December, 2011

Stability and Classification Performance of Feature Selection Techniques.

Huanjing Wang, *Western Kentucky University*
Taghi M. Khoshgoftaar, *Florida Atlantic University*
Qianhui Althea Liang

Stability and Classification Performance of Feature Selection Techniques

Huanjing Wang
huanjing.wang@wku.edu

Taghi M. Khoshgoftaaar
khoshgof@fau.edu

Qianhui (Althea) Liang
althea.liang@gmail.com

Abstract

Feature selection techniques can be evaluated based on either model performance or the stability (robustness) of the technique. The ideal situation is to choose a feature selection technique that is robust to change, while also ensuring that models built with the selected features perform well. One domain where feature selection is especially important is software defect prediction, where large numbers of metrics collected from previous software projects are used to help engineers focus their efforts on the most faulty modules. This study presents a comprehensive empirical examination of seven filter-based feature ranking techniques (rankers) applied to nine real-world software measurement datasets of different sizes. Experimental results demonstrate that signal-to-noise ranker performed moderately in terms of robustness and was the best ranker in terms of model performance. The study also shows that although Relief was the most stable feature selection technique, it performed significantly worse than other rankers in terms of model performance.

Keywords*: feature ranking, stability, classification.*

1 Introduction

Software metrics data collected during the software development process contain valuable information about a software project. Software defect prediction models are commonly built to detect faulty software modules based on software measurement data (software metrics). Previous studies have shown that the performance of these models improves when irrelevant and redundant metrics (features) are eliminated from the original dataset [15, 16]. During the past decade, numerous studies have examined feature selection with respect to classification performance, but very few studies focus on the robustness (or stability) of feature selection techniques. The purpose of studying the stability of feature selection techniques is to determine which techniques provides feature subsets that are robust to changes in the data (addition or deletion of instances to the dataset). These robust feature selection techniques would choose feature subsets that can be used even after changes to the number of instances in the dataset.

The primary focus of this paper is to evaluate seven filter-based rankers (feature selection techniques), including chi-square (CS), information gain (IG), gain ratio (GR), two forms of the ReliefF algorithm (RF and RFW), symmetrical uncertainty (SU), and signal-to-noise (S2N) in terms of stability (robustness) and classification performance using datasets from a real-world software project, Eclipse [19]. The software defect prediction models were built using Logistic Regression learner.

The key contributions of this research are that:

- We consider the stability of feature selection techniques by comparing the selected features before and after some instances are deleted from a dataset (or equivalently, before and after some instances are added), rather than directly comparing separate subsamples of the original dataset. This is an important distinction because in many real-world situations, software practitioners want to know whether adding additional instances to their dataset will change the results of feature selection.

- We investigate the stability and model performance of feature selection techniques together on real-world software metrics data.

The remainder of the paper is organized as follows. Section 2 provides an overview of related work, while Section 3 presents the seven filter-based feature ranking techniques (rankers). Section 4 describes the datasets and experimental results for stability and classification performance. Finally, we conclude the paper and provide suggestions for future work in Section 5.

2 Related Work

The main goal of feature selection is to select a subset of features that minimizes the prediction errors of classifiers. Guyon and Elisseeff [4] outlined key approaches used for attribute selection, including feature construction, feature ranking, multivariate feature selection, efficient search

methods, and feature validity assessment methods. Hall and Holmes [5] investigated six attribute selection techniques that produce ranked lists of attributes and applied them to several datasets from the UCI machine learning repository. Liu and Yu [11] provided a comprehensive survey of feature selection algorithms and presented an integrated approach to intelligent feature selection.

Feature selection has been applied in many data mining and machine learning applications. However, its application in the software quality and reliability engineering domain is limited. Chen et al. [2] have studied the applications of wrapper-based feature selection in the context of software cost/effort estimation. They concluded that the reduced dataset improved the estimation. Rodríguez et al. [14] applied attribute selection with three filter models and two wrapper models to five software engineering datasets using the WEKA [17] tool. Both techniques are feature subset selection and not ranking techniques. It was stated that the wrapper model was better than the filter model; however, that came at a very high computational cost.

