

# **RAMP: Rules Abstraction for Modeling and Prediction**

**C. Apte, S.J. Hong, J. Lepre, S. Prasad, and B. Rosen**

**IBM Research Division Technical Report  
RC-20271**

# RAMP: Rules Abstraction for Modeling and Prediction

Chidanand Apte, Se June Hong, Jorge Lepre, Seema Prasad, and Barry Rosen  
IBM T.J. Watson Research Center  
Yorktown Heights, NY 10598

January 12, 1996

## Abstract

Generating accurate and robust models is crucial to the successful use and deployment of classifiers on a large scale. Rule induction, i.e., generating decision rule models from data, is often a preferred approach to classification modeling and prediction, due to the enhanced explanatory capability and interpretability of decision rules. The RAMP system for rules abstraction and modeling is evolving with accuracy and robustness as primary goals. The system provides the following key capabilities: 1) feature analysis and selection based upon contextual merits technique, 2) “optimal” discretization of numerical features, 3) generation of minimal DNF (Disjunctive Normal Form) rules based upon the R-MINI algorithm, 4) rule based regression 5) rule pruning, weighting, and editing, 6) alternate rule application strategies, and 7) accuracy evaluation of the model on test data. In addition, RAMP also provides a hierarchical capability for case management, which helps end-users carry out multiple experiments on a data set, and manage these experiments as a set of related cases. RAMP has been utilized in several large-scale real-life applications and some benchmark tasks which demonstrate its robustness. We describe RAMP and its principal components in this paper.

## Keywords

Knowledge Mining and Discovery, Classification and Regression Modeling

## 1 Classification Modeling with Rule Induction

Modern classification techniques (also known as supervised learning, or non-parametric classification) operate in general by processing a set of data points (the training set) to model a class (response) feature from the independent (explanatory) features in the data. The model is formulated in a language specific to the classification technology. Well known classification methods include neural networks, decision trees, nearest neighbor techniques (used in many case based reasoning approaches), and disjunctive normal form (DNF) rule induction.

These different techniques can be compared or evaluated using a few key metrics, namely 1) accuracy and robustness, 2) cost of model generation, and 3) interpretability. Accuracy and robustness implies the ability to generate accurate classification descriptions of data across

a wide range of data behavior, including “noisy” data and “hard” (such as exclusive-OR) data. It is generally recognized that DNF rules offer a very high level of explainability and interpretability. How robust DNF rule model generation techniques can be has been an open question. Our experience with the RAMP system on some of the Statlog data sets [Michie *et al.*, 1994] indicates that its DNF rules can be quite robust.

There have been many approaches to generating DNF rules from data, a survey of which appears in [Indurkha and Weiss, 1991]. One primary approach, as exemplified by [Michalski *et al.*, 1986], [Clark and Niblett, 1989], and [Weiss and Indurkha, 1993a], works in principle by iteratively forming one rule at a time to cover some examples from the training data which are removed from consideration before repeating the iteration. The other primary approach, as seen in [Pagallo, 1989] and [Quinlan, 1993], is decision tree based, i.e., a decision tree is generated from a given set of training examples, using some combination of covering and pruning strategies, where each path from the root of the tree to each leaf of the tree represents a classification rule.

## 2 The RAMP Approach

The RAMP approach to “learning” rule-based classification models may be viewed as an alternate approach to those exemplified by systems such as Swap1 [Weiss and Indurkha, 1993a] and C4.5 [Quinlan, 1993]. While these systems rely upon some combination of greedy search based covering algorithms coupled with statistical pruning techniques, RAMP relies upon a rigorous and robust covering algorithm that requires little or no pruning. In addition, the algorithms that constitute RAMP are highly parameterized, and these parameters are made available to the user, so as to enable the fine tuning of these methods to the data sets to which they are being applied.

Sampling is employed to control computational cost, when appropriately required. Both simple as well as stratified sampling methods are used. If the raw data is uniformly distributed across the class feature values, then simple sampling suffices. If the data is skewed, stratified sampling permits the user to ensure that data points for class feature values that are highly under-represented in the raw data are over-sampled for the analysis steps. If the class feature is continuous-valued, the user will interactively examine the distribution of the values, and choose judicious cut-points to discretize the class feature.

Feature analysis is the next major step. Analyzing the independent features with respect to the class feature permits selection of a subset of features that are important to the classification process. This approach to feature analysis allows a user to discard or ignore unimportant features from any subsequent analysis. While performing feature analysis, RAMP determines the optimal cut-points for each of the numerical valued independent features, which is subsequently utilized for transforming numerical values to symbolic values.

