Induction of decision trees in numeric domains using set-valued attributes

Dimitrios Kalles, Athanasios Papagelis

Computer Technology Institute

PO Box 1122, 261 10, Patras, Greece

{kalles,papagel}@cti.gr

ABSTRACT

Conventional algorithms for decision tree induction use an attribute-value representation scheme for instances. This paper explores the empirical consequences of using set-valued attributes. This simple representational extension is shown to yield significant gains in speed and accuracy. To do so, the paper also describes an intuitive and practical version of pre-pruning. This method is shown to yield considerably better accuracy results when used as a pre-processor for numeric data. It is also shown to improve the accuracy for the second best classification option, which has valuable ramifications for post-processing.

INTRODUCTION

Describing instances using attribute-value pairs is a widely used practice in the machine learning and pattern recognition literature. In this paper we aim to explore the extent to which using such a representation for decision tree learning is as good as one would hope for. Specifically, we will make a departure from the attribute-single-value convention and deal with set-valued attributes.

We will use set-valued attributes to enhance our ability to effectively pre-process numeric variables (mainly). We will also show how these attributes can be used to enhance a potential post-processing scheme: by reporting the second best choice during classification (with increased confidence), one can evaluate alternatives in a more sophisticated fashion. In doing so, we face all guises of decision tree learning problems. Splitting criteria have to be re-defined. The set-valued attribute approach will put again pre-pruning into the picture, as a viable and reasonable pruning strategy. The classification task will also change. First, however, we provide an explanation of why we need set-valued attributes, and how to obtain them.

When using set-valued attributes during splitting, an instance may be instructed to follow both paths of a (binary) decision tree, thus ending up in more than one leaf. Such instance replication across tree branches allows splitting decisions to be based on larger instance samples. The idea is, then, that classification accuracy should increase. This increase is due to more educated splits and to instances following more than one path during testing too, thus delivering a combined decision on class membership.

The reader may be questioning the value of this recipe for instance proliferation, across and along tree branches. Truly, this might be a problem. At some nodes, it may be that attributes are not informative enough for splitting and that the overlap in instances between children nodes may be excessive. Defining what may be considered excessive has been a line of research in this work and it has turned out that very simple pre-pruning criteria suffice to contain this excessiveness.

As probabilistic classification has inspired this research, we would like to give credit, for some of the ideas that come up in this paper, to the work carried out by Quinlan (1987). However, we emphasize that the proposed approach does not relate to splitting using feature sets, as employed by C4.5 (Quinlan, 1993).

An earlier attempt to capitalize on these ideas appeared in (Kalles, 1994). Probably due to the immaturity of those results, the potential has not been fully explored. The work focuses on exploiting multiple classifications paths for a given testing instance, where these descriptions are all generated by the same decision tree. We do feel close to Quinlan's approach, at least in the start; we then depart to study what we consider more interesting views of the problem.

The rest of this paper is organized in four sections. The next section elaborates on the intuition behind the set-valued approach and how it leads to the heuristics employed. The actual modifications to the standard decision tree algorithms come next, including descriptions of splitting, pruning and classifying. We then demonstrate via an extended experimental session that the proposed heuristics indeed work and identify potential pitfalls. Finally, we put all the details together and discuss lines of research that have been deemed worthy of following.

AN OVERVIEW OF THE PROBLEM

Symbolic attributes are usually described by a single value ("Ford is the make of this car"). That a symbolic attribute may obtain an unequivocally specified value is a result of the domain itself (car makes are a finite number). On the other hand numeric values are drawn from a continuum of values, where errors or variation may turn up in different guises of the same underlying phenomenon: noise. Note that this can hold of seemingly symbolic values too; colors correspond to a de facto discretization of frequencies. It would be plausible that for some discretization, for a learning problem, we would like to be able to make a distinction between values such as green, not-so-green, etc. This explains why we feel that discretizing numeric values is a practice that easily fits the set-valued attributes context.

A casual browsing of the data sets in the Machine Learning repository (Blake et al., 1998) shows that there are no data sets where the same concept could apply in a purely symbolic domain. It comes as no surprise that such experiments are limited and the only practically justifiable ones have appeared in a linguistic context (in robust or predictive parsing, for example, where a word can be categorized as a verb or a noun).

