Framework for a Generic Knowledge Discovery Toolkit

Pat Riddle, Roman Fresnedo, David Newman

October 31, 1994

Industrial and commercial firms accumulate vast quantities of data in the course of their day-to-day business. The primary use of this data is to monitor business processes: inventory, maintenance actions, and so on. However this data contains much valuable information that, if accessible, would enhance the understanding of, and aid in improving the performance of, the processes being monitored.

Traditional statistical procedures provide some insight into this data, but they are often misused in non-expert hands. With the rapidly increasing quantity of data, it is no longer cost effective for trained statisticians to analyze all the data. The number of variables and observations in these datasets is often very large, and the number of candidate statistical models that might be considered is too large to permit manual systematic exploration. In this type of situation, a Knowledge Discovery (KD) tool is the most effective way to explore the data.

1 Generic Knowledge Discovery Toolkit

Our current goal is to develop a generic toolkit for KD. In our previous work we have developed a suite of tools and shown their potential on actual industrial data; these tools must now be enhanced to make them into generic tools. This suite of tools can be thought of as a collection of exploratory data analysis techniques; it uses nonparametric, multivariate methods to identify a large number of candidate models in the form of rules. The algorithms identify rules where the independent variables have a high probability of affecting the dependent variable. The effectiveness of these algorithms has been demonstrated at Boeing in several datasets containing massive numbers of variables and observations. We have successfully explored datasets with over 100,000 observations and datasets with over 140 variables.

Currently, the tools must be extensively hand-tuned by a data analysis expert (e.g., machine learning expert, statistician, etc.) to work with each dataset from each new domain. Hand-tuning refers to the initial data engineering, setting algorithm parameters, and analysis of the results. Data engineering refers to the transformation of the data from its initial form into an appropriate form for data analysis. The data analyst's expertise is currently a big factor in the successful use of this technology. We are currently automating this hand-tuning process. The reason we refer to it as a toolkit, is that the set of generic tools will be too complicated for a novice user to combine correctly. We envision a data analysis expert crafting a domain specific tool from the tools in the toolkit. This domain specific tool will be appropriate for novice use.

John Tukey's advice to the statistics community is "...we are going to see more data-dredging in all fields. So we must learn to work effectively with its results."(Tukey, 1986) The focus of our work is very much in the spirit of this advice; the creation of an automated tool for novice use. This is in the same spirit as the work in statistical expert systems (i.e., expert systems to help a novice do statistics).

2 Knowledge Discovery Environment

The environment of our knowledge discovery work is slightly non-standard and worth a little elaboration. The results we produce (for instance rules predicting rejected parts), will not be used to create an expert system for determining whether a part should be rejected. Instead these rules will be examined by a human to learn more about their process and thereby achieve process improvement. Also our dependent variables typically have multiple causes, e.g. there is probably more then one responsible cause for rejected parts. We therefore typically expect multiple hypotheses, but there is no reason why these multiple hypothesis should partition the space (i.e., certain observations can be covered by multiple rules). These two factors were two of the main reasons why we ventured away from a standard decision tree approach to our current approach.

Two terms common in machine learning are accuracy, positive coverage, and coverage and they are used throughout this paper. The accuracy is the (empirical) conditional probability of the rule's postcondition being true given that the rule's precondition is true. The positive coverage is the (empirical) conditional probability of the rule's precondition being true given that the rule's postcondition is true. The combination of these two measures gives the strength of the implication in each of the two directions. The rule's coverage is the probability that the rule's precondition is true.

The example used throughout this paper shows results from an actual dataset from the Boeing company. The variable names and values have been disguised for proprietary reasons but all the results are from this actual dataset. There were 6384 observations and 141 variables, which were ordinal (i.e., some discrete and some continuous) and categorical. The base rate of the dependent variable is 23.5%. The base rate is the proportion of the "value of interest" in the dependent variable. We transform the independent variable Y into a binary variable, called the target, by partitioning the range of Y into two sets: a set containing the value of interest, Target=True, and its complement. The rule is made up of a precondition and a postcondition (i.e., the target). The independent variables used in the precondition of the rule are also called predictor variables.

3 Boeing's Knowledge Discovery Framework

There are three main extensions to existing technology which are necessary to achieve this generic knowledge discovery capability. These research directions are:

1. to develop a statistically valid process for selecting and ordering the "good" rules produced by the algorithm and expressing the uncertainty associated with competing rules,

2. to develop a summarization and visualization methodology that allows process owners and data analysis experts to examine how the rules compete with and complement each other, and

3. to develop a methodology for deriving data representations which best capture the objectives of the process owners, and at the same time maximize the effectiveness of the rule extraction algorithms used for KD.

