Handling Class Imbalance in Mobile Telecoms Customer Churn Prediction

Clement Kirui
School of Information Science and Engineering
Central South University
Changsha, Hunan, China

LiHong
School of Information Science and Engineering
Central South University
Changsha, Hunan, China

Edgar Kirui
Faculty of Commerce
Egerton University
Eldoret, Kenya

ABSTRACT

Class imbalance is a major problem that is often experienced when dealing with rare events, such as churn recognition in the mobile telecommunications industry. In this work, various strategies of addressing the problem are studied and a demonstration of how under-sampling and Synthetic Minority Oversampling Technique (SMOTE) can be used to address the problem is given. The two techniques are implemented individually first, and then we take the hybrid approach by combining both SMOTE and undersampling. For performance evaluation, two predictive techniques, C4.5 decision tree and Naïve Bayes classifier with 10-fold cross validation are used. TPR and FPR values are obtained and used to generate ROC curves from which AUC values are calculated and performance comparison of the three techniques is performed. Results show that the hybrid approach achieves better performance.

General Terms

Data Mining, Experimentation, Mobile Telecommunications

Keywords

Class Imbalance, Customer Churn, Over-sampling, Under-sampling, Prediction

1. INTRODUCTION

While one of the key aspirations of every business is to build and maintain a loyal customer base, customer churn is a common problem that cuts across many sectors. Wireless telecommunications industry is one of the industries that experience this problem in a more serious manner due to many factors that include stiff competition, increasing innovation as a result of new technologies, low switching costs, and deregulation by governments[1]. These factors are even more elaborate especially in mature or saturated markets. In order to address the problem, players in this industry deploy prediction models that are able to identify the customers that are about to leave so that intervention strategies are deployed in a bid to retain them. In data mining, churn prediction is considered a two-class classification problem in which customers are classified as either churners or non-churners. When developing a predictive model, the two issues that should be addressed satisfactorily are attribute subset selection and class imbalance in telecom churn data sets[2], [3]. These two are important because they affect the ability of prediction model to positively identify the actual possible churners. Attributes with higher predictive importance should be carefully selected before feeding them into the churn prediction system. The problem of class imbalance on the other hand impairs the performance of some data mining algorithms like decision trees and rules[4]. There are many strategies that have been used to address this problem in other domains. These include cost-sensitive learning, using alternate evaluation metrics, sampling approaches, and ensemble learning[5], [6], [7], [8]. In this work, sampling approaches are studied further. The performance of under-sampling, SMOTE[9] and a combination of the two are compared in the context of churn prediction in the mobile telecommunications industry.

2. CLASS IMBALANCE IN TELECOM CHURN DATASETS

Datasets with imbalanced class distributions are quite common in many real life applications. For example in a customer churn prediction system, it is common to find that churning customers are significantly fewer than active customers who are willing to continue using the services of a service provider. In such an infrequent occurrence of the class of concern, a correct classification of the rare class has greater value than a correct classification of the majority class. Customer churn is a costly risk that threatens not only the profitability of a service provider, but also the existence of the same. Costs associated with customer churn include loss of revenue, costs of customer retention and reacquisition, advertisement costs, organizational chaos, as well as planning and budgeting chaos[1]. In addition, previous studies have shown that the cost of acquiring new customers is much higher than the cost of retaining the existing ones. Therefore, it makes business sense for players in this industry to identify the subscribers who are likely to attrite beforehand and develop intervention strategies in a bid to retain as many customers as possible. In such a scenario, therefore, correct prediction of customers who are about to leave is of greater value than active customers.

Although there are many measures for evaluating the performance of a prediction model, some of the measures are not appropriate when dealing with highly skewed data. The overall accuracy measure is commonly used to compare the performance of classifiers and predictive models. However, it may not be appropriate when dealing with imbalanced datasets due to the inherent skew in data. For example, in the data we used for this study, 94.4% of subscribers are classified as active while the remaining 5.6% are churners. If a prediction model achieves 90% overall accuracy but fails to predict the churners, it gives a false impression that we are correct 90% of the time. However, this is not true because the class we are concerned with-the churners-have not been identified. Therefore, alternative metrics should be used to evaluate churn prediction models.

3. TECHNIQUES FOR ADDRESSING CLASS IMBALANCE

3.1 Alternative evaluation measures

As mentioned in section 2, overall accuracy is the measure that is commonly used to compare performance of classifiers and predictive models, especially with balanced datasets. For imbalanced datasets, overall accuracy may not be appropriate because a prediction model can achieve high accuracy but fails to recognize minority class examples. Therefore, other evaluation measures need to be employed [5].

