Benchmarking Attribute Selection Techniques for Data Mining

Mark A. Hall
Geoffrey Holmes
Department of Computer Science, University of Waikato
Hamilton, New Zealand

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. All the methods produce an attribute ranking, a useful devise for isolating the individual merit of an attribute. Attribute selection is achieved by cross-validating the rankings with respect to a learning scheme to find the best attributes. Results are reported for a selection of standard data sets and two learning schemes C4.5 and naive Bayes.

1 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.

Attribute selection as a preprocessing step to learning generally involves a combination of search and attribute utility estimation. 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. This fact, along with the computational cost of some attribute selection techniques, has led to a situation where very few benchmark studies have been conducted.

This paper helps to fill the void by providing a benchmark comparison of 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). Section 4 presents the results. The last section summarizes the findings.

2 Attribute Selection Techniques

Attribute selection techniques can be categorized according to a number of criteria. One popular categorization has coined the terms "filter" and "wrapper" to describe the nature of the metric used to evaluate the worth of attributes [7]. 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 split further on the basis of the search technique commonly employed with each method to explore the space of attribute subsets[1]. Some attribute selection techniques can handle regression problems, that is, when the class is a numeric rather than discrete valued variable. This provides yet another dimension to categorize 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

[1]It 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

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. 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.

The rest of this section is devoted to a brief description of each of the 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.

2.1 Information Gain Attribute Ranking

This is one of the simplest (and fastest) attribute ranking methods and is often used in text categorization applications [3] where the sheer dimensionality of the data precludes more sophisticated attribute selection techniques. If A is an attribute and C is the class, Equations 1 and 2 give the entropy of the class before and after observing the attribute.

$$H(C) = -\sum_{c \in C} p(c) \log_2 p(c), \tag{1}$$

$$H(C|A) = -\sum_{a \in A} p(a) \sum_{c \in C} p(c|a) \log_2 p(c|a). \tag{2}$$

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 [10].

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

$$\begin{aligned} \mathrm{IG}_i &= H(C) - H(C|A_i) \\ &= H(A_i) - H(A_i|C) \\ &= H(A_i) + H(C) - H(A_i, C). \end{aligned} \tag{3}$$

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

2.2 Relief

Relief is an instance based attribute ranking scheme introduced by Kira and Rendell [6] and later enhanced by Kononenko [8]. 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 [8]. 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.

2.3 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. This has the disadvantage of increasing the dimensionality of the original space when multi-valued discrete attributes are present.

2.4 CFS

CFS (Correlation-based Feature Selection) [5] 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 intercorrelation among them. The heuristic (Equation 4) assigns high scores to subsets containing attributes that are highly correlated with the class and have low intercorrelation with each other.

$$Merit_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}, \tag{4}$$

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 intercorrelation. 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.

In order to apply Equation 4 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 (essentially Equation 3 normalized by the entropy of the attributes involved) to estimate the degree of association between discrete features.

2.5 Consistency-based Subset Evaluation

Several approaches to attribute subset selection use class consistency as an evaluation metric [1, 9]. 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 towards small feature subsets with high class consistency. Our consistency-based subset evaluator uses Liu and Setiono's [9] consistency metric:

$$Consistency_s = 1 - \frac{\sum_{i=0}^{J} |D_i| - |M_i|}{N}, \tag{5}$$

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 ith attribute value combination, $|M_i|$ is the cardinality of the majority class for the ith 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].

2.6 Wrapper Subset Evaluation

As described at the start of this section Wrapper attribute selection uses the 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 [7].

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.

3 Experimental Methodology

Our benchmark experiment applied the attribute selection techniques to sixteen standard machine learning data sets from the UCI collection [2]. These data sets and their characteristics are summarized in Table 1. 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.

Table 1: Data sets.

	Data Set	Instances	Num.	Nom.	Classes
1	glass-2	163	9	0	2
2	anneal	898	6	32	5
3	breast-c	286	0	9	2
4	credit-g	1000	7	13	2
5	diabetes	768	8	0	2
6	horse colic	368	7	15	2
7	heart-c	303	6	7	2
8	heart-stat	270	13	0	2
9	ionosphere	351	34	0	2
10	labor	57	8	8	2
11	lymph	148	3	15	4
12	segment	2310	19	0	7
13	soybean	683	0	35	19
14	vote	435	0	16	2
15	zoo	101	1	16	7

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. For the attribute selection techniques that require data pre-processing, a copy of each training split was made for them to operate on. 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.

3.1 Weka Experiment Editor

To perform the benchmark experiment we used Weka[2] (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 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 Explorer provides easy access to all of Weka's data preprocessing, learning, attribute selection and data visualization modules in an environment that encourages initial exploration of the data. The Experimenter allows large scale experiments to be run with results stored in a database for later retrieval and analysis. Figure 1 shows the configuration panel of the Experimenter.

