Expert Systems With Applications 172 (2021) 114595

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Review

Machine learning based methods for software fault prediction: A survey

Sushant Kumar Pandey *, Ravi Bhushan Mishra, Anil Kumar Tripathi

Department of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi, India

ARTICLE INFO

Keywords:
Machine learning
Fault proneness
Statistical techniques
Fault prediction
Systematic literature review

ABSTRACT

Several prediction approaches are contained in the arena of software engineering such as prediction of effort, security, quality, fault, cost, and re-usability. All these prediction approaches are still in the rudimentary phase. Experiments and research are conducting to build a robust model. Software Fault Prediction (SFP) is the process to develop the model which can be utilized by software practitioners to detect faulty classes/module before the testing phase. Prediction of defective modules before the testing phase will help the software development team leader to allocate resources more optimally and it reduces the testing effort. In this article, we present a Systematic Literature Review (SLR) of various studies from 1990 to June 2019 towards applying machine learning and statistical method over software fault prediction. We have cited 208 research articles, in which we studied 154 relevant articles. We investigated the competence of machine learning in existing datasets and research projects. To the best of our knowledge, the existing SLR considered only a few parameters over SFP's performance, and they partially examined the various threats and challenges of SFP techniques. In this article, we aggregated those parameters and analyzed them accordingly, and we also illustrate the different challenges in the SFP domain. We also compared the performance between machine learning and statistical techniques based on SFP models. Our empirical study and analysis demonstrate that the prediction ability of machine learning techniques for classifying class/module as fault/non-fault prone is better than classical statistical models. The performance of machine learning-based SFP methods over fault susceptibility is better than conventional statistical purposes. The empirical evidence of our survey reports that the machine learning techniques have the capability, which can be used to identify fault proneness, and able to form well-generalized result. We have also investigated a few challenges in fault prediction discipline, i.e., quality of data, over-fitting of models, and class imbalance problem. We have also summarized 154 articles in a tabular form for quick identification.

1. Introduction

Software fault prediction models can conveniently identify software classes or modules that are more likely to be faulty. It minimizes the maintenance cost before deployment of the product (Seliya, Khoshgoftaar, & Van Hulse, 2010). Prior detection of the fault will lead to the flawless product and delivery of high-quality software projects. SFP is an essential aspect of the software development life cycle. The complexity and dependency of the software are enlarging as per the quality of the product. SFP is a high priority task to enhance the quality of the software product.

Several software practitioners (Lessmann, Baesens, Mues, & Pietsch, 2008; Arisholm, Briand, & Johannessen, 2010) claims that the performance of SFP models mostly depends on software metrics that we choose rather than modeling techniques, they also reported that selecting of prediction model offers a lesser impact over the performance on SFP. But according to a few latest articles (Tong, Liu, & Wang, 2018; Zhu & Pham, 2018) the performance of the SFP model mostly depends on the prediction methods. There are few specific situations where deep learning (DL) based SFP can be employed, such as when the dataset doesn't have extracted features set and/or class labels. As DL algorithms require hyperparameters tuning of many control parameters, so they are implicitly complex in nature. Hyperparameters tunning and optimization mechanism are two critical aspects of the DL approaches, and if software practitioner is not properly taking care of these aspects, then the model may perform poorly than the ML approach, which is not recommended. DL architectures are hunger for training instances; the more the training set, the more can be the accuracy. The performance of SFP models also depends on selection of datasets. Shepperd, Song, Sun, and Mair (2013) claims over the quality of various NASA data repository. Tong et al. (2018) suggested a hybrid SFP model using ensemble learning and deep representation techniques. Borandag,

* Corresponding author.
E-mail addresses: sushantkp.rs.cse16@iitbhu.ac.in (S.K. Pandey), mishravi.cse@itbhu.ac.in (R.B. Mishra), aktripathi.cse@iitbhu.ac.in (A.K. Tripathi).

https://doi.org/10.1016/j.eswa.2021.114595
Received 9 July 2020; Received in revised form 19 December 2020; Accepted 8 January 2021
Available online 15 January 2021

Ozcift, Kilinc, and Yucalar (2019) also suggested a hybrid feature section based SFP models by using info gain, RelieF, and systematical uncertainty and combined their results using majority voting. Both hybrid models outperform over the classical ML models.

The performance of SFP models depends on various factors. We collected those factors and analyzed them accordingly. The comparable analysis is required to examine all the parameters and also needs counsel guidelines according to empirical evidence. We examined all the related factors that affect the SFP model's performance, such as ML techniques, feature section/reduction, software metrics, datasets, performance measures, and dataset issues.

In this article, we conducted an Systematic Literature Review (SLR) is to depict the current state-of-the-art of software fault prediction approaches. We conducted SLR using a snowball sampling method (Goodman, 1961). This method is used to review the references of the primary studies that are selected from the databases. It is also used to examine the references of related studies, which were not included in the review of the primary studies (Malhotra, 2015; Catal & Diri, 2009b). There are many standard parameters available through which SFP models can be constructed. SFP models can be constructed using fault prediction datasets that consist of software metrics information, which can be obtained from a similar launched product. It will be unchallengeable for software practitioners to target the testing resources in the prior phase of software development. For example, if there is only a 40% testing support system is available, information about the poor areas helps the tester in concentrating the available resources, the module, or fault-prone classes. So high quality and maintainable product with low cost can be built at the estimated time. The advantages of SFP are as follows.

- The quality of the software improves as the test process improves.
- The test process improves by focusing on fault-prone class/modules.
- Highly dependable system can be achieved.
- Best design can be selected using object-oriented metrics.
- Refactoring candidates can be identified that are anticipated as fault-prone.

Soft computing is a powerful tool to resolve many real-life problems. It uses different techniques to find out the primary information form intractable and challenging problems. In the literature of ML, it has been widely applied in prediction faulty modules or classes. For example, Singh, Kaur, and Malhotra (2010) suggested an Artificial Neural Network (ANN) and Decision Tree (DT) predict modules/classes at the severe different level of faults.

The ease of use of machine learning-based techniques in SFP, it is compulsory to an organized summarization of the empirical conclusions obtained from existing studies and literature. The objective of existing surveys is to acknowledge the different research-oriented questions (Catal & Diri, 2009b; Catal, 2011; Radjenović, Heričko, Torkar, & Živkovič, 2013; Malhotra, 2015), but according to the author's survey, there is a limited organized survey that focused on the ML techniques over SFP. Malhotra (2015), and they considered study till 2013. Still, they only investigated 64 studies, even they also not considered the existing threats and challenges in their SLR. Wohlin, Mendes, Felizardo, and Kalinowski (2020) suggested guidelines for SLR, and we followed their procedures to inflate our review process. We have investigated the studies of the existing survey along with 64% additional studies in our SLR. We have added 45 studies from 2013 to 2019 and additional 45 articles before 2013, which are rudimentary investigated. We have concluded and provided guidelines after analyzing all the 154 articles. Most of the surveys focuses specific over single parameters. Radjenović et al. (2013) conducted a survey over software metrics of SFP domain. In this article we investigated over all the studies since 1990 to June 2019. We restricted the search from 1990 to 2019 as the machine learning techniques were launched in the 1990s. Before the 1990s, few debugging articles (Akiyama, 1971) were presented; their primary conclusion

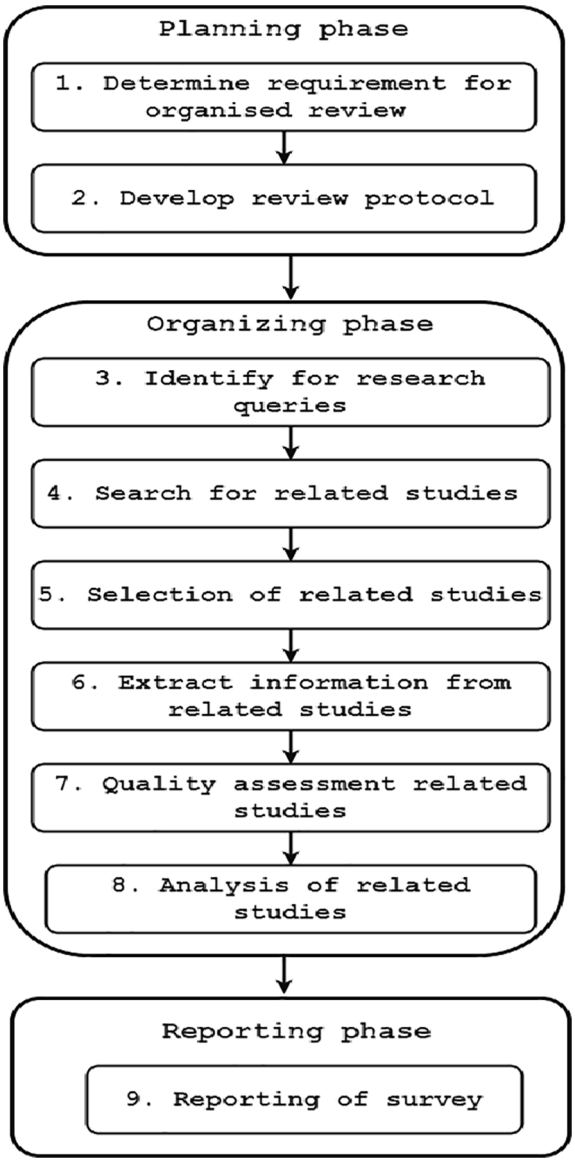


Fig. 1. Systematic literature review phases.

was covered in later articles. No doubt the conclusive evidence of the 1990s work also reported in the 2000s work, but it's better to revisit those earlier articles and explore all the conclusion, and it leads to a plethora of SFP works since 1990. According to our empirical study we recognized several ML techniques over SFP and we have followed the survey structure of Catal (2011) and Malhotra (2015). Li, Lessmann, and Baesens (2019) evaluated the performances of exiting SFP methods; they scrutinized 40 articles; they proposed a revised benchmarking over class distribution (sampling), performance metrics, and testing procedures, they specifically emphasized the performance of SFP. They mainly concluded that the results of the SFP mainly depend on projects. In our SLR, along with these points, we additionally covered few other sections, the various challenges of SFP, feature section/reduction performance comparison over statistical method, and strength & weakness of exiting SFP techniques. The contribution of the article is listed below:

I The article presents the systematic literature review over software fault prediction of 154 studies from 1990 to June 2019; we followed Keele guidelines for SLR. We also listed 154 articles in a tabular format for a quick review, and we also scored them according to the justification of research queries.

Table 1
Various research queries.

RQ-No.	Research queries	Motivation
RQ-1	What are the ML-based models which have been used for SFP till now?	Recollecting ML techniques usually used in SFP.
RQ-2	What are the different threats and challenges which are expecting from empirical validation using ML techniques in SFP discipline?	Need to investigate various threats and challenges.
RQ-3	What are the various feature selection/reduction techniques applied in SFP?	Identify techniques suitable for selection and reduction of software features.
RQ-4	Software metrics that are used in SFP??	Identification of metrics commonly used for SFP.
RQ-4.1	Which object-oriented metrics found thoroughly utility, partially utility, and not utility?	Identification of metrics which are appropriate and not appropriate for SFP.
RQ-5	What are the various datasets that are employed in the SFP arena and how to get those datasets?	Identification of appropriate dataset used for SFP, and how to get that dataset.
RQ-5.1	What are fault distribution in the dataset for SFP?	Identification of fault distribution in the dataset.
RQ-6	Performance analysis of various evaluation metrics in ML-based SFP models?	Performance measure of ML techniques to analyses metrics.
RQ-6.1	Comparison between of various ML techniques and classical statistical techniques over SFP arena?	Investigation of the performance of ML techniques and statistical techniques over SFP.
RQ-6.2	Which ML techniques are better and outperforms over exiting SFP?	Performance analysis of ML techniques over ML techniques.
RQ-7	What are the various strength and weakness of ML techniques?	Aggregation of information about ML techniques.

