Hindawi Publishing Corporation
Advances in Artificial Intelligence
Volume 2011, Article ID 587285, 12 pages
doi:10.1155/2011/587285

Research Article

Tuning Expert Systems for Cost-Sensitive Decisions

Atish P. Sinha and Huimin Zhao

Sheldon B. Lubar School of Business, University of Wisconsin-Milwaukee, P.O. Box 742, Milwaukee, WI 53201-0742, USA

Correspondence should be addressed to Huimin Zhao, hzhao@uwm.edu

Received 15 December 2010; Accepted 22 March 2011

Academic Editor: Filip Zelezny

There is currently a growing body of research examining the effects of the fusion of domain knowledge and data mining. This paper examines the impact of such fusion in a novel way by applying validation techniques and training data to enhance the performance of knowledge-based expert systems. We present an algorithm for tuning an expert system to minimize the expected misclassification cost. The algorithm employs data reserved for training data mining models to determine the decision cutoff of the expert system, in terms of the certainty factor of a prediction, for optimal performance. We evaluate the proposed algorithm and find that tuning the expert system results in significantly lower costs. Our approach could be extended to enhance the performance of any intelligent or knowledge system that makes cost-sensitive business decisions.

1. Introduction

Expert systems have been used to solve different types of problems, such as interpretation, prediction, design, planning, monitoring, diagnosis, debugging, repair, and control [1]. They have been applied to several domains, such as medicine, geology, chemistry, engineering, computer systems, auditing, marketing, sales, and finance [1, 2]. Business applications for which expert systems have been developed include financial analysis, credit rating, sales, foreign exchange trading, trade advising, tax planning, auditing, and labor management [3, 4].

Over the years, *prediction* expert systems have witnessed widespread use in finance and related domains. Those systems use stored knowledge acquired from financial experts to make predictions on stock prices, bankruptcy, credit worthiness, financial markets, and so forth. Prediction problems can be classified into two types: *classification* and *regression*. Expert systems used for a classification problem classify the outcome of a problem into one out of multiple categories. In the finance domain, the focus has been mainly on the binary classification problem, which involves classifying the outcome into one of two possible categories (e.g., good loan or bad loan, high or low risk of insolvency, etc.).

The measure that has been commonly used for evaluating the effectiveness of financial expert systems is *classification accuracy*. Using this measure implies that the lower the number of errors a system makes, the better it is. However, in many financial problems, such as credit evaluation and bankruptcy prediction, the costs for different types of misclassification errors are not the same [5–7]. Hence, the accuracy measure, which assumes equal costs, is not a valid criterion to use for such problems, where the focus should be on minimizing the overall *misclassification cost* [8, 9].

In a binary classification problem, a specific instance or *case* can belong to one of two classes: *positive* or *negative*. Tests are conducted to detect if a case is positive (e.g., bankrupt firm, bad credit, etc.) or negative. A *false positive* results when the test results are positive but the case actually belongs to the negative class. A *false negative* results when the test is negative but the case actually belongs to the positive class. For classification problems having unequal costs associated with the two types of errors, the focus should be on minimizing the overall misclassification cost. It is important for an expert system to correctly identify true positives and true negatives, instead of simply finding the total number of correct or wrong answers.

Data mining and knowledge engineering have remained largely independent fields, although, of late, there has been a movement toward examining the fusion of the two [10]. Researchers, (e.g., [7, 9, 11]) have studied the impact of incorporating some form of domain knowledge—embedded in an expert system or acquired from a human expert—into the data mining process. Machine learning algorithms typically used in data mining have been applied to learn rules for an expert system based on examples provided by experts [12]. Our study examines the impact of the fusion in a novel way by applying validation techniques and data employed for data mining to enhance the performance of knowledge-based expert systems.

In this paper, we present an approach for enhancing the performance of a knowledge-based expert system by applying a data mining cross-validation technique. We propose an algorithm that applies the expert system to classify cases in the training sample, estimates the expected misclassification cost under a range of certainty factor cutoffs, and determines the optimal cutoff with respect to expected misclassification cost. We evaluate the efficacy of the approach by comparing the performance of the tuned expert system with that of the original, untuned version. We also generate a receiver operating characteristic (ROC) curve to visually depict the performance of the expert system classifiers, tuned using a range of decision thresholds. The evaluation yields promising results; tuning the expert system results in significantly lower costs.

With the growing influence of business intelligence applications in management decisions, our study makes an important contribution by presenting an approach that could be applied to any knowledge system dealing with uncertainty. While the certainty factor algebra is germane to expert systems, the algorithm that we have developed could be applied to tune any intelligent or knowledge system that generates uncertain conclusions and operates under unequal misclassification costs.

The paper is organized as follows. Section 2 presents the background. Section 3 describes the problem domain. Sections 4 and 5 describe the development of an expert system, and the computation of certainty factors by the expert system. In Section 6 we present the proposed approach for enhancing the performance of an expert system and in Section 7 we present the results of an empirical evaluation. We discuss the results of the study in Section 8 and finally conclude the paper in Section 9.

2. Background

In the 1970s, artificial intelligence (AI) scientists focused on techniques such as *representation* and *search* to make computer programs intelligent. In the late 1970s, AI researchers made a major breakthrough by incorporating high-quality, specific knowledge of a given problem area into computer programs. Those programs, called *expert systems*, emulate human experts in a narrow problem domain by storing and applying their knowledge.

The process of building an expert system is known as *knowledge engineering*. It involves a knowledge engineer eliciting procedures, strategies, and rules of thumb from a domain expert and transferring this heuristic knowledge into a computer program [1]. The resulting expert system solves problems in much the same way as the expert does, by using short-cuts and tricks and ignoring irrelevant information. It uses the stored knowledge to achieve high performance.

Human experts develop their knowledge through years of experience in solving problems in a narrow area. Because experts usually find it difficult to articulate the heuristics or rules of thumb that they apply, knowledge engineers need to overcome what is known as the *knowledge acquisition bottleneck*. The more competent human experts become, the less able they are in describing the knowledge they use to solve problems [13]. Several knowledge acquisition techniques—such as interviews, protocol analysis, observation, and focus groups—are available for facilitating knowledge transfer from experts.

The expert system development life cycle includes the following phases: identification, conceptualization, formalization, implementation, and testing [1]. In the *identification* phase, the type and scope of the problem, along with the required resources, are identified. In the *conceptualization* phase, the knowledge engineer and the domain expert conceptualize the problem in terms of the concepts, relations, control mechanisms, subtasks, and strategies related to the problem-solving activity. In the *formalization* phase, those concepts are formalized within the framework of an expert system development tool, selected based on the knowledge representation technique considered to be most appropriate for the problem. *If-Then* rules are the most popular means for representing knowledge. In the *implementation* phase, the formalized knowledge is translated into an operational computer program. Finally, in the *testing* phase, the performance and utility of the expert system are evaluated.

