Dorien Herremans,* Kenneth Sörensen,* and David Martens†
*University of Antwerp Operations Research Group ANT/OR
†Applied Data Mining Research Group
*†University of Antwerp
Prinsstraat 13, 2000 Antwerp, Belgium
{dorien.herremans, kenneth.sorensen, david.martens}@uantwerpen.be

Classification and Generation of Composer-Specific Music Using Global Feature Models and Variable Neighborhood Search

Abstract: In this article a number of musical features are extracted from a large musical database and these were subsequently used to build four composer-classification models. The first two models, an if–then rule set and a decision tree, result in an understanding of stylistic differences between Bach, Haydn, and Beethoven. The other two models, a logistic regression model and a support vector machine classifier, are more accurate. The probability of a piece being composed by a certain composer given by the logistic regression model is integrated into the objective function of a previously developed variable neighborhood search algorithm that can generate counterpoint. The result is a system that can generate an endless stream of contrapuntal music with composer-specific characteristics that sounds pleasing to the ear. This system is implemented as an Android app called FuX.

The task of recognizing a composer by listening to a musical fragment used to be reserved for those well-versed in music theory. The question that is tackled in our research is, “Can a computer accurately recognize who composed a musical piece?” We take a data-driven approach, by scanning a large database of existing music and developing four classification models that can accurately classify a musical piece in groups of three composers. This research builds predictive classification models that can be used both for theory building and to calculate the probability that a piece is composed by a certain composer.

The first goal of this article is to build a rule set and a decision tree that gives the reader an understanding of the differences between styles of composers (J. S. Bach, Joseph Haydn, and Ludwig van Beethoven). These models give the reader more insight into why a piece belongs to a certain composer. The second goal is to build more accurate classification models that can help an existing music composition algorithm generate composer-specific music, i.e., music that contains characteristics of a specific composer. In previous papers, we developed a variable neighborhood search (VNS) algorithm that can compose contrapuntal music (Herremans and Sörensen 2012, 2013a). The logistic regression model developed in this article is incorporated into the objective function of the VNS. The resulting system is able to play a stream of continuously generated contrapuntal music with composer-specific traits.

Computer Music Journal, 39:3, pp. 71–91, Fall 2015
doi:10.1162/COMJ_a_00316
© 2015 Massachusetts Institute of Technology.

Prior Work

The digitization of the music industry has attracted growing attention to the field of music information retrieval (MIR). This is a multidisciplinary domain, concerned with retrieving and analyzing multifaceted information from large music databases (Downie 2003). According to Byrd and Crawford (2002), the earliest publication on MIR dates back to the mid 1960s (Kassler 1966). Michael Kassler used the term MIR to name the programming language he developed for extracting information from music files. In recent years, numerous MIR systems have been developed and applied to a broad range of topics. An interesting example is the content-based music search engine Query by Humming (Ghias et al. 1995). This MIR system allows the user to find a song based on a tune that he or she hums. Another application of MIR is measuring the similarity between two musical pieces (Berenzweig et al. 2004). In our

Table 1. Data Set Used in This Study

Composer	*# Instances*
Haydn (HA)	254
Beethoven (BE)	196
Bach (BA)	595

research, the focus lies on using MIR for composer classification.

When it comes to automatic music classification, machine-learning tools are used to classify musical pieces by genre (Tzanetakis and Cook 2002; Conklin 2013), cultural origin (Whitman and Smaragdis 2002), mood (Laurier, Grivolla, and Herrera 2008), hit ranking (Herremans, Martens, and Sörensen 2014), etc. The general task of automatically classifying music by genre has received a great deal of attention; see Conklin (2013) for a more complete overview. The more specific task of composer classification has remained largely unexplored in the past (Geertzen and van Zaanen 2008), yet it has gained more attention in the last decade.

The studies we describe herein usually list the accuracy rates as a performance measure. It should, however, be noted that accuracy is not always the best performance measure—for instance in the case of an unbalanced data set. The receiver operating characteristic (ROC, see Fawcett 2003) offers a more correct measure and is therefore used to evaluate the performance of the models in this research. This metric takes into account the true positives versus the false positives, which makes it more suited when the data set is slightly skewed, as in Table 1. When evaluating the models listed, one should take into account that accuracy is not always comparable, depending on the characteristics of the data set.

In 1958, Joseph Youngblood was one of the first to tackle the composer classification problem with measures from information theory (Youngblood 1958). He manually studied features such as entropy, tonal frequencies, and transition probabilities for pieces by Franz Schubert, Felix Mendelssohn, and Robert Schumann. These days, computers offer researchers the power to create classification models that are both more accurate and more complex. A system to classify string quartet pieces by composer has been implemented by Kaliakatsos-Papakostas, Epitropakis, and Vrahatis (2011). In that system, the four voices of the quartets are treated as a monophonic melody, so that it can be represented through a discrete Markov chain. The weighted Markov chain model reaches a classification success of 59% to 88% in classifying between two composers. The hidden Markov models designed by Pollastri and Simoncelli (2001) for the classification of 605 monophonic themes by five composers has a lower accuracy rate. Their best result has an accuracy of 42% of successful classifications, on average. It must be noted, however, that this accuracy is not measured for classification between two classes as in the previous example, but classification is done with five classes or composers. The accuracies given in this section should not be treated as the final word, because they depend greatly on the chosen data set (a balanced set; how many and which composers, etc.).

Wołkowicz, Kulka, and Kešelj (2008) show that another machine-learning technique, called "*n*-grams," can be used to classify piano files into groups of five composers. An *n*-gram model tries to find patterns in properties of the training data. These patterns are called *n*-grams, in which *n* is the number of symbols in a pattern. Hillewaere, Manderick, and Conklin (2010) also use *n*-grams and global feature models to classify string quartets for two composers (Haydn and Mozart). Their trigram approach to composer recognition of string quartets has a classification accuracy of 61.4% for violin and viola, and 75.4% for cello.

The *n*-gram method belongs to the family of grammars, a group of techniques that use a rule-based approach to specify patterns (Searls 2002). Giuseppe Buzzanca states that the use of grammars such as *n*-grams for modeling music style is unsatisfying because they are vulnerable with regard to creating ad hoc rules and they cannot represent ambiguity in the musical process (Buzzanca 2002). He worked with recognition of the style of Giovanni Pierluigi da Palestrina, which could be considered a more general problem than composer recognition. Instead of *n*-grams, he implemented a neural network that can classify with 97% accuracy on the test set. It should be noted, however, that all the pieces of the music database are heavily preprocessed and classification is only done on short "main themes." One of the disadvantages of neural networks is that these models are in essence a black box, as they provide a complex, nonlinear output rating. As they stand, they do not give any new music-theoretical insights into the differences between two composers. Manaris and co-workers (2005) use artificial neural networks to distinguish five composers from various genres with 93.6% to 95% accuracy. Their model is based on 20 simple global metrics based on Zipf's law (Zipf 1935). To generate a comprehensible model, rules could be extracted from an existing black-box neural network, using pedagogical rule-extraction techniques like Trepan and G-REX (see Martens et al. 2007).

Van Kranenburg and Backer (2004) apply other types of machine-learning algorithms to a database of 320 pieces from the 18th and early 19th century. Twenty high-level style markers based on properties of counterpoint are examined. The *k*-means clustering algorithm they developed shows that musical pieces of the chosen five composers do form a cluster in feature space. A decision tree (C4.5) and a nearest-neighbor classification algorithm show that it is possible to classify pieces with a fairly low error rate. Although the features are described in the paper, a detailed description of the models is missing.

Mearns, Tidhar, and Dixon (2010) also use high-level features based on counterpoint and intervallic features to classify similar musical pieces. The C4.5 decision tree and naive Bayes models that they developed correctly classified 44 out of 66 pieces with seven groups of composers. Although the actual decision tree is not displayed in the paper, it could give music theorists an insight into the differences between styles of composers. The system developed by Dor and Reich (2011) uses a number of global characteristics to build C4.5, naive Bayes, random forest, simple logistic, support vector machines, and RIPPER classifiers. The system reaches an accuracy of 75% when classifying keyboard scores between Mozart and Haydn. For pairs of composers that are more easy to differentiate, such as Mozart and Joplin, the algorithm reaches accuracies as high as 99%.

In the next sections, a technique is described to extract useful musical features from a database. These features are then used to build four accurate classification models. In contrast to many existing studies, the models described in this research are both accurate as well as comprehensible and the full details are described in this article. The developed models give insights into the styles of Haydn, Beethoven, and Bach. In a next phase, one of the models is incorporated into a previously developed VNS algorithm that can compose music (Herremans and Sörensen 2013a), which results in a system that is able to generate music that has characteristics of a specified composer.

Feature Extraction

Traditionally, a distinction is made between *symbolic MIR* and *audio MIR*. Symbolic music representations, such as MIDI, contain very high-level structured information about music, e.g., which note is played by which instrument. Most existing work, however, revolves around audio MIR, in which automatic audition techniques are used to extract relevant information from audio signals (Tzanetakis, Ermolinskyi, and Cook 2003). Different features can be examined, depending on the type of file that is being analyzed. These features can be roughly classified into three groups:

1. *low-level features* extracted by automatic audition techniques from audio signals such as WAV files; e.g., spectral flux and zero-crossing rate (Tzanetakis, Ermolinskyi, and Cook 2003)
2. *high-level features* extracted from structured files such as MIDI; e.g., interval frequencies, instrument presence, and number of voices (McKay and Fujinaga 2006)
3. *metadata* such as factual and cultural information information related to a file, which can be either structured or unstructured; e.g., play list co-occurrence (Casey et al. 2008)

It is not a simple task to extract note information from audio recordings of polyphonic music (Gómez Gutiérrez 2006). Because the high-level features used in this research require detailed note information, we chose to work with MIDI files. Symbolic files such as MIDI files are more comparable to musical scores. They describe the start, duration, volume, and instrument of each note in a musical fragment, and therefore allow the extraction of characteristics that might provide meaningful insights to music theorists. It must be noted that MIDI files do not capture the full richness of a musical performance in the way audio files do (Lippincott 2002). They are, however, very suitable for the features analyzed in this research.

