Model based classifier improvement through the generation and selection of instance candidate subsets

Walter Bennette (corresponding author)[1], Sigurdur Olafsson[2]
Dept of Industrial and Manufacturing Systems Engineering, Iowa State University, Ames, IA 50010
[1]bennette@iastate.edu
[2] olafsson@iastate.edu

Keywords:
Instance Selection
Training Set Selection
Naive Bayes
Decision Tree
Logistic Regression

Abstract

Data preprocessing to improve the quality of data utilized by classification algorithms is an important step in knowledge discovery that can be performed in part by instance selection algorithms. In this paper, concepts from column generation and the set covering problem are used to construct high quality subsets of training instances to be used as decision variables in such instance selection procedures. A greedy and Nested Partitions selection procedure are used with the constructed decision variables and show that improvements of classifier accuracy are possible for a variety of datasets using decision tree, naïve Bayes, and logistic regression classifiers.

1. Introduction

In today's technological world, huge amounts of data are being collected and stored to be exploited through any number of knowledge discovery processes. Knowledge discovery processes are typically implemented in part, as data mining algorithms, but the quality of knowledge that these algorithms are able to obtain is highly dependent on the quality of the data from which they are learned, or trained upon [1]**.** In practice, an algorithm's training data can often contain useless, misleading, or superfluous information that only serves to confuse the algorithm and negatively affect its performance [2]. Therefore, it is sometimes desirable to perform certain data preprocessing methods and improve the quality of the training dataset to facilitate better knowledge discovery.

Data preprocessing for dataset improvement can be grouped into the areas of data cleaning, attribute selection, and instance selection. Data cleaning is a well studied practice and ensures that a data mining algorithm is learning from the correct information, utilizing methods such as outlier identification and the correction of erroneous data entries. Attribute selection is another well studied practice, and it improves the usefulness of a training dataset by finding good subsets of the data's original attributes from which to learn. The purpose of attribute selection is to help the data mining algorithm focus solely on information relevant to the current knowledge discovery task. Alternatively, instance selection is an emerging field of study that improves a training dataset by selecting a subset of good instances from which to learn. It is hoped that the selected instances exhibit the underlying concepts of the data and make the desired knowledge easier to discover by the data mining algorithm [2]. Instance selection and its ability to select good training datasets, and consequently, its ability to improve the performance of classification data mining algorithms warrants further investigation and is the focus of this study.

The process of instance selection was first utilized for instance based classifiers, such as k-Nearest Neighbors (k-NN), because faster classifications could be obtained by maintaining only certain necessary instances in the classifier's dataset[3, 4, 5]. However, instance selection not only reduced the size of the dataset, but it also improved the dataset quality by not selecting to maintain outlying, noisy, contradictory or simply redundant instances. Therefore, instance selection should be beneficial to other data mining algorithms because sets of training data often contain instances that are not helpful in the knowledge discovery process, and it is believed that good data mining results can be achieved by learning from subsets of the training data that do not contain these bad instances [1,6].

In addition to the early uses of instances selection, bagging and boosting are two widely accepted methods also capable of improving the accuracy of a classifier through the manipulation of the instances of a training dataset. Both of these meta-learners sample instances with replacement from a training dataset and create a collection of new training datasets and classifiers. Both meta-learners then use a system of classifier voting to determine how a test instance should be classified. The difference between bagging and boosting is that bagging assigns a weight to increase or decrease the probability of an instance being sampled based on the classification results of the previous model [7]. Unlike the classifier models created by instance selection, bagging and boosting inherently lead to models that are more complex than the underlying classifier. However, these methods do set a precedent for the use of instance manipulation to create training datasets.

As a consequence of instance selection's use in instance based classifiers and the acceptable use of instance manipulation in bagging and boosting, a popular use of instance selection outside of instance based classifiers is to select good training data for model dependent classifiers. Instance selection methods of this variety are discussed in a thorough review by Nicolas Garcia-Pedrajas [8], and it is noted that the majority of these methods find an acceptable set of training instances by utilizing random search with evolutionary algorithms. Evolutionary algorithms are a popular search technique because they are capable of taking into consideration the particular bias of a specific learning algorithm, and as such, lead to a good selection of training instances for that classifier. Most instance selection methods for training set selection are designed for neural networks and decision trees, but could be adapted for a variety of different classification algorithms.

Training set instance selection procedures for neural networks serve to increase the generalization ability of the networks by selecting good instances on which to train. These procedures should be beneficial to neural networks because in the presence of noisy data, the networks become overfitted to the training data and subsequently are poor classifiers to unseen instances [9]. Reeves et. al [10] and Reeves et. al [11] perform instance selection for radial basis function (RBF) nets with evolutionary algorithms where the fitness of a solution is determined by the accuracy of the RBF net, and the instances contained in the training dataset are evolved. It is found that indeed the generalization performance of RBF nets can be increased through training set instance selection. Kim et. al [9] also performs instance selection for increased generalization ability, but this work is focused on feed forward neural networks and financial forecasts. Here, the training instances and the structure of the network are determined by an evolutionary algorithm, and the fitness of a solution is evaluated by the accuracy of the network. Again, the results indicate that the generalization performance of neural networks can be improved by selecting instances to create a higher quality training dataset.

The decision tree classifier is another popular choice for training set instance selection because unhelpful instances in the decision tree's training data cause the tree to unnecessarily grow its structure [1,12]. This overfitting problem defeats the purpose of the decision tree by hiding or confusing the discovered knowledge in a large and un-interpretable tree structure that has poor generalization abilities. Performing instance selection often results in a smaller and more interpretable tree, an indication that the unhelpful instances have not been selected. Endou and Zhao [13], Cano et al. [14], and Cano et al. [15], evolve the instances contained in a subset of the training data in the hopes of finding a collection of instances that

adequately describe the full dataset. Endou and Zhao [13], as well as Cano et al. [15], evaluate the fitness of the selected training data through the construction and evaluation of a decision tree, and both of these methods are successful in reducing decision tree size, while still maintaining acceptable, if not improved, levels of accuracy. Cano et al. [14], evaluates the fitness of the training data using the 1-NN classifier, and interestingly, this method is also successful in creating simple, yet accurate, decision trees. These methods indicate that decision tree learning can indeed be improved by selecting a good set of training instances through instance selection.

Despite the existence of instance selection methods for selecting good sets of training data, there is still no convenient formulation of the root problem. Finding the optimal subset of instances from which to learn a classifier requires checking the performance of every subset of the original instances. Past methods have looked for this best subset of instances, but their search space has 2^n possible solutions for an original training set of size n, prohibitively large for anything but random search. In this work we propose a new formulation of the instance selection problem that employs ideas from the set covering problem, with the intent of improving the quality of the space to be searched.

The set covering problem is a constrained binary integer program where the decision to include a particular variable, or to make the value of a variable one, satisfies one or more constraints and is associated with some cost. The objective of this problem is to make a minimal cost selection of the variables so that all of the constraints are satisfied. The set covering problem has been used for practical purposes such as workforce scheduling, where the constraints are times of the day that require workers, the variables are feasible schedules of the workers, and the objective is to select schedules that result in the cheapest coverage of the required time slots. In cases where there are too many feasible schedules for the problem to be computationally practical, the number of variables can be reduced by constructing a collection of good schedules, and then a good, but perhaps sub-optimal, selection can be made. This technique is known as column generation [16].

In the context of instance selection, the set covering problem would have variables that coincide with subsets of instances able to represent, or cover, portions of the original training data, and constraints requiring all of the original training instances be covered by the selected subsets. However, considering all of the possible subsets of the training data in the set covering formulation would be no more practical than searching through all of the subsets to find the optimal selection of instances. Therefore, column generation should be used to simplify the problem by constructing a good but reduced collection of instance subsets. An objective function for this formulation could be one of several different cost minimization functions related to the coverage of the candidate subsets, but deriving an effective function to represent the complex nature of evaluating the performance of a training dataset would be very difficult if not impossible. Despite this obvious shortcoming of the set covering formulation of the instance selection problem, it is proposed that the intelligently constructed candidate subsets can still be beneficial as high quality decision variables in a solution search space.

The remainder of the work is organized as follows. In Section 2, we will develop efficient methods for generating instance candidate subsets. In Section 3, we will present experimental results for sixteen datasets concerning the effectiveness of the construction methods and recommendations based on those results. Then, in Section 4 we develop a search technique for selecting sets of training data from the instance candidate subsets. Again, experimental results and recommendations will be presented. Finally, in Section 5, overall results and implications will be examined along with thoughts for future areas of research.

