

Benchmarking Attribute Selection Techniques for Discrete Class Data Mining

Mark A. Hall, Geoffrey Holmes

Abstract

Data engineering is generally considered to be a central issue in the development of data mining applications. The success of many learning schemes, in their attempts to construct models of data, hinges on the reliable identification of a small set of highly predictive attributes. The inclusion of irrelevant, redundant and noisy attributes in the model building process phase can result in poor predictive performance and increased computation.

Attribute selection generally involves a combination of search and attribute utility estimation plus evaluation with respect to specific learning schemes. This leads to a large number of possible permutations and has led to a situation where very few benchmark studies have been conducted.

This paper presents a benchmark comparison of several attribute selection methods for supervised classification. All the methods produce an attribute ranking, a useful device for isolating the individual merit of an attribute. Attribute selection is achieved by cross-validating the attribute rankings with respect to a classification learner to find the best attributes. Results are reported for a selection of standard data sets and two diverse learning schemes C4.5 and naive Bayes.

Keywords

Attribute Selection, Classification, Benchmarking.

I. INTRODUCTION

MANY factors affect the success of data mining algorithms on a given task. The quality of the data is one such factor—if information is irrelevant or redundant, or the data is noisy and unreliable, then knowledge discovery during training is more difficult. Attribute subset selection is the process of identifying and removing as much of the irrelevant and redundant information as possible. Learning algorithms differ in the amount of emphasis they place on attribute selection. At one extreme are algorithms such as the simple nearest neighbour learner, that classifies novel examples by retrieving the nearest stored training example, using all the available features in its distance computations. At the other extreme are algorithms that explicitly try to focus on relevant features and ignore irrelevant ones. Decision tree inducers are examples of this approach. By testing the values of certain attributes, decision tree algorithms attempt to divide training data into subsets containing a strong majority of one class. This necessitates the selection of a small number of highly predictive features in order to avoid over fitting the training data. Regardless of whether a learner attempts to select attributes itself or ignores the issue, attribute selection prior to learning can be beneficial. Reducing the dimensionality of the data reduces the size of the hypothesis space and allows algorithms to operate faster and more effectively. In some cases accuracy on future classification can be improved; in others, the result is a more compact, easily interpreted representation of the target concept.

Many attribute selection methods approach the task as a search problem, where each state in the search space specifies a distinct subset of the possible attributes [1]. Since the space is exponential in the number of attributes, this necessitates the use of a heuristic search procedure for all but trivial data sets. The search procedure is combined with an attribute utility estimator in order to evaluate the relative merit of alternative subsets of attributes. When the evaluation of the selected features with respect to learning algorithms is considered as well it leads to a large number of possible permutations.

©2003 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

The authors are with the Department of Computer Science, University of Waikato, Hamilton, New Zealand. E-mail {mhall, geoff}@cs.waikato.ac.nz

This fact, along with the computational cost of some attribute selection techniques, has led to a situation where very few benchmark studies on non-trivial data sets have been conducted.

Good surveys reviewing work in machine learning on feature selection can be found in [1], [2]. In particular, Liu et. al. [2] use small artificial data sets to explore the strengths and weaknesses of different attribute selection methods with respect to issues such as noise, different attribute types, multi-class data sets and computational complexity. This paper, on the other hand, provides an empirical comparison of six major attribute selection methods on fourteen well known benchmark data sets for classification. Performance on a further three “large” data sets (two containing several hundreds of features, and the third over a thousand features) is reported as well. In this paper we focus on attribute selection techniques that produce ranked lists of attributes. These methods are not only useful for improving the performance of learning algorithms; the rankings they produce can also provide the data miner with insight into their data by clearly demonstrating the relative merit of individual attributes. The next section describes the attribute selection techniques compared in the benchmark. Section 3 outlines the experimental methodology used and briefly describes the Weka Experiment Editor (a powerful Java based system that was used to run the benchmarking experiments). Sections 4 and 5 present the results. The last section summarises the findings.

II. ATTRIBUTE SELECTION TECHNIQUES

Attribute selection techniques can be categorised according to a number of criteria. One popular categorisation has coined the terms “filter” and “wrapper” to describe the nature of the metric used to evaluate the worth of attributes [3]. Wrappers evaluate attributes by using accuracy estimates provided by the actual target learning algorithm. Filters, on the other hand, use general characteristics of the data to evaluate attributes and operate independently of any learning algorithm. Another useful taxonomy can be drawn by dividing algorithms into those which evaluate (and hence rank) individual attributes and those which evaluate (and hence rank) *subsets* of attributes. The latter group can be differentiated further on the basis of the search technique commonly employed with each method to explore the space of attribute subsets¹. Some attribute selection techniques can handle regression problems, that is, when the class is a numeric rather than a discrete valued variable. This provides yet another dimension to categorise methods. Although some of the methods compared herein are capable of handling regression problems, this study has been restricted to discrete class data sets as all the methods are capable of handling this sort of problem.

By focusing on techniques that rank attributes we have simplified the matter by reducing the number of possible permutations. That is not to say that we have ignored those methods that evaluate subsets of attributes; on the contrary, it is possible to obtain ranked lists of attributes from these methods by using a simple hill climbing search and forcing it to continue to the far side of the search space. For example, forward selection hill climbing search starts with an empty set and evaluates each attribute individually to find the best single attribute. It then tries each of the remaining attributes in conjunction with the best to find the best pair of attributes. In the next iteration each of the remaining attributes are tried in conjunction with the best pair to find the best group of three attributes. This process continues until no single attribute addition improves the evaluation of the subset. By forcing the search to continue (even though the best attribute added at each step may actually decrease the evaluation of the subset as a whole) and by noting each attribute as it is added, a list of attributes ranked according to their incremental improvement to the subset is obtained. Figure 1 demonstrates this process graphically.

Several of the attribute selection techniques compared in the benchmark only operate on discrete valued features. In order to apply these techniques to data with numeric features discretisation is applied as a preprocessing step. We used the state-of-the-art supervised discretisation technique developed by

¹It is important to note that *any* search technique can be used with a method that evaluates attribute subsets and that many of the possible permutations that this leads to have yet to be explored

Features : {A B C D}				Best Single Addition (ranking)	
	Feature Set	Score			
Iteration 0	[]	0.00			
Iteration 1	[A]	0.20			
	[B]	0.40		B	
	[C]	0.30			
	[D]	0.15			
Iteration 2	[A B]	0.38			
	[B C]	0.65		C	
	[B D]	0.47			
Iteration 3	[A B C]	0.60		A	
	[B C D]	0.57			
Iteration 4	[A B C D]	0.62		D	

Fig. 1. Forward selection search modified to produce a ranked list of attributes. Normally the search would terminate after iteration 3 because no single attribute addition improves the best subset from iteration 2. In this case the search has been forced to continue until all attributes have been included.

