Machine Listening with Very Small Training Datasets

A Thesis

Presented to the Faculty of

The Department of Computer Science

Kutztown University of Pennsylvania

Kutztown, Pennsylvania

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Wissam Malke

January, 2017

APPROVAL TO SUBMIT THESIS

This thesis presented by Wissam Malke entitled

Machine Listening with Very Small Training Datasets

is hereby approved:

Approved: _____ _____

Date Dr. Dale Parson

_____ _____

Date Dr. Oskars Rieksts

_____ _____

Date (Chair, Department of Computer Science)

_____ _____

Date (Dean, Graduate Studies)

Abstract

The human brain and its learning mechanics are great secrets that researchers are still working on understanding, and revealing their mysteries. Despite the complexities of the human brain and its learning mechanics, researchers are able to recognize some obvious facts about the brain that improve its learning experience, such as, using small data sets and distributed sessions for training the brain is considered to be the most effective method for learning, whereas, cramming information, and traditional teaching methods such as stand-and-deliver lectures for long hours are proven to be ineffective and have very little results on the brain learning experience [1], because the brain cannot absorb this information and process it, and as a result it cannot develop the links between the neurons' cells so the brain can acquire and retain the information.

In our research, we work on finding machine learning models that are capable of learning from small datasets similar to the ones used for human learning, mimicking the best human brain learning method, and predicting results that are statistically similar to the results of short-term human learning on similar tasks.

After considering multiple datasets to use in our research, we settled on the datasets of the Sonification research [2] that was led by Dr. Dale Parson for 2 years. The datasets of the Sonification research are very convenient and suitable for our research for many reasons.

First, the datasets were very small which conform to our main theoretical concept of using small datasets for training the machine learning models.

Second, the Sonification research results are available and can be compared with the results of our machine learning models.

Third, we will be able to employ the results of our research to benefit the completion of the Sonification research, in the sense that Dr. Parson will be using the result of our research which is represented by the Virtual Survey Listener Tool to test new Sonification approaches for finding the best one among all of the Sonification approaches.

Acknowledgments

I would like to express my sincere gratitude to all those who have made this thesis possible. Especially my thesis advisor, Dr. Dale Parson, who has guided my work with patience, enthusiasm, and immense knowledge. Without Dr. Parson's continuous support and motivation, this thesis could not have been successfully accomplished.

Besides my advisor, I would like to thank Dr. Rieksts for taking the time to remediate my thesis documentation.

Also, my sincere thanks goes to Dr. Spiegel, who has been always available for advisement and guidance in the Master Program's journey.

Finally, I would like to thank my family, especially my wife, for all the support and motivation.

Table of Contents

List of Tables and Figures

Chapter 1

1. Introduction

Reverse engineering the human brain has been always a main objective for researchers. Many models and machine learning algorithms were developed for mimicking the brain learning mechanics, and inference process. Despite the progress these models have made, in many cases the results were deviating from the correct prediction for various reasons. Thus, researchers have dealt with these problems by introducing correction factors such as adding hidden variable, taking event order in consideration, and introducing quantum mechanics as possible solution for undetermined deficiencies.

Since it is proven that human brain learning process is more efficient on a smaller data set, in our research, we will consider the size of the training data set as a new factor in training machine learning systems for finding the best performing model.

Our research is tightly coupled with the Sonification Research [2], therefore, getting familiar with some of the terms such as Sonification, training dataset, and testing dataset, would be helpful for understanding the context of the research.

Sonification is the aural counterpart to visualization. However, instead of mapping data attribute values to visual structure as in the case of visualization, Sonification maps data attribute values to properties of sound [3].

Training dataset is a set of data used to discover potentially predictive relationships [4].

Testing dataset is a set of data used to assess the strength and utility of a predictive relationship [4].

We will use the same datasets and processes that were used in the Sonification research [2] for training the machine learning models. In the Sonification research, the researchers conducted a survey on many students to find out the best Sonification approach out of three different ones. The survey process was as follows: Each participant would listen to a small set of reference sounds for each of the Sonification approaches as a training phase, then the classification phase starts by making the participant listen to another set of sounds where the participant's job is to classify each of these sounds as closest to one of the reference sounds that was trained on. This process is a common case for human learning experience, where we get exposed to a small set of data that the brain processes for making future decisions. In order to make the Sonification data usable in our research, we need to transform the audio wave files that the survey taker was listening to into data files that can be input to the machine learning models for training and classification. The performance of these models will be measured based on their results compared to the actual results of the Sonification survey.

Chapter 2

1. Background

1.1. What Is Machine Learning?

Machine learning is subfield of computer science and artificial intelligence [5] that concentrates on developing self-learning programs that use pattern recognition and computational learning theory for finding data pattern on which the software rely for adapting their decision making strategy and improve their performance, without the need for explicit human interaction and coding.

Machine learning and data mining overlap in the sense that both focused on finding patterns in the data sets. However, instead of re-forming these data to be comprehensible by human, in the case of data mining, machine learning uses these patterns to grow and improve their performance and decision making strategies [6].

Also, machine learning is used for tasks that are not feasible through classical designing and programming explicit algorithms. Such applications include spam filtering, optical character recognition (OCR), search engines and computer vision [7].

1.2. What Are Different Types of Machine Learnings?

Machine learning can be categorized in two different methods [8]:

The first method classifies machine learning algorithms based on the nature of the data that is available for training the system as follows:

1) Supervised learning: the system is initialized with training data sets that are comprised of a system features data and their corresponding outputs, in order to allow the system to find general rules and patterns that can rely on for making future predictions and decisions.
2) Unsupervised learning: In unsupervised learning, the system is fed data without labels, then the system will process the data in order to find different structures that can be used for clustering and other forms of statistical correlation.
3) Reinforcement learning: The system interacts with dynamic environment in order to learn and gain more experience, in order to achieve a certain goal, such as a self-driving car program.

The second method classifies machine learning based on the output of the system:

1) Classification: In this type, the system is fed data that is classified into discrete classes; then the job of the system is to correctly classify new instances of data. This type falls under the supervised learning model.
2) Regression: The output of this system is similar to the classification category, but there will be unlimited categories, therefore the output will be continuous.
3) Clustering: the output of the system will be comprised of multiple groups that are not labeled. Unlike classification type where classes are known before hand, the groups will be inferred after the data processing. This type falls under the unsupervised learning model.
4) Density estimation: The output will be the distribution of the input in some space.

5) Dimensionality reduction simplifies inputs by mapping them into a lower-dimensional space.

1.3. What Are Available Machine Learning Algorithms?

Nowadays, there are a lot of machine learning algorithms; these algorithms can be grouped in many different ways. In this section we will follow the approach of grouping algorithms by their function similarity [9]. For our research purpose, we will get a brief introduction to some of the algorithm groups that were used for finding the best result algorithm.

The list is not meant to be exhaustive, but it will be sufficient for having a good idea about these different groups.

1.3.1. Bayesian Algorithms:

Bayesian methods are those that explicitly apply Bayes' Theorem for problems such as classification and regression. Bayes' Theorem employs classical probability for inferring and predicting the result of a system.

Bayes' Theorem is represented mathematically in the following formula:

$$P(H|E) = P(E|H) \,.\, P(H) \,/\, P(E)$$

Equation 1 - Bayes' Theorem

Where

| denotes a conditional probability (so that *(A|B)* means *A given B*).

- *H*: stands for any *hypothesis* whose probability may be affected by data (called *evidence* below). Often there are competing hypotheses, and the task is to determine which is the most probable.
- The *evidence E* corresponds to new data that were not used in computing the prior probability.
- *P(H):* the *prior probability*, is the estimate of the probability of the hypothesis *H before* the data *E*, the current evidence, is observed.
- *P(H|E),* the *posterior probability*, is the probability of *H given E*, i.e., *after E* is observed. This is what we want to know: the probability of a hypothesis *given* the observed evidence.
- *P(E|H)* is the probability of observing *E given H*. As a function of *E* with *H* fixed, this is the *likelihood* – it indicates the compatibility of the evidence with the given hypothesis. The likelihood function is a function of the evidence, *E,* while the posterior probability is a function of the hypothesis, *H*.
- *P(E)* is sometimes termed the marginal likelihood or "model evidence". This factor is the same for all possible hypotheses being considered (as is evident from the fact that the hypothesis *H* does not appear anywhere in the symbol, unlike for all the other factors), so this factor does not enter into determining the relative probabilities of different hypotheses.

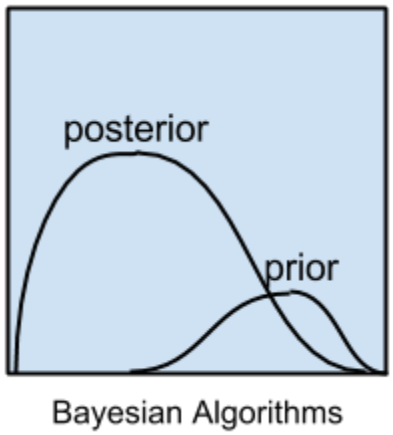

Figure 1 - Bayesian Algorithms

In other words, and as we can see in figure 1, the inferred result, which is the posterior probability $P(H|E)$, is a function of the prior probability of the hypothesis $P(H)$, and the likelihood for the observed data $P(E|H)$.

Therefore, to train a system that uses the Bayesian method, it requires big training data sets with a lot of instances for calculating accurate event probabilities. In addition, there is a need for a prior probability knowledge which is always a debatable subject. These needs do not align with our research objective of finding a machine learning model that requires small data set for training the system, therefore, we predicted that this group of algorithms will yield bad results.

The most popular Bayesian algorithm implementations are:

- *Naive Bayes*: Class for building and training the classifier assuming that all values of a system attribute are independent of the values of any other attributes, given the class variable, therefore it is a special case of BayesNet classifier.[1]
- *Bayesian Network (BN)*: Class that is using various search algorithms and quality measures for building and training *Bayesian Network* classifiers. This algorithm provides data structures (network structure, conditional probability distributions, etc.) and facilities common to Bayes Network learning algorithms like K2 and B for searching and finding the structures in the data.
- *Gaussian Naive Bayes*: Class for building *Naive Bayes* classifier that uses Gaussian distributions by default for numeric attributes. However, it has options to use supervised discretization or kernel density estimation.
- *Multinomial Naive Bayes*: Class for building and using a *Multinomial Naive Bayes* classifier, which is a specific instance of a *Naive Bayes* classifier that uses a multinomial distribution for each of the features.

1.3.2. Association Rule Learning Algorithms

Association rule learning is a method to extract rules that best explain observed relationships between variables in data.

Following the original definition by Agrawal et al. [11] the problem of association rule mining is defined as:

Let $I = \{i_1, i_2, i_3, \ldots, i_n\}$ be a set of n binary attributes called *items*.

[1] Machine learning algorithm summaries come from the Weka toolkit on-line documentation [10].

Let $D = \{t_1, t_2, t_3, \ldots, t_m\}$ be a set of transactions called the *database*.

Each *transaction* in *D* has a unique transaction ID and contains a subset of the items in *I*.

A *rule* is defined as an implication of the form:

X => Y

Where $X, Y \subseteq I$ and $X \cap Y = \emptyset$.

Every rule is composed by two different sets of items, also known as *item sets*, X and Y, where X is called *antecedent* or left-hand-side (LHS) and Y *consequent* or right-hand-side (RHS).

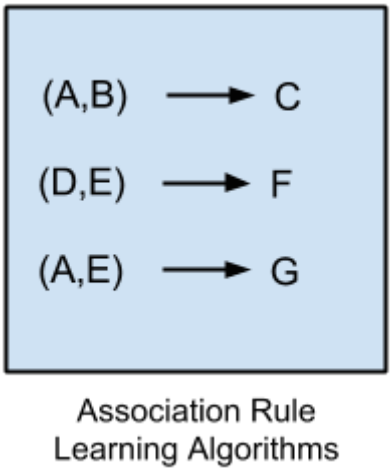

Figure 2 - Association Rule

The illustration in figure 2 shows three association rules. Each of them consist of multiple *antecedent*s on the left hand side that belong to the set of items *I,* and the *consequent* that is shown on the right hand side and belong to the set of transactions D.

The *consequents* are results of certain data pattern that are represented by the *antecedent*s' groupings.