2. Column generation as candidate subset construction

As stated in Section 1, column generation can be used to construct high quality subsets of instances that could be used as decision variables in a solution search space for an instance selection procedure. In the

set covering problem, the purpose of column generation is to create variables that satisfy a large number of the problem's constraints, and in the context of instance selection, the purpose would be to construct instance subsets able to represent a large number of the training instances. This motivation suggests a need for a candidate subset construction technique to build subsets of training instances that alone lead to classifiers with high predictive accuracy. The remainder of this section will introduce possible construction techniques, and special considerations for creating a solution search space of high quality instance subsets.

Instance candidate subsets can be constructed in incremental or decremental greedy fashions that include or exclude instances in candidate subsets based on the contribution of those instances to the predictive accuracy of some classifier. In the case of the greedy addition of instances to candidate subsets, candidate subsets originally contain no instances, and an instance is included in the subset if doing so improves the predictive accuracy of a classifier built from that subset. Alternatively, in the case of the greedy subtraction of instances from candidate subsets, candidate subsets initially contain all of the original training instances, and an instance is removed if doing so does not greatly hinder the predictive accuracy of a classifier built from that subset. These two approaches create candidate subsets that are composed of instances that together work good for classification as judged by some classifier. However, it must be kept in mind that we desire to create a search space with a variety of these good candidate subsets and that we do not necessarily want to deprive any particular instance an opportunity to belong to the final set of training data. A single candidate subset is therefore created for each instance of the original training data with that instance automatically included in the subset, and the remainder of the original training instances considered for inclusion or exclusion to the subset in a random order.

The greedy addition or subtraction of instances to a candidate subset requires that the predictive accuracy of a classifier built from the candidate subset be measured both with and without the instances under consideration. To calculate this accuracy, a classifier is built from the subset's instances and the classifier is used to predict the class values of the original training data. The known class values from the original training data are then used to calculate the predictive accuracy of the classifier. Of course, this measurement needs to be calculated a great number of times throughout the construction of the candidate subsets, and as such, should be evaluated by a simple classifier that allows for quick calculations. Two simple classifiers that serve this purpose are the naïve Bayes and decision stump classifiers [7].

Given the two methods for including instances in subsets, and the two classifiers for calculating predictive accuracy, there are four methods for constructing instance candidate subsets. However, only three methods of constructing candidate subsets will be discussed, because only the naïve Bayes classifier has an apparent issue from the greedy addition of instances, as observed in preliminary experiments. That is, the naïve Bayes classifier considers all of the data's attributes when making a prediction and a large number of instances may make a minute positive contribution to the classifier's data by improving the probability estimation of one attribute or another. This practice can result in subsets with a large number of instances when fewer, but higher quality instances can achieve the same result. This problem was not observed when using the decision stump classifier to include instances, perhaps because the decision stump classifier only considers one attribute when making a prediction and fewer instances make a significant impact on a single attribute. The following two figures outline the possible construction methods for creating instance candidate subsets where the calculation of predictive accuracy is calculated with either the naïve Bayes or decision stump classifiers as earlier described.

Forward Selection (greedy addition of instances)

Step 0:

Given a training set $T = \{x_1, x_2, \ldots, x_n\}$ *and* $n = |T|$.

Step 1:

For $i = 1$ *to* n

$S^{(i)} = \{x_i\}$

$U^{(i)} = \left\{u_1^{(i)}, u_2^{(i)}, \ldots, u_{n-1}^{(i)}\right\}$, *where* u_j^i *is the* j^{th} *integer of a randomization of the integers from 1 to n, less the integer i.*

Step 2:

For $i = 1$ *to* n

For $j = 1$ *to* $n - 1$,

If Predictive Accuracy $\left(S^{(i)} \cup \left\{x_{u_j^i}\right\}\right) >$ *Predictive Accuracy* $\left(S^{(i)}\right)$,

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

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

Figure 1. A description of the forward selection technique where predictive accuracy is calculated by building a decision stump or naïve Bayes classifier from the provided instances and predicting the original training instances.

Backward Selection (greedy subtraction of instances)

Step 0:

Given a training set $T = \{x_1, x_2, \ldots, x_n\}$ *with* $n = |T|$

and an A, *such that* A *is an acceptable loss of percentage points.*

Step 1:

For $i = 1$ *to* n

$S^{(i)} = \{x_1, x_2 \ldots x_n\}$

$U^{(i)} = \left\{u_1^{(i)}, u_2^{(i)}, \ldots, u_{n-1}^{(i)}\right\}$, *where* u_j^i *is the* j^{th} *integer of a randomization of the integers from 1 to n, less the integer* i.

Step 2:

$S^i = \{x_1, x_2 \ldots x_n\}$

For $i = 1$ *to* n

For $j = 1$ *to* $n - 1$

If $Predictive\ Accuracy\left(S^{(i)} / \left\{x_{u_j^i}\right\}\right) \geq Predictive\ Accuracy(S) - A$

Then $S^{(i)} = S^{(i)} / \left\{x_{u_j^i}\right\}$

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

Figure 2. A description of the backward selection technique where predictive accuracy is calculated by building a decision stump or naïve Bayes classifier from the provided instances and predicting the original instances.

The next section will present results regarding the quality of constructed instance candidate subsets from sixteen experimental datasets from the UCI machine learning data repository. Then, Section 4 will show that these constructed candidate subsets can be utilized in a random search method with an objective to maximize the generalization ability of classifiers built from the selected training instances.

3. Subset quality with experimental results

The goal of constructing instance candidate subsets is to create subsets of instances that lead to classifiers with high levels of predictive accuracy that can be used as decision variables in an improved search space in an instance selection procedure. This section will assess the effectiveness of the suggested methods in obtaining this goal by constructing and evaluating the quality of instance candidate subsets for sixteen experimental datasets from the UCI machine learning data repository. First, an experimental approach and evaluation procedure will be discussed, then results will be presented, and finally recommendations will be made.

Dataset Name	Num Instances	Num Attributes	Num Classes	Decision Stump Accuracy	Naïve Bayes Accuracy
Abalone*	474	8	2	84.45%	84.03%
Abalone*^{T}	696	9	2	68.68%	55.75%
Balance	416	4	3	63.16%	90.43%
Breast Cancer (Wisconsin)	466	9	2	91.85%	94.42%
Chess (King-Rook vs. King-Pawn)	473	36	2	66.24%	83.97%
Congressional Voting Records	290	16	2	95.17%	91.03%
Credit Approval	460	15	2	85.65%	78.26%
Ecoli	224	7	8	57.89%	83.04%
Ionosphere	234	34	2	82.05%	82.91%
Mamogram Mass	640	4	2	76.01%	77.26%
Pima Diabetes	512	8	2	72.66%	72.66%
Statlog (German Credit Data)*	666	20	2	70.06%	72.75%
Statlog (Landsat Satellite)	296	36	6	44.59%	79.73%
Statlog (Landsat Satellite)*^{T}	102	36	2	88.46%	84.62%
Thyroid Disease*^{T}	622	28	2	99.04%	94.55%
Tic-Tac-Toe Endgame	638	9	2	69.06%	70.31%

Table 1. A description of the experimental datasets. All have been partitioned into training and test datasets. The number of instances describes the size of the training dataset. The decision stump and naïve Bayes classifiers are built from all of the training data and their predictive accuracy is calculated on the withheld test datasets. * indicates that the dataset has been reduced from its original size. ^{T} indicates a reduction in the number of classes of the dataset to help favor the decision stump classifier.

Table 1 presents a variety of small to medium size datasets that have been chosen for the experimental study. The selected datasets exhibit a varying number of class values, attributes, and attribute types, as well as differing performances for the naïve Bayes and decision stump classifiers. Each dataset has been split into a two thirds training dataset and a one third test dataset, with the test dataset being reserved for evaluations of non intermediate classifiers. Five of the datasets have been further reduced from their original size for computational considerations, and three have had the frequency of different class values manipulated to give an advantage to the decision stump classifier. These experimental datasets have been chosen to test the robustness of the different candidate subset construction techniques and allow for experimentation to help formulate recommendations concerning which construction technique to implement for which dataset type.