Fayyad and Irani [4]. Essentially the technique combines an entropy based splitting criterion (such as that used by the C4.5 decision tree learner [5]) with a minimum description length stopping criterion. The best cut point is the one that makes the subintervals as pure as possible, i.e where the information value is smallest (this is the same as splitting where the information *gain*, defined as the difference between the information value without the split and that with the split, is largest). The method is then applied recursively to the two subintervals. For a set of instances S , a feature A and a cut point T , the class information entropy of the partition created by T is given by

$$E(A; T; S) = \frac{|S_1|}{|S|} \text{Ent}(S_1) + \frac{|S_2|}{|S|} \text{Ent}(S_2) \quad (1)$$

where S_1 and S_2 are two intervals of S bounded by cut point T , and $\text{Ent}(S)$ is the class entropy of a subset S given by

$$\text{Ent}(S) = \sum_{i=1}^C p(C_i, S) \log_2(p(C_i, S)). \quad (2)$$

The stopping criterion prescribes accepting a partition T if and only if the cost of encoding the partition and the classes of the instances in the intervals induced by T is less than the cost of encoding the classes of the instances before splitting. The partition created by T is accepted *iff*

$$\text{Gain}(A; T; S) > \frac{\log_2(N-1)}{N} + \frac{\log_2(3^c - 2) - [c\text{Ent}(S) - c_1\text{Ent}(S_1) - c_2\text{Ent}(S_2)]}{N}. \quad (3)$$

In Equation 3, N is the number of instances, c , c_1 , and c_2 are the number of distinct classes present in S , S_1 , and S_2 respectively. The first component is the information needed to specify the splitting point; the second is a correction due to the need to transmit which classes correspond to the upper and lower subintervals.

The rest of this section is devoted to a brief description of each of the attribute selection methods compared in the benchmark. There are three methods that evaluate individual attributes and produce a ranking unassisted, and a further three methods which evaluate subsets of attributes. The forward

selection search method described above is used with these last three methods to produce ranked lists of attributes. The methods cover major developments in attribute selection for machine learning over the last decade. We also include a classical statistical technique for dimensionality reduction.

A. Information Gain Attribute Ranking

This is one of the simplest (and fastest) attribute ranking methods and is often used in text categorisation applications [6], [7] where the sheer dimensionality of the data precludes more sophisticated attribute selection techniques. If A is an attribute and C is the class, Equations 4 and 5 give the entropy of the class before and after observing the attribute.

$$H(C) = - \sum_{c \in C} p(c) \log_2 p(c), \quad (4)$$

$$H(C|A) = - \sum_{a \in A} p(a) \sum_{c \in C} p(c|a) \log_2 p(c|a). \quad (5)$$

The amount by which the entropy of the class decreases reflects the additional information about the class provided by the attribute and is called information gain [5].

Each attribute A_i is assigned a score based on the information gain between itself and the class:

$$\begin{aligned} \text{IG}_i &= H(C) - H(C|A_i) \\ &= H(A_i) - H(A_i|C) \\ &= H(A_i) + H(C) - H(A_i, C). \end{aligned} \quad (6)$$

Data sets with numeric attributes are first discretized using the method of Fayyad and Irani [4].

B. Relief

```

set all weights  $W[A] = 0.0$ 
for  $i = 1$  to  $m$  do
  begin
    randomly select an instance  $R$ 
    find  $k$  nearest hits  $H_j$ 
    for each class  $C \neq \text{class}(R)$  do
      find  $k$  nearest misses  $M_j(C)$ 
    for  $A = 1$  to #attributes do
       $W[A] = W[A] - \sum_{j=1}^k \text{diff}(A, R, H_j) / (m \times k) +$ 
       $\sum_{C \neq \text{class}(R)} \left[ \frac{P(C)}{1 - P(\text{class}(R))} \sum_{j=1}^k \text{diff}(A, R, M_j(C)) \right] / (m \times k)$ 
    end
  end

```

Fig. 2. ReliefF algorithm.

Relief is an instance based attribute ranking scheme introduced by Kira and Rendell [8] and later enhanced by Kononenko [9]. Relief works by randomly sampling an instance from the data and then locating its nearest neighbour from the same and opposite class. The values of the attributes of the nearest neighbours are compared to the sampled instance and used to update relevance scores for each attribute. This process is repeated for a user specified number of instances m . The rationale is that a useful attribute should differentiate between instances from different classes and have the same value for instances from the same class.

Relief was originally defined for two-class problems and was later extended (ReliefF) to handle noise and multi-class data sets [9]. ReliefF smoothes the influence of noise in the data by averaging the contribution of k nearest neighbours from the same and opposite class of each sampled instance instead of the single nearest neighbour. Multi-class data sets are handled by finding nearest neighbours from each class that is different from the current sampled instance and weighting their contributions by the prior probability of each class. Figure 2 shows the ReliefF algorithm. The function $\text{diff}(\text{Attribute}, \text{Instance1}, \text{Instance2})$ computes the difference between the values of *Attribute* for two instances. For discrete attributes the difference is either 1 (the values are different) or 0 (the values are the same), while for continuous attributes the difference is the actual difference normalised to the interval $[0,1]$.

Kononenko [9] notes that the higher the value of m (the number of instances sampled), the more reliable ReliefF's estimates are—though of course increasing m increases the running time. For all experiments reported in this paper, we set $m = 250$ and $k = 10$ as suggested in [9], [10].

C. Principal Components

Principal component analysis is a statistical technique that can reduce the dimensionality of data as a by-product of *transforming* the original attribute space. Transformed attributes are formed by first computing the covariance matrix of the original attributes, and then extracting its eigenvectors. The eigenvectors (principal components) define a linear transformation from the original attribute space to a new space in which attributes are uncorrelated. Eigenvectors can be ranked according to the amount of variation in the original data that they account for. Typically the first few transformed attributes account for most of the variation in the data and are retained, while the remainder are discarded.

It is worth noting that of all the attribute selection techniques compared, principal components is the only unsupervised method—that is, it makes no use of the class attribute. Our implementation of principal components handles k -valued discrete attributes by converting them to k binary attributes. Each of these attributes has a '1' for every occurrence of the corresponding k 'th value of the discrete attribute, and a '0' for all other values. These new synthetic binary attributes are then treated as numeric attributes in the normal manner. This has the disadvantage of increasing the dimensionality of the original space when multi-valued discrete attributes are present.

