Evaluating Six Candidate Solutions for the Small-Disjunct Problem and Choosing the Best Solution via Meta-Learning

Deborah R. Carvalho
Universidade Tuiti do Parana (UTP)
Computer Science Dept.
Av. Comendador Franco, 1860.
Curitiba-PR, 80215-090, Brazil
deborah@utp.br

Alex A. Freitas
Computing Laboratory
University of Kent
Canterbury, CT2 7NF, UK
A.A.Freitas@kent.ac.uk
http://www.cs.kent.ac.uk/~aaf

Abstract

A set of classification rules can be considered as a disjunction of rules, where each rule is a disjunct. A small disjunct is a rule covering a small number of examples. Small disjuncts are a serious problem for effective classification, because the small number of examples satisfying these rules makes their prediction unreliable and error-prone. This paper offers two main contributions to the research on small disjuncts. First, it investigates 6 candidate solutions (algorithms) for the problem of small disjuncts. Second, it reports the results of a meta-learning experiment, which produced meta-rules predicting which algorithm will tend to perform best for a given data set. The algorithms investigated in this paper belong to different machine learning paradigms and their hybrid combinations, as follows: two versions of a decision-tree (DT) induction algorithm; two versions of a hybrid DT/genetic algorithm (GA) method; one GA; one hybrid DT/instance-based learning (IBL) algorithm. Experiments with 22 data sets evaluated both the predictive accuracy and the simplicity of the discovered rule sets, with the following conclusions. If one wants to maximize predictive accuracy only, then the hybrid DT/IBL seems to be the best choice. On the other hand, if one wants to maximize both predictive accuracy and rule set simplicity – which is important in the context of data mining – then a hybrid DT/GA seems to be the best choice.

Keywords: classification, data mining, decision trees, genetic algorithms, instance-based learning

1. Introduction

This paper addresses the well-known classification task of data mining [17], where the objective is to predict the class of an example (record) based on the values of the predictor attributes for that example. Among the several kinds of knowledge representation that can be used to represent the knowledge discovered by a classification algorithm [21], a popular one consists of IF-THEN classification rules of the form:

```
IF <condition_1> AND ….. AND <condition_i> AND ….. AND <condition_m>
THEN <prediction (class)>
```

where each condition is typically a triple <Attribute, Operator, Value>, such as “Age < 21” or

"Gender = female". This knowledge representation has the advantage of being intuitively comprehensible for the user. From a logical viewpoint, typically the discovered rules are expressed in disjunctive normal form, where each rule represents a disjunct and each rule condition represents a conjunct. In this context, a small disjunct can be defined as a rule which covers a small number of training examples [19].

The concept of small disjunct is illustrated in Figure 1. This figure shows a decision tree where the size of the ellipsis representing each node is roughly proportional to the number of examples belonging to that node. In addition, the number inside each leaf node represents the number of examples belonging to that node. Intuitively, the two leaf nodes at the right bottom can be considered small disjuncts, since they have just 4 and 3 examples. (A more precise definition of small disjunct in the context of this research will be given in section 2, when we will revisit Figure 1.)

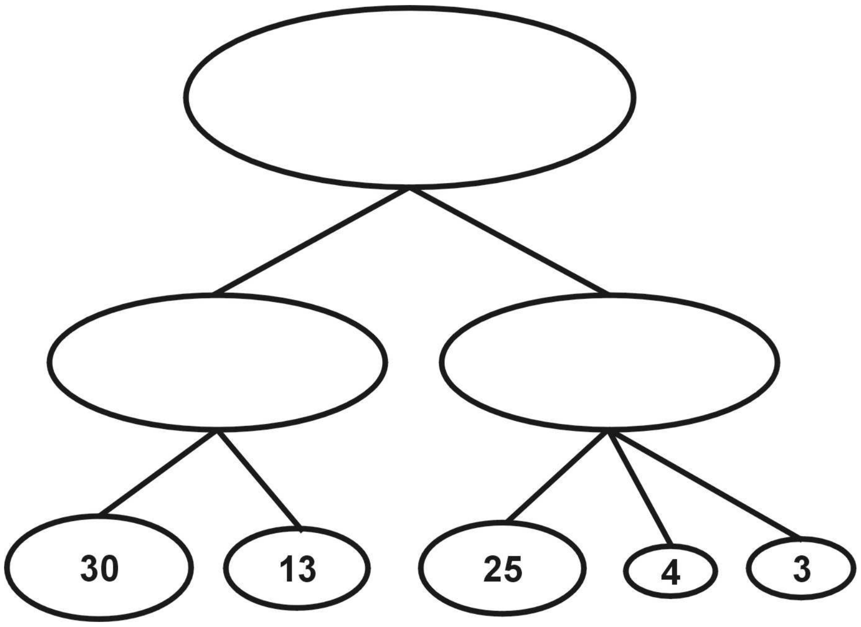

Figure 1: A hypothetical decision tree illustrating the concept of small disjuncts

The vast majority of rule induction algorithms have a bias that favors the discovery of large disjuncts, rather than small disjuncts. For instance, decision tree induction algorithms usually have a bias that favors smaller trees (whose leaf nodes are larger disjuncts) over larger trees [22]. The motivation for this bias seems clear, involving the belief that the relationship between the class and predictor attributes represented by a large disjunct, discovered in the training set, will probably generalize better to the test set. In other words, intuitively, the larger the number of examples covered by a disjunct, the more reliable the accuracy estimate associated with that disjunct. The challenge is to accurately predict the class of examples covered by small disjuncts. Clearly, this prediction is much less reliable, since the number of examples supporting the prediction is much smaller.

At first glance small disjuncts seem to have a small impact on predictive accuracy, since they contain just a small number of examples. However, in many application domains ignoring small disjuncts will lead to a significant degradation in predictive accuracy. The reason is that, even though each small disjunct covers a small number of examples, the set of all small disjuncts can cover a large number of examples. For instance, Danyluk and Provost [11] report a real-world application where small disjuncts cover roughly 50% of the training examples. In such cases we need to discover accurate small-disjunct rules in order to obtain a good predictive accuracy.

Other projects showing the relevance of the problem of small disjuncts are as follows. Weiss investigated the interaction of noise with rare cases (true exceptions) and showed that this interaction led to degradation in classification accuracy when small-disjunct rules are eliminated [29]. However, these results have a limited utility in practice, since the analysis of this interaction was made possible by using artificially generated data sets. In real-world data sets the correct concept to be discovered is not known a priori, so that it is not possible to make a clear distinction between noise and true rare cases. Weiss performed experiments showing that, when noise is added to real-world data sets, small disjuncts contribute disproportionally and significantly to the total number of classification errors made by the discovered rules [30].

More recently, Weiss and Hirsh presented a quantitative measure for evaluating the effect of small disjuncts on learning [31]. The authors reported more extensive experiments with a number of data sets to assess the impact of small disjuncts on learning, especially when factors such as training set size, pruning strategy, and noise level are varied. Their results confirmed that small disjuncts do have a negative impact on predictive accuracy in many cases.

It should be noted that the previously-mentioned projects focused mainly on understanding the problem of small disjuncts and its effect on learning. By contrast, this paper investigates several solutions for the problem of small disjuncts, where each solution corresponds to a different classification algorithm. In addition, this paper also reports the results of a meta-learning experiment, which produced meta-rules predicting which kind of algorithm will tend to perform best for a given kind of data set. Each of the algorithms investigated here has been proposed in the literature. Hence, the goal of this paper is not to propose a new algorithm for the problem of small disjuncts. Rather, it is to compare the performance of several algorithms for solving this problem. We emphasize that none of the algorithms should be considered a complete solution to this problem, which is indeed a very difficult problem and can hardly be completely solved. Rather, the algorithms investigated here should be considered as "candidate solutions" to this very difficult problem, but here we refer to them as "solutions" for short.

We have performed an extensive set of experiments, comparing 7 algorithms across 22 data sets. In essence, the 7 algorithms being compared can be categorized with respect to their machine learning paradigm [22], as follows: a) three versions of a decision-tree induction algorithm; b) two versions of a hybrid decision tree (DT)/genetic algorithm (GA) method; (c) one genetic algorithm; (d) one hybrid decision tree/instance-based learning algorithm.

Out of these 7 algorithms, 6 can be considered solutions for the problem of small disjuncts, whilst the other (a decision tree induction algorithm) is used as a baseline algorithm, as will be discussed in section 2. The performance of the algorithms is compared with respect to two criteria, namely their predictive accuracy and the simplicity of the discovered rule set.

The remainder of this paper is organized as follows. Section 2 briefly describes each of the above-mentioned 7 algorithms. Section 3 reports the results of extensive experiments evaluating the performance of the 7 algorithms. Section 4 reports the results of meta-learning experiments. Finally, section 5 concludes the paper.

2. A Summary of Classification Algorithms for the Problem of Small Disjuncts

Out of the 7 algorithms being compared in this paper, 3 of them make no distinction between small and large disjuncts. The other 4 algorithms are more flexible and have been designed specifically for solving the problem of small disjuncts. They treat small disjuncts and large disjuncts in a very different way. Hence, before we describe the algorithms themselves, we explain the criterion that we use to categorize a given rule as a small or large ("non-small") disjunct. This criterion is the same for all the 4 algorithms designed specifically for solving the small-disjunct problem, and it is based on the use of a conventional decision-tree induction algorithm, as follows.

First, one runs a decision-tree induction algorithm – viz., C4.5 [25] – and the induced (and pruned) tree is transformed into a set of IF-THEN classification rules in the usual way. In other words, each path from the root to a leaf node is transformed into a classification rule predicting the class that is the label of the corresponding leaf node. This set of classification rules is expressed in disjunctive normal form, so that each rule corresponds to a disjunct. Each rule is considered either as a small disjunct or as a "large" (non-small) disjunct, depending on whether or not its coverage (the number of examples covered by the rule) is smaller than or equal to a given threshold, called the small-disjunct size threshold (*S*). This process of identifying small disjuncts can be illustrated by revisiting Figure 1, where the numbers inside each leaf node denote the number of examples covered by the corresponding rule. If we set *S* to 3, there would be just one small disjunct in the tree (the rightmost leaf node), whilst if we set *S* to 15 there would be three small disjuncts in the tree. Hence, intuitively the value of the parameter *S* seems to have a significant influence in the performance of an algorithm that treats small disjuncts and large disjuncts in a very different way, which is the case for several algorithms discussed in this paper. Therefore, we did experiments with different values of the threshold *S*, in order to investigate the influence of this parameter in the performance of the algorithms, as will be discussed in the section on Computational Results.