The experimental procedure begins with the construction of instance candidate subsets for the datasets of Table 1 as outlined in Figure 1 and Figure 2 of Section 2. Specifically, candidate subsets are constructed

with the decision stump classifier utilizing forward selection (DS), the naïve Bayes classifier utilizing forward selection (NB_F), and with the naïve Bayes classifier utilizing backward selection (NB_10). It should be noted that the backward selection procedure allows a loss of ten percentage points from the classifier's original training accuracy, a value that was experimentally determined to create accurate, yet reasonably sized candidate subsets. Next, subsets of randomly selected instances are created for each dataset to mimic each type of candidate subset. Meaning, for a particular dataset, collections of random subsets are created such that an individual random subset contains the average number of instances as the subset type it is mimicking, and such that the number of random subsets is the same as the number of subsets in the mimicked subset type. These random subsets are R_DS, R_NB_F, and R_NB_10. As a final consideration, to take into account the aspects of randomness from the different construction techniques, the six types of candidate subsets are constructed 30 times for each dataset.

The effectiveness of the different construction techniques in creating candidate subsets is evaluated by the quality of the different subset types. The quality of a group of subsets is measured as the average percent of correct predictions for classifiers built from the individual subsets when predicting the class values of the original training data,

$$quality = \frac{\sum_{i=1}^{n} Predictive\ Accuracy\left(S^{(i)}\right)}{n}.$$

For the measurement of quality, n is the number of subsets in the group, $S^{(i)}$ is the i^{th} subset of the group, and predictive accuracy is found from a single classifier type, either the C4.5 decision tree [17], naïve Bayes, or logistic regression classifier. These three classifiers were chosen for the measurement of predictive accuracy because they build fundamentally different models of the training dataset, and as a consequence, result in fundamentally different measures of quality. Candidate subsets that could function as an improved search space should contain the good instances of a dataset, and therefore, should have a higher quality measure than their associated random subsets regardless of the classifier used to measure quality.

For each of the three types of quality measure there are two groups of interesting comparisons. First, are the comparisons between the intelligently constructed candidate subsets and the randomly constructed subsets, DS vs. R_DS, NB_F vs. R_NB_F, and NB_10 vs. R_NB_10. Second, are the comparisons between the different methods of candidate subset construction, DS vs. NB_F, DS vs. NB_10, and NB_F vs. NB_10. The first group of comparisons is concerned with whether or not the intelligently constructed subsets have higher quality than the randomly selected instances, and the second group of comparisons is concerned with which construction technique is best. Table 2 and Table 3 summarize these comparisons for the 30 replications with two sided hypothesis tests for differences of mean performed at 95% confidence with Bonferoni's adjustment for multiple comparisons.

	Quality Measure								
	Decision Tree			Naïve Bayes			Logistic Regression		
Comparison (First vs. Second)	First Better	Second Better	Equal	First Better	Second Better	Not Different	First Better	Second Better	Not Different
DS vs. R_DS	14	2	0	9	6	1	11	4	1
NB_F vs. R_NB_F	7	7	2	16	0	0	10	5	1
NB_10 vs. R_NB_10	16	0	0	16	0	0	16	0	0

Table 2. The comparisons are between the subsets constructed with the decision stump classifier using forward selection and random subsets (DS vs. R_DS), between the subsets constructed with the naïve Bayes classifier using forward selection and random subsets (NB_F vs. R_NB_F), and between the subsets constructed with the naïve Bayes classifier using backward selection and random subsets (NB_10, R_NB_10). Each comparison is evaluated with subset quality measured by the decision tree, naïve Bayes, and logistic regression classifiers.

The comparisons summarized in Table 2 reveal that the intelligently constructed candidate subsets often have higher quality than their associated random subsets, meaning that helpful subsets have been constructed. However, not all of the candidate subsets can be considered helpful. For example, for all of the datasets, when quality is measured with the naïve Bayes classifier, the subsets constructed with the NB_F method have higher quality than the R_NB_F constructed subsets. This is an indication that instances good at constructing a naïve Bayes classifier have been aggregated in the NB_F candidate subsets. Yet, when the same NB_F and R_NB_F subsets have their quality measured with a decision tree classifier the random subsets have higher quality for seven out of the 16 datasets, an indication that some of the instances contained in the candidate subsets may only be good at creating a naïve Bayes classifier and are unhelpful otherwise. Unfortunately, the goal is to create candidate subsets that contain helpful instances regardless of the classifier used to measure quality, and to some extent, both the NB_F and DS constructed subsets fail in this regard. The NB_10 constructed subsets are seemingly more successful in selecting universally helpful instances.

To understand the discrepancy between the NB_10 and the other candidate subset construction techniques for the comparisons of the random subsets, the number of instances included in each candidate subset type needs to be discussed. In general, the DS method creates very small candidate subsets, the NB_F method, as mentioned in Section 2, creates unnecessarily large candidate subsets, and the NB_10 method creates candidate subsets somewhere in between the two sizes. First, the very small subsets of the DS method are good for the decision tree classifier because of the close relationship between the decision stump and the decision tree, but in some cases their small sizes most likely exclude instances that could be helpful to other classifiers. Next, the unnecessarily large subsets of the NB_F method include every instance that is even remotely helpful in creating a good naïve Bayes classifier, meaning that the randomly selected instances are competing against instances that could be contributing very little to the subset quality. Finally, for the NB_10 approach, instances are included in the subset only if they are necessary to maintain a sufficiently good naïve Bayes classifier, and these instances are likely helpful for other classifiers because they contain this necessary information. Therefore, the random subsets associated with the NB_10 method are competing against only the absolutely necessary instances to build a sufficiently good naïve Bayes classifier, and the subsets are compact in size so the random instances have fewer opportunities to usurp the usefulness of the included instances. Most likely for this reason, the NB_10 construction approach creates candidate subsets that are robust for the three measure of quality, an indication that the subsets are indeed collections of instances useful for classification and possibly useful decision variables for an improved search space.

Given that the NB_10 method is more consistent in its ability to create candidate subsets that outperform the randomly selected instances than is the NB_F method, the rest of the analysis is performed with the NB_F subsets removed from consideration. Therefore, Table 3 shows only the comparison between the DS constructed subsets and the NB_10 constructed subsets, as faceted by dataset type, where each dataset is classified into one of three groups. The classification scheme is arrived at by analyzing Table 1 and placing the datasets into three different categories based on which classifier has better predictive accuracy when built from the full set of training data. The three groups are datasets that are better for the naïve Bayes classifier, datasets that are better for the decision stump classifier, and datasets where the predictive accuracies of the two classifiers are very close in value.

	Quality Measure					
	Decision Tree		Naïve Bayes		Logistic Regression	
Best training set classifier	DS Better	NB_10 Better	DS Better	NB_10 Better	DS Better	NB_10 Better
Naïve Bayes	2	4	0	6	0	6
Decision Stump	4	1	2	3	3	2
Tie (+/- 2%)	2	3	0	5	0	5

Table 3. Summary of the comparison between the candidate subsets constructed with the decision stump classifier using forward selection and the candidate subsets constructed with the naïve Bayes classifier utilizing backward selection (DS vs. NB_10). The results are faceted by which classifier is best for the particular dataset and the comparison is performed for subset quality measured by the decision tree, naïve Bayes, and logistic regression classifiers.

Table 3 reveals that for most cases the NB_10 candidate subset construction method results in subsets with the best quality. However, exceptions can be seen for two situations. First, the DS construction method sometimes results in better quality subsets when the quality is measured by the decision tree, and as mentioned earlier, this is to be expected because of the close relationship between the decision tree and the decision stump classifiers. Second, when the decision stump classifier is a better classifier for the full training dataset, the DS method sometimes creates candidate subsets with higher quality, regardless of quality measure. This observation makes intuitive sense because if the underlying structure of the dataset is easier to exploit with the decision stump classifier than with the naïve Bayes classifier then subsets built with the decision stump may be more likely to find the instances that describe this structure, and therefore have better quality. This observation is important because the time required to construct subsets with the DS method is less than with the NB_10 method, meaning, for the right dataset, time can be saved when constructing an improved search space.

The observations from Table 2 and Table 3 lead to the following recommendations concerning how to best construct an improved search space for the instance selection problem.

Recommendation One:

Construct instance candidate subsets with backward selection as outlined in Figure 2 and with predictive accuracy measured by the naïve Bayes classifier for most dataset types.

