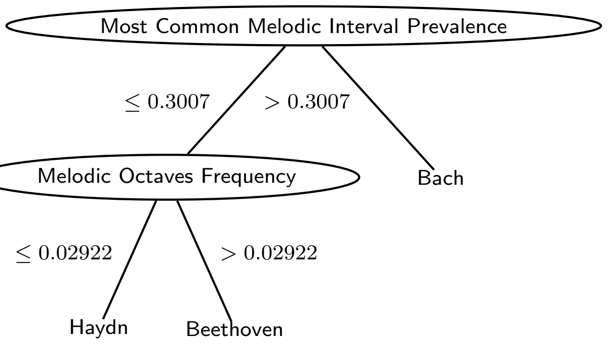

Most Common Melodic Interval Prevalence
≤ 0.3007
> 0.3007
Melodic Octaves Frequency
Bach
≤ 0.02922
> 0.02922
Haydn
Beethoven

Fig. 14.2 C4.5 decision tree (T1)

We will not show the third tree (T3) here, as it is too large (54 leaves). With an accuracy of 80% and an AUC of 79%, it is the most accurate model; however, it is also the least easy to comprehend. The confusion matrix of this best model is displayed in Table 14.4. This matrix reveals that there is a relatively higher misclassification rate between Haydn and Beethoven and between Haydn and Bach. This could be due to the fact that the dataset was larger for Bach and Haydn. It might also be due to the fact that Haydn and Beethoven's styles are indeed more similar, as they had a greater amount of chronological and geographical overlap in their lives, just as Haydn and Bach had chronological overlap. Furthermore, Haydn was once Beethoven's teacher (DeNora, 1997). Bach and Beethoven on the other hand never lived in the same country, nor during the same time period (Greene, 1985). For more details on the musicological similarities and background of Haydn and Beethoven, the reader is referred to Rosen (1997).

When examining the three models, the importance of the 'Most common melodic interval prevalence' feature for composer recognition is clear, as it is the root node of

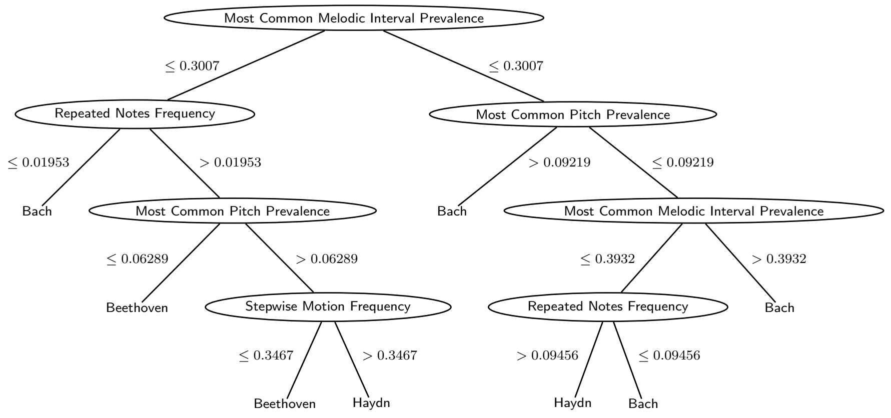


Fig. 14.3 C4.5 decision tree (T2)

all three models. It indicates that Bach focuses more on one particular interval than the other composers. Bach also seems to use fewer repeated notes than the other two composers. The ‘Melodic octaves frequency’ feature indicates that Beethoven uses more octaves in melodies than Haydn.

14.4.2 RIPPER

As with the decision trees just described, the rulesets presented in this section were built with an inductive rule-learning algorithm (as opposed to rule-extraction techniques such as Trepan and G-REX (Martens et al., 2007)). They are comprehensible models based on “if-then” rules, that are computationally efficient to implement.

JRip, Weka’s implementation of the ‘Repeated Incremental Pruning to Produce Error Reduction’ (RIPPER) algorithm, was used to build a ruleset for composer classification (Cohen, 1995). RIPPER uses sequential covering to generate the ruleset. It consists of a building stage and an optimization stage. The building stage starts by growing one rule by greedily adding antecedents (or conditions) to the rule, until it is perfect (i.e., 100% accurate) based on an initial growing and pruning set (ratio 2:1). The algorithm tries every possible value for each attribute and selects the condition with the highest information gain. Then each condition is pruned in last-to-first order. This is repeated until either there are no more positive examples, or the description length (DL) is 64 bits greater than the smallest DL found so far, or the error rate is $>50\%$. In the optimization phase, the instances covered by existing rules are removed from the pruning set. Based on this new pruning set, each rule is reconsidered and two variants are produced. If one of the variants offers a better description length, it replaces the rule. This process is repeated (Hall et al., 2009). The models below were created with 50 optimizations.

