Journal of the Operational Research Society

ISSN: 0160-5682 (Print) 1476-9360 (Online) Journal homepage: https://www.tandfonline.com/loi/tjor20

# Benchmarking state-of-the-art classification algorithms for credit scoring

B Baesens, T Van Gestel, S Viaene, M Stepanova, J Suykens & J Vanthienen

**To cite this article:** B Baesens, T Van Gestel, S Viaene, M Stepanova, J Suykens & J Vanthienen (2003) Benchmarking state-of-the-art classification algorithms for credit scoring, Journal of the Operational Research Society, 54:6, 627-635, DOI: 10.1057/palgrave.jors.2601545

**To link to this article:** https://doi.org/10.1057/palgrave.jors.2601545

Journal of the Operational Research Society (2003) 54, 627–635 

www.palgrave-journals.com/jors

# Benchmarking state-of-the-art classification algorithms for credit scoring

B Baesens[1]*, T Van Gestel[2], S Viaene[1], M Stepanova[3], J Suykens[2] and J Vanthienen[1]

[1]*K.U.Leuven, Leuven, Belgium;* [2]*K.U.Leuven, Leuven, Belgium; and* [3]*UBS AG, Financial Services Group, Zurich, Switzerland*

In this paper, we study the performance of various state-of-the-art classification algorithms applied to eight real-life credit scoring data sets. Some of the data sets originate from major Benelux and UK financial institutions. Different types of classifiers are evaluated and compared. Besides the well-known classification algorithms (eg logistic regression, discriminant analysis, *k*-nearest neighbour, neural networks and decision trees), this study also investigates the suitability and performance of some recently proposed, advanced kernel-based classification algorithms such as support vector machines and least-squares support vector machines (LS-SVMs). The performance is assessed using the classification accuracy and the area under the receiver operating characteristic curve. Statistically significant performance differences are identified using the appropriate test statistics. It is found that both the LS-SVM and neural network classifiers yield a very good performance, but also simple classifiers such as logistic regression and linear discriminant analysis perform very well for credit scoring.
*Journal of the Operational Research Society* (2003) **54,** 627–635. doi:10.1057/palgrave.jors.2601545

**Keywords:** credit scoring; classification; benchmarking

## Introduction

The objective of quantitative credit scoring is to develop models that accurately distinguish good applicants (likely to repay) from bad applicants (likely to default). Nowadays, financial institutions see their loan portfolios expand and are actively investigating various alternatives to improve the accuracy of their credit scoring practice. Even an improvement in accuracy of a fraction of a per cent might translate into significant future savings.[1]

Numerous classification techniques have been adopted for credit scoring. These techniques include traditional statistical methods (eg discriminant analysis and logistic regression[2–4]), non-parametric statistical models (eg *k*-nearest neighbour[1,3] and decision trees[5]) and neural networks.[3,5,6] Often conflicts arise when comparing the conclusions of some of these studies. For example, Desai *et al*[6] found that neural networks performed significantly better than linear discriminant analysis for predicting the bad loans, whereas Yobas *et al*[5] reported that the latter outperforms the former. Furthermore, most of these studies only evaluate a limited number of classification techniques on one particular credit scoring data set. Hence, the issue of which classification technique to use for credit scoring remains a very difficult and challenging problem. In this paper, we will conduct a benchmarking study of various classification techniques on eight real-life credit scoring data sets originating, among others, from major Benelux (Belgium, The Netherlands and Luxembourg) and UK financial institutions. Techniques that will be implemented are logistic regression, linear and quadratic discriminant analysis, linear programming (least squares) support vector machines (SVMs), neural networks (tree augmented) naive Bayes and nearest-neighbour classifiers. Especially the power and usefulness of the SVM, least-squares support vector machine (LS-SVM) and tree augmented naive Bayes (TAN) classifiers have not yet been thoroughly investigated in the context of credit scoring. All techniques will be evaluated in terms of the percentage of correctly classified observations and the area under the receiver operating characteristic curve. The latter basically illustrates the behaviour of a classifier without regard to class distribution or misclassification cost. Both performance measures will be compared by using the appropriate test statistics.

The remainder of this paper is organised as follows. We start with a short overview of the classification techniques used. Next, the classification performance criteria are discussed. This is followed by a description of the empirical set-up and the results.

*Correspondence: Dr B. Baesens, Department of Applied Economic Sciences, K.U. Leuven, Naamsestraat 69, Leuven B-3000, Belgium.
E-mail: Bart.Baesens@econ.kuleuven.ac.be

## Classification algorithms for credit scoring

### *Logistic regression, linear and quadratic discriminant analysis*

Given a training set of $N$ data points $D=\{(\mathbf{x}_i,y_i)\}_{i=1}^{N}$, with input data $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels

$y_i \in \{0,1\}$, the logistic regression approach to classification (LOG) tries to estimate the probability $P(y=1|\mathbf{x})$ as follows:

$$P(y=1|\mathbf{x}) = \frac{1}{1 + \exp(-(w_0 + \mathbf{w}^T\mathbf{x}))} \quad (1)$$

whereby $\mathbf{x} \in \mathbb{R}^n$ is an $n$-dimensional input vector, $\mathbf{w}$ is the parameter vector and the scalar $w_0$ is the intercept. The parameters $w_0$ and $\mathbf{w}$ are then typically estimated using the maximum likelihood procedure.

Discriminant analysis assigns an observation $\mathbf{x}$ to the class $y \in \{0,1\}$ having the largest posterior probability $p(y|\mathbf{x})$. It hereby uses Bayes' theorem to compute the posterior probability:[7]

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \quad (2)$$

When one assumes that the class-conditional distributions $p(\mathbf{x}|y)$ are multivariate Gaussian

$$p(\mathbf{x}|y=1)\frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}_1|^{1/2}} \times \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\mathbf{\Sigma}_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)\right\} \quad (3)$$

where $\boldsymbol{\mu}_1$ is the mean vector of class 1 and $\mathbf{\Sigma}_1$ the covariance matrix of class 1, the classification rule becomes: decide $y=1$ if

$$\begin{aligned}(\mathbf{x}-\boldsymbol{\mu}_1)^T\mathbf{\Sigma}_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) - (\mathbf{x}-\boldsymbol{\mu}_0)^T\mathbf{\Sigma}_0^{-1}(\mathbf{x}-\boldsymbol{\mu}_0) \\ < 2(\log(P(y=1)) - \log(P(y=0))) \\ + \log|\mathbf{\Sigma}_0| - \log|\mathbf{\Sigma}_1|\end{aligned} \quad (4)$$

and $y=0$ otherwise. The presence of the quadratic terms $\mathbf{x}^T\mathbf{\Sigma}_1^{-1}\mathbf{x}$ and $-\mathbf{x}^T\mathbf{\Sigma}_0^{-1}\mathbf{x}$ indicates that the decision boundary is quadratic in $\mathbf{x}$ and therefore this classification technique is called *quadratic discriminant analysis* (*QDA*). A simplification occurs if $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_1 = \mathbf{\Sigma}$. In this case, the quadratic terms $\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x}$ and $-\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x}$ cancel and the classification rule becomes linear in $\mathbf{x}$ and the corresponding classification technique is called *linear discriminant analysis* (*LDA*).