Figure 1 shows the interrelations between these three foci and the rule extraction algorithm. These three aspects of applying KD tools have received little attention from academic investigators. But more importantly, these foci are the aspects of this technology which must be resolved (at least partially) if KD is to become really useful for commercial applications.

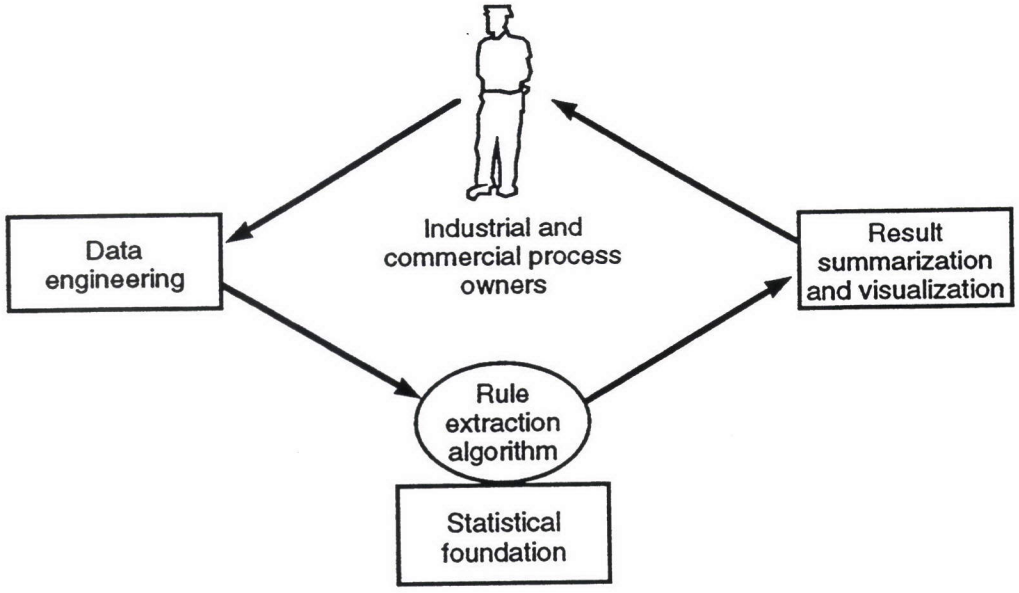

Figure 1: Boeing Knowledge Discovery Framework

W. Heisenberg states "It is probably true quite generally that in the history of human thinking the most fruitful developments frequently take place at those points where 2 different lines of thought meet." The fields of Artificial Intelligence (of which machine learning (ML) is a subfield) and Statistics have begun to cross-pollinate in the last few years. We intend to marry these two fields by using new and innovative algorithms and also providing a firm statistical foundation for them to rest upon.

The rule extraction algorithm on which we are currently focusing is the Brute/Beam-Brute algorithm. The dependent variable must be represented in categorical form, while the other variables can be either categorical or ordinal. Note that the dependent variable need only be categorical but is treated as binary once the value of interest is chosen. For instance if the dependent variable contains a value from the set duck, dog, cat and the value of interest is defined as "duck", the algorithm treats the variable as a "duck" "not duck" variable. The requirement that the dependent variable be categorical means that in cases where the dependent variable is ordinal it must be discretized. We have done this with datasets in the past by using quantiles.

Brute determines conjunctive rules which predict the target. It exhaustively explores all possible conjunctive rules over the independent variables up to the maximum depth, orders these rules using an evaluation function, and returns the top X rules where X is the size of the rule bucket (e.g., the top 200 rules ordered by the evaluation function). Brute was developed by Etzioni & Segal at University of Washington Computer Science Department in collaboration with Riddle

at Boeing, based on their experiences using the IND algorithm (Buntine and Caruana, 1991) on Boeing datasets. IND simulates several decision tree algorithms including CART and C4.5. The difficulty in using decision tree algorithms in these domains which led to the development of Brute is discussed in (Riddle *et al.*, 1994) which also contains a thorough description of the algorithm and experiments comparing it to CART and C4.5. The major focus in developing Brute was to develop an algorithm which would discover a few good rules which may only cover a small number of observations instead of focusing on covering the entire space of observations with a decision tree. This is based on the notion that we are dealing with domains which are fundamentally multiple causal domains.

In our use of Brute within Boeing, statistical measures (accuracy, positive coverage, and χ^2) derived by the algorithm during learning are an integral part of the resulting learned rule. This is a fundamental difference between our work and most of the historical work in the field of machine learning.

4 Statistical Foundation

Brute discovers collections of rules (models). We must determine how to select, rank, compare and validate these models. KD often generates very different rules which cover disjoint or nearly disjoint groups of observations in the dataset. Informally, we could say that rules can be competing, complementary or subsumed. Competing rules "explain" roughly similar sets of observations using different sets of predictor variables. Complementary rules "explain" roughly disjoint pieces of the space of observations. Subsumed rules cover nested subsets of the observation space, covering less and less points but being increasingly accurate. Widely varying complementary rules may indicate extreme heterogeneity in the dataset. The possibility of differing explanations for different parts of the dataset as complementary must be admitted.

