2011

Instance selection for simplified decision trees through the generation and selection of instance candidate subsets

Walter Dean Bennette
Iowa State University, bennette@iastate.edu

Instance selection for simplified decision trees through the generation and selection of instance candidate subsets

by

Walter Dean Bennette

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Industrial Engineering

Program of Study Committee:
Sigurdur Olafsson, Major Professor
Sarah Ryan
Dianne Cook

Iowa State University

Ames, Iowa

2011

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Decision trees are a useful tool to help in the extraction of information from databases, but all too often this ability is clouded by the complexity of the tree structure resulting from the decision tree algorithm. Methods such as tree pruning, attribute selection, and most recently, instance selection, currently exist to simplify the decision tree structure. We present an alternative instance selection procedure for simplifying decision trees that improves upon previous methods by increasing the quality of the space to be traversed for finding an acceptably simplified decision tree through the identification and grouping of important instances. Experimental results from this procedure are then presented and compared to decision trees with no prior simplification effort applied. We show that in some cases we are indeed able to identify important group of instances, and subsequently are able to generate a high quality solution space for finding simplified decision trees.

CHAPTER 1 INTRODUCTION

1.1 Motivation

In this ever increasingly technological world, huge amounts of data are being stored and collected in databases. These databases can describe customer shopping habits, patient experiences at hospitals, the scheduling practices of a large manufacturing plant, and countless other situations. Hidden inside of these databases are pieces of information describing relationships between situational variables and outcomes. Policy makers of all types can hope to gain insight about the best practices of their industry from the careful implementation of data mining as an analysis tool. Schedulers may discover a new schedule to improve their production line, doctors may identify better means of patient care, retailers might discover new ways to increase sales, and the list could go on. Current methods of data mining, or automated processes to find descriptive relationships through the analysis of databases, are available to aid policy makers. Classification is one of those data mining techniques and is a great tool to help policy makers make observations about their industries.

In data mining, classification is the task of assigning a class value to previously unclassified instances, and in doing so learning what makes a particular instance belong to one class or another. In this research an instance represents a collection of descriptive attributes that describe a particular event or object, and has a class value that is either known or unknown. By learning a classification scheme, or classifier, that is based on training instances, or instances with known class values, hidden relationships become easier for

policy makers to recognize and make use of. A popular implementation of classification is to create a decision tree because of its ease of interpretability.

As a simple, yet classic example of classification, imagine recording the weather conditions each day for a two week period and making note of whether or not an individual plays tennis on a particular day. Each day represents an instance and is defined by different attributes as well as a class value that reveals if tennis was played on that day. The attributes used to describe each instance could be the weather outlook, temperature, and humidity. Given the opportunity to learn from these previously recorded, or training instances, a classifier can be built. If this classifier is accurate, analyzers can predict whether or not tennis will be played on a yet to be observed day. In addition to this predictive ability, and given that analyzers are using an interpretable classification technique such as a decision tree, there is potential to discover the underlying factors that lead to an individual playing tennis on a particular day.

A decision tree for classification is constructed in a top down fashion, with the attribute presumed to be the most beneficial for separating the data into homogeneous groups being selected to be the root attribute, or beginning of the tree. The tree branches the data into separate groups based on the possible values of that attribute, and has the goal of separating the data into groups composed of instances from a single class. This procedure of selecting the best attribute and then splitting the data is repeated for lower and lower levels of the tree, until the point where further partitioning of the data leads to a declaration of class value. At this leaf level of the decision tree, each partitioning of the data is almost completely homogeneous in class value, and the tree is complete. It is because of this simple

hierarchical and relational structure that decision trees are among the most popular classification techniques. However, decision trees are rarely as simple and easy to interpret as a policy maker may wish they were. A decision tree with a large number of branching points may be as hard to interpret as the database itself.

One possible reason for complex decision trees is the inclusion of unhelpful instances in the classifier's training set. A practice known as instance selection is an optional preprocessing step of data mining where a subset of the original training dataset is selected to learn from. This subset can be chosen with a specific goal in mind, for example, to create simple and easily interpretable decision trees for policy makers. Therefore, the motivation of this research is the need for simple decision trees, or as simplicity often implies, decision trees that have been reduced in size. Our premise is that some instances are not helpful when learning decision trees and that learning from only select instances in the training set may result in improved decision trees, and in particular smaller more interpretable ones.

1.2 Objective

The premise of this thesis is that instance selection can be used to create easily interpretable and accurate decision trees. It could be argued that some instances are the most representative of a data set, and that by learning from these instances the best decision tree will be constructed. However, the number of possible instance subsets, that is, the possible combinations of n instances is 2^n, a prohibitively large space to search. Therefore, the objective of this thesis is to devise an approach that can improve the quality of the solution search space and reduce the amount of searching required to obtain a suitable solution.

The research plan for addressing this objective is to model instance selection as a set covering problem. The traditional set covering problem has two inputs, a set of elements and a collection of subsets made from the original elements. The goal of this problem is to make a minimal selection of the provided subsets in such a way that the selected subsets contain all of the elements from the original set. In the context of instance selection, the original elements would be the training instances and the task would be to select a set of candidate training subsets that are able to accurately represent the original data. Due to the huge number of possible subsets from a set of training data, it is not computationally practical to use every subset in the set covering problem. Instead, it is proposed that a collection of good subsets can be constructed and used. To complete the research objective this work has been broken into three research tasks.

Research Task 1: We will formulate a set covering problem for instance selection, and in particular develop efficient methods for generating candidate instance subsets to select from. Generating candidate instance subsets will be accomplished through the greedy addition of instances as judged by a very simple classifier.

Research Task 2: We will relax the linear set covering problem formulation in Task 1 by introducing a non-linear objective function that reflects the estimated prediction accuracy when using the selected instance subsets.

Research Task 3: We will develop and compare solution methods for both of the above formulations.

Following this research plan we hope to show that accurate and easy to interpret decision trees can be created. In doing so, we are providing policy makers with means of analyzing databases and developing actionable information.

1.3 Thesis Organization

This research has been broken into four sections. The first section formally introduces the need for simplifying decision trees as well as introduces current techniques for improving decision trees such as tree pruning, attribute selection, and instance selection. In the second section, the formulation of our procedure is presented. A method of creating good candidate subsets, methods for selecting final sets of training data from amongst those subsets, and the construction of decision trees, will all be discussed. Next, nine different datasets are tested to see if indeed our instance selection procedure will return favorable results. Then finally, overall results and implications will be examined along with thoughts for future areas of research.

CHAPTER 2 LITERATURE REVIEW

2.1 Decision Trees

A Decision Tree is a useful and popular classification technique that inductively learns a model from a given set of data. One reason for its popularity stems from the availability of existing algorithms that can be used to build decision trees, such as CART (Breiman et al., 1984), ID3 (Quinlan, 1986), and C4.5 (Quinlan, 1993). These algorithms all learn decision trees from a supplied set of training data, but do so in slightly different ways. As discussed in the introduction, a classifier is built by analyzing training data. That is to say, a classifier is built by analyzing a collection of instances where each instance is composed of a set of attribute values and a known class value. These decision tree algorithms build top down structures that partition instances into separate classes, and it is hoped that these structures generalize well to instances with unknown class values. This would mean that the decision trees have fulfilled their objectives and have indeed discovered some underlying property of the data (Quinlan, 1986).

The idea behind the decision tree's structure is to split instances into separate groups that are as homogeneous as possible, and these splits are based upon the different values for particular attributes. Typically, the most meaningful attribute of a dataset is the attribute that when data is split along its possible values leads to the most homogeneous groups of data being formed. In most situations, it is this attribute that is selected to be the first branch of the tree. Further partitioning of the data is usually required to achieve completely

homogeneous groups, or groups composed of instances from a single class, and this is done based upon the values of less and less meaningful attributes.

The main difference between CART, ID3 and C4.5 is how the partitioning of data is performed. CART uses the Gini index to select the splitting attribute, whereas ID3 and C4.5 use a measurement of information gain and the information gain ratio. The Gini index measures the impurity of partitions by looking at the probability that the instances belonging to a specific group are from one class or another, and selects the attribute that has the highest probability of being a completely homogeneous split. Information gain is a measurement developed by Claude Shannon that quantifies the amount of information needed to classify a group of instances, and the gain ratio is an improvement upon this measurement. When information gain or the gain ratio are used to partition data, the attribute that requires the least information to classify the resulting groups is selected to split the data, leading to simple yet descriptive partitions (Han et al., 2006) .

The available decision tree algorithms also differ in their ability to handle different types of data, and by far the most advanced algorithm of the group is C4.5. C4.5 is based upon Quinlan's earlier work with ID3 and is capable of handling continuous data, data with missing values, and even has built in steps for simplifying the resultant decision tree.

Another reason behind the popularity of decision trees is that they are often interpretable by human analyzers. The structure of a decision tree provides reasoning for every declaration of a class value, and it is hoped that this reasoning is easy to comprehend (Endou et al., 2002). Comprehensibility allows the human user to gain insight to the workings of their system, and is one of the most desired aspects of the decision tree.

One drawback of decision trees is that they are prone to over fitting their structure to the training instances and hence, lose some ability to generalize well to instances with unknown class values. A related problem of the decision tree is that superfluous instances such as noisy data and outliers are given equal consideration in the building of the decision tree. This often results in unnecessary structure at the bottom of the decision tree because branching occurs until groups are as homogeneous as possible, meaning that a branching may occur to accommodate just a single unimportant instance.

The consequence of these drawbacks is that a decision tree can become more complex than necessary, sometimes to the point of being incomprehensible (Sebban et al., 2000). Furthermore, experimental results by Oates and Jensen (1997) have shown that the structure of a decision tree continues to grow from the addition of new instances to the training set, even after the accuracy has stopped improving. This suggests that indeed only some of the training instances are required to construct an accurate and interpretable decision tree.

