Expert Systems with Applications 36 (2009) 4626–4636

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Handling class imbalance in customer churn prediction

J. Burez, D. Van den Poel *

Marketing Modeling/Analytical Customer Relationship Management at Ghent University, Faculty of Economics and Business Administration, Department of Marketing, Tweekerkenstraat 2, B-9000 Gent, Belgium

ARTICLE INFO

Keywords:
Rare events
Class imbalance
Under-sampling
Oversampling
Boosting
Random forests
CUBE
Customer churn
Classifier

ABSTRACT

Customer churn is often a rare event in service industries, but of great interest and great value. Until recently, however, class imbalance has not received much attention in the context of data mining [Weiss, G. M. (2004). Mining with rarity: A unifying framework. *SIGKDD Explorations, 6* (1), 7–19]. In this study, we investigate how we can better handle class imbalance in churn prediction. Using more appropriate evaluation metrics (AUC, lift), we investigated the increase in performance of sampling (both random and advanced under-sampling) and two specific modelling techniques (gradient boosting and weighted random forests) compared to some standard modelling techniques.

AUC and lift prove to be good evaluation metrics. AUC does not depend on a threshold, and is therefore a better overall evaluation metric compared to accuracy. Lift is very much related to accuracy, but has the advantage of being well used in marketing practice [Ling, C., & Li, C. (1998). Data mining for direct marketing problems and solutions. In *Proceedings of the fourth international conference on knowledge discovery and data mining (KDD-98)*. New York, NY: AAAI Press].

Results show that under-sampling can lead to improved prediction accuracy, especially when evaluated with AUC. Unlike Ling and Li [Ling, C., & Li, C. (1998). Data mining for direct marketing problems and solutions. In *Proceedings of the fourth international conference on knowledge discovery and data mining (KDD-98)*. New York, NY: AAAI Press], we find that there is no need to under-sample so that there are as many churners in your training set as non churners. Results show no increase in predictive performance when using the advanced sampling technique CUBE in this study. This is in line with findings of Japkowicz [Japkowicz, N. (2000). The class imbalance problem: significance and strategies. In *Proceedings of the 2000 international conference* on *artificial intelligence (IC-AI'2000): Special track on inductive learning*, Las Vegas, Nevada], who noted that using sophisticated sampling techniques did not give any clear advantage. Weighted random forests, as a cost-sensitive learner, performs significantly better compared to random forests, and is therefore advised. It should, however always be compared to logistic regression. Boosting is a very robust classifier, but never outperforms any other technique.

1. Introduction

Customer retention is one of the most important issues for companies. Customer churn prevention, as part of a Customer Relationship Management (CRM) approach, is high on the agenda. Big companies implement churn prediction models to be able to detect possible churners before they effectively leave the company. When predicting churn, more and more data mining techniques are applied.

Fortunately for the companies involved, churn is often a rare object (e.g. Neslin, Gupta, Kamakura, Lu, & Mason, 2006), but of great interest and great value. Based on the current academic literature and the needs of the *leading edge* industry practitioners, Gupta et al. (2006) state that understanding how to model rare events is one *the* issues that represent opportunities for future research:

> *"The models developed in marketing are typically applied to situations where the events of interest occur with some frequency (e.g., customer churn, customer purchases). These models can break down when applied to setting where the behaviour of interest is rare. For example, when modelling the correlates of customer acquisition in a low-acquisition rate setting, the performance of the familiar logit model is often unacceptable. There may be opportunity to gain valuable insights from the statistics literature on the modelling of rare events."* (Gupta et al., 2006, 149).

Until recently, however, class imbalance has not received much attention in the context of data mining (Weiss, 2004). Now, as increasingly complex real-world problems are addressed, the problem of imbalanced data is taking centre stage. Weiss (2004)

* Corresponding author. Tel.: +32 9 264 89 80; fax: +32 9 264 42 79.
E-mail address: dirk.vandenpoel@UGent.be (D. Van den Poel).
URL: http://www.crm.UGent.be (D. Van den Poel).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.05.027

defined six data mining problems related to rarity, and lists ten methods to address them. In this study, we investigate how we can better handle class imbalance in churn prediction, applying four of those ten methods.

First, we use more appropriate evaluation metrics. ROC analysis is used, as AUC does not place more emphasis on one class over the other, so it is not biased against the minority class. We explain the relationship of the ROC curve with the cumulative gains and the lift curve, both often used in churn prediction practice. Lift is the preferred evaluation matrix, used by CRM managers in the field (Ling & Li, 1998). By using weighted random forests, we apply a second method, namely that of cost-sensitive learning. Sampling is a third possible method to deal with class imbalance. We both apply a basic sampling method (under-sampling) and an advanced one (CUBE, an efficient balanced sampling method). Last method used is boosting, as we estimate a stochastic gradient boosting learner.

A lot of comparative studies have been carried out. The question can be raised: why yet another comparative study? This study differs from previous studies in that our datasets for churn modelling are substantially different from those traditionally used for comparative studies. While often comparative studies are carried out on the UCI repository (Salzberg, 1997), with easy classification tasks, and high accuracy, in this study churn prediction is the issue. For six different companies churn prediction models are created: the data is real, and big, the models developed are relevant, classification is harder.

2. Handling class imbalance: four possible solutions

Weiss (2004) draws up six categories of problems that arise when mining imbalanced classes.

1. *Improper evaluation metrics*: often, not the best metrics are used to guide the data mining algorithms and to evaluate the results of data mining.
2. *Lack of data: absolute rarity*: the number of examples associated with the rare class is small in an absolute sense, which makes it difficult to detect regularities within the rare class.
3. *Relative lack of data: relative rarity*: objects are not rare in absolute sense, but are rare relative to other objects, which makes it hard for greedy search heuristics, and more global methods are, in general, not tractable.
4. *Data fragmentation*: Many data mining algorithms, like decision trees, employ a divide-and-conquer approach, where the original problem is decomposed into smaller and smaller problems, which results in the instance space being partitioned into smaller and smaller pieces. This is a problem because regularities can then only be found within each individual partition, which will contain less data.
5. *Inappropriate inductive bias*: Generalizing from specific examples, or induction, requires an extra-evidentiary bias. Without such a bias "inductive leaps" are not possible and learning cannot occur. The bias of a data mining system is therefore critical to its performance. Many learners utilize a general bias in order to foster generalization and avoid overfitting. This bias can adversely impact the ability to learn rare cases and rare classes.
6. *Noise*: Noisy data will affect the way any data mining system behaves, but interesting is that noise has a greater impact on rare cases than on common cases.

Weiss (2004) also describes ten methods for dealing with those problems associated with rarity. Table 1 (Weiss, 2004) summarizes the mapping of problems with rarity to the methods for addressing these problems. Note that for each problem multiple solutions are available. In these cases, the best (most direct, most useful) solutions are listed first and those solutions that only indirectly address the underlying problem are italicized.

In this study, we focus on problems 1, 3 and 6. As this study evolves around churn prediction – using six real-life churn data sets – those problems are the most relevant. Churn data sets are generally rather big, what makes that absolute rarity is not an issue. Logistic regression does not know the data fragmentation problem, whereas ensemble methods should not be bothered by it as much as single decision trees are. Besides, data fragmentation is more of a concern when there is absolute rarity. The use of a more appropriate inductive bias is something for further research.

We apply in this study more appropriate evaluations metrics, a cost-sensitive learner, two sampling techniques and boosting. They are sketched hereunder.