Recommendation Two:

Construct instance candidate subsets with forward selection as outlined in Figure 1 with predictive accuracy measured by the decision stump classifier for datasets where the whole dataset is better classified by the decision stump classifier than by the naïve Bayes classifier and where the extra time associated with recommendation one is undesirable.

In this section we have shown that it is possible to construct instance candidate subsets whose instances are more helpful for creating classifiers than randomly selected instances. Also, we have made recommendations for constructing these candidate subsets with an intention of using the subsets as decision variables in a search space for the training set instance selection problem. This improved search space eliminates the need to consider the inclusion of instances to the final training dataset on an instance to instance basis. Section 4 will show that the constructed subsets can indeed be used as a solution search space by first selecting candidate subsets with a greedy selection procedure, then with an advanced Nested Partitions selection procedure, and finally by comparing these results to results from the original datasets.

4. Final Selection Procedures with Experimental Results

Having constructed instance candidate subsets to pose as decision variables, an improved search space is available for a variety of selection procedures. This section outlines two such procedures and compares experimental results of each to the accuracy of the original training dataset and to one another. Each selection procedure is performed for seven of the sixteen experimental datasets with subsets constructed using the NB_10 method, and with results judged by the C4.5 decision tree, naïve Bayes, and logistic regression classifiers. These three classifiers are again chosen because of the fundamentally different models they construct.

The first selection procedure used to search for an acceptable collection of candidate subsets is a greedy selection procedure. This greedy selection begins with no selected candidate subsets and adds the most beneficial candidate subset to the selection until no immediate improvement of classier accuracy can be made. The most beneficial candidate subset is said to be a previously unselected subset that most increases the classifier's accuracy from the union of that subset's instances and the current selection's instances. All accuracies are calculated on the original training instances, and the final training dataset is constructed by the union of the instances of the selected candidate subsets.

The second selection procedure is a Nested Partitions procedure [18]. The Nested Partitions method is essentially an adaptive sampling method in which the set of possible solutions is partitioned and feasible solutions are randomly generated on those partitions, with the sampling being concentrated in subsets of the partitions that appear most promising. Nested Partitions consists of the following steps:

1. Partition the feasible region of solutions into sub-regions.
2. Evaluate the potential of each region based on randomly generated sample solutions.
3. Select the most promising region.
4. Recursively repeat the process for the most promising region
5. Allow backtracking to previous regions if the random sampling indicates a better solution could be found outside the current most promising region.

Meaning, the Nested Partitions procedure searches through the search space containing the instance candidate subsets for a good selection of training data. 30 replications of the Nested Partitions procedure were executed to help gauge the variability of these selections.

Table 4, Table 5, and Table 6 summarize results for seven experimental datasets utilizing three types of instance manipulation: no instance selection performed on the original dataset, instance selection performed on the original dataset using the greedy selection procedure, and instance selection performed on the original dataset using the Nested Partitions procedure. The classifier accuracy used in both selection procedures is evaluated with either the decision tree, naïve Bayes, or logistic regression classifiers, and the accuracies reported in the tables are found from the evaluation of the constructed models on the withheld test and training datasets using the same classifier. Due to the variability of the

nested partitions method, 30 replications of that selection procedure were performed for each dataset, allowing for 95% confidence intervals to be constructed around the mean accuracy. Table 7 describes the reduction in dataset size as a percent reduction from the original number of instances.

	Naïve Bayes Test Accuracy			Naïve Bayes Train Accuracy		
Dataset	Original	Greedy	NP (95% CI)	Original	Greedy	NP (95% CI)
Balance	90.4%	91.4%	[90.7%,90.3%]	90.9%	92.5%	[92.3%,92.1%]
Chess (King-Rook vs. King-Pawn)	84.0%	89.9%	[90.3%,89.7%]	89.2%	94.7%	[93.3%,93.0%]
Credit Approval	78.3%	84.8%	[83.5%,82.2%]	78.9%	88.0%	[83.9%,83.3%]
E-coli	83.0%	83.0%	[84.2%,83.5%]	90.6%	92.9%	[91.9%,91.7%]
Statlog (landsat Satellite)	79.7%	83.1%	[81.1%,79.9%]	76.7%	85.8%	[84.2%,83.5%]
Thyroid Disease	94.6%	96.8%	[96.6,95.6%]	94.1%	95.5%	[95.5%,94.9%]
Tic-Tac-Toe Endgame	70.3%	72.2%	[73.2%,72.2%]	73.7%	77.6%	[75.9%,75.6%]

Table 4. Summary of results for the decision tree classifier. The greedy selection procedure's test accuracy and the Nested Partitions selection procedure's test accuracy are as good if not better than the original dataset's test accuracy, for all of the datasets. It should be observed that the models are not overfitted to the test dataset as indicated by the reduction in accuracy from the training dataset to the test dataset.

	Decision Tree Test Accuracy			Decision Tree Train Accuracy		
Dataset	Original	Greedy	NP (95% CI)	Original	Greedy	NP (95% CI)
Balance	81.3%	80.9%	[81.2%,80.0%]	90.4%	89.7%	[86.0%,85.7%]
Chess (King-Rook vs. King-Pawn)	97.5%	94.5%	[97.9%,97.8%]	97.7%	94.9%	[97.7%,97.6%]
Credit Approval	85.2%	85.7%	[85.1%,83.9%]	90.4%	86.7%	[89.3%,88.8%]
E-coli	82.1%	81.3%	[81.3%,79.8%]	90.2%	91.5%	[91.5%,91.0%]
Statlog (landsat Satellite)	73.6%	71.6%	[75.1%,72.7%]	95.3%	92.9%	[87.3%,86.2%]
Thyroid Disease	100.0%	92.0%	[99.0%,98.4%]	99.0%	92.1%	[97.5%,97.2%]
Tic-Tac-Toe Endgame	81.3%	70.6%	[78.4%,77.1%]	89.7%	75.7%	[84.8%,83.4%]

Table 5. Summary of results for the decision tree classifier. The greedy selection procedure's test accuracy is only better than the original dataset's test accuracy for the Credit Approval dataset. The Nested Partitions selection procedure's test accuracy is significantly better than the original dataset's test accuracy only for the Chess (King-Rook vs. King-Pawn) dataset.

	Logistic Regression Test Accuracy			Logistic Regression Training Accuracy		
Dataset	Original	Greedy	NP (95% CI)	Original	Greedy	NP (95% CI)
Balance	90.4%	90.4%	[91.0%,90.2%]	90.1%	93.8%	[92.4%,92.2%]
Chess (King-Rook vs. King-Pawn)	96.2%	93.2%	[92.9%,91.2%]	97.9%	95.3%	[96.4%,95.9%]
Credit Approval	82.2%	87.0%	[82.9%,81.5%]	90.4%	86.3%	[88.4%,87.4%]
E-coli	83.9%	75.0%	[82.4%,80.7%]	90.2%	93.3%	[91.8%,90.9%]
Statlog (landsat Satellite)	65.5%	80.4%	[70.9%,68.1%]	100.0%	82.4%	[86.2%,85.3%]
Thyroid Disease	96.5%	93.9%	[97.9%,97.5%]	96.3%	95.2%	[96.5%,96.3%]
Tic-Tac-Toe Endgame	97.5%	98.1%	[98.2%,97.8%]	98.4%	98.9%	[98.7%,98.6%]

Table 6. Summary of results for the logistic regression classifier. The test accuracy of the greedy procedure's selection is much better than the original dataset's accuracy for the Credit Approval dataset and for the Statlog (landsat Satellite) dataset. The Nested Partitions selection procedure's test accuracy is significantly better than the original dataset's test accuracy for the Statlog (landsat Satellite), the Thyroid Disease, and the Tic-Tac-Toe Endgame datasets.

		Percent Reduction From Original					
		Naïve Bayes		Decision Tree		Logistic Regression	
Dataset	Original	Greedy	NP (Mean)	Greedy	NP (Mean)	Greedy	NP (Mean)
Balance	416	86%	66%	50%	53%	78%	75%
Chess (King-Rook vs. King-Pawn)	473	80%	60%	85%	66%	82%	59%
Credit Approval	460	91%	91%	95%	66%	98%	62%
E-coli	224	61%	27%	57%	38%	48%	40%
Statlog (landsat Satellite)	296	82%	78%	48%	56%	89%	56%
Thyroid Disease	622	98%	95%	99.8%	95%	99%	97%
Tic-Tac-Toe Endgame	638	93%	88%	95%	47%	92%	85%

Table 7. Summary of the size of the selected training dataset, represented as the percent reduced from the original dataset. In most cases the greedy selection procedure results in fewer instances being selected than for the Nested Partitions selection procedure, and both reduce the number of instances from the original dataset by a least 27%.

