11-2011

Measuring Stability of Threshold-based Feature Selection Techniques

Huanjing Wang

Taghi Khoshgoftaar

November, 2011

Measuring Stability of Threshold-based Feature Selection Techniques

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

Measuring Stability of Threshold-based Feature Selection Techniques

Huanjing Wang
Western Kentucky University
Bowling Green, Kentucky 42101
huanjing.wang@wku.edu

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

Abstract

Feature selection has been applied in many domains, such as text mining and software engineering. Ideally a feature selection technique should produce consistent outputs regardless of minor variations in the input data. Researchers have recently begun to examine the stability (robustness) of feature selection techniques. The stability of a feature selection method is defined as the degree of agreement between its outputs to randomly-selected subsets of the same input data. This study evaluated the stability of 11 threshold-based feature ranking techniques (rankers) when applied to 16 real-world software measurement datasets of different sizes. Experimental results demonstrate that AUC (Area Under the Receiver Operating Characteristic Curve) and PRC (Area Under the Precision-Recall Curve) performed best among the 11 rankers.

Keywords*: threshold-based feature ranking, stability, robustness, software metrics.*

1 Introduction

Feature selection is an important data mining preprocessing step used to find a subset of features, which will then be used in further study, such as for defect prediction models. Feature selection techniques can be broadly classified as *feature ranking* and *feature subset selection*. Feature ranking sorts the attributes according to their individual predictive power, while feature subset selection finds subsets of attributes that collectively have good predictive power. Feature selection techniques can also be categorized as *filters*, *wrappers*, or *embedded* methods. Filters are algorithms in which a feature subset is selected without involving any learning algorithm. Wrappers are algorithms that use feedback from a learning algorithm to determine which feature(s) to include in building a classification model. Embedded methods do not perform explicit feature selection like filters and wrappers; instead, feature selection is incorporated within a learning algorithm.

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 stability of a feature selection method is defined as the degree of agreement between its outputs when applied to randomly-selected subsets of the same input data [15, 18]. In this paper, we present our newly proposed threshold-based feature selection techniques (TBFS) and assess the stability performance of these 11 rankers. This assessment is based on the degree of agreement between a filter's outputs on both the original datasets and on modified datasets which have had some instances removed. The experiment was implemented through a case study of four consecutive releases of a very large telecommunications software system (denoted as LLTS), three datasets from NASA project KC1, and nine datasets from the Eclipse project. The experimental results showed that AUC and PRC performed best among the 11 rankers and OR (Odds Ratio), PR (Probability Ratio), and GI (Gini Index) performed worst.

The main contribution of the present work is 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. The experiments discussed in this study contain the answer. To our knowledge, no previous comprehensive empirical investigation has been performed comparing the stability performance of 11 threshold-based feature selection techniques in the domain of software reliability engineering and perhaps other application domains.

The remainder of the paper is organized as follows. Section 2 provides an overview of related work, while Section 3 presents the 11 threshold-based rankers and our stability metric. Section 4 describes the datasets, experimental design, and experimental results. Finally, we conclude the pa-

per in Section 5 and provide suggestions for future work.

2 Related Work

Feature selection has been applied in many data mining and machine learning applications. The main goal of feature selection is to select a subset of features that minimizes the prediction errors of classifiers. Guyon and Elisseeff [9] 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 [10] investigated six attribute selection techniques that produce ranked lists of attributes and applied them to several datasets from the UCI machine learning repository. Forman [7] investigated multiple filter-based feature ranking techniques. Liu and Yu [17] provided a comprehensive survey of feature selection algorithms and presented an integrated approach to intelligent feature selection.

Although feature selection has been widely applied in numerous application domains for many years, its application in the software quality and reliability engineering domain is limited. Chen et al. [4] have studied the applications of wrapper-based feature selection in the context of software cost/effort estimation. They concluded that the reduced data set improved the estimation. Rodríguez et al. [20] applied attribute selection with three filter models and two wrapper models to five software engineering data sets using the WEKA [23] 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.