The data sample, with the feature subset chosen from the previous step, is transformed into a completely categorical set of data points, and rule generation is applied to this data set. The RAMP rule generation employs a logic minimization inspired approach to generating a complete, consistent, and minimal model. The model is a sequence of sets of DNF rules, one set per class. Immediately after the rule generation process, comprehensive statistics are gathered about each rule.

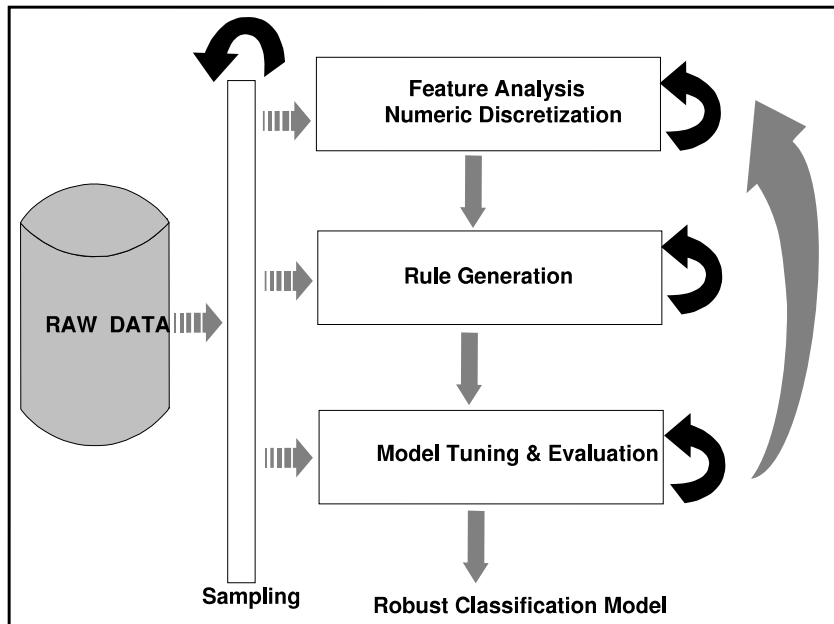


Figure 1: RAMP Classification Methodology

Utilizing the information generated from the feature analyzer, new samples (or the entire raw data) can be selected, transformed, and used as test data sets to evaluate the accuracy of the rules. If the accuracy is found to be unsatisfactory, various parameters can be modified to try and improve the classification accuracy. These include rule pruning to eliminate rules that cover potential outliers, rule weighting to assign “importance” to rules as per one of several metrics, and rule application in some precedence order, if warranted. A similar set of fine tuning parameters are available if the rules are required to do regression, that is, predict a continuous class value, as opposed to a discrete class value.

In this general sequence of problem solving, the user has the ability to iterate through several times any given stage to “optimize” the result at that stage, before proceeding to the next step. This approach is illustrated in Figure 1. The user can also repeat the whole process with new sets of parameters, as new insights are gained from the results of a previous iteration. This may be thought of as an interactive “wrapper” approach in the same spirit as the automatic “wrapper” approach introduced and described in [John *et al.*, 1994].

Each step in Figure 1 has various parameters (some of which are sketched in later sections), and it may be necessary to perform all steps before the best choices of parameter values can be determined. The RAMP system automatically maintains a natural hierarchy of experiments and supports point-and-click navigation of this hierarchy. The user may consider several feature analyses, under each of which several rule sets may be generated. Under each rule set, several choices of classification or regression parameters may be considered.

### 3 Contextual Feature Analysis and Subset Selection

Classification model generators will typically work only as well as the quality of the features from which they are trying to generate a model. Poor features will almost always result in weakly performing classification models. Even when sufficient good features are present, they may be overwhelmed by less important or even irrelevant features, if the latter are highly prevalent.

Various approaches have been used to alleviate this problem. The decision tree based methods have relied on information theoretic measures (such as the “entropy” and “gini” functions) to determine the best feature to use at each node while expanding the decision tree [Breiman *et al.*, 1984]. This basic principle may be thought of as a 1-level lookahead algorithm that determines the best feature to use at a node based on how well the feature partitions the training examples into their respective classes. Variants of this method include multi-level (usually 2) lookahead methods as well as employing simple conjuncts of features (instead of single features) as decision tests for nodes.