We now describe each of the 7 classification algorithms compared in our experiments. Since each of these algorithms has been published in the literature, our discussion here will be brief, by focusing on the main characteristics of the algorithms and their similarities and differences. Out of the 7 algorithms, one is standard C4.5, a very well-known decision tree induction algorithm [25], which is used as a baseline algorithm throughout our experiments. The other 6 algorithms represent different solutions to the problem of small disjuncts. We will also briefly review the rationale for using each of the algorithms as a solution for the problem of small disjuncts. Of course, a more detailed description about these algorithms can be found in the references cited below. The 7 algorithms are as follows.

a) Default C4.5 – This is just C4.5, with its default parameters, including its default tree pruning method. Note that C4.5 makes no distinction between small disjuncts and large disjuncts, i.e., it does not use the parameter *S*.

b) C4.5 without pruning – This is C4.5 with its default parameters, with one exception: it returns as its output the unpruned decision tree. It is well known that in general C4.5 with pruning usually obtains a better predictive accuracy (on the test set) than C4.5 without pruning. However, in the context of our work there is a motivation for evaluating the results of C4.5 without pruning. Recall that we are looking for small-disjunct rules, which tend to be considerably more specific (i.e., have more conditions in their antecedent) than large-disjunct rules. Turning off the pruning procedure of C4.5 does lead to more specific rules. Of course, there is a danger that C4.5 without pruning will overfit the data, and this approach will produce more specific rules not only for small-disjunct examples, but also for large-disjunct examples. In any case, it is worth trying C4.5 without pruning as a possible solution for the problem of small disjuncts, since this is a very simple approach and requires no new algorithm for solving the problem of small disjunct. Hence, this approach can be regarded as a very simple solution to the problem of small disjuncts, to which more sophisticated solutions (algorithms) will be compared.

c) Double C4.5 – This is another way of using C4.5 as a classification algorithm, and it can be considered an algorithm specifically developed for solving the problem of small disjuncts [4], [5], [7]. The basic idea is to build a classifier by running C4.5 twice. The first run considers all examples in the original training set, producing a first decision tree. Once the system has identified which leaf nodes are small disjuncts, it groups all the examples belonging to the leaf nodes identified as small disjuncts into a single example subset, called the second training set. Then C4.5 is run again on this second, reduced training set, producing a second decision tree.

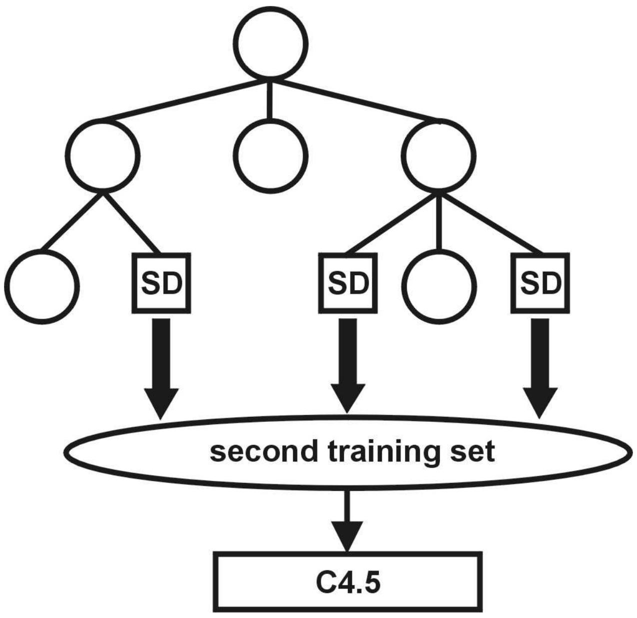

Figure 2: The basic idea of double C4.5

In order to classify a new example of the test set, the rules discovered by both runs of C4.5 are used as follows: first, the system checks whether the new example belongs to a large disjunct of the first decision tree; if so, the class predicted by the corresponding leaf node is assigned to the new example; otherwise (i.e., the example belongs to one of the small disjuncts of the first decision tree), the new example is classified by the second decision tree. This process is illustrated in Figure 2, where the leaf nodes identified as small disjuncts (in the first tree, built from the entire training set) are represented by a square with the acronym “SD” inside. The motivation for this more elaborate use of C4.5 was an attempt to create a simple algorithm that was more effective to cope with small disjuncts, by comparison with a single run of C4.5.

d) Hybrid C4.5/IB1 – This is a hybrid decision-tree algorithm (C4.5)/instance-based learning algorithm (IB1), proposed by [28]. In essence, the first step of this hybrid algorithm is to run C4.5 and identify which leaf nodes of the induced tree are considered small disjuncts, as previously discussed. The next step consists of classifying new examples in the test set (unseen during training), as follows. Each test example is pushed down the tree until it reaches a leaf node. If that leaf node is a large disjunct, the example is classified by the decision tree. On the other hand, if that leaf node is a small disjunct, the example is classified by IB1 – a simple 1-NN (one nearest neighbor) algorithm, which assigns the test example to the class of the nearest example in the data space.

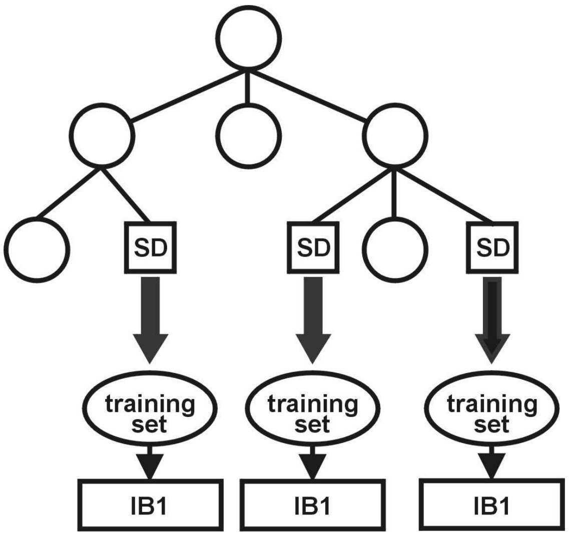

Figure 3: The basic idea of the hybrid C4.5/IB1

This process is illustrated in Figure 3, where again small disjuncts are denoted by a square with "SD" inside. IB1 uses as its training set the set of examples belonging to the corresponding leaf node of the induced tree. The motivation for this hybrid method is that the correct classification of small disjuncts tends to require a specificity bias, and, as pointed out by [28], instance-based learning seems to have the maximum specificity bias required for this kind of problem.

e) Hybrid C4.5/GA-Small – This is a hybrid C4.5/genetic algorithm (GA), proposed by [2], [3], [5]. The GA is called GA-Small, to emphasize that it is a GA specifically designed for solving the problem of small disjuncts. Let us first review the hybrid algorithm as a whole, and next briefly review GA-Small.

The basic idea of this hybrid algorithm is, at a high level of abstraction, similar to the idea of the hybrid C4.5/IB1. Again, the first step is to run C4.5 and identify which leaf nodes of the induced tree are considered small disjuncts. The major difference between the two hybrid methods lies in how they deal with small disjuncts. Instead of using IB1, the hybrid C4.5/GA-Small uses the rules discovered by GA-Small. More precisely, after all the leaf nodes considered small disjuncts are identified, the set of examples belonging to each of those leaf nodes is given as a training set to GA-Small. Therefore, GA-Small is run *k* times, each time with a different training set, where *k* is the number of leaf nodes considered small disjuncts. Each run of GA-Small discovers a rule set that will be used to classify the test examples that reach the corresponding leaf node from which GA-Small was trained. In other words, after all the *k* runs of GA-Small have been completed, each test example is pushed down the tree until it reaches a leaf node. Again, if that leaf node is a large disjunct, the example is classified by the decision tree. On the other hand, if that leaf node is a small disjunct, the example is classified by the rule set discovered by the corresponding run of GA-Small. This process is illustrated in Figure 4. The motivation for this hybrid method is that attribute interactions are considered one of the causes of small disjuncts [26], [27], [13] and GAs tend to cope better with attribute interaction than conventional greedy rule induction and decision-tree induction algorithms [14],

[10], [23], [13], [15]. In addition, by comparison with C4.5/IB1, C4.5/GA-Small has the advantage that GA-Small discovers knowledge in the form of high-level classification rules, unlike IB1.

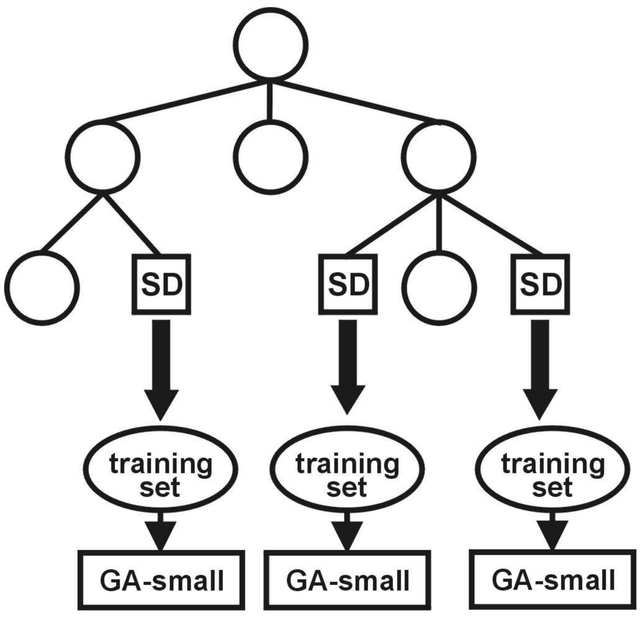

Figure 4: The basic idea of the hybrid C4.5/GA-Small

In a nutshell, the main characteristics of GA-Small are as follows. Each individual of the population represents a candidate classification rule. In addition to standard genetic operators (one-point crossover and mutation), it has a task-dependent rule pruning operator, i.e., a rule pruning operator designed specifically for pruning classification rules. This operator is applied to every individual of the population, right after the individual is formed. Unlike the usually simple operators of GA, this rule-pruning operator is an elaborate procedure based on information theory [8]. The basic idea is that, the smaller the information gain of a rule condition (an attribute-value pair), the higher the probability that that condition will be removed from the rule. The fitness function of GA-Small is: (TP / (TP + FN)) * (TN / (FP + TN)), where TP, FN, TN and FP – standing for the number of true positives, false negatives, true negatives and false positives – are well-known variables often used to evaluate the performance of classification rules [17]. Some limitations of GA-Small will be discussed in the next item.

f) Hybrid C4.5/GA-Large – This is also a hybrid C4.5/genetic algorithm (GA), proposed by [4] [7]. Although the GA component of the method was specifically designed for solving the problem of small disjuncts, this GA is called GA-Large, rather than GA-Small. The reason for this terminology is that this GA effectively learns from a large training set, rather than from a small training set. Once more, the first step is to run C4.5 and identify which leaf nodes of the induced tree are considered small disjuncts. However, instead of running GA-Small once for each small disjunct, the system groups all the examples belonging to the leaf nodes identified as small disjuncts into a single example subset, called the second training set. This is exactly the same training set used for the second run of C4.5 in the above-described "double C4.5" algorithm. The difference is that, instead of running C4.5 on the second training set, the system runs GA-Large on the second training set. This process is illustrated in Figure 5.

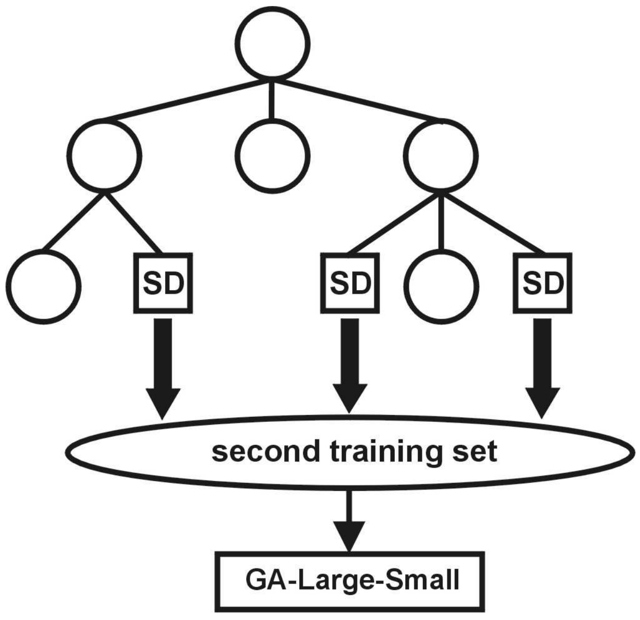

Figure 5: The basic idea of the hybrid C4.5/GA-Large

After GA-Large has run, each test example is pushed down the tree until it reaches a leaf node. Again, if that leaf node is a large disjunct, the example is classified by the decision tree. On the other hand, if that leaf node is a small disjunct, the example is classified by the rule set discovered by GA-Large.

