Problem Definition, Data Cleaning, and Evaluation: A Classifier Learning Case Study

Foster Provost
Bell Atlantic Science & Technology
500 Westchester Ave., White Plains, NY 10604, USA
Phone: 914-644-2169, Fax: 914-949-9589
E-mail: provost@acm.org

Andrea Pohoreckyj Danyluk
Williams College, Department of Computer Science
Williamstown, MA 01267, USA
Phone: 413-597-2178, Fax: 413-597-4116
E-mail: andrea@cs.williams.edu

Keywords: classifier learning, evaluation, data engineering

Edited by: Xindong Wu

Received: September 1996 **Revised:** September 1998 **Accepted:** December 1998

Problem definition, data cleaning, and evaluation constitute much of the process of building useful, real-world classifiers with inductive algorithms. This paper is a case study of this process based on a long-term project addressing the automatic dispatch of technicians to fix faults in the local loop of a telephone network. The bottom line of the project is that simple learning techniques can be effective. However, constructing a convincing argument to that effect is far from simple. In particular, we had to consult multiple sources to obtain class labels, use domain knowledge to clean up data, compare with existing methods, and evaluate with data from multiple locations. Finally, it was necessary to use decision-analytic techniques to evaluate the cost-effectiveness of the learned classifiers, because evaluation based on classification accuracy is misleading without an analysis of cost-effectiveness. Our view is that application studies should be helpful in guiding future research. Therefore, we conclude by outlining useful directions suggested by our experience on this long-term project.

1 Introduction

This paper presents a case study into the process of real-world classifier learning. The case study has been taken from the long-term MAX project, which addresses the automatic dispatch of technicians to fix faults in the local loop of a telephone network. For this paper, we use the term *machine learning* to denote the automatic generation of local-loop dispatch classifiers from historical data.

In the MAX domain we wish to learn classifiers for dispatching technicians to troubleshoot telephone line problems reported by phone company customers. In this domain, a small increase in accuracy can have a large impact on the company's bottom line. For example, if we are willing to ignore details for the moment, New York State alone has over three million residential trouble reports per year. If an erroneous dispatch costs the company (on average) $100, then even a one-percentage-point decrease in dispatch error rate can save the company over $3 million annually. Therefore, it is worthwhile to investigate methods for increasing the effectiveness of local-loop trouble diagnosis.

We discuss several interrelated issues involved in determining the efficacy of inductive learning programs in this domain. As the case study will show, such a determination is more complicated than merely running a handful of learning programs on a data set and comparing the accuracies of the resulting classifiers. To be convincing, we had to use multiple sources for class labels, and use domain knowledge to clean the data. For the first set of experiments presented, we use error rate as our primary metric. However, absolute error rates are not useful for comparisons between the experiments because the problem formulation varies. To facilitate inter-study comparison, we also report the percentage decrease in error rate (PDER) as compared to classifying all instances as the most frequently occurring class (the *default class*).

Next we discuss comparisons with existing methods, including both a set of experts and an existing expert system, as well as comparisons with data drawn from geographically disparate locations. The breadth of this comparison study increased our confidence in our evaluation. Finally, we discuss that an error-rate comparison, albeit a fine starting point, is not sufficient for classifier evaluation in this real-world domain. What is important is the cost-effectiveness of the system, rather than its accuracy. Moreover, we show that a naive evaluation of cost-effectiveness also is not satisfactory, so we utilize techniques from decision analysis. From this case study a cost-sensitive learning method emerges as the most effective technique.

Our view is that application studies help to guide future research. Therefore we conclude by presenting a summary of general lessons learned and by outlining useful research directions suggested by our experience on this long-term project.

2 MAX and Machine Learning

MAX (Rabinowitz et al. 1991) is an expert system developed by NYNEX[1] Science and Technology for the purpose of troubleshooting customer-reported telephone problems. MAX deals specifically with problems in the local loop, the part of the telephone network between the central office and the customer's premises.

[1]Now Bell Atlantic. At the time MAX was developed, NYNEX was the parent company of New England Telephone and New York Telephone.

When a customer has difficulty with his telephone line he calls the phone company to report the problem (the *trouble*). A phone company representative creates a trouble report and also initiates electrical tests on the customer's line, called the Mechanized Loop Test (MLT).[2] The MLT measures the electrical signature of the customer's line and gives such information as voltages and resistances. All this information is then sent to a Maintenance Administrator (MA) who determines a high-level diagnosis for the trouble, and dispatches a technician to fix it. MAX (Maintenance Administrator eXpert) plays the role of an MA. It gives a high-level diagnosis of a trouble based upon MLT results, and other information about the customer. MAX can take one of five possible actions.

1. dispatch a cable technician (PDF);
2. dispatch an outside repair technician to the distribution wiring or customer premises (PDO);
3. dispatch a technician to the central office (PDI);
4. queue the trouble for further testing (PDT);
5. send the trouble to a human MA for diagnosis (PSH).

The problem of local-loop diagnosis was a particularly promising machine learning application for the following reasons.

1. Diagnosis in this domain is a static problem, i.e., all data are gathered and the dispatch decision is based on the values given. Difficult problems such as incorporating time are not an issue.
2. Data are abundant.
3. A knowledge base already exists, providing a wealth of information about the domain.
4. Small decreases in error rate can have a large impact.

Machine learning is also appealing because of its potential for generating dispatch knowledge that captures local differences and because of its

[2]MLT is a product of AT&T.

potential for tracking changes in dispatch knowledge as the network equipment degrades or is replaced.

Several approaches to the problem of automatically generating dispatch knowledge from data have been investigated: (1) The application of inductive learning to generate completely new knowledge bases for specific locations (Danyluk & Provost 1993a,b). (2) The application of analytic and inductive learning to modify the existing knowledge base for specific locations (Pazzani & Brunk 1993). (3) The application of techniques to perform parameter tuning (Merz, et al., 1996). This paper discusses the first of these only.

Unless stated otherwise, all results reported in this paper were generated using the C4.5 decision tree learner (Quinlan 1993) with default settings.[3] Results given are after pruning. Numbers of test examples are given with each set of runs. All results reported have been averaged over 10 runs with independent training and test sets chosen randomly. Unless indicated otherwise, all data used in the runs in this paper are taken from a single site during a period of approximately eight months, and are described by 22 features used by MAX.

3 Multiple Data Sources

Determining whether learning programs can produce effective classifiers in this domain is complicated by a general belief that it is very difficult to ascertain the "correct" dispatches for historical trouble records, which has led to a general distrust of the class labels of the examples. To produce a robust evaluation we considered three different sources for the class labels, each of which created a slightly different learning problem.

First, we used MAX to generate class labels. If one assumes that MAX is performing the task satisfactorily, the ability to learn to duplicate MAX's performance is solid evidence that machine learning approaches can be effective in generating dispatch classifiers from clean data. Second, we had experts generate class labels. If one assumes that the experts have knowledge not yet captured in MAX, then it would be useful to be able to model the classification performance of experts.

