©The Journal of Risk and Insurance, 2002, Vol. 69, No. 3, 373-421

A Comparison of State-of-the-Art Classification Techniques for Expert Automobile Insurance Claim Fraud Detection

Stijn Viaene
Richard A. Derrig
Bart Baesens
Guido Dedene

Abstract

Several state-of-the-art binary classification techniques are experimentally evaluated in the context of expert automobile insurance claim fraud detection. The predictive power of logistic regression, C4.5 decision tree, k-nearest neighbor, Bayesian learning multilayer perceptron neural network, least-squares support vector machine, naive Bayes, and tree-augmented naive Bayes classification is contrasted. For most of these algorithm types, we report on several operationalizations using alternative hyperparameter or design choices. We compare these in terms of mean percentage correctly classified (PCC) and mean area under the receiver operating characteristic (AUROC) curve using a stratified, blocked, ten-fold cross-validation experiment. We also contrast algorithm type performance visually by means of the convex hull of the receiver operating characteristic (ROC) curves associated with the alternative operationalizations per algorithm type. The study is based on a data set of 1,399 personal injury protection claims from 1993 accidents collected by the Automobile Insurers Bureau of Massachusetts. To stay as close to real-life operating conditions as possible, we consider only predictors that are known relatively early in the life of a claim. Furthermore, based on the qualification of each available claim by both a verbal expert assessment of suspicion of fraud and a ten-point-scale expert suspicion score, we can

Stijn Viaene, Bart Baesens, and Guido Dedene are at the K. U. Leuven Department of Applied Economic Sciences, Leuven, Belgium. Richard Derrig is with the Automobile Insurers Bureau of Massachusetts, Boston.

Presented at Fifth International Congress on Insurance: Mathematics & Economics July 23-25, 2001, Penn State University. This work was sponsored by the KBC Insurance Research Chair Management Informatics at the K. U. Leuven Department of Applied Economic Sciences. The KBC Research Chair was set up in September 1997 as a pioneering collaboration between the Leuven Institute for Research on Information Systems and the KBC Bank & Insurance group. We are grateful to the Automobile Insurers Bureau (AIB) of Massachusetts and the Insurance Fraud Bureau (IFB) of Massachusetts for providing us with the data that was used for this benchmark study.

compare classification for different target/class encoding schemes. Finally, we also investigate the added value of systematically collecting nonflag predictors for suspicion of fraud modeling purposes. From the observed results, we may state that: (1) independent of the target encoding scheme and the algorithm type, the inclusion of nonflag predictors allows us to significantly boost predictive performance; (2) for all the evaluated scenarios, the performance difference in terms of mean PCC and mean AUROC between many algorithm type operationalizations turns out to be rather small; visual comparison of the algorithm type ROC curve convex hulls also shows limited difference in performance over the range of operating conditions; (3) relatively simple and efficient techniques such as linear logistic regression and linear kernel least-squares support vector machine classification show excellent overall predictive capabilities, and (smoothed) naive Bayes also performs well; and (4) the C4.5 decision tree operationalization results are rather disappointing; none of the tree operationalizations are capable of attaining mean AUROC performance in line with the best. Visual inspection of the evaluated scenarios reveals that the C4.5 algorithm type ROC curve convex hull is often dominated in large part by most of the other algorithm type hulls.

INTRODUCTION

Detection of fraudulent claims has blossomed into a high-priority and technology-laden problem for insurers. This was not always so. Until the early 1980s, the polite way to discuss underwriting and claims settlement fraud was to include it with other potential adverse actions by policy holders and claimants under the rubric of moral hazard.[1] The common thread of all occurrences of moral hazard in insurance is that parties other than the insurer may hold unrevealed information that can materially affect the true size of the risk exposure or accidental loss. Policy holders may misinform the insurer *ex ante* about the expected level of care taken to minimize the exposure to specified insurable risks, while claimants may hold back *ex post* the true level of injury and/or medical treatment necessary and appropriate for the claimed injury. The adverse effects of moral hazard on insurers have been controlled historically through contract design as well as information-gathering activities during the underwriting and claims settlement processes.

Specific attention to fraudulent insurance transactions began to emerge in the United States with the designation of groups of experienced claims adjusters with specialized skills in the investigation of claims. The units came to be known generally as Special Investigation Units, or SIUs (Ghezzi, 1983), and are commonplace now in claims operations in the United States. Canadian and European insurers recognized the fraud problem as well and moved to adopt the SIU format for handling suspicious claims (Clarke, 1986, 1990; Comité Européen des Assurances, 1996; Dionne and Belhadji, 1996; Dionne, Gibbens, and St.-Michel, 1993; Insurance Bureau of Canada, 1994). By the late 1990s, companies in the United States had developed extensive internal procedures to cope with fraud while individual state governments established fraud bureaus to investigate and prosecute perpetrators criminally (Insurance Research Council, 1997). Automobile insurance, and bodily injury coverage in particular, were examined systematically in the 1990s for their claiming patterns, including fraud and excessive medical treatment known as buildup (Cummins and Tennyson, 1992, 1996; Derrig,

[1] For an up-to-date discussion of the full ramifications of moral hazard, see Doherty (2000).

Weisberg, and Chen, 1994; Dionne and Belhadji, 1996; Insurance Research Council, 1996; Weisberg and Derrig, 1991).

Operationally, company claims adjustment units identified those claims needing attention by noting the presence of one or more claim characteristics known as fraud indicators, or so-called red flags. Claims adjusters were trained to recognize (informally) those claims that had combinations of red flags that experience showed were associated with suspicious claims. Typically, the assessment of the probability or suspicion of fraud depended heavily on claims personnel to notice abnormalities in paper documents. Nowadays, the increasing systematic collection of data has made the use of pattern recognition techniques a valid and worthwhile endeavor (Derrig, 1999). Formal relations between red flag combinations and suspicious claims were established in closed claims empirical studies using Massachusetts data with regression techniques (Weisberg and Derrig, 1995, 1998), with fuzzy clustering (Derrig and Ostaszewski, 1995), and with unsupervised neural networks (Brockett, Xia, and Derrig, 1998); using Canadian data with regression and probit models (Belhadji, Dionne, and Tarkhani, 2000); and using Spanish market data with regression models (Artís, Ayuso, and Guillén, 1999, 2000). Four Massachusetts companies participated in a real-time experiment on a cohort of 1996 personal injury protection (no-fault) claims where claims were periodically scored using red flags and a regression scoring model to identify claims for investigation (Derrig and Weisberg, 1998). It needs no longer to be emphasized that using automated types of fraud detection should make it possible to reduce the investigative process lead time and allow for more optimal allocation of scarce investigative resources.

This article builds on work published by the Automobile Insurers Bureau (AIB) of Massachusetts on personal injury protection (PIP) automobile insurance claim fraud detection. The data underlying our experiments were studied and documented by the AIB in (Weisberg and Derrig, 1991, 1995, 1998). Here, the claims handling process is conceptualized as a two-stage process. In the first stage, the claim is judged by a front-line adjuster, whose main task is to assess the exposure of the insurance company to payment of the claim. In the same effort, the claim is scanned for claim padding and fraud. Claims that are modest or appear to be legitimate are settled in a routine fashion. Claims that raise questions and involve a substantial payment are scheduled to pass a second reviewing phase. In case fraud is suspected, this might come down to a referral of the claim to an SIU.

In line with the analyses using pattern recognition technology for developing models for first-stage claims screening, as discussed above, we set up a benchmarking study including several state-of-the-art binary classification techniques: logistic regression, C4.5 decision tree, *k*-nearest neighbor, Bayesian learning multilayer perceptron neural network, least-squares support vector machine, naive Bayes, and tree-augmented naive (TAN) Bayes classification. For most of these algorithm types we report on several operationalizations. We compare these on the basis of their ability to reproduce human expert decisions to further investigate a claim or not. We build robustness analysis with regard to the target/class concept into the experimental plan by doing the same comparison experiment for different referral definition thresholds. This allows us to mimic different company policies or target concept drift. The latter is possible by the qualification of each available claim by both a verbal expert assessment of suspicion of fraud and a ten-point-scale expert suspicion score

(Weisberg and Derrig, 1991, 1995, 1998). We compare the algorithms using a stratified, blocked, ten-fold cross-validation setup. We specify the performance of the algorithms in terms of mean percentage correctly classified and mean area under the receiver operating characteristic curve. We also contrast algorithm type performance visually by means of the convex hull of the receiver operating characteristic curves associated with the alternative operationalizations per algorithm type.

Benchmarking different classification methods is not an easy task. As pointed out in previous research (Duin, 1996; Friedman, Geiger, and Goldsmidt, 1995; Michie, Spiegelhalter, and Taylor, 1994), no classification method is universally better than any other. Moreover, benchmarking is not a one-dimensional problem. There are different bases to compare algorithms or algorithm types (for example, modeling power, speed of training, tuning, classifying, interpretability, and actionability of the model). In addition, comparisons are biased by the characteristics of the data (for example, the dimensionality of the data, the number of data instances, the predictor encoding), the application domain, and the skill of the researcher. It is important to define the application domain carefully beforehand and to make sure that the data underlying the study are representative of this domain. The above discussion should allow the reader to assess the scope and representativeness of our comparison. Furthermore, in order to remove undue bias due to the skill of the researcher, which *a priori* might favor some algorithms over others, we adhere to a strategy of minimally human expert tuned algorithms. Some of the discussed techniques require no tuning. Other techniques are characterized by hyperparameters that require tuning. As in the *Statlog* project (Michie, Spiegelhalter, and Taylor, 1994), we attempt to apply minimal human expert tuning, so we rely on default—that is, widely accepted—hyperparameter settings as much as possible. Where this is not possible, we tune hyperparameters using the training data or we report on several operationalizations using alternative, *a priori* sensible design or hyperparameter choices. This ensures that the obtained results are largely independent of the human experts that perform the analyses. Notice that, although the results will probably be inferior to those that would have been obtained using human expert tuning, this way of working guarantees that the algorithms are compared apple to apple.

An additional contribution of this work resides in the confrontation of models that have been trained using fraud indicator/flag predictors only and models that make use of a combination of fraud indicators and nonindicator predictors. This enables us to investigate the added value of systematically collecting nonflag predictors for suspicion of fraud modeling purposes. In our choice of predictors for this study we were led by previous work by the AIB on the timeliness of predictor availability within the life of a claim (Derrig and Weisberg, 1998). The timing of the arrival of the information on claims is crucial to its usefulness for the development of an early claims screening facility. To stay as close to real-life operating condition requirements for early screening as possible, we only include predictors that are known relatively early in the life of a claim.

The article is then organized as follows. The second section briefly covers the essentials of the data set that is used in the analyses. Details on how the performance of the various algorithms is measured are given in the third section. In the fourth section we elaborate on the essentials of the algorithms figuring in the benchmarking study. Results and discussion are found in the fifth section. The discussion is closed with a summary and conclusions.

PIP CLAIMS DATABASE

This study is based on a data set of 1,399 PIP claims from 1993 accidents collected by the AIB. Details on the exact composition and semantics of the data set and the data collection process can be found in Weisberg and Derrig (1991, 1995, 1998).

Guided by an analysis of the timing of claim information in Derrig and Weisberg (1998), we retained the binary fraud indicators/flags in Table 1 as predictors for the development of our first-stage claims screening models (Weisberg and Derrig, 1998). The listed fraud indicators may all be qualified as typically available relatively early in the life of a claim. The indicator names are identical to the ones used in previous work. Notice that no indicators related to the medical treatment are included in the data set. None of these fraud indicators are typically available early enough to be taken up into the data set (Derrig and Weisberg, 1998). In our selection of fraud indicators, we also imposed a pre-condition of having at least ten data instances in our data set where the flag was set—that is, where the fraud indicator fired. A similar selection operation was performed by Belhadji, Dionne, and Tarkhani (2000). This led us to drop ACC07, LW04, and LW07 from the initial list of early fraud indicators, since these fired for only five, four, and three data instances, respectively. We emphasize that this does not imply that these indicators are considered unimportant per se. Remember, they are supposed to make sense to adjusters. Dropping infrequent indicators is motivated primarily by considerations pertaining to model convergence and stability during estimation. However, this should only be done after careful deliberation, pondering all the aspects of concern (for example, taking into account the aim of the study, the characteristics of the available data, the experimental setup, the characteristics of the classification and estimation techniques, and the domain specifics). Concerning the latter, Massachusetts insurers are denying about 1 percent of injury claims. That would amount to 14 claims in the data set.

To evaluate the additional information content of nonflag predictors we also decided to consider the predictors in Table 2 for inclusion in the classification models. The selection of the predictors was steered by discussion with domain experts within the limits of what was available in the coded data. We thus emphasize that this is only an initial example of adding nonflag predictors, not an attempt at a complete or efficient model. Again, information regarding the retained predictors is usually obtained relatively early in the life of a claim. We discretized continuous predictors before their

TABLE 1
PIP Red Flag Predictors (0 = unset, 1 = set) Arriving Early in the Claims Process

Subject	Binary Fraud Indicators
Accident	ACC01, ACC04, ACC09, ACC10, ACC11, ACC14, ACC15, ACC16, ACC18, ACC19
Claimant	CLT02, CLT04, CLT07
Injury	INJ01, INJ02, INJ03, INJ05, INJ06, INJ11
Insured	INS01, INS03, INS06, INS07
Lost Wages	LW01, LW03

TABLE 2
PIP Nonflag Predictors Arriving Early in the Claims Process

Predictor	Description	Values and Discretization
AGE	Age of the claimant at the time of the accident (in years)	20-year bins: {0-20,21-40,41-60,61-80,81+}
POL_LAG	Lag from the beginning of the (one year) policy to the accident (in days)	variable bin width: {0-7,8-15,16-30,31-45, 46-60,61-90,91-180,181-270,271+}
REPT_LAG	Lag from the accident to its reporting (in days)	variable bin width: {0-7,8-15,16-30,31-45, 46-60,61-90,91-180,181-270,271-360,361+}
TRT_LAG1	Lag from the accident to medical provider 1 first outpatient treatment (in days); zero value meaning no outpatient medical treatment or only emergency treatment within first 48 hours	variable bin width: {0-7,8-15,16-30,31-45, 46-60,61-90,91-180,181-270,271-360,361+}
$TRT\_LAG1^{0/1}$	If TRT_LAG1 = 0, then $TRT\_LAG1^{0/1} = 0$, otherwise $TRT\_LAG1^{0/1} = 1$	binary predictor: {0,1}
TRT_LAG2	Lag from the accident to medical provider 2 first outpatient treatment (in days); zero value if not applicable	variable bin width: {0-7,8-15,16-30,31-45, 46-60,61-90,91-180,181-270,271-360,361+}
$TRT\_LAG2^{0/1}$	If TRT_LAG2 = 0, then $TRT\_LAG2^{0/1} = 0$, otherwise $TRT\_LAG2^{0/1} = 1$	binary predictor: {0,1}
AMBUL	Ambulance charges amount (in US$)	100-US$ bins: {0-100,101-200,201-300, 301-400,401-500,501+}
$AMBUL^{0/1}$	If AMBUL = 0, then $AMBUL^{0/1} = 0$, otherwise $AMBUL^{0/1} = 1$	binary predictor: {0,1}

PART_DIS1$^{0/1}$	Was the claimant partially disabled due to the accident? The actual number of weeks the claimant was diagnosed partially disabled is a late predictor and therefore not included in the predictor set.	binary predictor: {0 = no,1 = yes}
TOT_DIS1$^{0/1}$	Was the claimant totally disabled due to the accident? The actual number of weeks the claimant was diagnosed totally disabled is a late predictor and therefore not included in the predictor set.	binary predictor: {0 = no,1 = yes}
SCLEGREP	Was the claimant represented by an attorney? Note the difference with CLT05, a flag indicating whether a high-volume attorney was retained, but which was omitted from Table 1.	binary predictor: {0 = no,1 = yes}

TABLE 3
Semantics of the Alternative Target Encoding Schemes

Encoding	$t = -$	$t = +$	$\hat{p}(t = +)$
1+	suspicion rate < 1	suspicion rate ≥ 1	42.96%
4+	suspicion rate < 4	suspicion rate ≥ 4	28.31%
7+	suspicion rate < 7	suspicion rate ≥ 7	8.79%
vnl	code1	code2 up to code5	36.03%
vop	code1 and code2	code3 up to code5	16.30%

inclusion in the models by dividing up their continuous value ranges in bins and replacing the actual value with the respective bin numbers 1, 2, 3, and so on. Although algorithmic means of discretization could have been applied at this stage, we relied on prior domain expertise and inspection of the distribution of the values within the continuous value ranges. We statistically normalized all predictors by subtracting their mean over the data and dividing by their standard deviation before their inclusion in the models (Bishop, 1995).

A senior claims manager reviewed each claim file on the basis of all available information (Weisberg and Derrig, 1995). This closed claims reviewing was summarized into a ten-point-scale expert assessment of suspicion of fraud, with zero standing for no suspicion of fraud. Claims were also categorized in terms of the following five-level verbal assessment hierarchy: probably legitimate (code1), excessive treatment (buildup) only (code2), suspected opportunistic fraud (2 types: code3 and code4), and suspected planned fraud (code5).

Based on the qualification of each available claim by both a verbal expert assessment of fraud suspicion as well as a ten-point-scale suspicion score, we are able to compare classification for different fraud suspicion definition thresholds. It allows us to assess the robustness of the algorithms with regard to the target definition threshold under different detection policies. This gives rise to the alternative target encoding schemes in Table 3. Usually, 4+ target encoding is the operational domain expert choice. For that reason, specific attention should be attributed to the outcome and discussion of the comparison experiment for this target encoding scheme. Information on the target label $t \in \{-,+\}$ for the claims data is summarized by the data prior $\hat{p}(t = +)$ in the last column of Table 3.[2]

EXPERIMENTAL SETUP

In this section we give an overview of the experimental setup used for our benchmarking study. In the first part of this section we present the employed performance criteria for classification. In the second part of this section the evaluation procedure is detailed. The discussion is limited to the case of binary classification.