At a high level of abstraction, the motivation for this hybrid method is the same as the motivation for the hybrid C4.5/GA-Small, i.e., attribute interactions are considered one of the causes of small disjuncts, and GAs tend to cope better with attribute interaction than conventional greedy rule induction and decision-tree induction algorithms. At a lower level of abstraction, the development of C4.5/GA-Large was motivated by the need for avoiding some limitations of C4.5/GA-Small, as follows: (a) Each run of GA-Small has access to a very small training set, consisting of just a few examples belonging to a single leaf node of a decision tree. Intuitively, this makes it difficult to induce reliable classification rules in some cases. (b) Although each run of the GA is relatively fast (since it uses a small training set), the hybrid C4.5/GA-Small as a whole has to run the GA many times (since the number of GA-Small runs is proportional to the number of small disjuncts). Hence, the hybrid C4.5/GA-Small turns out to be considerably slower than the use of C4.5 alone. (c) Since GA-Small discovers more than one rule for each leaf node considered a small disjunct, the hybrid C4.5/GA-Small discovers a larger number of rules than C4.5 alone, reducing the simplicity of discovered knowledge. The hybrid C4.5/GA-Large avoids these problems, as will be shown in the section on computational results.

It should be noted that the differences between GA-Small and GA-Large go beyond the training set used by the two GAs. Another difference is as follows. Due to an increase in the cardinality of its training set (by comparison with GA-Small), GA-Large needs to discover many rules covering the second training set. GA-large uses the “sequential covering” approach – a popular approach in conventional rule induction algorithms [32] – to discover a diverse set of rules. In essence, the first run of GA-Large is initialized with the full second training set and an empty set of rules. After each run of GA-Large, the best evolved rule is added to the set of discovered rules and the examples correctly covered by that rule are removed from the second training set. Hence, the next run of GA-Large will consider a smaller second training set. This process proceeds until all or almost all examples have been covered.

Yet another difference between GA-Small and GA-Large involves the rule pruning procedures used by these algorithms. As mentioned earlier, GA-Small uses a pruning procedure based on information theory. This procedure involves computing the information gain [8] of each attribute in a preprocessing step and using that measure to stochastically select the attributes to be removed from a given rule. Note that this is a data-driven rule pruning procedure, since the information gain values are computed directly from the training set, regardless of the result of any classification algorithm. By contrast, GA-Large uses a hypothesis-driven rule pruning procedure, which uses information about the classification accuracy (on the training set) of the induced decision tree to estimate the predictive power of each attribute, in order to decide which conditions (attribute-value pairs) should be pruned from a given rule. Hence, GA-Large can directly exploit information obtained from the hypothesis (decision tree) produced by a classification algorithm, unlike GA-Small.

These differences between the two algorithms are described in more detail in [4].

g) GA-Large alone – This algorithm consists of simply running GA-Large in the entire training set, and using the discovered rules to classify all test examples, without distinguishing between small disjuncts and large disjuncts. This algorithm is included in our experiments to determine whether or not the hybrid C4.5/GA-Large really "combines the best of both worlds", in the sense of performing better than both C4.5 and GA-Large separately.

3. Computational Results

We have performed extensive experiments to compare the effectiveness of the 7 classification algorithms described in the previous section. The experiments used 22 real-world data sets. 12 of these 22 data sets are public-domain data sets of the well-known UCI's data repository, available at: http://www.ics.uci.edu/~mlearn/MLRepository.html.

Table 1: Data sets used in the experiments

Data set	No. of examples	No. of attributes	No. of classes
Connect	67557	42	3
Adult	45222	14	2
Crx	690	15	2
Hepatitis	155	19	2
House-votes	506	16	2
Segmentation	2310	19	7
Wave	5000	21	3
Splice	3190	60	3
Covertype	8300	54	7
Letter	20000	16	26
Nursery	12960	8	5
Pendigits	10992	16	9
ds-1	5690	23	3
ds-2	5690	23	3
ds-3	5690	23	3
ds-4	5690	23	2
ds-5	5690	23	2
ds-6	5894	22	3
ds-7	5894	22	3
ds-8	5894	22	3
ds-9	5894	22	2
ds-10	5894	22	2

The other 10 data sets are derived from a database of the CNPq (the Brazilian government's National Council of Scientific and Technological Development), whose details are confidential. The database contains data about the scientific production of researchers. We have identified, with the help of the user, five possible class attributes (to be predicted) in this database. These class attributes involve information about the number of publications of researchers, with each class attribute referring to a specific kind of publication. For each class attribute, we have extracted two data sets from the database, with a somewhat different set of predictor attributes in the two data sets. This has led to the extraction of 10 data sets, denoted ds-1, ds-2, ds-3, ..., ds-10. The number of examples, attributes and classes for each of the 22 data sets is shown in Table 1. The examples that had some missing value were removed from these data sets before the classification algorithm was applied.

The performance of the seven classification algorithms was compared with respect to two criteria, namely predictive accuracy and simplicity of the discovered rule set. Let us first explain the methodology used to evaluate predictive accuracy.

All accuracy rates reported here refer to results in the test set – which was not accessed during training. In three of the public domain data sets, Adult, Connect and Letter, we have used a single division of the data set into a training and a test set. This is justified by the fact that the data sets are relatively large, so that cross-validation is not necessary [17]. In the Adult data set we used the predefined division of the data set into a training and test sets. In the Letter and the Connect data sets, since no such predefined division was available, we have randomly partitioned the data into a training and a test sets. In the Letter the data set, the training and test set had 14,000 examples and 6,000 examples, respectively. In the Connect data set the training and test sets had 47,290 and 20,267 examples, respectively. In the other public domain datasets, as well as in the 10 data sets derived from the CNPq's database, the accuracy rate on the test set was estimated by running a well-known 10-fold cross-validation procedure [17].

Intuitively, the performance of 4 out of the 7 classification algorithms described in section 2 depends on the parameter *S*, a threshold on the small disjunct size. (Recall that a decision-tree leaf is considered a small disjunct if and only if the number of examples belonging to that leaf is smaller than or equal to a fixed size *S*.) Hence, we did experiments with four different values of *S*, namely *S* = 3, 5, 10 and 15. We use the term "experiment" to refer to all the runs performed for all algorithms and for all the above-mentioned 22 data sets, for each value of *S*.

The values of *S* used in our experiments are approximately within the range of values of this parameter previously used in the literature – see, e.g., [19], [24], [11]. Note that a value of *S* significantly larger than 15 does not seem to make much sense, at least if we take into account the natural meaning of the term *small* disjunct. In addition, most of the algorithms investigated in this paper assume that C4.5 is adequate to correctly classify examples belonging to large disjuncts. If a leaf node of the tree produced by C4.5 had many more than 15 examples, it seems safe to assume that that leaf node would make a reliable prediction of the class of a new example. Otherwise C4.5 would not have generated that leaf node – it would, instead, have further expanded the tree, transforming that leaf node into an internal one and creating new branches and children nodes. In any case, we make no claim that the values of *S* used in our experiments are “optimal”. They just represent a reasonable range of different values of this parameter. Finding the “optimal” value of *S* might be done in future research, but it would be even more computationally expensive (our current experiments with four values of *S* already involve more than 15,000 algorithm runs in total) and the result would probably be data set-dependent.

There was a small difference in the number of algorithms which were run in some experiments, as follows. For $S = 10$ and $S = 15$ we have run the 7 above-mentioned algorithms. For $S = 3$ and $S = 5$ we have run 6 out of the 7 algorithms. The only algorithm which was not evaluated in these two experiments was C4.5/GA-Small. The reason is that, for $S = 3$ and $S = 5$, each of the training sets that would be used by GA-Small would have a very small cardinality. Intuitively, this small cardinality would hinder the performance of GA-Small, making it very difficult to discover reliable classification rules that generalize well from such a small number of examples. This intuition was indeed confirmed by our preliminary experiments with GA-Small. Note that, although the IB1 component of C4.5/IB1 uses the same small training sets as used by GA-Small, in general IB1 does not have any problem with the values $S = 3$ and $S = 5$. The reason is that IB1 does not try to generalize from the small set of examples to discover a classification rule. It performs classification by using the examples themselves.

In all the experiments, whenever a GA (GA-Small or GA-Large) was run, that GA was run ten times, varying the random seed used to generate the initial population of individuals. The GA-related results reported below are based on an arithmetic average of the results over these ten different random seeds. Therefore, GA-Large’s and GA-Small’s results are based on average results over 100 runs (10 seeds x 10 cross-validation folds), except in the Adult, Connect and Letter data sets, where the results were averaged over 10 runs (10 seeds). In all the experiments GA-Small and GA-Large were always run with a population of 200 individuals, and the GA was run for 50 generations. In addition, in all the experiments GA-Small and GA-Large used one-point crossover with a probability of 80%, condition mutation with a probability of 1%, and tournament selection with tournament size of 2.

In the next two subsections we report the results with respect to predictive accuracy and rule set simplicity, respectively. A subset of these results has been previously published in [4], [5], [6], [7]. This paper differs from those previous papers in two ways, as follows. On one hand, those previous papers explained in detail some of the algorithms investigated in this paper (in particular the algorithms involving GAs), whilst this paper presents just a brief summary of those algorithms. On the other hand, this paper contains more extensive computational results than those previous papers. More precisely, this paper contains results for more data sets and more values of the parameter *S*, results for two other algorithms (namely GA-Large alone and C4.5/IB1) and extensive results about rule set simplicity. The latter extension is particularly important in the context of data mining – see section 3.2.

Furthermore, this paper reports – in section 4 – new meta-learning results that have not been published yet in the literature. These results are useful to predict which algorithm tends to obtain the best performance, depending on the characteristics of the data set being mined.

3.1 Computational Results Evaluating Predictive Accuracy

The detailed results of the experiments for *S* = 3, 5, 10, and 15 are reported in Tables A.1, A.2, A.3 and A.4 (in the Appendix), respectively. These tables are put in the Appendix because they contain a large amount of detail. Hence, in this section we present and discuss only a summary of the results reported in Tables A1. to A.4. This improves the readability of the paper and is also supported by the fact that results were qualitatively similar across the four values of *S* (3, 5, 10 and 15), i.e., across Tables A.1, A.2, A.3 and A.4. Hence, our analysis of the results focus on the overall performance of each algorithm taking into account the four values of *S*.

A summary of the results of Tables A1 to A.4 is reported in Table 2. This table shows, for each value of *S*, two indicators of the performance of each algorithm. First, the columns titled “win” indicate in how many data sets each algorithm obtained a predictive accuracy *significantly higher* than the accuracy of default C4.5, which was used as a baseline method. Second, the columns titled "loss" indicate in how many data sets each algorithm obtained a predictive accuracy *significantly lower* than the accuracy of default C4.5. The definition of significantly higher (lower) performance is explained in the Appendix.

In Table 2 the columns associated with the results of C4.5/GA-Small for *S* = 3 and *S* = 5 have an N/A (Not Applicable) flag, since this algorithm was not used with these small values of the small disjunct size threshold – as discussed earlier, this is a limitation of this algorithm.