II We analyzed various factors that affect SFP's performance; to the best of our knowledge, there is rudimentary work has been done that also considered those factors. We also provided guidelines for software practitioners to create a useful SFP model over different conditions.

III We have also investigated various threats and challenges SFP discipline; the existing SLR rudimentary considered these provocations. We have also inspected the current solutions that subjugate these problems.

IV We also compared the performance between ML-based and statistical learning-based approaches, and we found the ML-based method outperforms in most scenarios.

The primary objective of this SLR is to summarize, examine and then evaluate the empirical evidence about: (1) ML techniques that have been applied over SFP domain, (2) potential contriving, and the ability for constructing SFP models, (3) comparative analysis of ML and statistical methods, (4) performance measurement, the accuracy and ROC of different ML techniques, (5) outline the strength and weakness of various ML techniques. We stated score value to the most relevant articles which address all of our research queries. We have summarized all the 154 relevant studies in tabular structure. This survey also delivers future guidelines for software practitioners and researchers about the implementation of different ML techniques over SFP. To accomplish the objective of this survey, we explore through eight digital libraries and discover 154 studies to answer the research queries (RQ) concerning the utility of the ML techniques for the SFP discipline.

We followed Keele guidelines (Keele, 2007) to perform SLR over software engineering, Fig. 1 demonstrates the Keele guidelines in our SLR; the recent works (Malhotra, 2015; Catal & Diri, 2009b) also applied similar procedures, so our SLR also based on existing literature. We organized the SLR in three different phases, planning, organizing, and reporting phase, as shown in Fig. 1. The development protocol is an essential phase in the SLR, and it minimizes the risk and bias results of distinct research.

In the planning phase, we have determined requirement specification for the survey, and then determine the protocol for review. The protocol is mainly rules which decide the flow of survey articles. In the organizing phase, we are identified various research queries, research articles related to the topic of study, and selection of most relevant articles. Information extraction form those articles, quality assessment studies, and analysis over-extracted information. In the last section, we report our survey by addressing the various research queries and extracting the conclusion from it. The Sections 3, and 4 are the part of organizing phase. We have listed few threats to validity in Section 5. The Section 6 belongs to reporting phase. In Section 2, we will discuss some related work in the SFP domain. In the appendix A, we have summarized 154 articles and scored them according to their justification of research queries.

2. Related work

Software fault prediction has been a tremendous impact on the software development life cycle; it reduces and allocates the required testing effort. Software practitioner utilized snowball sampling (Atkinson & Flint, 2001) techniques for review/survey article, and few of them (Nam, 2014; Abaei & Selamat, 2014) uses the database search method (Jalali & Wohlin, 2012). Table 2 presents a some of the recent similar work that has been done. Catal and Diri (2009b) published review article over SFP, and they have analyzed 74 research articles from 11 different journals until 2007. They categorized various machine learning-based approaches and software metrics, which were mainly utilized in SFP discipline. They followed the snowball sampling method (Atkinson & Flint, 2001) in their survey article.

Radjenović et al. (2013) analyzed different software metrics, which were applied in software fault prediction. They carried out SFP survey by employing the snowballing technique. They reviewed 106 research articles from 1991 to 2011, and concluded object-oriented metrics (49%) were mostly used, compared to other traditional metrics, i.e., code metrics (27%) or process metrics (24%), frequently used. Catal (2011) has performed a survey on 90 software fault prediction papers published between the year 1990 and 2009, they survey about metrics, methods, datasets, performance evaluation metrics, and experimental results perspectives in an easy and systematic manner.

Malhotra (2015) conducted SLR over the SFP domain, and they have investigated articles form 1991 to 2013 with 72 primary studies. They accomplished statistical analysis of various ML methods applied to fault prediction, software metrics, different datasets, and various performance measures. They followed the snowballing techniques for their survey. We have followed Malhotra (2015) article for analysis of our investigation, they have carried out an excellent review of 72 articles. Hall, Beecham, Bowes, Gray, and Counsell (2011) conducted an SLR of the performances of SFP methods. They investigated the correlation between the independent variables, modeling technique,and evaluation metrics. They included 208 SFP studies published from January 2000 to December 2010. They found models that outperformed tend to be based on simple modeling techniques such as Naive Bayes or Logistic Regression. They also reported that, the method used to construct predictive models seems to be influential to predictive performance. Wahono (2015) conducted an SLR over SFP over 71 defect prediction

Table 2
Comparison of existing work.

Article	# of articles & duration	Approach	Details
Rathore and Kumar (2019)	190 article, 1993 to 2018	statistical analysis, and database analysis	Group into three classes: software metrics, dataset quality and classification techniques.
Li et al. (2019)	40 articles, 2007-2018	Scrutnize exiting SFP studies, database search method	Mainly focused on class distribution sampling, evaluation metrics, and testing procedures.
Li et al. (2018)	70 article, Jan 2014 to April 2017	Comprehensive review, and statistical analysis	They summarized the review in four aspects: ML-based prediction, manipulating the data, effort-aware prediction and empirical studies.
Karim et al. (2017)	23 articles, 2005-2017	Classical tabular formation and statistical investigation	Investigated only those articles which applied PROMISE datasets and also examined various ML methods over these datasets.
Malhotra (2015)	72 articles from 1991-2013	Applied snowballing technique	Statistical techniques for analysis over software metrics, datasets, performance evaluator, etc.
Wahono (2015)	71 article from Jan 2000 to Dec 2013	Selection of important topic for SLR	Investigated over classification, clustering and association based SFP.
Nam (2014)	76 articles, till 2013	Database search method	Mainly introduce the SFP process used in existing work, compare the SFP techniques based on software features, learning algorithms, also explained about various cross-project fault prediction techniques.
Radjenović et al. (2013)	106 article, 1991 to 2011	Snowballing method	Investigated software metrics that are employed in SFP, also examined the variation in performance of SFP by considering various software features.
Catal (2012)	85 articles from 1994 2011	Statistical analysis over existing performance evaluation work	They group them into two parts, first about classification of faulty module and second about number of faults in every module.
Catal (2011)	90 articles, 1990 to 2009	Database search method along with statistical analysis	Investigated about software metrics, learning methods, datasets, and evaluation metrics.
Hall et al. (2011)	208 article, Jan 2000 to Dec 2010	Correlation analysis	Analyzed the correlation between independent variable and prediction model.
Malhotra and Jain (2011)	34 articles, 1993-2010	Statistical analysis	Specific for object oriented software projects and corresponding metrics.
Catal and Diri (2009b)	74 article, 1992 to 2007	Snowballing method	Categorized SFP method based over ML techniques and software features.

articles published between January 2000 and December 2013. They selected four top topics in their review article, i.e., association, classification, clustering, and dataset analysis.

Li, Jing, and Zhu (2018) surveyed over 70 articles from January 2014 to April 2017. They have only considered manuscripts of three recent years; their main objective is to analyze and summarize those 70 articles into four different aspects: datasets, learning algorithms, effort aware prediction, and empirical studies. Rathore and Kumar (2019) reviewed SFP from 1993 to 2018; they aggregated the study into three different classes, i.e., software metrics, fault prediction techniques, and dataset quality issues. For each class, they have presented taxonomic classification (other methods) and observations.

A review over performance evaluation metrics for SFA was carried out by Catal (2012). They investigated 85 SFP manuscripts and categorized them into two main groups according to performance metrics: first, metrics are utilized to evaluate the system's performance that only predicts faulty or not faulty modules, whereas second is for such systems that estimate the number of faults presents in modules. Malhotra and Jain (2011) presented a survey for object-oriented systems; the review is mainly based on software metrics and fault proneness. They mostly emphasize various object-oriented metrics, datasets, learning mechanisms, and performance measures. Singh, Panda, and Sangwan (2015) presented an analysis for SFP techniques; they have reviewed 15 most relevant articles in the survey and concluded that few software metrics could be useful for early fault prediction. Karim, Warnars, Gaol, Abdurachman, and Soewito (2017) performed an SFP literature review over the PROMISE dataset and corresponding metrics; they investigated 23 articles and several ML-based SFP methods.

3. Procedure

The planning, organizing, and reporting of the SLR executed in the paper is accomplished by the methodology suggested by Keele (2007). Each step that is required in SLR is presented in Fig. 1. In the planning phase, we developed a review protocol, then in organizing phase we framed seven research queries & select related studies and, finally, reporting our investigation in the last phase. The complete review analysis process includes identifying research queries, searching, & selecting related studies, extraction, and analysis of relevant studies. After the development of the protocol, we have carried out the following steps. Firstly we are trying to find out research queries, we searched and investigates for the explanations to those queries. In the next step, we are selecting & extracting related information; it also includes inclusion & exclusion of information regarding queries. In the next step, we quantified & analyzed the relevant information, and in the final phase, we are reporting our review. In the next subsections, we will discuss the databases from which we are extracting research articles, after that, we have framed research queries, and 13 standard assessment queries.

3.1. Research queries

The objective of SLR is to provide sufficient empirical evidence for SFP using several ML techniques. After a look over many relevant studies, we simplified various queries over SFP discipline. List of all Research Query (RQ) is shown in Table 1, we have framed seven full RQs, and those queries consist few subqueries concerning various SFP models. The RQ-4, and RQ-5 comprised of one full query and one subquery, whereas RQ-6 consists of one full query and two subqueries, as shown in Table 1. After a review, we discover various ML techniques that have applied for fault prediction (RQ-1). These question aims to recollect information about all ML techniques used for SFP arena till now. Empirical evidence of various threats and challenges using ML techniques over SFP domain in RQ-2. Technologies required for features selection and feature reduction in RQ-3. As features selection and modification is an indispensable phases in SFP. Software metrics have a crucial role in SFP, and different researchers use different metrics for SFP, RQ-4, RQ-4.1 associate over a study of software metrics. All the experiments required datasets, accessing datasets, and the quality of datasets is an essential aspect. The aim of queries RQ-5, RQ-5.1 is to addresses those questions that are related to datasets. Potential investigation of different ML techniques, and performance comparison between ML and statistical methods (RQ-6, RQ-6.1). Few ML techniques that are underperformed and outperformed, the analysis of performances of ML techniques (RQ-6.2). The strength and weakness of ML-

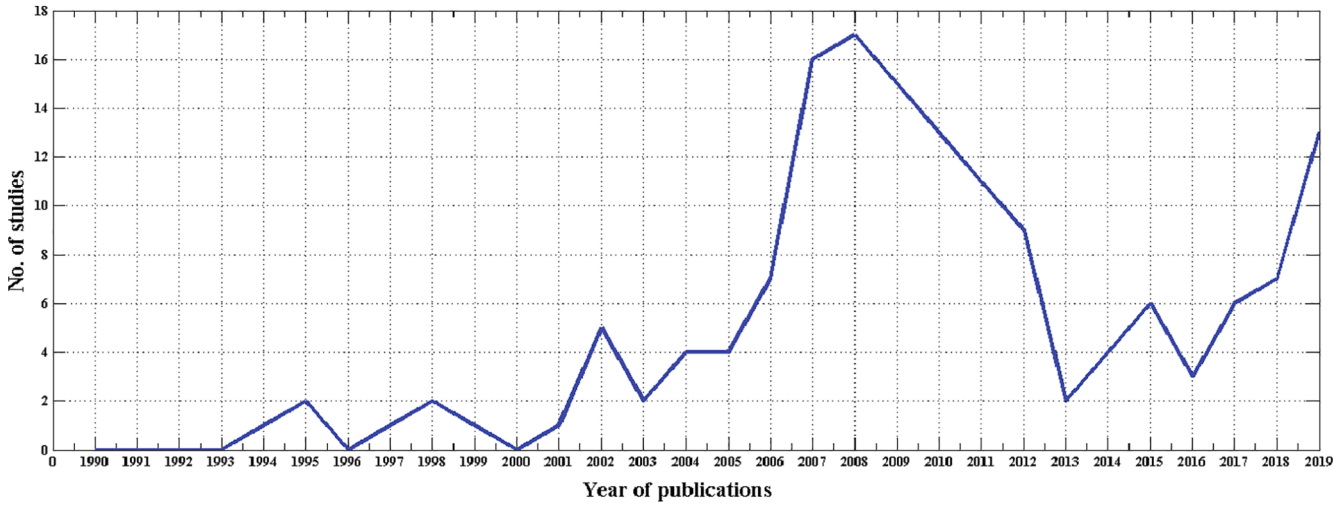


Fig. 2. Distribution of studies during various years.

based SFP techniques in RQ-7.

