A Novel Feature Selection Technique for Highly Imbalanced Data

Taghi M. Khoshgoftaar
Florida Atlantic University
Boca Raton, Florida 33431
taghi@cse.fau.edu

Kehan Gao
Eastern Connecticut State University
Willimantic, Connecticut 06226
gaok@easternct.edu

Jason Van Hulse
Florida Atlantic University
Boca Raton, Florida 33431
jvanhulse@gmail.com

***Abstract*—Two challenges often encountered in data mining are the presence of excessive features in a data set and unequal numbers of examples in the two classes in a binary classification problem. In this paper, we propose a novel approach to feature selection for imbalanced data in the context of software quality engineering. This technique consists of a repetitive process of data sampling followed by feature ranking and finally aggregating the results generated during the repetitive process. This repetitive feature selection method is compared with two other approaches: one uses a filter-based feature ranking technique alone on the original data, while the other uses the data sampling and feature ranking techniques together only once. The empirical validation is carried out on two groups of software data sets. The results demonstrate that our proposed repetitive feature selection method performs on average significantly better than the other two approaches, especially when the data set is highly imbalanced.**

I. Introduction

Many software quality data sets include an overabundance of features or attributes. In the context of software quality data, features generally consist of software metrics such as the number of lines of code or the number of operands in a program module. However, not all features make the same contributions to the class. Selecting a subset of features that are most relevant to the class attribute is a necessary step in creating more meaningful and usable software quality models. In addition, for a two-group classification problem, class imbalance is frequently encountered [1], [2]. Class imbalance refers to the situation where the examples of one class are outnumbered by the examples of the other class in a data set. For instance, in software quality classification, the class often indicates whether or not the software module contain more than t faults. Fault-prone (*fp*) modules typically are much less common than not-fault-prone (*nfp*) modules. Traditional classification algorithms attempt to improve classification accuracy without considering the relative significance of the different classes, resulting in a large number of misclassifications from minority class (e.g., *fp*) to majority class (e.g., *nfp*). This type of misclassification (false negative) is extremely severe in some domains such as software quality assurance, implying a lost opportunity to correct a faulty module prior to deployment and operation.

The main contribution of this study is the proposal of a novel technique combining a feature ranking technique and data sampling (random undersampling or RUS). RUS is used to handle class imbalance because it has performed well in previous studies [3] and it is very efficient even for large datasets since it reduces the size of the training data set by undersampling the majority class. Our procedure first uses RUS to balance the two classes of a data set. Feature ranking techniques are then applied to the sampled data to rank the features according to their predictive power. In this work, we investigate six filter-based feature ranking techniques (chi-square (CS), information gain (IG), gain ratio (GR), two types of ReliefF (RF and RFW), and symmetrical uncertainty (SU)) to select a subset of attributes. In order to eliminate the biased outcome which may occur when using RUS only once, this process is repeated k times and the k different rankings are aggregated by using mean (average). Finally, the best set of features is selected from the original data to form the training data set.

The experiments of this study were carried out in the context of software quality modeling. The purpose of software quality modeling is to apply data mining techniques to software metrics collected during the software development process in order to identify the potential high-risk (fault-prone) program modules. As a result, practitioners can intelligently allocate project resources and focus more on those potentially problematic modules.

To validate the effectiveness of our proposed method, this repetitive feature selection technique was applied to two groups of software data sets, each containing three separate releases. The class distributions of the two groups of data are different, one group having 92-94% *nfp* and the other group having 71-76% *nfp*. Our proposed method is compared to two other approaches, (1) using a feature ranking technique alone and (2) using the data sampling and feature ranking techniques together once. Three different classifiers are applied to the training data sets with the attributes selected by three different approaches. The experimental results demonstrate that our proposed method performs on average significantly better than the other two approaches. Moreover, the comparison results show that the performance strength of the repetitive feature selection over other methods becomes evident when a training data set is highly imbalanced (i.e., the percentage of *nfp* examples is very high).

The remainder of the paper is organized as follows. Section II presents related work. The six filter-based feature ranking techniques and our newly proposed repetitive feature selection method, as well as the three classifiers and the associated classification performance metric used in the study are described in Section III. A case study is provided in Section IV. Finally, conclusions are drawn in Section V.