### *Linear programming*

Linear programming (LP) is probably one of the most commonly used techniques for credit scoring in the industry nowadays. A very popular formulation goes as follows: [8,9]

$$\min_{\mathbf{w},\boldsymbol{\xi}} \sum_{i=1}^{N} \xi_i \quad (5)$$

subject to

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i \geqslant c - \xi_i & y_i = +1 \\ \mathbf{w}^T\mathbf{x}_i \leqslant c + \xi_i & y_i = -1 \\ \xi_i \geqslant 0, & i = 1, \ldots, N \end{cases} \quad (6)$$

whereby $\boldsymbol{\xi}$ represents the vector of $\xi_i$ values. The first (second) set of inequalities tries to separate the goods (bads) from the bads (goods) by assigning them a score $\mathbf{w}^T\mathbf{x}_i$ that is higher (lower) than the prespecified cutoff $c$ (eg $c=1$). However, since one has to take into account misclassifications, the positive slack variables $\xi_i$ are entered. The aim is then to minimise the misclassifications by minimising the sum of the slack variables $\xi_i$. Note that variants of this method have also been proposed in the literature.[9] For example, Glen[10] suggested a mixed integer programming approach for classification. One of the advantages of using LP methods for credit scoring is that they can easily model domain knowledge or *a priori* bias by including additional constraints.[9]

### *Support vector machines*

Given a training set of $N$ data points $\{(\mathbf{x}_i,y_i)\}_{i=1}^N$, with input data $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{-1,+1\}$, the SVM classifier, according to Vapnik's original formulation, satisfies the following conditions:[11]

$$\begin{cases} \mathbf{w}^T\varphi(\mathbf{x}_i) + b \geqslant +1 & \text{if } y_i = +1 \\ \mathbf{w}^T\varphi(\mathbf{x}_i) + b \leqslant -1 & \text{if } y_i = -1 \end{cases} \quad (7)$$

which is equivalent to

$$y_i[\mathbf{w}^T\varphi(\mathbf{x}_i) + b] \geqslant 1, \qquad i = 1, \ldots, N \quad (8)$$

The non-linear function $\varphi(\cdot)$ maps the input space to a high (possibly infinite)-dimensional feature space. In this feature space, the above inequalities basically construct a hyperplane $\mathbf{w}^T\varphi(\mathbf{x}) + b = 0$ discriminating between both classes. This is visualised in Figure 1 for a typical two-dimensional case.

In primal weight space, the classifier then takes the form

$$y(\mathbf{x}) = \text{sign}[\mathbf{w}^T\varphi(\mathbf{x}) + b] \quad (9)$$

but, on the other hand, is never evaluated in this form. One defines the convex optimisation problem

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{J}(\mathbf{w},b,\boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \quad (10)$$

subject to

$$\begin{cases} y_i[\mathbf{w}^T\varphi(\mathbf{x}_i) + b] \geqslant 1 - \xi_i, & i = 1, \ldots, N \\ \xi_i \geqslant 0, & i = 1, \ldots, N \end{cases} \quad (11)$$

The variables $\xi_i$ are slack variables that are needed in order to allow misclassifications in the set of inequalities (eg because of overlapping distributions). The first part of the objective function tries to maximise the margin between both classes in the feature space, whereas the second part minimises the misclassification error. The positive real constant $C$ should be considered as a tuning parameter in

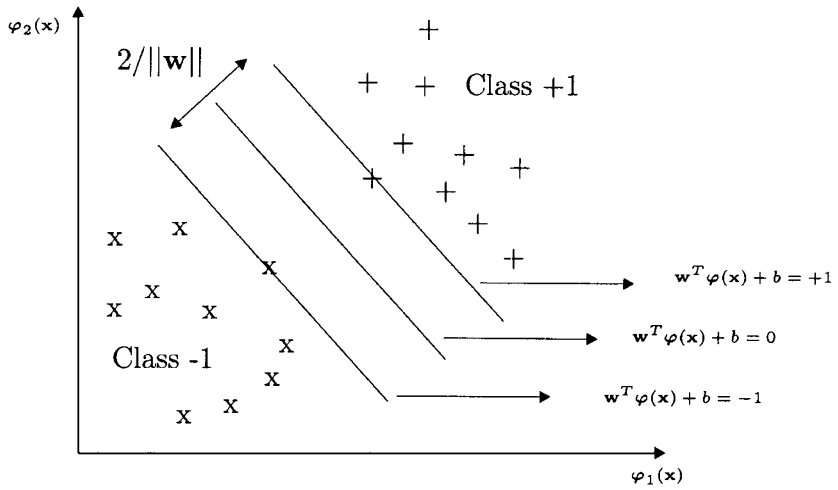


**Figure 1** Illustration of SVM optimisation of the margin in the feature space

the algorithm. Notice that this formulation is closely related to the LP formulation. The major differences are that the SVM classifier introduces a large margin (or regularisation) term $\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$ in the objective function, considers a margin to separate the classes, and allows for non-linear decision boundaries because of the mapping $\varphi(\cdot)$.

The Lagrangian to the constraint optimisation problem (10) and (11) is given by

$$\begin{aligned}\mathscr{L}(\mathbf{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\nu}) =& \mathscr{J}(\mathbf{w}, b, \boldsymbol{\xi}) \\ &- \sum_{i=1}^{N} \alpha_i \{y_i[\mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_i) + b] - 1 + \xi_i\} \\ &- \sum_{i=1}^{N} \nu_i \xi_i\end{aligned} \tag{12}$$

The solution to the above optimisation problem is given by the saddle point of the Lagrangian, that is, by minimising $\mathscr{L}(\mathbf{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\nu})$ with respect to $\mathbf{w}$, $b$, $\boldsymbol{\xi}$ and maximising it with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\nu}$:

$$\max_{\alpha, \nu} \min_{\mathbf{w}, b, \xi} \mathscr{L}(\mathbf{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\nu}) \tag{13}$$

One obtains

$$\begin{cases} \frac{\partial L}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \varphi(\mathbf{x}_i) \\ \frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \\ \frac{\partial L}{\partial \xi_i} = 0 \rightarrow 0 \leqslant \alpha_i \leqslant C, \qquad i = 1, \ldots, N \end{cases} \tag{14}$$

By substituting the first expression into (9), the resulting classifier now becomes

$$y(\mathbf{x}) = \operatorname{sign}\left[\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right] \tag{15}$$

whereby $K(\mathbf{x}_i, \mathbf{x}) = \varphi(\mathbf{x}_i)^{\mathrm{T}}\varphi(\mathbf{x})$ is taken with a positive-definite kernel satisfying the Mercer theorem.

