International Journal on Artificial Intelligence Tools
© World Scientific Publishing Company

A STUDY OF SOFTWARE METRIC SELECTION TECHNIQUES: STABILITY ANALYSIS AND DEFECT PREDICTION MODEL PERFORMANCE

Huanjing Wang
Department of Computer Science
Western Kentucky University
Bowling Green, Kentucky, USA
huanjing.wang@wku.edu

Taghi M. Khoshgoftaar
Department of Computer Science and Engineering
Florida Atlantic University
oca Raton, Florida, USA
khoshgof@fau.edu

Qianhui (Althea) Liang
HP Labs, Singapore
althea.liang@gmail.com

Software metrics (features or attributes) are collected during the software development cycle. Metric selection is one of the most important preprocessing steps in the process of building defect prediction models and may improve the final prediction result. However, the addition or removal of program modules (instances or samples) can alter the subsets chosen by a feature selection technique, rendering the previously-selected feature sets invalid. Very limited research have been done considering both stability (or robustness) and defect prediction model performance together in the software engineering domain, despite the importance of both aspects when choosing a feature selection technique. In this paper, we test the stability and classification model performance of eighteen feature selection techniques as the magnitude of change to the datasets and the size of the selected feature subsets are varied. All experiments were conducted on sixteen datasets from three real-world software projects. The experimental results demonstrate that Gain Ratio shows the least stability while two different versions of ReliefF show the most stability, followed by the PRC- and AUC-based threshold-based feature selection techniques. Results also show that the signal-to-noise ranker performed moderately in terms of robustness and was the best ranker in terms of model performance. Finally, we conclude that while for some rankers, stability and classification performance are correlated, this is not true for other rankers, and therefore performance according to one scheme (stability or model performance) cannot be used to predict performance according to the other.

Keywords: software metric selection, stability, defect prediction, model performance.

1. Introduction

Data preprocessing is a critical step in the software defect prediction modeling process. The software measurement data (metrics) collected during the software development process contain valuable information about a software project's status, process, quality, performance, and evolution. In the past, these data have been used to improve the fault detection process. 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 [1,2].

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 provide feature subsets that are robust to changes in the data (addition or deletion of program modules 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. This is important for software quality practitioners because it allows them to continue using a model even after additional data is collected, without having to perform feature selection again.

In this study we examine the stability of eighteen different feature selection techniques, including six commonly used feature selection techniques, eleven threshold-based feature selection (TBFS) techniques, and a relatively new technique called Signal to Noise. We evaluate the stability of a metric on a dataset by measuring the variance between the subset chosen using the full dataset and that chosen from modified datasets with instances removed. One could also view this information as before and after instances have been added to the dataset. In addition to examining how choice of feature selection technique, size of feature subset, and degree of dataset perturbation can affect the feature subset stability, we also evaluate the effectiveness of defect predictors that estimate the quality of program modules, e.g., fault-prone (*fp*) or not-fault-prone (*nfp*). After all, a feature selection technique must be both stable and accurate (providing features useful for classification) in order for it to be a good technique. In this study, we built defect prediction models using naïve Bayes (NB), support vector machines (SVM), and logistic regression (LR) on the smaller subsets of selected attributes. Each model is assessed using the area under the Receiver Operating Characteristic (ROC) curve (AUC). Experimental results show that a feature selection technique being stable (or robust) does not mean that the technique has better defect prediction performance (and vice versa), proving that both must be evaluated separately to find the best feature ranking techniques.

The empirical validation of the stability measure and model performance 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. In the experiments, ten runs of five-fold cross-validation were performed. We ranked the features and selected the top features according to their respective scores. The experimental results demonstrate that a robust ranker may not mean better classification model performance.

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 defect prediction model performance of feature selection techniques together on real-world software metrics data.

The rest of the paper is organized as follows. We review relevant literature on feature selection techniques in Section 2. Section 3 provides detailed information about the feature ranking, learners, and performance metric used. Section 4 describes the datasets used in the study. Section 5 presents stability results and analysis, while Section 6 shows model performance results and analysis. Finally, in Section 7, the conclusion is summarized and suggestions for future work are indicated.

2. Related Work

Feature selection is the process of choosing a subset of features (software metrics in our study). Feature selection 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 which collectively have good predictive power. Feature selection techniques can also be categorized as *filters* and *wrappers* [3]. *Filters* are algorithms in which a feature set is selected without involving any learning algorithm. *Wrappers* are algorithms that use feedback from a learner to select features.

The main goal of feature selection is to select a subset of features that minimizes the prediction errors of classifiers. Guyon and Elisseeff [3] 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 [4] 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 [5] 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. [6] 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. [7] applied attribute selection with three filter models and two wrapper models to five software engineering datasets using the WEKA [8] 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. In a recent study [9] by Gao et al., a comparative investigation in the context of software quality estimation is presented for evaluating a proposed hybrid attribute selection approach, in which feature ranking is first used to reduce the search space, followed by a feature subset selection.

The stability of a feature selection method is normally defined as the degree of agreement between its outputs to randomly selected subsets of the same input data [10,11]. 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. [12] used the Spearman rank correlation coefficient. Abeel et al. [13] studied the process for selecting biomarkers from microarray data and presented a general framework for stability analysis of such feature selection techniques. Lustgarten et al. [14] devised a new stability measure called Adjusted Stability Measure (ASM) that can be applied to classifier based feature selecting methods. Kalousis et al. [15] used different measures of correlation to measure the stability of the feature ranker.

To assess robustness of feature selection techniques, past works have used different similarity measures, such as Hamming distance [16], correlation coefficient [15], consistency index [10], and entropy [17]. 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 [10]. The consistency index is a measure of similarity between two different feature subsets. They devised this measure as a way to choose the best set of features for an experiment. If stability was high, then the number of features chosen in rank order by stability that has the minimum local error is chosen as the feature subset. If stability was low, the best individual subset would be used.

3. Filter-based Feature Rankers

This work focuses on filter-based feature ranking. Filter-based feature ranking techniques rank features independently without involving any learning algorithm. Feature ranking consists of scoring each feature according to a particular method, then selecting features based on their scores. In this work, the feature rankers (filters) chosen can be placed into three categories: six commonly used filter based feature selection techniques, eleven threshold-based feature selection techniques (TBFS) that were developed by our research team, and a new filter technique called Signal to Noise. Table 1 contains all of the feature selection techniques used and their abbreviations.