We again built three models (R1, R2 and R3), each with varying levels of complexity. This was achieved by varying the minimum total weight of the instances in a rule (minNo). Setting a higher level for this parameter forces the algorithm to have more instances for each rule, thus reducing the total number of rules in the model. The settings and performance results of the models are summarized in Table 14.5.

The first model (R1) was created by setting the minimum total weight of the instances in a rule very high. This resulted in an easy-to-understand ruleset (see Fig. 14.4). As can be seen in Table 14.5, its performance is slightly lower than the other models, yet comparable to the performance of the decision trees, described in the previous section.

Table 14.4 Confusion matrix for C4.5 (model T3)

a	b	c	classified as
175	39	40	a = HA
66	**110**	20	b = BE
24	21	**550**	c = BA

Table 14.5 Performance and settings of the RIPPER rulesets (10CV)

ID	minNo	Number of rules	Accuracy	AUC
R1	50	3	75%	75%
R2	25	5	78%	80%
R3	2	12	81%	85%

```
if (Most Common Melodic Interval Prevalence) $\leq$ 0.2678 and (Melodic Octaves Frequency $\geq$ 0.03006) then
    Composer = BE
else if (Stepwise Motion Frequency $\leq$ 0.5464) and (Chromatic Motion Frequency $\geq$ 0.1726) and (0.06466 $\leq$ Most Common Pitch Prevalence $\leq$ 0.1164) then
    Composer = HA
else
    Composer = BA
end if
```

Fig. 14.4 RIPPER Ruleset (R1)

The second model (R2) was created with a higher setting for the minimum total weight of the instances in a rule. This resulted in the five “if-then” rules shown in Fig. 14.5. The model is slightly more difficult to understand, but it also performed better on the dataset used here (see Table 14.5).

A final model (R3) was created by setting minNo very low. The resulting model, consisting of 12 rules, is not shown here as it is too extensive. However, with an accuracy of 81% and AUC of 87% it does outperform the previous two models. The

```
if (Most Common Melodic Interval Prevalence) $\leq$ 0.2688 and (Melodic Octaves Frequency $\geq$ 0.04438) then
    Composer = BE
else if (Most Common Pitch Prevalence $\leq$ 0.07191) and (Most Common Melodic Interval Prevalence $\leq$ 0.2956) and (Melodic Octaves Frequency $\geq$ 0.02489) then
    Composer = BE
else if (Most Common melodic Interval Prevalence $\leq$ 0.328) and (Melodic Thirds Frequency $\geq$ 0.1119) and (Chromatic Motion Frequency $\geq$ 0.1692) and (Most Common Pitch Prevalence $\leq$ 0.106) then
    Composer = HA
else if (Stepwise Motion Frequency $\leq$ 0.5245) and (Chromatic Motion Frequency $\geq$ 0.1166) and (Repeated Notes Frequency $\geq$ 0.1972) then
    Composer = HA
else
    Composer = BA
end if
```

Fig. 14.5 RIPPER Ruleset (R2)

Table 14.6 Confusion matrix for RIPPER

a	b	c	classified as
189	32	33	a = HA
48	**124**	24	b = BE
37	21	**537**	c = BA

confusion matrix of the model is shown in Table 14.6. The misclassification rates are very similar to those of the decision trees in the previous section, with fewest misclassifications occurring between Beethoven and Bach.

It is noticeable that the first feature evaluated by the rulesets is the same as the root feature of the decision trees in Sect. 14.4.1 above, which confirms its importance. The rulesets and decision trees can be interpreted in a similar way. Like the decision trees, the rulesets suggest that Beethoven uses more melodic octaves than the other composers and that Haydn uses more repeated notes than Bach.

14.4.3 Logistic Regression

In the previous sections, two techniques were explored to build comprehensible models. Both trees and rulesets provide crisp classification. This means that they classify a musical piece as being either by a certain composer or not. The logistic regression model built in this section offers a continuous measure that indicates the probability that a piece is by each composer under consideration. These models are built for each composer and the one with the highest probability is chosen as the predicted class. Just like the previously discussed models, implementing logistic regression models is computationally efficient. They are also less prone to overfitting than other models such as neural networks (Tu, 1996).