Until we have a firmer probabilistic foundation and understanding of our searching algorithms, we will consider the rules discovered by Brute as descriptive statistics. Because of this, the only type of "bad rules" we are trying to prevent from being reported by our statistical methodology are those rules which are random rules. KD should not find rules when the target is statistically independent of the predictor variables. We can imagine a mechanism that assigns the dependent variable's values randomly (blindly) to a sample of predictors. If this is the case, then we should not find any rules, whatever the distribution of the predictors. Any mechanism other than randomness, like use of an unseen variable to assign the values of the dependent variable, may cause rules to be found, if this unseen variable is associated with seen ones.

	Target T	Target F
Precondition T	P_{11}	P_{12}
Precondition F	P_{21}	P_{22}

Table 1: 2x2 Table

Our current experimental methodology for insuring the statistical validity of our rules is as follows. We run Brute on all the data and use the χ^2 statistic as Brute's internal evaluation function to order the rules and keep the best ones in a very large bucket. The χ^2 statistic comes

from the 2x2 table generated by the target being treated as binary and the rule's precondition being true or false. This is shown in Table 1. Then we use a 1000-fold bootstrap with the same data to determine the the 20 percentile of the accuracy of each rule. If the base rate, is above this percentile, then the rule is removed from consideration. It would be natural, for base rates closer to 0, to demand the bootstrapped 20 percentile to be an integer multiple of the base rate. We choose accuracy because it is usually the measure that users care about the most. We used to use accuracy as the evaluation function internal to Brute, but we found it did not give good rule selections when the coverage of a rule was small. Any other simple statistics would be easily added to the bootstrap evaluator. The accuracy and positive coverage statistics given previously, can now be formally defined as

$$accuracy = P_{11}/(P_{11} + P_{12})$$

$$positive\ coverage = P_{11}/(P_{11} + P_{21})$$

The other popular resampling technique, cross-validation, has been tried, but the maze of rules generated by Brute multiplies. For each cross-validation fold Brute would learn a new set of rules. Strong rules tend to appear in all subsamples almost unchanged, but weaker ones sometimes disappear or are greatly modified. In how many subsamples must a rule appear to be accepted? How much variation of its ranking and χ^2 across subsamples would be acceptable? The task of managing all this variation seems difficult.

As we stated, we are using χ^2 to order the rules and the bootstrapping on accuracy to define the cut-off point. χ^2 does a good job of ordering the rules but cannot be used as a cut-off since the size of the dataset has such a large impact on the results. Normalizing χ^2, as in Pearson's contingency coefficient, $\sqrt{\chi^2/(samplesize + \chi^2)}$ would solve this last problem, but we would still have to work to find the right cut-off value. For 2x2 tables, χ^2 is known to be a bad discriminator of the hypothesis of independence vs. the hypothesis "chosen at random", even for sample sizes as large as 500 (Diaconis and Efron, 1985). Although all these objections to the χ^2 could be overcome, we are still experimenting with other measures of association because of the following problem: χ^2 rates highly those rules which are both very predictive of the target and very unpredictive of the target. But we are only interested in the predictive ones. Natural candidates are the cross product ratio and an a-posteriori odds ratio. The first one,

$$\alpha = \frac{p_{11}/p_{12}}{p_{21}/p_{22}}$$

, computed over the 2x2 table, has an intuitive interpretation: it is the ratio where the numerator equals the odds of the target being true given that the rule is true and the denominator equals the odds of the target being true given that the rule is false A large value of alpha will signal a strong rule. Another possibility is the entropy statistic as used by CN2 (Clark and Niblett, 1989).

We did a number of experiments with the combination of χ^2 and bootstrap. One question which arises is whether the ordering imposed by χ^2 is close enough to the ordering given by the bootstrapped accuracy, so that rules left outside the χ^2 bucket would not found to be "good" rules by the bootstrap evaluator. The bootstrap evaluator cannot be used as the internal evaluation function of Brute for computational reasons. The Brute process is very CPU-intensive and the evaluation function must be modest in terms of its computational requirements. A further advantage of the bootstrap, not exercised yet, is that it allows the construction of simultaneous confidence intervals

for all these rules (models) and their accuracies. We normally treat each rules as an isolated model, but one must not forget that they are highly redundant and statistically dependent.

The following experiment shows that if you make your bucket large enough the chi-square ordering will not put any good predictive rules lower than any bad predictive rules (which are of the right sign). So the combination of χ^2 with a large bucket and bootstrapping to decide cutoff works well. The horizontal line represents the base rate in this dataset.

