Profit Maximizing Logistic Model for Customer Churn Prediction Using Genetic Algorithms

Eugen Stripling[a,*], Seppe vanden Broucke[a], Katrien Antonio[b,c], Bart Baesens[a,d], Monique Snoeck[a]

[a] *Department of Decision Sciences and Information Management, KU Leuven, Leuven, Belgium*
[b] *Department of Accountancy, Finance and Insurance, KU Leuven, Leuven, Belgium*
[c] *Section of Quantitative Economics, University of Amsterdam, Amsterdam, The Nederlands*
[d] *School of Management, University of Southampton, Southampton, UK*

Abstract

To detect churners in a vast customer base, as is the case with telephone service providers, companies heavily rely on predictive churn models to remain competitive in a saturated market. In previous work, the expected maximum profit measure for customer churn (EMPC) has been proposed in order to determine the most profitable churn model. However, profit concerns are not directly integrated into the model construction. Therefore, we present a classifier, named ProfLogit, that maximizes the EMPC in the training step using a genetic algorithm, where ProfLogit's interior model structure resembles a lasso-regularized logistic model. Additionally, we introduce threshold-independent recall and precision measures based on the expected profit maximizing fraction, which is derived from the EMPC framework. Our proposed technique aims to construct profitable churn models for retention campaigns to satisfy the business requirement of profit maximization. In a benchmark study with nine real-life data sets, ProfLogit exhibits the overall highest, out-of-sample EMPC performance as well as the overall best, profit-based precision and recall values. As a result of the lasso resemblance, ProfLogit also performs a profit-based feature selection in which features are selected that would otherwise be excluded with an accuracy-based measure, which is another noteworthy finding.

Keywords: Data mining, customer churn prediction, lasso-regularized logistic regression model, profit-based model evaluation, real-coded genetic algorithm

1. Introduction

In saturated markets such as the telephone service industry, companies constantly endeavor to identify customers who intend to voluntarily switch to a competitor. Attracting new customers in such markets is eminently challenging, and costs five to six times more than to prevent existing customers from churning [1]. However, detecting would-be churners out of typically millions of customers is a difficult task. For that reason, companies unavoidably have to rely on predictive churn models if they wish to remain competitive. As a consequence, predictive classification techniques for customer churn are increasingly researched [2]. Yet, these models often do not directly focus on the most important business requirement: *profit maximization*. Therefore, correctly identifying potential churners is one challenge; another is to also detect those who are the most profitable to the business. The ideal churn model is thus capable of effectively identifying churners and simultaneously taking profit concerns of the business into account.

*Corresponding author: eugen.stripling@kuleuven.be

Frequently, the winning churn model is selected based on accuracy related performance measures, which do not account for profit maximization in any manner. Considering binary classification problems, for instance, a popular choice for model selection is the *area under the ROC curve* (AUC), because of its simplicity and objectivity. It provides an intuitive interpretation and a holistic summary of the classification performance. However, Hand [3] showed that the AUC implicitly imposes unrealistic assumptions about the misclassification costs which also alter across classifiers. Reasonably, misclassification costs are a property of the classification problem, and should not depend on the applied classifier. For appropriate model selection, Hand therefore proposed the H measure that minimizes a cost function with fixed misclassification costs across classifiers.

Next to costs, it is also generally recommended to incorporate the *benefits* of making a correct classification into the performance metric [4]. For predictive churn models, Verbraken et al. [5] proposed a profit-based performance metric, the *expected maximum profit measure for customer churn* (EMPC), that allows identifying the most profitable model. Performance is measured based on the average classification profit with costs and benefits specified that are associated with a retention campaign. Additionally, they proposed the *expected profit maximizing fraction for customer churn* ($\bar{\eta}_{empc}$) that determines the optimal fraction of the customer base to target in the retention campaign for maximum profit. In an extensive case study, Verbraken et al. [5] showed that there are great discrepancies between the EMPC and AUC, and that model selection based on the AUC leads to suboptimal profit. For retention campaigns, the authors therefore recommend to use the EMPC for the selection of the churn model to attain maximum profit. Although the EMPC permits a *profit-based model evaluation*, profit concerns are however not directly incorporated into the *model construction*.

Hence, we propose a profit maximizing classifier for customer churn, called ProfLogit, that optimizes the EMPC in its training step. In our approach, a logistic model structure is utilized to compute churn scores, which are required for the profit measure, but the regression coefficients of the model are optimized according to the EMPC using a real-coded genetic algorithm (RGA). The choice for the usage of a RGA is justified because classical gradient-based optimization methods such as BFGS are not applicable for the EMPC optimization. Additional motivation for the application of RGA in contrast to other nature-inspired optimization algorithms is provided in Appendix A. In this paper, we refine our previous body of work [6] by incorporating significant enhancements such as a *lasso-regularized fitness function* as well as a *soft-thresholding operator* into ProfLogit. Additionally, we introduce precision and recall measures, along with the F_1 measure, that are defined based on the expected profit maximizing fraction $\bar{\eta}_{empc}$, which frees users from manually specifying a classification threshold. Our contributions can be summarized as follows:

- Providing empirical evidence for the feasibility of profit maximizing modeling through RGA, showing that the proposed approach attains overall highest profitability.
- Significant improvements to the fitness functions that make the evolutionary search more efficient, yielding a higher average EMPC performance.
- Introducing profit-based precision and recall measures, as well as a F_1 measure thereof, that do

not depend on the classification threshold and account for maximum profit.

The remainder of this paper is structured as follows. In the next section, we discuss the essential building blocks that are relevant to ProfLogit, followed by detailed explanations of our proposed approach in Section 3. The subsequent section describes the experimental setup, and the results of an extensive benchmarking study as well as a discussion thereof. Finally, we conclude the paper by summarizing the research findings in Section 5.

2. Preliminaries

2.1. Notation

For computational reasons, the EMPC measure requires that the class of interest (i.e., churn) is encoded as zero [7, p. 37]. Hence, throughout the text, the binary response variable $Y \in \{0, 1\}$ signifies 'churn' if Y takes the value zero and 'no churn' if it takes the value one. A positive side effect of using this nonstandard notation is that it simplifies the mathematical description of the profit-based performance measures [8].

Furthermore, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ denotes the set of N observed predictor-response pairs, where $y_i \in \{0, 1\}$ symbolizes the response and $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T$ represents the p associated predictor variables (or features) of observation i.

2.2. Logistic Regression Model for Churn Prediction

The logistic regression model, also called logit model, is a popular classification technique that models the nonlinear relationship between a binary response variable and a set of features [9]. Given a data set $\mathcal{D}$, logistic regression models the likelihood of churn for instance i as a conditional probability:

$$\Pr(Y = 0 \mid \mathbf{x}_i) = \frac{e^{\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i}}{1 + e^{\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i}}, \tag{1}$$

where $\beta_0 \in \mathbb{R}$ represents the intercept, $\boldsymbol{\beta} \in \mathbb{R}^p$ is the p-dimensional vector of regression coefficients, and $\mathbf{x}_i, Y$ and $\mathcal{D}$ as defined in the notation section. For notational convenience, we define the churn probability (or score) of instance i as $s(\mathbf{x}_i) \coloneqq \Pr(Y = 0 \mid \mathbf{x}_i)$ and the set of all churn scores as s. It is obvious from (1) that the churn scores lie between zero and one. Given the adapted notation, note that a lower churn score indicates a higher likelihood of churning.

To fit the logistic model, the binomial log-likelihood function of the data, $l(\cdot)$, is maximized, which in turn yields the maximum likelihood estimates for the unknown parameters β_0 and $\boldsymbol{\beta}$. Yet, for our purposes, we will consider an objective function, $Q_\lambda^l(\cdot)$, that is the lasso-regularized version of the log-likelihood function [10]:

$$Q_\lambda^l(\beta_0, \boldsymbol{\beta}) = l(\beta_0, \boldsymbol{\beta}) - \lambda ||\boldsymbol{\beta}||_{\ell_1} \tag{2}$$

with

$$l(\beta_0, \boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} \left[(1 - y_i)(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i) - \log\left(1 + e^{\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i}\right) \right] \tag{3}$$

where $\lambda \geq 0$ is the regularization parameter and $||\boldsymbol{\beta}||_{\ell_1} = \sum_{j=1}^p |\beta_j|$ is the ℓ_1-norm of $\boldsymbol{\beta}$. Note that the lasso regularization only penalizes the regression coefficients in $\boldsymbol{\beta}$—not the intercept β_0. Clearly, the

Table 1: Confusion Matrix with Cost Benefit Structure

Predicted class	**Actual class**	
	Class 0	Class 1
Class 0	$\pi_0 F_0(t) N$ $[b_0 = c(0 \mid 0)]$	$\pi_1 F_1(t) N$ $[c_1 = c(0 \mid 1)]$
Class 1	$\pi_0(1 - F_0(t)) N$ $[c_0 = c(1 \mid 0)]$	$\pi_1(1 - F_1(t)) N$ $[b_1 = c(1 \mid 1)]$

Confusion matrix with associated benefits (b_k) and costs (c_k), $k \in \{0, 1\}$, for a correct and incorrect classification, respectively [5]. For the EMPC measure, churn is encoded as zero.

larger λ, the stronger the lasso penalty. Typically, the predictors are standardized in the lasso model so that they have zero mean (i.e., $\frac{1}{N}\sum_{i=1}^{N} x_{ij} = 0$) and unit variance (i.e., $\frac{1}{N}\sum_{i=1}^{N} x_{ij}^2 = 1$) [10]. The regularization parameter λ cannot be directly estimated from the data and has to be determined by means of hyperparameter optimization strategies such as grid search in combination with cross-validation. Assuming the optimal λ value has been found, the lasso-regularized logistic regression aims to achieve a good balance between model fit and model complexity in which only predictors with sufficiently large predictive power have a nonzero regression coefficient.

2.3. Profit-based Classification Performance Evaluation

A classification performance measure is necessary to assess the quality of the churn model, which in turn allows selecting the best classifier from a set of candidate models. In this subsection, we discuss common performance measures used to evaluate models for binary classification problems, and elaborate on the expected maximum profit measure for customer churn (EMPC).

Given a classification threshold t, the *confusion matrix* can be constructed which tabulates the numbers of correct and incorrect classifications based on the churn scores produced by the predictive model (Table 1). Note that π_k denotes the prior class probability of class k, $k \in \{0, 1\}$, where π_0 represents the base churn rate. And, $f_k(s)$ and $F_k(s)$ respectively stand for the probability density function and the cumulative distribution function of class k computed based on the churn scores s. To make crisp class predictions, all instances with s smaller than t are classified as churners (i.e., class 0), whereas instances with s larger than t are classified as nonchurners (i.e., class 1). Generally, labeling an instance from class k as a class l instance is associated with a cost or benefit $c(l \mid k)$, $l, k \in \{0, 1\}$, which can take different values for each cell in the confusion matrix. Correct classifications (i.e., upper left and lower right cell) are rewarded with a benefit $b_k = c(l \mid k), l = k$. Incorrect classifications (i.e., lower left and upper right cell) are regarded as costs $c_k = c(l \mid k), l \neq k$. Profit is then computed by offsetting the costs and benefits against each other.

