2011 23rd IEEE International Conference on Tools with Artificial Intelligence

# Impact of Data Sampling on Stability of Feature Selection for Software Measurement Data

Kehan Gao*
Taghi M. Khoshgoftaar†
Amri Napolitano‡

## Abstract

*Software defect prediction can be considered a binary classification problem. Generally, practitioners utilize historical software data, including metric and fault data collected during the software development process, to build a classification model and then employ this model to predict new program modules as either fault-prone (**fp**) or not-fault-prone (**nfp**). Limited project resources can then be allocated according to the prediction results by (for example) assigning more reviews and testing to the modules predicted to be potentially defective. Two challenges often come with the modeling process: (1) high-dimensionality of software measurement data and (2) skewed or imbalanced distributions between the two types of modules (fp and nfp) in those datasets. To overcome these problems, extensive studies have been dedicated towards improving the quality of training data. The commonly used techniques are feature selection and data sampling. Usually, researchers focus on evaluating classification performance after the training data is modified. The present study assesses a feature selection technique from a different perspective. We are more interested in studying the stability of a feature selection method, especially in understanding the impact of data sampling techniques on the stability of feature selection when using the sampled data. Some interesting findings are found based on two case studies performed on datasets from two real-world software projects.*

Keywords: *feature selection, data sampling, software metrics, defect prediction, stability*

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

†Taghi M. Khoshgoftaar is with Florida Atlantic University, Boca Raton, Florida 33431, khoshgof@fau.edu.

‡Amri Napolitano is with Florida Atlantic University, Boca Raton, Florida 33431, amrifau@gmail.com.

## 1. Introduction

Software metrics collected during the software development process may contain a great deal of valuable information that can help software quality prediction. A general process is that practitioners utilize the collected metrics and data mining algorithms to build a classification model and then apply this model to predict the quality of new program modules (e.g., classify the program modules as either fault-prone (*fp*) or not-fault-prone (*nfp*)) [17]. As a result, the project resources can be reasonably allocated according to the prediction. For example, more inspection and testing goes to the potentially problematic modules, thereby improving the quality of the product.

Two challenges often arise during the modeling process: (1) high-dimensionality is found in most software data repositories, and (2) class imbalance occurs for those datasets. High-dimensionality occurs when the number of available software metrics is too large for a classifier to work well. A number of problems may arise due to high-dimensionality, such as extensive computation and a decline in predictive performance. Class imbalance occurs when one of the classes in a dataset is greatly outnumbered by instances of the other classes. This problem is more prevalent in high-assurance and mission-critical software systems, where the proportion of *nfp* modules is relatively larger compared to that of the *fp* modules. For a binary classification problem, there exist two types of misclassifications, called Type I and Type II. A Type I misclassification (error) occurs when a *nfp* module is classified as *fp*, while a Type II misclassification occurs when a *fp* module is classified as *nfp*. A Type II error is more costly than a Type I error, as customer-discovered faults have very serious consequences and are very expensive to repair. The primary drawback of imbalanced data is that a traditional classification algorithm tends to misclassify *fp* modules as *nfp*, causing more Type II errors.

Feature selection and data sampling are often employed to alleviate these problems. Feature selection (FS) is a process of choosing the most important features that can best describe the class attribute. Data sampling is a common technique to counter the class imbalanced problem, where the training dataset is sampled to alter the relative distribution of the *nfp* and *fp* modules before FS is performed. To evaluate a FS technique, most previous research focuses on comparing the performance of classification models before and after a specific FS technique is performed. In this paper, we assess FS techniques from a different perspective. We are more interested in studying the stability (robustness) of FS, which has not received much attention in the past.

We would like to investigate which FS techniques are relatively robust to variations in the input data, more specifically, to examine the stability of FS techniques with respect to various data sampling approaches. The strategy we adopted is that the ranking from each sampled dataset is compared to the ranking from the original dataset which it came from. Those FS techniques whose outputs are insensitive to the different perturbations (due to sampling) in the input data are said to be stable (robust), and they are favored over those that produce inconsistent outputs. To our knowledge, this is the first study to investigate the impact of data sampling on the stability of feature selection. The experiments are carried out on seven datasets from two real-world software projects. We examine six commonly used filter-based FS techniques and three data sampling approaches, each combined with two post-sampling proportion ratios. Some interesting findings are obtained from the experiments.