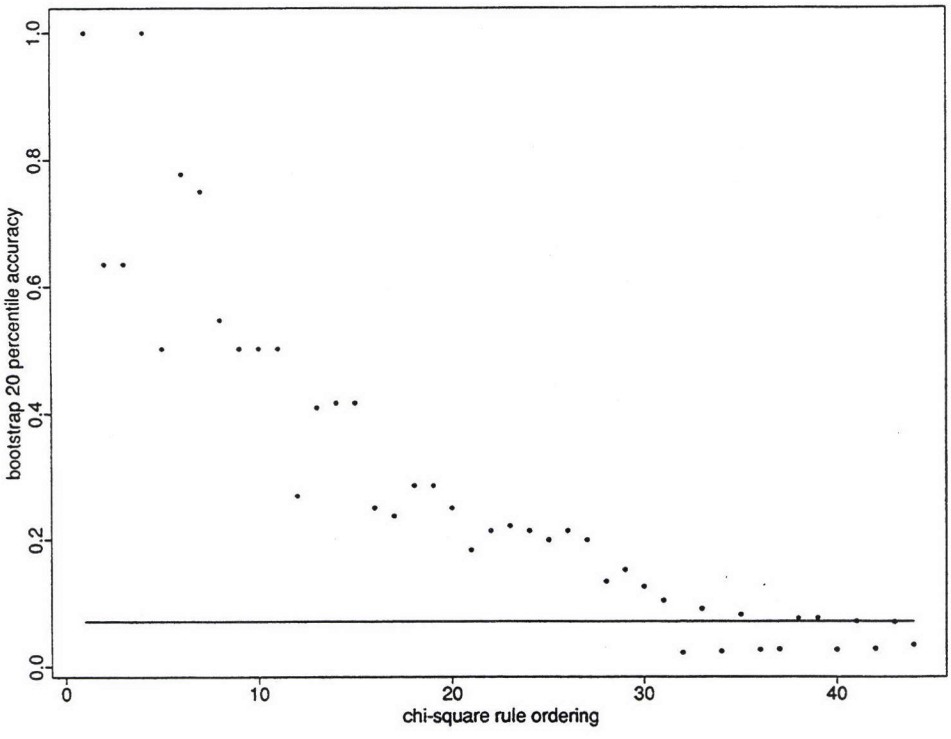

Figure 2: Experimental Results

We know that the users' temptation to make inferences form these models will be great. We are exploring the application of graphical models (Whittaker, 1990) to be able to assess how much these models are supported by the data.

5 Result Summarization & Visualization

Model summarization and visualization tools are needed because the aggressive search strategies for rule extraction in our current algorithms produce too many rules (models) for detailed human review. After statistical screening and validation, the user is left with many rules that are either "competing", "complementary" or "subsumed". The set of rules produced are therefore naturally grouped into families. We currently use statistical hierarchical clustering techniques to automate the family creation process.

The intention is to help the process owners choose from among those models which pass the statistical screen of model selection. It is important to present as many of the valid models as possible to the process owners, as it is a priori unclear which of the models might make the most sense to them. Competing models will allow the user to work with several "working hypothesis" until further analysis sheds more light on the subject; they also show correlations between predictor variables that could suggest further statistical modeling. One important notion to remember is that when we say two rules are competing (e.g., aliasing for each other), we are viewing this through the projection of the postcondition. This is very different than two variables which are directly aliasing for each other in all situations. We use the term "aliasing" to specify that two variables are associated or correlated with each other. Complementary models may suggest actions with different cost/payback ratios and thus allow the user to direct efforts where returns are higher or immediate.

5.1 Visualizing Multiple Rules

One of the major problem's with the Brute methodology is that we end up with too many rules for a person to process efficiently. This is especially a problem when the variables alias for each other. One approach we have taken to this is hierarchically clustering the rules. We produce a matrix that has the rules as rows and 2x the observations as columns. For each rule we specify which observations satisfy its precondition of the rule and which observations satisfy both its precondition and postcondition (hence the doubling of observations). Once this matrix is created we do a hierarchical clustering with a euclidean distance metric. This metric is

$$d^2(R1, R2) = \sum_{obs}(R1(obs) - R2(obs))^2 + \sum_{obs}((R1 \wedge Target)(obs) - (R2 \wedge Target)(obs))^2$$

The rules are considered close when their preconditions (R1 and R2) cover roughly the same points and roughly the same positives.

This allows us to see obvious sources of aliasing. An example of this is shown in Figure 3, only a portion of the hierarchical cluster tree for the 121 good rules produced by Brute is shown. Notice the rules on the branch marked X. These rules are clearly aliasing off the last conjunct. Our reason for presenting equivalent rules is that we are trying to derive rules for a person to understand the process. This is very different from producing a decision tree which will be used for prediction. Since a human is going to process this information and we do not know what will or won't make sense to them, we want to present all the possibilities.