Table 2: Summary of predictive accuracy results

	C4.5 without pruning		Double C4.5		GA-Large		C4.5/IB1		C4.5 / GA-Small		C4.5 / GA-Large	
S	Win	Loss	Win	Loss	Win	Loss	Win	Loss	Win	Loss	Win	Loss
3	0	7	8	4	1	11	8	1	N/A	N/A	9	2
5	0	7	6	3	1	11	8	1	N/A	N/A	8	2
10	0	7	4	4	1	11	8	1	7	4	9	2
15	0	7	7	4	1	11	10	1	7	3	8	3

Note that in Table 2 the number of "wins" and "losses" for C4.5 without pruning and GA-Large is constant for all values of *S*. This is a consequence of the fact that these algorithms do not use the threshold size *S* to identify small disjuncts – i.e., they classify all the examples without any distinction between large disjuncts and small disjuncts. Let us now analyze the performance of each algorithm in turn.

(a) C4.5 without pruning – This algorithm obtained bad results. It did not significantly improved classification accuracy in any data set, and it significantly reduced classification accuracy (with respect to the baseline C4.5 with pruning) in 7 out of the 22 data sets.

(b) Double C4.5 – This algorithm performed considerably better. More precisely, for three values of *S*, namely *S* = 3, 5 or 15, double C4.5 significantly improved classification accuracy more often than it significantly reduced it. In the other value of *S* (10) double C4.5 obtained a "neutral" result (neither good nor bad), i.e., it significantly improved classification accuracy as often as it significantly reduced it.

(c) GA-Large – This algorithm obtained the worst results. It significantly improved classification accuracy (with respect to the baseline default C4.5) in just 1 data set, and it significantly reduced classification accuracy in 11 data sets. This result is not very surprising, considering that GA-Large was designed to classify only small-disjunct examples, rather than classifying all the examples.

(d) C4.5/IB1 – This hybrid algorithm obtained very good results. For three values of *S*, namely *S* = 3, 5 or 10, it significantly improved classification accuracy in 8 data sets, and significantly reduced classification accuracy in only 1 data set. It did even better when *S* = 15, where it significantly improved classification accuracy in 10 data sets, and significantly reduced classification accuracy in only 1 data set.

(e) C4.5/GA-Small – This hybrid algorithm obtained good results. It significantly improved the classification accuracy in 7 data sets, whereas it significantly reduced it in only 4 or 3 data sets, when *S* = 10 or 15, respectively.

(f) C4.5/GA-Large – This hybrid algorithm obtained very good results. It significantly improved classification accuracy in 8 or 9 data sets, and it significantly reduced classification accuracy in only 2 or 3 data sets, depending on the value of *S*.

To summarize, the 6 solutions (algorithms) for the problem of small disjuncts evaluated in this paper can be divided into three groups, with respect to classification accuracy. The first group consists of the most successful algorithms, namely C4.5/IB1 and C4.5/GA-Large. These algorithms can be considered very good solutions to the problem of small disjuncts – with respect to classification accuracy. The second group consists of double C4.5 and C4.5/GA-Small. Although these two algorithms were successful, in the sense of performing better (overall) than the baseline default C4.5, they were not so successful as C4.5/IB1 and C4.5/GA-Large. Finally, the third group consists of C4.5 without pruning and GA-Large alone. These two algorithms obtained bad results, considerably worse than the results of the baseline default C4.5.

It is interesting to note that there was only one data set in which both the best algorithms (C4.5/IB1 and C4.5/GA-Large) consistently obtained a predictive accuracy *significantly worse* than default C4.5 in all the four values of *S* – i.e., in Tables A.1, A.2, A.3 and A.4. The data set in question is Segmentation. The explanation for this phenomenum seems to be that this data set contains a considerable degree of class noise. This is due to the fact that the class labels for this data set were produced by manually labeling each region of an image – where the regions were previously identified by a segmentation algorithm. This kind of manual labeling tends to introduce a significant degree of class noise [1]. Indeed, [20] reports a significant improvement on predictive accuracy in this data set by using an ensemble technique particularly designed for coping with noisy data. Although class noise is a difficult problem for any classification algorithm, the problem tends to be more serious in algorithms designed for discovering small disjuncts than in (conventional) algorithms for discovering large disjuncts. The reason is that the latter have a bias in favor of discovering more general rules, and so they tend to treat small-disjunct examples as noise data. Such a bias tends to be appropriate to very noisy data sets. By contrast, algorithms designed for discovering small disjuncts have a bias in favor of discovering more specific rules, which seems appropriate if the data contains many small-disjunct examples that represent true exceptions in the data, rather than noisy data. As a result, algorithms designed for coping with small disjuncts seem particularly sensitive to high levels of class noise.

It is also important to observe how the performance of some algorithms varied for different values of the parameter *S*, the size threshold used to identify small disjuncts. Recall that this parameter is used by four algorithms – namely, double C4.5, C4.5/IB1, C4.5/GA-Small and C4.5/GA-Large. Overall, these algorithms turned out to be quite robust to different values of this parameter, as can be observed in Table 2.

3.2 Computational Results Evaluating Simplicity of the Discovered Rules

Rule set simplicity is important in the context of data mining [12], [16], [32] because it allows the user to validate discovered knowledge and use it – combined with his/her own background knowledge – to make an intelligent decision. In addition, if discovered knowledge is so complex that the user does not understand it, then the user will not trust it [18].

Hence, we have also measured the simplicity of the rule set discovered by each of the algorithms investigated in this paper, with the exception of the hybrid C4.5/IB1. The latter classifies small-disjunct examples by using the instance-based learning paradigm, which does not produce comprehensible classification rules. Note that the sizes of decision trees and rule sets are not directly comparable, because they have different structures. Hence, we converted the decision trees into rule sets, as described next, in order to make the comparison of the simplicity of the discovered rules among the algorithms as fair as possible. The simplicity of the discovered rule set was measured by the number of rules and the average number of conditions per rule. These measures were computed for each algorithm, as follows.

In the case of default C4.5 and C4.5 without pruning, first the induced tree is converted into a set of rules in the usual way. That is, each path from the root to a leaf node is transformed into a rule, whose antecedent consists of the conditions on the attribute values specified along that path and whose consequent is the class predicted by the leaf node. Hence, the number of rules is the number of leaf nodes in the tree. Next we apply a simple, *purely syntactical* post-processing method to the set of rules, with the goal of reducing the size of the rules *without changing its coverage or predictive accuracy*. This post-processing method just merges all the conditions of a rule referring to the same attribute (which often happens with continuous attributes) into a single equivalent condition. For instance, suppose that C4.5 generated a rule including the conditions “age > 21” and “age > 25”. These two conditions are converted into a single equivalent condition “age > 25”. Once this simple post-processing has been done, the average number of conditions per rule is simply computed as the total number of conditions (in all rules) divided by the number of rules.

In the case of double C4.5, C4.5/GA-Small and C4.5/GA-Large, the number of discovered rules is computed as follows. The first step is to identify in the tree generated by C4.5 the leaf nodes that are considered large disjuncts. Each path from the root node to one of these leaf nodes is converted into a rule, as described earlier. Let r_{large} be the number of large-disjunct rules created in this step. The second step is to determine the number of small-disjunct rules (called r_{small}) and the number of rule conditions in those rules. In the case of double C4.5, these values are obtained by converting the tree generated by the second execution of C4.5 (i.e., from the second training set) into a set of rules, using the same method as used for standard C4.5 and C4.5 without pruning, as described earlier. In the case of C4.5/GA-Small and C4.5/GA-Large, the number or small-disjunct rules (r_{small}) and the number of conditions in those rules are simply the number of rules and conditions discovered by the GA component of the hybrid method.

For each of these three algorithms (double C4.5, C4.5/GA-Small and C4.5/GA-Large) the total number of discovered rules is the number of large-disjunct rules (r_{large}) – which is the same for the three methods – plus the number of small-disjunct rules discovered by the algorithm in question. The average number of conditions per rule is the total number of conditions (in large-disjunct rules or in small-disjunct rules) divided by the total number of rules (r_{large} + r_{small}).

In the case of GA-Large alone, the system just has to count the number of discovered rules and the corresponding average number of conditions, since this algorithm does not distinguish between large-disjunct and small-disjunct rules.

As explained earlier, C4.5/GA-Small discovers a larger number of rules than default C4.5, because, for each small disjunct identified in the tree generated by C4.5, the GA component of this hybrid algorithm discovers several rules. However, double C4.5 and C4.5/GA-Large do not have this disadvantage. These two algorithms can discover a total number of rules (r_{large} + r_{small}) considerably smaller than the number of rules discovered by standard C4.5. This is due to the fact that these two methods use a relatively large second training set, which provides them with an opportunity to discover few rules – each of them with a large coverage – in order to cover the small disjunct examples.

The number of discovered rules and average number of conditions per rule was computed for each algorithm (except C4.5/IB1, as mentioned earlier) in each of the 22 data sets, for each of the previously-mentioned values of the parameter *S*, namely *S* = 3, 5, 10 and 15 (except C4.5/GA-Small for *S* = 3 and *S* = 5, as explained earlier). Since the full set of results would occupy too much space of this paper, we present here a summary of these results, shown in Table 3. Note that, when comparing two rule sets, it is not easy to determine which rule set is simpler when one rule set has a smaller number of rules but a larger number of conditions per rule. In order to avoid this problem and simplify the analysis of the results, Table 3 was generated by considering that the simplicity of a rule set RS_1 is better than the simplicity of a rule set RS_2 if and only if RS_1 dominates RS_2 in the following sense (inspired by the concept of Pareto dominance often used in the literature on multiobjective optimization [9]): RS_1 has a *significantly better* simplicity than RS_2 in at least one of the two simplicity criteria (number of rules and average number of conditions per rule); and RS_1 is *not significantly worse* than RS_2 in any of the two simplicity criteria. The meaning of *significantly better/worse* here is the same as in the discussion of the predictive accuracy results (see Appendix), i.e., a difference in the number of rules or conditions discovered by two algorithms is deemed significant if the corresponding intervals (taking into account the standard deviations) do not overlap.

Table 3 shows, for each value of *S*, two indicators of the performance of each algorithm. First, the columns titled “win” indicate in how many data sets each algorithm discovered a rule set *significantly simpler* than the rule set discovered by default C4.5, which was used as a baseline method. Second, the columns titled “loss” indicate in how many data sets each algorithm discovered a rule set *significantly more complex* than the rule set discovered by default C4.5.

Table 3: Summary of rule set simplicity results

	C4.5 without pruning		Double C4.5		GA-Large		C4.5 / GA-Small		C4.5 / GA-Large	
S	Win	Loss	Win	Loss	Win	Loss	Win	Loss	Win	Loss
3	0	12	4	0	17	0	N/A	N/A	2	0
5	0	12	4	0	17	0	N/A	N/A	6	0
10	0	12	5	0	17	0	0	13	15	0
15	0	12	3	1	17	0	0	13	18	0

Note that in Table 3, analogously to Table 2, the number of significant “wins” and “losses” for C4.5 without pruning and GA-Large is constant for all values of *S*. Again, this is a consequence of the fact that these algorithms classify all the examples without any distinction between large disjuncts and small disjuncts. Let us now analyze the performance of each algorithm in turn.