KernScores Database

The KernScores database is a large collection of virtual musical scores made available by the Center for Computer Assisted Research in the Humanities at Stanford University (CCARH). It holds a total of 7,866,496 notes and is available online at kern.ccarh.org. This database was specifically created for computational analysis of musical scores (Sapp 2005). Bach, Beethoven, and Haydn were selected for inclusion in our classification models because a large number of musical works is available for these three composers in the KernScores database. Having a large number of instances available for each composer allows the creation of models that are more accurate. In sum, 1,045 musical pieces from a total of three composers were downloaded from the database. Almost all available musical pieces for each composer were selected, except for a few very short fragments. An overview of the selected pieces was given in Table 1.

The KernScores database contains musical pieces in **KERN notation, in ABC notation, and as MIDI files. For this research, the MIDI files were used because they are compatible with the feature-extraction software jSymbolic. This software is part of jMIR, a toolbox designed for automatic music classification (McKay and Fujinaga 2009). Van Kranenburg and Backer (2004) point out that MIDI files are typically the representation of a performance and are therefore not always an accurate representation of the score. It is true that MIDI files are often recorded by a human playing the score, which results in inaccurate timing. Because the KernScore database is instead encoded by hand from **KERN files, it offers a reliable source of accurate MIDI files.

Table 2. Features Analyzed

Variable	*Feature Description*
x_1	Chromatic motion frequency: fraction of melodic intervals corresponding to a semitone
x_2	Frequency of melodic fifths
x_3	Frequency of melodic octaves
x_4	Frequency of melodic thirds
x_5	Most common melodic interval prevalence
x_6	Most common pitch prevalence
x_7	Most common pitch class prevalence
x_8	Relative strength of most common intervals: fraction of intervals belonging to the most common and second most common melodic intervals
x_9	Relative strength of top pitch classes
x_{10}	Relative strength of top pitches
x_{11}	Repeated notes: fraction of notes that are repeated melodically
x_{12}	Stepwise motion frequency

Pitch refers to an absolute pitch, e.g., C in the 7th octave. Pitch class refers to a note without the octave, e.g., C.

Implementation of Feature Extraction

The software used to extract the features is jSymbolic, a Java-based open-source software package that allows easy extraction of high-level features from MIDI files (McKay and Fujinaga 2007). Twelve features are extracted from our data set (see Table 2). All of these features offer information regarding melodic intervals and pitches. They are measured as occurrence frequencies normalized to the range from 0 to 1.

The feature set we examined was deliberately kept small to avoid overfitting the model (Gheyas and Smith 2010). McKay and Fujinaga (2006) refer to the "curse of dimensionality," whereby the number of labeled training and testing samples needed increases exponentially with the number of features. Not having too many features allows a thorough testing of the model with limited instances and can thus improve the quality of the classification model (McKay and Fujinaga 2006). In this research, a selection of features was made from the 111 features available in jSymbolic. During this selection process, one-dimensional features that calculated frequency information related to intervals or pitches were preferred because of their normalized nature and ease of handling. All features dependent upon the key of the piece, as well as nominal features,were omitted. Features related to instruments, such as "electric guitar fraction," were omitted because they are not relevant for the chosen corpus. Rhythmic features were not used because the music-generation algorithm currently does not make changes in the rhythm. Because evaluating how much of a particular composer's influence a generated piece has is one of the goals of the developed models, these features were not included. The resulting twelve features are displayed in Table 2. Some preliminary experiments were performed with automatic feature selection on this limited data set, but they did not yield any improvements to the results provided in this article.

The jSymbolic software stores the extracted features of all instances in autonomous classification engine XML files, a format developed by McKay and colleagues (2005). We use the jMIRUtilities, another tool from the jMIR toolbox (McKay and Fujinaga 2009) to convert these XML files to the attribute-relation file format used in the Waikato Environment for Knowledge Analysis (WEKA). In the next sections, four classification models are developed based on the extracted data.

Composer Classification Models

Shmueli and Koppius (2011) point out that not only can predictive models be used as practically useful classification models, but they can also play a role in building and testing theories. In the research described in this article, models were built for two different purposes, reflecting this distinction.

Our first objective is to develop a model from which insight can be gained into the characteristics of musical pieces composed by a certain composer. This resulted in a rule set built with an algorithm called "repeated incremental pruning to produce error reduction" (RIPPER) and with a C4.5 decision tree. The second objective is to build predictive models (logistic regression and support vector machines) that can accurately determine the probability that a musical piece is in the style of a given composer. One of these models is then incorporated into the existing objective function of the music-generation algorithm, leading to a new metric that allows the VNS algorithm to automatically assess how well a generated musical composition fits into a certain composer's style. Because of the need to accurately modify the developed model for inclusion in the objective function, not all classification models are suitable. A logistic regression model was chosen for this purpose. Its implementation is described in the section "Generating Composer-Specific Music."

A number of machine-learning methods, such as neural networks, Markov chains, and clustering, have already been implemented for modeling musical style (Dubnov et al. 2003). Because most of the research papers give neither a detailed description of the model nor a full feature list, the existing research could only be used as inspiration for the models we developed.

Based on the extracted features discussed in the previous section, four supervised learning algorithms are applied to the data set. Our data set includes labeled instances, so supervised learning techniques can be used to learn a classification model based on these labeled training instances. The open-source software WEKA is used to create the classification models (Witten and Frank 2005). WEKA offers a toolbox and framework for machine learning and data mining that is recognized as a landmark system in this field (Hall et al. 2009).

This section describes four classifier models, which were developed with RIPPER, C4.5, logistic regression, and support vector machines. The first two models are of a more linguistic nature and therefore more comprehensible (Martens et al. 2011). The other two models are not as comprehensible, but have better performance. One of these latter

Figure 1. The if–then rule set used for identifying composers. BA = J.S. Bach; BE = Ludwig van Beethoven, and HA = Joseph Haydn.

```
if (Most Common Melodic Interval Prevalence) ≤ 0.2688
        and (Melodic Octaves Frequency ≥ 0.06399) then
    Composer = BE
else if (Most Common Melodic Interval Prevalence ≤ 0.2823)
        and (Most Common Pitch Prevalence ≤ 0.07051)
        and (Melodic Octaves Frequency ≥ 0.02489)
        and (Relative Strength of Top Pitches ≤ 0.9754) then
    Composer = BE
else if (Most Common Melodic Interval Prevalence ≤ 0.328)
        and (Repeated Notes Frequency ≥ 0.07592)
        and (Most Common Pitch Prevalence ≤ 0.1076) then
    Composer = HA
else if (Stepwise Motion Frequency ≤ 0.5732)
        and (Chromatic Motion Frequency ≥ 0.1166)
        and (Repeated Notes Frequency ≥ 0.3007) then
    Composer = HA
else
    Composer = BA
end if
```

Table 3. Model Evaluation with Tenfold Cross Validation

Method	*Accuracy (%)*	*WAUC (%)*
RIPPER rule set	*77* (3.82)	*82* (3.51)
C4.5 decision tree	*79* (2.97)	*88* (2.55)
Logistic regression	*83* (3.27)	94 (2.12)
Support vector machines	<u>**86**</u> (4.15)	<u>**96**</u> (2.12)

$p < 0.01$: *italic; best:* <u>**underlined bold**</u>.
Standard deviations are shown in parentheses.

models is integrated into the objective function of the music-generation algorithm, as discussed in the next section. The performance results based on accuracy and area under the curve (AUC) of all four models are displayed in Table 3. Although the distribution is not heavily skewed (see Table 1), it is not completely balanced either. Because of this, the use of the accuracy measure to evaluate our results is not suited and the weighted AUC (WAUC) per class size was used instead (cf. Fawcett 2003). Both measures are displayed in Table 3 for the sake of completeness.

Each algorithm was evaluated with stratified tenfold cross validation (10CV). During the cross-validation procedure, the data set is divided into ten folds. Nine of them are used for model building and one is used for testing. This procedure is repeated ten times. The displayed AUC and accuracy are the average results over the ten test sets. The resulting model is built on the entire data set and can be expected to have a performance that is at least as good as the 10CV performance. A Wilcoxon signed-rank test is conducted to compare the performance of each of the models with the model that performed best. The null hypothesis of this test is that there is no difference between the performance of a model and that of the best model.

RIPPER If–Then Rule Set

The advantage of using high-level musical features is that they can give useful insights into the characteristics of a composer's style. A number of techniques are available for obtaining a comprehensible model from these features. Rule sets and trees can be considered as the most easily understood classification models because of their linguistic nature (Martens 2008). Such models can be obtained by using *rule induction* and *rule extraction* techniques. The first category simply induces rules directly from the data, whereas rule extraction techniques attempt to extract rules from a trained black-box model (Martens et al. 2007). The present research focuses on using rule-induction techniques to build a rule set and a decision tree.

This section describes the use of an inductive rule-learning algorithm to learn if–then rules. Rule sets have been used in other research domains to gain insight into credit scoring (Baesens et al. 2003), medical diagnosis (Kononenko 2001), customer relationship management (Ngai, Xiu, and Chau 2009), diagnosis of technical processes (Isermann and Balle 1997), and more.