2.1. More appropriate evaluation metrics

2.1.1. Classification accuracy

It is often hard or nearly impossible to construct a perfect classification model that would correctly classify all examples from the test set. Therefore, we have to choose a suboptimal classification model that best suits our needs and works best on our problem domain.

In our case, we could use a classifier that makes a binary prediction (i.e. the customer will either stay with the company or not) or a classifier that gives a probabilistic class prediction to which class an example belongs. The first is called binary classifier and the latter is called probabilistic classifier. One can easily turn a probabilistic classifier into a binary one using a certain threshold (traditionally so that the Yrate in the test set is equal to the churn rate in the original training set – see further).

2.1.2. Binary classifiers

When dealing with a binary classification problem we can always label one class as a positive (in our case a churner) and the other one as a negative class (a non churner). The test set consists of P positive and N negative examples. A classifier assigns a class to each of them, but some of the assignments are wrong. To assess the classification results we count the number of true positive (TP), true negative (TN), false positive (FP) (actually negative, but classified as positive) and false negative (FN) (actually positive, but classified as negative) examples.

It holds

$$\mathrm{TP} + \mathrm{FN} = P$$

and

$$\mathrm{TN} + \mathrm{FP} = N$$

The classifier assigned TP + FP examples to the positive class and TN + FN examples to the negative class. Let us define a few well-known and widely used measures:

$$\text{specificity} = \frac{\mathrm{TN}}{N} \Rightarrow 1 - \text{specificity} = \frac{\mathrm{FP}}{N} = \mathrm{FPrate}$$

$$\text{sensitivity} = \frac{\mathrm{TP}}{P} = \mathrm{TPrate} = \text{recall}$$

$$\mathrm{Yrate} = \frac{\mathrm{TP} + \mathrm{FP}}{P + N}$$

$$\text{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

$$\text{accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N}$$

$$\Rightarrow \text{misclassification error} \quad (\mathrm{MER}) = 1 - \text{accuracy}$$

Precision, recall and accuracy (or MER) are often used to measure the classification quality of binary classifiers. The FPrate measures

Table 1
A mapping of data mining problems associated with rarity to methods addressing these problems

	Data mining problem	Method to address the problem	
1.	Improper evaluation metrics • for evaluating final result • to guide data mining	**1.** **8.**	**More appropriate evaluation metrics** **Cost-sensitive learning**
2.	Absolute rarity	9.	Over-sampling Remainder identical to cell below
3.	Relative rarity "needles in a haystack"	5. 6. **9.** 2. 10.2 7. **8.** 4. 10.3 **1.** 3. **10.1**	Learn only the rare class Segmenting the data **Sampling (over- and under)** Non-greedy search techniques Two-phase rule induction Accounting for rare items **Cost-sensitive learning** Knowledge/human interaction Rare cases into separate classes **More appropriate evaluation metrics** More appropriate inductive bias ***Boosting***
4.	Data fragmentation "rare classes/cases split apart"	2. 10.2 5. 10.3 **9.**	Non-greedy search techniques Two-phase rule induction Learn only the rare class Rare cases into separate classes ***Sampling***
5.	Inappropriate bias	3. **1.** **8.**	More appropriate inductive bias ***Appropriate evaluation metrics*** ***Cost-sensitive learning***
6.	Noise	**9.2** 3.	**Advanced sampling** More appropriate inductive bias

the fraction of non churners that are misclassified as churners. The TPrate or recall measures the fraction of churners correctly classified. Precision measures that fraction of examples classified as churner that are truly churner.

Lift tells how much better a classifier predicts compared to a random selection. It compares the precision to the overall churn rate in the test set

$$\text{Lift} = \frac{\text{Precision}}{P/(P+N)} = \frac{\text{Sensitivity}}{\text{Yrate}}$$

2.1.3. Probabilistic classifiers

Probabilistic classifiers assign a score or a probability to each example. A probabilistic classifier is a function $f{:}X \to [0, 1]$ that maps each example x to a real number $f(x)$. Normally, a threshold t is selected for which the examples where $f(x) \geqslant t$ are considered churner and the others are considered non churner.

This implies that each pair of a probabilistic classifier and threshold t defines a binary classifier. Measures defined in the section above can therefore also be used for probabilistic classifiers, but they are always a function of the threshold t.

Note that $\text{TP}(t)$ and $\text{FP}(t)$ are always monotonic descending functions. For a finite example set they are stepwise, not continuous. By varying t we get a family of binary classifiers.

Note that some classifiers return a score between 0 and 1 instead of probability. For the sake of simplicity we shall call them also probabilistic classifiers, since an uncalibrated score function can be converted to a probability function.

2.1.4. ROC, cumulative gains and lift curve

When we want to assess the accuracy of a classifier independent of any threshold, ROC analysis can be used. A ROC graph is defined by a parametric definition

$$x = 1 - \text{specificity}(t), \quad y = \text{sensitivity}(t)$$

Each binary classifier (for a given test set of examples) is represented by a point (1-specificity, sensitivity) on the graph. By varying the threshold of the probabilistic classifier, we get a set of binary classifiers, represented with a set of points on the graph. An example is shown in Fig. 1. The ROC curve is independent of the $P{:}N$ ratio and is therefore suitable for comparing classifiers when this ratio may vary. Note that the precision–recall curve can also be computed, but Davis and Goadrich (2006) showed the curve is equivalent to the ROC curve (an example in Fig. 4).

Area under ROC curve is often used as a measure of quality of a probabilistic classifier. It is close to the perception of classification quality that most people have. AUC is computed with the following formula:

$$\text{AUC} = \int_0^1 \frac{\text{TP}}{P}\, \text{d}\frac{\text{FP}}{N} = \frac{1}{P \cdot N} \int_0^N \text{TP}\, \text{dFP}$$

A random classifier (e.g. classifying by tossing up a coin) has an area under curve 0.5, while a perfect classifier has 1. Classifiers used in practice should therefore be somewhere in between, preferably close to 1.

What does AUC really express? For each negative example count the number of positive examples with a higher assigned score than the negative example, sum it up and divide everything with $P^* N$. This is exactly the same procedure as used to compute the probability that a random positive example has a higher assigned score than random negative example.

$$\text{AUC} = P(\text{Score}_{\text{Random churner}} > \text{Score}_{\text{Random non churner}})$$

The cumulative gains chart, and the (cumulative) lift chart derived from it, are well known in the data mining community specialized in marketing and sales applications (Berry & Linoff, 1999). Apart from their primarily presentational purpose lift charts have not been studied extensively.

The cumulative gains chart (Fig. 2) is also defined by a parametric definition

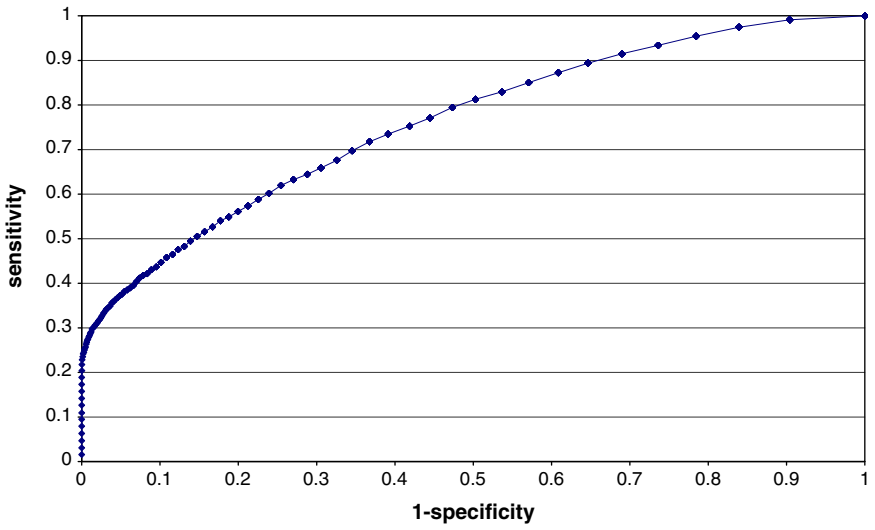

Fig. 1. ROC curve.