II. Related Work

Generally, feature selection is divided into two groups, *wrapper*-based and *filter*-based feature selections. For a wrapper-based technique, a learner is involved in the feature selection process. The potential problems of a wrapper-based technique lie in its high computational cost and a risk of overfitting to the model. On the other hand, for a filter-based technique, instead of a learner, the intrinsic characteristics of the data are used to assess the feature(s). Therefore, the filter method is computationally faster compared to the wrapper technique, and the selected features are determined by the data and its characteristics rather than an external learner.

Many extensive studies on feature selection have been made in data mining and machine learning during recent years. Liu and Yu [4] made a comprehensive survey of feature selection algorithms and presented an integrated approach to intelligent feature selection.

Numerous variations of feature selection have been developed and employed in a range of fields [5], [6], [7]. Jong et al. [7] introduced methods for feature selection based on support vector machines (SVM). Ilczuk et al. [6] investigated the importance of attribute selection in judging the qualification of patients for cardiac pacemaker implantation. In the context of text mining, Forman [5] investigated multiple filter-based feature ranking techniques.

Although feature selection has been widely applied in various data mining problems, its application in software quality and reliability engineering has been limited. Rodríguez et al. [8] applied feature subset selection with three filter-based models and two wrapper-based models to five software engineering data sets. The results showed that the reduced data sets maintained their prediction capability. Moreover, while it was stated that the wrapper-based model was better than the filter-based model, it came at a high computational cost. Chen et al. [9] have studied feature selection using wrappers in the context of software cost/effort estimation. They concluded that the reduced data set could improve the estimation and recommended feature selection in cost modeling, particularly when dealing with very small data sets.

Our research group recently studied various feature selection techniques including filter-based and wrapper-based methods [10], [11] and applied them to a variety of software data sets. The results demonstrate that the performances of the classification models were maintained or even improved when over 85 percent of the features were eliminated from the original data sets.

III. Methodology

A. Filter-Based Feature Ranking Techniques

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 standard 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 - χ^2 (CS) test [12] is used to examine the distribution of the class as it relates to the values of the feature in question. The null hypothesis is that there is no correlation; each value is as likely to have instances in any one class as any other class. Given the null hypothesis, the χ^2 statistic measures how far away the actual value is from the expected value:

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{n_c}\frac{(O_{i,j}-E_{i,j})^2}{E_{i,j}}$$

where r is the number of different values of the feature, n_c is the number of classes (in this work, $n_c = 2$), $O_{i,j}$ is the observed number of instances with value i which are in class j, and $E_{i,j}$ is the expected number of instances with value i and class j. The larger this χ^2 statistic, the more likely it is that the distribution of values and classes are dependent; that is, the feature is relevant to the class.

Information gain, gain ratio, and symmetrical uncertainty are measures based on the concept of entropy from information theory [13]. 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 to 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 solve 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.

Algorithm 1: Repetitive Feature Selection Algorithm

```
input :
1. Data set $D$ with features $F^j, j = 1, \ldots, m$;
2. Each instance $\mathbf{x} \in D$ is assigned to one of two classes
$c(\mathbf{x}) \in \{fp, nfp\}$;
3. Filter-based feature ranking technique $\omega \in$ {CS, GR, IG, RF, RFW,
SU} ;
4. Data sampling technique: RUS (random undersampling) ;
5. A number of repetitions $k$;
6. A predefined threshold: number (or percentage) of the features to be
selected.
output:
Ranking $\mathbb{R} = \{r^1, \ldots, r^m\}$ where $r^j$ represents the rank for attribute
$F^j$.
for $i = 1$ to $k$ do
    Use RUS to balance $D$ and get the balanced data $D_i$ ;
    Employ $\omega$ to rank features $F^j$ on $D_i$, and get new rankings
    $\omega_i(F^j), j = 1, \ldots, m$ ;
Create feature ranking $\mathbb{R}$ by combining the $k$ different rankings
$\{\omega_i(F^j) \mid i = 1, \ldots, k\}\ \forall j$ with mean (average).
Select features according to feature ranking $\mathbb{R}$ and the predefined
threshold.
```

Relief is an instance-based feature ranking technique introduced by Kira and Rendell [14]. ReliefF is an extension of the Relief algorithm that can handle noise and multiclass data sets, and is implemented in the WEKA tool [13]. 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.