The association rule method is used mainly for discovering regularities between data in large-scale transactions, so it can be exploited by an organization for data mining on large-scale of data. As a result, this machine learning method requires a big training set for extracting sufficient rules for more accurate prediction, an approach that does not align with our objective of finding a machine learning algorithm that can perform well with small training data set.

The most popular association rule learning algorithm implementations are:

- *OneR*: Class for building and using a 1R classifier; in other words, uses the minimum-error attribute for prediction, discretizing numeric attributes.
- *Apriori*: Class for implementing an Apriori-type algorithm. Iteratively reduces the minimum support until it finds the required number of rules with the given minimum confidence.
- *Eclat*: Classifier for building an *Eclat* algorithm. It is an internal classifier that is trained to predict the correct frame before the second classifier is trained to predict the actual class.

1.3.3. Decision Tree Algorithms

Decision tree methods, as the name indicates, use tree model for making decisions. These models use the attribute values as a determining factor for which path to follow in the decision tree, where each decision tree path leads to a specific prediction. Decision trees

can be used for classification and regression problems. Decision trees can be preferred choice because they are easy to understand, fast, and accurate.

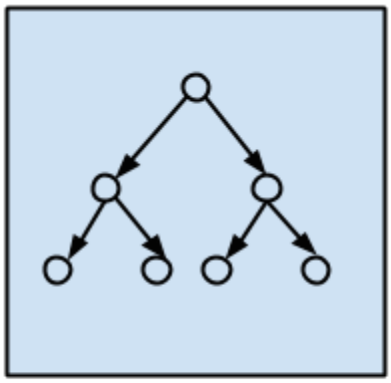

Figure 3 - Decision Tree

The illustration in figure 3 shows a decision tree where each of the nodes represents a system attribute, and each directed edge or arrow that connects two nodes represents a distinct value for the attribute that the edge started from. The leaves of the tree represent the classes or predictions of the model. As we can see from the illustration, each distinct path leads to a certain prediction that depends on the data attribute's values.

Decision trees do not necessarily need a big data set to generate an accurate model, especially, if the training data set contains instances with distinctive and dominant attributes, where the values of the dominant features can determine and distinguish the target class. Therefore, modeling a system with a dominant set of features using decision tree is feasible with a small training data set that contains sufficient and distinguished information about the dominant attributes. As a result, by satisfying the previously

mentioned requirement for a system, decision tree model will become a possible candidate for our research, where a small training data set can be sufficient for generating a well-trained model.

The most popular decision tree algorithms implementations are:

- *DecisionStump*: Class for building and using a decision stump. Usually used in conjunction with a boosting algorithm. It does regression based on mean-squared error or classification based on entropy. Missing values are treated as a separate value.
- *RandomTree*: Class for constructing a tree that considers K randomly chosen attributes at each node. Performs no pruning. Also has an option to allow estimation of class probabilities (or target mean in the regression case) based on a hold-out set (back-fitting).
- *RandomForest*: Class for constructing a forest of *RandomTree* classifiers.
- *HoeffdingTree*: A *Hoeffding* tree (VFDT) is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams, assuming that the distribution generating examples does not change over time. *Hoeffding* trees exploit the fact that a small sample can often be enough to choose an optimal splitting attribute.
- *J48*: Class for generating a pruned or unpruned C4.5 decision tree.
- *logistic model trees (LMT)*: Classifier for building *logistic model trees*, which are classification trees with logistic regression functions at the leaves.

- *REPTree*: Fast decision tree learner. Builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning (with back-fitting).

1.3.4. Instance-based Algorithms

In instance-based learning model, the training data is saved in memory to be used later in the classification process, where the newly classified instances are compared with the saved training data using a similarity function and similarity measures for determining the best match of the new instance with the saved data for making the prediction. Since instance-based learning model is storing the training data in the memory, it is also called memory-based learning model.

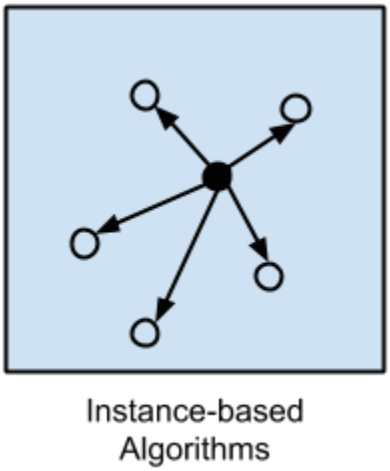

Figure 4 - Instance-Based Learning

The illustration in figure 4 shows a new instance of data that is represented by the filled circle which could be belong to a test dataset, and multiple hollow circles that represent the classes on which the system is trained. These classes are normally saved in memory in the case of instance-based learning. The arrows or edges that connect the filled circle

with the hollow circles in figure 4 represent the classification process, where the model classifies the new instance based upon a similarity function that measures the distance of the new instance from the classes that were trained on, then classify the instance accordingly. The similarity function varies based on the machine learning model that is used.

The training and classification process of the instance-based learning models is very similar to the human learning and classification process, where in the case of human, the brain get exposed to a set of data that get processed and saved in the brain forming the prior knowledge, which is the counterpart to the training classes that are saved in the memory in an instance-based learning model, Then the brain classifies new data instances by measuring the similarity distance between the new data and the prior knowledge using some similarity functions. This is identical to the classification process of instance-based learning models.

The similarity between both processes provides a strong hint about instance-based learning model being a strong candidate for mimicking human learning experience using small training datasets, which is the goal of our research. And it bears average correct results that are in the same range of human average correct results.

The most popular instance-based algorithms implementations are:

- *KStar*: is an instance-based classifier that is the class of a test instance and is based upon the class of those training instances similar to it, as determined by

some similarity function. It differs from other instance-based learners in that it uses an entropy-based distance function.

- *k-Nearest Neighbor (kNN)*: *K-nearest neighbor* classifier can select appropriate value of K based on cross-validation. Can also do distance weighting.
- *Locally Weighted Learning (LWL)*: *Locally weighted learning* classifier uses an instance-based algorithm to assign instance weights which are then used by a specified *Weighted Instances Handler*. It is capable of doing classification or regression.

In our research we used *KStar* instance-based learner as a representative of instance-based learning models.

1.3.5. Ensemble Algorithms

Ensemble machine learning model relies on combining multiple weaker machine learning models, which are created and trained separately using the same training data set, in order to create a strong model that uses certain function to make its overall prediction decisions.

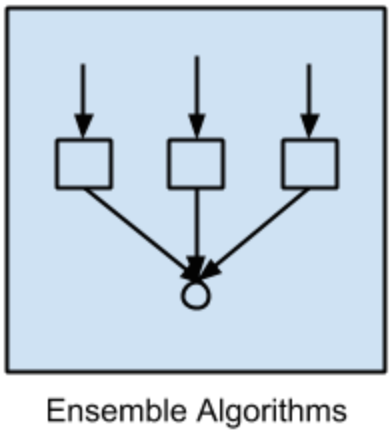

Figure 5 - Ensemble Algorithms

The illustration in figure 5 shows three boxes. Each represents a weak machine learning model that is trained separately using the same training dataset. The hollow circle in figure 5 represents the function of the ensemble learner that takes the output of every weak learner as an input; the arrows connecting the boxes with the circle represent this process; the function processes the data and produces the final prediction accordingly.

This method is a very powerful technique, and each type of the ensemble learners has its own functionalities.

The most popular ensemble algorithms implementations are:

- *AdaBoostM1 (Adaptive Boosting)*: Class for boosting a nominal class classifier using the Adaboost M1 method. Only nominal class problems can be tackled. Often dramatically improves performance, but sometimes over-fits.
- *Bootstrapped Aggregation (Bagging)*: Class for bagging a classifier to reduce variance. Can do classification and regression depending on the base learner.

- *Stacked Generalization (blending):* Combines several classifiers using the stacking method. Can do classification or regression.
- *Gradient Boosting Machines* (GBM): Meta classifier that enhances the performance of a regression base classifier. Each iteration fits a model to the residuals left by the classifier on the previous iteration. Prediction is accomplished by adding the predictions of each classifier.

In terms of our research, ensemble learners were very successful in finding similar or even better results than human results with small data sets. We tested with *AdaBoostM1*, *Bagging* and *RandomForest* algorithms. These algorithms can be used in conjunction with many other types of learning algorithms to improve their performance where the output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier [10].

1.4. What Is Machine Listening?

Machine Listening is a method for extracting meaningful information from audio signals using machine learning algorithms.

Machine Listening is an important concept for creating a system that is on par with humans. Since human is capable of interpreting audio signals using complex perceptual mechanism, a comparable system to human should be utilizing *Machine listening* technique for analyze the characteristics of audio signal and act upon it.

1.5. What Is Weka Tool?

Weka is a popular machine learning implementation tool, developed at the University Of Waikato, New Zealand. It is free software licensed under the GNU General Public License [10].

Weka includes a variety of tools for visualization, data analysis, and predictive modeling, transforming datasets, such as the algorithms for discretization. Weka can be interacted with using command line interface or graphical user interface. Weka provides you with the ability to preprocess a dataset, feed it into a learning scheme, and analyze the resulting classifier and its performance. Weka workbench has functionality that provides the user with the ability of creating, training, and testing many machine learning classifiers such as regression, clustering, association rule, decision tree, ensemble, and many more classifiers. All classifiers in Weka take their input in the form of single relational table in the ARFF format described in the next section.

Some machine learning practitioners consider Weka to be deficient because its training data sets must reside in main memory during learning. This requirement for memory residence is considered a weakness when dealing with very large datasets. However, because the focus of the present research is on small datasets amenable to fast human learning, Weka is a good fit for this research.

1.6. What is ARFF format?

ARFF (Attribute Relation File Format) is a file format created for standardizing datasets representation.

An ARFF file is an ASCII text file that describes a list of instances sharing a set of attributes. The file consists of a two-dimensional table of data (a relation), in which a column gives a single-valued attribute possessed by every instance in the data set, and a row is one instance, also known as record or tuple, in the data set. The ARFF file data is not normalized according to the rules of relational data; they are in first normal form. Attribute values of an ARFF file data can be unknown for a given instance.

1.6.1. ARFF File Syntax

Figure 6 shows an example of ARFF file syntax.

The lines that beginning with a % sign are comments. Following the comments at the beginning of the file are the name of the relation (weather) and a block defining the attributes (outlook, temperature, humidity, windy, play?). Nominal attributes are followed by the set of values they can take on, enclosed in curly braces. Values can include spaces; if so, they must be placed within quotation marks. Numeric values are followed by the keyword numeric.

For more information about ARFF format you can refer to Weka data mining book [10].

```
% ARFF file for the weather data with some numeric features
%
@relation weather

@attribute outlook { sunny, overcast, rainy }
@attribute temperature numeric
@attribute humidity numeric
@attribute windy { true, false }
@attribute play? { yes, no }

@data
%
% 14 instances
%
sunny, 85, 85, false, no
sunny, 80, 90, true, no
overcast, 83, 86, false, yes
rainy, 70, 96, false, yes
rainy, 68, 80, false, yes
rainy, 65, 70, true, no
overcast, 64, 65, true, yes
sunny, 72, 95, false, no
sunny, 69, 70, false, yes
rainy, 75, 80, false, yes
sunny, 75, 70, true, yes
overcast, 72, 90, true, yes
overcast, 81, 75, false, yes
rainy, 71, 91, true, no
```

Figure 6 - ARFF file for the weather data

Chapter 3

1. Scope of Data

1.1. Sonification Research Overview

Sonification research [2] was our main source of data that was used for training and testing the models in our project, thus, a detailed overview about the Sonification research will be useful in understanding the nature of the data sets, and it will provide a solid background for understanding the theoretical concept and the implementation goals of our research.

We will start by defining some terms to familiarize ourselves with the processes used.

Sonification is the aural counterpart to visualization. However, instead of mapping data attribute values to visual structure as in the case of visualization, Sonification maps data attribute values to properties of sound [3].

Parallel Coordinate Plotting: is a data visualization technique that provides means for graphing and exploring multidimensional relational datasets on a two-dimensional display [12].

