Computer Languages, Systems & Structures 47 (2017) 211–231

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

journal homepage: www.elsevier.com/locate/cl

Empirical analysis of search based algorithms to identify change prone classes of open source software

Ankita Bansal

Netaji Subhas Institute of Technology, Dwarka, India

ARTICLE INFO

Article history:
Received 4 December 2015
Received in revised form
13 October 2016
Accepted 21 October 2016
Available online 1 November 2016

Keywords:
Change proneness
Metrics
Object oriented paradigm
Search based algorithms
Software quality
Empirical validation

ABSTRACT

There are numerous reasons leading to change in software such as changing requirements, changing technology, increasing customer demands, fixing of defects etc. Thus, identifying and analyzing the change-prone classes of the software during software evolution is gaining wide importance in the field of software engineering. This would help software developers to judiciously allocate the resources used for testing and maintenance. Software metrics can be used for constructing various classification models which can be used for timely identification of change prone classes. Search based algorithms which form a subset of machine learning algorithms can be utilized for constructing prediction models to identify change prone classes of software. Search based algorithms use a fitness function to find the best optimal solution among all the possible solutions. In this work, we analyze the effectiveness of hybridized search based algorithms for change prediction. In other words, the aim of this work is to find whether search based algorithms are capable for accurate model construction to predict change prone classes. We have also constructed models using machine learning techniques and compared the performance of these models with the models constructed using Search Based Algorithms. The validation is carried out on two open source Apache projects, Rave and Commons Math. The results prove the effectiveness of hybridized search based algorithms in predicting change prone classes of software. Thus, they can be utilized by the software developers to produce an efficient and better developed software.

1. Introduction

Prediction of various quality attributes during the early life cycle phases is a promising approach for quality improvement. Some of the attributes such as maintainability, defect and change proneness can be predicted with the help of various software metrics [1–8] proposed in literature. Prediction of change prone classes leads to a higher quality software product. In today's software industries, changes in software are inevitable. While these changes may be beneficial for the end users, at the same time, they may also introduce some risks in the software. For example, the changes may lead to addition of some new functionality, fixing of defects etc. which will result in a higher quality product/ software and thus benefit the end users. But at the same time, incorporating such changes in the software may introduce some undesirable effects such as injecting new defects etc. leading to decreased performance of the software. Among all the phases in the software development life cycle, changes that the customer demands after the acceptance of the product are incorporated in the software

E-mail address: ankita.bansal06@gmail.com

http://dx.doi.org/10.1016/j.cl.2016.10.001
1477-8424/

during the maintenance phase. Thus, lots of resources (time, money and manpower) and rework is required during the maintenance phase. If the practitioners are aware of the classes which are more prone to changes, they can focus their resources of such classes. Thus, a timely identification of the classes which are more prone to changes facilitates an efficient allocation of resources and brings some insights on the design of the software. Moreover, correct prediction of changes also brings some insight on the design of the software. For example, if changes in one module are due to addition of new features or requirements, then if these changes are affecting another class to a large extent, this implies that there is high coupling between the classes. In other words, if change (can be due to any reason) in one module have a considerable effect on other parts of the system, then the coupling between the modules may need to be reduced. Coupling can be reduced with the help of coupling metrics used in the system.

Despite the significant advantages of identifying change prone classes, there are only few studies in literature which have established relationship between various Object Oriented (OO) metrics and change proneness [9–13]. The studies [14,15] have developed machine learning (ML) and statistical models for predicting change prone classes. In addition to these techniques, Search Based Algorithms (SBAs) which form the subset of ML techniques can also be used for model construction. SBAs make use of a fitness function to find an optimal or near-optimal or good-enough solution among all the possible solutions. These are basically meta-heuristic techniques which may use various operators like reproduction, mutation, recombination, and selection to find the best set of candidate solutions. Thus, the process to obtain candidate solutions continues until optimal or near optimal solutions is obtained. Near optimal solutions may refer to the solutions obtained when number of iterations or generations set for each algorithm is over. Moreover, these operators help to improve the current solution by optimizing the fitness function [16]. In addition to this, SBAs are robust, flexible and can work well on the projects with varying characteristics. Also, SBAs can handle imbalanced, noisy and inaccurate data. Due to these numerous advantages of SBAs, they are used in various fields such as bioinformatics, genetics etc. In the field of software engineering, they are used in the area of software testing [17,18]. But to the best of the authors' knowledge, they are not much explored for constructing various change prediction models. SBAs are highly suitable for model construction and should be widely used because of the following reasons: (1) Usually, the constructed change prediction models are applied to different projects having different characteristics. (2) The datasets used in the projects obtained from open source software repositories are imbalanced and contain noisy data. (3) Due to their self-optimizing nature, SBAs may produce good classification rules for identifying change prone classes. Thus, in this work, the primary objective is to analyze the effectiveness and performance of various hybridized SBAs to construct prediction models which can be used to identify change prone classes. Hybridized SBAs posses the characteristics and advantages of both the SBAs as well as non-SBAs. To analyze the effectiveness, we have also constructed ML models for the same purpose and compared the performance of ML models with the models constructed using SBAs.

In this work, we address the following Research Questions (RQ):

- RQ1: Are OO metrics related to change proneness?
- RQ2: What is the predictive performance of SBA's for predicting change prone classes?
- RQ3: Among multiple SBAs used in this study, which algorithm can be used for predicting change prone classes?
- RQ4: Is the performance of SBAs better than ML techniques for predicting change-prone classes?
- RQ5: Are the performance results consistent with different evaluation measures, viz. *g*-mean and accuracy?
- RQ6: What is the computational time (CPU time) taken by change prediction models developed using different SBAs and ML techniques? What is the trade-off between the CPU time and predictive performance of different SBAs and ML techniques?

We have considered the use of various OO metrics to construct different classification models for predicting change prone classes. For the purpose of validation, two open source Apache projects, Rave and Commons Math developed using Java language are analyzed. Classification models are constructed using a number of hybridized SBAs. The use of such large number of algorithms lead to fair evaluation of all the algorithms and allow to judge the superiority of one algorithm over the other. The performance of the models is measured using two evaluation parameters i.e. *g*-mean and accuracy (explained in detail in the later section).

This paper is organized as follows. Section 2 summarizes the related work. Section 3 focuses on the variables and dataset used in this study. Various data analysis methods are discussed in Section 4. Section 5 describes the performance measures and results of correlation analysis. The results obtained using 10-cross validation is summarized in Section 6. Section 7 discusses and summaries the answers to various research questions stated in Section 1. Section 8 explains the application of the work. Section 9 summarizes threats to validity in this work. Finally, the work is concluded in Section 10.

2. Related work

Empirical research involving the development of software quality prediction models using various ML and statistical techniques has been conducted to a large extent. There are various structural measures or metrics which are found to be significant predictors of quality attributes. Hence, such metrics can be extensively used to construct models which can be used to improve quality.

The empirical study by Lindvall studied the effect of size of a class on the probability of change in that class [19]. Koru and Tian [12,13] worked on Pareto's law, 80:20 principle which states that large volume (around 80%) of the problems are generally located in a small proportion (around 20%) of the modules. They proposed a method of ranking to find top-change modules and those having top measurement values. They also used a voting mechanism to identify the modules with the largest size, highest coupling, highest cohesion and largest inheritance. The authors concluded that when ranking or voting mechanism is used, top change modules were not the top-measurement modules. Han et al. developed design models for predicting change proneness [10]. They calculated Behavioural Dependency Measure (BDM) for predicting change proneness and the results concluded that BDM is as an effective indicator for change proneness prediction. Along with BDM, the effect of polymorphism and inheritance relationships has also been taken into account by their another study to predict change proneness [11]. The results concluded that when the system uses high amount of inheritance and polymorphism, then BDM is an important indicator of change proneness. Zhou et al. examined whether there is any confounding effect of class size on the relationship between OO metrics and change proneness [14]. The authors concluded that the size has a confounding effect and thus, should be taken into account to correctly analyze the results. The analysis of the confounding effect of size was based on linear regression equations. A recent study by Lu et al. found significant relationship between OO metrics and change proneness [15]. The study considered a large number of OO metrics (62) covering four dimensions (size, cohesion, coupling and inheritance) and used 102 Java systems to analyze the change-proneness predictive ability of these OO metrics. To compute the average change proneness predictive ability of OO metrics over all the systems, random effect meta - analysis models were used. The results concluded that the size metrics have moderate ability to identify change and non- change prone classes, coupling and cohesion metrics have lower predictive ability than size metrics and inheritance metrics have poor predictive ability. Elish and Al-Khiaty [20] recommended a combination of evolution-based metrics and Chidamber and Kemerer metrics [5] for determining change-prone classes. The study by Malhotra and Khanna [21] determined the effectiveness of ML techniques for predicting change prone classes and compared their performance with the statistical technique. The authors concluded that the performance of ML techniques is comparable to the statistical technique.

Apart from the ML and statistical techniques used for construction of prediction models, some studies have used various SBAs and soft computing techniques for constructing models used for fault [22–27] and effort [28–31] prediction. Malhotra and Khanna [32] have investigated the use of various SBAs for change proneness prediction. They also compared the performance of SBAs with ML and statistical methods. According to Harman and Jones [18], SBAs should be widely used for the problems of software engineering as SBAs (or meta-heuristic algorithms) can handle inconsistent, imbalanced and noisy data. In addition, they produce accurate rules for computing the solution.

Thus, SBAs have been used for fault and effort prediction, but have been not much used for change prediction. Thus, our main contribution of this study is to analyze and assess the use of SBAs to identify change prone classes. In other words, this study analyzes the applicability and suitability of SBAs for the prediction of change prone classes. Besides this, Since ML techniques have been used for change prediction in literature, we have also compared the performance of SBAs with ML techniques. The better performance of SBAs over ML techniques or even comparable performance will prove that SBAs are effective in predicting change prone classes. In this study, we have evaluated the performance of various SBAs for change proneness prediction of two open source Apache projects. The best SBA which outperforms the other SBAs is determined which can be used by researchers and practitioners for predicting change prone classes of a similar projects.