The remainder of the paper is organized as follows: Section 2 discusses related work. Section 3 outlines the methods and techniques used in this paper. Section 4 describes two groups of software measurement datasets used in the case studies. Section 5 presents two case studies including design, results, and analysis. Finally, we summarize our conclusions and provide suggestions for future work in Section 6.

## 2. Related Work

Feature selection (FS), also known as attribute selection, is a process of selecting some subset of the features which are useful in building a classifier. It removes as many unnecessary features as possible, leaving only those features that are important to the class attribute. FS techniques can be divided into *wrappers*, *filters*, and *embedded* categories [18]. Wrappers use a search algorithm to search through the space of possible features and determine which ones are finally selected in building a classifier, evaluating each subset through a learning algorithm. Filters use a simpler statistical measure to evaluate each subset or individual feature rather than using a learning algorithm. Embedded approaches, like the wrapper methods, are also linked to a learning algorithm, however this link is much stronger compared to wrappers, as FS is included into the classifier construction in this case. Feature selection may also be categorized as *ranking* or *subset selection* [8]. Feature ranking scores the attributes based on their individual predictive power, while subset selection selects subset of attributes that collectively have good prediction capability. In this study, the FS techniques used belong to the *filter-based feature ranking* category.

Numerous variations of FS have been employed in a range of fields [10, 11]. Jong et al. [11] introduced methods for FS based on support vector machines (SVM). Ilczuk et al. [10] investigated the importance of attribute selection in judging the qualification of patients for cardiac pacemaker implantation. Class imbalance, which appears in various domains [7, 13], is another significant problem in data mining. One effective method for alleviating the adverse effect of skewed class distribution is sampling, which will add or remove instances from a data set until it becomes more balanced with respect to its class distribution [3, 22]. While considerable work has been done for feature selection and data sampling separately, limited research can be found on investigating them both together, particularly in the software engineering field. Chen et al. [4] have studied data row pruning (data sampling) and data column pruning (feature selection) in the context of software cost/effort estimation. However, the data sampling in their study was not specific for the class imbalance problem, and unlike this study, the classification models are meant for non-binary problems.

To evaluate FS techniques, most existing research works on comparing the classification behaviors of models built with the selected features to those built with the complete set of features. Instead of using classification performance, the present work assesses FS techniques using a different method: we are concerned more with the stability of FS.

The stability of a FS method is normally defined as the degree of consensus between the output of that FS method as it pertains to randomly-selected subsets of the same input data [15, 19, 20]. Lustgarten et al. [20] presented an adjusted stability measure that computes robustness of a feature selection method with respect to random feature selection. Kalousis et al. [12] examined three measures to assess the stability of feature preferences and conducted the experiment with high dimensional data from three different application domains, namely proteomics, genomics, and text mining. Saeys et al. [21] assessed the robustness of feature selection techniques using the Spearman rank correlation coefficient and Jaccard index. Dunne et al. [6] discussed the instability of the wrapper approach to FS and recommended a measure based on the Hamming distance to evaluate the stability of a FS technique. Loscalzo et al. [19] demonstrated a strong dependency between the sample size (in terms of number of instances in a dataset) and the stability of a FS method. Abeel et al. [1] studied the process for selecting biomarkers from microarray data and presented a general framework for stability analysis of such FS techniques. They showed that stability could be improved through ensemble FS.

## 3. Methodology

### 3.1. Filter-Based Feature Ranking Techniques

The procedure of filter-based feature ranking is to score each feature (attribute) according to a particular method (metric), allowing the selection of the best set of features. In this study, we used six commonly used 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). The chi-square, $\chi^2$, test is used to examine the distribution of the class as it relates to the values of the given feature. The null hypothesis is that there is no correlation, i.e., each value is as likely to have instances in any one class as any other class. Given the null hypothesis, the $\chi^2$ 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 $\chi^2$ statistic, the more likely it is that the distribution of values and classes are dependent, i.e., the feature is relevant to the class.

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.

Relief is an instance-based feature ranking technique introduced by Kira and Rendell [14]. It measures the importance of features by considering how much their values change when comparing a randomly chosen instance with its nearest hit (an instance from the same class) and its nearest miss (one from a different class). ReliefF is an extension of the Relief algorithm that can handle noise and multiclass datasets, and is implemented in the WEKA tool [24]. 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.

