Differential Scorecards for Binary and Ordinal data

Pedro F. B. Silva, Jaime S. Cardoso
INESC Porto and Faculdade de Engenharia, Universidade do Porto
pfbsilva@fe.up.pt, jaime.cardoso@inescporto.pt

Abstract

Generalized additive models are well-known as a powerful and palatable predictive modeling technique. Scorecards, the discretized version of generalized additive models, are a long-established method in the industry, due to its balance between simplicity and performance. Scorecards are easy to apply and easy to understand. Moreover, in spite of their simplicity, scorecards can model nonlinear relationships between the inputs and the value to be predicted. In the scientific community, scorecards have been largely overlooked in favor of more recent models such as neural networks or support vector machines. In this paper we address scorecard development, introducing a new formulation more adapted to support regularization. We tackle both the binary and the ordinal data classification problems. In both settings, the proposed methodology shows advantages when evaluated in real datasets.

1. Introduction

Learning from examples is one of the most successful areas in machine learning, having predictive modelling as one of the fundamental learning frameworks. Several methods for the predictive modelling of data have been proposed during the last decades, ranging from simple techniques as homoscedastic Gaussian models to more evolved solutions as neural networks.

According to the nature of their assumptions, solutions can be generically considered as parametric or non-parametric. Both parametric and non-parametric approaches have different advantages and drawbacks. Parametric models are generally understood as more interpretable and simpler than their non-parametric equivalents. Also, they generally allow the input of relevant domain knowledge by experts which can frequently improve the quality of the solution for a given problem. On the other hand, non-parametric technologies can approximate any relationship, no matter how complicated, between a

set of predictive variables and a set of classification labels or a scoring variable, being therefore more versatile and flexible than parametric models. Still, their enormous flexibility is often accompanied by a lack of model interpretability which make their use unattractive in many important business applications of modelling like medical decision support and credit scoring.

Scorecards based on generalized additive models, a family of well-known, powerful, yet interpretable, predictive modelling techniques with a wide applicability range, have been used in business applications like credit scoring. Scorecards seem to be a good solution to fulfil the gap between parametric and non-parametric predictive modelling techniques, providing a good trade-off between interpretability and predictive power.

In the present work, we introduce a new formulation for scorecard design by adopting a differential encoding, which facilitates the regularization process. Additionally, we extend the formulation for ordinal data, which is not typically considered in scorecard design. The experimental comparison confirms the interest of the proposed approach.

This paper is organized as follows: Section I contains the present introduction; Section II presents the state of the art of predictive modelling techniques commonly used in practice to develop scorecards, and Section III summarizes the theoretical background needed for the construction of the scorecards used in this paper. In section IV we describe the general process of scorecard development and the proposed methodology; Section V summarizes the results of the experimental studied carried out in the context of this research and section VI presents the main conclusions of this paper.

2. Related Work

The predictive modelling problem consists in inferring, from a set of labelled observations (*training set*), the probable class of unknown observations (*test set*). Assume, in the following, that $X_1, X_2, \ldots, X_p$ is a collection of input continuous variables (features), Y is a target variable (to be predicted) and $\{(y_i, x_{i_1}, \ldots, x_{i_p})\}$, where $i = 1, \ldots, n$, is a set of n independent realizations of $Y, X_1, X_2, \ldots, X_p$. In this paper we are interested in both binary ($Y = \{0, 1\}$) and ordinal multiclass prediction ($Y = \{C_1, C_2, C_3, \ldots, C_K\}$ such that $C_1 \prec C_2 \prec C_3 \prec \ldots \prec C_K$, where $\prec$ defines an order in Y).

2.1. Generalized Additive Models

Generalized Additive Models [15] (GAM), introduced by Hastie and Tibshirani, are an extension of Generalized Linear Models (GLM) which, on their turn, are an extension of Linear Regression (LR).

Linear Regression assumes that $E(Y|X_1, X_2, \ldots, X_p)$ verifies a linear relationship in the input variables, i.e., $E(Y|X_1, X_2, \ldots, X_p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$ for some coefficients β_i, where $i = 0, 1, \ldots, p$. The general linear regression expression is $y_i = \beta_1 x_{i\,1} + \beta_2 x_{i\,2} + \ldots \beta_p x_{i\,p} + \epsilon_i$, where ϵ_i represents an unknown disturbance variable, assumed to follow a normal distribution.

The hypothesis of normally distributed regression errors is very restrictive and to overcome this problem, Nelder and Wedderburn proposed the Generalized Linear Models (GLM) [22], expanding the applicability of regression analysis beyond disturbances with normal distribution.

GLM consist of three parts: a random component, a systematic component and a link function aggregating both components. The random part of the GLM is related with the assumption that the target variable Y has exponential density of the form $f_Y(y, \theta, \phi) = \exp\{\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\}$, where θ and ϕ are named the natural and the scale parameters, respectively. The systematic component of the GLM consists in the assumption of a deterministic form for the predictor η, which is supposed to be linear on the predictive variables, i.e., $\eta = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$. The link function $g(.)$ brings the random and the systematic components together by establishing that the expected value of Y, $E(Y) = \mu$, is related to the predictive variables as $g(\mu) = \eta$.

GAM generalizes the previous regression procedure by substituting the linear predictor η with the more general version $\eta = s_0 + \sum_{j=1}^{p} s_j(X_j)$, where $s_j(.)$ are smooth functions standardized to verify $E(s_j(X_j)) = 0$ for $j = 1, \ldots, p$. Using GAM in practice relies on two well-known estimation algorithms from regression analysis, namely, the local scoring algorithm [5] and the backfitting algorithm [15], implemented as nested loops: inside each step of the local scoring algorithm (outer loop), a weighted backfitting algorithm (inner loop) is used until convergence or the residual sum of squares fails to decrease; then, based on the estimates of the backfitting algorithm, a new set of weights is calculated and the next iteration of the scoring algorithm starts. The scoring algorithm stops when a convergence

criterion is satisfied or when the deviance of the estimates stop decreasing.

The functions $s_j(.)$ may also be specified non-parametrically. As previously mentioned, the use of parametric techniques, although convenient for their interpretability, may result in hindered predictive power. Waal et al. [11] proposed a compromise between the simplicity and interpretability of logistic regression and the predictive power of neural networks by means of a generalized additive neural network (GANN). Although usually presented, as previously done, as extensions of linear models, GAM can be understood as constrained forms of artificial neural networks, in which case they acquire the designation of GANN. Estimating GANN consists in assembling a separate multilayer perceptron with a single hidden layer of neurons for each input variable X_j.

Nevertheless, in many applications found in literature, especially in the field of credit scoring, logistic regression [18], a particular member of the GAM family, is still a model of choice given its simplicity and the absolute need of interpretability.