A logistic regression model was built with Weka's SimpleLogistic function. This implementation uses LogitBoost, an algorithm that performs additive logistic regression (Witten and Frank, 2005). LogitBoost sequentially applies a simple regression function to re-weighted versions of the training data. The optimal number of LogitBoost iterations to perform is cross-validated, which leads to automatic attribute selection (Landwehr et al., 2005). This simple boosting strategy often results in dramatic performance improvements (Friedman et al., 2000).

The resulting model that we obtained outputs the probability, $P(L_y)$, that a piece is by a certain composer, y. $P(L_y)$ is defined as follows:

$$P(L_y) = \frac{1}{1+e^{-L_y}} \, , \tag{14.1}$$

where

$$\begin{aligned} L_{HA} = -3.39 + 21.19 \cdot x_1 + 3.96 \cdot x_2 + 6.22 \cdot x_3 + 6.29 \cdot x_4 - 4.9 \cdot x_5 \\ - 1.39 \cdot x_6 + 3.29 \cdot x_7 - 0.17 \cdot x_8 + 0 \cdot x_9 \\ - 0.72 \cdot x_{10} + 8.35 \cdot x_{11} - 4.21 \cdot x_{12} \,, \end{aligned} \tag{14.2}$$

$$\begin{aligned} L_{BE} = 6.19 + 5.44 \cdot x_1 + 14.69 \cdot x_2 + 24.36 \cdot x_3 - 0.45 \cdot x_4 - 6.52 \cdot x_5 \\ - 29.99 \cdot x_6 + 3.84 \cdot x_7 - 0.38 \cdot x_8 - 3.39 \cdot x_9 \\ - 2.76 \cdot x_{10} + 2.04 \cdot x_{11} - 0.48 \cdot x_{12} \,, \end{aligned} \tag{14.3}$$

$$\begin{aligned} L_{BA} = -4.88 - 13.15 \cdot x_1 - 6.16 \cdot x_2 - 5.28 \cdot x_3 - 11.63 \cdot x_4 + 11.92 \cdot x_5 \\ + 34 \cdot x_6 - 13.21 \cdot x_7 + 3.1 \cdot x_8 + 2.37 \cdot x_9 \\ + 0.66 \cdot x_{10} - 5.05 \cdot x_{11} + 3.03 \cdot x_{12} \,. \end{aligned} \tag{14.4}$$

In these expressions, x_i refers to the corresponding feature value from Table 14.1.

This type of continuous output score allows it to be included in an evaluation metric used by a music generation algorithm. In a previous study, the authors used a local search heuristic to generate music intended to have the characteristics of a certain composer. The amount of influence of a certain composer contained within a certain piece was measured by the probability of a logistic regression model (Herremans et al., 2015).

The logistic regression equations are not as straightforward to interpret as the previous two models. Yet they still offer a lot of information about the differences between the styles of the composers. When a feature has a high coefficient, it means that it is important for distinguishing a particular composer from other composers. For instance, x_5 ('Most Common Melodic Interval Frequency') has a high coefficient value, especially for BA. When looking at the previous models, this feature is also at the top of the decision trees (see Figs. 14.2 and 14.3) and occurs in almost all of the rules in the rulesets (see Figs. 14.4 and 14.5).

The logistic regression model that we obtained outperforms the two previous models with an AUC of 92% and accuracy of 83% and is the second best model overall (see Table 14.2). The confusion matrix, shown in Table 14.7, reflects this higher accuracy rate. When examining the misclassified pieces, we notice that their average probability is 64%. Examples of misclassified pieces include the fourth movement of Beethoven's String Quartet No. 9 in C major, Op. 59, No. 3 (Allegro molto), which is classified as Haydn with a probability of 4% and the first movement of Bach's Brandenburg Concerto No. 5 in D major (BWV 1050), which is classified as Haydn with a probability of 37%.

Table 14.7 Confusion matrix for logistic regression

a	b	c	classified as
190	30	34	a = HA
57	**119**	20	b = BE
25	15	**555**	c = BA

14.4.4 Naive Bayes

We also used Weka to build a naive Bayes classifier. Like the logistic regression model, a naive Bayes model outputs the probability that a piece is by a certain composer. This probability estimate is based on the assumption that the features are conditionally independent. Given class label (i.e., composer) y, this independence assumption can be represented as follows (Tan et al., 2007):

$$P(\mathbf{x} \mid Y = y) = \prod_{j=1}^{M} P(x_j \mid Y = y) \,, \tag{14.5}$$

where each attribute set $\mathbf{x} = \{x_1, x_2, \ldots, x_M\}$ consists of M attributes.

