Information Sciences 408 (2017) 84–99

Contents lists available at ScienceDirect

Information Sciences

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

An empirical comparison of techniques for the class imbalance problem in churn prediction

Bing Zhu[a,b], Bart Baesens[b,c,*], Seppe K.L.M. vanden Broucke[b]

[a] Business School, Sichuan University, Chengdu 610064, PR China
[b] Department of Decision Sciences and Information Management, KU Leuven, Leuven 3000, Belgium
[c] School of Management, University of Southampton, Highfield Southampton SO17 1BJ, United Kingdom

ARTICLE INFO

Article history:
Received 18 July 2016
Revised 7 April 2017
Accepted 11 April 2017
Available online 18 April 2017

Keywords:
Churn prediction
Class imbalance
Benchmark experiment
Expected maximum profit measure

ABSTRACT

Class imbalance brings significant challenges to customer churn prediction. Many solutions have been developed to address this issue. In this paper, we comprehensively compare the performance of state-of-the-art techniques to deal with class imbalance in the context of churn prediction. A recently developed expected maximum profit criterion is used as one of the main performance measures to offer more insights from the perspective of cost-benefit. The experimental results show that the applied evaluation metric has a great impact on the performance of techniques. An in-depth exploration of reaction patterns to different measures is conducted by intra-family comparison within each solution group and global comparison among the representative techniques from different groups. The results also indicate there is much space to improve solutions' performance in terms of profit-based measure. Our study offers valuable insights for academics and professionals and it also provides a baseline to develop new methods for dealing with class imbalance in churn prediction.

1. Introduction

Customer churn prediction is an important issue in customer relationship management. In today's increasingly competitive business environment, customers can easily switch between competitors. Some studies have shown that acquiring a new customer is usually five to six times more expensive than retaining an existing customer [9]. Meanwhile, long-term customers tend to be less sensitive to competitive marketing activities and produce higher profits. Therefore, companies are shifting their focus from acquiring new customers to retaining their existing customer base. Accurate churn prediction helps companies to target the right customers to retain and it has hence been recognized as a marketing priority. Consequently, we have witnessed various applications of data mining techniques in this field.

Customer churn is a rare event in many industries, meaning that the number of churners is significantly outnumbered by non-churners. In the telecom sector, for example, the monthly churn rate is roughly 2%. Thus customer churn prediction can be presented as a binary classification task with an imbalanced class distribution, where the churners belong to the minority class and non-churners belong to the majority class. The Class imbalance problem produces a great negative influence on standard classification learning algorithms. Most of them tend to bias toward to the majority class. In extreme cases, they

* Corresponding author.
E-mail address: Bart.Baesens@kuleuven.be (B. Baesens).

http://dx.doi.org/10.1016/j.ins.2017.04.015
0020-0255/

may classify all instances to the majority class, resulting in high overall accuracy but unacceptably low precision with respect to the minority class of interest. For instance, when a model is trained on a data set with 1% of instances from the minority class, a 99% accuracy rate can be achieved by simply classifying all instances as belonging to the majority class. Indeed, the issue of learning on imbalanced data sets is considered to be one of the ten challenging problems in data mining research.

In order to solve the problem of learning from imbalanced data sets, many solutions have been proposed in the past few years. Most well-known proposed solutions roughly fall into three groups: data-level, algorithm-level and ensemble solutions. Data-level solutions apply resampling as a preprocessing step to reduce the negative effect caused by class imbalance. Algorithm-level solutions aim to develop new algorithms or modify existing ones to bias learning towards the minority class. Ensemble solutions either modify the ensemble learning algorithms at the data-level to preprocess the data before the learning stage of base classifiers or embed a cost-sensitive framework in the ensemble learning process. Each type of solution has pros and cons, without a clear consensus on what makes for the best solution to address the problem of class imbalance in churn prediction.

In this paper, our aim is to experimentally compare state-of-the-art methods for dealing with class imbalance on real-world churn prediction data sets. Our contributions are twofold: first, we launch a large scale benchmark study of 21 techniques across 11 real churn data sets. So far, no such exhaustive benchmark comparison was performed in the context of churn prediction. Several techniques are assessed for the first time in this field. Second, we include a recently-developed profit-based measure called Expected Profit Measure (EMP) as one of the performance measures in the experiment. As pointed out by Raeder [30], the choice of evaluation metrics plays an important role in imbalanced learning. Traditionally, the area under the receiver operating curve (AUC) and top-decile lift are widely used in churn prediction [25]. However, both measures do not take the real costs and benefits brought by the churn model into account. Meanwhile, the EMP measure is developed from a cost-benefit perspective by Verbraken et al. [43]. It combines the outputs of churn prediction models with the profits brought by the retention campaign. Therefore, our study tries to explore the performance of techniques dealing with class imbalance by using the EMP measure together with the traditional AUC and top-decile lift measure. Experimental results show that the used evaluation metrics have a great impact on the performance of the techniques. We offer an in-depth exploration of the sensitivity of each technique with respect to each evaluation measure. We find that there is still much room to improve the techniques handling class imbalance in terms of the profit-based measure in churn prediction.

The remainder of this paper is organized as follows. In Section 2, we briefly review churn prediction research and describe performance measures used in this field. In Section 3, we discuss the solutions to class imbalance. Section 4 presents the detailed experimental methodology and Section 5 analyzes the experimental results. Finally, Section 6 gives conclusions and directions for further research.

2. Churn prediction

2.1. Churn prediction modeling

Churn prediction aims at identifying potential churning customers based on past information and prior behavior and offering them some incentives to stay. Neslin et al. [29] suggested that prediction techniques have a huge impact on the returns of subsequent retention actions and that a small improvement in a churn prediction model can lead to a significant increase in profits. As a result, the optimization of churn prediction models has been explored extensively in the last few years.

The quality of a churn prediction model is determined by both data and learning algorithm. Therefore, current studies focus on these two aspects to optimize the model performance. From the aspect of learning algorithms, customer churn prediction involves binary classification and numerous classification algorithms from the research fields of machine learning and statistics have been adopted such as logistic regression, decision trees, neural networks, support vector machines, partial least squares(PLS) and hazard models. Due to the ease of use and interpretability, logistic regression and decision trees are commonly used by academics and practitioners [29]. However, more advanced models also demonstrated their predictive power with the increase of model complexity. For example, Keramati et al. [22] used decision trees, artificial neural networks, and k-nearest neighbor for predicting customer churn in a telecommunication company, where the results showed the superiority of artificial neural networks. Gordini and Veglio [17] developed a churn prediction model tailored for B2B e-commerce industry based on support vector machines (SVM). Their experimental results showed SVM outperforms logistic regression and neural networks.

All the works listed above implement the prediction model as a single model. Recently, researchers have also explored the usage of so-called ensemble or hybrid learning techniques in customer churn prediction, which have demonstrated substantial improvement over a single classification model. For example, stochastic gradient boosting, and bagging models have been used by Lemmens and Croux [25] to model churn for a wireless telecommunications company, which showed the two ensemble methods obtain substantial better results than logistic regression. Tsai and Lu [36] tested a hybrid data mining technique which combines back-propagation artificial neural networks (ANN) and self-organizing maps in telecom churn prediction, and found that the hybrid models outperform the single neural network baseline model in terms of prediction accuracy. Although various algorithms have been applied, there appears no single dominating approach which is useful in every churn-related context.

The second research stream puts the emphasis on the data used to build the churn model. Some researchers tried to investigate the usefulness and representation of certain types of data information. For instance, Zhang et al. [39] investigated the effect of interpersonal influence information on the accuracy of churn prediction, and combined it with customers' personalized characters to produce a new prediction model. Chen et al. [7] integrated static customer data and longitudinal behavioral data to improve the performance of prediction models. Other researchers investigated the impact of data preprocessing on the model predictive accuracy. Coussement et al. [10] examined data-preparation alternatives to enhance the prediction performance in a churn prediction modeling context. Their study showed the data-preparation technique actually affects prediction performance. All these data-oriented studies indicate that by carefully checking the data characteristics and developing corresponding techniques the churn prediction model can significantly improve. However, the research in this field is still under-developed. The research in this paper falls into the second stream.

2.2. Evaluation metrics in churn prediction

Evaluating model performance is a key step in successful churn prediction modeling. Traditionally, AUC and top-decile lift are used as the main performance measures.

Receiver operating characteristic (ROC) curve analysis is one of the popular approaches that can be used to measure the performance of classifiers on imbalanced data sets. ROC analysis has its origin in signal detection theory as a method to choose a threshold or an operating point for the receiver to detect the presence or absence of a signal. The ROC graph plots true positive rates versus false positive rates. Classifiers can be selected based on their tradeoffs between true positives and false positives. The AUC can be expressed as follows:

$$AUC = \int_0^1 TPR(t)dFPR(t) \tag{1}$$

where t is the threshold, TPR is true positive rate and FPR is the false positive rate. Rather than visually comparing curves, the area under the ROC curve metric (AUC) aggregates the performance of a classification model into a single number, which makes it easier to compare the overall performance of multiple classification models. A random classifier has an AUC of 0.5 and a perfect classifier possesses an AUC equal to 1.

Top-decile lift is another widely used performance measure in churn prediction. To calculate top-decile lift, customer instances are first sorted based on the churn propensity scores obtained by the prediction model in a descending order. Then, the top-decile lift is computed as the ratio of the percentage of correctly classified churners (the minority class) in the top 10% ranked cases ($\beta_{10\%}$) and the percentage of actual churners in the entire data set as follows:

$$top-decile \quad lift = \beta_{10\%}/\beta_0 \tag{2}$$

For instance, a top-decile lift value of 2 indicates that the classifier identifies twice as many churners in the top 10% ranked customer group as a random classifier would do.