Table 1. List of 18 Filter-Based Feature Selection Techniques

Abbreviation	Name
Standard Filter-Based Feature Selection Techniques	
CS	Chi-squared
GR	Gain Ratio
IG	Information Gain
RF	ReliefF
RFW	ReliefF — Weight by Distance
SU	Symmetric Uncertainty
Threshold-Based Feature Selection Techniques	
FM	F-measure
OR	Odds Ratio
PO	Power
PR	Probabiltiy Ratio
GI	Gini Index
MI	Mutual Information
KS	Kolmogorov-Smirnov Statistic
Dev	Deviance
GM	Geometric Mean
AUC	Area Under the ROC Curve
PRC	Area Under the Precision-Recall Curve
S2N	Signal to Noise

3.1. *Standard Filter-based Feature Ranking Techniques*

The six standard, commonly-used filter-based feature ranking methods include: chi-squared [8], information gain [4,8], gain ratio [8], two types of ReliefF [18], and symmetrical uncertainty [8,19]. All of these feature selection methods are available within WEKA [8]. Since these methods are widely known, we provide only a brief summary; the interested reader should consult with the included references for further details.

The chi-square (CS)[20] test is used to examine if there is ‘no association’ between two attributes, i.e., whether the two variables are independent. CS is more likely to find significance to the extent that (1) the relationship is strong, (2) the sample size is large, and/or (3) the number of values of the two associated features is large.

Information gain, gain ratio, and symmetrical uncertainty are measures based on the concept of entropy, which is based on information theory. Information gain (IG) [21] 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) [4] 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. These four techniques (CS, IG, GR, and SU) utilize the method of Fayyad and Irani [22] to discretize continuous attributes, and all four methods are bivariate, considering the relationship between each attribute and the class, excluding the other independent variables.

Relief is an instance-based feature ranking technique [23]. ReliefF is an extension of the Relief algorithm that can handle noise and multi-class datasets. 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. *Threshold-based Feature Ranking Techniques*

Eleven threshold-based feature selection techniques (TBFS) were developed and implemented by our research group within WEKA [8]. 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 (software predictor variable) is then paired individually with the class attribute (fault-prone or not-fault-prone label) and the reduced dataset is evaluated using five different performance metrics based on a 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 relative class distribution 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, the values above the threshold are classified as positive, and negative otherwise. We then consider swapping the positive and negative, i.e. values about the threshold are classified as negative, and positive otherwise. Whichever direction of the positive and negative labeling produces the more optimal attribute values is used. In a binary classification problem such as fault-prone (positive) or not-fault-prone (negative), there are four possible classification rates: true positive rate TPR), true negative rate (TNR), false positive rate (FPR), false negative rate (FNR), as well as two additional commonly-used performance metrics, precision (PRE) and negative predicted value (NPV) [24]. These four classification rates can be calculated at each threshold $t \in [0,1]$ relative to the normalized attribute $\hat{F}^j$. The threshold-based feature ranking technique utilizes the classification 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 [8].

$$\text{FM} = \max_{t\in[0,1]} \frac{(1+\beta^2) \times TPR(t) \times PRE(t)}{\beta^2 \times TPR(t) + PRE(t)}. \quad (1)$$