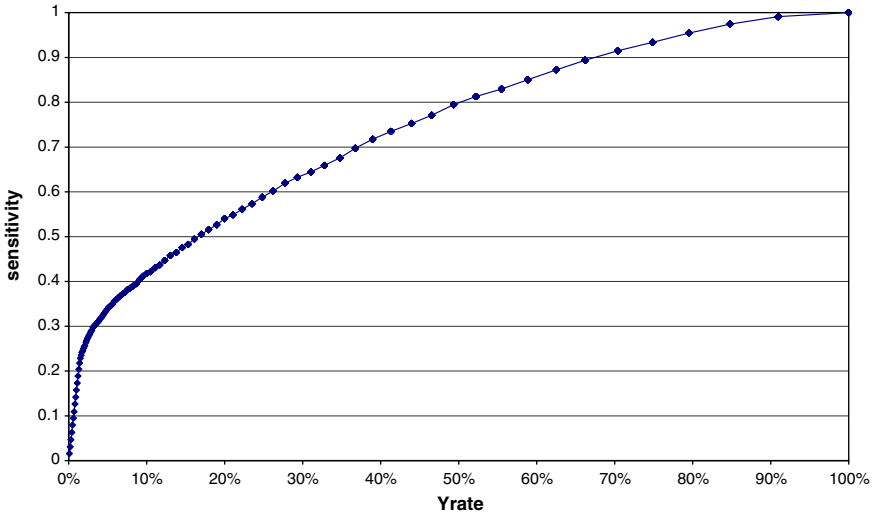

Fig. 2. Cumulative gains chart.

$$x = \text{Yrate}(t), \quad y = \text{sensitivity}(t)$$

Each binary classifier (for a given test set of examples) is represented by a point (Yrate, sensitivity) on the graph. By varying the threshold of the probabilistic classifier, we get a set of binary classifiers, represented with a set of points on the graph. It gives a graphical interpretation of what percentage of customers one has to target to reach a certain percentage of all churners. A purely random sample is presented by a diagonal through (0, 0) and (1, 1), as 10% of the customers will account for 10% of the churners, etc.

The cumulative lift chart (Fig. 3) gives the ratio of the classifier, compared to random sampling, and is thus defined parametrically by

$$x = \text{Yrate}(t), \quad y = \frac{\text{sensitivity}}{\text{Yrate}}(t)$$

The lift curve differs significantly from the earlier mentioned AUC in that the lift curve depends on the churn rate. The AUC has the advantage of being independent of the churn rate.

2.2. Cost-sensitive learning

In many data mining tasks, including churn prediction, it is the rare cases that are of primary interest. Metrics that do not take this into account generally do not perform well in these situations. One solution is to use cost-sensitive learning methods (Weiss, 2004). These methods can exploit the fact that the value of correctly identifying the positive (rare) class outweighs the value of correctly identifying the common class. For two-class problems this is done by associating a greater cost with false negatives than with false positives.

Assigning a greater cost to false negatives than to false positives will improve performance with respect to the positive (rare) class. If this misclassification cost ratio is 3:1, then a region that has ten negative examples and four positive examples will nonetheless be labeled with the positive class. Thus non-uniform costs can bias the classifier to perform well on the positive class – where in this case the bias is desirable.

2.2.1. Weighted random forests

Since the random forests (RF, see methodology) classifier tends to be biased towards the majority class, one can place a heavier penalty on misclassifying the minority class (Chen, Liaw, & Breiman, 2004). A weight is assigned to each class, with the minority class given larger weight (i.e., higher misclassification cost). The class weights are incorporated into the RF algorithm in two places. In the tree induction procedure, class weights are used to weight

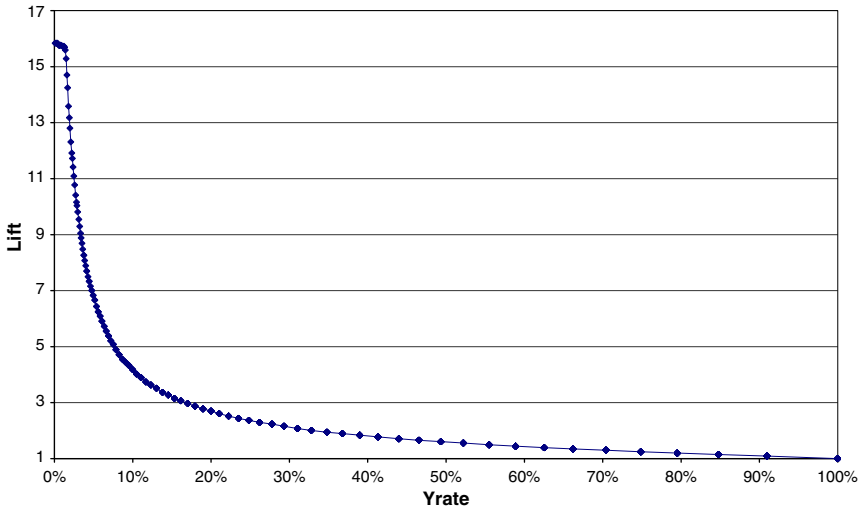

Fig. 3. Cumulative lift curve.

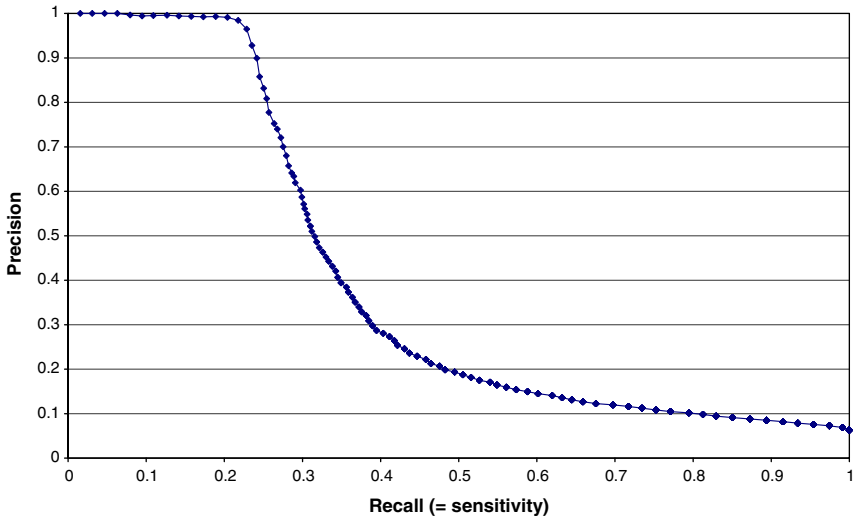

Fig. 4. Precision–recall graph.

the Gini criterion for finding splits. In the terminal nodes of each tree, class weights are again taken into consideration. The class prediction of each terminal node is determined by "weighted majority vote"; i.e., the weighted vote of a class is the weight for that class times the number of cases for that class at the terminal node. The final class prediction for RF is then determined by aggregating the weighted vote from each individual tree, where the weights are average weights in the terminal nodes. Class weights are an essential tuning parameter to achieve desired performance. The out-of-bag estimate of the accuracy from RF can be used to select weights.