The Sonification research supplying data to this study used Parallel Coordinate plotting along with some other algorithms for translating data sets into sounds to be used for perceptual exploration [2]. The similarity between the parallel coordinating plotting and time domain signal audio waveform, made the Sonification process possible.

The main objective of the Sonification research was to find the most effective Sonification approach among three other approaches (harmonic, melodic, and waveform) in terms of generating sounds that are distinguishable by listeners for classification purposes.

In the process of preparing the research data, researchers plotted a dataset from previous studies [13] [14] using a homegrown software tool for parallel coordinate plotting to generate the graph shown in figure 7. Each thin multi-segment path represents one record in the dataset, intersecting a vertical axis at the point of the value for that record's attribute. Also, the graph shows three thick paths and three Mid-thickness paths for the mean and population standard deviation, respectively, of three sets of instances.

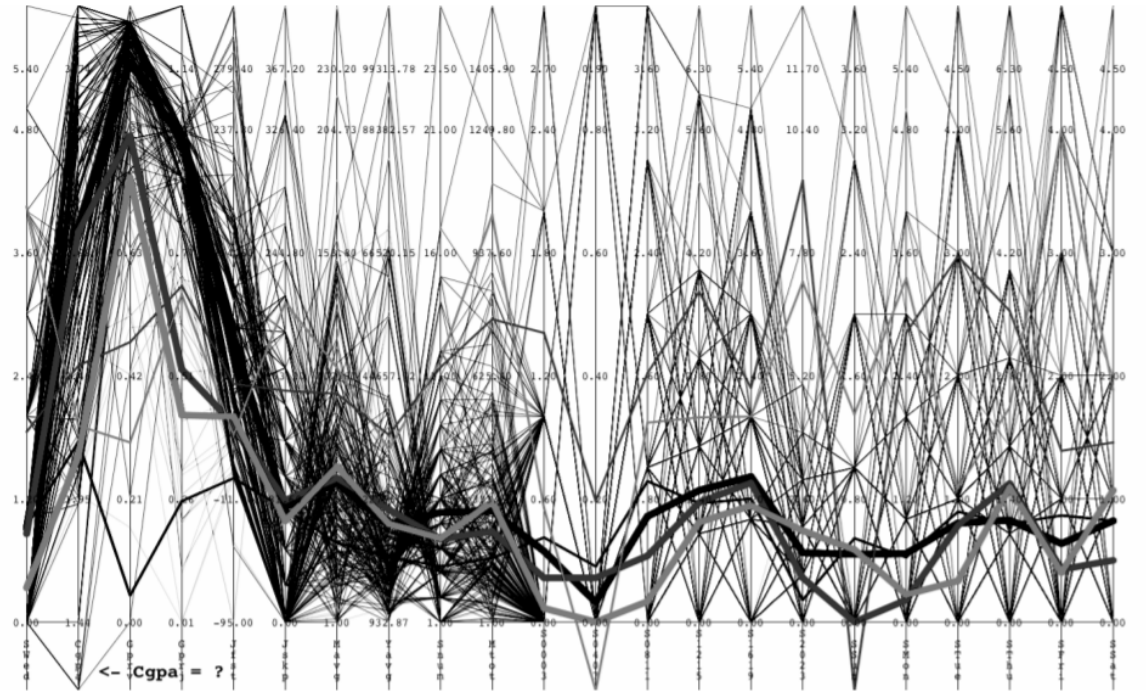

Figure 7 - Parallel Coordinates Plot of 22 of the 106 attributes in the Student Work Pattern

These mean paths were determined to be the reference instances based on the value ranges of certain attributes. The thick black path of mean values for class 0 instances are referred to as Reference Set 0, the thick medium-gray path of mean values for class 1 instances are referred to as Reference Set 1, and the thick light-gray path of mean values for class 2 instances are referred to as Reference Set 2. Partitioning of instances into one of Set 0, Set 1, or Set 2 is an application-specific classification specific to the dataset. The sonification study and the present study take the classification of each instance as a given. The focus of the sonification study is to convert each instance of classification into sound; the focus of the present study is to recognize each classification in these sounds by using machine listening, which is machine learning applied to patterns in sound.

Using the dataset instances and the reference dataset instances that were inferred from the parallel coordinate plotting graph, sonification generates sounds for each data record in the original dataset and for each of the three reference data instances respectively. Researchers repeated the Sonification process using three Sonification approaches, harmonic, melodic, and waveform. All three algorithms generate sounds in some relation to a baseline frequency, which is 220 Hz.

The *Harmonic Sonification Algorithm* generates simultaneous notes in a chord and maintains two tables of frequencies, *Sweet* Notes and *Sour* Notes, that are in-scale and out-of-scale respectively for the baseline note of 220 Hz. *Sweet* and *Sour* Note refers to 2 different types of notes based on the frequency range that they belong to; so the generated sound is distinctive for each type. A high Sweet value, which comes from a close proximity to the Reference Set 0 (mean) value for an attribute being sonified, gives a

strong consonant (sweet) sound. A high Sour value, which comes from a distant proximity to the Reference Set 0 value for an attribute being sonified, gives a strong dissonant (sour) sound. Each sonified attribute contributes a mix of sweetness and sourness. The *Sweet* and *Sour* notes for each of the Sonification algorithms are generated using a helper algorithm that extracts two numeric values for the most distinguishing attributes in the dataset of mean values and the dataset of the data records. Then it computes the difference between an attribute being sonified and that attribute value for Reference Set 0, and based on the result, the algorithm assigns a *Sweet* weight and a *Sour* weight to that attribute [2]. Figure 8 shows the *Sweet* and *Sour* values as a function of attribute standard deviation. These Sweet and Sour strengths applied to each attribute's distance-from-Reference-Set-0-mean is a non-linear step function at the boundaries of one sample standard deviation and two sample standard deviations from the Reference Set 0 mean value. "The most distinguishing attributes in the dataset" are 5 attributes that most clearly distinguish the closest Reference Set for the majority of data records in this dataset.

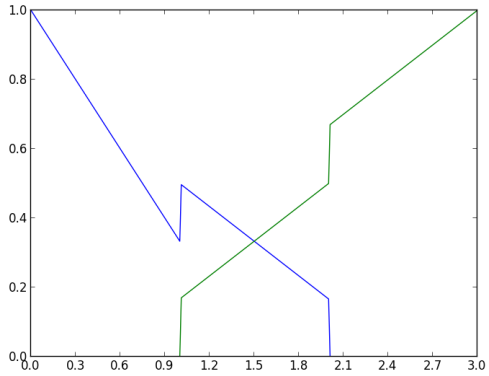

Figure 8 - Sweet and Sour values as a function of attribute standard deviation

The *Melodic Sonification Algorithm* (technically, a set of melodic intervals) generates note frequencies, amplitudes, and waveforms that are identical to the *Harmonic Sonification* approach, but it generates them to play only one attribute's *Sweet* and *Sour* notes at a time, sequencing them across a 2 second duration.

The *Waveform Sonification Algorithm* is substantially different from the other two; the five most distinguishing attributes of Figure 7 are sorted to approximate a triangular waveform for the Reference Set 0 mean values.

After the completion of the Sonification processes, researchers conducted a survey where each listener classifies a data record's sound as being closest to the reference sound of Set 0, Set 1 or Set 2, thereby classifying the record as belonging to Set 0 or 1 or 2. The participant classifies a representative subset of the dataset that was sonified repeatedly for each of the three Sonification algorithms. At the end of the research, researchers were able to find the most effective approach based on the survey result data.

1.2. Sonic Survey Results

The first sonic survey was conducted in fall of 2015 using the formerly mentioned algorithms. The results were in favor of the *Waveform Sonification Algorithm* as we can see in table 1.

Sonification (Fall 2015)	Category	Mean correct responses
Harmonic	All 3 sets	55.8%
Melodic	All 3 sets	55.4%
Waveform	All 3 sets	61.4%

Table 1 - fall 2015 Sonification survey results

The result was a surprise to the research leader Dr. Parson, who anticipated that *Waveform Sonification Algorithm* would be the worst Sonification approach among the three formerly mentioned Sonification algorithms. This result encouraged the researchers to develop new algorithms called *WaveformDouble*, *WaveformFourThirds*, and *WaveformOnePt95* that are variations of the *Waveform Sonification Algorithm.* The details of these algorithms are explained later in chapter 6. The latter Sonification algorithms were surveyed in the spring of 2016 where the results are shown in Table 2.

Sonification (Spring 2016)	Category	Mean correct responses
WaveformDouble	All 3 sets	67.8%
WaveformFourThirds	All 3 sets	65.8%
WaveformOnePt95	All 3 sets	67.6%

Table 2 - spring 2016 Sonification Survey Results

In our research we used the spring of 2016 sonic survey results to compare with our machine learning models results for evaluating the models performance.

1.3. Nature of Data

Now that we have a good overview of the Sonification research and its data, we can examine the benefits of using it in our research:

First, the *training data set*, which comprises of the three reference sounds, is small, which makes it conform to our proposal of considering a small data set as new factor for training the models.

Second, the *training data set* is different than the *test data set*, which is the most common case in human learning process, where the human will make a prediction by seeing completely new data, while using previous knowledge and seen data as reference point.

Third, having two sets of data, one for training and another for testing, will help us in preventing the over-fitting problem. The reason for that is, we are predicting the accuracy of a classifier by using a testing data set that is completely different than the training data set. Thus, finding a classifier with good results, means that the classifier was trained well and not over-fitted.

Fourth, the survey results are available to compare with the classifiers results in order to evaluate their performance, whether it is giving similar, better or worse performance than humans' performance.

1.4. Data Transformation

A data transformation step was required to make the data usable in our project, because the Sonification data sets are in the WAV audio file format. Therefore we need to transform these WAV files into data files that are consumable as an input by the machine learning models.

In order to get a better understanding of the transformation process, we will start by defining a couple of terms:

1. *ChucK* is a programming language for real-time sound synthesis and music creation. *ChucK* presents a unique time-based, concurrent programming model that's precise and expressive, dynamic control rates, and the ability to add and modify code on-the-fly [15].
2. *Spectral centroid* is a measure used in digital signal processing to characterize a spectrum. It indicates where the "center of mass" of the spectrum is. Perceptually, it has a robust connection with the impression of "brightness" of a sound [16].
3. *Audio power* is the electrical power transferred from an audio amplifier to a loudspeaker, measured in watts.
4. *Roll-off* is the steepness of a transmission function with frequency, particularly in electrical network analysis, and most especially in connection with filter circuits in the transition between a passband and a stopband.
5. *Fast Fourier transform (FFT)* is an algorithm that computes the Fourier transform of a sequence, or its inverse. Fourier analysis converts a signal from its original

domain (often time or space) to a representation in the frequency domain and vice versa.

In order to do the transformation process, Dr. Parson created a *ChucK* program that analyzes the WAV files from the survey. It analyzes each WAV signal for a second and extracts the following data:

- Centroid
- RMS (root-mean-squared) power
- 25% Roll-off point – frequency at which 25% of the energy has dissipated
- 50% Roll-off point – frequency at which 50% of the energy has dissipated
- 75% Roll-off point – frequency at which 50% of the energy has dissipated
- 64-window FFT where FFT is the histogram of frequency strengths in the signal and it can be shown in Figure 9 for all of the three Reference sets of the waveform Sonification approach.