(a) C4.5 without pruning – This algorithm significantly degraded simplicity (by comparison with the baseline default C4.5 with pruning) in 12 out of the 22 data sets. Of course, this bad result was expected, since an unpruned tree tends to be considerably larger than a pruned tree.

(b) Double C4.5 – This algorithm obtained reasonably good simplicity results. More precisely, for *S* = 3, 5 or 10 double C4.5 generated a rule set significantly simpler than default C4.5 in 4 or 5 data sets, and its discovered rule set was not significantly more complex than default C4.5’s rule set in any data set. When *S* = 15 the performance of double C4.5 was not so good, but it still obtained a “positive” performance (3 wins against 1 loss).

(c) GA-Large – This algorithm obtained very good simplicity results. It discovered a rule set significantly simpler than default C4.5’s rule set in 17 out of the 22 data sets. Unfortunately, however, this algorithm obtained bad predictive accuracy results, as shown earlier. Hence, this algorithm by itself seems too much biased towards the discovery of general rules, with a high coverage. One should recall, however, that this algorithm was not designed to be used as a stand alone classifier. It was designed to discover only small-disjunct rules, as mentioned earlier.

(d) C4.5/GA-Small – This algorithm obtained bad simplicity results. It discovered a rule set significantly more complex than default C4.5's rule set in 13 out of the 22 data sets. This kind of result was expected, since, as explained earlier, the GA component of this hybrid algorithm discovers several rules for each leaf node of the tree built by C4.5 that is considered a small disjunct.

(e) C4.5/GA-Large – This algorithm obtained, overall, very good simplicity results. For the four values of *S*, there was no data set where this algorithm discovered a rule set significantly more complex than the one discovered by default C4.5. In addition, this algorithm discovered a rule set significantly simpler than the one discovered by default C4.5 in 2, 6, 15 or 18 data sets, for the values of *S* = 3, 5, 10, or 15, respectively.

To summarize, overall the five algorithms analyzed in this section can be ranked as follows, with respect to the simplicity of the rule sets discovered by them. The three best algorithms are, in this order, GA-Large, C4.5/GA-Large and double C4.5. The performance of the second best algorithm (C4.5/GA-Large) is about as good as the performance of the best algorithm (GA-Large) for the two largest values of *S*, viz. *S* = 10 or 15; and the performance of C4.5/GA-Large is about as good as the performance of the third best algorithm (double C4.5) for the two smallest values of *S*, viz. *S* = 3 or 5. In the fourth and last position of the ranking we can include both C4.5 without pruning and GA-Small, since both obtain very similar and very bad performances with respect to rule set simplicity.

4. Meta-Learning Results

The extensive set of computational results reported in the previous section has motivated us to apply a "meta-learning" algorithm to those results, in order to learn rules predicting which of the previously-discussed algorithms will obtain the highest predictive accuracy for a given data set. These new meta-learning experiments were carried out as follows.

A meta-learning data set was created containing 11 predictor meta-attributes, a class meta-attribute and 88 meta-examples. Each of the 88 meta-examples corresponds to one combination of data set and value of the parameter *S* (22 data sets x 4 values of *S* = 88 meta-examples). For each meta-example, the meta-class attribute value is the name of the algorithm that obtained the highest predictive accuracy for the corresponding data set and value of *S*. Hence, there are seven meta-class values, namely: default C4.5, C4.5 without pruning, double C4.5, GA-Large alone, C4.5/IB1, C4.5/GA-Small and C4.5/GA-Large. The 11 predictor meta-attributes were defined as follows.

a) Error in small disjunct classification (SD-error) – This is a continuous meta-attribute. Its value is given by the formula: (x/y)100, where x is the number of training examples (in the base data set) belonging to small disjuncts wrongly classified by C4.5 and y is the number of training examples belonging to small disjuncts (identified by running C4.5), for a given value of *S*.

b) C4.5's error rate (C4.5-error) – This is a continuous meta-attribute whose value is the error rate obtained by default C4.5 in the training set.

c) Number of small disjuncts (Num-SD) – This is a continuous meta-attribute, whose value is simply the number of small disjuncts in the training set.

d) Average size of small disjuncts (SD-size) – This is a continuous meta-attribute, whose value is the average number of examples per small disjunct of the training set.

e) Percentage of examples in small disjuncts (SD-perc) – This is a continuous meta-attribute, whose value is the ratio of the number of training examples belonging to a small disjunct divided by the total number of training examples.

f) Number of examples (Num-Examp) – This is a categorical meta-attribute indicating that the number of examples in the training set is in one of the following categories: very small (less than 1,000 examples), small (between 1,000 and 5,000 examples), medium (between 5,000 and 20,000 examples), and large (20,000 or more examples). These thresholds were manually chosen. Of course, the term "large" has to be interpreted in the context of the size of the data sets used in the experiments, rather than in the usual sense of the term in data mining.

g) Number of classes (Num-Classes) – a continuous meta-attribute. This and the next three meta-attributes have a self-explained meaning.

h) Number of categorical attributes (Num-Cat-Att) – a continuous meta-attribute.

i) Number of continuous attributes (Num-Con-Att) – a continuous meta-attribute.

j) Total Number of attributes (Num-Att) – a continuous meta-attribute, whose value is given by the summation of the values of the two previous meta-attributes.

k) Imbalance of Class Distributions (Class-Imbal) – This is a categorical meta-attribute indicating that the degree of imbalance of class distributions in the base data set belongs to one of the following categories: strongly imbalanced, imbalanced, and balanced. The category to be assigned to a particular data set is computed by the following procedure (again, the thresholds were manually chosen):

```
IF ((FreqMaj – FreqMin) > 70%) OR (FreqMin < 1%)
   THEN “strongly imbalanced”
   ELSE IF ((FreqMaj – FreqMin) > 25%)
            THEN “imbalanced”
            ELSE “balanced”
```

Where FreqMaj is the relative frequency of the majority class (in %) and FreqMin is the relative frequency of the minority class (also in %).

Note that the values of SD-error, Num-SD, SD-size and SD-perc are directly dependent on the value of *S*. The values of the other meta-attributes are independent of the value of *S*.

The values of the previously-defined attributes were computed for each combination of data set and value of *S* used in our experiments, giving a total of 88 meta-examples, as mentioned before. Once this meta-data set is available, we could apply any classification algorithm to it. In this work we applied the five algorithms that obtained at least reasonable results in the experiments reported in the previous section, i.e., default C4.5, double C4.5, C4.5/GA-Large, C4.5/GA-Small and C4.5/IB1. (The only two algorithms not tried were C4.5 without pruning and GA-Large alone, which obtained bad results in the previous sections.) In these meta-learning experiments we also applied those five algorithms with the same four different values of *S* as used in the previous section (i.e., *S* = 3, 5, 10, 15).

Table 4 reports the predictive accuracy in the test set measured by a 10-fold cross-validation procedure. As can be seen in the table, for each value of *S* the best result was obtained by the C4.5/GA-Large algorithm – whose results are shown in bold. That algorithm also obtained the best results concerning the number of discovered rules and the number of conditions per rule, for each value of *S*, as shown in Tables 5 and 6. The results in these two tables were also measured by 10-fold cross-validation.

Table 4: Accuracy rate (%) on the test set in meta-learning experiments

Algorithm	S = 3	S = 5	S = 10	S = 15
Default C4.5	77.62	77.62	77.62	77.62
Double C4.5	79.45	73.83	63.44	63.44
C4.5/GA-Large	**81.82**	**84.11**	**79.97**	**79.97**
C4.5/GA-Small	76.19	75.61	77.02	74.80
C4.5/IB1	77.75	73.40	72.92	72.92

Table 5: Number of discovered rules in meta-learning experiments

Algorithm	S = 3	S = 5	S = 10	S = 15
Default C4.5	19	19	19	19
Double C4.5	13	17	23	23
C4.5/GA-Large	**12**	**13**	**17**	**17**
C4.5/GA-Small	25	33	41	41

Table 6: Number of conditions per rule in meta-learning experiments

Algorithm	S = 3	S = 5	S = 10	S = 15
Default C4.5	4.7	4.7	4.7	4.7
Double C4.5	**2.6**	**2.5**	**2.2**	**2.2**
C4.5/GA-Large	3.9	3.7	3.5	3.5
C4.5/GA-Small	5.3	4.9	5.2	5.2

Since the best predictive accuracy among all the entries in Table 4 was obtained by running C4.5/GA-Large with *S* = 5, we have run this algorithm with this value of the parameter *S* in the entire meta-data set (with the 88 meta-examples), in order to produce the final set of rules to be analyzed as a means of getting insight into the difficult problem of predicting which algorithm (among the seven ones investigated in this paper) will obtain the highest predictive accuracy in a given data set. This final rule set is shown in Figure A.1 in the Appendix.

As can be observed in that rule set, overall, the two meta-attributes with the greatest predictive power were Num-SD and Num-Examp. Indeed, Num-SD was chosen by C4.5 as the attribute to label the root node of the tree, and it was also used in 6 out of the 9 rules discovered by GA-Large. Num-Examp appears in a tree level right below the root in all the subtrees shown in Figure A.1, and it appears in 8 out of the 9 rules discovered by GA-Large.

Another point to be noted is that in total there are 3 rules predicting that the algorithm with highest predictive accuracy will be C4.5/IB1 and 4 rules predicting that the algorithm with highest predictive accuracy will be C4.5/GA-Large. The other algorithms have fewer rules predicting their superiority. This is, of course, consistent with the fact that C4.5/IB1 and C4.5/GA-Large were most often the winners – with respect to predictive accuracy – in the experiments reported in section 3.1. More precisely, analyzing Tables A1 to A4 we find that C4.5/IB1 was the winner in 17 cases and C4.5/GA-Large was the winner in 37 cases.

Since the two most successful algorithms with respect to predictive accuracy were C4.5/IB1 and C4.5/GA-Large, it is important to analyze in more detail the rules discovered in this meta-learning experiment predicting when each of these two algorithms will be the winner. The 3 rules predicting that the winner will be C4.5/IB1 are as follows:

```
IF Num-SD > 444 THEN  C4.5/IB1 (8)
IF Num-SD < 307 AND Num-Examp in {Large,Medium} AND Num-Classes >= 3
THEN C4.5/IB1 (11/6)
IF Num-SD >= 196 AND Num-Examp = Large AND Num-classes < 9
THEN C4.5/IB1  (5)
```

The first above rule is a large disjunct rule discovered by C4.5, whereas the other two rules are rules 8 and 9 discovered by GA-Large, as shown in Figure A.1. By contrast, the 4 rules predicting that the winner will be C4.5/GA-Large are as follows:

```
IF Num-SD <= 141 AND Num-Exampl in {Medium,Small} AND C4.5-error > 4.6%
    AND SD-error <= 56.24% AND SD-perc > 1.06%
THEN C4.5/GA-Large  (17/2)

IF 141 < Num-SD <= 353 AND Num-Examp in {Medium,Small} AND
   4.6% < C4.5-error < 39.3% AND SD-perc > 8.49% AND SD-error > 47.93%
THEN C4.5/GA-Large    (7)

IF Num-SD <= 444 AND Num-Examp = VerySmall AND C4.5-error > 7.1% AND
   SD-error <= 51.52
THEN C4.5/GA-Large   (6/2)

IF Num-SD <= 298 AND Num-Examp in {VerySmall,Medium} AND Num-Classes <= 13
THEN C4.5/GA-Large   (24/16)
```

The first three rules are large disjunct rules discovered by C4.5, whereas the fourth rule is the rule 6 discovered by GA-Large, as shown in Figure A.1. Here we have simplified the rules extracted from the decision tree, by merging two different conditions referring to the same attribute (in the same rule) into a single rule condition. For instance, in the second above rule the conditions C4.5-error > 4.6% and C4.5-error < 39.3% were merged into the single condition 4.6% < C4.5-error < 39.3%.