Testing involves both *verification* and *validation*. While verification is defined as building the *system right*, validation is defined as building the *right system* [14]. Verification entails checking the knowledge base for logical completeness and consistency, as well as addressing engineering issues, such as efficiency, maintainability, portability, and reliability [15]. In contrast, validation focuses on evaluating the expert system with respect to performance (e.g., accuracy, correctness of reasoning, and Turing test) as well as usage (e.g., usability, dialog, and explanation quality). But the prerequisite for extensive usage-oriented validation is an acceptable level of performance by the expert system [15].

Validation has usually focused on assessing an expert system's performance relative to some acceptable performance level, such as a given level of accuracy. A typical validation session entails running a set of carefully chosen test cases on the expert system and having domain experts validate the results with respect to accuracy. Other criteria, such as explanation capability, usability, and depth, breadth, and correctness of reasoning, have also been used. But the focus of validation has been on "accurate performance" [15].

Research in expert system validation has largely ignored the issue of costs and benefits even though it was identified as a major concern long ago [16]. In estimating costs, one needs to factor in the uncertainty inherent in expert system reasoning, represented as certainty factors of rule conclusions. Recognizing the fact that there has been no systematic treatment of weights for rules in management expert systems, O'Leary [17] provides an empirical approach for verifying knowledge bases with certainty factors. So far, however, a similar approach for validating uncertainty in expert systems has not been proposed.

Researchers in the data mining field have started addressing the issue of unequal costs. Based on his investigation into the learning of Bayesian and decision tree classifiers under different misclassification costs for a two-class problem, Elkan [18] recommends that classifiers learn from the training data as given, and then determine the optimal decision thresholds empirically. Sinha and May [8] analyze the performance of five popular data mining methods on a binary credit classification problem. Instead of classification accuracy, they employ misclassification cost as a basis for comparison. They also propose a method for tuning the models by empirically determining optimal decision thresholds under a set of costs and prior probabilities. Use of the proposed tuning method resulted in improved performance for all the methods examined in their study. In a recent study, Bansal et al. [19] extend the idea to tune regular regression models for lowering misprediction costs.

ROC curves, which were originally used in signal detection theory, are being increasingly used in data mining research for analyzing classifier performance [8, 20–22]. ROC curves are useful for visualizing a classifier's performance [23], especially vis à vis other classifiers. Provost et al. [21], for example, use ROC analysis to investigate whether a dominating model exists in ROC space; if such a model does not exist, none of the models under investigation can be considered to be best under all target scenarios. Sinha and May [8] use ROC analysis to compare the performance of models built using different methods and use the ROC results to determine optimal classifiers.

The focus of an expert system is on the *knowledge* acquired from a human expert in a given problem domain. In contrast, data mining methods do not use domain knowledge but try to learn new patterns from the available *data*. While expert systems are *knowledge driven*, data mining systems are *data driven*. Expert systems and data mining, therefore, are radically different in their approaches, and that may be the reason why they have remained largely independent fields. Lately, however, upon the realization that the two approaches could play complementary roles, there has been a growing interest in examining the effects of their fusion. Dybowski et al. [10] stress that in domains where both data and expertise are available, the fusion of domain knowledge and data mining is an interesting issue. Studies, (e.g. [9, 24–27]), that incorporated domain knowledge in some fashion into the data mining process have produced encouraging results. Decision tree learning algorithms have been successfully used in inducing rules from expert-provided examples for building expert systems [12].

3. Problem Domain

The problem domain for this study is that of bank loans for automobiles. Evaluation and approval of loan applications is becoming even more critical for banks in the midst of the current financial crisis. Increasing loan losses in the third quarter of 2008 left credit unions barely in the black [28]. Nationally, with mounting loan losses, the number of problem banks on the FDIC's official list reached, in November 2008, 171, the most since 1995 [29].

Two types of attributes are present in a loan application: application attributes and credit bureau characteristics. The application attributes relate to the items on the loan application itself, such as years in current residence, years in previous residence, monthly income, and monthly payments. However, other major attributes for making lending decisions are not available in the application; they come from the credit bureau report. The credit bureau characteristics include attributes such as the months on file, number of satisfactory trades, number of major trades, and number of minor trades.

The *months on file* attribute represents the total number of months the credit bureau has maintained a record on the applicant. Typically, the longer the number of months on file, the more credit history the applicant has. In a credit report, instances of credit are represented in a trade line, which stores the vital statistics of each instance of borrowing. A new trade line is opened every time the borrower obtains a different loan or credit card.

Trades that have a sufficient payment history are rated on a scale of 1 to 9. Those trades for which all payments were received in a timely manner (i.e., none of them was received more than 29 days past the due date) are called *satisfactory* trades and receive a rating of 1, the best possible rating. If one of the payments for a loan was late and the late time period was between 29 and 60 days (exclusive), the loan would be classified as *minor* with a rating of 2. If any of the payments was received 60 days or more past the due date, the loan is classified as *major* with a rating of 3 or more, depending on how bad the payment history is. A rating of 9 indicates that the loan was "charged off" and was considered uncollectible by the creditor. The *worst rating* attribute indicates the worst rating among all the trade lines associated with the borrower.

4. Expert System Development

The loan expert system was developed using a backward-chaining expert system shell. The expert system has close to 100 rules, structured in the following format (see Figure 1).

The *If* part of the rule is called the *premise* of the rule. The *Then* part is called the rule *conclusion*. In a backward-chaining system, the inference engine works backward from the conclusion. It hypothesizes a conclusion and works backward toward the hypothesis-supporting facts. To satisfy a hypothesis, it checks if all the conditions in the premise hold true. A condition may be supported by a fact or input provided to the expert system; if not, the inference engine treats that condition as an intermediate hypothesis, and checks the knowledge base to determine if there is any other

```
If      condition_1
        condition_2
        ........
        condition_n
Then    conclusion
```

Figure 1

```
Rule-4:   If    number of minor trades >= 3 and
                number of minor trades <= 6
          Then  credit rating = fair cf .60
Rule-24:  If    number of major trades = 0 and
                number of minor trades = 0 and
                number of satisfactory trades > 10 and
                bankruptcy = no
          Then  credit rating = excellent cf .99
```

Figure 2

```
Rule-38:  If    credit rating = fair and
                years in current residence > 1 and
                years employed by current employer <= 2 and
                debt-to-income ratio > .36
          Then  recommendation = deny loan cf .90
Rule-71:  If    credit rating = excellent and
                debt-to-income ratio <= .36 and
                years employed by current employer >= 2 and
          Then  recommendation = approve loan cf .90
```

Figure 3

rule that can help support the hypothesis. If it finds one, then the conclusion of that rule becomes the new hypothesis to prove and the same process is repeated. In this way, the inference engine builds a chain of goals and subgoals, starting with the main goal or hypothesis and working backward. If after exploring all possible avenues, a hypothesis cannot be supported, then the rule fails.