Another type of visualization we have used is a subsumption matrix. Figure 4 presents the results of this analysis. It is based on the matrix which specifies for each rule which observations satisfy both the preconditions and postconditions. The system calculates which rules cover the same observations to some level of precision 80-90% and displays the list in the order of increasing coverage. It shows each rule's accuracy, positive coverage, and chi-squared.

5.2 Visualizing Single Rules

We have used visualization of single rules in two ways. We have developed specific visualizations which present accuracy, positive coverage, and χ^2 for a rule to show how it varies as one variable moves a ordinal boundary. This is shown in Figure 5.

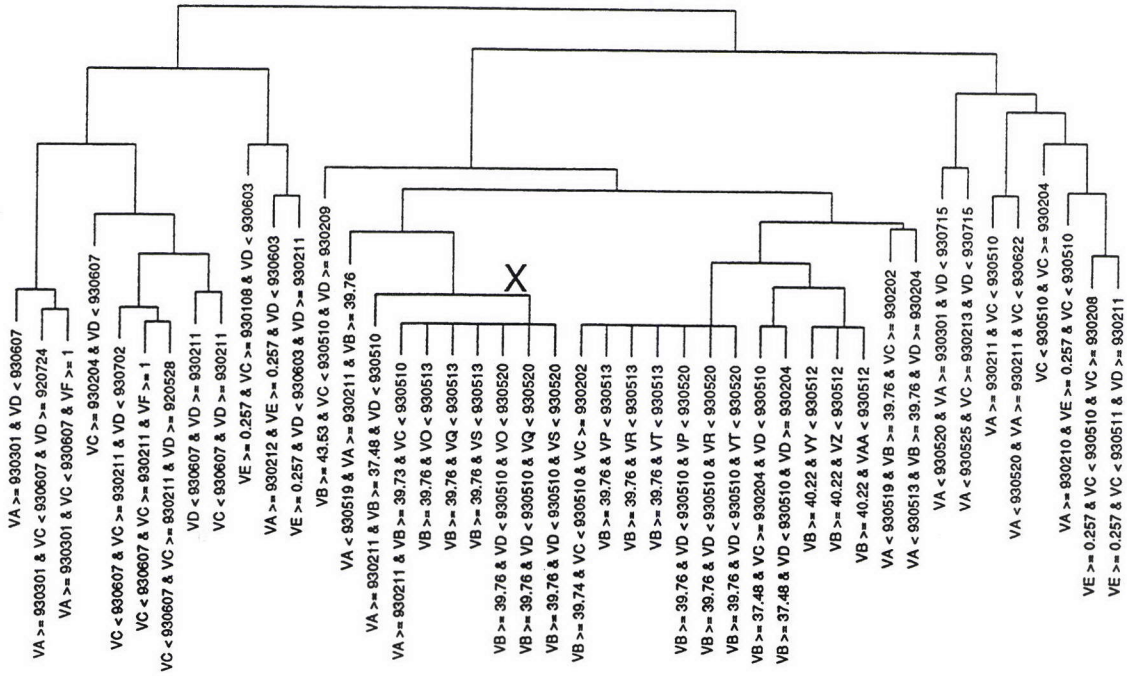

Figure 3: Rule Clustering

We have also used standard statistical displays which were suggested by specific rules. This is demonstrated in Figure 6 which was suggested by the following rules: IF airline = X $\wedge$ component = 1 THEN target = T and IF airline = X $\wedge$ component = 2 THEN target = T.

Further displays are currently under development. One interesting display shows a sequence of good rules with strictly increasing cumulative positive coverage. It is plausible, and has been our experience, that we can exhaust the set of excellent rules and yet leave a non-negligible proportion of positives unexplained. This last phenomena should be common in observational studies when some of the variables that explain positives are left unobserved.

6 Data Engineering

It is easy to see that each of the different forms in which the data is represented will allow different types of patterns to be found in the data. The data representation is the result of formulating the problem by selecting the variables and specifying one as the single predicted variable. The success of the entire KD process is dependent upon the form in which the data is presented to the tool. The first step is data cleaning where the data is made as error free and as consistent as possible. But even if clean data is presented in a bad form, no patterns may be found even if there exist patterns in the data. In principle there are many reformulations of the data (i.e., functions over the data) which could be computed.

