8-2011

Measuring Robustness of Feature Selection Techniques on Software Engineering Datasets

Huanjing Wang

Taghi Khoshgoftaar

Randall Wald

August, 2011

Measuring Robustness of Feature Selection Techniques on Software Engineering Datasets

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

Measuring Robustness of Feature Selection Techniques on Software Engineering Datasets

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

Randall Wald
Florida Atlantic University
Boca Raton, Florida 33431
rdwald@gmail.com

***Abstract*—Feature Selection is a process which identifies irrelevant and redundant features from a high-dimensional dataset (that is, a dataset with many features), and removes these before further analysis is performed. Recently, the robustness (e.g., stability) of feature selection techniques has been studied, to examine the sensitivity of these techniques to changes in their input data. In this study, we investigate the robustness of six commonly used 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 16 datasets from three real-world software projects. The experimental results demonstrate that Gain Ratio shows the least stability on average while two different versions of ReliefF show the most stability. Results also show that making smaller changes to the datasets has less impact on the stability of feature ranking techniques applied to those datasets.**

***Index Terms*—Robustness of feature selection, software metrics, software quality classification, fault-prone program module.**

I. Introduction

Software quality modeling is the process of using software metrics from previous iterations of development to locate potentially faulty modules in under-development code. This has become an important part of the software development process, allowing practitioners to focus development efforts where they are most needed, and avoid wasting time checking already-good code. Much research in recent years has examined this problem, using data mining and machine learning tools to build models to predict whether a given software module is fault-prone or not fault-prone [12], [24].

One difficulty encountered in software quality modeling is the problem of high dimensionality, where the number of available software metrics is too large for classifiers to work well. In these cases, many of the features may be redundant or irrelevant to defect prediction results. Relatively little research has focused on improving software quality models by using metric (feature) selection techniques for constructing defect prediction models. In those papers which do examine feature selection in the context of software quality modeling, feature selection techniques are evaluated according to their model performance; that is, models are built using only the selected features, and their performance is compared to that of models built using the full feature set [4], [8], [21], [23].

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 during the past. Few studies exist on the stability of feature selection algorithms. The purpose of investigating the robustness of feature selection techniques is to find which feature selection techniques are best at providing software practitioners a subset of software metrics which are relatively robust to variations of input data.

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 [16], [19]. 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. [22] 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. [19] 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.

The main contribution of the present work is that we consider 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.

In this study, we assess the stability performance of six commonly used feature selection techniques, including chi-squared (CS), information gain (IG), gain ratio (GR), two types of ReliefF (RF and RFW), and symmetrical uncertainty (SU). The assessment is based on the degree of agreement between a filter's output on both the original datasets and on modified datasets which have had some instances removed. The experimental results showed that GR and SU performed significantly worst among the six rankers and RF and RFW performed best. Also, the fewer instances that are deleted from (or equivalently, added to) a dataset, the more stable that feature ranking will be on that data.

The reminder of the paper is organized as follows: Section II presents six commonly used feature selection techniques. Section III describes the datasets and experimental design. Section IV presents the experimental results and analysis. Finally, we conclude the paper in Section V and provide suggestions for future work.

II. Filter-based Feature Ranking Techniques

The main goal of feature selection is to select a subset of features that excludes features which are irrelevant (not useful for predicting the class) or redundant (contain information already found in other features). 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.

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. Liu and Yu [18] 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 many application domains for many years, its application in the software quality and reliability engineering domain are limited. Chen et al. [4] have studied the applications of wrapper-based feature selection in the context of software cost/effort estimation. They conclude that the reduced dataset improved the estimation.

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. Feature ranking consists of scoring each feature according to a particular method, then selecting features based on their scores. This paper uses six commonly used filter-based feature ranking techniques including chi-squared [26], information gain [10], [26], gain ratio [26], two types of ReliefF [14], and symmetrical uncertainty [10], [26]. All of these feature selection methods are available within the WEKA machine learning software suite [26]. 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) [3] 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) [20] 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 based on attribute X, 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) [10] 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 techniques utilize the method of Fayyad and Irani [6] 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 which measures how much the feature's value changes when comparing an instance to its nearest same-class and different-class neighbors [13]. 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.

III. Experimental design

To test the stability of different feature selection techniques under different circumstances, we performed a case study on 16 different software metric datasets, using six 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.