3. Research background

In this section, we give an overview of the variables (independent and dependent) used in this study. The detailed explanation of the change collection process is also provided. This is followed by descriptive statistics of all the metrics of both the projects.

Table 1
Independent variables used.

S.no	Metrics	Definition	Dimension
1.	LCOM	Measure of dissimilarity of methods in a class.	Cohesion
2.	CBO	Counts the number of classes which are coupled with other classes and vica-versa.	Coupling
3.	RFC	Counts the number of internal methods (the methods in the class) and the external methods (the methods invoked by the internal methods).	Coupling
4.	DIT	Counts the number of ancestor classes (the number of classes from the particular node to the root node).	Inheritance
5.	NOC	Counts the number of descendants in a hierarchy.	Inheritance
6.	WMC	Sum of complexities of every method in a class.	Size
7.	SLOC	Counts the number of lines of code that contain only the source code.	Size

3.1. The variables

We have used the famous OO metrics proposed by Chidamber and Kemerer (CK) [5] as independent variables: Coupling Between Objects (CBO), Number Of Children (NOC), Response For a Class (RFC), Depth of Inheritance Tree (DIT), Lack of Cohesion Among Methods (LCOM), and Weighted Methods of a class (WMC). Along with the CK metrics, one additional metrics is used to measure the size of software, Source Lines of Code (SLOC). These metrics cover various dimensions of OO paradigm such as coupling, cohesion, inheritance and size. Coupling shows the degree of interdependence between the classes. Cohesion on the other hand, measures the interdependence between the variables and methods within a class. One of the most important goals of OO design is to have high cohesion and loose coupling between the classes. Inheritance allows us to define a class in terms of another class, i.e. inheritance metrics deal with the information about ancestors and descendants of a class [33]. We have computed these metrics with the help of an open source CKJM tool (http://gromit.iiar.pwr.wroc.pl/pinf/ckjm/metric.html). The brief description of each metric along with the dimension they cover is listed in Table 1.

The dependent variable used is a binary variable, change proneness. If a class is changed in the next or future release, it is said to be change prone and not change prone otherwise. Thus, we have compared the lines of code of common classes between the consecutive releases and identified the classes which are changed in the next release with respect to the class of the same name in the current release.

3.2. Empirical data collection

We have used two open source Apache projects, Rave and Commons Math. Apache Rave (http://rave.apache.org/) provides an extensible platform for using, integrating and hosting various OpenSocial and W3C Widget related features, technologies and services. Rave is popularly used as engine for internet and intranet portals. Apache Commons Math (https://commons.apache.org) is the Apache Commons Mathematics Library. It contains lightweight and self contained components of mathematics and statistics. Commons Math is basically a package which focuses on small components that are easily integrated and have limited dependencies. In other words, it is not a large library with complex dependencies and configurations. We have analyzed two releases of each project; Rave 0.22, Rave 0.23, Math 2.2 and Math 3.0. Change is calculated between every two successive releases. For example, change is calculated for the common classes between the releases Rave 0.22 and Rave 0.23. We briefly describe the change collection process. To collect the data, DCRS tool (Defect Collection and Reporting System) developed by the students of Delhi Technological University is used [34]. The tool takes as input two predetermined releases and analyzes its source code. Following steps are used to collect the change data using the DCRS tool:

Step 1: Obtain change logs

The DCRS tool extracts the change logs using various Git Bash commands. A Change Log represents all the relevant information related to any modification or alteration done in the source code. The modifications in the source code may occur due to the presence of defects, to add some new functionality, to improve the quality of software etc. Each modification is represented as an individual change record in the Change log. A change record includes unique identifier, timestamp of committing, description of change and list of modified lines of source code. The change logs of the available releases (Rave 0.22, Rave 0.23, Math 2.2 and Math 3.0) of two open source software are obtained from the repository. The releases that did not have any change or were not available are not included in this analysis.

Step 2: Calculating OO metrics

The open source CKJM tool (http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/metric.html) is used for computing the values of OO metrics (indicated in Section 3.1). The metrics are collected for all the Java files and the metrics for all the other types of files (such as xml and other resources) are ignored. For each common class between the two releases, the metrics are collected.

Step 3: Mapping change records to the Java classes

The change records collected in step 1 are finally mapped to the Java classes in the source code. Hence, each class is associated with a binary variable indicating whether a class has been modified ('yes') or not modified ('no').

Thus, the final change - report comprises of the following parts:

- Class name of each Java file.
- Binary dependent variable associated with each class ('yes' or 'no').
- Independent variables (OO metrics) for each class.

After the process of change collection, we found that out of a total of 685 common classes between Rave 0.22 and Rave 0.23, only 225 number of classes have changed. In other words, out of total of 685 classes in rave 0.22, 225 classes have been modified and thus, the binary dependent variable associated with these classes is 'yes'. Similarly, for Apache Commons Math, out of a total of 756 classes in Math 2.2, 178 classes have been modified, corresponding to which the dependent variable is 'yes'. Thus, we can observe that percentage of classes changed is only 32.8% and 23.54% for Rave and Math respectively. This demonstrates that the data is imbalanced.

3.3. Descriptive statistics

We have calculated different statistics for all the independent variables used. This helps in analysing and understanding the data more easily. We can compare the data with other datasets in terms of different statistics and draw important conclusions. For example, in this study, we have found descriptive statistics for the metrics of both the datasets. This helps to understand and analyze the common characteristics of both the projects. To describe the central tendency of the values, we have calculated mean and median. Median is the numeric value that separates the higher half of the data from the lower half, whereas mean represents the average value. To describe the dispersion in the data, different quartiles (25% and 75%) and standard deviation are calculated. Skewness and kurtosis values of the metrics are also calculated. Skewness measures the degree of asymmetry, whereas kurtosis measures peakedness or flatness of the distribution. For a normal distribution, the skewness and kurtosis statistics lie between -1 and $+1$ [14,35]. Besides these statistics, the most common statistics such as minimum and maximum values of the metrics are also found.

Tables 2 and 3 show mean (Mean), median (Median), standard deviation (SD), skewness (Skew), kurtosis (Kurtosis), minimum (Min.), maximum (Max.) and percentiles (Percentiles) of all the metrics of Rave and Math projects respectively.

The following important characteristics about the projects can be observed from the tables:

The size of a class measured in terms of SLOC ranges from 0 to 858 (Rave) and 0 to 6946 (Math). This shows that both Rave and Math are medium - sized projects.

The median value of NOC for both the projects is 0, showing that at least half of the classes do not have any child. This is also indicated by the percentile statistic, which shows that at least 75% of the classes have no children. Also, the mean value of DIT is low, 1.01 for Rave and 0.72 for Math. All this information depicts poor use of inheritance in the projects. Similar results have been shown by [4,36].

The LCOM measure which counts the number of classes with no attribute usage in common, has high values. Thus, the systems support high cohesion. Similar results were observed by other studies as well [33,37].

Thus, we found that both the projects possess some similar characteristics. Therefore, we construct models using the metrics of both the projects and compare their performance.

Table 2
Descriptive statistics of Apache Rave.

		WMC	DIT	NOC	CBO	RFC	LCOM	SLOC
Mean		9.66	1.01	0.02	0.34	10.63	114.49	55.37
Median		6	1	0	0	7	15	36
Mode		5	1	0	0	6	10	31
SD		12.068	0.329	0.19	1.277	12.091	536.706	72.791
Skew		4.904	5.425	11.413	14.035	4.878	12.166	5.193
Kurtosis		35.509	52.546	142.125	279.196	35.263	188.683	39.292
Min.		0	0	0	0	0	0	0
Max.		142	4	3	27	143	10011	858
Percentiles	25	4	1	0	0	5	6	20
	75	11	1	0	0	12	55	63.5

Table 3
Descriptive statistics of Apache Math.

		WMC	DIT	NOC	CBO	RFC	LCOM	SLOC
Mean		8.28	0.72	0.32	7.32	11.69	91.07	103.93
Median		4	1	0	5	7	6	34
Mode		4	1	0	4	0	0	0
SD		11.636	0.615	1.31	11.002	14.88	371.691	406.421
Skew		3.544	1.794	6.95	5.347	2.782	9.484	13.979
Kurtosis		17.466	11.934	61.004	37.966	10.02	124.547	222.144
Min.		0	0	0	0	0	0	0
Max.		113	5	15	123	114	6328	6946
Percentiles	25	2	0	0	2	3	0	9
	75	10	1	0	8	14	36	90

Table 4
Machine learning techniques used.

S.no.	Technique used	Description	Category
1.	NaïveBayes (NB)	Naive Bayes is widely used inductive learning algorithm for ML and data mining. It is a simple probabilistic classifier that uses Bayes' theorem.	BL
2.	BayesNet (BN)	BN is also based on conditional probabilities and uses K2 as search algorithm.	BL
3.	Logitboost (LB)	LB is one of the popular releases of boosting. It uses a regression method as the base learner and it performs additive logistic regression [40].	EL
4.	Adaboost (AB)	AB combines a number of weak hypotheses to get better classification performance. For this, equal weights are assigned to all the training examples and then the weights of the incorrectly classified examples are increased on each round so that a weak learner is forced to focus on the hard examples in the training set [40].	EL

Table 5
Control parameter settings for SBAs.