Table 1: Confusion matrix

		Predicted Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

- *True positive (TP):* Number of positive instances correctly predicted.
- *False negative (FN):* Number of positive instances wrongly predicted as negative.
- *False positive (FP):* Number of negative instances wrongly predicted as positive
- *True negative (TN):* Number of negative instances correctly predicted.

From the confusion matrix in Table 1, the following related measures can be derived [5]:

*True positive rate (TPR):*Ratio of positive instances predicted correctly against the sum of positive instances correctly predicted and false negative instances. It corresponds to *sensitivity*and can be expressed as$TPR = \frac{TP}{TP+FN}$.

*True negative rate (TNR):*Ratio of negative instances predicted correctly against the sum of negative instances correctly predicted and false positive instances. It corresponds to *specificity* and can be expressed as$TPR = \frac{TN}{TN+FP}$.

*False positive rate (FPR):*Ratio of positive instances wrongly predicted against the sum of negative instances correctly predicted and false positive instances. It is expressed as$FPR = \frac{FP}{TN+FP}$.

*False negative rate (FNR):*Ratio of negative instances wrongly predicted against the sum of positive instances correctly predicted and negative instances predicted wrongly. It is expressed as $FNR = \frac{FN}{TP+FN}$.

Precision: fraction of records that actually turn out to be positive in the group the classifier has declared as positive. The higher the precision is, the lower the number of false positive errors committed by the classifier.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Fraction of positive instances correctly predicted by the classifier. Its value is equivalent to true positive rate. The higher the value of recall the fewer the number of instances misclassified as negative.

F-Measure: Represents the harmonic mean between recall and precision.

$$F_1 = \frac{2\times recall \times precision}{recall \times precision} = \frac{2\times TP}{2\times TP+FP+FN}$$

$$= \frac{2}{\frac{1}{recall}+\frac{1}{precision}}$$

The harmonic mean of two numbers *x* and *y* tends to be closer to the smaller of the two numbers. Thus, a high value of F1 measure ensures that both precision and recall are reasonably high.

3.2 The Receiver Operating Characteristic (ROC) Curve

An ROC curve is a graphical plot that shows the representation of the tradeoff between true positive rate and false positive rate of a predictor[10]. The vertical axis represents the true positive rate while the horizontal axis represents the false positive rate. Each point along the curve corresponds to one of the models induced by the classifier.

A classifier located closer to the upper left corner of the diagram performs better while one that makes random guesses reside along the diagonal connecting the points (0,0) and (1,1). Random guessing means that a case is classified as a positive class with a fixed probability irrespective of its attribute set.

From the ROC curve, area under ROC curve (AUC) for each curve can be calculated. The curve that is closest to the top left corner has the largest AUC and shows better performance. An area under curve value of 1 means the model is perfect while a curve plotted from a random classifier model would achieve a value of 0.5. The larger the AUC value the better the model.

3.3 Cost-sensitive learning

In cost-sensitive learning, a cost matrix is taken into consideration during model building during which a model with the lowest cost is generated[7]. In churn prediction modeling, the value of predicting the churners (rare class) far outweighs the value of correctly predicting the non-churners (majority class). The main intent of the costs is to punish false prediction. As such, higher costs are applied to false negatives than with false positives (with churners being the positive class).

Cost information can be incorporated into classification algorithms in a number of ways. For example, for decision tree classifiers, costs can be used to: choose the best attribute used to split the data; determining whether a sub-tree should be pruned; manipulate the weights of training records so that the decision tree converges to a decision tree that has the lowest cost; and to modify the decision rule at each leaf node.

3.4 Sampling-based approaches

The idea of sampling is to modify the distribution of cases so that the rare class is well represented in the training set. Some of the common sampling techniques are as discussed below.

3.4.1 Under-sampling

Under-sampling eliminates majority-class instances from the training set. However, this increases the risk of discardingpotentially useful majority-class instances, resulting in a sub-optimal model. In order to reduce this risk, under-sampling can be performed in multiple iterations in the same way induction of multiple classifiers is performed in ensemble

learning approach[6]. Also, focused under-sampling methods may be used. This is where the sampling procedure makes an informed choice regarding negative instances that should be eliminated, e.g. those located far away from the decision boundary.