B. The Repetitive Feature Selection Approach

The proposed method is designed to deal with feature selection for imbalanced data. Algorithm 1 presents the procedure of this new approach. It consists of two basic steps:

1) Using the random undersampling (RUS) technique to balance data. RUS creates the balanced data by randomly removing examples from the majority class. After sampling, the ratio between the majority (*nfp*) examples and minority (*fp*) examples is 50-50 in this study.
2) Applying a filter-based feature ranking technique to the sampled (balanced) data and ranking all the features according to their predictive powers (scores).

To alleviate the biased results generated due to the sampling process, we repeat the two steps k times ($k = 10$ in this study) and aggregate k rankings by using mean (average). Finally, the best set of attributes is selected from the original data to form the training data set.

C. Classifiers

The three learners that are selected to build the software quality classification models are naïve Bayes (NB) [13], K-nearest neighbors (KNN) [13], and support vector machine (SVM) [15]. These learners are commonly used in the software engineering domain and data mining, and also they do not possess a built-in feature selection capability. We employ the WEKA tool [13] to implement these classifiers with changes to the following default parameters. For the KNN learner, the `distanceWeighting` parameter was set to 'Weight by 1/distance', the `kNN` parameterwas set to '5', and the

TABLE I
DATA CHARACTERISTICS

Data set	Rel.	t	#Attri.	#Obj.	#fp	%fp	#nfp	%nfp
Eclipse-I	2.0	10	209	377	23	6.1	354	93.9
	2.1	5	209	434	34	7.8	400	92.2
	3.0	10	209	661	41	6.2	620	93.8
Eclipse-II	2.0	3	209	377	101	26.8	276	73.2
	2.1	2	209	434	125	28.8	309	71.2
	3.0	3	209	661	157	23.8	504	76.2

`crossValidate` parameter was turned on (set to 'true'). SVM [15] had two changes to the default parameters: the `complexity constant c` was set to '5.0' and `build Logistic Models` was set to 'true'. By default, a linear kernel was used. No changes were made to the NB learner.

D. Classification Performance Metric

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 (the positive class is synonymous with the minority class). Traditional performance metrics for classifier evaluation consider only the default decision threshold of 0.5. 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. It has also been shown that AUC is of lower variance and is more reliable than other performance metrics such as precision, recall, and F-measure [16].

IV. A CASE STUDY

A. Data Sets

Publicly available data, namely the Eclipse defect counts and complexity metrics data set obtained from the PROMISE data repository (http://promisedata.org), are used in the experiments. In particular, metrics and defects data at the software packages level are used. The original data for Eclipse packages consists of three releases denoted 2.0, 2.1, and 3.0 respectively. Each release as reported by [17] contains the following information: the name of the package for which the metrics are collected (name), the number of defects reported six months prior to release (pre-release defects), the number of defects reported six months after release (post-release defects), a set of complexity metrics computed for classes or methods and aggregated in terms of average, maximum, and total (complexity metrics), and the abstract syntax tree of the package consisting of the node size, type, and frequency (structure of abstract syntax tree(s)). For our study we transform the original data by: (1) removing all non-numeric attributes, including the package names, and (2) converting the post-release defects attribute to a binary class attribute with fault-prone (*fp*) being the minority class and not-fault-prone (*nfp*), the majority class. Membership in each class is determined by a post-release defects threshold thd, which separates *fp* from *nfp* packages by classifying packages with thd or more post-release defects as *fp* and the remaining as *nfp*. In our study, we use thd = {10, 3} for releases 2.0 and 3.0 while we use thd = {5, 2} for release 2.1. This results in two groups. Each group contains three data sets, one for each release. The reason why a different set of thresholds is chosen for release 2.1 is that we would like to keep similar class distributions for the data sets in the same group. All data sets contain 209 attributes (208 independent attributes and 1 dependent attribute). Table I shows the characteristics of the data sets after transformation for each group. These data sets exhibit different distribution of class skew (i.e., the percentage of *fp* examples).