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. Threshold-based feature ranking technique $\omega \in$ {FM, OR, PO, PR, GI, MI, KS, Dev,
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 $\omega$ using attribute $\hat{F}^j$, $\omega_i(\hat{F}^j)$.
Create feature ranking $\mathbb{R}$ using $\omega_i(\hat{F}^j) \forall j$.
Select features according to feature ranking $\mathbb{R}$ and a predefined threshold.
```

β is set to 1 in this study.

- *Odds Ratio (OR)*: is the maximum value of the ratio of the product of correct (true positive rate times true negative rate) to incorrect (false positive rate times false negative rate) predictions. The odds ratio is defined as:

$$\text{OR} = \max_{t \in [0,1]} \left(\frac{TPR(t)}{FPR(t)} \right) \left(\frac{TNR(t)}{FNR(t)} \right) \tag{2}$$

- *Power (PO)*: is a measure that avoids common false positive cases while giving stronger preference for positive cases [25]. Power is defined as:

$$\text{PO} = \max_{t \in [0,1]} \left((TNR(t))^k - (FNR(t))^k \right) \tag{3}$$

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 [25]. The probability ratio is defined as:

$$\text{PR} = \max_{t \in [0,1]} \frac{TPR(t)}{FPR(t)} \tag{4}$$

- *Gini Index (GI)*: measures the impurity of a dataset [26]. GI for the attribute is then the minimum Gini index at all decision thresholds $t \in [0, 1]$.

$$\text{GI} = \min_{t \in [0,1]} [2PRE(t)(1 - PRE(t)) + 2NPV(t)(1 - NPV(t))]. \tag{5}$$

- *Mutual Information (MI)*: measures the mutual dependence of the two random variables [27]. 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 function of the attribute values of instances in each class [28]. The larger the distance between the distribution functions, the better the attribute is able to distinguish between the two classes. It is effectively the maximum difference between the curves generated by the true positive and false positive rates as the decision threshold changes from 0 and 1.
- *Deviance (Dev)*: is the residual sum of squares based on a threshold t. It measures the sum of the squared errors from the mean class given a partitioning of the space based on the threshold t. As deviance represents errors, the minimum value is considered optimal.
- *Geometric Mean (GM)*: is a single-value performance measure that ranges from 0 to 1 which is calculated by finding the maximum geometric mean of TPR and TNR as the decision threshold is varied between 0 and 1:

$$\text{GM} = \max_{t \in [0,1]} \sqrt{TPR(t) \times TNR(t)} \tag{6}$$

- *Area Under ROC (Receiver Operating Characteristic) Curve* (AUC): has been widely used to measure classification model performance [29]. AUC is a single-value measurement that ranges from 0 to 1. 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.

3.3. *Signal-to-Noise*

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. The signal-to-noise (S2N) ratio can also be used as feature ranking method [30]. For a binary class problem (such as fp, nfp), the equation for signal to noise is:

$$S2N = (\mu_P - \mu_N)/(\sigma_P + \sigma_N) \tag{7}$$

where μ_P and μ_N are the mean values of that particular attribute in all of the instances which belong to a specific class, either P or N (the positive and negative classes). σ_P and σ_N are the standard deviations of that particular attribute as it relates to the two classes, respectively. If one 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. The larger the S2N ratio, the more relevant a feature is to the dataset [31].

4. Dataset Characteristics

Experiments conducted in this study used software metrics and defect data collected from real-world software projects, including a very large telecommunications software system (denoted as LLTS) [32], the Eclipse project [33,34], and NASA software project KC1 [35].

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 [32]. The dependent variable is the class of the program module: fault-prone (*fp*) or not fault-prone (*nfp*). A program module with one or more faults is considered *fp*, and *nfp* otherwise.

From the PROMISE data repository [34], 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 the 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 while we use $t \in \{5, 4, 2\}$ for release 2.1. These values are selected in order to have datasets with different level of class imbalance. 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 [35], 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 *fp* class. The three datasets are named KC1-20, KC1-10, and KC1-5.

Table 2 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).

5. Experiments - Stability

We used sixteen different software datasets to test the stability of our eighteen feature selection techniques. With each feature ranker and dataset combination, we used four different levels of dataset perturbation along with nine different numbers of features chosen.

5.1. *Dataset Perturbation*

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 re-

Table 2. Software Dataset Characteristics

	Data	**#Metrics**	**#Modules**	**#fp**	**%fp**	**#nfp**	**%nfp**
LLTS	SP1	42	3649	229	6%	3420	94%
	SP2	42	3981	189	5%	3792	95%
	SP3	42	3541	47	1%	3494	99%
	SP4	42	3978	92	2%	3886	98%
Eclipse	Eclipse2.0-10	208	377	23	6%	354	94%
	Eclipse2.0-5	208	377	52	14%	325	86%
	Eclipse2.0-3	208	377	101	27%	276	73%
	Eclipse2.1-5	208	434	34	8%	400	92%
	Eclipse2.1-4	208	434	50	12%	384	88%
	Eclipse2.1-2	208	434	125	29%	309	71%
	Eclipse3.0-10	208	661	41	6%	620	94%
	Eclipse3.0-5	208	661	98	15%	563	85%
	Eclipse3.0-3	208	661	157	24%	504	76%
KC1	KC1-20	62	145	10	7%	135	93%
	KC1-10	62	145	21	14%	124	86%
	KC1-5	62	145	36	25%	109	75%

moved $1-c$ 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, 16 datasets $\times 4$ levels of perturbation $\times 30$ repetitions $= 1920$ datasets are generated.

5.2. *Feature Selection*