Finally, we generated class labels by cross-referencing a database of the resolutions reported by technicians in the field. The class labels generated from these resolutions are considerably noisy, due to errors in reporting and ambiguities in translation.[4] However, the ability to learn high-quality classifiers from these data would be very useful, because the potential exists to learn classifiers that capture knowledge unknown to the experts, and because the volume of data is potentially very large.

For the experiments reported in this section and the next, we evaluate learning results in two ways: (i) we measure error rate on independent test sets; and (ii) we measure the percentage decrease in error rate (PDER) of the learned concept description over the error rate of the default class. The PDER indicates the extent to which the learned decision tree decreases the error rate that would result from classifying all cases identically using the class that occurs most frequently in the training data (for which we use "the default class" as a shorthand).

The PDER is important because different data sets have different numbers of classes. MAX, for instance, has the option of diagnosing cases as being the type that need to be looked at by a human expert. These tend to be cases where the data required to analyze the trouble are missing. Field technicians, on the other hand, are not allowed such latitude. Therefore, a data set obtained from MAX will have more diagnostic class options than a data set of troubles analyzed by field technicians. Moreover, the machine learning studies also have different sets of dispatch options, which are described in detail below. Because different data sets have different numbers of classes, comparisons of absolute error rate are affected and the PDER becomes an important measure of the relative quality of the learned diagnostic knowledge.

3.1 Class labels obtained from MAX

As reported previously (Danyluk & Provost 1993a,b), we used the existing MAX expert system to create a "clean" data set from which to

[3] Earlier results were obtained with other learning techniques, including rule learners and neural network learners, but C4.5 consistently has yielded results that are at least as good as the other systems.

[4] Some of the codes used to describe resolutions do not map uniquely to dispatch classes.

learn. We ran a series of experiments with the goal of showing that given good data we could learn knowledge to recreate MAX's behavior. We found that given a large enough quantity of data, using machine learning we can duplicate MAX's performance very well. As shown in Table 1, training on 1000 troubles yields an error rate of 0.09; training on 5000 troubles yields an error rate of 0.04. Although these results show promise for machine learning as a method of creating the knowledge base for a dispatch system, they do not offer a solution to the problem of generating knowledge that will increase the performance of MAX.

Table 1: MAX data; five class problem. Size of test set = 871. Average error rate (ER) with the default class (PDT) = 0.54.

Training	ER	StDev	Avg PDER	StDev
100	.34	.04	36.51	7.92
500	.14	.02	73.44	3.90
1000	.09	.01	82.41	2.58
2000	.06	.01	89.31	2.61
5000	.04	.01	93.62	1.53

3.2 Class labels obtained from experts

In order to evaluate the potential of machine learning as a tool to build *a better* MAX, we enlisted the help of several experts in local-loop troubleshooting. The experts were phone company veterans with many years of experience in the areas of maintenance and repair of the local loop. We ran a set of experiments testing the ability to learn dispatch knowledge from expert-classified data. The rationale behind this set of experiments is that if machine learning can create knowledge that models the behavior of human experts well, then it may be possible, albeit resource consuming, to have local experts analyze large numbers of troubles and then to learn classifiers from these data.

Table 2 shows results for one expert who analyzed 500 troubles from one site. The results show that C4.5 can model the expert's behavior fairly well as compared to the default. Similar analyses of other experts' answers yielded comparable results. The large PDER suggests the potential for learning programs to model the behavior of human experts. Unfortunately, the size of the data set in these experiments was limited due to the limited availability of experts. The previous results of modeling MAX suggest that 400 examples may be too few for effective learning. An analysis of the classifiers learned from the MAX data explains why many examples are needed: very small disjuncts comprise a large portion of the concept description (Danyluk & Provost 1993a). Large data sets are necessary to learn small disjuncts with confidence (Provost & Aronis 1996).

Table 2: Expert data; five class problem. Size of test set = 100. Average error rate (ER) with the default class (PDT) = 0.58.

Training	ER	StDev	Avg PDER	StDev
100	.39	.04	32.88	8.59
400	.35	.04	38.75	6.60

Our intention had been to increase the volume of data by using multiple experts to generate a larger expert-classified data set. However, this exercise revealed that there is not a high degree of agreement among experts as to the correct classification for a trouble. In fact, the error rate of the classifiers learned from the expert data was approximately equal to the error rate obtained when one expert was used to generate class labels for the evaluation of another expert. This suggests that the problem is much more difficult than previously thought. It also offers an explanation for the general distrust of class labels.

3.3 Class labels obtained from field technicians

The third data source from which we obtained class labels for troubles is the reporting of field technicians who fix ("resolve") the troubles. In order to generate class labels, we translated their resolution codes into the corresponding dispatches using a standard mapping. As the results in Table 3 show, the performance of the learned decision trees is less than inspiring. However, the learned trees do perform slightly better than the default.

Table 3: Technicians' data; four class problem. Size of test set = 863. Average error rate (ER) with the default class (PDF) = 0.62.

All Features				
Training	ER	StDev	Avg PDER	StDev
100	.61	.03	0.96	5.72
500	.60	.02	3.73	3.90
1000	.59	.01	4.51	2.53
5000	.58	.01	6.56	2.68

Vercode Only				
Training	ER	StDev	Avg PDER	StDev
100	.62	.04	-1.31	7.03
500	.54	.01	11.63	2.80
1000	.53	.02	13.24	2.80
5000	.52	.01	15.98	2.10

Quite surprisingly, we were able to increase significantly our ability to dispatch accurately by reducing the feature set to a single feature: *vercode*. Reducing the feature set to a single feature produces decision stumps, i.e., decision trees that split on a single feature only (Holte 1993). Vercode, generated by MLT, is a summary of the electrical readings into 50–150 categories; the decision stumps therefore have 50–150 leaves. As the results in Table 3 show, the decision stumps learned by C4.5 on the field data have higher accuracy than the decision trees learned with larger feature sets.

4 Cleaning up the Data

The experiments with class labels generated by MAX suggest considerable promise for machine learning in this domain. The experiments with class labels generated by the experts suggest that it is possible to model expert behavior to some degree, but that small expert-classified data sets are not sufficient to model expert behavior with high accuracy. Moreover, the disagreement among experts suggests that even if a given expert's behavior can be modeled with high accuracy, there will still be questions about the expert's performance. The most promising source of class labels is the field technician database. This database is very large and (arguably) based on fact rather than conjecture. Unfortunately, the learning programs had the most difficulty modeling these data. This almost certainly is because, with the given set of features, MAX generates class labels deterministically (and probably so do the experts), while the technicians' class labels are inherently probabilistic.