Although previous work has focused on the performance of models built using the selected features, another way to evaluate a feature selection technique is robustness (stability), which has received less attention in the past. Few studies exist on the stability of feature selection algorithms. 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 [15, 18]. Recent work in this area mainly focuses on consistency of the outputs by measuring the variations between subsets of features obtained from different subsamples of the original training dataset. Saeys et al. [21] assessed the robustness of feature selection techniques using the Spearman rank correlation coefficient and Jaccard index. Dunne et al. [5] addressed the instability of the wrapper approach to feature selection. They suggested a measure based on the Hamming distance to assess the stability of a feature selection technique. Loscalzo et al. [18] demonstrated a strong dependency between the sample size (in terms of number of instances in a dataset) and the stability of a feature selection method. Abeel et al. [1] studied the process for selecting biomarkers from microarray data and presented a general framework for stability analysis of such feature selection techniques. They showed that stability could be improved through ensemble feature selection, where the training data is bootstrapped, recursive feature elimination (RFE) is applied to each subset, and either a complete linear or complete weighted linear aggregation method is used to get a consensus output.

3 Methodology

3.1 Threshold-based Feature Selection Techniques

In this study, we focus on filter-based feature ranking techniques and applied these feature ranking techniques to software engineering datasets. Filter-based feature ranking techniques rank features independently without involving any learning algorithm. Eleven threshold-based feature selection techniques (TBFS) were recently proposed and implemented by our research group within WEKA [23]. The procedure is shown in Algorithm 1. First, each attribute's values are normalized between 0 and 1 by mapping F^j to $\hat{F}^j$. The normalized values are treated as posterior probabilities. Each independent attribute is then paired individually with the class attribute and the reduced two attribute dataset is evaluated using 11 different performance metrics based on this set of "posterior probabilities." In standard binary classification, the predicted class is assigned using the default decision threshold of 0.5. The default decision threshold is often not optimal, especially when the class is imbalanced. Therefore, we propose the use of performance metrics that can be calculated at various points in the distribution of $\hat{F}^j$. At each threshold position, we classify values above the threshold as positive, and below as negative. Then we go in the opposite direction, and consider values above as negative, and below as positive. Whatever direction produces the more optimal performance metric values is used. The true positive (TPR), true negative (TNR), false positive (FPR), and false negative (FNR) rates can be calculated at each threshold $t \in [0, 1]$ relative to the normalized attribute $\hat{F}^j$. The threshold-based attribute ranking techniques we propose utilize these rates as described below.

- *F-measure (FM)*: is a single value metric derived from the F-measure that originated from the field of information retrieval [23]. The maximum F-measure is obtained when varying the decision threshold value between 0 and 1.

- *Odds Ratio (OR)*: is the ratio of the product of correct (TPR times TNR) to incorrect (FPR times FNR)

Algorithm 1: Threshold-based Feature Selection Algorithm

input :
1. Dataset D with features $F^j, j = 1, \ldots, m$;
2. Each instance $x \in D$ is assigned to one of two classes $c(x) \in \{fp, nfp\}$;
3. The value of attribute F^j for instance x is denoted $F^j(x)$;
4. Metric $\omega \in$ {FM, OR, PO, PR, GI, MI, KS, DV, GM, AUC, PRC};
5. A predefined threshold: number (or percentage) of the features to be selected.

output:
Selected feature subsets.

for $F^j, j = 1, \ldots, m$ **do**
  Normalize $F^j \mapsto \hat{F}^j = \frac{F^j - \min(F^j)}{\max(F^j) - \min(F^j)}$;
  Calculate metric ω using attribute $\hat{F}^j$ and class attribute at various decision threshold in the distribution of $\hat{F}^j$. The optimal ω is used, $\omega(\hat{F}^j)$.

Create feature ranking $\mathbb{R}$ using $\omega(\hat{F}^j)\ \forall j$.
Select features according to feature ranking $\mathbb{R}$ and a predefined threshold.

predictions. The maximum value is taken when varying the decision threshold value between 0 and 1.

- *Power (PO)*: is a measure that avoids common false positive cases while giving stronger preference for positive cases [7]. Power is defined as:

$$\text{PO} = \max_{t \in [0,1]} \left((TNR(t))^k - (FNR(t))^k \right)$$

  where $k = 5$.

- *Probability Ratio (PR)*: is the sample estimate probability of the feature given the positive class divided by the sample estimate probability of the feature given the negative class [7]. PR is the maximum value of the ratio when varying the decision threshold value between 0 and 1.

- *Gini Index (GI)*: measures the impurity of a dataset [3]. GI for the attribute is then the minimum Gini index at all decision thresholds $t \in$[0, 1].

- *Mutual Information (MI)*: measures the mutual dependence of the two random variables [19]. High mutual information indicates a large reduction in uncertainty, and zero mutual information between two random variables means the variables are independent.