One well known method used in a DNF rule generator is backtracking based local optimization [Weiss and Indurkha, 1993a]. This method works in principle by attempting to constantly improve the performance of a rule while being constructed by swapping member tests (features) with new tests. Although this method appears more powerful than the decision tree methods, it may not perform well in the presence of extremely large number of features.

In most dynamic feature selection methods the discretization of numerical features is done in-process during model generation. This has an inherent weakness due to the serial nature of selecting the landmark values, or the cut points. Consider a problem where a numeric feature inherently has two or more optimum cut points, i.e. the intervals partitioned by them are meaningful in the problem domain. It is not likely that one of these points will be chosen in a serial process where the one “best” point is sought one at a time based on its ability alone to distinguish the classes.

#### 3.1 Contextual Feature Analysis

The RAMP system employs a new approach; a contextual feature analyzer that simultaneously ranks the features in terms of their classificatory power as well as determining the “optimal” cuts for each numerical feature for discretization so as to maximize that feature’s ability to discriminate. Features are ranked based upon merits that are computed by considering for each example in a class, a set of “nearest” counter examples, and accumulating a figure for each feature that is a function of the example-pair feature values.

Contextual merit is assigned to a feature taking into account the degree to which other features are capable of discriminating between the same examples as the feature under consideration. If two examples in different classes differ in only one feature, then this feature is an important feature, and is assigned additional merit. This process is repeated across all features, and between all possible “nearest” example pairs, in which the examples belong to different classes.

To compute contextual merit, the distance  $d_{rs}^k$  between the values  $z_{kr}$  and  $z_{ks}$  taken by feature  $k$  for examples  $r$  and  $s$  is used as a basis. For symbolic features, the inter-example distance is 0 if  $z_{kr} = z_{ks}$ , and 1 otherwise. For numerical features, the inter-example distance

is  $\min(|z_{kr} - z_{ks}|/t_k, 1)$ , where  $t_k$  is a threshold for feature  $k$  (usually,  $1/2$  of the magnitude of the range of the feature). The total distance between examples  $r$  and  $s$  is  $D_{rs} = \sum_{k=1}^{N_f} d_{rs}^k$ , where  $N_f$  is the total number of features. The merit of a feature  $f$  is now defined as  $M_f = \sum_{r=1}^N \sum_{s \in \bar{C}(r)} w_{rs}^f d_{rs}^f$ , where  $N$  is the total number of examples,  $\bar{C}(r)$  is the set of examples not in the same class as examples  $r$ , and  $w_{rs}^f$  is a weight function chosen so that examples that are close together are given greater influence in determining each feature's merit. In practice, it has been observed that  $1/D_{rs}^2$  if  $s$  is one of  $k$  nearest neighbors to  $r$ , and 0 otherwise, provides robust behavior as a weight function. Additionally, using  $\log_2 \#\bar{C}(r)$  as the value for  $k$  has also exhibited robust behavior.

This approach to computing and ordering features by their merits has been observed to be very robust, across a wide range of examples. Detailed studies on this method are reported in [Hong, 1994b] and [Hong *et al.*, 1995].

### 3.2 Optimal Discretization of Numerical Variables

Classification modeling methods such as decision trees and rule induction require discretization of numerical features so as to enable discrimination tests on these features. One popular approach to this problem is to dynamically chose discrete cut-points, treating each cut point as if it is one of the features to be selected. Although providing a potentially faster solution, the approach lacks robustness due to the underlying greedy search nature of dynamic feature selection. An alternate approach, as provided in the RAMP system, is to provide optimal discretization of numerical features, in conjunction with static contextual merit computations.

The principal in this method is to utilize information from the contextual merit computation as a goal, for a dynamic programming method to calculate the desired number of cuts for each numerical feature, and their individual values. For each numerical feature  $k$ , while computing the contextual merit, the values of the feature for the “near” example pairs  $(rs)$ ,  $z_{kr}$  and  $z_{ks}$  are collected, along with their individual “score”, which is the weight function  $(1/D_{rs}^2)$  used in the contextual merit computation:  $[z_{kr}, z_{ks}, 1/D_{rs}^2]$ . This collection of intervals, along with the score values, are processed by an interval covering dynamic program [Aggarwal *et al.*, 1993] to produce “optimal” cuts. An interval contributes to the cut-score if a cutpoint is within the interval. For a given number of cutpoints the interval covering algorithm produces the cut points that maximize the cut-score. The number of cuts are determined with consideration to the feature's contextual merit and the efficiency of the cuts. Further details of this method are provided in [Hong, 1994b].

