Applied Intelligence
https://doi.org/10.1007/s10489-022-04427-x

Classification framework for faulty-software using enhanced exploratory whale optimizer-based feature selection scheme and random forest ensemble learning

Majdi Mafarja[1] · Thaer Thaher[2,3] · Mohammed Azmi Al-Betar[4] · Jingwei Too[5] · Mohammed A. Awadallah[6,7] · Iyad Abu Doush[8,9] · Hamza Turabieh[10]

Accepted: 23 December 2022

Abstract

Software Fault Prediction (SFP) is an important process to detect the faulty components of the software to detect faulty classes or faulty modules early in the software development life cycle. In this paper, a machine learning framework is proposed for SFP. Initially, pre-processing and re-sampling techniques are applied to make the SFP datasets ready to be used by ML techniques. Thereafter seven classifiers are compared, namely K-Nearest Neighbors (KNN), Naive Bayes (NB), Linear Discriminant Analysis (LDA), Linear Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), and Random Forest (RF). The RF classifier outperforms all other classifiers in terms of eliminating irrelevant/redundant features. The performance of RF is improved further using a dimensionality reduction method called binary whale optimization algorithm (BWOA) to eliminate the irrelevant/redundant features. Finally, the performance of BWOA is enhanced by hybridizing the exploration strategies of the grey wolf optimizer (GWO) and harris hawks optimization (HHO) algorithms. The proposed method is called SBEWOA. The SFP datasets utilized are selected from the PROMISE repository using sixteen datasets for software projects with different sizes and complexity. The comparative evaluation against nine well-established feature selection methods proves that the proposed SBEWOA is able to significantly produce competitively superior results for several instances of the evaluated dataset. The algorithms' performance is compared in terms of accuracy, the number of features, and fitness function. This is also proved by the 2-tailed P-values of the Wilcoxon signed ranks statistical test used. In conclusion, the proposed method is an efficient alternative ML method for SFP that can be used for similar problems in the software engineering domain.

Keywords Software fault prediction · Machine learning · SMOTE · Dimension reduction · Meta-heuristics · Imbalanced data

Abbreviations

AAE	Average absolute error
ABC	Artificial bee colony
ACO	Ant colony optimization
ADASYN	Adaptive synthetic sampling method
ALO	Ant lion optimizer
ANN	Artificial neural networks
ARE	Average Relative Error
ASD	Agile software development model
AUC	Area under the curve
BBAT	Binary bat algorithm
BCS	Binary cuckoo search
BFFA	Binary firefly algorithm
BGWO	Binary grey wolf optimization
BHHO	Binary harris hawk optimization
BJAYA	Binary jaya algorithm
BMFO	Binary moth flame optimization
BN	Bayesian networks
BQSA	Binary queuing search algorithm
BWOA	Binary whale optimization algorithm
CBL	Case-based learning

✉ Mohammed Azmi Al-Betar
m.albetar@ajman.ac.ae

Extended author information available on the last page of the article.

COA	Coyote optimization algorithm
CS	Chi-square
CSA	Crow search algorithm
DA	Dragonfly algorithm
DE	Differential evolution
DT	Decision tree
EA	Evolutionary algorithm
FFA	Firefly algorithm
FIS	Fuzzy inference system
FN	False negative
FP	False positive
FS	Feature selection
GA	genetic algorithm
GBRCR	Gradient boosting regression-based combination rule
GOA	Grasshopper optimization algorithm
GP	Genetic programming
GWO	Grey wolf optimizer
HHO	Harris hawks optimization
IG	information gain
KNN	K-nearest neighbors
LDA	Linear discriminant analysis
LR	Linear regression
LRCR	linear regression-based combination rule
ML	Machine learning
MLP	Multi-layer perceptron
MLR	Multi-nomial logistic regression
MVO	multiverse optimizer
NB	Naive Bayes
OO	Object-Oriented
PCA	Principle component analysis
PCC	Pearson correlation coefficient
PSO	Particle swarm optimization
QMOOD	Quality metrics for object-oriented design
RF	Random forest
ROC	Receiver operating characteristic
SBWOA	Binary whale optimization algorithm with S-shaped transfer function
SBEWOA	Enhanced SBWOA
SC	Soft computing
SDLC	Software sevelopment life cycle
SDP	Software defect prediction
SFP	Software fault prediction
SMOTE	Synthetic minority oversampling technique
SSA	Salp swarm algorithm
SVM	Support vector machine
TF	Transfer function
TN	True negative
TNR	True negative rate
TP	True positive
TPR	True positive rate
VBWOA	Binary whale optimization algorithm with V-shaped transfer function
WOA	Whale optimization algorithm