- *Kolmogorov-Smirnov (KS)*: utilizes the Kolmogorov-Smirnov statistic to measure the maximum difference between the empirical distribution functions of the attribute values of instances in each class [13]. It is effectively the maximum difference between the curves generated by the true positive and false positive rates as the decision threshold changes between 0 and 1.

- *Deviance (DV)*: is the residual sum of squares based on a threshold t. That is, it measures the sum of the squared errors from the mean class given a partitioning of the space based on the threshold t and then the minimum value is chosen.

- *Geometric Mean (GM)*: is a single-value performance measure which is calculated by finding the maximum geometric mean of TPR and TNR as the decision threshold is varied between 0 and 1.

- *Area Under ROC (Receiver Operating Characteristic) Curve* (AUC): has been widely used to measure classification model performance [6]. The ROC curve is used to characterize the trade-off between true positive rate and false positive rate. In this study, ROC curves are generated by varying the decision threshold t used to transform the normalized attribute values into a predicted class.

- *Area Under the Precision-Recall Curve* (PRC): is a single-value measure that originated from the area of information retrieval. The area under the PRC ranges from 0 to 1. The PRC diagram depicts the trade off between recall and precision.

3.2 Stability

To assess the robustness of feature selection techniques, past works have used different similarity measures, such as Hamming distance [5], correlation coefficient [11], consistency index [15], and entropy [16]. 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. Our stability metric is defined as follows: First, we assume the original dataset has m instances and n features. Let T_i and T_j be subsets of features, where $|T_i| = |T_j| = k$. The consistency index [15] is obtained as follows:

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

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.

4 Experiments

To test the stability of different feature selection techniques under different circumstances, we performed a case study on 16 different software metric datasets, using 11 threshold-based feature selection techniques, four different levels of change to the datasets, and nine different numbers of chosen features. Discussion and results from this case study are presented below.

4.1 Datasets

The software metrics and fault data for this case study were collected from real-world software projects, including a very large telecommunications software system (denoted as LLTS) [8], the Eclipse project [24], and NASA software project KC1 [14].

LLTS contains data from four consecutive releases, which are labeled as SP1, SP2, SP3, and SP4. The software measurement datasets consist of 42 software metrics, including 24 product metrics, 14 process metrics, and four execution metrics [8].

From the PROMISE data repository [24], we also 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 λ, which separates *fp* from *nfp* packages by classifying packages with λ or more post-release defects as *fp* and the remaining as *nfp*. In our study, we use $\lambda \in \{10, 5, 3\}$ for release 2.0 and 3.0 while we use $\lambda \in \{5, 4, 2\}$ 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.

The NASA project, KC1 [14], includes 145 instances containing 95 attributes each. After removing 32 Halstead derived measures, we have 63 attributes. We used three different thresholds to define defective instances, thereby obtaining three structures of the preprocessed KC1 dataset. The thresholds are 20, 10, and 5, indicating instances with numbers of defects greater than or equal to 20, 10, or 5 belong to the *fp* class. The three datasets are named KC1-20, KC1-10, and KC1-5.

Table 1 lists the characteristics of the 16 datasets utilized in this work, which exhibit different distributions of class skew (i.e., the percentage of *fp* modules).

Table 1. Software Datasets Characteristics

	Data	#Metrics	#Modules	%fp	%nfp
LLTS	SP1	42	3649	6.28%	93.72%
	SP2	42	3981	4.75%	95.25%
	SP3	42	3541	1.33%	98.67%
	SP4	42	3978	2.31%	97.69%
Eclipse	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%
	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%
	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%
KC1	KC1-20	63	145	6.90%	93.10%
	KC1-10	63	145	14.48%	85.52%
	KC1-5	63	145	24.83%	75.17%

4.2 Experimental Design

For this study, we consider stability based on changes to the datasets (perturbations) at the instance level. Consider a dataset with m instances: a smaller dataset can be generated by randomly removing a fraction c of instances from the original data, where c is greater than 0 and less than 1. For a given c, this process can be performed x times. This will create x new datasets, each having $(1 - c) \times m$ instances, where each of these new datasets is unique (since each was built by randomly removing $c \times m$ instances from the original dataset). In this study, x was set to 30 and c was set to 0.05, 0.1, 0.2, or 1/3 (giving new datasets with $0.95 \times m$, $0.9 \times m$, $0.8 \times m$, or $2/3 \times m$ instances, respectively), thereby obtaining 30 datasets for each original dataset and choice of c. In total, $16 \times 4 \times 30 = 1920$ datasets are generated.

