An Evaluation of Sampling on Filter-Based Feature Selection Methods

Kehan Gao
Eastern Connecticut State University
83 Windham St., Willimantic, CT 06226
gaok@easternct.edu

Taghi Khoshgoftaar & Jason Van Hulse
Florida Atlantic University
777 Glades Rd., Boca Raton, FL 33431
taghi@cse.fau.edu; jason@gmail.com

Abstract

Feature selection and data sampling are two of the most important data preprocessing activities in the practice of data mining. Feature selection is used to remove less important features from the training data set, while data sampling is an effective means for dealing with the class imbalance problem. While the impacts of feature selection and class imbalance have been frequently investigated in isolation, their combined impacts have not received enough attention in research. This paper presents an empirical investigation of feature selection on imbalanced data. Six feature selection techniques and three data sampling methods are studied. Our efforts are focused on two different data preprocessing scenarios: data sampling used *before* feature selection and data sampling used *after* feature selection. The experimental results demonstrate that the *after* case generally performs better than the *before* case.

Introduction

Data preprocessing, a critical initial step in data mining and machine learning projects, is often used to improve the quality of training data set. Data collection methods are often loosely controlled, resulting in many data quality problems such as feature redundancy and irrelevance, data instance conflict and abnormality, and missing values. In this paper, we focus on feature selection in the context of software quality prediction. In software quality assurance practice, practitioners often use software metrics (attributes or features) gathered during the software development process and various data mining techniques to build classification models for predicting whether a given program module (instance or example) is in the fault-prone (*fp*) or not fault-prone (*nfp*) class. However, not all collected software metrics are useful or have the same contributions to classification results. The goal of feature selection is to choose the most relevant and important attributes from the raw data set that can significantly contribute to the modeling process. In this paper, we investigate six filter-based feature ranking techniques to select subsets of attributes.

In addition, for binary classification problems, class imbalance is frequently encountered. In software quality engineering practice, this refers to the overabundance of *nfp* class examples vis-à-vis *fp* class examples. Similar problems have also emerged in other application domains (Kamal *et al.* 2009; Zhao *et al.* 2007; Engen, Vincent, & Phalp 2008). Traditional classification algorithms attempt to maximize classification accuracy without regard to the significance of the different classes, resulting in a large number of misclassifications from *fp* to *nfp*. This type of misclassification is extremely severe in software quality assurance, implying a missed opportunity to correct a faulty module prior to operation. A variety of techniques have been proposed to alleviate the problems associated with class imbalance (Van Hulse, Khoshgoftaar, & Napolitano 2007). This paper employs data sampling to solve the problem. Data sampling attempts to improve classification performance by balancing the class distribution of training data sets. This can be implemented in one of two ways: oversampling and undersampling. Oversampling creates a more balanced data set by increasing the number of examples in the minority class. Undersampling, on the other hand, reduces the number of examples belonging to the majority class. Both under and oversampling can be done randomly, or using more "intelligent" algorithms. In this work, we examine the performance of three different data sampling techniques, including both random and intelligent over and under sampling.

Although a great deal of work has been done for feature selection and data sampling separately, limited research has been done on both together, particularly in the software engineering field. Chen et al. have studied data row pruning (data sampling) and data column pruning (feature selection) in the context of software cost/effort estimation (Chen *et al.* 2005). However, the data sampling in their study was not to deal with the class imbalance problem, and the study was not about the two-group classification issue. Also, their research focused only on the case in which data sampling is used prior to feature selection. In contrast, we present a comprehensive empirical investigation of feature selection on imbalanced data using various sampling techniques. In our study, two different data preprocessing scenarios are considered: data sampling used *prior to* feature selection and data sampling used *after* feature selection. We would like to compare both cases and see which one is more appropriate for the given data sets in software quality prediction. To our knowledge, no such studies have ever been done in this field or the data mining and machine learning community.