We are creating a library of low level generic data transformations. A transformation is a

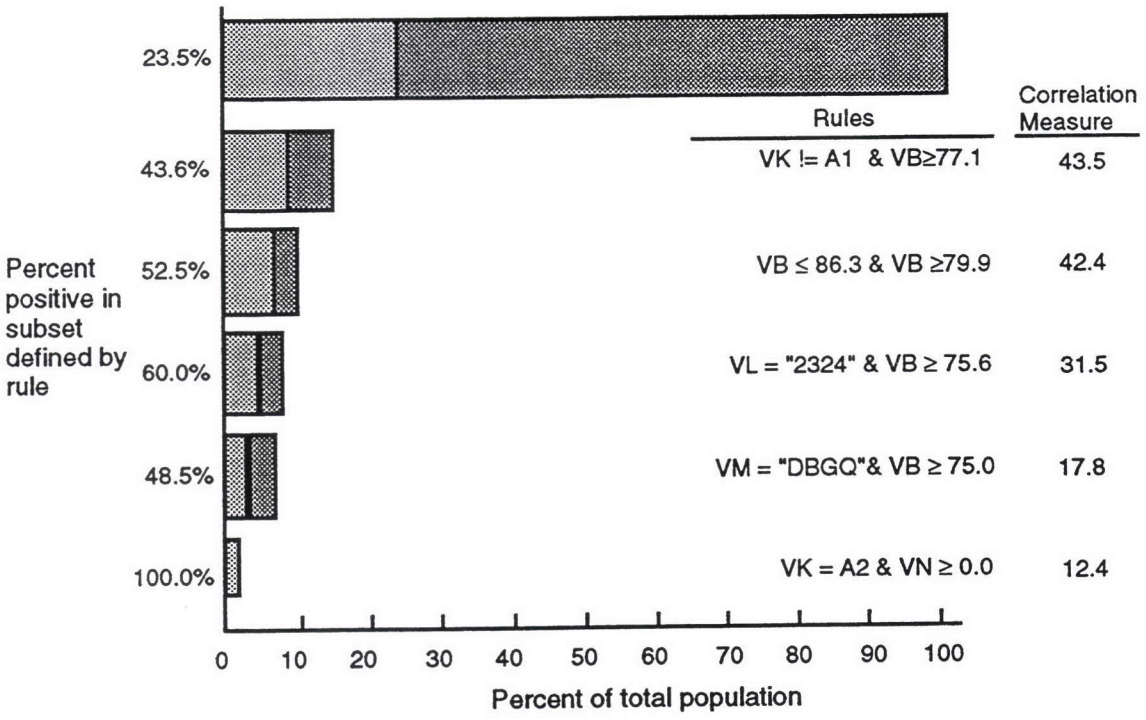

Figure 4: Subsumption Matrix

function which translates one variable or combination of variables into another variable. Examples of these generic data transformations are discretizing and interpolating. We have become very proficient at reformatting the data by iterating between changes to the data representation and rule extraction algorithm runs. We are capturing this knowledge to embed it in a semi-automated procedure, a reformulation advisor, which a process owner can use to reformat his own data, so that the tool will perform well in that domain.

It is important to determine what effect the use of a lot of data transformations will have on the algorithm's statistical validation measures. This problem is another side of the overfitting problem mentioned above. Bear in mind our original description of a toolkit. We are developing generic tools which can be assembled together by a data analysis expert for a particular domain. A novice user then uses the tools within that domain to do his analysis. In this methodology a data analysis expert will be available to assure that overfitting does not occur. Our current stance on this problem is that a transformation which is "natural" in the domain will not cause overfitting. Only artificial transformations are suspect.

For instance, if we take monthly wear information on an airplane (usually measured in flight time or number of cycles) and determine its "age" in flight hours or cycles and find correlations between this notion of age and our predictor, then we would claim that is a real correlation and not overfitting because airplane age measured in this way is a "natural" attribute in this domain. If we took monthly wear information and randomly took a mathematical function of it (square, sin, etc.), then the possibilities of overfitting need to be considered. In our domains we have found cases where there are hundreds of natural transformations. It is interesting to note that the Brute