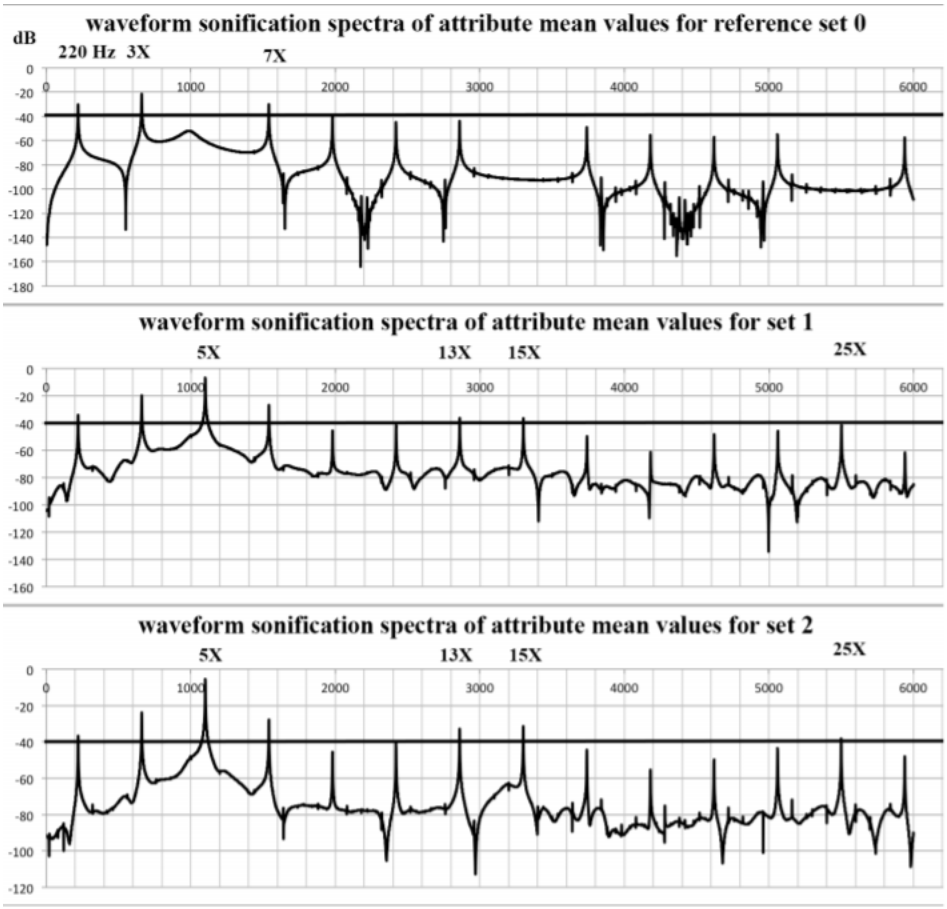

Figure 9- Filtered waveform frequency-domain spectra for Reference Set 0, Set 1, and Set 2 mean values

Then we pass the *ChucK* program output to python data-reformatting scripts for assembling the data and generating ARFF files that are consumable by Weka machine learning classifiers.

Chapter 4

1. Design and Implementation

1.1. Data Dissection

As mentioned before, the goal is to find machine learning algorithms that are capable of mimicking human leaning experience using the same training data sets and producing results that are comparable to the human correctness average results.

Now that we are familiar with the Sonification research data and transformation process, we will dig further into the structure of the datasets that we will be using for our research. After processing the WAV files, which were produced in the Sonification research and used in the survey, through the transformation steps that was explained earlier, we end up with three groups of datasets, each group corresponding to one of Sonification approaches and comprising 50 datasets , where a dataset is an ARFF file that comprises 39 instances, each instance being the data that was generated in the transformation process of one WAV file used in the Sonification survey for one student. There are 39 instances because the survey used 39 Sonified WAV files that are split evenly among the three classification reference Sounds. In another word, there are 13 instances for each of the classes and they are presented randomly in one survey. Therefore, an ARFF file in a dataset represents one student's worth of survey data that can be entered into a machine learning model for classification. In addition, for each of the three groups there is one training ARFF file that contains the training instances that are generated from the

transformation process of the reference sounds WAV files. Thus, each of the groups has its own training dataset that corresponds to the Sonification approach that the group is representing.

1.2. Weka Tool Limitation

Knowing the structure of the datasets that we are working with allows us to define the steps that are needed to find the best performing machine learning model as follow:

- Train the model using an ARFF training file that corresponds to one of the Sonification approaches, where the ARFF file is the representation of the training dataset that was used for training the students in the sonic survey for that particular Sonification approach.
- Input the test data to the machine learning model for classification, where the test data is the set of all ARFF files that were generated from the WAV files of that Sonification approach and used in every student survey.
- Calculate the average correct result and standard deviation for the results that were produced by the model using each of the ARFF files, and then calculate the overall average correct results of all averages.
- Repeat the previous steps for every Sonification approach.
- Repeat the steps for every possible machine learning model and its option variations.

Weka tool is very suitable choice for performing the previous steps, because it provides the implementations for all the selected machine learning models, and it provides great capabilities for generating the needed results, however, it lacks the scalability functionality that is needed for completing the models' evaluation process. In our research, we need to run all the datasets through the permutation of every machine learning model with its options variations, then calculate the results as mentioned in the previous section for every machine learning model, then calculate the overall result for all models. Running all these experiments and extracting all the needed results is almost impossible without the help of an automated process, therefore, we designed and implemented a new tool that extends the capabilities of the Weka tool and allows us to get the results that we need in an easy and efficient automated process. This tool is called *Machine Learning Evaluator Tool*.

1.3. What is The Machine Learning Evaluator Tool?

The *Machine Learning Evaluator Tool* is a Java application that uses Weka as a library for extending and scaling *Weka Tool* capabilities. The *Machine Learning Evaluator Tool* is capable of:

- Navigating through specified directories for loading training and testing datasets.
- Training data models using training datasets.
- Running test datasets against all possible permutations of classifiers and their options variations.
- Extracting results for every classifier and dataset run.

- Calculating the overall result for each classifier for every option variation.
- Filtering classifiers based on their average result's correct percentages.
- Saving detailed, summarized, and filtered results' reports in csv and text files formats.

1.4. Machine Learning Evaluator Tool Design and Implementation:

The *Machine Learning Evaluator Tool* is designed in a modular way that makes it easy to maintain and scale, where each class is self-contained and responsible for achieving one job that is not shared with any other classes.

In addition, adding new functionality such as adding a new machine learning evaluator is made simple and can be done by adding one class only, that extends the ClassifierEvaluator.java class and implements run() method.

Also, classifiers configurations are consolidated in one class for ease of access in all classes.

The *Machine Learning Evaluator Tool* UML Diagram is shown in figure 10

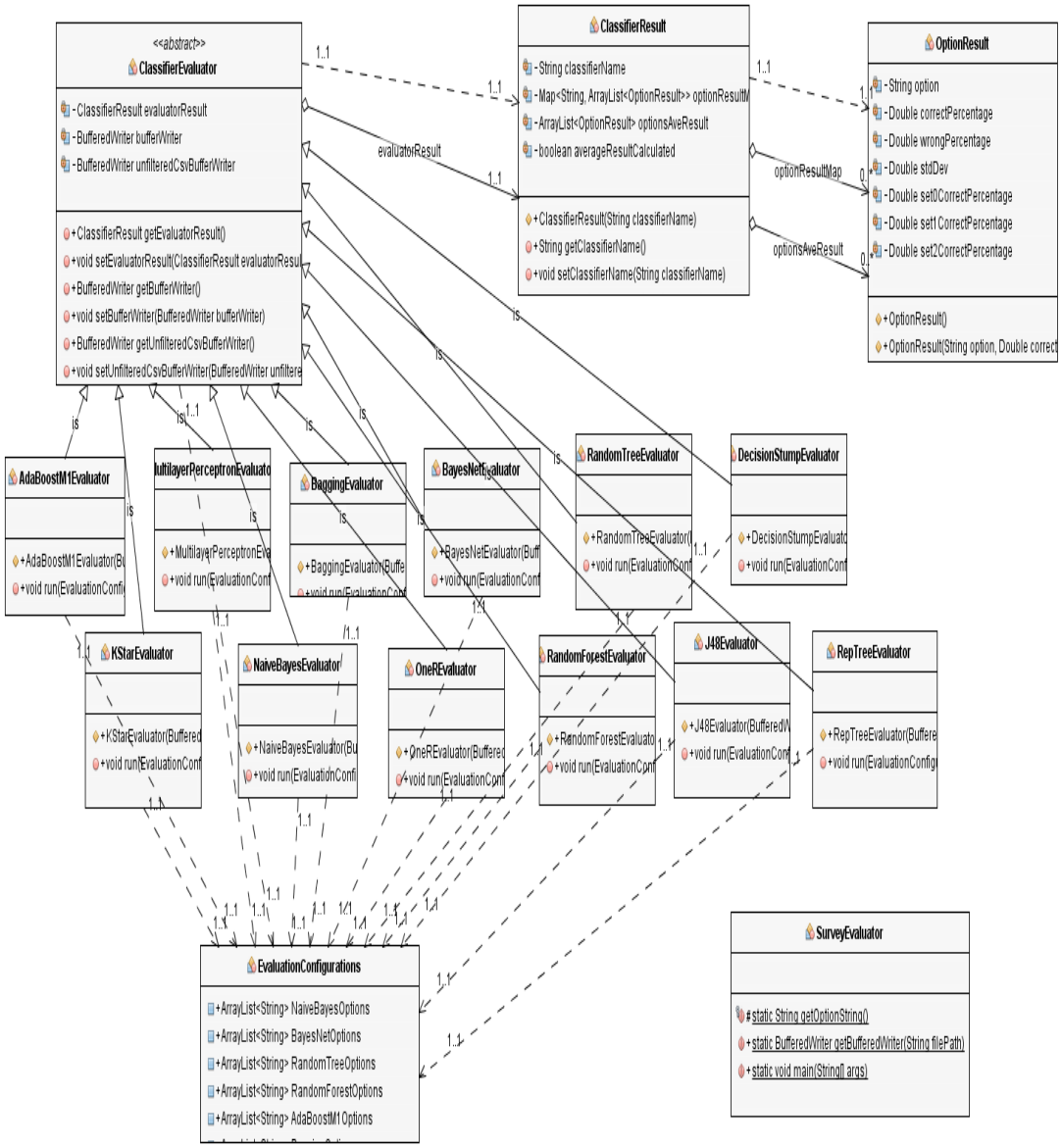

Figure 10 - Machine Learning Evaluator UML Diagram

The *Machine Learning Evaluator Tool* is comprised of the following Classes:

1. *SurveyEvaluator.java*: Is the main class that parses the command line arguments, and runs the machine learning evaluators accordingly.
2. *ClassifierEvaluator.java*: Is an abstract class to be extended by every machine learning evaluator class. It implements the functions that are shared between all evaluators and specifies the abstract functions that need to be implemented on every machine learning evaluator class. The main function in this class is the *execute()* method that contains the logic for applying a machine learning classifier using all its option variations on a specific dataset.
3. *ClassifierResult.java*: As the name implies, this class contains the result of using a classifier with all its option variations. It contains a hash map that maps between each classifier option and the corresponding results, where the results consist of every dataset run using that particular option. Also it contains the average result and standard deviation for each option across all datasets results.
4. *EvaluationConfigurations.java*: This class contains all the information that is needed for running the classifier's evaluators such as classifiers' options, and the information needed for storing the results.
5. *OptionResult.java*: is a helper class that is used by *ClassifierResult.java* class. It carries the result of running one dataset on a particular classifier. The result is comprised of the average correct result, average wrong result and the standard deviation of running that particular dataset.

6. *AdaBoostM1Evaluator.java*: Is a driver class that runs the evaluation for AdaBoostM1 classifier against the provided datasets.
7. *BaggingEvaluator.java*: Is a driver class that runs the evaluation for Bagging classifier against the provided datasets.
8. *BayesNetEvaluator.java*: Is a driver class that runs the evaluation for BayesNet classifier against the provided datasets.
9. *DecisionStumpEvaluator.java*: Is a driver class that runs the evaluation for DecisionStump classifier against the provided datasets.
10. *J48Evaluator.java*: Is a driver class that runs the evaluation for J48 classifier against the provided datasets.
11. *KStarEvaluator.java*: Is a driver class that runs the evaluation for KStar classifier against the provided datasets.
12. *MultilayerPerceptronEvaluator.java*: Is a driver class that runs the evaluation for MultilayerPerceptron classifier against the provided datasets.
13. *NaiveBayesEvaluator.java*: Is a driver class that runs the evaluation for NaiveBayes classifier against the provided datasets.
14. *OneREvaluator.java*: Is a driver class that runs the evaluation for OneR classifier against the provided datasets.
15. *RandomForestEvaluator.java*: Is a driver class that runs the evaluation for RandomForest classifier against the provided datasets.
16. *RandomTreeEvaluator.java*: Is a driver class that runs the evaluation for RandomTree classifier against the provided datasets.

17. *RepTreeEvaluator.java*: Is a driver class that runs the evaluation for RepTree classifier against the provided datasets.