The experiments are carried out on four data sets from a very large legacy telecommunications software system. The distributions of the two classes (*fp* and *nfp*) for the four data sets are significantly different; between 1% and 7% of the data instances are in the minority (*fp*) class. In the experiments, six feature selection techniques and three data sampling methods are considered together for data preprocessing. Two different scenarios, *before* (data sampling used prior to feature selection) and *after* (data sampling used following feature selection), are studied for each combination of the sampling and feature ranking technique. The classifier employed in this work is SVM (support vector machine), one of the most popular learners in data mining and machine learning. Our results show that data sampling used after feature selection generally performs better than the reverse case, but this is associated with the sampling method and feature selection technique involved in the data preprocessing.

Related Work

Feature selection, as an important activity in data preprocessing, has been extensively studied for many years in data ming and machine learning. A comprehensive survey of feature selection algorithms is presented in (Liu & Yu 2005). Hall and Holmes (Hall & Holmes 2003) investigated six attribute selection techniques that produce ranked lists of attributes and applied them to 15 data sets from the UCI collection. The comparison results showed no single best approach for all situations. However, a wrapper-based approach was the best overall attribute selection scheme in terms of accuracy if speed of execution was not considered. Saeys et al. (Saeys, Abeel, & Peer 2008) investigated the use of ensemble feature selection techniques, where multiple feature selection methods were combined to yield results. They showed that the ensemble approach presented great promise for high-dimensional domains with small sample sizes and provided more robust feature subsets than a single feature selection technique. Many applications of feature selection in various fields have been reported. Jong et al. applied feature selection to proteomic pattern data based on support vector machines (Jong *et al.* 2004). Ilczuk et al. highlighted the importance of attribute selection in judging the qualification of patients for cardiac pacemaker implantation (Ilczuk *et al.* 2007). In the context of text mining, where attributes are binary in value, Forman investigated multiple filter-based feature ranking techniques (Forman 2003).

Feature selection has been studied extensively, but it is not the only problem facing many data sets; frequently they are plagued by imbalanced data as well. To overcome the difficulties associated with learning from imbalanced data, various techniques have been developed. A natural solution is to modify the data set to balance it. These approaches are collectively known as sampling. Numerous variants of undersampling and oversampling have been investigated and applied to a variety of domain data sets (Chawla *et al.* 2002; Cieslak, Chawla, & Striegel 2006). Imbalanced data can also be dealt with directly when creating the classifier. This is easier with some classifiers than with others. Some classifiers, such as Naïve Bayes, can produce a probability that an instance is in a given class. One can just adjust the threshold at which an instance is put into a certain class and solve the problem of attempting to classify everything as being in the majority class to improve accuracy. A related approach is cost-sensitive classification (Elkan 2001). However, one of the problems related to cost-sensitive learning is that the costs of different types of errors or misclassifications are not easy to obtain or estimate, especially during the modeling process. Therefore, in this study, we focus on applying data sampling to improve the performance of classifiers for imbalanced data.

While the impacts of attribute selection and data sampling have been frequently investigated in isolation, their combined impacts have not gained enough attention in research. In one of the few studies, Liu et al. (Liu, Motoda, & Yu 2004) introduced the concept of active feature selection, and investigated a selective sampling approach to active feature selection in a filter model setting. But the purpose of sampling in their research was to deal with data sets with a large number of instances instead of class imbalance. In this paper, we use both feature selection and data sampling together to process our training data sets. The objective of this research is to address the following simple but important question: "In what order should feature selection and data sampling be applied for classifier development?" In other words, should feature selection be performed before or after data sampling? We investigate both cases in our study. To our knowledge, no similar study has ever been done or reported in related literatures.

Filter-Based Ranking Techniques

Feature selection (also known as attribute selection) is the process of choosing some subset of the features and building a classifier based solely on those. Feature selection methods can be categorized as either *wrappers* or *filters* based on whether a learning algorithm is involved in the selection process. In addition, feature selection techniques can also be categorized into *feature subset selection* and *feature ranking*. Feature subset selection approaches select subsets of attributes that collectively have good predictive capability, while feature ranking just evaluates attributes individually and ranks attributes according to their individual predictive power. The advantage of feature ranking is that it requires only the computation and sorting of the scores of each feature individually.