[2] In the rest of the article we will consistently use $\hat{p}$ as the notation for the estimate of p.

Performance Criteria for Classification

The percentage correctly classified (PCC), or classification accuracy, an estimate of a classifier's probability of a correct response, is undoubtedly the most commonly used classifier performance evaluation and comparison basis. It is calculated as the proportion of correctly classified data instances for a (test) set of (unseen) labeled data $\{x_i,t_i\}_{i=1}^{N}$ representative of the studied population. Formally, it can be described as:

$$\text{PCC} = \frac{1}{N}\sum_{i=1}^{N}\delta(y_i^t,t_i), \tag{1}$$

where y_i^t is the predicted target for measurement vector x_i, t_i is its true target, and $\delta(.,.)$ is 1 if both arguments are equal, 0 otherwise. Instead of estimating a classifier's probability of a correct response by means of its PCC on a separate test set of previously unseen data, one very often opts for a cross-validated estimate (see below). In what follows we will narrow the discussion down to the case of binary classification.

In a number of circumstances, classification accuracy may not be the most appropriate performance criterion for comparing classifiers. For one, maximizing classification accuracy tacitly assumes equal misclassification costs for false positive and false negative predictions (Bradley, 1997; Duda, Hart, and Stork, 2001; Hand, 1997; Provost and Fawcett, 2001; Provost, Fawcett, and Kohavi, 1998; Webb, 1999).[3,4] This assumption is problematic, since for many real-life decision making situations it is most likely violated. Another implicit assumption of the use of classification accuracy as an evaluation and comparison metric is that the class distribution is presumed constant over time and relatively balanced (Provost, Fawcett, and Kohavi, 1998). For example, in the case of binary classification, when confronted with a situation characterized by a naturally very skewed class distribution in which faulty predictions for the rare class are very costly, a model optimized on classification accuracy alone may very well always predict the most prevalent class and thus in terms of PCC yield relatively high performance. The performance of this majority model may be difficult to beat, although it presumably is unacceptable if a nontrivial solution is required. The evaluation of classifiers based on classification accuracy then proves inappropriate. Class distributions and misclassification costs are rarely uniform. Furthermore, taking into account class distributions and misclassification costs for building and evaluating classifiers proves to be quite hard, since in practice they can rarely be specified precisely and are often subject to change (Fawcett and Provost, 1997; Provost and Fawcett, 2001; Provost, Fawcett, and Kohavi, 1998). In spite of the above, comparisons based on classification accuracy often remain useful because they are indicative of a broader notion of good performance.

---

[3] This is under the assumption that misclassification costs have not been accounted for through distortions of the original data set—that is, by over- or undersampling the data instances according to their relative misclassification costs (Breiman, Friedman, Olshen, and Stone, 1994; Domingos, 1999).

[4] In a binary classification context, the class of most interest is often termed positive (+), the other negative (−). For this study, the positive target coincides with suspicion of fraud, as defined for each of the target encoding schemes in the third column of Table 3. For the numeric encoding of the classes, see the fourth section.

TABLE 4
Confusion Matrix for Binary Classification

	Actual Target	
Predicted Target	+	−
+	True Positive (TP)	False Positive (FP)
−	False Negative (FN)	True Negative (TN)

Descriptive statistics such as the false positives, false negatives, sensitivity, and specificity can provide more meaningful results. Class-wise decomposition of the classification yields a confusion matrix, as specified in Table 4. The following performance measures can readily be distilled from Table 4.

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}}. \tag{3}$$

Thus, the sensitivity and specificity measure the proportion of + data instances that are predicted + and the proportion of − data instances that are predicted −, respectively. Using the notation of Table 4, we may formulate the PCC as:

$$\text{PCC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{4}$$

All the above measures of performance are still characteristic of one particular operating point—that is, a specific set of operating conditions. Most classifier predictive performance comparisons, by their use of the PCC as an evaluation measure, implicitly go for the operating point that minimizes error rate. The assumptions underlying this choice have been explained above. More generally, we want to choose an operating point that minimizes misclassification cost—that is, one that takes into account the cost of false positive and the cost of false negative predictions. However, since costs can rarely be specified unambiguously, and hence the exact operating conditions are often unknown, it would be helpful if we could somehow, without too much extra work, compare classification behavior under various alternative operating conditions. This can be done as follows.

Many classifiers naturally produce, or are capable of producing, a continuous output to which different classification or decision thresholds may be applied to predict class membership.[5] Varying this classification threshold is a commonly adopted means for incorporating the relevant operating conditions into the classification model's construction process (Bradley, 1997; Duda, Hart, and Stork, 2001; Hand, 1997; Provost

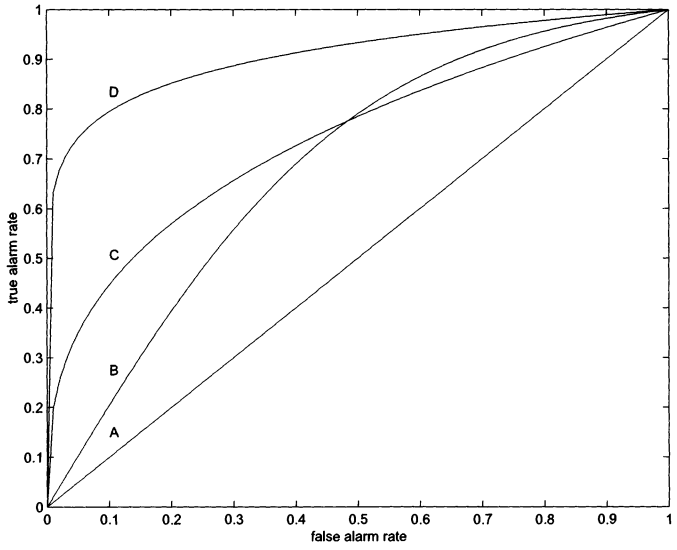
[5] Depending on the context, the term *classifier* may thus refer to the continuous-output model or the actual classification model—that is, the continuous-output model to which a classification threshold has been applied to predict class membership.

and Fawcett, 2001; Provost, Fawcett, and Kohavi, 1998; Webb, 1999).[6] The sensitivity, specificity, and PCC vary together as the classification threshold on a classifier's continuous output is varied between its extremes. The receiver operating characteristic (ROC) curve is a two-dimensional visualization of the sensitivity—that is, the true alarm rate—on the y-axis versus 1-specificity—that is, the false alarm rate—on the y-axis for various values of the classification threshold. It basically illustrates the behavior of a classifier in terms of $(\frac{FP}{FP+TN}, \frac{TP}{TP+FN})$ pairs without regard to class distribution or error cost. So it effectively decouples classification performance from these factors (Swets, 1979; Swets and Pickett, 1982). For a good understanding, each point in ROC space corresponds to a particular $(\frac{FP}{FP+TN}, \frac{TP}{TP+FN})$ pair. It corresponds to a continuous-output classifier evaluated under one specific operating condition setting, defined by a specific classification threshold. Many classifiers are capable of producing an estimate of a data instance's posterior + class membership probability as a byproduct of the classification process. These can then be used for scoring the data instances, ranking them, and applying ROC curve analysis. Strictly speaking, however, for the application of ROC analysis we do not need exact or well-calibrated probability estimates. ROC analysis can be applied to any classifier that produces a continuous output to which different classification thresholds may be applied to predict class membership. ROC analysis has been used previously to assess the quality of red flag models (Weisberg and Derrig, 1995).

Figure 1 provides an example of several ROCs. Each curve passes through the corner points (0,0) and (1,1). The former point represents the situation whereby the classification threshold exceeds the highest output value of the classifier for any of the data instances to be classified, meaning all data instances are classified as −. For the latter point in ROC space the classification threshold is lower than the lowest output value, meaning all data instances are classified as +. A straight line through the corner points (0,0) and (1,1) represents a classifier with poor discriminative power, since the sensitivity always equals 1 − specificity for all possible values of the classification threshold (curve A). Informally, the more the ROC approaches the (0,1) point, the better the classifier will discriminate under various operating conditions (for example, curve D dominates curves A, B, and C). ROCs for different classifiers may, however, intersect, making a general performance comparison less obvious (for example, curves B and C). To overcome this problem, one often calculates the area under the ROC (AUROC) curve (Bradley, 1997; Hand, 1997; Hanley and McNeil, 1982, 1983).

[6] Varying the classification threshold is by no means the only way of accounting for different operating conditions or misclassification costs. Costs may also be incorporated through distortions of the original data set—that is, by over- or undersampling data instances according to their relative misclassification costs (Breiman, Friedman, Olshen, and Stone, 1984; Domingos, 1999). Alternatively, cost considerations may be included in the actual model construction phase by using cost-sensitive optimization during learning (see, for example, Breiman, Friedman, Olshen, and Stone, 1984; Knoll, Nakhaeizadeh, and Tausend, 1994; Verrelst, Moreau, Vandewalle, and Timmerman, 1997). MetaCost, a generic wrapper or meta-learning method from Domingos (1999), is yet another method of making cost-sensitive classification decisions. A more elaborate and comprehensive discussion of cost-sensitive learning and decision making is beyond the purposes of this study. See Turney (1996) for an up-to-date bibliography.

FIGURE 1
Example ROCs



It should be clear that a comparison of classifiers based on this whole-curve, single-figure AUROC measure is only appropriate for cases in which specific operating conditions are unknown or vague and more general comparison over a range of operating conditions is appropriate. The AUROC then provides a simple figure of merit for the classifier's classification performance across a wide range of operating conditions. Moreover, the AUROC is equivalent to the non-parametric Wilcoxon-Mann-Whitney statistic, which provides an estimate of the probability that a randomly chosen positive data instance is correctly rated or ranked higher than a randomly selected negative data instance (Hand, 1997; Hanley and McNeil, 1982, 1983). This equivalence implies that the AUROC can be used as an estimate of the relative quality, be it in general terms, of the posterior class membership probability estimates produced by a classifier (Provost and Domingos, 2000).

Two comments are in order. First, under specific operating conditions, a classifier with higher AUROC may well be worse than a classifier with lower AUROC, since then we are to compare points in ROC space, not whole curves. Second, AUROC is based solely on the relative ranking of data instances—specifically, according to the continuous output of a classifier. Any monotonic increasing transformation of the classifier's continuous output would produce the same ranking and would lead to the same ROC and AUROC. Thus, the AUROC analysis does not allow us to assess whether posterior class membership probability estimates are well-calibrated. However, in case calibration implies fitting a monotonic increasing function to the uncalibrated output to produce well-calibrated probabilities, then the AUROC analysis may well be assumed to give an assessment of the inherent ability of a classifier to produce good probability estimates. An example of the latter can be found in Platt (2000), where the continuous output of the uncalibrated support vector machine classifier is assumed to be proportional to the log odds of a positive data instance, and consequently a

parameterized sigmoid is fitted to the data to obtain better-calibrated posterior class membership probability estimates.

The algorithm used for generating a ROC from a set of ranked data instances is described in Provost and Fawcett (2001). As in Provost, Fawcett, and Kohavi (1998), the curve is treated as a function—that is, we use linear interpolation between the generated points in ROC space—and sampled at 100 points regularly spaced along the x-axis. The AUROC is calculated using a trapezoidal approach. It has been stated that the trapezoidal calculation underestimates the AUROC. However, the underestimation is systematic and, provided there are a reasonable number of points on the ROC, the underestimation should not be too severe (Bradley, 1997).

Experimental Evaluation

We present our results in terms of mean PCC and mean AUROC using a stratified, blocked, ten-fold cross-validation experiment. We assess algorithm performance differences using a two-way ANOVA and Duncan's multiple range test.

In our experiments, we use a stratified, ten-fold cross-validation procedure to estimate expected algorithm performance. The data set is randomly split into ten mutually exclusive subsets or folds of comparable size, while approximately respecting the class proportions of the initial data set—that is, the folds are stratified. Ten-fold cross-validation is a resampling technique in which ten classifiers are trained each time using only nine folds of the data and the remaining fold is used for testing. As a result, all observations are used for training and each observation is used exactly once for testing. The ten-fold cross-validation performance estimates are averaged to obtain an estimate of the classification rule's true performance. Ten-fold cross-validation has been extensively studied and has been shown to provide good point estimates of the true performance of classification algorithms using a limited set of data (Bradley, 1997; Kohavi, 1995; Lim, Loh, and Shih, 2000; Webb, 1999).

For comparing the performance of the classification algorithms, we make use of a two-way ANOVA analysis and Duncan's multiple range test. The chosen methodology for comparing algorithm performance was proposed and discussed in Bradley (1997). A blocked experimental design is realized by using exactly the same training and test set folds for all the classification algorithms. To test which of the means are significantly different from which others, we apply Duncan's multiple range test to the ten-fold cross-validation mean performance criteria using the training and test set partitioning as a blocking factor. The latter separates the variance due to the different learning algorithms from the variance due to the training and test set partitions. This results in a linear model that relates a performance criterion—specifically, the PCC or AUROC—to both the algorithm effects and the training and test set partitioning effects. When the ANOVA test on the performance measure provides evidence that not all of the means are equal, due to different algorithms and different training and test set partitions,[7] we use Duncan's multiple range test to separate significantly different algorithm means into subsets of homogeneous means. We stress that this is an approximate statistical test for comparing the performance of algorithms using

[7] This is a test that simultaneously tests two hypotheses: All means are equal due to different training and test set partitions, and all means are equal due to different learning algorithms (Bradley, 1997). In this case, we are actually only interested in the latter hypothesis.

ten-fold cross-validation. A more elaborate discussion on the topic of statistical algorithm performance comparison using a single small data set is beyond the scope of this article. See Dietterich (1998), for a position statement. The ANOVA analyses were carried out using PROC GLM in SAS for Windows V8.

Alternatively, we can plot ROCs to visually compare ten-fold cross-validated model performance in function of alternative operating conditions. In this work, we obtain a cross-validated model's ROC by averaging the ROCs over the ten sub-experiments of the ten-fold cross-validation, as described in Provost, Fawcett, and Kohavi (1998).

ALGORITHMIC ESSENTIALS

In this section we give an overview of the essentials of the classification techniques that are taken up in our benchmarking study. Since we are unable to cover all of the technical details in this article, we refer to the relevant literature where needed. The algorithm types we discuss are logistic regression, k-nearest neighbor, C4.5 decision tree, Bayesian learning multilayer perceptron neural network, least-squares support vector machine, naive Bayes, and tree-augmented naive Bayes classification. The discussion is limited to the case of binary classification.

Logistic Regression Classifiers

Many empirical studies have used logistic regression as a benchmark (Agresti, 1990; Duda, Hart, and Stork, 2001; Sharma, 1996; Webb, 1999). Logistic regression makes the assumption that the difference between the natural logarithms of the class-conditional data density functions is linear in the inputs/predictors as follows. Given a training set $D = \{x_i, t_i\}_{i=1}^{N}$ with input vectors $x_i = (x_i^{(1)}, \ldots, x_i^{(n)})^T \in \mathbb{R}^n$ and target labels $t_i \in \{0,1\}$, logistic regression assumes that:

$$\ln\left(\frac{p(x|t=1)}{p(x|t=0)}\right) = b + w^T x, \tag{5}$$

where $p(x|t=1)$ and $p(x|t=0)$ represent the class-conditional data density functions, w represents the weight or coefficient vector, and b the intercept bias. The posterior class membership probabilities $p(t|x)$ underlying classification can easily be obtained from the model in Equation (5). This gives:

$$\begin{aligned} p(t=1|x) &= \frac{\exp(b' + w^T x)}{1 + \exp(b' + w^T x)} \\ p(t=0|x) &= \frac{1}{1 + \exp(b' + w^T x)} \end{aligned} \tag{6}$$

where $b' = b + \ln\left(\frac{p(t=1)}{p(t=0)}\right)$, with $p(t=1)$ and $p(t=0)$ the class priors. We can use this posterior class membership probability specification to obtain maximum likelihood estimates for w and b'. We assume that the prior class proportions in the data are representative of the true class priors.

Discrimination between the two classes is governed by the ratio $\frac{p(t=1|x)}{p(t=0|x)}$. It can easily be shown that it is determined solely by the linear function $b' + w^T x$. In brief, logistic discrimination uses a linear discriminant function with the additional assumption in Equation (5), which underlies maximum likelihood estimation. Besides the assumption in Equation (5), logistic regression does not make any distributional assumptions for the inputs and has been shown to work well in practice for data that depart significantly from conventional normality assumptions (see, for example, Michie, Spiegelhalter, and Taylor (1994)). The analyses were carried out using PROC LOGISTIC in SAS for Windows V8 with the default hyperparameter settings at TECHNIQUE=FISHER and RIDGING=RELATIVE.

k-Nearest Neighbor Classifiers

Nearest neighbor methods estimate the probability that a data instance with measurement vector $x_i = (x_i^{(1)}, \ldots, x_i^{(n)})^T \in \mathbb{R}^n$ belongs to class $t \in \{0,1\}$, that is, $p(t|x)$, by the proportion of points in the training set $D = \{x_i, t_i\}_{i=1}^N$ in the neighborhood of x that belong to that class. The concept of neighborhood is then defined by the distance from x to the k^{th} nearest point in the training set. Here k fulfills the role of smoothing hyperparameter: A larger k will mean less variance in the probability estimates, but at the risk of introducing more bias. Essentially, nearest neighbor methods estimate average probabilities over a local neighborhood. Only if

$$p(t|x) = \int_{N_x} p(t|z)p(z)\mathrm{d}z, \tag{7}$$

where $p(z)$ is the prior density over the input data and the integral is taken over the k-nearest neighborhood N_x centered around x, will the estimate be unbiased (Hand, 1997).