S.No.	SBAs	Control parameters
1.	GFS-AB	1. No. of labels: 3 2. No. of rules: 8
2.	GFS-LB	1. No. labels: 3 2. No. of rules: 25
3.	GFS-MaxLB	1. No. of labels:3 2. No. of rules: 8
4.	NNEP	1. No. of hidden nodes: 4 2. Transfer: Product Unit 3. No. of generations: 200
5.	HIDER	1. Population size: 100 2. No. of generations: 100 3. Probability of mutation: 0.5 4. Probability of extreme mutation: 0.05 5. Penalty factor: 1 6. Error coefficient: 0
6.	PSOLDA	1. Maximum No. of Iterations:400 2. No. of non-improving iterations: 150 3. Cognition learning factor: 0.8 4. Social learning factor: 1.2 5. Inertial weight (w): 0.5 6. Number of particles: 15
7.	GFS-GP	1. No. of labels: 3 2. No. of rules: 8 3. Population size: 30 4. No. of iterations: 10,000 5. Probability of mutation: 0.01 6. Maximum tree height: 8
8.	GFS-SP	1. No. of labels: 3 2. No. of rules: 8 3. No. of iterations: 10,000 4. Probability of mutation for GA: 0.5 5. Maximum tree height: 8
9.	SLAVE	1. No. of labels: 5 2. Population size: 100 3. No. of iterations: 500 4. Probability of mutation: 0.01 5. Usage of rule weights: yes

4. Data analysis methods

This section explains various hybridized SBAs and ML techniques used for the model construction. ML models are constructed using the default settings of an open source tool, WEKA (http://www.cs.waikato.ac.nz/ml/weka/). Whereas, for each of the SBAs, the default settings of an open source tool, KEEL (http://www.keel.es/) are used. We have not tuned the parameters as over-fitting of the parameters may become a threat to external validity [38]. Thus, we have used default settings of all the parameters. We have specified the control parameter settings of SBAs in Table 5, whereas in Table 6, the control parameter settings of ML models are mentioned. In addition to control parameter settings, we have also mentioned the fitness function for each SBA in Table 7. With the help of parameter settings and the fitness functions, the results can be replicated in various ways [39]. For example, different open source software can be used or different independent variables can be used to construct same models (with same control parameter settings) and then the performance can be compared.

Table 6
Control parameter settings for ML methods.

S.No.	ML Techniques	Control Parameters
1.	NB	1. Use kernel estimator: False 2. Use supervised discretization: False
2.	BN	1. Estimator: SimpleEstimator –A0.5 2. Search Algorithm: K2 –P 1 –S BAYES 3. Use ADTree: False
3.	LB	1. Method threshold: −1.7977 2. No. of folds: 0 3. No. of iterations : 10 4. No. of runs: 1 5. Seed: 1 6. Use resampling: False 7. Weight threshold: 100
4.	AB	1. Classifier: Decision Stump 2. No. of iterations: 10 3. Seed: 1 4. Use resampling: False 5. Weight threshold: 100

Table 7
Fitness functions of SBAS.

S.No.	SBAs	Fitness function
1.	GFS-AB	The fitness of the fuzzy rule "If x belongs to fuzzy set A_j, then class of x is c_j" is calculated as the weighted difference between the sum of the memberships of "class 1" and "class 1" examples. The detailed description can be obtained from [41].
2.	GFS-LB	The fitness of the fuzzy rule "If x belongs to fuzzy set A_j, then class of x is c_j" is calculated as the squared error between the desired output and the logistic transform of the classifier's output. The detailed description can be obtained from [42].
3.	GFS-MaxLB	Fitness $(A^1, ., A^N)=Z_{ik}-(\sum_j I(x_i,j).A^j(x_i).S_k^j)^2$ where, N=no. of membership functions (A); $I(X_i,j)=1$ if rule j is the winner when classifying input x, and 0 if not; S_k^j=weights of the rules Z_{ik}=dependent variable
4.	NNEP	The fitness measure is a strictly decreasing transformation of the error function, $l(\Theta)$, given as: $F(x)=\frac{1}{1+l(\theta)}$
5.	HIDER	Fitness for rule r: $F(r)=2$ (Num–Class Error (r))$+G(r)+$Coverage (r) Num is the total number of processed examples, Class error represents the error when an instance does not belong to the same class but is encompassed in the region defined by the rule, coverage represents the proportion of the entire search space which is included by a specific rule r. It is a two-objective optimization which minimize the errors while maximizing the number of correct classifications.
6.	PSOLDA	Fitness=(TCP+TNCP)/(TCP+FNCP+FCP+TNCP)
7.	GFS-GP	Minimizing the expected classification error and minimizing the complexity of linguistic description.
8.	GFS-SP	Minimizing the expected classification error and minimizing the complexity of linguistic description.
9.	SLAVE	The fitness for the rule is defined as the one which simultaneously has the highest degrees of consistency and completeness and combines both factors using a product operator.

We have used four ML techniques, two of which are Bayesian Learners (BL) and the other two are the Ensemble Learners (EL). These are briefly explained in Table 4.

Next, we discuss various hybridized SBAs used in this study. In hybridized algorithms, the advantages of SBAs as well as non-SBAs are combined into one algorithm. Following are the hybridized algorithms used in this study:

1. Fuzzy Adaboost (GFS-AB)
2. Fuzzy Logitboost (GFS-LB)
3. Logitboost with Single Winner Inference (GFS-MaxLB)
4. Neural net evolutionary programming (NNEP)
5. Hierarchical decision rules (HIDER)
6. Particle Swarm Optimization - Linear Discriminant Analysis (PSOLDA)
7. Fuzzy Learning based on Genetic Programming (GFS-GP)
8. Fuzzy Learning based on Genetic Programming Grammar Operators and Simulated Annealing (GFS-SP)
9. Structural Learning Algorithm in a Vague Environment with Feature Selection (SLAVE)

GFS-AB, GFS-LB, GFS-MaxLB: AB and LB are the boosting algorithms which combine multiple weak classifiers to produce a classifier which has a higher performance than any of its individual elements (or classifiers). When a new classifier is added, each training example is re-weighted and a voting strength is assigned to the classifier. Then, a voting scheme is used to generate a compound classifier by combining all the weak classifiers. GFS-AB and GFS-LB are used to learn fuzzy rule based classifiers. In case of fuzzy rule based classifier, the fuzzy rules are treated as weak hypothesis. Thus, a fuzzy rule base is interpreted as a weighted combination of weak hypotheses. Boosting fuzzy rules imply fitting a single rule to a set of weighted training example. This process continues as many times as there are rules in the base [40]. Thus, GFS-AB and GFS-LB help to build the rule base in an incremental manner by down-weighting the training instances which are correctly classified by a rule instead of removing them which avoids conflicting rules in the rule base [41,42]. The authors Sánchez and Otero [43] have analysed the disadvantages of using boosting algorithms to learn fuzzy classifiers. The high interaction between the rules leading to poor quality rule base is the main drawback. The underlying reason of this drawback is the use of 'sum of votes' scheme to generate the output. Thus, the authors propose an interface scheme known as "single winner" which does not combine the rules with the arithmetic sum [43]. Using this interface scheme, the authors showed that the resultant fuzzy rule base is of high quality as well as good quality.

NNEP: NNEP classification is based on feed-forward neural network. It corresponds to a special class of feed-forward neural network called as product-unit based neural networks (PUNN) which are proposed by Durbin and Rumelhart [44]. PUNN are similar to sigmoidal neural networks which use multiplicative nodes instead of additive ones. The basis functions are nonlinear and represent interactions between variables. The authors in [45] have used an evolutionary algorithm to identify the structure of PUNN model and to estimate the coefficients of the model which minimize the cross entropy error function and other corresponding constraints.

HIDER: HIDER is a supervised learning algorithm which is used for learning rules in continuous and discrete domains [46]. The algorithm produces hierarchical set of rules. In other words, the rules are produced and applied sequentially until the appropriate rule is found which satisfies the conditions. The entire rule base is searched. This implies that if a new training example is correctly classified by a rule r, then all the previous (r-1) rules have been checked and they do not match the conditions.

PSOLDA: Linear Discriminant Analysis (LDA) is a classification method which is used for constructing a classification model. But before any classification model could be constructed, feature selection techniques should be applied to remove the features which are highly correlated with each other. The study [47] shows that if feature selection techniques are not applied, the classification performance of LDA deteriotes. Thus, the study uses a particle swarm optimization (PSO) based technique known as PSOLDA (Particle Swarm Optimization - Linear Discriminant Analysis) to find the useful features and thus, improve the classification accuracy of LDA.

SLAVE: SLAVE is a genetic algorithm which uses the iterative approach to learn fuzzy rules. It learns one fuzzy rule in each execution and the process continues until all the rules are obtained [48]. Thus, the complete set of rules are obtained through sequential repetition of executions of the genetic algorithm used in SLAVE. In several classification problems, the feature space is very large due to existence of large number of variables. Some of the features are highly correlated with each other and thus, it is very important to remove such irrelevant features to produce good classification results. Since, a particular type of rule is used in SLAVE, the irrelevant variables present in a rule are eliminated during the learning or training process. But the computational cost of removing a variable in a rule is very high. Thus, the authors, Gonzalez and Raul [48] have modified SLAVE by improving the genetic algorithm used in SLAVE so that it can learn efficiently when the search space is too large.

GFS-GP, GFS-SP: The study [49] has worked on grammar based GP in which only the subtrees produced from the same rules can be interchanged. However, for conducting mutation, an individual to be mutated is crossed with a randomly generated individual and then, one of the offspring's is selected. The authors have used this mutation operator within an SA-based search to learn both the fuzzy partitions as well as surface structure of a rule base. The study has showed that SA is a good algorithm when the size of the rule base is large.

5. Research methodology

In this section, we describe the parameters used for evaluating the models. The results of the correlation analysis are also reported in this section.

5.1. Performance evaluation measures

To evaluate the performance of the proposed models, we have used two evaluation parameters namely *g*-mean and accuracy. We have observed that for Apache Rave and Math, the data is imbalanced (the number of change pone classes is very few as compared to non-change prone classes). Due to the imbalanced nature of the data, we choose to select *g*-mean. Besides this, some of the previous work has also shown that *g*-mean is the best accuracy estimator of a prediction model [50]. Moreover, these measures focus on identifying the model which correctly predicts majority of the classes as change prone.