The Lagrange multipliers $\alpha_i$ are then determined by means of the following optimisation problem (dual problem):

$$\max_{\alpha_i} -\frac{1}{2}\sum_{i,j=1}^{N} y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)\alpha_i\alpha_j + \sum_{i=1}^{N} \alpha_i \tag{16}$$

subject to

$$\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leqslant \alpha_i \leqslant C, \qquad i = 1, \ldots, N \end{cases} \tag{17}$$

The entire classifier construction problem now simplifies to a convex quadratic programming (QP) problem in $\alpha_i$. Note that one does not have to calculate $\mathbf{w}$ or $\varphi(\mathbf{x}_i)$ in order to determine the decision surface. Thus, no explicit construction of the non-linear mapping $\varphi(\mathbf{x})$ is needed. Instead, the kernel function $K$ will be used. In this paper, we will use both radial basis function (RBF) kernels $K(\mathbf{x}_i,\mathbf{x}_j) = \exp\{-||\mathbf{x}_i-\mathbf{x}_j||_2^2/\sigma^2\}$ and linear kernels $K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$.

Besides standard SVMs, we will also use LS-SVMs which is a modified version of SVMs suggested by Suykens and Vandewalle[12] and Suykens *et al.*[13] LS-SVMs use a least-squares cost function and replace the inequality constraints by equality constraints as follows:

$$\min_{\mathbf{w}, b, \mathbf{e}} \mathscr{J}(\mathbf{w}, b, \mathbf{e}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + \gamma\frac{1}{2}\sum_{i=1}^{N} e_i^2 \tag{18}$$

subject to the equality constraints

$$y_i[\mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_i) + b] = 1 - e_i, \qquad i = 1, \ldots, N \tag{19}$$

Following the same reasoning as with standard SVMs, it can be shown that the LS-SVM classifier is obtained

as the solution to the following linear system of equations:[12,13]

$$\left[\begin{array}{c|c} 0 & \mathbf{y}^{\mathrm{T}} \\ \hline \mathbf{y} & \Omega + \gamma^{-1}\mathbf{I} \end{array}\right]\left[\begin{array}{c} b \\ \hline \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{1} \end{array}\right] \tag{20}$$

with $\mathbf{y}=[y_1;\ldots;y_N]$, $\mathbf{1}=[1;\ldots;1]$ and $\mathbf{e}=[e_1;\ldots;e_N]$, $\boldsymbol{\alpha}=[\alpha_1;\ldots;\alpha_N]$, whereby Mercer's theorem can be applied within the $\boldsymbol{\Omega}$ matrix, $\boldsymbol{\Omega}_{ij}=y_iy_j\varphi(\mathbf{x}_i)^{\mathrm{T}}\varphi(\mathbf{x}_j)=y_iy_jK(\mathbf{x}_i,\mathbf{x}_j)$. Furthermore, the LS-SVM classifier has also been related to Kernel Fisher Discriminant analysis.[13,14]

### *Neural networks*

Neural networks (NNs) are mathematical representations inspired by the functioning of the human brain. Many types of NNs have been suggested in the literature.[7] We will discuss the multilayer perceptron (MLP) NN in more detail because it is the most popular NN for classification.

An MLP is typically composed of an input layer, one or more hidden layers and an output layer, each consisting of several neurons. Each neuron processes its inputs and generates one output value that is transmitted to the neurons in the subsequent layer. Figure 2 provides an example of an MLP with one hidden layer and one output neuron.

The output of hidden neuron $i$ is then computed by processing the weighted inputs and its bias term $b_i^{(1)}$ as follows:

$$h_i = f^{(1)}\left(b_i^{(1)} + \sum_{j=1}^{n}\mathbf{W}_{ij}x_j\right) \tag{21}$$

$\mathbf{W}$ is the weight matrix whereby $\mathbf{W}_{ij}$ denotes the weight connecting input $j$ to hidden unit $i$. In an analogous way, the output of the output layer is computed as follows:

$$y = f^{(2)}\left(b^{(2)} + \sum_{j=1}^{n_{\mathrm{h}}}\mathbf{v}_jh_j\right) \tag{22}$$

with $n_{\mathrm{h}}$ being the number of hidden neurons and $\mathbf{v}$ the weight vector whereby $\mathbf{v}_j$ represents the weight connecting hidden unit $j$ to the output neuron. The bias inputs play a similar role as the intercept term in a classical linear regression model. A threshold function is then typically applied to map the network output $y$ to a classification label. The transfer functions $f^{(1)}$ and $f^{(2)}$ allow the network to model non-linear relationships in the data. Examples of transfer functions that are commonly used are the sigmoid $f(x)=1/(1+\exp(-x))$, the hyperbolic tangent $f(x)=(\exp(x)-\exp(-x))/(\exp(x)+\exp(-x))$ and the linear transfer function $f(x)=x$. For a binary classification problem, it is convenient to use the logistic transfer function in the output layer ($f^{(2)}$), since its output is limited to a value within the range [0, 1]. This allows the output $y$ of the MLP to be interpreted as a conditional probability.[7]

Note that multiple hidden layers might be used but theoretical works have shown that NNs with one hidden layer are universal approximators capable of approximating any continuous function to any desired degree of accuracy on a compact interval.[7] The weights $\mathbf{W}$ and $\mathbf{v}$ are the crucial parameters of the network and need to be estimated during a learning process. Many algorithms have been suggested in the literature to do this.[7]

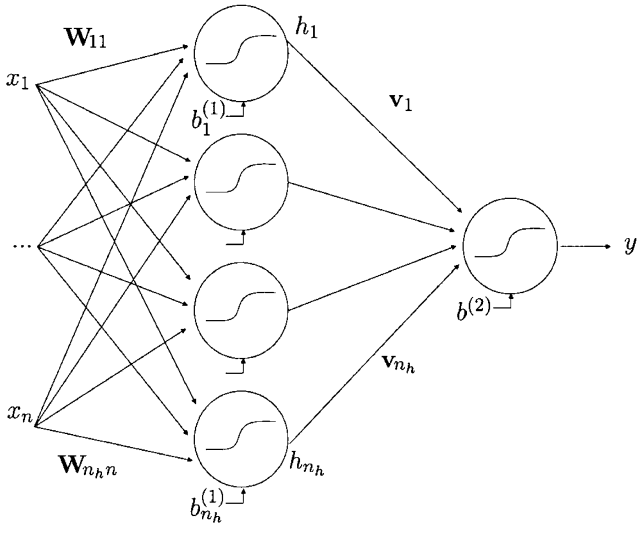


**Figure 2** Architecture of a multilayer perceptron with one hidden layer

### *Bayesian network classifiers*

A simple classifier that in practice often performs surprisingly well, is the naive Bayes classifier.[15] This classifier basically learns the class-conditional probabilities $p(x_i|y)$ of each input $x_i$, $i=1,\ldots,n$ given the class label $y$. A new test case is then classified by using Bayes' rule to compute the posterior probability of each class $y$ given the vector of observed attribute values:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \tag{23}$$

The simplifying assumption behind the naive Bayes classifier then assumes that the attributes are conditionally independent given the class label. Hence,

$$p(\mathbf{x}|y) = \prod_{i=1}^{n}p(x_i|y) \tag{24}$$

The probabilities $p(x_i|y)$ are estimated by using the frequency counts for the discrete attributes and a normal or kernel density based method for the continuous attributes.[15] Friedman *et al*[16] recently introduced tree augmented naive Bayes classifiers (TANs) as an extension of the naive Bayes classifier. TANs dispense the indepen-

dence assumption by allowing tree-structured dependencies between the attributes. A dependency from $x_i$ to $x_j$ then implies that the impact of $x_i$ on the class variable also depends on the value of $x_j$. In this paper, we will also use these TAN classifiers for the discretised credit scoring data sets.