In order to build a rule set for composer classification, the propositional rule learner RIPPER was used (Cohen 1995). JRip is the WEKA implementation of RIPPER. This algorithm uses sequential covering to generate the rule set. It starts by learning one rule, removes the training instances that are covered by the rules, and then repeats this process (Hall et al. 2009).

Five rules were extracted by JRip, one for each composer, with 10CV. The rules are displayed in Figure 1. This figure shows that seven different

Table 4. Confusion Matrix for RIPPER

a	*b*	*c*	*Classified as*
145	42	67	a (Haydn)
41	**110**	45	b (Beethoven)
25	17	**553**	c (Bach)

Table 5. Detailed Results for RIPPER by Composer

Composer	*TP Rate*	*FP Rate*	*Precision*	*Recall*	*AUC (%)*
Haydn	0.57	0.08	0.687	0.57	79
Beethoven	0.56	0.07	0.65	0.56	82
Bach	0.93	0.25	0.83	0.93	86

features are used to decide if a piece is composed by Haydn, Bach, or Beethoven. The "most common melodic interval prevalence," or the occurrence frequency of the interval that is most used, is present in most of the rules. This indicates that, for instance, Beethoven typically does not focus on using one particular interval, in contrast to Haydn or Bach, for whom the prevalence of the most common melodic interval is not as restrictive.

The AUC value weighted by class size and accuracy for each technique are discussed, as well as true-positive (TP) rate, false-positive (FP) rate, recall, precision, and AUC per composer. Precision measures the accuracy, provided that a specified class has been predicted (positive predictive value). Recall is a measure of the ability of the model to correctly classify instances of a certain class (sensitivity). A high value of the F-measure indicates that both precision and recall are high (Tan, Steinbach, and Kumar 2005).

A good learning algorithm should be able to accurately predict new samples that are not in the training set. The accuracy of classification and AUC with 10CV are displayed in Table 3. The confusion matrix is displayed in Table 4. The latter table shows the least confusion occurs between Bach and Beethoven. The relatively higher misclassification rates between Haydn and Beethoven and between Haydn and Bach could be due to the fact that the data set was larger for Bach and Haydn. A second reason could be that Haydn and Beethoven's styles are indeed more similar, as suggested by the greater amount of chronological and geographical overlap between their lives, and by the fact that Haydn was once Beethoven's teacher (DeNora 1997).

While running the algorithm, the minimum number of instances in a rule was deliberately set high to obtain a smaller and thus more comprehensible tree, albeit with a slight loss of accuracy. Table 5 displays results in greater detail for each composer. It is noticeable that the recall and precision rates are much higher for Bach, which is possibly because of the greater amount of data available for Bach in the data set. Still, with 77% correctly classified and a weighted average AUC of 82%, the model developed with RIPPER is reasonably good overall.

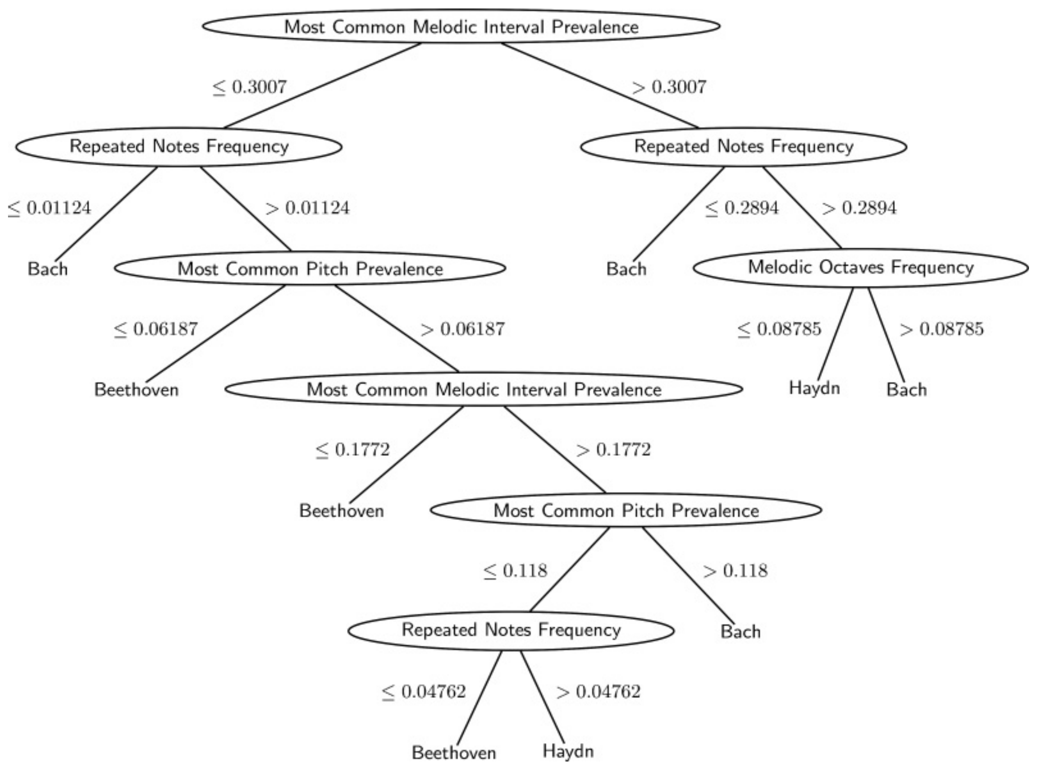
C4.5 Decision Tree

A second, tree-based model was induced to get a more visual understanding of the classification process. The J48 algorithm from WEKA (Witten and Frank 2005) is used to build a decision tree with the C4.5 algorithm (Quinlan 1993).

A decision tree is a tree data structure that consists of decision nodes and leaves. The leaves specify the class value, in this case the composer, and the nodes specify a test of one of the features. A path from the root to a leaf of the tree can be followed based on the feature values of the particular musical piece and corresponds to a predictive rule. The class at the resulting leaf indicates the predicted composer (Ruggieri 2002). Although other methods may provide classification results that are more accurate, decision trees are often useful for understanding how the classification is done.

Similar to rule sets, decision trees have been applied to a broad range of topics such as medical diagnosis (Wolberg and Mangasarian 1990), credit scoring (Hand and Henley 1997), estimation of toxic hazards (Cramer, Ford, and Hall 1976), land-cover mapping (Friedl and Brodley 1997), prediction of customer behavior changes (Kim et al. 2005), and others. Also like rule sets, one of the main advantages of a decision tree model is its comprehensibility (Craven and Shavlik 1996).

Figure 2. C4.5 decision tree.



Unlike the covering algorithm implemented in the previous model, C4.5 builds trees recursively with a "divide and conquer" approach (Quinlan 1993). This type of approach works from the top down, seeking a feature that best separates the classes, after which the tree is pruned from the leaves to the root (Wu et al. 2008).

The resulting decision tree is displayed in Figure 2. All four features from this tree model also occur in the rule set. It is noticeable that the feature evaluated at the root of the tree is the same feature that occurs in many of the rules from the if–then rule set (cf. Figure 1). The importance of the "most common melodic interval prevalence" feature for composer recognition is again confirmed, as it is the root node of the tree model. The "melodic octaves frequency" feature indicates that Bach uses more octaves than Haydn. Bach also seems to use fewer repeated notes.

Table 3 shows that the accuracy (79%) and weighted average AUC (88%) values of the tree are very comparable to those of the if–then rules extracted in the previous section. Again, the comprehensibility of the model was favored above accuracy. Therefore, the minimum number of instances per leaf was kept high. The confusion matrix (see Table 6) is also comparable, with most classification errors occurring between Haydn and Beethoven. The least confusion can be seen between Beethoven and Bach, as in the previous model. Similarly to previously discussed model, the precision and recall rates in Table 7 are much higher for Bach.

Table 6. Confusion Matrix for C4.5

a	*b*	*c*	*classified as*
174	31	49	a (Haydn)
63	**106**	27	b (Beethoven)
40	14	**541**	c (Bach)

Figure 3. *Probability that a piece is by composer* i.

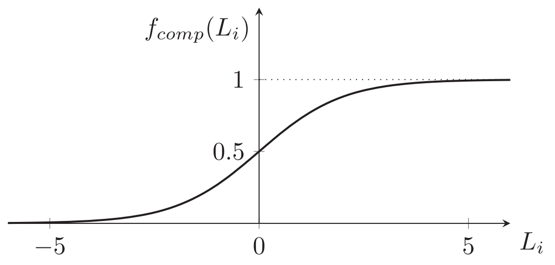


Table 7. Detailed Results for C4.5 by Composer

Composer	*TP Rate*	*FP Rate*	*Precision*	*Recall*	*AUC (%)*
Haydn	0.68	0.13	0.63	0.69	83
Beethoven	0.54	0.05	0.70	0.54	88
Bach	0.91	0.17	0.88	0.91	91

Logistic Regression

In the previous sections, two comprehensible models were described. These models provide crisp classification, which means that they determine if a musical piece is either composed by a certain composer or not. They do not offer a continuous measure that indicates "how much" of the characteristics of a certain composer are in a piece. In this section, a scoring model is developed that can accurately describe how well a musical piece fits a composer's style.

The SimpleLogistic function from the WEKA software was used to build a logistic regression (LR) model, which was fitted using LogitBoost. The LogitBoost algorithm performs additive logistic regression (Witten and Frank 2005). Boosting algorithms like LogitBoost sequentially apply a classification algorithm, a simple regression function in this case, to reweighted versions of training data. For many classifiers this simple boosting strategy results in dramatic performance improvements (Friedman, Hastie, and Tibshirani 2000).