One of the typical characteristics of decision trees is that their divide-and-conquer approach quickly trims down the availability of large instance chunks near the tree fringe. The splitting process calculates some level of (some type of) information gain and splits the tree accordingly. In doing so, it forces instances that are near a splitting threshold to follow one path. The other one, even if missed by some small $\Delta\varepsilon$ is a loser. This amounts to a reduced sample in the losing branch. By treating numeric attributes as set-valued ones we artificially enlarge the learning population at critical nodes. This can be done with a discretization step, before any splitting starts. It also means that threshold values do not have to be re-calculated using the actual values but the substituted discrete ones (actually, set of values, where applicable).

During testing, the same rule applies and test instances may end up in more than one leaf. Based on the assumption that such instance replication will only occur where there is doubt about the existence of a single value for an attribute, it follows that one can determine class membership of an instance by considering all leaves it has reached. This delivers a more educated classification as it is based on a larger sample and, conceptually, resembles an ensemble of experts. For this work, we have used a simple majority scheme.

LEARNING WITH SET VALUED ATTRIBUTES

To obtain set-valued attributes we have to discretize raw data. The discretization step produces instances that have (integer) set-valued attributes. Our algorithm uses these normalized instances to build the tree.

Every attribute's values are mapped to integers. Instances sharing an attribute value are said to belong to the same bucket, which is characterized by that integer value. Each attribute has as many buckets as distinct values.

For all continuous (numeric) attributes we split the continuum of values into a small number of non-overlapping intervals with each interval assigned to a bucket (one-to-one correspondence). An instance's value (a point) for a specific attribute may then belong to more that one of these buckets. Buckets may be merged when values so allow. Missing values are directed by default to the first bucket for the corresponding attribute (admittedly, an ad hoc approach which does not help the proposed modification and which will be dropped in subsequent work; however, very straightforward to implement).

We use the classic information gain (Quinlan, 1986) metric to select which attribute value will be the test on a specific node. An attribute's value that has been used is excluded from being used again in any subtree. Every instance follows at least one path from the root of the tree to some leaf. An instance can follow more than one branch of a node when the attribute being examined at that node has two values that direct it to different branches. Thus, the final number of instances on leaves may be larger than the starting number of instances.

An interesting side effect of having instances following both branches of a tree is that a child node can have exactly the same instances with its father. Although this is not necessarily a disadvantage, a repeating pattern of this behavior along a path can cause a serious overhead due to the size of the resulting tree.

These considerations lead to an extremely simple pre-pruning technique. We prune the tree at a node (we make that node a leaf) when that node shares the same instance set with some predecessors. Quantifying the some term is an ad hoc policy, captured by the notion of the pruning level (and thus allowing flexibility). So, for example, with a pruning level of 1 we will prune the tree at a node that shares the same instance set with its father but not with its grandfather. We expect that any information (lost) is likely contained in some of the surviving (replicated) instance sets.

Using the proposed method may result in testing instances that assign their values to more than one bucket. Thus, the classification stage requires an instance to be able to follow more than one branch of a node ending up, maybe, in more than one leaf. Classification is then straightforward by averaging the instance classes available at all the leaves reached by an instance.

An algorithm that uses the χ^2 metric, Chimerge (Kerber, 1992), was used to discretize continuous attributes. Chimerge employs a χ^2-related threshold to find the best possible points to split a continuum of values.

The value for χ^2-threshold is determined by selecting a desired significance level and then using a table to obtain the corresponding χ^2 value. For example, when there are 3 classes (2 degrees of freedom) the χ^2 value at the .90 percentile level is 4.6. The meaning of χ^2-threshold is that among cases where the class and attribute are independent there is a 90% probability that the computed χ^2 value will be less than 4.6; χ^2 values in excess of this threshold imply that the attribute and the class are not independent. Choosing higher values for χ^2-threshold causes the merging process to continue longer, resulting in fewer and larger intervals (buckets). The user can override χ^2-threshold by setting a max-buckets parameter, thus specifying an upper limit on the number of intervals to create.

During this research we extended the use of the χ^2 metric to create left and right margins extending from a bucket's boundaries. Every attribute's value belonging to a specific bucket but also belonging to the left (right) margin of that bucket, was also considered to belong to the previous (next) bucket respectively.

To define a margin's length -for example a right margin- we start by constructing a test bucket, which initially has only the last value of the current bucket and test it with the next bucket as a whole using the χ^2 metric. While the result doesn't exceed χ^2-threshold, we extend the right margin to the left, thus enlarging the test bucket. We know (from the initial bucket construction) that the right margin will increase finitely and within the bucket's size.