A. Datasets

Experiments conducted in this study used software metrics and fault data collected from real-world software projects, including a very large telecommunications software system (denoted as LLTS) [7], the Eclipse project [27], and NASA software project KC1 [15]. These are all binary class datasets. That is, each instance is assigned one of two class labels: fault-prone (*fp*) and not fault-prone (*nfp*).

The software measurement data set of LLTS contains data from four consecutive releases, which are labeled as SP1, SP2, SP3, and SP4. This dataset includes 42 software metrics, including 24 product metrics, 14 process metrics, and four execution metrics [7]. 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 [27], 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 non-numeric 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 $t \ \epsilon \ \{10, 5, 3\}$ for release 2.0 and 3.0, while we use $t \ \epsilon \ \{5, 4, 2\}$ for release 2.1. These values are selected in order to have datasets with different levels of class imbalance. All nine derived datasets contain 208 independent attributes. Releases 2.0, 2.1, and 3.0 contain 377, 434, and 661 instances respectively.

The original NASA project, KC1 [15], includes 145 instances containing 94 independent attributes each. After removing 32 Halstead derived measures, we have 62 attributes left. 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 that 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.

The sixteen original datasets used in the work reflect software projects of different sizes with different proportions of *fp* and *nfp* modules. Table I lists the characteristics of the 16 datasets utilized in this work.

B. Dataset Perturbation (Changing)

For this study, we consider stability 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 c fraction 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.

TABLE I
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	Eclipse2.0-10	208	377	6.1%	93.9%
	Eclipse2.0-5	208	377	13.79%	86.21%
	Eclipse2.0-3	208	377	26.79%	73.21%
	Eclipse2.1-5	208	434	7.83%	92.17%
	Eclipse2.1-4	208	434	11.52%	88.48%
	Eclipse2.1-2	208	434	28.8%	71.2%
	Eclipse3.0-10	208	661	6.2%	93.8%
	Eclipse3.0-5	208	661	14.83%	85.17%
	Eclipse3.0-3	208	661	23.75%	76.25%
KC1	KC1-20	62	145	6.90%	93.10%
	KC1-10	62	145	14.48%	85.52%
	KC1-5	62	145	24.83%	75.17%

C. Feature Selection

Once the features are ranked according to their relevance to the class, the next step is to select a subset consisting of the most relevant ones. 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 [25].

D. Stability Measure

To assess robustness of feature selection techniques, past works have used different similarity measures, such as Hamming distance [5], correlation coefficient [11], consistency index [16], 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. 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 [16] is obtained as follows:

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

where d is the cardinality of the intersection between subsets T_i and T_j, and $-1 < I_C\left(T_i, T_j\right) \leq +1$. The greater the consistency index, the more similar the subsets are.

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:

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

This is the average of the consistency index 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, 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 $4 \times 9 = 36$).

IV. Results and Analysis

Experiments are conducted with six filter-based feature ranking techniques (CS, GR, IG, RF, RFW, and SU) 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 further aggregates across size of feature subset and choice of filter. Table II shows each dataset separately, but only presents results for one chosen degree of permutation and size of feature subset. Further analysis of the results across all the parameters was not possible due to space limitations.

A. Average Stability Performance

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

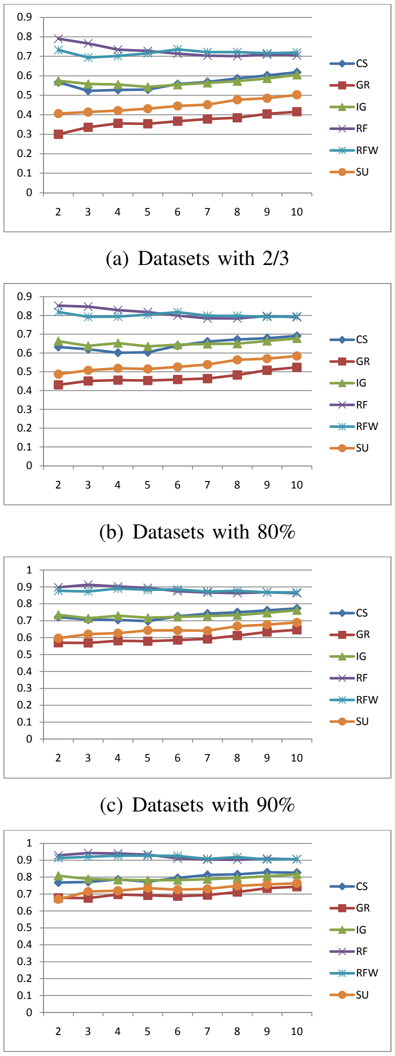
- Among the six filters, GR shows the least stability on average while RF and RFW show the most stability.
- The size of the feature subset can influence the stability of a feature ranking technique. For most rankers, stability is improved by an increased number of features in the selected subset.