A LR model was chosen because it can indicate the statistical probability that a piece is written by a certain composer. This is useful for inclusion in the objective function of the music-generation algorithm, as described in the next section. Logistic regression models are less prone to overfitting than other models, such as neural networks, and they only require limited computing power (Tu 1996). Logistic regression models can again be found in many areas, including the creation of habitat models for animals (Pearce and Ferrier 2000), medical diagnosis (Kurt, Ture, and Kurum 2008), credit scoring (Wiginton 1980), and others.

The resulting logistic regression model for composer recognition is displayed in Equations 1–4. The function $f_{comp}(i)$ represents the probability that a piece is composed by composer i. This probability follows a logistic curve, as displayed in Figure 3. The advantage of using this model is that it produces a number in the interval [0,1], which can easily be integrated into the objective function of the music-generation algorithm to assess how well a fragment fits into a given composer's style.

$$f_{comp}(L_i) = \frac{1}{1+e^{-L_i}}, \tag{1}$$

whereby

$$\begin{aligned} L_{HA} = {} & -3.39 + 21.19 \cdot x_1 + 3.96 \cdot x_2 + 6.22 \cdot x_3 \\ & + 6.29 \cdot x_4 - 4.9 \cdot x_5 - 1.39 \cdot x_6 + 3.29 \cdot x_7 \\ & - 0.17 \cdot x_8 + 0 \cdot x_9 - 0.72 \cdot x_{10} + 8.35 \cdot x_{11} \\ & - 4.21 \cdot x_{12}, \end{aligned} \tag{2}$$

$$\begin{aligned} L_{BE} = {} & 6.19 + 5.44 \cdot x_1 + 14.69 \cdot x_2 + 24.36 \cdot x_3 \\ & - 0.45 \cdot x_4 - 6.52 \cdot x_5 - 29.99 \cdot x_6 + 3.84 \cdot x_7 \\ & - 0.38 \cdot x_8 - 3.39 \cdot x_9 - 2.76 \cdot x_{10} + 2.04 \cdot x_{11} \\ & - 0.48 \cdot x_{12}, \end{aligned} \tag{3}$$

$$\begin{aligned} L_{BA} = {} & -4.88 - 13.15 \cdot x_1 - 6.16 \cdot x_2 - 5.28 \cdot x_3 \\ & - 11.63 \cdot x_4 + 11.92 \cdot x_5 + 34 \cdot x_6 - 13.21 \cdot x_7 \\ & + 3.1 \cdot x_8 + 2.37 \cdot x_9 + 0.66 \cdot x_{10} - 5.05 \cdot x_{11} \\ & + 3.03 \cdot x_{12}, \end{aligned} \tag{4}$$

and x_i refers to the corresponding feature value from Table 2.

A musical piece is classified as being by composer i when it has the highest probability for that specific composer according to Equation 1 compared with the probabilities for other composers. A coefficient with a high absolute value indicates a feature that is

Figure 4. Illustration of support vector machine optimization of the margin in the feature space.

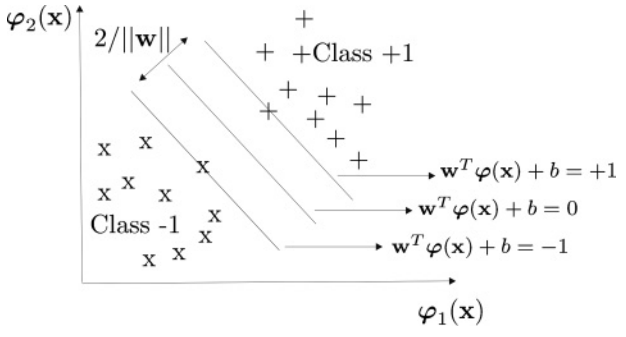

$\varphi_2(\mathbf{x})$
$2/||\mathbf{w}||$
+Class +1
Class -1
$\mathbf{w}^T\varphi(\mathbf{x})+b=+1$
$\mathbf{w}^T\varphi(\mathbf{x})+b=0$
$\mathbf{w}^T\varphi(\mathbf{x})+b=-1$
$\varphi_1(\mathbf{x})$

Table 8. Detailed Results for Logistic Regression by Composer

Composer	*TP Rate*	*FP Rate*	*Precision*	*Recall*	*AUC (%)*
Haydn	0.75	0.10	0.70	0.75	92
Beethoven	0.61	0.05	0.73	0.61	91
Bach	0.93	0.12	0.91	0.93	96

Table 9. Confusion Matrix for Logistic Regression

a	*b*	*c*	*classified as*
190	30	34	a (Haydn)
57	**119**	20	b (Beethoven)
25	15	**555**	c (Bach)

important for distinguishing a particular composer. For example, x_5 (most common melodic interval frequency) has a high coefficient value, especially for Bach. This feature is also at the top of the decision tree (see Figure 2) and occurs in almost all of the rules from the rule set (see Figure 1). In Table 8 an improvement can be seen compared with the two previous models. The results for the composers who are less represented in the data set (Beethoven and Haydn) are much improved. All of the individual AUC values are now over 91%. Logistic regression is a stronger classification model than both the previous models, which is reflected in its ability to get more accurate results for classes with fewer data.

With 83% correctly classified instances from the test set and an AUC value of 94%, the logistic regression model outperforms the previous models (see Table 3). This higher prediction accuracy is reflected in the confusion matrix (see Table 9). The average probability of the misclassified pieces is 64%. Examples of misclassified pieces include the first movement from Bach's Brandenburg Concerto No. 5 in D major, BWV 1050, which is classified as Haydn with a probability of 37%, and the Allegro molto from Beethoven's String Quartet No. 9 in C major, op. 9, no. 3, which is classified as Haydn with a probability of 4%.

Support Vector Machines

In this section, we describe using LibSVM to build a support vector machine (SVM) classifier (Chang and Lin 2011). The SVM is a learning procedure based on statistical learning theory (Vapnik 1995). This method has been applied successfully in many areas, including stock market prediction (Huang, Nakamori, and Wang 2005), text classification (Tong and Koller 2002), gene selection (Guyon et al. 2002), and others. Given a training set of N data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with input data $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{-1, +1\}$, the SVM classifier should fulfil the following conditions (Vapnik 1995; Cristianini and Shawe-Taylor 2000):

$$\begin{cases} \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b \geq +1, & \text{if } y_i = +1 \\ \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b \leq -1, & \text{if } y_i = -1 \end{cases}, \tag{5}$$

which is equivalent to

$$y_i[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b] \geq 1, \quad i = 1, \ldots, N. \tag{6}$$

The nonlinear function $\boldsymbol{\varphi}(\cdot)$ maps the input space to a high-, possibly infinite-dimensional, feature space. In this feature space, these inequalities basically construct a hyperplane $\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b = 0$ discriminating between the two classes (see Figure 4). By minimizing $\mathbf{w}^T\mathbf{w}$, the margin between both classes is maximized.

In primal weight space the classifier then takes the form

$$y(\mathbf{x}) = \text{sign}[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b], \tag{7}$$

but, on the other hand, it is never evaluated in this form. One defines the convex-optimization problem:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{J}(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \qquad (8)$$

subject to

$$\begin{cases} y_i[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b] \geq 1 - \xi_i, & i = 1, \ldots, N \\ \xi_i \geq 0, & i = 1, \ldots, N. \end{cases} \qquad (9)$$

The variables ξ_i are slack variables that are needed to allow misclassifications in the set of inequalities (e.g., due to overlapping distributions). The first part of the objective function tries to maximize the margin between both classes in the feature space and is a regularization mechanism that penalizes for large weights, whereas the second part minimizes the misclassification error. The regularization coefficient C, a positive real constant, should be considered as a tuning parameter in the algorithm. This leads to the following classifier (Cristianini and Shawe-Taylor 2000):

$$y(\mathbf{x}) = \text{sign}\left[\sum_{i=1}^{N}\alpha_i\, y_i\, K(\mathbf{x}_i, \mathbf{x}) + b\right], \qquad (10)$$

whereby $K(\mathbf{x}_i, \mathbf{x}) = \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x})$ is taken with a positive definite kernel satisfying the Mercer theorem. The Lagrange multipliers α_i are then determined by optimizing the dual dual problem. In this research, the radial basis function (RBF) kernel was used to map the feature space to a hyperplane:

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

where d, c, and σ are constants.

A hyperparameter optimization procedure was conducted with GridSearch in WEKA to determine the optimal setting for the regularization parameter C $(0.0001, 0.001, \ldots 10,000)$ and the σ for the RBF kernel $(\sigma = 0.1, 1, \ldots 100{,}000)$. The choice of hyperparameters to test was inspired by settings suggested by WEKA (Waikato Environment for Knowledge Analysis) (2013b). The WEKA implementation of GridSearch performs twofold cross validation on the initial grid. This grid is determined by the two input parameters (C and σ for the RBF kernel). Tenfold cross validation is then performed on the best point of the grid based on the weighted AUC by class size and its adjacent points. If a better pair is found, the procedure is repeated on its neighbors until no better pair is found or the border of the grid is reached (WEKA 2013a).