2.2 Decision Tree Simplification

Unfortunately, approximating the minimal representation of a decision tree is known to be an NP-hard problem (Hancock et al., 1996). Thus, heuristic methods for finding simple decision trees are justifiable. Procedures like tree pruning, attribute selection, and other specialized techniques currently exists to aid in the creation of simple and accurate decision trees.

The goal of decision tree pruning is to reduce the size of a tree that has been over fitted to the training data, in hopes of increasing its intelligibility and predictive accuracy.

Pruning approaches these goals in one of two ways. The first type of pruning is considered in the construction of the decision tree, and simply establishes stopping criteria for the growth of the tree. For example, a practice may be to stop the growth of a decision tree once there are less than a specified number of instances belonging to a leaf node, or to stop the growth when all instances belonging to a leaf node have the same attribute values. The second type of pruning is post-pruning, where knowledge is forgotten if doing so increases the predictive accuracy of the tree. Reduced Error Pruning (Quinlan, 1987), Minimum Error Pruning (Niblett, 1987), and Critical Value Pruning (Mingers, 1987) are some generally accepted and well utilized post-pruning methods. Experimental results by Esposito, Malerba, and Semeraro (1997) show that these pruning methods rarely decrease the predictive accuracy of decision trees. This result indicates that these pruning methods are a viable tool for tree simplification.

Attribute selection is another worthwhile approach to reduce the size of a decision tree, and is the task of finding a good subset of the data's attributes from which to learn. John, Kohavi, and Pfleger (1994) argue that, "Ideally, the induction algorithm should use only the subset of features (attributes) that leads to the best performance". Instead of learning from all of the attributes that describe instances, only necessary attributes need to be considered to allow the learning algorithm to focus on the relevant information. The selection of attributes can be done through either a filter or wrapper approach.

Filter attribute selection methods use a function to determine the value of learning from a specific attribute, and returns the ranking of these attributes according to that measure. Typically, this is accomplished with a trivial calculation on the training data, and includes

techniques such as analyzing Information Gain and a method known as Correlated-based Feature Selection. Information Gain, as developed by Claude Shannon and discussed earlier, has its basis in the information sciences and measures the impurity of partitions achieved from splitting the data along the different values of an attribute (Han et al., 2006). When used for attribute selection the goal is to find attributes that lead to the most pure partitions. Correlated-based Feature Selection is used for attribute selection by identifying attributes that are highly correlated with the class attribute, but have relatively little correlation between each other (Hall, 1998). The underlying concept of these filter attribute selection methods is to find a group of attributes that should work well for any number of learning algorithms.

On the other hand, wrapper attribute selection methods, find a good subset of attributes to induce a decision tree from, and do so by evaluating the effectiveness of different subsets of attributes with the intended induction algorithm. The wrapper problem boils down to searching for the best candidate solution from a search space that includes all of the possible subsets of the attribute set. This is a complex combinatorial problem with no easy solution. Known search methods such as greedy search heuristics or genetic algorithms are typically used to find a suitable subset of the attributes (John et al., 1994).

Some methods for finding simpler and more interpretable decision trees strive to improve the quality of the learning dataset by manipulating the occurrences of instances. Quinlan recommends that the tree produced by the ID3 algorithm can be reduced in size by using a subset of the training data to build the tree, and then improved by iteratively adding misclassified instances to the training subset, until a decision tree is built that correctly classifies all instances (Quinlan, 1986). Robust-C4.5 on the other hand, does the opposite by

iteratively removing misclassified instances from the training data until an adequate decision tree is built. This technique has been found to significantly reduce the size of the learned decision tree but does so at the sacrifice of the tree's accuracy (John, 1995). Another technique uses a "Consensus Filter". This filter finds and removes instances that have been mislabeled in the training data. Applying the "Consensus Filter" maintains the accuracy of the decision tree but does not significantly reduce its size (Brodley, 1996). In general, these methods closely resemble instance selection in that their goal is to find good sets of training data, but they do not directly address the desire for selecting only the highest quality instances for learning.

2.3 Instance Selection

Instance selection is an emerging area of interest for creating simple decision trees, and the objective is to select a subset of the original training data that is composed of high quality instances and that leads to desirable decision trees. These methods are similar to techniques for attribute selection in that different instance selection methods can be placed into one of two categories, wrapper or filter. Wrapper methods select a subset of training instances based upon some performance obtained by a classifier, whereas filter methods rank the utility of instances through some non-classifier based function and select instances from this ranking. Filter instance selection methods mostly try to identify instances that describe the boarders between different classes or find instances that are thought to be descriptive of a class. Filter methods work well for a number of different classifiers, are inherently faster than wrapper methods, and have been shown to deliver good results. However, most instance

selection methods are of the wrapper type because of their customizability to a specific data mining goal.

Early instance selection algorithms were designed for the k Nearest Neighbors (k-NN) classifier, an instance based learning algorithm where an entire set of training data is analyzed whenever a prediction is made. The goal of this method is to find an unclassified instance's closest neighbors, and make a declaration of class based on the class values of those neighbors. This algorithm uses all of the training instances for every new case analyzed, and because of this, it makes sense to store as few instances as necessary. Condensed Nearest Neighbor (Hart, 1968), Selective Nearest Neighbor (Ritter et al., 1974), and Edited Nearest Neighbor (Wilson, 1972) all select subsets of instances that lead to good classifications**,** and significantly reduce the number of instances that need to be stored. Some of the newest and best techniques for reducing the storage requirements of the k-NN classifier are the DROP methods. DROP1-5 is a collection of instance selection algorithms that have been empirically shown to work extremely well for instance based learning (Wilson et al., 2000). Despite the fact that the bulk of instance selection methods are designed for the k-NN classifier, the potential for instance selection methods in regards to building interpretable decision trees has been identified. To date the volume of literature on this subject is limited.

Current instance selection methods for creating simple decision trees are of the wrapper type and search the space of possible instance subsets with genetic algorithms. Endou and Zhao (2002) propose a genetic algorithm that evolves the content of preselected training subsets in the hopes of finding a subset that adequately describes the full dataset.

The fitness of these subsets is found through the construction and evaluation of a decision tree, and the resultant training set is of a fixed size to be declared by the user. This algorithm was shown to significantly reduce the size of decision trees built on redundant databases while still maintaining a certain level of accuracy. Perhaps not surprisingly, as there are a limited number of instances that can be selected for learning, this technique worked poorly for data with a large number of unique instances.

In a 2003 article "Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study" Cano (2003) et al. purposed to use instance selection to reduce the size of a dataset, and instance selection was again accomplished with the use of a genetic algorithm. This algorithm evolved a training set of data using the 1-NN classifier as a fitness function. This procedure greatly reduced the sizes of the training datasets and it was also noted that decision trees built from these training sets were accurate and easy to interpret (Cano, 2003). In Cano's study it appears that what is being sought after in this research is discovered, the use of instance selection for the creation of interpretable decision trees.

In a related study also performed by Cano (2007) et al., a second genetic search algorithm was designed. The objective of this search algorithm was to find sets of training data that would lead to small decision trees from large datasets. This algorithm addresses the issue of large datasets by performing instance selection in stratified layers, a popular technique for evading possible scaling up issues from the large amount of analysis that needs to occur. In contrast to their previous study, the fitness of these evolving training sets is measured by implementing the C4.5 decision tree learning algorithm. The results from this

study strongly indicate that training sets can be selected for the creation of simple decision trees. Three out of the five datasets used in this study showed improved or equal accuracy in comparison to the original decision tree, while all showed a reduction of, or at least maintained, decision tree size.

Despite the existence of instance selection methods for the creation of simple decision trees, there is still no convenient formulation of the root problem. Finding the optimal subset of instances from which to learn a decision tree from would require checking every subset of the original instances. Past methods have tried using clever search algorithms to look for the best possible subset of instances, but their search space is prohibitively large at 2^n possible solutions for an original training set of size n. In this work we propose a new formulation of the instance selection problem. The intent of this new formulation is to improve the quality of the space to be searched, allowing for fewer iterations of a searching algorithm to find a suitable solution. We will provide high quality candidate subsets to be considered in the construction of a simple decision tree, and eliminate the need to consider instances for addition to the final dataset with no prior knowledge of their ability to build good classifiers.

CHAPTER 3 MODEL DEVELOPMENT

3.1 Candidate Subsets

Given a set of training data, and the desire for a simple decision tree, our proposed method for instance selection can be implemented. This instance selection procedure begins with the creation of intelligently designed candidate subsets composed of instances from the original training data. After these candidate subsets are created, a selection is made amongst them, and the selected subsets are combined to obtain a new and reduced training dataset**.** This new dataset composed of high quality instances will then be used in the creation of a decision tree.

The motivation for creating candidate subsets is to form a solution search space that is composed of high quality elements, and this space can be traversed to find a selection of subsets that lead to the creation of a simple and interpretable decision tree. Unlike previous instance selection techniques, this approach allows groups of instances believed to work well in conjunction with one another to be selected in a single step. We believe the upfront cost for creating candidate subsets is outweighed by the benefit of creating a higher quality search space designed specifically for the creation of simple decision trees.

To create candidate subsets, instances from the original training data are considered for inclusion to the subsets in a greedy manner. Meaning, if an instance improves the immediate predictive contribution of a subset, as calculated by a simple classifier, it is included in that subset, and if not it is discarded. This procedure creates candidate subsets that are composed of instances that together lead to accurate classifiers. Keeping in mind

that the intent of creating these subsets is to create a high quality searchable space where the individual elements are actually the candidate subsets, we do not want each element in the space to be identical. Rather, the candidate subsets should differ, implying a different combination of instances, but with each combination still leading to an accurate classifier. Due to the fact that instances are considered for addition to the candidate subsets in a greedy fashion, and that there is a desire for differing candidate subsets, the order in which the instances are considered for inclusion becomes a concern.