In this study, we use filter-based feature ranking techniques (i.e., ranking features independently without involving any learning algorithm). The procedure of feature ranking is to score each feature according to a particular method, allowing the selection of the best set of features. The six filter-based feature ranking techniques used in this work include: chi-square (CS), information gain (IG), gain ratio (GR), two types of ReliefF (RF and RFW), and symmetrical uncertainty (SU). The chi-square (CS) test (Cameron & Trivedi 1998) is used to examine whether the two variables are independent. CS is more likely to find significance to the extent that (1) the relationship is strong, (2) the sample size is large, and/or (3) the number of values of the two associated features is large. Information gain, gain ratio, and symmetrical uncertainty are measures based on the concept of entropy from information theory (Witten & Frank 2005). Information gain (IG) is the information provided about the target class attribute Y, given the value of another attribute X. IG measures the decrease of the weighted average impurity of the partitions, compared with the impurity of the complete set of data. A drawback of IG is that it tends to prefer attributes with a larger number of possible values, i.e., if one attribute has a larger number of values, it will appear to gain more information than those with fewer values, even if it is actually no more informative. One strategy to counter this problem is to use the gain ratio (GR), which penalizes multiple-valued attributes. Symmetrical uncertainty (SU) is another way to overcome the problem of IG's bias toward attributes with more values, doing so by dividing by the sum of the entropies of X and Y. Relief is an instance-based feature ranking technique introduced by Kira and Rendell (Kira & Rendell 1992). ReliefF is an extension of the Relief algorithm that can handle noise and multiclass data sets, and is implemented in the WEKA tool (Witten & Frank 2005). When the `WeightByDistance` (weight nearest neighbors by their distance) parameter is set as default (false), the algorithm is referred to as RF; when the parameter is set to true, the algorithm is referred to as RFW.

Sampling Techniques

This work considers three different data sampling techniques. Random oversampling (ROS) and random undersampling (RUS) achieve more balanced data sets by randomly duplicating examples of the minority class (ROS) or randomly removing examples from the majority class (RUS). Synthetic Minority Oversampling Technique, or SMOTE (SMO) (Chawla *et al.* 2002) creates new minority class examples by interpolating between existing minority class examples.

Experimental Data Sets

For this study, we conduct our experiments on four data sets from a very large legacy telecommunications software system (denoted as LLTS). The LLTS software system was developed in a large organization by professional programmers using PROTEL, a proprietary high level procedural language (similar to C). The system consists of four successive releases of new versions of the system, and each release was comprised of several million lines of code. The data collection effort used the Enhanced Measurement for Early Risk Assessment of Latent Defect (EMERALD) system (Hudepohl *et al.* 1996). A decision support system for software measurements and software quality modeling, EMERALD periodically measures the static attributes of the most recent version of the software code. We refer to these four releases as Rel.1, Rel.2, Rel.3 and Rel.4. Each set of associated source code files is considered as a program module. The LLTS data sets consist of 42 software metrics, including 24 product metrics, 14 process metrics and 4 execution metrics. The dependent variable is the class of the software module, *fp* or *nfp*. The fault-proneness is based on a selected threshold, i.e., modules with one or more faults are considered as *fp*, *nfp* otherwise. Table 1 summarizes the numbers of the *fp* and *nfp* modules and their percentages in each data set.

Table 1: Data Set Summary

Data Set	nfp		fp		Total
	#	%	#	%	
Rel.1	3420	93.7	229	6.3	3649
Rel.2	3792	95.3	189	4.7	3981
Rel.3	3494	98.7	47	1.3	3541
Rel.4	3886	97.7	92	2.3	3978

Experimental Design

As previously mentioned, the main goal of the paper is to investigate the learning impact of the two data preprocessing activities (feature selection and data sampling) working simultaneously on the given software engineering data sets. Six filter-based feature ranking techniques and three data sampling methods are used in the experiments. Two different scenarios are taken into account for each given paired ranker and sampler: (1) data sampling used *before* feature selection, and (2) data sampling used *after* feature selection.