### 3.2. Data Sampling Techniques

We here present three common data sampling techniques, each of which has been shown to be effective at improving classification performance in previous research [22, 23]. These three techniques represent the major paradigms in data sampling: random and intelligent under and oversampling.

1 Random Sampling Techniques

The two most common (largely due to their simplicity) data sampling techniques used in this work are *random oversampling* (ROS) and *random undersampling* (RUS). Random oversampling duplicates instances (selected randomly) of the minority class. While this does help to balance the class distribution, no new information is added to the dataset and this may lead to overfitting [5]. Also, the size of the training datasets is increased, which causes longer model training times. Random undersampling randomly discards instances from the majority class. In doing so, the class distribution can be balanced, but important information may be lost when examples are discarded randomly.

2 Synthetic Minority Oversampling Technique (SMO)

Chawla et al. proposed an intelligent oversampling method called Synthetic Minority Oversampling Technique (SMOTE) [3]. SMOTE (denoted SMO in this work) adds new, artificial minority examples by extrapolating between preexisting minority instances rather than simply duplicating original examples. The newly created instances cause the minority regions of the feature-space to be fuller and more general. This sampling process causes a classifier to learn a larger and more general decision region in the feature-space, ideally alleviating the problems caused by class imbalance.

### 3.3. Stability Measure

Previous works have employed different similarity measures to evaluate the stability of feature selection techniques. The measures used include correlation coefficient [12], consistency index [15], Hamming distance [6], and entropy [16]. Among these similarity measures, consistency index is the only one which considers bias due to

**Table 1. Dataset summary**

| Project | Data | thd | #Attri. | nfp # | nfp % | fp # | fp % | Total # |
|---|---|---|---|---|---|---|---|---|
| LLTS | Rel.1 | 1 | 42 | 3420 | 93.7 | 229 | 6.3 | 3649 |
| | Rel.2 | 1 | 42 | 3792 | 95.3 | 189 | 4.7 | 3981 |
| | Rel.3 | 1 | 42 | 3494 | 98.7 | 47 | 1.3 | 3541 |
| | Rel.4 | 1 | 42 | 3886 | 97.7 | 92 | 2.3 | 3978 |
| Eclipse | E2.0 | 10 | 208 | 354 | 93.9 | 23 | 6.1 | 377 |
| | E2.1 | 5 | 208 | 400 | 92.2 | 34 | 7.8 | 434 |
| | E3.0 | 10 | 208 | 620 | 93.8 | 41 | 6.2 | 661 |

chance. For this reason, we use consistency index in this study. Assuming that the original dataset has $m$ instances, each containing $n$ features, and $T_i$ and $T_j$ are two subsets of the $n$ features such that $|T_i| = |T_j| = k$, where $0 < k < n$, the consistency index [15] is defined as follows:

$$I_C\left(T_i, T_j\right) = \frac{dn - k^2}{k\left(n - k\right)} \ , \qquad (1)$$

where $d$ is the cardinality of the intersection between subsets $T_i$ and $T_j$, i.e., $d = |T_i \cap T_j|$, and $I_C\left(T_i, T_j\right) \in (-1, 1]$. The greater the value of $I_C$, the higher the similarity between the subsets $T_i$ and $T_j$.

## 4. Data Description

The experiments are performed based on two groups of software measurement data. Following is an introduction for each of them.

The first group of data is 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 [9]. 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 datasets 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, which is 1 in this study. In other words, modules with one or more faults are considered as *fp*, *nfp* if modules are fault-free.

The second group of data is obtained from the publicly available PROMISE software project data repository [2]. We study the software measurement datasets for the Java-based Eclipse project [25]. The software metrics and defect data are aggregated at the software packages level; hence, a program module is a Java package in Eclipse. We consider three releases of the Eclipse system, where the releases are denoted as 2.0, 2.1, and 3.0 [25]. Each system release contains the following information [25]: name of the package for which the metrics are collected (*name*), number of defects reported six months prior to release (*pre-release defects*), number of defects reported six months after release (*post-release defects*), a set of complexity metrics computed for classes or methods and aggregated by using average, maximum, and total at the package level (*complexity metrics*), and structure of abstract syntax tree(s) of the package consisting of the node size, type, and frequency (*structure of abstract syntax tree(s)*). We modify 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. A program module's membership in a given class is determined by a post-release defects threshold, $thd$. A program module (package) with $thd$ or more post-release defects is labeled as *fp*, while those with fewer than $thd$ defects are labeled as *nfp*. We consider the following values of $thd$ for the three releases: {10, 5, 10}. A different threshold is chosen for release 2.1 because we want to maintain relatively similar types of class distributions as those of Releases 2.0 and 3.0.