In the appendix Section A, we have summarized the most relevant 154 articles in tabular structure. The research articles have a unique attribute, i.e., Article Number (AN), which we have used to refer to any article. In those Tables (Table A.10 to Table A.13), we have mentioned authors & journal details, methodologies used, datasets, software metrics, and evaluation metrics. We have summarized article from AN-1 to AN-50, AN-51 to AN-95, AN-96 to AN-136, and AN-137 to AN-154 in Table A.10,A.11,A.12,A.13, respectively.

3.2. Search and selection of related studies

Searching relevant items and selecting those articles is a requisite phase of the SLR. We have employed the boolean expression 'AND' & 'OR'. The general search terms which are used as 'Software AND (error OR defect OR fault OR bug) AND (prediction OR probability OR proneness OR prone OR detection OR identification OR vulnerabilities) AND T,' where T is the techniques which include methods, 'machine learning' (Genetic Algorithm/Genetic Programming (GP) or Support Vector Machine (SVM) or Random Forest (RF) or Decision Tree (DT) or Bayesian Network (BN)) or 'soft computing' or 'regression' or 'data mining' or 'classification technique' or 'statistical method' or 'deep neural network' or 'ensemble learning' or 'boosting' or 'deep learning'.

The techniques on ML was derived from books and research papers (Mitchell, 1997; Witten, Frank, Hall, & Pal, 2016). After identification of search items, we selected eight major and famous digital library. A list of those digital libraries is given below.

i *ScienceDirect*
ii *IEEE Xplore*
iii *Scopus*
iv *Springer*
v *Wiley Online Library*
vi *Google Scholar*
vii *ACM Digital Library*
viii *Web of Science*

These electronic databases are accessible in our college, so the selection of research items is independent of the availability of digital libraries. By applying a target search string, an independent search was accomplished on eight electronic databases. As ML techniques introduced in 1990, so our search is restricted from 1990 to June 2019. An initial search begun regarding primary studies after determining the electronic database. During an initial search, the selection of relevant text document or full-text papers were performed by using inclusion and exclusion principle. We also considered those studies, which are from the bibliography section of the selected articles.

After investigating huge number of searched articles, we conducted the inclusion–exclusion principle to choose the most relevant articles from the searches items. Finally, we have chosen the 154 most relevant articles for SLR. These one hundred fifty-four research articles were determined for further analysis. The selected items are listed in four Tables of the appendix section (Table A.10 to Table A.13). The inclusion and exclusion principle is shown below.

Inclusion principle:

a. Experimental investigation reported using ML techniques for SFP.
b. Experimental investigation reported by associating non ML techniques.
c. Experimental investigation reported by comparing statistical techniques and ML-based prediction models.

Exclusion principle:

(a) Analysis without experimental study or conclusion of ML technique over SFP domain.
(b) Analysis of ML techniques considering other than SFP discipline.
(c) Analysis related to dependent variable besides fault-proneness.
(d) The result in the conference and extended version of a journal of the same author is similar; then, the analysis will continue.

If an article belongs to any of the criteria of inclusion, then it will be considered further, and if an article belongs to any criteria of exclusion, then it will be discarded for study. We didn't consider all the criteria need to be fulfilled to select or reject any article. The inclusion and exclusion principle was examined by three researchers unconventionally. To choose the appropriate studies, two semi screening were performed by two different researchers. Finally, after amalgamating semi-screening results, the third researcher conducted one final screening process to choose the most relevant 154 articles. All they accomplished a joint decision after detailed analysis. The relevant question of the researchers determines the standard of the study. The related studies with the similar result by the corresponding author was removed. After applying the above selection principle, 154 studies were chosen in the final phase. The quality assessment principle is given in the below section to achieve conclusive analysis.

We have plotted the number of articles versus their time of publication graph that are considered in our studies is shown in Fig. 2. Fig. 2 also reports the research impact over SDP throughout the year. The high peaks in the graph between 2006 to 2010, here most of the hybrid SFP

Table 3
Various standard assessment queries.

Q-No.	Standard queries	No	Partly	Yes
Q-1	Is Objective of the research is ultimately stated?			
Q-2	Every independent variable adequately defined?			
Q-3	Size of the dataset is adequate?			
Q-4	Procedure of data collection is adequately elaborated?			
Q-5	Are suitable metrics are selected?			
Q-6	Definition and justification of ML techniques defined correctly?			
Q-7	Performance measure and result analysis are correctly stated?			
Q-8	Are Limitation of studies are mentioned?		=	
Q-9	Is repetition of the methodology is available?			
Q-10	Are comparative study of ML and statistical method available?			
Q-11	Are comparative study of different ML techniques available?			
Q-12	Does review add or contribute to literature?			
Q-13	Does review have the sufficient number of citation per year?			

models were proposed, and various ensemble learning models were also introduced. Those hybrid and ensemble learning based models outperform over the classical SFP model. After 2016 the number of studies increases as shown in Fig. 2, it is because of the application of deep learning in the SFP domain.

3.3. Quality assessment of related studies

We have organized standard queries for assessing the appropriateness and robustness of the relevant studies. The standard survey evolved by considering the recommendation of Wen, Li, Lin, Hu, and Huang (2012). Table 3 describes the list of quality assessment queries. The queries are evaluated by score, 0 (no), 0.5 (partly) & 1 (yes). After the addition of the evaluated values of each query, the endmost grade is secure. The maximum score is 13, and the minimum score is 0 for any of the articles.

Three self-sufficient practitioners give a grade to the standard questions for primary queries and discuss each practitioner in case of disapproval. After an in-depth analysis and a lot of discussions, and inclusion & exclusion principle for every study was made.

3.4. Analysis of related study

The purpose of analyzing each study is to identify which research queries were convinced by initial studies. We outline the title, author name, publishing information, dataset details, ML technique, and metrics used (independent variable). The explanations of each research queries are given in the analysis card. It is used to store every information regarding the primary study. The primary objective of the analysis is to summarized (facts and figures), formulate, and resolve each research queries (Wen et al., 2012). Collection of similar reviews and extract the specific part of research queries.

We investigated and evaluated the quantitative data, incorporating the utility of different performance measures such as AUC, prediction accuracy, etc. The qualitative analysis includes strengths & weaknesses of ML techniques, classification of various methods, datasets, feature selection/reduction used in the SFP discipline. To explain every research question, we used different types of line-graph, pie-chart, and bar-chart, etc.

4. Analysis and discussion

In this section, we will discuss the results obtained from various studies regarding SFP. Firstly, we describe the summary of the primary research, besides their corresponding ranking. The ranking is given by the score, which is the estimation of quality assessment queries. Secondly, every subsection delivers the explanation of each research review questions. At last, by the facts and information that extracted from our primary study, we interpreted and concluded.

Table 4
Publications summary.

Name of publication	Type	Impact factor	No. of studies
IEEE Transactions on Software Engineering	Journal	6.11	12
Information Science	Journal	5.91	4
Expert systems with applications	Journal	5.45	9
Applied Soft Computing	Journal	5.41	4
IEEE Transactions on Knowledge and Data Engineering	Journal	4.93	2
Empirical Software Engineering	Journal	3.15	10
Information and Software Technology	Journal	2.72	12
Journal of Systems and Software	Journal	2.45	11
Automated Software Engineering	Journal	1.85	2
Software Quality Journal	Journal	1.46	5
International Joint Conference on Neural Network (IJCNN) and Applications	Conference	-	2
METRICS, IEEE	Conference	-	2
International Symposium on Software Reliability Engineering (ISSRE) - IEEE	Conference	-	5
International Conference on predictive models and data analytics in software engineering (PROMISES)	Conference	-	3
International Conference in Software Engineering (ICSE), IEEE	Conference	-	2
Computer Languages, Systems and Structures	Journal	-	2

4.1. Explanation of primary study

In this section, we will explain the chosen relevant studies. We selected 154 original research articles (according to the quality of the question), which uses ML techniques for the SFP domain. Most of the studies based on open sources journal and top conferences. We deeply study the content of selected articles, then extracted all the essential material from them. Afterward, we try to address the answer of every research query, and at the end, we will give the score to every article based on the justification of standard assessment query (see Table 3).

4.1.1. Source of publications

We have summarized the list of the top 88 publications in Table 4. The listed venues (conferences, journals, or book chapters) are more influential and consist of at least two published articles. The publication list is sorted according to the value of the impact factor (current date). Most of the studies are from IEEE Transactions on Software Engineering, which contributed 12 number of studies. We selected 11 and 12 articles from each of the Journal of System Software and Information and Software Technology Journal, respectively. We selected 10 articles from Empirical Software Engineering and 9 articles from the Expert System with an Applications journal. We also chosen 5 articles from ISSRE and 2 articles ICSE conference and so on. The rest of the 66 items are also from other reputed venues.

4.1.2. Quality assessment related study

The score is evaluated according to the justification of quality assessment queries. The quality of the articles is divided into four categories, excellent ($13 \geqslant$ score > 10), good ($10 \geqslant$ score > 8), average ($8 \geqslant$ score > 5) and below-average ($5 \geqslant$ score > 0). The maximum score is 13, and the minimum is 0. The 154 articles that we have considered for our primary studies is the list in the appendix section from Table A.10 to Table A.13. All four Tables consist of a unique attribute and other technical details.

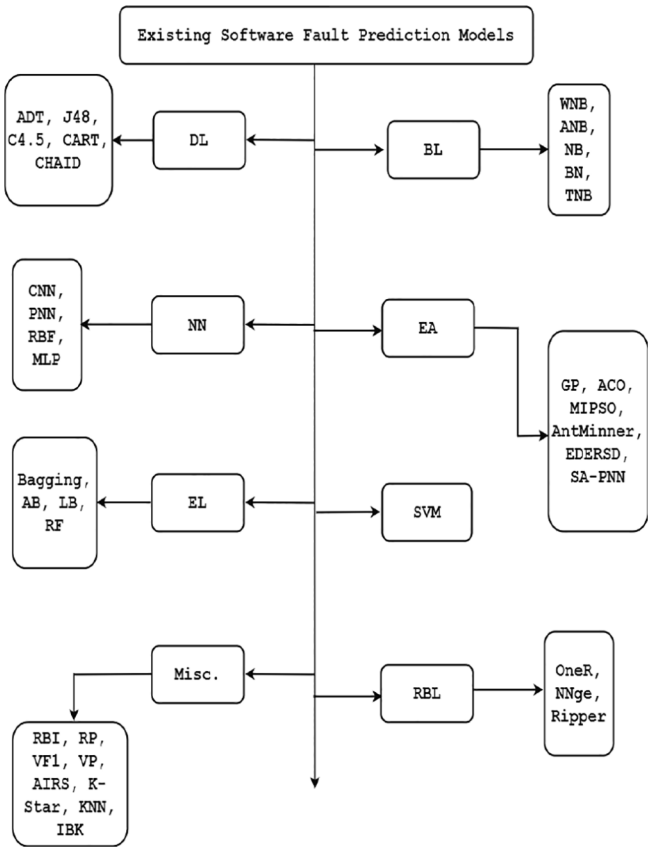


Fig. 3. Anatomy of various Machine learning techniques that are used for SFP. CHAID: CHi-squared Automatic Interaction Detection, CART: Classification and Regression Trees , ADT: Alternating Decision Tree, RF: Random Forest, NB: Na've Bayes , BN: Bayesian Networks, ABN: Augmented Bayesian Networks , WNB : Weighted Bayesian Networks, TNB: Transfer Bayesian Networks, MLP: Multilayer Perceptron , PNN: Probabilistic Neural Network, RBF: Radial Basis Function , LB: Logit Boost , AB: Ada Boost , NNGE: Neighbor With Generalization, GP: Genetic Programming , ACO: Ant Colony Optimization, SVM: Support Vector Machine s, 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 Probabilisti c Neural Network, VP: Voted Perceptron.