1 Introduction

Software Development Life Cycle (SDLC) represents the phases that software passes through while it is being developed. Starting with requirements elicitation, then the analysis and design of the collected requirements. After that, the programmers start developing the proposed software based on the analysis and design phases. A vital phase in SDLC is software testing. This phase follows the development phase and consists of a set of activities that assure the team is developing the right software with high-quality levels [1]. Numerous testing types are available to test various aspects of a software product. These tests include but are not limited to unit testing, component testing, integration testing, regression testing, and user acceptance testing. Many software development methodologies are available to be used by the development team. The most popular SDLC models are waterfall, agile and spiral models.

The testing stage plays an essential role in the development process. It is usually performed as a traditional linear model (e.g., waterfall) or a cyclic model (e.g., agile model). Testing process concerns with enhancing the software quality and reducing the total cost [2, 3]. However, many factors affect the results of the testing process, such as the limited resources (e.g., time or software testers). Therefore, early-stage procedures such as Software Fault Prediction (SFP) are utilized to facilitate the testing process in an optimal way [4]. In SFP, the faulty components of the software are detected prior to system deployment in the early stages of the SDLC. This is achieved by utilizing software faults datasets collected from previous projects or predefined software metrics. It is worth mentioning that the SFP process became more straightforward since the adoption of the agile software development (ASD) model in 2001 [5] as a replacement for the waterfall model which was introduced in 1970 [6]. Adopting the ASD methodology has many benefits since the software is developed incrementally. Moreover, ASD opens the door to adopting volatile requirements, optimizing resources (time and cost), bridging the gap between the development team and business owners [7], and facilitating performance software engineering tasks regularly such as review, maintenance, and testing [2].

The early prediction of faults in software components such as modules, classes, and so on has a significant impact in reducing the needed time and effort for the project outcomes to be delivered to the end-user. SFP is one of the approaches that help in optimizing the development process by reducing the number of potential faults in the early stages of the SDLC process [4]. Various SFP approaches were recorded in the literature. The main approaches include but are not limited to Soft Computing (SC) and Machine Learning (ML) [8]. These methods need data to be able to predict software faults. Design features (metrics) gathered during the design stage or historical fault datasets accumulated during the implementation of previous versions of similar projects are two essential resources to be used with SFP approaches for benchmarking [9].

Various types of metrics such as method-level and class-level have been proposed for the SFP problem [10]. Method-level metrics can be collected from structured programming or object-oriented programming-based source codes. Halstead [11], and McCabe [12] metrics are the most common method-level measures used by many researchers. Class-level metrics are only appropriate when developing SFP models for object-oriented programming-based projects. Examples of class-level suite of metrics for object-oriented design are CK (Chidamber–Kemerer) [13], L&K (Lorenz-Kidd) [14], and quality metrics for object-oriented design (QMOOD) [15]. However, in comparison with other suites, CK metric is mostly applied when class-level metrics are chosen [10].

Automated systems become available for almost all fields in real life. With the advancement of software development, and the availability of large-scale projects, analyzing the collected software metrics becomes complicated and forms a significant challenge. Thus, ML techniques have been proposed as SFP solutions and shown a good performance [16]. The main purpose behind these techniques is to predict the faulty components in software based on the supplied datasets. Examples of ML techniques that have been used as SFP approaches are K-Nearest Neighbors (KNN), Naive Bayes (NB), Linear Discriminant Analysis (LDA), Linear Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), and Random Forest (RF) [4, 17, 18].