Table 8
Confusion matrix.

OBSERVED	PREDICTED	
	0 (non-change prone)	1 (change prone)
0 (non-change prone)	TNCP	FCP
1 (change prone)	FNCP	TCP

Fig. 1. An example showing 10-cross validation.

The computational time (CPU time) required by each technique to develop the predictive models is also analyzed. This allows us to assess the trade-off between the predictive performance of the models and their respective CPU time. Thus, one of the objectives is to analyze the performance of the models based on their CPU time. In other words, we have not kept the CPU time consistent across all the models because we are not trying to optimize on CPU time. The focus of this study is on measuring and comparing the accuracy of the SBA and ML models in predicting change prone classes. We aim to identify which model accurately predicts the change prone classes. In other words, we want to reduce the classification error rate, hence, predictive accuracy of the classifier is the most important factor.

Moreover, given the advent of cloud computing and fast processing, CPU time has become relatively less important as the performance measure. Few studies in literature state that SBAs may consume more running time when applied to large problems [51,52]. Thus, they propose the use of parallel or cloud-based search-based software engineering for effective implementation of search-based techniques. These implementations may lead to significant reduction in the running time for the model development. If we measure the performance of the models using the equal CPU time as the performance measure, then it might be possible that ML techniques outperform SBA techniques. But this will not imply that ML techniques are more accurate in correctly identifying change prone classes. In other words, the objective is not to measure the performance based on the CPU time, but instead, we want to measure the accuracy of all the models.

To describe the performance parameters, we use the following confusion matrix (Table 8):

TCP (True Change Prone): Denotes the number of correctly predicted change prone classes;
FCP (False Change Prone): Denotes the number of incorrectly predicted change prone classes;
TNCP (True Not Change Prone): Denotes the number of correctly predicted non - change prone classes;
FNCP (False Not Change Prone): Denotes the number of incorrectly predicted non - change prone classes.

1. *g*-mean: *g*-mean is based on two accuracies: accuracy of positives ($a+$) and accuracy of negatives ($a-$). The formulae for a+ and a- is given as (refer Table 8):

$$a+ = TCP/(FCP+TCP)$$

$$a- = TNCP/(TNCP+FNCP)$$

Mathematically,
g-mean $= \sqrt{(a+)*(a-)}$
By definition, it is clear that high values of $a+$ and $a-$ are desirable. Thus, when both $a+$ and $a-$ are high, the calculated g-mean is also high.

2. Accuracy: Accuracy is also known as correctness. It is defined as the ratio of the number of classes correctly predicted to the total number of classes.

Mathematically,
$Accuracy = (TCP+TNCP)/(TCP+FNCP+FCP+TNCP)$
Validation method used

The model may be over-fitted if the training and validation data are the same. In other words, the performance of the model on the unseen or future data may be poor. Thus, to validate the model, the data which is not used for training the model should be used. For this purpose, we have used a validation technique known as k-cross validation. In k-cross validation, the data is divided into k equal parts. One part out of the k parts is used for validating (testing) the model and the remaining $k-1$ parts are used for training the model. This process is repeated for all the k parts which allows each data point to be in the validation set. The k results from the folds are then combined to produce a single estimation. This technique provides accurate performance estimation. We have used the value of k as 10. Fig. 1 demonstrates an example with $k=10$ [53].

5.2. Correlation analysis

It is important to have a set of good features (independent and dependent variables) to have a good classification model. A feature set is said to be good if the independent variables are highly correlated with the dependent variable, but are not related with other. Independent variables which are correlated with each other represent redundant information. Thus, we have conducted correlation analysis to study the variation among variables and the amount of correlation between them. The correlation coefficient in the correlation analysis determines the extent of correlation between different variables. The value of correlation ranges from -1 to $+1$. If there is a positive correlation between two variables (metrics), it implies that the value of those two metrics increase together. Whereas, negative correlation implies that as one value increases, the other decreases. Since, the dependent variable in this study is categorical, we cannot check for the normality of the data. Hence, we calculate correlation using a non - parametric correlation test known as Spearman's correlation. Tables 9 and 10 show the Spearman's Rho coefficient of correlation between all the metrics. In this study, the correlations larger than 0.8 are considered as high, in between 0.5 and 0.8 are considered as moderate, and below 0.5 as low [54,55]. In the tables, we have shown the correlation coefficients greater than 0.8 in bold. By definition of correlation, it is clear that correlation of a variable with itself will be totally perfect. In other words, the correlation coefficient value will be 1 (as shown by diagonal elements in the tables).

From the Table 9 (for Rave), there are high correlations among SLOC–WMC and SLOC–RFC. Some of the metrics are moderately correlated with one another. For the metrics of Math dataset (Table 10), we can observe that WMC shows high correlation with LCOM and SLOC. There is also a high correlation between SLOC and RFC. These high correlations are undesirable and may cause collinearity as the metrics are not totally independent and represents redundant information. Thus, we have further calculated the condition number for the metrics. Principal Component (PC) method is applied on the independent variables to find the maximum eigenvalue (e_max) and minimum eigenvalue (e_min). Then we find the conditional number given as $\lambda = \sqrt{\text{e\_ max}/\text{e\_ min}}$. If the value of conditional number is less than 30, we say that there does not exist multicollinearity between the variables [56]. We calculated the condition number for all the metrics of each release. We observed that the condition number is well below 30 for both software. This implies, we can use all the metrics together in conjunction with one another. In other words, multicollinearity is tolerable.

Table 9
Correlation analysis of Rave.

	WMC	DIT	NOC	CBO	RFC	LCOM	SLOC
WMC	1.000						
DIT	−0.031	1.000					
NOC	0.082	0.002	1.000				
CBO	0.208	−0.181	0.241	1.000			
RFC	0.652	−0.031	0.082	0.208	1.000		
LCOM	0.552	−0.034	0.082	0.209	0.672	1.000	
SLOC	**0.865**	−0.023	0.091	0.119	**0.865**	0.664	1.000

Table 10
Correlation analysis of Math.

	WMC	DIT	NOC	CBO	RFC	LCOM	SLOC
WMC	1.000						
DIT	−0.158	1.000					
NOC	0.250	−0.102	1.000				
CBO	0.446	−0.144	0.323	1.000			
RFC	0.546	−0.172	0.268	0.499	1.000		
LCOM	**0.893**	−0.196	0.196	0.370	0.675	1.000	
SLOC	**0.853**	−0.230	0.239	0.418	**0.937**	0.678	1.000

Table 11
Performance of the hybridized SBAs models for Rave dataset.

S.No.	Technique	*g*-Mean (%)	Accuracy (%)	Time (s)
1.	GFS-AB	0.63	0.76	480
2.	GFS-LB	0.64	**0.76**	380
3.	GFS-MaxLB	0.61	0.75	1020
4.	NNEP	0.58	0.74	900
5.	HIDER	0.58	0.74	**60**
6.	PSOLDA	**0.65**	0.75	360
7.	GFS-GP	0.59	0.74	4680
8.	GFS-SP	0.61	0.72	1440
9.	SLAVE	0.58	0.74	**240**

6. Result analysis

In this section, we discuss the results of various hybridized SBAs as well as ML techniques used for the prediction of change prone classes. For each of the software used, different hybridized and ML models are constructed and their performances are measured using two performance measures, g-mean and accuracy. The computational time required by each SBA and ML to complete their execution and produce the models is also measured. Before the construction of models, it is very essential to identify a subset of significant variables (OO metrics). We have used a technique known as correlation based feature selection (CFS) [57] to obtain a subset of good features. M.A. Hall proved that "classification accuracy using reduced feature set is equal or better than accuracy using complete feature set." [58]. The independent variables obtained after applying CFS are used in the construction of various prediction models. The advantages of using CFS are efficient and more accurate model prediction and reduction in dimensionality. The metrics obtained after applying CFS to the metrics of Rave are WMC, RFC and SLOC. Whereas, the metrics obtained after applying CFS to the metrics of Math are RFC, LCOM and SLOC. Thus, we observe that among a total of six metrics used as independent variables, SLOC and RFC are good indicators of change proneness as these metrics are selected in the feature subset obtained after applying the CFS method. Thus, they can be used by researchers and practitioners for predicting change prone classes of some other similar software as Rave and Math.

6.1. Model evaluation using 10-cross validation

The accuracy of the models predicted would be highly optimistic if testing is done on the same dataset from which the training model is derived. To predict the correct accuracy of the model, it should be applied on a different dataset. Thus, we have used 10-cross validation. Tables 11, 12, 15 and 16 report the results of 10-cross validation of different hybridized SBAs and ML techniques respectively. Each of the table presents the results using two different evaluation measures; g-mean and accuracy. In addition to this, we have also reported the computational time (in seconds) taken by each technique for developing the models. The *g*-mean, accuracy and CPU time of SBAs for Rave and Math are represented in Tables 11 and 12. Similarly, the g-mean, accuracy and CPU time of the ML techniques for Rave and Math are represented in Tables 15 and 16.

Table 11 shows that among the prediction models of Rave, the highest g-mean (0.65) is obtained by PSOLDA model, followed by GFS-LB. Whereas, we observe that the accuracy obtained by most of the models is comparable, with the highest being of GFS-LB followed by PSOLDA as well as GFS-MaxLB. Infact, most of the models constructed for Rave dataset have shown exactly the same accuracy of 0.74.

Among the models of Math (Table 12), similar and comparable results are obtained as the results of the models constructed for Rave. The highest g-mean (0.69) and accuracy (0.78) is given by PSOLDA model. The accuracy of most of the models is quite similar and most of the models have obtained exactly the same accuracy. Thus, we say that PSOLDA has given high values of g-mean and accuracy. However, with respect to CPU time, PSOLDA has not taken the minimum time to

Table 12
Performance of the hybridized SBAs models for Math dataset.