The expert used for developing the expert system was a loan officer with over 15 years of experience and a strong reputation when it came to loan charge-offs. A knowledge engineer interviewed the expert several times to acquire his knowledge. At the beginning, unstructured interviews were used to develop an initial understanding of the domain. A generic loan application was used to identify the main steps and factors involved in the decision. The initial set of interviews was followed with more specific, structured interviews which elicited detailed knowledge of how lending decisions are made.

When the expert reviews a loan application, the first thing he considers is the credit report. His assessment of the credit history is based upon attributes such as the number of trades rated as satisfactory, the number of major derogatory trades, the number of minor derogatory trades, and the months the applicant has been on file. Provided below are two examples of rules acquired from the expert for rating the applicant's credit history (see Figure 2).

Rule-24 states that if the applicant had no major or minor derogatory trades, had more than 10 satisfactory trades, and was never declared bankrupt, then her credit rating is excellent. A certainty factor (cf) is associated with the rule, reflecting the expert's confidence in the conclusion (see Section 5 for a detailed discussion of certainty factors).

After rating the applicant's credit, the expert reviews attributes on the loan application itself, such as years in current and previous residence, years with current and previous employer, monthly income, monthly payments, price of the car, down payment, and so forth, and makes a decision on whether the loan should be approved or denied. Two examples of rules for making loan decisions are given below (see Figure 3).

A consultation with the loan expert system starts with the goal to determine the value for *recommendation*, which could be "approve" or "deny." Since the recommendation is not available as a fact in the knowledge base, the inference engine tries to find rules that have "recommendation" in the conclusion.

Suppose rule-38 is the rule it finds first. It checks and finds that there are no facts currently in the knowledge base to satisfy the first condition: credit rating. Because the value of credit rating is not known, the inference engine creates "credit rating" as a subgoal and tries to find rules that could help achieve that subgoal. Suppose it finds rule-4. If the number of minor trades is between 3 and 6, the rule fires concluding that credit rating is fair. Control then returns to rule-38, which then checks if the remaining conditions in its premise are true or not. If rule-4 had failed to match the facts, rule-38 would have called other rules that could conclude a value of "fair" for credit rating.

5. Certainty Factors

The most widely used approach for representing uncertainty in expert systems is that of certainty factors (CFs). The MYCIN system [30], which was developed in the mid 1970s to help physicians diagnose and treat infectious blood diseases, pioneered the use of CFs. The uncertainty in expert systems stems from two sources: (1) the uncertainty of a fact or hypothesis that experts need to reason about, and (2) the uncertainty in the validity of inference rules used by experts [31]. The CF associated with a hypothesis, or an intermediate hypothesis, represents the degree of belief based on all the evidence that has been used so far. The CF for a rule represents the degree of belief in the rule's conclusion based on its premise.

Consider rule-38 from the auto loan expert system again. The rule states that if all the conditions in the premise hold, then the expert's recommendation is to deny the loan, but he is 90% certain, not absolutely certain, of the conclusion. This is an example of the second type of uncertainty discussed above. An example of the first type of uncertainty is:

$$\text{credit rating} = \text{fair cf } .70 \tag{1}$$

which represents the belief that the credit rating of the applicant is fair, but only to a degree of 70%. Note that though credit rating is part of the premise of rule-38, it could be a hypothesis (conclusion) of another rule, such as the one shown below (see Figure 4).

This rule states that if the number of minor trades that the applicant has had is between 3 and 6, then the expert is

Rule-4: If number of minor trades >= 3 and
number of minor trades <= 6
Then credit rating = fair cf .60

Figure 4

only 60% sure that the credit rating is fair. Note that when this rule fires, its conclusion satisfies the first condition of rule-38. That is typical of a rule-based expert system in which when a rule fires, its conclusion becomes the evidence for the next one in the chain.

The evidence for a rule does not necessarily have to be a conclusion derived from another rule. It could simply be a fact or situation known to the expert system. During a consultation session, the loan expert system uses as inputs values for variables such as years in residence, years employed, number of minor trades, and debt-to-income ratio. If the credit bureau report states that the number of minor trades the applicant has had is 5, then the expert system uses the following fact as an input:

$$\text{minor trades} = 5. \tag{2}$$

When no CF is associated with a fact or situation, the expert system employs the default certainty of 100%. That is, the expert system interprets the input as:

$$\text{minor trades} = 5 \text{ cf } 1.00. \tag{3}$$

A consultation with the expert system is geared toward determining the values of the goals (hypotheses) of the consultation. The final goal is to find the recommendation for the loan application, which is whether the loan should be approved or denied. To reach that goal, the expert system has to first achieve an intermediate goal, which is to determine the credit rating of the applicant.

As new evidence comes in, the CFs are used to update the beliefs in the hypotheses. Given that uncertainty can arise from the two sources identified above, the confidence in the conclusion generated from a rule's firing is a combination of the rule's CF and the certainty of the rule's premise.

Let $P(H)$ represent the a priori or unconditional probability of hypothesis H. Based on accumulating evidence, $P(H)$ can increase or decrease across a range of values from 0 to 1. In the context of expert systems, $P(H)$ can be interpreted as the subjective degree of belief in H. With the arrival of new evidence E, the belief in H is updated. The conditional probability $P(H \mid E)$—that is, the probability that H is true given that E is true—represents the updated belief. $P(H \mid E)$ greater than $P(H)$ implies that E increases the expert's belief in H while $P(H \mid E)$ less than $P(H)$ implies that E decreases the belief.

The certainty factor approach is grounded in three measures of uncertainty: (1) MB(H, E), the increased belief in H based on E, (2) MD(H, E), the increased disbelief in H based on E, and (3) CF(H, E), the combination of MB and MD into a single certainty measure [31]. The definitions of these measures, from Stefik [31], are reproduced below (see Figure 5).

If $P(H) = 1$, MB(H, E) = 1,
else MB(H, E) = [max[$P(H \mid E)$, $P(H)$] – $P(H)$] / [1 – $P(H)$].
If $P(H) = 0$, MD(H, E) = 1,
else MD(H, E) = [min[$P(H \mid E)$, $P(H)$] – $P(H)$] / [0 – $P(H)$].
CF(H, E) = MB(H, E) – MD(H, E)

Figure 5

If both CF_1, $CF_2 > 0$,
Then CF-noted = $CF_1 + CF_2 * (1 - CF_1)$
If both CF_1, $CF_2 < 0$,
Then CF-noted = $CF_1 + CF_2 * (1 + CF_1)$
If $CF_1 < 0 < CF_2$,
Then CF-noted = $(CF_1 + CF_2) / [1 - \min(|CF_1|, |CF_2|)]$

Figure 6

From the above definitions, we find that the values of MB and MD range from 0 to 1, while those for CF range from −1 to 1. When H is absolutely certain, MB = 1, MD = 0, and CF = 1. When $P(H) = 0$, MB = 0, MD = 1, and CF = −1. A CF of 1 indicates absolute certainty in the validity of a fact or rule, while a CF of −1 indicates that a fact or hypothesis is definitely wrong. A CF of 0 can be interpreted as a complete lack of evidence in a fact or hypothesis.

