Software Metrics Reduction for Fault-proneness Prediction of Software Modules

Yunfeng Luo[1], Kerong Ben[1], Lei Mi[1]

[1] Department of Computer Engineering Navy University of Engineering
Wuhan, China
hehelyf@gmail.com, benkerong08@21cn.com, milei1981@126.com

Abstract. It would be valuable to use metrics to identify the fault-proneness of software modules. However, few research works are on how to select appropriate metrics for fault-proneness prediction currently. We conduct a large-scale comparative experiment of nine different software metrics reduction methods over eleven public-domain data sets from the NASA metrics data repository. The Naive Bayes data miner, with a log-filtering preprocessor on the numeric data, is utilized to construct the prediction model. Comparisons are based on the analysis of variance. Our conclusion is that, reduction methods of software metrics are important to build adaptable and robust software fault-proneness prediction models. Given our results on Naive Bayes and log-filtering, discrete wavelet transformation outperforms other reduction methods, and correlation-based feature selection with genetic search algorithm and information gain can also obtain better predicted performance.

Keywords: software fault-proneness; metrics reduction; analysis of variance

1 Introduction

Software fault static prediction technologies are defined as the methods which predict the amount or distribution of the software faults based on the software metrics [1]. In general, it aims at answering one or several of the following questions [2]: 1) Which metrics that are easy to be collected are good fault predictors? 2) Which models, quantitative, qualitative, hybrid, etc., should be used for fault prediction? 3) How accurate are those models? Software metric is a measure of some property of a piece of software or its specifications. It is very common nowadays for an organization to maintain several software metrics repositories for each undertaken project [3]. The metrics which are appropriate for predicting software fault-proneness have product and process categories [4]. The datasets comes from the NASA Metrics Data (MDP) repository involves more than 40 method-level metrics, and 8 class-level metrics [5]. Menzies proposed that mining static code attributes to learn fault predictors was useful, but he suggested not seeking “best” subsets of static code attributes, and building fault predictors using all available attributes metrics, followed by finding the most appropriate particular subset for a particular domain[6]. It is evident that the set of metrics should be reduced to the minimum set of representative parameters in order to avoid metrics which are not useful in the analysis or which introduce noise in the evaluation of data [7]. We organize a large-scale comparative experiment of nine different software metrics reduction methods over eleven public-domain data sets from the NASA MDP. We utilize Naive Bayes data miner with a log-filtering preprocessor on the numeric data to construct the prediction model [6]. Comparisons are based on the analysis of variance (ANOVA), which is a statistical technique of examining whether independent groups or populations are significantly different from one another. In our study, the one-way ANOVA is selected to analyze the performances of nine reduction methods of metrics.

2 Related Work

Since some independent variables might be highly correlated, i.e. multicollinearity, and serious multicollinearity problem will affect the stability of the results of regression analysis. Some methods are proposed to overcome the multicollinearity problem. Nagappan utilized principal component analysis (PCA) to reduce the software metrics [8]. Shatnawi analyzed the multicollinearity through three tests [9]: the Spearman correlation analysis, the Variance inflation factor (VIF) analysis, and the condition number analysis (use 30 as the cutoff value). The Spearman correlation in the metrics group was calculated first. After the high correlations were noted, they calculated the condition number and the VIFs for the group. If the condition number was greater than 30, the metrics that had the highest VIF value was dropped, and the condition number was recalculated; this process continued until the condition number was below 30; the remaining metrics were candidates to enter the MLR models. A correlation-based feature selection technique (CFS) was applied to down-select the best predictors out of the 21 independent variables in the datasets [10]. This involves searching through all possible combinations of variables in the dataset to find which subset of variables works best for prediction.

Olague utilized simple correlation calculation to ascertain whether the currently available metrics can predict the initial quality of a software system [11]. Zhou utilized univariate regression analysis to examine the effect of each metric separately, identifying the metrics which are significantly related to fault-proneness of classes and identifying potential fault-proneness predictors to be used in multivariate analysis [12]. Vandecruys went through an input selection procedure using a χ^2 -based filter [13]. First, the observed frequencies of all possible combinations of values for class and variable are measured. Based on this, the theoretical frequencies, assuming complete independence between the variable and the class, are calculated. The hypothesis of equal odds provides a χ^2 test statistic; higher values allow one to reject the null hypothesis of equal odds more confidently; hence, these values allow one to rank the variables according to predictive power.

