Applied Soft Computing 27 (2015) 504–518

Contents lists available at ScienceDirect

Applied Soft Computing

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

A systematic review of machine learning techniques for software fault prediction

Ruchika Malhotra

Department of Software Engineering, Delhi Technological University, Bawana Road, Delhi, India

ARTICLE INFO

Article history:
Received 31 December 2013
Received in revised form 8 October 2014
Accepted 22 November 2014
Available online 29 November 2014

Keywords:
Machine learning
Software fault proneness
Systematic literature review

ABSTRACT

Background: Software fault prediction is the process of developing models that can be used by the software practitioners in the early phases of software development life cycle for detecting faulty constructs such as modules or classes. There are various machine learning techniques used in the past for predicting faults.
Method: In this study we perform a systematic review of studies from January 1991 to October 2013 in the literature that use the machine learning techniques for software fault prediction. We assess the performance capability of the machine learning techniques in existing research for software fault prediction. We also compare the performance of the machine learning techniques with the statistical techniques and other machine learning techniques. Further the strengths and weaknesses of machine learning techniques are summarized.
Results: In this paper we have identified 64 primary studies and seven categories of the machine learning techniques. The results prove the prediction capability of the machine learning techniques for classifying module/class as fault prone or not fault prone. The models using the machine learning techniques for estimating software fault proneness outperform the traditional statistical models.
Conclusion: Based on the results obtained from the systematic review, we conclude that the machine learning techniques have the ability for predicting software fault proneness and can be used by software practitioners and researchers. However, the application of the machine learning techniques in software fault prediction is still limited and more number of studies should be carried out in order to obtain well formed and generalizable results. We provide future guidelines to practitioners and researchers based on the results obtained in this work.

1. Introduction

The growing complexity and dependency of the software has increased the need for delivering high quality, maintainable software at lower costs. Software fault prediction is very important and essential activity in order to improve the software quality and reduce the maintenance effort before the system is deployed [1]. Early detection of faults may lead to timely correction of these faults and delivery of maintainable software. There are various software metrics available in the literature. These software metrics and fault data can be used to construct models that can be used for predicting faulty modules/classes in the early phases of software development life cycle.

Software fault prediction (SFP) can be done by classifying the modules/classes as fault prone and not fault prone. The software metrics and fault data obtained from a similar project or previous release can be used to construct SFP models. The model constructed can be subsequently applied to the modules/classes of current software projects for classifying them as fault prone or not fault prone. The software practitioners can then target the available testing resources on the fault prone areas of the software in the early phases of software development. For example, if only 30% of testing resources are available, the knowledge of the weaker areas will help the testers in focusing the available resources on fixing the classes/modules that are more prone to faults. Hence, a low cost, high quality and maintainable software can be produced in the given time and budget.

Soft computing techniques are well suited for real life problems that use methods to extract useful information from complex and intractable problems in less time. They are tolerant to data that is imprecise, partially incorrect or uncertain. One of the important components of the soft computing techniques includes the machine learning techniques. The machine learning (ML) techniques have been used in the literature in order to predict models for estimating the faulty modules/classes. For example, Singh et al. [2] have

E-mail address: ruchikamalhotra2004@yahoo.com

http://dx.doi.org/10.1016/j.asoc.2014.11.023
1568-4946/

employed decision trees and artificial neural networks in order to predict classes at various severity levels of faults. There are several advantages of using these classification techniques for predicting faulty modules/classes.

To facilitate the use of ML techniques in software fault prediction it is necessary to systematically summarize the empirical evidence obtained on these techniques from the existing literature and studies. The existing literature reviews aim to answer various research questions [3–5], but to the best of the authors knowledge there is no systematic review that focus on the ML techniques for the software fault prediction i.e. whether a module/class is faulty or not. This paper extensively reviews all the studies between the period of 1991 and 2013. In order to perform the review we have identified seven categories of the ML techniques. The aim of this systematic literature review is to summarize, analyze and assess the empirical evidence regarding: (1) ML techniques for SFP models (2) performance accuracy and capability of ML techniques for constructing SFP models (3) comparison between the ML and statistical techniques (4) comparison of performance accuracy of different ML techniques (5) summarize the strength and weakness of the ML techniques. We further provide future guidelines to software practitioners and researchers regarding the application of the ML techniques in SFP. In order to achieve this aim we extensively searched through seven digital libraries and identified 64 studies to answer the research questions pertaining to the use of the ML techniques for the SFP. The primary studies were selected according to the quality assessment of the studies and relevance.

The rest of the paper is organized as follows: Section 2 presents the research questions that are addressed in this systematic review and the research criteria followed in this study for selection of primary studies. Section 3 presents the answers to the research questions identified in this work. Section 4 provides the limitation of this work and Section 5 provides conclusions and future directions obtained from this systematic review.

2. Method

The planning, conducting and reporting of the systematic review carried out in this paper is done by following the procedure given by [6]. The process is depicted in Fig. 1. In the planning stage we developed the review protocol that included the following steps: research questions identification, search strategy design, study selection criteria, study quality assessment, data extraction process and data synthesis process. After formation of the review protocol a series of steps were carried out in the review. In the first step we formed the research questions that addressed the issues to be answered in the systematic literature review (SLR). In the second step, we described the search strategy including identification of search terms and selection of sources to be searched in order to identify the primary studies. The third step involves determination of relevant studies based on the research questions. This step also determines the inclusion and exclusion criteria for each primary study. In the next step, we identify quality assessment criteria by forming the quality assessment questionnaire in order to analyze and assess the studies. The second last step involves the design of data extraction forms to collect the required information in order to answer the research questions and in the last final step we devise methods for data synthesis.

Development of review protocol is an important step in a SLR as it reduces the possibility and risk of research bias in the SLR. We established the review protocol in this work by frequently holding meetings and group discussions in the group formed comprising of senior Assistant Professors. In the following subsections we describe the research questions and the steps followed while conducting the SLR.

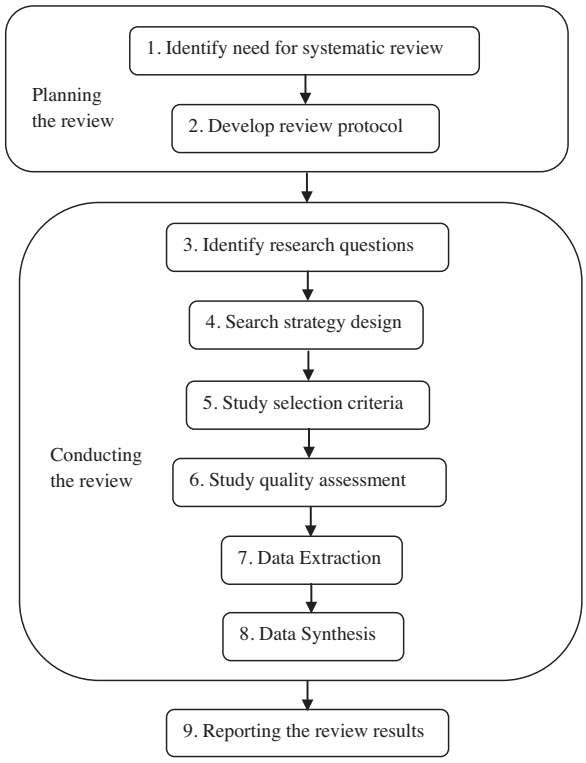


Fig. 1. Systematic review process.

2.1. Research questions

The aim of this SLR is to provide and assess the empirical evidence obtained from the studies using the ML techniques for SFP in the literature. Table 1 presents six research questions addressed in this SLR. From the primary studies we identified the ML techniques used for predicting software faults (RQ1). We analyzed these studies using the ML techniques for SFP to answer RQ2, RQ2.1, RQ2.2, RQ2.3, RQ2.4, RQ2.5, RQ2.6 and RQ2.7. In the next research question, we analyzed the performance of the ML techniques for SFP (RQ3). This question focuses on the values of the performance measures for SFP. RQ4 compares the performance of ML techniques with statistical techniques. The aim of this question is to determine whether ML techniques are better than the statistical techniques. RQ5 assesses which ML techniques outperformed the other ML techniques in order to determine which ML technique is consistently better than the other ML techniques. The final question (RQ6) determines the strengths and weakness of different ML techniques to provide software practitioners guidance regarding the selection of an appropriate ML technique.

2.2. Search strategy and study selection

We formed the sophisticated search terms by incorporating alternative terms and synonyms using Boolean expression 'OR' and combining main search terms using 'AND'. The following general search terms were used for identification of primary studies:

Software AND (fault OR defect OR error) AND (proneness OR prone OR prediction OR probability) AND (regression OR machine learning OR soft computing OR data mining OR classification OR Bayesian network OR neural network OR decision tree OR support vector machine OR genetic algorithms OR random forest)

The terms on ML were derived from textbooks and research papers on ML [7,8]. After identifying the search terms, the relevant and important digital portals were selected. The selection was not restricted by the availability of digital portals at the home

Table 1
Research questions.

RQ#	Research questions	Motivation
RQ1	Which ML techniques have been used for SFP?	Identify the ML techniques commonly being used in SFP.
RQ2	What kind of empirical validation for predicting faults is found using the ML techniques found in RQ1?	Assess the empirical evidence obtained.
RQ2.1	Which techniques are used for feature reduction for SFP?	Identify techniques reported to be appropriate for selecting and extracting relevant features.
RQ2.2	Which metrics are commonly used for SFP?	Identify metrics commonly used for SFP.
RQ2.3	Which metrics are found useful for SFP?	Identify metrics reported to be appropriate for SFP.
RQ2.4	Which metrics are found not useful for SFP?	Identify metrics reported to be inappropriate for SFP.
RQ2.5	Which data sets are used for SFP?	Identify data sets reported to be appropriate for SFP.
RQ2.6	What is the fault distribution in the data sets used for SFP?	Identify the fault distribution in the data sets used for SFP.
RQ2.7	Which performance measures are used for SFP?	Assess the performance of the ML techniques for SFP.
RQ3	What is the overall performance of the ML techniques for SFP?	Investigate the performance of the ML techniques for SFP.
RQ4	Whether the performance of the ML techniques better than the statistical techniques?	Investigate the performance of the ML techniques over statistical techniques for SFP.
RQ5	Are there any ML techniques that significantly outperform other ML techniques?	Investigate the performance of the ML techniques over the other ML techniques for SFP.
RQ6	What are the strengths and weaknesses of the ML techniques?	Determine the information about ML techniques.

universities. The following seven electronic databases were used for searching the primary studies:

(1) IEEE Xplore
(2) ScienceDirect
(3) ACM Digital Library
(4) Wiley Online Library
(5) Google scholar
(6) SpringerLink
(7) Web of Science

The search was made on the above seven electronic databases using the target search string. In addition to searching the above electronic databases the relevant journals and conferences listed on http://web.engr.illinois.edu/ taoxie/seconferences.htm were searched.

We restricted the search from 1991 to 2013 as the machine learning techniques were launched in the 1990s. After determining which electronic databases to search, an initial search to identify the candidate primary studies was performed. After performing an initial search, the relevant studies were determined by obtaining the full text papers following the inclusion and exclusion criteria described in the next section. We also included those studies that were found important from the reference section of the relevant studies.

We included the empirical studies using the ML techniques for SFP in the SLR. We identified 96 primary studies for inclusion in the SLR. After scanning the reference lists of the relevant study further 26 studies were included. Hence, total number of 122 studies [1,2,20–139] were identified for further processing and analysis. The candidate studies were selected after following the inclusion and exclusion criteria given below:

Inclusion criteria:

- Empirical studies using the ML techniques for SFP.
- Empirical studies combining the ML and non ML techniques.
- Empirical studies comparing the ML and statistical techniques.

Exclusion criteria:

- Studies without empirical analysis or results of use of the ML techniques for SFP.
- Studies based on dependent variable other than fault proneness.
- Studies using the ML techniques in context other than SFP.
- Similar studies i.e. studies by same author in conference as well extended version in journal. However, if the results were different in both the studies they were retained.
- Review studies

The above inclusion and exclusion criteria were tested by two researchers independently and they reached a common decision after detailed discussion. In case of any doubt, full text of the study was reviewed and a final decision regarding the inclusion/exclusion was made. In addition, the quality of the studies was determined by their relevance to the research questions. The similar studies with the same results by a particular author were removed.