Table 10. Confusion Matrix for Support Vector Machine

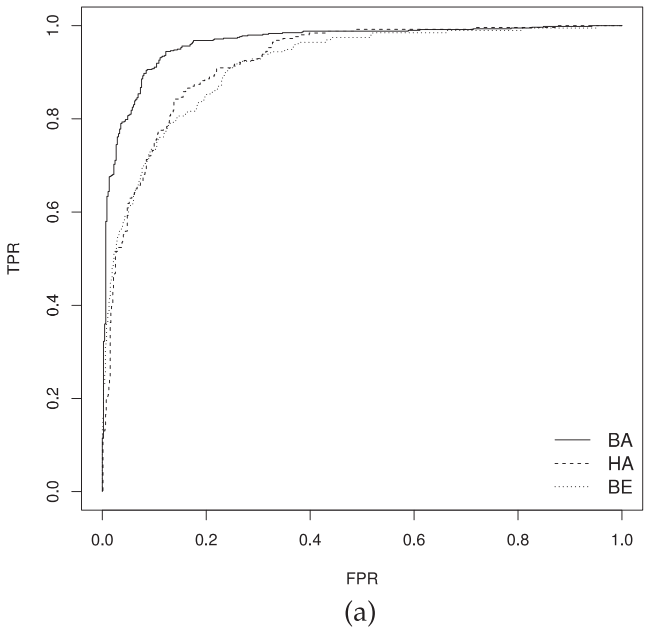
a	*b*	*c*	*classified as*
204	26	24	a (Haydn)
49	**127**	20	b (Beethoven)
22	10	**563**	c (Bach)

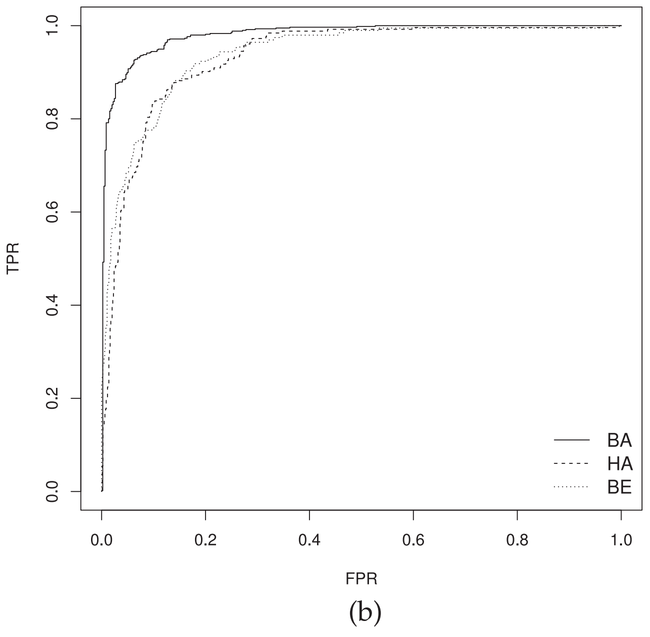
Table 11. Detailed Results for Support Vector Machine by Composer

Composer	*TP Rate*	*FP Rate*	*Precision*	*Recall*	*AUC (%)*
Haydn	0.80	0.09	0.74	0.80	93
Beethoven	0.65	0.04	0.77	0.65	94
Bach	0.95	0.10	0.93	0.95	98

The SVM classifier with nonlinear kernel is a complex, nonlinear function. Trying to comprehend the logic of the classifications made with SVM is quite difficult, if not impossible (Martens, van Gestel, and Baesens 2009; Martens and Provost 2014). The resulting accuracy is 86% and the weighted AUC value is 96% for the SVM with RBF kernel (see Table 3). The confusion matrix (see Table 10) confirms that SVM is the best model for classifying between Haydn, Beethoven, and Bach. Most misclassification occurs between Haydn and Beethoven, which can be explained by the geographical and temporal overlap between the lives of these composers, as discussed earlier. Table 11 reveals that Beethoven has a high AUC, although the recall and true-positive rates are low and the precision rate is higher. This means that pieces that were predicted as being composed by Beethoven have a high accuracy. The model will be conservative in assigning this label, however. The same pattern can be observed in the previous models and seems to be typical when classifying pieces of Beethoven.

Figure 5. Receiver operating characteristic (ROC) curves of the models that performed best: logistic regression (a) and support vector machines (b). The curves plot the false-positive rate (FPR) against the true-positive rate (TPR).





The ROC curves of the two best models according to Table 3 are displayed in Figure 5. The ROC curve displays the trade-off between true-positive rate (TPR) and false-positive rate (FPR). Both models clearly score better than a random classification, which is represented by the diagonal through the origin. Although both models have a high AUC value, the ROC curves for the SVM score slightly better. 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 Allegro moderato from Haydn's String Quartet in C major, op. 74, no. 1, which was classified as Bach with 38% probability, and the theme from Beethoven's Six Variations on a Swiss Song, WO 64, which was classified as Bach with a probability of 38%.

A second experiment was conducted in a similar way with 10CV. Three balanced data sets were randomly extracted from the data set described in the previous section. Each of the data sets (R1, R2, and R3) contains 196 pieces per composer. The results of this experiment are displayed in Table 12. As is to be expected with a smaller data set, the accuracy and AUC values are lower than those from the previous experiment with the full data set (cf. Table 3). They are still reasonably high, however, reaching AUC values of 94% and accuracy of 81% when classifying between the three composers. The model that performed best is again SVM, closely followed by LR.

Table 13 shows the results of a third experiment, wherein pairwise classification models were built. One composer was removed from the data set for each run, so that only two remained. This process was performed using both the full data set and a randomly balanced data set. Classifying between Bach and Beethoven again seemed to be the easiest task. Support vector machines reach an accuracy of 95% on the full data set and 94% on the balanced data set. The AUC values of the classifier reach 99% and 98%, respectively. These results come close to those reported by Dor and Reich (2011), who reach accuracy values of 96% to 98% when classifying between these two composers. Table 13 shows that when classifying between Beethoven and Haydn an accuracy of 80% and 82% can be reached (AUC 88% and 90%, respectively). These results are comparable to the success rates of Kaliakatsos-Papakostas, Epitropakis, and Vrahatis (2011), who

Table 12. Model Evaluation with Tenfold Cross Validation on Reduced Balanced Data Sets

	R1		*R2*		*R3*	
Method	*Acc. (%)*	*WAUC (%)*	*Acc. (%)*	*WAUC (%)*	*Acc. (%)*	*WAUC (%)*
RIPPER	*70* (5.90)	*81* (4.63)	*73* (5.99)	*83* (4.44)	*73* (6.15)	*82* (4.70)
C4.5	*75* (3.06)	*85* (4.11)	*72* (6.00)	*84* (4.11)	*71* (3.74)	*84* (4.05)
LR	78 (3.75)	91 (2.32)	76 (4.31)	91 (2.32)	75 (5.32)	*92* (2.62)
SVM	**81** (4.80)	**93** (2.44)	**79** (4.39)	**92** (1.85)	**80** (3.64)	**94** (2.07)

p < 0.01: *italic; best:* **underlined bold**.
Standard deviations are shown in parentheses.

Table 13. Model Evaluation with Tenfold Cross Validation and Pairwise Classification

		RIPPER (%)	*C4.5 (%)*	*LR (%)*	*SVM (%)*
Bach–Beethoven	Acc.	92 (4.09)	92 (3.10)	92 (2.49)	**95** (2.09)
	WAUC	*89* (5.31)	*91* (4.83)	97 (1.14)	**99** (1.26)
Bach–Beethoven (balanced)	Acc.	*88* (3.97)	*86* (4.63)	**91** (5.28)	**94** (3.66)
	WAUC	*88* (4.05)	*87* (4.38)	96 (3.45)	**98** (2.46)
Bach–Haydn	Acc.	89 (3.15)	*87* (2.85)	91 (2.94)	**94** (2.72)
	WAUC	*87* (4.34)	*88* (3.98)	96 (2.39)	**97** (2.14)
Bach–Haydn (balanced)	Acc.	**87** (5.92)	*83* (2.94)	**91** (3.09)	**92** (4.20)
	WAUC	*88* (6.25)	*88* (3.17)	96 (1.30)	**98** (1.54)
Beethoven–Haydn	Acc.	76 (6.96)	75 (5.28)	**79** (4.33)	**80** (4.44)
	WAUC	*76* (7.44)	*77* (7.68)	86 (4.25)	**88** (3.99)
Beethoven–Haydn (balanced)	Acc.	76 (5.91)	74 (5.55)	79 (4.57)	**82** (6.11)
	WAUC	77 (5.09)	*76* (5.09)	88 (5.99)	**90** (4.62)

p < 0.01: *italic;* p > 0.05: ***bold****; best:* **underlined bold**.
Standard deviations are shown in parentheses.

reported 59% to 88% success performance when classifying between Beethoven and Haydn. To be able to compare results of different studies without bias, however, the exact same database should be used. The SVM classifier again outperforms the others. The quality of logistic regression is comparable, as the statistical tests almost all have a *p*-value greater than 0.01 (with only one exception).

Generating Composer-Specific Music

The idea of algorithmic composition has been present at least since Mozart invented his musical dice game (*Musikalisches Würfelspiel*, cf. Boenn et al. 2009). More recently, composers such as John Cage have made use of chance in their compositional process. His piece *Reunion* (1968) is performed differently each time, because it is controlled by moves on a chess board. Each move on the board is registered by photo-receptors and triggers electronic sounds (Fetterman 1996). Charles Dodge used a natural phenomenon to inspire his piece *The Earth's Magnetic Field* (1970). This piece is a musical translation of the fluctuations of the earth's magnetic field (Alpern 1995). Lejaren Hiller and Leonard Isaacson used the ILLIAC computer in 1957 to generate the score for a string quartet, called the *Illiac Suite* (Alpern 1995). Newer compositional systems include, among others, the

Continuator (Pachet 2003), OMax (Assayag et al. 2006), and MIMI (Schankler, Chew, and François 2014). An approach related to this research, yet using different classes and features, is described by Pachet (2009). Global features are used to develop an SVM classification model that can classify between tonal, Brownian, serial, long, and short melodies. Existing melodies were then transformed into another type by improving the SVM's rating for the particular melody. For a more complete overview of algorithmic composition systems the reader is referred to papers by Herremans and Sörensen (2012) and Fernández and Vico (2013).