For feature ranking, we select the top $\lceil \log_2 n \rceil$ attributes from each of the ranked lists as the subsets of attributes, where n is the number of independent attributes in the original data set. The reasons why we choose the top $\lceil \log_2 n \rceil$ attributes (i.e., six metrics in this study) include (1) related literature lacks guidance on the number of features that should be selected when using a feature ranking technique; (2) our preliminary study shows that it is appropriate to use $\lceil \log_2 n \rceil$ features when applying SVM learner to binary classification in general and imbalanced data sets in particular; and (3) a software engineering expert with more than 20 years experience recommended selecting $\lceil \log_2 n \rceil$ number of metrics for software system such as our case study. In addition, the ratio between the *nfp* examples and *fp* examples in the post sampling data set is set to 65-to-35 and 50-to-50. We only present the results with the 65-to-35 case for space limitations. Similar results were also obtained from the 50-to-50 case. This work uses SVM (support vector machine) as the classifier and AUC (the area under the ROC curve) as the evaluation metric.

Support vector machines (SVMs) are a category of generalized linear classifiers (Shawe-Taylor & Cristianini 2000) that map input vectors to a higher dimensional space where a maximal margin hyperplane is constructed. The sequential minimal optimization algorithm (SMO) provides an effective method to train support vector machines (Platt 1999). In our study, the complexity parameter, `c`, is set to '5.0', while the `buildLogisticModels` parameter is set to 'true' to produce proper probability estimates. Other classifiers such as naïve Bayes, multilayer perceptrons, logistic regression and k nearest neighbors were also performed in the experiments. However, due to the space limitation we only present

Table 2: Performance of SVM models measured using AUC

Release 1

	RUS			ROS			SMO		
	BEF	AFT	p	BEF	AFT	p	BEF	AFT	p
CS	.7923	**.8012**$^\diamond$	.01	.7586	**.8010**$^\diamond$	.00	.7629	**.8014**$^\diamond$	.00
GR	**.7873**$^\diamond$	.7720	.00	.7640	**.7695**	.29	.7507	**.7671**$^\diamond$	.00
IG	.7910	**.7993**$^\diamond$	.01	.7545	**.7998**$^\diamond$	.00	.7676	**.7993**$^\diamond$	.00
RF	**.8148**	.8110	.06	**.8136**	.8131	.67	**.8128**	.8126	.89
RFW	.8091	**.8111**	.45	$\underline{.8130}$	$\underline{.8130}$	.99	$\underline{.8114}$	$\underline{.8114}$	.99
SU	.7879	**.8004**$^\diamond$	.00	.7557	**.8009**$^\diamond$	.00	.7459	**.7998**$^\diamond$	.00

Release 2

	RUS			ROS			SMO		
	BEF	AFT	p	BEF	AFT	p	BEF	AFT	p
CS	.8190	**.8263**$^\diamond$	.00	.7964	**.8245**$^\diamond$	.00	.8036	**.8218**$^\diamond$	.00
GR	**.7952**	.7949	.94	**.8131**$^\diamond$	.7956	.00	.7659	**.7933**$^\diamond$	.00
IG	.8187	**.8203**	.34	.7972	**.8199**$^\diamond$	.00	.8024	**.8178**$^\diamond$	.00
RF	**.8298**$^\diamond$	.8194	.00	**.8282**$^\diamond$	.8224	.00	**.8272**$^\diamond$	.8208	.00
RFW	.8146	**.8210**$^\diamond$	.03	**.8268**	.8239	.11	**.8246**	.8223	.08
SU	.8003	**.8017**	.73	.7958	**.8030**$^\diamond$	.03	**.8083**	.8025	.11