By applying the above selection criteria, 79 studies were selected. Finally, the quality assessment criteria given in the following section was used to obtain the final studies.

2.3. Quality assessment criteria

We formed a quality questionnaire for assessing the relevance and strength of the primary studies. The quality questionnaire was developed by considering the suggestions given by Wen et al. [9]. The quality assessment is used for weighing the studies. Table 2 presents the quality assessment questions. The questions are ranked 1 (yes), 0.5 (partly) and 0 (no). The final score is obtained after adding the values assigned to each question. A study could have maximum score of 15 and minimum score of 0.

Two independent researchers ranked the quality questions for each primary question and consulted other researchers in case of any disagreement. Finally after thorough reviews, discussions and brain storming sessions a final decision about the inclusion/exclusion for each study was made.

2.4. Data extraction and data synthesis

We filled a form for each of the selected primary study (step 7). The purpose of using data extraction form is to determine which research question was satisfied by a primary study. We summarized author name, title, publishing details, data set details, independent variables (metrics) and the ML techniques. The details of which specific research questions answered by each primary study were present in the data extraction card. These data extraction cards were used to collect information from the primary studies. Two independent researchers collected the information required for each primary study from the data extraction card. The two researchers then matched their results and if there was any disagreement among the two, other researchers were consulted to

Table 2
Quality assessment questions.

Q#	Quality questions	Yes	Partly	No
Q1	Are the aims of the research clearly stated?			
Q2	Are the independent variables clearly defined?			
Q3	Is the data set size appropriate?			
Q4	Is the data collection procedure clearly defined?			
Q5	Are metrics sub selected?			
Q6	Are the ML techniques justified?			
Q7	Are the ML techniques clearly defined?			
Q8	Are the performance measures used to assess the SFP models clearly defined?			
Q9	Are the results and findings clearly stated?			
Q10	Are the limitations of the study specified?			
Q11	Is the research methodology repeatable?			
Q12	Is there any comparative analysis conducted (statistical vs ML)?			
Q13	Is there any comparative analysis conducted (ML vs ML)?			
Q14	Does the study contribute/add to the literature?			
Q15	Does the study have adequate number of the average citation count per year?			

resolve these disagreements. The resultant data was saved into a excel file for further use during the data synthesis process.

The basic objective while synthesizing data is to accumulate and combine facts and figures from the selected primary studies in order to formulate a response and resolve the research questions [9]. Collection of a number of studies which state similar and comparable viewpoints and results helps in providing research evidence for obtaining conclusive answers to the research questions. We scrutinized and evaluated both the quantitative data which includes values of various performance metrics like AUC, prediction accuracy and qualitative data which includes strengths and weaknesses of the ML methods, categorization of various ML methods, feature sub-selection methods and data sets used. We utilize a number of techniques to synthesize data collected from our primary studies. In order to answer the research questions we used visualization techniques such as line graph, box plots, pie charts and bar charts. We also used tables for summarizing and presenting the results.

3. Results and discussion

This section presents the results obtained from primary studies selected for SFP. First, we present the overview of each primary study along with their corresponding ranking based on the scores assigned to the quality assessment questions. Second, the answers to the review research questions are provided in each sub section. Then we provide the discussion and interpretation of the results in the view of the facts obtained from each primary study.

3.1. Description of primary studies

In this section, we provide the brief description of the selected primary studies. We have selected 64 primary studies, out of 79 studies (based on the quality questions), which use the ML techniques for SFP. The studies were mostly based on public or open source data sets.

Table 3
Summary of top publications.

Publication name	Type	Number	Percent
IEEE Transactions on Software Engineering	Journal	9	16
Journal of Systems and Software	Journal	6	10
Empirical Software Engineering	Journal	6	10
Information and Software Technology	Journal	3	5
IEEE International Symposium on Software Reliability (ISSRE)	Conference	3	5
International Conference on Predictor Models in Software Engineering (PROMISE)	Conference	2	4
Software Quality Journal	Journal	2	4
Automated Software Engineering	Journal	2	4
Information Sciences	Journal	2	4
Expert Systems with Applications	Journal	1	2
Engineering Applications of Artificial Intelligence	Journal	1	2
Product Focus Software Process Improvement	Journal	1	2
IEEE Software	Journal	1	2
International Journal of Systems Assurance Engineering and Management	Journal	1	2

3.1.1. Publication source

Table 3 summarizes the details of the publications in top journals and conferences along with the number and percentage of primary studies in the corresponding journal. The major publications were in IEEE Transactions on Software Engineering, Information and Software Technology, Journal of Systems and Software, Software Quality Journal, Empirical Software Engineering, Information Sciences and so on. The statistics show that 56% of the primary studies were present in journals and 44% of the primary studies were present in conferences. Hence, the number of studies published in journals (36) was slightly more than published in conferences (28). The top 14 sources consisted of 72% of the primary studies. Table 3 shows that 16% of studies were published in IEEE Transactions on Software Engineering and 10% each in Information and Software Technology and Empirical Software Engineering.

3.1.2. Quality assessment questions

The scores were assigned to the quality assessment questions. We divided the scores into various categories very high ($13.5 \leq \text{score} \leq 15$), high ($9.5 \leq \text{score} \leq 13$), medium ($5 \leq \text{score} \leq 9$) and low ($0 \leq \text{score} \leq 4.5$). The maximum score that could be assigned to a study was 15 and minimum 0.

Table 4 provides a unique identifier to each selected primary study along with the reference. These unique identifiers will be used in all subsequent sections to refer to their corresponding selected primary study. We also wanted to assess the studies based on the visibility they have gained in the community. However, recognizing that the recent papers published in 2013 may not have yet obtained high citations, we assessed three papers published in 2013 (SF62–SF64) excluding quality question Q15. Studies SF12, SF4, SF23, were the top three rankers in terms of average citations count per year. It is also worth mentioning that all of these papers have been published in the IEEE Transactions on Software Engineering. Five studies (SF10, SF23, SF27, SF45, SF64) obtained maximum score. Hence, the interested readers may take these publications into consideration for further reading. Studies SF7, SF9, SF20, SF23, SF25, SF29, SF32, SF36, SF37, SF44, SF45, SF48, SF49, SF57, SF59, SF60, SF61, SF63, SF64 that compared the performance of the ML techniques with statistical techniques were selected in the SLR. The studies with score less than 9.5 were excluded from the analysis. Hence, after quality assessment further 15 studies

Table 4
Selected primary studies.

Study no.	Paper	Ref no.	Study no.	Paper	Ref no.
SF1	Sherer (1995)	[20]	SF33	Singh (2009b)	[51]
SF2	Guo (2003)	[21]	SF34	Tosun (2009)	[52]
SF3	Guo (2004)	[22]	SF35	Zimmermann (2009)	[53]
SF4	Gyimothy (2005)	[23]	SF36	Afzal (2010)	[54]
SF5	Koru (2005)	[24]	SF37	Arisholm (2010)	[55]
SF6	Zhou (2006)	[25]	SF38	Carvalho (2010)	[56]
SF7	Arisholm (2007)	[26]	SF39	Liu (2010)	[57]
SF8	Catal (2007)	[27]	SF40	Malhotra (2010)	[58]
SF9	Jiang (2007)	[28]	SF41	Ostrand (2010)	[59]
SF10	Kanmani (2007)	[29]	SF42	Pendharkar (2010)	[60]
SF11	Li (2007)	[30]	SF43	Seliya (2010)	[1]
SF12	Menzies (2007)	[31]	SF44	Singh (2010)	[61]
SF13	Ma (2007)	[32]	SF45	Zhou (2010)	[62]
SF14	Pai (2007)	[33]	SF46	Azar (2011)	[63]
SF15	Turhan (2007)	[34]	SF47	Diri (2011)	[64]
SF16	Turhan (2007a)	[35]	SF48	Malhotra (2011)	[65]
SF17	Carvalho (2008)	[36]	SF49	Martino (2011)	[66]
SF18	Elish (2008)	[37]	SF50	Mishra (2011)	[67]
SF19	Gondra (2008)	[38]	SF51	Misirh (2011)	[68]
SF20	Jiang (2008)	[39]	SF52	Ricca (2011)	[69]
SF21	Kaur (2008)	[40]	SF53	Rodriguez (2011)	[70]
SF22	Kim (2008)	[41]	SF54	Song (2011)	[71]
SF23	Lessmann (2008)	[42]	SF55	Twala (2011)	[72]
SF24	Menzies (2008)	[43]	SF56	Chen (2012)	[73]
SF25	Moser (2008)	[44]	SF57	Malhotra (2012)	[74]
SF26	Turhan (2008)	[45]	SF58	Okutan (2012)	[75]
SF27	Vandecruys (2008)	[46]	SF59	Yang (2012)	[76]
SF28	Bener (2009)	[47]	SF60	Yu (2012)	[77]
SF29	Catal (2009)	[48]	SF61	Zhou (2012)	[78]
SF30	Menzies (2009)	[49]	SF62	Cahill (2013)	[79]
SF31	Singh (2009)	[50]	SF63	Chen (2013)	[80]
SF32	Singh (2009a)	[2]	SF64	Dejaeger (2013)	[81]

[96,100–102,108,118–121,125,127,132–135] were removed from the analysis.

The details of the studies addressing each specific research question are provided in Table A.1 in Appendix.

3.1.3. Publication year

Fig. 2 presents the distribution of the studies from the year 1991 to 2013. Fig. 2 shows that before 2007 out of 64 only 6 studies examined the use of the ML techniques for SFP. Most of the studies were conducted from 2007 onwards. The number of studies in the years 2007, 2008, 2009, 2010 and 2011 were 10, 11, 8, 10 and 10, respectively accounting about 90% of the studies between 2007 and 2013. For year 2013, data is not complete as the search was started in October 2013 and only few studies were online till then. The availability of the public domain data sets from the promise repository from 2005 made a significant impact on the increase in the studies. Thus, since year 2007 the studies in the literature have started examining the effectiveness of the ML techniques for SFP. This confirms the inclusion of recent and relevant studies in this SLR. This also shows that the studies have recently started using the ML techniques for SFP and this research domain is relatively new.

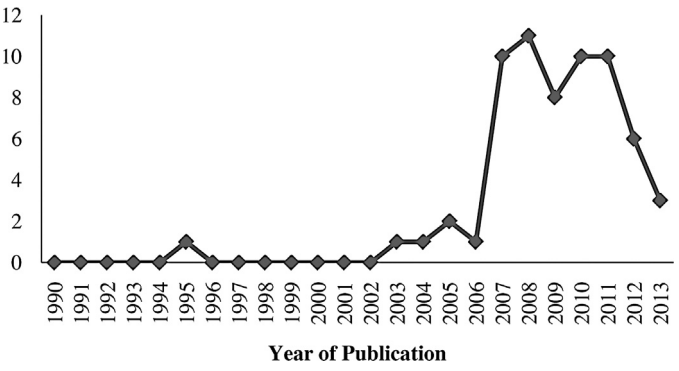


Fig. 2. Year-wise distribution of studies.

3.1.4. RQ1: which ML techniques have been used for SFP?

This section provides the details of the ML techniques used in the primary studies selected in this SLR. Based on the selected studies we categorized the ML techniques used for defect prediction as follows:

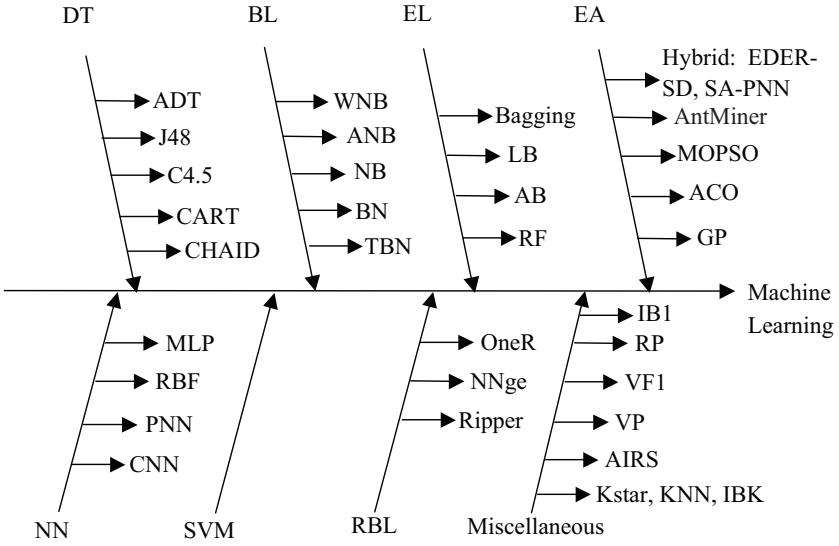
- Decision trees (DT)
- Bayesian learners (BL)
- Ensemble learners (EL)
- Neural networks (NN)
- Support vector machines (SVM)
- Rule based learning (RBL)
- Evolutionary algorithms (EA)
- Miscellaneous

Fig. 3 presents the classification taxonomy of ML techniques for SFP used in this SLR.