### *Decision trees and rules*

Many decision tree and rule induction algorithms have already been suggested in the literature. One of the most popular is the C4.5 algorithm.[17] C4.5 induces decision trees based on information theoretical concepts. Let $p_1(p_0)$ be the proportion of examples of class 1 (0) in sample $S$. The entropy of $S$ is then calculated as follows:

$$\text{Entropy}(S) = -p_1 \log_2(p_1) - p_0 \log_2(p_0) \qquad (25)$$

whereby $p_0 = 1 - p_1$. Note that the entropy is maximally equal to 1 when $p_1 = p_0 = 0.5$ and minimal 0 when $p_1 = 0$ or $p_0 = 0$. Gain($S$, $x_i$) is defined as the expected reduction in entropy because of sorting (splitting) on attribute $x_i$:

$$\text{Gain}(S, x_i) = \text{Entropy}(S) - \sum_{v \in values(x_i)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \qquad (26)$$

where $S_v$ represents a subsample of $S$ where the attribute $x_i$ has one specific value. However, when the Gain criterion is used to decide upon the node splits, the algorithm favours splits on attributes with many distinct values. Hence, when the data set contains an ID attribute with a distinct value for each customer, the Gain criterion will select it as the best splitting decision. In order to rectify this, C4.5 applies a normalisation and uses the Gainratio criterion which is defined as follows:

$$\text{Gainratio}(S, x_i) = \frac{\text{Gain}(S, x_i)}{\text{SplitInformation}(S, x_i)},$$
$$\text{with SplitInformation}(S, x_i) = -\sum_{k \in values(x_i)} \frac{|S_k|}{|S|} \log_2 \frac{|S_k|}{|S|}, \qquad (27)$$

where $S_k$ represents the subsample of $S$ where the attribute $x_i$ has one specific value. The SplitInformation is the entropy of $S$ with respect to the values of $x_i$. C4.5 then greedily favours splits with the largest gainratio with the additional constraint that the information gain must be at least as large as the average gain over all splits examined. The tree is then constructed by means of recursive partitioning. This tree growing strategy often results in a complex tree with many internal nodes that overfits the data. C4.5 tries to remedy this by a pruning procedure that is executed retrospectively once the full tree has been grown.[17] The unpruned C4.5 tree can be easily converted into a rule set by deriving a rule for each path from the root of the unpruned tree to a leaf node (C4.5rules). These rules may then be further pruned by removing conditions based on a similar procedure as with the tree.

### *k-Nearest neighbour classifiers*

$k$-Nearest-neighbour classifiers (KNN) classify a data instance by considering only the $k$-most similar data instances in the training set.[18] The class label is then assigned according to the class of the majority of the $k$-nearest neighbours. Ties can be avoided by choosing $k$ odd. One commonly opts for the Euclidean distance as the similarity measure:

$$d(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{x}_i - \mathbf{x}_j|| = [(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}(\mathbf{x}_i - \mathbf{x}_j)]^{1/2} \qquad (28)$$

where $\mathbf{x}_i$, $\mathbf{x}_j \in \mathbb{R}^n$ are the input vectors of data instance $i$ and $j$, respectively. Note that also more advanced distance measures have been proposed in the literature.[1]

### *Performance criteria for classification*

The percentage correctly classified (PCC) observations measures the proportion of correctly classified cases on a sample of data. In a number of cases, the PCC may not be the most appropriate performance criterion. It tacitly assumes equal misclassification costs for false-positive and false-negative predictions. This assumption is problematic, since for most real-life problems, one type of classification error may be much more expensive than the other. Another implicit assumption of the use of PCC as an evaluation criterion is that the class distribution (class priors) among examples is presumed constant over time and relatively balanced.[19] Thus, using PCC alone often proves to be inadequate, since class distributions and misclassification costs are rarely uniform. However, taking into account class distributions and misclassification costs proves to be quite hard, since in practice they can rarely be specified precisely and are often subject to change.[20]

If TP, FP, FN and TN represent the number of true positives, false positives, false negatives and true negatives, then the sensitivity measures the proportion of positive examples that are predicted to be positive (TP/(TP + FN)), whereas the specificity measures the proportion of negative examples that are predicted to be negative (TN/(FP + TN)). Note that sensitivity, specificity and PCC vary together as the threshold on a classifier's continuous output is varied between its extremes. The receiver operating characteristic curve (ROC) is a two-dimensional graphical illustration of the sensitivity on the $Y$-axis *versus* (1-specificity) on the $X$-axis for various values of the classification threshold. It basically illustrates the behaviour of a classifier without regard to class distribution or misclassification cost, so it

effectively decouples classification performance from these factors.[21,22]

In order to compare ROC curves of different classifiers, one often calculates the area under the receiver operating characteristic curve (AUC). The AUC then provides a simple figure of merit for the performance of the constructed classifier and is closely related to the Gini coefficient.[9] An intuitive interpretation of the AUC is that it provides an estimate of the probability that a randomly chosen instance of class 1 (positive instance) is correctly rated (or ranked) higher than a randomly selected instance of class 0 (negative instance).

Note that since the area under the diagonal is 0.5, a good classifier should have an AUC much larger than 0.5.

## Empirical set-up and results

### *Data sets and experimental set-up*

Table 1 displays the characteristics of the data sets that will be used to evaluate the performance of the different classification techniques. The Bene1 and Bene2 data sets were obtained from two major financial institutions in the Benelux (Belgium, The Netherlands and Luxembourg). The UK1 (UK3) and UK2 (UK4) data sets are from the same financial institution located in the UK but with other prior class distributions. For the Bene1, Bene2, UK1, UK2, UK3 and UK4 data sets, a bad customer was defined as someone who has missed three consecutive months of payments.[8] The German credit and Australian credit data sets are publicly available at the UCI repository (http://kdd.ics.uci.edu/). Since all data sets have a rather large number of observations, we will split them into $\frac{2}{3}$ training set and $\frac{1}{3}$ test set.

The topic of choosing the appropriate class distribution for classifier learning has received much attention in the literature. In this benchmarking study, we dealt with this problem by using a variety of class distributions ranging from 55.5/44.5 for the Australian credit data set to 90/10 for the UK3 data set, and using the AUC as a performance criterion. The latter is known to be independent with respect to the class distribution and allows us to sidestep the specification of misclassification costs, which are usually very hard to identify for credit scoring.