S.No.	Technique	g-Mean (%)	Accuracy (%)	Time (s)
1.	GFS-AB	0.61	0.76	360
2.	GFS-LB	0.64	0.77	380
3.	GFS-MaxLB	0.59	0.76	600
4.	NNEP	0.60	0.76	900
5.	HIDER	0.61	0.76	60
6.	PSOLDA	**0.69**	**0.78**	300
7.	GFS-GP	0.52	0.75	4200
8.	GFS-SP	0.50	0.75	1200
9.	SLAVE	0.50	0.76	180

Table 13
Modified parameter settings.

Technique	Number of generations/iterations	Population size
Hider_Rave	400	400
Slave_Rave	800	400
Hider_Math	400	400
Slave_Math	500	300

Table 14
Performance re-evaluated for fair comparison.

Technique	g-Mean (%)	Accuracy (%)	Time (s)
Hider_Rave	0.58	0.74	360
Slave_Rave	0.57	0.73	420
Hider_Math	0.60	0.72	300
Slave_Math	0.48	0.73	360

Table 15
Performance of the ML models for Rave dataset.

S.No.	Technique	g-Mean	Accuracy	Time (s)
1.	AB	0.57	0.60	3
2.	LB	0.57	0.60	2
3.	NB	0.55	0.59	2
4.	BN	0.58	0.60	3

Table 16
Performance of the ML models for Math dataset.

S.No.	Technique	g-Mean	Accuracy	Time (s)
1.	AB	0.57	0.66	2
2.	LB	0.56	0.65	3
3.	NB	0.55	0.65	3
4.	BN	0.58	0.67	3

develop prediction model. It has taken 360 and 300 s for model development of Rave and Math respectively, in contrast to Hider and Slave which have taken comparatively lower CPU time to execute. Besides Hider and Slave, all the SBAs have taken more execution time and have also shown lower performance results than as those of PSOLDA. Also, some of the SBAs have taken a very long time as compared to other SBAs. We observe that for both Rave and Math datasets, GFS-GP and GFS-SP have taken a long time as compared to other SBAs. However, the difference in the performance of GFS-GP and GFS-SP when compared with other SBAs is not much as compared to the difference in their CPU time. Thus, they may be carefully selected since they have shown decent predictive performance but have consumed large amount of CPU time for model development.

From the present discussion, it appears that all the SBAs except Hider and Slave have shown lower performance or are poor predictors of change proneness. However, to enable better and fair comparisons amongst PSOLDA, Hider and Slave, we

have conducted another experiment where we have tried to run Hider and Slave for atleast the same amount of time as PSOLDA and then determined their accuracy of change prediction [39]. In other words, we have tried to run Hider and Slave of Rave dataset for 360 and 420 s respectively, whereas for Math dataset, they are run for 300 and 360 s respectively. There are number of ways by which execution time can be increased. For example, increasing the number of generations/iterations or population size will lead to increase in execution time. Thus, we have experimented with these parameters to increase the execution time. Also, we understand that increasing the execution time indirectly implies that we are increasing the number of fitness evaluations. It seems that if the number of fitness evaluations are increased, then the accuracy might increase. In other words, it might be possible that Hider and Slave which have shown lower performance than PSOLDA, perform better or comparable as POLDA. Thus, to prove/disprove this, we have increased the execution time by changing the default parameter settings of Hider and Slave as shown in Table 13. All the other control parameters are kept same.

The performance results of Hider and Slave (of both the datasets) along with their time of execution is shown in Table 14. It can be observed that by using the modified parameter settings (Table 13), we are successful in achieving approximately the desired time. In other words, both Hider and Slave are run for approximately the same time or more time than was required by PSOLDA (360 seconds for Rave and 300 seconds for Math). Now, comparing the performance of Hider and Slave with PSOLDA allows for fair comparison.

Table 14 shows that the models of Rave have shown nearly the same accuracy and g-mean values as compared to the values obtained when number of fitness evaluations were less. By modifying the default parameter settings, Slave model has consumed more execution time than that of PSOLDA model of Rave and despite that it has shown lower performance than PSOLDA.

Similar inference is obtained for the models of Math dataset. We observed that their performance is not even the same, but it further degraded after they are made run for more amount of time. For example, the accuracy of Hider was 0.76% with execution time = 60 s. When we tried to run Hider for same amount of time as PSOLDA, we observed that the accuracy came down to 0.72%.

For more comprehensive analysis, the accuracy of all the three models (PSOLDA, Hider and Slave) have been compared graphically as shown in Fig. 2. The graph clearly shows that the accuracy of Hider and Slave after they have been executed for atleast the same time as PSOLDA is not even comparable to PSOLDA. We can observe that the accuracy of Hider and Slave is either comparable or even less as it was when they were run for less time (can be seen from Tables 11 and 12).

Hence, we found that even with increasing population size, number of generations/iterations, and hence more number of fitness evaluations, better or even comparable results are not obtained. Overall, we summarized that we cannot find better prediction results, even after increasing the number of fitness evaluations.

From the above discussion, we can conclude that among all the hybridized SBAs constructed to predict change prone classes of both Rave and Math datasets, PSOLDA model has outperformed all the other models. Thus, researchers and practitioners may use PSOLDA model to predict change prone classes of some future release of Rave and Math. At the same time, the model can also be used for some other software possessing similar characteristics as Rave and Math. Thus, we observe that the models once constructed can be used to predict change prone classes of future or upcoming release of the same software or other similar software.

We have also constructed ML models and measured their performance using the same performance measures. This allows to draw a comparison between the performance of the SBAs and the ML techniques. From Tables 15 and 16, we observe that all the four ML models have shown very similar performance. The g-mean of the constructed models of both

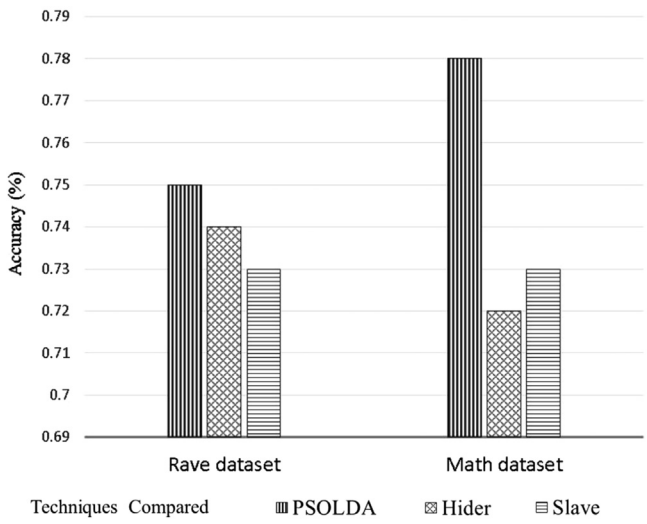

Fig. 2. Accuracy of Hider and Slave compared with PSOLDA.

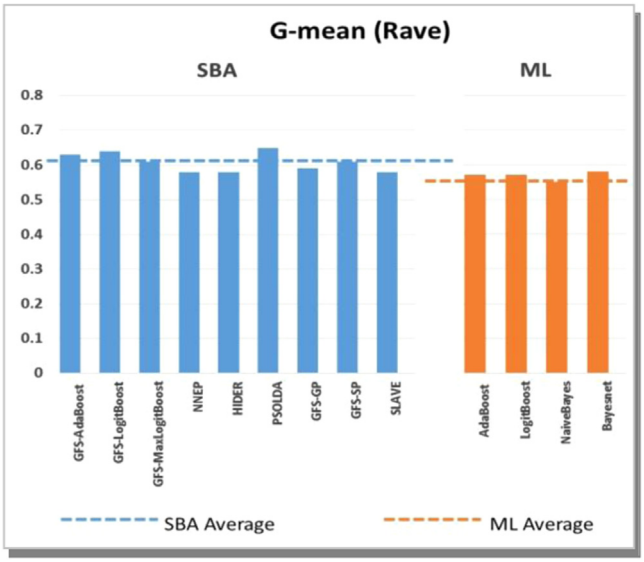

Fig. 3. Comparison of g-mean for the models of Rave.

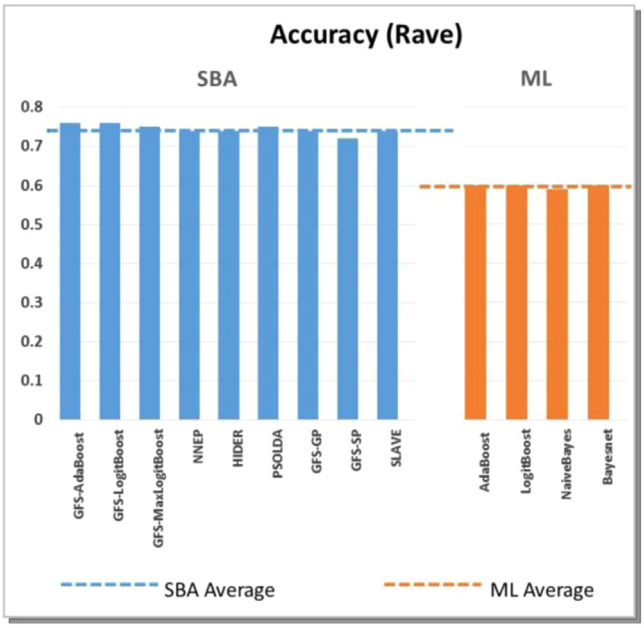

Fig. 4. Comparison of accuracy for the models of Rave.

Rave and Math lie within the range of 0.55 and 0.58. We can observe that all the ML models are very effective in terms of CPU time utilization. In other words, they have taken less time for developing models used for change prediction.

Similar to SBAs, we tried to increase the running /execution time of ML models by changing their default control parameters. We observed that even after increasing the parameters like number of iterations, number of runs, there is very less or no effect on execution time (it increases only by some milli seconds). The performance of the models measured in terms of g-mean and accuracy is also consistent and has not increased or decreased.

Overall, we conclude that the performance results of all the ML models are comparable with respect to all the performance metrics (g-mean, accuracy and time).