Menzies ranked the attributes using the information gain [6]. If A is a metric and C is the class, $n(c)$ is the examples amount of every class, $N = \sum_{c \in C} n(c)$, $p(c)=n(c)/N$, and $p(c/a)$ is the probability of the metric a belongs to class c. Eq.1 gives the entropy of the class before and after observing the metric.

$$H(C) = -\sum_{c \in C} p(c)\log_2 p(c)$$
$$H(C \mid A) = -\sum_{a \in A} p(a) \sum_{c \in C} p(c \mid a)\log_2 p(c \mid a) \qquad (1)$$

Each metric A_i is assigned a score based on the information gain between itself and the class:

$$InfoGain(A_i) = H(C) - H(C \mid A_i) \qquad (2)$$

3 Proposed Methods

In this section, we introduce the data reduction techniques firstly, which include nine data reduction methods. The Naive Bayes data miner, with a log-filtering preprocessor on the numeric data, is utilized to construct the prediction model. Finally, we describe our model evaluation criteria and the analysis of variance.

3.1 Data Reduction Techniques

Feature extraction and feature selection are two different approaches for the reduction of dimensionality. Feature extraction involves linear or nonlinear transformation from the original feature space to a new one of lower dimensionality. Although it does reduce the dimensionality of the vectors fed to the classifier, the number of features that must be measured remains the same. Feature selection, on the other hand, directly reduces the number of original features by selecting a subset of them that still retains sufficient information for classification. In general, feature selection approaches can be grouped into two categories: filter methods and wrapper methods [14]. Acquiring no feedback from classifiers, the filter methods estimate the classification performance by some indirect assessments, such as distance measures which reflect how well the classes separate from each other. The wrapper methods, on the contrary, are classifier-dependent. Based on the classification accuracy, the methods evaluate the “goodness” of the selected feature subset directly, which should intuitively yield better performance. In spite of the good performance, the wrapper methods have limited applications due to the high computational complexity involved. Filters and wrappers hava the similar search approach. One popular filter metrics for classification problems are correlation. We choose CFS as our filter method. For wrapper method, we choose J48 as the classifier. Two search approaches, best first and genetic algorithm, are choosed for subset selection.

For feature extraction, we focus on two popular and effective methods of lossy dimensionality reduction [15]: principal component analysis (PCA) and discrete wavelet transformation (DWT). PCA searches for k n-dimensional orthogonal vectors that can best be used to represent the data, where $k \le n$. The original data are thus projected onto a much smaller space, resulting in dimensionality reduction. Unlike feature subset selection, which reduces the metric set size by retaining a subset of the initial set of metrics, PCA combines the essence of metrics by creating an alternative, smaller set of variables. Then the initial data can be projected onto this smaller set.

PCA often reveals relationships that are not previously suspected and thereby allows interpretations that would not ordinarily result. DWT is a linear signal processing technique that, that is applied to transform a data vector X to a numerically different vector X' of wavelet coefficients. The two vectors are of the same length. The usefulness lies in the fact that the wavelet transformed data can be truncated. A compressed approximation of the data can be retained by storing only a small fraction of the strongest of the wavelet coefficients.

A particular reduction method evaluated features based on information gain, which evaluates the worth of a metric by measuring the information gain with respect to the class. Menzies ranked the attributes using the information gain [6]. In our study, we got a metrics queue based the value of information gain, and choose top 3, 4 and 5 metrics respectively to enter the prediction model.

The No-Reduction method, which does not reduce metrics and allows all metrics to enter the prediction model, is used as a baseline method to compare against the investigated metrics reduction methods. With these comparisons, we can determine whether some (or even all) of the metrics reduction methods are better than No-Reduction method in terms of their predictive accuracy on faults or vice versa.

These reduction methods are summarized in Table 1. These methods are implemented in the WEKA data mining tool [16].

Table 1. Software metrics reduction methods

No.	*Reduction methods*	*Description*
1	CfsSubsetEval+ BestFirst	Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. The search approach is greedy hill-climbing augmented with a backtracking facility.
2	CfsSubsetEval+ GeneticSearch	Performs a search using the simple genetic algorithm described in Goldberg
3	InfoGainAttributeEval (top 3)	Evaluates the worth of an attribute by measuring the information gain with respect to the class. Choose top 3 features in the features rank.
4	InfoGainAttributeEval (top 4)	Choose top 4 features in the features rank.
5	InfoGainAttributeEval (top 5)	Choose top 5 features in the features rank.
6	WrapperSubsetEval+J48+Best First	Evaluates attribute sets by using a learning scheme. Cross validation is used to estimate the accuracy of the learning scheme (we choose J48) for a set of attributes. The search method is BestFirst.
7	WrapperSubsetEval+J48+ GeneticSearch	The search method is GeneticSearch
8	DWT	A filter for wavelet transformation.
9	PCA	Performs a principal components analysis and transformation of the data. Dimensionality reduction is accomplished by choosing enough eigenvectors to account for some percentage of the variance in the original data -- default 0.95 (95%).
10	No-Reduction	Allow all metrics to enter the prediction model, which is used as a baseline method.