For example, suppose we have an attribute named *whale-size*, which represents the length of, say, 50 whales. To create buckets we sort the values of that attribute in ascending order and we use Chimerge to split the continuum of values to buckets (see Figure 1).

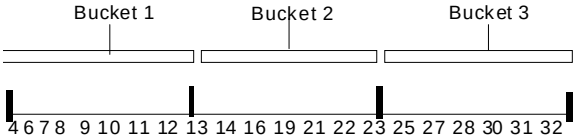


Figure 1: Splitting the continuum of values to buckets.

We then move to find the margins (see Figure 2). The first bucket has only a right margin and the last has only a left margin. Consider bucket 2 on the figure below; its boundary values are 13 and 23 and its left and right margin are at values 16 and 22 correspondingly.

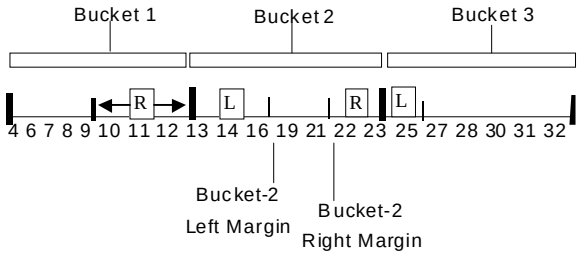


Figure 2: Getting margins for every bucket.

To obtain the right margin we start by testing value 23 with bucket 3 as a whole using the χ^2 metric. Suppose that the first test returns a value lower than χ^2-threshold; we set 23 as the right margin of bucket 2 and combine 23 with 22 (the next-to-the-left value of bucket 2) to a test bucket. This test bucket is tested again with bucket 3 using the χ^2 statistic. Suppose that, once more, the returned value does not exceed χ^2-threshold; we extend bucket's 2 right margin to 22. If the next attempt to extend the margin fails, because χ^2-threshold is exceeded, from then on, when an instance has the value 22 or 23 for its *whale-size* attribute, this value is mapped to both buckets 2 and 3.

EXPERIMENTAL VALIDATION

Experimentation consisted by using several databases from the Machine Learning Repository (Blake et al., 1998). We compared the performance of the proposed approach to the results obtained by experimenting with ITI (Utgoff, 1997) as well. ITI is a decision tree learning program that is freely available for research purposes. All databases were chosen to have continuous valued attributes to demonstrate the benefits of our approach. Testing took place on a Sun SparcStation/10 with 64 MB of RAM under SunOS 5.5.1.

The main goal during this research was to deliver more accurate results than existing algorithms. A parallel goal was to build a time-efficient algorithm.

As the algorithm depends on a particular configuration of χ^2-thresholds, maximum number of buckets and pruning level, we conducted our testing for various combinations of those parameters. Specifically, every database was tested using χ^2-threshold values of 0.90, 0.95, 0.975 and 0.99, with each of those tested for 5, 10, 15, and 20 maximum number of buckets and with a pruning level of 1, 2, 3, and 4, thus resulting in 64 different tests for every database. These tests would be then compared to a single run of ITI. Note that each such test is the outcome of a 10-fold cross-validation process. The databases used are shown in Table 1.

Database	Characteristics	Classes
Abalone	4177 instances, 8 attributes (one nominal), no missing values	3
Adult	48842 instances, 14 attributes (8 nominal), missing values exist	2
Crx	690 instances, 15 attributes (6 nominal), no missing values	2
Ecoli	336 instances, 8 attributes (one nominal), no missing values	8
Glass	214 instances, 9 attributes (all numeric), no missing values	6
Pima	768 instances, 8 attributes (all numeric), no missing values	2
Wine	178 instances, 13 attributes (all numeric), no missing values	3
Yeast	1484 instances, 8 attributes (one nominal), no missing values	10

Table 1: Databases for the experimental session.

Three metrics were recorded: accuracy, speed and size. Accuracy is the percentage of the correctly classified instances, speed is the time that the algorithm spends to create the tree, and size is the number of leaves of the resulting decision tree.

As stated earlier in this paper, one of the goals was to demonstrate that the proposed modifications improve not only accuracy in the conventional sense, but also our ability to generate good second- or third-place alternatives, for use in post-processors. We term this extension Coverage: reporting for a coverage of 2 means that we report the accuracy when we allow the real class to be among the best 2 returned by the classifier.