B. Experiment Design

The main purpose of the experiment is to compare performance of the classification models when using different feature selection strategies. The three scenarios considered in the experiment are

- **Scenario 1**: Feature ranking technique used alone (denoted **NS**).
- **Scenario 2**: Data sampling followed by feature ranking (denoted **nonRep**).
- **Scenario 3**: A repetitive process of data sampling followed by feature ranking (denoted **Rep**).

Note that the training data sets generated from three scenarios are extracted from the original data set.

C. Results & Analysis

During the experiment, one parameter needs to be determined in advance, that is, how many features will be selected for modeling? In this paper, we choose $\lceil \log_2 n \rceil$ attributes that have the highest scores, where n is the number of the independent attributes in the original data set. For the two groups of Eclipse data $n = 208$, so $\lceil \log_2 n \rceil = 8$. We select $\lceil \log_2 n \rceil$ attributes, because 1) related literature does not provide guidance on the appropriate number of features to select; and 2) one of our recent empirical studies [18] recommended using $\lceil \log_2 n \rceil$ features when employing WEKA to build random forests learners for binary classification in general and imbalanced data sets in particular. Although we use different learners here, a preliminary study showed that $\lceil \log_2 n \rceil$ is still appropriate for various learners.

After feature selection, we applied three different classifiers to the training data sets with the selected attributes, and we used AUC to evaluate the performance of the classifications. All the results are reported in Table II and Table III, each representing the results for each group of the data sets. In the experiments, ten runs of five-fold cross-validation were performed. The values presented in the tables represent the average AUC for every classification model constructed over the ten runs of five-fold cross-validation. The best feature selection technique for each classifier is highlighted with **bold**. We also present the average performance (last row of the tables) for each feature selection scenario and for each learner over the six feature ranking techniques and across three releases.

The results demonstrate that the repetitive method (Rep) outperformed the other two approaches (NS and nonRep) for all the cases on Eclipse-I except the case where the IG ranker and the KNN learner were used. For Eclipse-II, the repetitive method performed better than or equal to the other two approaches for 11 out of 18 cases, although the repetitive method was better based on the average performances for all three scenarios.

We also performed a three-way ANalysis Of VAriance (ANOVA) F test [19] on the classification performance for each group of the data sets (Eclipse-I and Eclipse-II) separately to examine if the performance difference (better/worse) is statistically significant or not. Three factors are designed as follows: Factor A represents six feature ranking techniques (CS, IG, GR, RF, RFW and SU), Factor B represents three classifiers (NB, KNN, and SVM), and Factor C represents the three scenarios designed in the experiment (NS, nonRep, and Rep). The null hypothesis for the ANOVA test is that all the group population means are the same while the alternate hypothesis is that at least one pair of means is different. In addition, the interactions between each pair of factors as well as for all three factors are also examined in the test. Table IV and Table V show the

TABLE II
CLASSIFICATION PERFORMANCE IN TERMS OF AUC FOR ECLIPSE-I

	NB			KNN			SVM		
	NS	nonRep	Rep	NS	nonRep	Rep	NS	nonRep	Rep
CS	0.8355	0.8471	**0.8542**	0.8829	0.8841	**0.8919**	0.8911	0.8958	**0.9038**
GR	0.8152	0.8460	**0.8553**	0.8337	0.8751	**0.8940**	0.8604	0.8946	**0.9077**
IG	0.8527	0.8476	**0.8589**	**0.9003**	0.8799	0.8966	0.9029	0.8965	**0.9076**
RF	0.8174	0.8488	**0.8858**	0.8275	0.8771	**0.8931**	0.8519	0.8768	**0.8957**
RFW	0.8132	0.8126	**0.8603**	0.7752	0.8199	**0.8797**	0.8523	0.8569	**0.8923**
SU	0.8323	0.8458	**0.8610**	0.8689	0.8806	**0.8982**	0.8820	0.8943	**0.9098**
Avg.	0.8277	0.8413	**0.8626**	0.8481	0.8694	**0.8922**	0.8734	0.8858	**0.9028**