Table 1 summarizes the numbers of the *fp* and *nfp* modules and their percentages in each dataset.

## 5. Two Case Studies

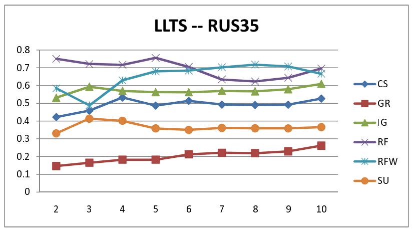
The experiment is performed on each individual dataset separately, but the results are analyzed and summarized over the respective groups of datasets.

### 5.1. Experimental Design

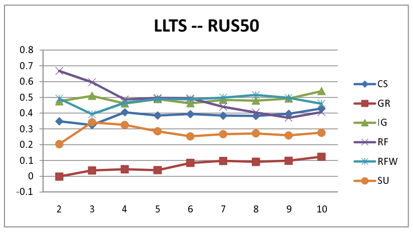
As mentioned earlier, the main purpose of this paper is to study the impacts of the data sampling techniques on the stability of feature selection. In the experiments, we examine

- six different sampling approaches (RUS35, RUS50, ROS35, ROS50, SMO35, and SMO50), in which three different sampling techniques are used, each in conjunction with two different post-sampling proportion ratios, where 35 refers the proportion ratio between *fp* and *nfp* modules being 35:65, while 50 refers to the proportion ratio being 50:50; and

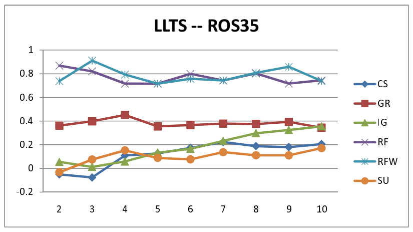
- six standard feature ranking techniques (CS, GR, IG, RF, RFW, and SU).



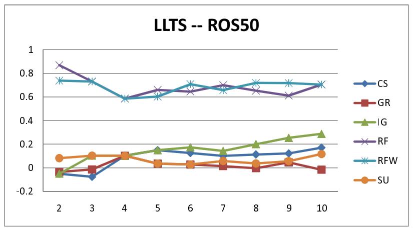
(a) RUS35



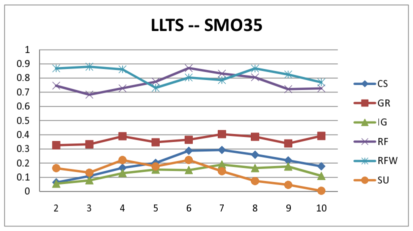
(b) RUS50



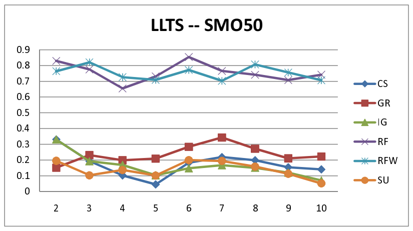
(c) ROS35



(d) ROS50



(e) SMO35



(f) SMO50

**Figure 1. LLTS data: KI Values**

The procedure of the experiments consists of the following steps. For a given dataset and a given data sampling technique,

1. Apply the sampling technique to the original dataset, $D$, and get a sampled data, $D_i$. To avoid a biased result, repeat the sampling procedure $x$ times, where $x = 30$. Therefore, $x$ datasets of same size (same number of data instances), $\{D_1, D_2, ..., D_x\}$, are generated;

2. Apply a particular feature ranking technique to every sampled data, $D_i$, to obtain the feature subsets, $T_i^k$, where $k$ represents the number of the features retained in each feature subset, i.e., $|T_i^k| = k$, $k = 2, 3, ..., 10$ and $i = 1, 2, ..., x$;

3. Apply the same feature ranking technique to that original dataset to obtain the feature subsets, $T_0^k$, where $k = 2, 3, ..., 10$;

4. For a given $k$ value, calculate the single stability index, $KI$, by the following formula:

$$KI = \frac{1}{x}\sum_{i=1}^{x} I_C\left(T_0^k, T_i^k\right), \qquad (2)$$

where $k = 2, 3, ..., 10$, and $I_C$ is the consistency index defined in Formula (1). This $KI$ is the average of consistency index over $x$ pairs of the feature subsets, each pair selected from the original dataset and one of the $x$ sampled datasets. Note that the use of $I_C$ here is different than the traditional consistency measure, but the consistency index, $I_C$, is still a core component of the measure. Therefore, we retain the name.