Although AUC and top-decile lift are two widely used performance measures in churn prediction, both measures do not take the real cost/benefit brought by churn models and consequent retention campaigns into account. To overcome the ignorance of the cost-benefit factor, profit-based measures were recently proposed by Verbraken et al. [43] and Verbeke et al. [42]. Before giving the description of these measures, let us outline the dynamical process of customer churn and retention programs as presented in Fig. 1. As the figure shows, after building a churn prediction model, the firm will first apply the model to its current customer base. Then, a fraction α of customers with the highest churn propensities are targeted and offered some incentives. There are true and false would-be churners within the target customers. In the true would-be churner group, a fraction γ of customers will accept the offer and stay active, whereas the remaining fraction $1-\gamma$ will still defect. In the false would-be churner group, the assumption here is that all customers will accept the incentives and that they do not churn because they had no intention to leave. Meanwhile, in the $1-\alpha$ fraction (the non-target group), all would-be churners will leave whereas non-would-be churners will stay. We can see that the profit of launching a retention program is influenced by customers' churn propensities in the target fraction, their probabilities of accepting incentive offers, the cost of the incentive offer, customer lifetime values and so on. Neslin et al. [29] established the following expression to get the total profit for a retention campaign:

$$P = N\alpha[\beta\gamma(CLV - c - \delta) + \beta(1-\gamma)(-c) + (1-\beta)(-c-\delta)] - A \tag{3}$$

where P is the profit of the retention campaign, N is the number of customers in the current customer base, α is the fraction of the target customers in the retention campaign, β is the portion of churners within the targeted customers, γ is the probability of would-be churners accepting the offer and staying in the company, CLV is the average customer lifetime value of retained customers, c is the cost of the incentive when a customer accepts the offer, δ is the cost of contacting a customer to offer the incentive, and A is the fixed administrative cost of running a retention program. The first term in the bracket of Eq. (3) reflects the revenue contributed by the retained potential churners. The second term presents the loss coming from the identified potential customers who do not accept the offer and finally defect. The last term in the bracket is the loss of sending incentives to the customers who are actually non-churners. This equation can be also rewritten as follows:

$$P = N\alpha[(\gamma CLV + \delta(1-\gamma))\beta_0\lambda - \delta - c] \tag{4}$$

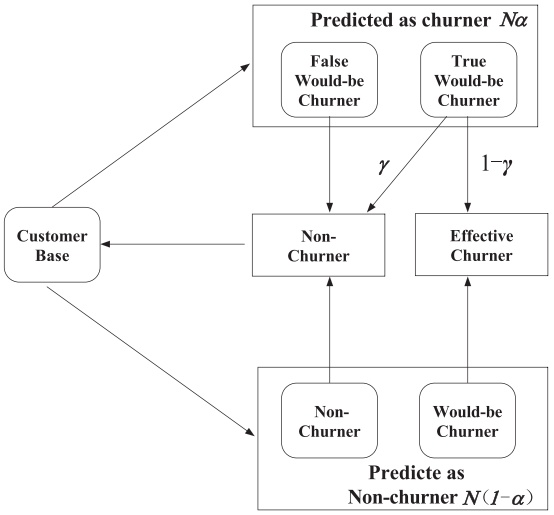

Fig. 1. The dynamic process of customer churn and retention.

where β_0 is churn rate in the whole customer base, λ is the lift corresponding to the target fraction α. Based on Eq. (4), Verbeke et al. [42] presented the maximum profit (MP) criterion which is defined as follows:

$$MP = \max_{\alpha}(N\alpha[(\gamma CLV + \delta(1-\gamma))\beta_0\lambda - \delta - c]) \tag{5}$$

By selecting the optimal target size $N\alpha$, the MP measure will give the maximum profit that a retention campaign can achieve given the churn prediction model. In the MP measure, all parameters except target size are supposed to be known. In practice, customer lifetime value *CLV*, the cost of the incentive δ and the contacting cost ϕ can be estimated with sufficient reliability. However, estimating the probability of a churner accepting the retention offer is not a trivial task. Therefore, Verbraken et al. [43] adopted a probability distribution for γ and presented a probabilistic profit-based performance measure, i.e. the expected maximum profit measure (EMP) for a churn model as follows:

$$EMP = \int_{\gamma} P_c(T(\gamma); CLV, \delta, \phi) \cdot h(\gamma) d\gamma \tag{6}$$

where $h(\gamma)$ is the probability density function for γ. Given a γ value, $T(\gamma)$ is the optimal threshold and $P_c(T(\gamma); CLV, \delta, \phi)$ is the maximum profit calculated as in Eq. (5). γ is assumed to follow a Beta distributed random variable $B(\alpha, \beta)$ with two parameters α and β. More details about the empirical calculation of the EMP measure can be found in [43]. Both MP and EMP measures have been successfully used in churn prediction [42,43] and it has been shown that using the two profit-based measures to select churn prediction models will provide a significant increase in profits compared to AUC and top-decile lift. By taking into account the uncertainty of the classification costs and benefits, the EMP measure provides a more general form to measure the expected profits produced by churn models and we will use it as one of the main measures in our experimental study.

3. Class imbalance

Class imbalance is an important characteristic of many data sets used for churn prediction modeling. Most standard classification algorithms frequently exhibit a bias toward the majority class, so class imbalance often hinders the performance of standard classifiers, especially in terms of the minority class of interest. A great variety of solutions have been proposed to address this issue, which can be roughly divided into three categories: data-level, algorithm-level and ensemble solutions. We will give a brief description for each type of solution in the following subsections.

3.1. Data-level solution

Data-level solutions try to re-balance the class distribution by resampling the original, unbalanced data set. Data-level solutions consist of many different forms of resampling techniques. Random sampling techniques offer the simplest approaches, which include random oversampling (ROS) and random undersampling (RUS). ROS tries to create a superset of the original data set by randomly replicating the minority instances from the existing data set. RUS aims to balance the class distribution through random elimination of majority class examples. Both techniques are easy to use and understand, though RUS can discard potentially useful data instances whereas ROS can increase the likelihood of overfitting by making exact copies of existing instances. In order to deal with these drawbacks, some intelligent sampling methods have been proposed,

of which synthetic minority oversampling technique (SMOTE) [5] is probably the most popular one. SMOTE first randomly selects several nearest neighbors of a minority class instance and produces new instances based on linear interpolations between the original examples and the randomly selected nearest neighbors. Although SMOTE is well acknowledged in the academic community, it still suffers from some drawbacks. For instance, SMOTE generates the same number of synthetic minority instances for each original minority example and does not take neighboring examples belonging to the majority class into consideration, which can result in an increase of overlap between the classes. To overcome the shortcomings of SMOTE, some variants have been developed. Adaptive synthetic sampling (ADASYN) uses the density distribution to decide on the number of synthetic samples that need to be generated for each minority example [18]. ADASYN first finds the nearest neighbors for each minority instance and then counts the number of majority examples in these neighbor instances to calculate a ratio. The number of instances generated for each instance is then decided by this ratio. Recently, a new intelligent sampling method, called majority weighted minority oversampling technique (MWMOTE) was also presented [2]. MWMOTE first identifies the hard-to-learn informative minority class samples and assigns them weights according to the distance from the nearest majority class samples. It then generates synthetic samples from the weighted informative minority samples using a clustering approach.

Some intelligent methods for undersampling have also been proposed. Kubat and Matwin [23] proposed the one-sided selection (OSS) technique, which selectively removes the majority instances that either are redundant or borderline majority examples. The cluster-based undersampling algorithm (CLUS) presented by Yen et al. [38] organizes training data into groups with homogeneous characteristics and then downsizes the number of majority class samples in each cluster.

There are also several hybrid sampling techniques. Some of them combine oversampling with data cleaning techniques to reduce the overlapping introduced by oversampling methods. Typical examples are SMOTE-Tomek and SMOTE-ENN [3]. SMOTE-Tomek finds pairs of minimally distanced nearest neighbors of opposite classes. When two samples form a Tomek link, either one of them is noisy or both samples are borderline instances. SMOTE-Tomek deletes Tomek links after SMOTE sampling so that all minimally distanced nearest neighbor pairs are in the same class. SMOTE-ENN is similar to SMOTE-Tomek, but it uses Wilson's edited nearest neighbor (ENN) rule to remove instances after SMOTE sampling. That is, any example that is misclassified by its three nearest neighbors is removed from the data set. Apart from the usage of data cleaning techniques, some researchers presented hybrid sampling methods that apply computational intelligence technologies to identify useful instances. For instance, García and Herrera [14] proposed a method called evolutionary under-sampling, which uses evolutionary algorithms to select prototype instances.

3.2. Algorithm-level solution

This category tries to strengthen the learning of classification algorithms with regards to the minority class. The most popular branch in this group is cost-sensitive learning. Cost-sensitive methods assign different misclassification costs for different classes, generally a high cost for the minority class and a low cost for the majority class. The cost-sensitive learning process then tries to minimize the total cost of misclassification. So far, many cost-sensitive learning methods have been proposed to deal with class imbalance. For instance, Ting presented an instance-weighting method to induce cost-sensitive trees [37]. Veropoulos et al. proposed a biased support vector machine algorithm by setting different costs to the majority and minority class in the objective function [44].

Other algorithm-level solutions include one-class learning and active learning. One-class learning aims to recognize instances of a concept by using only a single class of examples rather than differentiating between instances of both classes. One representative type of one-class learners are one-class SVMs [33]. Active learning methods were originally developed to solve problems related to unlabeled training data. Recently, it has found increasing usage in imbalanced learning applications. For example, Ertekin et al. proposed an SVM-based active learning method which queries a small pool of data at each iterative step of active learning instead of using the entire data set [12]. In this paper, we choose the most popular cost-sensitive learning as the representative technique of this group.

3.3. Ensemble solution

Ensemble solutions have become popular in recent years. A good review of ensemble solution for learning with imbalanced classes can be found in [16]. Ensemble learning tries to improve the performance of single classifiers by inducing several classifiers and combining them to obtain a new, better performing classifier. To address the class imbalance issue, ensemble solutions combine ensemble learning with the above mentioned data or algorithmic level approaches. Most ensemble solutions are based on known strategies such as bagging, boosting, random forests and their hybrids.

1. **Bagging-based ensemble**: Bagging-based ensembles integrate bagging with data sampling techniques. They collect the training samples and use different sampling methods to generate training sets and combine classifiers trained for each training set. For example, Underbagging and Overbagging [45] use random undersampling or oversampling so that the class distribution is balanced when building each model in the ensemble. Roughly Balance Bagging [19] is also based on undersampling, but the size of the majority examples is determined probabilistically according to a negative binomial distribution, whereas the number of minority examples is always the same. In SMOTEBagging [45], each training data set is composed of a bootstrap sample taken from the majority class, and a sample of the minority class created through a combination of oversampling and SMOTE.