Table 5 presents the number and percentage of the studies with respect to the ML techniques. Among the above listed categories of

Table 5
Distribution of studies across ML techniques based on classification.

Method	# of Studies	Percent
DT	31	47.7
NN	17	26.16
SVM	18	27.7
BL	31	47.7
EL	12	18.47
EA	8	12.31
RBL	5	7.7
Misc.	16	24.62



*CHAID: **CH**i-squared **A**utomatic **I**nteraction **D**etection, CART: Classification and Regression Trees, ADT:* Alternating Decision Tree*, RF: Random Forest, NB: Naïve Bayes, BN: Bayesian Networks, ABN: Augmented Bayesian Networks, WNB: Weighted Bayesian Networks, TNB: Transfer Bayesian Networks, MLP: Multilayer Perceptron, PNN: Probabilistic Neural Network, RBF: Radial Basis Function, LB: Logit Boost, AB: AdaBoost, NNGE: Neighbor With Generalization, GP: Genetic Programming, ACO: Ant Colony Optimization, SVM: Support Vector Machines, RP: Recursive Partitioning, AIRS: Artificial Immune System, KNN: K-Nearest Neighbor, VFI: Voting Feature Intervals, EDER-SD: Evolutionary Decision Rules For Subgroup Discovery, SA-PNN: Simulated Annealing Probabilistic Neural Network, VP: Voted Perceptron*

Fig. 3. Classification of ML Techniques for SFP. CHAID: Chi-squared automatic interaction detection, CART: classification and regression trees, ADT: alternating decision tree, RF: random forest, NB: naïve Bayes, BN: Bayesian networks, ABN: augmented Bayesian networks, WNB: weighted Bayesian networks, TNB: transfer Bayesian networks, MLP: multilayer perceptron, PNN: probabilistic neural network, RBF: radial basis function, LB: logit boost, AB: AdaBoost, NNGE: neighbour with generalization, GP: genetic programming, ACO: ant colony optimization, SVM: support vector machines, RP: recursive partitioning, AIRS: artificial immune system, KNN: K-nearest neighbour, VFI: voting feature intervals, EDER-SD: evolutionary decision rules for subgroup discovery, SA-PNN: Simulated annealing probabilistic neural network, VP: Voted perceptron

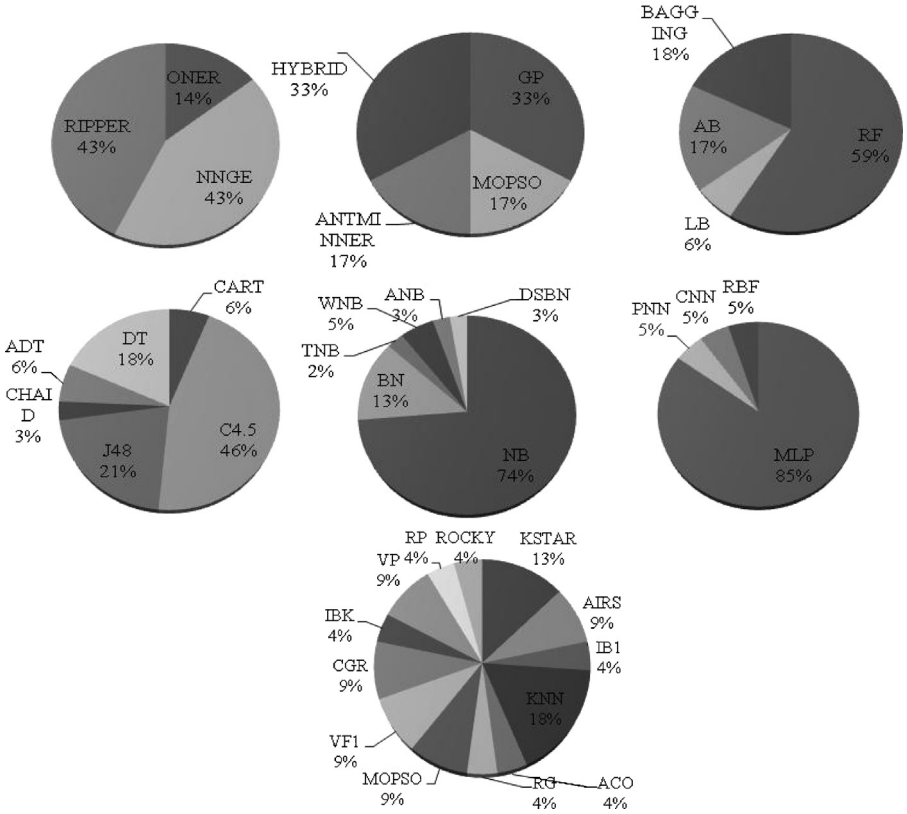


Fig. 4. Distribution of sub categories in (a) DT, (b) BL, (c) NN, (d) RL, (e) EA, (f) EL, (g) Misc.

the ML techniques the most frequently used techniques were from categories DT, BL, SVM and NN analyzed in 47.7%, 47.7%, 27.7%, 26.16% studies, respectively. These ML techniques were used for SFP. The primary studies consisted of two types of studies: (1) studies comparing statistical and ML techniques (2) studies comparing ML techniques. The distribution of the ML techniques for SFP with respect to each identified category is presented in Fig. 4. The most frequently used the ML techniques for SFP were C4.5 in DT category (46%), NB in BL category (74%), MLP in NN category (85%) and RF in EL category (59%). The top five techniques included C4.5, NB, MLP, SVM and RF which were assessed in 15, 28, 17, 19, 10 studies, respectively.

3.2. RQ2: what kind of empirical validation for predicting faults is found using ML techniques found in RQ1?

This section determines the techniques used for metrics reduction, commonly used metrics, useful and not useful metrics, nature of data sets used, and the performance measures used in the selected primary studies.

3.2.1. Which techniques are used for feature reduction for SFP using ML techniques?

Over the years, a number of techniques have been used in order to reduce the dimensionality of features and develop models with better results. There are two basic kinds of techniques for feature reduction: feature extraction and feature selection. While feature selection helps in selection of the most relevant features, feature extraction involves combination of a set of relevant features into a new smaller feature set. However, only 49% of the selected primary studies used a feature reduction technique to aid its model development.

The most commonly used feature selection technique in the selected studies was co-relation based feature selection (CFS) [10] (used by SF18, SF21, SF37, SF38, SF48, SF62, SF53, SF55, SF58). The CFS technique eliminates noisy and redundant features and retains only those features, which are highly co-related with the fault proneness attribute. Apart from CFS, some other commonly used feature selecting techniques were infogain method (used by SF12, SF15, SF31, SF50, SF51), χ^2 based filter (used by SF27, SF50), correlation analysis (used by SF14) and wrapper attribute selection (used by SF53, SF54, SF62). Some less commonly used feature selection techniques include Markov Blanket feature selection (used by SF64), consistency based selection (used by SF53, SF62), chi-square feature selection (SF22) and random forests (used by SF3). While majority of the studies used a feature selection technique, few studies used principal component analysis for feature extraction (SF10, SF15, SF19, SF28, SF44). Another technique called Isomap was also used for feature extraction by SF15.

3.2.2. Which metrics are commonly used in SFP?

In order to predict fault proneness, the primary studies use software metrics as independent variables. There are a number of metrics used in software engineering domain to quantify the characteristics of a software. We define the categorization of the selected primary studies based on the type of metrics used by these studies for predicting fault proneness as follows. Similar categorisation is given in Radjenovic et al. [3].

- *Studies using procedural metrics*: These studies use metrics that include the traditional static code metrics which are defined by Halstead [11] and McCabe [12] along with the size metrics such as the LOC (lines of code) metric. Studies which are included in this category are SF2, SF3, SF11, SF12, SF13, SF15, SF16, SF18, SF19, SF20, SF22, SF23, SF24, SF25, SF27, SF28, SF29, SF30, SF31, SF39, SF42, SF43, SF44, SF47, SF50, SF51, SF53, SF54, SF55, SF56, SF60, SF62, SF64.
- *Studies using object-oriented metrics*: These studies use metrics that measure various attributes of object-oriented (OO) software like cohesion, coupling and inheritance for an OO class [13–15]. Though, LOC is a traditional metric, many studies also include LOC or SLOC (source lines of code) while using OO metrics. For the purpose of simplicity, we include size metrics in both the categories (procedural as well as OO). Studies which are included in this category are SF4, SF5, SF6, SF7, SF8, SF10, SF14, SF17, SF21, SF32, SF33, SF40, SF45, SF46, SF48, SF49, SF57, SF58.
- *Studies using hybrid metrics*: Some studies (SF26, SF38, SF47, SF59, SF61) used both OO metrics as well as procedural metrics for fault proneness prediction.
- *Miscellaneous metrics*: These studies include metrics such as requirement metrics (SF9), change metrics (SF25), network metrics extracted from dependency graph (SF34), churn metrics (SF35), fault slip through metrics (SF36), process metrics (SF37), file age, size, changes and faults in previous versions metrics (SF41), elementary design evolution metrics (SF52), and other miscellaneous metrics which cannot be grouped as either procedural or OO metrics.

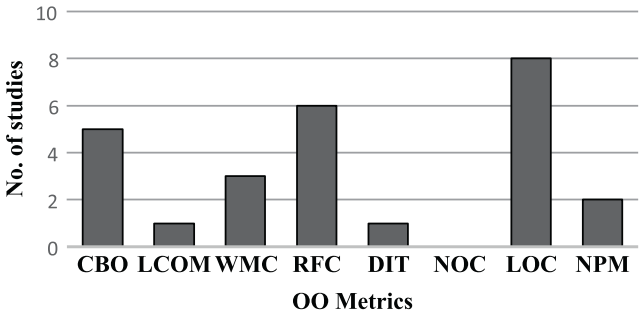


Fig. 5. Useful OO metrics for SFP.

The procedural metrics are most commonly used metrics (51%) in the selected primary studies. A number of OO metrics have also been used for SFP and the most commonly used metrics suite was defined by Chidamber and Kemerer [13] which was used in all the 28% studies which use OO metrics.

3.2.3. Which metrics are found useful for SFP?

Certain studies (SF7, SF8, SF14, SF18, SF21, SF27, SF37 SF48 etc.) report OO metrics which are highly correlated to fault proneness. Fig. 5 reports the OO metrics which are found useful for SFP. According to the figure, CBO (coupling between objects), RFC (response for a class) and LOC are highly useful metrics for SFP using feature selection methods. However, studies using feature extraction techniques have not clearly specified useful metrics for SFP.

As far as the usefulness of procedural metrics is concerned, the studies do not yield a conclusive result as the primary studies do not compare the usefulness of specific procedural metrics. Only SF38 comments about unique operand and EV(g) as useful indicators for SFP.

3.2.4. Which metrics are not found useful for SFP?

While CBO, RFC and LOC are good predictors for fault proneness, there are certain metrics which were not useful for SFP. The results obtained from the primary selected studies indicate that the NOC (number of children) and DIT (depth of inheritance tree) as not useful metrics for SFP. Studies which support this are SF8, SF14 and SF58. The primary studies do not give any conclusive results for procedural metrics which are not useful for SFP as these studies have not stated and analyzed usefulness of specific procedural metrics.