Among the various ML models, ensemble learning has proven excellent performance in dealing with various complex classification problems [19]. Ensemble learning combines a number of ML models to create an ensemble learner to improve the model performance by proving a more general robust model. The RF is recognized as a well-regarded ensemble technique that was originally introduced by Breiman, Leo [20]. In RF, a number of DT classifiers are fit on various sub-samples of the dataset and combine the output of all the trees. The RF has several merits that make it superior when compared to other traditional ML models. It controls the over-fitting problem of DT, reduces the variance within the forest, and thus enhances the predictive accuracy [21, 22].

The performance of the ML-based SFP approaches depends mainly on a set of factors which is the applied ML technique and the quality of the utilized dataset (in terms of noise, irrelevant features, and imbalanced representation of data) [23]. Therefore, dimensionality reduction (e.g., feature selection) and data resampling (e.g., Synthetic Minority Oversampling Technique (SMOTE)) techniques are needed before applying the ML technique. These features can be defined in the context of the feature selection problem which can be tackled by feature selection techniques.

In feature selection (FS) the problems with high-dimension feature space increase the hardness of the search process. In common, various search strategies, including complete, random, and heuristic, are available for searching the feature space to obtain the optimal subset of features [24]. The complete search requires generating and evaluating all possible subsets of features. In this way, for a set of m features, 2^m features subsets will be formed. For example, if the given problem has four features, sixteen subsets of features will be produced. In case of random search, the next candidate solution (subset of features) is generated randomly while heuristic search strategies conduct the search in adaptive way, and generate possible solutions (feature subsets) for the problem [24–27].

Recently, metaheuristics is widely used by the research community as a successful FS method. The metaheuristics are conventionally categorized based on the initial solutions into population-based, and trajectory-based [28]. A trajectory-based metaheuristic is initiated with a single solution. The search follows a trajectory in the search space based on the local modification of the current solution until a local optimum is obtained. These methods like tabu search [29], β-hill climbing [30], stochastic local search [31], and variable neighborhood search [31]. In contrast, the population-based algorithm is initiated with a population of individuals. Iteratively, the population inherits its strong elements to come up with an optimal solution. Normally, population-based algorithms are classified into evolutionary algorithms (EAs) and swarm intelligence approaches. Genetic Algorithm (GA) [32], Genetic Programming (GP) [33], and Differential Evolution (DE) [34] are the base EAs for feature selection. A swarm-based algorithm is normally built based on the idea of a group of solutions where the group members are divided into leaders and followers. Particle Swarm Optimization (PSO) algorithm and Ant Colony Optimization (ACO) are the base swarm intelligence methods. Quite recently, several swarm intelligence methods have been proposed for FS such as PSO [35], Salp Swarm Algorithm (SSA) [36, 37], Dragonfly Algorithm (DA) [38], Rate Swarm Optimizer [39], Ant Lion Optimizer (ALO) [40], Harmony Search [41], Coronavirus herd immunity optimizer [42], ant colony optimization (ACO) [43], β-hill climbing optimizer [44], Crow Search Algorithm (CSA) [45], JAYA algorithm [46], Firefly algorithm (FFA) [47], Artificial Bee Colony (ABC) algorithm [48], Coyote Optimization Algorithm (COA) [49], and Grasshopper Optimization Algorithm (GOA) [50]. Furthermore, the hybrid between them such as Genetic-Whale-Ant colony algorithms [51], Grey Wolf optimizer and Random Forest [52], hybrid Salp Swarm Algorithm [53], genetic and coral reefs [54], etc. There are also real opportunities to adapt newly-established optimization algorithms for FS problems like starling murmuration optimizer [55], Quantum-based avian navigation optimizer [56], Farmland fertility algorithm [57], African vultures optimization algorithm [58], and artificial gorilla troops optimizer [59].