2. **Boosting-based ensemble**: Boosting-based ensembles include techniques that embed data preprocessing or cost-sensitive learning in a boosting procedure. Cost-sensitive boosting keeps the general learning framework of boosting algorithms, but introduces cost items into the weight update formula. Well-known cost-sensitive boosting techniques are AdaC1, AdaC2, AdaC3, CSB1 and CSB2 [34]. These methods usually differ in the way that they modify the weight update rule. Data-preprocessing-based boosting exploits sampling to bias the data distribution towards the minority class before the classifier generation step. For example, Chawla et al. proposed an approach called SMOTEBoost by the combination of SMOTE sampling and a boosting procedure [6], in which SMOTE sampling is used before computing the new example weights in each iteration. Seiffert et al. [32] presented a different ensemble method named RUSBoost, which removes instances from the majority class by random undersampling in each iteration of boosting.
3. **Random forest-based ensemble**: Chen et al. [8] proposed the balanced random forest (BRF) and weighted random forest (WRF) models to learn from imbalanced data. The idea of BRF is to combine undersampling with a random forest approach. For each repetition, the same number of instances from the minority class and the majority class are randomly drawn with replacement, respectively. WRF follows the idea of cost-sensitive learning, where a heavier penalty is placed on the node-splitting condition and terminal-node class weights for misclassifying the minority class.
4. **Hybrid ensemble**: Representative techniques in this category include EasyEnsemble and BalanceCascade [26], which are hybrid approaches that combine bagging and boosting. More specifically, EasyEnsemble uses bagging as the main ensemble learning method. Instead of training a single classifier in each bag , Adaboost is used to train on a balanced dataset formed by random undersampling. BalanceCascade works in a sequentially supervised manner. In each bagging iteration after learning the AdaBoost classifier, the majority class examples that are correctly classified by the current Adaboost classifier are permanently removed from the data set.

Some other ensemble solutions based on ensemble selection or ensemble pruning are also explored. One typical examples are the ordering-based ensemble pruning proposed by Galar et al. [15], which refine the ensemble by choosing the best ensemble members. However, the above mentioned four types of ensemble solutions are still considered the state-of-the-art methods.

3.4. Summary and comments

Although various solutions have been proposed, each type of solution has pros and cons. Data-level approaches are simple and easy to use, but shortcomings are also obvious. Sampling methods alter the original class distribution of imbalanced data which may lead to some unexpected issues [35]. Moreover, the optimal class distribution of the training data is usually unknown [34]. The algorithm-level solutions are specific to a certain classification algorithm and are hence often sensitive to the context in which they are applied. To develop an algorithm level solution, one needs to possess extensive knowledge about both the learning algorithm and application domain. Ensemble solutions embed either data-level or algorithm-level solutions in the ensemble learning framework. Therefore, they will inherit the drawbacks of the two solutions. Moreover, ensemble solutions also bring extra model complexity and computational cost.

In order to figure out better solutions for dealing with class imbalance, many comparative studies have been done. Van Hulse et al. [41] presented a comprehensive experiment with eight sampling methods. It was found that random sampling methods perform better than intelligent sampling methods like SMOTE. García et al. [13] investigated the influence of both imbalance ratio and classifier on several resampling strategies to deal with imbalanced data sets. Experiments showed that oversampling consistently outperforms undersampling when data sets are strongly imbalanced, whereas there are no significant differences on data sets with a low imbalance. Khoshgoftaar et al. [24] compared boosting with bagging techniques in the context of noisy and imbalanced data. The experiments showed that bagging techniques generally outperform boosting in noisy data environments. Galar et al. [16] developed a thorough empirical comparison by the consideration of a wide range of ensembles, from simple modifications of bagging and boosting to cost-sensitive or hybrid approaches. Their main conclusions state that SMOTEbagging, RUBoost and UnderBagging give the best AUC results. López et al. [27] analyzed the performance of sampling techniques against cost-sensitive learning in the framework of imbalanced learning. The results showed that there are no differences among them. López et al. [28] carried out an experimental study to contrast sampling, cost-sensitive learning and ensemble techniques. The results highlighted the dominance of ensemble approaches UnderBagging and SMOTEBagging when C4.5 and k-NN are used as weak classifiers while the best results are achieved by SMOTE sampling and cost-sensitive learning when SVM is considered. Other comparative studies can also be found in [31]. To summarize the above comparative studies, we find that conclusions about the performance of different solutions do not reach a clear consensus.

In churn prediction, the solutions to deal with class imbalance do not go beyond the scope of data-level, algorithm-level and ensemble solution. For example, Chen et al. [7] used random undersampling as a data-preprocessing strategy to churn prediction. SMOTE sampling is used by Ali and Aritürk [1] in their dynamic modeling framework. Idris et al. [20] investigated the significance of a particle swarm optimization (PSO) based undersampling method to handle the imbalanced data distribution. Xie et al. [46] proposed the improved balanced random forests approach, which combines balanced random forests and weighted random forests.

Table 1
Summary of churn data sets used in the experiments.

Dataset	Abbr.	Source	Region	#Obs.	#Att.	Churn rate (%)
Chile	Chile	Operator	South American	5300	41	5.66
Duke_current	Duke1	Duke	North American	51,306	173	1.80
Duke_future	Duke2	Duke	North American	100,462	173	1.80
KDDcup	KDDcup	KDD CUP 2009	Europe	50,000	231	7.34
Korean1	K1	Operator	East Asia	2019	10	3.96
Korean2	K2	Operator	East Asia	2941	14	4.42
Korean3	K3	Operator	East Asia	5990	36	4.34
Korean4	K4	Operator	East Asia	2183	9	4.58
Korean5	K5	Operator	East Asia	26,224	11	4.19
Tele1	Tele1	Operator	Europe	4350	87	8.05
UCI	UCI	UCI ML repository	–	3333	19	14.5

4. Experimental framework

Since model performance on imbalanced data sets is influenced by many factors, we argue that it is not feasible to determine which technique is best suited in any general context. Therefore, we narrow our contribution and scope to the churn prediction domain. Burez and Van den Poel [4] have performed some pioneering studies to compare different solutions for churn prediction. But only two sampling techniques and one cost-sensitive ensemble learning technique were considered. Besides, they only consider AUC like most previous studies.

In this paper, we will conduct a more comprehensive experimental study and present an in-depth exploration with the domain-specific profit-based measure together with the commonly used AUC measure. Different from our previous research in [40], which only compares the data-level solutions and focuses on the MP measure, we have also included the algorithm-level and ensemble solutions and consider the more general EMP measure. To summarize, we aim at answering the following research questions:

RQ1 Do different evaluation metrics have similar influence on the performance of different solutions?
RQ2 Does each solution truly improve the performance in comparison with the original technique without considering class imbalance?
RQ3 What is the best technique for each given evaluation measure, respectively?

In this section, we present the framework that will be used to carry out the experiments. First, we provide details of the real-world customer data sets used in Section 4.1. Then, we give a brief summary of all the techniques used in the experiment as well as other experimental settings in Section 4.2. Finally, Section 4.3 presents the statistical tests used in the experimental analysis.

4.1. Data sets

The eleven real-world data sets used in the experiments stem from the telecommunication industry. Table 1 summarizes the main characteristics of the data sets, where each column presents name, abbreviation (Abbr.), data source, region, number of observation (#Obs.), attributes (#Att.) and churn rate. As Table 1 shows, the churn rates range from 1.8% to 14.5%, and most of them are highly imbalanced.

Two steps are performed for data preparation before proceeding with model construction. The first step involves dealing with missing values. In case of categorical variables, the missing values are encoded as a new level. For continuous variables, if more than 50% of the values are missing, the variable is removed. Otherwise, median imputation is used to avoid the influence of extreme values. In order to avoid the "curse of dimensionality" problem, a feature selection procedure was conducted as the second step for data preparation. More specifically, a feature selection approach based on the Fisher score is used on five data sets: Chile, Duke1, Duke2, KDD and Tele1. The following formula is applied to calculate the Fisher score for each variable:

$$Fisher\ Score = \frac{|\bar{x}_c - \bar{x}_{nc}|}{\sqrt{s_c^2 + s_{nc}^2}} \tag{7}$$

where $\bar{x}_c$ and $\bar{x}_{nc}$ are the mean value, and s_c^2 and s_{Nc}^2 are the variances of variable for the churners and the non-churners, respectively. Features with highest Fisher scores on the five above mentioned data sets are retained and the number of variables in these data sets is reduced to 30 as including more variables does not necessarily increase performance.

4.2. Experimental settings

A five times two-fold cross-validation strategy was applied in our study as follows: each original data set was randomly split into two equally-sized parts. First, one part was used as training data to the build model. The parameter tuning is

Table 2
Techniques used in the experimental study.