D. CFS

CFS (Correlation-based Feature Selection) [11], [12] is the first of the methods that evaluate subsets of attributes rather than individual attributes. At the heart of the algorithm is a subset evaluation heuristic that takes into account the usefulness of individual features for predicting the class along with the level of inter-correlation among them. The heuristic (Equation 7) assigns high scores to subsets containing attributes that are highly correlated with the class and have low inter-correlation with each other.

$$\text{Merit}_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}, \quad (7)$$

where Merit_s is the heuristic “merit” of a feature subset S containing k features, $\overline{r_{cf}}$ the average feature-class correlation, and $\overline{r_{ff}}$ the average feature-feature inter-correlation. The numerator can be thought of as giving an indication of how predictive a group of features are; the denominator of how much redundancy there is among them. The heuristic handles irrelevant features as they will be poor predictors of the class. Redundant attributes are discriminated against as they will be highly correlated with one or more of the other features. Because attributes are treated independently, CFS cannot identify strongly interacting features such as in a parity problem. However, it has been shown that it can identify useful attributes under moderate levels of interaction [11].

In order to apply Equation 7 it is necessary to compute the correlation (dependence) between attributes. CFS first discretizes numeric features using the technique of Fayyad and Irani [4] and then uses symmetrical uncertainty to estimate the degree of association between discrete features (X and Y):

$$SU = 2.0 \times \left[\frac{H(X) + H(Y) - H(X, Y)}{H(X) + H(Y)} \right]. \quad (8)$$

After computing a correlation matrix CFS applies a heuristic search strategy to find a good subset of features according to Equation 7. As mentioned at the start of this section we use the modified forward selection search, which produces a list of attributes ranked according to their contribution to the goodness of the set.

E. Consistency-based Subset Evaluation

Several approaches to attribute subset selection use class consistency as an evaluation metric [13], [14]. These methods look for combinations of attributes whose values divide the data into subsets containing a strong single class majority. Usually the search is biased in favour of small feature subsets with high class consistency. Our consistency-based subset evaluator uses Liu and Setiono's [14] consistency metric:

$$Consistency_s = 1 - \frac{\sum_{i=0}^J |D_i| - |M_i|}{N}, \quad (9)$$

where s is an attribute subset, J is the number of distinct combinations of attribute values for s , $|D_i|$ is the number of occurrences of the i th attribute value combination, $|M_i|$ is the cardinality of the majority class for the i th attribute value combination and N is the total number of instances in the data set.

Data sets with numeric attributes are first discretized using the method of Fayyad and Irani [4]. The modified forward selection search described at the start of this section is used to produce a list of attributes, ranked according to their overall contribution to the consistency of the attribute set.

F. Wrapper Subset Evaluation

As described at the start of this section Wrapper attribute selection uses a target learning algorithm to estimate the worth of attribute subsets. Cross-validation is used to provide an estimate for the accuracy of a classifier on novel data when using only the attributes in a given subset. Our implementation uses repeated five-fold cross-validation for accuracy estimation. Cross-validation is repeated as long as the standard deviation over the runs is greater than one percent of the mean accuracy or until five repetitions have been completed [3]. The modified forward selection search described at the start of this section is used to produce a list of attributes, ranked according to their overall contribution to the accuracy of the attribute set with respect to the target learning algorithm.

Wrappers generally give better results than filters because of the interaction between the search and the learning scheme's inductive bias. But improved performance comes at the cost of computational expense—a result of having to invoke the learning algorithm for every attribute subset considered during the search.

III. EXPERIMENTAL METHODOLOGY

Our benchmark experiment applied the six attribute selection techniques to fifteen standard machine learning data sets from the UCI collection [15]. These data sets range in size from less than 100 instances up to several thousand, with each having less than 100 attributes. A further three data sets (also available from the UCI repository) are included in order to see how the attribute selection techniques fare in situations where there are larger numbers of features. The full characteristics of all

TABLE I
DATA SETS.

Data Set	Train size	Test size	Num.	Nom.	Classes
anneal	898	CV	6	32	5
breast-c	286	CV	0	9	2
credit-g	1000	CV	7	13	2
diabetes	768	CV	8	0	2
glass-2	163	CV	9	0	2
horse colic	368	CV	7	15	2
heart-c	303	CV	6	7	2
heart-stat	270	CV	13	0	2
ionosphere	351	CV	34	0	2
labor	57	CV	8	8	2
lymph	148	CV	3	15	4
segment	2310	CV	19	0	7
soybean	683	CV	0	35	19
vote	435	CV	0	16	2
zoo	101	CV	1	16	7
arrhythmia	298	154	206	73	13
anonymous	29589	11122	0	293	2
internet-ads	2164	1115	2	1555	2

the data sets are summarised in Table I. In order to compare the effectiveness of attribute selection, attribute sets chosen by each technique were tested with two learning algorithms—a decision tree learner (C4.5 release 8) and a probabilistic learner (naive Bayes). These two algorithms were chosen because they represent two quite different approaches to learning and they are relatively fast, state-of-the-art algorithms that are often used in data mining applications. The naive Bayes algorithm employs a simplified version of Bayes formula to decide which class a test instance belongs to. The posterior probability of each class is calculated, given the feature values present in the instance; the instance is assigned to the class with the highest probability. Naive Bayes makes the assumption that feature values are statistically independent given the class. Learning a naive Bayes classifier is straightforward and involves simply estimating the probability of attribute values within each class from the training instances. Simple frequency counts are used to estimate the probability of discrete attribute values. For numeric attributes it is common practise to use the normal distribution [16]. C4.5 is an algorithm that summarises the training data in the form of a decision tree. Learning a decision tree is a fundamentally different process than learning a naive Bayes model. C4.5 recursively partitions the training data according to tests on attribute values in order to separate the classes. Although attribute tests are chosen one at a time in a greedy manner, they are dependent on results of previous tests.

For the first fifteen data sets in Table I the percentage of correct classifications, averaged over ten ten-fold cross validation runs, were calculated for each algorithm-data set combination before and after attribute selection. For each train-test split, the dimensionality was reduced by each attribute selector before being passed to the learning algorithms. Dimensionality reduction was accomplished by cross validating the attribute rankings produced by each attribute selector with respect to the current learning algorithm. That is, ten-fold cross validation on the training part of each train-test split was used to estimate the worth of the highest ranked attribute, the first two highest ranked attributes, the first three highest ranked attributes etc. The highest n ranked attributes with the best cross validated accuracy was chosen as the best subset. The last three data sets in Table I were split into a training set containing two thirds of the data and a test set containing the remaining data. Attribute selection was performed using the training data and each learning scheme was tested using the selected features on the test data.