## 4 Minimal Rule Generation

The RAMP rule generation algorithm (R-MINI [Hong, 1994a]) generates “minimal” classification rules from tabular data sets where one of the columns is a “class” variable and the remaining columns are “independent” features. The data set is completely discretized (i.e., continuous valued features are discretized into a finite set of discrete values, categorical features are left untouched) by the optimal numerical discretization step prior to rule generation.

While the RAMP approach to generating classification rules is similar to techniques that directly generate rule from data, it's primary goal is to strive for a “minimal” rule set that is com-

plete and consistent with the training data. Completeness implies that the rules cover all of the examples in the training data while consistency implies that the rules cover no counter-examples for their respective intended classes. Others too have argued for generating complete and consistent classification models before applying error minimizing pruning processes [Breiman *et al.*, 1984, Weiss and Kulikowski, 1991]. The RAMP system utilizes a logic minimization methodology, called R-MINI, to generate “minimal” complete and consistent rules. This technique was first developed for programmable logic array circuit minimization (MINI, [Hong *et al.*, 1974]) and is considered to be one of the best known logic circuit minimization techniques.

The merits of minimality have been well discussed [Blumer *et al.*, 1989, Rissanen, 1989]. The principal hypothesis here is that a simpler model tends to have higher accuracy. Thus, if two different models (in the same representation) both describe a particular data set, the less complex of the two will be more accurate in its description. Complexity is measured differently for different modeling techniques. For DNF rules, it would be total number of rules and total number of tests in all the rules. A smaller description will tend to be better in its predictive accuracy, and this has been borne out in our extensive evaluations.

A data set with  $N$  features may be thought as a collection of discrete points (one per example) in an  $N$ -dimensional space. A classification rule is a hypercube in this space that contains one or more of these points. When there is more than one cube for a given class, all the cubes are Or-ed to provide a complete logical classification function for the class. Within a cube the conditions for each part are And-ed, thereby giving the DNF representation for the overall classification model. The size of a cube indicates its generality, i.e., the larger the cube, the more vertices it contains, and potentially cover more example-points. RAMP’s minimality objective is first driven by the minimal number of cubes, and then the most general cubes. The most general cubes are prime cubes that cannot be further generalized without violating the consistency of that cube.

The minimality objective translates to finding a minimal number of prime cubes that cover all the example-points of a class and not cover any example-points of any counter-class. This objective is similar to many switching function minimization algorithms.

The core heuristics used in the RAMP rule generation system for achieving minimality consists of iterating (for a reasonable number of rounds) over two key sub-steps:

1. Generalization step, R-EXPAND, which takes each rule in the current set (initially each example is a rule) and opportunistically generalizes it to remove other rules that are subsumed.
2. Specialization/Reformulation, R-REDUCE, which takes each rule in the current set and specializes it to the most specific rule necessary to continue covering only the unique examples it covers. Redundant cubes disappear during this step.

This annealing-like approach to rule generation (via iterative improvements) may be potentially indefinite. A limit is used that controls how long the system should keep iterating without observing a reduction. If no reduction takes place within this limit, we can stop the minimization process. In practice, we have observed that RAMP rule generation satisfactorily converges the rule set as long as it is allowed to go through 5-7 iterations without a reduction, on most well known test data sets, as well as on some specific industrial and business data.

RAMP has been applied to many real data sets, up to those with a few hundred features and tens of thousands of examples. Preliminary evaluations suggest that complete and consistent

full cover rule sets that result from applying RAMP are much smaller than similar rule sets generated by other known techniques. Initial benchmarking studies also suggest that the predictive accuracy of RAMP's rule sets is often higher than the DNF rule sets generated by other well known methods. A detailed discussion of the rule generation component of RAMP appears in [Hong, 1994a].

## 5 Rule-based Regression

The advantages of DNF rules in classification modeling may be also useful for regression modeling, provided continuous valued class features can be handled. RAMP provides functionality for rule-based regression. At the outset, prior to performing feature analysis, a continuous valued class feature is discretized by the user, using exogenous domain knowledge. For example, if the data base contained financial securities information and the class feature was the return on a financial instrument, then this feature could be discretized using the notions of "very good", "good", "average", "poor", and "very poor". The system allows a user to view the distributions of values for the class feature, and allows mapping cut-points onto the feature utilizing exogenous information such as above coupled with the observed distributions.