Analyzing the different trouble resolutions reported by the field technicians suggests some concrete reasons why machine learning programs would have a difficult time modeling the data. For some borderline resolutions at the interface between the cable and the distribution wiring, it is not clear what the correct dispatch should have been because the diagnosis cannot be mapped to a dispatch unambiguously. Furthermore, there are many cases for which the resolution is a "Test OK." This resolution indicates that the technician retested the line in the process of attempting to locate the trouble, and found that there was no longer a problem. Unfortunately, it is impossible to tell the difference between cases where there was no longer a problem to fix (e.g., the customer's second phone had been off the hook and was subsequently placed back on) and cases where the manifestation of the problem was transient (e.g., the trouble had been a short circuit due to the presence in a cable of water that had dried by the time the technician retested the line). Thus determining what the correct dispatch should have been is difficult.

We wanted to evaluate whether increasing the quality of the field data would improve the ability of a learning program to produce accurate classifiers. To this end, we used prior knowledge of trouble resolutions and dispatches to clean up the field data. Specifically, we eliminated from the data all troubles for which the resolution was "Test OK." Additionally, we removed cases where it was impossible to determine from the resolution codes the correct dispatch, especially borderline cases. The effect of the data cleaning was to provide us with a set of cases for which we have only three class labels (PDF, PDO, PDI), but for which we have (relatively) high confidence in the correctness of those labels. We now describe a set of experiments that investigate the effect on learning of cleaning up the data.

As the learning results in Table 4 show, the per-

Table 4: Cleaned data; three class problem. Size of test set = 686. Average error rate (ER) with the default class (PDF) = 0.47.

All Features				
Training	ER	StDev	Avg PDER	StDev
100	.41	.04	12.04	7.52
500	.38	.03	19.31	6.94
1000	.37	.02	20.72	3.94
2000	.36	.02	23.23	3.25

Vercode Only				
Training	ER	StDev	Avg PDER	StDev
100	.38	.04	18.97	10.07
500	.35	.02	26.48	3.90
1000	.35	.01	26.73	2.87
2000	.34	.02	27.44	4.15

formance on the cleaned-up data is considerably better than the performance on the original field data. It is important to note that the cleaned-up data have only three classes instead of four, and using the default yields a lower error rate than on the previous data. However, as the results in Table 4 show, the percentage decrease in error rate (PDER) for the learned concept descriptions is larger than with the original data.

These results provide support for the conclusion that from clean field data it is possible to learn more accurate classifiers. It must be noted, however, that by separating out the cases for which the final resolution is unambiguous, we may also be separating out the cases that are "easy" to diagnose. The effect of using this learned knowledge on the entire spectrum of troubles is still an open question, made very difficult by our inability to know the "correct" answer.

It should be noted that an alternative problem redefinition may also be effective. Specifically, much of the aforementioned ambiguity can be eliminated by combining two of the three dispatch classes. PDF (dispatch to a cable technician) and PDO (dispatch to an outside repair technician) both address problems in the "outside plant." There are *a priori* reasons why it might be desirable to combine these classes into a single "dispatch out" class. For example, training technicians to handle a larger class of problems may eliminate the need to separate problems in the outside plant. In order to test the hypothesis that we could differentiate accurately between dispatching "in" to the central office and dispatching "out" to the outside plant, we combined the two outside plant dispatches in the cleaned-up dataset. In doing so, we were able to reinsert those troubles eliminated because of PDF/PDO ambiguity. The results for the two-class problem are given in Table 5. As the table shows, the performance of the learned decision models is considerably better than the default when trained on large (2000 examples) data sets. In the table we report particularly high standard deviations for PDER in two cases. Inspection of the 10 runs shows that in two cases, the learned model performed similarly to the default, but in the remaining eight, it outperformed the default significantly.

Table 5: In vs out; two class problem. Size of test set = 738. Average error rate (ER) with the default class (PDO) = 0.09.

All Features				
Training	ER	StDev	Avg PDER	StDev
100	.09	.01	0.35	1.04
500	.09	.01	-0.50	2.11
1000	.08	.02	10.12	13.29
2000	.07	.01	24.89	2.93

Vercode Only				
Training	ER	StDev	Avg PDER	StDev
100	.09	.01	0.15	0.19
500	.09	.01	0.15	0.19
1000	.09	.01	0.15	0.19
2000	.07	.01	21.41	11.70

5 Comparison with Existing Methods

In the previous sections we compared the ability of learning programs to produce accurate classifiers from several different perspectives. The use of the field data as the source of class labels allows us to compare the performance of the learned classifier with the performance of MAX (and with the performance of the experts). Such a comparison has been a major component of Bell Atlantic's evaluation of the potential for learned knowledge to help with local-loop dispatch.

Table 6 compares the performance of the vercode decision stumps with the MAX expert system on the three different versions of the field data (discussed above).[5] The comparison is complicated because MAX does not give solid dispatches on all the cases; it routes some difficult cases to a human analyst, and for others it requests additional tests. The decision stumps, on the other hand, produce a dispatch for every case.

It is important that we be as fair as possible in our comparison of the learned decision stumps and MAX. It is inappropriate to assume that MAX is in error each time it routes a trouble to an analyst. On the other hand, it is unfair to penalize the learned decision stump for being forced to make a decision on all cases. In order to make the comparison equitable, Table 6 reports

- error rates for MAX and for the learned decision stump (LDS) on all the test data
- error rate for MAX on the subset of cases for which it chose to make a dispatch (MAX-D)
- error rate for the learned decision stump on the subset of cases for which MAX made a dispatch (LDS-D)
- error rate for the learned decision stump on the subset of cases for which it was confident (LDS-C).[6]

Table 7 gives the sizes of the subsets, as a percentage of the entire dataset.

As Table 6 shows, with little exception, the learned decision stump outperforms MAX. The increase in performance using the learned vercode mapping over the MAX system is one piece of evidence supporting the conclusion that by looking at the data we can extract dispatch knowledge that can improve MAX's performance.

[5]A comparison with the experts is not included in the summary, because the small number of troubles analyzed by the experts makes the performance of the experts incomparable.

[6]In all cases, a decision was deemed "confident" only if the estimated probability of membership in the predicted class was at least 0.6.

A potential criticism of the above argument is that the learning is fitting systematic error in the data (and that MAX actually may be as good or better at dispatching). Support for the contention that the learned knowledge is *not* just modeling errors in the data comes from a comparison of the effectiveness of the learned knowledge for dispatch in other geographic areas. In order for the effect of modeling systematic error to generalize across locations, the error must be systematic throughout the company. Furthermore, since we are using a vercode decision stump, the error must be systematic with respect to the vercode alone. We believe that this combination is highly unlikely.

To test the hypothesis that positive results are not just from modeling local systematic error, we trained decision stumps on the data from one location (X) and used them for dispatch in four other areas (A,B,C,D). As shown in Table 8, in three of the four comparisons, the knowledge learned in one area transfers well to the other areas. Note that this is especially true for the Cleaned data. The number of training examples were 5000, 2000, and 3000 for Field, Cleaned, and In vs Out, respectively. The number of test examples varies for each site. All numbers reported are the averages of testing ten decision trees on all of the data from each of the sites A, B, C, and D. Note that we report PDER as well as error rates, due to the differing class distributions among the sites.