For the attribute selection techniques that require data pre-processing, a copy of each training split was made for them to operate on. It is important to note that pre-processed data was *only*

used during the attribute selection process, and with the exception of Principal components—where data transformation occurs—original data (albeit dimensionally reduced) was passed to each learning scheme. The same folds were used for each attribute selector-learning scheme combination. Although final accuracy of the induced models using the reduced feature sets was of primary interest, we also recorded statistics such as the number of attributes selected, time taken to select attributes and the size of the decision trees induced by C4.5.

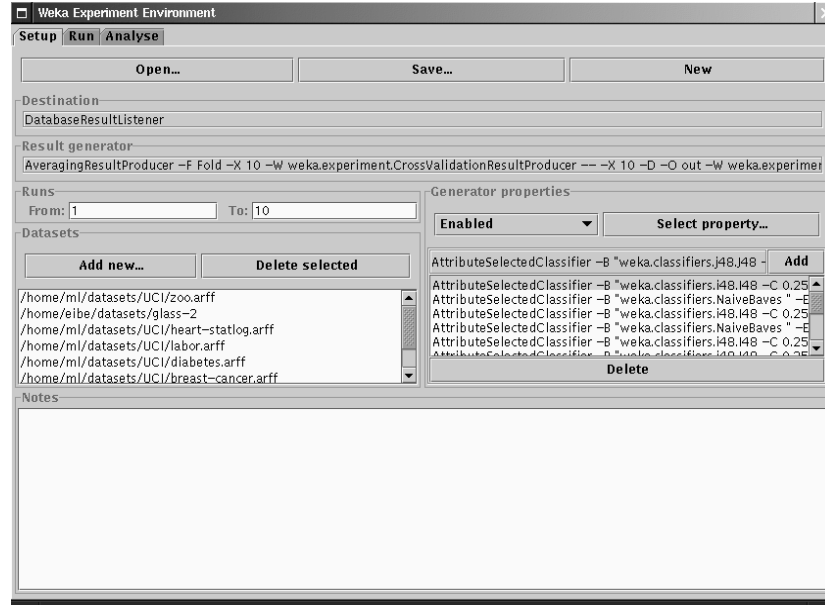


Fig. 3. Weka Experimenter.

TABLE II
RESULTS FOR ATTRIBUTE SELECTION WITH NAIVE BAYES

Data Set	NB	IG	RLF	CNS	PC	CFS	WRP
anneal	86.51	87.06 ○	89.17 ○	89.71 ○	90.65 ○	87.16	92.91 ○
breast-c	73.12	72.84	70.99 ●	71.79	73.54	73.01	72.28
credit-g	74.98	74.36	74.49 ●	74.06 ●	73.3 ●	74.33	74.35
diabetes	75.73	76.24	75.95	75.64	74.42 ●	76.19	76.12
glass2	62.33	67.42 ○	63.83 ○	68.31 ○	66.74 ○	71.08 ○	75.06 ○
horse colic	78.28	83.2 ○	82.58 ○	82.77 ○	78.56	83.01 ○	82.61 ○
heart-c	83.83	82.54 ●	82.12 ●	82.28 ●	81.85 ●	82.64 ●	82.68 ●
heart-stat	84.37	85.11	86 ○	83.48 ●	82.07 ●	85.07	85
ionosphere	82.6	88.78 ○	89.52 ○	89.95 ○	90.72 ○	89.75 ○	91.28 ○
labor	93.93	89.17 ●	90.97 ●	92 ●	89.77 ●	89.2 ●	85.77 ●
lymph	83.24	82.63	81.47 ●	82.55	79.67 ●	82.35	84.11
segment	80.1	87.17 ○	86.97 ○	85.98 ○	90.03 ○	89.03 ○	89.57 ○
soybean	92.9	92.43 ●	92.56 ●	92.81	90.93 ●	92.46	92.64
vote	90.19	95.63 ○	95.33 ○	95.82 ○	92.32 ○	95.63 ○	95.93 ○
zoo	95.04	94.34 ●	93.37 ●	93.85 ●	93.86	93.94 ●	94.34

○, ● statistically significant improvement or degradation

A. Weka Experiment Editor

To perform the benchmark experiment we used Weka² (Waikato Environment for Knowledge Analysis)—a powerful open-source Java-based machine learning workbench that can be run on any computer that has a Java run time environment installed. Weka brings together many machine learning algorithms

²Weka is freely available from <http://www.cs.waikato.ac.nz/~ml>

and tools under a common framework with an intuitive graphical user interface. Weka has two primary modes: a data exploration mode and an experiment mode. The data exploration mode (Explorer) provides easy access to all of Weka's data preprocessing, learning, attribute selection and data visualisation modules in an environment that encourages initial exploration of the data. The experiment mode (Experimenter) allows large scale experiments to be run with results stored in a database for later retrieval and analysis. Figure 3 shows the configuration panel of the Experimenter.

IV. RESULTS ON FIFTEEN BENCHMARK DATA SETS

Table II shows the results on the first fifteen data sets for attribute selection with naive Bayes. The table shows how often each method performs significantly better (denoted by a \circ) or worse (denoted by a \bullet) than performing no feature selection (column 2). Throughout we speak of results being significantly different if the difference is statistically significant at the 1% level according to a paired two-sided t test. From Table II it can be seen that the best result is from the Wrapper which improves naive Bayes on six data sets and degrades it on two. CFS is second best with improvement on five datasets and degradation on three. The simple information gain technique (IG) results in six improvements and four degradations. The consistency method (CNS) improves naive Bayes on six data sets and degrades it on five. ReliefF gives better performance on seven data sets but also degrades performance on seven. Principal components comes out the worst with improvement on five data sets and degradation on seven.

TABLE III
WINS VERSUS LOSSES FOR ACCURACY OF ATTRIBUTE SELECTION WITH NAIVE BAYES.

Scheme	Wins– Losses	Wins	Losses
WRP	30	34	4
CFS	7	21	14
CNS	2	21	19
IG	-2	17	19
RLF	-3	19	22
NB	-7	28	35
PC	-27	17	44

Table III ranks the attribute selection schemes. A pairwise comparison is made between each scheme and all of the others. The number of times each scheme is significantly more or less accurate than another is recorded and the schemes are ranked by the total number of “wins” minus “losses”. From this table it can be seen that the Wrapper is clearly the best with 34 wins and only four losses against the other schemes. CFS and the consistency method are the only other schemes that have more wins than losses.