4.1. Comparing selection algorithms

The greedy selection procedure and Nested Partitions selection procedure yield mixed results in comparison to one another. The objective of both procedures is to obtain selections of instances that lead to classifiers with the highest possible accuracy for the training dataset. This objective is utilized because of the belief that high training dataset accuracy will equate to high test dataset accuracy, that is, if the problem of overfitting can be avoided. However, it is believed that these selection procedures do indeed combat the problem of overfitting because they typically select only a small subset of the original instances to belong to the final training dataset, as exhibited in Table 7. Despite the selection procedures' shared objective function, Table 4, Table 5, and Table 6, indicate that sometimes the greedy selection procedure obtains better training dataset accuracy than the more sophisticated Nested Partitions selection procedure, most notably in Table 4 for the Credit Approval dataset. This shortcoming could be addressed by incorporating the greedy selection procedure into the Nested Partitions selection procedure, but the experimental results suggest that this may not be desirable.

Table 4, Table 5, and Table 6 show that for the discussed selection procedures, higher training dataset accuracy does not always equate to higher test dataset accuracy, as was predicted. For example, there are

cases where the Nested Partitions selection procedure has higher training dataset accuracy than the greedy selection procedure, but lower test dataset accuracy. Obviously for these cases the Nested Partitions' selection leads to a classifier that has been overfitted to the training dataset. This can be seen for the logistic regression classifier with the Chess (King-Rook vs. King-Pawn), Credit Approval, and Statlog (landsat Satellite) datasets, and for the decision tree classifier with the Credit Approval dataset. These results indicate that overfitting to the original training dataset can be an issue for classifiers built from reduced datasets, and that the correct procedure of selecting instances may not always be to try and obtain the highest possible training set accuracy.

4.2. The effect of instance selection for different classifiers and datasets

For the selected experimental datasets, both instance selection procedures always improve upon or match the testing dataset accuracy of the original naïve Bayes classifier. Instance selection rarely leads to an increase or large decrease in decision tree accuracy for any of the experimental datasets, and the resulting decision trees are in general smaller than the original decision trees. The results of instance selection for the logistic regression classifier are mixed. For some datasets, instance selection greatly improves the accuracy of the original logistic regression classifier, but for other datasets, instance selection causes an unacceptable loss of accuracy. Instance selection also reduces the size of the logistic regression model for three datasets, while for the other datasets it has no affect on model size. These observations will be discussed in detail in the subsections that follow.

4.2.1. Analyzing poor results

When compared to the original logistic regression classifier, the Chess (King-Rook vs. King-Pawn) and the E-coli datasets are greatly hindered by instance selection. The E-coli dataset is not improved because the training dataset most likely does not contain the necessary information to successfully classify the testing dataset. In fact, when using Ada Boost and Bagging**,** methods that strive to improve classifier performance by re-sampling instances, no accuracy higher than the original is achieved. This can be seen in Table 12. For the Chess (King-Rook vs. King-Pawn) dataset and the logistic regression classifier, the intelligent removal of instances also does not seem beneficial. The logistic regression classifiers built from these reduced datasets have a difficult time distinguishing between the two different classes, meaning the selection procedure removed instances that would have allowed a better separation to be found. Which instances these are is not clear. However, instance selection does not perform so poorly on the remainder of the datasets, regardless of the selected classifier.

4.2.2. Analyzing good results

Improvements in classifier accuracy from instance selection can be seen for all of the datasets with the naïve Bayes classifier. For the decision tree classifier only the Credit Approval and Chess (King-Rook vs. King-Pawn) datasets have their classifier accuracy improved by instance selection, and for the logistic regression classifier, the Credit Approval, Statlog (landsat Satellite), Thyroid Disease, and Tic-Tac-Toe Endgame datasets benefit. Particularly good improvements in classifier accuracy can be seen for the naïve Bayes and logistic regression classifier for the Credit Approval and Statlog (landsat Satellite) datasets.

Specifically, the greedy instance selection procedure performed for the Credit Approval dataset increases the accuracy of the original naïve Bayes classifier by nearly seven percentage points. The most notable change in the composition of the instances is that the selected instances do not exhibit the extreme values of the original dataset. An example of this reduced range can be seen for variables A14 and A15 in Figure 3. In the original dataset, variable A14 has a maximum value of 2,000 and the selected instances have a maximum value of 360. In the original dataset, variable A15 has a maximum value of 51,100 and the selected instances have a maximum value of only 5,000. Of course, other variables also exhibit a

reduced range, but not to this large of an extent. Eliminating instances with extreme values from the dataset has most likely allowed the classifier to focus on instances that are not harmful or superfluous to the classification task. The benefit of fewer instances with extreme values will be explored further in Section 4.2.3 for the easily pictured decision tree classifier**.**

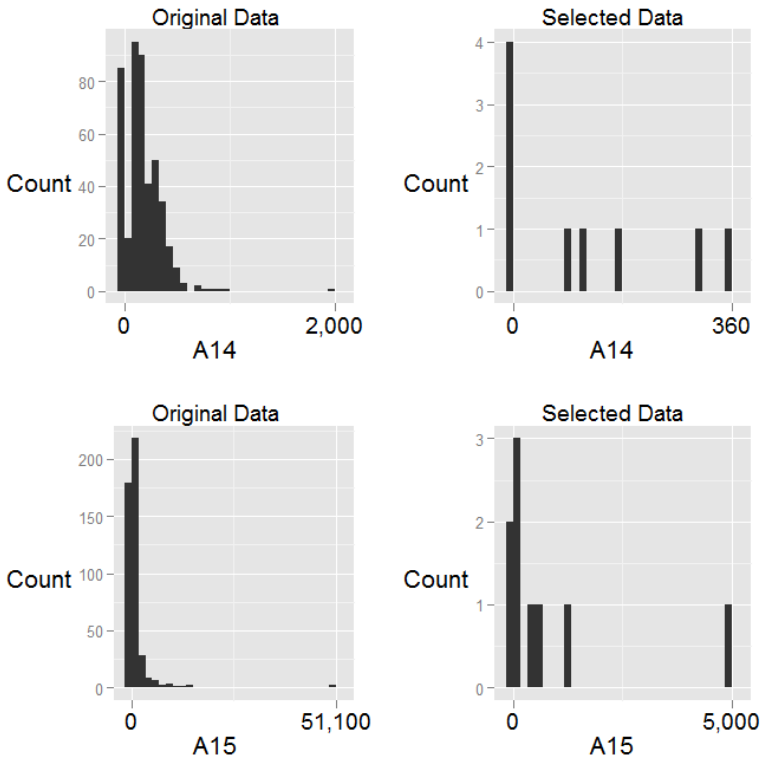
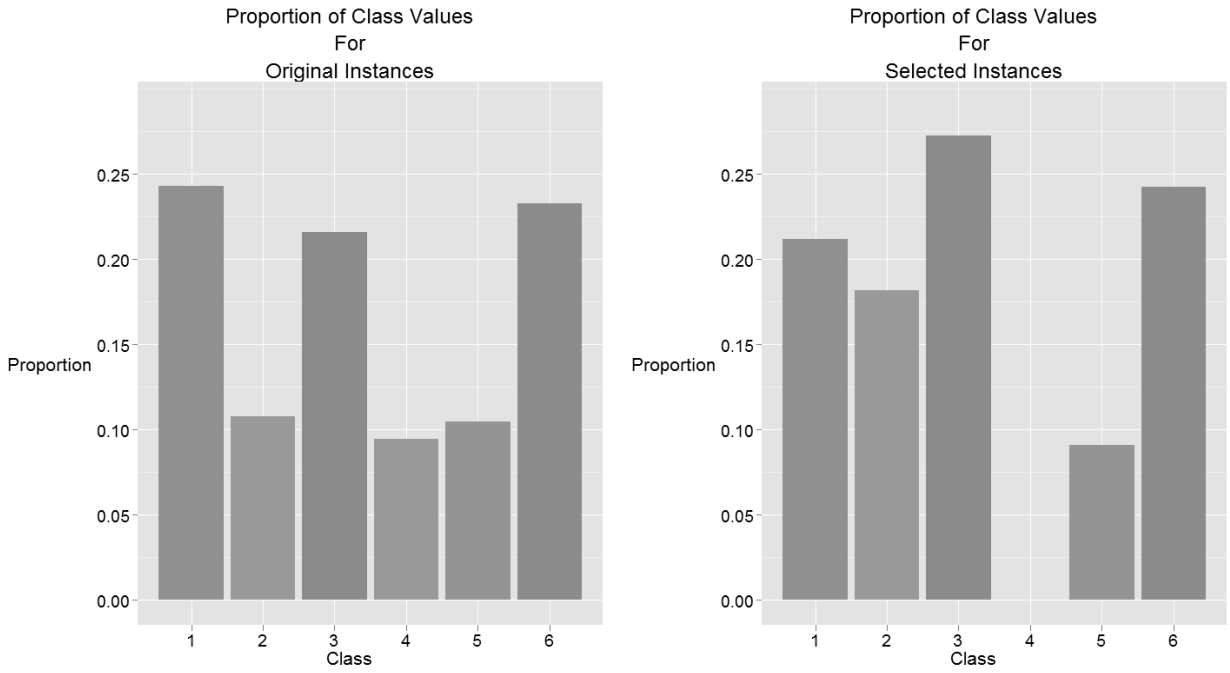