TABLE III
CLASSIFICATION PERFORMANCE IN TERMS OF AUC FOR ECLIPSE-II

	NB			KNN			SVM		
	NS	nonRep	Rep	NS	nonRep	Rep	NS	nonRep	Rep
CS	**0.7877**	0.7830	0.7857	**0.8618**	0.8531	0.8562	**0.8852**	0.8847	**0.8852**
GR	0.7682	0.7668	**0.7753**	**0.8419**	0.8122	0.8126	0.8669	0.8650	**0.8714**
IG	**0.7889**	0.7842	0.7872	**0.8600**	0.8547	0.8558	0.8861	0.8849	**0.8862**
RF	0.7879	0.8032	**0.8079**	0.7530	0.7896	**0.8058**	0.8424	0.8706	**0.8735**
RFW	0.8032	0.8044	**0.8065**	0.7605	0.7882	**0.8070**	0.8559	0.8728	**0.8751**
SU	**0.7854**	0.7801	0.7844	0.8562	0.8517	**0.8579**	0.8851	0.8821	**0.8858**
Avg.	0.7869	0.7869	**0.7912**	0.8222	0.8249	**0.8325**	0.8703	0.8767	**0.8795**

TABLE IV
THREE-WAY ANOVA FOR ECLIPSE-I

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	0.3017	5	0.0603	51.09	0
B	0.5172	2	0.2586	218.96	0
C	0.3546	2	0.1773	150.14	0
A×B	0.0861	10	0.0086	7.29	0
A×C	0.1739	10	0.0174	14.72	0
B×C	0.0106	4	0.0027	2.25	0.062
A×B×C	0.0456	20	0.0023	1.93	0.008
Error	1.8495	1566	0.0012		
Total	3.3392	1619			

TABLE V
THREE-WAY ANOVA FOR ECLIPSE-II

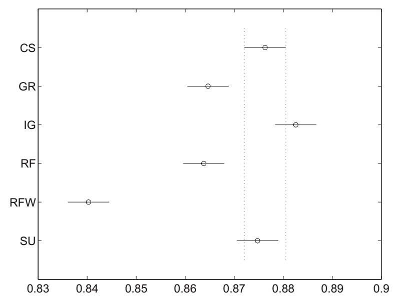
Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	0.2407	5	0.0481	61.09	0
B	2.0619	2	1.0309	1308.37	0
C	0.0174	2	0.0087	11.02	0
A×B	0.4472	10	0.0447	56.76	0
A×C	0.0746	10	0.0075	9.47	0
B×C	0.0032	4	0.0008	1.01	0.402
A×B×C	0.0354	20	0.0018	2.25	0.001
Error	1.2339	1566	0.0008		
Total	4.1144	1619			

ANOVA results for Eclipse-I and Eclipse-II data sets respectively. The p-value is less than the typical cutoff 0.05 for each main factor and most interactions of each group of data sets except interaction B×C, where p-value is greater than 0.05 for both groups of data. This means that for each main factor, the alternate hypothesis is accepted, namely, at least two group means are significantly different from each other. In addition, the interactions (A×B, A×C and A×B×C) significantly affect classification performance in each group of data sets. In other words, changing value of one factor will significantly influence the value of the other factor(s), and vice versa.

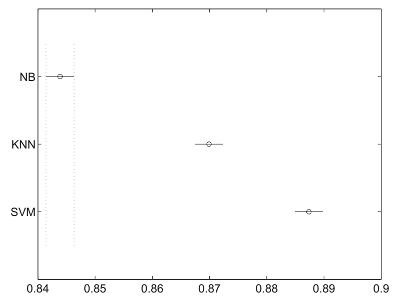
We further carried out a multiple comparison test [19] on each main factor and the interactions (A×C and B×C) with Tukey's honestly significant difference criterion. Since this study is more interested in comparing the performances of the three different feature selection scenarios, we only present the pairwise interactions that involve the three scenarios (Factor C). Note that for all the ANOVA and multiple comparison tests, the significance level α was set to 0.05.