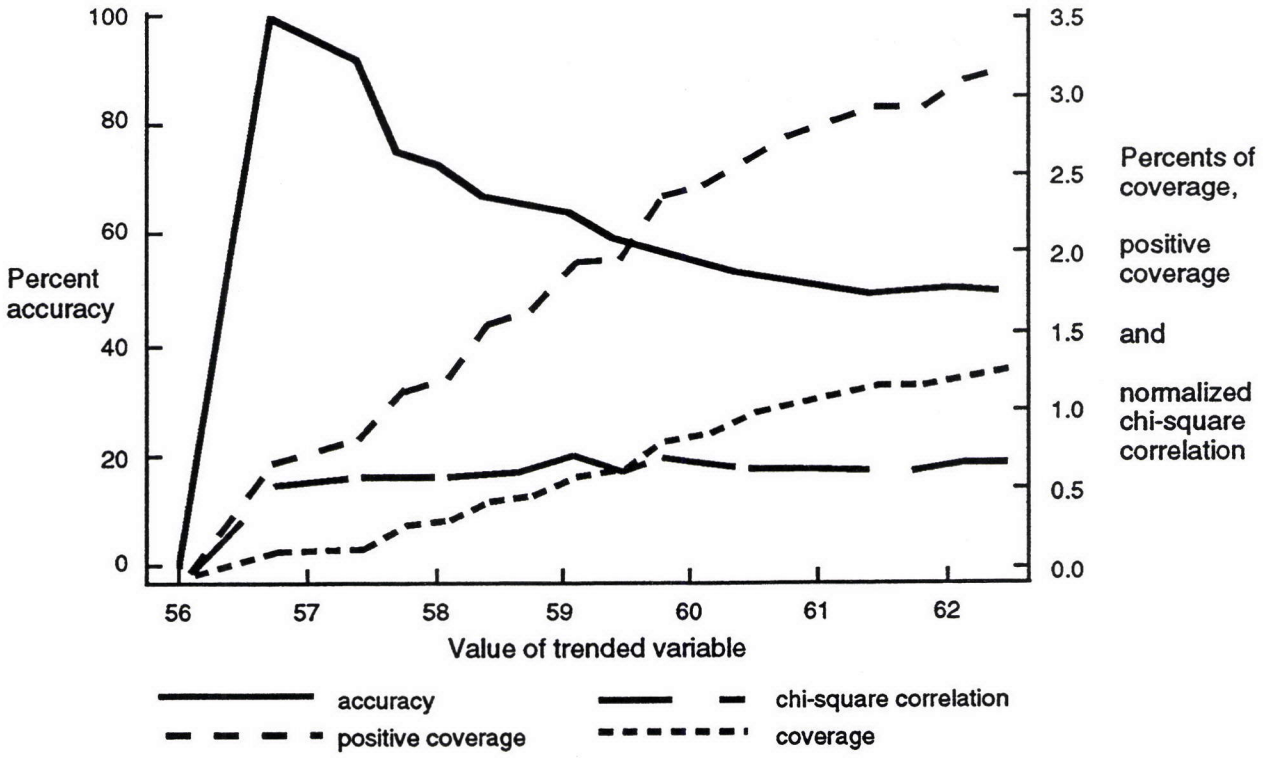

Figure 5: Trend Diagram

algorithm is invariant over monotone functions, so those need not be considered in any case.

One effect of variable-redefinition is Simpson's paradox (Whittaker, 1990) where the statistical measures of correlation can validate a model which is incorrect in the untransformed dataset. This paradox is caused by not making a natural distinction in the data, rather than making too many. So the use of data transformations to add more independent variables, will ease the Simpson's paradox problem.

7 Related Work

At the framework level there is little related work to report. Most places are working on the discover algorithms themselves (e.g., new Brutes) instead of total frameworks within which to place them. On the other hand each of the pieces of our framework has its own field of related work.

The Brute algorithm itself is most similar to ITRULE. The evaluation function they used is more tuned to finding high coverage rules instead of high predictive rules as is the Brute evaluation function. A comparison between Brute, ITRULE and other rule extraction systems can be seen in (Riddle *et al.*, 1994).

Our current visualization work is a small portion of the field of scientific visualization. We have not delved into the more advanced techniques in this field. Currently the more mundane aspects of this field have been sufficient for our framework.

Likewise the field of data engineering is vast in itself in the areas of representation reformulation