6 Cost-effective Dispatch

If the field technicians' resolutions are taken to be reasonably reliable, the previous analysis seems to imply that MAX's performance is poor. We are faced with the issue of analyzing this seemingly poor performance in light of evidence to the contrary. The system has been in use for many years and has not had a negative effect on the operations of the company. One explanation could be that the technicians just do not code the trouble resolutions correctly. However, as the results below show, much of this seeming discrepancy can be explained by the fact that accuracy (or error rate) is not the best metric with which to evaluate dispatch effectiveness.

In the domain of local-loop repair and maintenance, the costs associated with the diagnoses

Table 6: Comparison of error rates of Learned Decision Stumps (LDS) and MAX. Standard deviations are given in parentheses, except (*), which indicates that the evaluation was performed on the entire data set, rather than on a small test set reserved after learning.

	Field Data (4 class)	Cleaned (3 class)	In vs. Out (2 class)
MAX	.67 (*)	.79 (*)	.58 (*)
LDS	.52 (.01)	.34 (.01)	.07 (.01)
MAX-D	.67 (.01)	.42 (.03)	.04 (.01)
LDS-D	.52 (.01)	.31 (.02)	.04 (.01)
LDS-C	.34 (.10)	.27 (.02)	.06 (.01)
Default	.62 (.01)	.47 (.01)	.09 (.01)

Table 7: Coverages of test data. Standard deviations are given in parentheses.

	Field Data (4 class)	Cleaned (3 class)	In vs. Out (2 class)
MAX	100 (0.00)	100 (0.00)	100 (0.00)
LDS	100 (0.00)	100 (0.00)	100 (0.00)
MAX-D	99.43 (0.18)	56.59 (1.90)	55.66 (1.78)
LDS-D	99.43 (0.18)	56.59 (1.90)	55.66 (1.78)
LDS-C	9.30 (3.15)	72.19 (5.60)	99.27 (0.52)
Default	100 (0.00)	100 (0.00)	100 (0.00)

vary substantially. Typically, the cost associated with dispatching a trouble outside of the central office is greater than dispatching to the central office, with the highest cost being associated with dispatching cable technicians. By analyzing the cases for which the decision stump and MAX differ in their dispatches, we find that MAX is making conservative decisions with respect to cost. Thus a convincing comparison of methods for local-loop dispatch must be made with respect to cost-effectiveness in addition to accuracy (Pazzani et al. 1994, Provost 1994). Our focus for this section will be on the three-class version of the MAX problem (cleaned-up data).

6.1 Evaluating Results: Cost-Effectiveness and Accuracy

We now consider the cost that would be incurred by any incorrect decisions made. This task is complicated by the fact that, as discussed in the decision analysis literature (Weinstein & Fineberg 1980), it is often difficult to estimate costs. For instance, certain tests in the central office might require much more time than others, resulting in higher labor costs to determine that the trouble is elsewhere. We interviewed experts to determine, as well as we could, the error costs associated with each of the three dispatchs (PDF, PDO, PDI). Our best approximation is a *cost ratio* of 3:2:1 (PDF:PDO:PDI), with the cost of a central office dispatch (PDI), the *base cost*, about $50.

A naive approach to cost-sensitive classification is to use error costs such as these in combination with estimates of the probabilities of the classes to determine which dispatch will yield the lowest expected cost (EC). The corresponding naive approach to evaluating cost-effectiveness is to classify a test set with the learned classifier and sum up the costs of each incorrect dispatch, using the costs defined above. This is the approach that has been taken in most prior work on cost-sensitivity in the machine learning literature (Turney 1996).

This naive approach is problematic for multiclass problems, because it assumes that after the dispatch is identified as being incorrect, the subsequent dispatch will be correct. The problem can be seen clearly in the following example. Given

Table 8: Comparison of error rates of knowledge learned from location X when applied to other locations.

Location	Field	PDER	Cleaned	PDER	In vs Out	PDER
X	.52 (.01)	16	.34 (.02)	27	.07 (.01)	29
A	.54 (.01)	19	.25 (.01)	51	.05 (.01)	74
B	.57 (.01)	4	.38 (.01)	25	.07 (.01)	-12
C	.56 (.01)	7	.21 (.01)	58	.03 (.01)	2.3
D	.64 (.01)	-2	.51 (.04)	42	.18 (.01)	-2.2

the 3:2:1 cost ratio defined above, assume that the estimated probability distribution of classes (PDF:PDO:PDI) is 0.5:0.4:0.1. In this case the dispatch with the (naive) minimum expected cost is PDI: $EC(PDI) = .9 * 50 = 45$, $EC(PDO) = .6 * 100 = 60$, $EC(PDF) = .5 * 150 = 75$. However, a choice of PDI would be incorrect 90% of the time, and in most cases would not make the choice between PDO and PDF any easier.[7] Indeed, such a naive strategy yields undesirable results in practice.

The alternative is to take a more complex, decision-analytic approach, in which the expected-cost calculation takes subsequent decisions into account. Ideally, for determining the best dispatch for a given trouble, we would like to use the frequencies of classes at the leaves of the decision stump to estimate the class probability distributions for all possible combinations of decisions, in order to calculate the minimum expected-cost dispatch. However, one goal of this analysis is a comparison with the dispatch decisions of the MAX expert system. For MAX, we know only the first dispatch; we do not know what subsequent decisions MAX would make. Thus, using the probability distributions at the leaves of the decision stump for more than just the first decision may give the decision stump an unfair advantage in the comparison, because if MAX were programmed differently, it would be able to issue recommendations for subsequent dispatches as well.

In sum, we are faced with a dilemma; it is obviously important to take subsequent dispatches into account, but we do not know what subsequent dispatches MAX would make. To resolve the dilemma, we used the prior probability distribution of the classes to determine likely subsequent decisions. This information is built into a cost matrix, so that it can be used both to evaluate the decisions of classifiers (such as MAX) that give only a single answer, and to choose cost-sensitive dispatches in cost-sensitive classifiers. Fortunately, we found that there is very little difference in cost-effectiveness between using the prior probability distribution and the leaf probability distribution for determining the second dispatch when the first is wrong. We will now describe in detail the process of building cost matrices that take subsequent (expected) errors into account.

[7] In fact, we assume independence of solutions.

First let us define the function $cost(x)$, which, based on the cost vector, gives the cost of mistakenly choosing dispatch x.