Because of the independence assumption, we do not need to calculate the class-conditional probability for every combination of $\mathbf{x}$. Only the conditional probability of each x_i given that $Y = y$ has to be estimated. This offers a practical advantage, since a good estimate of the probability can be obtained without the need for a very large training set. Given a test piece, the posterior probability for each composer Y can be calculated by the following formula (Lewis, 1998):

$$P(Y \mid \mathbf{x}) = \frac{P(Y) \cdot \prod_{j=1}^{M} P(x_j \mid Y)}{P(\mathbf{x})} \,. \tag{14.6}$$

Since the attributes are continuous, a normal distribution is often chosen to represent the class-conditional probability. However, we found that better performance was achieved by using a kernel estimator instead of a normal distribution (John and Langley, 1995). However, unlike the previous models, the model produced is too extensive to show in this chapter and is not easily comprehensible. Its results are included as a benchmark value for the other models. The resulting model has an accuracy of 80% and an AUC value of 90%, which is less good than the logistic regression model described in the previous section. The confusion matrix is shown in Table 14.8. As can be seen in this table, many pieces by Haydn are misclassified as Beethoven. Other than that, the misclassification errors are comparable to those for the previous models (cf. Figs. 14.4, 14.6 and 14.7), with least confusion between Beethoven and Bach.

Table 14.8 Confusion matrix for naive Bayes

a	b	c	classified as
199	28	27	a = HA
69	**118**	9	b = BE
41	34	**520**	c = BA

14.4.5 Support Vector Machine

In order to provide a benchmark for the performance of the comprehensible models presented above, a support vector machine classifier was implemented. Support vector machines (SVMs) are black-box models, yet they outperform more traditional models in many areas including stock market prediction (Huang et al., 2005), text classification (Tong and Koller, 2002), Celtic violin performer identification (Ramirez et al., 2011), gene selection (Guyon et al., 2002) and others.

In this section, the library LibSVM (Chang and Lin, 2011) was used to build a support vector machine (SVM) classifier. This is a learning procedure based on statistical learning theory (Vapnik, 1995). Given a training set of N data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where the features $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{-1,+1\}$, the SVM classifier should fulfil the following conditions (Cristianini and Shawe-Taylor, 2000; Vapnik, 1995):

$$\begin{cases} \mathbf{w}^T \varphi(\mathbf{x}_i) + b \geq +1, & \text{if } y_i = +1 \\ \mathbf{w}^T \varphi(\mathbf{x}_i) + b \leq -1, & \text{if } y_i = -1 \end{cases} \tag{14.7}$$

which is equivalent to

$$y_i \times (\mathbf{w}^T \varphi(\mathbf{x}_i) + b) \geq 1, \qquad i = 1, ..., N \, . \tag{14.8}$$

The input space is mapped to a high (possibly infinite) dimensional feature space by the non-linear function $\varphi(\cdot)$. In this new feature space, the above inequalities construct a hyperplane $\mathbf{w}^T \varphi(\mathbf{x}) + b = 0$ discriminating between the two classes. The margin between the two classes is maximized by minimizing $\mathbf{w}^T \mathbf{w}$. Describing the inner workings of the SVM falls beyond the scope of this chapter. The interested reader is referred to Cristianini and Shawe-Taylor (2000), who describe the optimization problem that results in the following formula for the actual classifier:

$$y(\mathbf{x}) = \text{signum}(\textstyle\sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b) \, , \tag{14.9}$$

where $K(\mathbf{x}_i, \mathbf{x}) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x})$ is taken with a positive definite kernel satisfying the Mercer theorem and α_i are the Lagrange multipliers, determined by optimizing the dual problem. Here, the Radial Basis Function (RBF) kernel was used to map the feature space to a hyperplane:

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\{-\|\mathbf{x} - \mathbf{x}_i\|^2 / \sigma^2\}, \text{ (RBF kernel)}$$

where σ is a constant.

The GridSearch procedure in Weka was used to determine the optimal settings for the regularization parameter (see Cristianini and Shawe-Taylor (2000)) of the optimization problem and the σ for the RBF kernel (Weka, 2013).