The most commonly used metric for evaluating the distance between two measurement vectors x_1 and x_2 is 2-norm distance—that is, the Euclidean distance:

$$\mathrm{dist}(x_1,x_2) = \|x_1 - x_2\|_2 = \sqrt{(x_1 - x_2)^T(x_1 - x_2)}. \tag{8}$$

Notice that, since all inputs are to be treated equally if no prior information is available stating otherwise, it is important to normalize the input values to ensure that the k-nearest neighbor rule is independent of the measurement units. In this case, as stated in the second section, we statistically normalize all inputs to zero mean and unit variance before their inclusion in the models.

The version of k-nearest neighbor that was implemented for this study was chosen because it is especially appropriate for handling discrete data (Webb, 1999). The problem with discrete data is that several training data instances may be at the same distance from a test data instance x as the k^{th} nearest neighbor, giving rise to a nonunique set of k-nearest neighbors. The k-nearest neighbor classification rule then works as follows. Let the number of training data instances at the distance of the k^{th} nearest neighbor be n_k, with n_{k1} data instances in class $t = 1$ and n_{k0} data instances in class $t = 0$. Let the total number of training data instances within, but excluding, this distance be N_k, with N_{k1} data instances in class $t = 1$ and N_{k0} data instances in class $t = 0$. Classify a test data instance in class $t = 1$ if

$$N_{k1} + \frac{k - N_k}{n_k} n_{k1} \geq N_{k0} + \frac{k - N_k}{n_k} n_{k0}, \tag{9}$$

where $N_k < k \leq N_k + n_k$. Now all training data instances at the distance of the k^{th} nearest neighbor are used for classification, although on a proportional basis.

C4.5 Decision Tree Classifiers

CART (Breiman, Friedman, Olshen, and Stone, 1984), CHAID (Kass, 1980), and C4.5 (Quinlan, 1993) are among the best-known tree-based classification algorithms. The main differences between the algorithms stem from the differences in the rules for splitting nodes when growing the tree and the tree pruning strategy. For this study, we report on the use of C4.5 due to Quinlan (1993). We used C4.5 Release 8 for all analyses.[8] A full tree is grown following a divide-and-conquer approach based on greedy predictor choice using the Gain ratio node partitioning rule (see below). To avoid near-trivial partitioning, C4.5 imposes the additional stopping condition that at least two of the subsets created by partitioning a node must contain a minimum default of two training data instances.

The Gain ratio is an information-based measure for partitioning a data set $D = \{x_i, t_i\}_{i=1}^{N}$ corresponding to a node in the tree, with input vector $x_i = (x_i^{(1)}, \ldots, x_i^{(n)})^T \in \mathbb{R}^n$ and target labels $t_i \in \{0,1\}$. The rationale underlying C4.5's data partitioning is that the information conveyed by a message depends on its probability and can be measured in bits as the natural logarithm of that probability. One can then express the residual information about the class to which a data instance in D belongs (a.k.a., entropy)—that is, the average amount of information needed to identify its class label—as:

$$\text{Info}(D) = -\sum_{t=0}^{1} \hat{p}(D,t) \ln\big(\hat{p}(D,t)\big), \tag{10}$$

where $\hat{p}(D,t)$ is the proportion of data instances in D that are labeled t (Quinlan, 1993).

By dividing D into non-overlapping subsets D_{a_j}, where D_{a_j} contains those data instances in D that have predictor $x^{(m)}$ set to value a_j, with predictor value index $j \in \{1, \ldots, v\}$ and v representing the number of different values for the splitting predictor $x^{(m)}$, the information gained is specified as:

$$\text{Gain}\big(D, x^{(m)}\big) = \text{Info}(D) - \sum_{j=1}^{v} \frac{|D_{a_j}|}{|D|} \text{Info}\big(D_{a_j}\big), \tag{11}$$

where $|\cdot|$ stands for the number of instances in the data set used as its argument (Quinlan, 1993).

We normalize the Gain measure using the Split information associated with a particular partitioning defined as:

$$\text{Split}\big(D, x^{(m)}\big) = -\sum_{j=1}^{v} \frac{|D_{a_j}|}{|D|} \ln\left(\frac{|D_{a_j}|}{|D|}\right). \tag{12}$$

[8] The source code can be obtained at http://www.cse.unsw.edu.au/~quinlan/.

Using Gain as a splitting criterion would favor partitioning based on discrete predictors having many values. Normalizing by the Split information specifies the desirability of a particular partitioning as the ratio of Gain and Split. Define the Gain ratio as (Quinlan, 1993):

$$\text{Gain ratio}\left(D,x^{(m)}\right) = \frac{\text{Gain}\left(D,x^{(m)}\right)}{\text{Split}\left(D,x^{(m)}\right)}. \tag{13}$$

We then assess the Gain ratio of every possible partitioning—specifically, based on a single discrete predictor—and among those with at least average Gain one can select the partitioning with the maximum Gain ratio (Quinlan, 1993). By recursively partitioning the training data at the leaf nodes until the stopping conditions are fulfilled, a full tree is grown.

To avoid the tree from overfitting the training data—that is, fitting the idiosyncracies of the training data that are not representative of the population—the full tree is pruned using C4.5's error-based pruning (see below) with the default confidence factor α = twenty-five percent (Quinlan, 1993). Pruning is performed by identifying subtrees that contribute little to the predictive accuracy and replacing each by a leaf or one of the branches. Of course, training error at a node—that is, estimating the true error by the number of wrongly classified training data instances at the node—is not an appropriate error estimate for making that evaluation. Since the tree is biased toward the training data that underlie its construction, training error is usually too optimistic an estimate of the true error. Instead, C4.5 goes for a more pessimistic estimate. Suppose a leaf covers N training data instances, E of which are misclassified. Then C4.5 equates the predicted error rate at a node with the upper bound of an α percent confidence interval assuming a binomial $\left(N, \frac{E}{N}\right)$ distribution at each node. This is known as C4.5's pessimistic error rate estimation. A subtree will then be pruned if the predicted error rate multiplied by the number of training data instances covered by the subtree's root node exceeds the weighted sum of error estimates for all its direct descendants. For details, see Quinlan (1993).

One can use decision trees to estimate probabilities of the form $p(t|x)$. These can then be used for constructing the ROC and calculating the AUROC. Such trees have been called class probability trees (Breiman, Friedman, Olshen, and Stone, 1984) or probability estimation trees (PETs) (Provost and Domingos, 2000). The simplest method that has been proposed to obtain estimates $\hat{p}(t = 1|x)$ uses the training data instance frequencies at the leaves—that is, a test data instance gets assigned the raw frequency ratio $\hat{p} = \frac{k}{l}$ at the leaf node to which it belongs, where k and l stand for the number of positive training data instances and the total number of training data instances at the leaf, respectively. However, since the tree was built to separate the classes and to make the leaves as homogeneous as possible, the raw estimates $\hat{p}$ systematically tend to be too extreme at the leaves—that is, they are systematically shifted toward 0 or 1. Furthermore, when the number of training data instances associated with the leaf is small, frequency counts provide unreliable probability estimates.[9]

Several methods have been proposed to overcome this, including applying simple Laplace correction and m-estimation smoothing (Buntine, 1992; Cussens, 1993; Oliver

[9] In principle, pruning and stopping conditions requiring a minimum number of training data instances at the tree leaves counter this.

and Hand, 1995; Simonoff, 1998; Zadrozny and Elkan, 2001a). For two-class problems, simple Laplace correction replaces $\hat{p}$ by $\hat{p}' = \frac{k+1}{l+2}$. This smoothes the data instance score toward 0.5, making it less extreme depending on the number of training data instances at the leaf. This approach has been applied successfully (see, for example, Provost and Domingos (2000); Zadrozny and Elkan (2001a)). However, from a Bayesian perspective, conditional probability estimates should be smoothed toward the corresponding unconditional probability—that is, the prior or base rate (Zadrozny and Elkan, 2001a). This can be done using m-estimation smoothing instead of Laplace correction. m-estimation replaces $\hat{p}$ by $\hat{p}' = \frac{k+bm}{l+m}$, where b is the base rate estimate $\hat{p}(t = 1)$—specifically, we assume that the prior class proportions in the data are representative of the true class priors— and m is a hyperparameter that controls how much the raw score is shifted toward b. Given a base rate estimate b, we can follow the suggestion by Zadrozny and Elkan (2001a) and specify m such that $bm = 5$, approximately. This heuristic is motivated by its similarity to the rule of thumb that says that a chi-squared goodness-of-fit test is reliable if the number of data instances in each cell of the contingency table is at least five.

Bayesian Learning Multilayer Perceptron Neural Network Classifiers

For binary classification, one commonly opts for a feed-forward multilayer perceptron (MLP) neural network with one hidden layer consisting of n_n neurons—that is, processing units accepting signals from the previous layer and transforming them into a single output signal—and an output layer consisting of one neuron (Bishop, 1995). Given a training set $D = \{x_i, t_i\}_{i=1}^{N}$ with input vectors $x_i = (x_i^{(1)}, \ldots, x_i^{(n)})^T \in \mathbb{R}^n$ and target labels $t_i \in \{0,1\}$, the MLP classifier then performs the following function mapping. The output of hidden neuron j, that is, $h^{(j)}(x)$, and the output of the output layer, that is, $y(x)$, are computed as:

$$\begin{aligned} \text{Hidden Layer: } h^{(j)}(x) &= f_1\left(b_1^{(j)} - \sum_{k=1}^{n} u^{(j,k)} x^{(k)}\right) \\ \text{Output Layer: } y(x) &= f_2\left(b_2 - \sum_{k=1}^{n_h} v^{(k)} h^{(k)}(x)\right) \end{aligned} \tag{14}$$

where $b_1^{(j)}$ is the bias corresponding to hidden neuron j, $u^{(j,k)}$ denotes the weight connecting input $x^{(k)}$ to hidden neuron j, b_2 is the output bias, and $v^{(k)}$ denotes the weight connecting hidden neuron k to the output neuron. The biases—that is, the bias vector b_1 and bias b_2—and weights—that is, the weight matrix u and weight vector v—together make up weight vector w. f_1 and f_2 are transfer functions and essentially allow the network to perform complex nonlinear function mappings. In the hidden layer, we use hyperbolic tangent transfer functions. Logistic transfer function is used in the output layer.[10] The latter allows the MLP's output $y(x)$ to be interpreted as an estimated probability of the form $p(t = 1|x)$ (Bishop, 1995). We randomly initialize and iteratively adjust w so as to minimize a cross-entropy objective function G defined as (Bishop, 1995):

$$G = -\sum_{i=1}^{N} \left(t_i \ln(y_i) + (1 - t_i) \ln(1 - y_i)\right). \tag{15}$$

[10] The transfer functions in the hidden and output layer are standard choices.

We may avoid overfitting the training data by adding automatic relevance determination (ARD) weight regularization to the objective function as (Bishop, 1995; MacKay, 1992a, 1992b):

$$F(w) = G + \sum_m \alpha_m E_{W(m)}. \tag{16}$$

where $E_{W(m)} = \frac{1}{2}\sum_j (w^{(j)})^2$, with j running over all weights of weight class $W(m)$. The formulation in (16) considers $n + 3$ weight classes within weight vector w, each associated with one weight decay hyperparameter α_m: one weight decay hyperparameter equation is associated with each group of weights corresponding to the connections from an input $x^{(m)}$ to the hidden layer; one hyperparameter is associated with the hidden layer biases; one hyperparameter is associated with the connections from the hidden layer neurons to the output neuron; and one hyperparameter is associated with the output bias. The ARD specification allows us to control the size of the weights associated with the connections out of each input. This setup is considered the right thing to do when some inputs may be less relevant than others. One of the main advantages of ARD is that it allows us to include a large number of potentially relevant inputs without damaging effects (Bishop, 1995; MacKay, 1992a, 1992b; Neal, 1998).

The Bayesian learning paradigm has been suggested for dealing with the above setup in a systematic way during MLP training (Bishop, 1995; MacKay, 1992a, 1992b). Basically, all prior assumptions are made explicit, and the weights and hyperparameters are determined by applying Bayes' theorem to map the prior assumptions into posterior knowledge after having observed the training data. For this study we adopt the evidence framework due to MacKay (1992a, 1992b) as an example implementation of Bayesian MLP learning. The MLP analyses were carried out using the Netlab toolbox for Matlab implemented by Bishop (1995) and Nabney (2001).[11]

Let $p(w|\alpha,H)$ be the prior probability density over the weights w given the MLP's architecture or functional form H and the hyperparameter vector α, the vector of all hyperparameters α_m. When we observe the training data D, we adjust the prior density to a posterior density using Bayes' theorem (level-1 inference). This gives:

$$p(w|D,\alpha,H) = \frac{p(D|w,H)p(w|\alpha,H)}{p(D|\alpha,H)}. \tag{17}$$

In the above expression, $p(D|w,H)$ is the likelihood function—that is, the probability of the data D occurring given the weights w and the MLP's functional form H. In Equation (17) $p(D|\alpha,H)$ is the evidence for α that guarantees that the righthand side of the equation integrates to one over the weight space.

Obtaining good predictive models is dependent on the use of the right priors. MacKay (1992a, 1992b) uses Gaussian priors. One can then choose the most probable weights w^{MP} given the current α so as to maximize the posterior density $p(w|D,\alpha,H)$. It can be shown that the weights w^{MP}, given the current setting of α, are found by minimizing the objective function $F(w)$ in Equation (16) (MacKay, 1992a, 1992b). One can use standard optimization methods to perform this task. We used a scaled conjugate gradient method to optimize the regularized cross-entropy objective function $F(w)$ (Bishop, 1995).

[11] The source code can be obtained at http://www.ncrg.aston.ac.uk/netlab/.

The hyperparameter vector α can also be optimized by applying Bayes' theorem (level-2 inference). This yields:

$$p(\alpha|D,H) = \frac{p(D|\alpha,H)p(\alpha|H)}{p(D|H)}. \tag{18}$$

Assume that all weights of weight class $W(m)$ are distributed according to a Gaussian prior with mean 0 and variance $\sigma_m^2 = \frac{1}{\alpha_m}$. This is consistent with the rationale that inputs associated with large hyperparameter values are less relevant to the network. Starting from Equation (18) and assuming a uniform (uninformative) prior $p(\alpha|H)$, we obtain the most probable hyperparameters α^{MP} by maximizing the likelihood function $p(D|\alpha,H)$. Optimization is discussed in detail in (Bishop, 1995; MacKay, 1992a, 1992b). Basically, the α hyperparameter vector is randomly initialized and the network is then trained in the usual manner, with the novelty that training is periodically halted for the weight decay hyperparameters to be updated (Bishop, 1995; MacKay, 1992a, 1992b).

MacKay argues in favor of moderating the output of the most probable network in relation to the error bars around w^{MP}, so that these point estimates may better represent posterior probabilities of class membership given the data D. The moderated output is similar to the most probable output in regions where the data are dense. Where the data are more sparse, moderation smoothes the most probable output toward a less extreme value—specifically, 0.5—[12] reflecting the uncertainty in sparse data regions. We will not go into the details of the approximation to the moderated probability proposed by MacKay. For details, see MacKay (1992a, 1992b).

One can is also choose between network architectures in a Bayesian way, using the evidence $p(D|H)$ attributed to an architecture H (MacKay, 1992a, 1992b) (level-3 inference). Models can be ranked according to their evidence. However, Roberts and Penny (1999) empirically show that for larger data sets, the training error is as good a measure for model selection as is the evidence. For further details on Bayesian MLP learning, see Bishop (1995) and MacKay (1992a, 1992b).

Least-Squares Support Vector Machine Classifiers

Given a training set $D = \{x_i,t_i\}_{i=1}^N$ with input vectors $x_i = (x_i^{(1)},\ldots,x_i^{(n)})^T \in \mathbb{R}^n$ and target labels $t_i \in \{-1,+1\}$, the support vector machine (SVM) classifier, according to Vapnik's original formulation (Cristianini and Shawe-Taylor, 2000; Vapnik, 1995, 1998), satisfies the following conditions:

$$\begin{cases} w^T\varphi(x_i) + b \geq +1, & \text{if } t_i = +1 \\ w^T\varphi(x_i) + b \leq -1, & \text{if } t_i = -1 \end{cases} \tag{19}$$

which is equivalent to:

$$t_i\left[w^T\varphi(x_i) + b\right] \geq 1, \quad i = 1,\ldots,N, \tag{20}$$

[12] As for the simple Laplace correction proposed in the context of decision trees, the original posterior class membership probability estimates are smoothed toward 0.5. As stated in the third part of this section, from a Bayesian point of view, it is better to smooth the estimates toward the corresponding prior. However, the approximation to the moderated posterior probability proposed by MacKay is not readily adaptable to this requirement.

Where w represents the weight vector and b the bias. The nonlinear function $\varphi(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{n_h}$ maps the input or measurement space to a high-dimensional, and possibly infinite-dimensional, feature space. Equation (20) then comes down to the construction of two parallel bounding hyperplanes at opposite sides of a separating hyperplane $w^T\varphi(x) + b = 0$ in the feature space, with the margin width between both hyperplanes equal to $\frac{2}{\|w\|^2}$. In primal weight space, the classifier then takes the form:

$$y^t(x) = \text{sign}(w^T\varphi(x) + b), \tag{21}$$

but, on the other hand, it is never evaluated in this form.