A limited number of researchers have investigated automatic composition in the style of a specific composer. David Cope's "Experiments in Musical Intelligence" (EMI) software extracts signatures of musical pieces using pattern matching to understand a composer-specific style. He uses a grammar-based system to generate music in the style of a chosen composer (Cope 1991). The generation of Bach chorales has received some attention in the field of automatic composition. Kemal Ebcioğlu created CHORAL, an expert system that uses 350 rules for harmonizing four-part Bach chorales (Ebcioğlu 1988). A Bach harmonization system was also developed by Hild, Feulner, and Menzel (1992). They created a hybrid system with artificial neural networks called HARMONET. Somnuk Phon-Amnuaisuk implemented a control language to harmonize Bach chorales (Phon-Amnuaisuk 2002), and Randall Spangler built a system that extracts rules from Bach chorales (Spangler 1999). These rules were implemented in a system that can harmonize in the style of Bach in response to input in real time.

Although there has been some research about automatic composition in the style of one particular composer, combining multiple composers has not received as much attention in research to date. The composition system developed by Cope (2000) called SARA can combine the influence of selected composers in the generated music. Most of the existing systems, however, focus on learning or defining one particular style, which might be extracted from a collection of works by one composer (e.g., Farbood and Schoner 2001; Herremans, Sörensen, and Conklin 2014). In this research, we take a novel approach by allowing the user to dynamically alter and combine the influence of multiple composers in a stream of newly generated music.

In previous research, we developed a VNS algorithm that could efficiently generate music in the style of fifth-species counterpoint, a type of polyphonic music (Fux 1971). To do this, the process of composing music was modeled as a combinatorial optimization problem whereby the objective is to find a musical fragment that fits the counterpoint style as well as possible. In order to evaluate how well a fragment adheres to the counterpoint style, the rules of this style were quantified to form an objective function. A detailed description of the inner workings of the VNS and the objective function has been published by the authors elsewhere (Herremans and Sörensen 2013a).

To generate music with composer-specific characteristics, the existing objective function for evaluating counterpoint (f_{cp}) was extended with the probabilities of the LR model. This model was preferred over the slightly more accurate SVM model because it is easier to comprehend (Martens et al. 2007) and returns a clearly defined probability for each composer. The resulting objective function for composer i is displayed in Equation 11. When composing with characteristics of a certain composer i, the weights a_i should be set high and the others to 0. This ensures that only the characteristics of the counterpoint and of composer i are taken into account. A low rating corresponds to better contrapuntal music with more influences of composer i.

$$f_i = f_{cp} + \sum_{i \in BE,BA,HA} a_i \cdot (1 - f_{comp}(L_i)). \tag{11}$$

The new model was added to the existing objective function for counterpoint to ensure that some basic harmonic and melodic rules are still checked. By temporarily removing the first term from Equation 11, it is quickly confirmed by listening that generating music that only adheres to the rules extracted in the previous section, without optimizing f_{cp}, does not result in musically meaningful results. The counterpoint rules are therefore

Figure 6. User interface of FuX.

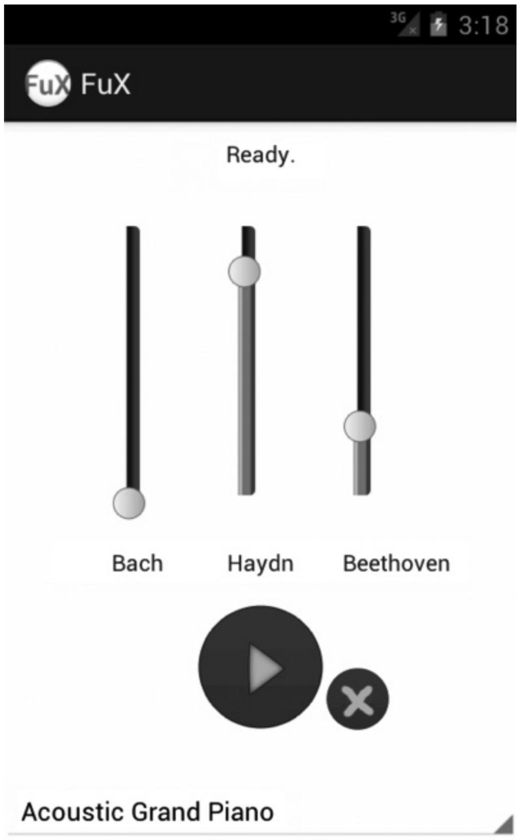
necessary to ensure that the generated music also optimizes some basic musical properties such as "only consonant intervals are permitted." With this new objective function, the VNS algorithm is able to generate contrapuntal music with characteristics of a given composer.

Implementation: FuX

The music-generation algorithm with the objective function discussed in the previous section (based on counterpoint rules and the logistic regression model) was implemented as an Android application called FuX, named after the author one of the most influential books on counterpoint, *Gradus ad parnassum* (Fux 1971). Johann Fux was an Austrian composer and theorist who lived in the 17th and 18th centuries. The FuX app is available as open source software in the Google Play store (play.google.com/store/apps/details?id=com.dh.fux2) and can be run on any Android-based device. The developed app can generate a stream of contrapuntal music with composer-specific characteristics. This music is continuously generated while it is being played.

Because resources are typically limited on mobile devices, careful consideration had to be given to the question of how implement the music-generation algorithm. Instead of using the Android software development kit to develop Java applications (Google 2013), Son and Lee (2011) recommend using the Android native development kit (NDK) for computationally expensive tasks, and confirm the recommendation with benchmark experiments. For developing the music-generation algorithm, the Android NDK was used to compile C++ code for the Android platform. Java's multithreading capabilities were then used to allow continuous playback while new music is being generated. A detailed description of the implementation details of FuX 1.0 is given by the authors in a separate publication (Herremans and Sörensen 2013b).

The graphical user interface of FuX 2.0 is displayed in Figure 6. The three sliders (or SeekBars) give the user control over the weights a_i of the objective

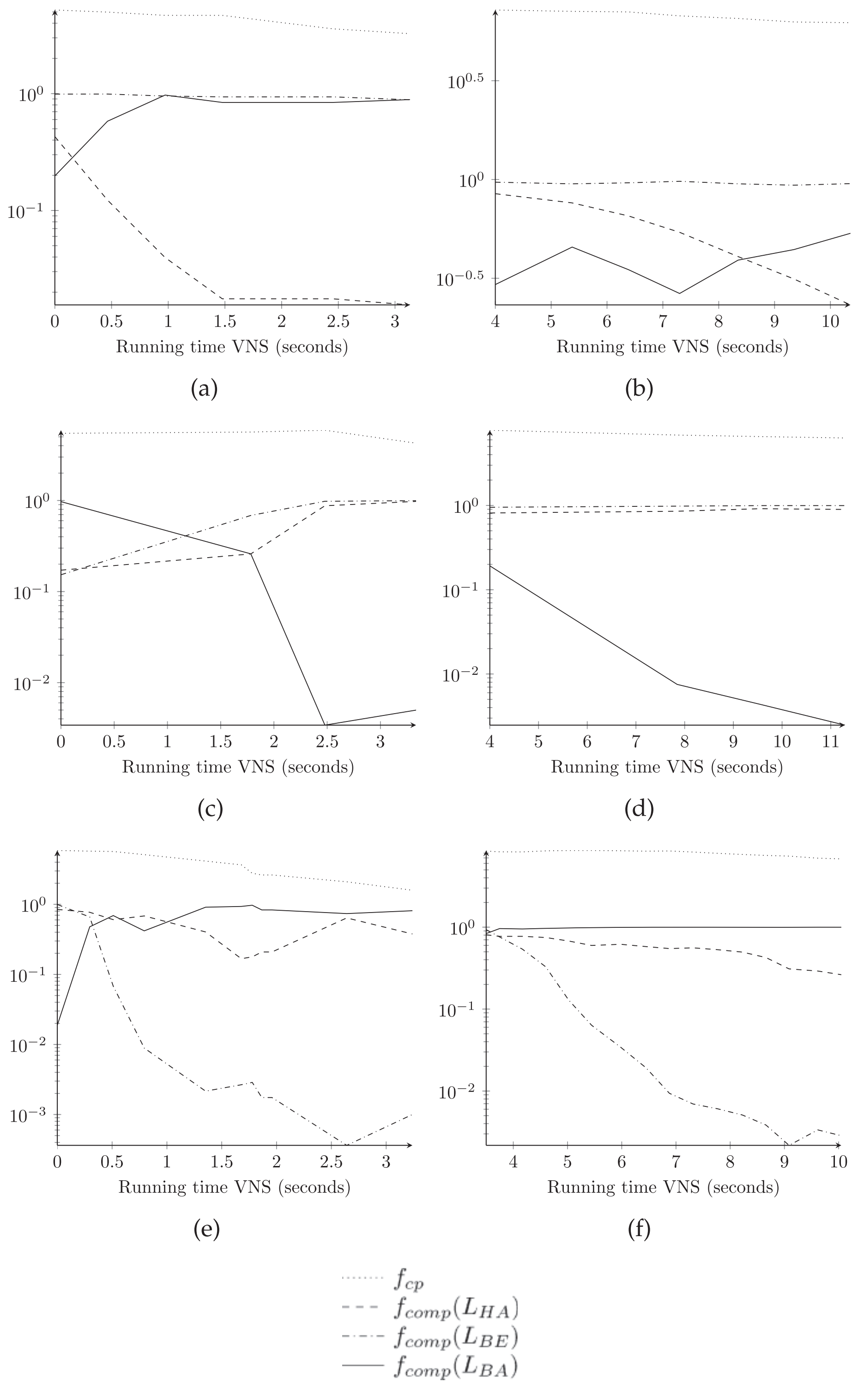


function (see Equation 11). This even allows a user to generate music consisting of a mix of multiple composers' styles if he or she wishes to do so. The playback instrument can be chosen and dynamically changed by the user.