In general, the rules predicting C4.5/IB1 suggest that this algorithm will tend to be the winner in relatively large data sets, where the value of the meta-attribute Num-Examp is Large or Medium, or where the number of small disjuncts (Num-SD) is large. In particular the first rule predicting C4.5/IB1 has only a single condition requiring that Num-SD > 444. This rule covers 8 out of the 88 meta-examples of the meta-data set, and all those 8 meta-examples are correctly covered by this rule. In addition, the third rule predicting C4.5/IBI as the winner has two conditions requiring that Num-SD >= 196 and Num-Examp = Large. This rule also correctly classifies all the 5 examples that it covers. Out of the 3 discovered rules predicting C4.5/IBI as a winner, the only one that does not require a large number of small disjuncts or examples is the second one, which requires that Num-SD < 307 AND Num-Examp in {Large,Medium}. However, this rule is less reliable than the other two rules predicting C4.5/IB1, since the former is misclassifying 6 out of the 11 meta-examples that it covers.

By contrast, in general the rules predicting C4.5/GA-Large suggest that this algorithm will tend to be the winner in relatively small or medium-sized data sets, where the value of the meta-attribute Num-Examp is Very Small, Small or Medium, and where the number of small disjuncts (Num-SD) is not so large. In particular, 3 out of the 4 discovered rules predicting C4.5/GA-Large specify conditions of the form Num-SD <= *t*, where t is a threshold, and just one rule specifies the condition 141 < Num-SD <= 353. No discovered rule predicting C4.5/GA-Large specifies a condition of the form Num-SD > *t* – unlike, for instance, 2 discovered rules predicting C4.5/IB1. In addition, no discovered rule predicting C4.5/GA-Large has a condition specifying Num-Exampl = Large (not even Large *or* another value), unlike 2 discovered rules predicting C4.5/IB1. Another evidence that C4.5/GA-Large tends to be the winner in small data sets is the fact that, in the meta-learning experiments reported here – which certainly involve a very small meta-data set – C4.5/GA-Large was the winner, as shown in Table 4.

5. Conclusions

As mentioned earlier, the goal of this paper was not to introduce a new algorithm. Rather, the goal of this paper was to investigate the performance of 6 different kinds of algorithm – namely, two versions of a decision-tree (DT) induction algorithm; two versions of a hybrid DT/genetic algorithm (GA) method; one GA; and one hybrid DT/instance-based learning (IBL) algorithm – as potential solutions to the problem of small disjuncts.

The algorithms were evaluated in extensive experiments with 22 data sets. In total, taking into account all the iterations of the cross-validation procedure, all the different runs of the GAs with different random seeds (since GAs are stochastic methods), and the different values of the parameter *S* (small-disjunct threshold size), the number of algorithm runs was 15,247.

Overall, taking into account the results of all these experiments, the best predictive accuracy was obtained by the hybrid DT/IBL (C4.5/IB1) algorithm, and the second best predictive accuracy was obtained by a hybrid DT/GA (C4.5/GA-Large). Hence, in general these two algorithms seem suitable for mining data sets with small disjuncts, at least with respect to the goal of maximizing predictive accuracy.

However, with respect to rule set simplicity, the hybrid C4.5/IB1 has the disadvantage that IB1 (and the paradigm of IBL in general) does not discover any comprehensible rule, whilst the hybrid C4.5/GA-Large has the advantage of discovering a rule set considerably simpler than the rule set discovered by standard C4.5 alone.

Hence, the general conclusion of the experimental results is as follows: If one wants to maximize predictive accuracy only, then the hybrid C4.5/IB1 seems to be the best choice among the algorithms evaluated in this paper. On the other hand, if one wants to maximize both predictive accuracy and rule set simplicity – which is usually the goal in data mining – then the hybrid C4.5/GA-Large seems to be the best choice.

We have also performed a meta-learning experiment, in order to predict which algorithm would obtain the best predictive accuracy in a given data set, by taking into account some characteristics of the data set at hand – including characteristics related to the occurrence of small disjuncts. The results of this meta-learning experiment were prediction rules indicating that, in general: (a) C4.5/IB1 tends to be the winner in relatively large data sets, with a large number of examples or small disjuncts; (b) C4.5/GA-Large tends to be the winner in relatively small or medium-sized data sets, with a small or medium number of examples and with a not very large number of small disjuncts.

To the best of our knowledge, this is the first paper to perform such an extensive investigation of 6 different solutions to the problem of small disjuncts, and the first paper to report meta-learning results for the problem of small disjuncts.

An interesting research direction involves another rule quality criterion: rule surprisingness (novelty, unexpectedness). The motivation for this criterion is that many rules that have a high predictive accuracy and are highly comprehensible may be uninteresting for the user, because they represent an obvious pattern in the data. The classic example is the rule “IF patient is pregnant THEN gender is female”.

Small disjuncts have a good potential to represent novel, surprising knowledge to the user, because they tend to represent exceptions in the data, by contrast with the most general patterns in the data that are probably already known by the user. Hence, it is interesting to investigate the quality of the rule set discovered by the algorithms used in this paper with respect to rule surprisingness as well. We are currently investigating this research direction.

Acknowledgment

We thank Dr. Wesley Romao for having prepared the CNPq data sets for data mining purposes, allowing us to use those data sets in our experiments.

References

[1] C. E. Brodley and M.A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research 11 (1999)¸*131-167.

[2] D. R. Carvalho and A. A. Freitas, A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in Data Mining. *Proc 2000 Genetic and Evolutionary Computation Conf. (Gecco-2000),* 2000, *1061-1068*. Las Vegas, NV, USA. July.

[3] D. R. Carvalho and A. A. Freitas, A genetic algorithm-based solution for the problem of small disjuncts. Principles of Data Mining and Knowledge Discovery (*Proc. 4th European Conf., PKDD-2000*. Lyon, France). *LNAI 1910*, 345-352. Springer-Verlag, 2000.

[4] D. R. Carvalho and A. A. Freitas. A genetic algorithm with sequential niching for discovering small-disjunct rules. *Proc. Genetic and Evolutionary Computation Conf. (GECCO-2002),* pp. 1035-1042. San Francisco, CA, USA: Morgan Kaufmann, 2002.

[5] D. R. Carvalho and A. A. Freitas. A genetic algorithm for discovering small disjunct rules in data mining. *Applied Soft Computing 2(2),* pp. 75-88. Dec. 2002.

[6] D. R. Carvalho and A. A. Freitas. New results for a hybrid decision tree/genetic algorithm

for data mining. *Proc. 4th Int. Conf. on Recent Advances in Soft Computing (RASC-2002)*, pp. 260-265. Published in CD-ROM (ISBN: 1-84233-0764), Nottingham Trent Univ, UK. 2002.

[7] D. R. Carvalho and A. A. Freitas. A hybrid decision tree/genetic algorithm method for data mining. *Special issue on Soft Computing Data Mining – Information Sciences 163(1-3)*, pp. 13-35. 14 June 2004.

[8] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley&Sons, 1991.

[9] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[10] V. Dhar, D. Chou and F. Provost, Discovering Interesting Patterns for Investment Decision Making with GLOWER – A Genetic Learner Overlaid With Entropy Reduction. *Data Mining and Knowledge Discovery 4(4)*, Oct. 2000., 251-280.

[11] A. P. Danyluk, F. J. Provost, Small Disjuncts in Action: Learning to Diagnose Erros in the Local Loop of the Telephone Network, *Proc. 10th International Conference Machine Learning*, 1993, 81-88.

[12] U. M. Fayyad, G. Piatetsky-Shapiro and P. Smyth. From data mining to knowledge discovery: an overview. In: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.) *Advances in Knowledge Discovery and Data Mining*, 1-34. AAAI/MIT Press, 1996.

[13] A. A. Freitas, Understanding the crucial role of attribute interaction in data mining. *Artificial Intelligence Review 16(3)*, Nov. 2001, 2001, 177-199.

[14] A. A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* Springer-Verlag, 2002.

[15] A. A. Freitas, Evolutionary Computation. In: W. Klosgen and J. Zytkow (Eds.) *Handbook of Data Mining and Knowledge Discovery*, 698-706. Oxford University Press, 2002.

[16] J. Han and M. Kamber. *Data Mining: concepts and techniques*. Morgan Kaufmann, 2001.

[17] D. J. Hand, *Construction and Assessment of Classification Rules*. John Wiley, 1997.

[18] R.J. Henery. Classification. In: D. Michie, D.J. Spiegelhalter and C.C. Taylor. *Machine Learning, Neural and Statistical Classification*, pp. 6-16. Ellis Horwood, 1994.

[19] R. C. Holte, L. E. Acker, and B. W. Porter, Concept Learning and the Problem of Small Disjuncts, *Proc. IJCAI – 89*, 1989, 813-818.

[20] A. Krieger, C. Long and A. Wyner. Boosting noisy data. *Proc. Eighteenth Int. Conf. on Machine Learning (ICML-2001)*, 274-281. Morgan Kaufmann, 2001.

[21] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, 1996.

[22] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[23] A. Papagelis and D. Kalles, Breeding decision trees using evolutionary techniques. *Proc. 18th Int. Conf. on Machine Learning (ICML-2001)*, 393-400. Morgan Kaufmann, 2001.

[24] J. R. Quinlan. Improved estimates for the accuracy of small disjuncts, *Machine Learning 6(1)*. p. 93-98, 1991.

[25] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publisher, 1993.

[26] L. Rendell and R. Seshu, Learning hard concepts through constructive induction: framework and rationale. *Computational Intelligence 6*, 1990, 247-270.

[27] L. Rendell and H. Ragavan, Improving the design of induction methods by analyzing algorithm functionality and data-based concept complexity. *Proc. 13th Int. Joint Conf. on Artif. Intel. (IJCAI-93)*, 1993. 952-958.

[28] K. M. Ting, The Problem of Small Disjuncts: its remedy in Decision Trees, *Proc. 10th Canadian Conference on AI*, 1994. 91-97.

[29] G. M. Weiss, Learning with Rare Cases and Small Disjuncts, *Proc. 12th International Conference on Machine Learning (ICML-95)*, 1995. 558-565.

[30] G. M. Weiss, The Problem with Noise and Small Disjuncts, *Proc. Int. Conf. Machine Learning (ICML-98)*, 1998, 574-578.

[31] G. M. Weiss and H. Hirsh, A Quantitative Study of Small Disjuncts, *Proc. of Seventeenth National Conference on Artificial Intelligence.* Austin, Texas, 2000. 665-670.

[32] I.H. Witten and E. Frank. *Data Mining*: practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, 2000.

Appendix

In Tables A.1 to A.4 the first column indicates the data set, and the other columns report the corresponding accuracy rate in the test set (in %) obtained by the algorithm indicated at the title of the column. Note that the accuracy rates of default C4.5, C4.5 without pruning and GA-Large alone in Tables A.1, A.2, A.3 and A.4 are exactly the same, since the accuracy rates of these algorithms do not depend on the value of *S*.