If the a priori belief in the hypothesis is small, that is, $P(H) \approx 0$, then $CF(H, E) \approx P(H \mid E)$. The CF of a hypothesis confirmed by evidence can therefore be roughly interpreted as the conditional probability based on that evidence.

Evidence for or against a hypothesis may come from multiple sources. Suppose E_1 and E_2 are two pieces of evidence that confirm or reject hypothesis H. Let $CF_1 = CF(H, E_1)$ and $CF_2 = CF(H, E_2)$. The expert system combines the two pieces of evidence in the following way (see Figure 6) where CF-noted is the certainty factor stored for the hypothesis in the cache. The cache is a repository of all conclusions made during a consultation with the expert system.

If the premise of a rule is uncertain, the expert system factors in that uncertainty into the uncertainty of the rule (CF_{rule}) by taking the product of the two uncertainties, that is,

$$\text{CF-noted} = CF_{rule} * CF_{premise}. \tag{4}$$

If the premise itself is a conjunction of conditions—that is, AND conditions—then the certainty of the conjunction noted is the minimum of the CFs, that is,

$$\text{CF-noted} = \min(CF_1, CF_2, \ldots, CF_n). \tag{5}$$

In rule-38, for example, if credit rating = fair cf .70 and all the other conditions are true with absolute certainty, then the certainty for the conclusion (deny loan) is

$$\text{CF-noted} = .90 * \min(.70, 1, 1, 1) = .63. \tag{6}$$

On the other hand, if the premise is a disjunction of conditions—that is, OR conditions—then the rule fires multiple times, once for each condition, and the CF of the

conclusion is computed based on the equations shown above for combining evidence.

As in MYCIN, our expert system uses a threshold CF of .20 for a rule to fire. That means that if $\mathrm{CF}_{\mathrm{premise}} < 20$, the rule fails. Having such a threshold helps the inference engine avoid pursuing hypotheses that are not likely to be true.

It is important to note the following points regarding the computation of CF. The final CF for a hypothesis is independent of the order in which evidence is found. As positive evidence accumulates, the resulting CF approaches but cannot exceed 1. Similarly, with mounting negative evidence, the resulting CF approaches but cannot go lower than -1. Once the certainty of a conclusion reaches 1 or -1, it cannot be changed by additional incoming evidence. If 0 is combined with any CF, it leaves the CF unchanged.

6. Enhancing the Performance of an Expert System

In this section, we describe the proposed approach for enhancing the performance of an expert system. We first define the performance measure used to evaluate the performance of an expert system. We next present an algorithm for tuning an expert system for optimal performance.

6.1. Performance Measurement. In general, a binary classification problem is described by a pair $\langle X, Y\rangle$, where $X = X_1 \times X_2 \times \cdots \times X_m$ is the domain of an m-dimensional vector of variables $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\rangle$, called *features* or *independent variables*, and $Y = \{0, 1\}$ is the domain of a binary variable $\mathbf{y}$, called *class* or *dependent variable*. An expert system built for solving such a classification problem essentially provides a mapping, $f : X \rightarrow Y$, consisting of a set of mapping rules for predicting the value of $\mathbf{y}$ based on the value of $\mathbf{x}$.

The prediction made by an expert system is subject to two types of errors. A false positive error is made when the expert system misclassifies an actual negative case as positive (i.e., $\mathbf{y} = 0$, $f(\mathbf{x}) = 1$). A false negative error occurs when the expert system misclassifies an actual positive case as a negative (i.e., $\mathbf{y} = 1$, $f(\mathbf{x}) = 0$). The performance of an expert system f can be measured by its *expected misclassification cost*, which is defined as

$$C(f) = C_{01}\, p\, p_{01} + C_{10}(1 - p)\, p_{10}, \tag{7}$$

where $p = \Pr[\mathrm{y} = 1]$ is the prior probability of a positive case, $(1 - p)$ is the prior probability of a negative case, $p_{01} = \Pr[f(\mathrm{x}) = 0 | \mathrm{y} = 1]$ and $p_{10} = \Pr[f(\mathrm{x}) = 1 | \mathrm{y} = 0]$ are false negative and false positive error rates, respectively, and C_{01} and C_{10} are the unit costs of false negative and false positive errors, respectively. Since the assignment of the unit costs of the two types of errors is often subjective, the ratio between the two costs is more interesting than their absolute values. Let $r = C_{01}/C_{10}$ denote the cost ratio of false negative to false positive, an adjusted measure for the expected misclassification cost can be defined, by setting $C_{10}(1 - p)$ constantly at one, as

$$C'(f) = \frac{C(f)}{C_{10}(1 - p)} = \frac{rp}{(1 - p)}\, p_{01} + p_{10}. \tag{8}$$

Table 1: Confusion matrix of an expert system on a sample of solved cases.

		Predicted class		
		1	0	Sum
Actual class	1	n_{11}	n_{01}	n_1
	0	n_{10}	n_{00}	n_0
				n

An advantage of using this adjusted measure is that it is invariant with regard to different settings of p and r as long as the weight ratio between the two classes, $rp/(1 - p)$, holds constant. For example, the two settings, $p = 0.2$, $r = 4$ and $p = 0.5$, $r = 1$, yield the same performance measure. The conditional probabilities, p_{01} and p_{10}, can be estimated by the corresponding sample proportions based on a sample of previously *solved cases* (also referred to as *instances* or *examples*), each of which is a pair $\langle x, y\rangle$, where $x = \langle x_1, x_2, \ldots, x_m\rangle \in X$ and $y \in Y$. A confusion matrix (a template is shown in Table 1) can be constructed based on the classification outcomes of the expert system on the provided sample. Let n_1 (n_0) denote the number of positive (negative) cases in the provided sample, and n_{01} (n_{10}) denote the number of positive (negative) cases misclassified as negative (positive). Then the conditional probabilities, p_{01} and p_{10}, can be estimated as

$$\hat{p}_{01} = \frac{n_{01}}{n_1}, \qquad \hat{p}_{10} = \frac{n_{10}}{n_0}. \tag{9}$$

The expected misclassification cost of the expert system can therefore be estimated based on the provided sample as

$$\hat{C}(f) = \frac{rp}{(1 - p)}\hat{p}_{01} + \hat{p}_{10} = \frac{rp}{(1 - p)}\frac{n_{01}}{n_1} + \frac{n_{10}}{n_0}. \tag{10}$$

The prior probability p can be estimated by the proportion of positive cases in a representative sample; otherwise, it could be (subjectively) estimated by domain experts. The domain experts should also be able to assign a value for cost ratio r based on their experience.