2.3. Sampling

One of the most common techniques for dealing with rarity is sampling. The basic idea is to eliminate or minimize rarity by altering the distribution of training examples.

2.3.1. Basic sampling methods

The basic sampling methods include under-sampling and over-sampling. Under-sampling eliminates majority-class examples while over-sampling, in its simplest form, duplicates minority-class examples. Both of these sampling techniques decrease the overall level of class imbalance, thereby making the rare class less rare. These sampling methods do, however, have several drawbacks (Weiss, 2004). Under-sampling discards potentially useful majority-class examples and thus can degrade classifier performance. Because over-sampling introduces additional training cases, it can increase the time necessary to build a classifier. Worse yet, because over-sampling often involves making exact copies of examples, it may lead to overfitting (Chawla, Bowyer, Hall, & Kegelmeyer, 2002; Drummond & Holte, 2003). As an extreme case, classification rules may be introduced to cover a single, replicated, example. More importantly, over-sampling introduces no new data – so it does not address the fundamental "lack of data" issue. This explains why some studies have shown simple over-sampling to be ineffective at improving recognition of the minority class (Ling & Li, 1998; Drummond & Holte, 2003) and why under-sampling seems to have an edge over over-sampling (Chen et al., 2004). For those reasons, in this study we will use under-sampling.

2.3.2. Advanced sampling methods

Advanced sampling methods may use intelligence when removing/adding examples or combine under-sampling and over-sampling techniques. As we choose for under-sampling as our basic

sampling method, we apply CUBE (Deville & Tillé, 2004) as an advanced under-sampling method. The CUBE method is a general method that allows the selection of approximately balanced samples, in that the Horvitz–Thompson estimates for the auxiliary variables are equal, or nearly equal, to their population totals. This method is appropriate for a large set of qualitative or quantitative balancing variables, it allows unequal inclusion probabilities, and it permits us to understand how accurately a sample can be balanced. Moreover, the sampling design respects any fixed, equal or unequal, inclusion probabilities. The method can be viewed as a generalisation of the splitting procedure (Deville & Tillé, 1998) which allows easy construction of new unequal probability sampling methods.

While there is already a very fast implementation of the cube method (Chauvet & Tillé, 2006), it is not yet available, and hence, the original version was used.

2.4. Boosting

Boosting is a technique for improving the accuracy of a predictive function by applying the function repeatedly in a series and combining the output of each function with weighting so that the total error of the prediction is minimized. In many cases, the predictive accuracy of such a series greatly exceeds the accuracy of the base function used alone. The first popular boosting algorithm was AdaBoost, short for adaptive boosting, by Freund and Schapire (1997). It is a meta-algorithm, and can be used in conjunction with many other learning algorithms to improve their performance. AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favour of those instances misclassified by previous classifiers.

Gradient boosting (Friedman, 2002) constructs additive regression models by sequentially fitting a simple parameterized function (base learner) to current "pseudo"-residuals by least squares at each iteration. The pseudo-residuals are the gradient of the loss function being minimized, with respect to the model values at each training data point evaluated at the current step. It is shown that both the approximation accuracy and execution speed of gradient boosting can be substantially improved by incorporating randomization into the procedure. Specifically, at each iteration a sub sample of the training data is drawn at random (without replacement) from the full training data set. This randomly selected sub sample is then used in place of the full sample to fit the base learner and compute the model update for the current iteration. This randomized approach also increases robustness against overcapacity of the base learner. Using the connection between boosting and optimization, Friedman (2001) proposes the Gradient Boosting Machine.

While Weiss (2004) suggests using a basic form of boosting (e.g. AdaBoost), Friedman made those two improvements just mentioned to the boosting algorithm (2002 and 2001). We thus use his gradient boosting machine algorithm in this paper, with the following options: loss function Bernoulli, 10.000 as number of iterations, and shrinkage = 0.005.

3. Data and methodology

3.1. Data

In this study, we make use of six real-life proprietary European churn modelling data sets. All data sets were constructed for company-driven applications, and hence represent a sizeable test bed for comparing the alternative methods mentioned above on predictive accuracy. All cases are customer churn classification cases.

In Table 2, we present a case description. The first five cases involve a contractual setting, the supermarket case is non contractual. Of those five contractual cases, two are situated in the financial services sector. In both cases (at different financial institutions), churn is defined as partial defection on one specific product (Bank2), or on at least one product category (Bank1) (Larivière & Van den Poel, 2004). The three other contractual cases are subscription services (Burez & Van den Poel, 2007, in press). In all three cases, churn is defined as a tacitly renewal of a subscription. As this is often done on contract level, this definition corresponds to total churn. In the supermarket case, partial defection is modelled (Buckinx & Van den Poel, 2005).

In Table 3, we specify some descriptive statistics about the data-sets used, namely (i) the data set, (ii) the number of observations, (iii) the number of churners, (iv) the percentage of churners, and (v) the number of predictive features in the data set.

Note that for this study, only those predictive features were used that resulted from a (forwards) stepwise variable selection procedure.

3.2. Methodology

3.2.1. Cross validation

The usual method to compare classification algorithms is to perform k-fold cross validation experiments to estimate the accuracy of the algorithms and use t-tests to confirm if the results are significantly different (Dietterich, 1998). In cross validation, the data D are divided into k non-overlapping sets, $D_1,\ldots,D_k$. At each iteration i (from 1 to k), the classifier is trained with $D\backslash D_i$ and tested on D_i. While the approach has as advantage that each test is independent from the others, it suffers from that the training sets overlap. It has been shown that comparing algorithms using t-tests on cross validation experiments results in an increased type-I error: the results are incorrectly deemed significantly different more often than expected given the level of confidence used in the test (Dietterich, 1998).

To cope with this problem, we followed the procedure recommended by Dietterich (1998) and Alpaydin (1999) and used five iterations of two-fold cross validation (5×2 cv). In each iteration, the data were randomly divided in halves. One half was input to the algorithms, and the other half was used to test the final solution; and the other way around. The accuracy results presented in the next section are the average AUC, classification error and lift values of the ten tests.

Table 2
Case description of data sets used

Dataset	Churn
Bankl	Partial defection: whether customers churn on at least one product in a three month period
Bank2	Partial defection of current account holders on a twelve month period
Mobile	Defection of residential postpaid customers on a monthly basis
Newspaper	Renewal of newspaper subscribers (12 month subscriptions)
PayTV	Renewal of payTV subscribers (12 month subscriptions)
Supermarket	Partial defection of customers of a supermarket chain on a four month period

Table 3
Descriptive statistics of the data sets used

Dataset	Number of observations	Number of churners	Churn rate	Number of predictive features
Bankl	117.808	7.419	6.30	74
Bank2	102.279	6.122	5.99	38
Mobile	100.205	2.983	2.98	73
Newspaper	122.189	7.703	6.30	33
PayTV	143.198	18.720	13.07	81
Supermarket	32.371	8.140	25.15	21

For the under-sampling, we further sampled randomly from the non churners of the ten training sets, so that the churn rate in the training set had the desired percentage. Those desired churn rates range from the actual churn rate up to 95%, following Weiss and Provost (2003).