After compiling the code, the tool can be run using the command line interface using the following options:

- *-h* :outputs help information.
- *-b* <base directory path>: sets the base directory path where training and testing datasets are located.
- *-tf* <training file name> : sets the training file name.
- *-Tf* <test file name> : sets the test file name, where % is used as wild character to be replaced with sequence numbers in the application.
- *-rn* <number of sets>: specifies number of records.
- *-c* <classifier name>: specifies which classifier to run by name.
- *-rac* <minimum avg. correct percentage > <maximum avg. correct percentage >: filters classifiers' average results by provided average correct result range.
- *-raw* <minimum avg. wrong percentage > <maximum wrong percentage >: filters classifiers' average results by provided average wrong result range.
- *-resD* <result directory path>: specifies directory path for storing the results.

The following is a command line example for running the datasets with all classifiers:

java -jar SurveyEvaluator.jar -b <base directory path where training and testing data are available> -tf waveformDoubleTraining.una.arff -Tf waveformDoubleTest-

%.una.arff -rn <number of sets> -rac <minimum correct range> <maximum correct range> –resD <result directory path>

Using the Machine Learning Evaluator tool made our research possible, in the sense that testing all different permutations of machine learning classifiers with their option variations was impossible without the *Machine Learning Evaluator Tool*. In addition, the *Machine Learning Evaluator Tool* can be very useful not only for this particular research, but also for future research work evaluating machine learning models performances with any sets of data.

Chapter 5

1. Data Analysis

1.1. Machine Learning Evaluation Process Summary

A full machine learning model evaluation process can be shown in the workflow in figure 11.

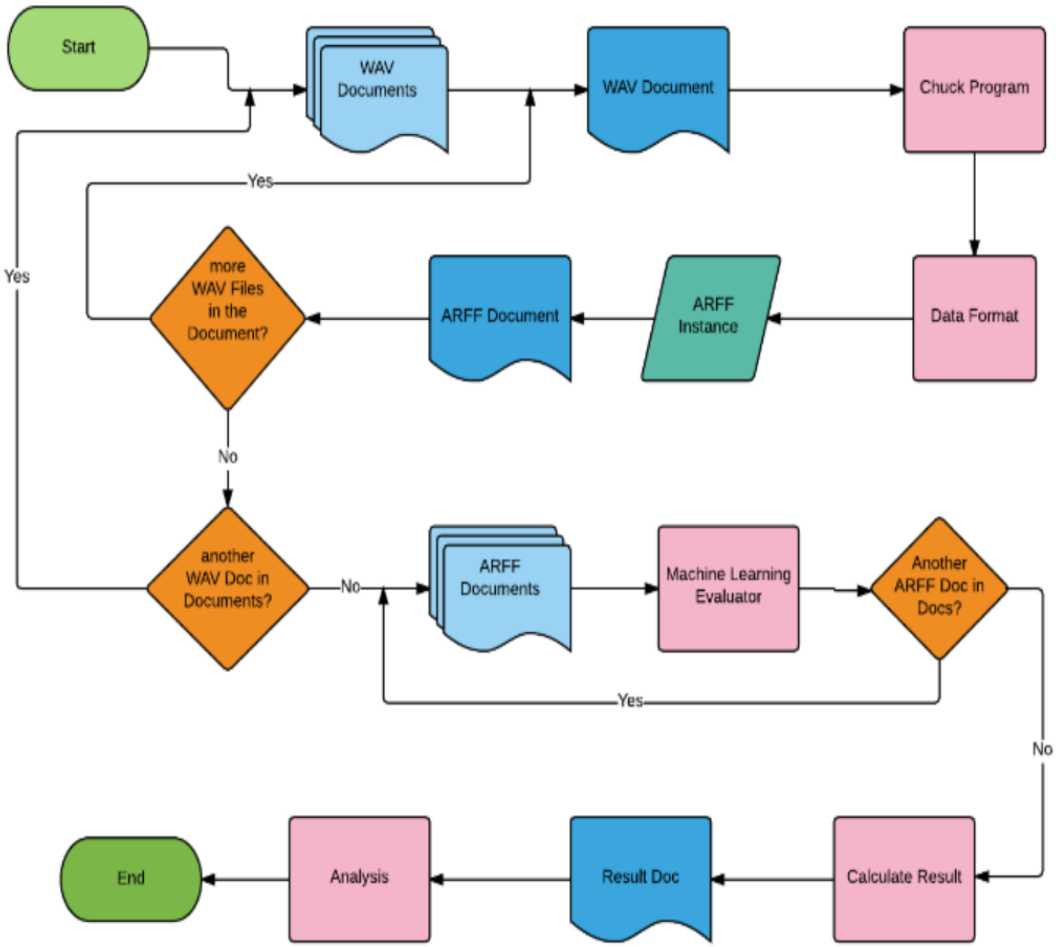

Figure 11 - Machine Learning Evaluation Process flow chart

The WAV files that are used in the Sonification survey are passed into transformation and data formatting steps to produce the ARFF data files, which in turn are passed into the *Machine Learning Evaluator Tool* that trains the machine learning model and process the instances for classification. Then it calculates the results to produce the average correct classification result. This process is repeated for every machine learning model option variation and for every Sonification approach.

1.2. Machine Learning Evaluation Results

The result data of the Sonification survey that was conducted in the spring of 2016 can be shown in table 2.

By looking at the results in table 2, we can see that the human average correct results for the three types of Sonification approaches are in the range between 64% - 68 %. Therefore, we will be looking mainly for the classifiers that can achieve close to this range across the three Sonification approaches.

After applying the process that was mentioned in the previous section on all ARFF files, we get the results that are shown in table 3, where this table consists of five columns.

- First column is the classifier name, which corresponds to the machine learning algorithm that was used for classification.
- Second column is the option that was used while running the classifier. *Weka Tool* provides each classifier with many options to use while running the algorithm, which can significantly affect the performance of the classifier.

- Third column is the average correct result for *WaveformDouble* Sonification approach.
- Fourth column is the average correct results for *WaveformFourThirds* Sonification approach.
- Fifth column is the average correct results for *WaveformOnePt95*Sonification approach.

Classifier	**Option**	**Correct result Avg. Double**	**Correct result Avg. Four Third**	**Correct result Avg. 1.95**
NaiveBayes	no-arg	58.15384615	60.1025641	66.66666667
NaiveBayes	K	71.28205128	57.8974359	70.46153846
BayesNet	-D -Q weka.classifiers.bayes.net.search.local.HillClimber -- -P 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5	33.33333333	33.33333333	33.33333333
BayesNet	-D -Q weka.classifiers.bayes.net.search.local.TabuSearch -- -L 5 -U 10 -P 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5	33.33333333	33.33333333	33.33333333

BayesNet	-D -Q weka.classifiers.bayes.net.search.local.RepeatedHillClimber -- -U 10 -A 1 -P 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5	33.33333333	33.33333333	33.33333333
BayesNet	-D -Q weka.classifiers.bayes.net.search.local.LAGDHillClimber -- -L 2 -G 5 -P 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5	33.33333333	33.33333333	33.33333333
BayesNet	-D -Q weka.classifiers.bayes.net.search.local.SimulatedAnnealing -- -A 10.0 -U 10000 -D 0.999 -R 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5	33.33333333	33.33333333	33.33333333
BayesNet	-D -Q weka.classifiers.bayes.net.search.local.K2 -- -P 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5	33.33333333	33.33333333	33.33333333
RandomTree	-K 9 -M 1.0 -V 0.001 -S 0	66.05128205	59.12820513	69.28205128
RandomTree	-K 9 -M 1.0 -V 0.001 -S 1	64.82051282	55.43589744	57.28205128
RandomTree	-K 8 -M 1.0 -V 0.001 -S 9	76.51282051	55.64102564	56.30769231

RandomTree	-K 0 -M 1.0 -V 0.001 -S 0	58.92307692	58.76923077	69.28205128
RandomTree	-K 9 -M 1.0 -V 0.001 -S 8	52.05128205	55.12820513	59.12820513
RandomTree	-K 6 -M 1.0 -V 0.001 -S 1	57.12820513	55.38461538	62.87179487
RandomTree	-K 9 -M 1.0 -V 0.001 -S 9	69.69230769	55.38461538	64
RandomTree	-K 6 -M 1.0 -V 0.001 -S 2	67.43589744	53.69230769	68.92307692
RandomTree	-K 9 -M 1.0 -V 0.001 -S 6	59.74358974	56.15384615	59.33333333
RandomTree	-K 9 -M 1.0 -V 0.001 -S 7	69.48717949	64.61538462	63.69230769
RandomTree	-K 6 -M 1.0 -V 0.001 -S 0	61.48717949	60.25641026	69.28205128
RandomTree	-K 9 -M 1.0 -V 0.001 -S 4	61.94871795	67.17948718	54.20512821
RandomTree	-K 6 -M 1.0 -V 0.001 -S 5	60.30769231	60.56410256	52.92307692
RandomTree	-K 9 -M 1.0 -V 0.001 -S 5	63.64102564	58	57.94871795
RandomTree	-K 6 -M 1.0 -V 0.001 -S 6	64.05128205	56.76923077	60.05128205
RandomTree	-K 9 -M 1.0 -V 0.001 -S 2	51.02564103	55.43589744	68.30769231
RandomTree	-K 6 -M 1.0 -V 0.001 -S 3	62.35897436	72.82051282	52.1025641
RandomTree	-K 9 -M 1.0 -V 0.001 -S 3	52.05128205	63.33333333	53.74358974
RandomTree	-K 6 -M 1.0 -V 0.001 -S 4	61.94871795	56.92307692	62.71794872
RandomTree	-K 4 -M 1.0 -V 0.001 -S 10	54.61538462	63.33333333	47.58974359
RandomTree	-K 3 -M 1.0 -V 0.001 -S 10	52.05128205	61.84615385	52.76923077
RandomTree	-K 10 -M 1.0 -V 0.001 -S 4	69.84615385	59.79487179	55.74358974
RandomTree	-K 10 -M 1.0 -V 0.001 -S 5	67.84615385	58	55.53846154

RandomTree	-K 2 -M 1.0 -V 0.001 -S 10	54.61538462	52.92307692	51.02564103
RandomTree	-K 10 -M 1.0 -V 0.001 -S 6	60.30769231	67.43589744	53.58974359
RandomTree	-K 10 -M 1.0 -V 0.001 -S 7	70.05128205	51.02564103	56
RandomTree	-K 10 -M 1.0 -V 0.001 -S 8	54.61538462	55.12820513	65.02564103
RandomTree	-K 10 -M 1.0 -V 0.001 -S 9	67.64102564	52.25641026	58.87179487
RandomTree	-K 10 -M 1.0 -V 0.001 -S 10	52.05128205	63.8974359	55.53846154
RandomTree	-K 5 -M 1.0 -V 0.001 -S 0	58.35897436	47.23076923	66.30769231
RandomTree	-K 6 -M 1.0 -V 0.001 -S 8	54.61538462	51.02564103	65.02564103
RandomTree	-K 5 -M 1.0 -V 0.001 -S 1	57.12820513	55.53846154	63.69230769
RandomTree	-K 6 -M 1.0 -V 0.001 -S 7	60.92307692	64.76923077	60.41025641
RandomTree	-K 0 -M 1.0 -V 0.001 -S 9	76.51282051	52.25641026	58.87179487
RandomTree	-K 6 -M 1.0 -V 0.001 -S 9	74	54.05128205	64
RandomTree	-K 10 -M 1.0 -V 0.001 -S 1	60.25641026	55.38461538	58.92307692
RandomTree	-K 5 -M 1.0 -V 0.001 -S 4	71.64102564	58	62.71794872
RandomTree	-K 10 -M 1.0 -V 0.001 -S 0	64.05128205	59.43589744	66.41025641
RandomTree	-K 5 -M 1.0 -V 0.001 -S 5	65.8974359	60.56410256	50.61538462
RandomTree	-K 10 -M 1.0 -V 0.001 -S 3	49.48717949	67.69230769	52.1025641
RandomTree	-K 5 -M 1.0 -V 0.001 -S 2	68.97435897	51.79487179	56
RandomTree	-K 8 -M 1.0 -V 0.001 -S 0	60.92307692	59.12820513	69.28205128
RandomTree	-K 10 -M 1.0 -V 0.001 -S 2	51.02564103	55.43589744	60.66666667