Note that classification error rate (or inversely, accuracy), a measure that has been frequently employed in the past, is a special case of expected misclassification cost under the situation where the two prior probabilities, as well as the two types of misclassification costs, are equal. However, these assumptions are not realistic in domains such as credit evaluation. First, for a profitable bank, the prior probability of a bad loan is usually much lower than that of a good loan. Second, the cost of misclassifying a bad loan as good (false negative) is much more than that of misclassifying a good loan as bad (false positive). Hence, expected misclassification cost is a more appropriate performance measure to use than overall accuracy for cost-sensitive decisions, such as credit evaluation [18, 21, 32].

6.2. Performance Tuning. The prediction of an expert system typically involves uncertainty. As we discussed earlier, the most widely used technique for representing uncertainty in expert systems is the certainty factor approach. The CF of

a prediction made by an expert system ranges from -1 to 1, where 1 indicates that a hypothesis is certainly true, -1 indicates that a hypothesis is certainly false, and 0 indicates a complete lack of evidence for or against the validity of the hypothesis. In binary classification problems, since the CF of a negative prediction is the same as that of a positive prediction, with only the sign reversed, we will henceforth use CF to refer to only the certainty factor of a positive prediction.

For the credit evaluation problem, a prediction is considered to be positive when the loan is considered to be bad and denied. When the expert system denies a loan, the prediction is positive and the CF is equal to the certainty factor computed by the system. When the expert system approves a loan, the CF is equal to the certainty associated with the approval decision, with the sign reversed, because we use CF to refer only to a positive (deny) prediction. If the expert system gives multiple predictions, the certainty factors provided by the expert system along with the predictions are consolidated into a single CF using the method previously described in Section 6. When the CF is greater than or equal to 0, the loan is denied; otherwise, when the CF is negative, the loan is approved. The default threshold for approving a loan is therefore 0. The default cutoff of 0 may be reasonable for cost-insensitive classification problems—where the two types of misclassification errors are equally costly and the prior probabilities of the two classes are equal—but may not produce optimal performance with respect to expected misclassification cost for cost-sensitive problems.

The cost-sensitive performance of an expert system can be visualized using the ROC curve, which has been widely used with learned classification systems [20]. Given a sample of solved cases and a selected cutoff, a confusion matrix can be constructed in the form of Table 1. Based on the confusion matrix, we define *sensitivity* as being equal to $n_{11}/n_{11} + n_{01}$, which is the true positive rate, and *specificity* as being equal to $n_{00}/n_{00} + n_{10}$, the complement of the false positive rate. A series of sensitivity and specificity values can be obtained by varying the cutoff. An ROC curve plots *sensitivity* against *1-specificity* across the cutoffs and allows us to visualize the aggregate performance of the expert system. Given the prior probabilities of the two classes and the costs of the two types of misclassification errors, we can determine an optimal cutoff (corresponding to a point on the ROC curve) that minimizes the expected misclassification cost.

We propose a method for tuning an expert system by determining the optimal cutoff for cost-sensitive problems, such as credit evaluation. We have developed an algorithm for tuning an expert system for optimal performance (see Algorithm 1). Given an expert system, a sample of n solved cases, a cost ratio between the two types of errors, and the prior probabilities of the two classes, the algorithm applies the expert system to classify the cases in the sample, estimates the expected misclassification cost under each possible cutoff, and finds the cutoff that results in the lowest expected misclassification cost.

The procedure starts with a loop that goes through all the n cases in the given sample ($S[i]$, $i = 1, 2, ..., n$), applying the expert system to classify each case, getting the CF value for each case (stored in $S[i].CF$), and counting n_1, the number of positive cases (bad loans) and n_0, the number of negative cases (good loans). The cases are then sorted based on the CF values in ascending order. Another loop is then used to go through all the cases and find the best cutoff. During each iteration, the cutoff is assumed to be the average of the CF values of two neighboring cases (stored in the variables τ_1 and τ_2). For the first iteration, the cutoff is assumed to be the minimum possible CF value, -1. For the last iteration, the cutoff is assumed to be the maximum possible CF value, 1. The number of false positive errors, n_{10}, and the number of false negative errors, n_{01}, are initially set to n_0 and 0, respectively, assuming that the cutoff is -1 and all cases are classified as positive. The two numbers are then updated at each iteration according to the actual outcome of the case under consideration. The expected misclassification cost is estimated using the previously defined formula and stored in the variable $\hat{C}$. The minimum expected misclassification cost found so far is maintained in the variable C^*. The CF values of the two neighboring cases that give the best cutoff so far are stored in the variables τ_1^* and τ_2^*. The procedure finally returns the average of τ_1^* and τ_2^* as the tuned cutoff for CF.

The time complexity for classifying the n cases in the sample using the expert system (i.e., the first loop) is $O(n)$. The time complexity for scanning through the n+1 possible cutoffs (i.e., the second loop) is also $O(n)$. The overall time complexity of the algorithm is therefore dominated by that of sorting the n cases according to their CF values, for which sorting algorithms as efficient as $O(n \log n)$ exist.

7. Empirical Evaluation

We empirically compared the performance (i.e., expected misclassification cost) of the expert system tuned using the proposed method and that of the original expert system, which uses the default cutoff of 0 for making classification decisions. The problem domain is credit evaluation, and the specific problem addressed is auto loan decisions. The sample used for tuning the expert system is a balanced sample, consisting of 110 positive cases (bad loans) and 110 negative cases (good loans). We varied the cost ratio r from 1 to 25 in steps of 1 and the prior probability p from 0.05 to 0.5 in steps of 0.05. We ran the proposed algorithm for each combination of r and p. Note that our performance measure is invariant with regard to different settings of p and r as long as the weight ratio between the two classes, $rp/(1-p)$, holds constant. Our choices of r and p cover a reasonably wide range of potential weight ratios (from 1 : 19 to 25 : 1).