Figures 1 and 2 show the multiple comparisons on the two groups of data sets, each with five subfigures representing Factor A, B and C, and interactions A×C and B×C, respectively. The figures display graphs with each group mean represented by a symbol (○) and 95% confidence interval as a line around the symbol. Two means are significantly different if their intervals are disjoint, and are not significantly different if their intervals overlap. Matlab was used to construct the ANOVA models and perform the multiple comparisons presented in this work, and the assumptions for constructing ANOVA models were validated. From these figures we conclude the following:

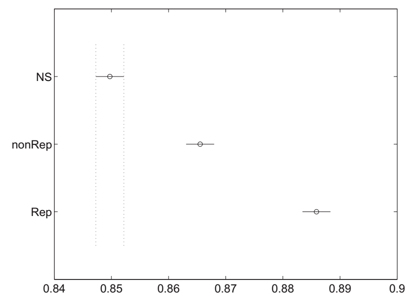
- Among the six standard filter-based feature selection methods, IG performed best, followed by CS and SU; the other three techniques, GR, RF, and RFW, performed significantly worse than the first three. This is true for both groups of data sets. However, the significance levels are a bit different with respect to the different class distribution of each group of data sets. For example, the distinction between the pair of IG and CS can be clearly seen in Fig. 1(a) but not in Fig. 2(a).
- Of the three classifiers, SVM performed best, followed by KNN; NB performed worst.
- For the three feature selection strategies, our proposed repetitive feature selection method performed best followed by nonRep; NS performed worst. This pattern is reflected by both groups of data sets. However, a slight difference exists between groups of data. In particular, nonRep significantly outperformed NS for Eclipse-I but insignificantly outperformed NS for Eclipse-II.
- There are 18 groups for interaction A×C; these are formed by each of six feature selection techniques being combined with three feature selection strategies. The group means demonstrate that in addition to the two main factors, the classification performance is greatly influenced by their interactions. For example, for Eclipse-1, IG significantly outperformed GR for NS, but this pattern was not observed when nonRep and Rep strategies were adopted (see Fig. 1(d)). Also, the figures display different performance distributions of the six filter-based rankers with respect to the different feature selection strategies. When data sampling was not involved in feature selection (NS), the six rankers exhibit quite different performances. On the other hand,



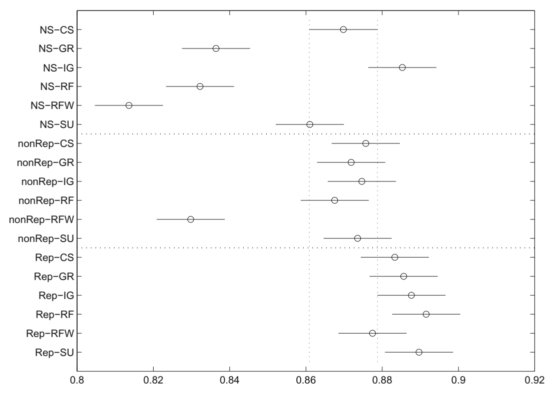
(a) Factor A



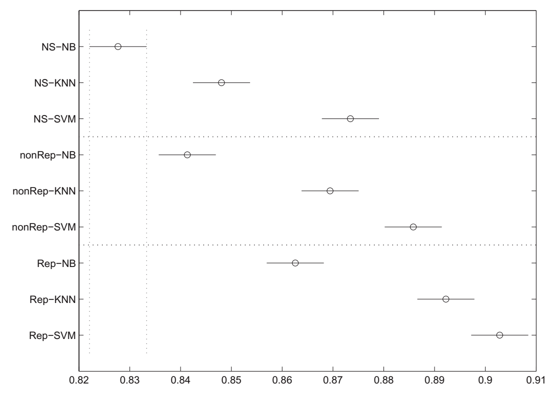
(b) Factor B



(c) Factor C



(d) Factor A$\times$C



(e) Factor B$\times$C

Fig. 1. Eclipse-I: Multiple Comparison

the six rankers demonstrate relatively consistent performances when data sampling was performed with feature selection (non-Rep and Rep). This pattern is especially obvious for Eclipse-I.