Methods	Description	Parameters
(a) Sampling solution		
AYDYSN	Adaptive synthetic sampling [18]	Number of nearest neighbors $K = 5$
CLUS	Cluster-based undersampling algorithm [38]	Number of clusters $K = 3$
MWMOTE	Majority weighted minority oversampling technique [2]	$k1$=5, $k2$=3, $k3 = \|S_{min}\|/2$ C_p=3, $C_f(th)$=5, $CMAX$=2
ROS	Random oversampling	–
RUS	Random undersampliing	–
SMOTE	Synthetic minority oversampling [5]	Number of nearest neighbors $K = 5$
SMOTE-ENN	SMOTE sampling with ENN cleaning [3]	Number of nearest neighbors in SMOTE $K = 5$
SMOTE-Tomek	SMOTE sampling with Tomek-link cleaning [3]	Number of nearest neighbors in SMOTE $K = 5$
(b) Cost-sensitive solution		
CS-Tree	Cost-sensitive trees based on in instance-weighting [37]	$C_{non-churner} = 11$, $C_{churner} = 56$
CS-SVM	Cost-sensitive support vector machine [44]	$C_{non-churner} = 11$, $C_{churner} = 56$
(c) Ensemble solution		
Bagging	Bagging	Number of bags $N = 40$
OverBagging	OverBagging [45]	Number of bags $N = 40$
RBBagging	Roughly balanced bagging [19]	Number of bags $N = 40$
SMOTEBagging	SMOTEBagging[45]	Number of bags $N = 40$ Number of nearest neighbors in SMOTE $K = 5$
UnderBagging	UnderBagging [45]	Number of bags $N = 40$
AdaBoost	AdaBoost	Number of iterations $T = 40$
AdaC2	AdaC2 [34]	Number of iterations $T = 40$ $C_{non-churner} = 11$, $C_{churner} = 56$
RUSBoost	RUSBoost [32]	Number of iterations $T = 40$
SMOTEBoost	SMOTEBoost [6]	Number of iterations $T = 40$
RF	Random forest	Number of trees $N = 40$
BRF	Balanced random forest [8]	Number of trees $N = 40$
WRF	weighted random forest [8]	Number of trees $N = 40$ $C_{non-churner} = 11$, $C_{churner} = 56$
EasyEnsemble	EasyEnsemble[26]	Number of subsets $S = 4$ Number of iterations in AdaBoost $T = 10$
BalanceCascade	BalanceCascade[26]	Number of subsets $S = 4$ Number of iterations in AdaBoost $T = 10$

also carried out on the training data set by using a repeated holdout validation. The other part acted as independent test data to calculate the value of performance measure. Then, the two parts switched their roles. This process was repeated five times and values of performance measures on the test data sets were averaged to get the final estimates. We selected AUC, top-decile lift and expected maximum profit (EMP) to measure model performance. To calculate the EMP measure, the parameters *CLV*, δ, and c in Eq. (6) were set to be 200, 10 , and 1, respectively. The two parameter in $B(\alpha, \beta)$ were set to be 6 and 14. The parameter settings were based on previous scientific literature [21,29,43] and discussion with data scientists in the telecommunication industry. Twenty-one techniques were chosen from the data-level, algorithm-level and ensemble solutions. All the selected methods have been listed in Table 2 together with their short description and parameter settings. Most parameter values were selected according to the recommendation of the corresponding authors of each algorithm. As far as we know, many solutions are applied here for the first time in a churn prediction task, like SMOTE-ENN, UnderBaging and EasyEnsemble. The experiments were conducted using the open-source R language.

Eight sampling methods were chosen as the representatives of data-level solutions. Sampling rate is an important parameter for all sampling methods. Our previous research has found that roughly balanced strategy results in universal good performance, in which the final class distributions were set to 1:3 (minority class vs majority class) [40]. Experimental results of sampling solutions were based on that configuration. For SMOTE-ENN and SMOTE-Tomek, one cannot control the exact class distributions, so we apply the same parameter setting as the one that lets SMOTE reach less balanced class ratios (1:3). For CLUS sampling, we have experimented with several values for the number of clusters $K \in \{3, 5, 7\}$ with similar results, so that the one requiring the minimum computational cost $K = 3$ was selected. To investigate the impact of sampling methods, the following two classifiers were used as baselines:

1. **C4.5 decision tree**: We have set the confidence level to 0.25, the minimum number of instances per leaf was set to 2 and pruning was used to obtain the final tree.
2. **Support vector machine (SVM)**: With a radial basis function kernel (RBF), with heuristic search of the optimal kernel width σ and the regularization term set to 1.0.

For algorithm-level solutions, cost-sensitive learning methods were selected as the representatives. Two cost-sensitive learning methods were used in the experiments, the instance-weighting based cost-sensitive tree (CS-Tree) proposed by Ting [37] and the cost-sensitive support vector machine (CS-SVM) proposed by Veropoulos et al. [44]. The following costs

were applied: misclassifying a non-churner as a churner leads to an average loss equal to the incentive offer value and the contact cost. Meanwhile, misclassifying a churner as a non-churner leads to an average loss equal to the expected net customer value minus contact cost. So, we set misclassification costs for non-churners $C_{non-churner} = \delta + c = 11$ and churners as $C_{churn} = E(\gamma(CLV - \delta) - c) = 56$. Other parameters were kept to use the same settings as in the baseline classifiers.

Four types of ensemble solutions were considered in our study: bagging-based, boosting-based, random-forest-based and hybrid ensemble. Eleven techniques were considered in the experiments and we also included the original Bagging, AdaBoost and random forest algorithm to make the comparison. In all ensemble techniques, C4.5 decision tree was used as a base classifier, since most previous ensemble methodologies were proposed in combination with C4.5, with the settings kept equal to those as used in the base classifiers. We set the number of bags in bagging-based ensembles, the number of iterations in boosting-based ensembles and the number of trees in random-forest-based ensembles to be 40. For hybrid ensemble methods EasyEnsemble and BalanceCascade, 4 bags and 10 iterations were considered. These settings were consistent with previous research [16,26] and our experiments also show that including more classifiers does not achieve better results. As noted above, there is a wide variety of cost-sensitive boosting techniques. Previous empirical studies showed that the AdaC2 algorithm stands out with respect to the others [34]. To avoid the repetition of similar experiments, we only empirically studied this algorithm from the set of cost-sensitive boosting techniques. In WRF and AdaC2, the misclassification costs were set to be the same as cost-sensitive solutions.

4.3. Statistical tests

Following the recommendation of Demšar [11], two types of statistical tests were considered for pairwise comparison and multiple comparison, respectively. When only two algorithms were compared, a Wilcoxon signed-rank test was used. For multiple comparisons, the Friedman test with Iman-Davenport extension was applied. Iman-Davenport is based on the average ranks of the algorithms over multiple data sets. To get the average ranks, all algorithms were sorted from best to worst on each data set, where the best method obtained rank 1, the second received rank 2 and so on. Then the ranks were averaged across all data sets. Given N data sets, k algorithms and average ranks $R_i, i = 1, ..., k$, the Iman-Davenport test calculates a F-distribution statistic with $k-1$ and $(k-1)(N-1)$ degrees of freedom to validate the null hypothesis as follows:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \tag{8}$$

where

$$\chi_F^2 = \frac{12N}{k(k+1)}[\sum_j R_i^2 - \frac{k(k+1)^2}{4}] \tag{9}$$

If the null hypothesis that all methods have equivalent performance was rejected, a further post-hoc test should be applied to point out where the differences lay. In our study, we applied Holm's post-hoc test to do so, as it can appropriately control the family-wise error rate (FWER) and hence is more powerful than Bonferroni-Dunn's test in a rigorous comparison [11]. Holm's test first selects the algorithm with the lowest mean rank as a control method, and then uses a step-down procedure to compare the control method with the other ones. The Holm's test utilizes the Z statistic as follows:

$$Z = \frac{(R_i - R_j)}{\sqrt{k(k+1)/(6*N)}} \tag{10}$$

where R_i and R_j are the average ranks of methods i and j. After using the Z statistic to find the corresponding probability from the normal distribution table, Holm's test orders p-values by $p_1 \le p_2 \le p_3 ... \le p_{k-1}$. Starting from the most significant p value, Holm's procedure compares each p_i with $\alpha/(k-i)(i = 1, 2, ..., k-1)$, where α is the given significance level. If p_1 is below $\alpha/(k-1)$, the corresponding hypothesis is rejected and we are allowed to compare p_2 with $\alpha/(k-2)$. If the second hypothesis is rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all the remaining hypotheses are retained as well. In all statistical tests, we fixed the significance level $\alpha = 0.05$

5. Experimental results

In this section, we offer the analysis of the experimental results. Interested readers can refer to the full experimental results in the Appendix. To answer the three research question in Section 4, we divide this section into five subsections. In the first three subsections, we will do intra-family analysis to investigate the results within each group of solution, separately. In the fourth subsection, we will perform a complete global comparison among the representatives coming from each family to find better performing techniques. In each subsection, we will explore behavior patterns of different techniques with respect to each evaluation measure. Finally, the fifth subsection summarizes the main findings from the experimental results.

Table 3
Experimental performance of sampling techniques.

Sampling	C4.5			SVM		
	AUC (Rank)	Top-decile lift (Rank)	EMP (Rank)	AUC (Rank)	Top-decile lift (Rank)	EMP (Rank)
None	0.6557 (5.8182)	4.5330 (**2.7273**)	1.0359 (4.9091)	0.6756 (8.6364)	2.7039 (7.9091)	0.7233 (6.0000)
ADASYN	0.6899 (4.4091)	3.8553 (5.6818)	1.0542 (4.4545)	0.7264 (4.8182)	3.3536 (4.5000)	0.8858 (5.0000)
CLUS	0.7255 (3.0909)	3.6925 (6.3636)	0.9809 (5.9091)	0.7265 (5.9091)	3.2224 (5.9545)	0.8407 (6.2727)
MWMOTE	0.6715 (6.5455)	3.7751 (6.0000)	1.0233 (6.2727)	0.7279 (4.1818)	3.3799 (4.4091)	0.8898 (5.3636)
ROS	0.6464 (8.1818)	4.2573 (5.6364)	0.9958 (6.5455)	0.7326 (**3.3636**)	3.4144 (4.1818)	0.8918 (4.7273)
RUS	0.7322 (**2.5455**)	3.8315 (4.5909)	1.0590 (**3.9091**)	0.7270 (5.0909)	3.3110 (4.9091)	0.8712 (**3.9091**)
SMOTE	0.6909 (4.0455)	3.8766 (4.5000)	1.0688 (4.0909)	0.7269 (5.1818)	3.3840 (4.9545)	0.8928 (4.9091)
SMOTE-ENN	0.6923 (5.5455)	3.9875 (4.7273)	1.0707 (4.6364)	0.7335 (3.5455)	3.3831 (4.2727)	0.8943 (4.6364)
SMOTE-TL	0.6885 (4.8182)	3.8906 (4.7727)	1.0668 (4.2727)	0.7297 (4.2727)	3.4136 (**3.9091**)	0.8959 (4.1818)
Iman-Davenport	$p < 0.0001$	$p = 0.0760$	$p = 0.0954$	$p < 0.0001$	$p = 0.0136$	$p = 0.5414$

Table 4
Experimental performance of cost-sensitive techniques.