2.2. Scorecards

Scorecards are GAMs, where the functions $s_j(.)$ are piece-wise constant. The general approach to scorecard development involves the binning of the predictive variables and the specification of a fitting objective function, which includes specifying a target for prediction and guiding the search for the best model. The literature on scorecard design is scarce, being more associated with commercial solutions [1, 2]. Next, we refresh some concepts to facilitate the presentation of scorecard design that will follow.

2.3. AdaBoost

Although not typically studied as such, Adaboost (Adaptive Boosting), when using a decision stump as the weak learner, can also be understood as a scorecard. AdaBoost is a boosting algorithm introduced by Freund and Schapire [14]. Boosting algorithms are a part of a big set of machine learning techniques called ensemble methods which general idea is to use several models to classify observations and combine them together to obtain a classifier with a predictive performance superior than any of its constituents. AdaBoost uses a *weak learner* to classify observations. A weak learner is defined as a

classifier which is only slightly correlated with the true data labels. In the case of binary prediction, a weak learner is a classifier which is only slightly better than throwing a coin and deciding an object's class according to the trial's result.

During each iteration, the algorithm trains a weak learner $h_t(x_i)$ using an iteratively determined distribution and selects the weak hypothesis minimizing the expected error rate. After selecting the best weak hypothesis h_t for the distribution D_t, the observations x_i correctly identified by h_t are weighted less than those misclassified, so that the algorithm will, when fitting a new weak hypothesis to D_{t+1} in the next iteration, select one such rule which identifies better those observations that its predecessor failed. The output of the AdaBoost algorithm is a final or combined hypothesis H. H is simply the sign of a weighted combination of the weak hypothesis, i.e., H is a weighted majority rule of the weak classifiers, $H(x) = \text{sign}\Big(\sum_{t=1}^{T} \alpha_t h_t(x)\Big)$.

AdaBoost is a state of the art, widely-used classification method. Addressing the aforementioned dichotomy between statistical methods and machine learning methods, Creamer and Freund [10] reported on the use of AdaBoost for the development of scorecards to classify equity investments. Their experimental study points out that AdaBoost performed in line with logistic regression, while being able to overcome some of its problems.

3. Background

We detail next some of the discretization techniques of continuous variable and discuss fitting objective functions together with regularization techniques.

3.1. Discretization Methods

Many machine learning algorithms are known to perform better if data has been discretized (binned) prior to classification [19]. Determining a discretization scheme for X_j, where $j = 1, \ldots, p$, consists simply in determining sets $D_j = \{d_{j\,1}, \ldots, d_{j\,n}\}$ such that $X_j = \bigcup_{i=1}^{n-1}[d_{j\,i}, d_{j\,i+1}]$ and $\bigcap_{i=1}^{n-1}]d_{j\,i}, d_{j\,i+1}[= \emptyset$, i.e., partitioning each feature $X_j, \forall j = 1, \ldots, p$. Each $d_{j\,i}$ is said to be a cut-off point for the variable X_j. In the context of predictive modelling, the general goal of binning is to discretize the continuous

features into a (possibly small) number of intervals with good class coherence, maximizing the inter-dependence between class labels and improving, therefore, prediction accuracy. Discretization methods work on the thin line between information quality, i.e., obtaining homogeneous bins according to the class attribute to predict and statistical quality, i.e., guaranteeing sufficient sample size in every bin to ensure inference quality.

During the last few years, many discretization methods have been proposed by machine learning researchers [19]. These can be grouped into several categories according to different aspects of their nature: 1) supervised or unsupervised, 2) direct or incremental, 3) global or local), 4) static or dynamic and 5) top-down or bottom-up.

Unsupervised methods do not make use of the class membership information during the discretization process. The simplest, most widely known and used algorithms of this kind include the equal-width and equal-frequency interval binning algorithms.

Equal-width and equal-frequency interval binning algorithms are very similar. The first consists simply in determining the minimum and maximum of the continuous feature and dividing this range into a user-defined number of bins with equal width. The second, on the other hand, divides the range into a user-defined number of bins such that each bin contains the same number of observations.

Opposed to unsupervised methods, supervised methods use the class membership estimation to estimate the cut-off points. A study carried out by Dougherty seems to point out that supervised methods outperform unsupervised methods [12].

Incremental methods start with a simple discretization scheme and improve this initial discretization by iteratively adding cut-off points until a certain tolerance value is achieved. Direct procedures, on the other hand, start with a user-defined number of intervals (cf. equal-width interval binning algorithm). The number of bins in an indirect method is not known *a priori* and depends on the toleration criterion specified.

Algorithms which discretize continuous features as a pre-processing step, i.e., prior to classification, are called global methods. Conversely, if discretization is done during the execution of a classifier as, for instance, the decision tree generating C4.5 algorithm, they are called local methods.

Static methods discretize one input feature at a time, while methods which take features' interaction into account are named dynamic.

Top-down algorithms start with one big interval containing all feature's values and partition it increasingly into smaller and smaller intervals. Bottom-up methods, on the contrary, consider the intervals defined by a set of boundary points and consecutively combine them together.

In the rest of this section, we introduce two state-of-the-art discretization techniques and reinterpret AdaBoost as a discretization method.

3.1.1 CAIM

The CAIM (Class-Attribute Interdependence Maximization) algorithm [21] is a supervised, indirect, global, static, top-down discretization method. CAIM is a fully automatic procedure, not requiring user intervention. The goal of the CAIM algorithm is to maximize the dependence between continuous features and the labelling variable, assuring at the same time that the number of discrete intervals obtained is small.

Assume X is one training set for a certain classification task containing M examples belonging to one of S classes. Let F indicate any of the continuous features present in X and let C denote a labelling variable. Remembering the definition given earlier, a discretization scheme on F is simply a set $D = \{d_0, d_1, \ldots, d_n\}$ partitioning F.

Consider that D is sorted, in which case $d_0 = \min(F)$ and $d_n = \max(F)$. Any value present in F can be associated with one and only one bin defined by the boundary set D. In this context, it is possible to construct a two-dimensional frequency matrix (called quanta matrix) relating the binning of F and the labelling variable C. Table 1 illustrates the general structure of a quanta matrix.