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 are deemed reasonable after some preliminary experimentation conducted on the corresponding datasets [22].

For each original dataset and choice of c (the percentage of instances to remove), let T_0 represent the set containing the top k ranked features obtained by a particular feature ranking technique on that particular original dataset. Then x datasets of same size are generated by deleting data instances from that original dataset. Accordingly, let $\{T_1, T_2, ..., T_x\}$ be the sets of features selected from the datasets generated. A single stability index (KI) is obtained as follows:

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

This is the average of the consistency index (see Section 3.2) for each pairing of the original dataset and one of the x new datasets. Note that although this use is not identical to more traditional KI consistency measures, since the consistency index I_C is still a core component of the measure, we retain the name. Thus, given a dataset and a feature ranking technique, 36 KI values are obtained, since each of the four choices of c and nine choices of feature subset size gives one KI value (and $4 \times 9 = 36$).

4.3 Results and Analysis

Experiments were conducted with 11 threshold-based feature ranking techniques on 16 software engineering metrics datasets. Aggregated and selected results are presented below. To briefly summarize, Figure 1 shows combining all 16 datasets to highlight variations caused by degree of perturbation to the dataset, size of feature subset, and choice of filter. Figure 2 shows degree of perturbation impact on stability. Table 2 shows each dataset separately, but only presents results for one chosen degree of permutation (95%) and feature subsets of size four. Table 3 and Figure 3 show ANOVA results and multiple comparison results for one chosen size (size four) of feature subset. Further analysis of the results across all the parameters was not possible due to space limitations.

4.3.1 Average Stability Performance

Figure 1 shows how the average stability performance of each ranker is affected by the nine different feature subset sizes, averaged across all sixteen datasets and with the four different perturbation levels shown in separate graphs. These graphs show the following observations:

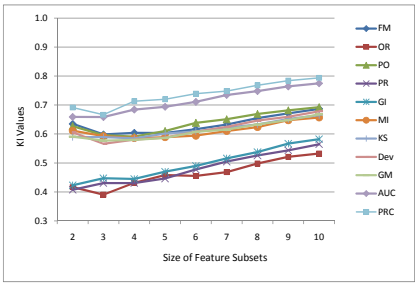
- Among the eleven filters, OR shows the least stability on average, followed by PR and GI, while PRC and AUC show the most stability.
- The size of the feature subset can influence the stability of a feature ranking technique. For most rankers, stability is improved by increasing the number of features in the selected subset.

4.3.2 Degree of Perturbation Impact on Stability

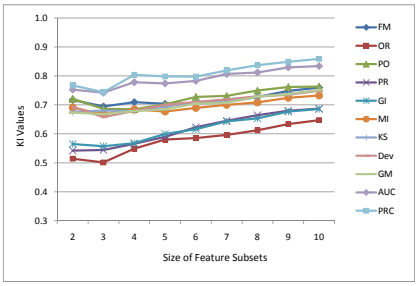
Figure 2 shows the effect of the degree of dataset perturbation on the stability of feature ranking techniques across all sixteen datasets and nine feature subsets. The figure demonstrates that the more instances retained in a dataset (e.g., the fewer instances deleted from the original dataset), the more stable the feature ranking on that dataset will be.

4.3.3 Most and Least Stable Filters

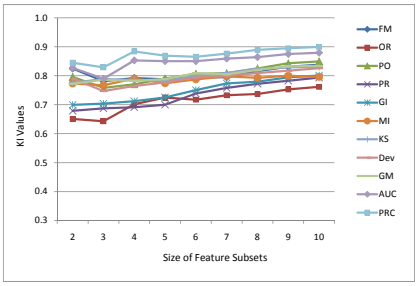
Table 2 summarize the results of robustness analysis for feature subsets of size four and datasets which contain 95% of the instances from their original dataset. In general, it can be observed that PR shows the least stability for 7 out of 16 and AUC and PRC shows the most stability for 4 out of 16 each. The inconsistent performance of KS, MI, OR, and GM appears to stem from their greater sensitivity to the specific choice of dataset. The last column of Table 2 shows average performance across all 16 datasets.