Once discretized, the problem reduces to a classification modeling one. Feature analysis, numerical feature discretization, and minimal rule generation can be now performed as usual. As part of post-processing for regression, additional metrics for the rules are computed, based upon example data-points that are covered by each rule in the training data. Three parameters are attached to each rule,  $\mu$ , the mean of all the original class values of training examples covered by that rule;  $\sigma$ , the standard deviation of these values; and  $N$ , the total number of training examples covered by that rule.

When a rule set of this nature is applied to hidden "test" or unseen data, each example in that set will potentially have zero or more rules that apply to that example. In the case that no rules apply to an unseen example, we assign or predict a numerical value that suggests a default for the domain, based upon priors such as the mean for the class. In the case that one or more rules apply to an unseen example, we compute an average from the rule coverage metrics, and assign that value as the class label for that example. Two straightforward averaging approaches are the simple and weighted approach. In the simple averaging approach, we compute for each example the simple average of  $\mu$  of all rules that cover it as its predicted value (assigning a prediction value that is the mean of the class feature values in the training set if no rules cover it). Therefore, for each example in the test data, if  $M$  is the total number of rules that cover the example, its predicted class value is  $\frac{\sum_{i=1}^M \mu_i}{M}$ . A similar rule-based regression technique was employed in [Weiss and Indurkha, 1993b].

In the weighted average approach, there are several options available for weighting the rules. One of these options is to compute and assign a prediction of the weighted average, e.g.,  $\frac{\sum_{i=1}^M \frac{\sqrt{N_i}}{\sigma_i} \mu_i}{\sum_{i=1}^M \frac{\sqrt{N_i}}{\sigma_i}}$ . In general, weighted averaging usually leads to smoother correlations between predicted and actual values.

We have experimented with a suite of weighting options and error estimation techniques for rule-based regression. It has been our observation that no unique combination of weighting and error estimation seems to be uniformly applicable. The user has the ability to evaluate these

combinations on test data, determine which one tends to be most accurate, and then utilize that metric for fine tuning the model.

## 6 Rule Refinement and Application Strategies

The RAMP classification methodology takes the position that classification modeling is not a one-step process, as illustrated earlier in Figure 1. Rather, once model generation is completed, careful interactive analysis, often involving several iterations, is recommended to fine tune the model for maximized performance. To this end, several statistics are gathered immediately after the rule generation process and made available to the user. As an example, consider the following excerpt from the rule set generated from the AUSTRALIAN data set in the Statlog [Michie *et al.*, 1994] repository:

```
CLASS: Approve
Total Class Examples: 208

Rule 1:
  F4:NOT ( p )
  F8:( t )
  F10 :( 3.50 <= X )
  F13 :NOT ( 61.00 <= X < 161.50; 253.50 <= X < 329.50; )
Then ==> Approve
Examples Coverage: ( 65 0 ) New Unique Examples: 65
Running Coverage: 65/208 : 31.25% Rules: 1
Cube Size: 6.77376e+06

Rule 2:
  F4:NOT ( p )
  F5:NOT ( d k w )
  F6:NOT ( ff v )
  F7 :NOT ( 0.10 <= X < 0.60; )
  F8:( t )
  F10 :( X < 3.50; )
  F13 :( X < 161.50; 253.50 <= X < 329.50; )
Then ==> Approve
Examples Coverage: ( 44 0 ) New Unique Examples: 44
Running Coverage: 109/208 : 52.4038% Rules: 2
Cube Size: 4.25779e+06
```

As can be seen, for each rule, the statistics gathered include example coverage for that rule, as well as new unique example coverage, which is the total number of examples covered by that rule which are not covered by any of the preceding rules. Also, a running coverage indicates the fraction of the total space of training examples that are covered upto and including that rule. The cube size indicates the total number of possible data points that satisfy that rule.