From the confusion matrix, classification performance measures are derived in order to evaluate the discrimination power of the predictive model. Widely accepted measures for binary classification problems in the data mining community are [7, 11, 12, 13, 14, 15]:

$$\mathit{Accuracy}(t) = \pi_0 F_0(t) + \pi_1(1 - F_1(t)), \tag{4}$$

$$\mathit{Error\ rate}(t) = \pi_0(1 - F_0(t)) + \pi_1 F_1(t), \tag{5}$$

$$\mathit{Recall}(t) = F_0(t), \tag{6}$$

$$\mathit{Precision}(t) = \pi_0 F_0(t)/(\pi_0 F_0(t) + \pi_1 F_1(t)), \tag{7}$$

$$F_1\ \mathit{measure}(t) = 2\pi_0 F_0(t)/(\pi_0 + \pi_0 F_0(t) + \pi_1 F_1(t)), \tag{8}$$

$$\mathit{MER} = \min_{\forall t}\{\mathit{Error\ rate}(t)\}, \tag{9}$$

$$\mathit{AUC} = \int_{-\infty}^{+\infty} F_0(s) f_1(s)\, ds. \tag{10}$$

Undoubtedly, the *area under the ROC curve* (AUC) is one of the most popular measures to objectively evaluate the classification performance, since it frees the user from manually specifying the classification threshold. Note that the *minimum error rate* (MER) is also independent of the threshold. Simply said, the AUC can be interpreted as being the probability that a classifier will allocate a lower score to a randomly chosen churner than to a randomly chosen nonchurner [7, 13]. Thus, the higher the AUC, the better the classification performance. However, the listed performance measures do not explicitly take classification costs or benefits into account, making their application for classification problems with high class imbalance such as churn inappropriate.

Moreover, Hand [3] showed that the AUC implicitly treats the relative severities of misclassification differently among classifiers; ergo, a fair model comparison cannot be established. These severities ultimately are properties of the classification problem at hand, and should be independent of the applied classifier. For this reason, Hand proposed the H measure that fixes the distribution of relative severities in order to establish a fair comparison and explicitly account for misclassification costs [3]:

$$H = 1 - \frac{\int Q(t_{opt}(c); a, c) u_{\alpha,\beta}(c)\, dc}{\pi_0 \int_0^{\pi_1} c\, u_{\alpha,\beta}(c)\, dc + \pi_1 \int_{\pi_1}^{1} (1 - c) u_{\alpha,\beta}(c)\, dc}, \tag{11}$$

where $Q(\cdot)$ is the average classification loss, t_{opt} is the optimal classification threshold that minimizes the loss, $c = c_0/a$ with $a = c_0 + c_1$ is the cost ratio, and $u_{\alpha,\beta}(\cdot)$ is a unimodal beta distribution, i.e., its parameters are restricted to be larger than one ($\alpha > 1$, $\beta > 1$). To specify appropriate misclassification costs if class imbalance is present, Hand and Anagnostopoulos [16] suggested to set the parameters of the beta distribution to $\alpha = \pi_0 + 1$ and $\beta = \pi_1 + 1$. A H measure value closer to one indicates superior classification performance.

It is, however, generally recommended to incorporate both costs and benefits into a performance measure [4]. Accordingly, Verbraken et al. [5] proposed the cost benefit analysis framework for customer churn, which incorporates the costs associated with a retention campaign and the benefits of retained customers. This logic is succinctly expressed by the *average classification profit* function [5]:

$$\begin{aligned}\Pi_C(t; \gamma, CLV, \delta, \phi) = {} & CLV(\gamma(1 - \delta) - \phi)\pi_0 F_0(t) \\ & - CLV(\delta + \phi)\pi_1 F_1(t),\end{aligned} \tag{12}$$

where t is the classification threshold and γ is the probability that a targeted would-be churner accepts a special offer and remains a customer. CLV represents the constant customer lifetime value per retained customer (200 €). The two dimensionless parameters $\delta = d/CLV$ and $\phi = f/CLV$ are derived from d, the constant cost of the retention offer (10 €), and f, the constant cost of contact (1 €). Furthermore, it is assumed that these parameters are strictly positive and $CLV > d$. Note also that the values between brackets are the recommended default values for churn management campaigns in the telecommunication sector [5].

The cost benefit framework encompasses a *deterministic* and a *probabilistic* profit-based performance measure. The former is the *maximum profit measure for customer churn* (MPC) [5]:

$$\text{MPC} = \max_{\forall t} \left\{ \Pi_C(t; \gamma, CLV, \delta, \phi) \right\}. \tag{13}$$

The latter assigns a beta distribution to γ, denoted as $h(\gamma)$ with the restriction that its parameters are $\alpha' > 1$ and $\beta' > 1$, which yields the *expected* maximum profit measure for customer churn (EMPC) [5]:

$$\text{EMPC} = \int_\gamma \Pi_C(t_{opt}(\gamma); \gamma, CLV, \delta, \phi) h(\gamma)\, d\gamma, \tag{14}$$

where t_{opt} is the optimal classification threshold that maximizes the profit for given γ. As recommended by [5], α' and β' are by default set to 6 and 14, respectively. Clearly, the probabilistic measure is much richer than the MPC, in the sense that it considers a range of γ values—not just a single value (i.e., the mean of the beta distribution: $\alpha'/(\alpha' + \beta')$). Therefore, we mainly focus on the EMPC. Yet, both profit measures allow identifying the most profitable classifier unambiguously.

Additionally, the proposed cost benefit framework provides the *(expected) profit maximizing fraction for customer churn*, $\bar{\eta}$, which permits practitioners to estimate the optimal fraction of customers to target in the retention campaign for maximum profit. Following the deterministic approach, it becomes [5]:

$$\bar{\eta}_{mpc} = \pi_0 F_0(t_{opt}) + \pi_1 F_1(t_{opt}) \tag{15}$$

with

$$t_{opt} = \arg\max_{\forall t} \left\{ \Pi_C(t; \gamma, CLV, \delta, \phi) \right\}, \tag{16}$$

whereas the profit maximizing fraction derived from the probabilistic approach is defined as [5]:

$$\bar{\eta}_{empc} = \int_\gamma [\pi_0 F_0(t_{opt}(\gamma)) + \pi_1 F_1(t_{opt}(\gamma))] h(\gamma)\, d\gamma. \tag{17}$$

Instead of making an arbitrary choice of taking, for example, the top 10% of predicted would-be churners, which likely results in suboptimal profit ([5, 17]), the $\bar{\eta}$ estimates help to determine how many customers should be targeted in the retention campaign. These estimates are especially appealing in a practical setting.

2.4. Genetic algorithms

Genetic algorithms (GAs) are metaheuristic optimization algorithms inspired by the biological process of evolution to solve complex problems [18, 19, 20, 21, 22, 23]. They are a subclass of evolutionary algorithms (EAs). Our focus is on real-coded genetic algorithms (RGAs), which encode a candidate solution, also called *chromosome*, as a vector of real numbers; unlike the regular GA that encodes a chromosome as a binary string. Note that elements of the chromosome are also called *genes*. Independent from the chosen coding scheme, the fundamental concepts behind GAs remain the same. The most significant advantages of GAs are their global search capabilities as well as their adaptability to a broad spectrum of problems [24]. That is, GAs are capable of obtaining useful results even in circumstances in which traditional techniques fail such as strong nonlinearities, nondifferentiability, noisy and time-varying objective function values, or a large search space with high dimensionality [21].

The key characteristic of GAs is that they are based on a population of chromosomes, performing a *parallel adaptive search* to explore the solution space, that progressively evolves toward the optimum with the aid of *genetic operators* [25]. The evolutionary search is biased by the *fitness function*, which is a mathematical expression that assesses the quality of the chromosomes. By convention, the higher the fitness value of a chromosome, the better its quality. A GA usually consists of at least the following genetic operators: *selection*, *crossover*, and *mutation*. All operators can be specified in numerous ways for different types of chromosome representations from very generic to very problem-specific expressions. However, in their essence, each genetic operator has to fulfill its specific role.

Now, suppose a finite population of chromosomes for a given optimization problem is available, selection then takes a random subset of the population members based on their fitness values, which forms the basis for the creation of the next generation. Selection operators are designed such that high fit chromosomes have a high chance of being selected for reproduction. Once the selection has been made, the remaining generic operators are applied in turn. That is, crossover and mutation are responsible for exchanging and modifying the gene material to create new chromosomes. The population is then updated to the next generation. In general, good operators should promote diversity and simultaneously establish a high *fitness correlation*, meaning that parents with high fitness values should also, on average, produce highly fit offspring [25]. By repeatedly applying these operators, the GA converges toward the optimum, and, given an infinite amount of time, GAs are capable of finding the global optimum in the search space.

Given the capabilities outlined above, it should therefore come as no surprise that GAs can also be applied to find the optimal parameter values for the logistic regression model (1). Although faster algorithms exist to maximize the binomial log-likelihood function (2), applying a GA, in which the collection of regression coefficients acts as a chromosome, to solve the optimization problem is fairly straightforward [26].

2.5. Related Work

A profit-based performance measure also exists for credit risk modeling with the same rationale as for the EMPC [8]. That is, this measure also stems from the general cost benefit framework [5], but its classification costs and benefits are motivated based on expected profits and losses in a credit granting setting. As for the EMPC, the objective is to make a model selection based on profit-driven criteria. Moreover, the *expected maximum profit measure for credit scoring* additionally allows computing the optimal classification threshold, which is crucial for the implementation of the model. Verbraken et al. [8]

benchmarked their approach, and found that their profit-based metric outperforms alternative approaches in both accuracy and monetary value.

An alternative to class-based cost allocation is to measure classification costs at the instance level. This way, individual costs associated with customers can be set, which in turn allows a more sensitive specification of how costly the misclassification of a particular customer is. The classification performance is then measured by the total cost, which is the sum of all individual costs. Such an example-dependent cost-sensitive framework for churn is proposed by [27]. However, unlike in the cost benefit framework of the EMPC, the customer lifetime value is treated as a cost for effectively churned customers—not as a benefit of retained customers. Thus, their framework does not account for any benefits, and is purely cost-based. Their suggested approach substitutes the objective function of the logistic model with an example-dependent cost function, and applies a binary GA to minimize it. Using a real-life data set, the authors concluded that the cost-sensitive approach yields up to 26% higher cost savings than a cost-insensitive one. Additionally, they emphasize the importance of incorporating cost-sensitive measures into the model construction step, because, in this manner, it helps best to improve the classification performance.

The idea of directly including the performance measures into the model construction is also considered by [28]. Here, a classifier, called RIMARC, is proposed that maximizes the AUC directly. In a benchmark study, they show that in about 60% of the cases their proposed classifier significantly outperforms other techniques in terms of the AUC. This suggests that the direct incorporation of the performance measure into the model construction can indeed lead to a superior classification performance.

3. Profit Maximizing Modeling

In this section, we introduce our ProfLogit classification technique, which is based on the logistic model structure (1) but optimizes the regression parameters according to the EMPC, aiming to produce the most profitable classifier. A real-coded genetic algorithm (RGA) is utilized to find the optimal parameter vector that corresponds to a maximum on the EMPC landscape. Note that, given the EMPC definition in (14), it is clear that gradient-based optimization methods are not applicable for the maximization problem. Additionally, a small comparison study revealed that RGA yields a better average EMPC performance than other nature-inspired optimization algorithms such as differential evolution (DE) and particle swarm optimization (PSO) (see Section A). In what follows, we provide detailed explanations of the building blocks of ProfLogit.[1]

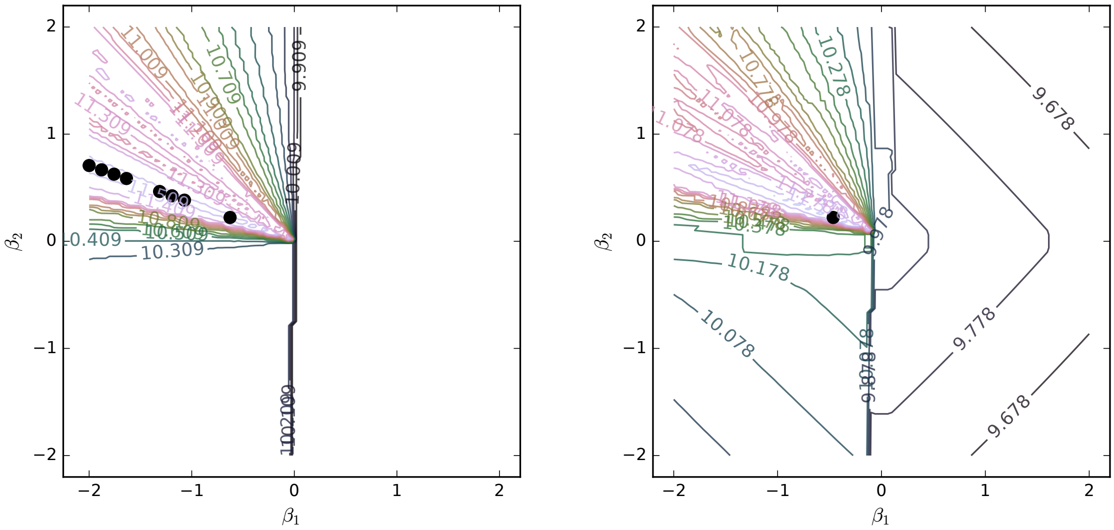
3.1. ProfLogit: Profit Maximizing Logistic Regression

3.1.1. Fitness Function and Soft-thresholding

ProfLogit's objective function is defined by substituting the binomial log-likelihood in (2) with the EMPC measure (14), ultimately yielding a profit-sensitive classification model:

$$Q_{\lambda}^{empc}(\boldsymbol{\theta}) = EMPC(\boldsymbol{\theta}) - \lambda||\boldsymbol{\beta}||_{\ell_1}, \tag{18}$$

[1]A Python 3 implementation of ProfLogit is available from the corresponding author upon request.