Whale Optimization Algorithm (WOA) is a recent swarm intelligence imitates the behavior of humpback whales in hunting fish in the oceans [60]. It has impressive characteristics over other optimization methods such as it has few control parameters, easy to implement, simple structure, and it has maneuver behavior to find a suitable balance between local exploitation and global exploration. Due to its successful attributes, WOA has been widely utilized to deal with feature selection problems [25, 61–65]. The original version of WOA was designed to handle continuous search space problems. In this paper, to match the binary search space of the FS problem, WOA was boosted with eight fuzzy transfer functions from S-shaped and V-shaped families. Due to the No Free Lunch [66] argument which points out that there is no superb optimization algorithm that can excel all others for all optimization problems, therefore, the opportunity is still possible to investigate modifying efficient methods to handle the SFP to improve the algorithm efficiency.

In this paper, a systematic SFP approach that considered several ML techniques with different pre-processing methods was proposed. The major contributions are summarized as follows:

- Several pre-processing and re-sampling techniques are applied to prepare SFP datasets to be suitable to the ML techniques.
- Various classification techniques, namely KNN, LDA, SVM, LR, DT, and NB, RF, are applied. Their performance is compared in the same environment to adopt one technique for further experiments. As a result, the RF classifier is adopted in this stage.
- A dimensionality reduction method based on the Binary version of WOA was utilized to eliminate the irrelevant/redundant features to enhance the performance of the RF classifier. The newly proposed method is called BWOA, which utilizes eight transfer functions where a transfer function that yields good results is chosen.
- An enhanced WOA version (EWOA) is introduced, where the exploration strategies from the grey wolf optimizer (GWO) and harris hawks optimization (HHO) algorithms are used to enhance the diversity of the WOA. By means of this enhancement mechanism, the performance of WOA is improved to deal more efficiently with the search space of the FS problem. This yields a superior optimization framework for the faulty-software prediction problem.

The newly proposed EWOA reveals very successful outcomes in terms of choosing the most informative features in the area of SFP. The findings prove that the classification performance can be significantly improved by removing useless features. The performance is compared with nine state-of-the-art methods and it shows the viability of the proposed method in terms of the accuracy, number of features, and fitness function.

The remainder of the paper is structured as follows: a review of the related works is presented in Section 2. In Section 3, a theoretical background of the related aspects to this paper is introduced. Section 4 presents the proposed methodology. The experimental design and the obtained results are discussed in Section 5. Finally, Section 6 includes a conclusion about the main findings of this paper in addition to some future work directions.

2 Related works

Recently, different ML approaches were considered to solve the SFP problem with remarkable success [4]. Accordingly, different datasets (e.g., PROMISE repository, NASA datasets, and Qualitas corpus) became publicly available to the researchers [4, 67]. This section presents the most relevant related work in the field of SFP. A general overview of the SFP techniques is provided, and the related ML approaches are investigated, followed by the related work of the enhanced ML approaches by applying some preprocessing approaches like feature selection.

2.1 ML based SFP

Different supervised and unsupervised ML techniques were applied as prediction models in SFP. Examples of the ML that were used with SFP are: SVM [68], DT [69], Bayesian Networks (BN) [70], NB [71], KNN [72], Multi-layer Perceptron (MLP) [73], Artificial Neural Networks (ANN) [2, 74], LR [75], Multi-nomial Logistic Regression (MLR) [73], RF [76] and ensemble MLP [77].

Singh and Malhotra [68] conducted an empirical study to evaluate the performance of an SVM classifier in determining the relationship between some software Object-Oriented (OO) design matrices and fault proneness. A dataset from the NASA repository (KC1) and Receiver Operating Characteristic (ROC) were used to evaluate the proposed model. The study shows that the SVM classifier was feasible and helpful in predicting faulty classes in OO-based systems.