For the *naive approach*, the cost matrix is built by assigning

$$NCost(p)(a) = \begin{cases} cost(p) & \text{if } p \neq a \\ 0 & \text{otherwise} \end{cases}$$

where $p = predicted$ and $a = actual$. For the *decision-analytic approach*, we assume that the subsequent dispatch will be the minimum expected-cost dispatch of the remaining choices, based on the prior probability distribution. Suppose there are three classes, X, Y, and Z, and let $p = X$.

$$DACost(p)(a) = \begin{cases} cost(X) + SecCost & \text{if } a \neq X \\ 0 & \text{otherwise} \end{cases}$$

Without loss of generality, let $a = Y$, then

$$SecCost = \begin{cases} cost(Z) & \text{if Z is the min exp-cost class} \\ & \text{between Y and Z} \\ 0 & \text{otherwise} \end{cases}$$

In this case, the expected cost of a secondary dispatch, e.g., Y, is the probability of Y being wrong times the cost of being wrong,[8] or $(1-(p(Y)/(p(Y)+p(Z)))) * cost(Y)$, where $p(Y)$ and $p(Z)$ are the prior probabilities. The fractional probability term is due to the removal of X as a possible correct *secondary* dispatch.

An example of a decision-analytic cost matrix calculated from example costs and data priors is given in Table 9. Note that all costs here are error costs. The cost is zero for correct dispatches.

6.2 Building a Cost-sensitive Decision Stump

We built cost-sensitive decision stumps by recording at each leaf a frequency-based probability estimate for each class. The estimate was calculated as (TP/(TP+FP)), where TP is the true-positive coverage of the leaf and FP is the false-positive coverage of the leaf. When the cost-sensitive stump is used, it uses the conditional probabilities at the leaves to dispatch to the minimum expected-cost class, using the decision-analytic cost matrix (built using prior probabilities from the training data to determine expected subsequent dispatches, as described above). Note that Pazzani et al. (1994) found that estimating class probabilities at the leaves of a decision tree and using these for a minimum expected-cost calculation is not effective at reducing cost; they account for this phenomenon by noting that the probability estimates at the leaves of a decision tree are based on small samples, and thus are inaccurate. Since we use a decision stump, we hope that the larger numbers of examples at the leaves will lead to better probability estimates.[9]

Results comparing MAX with the vercode decision stump and cost-sensitive decision stump are summarized in Table 10. The cost matrix used to generate these results is that in Table 9. A simplistic comparison of the performance of MAX, the vercode stump, and the cost-sensitive stump (first, second, and fifth rows of the table) shows that although the dispatches made by the vercode decision stump are more accurate than those of MAX, the decisions made by MAX are more cost-effective. The cost-sensitive decision stump reduces the cost without losing accuracy.

However, this comparison masks an important subtlety. Specifically, as with the earlier error-rate comparisons, MAX only gives a dispatch recommendation on (approximately) 57% of the cases; the rest are routed for further testing or for human analysis. On other hand, the stumps give dispatch recommendations on 100% of the cases.

In Table 10, we therefore also report the error rate (ER) and Error Cost per Dispatch for the stumps on those cases for which MAX gave a dispatch recommendation (MAX-D), and on those cases for which the stumps were confident of their recommendation (i.e., the probability of class membership was ≥ 0.6).

As expected, the decision stumps perform considerably better on both subsets of cases, in terms of both error rate and cost. Perhaps surprisingly, the difference in performance between the cost-sensitive and non-cost-sensitive stumps is no longer apparent when they are evaluated on the subsets. This is because as the required confidence level is raised, the behaviors of the two types of stump are more and more similar, eventually becoming identical. Apparently, a threshold of 0.6 is sufficient (effectively) for the cost matrix being used.

6.3 Sensitivity Analysis

While the results above suggest that it is possible to learn cost-sensitive decision stumps that are both more accurate and more cost-effective than MAX, we must have confidence that this is not due to a fortuitous choice of costs (especially since the specification of costs is far from perfect). To this end, we perform an analysis of the evaluation's sensitivity to changes in the cost ratio.

For this paper, we consider varying only the ratio PDF:PDO, holding the ratio PDO:PDI at 2:1. Consider the cost ratio to be X:1:0.5 (PDF:PDO:PDI). Figure 1 shows the effect of varying X from 1 to 3 in increments of 0.1 on the error costs associated with the dispatches made by MAX, the decision-stump, and the cost-sensitive decision stump, using decision-analytic cost matrices constructed as described above. Figure 2

[8] When Y is correct, the error cost is zero.

[9] Recent work suggests that cost-sensitive classification with decision trees can be quite effective, if the probabilities are generated using the Laplace estimate rather than a simple frequency-based estimate (Bradford et al. 1998). The Laplace estimate protects against unwarranted optimism due to small samples.

Table 9: Cost matrix for dispatch classes in the MAX domain. Rows are predicted classes. Columns are actual classes. Classes:(PDF:PDO:PDI) Costs:(150:100:50) Priors:(0.51:0.35:0.14)

	PDF	PDO	PDI
PDF	0	150	250
PDO	100	0	250
PDI	50	200	0

Table 10: Comparison of Vercode Decision Stumps and MAX. Training sets of 2000 examples used to build the Vercode stump and Cost-sensitive stump (ER = error rate). Independent test sets of 686 examples used to test. All avgs are over 10 runs. Standard deviations are given in parentheses.

	Total Preds Made	ER	Error Cost per Dispatch
MAX (MAX-D)	56.69% (1.9)	0.42 (.03)	50.61 (3.87)
Vercode stump	100% (0.0)	0.34 (.01)	51.44 (2.35)
Vercode stump (MAX-D)	56.69% (1.9)	0.31 (.02)	43.77 (2.66)
Vercode stump (Conf)	72.19% (5.6)	0.27 (.02)	39.41 (3.00)
Cost-sensitive stump	100% (0.0)	0.35 (.02)	47.50 (2.47)
Cost-sensitive (MAX-D)	56.59% (1.9)	0.33 (.01)	43.78 (2.58)
Cost-sensitive (Conf)	70.90% (5.1)	0.27 (.02)	39.08 (2.75)
Always dispatch PDF	100% (0.0)	0.47 (.01)	83.92 (2.41)
Always dispatch PDO	100% (0.0)	0.66 (.01)	85.93 (2.15)
Always dispatch PDI	100% (0.0)	0.87 (.01)	94.88 (2.67)

and Figure 3 show the effects of varying X for the stumps when evaluated on MAX-dispatched and confident cases, respectively.

As would be expected, the cost per dispatch of the decision stump increases smoothly (and linearly) with the increasing cost of making PDF errors. The decision stump always makes approximately the same percentage of PDF errors, so as the cost of a PDF error increases linearly, so will the cost-per-dispatch of the decision stump.

The performance of MAX as the cost of a PDF error increases is more interesting. Inspection reveals that the curve representing MAX's error cost per dispatch is (approximately) piecewise linear with increasing PDF error cost, and the slope of each segment is less than the slope of the decision stump curve. The relatively low slope of each segment is due to the fact that MAX errs on the conservative side; specifically, it makes fewer PDF errors than the decision stump. Thus, the growth of the overall cost per dispatch as the PDF error cost grows will be smaller.