Release 3

	RUS			ROS			SMO		
	BEF	AFT	p	BEF	AFT	p	BEF	AFT	p
CS	.8212	**.8257**	.56	.7241	**.8272**$^\diamond$	.00	.8021	**.8257**$^\diamond$	.04
GR	**.8124**$^\diamond$	.7882	.01	.7154	**.7877**$^\diamond$	.00	.7319	**.7847**$^\diamond$	.00
IG	.8203	**.8221**	.83	.7188	**.8251**$^\diamond$	.00	.8094	**.8239**	.14
RF	.8254	**.8306**	.41	**.8421**	.8352	.10	**.8368**	.8335	.52
RFW	.8175	**.8274**	.13	**.8433**	.8369	.17	**.8368**	.8342	.64
SU	**.8207**$^\diamond$	.8025	.02	.7175	**.8037**$^\diamond$	.00	.7328	**.7994**$^\diamond$	.00

Release 4

	RUS			ROS			SMO		
	BEF	AFT	p	BEF	AFT	p	BEF	AFT	p
CS	.7913	**.8051**	.07	.7586	**.8212**$^\diamond$	.00	$\underline{.8216}$	$\underline{.8217}$	.99
GR	**.7869**	.7776	.25	.7578	**.7767**$^\diamond$	.04	**.7831**	.7763	.29
IG	.7926	**.8190**$^\diamond$	.00	.7585	**.8266**$^\diamond$	.00	.8217	**.8262**	.25
RF	.8105	**.8140**	.49	**.8224**	.8191	.50	**.8172**	.8168	.93
RFW	.8064	**.8163**	.12	.8203	**.8222**	.66	.8167	**.8193**	.56
SU	**.7835**	.7826	.91	.7606	**.7788**$^\diamond$	.01	**.7902**	.7783	.06

value$^\diamond$ represents significantly better case between the paired t-test ($p \leq 0.05$)
value represents better case between the paired t-test ($0.05 < p < 0.95$)
$\underline{\text{value}}$ represents equal case between the paired t-test ($p \geq 0.95$)

the results obtained from SVM.

In this study, we use the Area Under the ROC (receiver operating characteristic) curve (i.e., AUC) to evaluate classification models. The ROC curve graphs true positive rates versus the false positive rates (Fawcett 2006) (the positive class is synonymous with the minority class). Traditional performance metrics for classifier evaluation consider only the default decision threshold of 0.5 (Seliya, Khoshgoftaar, & Van Hulse 2009). ROC curves illustrate the performance across all decision thresholds. A classifier that provides a large area under the curve is preferable over a classifier with a smaller area under the curve. A perfect classifier provides an AUC that equals 1. AUC is one of the most widely used single numeric measures that provides a general idea of the predictive potential of the classifier. It has also been shown that AUC is of lower variance and more reliable than other performance metrics such as precision, recall, and F-measure (Jiang *et al.* 2009).

In the experiments, we use ten runs of five-fold cross-validation to build and test our classification models. That is, the data sets are partitioned into five folds, where four folds are used to train the model, and the remaining (hold out) fold is used to validate the model. This is repeated five times so that each fold is used as hold out data once. In addition, we perform ten independent repetitions (runs) of each experiment to remove any biasing that may occur during the random selection process. A total of 7,200 models are trained and evaluated during the course of our experiments. Note that all data preprocessing activities (sampling and feature selection) are performed solely on the training data sets but not on the validation (test) data sets.