For each dataset and feature ranking technique, the features are ranked first according to their relevance to the class. The rankings are applied to each combination of dataset and level of perturbation. Therefore, (16 original datasets $\times$ 18 feature rankers) + (16 datasets $\times$ 4 levels of perturbation $\times$ 30 repetitions $\times$ 18 feature rankers) = 34848 different rankings were computed. 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 [2].

5.3. *Stability Measure*

In order to measure stability, we decided to use consistency index [10] 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 [10] is obtained as follows:

$$I_C(T_i, T_j) = \frac{dn - k^2}{k(n-k)}, \tag{8}$$

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 the 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. Thus, given a dataset and a ranker, 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).

5.4. ***Experimental Results***

The experiment was conducted using eighteen feature selection techniques on sixteen real-world software datasets. There are three main factors which can be examined to observe their effects on stability: degree of perturbation, number of features used, and choice of filter. Tables 3 through 6 contain the results of our stability experiments. Each table focuses on one of the following three factors: level of perturbation, range of features used, and choice of filter. All datasets are averaged together. The top value at each column is in **boldface**. In general, it can be observed that RF, RFW, and PRC show the most stability and GR shows the least stability. This conclusion can also be observed in Table 7. Fig. 1 shows the effect of the degree of dataset perturbation on the stability of feature ranking techniques across all 16 datasets and nine feature subsets. From this figure, we can observe 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. This leads us to state that after enough change any feature selection method becomes unstable. The second trend is that in general, stability increases as more features are used. Table 8 shows that the performance increases as more features are used.

To further investigate these results we performed an ANalysis Of VAriance (ANOVA) [36] 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 18 filters. For the ANOVA test, the perturbation level was held constant at 80%, and the results from all sixteen datasets were taken into account together. The ANOVA results are presented in Table 9. The p value of Factor A is 0, which indicates there was a significant difference between the average I_C values of the eighteen rankers.

Additional multiple comparisons for the main factor were performed to investigate the differences among the respective groups (levels). All tests of statistical

Table 3. Average Stability Across All Sixteen Datasets, 67%

	Number of Features Used								
	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
FM	0.6349	0.5984	0.6039	0.6036	0.6165	0.6326	0.6548	0.6710	0.6869
OR	0.4164	0.3906	0.4313	0.4582	0.4554	0.4683	0.4981	0.5213	0.5325
PO	0.6271	0.5959	0.5910	0.6104	0.6386	0.6508	0.6691	0.6817	0.6924
PR	0.4075	0.4307	0.4307	0.4463	0.4774	0.5047	0.5257	0.5436	0.5646
GI	0.4227	0.4472	0.4446	0.4698	0.4899	0.5156	0.5377	0.5668	0.5821
MI	0.6114	0.5926	0.5860	0.5890	0.5942	0.6103	0.6236	0.6474	0.6568
KS	0.5895	0.5894	0.5850	0.6016	0.6091	0.6222	0.6329	0.6520	0.6648
Dev	0.6056	0.5662	0.5803	0.5876	0.6077	0.6246	0.6464	0.6593	0.6785
GM	0.5909	0.5772	0.5797	0.5863	0.6074	0.6153	0.6328	0.6480	0.6676
AUC	0.6590	0.6584	0.6843	0.6940	0.7109	0.7348	0.7483	0.7643	0.7749
PRC	0.6915	0.6664	0.7132	0.7200	**0.7392**	**0.7482**	**0.7682**	**0.7842**	**0.7940**
S2N	0.6944	0.6571	0.6391	0.6350	0.6237	0.6250	0.6277	0.6381	0.6429

Table 4. Average Stability Across All Sixteen Datasets, 80%

	Number of Features Used								
	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
FM	0.7172	0.6945	0.7094	0.7040	0.7055	0.7097	0.7275	0.7487	0.7600
OR	0.5142	0.5013	0.5488	0.5804	0.5855	0.5965	0.6124	0.6339	0.6474
PO	0.7211	0.6863	0.6850	0.7019	0.7276	0.7310	0.7497	0.7620	0.7632
PR	0.5424	0.5443	0.5658	0.5897	0.6229	0.6451	0.6645	0.6801	0.6870
GI	0.5650	0.5574	0.5677	0.5998	0.6162	0.6429	0.6533	0.6769	0.6859
MI	0.6907	0.6677	0.6845	0.6769	0.6896	0.7003	0.7087	0.7245	0.7325
KS	0.6761	0.6803	0.6782	0.6935	0.7103	0.7119	0.7282	0.7386	0.7501
Dev	0.6934	0.6579	0.6798	0.7000	0.7102	0.7190	0.7299	0.7346	0.7488
GM	0.6731	0.6683	0.6789	0.6847	0.7045	0.7073	0.7289	0.7372	0.7511
AUC	0.7527	0.7422	0.7782	0.7740	0.7829	0.8067	0.8121	0.8297	0.8343
PRC	0.7673	0.7439	0.8039	0.7983	0.7977	**0.8190**	**0.8370**	**0.8488**	**0.8591**
S2N	0.7819	0.7628	0.7314	0.7269	0.7127	0.7152	0.7214	0.7372	0.7497

significance utilize a significance level α of 5%. The multiple comparison test results are shown in Fig. 2. The figure presents each group mean with a symbol ($\circ$) and the 95% confidence interval with 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 presented in this work, and the assumptions for constructing ANOVA models were validated. The results show the following facts: For factor A, GR and SU performed

Table 5. Average Stability Across All Sixteen Datasets, 90%

	Number of Features Used								
	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
FM	0.8247	0.7811	0.7923	0.7878	0.7949	0.7949	0.8163	0.8310	0.8383
OR	0.6511	0.6429	0.7016	0.7249	0.7176	0.7327	0.7371	0.7532	0.7618
PO	0.7961	0.7582	0.7706	0.7906	0.8067	0.8090	0.8254	0.8432	0.8496
PR	0.6791	0.6877	0.6914	0.7002	0.7390	0.7581	0.7725	0.7831	0.7940
GI	0.6996	0.7037	0.7123	0.7254	0.7506	0.7737	0.7796	0.7946	0.8008
MI	0.7732	0.7673	0.7904	0.7743	0.7875	0.7981	0.7929	0.8000	0.7964
KS	0.7762	0.7887	0.7854	0.7818	0.7998	0.8100	0.8255	0.8290	0.8312
Dev	0.7905	0.7471	0.7648	0.7768	0.7943	0.8004	0.8092	0.8180	0.8260
GM	0.7723	0.7861	0.7823	0.7882	0.8089	0.8052	0.8203	0.8327	0.8340
AUC	0.8268	0.7896	0.8531	0.8506	0.8504	0.8595	0.8646	0.8755	0.8795