One defines the optimization problem as:

$$\min_{w,b,\xi} \mathcal{J}(w,\xi) = \frac{1}{2}w^Tw + c\sum_{i=1}^{N}\xi_i \tag{22}$$

subject to:

$$\begin{cases} t_i\left(w^T\varphi(x_i) + b\right) \geq 1 - \xi_i, & i = 1,\ldots,N \\ \xi_i \geq 0, & i = 1,\ldots,N \end{cases}. \tag{23}$$

The variables ξ_i are slack variables needed to allow misclassifications in the set of inequalities. $c \in \mathbb{R}^+$ is a tuning hyperparameter, weighting the importance of classification errors vis-à-vis the margin width. The solution of the optimization problem is obtained after constructing the Lagrangian. From the conditions of optimality, one obtains a quadratic programming (QP) problem in the Lagrange multipliers α_i (Vapnik, 1995, 1998). A multiplier α_i exists for each training data instance. Data instances corresponding to non-zero α_i are called support vectors.

As is typical for SVMs, we never calculate w or $\varphi(x)$. This is made possible due to Mercer's condition, which relates the mapping function $\varphi(\cdot)$ to a kernel function $K(\cdot,\cdot)$ as follows (Cristianini and Shawe-Taylor, 2000; Vapnik, 1995, 1998):

$$K(x_i,x_j) = \varphi(x_i)^T\varphi(x_j). \tag{24}$$

For the kernel function $K(\cdot,\cdot)$ one typically has several design choices, such as $K\left(x_i,x_j\right) = x_i^Tx_j$ (linear kernel), $K(x_i,x_j) = (x_i^Tx_j + 1)^d$ (polynomial kernel of degree d), and $K(x_i,x_j) = \exp\left\{\frac{-\|x_i-x_j\|_2^2}{\sigma^2}\right\}$ (radial basis function (RBF) kernel), where $d \in \mathbb{N}$ and $\sigma \in \mathbb{R}^+$ are constants. Then construct the SVM classifier as:

$$y^t(x) = \text{sign}\left(\sum_{i=1}^{N}\alpha_it_iK(x,x_i) + b\right). \tag{25}$$

Optimization is discussed in Cristianini and Shawe-Taylor (2000) and Vapnik (1995, 1998).

Vapnik's SVM classifier formulation was modified by Suykens and Vandewalle (1999) into the following least-squares or LS-SVM formulation:

$$\min_{w,b,e} \mathcal{J}(w,e) = \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{i=1}^{N} e_i^2, \tag{26}$$

subject to the following equality constraints:

$$t_i\left(w^T \varphi(x_i) + b\right) = 1 - e_i, \quad i = 1, \ldots, N. \tag{27}$$

This formulation now consists of equality instead of inequality constraints and takes into account a squared error with a regularization term similar to ridge regression. The solution is obtained after constructing the Lagrangian. The formulation as it stands results in a linear system instead of a QP problem (Suykens and Vandewalle, 1999). Again, one has several choices for the kernel function. For this study, we report on LS-SVM classification using $K(x,x_i) = x^T x_i$ (linear kernel), $K(x,x_i) = (x^T x_i + 1)^d$ (polynomial kernel of degree $d = 2$ and $d = 3$), and $K(x,x_i) = \exp\left\{\frac{-\|x-x_i\|_2^2}{\sigma^2}\right\}$ (RBF kernel). Notice that Mercer's condition holds for all $\sigma \in \mathbb{R}^+$ and $d \in \mathbb{N}$ values in the RBF and polynomial cases. The LS-SVM classifier is constructed as in Equation (25). Optimization is discussed in Suykens, Lukas, Van Dooren, De Moor, and Vandewalle (1999). The objective function hyperparameter γ and the RBF kernel hyperparameter σ are tuned automatically using the training data by means of the grid search mechanism described in Van Gestel, Suykens, Baesens, Viaene, Vanthienen, Dedene, De Moor, and Vandewalle (2000). The output of the LS-SVM classifier before applying the sign operator is used for ROC analysis. [13]

Naive Bayes and Tree-Augmented Naive Bayes Classifiers

A Bayesian network represents a joint probability function over a set of variables. It is a probabilistic white-box model consisting of a qualitative part specifying the conditional independencies between the variables and a quantitative part specifying conditional probabilities on the set of variables (Friedman, Geiger, and Goldszmidt 1997; Pearl, 1998). A Bayesian network is visualized as a directed acyclic graph between nodes (variables) whereby each directed edge represents a probabilistic dependency from a parent node to a probabilistically dependent child node. Bayesian networks use Bayes' theorem to compute the probability of one node having a particular value given values assigned to the other nodes. Hence, one may adopt Bayesian networks as classifiers by computing the posterior probability of a class node given the values of the predictor nodes. In this configuration, the class node acts as a parent to all the

[13] We will not discuss the issue of producing well-calibrated posterior class membership probability estimates for SVM classifiers. Having exact estimates of the latter is not strictly necessary for ROC analysis. In Platt (2000), a method is presented for fitting a monotone sigmoid transformation to the output of the SVM classifier before applying the sign operator. The SVM+sigmoid combination proves capable of producing posterior class membership probability estimates that are of comparable quality to a regularized likelihood kernel method. Note that, since these probability estimates are a monotonic increasing transformation of the continuous SVM classifier output, we would obtain the same ROC with or without this transformation.

predictor nodes. The two major tasks in learning a Bayesian network are structure learning, the qualitative part, and parameter learning, the quantitative part. All analyses were carried out using the Bayes Net toolbox for Matlab by Murphy (2001).[14] In the rest of this section we will focus on naive Bayes and TAN Bayes classification.

The simplest form of Bayesian network classifier is known as the naive, or simple, Bayes classifier (Domingos and Pazzani, 1997; Duda, Hart, and Stork, 2001; Friedman, Geiger, and Goldszmidt, 1997; Hand, 1992; Webb 1999). Given a training set $D = \{x_i, t_i\}_{i=1}^N$ with input vectors $x_i = (x_i^{(1)}, \ldots, x_i^{(n)})^T \in \mathbb{R}^n$ and target labels $t_i \in \{0,1\}$, the simplifying structural assumption underlying the naive Bayes classifier (and avoiding structure learning) is the conditional independence of predictors $x^{(m)}$ given the class label t. It follows that:

$$p(x|t) = \prod_{m=1}^{n} p\left(x^{(m)}|t\right). \tag{28}$$

We estimate the probabilities on the righthand side of Equation (28) using the training data. For discrete data we specify these conditional density estimates as conditional probability tables—that is, we specify them using frequency counts on the training set (with continuous predictors discretized as specified in the second section) (Domingos and Pazzani, 1997; Duda, Hart, and Stork, 2001; Friedman, Geiger, and Goldszmidt, 1997; Hand, 1992; Webb 1999). One can then easily compute the posterior class membership probability estimates $\hat{p}(t|x)$ used for classification by using Bayes' theorem:

$$\hat{p}(t|x) = \frac{\hat{p}(t)\hat{p}(x|t)}{\hat{p}(x)}. \tag{29}$$

We estimate $p(t)$ from the data—that is, we assume that the prior class proportions in the data are representative of the true class priors. Instead of computing $\hat{p}(x)$ in Equation (29), we normalize the numerator on the righthand side of Equation (29) by demanding that $\sum_{t=0}^{1} \hat{p}(t|x) = 1$. The naive Bayes classifier coincides with the Bayes optimal classifier only if all predictors are in fact independent given the class. However, even in domains where this model bias is *a priori* considered inappropriate, the naive Bayes classifier often outperforms more powerful classifiers (Domingos and Pazzani, 1997; Duda, Hart, and Stork, 2001; Friedman, Geiger, and Goldszmidt, 1997; Hand, 1992; Webb 1999).

More complex interactions between predictors may be represented by augmented naive Bayes networks by allowing directed edges (probabilistic dependencies) between predictors, dispensing with the strong independence assumption of naive Bayes. Friedman Geiger, and Goldszmidt (1997) propose TAN networks, which are characterized by the structural restriction that the class node has no parents and each predictor has as its parents the class node and at most one other predictor. This gives:

$$p(x|t) = \prod_{m=1}^{n} p\left(x^{(m)}|\pi^{(m)}\right), \tag{30}$$

where $\pi^{(m)}$ is the set of parents of the node corresponding to predictor $x^{(m)}$. The procedure for learning the TAN structure from the training data is based on a method due to Chow and Liu (Pearl, 1988; Webb 1999).

[14] The source code can be obtained at http://www.cs.berkeley.edu/~murphyk/Bayes/.

Probability estimates $\hat{p}\left(x^{(m)}|\pi^{(m)}\right)$ based on raw frequency counts after partitioning the training data according to all possible (discrete) values of $\pi^{(m)}$ may be unreliable in sparsely populated partitions. Therefore, one can apply an additional smoothing operation based on uniform Dirichlet prior smoothing with confidence factor N^0 (Friedman, Geiger, and Goldsmidt, 1997)—that is, for a data partition consisting of c data instances and for a predictor $x^{(m)}$ having v discrete values, compute smoothed estimates as:

$$\hat{p}_s\left(x^{(m)}|\pi^{(m)}\right) = \frac{c}{c+N^0}\hat{p}\left(x^{(m)}|\pi^{(m)}\right) + \frac{N^0}{c+N^0}\frac{1}{v}. \tag{31}$$

The smoothed model has the same qualitative structure as the original model but has different numerical parameters. See Friedman, Geiger, and Goldszmidt (1997) for details.

RESULTS AND DISCUSSION

An overview of the results of our benchmarking study are presented in this section. In the first part of this section we look at the performance assessment tools. In the second part of the section we discuss the results in detail.

Performance Assessment Tools

We can evaluate and contrast the predictive power of the algorithms in several ways. Within the framework of this article we report on algorithm performance, looking at the following complementary assessment tools:

- The absolute difference in performance point estimates using ten-fold cross-validation

One way of comparing algorithm performance is by simply contrasting estimated single-figure predictive performance indicators for the algorithms using ten-fold cross-validation. Table 5 reports the mean PCC performance for the benchmarked algorithms using the ten-fold cross-validation setup described in the second part of the third section for the alternative target encoding schemes of the second section—that is, 1+, 4+, 7+, *vnl* and *vop*. Table 6 does the same for the mean AUROC performance. Each algorithm in the leftmost column is evaluated on the indicator predictors only (that is, indicator models) as well as on the extended predictor set containing both indicator and non indicator predictors (that is, extended predictor models). For several of the algorithm types discussed in the fourth section we chose to report and comment on alternative operationalizations. The Majority classifier assigns the majority class in the training set to all data instances in the test set. This plurality rule ignores the information in the predictors and stands for the benchmark classification of minimal work. The highest performance per evaluation scenario (per column) is boldface.

- A comparison of algorithm differences using Duncan's multiple range test

We can use an approximate statistical test that explicitly makes use of the blocked nature of the ten-fold cross-validation resampling discussed in the second part of the third section for comparing the performance of the classification algorithms—specifically, we use a two-way ANOVA and Duncan's multiple range test. In order not to blur the general picture, we do not include the complete algorithm subgroup specification produced by Duncan's multiple range test for each of the evaluation scenarios, but rather underline those algorithm means that do not significantly differ from the

algorithm with the highest performance for that evaluation scenario according to Duncan's multiple range test using a 5 percent significance level.

In addition, we include two extra columns and two extra rows in Tables 5 and 6 summarizing the information from Duncan's multiple range test: R, R^g, C, and C^g. Let R be a summary measure per algorithm (per row) that stands for the ratio of, in the numerator, the number of times the algorithm's performance is not significantly different from the boldface one and, in the denominator, the number of reference evaluation scenarios. Let R^g be a summary ratio per algorithm type, specifically, the algorithm types are separated by a blank horizontal line. This summary measure reflects the ratio of, in the numerator, the number of evaluation scenarios for which at least one operationalization can attain performance statistically similar to the highest according to Duncan's multiple range test and, in the denominator, the number of reference evaluation scenarios. The meaning of the rows labeled C and C^g is similar, but now we summarize performance per evaluation scenario—that is, per column. Let C stand for the cardinality of Duncan's algorithm subgroup with the highest performance over the total number of nontrivial algorithms in the comparison—that is, excluding the Majority classifier. Let C^g be the ratio of, in the numerator, the number of algorithm types for which at least one operationalization can attain performance statistically similar to the highest according to Duncan's multiple range test and, in the denominator, the number of algorithm types in the comparison.

- A comparison of algorithm type ROC convex hulls

The ROC provides an interesting visual tool for assessing the classification performance of the algorithms in function of a range of operating conditions (Bradley, 1997; Provost and Fawcett, 2001; Provost, Fawcett, and Kohavi, 1998). As noted in the first part of the third section, a potential criticism of the comparison and evaluation of algorithms on the basis of the AUROC is that a whole-curve, single-number measure is optimized. Under specific operating conditions, an algorithm with maximum AUROC may well be suboptimal. Thus, in addition to considering the results in Tables 5 and 6, it may be interesting to look at the ROCs themselves. For the purposes of this article we have aggregated the ROCs per algorithm type (for each of the target encoding schemes and for both indicator models and extended predictor models) by constructing the convex hull (Provost and Fawcett, 2001) of all cross-validated model ROCs (see the second part of the third section) of the algorithm type's operationalizations. This allows us to compare algorithm types visually by means of their ROC convex hull.[15] The convex hull clearly embodies information about algorithm type performance that is not available from the results in Table 6, where we are comparing the mean AUROC for separate algorithm type operationalizations. By constructing the convex hull per algorithm type, we identify a subset of points from among the ROC points of its operationalizations that are potentially optimal. The area under the convex hull is then, per definition, greater than or equal to the AUROC performance

[15] Notice that the use of the ROC convex hull semantics for model comparison and selection implies an assumption of potential *optimality* of points on the convex hull as articulated in Provost and Fawcett (2001). Provost and Fawcett (2001) show that the ROC convex hull visual semantics are robust under a wide variety of realistic optimization measures, including the minimum expected cost and the Neyman-Pearson criterion.

TABLE 5
Mean PCC Using Ten-Fold Cross-Validation

	Indicator Models					Extended Predictor Models						
	1+	4+	7+	*vnl*	*vop*	1+	4+	7+	*vnl*	*vop*	R	R^g
logit	68.12	76.34	90.99	70.77	**84.92**	79.49	**79.20**	90.71	79.70	83.35	10/10	10/10
C4.5^s	66.26	74.77	91.21	68.91	83.70	76.13	76.98	91.21	77.70	83.70	8/10	8/10
C4.5^s_u	65.12	74.41	91.21	68.48	83.70	71.77	73.62	91.21	73.70	83.70	5/10	
C4.5^s_c	65.05	74.84	91.21	67.69	83.70	74.70	75.91	91.21	75.70	83.70	5/10	
1NN	62.55	71.34	88.63	65.48	80.77	70.19	75.55	87.56	73.20	80.41	0/10	9/10
10NN	64.76	73.20	**91.64**	67.19	83.56	75.27	77.27	91.42	77.70	82.99	6/10	
100NN	66.62	73.91	91.21	70.05	83.70	77.84	75.91	91.21	77.63	83.77	8/10	
500NN	58.61	71.69	91.21	63.97	83.70	67.55	71.69	91.21	65.90	83.70	4/10	
MLP1	67.83	76.34	91.35	70.98	**84.92**	79.27	78.41	91.21	**79.84**	84.20	10/10	10/10
MLP2	67.83	**76.48**	91.28	70.55	84.56	78.99	78.06	91.07	79.77	83.92	10/10	
MLP3	67.55	76.13	90.78	70.26	84.56	78.41	78.77	90.71	79.34	**84.70**	10/10	
Lin-LS-SVM	**68.76**	75.98	91.49	**71.27**	84.49	**79.63**	**79.20**	**91.49**	**79.84**	**84.70**	10/10	10/10
Poly2-LS-SVM	65.76	75.27	89.64	70.19	83.20	73.12	76.48	87.92	75.55	80.70	2/10	
Poly3-LS-SVM	63.19	71.27	87.78	66.41	80.20	72.55	74.34	86.70	74.27	77.98	0/10	
RBF-LS-SVM	67.91	76.34	91.42	**71.27**	84.42	79.27	78.84	91.35	79.06	84.35	10/10	
NB	67.26	73.98	88.56	69.48	81.49	77.06	76.70	83.70	77.84	79.13	3/10	5/10
NB$^s_5$	67.26	74.27	88.92	69.41	81.99	77.63	77.34	83.77	78.91	79.49	5/10	
NB$^s_{25}$	67.33	73.91	89.56	69.76	82.92	77.48	76.98	84.77	78.70	79.06	5/10	
NB$^s_{50}$	67.05	73.41	89.85	69.76	83.20	77.27	77.27	87.28	78.63	80.06	5/10	

TAN	66.91	73.91	88.71	69.48	81.92	75.27	76.56	82.63	77.84	78.48	3/10	5/10
TAN$^s_5$	66.98	74.20	88.99	70.12	82.49	76.41	77.84	86.85	78.27	81.20	4/10	
TAN$^s_{25}$	67.19	74.48	89.92	70.34	82.92	76.98	77.63	87.35	78.70	81.49	5/10	
TAN$^s_{50}$	67.41	74.34	89.99	70.26	83.13	77.20	77.70	87.92	79.13	81.63	4/10	
C	15/23	11/23	12/23	17/23	12/23	10/23	14/23	12/23	17/23	12/23		
C^g	6/7	5/7	5/7	7/7	5/7	5/7	7/7	5/7	7/7	5/7		
Majority	57.04	71.69	91.21	63.97	83.70	57.04	71.69	91.21	63.97	83.70		

Classification algorithms: logistic regression (logit); *m*-estimation smoothed C4.5 (C4.5^s), *m*-estimation smoothed and unpruned C4.5 (C4.5^s_u), *m*-estimation smoothed and curtailed C4.5 (C4.5^s_c); 1-nearest neighbor (1NN), 10-nearest neighbor (10NN), 100-nearest neighbor (100NN), 500-nearest neighbor (500NN); MLP having one output neuron and one hidden layer with up to three hidden layer neurons (MLP1,MLP2,MLP3); linear kernel LS-SVM (Lin-LS-SVM), polynomial kernel LS-SVM of degree 2 and 3 (Poly2-LS-SVM, Poly3-LS-SVM), RBF kernel LS-SVM (RBF-LS-SVM); naive Bayes (NB), naive Bayes with Dirichlet smoothing confidence factor 5 (NB$^s_5$), 25 (NB$^s_{25}$), and 50 (NB$^s_{50}$); TAN Bayes, TAN Bayes with Dirichlet smoothing confidence factor 5 (TAN$^s_5$), 25 (TAN$^s_{25}$), and 50 (TAN$^s_{50}$); majority classifier (Majority).