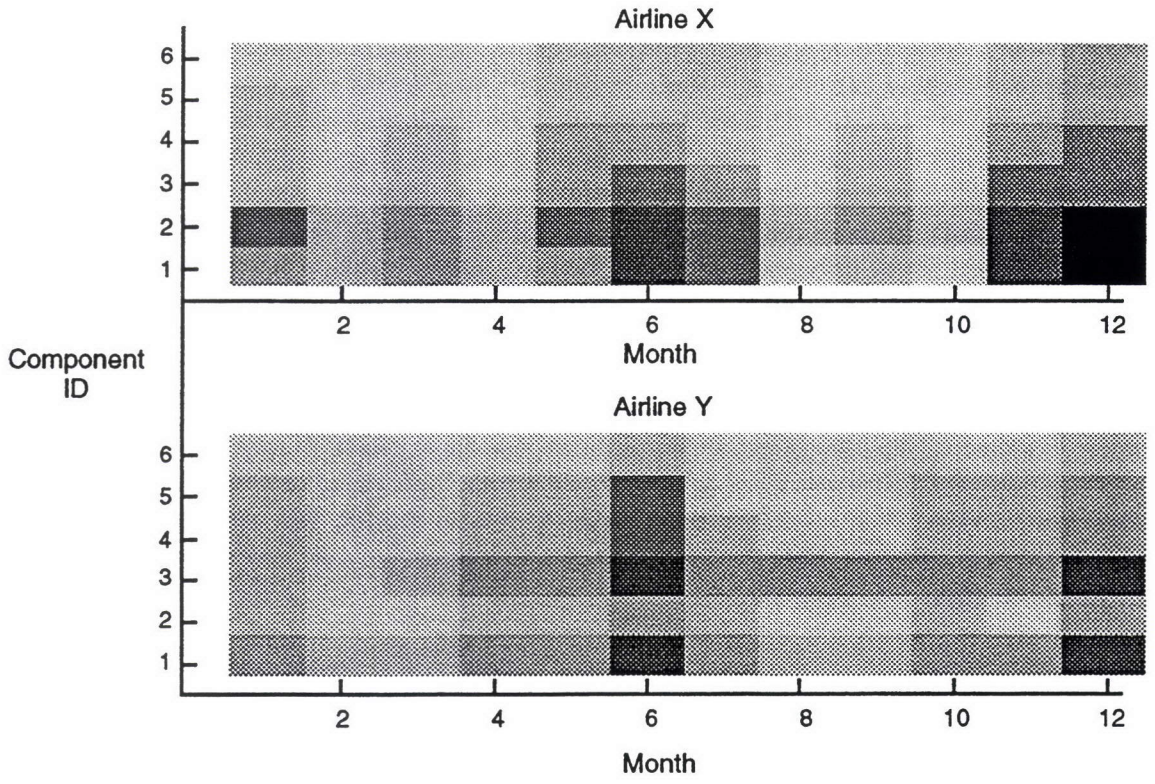

Figure 6: Image Plots

and constructive induction. Since our data is in an attribute value representation, we have not had to use the more advanced techniques in this field either. But this is the portion of our framework which has received the least work and still remains a form of black art which requires the user to know his data well.

In the area of the statistical foundation of our rules, there is the whole history of statistical thought which is the vastest field yet. We have used numerous techniques form these fields most cited in the statistical foundation section itself. It is not easy to find in the statistical literature an analysis of the statistical validity of a collection of rules; graphical models could provide a solution (see next section). We generally find ourselves retrofitting techniques developed for other purposes.

8 Future Directions

Even with the bootstrap pruning and the rule clustering, in some domains with many variables which alias each other, there will be too many good rules and the process owners will feel overwhelmed looking at them all. We are currently looking into the benefits of using a two pass system.

For instance, upon examining the first set of rules which are produced, more data engineering can be done. First, variables which never appear in the rules can be removed. Notice this will only affect the efficiency of the algorithm not the results, unless it allows enough efficiency to allow the search to go to a lower depth (e.g., more conjunct allowed per rule). Second, variables which obviously alias for one another can be combined. Remember we consider two variables to alias for one another if they can be substituted for each other in rules which predict the target. So the aliasing is seen through a projection on the target and may not extend to a total aliasing of the variables in all situations. When we discovered alias variables, we can remove all the duplicate variables but one, or we can actually combine them. For instance if V1 = A seems to be aliased by V2 $\geq$ 5, then both these variables can be removed and a new variable added which is true when either V1 = A or when V2 $\geq$ 5 and false otherwise.

It is unclear whether in a two pass system both rule extraction passes would be performed by the Brute algorithm. We are also contemplating using a Graphical models algorithm in this second pass. This could be done either by treating the rules' preconditions produced by Brute as the random variables in the graphical model and see how these relate to each other. Or in similar fashion to the two-pass Brute described above, the newly engineering variables from the first pass of Brute could be used as the random variables in the graphical model. The use of graphical models would give us more complete information about dependencies, but we are not certain it can be adapted for novice use.

References

L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth & Brooks, 1984.

Wray Buntine and Rich Caruana. Introduction to ind and recursive partitioning. NASA Ames Research Center, Mail Stop 269-2 Moffet Field, CA 94035, September 1991.

Peter Clark and Tim Niblett. The cn2 induction algorithm. *Machine Learning*, 3(4):261–284, March 1989.

Persi Diaconis and Bradley Efron. Testing for independence in a two-way table: New interpretations of the chi-square statistics. *The Annals of Statistics*, 13(3):845–874, 1985.

P.J. Riddle, R. Segal, and O. Etzioni. Representation design and brute-force induction in a boeing manufacturing domain. *Applied Artificial Intelligence*, 8(1):125–147, 1994.

J.W. Tukey. An alphabet for statisticians' expert systems. In W. A. Gale, editor, *Artificial Intelligence and Statistics.* Addison-Wesley, Reading, Mass, 1986.

J. Whittaker. *Graphical Models in Applied Multivariate Statistics.* John Wiley & Sons, Chichester, England and New York, 1990.