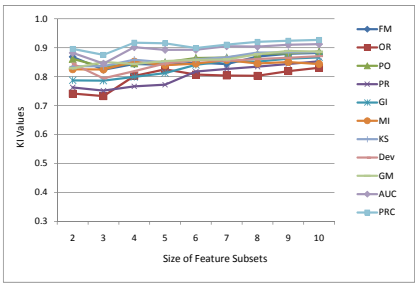
(a) Datasets with 2/3



(b) Datasets with 80%



(c) Datasets with 90%



(d) Datasets with 95%

Figure 1. Average Stability Performance on 16 Datasets

Table 2. Stability Performance for feature subset of Size Four and Datasets with 95% of the instances from their original datasets

	SP1	SP2	SP3	SP4	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	KC1-5	KC1-10	KC1-20	Average
FM	0.908	0.889	0.908	0.825	0.762	0.822	0.915	0.839	0.788	0.907	0.745	0.915	0.788	0.786	0.831	0.875	0.844
OR	0.908	0.797	0.917	0.705	0.779	0.949	0.805	0.771	0.771	0.737	0.779	0.779	0.941	0.510	0.804	0.884	0.802
PO	0.889	0.945	0.770	0.816	0.694	0.805	0.941	0.873	0.830	0.813	0.762	0.932	0.890	0.831	0.857	0.893	0.846
PR	0.936	0.843	0.926	0.650	0.405	0.703	0.737	0.584	0.941	0.711	0.567	0.796	0.839	0.840	0.884	0.902	0.766
GI	0.936	0.843	0.843	0.696	0.558	0.788	0.788	0.703	0.932	0.711	0.669	0.856	0.839	0.840	0.884	0.902	0.799
MI	0.889	0.991	0.963	0.880	0.856	0.949	0.958	0.847	0.873	0.796	0.737	0.915	0.762	0.679	0.706	0.831	0.852
KS	0.963	0.807	1.000	0.724	0.805	0.932	0.839	0.779	0.898	0.924	0.822	0.754	0.754	0.929	0.893	0.920	0.859
Dev	0.834	0.751	0.834	0.788	0.669	0.788	0.983	0.915	0.771	0.924	0.652	0.907	0.771	0.911	0.768	0.840	0.819
GM	0.908	0.797	1.000	0.733	0.788	0.941	0.915	0.703	0.813	0.924	0.873	0.711	0.822	0.822	0.822	0.929	0.844
AUC	0.991	0.991	0.779	0.761	0.856	1.000	1.000	0.915	0.915	0.992	0.898	0.915	0.881	0.822	0.884	0.831	0.902
PRC	0.982	0.926	0.917	0.834	0.847	0.983	1.000	0.941	0.924	0.983	0.966	0.975	0.830	0.742	0.938	0.893	0.917

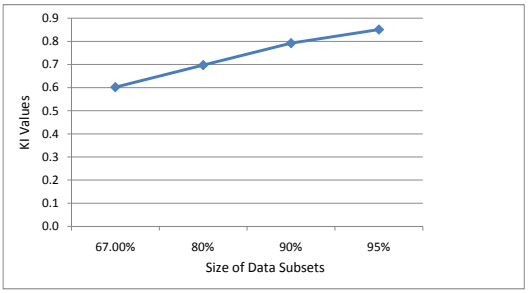


Figure 2. Degree of Perturbation Impact on Stability

4.3.4 ANOVA Analysis

Although our previous study showed that three features (software metrics) are sufficient for building software quality classification models [22, 12], we found there is no significant difference between the models built with three, four, and five features. Thus, to simplify things, in this section we present stability results only for size four feature subsets. A two-way ANalysis Of VAriance (ANOVA) F test [2] was performed for datasets with feature subsets of size four. The two factors are Factor A, in which 11 rankers were considered, and Factor B, in which four different levels of deletion (c values) were included. In this ANOVA test, the results from all sixteen datasets were taken into account together. The ANOVA results are presented in Table 3. The p values of Factor A and Factor B are 0, which indicates there was a significant difference between the average KI values of the 11 rankers and four different levels of deletion (c values). We further conducted multiple pairwise comparison tests on Factor A. The multiple comparison test results are shown in Figure 3. The figure displays graphs with each group mean represented by a symbol (○) 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. Matlab was used to perform the ANOVA and multiple comparisons

Table 3. Analysis of Variance

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	3.0871	10	0.30871	23.69	0
B	7.6609	3	2.55364	195.97	0
Error	8.9911	690	0.01303		
Total	19.7391	703			