Note that in the feature selection process (on steps 2 and 3), a key step is to select a subset of features according to their relevance to the class. In this study, nine subsets are chosen for each dataset. The number of the features retained in each subset varies from 2 to 10, that is, $k = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Preliminary experiments conducted on the corresponding datasets show that these numbers are appropriate. As we have seven datasets, six data sampling approaches and six feature rankers, we repeat this process (steps 1 through 4) 252 times.

## 5.2. Results on LLTS Data

The experiments were performed with the six filter-based feature ranking techniques on four LLTS datasets. Aggregated results in terms of the stability index ($KI$) are shown in Figure 1. This figure consists of six charts, each displaying for a given sampling approach how the average stability performance ($y$-axis) of each ranker is affected by the sizes of nine different feature subsets ($x$-axis), averaged over four datasets. Some points are observed from the graphics.

- Among the six rankers, RF and RFW show the best stability behavior regardless of which sampling technique is performed prior to the feature selection. This is especially obvious when the training data is modified by the ROS or SMO techniques. On the other hand, other rankers present different stability with respect to various sampling techniqes. For example, GR shows least stability with the RUS techniques but moderate behavior with ROS35 and SMO35.

- The size of feature subset influences the stability of a feature ranking technique. An interesting finding is

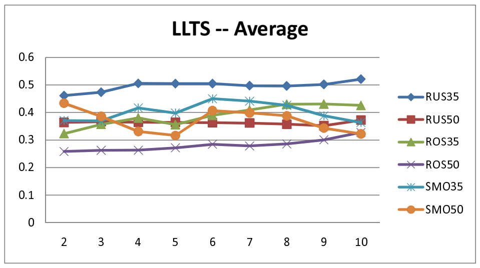
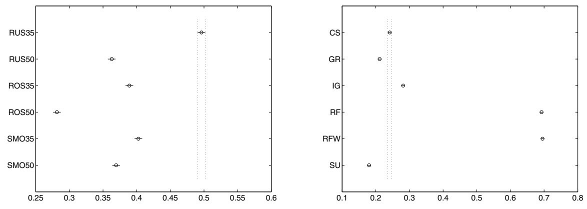


**Figure 2. LLTS data: average KI Values**

| Source | Sum Sq. | d.f. | Mean Sq. | F | $p$-value |
|---|---|---|---|---|---|
| A | 156.30 | 5 | 31.26 | 596.40 | 0 |
| B | 1906.39 | 5 | 381.28 | 7274.42 | 0 |
| Error | 2037.26 | 38869 | 0.05 | | |
| Total | 4099.94 | 38879 | | | |

(a) Two-way ANOVA



(b) Factor A: sampling techniques (c) Factor B: rankers

**Figure 3. LLTS: multiple comparison**

that when the number of features retained is four or greater, the change of the stability index ($KI$)is relatively minor.

Figure 2 shows how the averaged stability of each sampling technique is affected by the size of different feature subsets, across four datasets and six rankers. The results demonstrate that

- Among the six sampling approaches, RUS35 shows the highest stability across all different sizes of feature subsets, while ROS50 shows the lowest stability. All other sampling techniques performed averagely.
- Between the two post-sampling proportion ratios, 35:65 shows better stability behavior than 50:50. This is true for RUS and ROS across all different sizes of feature subsets, and for SMO when the size of feature subset is four or greater.

We also performed a two-way ANOVA (analysis of variance) F test to examine the stability of FS over all four datasets. The two factors in the test are designed as follows: Factor A, representing six sampling approaches, and Factor B, representing six feature ranking techniques. The null hypothesis for the ANOVA test is that all the group population means are the same and the alternate hypothesis is that at least one pair of means is different. If the ANOVA result shows to accept the alternate hypothesis, a multiple comparison test can be used to detect which group mean is different from one another. Figure 3 shows the ANOVA and multiple comparison results of this study. Fig 3 (a) presents the ANOVA outcome. The top two $p$-values are 0s ($\leq 0.05$), implying that for both main factors (A and B), at least two group means are significantly different from each other. The table is followed by the two charts, each displaying a graph with each group mean represented by a symbol ($\circ$) and the 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. The multiples comparison outcomes show that

- Among the six sampling techniques, RUS35 shows the best stability behavior, followed by SMO35 and ROS35, then SMO50 and RUS50. ROS50 shows the worst stability.
- Between the two post-sampling proportion ratios, 35:65 shows better stability than 50:50.
- Among the six filter-based rankers, RF and RFW show much better stability behavior than the other four techniques. For these four inferior techniques, the order in terms of their stabilities ranked from highest to lowest is IG, CS, GR, and SU. All of their $KI$ values are significantly different from one another.