In Table 1, q_{ir} stands for the number of observations belonging, simultaneously, to the interval $]d_{r-1}, d_r]$ and the i^{th} class in C. M_{i+} denotes the number of observations belonging to the i^{th} class in C and M_{+r} represents the number of observations contained in the interval $]d_{r-1}, d_r]$, where $i = 1, 2, \ldots, S$ and $r = 1, 2, \ldots, n$. The CAIM criterion, measuring the dependence between the labelling variable C and the discretization variable D for feature F, is $\text{CAIM}(C, D|F) = n^{-1} \sum_{r=1}^{n} \frac{\max_r{}^2}{M_{+r}}$, where n is the

Class	Interval					Class Total
	$[d_0, d_1]$	...	$]d_{i-1}, d_r]$	...	$]d_{n-1}, d_n]$	
C_1	q_{11}	...	q_{1r}	...	q_{1n}	M_{1+}
$\vdots$	$\vdots$	...	$\vdots$	...	$\vdots$	$\vdots$
C_i	q_{i1}	...	q_{ir}	...	q_{in}	M_{i+}
$\vdots$	$\vdots$	...	$\vdots$	...	$\vdots$	$\vdots$
C_S	q_{S1}	...	q_{Sr}	...	q_{Sn}	M_{S+}
Interval Total	M_{+1}	...	M_{+r}	...	M_{+n}	M

Table 1. General structure of the quanta matrix

number of bins, $\max_r$ is the maximum of the q_{ir} values (i.e., the maximum of the r^{th} column of the quanta matrix).

The CAIM criterion favours discretization schemes in which the observations within each bin belong to the same class. The sum value is divided by the number of intervals n to favor discretization schemes with a small number of intervals. The higher the CAIM criterion value, the higher the correlation between the labelling variable C and the discretized variable D. Since finding the discretization scheme with the globally optimal CAIM value would require a strong computational cost, CAIM algorithm only finds a local maximum CAIM to generate a sub-optimal discretization scheme.

3.1.2 FCAIM

The FCAIM [20] (Fast Class-Attribute Interdependence) algorithm is a simple modification of the CAIM algorithm, trying to maintain all its advantages, namely, guaranteeing the highest interdependency between the labelling variable and the discretization scheme, and diminishing computational effort.

FCAIM and CAIM algorithms are identical apart from one instruction. CAIM algorithm initializes considering the minimum and maximum values of each continuous feature and additionally all the midpoints of all the feature's adjacent values. FCAIM algorithm initializes, on the contrary, considering the minimum and maximum values of each continuous feature and all the midpoints of all the feature's adjacent values belonging to different classes. This new definition allows speeding up the discretization process as the number of boundary points to test greatly decreases.

FCAIM criterion relies on the results published by Fayyad and Irani [13] stating that for discretization

schemes using entropy-based criteria, the generated boundary points are always between two data points belonging to two different classes. However, there is no proof that these results are valid for the CAIM criterion. Nonetheless, experimental results seem to indicate that the overall quality of the discretization provided by FCAIM is similar to that of CAIM [20].

3.1.3 AdaBoost

The AdaBoost algorithm can be used as a discretization method. AdaBoost will, for each iteration $t = 1, \ldots, T$, pick the best weak learner available over the features set. This procedure implicitly defines a cascade of selected features and their respective threshold values which can be used to assemble a discretization scheme for all the selected features. Unlike the previously introduced methods, AdaBoost does not necessarily discretize all input features. Used as a pre-processing method, AdaBoost is also a variable selection mechanism.

3.2. Regularization

As already referred, the goal of the traditional multivariate linear regression is to find the best linear combination of $X_1, X_2, \ldots, X_p$ that predicts Y. This problem can be formulated as finding adequate values of the coefficients w_j in the model $y_i = w_0 + \sum_{j=1}^{p} x_{ij} w_j$, where $i = 1, \ldots, n$.

Let X denote the $n \times p$ matrix which columns are $X_1, \ldots, X_p$, $W = (w_1, w_2, \ldots, w_p)^T$ and $Y = (y_1, y_2, \ldots, y_n)^T$. Setting as the fitting objective function the Sum of Squares (RSS), defined as $\mathrm{RSS} = \sum_{i=1}^{n} (y_i - w_0 - \sum_{j=1}^{p} x_{ij} w_j)^2$, we get, under typical assumptions, $W = (X^T X)^{-1} X^T Y$, where $X^T X$ is the Hessian matrix of RSS.

It is well known that the sole minimization of the error in the training set may incur in model overfitting. It is therefore typical to introduce regularization both to prevent overfitting and to solve ill-posed problems, since the Hessian matrix $X^T X$ is often singular or nearly singular. From a statistical point of view, regularization can be seen as introducing the prior knowledge that w_j should not be too large. From an optimization point of view, regularization can be interpreted as a balance mechanism between large values of the continuous features and the target variable, i.e., a compromise between minimizing

the training error and having small weights. In this sense, regularization is a method for improving performance.

In its simplest formulation, regularization consists in adding a positive constant to the diagonal of the Hessian matrix $X^T X$ (L2 regularization). The new expression for the explicit calculation of the weight vector W becomes $W = (X^T X + \lambda I)^{-1} X^T Y$, where I denotes a compatible identity matrix and $\lambda \in \mathbb{R}^+$ is the regularization parameter. The new vector W minimizes the modified loss function $\mathrm{RSS} = ||XW - y||^2 + \lambda ||W||^2$. In practice, w_0 is generally not penalized to avoid making the model dependent on the mean value of the data.

While being simple, L2 regularization does not exactly promote model parsimony. The models estimated using this technique typically have non-zero values for all weights, which often makes model interpretation more challenging. Addressing this problem, Tibshirani proposed the Least Absolute Selection and Shrinkage Operator (LASSO) [27]. LASSO is a technique to 'shrink' weights (same goal as L2 regularization) and simultaneously force higher values of the regularization parameter λ to make weights more similar with each other to minimize their joint L2-norm. This is achieved considering L1 regularization.

With L1 regulation, the RSS function becomes $\mathrm{RSS} = ||XW - Y||^2 + \lambda ||W||_1$, where $||.||_1$ stands for the L1 norm (taxicab). This new problem is still an unconstrained convex optimization problem in terms of W. However, RSS is now a non-differentiable function each time, at least, one element of W is zero. Consequentially, obtaining a closed form solution for W is impossible. Many solutions have been proposed to overcome this problem [24].

One interesting property of LASSO is that it works as a weight estimation and variable selection procedure simultaneously. LASSO assigns the weight 0 to all input features considered irrelevant. Experimental evidence shows that L1 regularization tends to outperform L2 regularization if irrelevant features are present among the input variables [16].

Despite being a great contribution to ensure model parsimony and interpretability, LASSO also has some limitations. Hastie and Zou [28] identified two problems with LASSO which compromise its success as a variable selection mechanism: if $p > n$, LASSO selects at the most n variables before

saturating and given a strong pairwise correlation between predictive variables, LASSO selects only one variable without caring which. On the other hand, Tibshirani observed that in the usual situation in which $n > p$, if high correlations between predictive variables are present, the prediction performance of LASSO is inferior to that of L2 regularization.

The elastic net [28] was proposed by Hastie and Zou to overcome these issues. Elastic nets bring the L1 and L2 regularization penalizations together by considering the criterion $L_W(\lambda_1, \lambda_2) = ||XW - Y||^2 + \lambda_1||W||_1 + \lambda_2||W||^2$, where λ_1 and λ_2 control the L1 and L2 regularization, respectively.