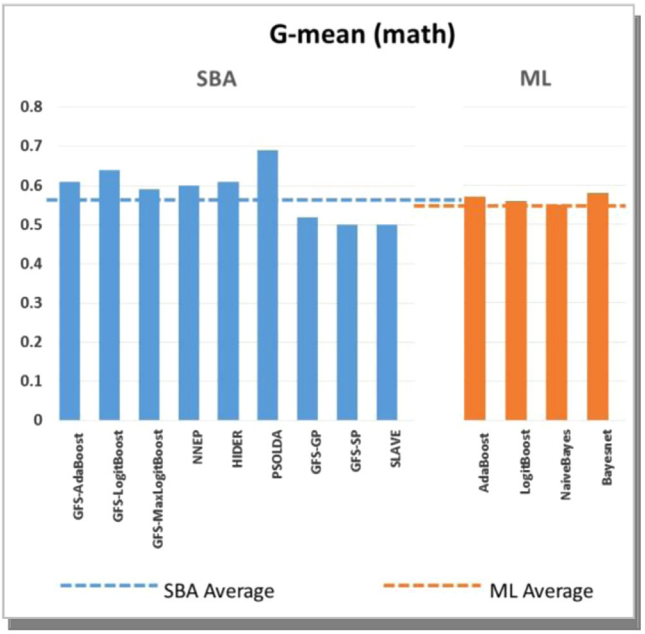

Fig. 5. Comparison of g-mean for the models of Math.

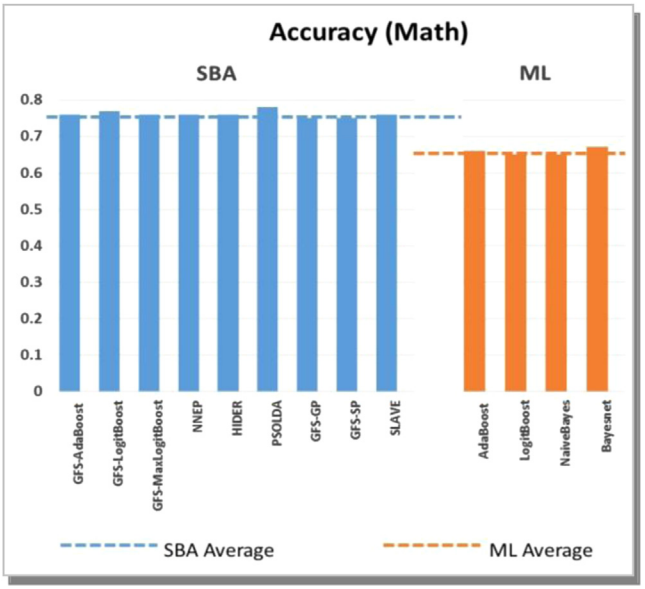

Fig. 6. Comparison of accuracy for the models of Math.

6.2. Comparison of SBAs and ML techniques

When we compare the performance results of SBAs with ML techniques, we observe that most of the SBAs have outperformed the ML techniques by showing higher values of g-mean and accuracy. The comparison is also demonstrated diagrammatically in Figs. 3, 4, 5 and 6.

Figs. 3–6 clearly demonstrate that the average values of g-mean and accuracy of the SBAs are higher than average values of g-mean and accuracy of ML models. We observe that among the models constructed for Rave, the range of g-mean obtained by the SBAs is 0.59 to 0.65. Whereas, the ML models of Rave have obtained the highest g-mean of only 0.58. Similarly, the accuracy obtained by SBAs is much higher than the accuracy of ML techniques. The models constructed using SBAs to predict change prone classes of Math have also shown higher values of g-mean and accuracy than the ML models. One of the important reasons for this superior performance of hybridized SBAs is that they combine the advantages of both

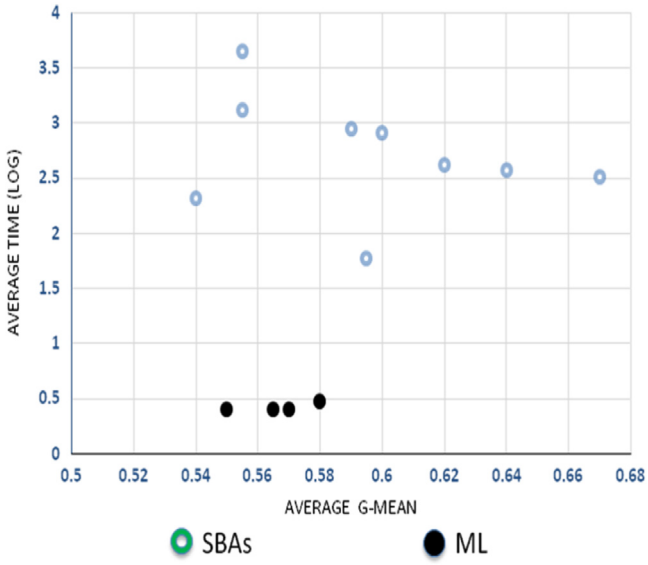

Fig. 7. Average CPU time vs. average G-mean.

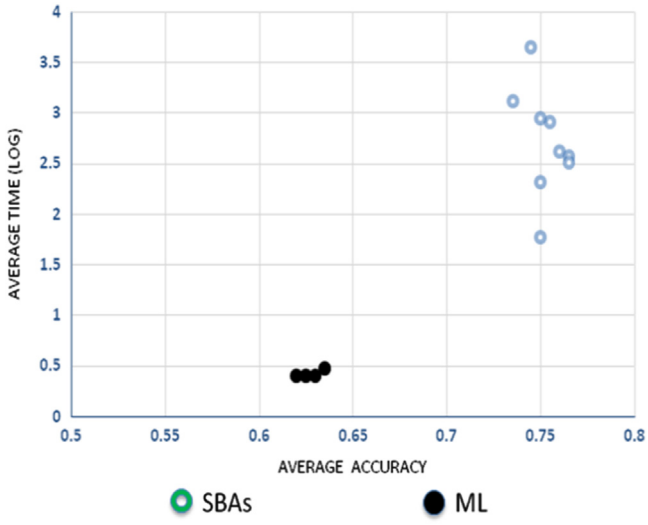

Fig. 8. Average CPU time vs. average accuracy.

SBAs as well as non - SBAs. Thus, we conclude that the hybridized SBAs should be used for predicting change prone classes over the ML models.

However, when we take into account the computational time and then analyze the performance, we find that the running or the computational time of the SBA models is large as compared to the ML models. When we analyze the trade - off between performance and CPU time, we find that the difference in the performance of SBAs and ML models is not much as compared to the difference in their computational time.

In order to evaluate the trade-off between CPU time and predictive performance, we plotted graphs between CPU time and the performance measures as shown in Figs. 7 and 8. Fig. 7 depicts the average CPU time and the average G-mean of all the SBAs and ML techniques of both the datasets. Similarly, Fig. 8 depicts average CPU time and the average accuracy of all the SBAs and ML techniques of both the datasets. We have taken $\log_{10}$ of average time, just for scaling purposes. The figures represent each technique as a point in the space according to its corresponding average CPU time taken by it and the average value of performance metric achieved by it on both the datasets. The hollow dots in the figures represent SBAs and the solid dots represent the ML techniques. The figures show that the ML techniques are effective in terms of the average CPU time as compared to the SBAs. However, the performance of ML techniques in terms of g-mean and accuracy is not exceptional and it is lower than most of the SBAs. On the other hand, SBAs have shown high values of g-mean and accuracy, but they have

exhibited average to high CPU time for model development. From Fig. 8, we can observe that all the SBAs used in this study have exhibited much high accuracy values as compared to ML techniques. Large running time of SBAs is also stated as a drawback in some studies [51,52]. However, the new upcoming technology, multi-processor or parallel systems and cloud based Search-based Software Engineering may lead to significant reduction in computational time. Besides this, some authors [59–61] have also suggested and proposed certain parallel search based approaches which can be integrated or used along with SBA models to produce promising results with less computational efforts. Thus, given the advantages of SBAs over ML models, we conclude that SBAs should be used for accurate prediction of change prone classes.

6.2.1. Comparison using statistical test

We checked if there is a significant difference in the performance of both the types of models (SBAs and ML) using a statistical test known as Wilcoxon Signed Rank test [62]. We found that among SBAs, PSOLDA has given the best performance. Thus, we perform a statistical comparison between the performance of PSOLDA and all the ML models.

The detailed description of Wilcoxon Signed Rank is as follows:

Wilcoxon signed-rank test is a non-parametric test which performs pairwise comparisons of the difference in performance of the models. It is based on the following hypothesis:

Null Hypothesis (Ho): There is no statistical difference in the performance of PSOLDA and ML models.
Alternative hypothesis (Hi): There exists statistical difference in the performance of PSOLDA and ML models.

It is defined as

$$R^{+} = \sum rank\,(m_i),\ \text{when } m_i > 0$$

$$R^{-} = \sum rank(m_i),\ \text{when } m_i < 0$$

Where, m_i is the difference between performance measure of two subjects. The differences are ranked based on their absolute values. Here, R^{+} is the sum of ranks for the given data sets where the second model outperforms the first model and R^{-} is the sum of ranks for the given datasets where the first model outperforms the second model.

We exclude the pairs where the absolute difference is 0. Let n_r be the reduced number of pairs.

Finally, the Wilcoxon value (Z) is calculated as

$$Z = \left[Q - \{(1/4)n_r(n_r+1)\}\right]/A,$$

where,

$$A = \sqrt{(1/24)n_r(n_r+1)\,(2n_r+1)}$$

$$Q = min\left(R^{+},\ R^{-}\right)$$

If the Wilcoxon value (z) is in the critical region with specific level of significance (0.05), then the Null hypothesis is rejected, otherwise Null hypothesis is accepted.

If PSOLDA model give comparable results with the ML models, we conclude that SBAs are effective in identifying change prone classes.