TABLE 6
Mean AUROC Using Ten-Fold Cross-Validation

	Indicator Models					Extended Predictor Models						
	1+	4+	7+	*vnl*	*vop*	1+	4+	7+	*vnl*	*vop*	R	R^s
logit	74.82	77.46	76.99	76.20	77.93	86.46	86.47	83.14	88.24	84.52	10/10	10/10
C4.5^s	71.60	73.23	55.32	71.17	62.97	82.20	80.81	54.64	82.51	78.17	0/10	0/10
C4.5^s_u	70.78	73.67	71.03	72.89	72.65	80.61	80.53	76.92	84.40	75.78	0/10	
C4.5^s_c	70.41	74.57	69.78	72.63	73.04	82.83	83.41	76.28	84.30	81.73	0/10	
1NN	61.40	65.92	56.80	64.47	61.65	69.56	69.30	57.54	71.03	62.97	0/10	10/10
10NN	69.63	74.40	70.22	72.64	72.79	82.50	83.77	72.21	84.96	76.73	0/10	
100NN	72.59	75.79	76.09	75.40	77.30	85.98	85.61	81.96	87.68	83.92	9/10	
500NN	73.15	76.26	77.64	75.79	77.86	84.82	84.22	81.93	86.49	83.01	10/10	
MLP1	74.55	77.62	76.70	75.67	78.47	86.36	86.52	83.64	88.45	85.02	10/10	10/10
MLP1^s	74.55	77.58	77.38	75.69	78.45	86.38	86.56	**83.81**	**88.49**	**85.20**	10/10	
MLP2	74.17	77.59	76.95	75.47	78.33	86.30	86.50	82.85	88.07	84.89	10/10	
MLP2^s	74.18	77.59	77.43	75.49	78.39	86.31	**86.58**	82.90	88.10	85.00	10/10	
MLP3	73.85	76.86	75.44	75.35	77.17	85.51	85.95	80.67	87.69	84.25	10/10	
MLP3^s	73.97	77.03	76.13	75.37	77.09	85.62	86.03	81.88	87.76	84.83	10/10	
Lin-LS-SVM	75.00	77.43	77.90	**76.42**	78.03	86.50	86.51	82.36	88.33	84.66	10/10	10/10
Poly2-LS-SVM	70.83	73.11	68.40	74.42	71.89	80.10	78.04	64.32	80.23	68.95	1/10	
Poly3-LS-SVM	61.10	60.85	54.80	60.47	59.11	71.67	67.90	58.28	72.21	58.74	0/10	
RBF-LS-SVM	**75.03**	**77.69**	78.18	**76.42**	78.27	**86.52**	86.53	82.59	88.14	84.39	10/10	

NB	72.97	76.33	73.98	73.95	78.20	83.19	82.48	71.91	75.53	77.70	4/10	10/10
NB$^s_5$	73.07	76.73	77.74	75.40	78.33	85.09	85.09	80.45	87.03	82.96	10/10	
NB$^s_{25}$	73.00	76.69	77.92	75.35	78.15	85.04	84.98	81.13	86.83	83.34	10/10	
NB$^s_{50}$	72.85	76.57	77.14	75.21	77.73	84.88	84.82	80.50	86.77	83.19	9/10	
TAN	71.37	74.69	71.05	73.95	75.98	73.84	74.61	60.72	75.53	65.17	1/10	9/10
TAN$^s_5$	72.65	76.46	78.45	75.82	78.43	83.24	83.69	76.78	85.76	80.90	4/10	
TAN$^s_{25}$	72.92	76.47	**78.72**	75.87	**78.52**	84.15	84.27	79.10	86.45	82.33	8/10	
TAN$^s_{50}$	72.92	76.46	78.35	75.76	78.09	84.46	84.36	79.29	86.72	82.47	9/10	
C	13/26	18/26	18/26	18/26	19/26	15/26	16/26	16/26	16/26	16/26		
C^g	5/7	6/7	6/7	6/7	6/7	6/7	6/7	6/7	6/7	6/7		
Majority	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00		

Classification algorithms: logistic regression (logit); *m*-estimation smoothed C4.5 (C4.5^s), *m*-estimation smoothed and unpruned C4.5 (C4.5^s_u), *m*-estimation smoothed and curtailed C4.5 (C4.5^s_c); 1-nearest neighbor (1NN), 10-nearest neighbor (10NN), 100-nearest neighbor (100NN), 500-nearest neighbor (500NN); MLP having one output neuron and one hidden layer with up to three hidden layer neurons (MLP1,MLP2,MLP3) and smoothed versions (MLP1^s,MLP2^s,MLP3^s); linear kernel LS-SVM (Lin-LS-SVM), polynomial kernel LS-SVM of degree 2 and 3 (Poly2-LS-SVM, Poly3-LS-SVM), RBF kernel LS-SVM (RBF-LS-SVM); naive Bayes (NB), naive Bayes with Dirichlet smoothing confidence factor 5 (NB$^s_5$), 25 (NB$^s_{25}$), and 50 (NB$^s_{50}$); TAN Bayes, TAN Bayes with Dirichlet smoothing confidence factor 5 (TAN$^s_5$), 25 (TAN$^s_{25}$), and 50 (TAN$^s_{50}$); majority classifier (Majority).

for each individual operationalization (Provost and Fawcett, 2001). The algorithm type ROC convex hull plots are taken up as Figures 2–11.

Discussion of the Performance

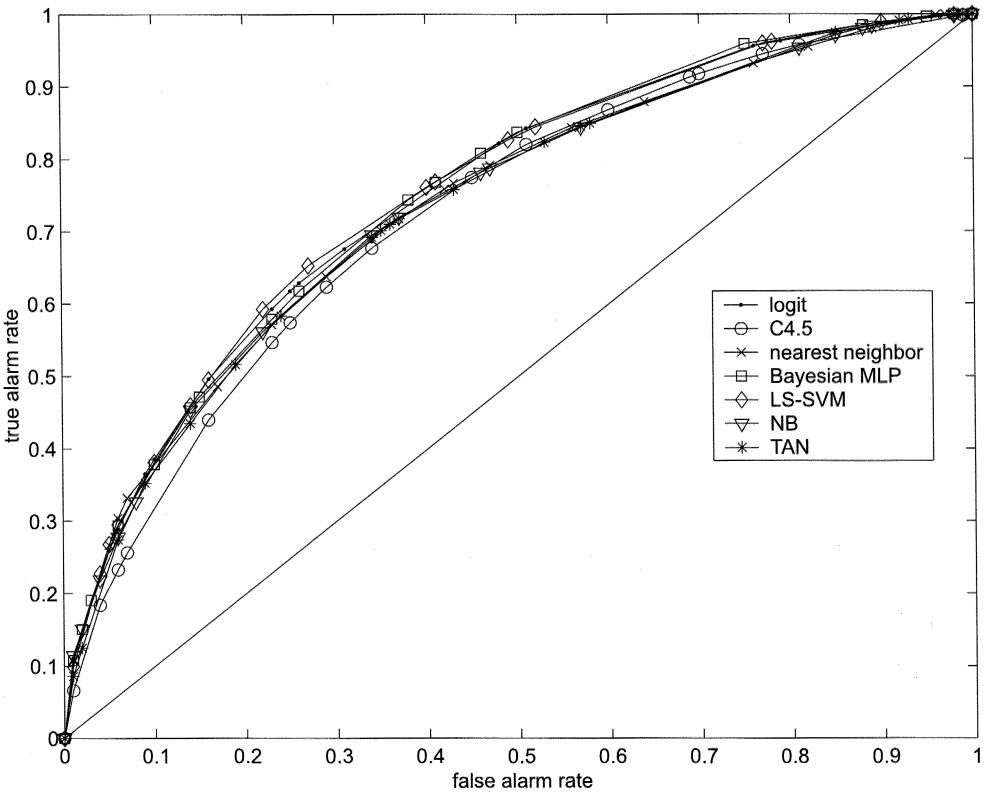
A comparison of the mean algorithm performances in Tables 5 and 6 in absolute terms and in terms of Duncan's multiple range difference assessment allows us to conclude that the performance (specifically, in terms of mean PCC and mean AUROC) of many algorithms is sufficiently similar to state that their differences are rather small. We should not attribute much weight to the (high) mean PCC for the more skewed target encoding schemes, *7+* and *vop*. For both cases, many of the underlined performances are either inferior to or equal to the performance of the Majority classifier of minimal work. As discussed in the first part of the third section, the AUROC may prove a more meaningful measure of classification or scoring performance for these cases in that it summarizes the tradeoff between the false positives and false negatives over the range of operating conditions and thus better reflects the work done by the algorithms in separating the data instances.

No algorithm type jumps out as consistently and significantly superior in terms of predictive performance over all considered evaluation scenarios. In fact, the set of competitive algorithms is quite large for each of the evaluation scenarios in Tables 5 and 6. We first take a closer look at the mean PCC results in Table 5 per evaluation scenario. The discussion makes abstraction of the mean PCC results for the columns labeled *7+* and *vop*. This leaves six relevant evaluation scenarios in Table 5. Summary measure C^8 indicates that in three out of six relevant evaluation scenarios all seven algorithm types have at least one operationalization among the best. The logistic regression (logit), the Bayesian learning MLP, and the LS-SVM algorithm types have at least one operationalization among the best for all relevant columns. The naive Bayes operationalizations fail to measure up to the best when trained on the indicator predictor set for 4+ but regain their place among the best when using the extended predictor set. Conversely, the TAN Bayes operationalizations are cast out of the set of best performers by Duncan's multiple range test when trained on the extended predictor set for 1+. The C4.5 decision tree operationalizations consistently fail for 1+.

The mean AUROC results presented in Table 6 expand the view on predictive performance, measuring the quality of an algorithm's classification work across all possible classification thresholds. Summary measure C^8 in Table 6 indicates that in nine out of ten evaluation scenarios at least six out of seven algorithm types have at least one operationalization among the best. The one column for which this is not the case is for 1+ indicator models. Here, neither the C4.5 decision tree nor the TAN Bayes operationalizations are counted among the best. However, with the addition of nonflag predictors, the TAN Bayes algorithm type regains its place at the top. Although the C4.5 decision trees clearly benefit from the addition of the nonflag predictors, they are not capable of keeping up with the best performers. And 1+ is not an exception: The C4.5 decision tree operationalizations report consistently inferior mean AUROC performance for all the evaluation scenarios.

Finally, to complete the picture, we should also have a look at Figures 2–11. The ROC convex hull per algorithm type allows us to visually monitor the classification

FIGURE 2
Algorithm Type ROC Curve Convex Hull for 1+ Indicator Models

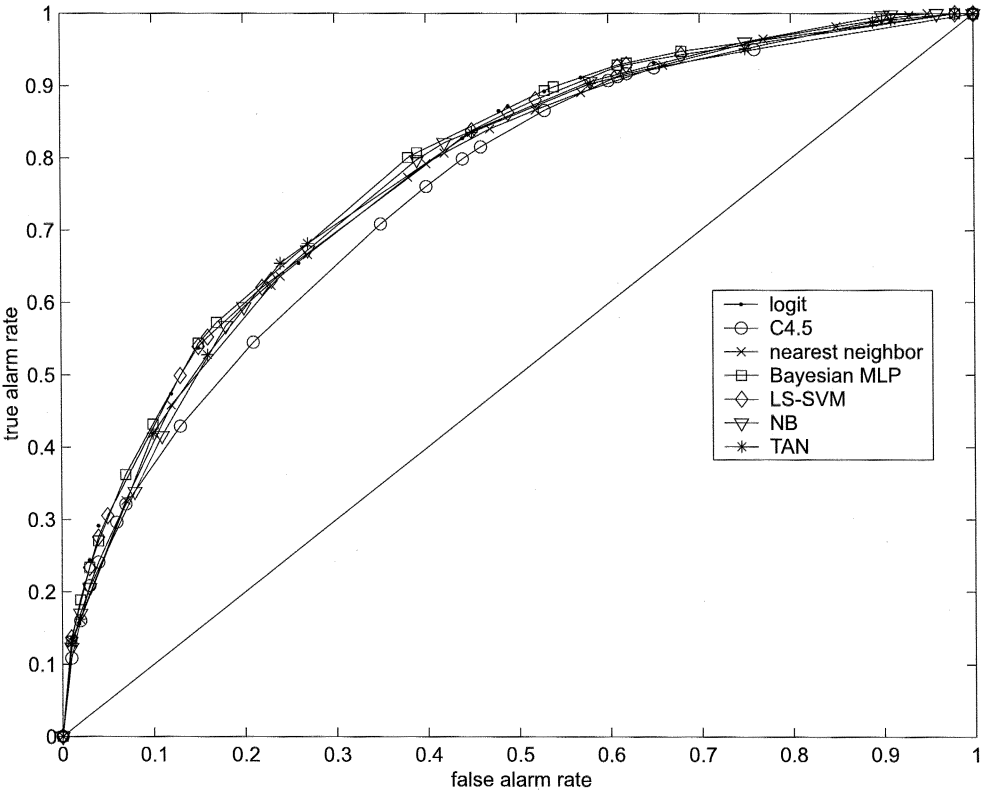


behavior of the algorithm types as the operating conditions change. No surprises here. Consistent with the mean AUROC performance patterns observed in Table 6, many of the algorithm type ROC convex hulls are quite close together over the whole range.[16] Intersections between the curves are not uncommon. This illustrates the point made in the first part of the third section that AUROC assessment is only considered appropriate when evaluating (continuous-output) classifiers under various operating conditions. Under specific operating conditions, ROC curve analysis, even though purely visual, may be more appropriate. Observe that, for all the depicted evaluation scenarios, the C4.5 decision tree hull is dominated in large part by most, and sometimes all, of the other algorithm type hulls.

Taking abstraction of the C4.5 decision trees—we will comment on their performance below—we may conclude that in terms of predictive performance we clearly have a

[16] For some evaluation scenarios the distance between the convex hulls in ROC space is nevertheless clearly larger than for others. For instance, contrast the evaluation scenarios for the 1+ and the 7+ target encoding schemes. The distance between curves is, however, to be evaluated in relation to the size of the spread of the mean AUROC values within the group of best performing algorithms—that is, those with boldfaced or underlined means—for the evaluation scenarios in Table 6. We should take into account this spread when comparing the set of ROCs from one scenario to the other.

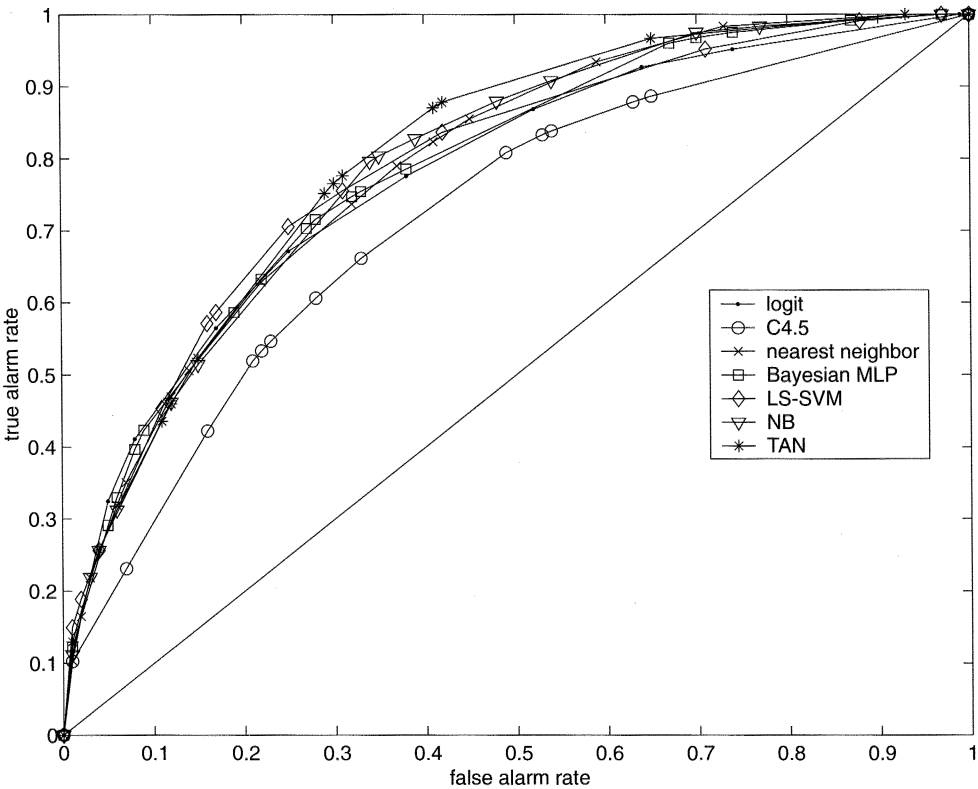
FIGURE 3
Algorithm Type ROC Curve Convex Hull for 4+ Indicator Models



set of competitive algorithm types that all seem up to the job. In this case, we may wish to select an algorithm (type) based on other criteria such as the time needed for training, tuning, or classification or the comprehensibility of the produced results. If we add time considerations to the equation, for instance, picking logit as the method of choice would be easily defensible. Not once did logit fail to rank among the best, as can be seen from the R^g summary measure in Tables 5 and 6. This assessment is confirmed by the ROC convex hulls displayed in Figures 2–11. Moreover, logit is a classification tool that has been well studied over the years. It also is readily available in most common off-the-shelf statistical or data mining packages.