RandomTree	-K 5 -M 1.0 -V 0.001 -S 3	57.23076923	67.69230769	53.84615385
RandomTree	-K 0 -M 1.0 -V 0.001 -S 3	52.05128205	57.43589744	56.51282051
RandomTree	-K 8 -M 1.0 -V 0.001 -S 2	56.15384615	55.38461538	64.97435897
RandomTree	-K 0 -M 1.0 -V 0.001 -S 4	56.82051282	67.48717949	63.33333333
RandomTree	-K 8 -M 1.0 -V 0.001 -S 1	70.51282051	61.38461538	59.12820513
RandomTree	-K 0 -M 1.0 -V 0.001 -S 1	57.12820513	61.23076923	53.23076923
RandomTree	-K 8 -M 1.0 -V 0.001 -S 4	71.07692308	68.25641026	63.33333333
RandomTree	-K 0 -M 1.0 -V 0.001 -S 2	62.1025641	48.76923077	68.30769231
RandomTree	-K 6 -M 1.0 -V 0.001 -S 10	49.48717949	63.33333333	53.84615385
RandomTree	-K 8 -M 1.0 -V 0.001 -S 3	52.05128205	57.43589744	52.1025641
RandomTree	-K 0 -M 1.0 -V 0.001 -S 7	69.48717949	63.8974359	55.28205128
RandomTree	-K 8 -M 1.0 -V 0.001 -S 6	64.41025641	63.8974359	63.07692308
RandomTree	-K 0 -M 1.0 -V 0.001 -S 8	52.05128205	51.28205128	59.8974359
RandomTree	-K 8 -M 1.0 -V 0.001 -S 5	56.35897436	60.56410256	52.1025641
RandomTree	-K 8 -M 1.0 -V 0.001 -S 8	54.61538462	51.79487179	65.02564103
RandomTree	-K 0 -M 1.0 -V 0.001 -S 5	62.15384615	58	50.61538462
RandomTree	-K 0 -M 1.0 -V 0.001 -S 6	63.74358974	63.8974359	60.05128205
RandomTree	-K 5 -M 1.0 -V 0.001 -S 10	49.48717949	61.84615385	52.15384615
RandomTree	-K 8 -M 1.0 -V 0.001 -S 7	69.64102564	63.8974359	55.28205128
RandomTree	-K 3 -M 1.0 -V 0.001 -S 7	63.84615385	52.61538462	55.43589744

RandomTree	-K 3 -M 1.0 -V 0.001 -S 6	61.23076923	56.76923077	60.61538462
RandomTree	-K 3 -M 1.0 -V 0.001 -S 5	64.20512821	56.92307692	56.25641026
RandomTree	-K 3 -M 1.0 -V 0.001 -S 4	70.51282051	60.56410256	63.79487179
RandomTree	-K 5 -M 1.0 -V 0.001 -S 7	62.56410256	63.8974359	56.1025641
RandomTree	-K 5 -M 1.0 -V 0.001 -S 6	70.05128205	58.61538462	59.84615385
RandomTree	-K 3 -M 1.0 -V 0.001 -S 9	74	60.35897436	58.61538462
RandomTree	-K 5 -M 1.0 -V 0.001 -S 9	66.25641026	65.64102564	57.12820513
RandomTree	-K 3 -M 1.0 -V 0.001 -S 8	54.61538462	52.05128205	62.46153846
RandomTree	-K 5 -M 1.0 -V 0.001 -S 8	54.61538462	56.87179487	64.82051282
RandomTree	-K 3 -M 1.0 -V 0.001 -S 2	60.51282051	51.48717949	77.8974359
RandomTree	-K 3 -M 1.0 -V 0.001 -S 3	58.92307692	72.82051282	57.94871795
RandomTree	-K 3 -M 1.0 -V 0.001 -S 0	61.48717949	63.07692308	70.41025641
RandomTree	-K 3 -M 1.0 -V 0.001 -S 1	68.82051282	59.12820513	63.69230769
RandomTree	-K 0 -M 1.0 -V 0.001 -S 10	54.61538462	57.38461538	55.23076923
RandomTree	-K 7 -M 1.0 -V 0.001 -S 0	58.92307692	58.76923077	69.28205128
RandomTree	-K 7 -M 1.0 -V 0.001 -S 1	57.12820513	61.23076923	53.23076923
RandomTree	-K 7 -M 1.0 -V 0.001 -S 2	62.1025641	48.76923077	68.30769231
RandomTree	-K 7 -M 1.0 -V 0.001 -S 3	52.05128205	57.43589744	56.51282051
RandomTree	-K 7 -M 1.0 -V 0.001 -S 4	56.82051282	67.48717949	63.33333333
RandomTree	-K 7 -M 1.0 -V 0.001 -S 5	62.15384615	58	50.61538462

RandomTree	-K 7 -M 1.0 -V 0.001 -S 6	63.74358974	63.8974359	60.05128205
RandomTree	-K 7 -M 1.0 -V 0.001 -S 7	69.48717949	63.8974359	55.28205128
RandomTree	-K 7 -M 1.0 -V 0.001 -S 8	52.05128205	51.28205128	59.8974359
RandomTree	-K 7 -M 1.0 -V 0.001 -S 9	76.51282051	52.25641026	58.87179487
RandomTree	-K 1 -M 1.0 -V 0.001 -S 10	54.61538462	57.53846154	51.79487179
RandomTree	-K 8 -M 1.0 -V 0.001 -S 10	57.17948718	52.20512821	57.43589744
RandomTree	-K 7 -M 1.0 -V 0.001 -S 10	54.61538462	57.38461538	55.23076923
RandomTree	-K 1 -M 1.0 -V 0.001 -S 1	64.82051282	57.23076923	65.43589744
RandomTree	-K 1 -M 1.0 -V 0.001 -S 0	57.79487179	66.92307692	62.87179487
RandomTree	-K 2 -M 1.0 -V 0.001 -S 6	52.15384615	60.41025641	61.43589744
RandomTree	-K 2 -M 1.0 -V 0.001 -S 5	63.74358974	62	58.71794872
RandomTree	-K 2 -M 1.0 -V 0.001 -S 4	56.82051282	58	61.43589744
RandomTree	-K 2 -M 1.0 -V 0.001 -S 3	61.48717949	72.82051282	66.05128205
RandomTree	-K 2 -M 1.0 -V 0.001 -S 9	58.35897436	58.82051282	58.61538462
RandomTree	-K 2 -M 1.0 -V 0.001 -S 8	65.48717949	49.28205128	62.82051282
RandomTree	-K 2 -M 1.0 -V 0.001 -S 7	68.1025641	72.20512821	63.8974359
RandomTree	-K 4 -M 1.0 -V 0.001 -S 1	68.82051282	61.02564103	60.30769231
RandomTree	-K 1 -M 1.0 -V 0.001 -S 8	65.48717949	55.48717949	62.92307692
RandomTree	-K 4 -M 1.0 -V 0.001 -S 2	64.51282051	54.51282051	70.15384615
RandomTree	-K 1 -M 1.0 -V 0.001 -S 9	61.48717949	53.02564103	55.23076923

RandomTree	-K 4 -M 1.0 -V 0.001 -S 3	57.23076923	54.87179487	52.92307692
RandomTree	-K 1 -M 1.0 -V 0.001 -S 6	67.28205128	61.8974359	76.35897436
RandomTree	-K 1 -M 1.0 -V 0.001 -S 7	64.51282051	66.35897436	56.20512821
RandomTree	-K 4 -M 1.0 -V 0.001 -S 4	71.64102564	58	57.64102564
RandomTree	-K 1 -M 1.0 -V 0.001 -S 4	65.94871795	60.56410256	66.35897436
RandomTree	-K 1 -M 1.0 -V 0.001 -S 5	68.35897436	62	59.23076923
RandomTree	-K 1 -M 1.0 -V 0.001 -S 2	61.07692308	67.33333333	64.82051282
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	64.05128205	64.87179487	64.76923077
RandomTree	-K 1 -M 1.0 -V 0.001 -S 3	65.43589744	60.25641026	60.97435897
RandomTree	-K 9 -M 1.0 -V 0.001 -S 10	54.61538462	62.35897436	53.23076923
RandomTree	-K 4 -M 1.0 -V 0.001 -S 9	64	53.69230769	55.53846154
RandomTree	-K 4 -M 1.0 -V 0.001 -S 8	54.61538462	53.74358974	66.82051282
RandomTree	-K 4 -M 1.0 -V 0.001 -S 7	64.30769231	69.02564103	73.38461538
RandomTree	-K 4 -M 1.0 -V 0.001 -S 6	62.76923077	62.56410256	69.12820513
RandomTree	-K 4 -M 1.0 -V 0.001 -S 5	57.38461538	58	47.58974359
RandomTree	-K 2 -M 1.0 -V 0.001 -S 0	58.30769231	64.87179487	63.8974359
RandomTree	-K 2 -M 1.0 -V 0.001 -S 1	60.25641026	57.53846154	56
RandomTree	-K 2 -M 1.0 -V 0.001 -S 2	59.07692308	64.76923077	70.15384615
Random Forest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 4	58.05128205	56.56410256	58.61538462

Random Forest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 5	57.79487179	56.51282051	58.35897436
Random Forest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 10	57.64102564	56.82051282	61.53846154
Random Forest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 6	57.38461538	58.56410256	58.71794872
Random Forest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 7	57.64102564	59.17948718	57.94871795
Random Forest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1	57.79487179	58.92307692	57.69230769
Random Forest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 2	58.30769231	56.76923077	60.41025641
Random Forest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 3	57.8974359	57.12820513	58.30769231
Random Forest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 9	58.15384615	58.87179487	58.35897436
Random Forest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 8	57.07692308	58.41025641	61.28205128
AdaBoostM1	-P 100 -S 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2	33.33333333	33.33333333	33.33333333
AdaBoostM1	-P 100 -S 1 -I 10 -W weka.classifiers.trees.DecisionStump	66.56410256	67.02564103	71.43589744
AdaBoostM1	-P 100 -S 1 -I 10 -W weka.classifiers.trees.RandomForest -- -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1	72.41025641	67.23076923	70.1025641

AdaBoostM1	-P 100 -S 1 -I 10 -W weka.classifiers.trees.HoeffdingTree -- -L 2 -S 1 -E 1.0E-7 -H 0.05 -M 0.01 -G 200.0 -N 0.0	73.53846154	68.25641026	71.8974359
AdaBoostM1	-P 100 -S 1 -I 10 -W weka.classifiers.lazy.KStar -- -B 20 -M a	53.84615385	57.07692308	59.84615385
AdaBoostM1	-P 100 -S 1 -I 10 -W weka.classifiers.bayes.NaiveBayes	58.15384615	60.1025641	66.66666667
AdaBoostM1	-P 100 -S 1 -I 10 -W weka.classifiers.bayes.BayesNet -- -D -Q weka.classifiers.bayes.net.search.local.K2 -- -P 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5	33.33333333	33.33333333	33.33333333
Bagging	-P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.rules.DecisionTable -- -X 1 -S "weka.attributeSelection.BestFirst -D 1 -N 5"	33.33333333	33.33333333	33.33333333
Bagging	-P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2	33.33333333	33.33333333	33.33333333

Bagging	-P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.HoeffdingTree -- -L 2 -S 1 -E 1.0E-7 -H 0.05 -M 0.01 -G 200.0 -N 0.0	73.53846154	68.05128205	71.8974359
Bagging	-P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.bayes.NaiveBayes	61.79487179	62.15384615	66.66666667
Bagging	-P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0	33.33333333	33.33333333	33.33333333
OneR	-B 3	33.33333333	33.33333333	33.33333333
OneR	-B 6	33.33333333	33.33333333	33.33333333
J48	-U -M 2	33.33333333	33.33333333	33.33333333
J48	-C 0.25 -M 2	33.33333333	33.33333333	33.33333333
REPTree	-M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0	33.33333333	33.33333333	33.33333333
REPTree	-M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0 -R	33.33333333	33.33333333	33.33333333
Decision Stump	-batch-size 200	49.23076923	55.12820513	58.15384615
Decision Stump	no-arg	58.15384615	60.1025641	66.66666667
KStar	-B 20 -E -M a	77.12820513	70.92307692	68.87179487
KStar	-B 20 -M a	71.38461538	58.05128205	70.41025641
Multilayer Perceptron	-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a	61.12820513	54.66666667	70.76923077
Multilayer Perceptron	-L 0.3 -M 0.2 -N 500 -V 0 -S 4 -E 20 -H a	63.28205128	51.48717949	71.07692308

Table 3 - Evaluator Results using spring 2016 Sonification approaches

1.2.1. Classifiers' Options Overview

Before analyzing the data in the table, an overview about some of the classifiers' options that are used in the machine learning models evaluation process will be useful, as it will help in understanding the impact of these options on the classifiers performance.