Sampling	C4.5			SVM		
	AUC	Top-decile lift	EMP	AUC	Top-decile lift	EMP
None	0.6557	4.5330	1.0359	0.6756	2.7039	0.7233
Cost-sensitive	0.6780	5.2925	1.0364	0.7255	3.3549	0.8864
Wilcoxon test	$p = 0.4235$	$p = 0.4235$	$p = 0.959$	$p = 0.0076$	$p = 0.0033$	$p = 0.1549$

5.1. Comparison of sampling methods

We start our investigation by comparing different sampling solutions. Table 3 shows the performance of different sampling techniques in combination with two baseline classifiers. Apart from the sampling methods, we also include a non-sampling strategy in the experiments to analyze whether the use of sampling is beneficial, which is denoted as "none". Each entry in the table gives the mean value of the three metrics as well mean ranks over eleven data sets and the last row shows the p-values of the Iman-Davenport test. The best rank in each column is in bold and the ranks that are significantly worse than the best one at the 5% significance level are underlined based on Holm's post-hoc test. The results show that sampling methods indeed exhibit different behavior with different evaluation measures, and that this is interrelated with the classifier used.

With respect to C4.5, the effectiveness of sampling methods depends on the measures as shown in Table 3. In terms of AUC and EMP measure, most sampling methods have a positive impact on the results and RUS has the best results for both measures. On the contrary, the non-sampling strategy comes first when using top-decile lift, which means sampling techniques have a negative impact. According to Holm's test, RUS is significantly better than the non-sampling strategy and the other two sampling methods at the 5% significance level with regards to AUC. Meanwhile, the non-sampling strategy has a statistically better performance than the two other sampling methods with regards to top-decile lift. There is no statistical difference with respect to EMP as shown by the Iman-Davenport test. So when you want to build a decision tree model for churn prediction, which sampling procedure to use depends upon the performance measure adopted. When AUC and EMP are used, RUS is recommended. However, the sampling solution is useless to improve top-decile lift.

Different from C4.5, most sampling techniques work well with SVM classifier across different measures. As demonstrated in Table 3, the non-sampling strategy stays at the bottom part of the ranking list. Holm's test shows that non-sampling is also statistically worse than the top ranking methods for both AUC and top-decile lift. For the EMP measure, there is no statistical difference between the different methods according to the Iman-Davenport test. We further investigate the results of each individual measure. ROS demonstrates its excellence in terms of AUC and top-decile lift, ranking first and second for the two measures, respectively. It is interesting to note that RUS is also the best performing sampling technique when using EMP. The results indicate that sampling solutions can be used for SVMs without any hesitation.

5.2. Comparison of cost-sensitive learning

This subsection discusses the results of cost-sensitive solutions. Table 4 shows the average performance of cost-sensitive solutions in comparison to using no sampling method and baseline classifiers that assume equal misclassification costs ("None"). As before, each entry in the table presents the mean value of the three evaluation metrics and the last row offers the p-values of the Wilcoxon signed-rank test. From the table, we can see that the cost-sensitive solutions definitely improve the results over the two baseline classifiers across the three measures, but the degrees of improvement vary. For C4.5 decision tree, cost-sensitive learning does not have a significant increase for all the three measures according to the Wilcoxon signed-rank test. On the other hand, cost-sensitive SVM gives a statistically significant increase for both AUC and top-decile lift. However, there is still no significant difference for the EMP measure as the cost-sensitive tree. In summary,

Table 5
Experimental performance of bagging-based methods.

Method	AUC (Rank)	Method	Top-decile lift (Rank)	Method	EMP(Rank)
UnderBagging	0.8047 (1.5454)	UnderBagging	4.5440 (2.4091)	Bagging	1.2906 (1.7273)
RBBagging	0.8056 (1.7272)	Bagging	4.4909 (2.4091)	UnderBagging	1.2871 (2.5455)
SMOTEBagging	0.7811 (3.1818)	RBBagging	4.5061 (2.9091)	RBBagging	1.2839 (3.3636)
OverBaggging	0.7718 (3.8182)	SMOTEBagging	4.3929 (3.4545)	SMOTEBagging	1.2523 (3.3636)
Bagging	0.7473 (4.7272)	OverBaggging	4.3397 (3.8182)	OverBaggging	1.2356 (4.0000)
Iman-Davenport test	$p < 0.0001$		$p = 0.1292$		$p = 0.0044$

Table 6
Experimental performance of boosting-based methods.

Method	AUC (Rank)	Method	Top-decile lift (Rank)	Method	EMP(Rank)
RUSBoost	0.7808 (**1.5455**)	RUSBoost	4.2400 (**1.6364**)	SMOTEBoost	1.2004 (2.3636)
AdaBoost	0.7610 (2.5454)	AdaBoost	4.1327 (2.3636)	AdaBoost	1.1992 (2.3636)
SMOTEBoost	0.7647 (2.6364)	SMOTEBoost	4.1689 (2.4545)	RUSBoost	1.1726 (2.4545)
AdaC2	0.7523 (3.2727)	AdaC2	3.9360 (3.5454)	AdaC2	1.0771 (2.8182)
Iman-Davenport test	$p = 0.0111$		$p = 0.0027$		$p = 0.8329$

cost-sensitive techniques are useful for SVM and C4.5 to deal with class imbalance in churn prediction when the misclassification costs are known.

5.3. Comparison of ensemble solution

In this subsection, we will discuss the results of ensemble solutions. As mentioned in Section 3, there are mainly four types of ensemble solutions: bagging-based, boosting-based, random-forest-based and hybrid ensemble, so that we will analyze the experimental results of each group, separately. The inter-group comparison can be found in the global comparison of the next subsection. In order to validate wether the solutions are beneficial to deal with class imbalance, the original Bagging, AdaBoost and random forest techniques are also used to provide contrast. Tables 5 to 8 show the experimental results in each group, where each entry in these tables gives the mean value as well as mean rank of each metric over all the eleven data sets. The last row provides the p-values of Iman-Davenport test or the Wilcoxon signed-rank test. The methods are arranged according to their mean ranks in an ascending order and the ranks that are significantly worse than the best one are underlined based on Holm's post-hoc test.

1. **Bagging-based ensemble**. As Table 5 shows, all bagging-based solutions improve AUC performance over the Bagging method. However, only UnderBagging has a slight improvement over Bagging for top-decile lift and their ranks are the same. Meanwhile, no improvement can be found for any bagging-based solutions in terms of EMP. UnderBagging has the best performance in terms of AUC and it is significantly better than SMOTEBagging, OverBagging and Bagging methods according to the Holm's test. There is no statistical difference among the five methods for top-decile lift based on the Iman-Davenport test. The priority of Bagging with respect to EMP is significant except for the comparison with UnderBagging. It can also be found that the integration of undersampling with bagging is more efficient than that with oversampling for all the three measures. UnderBagging and RBBagging consistently demonstrate better ranks than OverBagging and SMOTEBagging. They should be considered when AUC and top-decile lift are used while no solutions can contribute to address the class imbalance in terms of EMP.

2. **Boosting-based ensemble**. Table 6 summarizes the experimental results of boosting-based solutions. As can be seen, the effectiveness of boosting-based techniques is ambiguous. In comparison with AdaBoost, AdaC2 has the worst performance for all the three metrics. RUSBoost takes the first place for both AUC and top-decile lift and it is worse than AdaBoost for EMP. SMOTEBoost only presents slightly better mean EMP values than AdaBoost but their mean ranks are the same. According to Holm's test, RUSBoost is significantly better than the other three methods with respect to AUC while it is only significantly better than AdaC2 in terms of top-decile lift. For EMP, SMOTEBoost and Adaboost are the top two methods, but there is no significant difference among the four methods according to the Iman-Davenport test. All the results show that in churn prediction, cost-sensitive ensembles are not a good option to cope with class imbalance. Moreover, RUSBoost should be considered for AUC and the top-decile measure while SMOTEBoost is recommenced for EMP among the boosting-based ensemble solutions.

3. **Random-forest-based ensemble**. After careful examination, we can get some insights from Table 7. As for the boosting-based group, the effectiveness here heavily depends on the performance measure. Both weighted random forest (WRF) and balance random forest (BRF) show improvements over the original random forest for AUC but only the latter (BRF) increases the top-decile lift. At the same time, they are still worse than the original random forest algorithm in terms of EMP. According to Holm's test, BRF is significantly better than WRF and RF in terms of AUC and (original) random forest is statistically better than BRF for EMP based on Holm's test. BRF also ranks first in terms of top-decile lift but the difference is not significant among the three methods based on the Iman-Davenport test. The results advocate balanced random forests

Table 7
Experimental performance of random-forest-based methods.

Method	AUC(Rank)	Method	Top-decile lift (Rank)	Method	EMP(Rank)
BRF	0.7952 (1.2727)	BRF	4.2890 (1.7273)	RF	1.2376 (1.6364)
WRF	0.7671 (2.3636)	RF	4.2621 (1.9091)	BRF	1.1929 (1.7273)
RF	0.7611 (2.3636)	WRF	4.0250 (2.3636)	WRF	1.1123 (2.6364)
Iman-Davenport test	$p = 0.0064$		$p = 0.3209$		$p = 0.0260$

Table 8
Experimental performance of hybrid ensemble methods.

Method	AUC(Rank)	Top-decile lift (Rank)	EMP(Rank)
BalanceCascade	0.7675	4.1564	1.1851
EasyEnsemble	0.7900	4.2973	1.1947
Wilcoxon test	$p = 0.0076$	$p = 0.0409$	$p = 0.2477$

Table 9
Experimental performance of global comparison.

Method	AUC			Method	EMP		
	Rank	PV(Holm's)	Action		Rank	PV(Holm's)	Action
UnderBagging	1.3636	–	–	Bagging	1.9091	–	–
BRF	2.5455	0.2578	Not reject	Random forest	3.5455	0.1172	Not Reject
EasyEnsemble	2.5455	0.2578	Not reject	EasyEnsemble	4.0000	0.0453	Reject
RUSBoost	4.2727	0.0053	Not reject	SMOTEBoost	4.4545	0.0148	Reject
ROS+SVM	5.2727	0.0002	Reject	RUS+SVM	4.7273	0.0070	Reject
CS-SVM	6.0909	<0.0001	Reject	CS-SVM	4.8182	0.0053	Reject
RUS+C4.5	6.5455	<0.0001	Reject	RUS+C4.5	6.1364	<0.0001	Reject
CS-Tree	7.3636	<0.0001	Reject	CSTree	6.4091	<0.0001	Reject

in terms of AUC and top-decile lift while no random-forest-based technique works well to address the class imbalance issue with regards to EMP measure.