(a) Without lasso penalty.

(b) With lasso penalty.

Figure 1: ProfLogit's fitness landscapes computed based on a real-life churn data set. If the fitness function (18) only consists of the EMPC measure (i.e., $\lambda = 0$) as in (a), multiple maxima with identical fitness ($\bullet$) exist. On the other hand, if fitness function (18) is augmented with the lasso penalty ($\lambda > 0$) as in (b), only one maximum ($\bullet$) exists for these data. Additionally, the soft-thresholding operator induces an incline on the fitness surface that makes the evolutionary search of ProfLogit more focused and efficient.

where $\boldsymbol{\theta} = (\beta_0, \boldsymbol{\beta}) \in \mathbb{R}^{p+1}$ is the parameter vector, consisting of the intercept β_0 and the regression coefficients $\boldsymbol{\beta}$ as defined in (1), and the second term is the lasso penalty identical as in (2). Note that the penalty only applies to the regression coefficients in $\boldsymbol{\beta}$. In the RGA, $\boldsymbol{\theta}$ represents a chromosome in which the regression coefficients act as the genes, and (18) is the fitness function that is maximized by the RGA. $\boldsymbol{\theta}$ can also be interpreted as a classification model that outputs churn scores that serve as an input for the EMPC measure to compute the classification profit for the given $\boldsymbol{\theta}$. Given the underlying RGA, ProfLogit thus works with a population of churn models.

In a previous version of ProfLogit [6], the fitness function only contained the EMPC measure (which corresponds to $Q^{empc}_{\lambda=0}(\boldsymbol{\theta})$), yielding in 50% of the cases a better EMPC performance than the standard, unregularized logistic model. In an attempt to identify potential performance boost mechanisms, we conducted an elaborated fitness landscape analysis. We found that a pure EMPC fitness function (i.e., $\lambda = 0$) exhibits *multiple maxima* with identical fitness (Figure 1a), and hence potentially many solutions that have the same fitness value but different parameter values are returned by the RGA. That is because candidate solutions that correspond to these maxima cannot be differentiated by their fitness values anymore, and therefore the RGA becomes indecisive in selecting one solution. Consequently, every time ProfLogit is executed it likely returns a different solution, which entirely depends on the random seed used for the initialization of the population.

Therefore, we augment ProfLogit's fitness function with the lasso penalty to avoid the undesirable behavior of returning "unstable" solutions. Generally, the lasso regularization penalizes model complexity and biases the evolutionary search toward simpler models. When considering an example based on real-life data, the penalty helps to reduce the number of maxima from many to one (Figure 1b). Note that we do not claim that the inclusion of the lasso penalty generally results in a unique maximum. In this example, the effect of the augmented fitness function is clearly visible, i.e., (β_1, β_2)-pairs far away from $(0, 0)$ have a lower fitness. The inclusion of the penalty term creates an incline on the fitness surface that noticeably helps the RGA to find the maximum more efficiently. De Jong [25] refers to the build-in of such an incline as making the fitness function "evolution friendly," meaning that the fitness function has to be designed such that it provides clues for the evolutionary search of where to find solutions with high fitness.

Additionally, the *soft-thresholding operator*, $S_\lambda(\cdot)$, is utilized in order to penalize individual regression coefficients in $\boldsymbol{\beta}$, which in turn introduces a biasing of the coefficient values toward zero. This useful mechanism is also employed in the regular lasso-penalized logistic model [10], and has been popularized by [29]. The soft-thresholding operator is defined as follows [10, 30]:

$$S_\lambda(\beta) = \text{sign}(\beta)(|\beta| - \lambda)_+. \tag{19}$$

It sets the individual coefficient β to zero if its absolute value is smaller or equal to the regularization parameter, and otherwise pulls β toward zero by the magnitude of λ. Unlike for the lasso model, ProfLogit has no theoretical justification for the usage of the soft-thresholding operator. However, $S_\lambda(\cdot)$'s property of zero-biasing promotes the return of less complex models, and ultimately simpler churn models are preferred since they are more likely to perform better on new, unseen data. Additionally, the shrinkage toward zero has also the advantage that the search space boundaries for the RGA can be set closer to zero, which keeps the search space itself small. An empirically based justification for the application of the soft-thresholding operator is provided in Section 3.2.

3.1.2. Initialization of the Parameter Vector Population

To apply the RGA, a population of parameter vectors has to be first created. Let $\mathcal{P}_g$ be the collection of parameter vectors $\boldsymbol{\theta}$, as defined above, representing the g^{th} generation of the population for $g = 0, \ldots, G$, where $\mathcal{P}_0$ is the initial population. Note that the population size is held constant in ProfLogit throughout the entire evolutionary search (i.e., $\forall g \colon |\mathcal{P}_g| = |\mathcal{P}|$). Similarly, the length of each parameter vector, $|\boldsymbol{\theta}|$, is fixed. To initialize a parameter vector, a random number from a uniform distribution is assigned to each $\beta_j \in \boldsymbol{\theta}$, where β_j corresponds to the j^{th} predictor (except β_0, which is the intercept):

$$\beta_j \leftarrow \text{Unif}(L_j, U_j), \tag{20}$$

with L_j and U_j being the lower and upper boundary of the search space for β_j, respectively. By default, L_j is set to -3 and U_j is set to $3, \forall j$. This way, candidate solutions are created that are randomly scattered over the search space. The coverage density of the space can somewhat be influenced by the population size. Note that the population-based approach provides a natural mechanism of a parallel adaptive search [25]. That is, the larger the size of the population, the more the RGA explores the search space in parallel. However, this comes with a trade-off. The larger the population is, the more computationally expensive it becomes to carrying out the calculations. Depending on the optimization problem, EA designers typically start with small populations, and increase the size if necessary. Even with a small population size, EAs still can attain acceptable results.

Once $\mathcal{P}_0$ is created, each $\boldsymbol{\theta}$ first undergoes soft-thresholding (19), and then its fitness is evaluated using (18). Soft-thresholding is only applied on the regression coefficients in the $\boldsymbol{\beta}$ vector—not on the intercept β_0. Note also that the soft-thresholding operator is only applied *once* on all $\beta \in \boldsymbol{\beta}$ just before the fitness evaluation. When the fitness values are available, copies of the fittest $\boldsymbol{\theta}$s are put into a so-called *elite pool*, which is part of a mechanism called *elitism* that is explained in more detail in Section 3.1.6.

3.1.3. Probabilistic Selection of the Fittest

A selection $\mathcal{S}_g \subseteq \mathcal{P}_g$ (with replacement) is conducted such that $\boldsymbol{\theta}$s with high fitness are more likely to be selected for reproduction than $\boldsymbol{\theta}$s with low fitness. Note that the size of the selection equals the size of the population, i.e., $|\mathcal{S}_g| = |\mathcal{P}_g| = |\mathcal{P}|$, and it remains constant in each processing step that follows. Note also that, in some literature (e.g., [19]), $\mathcal{S}_g$ is referred to as the *mating pool*. Various strategies exist to carry out a selection [18, 19]. ProfLogit applies a strategy called *linear scaling*, which is based on the idea of *roulette wheel selection* (RWS)—a probabilistic approach. That is, the selection of a parameter vector is proportional to its fitness value: the higher the fitness value, the higher the chance of being selected. Unfortunately, applying RWS on the raw fitness values will bias the evolutionary search too much toward the fittest population member, which likely causes that the RGA gets trapped in a local optimum. To avoid this bias, the fitness values are scaled in a way that promotes *exploration* in the early stage and *exploitation* in the later stage of the evolutionary search. Let $f^{(q)}$ be the fitness value of the population member $\boldsymbol{\theta}^{(q)}$ for $q = 1, \ldots, |\mathcal{P}|$. The fitness values are then scaled as follows:

$$f_s^{(q)} = af^{(q)} + b, \tag{21}$$

where $f_s^{(q)}$ is the scaled fitness value with a and b being the scaling parameters. The way how a and b are specified has a great influence on the performance of the RGA; we follow the method described in [31]. This scaling scheme makes sure that the average fitness value does not change its value after scaling, i.e., $\bar{f}_s = \bar{f}$. Furthermore, let f_{min} and f_{max} be the original minimum and maximum fitness value, respectively. If $f_{min} > (C\bar{f} - f_{max})/(C-1)$, a and b are specified such that the scaled maximum fitness value becomes C times as large as $\bar{f}_s$:

$$f_s^{(q)} = \frac{\bar{f}(C-1)}{f_{max} - \bar{f}}\, f^{(q)} + \frac{\bar{f}(f_{max} - C\bar{f})}{f_{max} - \bar{f}}, \tag{22}$$

where C controls the dominance of the fittest $\boldsymbol{\theta}$. C is commonly set to two, ensuring that the fittest population member is not selected too often. In case the above condition is not true, a and b become:

$$f_s^{(q)} = \frac{\bar{f}}{\bar{f} - f_{min}}\, f^{(q)} - \frac{\bar{f} \times f_{min}}{\bar{f} - f_{min}}. \tag{23}$$

Proportional selection schemes require that all fitness values are positive, and (23) makes sure that the scaled fitness values do not become negative. This, however, comes at a price that the parameter vector with the corresponding scaled minimum fitness value has no chance of being selected (i.e., its scaled fitness value is zero). After the scaling, the selection probability is computed as follows:

$$p^{(q)} = \frac{f_s^{(q)}}{\sum_{q=1}^{|\mathcal{P}|} f_s^{(q)}}. \tag{24}$$

Clearly, the higher the quantity $p^{(q)}$ of the corresponding parameter vector, the more likely it gets selected. Now that a selection of relatively high fitness parameter vectors $\mathcal{S}_g$ is available, the next step is to generate new candidate solutions.