The final results should always be tested on unseen data. The accuracy results that we present are obtained by testing the final solutions on the half of the data that has not been considered at all by the algorithms.

3.2.2. 5 × 2 CV F test

Having an outer 5 × 2 cross validation loop allows us to partition the data to do proper comparisons on unseen testing data and also use the combined *F* test proposed by Alpaydin (1999), which is an improvement over the 5 × 2 cv Paired F test as proposed by Dietterich (1998). The combined F test ameliorates the problems of the cross validated *t*-test (when doing *k*-fold cv) and has high power.

Let $p_i^{(j)}$ denote the difference in the accuracy of two classifiers in fold *j* of the *i*th iteration of 5 × 2 cv, $\bar{p}, = (p_i^{(1)} + p_i^{(2)})/2$ denote the mean, and $s_i^2 = (p_i^{(1)} - \bar{p})^2 + (p_i^{(2)} - \bar{p})^2$ the variance, then

$$f = \frac{\sum_{i=1}^{5}\sum_{j=2}^{2}(p_i^{(j)})^2}{2\sum_{i=1}^{5}s_i^2}$$

is approximately *F* distributed with ten and five degrees of freedom. We rejected the null hypothesis that the two algorithms have the same error rate with $\alpha = 0.10$ significance level if $f > 3.297$. All the algorithms used the same training and testing data in the two folds of the five cross validation experiments.

3.2.3. Comparing correlated AUCs

To test statistical significant differences between average AUC over 5 × 2 cv, there is – to our knowledge – no statistic yet. We therefore used the method proposed by DeLong, DeLong, and Clarke-Pearson (1988) to compare the AUC of two correlated algorithms, on each of the 5 × 2 sets. Correlated in this context means that different algorithms are applied on the same test set. If, on at least 7 of the 10 runs, the difference in AUC was significant, we considered the 2 average AUCs significantly different.

3.3. Techniques

As mentioned before, we use weighted random forests and stochastic gradient boosting as a means to handle imbalanced churn data. To compare different sampling techniques however, we used the normal version of random forests, and logistic regression, which is still the standard technique used in practice (Neslin et al., 2006).

3.3.1. Random forests

Random forests blends elements of random subspaces and bagging in a way that is specific to using decision trees as base classifier. At each node in the tree, a subset of the available features is randomly selected and the best split available within those features is selected for that node. Also, bagging is used to create the training set of data items for each individual tree. The number of features randomly chosen (from n total) at each node is a parameter of this approach. Following Breiman (2001), we considered versions of random forests created with random subsets of size $[\log_2(n) + 1]$, rounded to the above integer. Random forests have been used for customer retention modeling by Larivière and Van den Poel (2005).

3.3.2. Logistic regression

Logistic regression modelling is very appealing for four reasons: (1) logit modelling is well-known, conceptually simple and frequently used in marketing (Bucklin & Gupta, 1992), especially at the level of the individual consumer (Neslin et al., 2006); (2) the ease of interpretation of logit is an important advantage over other methods (e.g. neural networks); (3) logit modeling has been shown to provide good and robust results in general comparison studies, for both churn prediction (Neslin et al., 2006) and credit scoring (Baesens et al., 2003) and (4) more specifically in database marketing, it has been shown by several authors (Levin & Zahavi, 1998) that logit modeling may even outperform more sophisticated methods.

4. Results

4.1. Under-sampling and CUBE

We start with investigating the effect of under-sampling on predictive performance. We do this separately for logistic regression (LR) and random forests (RF). A last part of these results will compare both techniques together with two other techniques.

In Table 4 (for LR) and Table 5 (for RF), the averages over 5 × 2 cv of both AUC, classification error and lift are reported for the different samples and the different cases. Under the name of the case,

Table 4
Under-sampling and logistic regression

Logistic regression		Original	5	10	20	30	40	50	60	70	80	90	95
Bankl 6.30	AUC	0.6959		0.6966	0.6971	**0.6974**	**0.6974**	0.6965	0.6949	0.6934	0.6893	0.6802	0.6604
	Error	0.1032		0.1033	0.1033	0.1032	0.1033	0.1036	0.1039	0.1042	0.1048	0.1057	0.1091
	Lift	2.8689		2.8659	2.8626	*2.8692*	2.8591	2.8215	2.7884	2.7513	2.6830	2.5693	2.2362
Bank2 5.99	AUC	0.8277		0.8289	0.8309	0.8315	0.8317	**0.8318**	0.8315	0.8303	0.8287	0.8246	0.8155
	Error	0.0829		0.0828	0.0830	0.0829	0.0830	0.0834	0.0838	0.0843	0.0854	0.0869	0.0907
	Lift	5.1477		*5.1488*	5.1346	5.1369	5.1217	5.0749	5.0184	4.9416	4.7917	4.5870	4.0838
Mobile 2.98	AUC	0.6785	**0.6790**	**0.6790**	0.6776	0.6767	0.6749	0.6722	0.6675	0.6627	0.6524	0.6315	0.6097
	Error	0.0518	0.0518	0.0519	0.0519	0.0521	0.0523	0.0526	0.0530	0.0536	0.0543	0.0568	0.0758
	Lift	4.4099	*4.4176*	4.3601	4.3329	4.2282	4.0941	3.9632	3.7207	3.3921	2.9927	2.0839	1.4725
Newspaper 6.30	AUC	0.6763		0.6765	0.6771	**0.6773**	0.6772	0.6769	0.6768	0.6755	0.6736	0.6695	0.6613
	Error	0.1028		0.1029	0.1033	0.1036	0.1039	0.1044	0.1052	0.1062	0.1078	0.1091	0.1109
	Lift	*2.9351*		2.9147	2.8662	2.8252	2.7900	2.7343	2.6318	2.5090	2.3035	2.1585	1.9520
PayTV 13.07	AUC	0.7714			0.7730	0.**77451**	0.7752	**0.7758**	**0.7758**	0.7753	0.7737	0.7680	0.7594
	Error	0.1383			0.1382	0.1383	0.1384	0.1386	0.1389	0.1392	0.1401	0.1419	0.1448
	Lift	3.6026			*3.6074*	3.6045	3.5990	3.5958	3.5862	3.5759	3.5503	3.4994	3.4124
Supermarket 25.15	AUC	**0.8175**				0.8174	0.8174	0.8173	0.8171	0.8168	0.8161	0.8144	0.8091
	Error	0.2097				0.2096	0.2094	0.2098	0.2099	0.2105	0.2109	0.2120	0.2153
	Lift	2.3183				2.3194	*2.3205*	2.3179	2.3168	2.3119	2.3087	2.3003	2.2739