The LDA, QDA, LOG and LP classifiers require no parameter tuning. For the SVM and LS-SVM classifiers, we used both linear and RBF kernels and adopted a grid search mechanism to tune the hyperparameters.[14] Starting from an initial grid, this mechanism chooses the optimal hyperparameter values as those yielding the best 10-fold cross-validation performance on the training set. Note that also Bayesian training methods have recently been suggested to optimise the LS-SVM hyperparameters.[13] For the SVM and LS-SVM analyses, we used the Matlab$^{TM}$ toolbox of Cawley (http://theoval.sys.uea.ac.uk/~gcc/svm/) and the LS-SVM lab toolbox (http://www.esat.kuleuven.ac.be/sista/lssvmlab/), respectively.

The NN classifiers were trained using the Bayesian evidence framework according to MacKay with the ARD extension.[23,24] The number of hidden neurons was varied between 1 and 5 and the network with the best training set performance was selected for test set evaluation. The NN analyses were conducted using the Netlab toolbox (http://www.ncrg.aston.ac.uk/netlab/). For the naive Bayes classifier, we use the kernel approximation for continuous attributes. We set the confidence level for the pruning strategy of C4.5 and C4.5 rules to 0.25, which is the default value. Furthermore, we will conduct the C4.5 (rules) analyses using both the original continuous data and the discretised versions. The discretisation process will be carried out using the discretisation algorithm of Fayyad and Irani.[25] The discretised data will also be used to train and evaluate the TAN classifiers. We use McNemar's test[26] to compare the PCCs and the test of De Long *et al*[27] to compare the AUCs of the different classifiers.

## Results

Table 2 reports the test set PCC, sensitivity and specificity of all classifiers on the eight credit scoring data sets. We have used a special notational convention whereby the best test set PCC performance per data set is underlined and denoted in bold face, performances that are not significantly different at the 5% level from the top performance with respect to a one-tailed test are tabulated in bold face, and statistically significant underperformances at the 1% level are empha-

**Table 1** Characteristics of credit scoring data sets

| | *Inputs* | *Data set size* | *Training set size* | *Test set size* | *Goods/bads* |
|---|---|---|---|---|---|
| Bene1 | 33 | 3123 | 2082 | 1041 | 66.7/33.3 |
| Bene2 | 33 | 7190 | 4793 | 2397 | 70/30 |
| UK1 | 16 | 9360 | 6240 | 3120 | 75/25 |
| UK2 | 16 | 11700 | 7800 | 3900 | 80/20 |
| UK3 | 19 | 3960 | 2640 | 1320 | 90/10 |
| UK4 | 19 | 1980 | 1320 | 660 | 80/20 |
| Germ | 20 | 1000 | 666 | 334 | 70/30 |
| Austr | 14 | 690 | 460 | 230 | 55.5/44.5 |

**Table 2** Test set classification accuracy on credit scoring data sets assuming a cutoff of 0.5

| *Technique* | *Bene1* | | | *Bene2* | | | *Germ* | | | *Austr* | | | *UK1* | | | *UK2* | | | *UK3* | | | *UK4* | | | *AR* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* |
| LDA | **72.2** | 82.3 | 53.2 | 73.7 | 87.7 | 40.9 | **74.6** | 90.0 | 41.0 | **88.3** | 87.4 | 89.3 | 74.4 | 98.3 | 3.56 | 80.2 | 98.0 | 6.83 | *88.6* | 98.6 | 3.60 | **82.0** | 96.1 | 21.0 | 6.94 |
| QDA | *56.9* | 40.9 | 87.2 | *41.2* | 17.3 | 96.9 | **71.0** | 79.5 | 52.4 | *83.0* | 92.9 | 70.9 | *71.5* | 89.4 | 18.4 | *77.8* | 91.4 | 22.1 | *84.6* | 93.0 | 13.7 | 77.7 | 89.9 | 25.0 | *15.8* |
| LOG | **72.0** | 82.3 | 52.6 | **74.4** | 87.8 | 43.1 | **74.6** | 89.5 | 41.9 | 87.4 | 91.3 | 82.5 | *74.4* | 98.4 | 2.93 | 80.3 | 98.5 | 5.12 | **89.2** | 99.5 | 1.44 | **82.1** | 96.3 | 21.0 | **6.06** |
| LP | 71.2 | 79.8 | 54.9 | **74.2** | 88.4 | 40.9 | **71.9** | 92.6 | 26.7 | **88.3** | 86.6 | 90.3 | **74.8** | 100 | 0.00 | **80.5** | 100 | 0.00 | **89.5** | 100 | 0.00 | **81.2** | 100 | 0.00 | **6.50** |
| RBF LS-SVM | **73.1** | 83.9 | 52.6 | **74.3** | 88.4 | 41.3 | **74.3** | 96.5 | 25.7 | 89.1 | 89.8 | 88.3 | **74.8** | 100 | 0.00 | **80.7** | 99.7 | 2.23 | **89.0** | 99.1 | 3.60 | **82.1** | 97.4 | 16.1 | **3.56** |
| Lin LS-SVM | 71.4 | 84.3 | 46.8 | 73.7 | 89.5 | 36.9 | **73.7** | 91.3 | 35.2 | **88.3** | 87.4 | 89.3 | **74.6** | 99.2 | 1.65 | 80.4 | 99.5 | 1.58 | **89.5** | 100 | 0.00 | **81.8** | 97.9 | 12.1 | **6.75** |
| RBF SVM | 71.9 | 82.1 | 52.4 | 73.9 | 89.2 | 38.2 | **74.0** | 92.6 | 33.3 | 87.0 | 89.8 | 83.5 | **74.8** | 99.9 | 0.25 | 80.2 | 99.1 | 2.37 | **89.2** | 99.3 | 2.88 | **81.2** | 98.9 | 4.84 | 8.19 |
| Lin SVM | 71.2 | 82.1 | 50.4 | 73.9 | 88.4 | 40.2 | 71.0 | 95.6 | 17.1 | **88.3** | 86.6 | 90.3 | **74.8** | 100 | 0.00 | **80.5** | 100 | 0.00 | **89.5** | 100 | 0.00 | **81.2** | 100 | 0.00 | 6.94 |
| NN | **72.4** | 81.8 | 54.6 | **75.1** | 86.7 | 48.1 | **73.7** | 85.2 | 48.6 | **88.3** | 92.1 | 83.5 | **75.0** | 99.2 | 3.05 | **80.6** | 99.9 | 0.92 | **89.4** | 99.7 | 2.16 | **80.9** | 93.3 | 27.4 | **5.19** |
| NB | *65.8* | 55.1 | 86.1 | *59.0* | 51.1 | 77.5 | **72.2** | 87.8 | 38.1 | *82.2* | 100 | 60.2 | *70.8* | 87.2 | 22.1 | *77.8* | 90.6 | 25.4 | *88.1* | 97.1 | 11.5 | *77.9* | 88.4 | 32.3 | *15.1* |
| TAN | *69.5* | 78.2 | 53.2 | 73.4 | 82.7 | 51.7 | **72.5** | 87.3 | 40.0 | 87.4 | 92.9 | 80.6 | *72.9* | 93.9 | 10.6 | *77.4* | 90.6 | 22.9 | *87.1* | 96.2 | 10.1 | *76.8* | 87.9 | 29.0 | *12.9* |
| C4.5 | *68.9* | 77.4 | 52.6 | *69.8* | 81.3 | 43.0 | **72.2** | 88.2 | 37.1 | *84.3* | 93.7 | 72.8 | *71.1* | 89.3 | 16.8 | *79.3* | 95.0 | 14.3 | **89.5** | 100 | 0.00 | **81.2** | 100 | 0.00 | 12.1 |
| C4.5rules | **71.5** | 80.4 | 54.6 | *71.4* | 76.8 | 58.8 | 71.0 | 91.3 | 26.7 | *85.2* | 90.6 | 78.6 | *67.1* | 77.4 | 36.6 | 80.4 | 99.9 | 0.13 | *88.5* | 98.6 | 2.16 | **81.7** | 97.0 | 12.9 | *11.3* |
| C4.5dis | *69.3* | 78.6 | 51.5 | 73.1 | 85.3 | 44.6 | **74.6** | 87.3 | 46.7 | **89.1** | 94.5 | 82.5 | **74.8** | 100 | 0.00 | **80.5** | 100 | 0.00 | **89.5** | 100 | 0.00 | **81.2** | 100 | 0.00 | **6.69** |
| C4.5rulcs dis | 70.2 | 80.6 | 50.4 | **73.9** | 84.8 | 48.5 | **74.3** | 89.5 | 41.0 | **90.4** | 94.5 | 85.4 | *72.7* | 94.6 | 7.76 | *79.8* | 98.1 | 4.47 | 88.6 | 98.2 | 6.47 | **80.8** | 97.6 | 8.06 | 9.19 |
| KNN10 | *66.7* | 83.9 | 34.0 | *70.8* | 92.2 | 21.0 | **70.7** | 94.8 | 18.1 | 86.5 | 93.7 | 77.7 | *73.2* | 96.1 | 5.22 | *79.8* | 98.1 | 4.20 | **89.5** | 100 | 0.00 | **81.2** | 97.9 | 8.87 | 11.9 |
| KNN100 | 70.3 | 83.6 | 45.1 | *72.0* | 96.2 | 15.3 | *68.6* | 100 | 0.00 | **88.7** | 95.3 | 80.6 | **74.8** | 100 | 0 13 | **80.5** | 100 | 0.00 | **89.5** | 100 | 0.00 | **81.2** | 100 | 0.00 | **7.94** |