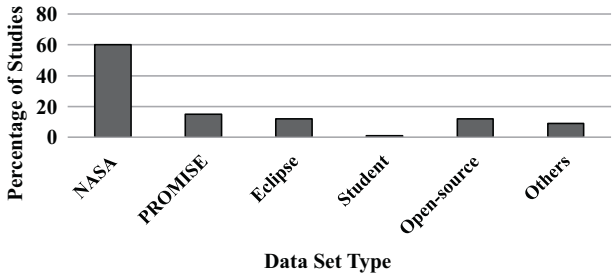


Fig. 6. Data sets for SFP.

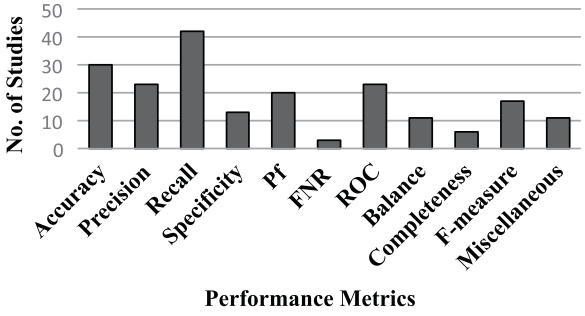


Fig. 8. Studies using different performance measures for SFP.

3.2.5. Which data sets are used for SFP?

A variety of data sets have been used in the SFP studies. Fig. 6 depicts the percentage of studies using different data sets. The data sets used can be publically available or private in nature. Public data sets are freely available while private data sets are not shared by researchers and thus results on such data sets cannot be verified and such studies are not repeatable. The major data sets used and their categorization are as follows:

- *NASA data sets*: These data sets are publically available in the NASA repository by NASA Metrics Data Programme and are the most commonly used data sets for SFP. They are used in 60% of the selected primary studies.
- *PROMISE repository data sets*: These data are freely publically available in the PROMISE repository (https://code.google.com/p/promisedata/) and include AR data sets, Jedit software data sets amongst other data sets. They are used in 15% of the selected primary studies.
- *Eclipse data set*: Around 12% of our selected primary studies used eclipse data set, which is an open source project whose defect data set is publically available. This project is an integrated development environment developed in Java language.
- *Open source project data sets*: This categorization involves studies which use other open source projects such as Lucene, Xylan, Ant, Apache, POI, etc. amongst others used by about 12% of selected primary studies.
- *Student data set*: This is an academic software developed by students and is used by SF10.
- *Others*: This category includes some industrial and private data sets like that of a commercial bank, commercial Java application, telecommunication company data set etc. They are used by about 9% of selected primary studies.

There are very few studies (about 6%) that use industrial datasets to examine the effectiveness of the ML techniques for SFP.

3.2.6. What is the fault distribution in the data sets used for SFP?

The fault distribution of the consistently used data sets used in the selected primary studies is shown in Fig. 7. In Fig. 7, the percentage of faulty classes/modules in the widely used data sets (used at least in 3% of studies) is shown. For example, KC1 is the most widely used in NASA data set (both module and class level) which is applied in 43% of studies and has 15.4% faulty modules and 40.6% faulty classes. Similarly, PC1 data set is used in 40% of studies and contains only 6.8% of faulty modules.

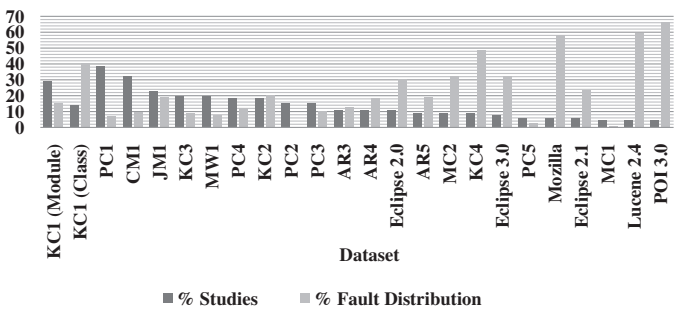


Fig. 7. Distribution of faults in studies.

The results show that 8% of datasets used in studies contain 0–4% of faulty classes/modules, 17% of datasets used in studies contain 5–9% of faulty classes/modules, 17% of datasets used in studies contain 10–14% of faulty classes/modules, and 14% of datasets used in studies contain 15–19% of faulty classes/modules. Hence, 60% of datasets used in studies contain less than 20% of faulty classes/modules which shows that very less amount of faulty classes/modules were present in data sets in more than half of the selected primary studies. The data having non proportionate ratio of faulty and not faulty modules/classes is often known as unbalanced data in the literature. The details on fault distribution in datasets corresponding to the studies are given in Appendix, Table A.2.

3.2.7. Which performance metrics are used for SFP?

A number of metrics are used for gauging the performance of different SFP models. These performance metrics are used for comparing and evaluating models developed using various ML and statistical techniques. Table 6 below depicts the performance metrics, their definitions and the studies in which they are used.

A depiction of the number of studies using each performance metric is used in Fig. 8. According to the figure, the most commonly used performance metric is Recall which is closely followed by Accuracy, Precision and AUC measures. Specificity, Pf and F-measures are other commonly used metrics. Some less commonly metrics are grouped in the miscellaneous category namely H-measure, Precision-recall curve, error rate, G-mean, G-mean1 and G-mean2.

There has been debate on the use of performance measures given the imbalanced nature of datasets in literature. Zhang et al. [16] criticize the use of recall and Pf for construction of prediction models. Menzies et al. [17] criticize the use of precision and pf, however recent studies (Lessmann et al. [42]; Menzies et al. [31]) advocate the use of AUC as performance measure for construction of SFP models. The detailed discussion on the use of performance measures is presented in Section 3.4.

3.3. RQ3: what is the overall performance of ML techniques for SFP?

In this section we summarize the results obtained by examining the results of SFP using the ML techniques. We provide the results of the ML techniques that were assessed in at least five out of 64 selected primary studies. The results are provided using two frequently used performance measures in the 64 primary studies (refer Section 3.2.5). The results in Section 3.2.5 show that Accuracy,

Table 6
Performance measures used in studies.

Performance metric	Definition	Studies
Accuracy	It represents the proportion of total number of correct predictions amongst total number of correct as well as incorrect predictions. It is also referred as *Precision* in certain studies.	SF2, SF3, SF4, SF6, SF8, SF11, SF13, SF14, SF18, SF20, SF21, SF22, SF25, SF27, SF31, SF32, SF35, SF37, SF38, SF40, SF41, SF42, SF43, SF46, SF48, SF49, SF53, SF57, SF60, SF63.
Precision/ correctness	It represents the proportion of correctly classified fault prone classes amongst total number of classified fault prone classes.	SF4, SF5, SF6, SF7, SF8, SF10, SF13, SF18, SF20, SF21, SF22, SF33, SF34, SF35, SF37, SF38, SF44, SF49, SF50, SF51, SF53, SF59, SF63.
Recall (sensitivity, Pd, true positive rate (TPR))	It is the proportion of correctly predicted fault prone classes amongst all actual fault prone classes.	SF2, SF3, SF5, SF7, SF8, SF9, SF12, SF13, SF14, SF15, SF16, SF18, SF20, SF21, SF22, SF24, SF25, SF26, SF27, SF28, SF30, SF32, SF33, SF34, SF35, SF36, SF37, SF38, SF40, SF43, SF44, SF48, SF49, SF50, SF51, SF53, SF56, SF57, SF59, SF60, SF62, SF63.
Specificity (true negative rate (TNR))	It is the proportion of correctly predicted non-fault prone classes amongst all actual non-fault prone classes.	SF2, SF3, SF8, SF14, SF20, SF27, SF32, SF33, SF40, SF43, SF44, SF48, SF53, SF57, SF60.
Pf (false positive rate (FPR))	It is the ratio of all non-fault prone classes which are incorrectly predicted as fault prone.	SF2, SF3, SF9, SF12, SF14, SF15, SF16, SF24, SF25, SF26, SF28, SF29, SF30, SF34, SF36, SF43, SF50, SF51, SF56, SF62.
FNR (false negative rate)	It is the ratio of faulty classes which are classified as non-fault prone.	SF14, SF29, SF43.
AUC (area under the curve)	ROC (receiver operating characteristic curve) is plotted with recall values on the *y*-axis and the 1-specificity values on the *x*-axis. The effectiveness of the model is calculated by measuring the area under the ROC curve.	SF7, SF13, SF17, SF20, SF21, SF23, SF32, SF33, SF36, SF37, SF38, SF40, SF43, SF44, SF45, SF48, SF49, SF54, SF56, SF57, SF58, SF63, SF64.
Balance	It represents the optimal cutoff point of (pf, sensitivity) i.e. the normalized eucledian distance from which from (0,1) point in the ROC curve.	SF12, SF15, SF16, SF24, SF26, SF28, SF30, SF34, SF51, SF54, SF61.
Completeness	It is the sum of all faults in classified fault prone classes over the total number of actual faults in a system.	SF4, SF6, SF10, SF32, SF33, SF40.
F-measure	It is the harmonic mean of precision and sensitivity.	SF5, SF8, SF13, SF18, SF20, SF21, SF22, SF38, SF43, SF49, SF50, SF52, SF53, SF56, SF59, SF61, SF63.
Other miscellaneous measures	They include H-measure, Precision-Recall Curve, Error Rate, G-mean1 and G-mean2.	SF8, SF13, SF20, SF29, SF31, SF39, SF43, SF55, SF64.

F-measure, Precision Recall and AUC are the most frequently used performance measures in the selected primary studies. However, we provide the summarized results of two performance measures Accuracy and AUC. This is due to the fact that Menzies et al. [17] has criticized the use of F-measure and precision due to their unstable nature. In addition, most of the ML techniques being examined evaluate the effectiveness of the models using AUC.

Table 7
Results of SFP models using RF, J48, MLP and NB techniques.

Method	Count	Performance measure	Min.	Max.	Mean	Median	Std.
RF	7	Acc.	55	93.4	75.63	75.94	15.66
	35	AUC	0.66	1	0.83	0.82	0.09
J48	8	Acc.	78	92.562	83.02	81.7	4.54
MLP	10	Acc.	64.02	93.44	82.23	83.46	9.44
	40	AUC	0.54	0.95	0.78	0.77	0.09
NB	18	Acc.	61.92	92.56	80.57	81.4	7.63
	47	AUC	0.64	0.95	0.78	0.78	0.08

We have used box plots to gain insight about the prediction accuracy of the ML techniques. In Fig. 9 we present the box plots for AUC and accuracy values of the ML techniques. The outliers were removed from the analysis before providing the statistics.

Tables 7 and 8 present the minimum (Min.), maximum (Max.), mean, median and standard deviation (Std.) values for two selected performance measures. The higher values of the performance measures indicate more accurate prediction. In terms of AUC, the RF performed the best with 0.83 value, followed by the MLP, NB and BN models. The SVM models performed the worst with 0.7 value of AUC. The results indicate that the mean values of AUC for all the seven ML techniques range from 0.7 to 0.83. Similarly the mean values of accuracy range from 75% to 85%. This shows that the ML techniques have reasonable prediction capability (the values of AUC and accuracy indicate acceptable levels).

3.4. RQ4: whether the performance of ML techniques better than the statistical techniques?

The models predicted using the ML techniques have been compared with the models predicted using the traditional logistic regression (LR) technique, which is the most widely used statistical technique in the literature. In order to answer RQ4 we compared the LR models with ML models. The ML models showing above 2% improvement in the values of AUC were considered to outperform the corresponding LR models under comparison. It was observed that only 29% of studies compared LR and ML models (19 out of 64). We analyzed the results of those techniques which were compared with LR technique in at least three studies and five data sets.

AUC computed from ROC analysis is widely used in medical diagnosis since the past many years, and its use is increasing in the field of data mining research. Carvalho et al. [56] advocated AUC to be the relevant criterion for dealing with unbalanced and noisy data as AUC is insensitive to the changes in distribution of class. He and Garcia [18] have recommended the use of AUC for dealing the issues of imbalanced data as with regard to class distributions, it provides a visual representation of the relative trade-offs between the advantages (represented by true positives) and costs (represented by false positives) of classification. Further, recent studies (Lessmann et al. [42]; Menzies et al. [31]) have pointed out the existence of imbalanced data and use of AUC for construction of fault prediction models. We have seen the presence of imbalanced data in various studies in literature (refer Section 3.2.5), hence AUC

Table 8
Results of SFP models using BN, SVM and C4.5 techniques.