B. 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.

C. Most and Least Stable Filters

Although our previous study showed that three features (software metrics) are sufficient for building software quality classification models [25], 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



(a) Datasets with 2/3

(b) Datasets with 80%

(c) Datasets with 90%

(d) Datasets with 95%

Fig. 1. KI Values

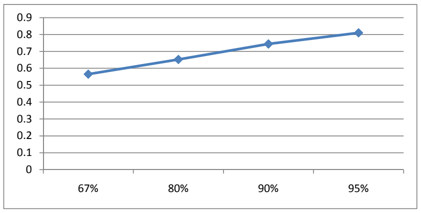


Fig. 2. Degree of Perturbation Impact on Stability

TABLE II
KI VALUES

(a) LLTS

	SP1	SP2	SP3	SP4	Average
CS	0.1588	0.7789	0.6259	0.6941	0.5644
GR	0.4477	0.7421	0.7417	0.4137	0.5863
IG	0.2523	0.9447	0.6615	0.4647	0.5808
RF	0.3458	0.6516	0.5118	0.6771	0.5466
RFW	0.6686	0.7026	0.4382	0.7239	0.6333
SU	0.1163	0.6941	0.2447	0.5101	0.3913

(b) Eclipse 2.0

	Eclipse2.0-10	Eclipse2.0-5	Eclipse2.0-3	Average
CS	0.2438	0.8526	0.7506	0.6157
GR	0.4392	0.8158	0.8397	0.6982
IG	0.3458	1.0000	0.7773	0.7077
RF	0.4647	0.7451	0.5947	0.6015
RFW	0.6771	0.7876	0.5395	0.6681
SU	0.3373	0.7876	0.4197	0.5149

(c) Eclipse 2.1

	Eclipse2.1-5	Eclipse2.1-4	Eclipse2.1-2	Average
CS	0.7621	0.4817	0.9171	0.7203
GR	0.5497	0.6176	0.8066	0.6580
IG	0.5667	0.4817	1.0000	0.6828
RF	0.7536	0.6516	0.8895	0.7649
RFW	0.7951	0.7621	0.8895	0.8156
SU	0.5813	0.4647	0.9065	0.6509

(d) Eclipse 3.0

	Eclipse3.0-10	Eclipse3.0-5	Eclipse3.0-3	Average
CS	0.8664	0.9065	0.5327	0.7685
GR	0.9466	0.6261	0.8131	0.7953
IG	0.8753	0.8131	0.6176	0.7687
RF	0.8158	0.7536	0.7961	0.7885
RFW	0.6316	0.8753	0.8386	0.7818
SU	0.5395	0.6704	0.6516	0.6205

(e) KC1

	KC1-5	KC1-10	KC1-20	Average
CS	0.9632	0.8842	0.9745	0.9406
GR	0.9171	1.0000	0.7451	0.8874
IG	1.0000	0.8931	0.8725	0.9219
RF	0.9660	0.9171	0.7706	0.8846
RFW	0.9575	0.8526	0.9198	0.9100
SU	0.9660	0.6224	0.7328	0.7737

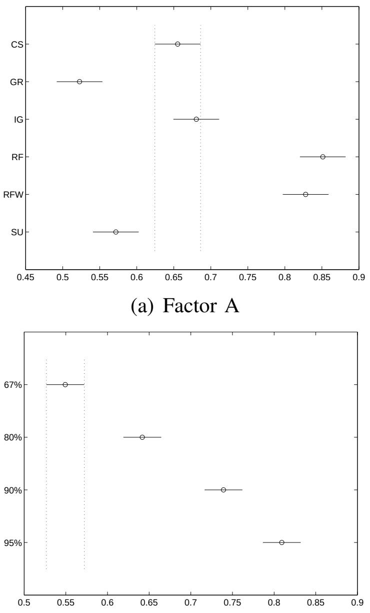
TABLE III
ANALYSIS OF VARIANCE

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	5.6412	5	1.12824	75.06	0
B	3.7092	3	1.23639	82.26	0
Error	5.6363	375	0.01503		
Total	14.9867	383			