3.4.2 Over-sampling

Over-samplingreplicates the rare class instances until the training set has an equal number of rare and majority class instances[4]. Because of this, over-sampling introduces additional training instances, thus, increasing the time necessary to build the model. Besides, for noisy data, over-sampling may cause model over-fitting because some of the noisy instances may be replicated many times.

3.4.3 Hybrid approach

Hybrid approach uses a combination of under-sampling the majority class and oversampling the minority class to achieve a balanced class distribution. Under-sampling can be done using random or focused sampling while oversampling can be done by replicating minority class instances. Foroversampling, the *k*-nearest neighbors for each exiting minority class instance are first determined. A new minority instance is then generated at some random point along the line segment that joins the minority instance to one of its *k*-nearestneighbors. This process is repeated until the sample is balanced[5].

4. METHODOLOGY

4.1 Data collection

The dataset that was used in this experiment was obtained from a European telecommunications company and was collected from August to October 1997. Originally, the dataset consisted of 112 attributes and 106,495 instances. New features with higher predictive importance were then extracted and combined with the original data set to form a new modified data set. The procedure is demonstrated in [3]. The modified set was then used for further experiments. In the data set, 5.6% were churners and the rest were active subscribers. Figure 4-1 below shows the distribution and the level of class imbalance between the churn examples and the active examples (non-churners) that are present in the data set.

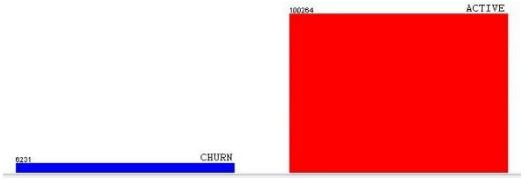

Figure 1: Class distributions of CHURN and ACTIVE classes

4.2 Data preprocessing

In this phase, we focused on data cleaning and feature selection. Data cleaning entails removal of noise and addressing various inconsistencies in data. Noise is the irrelevant information which would cause problems for the subsequent processing steps. Therefore, noisy data should be removed. This irrelevant information includes wrong spelling words caused by human error, special symbols like mathematical symbols and punctuation marks, missing values, duplicated information (e.g. the same attributes with the same values are in different tables of a database). This noise can be removed by finding their locations and using the correct values to replace them, or some times by deleting them if the missing values are too many.

Feature selection plays an important role in determining the performance of predictive models in the terms of prediction rates (high TP and low FP) for churn recognition. If a robust set of features is selected in this phase, churn prediction rates can be significantly improved. The modified data set consisted of 134 attributes. However, not all these attributes have significant predictive importance. Therefore, careful feature selection must be performed in order to select only the important features. In this phase, information gain filter was used to rank the features in descending order of their information content. After which top 60 features were selected to be used in the prediction model.

4.3 Prediction techniques

Two prediction techniques were used in the experiment, i.e. C4.5 decision tree and Naïve Bayes classifier.

4.3.1 C4.5 Decision Tree

The C4.5 decision tree [11]uses the 'divide and conquer' method to construct a model based on a tree structure. Nodes in the tree represent features, with branches representing possible values connecting the features. A leaf representing the class terminates a series of nodes and branches. Initially, the method starts to search an attribute with best information gain at root node and divide the tree into sub-trees. Similarly, each sub-tree is further separated recursively following the same rule. The partitioning stops if the leaf node is reached or there is no information gain. Once the tree is created, rules can be obtained by traversing each branch of the tree.

4.3.2 Naive Bayes

Naive-Bayes classifier[10] is based on the Bayesian theorem. It analyses the relationship between each feature and the class for each instance to derive a conditional probability for the relationships between the feature values and the class. We assume that X is a vector of instances where each instances is described by attributes $\{X_1,...,X_k\}$ and a random variable C denoting the class of an instance. Let x be a particular instance and c be a particular class. During training, the probability of each class is computed by counting how many times it occurs in the training dataset. This is called the prior probability $P(C=c)$. In addition to the prior probability, the algorithm also computes the probability for the instance x given c. Under the assumption that the attributes are independent this probability becomes the product of the probabilities of each single attribute. Naive Bayes has achieved good results in many cases even when this assumption is violated.