Table IV shows the results for attribute selection with C4.5 and Table V shows the “wins” minus “losses” ranking for each scheme when compared against the others. The results are somewhat different than for naive Bayes. The best scheme for C4.5 is ReliefF which improves C4.5's performance on two data sets and degrades it on one. It is also top of the ranking with 22 wins and only seven losses against the other schemes. Consistency is the only other scheme that is ranked higher than using no feature selection with C4.5; it improves C4.5's performance on three data sets and degrades performance on three data sets. CFS and the Wrapper are tied at fourth in the ranking. CFS improves C4.5's performance on four data sets (more than any other scheme) but also degrades performance on four datasets. The Wrapper improves performance on two datasets and degrades performance on three.

The success of ReliefF and consistency with C4.5 could be attributable to their ability to identify attribute interactions (dependencies). Including strongly interacting attributes in a reduced subset increases the likelihood that C4.5 will discover and use interactions early on in tree construction before the data becomes too fragmented. Naive Bayes, on the other hand, is unable to make use of

TABLE IV
RESULTS OF ATTRIBUTE SELECTION WITH C4.5

Data Set	C4.5	IG	CFS	CNS	RLF	WRP	PC
anneal	98.58	98.72	98.47	98.65	98.73	98.66	96.26 ●
breast-c	73.87	73.75	73.7	72.24 ●	72.77	73.43	70.62 ●
credit-g	71.18	72.72	72.99 ○	72.2	71.63	72.23	69.34 ●
diabetes	73.74	73.92	73.67	73.71	73.58	73.5	71.51 ●
glass2	77.97	78.35	78.53	77.05	79.53	76.53	66.41 ●
horse colic	85.44	84.18 ●	83.94 ●	84 ●	84.9	84.14 ●	78.18 ●
heart-c	76.64	78.95 ○	79.11 ○	80.23 ○	80.4 ○	77	82.65 ○
heart-stat	78.67	84.52 ○	85.33 ○	84.11 ○	82 ○	82.11 ○	82.22 ○
ionosphere	89.74	89.4	91.09 ○	91.05	91.43	91.8 ○	88.8
labor	80.2	80.6	81	79.73	79.53	78.33	88.6 ○
lymph	75.5	73.09 ●	73.41	75.43	76.83	76.63	74.6
segment	96.9	96.81	96.94	96.87	96.89	96.92	93.95 ●
soybean	92.48	92.4	91.14 ●	92.43	92.43	92.19	83.75 ●
vote	96.46	95.84 ●	95.65 ●	95.98 ●	95.79 ●	95.74 ●	92.07 ●
zoo	92.26	91.65	91.06 ●	93.65 ○	92.95	90.45 ●	91.49

○, ● statistically significant improvement or degradation

TABLE V
WINS VERSUS LOSSES FOR ACCURACY OF ATTRIBUTE SELECTION WITH C4.5

Scheme	Wins— Losses	Wins	Losses
RLF	15	22	7
CNS	12	20	8
C4.5	7	23	16
CFS	5	21	16
WRP	5	17	12
IG	2	15	13
PC	-46	12	58

interacting attributes because of its attribute independence assumption. Two reasons could account for the poorer performance of the Wrapper with C4.5. First, the nature of the search (forward selection) used to generate the ranking can fail to identify strong attribute interactions early on, with the result that the attributes involved are not ranked as highly as they perhaps should be. The second reason has to do with the Wrapper's attribute evaluation—five fold cross validation on the training data. Using cross validation entails setting aside some training data for evaluation with the result that less data is available for building a model.

Table VI compares the size (number of nodes) of the trees produced by each attribute selection scheme against the size of the trees produced by C4.5 with no attribute selection. Smaller trees are preferred as they are easier to interpret. From Table VI and the ranking given in Table VII it can be seen that principal components produces the smallest trees, but since accuracy is generally degraded it is clear that models using the transformed attributes do not necessarily fit the data well. CFS is second in the ranking and produces smaller trees than C4.5 on 11 data sets with a larger tree on one dataset. Information gain, ReliefF and the Wrapper also produce smaller trees than C4.5 on 11 data sets but by and large produce larger trees than CFS. Consistency produces smaller trees than C4.5 on 12 data sets and never produces a larger tree. It appears quite low on the ranking because it generally produces slightly larger trees than the other schemes.

Table VIII shows the average number of attributes selected by each scheme for naive Bayes and Table IX shows the “wins” versus “losses” ranking. Table VIII shows that most schemes (with the exception of principal components) reduce the number of features by about 50% on average. Principal components sometimes increases the number of features (an artifact of the conversion of multi-valued discrete attributes to binary attributes). From Table IX it can be seen that CFS chooses fewer features

TABLE VI
SIZE OF TREES PRODUCED BY C4.5 WITH AND WITHOUT ATTRIBUTE SELECTION.

Data Set	C4.5	IG	CFS	CNS	RLF	WRP	PC
anneal	49.75	48.45	50.06	46.83 ◦	46.73 ◦	48.63	38.94 ◦
breast-c	12.38	10.47	10.26	15.09	11.8	8.42 ◦	7.72 ◦
credit-g	125.05	57.34 ◦	60.39 ◦	61.82 ◦	68.52 ◦	63.48 ◦	10.98 ◦
diabetes	41.54	14.62 ◦	15.92 ◦	16.54 ◦	16.74 ◦	17.06 ◦	30.52 ◦
glass2	23.78	14.88 ◦	16.28 ◦	16.26 ◦	17.12 ◦	16.22 ◦	11.18 ◦
horse colic	8.57	21.18 •	25.75 •	8.81	20.64 •	20.9 •	6.42 ◦
heart-c	42.34	19.72 ◦	19.45 ◦	22.48 ◦	23.17 ◦	24.2 ◦	8.16 ◦
heart-stat	34.84	12.12 ◦	11.98 ◦	13.52 ◦	13.66 ◦	14.92 ◦	4.82 ◦
ionosphere	26.58	21.84 ◦	16.64 ◦	17.14 ◦	17.22 ◦	13.9 ◦	20.04 ◦
labor	6.96	6.22 ◦	6.1 ◦	6.18 ◦	5.48 ◦	6.13 ◦	5.88 ◦
lymph	27.41	14.71 ◦	14.35 ◦	12.26 ◦	14.56 ◦	14.43 ◦	18.18 ◦
segment	81.86	80.82	80.26	79.44 ◦	80.96	79.5	119 •
soybean	92.27	86.5 ◦	88.29 ◦	92.25	91.21	90.75	88.84
vote	10.64	9.44 ◦	8.64 ◦	9.92 ◦	9 ◦	9.72 ◦	20.44 •
zoo	15.64	13.22 ◦	13.74 ◦	13.44 ◦	13.04 ◦	13.98 ◦	13.02 ◦

◦, • statistically significant improvement or degradation

TABLE VII
WINS VERSUS LOSSES FOR C4.5 TREE SIZE

Scheme	Wins— Losses	Wins	Losses
PC	21	47	26
CFS	15	30	15
IG	13	29	16
RLF	7	25	18
CNS	6	26	20
WRP	0	22	22
C4.5	-62	6	68

compared to the other schemes—retaining around 48% of the attributes on average. The Wrapper, which was the clear winner on accuracy, is third in the ranking—retaining just over 50% of the attributes on average.