3.2 Classification Technique and Evaluation Strategy

We use Naïve Bayes classifiers as our prediction model to evaluate the metrics reduction methods [6]. Naïve Bayes classifier is based on Bayes' Theorem. Formally,

$$P(H \mid E) = \frac{P(H)}{P(E)} \prod_i P(E_i \mid H) \qquad (3)$$

that is, given fragments of evidence E_i and a prior probability for a class *P(H)*, the posteriori probability *P(H|E)* is calculated. The predicted variable is whether one or more faults exist in the given module.

We use the following set of evaluation measures. The Probability of Detection (*pd*) is defined as the probability of the correct classification of a module that contains a fault: (Note that *pd* is also called recall). The Probability of False alarm (*pf*) is defined as the ratio of false positives to all non-defect modules.For convenience, we say that *notpf* is the complement of *pf*: *notpf*=1-*pf*. In practice, engineers balance between *pf* and *pd*. To operationalize this notion of balance, *bal* is defined to be the Euclidean distance from the sweet spot *pf*=0, *pd*=1 to a pair of (*pf*, *pd*).

$$balance = bal = 1 - \frac{\sqrt{(0-pf)^2 + (1-pd)^2}}{\sqrt{2}} \qquad (4)$$

Hence, better and higher balances fall closer to the desired sweet spot of *pf*=0, *pd*=1.

3.3 Analysis of variance

In our study, the one-way ANOVA design is selected to analyze the performances of the nine metrics reduction methods. In this design, a reduction method corresponds to a group.

Let $Y_{1j}, \cdots, Y_{n_j j}$ represent a random sample of n_j observations taken from the population of group *j*. In our experiment, ten different data mining algorithms are used. In addition, eleven datasets are available and the dependent variable is known for all the datasets. Y_{ij}, the *i*th observation in group *j* (where *i* = 1,2,⋯,11 and *j* = 1,2,⋯,10), can be represented by the following model

$$Y_{ij} = \mu + \delta_j + \varepsilon_{ij} \qquad (5)$$

where μ is the overall effect to all the observations; $\delta_j = \mu_j - \mu$ is the treatment effect related to the *j*th group; $\varepsilon_{ij} = Y_{ij} - \mu_j$ is the experimental error associated with the *i*th observation in group j; μ_j is the true mean of the *j*th group. The F statistic validates whether the 10 population means are equal. In one way ANOVA, between group variance is denoted by S_b^2, and with group variance is denoted by S_w^2, the following equation shows how the F-test is calculated in our experiment.

$$F = \frac{S_b^2 / 10}{S_w^2 // \sum_{j=1}^{10} (n_j - 1)} \qquad (6)$$

4 Empirical Evaluation

4.1 Data

Eleven data sets are available in MDP, as shown in Table 2. Each module of each datasets describes the metrics of corresponding module, plus the number of defects known for that module. The error count column was converted into a Boolean metrics called *fault- proneness*? as follows:

$$Fault\text{ - }proneness? = (error\ count \geq 1)$$

Finally, the error density column is removed (since it can be derived from line counts and error count). The preprocessed data sets have two categories: 38 attributes and 22 attributes, plus one target attribute (fault-proneness?), and include Halstead, McCabe, lines of code, and other miscellaneous metrics [6]. The difference of two categories is that one includes the miscellaneous metrics and the other doesn't.

Menzies found that these values in the data sets formed an exponential distribution with many small values and a few much larger values. So they suggested that a logarithmic filter on all numeric values might improve predictor performance [17]. Such a filter replaces all numerics *n* with their logarithms, *Ln(n)*. In our experiment, we applied a logarithmic filter on all numeric values, and added all numbers in the datasets with 0.00001 to avoid numerical errors with ln(0).