Method	Count	Performance measure	Min.	Max.	Mean	Median	Std.
BN	4	Acc.	74.89	81.43	76.88	75.6	3.06
	25	AUC	0.62	0.9	0.78	0.79	0.08
SVM	24	Acc.	63.34	97.6	80.77	80.41	9.34
	17	AUC	0.5	0.94	0.7	0.71	0.17
C4.5	11	Acc.	68.81	93.59	84.79	84.1	7.48
	20	AUC	0.5	0.99	0.77	0.77	0.12

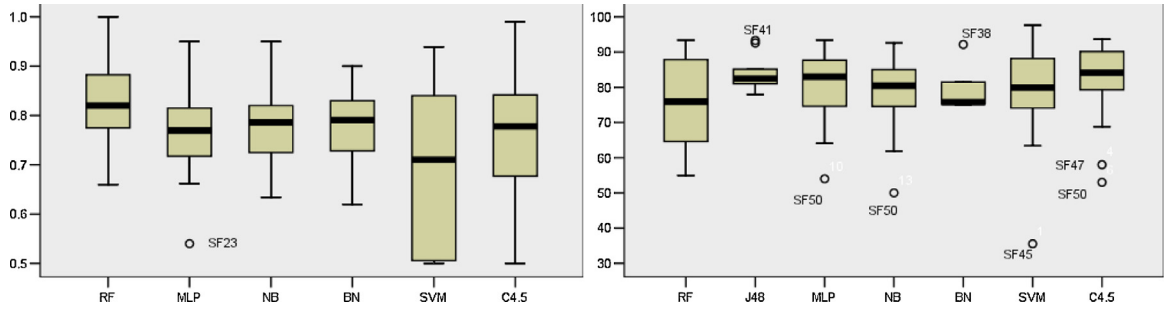


Fig. 9. Box plots for AUC and accuracy (outliers are labelled with study identification nos.).

is used for comparing the performance of models predicted using ML and LR techniques in this SLR.

The Fig. 10 indicates the percentage of the ML models (shown by bars above the zero-line) having value greater than the LR models with respect to datasets. The results show that 65% of the ML models outperform in terms of AUC as compared to LR models with respect to the datasets. The comparison results show that the RF, NB, C4.5, MLP and SVM are the most commonly used techniques for comparison in the literature. The RF (80%) and Bagging (100%) models outperform the LR models in most of the datasets. The NB model outperforms the LR models in 66% of studies followed by the MLP and C4.5 in 60% of studies. The LR models outperform the SVM models in 60% of studies. The models predicted using the J48 technique is compared with the LR models in only four studies (SF9, SF20, SF25, SF60). No more than three studies compared the AB (SF49, SF58, SF64) and Bagging (SF20, SF49, SF58) models with the LR model. The remaining ML techniques are compared in only one or two studies. Hence, the results obtained from the comparison of the ML techniques with the LR technique in one or two studies are not assessed and shown in this section as the less number of comparisons limits the possibility of generalization of results.

We may conclude that the ML models outperform the LR models in general. However, there is a threat to the validity to this conclusion. Due to lack of the studies comparing the performance of the LR and ML models (only 29%), it is difficult to obtain and generalize the conclusions.

Thus, more number of studies comparing ML models and LR models for SFP should be carried out in order to obtain well-formed and generalized results.

3.5. RQ5: are there any ML techniques that significantly outperform other ML techniques?

For the purpose of comparing the results of ML models with respect to other ML models we adopted the same methodology as in RQ4. The AUC was used to compare the results of ML models with other ML models. Table 9 presents the summary of average AUC,

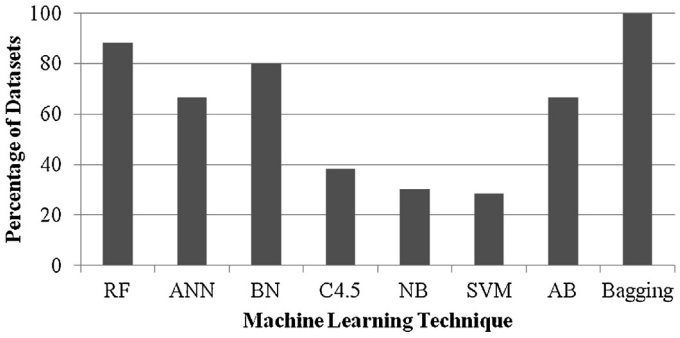


Fig. 10. Comparison of AUC between ML models and LR models.

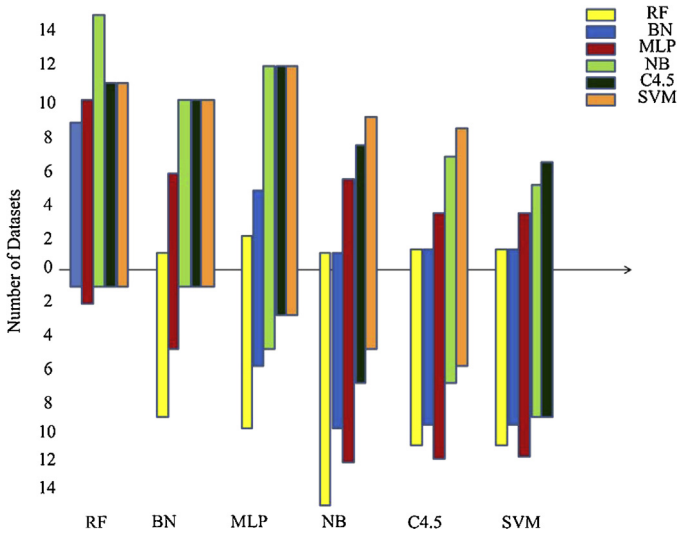


Fig. 11. Comparison of AUC between ML models vs ML models.

the best technique and the percentage of faulty modules/classes corresponding to the datasets. The results show that irrespective of the percentage of faulty data points in the datasets the AUC is greater than 0.6.

Fig. 11 shows the comparison results of ML models with other ML models with respect to the AUC value. The bars above the

Table 9
Average AUC corresponding to datasets.

Best technique	Dataset	Faulty %	Average AUC
RF	CM1	9.49	0.810
SVM	Cos	50.00	0.793
RF	Eclipse 2.0a	14.49	0.820
RF	Eclipse 2.1 a	10.83	0.740
RF	Eclipse 3.0a	14.80	0.770
ANN	Eclipse 2.0	30.80	0.662
ANN	Eclipse 2.1	23.48	0.710
NB	Eclipse 3.0	32.26	0.649
BN	Ivy	11.36	0.846
BN	Jedit	2.24	0.658
RF	JM1	19.34	0.747
RF	KC1	15.42	0.800
SVM	KC1 (class)	40.68	0.890
BN	KC2	20.38	0.850
RF	KC3	9.36	0.860
RF	KC4	48.8	0.850
RF	MC1	0.72	0.915
RF	MW1	7.67	0.810
RF	PC1	6.86	0.850
ANN	PC2	0.41	0.885
RF	PC3	10.23	0.807
ANN	PC4	12.20	0.945
RF	PC5	3.00	0.970

Table 10
Strengths and weaknesses of ML techniques.

Technique name	Strengths	Supporting studies
C4.5	The model developed is cost-effective and simple.	SF7, SF31
	Easy to build and apply.	SF7, SF27
	Comprehensive capability.	SF27, SF36
RF	It can handle large data and are consistent performers.	SF3, SF21
	The method is robust to noisy and missing data.	SF3, SF21
	Fast to train, robust towards parameter setting.	SF2, SF23, SF21
	Provide understandable model.	SF23, SF21
	Comprehensive capability.	SF23, SF36
	Runs efficiently on large data sets.	SF3, SF48, SF57
	Helps in identifying most important independent variables.	SF3, SF48, SF57
NB	It is robust in nature.	SF29, SF47
	It is easy to interpret and construct.	SF23, SF64
	Computationally efficient.	SF23, SF64
SVM	Good tolerance for high-dimension space and redundant features.	SF49, SF62, SF63
	Robust in nature.	SF23, SF62
	It can handle complex functions.	SF23, SF40
	It can handle non linear problems.	SF40, SF49
ANN	It can infer complex non-linear I/O transformation.	SF19, SF32
Technique name	Weaknesses	Supporting studies
NB	Does not consider feature correlation.	SF28, SF20
	Performance of model dependent on attribute selection technique used.	SF50, SF64
	Unable to discard irrelevant attributes.	SF50, SF64

zero-line depict the percentage of datasets in which the ML technique outperforms the other ML techniques. The bars below the zero-line were only used when the ML technique did not outperform the other ML techniques in any of the dataset. For example, the RF models outperform BN models in 9 datasets and underperforms BN model in 1 dataset. The RF models outperform MLP models in 10 datasets and underperform MLP models in 2 datasets. The RF models outperform NB models in 15 datasets and underperform NB model in 1 dataset. Similarly, the RF model outperform C4.5 and SVM models in 11 datasets an underperform C4.5 and SVM models in 1 dataset.

The results show that the RF models outperformed in most of the studies in comparison to the models predicted using the MLP, NB, BN, C4.5 and SVM techniques. The NB models outperform the MLP and SVM models in all the studies and the BN models outperformed the MLP, C4.5 and SVM models. Except the BN model, the SVM models did not outperformed any other ML technique.

3.6. RQ6: what are the strengths and weaknesses of the ML techniques?

In this section we outline the strengths and weaknesses exhibited by the ML techniques as recorded by researchers. These are representative of author's opinion and may not be reliable. Hence, the strengths and weaknesses of the ML techniques supported by more than one study are presented in this section. The NB technique has been reported to exhibit good performance in defect prediction problems and is also good in handling multiple datasets with varying properties. At the same time this technique does not consider the correlation among features in the input and the performance of the NB technique varies heavily with the attribute selection technique used. The SVM has been applauded for its excellent ability to deal with the redundant features and high-dimensionality. The C4.5 algorithm is simple in implementation and provides good results, so is a good way of including the power of ML in one's research without much overhead. Table 10 summarizes the strengths and weaknesses of the ML techniques for SFP.

Summarily, different techniques have different pros and there is not a one size fits all solution to the problem of SFP in form of a ML technique. It all depends on the domain of application; in a situation where false positives are fatal would favour some techniques in comparison to a situation where cost is a factor.

4. Limitations

This systematic review considered a number of primary studies to evaluate and assess the performance of various ML techniques amongst themselves and with the LR technique. A limitation in this review is that only 19 out of the 64 primary studies compare LR and ML techniques. Thus, the LR vs ML comparison is not definitely conclusive. Moreover, while comparing ML techniques, each primary study may use different experimental settings which include data sets, feature reduction methods, weighing methods and preprocessing methods [9]. Though we have exhaustively searched all the stated digital search libraries, there still may be a possibility that a suitable study may be left out. Also this review does not include any unpublished research studies [19]. We have assumed that all the studies are impartial, however if this is not the case then it poses a threat to this study.

5. Conclusion and future guidelines

In this paper we perform a systematic literature review in order to analyze and assess the performance of the ML techniques for software fault prediction (SFP). First, after a thorough analysis by following a systematic series of steps and analysing the quality of the studies we identified 64 primary studies (1991–2013). Second, we summarized the characteristics of the primary studies based on metrics reduction techniques, metrics, data sets and performance measures. Third, we assessed the performance of the ML techniques for software fault prediction. Fourth, we compared the performance of models predicted using the ML techniques with the models predicted using the logistic regression technique. Then, we analyzed the performance of these predicted models using the ML techniques with other ML techniques. Finally, we summarized the strengths and weaknesses of the ML techniques. The main findings obtained from the selected primary studies are:

- The ML techniques were broadly categorized into Decision Tree, Bayesian Learning, Ensemble Learning, Rule Based Learning, Evolutionary Algorithms, Neural Networks and Miscellaneous. The most frequently used ML techniques for Software Fault Prediction were C4.5, Naive Bayes, Multilayer Perceptron, Support Vector Machines and Random Forest.
- The most commonly used technique for sub selection of metrics in the studies was Correlation based Feature Selection. The procedural metrics were found to be the most commonly used metrics in the literature. Among Object Oriented metrics CBO, RFC and LOC were found to be useful. The results show that the NASA data

set was the most frequently used data set in the literature. However, it was observed that very few studies use industrial datasets for evaluating the effectiveness of the ML techniques. The results depict that Accuracy, precision, Recall, AUC, F-Measure are the most commonly used performance measures in the primary studies.