Trying to comprehend the logic of the classifications made is quite difficult, if not impossible since the SVM classifier with non-linear kernel is a complex, nonlinear function (Martens and Provost, 2014; Martens et al., 2009). It does however

Table 14.9 Confusion matrix for support vector machines

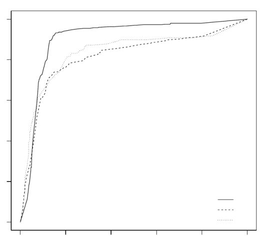
a	b	c	classified as
204	26	24	a = HA
49	**127**	20	b = BE
22	10	**563**	c = BA

outperform the previous models. The resulting accuracy is 86% and the AUC-value is 93% for the SVM with RBF kernel (see Table 14.2). However, when testing for the difference in AUC performance between SVM and other models, the p-value remained > 0.01 for both logistic regression and naive Bayes. This indicates that these two models closely match the performance of the SVM. The confusion matrix (see Table 14.9) confirms that, of the types of model that we have considered, SVM provides the best model for discriminating between Haydn, Beethoven and Bach. Most misclassification occurs between Haydn and Beethoven, which might be correlated with the geographical and temporal overlap between the lives of these composers as mentioned in Sect. 14.4.1. When examining the misclassified pieces, they all seem to have a very low probability, with an average of 39%. Examples of misclassified pieces are the first movement of Haydn's String Quartet in C major, Op. 74, No. 1, which is misclassified as Bach with 38% probability; and the theme from Beethoven's Six Variations on a Swiss Song, WO 64, which is misclassified as Bach with a probability of 38%.

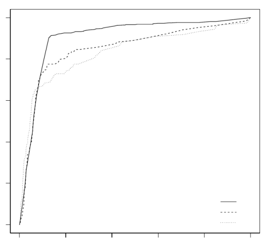
14.5 Summary of Results

The receiver operating characteristic (ROC) of the most accurate model for each of the different classifiers is shown in Fig. 14.6. The ROC curve displays the trade-off between true positive rate (TPR) and false negative rate (FNR). The logistic regression and the SVM classifiers score best, which is confirmed by their high AUC value in Table 14.2. The SVM classifier achieves the highest AUC value. Yet when testing for the difference in AUC performance between SVM and other models, the p-value remains > 0.01 for logistic regression, which makes this the best-performing comprehensible model. Although trees and rulesets are more intuitive to understand, their performance is slightly lower, which is reflected in their ROC curves.

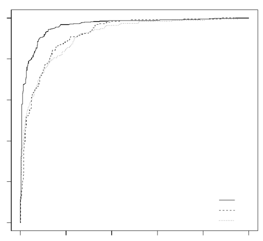
All models clearly score better than a random classification, which is represented by the diagonal through the origin. While the SVM significantly outperforms the other models (except the AUC of logistic regression and naive Bayes), they can still be used to get a better understanding of the style characteristics of the three composers.



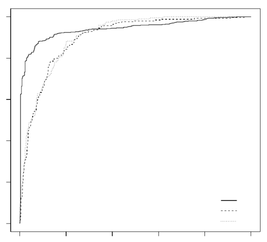
(a) ROC decision tree



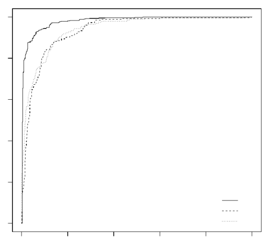
(b) ROC ruleset



(c) ROC logistic regression



(d) ROC naive Bayes



(e) ROC support vector machine

Fig. 14.6 ROC curves of the models

14.6 Conclusions

In this study, a number of global musical features were extracted from a large database of music consisting of pieces by three composers (Bach, Beethoven and Haydn). Based on these features, five types of classification model were built. The first three models are more comprehensible and thus offer more insight and understanding into the characteristics of each composer's style and the differences between them. The latter two models serve as a performance benchmark as they are too complex or extensive to be easily understood. While black-box models (SVM) have the highest performance (AUC 93% and accuracy 86%), comprehensible models such as the RIPPER ruleset still perform well (AUC 85% and accuracy 81%).

The comprehensible models give us musical insights and can suggest directions for further musicological research. For example, the results of this study suggest that Beethoven typically does not focus on using one particular interval, in contrast to Haydn or Bach, who have a higher prevalence of the most common melodic interval. Clearly, this result is based on a limited corpus and cannot be generalized without further investigation.