4. **Hybrid ensemble**. As Table 8 shows, EasyEnsemble is better than BalanceCascade across three measures. Since there are only two methods, we used the Wilcoxon signed-rank test. According to this test, the superiority of EasyEnsemble is significant for both AUC and top-decile lift but not for the EMP measure.

5.4. Global comparison

In this subsection, a global comparison is conducted to find well-performing solutions for each evaluation metric. To make the analysis more concise, we focus our discussion on only two measures: AUC (general) and EMP (domain specific). To achieve this goal, we pick the representative methods which obtain better average ranks with respect to the two different measures from each group for a given measure. More specifically, from the sampling solutions, we select the best sampling methods for each baseline classifier. Since the two cost-sensitive learning solutions can improve over the baseline classifiers, they stand out from the baseline classifiers. Among the ensemble solutions, one method will be retained from the bagging-based, boosting-based, random forest-based and hybrid ensemble group, respectively. In total, eight representative methods are chosen to perform the global comparison for each measure. Table 9 shows the summary of the global comparison, where the second and the sixth column give the mean ranks over the eleven data sets. By using the Iman-Davenport test, we get two p-values that are smaller than 0.0001 for both measures, which help us to reject the null hypothesis that all representative methods have equivalent performance. Consequently, a Holm's test is performed and the third and seventh column in Table 9 give the p-values.

In general, we can see from Table 9 that ensemble solutions dominate the results. The four representative ensemble methods are suited at the top part of the table, especially the bagging-based methods. Another observation is that sampling solutions seem to be more effective than cost-sensitive solutions. For both SVM and C4.5, the sampling treatment always obtains a better ranking than the corresponding cost-sensitive versions with respect to both AUC and EMP measure. If we go one step further, we can see that the representative methods demonstrate some different behavior in terms of AUC and EMP. When AUC is considered, UnderBagging has the best value and it performs statistically better than all the non-ensemble techniques. When EMP is used, Bagging and random forest, two ensemble methods without considering class imbalance, have the best ranks. To summarize, the ensemble solutions are the best performing techniques in dealing with class imbalance, but they do not contribute to improve the profit-based measures.

5.5. Summary and discussion

From the above experimental analysis, we can summarize the main experiment findings and try to answer the research questions as follows:

RQ1 Do different evaluation metrics have similar influence on the performance of different solutions?
A: The evaluation metrics have a large impact on the (measured, ranked) performance of different techniques. Using profit-based measures leads to significantly different performance ranks in contrast with both top-decile lift and AUC, not only in intra-family comparison but also in the global comparison. This suggests that previous conclusions based on the general AUC measure should be carefully validated in a churn prediction context when profit is of concern.

RQ2 Does each solution truly improve the performance in comparison with the original technique without considering class imbalance?
A: As shown in the intra-family comparisons, we can see that different types of solutions have different behaviors in comparison with their counterpart without considering class imbalance. The specific behavior is listed as follows:

- Sampling methods do not grantee the improvement and their performance depends on the classifier. For C4.5 decision tree models, the influence depends on the evaluation measure: most sampling methods can improve AUC and EMP, but none can improve the top-decile measure. At the same time, SVM can gain benefits from resampling for all the three measures. In general, more complex sampling methods do not demonstrate any superior performance compared to simple random sampling methods such as RUS and ROS.
- Cost-sensitive learning can improve over baseline classifiers. The improvement is more obvious for cost-sensitive SVM models than cost-sensitive C4.5.
- With respect to ensemble solutions, most techniques integrated with sampling methods increase the performance in terms of AUC and some of them improve top-decile lift. However, ensemble methods embedding cost-sensitive learning decay the performance of these two measures. No ensemble solution has significant better EMP values than its counterpart without considering class imbalance.

RQ3 What is the best technique for each given evaluation measure, respectively?
A: From the perspective of a global comparison, ensemble-based methods are still the most promising methods. For AUC, the combination of Bagging with random undersampling (UnderBagging, BRF, EasyEnsemble and RUSBoost) shows its superiority, especially UnderBagging. However, while techniques designed for imbalanced data sets do not show any superiority in terms of EMP, the original Bagging and Random forest methods beat other methods based on this measure. The reason behind this needs future research. Furthermore, there is still room to develop algorithms that can deal with class imbalance and optimize for a profit-based evaluation.

6. Conclusions

Customer relationship management is an indispensable part of all business, in which churn prediction is the key step. Class imbalance is intrinsic to data sets in churn prediction. Dealing with this issue and identifying the minority churner group from a large number of non-churners is the core of successful churn modeling. In our study, we present a comprehensive comparison of state-of-the art techniques for dealing with class imbalance. A recently developed profit-based measure EMP is used to evaluate the results from the perspective of costs and benefits. The experimental results show that different types of solutions demonstrate different behaviors. An in-depth exploration of the reaction patterns to different measures is carefully provided. It is also found that ensemble methods are the most prominent methods and one interesting finding is that the original Bagging and random forest learning algorithm without considering class imbalance get the best results with respect to the profit-based measure. Our research provides some guidance to choose suitable methods to deal with class imbalance in both the academic community as well as practice. In the future, we plan to develop some new ensemble techniques to optimize the profit-based measure in the context of churn prediction with class imbalance.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 71401115) and the MOE (Ministry of Education in China) Youth Project of Humanities and Social Sciences (Grant No. 13YJC630249). Bing Zhu is supported by the postdoctoral fellowship from the China Scholarship Council (CSC). This work is also funded by Sichuan University (Grant No. skqy201742).

Appendix

See Tables 10–12

Table 10
Detailed experimental results (AUC).

method	Chile	Duke1	Duke2	KDDcup	K1	K2	K3	K4	K5	Tele1	UCI
C4.5	0.6114	0.5552	0.5921	0.5676	0.5000	0.5574	0.7421	0.9599	0.6312	0.6368	0.8591
C4.5_ADASYN	0.6518	0.5160	0.5388	0.5989	0.6554	0.6377	0.7466	0.9558	0.7492	0.7055	0.8333
C4.5+CLUS	0.6517	0.5192	0.5394	0.6230	0.7750	0.7528	0.8665	0.9495	0.7484	0.7001	0.8552
C4.5+MWMOTE	0.6357	0.5088	0.5203	0.5726	0.5675	0.6226	0.7414	0.9470	0.7089	0.7173	0.8447
C4.5+ROS	0.6346	0.5027	0.5084	0.5285	0.5838	0.5946	0.7074	0.9342	0.6480	0.6550	0.8133
C4.5+RUS	0.6296	0.5374	0.5619	0.6296	0.7620	0.7780	0.8555	0.9621	0.7473	0.7334	0.8570
C4.5+SMOTE	0.6564	0.5250	0.5388	0.6046	0.6264	0.6477	0.7417	0.9434	0.7560	0.7068	0.8529
C4.5+SMOTE-ENN	0.6479	0.5226	0.5364	0.5860	0.6272	0.6672	0.8440	0.9244	0.7405	0.6867	0.8328
C4.5+SMOTE-TL	0.6319	0.5181	0.5365	0.5968	0.6167	0.6296	0.7588	0.9491	0.7626	0.7346	0.8393
SVM	0.6550	0.5289	0.5351	0.5733	0.6431	0.7786	0.8672	0.6529	0.7111	0.6014	0.8851
SVM+ADASYN	0.6827	0.5506	0.5506	0.6236	0.6763	0.8142	0.9393	0.7480	0.7814	0.7361	0.8874
SVM+CLUS	0.6953	0.5454	0.5402	0.6818	0.6716	0.8395	0.9241	0.6998	0.7416	0.7843	0.8676
SVM+MWMOTE	0.6897	0.5536	0.5555	0.6366	0.6767	0.8169	0.9348	0.7458	0.7796	0.7277	0.8903
SVM+ROS	0.6984	0.5471	0.5564	0.6379	0.7074	0.8317	0.9363	0.7314	0.7856	0.7380	0.8881
SVM+RUS	0.7072	0.5494	0.5443	0.6632	0.6634	0.8438	0.9259	0.7037	0.7607	0.7583	0.8775
SVM+SMOTE	0.6977	0.5523	0.5501	0.6268	0.7125	0.8072	0.9340	0.7209	0.7718	0.7374	0.8850
SVM+SMOTE-ENN	0.7000	0.5730	0.5545	0.6474	0.6983	0.8213	0.9319	0.7400	0.7722	0.7484	0.8820
SVM+SMOTE-TL	0.6928	0.5570	0.5492	0.6380	0.6980	0.8196	0.9296	0.7370	0.7759	0.7407	0.8885
CSSVM	0.6978	0.5287	0.5418	0.6516	0.6945	0.8278	0.9316	0.6890	0.7735	0.7623	0.8823
CSTree	0.6417	0.5054	0.5098	0.5593	0.5760	0.6117	0.9343	0.9364	0.6537	0.7004	0.8297
AdaBoost	0.6993	0.5391	0.5538	0.6502	0.7329	0.8233	0.9215	0.9710	0.7714	0.7957	0.9126
AdaC2	0.6975	0.5348	0.5573	0.6306	0.7621	0.8145	0.8883	0.9643	0.7600	0.7757	0.8896
RUSBoost	0.6931	0.5673	0.5788	0.6748	0.8115	0.8631	0.9284	0.9655	0.7979	0.8093	0.8995
SMOTEBoost	0.7141	0.5311	0.5564	0.6508	0.7521	0.8534	0.9276	0.9628	0.7773	0.7752	0.9109
Bagging	0.6834	0.5143	0.5158	0.6760	0.6454	0.8577	0.9302	0.9595	0.7369	0.7890	0.9116
OverBagging	0.7214	0.5452	0.5615	0.6757	0.7457	0.8638	0.9140	0.9672	0.7878	0.7978	0.9095
SMOTEBagging	0.7086	0.5607	0.5716	0.6817	0.7702	0.8707	0.9368	0.9599	0.8099	0.8099	0.9118
RBBagging	0.7217	0.6005	0.6117	0.7179	0.8442	0.8800	0.9479	0.9651	0.8281	0.8284	0.9158
UnderBagging	0.7140	0.6021	0.6147	0.7180	0.8316	0.8846	0.9444	0.9648	0.8290	0.8292	0.9194
Random forest	0.6796	0.5409	0.5376	0.6700	0.7517	0.8386	0.9112	0.9712	0.7839	0.7745	0.9126
BRF	0.7035	0.5768	0.6165	0.7097	0.8288	0.8810	0.9162	0.9662	0.8233	0.8190	0.9062
WRF	0.7085	0.5325	0.5661	0.6473	0.7882	0.8566	0.8954	0.9618	0.7960	0.7841	0.9017
EasyEnsemble	0.7080	0.5768	0.5831	0.6895	0.8170	0.8788	0.9378	0.9658	0.8080	0.8202	0.9047
BalanceCascade	0.6944	0.5243	0.5498	0.6404	0.7898	0.8439	0.9315	0.9720	0.7742	0.7956	0.9060