Results

The resulting composer-specific music-generation algorithm was tested on an Eclipse Android Virtual Device with Android 4.0.3, an ARM processor, and 512 MB of RAM. The emulator was installed on an OpenSuse system with an Intel Core 2 Duo CPU running at 2.20 GHz with 3.8 GB RAM. Figure 7 displays the evolution of the solution quality over time. The plots on the left-hand side describe the generation of the *cantus firmus* or bass line. The

Figure 7. Evolution of solution quality over time: *Haydn* cantus firmus *(a), Haydn counterpoint (b), Bach* cantus firmus *(c), Bach counterpoint (d), Beethoven* cantus firmus *(e), and Beethoven counterpoint (f).*



plots on the right-hand side describe the evolution of the rating of the counterpoint line or top line. In the experiment, 16 measures are generated with a cutoff time of 12 seconds. FuX first generates the bass line and continues with the top line. Because the main objective is to produce music with composer-specific characteristics, the weight of the respective composer's rating is set very high (100); this ensures that the probability of a composer is preferred by the algorithm over the counterpoint rules. Figure 7 shows a drastic improvement of the selected composer's rating for each of the three composers. For example, in Figure 7a $f_{comp}(L_{HA})$ goes down rapidly over time, while $f_{comp}(L_{BE})$ and $f_{comp}(L_{BA})$ remain relatively stable. This means that the generated music actually contains composer-specific elements according to the LR model that was built. When optimizing for a specific composer, no real change in the ratings for the other composers can be noted. The improvements of the counterpoint rating are not very high, but are enough to add basic musical properties to the fragment. Of course, the thought processes of the great composers are far more complex than can be captured by the melodic features used in this research. The limited set of composer-specific characteristics (see Table 2) that FuX overlays on the counterpoint fragment are not enough by themselves to generate a piece of music that would be recognized as being composed by one of the selected composers. The generated music does contain characteristics of the selected composer, however. All three composers used in this research composed music that differs in much more than the limited set of characteristics that FuX controls. Bach, e.g., composed during the Baroque period, whereas Beethoven's work was composed during the transition from the Classical to the Romantic era. The differences between the musical styles characteristic of these different periods are vastly more encompassing than simple variations in the melodic characteristics recognized by FuX. This research can be seen as a first step towards creating a system that is able to generate more complete musical pieces in the style of a given composer.

Owing to large differences in computing power between different Android devices, the quality of the generated music is highly dependent on the architecture of the mobile device on which it is run. Still, the subjective opinion of the authors is that the generated stream of music sounds pleasant to the ear, even on relatively modest hardware. The reader is invited to install the app and listen to the resulting music.

Conclusions

A number of musical features were extracted from a large database of music. Based on these features, four classification models were built. The first two models, an if–then rule set and a decision tree, give the user more insight into the musical style of a composer, e.g., "Beethoven is less likely to focus on using one particular melodic interval, in contrast to Haydn or Bach, who have a higher prevalence of the most common melodic interval." The other two models, a logistic regression and a support vector machine classifier, can more accurately classify musical pieces from Haydn, Beethoven, and Bach. The first of these models is integrated into the objective function of a variable neighborhood search algorithm that can efficiently generate contrapuntal music. The resulting algorithm was implemented as an Android app called FuX, which is able to play a stream of contrapuntal music that has composer-specific characteristics and sounds pleasing, at least to the subjective ear of the authors.

Combining a certain composer's characteristics with the contrapuntal style creates a peculiar fusion of styles, yet this approach is merely an initial step towards a more complete system. In the future it would be interesting to work with composer classification models built on a data set of one particular style. Enforcing basic musical properties while generating pieces with characteristics specific to a composer might then be done by integrating rules specific to the chosen style instead of the currently used counterpoint rules. Other future extensions of this research include working with other musical styles and more voices, and adding a recurring theme to the music. FuX might also be ported to other platforms, such as iOS from Apple.

References

Alpern, A. 1995. "Techniques for Algorithmic Composition of Music." Available online at citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.9364&rep=rep1&type=pdf&gt. Accessed July 2015.

Assayag, G., et al. 2006. "OMax Brothers: A Dynamic Topology of Agents for Improvization Learning." In *Proceedings of the ACM Multimedia Workshop on Audio and Music Computing for Multimedia*, pp. 125–132.

Baesens, B., et al. 2003. "Using Neural Network Rule Extraction and Decision Tables for Credit-Risk Evaluation." *Management Science* 49(3):312–329.

Berenzweig, A., et al. 2004. "A Large-Scale Evaluation of Acoustic and Subjective Music-Similarity Measures." *Computer Music Journal* 28(2):63–76.

Boenn, G., et al. 2009. "Automatic Composition of Melodic and Harmonic Music by Answer Set Programming." In M. G. Banda, and E. Pontelli, eds. *Logic Programming*. Berlin: Springer, pp. 160–174.

Buzzanca, G. 2002. "A Supervised Learning Approach to Musical Style Recognition." Available online at citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.2159&rep=rep1&type=pdf. Accessed September 2014.

Byrd, D., and T. Crawford. 2002. "Problems of Music Information Retrieval in the Real World." *Information Processing and Management* 38(2):249–272.

Casey, M., et al. 2008. "Content-Based Music Information Retrieval: Current Directions and Future Challenges." *Proceedings of the IEEE* 96(4):668–696.

Chang, C.-C., and C.-J. Lin. 2011. "LIBSVM: A Library for Support Vector Machines." *ACM Transactions on Intelligent Systems and Technology* 2(3):27. Available online at dx.doi.org/10.1145/1961189.1961199. Subscription required.

Cohen, W. 1995. "Fast Effective Rule Induction." In A. Prieditis and S. Russell, eds. *Proceedings of the Twelfth International Conference on Machine Learning*. Burlington, Massachusetts: Morgan Kaufmann, pp. 115–123.

Conklin, D. 2013. "Multiple Viewpoint Systems for Music Classification." *Journal of New Music Research* 42(1):19–26.

Cope, D. 1991. *Computers and Musical Style*. Oxford: Oxford University Press.

Cope, D. 2000. *The Algorithmic Composer*. Madison, Wisconsin: A-R Editions.

Cramer, G., R. Ford, and R. Hall. 1976. "Estimation of Toxic Hazard: A Decision Tree Approach." *Food and Cosmetics Toxicology* 16(3):255–276.

Craven, M., and J. Shavlik. 1996. "Extracting Tree-Structured Representations of Trained Networks." In *Advances in Neural Information Processing Systems*. Burlington, Massachusetts: Morgan Kaufmann, pp. 24–30.

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

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

Dor, O., and Y. Reich. 2011. "An Evaluation of Musical Score Characteristics for Automatic Classification of Composers." *Computer Music Journal* 35(3):86–97.

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

Dubnov, S., et al. 2003. "Using Machine-Learning Methods for Musical Style Modeling." *Computer* 36(10):73–80.

Ebcioğlu, K. 1988. "An Expert System for Harmonizing Four-Part Chorales." *Computer Music Journal* 12(3):43–51.

Farbood, M., and B. Schoner. 2001. "Analysis and Synthesis of Palestrina-style Counterpoint Using Markov Chains." In *Proceedings of the International Computer Music Conference*, pp. 471–474.

Fawcett, T. 2003. "ROC Graphs: Notes and Practical Considerations for Data Mining Researchers." Technical Report HPL-2003–4. Palo Alto, California: Hewlett-Packard Laboratories. Available online at www.hpl.hp.com/techreports/2003/HPL-2003–4.pdf. Accessed September 2014.

Fernández, J. D., and F. Vico. 2013. "AI Methods in Algorithmic Composition: A Comprehensive Survey." *Journal of Artificial Intelligence Research* 48:513–582.

Fetterman, W. 1996. *John Cage's Theatre Pieces: Notations and Performances*. London: Routledge.

Friedl, M., and C. Brodley. 1997. "Decision Tree Classification of Land Cover from Remotely Sensed Data." *Remote Sensing of Environment* 61(3):399–409.

Friedman, J., T. Hastie, and R. Tibshirani. 2000. "Additive Logistic Regression: A Statistical View of Boosting." *Annals of Statistics* 28(2):337–407.

Fux, J. 1971. *The Study of Counterpoint: From Johann Joseph Fux's* Gradus ad Parnassum, trans. A. Mann. New York: Norton.

Geertzen, J., and M. van Zaanen. 2008. "Composer Classification Using Grammatical Inference." In *Proceedings of the International Workshop on Machine Learning and Music*, pp. 17–18.

Gheyas, I., and L. Smith. 2010. "Feature Subset Selection in Large Dimensionality Domains." *Pattern Recognition* 43(1):5–13.

Ghias, A., et al. 1995. "Query by Humming: Musical Information Retrieval in an Audio Database." In *Proceedings of the ACM International Conference on Multimedia*, pp. 231–236.

Gómez Gutiérrez, E. 2006. "Tonal Description of Music Audio Signals." PhD dissertation, Universitat Pompeu Fabra, Departament de Tecnologia, Barcelona.

Google. 2013. *Google Android SDK*. Available online at developer.android.com/sdk. Accessed November 2013.

Guyon, I., et al. 2002. "Gene Selection for Cancer Classification Using Support Vector Machines." *Machine Learning* 46(1–3):389–422.

Hall, M., et al. 2009. "The WEKA Data Mining Software: An Update." *SIGKDD Explorations* 11(1):10–18.