A second concern of the subset creation process is that the intent of constructing candidate subsets is not to exclude instances from the final training dataset, but rather, it is to identify groups of instances that are believed to work well together. Therefore, each of the instances from the original training dataset should be given an opportunity to belong to at least one candidate subset. However, as a result of the greedy addition structure for creating candidate subsets, instances considered early in the subset creation process are more likely to be included in the subset than their counterparts that are considered later. In the early phases of subset creation almost any instance considered for the addition to a subset will reveal information not yet discovered about the data, and will likely result in an increase of the subset's predictive contribution and the inclusion of that instance. The more instances a subset accumulates, the more difficult it will be for an instance to reveal new information about the dataset, and lower are the chances of that instance being included. As a result, some instances may never be included in a candidate subset.

To address and fix the above concerns, randomizations of the instances from the original training dataset are created to dictate the order in which instances are to be

considered for inclusion to candidate subsets. To satisfying the desire for every instance to belong to at least one candidate subset, each instance is allowed to be the first element in a randomization of the instances. Then, a new candidate subset is built from every randomization and because subsets can be thought of as beginning empty and having a predictive contribution of zero, the first instance from every randomization will be included in its own subset. With each instance beginning one randomization, each instance will undoubtedly belong to at least one subset and the randomization of the remaining instances further guarantees the creation of a diverse population of candidate subsets

3.1.1 Classifier for Candidate Subset Construction

To accomplish this greedy addition of instances, the classification contribution of a subset needs to be calculated a large number of times, and in the interest of computational efficiency, a simple classifier should be chosen for the task. Using a simple classification algorithm for testing the contribution of a subset allows for the necessarily speedy construction and evaluation of the classifier built at every stage of a subset's construction.

In ad hoc experiments, the naïve Bayes and decision stump classifiers were tested for use in the construction of candidate subsets. Both classifiers are constructed through a few simple calculations and were thought to be good possibilities for testing the contribution of the candidate subsets. These classifiers were tested for the speed in which they helped construct subsets, and also for the quality of the subsets that they created. Through these experiments it was determined that the decision stump classifier was a better choice than the slightly more complex naïve Bayes classifier. The decision stump classifier returned subsets in a much faster manner than did the naïve Bayes classifier and the subsequent subsets

seemed to be of a comparable quality to those constructed by the other method. Therefore, the decision stump classifier was chosen to evaluate the contribution of the subsets in the subset creation process.

The decision stump classifier is a simplification of the decision tree, but instead of many branching points leading to the declaration of a class value, only a single branching occurs. From this single branching the majority class value from each branch, as determined by the training data, is assigned to instances following that specific branch, or path. Indeed a simple classifier suitable for the candidate subset creation process.

The single partitioning of data that defines a decision stump occurs along the attribute that will split the data into the most homogeneous groups. To find this attribute, a measurement known as the gain ratio is calculated for each attribute of the original training data. The gain ratio is an improvement upon a measurement known as information gain, and corrects for the information gain's bias to identify an attribute with a large number of possible values as being a better splitting attribute than an attribute with fewer possible values. To calculate the gain ratio for every attribute of a data set, and as a result to identify the splitting attribute to be used in a decision stump, the idea of information and information gain must first be introduced.

Information is a measurement of how many bits would be required to correctly classify instances found in a data set. If a data set is mostly homogeneous, in other words, if a dataset contains instances mostly belonging to the same class, the information would be fairly low because most instances could be classified as the majority class. Little extra knowledge would be required to identify the instances belonging to the minority. Simply

put, information gives an idea of how homogeneous a set of data is. The amount of information, *Info(D)*, needed to classify instances found in a data set *D*, with *m* possible class values, can be calculated by summing the products of the probability that an instance belongs to class *i*, with the base two log of that same probability;

$$Info(D) = \sum_{i=1}^{m} p_i log_2(p_i).$$

Here, p_i represents the probability that an instance belongs to the class i, and this can be estimated by dividing the number of instances in class *i* by the total number of instances in the dataset,

$$p_i \approx \frac{\#\ instances\ in\ class\ i}{total\ \#\ of\ instances}.$$

Of more interest than information, is the idea of information gain. Information gain measures the improvement of homogeneity in a data set from the imparting of some structure to that data set, possibly from the construction and implementation of a decision stump. The information needed to classify instances with known values for a specific attribute, $Info_A(D)$, is one of the measurements that must be calculated to find information gain. If an attribute *A* has *v* possible values, then this measurement can be found by summing together *v* different calculations. For each possible value of the attribute *A*, one of the *v* calculations is found by dividing the number of instances that have attribute *A* equal to *j* by the total number of instances in the dataset, and then multiplying this value with the amount of information needed to classify a dataset made of only the instances with attribute *A* equal to *j*;

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j),$$

where D_j is a collection of the instances in the dataset D with attribute A having value j, or

$$D_j = \{d \in D : A = j\}.$$

Then, the information gained from selecting attribute A to split the instances, *Gain(A)*, and alternatively the information gained from knowing the value of attribute A, can be calculated by subtracting the information required to classify the dataset when knowing the value of attribute A, from the information required to classify the unstructured dataset;

$$Gain(A) = Info(D) - Info_A(D).$$

Finally, the gain ratio can be calculated. This measurement of gain ratio is desired over information gain because of the information gain's inherit bias to attributes with a large number of possible values. If a dataset contained an attribute that was unique for each instance, perhaps an id number, a split on that attribute would lead to completely homogeneous but uninteresting groups. Information gain would identify this useless attribute as the best attribute on which to split the data, but in reality this attribute would be less than helpful for identifying interesting relationships. The gain ratio tries to correct for the misidentification of these types of attributes as being helpful, and does so by taking into account the number of possible values an attribute can assume. The gain ratio for splitting the instances on the values of an attribute A, *Gain Ratio(A)*, can be calculated by taking the information gained from selecting the splitting attribute to be A, and dividing it by a value known as the split information, *Split Info(A)*, which will be explained next;

$$Gain\ Ratio(A) = \frac{Gain(A)}{Split\ Info(A)}.$$

The split information for an attribute *A*, *Split Info(A)*, is the normalizing measurement that takes into consideration how many possible values there are for that attribute *A*. It is calculated by taking the negative value of the sum of *v* different values, one for each possible value of *A*. For each possible value of the attribute *A*, one of the *v* values is calculated by dividing the number of instances that have attribute *A* equal to *j* by the total number of instances in the dataset, and multiplying this with the base two log of the same ratio;

$$Split\ Info(A) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times log_2\left(\frac{|D_j|}{|D|}\right),$$

(Han et al., 2006).

Once able to calculate the Gain Ratio it is easy to identify the attribute that will be selected for use in the decision stump. The splitting attribute, *SA(D)*, of a dataset, *D*, can be identified by finding the attribute *A* that leads to the highest gain ratio;

$$SA(D) = arg\max_{A}\ Gain\ Ratio(A).$$

After having identified the split attribute for the decision stump, the class value that each branch of the decision stump leads to needs to be determined. The declared classes should be the same as the majority class found in the partitions of the data that relate to the different values, or branches, of the splitting attribute. Therefore, finding the class value to assign to an instance with split attribute equal to *j*, $V_{SA}(j)$, when classifying with a decision

stump, is calculated by finding the class value *i* that relates to the majority class of the training instances whose value for the split attribute, *SA(D)*, is equal to *j;*

$$V_{SA}(j) = arg\max_{i} \left|D_j^i\right|,$$

where, D_j^i , is a subset of the dataset D_j, where all of the instances have class value *I*, or

$$D_j^i = \{d \in D_j : Class = i\},$$

and where D_j is a subset of the original training instances with split attribute, *SA(D)*, equal to *j*, or

$$D_j = \{d \in D : SA(D) = j\}.$$

We now posses a method for creating the simple decision stump classifier for the candidate subsets and next will be introduced a method of finding the contribution of a subset with this classifier.

3.1.2 Candidate Subset Contribution

Upon the initialization of a new candidate subset, a decision stump classifier can be constructed, and the subset's contribution can be tested. When subsequent instances are considered for the addition to the new subset they are accepted if the contribution of the subset improves from their inclusion. However, if the addition of the instance fails to improve the predictive contribution of the subset, the instance is removed from consideration.

To calculate the predictive contribution of these different subsets, a decision stump classifier is built from the subset's instances, and the classifier is used to predict class values for the original training data. Then, the known class values from the original training data

are used to check the accuracy of these predictions. Whenever a prediction is correctly made the subset's contribution is increased by one.

It should be noticed that this contribution is based on the ability of the classifier to correctly classify the original training instances. This is considered a re-substitution error and is a problem that arises from using training data to test the accuracy of a classifier. The result of this is that the classifier is all too good at predicting the training instances, and incorrectly implies a high accuracy for the classifier. It is of course, rather obvious and stands to reason, that the accuracy of a classifier built from some training data and tested on that same training data should be high, because the classifier was induced from that specific dataset. Ideally, to circumvent this problem a second, test set, of data is used for testing classifier accuracy and gives a true estimate of the classifier's error. However, in the case of constructing candidate subsets, few instances are seen by the decision stump classifier and this error is ignored because of its presumably small impact on the subset construction.

3.1.3 Subset Creation Overview

The algorithm to be presented in Figure 1 can be used to create candidate subsets, and once the candidate subsets have been created, we have hopefully improved the quality of the space needed to be traversed to solve the instance selection problem. No longer do instances have to be considered for the inclusion or exclusion to the final training dataset with little idea of their worth. Instead, instances are considered for the final training dataset in high quality groups, with some knowledge of their predictive powers.