Table 11
Detailed experimental results (Top-decile lift).

method	Chile	Duke1	Duke2	KDDcup	K1	K2	K3	K4	K5	Tele1	UCI
C4.5	5.7673	1.3499	1.5751	1.3222	4.7970	4.5144	5.4215	9.0581	6.0506	4.8807	5.1267
C4.5+ADASYN	3.2579	1.1522	1.4752	2.0474	3.3356	3.8832	6.0018	8.6186	4.1551	3.5806	4.9009
C4.5+CLUS	2.0252	1.0990	1.0581	1.7986	3.6223	3.6329	7.2286	8.8671	4.7056	2.3853	4.1943
C4.5+MWMOTE	3.0943	1.1968	1.2286	1.9798	2.8555	4.0742	5.7520	8.8478	4.2405	3.6383	4.6185
C4.5+ROS	4.4340	0.6547	0.7604	1.7330	5.9496	5.8972	6.5454	8.7715	4.1081	3.2765	4.6996
C4.5+RUS	2.3899	1.2840	1.3753	2.3230	2.5668	3.7212	6.7070	9.1345	4.7021	3.1874	4.7557
C4.5+SMOTE	3.2704	1.2797	1.3883	2.0641	3.4797	3.7654	6.0092	8.8479	4.2858	3.6907	4.5616
C4.5+SMOTE-ENN	3.1321	1.2478	1.3492	2.1675	3.7436	3.9273	7.2653	8.8097	4.2945	3.5544	4.3710
C4.5+SMOTE-TL	3.2579	1.2840	1.3275	2.0504	3.3594	4.0157	6.2736	8.8479	4.1847	3.7117	4.4843
SVM	3.0377	1.1522	1.2721	1.3308	2.0638	3.5156	6.0606	2.2932	3.3032	1.2267	4.4875
SVM+ADASYN	3.4151	1.3669	1.4589	2.1433	1.6318	4.1628	7.8530	2.6563	3.9460	3.6697	4.5862
SVM+CLUS	3.4591	1.3265	1.2036	2.5218	2.3997	4.0010	7.8383	2.2169	3.0053	3.2870	4.1871
SVM+MWMOTE	3.4151	1.4009	1.5045	2.2992	2.1357	4.0010	7.9192	2.5990	3.7980	3.6121	4.4947
SVM+ROS	3.4465	1.3095	1.4361	2.3719	2.2558	4.0010	7.9265	2.4651	4.1203	3.7379	4.4873
SVM+RUS	3.6038	1.3754	1.3286	2.4259	2.0397	4.3100	7.8457	2.1213	3.3049	3.7588	4.3073
SVM+SMOTE	3.3396	1.2861	1.4383	2.3941	2.4958	4.1922	7.8751	2.2169	3.8502	3.6802	4.4553
SVM+SMOTEENN	3.2453	1.5093	1.4926	2.5198	2.1599	4.1774	7.8090	2.5608	3.8624	3.6907	4.1870
SVM+SMOTETL	3.3208	1.4243	1.4730	2.4592	2.1598	4.2952	7.8016	2.5226	3.8851	3.6855	4.5227
CSSVM	3.6289	1.0948	1.3014	2.4739	2.2798	4.1482	7.8090	2.1593	3.9164	3.6645	4.4275
CSTree	4.9120	1.0416	1.1265	1.8324	5.5418	8.6789	7.8604	9.5550	9.3729	4.1153	4.1801
AdaBoost	3.7987	1.3477	1.4882	2.1453	3.0715	3.6038	7.2580	8.9244	4.4234	3.8375	5.5614
AdaC2	3.4591	1.1819	1.4763	1.9424	3.8155	3.5744	6.3103	8.9435	4.2858	3.5334	4.7736
RUSBoost	3.4465	1.4051	1.6870	2.2614	4.1516	3.7068	7.2066	9.0200	4.6812	4.1311	4.9429
SMOTEBoost	3.6730	1.2563	1.5111	2.1206	3.5996	3.6774	7.4490	8.8862	4.4948	3.7379	5.4520
Bagging	3.7925	1.1947	1.2199	2.6848	3.7192	4.2658	8.2203	9.2109	5.1272	4.4771	5.4873
OverBagging	3.8931	1.3988	1.5654	2.3305	3.4556	3.8392	7.8016	9.1154	4.7161	4.1625	5.4589
RBBagging	3.7925	1.7559	1.7392	2.7645	4.2473	3.9127	7.9412	8.9817	4.9251	4.1048	5.4026
SMOTEBagging	3.7736	1.4264	1.6186	2.3381	3.7195	4.2068	7.8530	8.9245	4.7980	4.2149	5.4484
UnderBagging	3.9182	1.7516	1.7544	2.7671	4.3434	4.1039	7.8604	8.9817	5.0279	4.1048	5.3707
Random forest	3.6541	1.2287	1.3210	2.2745	3.5036	3.7214	7.5739	8.9434	4.9426	4.1835	5.5366
BRF	3.6478	1.3881	1.7522	2.5041	3.8633	4.0450	7.1625	8.8670	5.0279	3.8218	5.0988
WRF	3.7170	1.1394	1.3492	1.6533	3.9595	4.2069	6.6409	8.7142	4.8171	3.0668	5.0102
EasyEnsemble	3.6541	1.4732	1.7316	2.3643	3.8156	3.9569	7.5739	8.9627	4.6847	4.0472	5.0067
BalanceCascade	3.5723	1.1245	1.3492	2.0721	3.6716	3.8392	7.3976	9.0009	4.4304	4.0682	5.1940

Table 12
Detailed experimental results (EMP).

method	Chile	Duke1	Duke2	KDDcup	K1	K2	K3	K4	K5	Tele1	UCI
C4.5	0.6893	0.0000	0.0000	0.0759	0.0328	0.1220	1.0435	2.2213	0.5480	1.0329	5.6294
C4.5+ADASYN	0.5269	0.0000	0.0000	0.1694	0.1963	0.2932	1.0192	2.1745	0.4991	1.2296	5.4883
C4.5+CLUS	0.2515	0.0000	0.0000	0.0788	0.1394	0.3280	1.0879	2.1562	0.4296	0.5813	5.7375
C4.5+MWMOTE	0.4729	0.0000	0.0000	0.1384	0.1313	0.2296	0.9836	2.1838	0.4225	1.1812	5.5126
C4.5+ROS	0.5833	0.0000	0.0000	0.0696	0.1039	0.2247	0.7694	2.2190	0.4443	1.0890	5.4509
C4.5+RUS	0.1805	0.0000	0.0006	0.2621	0.2272	0.4075	1.1776	2.1345	0.4675	1.1598	5.6320
C4.5+SMOTE	0.5643	0.0000	0.0000	0.1583	0.1985	0.3481	1.0446	2.1724	0.4768	1.1818	5.6119
C4.5+SMOTE-ENN	0.4966	0.0000	0.0000	0.1750	0.1049	0.3339	1.2734	2.1617	0.4645	1.2259	5.5420
C4.5+SMOTE-TL	0.4914	0.0000	0.0000	0.1786	0.1855	0.2341	1.1225	2.1628	0.4711	1.1841	5.7047
SVM	0.7357	0.0006	0.0012	0.0947	0.0327	0.2985	0.9602	0.1005	0.1409	0.1704	5.4205
SVM+ADASYN	0.6523	0.0000	0.0007	0.2458	0.0727	0.3906	1.4136	0.1498	0.2943	1.2742	5.2499
SVM+CLUS	0.7241	0.0002	0.0006	0.2335	0.0894	0.3334	1.4786	0.0826	0.1571	1.1274	5.0207
SVM+MWMOTE	0.6858	0.0004	0.0003	0.3123	0.0588	0.4174	1.4364	0.0624	0.3053	1.2689	5.2400
SVM+ROS	0.7053	0.0007	0.0007	0.3019	0.1060	0.3698	1.3993	0.0936	0.3389	1.2455	5.2478
SVM+RUS	0.6980	0.0009	0.0052	0.3098	0.0287	0.4337	1.4921	0.1071	0.1800	1.2870	5.0407
SVM+SMOTE	0.6327	0.0004	0.0006	0.3155	0.1234	0.3708	1.4056	0.1310	0.2558	1.2407	5.3447
SVM+SMOTE-ENN	0.5778	0.0000	0.0005	0.3441	0.0733	0.3803	1.4528	0.1339	0.2852	1.2813	5.3077
SVM+SMOTE-TL	0.6558	0.0000	0.0011	0.3230	0.1057	0.4274	1.4310	0.1287	0.3113	1.2619	5.2094
CSSVM	0.7301	0.0012	0.0037	0.3372	0.0379	0.4244	1.4474	0.0636	0.2435	1.2400	5.2209
CSTree	0.6584	0.0000	0.0000	0.0855	0.0863	0.1788	1.4556	2.1407	0.4287	1.2466	5.1198
AdaBoost	0.9058	0.0004	0.0019	0.2191	0.1656	0.2687	1.2387	2.2859	0.5450	1.3578	6.2028
AdaC2	0.7266	0.0016	0.0029	0.1385	0.2166	0.2727	0.9751	2.3295	0.5161	1.1508	5.5172
RUSBoost	0.6833	0.0002	0.0018	0.2580	0.3252	0.3723	1.1767	2.2004	0.5578	1.4773	5.8458
SMOTEBoost	0.8574	0.0003	0.0021	0.2001	0.2132	0.3585	1.2703	2.2503	0.5222	1.2680	6.2616
Bagging	0.8749	0.0009	0.0063	0.4295	0.1902	0.4695	1.4552	2.2814	0.6525	1.5778	6.2587
OverBagging	0.9115	0.0012	0.0020	0.2827	0.1345	0.4297	1.3570	2.2498	0.5786	1.4740	6.1710
RBBagging	0.8199	0.0001	0.0010	0.4666	0.2881	0.4404	1.4496	2.2581	0.6321	1.5254	6.2419
SMOTEBagging	0.8706	0.0003	0.0044	0.2851	0.2306	0.4072	1.3867	2.2769	0.5805	1.5442	6.1891
UnderBagging	0.8429	0.0004	0.0020	0.4684	0.2484	0.4635	1.4510	2.2587	0.6267	1.5325	6.2632
RF	0.8534	0.0002	0.0038	0.3227	0.2179	0.2995	1.3235	2.2328	0.6188	1.4450	6.2960
BRF	0.7034	0.0010	0.0016	0.3798	0.2667	0.4402	1.1396	2.0986	0.5979	1.3831	6.1104
WRF	0.7030	0.0007	0.0003	0.0685	0.2521	0.4073	1.0016	2.1471	0.5721	1.0969	5.9859
EasyEnsemble	0.7261	0.0004	0.0012	0.2974	0.2691	0.4075	1.3084	2.1494	0.5532	1.4750	5.9547
BalanceCascade	0.7143	0.0004	0.0012	0.1908	0.2226	0.3506	1.2659	2.2757	0.5472	1.4430	6.0242