Table 2. Datasets used in this study

Data set	*language*	*Total LOC*	*#modules*	*%fault-proneness*
CM1	C	20K	505	16.04
JM1	C	315K	10878	19.32
KC1	C++	43 K	2107	13.91
KC3	java	18K	458	6.33
KC4	Perl	25 K	125	48
MC1	C&C++	63K	9466	0.64
MW1	C	8K	403	6.70
PC1	C	40K	1107	6.59
PC2	C	26K	5589	0.39
PC3	C	40K	1563	10.24
PC4	C	36K	1458	12.21

4.2 Experimental Result

We applied nine different software metrics reduction methods shown in Table 1 in addition to No-Reduction, over eleven public-domain data sets in Table 2. The metrics subsets, chosen by reduction method 6 over datasets KC3, PC2 and PC3, and chosen by reduction method 7 over dataset PC2 respectively, are null sets. Reduction method 9 classified all modules in MC1 as no fault-proneness. Therefore, the values of *pd*, *notpf* and *bal* calculated by these reduction methods over corresponding datasets are null. The detailed predicted results are shown in Table 3 - Table 5.

Table 3. The predicted results for *pd*

Reduction Methods	*CM1*	*JM1*	*KC1*	*KC3*	*KC4*	*MC1*	*MW1*	*PC1*	*PC2*	*PC3*	*PC4*
1	0.679	0.419	0.768	0.552	0.833	0.295	0.630	0.685	0.318	0.819	0.860
2	0.728	0.423	0.768	0.552	0.783	0.328	0.667	0.822	0.682	0.869	0.899
3	0.210	0.199	0.741	0.448	0.817		0.593	0.342	0.773	0.775	0.978
4	0.370	0.222	0.853	0.483	0.867	0.033	0.593	0.767	0.773	0.863	0.972
5	0.469	0.221	0.860	0.586	0.833	0.426	0.704	0.849	0.773	0.856	0.916
6	0.222	0.346	0.379		0.867	0.262	0.222	0.616			0.910
7	0.457	0.330	0.696	0.862	0.700	0.459	0.704	0.753		0.844	0.848
8	0.617	0.655	0.846	0.828	0.783	0.902	0.704	0.712	0.727	0.813	0.876
9	0.222	0.254	0.556	0.517	0.583		0.037	0.342	0.318	0.556	0.826
10	0.679	0.475	0.846	0.862	0.700	0.754	0.704	0.822	0.727	0.881	0.910

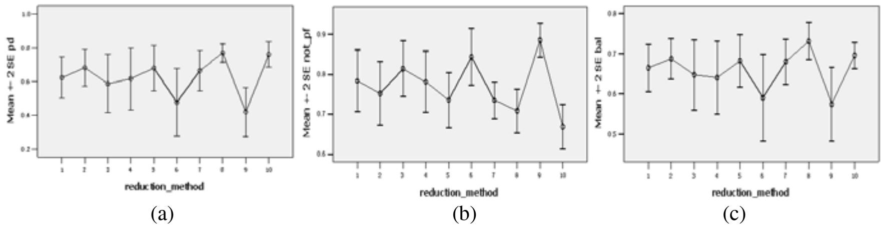
Table 4. The predicted results for *notpf*

Reduction Methods	*CM1*	*JM1*	*KC1*	*KC3*	*KC4*	*MC1*	*MW1*	*PC1*	*PC2*	*PC3*	*PC4*
1	0.613	0.831	0.641	0.928	0.692	0.950	0.803	0.788	0.969	0.623	0.784
2	0.590	0.829	0.641	0.907	0.723	0.939	0.803	0.638	0.884	0.556	0.765
3	0.950	0.929	0.698	0.862	0.677		0.875	0.900	0.866	0.678	0.709
4	0.894	0.884	0.596	0.823	0.631	0.979	0.840	0.680	0.880	0.588	0.709
5	0.840	0.872	0.573	0.797	0.692	0.843	0.771	0.589	0.878	0.628	0.747
6	0.913	0.865	0.871		0.754	0.942	0.963	0.675			0.772
7	0.757	0.842	0.716	0.655	0.800	0.781	0.750	0.676		0.602	0.773
8	0.604	0.617	0.598	0.690	0.738	0.897	0.761	0.696	0.809	0.670	0.710
9	0.906	0.919	0.809	0.893	0.862		0.981	0.907	0.975	0.795	0.813
10	0.550	0.764	0.592	0.643	0.800	0.675	0.734	0.582	0.786	0.547	0.675