Now we will take a closer look at the individual algorithm type performance. In Tables 5 and 6 we report on C4.5 with the default hyperparameter settings estimating posterior class membership probabilities at the tree leaves using *m*-estimation smoothing ($C4.5^s$), unpruned C4.5 with *m*-estimation smoothing at the leaves ($C4.5^s_u$), and curtailed C4.5 with *m*-estimation smoothing at the leaves ($C4.5^s_c$). $C4.5^s_c$ uses the unpruned tree and a form of *neighborhood sizing* to obtain posterior class membership probability estimates and to classify new observations. $C4.5^s_c$ is conceived as follows (Zadrozny and Elkan, 2001a, 2001b). First, a full (unpruned) C4.5 tree is grown. Then, instead of estimating class membership from the leaves of the unpruned tree (using *m*-estimation) as for $C4.5^s_u$, we backtrack through the parents of the leaf until we find

FIGURE 4
Algorithm Type ROC Curve Convex Hull for 7+ Indicator Models

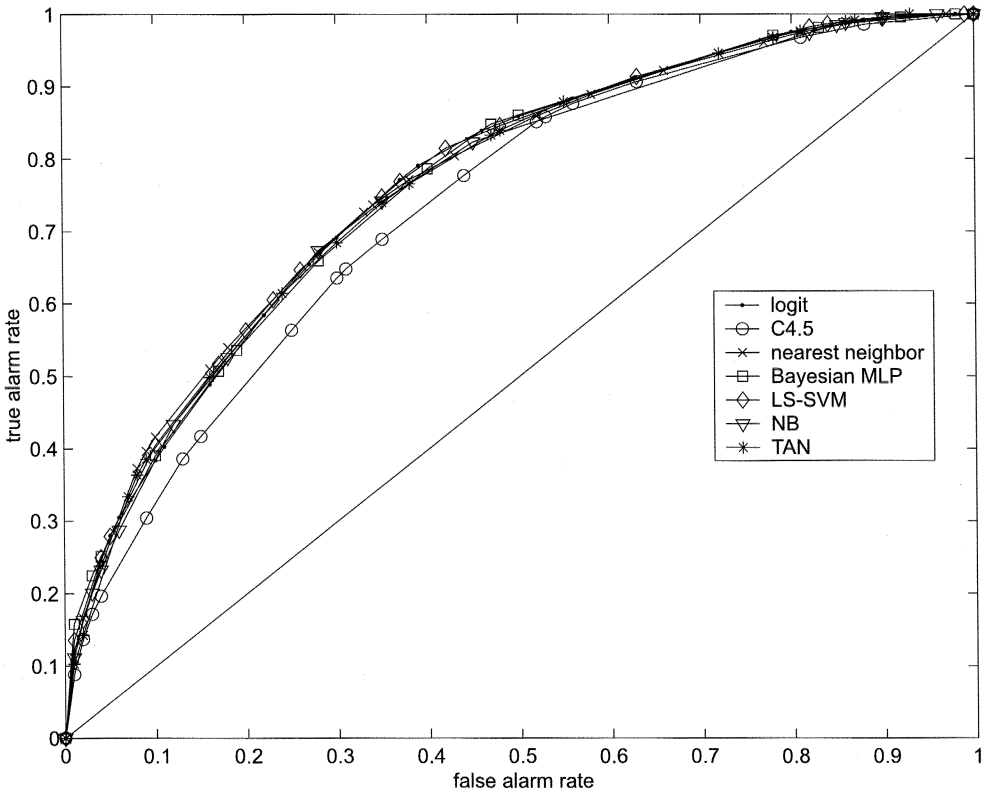


a subtree that contains k or more training data instances. Thus, the estimation neighborhood of a data instance is enlarged until we find a subtree that classifies k or more training data instances. The choice of k is determined during training using ten-fold cross-validation on the training data and choosing $k \in \{5j : j = 1, \ldots, 20\}$ for which the cross-validated mean AUROC performance is highest. The inclusion of $C4.5_u^s$ and $C4.5_c^s$ in our study was influenced by work presented in Provost and Domingos (2000) and Zadrozny and Elkan (2001a, 2001b). Both were included in the study to investigate their effect on the AUROC performance.[17] This is further discussed below. We note that results for m-estimation smoothing and Laplace correction did not differ much. Therefore, we do not report results for Laplace correction.

In terms of comprehensibility, decision trees definitely belong to the most attractive machine learning algorithm types available. This largely explains their popularity. In terms of mean PCC they can often compete with the best, as is illustrated by the results in Table 5. In four out of six relevant evaluation scenarios—specifically, for the 4+ and *vnl* indicator and extended predictor models one C4.5 decision tree operationalization attains performance statistically similar to the highest according to Duncan's multiple range test. However, even though a C4.5 decision tree operationalization is ranked

[17] This is why k was tuned during training by optimizing the AUROC, rather than the PCC.

FIGURE 5
Algorithm Type ROC Curve Convex Hull for *vnl* Indicator Models

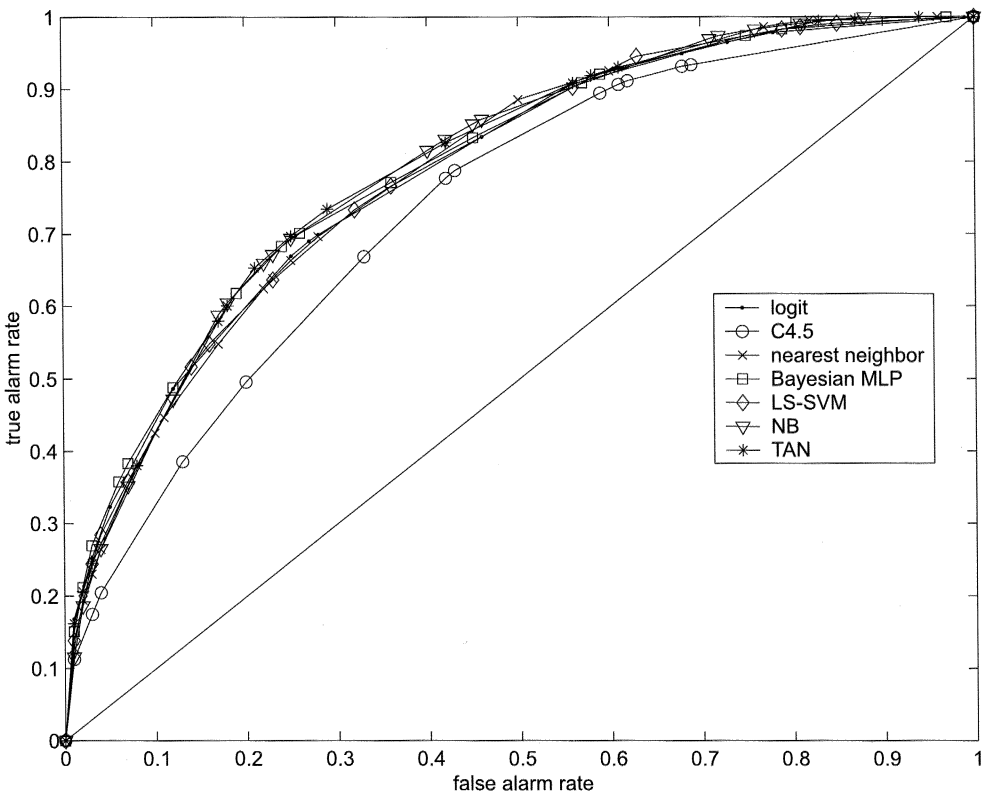


among the best for the 4+ and *vnl* target encoding schemes, the absolute difference between the highest mean PCC and the maximal mean C4.5 decision tree operationalization PCC for either of the four evaluation scenarios still amounts to 1.64 percent for 4+ indicator models, or more, with a maximum of 2.36 percent for *vnl* evaluated on the indicator predictor set. $C4.5_u^s$ is inferior in terms of mean PCC, which could actually have been expected, since disabling pruning seriously increases the risk of overfitting the training data.

The mean AUROC performance reported in Table 6 shows another clear pattern: Not one C4.5 decision tree operationalization in this study is capable of attaining mean AUROC performance comparable to the best, as evidenced by the last column of Table 6. This inferior performance is confirmed by the ROC convex hull plots in Figures 2–11. Visual inspection of the plotted scenarios reveals that the C4.5 algorithm type ROC convex hull is often dominated in large part by most of the other algorithm type hulls.

Provost and Domingos (2000) have commented on the use of decision trees as PETs. Their discussion may shed some light on the observed situation. They argue that, in spite of the fact that the decision tree representation is not intrinsically inadequate for probability estimation, due to a bias of the conventional tree induction algorithms toward maximizing classification accuracy and minimizing tree size, decision trees

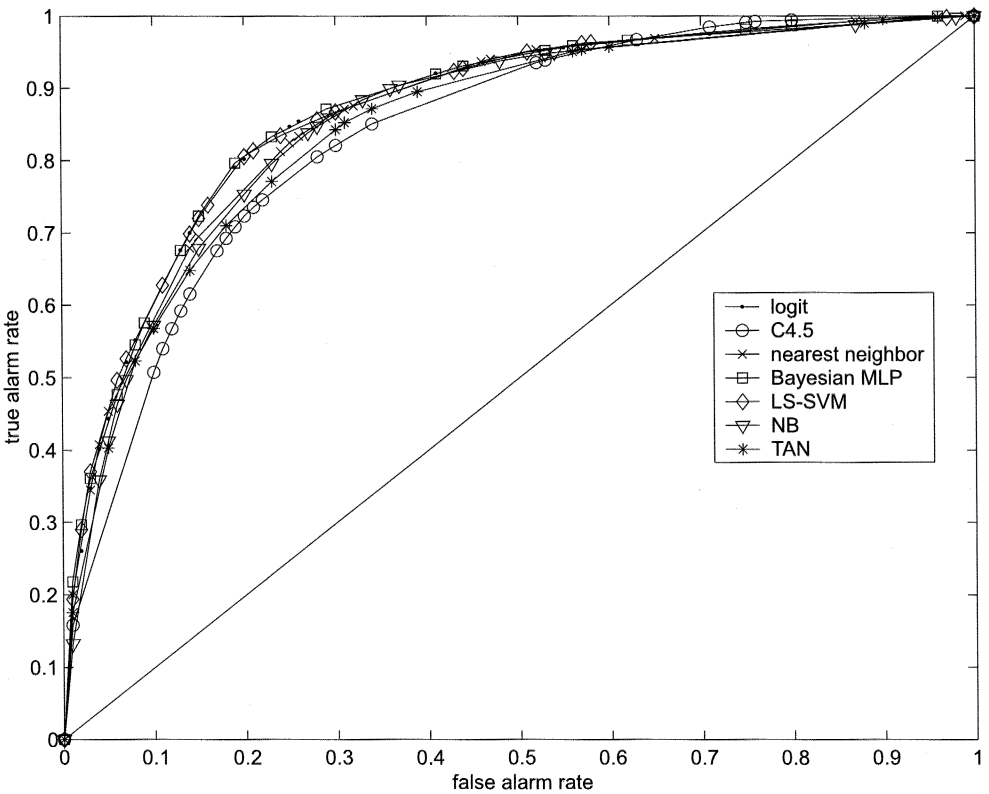
FIGURE 6
Algorithm Type ROC Curve Convex Hull for *vop* Indicator Models



often prove bad probability estimators.[18] Provost and Domingos (2000) provide a thorough overview of previous work and discuss the merits of several potential solutions. Furthermore, they illustrate their discussion by means of AUROC assessment on UCI benchmark data (Blake and Merz, 1998). The results reported by Provost and Domingos (2000), as well as those presented by Zadrozny and Elkan (2001a, 2001b), caused us to consider $C4.5_u^s$ and $C4.5_c^s$ as potential improvements to $C4.5^s$ for the purpose of probability estimation. The discussion and experimental results for benchmark data in Provost and Domingos (2000) and Zadrozny and Elkan (2001a, 2001b) are in line with our results. In the face of limited data availability, the simple operations (or a combination) of smoothing the probability estimates at the leaves, using an unpruned tree or using a curtailed tree, tend to help and are definitely worth considering when using induction trees as PETs. We observe from Table 6 that in nine out of ten evaluation scenarios at least one of $C4.5_u^s$ and $C4.5_c^s$ outperforms $C4.5^s$ in absolute terms. Note that $C4.5_u^s$ and $C4.5_c^s$ were *a priori* hypothesized to provide better AUROC performance. Nevertheless, we also observe that, in order to obtain performance that

[18] In Provost and Domingos (2000), the bad performance of PETs from conventional decision tree learners is attributed to a combination of algorithmic bias and limited data availability. It is left as an issue for further research to investigate the effect on the behavior of those PETs when data availability increases.

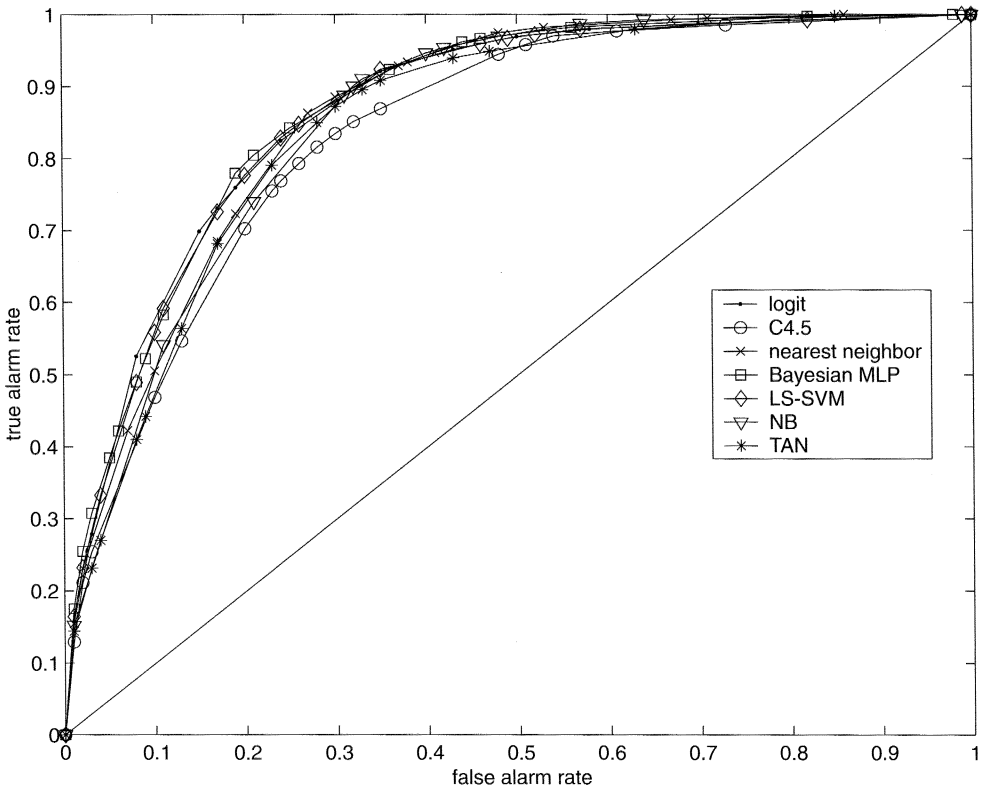
FIGURE 7
Algorithm Type ROC Curve Convex Hull for 1+ Extended Predictor Models



is comparable to the best, more involved algorithmic means will prove to be necessary. As pointed out earlier, for none of the ten evaluation scenarios in Table 6 a C4.5 decision tree operationalization is ranked among the best. Changing the tree pruning strategy with the intention of explicitly preserving the distinctions that are important to probability estimation is considered as an option for improvement by Provost and Domingos (2000). The results they report for this strategy are rather disappointing. However, their experiments using a combination of decision trees by bagging multiple trees (Breiman, 1996) to obtain good probability estimates are most promising (Provost and Domingos, 2000). Note that, on many occasions, the use of an ensemble of decision trees (for example, via bagging) has been observed to significantly improve classification accuracy (Bauer and Kohavi, 1999; Breiman, 1996; Opitz and Maclin, 1999). We leave more thorough investigation as an issue for further research.