4 Results

Table 2: Results for attribute selection with naive Bayes

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

○, • statistically significant improvement or degradation

[2]http://www.cs.waikato.ac.nz/∼ml

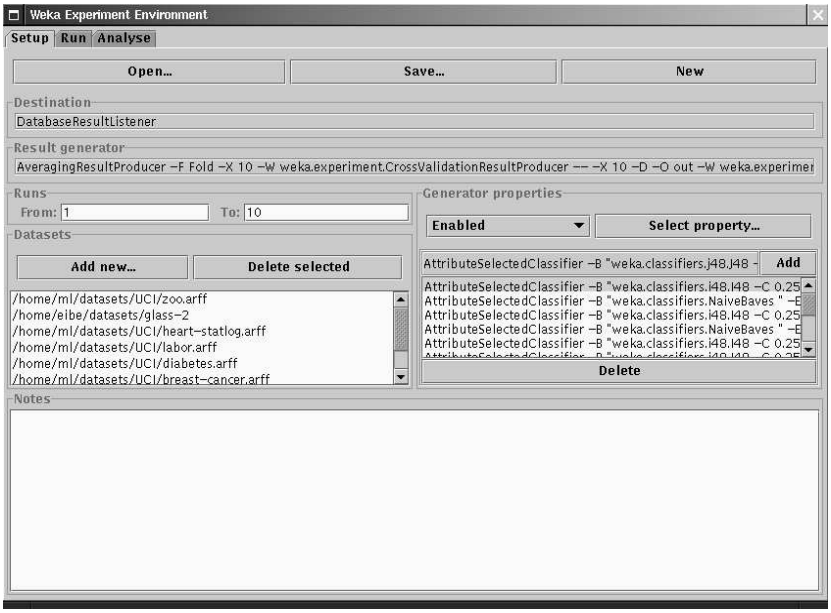

Figure 1: Weka Experimenter.

Table 3: 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 2 shows the results 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 2 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 3 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 4: Results of attribute selection with C4.5

Data Set	C4.5	IG	CFS	CNS	RLF	WRP	PC
zoo	92.26	91.65	91.06 •	93.65 ◦	92.95	90.45 •	91.49
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
diabetes	73.74	73.92	73.67	73.71	73.58	73.5	71.51 •
vote	96.46	95.84 •	95.65 •	95.98 •	95.79 •	95.74 •	92.07 •
credit-g	71.18	72.72	72.99 ◦	72.2	71.63	72.23	69.34 •
soybean	92.48	92.4	91.14 •	92.43	92.43	92.19	83.75 •
heart-c	76.64	78.95 ◦	79.11 ◦	80.23 ◦	80.4 ◦	77	82.65 ◦
glass2	77.97	78.35	78.53	77.05	79.53	76.53	66.41 •
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
breast-c	73.87	73.75	73.7	72.24 •	72.77	73.43	70.62 •
segment	96.9	96.81	96.94	96.87	96.89	96.92	93.95 •
anneal	98.58	98.72	98.47	98.65	98.73	98.66	96.26 •
horse colic	85.44	84.18 •	83.94 •	84 •	84.9	84.14 •	78.18 •

◦, • statistically significant improvement or degradation

Table 5: 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

Table 4 shows the results for attribute selection with C4.5 and Table 5 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 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 6: Size of trees produced by C4.5 with and without attribute selection.

Data Set	C4.5	IG	CFS	CNS	RLF	WRP	PC
zoo	15.64	13.22 ○	13.74 ○	13.44 ○	13.04 ○	13.98 ○	13.02 ○
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 ○
diabetes	41.54	14.62 ○	15.92 ○	16.54 ○	16.74 ○	17.06 ○	30.52 ○
vote	10.64	9.44 ○	8.64 ○	9.92 ○	9 ○	9.72 ○	20.44 ●
credit-g	125.05	57.34 ○	60.39 ○	61.82 ○	68.52 ○	63.48 ○	10.98 ○
soybean	92.27	86.5 ○	88.29 ○	92.25	91.21	90.75	88.84
heart-c	42.34	19.72 ○	19.45 ○	22.48 ○	23.17 ○	24.2 ○	8.16 ○
glass2	23.78	14.88 ○	16.28 ○	16.26 ○	17.12 ○	16.22 ○	11.18 ○
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 ○
breast-c	12.38	10.47	10.26	15.09	11.8	8.42 ○	7.72 ○
segment	81.86	80.82	80.26	79.44 ○	80.96	79.5	119 ●
anneal	49.75	48.45	50.06	46.83 ○	46.73 ○	48.63	38.94 ○
horse colic	8.57	21.18 ●	25.75 ●	8.81	20.64 ●	20.9 ●	6.42 ○