Table X shows the average number of features selected by each scheme for C4.5 and Table XI shows the “wins” versus “losses” ranking. As to be expected, fewer features are retained by the schemes for C4.5 than for naive Bayes. CFS and the Wrapper retain about 42% of the features on average. ReliefF, which was the winner on accuracy, retains 52% of the features on average. As was the case with naive Bayes, CFS chooses fewer features for C4.5 than the other schemes (Table XI). ReliefF is at the bottom of the ranking in Table XI but its larger feature set sizes are justified by higher accuracy than the other schemes.

It is interesting to compare the speed of the attribute selection techniques. We measured the time taken (in milliseconds³) to select the final subset of attributes. This includes the time taken to generate the ranking and the time taken to cross validate the ranking to determine the best set of features. Table XII shows the “wins” versus “losses” ranking for the time taken to select attributes for naive Bayes. CFS and information gain are much faster than the other schemes. As expected, the Wrapper is by far the slowest scheme. Principal components is also slow, probably due to extra data set pre-processing and the fact that initial dimensionality increases when multi-valued discrete attributes are present.

Table XIII ranks the schemes by the time taken to select attributes for C4.5. It is interesting to note that the consistency method is the fastest in this case. While consistency does not rank attributes as

³This is an approximate measure. Obtaining true cpu time from within a Java program is quite difficult.

TABLE VIII

NUMBER OF FEATURES SELECTED FOR NAIVE BAYES. FIGURES IN BRACKETS SHOW THE PERCENTAGE OF THE ORIGINAL FEATURES RETAINED.

Data Set	Orig	IG	RLF	CNS	PC	CFS	WRP
anneal	38	10.1(27%)	3.7 (10%)	5.4 (14%)	38.9(103%)	7.1 (19%)	25.4 (67%)
breast-c	9	3.8 (42%)	7.4 (82%)	5.7 (63%)	5.2 (57%)	2.7 (30%)	3.2 (36%)
credit-g	20	13.2(66%)	14.3 (72%)	13.6 (68%)	19.9(100%)	12.4 (62%)	10.7 (53%)
diabetes	8	2.7 (34%)	3.6 (45%)	4 (50%)	5.9 (74%)	2.8 (35%)	4.1 (53%)
glass2	9	2.7 (30%)	3.2 (35%)	3.9 (44%)	4.5 (50%)	2.1 (24%)	1.9 (22%)
horse colic	22	5.8 (26%)	4.1 (18%)	3.9 (18%)	22.8(104%)	5.8 (26%)	6.2 (28%)
heart-c	13	7.1 (55%)	8.6 (66%)	8.7 (67%)	3.6 (28%)	7.2 (55%)	8.7 (67%)
heart-stat	13	7.8 (60%)	9.2 (71%)	10.2 (79%)	2.6 (20%)	7.9 (61%)	10 (77%)
ionosphere	34	7.9 (23%)	8.1 (24%)	10.5 (31%)	18.1(53%)	12.6 (37%)	11.7 (34%)
labor	16	12.1(75%)	13.6 (85%)	13.7 (86%)	4.3 (27%)	11.8 (74%)	9 (56%)
lymph	18	16.6(92%)	13.1 (73%)	14.3 (79%)	15.3(85%)	15 (84%)	13.1 (73%)
segment	19	11 (58%)	11.1 (58%)	5 (26%)	15.2(80%)	7.9 (42%)	9.2 (48%)
soybean	35	30.9(88%)	31.3 (89%)	32.7 (93%)	36 (103%)	25.8 (74%)	20.8 (59%)
vote	16	1 (6%)	1.7 (11%)	2.6 (16%)	14.9(93%)	1 (6%)	3 (19%)
zoo	17	12.8(75%)	12.5 (74%)	16.3 (96%)	4.7 (28%)	13.6 (80%)	10.5 (62%)

TABLE IX

WINS VERSUS LOSSES FOR NUMBER OF FEATURES SELECTED FOR NAIVE BAYES.

Scheme	Wins— Losses	Wins	Losses
CFS	24	42	18
IG	11	35	24
WRP	9	35	26
RLF	-1	30	31
CNS	-15	21	36
PC	-28	21	49

fast as information gain, speed gains are made as a by-product of the quality of the ranking produced—with C4.5 it is faster to cross validate a good ranking than a poor one. This is because smaller trees are produced and less pruning performed early on in the ranking where the best attributes are. If poorer attributes are ranked near the top then C4.5 may have to “work harder” to produce a tree. This effect is not present with naive Bayes as model induction speed is not affected by attribute quality. Although ReliefF produces the best attribute rankings for C4.5, it is not as fast as information gain. The instance-based nature of the algorithm makes it slower at producing an attribute ranking.

V. RESULTS ON LARGE DATA SETS

Figure 4 shows the results on the three large data sets for attribute selection with naive Bayes. Error bars denote the boundaries of a 95% confidence interval.

On the arrhythmia data set information gain, ReliefF, CFS and the Wrapper improve the performance of naive Bayes. Consistency gives roughly the same performance as naive Bayes and principal components degrades the performance of naive Bayes. From Table XIV it can be seen that only a small percentage of the original number of attributes is needed by naive Bayes on this data set—CFS retains just 3% of the features, and at the other end of the scale, information gain retains 22% of the features.

On the anonymous and internet-ads data sets the confidence intervals are much tighter due to larger test set sizes. There is no result for the Wrapper on the internet-ads data set because of the length of time it would take to run⁴. All methods, with the exception of principal components, perform at roughly the same level on these two data sets. CFS is fractionally better than the others on the

⁴We estimated that about 140 days of runtime on our 1400 MHz processor would be required in order to produce the attribute ranking

TABLE X

NUMBER OF FEATURES SELECTED FOR C4.5. FIGURES IN BRACKETS SHOW THE PERCENTAGE OF THE ORIGINAL FEATURES RETAINED.