The discontinuity when the PDF:PDO error cost ratio reaches 2:1 can be explained by examining the changes in the cost matrices as the ratio increases. In particular, consider the two cost matrix entries $DACost(PDF)(PDO)$ and $DACost(PDO)(PDF)$.[10] Across the range of ratios represented in the graph, $DACost(PDF)(PDO) = cost(PDF)$, because in this range PDO is always the minimum expected-cost secondary dispatch. Similarly, when the ratio of the error cost of PDF to PDO is in the range $[1, 2)$, $DACost(PDO)(PDF) = cost(PDO)$. This explains technically why the slope of the MAX curve is less: $cost(PDO)$ is constant; $cost(PDF)$ increases linearly. However, when the ratio is in the range $[2, 3]$, $DACost(PDO)(PDF) = cost(PDO) + cost(PDI)$, because due to the prior distribution

[10] Recall that the cost matrix entries are of the form $DACost(predicted)(actual)$.

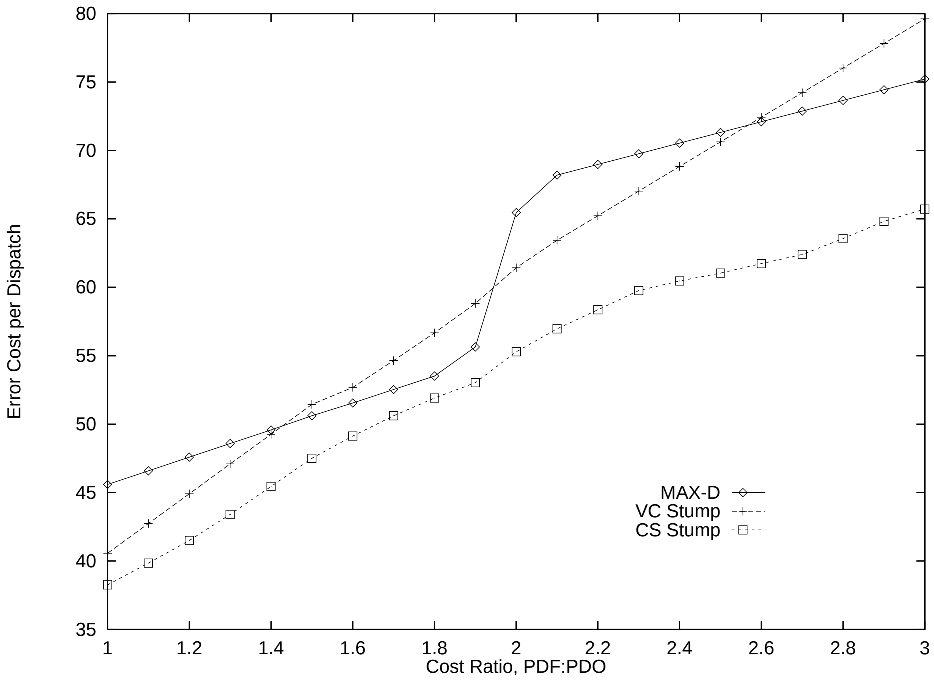

Figure 1: Effect of Varying Error Cost Ratios on Cost of Errors Made

of classes, PDI becomes the secondary dispatch of choice. Thus $DACost(PDO)(PDF)$ is still constant, but it is greater than it was over the prior range. Hence the curve's piecewise linearity. This reasoning applies to the vercode stump as well, though the difference in slopes of the two line segments is very slight, and thus difficult to recognize from the graph.

The result of these two factors is that MAX's performance, in terms of error cost per dispatch, is better than the decision stump when the ratio of the cost of a PDF error to a PDO error is in the range $(1.4, 2)$. As mentioned above, our problem analysis determined (independently) that the actual cost ratio is approximately 1.5. Thus, the design of MAX and the years of tuning its performance in the field seem to have been effective.

The graph also shows that the cost-sensitive stump adjusts for the different cost ratios automatically. Across the entire range, the cost-sensitive stump out-performs the other two methods, although the difference is small in the range of MAX's maximal effectiveness. Statistical tests on individual points do not indicate that the difference is statistically significant, and statistical tests on the entire curves are difficult because of interdependencies in the generation of the different points. However, it is not clear that statistical significance is particularly important. Even if the performance were indistinguishable, the cost-sensitive decision stumps are preferable for their simplicity, for their flexibility in adapting to different cost scenarios, and for their ease of updating. From the perspective of business significance, a potential cost savings of two or three dollars per dispatch is very significant. Also, it should be noted that (as above) these comparisons are somewhat unfair to the decision stumps, because they are making recommendations on all cases, whereas MAX is only making recommendations on about half of the cases.

Figure 2 shows that both the vercode decision stump and the cost-sensitive decision stump outperform MAX across the entire range, when classifying only those cases on which MAX makes a recommendation. Again, as would be expected, the cost per dispatch of the decision stump increases smoothly and linearly with the increasing cost of making PDF errors. It simply makes fewer of these errors when asked to make a call on fewer cases.

Also, as expected, the vercode decision stump and the cost-sensitive stump perform similarly in the range [1,2), where the cost ratio PDF:PDO is relatively low. When the PDF cost becomes increasingly large, the cost-sensitive stump adjusts

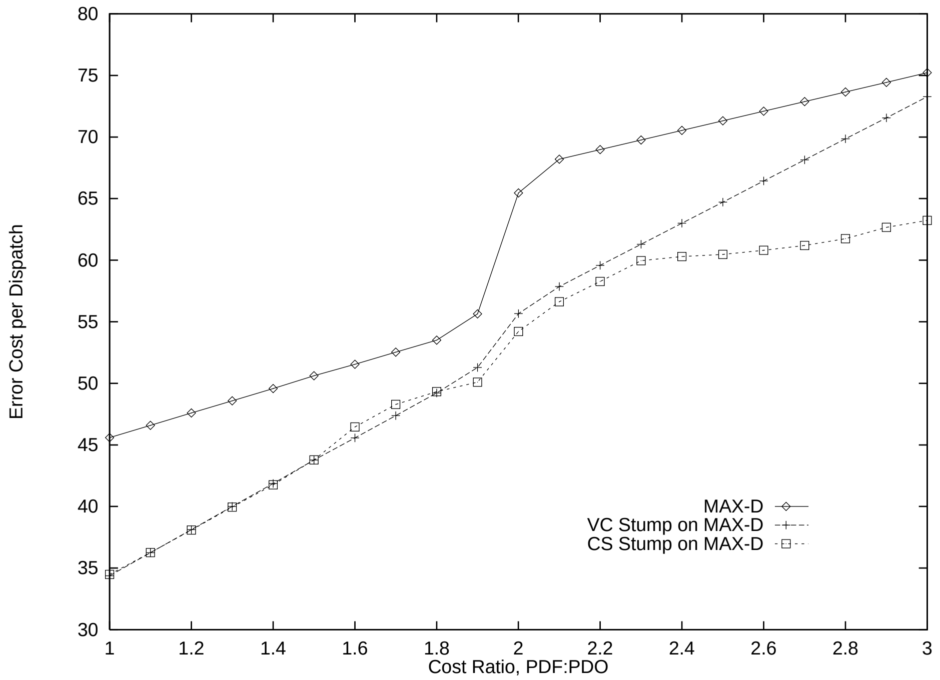

Figure 2: Effect of Varying Error Cost Ratios on Cost of Errors Made. Data are restricted to MAX-dispatched cases.

for that cost, diverting classifications to less expensive dispatches.