Step 1:

For $i = 1$ to n, where n is the number of instances in the training set T, and the number of candidate subsets to be created, initialize the contents of the subset $S^{(i)}$ as,

$S^{(i)} = \{x_i\}$, where x_i is the i^{th} element of the training set T.

Also, for every subset $S^{(i)}$, generate a randomization of the integers from 1 to n, less the integer i, to be used as an order from which to consider the addition of new instances to the candidate subset $S^{(i)}$. This set should be called $U^{(i)}$ where,

$U^{(i)} = \left\{u_1^{(i)}, u_2^{(i)}, \dots, u_n^{(i)}\right\}$, and where $u_j^{\,i}$ is the j^{th} instance of the randomization $U^{(i)}$.

Step 2:

For $i = 1$ to n, where n is the number of candidate subsets, and

*For $j = 1$ to $n - 1$, where n-1 is the number of instances to be considered for the addition to the subset $S^{(i)}$, consider the union of instance x_{uji} with subset $S^{(i)}$ by comparing the predictive contribution of $S^{(i)}$ without x_{uji}, and with x_{uji}, where x_{uji} is the $u_j^{\,ith}$ instance of the training set T. Therefore, the contribution of $S^{(i)}$ without x_{uji}, Contribution($S^{(i)}$), can be found by going to **Step 3** and supplying the subset $S^{(i)}$, and the contribution of $S^{(i)}$ with x_{uji}, Contribution($S^{(i)} \cup \{x_{uji}\}$), can be found by going to **Step 3** and supplying the subset $S^{(i)}$ with instance x_{uji}.*

Then,

$If\ Contribution\left(S^{(i)} \cup \left\{x_{u_{ji}}\right\}\right) > Contribution\left(S^{(i)}\right)$,

meaning, if the contribution of $S^{(i)}$ improves from the addition of x_{uji}, make the addition permanent;

$Then\ S^{(i)} = S^{(i)} \cup \left\{x_{u_{ji}}\right\}$.

Otherwise $S^{(i)}$ remains unchanged, meaning

$Else\ S^{(i)} = S^{(i)}$.

Step 3:

Initialize the contribution of the provided subset $S^{(i)}$ *as,*

$Contribution(S^{(i)}) = 0$, *and build the decision stump classifier from* $S^{(i)}$.

Then,

$For\ q = 1\ to\ n$, *where n is the number of instances in the original training data,*

if the class value predicted by the decision stump classifier, $V_{SA}(x_q^A)$, *for the* q^{th} *instance,* x_q, *which is known to have value* x_q^A *for the splitting attribute, is the same as the true class value for that instance,* x_q^C, *or*

$If\ V_{SA}(x_q^A) = x_q^c$,

then the contribution of $S^{(i)}$ *increases by one, meaning*

$Then\ Contribution(S^{(i)}) = Contribution(S^{(i)}) + 1.$

Otherwise, the contribution remains the same, or

$Else\ Contribution(S^{(i)}) = Contribution(S^{(i)}).$

When all of the instances have been tested,

$Return\ Contribution(S^{(i)})$, *meaning, return the contribution value calculated for the provided subset* $S^{(i)}$ *to* ***Step 2.***

Figure 1. Creating candidate subsets, pseudo code.

Ideally, the selection of the newly created candidate subsets will be made in such a way that the selected subsets will accurately represent the original training data. This model formulation closely resembles the set covering problem, in that a selection of candidate subsets must be made in such a way that the original training data is fully represented. Keeping the set covering problem in mind there are a variety of techniques and strategies for selecting the candidate subsets to be included in the final training dataset.

3.2 Candidate Subset Selection

The motivation for selecting a collection of instance subsets is to construct a new set of training data that will lead to the construction of a simple decision tree. Time has been taken to create high quality subsets of instances, and it is believed that some combination of these subsets will lead to an accurate, yet easy to interpret decision tree. Any number of techniques can be used to select the candidate subsets for the final training dataset and two are recommended below.

The first method considered for selecting candidate subsets is stringently based on the set covering problem. The objective of this method is to select subsets that accurately predict the original training data and that contain the fewest total instances. The second method is a relaxation of the set covering problem and has a highly non-linear objective function, the objective being to select the subsets that lead to the most accurate decision tree. These two methods will be implemented in the following chapter, and their ability to select training sets of data leading to simple decision trees will be evaluated.

3.2.1 Minimalistic Selection of Training Instances

The first method for selecting candidate subsets is a strict reformulation of the instance selection problem into the set covering problem. An integer program with a linear objective function is used to select a collection of candidate subsets that will have the lowest possible cost while still possessing the ability to predict all of the original training instances. The cost associated with an individual subset is the same as the number of instances contained in that subset and minimizing this cost relates to selecting as few instances as possible. The instances each subset can predict is calculated throughout the subset creation

process, and a subset is said to be able to predict a particular training instance if at some point in the subset's creation phase it correctly classified that instance.

This approach of selecting the minimum number of instances plays on the experimental results from Oates and Jensen (1997) that show the structure of a decision tree continues to grow from the addition of new instances to the training set, even after the accuracy has stopped improving. By selecting subsets so that the fewest possible instances are used, while still possessing the ability to predict the original training data, it is believed that a set of training data leading to a simple decision tree will be constructed.

Evaluating the cost C_i, associated with a candidate subset $S^{(i)}$, can be calculated by finding the size of that subset. Meaning,

$$C_i = \left|S^{(i)}\right|.$$

Finding the training instances that a subset $S^{(i)}$ can predict is a process best executed while constructing that subset. If at any point in the creation of the i^{th} subset, the subset is found to accurately predict an instance x_q, this should be noted in the set $Z^{(i)}$. Where $Z^{(i)}$ is,

$$Z^{(i)} = \left\{z_1^{(i)}, z_2^{(i)}, \ldots, z_n^{(i)}\right\},$$

and initially all z_j^i ,that is, the j^{th} elements of $Z^{(i)}$, are set equal to zero. Conveniently, Step 3 in Figure 1 can easily be modified to record the predictable training instances in the set $Z^{(i)}$. This is shown in Figure 2.

Step 3:

Initialize the contribution of the provided subset $S^{(i)}$ *as,*

$Contribution(S^{(i)}) = 0$, *and build the decision stump classifier from* $S^{(i)}$.

Then,

$For\ q = 1\ to\ n$, *where n is the number of instances in the original training data,*

if the class value predicted by the decision stump classifier, $V_{SA}(x_q^A)$, *for the* q^{th} *instance,* x_q, *which is known to have value* x_q^A *for the splitting attribute, is the same as the true class value for that instance,* x_q^C, *or*

$If\ V_{SA}(x_q^A) = x_q^c$,

then the contribution of $S^{(i)}$ *increases by one, meaning*

$Then\ Contribution(S^{(i)}) = Contribution(S^{(i)}) + 1.$

and also the q^{th} *instance is able to be predicted by the decision stump, so the value of* $z^{(i)}_q$ *is changed to one, or*

$z_q^{(i)} = 1.$

Otherwise, the contribution remains the same, or

$Else\ Contribution(S^{(i)}) = Contribution(S^{(i)}).$

When all of the instances have been tested,

$Return\ Contribution(S^{(i)})$, *meaning, return the contribution value calculated for the provided subset* $S^{(i)}$ *to* ***Step 2.***

Figure 2. Modified Step 3 for documenting the predictive power of candidate subsets, pseudo code.

Now having creadeted candidate subsets, documented the subsets' predictive abilities, and found their associated costs, a selection of instances can finally be made. To begin, a linear integer program will be formulated to select candidate subsets in such a way that the minimum number of instances is chosen, but also, so that all of the original training instances were at one predicted by a decision stump associated with the selected candidate subsets. Then, the selected subsets will be combined to form the final set of training data. This procedure can be followed in Figure 3.

Step 1:

Solve the following linear integer program.

Decision Variables:

$$y_i = \begin{cases} 1 \; if \; S^{(i)} is \; selected \; for \; the \; new \; training \; dataset \\ 0 \;\; otherwise \end{cases}$$

Linear Integer Program:

$$\min \sum_{i}^{n} C_i * y_i$$

$$s.t$$

$$Z^T * y \geq 1,$$

where Z^T is the transpose of the n different $Z^{(i)}$ vectors, and C_i is the size of the subset $S^{(i)}$. This linear integer program requires that all of the original training instances were at one point predicted by the selection of the different candidate subsets, as indicated by their associated Z vector, and does so in a manner that minimizes the cost C_i, or as that represents, the number of instances in the selected $S^{(i)}$ subsets.

Step 2:

Combine the selected subsets into a single set of final training data as follows.

The final set of training data T' is initialized as an empty set, or

$T' = \{\}$.

For $i = 1$ *to* n, *where n is the number of candidate subsets,*

if the integer program selects the i^{th} *subset to belong to the final set of training data by setting the value of* y_i *equal to one, meaning*

If $y_i = 1$,

then the final training data T', should be redefined to include the i^{th} *subset, meaning that*

Then $T' = T' \cup S^{(i)}$.

Otherwise the final set of training data will remain unchanged, or

Else $T' = T'$.

Figure 3. Minimalistic candidate subset selection for the final training dataset, pseudo code.

It is hoped that a decision tree learned from this new training set maintains a certain level of accuracy when compared to a decision tree built from the original training data, and it is also hoped that this decision tree is substantially smaller than a tree built from the original data. This method is implemented in the following chapter to look for these results.

3.2.2 Greedy Selection of Training Instances

Another possible method for selecting candidate subsets for the new training dataset *T'*, is to use a greedy search heuristic. This search heuristic will consider the inclusion of candidate subsets to the final training dataset based on the change in accuracy of a classifier built from the proposed final dataset. This accuracy will be estimated by inducing a C4.5

decision tree classifier from the proposed training instances, and then checking the correctness of predictions made on the original training dataset.