It would be interesting to examine whether an approach based on local features could contribute to gaining even more insight into the styles of composers. Classification models with an even higher accuracy rate might also be developed. It could be interesting to extract comprehensible rules from SVM models with rule extraction techniques such as Trepan and G-REX (Martens et al., 2007). This might provide more accurate comprehensible models. According to Backer and van Kranenburg (2005), the composers examined in the study reported in this chapter are relatively easy to distinguish. It would therefore be interesting to apply the methods from this chapter to distinguishing between composers whose styles are more similar, such as Bach, Telemann and Handel.

References

Backer, E. and van Kranenburg, P. (2005). On musical stylometry—a pattern recognition approach. *Pattern Recognition Letters*, 26(3):299–309.

Berenzweig, A., Logan, B., Ellis, D., and Whitman, B. (2004). A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76.

Bohak, C. and Marolt, M. (2009). Calculating similarity of folk song variants with melody-based features. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 597–602, Kobe, Japan.

Casey, M., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., and Slaney, M. (2008). Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696.

CCARH (2012). *KernScores, http://kern.ccarh.org*. Last accessed: November 2012.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Chew, E., Volk, A., and Lee, C.-Y. (2005). Dance music classification using inner metric analysis. In Golden, B. L., Raghaven, S., and Wasil, E. A., editors, *The Next Wave in Computing, Optimization, and Decision Technologies*, pages 355–370. Springer.

Cohen, W. (1995). Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA.

Conklin, D. and Witten, I. H. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73.

Cosi, P., De Poli, G., and Lauzzana, G. (1994). Auditory modelling and self-organizing neural networks for timbre classification. *Journal of New Music Research*, 23(1):71–98.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press.

DeNora, T. (1997). *Beethoven and the Construction of Genius: Musical Politics in Vienna, 1792-1803*. University of California Press.

Downie, J. (2003). Music information retrieval. *Annual Review of Information Science and Technology*, 37(1):295–340.

Eerola, T., Järvinen, T., Louhivuori, J., and Toiviainen, P. (2001). Statistical features and perceived similarity of folk melodies. *Music Perception*, 18(3):275–296.

Eerola, T. and Toiviainen, P. (2004). MIR in Matlab: The Midi Toolbox. In *Proceedings of 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 22–27, London, UK.

Fawcett, T. (2004). ROC graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Laboratories, Palo Alto, CA.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals Of Statistics*, 28(2):337–407.

Geertzen, J. and van Zaanen, M. (2008). Composer classification using grammatical inference. In *Proceedings of the International Workshop on Machine Learning and Music (MML 2008)*, pages 17–18, Helsinki, Finland.

Gheyas, I. and Smith, L. (2010). Feature subset selection in large dimensionality domains. *Pattern Recognition*, 43(1):5–13.

Ghias, A., Logan, J., Chamberlin, D., and Smith, B. (1995). Query by humming: Musical information retrieval in an audio database. In *Proceedings of the Third ACM International Conference on Multimedia*, pages 231–236, San Francisco, CA.

Greene, D. (1985). *Greene's Biographical Encyclopedia of Composers*. The Reproducing Piano Roll Foundation. Edited by Alberg M. Petrak.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. (2009). The Weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Herlands, W., Der, R., Greenberg, Y., and Levin, S. (2014). A machine learning approach to musically meaningful homogeneous style classification. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, pages 276–282, Quebec, Canada.

Herremans, D., Martens, D., and Sörensen, K. (2014). Dance hit song prediction. *Journal of New Music Research*, 43(3):291–302.

Herremans, D., Sörensen, K., and Martens, D. (2015). Classification and generation of composer-specific music using global feature models and variable neighborhood search. *Computer Music Journal*, 39(3). In press.

Hillewaere, R., Manderick, B., and Conklin, D. (2009). Global feature versus event models for folk song classification. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan.

Hillewaere, R., Manderick, B., and Conklin, D. (2010). String quartet classification with monophonic models. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, The Netherlands.

Hillewaere, R., Manderick, B., and Conklin, D. (2012). String methods for folk tune genre classification. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, pages 217–222, Porto, Portugal.

Hontanilla, M., Pérez-Sancho, C., and Iñesta, J. (2013). Modeling musical style with language models for composer recognition. In Sanchez, J. M., Micó, L., and Cardoso, J., editors, *Pattern Recognition and Image Analysis: 6th Iberian Conference, IbPRIA 2013, Funchal, Madeira, Portugal, June 5–7. 2013, Proceedings*, volume 7887 of *Lecture Notes in Computer Science*, pages 740–748. Springer.

Huang, W., Nakamori, Y., and Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513–2522.

Huron, D. (2002). Music information processing using the Humdrum Toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26.