- For interaction B$\times$C, there are nine groups that are formed by each of three learners being combined with three feature selection strategies. The group means show that the two main factors played an important role in the classification performance while their interactions did not.
- One point that is clearly observed in the multiple comparisons is that the performance advantage of the repetitive feature selection approach over other methods becomes increasingly dramatic as the class imbalance becomes more serious (i.e., the percentage of *fp* examples becomes increasing low). This may imply that our proposed method is especially appropriate when classification occurs for highly imbalanced data (e.g., the percentage of *fp* examples is less than 10%).

Our recent work [20] has shown that classification models built on smaller subsets of attributes via the six filter-based feature ranking techniques had similar or better performances than those built with a complete set of attributes. Thus, we did not present the results for full data sets in this paper.

V. Conclusion

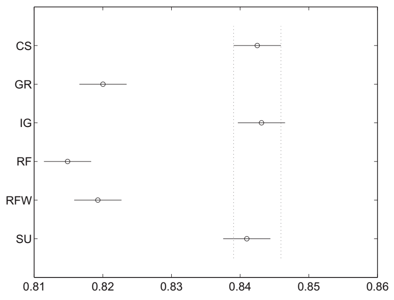
Feature selection for highly imbalanced data is a major challenge in software quality modeling. We have proposed a repetitive feature selection method to deal with this problem. To evaluate the effectiveness of the proposed method, we applied our technique to two groups of software data sets, each group having three separate releases. We used three different classifiers to build a number of classification models with the selected attributes. We also compared the new method to two other approaches - (i) feature ranking used alone and (ii) data sampling and feature ranking used together but only once. The experimental results demonstrate that (1) data sampling improves feature selection when the data sets have unequal numbers of examples in the two classes; (2) our proposed method performed significantly better than the other two approaches on average; and (3) the performance strength of our proposed method over other methods becomes increasingly evident as the class imbalance becomes more serious. The conclusions obtained in this paper regarding the effectiveness of the proposed feature selection approach are based on the experiments conducted on the data sets from this specific software system. Future work will involve more experiments on the data sets from different domains. Moreover, different data sampling techniques can also be considered in the repetitive feature selection process.

References

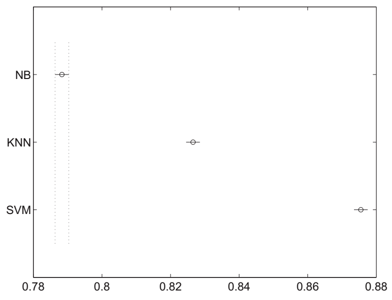
[1] X.-M. Zhao, X. Li, L. Chen, and K. Aihara, "Protein classification with imbalanced data," *Proteins: Structure, Function, and Bioinformatics*, vol. 70, no. 4, pp. 1125 – 1132, 2007.

[2] V. Engen, J. Vincent, and K. Phalp, "Enhancing network based intrusion detection for imbalanced data," *International Journal of Knowledge-Based and Intelligent Engineering Systems*, vol. 12, no. 5-6, pp. 357–367, 2008.

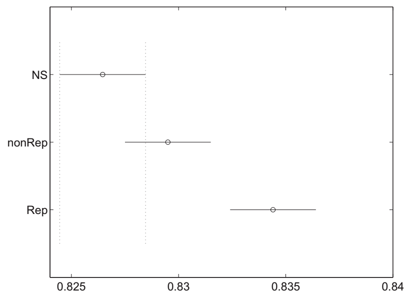
[3] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proceedings of the*