In addition, a detailed view of the same rules offers more statistics on individual terms in the rule. Essentially, since each term in the antecedent of a rule is a subset membership test on a feature, the additional statistic measures, in the training example set, the number of examples that satisfied each individual member in the subset test. The following detailed display of the same two rules illustrates this view:

```
CLASS: Approve
Total Class Examples: 208
```

```
Rule 1:
```

```
F4:( g/65 gg/0 )
F8:( t/65 )
F10 :( [3.50, ]/65 )
F13 :( [ ,61.00]/38 [161.50, 253.50]/13 [329.50, ]/14 )
```

```
Then ==> Approve
```

```
Examples Coverage: ( 65 0 ) New Unique Examples: 65
```

```
Running Coverage: 65/208 : 31.25% Rules: 1
```

```
Cube Size: 6.77376e+06
```

```
Rule 2:
```

```
F4:( g/44 gg/0 )
F5:( aa/5 c/8 cc/7 e/3 ff/0 i/1 j/1 m/2 q/12 r/0 x/5 )
F6:( bb/22 dd/1 h/19 j/1 o/0 z/1 )
F7 :( [ ,0.10]/4 [0.60, 1.69]/15 [1.69, 3.88]/8 [3.88, ]/17 )
F8:( t/44 )
F10 :( [ ,3.50]/44 )
F13 :( [ ,61.00]/23 [61.00, 101.00]/9 [101.00, 161.50]/5
      [253.50, 329.50]/7 )
```

```
Then ==> Approve
```

```
Examples Coverage: ( 44 0 ) New Unique Examples: 44
```

```
Running Coverage: 109/208 : 52.4038% Rules: 2
```

```
Cube Size: 4.25779e+06
```

A domain expert may observe from the detailed display that Rule 1 did not contain any actual examples for “F4=gg”. This well could be due to the lack of examples in the training set, and the expert can decide whether “F4=gg” is a proper partial condition for this rule, and if not, edit the rule to modify the test.

In addition to these statistics, a user can also examine rules by occurrence of feature tests in their antecedents. That is, a user can selectively view only those rules that have, for example, a specific feature as an antecedent.

RAMP permits a user to perform simple rule pruning. For each class, the user may choose to remove any rules that cover less than some minimum number of examples. This threshold may be set by the user. This rule removal operation allows a user to discard rules that could be just covering potential outliers, in terms of their property of covering extremely small number of examples, in relation to total size of the example space for that class in the training set.

A rule model may be applied to new data in two ways: sequential or parallel. Individual rules may be applied in some sequence, applying rules until a match occurs. As soon as a match occurs, that example is assigned to the class of the matching rule, and no more rules are attempted for a match. If applied in parallel, one applies all rules to a new example, and then a weighted decision is made if conflicting class rules satisfy the example. For the conflict resolution step, one may assign a weight to each rule, and then choose to either sum the weights for rules in the same class, or take the maximum. and then assign the class corresponding to the “winning” rule set. The weight on each individual rule can include factors such as unity, number of examples covered in the training set, fraction of examples of a class covered in a training set, etc. Additionally, rules that didn’t quite match, but missed a full match by just 1 or 2 tests, may be brought into play by assigning a low weight, much less than 1 but greater than 0.

All these parameter settings allow a rule model to behave in different ways when applied to test data. Although RAMP provides reasonable defaults for the parameters, it also allows a user to iterate through different settings of these parameters to test whether the performance can be improved.

## 7 Accuracy Estimation

Performance evaluation and error estimation are important steps in classification modeling. Error estimation techniques have been developed in statistics for evaluating the accuracy of a classification model. These range from methods that are specifically designed for models generated from small or insufficient data sets, such as leave-one-out, bootstrap, and cross-validation [Weiss and Kulikowski, 1991], [Breiman *et al.*, 1984]. RAMP currently offers performance evaluation metrics that are common to all these techniques, and lets the user use external processes to simulate one of the above techniques if the data sets are small. If the data sets from which the modeling is being done are sufficiently large, then the basic measures of recall/precision, and total classification error, are sufficiently robust.

In general, irrespective of the size of the sample from which the rules are generated, one can evaluate the rules on as large a data set as possible, since applying rules to new data is not a computationally expensive process. The larger the data set on which the rules are being tested, the more robust the performance evaluation is.

For any given rule application to new data, the following “confusion” matrix is computed by RAMP. In this table, a matrix represents the results of classifying the test examples using the rules. Classification errors will show up in the matrix as nonzero non-diagonal entries. For example, in the following illustration, of the 99 examples that belonged to class “Approve” in the test set, 79 were correctly identified. In addition, 17 examples of class “Reject” were mis-identified as class “Approve” examples by the rules. This results in a recall of 79.80% and a precision of 82.29% for class “Approve”. Similar measures are reported for all classes in the data set. In addition, an average recall and average precision are also reported, as well as the total classification error, which is the sum of all the non-diagonal entries over the total number of examples in the test data.