The stability of a feature selection method is normally defined as the degree of agreement between its outputs when applied to randomly-selected subsets of the same input data [9, 12]. To assess robustness of feature selection techniques, past works have used different similarity measures, such as Hamming distance [3], correlation coefficient [7], consistency index [9], and entropy [10]. Among these four similarity measures, consistency index is the only one which takes into consideration bias due to chance. Because of this, in our work the consistency index was used as stability measure. The term consistency index was defined by Kuncheva et al [9]. The consistency index is a measure of similarity between two different feature subsets. He devised this measure as a way to choose the best set of features for an experiment.

3 Filter-based Feature Ranking Techniques

Filter-based feature ranking techniques (rankers) rank features independently without involving any learning algorithm, and then the best features are selected from the ranked list. Researchers have developed a large number of methods to rank features. In this study, we use seven filter-based feature ranking methods, including chi-square, information gain, gain ratio, two types of ReliefF, symmetrical uncertainty, and signal-to-noise.

The chi-square (CS) test [13] 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. 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, which is based on information theory. Information gain (IG) [17] is the information provided about the target class attribute Y, given the value of independent attribute X. Information gain 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; that is, 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) [5] is another way to overcome the problem of IG's bias toward attributes with more values, doing so by dividing IG by the sum of the entropies of X and Y.

Relief is an instance-based feature ranking technique introduced by Kira and Rendell [8]. ReliefF is an extension of the Relief algorithm that can handle noise and multiclass datasets, and is implemented in the WEKA tool [17]. 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.

Signal-to-noise is a measure used in electrical engineering to quantify how much a signal has been corrupted by noise. It is defined as the ratio of signal's power to the noise's power corrupting the signal. Signal-to-noise (S2N) can also be used as feature ranking method [18]. For a binary class problem (such as fp vs. nfp), the S2N is defined as the ratio of the difference of class means ($\mu_{fp} - \mu_{nfp}$) to the sum of the standard deviations of each class ($\sigma_{fp} + \sigma_{nfp}$). If an attribute's expression in one class is quite different from its expression in the other, and there is little variation within the two classes, then the attribute is predictive. Therefore S2N favors attributes where the range of the expression vector is large, but where most of that variation is due to the class distribution. S2N was implemented by our research group in the framework of WEKA tool.

4 Experiments

To test the stability and model performance of different feature selection techniques under different circumstances, we performed a case study on nine different software metric datasets, using seven filter-based feature selection techniques, four different levels of dataset perturbation, and nine different numbers of features chosen. Discussion and results from this case study are presented below.

Table 1. Software Datasets Characteristics

Project	Data	#Metrics	#Modules	%fp	%nfp
Eclipse 2.0	E2.0-10	209	377	6.1%	93.9%
	E2.0-5	209	377	13.79%	86.21%
	E2.0-3	209	377	26.79%	73.21%
Eclipse 3.0	E2.1-5	209	434	7.83%	92.17%
	E2.1-4	209	434	11.52%	88.48%
	E2.1-2	209	434	28.8%	71.2%
Eclipse 3.1	E3.0-10	209	661	6.2%	93.8%
	E3.0-5	209	661	14.83%	85.17%
	E3.0-3	209	661	23.75%	76.25%

Table 2. Average Stability Across All Datasets with 2/3rds of the Datasets

	Number of Features Selected								
Ranker	2	3	4	5	6	7	8	9	10
CS	0.5669	0.5230	0.5273	0.5294	0.5575	0.5687	0.5865	0.6021	0.6183
GR	0.3001	0.3358	0.3561	0.3539	0.3673	0.3778	0.3848	0.4043	0.4157
IG	0.5743	0.5582	0.5555	0.5431	0.5544	0.5637	0.5724	0.5863	0.6051
RF	**0.7905**	**0.7661**	**0.7342**	**0.7275**	0.7138	0.7034	0.7008	0.7094	0.7043
RFW	0.7324	0.6935	0.7014	0.7160	**0.7359**	**0.7212**	**0.7212**	**0.7155**	**0.7191**
SU	0.4059	0.4138	0.4217	0.4313	0.4450	0.4518	0.4769	0.4849	0.5024
S2N	0.6944	0.6571	0.6391	0.6350	0.6237	0.6250	0.6277	0.6381	0.6429