Jesser, B. (1991). *Interaktive Melodieanalyse*. Peter Lang.

John, G. and Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in Artificial Intelligence*, pages 338–345, Montreal, Canada.

Kaliakatsos-Papakostas, M., Epitropakis, M., and Vrahatis, M. (2011). Weighted Markov chain model for musical composer identification. In Chio, C. D., Cagnoni, S., Cotta, C., Ebner, M., and et al., A. E., editors, *Applications of Evolutionary Computation*, volume 6625 of *Lecture Notes in Computer Science*, pages 334–343. Springer.

Kassler, M. (1966). Toward musical information retrieval. *Perspectives of New Music*, 4(2):59–67.

Landwehr, N., Hall, M., and Frank, E. (2005). Logistic model trees. *Machine Learning*, 59(1-2):161–205.

Laurier, C., Grivolla, J., and Herrera, P. (2008). Multimodal music mood classification using audio and lyrics. In *Seventh International Conference on Machine Learning and Applications (ICMLA'08)*, pages 688–693, La Jolla, CA.

Lewis, D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In Nedellec, C. and Rouveirol, C., editors, *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, pages 4–15. Springer.

Li, X., Ji, G., and Bilmes, J. (2006). A factored language model of quantized pitch and duration. In *International Computer Music Conference (ICMC 2006)*, pages 556–563, New Orleans, LA.

Mandel, M. and Ellis, D. (2005). Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 594–599, London, UK.

Martens, D. (2008). Building acceptable classification models for financial engineering applications. *SIGKDD Explorations*, 10(2):30–31.

Martens, D., Baesens, B., Van Gestel, T., and Vanthienen, J. (2007). Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183(3):1466–1476.

Martens, D. and Provost, F. (2014). Explaining data-driven document classifications. *MIS Quarterly*, 38(1):73–99.

Martens, D., Van Gestel, T., and Baesens, B. (2009). Decompositional rule extraction from support vector machines by active learning. *IEEE Transactions on Knowledge and Data Engineering*, 21(2):178–191.

McKay, C. and Fujinaga, I. (2004). Automatic genre classification using large high-level musical feature sets. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, Barcelona, Spain.

McKay, C. and Fujinaga, I. (2006). jSymbolic: A feature extractor for MIDI files. In *Proceedings of the International Computer Music Conference (ICMC 2006)*, pages 302–5, New Orleans, LA.

McKay, C. and Fujinaga, I. (2007). Style-independent computer-assisted exploratory analysis of large music collections. *Journal of Interdisciplinary Music Studies*, 1(1):63–85.

McKay, C. and Fujinaga, I. (2008). Combining features extracted from audio, symbolic and cultural sources. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pages 597–602, Philadelphia, PA.

McKay, C. and Fujinaga, I. (2009). jMIR: Tools for automatic music classification. In *Proceedings of the International Computer Music Conference (ICMC 2009)*, pages 65–8, Montreal, Canada.

Mearns, L., Tidhar, D., and Dixon, S. (2010). Characterisation of composer style using high-level musical features. In *Proceedings of 3rd International Workshop on Machine Learning and Music*, pages 37–40, Florence, Italy.

Mendel, A. (1969). Some preliminary attempts at computer-assisted style analysis in music. *Computers and the Humanities*, 4(1):41–52.

Moreno-Seco, F., Inesta, J., Ponce de León, P. J., and Micó, L. (2006). Comparison of classifier fusion methods for classification in pattern recognition tasks. In Yeung, D.-Y., Kwok, J. T., Roli, A. F. F., and de Ridder, D., editors, *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2006 and SPR 2006, Hong Kong, China, August 17–19, 2006. Proceedings*, volume 4109 of *LNCS*, pages 705–713. Springer.

Pearce, M., Conklin, D., and Wiggins, G. (2005). Methods for combining statistical models of music. In Kronland-Martinet, R., Voinier, T., and Ystad, S., editors, *Computer Music Modeling and Retrieval*, volume 3902 of *LNCS*, pages 295–312. Springer.

Pérez-Sancho, C., Rizo, D., and Inesta, J. M. (2008). Stochastic text models for music categorization. In da Vitoria Lobo, N. and others, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 5342 of *LNCS*, pages 55–64. Springer.

Pfeiffer, S., Fischer, S., and Effelsberg, W. (1997). Automatic audio content analysis. In *Proceedings of the 4th ACM International Conference on Multimedia*, pages 21–30, Boston, MA.