A notable characteristic of ITI's behavior when it has to be modified to allow for coverage results is that it implements a virtual pruning scheme. Under this scheme, a tree is always maintained in a dual form: the non-pruned version implements an exhaustive class separation policy whereas the pruned version allows classes to be mixed at a given leaf. In this work, coverage results are only reported for the pruned version.

Note that coverage *k* results for data sets of *k* classes is necessarily 100% (the *k*-th best class is certainly among the best *k* options). Such trivial results are not reported.

We first present the accuracy results (see Appendix A). The rather unconventional numbering on the x-axis is simply an indexing scheme; we felt that a graphical representation of the relative superiority of the proposed approach would be more emphatic. Two lines (one straight and one dotted) are used to demonstrate the results of ITI when pruning was turned on and off accordingly.

The experiments are sorted starting from a 0.90 χ^2-threshold, with 5 buckets (maximum) and a pruning level of 1. The first four results are for pruning levels 1 - 4. The number of buckets is then increased by 5 and the pruning level goes again from 1 - 4. The same procedure continues until the number of buckets reaches 20 where we change the χ^2-threshold to 0.95 and start again. The whole procedure is repeated for a χ^2-threshold of 0.975 and 0.99.

Coverage results are shown in Appendix B, where applicable.

As for speed, Appendix C demonstrates how the proposed approach compares to ITI in inducing the decision tree (the reported results for our approach are the splitting time and the sum of splitting and discretization time).

Regarding size, (see Appendix D) it is clear that the set-valued approach produces more eloquent trees.

Accuracy and coverage both seem to be benefited. It is interesting to note that under-performance (regardless of whether this is observed against ITI with or without pruning) is usually associated with lower χ^2-threshold values, or lower max-buckets values. Finding a specific combination of these parameters under which our approach does best for all circumstances is a difficult task and, quite surely, theoretically intractable. It rather seems that for every database depending on its idiosyncrasy different combinations are suited. However, coverage results are almost throughout consistent. It can be also be safely argued that unless χ^2-threshold is set to low values, thus injecting more "anarchy" than would be normally accommodated, and the max-buckets value is set too low, thus enforcing unwarranted value interval merges, we can expect accuracy to rise compared to ITI. The reported results clearly show that χ^2-threshold values above 0.95 and max-buckets values of about 10-20 suffice to universally ensure that the proposed modifications consistently and comfortably outperform ITI (with the exception of the notoriously low performance abalone and glass databases).

Accuracy is also dependent on the pruning level, but not with the same pattern throughout. If we focus on experiments on the right-hand of the charts (where χ^2-threshold and max-buckets values are reasonable), we observe that accuracy displays a data-set specific pattern of correlation with the pruning level, yet this display is

different across domains. We consider this to be a clear indication that pre-pruning is not a solution per se but should be enhanced with a post-pruning step to enhance the predictability of the outcomes. As stated below, post-pruning is an important item on our research agenda.

The results are quite interesting, as far as speed is concerned. While experimenting with most databases, we were confident that the discretization step would be of minor overhead. By experimenting with the wine and adult databases, however, we were surprised to see that our approach spent most of the time discretizing numeric attributes. Although this lead to a thorough re-evaluation of the software code and a decision to re-haul it as soon as possible, it also stimulated us to provide the graphs as presented, with a clear view of the speed of the splitting phase. We can claim that splitting speed is comfortably improved yet discretization should be further looked into. Note that pre–pruning also helps keeping the tree at a reasonable (yet larger than usual) size with a small price in time.

The main drawback of the algorithm is the size of the tree that it constructs. Having instances replicate themselves can result in extremely large, difficult to handle trees. Two things can prevent this from happening: normalization reduces the size of the data that the algorithm manipulates and pre-pruning reduces the unnecessary instance proliferation along tree branches. We have seen that both approaches seem to work well, yet none is a solution per se.

DISCUSSION

We have proposed a modification to the classical decision tree induction approach in order to handle set-valued attributes with relatively straightforward conceptual extensions to the basic model. Our experiments have demonstrated that by employing a simple pre-processing stage, the proposed approach can handle more efficiently numeric attributes (for a start) and yet yield significant accuracy improvements.