Defining $\alpha = \frac{\lambda_2}{\lambda_1+\lambda_2}$, this formulation is equivalent to determining $\hat{W}$ in the constrained least squares problem given by $\hat{W} = \arg\min_W ||XW - Y||^2$ subject to the constraint $(1-\alpha)||W||_1 + \alpha W||^2|| \leq t$ for some $t \in \mathbb{R}^+$. This constraint is called the elastic net penalty, which is a convex combination of the L1 and L2 penalties. Notice that when $\alpha = 1$ the elastic net becomes L2 regularization and when $\alpha = 0$, it becomes L1 regularization. Additionally, if $\alpha \in]0, 1[$, the elastic net penalty is convex and conserves, therefore, the properties of both L1 and L2 regulatization methods.

4. Constructing Scorecards

Consider, for instance, the problem of classifying a certain group of bank clients, who subscribed some credit service, in two groups, according to the risk of not being able to successfully repay their credit.

Suppose that previous experience from credit analysis suggests that X_1: "client's age in years", X_2: "client's monthly salary (in K Euros)", X_3: "number of client's late payments during the last year" and X_4: "percentage of credit paid" are adequate features for this predictive modelling problem. Assume as well that enough statistical history is available to assemble a discretization scheme for the features and weight the respective bins.

A scorecard is, in practice, a table-like, discrete classification scheme as depicted in Table 2.

To classify a new observation, it is necessary to calculate its score value and compare it with a pre-calculated threshold. Typically the threshold is set according to business rules and may be periodically updated due to changes in the operational conditions. For example, if a certain client is 23 years old,

X_1			X_2			X_3			X_4		
B	Range	W	B	Range	W	B	Value	W	B	Range	W
1	$[18, 25[$	1	1	$]0, 0.5]$	5	1	$[0, 1]$	20	1	$[0, 0.05]$	1
2	$[25, 40[$	3	2	$]0.5, 0.75]$	6	2	$]1, 3]$	8	2	$]0.05, 0.1]$	5
3	$[40, 65[$	5	3	$]0.75, 1]$	8	3	$]3, 4]$	5	3	$]0.1, 0.2]$	10
4	> 65	7	4	$]1, 1.5]$	10	4	> 4	1	4	$]0.2, 0.3]$	15
			5	$]1.5, 2]$	15				5	$]0.3, 0.5]$	20
			6	> 2	35				6	$]0.5, 0.8]$	40
									7	$]0.8, 1]$	100

Table 2. Example of a Scorecard. The 'W' column represents the score due to the corresponding feature.

has a monthly salary of 1.4K Euros, always paid his/her loan in time and has already repaid 30% of the credit, he/she will have a score of 46 (1+10+20+15) and, for a threshold of 30, belongs to the safe group.

Formally, a scorecard can be understood as a particular case of a generalized additive model, in which $s_j(X_j)$ are piecewise-constant functions. The determination of $s_j(.)$ consists in estimating a discretization scheme for each continuous feature X_j and the value of $s_j(X_j)$ for each bin in X_j.

Several alternatives exist to compute both the discretization scheme and the weighting factors which can or cannot include expert domain knowledge (cf. Section III). Ideally, the computation of the discretization scheme would be done together with the weight estimation process. This is, however, a difficult problem and in practice, weight estimation follows the determination of the discretization scheme.

Algorithm 1 summarizes the usual process of scorecard development. In practice, data cleansing takes place prior to scorecard construction.

Algorithm 1 Scorecard

Require: Training set X
1) Infer a discretization scheme D from X
2) Obtain P, a discretized version of X using D
3) Infer a set of weights W for each binned feature in P
return W

The techniques discussed for GLM (cf. Section III) can be used to estimate scorecard weights. In practice, it is just necessary to construct an appropriate input matrix P to replace the matrix X of the continuous features. Several option exist to perform this coding.

4.1. Weight of Evidence coding

A common practice is to compute the weights in two steps [1, 2]. First, for each feature, the relative importance (score) of each bin is estimated; then, the relative importance of each feature is optimized. A standard way to estimate the relative importance of each bin is using the weight of evidence (WOE) [1, 2]. This effectively replaces the original feature value by the WOE computed in the corresponding bin.

The WOE value is based on the odds ratio of regression analysis and measures the strength of a grouping variable to separate between two groups. WOE is commonly defined as $\mathrm{WOE} = \ln\left(\frac{\#\{Y==1\}}{\#\{Y==0\}}\right)$, where $Y == 1$ and $Y == 0$ codify, respectively, the positive and the negative outcomes. This definition allows the computation of the WOE value for each bin i of feature j, denoted WOE_{ji}. Since in practice some bins may have just a few observations (or no observations at all in the equal width discretization method), we adopt the Laplace's rule of succession (also known as Laplace correction or add-one smoothing) to estimate the underlying probabilities:

$$\mathrm{WOE} = \ln\left(\frac{\#\{Y == 1\} + 1}{\#\{Y == 0\} + 1}\right), \tag{1}$$

The additive scorecard becomes

$$Score = W_0 + \sum_{j=1}^{p}\sum_{i=2}^{q_j} W_j \mathrm{WOE}_{ji}. \tag{2}$$

Note that in this case the score value for bin i of feature j is $W_{ji} = W_j \mathrm{WOE}_{ji}$. The optimization of the weights among the features, W_j, consists in solving a linear problem in the WOE-coded data matrix, P_{WOE}.

4.2. 1-out-of-K coding

A potentially better approach than the WOE coding is to simultaneously optimize the weights within each feature and among all features. As already seen, the mathematical formula for an additive scorecard

can be written as

$$Score = s_0 + \sum_{j=1}^{p} s_j(X_j),$$

where s_0 is the intercept and $s_j(.)$ can be written as

$$s_j = \sum_{i=1}^{q_j} W_{ji} P_{ji}, \tag{3}$$

where q_j is the number of bins in feature j, W_{ji} is the score weight associated with the bin i of feature j and P_{ji} is a dummy indicator variable of a feature, e.g.,

$$P_{ji} = \begin{cases} 1 \text{ if } X_j \text{ takes value in bin } i \\ 0 \text{ otherwise} \end{cases}$$

This coding scheme of P_j is commonly named 1-out-of-q coding scheme in which P_j is a vector of length q_j such that if the feature takes value in bin i, then all elements P_{ji} of P_j are zero except one, which takes the value 1.