1. *RandomTree* classifier is run using the following options:
    - -K: number of attributes.[2]
    - -M: minimum number of instances per leaf.
    - -V: minimum numeric class variance proportion of train variance for split.
    - -S: Seed for random number generator.
2. *AdaBoostM1* classifier is run using the following options:
    - -P: Percentage of weight mass to base training on.
    - -S: Random number seed.
    - -I: Number of iterations.
    - -W: Full name of base classifier. In addition to options specific to the classifier.
3. *Bagging* classifier is run using the following options:
    - -P: Size of each bag, as a percentage of the training set size.
    - -S: Random number seed.
    - -I: Number of iterations.
    - -W: Full name of base classifier. In addition to options specific to the classifier.

[2] Machine learning algorithms options come from the Weka on-line documentation [10].

4. *HoeffdingTree* classifier is run using the following options:
    - -L: The leaf prediction strategy to use. 0 = majority class, 1 = naive Bayes, 2 = naive Bayes adaptive.
    - -S: The splitting criterion to use. 0 = Gini, 1 = Info gain.
    - -E: The allowable error in a split decision - values closer to zero will take longer to decide.
    - -H: Threshold below which a split will be forced to break ties.
    - -M: Minimum fraction of weight required down at least two branches for info gain splitting.
    - -G: Grace period - the number of instances a leaf should observe between split attempts.
    - -N: The number of instances (weight) a leaf should observe before allowing naive Bayes to make predictions (NB or NB adaptive only).

1.3. Data Analysis

By looking at the data in table 3 we can see three types of classifiers: classifiers that performed poorly in terms of the classification's results, classifiers that achieved results in the same range of human results, and classifiers that performed better than human results.

1.3.1. Poor Classifiers

BayesNet, *OneR*, *J48*, *REPTree*, and ensemble classifiers *AdaBoostM1* and *Bagging*, that use the formerly mentioned classifiers as options in the process of building their models, produced very low average correct results' percentages equal to 33.33%. This is equivalent to making random classification, because we have three classes only, and each test dataset contains instances that are distributed evenly over the three classes, therefore, the classes have equal probability in the test dataset, which mean that random classification, in the case of sufficient number of instances should results to 33.33% correct classifications' results. These results are expected for those classifiers because they usually need big datasets for training their models in order to produce accurate predictions, which is different than our case where we are using small data sets; therefore, the prediction was almost random.

1.3.2. Classifiers within Human Results' Range

RandomTree Classifier with options *-K 4 -M 1.0 -V 0.001 -S 0*, was the only classifier that fell in the same range of the Student survey results, where the classifier average results' ranges were between 64% and 65%.

RandomTree classifier considers *K* randomly chosen attributes at each node while building its model, therefore, these attributes impact significantly the accuracy of the model, in the sense that, the more dominant these attributes are, the more accurate the model. In addition, *RandomTree* classifier does not perform pruning; lack of pruning retains all nodes and paths in its classification trees; retaining as much decision logic as

possible is especially useful in the case of small training dataset. Pruning is more appropriate when there are large training sets that need to be generalized, in part, via pruning. In our case, the classifier with the formerly mentioned options was capable of producing a model with results equivalent to the human results using very small training dataset. However, we should keep in mind that the nature of the data that is used for building the classifier is very crucial for its accuracy.

1.3.3. Classifiers Which Exceeded Human Results' Range

Kstar, and the ensemble classifiers *AdaBoostM1* with options, *-P 100 -S 1 -I 10 -W weka.classifiers.trees.HoeffdingTree -- -L 2 -S 1 -E 1.0E-7 -H 0.05 -M 0.01 -G 200.0 -N 0.0*, and *Bagging* classifier with options, *-P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.HoeffdingTree -- -L 2 -S 1 -E 1.0E-7 -H 0.05 -M 0.01 -G 200.0 -N 0.0* , performed very well. Results were in the range of 68% – 78% average correct classifications' percentages.

Kstar is an instance-based classifier, where predicting the class of an instance is based upon the class of those training instances similar to it, as determined by some similarity function [10]. This process is similar to the human learning and classification process in the sense that the human gets exposed to a dataset, then he uses the learned data as a reference point for classifying newly coming instances based on some similarity function.

Before running the evaluation process, we expected that *Kstar* would be a strong candidate for producing good results; however, it exceeded our expectations. *Kstar*

classifier was able to produce much better results than the human results using the same small training datasets.

The ensemble classifier *AdaBoostM1* with options, *-P 100 -S 1 -I 10 -W weka.classifiers.trees.HoeffdingTree -- -L 2 -S 1 -E 1.0E-7 -H 0.05 -M 0.01 -G 200.0 -N 0.0*, relies on boosting the performance of *HoeffdingTree* classifier, which exploits the fact that a small sample can often be enough to choose an optimal splitting attribute. This idea is supported mathematically by the *Hoeffding* bound, which quantifies the number of observations needed to estimate some statistics within a prescribed precision [10]. This classifier conforms to the principle with our main goal of using a small dataset. The *AdaBoostM1* was able to improve the performance of the *HoeffdingTree* classifier to produce results better than human correctness results.

The second ensemble classifier that produced good results is the *Bagging* classifier with options, *-P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.HoeffdingTree -- -L 2 -S 1 -E 1.0E-7 -H 0.05 -M 0.01 -G 200.0 -N 0.0*, where also it worked on improving the performance of *HoeffdingTree* classifier that we discussed in the previous ensemble classifier.

Chapter 6

1. Identifying Best Sonification Approach

1.1. Sonification Algorithms Overview

The Sonification algorithms that were used in the Sonification research have evolved gradually according to the students' survey results.

In fall of 2015, they conducted the survey using *Harmonic*, *Melodic*, and *Waveform* algorithms. The results of that survey, which are shown in table 1, was a surprise to the research leader Dr. Parson, where Waveform algorithm unexpectedly achieved a better performance than the other two algorithms.

Sonification (Fall 2015)	Category	Mean correct responses
Harmonic	All 3 sets	55.8%
Melodic	All 3 sets	55.4%
Waveform	All 3 sets	61.4%

Table 4 - fall 2015 Sonification survey results

These results encouraged the researchers to explore more variations of the *Waveform* Sonification algorithm, therefore, they developed *WaveformDouble*, *WaveformFourThirds* and *WaveformOnePt95* algorithms.

WaveformDouble sums two copies of the waveform in the original *Waveform* sonification algorithm, one at the original frequency and another with a frequency that is 2.0 X the original waveform.

WaveformFourThirds and *WaveformOnePt95* are similar to *WaveformDouble,* except that they sum second frequencies that are 4.0/3.0 X, and 1.95 X the original *Waveform* respectively, instead of 2.0 in the *WaveformDouble*.

In spring of 2016 researchers conducted the survey using *WaveformDouble*, *WaveformFourThirds* and *WaveformOnePt95* algorithms where the survey results are shown in Table 2.

Sonification (Spring 2016)	**Category**	**Mean correct responses**
WaveformDouble	All 3 sets	67.8%
WaveformFourThirds	All 3 sets	65.8%
WaveformOnePt95	All 3 sets	67.6%

Table 5 - spring 2016 Sonification Survey Results

Despite the fact that the results were better than the results of fall 2015 survey, Dr. Parson was still thinking of improving the performance of the previously used algorithms by fixing the curve that was used for generating the *Sweet* and *Sour* notes for each of the data attributes. As you can see in the original curve shown in figure 12, the *Sweet* note that is represented by the blue curve, is changing the direction at 0.9 standard deviation value. The fix for that issue can be shown in the curve in figure 13. In addition, Dr. Parson generated a new linear version of the *Sweet* and *Sour* curve as shown in figure 14, which can be also used for generating the *Sweet* and *Sour* notes for the Sonification algorithms instead of the original and the fixed curves versions.

In addition, Dr. Parson had new ideas for completely new Sonification approaches that he wanted to implement hoping for better results, where the new Sonification algorithms will rely on using different wave types other than the triangular one, which was the main

type of the previously tested waveform algorithm variations (*WaveformDouble*, *WaveformFourThirds* and *WaveformOnePt95*).

1.2. New Sonification Algorithms

The first set of new Sonification algorithms are *Waveformdblsaw*, *Waveform43rdssaw*, and *Waveform195saw*. These algorithms use a sawtooth waveform that starts at the lowest level for an attribute and rises to the highest, then falling back to lowest, instead of a triangular waveform. In addition, these algorithms use frequencies that are 2.0 X, 4.0/3.0 X, and 1.95 X the frequency of the original waveform frequency.

The second set of new algorithms consists of *Waveformdblrevs*, *Waveform43rdsrevs*, and *Waveform195revs*. These algorithms use a reverse-sawtooth (high-to-low ramps) waveform that starts at the highest level for an attribute and drops down to the lowest, then rises up to highest, instead of a triangular waveform. In addition, these algorithms use second frequencies that are 2.0 X, 4.0/3.0 X, and 1.95 X the frequency of the original waveform frequency.

We implemented the two sets of new algorithms using the fixed and the linear version of the *Sweet* and *Sour* curves shown in figure 13 and 14.

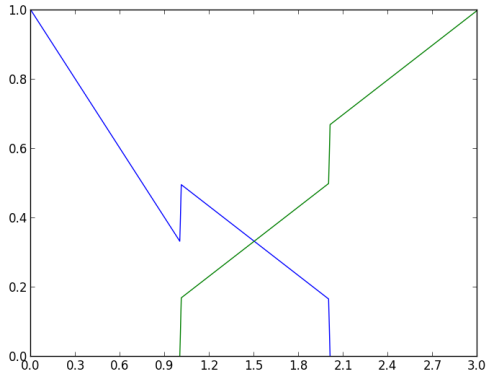

Figure 12 - original curves in Sonification research [2]

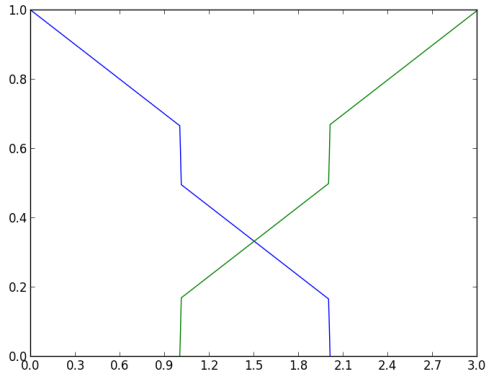

Figure 13 - fixed Sweet and Sour curves

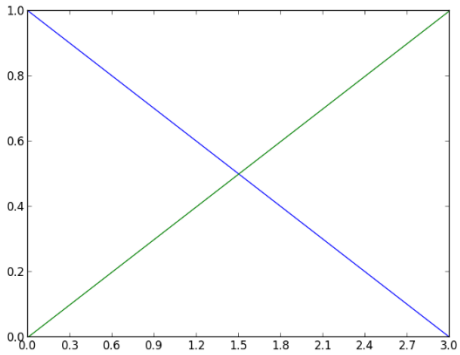

Figure 14 - Sweet and Sour curves for linear sonifications

1.3. Research Result Utilization

The process of evaluating new Sonification algorithms is cumbersome, because it takes a long time and a lot of arrangement to conduct the necessary surveys for getting all needed results.

Thus, we will employ the result of our research to facilitate and automate the process of evaluating the performance of new Sonification algorithms, where the combination of the *Machine Learning Evaluator Tool* with the machine learning model that was capable of mimicking human results will produce the *Virtual Survey Listener tool* that can be used for conducting the Sonification surveys to evaluate the new Sonification algorithms.