4.2. Justifications of research queries

In this section we will address each RQs that we have discussed in Section 3.1.

4.2.1. RQ-1: What are the ML-based models which have been used for SFP till now?

We have grouped various ML techniques that have been used in SFP discipline into eight categories are shown below.

(i) Bayesian learners (BL)
(ii) Decision Tree (DT)
(iii) Evolutionary Algorithm (EA)
(iv) Ensemble Learners (EL)
(v) Neural Networks (NN)
(vi) Support Vector Machine (SVM)
(vii) Rule Based Learning (RBL)
(viii) Miscellaneous

The anatomy of several ML techniques that have been applied in the SFP domain is shown in Fig. 3. We have categorized various subclasses of ML-based methods. *Note:* Caption of Fig. 3 also reports the full form of linked ML methods. Table 5 listed the categories of ML classes with their corresponding number of studies. Table 5 reports that Miscellaneous, DT, and BL are the most frequent techniques used in the SFP domain, with 47, 43, and 41, receptively number of studies. Few articles used more than one method due to that the sum of the percentage of the Table is more than 100, and the total quantity of the article is more than 154. Arisholm, Briand, and Fuglerud (2007) uses NN, DT, EL, and statistical model over JAVA based projects in their studies. Gyimothy, Ferenc, and Siket (2005) utilized NN and DT over Mozilla data in their studies. Singh et al. (2010) also used DT, NN, Logistic Regression (LR) over the NASA dataset, and achieve optimal results over baseline methods. Fig. 4 and Fig. 5 represents classification of ML techniques used for SFP till now. Fig. 4 indicates the number of studies employed various distribution of subclasses of NN, EA, BL, EL, DT, RBL, & miscellaneous classification techniques. In contrast, Fig. 5 represents their corresponding percentage of studies through which they utilized these ML techniques. Fig. 5 precisely indicates which classification models are mostly used in SFP. The articles that utilize subclass techniques of NN, EA, BL, EL, DT, RBL, and Miscellaneous ML models are shown in the pie-chart structure from Fig. 5a to Fig. 5g, respectively. The maximum contribution produced by subclass techniques of NN, EA, BL, EL, DT, RBL, and Miscellaneous methods is MLP (80%), HYBRID (36%) & GP (36%), NB (66%), RF (56%), C4.5 (43%), RIPPER (46%), and KNN (19%), respectively.

Table 5
Number of studies of Machine learning based software fault prediction techniques.

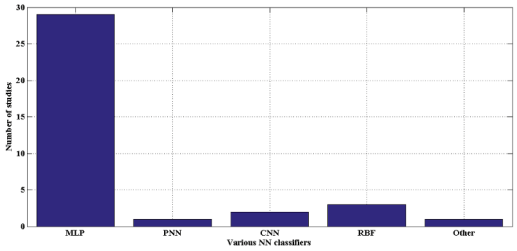
ML techniques	Number of studies	% of study
BL	41	27.15
DT	43	28.47
EA	20	13.24
EL	16	10.59
NN	36	23.83
SVM	25	16.55
RBL	7	4.6
Miscellaneous	47	31.12

Fig. 4 shows the bar-graph that indicates the number of studies that use various subclasses of ML techniques in SFP discipline. Fig. 4a to Fig. 4g shows the study count of NN, EA, BL, EL, DT, RBL, Miscellaneous based SFP models. The overall most frequently uses ML techniques over SFP are MLP, NB, SVM, C4.5, RF, & DT with number of studies are 29, 27, 25, 18, 9, &, 9, respectively as shown in Fig. 4.

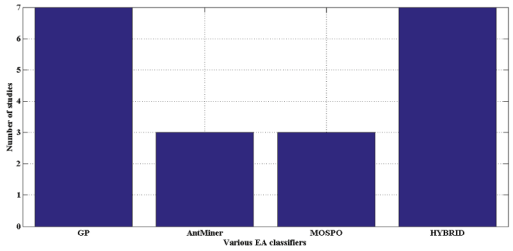
Elish and Elish (2008) applied SVM over the NASA dataset and achieve a high probability of detection.; they also concluded the SVM outperforms over LR model. Turhan and Bener (2007a) uses multivariate BN over the NASA dataset to deliver high specificity; they reached the results is outperformed over statistical technique. Pandey, Mishra, and Tripathi (2020) utilizes heterogeneous EL over 12 NASA datasets and concluded, high ROC over most of the state of the arts, and EL subjugate overfitting problem. Arisholm et al. (2010) proposed an SFP model by demonstrating variation of C4.5 over public data and found the proposed approach surpasses performance over classical SVM and LR based SFP model.

4.2.2. RQ-2: What are the different threats and challenges which are expecting from empirical validation using ML techniques in SFP discipline?

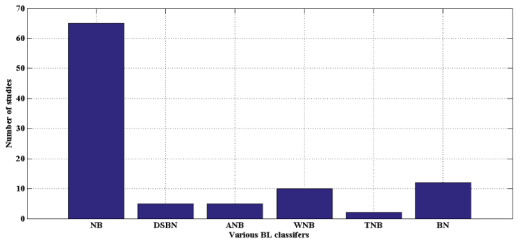
ML-based SFP model detects the faulty and non faulty modules over training set and validated the results of test or development set. The development of SFP is a sequential process. Software practitioners first collect datasets for various data repositories. After that, they do data preprocessing such as data cleaning, filling missing values, normalization of data, removal of duplicate instances, etc. Then they can apply sampling techniques (Seiffert, Khoshgoftaar, & Van Hulse, 2009) and feature selection method (Khoshgoftaar & Gao, 2009) to extract features (software metrics). Most of the fault prediction datasets are imbalanced datasets (Khoshgoftaar, Gao, & Seliya, 2010), i.e., more non-faulty instances and few faulty instances. The sampling techniques are also used



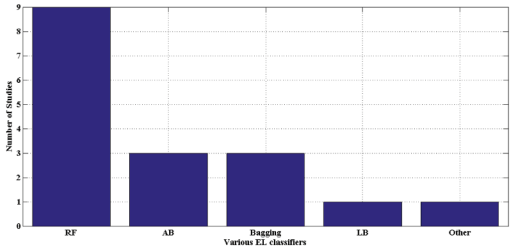
(a) Bar graph of Neural network.



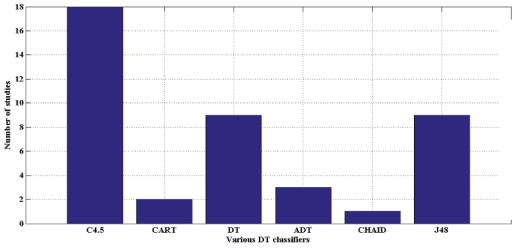
(b) Bar graph of Evolutionary algorithm.



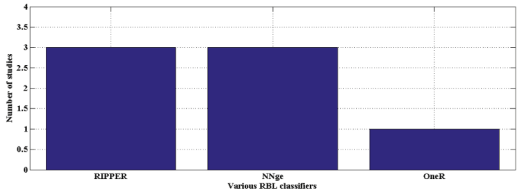
(c) Bar graph of Bayesian learner.



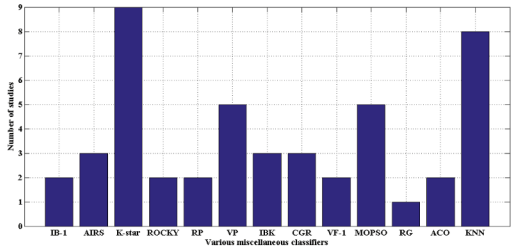
(d) Bar graph of Ensemble learners.



(e) Bar graph of Decision trees.



(f) Bar graph of Rule based learning.



(g) Bar graph of Miscellaneous.

Fig. 4. Number of studies used by various classes of ML techniques.

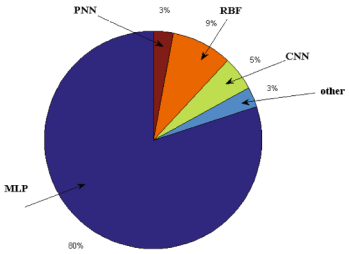
to address the class imbalance problem. Overfitting Hawkins (2004) is also a prominent challenge in SFP domain. ML models outperforms over training set but fails perform over test set. Few researcher suggest EL based model (Huda et al., 2018; Rätsch, Onoda, & Müller, 1998) and k-fold cross validation (Bengio & Grandvalet, 2004) approach to conquer this problem. Regularization techniques (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) also applicable in NN based model.

Shepperd et al. (2013) claims over the quality of the NASA datasets. Kim, Zhang, Wu, and Gong (2011) claimed that collected data contain noise because the current fault collection practices are based on optional keywords, and faulty instances are not properly linked with changelogs. Since the quality of changelogs is varying across the projects. Kumar, Sripada, Sureka, and Rath (2017) claimed that in some projects, the fault information is not available in the source. Although the cost parameter can be incorrectly collected. Wagner (2006) suggested a few techniques to handle cost parameter issues. Software projects which are developed within the organization (banking) may contain different fault pattern, and the results of the SFP technique can be varied if model directly applied without analyzing the fault pattern (Allen, 2002).

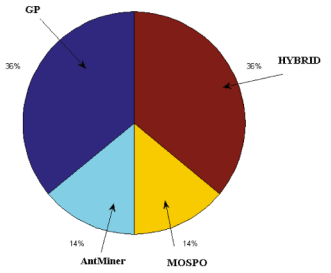
The performance of the existing SFP depends upon the system, and the tunning of parameters/hyperparameters plays a critical role in the performance of the SFP model. There is limited research has done over the tunning of these parameters/hyperparameters. Most of the software industries don't reveal their project associated datasets. As most of the datasets are classical datasets, more accessible datasets leads to diverse results. An appropriate selection of performance measures is required. The accuracy is not adequate for imbalanced datasets, it does not measure the effects of negative classes, whereas the Precision-Recall curve (Saito & Rehmsmeier, 2015) or F-score are more efficient in such scenarios. Few researchers (He & Garcia, 2009; de Carvalho, Pozo, Vergilio, & Lenz, 2008) suggested ROC is somehow sufficient form noise and imbalanced datasets.

4.2.3. RQ-3: What are the various feature selection and feature reduction techniques that are applied in SFP using ML techniques.

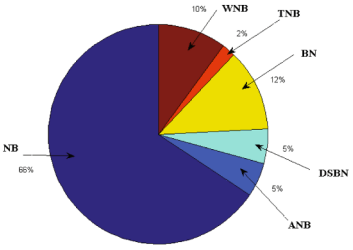
The selection of relevant features and reduction in the size of features are two indispensable phases to get a more optimal result. Primarily feature reduction involved feature selection and feature extraction. Feature selection included the selection of most relevant features



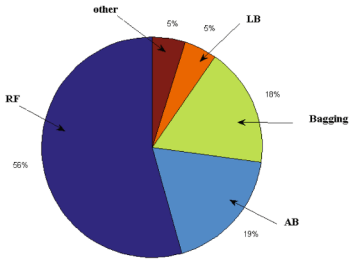
(a) Neural network based SFP techniques.