Using the fact that $\sum_i P_{ji} = 1$, $s_j(.)$ can be rewritten as

$$s_j = W_{j1} + \sum_{i=2}^{q_j} (W_{ji} - W_{j1}) P_{ji} \tag{4}$$

and the overall score as

$$Score = s_0 + \sum_{j=1}^{p} W_{j1} + \sum_{j=1}^{p} \sum_{i=2}^{q_j} (W_{ji} - W_{j1}) P_{ji} \tag{5}$$

Noticing that $\sum_j W_{j1}$ can be absorbed in the intercept, the model can finally be formulated as

$$Score = W_0 + \sum_{j=1}^{p} \sum_{i=2}^{q_j} W_{ji} P_{ji} \tag{6}$$

This results in a linear model defined over an extended feature set, where each feature is replaced by a vector of length $q_j - 1$. The training set can be summarized in a $n \times \left(\sum_{j=1}^{p}(q_j - 1)\right)$ P matrix. Each row of P corresponds to an observation in the discretized training set for which P_{ji} is set to 1 whenever the value of the j^{th} continuous feature lies within its $i^{\text{th}} \geq 2^{nd}$ bin. This concept is illustrated in Table 3.

	w_{12}	w_{13}	$\ldots$	w_{1q_1}	$\ldots$	w_{p2}	w_{p3}	$\ldots$	w_{pq_p}
Obs. 1	0	0	$\ldots$	1	$\ldots$	1	0	$\ldots$	0
Obs. 2	0	1	$\ldots$	0	$\ldots$	0	1	$\ldots$	0
$\vdots$	0	0	$\ldots$	0	$\ldots$	0	0	$\ldots$	0
Obs. N	1	0	$\ldots$	0	$\ldots$	0	0	$\ldots$	0

Table 3. Exemplification of the general structure of the indicator matrix P

4.3. Differential-coding

In this work we propose an alternative to the direct coding of the weights associated with each bin, coding the differences between weights of consecutive bins: $W_{ji} = W_{ji-1} + \delta_{ji}$. This options leads immediately to $W_{ji} = \sum_{\ell=1}^{i} \delta_{j\ell}$. Now the model (6) can the rewritten as

$$Score = W_0 + \sum_{j=1}^{p}\sum_{i=2}^{q_j}\left(\sum_{\ell=2}^{i}\delta_{j\ell}\right)P_{ji} \tag{7}$$

After a simple algebraic manipulation, one gets the final formulation:

$$Score = \delta_0 + \sum_{j=1}^{p}\sum_{i=2}^{q_j}\delta_{ji}P^{\star}_{ji}, \tag{8}$$

where $P^{\star}$ is a binary, $n \times \left(\sum_{j=1}^{p}(q_p - 1)\right)$ matrix, with each row corresponding to an observation in the discretized training set for which $P^{\star}_{ji}$ is set to 1 whenever the value of the j^{th} continuous feature lies in a bin ℓ with $\ell \geq i \geq 2$. This concept is illustrated in Table 4.

The key advantage of the proposed differential-coding comes into play when regularization is required. While in the 1-out-of-K coding regularization will enforce small values for the weights – which is likely a not intuitive assumption –, in the differential-coding regularization will promote smooth variations in the score between consecutive bins – a much more intuitive and desirable setting.

	δ_{12}	δ_{13}	...	δ_{1q_1}	...	δ_{p2}	δ_{p3}	...	δ_{pq_p}
Obs. 1	1	1	...	1	...	1	0	...	0
Obs. 2	1	1	...	0	...	1	1	...	0
⋮	0	0	...	0	...	0	0	...	0
Obs. N	1	0	...	0	...	0	0	...	0

Table 4. Exemplification of the general structure of the indicator matrix $P^{\star}$

4.4. Fitting Objective Function

All coding options we considered, WOE, 1-out-of-K, and differential-coding, result in linear classification models in the new space defined by P_{WOE}, P, and $P^{\star}$. Conventional linear methods may now be applied in order estimate scorecard weights. It is important to emphasize that, although the problem being solved is linear, the resulting scorecard is still a nonlinear model in the original features, due to the binning process.

Almost all the methodologies aim at minimizing the "misclassification error rate" (MER). Since the direct minimization of MER is a difficult problem, such goal is often replaced by more amenable loss functions as the hinge or squared error functions.

Other methodologies focus on the optimization of the "area under the curve" (AUC). The AUC is the area spanned by any Receiver Operating Characteristics (ROC) curve, which consists, in the case of a binary problem, in expressing the true positive rate as a function of the false positive rate. Frequently, when dealing with a binary classification problem, obtaining a classifier which minimizes MER may not be the most desired situation. In fact, it is often more important to achieve a high correlation between output scores and the probability of correct classification.

For even class distributions, the average AUC value is identical to the classifier's accuracy. On the other hand, for uneven class distributions, the average AUC value is a monotonic function of the classifier's accuracy. This means, as observed by Cortes and Mohri [9] that, on average, there is no gain in designing specific learning algorithms for AUC maximization for even class distributions. However, for uneven class distributions, classifiers exhibiting equally low accuracy values may yield significantly different AUC values.

Along the last decades, several estimation methods for AUC maximization have been proposed. Most

of these methods consist in a reformulation of existent estimation techniques through the customization of each method's objective function to achieve AUC maximization. Herbrich, Grapel and Obermayer [17] introduced a framework for ordinal regression using support vector machines, detailing the use of these classifiers for AUC optimization. Posterior research on AUC optimization focus on improvements of this technique and the study of other techniques such as linear and quadratic programming, least squares estimation, among others.

4.5. Scorecards for Ordinal Data

One of the main challenges when designing models for ordinal data is the proper integration of the order information in the design process. Several of the state of the art methods impose, explicitly or implicitly, constraints among the boundaries between the classes, assuming for instance that they should not cross each other [25, 26, 6]. In the linear setting, this process corresponds to find a direction common to all boundaries and a threshold specific to each boundary such that the MER is minimized. In case of the scorecard design, it consists in simultaneously finding the weights and an adequate set of boundary points $\{o_1, o_2, \ldots, o_n\}$, where $o_1 < o_2 < \ldots < o_n$, defining a partition of the score variable

$$\text{Score} =]-\infty, o_1] \cup]o_1, o_2] \cup \ldots \cup]o_{n-1}, o_n] \cup]o_n, +\infty[.$$

The label i of a new observation is then predicted using its estimated score s^* by determining the unique value of i satisfying $s^* > o_{i-1} \wedge s^* \leq o_i$.

In some state of the art approaches, the design of methods embodying this rational requires the design of algorithms specifically for this task [25, 26]. Other methods [6] can be framed under the single binary classifier (SBC) reduction, an approach for solving multiclass problems via binary classification relying on a single, standard binary classifier. SBC reductions can be obtained by embedding the original problem in a higher-dimensional space consisting of the original features, as well as one or more other features determined by fixed vectors, designated here as extension features. This embedding is implemented by replicating the training set points so that a copy of the original point is concatenated with

each of the extension features vectors. The binary labels of the replicated points are set to maintain a particular structure in the extended space. This construction results in an instance of an artificial binary problem, which is fed to a binary learning algorithm that outputs a single binary classifier. To classify a new point, the point is replicated and extended similarly and the resulting replicas are fed to the binary classifier, which generates a number of signals, one for each replica. The class is determined as a function of these signals. This is the approach we propose and adopt in this work.