○, ● statistically significant improvement or degradation

Table 7: 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

Table 6 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 in-

terpret. From Table 6 and the ranking given in Table 7 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 8: 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
zoo	17	12.8(75%)	12.5 (74%)	16.3 (96%)	4.7 (28%)	13.6 (80%)	10.5 (62%)
heart-c	13	7.1 (55%)	8.6 (66%)	8.7 (67%)	3.6 (28%)	7.2 (55%)	8.7 (67%)
ionosphere	34	7.9 (23%)	8.1 (24%)	10.5 (31%)	18.1(53%)	12.6 (37%)	11.7 (34%)
soybean	35	30.9(88%)	31.3 (89%)	32.7 (93%)	36 (103%)	25.8 (74%)	20.8 (59%)
glass2	9	2.7 (30%)	3.2 (35%)	3.9 (44%)	4.5 (50%)	2.1 (24%)	1.9 (22%)
vote	16	1 (6%)	1.7 (11%)	2.6 (16%)	14.9(93%)	1 (6%)	3 (19%)
heart-stat	13	7.8 (60%)	9.2 (71%)	10.2 (79%)	2.6 (20%)	7.9 (61%)	10 (77%)
lymph	18	16.6(92%)	13.1 (73%)	14.3 (79%)	15.3(85%)	15 (84%)	13.1 (73%)
labor	16	12.1(75%)	13.6 (85%)	13.7 (86%)	4.3 (27%)	11.8 (74%)	9 (56%)
diabetes	8	2.7 (34%)	3.6 (45%)	4 (50%)	5.9 (74%)	2.8 (35%)	4.1 (53%)
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%)
segment	19	11 (58%)	11.1 (58%)	5 (26%)	15.2(80%)	7.9 (42%)	9.2 (48%)
horse colic	22	5.8 (26%)	4.1 (18%)	3.9 (18%)	22.8(104%)	5.8 (26%)	6.2 (28%)
anneal	38	10.1(27%)	3.7 (10%)	5.4 (14%)	38.9(103%)	7.1 (19%)	25.4 (67%)

Table 9: 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

Table 8 shows the average number of attributes selected by each scheme for naive Bayes and Table 9 shows the "wins" versus "losses" ranking. Table 8 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. From Table 9 it can be seen that CFS chooses fewer features 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 10 shows the average number of features selected by each scheme for C4.5 and Table 11 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 11). ReliefF is at the bottom of the ranking in Table 11 but its larger feature set sizes are justified by higher accuracy than the other schemes.

Table 10: 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
zoo	17	11.4(67%)	9 (53%)	11.2 (66%)	10.5 (62%)	7.1 (42%)	10.5(62%)
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%)
diabetes	8	3.2 (40%)	3.4 (43%)	3.6 (45%)	3.9 (49%)	3.8 (47%)	5.9 (74%)
vote	16	11.6(72%)	9.6 (60%)	6.5 (40%)	10.6 (66%)	8.6 (54%)	11.2(70%)
credit-g	20	7.8 (39%)	6.7 (34%)	8.1 (41%)	9.1 (45%)	7.7 (39%)	3.9 (19%)
soybean	35	29.5(84%)	23.7 (68%)	35 (100%)	32.4 (93%)	19.2 (55%)	30.2(86%)
heart-c	13	3.9 (30%)	3.5 (27%)	4 (31%)	5.1 (39%)	5.9 (45%)	3.8 (29%)
glass2	9	4.2 (47%)	4.6 (51%)	4.4 (48%)	4.7 (52%)	4 (44%)	4.2 (47%)
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%)
breast-c	9	4.4 (49%)	4 (44%)	6.6 (73%)	6.9 (77%)	3.98 (44%)	4.4 (49%)
segment	19	16.4(86%)	11.9 (63%)	9.5 (50%)	12.6 (66%)	9.2 (48%)	16.4(86%)
anneal	38	16.6(44%)	21.3 (56%)	15.5 (41%)	20.4 (54%)	18.2 (48%)	36.4(96%)

Table 11: 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

It is interesting to compare the speed of the attribute selection techniques. We measured the time taken (in milliseconds[3]) 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 12 shows the "wins" versus "losses" ranking for 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.

[3]This is a rough measure. Obtaining true cpu time from within a Java program is quite difficult.

Table 12: 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 13: 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

Table 13 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 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 quite slow to produce an attribute ranking.

5 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, 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] H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 547–552. AAAI Press, 1991.

[2] 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/∼mlearn/MLRepository.html].

[3] 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*, pages 148–155, 1998.

[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*, pages 1022–1027. Morgan Kaufmann, 1993.

[5] M. A. Hall. *Correlation-based feature selection for machine learning*. PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1998.

[6] K. Kira and L. Rendell. A practical approach to feature selection. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 249–256. Morgan Kaufmann, 1992.

[7] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

[8] I. Kononenko. Estimating attributes: Analysis and extensions of relief. In *Proceedings of the Seventh European Conference on Machine Learning*, pages 171–182. Springer-Verlag, 1994.

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

[10] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA., 1993.