3.1.4. Crossover for Information Exchange

The crossover operator manipulates the elements of the selected parameter vectors, aiming to produce new candidate solutions with higher fitness quality. There are again various strategies available to perform a crossover, which vary heavily between representation types. ProfLogit employs a *local arithmetic crossover* that is applicable on real-encoded chromosomes. It first randomly picks (without replacement) two parameter vectors from the selection $\mathcal{S}_g$ (obtained from the previous step), which are referred to as parents, e.g., $\boldsymbol{\theta}_{parent}^{(1)}$ and $\boldsymbol{\theta}_{parent}^{(2)}$. Next, a sample of random numbers, denoted as $\mathbf{w}$, is taken from the standard uniform distribution, Unif$(0,1)$, where the sample size equals the vector length $|\boldsymbol{\theta}|$. Parental gene material is then exchanged as follows [19]:

$$\begin{aligned}\boldsymbol{\theta}_{child}^{(1)} &= \mathbf{w} * \boldsymbol{\theta}_{parent}^{(1)} + (\mathbf{1} - \mathbf{w}) * \boldsymbol{\theta}_{parent}^{(2)} \\ \boldsymbol{\theta}_{child}^{(2)} &= \mathbf{w} * \boldsymbol{\theta}_{parent}^{(2)} + (\mathbf{1} - \mathbf{w}) * \boldsymbol{\theta}_{parent}^{(1)}\end{aligned} \tag{25}$$

where $\boldsymbol{\theta}_{child}^{(\cdot)}$ are the newly created parameter vectors, $\mathbf{1}$ is the all-ones vector, and $*$ symbolizes element-wise multiplication. The elements in $\mathbf{w}$ can also be regarded as weights, indicating how much gene material is inherited from which parent. For example, if the first element is $w_1 = 0.7$, this means $\boldsymbol{\theta}_{child}^{(1)}$'s first gene inherits 70% from $\boldsymbol{\theta}_{parent}^{(1)}$ and 30% from $\boldsymbol{\theta}_{parent}^{(2)}$; whereas the first gene of $\boldsymbol{\theta}_{child}^{(2)}$ inherits 30% from $\boldsymbol{\theta}_{parent}^{(1)}$ and 70% from $\boldsymbol{\theta}_{parent}^{(2)}$. Not all parameter vectors in $\mathcal{S}_g$ experience a crossover, i.e., the operator is, by default, applied with a probability of $p_c = 0.8$, which is invariant over generations. More specifically, a crossover between two randomly chosen parents is performed if $u < p_c$, where u is drawn from Unif$(0,1)$. When a crossover is performed, the children take the positions of their parents in $\mathcal{S}_g$. The processed mating pool after the crossover is denoted as $\mathcal{S}'_g$.

3.1.5. Mutation to Discover New Solutions

The mutation operator aims to create candidate solutions that are unlikely to be produced by information exchange alone, stimulating the exploration of the search space. As for most of the genetic operators, there are also many strategies for mutation available. After the crossover has been performed, ProfLogit applies a *uniform random mutation* with a fixed default probability of $p_m = 0.1$. Typically, the mutation rate is set to a low value in order to avoid too much disturbance in the late exploitation phase. Similar as with the crossover, a mutation is performed if $u < p_m$, where u is drawn from Unif$(0,1)$. In this case, a parameter vector $\boldsymbol{\theta}$ is randomly picked (without replacement) from $\mathcal{S}'_g$, and a new value is assigned to a randomly selected $\beta \in \boldsymbol{\theta}$. The new value is obtained from the same uniform distribution as used for the initialization (Eq. (20)). Also here, the mutated parameter vector replaces its predecessor in $\mathcal{S}'_g$. The processed mating pool, now denoted as $\mathcal{S}''_g$, forms, in principle, the next generation. The fitness of all new parameter vectors in $\mathcal{S}''_g$ is evaluated, which have been created by either crossover, mutation, or underwent both operators. Like in the initialization, first the soft-thresholding operator (19) is applied on the $\boldsymbol{\beta}$ part of $\boldsymbol{\theta}$, then the fitness is evaluated using (18).

3.1.6. Update to the Next Generation

After the genetic operators have been applied, the current population $\mathcal{P}_g$ is updated through $\mathcal{S}_g''$ to create the next generation of candidate solutions $\mathcal{P}_{g+1}$. In this way, the RGA converges toward the maximum from generation to generation. There are several update strategies, but they can mainly be categorized into *overlapping*- and *nonoverlapping*-generation models [25]. ProfLogit utilizes a mechanism called elitism, which belongs to the former category. As foreshadowed in the initialization step, in every generation, a proportion of the fittest $\boldsymbol{\theta}$s is put aside in an elite pool, denoted as $\mathcal{E}$, that will survive to the next generation. When the population is updated from $\mathcal{P}_g$ to $\mathcal{P}_{g+1}$, parameter vectors in $\mathcal{S}_g''$ have to compete for their survival against the members in $\mathcal{E}$. That is, parameter vectors in $\mathcal{E}$ replace the $\boldsymbol{\theta}$s with the lowest fitness in $\mathcal{S}_g''$, which then finally becomes the next population $\mathcal{P}_{g+1}$. Having now the next generation of parameter vectors, the elite pool is updated with the new, fittest $\boldsymbol{\theta}$s as well. Elitism induces increased selection pressure, and $\boldsymbol{\theta}$s have to compete for their survival across generations. By default, ProfLogit keeps at least one or at most 5% of the population size in the elite pool, i.e., $|\mathcal{E}| = \max\{1, 0.05 \times |\mathcal{P}|\}$. Note that also the size of the elite pool remains fixed over generations. An advantage of using elitism is that the RGA keeps track of the best solution(s) obtained so far. Consequently, the best fitness value does not decreases with generations. A *best-so-far fitness curve* then refers to a line plot in which the best fitness values are plotted against the generation index g, representing a monotonically nondecreasing function.

3.1.7. Termination: Returning a Solution

By repeatedly applying the genetic operators, the RGA progressively explores the search space—converging to the maximum—until a satisfied termination criterion stops the evolutionary search. Reaching a prespecified maximum number of generations, G, is a common termination criterion. Another fairly standard criterion is to terminate the search if the best fitness value has not improved for a predetermined number of generations. If one termination criterion is met, the parameter vector with the highest fitness value, $\widehat{\boldsymbol{\theta}}$, in $\mathcal{P}_g$ is returned, which represents the final solution. An overview of the ProfLogit algorithm to construct a profit maximizing churn model is given in Algorithm 1.

3.2. Motivation for Subjective Choices

We integrate the soft-thresholding operator (19) into ProfLogit based on subjective grounds, and wish to motivate our choice in this subsection. More specifically, we empirically show that soft-thresholding is more efficient for finding the optimum in less generations, helps to reduce variability, and effectively sets regression coefficients with no predictive power to zero. Note that the last argument ultimately implies that ProfLogit performs a feature selection in a profit maximizing manner. Yet, before elaborating on the soft-thresholding operator, we discuss the hyperparameter tuning of λ in (18).

3.2.1. Tuning the Regularization Parameter

To infer the optimal regularization parameter value, λ_{opt}, we generate a grid of λ values as follows:

$$\Lambda = \{\lambda \mid \lambda_{min} < \lambda < \lambda_{max}\}, \tag{26}$$

Algorithm 1 ProfLogit: Profit Maximizing Evolutionary Logistic Model with Lasso Regularization

Inputs: Churn data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with p predictors that have zero mean $\frac{1}{N}\sum_{i=1}^N x_{ij} = 0$ and unit variance $\frac{1}{N}\sum_{i=1}^N x_{ij}^2 = 1$;

λ, regularization parameter;

$|\mathcal{P}|$, size of the population of parameter vectors $\boldsymbol{\theta} = (\beta_0, \boldsymbol{\beta}) \in \mathbb{R}^{p+1}$;

$|\mathcal{E}|$, size of the elite pool (default: $\max\{1, 0.05 \times |\mathcal{P}|\}$);

L_j, lower search boundary for β_j (default: $-3, \forall j$);

U_j, upper search boundary for β_j (default: $3, \forall j$);

p_c, crossover probability (default: 0.8);

p_m, mutation probability (default: 0.1);

G, maximum number of generations

{**Initialization of the Real-coded Genetic Algorithm (RGA)**}

Initialize: $g \leftarrow 0$ `# generation index`

Create initial $\boldsymbol{\theta}$ population, $\mathcal{P}_g$, of size $|\mathcal{P}|$ according to (20)

Evaluate: $\forall \boldsymbol{\theta} \in \mathcal{P}_g$ apply soft-thresholding on all $\beta \in \boldsymbol{\beta}$ (not on the intercept β_0) as in (19) and compute the lasso-regularized EMPC fitness as in (18) with given λ; copy the fittest $\boldsymbol{\theta}$s into the elite pool $\mathcal{E}$

{**Main Loop of the RGA**}

while $g \leq G$ and the fitness (18) improves **do**

Select: $\boldsymbol{\theta}$s in $\mathcal{P}_g$ into $\mathcal{S}_g$ based on fitness values processed according to (22)-(24)

Crossover: $\boldsymbol{\theta}$s $\in \mathcal{S}_g$ as in (25) with probability p_c; processed $\mathcal{S}_g$ is denoted as $\mathcal{S}'_g$

Mutate: $\boldsymbol{\theta}$s $\in \mathcal{S}'_g$ as described in Section 3.1.5 with probability p_m; processed $\mathcal{S}'_g$ is denoted as $\mathcal{S}''_g$

Evaluate: the fitness of newly created $\boldsymbol{\theta}$s $\in \mathcal{S}''_g$ as in the initialization, i.e., applying soft-thresholding on all $\beta \in \boldsymbol{\beta}$ before fitness evaluation

Update: $\mathcal{P}_g$ to $\mathcal{P}_{g+1}$ by processing $\mathcal{S}''_g$ and $\mathcal{E}$ as in Section 3.1.6; update the elite pool $\mathcal{E}$ with the fittest $\boldsymbol{\theta}$s;

$g \leftarrow g + 1$

end while

return $\widehat{\boldsymbol{\theta}}$, the parameter vector with the highest fitness in $\mathcal{P}_g$

with $\lambda_{max} = \max_j \left|\frac{1}{N}\langle \mathbf{x}_j, \mathbf{y}\rangle\right|$ and $\lambda_{min} = \epsilon\lambda_{max},\ \epsilon > 0$. [10, 30]. Given a standardized data set $\mathcal{D}$ as described in Section 2.2, a value for the regularization parameter can be determined (i.e., λ_{max}) that is just large enough so that the lasso penalty sets all regression parameters to zero [10, 30]. Any value below λ_{max} relaxes the penalization and the coefficient values start to become nonzero. In other words, predictors with high predictive power obtain first a nonzero value. Values above λ_{max} do not have any effect, since all coefficients are already zero. Going all the way down to $\lambda = 0$ is also illogical, because then the penalization vanishes and the problem of multiple maxima as discussed in Section 3.1 reappears. For this reason, λ_{min} should also not be set too close to zero, and we therefore specify $\epsilon = 0.1$. For the experiments in Section 4, the grid consists of $|\Lambda| = 15$ equidistant values between λ_{min} and λ_{max}, and λ_{opt} then corresponds to the $\lambda \in \Lambda$ with the highest EMPC performance (14) on the validation set. Note that $|\Lambda| = 15$ is an arbitrary choice, but it should be large enough to create a dense grid.

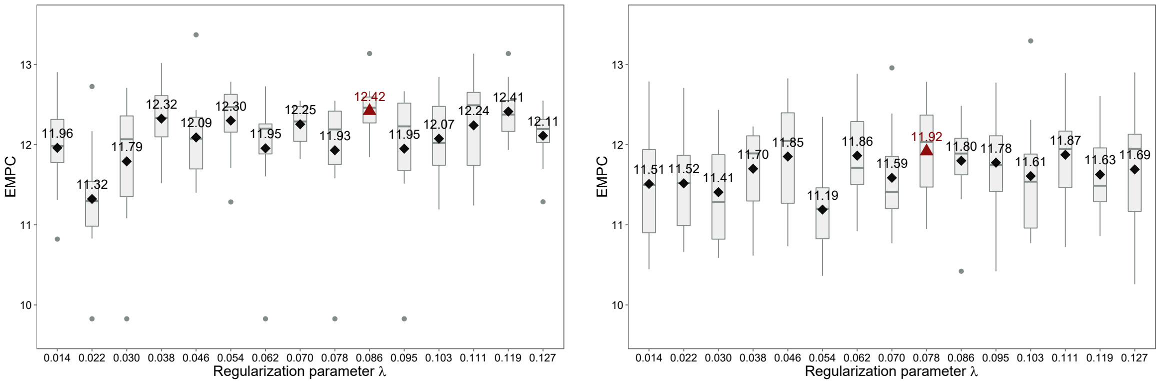
For each $\lambda \in \Lambda$, a reliable performance estimate has to be obtained in order to confidently determine

λ_{opt}. Yet, the underlying RGA of ProfLogit is *stochastic* in nature, which requires that the analysis has to be repeated several times to obtain average performance estimates. Thus, we split the original data set into a training, validation, and test set, which are stratified according to the churn indicator. Note that the test set is sometimes also referred to as *hold-out sample* (we will use both terms interchangeably) and it is independent from the training and validation set. The hold-out sample will *only* be used to evaluate the final model performance, which in turn provides so-called *out-of-sample estimates.* Next, ProfLogit is trained R times with a given λ on the training set, and its EMPC performance is measured on the validation set. Next, the average of the R estimates becomes the associated performance estimate for the given λ. This procedure is performed for all $\lambda \in \Lambda$, and λ_{opt} corresponds to the λ value with the highest average EMPC performance on the validation set. To determine the final classification performance, ProfLogit is trained M times with λ_{opt} on the union of the training and validation set, and its final performance is evaluated based on the hold-out sample, which has previously not been used for either training or finding λ_{opt}. According to this procedure, the tuning of the regularization parameter and the final estimation of ProfLogit's classification performance requires in total the construction of $|\Lambda| \times R + M$ models. For the experiments in Section 4, we decided that $R = 10$ and $M = 30$ are sufficiently large values, and we tune and train ProfLogit in the exact same way as described in this paragraph. For the sake of performing the entire analysis in a reasonable amount of time, we dismiss resampling techniques such as cross-validation for ProfLogit and perform the analysis only on stratified data splits. Note that the approach explained in this section only applies to ProfLogit, we still will use cross-validation on the union of the training and validation set for the competitive classification techniques. Thus, we deliberately make the comparison for ProfLogit more challenging.