Results and Analysis

The classification results using the SVM learner for the two data preprocessing scenarios are reported in Table 2. For each given ranking technique and sampling method, we compare the *before* (denoted BEF) and *after* (denoted AFT) situations. A total of 18 paired outcomes are reported in each of the four data sets (Rel.1 to 4). Note that each result represents the average over the ten runs of five-fold cross-validation outcomes. An unpaired two tailed t-test is used for each paired comparison. The unpaired t method tests the null hypothesis that the population means related to two independent group samples are equal against the alternative hypothesis that the population means are different. p-values are provided for each pair of comparisons in the table. The significance level is set to 0.05; when the p-value is less than 0.05, the two group means (in our case between the BEF and AFT scenarios) are significantly different from one another. For example, for Rel.1, when using CS to select features and RUS to sample the data, the result demonstrates that using RUS after CS significantly outperformed using RUS prior to CS, because the p-value (0.01) is less than the specified cut-off 0.05. For the RFW ranker and RUS sampler, the AFT case is better than the BEF case, but the difference is not statistically significant. As another example, for the RFW ranker and ROS sampler, the classification performance in terms of AUC for the two scenarios BEF and AFT are equal or almost equal. This can also be confirmed by the p-value, which is 0.99. In this study, we categorize the two means as equal when $p \geq 0.95$. In Table 2, for each paired comparison between BEF and AFT, one can always find which one performs better (marked with **bold**) or significantly better (marked with **bold**$^\diamond$) than the other or whether they have the same performance (marked with $\underline{\text{underline}}$).

Table 3 shows the summary of the comparisons between BEF and AFT over all the 18 cases for each data set (release). 'SB' means significantly better, i.e., $p \leq 0.05$, 'B' means insignificantly better, i.e., $0.05 < p < 0.95$, while 'E' represents the case that two group means are the same, where $p \geq 0.95$. For 2 out of 18 cases for Rel.1, the classification models display the same performance for two scenarios, while for the other 16 cases, AFT performs better or significantly better than BEF for 12 cases and worse or significantly worse for the remaining four. For Rel. 2-4, similar results were obtained — the situations where AFT outperforms BEF dominate. In summary, AFT performs better than BEF for 63% of cases (AFT is significantly better for 42% of cases), worse than BEF for 34% of cases (AFT is significantly worse than BEF for only 10% of cases) and equal to BEF for the remainder.

In addition, we performed a three-way ANOVA test for the results. The three factors include Factor A, which represents the two orders of the data preprocessing activities (BEF and AFT), Factor B, which represents the six filter-based feature rankers (CS, GR, IG, RF, RFW and SU), and

Table 3: Performance Comparison between BEF and AFT

	Difference ($p < .95$)					Equal
	BEF		**AFT**		Total	($p \geq .95$)
	SB	B	SB	B		
Rel.1	1	3	10	2	16	2
	6%	17%	56%	11%	89%*	11%
Rel.2	4	4	8	2	18	0
	22%	22%	44%	11%	100%*	0%
Rel.3	2	4	7	5	18	0
	11%	22%	39%	28%	100%	0%
Rel.4	0	6	5	6	17	1
	0%	33%	28%	33%	94%	6%
Total	7	17	30	15	69	3
	10%	24%	42%	21%	96%*	4%

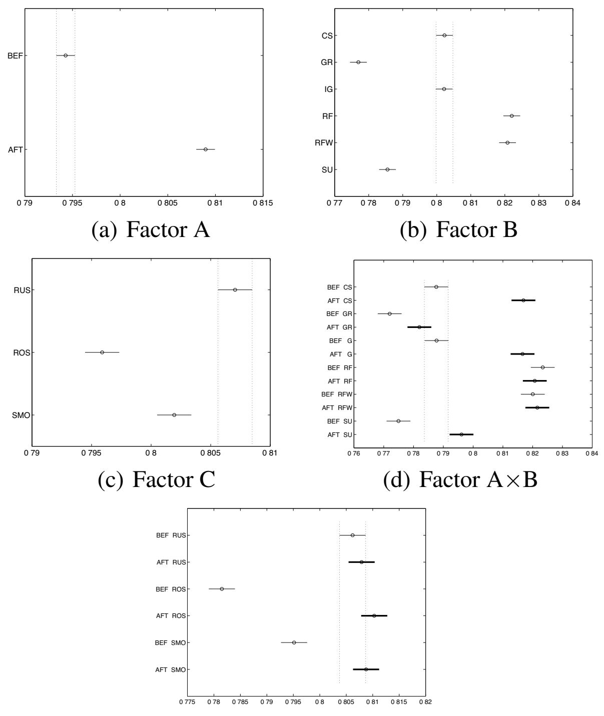
B: better at $0.05 < p < 0.95$

SB: significantly better at $p \leq 0.05$

*: The sum of SB% and B% for BEF and AFT differs from the Total% in 1% due to rounding error.