As we speculated in the introductory sections, the applicability of the approach to numeric attributes seemed all too obvious. Although numeric attributes have values that span a continuum, it is not difficult to imagine that conceptual groupings of similar objects would also share neighboring values. In this sense, the discretization step simply represents the anyway present clusters of the instance space along particular axes. We view this as an example of everyday representation bias; we tend to favor descriptions that make it easier for us to group objects rather than allow fuzziness. This bias can be captured in a theoretical sense by the observation that entropy gain is maximized at class boundaries for numeric attributes (Fayyad and Irani, 1992).

When numeric attributes truly have values that make it difficult to identify class- related ranges, splitting will be "terminally" affected by the observed values and testing is affected too. Our approach serves to offer a second chance to near-misses and as the experiments show, near-misses do survive. The χ^2 approach to determine secondary bucket preference ensures that neighboring instances are allowed to be of value longer than the classical approach would allow. For the vast majority of cases this interaction has proved beneficial by a comfortable margin.

As far as the experiments are concerned, the χ^2-threshold value for merging buckets has proved important. The greater the confidence level required, the better the accuracy attained, in most cases. However, there have been cases, where increasing χ^2-threshold decreased the accuracy (compared to lower χ^2-threshold values). We attribute such a behavior to the fact that the corresponding data sets do demonstrate a non-easily-separable class property. In such cases, increasing χ^2-threshold, thus making buckets more rigid, imposes an artificial clustering which deteriorates accuracy. That our approach still over-performs ITI is a sign that the few instances which are directed to more than one bucket compensate for the non-flexibility of the “few” buckets available.

Scheduled improvements to the proposed approach for handling set-valued attributes span a broad spectrum. Priority is given to the classification scheme. We aim to examine the extent to which we can use a confidence status for each instance traveling through the tree (for training or testing) to obtain weighted averages for class assignment at leaves, instead of simply summing up counters.

It is interesting to note that using set-valued attributes naturally paves the way for deploying more effective post-processing schemes for various tasks. By following a few branches, one need not make a guess about the second best option based on only one node, as is usually the case. Our coverage results show that significant gains can be materialized.

A straightforward example of post-processing arises in a character recognition setting. Robust OCR customarily employs dictionary look-up techniques to search for potential word forming and thus guide the search. In such contexts, where ambiguity between classes is quite complex (for example, 1 –the digit- and l –the letter-), being able to pin-point the second best option accurately can result in significant post-processing time savings.

Another important line of research concerns the development of a suitable post-pruning strategy, as instance replication may complicate the mathematics involved in estimating error (for example, it is not at all obvious how error-complexity pruning (Breiman et al., 1984) might be readily adapted to set-valued attributes). Last, but not

least, we need to explore the extent to which instance replication does (or, hopefully, does not) seriously affect the efficiency of incremental decision tree algorithms.

We feel that the proposed approach opens up a range of possibilities for decision tree induction. By beefing it up with the appropriate mathematical framework we may be able to say that a little bit of carefully introduced fuzziness improves the learning process.

REFERENCES

(Blake et al., 1998) C. Blake, E. Keogh, and J Merz. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998).

(Breiman et al., 1984) L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA (1984).

(Fayyad and Irani, 1992) U.M. Fayyad, K.B. Irani. On the Handling of Continuous-Valued Attributes on Decision Tree Generation. Machine Learning, 8: 87-102 (1992).

(Kalles, 1994) D. Kalles. Decision trees and domain knowledge in pattern recognition. Phd Thesis, University of Manchester (1994).

(Kerber, 1992) R. Kerber. Chimerge:Discretization of numeric attributes. 10th National Conference on Artificial Intelligence, San Jose, CA. MIT Press. 123-128 (1992).

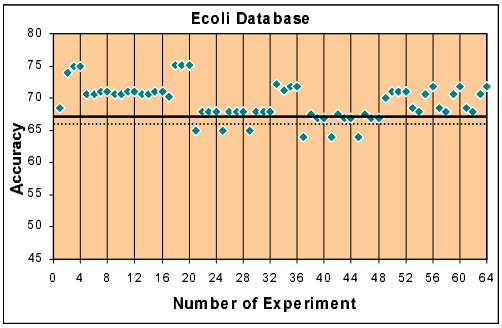
(Quinlan, 1986) J.R. Quinlan. Induction of Decision Trees, Machine Learning, 1:81-106 (1986).