The result of comparison between PSOLDA and ML models of both Rave and Math datasets is shown in Table 17. The significance value for each of the comparison is shown in brackets. Table 17 shows that the significance level obtained when PSOLDA model is compared with ML models is more than 0.05. Thus, null hypothesis is accepted, i.e. there is no statistical significant difference in the performance of PSOLDA and ML models. Now, based on the sum of the ranks (given by Wilcoxon Signed Rank test), the results show that the performance of PSOLDA model is better than each of the ML models (denoted by ↑).

This shows that SBAs can be effectively utilized to identify change prone classes in the upcoming releases of the same dataset or other similar dataset.

7. Discussion of research questions

In this section, we discuss various RQs stated by us in Section 1. We have provided answers to each one of these RQs in various sections of this paper. Here, we try to summarize them.

Table 17
Result of Wilcoxon test.

	AB	LB	NB	BN
PSOLDA	↑ **(0.18) > 0.05**	↑ **(0.18) > 0.05**	↑ **(0.18) > 0.05**	↑ **(0.18) > 0.05**

RQ1: Are OO metrics related to change proneness?

In this study, we have used CK metrics along with SLOC as independent variables to construct prediction models. In Section 6, we discussed the use of CFS technique to identify the metrics which are significant in predicting the dependent variable (change proneness). The results of CFS showed that the 3 metrics each of Rave and Math are related to change proneness. These metrics can be used to identify change prone classes of some future release of the same or similar software.

RQ2: What is the predictive performance of SBA's for predicting change prone classes?

We have used 9 SBAs in this study and assessed their performance in predicting change prone classes. Their performance has been measured using g-mean and accuracy. The results are shown in Tables 11 and 12, Section 6.1. The results prove the effectiveness of SBAs in predicting change prone classes.

RQ3: Among multiple SBAs used in this study, which SBA can be used for predicting change prone classes?

Among multiple SBAs used in this study, the results of 10-cross validation in Tables 11 and 12 show the best performance by PSOLDA model constructed for both the datasets. PSOLDA has shown the highest g-mean for both Rave and Math.

RQ4: Is the performance of SBAs better than ML techniques for predicting change-prone classes?

The comparison between the SBAs and the ML techniques can be clearly seen from the Figs. 3, 4, 5 and 6. From the figures, it can be seen that the average values of g-mean and accuracy are higher for SBAs as compared to ML techniques. This holds true for the models of both the datasets (Rave and Math). Thus, we say that the performance of SBAs is better than ML techniques for predicting change prone classes. However, the better performance is not statistically significant as shown by the Wilcoxon test in Section 6.2.1.

RQ5: Are the performance results consistent with different evaluation measures, viz. g-mean and accuracy?

Yes, the performance results are consistent with different evaluation measures. SBAs have shown higher values of both g-mean and accuracy as compared to ML techniques.

RQ6: What is the computational time (CPU time) taken by change prediction models developed using different SBAs and ML techniques? What is the trade-off between the CPU time and predictive performance of different SBAs and ML techniques?

The CPU time required by each SBA and the ML technique for model development is shown in Tables 11, 12, 15 and 16. To analyze the trade-off between the CPU time and the predictive performance, we have plotted different graphs as shown in Figs. 7 and 8. The figures show that the predictive performance of SBAs is better than the ML techniques. However, the CPU consumption for SBAs is moderate to large as compared to ML techniques. With the help of parallel and cloud based computing, the CPU time can be reduced. Thus, there is a trade-off between the predictive performance and the CPU time required for model development.

8. Application of the work

The results of this study could be used by software practitioners and researchers in their work in the following way:

1. We found that among a number of OO metrics used, SLOC and RFC are highly related to change proneness. In both the projects used for analysis, we found these metrics to be strongly associated with the dependent variable (change proneness). Thus, practitioners may concentrate on these metrics during the design of the system which would help to identify change prone classes during the early phases of software development life cycle. This would help in improving software quality and produce a good quality software product. Conducting more such replicated studies would result in more generalized conclusions and results. Once the change prone classes are identified, it will help the organization in the following ways:

- The designers can redesign such classes so that less number of classes undergo change during the later phases of software development as well as less number of changes are encountered. For example, we found that RFC is strongly connected to change proneness and RFC is a coupling metric. Thus, to alter the design, the designers can redesign the class in such a way that the number of classes to which that class is coupled is reduced.
- Besides this, the testers can focus their testing resources and perform rigorous verification activities on the classes identified to be change prone.

- The developers can take a decision regarding the distribution of the limited resources available with them. For example, they can decide that on which classes the resources (time, money and manpower) should be focused on [13,20]. This would result in a good quality product as less number of defects and changes are introduced during the later phases.

2. The design measurements, particularly related to RFC and SLOC can be used by software practitioners to establish quality benchmarks across different software organizations. In other words, the metric values of any software product of similar nature (as Apache Rave and Math) can be compared with the quality benchmarks to have an approximate idea about the permissible range of values. If there is large deviation in the values, then the developers can take corrective and remedial actions.
3. Among the two popularly used domains under which most of the techniques used to predict change prone classes fall are ML and SBAs. This study allows the practitioners to take a decision regarding which method should be used. We have found that SBAs have outperformed ML techniques. Thus, SBAs can be used for efficient model building.

9. Threats to validity

In this section, we present various threats to validity which must be taken into consideration.

9.1. Construct validity

Construct validity refers to the extent to which the dependent and independent variables accurately measure the concept they intend to measure [14].

We have collected the dependent variable using the DCRS tool, which efficiently extracts the change information from change logs. Thus, the dependent variable is accurately measured. However, we did not take into account the type of change (corrective, adaptive, perfective or preventive) a class may go through which poses a threat to the construct validity of the dependent variable. Moreover, in literature, there have been experiments [1,2,58] to validate the correctness of OO metrics which further reduces this threat for the independent variables of the study.

9.2. Internal validity

The threat to internal validity is present if there is “causal effect” of the OO metrics on change proneness. However, in order to determine this effect, we need to conduct controlled experiments. In controlled experiments, the values of other OO metrics are controlled in order determine the “causal effect” of one OO metric [36]. There are some studies [14,15,63] which concluded that size has strong confounding effect on the relationship between change proneness and the metrics and thus, should be taken into account. Since, we did not aim to determine the causal effect, this threat exists in the study. This threat also exists in other studies [21,36].

9.3. External validity

External validity is the extent to which our results can be generalized universally. Since, we have taken into account only two open source datasets, we cannot generalize our results. For generalization, we need to take into account different data sets belonging to varying applications and developed in different environments. We plan to carry out the replicated study using different application domain datasets.

10. Conclusion

An accurate and correct prediction of classes of software that are more prone to changes in the next release of the software has become one of the important activities of software engineering. This would lead to substantial saving of resources by judiciously allocating them to the change prone classes which require focused attention.

In this paper, the aim is to investigate the performance of various hybridized SBAs for correct and accurate prediction of change prone classes in the earlier phases of software development life cycle. For this purpose, we have used various OO metrics and investigated the use of these metrics in the prediction of change proneness in two open source software, Apache Rave and Math. The performance of different models is evaluated using g-mean and accuracy. We have also constructed ML models to predict change prone classes so that a comparison can be drawn between the performances of SBAs and ML models. Furthermore, the study also measured the computational time (CPU time) required by each technique to develop change prediction and analyzed the trade-off between CPU time and predictive capability of different techniques.

The important findings of the work are:

1. The metrics SLOC and RFC are significant predictors of change proneness, as these are selected after applying CFS technique on both the datasets. Thus, they may be used by researchers to predict change prone classes and construct various prediction models.
2. The hybridized SBAs have proved to be capable of predicting change prone classes by showing high values of g-mean and accuracy. Among multiple SBAs used, PSOLDA has outperformed the other models with the highest values of g-mean and accuracy. It has also consumed less CPU time as compared to all the SBAs except Hider and Slave.
3. To conduct fair evaluation amongst the PSOLDA, Hider and Slave, the execution time of Hider and Slave is increased to make them atleast same as that of PSOLDA. For increasing the time, the parameters like population size, number of generations/iterations are increased and then the performance is evaluated and compared. However, we observed that even with increasing population size, number of generations/iterations, and hence more number of fitness evaluations, better or even comparable results as that of PSOLDA are not obtained. Thus, we conclude PSOLDA has outperformed followed by GFS-LB. Since, PSOLDA has given the best performance among SBAs, we have performed a statistical comparison between the performance of PSOLDA and all the ML models using a statistical test known as Wilcoxon Signed Rank test. We conclude that there is no statistical significant difference in the performance of PSOLDA and ML models.
4. There exists a trade-off between the predictive performance of the models developed using various techniques and the CPU time required by them for model development. It is observed that ML techniques are very effective in terms of CPU time, but they give low performance as compared to SBAs. In contrast, SBAs have shown superior predictive performance than the ML models. The range of g-mean and accuracy obtained by SBAs is quite high as compared to the range of ML models. But some of the SBAs (GFS-GP and GFS-SP) take very long time for model development. Thus, researchers should be careful while selecting them for developing change prediction model. The long running time can be reduced with the help of parallel and cloud based computing.

We plan to replicate our work across various datasets to give generalized results across different organizations. In this study, the confounding effect of size on the relationship between change proneness and the metrics is not taken into account. We intend to pursue this in our future work. Lastly, to understand the economic viability of the models, we may carry out cost benefit analysis of the models.