## 5.3. Results on Eclipse Data

The same experimental process performed for the LLTS data was also carried out for the Eclipse datasets. Although due to space considerations we do not show the average stability of each ranker across different sizes of feature subsets for a given sampling technique as we did in LLTS (Figure 1), we present our findings based on the experimental results as follows.

- Among the six rankers, RF and RFW still show the best stability regardless of which sampling technique is performed prior to the feature selection. This is once again obvious when the training data is balanced by the ROS or SMO techniques. On the other hand, the CS ranker presents worst stability for every sampling technique. All other techniques display an average stability with respect to various sampling techniques.
- The size of a feature subset affects the stability of a feature ranking technique. For most rankers, the stability is increased by an increased number of features in the selected subset.

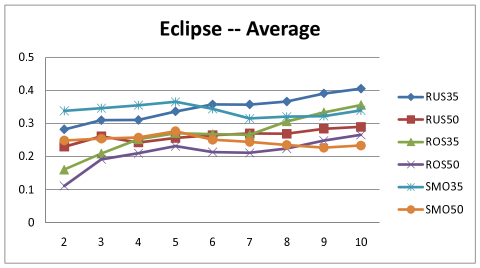
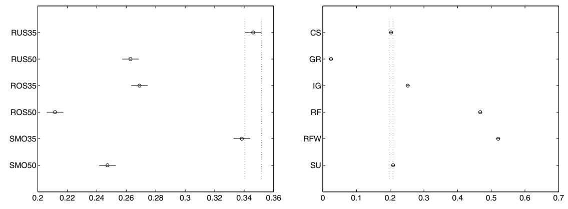


**Figure 4. Eclipse data: average KI Values**

| Source | Sum Sq. | d.f. | Mean Sq. | F | $p$-value |
|---|---|---|---|---|---|
| A | 67.68 | 5 | 13.54 | 352.79 | 0 |
| B | 826.65 | 5 | 165.33 | 4309.02 | 0 |
| Error | 1118.40 | 29149 | 0.04 | | |
| Total | 2012.73 | 29159 | | | |

(a) Two-way ANOVA



(b) Factor A: sampling techniques (c) Factor B: rankers

**Figure 5. Eclipse: multiple comparison**

In this case study, we also present the average stability of each sampling technique across different sizes of feature subsets, as shown in Figure 4. The results demonstrate that

- Among the six sampling approaches, RUS35 and SMO35 show better stability than other techniques, while ROS50 once again shows the lowest stability. All other sampling techniques are moderate.
- Between the two post-sampling proportion ratios, 35:65 shows better stability behavior than 50:50. This is true for all three sampling techniques.

We also performed the same two-way ANOVA test for this study. The two factors in the test are defined the same way as in the previous case study. Figure 5 shows the ANOVA and multiple comparison results of this study. The ANOVA table, as shown in Fig 5 (a), demonstrates that for both main factors (A and B), at least two group means are significantly different from each other. The following two charts show the multiple comparisons for Factor A and Factor B respectively. The results demonstrate that

- Among the six sampling techniques, RUS35 and SMO 35 show better stability behavior than other techniques, followed by ROS35 and RUS50, then SMO50, and finally ROS50.

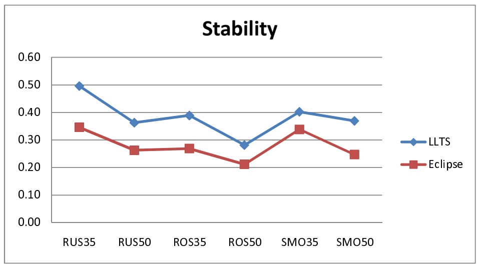


**Figure 6. Stability of LLTS and Eclipse data**

- Between the two post-sampling proportion ratios, 35:65 shows significantly better stability than 50:50 for every sampling technique.
- Among the six filter-based rankers, RF and RFW once again show much better stability performance than other techniques, and RFW is better than RF. For the remaining four techniques, the order in terms of their stability indexes from highest to lowest is IG, SU, CS, and GR.