Total Number Of Examples = 228

-----

	Approve	Reject	Total	Precision
Approve	79	17	96	82.29
Reject	20	112	132	84.80
NONE	0	0	0	
Total	99	129	228	
Recall	79.80	86.82		

Average Recall = 83.77%  
 Average Precision = 83.77%  
 Total Classification Error = 16.22%

Using the information in this table, a user may repeatedly iterate through the rule refinement step, modifying rule weighting and application parameters, thereby changing each of the performance measures, towards satisfying the desired performance goal.

## 8 Discussion

RAMP has been evolving over the past few years by constantly exercising it on real-world problems, represented by large scale industrial and business data. Some of the key areas in which RAMP has been applied in advanced experiments include financial markets analysis, customer underwriting selection profiling, coin sorting rules design, chemical bimetallic salt typing, insurance fraud detection, and optical character recognition. Utilizing the methodology and the approach presented in this paper has resulted in highly satisfactory modeling, even for domains which exhibited extremely noisy data or heavily skewed data. One such experimental exercise is reported in detail in [Apté and Hong, 1995].

We applied RAMP to 8 Statlog [Michie *et al.*, 1994] datasets for a benchmark experiment. All datasets except the DNA dataset contained numerical features, which required RAMP to perform pre-discretization using the contextual merit based optimal cutting algorithm. For each data set, the classification model was generated by executing the minimal rule generation 5 times, thereby generating 5 sets of rules for each class, except for the Letter dataset for which only 2 rule sets were generated. The multiple rule sets were combined and no pruning was applied. A simple voting scheme (the default in RAMP) was applied to arrive at a classification decision for each test sample.

The default rule application strategy (voting scheme) in RAMP is as follows: each rule is assigned a weight computed as  $1 + N1/N2$  where  $N1$  is the number of training examples the rule covers, and  $N2$  is the number of training examples in the rule's class. For each test example, we first determine the set of rules FM that are "fully matched" and the set of AM rules that are "almost matched", i.e., matched in all but one one of the constrained features. All the rules in FM contribute their full weight to the rule's class while the rules in AM (by default setting) contribute 0.1 of their weight to the rule's class. The class that accumulates the highest total weight wins. The evaluation was performed exactly the same way (either n-fold cross-validation or separate train and test sets) as done by the Statlog experiments.

In table 8, the error rates of the best method (as reported in the Statlog experiments) as well as those of C4.5 [Quinlan, 1993], CART [Breiman *et al.*, 1984] (in the case of Letter, we show instead results of Ind-CART), and CN2 [Clark and Niblett, 1989] are shown for reference. The RAMP error rates, its rank amongst the methods in Statlog (out of a total of 24, including RAMP), and the number of minimal complete and consistent rules (smallest amongst the multiple rule sets that were generated), averaged over the n-fold trials, are shown in the next three columns. The final column shows the average number of nodes in a MDL-pruned decision tree [Mehta *et al.*, 1995] system. Although there is a correlation between the minimal rule set size and the MDL-pruned tree size, a more direct comparison would be with the the number of leaves in the tree.

The RAMP generated rules (even without pruning and combined with an extra step of discretization, but with simple voting on multiple rule sets) ranked in the top 5 in five of the eight cases that were tested. Except for the shuttle case, that has an extremely low error rate for most of the techniques, the RAMP error rates are smaller than those of CN2 which also generates DNF rules. With a proper pruning strategy, the modeling accuracy of RAMP’s minimal rule sets can be made even more robust for a wide variety of data, as long as adequate features are available.

The two cases where RAMP has a low rank, Diabetes and Vehicle, may contain much noise (thereby making pruning a mandatory requirement), or the decision surfaces in the given feature space may not be adequate for modeling by DNF rules. Many anomalies have been observed in the Diabetes dataset, such as the use of the numeric value “zero” to record a patient’s blood pressure, when it was not known. Effective techniques for both removal of outliers (pruning training example space) as well as the minimal rule sets are being investigated. In addition, a more comprehensive benchmark that will cover more of the Statlog datasets is being pursued.