References

[1] Briand L, Daly W, Wust J. Unified framework for cohesion measurement in object-oriented systems. Empir Softw Eng 1998;3:65–117.
[2] Briand L, Daly W, Wust J. A unified framework for coupling measurement in object-oriented systems. IEEE Trans Softw Eng 1999;25:91–121.
[3] Bieman J, Kang B. Cohesion and reuse in an object-oriented system. In: Proceedings of the ACM Symposium on Software Reusability (SSR'94); 1994. p. 259–262.
[4] Cartwright M, Shepper M. An empirical investigation of an object-oriented software system. IEEE Trans Softw Eng 1999;26:786–96.
[5] Chidamber S, Kemerer C. A metrics suite for object-oriented design. IEEE Trans Softw Eng 1994;20:476–93.
[6] Li W, Henry W. Object-oriented metrics that predict maintainability. J Softw Syst 1993;23:111–22.
[7] Lorenz M, Kidd J. Object-oriented software metrics. Prentice-Hall; 1994.
[8] Bansiya J, Davis C. A hierarchical model for object-oriented design quality assessment. IEEE Trans Softw Eng 2002;28(2002):4–17.
[9] Askari M, Holt R. Information theoretic evaluation of change prediction models for large-scale software. In: ACM: Proceedings of international workshop on mining software repositories; 2006. p. 126–132.
[10] Han AR, Jeon SU, Bae DH, Hong JE. Behavioural dependency measurement for change-proneness prediction in UML 2.0 design models. In: Proceedings of the 32nd IEEE International Computer Software and Applications Conference; 2008. *p.* 76–83.
[11] Han AR, Jeon SU, Bae DH, Hong JE. Measuring behavioural dependency for improving change-proneness prediction in UML-based design models. J Syst Softw 2010;83:222–34.
[12] Koru AG, Tian J. Comparing high-change modules and modules with the highest measurement values in two large-scale open-source products. IEEE Trans Softw Eng 2005;31:625–42.
[13] Koru AG, Liu H. Identifying and characterizing change-prone classes in two large-scale open-source products. J Syst Softw 2007;80:63–73.
[14] Zhou Y, Leung H, Xu B. Examining the potentially confounding effect of class size on the associations between object-oriented metrics and change-proneness. IEEE Trans Softw Eng 2009;35:607–23.
[15] Lu H, Zhou Y, Xu B, Leung H, Chen L. The ability of object-oriented metrics to predict change-proneness: a meta-analysis. Empir Softw Eng 2011;17:200–42.
[16] M. Harman. The relationship between search based software engineering and predictive modelling. In: Proceedings of PROMISE'10; 2010.
[17] Jones BF, Sthamer HH, Eyers D. Automatic structural testing using genetic algorithms. Softw Eng J 1996;11:299–306.
[18] Harman M, Jones BF. Search based software engineering. Inf Softw Technol 2001;43:833–9.
[19] Lindvall M. Are large C++ classes change-prone? An empirical investigation Softw Pract Exp 1998;28:1551–8.
[20] Elish MO, Al-Khiaty MA. A suite for quantifying historical changes to predict future change-prone classes in object-oriented software. J Softw: Evol Process 2013;25:407–37.
[21] Malhotra R, Khanna M. Investigation of relationship between object-oriented metrics and change proneness. Int J Mach Learn Cybern 2013;4:273–86.
[22] Yuan X, Khoshgoftaar TM, Allen EB, Ganesan K. An application of fuzzy clustering to software quality prediction. In: Proceedings of the 3rd IEEE Symposium on Application Specific Systems and Software Engineering Technology; 2000. p. 85–90.
[23] Gyimothy T, Fernec R, Siket I. Empirical validation of object – oriented metrics on open source software for fault prediction. IEEE Trans Softw Eng 2005;31:897–910.
[24] C. Catal B. Diri B. Ozumut An artificial immune system approach for fault prediction in object-oriented software. In: Proceedings of the second international conference on dependability computer systems; 2007. p. 1–8.
[25] Vandecruys O, Martens D, Baesens B, Mues C, Backer MD, Haesen R. Mining software repositories for comprehensible software fault prediction models. J Syst Softw 2008;81:823–39.
[26] Carvalho AB, Pozo A, Vegilio SR. A symbolic fault-prediction model based on multiobjective particle swarm optimization. J Syst Softw 2010;83:868–82.
[27] Pendharkar PC. Exhaustive and heuristic search approaches for learning a software defect prediction model. Eng Appl Artif Intell 2010;23:34–40.
[28] Shukla KK. Neuro-genetic prediction of software development effort. Inf Softw Technol 2000;42:701–13.

[29] Burgess CJ, Lefley M. Can genetic programming improve software effort estimation? A comparative evaluation Inf Softw Technol 2001;43:863–73.
[30] Bardsiri VK, Jawawi DNA, Hashim SZM, Khatibi E, PSO-based A. model to increase the accuracy of software development effort estimation. Softw Qual J 2013;21:501–26.
[31] Minku LL, Yao X. Software effort estimation as a multiobjective learning problem. ACM Trans Softw Eng Methodol 2013;22.
[32] Malhotra R, Khanna M. The ability of search-based algorithms to predict change-prone classes. Softw Qual Prof 2014;17:17–31.
[33] Aggarwal KK, Singh Y, Kaur A, Malhotra R. Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: a replicated study. Softw Process: Improv Pract 2009;16:39–62.
[34] Malhotra R, Pritam N, Nagpal K, Upmanyu P. Defect collection and reporting system for git based open source software. In: International Conference on Data Mining and Intelligent Computing; 2014. p. 1–7.
[35] Malhotra R, Jain A. Fault prediction using statistical and machine learning methods for improving software quality. J Inf Process Syst 2012;8:241–62.
[36] Briand L, Wust J, Lounis H. Replicated case studies for investigating quality factors in object oriented designs. Empir Softw Eng J 2001;6:11–58.
[37] Singh Y, Kaur A, Malhotra R. Empirical validation of object-oriented metrics for predicting fault proneness. Softw Qual J 2010;18:3–35.
[38] Arcuri A, Fraser G. On parameter tuning in search based software engineering, search based software engineering. Lecture notes in computer science, P. 33–47.
[39] Črepinšek M, Liu SH, Mernik M. Replication and comparison of computational experiments in applied evolutionary computing: common pitfalls and guidelines to avoid them. Appl Soft Comput 2014;19:161–70.
[40] Freund Y, Schapire RE, Short A. Introduction to Boosting. J Jpn Soc Artif Intell 1999;14:771–80.
[41] Jesus MJ, Hoffmann F, Navascués LJ, Sánchez L. Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. IEEE Trans Fuzzy Syst 2004;12:296–308.
[42] Otero J, Sanchez L. Induction of descriptive fuzzy classifiers with the Logitboost algorithm. Soft Comput 2006;10:825–35.
[43] Sanchez L, Otero J. Boosting fuzzy rules in classification problems under single-winner inference. Int J Intell Syst 2007;22:1021–34.
[44] Durbin R, Rumelhart D. Products units: a computationally powerful and biologically plausible extension to back propagation networks. Neural Comput 1989;1:133–42.
[45] Martınez-Estudilloa FJ, Hervas-Martınezb C, Gutierrezb PA, Martınez-Estudilloa AC. Evolutionary product-unit neural networks classifiers. Neurocomputing 2008;72:548–61.
[46] Aguilar-Ruiz JS, Riquelme JC, Toro M. Evolutionary learning of hierarchical decision rules. IEEE Trans Syst, Man, Cybern b: Cybern 2003;33:324–31.
[47] Lin SW, Chen SC. PSOLDA: a particle swarm optimization approach for enhancing classification accuracy rate of linear discriminant analysis. Appl Soft Comput 2009;9:1008–15.
[48] Gonzalez A, Perez R. Selection of relevant features in a fuzzy genetic learning algorithm. IEEE Trans Syst, Man, Cybern b: Cybern 2001;31:417–25.
[49] Sanchez L, Couso I, Corrales JA. Combining GP operators with SA search to evolve fuzzy rule based classifiers. Inf Sci 2001;136:175–91.
[50] Ma Y, Cukic B. Adequate and precise evaluation of quality models in software engineering studies. In: Proceedings of the 3rd IEEE/ACM workshop predictor models in software engineering (PROMISE); 2007.
[51] Di Gerinimo L, Ferrucci F, Murolo A, Sarro F. A parallel genetic algorithm based on hadoop mapreduce for the automatic generation of junit test suites. In: Proceedings of the 5th international conference on software testing, verification and validation; 2012. p. 785–93.
[52] White DR. Cloud engineering is search based software engineering too. J Syst Softw 2013;86:2225–41.
[53] Stone M. Cross-validatory choice and assessment of statistical predictions. J R Soc Ser A 1974;36:111–4.
[54] Shatnawi R, Li W. The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process. J Syst Softw 2008;81:1868–82.
[55] Succi G, Pedrycz W, Djokic S, Zuliani P, Russo B. An empirical exploration of the distributions of the Chidamber and Kemerer object-oriented metrics suite. Empir Softw Eng 2005;10:81–103.
[56] Belsley DA, Kuh E, Welsch RE. Regression Diagnostics: identifying Influential Data and Sources of Collinearity. *USA*: *John Wiley and Sons*; 1980.
[57] Michalak K Kwasnicka H. Correlation-based feature selection strategy in neural classification. In: Proceedings of the sixth international conference on intelligent systems design and applications; 2016. p. 741–6.
[58] Hall MA.Correlation-based feature selection for discrete and numeric class machine learning, In: Proceedings of the seventeenth international conference on machine learning; 2000. p. 359–66.
[59] Mitchell BS, Traverso M, Mancoridis S. An architecture for distributing the computation of software clustering algorithms. In: Proceedings of the working conference on software architecture; 2001. p. 181–90.
[60] Mahdavi K, Harman M, Hierons RM. A multiple hill climbing approach to software module clustering In: Proceedings of the international conference on software maintenance; 2003. p. 315–24.
[61] Asadi F, Antoniol G, Gueheneuc Y. Concept locations with genetic algorithms: a comparison of four distributed architectures. In: Proceedings of the symposium on search based software engineering; 2010.
[62] Wilcoxon F. Individual comparisons by ranking methods; 1945. p. 80–3.
[63] Briand LC, Wust J, Daly JW, Porter DV. Exploring the relationships between design measures and software quality in object-oriented systems. J Syst Softw 2000;51:245–73.