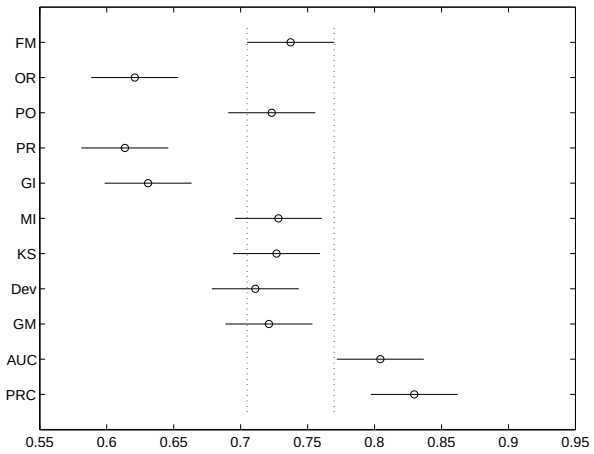
presented in this work, and the assumptions for constructing ANOVA models were validated. The results show the following facts:

- For factor A, we can observe that three distinct patterns emerge when we are ordering the 11 rankers: (1) PRC and AUC performed significantly best among the 11 rankers; (2) PR, OR, and GI performed significantly worst; (3) Dev, GM, PO, KS, MI, and FM performed moderately.
- For Factor B, datasets with 95% instances of original datasets performed best, followed by 90%, 80%, and 2/3.

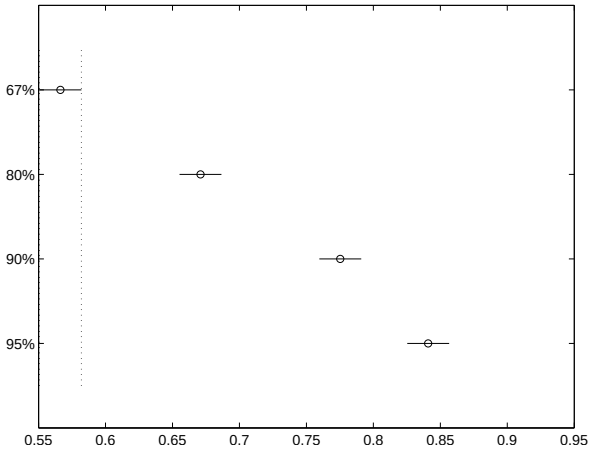
5 Conclusion

This paper examines the stability performance of 11 threshold-based feature selection techniques on 16 real-world software datasets. Experimental results demonstrate that AUC and PRC performed significantly best among the 11 rankers and OR, GI, and PR performed worst. Results also showed 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 feature ranking performed on this dataset will be more stable. Hence, the findings of this study suggest that the choice of feature selection technique, the dataset size, and the size of feature subset should be considered in the instability problem.

Future work will involve conducting additional empirical studies with data from other software projects and applica-



(a) Factor A



(b) Factor B

Figure 3. Multiple Comparisons

tion domains and experiments with other similarity measures for stability analysis.

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] 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.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC Press, Boca Raton, FL, 1984.

[4] Z. Chen, T. Menzies, D. Port, and D. Boehm. Finding the right data for software cost modeling. *Software, IEEE*, 22(6):38–46, Nov.-Dec. 2005.

[5] 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.

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

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

[8] K. Gao, T. M. Khoshgoftaar, and H. Wang. An empirical investigation of filter attribute selection techniques for software quality classification. In *Proceedings of the 10th IEEE International Conference on Information Reuse and Integration*, pages 272–277, Las Vegas, Nevada, August 10-12 2009.

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

[10] 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.

[11] 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, December 2006.

[12] T. M. Khoshgoftaar, K. Gao, and N. Seliya. Attribute selection and imbalanced data: Problems in software defect prediction. In *Proceedings of 22nd IEEE International Conference on Tools with Artificial Intelligence*, pages 137–144, Arras, France, October 27-29 2010.

[13] T. M. Khoshgoftaar and N. Seliya. Fault-prediction modeling for software quality estimation: Comparing commonly used techniques. *Empirical Software Engineering Journal*, 8(3):255–283, 2003.

[14] A. G. Koru, D. Zhang, K. E. Emam, and H. Liu. An investigation into the functional form of the size-defect relationship for software modules. *IEEE Trans. Software Eng.*, 35(2):293–304, 2009.

[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.

[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] 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.

[18] 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.

[19] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[20] 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.

[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] 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.

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

[24] 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.