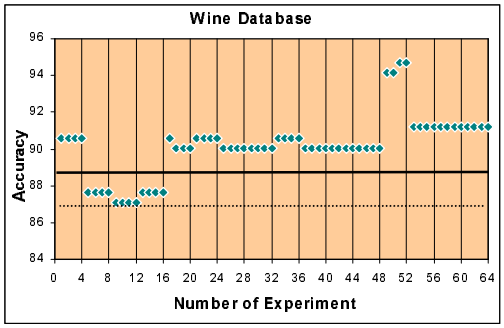
(Quinlan, 1987) J.R. Quinlan. Decision trees as probabilistic classifiers. In Proceedings of the 4th International Workshop on Machine Learning, pages 31-37, Irvine, CA, June 1987.

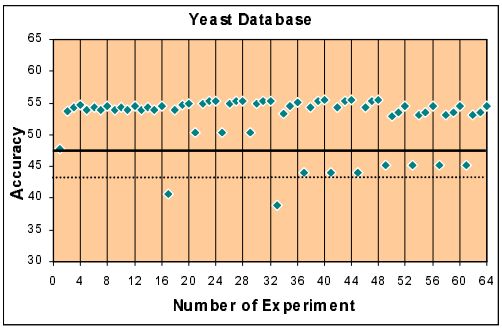
(Quinlan, 1993) J.R. Quinlan. C4.5 Programs for machine learning. San Mateo, CA, Morgan Kaufmann (1993).

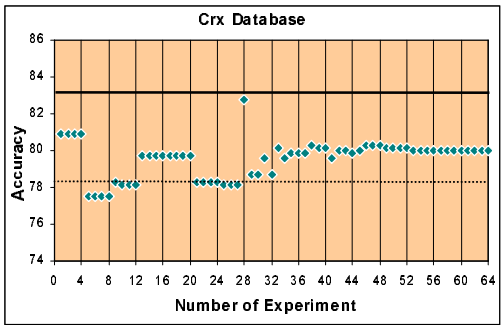
(Utgoff, 1997) P.E. Utgoff. Decision Tree Induction Based on Efficient Tree Restructuring, Machine Learning, 29:5-44 (1997).