PRC	0.8452	0.8290	0.8842	0.8696	0.8655	**0.8768**	**0.8896**	**0.8949**	**0.8999**
S2N	0.8636	0.8444	0.8341	0.8324	0.8112	0.8143	0.8248	0.8323	0.8471

Table 6. Average Stability Across All Sixteen Datasets, 95%

	Number of Features Used								
	2	3	4	5	6	7	8	9	
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
FM	0.8685	0.8242	0.8438	0.8408	0.8483	0.8433	0.8695	0.8792	0.8821
OR	0.7415	0.7328	0.8022	0.8255	0.8072	0.8041	0.8032	0.8191	0.8310
PO	0.8615	0.8335	0.8462	0.8498	0.8644	0.8661	0.8786	0.8803	0.8866
PR	0.7628	0.7519	0.7664	0.7724	0.8176	0.8265	0.8345	0.8437	0.8531
GI	0.7873	0.7863	0.7991	0.8124	0.8407	0.8541	0.8519	0.8623	0.8667
MI	0.8253	0.8247	0.8520	0.8395	0.8451	0.8572	0.8455	0.8504	0.8436
KS	0.8357	0.8361	0.8587	0.8494	0.8599	0.8671	0.8835	0.8836	0.8847
Dev	0.8448	0.7945	0.8190	0.8456	0.8573	0.8549	0.8609	0.8651	0.8709
GM	0.8285	0.8508	0.8436	0.8537	0.8583	0.8594	0.8775	0.8889	0.8866
AUC	0.8820	0.8453	0.9018	0.8927	0.8928	0.9051	0.9038	0.9103	0.9124
PRC	0.8958	0.8754	0.9175	0.9153	0.8988	**0.9107**	**0.9202**	**0.9242**	**0.9271**
S2N	0.9084	0.8826	0.8867	0.8714	0.8528	0.8626	0.8691	0.8812	0.8889

significantly worst among the 18 rankers and RF and RFW performed best, followed by PRC- and AUC-based feature selection techniques.

6. Experiments - Defect Prediction Model 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 classi-

Table 7. Average Stability of the Filters Across All Levels of Perturbation and Feature Subset Sizes

Filter	Average Stability
CS	0.6845
GR	0.5335
IG	0.6868
RF	0.8352
RFW	0.8281
SU	0.5893
S2N	0.7731
FM	0.7539
OR	0.6384
PO	0.7584
PR	0.6585
GI	0.6734
MI	0.7347
KS	0.7472
Dev	0.7408
GM	0.7449
AUC	0.8121
PRC	0.8317

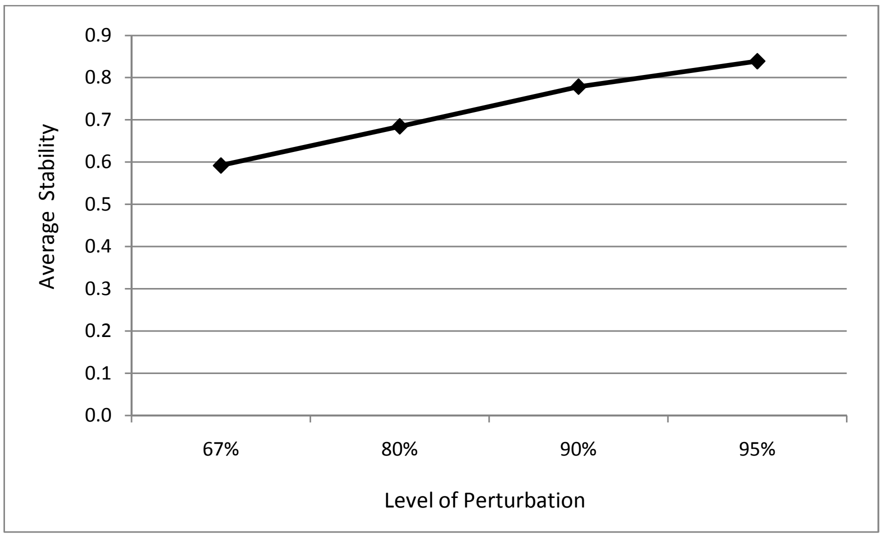


Fig. 1. Degree of Perturbation Impact on Stability

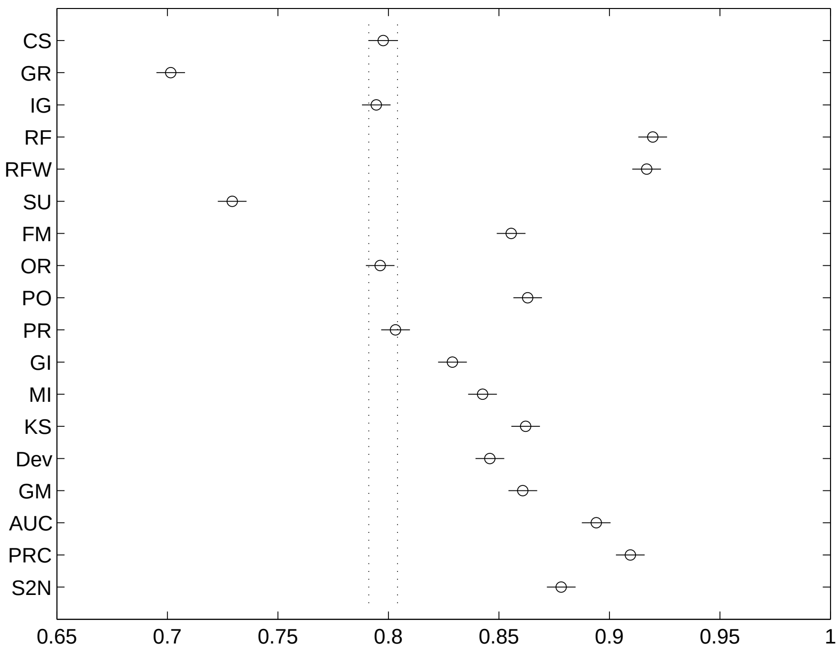
fication model performance. Software defect prediction models have been used to improve the fault prediction and risk assessment process. Finding faulty components in a software system can lead to a more reliable final system and reduce development and maintenance costs.

Table 8. Average Stability of the Feature Subset Sizes Used Across All Levels of Perturbation and All Filters

Number of Features	Average Stability
2	0.7056
3	0.6944
4	0.7072
5	0.7110
6	0.7198
7	0.7272
8	0.7387
9	0.7502
10	0.7582

Table 9. Analysis of Variance, Stability

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	267	17	15.7056	524.92	0
Error	2326.03	77742	0.0299		
Total	2593.02	77759			



(a) Factor A, Filters

Fig. 2. Multiple Comparisons, Stability

6.1. *Classifiers*

Software defect prediction models are built with three well-known classification algorithms including naïve Bayes (NB), support vector machine (SVM), and logistic regression (LR). All three learners themselves do not have a built-in feature selection capability and are commonly used in the software engineering and other data mining applications. All classifiers were implemented in the WEKA tool [8]. We used default parameter settings for the different learners as specified in WEKA. Parameter settings were changed only when doing so improved classifier performance significantly.

(1) Naïve Bayes classifier (NB) [37] utilizes Bayes's rule of conditional probability and is termed 'naïve' because it assumes conditional independence of the features.
(2) Support Vector Machine (SVM) [38], also called SMO in WEKA [8], had two changes to the default parameters: the 'complexity constant c' was set to 5.0 and 'build Logistic Models' was set to true. By default, a linear kernel was used.
(3) Logistic Regression (LR) [39] is a statistical regression model for categorical prediction by fitting data to a logistic curve.

6.2. *Performance Metric*

Traditional performance measures such as F-measure, overall classification accuracy, or its complement, misclassification rate, are inappropriate when dealing with the classification of imbalanced data. In a domain such as software defect prediction, the number of *fp* (fault-prone) modules is much lower than the *nfp* (not fault-prone) modules. Instead, we use a performance metric that considers the ability of a classifier to differentiate between the two classes: the area under the ROC (Receiver Operating Characteristic) curve (AUC). It has been shown that AUC has lower variance and more reliability than other performance metrics (such as precision, recall, F-measure) [40]. AUC has been frequently used in the software engineering domain [40,41,42].

In a two-group classification problem, such as fault-prone and not fault-prone, there can be four possible prediction outcomes: true positive (TP), false positive (FP), true negative (TN), and false negative (FN), where positive represents fault-prone and negative represents not fault-prone. The AUC is a single-value measurement whose value ranges from 0 to 1. The ROC curve is used to characterize the trade-off between hit (true positive) rate and false alarm (false positive) rate [29]. The true positive rate is calculated as $\frac{|TP|}{|TP|+|FN|}$ and the false positive rate is computed as $\frac{|FP|}{|FP|+|TN|}$. A classifier that provides a large area under the curve is preferable over a classifier with a smaller area under the curve. Traditional performance metrics consider only the default decision threshold of 0.5. ROC curves illustrate the performance across all decision thresholds. A perfect classifier provides an AUC that equals 1.

Note that the metric used to measure the performance of the classifiers is completely independent from the metric in the TBFS algorithm. AUC is used both to select the most predictive subset of features in TBFS and to evaluate the classification models constructed using this set of features.

6.3. *Experimental Results*