Table 5. The predicted results for *bal*

Reduction Methods	*CM1*	*JM1*	*KC1*	*KC3*	*KC4*	*MC1*	*MW1*	*PC1*	*PC2*	*PC3*	*PC4*
1	0.645	0.572	0.698	0.679	0.753	0.500	0.703	0.732	0.517	0.704	0.818
2	0.652	0.574	0.698	0.676	0.751	0.523	0.726	0.715	0.761	0.673	0.819
3	0.440	0.432	0.719	0.598	0.737		0.699	0.530	0.813	0.722	0.793
4	0.549	0.444	0.696	0.613	0.722	0.316	0.691	0.720	0.818	0.693	0.793
5	0.608	0.442	0.682	0.674	0.753	0.579	0.735	0.690	0.818	0.718	0.811
6	0.447	0.528	0.552		0.802	0.477	0.449	0.645			0.827
7	0.579	0.513	0.706	0.737	0.745	0.587	0.726	0.712		0.698	0.807
8	0.610	0.636	0.696	0.749	0.760	0.899	0.731	0.704	0.765	0.732	0.777
9	0.446	0.469	0.658	0.650	0.690	0.293	0.319	0.530	0.518	0.654	0.819
10	0.609	0.593	0.692	0.730	0.745	0.712	0.718	0.679	0.755	0.669	0.762

Based on data from Table 3 - Table 5, we use the one-way ANOVA design to analyze the performances of the nine metrics reduction methods Fig. 1 shows the values of the mean and standard error of mean for *pd*, *notpf* and *bal*. We observe that DWT (method 8) performs the best on mean for *pd*, PCA (method 9) performs the best for *notpf*, and DWT performs the best for *bal*. It is also worth noting that the means of *pd*, *notpf* and *bal* vary according to the reduction methods, so it is for the standard error.



(a) (b) (c)

Fig. 1. The error chart for (a) *pd*, (b) *Notpf* , (c) *bal*

Our null hypothesis is that there is no significance difference for performances of the ten metrics reduction methods. The F-value is selected at 90% confidence level (i.e., α = 0.10) [3]. The p-value related to the F-test is also provided. Table 6 - Table 8 present the results of ANOVA. Since the p-value is well below the significance level α, we can reject the null hypothesis and conclude that there is a significant difference in the means of the 10 metrics reduction methods.

Table 6. ANOVA for *pd*

	Sum of Squares	*df*	*Mean Square*	*F*	*p-value*
Between Groups	1.111	9	0.123	2.615	0.010
Within Groups	4.437	94	0.047		
Total	5.548	103			

Table 7. ANOVA for *notpf*

	Sum of Squares	*df*	*Mean Square*	*F*	*p-value*
Between Groups	0.386	9	0.043	3.701	0.001
Within Groups	1.088	94	0.012		
Total	1.473	103			

Table 8. ANOVA for *bal*

	Sum of Squares	*df*	*Mean Square*	*F*	*p-value*
Between Groups	0.206	9	0.023	1.783	0.082
Within Groups	1.208	94	0.013		
Total	1.415	103			

Table 9 presents the description of *pd*, *notpf* and *bal* in detail. The results indicate that the all metrics reduction methods improve *notpf*, and most methods improve the *bal* compared with NoRedcution. But at the same time, the *pd* decreases with the increase of *notpf.* The only exception is DWT. The means of *pd* and *notpf* for DWT both increase compared with NoRedcution. It is also worth noting that the *pf* for CfsSubsetEval+ GeneticSearch (method 2) and InfoGainAttributeEval+top5 (method 5) have a minimum decrease among the other 8 reduction methods compared with NoRedcution. We conclude that the metrics reduction methods could improve the predicted performance on average, the DWT outperforms the other reduction methods, and CFS with genetic search algorithm and information gain also could obtain better predicted performance.

Table 9. The data description for *pd*, *notpf* and *bal*

Reduction Methods	*N*	*pd*		*not_pf*		*Bal*	
		Mean	*Std. Deviation*	*Mean*	*Std. Deviation*	*Mean*	*Std. Deviation*
1	11	0.623	0.203	0.784	0.130	0.666	0.099
2	11	0.684	0.182	0.752	0.133	0.688	0.084
3	10	0.588	0.273	0.815	0.110	0.648	0.140
4	11	0.618	0.305	0.781	0.133	0.641	0.151
5	11	0.681	0.226	0.736	0.110	0.683	0.109
6	8	0.478	0.283	0.844	0.101	0.591	0.152
7	10	0.665	0.187	0.735	0.073	0.681	0.091
8	11	0.769	0.093	0.708	0.091	0.733	0.076
9	10	0.421	0.228	0.886	0.066	0.575	0.145
10	11	0.760	0.124	0.668	0.094	0.697	0.056
Total	104	0.635	0.232	0.768	0.120	0.663	0.117