Hand, D., and W. Henley. 1997. "Statistical Classification Methods in Consumer Credit Scoring: A Review." *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 160(3):523–541.

Herremans, D., D. Martens, and K. Sörensen. 2014. "Dance Hit Song Prediction." *Journal of New Music Research* 43(3):291–302.

Herremans, D., and K. Sörensen. 2012. "Composing First Species Counterpoint with a Variable Neighbourhood Search Algorithm." *Journal of Mathematics and the Arts* 6(4):169–189.

Herremans, D., and K. Sörensen. 2013a. "Composing Fifth Species Counterpoint Music with a Variable Neighborhood Search Algorithm." *Expert Systems with Applications* 40(16):6427–6437.

Herremans, D., and K. Sörensen. 2013b. "FuX, an Android App That Generates Counterpoint." In *Proceedings of IEEE Symposium on Computational Intelligence for Creativity and Affective Computing*, pp. 48–55.

Herremans, D., K. Sörensen, and D. Conklin. 2014. "Sampling the Extrema from Statistical Models of Music with Variable Neighbourhood Search." In *Proceedings of the Joint International Computer Music Conference and Sound and Music Computing Conference*, pp. 1096–1103.

Hild, H., J. Feulner, and W. Menzel. 1992. "HARMONET: A Neural Net for Harmonizing Chorales in the Style of J.S. Bach." In *Advances in Neural Information Processing Systems*, pp. 267–274.

Hillewaere, R., B. Manderick, and D. Conklin. 2010. "String Quartet Classification with Monophonic Models." *Proceedings of the International Society for Music Information Retrieval Conference*, pp. 537–542.

Huang, W., Y. Nakamori, and S.-Y. Wang. 2005. "Forecasting Stock Market Movement Direction with Support Vector Machine." *Computers and Operations Research* 32(10):2513–2522.

Isermann, R., and P. Balle. 1997. "Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes." *Control Engineering Practice* 5(5):709–719.

Kaliakatsos-Papakostas, M., M. Epitropakis, and M. Vrahatis. 2011. "Weighted Markov Chain Model for Musical Composer Identification." In *Applications of Evolutionary Computation*. Berlin: Springer, pp. 334–343.

Kassler, M. 1966. "Toward Musical Information Retrieval." *Perspectives of New Music* 4(2):59–67.

Kim, J., et al. 2005. "Detecting the Change of Customer Behavior Based on Decision Tree Analysis." *Expert Systems* 22(4):193–205.

Kononenko, I. 2001. "Machine Learning for Medical Diagnosis: History, State of the Art and Perspective." *Artificial Intelligence in Medicine* 23(1):89–109.

Kurt, I., M. Ture, and A. Kurum. 2008. "Comparing Performances of Logistic Regression, Classification and Regression Tree, and Neural Networks for Predicting Coronary Artery Disease." *Expert Systems with Applications* 34(1):366–374.

Laurier, C., J. Grivolla, and P. Herrera. 2008. "Multimodal Music Mood Classification Using Audio and Lyrics." In *International Conference on Machine Learning and Applications*, IEEE, pp. 688–693.

Lippincott, A. 2002. "Issues in Content-Based Music Information Retrieval." *Journal of Information Science* 28(2):137–142.

Manaris, B., et al. 2005. "Zipf's Law, Music Classification, and Aesthetics." *Computer Music Journal* 29(1):55–69.

Martens, D. 2008. "Building Acceptable Classification Models for Financial Engineering Applications." *SIGKDD Explorations* 10(2):30–31.

Martens, D., T. van Gestel, and B. Baesens. 2009. "Decompositional Rule Extraction from Support Vector Machines by Active Learning." *IEEE Transactions on Knowledge and Data Engineering* 21(2):178–191.

Martens, D., and F. Provost. 2014. "Explaining Data-Driven Document Classifications." *Management Information Systems Quarterly* 38(1):73–99.

Martens, D., et al. 2007. "Comprehensible Credit Scoring Models Using Rule Extraction from Support Vector Machines." *European Journal of Operational Research* 183(3):1466–1476.

Martens, D., et al. 2011. "Performance of Classification Models from a User Perspective." *Decision Support Systems* 51(4):782–793.

McKay, C., and I. Fujinaga. 2006. "jSymbolic: A Feature Extractor for MIDI Files." In *Proceedings of the International Computer Music Conference*, pp. 302–305.

McKay, C., and I. Fujinaga. 2007. "Style-Independent Computer-Assisted Exploratory Analysis of Large Music Collections." *Journal of Interdisciplinary Music Studies* 1(1):63–85.

McKay, C., and I. Fujinaga. 2009. "jMIR: Tools for Automatic Music Classification." In *Proceedings of the International Computer Music Conference*, pp. 65–68.

McKay, C., et al. 2005. "ACE: A Framework for Optimizing Music Classification." In *Proceedings of the International Society for Music Information Retrieval Conference*, pp. 42–49.

Mearns, L., D. Tidhar, and S. Dixon. 2010. "Characterisation of Composer Style Using High-Level Musical Features." In *Proceedings of International Workshop on Machine Learning and Music*, pp. 37–40.

Ngai, E., L. Xiu, and D. Chau. 2009. "Application of Data Mining Techniques in Customer Relationship Management: A Literature Review and Classification." *Expert Systems with Applications* 36(2):2592–2602.

Pachet, F. 2003. "The Continuator: Musical Interaction with Style." *Journal of New Music Research* 32(3):333–341.

Pachet, F. 2009. "Description-Based Design of Melodies." *Computer Music Journal* 33(4):56–68.

Pearce, J., and S. Ferrier. 2000. "Evaluating the Predictive Performance of Habitat Models Developed Using Logistic Regression." *Ecological Modelling* 133(3):225–245.

Phon-Amnuaisuk, S. 2002. "Control Language for Harmonisation Process." In *Music and Artificial Intelligence*. Berlin: Springer, pp. 155–167.

Pollastri, E., and G. Simoncelli. 2001. "Classification of Melodies by Composer with Hidden Markov Models." In *Proceedings of the International Conference on Web Delivering of Music*, pp. 88–95.

Quinlan, J. 1993. *C4.5: Programs for Machine Learning*, vol. 1. San Mateo, California: Morgan Kaufmann.

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

Sapp, C. 2005. "Online Database of Scores in the Humdrum File Format." In *Proceedings of the International Conference on Music Information Retrieval*, pp. 664–665.

Schankler, I., E. Chew, and A. R. François. 2014. "Improvising with Digital Auto-Scaffolding: How Mimi Changes and Enhances the Creative Process." In N. Lee, ed. *Digital Da Vinci*. Berlin: Springer, pp. 99–125.

Searls, D. B. 2002. "The Language of Genes." *Nature* 420(6912):211–217.

Shmueli, G., and O. Koppius. 2011. "Predictive Analytics in Information Systems Research." *Management Information Systems Quarterly* 35(3):553–572.

Son, K., and J. Lee. 2011. "The Method of Android Application Speed Up by Using NDK." In *Proceedings of the International Conference on Awareness Science and Technology*, IEEE, pp. 382–385.

Spangler, R. R. 1999. "Rule-Based Analysis and Generation of Music." PhD dissertation, California Institute of Technology, Pasadena, California.

Tan, P.-N., M. Steinbach, and V. Kumar. 2005. *Introduction to Data Mining*. Harlow: Pearson Education.

Tong, S., and D. Koller. 2002. "Support Vector Machine Active Learning with Applications to Text Classification." *The Journal of Machine Learning Research* 2:45–66.

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.

Tzanetakis, G., and P. Cook. 2002. "Musical Genre Classification of Audio Signals." *IEEE Transactions on Speech and Audio Processing* 10(5):293–302.

Tzanetakis, G., A. Ermolinskyi, and P. Cook. 2003. "Pitch Histograms in Audio and Symbolic Music Information Retrieval." *Journal of New Music Research* 32(2):143–152.

Van Kranenburg, P., and E. Backer. 2004. "Musical Style Recognition: A Quantitative Approach." In *Proceedings of the Conference on Interdisciplinary Musicology*, pp. 106–107.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory*. Berlin: Springer.

WEKA (Waikato Environment for Knowledge Analysis). 2013a. "GridSearch." Available online at weka.sourceforge.net/doc.stable/weka/classifiers/meta/GridSearch.html . Accessed October 2014.

WEKA (Waikato Environment for Knowledge Analysis). 2013b. "Optimizing Parameters." Available online at weka.wikispaces.com/Optimizing+parameters. Accessed October 2014.

Whitman, B., and P. Smaragdis. 2002. "Combining Musical and Cultural Features for Intelligent Style Detection." In *Proceedings of the International Conference on Music Information Retrieval*, pp. 47–52.

Wiginton, J. 1980. "A Note on the Comparison of Logit and Discriminant Models of Consumer Credit

Behavior." *Journal of Financial and Quantitative Analysis* 15(3):757–770.

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

Wolberg, W., and O. Mangasarian. 1990. "Multisurface Method of Pattern Separation for Medical Diagnosis Applied to Breast Cytology." *Proceedings of the National Academy of Sciences, U.S.A.* 87(23):9193–9196.

Wołkowicz, J., Z. Kulka, and V. Kešelj. 2008. "N-Gram-Based Approach to Composer Recognition." *Archives of Acoustics* 33(1):43–55.

Wu, X., et al. 2008. "Top Ten Algorithms in Data Mining." *Knowledge and Information Systems* 14(1):1–37.

Youngblood, J. 1958. "Style as Information." *Journal of Music Theory* 2(1):24–35.

Zipf, G. K. 1935. *The Psychobiology of Language*. Boston, Massachusetts: Houghton Mifflin.