Again, the re-substitution error is ignored for the calculation of classifier accuracy because the number of original training instances being included in the final dataset is relatively small, and thusly the classifier has seen relatively little of the dataset. Therefore, candidate subsets will be selected for the final set of training data if including their instances in the final dataset improves the dataset's decision tree's accuracy. The order in which candidate subsets are considered for inclusion to the final dataset will influence its composition, so the order of the *n* candidate subsets is randomized to avoid any bias to any particular subset. The greedy selection of candidate subsets can be seen in Figure 4.

Step 1:

The final set of training data T' is initialized as an empty set, or

$T' = \{\}$, *and is assigned an initial accuracy, Accuracy(T'), of zero,*

$Accuracy(T') = 0.$

Also created is a randomization of the integers from 1 to n represented by U, where U is

$U = \{u_1, u_2, \ldots, u_n\}$, *and* u_i *is the* i^{th} *integer in U.*

$For\ i = 1\ to\ n$, *where n is the number of candidate subsets,*

if the accuracy of a decision tree built from the final training dataset, as tested on the original training dataset, improves from the inclusion of a subset

$S^{(ui)}$, *where* $S^{(ui)}$ *is the* i^{th} *integer of the set U, meaning*

$If\ Accuracy(T') \leq Accuracy(T' \cup S^{(ui)})$,

then include $S^{(ui)}$ *in the final training set T', or*

$Then\ T' = T' \cup S^{(ui)}$.

Otherwise the final set of training data will remain unchanged, meaning

$Else\ T' = T'$.

Figure 4. Greedy search heuristic candidate subset selection for the final training dataset, pseudo code.

3.3 Decision Tree Construction

Now that instances have been selected from the original training data to belong to the final set of training data, *T',* it is time to construct the simple decision tree. Decision trees will be constructed following the C4.5 algorithm as developed by J. Ross Quinlan (1986), and described in the following section. This algorithm is a good choice for creating simple decision trees because of its ability to incorporate real and discrete attributes into the tree, its ability to handle attributes with missing values, and its use of the Gain Ratio to determine which attributes should be used for partitioning the data. This rather standard algorithm for

constructing decision trees should result in a simple tree because the newly chosen training data contains only select, yet high quality, instances from the original data.

Given a set of training instances, the C4.5 decision tree algorithm builds a tree structure by attempting to divide the training instances into groups composed exclusively of instances from a single class. Instances are split into different groups based upon the values they exhibit for particular attributes, and this splitting is done in a greedy manner until nearly homogeneous groups are achieved. In an example decision tree, an attribute is selected to be the first spilt, or branching, of the tree whose associated tree structure splits the original training data into the most homogeneous groups, as determined by class value. Attributes are then selected whose structure best split the resultant groups into even more homogeneous groups, and this procedure continues until all groups are composed of instances from a single class, or until further partitioning of the data does not improve the homogeneity of a group.

The structures associated with each selected branching attribute are collected and organized into a tree that can then be used to classify instances with an unknown class value. This classification can be executed by simply following the branching of the tree in accordance to the instance's attribute values. When the instance reaches a leaf node, or the bottom of the tree, a class is assigned to that instance by labeling it the same class as the majority class of the training instances in the particular terminating group.

C4.5 decision trees are built in this same top down fashion and the attributes used to split the data are greedily selected based on a measurement previously introduced as the gain ratio. As it was with the decision stump classifier, the gain ratio is a measurement based on information gain. It may be recalled that information gain is a measurement of the

improvement in homogeneity of possibly partitioned groups from the further partitioning of those groups. As an improvement upon information gain, the gain ratio corrects for the bias of information gain towards attributes with a large number of possible values by normalizing the measurement in regards to the possible number of outcomes.

The attributes and structures relating to the highest values of the gain ratio are selected to be the next branching of the tree, and once an attribute is selected its structure is imposed on the training data, resulting in a partitioning of the training instances. This imposition makes future calculations of the gain ratio possible, and will ultimately lead to the training data being mostly separated into different groups based on their class values. When further branching results in homogeneous groups of the training instances, or no improvement in the gain ratio, branching is terminated and groups are not further partitioned. Leaf nodes are then constructed at the bottom of the tree and are used to declare the class value for an instance following the specific path leading to a particular leaf. The class value that a leaf node will assign to an instance is the same as the class value that the majority of the training instances in terminating group belong.

The gain ratio is a versatile measurement that can be used for real valued attributes by considering a structure that will binarily split the instances, and for such an attribute with n possible values, the gain ratio is calculated $n-1$ times. The observed values for the attribute of interest are put into either ascending or descending order, and then the gain ratio is calculated for every partitioning of the attribute that could be described in a “greater than” or “less than” rule. The gain ratio is can also be calculated for discrete attributes by considering a structure that will create groups of the training instances by splitting the data along every

possible value of that attribute. In addition to being able to accommodate different types of attributes, the C4.5 decision tree algorithm and gain ratio can also handle attributes with missing values by simply ignoring the corresponding instances when calculating that particular gain ratio.

Due to the fact that the partitioning of data in the C4.5 decision tree algorithm continues until all groups are homogeneous, or until further partitioning does not improve the gain ratio, post pruning is performed to simplify the decision tree. Post pruning helps the C4.5 decision tree avoid the problem of over fitting by examining the tree, and identifying portions of the tree that can be removed. For every internal node, or non leaf node, the sub-tree expanding from that node is considered for removal through a cost complexity analysis. The cost complexity measurement is a function on the size and error rate of a sub-tree. This measurement is calculated for both the sub-tree of the internal node and for the node itself under the assumption that the sub-tree has been collapsed to the node, essentially making the internal node a leaf. The more favorable of the two measurements dictates whether or not the sub-tree is removed. The repeated application of this procedure is used to prune the C4.5 decision tree.

With the ability to handle discrete and real valued attributes, a built in pruning operation, the use of the gain ratio for selecting splitting attributes, and the readily available implementations of the C4.5 decision tree algorithm, this algorithm is a good choice for building decision trees from our final set of training data *T'*.

3.4 Discussion

The methods described in this chapter can be used when a set of training data leads to the induction of an overly complex decision tree. The intent of this procedure is to select a subset of the original instances that contain the best information and that will allow a simple decision tree to be built that represents the underlying patterns or ideas represented in the dataset. To effectively solve this selection problem, a simple classifier was used to create candidate subsets composed of instances that work well in conjunction with one another, and these subsets become our problem's solution search space. The elements of this space can then be searched and combined to form a final set of training data. Two techniques for searching this space and making a final selection of instances have been developed, and in the subsequent chapter our methods will be tested for their validity in creating simple and easy to interpret decision trees.

CHAPTER 4 NUMERICAL RESULTS

4.1 Introduction

In this chapter, several experiments are conducted to help show the effectiveness of our approach to creating simple decision trees through instance selection. Our method of making candidate subsets, and then selecting training data from amongst those subsets, is implemented and then tested on a collection of nine datasets found in the UCI Machine Learning Repository. The results of our methods are judged on whether or not the resulting trees maintain an acceptable level of accuracy in comparison to the original, and also whether or not they produce a decision tree smaller than the original.

Accuracy is an important indicator of our algorithm's performance because one premise of our work is that some instances contained in a dataset are more important than others. We believe that the unimportant instances in a dataset only serve to complicate the decision tree's structure, hampering the performance of decision tree induction and the future capabilities of the decision tree. Therefore, if our algorithm results in a large reduction of a tree's accuracy, then we have selected too few instances, and have neglected to select some of the more important pieces of data.

In our experimental results, the accuracy of a tree is calculated on a set of testing data composed of $1/3^{rd}$ of the available instances. The remaining $2/3^{rds}$ of the instances are used as the original training data and will provide the instances to be used in the creation of the original decision tree and the instances for the subsequent decision trees built through instance selection. This "$1/3^{rd}$ holdout" method of accuracy testing ensures that the decision

tree has not previously seen the data that it will be predicting, and gives a true representation of the tree's error rate.

One potential danger of calculating the accuracy of a decision tree with a $1/3^{rd}$ hold out is that some information will not be available for the tree to learn from. Denying a decision tree this extra information may inhibit its performance and affect the tree's ability to predict future instances. However, with the nine datasets selected for this study, decision trees built from $2/3^{rds}$ of the data are nearly identical to decision trees built from all of the data. This implies enough information is present in $2/3^{rds}$ of the data to make a $1/3^{rd}$ holdout an acceptable method for calculating the tree's error rate.

The size of the decision tree is another important metric of our algorithm's performance. Decision tree size is calculated by adding together the number of leaf and internal nodes, helping to reveal the complexity of the tree. In a decision tree, internal nodes represent the number of decision points, and the leaf nodes represent the number of possible paths any particular instance can follow on its classification route. Intuitively, the fewer number of decision points and possible paths in a decision tree, the simpler the tree. Similarly, the fewer paths and decision points a human analyzer has to view in a tree, the easier it will be for them to interpret. This reasoning reaffirms that the size of a decision tree is a good measurement of its complexity, and is another measurement to reveal if our algorithm is creating interpretable trees.

To begin the testing of our algorithm on the nine datasets, each dataset will have a C4.5 decision tree learned from the original training data, and this tree will be used as a baseline case for the size and accuracy of an unimproved decision tree. Then, because of the

randomness present in our algorithm, the whole instance selection procedure will be conducted 30 separate times. Meaning, we will build candidate subsets, select amongst those subsets for a final set of training data, and build a C4.5 decision tree from that selection, 30 different times. The results from these 30 decision trees will be averaged together, and a 95% confidence interval will be calculated to show the variability in the accuracy and size of the reduced trees. The 95% confidence interval will be constructed assuming that the values for accuracy and size of decision trees induced from a particular dataset are normally distributed around their respective means.