**Table 3** Test set classification accuracy on credit scoring data sets assuming a marginal good–bad rate around 5:1

| *Technique* | *Bene1* | | | *Bene2* | | | *Germ* | | | *Austr* | | | *UK1* | | | *UK2* | | | *UK3* | | | *UK4* | | | *AR* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* | *Sens* | *Spec* | *PCC* |
| LDA | *62.4* | 48.4 | 89.1 | *62.2* | 51.4 | 87.3 | *58.7* | 42.8 | 93.3 | *86.5* | 83.5 | 90.3 | *50.7* | 41.3 | 78.6 | *78.5* | 94.2 | 14.1 | *85.3* | 91.8 | 30.2 | *68.0* | 68.8 | 64.5 | *9.44* |
| QDA | *56.3* | 39.7 | 87.7 | *40.9* | 16.9 | 96.9 | 71.0 | 79.5 | 52.4 | *86.1* | 90.6 | 80.6 | *49.2* | 38.1 | 82.3 | *64.3* | 61.8 | 74.5 | *83.9* | 91.8 | 16.5 | *77.1* | 88.1 | 29.8 | *10.8* |
| LOG | *58.3* | 40.9 | 91.4 | *62.2* | 51.5 | 87.3 | *65.0* | 55.9 | 84.8 | *84.8* | 80.3 | 90.3 | *50.0* | 40.1 | 79.4 | *66.3* | 65.3 | 70.0 | **88.9** | 98.6 | 5.76 | *74.1* | 77.8 | 58.1 | *9.25* |
| LP | *59.1* | 42.1 | 91.4 | *58.3* | 43.8 | 92.1 | *64.7* | 56.3 | 82.9 | 88.3 | 86.6 | 90.3 | **74.8** | 100 | 0.00 | **80.5** | 100 | 0.00 | **89.5** | 100 | 0.00 | **81.2** | 100 | 0.00 | **5.06** |
| RBF LS-SVM | **67.1** | 58.1 | 84.4 | 67.4 | 62.8 | 78.3 | **75.1** | 87.3 | 48.6 | *47.8* | 6.30 | 99.0 | *73.1* | 93.6 | 12.3 | *74.1* | 81.6 | 43.2 | *87.0* | 95.9 | 12.2 | *76.1* | 82.1 | 50.0 | **5.88** |
| Lin LS-SVM | 65.3 | 54.7 | 85.5 | *65.8* | 60.0 | 79.4 | *58.7* | 42.8 | 93.3 | *87.4* | 85.0 | 90.3 | *43.9* | 29.5 | 86.6 | *75.3* | 85.2 | 34.4 | *86.7* | 94.6 | 20.1 | *76.4* | 82.8 | 48.4 | **8.19** |
| RBF SVM | *62.0* | 47.5 | 89.4 | *62. 2* | 51.4 | 87.3 | *63.5* | 54.6 | 82.9 | *85.2* | 81.9 | 89.3 | *26.1* | 1.41 | 99.2 | *74.6* | 88.2 | 18.8 | *86.9* | 95.4 | 14.4 | *77.1* | 87.1 | 33.9 | *8.94* |
| Lin SVM | *61.2* | 46.9 | 88.3 | *62.9* | 53.2 | 85.5 | *66.8* | 58.5 | 84.8 | 88.3 | 86.6 | 90.3 | **74.8** | 100 | 0.00 | **80.5** | 100 | 0.00 | **89.5** | 100 | 0.00 | **81.2** | 100 | 0.00 | **3.94** |
| NN | **67.1** | 58.1 | 84.1 | *64.1* | 53.8 | 88.2 | **70.4** | 68.6 | 74.3 | *81.3* | 74.8 | 89.3 | *49.3* | 36.8 | 86.3 | *61.1* | 55.0 | 86.2 | *80.0* | 84.5 | 41.7 | *61.5* | 59.3 | 71.0 | **10.3** |
| NB | *57.7* | 40.2 | 91.1 | *57.0* | 47.5 | 79.1 | *62.6* | 52.4 | 84.8 | 87.4 | 93.7 | 79.6 | *52.7* | 43.9 | 78.8 | *66.9* | 65.8 | 71.4 | *84.1* | 90.9 | 25.9 | *76.7* | 85.6 | 37.9 | *10.4* |
| TAN | *63.7* | 50.6 | 88.6 | *64.0* | 54.6 | 85.8 | *63.2* | 51.5 | 88.6 | *81.3* | 74.0 | 90.3 | *45.4* | 29.2 | 93.3 | *61.2* | 55.5 | 85.0 | *86.7* | 95.3 | 13.7 | *75.3* | 83.8 | 38.7 | 9.94 |
| C4.5 | *59. 7* | 46.2 | 85.2 | *31.1* | 1.91 | 99.3 | 68.9 | 68.1 | 70.5 | 88.7 | 88.2 | 89.3 | *61.4* | 66.0 | 47.8 | *71.3* | 75.5 | 53.7 | **89.4** | 99.9 | 0.00 | **81.1** | 99.8 | 0.00 | 7.06 |
| C4.5rules | *50.5* | 28.6 | 92.2 | **69.7** | 70.5 | 67.7 | *45.5* | 27.5 | 84.8 | *46.5* | 9.45 | 92.2 | *53.3* | 45.5 | 76.3 | *67.9* | 67.7 | 68.9 | *88.5* | 98.6 | 2.16 | *60.0* | 58.2 | 67.7 | 11.4 |
| C4.5dis | *53.2* | 33.0 | 91.6 | *61.9* | 54.1 | 80.1 | *64.1* | 57.6 | 78.1 | 89.1 | 94.5 | 82.5 | **74.8** | 100 | 0.00 | **80.5** | 100 | 0.00 | **89.5** | 100 | 0.72 | **81.1** | 98.8 | 0.00 | **5.81** |
| C4.5rules dis | *52.7* | 31.1 | 93.9 | *50.5* | 34.3 | 88.3 | **74.3** | 89.5 | 41.0 | **91.7** | 94.5 | 88.3 | *72.7* | 94.6 | 7.76 | *57.7* | 50.6 | 87.3 | *84.9* | 92.4 | 21.6 | *73.5* | 81.0 | 41.1 | 10.0 |
| KNN10 | *62.6* | 51.6 | 836 | *53.5* | 39.7 | 85.8 | *41.3* | 17.0 | 94.3 | *81.3* | 74.0 | 90.3 | *66.2* | 77.5 | 32.4 | *55.2* | 50.8 | 73.1 | *85.1* | 93.6 | 12.9 | *67.9* | 71.6 | 51.6 | *12.3* |
| KNN100 | *56.2* | 37.2 | 92.2 | *57.1* | 45.2 | 85.0 | *62.0* | 52.0 | 83.8 | *79.6* | 69.3 | 92.2 | *44.5* | 29.3 | 89.6 | *58.3* | 54.0 | 46.3 | *83.2* | 89.8 | 26.6 | *63.5* | 62.5 | 67.7 | *14.4* |