- The ML techniques have acceptable prediction capability for estimating software fault proneness. The average values of AUC ranges from 0.7 to 0.83 and the mean values of accuracy ranges from 75% to 85%.
- The ML models outperformed the LR models for SFP in general.
- The models predicted using the RF technique outperformed all the other ML models in all the studies. The models predicted using the NB and BN techniques outperformed the other ML models in most of the studies.

The following are the guidelines for the researchers and software practitioners for carrying out future research on software fault prediction using the ML techniques:

(1) More number of studies for software fault prediction should be carried out using the ML techniques in order to obtain generalizable results. There are few studies that compare the ML techniques with the LR techniques. Hence, more number of studies should compare and assess the performance of the ML techniques with LR technique.
(2) It is observed that there are very few studies that examine the predictive capability of RBL, LB, AB, Bagging, RBF, ADT. Some ML techniques have never been applied in FP. Studies should be conducted in order to assess the prediction accuracy of these rarely used techniques.
(3) There are very few studies that examine the effectiveness of evolutionary algorithms such as GP, ACO for software fault prediction. The future studies may focus on the predictive accuracy of evolutionary algorithms for software fault prediction.
(4) The data sets should be made available in public so that more number of studies can be carried out using the freely available data sets.
(5) The ML techniques should be carefully selected based on their properties. Before making the decision of selection of a ML technique, the characteristics of the ML technique should be understood completely by the researcher.
(6) In order to assess the true potential of a given ML technique, it should be compared with other ML/statistical techniques.
(7) Most of the studies were based on the NASA or publicly available open source software. Hence, studies that explore the effectiveness of the ML techniques on industrial data sets should be conducted in order to provide practical usefulness of the ML techniques.
(8) The study should clearly specify the parameters values for the respective ML techniques used so that the framework in the studies can be successfully repeated by the software community.

Acknowledgements

The author would like to thank Prof. Yogesh Singh for his guidance, constant input and suggestions throughout the work. The author is indebted to him for his advice and reviews on selection of primary studies. The author would also acknowledge the support of Megha Khanna and Nikita Jain for assisting in collecting and extracting the required data.

Appendix.

See Tables A.1 and A.2.

Table A.1
RQ's addressed in individual studies.

Study no.	RQ1	RQ2	RQ3	RQ4	RQ5	RQ6	Study No.	RQ1	RQ2	RQ3	RQ4	RQ5	RQ6
SF1	√	√	√				SF33	√	√	√			
SF2	√	√	√		√	√	SF34	√	√	√		√	
SF3	√	√	√		√	√	SF35	√	√	√			
SF4	√	√	√		√		SF36	√	√	√	√	√	√
SF5	√	√	√				SF37	√	√	√	√	√	
SF6	√	√	√		√		SF38	√	√	√		√	
SF7	√	√	√	√	√	√	SF39	√	√	√		√	
SF8	√	√	√		√		SF40	√	√	√		√	√
SF9	√	√	√	√	√		SF41	√	√	√		√	
SF10	√	√	√		√		SF42	√	√	√		√	
SF11	√	√	√		√		SF43	√	√	√		√	
SF12	√	√	√		√	√	SF44	√	√	√	√	√	
SF13	√	√	√		√		SF45	√	√	√	√	√	
SF14	√	√	√			√	SF46	√	√	√		√	
SF15	√	√	√				SF47	√	√	√			√
SF16	√	√	√		√		SF48	√	√	√	√	√	√
SF17	√	√	√		√		SF49	√	√	√	√	√	√
SF18	√	√	√		√		SF50	√	√	√			√
SF19	√	√	√		√	√	SF51	√	√	√		√	
SF20	√	√	√	√	√	√	SF52	√	√	√			
SF21	√	√	√		√	√	SF53	√	√	√		√	
SF22	√	√	√				SF54	√	√	√		√	
SF23	√	√	√	√	√	√	SF55	√	√	√		√	
SF24	√	√	√		√		SF56	√	√	√			
SF25	√	√	√	√	√		SF57	√	√	√	√	√	√
SF26	√	√	√		√		SF58	√	√	√			
SF27	√	√	√	√	√	√	SF59	√	√	√	√	√	
SF28	√	√	√		√	√	SF60	√	√	√	√	√	
SF29	√	√	√			√	SF61	√	√	√	√	√	
SF30	√	√	√				SF62	√	√	√		√	√
SF31	√	√	√		√	√	SF63	√	√	√	√	√	√
SF32	√	√	√	√	√	√	SF64	√	√	√	√	√	√

Table A.2
Fault distribution in studies.

Range of % fault distribution	# Datasets	Datasets with studies
0–4	9	**PC2** (SF3, SF12, SF15, SF16, SF23, SF24, SF28, SF54, SF62, SF64), **MC1** (SF24, SF54, SF64), **SP3** (SF43), **SP4** (SF43), **PC5** (SF20, SF24, SF54, SF64), **ARGO** (SF52), **Camel 1.0** (SF59), **Inventory System 13** (SF41), **SP2** (SF43)
5–9	17	**Inventory System 14** (SF41), **CCCS12** (SF43), **Inventory System 17** (SF41), **Inventory System 16** (SF41), **SP1** (SF43), **Ivy 1.4** (SF59), **PC1** (SF3, SF5, SF11, SF12, SF13, SF15, SF16, SF17, SF18, SF20, SF23, SF24, SF27, SF28, SF30, SF38, SF39, SF43, SF51, SF53, SF54, SF55, SF56, SF62, SF64), **Inventory System 12** (SF41), **Inventory System 15** (SF41), **Inventory System 11** (SF41), **AR1** (SF44, SF54), **MW1** (SF12, SF15, SF16, SF23, SF24, SF28, SF30, SF39, SF43, SF53, SF54, SF56, SF62), **Inventory System 10** (SF41), **Datatrieve** (SF42), **KC3** (SF12, SF15, SF16, SF18, SF23, SF24, SF28, SF39, SF43, SF53, SF54, SF56, SF62), **CM1** (SF3, SF5, SF9, SF12, SF13, SF15, SF16, SF18, SF20, SF23, SF24, SF28, SF30, SF38, SF43,SF51, SF53, SF54, SF56, SF62), **CCCS8** (SF43).
10–14	17	**Argo uml 0.26.2** (SF52), **Synapse 1.0** (SF59), **PC3** (SF12, SF15, SF16, SF23, SF24, SF28, SF51, SF54, SF62, SF64), **Arc** (SF48), **Inventory System 7** (SF41), **Eclipse 2.1a** (SF64), **Ant 1.5** (SF59), **Ivy 2.0** (SF59, SF58), **Inventory System 9** (SF41), **POI 2.0** (SF59), **PC4** (SF12, SF15, SF16, SF23, SF24, SF27, SF28, SF51, SF54, SF60, SF63, SF64), **Inventory System 8** (SF41), **AR3** (SF26, SF29, SF30, SF34, SF51, SF54, SF56), **Inventory System 6** (SF41), **Eclipse 2.0a** (SF64), **Eclipse 3.0a** (SF64), **AR6** (SF54, SF51).
15–19	15	**Xalan 2.4** (SF59), **Xerces 1.3** (SF59), **Inventory System 4** (SF41), **KC1** (SF3, SF9, SF12, SF15, SF18, SF20, SF23, SF24, SF27, SF30, SF38, SF39, SF43, SF53, SF54, SF56, SF60, SF62, SF64), **Inventory System 2** (SF41), **Ant 1.3** (SF59), **Xerces 1.2** (SF59), **Inventory System 5** (SF41), **Camel 1.4** (SF59), **AR4** (SF26, SF29, SF30, SF34, SF51,SF54, SF56), **JM1** (SF3, SF5, SF9, SF11, SF13, SF19, SF20, SF23, SF38, SF39, SF42, SF43, SF54, SF55, SF62, SF64), **AR5** (SF26, SF29, SF30, SF34, SF51, SF56), **Camel 1.6** (SF59), **CCCS4** (SF43), **Inventory System 3** (SF41).
20–24	7	**KC2** (SF2, SF3, SF5, SF12, SF13, SF20, SF24, SF30, SF38, SF39, SF43, SF53, SF56), **Ant 1.7** (SF58, SF59), **Jedit 4.3** (SF58), **Ant 1.4** (SF59), **Eclipse 2.1** (SF25, SF34, SF35, SF45), **Jedit 4.0** (SF49, SF59), **Eclipse Component: SWT** (SF61).
25–29	3	**Ant 1.6** (SF59), **Symapse 1.1** (SF59), **CCCS2** (SF43).
30–34	6	**Eclipse 2.0** (SF25, SF34, SF35, SF61, SF52, SF50, SF45), **Eclipse 3.0** (SF25, SF35, SF45, SF52, SF61), **MC2** (SF20, SF24, SF30, SF53, SF54, SF56), **Jedit 3.2** (SF59, SF21), **Synapse 1.2** (SF58, SF59), **Velocity 1.6** (SF58, SF59).
35–39	2	**Camel 1.2** (SF60), **Inventory System 1** (SF41).
40–44	2	**KC1** (SF5, SF6, SF8, SF14, SF32, SF33, SF38, SF40, SF50), **Library Management System** (SF10).
45–49	5	**Xalan 2.6** (SF59, SF61), **Lucene 2.0** (SF59), **Xerces init** (SF59), **Xalan 2.5** (SF58, SF59), **KC4** (SF15, SF16, SF20, SF23, SF28, SF54).
50–54	2	**Cos** (SF7, SF37), **Eclipse Component: JDT Core** (SF61)
55–59	5	**Ivy 1.1** (SF59), **Mozilla** (SF1), **Lucene 2.2** (SF59), **POI 1.5** (SF59), **Lucene 2.4** (SF58, SF59, SF61)
60–64	1	**POI 2.5** (SF59).
65–70	2	**Velocity 1.5** (SF59), **POI 3.0** (SF57, SF58, SF59).
>70	5	**Xerces 1.4** (SF59), **Velocity 1.4** (SF59), **Tomcat 6.0** (SF35, SF58), **Jedit1** (SF49, SF46), **Jedit2** (SF49, SF46).

Bold text signifies the names of the data sets used in the respective studies depicted in brackets.