3.2.2. Soft-thresholding to Improve Performance

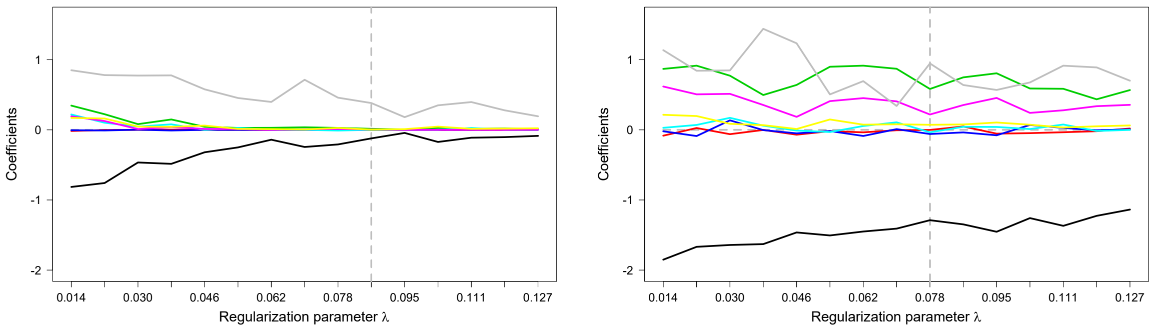
Unlike in the regular lasso model where the soft-thresholding operator comes naturally into play, we imposed the $S_\lambda(\cdot)$ operator (19) onto ProfLogit in order to benefit from its properties such as denoisation [29]. Given that $S_\lambda(\cdot)$ is introduced based on subjective grounds, we studied its application in ProfLogit and present here the empirical results for one real-life churn data set.

Generally, the incorporation of the soft-thresholding operator into ProfLogit is highly beneficial in terms of the EMPC, run time, and convergence of the RGA. It increases the average EMPC performance on the training and validation set (Figure 2 and Figure 4). It shrinks the regression coefficients toward zero, effectively performing a feature selection optimized according to the EMPC criterion (Figure 3 and Table 2). It helps to find the maximum quicker, as well as having a clearer average convergence pattern (Figure 4). The most important, however, is that the $S_\lambda(\cdot)$ operator also tremendously helps to improve the EMPC performance on the independent test set (Figure 5). On average, a $12.31 - 10.99 = 1.31$ € EMPC improvement is achieved with $S_\lambda(\cdot)$ compared to the ProfLogit approach without soft-thresholding. Such an improvement is immense for a telephone service provider with thousands or millions of customers. The most straightforward explanation for the significant improvement is that ProfLogit with the soft-thresholding operator yields less complex models, which, in turn, have a better generalization performance. Hence, a higher EMPC is achieved than without soft-thresholding. Even



(a) With soft-thresholding. (b) Without soft-thresholding.

Figure 2: Average EMPC performance (⧫) for each $\lambda \in \Lambda$ as defined in (26) computed based on the validation set (a) with and (b) without the soft-thresholding operator (19), where λ_{opt} is marked as ▲. Each box plot is constructed based on $R = 10$ values. Except when $\lambda = 0.022$, applying soft-thresholding results, on average, in a higher EMPC performance.



(a) With soft-thresholding. (b) Without soft-thresholding.

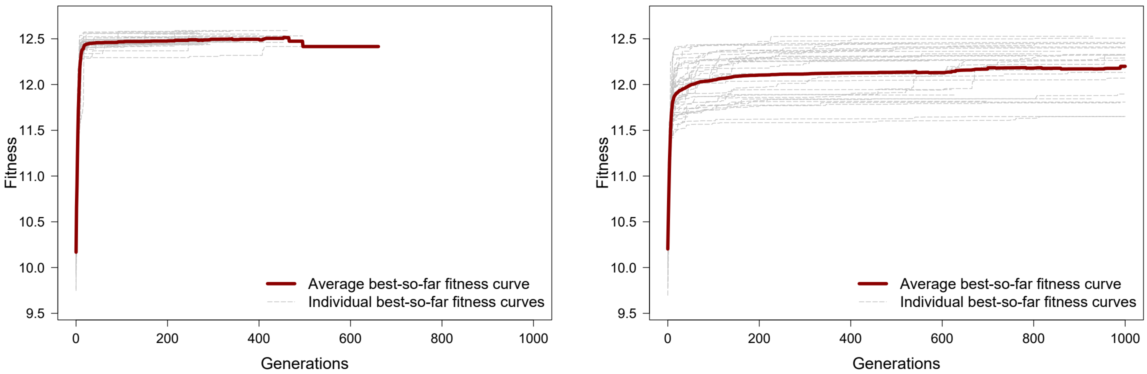
Figure 3: The shrinkage effect is clearly evident (a) with $S_\lambda(\cdot)$ compared to (b) without $S_\lambda(\cdot)$, setting predictors with low predictive power effectively to zero. Coefficient paths are computed based on a real-life churn data set with $p = 8$ predictors. Overall, it suggests that only two out of eight predictors have a particularly strong predictive power relevant for maximum profit. Note that the plotted coefficient values for each λ are averages from the $R = 10$ ProfLogit models. Note also that λ_{opt} equals 0.086 in (a) and 0.078 in (b) as indicated by the dashed line (- -).

when doing the comparison with the same λ_{opt}, the results remain almost identical as presented here. To summarize, empirical evidence clearly encourages the inclusion of the soft-thresholding operator.

3.3. Performance Measures based on the Expected Profit Maximizing Fraction for Customer Churn

In addition, we introduce three performance measures that are byproducts of the expected profit maximizing fraction (17). These measures are based on the notion of precision, recall, and the F_1 measure, but, unlike the measures presented in Section 2.3, they are independent of the classification threshold t.

Due to scarce resources, marketers can only focus on a fraction of the customer base in a churn management campaign. Therefore, they often wish to receive a *lead list* with the top would-be churners so that they know who to target in the campaign. To generate such a list, a subset of the customer base is taken based on the churn scores produced by the predictive model. According to our definitions in Section 2, customers with lower churn scores (i.e., higher likelihood of churning) are more likely to be



(a) With soft-thresholding.

(b) Without soft-thresholding.

Figure 4: With soft-thresholding (a), ProfLogit ($\lambda = \lambda_{opt}$) reaches a higher average EMPC performance trained on the union of the training and validation set than without (b). Furthermore, compared to (b), there is less variability in the best-so-far fitness curves between the $M = 30$ ProfLogit models in (a), indicating that the RGA converges more consistently to the maximum. Additionally, soft-thresholding enables the RGA to find the maximum quicker (i.e., it has a sharper elbow), and therefore requires less generations. Note that ProfLogit is trained with $\lambda_{opt} = 0.086$ in (a) and $\lambda_{opt} = 0.078$ in (b).

Table 2: Soft-thresholding: ProfLogit's Average Coefficient Estimates

	With $S_{\lambda_{opt}}(\cdot)$	Without $S_{\lambda_{opt}}(\cdot)$
Coefficients	**Estimate (SE)**	**Estimate (SE)**
β_0	0.108 (0.631)	0.138 (0.834)
β_1	—	0.003 (0.093)
β_2	−0.107 (0.089)	−1.379 (0.236)
β_3	0.016 (0.020)	0.495 (0.243)
β_4	0.404 (0.240)	1.126 (0.494)
β_5	0.000 (0.001)	0.264 (0.217)
β_6	0.014 (0.026)	0.046 (0.096)
β_7	—	0.029 (0.071)
β_8	—	−0.035 (0.089)

Applying the soft-thresholding operator (19), with the respective λ_{opt} value, biases the regression coefficients toward zero. This sets predictors with low predictive power effectively to zero, and implicitly performs an EMPC-based feature selection. A '—' cell means that the corresponding β_j have a zero coefficient value in all $M = 30$ ProfLogit models. The standard deviations (**SE**) are also overall smaller with soft-thresholding, indicating more precise estimates.

included in the list. However, manually setting the optimal length of the list is by no means obvious. Fortunately, the $\bar{\eta}_{empc}$ as defined in (17) helps us to exactly determine how many customers to target in the retention campaign for maximum profit. Hence, subjective subsetting is avoided.

Let B be the set of all customers, the customer base, which consists of two disjoint sets: would-be churners C and nonchurners $\bar{C}$, i.e., $B = C \cup \bar{C}$ and $C \cap \bar{C} = \varnothing$. For clarity, C is the set of current customers that intend to leave the company soon, and therefore as many as possible of these customers

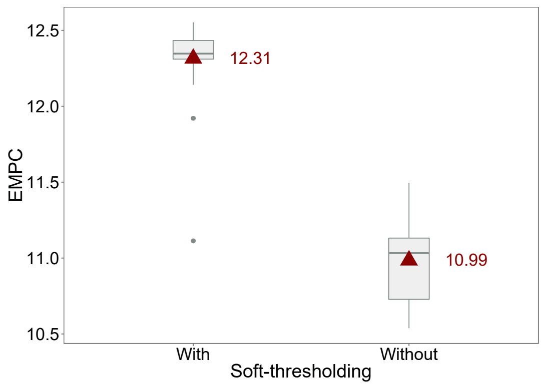


Figure 5: Applying soft-thresholding in ProfLogit significantly impacts the out-of-sample EMPC performance on the hold-out sample, which was completely independent from the entire model construction process. More specifically, with soft-thresholding, ProfLogit has a substantially higher average EMPC performance (▲) than without soft-thresholding. Note that in both scenarios ProfLogit has been trained with the respective λ_{opt}.

should be identified by the churn model for the campaign. Once the $\bar{\eta}_{empc}$ estimate of the churn model is available, the lead list $L \subseteq B$ can be generated by sorting B based on the churn scores from low to high and taking the top $\bar{\eta}_{empc}$ fraction of the predicted would-be churners. It is important to note that this way the length of the lead list is completely generated on objective grounds. The list, however, likely contains subsets of both would-be churners and nonchurners, i.e., $L = C_L \cup \bar{C}_L$, where $C_L \subseteq C$ and $\bar{C}_L \subseteq \bar{C}$. Once having L, we can compute popular performance measures.