In the following experimental work, we compare the aforementioned approaches for scorecard design for data with both binary and ordinal targets.

5. Experimental Study

The results of the experimental study carried out to evaluate the performance of the proposed differential-coding and of different alternatives for scorecard development are presented next.

5.1. Datasets

Table 5 presents the characteristics of the datasets used.

Group 1 - Binary Target			
Acronym	Nr. Obs.	Nr. Var.	Class Distribution
APP [3]	106	7	[85, 21]
IDS [4]	351	32	[126, 225]
LIVER [4]	345	7	[145, 200]
PIDD [4]	725	8	[476, 249]
WDBC [4]	569	32	[357, 212]
Group 2 - Ordinal Target			
Acronym	Nr. Obs.	Nr. Var.	Class Distribution
BALANCE	625	4	[288, 49, 288]
ERA	1000	4	[92, 142, 181, 172, 158, 118, 88, 31, 18]
ESL	488	4	[2, 12, 38, 100, 116, 135, 62, 19, 4]
LEV	1000	4	[93, 280, 403, 197, 27]
SWD	1000	10	[284, 438, 278]
BCCT	1144	30	[160, 592, 272, 120]

Table 5. List of considered datasets

Datasets from Group 1 were used as collected from the indicated sources with exception of IDS and LIVER. Both the missing values of IDS as well as its second feature (column of zeros) were removed.

The last feature of LIVER was removed, as it was not a proper continuous feature. All labelling variables were codified as $\{0, 1\}$.

Datasets from Group 2 were collected from the Weka website and they were adjusted so that both discrete features and the labelling variables would match the coding $\{1, 2, \ldots, N\}$. The exception is the BCCT dataset from a breast cancer application [7].

5.2. Methodology

5.2.1 Binary Problems

Each discretization method introduced (AdaBoost[1], CAIM, FCAIM, Equal Width[2] and Equal Frequency[2]) was combined with a different weight estimation method to construct a unique scorecard.

The following objective functions aiming for MER minimization were considered, namely: Least Squares (RSS), Maximum Likelihood (GLM with binomial response and logit link function) and margin maximization (linear SVM).

LS and ML objective functions were regularized by means of an elastic net, with the α parameter being determined by grid search over the range $\{0.01, 0.1, 0.4, 0.6, 0.9, 0.99\}$. These calculations were performed using the functions *lasso* and *lassoglm* in Matlab R2013a.

The linear SVM parameter, C, was determined by grid-search over the range $\{2^{-10}, 2^{-9}, \ldots, 2^{9}, 2^{10}\}$. All results involving SVM were obtained through the use of the library LIBSVM [8] (Version 3.17).

Additionally, the objective functions AUC-SVM [17] and AUC-RLS [23] using, respectively, support vector machines and regularized least squares were considered for comparison. AUC-RLS was regularized by means of Ridge regression with $\lambda \in \{10^{-8}, 10^{-4}, 10^{-1}, 1, 5, 10, 15, 20, 50, 100, 250\}$ and AUC-SVM used a linear SVM as previously described.

All scorecard weight estimation techniques except AUC-SVM used 10-fold cross validation. AUC-SVM was trained with 3-fold cross validation for computational reasons.

To assess the relative performance of scorecards we have applied other classifiers (logistic regression,

[1] AdaBoost's parameter T was set to 100.
[2] A division in 10 bins was considered.

nonlinear SVM, LDA and AdaBoost) to the same cross validation sets. SVM were tuned considering a radial basis function, with parameters (γ, C) determined by grid search over $\{2^{-10}, 2^{-9}, \ldots, 2^{9}, 2^{10}\} \times \{2^{-10}, 2^{-9}, \ldots, 2^{9}, 2^{10}\}$. Logistic regression was obtained using Matlab's function *glmfit*. These classifiers were selected to compare scorecard performance with simple, well-established linear techniques like LDA and powerful, more recent, nonlinear techniques like SVM. Furthermore, AdaBoost was included given its usual good performance and it is also another option for scorecard development. Logistic regression was included as it is considered state-of-the-art in many practical uses of scorecard development.

5.2.2 Ordinal Problems

In this paper, we considered two alternatives for the estimation of scorecard weights for ordinal data, namely regularized least squares (oRLS) and the data replication method mapped to linear support vector machines (oSVM). Cardoso et. al. [6] introduced the data replication method, a new machine learning paradigm specifically intended for ordinal classification, which reduces the ordinal problem to a single, standard binary problem that can be solved by applying well-established algorithms. For the oRLS method, class values were replaced by the middle values of K equal-sized intervals in the range $[0, 1)$ and the weights found by square error minimization. Method oRLS were regularized in the same fashion as AUC-RLS and oSVM, respectively. Both methods were trained with 10-fold cross validation.

Cardoso et. al. [7] carried an experimental study on ordinal data classification using kernel discriminant analysis. This study compared three linear discriminant analysis based approaches: a first model based on the Frank and Hall framework (FH_LDA), a second model based on the data replication method (oLDA) and a third, more recent model using kernel discriminant learning ordinal regression (KDLOR). Their conclusions point out that KDLOR does not offer any advantage over FH_LDA and oLDA and that these two were comparable in terms of accuracy, with a slight advantage to oLDA.

Motivated by these results, we have also compared scorecards' performance with oLDA. For computational reasons, oLDA was trained with 3-fold cross validation. The developed classifiers for ordinal data were evaluated considering the mean absolute error (MAE).

5.2.3 Test procedure

Datasets were randomly split into training (75% of data) and test sets (25% of data). The splitting process was repeated 40 times to ensure stability.

5.3. Results

5.3.1 Binary Problems

Tables 6, 7, and 8 resume the misclassification error rates obtained for the developed scorecards.