References

[1] O. Ali, U. Ariturk, Dynamic churn prediction framework with more effective use of rare event data: the case of private banking, Expert Syst. Appl. 41 (17) (2014) 7889–7903.
[2] S. Barua, M. Islam, X. Yao, MWMOTE-Majority weighted minority oversampling technique for imbalanced data set learning, IEEE Trans. Knowl. Data Eng. 26 (2) (2014) 405–425.
[3] G.E. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM Sigkdd Explorations Newsletter 6 (1) (2004) 20–29.
[4] J. Burez, D.V.d. Poel, Handling class imbalance in customer churn prediction, Expert Syst Appl 36 (3, Part 1) (2009) 4626–4636.
[5] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: syntheticminority over-sampling technique, J. Artif. Intell. Res. 16 (3) (2002) 321–357.
[6] N. Chawla, A. Lazarevic, L. Hall, K.W. Bowyer, SMOTEBoost: improving prediction of the minority class in boosting, in: Proceeding of PKDD, 2003, pp. 107–119.
[7] Z. Chen, Z. Fan, M. Sun, A hierarchical multiple kernel support vector machine for customer churn prediction using longitudinal behavioral data, Eur. J. Oper. Res. 223 (2) (2012) 461–472.
[8] C. Chen, A. Liaw, L. Breiman, Using Random Forests to Learn Imbalanced Data, Statistics Department of University of California at Berkeley, 2004 Technical report.
[9] M. Colgate, P. Danaher, Implementing a customer relationship strategy: the asymmetric impact of poor versus excellent execution, J. Acad. Market. Sci. 28 (3) (2000) 375–387.
[10] K. Coussement, S. Lessmann, G. Verstraeten, A comparative analysis of data preparation algorithms for customer churn prediction: a case study in the telecommunication industry, Decis. Support Syst. (2016). Online first.
[11] J.D. sar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[12] S. Ertekin, J. Huang, L. Bottou, C.L. Giles, Learning on the border: active learning in imbalanced data classification, in: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, ACM, 2007, pp. 127–136.
[13] V. V. García, J.S. Sànchez, R. Mollineda, On the effectiveness of preprocessing methods when dealing with different levels of class imbalance, Knowl. Based Syst. 25 (1) (2012) 13–21.
[14] S. García, F. Herrera, Evolutionary undersampling for classification with imbalanced datasets: proposals and taxonomy, Evolut. Comput. 17 (3) (2009) 275–306.
[15] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, Ordering-based pruning for improving the performance of ensembles of classifiers in the framework of imbalanced datasets, Inf. Sci. (Ny) 354 (2016) 178–196.
[16] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, IEEE Trans. Syst. Man Cybern. Part C 41 (4) (2012) 463–484.
[17] N. Gordini, V. Veglio, Customers churn prediction and marketing retention strategies, An Application of Support Vector Machines Based on the AUC Parameter-Selection Technique in B2B E-Commerce Industry, Industrial Marketing Management, 2016. Online first.
[18] H. He, Y. Bai, E.A. Garcia, S. Li, ADASYN: adaptive synthetic sampling approach for imbalanced learning, in: Proceedings of IEEE International Joint Conference on Neural Networks, 2008, pp. 1322–1328.
[19] S. Hido, H. Kashima, Y. Takahashi, Roughly balanced bagging for imbalanced data, Stat. Anal. Data Min. 2 (5-6) (2009) 412–426.
[20] A. Idris, M. Rizwan, A. Khan, Churn prediction in telecom using random forest and PSO based data balancing in combination with various feature selection strategies, Comput. Elect. Eng. 38 (6) (2012) 1808–1819.

[21] A. Jahromi, S. Stakhovych, M. Ewing, Managing b2b customer churn, retention and profitability, Ind. Market. Manage. 43 (7) (2014) 1258–1268.
[22] A. Keramati, R. Jafari-Marandi, M. Aliannejadi, I. Ahmadian, M. Mozaffari, U. Abbasi, Improved churn prediction in telecommunication industry using data mining techniques, Appl. Soft Comput. 24 (2014) 994–1012.
[23] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: one-sided selection, in: Proceeding of International Conference on Machine Learning, 1997, pp. 179–186.
[24] T.M. Khoshgoftaar, J.V. Hulse, A. Napolitano, Comparing boosting and bagging techniques with noisy and imbalanced data, IEEE Trans. Syst. Man Cybern. Part A 99 (2010) 1–17.
[25] A. Lemmens, C. Croux, Bagging and boosting classification trees to predict churn, J. Market. Res. (JMR) 43 (2006) 276–286.
[26] L. X, J. Wu, Z. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Trans. Syst. Man Cybern. Part B 39 (2) (2009) 539–550.
[27] V. López, A. Fernández, J.G. Moreno-Torres, F. Herrera, Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification, open problems on intrinsic data characteristics, Expert Syst. Appl. 39 (7) (2012) 6585–6608.
[28] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics, Inf. Sci. (Ny) 250 (2013) 113–141.
[29] S. Neslin, S. Gupta, W. Kamakura, J. Lu, C. Mason, Detection defection: measuring and understanding the predictive accuracy of customer churn models, J. Market. Res. 43 (2) (2006) 204–211.
[30] T. Raeder, G. Forman, N.V. Chawla, Learning from imbalanced data: evaluation matters, Data Mining: Found. Intell. Paradigms Springer, 2012, pp. 315–331.
[31] C. Seiffert, T.M. Khoshgoftaar, J.V. Hulse, A. Folleco, An empirical study of the classification performance of learners on imbalanced and noisy software quality data, Inf. Sci. (Ny) 259 (2014) 571–595.
[32] C. Seiffert, T.M. Khoshgoftaar, J.V. Hulse, RUSBOost: a hybrid approach to alleviating class imbalance, IEEE Trans. Syst. Man Cybern. Part A 40 (1) (2010) 185–197.
[33] H.J. Shin, D.H. Eom, S.S. Kim, One-class support vector machines-an application in machine fault detection and classification, Comput. Ind. Eng. 48 (2) (2005) 395–408.
[34] Y. Sun, M.S. Kamel, A.K. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recognit. 40 (12) (2007) 3358–3378.
[35] Z. Sun, Q. Song, X. Zhu, H. Sun, B. X., Y. Zhou, A novel ensemble method for classifying imbalanced data, Pattern Recognit. 48 (5) (2015) 1623–1637.
[36] C. Tsai, Y. Lu, Customer churn prediction by hybrid neural networks, Expert Syst. Appl. 36 (10) (2009) 12547–12553.
[37] K.M. Ting, An instance-weighting method to induce cost-sensitive trees, IEEE Trans. Knowl. Data Eng. 14 (3) (2002) 659–665.
[38] S.J. Yen, Y.S. Lee, Cluster-based under-sampling approaches for imbalanced data distributions, Expert Syst. Appl. 36 (3) (2009) 5718–5727.
[39] X. Zhang, J. Zhu, S. Xu, Y. Wan, Predicting customer churn through interpersonal influence, Knowl.-Based Syst. 28 (2012) 97–104.
[40] B. Zhu, B. Baesens, A. Backiel, S.v. Broucke, Benchmarking sampling techniques for imbalance learning in churn prediction, J. Operat. Res. Soc. (2017), doi:10.1057/s41274-016-0176-1. Online first.
[41] J.V. Hulse, T.M. Khoshgoftaar, A. Napolitano, Experimental perspectives on learning from imbalanced data, in: Proceedings of the 24th International Conference on Machine learning, 2007, pp. 935–942.
[42] 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, Eur. J. Oper. Res. 218 (1) (2012) 211–229.
[43] T. Verbraken, W. Verbeke, B. Baesens, A novel profit maximizing metric for measuring classification performance of customer churn prediction models, IEEE Trans. Knowl. Data Eng. 25 (5) (2013) 961–973.
[44] K. Veropoulos, C. Campbell, N. Cristianini, Controlling the sensitivity of support vector machines, in: Proceedings of the international joint conference on AI, 1999, pp. 55–60.
[45] S. Wang, X. Yao, Diversity analysis on imbalanced data sets by using ensemble models, in: IEEE Symposium on Computational Intelligence, 2009, pp. 324–331.
[46] Y. Xie, X. Li, E. Ngai, M. Ying, Customer churn prediction using improved balanced random forests, Expert Syst. Appl. 36 (3, Part 1) (2009) 5445–5449.