The probability that an instance x belongs to a class c can be computed by combining the prior probability and the probability from each attribute's density function using the Bayes formula:

$$P(C = c|X = x) = \frac{P(C = c)\prod_i P(X_i|C = c)}{P(X = x)}$$

The denominator is invariant across classes and only necessary as a normalizing constant (scaling factor). It can be computed as the sum of all joint probabilities of the enumerator:

$$P(X = x) = \sum_j P(C_j)P(X = x|C_j)$$

4.4 Evaluation criteria

In order to conduct performance evaluation, we used 10-fold cross validation technique[12]. In this process, the initial data are randomly partitioned into 10 mutually exclusive subsets or "folds," each of approximately equal size. Training and

testing is performed 10 times. In iteration *i*, partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the model. For classification, the accuracy estimate is the overall number of correct classifications from the 10 iterations, divided by the total number of records in the initial data. This provides a good indication of how well the prediction technique will perform on unseen data.

A confusion matrix similar to the one shown in table 1 is generated.

As mentioned previously, the overall accuracy is not an appropriate measure when evaluating imbalanced data. We therefore obtain the false churn rate (FP) and true churn rate (TP) of each of the sampling techniques used in this study for the two prediction techniques. Based on these pairs of FP and TP, ROC curves are plotted. The horizontal axis represents the false churn rates (FP) while the vertical axis represents the true churn rates (TP). Each ROC curve consists of a sequence of points, each of which presents a pair of prediction rates (FP, TP) for a specified sampling rate. Generally, an ROC curve which is closer to the left-top corner has the highest area under ROC curve (AUC) and presents the best prediction result[10].

5. RESULTS AND ANALYSIS

5.1 Under-sampling

Under-sampling eliminates majority-class instances from the training set. The possible problem with this approach is that it may discard potentially useful majority-class instances, resulting in a sub-optimal model. However, a solution to this would be to perform under-sampling multiple times and to induce multiple classifiers like in ensemble learning approach. Also, focused under-sampling methods may be used. This is where the sampling procedure makes an informed choice regarding negative instances that should be eliminated, e.g. those located far away from the decision boundary.

In order to achieve this, stratified random sampling was used. In this technique, the data is grouped into two homogeneous strata, CHURN and ACTIVE. Random sampling is then used in each stratum independently to obtain data of the required sample size. In this study, we maintained the size of CHURN at 100% and varied the size of ACTIVE from 10% to 100% in order to generate continuous data points. In effect, the majority class was under-sampled to various sample sizes in order to have continuous data points. The results are as shown in tables2 and 3.

5.2 Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE[9] is an over-sampling approach that introduces minority class examples into the training set by creating synthetic" examples rather than by over-sampling with replacement. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the *k* minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the *k* nearest neighbors are randomly chosen.

The common implementation uses five nearest neighbors. For instance, if the amount of over-sampling needed is 200%, only two neighbors from the five nearest neighbors are chosen and one sample is generated in the direction of each. Synthetic samples are generated as follows: Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general.

For instance, if there is a sample (8, 5) and another one (2, 4) being its neighbor, a new sample is generated as follows:

$S_{1\_1} = 8 \;\; S_{2\_1} = 2 \qquad S_{1\_1} - S_{2\_1} = -6$
$S_{1\_2} = 5 \;\; S_{2\_2} = 4 \qquad S_{2\_2} - S_{1\_2} = -1$

A new "synthetic" sample is generated as follows:

$$(S_1^{'}, S_2^{'}) = (8,5) + rand(0-1) \times (-6,-1)$$

The synthetic examples cause the classifier to create larger and less specific decision regions rather than smaller and more specific regions. More general regions are now learned for the positive class samples rather than those being subsumed by the majority negative class samples around them.

In this experiment, we oversampled the minority class examples by 100% to 1000% in order to generate continuous data points. The after which two predictive techniques – C4.5 decision tree and Naïve Bayes with 10-fold cross validation were used. Values of TP rate and FP rate for the two prediction techniques were recorded as shown in tables 2 and 3.

5.3 Combining SMOTE with Under-Sampling

Since under-sampling discards majority class examples from the training set and SMOTE over-samples the minority class examples, if the two techniques are combinedthe skew in the dataset gradually balances out and eventually favors the positive class.At lower degrees of under-sampling, the negative class has a lower number of cases in the training set. The converse is true for the higher degrees of under-sampling.Learning is done on the resultant dataset after applying both SMOTE to over-sample the positive class and under-sampling the negative class. In under-sampling, the majority class examples are randomly removed from the training set until the minority class examples become some specified percentage of the majority class. This ensures that the classifier learns from varying degrees of under-sampling, thus generating continuous data points.