Definition 1. *The $\bar{\eta}$-precision or hit rate for customer churn, $\bar{\eta}_p$, is the proportion of correct identifications of churners in the $\bar{\eta}_{empc}$-based lead list L:*

$$\bar{\eta}_p = |C_L|/|C_L \cup \bar{C}_L|. \tag{27}$$

Definition 2. *The $\bar{\eta}$-recall for customer churn, $\bar{\eta}_r$, is the proportion of churners that is included in the $\bar{\eta}_{empc}$-based lead list L:*

$$\bar{\eta}_r = |C_L|/|C|. \tag{28}$$

Frequently, precision and recall are summarized into the F_1 measure, which represents a compromise between the two performance measures. The $\bar{\eta}$-based F_1 measure for customer churn, $\bar{\eta}_{F_1}$, is defined as:

$$\bar{\eta}_{F_1} = 2\frac{\bar{\eta}_p \bar{\eta}_r}{\bar{\eta}_p + \bar{\eta}_r}. \tag{29}$$

For all three measures, it applies that higher values indicate higher effectiveness of the lead list, and ultimately better performance of the classifier. However, these are accuracy-based auxiliary measures only, which intend to free the user from manually setting the classification threshold. Since we aim for maximum profit, the model selection should ultimately be done based on the EMPC. Nevertheless, it might be beneficial to compare hit rates ($\bar{\eta}_p$) among classifiers in order to assess their effectiveness of correctly identifying churners.

4. Empirical Evaluation

In this section, we assess ProfLogit's churn classification performance by benchmarking it against other linear classifiers. To do so, we apply the classification techniques to nine real-life churn data sets from various telecommunication service providers, and evaluate them using the EMPC, MPC, the H measure, AUC, MER, $\bar{\eta}_p$, $\bar{\eta}_r$, and $\bar{\eta}_{F_1}$, as defined in Section 2.3 and Section 3.3. These data sets are either publicly available or have been provided to our research group from various telco operators, located around the world (i.e., North and South America, East Asia, and Europe). Since ProfLogit itself is a linear classifier, we only compare it to linear classification techniques in order to have an equitable comparison. For this reason, we do not consider models such as decision tree, random forest, neural network, support vector machine with nonlinear kernels, and so forth. In total, we compare ProfLogit to eight competitive classifiers: regular logistic regression model (Logistic) as well as the lasso-regularized (Lasso) and the ridge-regularized (Ridge) version of it, and a logistic model with elastic net penalty (ElasticNet) [10]. The remaining ones are linear discriminant analysis (LDA), support vector machine with linear kernel (Linear SVM), stepwise logistic regression (stepLogistic), and backward elimination LDA (backLDA).

We begin by describing the applied data preparation steps and the experimental setup, followed by presenting the results of the benchmark study, a sensitivity analysis of ProfLogit's crossover and mutation rates, and a discussion of the results in the next four subsections, respectively.

4.1. Experimental Setup of the Benchmark Study

For each data set, we fit each classifier on a training set (if applicable, hyperparameters are tuned) and evaluate it on an independent test set (or hold-out sample) to obtain *out-of-sample* classification performance estimates. To find the optimal hyperparameter values, we apply 10-fold cross-validation for the competitive techniques and the procedure discussed in Section 3.2.1 for ProfLogit on the training set. Unless training and test sets are already available from the source, we create them by randomly partitioning the data set into a 70% training and 30% test set, stratified according to the churn indicator to obtain similar churn distributions in the training and test set as observed in the original data set.

We standardize the predictors as described in Section 2.2. Standardization causes that estimated regression coefficient values are expected to be relatively close to zero. This allows us to set small search boundaries for the RGA. Hence, we use the respective default values of -3 and 3 for L_j and U_j in Eq. (20) for $j = 0, \ldots, p$. These default values should be wide enough to cover the essential area of the fitness landscape, and be narrow enough to run the RGA efficiently.

To avoid complex transformations of categorical predictors, we remove those that have more than five categories. The motivation behind is to have a clean approach across data sets and classifiers. The validity of the benchmark study is thereby not jeopardized, since we still provide the same data to all classification techniques. For the included categorical features, we apply dummy coding.

We also remove predictors that are highly correlated with each other to avoid (multi)collinearity problems. That is, the presence of extreme correlations can cause degeneration and wild behavior of the lasso [30]. Predictors that exhibit a near-zero variance are removed as well. Furthermore, observations

Table 3: Real-Life Churn Data Sets

ID	Source	# Predictors	# Observations		Churn rate [%]	
			Training Sample	Hold-out Sample	Training Sample	Hold-out Sample
D1	Duke	11	8,750	3,749	39.31	39.32
KDD	KDD Cup 2009 [a]	8	32,853	14,080	6.98	6.98
O1	Operator	37	4,940	2,116	29.15	29.11
O2	Operator	8	623	266	31.14	31.20
O3	Operator	11	9,522	4,079	22.59	22.58
O4	Operator	9	2,589	1,109	13.29	13.26
O5	Operator	21	40,678	17,433	13.96	13.96
O6	Operator	21	39,404	16,887	11.18	11.17
UCI	UCI [b]	11	3,333	1,667	14.49	13.44

[a] `http://www.kdd.org/kdd-cup/view/kdd-cup-2009`

[b] `http://www.sgi.com/tech/mlc/db`

Characteristics of real-life churn data sets from various telecommunication service providers after preprocessing. The training sample is used to tune and fit the classification models, whereas the hold-out sample serves to estimate the *out-of-sample classification performance*. It is important to note that the hold-out sample is *never* involved in the training of a model.

with missing values are excluded from the analysis. Fortunately, the relative frequencies of missing values are low for most of the data sets included in our setup.

Telephone service providers often log the number of calls and the duration of a call that ultimately allows them to study customers' call behavior. The frequency of calls and the call duration are however interrelated, yet call duration provides nevertheless richer information. To illustrate this, assume a client has the same numbers of calls to a churner, a former customer, and a nonchurner, a current customer. Then, specific call duration patterns (i.e., short versus long calls) provide more insights into the customer's churn behavior than the number of calls. In particular, when considering the average call duration, the number of calls is intrinsically taken into account to some extent. For this reason, we remove features related to the number of calls. This has the additional benefit of reducing the dimensionality, which decreases complexity and execution time of all classifiers.

To keep the analysis of the nine data sets manageable, we only consider main effects. Note that attempts of including all two-way interactions often resulted in nonconvergence of some algorithms, and selective inclusion of two-way interactions was not straightforward. The exclusion of interactions of any degree however does not adversely affect the validity of the study, since all classifiers are trained based on the same input data. Table 3 provides an overview of the data set characteristics after the preprocessing steps described above have been applied.

Moreover, we apply EMPC's default values as specified in Section 2.3. Regarding the parameters for the underlying RGA of ProfLogit, we set the population size equal to the parameter vector length multiplied by a factor of ten, i.e., $|\mathcal{P}| = 10|\boldsymbol{\theta}| = 10(p+1)$. In this way, the population size becomes a linear function of the input dimensions. This allows ProfLogit to better adapt to the classification problem than fixing the population size across all data sets a priori. Making the population size dependent on the input dimensions is reasonable, because the size of the search space grows exponentially with increased dimensions. Logically, it therefore makes sense to increase the size of the population as well, and thereby improve the parallel adaptive search capabilities of the RGA.

Finally, we specify the termination criteria of the evolutionary search as follows: terminate (i) if the number of generations has reached $G = 1{,}000$, or (ii) if the best-so-far fitness value has not improved for 250 generations. To optimize the regularization parameter of ProfLogit, we apply the strategy discussed in Section 3.2.1 to find λ_{opt}. Because of the random nature of the underlying RGA, we run $M = 30$ repetitions of ProfLogit with $\lambda = \lambda_{opt}$ on the training set and evaluate the average out-of-sample classification performance on the hold-out sample. For the competitive classifiers, we apply 10-fold cross-validation in which the EMPC measure is used as criterion to set the optimal hyperparameter values. Using the same criterion for the selection of optimal hyperparameter values as for ProfLogit allows a more appropriate model comparison.

4.2. Results of the Experiment

According to the EMPC, ProfLogit is overall the most profitable churn model (Figure 6 and Figure 7). Occupying the first place in six out of nine data sets results in an average rank of 1.67 on a scale from 1 to 9. In the remaining three data sets, it closely falls behind the respective best competitive technique. In five out of the six best cases, it significantly outperforms the respective best competitive classifier, yielding profit gains ranging from 0.09 to 1.43 € per customer. Except for one estimate in O4, for these five data sets, all out-of-sample EMPC estimates of ProfLogit are well above the competitors' estimates. In the best case (O2), ProfLogit's EMPC estimates range from 11.11 € to 12.55 €, and its average performance equals $\overline{EMPC} = 12.31 \pm 0.26$ €. This yields, on average, a profit gain of 1.43 € per customer over the respective best competitive model (Linear SVM). In the worst case (O6), ProfLogit's EMPC estimates range from 0.97 € to 1.04 €, and its average performance equals $\overline{EMPC} = 1.00 \pm 0.01$ €. This yields, on average, a loss of 0.01 € per customer over the respective best competitive model (LDA). When measuring the performance with the MPC, the deterministic profit measure, ProfLogit has an average rank of 1, being the most profitable churn model in all nine data sets (Figure 7).

Furthermore, ProfLogit exhibits overall the highest $\bar{\eta}$-precision (average rank: 2.33), meaning it most effectively identifies churners correctly. It also has the highest average $\bar{\eta}$-recall (average rank: 3.00), thus it is capable of detecting the most would-be churners. Logically, when considering the $\bar{\eta}$-based F_1 measure, ProfLogit is also the best performing classifier with an average rank of 1.44—best churn model in eight out of nine data sets.

In two instances, ProfLogit also has the highest H measure estimate, but is otherwise ranked low (average rank: 6.11). Its H superiority is especially pronounced in the UCI data set, which is 0.49, whereas the respective best competitive classifier (ElasticNet) has a value of 0.41. Note that the H measure can approximate the EMPC in which benefits, b_0 and b_1, are set to zero [5]. This might explain ProfLogit's occasional preeminence in terms of the H measure. ProfLogit also takes two times the first place in terms of the MER (KDD and O5), but the last position in 67% of the data sets, resulting in an average rank of 6.39.

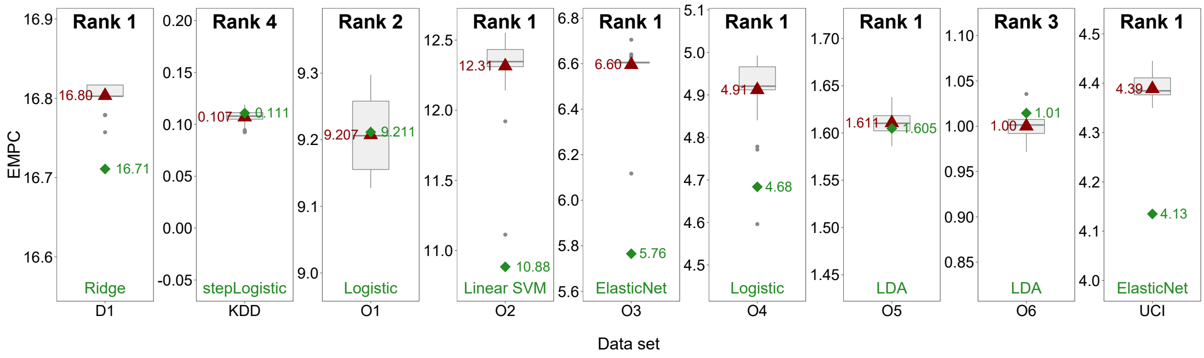


Figure 6: Based on the out-of-sample EMPC estimates, ProfLogit (▲) is in six out of nine data sets the most profitable churn model, and in five out of the six best cases it significantly outperforms the respective best competitive classifier (◆: label at the bottom). For the other three data sets, ProfLogit closely falls behind the best competitive technique. Note that the box plots have been constructed based on $M = 30$ EMPC estimates of ProfLogit measured on the hold-out sample. In order to have a clear view on the estimates, we only picture the respective best competitive classifier to avoid overlapping labels.