present stability results only for size four feature subsets. Table II summarizes the results of robustness analysis across the different datasets, using modified datasets which contain 95% of the instances from their original dataset. In general, it can be observed that SU shows the least stability for 7 out of 16 and RFW shows the most stability for 6 out of 16. The inconsistent performance of GR appears to stem from its greater sensitivity to the specific choice of dataset.

D. ANOVA Analysis

We performed an ANalysis Of VAriance (ANOVA) F test [2] 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 ANOVA model we built includes two factors: Factor A, in which six 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 III. 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 six rankers and four different levels of deletion (c values). Additional multiple comparisons for the main factors were performed to investigate the differences among the respective groups (levels). All tests of statistical significance utilize a significance level α of 5%. Both ANOVA and multiple comparison tests were implemented in MATLAB.



(a) Factor A

(b) Factor B

Fig. 3. Multiple comparisons

The multiple comparison results are presented in Figure 3, displaying 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. The results show the following facts:

- For factor A, GR and SU performed significantly worst among the six rankers and RF and RFW performed best. CS and IG performed moderately.
- For Factor B, datasets with 95% of the instances from their original dataset performed significantly best, followed by 90%, 80%, and two thirds.

V. CONCLUSIONS AND FUTURE WORK

Feature selection is an important preprocessing step in data mining, and its output is used for further study. Typically, the robustness or stability of a feature selection algorithm is found by creating many reduced datasets (by deleting instances from the original dataset) and comparing these reduced datasets to each other. In this study, rather than comparing these reduced datasets to each other, each is compared solely to the original dataset which it came from. Those techniques whose outputs are insensitive to different perturbations in the input data are said to be robust, and they are preferred over those that produce inconsistent outputs. In this study we conducted robustness (or stability) analysis on six commonly used feature selection techniques. Sixteen datasets from three real-world software projects were used, with different levels of class imbalance. Experimental results demonstrate that GR and SU performed significantly worst among the six rankers and RF and RFW performed best. Results also showed that the number of instances deleted from the dataset affects 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 (or the smaller dataset), and thus the feature ranking performed on this dataset will be more stable.

Future work will involve conducting additional studies with data from other software projects. Additional experiments with other feature selection techniques can also be conducted.

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*, vol. 26, no. 3, pp. 392–398, February 2010.

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

[3] A. C. Cameron and P. K. Trivedi, *Regression Analysis of Count Data*. Cambridge University Press, 1998.

[4] Z. Chen, T. Menzies, D. Port, and D. Boehm, "Finding the right data for software cost modeling," *Software, IEEE*, vol. 22, no. 6, pp. 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," Department of Computer Science, Trinity College, Dublin, Ireland, Tech. Rep. TCD-CD-2002-28, 2002.

[6] 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.

[7] 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.

[8] 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.

[9] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 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*, vol. 15, no. 6, pp. 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*, vol. 12, no. 1, pp. 95–116, Dec. 2006.

[12] T. M. Khoshgoftaar and K. Gao, "A novel software metric selection technique using the area under roc curves," in *Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering*, 2010, pp. 203–208.

[13] 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.

[14] I. Kononenko, "Estimating attributes: Analysis and extensions of RELIEF," in *European Conference on Machine Learning*. Springer Verlag, 1994, pp. 171–182.

[15] 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.*, vol. 35, no. 2, pp. 293–304, 2009.

[16] 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.

[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] 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.

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

[20] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[21] 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.

[22] 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.

[23] S. Shivaji, E. J. W. Jr., R. Akella, and S. Kim, "Reducing features to improve bug prediction," in *ASE*. IEEE Computer Society, 2009, pp. 600–604.

[24] H. Wang, T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Mining data from multiple software development projects," in *Proceedings of 9th IEEE International Conference on Data Mining - Workshops*, Miami, FL, USA, December 6-9 2009, pp. 551–557.

[25] 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.

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

[27] 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: IEEE Computer Society, 2007, p. 76.