	APP			IDS			LIVER			PIDD			WDBC		
	LS	ML	SVM	LS	ML	SVM	LS	ML	SVM	LS	ML	SVM	LS	ML	SVM
AdaBoost	17.4	16.8	18.5	10.1	9.2	10.1	33.9	31.9	37.2	27.0	26.1	24.7	5.3	3.9	4.0
CAIM	13.5	13.7	15.2	7.9	7.9	8.2	38.2	37.9	38.2	25.9	26.1	26.0	5.2	4.5	6.4
FCAIM	13.2	12.6	17.5	8.0	8.2	8.1	40.5	40.7	40.6	26.5	26.5	26.6	5.1	4.6	6.3
Eq. Width	15.9	14.8	15.6	11.2	7.3	8.7	35.2	31.6	38.4	25.7	23.8	23.7	3.0	3.2	4.8
Eq. Freq.	14.7	14.7	16.3	11.8	7.0	10.8	34.9	31.7	38.4	26.1	24.0	23.5	2.9	3.0	4.4

Table 6. Scorecards: MER (%) with WOE-coding

	APP			IDS			LIVER			PIDD			WDBC		
	LS	ML	SVM	LS	ML	SVM	LS	ML	SVM	LS	ML	SVM	LS	ML	SVM
AdaBoost	19.9	18.8	20.1	10.2	10.2	10.6	34.4	34.4	32.2	26.9	26.9	28.0	5.1	3.9	4.1
CAIM	12.3	11.7	13.9	7.7	7.8	7.7	38.1	38.0	38.3	26.2	26.2	26.9	4.9	4.4	4.5
FCAIM	12.7	12.5	13.7	7.7	8.2	8.3	40.6	40.6	38.0	26.7	26.5	26.6	5.3	4.5	4.6
Eq. Width	20.7	19.4	17.1	11.5	11.1	12.4	42.5	43.7	35.6	26.4	26.3	26.0	4.8	4.5	4.7
Eq. Freq.	18.9	18.1	15.3	12.0	11.3	12.2	36.8	41.7	34.5	26.2	26.5	26.7	4.7	5.0	5.4

Table 7. Scorecards: MER (%) with 1-out-of-K coding

	APP			IDS			LIVER			PIDD			WDBC		
	LS	ML	SVM	LS	ML	SVM	LS	ML	SVM	LS	ML	SVM	LS	ML	SVM
AdaBoost	19.3	17.9	19.1	9.4	9.6	9.5	32.4	31.9	31.7	26.3	25.8	27.0	5.0	3.9	3.8
CAIM	12.4	12.0	13.9	7.7	7.9	7.7	38.1	38.1	38.3	26.1	26.1	26.9	5.0	4.3	4.5
FCAIM	12.7	12.5	13.7	7.6	8.1	8.3	40.6	40.5	38.0	25.7	26.4	26.6	5.1	4.4	4.6
Eq. Width	13.7	12.7	13.9	7.0	7.5	7.8	32.2	30.9	31.1	24.1	24.2	24.6	3.9	2.8	3.1
Eq. Freq.	12.9	12.5	12.3	7.4	7.8	7.8	32.1	31.1	31.7	24.4	24.1	23.8	3.9	2.5	2.6

Table 8. Scorecards: MER (%) with differential-coding

The results exhibited in Tables 7 and 8 indicate that the proposed differential-coding generally outperforms the usual WOE coding, and also the 1-out-of-K formulation with exception of some scorecards constructed with the supervised methods CAIM and FCAIM. Wilcoxon signed rank test was applied to evaluate the statistical significance of these exceptions. For an α value of 0.01, all differences were not statistically significant.

Furthermore, results indicate that the supervised discretization techniques CAIM and FCAIM outperform the non-supervised techniques "equal width" and "equal frequency" just when scorecard weights

are estimated with the 1-out-of-K and WOE codings. Within the δ-coding framework, both supervised and non-supervised discretization techniques led to similar results. This validates our remark on the differential-coding, stating that it is able to correct possible inadequate discretization schemes (unsupervised methods were not optimized).

Within the supervised discretization techniques, CAIM seems to outperform FCAIM, although the differences were generally small. FCAIM may be preferred over CAIM given its superior computational efficiency.

Maximum likelihood based weight estimation seems to outperform both LS and SVM based versions. Exceptions to this behaviour in WOE, 1-out-of-K, and differential coding, which were shown to be statistically significant, were underlined in Tables 6, 7, and 8. The LS results present in Tables 6, 7, and 8 were obtained with elastic net regularization. Nonetheless, the differences to L2 regularization were meaningless. Consequently, since L2 regularization is more simple and computationally cheaper, Ridge least squares estimation was selected to compare MER minimization and AUC optimization approaches. Table 9 summarizes the obtained results for the proposed differential-coding.

		APP		IDS		LIVER		PIDD		WDBC	
		MER	AUC	MER	AUC	MER	AUC	MER	AUC	MER	AUC
Ridge LS	AdaBoost	18.4	0.79	9.2	0.96	31.2	0.74	26.1	0.80	4.8	0.99
	CAIM	12.8	0.82	7.9	0.95	38.0	0.63	25.7	0.76	5.0	0.99
	FCAIM	12.9	0.81	8.0	0.95	37.4	0.63	26.3	0.76	5.0	0.99
	Eq. Width	13.8	0.85	6.8	0.97	31.1	0.74	24.1	0.83	3.0	0.99
	Eq. Freq.	10.8	0.87	6.7	0.97	31.3	0.73	24.8	0.83	3.5	0.99
AUC-RLS	AdaBoost	20.1	0.76	10.8	0.94	37.4	0.67	39.3	0.57	5.9	0.98
	CAIM	14.1	0.82	9.7	0.94	39.7	0.59	27.8	0.67	6.5	0.97
	FCAIM	13.9	0.82	9.3	0.94	39.2	0.58	27.9	0.67	6.5	0.97
	Eq. Width	15.2	0.83	13.8	0.89	40.7	0.61	35.0	0.50	10.2	0.96
	Eq. Freq.	13.7	0.82	15.5	0.89	40.6	0.62	35.1	0.51	11.9	0.95
AUC-SVM	AdaBoost	17.9	0.81	10.3	0.95	32.6	0.73	26.9	0.79	4.0	0.99
	CAIM	13.1	0.81	10.6	0.92	37.9	0.63	26.6	0.75	5.0	0.99
	FCAIM	13.5	0.78	10.4	0.92	37.8	0.63	26.6	0.75	5.3	0.99
	Eq. Width	16.1	0.81	8.9	0.97	35.1	0.69	25.6	0.80	3.0	0.99
	Eq. Freq.	16.7	0.82	9.1	0.96	33.7	0.71	26.3	0.80	2.7	0.99

Table 9. Scorecards: MER (%) and AUC with differential-coding

Table 9 shows that Ridge LS consistently outperforms the alternative methods both in terms of MER and AUC criteria. Furthermore, AUC-SVM provides, on average, more satisfactory results than AUC-RLS. These results do not support the use of the AUC as fitting objective function, even for unbalanced datasets where it could be expected some advantage. Ridge LS and AUC-SVM provided comparable MER values in, for example, LIVER and PIDD datasets but AUC-SVM did not conduct to better AUC

values. A possible explanation for this finding lays in the fact that all features involved in the process of scorecard weight estimation are binary and the matrices P and $P^\star$ have very specific structures, while the AUC methods were originally formulated for continuous input features, under more standard conditions.