4.1 Datasets

The software metrics and fault data for this case study were collected from a real-world software project, the Eclipse project [19]. From the PROMISE data repository [19], we obtained the Eclipse defect counts and complexity metrics dataset. In particular, we use the metrics and defects data at the software package level. The original data for Eclipse packages consists of three releases denoted 2.0, 2.1, and 3.0 respectively. We transform the original data by: (1) removing all nonnumeric attributes, including the package names, and (2) converting the post-release defects attribute to a binary class attribute: fault-prone (*fp*) and not fault-prone (*nfp*). Membership in each class is determined by a post-release defects threshold *t*, which separates *fp* from *nfp* packages by classifying packages with *t* or more post-release defects as *fp* and the remaining as *nfp*. In our study, we use $t \in \{10, 5, 3\}$ for release 2.0 and 3.0 and $t \in \{5, 4, 2\}$ for release 2.1. All nine derived datasets contain 209 attributes. Releases 2.0, 2.1, and 3.0 contain 377, 434 and 661 instances respectively. Table 1 lists the characteristics of the nine datasets utilized in this work, which exhibit different distributions of class skew (i.e, the percentage of *fp* module).

4.2 Stability of Feature Selection

In this study, we consider stability based on changes to the datasets (perturbations) at the instance level. we chose a fraction c of instances to keep and randomly removed $1 - c$ of the instances from both the majority and minority classes separately, where c is greater than 0 and less than 1. We removed from each class instead of just from the dataset as a whole in order to maintain the original level of class balance/imbalance for each dataset. For each c this process was repeated thirty times giving us thirty new datasets for each original dataset (m instances) and level of c, each having $c \times m$ instances, where each of these new datasets is unique (since each was built by randomly removing $(1 - c) \times m$ instances from the original dataset). In this study, c was set to 0.95, 0.9, 0.8, or 2/3. In total, 9 datasets $\times 4$ levels of perturbation $\times 30$ repetitions $= 1080$ datasets are generated.

In order to measure stability, we decided to use consistency index [9] because it takes into consideration bias due to chance. We compute the consistency index between two feature subsets as follows. Let T_i and T_j be subsets of features, where $|T_i| = |T_j| = k$. The consistency index [9] is obtained as follows:

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

where n is the total number of features in the dataset, d is the cardinality of the intersection between subsets T_i and T_j, and $-1 < I_C(T_i, T_j) \leq +1$. The greater the consistency index, the more similar the subsets are. When we apply consistency index to the original dataset and one of the derivative datasets we can use the resultant consistency measurement as a measurement of stability for the feature ranker.

For each dataset and feature ranking technique, the features are ranked according to their relevance to the class, and then a subset consisting of the most relevant ones (top k features) is selected. In this study, nine subsets are chosen for each dataset. The number of features that is retained in each subset for each dataset are 2, 3, 4, 5, 6, 7, 8, 9, and 10. These numbers were deemed reasonable after some preliminary experimentation conducted on the corresponding datasets [16]. Thus, given a dataset and a feature ranking technique, 1080 I_C values are obtained, since each of the four choices of c and nine choices of feature subset size has 30 stability values (30 repetitions).

Tables 2 through 5 summarize the results of robustness analysis of each ranker, with each table holding perturbation level constant and showing the results for all feature rankers and number of selected features separately. All datasets are averaged together. The top value at each column is in **boldface**. In general, it can be observed that GR shows the least stability and RF and RFW show the most stability.

Figure 1 shows the effect of the degree of dataset perturbation on the stability of feature ranking techniques across all feature selection techniques, the nine datasets, and nine feature subsets. The figure demonstrates that the more instances retained in a dataset (e.g., the fewer instances deleted from the original dataset), the more stable the feature ranking on that dataset will be.

Table 3. Average Stability Across All Datasets with 80% of the Datasets

	Number of Features Selected								
Ranker	2	3	4	5	6	7	8	9	10
CS	0.6323	0.6199	0.6017	0.6039	0.6394	0.6610	0.6723	0.6791	0.6909
GR	0.4305	0.4512	0.4560	0.4535	0.4587	0.4639	0.4830	0.5085	0.5238
IG	0.6632	0.6379	0.6528	0.6352	0.6443	0.6483	0.6497	0.6642	0.6777
RF	**0.8520**	**0.8466**	**0.8282**	**0.8179**	0.7991	0.7844	0.7839	**0.7971**	0.7907
RFW	0.8184	0.7925	0.7937	0.8045	**0.8176**	**0.7986**	**0.7980**	0.7926	**0.7952**
SU	0.4875	0.5078	0.5186	0.5152	0.5266	0.5388	0.5640	0.5700	0.5842
S2N	0.7819	0.7628	0.7314	0.7269	0.7127	0.7152	0.7214	0.7372	0.7497