Classifiers were built with three well-known classification algorithms, naïve Bayes (NB), support vector machine (SVM), and logistic regression (LR) [8]. For our experiments, the default settings of WEKA [8] are used. The classification models are evaluated using the area under the ROC (Receiver Operating Characteristic) curve (AUC) performance metric. 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 eighteen rankers separately. Once the attributes are ranked, the top four attributes are selected (as well as the class attribute) [2] to yield the final training data. After feature selection, we applied the classifier to the training datasets with the selected features, and then we used AUC to evaluate the performance of the classification model. In total, 18 rankers × 16 datasets × 10 runs × 5 folds = 14400 combinations of feature ranking techniques were employed, and correspondingly 43200 classification models (14400 × 3 classifiers = 43200) were built.

Table 10. Classification Performance, NB

	LLTS				Eclipse									KC1			
Filter	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
CS	0.790	0.811	0.825	0.760	0.781	0.864	0.810	0.823	0.823	0.781	0.873	0.904	0.845	0.736	0.761	0.846	0.815
GR	0.707	0.702	0.760	0.723	0.786	0.739	0.687	0.766	0.761	0.785	0.832	0.889	0.813	0.734	0.771	**0.865**	0.770
IG	0.788	0.811	0.820	0.791	0.807	0.874	0.810	0.871	0.854	0.784	0.906	0.904	0.841	0.722	0.752	0.849	0.824
RF	**0.792**	0.810	0.817	0.766	0.829	0.852	0.745	0.742	0.683	0.678	0.820	0.810	0.772	**0.799**	0.758	0.826	0.781
RFW	0.790	0.804	0.809	0.755	0.830	0.855	0.767	0.759	0.653	0.818	0.753	0.806	0.779	0.786	0.751	0.827	0.784
SU	0.766	0.755	0.779	0.751	0.802	0.852	0.796	0.831	0.824	0.780	0.841	0.900	0.834	0.729	**0.774**	0.852	0.804
FM	0.787	0.813	0.822	0.759	0.834	0.866	0.787	0.877	0.842	0.788	0.892	0.902	0.842	0.758	0.708	0.853	0.821
OR	0.696	0.789	0.672	0.746	0.803	0.869	0.807	0.798	0.793	0.788	0.824	0.903	0.832	0.726	0.773	0.851	0.792
PO	0.786	0.809	0.818	0.789	0.818	0.859	0.811	0.798	0.794	0.778	0.906	0.909	0.846	0.726	0.765	0.842	0.816
PR	0.707	0.779	0.673	0.752	0.788	0.810	0.802	0.751	0.780	0.694	0.831	0.899	0.830	0.777	0.762	0.827	0.779
GI	0.707	0.777	0.702	0.699	0.801	0.807	0.803	0.748	0.786	0.692	0.831	0.898	0.829	0.777	0.763	0.823	0.778
MI	0.774	0.792	0.819	0.774	0.827	0.857	0.792	0.884	0.850	0.782	0.860	0.906	0.833	0.760	0.730	0.861	0.819
KS	0.773	0.769	0.814	0.753	0.805	0.848	0.800	0.879	0.852	0.786	0.877	0.903	0.833	0.778	0.725	0.859	0.816
Dev	0.787	0.811	0.818	0.781	0.831	0.865	0.782	0.856	0.835	0.785	0.891	0.903	0.841	0.756	0.747	0.849	0.821
GM	0.774	0.769	0.815	0.749	0.803	0.832	0.802	0.879	0.851	0.788	0.858	0.903	0.835	0.762	0.703	0.859	0.811
AUC	0.774	0.811	0.807	0.766	0.862	0.875	0.808	0.886	0.849	0.782	0.906	0.904	0.837	0.763	0.698	0.847	0.823
PRC	0.790	0.812	0.816	0.769	0.828	0.868	0.809	0.882	0.854	0.785	0.895	0.905	0.840	0.710	0.730	0.831	0.820
S2N	0.788	**0.818**	**0.825**	**0.817**	**0.868**	**0.884**	**0.809**	**0.895**	**0.864**	**0.838**	**0.919**	**0.910**	**0.894**	0.791	0.740	0.841	**0.844**

Experimental results are reported in Table 10 through 12. 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

Table 11. Classification Performance, SVM

	LLTS				Eclipse									KC1			
Filter	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
CS	0.674	0.693	0.735	0.649	0.847	0.916	0.860	0.885	0.889	0.889	0.922	0.943	0.909	0.785	0.769	0.838	0.825
GR	0.632	0.617	0.648	0.637	0.843	0.811	0.742	0.830	0.835	0.883	0.882	0.938	0.904	0.768	0.784	0.838	0.787
IG	**0.697**	0.692	**0.751**	0.660	0.864	0.918	0.861	0.913	0.902	0.890	0.935	0.942	0.908	0.784	0.763	0.869	0.834
RF	0.672	0.673	0.691	0.662	0.847	0.890	0.801	0.713	0.638	0.673	0.866	0.813	0.804	**0.811**	0.712	0.805	0.754
RFW	0.633	0.685	0.647	0.658	0.846	0.888	0.798	0.776	0.608	0.878	0.733	0.811	0.808	0.790	0.684	0.804	0.753
SU	0.643	0.652	0.664	0.630	0.846	0.905	0.857	0.898	0.886	0.889	0.902	0.941	0.909	0.781	0.786	0.847	0.815
FM	0.680	0.694	0.688	0.675	0.866	0.914	0.853	0.913	0.903	0.890	0.929	0.942	0.909	0.796	0.756	0.835	0.828
OR	0.649	0.702	0.638	**0.681**	0.842	0.915	0.863	0.863	0.863	0.891	0.900	0.943	0.905	0.766	**0.786**	0.843	0.816
PO	0.666	0.724	0.688	0.664	0.862	0.910	0.858	0.858	0.874	0.889	0.934	**0.944**	0.908	0.786	0.780	0.839	0.824
PR	0.627	0.666	0.621	0.640	0.834	0.862	0.859	0.826	0.846	0.796	0.888	0.942	0.904	0.784	0.782	0.830	0.794
GI	0.633	0.663	0.597	0.679	0.830	0.863	0.859	0.822	0.848	0.791	0.889	0.942	0.904	0.784	0.784	0.833	0.795
MI	0.690	0.697	0.720	0.634	0.867	0.914	0.855	0.922	0.906	0.891	0.922	0.941	0.909	0.793	0.754	0.858	0.829
KS	0.687	0.653	0.706	0.598	0.840	0.910	0.859	0.930	**0.906**	0.890	0.923	0.941	0.910	0.794	0.739	0.874	0.822
Dev	0.682	0.727	0.710	0.646	0.863	0.914	0.848	0.902	0.895	0.890	0.929	0.943	0.910	0.804	0.761	0.835	0.829
GM	0.686	0.633	0.720	0.636	0.841	0.897	0.860	**0.930**	0.904	0.890	0.916	0.940	0.910	0.794	0.736	**0.876**	**0.823**
AUC	0.683	0.688	0.696	0.634	0.884	**0.920**	0.863	0.927	0.906	**0.892**	**0.936**	0.941	0.909	0.792	0.748	0.867	0.830
PRC	0.690	0.720	0.707	0.638	0.861	0.917	**0.864**	0.915	0.899	0.892	0.931	0.943	0.908	0.786	0.768	0.843	0.830
S2N	0.675	**0.740**	0.715	0.645	**0.885**	0.905	0.848	0.913	0.894	0.883	0.923	0.935	**0.913**	0.779	0.750	0.835	0.827

Table 12. Classification Performance, LR

	LLTS				Eclipse									KC1			
Filter	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
CS	0.803	0.822	0.838	0.799	0.837	0.909	0.870	0.888	0.887	0.886	0.916	0.944	0.912	0.771	0.776	0.841	0.856
GR	0.744	0.716	0.770	0.748	0.832	0.825	0.759	0.832	0.841	0.881	0.892	0.939	0.906	0.775	0.787	0.827	0.817
IG	0.801	0.814	0.839	0.816	0.855	0.910	0.866	0.910	0.900	0.888	**0.927**	0.943	0.912	0.771	0.784	**0.861**	**0.862**
RF	**0.809**	0.820	0.839	0.797	0.859	0.883	0.800	0.774	0.718	0.722	0.878	0.843	0.804	**0.815**	0.761	0.809	0.808
RFW	0.809	0.817	**0.839**	0.797	0.843	0.883	0.803	0.810	0.686	0.875	0.792	0.837	0.807	0.800	0.733	0.785	0.807
SU	0.782	0.788	0.792	0.770	0.831	0.900	0.860	0.891	0.884	0.884	0.901	0.943	0.910	0.773	**0.788**	0.854	0.847