**Table 4** Test set AUC on credit scoring data sets

| *Technique* | *Bene1* | *Bene2* | *Germ* | *Austr* | *UK1* | *UK2* | *UK3* | *UK4* | *AR* |
|---|---|---|---|---|---|---|---|---|---|
| LDA | **77.1** | *77.1* | **78.4** | **92.8** | *64.1* | *73.6* | **74.4** | **72.3** | **5.38** |
| QDA | *73.4* | *72.4* | *71.8* | **91.5** | *63.3* | *72.1* | *68.1* | *68.3* | *10.8* |
| LOG | **77.0** | *78.0* | **77.7** | **93.2** | *63.9* | *73.0* | **74.6** | **72.7** | **4.38** |
| LP | *76.1* | *77.5* | **76.3** | **92.6** | *56.4* | *62.3* | *62.0* | *62.2* | *11.9* |
| RBF LS-SVM | **77.6** | 77.8 | **77.4** | **93.2** | 65.0 | 74.7 | **72.9** | **73.1** | **3.38** |
| Lin LS-SVM | **76.9** | *77.1* | **78.4** | **92.9** | *64.4* | *73.7* | **73.8** | **72.5** | **5.50** |
| RBF SVM | *76.7* | *77.1* | **77.2** | **92.6** | *59.3* | *65.4* | *67.3* | *63.4* | 9.13 |
| Lin SVM | *75.9* | *77.5* | **76.6** | **93.6** | *56.4* | *63.9* | *62.9* | *62.9* | 10.1 |
| NN | **76.9** | **79.1** | **78.7** | **91.7** | **66.4** | **75.8** | **74.6** | **72.9** | **3.25** |
| NB | **76.5** | *70.6* | **77.2** | **93.1** | **65.8** | *73.7* | *66.9* | 67.9 | 7.88 |
| TAN | *75.5* | **78.2** | **78.3** | **93.4** | **66.8** | 74.5 | *64.0* | *66.6* | **5.63** |
| C4.5 | *72.2* | *71.1* | **74.7** | 91.6 | *56.1* | *65.7* | *50.0* | *49.9* | *14.7* |
| C4.5rules | *71.6* | *74.2* | *62.0* | *85.3* | *61.7* | *70.4* | *60.3* | 68.4 | *13.0* |
| C4.5dis | *73.0* | *73.2* | **74.6** | **93.1** | *50.0* | *50.0* | *50.4* | *49.9* | *13.7* |
| C4.5rules dis | *73.0* | *71.5* | *64.4* | 93.1 | 65.2 | *71.5* | *66.7* | *64.9* | 10.8 |
| KNN10 | *71.7* | *69.6* | *70.2* | 91.4 | *58.9* | *65.4* | *63.0* | 67.0 | *14.1* |
| KNN100 | *74.9* | *71.5* | **76.1** | **93.0** | *62.8* | *69.9* | *70.0* | **70.4** | 9.5 |

sised. Performances significantly different at the 5% level but not at the 1% level are reported in normal script. We also compute a ranking of the different algorithms on each data set assigning rank 1 to the algorithm yielding the best PCC and rank 17 to the algorithm giving the worst PCC. These ranks are then averaged and compared using a Wilcoxon signed rank test of equality of medians (see last column of Table 2).

It can be seen from Table 2 that the RBF LS-SVM classifier yielded the best average ranking (AR) for the PCC measure. The LOG, LP, Lin LS-SVM, NN, C4.5 dis and KNN100 classifiers have statistically the same AR for the PCC as the RBF LS-SVM classifier at the 5% significance level. The PCC average rankings of the QDA, NB, TAN and C4.5rules classifiers are statistically inferior to the PCC AR of the RBF LS-SVM classifier at the 1% level. It can also be observed that both the C4.5 and C4.5rules classifiers achieve better average PCC rankings on the discretised data than on the continuous data.

Note that the performance measures reported in Table 2 were calculated assuming a cutoff value of 0.5 on the classifier's output. This may, however, not be the most appropriate threshold to use for the more skewed UK data sets. For these data sets, some classifiers have a tendency to always predict the majority class (good customer) yielding hereby 100% sensitivity and 0% specificity. Especially the KNN100 classifier tends to predict the majority class for many of the data sets considered. Hence, it might also be interesting to look at other cutoff values. Many alternative cutoff setting schemes have been proposed in the literature. Table 3 reports the results with cutoffs based on a marginal good–bad rate around 5:1.[9,28] Using this scheme, it can be observed that the Lin SVM classifier gave the best PCC AR, with that of the LP, RBF LS-SVM, Lin LS-SVM, NN, and C4.5dis classifiers not being statistically different from it at the 5% level. There is some similarity with the average PCC rankings reported in Table 2.