Table 4. Average Stability Across All Datasets with 90% of the Datasets

	Number of Features Selected								
Ranker	2	3	4	5	6	7	8	9	10
CS	0.7221	0.7070	0.7051	0.6973	0.7259	0.7417	0.7503	0.7612	0.7727
GR	0.5699	0.5689	0.5813	0.5788	0.5857	0.5927	0.6120	0.6335	0.6461
IG	0.7347	0.7143	0.7294	0.7174	0.7224	0.7280	0.7331	0.7469	0.7622
RF	**0.8977**	**0.9125**	**0.9025**	**0.8941**	0.8745	0.8668	0.8631	**0.8688**	0.8627
RFW	0.8768	0.8728	0.8903	0.8826	**0.8852**	**0.8716**	**0.8768**	0.8679	**0.8676**
SU	0.5965	0.6209	0.6269	0.6427	0.6430	0.6409	0.6672	0.6766	0.6898
S2N	0.8636	0.8444	0.8341	0.8324	0.8112	0.8143	0.8248	0.8323	0.8471

Table 5. Average Stability Across All Datasets with 95% of the Datasets

	Number of Features Selected								
Ranker	2	3	4	5	6	7	8	9	10
CS	0.7681	0.7715	0.7870	0.7729	0.7949	0.8127	0.8162	0.8285	0.8269
GR	0.6779	0.6758	0.6976	0.6922	0.6872	0.6937	0.7120	0.7344	0.7431
IG	0.8082	0.7895	0.7840	0.7816	0.7821	0.7878	0.7952	0.8056	0.8165
RF	**0.9285**	**0.9427**	**0.9400**	**0.9330**	0.9107	0.9039	0.9025	**0.9094**	0.9054
RFW	0.9128	0.9207	0.9269	0.9275	**0.9258**	**0.9080**	**0.9185**	0.9045	**0.9067**
SU	0.6691	0.7142	0.7200	0.7346	0.7263	0.7306	0.7485	0.7569	0.7641
S2N	0.9084	0.8826	0.8867	0.8714	0.8528	0.8626	0.8691	0.8812	0.8889

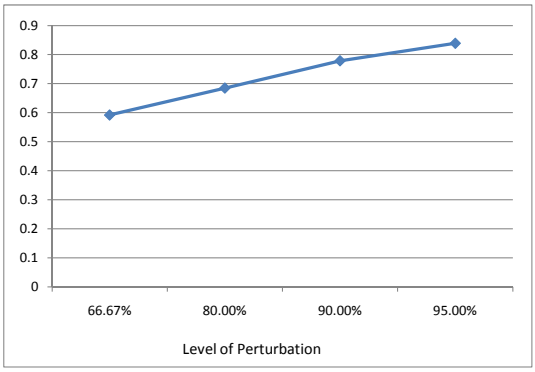

Figure 1. Degree of Perturbation Impact on Stability

Table 6. Analysis of Variance, Stability

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	290.45	6	48.409	995.86	0
Error	826.52	17003	0.0486		
Total	1116.97	17009			