Table 10 allows the comparison of the best performing scorecard (Tables 6, 7, and 8) with other classifiers.

Classifier	APP	IDS	LIVER	PIDD	WDBC
Scorecard	**11.7**	7.0	30.9	23.8	**2.5**
AdaBoost	16.4	10.0	31.4	24.5	3.8
Log. Reg.	13.6	15.7	31.7	**23.0**	5.2
GAM	14.3	18.3	46.8	25.4	5.4
LDA	12.6	15.4	35.8	24.2	3.8
SVM	13.0	**6.1**	**30.4**	23.3	2.6

Table 10. Scorecards vs. Other Classifiers: Misclassification Error Rate (%)

The obtained results indicate that scorecards achieved satisfactory performance, scoring the best in 2 of the 5 considered datasets and among the top 3 in the rest. These results reinforce that scorecards may be a reasonable tradeoff between interpretability and predictive power.

Finally, it is interesting to note that the proposed differential scorecard compares favorably with adaboost. As already noticed, Adaboost model can be re-shaped into the scorecard format. However, the opposite is also true: the scorecard model can be re-written as the sign of a weighted combination of decision stumps. Therefore, the hypothesis space of concepts is the same for both approaches and under this perspective, the differential scorecard can be understood as an improved Adaboost.

5.3.2 Ordinal Problems

Table 11 presents the MAE for the considered methods for ordinal data classification.

Our empirical results suggest that the performance difference between scorecards developed with oRLS and linear oSVM is not statistically significant. Moreover, the obtained results do not suggest that scorecards perform worse than the nonlinear (RBF kernel) ordinal method oLDA when classifying ordinal data. In fact, although the differences found were not statistically significant, scorecards (oRLS and oSVM) outperformed the ordinal method oLDA in 5 out of 6 times. Moreover, scorecards have

	Scorecard		oLDA	AdaBoost
	oRLS	oSVM		
BALANCE	0.06	**0.00**	0.05	0.23
ERA	**1.26**	1.30	1.28	1.48
ESL	0.34	0.35	**0.33**	0.62
LEV	**0.40**	0.42	0.44	0.60
SWD	0.46	**0.44**	0.47	0.53
BCCT	0.55	0.53	0.64	**0.38**

Table 11. Scorecards vs. oLDA and AdaBoost: Mean Absolute Error

significantly outperformed AdaBoost when classifying ordinal data with the sole exception of the BCCT dataset.

6. Conclusions

Despite the myriad of techniques that address classification problems, most of them do not find wide acceptance in the industry due to the lack of interpretability. Scorecards are one of the clear exceptions, being well-established in some fields.

In this paper we address the improvement of the performance of scorecards by properly reformulating the learning problems. The differential encoding, representing the difference between consecutive weights, facilitates the design problem. The experimental study carried out seems also to indicate that scorecards developed with the proposed differential coding technique outperform AdaBoost and perform in line with the best performing (but non-interpretable) classifiers. The extension to ordinal data classification has also proven very competitive with state of the art methods. These results suggest a wider attention from the scientific community to scorecard design.

In future work, we plan to study improvement to scorecard design, by iterating between the variable discretization and weight optimization. Moreover, since the P and $P^\star$ matrices present a very specific structure, we plan to study optimization procedures of the fitting objective function that take advantage of that knowledge.

Acknowledgement

This work was financed by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT within project PTDC/SAUENB/114951/2009. The work leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement nº FP7–600948.

References

[1] Building Credit Scoring Models with SAS Enterprise Miner. Technical report, SAS Institute, 2006. 4, 13

[2] Introduction to Scorecard for FICO Model Builder. Technical report, FICO, 2006. 4, 13

[3] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing 17:2-3 (2011) 255-287. 18

[4] K. Bache and M. Lichman. UCI machine learning repository, 2013. 18

[5] L. Breiman and J. H. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80(391):580–598, 1985. 3

[6] J. S. Cardoso and J. F. Pinto da Costa. Learning to classify ordinal data: The data replication method. *J. Mach. Learn. Res.*, 8:1393–1429, Dec. 2007. 17, 20

[7] J. S. Cardoso, R. Sousa, and I. Domingues. Ordinal data classification using kernel discriminant analysis: A comparison of three approaches. In *Proceedings of The Eleventh International Conference on Machine Learning and Applications (ICMLA)*, pages 473–477, 2012. 19, 20

[8] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. 19

[9] C. Cortes and M. Mohri. Auc optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems*. MIT Press, 2003. 16

[10] G. Creamer and Y. Freund. Using adaboost for equity investment scorecards. In *Howe School Research Paper; NIPS Workshop Machine Learning in Finance, 2005, Whistler, British Columbia, Canada.*, 2005. 5

[11] D. de Waal, J. du Toit, and T. de la Rey. An investigation into the use of generalized additive neural networks in credit scoring. In *Credit Scoring and Credit Control IX*, 2005. 4

[12] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, 1995. 6

[13] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 1022–1029, 1993. 8

[14] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, Euro-COLT '95, pages 23–37, London, UK, UK, 1995. Springer-Verlag. 4

[15] T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, 1:297–318, 1986. 3

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001. 10

[17] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *International Conference on Artificial Neural Networks*, pages 97–102, 1999. 17, 19

[18] D. W. Hosmer and S. Lemeshow. *Applied logistic regression (Wiley Series in probability and statistics)*. Wiley-Interscience Publication, 2 edition, 2000. 4

[19] S. Kotsiantis and D. Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006. 5, 6

[20] L. A. Kurgan and K. J. Cios. Fast class-attribute interdependence (caim) discretization algorithm. In C. Press, editor, *International Conference on Machine Learning and Applications*, 2003. 8, 9

[21] L. A. Kurgan and K. J. Cios. Caim discretization algorithm. *IEEE Trans. on Knowl. and Data Eng.*, 16(2):145–153, Feb. 2004. 7

[22] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society, Series A, General*, 135:370–384, 1972. 3

[23] T. Pahikkala, A. Airola, H. Suominen, J. Boberg, and T. Salakoski. Efficient auc maximization with regularized least-squares. In A. Holst, P. Kreuger, and P. Funk, editors, *SCAI*, volume 173 of *Frontiers in Artificial Intelligence and Applications*, pages 12–19. IOS Press, 2008. 19

[24] M. Schmidt. Least squares optimization with l1-norm regularization, 2005. 10

[25] A. Shashua and A. Levin. Ranking with large margin principle: Two approaches. In *Neural Information and Processing Systems (NIPS)*, 2002. 17

[26] B. Sun, J. Li, D. D. Wu, X. Zhang, and W. Li. Kernel discriminant learning for ordinal regression. *IEEE Transactions on Knowledge and Data Engineering*, 22:906–910, 2010. 17

[27] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994. 10

[28] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005. 10, 11