Figure 3. The selected instances for the Credit Approval dataset do not exhibit the extreme values seen in the original dataset for the variables A14 and A15, among others.

The improved accuracy of the logistic regression and naïve Bayes classifiers after instance selection on the Statlog (landsat Satellite) dataset most likely stems from the manipulation of instances that have class values that are frequently misclassified. For example, Class 3 of this dataset is frequently misclassified by the original logistic regression classifier, and is somewhat underrepresented in the original dataset. Yet, after instance selection, the selected instances have a higher proportion of Class 3. The classifier then does a better job classifying Class 3 in the final test dataset, likely because of this rescaling of the class proportions through the removal of redundant instances. An example of this for a particular selection of instances for the Statlog (landsat Satellite) dataset and the logistic regression classifier can be seen in Figure 4.



	Number of Test Instances Misclassified						
Classifier Built From	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Total
Original Instances	5	9	11	9	8	9	51
Selected Instances	1	3	3	14	5	2	28

Figure 4. The above bar charts show the proportion of the different class values for the original and greedily selected instances. The table shows the misclassifications made by a logistic regression classifier built from each selection. It can be seen that increasing the proportion of Class 2 and Class 3 reduces the number of misclassifications made for those classes in the test dataset.

Finally, perhaps the best experimental results of this study are that both instance selection procedures improve the accuracy of the original naïve Bayes classifier for each of the seven experimental datasets. However, this may not be a coincidence because the decision variables for these methods were built with the NB_10 construction procedure. As it may be remembered from Section 3, these subsets were designed to contain instances necessary for building a good naïve Bayes classifier, and these high quality decision variables have most likely led to the discovery of good instances for this classifier. This observation motivates the need to further investigate methods for constructing good candidate subsets, which could lead to more dependable ways of improving a classifier's accuracy.

4.2.3. Analyzing results concerning model reduction

Instance selection does not improve the testing accuracy for many of the decision trees built from the seven experimental datasets, but instance selection does reduce the size of most of these decision trees without greatly sacrificing accuracy. A similar observation can be made for three of the experimental datasets and the logistic regression classifier. For the Credit Approval and Statlog (landsat Satellite) datasets using the decision tree classifier, and for the Credit Approval and Thyroid Disease datasets using the logistic regression classifier, accuracy actually improves with the reduction in model size. For both

classifiers, model size can be interpreted as model complexity, and therefore smaller models are desired for their increased interpretability. Decision tree size is measured as the number of branches and leaf nodes of the tree, and the size of the logistic regression model is measured as the number of non-zero coefficients for the attributes of the training dataset. A summary of the observed model reductions can be seen in Table 8 and Table 9. The instance selection procedures have allowed simpler models to be discovered for both the decision tree and logistic regression classifiers by removing some of the datasets' unnecessary and unhelpful instances.

	Decision Tree Size		
Dataset	Original	Greedy	NP (95% CI)
Balance	63	61	[43.2,39.2]
Chess (King-Rook vs. King-Pawn)	23	13	[18.9,18.0]
Credit Approval	37	7	[22.2,17.0]
E-coli	17	21	[23.3,21.1]
Statlog (landsat Satellite)	37	33	[27.7,24.5]
Thyroid Disease	11	1	[3.9,3.2]
Tic-Tac-Toe Endgame	88	19	[55.8,48.4]

Table 8. In general decision tree size is reduced from the original decision tree. Tree size is measured as the number of branches and leaf nodes of the tree.

	Logistic Regression Model Size		
Dataset	Original	Greedy	NP (95% CI)
Chess (King-Rook vs. King-Pawn)	39	39	[38.2,37.5]
Credit Approval	42	19	[39.2,36.8]
Thyroid Disease	32	16	[20.6,17.7]

Table 9. The logistic regression models are smaller for these three datasets, and are more accurate for the Credit Approval dataset, as well as the Thyroid Disease dataset, for the Nested Partitions' selection. The logistic regression models for the other datasets were unaffected in size.

A specific example where instance selection helps reduce the size of a decision tree can be seen for the Credit Approval dataset. The decision tree built from the original training instances and the decision tree built from the greedily selected instances are shown in Figures 5 and 6 respectively. Both of these decision trees are identical through their first two branches, but soon diverge. The reason for the decision tree of the selected training dataset being much smaller than the original decision tree becomes apparent when analyzing the 84 of the 460 original training instances that still require separation after the first two branches have been constructed.

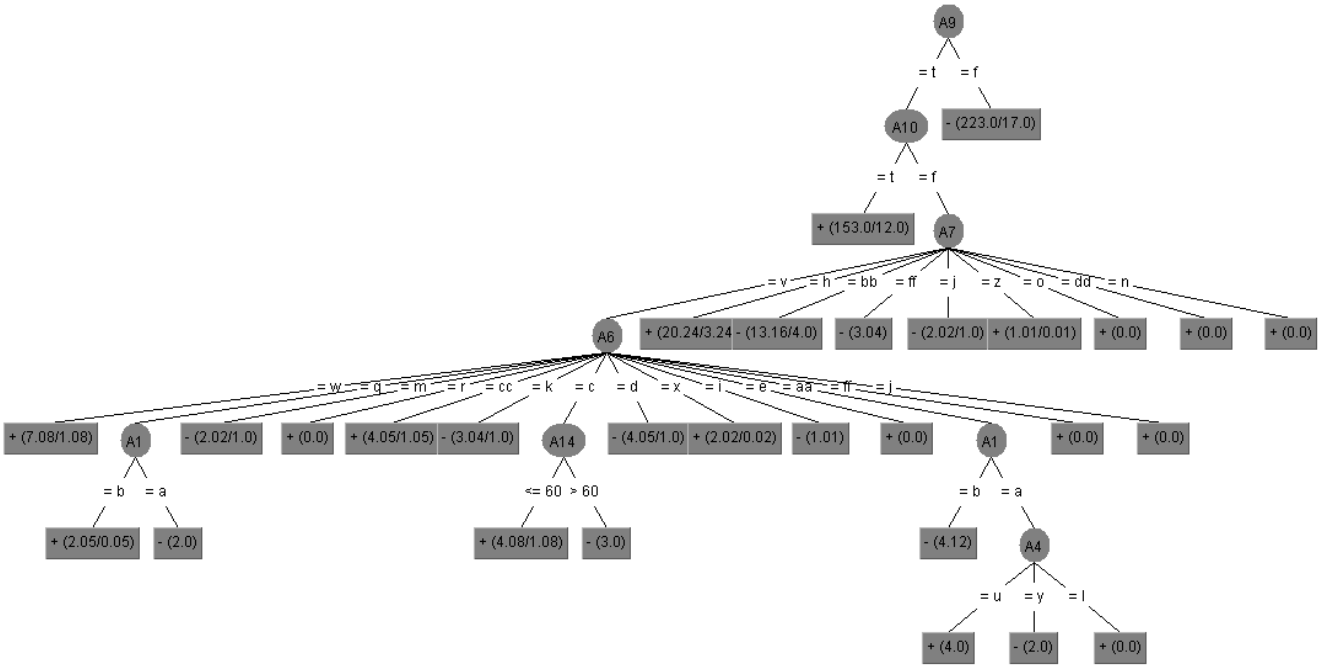


Figure 5. This decision tree is built from the original training instances. The first two branches classify 376 instances, and the remaining branches are used to classify just 84 instances.