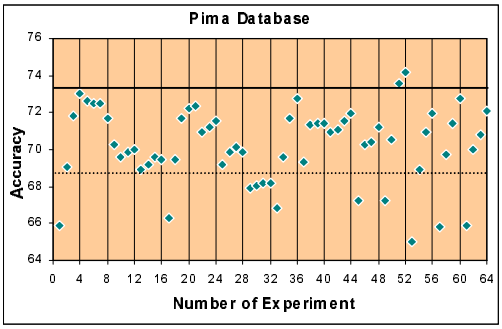
A APPENDIX ON ACCURACY RESULTS

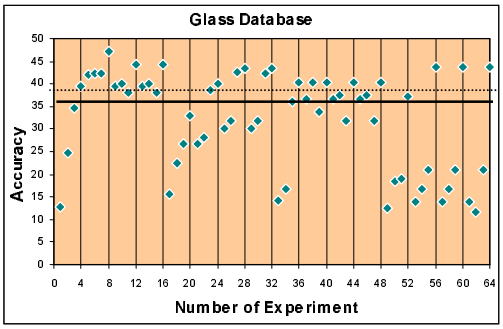


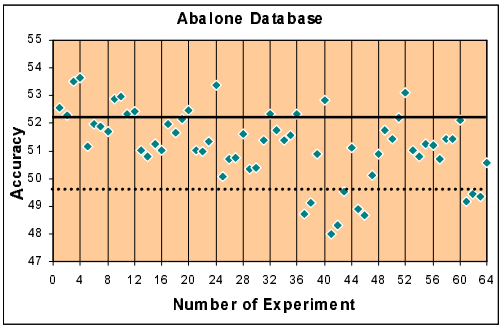


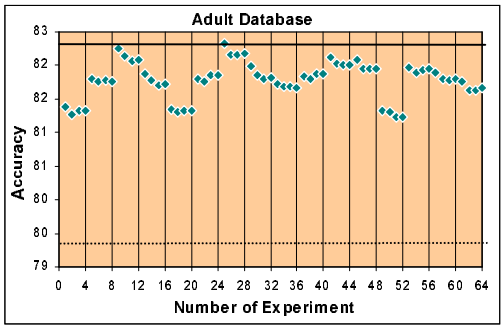




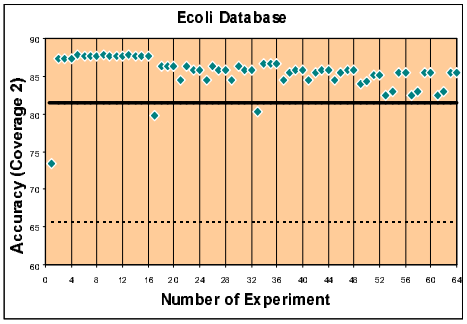


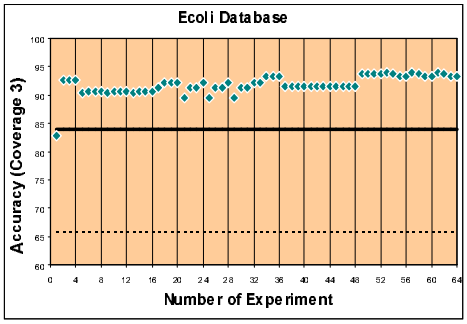


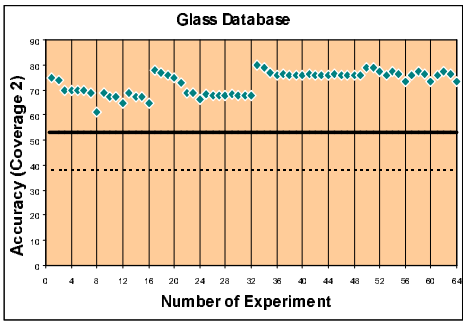


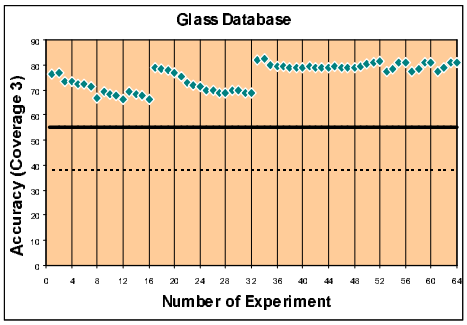


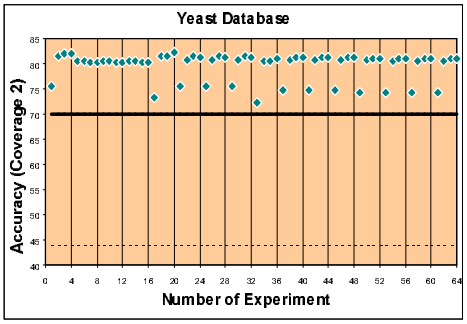
B APPENDIX ON ACCURACY RESULTS FOR COVERAGES 2 AND 3