To test the robustness of our instance selection procedure, nine datasets were selected for testing from the UCI Machine Learning Repository. The nine datasets were selected to ensure our procedure was exposed to a wide variety of different datasets with a variety of different features. Datasets were selected that had strictly categorical variables, strictly real valued variables, combinations of variable types, missing values, a large number of instances, a small number of instances, and so on and so forth.

4.2 Evaluation of Decision Tree Quality

In the previous chapter, two methods were proposed for selecting an improved set of training instances. The first method was a minimalistic approach to selecting instances and the second was an approach to select instances that maximized the accuracy of a resultant decision tree. The minimalistic selection approach selected candidate subsets for the final set of training data with the intention that a minimum number of instances would be selected, but also, with the requirement that a decision stump built from the selected candidate subsets was at one point able to predict all of the original training instances. The second selection

method chose the final instances through a greedy search heuristic, whose objective was to maximize the accuracy of a resultant decision tree. In this method, candidate subsets were added to the final set of training data if doing so improved the accuracy of a decision tree induced from that set. These two selection methods lead to the creation of two different types of trees, and therefore, a comparison is made between the performances of the final training instances as selected by each method.

4.2.1 Results from the Minimalistic Selection Approach

	Original		Minimalistic Selection			
	Accuracy	Size	Accuracy (95% CI)	Accuracy (Mean Difference in % Points)	Size (95% CI)	Size (Mean Difference)
Wisconsin Cancer	92.7%	19	[49%,100%]	-13	[2,4]	-16
Mushrooms	99.7%	23	[49%,53%]	-49	[1,1]	-22
Voting Data	93.8%	5	[46%,100%]	-17	[1,4]	-3
Car Data	88.9%	145	[29%,81%]	-34	[1,7]	-142
Thyroid	99.1%	19	[89%,98%]	-5	[1,6]	-15
Diabetes	72.7%	29	[47%,75%]	-12	[2,4]	-26
Blood Drive	74.4%	9	[45%,88%]	-8	[1,5]	-7
Mammogram	78.5%	19	[45%,63%]	-24	[1,5]	-17
Balance	81%	63	[31%,74%]	-22	[2,6]	-59

Table 1. Minimally selected candidate subsets result in very small decision trees but sacrifice accuracy.

By performing an analysis of Table 1 it can be seen that the minimalistic approach to instance selection results in very small decision trees, on average reducing tree sizes from the observed datasets by about 85%. Unfortunately this huge reduction in tree size comes at the sacrifice of accuracy. On average, the accuracy of a found decision tree was reduced by over 20 percentage points, too large of a reduction when the goal of tree simplification is to have decision trees that are helpful to policy makers.

This large reduction in decision tree accuracy stems from the fact that very few instances are selected to be in the final set of training data. In fact, most of the final training datasets selected only 1% of their original training instances. The poor performances of the decision trees built from these reduced datasets indicate that instances describing important concepts from the original datasets are missing. This problem of too few instances originates from the procedure used to select the candidate subsets.

The minimalistic selection method chooses candidate subsets for the final dataset in such a way that the minimum numbers of instances are selected, but also requires that the selected subsets were at one point able to predict all of the original training instances with a decision stump. The failure in this selection procedure is that as subsets have more instances added to them, and as subsets are combined for the final training dataset, the subsets' predictive powers change. What a subset was once able to predict with a decision stump does not necessarily indicate what it will be able to predict when combined with other instances. This method of evaluating a candidate subset's worth leads to the selection procedure choosing candidate subsets that in actuality may not be any more beneficial than any other candidate subset.

Despite the large reductions in accuracy and the seemingly arbitrary selection of candidate subsets, the minimalistic selection approach does do a great job of reducing the size of decision trees. However, with little data in the final training dataset, and subsequently, with fewer situations to represent, reductions in tree size are of little surprise. The Mushrooms dataset is a great example for illustrating the danger of oversimplification at the sacrifice of decision tree accuracy.

The Mushroom dataset can be used to induce a decision tree that uses the physical characteristics and environmental surroundings of a mushroom to decide if the mushroom is safe for human consumption. The original decision tree from this dataset is able to do this classification correctly nearly 100% of the time, but provides some rather complex reasoning to reach these accurate conclusions. Therefore, the motivation for reducing the size of this decision tree is to provide foragers with an easily remembered rule for collecting edible mushrooms. Unfortunately, most of the reduced decision trees made with this particular candidate subset selection method are too simple and tell the forager to always eat the mushroom. Doing so completely ignores the fact that poisonous mushrooms exist. If the insight from these decision trees were put into practice, it would almost certainly end in disaster.

The minimalistic approach to selecting candidate subsets ignores the important component of accuracy when simplifying decision trees for the use of policy makers. This selection method attempts to select the fewest instances possible because as presented in the literature review, experimental results by Oates and Jensen (1997) have shown that the structure of a decision tree continues to grow from the addition of new instances to the

training set, even after the accuracy has stopped improving. Unfortunately, the minimally selected instances are selected with little idea of their actual predictive contribution to a decision tree classifier. Therefore, the next heuristic method for selecting candidate subsets attempts to do a better job of considering the desire for accurate decision trees in the tree simplification process.

4.2.2 Results from the Greedy Selection Approach

	Original		Greedy Selection			
	Accuracy	Size	Accuracy (95% CI)	Accuracy (Mean Difference in % Points)	Size (95% CI)	Size (Mean Difference)
Wisconsin Cancer	92.7%	19	[91%,96%]	+1	[3,9]	-13
Mushrooms	99.7%	23	[98%,99%]	-1	[5,20]	-11
Voting Data	93.8%	5	[95%,95%]	+1	[3,3]	-2
Car Data	88.9%	145	[65%,81%]	-16	[1,17]	-140
Thyroid	99.1%	19	[99%,99%]	0	[6,13]	-10
Diabetes	72.7%	29	[69%,77%]	0	[3,23]	-16
Blood Drive	74.4%	9	[70%,78%]	0	[2,11]	-2
Mammogram	78.5%	19	[73%,79%]	-2	[2,16]	-10
Balance	81%	63	[75%,81%]	-3	[9,20]	-49

Table 2. Greedily selected candidate subsets result in adequate decision tree accuracy and a moderate reduction in tree size.

Decision trees built from the candidate subsets selected by the greedy selection approach have promising results. The objective of this search heuristic was to select subsets whose instances would maximize the resulting decision tree's accuracy, and the method

appears to have indeed selected good candidate subsets. An interesting observation from Table 2 is that the majority of the observed datasets performed well with this method for instance selection, despite the wide variety of features represented in the different datasets. One exception to this rule is the Car Dataset which had a poor performance, but this will be discussed in detail at a later time. On average, decision tree's for the observed datasets were reduced by 57%, but only lost two percentage points of accuracy. Unlike the final datasets selected by the minimalistic approach to instance selection, these datasets were on average reduced by 95%, indicating that the greedy selection procedure selected more of the original training instances.

The Voting Dataset is one dataset that performed very well under the greedy method of selecting candidate subsets. The Voting Dataset describes sixteen different votes made by individuals in the United States' congress, and then reveals to what political party the people belong. The greedy search heuristic led to the creation of decision trees that were of size three, a reduction of two from the original tree of size five. The instance selection procedure identifies a single vote to split political parties, and shows an improvement in accuracy of one percentage point. This improvement in accuracy, reduction of tree size, and the reduction in the number of instances used to induce the tree, indicate that the original decision tree was over fitted to the training data, and also, that some instances in the training data were more important than others. By only selecting the most important instances to learn from, we successfully reduced the size of the decision tree while allowing better generalization to unseen instances.

Another benefit of the good performance of our instance selection procedure on this dataset was that interesting instances were identified. A decision tree induced from these interesting instances revealed that there was a single vote to best split the political parties, a fact that was missed when inducing a tree from the whole dataset. The instance selection procedure allowed for the identification of the "freezing of physician's fees" as being an issue that the two political parties took separate stances on. Further structural insights from groups of interesting instances could be made for other datasets for which the instance selection procedure works well, and these insights are what policy makers will find beneficial.

The underlying reason for the good performance of the instance selection algorithm on the Voting Dataset is that the best tree classifier of this dataset is in actuality, a decision stump. The construction of our candidate subsets is rooted in the performance of a decision stump, so it is not surprising that our instance selection algorithm consistently found this best tree of size three. With candidate subsets continuing to grow until the accuracy of a decision stump stops improving, and because it is possible to achieve the best tree classification of this data with a decision stump, it is not implausible that some of our candidate subsets also achieved this best classification. After the construction of candidate subsets, our selection procedure had high quality elements to choose from, and very little searching had to be performed to find a good selection of final training data. Our greedy selection procedure always found the best tree classifier of this dataset.

Contrasting the promising performance of our instance selection procedure with the Voting Dataset is the Car Dataset. The Car Dataset is a collection of attributes describing the

condition and features of different vehicles, and also a class value that relates the desirability of those vehicles for purchase. Instance selection on this dataset leads to a loss of almost 16 percentage points in accuracy, and perhaps unimportantly a reduction of 140 in tree size. The results of our instance selection procedure on this dataset are not beneficial because of the large reduction in decision tree accuracy. When analyzing the instance selection process associated with the Car Dataset, several observations can be made. The first observation is that the tree built from the original training data is of size 145, and the second is that the majority of the candidate subsets contain only a single instance.

A decision tree of size 145 is a very large and very complex tree, suggesting perhaps, that a decision tree is not a well suited classifier for this dataset. Decision trees do a good job of identifying the boundaries between classes when the boundary is based on linear combinations of attributes, but decision trees have a down falling when those boundaries are defined by non-linear combinations of attributes. Decision trees are somewhat able to overcome this shortcoming but are forced to do so by creating a large number of decision points, in essence defining a stepwise function to separate the data along a non-linear boundary. This practice may very well be happening in the Car Dataset and can be noted by the large tree size and by the poor performance of the decision tree.