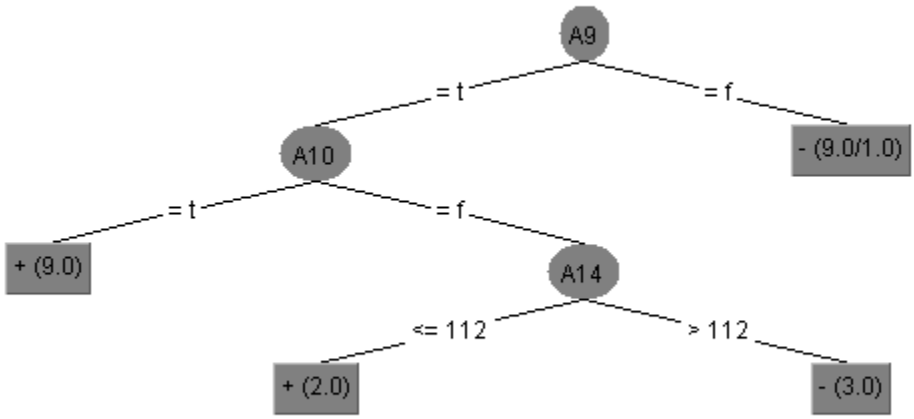


Figure 6. This decision tree is built from the greedy selected training instances. The first two branches are identical to the original decision tree, but the remainder of the tree is much simpler.

As seen in Figure 7**,** some of the original instances that still require classification after the first two branches of the tree have extreme values for the variable A14, and belong to both the positive and negative classes. A large number of branches are used to separate these remaining instances, and cause

the tree to become complex. The decision tree built from the greedily selected instances begins in the same fashion, but the instances remaining after the first two branches do not exhibit those extreme values for the variable A14. This smaller dataset can be fully classified utilizing only one additional branching. The resulting decision tree has worse training accuracy than the original decision tree, but better testing accuracy because it is no longer overfitted to the training data and the instances with extreme values for variable A14.

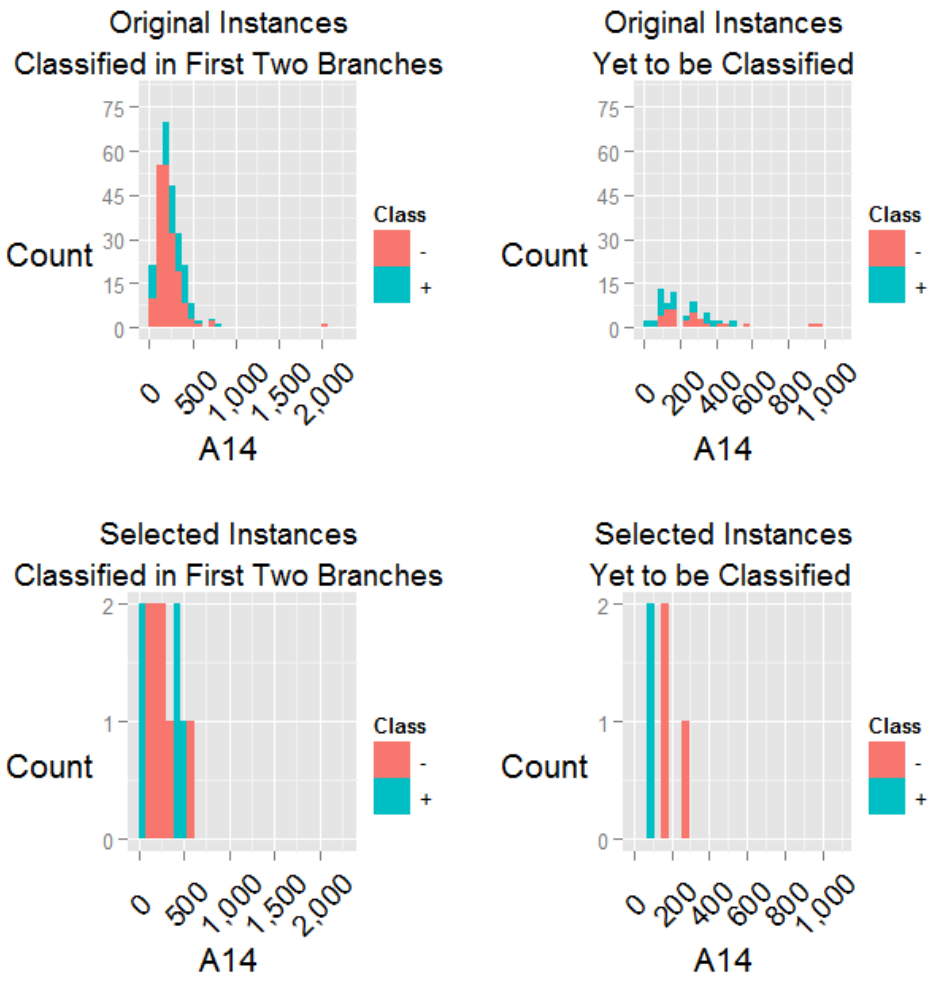


Figure 7. This figure shows the instances that have been classified from the first two branches of both the original and reduced decision trees in the left two plots. The right two plots show the instances that still require classification. It can be seen that the selected instances do not exhibit the extreme valued instances, and that a homogeneous split can easily be found.

4.3. Comparing instance selection to boosting and bagging

Tables 10, 11 and 12 compare the resulting accuracy of the two instance selection procedures to the accuracy of bagging and boosting with the naïve Bayes, decision tree, and logistic regression classifiers. For these experiments, boosting was performed with a modified Ada-Boost algorithm, drawing samples 100% the size of the original training dataset, and performing ten iterations of the algorithm [19].

Bagging as described in Breiman's "Bagging Predictors" was also performed for ten iterations, drawing samples 100% the size of the original training dataset [20]. Both the bagging and boosting procedures were repeated 30 times to allow 95% confidence intervals to be constructed.

	Naïve Bayes Test Accuracy				
Dataset	Original	Greedy	NP (95% CI)	Boosting (95% CI)	Bagging (95% CI)
Balance	90.4%	91.4%	[90.7%,90.3%]	[90.9%,90.9%]	[90.5%,90.3%]
Chess (King-Rook vs. King-Pawn)	84.0%	89.9%	[90.3%,89.7%]	[93.7%,93.7%]	[84.4%,83.6%]
Credit Approval	78.3%	84.8%	[83.5%,82.2%]	[81.3%,81.3%]	[78.5%,78.3%]
E-coli	83.0%	83.0%	[84.2%,83.5%]	[83.0%,83.0%]	[85.3%,84.6%]
Statlog (landsat Satellite)	79.7%	83.1%	[81.1%,79.9%]	[77.7%,77.7%]	[79.7%,79.2%]
Thyroid Disease	94.6%	96.8%	[96.6%,95.6%]	[94.6%,94.6%]	[94.4%,94.3%]
Tic-Tac-Toe Endgame	70.3%	72.2%	[73.2%,72.2%]	[84.7%,84.7%]	[70.4%,69.8%]

Table 10. Summary of the naïve Bayes test accuracy for the greedy and Nested Partitions selection procedures, as well as for boosting and bagging algorithms. The greedy selection procedure results in significantly better results than all of the other methods for the Balance, Credit Approval, Statlog (landsat Satellite), and Thyroid Disease datasets.

	Decision Tree Test Accuracy				
Dataset	Original	Greedy	NP (95% CI)	Boosting (95% CI)	Bagging (95% CI)
Balance	81.3%	80.9%	[81.2%,80.0%]	[79.4%,79.4%]	[83.8%,83.2%]
Chess (King-Rook vs. King-Pawn)	97.5%	94.5%	[97.9%,97.8%]	[96.2%,96.2%]	[97.6%,97.1%]
Credit Approval	85.2%	85.7%	[85.1%,83.9%]	[82.2%,82.2%]	[85.5%,84.8%]
E-coli	82.1%	81.3%	[81.3%,79.8%]	[82.1%,82.1%]	[81.8%,80.5%]
Statlog (landsat Satellite)	73.6%	71.6%	[75.1%,72.7%]	[82.4%,82.4%]	[78.6%,77.4%]
Thyroid Disease	100.0%	92.0%	[99.0%,98.4%]	[99.4%,99.4%]	[100%,99.8%]
Tic-Tac-Toe Endgame	81.3%	70.6%	[78.4%,77.1%]	[90.3%,90.3%]	[89.0%,88.0%]

Table 11. Summary of the decision tree test accuracy for the greedy and Nested Partitions selection procedures, as well as for the boosting and bagging algorithms. For the Chess (King-Rook vs. King-Pawn) and Credit Approval datasets, at least one of the instance selection procedures result in significantly better results than all of the other methods.