Moreover, Cahill et al. [78] introduced an approach named Rank Sum for data representation to improve the performance of fault porousness prediction modules. The proposed approach is evaluated by applying the well-known ML classifiers SVM and NB over various datasets from the NASA repository. It was found that NB is better compared to the SVM classifier. Erturk and Sezer [79] introduced an SFP model that combines Fuzzy Inference System (FIS) and Artificial Neural Network (ANN) classifiers. FIS was applied at the beginning of the project to make predictions depending on expert opinion because it does not need historical data for prediction, and then ANN was employed in the later iterations when some data about the software project are obtainable. The proposed iterative system was tested using a set of datasets including various versions of many projects from the PROMISE repository. The selected datasets consist of common OO metrics such as coupling between objects, response for a class, and weighted methods per class. The evaluation of the results according to the receiver operating characteristics (ROC) with the area under the curve (AUC) method shows that the iterative module is capable of locating fault-prone modules in the software.

An approach named multi-strategy classifier (RB2CBL) was introduced by Khoshgoftaar et al. [80] for the SFP problem, where Rule-Based (RB) classifier was hybridized with two variants of the Case-Based Learning (CBL) model. Moreover, an embedded GA was utilized to optimize the parameters of CBL models. The experimental results reveal that the proposed RB2CBL classifier is superior compared to the RB model alone. Carrozza et al. [81] proposed a new set of software matrices for detecting mandelbugs in complex software systems. In addition, considering the newly proposed matrices and the conventional software matrices, several algorithms, including DT, SVM, BN, NB, and MLR, were applied to various datasets from the NASA repository. The authors reported that MLR and SVM are the best among all examined algorithms in finding Mandelbug-prone modules.

A model based on the principle of ensemble learning methods was employed by Rathore and Kumar [82] to predict software faults in which linear regression-based combination rule (LRCR) and gradient boosting regression-based combination rule (GBRCR) approaches were used to ensemble the output of Genetic Programming (GP), MLP, LR algorithms. Moreover, eleven datasets belonging to six software projects were accumulated from the PROMISE data repository to assess the performance of the proposed ensemble models. Results of different performance evaluation measures, including Average Absolute Error (AAE) and Average Relative Error (ARE), provided evidence that ensemble techniques can produce better results for predicting software faults compared to individual fault prediction techniques. Choudhary et al. [83] defined a set of change matrices in addition to the existing ones to enhance the performance of SFP modules.

Various ML classifiers were applied along with code matrices and change matrices. Experimental results on different releases of Eclipse projects demonstrate that the newly introduced change matrices can improve the performance of fault prediction modules. In [84], Shatnawi used the ROC analysis to examine the relationship between software matrices (features) and faults where threshold values of matrices were identified accordingly. A threshold value is defined for each metric to be used for deciding whether a software module is faulty or not. Moreover, the results of ROC were also considered for selecting the most correlated matrices with faults. Only selected matrices were applied to train and test a set of ML classifiers (LR, NB, KNN, and decision trees C4.5).

From the previously investigated related work, researchers confirmed that having a considerable number of features in a dataset affects the performance of the ML technique. Therefore, many researchers considered dimensionality reduction methods to eliminate the irrelevant/redundant features from the datasets. The most popular dimensionality reduction technique is FS.

2.2 Preprocessing ML methods

FS is a well-known preprocessing step in the data mining process that aims to eliminate noisy, irrelevant, and redundant features to reduce data dimensionality, and hence, improve the performance of the employed ML technique [24, 85]. In many works in the field of SFP, different filter, and wrapper FS approaches were investigated. Catal, C. and Diri, B. [86] employed a correlation-based feature selection approach to select the highly relevant matrices with varying techniques of ML (i.e., RF, DT, NB, and AIRS). They found that FS positively affected the performance of the employed ML approaches and that the RF classifier outperformed other classification techniques. In [87], eighteen filter FS methods were employed on five datasets from the NASA repository with various classification techniques. The obtained results revealed that using FS enhanced the performance of the prediction models.

As presented in [88], a set of filter FS methods, including Chi-square (CS), information gain (IG), and Pearson Correlation Coefficient (PCC), was used to develop a hybrid feature selection method to improve the performance of Software Defect Prediction (SDP). In the hybrid FS method, the features were ranked and selected according to their values using these filter ranking methods. In addition, for