**4. Generation**

After analyzing the results of the classifiers, the first step in creating our algorithmic composition software was to choose one classifier to use going forward in the hybridization process. On top of providing class predictions, each classifier supplied a model, intended to inform the reader on how it’s decision rules were devised. These models are important, as they are the building block upon which we intend to build our music generator. Of the five classifiers, the first two eliminated were the rule-based and decision tree models, JRip and J48. While the classifiers provided positive features, such as easy to understand outputs that outlined the rules used explicitly, it was clear that these approaches were simply not of the same accuracy as their more complex counterparts.

Of our three remaining classifiers, we chose next to eliminate the complex classifiers, Multilayer Perceptron and Logistic Regression. Despite these algorithms statistically doing a better job of classifying the musical scores, MLPs and Logistic Regression models did not give a satisfactorily digestible answer as to why the classes were separated the way they were. For this reason, it was difficult to conceive of a way to use these classifiers to inform the generative process of any algorithmic composition software.

We decided to proceed using the Naïve Bayes approach because it supplied us with a nice middle ground between the previously mentioned choices. It provides an easy, statistical model for us to easily adapt to the generative process. On top of this, the Bayes model yielded a more respectable AUC value (.838) than the other simple algorithms of J48 (.753) and JRip (.773).

**4.1 Method**

In perhaps our most contributory work, we move to the generation process of the experiment. The task laid ahead of us was to find a way to utilize the intelligence gained from our Naïve Bayes classifier to inspire the algorithmic composition of music. After consideration of the classifier results and output, we decided to turn our attention to an avenue of algorithmic composition that has been less explored than some others such as artificial neural networks and formal grammars: Cellular automata.

**4.1.1 Cellular Automata**

The concept of cellular automata (Singular: Automaton) was first proposed by John von Neumann in the 1950’s and reached a peak in popularity during the 70’s due to John Conway’s now famous “Game of Life” 3-D cellular automata model [17]. Based upon the biological cellular replication process, a cellular automata model is represented by a grid of cells, each of which is represented as one of a finite number of states (i.e. “ON” or “OFF”). This grid can be of any finite number of dimensions. The grid progresses in temporally-linear fashion, with each cell shifting states at any given step in time. This shift of the cell states is based upon two factors: the states of the surrounding cells in a pre-determined area defined as it’s neighborhood, and a set of transitionary rules which dictate the outcome based on that neighborhood [17]. One of the most famous example of a cellular automata, the Wolfram Elementary Algorithms (Figure 7), adds a new line of cells below the previous generated line with each sequential step in time, with the states of these new cells based upon a neighborhood of the three cells directly above it, and a selected transitionary rule set [18]. With 256 possible rule sets, there are countless possibilities of how the algorithm can compose the sequence of cells, and many produce interesting patterns, such as fractals.

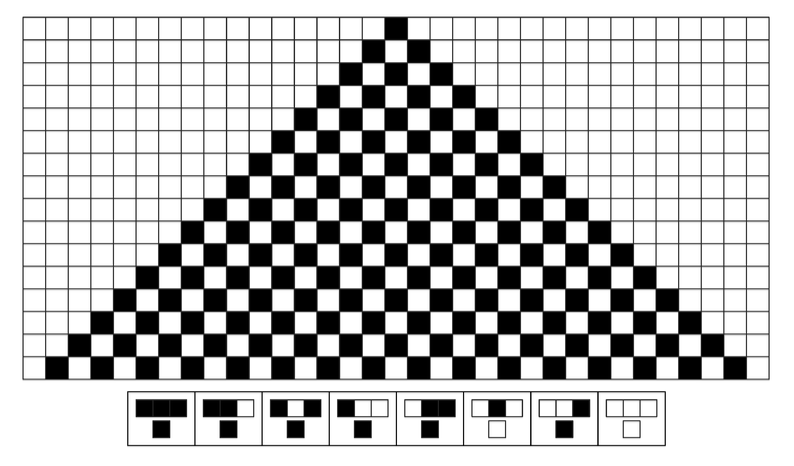


Figure 7 - Rule 250 in the Wolfram Elementary Algorithm Suite

Rule model’s such as Wolfram’s provide a unique avenue of exploration for musical composition. The patterns found within these automata rules provide a built-in approach to chaotic music composition. However, those preliminary cellular automata models were only able to create music in an “uncontrolled” way and resulted in music that was not necessarily homogenous with any preconceived style [6]. The next natural step was to create transitionary rules that were informed by the true tendencies of music, so as to control the music being generated.