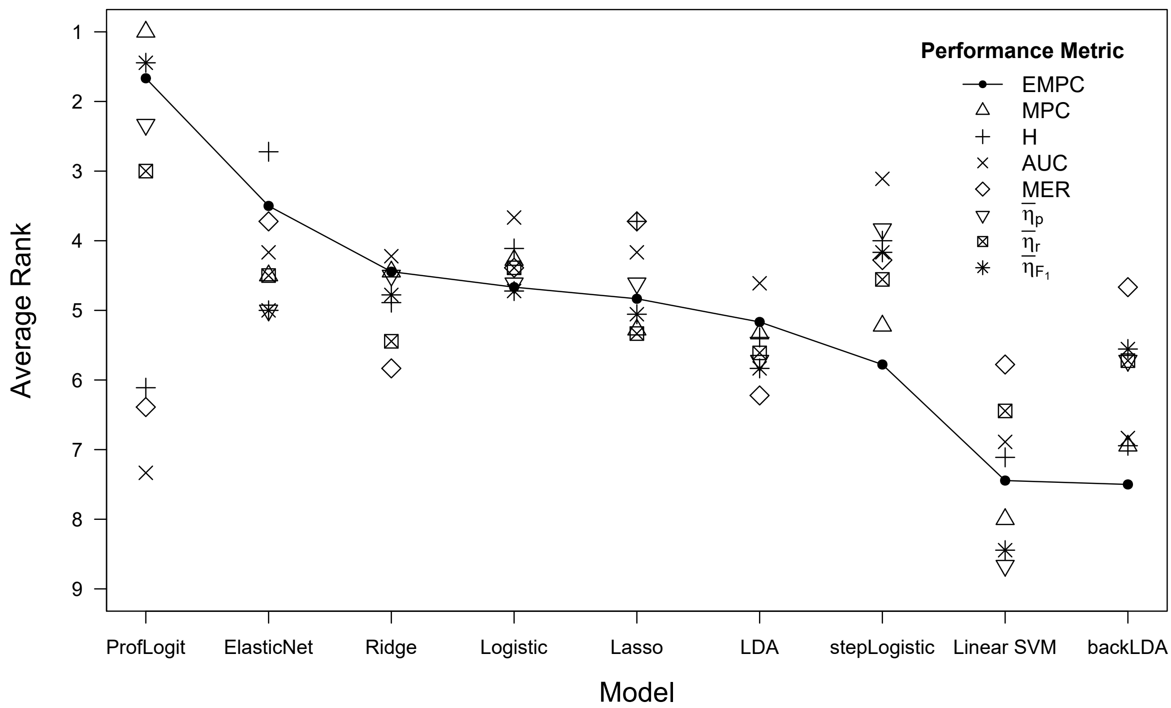


Figure 7: When averaging the ranks of the out-of-sample classification performance estimates over the data sets, ProfLogit has overall the best performance in terms of the EMPC and MPC, thus being the most profitable churn model. Additionally, it has the overall best profit-based hit rate ($\bar{\eta}_p$) and recall ($\bar{\eta}_r$) as well as the highest $\bar{\eta}$-based F_1 measure, which indicates that it is the most effective model to correctly identify churners and detecting the largest proportion of would-be churners. The large discrepancies between the profit- and accuracy-based measures empirically prove that model selection based on the latter category likely results in suboptimal profit.

Evidently, ProfLogit has the overall lowest AUC performance, having in eight out of nine cases a rank of six or larger (average rank: 7.33). However, as pointed out by [5], performance discrepancies between the EMPC and AUC were anticipated. Because ProfLogit maximizes the EMPC, significant discrepancies between these two measures were expected.

Table 4: Comparison of Regression Coefficients of Selected Models

Coef.	ProfLogit	Lasso	Ridge	Elastic Net	Logistic	stepLogistic
β_0	0.108 (0.631)	−0.895	−0.943	−0.943	−0.970 (0.125)***	−0.915 (0.119)***
β_1	—	—	−0.127	−0.127	−0.096 (0.108)	—
β_2	−0.107 (0.089)	−0.934	−1.188	−1.188	−1.566 (0.187)***	−1.550 (0.183)***
β_3	0.016 (0.020)	0.477	0.697	0.697	1.042 (0.187)***	0.963 (0.138)***
β_4	0.404 (0.240)	—	0.021	0.021	−0.083 (0.165)	—
β_5	0.000 (0.001)	0.116	0.224	0.224	0.173 (0.117)	—
β_6	0.014 (0.026)	—	0.312	0.312	1.530 (0.896) .	2.283 (0.766)**
β_7	—	—	0.059	0.059	0.073 (0.103)	—
β_8	—	—	0.008	0.008	−0.011 (0.096)	—
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1						

With the lasso-penalty, ProfLogit performs a profit-based feature selection, and, compared to its competitors, it considers other predictors to be relevant for constructing a profitable churn model (see, e.g., shaded row). Note that ProfLogit's coefficients have been averaged over the $M = 30$ models, each trained with $\lambda = \lambda_{opt}$. Results are based on the O2 data set. A '—' cell means that the corresponding β_j have a zero coefficient value.

Finally, due to the integrated lasso-penalty and the soft-thresholding operator, ProfLogit performs a profit-based feature selection in which the coefficients of predictors that are irrelevant to build a profitable churn model are effectively set to zero. More specifically, it allocates a relatively high (low) regression weight to predictors that are considered as irrelevant (relevant) by the other techniques, which optimize for a nonprofit objective function in the model construction (see, e.g., shaded row in Table 4).

4.3. Sensitivity Analysis of Crossover and Mutation Probabilities

To study the sensitivity of the parameter settings for the crossover probability (p_c) and mutation probability (p_m), we conduct a balanced two-factor experiment and analyze the data using Analysis of Variance (ANOVA). For the two factors, we hypothesize the following values:

$$\text{Factor A:} \quad p_c \in \{0.5, 0.6, 0.7, 0.8, 0.9\},$$

$$\text{Factor B:} \quad p_m \in \{0.05, 0.1, 0.2, 0.3, 0.4\}.$$

We perform the ANOVA analysis for two data sets: KDD and O2. In each configuration, we run ProfLogit 20 times and measure its EMPC performance. Other ProfLogit parameters take their default values (see Section 3), except the regularization parameter is set to its respective optimal value obtained from the previous analysis.

A close inspection of the experimental data revealed that the application of a regular two-way ANOVA is not appropriate because of the violations of the underlying model assumptions of normality and homoscedasticity. We therefore opt for a nonparametric alternative, which has been recently proposed by [32].

The nonparametric ANOVA analysis reveals that there is a significant interaction effect ($F_{16,475} = 2.20$; $p = 0.0047$) between the two factors for the O2 data; whereas no significant effects (i.e., neither an interaction nor a main effect) are found for the KDD data. Performing statistical comparison tests with appropriate p-value correction, the ANOVA analysis uncovers that the mutation rate has overall a stronger influence on EMPC performance than the crossover rate, yet it still significantly varies between p_c values. In particular, settings associated with the lowest p_m value of 0.05 are significantly inferior to other configurations, however the significance only holds for some levels of p_c.

To conclude the sensitivity analysis, we can summarize that ProfLogit's crossover rate and mutation rate are (i) data-dependent, and (ii) EMPC estimates do not differ significantly for settings in which the mutation rate has a value of at least 0.1.

4.4. Discussion

As the benchmark study demonstrates, the incorporation of the profit-based performance measure, EMPC, into an evolutionary-driven classifier can lead to significant profit gains. For example, in the O2 data set, ProfLogit achieves, on average, a profit gain that is at least 1.43 € per customer higher compared to the competitive techniques. Thus, the potential total profit gain could be enormous for a telecom operator with millions of customers. Compared to the previous version of ProfLogit [6], the incorporation of the lasso-regularization into the fitness function (18) and the individual penalization of regression coefficients via soft-thresholding (19) help to considerably improve the EMPC performance on the training set as well as on the test set. Thus, ProfLogit is the overall most profitable classifier in terms of the EMPC and MPC, where, for the latter, it is the best churn model in all nine data sets. This is most likely because of the fact that the MPC and the EMPC are closely related.

As empirically proven by the benchmark study, model selection purely based on accuracy related performance measures such as the AUC and MER likely results in suboptimal profit. ProfLogit, which maximizes profit in its training step, demonstrates this by being the overall most profitable classifier but simultaneously having the worst AUC values (see Figure 7). These results further reinforce the proposition of incorporating the EMPC into the model construction, rather than merely using it for model evaluation.

Although the $\bar{\eta}$-based performance measures introduced in Section 3.3 also rely on the notion of accuracy, ProfLogit interestingly also exhibits the overall highest $\bar{\eta}$-precision (see Figure 7), which is a desirable property as well. A company's first priority is profit maximization. Yet, a major subordinate business requirement is that the outcome of a data mining task is also actionable. This means marketers wish to have as many true churners on their lead list as possible. Fortunately, ProfLogit is not only the most profitable model, it also produces lead lists with the highest hit rates ($\bar{\eta}_p$). In other words, it is the most effective classifier to correctly identify churners, which allows companies to not only focus their marketing resources on the customers that intend to churn but also to focus on those who are the most *profitable* to the company. Recall that the produced lead lists are based on the $\bar{\eta}_{empc}$, and are generated in a completely objective manner. Thus, being able to deliver lead lists for the churn management campaign that are created with maximum profit in mind and simultaneously having the highest hit rate serves both objectives: profit maximization and efficient deployment of marketing resources.

In addition to high profit-based precision, ProfLogit also exhibits the overall highest $\bar{\eta}$-recall (another desirable property), meaning it can detect the largest proportion of would-be churners. Having high $\bar{\eta}_p$ and $\bar{\eta}_r$ estimates, consequently results in a high $\bar{\eta}$-based F_1 measure as well. Hence, as a result, ProfLogit's churn predictions are the overall most effective in detecting would-be churners as well as in identifying the largest proportion of all potential churners. On top, these predictions are optimized for maximum profit, which means that churn prevention efforts primarily focus on customers that are profitable to the organization.

Due to the integrated lasso-regularization and soft-thresholding, ProfLogit performs a profit-based feature selection, revealing which are the relevant predictors to construct a profit maximizing churn model (see Table 4). The selection deviates from the sets of features selected by competitive techniques. This highlights the importance of a profit-sensitive model construction to achieve maximum profit.

5. Conclusions and Future Work

In this paper, we proposed our churn classification technique ProfLogit that utilizes a real-coded genetic algorithm (RGA) to directly optimize the EMPC (14) in the model construction step. Costs and benefits associated with a retention campaign are comprehensively captured by the EMPC measure, which in turn permits the selection of the most profitable model. In this respect, ProfLogit aims to actively construct the most profitable model for a customer churn management campaign (Algorithm 1). Beneath ProfLogit, we exploit the logistic model structure (Eq. (1)) to compute churn scores, and use the RGA to optimize the regression coefficients according to the lasso-regularized EMPC fitness function (18).

In our benchmark study, ProfLogit is the overall most profitable model compared to eight other linear classification techniques. For the study, we applied the classifiers to nine real-life churn data sets, and evaluated their out-of-sample classification performances using accuracy, cost, and profit related performance measures. We firmly confirm [5]'s statement that model selection based on the AUC results in suboptimal profit. In the best cases, ProfLogit outperforms its competitors, leading to substantially higher profit gains; whereas, in the worst case, its profit losses are relatively small compared to the respective best competitive model.

Additionally, we introduced threshold-independent precision and recall measures, as well as a F_1 measure thereof, that are based on the expected profit maximizing fraction $\bar{\eta}_{empc}$. Next to being the most profitable, the benchmark study revealed that ProfLogit also has the overall highest profit-based hit rate ($\bar{\eta}_p$), making it the most effective model to correctly identify churners while aiming for maximum profit. Its superiority over the other classifiers is also clearly expressed with the $\bar{\eta}$-recall and the $\bar{\eta}$-based F_1 measure, indicating that ProfLogit is the overall best performing churn model.

Moreover, with the newly introduced enhancements, ProfLogit performs a profit-based feature selection optimized according to the EMPC. Findings revealed that different features become relevant when constructing a churn model for maximum profit compared to outcomes from accuracy-centric techniques.