Finally, we use Figure 6 to summarize the stability of FS affected by the data sampling techniques for both groups of datasets (LLTS and Eclipse). The two curves show the same pattern, implying that the conclusion is more persuasive. Note that the data presented in the figure are calculated by averaging the $KI$ values over six rankers, nine sizes of feature subsets, and across their respective groups of datasets.

## 6. Conclusion

This paper uses feature selection (FS) and data sampling together to preprocess data in the context of software quality classification. Instead of evaluating FS techniques based on classification performance after training data is changed, this paper focuses more on another important property of a FS method – *stability*, in particular, the sensitivity of a FS method when used with a data sampling technique.

We present 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)) and three sampling approaches (random oversampling (RUS), random oversampling (ROS), and synthetic minority oversampling (SMO)), each combined with two different proportion ratios (35:65 and 50:50), where data sampling is performed prior to FS and a training dataset is formed from the sampled data that consists of the selected features. The consistency index, $I_C$, is used to evaluate the stability of a FS technique with respect to various data sampling approaches, where $I_C$ is calculated between the two feature subsets, one obtained by using a feature ranking technique on the original data and the other obtained by using the same ranking technique but on the sampled data. The experiments were performed on two groups of datasets from two real-world software projects. The results demonstrate that 1) RF and RFW performed better than other rankers in terms of their stabilities; 2) RUS35 and SMO35 produced higher stability values than other sampling approaches; and (3) the post-sampling proportion ratio between *fp* and *nfp* of 35:65 showed better stability than the other ratio of 50:50.

Future work will include experiments using datasets from different software projects as well as other domains. In addition, different data sampling and FS techniques will be considered in the future. Moreover, stability and classification performance will be studied together for feature selection in future works.

## References

[1] T. Abeel, T. Helleputte, Y. Van de Peer, P. Dupont, and Y. Saeys. Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. *Bioinformatics*, 26(3):392–398, February 2010.

[2] G. Boetticher, T. Menzies, and T. Ostrand. Promise repository of empirical software engineering data, 2007.

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

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

[5] C. Drummond and R. C. Holte. C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on Learning from Imbalanced Data Sets II, International Conference on Machine Learning*, 2003.

[6] K. Dunne, P. Cunningham, and F. Azuaje. Solutions to Instability Problems with Sequential Wrapper-Based Approaches To Feature Selection. Technical Report TCD-CD-2002-28, Department of Computer Science, Trinity College, Dublin, Ireland, 2002.

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

[8] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.

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

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

[11] 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.

[12] A. Kalousis, J. Prados, and M. Hilario. Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge and Information Systems*, 12(1):95–116, 2007.

[13] A. H. Kamal, X. Zhu, A. S. Pandya, S. Hsu, and M. Shoaib. 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*, pages 259–269, New Orleans, LA, 2009.

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

[15] L. I. Kuncheva. A stability index for feature selection. In *Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications*, pages 390–395, Anaheim, CA, USA, 2007. ACTA Press.

[16] P. Křížek, J. Kittler, and V. Hlaváč. Improving stability of feature selection methods. In *Proceedings of the 12th international conference on Computer analysis of images and patterns*, CAIP'07, pages 929–936, Berlin, Heidelberg, 2007. Springer-Verlag.

[17] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4):485–496, July-August 2008.

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

[19] S. Loscalzo, L. Yu, and C. Ding. Consensus group stable feature selection. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 567–576, 2009.

[20] J. L. Lustgarten, V. Gopalakrishnan, and S. Visweswaran. Measuring stability of feature selection in biomedical datasets. In *AMIA Annu Symp Proc. 2009*, pages 406–410, 2009.

[21] Y. Saeys, T. Abeel, and Y. Peer. Robust feature selection using ensemble feature selection techniques. In *ECML PKDD '08: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, pages 313–325, Berlin, Heidelberg, 2008. Springer-Verlag.

[22] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. Rusboost: A hybrid approach to alleviate class imbalance. *IEEE Transactions on Systems, Man & Cybernetics: Part A: Systems and Humans*, 40(1), January 2010.

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

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

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