Since the performance measure (i.e., expected misclassification cost) derived based on the sample for tuning the expert system can be overly optimistic, we used 10-fold stratified *cross-validation* [33], a widely recommended performance estimation method, to estimate the performance of the tuned expert system. A stratified n-fold cross-validation randomly divides (with stratification) a sample into n subsets, called *folds*, and repeatedly selects each fold for testing while the rest of the sample is used for tuning. The average testing performance over the n runs is used as

Expert_System_Tuning (f, S, r, p)
Input:
f: An expert system. Given the input vector of a problem case, f suggests a CF value. By default, the expert system makes a positive classification decision if the CF is positive and a negative classification decision, otherwise.
S: A sample of n solved cases $S[i](i = 1, 2, \ldots, n)$, each of which is a pair $\langle x, y\rangle$, where $x = \langle x_1, x_2, \ldots, x_m\rangle \in X$ and $y \in Y$. In addition, let $S[i] \cdot CF$ denote the CF value suggested by the expert system for the case $S[i]$.
r: Cost ratio. $r = \dfrac{C_{01}}{C_{10}}$.
p: Prior probability of the positive class.
Output: A tuned cutoff for the CF value.
BEGIN
$n_1 := n_0 := 0$.
FOR $i := 1$ TO n,
Classify $S[i]$ using the expert system, that is, $S[i].CF := f(S[i].x)$.
IF $S[i] \cdot y = 1$, THEN
$n_1 := n_1 + 1$.
ELSE
$n_0 := n_0 + 1$.
Sort the cases in S on the CF value in ascending order.
$C^* := \infty$.
FOR $i := 0$ TO n,
IF $i = 0$, THEN
$\tau_1 := \tau_2 := -1$.
ELSE IF $i < n$, THEN
$\tau_1 := S[i] \cdot CF$.
$\tau_2 := S[i + 1] \cdot CF$.
ELSE
$\tau_1 := \tau_2 := 1$.
IF $i = 0$, THEN
$n_{10} := n_0$; $n_{01} := 0$;
ELSE IF $S[i] \cdot y = 1$, THEN
$n_{01} := n_{01} + 1$.
ELSE
$n_{10} := n_{10} - 1$.
Assuming the cutoff of $(\tau_1 + \tau_2)/2$, compute the expected misclassification cost as $\hat{C} = \dfrac{rp}{(1-p)} \dfrac{n_{01}}{n_1} + \dfrac{n_{10}}{n_0}$.
IF $\hat{C} < C^*$, THEN
$C^* := \hat{C}$; $\tau_1^* := \tau_1$; $\tau_2^* := \tau_2$.
RETURN $(\tau_1^* + \tau_2^*)/2$.
END.

Algorithm 1: Expert system tuning algorithm.

an estimate of the performance of the tuned expert system. In addition, to get a more reliable estimate, we performed 10-fold stratified cross-validation 20 times and took the average over the 20 times as the final estimate.

Table 2 presents the confusion matrix generated by the expert system with the default CF cutoff of 0 based on the sample. Figure 7 shows the ROC curve generated by the expert system based on the sample. Given a particular combination of r and p, the proposed method identifies the point on the curve that results in the minimum expected misclassification cost.

Figure 8 contrasts the expected misclassification cost of the turned expert system with that of the original untuned expert system. Since the original expert system maintains the default cutoff of 0 for CF, its confusion matrix remains the same and the expected misclassification cost is simply a linear function of the weight ratio. When the

Table 2: Confusion matrix for the expert system.

		Predicted class		
		1	0	Sum
Actual Class	1	81	29	110
	0	30	80	110
				220

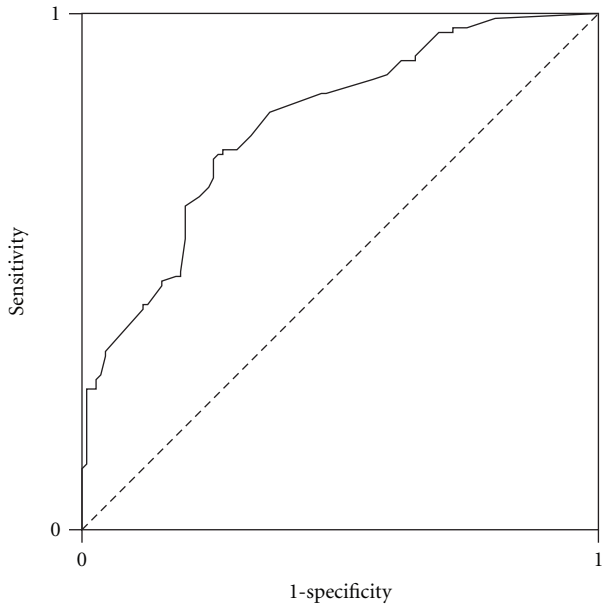


Figure 7: ROC curve for expert system.

weight ratio is close to 1 : 1, in the range of 1 : 3 to 3 : 1 (Figure 8(b)), tuning is not productive. However, when the weight ratio is far from 1 : 1, below 1 : 3, or over 3 : 1 (Figures 8(a) and 8(c)), the performance of the tuned system clearly dominates the performance of the original system; the expected misclassification cost of the tuned system is consistently lower than that of the original system. Paired t-tests and two nonparametric tests—Wilcoxon and sign tests—show that the tuned system yields significantly lower costs than the original system ($P < .001$). The farther the weight ratio is from 1 : 1, the greater is the influence of tuning on performance.

We also compared the optimal cutoff found on the tuning folds and the optimal cutoff found on the testing folds during the cross-validation. Figure 9 shows a scatter diagram of the cutoff values. The tuning cutoff and the testing cutoff are highly correlated ($r^2 = 0.989$), with a coefficient close to 1 (tuning cutoff $= 1.165 \times$ testing cutoff $- 0.048$), indicating that the cutoff found on the tuning folds is generalizable to the testing folds, thus demonstrating the robustness of the tuning method.

8. Discussion of Results

The results of the study strongly suggest that the auto loan expert system could be effectively tuned for optimal performance by applying the cross-validation technique for data mining. The optimal cutoffs determined by the proposed algorithm are based on data usually reserved for training a data mining model. Those cutoffs are then applied to an independent test set that was not used for determining the cutoffs. Therefore, the expected misclassification cost measure used in the study is an unbiased estimate of expert system performance.

We used a default cutoff of 0 for expert system decisions. Because the certainty of a conclusion in the $-.2 \leq \text{CF} \leq .2$ region is rather low—that is, there is very little support for or against the hypothesis—this CF space has been categorized as the "not known" region [17, 30]. Out of the 220 loan cases on which the expert system was applied, only for two did the final CF of the conclusion fall in that range. The overall results of the study would have therefore remained largely the same even if we had ignored the "not known" region in our analysis. Changing the cutoff, say from 0 to $-.2$, may have worked better for a specific weight ratio, but in general the expert system would still generate suboptimal performance results because the expected misclassification cost is a linear function of the weight ratio. In contrast, the expected misclassification cost for the tuned system remains almost flat as the weight ratio increases, thereby underscoring the efficacy of our approach.