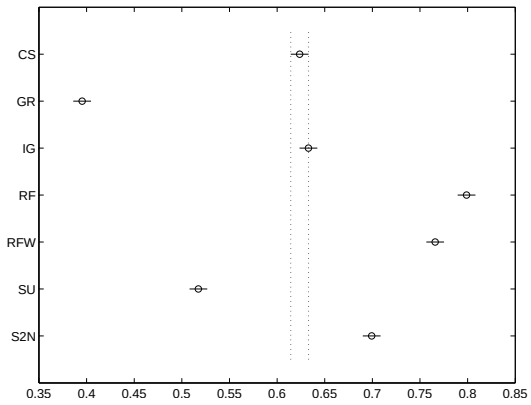

Figure 2. Multiple Comparisons, Stability

We performed an ANalysis Of VAriance (ANOVA) [1] to statistically examine the robustness (e.g., stability) of feature selection techniques. An n-way ANOVA can be used to determine if the means in a set of data differ when grouped by multiple factors. If they do differ, one can determine which factors or combinations of factors are associated with the difference. The factor A of our one-way ANOVA model includes the seven rankers. For the ANOVA test, the perturbation level was held constant at 80%, and the results from all nine datasets were taken into account together. The ANOVA results are presented in Table 6. The p value of Factor A is 0, which indicates there was a significant difference between the average I_C values of the seven rankers. Additional multiple comparisons for the main factor was performed to investigate the differences among the respective groups (levels). All tests of statistical significance utilize a significance level α of 5%. Both ANOVA and multiple comparison tests were implemented in MATLAB.

The multiple comparison results are presented in Figure 2, displaying graphs 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. It can be observed that RF comes out best, with RFW, S2N, IG, CS, SU, and GR following in that order.

Table 7. Classification Performance

	E2.0-10	E2.0-5	E2.0-3	E2.1-5	E2.1-4	E2.1-2	E3.0-10	E3.0-5	E3.0-3
CS	0.8369	0.9094	**0.8703**	0.8884	0.8868	0.8863	0.9165	**0.9437**	0.9117
GR	0.8325	0.8247	0.7593	0.8324	0.8413	0.8813	0.8917	0.9391	0.9062
IG	0.8553	**0.9104**	0.8657	0.9105	0.8998	**0.8876**	**0.9269**	0.9434	0.9122
RF	0.8589	0.8827	0.7997	0.7739	0.7178	0.7222	0.8779	0.8433	0.8037
RFW	0.8435	0.8829	0.8027	0.8103	0.6864	0.8751	0.7918	0.8373	0.8066
SU	0.8314	0.9002	0.8600	0.8906	0.8840	0.8843	0.9006	0.9434	0.9103
S2N	**0.8814**	0.9014	0.8487	**0.9220**	**0.9002**	0.8844	0.9219	0.9380	**0.9227**

Table 8. Analysis of Variance, Classification

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	0.88664	6	0.14777	70.53	0
Error	1.3052	623	0.0021		
Total	2.19184	629			

4.3 Classification Performance

To evaluate a feature selection technique, stability of feature selection may not be enough to find a good feature selection technique. We should also consider model performance. Software defect prediction models have been used to improve the fault prediction and risk assessment process. Finding faulty components in a software system can lead to a more reliable final system and reduce development and maintenance costs. Classifiers were built with a well-known classification algorithm, Logistic Regression. Logistic Regression (LR) [17] is a statistical regression model for categorical prediction which operates by fitting data to a logistic curve. Based on the training data, a logistic regression model is created which is used to decide the class membership of future instances. For our experiments, the default settings of WEKA [17] are used. The classification models are evaluated using the area under the ROC (Receiver Operating Characteristic) curve (AUC) performance metric. AUC is widely used, providing a general idea of the predictive potential of the classifier. A higher AUC is better, as it indicates that the classifier, across the entire possible range of decision threshold, has a higher true positive rate. It has been shown that AUC has lower variance and is more reliable than other performance metrics (such as precision, recall, or F-measure) [6].

During the experiments, ten runs of five-fold cross-validation were performed. For each of the five folds, one fold (20% of instances) is used as test data while the other four (80% of instances) are used as the training data. First, we ranked the attributes using the seven rankers separately. Once the attributes are ranked, the top four attributes are selected (as well as the class attribute) [16] to yield final training data. After feature selection, we applied the Logistic Regression classifier to the training datasets with the selected features, and then we used AUC to evaluate the performance of the classification model. In total, 7 rankers $\times$ 9 datasets $\times$ 10 runs $\times$ 5 folds = 3150 combinations of feature ranking techniques were employed, and corresponding models were built.

Experimental results are reported in Table 7. Each value presented in the table is the average over the ten runs of five-fold cross-validation outcomes. The best model for each dataset is highlighted with **boldfaced** print. The results demonstrate that among the seven rankers, S2N outperformed the others for 4 out of 9 cases.