Table 5
Under-sampling and random forests

Random forests		Original	5	10	20	30	40	50	60	70	80	90	95
Bankl 6.30	AUC	0.7619		0.7646	**0.7662**	0.7654	0.7629	0.7587	0.7527	0.7453	0.7321	0.7050	0.6817
	Error	0.0819		0.0821	0.0820	0.0827	0.0835	0.0848	0.0859	0.0884	0.0910	0.0998	0.1071
	Lift	*5.6033*		5.5905	5.5864	5.4967	5.4031	5.2524	5.1088	4.8228	4.5473	3.7964	3.2139
Bank2 5.99	AUC	0.8706		**0.8717**	0.8715	0.8703	0.8678	0.8655	0.8619	0.8558	0.8481	0.8331	0.8122
	Error	0.0647		0.0654	0.0671	0.0689	0.0707	0.0721	0.0741	0.0766	0.0796	0.0848	0.0928
	Lift	*7.6918*		7.5898	7.3555	7.1025	6.8489	6.6620	6.3953	6.0529	5.6830	5.1285	4.5178
Mobile 2.98	AUC	0.6446	0.6507	0.6567	0.6596	0.6619	**0.6626**	0.6584	0.6503	0.6372	0.6156	0.5860	0.5590
	Error	0.0536	0.0535	0.0531	0.0530	0.0531	0.0531	0.0537	0.0539	0.0543	0.0553	0.0575	0.0652
	Lift	3.8006	3.8567	4.0039	*4.0341*	3.9820	3.8810	3.6690	3.4842	3.2468	2.9556	2.3981	1.8833
Newspaper 6.30	AUC	0.7118		0.7168	0.7238	**0.7257**	0.7256	0.7243	0.7202	0.7107	0.6946	0.6747	0.6529
	Error	0.0980		0.0978	0.0975	0.0982	0.0986	0.0992	0.1004	0.1021	0.1053	0.1098	0.1182
	Lift	3.5755		3.6170	*3.6535*	3.5733	3.5156	3.4419	3.3165	3.1392	2.8180	2.5328	2.1374
PayTV 13.07	AUC	0.7581			0.7609	**0.7628**	0.7626	0.7613	0.7585	0.7513	0.7386	0.7094	0.6744
	Error	0.1448			0.1449	0.1452	0.1468	0.1487	0.1520	0.1566	0.1639	0.1767	0.1958
	Lift	3.4167			3.4153	3.4059	3.3592	3.3053	3.2106	3.0799	2.8800	2.5386	2.1031
Supermarket 25.15	AUC	0.8172				0.8176	**0.8179**	0.8171	0.8157	0.8131	0.8076	0.7963	0.7775
	Error	0.2081				0.2079	0.2076	0.2102	0.2114	0.2153	0.2208	0.2304	0.2444
	Lift	2.3288				2.3296	*2.3340*	2.3127	2.3032	2.2710	2.2272	2.1515	2.0425

the churn rate (in the test set) is given. Column headings indicate the percentage of churners present in the training set: '10' means that 10% of the new formed training set are churners. The non churners out of the original training set were under-sampled so that this percentage was reached. The models trained on the different training sets are then applied on the test set. Resulting evaluation metrics are calculated on the outcome of the application of the model on the test set. The original model indicates the model on the full training set, so without under-sampling.

As we started from the original training sets (with their original churn rate), a few blocks in Tables 4 and 5 are empty. For e.g. the PayTV case, the original churn rate is 13,07%. The first under-sampled training set has by definition a higher churn rate, in this case 20%. Columns 5 and 10 are thus left blank.

The highest AUC is printed in bold, the lowest error rate is underlined, and the highest lift is in italic. For every case, and for both AUC and the error, we calculated the test statistics. That is, we looked at the highest AUC or lowest error, and checked whether the other models were significantly worse. All models, of which we cannot say they are significantly worse, and thus which are assumed as good statistically as the best model, are highlighted in grey.

We illustrate this with the Bank2 case. For LR, the best AUC is obtained when the training set has 50% churners, and 50% non churners (AUC = 0.8318). The original training set, and all under-sampling up to a 60/40 proportion do not differ significantly what concerns AUC. For RF, the best AUC is obtained for proportion 10/90 (AUC = 0.8717). Original training set up to a 40/60 proportion do not differ significantly. This same information is plotted in Fig. 5. The bigger dots on the graphs are the significantly better models. The same is done for error in Fig. 6.

Overall, for logistic regression (Table 4) we see that, what concerns AUC, under-sampling gives us better results, but in only in two cases, this improvement is significant compared to the AUC achieved without under-sampling. When we look at error, none of the over-sampling models is significantly better then the original model.

When looking at random forests in Table 5, we see that again, what concerns AUC, for all of the cases, the highest AUC is obtained with under-sampling. In half of the cases, this is significantly better then the original model. When looking at error, only for one of the cases under-sampling is significantly better.

The under-sampling on the training set was done randomly. In this second step of the results, we compare the results of a training

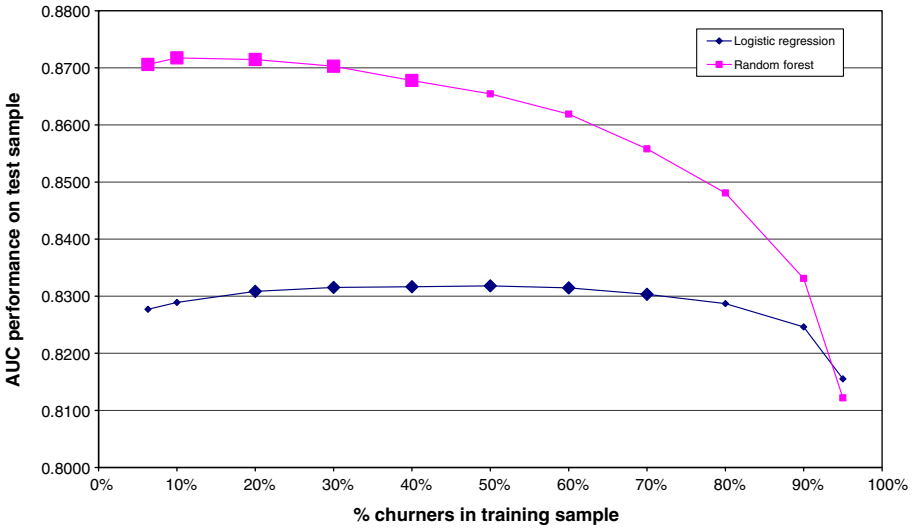

Fig. 5. AUC versus degree of under-sampling (case Bank2).

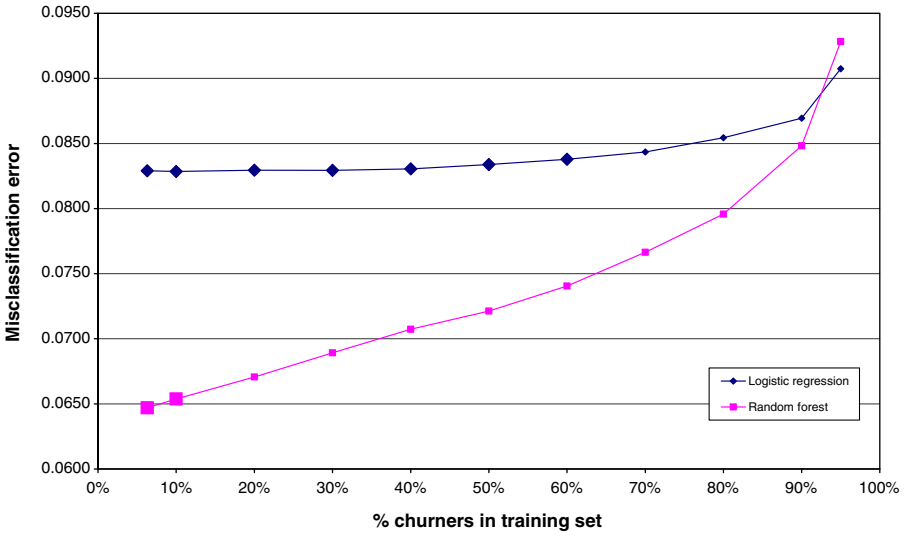

Fig. 6. Error versus degree of under-sampling (case Bank2).

set with 50% churners, but composed in two different ways. The column with heading '50' is the same as in Tables 4 and 5: out of the non churners of the original set, a few are randomly selected so that the training set includes 50% churners and 50% non churners. A second training set was composed with again all churners, and a few non churners to get the 50/50 proportion, but this time this selection of non churners was not done randomly, but based on the CUBE algorithm. Results in Table 6 show that, while CUBE improves AUC slightly, this improvement is never significant. The error stays almost the same.