As was seen in the previous chapter, *RandomTree* Classifier with options *-K 4 -M 1.0 -V 0.001 -S 0*, was the only classifier that fell in the same range of the 2016 spring survey results, therefore, we will use the *RandomTree* Classifier in conjunction with the *Machine Learning Evaluator Tool* for evaluating the new Sonification algorithms performance.

1.4. Data Analysis

After applying the evaluation process that is shown in the figure 11 on the new Sonification algorithms using the *Virtual Survey Listener tool* we got the results shown in table 4

Classifier	**Option**	**Sonification Algorithm**	**Correct Avg result %**
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform43rdsfixed	62.15384615
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform43rdsrevs	49.94871795
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform43rdssaw	54.35897436
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform195fixed	56.20512821
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform195revs	54.92307692
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform195saw	65.38461538
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveformdblfixed	53.38461538
RandomTree	-K 4 -M 1.0 -V 0.001	waveformdblrevs	61.58974359

	-S 0		
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveformdblsaw	76.05128205
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform43rdslin	59.02564103
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform43rdsrevslin	50.46153846
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform43rdssawlin	51.33333333
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform195lin	56.1025641
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform195revslin	54.87179487
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveform195sawlin	66.25641026
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveformdbllin	52
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveformdblrevslin	59.84615385
RandomTree	-K 4 -M 1.0 -V 0.001 -S 0	waveformdblsawlin	68.76923077

Table 6 - Virtual Survey Listener Results with new Sonification approaches

By looking at the data in the table we can see that most of the algorithms that use 4/3 X the original waveform frequency are performing poorly where they achieved results below 53% average correct classifications, however, the 4/3 algorithm that is using sawtooth waveform performed slightly better as it achieved 54.35897436%.

Algorithms that use 1.95 X the original frequency are performing better than the 4/3 algorithms where most of the results were in the range between 54% and 59% except for the 1.95 algorithms that used sawtooth waveform which are *Waveform195saw* and *Waveform195sawlin* algorithms where results are 65.38461538% and 66.25641026% respectively.

Algorithms that use double the original frequency were the best in terms of the results, in particular the algorithms that use the sawtooth waveform, as most of results were in the range between 59% and 77%.

As we noticed in all three groups the algorithms that used sawtooth waveform perform better than the algorithms that use the other types of waveforms. Sawtooth audio waveforms sound more “raspy” to human listeners than triangular waveforms. A triangle wave’s sound approaches the simple, somewhat “thin” sound of a sine wave. Therefore, the best Sonification approach overall is one of the algorithm that used the sawtooth waveform which is *Waveformdblsaw* algorithm, where the *Virtual Survey Listener tool* was capable of achieving 76.05128205% average correct classification for that particular Sonification approach.

1.5. Virtual Survey Listener Tool Benefit

Now that we can use the *Virtual Survey Listener tool*, testing Sonification approaches becomes very easy and more time and cost effective, since the process is automated and no need for human interaction to find the results.

The features of the *Virtual Survey Listener tool* will provide us with the necessary tools for driving the research further in the future, as we can test new Sonification approaches or improve the existing ones in a cost and time effective manner.

Chapter 7

1. Conclusion

Exercising machine learning systems in the same conditions and mechanics of human learning experience is crucial for finding systems that are capable of making similar predictions to humans in terms of the results.

In our research we considered a very small training data set as a key factor for training the machine learning models in order to mimic human learning experience, because it is proven that human brain can learn more effectively using small datasets and distributed learning sessions.

We used the Sonification research datasets as our main data source for training and testing the machine learning models for many reasons:

First, the Sonification research uses small training datasets for training the survey takers, which conform to our objective of using small datasets for training the machine learning models.

Second, the Sonification research results are available to be compared with the machine learning models results.

Third, a useful result of our project is the *Virtual Survey Listener Tool*, which will serve in the completion of the Sonification research by testing new Sonification approaches.

We tried to be as comprehensive as possible in terms of experimenting with the most common machine learning models, therefore, we tested with multiple models such as:

- Bayes algorithms such as Naïve Bayes and Bayes Net,
- Ensemble learners such as AdaBoostM1, Bagging, and Random Forest ,
- Rules learners such as, One R,
- Decision tree algorithms such as J48, REP Tree, Random Tree, Decision Stump,
- Instance-based learner (Lazy learner) such as K Star,
- Function learner such as Multilayer Perceptron.

Due to the lack of *Weka Tool* scalability and analysis capabilities, we created the *Machine Learning Evaluator Tool* that extended the capability of the *Weka Tool*. The *Machine Learning Evaluator Tool* allowed us to train and test all the machine learning models using the Sonification datasets used in the Sonification research, and it gave us an easy way to represent the results and analyze them in comparison with the Sonification research survey results.

After running the datasets that had already been surveyed in the Sonification research through the *Machine Learning Evaluator Tool*, we were able to distinguish three sets of machine learning models. The first set is represented by *BayesNet*, *OneR*, *J48*, *REPTree* algorithms and *AdaBoostM1*, *Bagging* algorithms that used the former algorithms for building their models, which demonstrated very poor results with 33.33% average correct classifications. The second set is represented by *RandomTree* classifier with options *-K 4 -M 1.0 -V 0.001 -S 0*, which generated results similar to human results with a range

between 64% and 65% average correct classifications. The third set is represented by *Kstar* classifier, and *AdaBoostM1* and *Bagging* classifiers that used *HoeffdingTree* classifier as option for building their models, which generated results better than human results with a range between 68% – 78% average correct classifications.

When we first started the research, we expected the *Kstar* classifier to be among the best machine learning models due to the process similarity with human learning and classification process, both of which save the training instances in memory (in the training phase) to be used later in the classification phase. Instance-based learning in humans and algorithms measures similarity of test data instances to members of a limited set of training instances, using some form of similarity metric.

By using the machine learning algorithm that was capable of mimicking the human results, the *RandomTree* classifier, in conjunction with the *Machine Learning Evaluator Tool*, we produced the *Virtual Survey Listener tool* that provided us with the ability of evaluating new Sonification approaches, as an effort for producing new results to be used for completing the Sonification research.

The new Sonification approaches are some type of variation of the *Waveform* Sonification algorithm. They used wave shapes other than the triangular one, the sawtooth and reverse-sawtooth wave shapes. After running the new Sonification algorithms datasets through the *Virtual Survey Listener tool*, we found out that the Sonification algorithms that use sawtooth waveform perform better than the other ones, and *Waveformdblsaw* Sonification algorithm in particular is the best in terms of correct

average classifications as it achieved 76.05128205% correct classifications, This makes it the best Sonification approach among all, old and new Sonification approaches.

The *Virtual Survey Listener tool* as a result of our research made the Sonification research more feasible in terms of time, cost effectiveness, and better results, because the testing can be repeated as many times as needed in order to produce an optimal Sonification algorithm, which can be used for developing applications that are as useful as the virtualization's applications.

Bibliography

[1] Anderson WL, Mitchell SM, Osgood MP (2005) *Comparison of student performance in cooperative learning and traditional lecture-based biochemistry classes*, Biochem Mol Biol Educ 33(6):387–393

[2] D. Parson, D. Hoch, and H. Langley, *TIMBRAL DATA SONIFICATION FROM PARALLEL ATTRIBUTE GRAPHS,* Proceedings of the 31st Annual Conference of The Pennsylvania Association of Computer and Information Science Educators April 01 - 02, 2016 Hosted by Kutztown University. http://faculty.kutztown.edu/parson/pubs/SonificationPacise2016Published.pdf

[3] T. Hermann, A. Hunt, and J. Neuhoff (editors), *The Sonification Handbook*, Logos Verlag, Berlin, Germany, 2011.

[4] J. Han, M. Kamber, and J. Pei, *Data Mining – Concepts and Techniques*, Third Edition, Morgan Kaufmann, 2012.

[5] P. Simon (March 18, 2013), *Too Big to Ignore: The Business Case for Big Data*, Wiley. p. 89. ISBN 978-1-118-63817-0.

[6] http://whatis.techtarget.com/definition/machine-learning this tertiary source reuses information from other sources but does not name them.

[7] Wernick, Yang, Brankov, Yourganov and Strother, *Machine Learning in Medical Imaging*, IEEE Signal Processing Magazine, vol. 27, no. 4, July 2010, pp. 25-38

[8] S. Russell, P. Norvig, (2003) [1995], *Artificial Intelligence: A Modern Approach (2nd ed.)*, Prentice Hall. ISBN 978-0137903955.

[9] J. Brownlee (November 25, 2013), *Machine Learning Algorithms,* http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/.

[10] Machine Learning Group at the University of Waikato, *Weka 3: Data Mining Software in Java*, http://www.cs.waikato.ac.nz/ml/weka/.

[11] R. Agrawal, T. Imieliński, A. Swami, (1993). *Mining association rules between sets of items in large databases*, Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93.
p. 207. doi:10.1145/170035.170072. ISBN 0897915925.

[12] A. Inselberg, Parallel Coordinates, *Visual Multidimensional Geometry and Its Applications*, 2009 Edition, Springer, 2009.

[13] D. Parson and A. Seidel, *Mining Student Time Management Patterns in Programming Projects*, Proceedings of FECS'14: 2014 Intl. Conf. on Frontiers in CS & CE Education, Las Vegas, NV, July 21 - 24, 2012. http://faculty.kutztown.edu/parson/pubs/FEC2189ParsonSeidel2014.pdf

[14] D. Parson, L. Bogumil and A. Seidel, *Data Mining Temporal Work Patterns of Programming Student Populations*, Proceedings of the 30th Annual Spring Conference of the Pennsylvania Computer and Information Science Educators (PACISE) Edinboro University of PA, Edinboro, PA, April 10-11, 2015. http://faculty.kutztown.edu/parson/pubs/StudentTimePacise2015Kutztown.pdf

[15] Ge Wang (2008), *The ChucK Audio Programming Language: A Strongly-timed and On-the-fly Environ/mentality,* (Ph.D.). Princeton University.

[16] Grey, J. M., Gordon, J. W., 1978*, Perceptual effects of spectral modifications on musical timbres*, Journal of the Acoustical Society of America 63 (5), 1493–1500

[17] J. Neuhoff, *Perception, Cognition and Action in Auditory Displays*, The Sonification Handbook, T. Hermann, A. Hunt, and J. Neuhoff (editors), Logos Verlag, Berlin, Germany, 2011, p. 63-85.

[18] Witten, Frank and Hall, *Data Mining: Practical Machine Learning Tools and Techniques,* Third Edition, Morgan Kaufmann, 2011.

[19] George H. John, Pat Langley: *Estimating Continuous Distributions in Bayesian Classifiers*. In: Eleventh Conference on Uncertainty in Artificial, San Mateo, 338-345, 1995.

[20] T. G. Dietterich, *An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization*, Department of Computer Science, Oregon State University, Corvallis, OR 97331, USA

[21] J. G. Cleary, L. E. Trigg, *K*: An Instance-based Learner Using an Entropic Distance Measure,* Dept. of Computer Science, University of Waikato, New Zealand

[22] L. Rokach, *Ensemble-based classifiers,* Published online: 19 November 2009 Springer Science+Business Media B.V. 2009

[23] Z. Abu Deeb, T. Devine, Z. Geng, *Randomized Decimation HyperPipes,*

[24] D. W. Aha, D. Kibler, M. K. Albert, *Instance-Based Learning Algorithms,* Department of Information and Computer Science, University of California, lrvine, CA 92717

[25] T. Hertz, A. Bar Hillel, D. Weinshall, *Learning a Kernel Function for Classification with Small Training Samples,* School of Computer Science and Engineering, The Center for Neural Computation, The Hebrew University of Jerusalem, Jerusalem, Israel 91904.

[26] E. Frank, M. Hall, and B. Pfahringer, *Locally Weighted Naive Bayes,* Department of Computer Science University of Waikato Hamilton, New Zealand

[27] L. Breiman, *Random Forests,* Statistics Department, University of California, Berkeley, CA 94720

[28] S. Rameshpant Kalmegh, *Comparative Analysis of WEKA Data Mining Algorithm RandomForest, RandomTree and LADTree for Classification of Indigenous News Data,* Associate Professor, Department of Computer Science, SGBAU, Amravati (M.S.), India.