	Logistic Regression Test Accuracy				
Dataset	Original	Greedy	NP (95% CI)	Boosting (95% CI)	Bagging (95% CI)
Balance	90.4%	90.4%	[91.0%,90.2%]	[90.4%,90.4%]	[96.6%,90.1%]
Chess (King-Rook vs. King-Pawn)	96.2%	93.2%	[92.9%,91.2%]	[96.2%,96.2%]	[95.4%,95.0%]
Credit Approval	82.2%	87.0%	[82.9%,81.5%]	[82.2%,82.2%]	[81.8%,81.0%]
E-coli	83.9%	75.0%	[82.4%,80.7%]	[83.9%,83.9%]	[83.8%,83.0%]
Statlog (landsat Satellite)	65.5%	80.4%	[70.9%,68.1%]	[65.5%,65.5%]	[70.5%,68.9%]
Thyroid Disease	96.5%	93.9%	[97.9%,97.5%]	[96.5%,96.5%]	[97.0%,96.6%]
Tic-Tac-Toe Endgame	97.5%	98.1%	[98.2%,97.8%]	[97.2%,97.2%]	[97.5%,97.3%]

Table 12. Summary of the logistic regression test accuracy for the greedy and Nested partitions selection procedures, as well as for the boosting and bagging algorithms. At least one of the instance selection procedures result in significantly better results than all of the other methods for the Credit Approval, Statlog (landsat Satellite), Thyroid Disease, and Tic-Tac-Toe Endgame datasets.

These results show that a classifier built from intelligently selected instances is sometimes significantly more accurate than the more complex models created from bagging or boosting algorithms. For the naïve Bayes classifier, the Balance, Credit Approval, Statlog (landsat Satellite), and Thyroid Disease datasets are all benefited more by instance selection. Similarly, instance selection is more beneficial to the Credit Approval, Statlog (landsat Satellite), Thyroid Disease, and Tic-Tac-Toe Endgame datasets for the logistic regression classifier. Bagging and boosting does however outperform instance selection for the majority of the datasets with the decision tree classifier. From the above results it appears that instance selection is another viable method for improving classifier accuracy, and it should be kept in mind that the resulting models of instance selection are naturally simpler than the models of bagging and boosting.

4.4. Conclusions from experimental results

The experimental results of this section have shown that the constructed instance candidate subsets can be effectively used in instance selection procedures as decision variables. These results indicate that instance selection with instance candidate subsets may not benefit every dataset, but could potentially fix a variety of dataset issues. Furthermore, these results confirm that it is possible to improve the accuracy of a classifier by eliminating some of the redundant, harmful, and superfluous instances of a training dataset, as seen for the naïve Bayes and logistic regression classifiers on the Credit Approval and the Statlog (landsat Satellite) datasets. It can also be observed that instance selection can lead to simpler and more desirable models for the same reasons.

5. Concluding Remarks

Data preprocessing to improve the quality of the data utilized by classification algorithms is an important step in knowledge discovery that can be performed in part by instance selection procedures. However, current instance selection procedures make use of genetic algorithms to traverse prohibitively large search spaces. Therefore, using concepts from column generation and the set covering problem, instance candidate subsets were constructed with the intention of improving the quality of the solution search space of such procedures. First, Section 2 outlined a procedure for constructing instance candidate subsets. Then, Section 3 showed that the constructed subsets were better for classification than randomly

collected instances, and that the NB_10 approach was a dependable method for constructing these subsets. Finally, in Section 4, it was shown that the instance candidate subsets could be used as high quality decision variables in instance selection procedures.

The improvement of classification accuracy achieved for the majority of the seven experimental datasets showed that improved classification results can be achieved by intelligently selecting instance subsets for inclusion to a training dataset. Furthermore, it was shown that when the final training dataset leads to an improved classifier it could be said that some of the harmful and unhelpful instances of the dataset have been removed.

Future studies should consider possible scaling issues of the instance candidate subset construction procedure to large datasets, perhaps alleviating problems with stratification of the data. In addition to possible scaling approaches, the improved decision variables should be investigated for the possibility of reducing the size of the solution search space. Most importantly, more methods for constructing good instance candidate subsets should be pursued. Higher quality decision variables should lead to better selections of instances, as exhibited by the consistently good performance of instance selection with the naïve Bayes classifier and the NB_10 subsets.

References

[1] M. Sebban, R. Nock, J.H. Chauchat, R. Rakotomalala, Impact of learning set quality and size on decision tree performances, International Journal of Computers, Systems and Signals, 1(1) (2000) 85-105.

[2] S.B. Kotsiantis, D. Kanellopoulos, P. E. Pintelas, Data Preprocessing for Supervised Learning, International Journal of Computer Science, 1(1) (2006) 112-117.

[3] P. Hart, The Condensed Nearest Neighbor Rule, IEEE Transactions on Information Theory (Corresp.), IT-14 (1968) 515-516.

[4] G. L. Ritter, H. B. Woodruff, S. R. Lowry, T. L. Isenhour, An Algorithm for a Selective Nearest Neighbor Decision Rule, IEEE Transactions on Information Theory, 21(6) (1975) 665-669.

[5] D. Wilson, Asymptotic Properties of Nearest Neighbor Rules Using Edited Data, IEEE Transactions on Systems, Man, and Cybernetics, 2(3) (1972) 408-421.

[6] J. Olvera-Lopez, J. Carrasco-Ochoa, J. Martinez-Trinidad, J. Kittler, A review of instance selection methods, Artif Intell Rev, 34 (2010) 133–143.

[7] J. Han, M. Kamber, Data Mining Concepts and Techniques, Morgan Kaufmann, San Francisco, CA, 2006.

[8] N. Garcia-Pedrajas, Evolutionary computation for training set selection, WIREs Data Mining and Knowledge Discovery, 1 (2011) 512-523.

[9] Kim K-J, Artificial neural networks with evolutionary instance selection for financial forecasting, Expert Syst Appl, 30 (2006) 519-526.

[10] C. Reeves, S. Taylor, Selection of training sets for neural networks by a genetic algorithm, Parallel Problem Solving from Nature- PSSN V, (1998) 633-642.

[11] C. Reeves, D Bush, Using genetic algorithms for training data selection in RBF networks, in: Instance Selection and Construction for Data Mining, H. Liu and H. Motoda (Eds), Kluwer, Norwell, MA, pp.339–356.

[12] T. Oates, D. Jensen, The Effects of Training Set Size on Decision Tree Complexity, in proceedings of the Fourteenth International Conference on Machine Learning (1997).

[13] T. Endou, Q. Zhao, Generation of Comprehensible Decision Trees Through Evolution of Training Data, in proceedings of the 2002 Congress on Evolutionary Computation, (2002) 1221-1225.

[14] J. Cano, F. Herrera, M. Lozano, Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study, IEEE Transactions on Evolutionary Computation, 7(6) (2003) 561-575.

[15] J. Cano, F. Herrera, M. Lozano, Evolutionary Stratified Training Set Selection for Extracting Classification Rules with Trade off Precision-Interpretability, Data & Knowledge Engineering, 60 (2006) 90-108.

[16] W. Wilhelm, A Technical Review of Column Generation in Integer Programming, Optimization and Engineering, 2(2) (2001) 159-200.

[17] J. Quinlan, C4.5 Programs for Machine Learning, Morgan Kaufmann, San Francisco, CA, 1992.

[18] S. Leyuan, Ó. Sigurdur, Nested Partitions Method, Theory and Applications, International Series In Operations Research & Management Science, 109 (2009).

[19] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, Machine Learning: Proceedings of the Thirteenth International Conference, (1996) 148-156.

[20] L. Breiman, Bagging Predictors, Machine Learning, 24(2) (1996) 123-140.

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