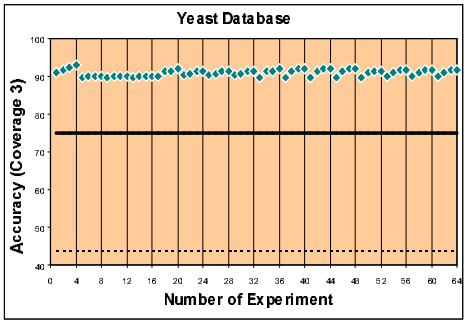


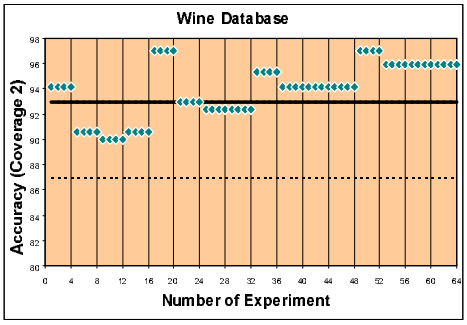


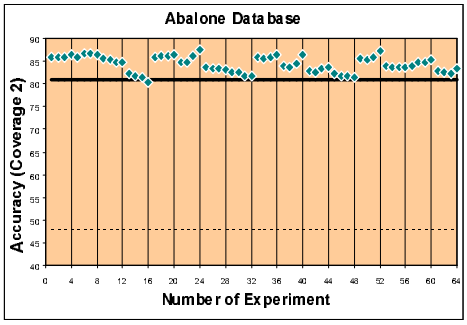




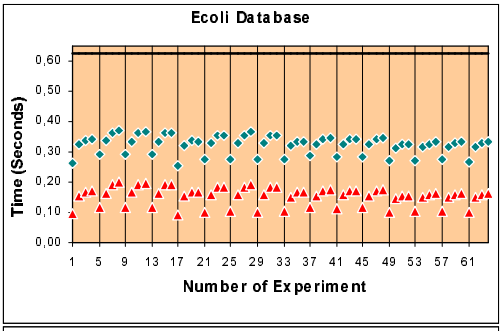


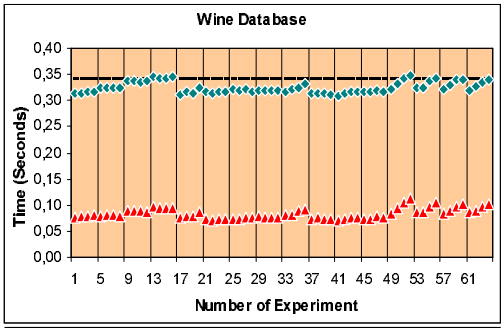


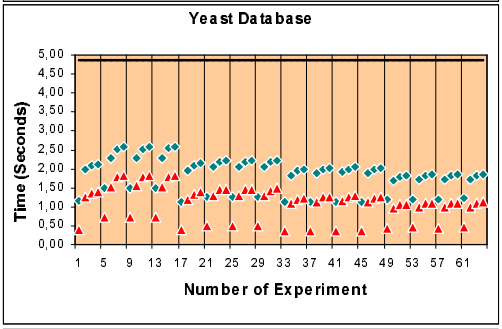


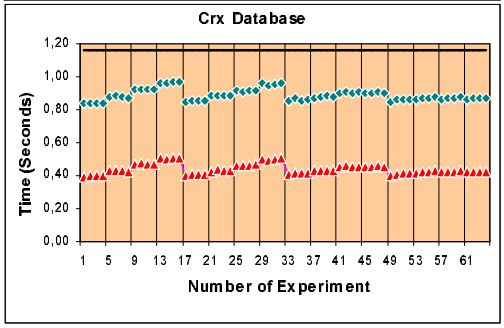


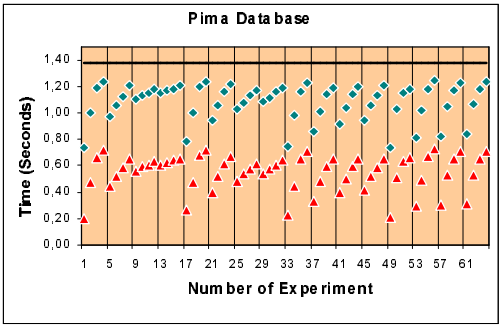
C APPENDIX ON SPEED RESULTS

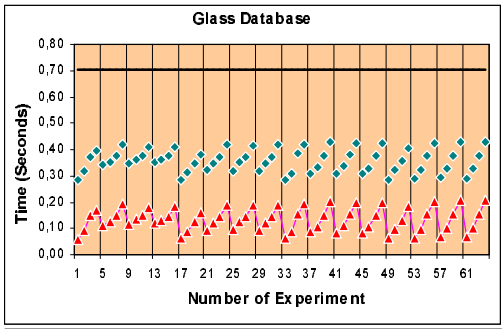


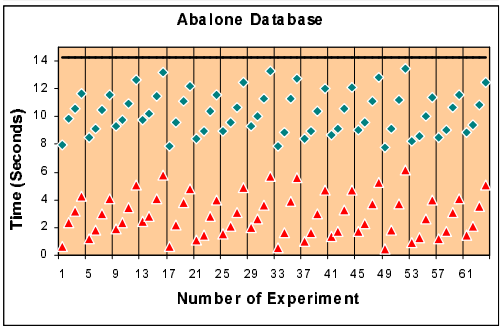


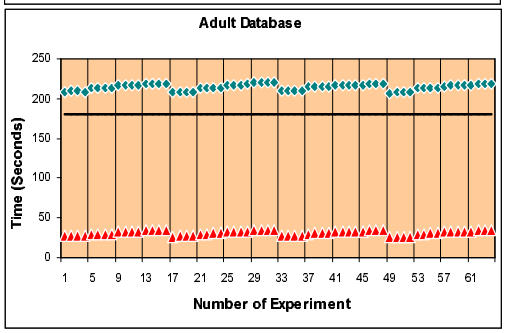












D APPENDIX ON SIZE RESULTS