A last step is to compare logistic regression (LR) and random forests (RF) to two modelling techniques proposed by Weiss (2004). He hinted at using a cost-sensitive learner and boosting algorithms. As cost-sensitive learner, we employ weighted random forests (wRF) by Breiman (2001). As boosting algorithm, we applied the gradient boosting machine (GBM) by Friedman (2001). Results are given in Table 7. When just looking at LR and RF, we can see that in half of the cases (Bank1, Bank2 and Newspaper) RF outperforms LR by a margin. In that case, wRF even improves that performance (significantly in two of the three cases). In two cases (Mobile and PayTV) LR outperforms RF, quite surprisingly. While boosting is very consistent in its performance, it never outperforms any other technique. This is a somewhat unexpected outcome, as Bernoulli is the loss function used for boosting.

In three case (weighted) RF clearly wins, other times LR wins by a huge margin. Running both, and taking the best of both seems the most secure option. Combining both the ideas of LR and RF might be another option, resulting in a powerful classifier (e.g. Prinzie & Van den Poel, 2008).

Table 6
CUBE versus random under-sampling

		Logistic regression			Random forest	
		CUBE	50		CUBE	50
Bankl	AUC	**0.6974**	0.6965	AUC	**0.7603**	0.7587
	Error	0.1033	0.1036	Error	0.0846	0.0848
	Lift	2.8572	2.8215	Lift	5.2876	5.2524
Bank2	AUC	**0.8319**	0.8318	AUC	0.8653	**0.8655**
	Error	0.0834	0.0834	Error	0.0725	0.0721
	Lift	5.0706	5.0749	Lift	6.6107	6.6620
Mobile	AUC	**0.6746**	0.6722	AUC	**0.6610**	0.6584
	Error	0.0526	0.0526	Error	0.0533	0.0537
	Lift	3.9204	3.9632	Lift	3.8008	3.6690
Newspaper	AUC	**0.6774**	0.6769	AUC	0.7242	**0.7243**
	Error	0.1042	0.1044	Error	0.0992	0.0992
	Lift	2.7533	2.7343	Lift	3.4625	3.4419
PayTV	AUC	0.7757	**0.7758**	AUC	**0.7619**	0.7613
	Error	0.1386	0.1386	Error	0.1488	0.1487
	Lift	3.5952	3.5958	Lift	3.3027	3.3053
Supermarket	AUC	0.8172	**0.8173**	AUC	**0.8173**	0.8171
	Error	0.2099	0.2098	Error	0.2100	0.2102
	Lift	2.3165	2.3179	Lift	2.3142	2.3127

Table 7
Predictive performance of a cost-sensitive learner and a boosting algorithm

		LR	RF	wRF	GBM
Bankl	AUC	0.6959	0.7619	**0.7622**	0.7284
	Error	0.1032	0.0819	0.0821	0.0943
	Lift	2.8689	*5.6033*	5.5533	4.0526
Bank2	AUC	0.8277	0.8706	**0.8735**	0.8534
	Error	0.0829	0.0647	0.0639	0.0746
	Lift	5.1477	7.6918	*1.1315*	6.2941
Mobile	AUC	**0.6785**	0.6446	0.6433	0.6727
	Error	0.0518	0.0536	0.0542	0.0523
	Lift	*4.4099*	3.8006	3.7875	4.1589
Newspaper	AUC	0.6763	0.7118	**0.7189**	0.7035
	Error	0.1028	0.0980	0.0981	0.1014
	Lift	2.9351	*3.5155*	3.5404	3.1718
PayTV	AUC	**0.7714**	0.7581	0.7614	0.7559
	Error	0.1383	0.1448	0.1439	0.1501
	Lift	*3.6026*	3.4167	3.4391	3.2582
Supermarket	AUC	0.8175	0.8172	0.8181	**0.0201**
	Error	0.2097	0.2081	0.2076	0.2086
	Lift	2.3183	2.3288	*2.3334*	2.3262

5. Conclusions

Churn is often a rare object, but of great interest and great value (Gupta et al., 2006). Until recently, however, class imbalance has not received much attention in the context of data mining (Weiss, 2004). In this study, we investigate how we can better handle class imbalance in churn prediction, applying 4 of 10 methods, proposed by Weiss (2004). To investigate the impact of those methods, we use six real-life customer churn prediction data sets. This is a major strength of this study, as other (mostly data mining) studies often use old and irrelevant or artificial data.

Both sampling (random and advanced under-sampling), boosting (gradient boosting machine) and a cost-sensitive learner (weighted random forests) are implemented. Using more appropriate evaluation metrics (AUC, lift), we investigated the increase in performance over standard techniques (logistic regression, random forests) and without sampling.

AUC and lift are good evaluation metrics. AUC does not depend on a threshold, and is therefore a better overall evaluation metric compared to error/accuracy. Lift is very much related to accuracy, but has the advantage of being well used in marketing practice (Ling & Li, 1998).

Results show that under-sampling can lead to improved prediction accuracy, especially when evaluated with AUC. Unlike Ling and Li (1998), we find that there is no need to under-sample so that there are as many churners in your training set as non churners. We can however confirm the findings of Weiss and Provost (2003) that there is no general answer as to which class distribution will perform best, and that the answer is surely method and case dependent.

The advanced sampling technique CUBE does not increase predictive performance. This is in line with findings of Japkowicz (2000), who noted that using the sophisticated sampling techniques did not give any clear advantage in the domain considered. Another advanced sampling technique (e.g. SMOTE for over-sampling) might perform better. Weighted random forests, as a cost-sensitive learner, performs significantly better compared to random forests, and is therefore advised. It should, however always be compared to logistic regression. Boosting is a very robust classifier, but never outperforms any other technique.

6. Limitations and directions for further research

Next to the four methods used in this study, Weiss (2004) also puts forward to try non-greedy search techniques (e.g. genetic algorithms), to use a more appropriate inductive bias (using hybrid methods), to learn only the rare case, etc. (see Table). All of those methods might be worth trying.

It would be interesting to compare the modelling techniques used in this study to some state of the art techniques, like support vector machines (Viaene et al., 2001; Coussement & Van den Poel, 2008), Bayesian methods (Baesens, Viaene, Van den Poel, Vanthienen, & Dedene, 2002), neural networks and so on.

For comparing AUCs over two or more algorithms on a data set, doing k fold cv or 5×2 cv, a test statistic should be developed to test significant differences as, to our knowledge, such statistic does not exist today.

For both RF and GBM, one has to set parameters. For both we did some form of optimization, but this was not done rigorously. Predictive performance might be slightly influenced by this parameterisation. E.g., the RF parameter representing the number of variables to be randomly at each node is not tuned in this study. Prinzie and Van den Poel (2008) showed that this can influence results. The same goes for the advanced sampling technique CUBE, where one has to assign a number of variables on which the CUBE algorithm will base its balanced sampling. Both the number of variables and the selection (which variables will you use) can influence the results.