FM	0.802	0.821	0.830	0.796	0.869	0.912	0.863	0.915	0.901	0.888	0.924	0.944	0.911	0.788	0.766	0.842	0.861
OR	0.756	0.804	0.705	0.774	0.811	**0.913**	0.862	0.865	0.861	0.888	0.896	0.943	0.908	0.769	0.775	0.832	0.835
PO	0.801	0.823	0.836	0.811	0.848	0.903	0.870	0.843	0.868	0.888	0.926	0.943	0.911	0.778	0.781	0.823	0.853
PR	0.765	0.793	0.706	0.775	0.825	0.858	0.857	0.824	0.843	0.794	0.886	0.942	0.906	0.783	0.779	0.801	0.821
GI	0.765	0.793	0.715	0.774	0.817	0.860	0.857	0.826	0.844	0.789	0.886	0.942	0.906	0.783	0.781	0.800	0.821
MI	0.789	0.790	0.838	0.795	0.863	0.911	0.864	0.917	0.908	0.886	0.917	0.943	0.910	0.787	0.779	0.845	0.859
KS	0.786	0.778	0.835	0.764	0.818	0.909	0.868	0.929	**0.909**	0.888	0.910	0.941	0.911	0.790	0.754	0.842	0.852
Dev	0.801	0.819	0.832	0.799	0.844	0.912	0.858	0.907	0.895	0.885	0.924	0.944	0.912	0.796	0.780	0.830	0.859
GM	0.786	0.781	0.835	0.769	0.821	0.897	0.869	**0.929**	0.907	0.888	0.908	0.940	0.911	0.782	0.766	0.847	0.852
AUC	0.784	0.814	0.832	0.778	0.876	0.912	0.875	0.923	0.900	0.889	0.927	**0.945**	0.913	0.783	0.744	0.849	0.859
PRC	0.803	0.822	0.830	0.793	0.856	0.911	**0.877**	0.907	0.901	**0.890**	0.922	0.944	0.911	0.778	0.764	0.838	0.859
S2N	0.807	**0.828**	0.837	**0.820**	**0.881**	0.901	0.849	0.922	0.900	0.884	0.922	0.938	**0.923**	0.802	0.763	0.819	0.862

demonstrate that among the eighteen rankers, S2N outperformed the others for 19 out of 48 cases, while the stability performance of S2N is moderate.

We also carried out a two-way ANOVA test [36] built on the raw results presented in Tables 10 through 12. The two factors include: Factor A, which represents the three classifiers, and Factor B, which represents the eighteen rankers. In this ANOVA test, the results from all sixteen 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 13. The test results indicate that for all the two main factors and their interaction, the alternative hypothesis was accepted, since all p-values (last column of the table) were less than the specified cutoff 0.05. The multiple comparison results are presented in Fig. 3. For Factor A, LR performed significantly better than the other two learners and NB performed worst. For Factor B, S2N performed best, followed by IG and AUC; RF and RFW performed worst, followed by GR.

Table 13. Analysis of Variance, Classification

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	2.3554	2	1.17768	213.38	0
B	3.6654	17	0.21561	39.07	0
A$\times$B	0.3169	34	0.00932	1.69	0.0074
Error	47.3866	8586	0.00552		
Total	53.7242	8639			

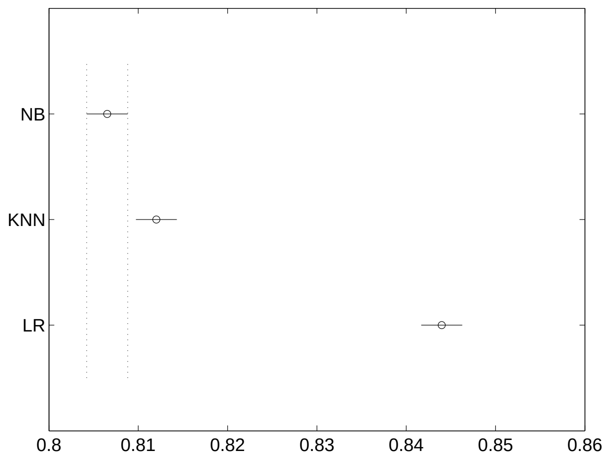
Comparing the results found in Figures 2a and 3b allows us to better understand the relationship between stability and model performance, as these two multiple comparisons tests show the relative performance of the various filters according to the two different schemes. First, we note that the top filters in terms of stability (RF and RFW) are at the very bottom of the list in terms of classification. On the other hand, the second-best filter in terms of classification (IG) comes in 16th out of 18 filters in terms of stability. This shows that good performance in terms of one does not necessarily predict good performance in terms of the other. However, some rankers, such as S2N and the AUC-, PRC- and FM-based TBFS rankers, showed decent performance according to both schemes. Thus, neither positive nor negative correlation can be found between stability and classification performance.

7. Conclusion

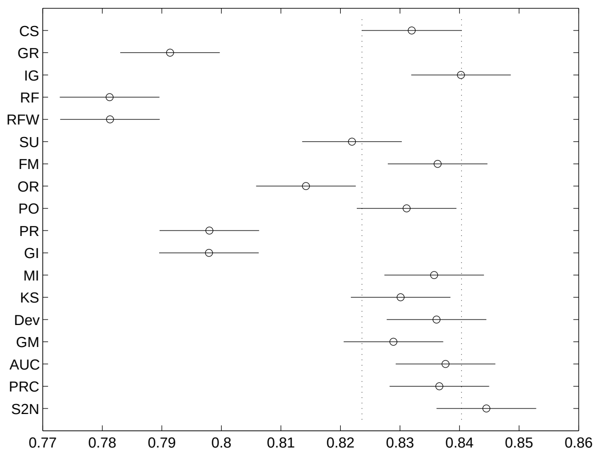
Feature (software metric) selection plays an important role in the software engineering domain. This empirical study investigates the stability (robustness) and classification performance of eighteen filter-based feature ranking techniques on three real-world software projects. For the evaluation of the feature ranking techniques, the consistency index proposed by Kuncheva [10] 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 feature ranking performed on this dataset will be more stable.

We also built classification models using naïve Bayes (NB), support vector machines (SVM), and logistic regression (LR) on the smaller subsets of selected attributes. The classification accuracy is evaluated in terms of the AUC performance metric. The experimental results demonstrate that the signal-to-noise ranker per-

November 12, 2012 19:47



(a) Factor A, Learners



(b) Factor B, Filters

Fig. 3. Multiple Comparisons, Classification

formed 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 it performed significantly worse than other rankers in terms of model performance. The main conclusion of our study is that just because one ranker performed best in terms of robustness, that doesn't mean that the ranker outperforms other rankers in terms of model performance, and vice versa.

Based on these results, we recommend the use of the signal-to-noise ranker for selecting software metrics when performing software quality analysis. This is because, although stability is important to ensure model reusability when applied to new data, performance is necessary to ensure that the model is useful in the first place. PRC may also be a decent option in some circumstances, as it shows slightly greater stability but slightly lesser classification performance than signal-to-noise. In addition, we would like to reiterate the importance of examining both the stability and classification performance of feature rankers before using them in a large-scale system; performing well with one does not predict how the ranker will perform with the other, and both are important to building the best possible models.

Future work may include experiments using additional datasets from other software engineering and non-software engineering domains, and experiments with other ranking techniques and classifiers for building classification models.

References