In this paper, we have shown that profit maximizing modeling for predictive churn modeling is feasible, and its application can lead to substantially higher profit gains than using other linear classifiers. Thus, our proposed classification algorithm aligns best with the most important business requirement: *profit maximization*. Additionally, ProfLogit produces lead lists, which are required for the execution of the retention campaign, that have the highest profit-based precision ($\bar{\eta}_p$). This enables companies to tailor their marketing resources toward potential churners more efficiently—with special focus on those who are also the most *profitable* to the business.

Concerning future research, we intend to develop a similar profit maximizing classification algorithm that substitutes the logistic model structure with a decision tree induction algorithm. The primary aim here is to construct a profit-driven classifier that can more easily cope with complex data structures such as nonlinearities.

Another important aspect is to study whether the application of alternative nature-inspired algorithms other than RGA would be more favorable for the optimization of ProfLogit's objective function. Promising candidate algorithms to consider would be, for example, the artificial bee colony algorithm (ABC) [33], differential evolution (DE) methods such as SHADE and MPEDE [34], and particle swarm optimization (PSO) methods such as HCLPSO described in [35] or [36].

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix A Comparison of Nature-inspired Algorithms

For the optimization of ProfLogit's objective function, we opt for the real-coded genetic algorithm (RGA) because it is a mature and well-established algorithm that passed the test of time. Nature-inspired algorithms such as differential evolution (DE) and particle swarm optimization (PSO) can be regarded as an interesting alternative to RGA. Therefore, we conduct a comparative study to investigate the performances of the three candidate optimizers.

To do so, we compute the average EMPC performance for the DE and PSO in the exact same manner as described in Section 3.2.1. That is, the regularization parameter is tuned under the given optimizer, and the out-of-sample EMPC performance is computed. We carry out the experiment on all available data sets. To ensure a fair comparison, the algorithms terminate when the maximum number of function evaluations ($\mathrm{NFE}_{\max}$) is reached. In the study, we set $\mathrm{NFE}_{\max}$ to 10,000 as it is similarly done in [37, 38]. As in Section 4, we apply the same rule for determining the population size of RGA and DE: $|\mathcal{P}| = 10\,|\boldsymbol{\theta}| = 10(p+1)$. Typically, the population or swarm size for PSO should be lower than for RGA and DE, here we apply the following rule: $|\mathcal{P}| = 0.25 \times (10\,|\boldsymbol{\theta}|) = 2.5(p+1)$.

Contrasting the EMPC estimates reveals that using the RGA optimizer attains the highest average and median EMPC performance in 9 out of 9 data sets (Figure 8). Hence, we can conclude that the application of RGA is preferable over PSO and DE.

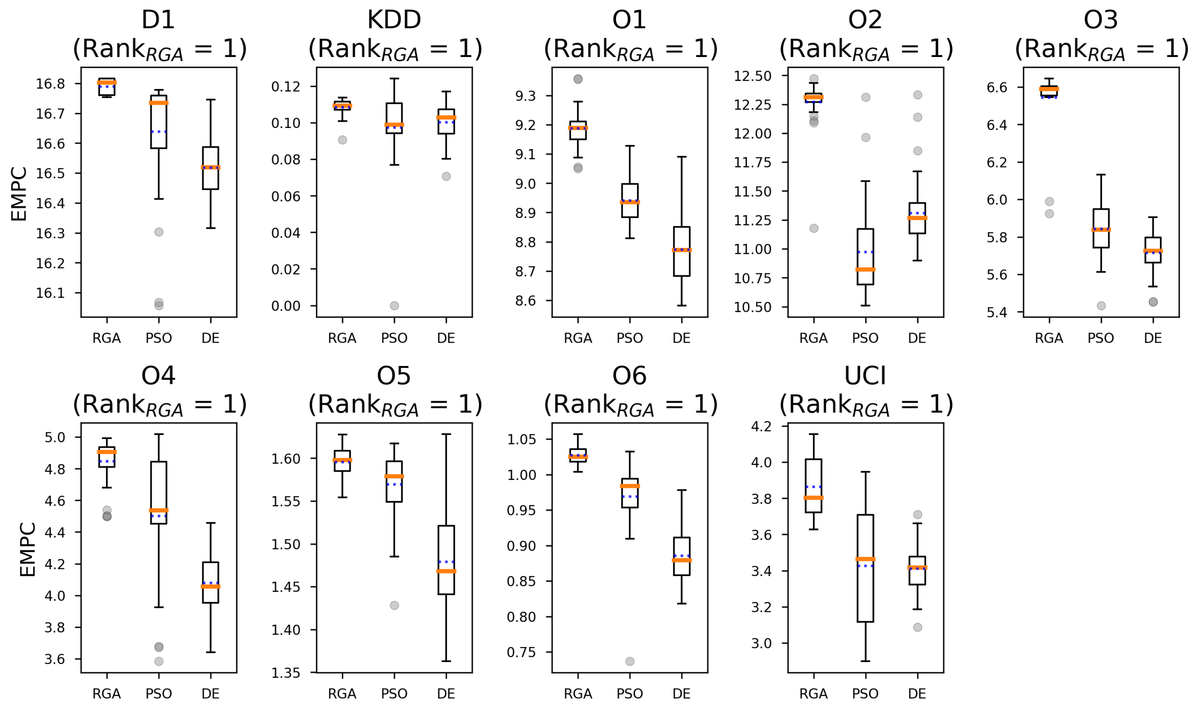


Figure 8: EMPC performance comparison between three nature-inspired algorithms on all data sets: real-coded genetic algorithm (RGA), particle swarm optimization (PSO), and differential evolution (DE). The average (median) performance of each optimizer is shown as a dotted blue (solid orange) line. In conclusion, RGA is the overall best optimizer: it attains the highest average and median out-of-sample EMPC performance in 9 out of 9 data sets.

References

[1] A. D. Athanassopoulos, Customer satisfaction cues to support market segmentation and explain switching behavior, Journal of Business Research 47 (3) (2000) 191–207.

[2] W. Verbeke, D. Martens, C. Mues, B. Baesens, Building comprehensible customer churn prediction models with advanced rule induction techniques, Expert Systems with Applications 38 (3) (2011) 2354–2364.

[3] D. Hand, Measuring classifier performance: a coherent alternative to the area under the ROC curve, Machine Learning 77 (1) (2009) 103–123.

[4] C. Elkan, The foundations of cost-sensitive learning, in: In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, 2001, pp. 973–978.

[5] T. Verbraken, W. Verbeke, B. Baesens, A novel profit maximizing metric for measuring classification performance of customer churn prediction models, IEEE Transactions on Knowledge and Data Engineering 25 (5) (2013) 961–973.

[6] E. Stripling, S. vanden Broucke, K. Antonio, B. Baesens, M. Snoeck, Profit maximizing logistic regression modeling for customer churn prediction, in: IEEE International Conference on Data Science and Advanced Analytics, 2015, pp. 1–10.

[7] T. Verbraken, Business-oriented data analytics: Theory and case studies, Ph.D. dissertation, Dept. LIRIS, KU Leuven, Leuven, Belgium, 2013.

[8] T. Verbraken, C. Bravo, R. Weber, B. Baesens, Development and application of consumer credit scoring models using profit-based classification measures, European Journal of Operational Research 238 (2) (2014) 505–513.

[9] D. W. J. Hosmer, S. Lemeshow, R. X. Sturdivant, Applied logistic regression. 3rd ed., Wiley Series in Probability and Statistics. John Wiley & Sons, 2013.

[10] T. Hastie, R. Tibshirani, M. Wainwright, Statistical Learning with Sparsity: The Lasso and Generalizations, Monographs on Statistics & Applied Probability, Chapman and Hall/CRC, 2015.

[11] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, H. Nielsen, Assessing the accuracy of prediction algorithms for classification: an overview, Bioinformatics 16 (5) (2000) 412–424.

[12] N. Chawla, Data mining for imbalanced datasets: An overview, in: O. Maimon, L. Rokach (Eds.), Data Mining and Knowledge Discovery Handbook, Springer US, 2005, pp. 853–867.

[13] T. Fawcett, An introduction to ROC analysis, Pattern Recogn. Lett. 27 (8) (2006) 861–874.

[14] A. P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recogn. 30 (7) (1997) 1145–1159.

[15] W. J. Krzanowski, D. J. Hand, ROC Curves for Continuous Data, 1st Edition, Chapman & Hall/CRC, 2009.

[16] D. Hand, C. Anagnostopoulos, A better beta for the H measure of classification performance, Pattern Recognition Letters 40 (0) (2014) 41–46.

[17] W. Verbeke, K. Dejaeger, D. Martens, J. Hur, B. Baesens, New insights into churn prediction in the telecommunication sector: A profit driven data mining approach, European Journal of Operational Research 218 (1) (2012) 211–229.

[18] A. E. Eiben, J. E. Smith, Introduction to Evolutionary Computing, SpringerVerlag, 2003.

[19] X. Yu, M. Gen, Introduction to Evolutionary Algorithms, Decision Engineering, Springer, 2010.

[20] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, Oxford, UK, 1996.

[21] T. Bäck, H. P. Schwefel, Evolutionary computation: an overview, in: Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pp. 20–29.

[22] I. Zelinka, A survey on evolutionary algorithms dynamics and its complexity - mutual relations, past, present and future, Swarm and Evolutionary Computation 25 (2015) 2–14.

[23] T. T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm and Evolutionary Computation 6 (2012) 1–24.

[24] T. Bäck, U. Hammel, H. P. Schwefel, Evolutionary computation: comments on the history and current state, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 3–17.

[25] K. De Jong, Evolutionary Computation: A Unified Approach, MIT Press, 2006.

[26] R. Baragona, F. Battaglia, I. Poli, Evolutionary Statistical Procedures: An Evolutionary Computation Approach to Statistical Procedures Designs and Applications, Statistics and Computing, Springer Berlin Heidelberg, 2011.

[27] A. Bahnsen, D. Aouada, B. Ottersten, A novel cost-sensitive framework for customer churn predictive modeling, Decision Analytics 2 (1) (2015) 5.

[28] H. A. Guvenir, M. Kurtcephe, Ranking instances by maximizing the area under ROC curve, IEEE Transactions on Knowledge and Data Engineering 25 (10) (2013) 2356–2366.

[29] D. L. Donoho, I. M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, Biometrika 81 (1994) 425–455.

[30] J. Friedman, T. Hastie, R. Tibshirani, Regularization paths for generalized linear models via coordinate descent, Journal of Statistical Software 33 (1) (2010) 1–22.

[31] S. N. Sivanandam, S. N. Deepa, Introduction to Genetic Algorithms, 1st Edition, Springer Publishing Company, Incorporated, 2007.

[32] J. O. Wobbrock, L. Findlater, D. Gergle, J. J. Higgins, The aligned rank transform for nonparametric factorial analyses using only ANOVA procedures, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11, ACM, New York, NY, USA, 2011, pp. 143–146.

[33] A. Rajasekhar, N. Lynn, S. Das, P. N. Suganthan, Computing with the Collective Intelligence of Honey Bees–A Survey, Swarm and Evolutionary Computation 32 (2017) 25–48.

[34] S. Das, S. S. Mullick, P. N. Suganthan, Recent Advances in Differential Evolution–An Updated Survey, Swarm and Evolutionary Computation 27 (2016) 1–30.

[35] N. Lynn, P. N. Suganthan, Heterogeneous Comprehensive Learning Particle Swarm Optimization with Enhanced Exploration and Exploitation, Swarm and Evolutionary Computation 24 (2015) 11–24.

[36] T. Kadavy, M. Pluhacek, A. Viktorin, R. Senkerik, Hypersphere Universe Boundary Method Comparison on HCLPSO and PSO, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2017, pp. 173–182.

[37] K. Mullen, Continuous Global Optimization in R, Journal of Statistical Software 60 (6) (2014) 1–45.

[38] M. M. Ali, C. Khompatraporn, Z. B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems, Journal of Global Optimization 31 (4) (2005) 635–672.