Finding the nearest neighbors for a given observation from among a set of training data instances is conceptually straightforward. Nevertheless, as the number of training data instances rises, the computational overhead and storage of the distance matrix may become a burden. Being faced with a rather small data set, this is not much of an issue to us. A definite advantage of this type of classifier is that the class information is not needed until the end of the classifier construction process. The similarity or distance calculation does not depend on it. So, as the classification shifts through

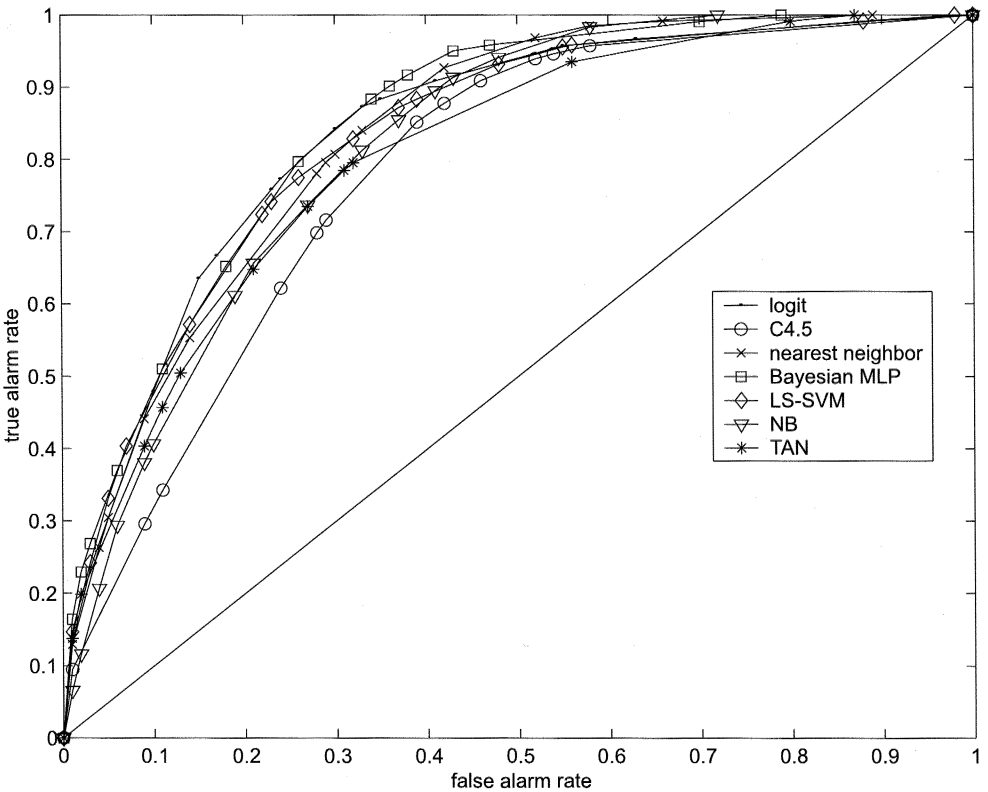
FIGURE 8
Algorithm Type ROC Curve Convex Hull for 4+ Extended Predictor Models



the target encoding schemes presented in the second section, no recomputation of the distance matrix is required. In our implementation we opt for the standard Euclidean distance metric as a measure for the similarity between data instances.[19] We report the performance of 1-nearest neighbor (1NN), 10-nearest neighbor (10NN), 100-nearest neighbor (100NN), and 500-nearest neighbor (500NN) classification. We simply report on these four values of k, but could have determined k using a form of tuning during training (for example, via cross-validation on the training data). In addition to being computationally more intensive, this would have involved a number of expert choices to be made: What range of values are to be tested for k? What kind of optimization criterion is to be used? When is the value of the optimization criterion significantly different to choose one k over the other, taking into account issues of limited data availability and overfitting? These choices obviously depend on the purpose of the study (for example, classification, where k could be tuned for optimal PCC, versus probability estimation, where we could base our tuning procedure on the AUROC). From the results in Tables 5 and 6, we may observe that, as the smoothing factor k is

[19] The choice of distance metric is an important *hot spot* for nearest neighbor classification. It essentially enables domain experts to incorporate prior domain knowledge into the classification process (for example, weighting some predictors more than others on the basis of prior domain expertise).

FIGURE 9
Algorithm Type ROC Curve Convex Hull for 7+ Extended Predictor Models

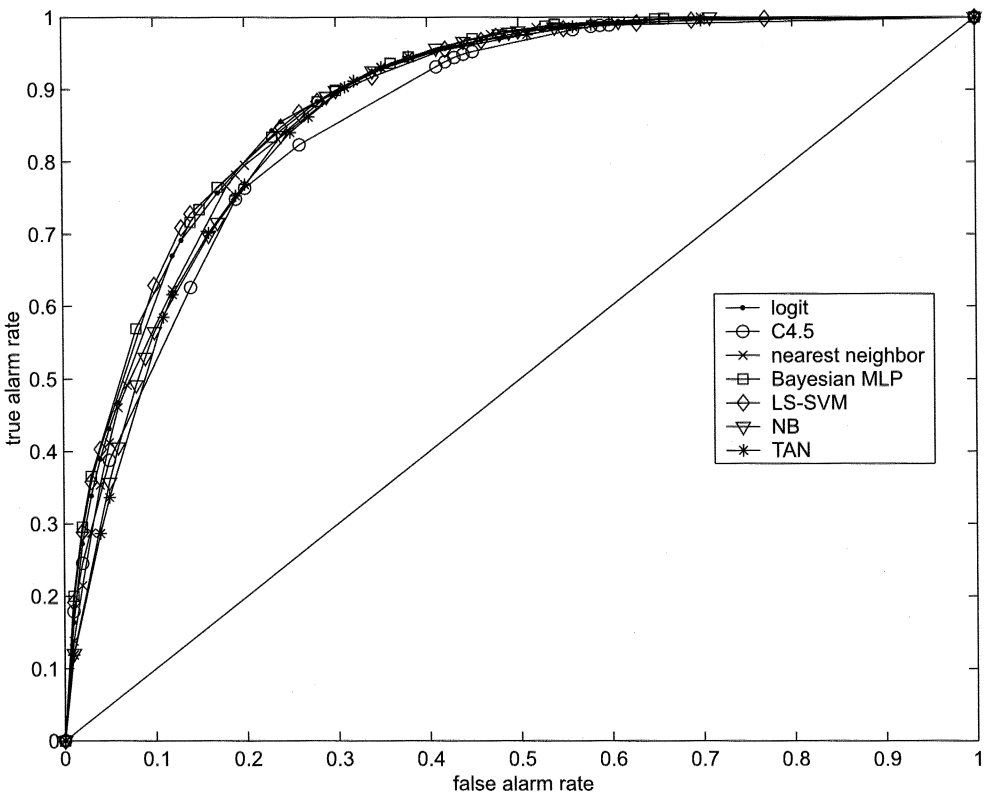


increased from 10 over 100 to 500, for some evaluation scenarios the mean AUROC increases, even though the mean PCC performance clearly degrades. More thorough investigation of the issue is beyond the scope of this study.

What we are interested in here is amply demonstrated by the reported results: In five out of six relevant evaluation scenarios for the mean PCC, and in ten out of ten for the mean AUROC, at least one of the nearest neighbor operationalizations attains performance that is statistically similar to the highest according to Duncan's multiple range test. This definitely attests to the prediction potential of nearest neighbor classification in the context of this study. Again, the plots in Figures 2–11 back this assessment. Note that 1NN is a binary and not a continuous-output classifier. On a set of training data it produces just one nontrivial point in ROC space.

We report the results of Bayesian learning MLPs with one output neuron and one hidden layer with up to three neurons (MLP1, MLP2, MLP3). Adding more hidden layer neurons to the architecture proves unnecessary. We also include the mean AUROC performance of the smoothed versions (MLP1^s, MLP2^s, MLP3^s) in Table 6. Since, by design, the smoothing operation has no effect on the PCC, we omitted the smoothed models from Table 5. The MLPs show excellent overall mean PCC and mean AUROC performance, as can be observed from contrasting the absolute performance figures and from the R and R^g columns in Tables 5 and 6. On very few occasions the smoothed

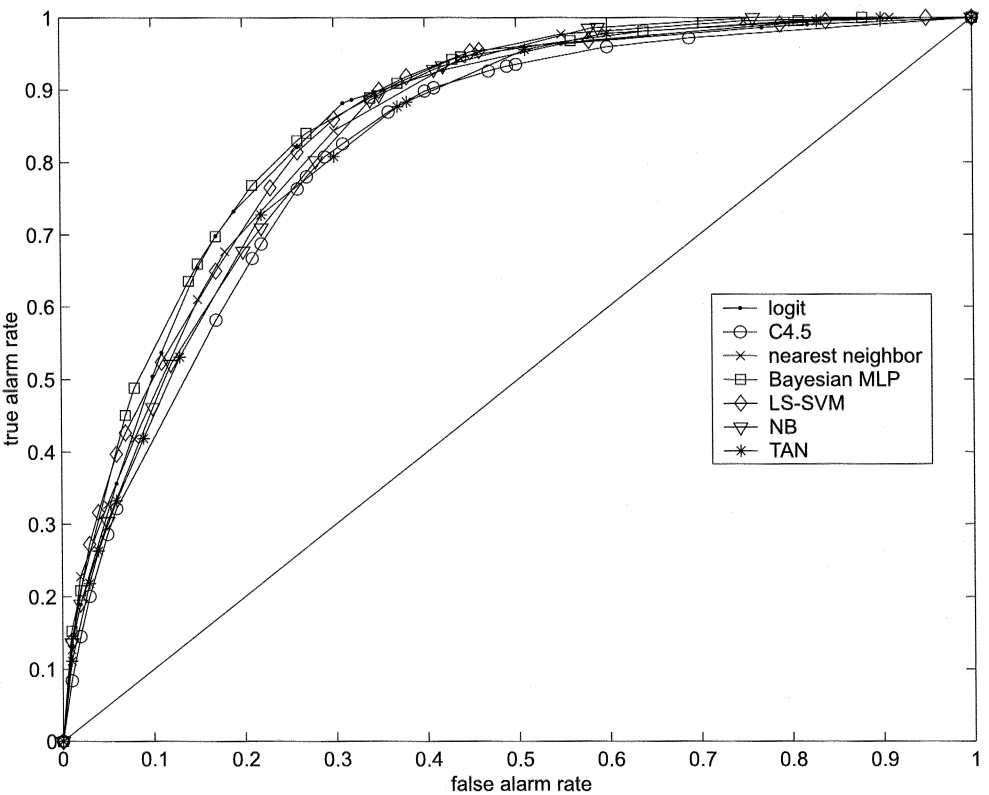
FIGURE 10
Algorithm Type ROC Curve Convex Hull for *vnl* Extended Predictor Models



MLPs fail to beat their unsmoothed counterparts in absolute mean AUROC terms. Nevertheless, the absolute difference in terms of mean AUROC between smoothed and unsmoothed versions remains small. The difference is never significant according to Duncan's multiple range test. Remember that this does not mean that smoothing does not give better-calibrated posterior class membership probability estimates (see the first part of the third section).

The MLPs are among the most computationally intensive algorithms, which makes them less attractive from a practical perspective. Furthermore, their universal modeling power seems to be of little extra value here. For example, under no evaluation scenario in this study is an MLP capable of statistically outperforming the simple linear logit discriminant function according to Duncan's multiple range test. Absolute differences in terms of mean PCC and mean AUROC performance are rather small too. Moreover, the overall results provide a strong argument in favor of a linear discriminant function in the measurement space. This is backed up by the LS-SVM results. The computationally much cheaper linear kernel LS-SVM (Lin-LS-SVMs); again based on constructing a linear discriminant function in the measurement space, are never statistically outperformed by the superior intrinsic modeling power of the computationally more demanding RBF kernel LS-SVMs (RBF-LS-SVMs), which first map the measurement space to a (high-dimensional) feature space using the mapping function

FIGURE 11
Algorithm Type ROC Curve Convex Hull for *vop* Extended Predictor Models



$\varphi(x)$ before constructing bounding hyperplanes (see the fifth part of the fourth section). The polynomial kernel LS-SVMs of degree 2 and 3 (Poly2-LS-SVM and Poly3-LS-SVM) do not attain performance in line with that of their linear and RBF kernel variants. The Lin-LS-SVMs and RBF-LS-SVMs consistently score very high in terms of mean PCC, mean AUROC, and algorithm type ROC convex hull. The reported results for the LS-SVMs on our insurance fraud data can be added to the excellent benchmark results for LS-SVM classification reported in Van Gestel, Suykens, Baesens, Viaene, Vanthienen, Dedene, De Moor, and Vandewalle (2000).

Finally, we report the results of standard (no smoothing) naive Bayes (NB), naive Bayes with Dirichlet smoothing confidence factor 5 (NB^s_5), 25 (NB^s_{25}), and 50 (NB^s_{50}); standard (no smoothing) tree augmented naive Bayes (TAN); TAN Bayes with Dirichlet smoothing confidence factor 5 (TAN^s_5), 25 (TAN^s_{25}), and 50 (TAN^s_{50}). From the results in Tables 5 and 6, we immediately observe that the smoothed naive Bayes algorithms often have their place among the best. In terms of mean PCC the naive Bayes operationalizations fail to measure up to the best for 4+ indicator models. However, the smoothed variants regain their place among the best when using the extended predictor set instead of the indicator predictors only. From the last column in Table 6, we observe that the naive Bayes algorithm type does not once fail to rank among the

best in terms of mean AUROC. Again, the algorithm type ROC curve convex hull plots in Figures 2–11 confirm this assessment.

Our findings for naive Bayes are in line with a long history of other practical studies reporting often surprisingly good performance for naive Bayes, even in cases where the independence assumption underlying it is clearly unrealistic (Domingos and Pazzani, 1997; Duda, Hart, and Stork, 2001; Friedman, Geiger, and Goldszmidt, 1997; Hand, 1992; Webb, 1999). Compared to the TAN Bayes operationalizations, naive Bayes does not require the computationally intensive structure learning phase. From the results in Tables 5 and 6 and considering the ROC convex hull behavior depicted in Figures 2–11, one can definitely question the need for the extra modeling power of the tree augmented models (with smoothing) in most cases. However, the smoothing operation seems to be an overall success for both NB and TAN Bayes. As discussed in the sixth part of the fourth section, smoothing is introduced in light of the limited availability of training data. The negative effect of data deficiency in certain regions of the measurement space is clearly illustrated by the mean PCC and mean AUROC performance of NB and TAN Bayes for all evaluation scenarios covering the extended predictor set. This may be attributed to the fact that many of the nonflag predictors are discretized into a rather large number of discrete values. Some data regions are less densely populated than others. Smoothing satisfactorily counters the adverse effect this has on the performance of naive Bayes and TAN Bayes.

Predictive Power of Nonflags

Independent of the target encoding scheme and the algorithm type, the inclusion of nonflag predictors allows us to significantly boost predictive performance, in terms of both mean PCC (where relevant) and mean AUROC. The increase in predictive performance observed from the inclusion of nonflags into the set of predictors is completely in line with our prior hypothesis that unexploited potential for early screening resides in the use of structural and systematic data mining for fraud detection that goes beyond the use of traditional red flags. Of course, as stated before, the selection of the predictors for this study is limited by what was available in the coded data and is only intended as an initial example of adding nonflag predictors, not an attempt at a complete or efficient model. Nevertheless, the results present a strong argument in favor of using a combination of flags and nonflags for modeling suspicion of fraud.

Summary and Conclusions

The main aim of this research effort was to report on an exploratory study for benchmarking the predictive power of several state-of-the-art binary classification techniques in the context of early screening for suspicion of fraud in PIP automobile insurance claims. The documentation on the problem domain, the data set, the experimental setup, and the algorithm types and operationalizations in the first four sections should enable the reader to assess the robustness and scope of the benchmarking. In an attempt to realize a fair comparison of algorithms, we relied on automatic classifiers as much as possible, using default (widely accepted) hyperparameter choices or tuning hyperparameters using the training data. For most algorithm design or hyperparameter, we reported on several operationalizations using alternative, *a priori* sensible hyperparameter or design choices, documenting the effect of the latter on the cases involved. Results were presented in terms of mean PCC and mean

AUROC curve using a stratified, blocked, ten-fold cross-validation experiment. Algorithm performance differences were assessed using a two-way ANOVA and Duncan's multiple range test (using a 5 percent significance level). We also visualized the performance per algorithm type by constructing the convex hull of the cross-validated model ROCs associated with its alternative operationalizations.

Our results show that: (1) independent of the target encoding scheme and the algorithm type, the inclusion of nonflag predictors significantly boosts predictive performance; and (2) for all the evaluated scenarios, the performance difference in terms of mean PCC and mean AUROC between many algorithm operationalizations turns out to be rather small; visual comparison of the ROC convex hulls for the respective algorithm types also shows limited difference in performance over the range of operating conditions. Noteworthy is the good overall performance of the relatively simple and efficient techniques such as logit and Lin-LS-SVM classification. Naive Bayes (with smoothing) also performs well. More complex and computationally demanding algorithm types such as Bayesian learning MLP, RBF-LS-SVM, and TAN Bayes classification tend to add little or no extra predictive power. Furthermore, the good overall performance of logit and Lin-LS-SVM classificatoin provides a strong argument in favor of a discriminant function that is linear in the input data. Assuming that the case at hand is representative of the domain, does this mean we may catalogue insurance claim fraud detection as being of a "linear," and thus rather simple, nature? Regrettably, we may not. The semantics of the labeled data do not allow us to draw such a conclusion. Remember, the objective of the classification models was to mimic human expert assessment of suspicion of fraud. Here the consensus is that only a small part of existing fraudulent behavior is actually detected by domain experts. The omission error of current fraud detection practice is widely recognized in the literature on insurance (claim) fraud detection (Artís, Ayuso, and Guillén, 2000; Caron and Dionne, 1999; Picard, 1996). The above characterization as linear, however, may well be hypothesized to apply to the nature of the human expert decision-making process with regard to detection of insurance claim fraud and how fraud experts perceive or are able to perceive fraudulent activity.

The C4.5 decision tree operationalization performance turns out to be rather disappointing. Even though a C4.5 operationalization is ranked among the best in terms of mean PCC according to Duncan's multiple range test for most of the evaluated scenarios (such as, for the 4+ operational domain expert choice), an absolute performance gap remains of at least 1.64 percent between the highest mean PCC and the maximal mean C4.5 operationalization PCC for each evaluation scenario. Moreover, none of the C4.5 decision tree operationalizations are capable of attaining mean AUROC performance similar to the best according to Duncan's multiple range test. For all evaluated scenarios, the C4.5 algorithm type ROC convex hull is often dominated in large part by most of the other algorithm type hulls. This demonstrates that the primary or conventional decision tree objective of classification (that is, discrimination between classes) reflected in clear algorithmic bias for growing and pruning the tree, may not be completely in line with scoring objectives or with the objective of producing good probability estimates in the face of limited data availability. Changing the decision tree growing and/or pruning strategy or using an ensemble of decision tree classifiers may be more appropriate in case the primary task is probability estimation or data instance scoring. Moreover, the machine learning literature provides clear evidence that this

latter strategy may substantially improve classification accuracy. Bearing in mind the popularity of decision tree algorithms, these issues definitely deserve further study.