Pollastri, E. and Simoncelli, G. (2001). Classification of melodies by composer with hidden Markov models. In *Web Delivering of Music, 2001. Proceedings. First International Conference on*, pages 88–95. IEEE.

Ponce de León, P. J. and Iñesta, J. (2003). Feature-driven recognition of music styles. In Perales, F. J., Campilho, A. J. C., de la Blanca, N. P., and Sanfeliu, A., editors, *Pattern Recognition and Image Analysis: First Iberian Conference, IbPRIA 2003, Puerto de Andratx, Mallorca, Spain*, volume 2652 of *Lecture Notes in Computer Science*, pages 773–781. Springer.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*, volume 1. Morgan Kaufmann.

Ramirez, R., Maestre, E., Perez, A., and Serra, X. (2011). Automatic performer identification in Celtic violin audio recordings. *Journal of New Music Research*, 40(2):165–174.

Rosen, C. (1997). *The Classical Style: Haydn, Mozart, Beethoven*, volume 1. Norton.

Ruggieri, S. (2002). Efficient C4. 5 [classification algorithm]. *Knowledge and Data Engineering, IEEE Transactions on*, 14(2):438–444.

Sapp, C. (2005). Online database of scores in the Humdrum file format. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 664–665, London, UK.

Shmueli, G. and Koppius, O. (2011). Predictive analytics in information systems research. *MIS Quarterly*, 35(3):553–572.

Steinbeck, W. (1982). Struktur und Ähnlichkeit. In *Methoden automatisierter Melodieanalyse*. Bärenreiter.

Stenzel, U. Lima, M. and Downes, J. . (2012). Study on Digital Content Products in the EU, Framework contract: Evaluation impact assessment and related services; Lot 2: Consumer's Policy. Technical report, EU, Brussels.

Tan, P. et al. (2007). *Introduction to Data Mining*. Pearson Education.

Tong, S. and Koller, D. (2002). Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66.

Tseng, Y.-H. (1999). Content-based retrieval for music collections. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 176–182.

Tu, J. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, 49(11):1225–1231.

Typke, R., Wiering, F., and Veltkamp, R. (2005). A survey of music information retrieval systems. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 153–160, London, UK.

Tzanetakis, G. and Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302.

Tzanetakis, G., Ermolinskyi, A., and Cook, P. (2003). Pitch histograms in audio and symbolic music information retrieval. *Journal of New Music Research*, 32(2):143–152.

van Kranenburg, P. (2008). On measuring musical style—The case of some disputed organ fugues in the J. S. Bach (BWV) catalogue. *Computing in Musicology*, 15:120–137.

van Kranenburg, P. (2010). *A computational approach to content-based retrieval of folk song melodies*. PhD thesis, Utrecht University.

van Kranenburg, P. and Backer, E. (2004). Musical style recognition—A quantitative approach. In *Proceedings of the Conference on Interdisciplinary Musicology (CIM04)*, pages 106–107, Graz, Austria.

van Kranenburg, P., Volk, A., and Wiering, F. (2013). A comparison between global and local features for computational classification of folk song melodies. *Journal of New Music Research*, 42(1):1–18.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer.

Volk, A. and van Kranenburg, P. (2012). Melodic similarity among folk songs: An annotation study on similarity-based categorization in music. *Musicae Scientiae*, 16(3):317–339.

Weihs, C., Ligges, U., Mörchen, F., and Müllensiefen, D. (2007). Classification in music research. *Advances in Data Analysis and Classification*, 1(3):255–291.

Weka (2013). Weka documentation, class GridSearch. Last accessed: October 2014.

Whitman, B., Flake, G., and Lawrence, S. (2001). Artist detection in music with Minnowmatch. In *Neural Networks for Signal Processing XI, 2001. Proceedings of the 2001 IEEE Signal Processing Society Workshop*, pages 559–568. IEEE.

Whitman, B. and Smaragdis, P. (2002). Combining musical and cultural features for intelligent style detection. In *Proceedings of the 3rd International Symposium on Music Information Retrieval (ISMIR 2002)*, pages 47–52, Paris, France.

Witten, I. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

Wold, E., Blum, T., Keislar, D., and Wheaten, J. (1996). Content-based classification, search, and retrieval of audio. *MultiMedia, IEEE*, 3(3):27–36.

Wołkowicz, J., Kulka, Z., and Keselj, V. (2007). N-gram-based approach to composer recognition. Master's thesis, Warsaw University of Technology.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., et al. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37.