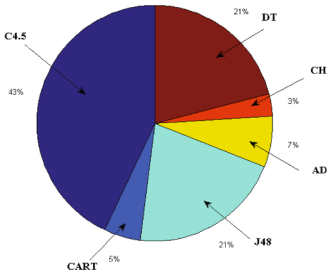
(b) Evolutionary algorithm based SFP techniques.



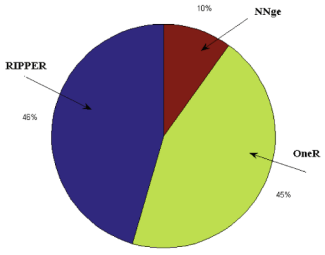
(c) Bayesian learner based SFP techniques.



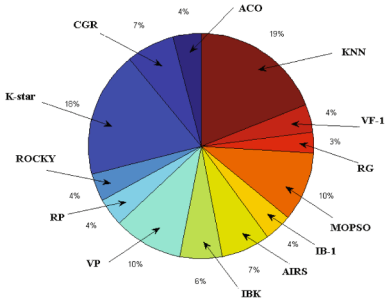
(d) Ensemble learners based SFP techniques.



(e) Decision trees based SFP techniques.



(f) Rule based learning based SFP techniques.



(g) Miscellaneous methods based SFP techniques.

Fig. 5. Distribution of sub classes of NN, EA, BL, EL, DT, RBL, & Miscellaneous classification techniques with their corresponding percentage (%) of studies.

required for analysis, and feature extraction involved, the combing of similar elements into a new smaller feature. In our primary investigation, 56% of studies are about feature selection/reduction.

Co-relation based Feature Selection (CFS) (Hall, 2000) is the most common feature selection technique in all related studies. The CFS technique removes noisy and unnecessary features, and keep only those features, which are mostly co-related. These techniques were used in AN-18, AN-21, AN-41, AN-42, AN-52, AN-66, AN-57, AN-59, and AN-62. Besides CFS, some other techniques were also applied in feature selection, such as the Infogain method. It is used in articles AN-12, AN-15, AN-31, AN-54, AN-66, AN-153, and AN-154. The χ^2 based filter used in AN-27, and AN-54, co-relation analysis in AN-14, and wrapped attribute selection used in AN-57, AN-58, and AN-66. Few less common feature selection techniques such as Markov Blanket used in AN-68, the consistency based selection used in AN-57, and AN-66, chi-square used in AN-22, the random forest used in AN-3. Principal Component Analysis (PCA) regarding feature extraction (AN-10, AN-15, AN-19, AN-28, and AN-48) very few works have done, and Isotamp was applied in AN-15. AN-154 uses the hybrid feature section method by using info gain, symmetrical uncertainty, ReliefF, and combined theirs by majority voting. The deep learning-based feature selection method is used in AN-139 and AN-147.

Sampling techniques were also applied in the preprocessing step of the SFP method. Many over-sampling, under-sampling, or hybrid techniques were utilized. Articles AN-74, AN-75, and AN-129 used SMOTE as a sampling technique, AN-75, AN-99, and AN-125 used random sampling. Attribute exhibiting with zero variance were applied in AN-67, feature ranking used in AN-68, and AN-99. In an article AN-73, the chi-squared and clustering attribute selection was used. Missing values remove were used in AN-82, AN-87, AN-89, and AN-95, Logarithmic transformation (AN-101), median adjustment class label (AN-110), K-means clustering (AN-111, and AN-73), Wrapper based subset method (AN-99, and AN-132). Few latest techniques, such as Zero factoring (AN-128), Local tangent space alignment (AN-133), Adaptive SMOTE (AN-134), were also being used as a feature selection/reduction method.

4.2.4. RQ-4: Software metrics that are used in SFP?

The analysis of software metrics is considered as the necessary and independent phase in our study. We categorize various software metrics that are employed for fault prediction, also described in (Radjenović et al., 2013), and the categories are given below.

- *Study about object-oriented metrics:* This includes study about metrics of different characteristics of object-oriented software such as inheritance, cohesion, and coupling of object class (Chidamber & Kemerer, 1994; Aggarwal, Singh, Kaur, & Malhotra, 2006; Lorenz & Kidd, 1994). LOC (line of code) is used as traditional metrics, but we have also considered SLOC (source lines of code) in our SLR. Articles AN-2, AN-5, AN-6, AN-7, AN-8, AN-10, AN-14, AN-17, AN-21, AN-32, AN-33, AN-43, AN-44, AN-49, AN-50, AN-52, AN-53, AN-61, AN-62, AN-81, AN-85, AN-96, AN-97, AN-105, AN-110, AN-114, AN-115, AN-116, AN-120, AN-124, AN-125, AN-131, AN-132, AN-135, AN-141, AN-150, AN-151, and AN-152 includes study about SLOC/LOC

  An article AN-81 used few other code metrics such as MLOC (LOC executable), PAR (number of parameter), NOF (number of attributes), NOM (number of methods), NSF (number of static attributes), NSM (number of static methods), NBD (nested block depth). Articles AN-25, AN-35, AN-70, and AN-81. AN-81 proposed a novel approach to rank more fault-prone modules using code mining, change history mining, and text mining to locate the faulty file rapidly. They used few change history metric, those are CodeChurn (sum of added lines of code - deleted lines of code), LOCAddes (sum over all revisions of the lines of code added to a file), LOCDeleted (sum over all revisions of the lines of code deleted from a file change), Revision (number of revisions of a file history), Age (age of a file in weeks), BugFixes (number of times a file was involved in a bug-fix transaction), Refactorings (number of times a file has been refactored), LastModDate (number of days a file has been latest modified). A bug tracking system article in AN-82 proposed an RBF based neural network technique to predicting errors in a fault-prone module. Their approach inform the probability of fault in the module, instead of just reporting module is faulty or not. They employed bug tracking metrics as, FP (function points), mts (max team size), Date (implementation date), EI (external inquiry), EO (external output), EE (external inquiry), ILF (internal logical), EIF (external interface files), Added (added lines of code),Changed (changed lines of code), Deleted (deleted lines of code).
- *Study about procedural metrics:* This includes conventional static code metrics, which are elaborated by Halsted (Halstead, 1977) and McCabe (McCabe, 1976), it also includes LOC metrics. Study section which lie into this categories are AN-2, AN-3, AN-11, AN-12, AN-12, AN-13, AN-15, AN-16, AN-18, AN-19, AN-20, AN-22, AN-23, AN-24, AN-25, AN-27, AN-28, AN-29, AN-31, AN-43, AN-46, AN-44, AN-48, AN-51, AN-54, AN-55, AN-57, AN-58, AN-59, AN-61, AN-64, AN-66, AN-68, AN-81, AN-82, AN-87, AN-88, AN-89, AN-90, AN-95, AN-96, AN-99, AN-104, AN-107, AN-112, AN-113, AN-119, AN-123, AN-127, AN-128, AN-129, AN-130, AN-131, AN-134, AN-136, AN-138, AN-139, AN-141, AN-142, AN-145, AN-146, AN-151, AN-152, AN-153, and AN-154
- *Study about hybrid metrics:* It includes both object-oriented metrics and procedural metrics for SFP discipline. Studies which come into this category are AN-26, AN-42, AN-51, AN-63, AN-65, AN-79, AN-83, AN-91, AN-92, AN-110, AN-111, AN-119, AN-126, AN-137, AN-140, AN-143, AN-150, AN-151, AN-152, and AN-154
- *Study about miscellaneous metrics:*This include network metrics (AN-38) extracted from dependency graph, requirement metrics (AN-9), change metrics (AN-25), fault slip through metrics (AN-40), process metrics (AN-41), churn metrics (AN-39), file age, size, changes and faults in previous versions metrics (AN-45), elementary design evolution metrics (AN-56), and other miscellaneous metrics that are not included under categories of either procedural or object oriented or others metrics. Some of the studies that utilized miscellaneous metrics are AN-94, AN-98, AN-99, AN-100, AN-100, AN-101, AN-109, AN-112, AN-118, AN-121, AN-149, and AN-154

The most frequently used Object Orientated (OO) and miscellaneous metrics are NM (number of methods), INST (number of instance variables), INTR (number of interfaces implemented), MI (maintainability index), FOUT (fan-out), RFC (response for a class), NSUB (number of children classes), NSUP (number of ancestor classes), MAXCC (maximum cyclomatic complexity), LCOM (lack of cohesion), NOC (Number of Children), DIT (Depth of Inheritance), WMC (Weighted Methods per Class),TOp (Total Operator), UOp (Unique Operator), UOpnd (Unique Operand) the articles AN-7, AN-21, AN-29, AN-37, AN-40, AN-42, AN-44, AN-49, AN-52, AN-54, AN-61, AN-62, AN-64, AN-74 and AN-75, AN-81, AN-85, etc. utilizes such metrics. The article AN-74 conducted experiments over OO metrics based datasets and tried to find the interaction between the sampling method, classifier, and software metrics. They found imbalance learning should be applied when the dataset is moderate of highly imbalanced. They also concluded the selection of imbalance learning is dependent on software metrics.

The articles AN-40, AN-90, AN-91, AN-92, AN-96, AN-102, AN-137, and AN-143 utilizes other miscellaneous software metrics and those are: SF-CR No. of faults slipping from (SF,) code review (CR), SF-MT (No. of fault slipping from (SF) module test (MT)), SF-MDR (No. of faults slipping from (SF) module design review (MDR)), ST-I (No. of faults slipping to (ST) integration level (I)), ST-S (No. of faults slipping to (ST) system level), SF-R No. of faults slipping from (SF) review (R), SF-U (No. of faults slipping from (SF) unit level (U)), SF-HD No. of faults slipping from (SF) hardware development (HD), SF-F (No. of faults slipping from (SF) function level (F)). An article AN-102 employed Case-Based Reasoning (CBR), they also evaluated the advantages of applying metric-selection operation over the CBR system.

The studies AN-79, AN-85, AN-143, and AN-149 used few rare cohesion and coupling metrics and those are: ACAIC (counts all clas attribute interactions from class c to ancestors of class c), ACMIC (attribute class method interaction), AMMIC (attribute method method interaction), CBO, CBO1 (Coupling between objects), DAC/DAC1 (data abstraction coupling), DCAEC (counts all class attribute interactions from descendants of class c to class c), DCMEC (data class-method interaction), DMMEC (data method method interactions), ICP (Information-flow-based coupling), IHICP (count inheritance based coupling), MPC (Message passing coupling), NIHICP (count non-inheritance based coupling), OCAEC (counts all class attribute interactions to class c from classes that are not friends or descendants of class c.), OCAIC (counts class attribute interactions from class c to classes that are not ancestors or inverse friends of class c), OCMEC (object class-method interaction), OCMIC (class method interaction), OMMEC (object mthod method interaction), OMMIC (object method method interaction coupling), RFC,TCC (tight class cohesion) LCOM1 (Lack of cohesion), LCOM2, LCOM3, LCOM4. The study AN-149 applied association rule mining over procedural metrics and found the ROC reached 81.1%, and their suggested model outperformed over exiting baseline methods. An article

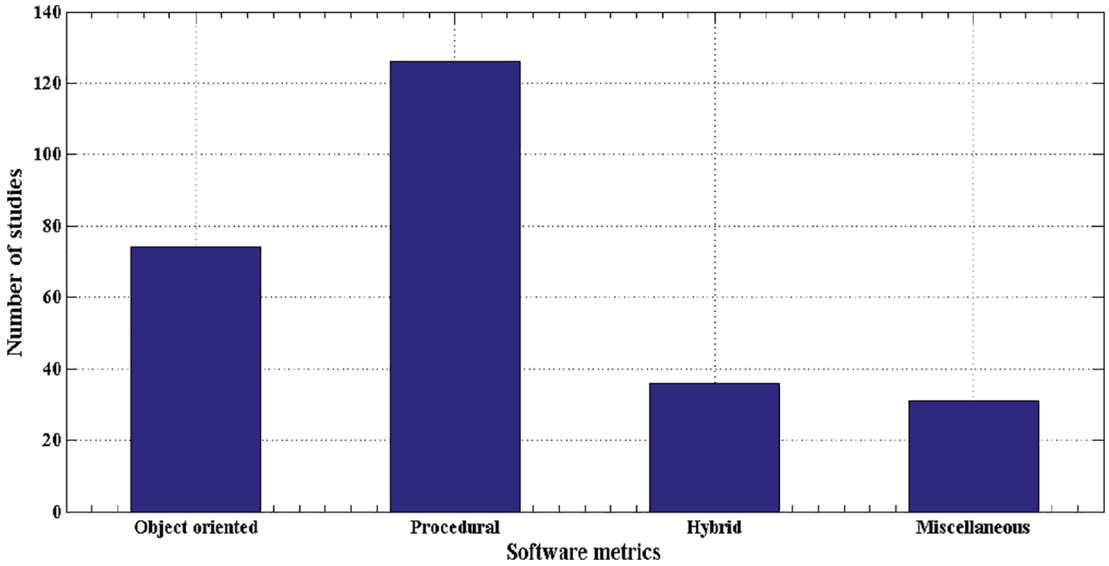


Fig. 6. Number of studies versus the software metrics that have been used in SFP.