Data Set	Orig	IG	CFS	CNS	RLF	WRP	PC
anneal	38	16.6(44%)	21.3(56%)	15.5(41%)	20.4(54%)	18.2(48%)	36.4(96%)
breast-c	9	4.4(49%)	4(44%)	6.6(73%)	6.9(77%)	3.98(44%)	4.4(49%)
credit-g	20	7.8(39%)	6.7(34%)	8.1(41%)	9.1(45%)	7.7(39%)	3.9(19%)
diabetes	8	3.2(40%)	3.4(43%)	3.6(45%)	3.9(49%)	3.8(47%)	5.9(74%)
glass2	9	4.2(47%)	4.6(51%)	4.4(48%)	4.7(52%)	4(44%)	4.2(47%)
horse colic	22	3.8(17%)	3.7(17%)	2.2(10%)	3.3(15%)	4.8(22%)	2.9(13%)
heart-c	13	3.9(30%)	3.5(27%)	4(31%)	5.1(39%)	5.9(45%)	3.8(29%)
heart-stat	13	3.2(25%)	3(23%)	3.6(28%)	5.6(43%)	4.6(35%)	2.1(16%)
ionosphere	34	12.2(36%)	6.9(20%)	9.3(27%)	8.7(26%)	7.2(21%)	10.2(30%)
labor	16	3.9(24%)	2.8(18%)	6.6(41%)	6.5(40%)	3.3(21%)	3.5(22%)
lymph	18	6.8(38%)	5.3(30%)	4(22%)	4.5(25%)	5.9(33%)	9.2(51%)
segment	19	16.4(86%)	11.9(63%)	9.5(50%)	12.6(66%)	9.2(48%)	16.4(86%)
soybean	35	29.5(84%)	23.7(68%)	35(100%)	32.4(93%)	19.2(55%)	30.2(86%)
vote	16	11.6(72%)	9.6(60%)	6.5(40%)	10.6(66%)	8.6(54%)	11.2(70%)
zoo	17	11.4(67%)	9(53%)	11.2(66%)	10.5(62%)	7.1(42%)	10.5(62%)

TABLE XI

WINS VERSUS LOSSES FOR NUMBER OF FEATURES SELECTED FOR C4.5

Scheme	Wins— Losses	Wins	Losses
CFS	24	35	11
WRP	13	30	17
CNS	2	26	24
IG	-8	18	26
PC	-8	17	25
RLF	-23	11	34

internet-ads data set. On the anonymous data set the Wrapper and CFS are marginally better than the others. With the exception of principal components, Table XIV shows that CFS selects the smallest attribute sets for naive Bayes. CFS's bias in favour of predictive uncorrelated attributes is particularly well suited to naive Bayes.

Figure 5 shows the results on two of the three large data sets for attribute selection with C4.5. There are no results on the anonymous data set because the length of time needed to cross validate attribute rankings using C4.5 was prohibitive⁵. On the arrhythmia data set information gain, ReliefF and CFS improve the performance of C4.5. ReliefF has the best performance on this data set. Table XV shows the number of features retained by the attribute selection methods. As was the case for naive Bayes, good performance can be obtained on the arrhythmia data set given a small number of the original features. All methods, with the exception of principal components, perform equally well on the internet-ads data set. Figures in Table XV and Table XVI help differentiate the methods. Disregarding principal components, the smallest feature sets are given by CFS and ReliefF. These two methods retain 5% and 43% of the original attributes respectively. ReliefF, consistency and CFS make a small reduction in the size of C4.5's trees.

VI. CONCLUSIONS

This paper has presented a benchmark comparison of six attribute selection techniques that produce ranked lists of attributes. The benchmark shows that in general, attribute selection is beneficial for improving the performance of common learning algorithms. It also shows that, like learning algorithms,

⁵C4.5's runtime is dominated by the number of instances (roughly log-linear). A single ten fold cross-validation on the training data takes over an hour to complete and we estimated that over 40 days of processing would be required to evaluate a ranked list of attributes

TABLE XII

WINS VERSUS LOSSES FOR TIME TAKEN TO SELECT ATTRIBUTES FOR NAIVE BAYES.

Scheme	Wins– Losses	Wins	Losses
CFS	50	57	7
IG	49	56	7
CNS	13	38	25
RLF	-10	29	39
PC	-36	17	53
WRP	-66	4	70

TABLE XIII

WINS VERSUS LOSSES FOR TIME TAKEN TO SELECT ATTRIBUTES FOR C4.5

Scheme	Wins– Losses	Wins	Losses
CNS	34	46	12
IG	29	42	13
CFS	25	40	15
RLF	12	36	24
PC	-34	20	54
WRP	-66	4	70

there is no single best approach for all situations. What is needed by the data miner is not only an understanding of how different attribute selection techniques work, but also the strengths and weaknesses of the target learning algorithm, along with background knowledge about the data (if available). All these factors should be considered when choosing an attribute selection technique for a particular application. For example, while the Wrapper using the forward selection search was well suited to naive Bayes, using a backward elimination search (which is better at identifying attribute interactions) would have been more suitable for C4.5.

Nevertheless, the results suggest some general recommendations. The wins versus losses tables show that, for accuracy, the Wrapper is the best attribute selection scheme, if speed is not an issue. Otherwise CFS, consistency and ReliefF are good overall performers. CFS chooses fewer features, is faster and produces smaller trees than the other two, but, if there are strong attribute interactions that the learning scheme can use then consistency or ReliefF is a better choice.

REFERENCES

- [1] Avrim Blum and Pat Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 245–271, 1997.
- [2] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 3, 1997.
- [3] Ron Kohavi and George H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, pp. 273–324, 1997.
- [4] U. M. Fayyad and K. B. Irani, "Multi-interval discretisation of continuous-valued attributes," in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*. 1993, pp. 1022–1027, Morgan Kaufmann.
- [5] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA., 1993.
- [6] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representations for text categorization," in *Proceedings of the International Conference on Information and Knowledge Management*, 1998, pp. 148–155.
- [7] Yiming Yang and Jan O. Pedersen, "A comparative study on feature selection in text categorization," in *International Conference on Machine Learning*, 1997, pp. 412–420.
- [8] K. Kira and L. Rendell, "A practical approach to feature selection," in *Proceedings of the Ninth International Conference on Machine Learning*. 1992, pp. 249–256, Morgan Kaufmann.
- [9] I. Kononenko, "Estimating attributes: Analysis and extensions of relief," in *Proceedings of the Seventh European Conference on Machine Learning*. 1994, pp. 171–182, Springer-Verlag.
- [10] M. Sikojia and I. Kononenko, "An adaptation of relief for attribute estimation in regression," in *Proceedings of the Fourteenth International Conference (ICML'97)*. 1997, pp. 296–304, Morgan Kaufmann.
- [11] M. A. Hall, *Correlation-based feature selection for machine learning*, Ph.D. thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1998.