The second observation made for the Car Dataset is that the majority of candidate subsets contain a single instance. The subsets that contain multiple instances only do so to improve the accuracy of their decision stump classifier to that of the accuracy of those subsets containing single instances. This accuracy also relates to the decision stump simply predicting the majority class, and no decision stump has improved upon this method. With

such poor performance of the decision stump classifier in grouping together helpful instances, it is obvious that the structure of the data is more complex than the decision stump can handle. As if to make matters worse, the subset selection procedure only considers the addition of one subset at a time, and in the majority of cases, this will result in only a single instance being considered for the addition to the final training dataset. Again, because a decision tree has difficulty with this dataset, adding a sole instance to the final training set rarely improves the accuracy of the tree, so the instance is passed by. Hardly ever are enough instance accumulated to reveal the complex structure of the data to the decision tree. The result is the poor performance of the instance selection procedure.

It is hypothesized that building candidate subsets with a classifier more suited to the Car dataset will improve the performance of the instance selection procedure. A different classifier may be able to take advantage of the different structures in the data, and may lead to the compilation of better candidate subsets. With candidate subsets composed of instances that are actually important, the greedy search heuristic may be able to create better decision trees.

One last observation to be made from the results of the greedy search heuristic is that in some cases the performance of a decision tree is above and beyond that of the original. For example, instance selection performed on the Wisconsin Cancer dataset, a dataset that describes the physical characteristics of different masses and reveals whether or not they are cancerous, has multiple cases where the resulting decision tree has an accuracy of 96% and a tree size of only seven. This is an improvement of four percentage points over the original tree and a reduction of 12 in decision tree size. The Diabetes and Blood Drive datasets

exhibit the same phenomenon of occasionally finding very good decision trees. This occasional discovery of very good decision trees may be explained by the way candidate subsets are being selected.

A greedy search heuristic is used to select the candidate subsets from which we build decision trees, so it is very possible that we are not always finding the best decision trees from our instance selection procedure. Multiple datasets show cases where a very good decision tree is occasionally found, implying that our search procedure is getting stuck in local optimums when these solutions are not found. Search algorithms such as simulated annealing or tabu search, allow back tracking in the process of solution searching and make the avoidance of such local optimums easier. A more sophisticated search procedure may allow for these better decision trees to be found more consistently.

One particular concern of our instance selection procedure is that the improvements in decision tree size and the differences in tree performance do not stem from the instance selection procedure but are simply a result of the reduction in data presented to the decision tree algorithm. This possibility is explored in the following section by evaluating the perceived worth of our candidate subsets.

4.3 Subset Evaluation

The initial aim of our instance selection procedure was to improve upon the previous methods of instance selection by increasing the quality of the search space needed to be traversed to find an acceptably simple, yet accurate decision tree. This goal was approached by transforming the elements of the old search space from single instances to subsets of instances that we believed would work well in conjunction with one another. However,

another possible approach to improving upon the previous methods of instance selection would be to reduce the size of the solution's search space, making the search for acceptable solutions quicker, and also giving a suggestion towards the quality of our created subsets. If removing a selection of candidate subsets from the search space does not inhibit the finding of an acceptable decision tree, then the remaining subsets are of a high enough quality to compensate for the loss of the removed candidate subsets.

Initially, candidate subsets were created for every instance, ensuring that every instance had the opportunity to be included in the final training data. Afterwards it was noticed that because the candidate subsets often included more than one instance, some instances were given more than a single opportunity of joining, and in an effort to reduce the solution's search space, we eliminated the candidate subsets that corresponded to instances already included in previous subsets. Meaning, a candidate subset starting with a particular instance would not be included in the search space if that instance had already been included in a previously constructed subset. This certainly did not eliminate the chance that an instance could belong to more than one subset, but it did eliminate at least one duplicate opportunity for joining the final dataset.

	Original		Greedy Selection (Reduced # of Subsets)				
	Accuracy	Size	# Subsets Reduced By	Accuracy (95% CI)	Accuracy (Mean Difference in % Points)	Size (95% CI)	Size (Mean Difference)
Wisconsin Cancer	92.7%	19	58%	[91%,96%]	0	[2,9]	-13
Mushrooms	99.7%	23	67%	[96%,100%]	-2	[5,18]	-11
Voting Data	93.8%	5	66%	[95%,95%]	+1	[3,3]	-2
Car Data	88.9%	145	22%	[65%,79%]	-17	[1,24]	-139
Thyroid	99.1%	19	22%	[98%,99%]	0	[6,11]	-11
Diabetes	72.7%	29	65%	[69%,75%]	0	[1,20]	-18
Blood Drive	74.4%	9	44%	[70%,78%]	0	[1,11]	-3
Mammogram	78.5%	19	70%	[74%,77%]	-3	[1,14]	-11
Balance	81%	63	59%	[72%,80%]	-5	[5,20]	-51

Table 3. When considering a reduced number of candidate subsets the instance selection procedure still appears to perform rather well.

The results for instance selection as performed with the reduced search space are summarized in Table 3. These favorable results indicate that the number of candidate subsets can indeed be reduced while still allowing for a decision tree with a good reduction in tree size and acceptable accuracy to be found. The least a solution search space was reduced by for the observed datasets was 22%, while the most was 70%, but in both cases the procedure was still able to achieve results similar to the results obtained with a search performed over all of the candidate subsets. This performance may indicate that we have improved the quality of the solution's search space, and that our candidate subsets may very well contain groups of high quality instances. However, we must be careful with such

optimism because these results may still simply be a result of a reduction in data presented to the classification algorithm. After all, we have not tested the performance of instance selection on these datasets with an unimproved solution search space. However, regardless of the implications from not testing this other search space, it may be a worthwhile endeavor to investigate exactly how many candidate subsets need to be included in our search space to still allow a favorable decision tree to be found.

The results from the previous experiments have yet to convince us that any of the observed benefits from our instance selection procedure are indeed from an improved solution search space. Therefore, we want to test to see if the elements of our search spaces are better than the elements of search spaces whose elements are the single instances contained in the original sets of training data. The benefits we are observing could still simply be from our search procedure and the reduced amount of data presented to the classification algorithm, not a result of our candidate subsets being composed of instances that together work well for classification.

To help ensure that the benefits noticed from the instance selection procedure are indeed from an improvement of the solution's search space, random sets of instances the same size as the minimalistic approach's final datasets are created. Decision trees are then induced from the 30 different datasets as selected by the random and minimalistic approaches and the metrics from the different decision trees are averaged together to allow a comparison of the two methods.

The reasoning for comparing the performance of randomly selected instances and the candidate subsets selected with the minimalistic selection approach is that the candidate

subsets selected by the minimalistic approach are not selected very intelligently. Rather, it seems the candidate subsets are chosen arbitrarily because of the lack of an adequate measurement to reveal their usefulness. Therefore, we only know that the training instances selected with the minimalistic approach are selected from amongst the candidate subsets, and that this was done poorly. We believe that if our candidate subsets are indeed composed of intelligently selected instances then they should outperform the randomly selected instances, and that this improvement is not from the careful selection of candidate subsets. On the other hand, if the selected candidate subsets do not outperform the randomly selected instances it would be safe to say the improvements noticed in decision tree size and accuracy are not from the consideration of candidate subsets.

	Original			Minimalistic			Data Reduction		
	Accuracy	Size	# Instances	Accuracy (Mean)	Size (Mean)	# Instances (Mean)	Accuracy (Mean)	Size (Mean)	# Instances
Wisconsin Cancer	93%	19	466	80%	3	6	77%	2	6
Mushrooms	100%	23	677	51%	1	3	50%	1	3
Voting Data	94%	5	290	76%	2	5	63%	1	5
Car Data	89%	145	1152	55%	3	8	69%	1	8
Thyroid	99%	19	1866	94%	4	16	92%	2	16
Diabetes	72%	29	512	61%	3	6	58%	2	6
Blood Drive	74%	9	498	67%	2	5	68%	2	5
Mammogram	79%	19	640	54%	2	6	55%	2	6
Balance	74%	63	416	52%	4	8	53%	3	8

Table 4. Our instance selection procedure created more accurate decision trees than the randomly selected training data in most cases, and trees still exhibited reductions in size.

From an analysis of Table 4, it can be seen that on average the randomly selected training instances result in decision trees 90% the size of the original, and exhibit a 22 percentage point reduction in accuracy. These numbers are to be compared to the decision trees constructed through the minimalistic selection procedure that were on average 85% the size of the original tree, and only represented a 20 percentage point reduction in accuracy. This difference indicates that for the observed datasets selecting instances that belong to the same candidate subset for the final training data is a better option than the random selection of instances. It also suggests that some of our candidate subsets do indeed contain groups of important instances.

A particular observation to be made from Table 4 is that the decision trees built from the selected candidate subsets of the Voting dataset outperforms the accuracy of the randomly selected instances by 13 percentage points. It was mentioned earlier that the best tree classifier of the Voting dataset is in fact a decision stump, and because of this, it is not surprising that candidate subsets created for the Voting dataset are beneficial. We believe that the candidate subsets built for this dataset are an improvement upon randomly selected instances and this is because the decision stump classifier is helpful in the classification tasks, this has led to the creation of candidate subsets composed of instances that together work well for classification. It is likely that the solution search space for this dataset has been improved.

Other decision trees built from minimally selected candidate subsets also outperform the randomly selected data, but they do not do so by such a large margin. The accuracy of decision trees induced from the candidate subsets of the Wisconsin Cancer, Diabetes, and Thyroid datasets only improve by two or three percentage points from the randomly selected instances in our experiments. It is believed that these candidate subsets exhibit this slight improvement upon randomly selected instance because the decision stump classifier is only moderately helpful in the classification task. Perhaps a classifier better suited to these datasets would lead to the creation of even better candidate subsets and an even higher quality solution search space.