AN-143 utilized a fuzzy inference system over coupling between object, LOC and, cyclomatic complexity of KC1 (PROMISE repository); they achieved maximum AUC, i.e., 0.8181.

Some additional miscellaneous metrics are Knots (number of knots), Eknots (essential number of knots), NEST (maximum number of nesting levels), Blocks (number of basic blocks), Exits (procedure exits, i.e., number of distinct calls or exits made within procedures.) were used by AN-91. Few other rare software metrics are which have used by AN-92 number of unique operator (η_1), Number of unique operands (η_2), Extended cyclomatic complexity ($V_2(g)$). The article AN-92 accomplished an empirical study over the validation of multivariate models. They concluded that the suitability of multivariant models could predict the faulty modules/classes of various software packages. Some call graph metrics and control flow graph metrics were also being used in defect prediction by AN-100. They applied CART, CART-LS, other variations of CART combined with PCA call graph metrics and found optimal results. Call graph metrics are UCT (Unique procedure calls), TCT (Total calls to others), NDI (Distinct files included) whereas Control flow graph metrics are VG (McCabe's cyclomatic complexity), NL (Number of loops), IFTH (Number of if-then structures), NELTOT (Total nesting level), PSCTOT (Total number of vertices within the span of loops or if-then structures), RLSTOT (Total edges plus vertices within loop structures).

PL/SQL software metrics were also being used for defect prediction in article AN-111, those metrics are NSC (number of select commands), NIUO (number of insert/update operations), NDO (number of delete operations), NT (number of table in from-clause), NSCC (number of search condition criteria in where-clause), NJQ (number of join-queries criteria). They applied the ward neural network and the general regression neural network; they considered it a regression problem and evaluated low error with high accuracy over the existing state of the art methods.

The procedural metrics are most frequent metrics in SFP as shown in Fig. 6, whereas OO metric is the second most widely employed software metric. The number of articles that utilizes procedural and OO metrics are 126, and 71 respectively. Whereas hybrid and miscellaneous metrics have least contributions. Note: Several studies used more than one kind of software metrics, due to which sum of studies in Fig. 6 is more than 154.

4.2.5. RQ-4.1: Which object-oriented metrics found thoroughly utility, partially utility, and not utility?

As we have reported in RQ-4, OO metrics are the second most widely software metrics in the SFP domain. Nowadays, most of the software

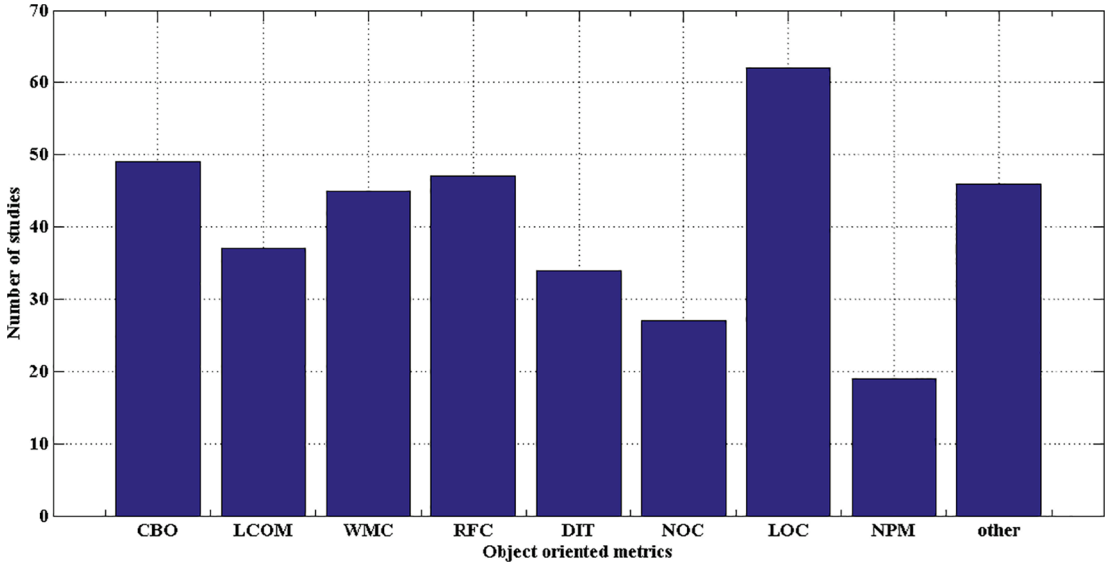


Fig. 7. List of object oriented software metrics that are utility for SFP.

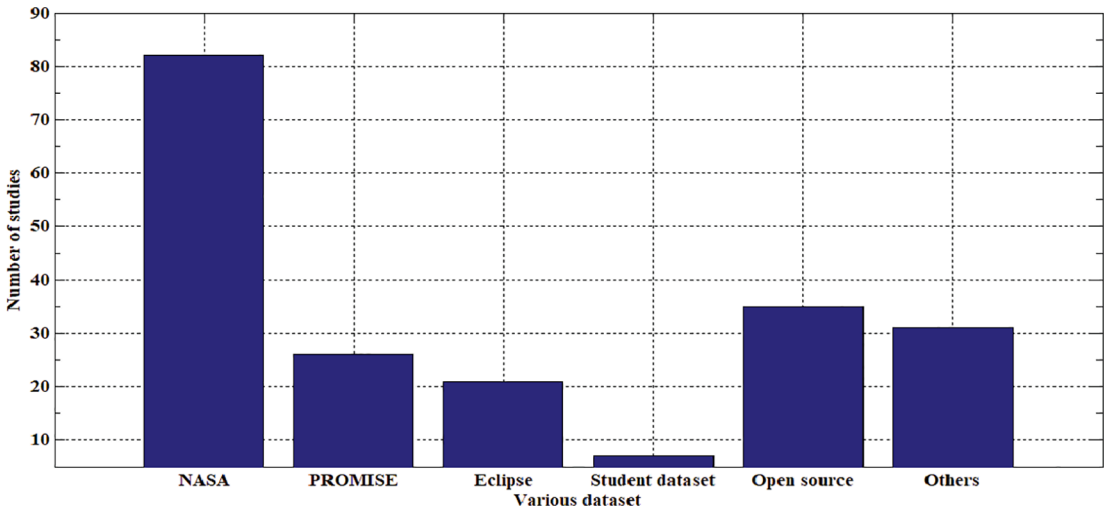


Fig. 8. Percentage of studies for different dataset that are used in SFP.

projects are OO based projects. So it will be more efficient to investigate about OO projects. During the survey (AN-7, AN-8, AN-14, AN-18, AN-21, AN-27, AN-41, AN-52, etc.), it has been found that OO metrics are more likely to be fault-proneness. We have considered that metrics that have been widely employed are more functional in SFP. In Fig. 7 we can see LOC is the most utility metric amongst all OO metrics, followed by CBO (Coupling between objects) and RFC (Response for class). Fig. 7 also reports that NOC and NPM (Network performance management) are lest employed in the SFP domain, and many research articles reported that these metrics are not functional for SFP. Whereas WMC, LCOM, and DIT have a modest contribution, so they are categories as partially utility for SFP. The articles which support this final analysis are AN-8, AN-14, AN-62, AN-84.

4.2.6. *RQ-5: What are the various datasets that are employed in the SFP arena and how to get those datasets?*

A massive variety of datasets have been utilized in SFP. According to our empirical evidence, we found that the NASA dataset is mostly applied in SFP discipline, followed by open source and PROMISE repository datasets, as shown in Fig. 8. We have categories the list of datasets into six types, as reported in Fig. 8. Most of them are freely available, whereas few of them are private datasets. The download link of many old datasets are expired/unavailable, but many researchers have shared those datasets over Github or kaggle or other scientific directories. The categories of various datasets are given below.

- *NASA dataset:* It is the most widely used datasets in SFP. According to our SLR, 82 studies have utilized NASA datasets for their empirical study, as shown in Fig. 8. NASA metrics DATA program repository of NASA has these datasets. It is freeware datasets and can be downloaded from https://github.com/klainfo/NASADefectDataset/tree/master/OriginalData/MDP.
- *PROMISE dataset:* It is also frequently employed datasets in the SFP domain. The datasets are freely available in PROMISE repository (Sayyad Shirabad & Menzies, 2005). In our survey, 25 articles used PROMISE datasets in their study. It can be downloaded from http://promise.site.uottawa.ca/SERepository/.
- *Eclipse dataset:* Most of its versions are freely available at http://bug.inf.usi.ch/download.php. According to our investigation, 21 studies used eclipse datasets.
- *Student dataset:* This is mainly for academic study, which is developed by students, out of 154, 4 studies used student datasets. Few of them can be downloaded from http://bug.inf.usi.ch/index.php.
- *Open-Source dataset:* It includes some other open-source software projects, such as Xylan, Lucene, Ant, Apache, KDE, Gnome, Mozilla, Openoffice, Klac, Kpdf, Kspread, and Firefox, etc. Our SLR suggests that 35 studies utilized open-source data. It can be downloaded from http://bug.inf.usi.ch/download.php.
- *Others:* This is some private dataset, or industrial datasets, such as commercial Java application or commercial banking dataset. Our survey suggests that 31 studies utilize such datasets.

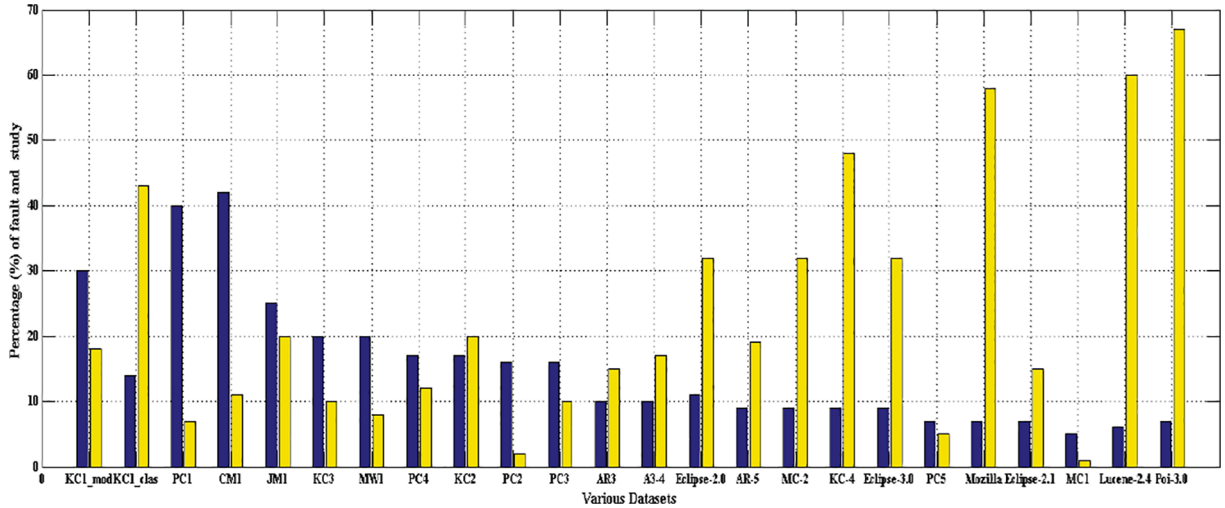


Fig. 9. Fault distribution of different datasets.