SMOTE over-samples the minority class (CHURN) examples by calculating *n* nearest neighbors and generating synthetic examples, in this case we used five nearest neighbors. By using SMOTE, the minority class examples were over-sampled from 100% to 1000% in order to generate different continuous data points. At the same time, we under-sampled the majority class examples from 10% to 100%. Just like in the previous experiment, we used C4.5 and Naïve Bayes algorithms and 10-fold cross validation for performance evaluation. The values of true churn and false churn were recorded as shown in tables 2 and 3.

5.3.1 Generating an ROC Curve

From the results of the two classifiers in tables 2 and 3, the values are used to rank their predictions, from the most likely record to be classified as a positive class to the least likely record. From Figure 4-3 and Figure 4-4, the horizontal axis

represents FP rate (false churn) while the vertical axis represents TP rate (true churn). TPR is plotted against FPR for all data points.

The TPR is plotted along the vertical axis while the FPR is plotted along the horizontal axis. Each point on the ROC curve corresponds to the inductive model of a classifier. Ideally, the curve nearest to top left has a higher AUC and therefore shows better performance.

Table 2: Comparison using C4.5 decision tree

Algorithm	Sample size (%)										
	S+U	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
	U	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
SMOTE + Under sampling	TP	0.368	0.56	0.697	0.787	0.855	0.901	0.844	0.961	0.982	0.991
	FP	0.052	0.06	0.058	0.051	0.045	0.035	0.062	0.017	0.009	0.005
Under sampling	TP	0.531	0.394	0.331	0.292	0.255	0.22	0.197	0.177	0.159	0.151
	FP	0.254	0.141	0.097	0.072	0.052	0.038	0.025	0.021	0.016	0.013
SMOTE	TP	0.447	0.549	0.639	0.731	0.774	0.808	0.827	0.84	0.873	0.888
	FP	0.025	0.033	0.035	0.026	0.034	0.034	0.035	0.044	0.037	0.04

Table 3: Comparison using Naive Bayes Classifier

Algorithm	Sample size (%)										
	SM+U	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
	U	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
SMOTE + Undersampling	TP	0.833	0.857	0.841	0.864	0.885	0.887	0.892	0.908	0.898	0.895
	FP	0.687	0.808	0.725	0.789	0.743	0.797	0.806	0.816	0.813	0.805
Undersampling	TP	0.877	0.809	0.815	0.836	0.823	0.774	0.801	0.789	0.778	0.776
	FP	0.788	0.677	0.686	0.715	0.697	0.639	0.668	0.658	0.648	0.643
SMOTE	TP	0.662	0.816	0.861	0.867	0.864	0.87	0.863	0.887	0.88	0.875
	FP	0.486	0.717	0.796	0.803	0.808	0.802	0.807	0.817	0.814	0.812

The TPR is plotted along the vertical axis while the FPR is plotted along the horizontal axis. Each point on the ROC curve corresponds to the inductive model of a classifier. Ideally, the curve nearest to top left has a higher AUC and therefore shows better performance.

From figures2 and 3 the combination of SMOTE plus under-sampling performs better than plain under-sampling. This can be attributed to the fact that the minority class instances are over-sampled while at the same time the majority class instances are under-sampled by certain percentage points to the required sample sizes. The effect is that when both under-sampling and over-sampling are performed in sync the skew in the data balances out at a higher rate, thus, the classifier can recognize both minority and majority class instances easily.As the over-sampling and the under-sampling average out, higher recognition rates are obtained at this point.

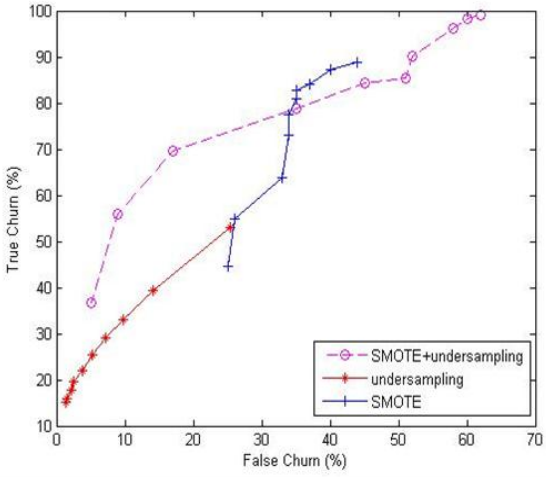

Figure 2: Comparison using C4.5 decision tree.