The candidate subsets from the remaining datasets result in decision trees that show little to no improvement over the randomly selected instances, or even worse, are outperformed by the randomly selected instances. This shows that the majority of these

candidate subsets are not composed of instances that together work well for classification, but instead, are composed of instances that do not exhibit a strong relationship. We believe that the benefits observed from instance selection with these datasets are only from the careful reduction of instances, and not from an improvement in the solution search space. With the decision stump classifier creating good candidate subsets for some datasets and not so helpful candidate subsets for other datasets, it becomes apparent that the construction of the candidate subsets is very important.

The main contribution of this instance selection procedure is intended to be an improved solution search space for finding simple decision trees through instance selection. If candidate subsets are not an improvement upon randomly selected instances, then the procedure has failed by not improving the quality of the search space. However, we should not limit ourselves with the current method for creating candidate subsets. Knowing more information about the nature of a dataset would certainly allow for the better selection of a simple classifier for use in the candidate subset creation process. Using a better classifier could lead to the better identification of a dataset's useful instances and to the creation of more beneficial candidate subsets, resulting in even more improved solution search spaces.

The Car dataset is one such dataset that may benefit from candidate subsets being constructed with a different classifier. A decision stump learned from the original training instances of this dataset obtains only 70% accuracy when classifying instances. Alternatively the naïve Bayes classifier is able to obtain 82% accuracy with this dataset. This suggests that the naïve Bayes classifier may lead to better candidate subsets for the Car dataset. Another opportunity to improve candidate subsets may lie with the Balance dataset from which a

decision stump can only correctly classify 63% of the instances but with which the naïve Bayes classifier can correctly classify 90% of the instances. Again, the naïve Bayes classifier could lead to the creation of candidate subsets that are truly made of important instances.

Regardless of the manner in which the important instances of a dataset are identified, whether it is from creating candidate subsets of instances with a decision stump or a naïve Bayes classifier, learning a decision tree from those important instances could be beneficial and useful. If instances are useful for classification they should contain descriptive information about the original dataset and a decision tree may reveal that information. Further investigation should be performed to delve into the importance of the classifier choice in the creation of candidate subsets and the benefits of inducing a decision tree from those candidate subsets.

CHAPTER 5 CONCLUSION AND FUTURE WORK

The premise of this thesis is that the intelligent selection of training instances could be used to increase the interpretability of decision trees through the reduction of decision tree size. Specifically, the objective was to improve the quality of the instance selection problem's solution search space through the creation of candidate subsets of instances that would work well together for classification. Experimental results have shown that indeed, for some datasets instance selection can be used to improve the interpretability of complicated decision trees without the sacrifice of decision tree accuracy, and that the improvement of the instance selection problem's solution search space is possible.

The contribution of this thesis is an effective approach to improving the instance selection problem's solution search space. Improving the elements of this search space allows heuristic search algorithms to consider groups of important instances for the addition to the final training dataset rather than just individual instances for which we have little prior knowledge of their classifier contribution. We believe that the identification and grouping of these important instances for the improvement of the solution search space is very beneficial to solving the instance selection problem because less searching will be required to find acceptably simple and accurate decision trees.

We also believe that it is important for future research to solve some of the issues raised in this thesis. One issue that certainly demands further consideration is how to improve the construction of candidate subsets. We observed that the decision stump

classifier successfully built helpful candidate subsets for some datasets, but failed to do so for others. However, it was postulated that it may be possible to improve these poor candidate subsets by using a different classifier in the subset construction process. The resolution of this problem would widen the breadth of datasets for which this procedure is effective and further aid in the discovery of useful information from databases.

A second issue raised in this thesis was the possibility of reducing the number of candidate subsets to be included in the instance selection problem's solution search space. Experimental results showed that it was not necessary to create a candidate subset for every instance, and that a good decision tree could still be found from searching over this reduced space. Additional research should look into the quality of candidate subsets and the relation between the number of subsets required for the discovery of a useful and simplified decision tree. If the solution search space for instance selection can be effectively reduced this will allow for a speedier instance selection algorithm, and again, widen the breadth of datasets for which this method is effective.

Yet another worthwhile endeavor identified in this thesis is to improve the search algorithm used to select candidate subsets. For a few select datasets, the greedy search heuristic occasionally found very favorable decision trees. These decision trees exhibited an improvement in accuracy and a reduction in size over that of the original tree. In these cases the instance selection procedure was most certainly able to find the important instances of the dataset, and a more involved searching technique may allow for these good decision trees to be found more often.

Lastly, an important question is yet to be answered. How much benefit is there from analyzing the important instances of a dataset? In the case of the voting dataset, analyzing only the important instances led to the discovery of an issue that almost completely split political parties. How can the structural insights made from the important instances of other datasets be taken advantage of for the benefit of different policy makers?

BIBLIOGRAPHY

Leo Breiman, Jerome Friedman, Richard Olshen, Charles Stone (1984), "Classification and Regression Trees", New York, NY: Chapman & Hall.

J.R. Quinlan (1986), "Induction of Decision Trees", *Machine Learning*, Vol. 1, No. 1, pp.81-106.

J.R. Quinlan (1992), "C4.5 Programs for Machine Learning", San Francisco, CA: Morgan Kaufmann.

Jiawei Han, Micheline Kamber (2006), "Data Mining Concepts and Techniques", San Francisco, CA: Morgan Kaufmann.

Taichirou Endou, Qiangfu Zhao (2002), "Generation of Comprehensible Decision Trees Through Evolution of Training Data", *in proceedings of the 2002 Congress on Evolutionary Computation*, pp. 1221-1225

M. Sebban, R. Nock, J.H. Chauchat, R. Rakotomalala (2000), "Impact of learning set quality and size on decision tree performances", *International Journal of Computers, Systems and Signals*, Vol. 1, No. 1, pp.85-105.

Tim Oates, David Jensen (1997), "The Effects of Training Set Size on Decision Tree Complexity", *in proceedings of the Fourteenth International Conference on Machine Learning.*

Thomas Hancock, Tao Jiang, Ming Li, John Tromp (1996), "Lower Bounds on Learning Decision Lists and Trees", *Information and Computation*, Vol. 126, No. 2, pp.114-122.

J.R. Quinlan (1987), "Simplifying Decision Trees", *International Journal of Man-Machine Studies*, Vol. 27, No. 3, pp.221-234.

T. Niblett (1987), "Constructing Decision Trees in Noisy Domains", in I. Bratko and N. Lavrac (Eds.), *Progress in Machine Learning*. England: Sigma Press.

John Mingers (1987), "Expert Systems—Rule Induction with Statistical Data", *The Journal of the Operational Research Society*, Vol. 38, No. 1, pp.39-47.

Floriana Esposito, Donato Malerba, Giovanni Semeraro (1997), "A Comparative Analysis of Methods for Pruning Decision Trees", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 5, pp.476-491.

George John, Ron Kohavi, Karl Pfleger (1994), "Irrelevant Features and the Subset Selection Problem", *in proceedings of the Eleventh International Conference on Machine Learning*, pp.121-129.

Mark Hall (2000), "Correlation-based feature selection for discrete and numeric class machine learning", Hamilton, N.Z: Dept. of Computer Science, University of Waikato.

George John (1995), "Robust Decision Trees: Removing Outliers from Databases", *in proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pp.174-179.

Carla Brodley, Mark Friedl (1996), "Identifying and Eliminating Mislabeled Training Instances", *in proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp.799-805.

Peter Hart (1968), "The Condensed Nearest Neighbor Rule", *IEEE Transactions on Information Theory (Corresp.)*, Vol. IT-14, pp.515-516.

G. L. Ritter, H. B. Woodruff, S. R. Lowry, T. L. Isenhour (1975), "An Algorithm for a Selective Nearest Neighbor Decision Rule", *IEEE Transactions on Information Theory*, Vol. 21, No. 6, pp.665-669.

Dennis Wilson (1972), "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 2, No. 3, pp.408-421.

D. Randall Wilson, Tony Martinez (2000), "Reduction Techniques for Instance-Based Learning Algorithms", *Machine Learning*, Vol. 38, pp.257-286.

Taichirou Endou, Qiangfu Zhao (2002), "Generation of Comprehensible Decision Trees Through Evolution of Training Data", *in proceedings of the 2002 Congress on Evolutionary Computation*, pp.1221-1225.

Jose Ramon Cano, Francisco Herrera, Manuel Lozano (2003), "Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study", *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 6, pp.561-575.

Jose Ramon Cano, Francisco Herrera, Manuel Lozano (2006), “Evolutionary Stratified Training Set Selection for Extracting Classification Rules with Trade off Precision-Interpretability”, *Data & Knowledge Engineering*, Vol. 60, pp.90-108.

Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science

O. L. Mangasarian, W. H. Wolberg (1990), "Cancer diagnosis via linear programming", *SIAM News*, Vol. 23, No. 5, pp.1-18.

Yeh I-Cheng, Yang King-Jang, Ting Tao-Ming (2008), "Knowledge discovery on RFM model using Bernoulli sequence, *Expert Systems with Applications*, Vol. 36, No. 2, pp. 5866-5871.

M. Elter, R. Schulz-Wendtland, T. Wittenberg (2007), “The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process”, *Medical Physics*, Vol. 34, No. 11, pp.4164-4172

ACKNOWLEDGEMENTS

At this time I would like to extend my sincere gratitude to my plan of study committee members, they have truly helped guide me through this research experience. To begin, I would like to thank Dr. Sigurdur Olafsson for his constant encouragement, reassurance, and commitment to my success. Additionally, I would like to thank Dr. Olafsson for supplying the interesting and challenging problem that formed the basis my thesis. Finally, I would like to thank Dr. Sarah Ryan and Dr. Dianne Cook for their support, input, and for their contribution to both my education and to the completion of this thesis.