**4.2.2 Adapted Musical Model**



Figure 8 – A table mapping the values of a four-bit binary sequence to

the values within the chromatic circle for use in conjunction with

cellular automata musical composition

In an attempt to explore this avenue of musically informed cellular automata, we devised a system inspired by the aforementioned Wolfram Algorithm. Using cells that have one of two states – “On” and “Off” – we are able to interpret a string of these cells as a binary sequence. We chose to map these cells as four-byte binary sequences (16 possible combinations) to the 12 notes of the chromatic circle, with the note C doubled to ease generation given the cyclical nature of the scale. While this system does not currently take into account rhythm, a rest musical character was also encoded for potential future works, as well as terminate and start. A comprehensive look at this binary-mapping is outlined in figure 8.

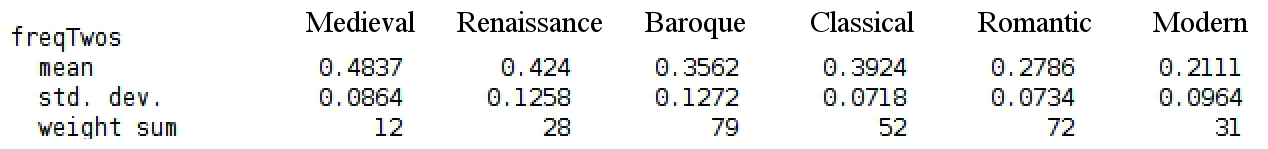
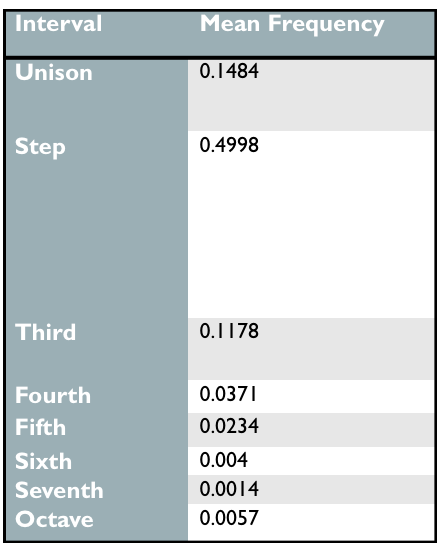


Figure 9 – An example of the statistical output provided by the Naïve Bayes classifier

pertaining to the frequency of stepwise intervals

After the groundwork of our cellular automata model was laid out, it was time to create transitionary rules inspired by the intelligence gained through our classification process. At the beginning of each transition, a random float value between 0.0 and 1.0 was generated. The Naïve Bayes classifier provided a statistical output from which we were able to derive the average probability of any single interval occurring at a given step in time. Figure 9 demonstrates how the probability of a single step interval is represented in this output. We were therefore able to map our randomly generated float value to one of the eight interval possibilities. Whichever interval corresponded to the randomly generated float value was determined to be the distance between the previous note and our new note. The states of each cell in the four-byte sequence would therefore transition from the previous note’s binary representation to a new binary sequence representing our newly found note. In essence, we are generating the interval between the notes, rather than the note itself. Along with creating more aurally pleasing musical phrases, this helps ease the challenges of representing key signatures within pieces of music.

To help visualize this process, figure 10 provides a mock example of this process. In this example, we are attempting to replicate the medieval era. Thus, the mean frequency values match those discovered by our Naïve Bayes classifier for the medieval era. The float value .6197 is randomly generated and mapped within the mean frequencies of the medieval era. It is determined that the float value falls within the stepwise interval partition of our chart. Therefore, if we were ascending from the note C, or 0001, we could arrive at D, or 0011.



.6197

Figure 10 – A visual representation of how a random float number is

mapped to the probabilities of each musical interval

To further demonstrate the potentials of this system, the software gives the user the ability to select which era of music they wish to replicate. At the click of a button, the system is able to swap the statistics used in transitionary rule generation to those indicated by the Naïve Bayes output to correspond with the user’s indicated era, so as to encourage the system to follow the tendencies of the desired era. This feature helps the software stand out and puts to use the predictive power of our classification approach to rule generation.

The last feature we implemented was a range-check system. In preliminary testing, we found that allowing the note to change in ascending or descending fashion on a 50-50 basis, while relatively common sight within the world of music, was not controlled enough for our experiment, as the true randomness allowed for many algorithmic compositions to get out of hand in terms of range. We therefore found the average distance between the highest note and lowest note within an era of music and dictated that the composition software stays within that range when composing. This allows music that has traditionally had more range to flourish in this sense, while static pieces from earlier eras stick within a more contained range of notes.