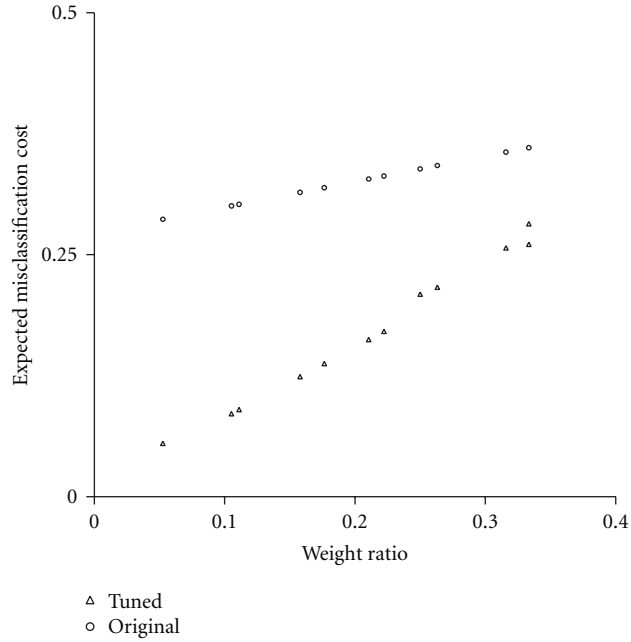
The validation technique that we employed was applied to cases with known outcomes, unlike prior work in validation which has mostly focused on validation by experts. While expert validation is a good idea, it is not possible for a human expert to vouch for the accuracy of a conclusion derived by the expert system. The fact that there is a good fit between the cutoffs derived using the training data and the test data suggest that our approach is generalizable to unseen cases.

Uncertainties associated with expert system conclusions were used to minimize misclassification costs. We used a data mining validation technique for tuning the system, by taking for granted the validity of the uncertainty of a hypothesis derived by the system. Strictly speaking, our approach, therefore, is aimed at validating the expert system with respect to misclassification cost. Because tuning is done post hoc, it does not affect any rules in the knowledge base, so there is no need to explicitly update the original expert system.

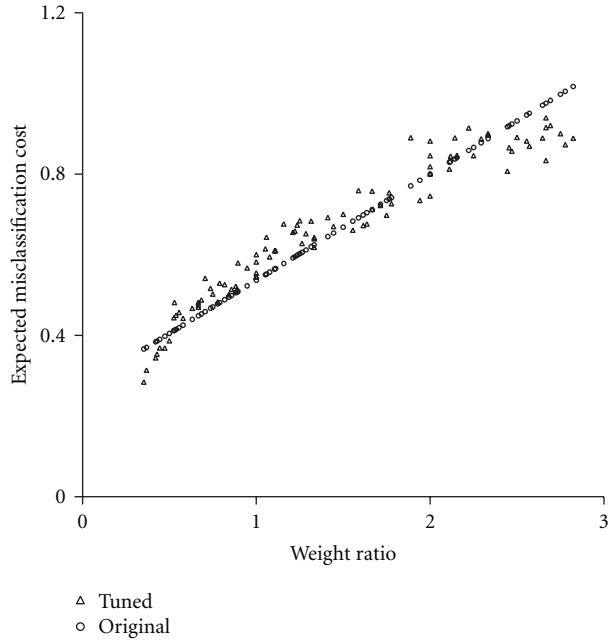
9. Conclusion

In this study, we have presented an empirical approach for tuning an expert system for making cost-sensitive decisions. The novel contribution of our work is in applying and validating a performance evaluation approach developed for data mining to tune expert systems for optimal cost performance. An important implication of the study is that although the empirical approach has been validated against an expert system, it could easily be extended to optimize any intelligent or knowledge system that generates uncertain conclusions. With the business intelligence (BI) market booming, we foresee the utility of the proposed approach for a host of BI applications.

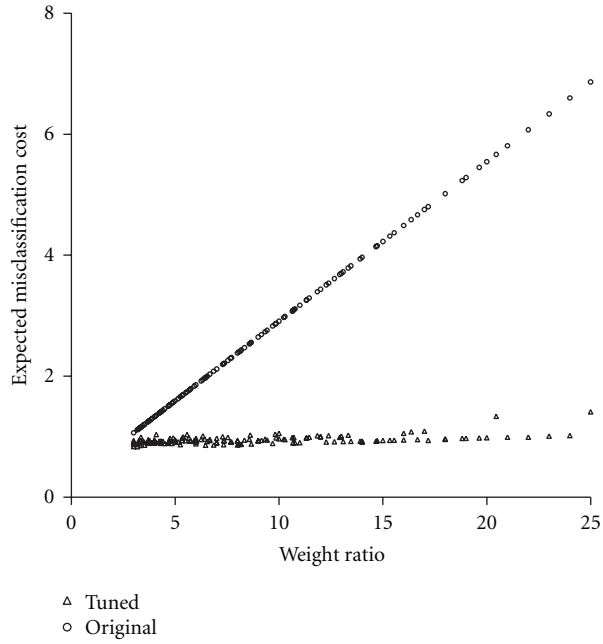
Our study opens up several avenues for future research. First, the proposed tuning method could be extended to tune expert systems that solve multiple-class problems. For a p-class ($p > 2$) classification problem, $p - 1$ cutoffs corresponding to the first $p - 1$ classes need to be tuned (the cutoff for the last class is determined by the cutoffs of the other $p - 1$ classes). A naïve approach would be to scan



(a) Weight Ratio below 1 : 3



(b) Weight Ratio between 1 : 3 and 3 : 1



(c) Weight Ratio over 3 : 1

Figure 8: Expected Misclassification costs of original and tuned expert systems.

through all permutations of the potential cutoffs of the $p - 1$ classes, resulting in a time complexity of $O(n^{p-1})$, where n is the number of tuning cases. When p is large, more efficient search algorithms such as hill-climbing, genetic algorithm, tabu search, and simulated annealing can be used.

Second, currently the proposed algorithm uses expected misclassification cost as a performance measure. This measure uses fixed costs for the different types of errors (false positive and false negative). However, in many applications, costs/benefits are actually different for different cases, depending on the actual outcomes. The current algorithm can be extended to evaluate a performance measure that captures such case-dependent, variable cost/benefit decisions.

Third, we focused on a classification task in this study, but future studies could examine if the results extend to regression or value prediction tasks, such as sales forecasting.