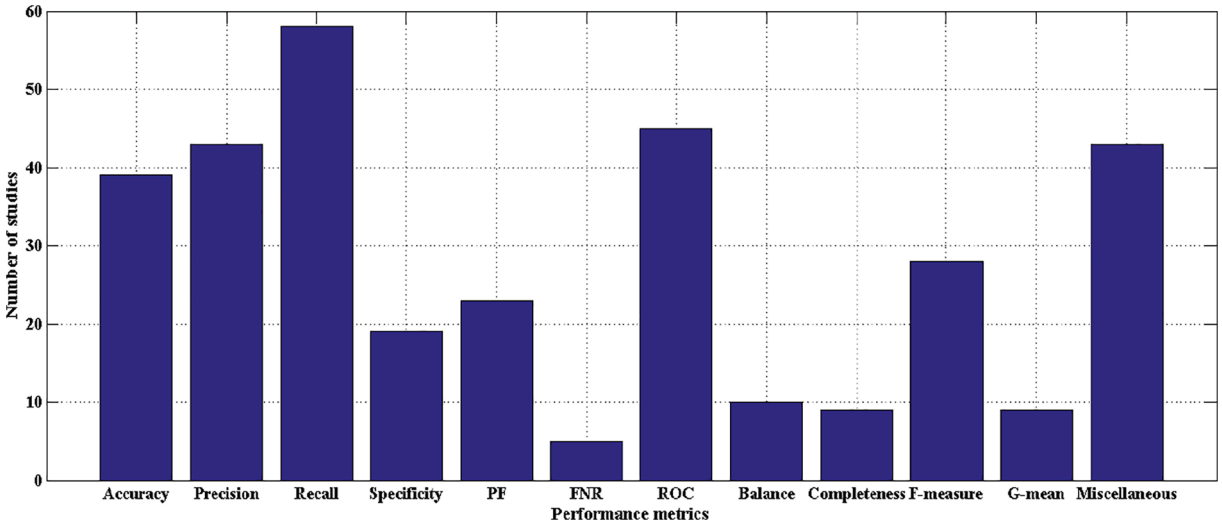


Fig. 10. Performance of metrics that are used in different studies.

Most of the NASA datasets are mainly procedural metrics based projects, whereas the PROMISE projects are OO based software systems. Eclipse software system contains hybrid software metrics projects. The majority of open-source projects are procedural and OO software features-based.

4.2.7. *RQ-5.1: What are fault distribution in dataset for SFP?*

Few software practitioners (Shepperd et al., 2013) challenges over the fault prediction dataset. It has been reported that there is a consistent fault in the different datasets. Fig. 9 shows the percentage of studies that uses a data called d and distribution of faulty module/classes over the same data d. The y-axis represents both % of studies and % of fault distribution in blue and yellow bar graphs, respectively, of the same data d. Fig. 9 reports that KC1 is most extensively used the dataset of a NASA (including class & module), which is 44% of relevant study, and it consists of 51% of the fault distribution. Followed by CM1, & PC1 with 43%, & 40% of relevant studies.

A few patterns can be found in Fig. 9 that the dataset with the lowest fault distribution is popularly used and vice versa. According to the bar graph, the NASA datasets have less fault distribution (CM1, PC1, KC3, MW1, PC4) compared with the PROMISE repository dataset (poi-3.0, lucene-2.4, eclipse-3.0, eclipse-2.1). One additional information this figure reports that in few cases, the datasets which have a large number of instanced have less fault distribution such as JM1. CM1 and JM1 are also substantially used datasets combine with 57% of studies and contain 31% of faulty modules.

4.2.8. *RQ-6: Performance analysis of various evaluation metrics in ML-based SFP models?*

In this section, we will discuss various evaluation metrics and their performance consequences over different SFP models. Fig. 10 shows the various performance measurements that have been used in multiple studies. Recall (58) and ROC (45) have most extensively used in SFP discipline. Miscellaneous consists of several rarely used evaluation measures, as shown in Table 6. The articles that have used these performance measures are listed in Table 6. Miscellaneous, Precision, and accuracy have also been significantly applied in SFP, with 44, 44, and 39 number of studies. F-measure, PF, specificity, balance, G-mean, completeness, and FNR are least utilized, as shown in Fig. 10 and Table 6. Performance measures on the faulty dataset caused imbalanced analysis, so there has been contradicting discussion regarding performance in (Zhang, 2007). They also censured the application of recall and PF for prediction methodology. Whereas Menzies, Greenwald, and Frank (2007) criticized for precision performance measure, and employing AUC in SFP are appreciated by Menzies et al. (2007) and Lessmann et al. (2008).

After examined various classical fault prediction articles, we have pondered the seven most popular models (see Table 7). We have manually enumerated all the accuracy and ROC values reported by these articles. After getting each value, we find their standard deviation (SD), mean, and median. In Table 7, we presented the results of various ML techniques based on SFP models, if the value of performance measure (AUC & Accuracy) is high, it indicates the high rate of accurate prediction. Note: We have mostly considered the AUC and accuracy of those articles that have employed sampling techniques to address class imbalance problem, which avoid biased results. The maximum, minimum, SD, mean, & median values of AUC and accuracy of various SFP models are shown in Table 7. The two maximum AUC produced by RF (EL) and C4.5 (DT) based SFP model, and their corresponding values are 1 and 0.99, respectively. However, the two utmost accuracies are yielded by SVM and C4.5 (DT) based SFP with 97.3%, and 94.87%, respectively. BN and NB are moderately performed, whereas the prediction performance of MLP is the least among all seven SFP models.

4.2.9. *RQ-6.1: Comparison between of various ML techniques and classical statistical techniques over SFP arena?*

The classical LR based SFP technique is the most widely statistical-based model. We have compared the performance of various ML models with the conventional LR method. After an investigation of eight studies over five datasets, it has been found that the AUC of the ML model is 4% higher than the LR model. The AUC calculated from the ROC curve is mostly employed in medical diagnosis and increases in the data-mining form for many years. de Carvalho et al. (2008) suggested that AUC is a relevant parameter when we are dealing with noisy or unbalanced data. In contrast, when there is a change in the distribution of class, the AUC is insensitive. However He and Garcia (2009) recommended the use of AUC in imbalanced data with respect to class distribution. They have also provided a visual representation of relative advantages (true-positive) and cost (false-negative) for classification. In the resent studies by Menzies et al. (2007) and Lessmann et al. (2008), they suggested the existence of imbalanced data and the importance of AUC for building fault prediction model.

Now we are comparing the AUC of various categories of ML models with the LR technique. Fig. 11 shows the percentage of ML models, which have higher AUC than the LR model considering the dataset as the parameter. According to a result, 72% of ML models (AUC consideration) outperformed over the LR model with the dataset as the parameter. Our empirical study reports that RF, MLP, C4.5, BN, NB AN, and Bagging are the most frequently used ML techniques for SFP.

Fig. 11 expresses that Bagging and RF outperform over the LR model across 100% and 90% of overall datasets, respectively. BN, MLP, AB,

Table 6
Analysis of performance measure over studies.

Metric performance	Description	Related studies
Accuracy	The total number of flawless prediction in overall. Similar to Precision	AN-2, AN-3, AN-4, AN-6, AN-8, AN-11, AN-13, AN-14, AN-18, AN-20, AN-21, AN-22, AN-25, AN-27, AN-31, AN-32, AN-35, AN-41, AN-43, AN-50, AN-52, AN-57, AN-61, AN-64, AN-67, AN-87, AN-99, AN-107, AN-108, AN-110, AN-114, AN-115, AN-118, AN-119, AN-124, AN-125, AN-133, AN-36, AN-142
Precision	The proportion of the total number of fault-proneness classes among the overall classified fault-prone class.	AN-4, AN-5, AN-6, AN-7, AN-8, AN-10, AN-13, AN-18, AN-20, AN-21, AN-22, AN-33, AN-34, AN-35, AN-41, AN-42, AN-43, AN-48, AN-53, AN-54, AN-55, AN-57, AN-63, AN-67, AN-86, AN-89, AN-95, AN-105, AN-109, AN-112, AN-114, AN-115, AN-118, AN-122, AN-124, AN-127, AN-131, AN-133, AN-140, AN-142, AN-146, AN-148, AN-151
Recall	Correctly predicted fault-prone class among every fault-prone classes	AN-2, AN-3, AN-5, AN-7, AN-8, AN-9, AN-12, AN-13, AN-14, AN-15, AN-16, AN-18, AN-20, AN-21, AN-22, AN-24, AN-25, AN-26, AN-27, AN-28, AN-30, AN-32, AN-33, AN-34, AN-35, AN-40, AN-41, AN-42, AN-44, AN-47, AN-48, AN-52, AN-53, AN-54, AN-55, AN-57, AN-60, AN-61, AN-63, AN-64, AN-66, AN-67, AN-86, AN-89, AN-95, AN-109, AN-114, AN-115, AN-118, AN-122, AN-124, AN-127, AN-131, AN-133, AN-134, AN-146, AN-148, AN-151
Specificity (True negative rate)	Correctly predicted non-fault-prone classes among every actual non-fault-prone classes	AN-2, AN-3, AN-8, AN-14, AN-27, AN-32, AN-33, AN-44, AN-47, AN-48, AN-52, AN-57, AN-64, AN-114, AN-115, AN-134, AN-140, AN-144, AN-150
False Positive Rate (Pf)	The ratio of every non-fault prone-classes that are falsely classified as fault-prone classes	AN-2, AN-3, AN-9, AN-12, AN-14, AN-15, AN-16, AN-24, AN-25, AN-26, AN-28, AN-29, AN-30, AN-38, AN-40, AN-47, AN-54, AN-55, AN-60, AN-66, AN-98, AN-112, 136
False Negative Rate (FNR)	The ratio of faulty classes that are classified are fault-prone classes	AN-14, AN-29, AN-47, AN-98, AN-112, AN-136
Area Under Curve (AUC)	Receiver operating characteristics curve(ROC) plot 1-specificity value of x-axis and recall value on y-axis, effectiveness is calculated area under ROC	AN-7, AN-13, AN-17, AN-20, AN-21, AN-23, AN-32, AN-37, AN-40, AN-41, AN-42, AN-43, AN-44, AN-47, AN-48, AN-49, AN-52, AN-53, AN-58, AN-60, AN-61, AN-62, AN-67, AN-68, AN-95, AN-98, AN-99, AN-115, AN-128, AN-129, AN-130, AN-132, AN-134, AN-136, AN-139, AN-140, AN-141, AN-142, AN-143, AN-144, AN-145, AN-147, AN-148, AN-149, AN-150
Balance	It is optimal cut-off point of sensitivity, i.e euclidean distance from(0,1) point in the curve	AN-10, AN-12, AN-15, AN-16, AN-32, AN-37, AN-43, AN-44, AN-128, AN-149
Completeness	Summation of every faulty module is classified as faulty module over total no. of actual faulty module	AN-4, AN-6, AN-10, AN-32, AN-37, AN-44, AN-91, AN-97, AN-105,
F-measure (f-score)	Harmonic mean of precision and sensitivity	AN-5, AN-8, AN-13, AN-18, AN-20, AN-21, AN-22, AN-42, AN-47, AN-53, AN-54, AN-56, AN-57, AN-60, AN-63, AN-65, AN-67, AN-89, AN-123, AN-127, AN-129, AN-131, AN-132, AN-133, AN-137, AN-139 AN-146, AN-147, AN-148
G-mean (G-mean1, G-mean2)	It combines true negative rate and true positive rate at 'one' specific threshold	AN-8, AN-13, AN-43, AN-125, AN-128, AN-129, AN-149, AN-154
Miscellaneous measure	It includes H-measure, Precision-Recall curve, Error rate, Gain value, Sum of square error, TIE, TIIE, Absolute Gap, t-value, p-value, type1, type2, variation, and biased	AN-8, AN-17, AN-24, AN-33, AN-35, AN-43, AN-47, AN-59, AN-68, AN-80, AN-81, AN-84, AN-85, AN-87, AN-88, AN-89, AN-100, AN-101, AN-102, AN-104, AN-105, AN-106, AN-107, AN-108, AN-111, AN-113, AN-114, AN-119, AN-121, AN-123, AN-125, AN-126, AN-127, AN-128, AN-129, AN-135, AN-137, AN-138, AN-140, AN-146, AN-149, AN-151, AN-152, AN-153