5 Conclusion and Future Work

In this paper, we have conducted a large-scale comparative experiment of nine different software metrics reduction methods over eleven public-domain data sets from the NASA Metrics Data (MDP) repository. It is shown that, on average, there is a significant difference in the predicted performance of the different metrics reduction methods. We also demonstrate that the metrics reduction methods can improve the predicted performance, the DWT outperforms the other reduction methods on average, and CFS with genetic search algorithm and information gain can also obtain better predicted performance. These two conclusions are critical for complex data mining problems such as software fault-proneness classification. The practitioner should not solely rely on sophisticated and/or robust algorithms to generate accurate predictions. As demonstrated in our study, it is advised to use some feature reduction methods to reduce software metrics in order to improve the predicted performance.

A source of bias in this study is the set of data reduction methods explored by this study. Data mining is a large and active field and any single study can only use a small subset of the known data mining algorithms. Therefore, we only suggest that DWT does best in the chosen nine typical reduction methods. Future works can investigate whether the classifier would affect the performance of reduction methods. In addition, it would be worthwhile to investigate whether the metrics reduction method is effective for the other metrics, such as object-oriented metrics and software process metrics.

Acknowledgments. This research is supported by the national prior research foundation of China under Grant No.513270104.

References

1. Wang, Q., Wu, S.J., Li, M.S.: Software Defect Prediction. Journal of Software 19, 1565--1580 (2008) (in Chinese)
2. Raimund, M., Witold, P., Giancarlo, S.: A Comparative Analysis of the Efficiency of Change Metrics and Static Code Attributes for Defect Prediction. In: 30th International Conference on Software Engineering, pp.181--190. ACM, New York (2008)
3. Khoshgoftaar, T.M., Rebours, P., Seliya, N.: Software Quality Analysis by Combining Multiple Projects and Learners. Software Quality Journal 17, 25--49 (2009)
4. Catal, C., Diri, B.: A Systematic Review of Software Fault Prediction Studies. Expert Systems with Applications 36, 346--7354 (2009)
5. NASA Metrics Data (MDP) Repository, http://mdp.ivv.nasa.gov
6. Menzies, T., Greenwald, J., Frank, A.: Data Mining Static Code Attributes to Learn Defect Predictors. IEEE Transactions on Software Engineering 33, 2--13 (2007)
7. Bellini, P., Bruno, I., Nesi, P., Rogai, D.: Comparing Fault-proneness Estimation Models. In: 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2005), pp. 205--214. IEEE Press, New York (2005)
8. Nagappan, N., Ball, T., Zeller, A.: Mining Metrics to Predict Component Failures. In: 28th International Conference on Software Engineering, pp.119--125. ACM, New York (2006)
9. Shatnawi, R., Li, W.: The Effectiveness of Software Metrics in Identifying Error-prone Classes in Post-release Software Evolution Process. Journal of Systems and Software 81, 1868--1882 (2008)
10. Elish, K.O., Elish, M.O.: Predicting Defect-prone Software Modules Using Support Vector Machines. Journal of Systems and Software 81, 649--660 (2008)
11. Olague, H.M., Etzkorn, L.H., Messimer, S.L.: An Empirical Validation of Object-oriented Class Complexity Metrics and Their Ability to Predict Error-prone Classes in Highly Iterative, or Agile Software: a Case Study. Journal of Software Maintenance and Evolution: Research and Practice 20, 171--197 (2008)
12. Yuming, Z., Hareton, L.: Empirical Analysis of Object-oriented Design Metrics for Predicting High and Low Severity Faults. IEEE Transactions on Software Engineering 32, 771--789 (2006)
13. Vandecruys, O., Martens, D., Baesens, B.: Mining Software Repositories for Comprehensible Software Fault Prediction models. Journal of Systems and Software 81, 823--839 (2008)
14. Kohavi, R. John, G.H.: Wrappers for Feature Subset Selection. Artificial Intelligence 97, 273--324 (1997)
15. Han, J., Kamber, M.: Data Mining: Concepts and Techniques (2nd). Morgan Kaufmann, San Francisco (2006)
16. Witten, I.H., Frank, E.: Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (2000)
17. Menzies, T., DiStefeno, J.S., Chapman, M.: Metrics that Matter. In: 27th Annual NASA Goddard Software Engineering Workshop (SEW 2002), pp. 51--57. IEEE Computer Society, Washington (2002)