1. 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).* Miami, FL: IEEE Computer Society, December 2009, pp. 507–514.
2. 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*, May 2011, pp. 69–74.
3. I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, March 2003.
4. M. A. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1437 – 1447, Nov/Dec 2003.
5. H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005.
6. Z. Chen, T. Menzies, D. Port, and B. Boehm, "Finding the right data for software cost modeling," *IEEE Software*, vol. 22, no. 6, pp. 38–46, November 2005.
7. D. Rodriguez, R. Ruiz, J. Cuadrado-Gallego, and J. Aguilar-Ruiz, "Detecting fault modules applying feature selection to classifiers," in *Proceedings of 8th IEEE International Conference on Information Reuse and Integration*, Las Vegas, Nevada, August 13-15 2007, pp. 667–672.
8. I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.
9. K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: an investigation on feature selection techniques," *Software: Practice and Experience*, vol. 41, no. 5, pp. 579–606, 2011.
10. 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.* Anaheim, CA, USA: ACTA Press, 2007, pp. 390–395.
11. 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, New York, NY, USA, 2009, pp. 567–576.
12. 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.* Berlin, Heidelberg: Springer-Verlag, 2008, pp. 313–325.
13. 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*, vol. 26, no. 3, pp. 392–398, February 2010.
14. J. L. Lustgarten, V. Gopalakrishnan, and S. Visweswaran, "Measuring stability of feature selection in biomedical datasets," in *AMIA 2009 Symposium Proceedings*, 2009, pp. 406–410.
15. A. Kalousis, J. Prados, and M. Hilario, "Stability of feature selection algorithms: a study on high-dimensional spaces," *Knowledge and Information Systems*, vol. 12, no. 1, pp. 95–116, Dec. 2006.
16. K. Dunne, P. Cunningham, and F. Azuaje, "Solutions to Instability Problems with Sequential Wrapper-Based Approaches To Feature Selection," Department of Computer Science, Trinity College, Dublin, Ireland, Tech. Rep. TCD-CD-2002-28, 2002.
17. 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*, ser. CAIP'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 929–936.
18. I. Kononenko, "Estimating attributes: Analysis and extensions of RELIEF," in *European Conference on Machine Learning.* Springer Verlag, 1994, pp. 171–182.
19. M. A. Hall and L. A. Smith, "Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper," in *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, 1999, pp. 235–239.
20. A. C. Cameron and P. K. Trivedi, *Regression Analysis of Count Data.* Cambridge University Press, 1998.
21. J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.
22. U. M. Fayyad and K. B. Irani, "On the handling of continuous-valued attributes in decision tree generation," *Machine Learning*, vol. 8, pp. 87–102, 1992.
23. K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of 9th International Workshop on Machine Learning*, 1992, pp. 249–256.
24. G. M. Weiss and F. Provost, "Learning when training data are costly: the effect of class distribution on tree induction," *Journal of Artificial Intelligence Research*, no. 19, pp. 315–354, 2003.
25. G. Forman, "An extensive empirical study of feature selection metrics for text classification," *J. Mach. Learn. Res.*, vol. 3, pp. 1289–1305, 2003.
26. L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees.* Boca Raton, FL: Chapman and Hall/CRC Press, 1984.
27. 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*, vol. 27, no. 8, pp. 1226–1238, 2005.
28. T. M. Khoshgoftaar and N. Seliya, "Fault-prediction modeling for software quality estimation: Comparing commonly used techniques," *Empirical Software Engineering Journal*, vol. 8, no. 3, pp. 255–283, 2003.
29. T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, June 2006.

30. 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*, Berlin, Heidelberg, 2008, pp. 112–119.
31. M. Wasikowski and X. wen Chen, “Combating the small sample class imbalance problem using feature selection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1388–1400, 2010.
32. 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*, Las Vegas, Nevada, August 10-12 2009, pp. 272–277.
33. G. Boetticher, T. Menzies, and T. Ostrand, “Promise repository of empirical software engineering data,” http://promisedata.org/, 2007.
34. T. Zimmermann, R. Premraj, and A. Zeller, “Predicting defects for eclipse,” in *ICSEW ’07: Proceedings of the 29th International Conference on Software Engineering Workshops*, Washington, DC, USA, 2007, p. 76.
35. 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 Transactions on Software Engineering*, vol. 35, no. 2, pp. 293–304, 2009.
36. M. L. Berenson, M. Goldstein, and D. Levine, *Intermediate Statistical Methods and Applications: A Computer Package Approach.* Englewood Cliffs, NJ: Prentice-Hall, 1983.
37. G. H. John and P. Langley, “Estimating continuous distributions in bayesian classifiers,” in *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*, vol. 2, San Mateo, 1995, pp. 338–345.
38. J. Shawe-Taylor and N. Cristianini, *Support Vector Machines*, 2nd ed. Cambridge University Press, 2000.
39. S. Le Cessie and J. C. Van Houwelingen, “Ridge estimators in logistic regression,” *Applied Statistics*, vol. 41, no. 1, pp. 191–201, 1992.
40. Y. Jiang, J. Lin, B. Cukic, and T. Menzies, “Variance analysis in software fault prediction models,” in *Proceedings of the 20th IEEE International Symposium on Software Reliability Engineering*, Bangalore-Mysore, India, Nov. 16-19 2009, pp. 99–108.
41. 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*, vol. 34, no. 4, pp. 485–496, 2008.
42. T. Menzies, J. Greenwald, and A. Frank, “Data mining static code attributes to learn defect predictors,” *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2–13, 2007.