Case	Evaluation Method	Statlog Results					RAMP Results		MDL tree size	
		Best	Error Rates				Rank in 24	Avg. best size		
			Best	C4.5	CART	CN2				RAMP
Aust.Cr	10 fold on 690	Cal5	.131	.155	.145	.204	.139	3	31.2	23.6
Diabetes	12 fold on 768	Logdisc	.223	.270	.255	.289	.272	16	55.1	34.8
Vehicle	9 fold on 846	Quadisc	.150	.266	.235	.314	.289	15	55.6	72.1
Segment	10 fold on 2310	Alloc80	.030	.040	.040	.043	.0329	3	35.6	56.2
DNA	2000Trn, 1186Tst	RBF	.041	.076	.085	.095	.0489	3	50	51
Satimage	4435Trn, 2000Tst	K-NN	.094	.150	.138	.150	.126	5	146	167
Letter	15000Trn, 5000Tst	Alloc80	.064	.132	.130*	.115	.108	4	693	1175
Shuttle	43500Trn, 14500Tst	NEWID	.0001	.001	.0008	.0003	.0011	8	20	—

Table 1: Preliminary results of RAMP benchmark on Statlog data

Although automatic classification methods such as RAMP provide an additional level of sophistication for culling out useful information from vast amounts of data, they are certainly not to be viewed as black box entities that can be applied in isolation. Every domain and its data has its own peculiarities, and careful introspective analysis of the data is an important necessity to maximize the value from these methods. We have attempted to provide in RAMP

the facility to carry out these experimental analyses, helping a user to understand the data entities, determining whether new and derived features need to play a role, determining which features to discard, and finally, carefully evaluating the performance of the generated rules in conjunct with experts. The nature and longevity of the predictive power of the rule models generated via this methodology suggest that our new approach has a major role to play in learning from data.

## References

- [Aggarwal *et al.*, 1993] A. Aggarwal, B. Schieber, and T. Tokuyama. Finding a Minimum Weight K-link Path in Graphs with Monge Property and Applications. Technical report, IBM Research Division, 1993.
- [Apté and Hong, 1995] C. Apté and S.J. Hong. Predicting Equity Returns from Securities Data with Minimal Rule Generation. In *Advances in Knowledge Discovery*. AAAI Press, 1995.
- [Blumer *et al.*, 1989] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension. *JACM*, 36:929–965, 1989.
- [Breiman *et al.*, 1984] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, Monterrey, CA., 1984.
- [Clark and Niblett, 1989] P. Clark and T. Niblett. The CN2 Induction Algorithm. *Machine Learning*, 3:261–283, 1989.
- [Hong *et al.*, 1974] S.J. Hong, R. Cain, and D. Ostapko. MINI: A Heuristic Algorithm for Two-Level Logic Minimization. *IBM Journal of Research and Development*, 18(5):443–458, September 1974.
- [Hong *et al.*, 1995] S.J. Hong, J.R.M. Hosking, and S. Winograd. Use of Randomization to Normalize Feature Merits. Technical Report RC 20072, IBM Research Division, 1995. Submitted to ICDE’96.
- [Hong, 1994a] S.J. Hong. R-MINI: A Heuristic Algorithm for Generating Minimal Rules from Examples. In *Proceedings of the 3rd Pacific Rim International Conference on Artificial Intelligence - PRICAI’ 94*, pages 331–337, August 1994.
- [Hong, 1994b] S.J. Hong. Use of Contextual Information for Feature Ranking and Discretization. Technical Report RC 19664, IBM Research Division, 1994.
- [Indurkha and Weiss, 1991] N. Indurkha and S. Weiss. Iterative Rule Induction Methods. *Journal of Applied Intelligence*, 1:43–54, 1991.
- [John *et al.*, 1994] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In H. Hirsh and W. Cohen, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129. Morgan Kaufmann, 1994.
- [Mehta *et al.*, 1995] M. Mehta, J. Rissanen, and R. Agrawal. MDL-based Decision Tree Pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining, KDDM-95*, 1995.

- [Michalski *et al.*, 1986] R. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. In *Proceedings of the AAAI-86*, pages 1041–1045, 1986.
- [Michie *et al.*, 1994] D. Michie, D. Spiegelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.
- [Pagallo, 1989] G. Pagallo. Learning DNF by Decision Trees. In *Proceedings of the Eleventh IJCAI*, pages 639–644, 1989.
- [Quinlan, 1993] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [Rissanen, 1989] J. Rissanen. Stochastic Complexity in Statistical Inquiry. *World Scientific Series in Computer Science*, 15, 1989.
- [Weiss and Indurkha, 1993a] S. Weiss and N. Indurkha. Optimized Rule Induction. *IEEE EXPERT*, 8(6):61–69, December 1993.
- [Weiss and Indurkha, 1993b] S. Weiss and N. Indurkha. Rule-Based Regression. In *Proceedings of IJCAI-93*, 1993.
- [Weiss and Kulikowski, 1991] S.M. Weiss and C.A. Kulikowski. *Computer Systems That Learn*. Morgan Kaufmann, 1991.