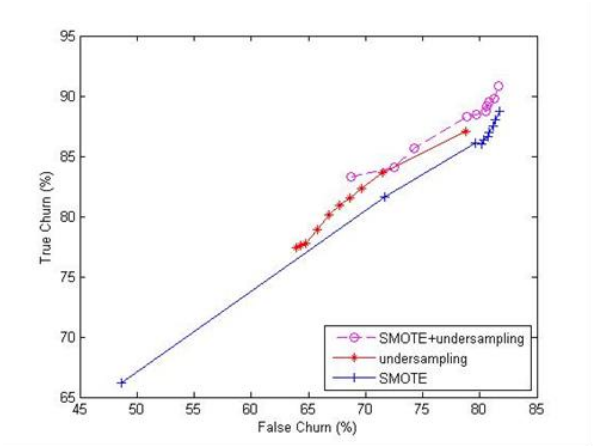

Figure 3: Comparison using Naive Bayes.

6. CONCLUSION

Class imbalance in churn data sets impedes learning by some predictive techniques like the decision tree models and decision rules. The common techniques used to address this problem are under-sampling, over-sampling, and hybrid techniques. In under-sampling, the instances of majority class are eliminated until a required sample size is obtained. In over-sampling, the instances of minority class are duplicated until a required sample size is achieved. Hybrid techniques combine a number of other techniques simultaneously, for example, over-sampling and under-sampling. In many studies, hybrid models perform better than individual models because they usually capitalize on the advantages of each individual technique. In this study, SMOTE is combined with under-sampling to improve churn recognition rates. SMOTE introduces synthetic instances of the minority class into the training set thereby over-sampling the minority class. Since SMOTE over-samples the minority class and under-sampling discards instances of the majority class, the skew in the data balances out gradually eventually favoring the positive class.

7. REFERENCES

[1] R. Mattison, The Telco Churn Management Handbook, XiT Press, 2006.

[2] H. B.Q, T. Kechadi, B. Buckley, G.Kiernan, E.Keogh and T.Rashid, "A new feature set with new window techniques for customer churn prediction in land-line telecommunications," *Expert Systems With Applications*, vol. 37, pp. 3657-3665, 2010.

[3] C. Kirui, L. Hong, W. Cheruiyot and H. Langat, "Predicting Customer Churn in Mobile Telecommunications Using Probabilistic Classifiers in Data Mining," *International Journal of Computer Science Issues*, vol. 10, no. 2, p. 165, 2013.

[4] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *International Conference on Artificial Intelligence (IC-AI'2000)*, 2000.

[5] P. N. Tan, M. Steinbach and V. Kumar, Introduction to Data Mining, Pearson Education Asia Inc., 2006.

[6] X.-Y. Liu, J. Wu and a. Z.-H. Zhou, "Exploratory Undersampling for Class-Imbalance Learning," *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS – PART B*, 2008.

[7] C. X. Ling and V. S. Sheng, "Cost-Sensitive Learning and the Class Imbalance Problem," *Encyclopedia of Machine Learning. C. Sammut (Ed.).*, 2008.

[8] M. Galar, A. Fern´andez, E. Barrenechea and H. Bustince, "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS*, 2011.

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, 2002.

[10] J. Han, M. Kamber and J. Pei, Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, 2012.

[11] Q. J.R, C4.5: Programs for Machine Learning, 1993.

[12] I. H. Witten, E. Frank and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2011.

[13] S. Y. Hung, D. C. Yen and H. Y. Wang, "Applying data mining to telecom churn management," *Expert Systems with Applications*, vol. 31, p. 515–524, 2006.

[14] B. Huang, T. Kechadi, B. Buckley, G. Kiernan, E. Keogh and T. Rashid, "A new feature set with new window techniques for customer churn prediction in land-line telecommunications," *Expert Systems with Applications*, vol. 37, p. 3657–3665, 2010.

[15] J. Burez and D. V. d. Poel, "Handling class imbalance in customer churn prediction," *Expert systems with Applications*, vol. 36, pp. 4626-4636, 2009.

[16] X. Guo, Y. Yin, C. Dong, G. Yang and G. Zhou, "On the class imbalance problem," in *Fourth International Conference on Natural Computation*, 2008.

[17] Y. Sangho, J. Koehler and A. Ghobarah, "Prediction of advertiser churn for google adwords," in *JSM Proceedings*, 2010.

[18] D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth and Y. Zhou, "Detecting adversarial advertisements in the wild," in *the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011.