The numbers after the “±“ symbol denote standard deviations. For each data set, the highest accuracy rate among all the algorithms is shown in bold. In each column containing the results of an algorithm (except the column with the results of default C4.5, the baseline algorithm) we indicate, for each data set, whether or not the accuracy rate of that algorithm is significantly different from the accuracy rates of default C4.5. This allows us to evaluate the extent to which each algorithm can be considered a good solution for the problem of small disjuncts. More precisely, the cases where the accuracy rate of each algorithm is significantly better (worse) than the accuracy rate of default C4.5 is indicated by the "+" ("-") symbol. A difference between two algorithms is deemed significant when the corresponding accuracy rate intervals (taking into account the standard deviations) do not overlap.

Table A.1: Accuracy rate (%) on the test set for *S* = 3

Data Set	Default C4.5	C4.5 without pruning	Double C4.5	GA-Large alone	C4.5/IB1	C4.5/GA-Large
Connect	72.60 ± 0.5	72.05 ± 0.5	78.06 ± 0.6 +	74.40 ± 0.6 +	**78.13 ± 0.5+**	77.86 ± 0.1 +
Adult	78.62 ± 0.5	77.00 ± 0.5 -	81.19 ± 0.5 +	78.90 ± 0.3	**85.87 ± 0.5+**	85.45 ± 0.1 +
Crx	91.79 ± 2.1	92.45 ± 1.9	92.57 ± 1.2	78.04 ± 1.2 -	92.97 ± 1.5	**93.69 ± 1.2**
Hepatitis	80.78 ± 13.3	77.50 ± 11.3	78.95 ± 6.9	81.28 ± 11.2	**95.84 ± 8.3**	89.25 ± 9.5
House-votes	93.62 ± 3.2	93.50 ± 1.7	97.32 ± 2.4	**97.63 ± 1.6**	96.22 ± 3.1	97.18 ± 2.5
Segmentation	**96.86 ± 1.1**	96.30 ± 0.9	76.62 ± 2.8 -	72.42 ± 4.6 -	81.45 ± 1.2 -	81.46 ± 1.1 -
Wave	75.78 ± 1.9	75.4 ± 2.1	68.18 ± 3.7 -	66.74 ± 4.6 -	83.81 ± 2.0+	**83.86 ± 2.0 +**
Splice	65.68 ± 1.3	66.54 ± 1.3	55.65 ± 6.0 -	60.26 ± 5.0	70.54 ± 8.9	**70.62 ± 8.6**
Covertype	71.61 ± 1.9	70.34 ± 1.9	72.88 ± 14.4	65.40 ± 3.1 -	73.03 ± 13.9	**73.04 ± 1.2**
Letter	**86.4 ± 1.1**	86.30 ± 1.1	83.82 ± 1.0 -	75.80 ± 0.3 -	86.17 ± 1.1	84.26 ± 0.2 -
Nursery	95.4 ± 1.2	**96.40 ± 0.9**	95.55 ± 0.6	82.13 ± 4.4 -	95.48 ± 0.6	95.35 ± 1.2
Pendigits	96.39 ± 0.2	96.36 ± 0.3	97.43 ± 0.3 +	89.01 ± 0.5 -	97.41 ± 0.2+	**97.54 ± 0.3 +**
CD-1	60.71 ± 3.0	58.26 ± 2.8	**62.75 ± 0.6**	61.40 ± 0.3	62.55 ± 3.5	62.53 ± 1.0
CD-2	65.55 ± 1.5	63.22 ± 1.6	**68.95 ± 4.8**	62.95 ± 5.1	68.44 ± 4.7	68.44 ± 2.3
CD-3	75.65 ± 2.4	71.64 ± 1.3 -	80.47 ± 2.1 +	69.24 ± 2.2 -	80.26 ± 1.9+	**80.54 ± 0.9 +**
CD-4	**92.97 ± 0.9**	89.47 ± 0.8 -	92.75 ± 1.4	89.7 ± 2.9	92.72 ± 1.1	92.72 ± 1.1
CD-5	82.7 ± 2.8	78.71 ± 1.9	89.74 ± 2.4 +	72.31 ± 2.2 -	90.15 ± 2.6	**90.17 ± 2.5 +**
CD-6	57.78 ± 2.1	55.36 ± 2.3	59.72 ± 2.4	59.37 ± 3.7	**59.78 ± 2.5**	59.65 ± 1.2
CD-7	65.18 ± 1.0	60.68 ± 1.4 -	**69.94 ± 1.5 +**	65.90 ± 1.6	69.75 ± 1.4+	69.66 ± 0.7 +
CD-8	75.57 ± 1.4	70.30 ± 1.7 -	**80.52 ± 2.3 +**	73.69 ± 1.9	80.45 ± 2.0+	80.41 ± 2.0 +
CD-9	93.00 ± 0.5	89.67 ± 1.4 -	93.72 ± 0.7	87.49 ± 2.9 -	93.86 ± 1.5	**93.87 ± 1.4**
CD-10	82.80 ± 1.7	78.45 ± 2.2 -	85.89 ± 0.5 +	73.89 ± 1.4 -	85.80 ± 1.3+	**85.90 ± 1.2 +**

Table A.2: Accuracy rate (%) on the test set for *S* = 5

Data Set	Default C4.5	C4.5 without pruning	Double C4.5	GA-Large alone	C4.5/IB1	C4.5/GA-Large
Connect	72.60 ± 0.5	72.05 ± 0.5	77.09 ± 0.6 +	74.40 ± 0.6 +	**78.19 ± 0.5 +**	77.85 ± 0.2 +
Adult	78.62 ± 0.5	77.00 ± 0.5 -	79.27 ± 0.5	78.90 ± 0.3	**85.94 ± 0.5 +**	85.50 ± 0.2 +
Crx	91.79 ± 2.1	92.45 ± 1.9	92.03 ± 1.0	78.04 ± 1.2 -	92.54 ± 1.2	**93.06 ± 1.6**
Hepatitis	80.78 ± 13.3	77.50 ± 11.3	75.67 ± 17.1	81.28 ± 11.2	86.53 ± 10.0	**89.48 ± 9.7**
House-votes	93.62 ± 3.2	93.50 ± 1.7	93.54 ± 3.9	**97.63 ± 1.6**	97.04 ± 1.1	97.44 ± 2.9
Segmentation	**96.86 ± 1.1**	96.30 ± 0.9	74.49 ± 3.4 -	72.42 ± 4.6 -	80.22 ± 1.0 -	80.41 ± 1.0 -
Wave	75.78 ± 1.9	75.4 ± 2.1	65.59 ± 4.4 -	66.74 ± 4.6 -	85.33 ± 2.1 +	**85.37 ± 2.4 +**
Splice	65.68 ± 1.3	66.54 ± 1.3	57.45 ± 8.7	60.26 ± 5.0	70.37 ± 8.2	**70.44 ± 7.8**
Covertype	71.61 ± 1.9	70.34 ± 1.9	71.34 ± 14.4	65.40 ± 3.1 -	71.63 ± 14.3	**71.66 ± 1.3**
Letter	86.4 ± 1.1	86.30 ± 1.1	83.62 ± 1.0 -	75.80 ± 0.3 -	**88.15 ± 1.1**	83.28 ± 0.2 -
Nursery	95.4 ± 1.2	96.40 ± 0.9	**96.57 ± 0.7**	82.13 ± 4.4 -	96.39 ± 0.6	96.25 ± 0.9
Pendigits	96.39 ± 0.2	96.36 ± 0.3	97.21 ± 0.4 +	89.01 ± 0.5 -	**97.86 ± 0.3 +**	96.72 ± 0.5
CD-1	60.71 ± 3.0	58.26 ± 2.8	63.77 ± 3.7	61.40 ± 0.3	63.26 ± 4.1	**63.83 ± 1.2**
CD-2	65.55 ± 1.5	63.22 ± 1.6	**71.06 ± 5.1**	62.95 ± 5.1	70.26 ± 5.2	70.57 ± 2.4 +
CD-3	75.65 ± 2.4	71.64 ± 1.3 -	81.47 ± 1.7 +	69.24 ± 2.2 -	81.17 ± 1.9 +	**81.69 ± 0.9 +**
CD-4	**92.97 ± 0.9**	89.47 ± 0.8 -	92.58 ± 1.0	89.7 ± 2.9	92.80 ± 1.0	92.84 ± 1.1
CD-5	82.7 ± 2.8	78.71 ± 1.9	86.68 ± 1.9	72.31 ± 2.2 -	87.08 ± 1.8	**87.19 ± 2.1**
CD-6	57.78 ± 2.1	55.36 ± 2.3	**60.50 ± 2.3**	59.37 ± 3.7	60.28 ± 2.2	60.27 ± 1.2
CD-7	65.18 ± 1.0	60.68 ± 1.4 -	70.74 ± 1.8 +	65.90 ± 1.6	70.95 ± 1.9 +	**71.34 ± 1.2 +**
CD-8	75.57 ± 1.4	70.30 ± 1.7 -	**81.18 ± 2.1 +**	73.69 ± 1.9	81.11 ± 2.0+	80.98 ± 2.1 +
CD-9	93.00 ± 0.5	89.67 ± 1.4 -	93.89 ± 0.8	87.49 ± 2.9 -	93.87 ± 1.3	**93.98 ± 1.3 +**
CD-10	82.80 ± 1.7	78.45 ± 2.2 -	**86.25 ± 1.1** +	73.89 ± 1.4 -	86.08 ± 1.5 +	85.83 ± 1.4

Table A.3: Accuracy rate (%) on the test set for *S* = 10

Data Set	Default C4.5	C4.5 without pruning	Double C4.5	GA-Large alone	C4.5/IB1	C4.5/GA-Small	C4.5/GA-Large
Connect	72.60 ± 0.5	72.05 ± 0.5	76.19 ± 0.6 +	74.40 ± 0.6 +	**78.05 ± 0.5 +**	76.87 ± 0.0 +	76.95 ± 0.1 +
Adult	78.62 ± 0.5	77.00 ± 0.5 -	76.06 ± 0.5 –	78.90 ± 0.3	**80.94 ± 0.5 +**	80.62 ± 0.0 +	80.04 ± 0.1 +
Crx	91.79 ± 2.1	**92.45 ± 1.9**	90.78 ± 1.2	78.04 ± 1.2 -	90.61 ± 1.1	90.89 ± 1.3	91.66 ± 1.8
Hepatitis	80.78± 13.3	77.50 ± 11.3	82.36 ± 18.7	81.28 ± 11.2	88.91 ± 8.8	94.40 ± 6.2	**95.05 ± 7.2**
House	93.62 ± 3.2	93.50 ± 1.7	89.16 ± 8.0	**97.63 ± 1.6**	97.45 ± 1.7	96.80 ± 1.7	97.65 ± 2.0
Segmentat.	**96.86 ± 1.1**	96.30 ± 0.9	72.93 ± 5.5 –	72.42 ± 4.6 -	78.42 ± 1.2 -	79.00 ± 1.0 –	78.68 ± 1.1 –
Wave	75.78 ± 1.9	75.4 ± 2.1	64.93 ± 3.9 –	66.74 ± 4.6 -	83.24 ± 1.9 +	79.86 ± 4.2	**83.95 ± 3.0 +**
Splice	65.68 ± 1.3	66.54 ± 1.3	61.51 ± 6.6	60.26 ± 5.0	67.49 ± 6.5	67.04 ± 4.2	**70.70 ± 6.3**
Covertype	**71.61 ± 1.9**	70.34 ± 1.9	68.64 ± 14.8	65.40 ± 3.1 -	67.34 ± 16.8	69.43 ± 15.9	68.71 ± 1.3