Table 4 reports the AUCs of all classifiers on the eight credit scoring data sets. This table has the same set-up as Table 2. It clearly indicates that the RBF LS-SVM classifier was two times ranked first, the NN classifier four times, and the Lin SVM and TAN classifiers each one time. It can be observed that the best average rank was attributed to the NN classifier. The AR of the LDA, LOG, RBF LS-SVM, Lin LS-SVM and TAN classifiers are not statistically inferior at the 5% significance level. The AR of the QDA, LP, C4.5, C4.5rules, C4.5dis, and KNN10 classifiers are statistically worse than the AR of the NN classifier at the 1% level. Note that for the Bene2, UK1 and UK2 data sets, the NN classifier always performed significantly better than the LOG and LDA classifiers in terms of AUC. Although the difference may be small in absolute terms, it has to be noted that in a credit scoring context, an increase in the discrimination ability of even a fraction of a percent may translate into significant future savings.[1]

In summary, when considering both the PCC and AUC performance measures, it can be concluded that the non–linear RBF LS-SVM and NN classifiers yield a very good performance. However, the more simple, linear classification techniques such as LDA and LOG also have a very good performance, which is in the majority of the cases not statistically different from that of the RBF LS-SVM and NN classifiers. This finding clearly indicates that most credit scoring data sets are only weakly non-linear. The majority of classification techniques yielded classification performances that are quite competitive with each other. Only a small number of classifiers (eg QDA, NB and C4.5rules) yielded weak performances in terms of both PCC and AUC, whereas others (eg LP and C4.5dis) gave a good PCC performance but a rather bad AUC.

## Conclusions

In this paper, we studied the performance of various classification techniques for credit scoring. The experiments were conducted on eight real-life credit scoring data sets. The classification performance was assessed by the percentage correctly classified cases (PCC) and the area under the receiver operating characteristic curve (AUC). Statistically significant differences were detected by using the appropriate test statistics for both criteria.

It was found that the RBF LS-SVM and NN classifiers yield a very good performance in terms of both PCC and AUC. However, it has to be noted that simple, linear classifiers such as LDA and LOG also gave very good performances, which clearly indicates that most credit scoring data sets are only weakly non-linear. The experiments also indicated that many classification techniques yield performances which are quite competitive with each other. Only a few classification techniques were clearly inferior to the others.

Starting from the findings of this study, several interesting topics for future research can be identified. One interesting topic would be to repeat the same experiment but consider the dependent variable as continuous, for example, denoting the number of months in arrears of a customer. Another promising avenue for future research is to investigate the power of classifier ensembles where multiple classification algorithms are combined and classifications are made based on voting schemes.

*Acknowledgements*—Part of this work was supported by GOA-Mefisto 666, IUAP V-10-29 and FWO project G.0407.02. TVG is on leave as a postdoctoral researcher with the FWO-Flanders at the K.U.Leuven - ESAT/SCD and is currently with Credit Methodology, Global Market Risk, Dexia Group. JS is a postdoctoral researcher with the FWO Flanders.

## References

1 Henley WE and Hand DJ (1997). Construction of a *k*-nearest neighbour credit-scoring system. *IMA J Math Appl Bus Ind* **8**: 305–321.
2 Steenackers A and Goovaerts MJ (1989). A credit scoring model for personal loans. *Insur Math Econ* **8**: 31–34.
3 West D (2000) Neural network credit scoring models. *Comput Opns Res* **27**: 1131–1152.
4 Stepanova M and Thomas LC (2001) PHAB scores: proportional hazards analysis behavioural scores. *J Opl Res Soc* **52**: 1007–1016.
5 Yobas MB, Crook JN and Ross P (2000). Credit scoring using neural and evolutionary techniques. *IMA J Math Appl Bus Ind* **11**: 111–125.
6 Desai VS, Crook JN and Overstreet Jr GA (1996) A comparison of neural networks and linear scoring models in the credit union environment. *Eur J Opl Res* **95**: 24–37.
7 Bishop CM (1995). *Neural Networks for Pattern Recognition*. Oxford University Press: Oxford, UK.
8 Thomas LC (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to customers. *Int J Forecast* **16**: 149–172.
9 Thomas LC, Edelman DB and Crook JN (2002). *Credit Scoring and Its Applications*. SIAM Monographs on Mathematical Modeling and Computation, SIAM: Philadelphia.
10 Glen JJ (2001). Classification accuracy in discriminant analysis: a mixed integer programming approach. *J Opl Res Soc* **52**: 328–339.
11 Vapnik V (1998). *Statistical Learning Theory*. John Wiley: New York.
12 Suykens JAK and Vandewalle J (1999). Least squares support vector machine classifiers. *Neural Process Lett* **9**: 293–300.
13 Suykens JAK, Van Gestel T, De Brabanter J, De Moor B and Vandewalle J (2002). *Least Squares Support Vector Machines*. World Scientific Publishing Co., Pte, Ltd: Singapore.
14 Van Gestel T, Suykens JAK, Baesens B, Viaene S, Vanthienen J, Dedene G, De Moor B and Vandewalle J (2003). Benchmarking least squares support vector machine classifiers. *Mach Learn*, forthcoming.
15 John GH and Langley P (1995) Estimating continuous distributions in Bayesian classifiers. Presented at the 11th Conference on Uncertainty in Artificial Intelligence (UAI), August 1995, Quebec, Canada.
16 Friedman N, Geiger D and Goldszmidt M (1997). Bayesian network classifiers. *Mach Learn* **29**: 131–163.
17 Quinlan JR (1993). *C4.5 Programs for Machine Learning*. Morgan Kaufmann: San Mateo, CA.
18 Hastie T, Tibshirani R and Friedman J (2001). *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*. Springer: New York.
19 Provost F, Fawcett T and Kohavi R (1998). The case against accuracy estimation for comparing classifiers. Presented at the 15th International Conference on Machine Learning (ICML), July 1998, San Francisco.
20 Fawcett T and Provost F (1997). Adaptive fraud detection. *Data Min Knowl Disc* **1**: 291–316.
21 Egan JP (1975). *Signal Detection Theory and ROC Analysis. Series in Cognition and Perception*. Academic Press: New York.
22 Swets JA and Pickett RM (1982). *Evaluation of Diagnostic Systems: Methods from Signal Detection Theory*. Academic Press: New York.
23 Baesens B, Viaene S, Van den Poel D, Vanthienen J and Dedene G (2002). Using Bayesian neural networks for repeat purchase modelling in direct marketing. *Eur J Opl Res* **138**: 191–211.
24 MacKay DJC (1992). The evidence framework applied to classification networks. *Neural Comput* **4**: 720–736.
25 Fayyad UM and Irani KB (1993). Multi-interval discretization of continuous-valued attributes for classification learning. Presented at the 13th International Joint Conference on Artificial Intelligence (IJCAI), August 1993, Chambery, France.
26 Sheskin DJ (2000). *Handbook of Parametric and Nonparametric Statistical Procedures*, 2nd edn. Chapman & Hall: Boca Raton, FL, US.
27 De Long ER, De Long DM and Clarke-Pearson DL (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics* **44**: 837–845.
28 Banasik J, Crook JN and Thomas LC (1996). Does scoring a subpopulation make a difference? *Int Rev Ret Dis Cons Res* **6**: 180–195.

*Received October 2001;*
*accepted January 2003 after one revision*