References

[1] N. Seliya, T.M. Khoshgoftaar, J. Van Hulse, Predicting faults in high assurance software, in: IEEE 12th International Symposium on High-Assurance Systems Engineering, 2010, pp. 26–34.
[2] Y. Singh, A. Kaur, R. Malhotra, Empirical validation of object-oriented metrics for predicting fault proneness models, Softw. Qual. J. 18 (2009) 3–35.
[3] D. Radjenovic, M. Hericko, R. Torkar, A. Zivkovic, Software fault prediction metrics: a systematic literature review, Inform. Softw. Technol. 55 (2013) 1397–1418.
[4] C. Catal, Software fault prediction: a literature review and current trends, Expert Syst. Appl. 38 (2011) 4626–4636.
[5] C. Catal, B. Diri, A systematic review of software fault prediction studies, Expert Syst. Appl. 36 (2009) 7346–7534.
[6] B.A. Kitchenham, Guidelines for performing systematic literature review in software engineering, Technical report EBSE-2007-001, UK, 2007.
[7] T.M. Mitchell, Machine Learning, McGraw Hill, Maidenhead, UK, 1997.
[8] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Elsevier, USA, 2005.
[9] J. Wen, S. Li, Z. Lin, Y. Hu, C. Huang, Systematic literature review of machine learning based software development effort estimation models, Inform. Softw. Technol. 54 (2012) 41–59.
[10] M.A. Hall, Correlation-based feature felection for fiscrete and numeric class machine learning, in: Proceedings of the Seventeenth Coference on Machine Learning, 2000, pp. 359–366.
[11] M. Halstead, Elements of Software Science, Elsevier, New York, 1977.
[12] T. McCabe, A complexity measure, IEEE Trans. Softw. Eng. 2 (1976) 308–320.
[13] S.R. Chidamber, C.F. Kemerer, A metrics suite for object oriented design, IEEE Trans. Softw. Eng. 20 (1994) 476–493.
[14] K.K. Aggarwal, Y. Singh, A. Kaur, R. Malhotra, Empirical study of object-oriented metrics, J. Obj. Technol. 5 (2006) 149–173.
[15] M. Lorenz, J. Kidd, Object-oriented Software Metrics, Prentice Hall, Englewood Cliffs, NJ, 1994.
[16] H. Zhang, X. Zhang, Comments on 'data mining static code attributes to learn defect predictors', IEEE Trans. Softw. Eng. 33 (2007) 635–636.
[17] T. Menzies, A. Dekhtyar, J. Distefance, J. Greenwald, Problems with precision: a response to comments on 'data mining static code attributes to learn defect predictors', IEEE Trans. Softw. Eng. 33 (2007) 637–640.
[18] H. He, E.A. Gracia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (2009) 1263–1284.
[19] W. Afzal, R. Torkar, On the application of genetic programming for software engineering predictive modelling: a systematic review, Expert Syst. Appl. 38 (2011) 11984–11997.
[20] S.A. Sherer, Software fault prediction, J. Syst. Softw. 29 (1995) 97–105.
[21] L. Guo, B. Cukic, H. Singh, Predicting fault prone modules by the dempster-shafer belief networks, in: 18th IEEE International Conference on Automated Software Engineering, 2003, pp. 3–6.
[22] L. Guo, Y. Ma, B. Cukic, H. Singh, Robust prediction of fault-proneness by random forests, in: 15th International Symposium on Software Reliability Engineering, 2004, pp. 417–428.
[23] T. Gyimothy, R. Ferenc, I. Siket, Empirical validation of object-oriented metrics on open source software for fault prediction, IEEE Trans. Softw. Eng. 31 (2005) 897–910.
[24] A.G. Koru, H. Liu, Building effective defect-prediction models in practice, IEEE Softw. 22 (2005) 23–29.
[25] Y. Zhou, H. Leung, Empirical analysis of object-oriented design metrics for predicting high and low severity faults, IEEE Trans. Softw. Eng. 32 (2006) 771–789.
[26] E. Arisholm, L.C. Briand, M. Fuglerud, Data mining techniques for building fault-proneness models in telecom java software, in: 18th IEEE International Symposium on Software Reliability, 2007, pp. 215–224.
[27] C. Catal, B. Diri, B. Ozumut, An artificial immune system approach for fault prediction in object-oriented software, in: 2nd International Conference on Dependability of Computer Systems, 2007, pp. 1–8.
[28] Y. Jiang, B. Cukic, T. Menzies, Fault prediction using early lifecycle data, in: 18th IEEE International Symposium on Software Reliability, 2007, pp. 237–246.
[29] S. Kanmani, V.R. Uthariaraj, V. Sankaranarayanan, P. Thambidurai, Object-oriented software fault prediction using neural networks, Inform. Softw. Technol. 49 (2007) 483–492.
[30] Z. Li, M. Reformat, A practical method for the software fault-prediction, in: IEEE International Conference on Information Reuse and Integration, 2007, pp. 659–666.
[31] T. Menzies, J. Greenwald, A. Frank, Data mining static code attributes to learn defect predictors, IEEE Trans. Softw. Eng. 33 (2007) 2–13.
[32] Y. Ma, L. Guo, B. Cukic, A statistical framework for the prediction of fault-proneness, Adv. Mach. Learn. Appl. Softw. Eng. IGI Global (2007) 237–263.
[33] G.J. Pai, J.B. Dugan, Empirical analysis of software fault content and fault proneness using bayesian methods, IEEE Trans. Softw. Eng. 33 (2007) 675–686.

[34] B. Turhan, A. Bener, A multivariate analysis of static code attributes for defect prediction, in: 7th International Conference on Quality Software, 2007, pp. 231–237.
[35] B. Turhan, A. Bener, Software defect prediction: heuristics for weighted naïve bayes, in: 2nd International Conference on Software and Data Technologies, 2007, pp. 244–249.
[36] A.B. De Carvalho, A. Pozo, S. Vergilio, A. Lenz, Predicting fault proneness of classes through a multiobjective particle swarm optimization algorithm, in: 20th IEEE International Conference on Tools with Artificial Intelligenec, 2008, pp. 387–394.
[37] K.O. Elish, M.O. Elish, Predicting defect-prone software modules using support vector machines, J. Syst. Softw. 81 (2008) 649–660.
[38] I. Gondra, Applying machine learning to software fault-proneness prediction, J. Syst. Softw. 81 (2008) 186–195.
[39] Y. Jiang, Y. Ma, B. Cukic, Techniques for evaluating fault prediction models, Empir. Softw. Eng. 13 (2008) 561–595.
[40] A. Kaur, R. Malhotra, Application of random forest in predicting fault-prone classes, in: International Conference on Advanced Computer Theory and Engineering, 2008, pp. 37–43.
[41] S. Kim, E.J. Whitehead, Y. Zhang, Classifying software changes: clean or buggy? IEEE Trans. Softw. Eng. 34 (2008) 181–196.
[42] S. Lessmann, B. Baesens, C. Mues, S. Pietsch, Benchmarking classification models for software defect prediction: a proposed framework and novel findings, IEEE Trans. Softw. Eng. 34 (2008) 485–496.
[43] T. Menzies, B. Turhan, A. Bener, G. Gay, B. Cukic, Y. Jiang, Implications of ceiling effects in defect predictors, in: 4th International Workshop on Predictive Models in Software Engineering – PROMISE'08, 2008, pp. 47–54.
[44] R. Moser, W. Pedrycz, G. Succi, A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction, in: 30th International Conference on Software Engineering, 2008, pp. 181–190.
[45] B. Turhan, G. Kocak, A. Bener, Software defect prediction using call graph based ranking (CGBR) framework, in: 34th Euromicro Conference on Software Engineering and Advanced Applications, 2008, pp. 191–198.
[46] O. Vandecruys, D. Martens, B. Baesens, C. Mues, M. D Backer, R. Haesen, Mining software repositories for comprehensible software fault prediction models, J. Syst. Softw. 81 (2008) 823–839.
[47] A. Bener, B. Turhan, Analysis of naïve bayes assumptions on software fault data: an empirical study, Data Knowl. Eng. 68 (2009) 278–290.
[48] C. Catal, U. Sevim, B. Diri, Clustering and metrics thresholds based software fault prediction of unlabeled program modules, in: Sixth International Conference on Information Technology: New Generations, 2009, pp. 199–204.
[49] T. Menzies, B. Turhan, J. D Stefano, A.B. Bener, On the relative value of cross-company and within-company data for defect prediction, Empir. Softw. Eng. 14 (2009) 540–578.
[50] P. Singh, S. Verma, An investigation of the effect of discretization on defect prediction using static measures, in: International Conference on Advances in Computing, Control, and Telecommunication Technology, 2009, pp. 837–839.
[51] Y. Singh, A. Kaur, R. Malhotra, Software fault proneness prediction using support vector machines, in: Procedings of the World Congress on Engineering, 2009, pp. 240–245.
[52] A. Tosun, B. Turhan, A. Bener, Validation of network measures as indicators of defective modules in software systems, in: Proceedings of 5th International Conference on Predictive Models Software Engineering, 2009.
[53] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, B. Murphy, Cross-project defect prediction: a large scale experiment on data vs domain vs process, in: 7th Joint Meeting of European Software Engineering Conference. ACM SIGSOFT Symposium Foundations of Software Engineering, 2009, pp. 91–100.
[54] W. Afzal, Using faults-slip-through metric as a predictor of fault-proneness, in: Asia Pacific Software Engineering Conference, 2010, pp. 414–422.
[55] E. Arisholm, L.C. Briand, E.B. Johannessen, A systematic and comprehensive investigation of methods to build and evaluate fault prediction models, J. Syst. Softw. 83 (2010) 2–17.
[56] A.B. Carvalho, A. Pozo, S.R. Vegilio, A symbolic fault-prediction model based on multiobjective particle swarm optimization, J. Syst. Softw. 83 (2010) 868–882.
[57] Y. Liu, T.M. Khoshgaftar, N. Seliya, Evolutionary optimization of software quality modeling with multiple repositories, IEEE Trans. Softw. Eng. 36 (2010) 852–864.
[58] R. Malhotra, Y. Singh, A. Kaur, Empirical validation of object-oriented metrics for predicting fault proneness at different severity levels using support vector machines, Int. J. Syst. Assur. Eng. Manag. 1 (2010) 269–281.
[59] T.J. Ostrand, E.J. Weyuker, R.M. Bell, Comparing the effectiveness of several modeling methods for fault prediction, Empir. Softw. Eng. 15 (2010) 277–295.
[60] P.C. Pendharkar, Exhaustive and heuristic search approaches for learning a software defect prediction model, Eng. Appl. Artif. Intell. 23 (2010) 34–40.
[61] Y. Singh, A. Kaur, R. Malhotra, Prediction of fault-prone software modules using statistical and machine learning methods, Int. J. Comput. Appl. 1 (2010) 6–13.
[62] Y. Zhou, B. Xu, H. Leung, On the ability of complexity metrics to predict fault-prone classes in object-oriented systems, J. Syst. Softw. 83 (2010) 660–674.
[63] D. Azar, J. Vybihal, An ant colony optimization algorithm to improve software quality prediction models: case of class stability, Inform. Softw. Technol. 53 (2011) 388–393.
[64] B. Diri, C. Catal, U. Sevim, Practical development of an eclipse-based software fault prediction tool using Naive Bayes algorithm, Expert Syst. Appl. 38 (2011) 2347–2353.
[65] R. Malhotra, Y. Singh, On the applicability of machine learning techniques for object-oriented software fault prediction, Softw. Eng. Int. J. 1 (2011) 24–37.
[66] S. Martino, F. ferrucci, C. Gravino, F. Sarro, A genetic algorithm to configure support vetcor machine for predicting fault prone components, in: Proceedings of the 12th International Conference on Product Focus Software Process Improvement, 2011, pp. 247–261.
[67] B. Mishra, K.K. Shukla, Impact of attribute selection on defect proneness prediction in OO software, in: 2nd International Conference on Computing, Communication and Technology, 2011, pp. 367–372.
[68] A.T. Mısırlı, A.B. Bener, B. Turhan, An industrial case study of classifier ensembles for locating software defects, Softw. Qual. J. 19 (2011) 515–536.
[69] S. Kpodjedo, F. Ricca, P. Galinier, Y.-G. Guéhéneuc, G. Antoniol, Design evolution metrics for defect prediction in object oriented systems, Empir. Softw. Eng. 16 (2011) 141–175.
[70] D. Rodríguez, R. Ruiz, J.C. Riquelme, J.S. Aguilar-Ruiz, Searching for rules to detect defective modules: a subgroup discovery approach, Inform. Sci. 191 (2011) 14–30.
[71] Q. Song, Z. Jia, M. Shepperd, S. Ying, J. Liu, A general software defect-proneness prediction framework, IEEE Trans. Softw. Eng. 37 (2011) 356–370.
[72] B. Twala, Software faults prediction using multiple classifiers, in: 3rd International Conference on Computing Research and Development, vol. 4, 2011, pp. 504–510.
[73] A. Chen, Y. Ma, X. Zeng, G. Luo, Transfer learning for cross-company software defect prediction, Inform. Softw. Technol. 54 (2012) 248–256.
[74] R. Malhotra, A. Jain, Fault prediction using statistical and machine learning methods for improving software quality, J. Inf. Process. Syst. 8 (2012) 241–262.
[75] A. Okutan, O.T. Yıldız, Software defect prediction using Bayesian networks, Empir. Softw. Eng. 19 (2012) 154–181.
[76] Y. Yang, Z. He, F. Shu, M. Li, Q. Wang, An investigation on the feasibility of cross-project defect prediction, Autom. Softw. Eng. 19 (2012) 167–199.
[77] L. Yu, An evolutionary programming based asymmetric weighted least squares support vector machine ensemble learning methodology for software repository mining, Inform. Sci. 191 (2012) 31–46.
[78] M. Li, H. Zhang, R. Whu, Z. Zhou, Sample-based software defect prediction with active and semi-supervised learning, Autom. Softw. Eng. 19 (2012) 201–230.
[79] J. Cahill, J.M. Hogan, R. Thomas, Predicting fault-prone software modules with rank sum classification, in: 22nd Australian Software Engineering Conference, 2013, pp. 211–219.
[80] N. Chen, S.C.H. Hoi, X. Xiao, Software process evaluation: a machine learning framework with application to defect management process, Empir. Softw. Eng. 19 (2013) 1531–1564.
[81] K. Dejaeger, T. Verbraken, B. Baesens, Toward comprehensible software fault prediction models using bayesian network classifiers, IEEE Trans. Softw. Eng. 39 (2013) 237–257.
[82] G.K. Upadhya, S. Das, U. Chandra, A.J. Paul, Modelling the investment casting process: a novel approach for view factor calculations and defect prediction, Appl. Math. Model. 19 (1995) 354–362.
[83] M. Evett, B. Raton, E. Allen, GP-based software quality prediction, in: Proceedings of 3rd Annual Genetic Programming Conference, 1998, pp. 60–65.
[84] E. Baisch, T. Liedtke, Comparison of conventional approaches and soft-computing approaches for software quality prediction, in: IEEE International Conference on Systems, Man and Cybernetics, Comput. Cybern. Simul. 2 (1997) 1045–1049.
[85] A.R. Gray, S.G. MacDonell, A comparison of techniques for developing predictive models of software metrics, Inform. Softw. Technol. 39 (1997) 425–437.
[86] M. Zhao, C. Wohlin, N. Ohlsson, M. Xie, A comparison between software design and code metrics for the prediction of software fault content, Inform. Softw. Technol. 40 (1998) 801–809.
[87] T. Kamiya, S. Kusumoto, K. Inoue, Prediction of fault-proneness at early phase in object-oriented development, in: 2nd IEEE International Symposium on Object-Oriented Real-Time Distribution and Computing, 1999, pp. 253–258.
[88] L.C. Briand, J. Wust, J.W. Daly, D.V. Porter, Exploring the relationships between design measures and software quality in object-oriented systems, J. Syst. Softw. 51 (2000) 245–273.
[89] X. Yuan, T.M. Khoshgoftaar, E.B. Allen, K. Ganesan, An application of fuzzy clustering to software quality prediction, in: 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, 2000, pp. 85–92.
[90] G. Denaro, P. Milano, An empirical evaluation of fault-proneness models, in: 24th International Conference on Software Engineering, 2002, pp. 241–251.
[91] T.M. Khoshgoftaar, N. Seliya, Tree-based software quality estimation models for fault prediction, in: 8th IEEE Symposium on Software Metrics, 2002, pp. 203–214.
[92] N.J. Pizzi, A.R. Summers, W. Pedrycz, Software quality prediction using median-adjusted class labels, in: International Joint Conference of Neural Network, 2002, pp. 2405–2409.
[93] M.D. Ambros, M. Lanza, R. Robbes, An extensive comparison of bug prediction approaches, in: 7th IEEE Working Conference on Mining Software Repositories, 2003, pp. 31–41.
[94] T.M. Khoshgoftaar, N. Seliya, Fault prediction modeling for software quality estimation: comparing commonly used techniques, Empir. Softw. Eng. (2003) 255–283.
[95] K. Kaminsky, G. Boetticher, Building a genetically engineerable evolvable program (GEEP) using breadth-based explicit knowledge for predicting software