An even greater disparity in error costs is seen when the vercode stump and cost-sensitive stump dispatch only on confident cases.

7 Summary and Discussion

For this case study we have selected a series of experiments that highlight the variety of perspectives that must be taken in order to determine the potential for inductive learning programs to be applied successfully. The case study highlights issues of problem definition, data cleaning, and evaluation that are usually glossed over (or simply ignored) in most published reports on classifier learning. Taken in total, the results provide solid evidence that simple inductive learning programs can learn effective classifiers for local-loop troubleshooting.

At first glance, the primary result is that decision stumps can be learned that are more accurate and more cost-effective than the troubleshooting system currently in place. What is more interesting, however, is that the stumps achieve at least equivalent performance with much less effort in design, implementation, and tuning. This suggests that dealing with new equipment or with different local environments (e.g., Manhattan versus Maine) will be much easier. In the long run, being able to do a better job of keeping systems well-tuned may magnify the differences in performance observed here.

From the standpoint of the machine learning and the knowledge discovery communities, the study is most interesting as a counterbalance to the prevailing narrow view of classifier learning. In the first place, in most inductive learning research the correctness of the class labels is a basic assumption that goes unquestioned. Perhaps more strikingly, although it is difficult to imagine a real-world problem for which all errors have equal costs, equal error cost is another unquestioned assumption of the vast majority of research on inductive learning.[11] This case study shows how each of these assumptions can lead to a misleading evaluation.

7.1 Lessons Learned

From this case study we can draw several general lessons that we believe are applicable to many

[11]For a detailed analysis of this assumption, see the recent paper by Provost, Fawcett and Kohavi (Provost, Fawcett, & Kohavi 1998).

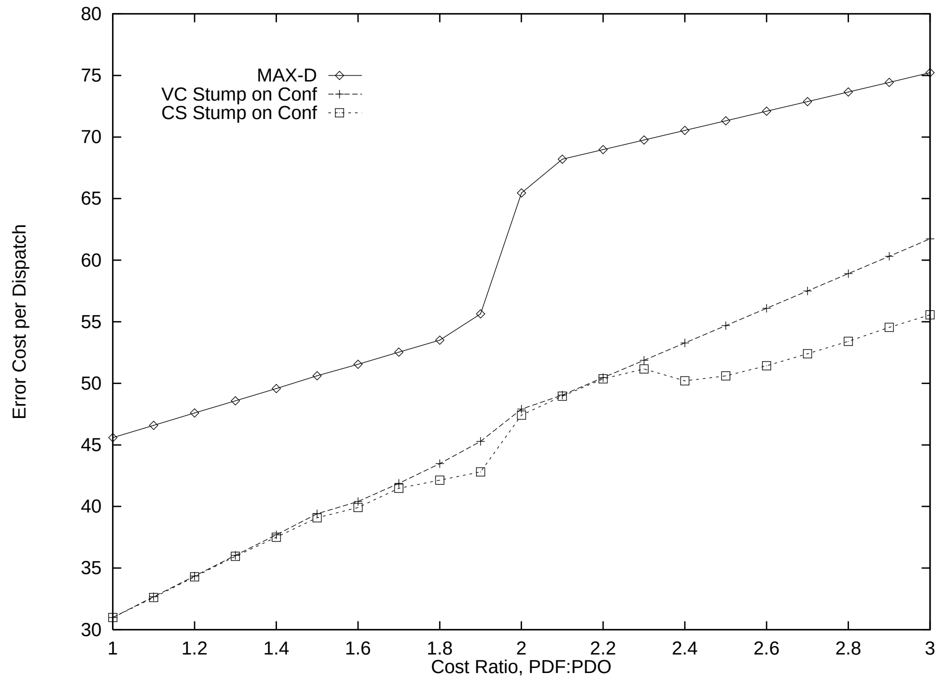

Figure 3: Effect of Varying Error Cost Ratios on Cost of Errors Made. Data are restricted to confidently dispatched cases.

real-world machine learning and data mining applications. We have found some to echo lessons learned in other applications work (Kohavi & Provost 1998).

Lesson 1: A single source of data gives a narrow view of the problem. In order to get the complete, well-rounded picture necessary to present a compelling argument for the real-world use of this technology, we found it necessary to use multiple data sources and to perform data cleaning based on domain knowledge.

Lesson 2: Superficial use of accuracy figures gives a shallow view of the problem. Be careful not to fall into the trap of ignoring the accuracy of simple methods. All too often the inexperienced data miner is elated by the seemingly good performance of his (or her) favorite learning program, only to discover that a simple method (e.g., a linear discriminant function or simple Bayes) works just as well, or more embarrassingly, that the class distribution is highly skewed. Hopefully, such a discovery is made before the results are presented to someone for whom retraction would be an embarrassment. This paper shows a case where a simple method (decision stump) actually outperforms a more complex decision-tree learner (as well as other complex learning programs).

Also, one should be careful not to compare incomparable accuracy figures. The most obvious reason why inter-study accuracy comparisons would not be valid is that the different data sets have different class distributions. We saw evidence of this in the section on data cleaning; the different class distributions were due to the fact that cleaning the data both eliminated classes from the data and removed troubles non-uniformly across the remaining classes. For our inter-study comparisons, we used the percentage decrease in error rate for each metric over the error rate of the default class.

Lesson 3: Broad comparison studies increase the confidence in the evaluation. When arguing for the use of inductive learning technology, the last thing you want is to be blindsided by questions like “How does it compare with the (currently used) FooBar system,” or “Well ... Brooklyn is a special case, have you tried data from Upstate?” We were lucky to have considerable management and peer support and enthusiasm, which is certainly not universal in real-world applications of emerging technologies. In addition to the use of multiple data bases, and multiple learning methods described in Lessons 1 and 2, we also found it necessary to produce multiple “existing” methods with which to compare, including the existing expert system, as well as the experts themselves. Furthermore, we found it necessary to collect data from geographically disparate locations to demonstrate the robustness of the learning.

Lesson 4: Don't lose sight of the real performance task. In real-world domains, accuracy is seldom the bottom line. More often, cost-effectiveness is. In many domains, different errors have different associated costs, so it is important that the learned knowledge produce the right trade-offs. In this domain, not only did we find that an analysis of the cost-effectiveness of the learned classifiers is essential, we also discovered that merely paying lip service to cost-effectiveness with a naive cost analysis is not sufficient. We had to bring in techniques from decision analysis (that are seldom even mentioned in machine learning/data mining research—more below).