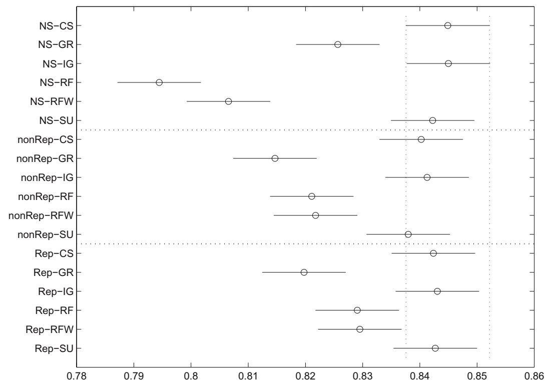
(a) Factor A



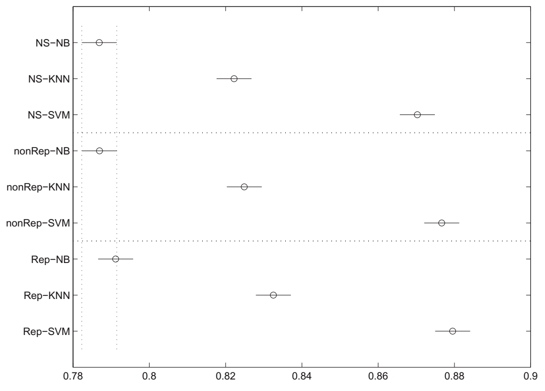
(b) Factor B



(c) Factor C



(d) Factor A×C



(e) Factor B×C

Fig. 2. Eclipse-II: Multiple Comparison

24th International Conference on Machine Learning, Corvallis, OR, USA, June 2007, pp. 935–942.

[4] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005.

[5] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of Machine Learning Research*, vol. 3, pp. 1289–1305, March 2003.

[6] G. Ilczuk, R. Mlynarski, W. Kargul, and A. Wakulicz-Deja, "New feature selection methods for qualification of the patients for cardiac pacemaker implantation," *Computers in Cardiology*, vol. 34, no. 2-3, pp. 423–426, 2007.

[7] K. Jong, E. Marchiori, M. Sebag, and A. van der Vaart, "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*, Oct 7-8 2004.

[8] D. Rodriguez, R. Ruiz, J. Cuadrado-Gallego, and J. Aguilar-Ruiz, "Detecting fault modules applying feature selection to classifiers," in *Proceedings of 8th IEEE International Conference on Information Reuse and Integration*, Las Vegas, Nevada, August 13-15 2007, pp. 667–672.

[9] Z. Chen, T. Menzies, D. Port, and B. Boehm, "Finding the right data for software cost modeling," *IEEE Software*, vol. 22, no. 6, pp. 38–46, November 2005.

[10] K. Gao, T. M. Koshogoftaar, and A. Napolitano, "Exploring software quality classification with a wrapper-based feature ranking technique," in *Proceedings of 21st IEEE International Conference on Tools with Artificial Intelligence*, Newark, New Jersey, Nov. 2-4 2009, pp. 67–74.

[11] T. M. Khoshgoftaar, L. A. Bullard, and K. Gao, "Attribute selection using rough sets in software quality classification," *International Journal of Reliability, Quality and Safty Engineering*, vol. 16, no. 1, pp. 73–89, 2009.

[12] R. L. Plackett, "Karl pearson and the chi-squared test," *International Statistical Review*, vol. 51, no. 1, p. 5972, 1983.

[13] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.

[14] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of 9th International Workshop on Machine Learning*, 1992, pp. 249–256.

[15] J. Shawe-Taylor and N. Cristianini, *Support Vector Machines*, 2nd ed. Cambridge University Press, 2000.

[16] Y. Jiang, J. Lin, B. Cukic, and T. Menzies., "Variance analysis in software fault prediction models," in *Proceedings of the 20th IEEE International Symposium on Software Reliability Engineering*, Bangalore-Mysore, India, Nov. 16-19 2009, pp. 99–108.

[17] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *Proceedings of the 29th International Conference on Software Engineering Workshops*. Washington, DC, USA: IEEE Computer Society, 2007, p. 76.

[18] T. M. Khoshgoftaar, M. Golawala, and J. Van Hulse, "An empirical study of learning from imbalanced data using random forest," in *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, vol. 2, Washington, DC, USA, 2007, pp. 310–317.

[19] M. L. Berenson, M. Goldstein, and D. Levine, *Intermediate Statistical Methods and Applications: A Computer Package Approach*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[20] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: An investigation on feature selection techniques," Florida Atlantic University, Tech. Rep. FAU-CSE-TR-2009-11-2, November 2009.