Limited data availability motivated us to consider probability smoothing operations proposed for some of the algorithms to produce better posterior class membership probability estimates to be used as a basis for classification under various operating conditions. The beneficial effect of smoothing for the Bayesian networks is most noticeable. For the Bayesian learning MLPs the performance difference in terms of mean AUROC between smoothed and unsmoothed versions is less pronounced. Here, the mean AUROC is improved in absolute terms, but this never proves to be statistically significant according to Duncan's multiple range test. However, this does not mean calibration of the estimates is not improved. For the C4.5 decision tree operationalizations we estimated posterior class membership probabilities at the tree leaves using m-estimation smoothing.

Now, a final commentary on the nature of the discussion presented in this study is in order. Algorithm choice evidently cannot simply be reduced to a question of predictive performance, although having a clear indication of expected predictive performance of an algorithm given a certain domain and data configuration remains an important facet of a more general evaluation. The issue is much more complex. The choice of algorithm from the ever growing bag of predictive tools and tool sets is as much dictated by issues such as speed of training, tuning, classifying, and the interpretability and actionability of the results. Algorithm choice is equally influenced by tool availability, expert knowledge, in-house expertise, domain characteristics, data (set) characteristics, and other (business) operating conditions. In many a business situation, a white-box or insightful model is chosen *a priori* over a black-box model simply because it better fits the evidence-based analysis and decision framework. From a practical or business perspective, statistically accurate prediction or scoring may well be an asset of a model but definitely not a sufficient one, especially if one is not capable of answering the important *Why?* underlying a model's decisions. This often means trading off the predictive performance of a powerful black-box tool for the simplicity of an understandable and actionable, but less powerful, white-box model. In other situations, a high-parameterized model, potentially requiring lengthy tuning, may be preferred over a more efficient, low-parameterized algorithm because of its capability to be tuned to complex data-, domain-, or organization-specific characteristics. One may then hope to realize a more optimal fusion between the algorithm and the prior domain or expert knowledge. Success of the more involved algorithm construction will of course largely depend on the skill of the people involved in constructing and operating the tools. One may thus ask whether human expert skill is not the foremost critical success factor of any model building or operating activity. In the end, the main contribution to the final performance is the translation from the business problem to the algorithmic context, which is determined by human expert skill (Webb, 1999).

The reader should be well aware of the limited setting of our comparison. In particular, one definitely has a valid point stating that more complex algorithms such as neural networks allow for more modeler manipulation than, for instance, statistical regression-type models. The real power of the former may for that reason be more apparent in a setup where one needs to incorporate valuable domain-specific information in a nonconventional manner during the training phase, the manner that statistical regression-type of tools may not be comfortable with, or when input information

arrives as a process rather than in batch. Neural networks can then provide more flexible ways of accommodating such circumstances. One must also keep in mind that very often low-parameterized methods implicitly make use of more assumptions than really necessary (for example, assumptions of normality or conditional independence of given the class predictors). Moreover, as one of the reviewers of this article noted, the conclusion coupled to the results can be drawn, both sides. Not only do the results illustrate the power of relatively simple algorithms such as logit, but they also show how more complex algorithms like the Bayesian learning MLPs stay in line with the former in this limited setting. Granted! According to the reviewers, this work should thus be complemented by future work to investigate how much may be gained by switching from a simple, yet efficient, exploratory model to a more flexible, complex algorithm if the model is to be set up to make use of domain-specific information in a nonconventional manner. For this study, we chose to minimize that aspect to test the algorithms per se.

Let us, however, help one misconception out of this world: The learning or training step is definitely not the only step in the knowledge discovery in databases process (Fayyad, Piatetsky-Shapiro, Smyth, and Uthurusamy, 1996) in which domain-specific or prior expert information can help steer the modeling effort. Preprocessing of the data can be an equally effective option to incorporate useful domain specifics or prior knowledge. Even a simple logit model can be very effectively manipulated by adding, deleting, or constructing predictors. In addition, for most real-life problems of data mining, domain-specific or prior expert knowledge proves indispensable in the post-processing stage, when one filters the often very limited, really interesting knowledge patterns from the often abundant pile of automatically generated, potentially relevant patterns. This, in turn, may lead data analysts and domain experts to feed the newly discovered chunks of knowledge back into earlier stages of the knowledge discovery process and start an iterative cycle of exploration.

Does this critical note invalidate the objective and results of this research effort? Not in the least. It helps put it into the right perspective. The perspective on algorithm performance that we have taken is definitely partial, but nevertheless valid and worthwhile. For example, the AUROC results in Table 6 would argue against the adoption of any of the C4.5 decision tree algorithms for this application, despite the attractiveness of the triage procedure it produces as a claim screen. Predictive performance benchmarking remains a very useful exploratory exercise, not least in light of the wide variety of available tools and tool sets and the accompanying claims of superior performance. It also enables us to set out directions for further research.

REFERENCES

Agresti, A., 1990, *Categorical Data Analysis*, Wiley Series in Probability and Mathematical Statistics. (New York: Wiley).

Artís, M., M. Ayuso, and M. Guillén, 1999, Modelling Different Types of Automobile Insurance Fraud Behavior in the Spanish Market, *Insurance: Mathematics & Economics*, 24: 67-81.

Artís, M., M. Ayuso, and M. Guillén, 2000, Detection of Automobile Insurance Fraud with Discrete Choice Models and Misclassified Claims, Paper presented at Fourth International Congress on Insurance: Mathematics & Economics, Barcelona, July.

Bauer, E., and R. Kohavi, 1999, An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants, *Machine Learning*, 36: 105-139.

Belhadji, E. B., G. Dionne, and F. Tarkhani, 2000, A Model for the Detection of Insurance Fraud, *The Geneva Papers on Risk & Insurance: Issues and Practice*, 25: 517-539.

Bishop, C. M., 1995, *Neural Networks for Pattern Recognition* (New York: Oxford University Press).

Blake, C. L., and C. J. Merz, 1998, UCI Repository of Machine Learning Databases, University of California, Information and Computer Science, Irvine, Calif. World Wide Web: http://www.ics.uci.edu/~mlearn/MLRepository.html.

Bradley, A., 1997, The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms, *Pattern Recognition*, 30: 1145-1159.

Breiman, L., 1996, Bagging Predictors, *Machine Learning*, 26: 123-140.

Breiman, L., J. Friedman, R. Olshen, and C. Stone, 1984, *Classification and Regression Trees (CART)* (Belmont, Calif.: Wadsworth International Group).

Brockett, P. L., X. Xia, and R. A. Derrig, 1998, Using Kohonen's Self-Organising Feature Map to Uncover Automotive Bodily Injury Claims Fraud, *Journal of Risk & Insurance*, 65: 245-274.

Buntine, W. L., 1992, Learning Classification Trees, *Statistics and Computing*, 2: 63-73.

Caron, L., and G. Dionne, 1999, Insurance Fraud Estimation: More Evidence from the Quebec Automobile Insurance Industry, in: G. Dionne and C. Laberge-Nadeau, eds., *Automobile Insurance: Road Safety, New Drivers, Risks, Insurance Fraud and Regulation*, (Boston, Mass.: Kluwer).

Clarke, M., 1986, The War Insurers Must Win, *Best's Review (P&C edition)*, pp. 51-52, 56, 58 & 60, June.

Clarke, M., 1990, The Control of Insurance Fraud: A Comparative View, *The British Journal of Criminology*, 30: 1-23.

Comité Européen des Assurances, 1996, The European Insurance Anti-Fraud Guide, CEA Info Special Issue 4, May.

Cristianini, N., and J. Shawe-Taylor, 2000, An Introduction to Support Vector Machines, (Cambridge, England: Cambridge University Press).

Cummins, J. D., and S. Tennyson, 1992, Controlling Automobile Insurance Costs, *Journal of Economic Perspectives*, 6: 95-115.

Cummins, J. D., and S. Tennyson, 1996, Moral Hazard in Insurance Claiming: Evidence from Automobile Insurance, *Journal of Risk & Uncertainty*, 12: 26-50.

Cussens, J., 1993, Bayes and Pseudo-Bayes Estimates of Conditional Probabilities and Their Reliability, Paper presented at Sixth European Conference on Machine Learning, Vienna, April.

Derrig, R. A., 1999, Patterns: Fighting Fraud with Data, *Contingencies*, September-October: 40-48.

Derrig, R. A., and K. M. Ostaszewski, 1995, Fuzzy Techniques of Pattern Recognition in Risk and Claim Classification, *Journal of Risk & Insurance*, 62: 447-482.

Derrig, R. A., and H. I. Weisberg, 1998, AIB PIP Screening Experiment Final Report — Understanding and Improving the Claim Investigation Process. AIB Cost Contain-

ment/Fraud Filing DOI Docket R98-41, AIB of Massachusetts. World Wide Web: http://www.ifb.org/ifrr/ifrr267.pdf.

Derrig, R. A., H. I. Weisberg, and X. Chen, 1994, Behavioral Factors and Lotteries Under No-Fault with a Monetary Threshold: A Study of Massachusetts Automobile Claims, *Journal of Risk & Insurance*, 61: 245-275.

Dietterich, T. G., 1998, Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, *Neural Computation*, 10: 1895-1924.

Dionne, G., and E. B. Belhadji, 1996, Evaluation de la Fraude à l'Assurance au Québec, *Assurances*, 64: 365-394.

Dionne, G., A. Gibbens, and P. St.-Michel, 1993, *An Economic Analysis of Insurance Fraud* (Montreal: University of Montreal).

Doherty, N. A., 2000, *Integrated Risk Management: Techniques and Strategies for Reducing Risk* (London: McGraw-Hill).

Domingos, P., 1999, MetaCost: A General Method for Making Classifiers Cost-Sensitive, Paper presented at Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, Calif., August.

Domingos, P., and M. Pazzani, 1997, On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss, *Machine Learning*, 29: 103-130.

Duda, R. O., P. E. Hart, and E. G. Stork, 2001, *Pattern Classification* (New York: Wiley).

Duin, R. P. W., 1996, A Note on Comparing Classifiers, *Pattern Recognition Letters*, 17: 529-536.

Fawcett, T., and F. Provost, 1997, Adaptive Fraud Detection, *Knowledge Discovery and Data Mining*, 1: 291-316.

Fayyad, U. M., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, eds., 1996, Advances in Knowledge Discovery and Data Mining (Menlo Park, Calif.: AAAI Press/MIT Press).

Friedman, J. H., 1995, Introduction to Computational Learning and Statistical Prediction: Tutorial, Paper presented at Twelfth International Conference on Machine Learning, Tahoe City, Calif., July.

Friedman, N., D. Geiger, and M. Goldszmidt, 1997, Bayesian Network Classifiers, *Machine Learning*, 29: 131-163.

Ghezzi, S. G., 1983, A Private Network of Social Control: Insurance Investigation Units, *Social Problems*, 30: 521-530.

Hand, D. J., 1992, Statistical Methods in Diagnosis, *Statistical Methods in Medical Research*, 1: 49-67.

Hand, D. J., 1997, *Construction and Assessment of Classification Rules* (Chichester, England: Wiley).

Hanley, J. A., and B. J. McNeil, 1982, The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve, *Radiology*, 143: 29-36.

Hanley, J. A., and B. J. McNeil, 1983, A Method of Comparing the Areas Under Receiver Operating Characteristic Curves Derived from the Same Cases, *Radiology*, 148: 839-843.

Insurance Bureau of Canada, 1994, *Insurance Fraud in Canada*, Report of the National Task Force on Insurance Fraud, January.

Insurance Research Council, 1996, *Fraud and Buildup in Auto Injury Claims—Pushing the Limits of the Auto Insurance System*, September.

Insurance Research Council, 1997, *Fighting Fraud in the Insurance Industry*, 2nd edition, October.

Kass, G. V., 1980, An Exploratory Technique for Investigating Quantities of Categorical Data, *Applied Statistics*, 29: 119-127.

Knoll, U., G. Nakhaeizadeh, and B. Tausend, 1994, Cost-Sensitive Pruning of Decision Trees, Paper presented at Eighth European Conference on Machine Learning, Catania, Italy, April.

Kohavi, R., 1995, A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, Paper presented at Fourteenth International Joint Conference on AI, Montreal, August.

Lim, T. S., W. Y. Loh, and Y. S. Shih, 2000, A Comparison of Prediction Accuracy, Complexity and Training Time of Thirty-Three Old and New Classification Algorithms, *Machine Learning*, 40: 203-229.

MacKay, D. J. C., 1992a, *Bayesian Methods for Adaptive Models*, Ph.D. thesis, California Institute of Technology, Computation and Neural Systems, Pasadena.

MacKay, D. J. C., 1992b, The Evidence Framework Applied to Classification Networks, *Neural Computation*, 4: 720-736.

Michie, D., D. J. Spiegelhalter, and C. C. Taylor, eds., 1994, *Machine Learning, Neural and Statistical Classification* (New York: Ellis Horwood).

Murphy, K. P., 2001, The Bayes Net Toolbox for Matlab, Paper presented at the Thirty-third Symposium on the Interface of Computing Science and Statistics, Orange County, Calif., June.

Nabney, I. T., 2001, Netlab: Algorithms for Pattern Recognition (New York: Springer).

Neal, R. M., 1998, Assessing Relevance Determination Methods Using Delve, in: C. M. Bishop, ed., *Neural Networks and Machine Learning* (New York: Springer).

Oliver, J. J., and D. J. Hand, 1995, On Pruning and Averaging Decision Trees, Paper presented at Twelfth International Conference on Machine Learning, Tahoe City, Calif., July.

Opitz, D., and R. Maclin, 1999, Popular Ensemble Methods: An Empirical Study, *Journal of Artificial Intelligence Research*, 11: 169-198.

Pearl, J., 1988, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (San Mates, Calif.: Morgan Kaufmann).

Picard, P., 1996, Auditing Claims in Insurance Market with Fraud: The Credibility Issue, *Journal of Public Economics*, 63: 27-56.

Platt, J., 2000, Probabilities for SV Machines, in: A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, eds. *Advances in Large Margin Classifiers* (Cambridge, Mass.: MIT Press).

Provost, F., and P. Domingos, 2000, Well-Trained PETs: Improving Probability Estimation Trees, CDER Working paper 00-04-IS, Stern School of Business, New York University.

Provost, F., and T. Fawcett, 2001, Robust Classification for Imprecise Environments, *Machine Learning*, 42: 203-231.

Provost, F., T. Fawcett, and R. Kohavi, 1998, The Case Against Accuracy Estimation for Comparing Classifiers, Paper presented at Fifteenth International Conference on Machine Learning, Madison, Wisc., July.

Quinlan, J. R., 1993, *C4.5: Programs for Machine Learning* (San Mateo, Calif.: Morgan Kaufmann).

Roberts, S. J., and W. D. Penny, 1999, Bayesian Neural Networks for Classification: How Useful is the Evidence Framework?, *Neural Networks*, 12: 877-892.

Sharma, S., 1996, *Applied Multivariate Techniques* (New York: Wiley).

Simonoff, J., 1998, Three Sides of Smoothing: Categorical Data Smoothing, Nonparametric Regression, and Density Estimation, *International Statistical Review*, 66: 137-156.

Suykens, J. A. K., L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle, 1999, Least-Squares Support Vector Machine Classifiers: A Large-Scale Algorithm, Paper presented at European Conference on Circuits Theory and Design, Stresa, August-September.

Suykens, J. A. K., and J. Vandewalle, 1999, Least-Squares Support Vector Machine Classifiers, *Neural Processing Letters*, 9: 293-300.

Swets, J. A., 1979, ROC Analysis Applied to the Evaluation of Medical Imaging Techniques, *Investigative Radiology*, 14: 109-121.

Swets, J. A., and R. M. Pickett, 1982, *Evaluation of Diagnostic Systems: Methods from Signal Detection Theory* (New York: Academic Press).

Turney, P., 1996, Cost-Sensitive Learning Bibliography. World Wide Web: http://home.ptd.net/~olcay/cost-sensitive.html.

Van Gestel, T., J. A. K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle, 2000, Benchmarking Least-Squares Support Vector Machine Classifiers, Technical Report 00-37, ESAT-SISTA, K. U. Leuven, Belgium. Accepted for Publication in *Machine Learning*.

Vapnik, V., 1995, *The Nature of Statistical Learning Theory* (New York: Springer).

Vapnik, V., 1998, *Statistical Learning Theory* (New York: Wiley).

Verrelst, H., Y. Moreau, J. Vandewalle, and D. Timmerman, 1997, Use of a Multi-Layer Perceptron to Predict Malignancy in Ovarian Tumors, Paper presented at Neural Information Processing Systems Conference, Denver, Colo., May.

Webb, A., 1999, *Statistical Pattern Recognition* (London: Arnold).

Weisberg, H. I., and R. A. Derrig, 1991, Fraud and Automobile Insurance: A Report on the Baseline Study of Bodily Injury Claims in Massachusetts, *Journal of Insurance Regulation*, 9: 497-541.

Weisberg, H. I., and R. A. Derrig, 1995, Identification and Investigation of Suspicious Claims, AIB Cost Containment/Fraud Filing DOI Docket R95-12, Automobile Insurers Bureau of Massachusetts. World Wide Web: http://www.ifb.org/ifrr/ifrr170.pdf.

Weisberg, H. I., and R. A. Derrig, 1998, Méthodes Quantitatives pour la Détection des Demandes D'indemnisation Frauduleuses, *Risques*, 35: 75-101.

Zadrozny, B., and C. Elkan, 2001, Learning and Making Decisions When Costs and Probabilities Are Both Unknown, Paper presented at Seventh ACM SIGKDD Conference on Knowledge Discovery in Data Mining, San Francisco, Calif., July.

Zadrozny, B., and C. Elkan, 2001, Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers, Paper presented at Eighteenth International Conference on Machine Learning, Williams College, Mass., June-July.