Letter	86.40 ± 1.1	86.30 ± 1.1	82.77 ± 1.0 –	75.80 ± 0.3 -	**89.24 ± 1.1** +	81.15 ± 0.0 –	79.24 ± 0.2 –
Nursery	95.40 ± 1.2	96.40 ± 0.9	**97.23 ± 1.0**	82.13 ± 4.4 -	97.13 ± 0.8	96.93 ± 0.6	96.77 ± 0.7
Pendigits	96.39 ± 0.2	96.36 ± 0.3	96.86 ± 0.4	89.01 ± 0.5 -	**97.91 ± 0.3** +	94.96 ± 1.0 –	95.72 ± 0.9
CD-1	60.71 ± 3.0	58.26 ± 2.8	63.82 ± 5.2	61.40 ± 0.3	63.28 ± 4.3	**64.53 ± 4.5**	63.43 ± 1.4
CD-2	65.55 ± 1.5	63.22 ± 1.6	72.52 ± 5.9	62.95 ± 5.1	72.71 ± 5.7	73.52 ± 5.0 +	**73.77 ± 2.5** +
CD-3	75.65 ± 2.4	71.64 ± 1.3 -	82.27 ± 1.3 +	69.24 ± 2.2 -	81.99 ± 2.2 +	83.16 ± 1.8 +	**84.15 ± 0.9** +
CD-4	92.97 ± 0.9	89.47 ± 0.8 -	92.58 ± 1.0	89.7 ± 2.9	92.60 ± 0.9	**93.14 ± 0.9**	92.72 ± 1.0
CD-5	82.7 ± 2.8	78.71 ± 1.9	83.01 ± 1.9	72.31 ± 2.2 -	83.15 ± 1.8	**84.38 ± 2.1**	83.36 ± 2.1
CD-6	57.78 ± 2.1	55.36 ± 2.3	60.68 ± 3.2	59.37 ± 3.7	60.69 ± 2.9	60.91 ± 2.9	**61.69 ± 1.6** +
CD-7	65.18 ± 1.0	60.68 ± 1.4 -	70.29 ± 2.4 +	65.90 ± 1.6	70.61 ± 2.4 +	**82.77 ± 2.0** +	71.27 ± 1.6 +
CD-8	75.57 ± 1.4	70.30 ± 1.7 -	81.03 ± 1.9 +	73.69 ± 1.9	81.35 ± 1.6 +	81.78 ± 2.0 +	**82.63 ± 1.9** +
CD-9	93.00 ± 0.5	89.67 ± 1.4 -	93.72 ± 1.2	87.49 ± 2.9 -	93.48 ± 1.3	87.33 ± 1.8 –	**93.80 ± 1.4**
CD-10	82.80 ± 1.7	78.45 ± 2.2 -	85.60 ± 1.4	73.89 ± 1.4 -	85.28 ± 1.3	86.76 ± 1.5 +	**86.88 ± 1.6** +

Table A.4: Accuracy rate (%) on the test set for *S* = 15

Data Set	Default C4.5	C4.5 without pruning	Double C4.5	GA-Large alone	C4.5/IB1	C4.5/GA-Small	C4.5/GA-Large
Connect	72.60 ± 0.5	72.05 ± 0.5	74.95 ± 0.6 +	74.40 ± 0.6 +	**77.91 ± 0.5** +	76.13 ± 0.0 +	76.01 ± 0.3 +
Adult	78.62 ± 0.5	77.00 ± 0.5 -	74.29 ± 0.5 -	78.90 ± 0.3	**80.37 ± 0.5** +	79.97 ± 0.0 +	79.32 ± 0.2 +
Crx	91.79 ± 2.1	**92.45 ± 1.9**	90.02 ± 0.8	78.04 ± 1.2 -	89.46 ± 1.1	88.94 ± 2.3	90.40 ± 2.4
Hepatitis	80.78± 13.3	77.50 ± 11.3	66.16 ± 19.1	81.28 ± 11.2	**85.21 ± 7.8**	79.36 ± 23.4	82.52 ± 7.0
House	93.62 ± 3.2	93.50 ± 1.7	88.53 ± 8.4	**97.63 ± 1.6**	95.68 ± 1.7	94.88 ± 2.4	95.91 ± 2.3
Segment.	**96.86 ± 1.1**	96.30 ± 0.9	73.82 ± 5.8 -	72.42 ± 4.6 -	76.86 ± 1.7 -	77.00 ± 1.7 -	77.11 ± 1.9 -
Wave	75.78 ± 1.9	75.4 ± 2.1	65.53 ± 4.0 -	66.74 ± 4.6 -	82.43 ± 1.7 +	76.39 ± 5.0	**82.65 ± 3.7** +
Splice	65.68 ± 1.3	66.54 ± 1.3	64.35 ± 4.7	60.26 ± 5.0	66.68 ± 5.5	66.53 ± 4.9	**70.62 ± 5.5**
Covertype	71.61 ± 1.9	70.34 ± 1.9	**68.87 ± 15.1**	65.40 ± 3.1 -	64.32 ± 17.1	68.51 ± 16.3	66.02 ± 1.3 –
Letter	86.40 ± 1.1	86.30 ± 1.1	81.35 ± 1.0 –	75.80 ± 0.3 -	**89.30 ± 1.1** +	80.04 ± 0.0 –	76.38 ± 0.6 –
Nursery	95.40 ± 1.2	96.40 ± 0.9	**97.66 ± 0.8** +	82.13 ± 4.4 -	97.26 ± 0.5 +	97.34 ± 1.2	96.64 ± 0.7
Pendigits	96.39 ± 0.2	96.36 ± 0.3	96.86 ± 0.4	89.01 ± 0.5 -	**97.98 ± 0.3** +	95.71 ± 1.5	95.01 ± 1.2
CD-1	60.71 ± 3.0	58.26 ± 2.8	63.34 ± 4.9	61.40 ± 0.3	63.36 ± 4.0	63.68 ± 4.4	**63.92 ± 1.2**
CD-2	65.55 ± 1.5	63.22 ± 1.6	72.99 ± 4.8 +	62.95 ± 5.1	73.29 ± 4.8 +	74.36 ± 3.9 +	**74.75 ± 2.1** +
CD-3	75.65 ± 2.4	71.64 ± 1.3 -	81.92 ± 2.7 +	69.24 ± 2.2 -	81.78 ± 2.3 +	83.00 ± 2.0 +	**83.06 ± 1.0** +
CD-4	92.97 ± 0.9	89.47 ± 0.8 -	92.75 ± 1.4	89.7 ± 2.9	92.99 ± 1.3	93.28 ± 1.2	**93.48 ± 1.3**
CD-5	82.7 ± 2.8	78.71 ± 1.9	82.52 ± 2.0	72.31 ± 2.2 -	81.21 ± 1.8	82.61 ± 2.3	**82.81 ± 2.3**
CD-6	57.78 ± 2.1	55.36 ± 2.3	61.51 ± 3.1	59.37 ± 3.7	60.44 ± 2.5	61.78 ± 3.0	**62.07 ± 1.6** +
CD-7	65.18 ± 1.0	60.68 ± 1.4 -	70.11 ± 2.6 +	65.90 ± 1.6	70.11 ± 2.6 +	**72.09 ± 3.1** +	70.44 ± 2.0 +
CD-8	75.57 ± 1.4	70.30 ± 1.7 -	80.88 ± 1.3 +	73.69 ± 1.9	81.43 ± 1.2+	**83.20 ± 1.7** +	81.79 ± 2.2 +
CD-9	93.00 ± 0.5	89.67 ± 1.4 -	93.60 ± 0.5	87.49 ± 2.9 -	93.58 ± 1.1	87.12 ± 1.6 –	**93.67 ± 1.3**
CD-10	82.80 ± 1.7	78.45 ± 2.2 -	85.59 ± 0.5 +	73.89 ± 1.4 -	84.92 ± 1.6	**86.71 ± 1.8** +	85.70 ± 2,0

Figure A.1 shows the final set of rules derived from the meta-learning experiments described in Section 4. This rule set consists of two parts. The first part contains the part of the decision tree built by C4.5 specifying large disjuncts. This includes all the leaf nodes covering more than 5 meta-examples (since *S* was set to 5) and the tree branches necessary to define each of those large disjuncts. Each leaf node is followed by an annotation of the form (x / y), where x is the number of meta-examples belonging to that node and y is the number of misclassified meta-examples belonging to that node. When y = 0 the value of y is omitted from the notation, which is simplified to the form (x). This is the standard C4.5 annotation for leaf nodes.

The second part of the rule set contains the rules discovered by the GA-Large component of the hybrid C4.5/GA-Large algorithm, mining the small-disjunct meta-examples. Each rule is followed by the same annotation (x / y) used in the leaf nodes of the tree discovered by C4.5, for the sake of uniformity.

******large disjuncts in the decision tree built by C4.5***********

```
Num-SD > 444 : C4.5/IB1 (8)

Num-SD <= 141 :
|   Num-Exampl in {Medium,Small} :
|   |   C4.5-error > 4.6% :
|   |   |   SD-error <= 56.24% :
|   |   |   |   SD-perc > 1.06% : C4.5/GA-Large (17/2)

Num-SD <= 444 :
|   Num-Examp in {Medium,Small} :
|   |   Num-SD > 141 :
|   |   |   SD-perc <= 8.49% :
|   |   |   |   C4.5-error > 24.3% : Double-C4.5 (8/2)

Num-SD <= 353 :
|   Num-Examp in {Medium,Small} :
|   |   C4.5-error > 4.6% :
|   |   |   Num-SD > 141 :
|   |   |   |   SD-perc > 8.49% :
|   |   |   |   |   C4.5-error <= 39.3% :
|   |   |   |   |   |   SD-error > 47.93% : C4.5/GA-Large (7)

Num-SD <= 444 :
|   Num-Examp = VerySmall :
|   |   C4.5-error > 7.1% :
|   |   |   SD-error <= 51.52 : C4.5/GA-Large (6/2)


********** rules discovered by GA-Large from small disjuncts *********
Rule 1
IF Num-SD <= 7 AND Num-Examp = VerySmall AND SD-error <= 39.48%
THEN GA-Large  (4)

Rule 2
IF Num-SD < 146 AND Num-Examp in {Medium,Small} AND C4.5-error < 3.1%
THEN Default-C4.5 (4/1)

Rule 3
IF SD-error > 58.44% AND SD-size > 0.2882 AND Num-Att <= 8
THEN Default-C4.5 (1)

Rule 4
IF Num-SD < 118 AND Num-Examp = Large AND Num-classes < 8
THEN Double-C4.5 (4/1)

Rule 5
IF Num-Examp = VerySmall AND SD-error > 40%
THEN C4.5-without-pruning (3/1)

Rule 6
IF Num-SD <= 298 AND Num-Examp in {VerySmall,Medium} AND Num-Classes <= 13
THEN C4.5/GA-Large (24/16)

Rule 7
IF C4.5-error < 38% AND Num-Examp = Medium AND Num-Classes < 8
THEN C4.5/GA-Small (8/2)

Rule 8
IF Num-SD < 307 AND Num-Examp in {Large,Medium} AND Num-Classes >= 3
THEN C4.5/IB1 (11/6)

Rule 9
IF Num-SD >= 196 AND Num-Examp = Large AND Num-classes < 9
THEN C4.5/IB1 (5)
```

Figure A.1: Rule set produced by C4.5/GA-Large with *S* = 5 in the entire meta-data set