Table 4: Three-way ANOVA for **LLTS** Data sets

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	0.0777	1	0.0777	215.73	0
B	0.3964	5	0.0793	220.13	0
C	0.0299	2	0.0150	41.57	0
A×B	0.0571	5	0.0114	31.73	0
A×C	0.0442	2	0.0221	61.32	0
B×C	0.0592	10	0.0059	16.43	0
A×B×C	0.0474	10	0.0047	13.15	0
Error	0.5057	1404	0.0004		
Total	1.2176	1439			



(a) Factor A

(b) Factor B

(c) Factor C

(d) Factor A×B

(e) Factor A×C

Figure 1: **LLTS**-Multiple Comparison

Factor C, which represents the three sampler (RUS, ROS and SMO). The ANOVA model can be used to test the hypothesis that the group means for each of the main factors are equal against the alternative hypothesis that at least one pair of group means are different. If the alternative hypothesis is accepted, numerous procedures can be used to determine which of the means are significantly different from the others. This involves the comparison of two means with the null hypothesis that the means are equal. In this study, we use Tukey's multiple comparison test. The significance level for all tests is set to 0.05.

The ANOVA results, as shown in Table 4, indicate that for all three main factors the alternate hypotheses are accepted, since the p-values are less than 0.05. In addition, the two-way and three-way interaction terms are also statistically significant. For example, the significance of interaction A×B implies that changing the value of Factor A significantly impacts the mean values of Factor B. Multiple comparison tests are also carried out for the three main factors and the interaction terms A×B and A×C, since this study is more interested in the two different data preprocessing scenarios (Factor A) as well as their association with other factors such as feature selection technique (Factor B) and sampling method (Factor C). The comparison results are presented in Figure 1. Each figure displays graphs with each group mean represented by a symbol (○) and the 95% confidence interval around the symbol. Two means are significantly different ($\alpha = 0.05$) if their intervals are disjoint, and are not significantly different if their intervals overlap. From the figures, we can see the following points. (1) For Factor A, between the two different data processing scenarios, AFT performs significantly better than BEF; (2) For Factor B, among the six filter-based feature ranking techniques, RF and RFW perform best followed by CS and IG, then SU and GR; (3) For Factor C, all three sampling methods perform significantly differently from one another. Among them, RUS performs best, ROS performs worst, and SMO is in-between; (4) For interaction A×B, Figure 1(d) presents the paired performances between BEF and AFT for each of the given filter-based rankers, where the BEF interval is represented by a thin line and the AFT interval is represented by a thick line. The figure demonstrates that AFT performs much better than BEF when the CS, GR, IG and SU rankers are used, but very similar to BEF when the RF and RFW rankers are employed. In other words, the choice of filter does have a significant impact on the differences between the mean AUC values of BEF and AFT; (5) For interaction A×C, Figure 1(e) presents the paired performance between BEF and AFT for each of the given samplers. The graph indicates that AFT significantly outperforms BEF for ROS and SMO, but the two scenarios demonstrate very similar performance when RUS is used. In addition, AFT presents more consistent (stable) performance than BEF with respect to different sampling techniques employed.

Conclusion

This study investigates feature selection on imbalanced data sets. Data sampling methods are used to counteract the negative effect of class imbalance while filter-based feature ranking techniques are used to select the most relevant features for classification. Six filter-based feature selection techniques and three data sampling methods are implemented. When both data sampling and feature selection are considered simultaneously for data processing, an interesting question is raised — "should data sampling be performed before or after feature selection?" Two different scenarios (*before* and *after*) are studied for each pair of feature selection technique and data sampling method. The experiments are conducted on four highly imbalanced software quality data sets and the results demonstrate that data sampling done *after* feature selection generally outperforms the reverse case, i.e., data sampling done *before* feature selection. In addition, the *after* scenario demonstrated more stable performance than the *before* scenario with respect to the various sampling techniques. Future work will conduct additional empirical studies with data from other application domains.