defects, in: IEEE Annual Meeting on Fuzzy Information and Process, 2004, pp. 10–15.
[96] T. Menzies, J.S. Di Stefano, How good is your blind spot sampling policy? in: 8th IEEE International Symposium on High Asurance System Engineering, 2004, pp. 129–138.
[97] T.S. Quah, M.M.T. Thwin, Prediction of software development faults in PL/SQL files using neural network models Inform. Softw. Technol. 46 (2004) 519–523.
[98] Q. Wang, B. Yu, J. Zhu, Extract rules from software quality prediction model based on neural network, in: 16th IEEE International Conference on Tools with Artificial Intelligence, 2004, pp. 191–195.
[99] G. Canfora, L. Cerulo, Impact analysis by mining software and change request repositories, in: 11th IEEE International Software Metrics Symposium, 2005, pp. 20–29.
[100] V.U.B. Challagulla, F.B. Bastani, I.L. Yen, R.A. Paul, Empirical assessment of machine learning based software defect prediction techniques, in: 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, 2005, pp. 263–270.
[101] F. Xing, P. Guo, M.R. Lyu, A novel method for early software quality prediction based on support vector machine, in: 16th IEEE International Symposium on Software Reliability Engineering, 2005, pp. 213–222.
[102] V.U.B. Challagulla, F.B. Bastani, I.L. Yen, A unified framework for defect data analysis using the MBR technique, in: 18th IEEE International Conference on Tools with Artificial Intelligence, 2006, pp. 39–46.
[103] T.M. Khoshgoftaar, N. Seliya, N. Sundaresh, An empirical study of predicting software faults with case-based reasoning, Softw. Qual. J. 14 (2006) 85–111.
[104] P. Knab, M. Pinzger, A. Bernstein, Predicting defect densities in source code files with decision tree learners, in: Proceedings of International Workshop on Mining Software Repositories, 2006, pp. 119–125.
[105] M. Mertic, M. Lenic, G. Stiglic, P. Kokol, Estimating software quality with advanced data mining techniques, in: International Conference on Software Engineering Advances, 2006, pp. 19–22.
[106] M.E.R. Bezerra, A.L.I. Oliveira, S.R.L. Meira, A constructive RBF neural network for estimating the probability of defects in software modules, in: Proceedings of International Joint Conference on Neural Networks, 2007, pp. 2869–2874.
[107] W. Li, R. Shatnawi, An empirical study of the bad smells and class error probability in the post-release object-oriented system evolution, J. Syst. Softw. 80 (2007) 1120–1128.
[108] N. Seliya, T.M. Khoshgoftaar, Software quality estimation with limited fault data: a semi-supervised learning perspective, Softw. Qual. J. 15 (2007) 327–344.
[109] P. Tomaszewski, J. Håkansson, H. Grahn, L. Lundberg, Statistical models vs. expert estimation for fault prediction in modified code – an industrial case study, J. Syst. Softw. 80 (2007) 1227–1238.
[110] B. Yang, L. Yao, H.-Z. Huang, Early software quality prediction based on a fuzzy neural network model, 3rd International Conference on Natural Computing 2 (2007) 760–764.
[111] T. Zimmermann, R. Premraj, A. Zeller, Predicting defects for eclipse, in: Third International Workshop on Predictive Models in Software Engineering, 2007, pp. 9–15.
[112] W. Afzal, R. Torkar, A comparative evaluation of using genetic programming for predicting fault count data, in: 3rd International Conference Software Engineering Advances, 2008, pp. 407–414.
[113] K.K. Aggarwal, Y. Singh, A. Kaur, R. Malhotra, Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: a replicated case study, Softw. Process Improv. Pract. 14 (2008) 39–62.
[114] S. Bibi, I. Vlahavas, G. Tsoumakas, I. Stamelos, Regression via classification applied on software defect estimation, Expert Syst. Appl. 34 (2008) 2091–2101.
[115] A. Marcus, D. Poshyvanyk, Using the conceptual cohesion of classes for fault prediction in object-oriented systems, IEEE Trans. Softw. Eng. 34 (2008) 287–300.
[116] S. Shafi, S.M. Hassan, A. Arshaq, M.J. Khan, S. Shamil, Software quality prediction techniques: a comparative analysis, in: 4th International Conference on Emerging Technologies, 2008, pp. 242–246.
[117] W. Afzal, R. Torkar, R. Feldt, Search-based prediction of fault count data, in: 1st International Symposium on Search Based Software Engineering, 2009, pp. 35–38.
[118] Y. Jiang, B. Cukic, Misclassification cost-sensitive fault prediction models, in: Proceeding of 5th International Conference on Predictive Models in Software Engineering, 2009, pp. 20–30.
[119] A. Kaur, P.S. Sandhu, A.S. Brar, Early software fault prediction using real time defect data, in: 2nd International Conference on Machine Vision, 2009, pp. 242–245.
[120] T.M. Khoshgoftaar, K. Gao, Feature selection with imbalanced data for software defect prediction, in: International Conference on Machine Learning Applications, 2009, pp. 235–240.
[121] Y. Singh, A. Kaur, R. Malhotra, Prediction of software quality model using gene expression programming, Prod. Softw. Process Improv. 32 (2009) 43–58.
[122] W. Afzal, R. Torkar, R. Feldt, G. Wikstrand, Search-based prediction of fault-slip-through in large software projects, in: 2nd International Symposium on Search Based Software Engineering, 2010, pp. 79–88.
[123] L. Hribar, D. Duka, Software component quality prediction using KNN and fuzzy logic, in: Proceedings of 33rd International Convention (MIPRO), 2010, pp. 402–408.
[124] T. Illes-Seifert, B. Paech, Exploring the relationship of a file's history and its fault-proneness: an empirical study, in: Testing: Academic and Industrial Conference on Practical and Research Techniques, 2008, pp. 13–22.
[125] P.S. Sandhu, A.S. Brar, R. Goel, J. Kaur, S. Anand, A model for early prediction of faults in software systems, in: 2nd International Conference on Computer Automation Engineering, vol. 4, 2010, pp. 281–285.
[126] N. Nagappan, A. Zeller, T. Zimmermann, K. Herzig, B. Murphy, Change bursts as defect predictors, in: IEEE 21st International Symposium on Software Reliability and Engineering, 2010, pp. 309–318.
[127] N.H. Chiu, Combining techniques for software quality classification: an integrated decision network approach, Expert Syst. Appl. 38 (2011) 4618–4625.
[128] M. Masud, J. Gao, L. Khan, J. Han, B.M. Thuraisingham, Classification and novel class detection in concept-drifting data streams under time constraints, IEEE Trans. Knowl. Data Eng. 23 (2011) 859–874.
[129] P. Bangcharoensap, A. Ihara, Y. Kamei, K. Matsumoto, Locating source code to be fixed based on initial bug reports – a case study on the Eclipse project, in: 4th International Workshop on Empirical Software Engineering Practice, 2012, pp. 10–15.
[130] P.S. Bishnu, V. Bhattacherjee, Software fault prediction using quad-tree based *k*-means clustering algorithm, IEEE Trans. Knowl. Data Eng. 24 (2012) 1146–1150.
[131] M.D. Ambros, M. Lanza, Evaluating defect prediction approaches: a benchmark and an extensive comparison, Empir. Softw. Eng. 17 (2012) 531–577.
[132] L. Pelayo, S. Dick, Evaluating staratification alternatives to improve software defect prediction, IEEE Trans. Reliab. 61 (2012) 516–525.
[133] R.G. Ramani, S.V. Kumar, S.G. Jacob, Predicting fault-prone software modules using feature selection and classification through data mining algorithms, in: International Conference on Computational Intelligence & Computing Research, 2012, pp. 1–4.
[134] S.S. Rathore, A. Gupta, Investigating object-oriented design metrics to predict fault-proneness of software modules, in: CSI Sixth International Conference on Software Engineering, 2012, pp. 1–10.
[135] P. Singh, S. Verma, Empirical investigation of fault prediction capability of object oriented metrics of open source software, in: International Joint Conference on Computer Science and Software Engineering, 2012, pp. 323–327.
[136] A. Mahaweerawat, P. Sophatsathit, C. Lursinsap, Adaptive Self-organizing Map Clustering for Software Fault Prediction, 2007, pp. 35–41.
[137] A. Mahaweerawat, P. Sophatsathit, C. Lursinsap, Software Fault Prediction using Fuzzy Clustering and Radial Basis Function Network (2002) 304–313.
[138] G. Mauša, Search Based Software Engineering and Software Defect Prediction, 2010, https://www.fer.unizg.hr/_download/repository/Kvalifikacijski_-_Goran_Mausa.pdf.
[139] A. Monden, T. Hayashi, S. Shinoda, K. Shirai, J. Yoshida, M. Barker, K. Matsumoto, Assessing the cost effectiveness of fault prediction in acceptance testing, IEEE Trans. Softw. Eng. 39 (2013) 1345–1357.