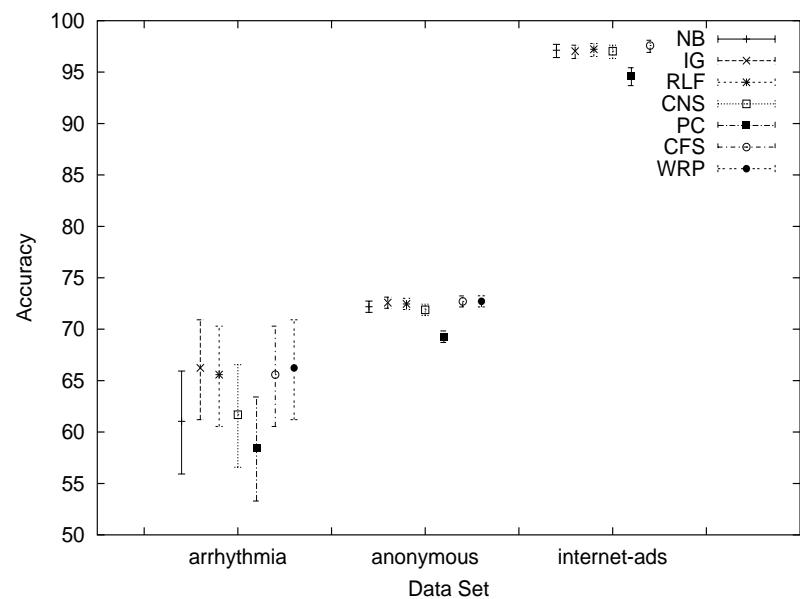


Fig. 4. Feature selection results for naive Bayes on large data sets.

TABLE XIV

NUMBER OF FEATURES SELECTED FOR NAIVE BAYES ON LARGE DATA SETS. FIGURES IN BRACKETS SHOW THE PERCENTAGE OF THE ORIGINAL FEATURES RETAINED.

Data Set	Orig	IG	RLF	CNS	PC	CFS	WRP
arrhythmia	227	51 (22%)	13 (6%)	7 (3%)	5 (2%)	6 (3%)	47 (21%)
anonymous	293	12 (4%)	18 (6%)	22 (8%)	122(42%)	11 (4%)	182 (62%)
internet-ads	1557	984(63%)	1476(95%)	602 (39%)	23 (1%)	345 (22%)	—

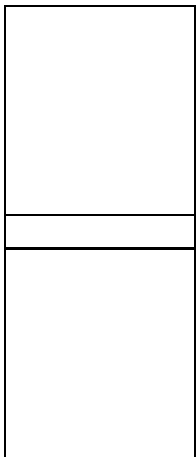
[12] Mark Hall, “Correlation-based feature selection for discrete and numeric class machine learning,” in *Proc. of the 17th International Conference on Machine Learning (ICML2000)*, 2000.

[13] H. Almuallim and T. G. Dietterich, “Learning with many irrelevant features,” in *Proceedings of the Ninth National Conference on Artificial Intelligence*. 1991, pp. 547–552, AAAI Press.

[14] H. Liu and R. Setiono, “A probabilistic approach to feature selection: A filter solution,” in *Proceedings of the 13th International Conference on Machine Learning*. 1996, pp. 319–327, Morgan Kaufmann.

[15] C. Blake, E. Keogh, and C. J. Merz, *UCI Repository of Machine Learning Data Bases*, University of California, Department of Information and Computer Science, Irvine, CA, 1998, [http://www.ics.uci.edu/~mllearn/MLRepository.html].

[16] P. Langley, W. Iba, and K. Thompson, “An analysis of Bayesian classifiers,” in *Proc. of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992, pp. 223–228, AAAI Press, [Langley92.ps.gz, from http://www.isle.org/~langley/papers/bayes.aaai92.ps].



Mark Hall received his PhD degree in computer science from the University of Waikato, New Zealand in 1999. He is currently a Lecturer and a member of the data mining group at Waikato University. He is the author of several journal and conference papers on machine learning and data mining and is a core contributor to the WEKA data mining software.

Geoffrey Holmes received his PhD from the University of Southampton, England in 1986. He is currently an Associate Professor of Computer Science at Waikato University. His primary research interests are in machine learning, data mining and data mining applications. He is head of the Waikato data mining group developing the WEKA data mining software. Details can be found at <http://www.cs.waikato.ac.nz/~ml>

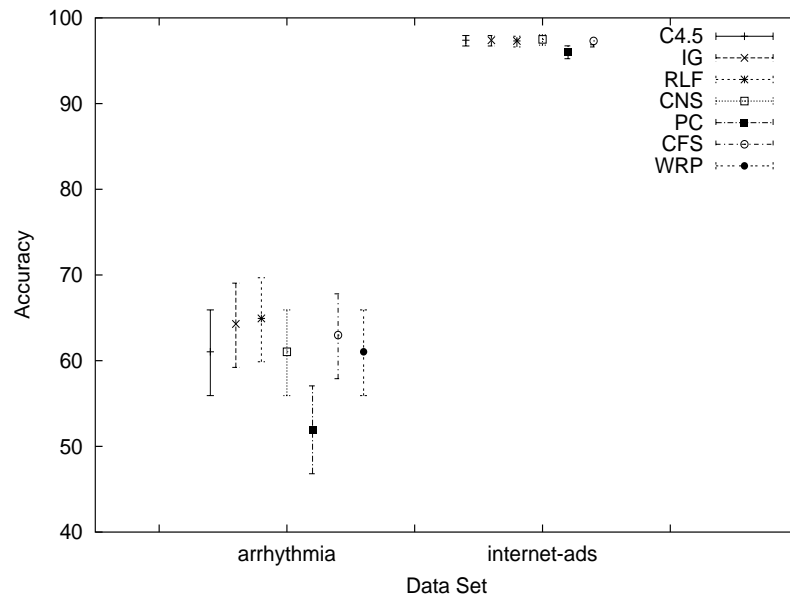


Fig. 5. Feature selection results for C4.5 on large data sets.

TABLE XV

NUMBER OF FEATURES SELECTED FOR C4.5 ON LARGE DATA SETS. FIGURES IN BRACKETS SHOW THE PERCENTAGE OF THE ORIGINAL FEATURES RETAINED.

Data Set	Orig	IG	RLF	CNS	PC	CFS	WRP
arrhythmia	227	28 (12%)	35 (15%)	19 (8%)	2 (1%)	11 (5%)	124 (55%)
internet-ads	1557	1387(89%)	663 (43%)	1094(70%)	16 (1%)	83 (5%)	—

TABLE XVI

SIZE OF TREES PRODUCED BY C4.5 WITH AND WITHOUT ATTRIBUTE SELECTION ON LARGE DATA SETS.

Data Set	C4.5	IG	CFS	CNS	RLF	WRP	PC
arrhythmia	67	61	63	63	53	61	33
internet-ads	47	47	43	43	43	—	49