7.2 Implications for Inductive Learning Research

We believe that studies such as this of the actual use of inductive learning, in a practical application where there is high-level support for the use of AI technologies (and therefore complexity does not come from a distrust of the techniques), should be a guiding influence to the research community. Therefore, let us discuss briefly the type of research results that would have been helpful to this effort.

Dealing with potentially erroneous data was a major issue. Most of the machine learning/data mining literature on learning in the presence of noise discusses random errors. However, the types of errors most often discussed in this domain (and many real-world domains) usually have some degree of systematicity—systematicity that may also appear in the evaluation data. Furthermore, we have not performed a detailed analysis of the effect of data cleaning. For example, we eliminated from the data borderline cases and cases for which we could not determine the correct resolution. How will this affect our evaluation? We believe that learning in the presence of possible systematic errors is a very interesting open problem (Weiss 1995, Beers 1957, Lee 1995).

The machine learning community has spent the last decade comparing learning programs on suites of benchmark problems *ad nauseam*. However, almost all evaluations have been based on classification accuracy, the result being a host of available systems that can maximize accuracy within their inductive biases. We believe that unless the accuracy is 100%, very few real-world domains use classification accuracy as the prime evaluation criterion. In fact, evaluations based on classification accuracy can be quite misleading (Saitta & Neri 1998, Provost et al. 1998).

When we were faced with the prospect of learning with sensitivity to the cost of errors, we found ourselves with only a handful of small-scale, comparatively inconclusive studies in the machine learning literature. We believe it is time for machine learning and data mining research to take off the blinders of classification accuracy and develop robust methods that can provide the cost-effective classification needed in the real world. Interested researchers can begin by referencing work in statistics (Duda & Hart 1973), decision analysis (Henrion et al. 1991, Keeney 1982, Weinstein & Fineberg 1980), and pattern recognition (Dattatreya & Kanal 1985). Also, Turney (1996) provides an on-line bibliography of work on cost-sensitive machine learning.

Acknowledgements

Tom Fawcett helped in developing new mappings from field technicians' resolution codes to dispatches. Kim Tabtiang performed the runs reported in Sections 3, 4, and 5. Peter Turney and Irene Po initiated the discussion of using a decision-analytic cost matrix, and built the first prototype. We would like to thank the following colleagues for valuable discussions over the course of the Machine Learning for MAX project: Charlie Bove, Bruce Buchanan, Jim Euchner, Tom Fawcett, Neil Melley, Mike Pazzani, Irene Po, Henry Rabinowitz, Jude Shavlik, Judy Spitz, Peter Turney, Rikki Wolin, and Yuling Wu.

References

[1] Beers Y. (1957) *An Introduction to the Theory of Error.* Addison-Wesley Publishing Company, Reading, MA.

[2] Bradford J., Kunz C., Kohavi R., Brunk C. & Brodley C. (1998) Pruning decision trees with

misclassification costs. *Proceedings of ECML-98*, p. 131-136.

[3] Danyluk A. P. & Provost F. J. (1993a) Small Disjuncts in Action: Learning to Diagnose Errors in the Local Loop of the Telephone Network. *Proceedings of the Tenth International Conference on Machine Learning*, p. 81-88. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

[4] Danyluk A. P. & Provost F. J. (1993b) Adaptive Expert Systems: Applying Machine Learning to NYNEX MAX. *Working Notes of the AAAI-93 Workshop on AI in Service and Support: Bridging the Gap Between Research and Applications*, Washington, DC, p. 50-58.

[5] Dattatreya G. R. & Kanal L. N. (1985) Decision Trees in Pattern Recognition. In L. N. Kanal and A. Rosenfeld (ed.), *Progress in Pattern Recognition 2*, p. 189-239. Elsevier Science Publishers B.V. (North-Holland).

[6] Duda R. O. & Hart P. E. (1973) *Pattern Classification and Scene Analysis.* John Wiley, New York.

[7] Henrion M., Breese J. S. & Horowitz E. J. (1991) Decision Analysis and Expert Systems. *AI Magazine*, 12, p. 64-91.

[8] Holte R. C. (1993) Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, *Machine Learning*, 11, 1, p. 63-90.

[9] Keeney R. L. (1982) Decision Analysis: An Overview. *Operations Research*, 30, p. 803-838.

[10] Kohavi R. & Provost F. (1998) *Special Issue on Applications of Machine Learning and the Knowledge Discovery Process*, Machine Learning, 30, 2/3.

[11] Lee Y. (1995) *Learning a Robust Rule Set.* Ph.D. Thesis. Department of Computer Science, University of Pittsburgh.

[12] Merz C. J., Pazzani M. & Danyluk A. P. (1996) Tuning Numeric Parameters to Troubleshoot a Telephone Network Local Loop. *IEEE Expert*, 11, 1, p. 44-49.

[13] Pazzani M. J. & Brunk C. (1993) Finding Accurate Frontiers: A Knowledge-Intensive Approach to Relational Learning. *Proceedings of AAAI-93*, p. 328-334, AAAI Press, Menlo Park, CA.

[14] Pazzani M., Merz C., Murphy P., Ali K., Hume T. & Brunk C. (1994) Reducing Misclassification Costs. *Machine Learning: Proceedings of the Eleventh International Conference*, p. 217-225. Morgan Kaufmann, San Mateo, CA.

[15] Provost F. (1994) Goal-Directed Inductive Learning: Trading off Accuracy for Reduced Error Cost. *Working Notes of the AAAI-94 Workshop on Goal-Driven Learning*, p. 94-101.

[16] Provost F. & Aronis J. (1996) Scaling Up Inductive Learning with Massive Parallelism. *Machine Learning*, 23, p. 33-46.

[17] Provost F., Fawcett T. & Kohavi, R. (1998) The Case Against Accuracy Estimation for Comparing Induction Algorithms. *Machine Learning: Proceedings of the Fifteenth International Conference*, p. 445-453. Morgan Kaufmann Publishers, San Francisco, CA.

[18] Quinlan J. R. (1993) *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, Inc., San Mateo, CA.

[19] Rabinowitz H., Flamholz J., Wolin E. & Euchner J. (1991) NYNEX MAX: A Telephone Trouble Screening Expert. In R. Smith and C. Scott (ed.), *Innovative Applications of AI 3*, p. 213-230, AAAI Press, Menlo Park, CA.

[20] Saitta L. & Neri F. (1998) Learning in the "Real World", *Machine Learning*, 30, 133-164.

[21] Turney P. (1996) Cost-sensitive learning bibliography.

http://ai.iit.nrc.ca/bibliographies/cost-sensitive.html.

[22] Weinstein M. C. & Fineberg H. V. (1980) *Clinical Decision Analysis.* W.B. Saunders Company, Philadelphia, PA.

[23] Weiss G. M. (1995) Learning with Rare Cases and Small Disjuncts. *Proceedings of the Twelfth International Conference on Machine Learning*, p. 558-565, Morgan Kaufmann, San Francisco, CA.