We also carried out a one-way ANOVA test [1] on the AUC performance metric. The factor A represents seven rankers. In this ANOVA test, the results from all nine datasets were taken into account together. A significance level of $\alpha = 5\%$ was used for all statistical tests. The ANOVA results are presented in Table 8. The test results indicate the classification performances of seven rankers (Factor A) were significantly different from each other (p = 0). The multiple comparison results are presented in Figure 3. The figure shows the following facts: for the classification models built with Eclipse, we can divide the seven rankers into three groups, $\{$RF, RFW$\}$, $\{$GR$\}$, and $\{$SU, CS, IG, S2N$\}$, ordered by their performances from worst to best. The rankers from different groups performed significantly different, while the rankers from same group performed similarly (or insignificantly different). For the last group, S2N outperformed IG, CS, and SU.

4.4 Stability vs. Classification

From Figures 2 and 3, we can observe that RF is the most stable ranker, however it is the worst ranker when classification model was built using the features selected by the ranker. One of the reasons for this is that the software metrics datasets we used in the study are imbalanced. Experimental results also demonstrate that the signal-to-noise (S2N) ranker performed moderately in terms of robustness and was the best ranker in terms of model performance. Therefore, S2N is recommended in this study.

5 Conclusion

Feature (software metric) selection plays an important role in the software engineering domain. This empirical study investigates the stability (robustness) and classification performance of seven filter-based feature ranking techniques on nine software metrics datasets.

For the evaluation of the feature ranking techniques, the consistency index proposed by Kuncheva [9] is used. Results also show that the number of instances deleted from the dataset affects the stability of the feature ranking techniques. The fewer instances removed from (or equivalently, added to) a given dataset, the less the selected features will change when compared to the original dataset, and thus the

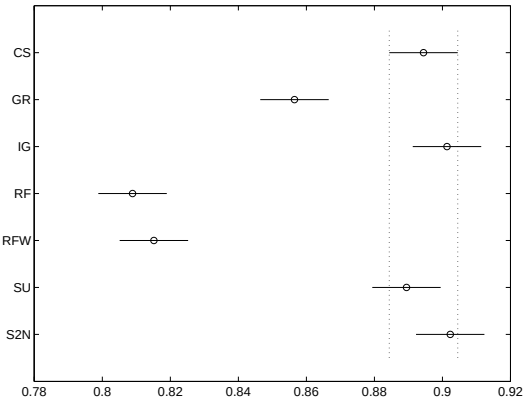

Figure 3. Multiple Comparisons, Classification

feature ranking performed on this dataset will be more stable. We also built classification models using Logistic Regression learner. The classification accuracy is evaluated in terms of the AUC performance metric. The experimental results demonstrate that the signal-to-noise ranker performed moderately in terms of robustness and was the best ranker in terms of model performance. The empirical study also shows that Relief was the most stable feature selection technique, even though performed significantly worse than other rankers in terms of model performance.

Future work will involve conducting additional empirical studies with data from other software projects and application domains, and experiments with other ranking techniques and classifiers for building classification models.

References

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

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

[3] 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.

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

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

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

[7] 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, Dec. 2006.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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, New York, NY, USA, 2009.

[13] R. L. Plackett. Karl pearson and the chi-squared test. *International Statistical Review*, 51(1):5972, 1983.

[14] 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*, pages 667–672, Las Vegas, Nevada, August 13-15 2007.

[15] J. Van Hulse, T. M. Khoshgoftaar, A. Napolitano, and R. Wald. Feature selection with high dimentional imbalanced data. In *Proceedings of the 9th IEEE International Conference on Data Mining - Workshops (ICDM'09)*, pages 507–514, Miami, FL, December 2009. IEEE Computer Society.

[16] H. Wang, T. M. Khoshgoftaar, and N. Seliya. How many software metrics should be selected for defect prediction? In *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*, pages 69–74, May 2011.

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

[18] C.-H. Yang, C.-C. Huang, K.-C. Wu, and H.-Y. Chang. A novel ga-taguchi-based feature selection method. In *IDEAL '08: Proceedings of the 9th International Conference on Intelligent Data Engineering and Automated Learning*, pages 112–119, Berlin, Heidelberg, 2008.

[19] T. Zimmermann, R. Premraj, and A. Zeller. Predicting defects for eclipse. In *ICSEW '07: Proceedings of the 29th International Conference on Software Engineering Workshops*, page 76, Washington, DC, USA, 2007. IEEE Computer Society.