References

Cameron, A. C., and Trivedi, P. K. 1998. *Regression Analysis of Count Data*. Cambridge University Press.

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

Chen, Z.; Boehm, B.; Menzies, T.; and Port, D. 2005. Finding the right data for software cost modeling. *IEEE Software* 22(6):38–46.

Cieslak, D. A.; Chawla, N. V.; and Striegel, A. 2006. Combating imbalance in network intrusion datasets. In *Proceedings of 2006 IEEE International Conference on Granular Computing*, 732– 737.

Elkan, C. 2001. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 239246.

Engen, V.; Vincent, J.; and Phalp, K. 2008. Enhancing network based intrusion detection for imbalanced data. *International Journal of Knowledge-Based and Intelligent Engineering Systems* 12(5-6):357–367.

Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27(8):861–874.

Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research* 3:1289–1305.

Hall, M. A., and Holmes, G. 2003. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering* 15(6):1437 – 1447.

Hudepohl, J. P.; Aud, S. J.; Khoshgoftaar, T. M.; Allen, E. B.; and Mayrand, J. 1996. EMERALD: Software metrics and models on the desktop. *IEEE Software* 13(5):56–60.

Ilczuk, G.; Mlynarski, R.; Kargul, W.; and Wakulicz-Deja, A. 2007. New feature selection methods for qualification of the patients for cardiac pacemaker implantation. In *Computers in Cardiology, 2007*, 423–426.

Jiang, Y.; Lin, J.; Cukic, B.; and Menzies., T. 2009. Variance analysis in software fault prediction models. In *Proceedings of the 20th IEEE International Symposium on Software Reliability Engineering*, 99–108.

Jong, K.; Marchiori, E.; Sebag, M.; and van der Vaart, A. 2004. Feature selection in proteomic pattern data with support vector machines. In *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*.

Kamal, A. H.; Zhu, X.; Pandya, A. S.; Hsu, S.; and Shoaib, M. 2009. The impact of gene selection on imbalanced microarray expression data. In *Proceedings of the 1st International Conference on Bioinformatics and Computational Biology; Lecture Notes in Bioinformatics; Vol. 5462*, 259–269.

Kira, K., and Rendell, L. A. 1992. A practical approach to feature selection. In *Proceedings of 9th International Workshop on Machine Learning*, 249–256.

Liu, H., and Yu, L. 2005. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering* 17(4):491–502.

Liu, H.; Motoda, H.; and Yu, L. 2004. A selective sampling approach to active feature selection. *Artificial Intelligence* 159(1-2):49–74.

Platt, J. C. 1999. Advances in kernel methods - support vector learning. In *Fast Training of Support Vector Machines using Sequential Minimal Optimization*, 185–208. MIT Press.

Saeys, Y.; Abeel, T.; and Peer, Y. 2008. Robust feature selection using ensemble feature selection techniques. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II (2008)*, 313–325.

Seliya, N.; Khoshgoftaar, T. M.; and Van Hulse, J. 2009. Aggregating performance metrics for classifier evaluation. In *Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI 2009)*, 35–40.

Shawe-Taylor, J., and Cristianini, N. 2000. *Support Vector Machines*. Cambridge University Press, 2 edition.

Van Hulse, J.; Khoshgoftaar, T. M.; and Napolitano, A. 2007. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th International Conference on Machine Learning*, 935–942.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2 edition.

Zhao, X.-M.; Li, X.; Chen, L.; and Aihara, K. 2007. Protein classification with imbalanced data. *Proteins: Structure, Function, and Bioinformatics* 70(4):1125 – 1132.