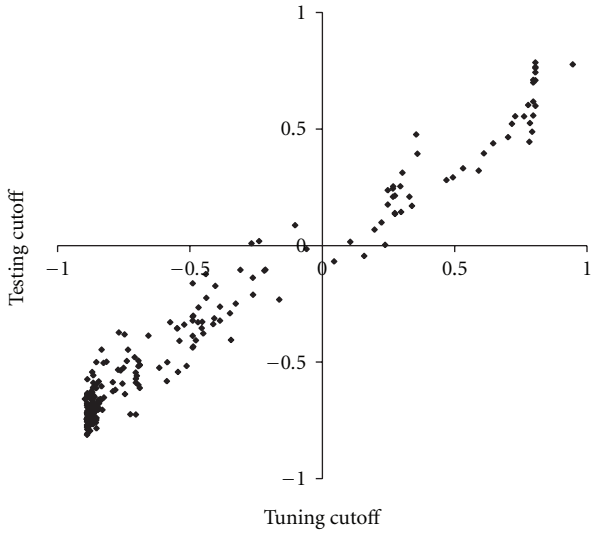


Figure 9: Cutoff values on tuning and testing folds.

Finally, the proposed method can be extended to problems with multiple dependent variables. In addition, the multiple dependent variables may be a mixture of both categorical variables and continuous variables.

Acknowledgment

An earlier and shorter version of this paper was published in the Proceedings of the INFORMS Workshop on Data Mining and Health Informatics, Washington, DC, October 2008.

References

[1] D. A. Waterman, *A Guide to Expert Systems*, Addison-Wesley, Reading, Mass, USA, 1986.

[2] C. W. Holsapple and A. B. Whinston, *Business Expert Systems*, Irwin, Homewood, Ill, USA, 1987.

[3] R. J. Mockler and D. G. Dologite, *Knowledge-Based Systems: An Introduction to Expert Systems*, Macmillan, New York, NY, USA, 1992.

[4] E. Turban, *Decision Support and Expert Systems: Management Support Systems*, Macmillan, New York, NY, USA, 3rd edition, 1993.

[5] H. Zhao, "A multi-objective genetic programming approach to developing Pareto optimal decision trees," *Decision Support Systems*, vol. 43, no. 3, pp. 809–826, 2007.

[6] H. Zhao, "Instance weighting versus threshold adjusting for cost-sensitive classification," *Knowledge and Information Systems*, vol. 15, no. 3, pp. 321–334, 2008.

[7] H. Zhao, A. P. Sinha, and W. Ge, "Effects of feature construction on classification performance: an empirical study in bank failure prediction," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2633–2644, 2009.

[8] A. P. Sinha and J. H. May, "Evaluating and tuning predictive data mining models using receiver operating characteristic curves," *Journal of Management Information Systems*, vol. 21, no. 3, pp. 249–280, 2004.

[9] A. P. Sinha and H. Zhao, "Incorporating domain knowledge into data mining classifiers: an application in indirect lending," *Decision Support Systems*, vol. 46, no. 1, pp. 287–299, 2008.

[10] R. Dybowski, K. B. Laskey, J. W. Myers, and S. Parsons, "Introduction to the special issue on the fusion of domain knowledge with data for decision support," *Journal of Machine Learning Research*, vol. 4, no. 3, pp. 293–294, 2004.

[11] I. Kopanas, N. M. Avouris, and S. Daskalaki, "The role of domain knowledge in a large scale data mining project," in *Applications of Artificial Intelligence*, I. P. Vlahavas and C. D. Spyropoulos, Eds., Lecture Notes in AI, no. 2308, pp. 288–299, Springer, Berlin, Germany, 2002.

[12] S. Muggleton, *Inductive Acquisition of Expert Knowledge*, Addison-Wesley, Reading, Mass, USA, 1990.

[13] P. E. Johnson, "What kind of expert should a system be?" *Journal of Medicine and Philosophy*, vol. 8, no. 1, pp. 77–97, 1983.

[14] R. M. O'Keefe, O. Balci, and E. P. Smith, "Validating expert system performance," *IEEE Expert*, vol. 2, no. 4, pp. 81–90, 1988.

[15] S. Lee and R. M. O'Keefe, "Developing a strategy for expert system verification and validation," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 4, pp. 643–655, 1994.

[16] D. E. O'Leary, "Validation of expert systems—with applications to auditing and accounting expert systems," *Decision Sciences*, vol. 18, no. 3, pp. 468–486, 1987.

[17] D. E. O'Leary, "Verification of uncertain knowledge-based systems: an empirical verification approach," *Management Science*, vol. 42, no. 12, pp. 1663–1675, 1996.

[18] C. Elkan, "The foundations of cost-sensitive learning," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pp. 973–978, Seattle, Wash, USA, 2001.

[19] G. Bansal, A. P. Sinha, and H. Zhao, "Tuning data mining methods for cost-sensitive regression: a study in loan charge-off forecasting," *Journal of Management Information Systems*, vol. 25, no. 3, pp. 315–336, 2009.

[20] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[21] F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in *Proceedings of the 15th International Conference on Machine Learning*, pp. 445–453, Madison, Wis, USA, 1998.

[22] G. M. Weiss and F. Provost, "The effect of class distribution on classifier learning: an empirical study," Tech. Rep. ML-TR-44, Dept. of Computer Science, Rutgers University, 2001.

[23] T. Fawcett, *ROC Graphs: Notes and Practical Considerations for Data Mining Researchers*, Intelligent Enterprise Technologies Lab, Hewlett-Packard, Palo Alto, Calif, USA, 2003.

[24] R. Ambrosino and B. G. Buchanan, "The use of physician domain knowledge to improve the learning of rule-based models for decision-support," in *Proceedings of Annual Fall Symposium American Medical Informatics Association*, pp. 192–196, Washington, DC, USA, 1999.

[25] H. Daniels, A. Feelders, and M. Velikova, "Integrating economic knowledge in data mining algorithms," in *Proceedings of the 8th International Conference of the Society for Computational Economics: Computing in Economics and Finance*, Aix-en-Provence, France, 2002.

[26] H. Langseth and T. D. Nielsen, "Fusion of domain knowledge with data for structural learning in object oriented domains," *Journal of Machine Learning Research*, vol. 4, no. 3, pp. 339–368, 2004.

[27] S. M. Weiss, S. J. Buckley, S. Kapoor, and S. Damgaard, "Knowledge-based data mining," in *Proceedings of the 9th*

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03), pp. 456–461, Washington, DC, USA, August 2003.

[28] Credit Union Journal, "Loan losses squeezing CUs," *American Banker*, vol. 173, 227, p. 5, 2008.

[29] J. Bjorhus, "Bad loans sap banks' profits," *Star Tribune*, 2008.

[30] B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems: The MYCIN experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, Mass, USA, 1984.

[31] M. Stefik, *Introduction to Knowledge Systems*, Morgan Kaufmann, San Francisco, Calif, USA, 1995.

[32] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, MIT Press, Cambridge, Mass, USA, 2001.

[33] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, C. S. Mellish, Ed., pp. 1137–1143, Morgan Kaufmann, San Mateo, Calif, USA, 1995.