7. Transferability of the approach

The fact that the 5×2 cross-validation test requires 10 models to be built and validated might be responsible for the fact that only few applications involve in such rigorous testing. However, for a variety of reasons, the widespread use of the one-shot train-and-test validation for predictive modelling is not without merit (Verstraeten & Van den Poel, 2006). Indeed, in practice, model builders require a more straightforward insight into the absolute performance of their models, while they would not necessarily proceed in testing whether significant differences occur between different model architectures. A company that realizes that its customers are leaving will want to apply a churn prediction model in a timely manner in order to address the customers at risk. Hence, this company might continue to lose a lot of customers during a very extensive validation procedure, so time efficiency translates seamlessly into cost efficiency, and the company might choose to adopt a more straightforward validation procedure. Additionally, also in scientific readings, the use of the one-shot train-and-test validation is still popular. For example, many recent well-appreciated predictive modeling studies in Marketing Science report the use of a single split (see, e.g. Montgomery, Li, Srinivasan, & Liechty, 2004; Park & Fader, 2004; Swait & Andrews, 2003) for model validation. However, since it has been proven that the results of such a validation procedure are highly dependent on the particular split of the data used (Malthouse, 2001), stratified sampling should be considered. In the case of binary classification, predicted outcome stratified sampling (Verstraeten & Van den Poel, 2006) is a possibility. DeLong et al. can be used to compare probabilistic classifiers, whereas McNemar's test (Everitt, 1977) has been shown to be a good statistical test for comparing binary classifiers, give a one shot train and test split (Dietterich, 1998).

Acknowledgements

The authors would like to express their highest gratitude to the other researchers at the modeling cluster of the Department of Marketing at Ghent University, who contributed the data sets they gathered during their PhDs in order to enable the experiments performed in this study. Logistic regression is implemented in every statistical package. The fortran code for random forests if freely available on the site of Leo Breiman. Any of these techniques, together with the gradient boosting machine of Friedman, is available in R, the free software environment for statistical computing and graphics (see function glm and packages randomForest and gbm in R).

References

Alpaydin, E. (1999). Combined 5×2cv F test for comparing supervised classification learning algorithms. *Neural Computation, 11*(8), 1885–1892.

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J. A. K., & Vanthienen, J. (2003). Benchmarking state of the art classification algorithms for credit scoring. *Journal of the Operational Research Society, 54*(6), 627–635.

Baesens, B., Viaene, S., Van den Poel, D., Vanthienen, J., & Dedene, G. (2002). Bayesian neural network learning for repeat purchase modelling in direct marketing. *European Journal of Operational Research, 138*(1), 191–211.

Berry, M. J. A., & Linoff, G. (1999). *Data mining techniques: for marketing, sales, and customer support*. Morgan Kaufmann Publishers.

Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32.

Buckinx, W., & Van den Poel, D. (2005). Customer base analysis: Partial defection of behaviorally-loyal clients in a non-contractual FMCG retail setting. *European Journal of Operational Research, 164*(1), 252–268.

Bucklin, R. E., & Gupta, S. (1992). Brand choice, purchase incidence, and segmentation: An integrated modeling approach. *Journal of Marketing Research, 29*(2), 201–215.

Burez, J., & Van den Poel, D. (2007). CRM at a pay-TV company: Using analytical models to reduce customer attrition by targeted marketing for subscription services. *Expert Systems with Applications, 32*(2), 277–288.

Burez, J., & Van den Poel, D. (in press). Separating financial from commercial customer churn: A modeling step towards resolving the conflict between the sales and credit department. *Expert systems with applications*. doi:10.1016/j.eswa.2007.07.036.

Chauvet, G., & Tillé, Y. (2006). A fast algorithm for balanced sampling. *Computational Statistics, 21*(1), 53–62.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Chen, C., Liaw, A., & Breiman, L. (2004). Using random forests to learn imbalanced data. Technical Report 666, Statistics Department, University of California at Berkeley.

Coussement, K., & Van den Poel, D. (2008). Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. *Expert systems with applications, 34*(1), 313–327.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In: *Proceedings of the 23rd international conference on machine learning (ICML)*.

DeLong, E. R., DeLong, D. M., & Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics, 44*, 837–845.

Deville, J.-C., & Tillé, Y. (1998). Unequal probability sampling without replacement through a splitting method. *Biometrika, 85*, 89–101.

Deville, J.-C., & Tillé, Y. (2004). Efficient balanced sampling: the cube method. *Biometrika, 91*, 893–912.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation, 10*(7), 1895–1923.

Drummond, C., & Holte, R.C. (2003). C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In: *Workshop on learning from imbalanced data sets II, international conference on machine learning*.

Everitt, B. S. (1977). *The analysis of contingency tables*. London: Chapman & Hall.

Freund, F., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119–139.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 29*(5), 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis, 38*(4), 367–378.

Gupta, S., Hanssens, D., Hardie, B., Kahn, W., Kumar, V., Lin, N., et al. (2006). Modeling customer lifetime value. *Journal of Service Research, 9*(2), 139–155.

Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In: *Proceedings of the 2000 international conference on artificial intelligence (IC-AI'2000): Special track on inductive learning*, Las Vegas, Nevada.

Larivière, B., & Van den Poel, D. (2004). Investigating the role of product features in preventing customer churn, by using survival analysis and choice modeling: The case of financial services. *Expert Systems with Applications, 27*(2), 277–285.

Larivière, B., & Van den Poel, D. (2005). Predicting customer retention and profitability by using random forests and regression forests techniques. *Expert Systems with Applications, 29*(2), 472–484.

Levin, N., & Zahavi, J. (1998). Continuous predictive modeling, a comparative analysis. *Journal of Interactive Marketing, 12*, 5–22.

Ling, C., & Li, C. (1998). Data Mining for direct marketing problems and solutions. In *Proceedings of the fourth international conference on knowledge discovery and data mining (KDD-98)*. New York, NY: AAAI Press.

Malthouse, E. C. (2001). Assessing the performance of direct marketing scoring models. *Journal of Interactive Marketing, 15*(1), 49–62.

Montgomery, A. L., Li, S., Srinivasan, K., & Liechty, J. C. (2004). Modeling online browsing and path analysis using clickstream data. *Marketing Science, 23*(4), 579–595.

Neslin, S., Gupta, S., Kamakura, W., Lu, J., & Mason, C. (2006). Detection defection: Measuring and understanding the predictive accuracy of customer churn models. *Journal of Marketing Research, 43*(2), 204–211.

Park, Y.-H., & Fader, P. S. (2004). Modeling browsing behavior at multiple websites. *Marketing Science, 23*(3), 280–303.

Prinzie, A., & Van den Poel, D. (2008). Random forests for multiclass classification: Random multinomial logit. *Expert Systems with Applications, 34*, 1721–1732.

Salzberg, S. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery, 1*, 317–328.

Swait, J., & Andrews, R. L. (2003). Enriching scanner panel models with choice experiments. *Marketing Science, 22*(4), 442–460.

Verstraeten, G., & Van den Poel, D. (2006). Using predicted outcome stratified sampling to reduce the variability in predictive performance of a one-shot train-and-test split for individual customer predictions. In: *Industrial conference on data mining – posters* 2006 (pp. 214–224).

Viaene, S., Baesens, B., Van Gestel, T., Suykens, J. A. K., Van den Poel, D., Vanthienen, J., et al. (2001). Knowledge Discovery in a direct marketing case using least squares support vector machines. *International Journal of Intelligent Systems, 16*(9), 1023–1036.

Weiss, G. M. (2004). Mining with rarity: A unifying framework. *SIGKDD Explorations, 6*(1), 7–19.

Weiss, G., & Provost, F. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research, 19*, 315–354.