SVM, C4.5, and NB exceed AUC over 80%, 68%, 65%, 38%, 38%, and 30% of all datasets, respectively. Fig. 11 also reports that EL (RF, Bagging, and AB) based models are more efficient over the LR technique. Although BL based models (BN, and NB) outperforms over more than 40% of all datasets. The articles that compared the performance of ML-based and LR based SFP models are, Bagging techniques are compared in AN-20, AN-53, and AN-62, AB model in AN-53, AN-62 and AN-68, J48 based method in AN-9, AN-20, AN-25, AN-64, and AN-129.

We have generalized our results over 45 articles, as these many articles claimed that ML-based SFP models outperform over many statistical-based SFP models such as LR, and ML-based SFP able to generalize and produce more accurate results.

4.2.10. RQ-6.2: Which ML techniques are better and outperforms over exiting SFP?

To compare the performance between various ML models we embraced some methodologies as we have used in RQ-6.1. The AUC of various SFP models can be compared. Table 7 indicated the AUC of the top seven SFP models. The result expresses that the average AUC for multiple techniques is more than 0.64 which is for all defective classes and datasets. As it shows that maximum AUC of RF (1) and a minimum of NB (0.649) based SFP methods. As the seventh column of Table 7 reports, the average of ROC and accuracy of ML-based SFP techniques. More the average AUC more optimal towards diverse datasets. We can see the C4.5 has maximum average accuracy (85.01), which implies that C4.5 is more optimal over various data repository, whereas RF is least optimal. As few researchers claims, ROC is better performance measure over accuracy for imbalance dataset (Wang & Yao, 2013). RF-based SFP has a maximum mean ROC value, i.e., 0.84, which indicates more suitable for imbalanced datasets. Whereas C4.5 is least suitable for imbalanced datasets. Standard Deviation (SD) gives an indication of how far the individual responses to a question vary or 'deviate'. The fifth column of Table 7 reports the SD of accuracy and ROC of various ML techniques. If the result is diverse, then the model is lesser stable (within this Table) and vice versa. The SD of ROC of MLP is maximum indicates lest stable, whereas SVM and RF are most stable. We are not considering accuracy because most of the datasets are imbalance.

According to the result, the RF model is the most efficient model amongst all ML or LR models; it is also the most widely used ML technique in the SFP arena. Followed by MLP, BN, C4.5, NB, and SVM are also substantially used as prediction models in many studies, NB outperforms over SVM and MLP, whereas BN outperforms over MLP, C4.5, and SVM.

4.2.11. RQ-7: What are the various strength and weakness of ML techniques?

In this section, we have summarized the various strength and weaknesses of several ML techniques, which are investigated through many articles. Table 8 express the different ML techniques with their strength and weakness, whereas the last column is the corresponding supporting articles. As from the reviews, NB reported excellent performance in prediction analysis. Whereas NB is also functional in the

Table 7
Result and analysis of SFP model over various ML techniques.

Techniques	Count	Performance measure	Max	Min	S.D	Mean	Median
C4.5 (DT)	16	Accuracy	94.87	68.75	7.50	85.01	84
	23	AUC	0.99	0.48	0.13	0.78	0.77
SVM	32	Accuracy	97.8	63.34	9.54	80.9	8.55
	20	AUC	0.95	0.49	0.18	0.72	0.72
BN (BL)	9	Accuracy	82.45	73.99	3.12	77.01	77.01
	29	AUC	0.9	0.631	0.09	0.8	0.79
NB (BL)	22	Accuracy	93.41	62	7.77	80.6	80.66
	52	AUC	0.96	0.65	0.09	0.8	0.79
MLP (NN)	15	Accuracy	93.87	64	9.54	82.23	82.40
	43	AUC	0.96	0.55	0.1	0.8	0.78
J48 (DT)	11	Accuracy	93.17	79	4.6	83	83.1
RF (EL)	10	Accuracy	94.12	55.7	16.02	75.7	75.65
	38	AUC	1	0.67	0.09	0.84	0.84

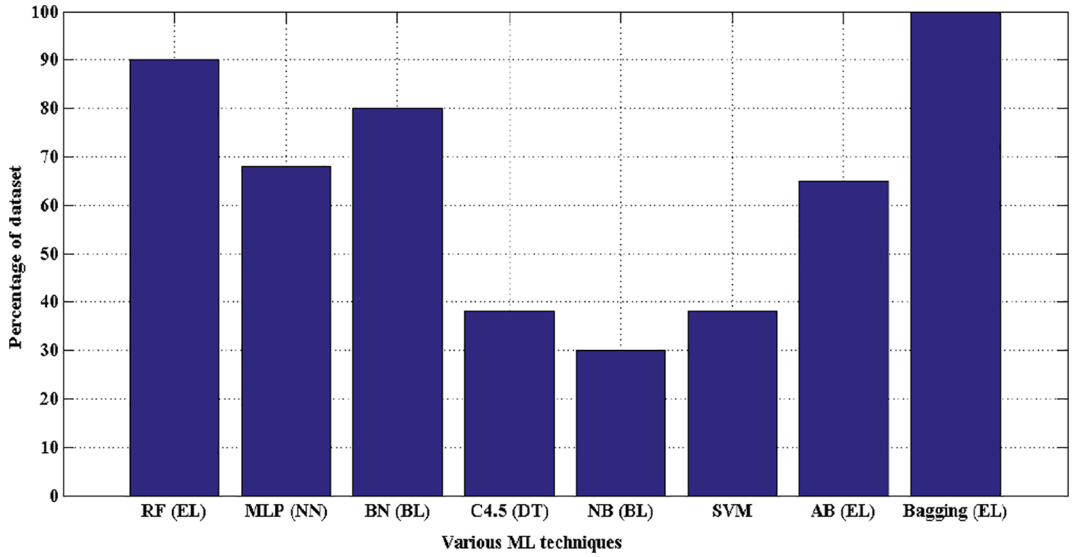


Fig. 11. AUC comparison between various ML models over LR models where dataset used as a parameter.

attribute selection method, it can be helpful in the high dimensional dataset, but not much suitable for co-related features. SVM has been praised for the high capability of irrelevant features and multidimensionality. The C4.5 technique is straightforward to implement and brings an adequate result, so it will be a great way to include in the prediction model. Table 8 reports that it is the most simple predictive model; it can also be considered as baseline methods for all the datasets. More useful for smaller projects, it takes lesser computational cost. As the Table reports, the MLP is most complicated for input/output transformation, so it will become a more complicated model when the hidden layer will increases. It may lead to gives much better prediction results; it takes high computational cost. RF is more suitable for large and imbalanced datasets. RF also avoids overfitting. There are various ML techniques which best fits different domains, they are capable of giving the solution, but that solution can optimal as it varies on types of problem.

4.3. Opportunities trends

Due to rapid growth in the Internet of Things (IoT), software projects are becoming bulkier, which consists of high dimensionality and has more number of modules. Different version of same projects are also arriving more promptly so cross-version fault prediction (CVFP) (Yang & Wen, 2018; Pandey & Tripathi, 2020) and cross-project fault prediction (CPFP) (Xia, Lo, Pan, Nagappan, & Wang, 2016; Herbold, Trautsch, & Grabowski, 2017) will be more useful. SFP prediction unable to generalize the learned features over the new project and yield to poor performance (Hosseini, Turhan, & Mäntylä, 2018); CVFP and CPFP subjugates this problem. As discussed earlier, that class imbalance is a prominent challenge in SFP. If a model avoids the class imbalance problem along with better pattern learning ability will be more efficient over existing techniques. Benchmark datasets are also playing an important role in SFP. Most of the industries must share their developed project's dataset for better modeling. If more diverse datasets are available, then a more accurate prediction model can be made.

4.4. Insightful discussion

This section will briefly illustrate the various conclusions or results obtained from SFP's industrial case studies, which can help the software engineering community and shape future research. First, we will discuss the conclusive evidence of various datasets and software metrics, then illustrate different ML techniques followed by applied performance measures. In last, we will discuss scenario-based prediction approaches. Catal and Diri (2009a) investigated the effect of data size in SFP by employing NASA and PROMISE repository. They found when the size of the dataset increases, the performance of the RF-based model also increases, whereas the NB approach is most favorable in a small dataset; they were using AUC as the performance measure. A similar conclusion is drawn by Dejaeger, Verbraken, and Baesens (2012), they also concluded that simple and comprehensive networks with less number of nodes could be structured using the BN technique other than the NB

Table 8
Strength and weakness of various ML techniques.

Techniques	Strengths	Reference articles
C4.5	Cost-effective model and simple model, easy to implement.	AN-7, AN-11, AN-31, AN-27, AN-40, AN-41, AN-47, AN-74, AN-107, AN-142
RF	Can able to manage large dataset, consistent performance, robust to missing data, parameter setting and noise. Fast to train. Understandable model, efficient on large dataset, better to identifying independent variable	AN-3, AN-21, AN-2, AN-23, AN-32, AN-40, AN-45, AN-52, AN-61, AN-74, AN-107, AN-108 AN-110,
NB	Robust, easy to construct and interpretation. Efficient computational	AN-11, AN-14, AN-20, AN-23, AN-28, AN-33, AN-33, AN-40, AN-51, AN-54, AN-68, AN-53, AN-66, AN-67, AN-74, AN-127, AN-128, AN-129, AN-130, AN-132, AN-151, AN-152, AN-154
SVM	High level of tolerance high dimension space, robust, redundant feature, difficult to manage function. able to mange non-linear problems	AN-7, AN-11, AN-18, AN-19, AN-22, AN-23, AN-32, AN-37, AN-40,AN-44, AN-48, AN-53, AN-61, AN-63, AN-66 AN-69, AN-72, AN-74, AN-99, AN-101, AN-107, AN-119
MLP	Can be complex for non-linear input output transformation	AN-1, AN-5, AN-19, AN-34, AN-36, AN-40, AN-48, AN-89, AN-107, AN-109, AN-120, AN-101,
Techniques	**Weakness**	**Reference article**
NB	Not considered feature co-relation, unable to refuse redundant attributes, performance depend on attribute.	AN-28, AN-20, AN-54, AN-68, AN-54, AN-68, AN-73, AN-40, AN-74, AN-130, AN-132

method. They also conclude that aspects of estimating performance and comprehensibility should be balanced out, and the development context should be considered during model selection. Sunil, Kumar, and Neti (2018) were unable to confirm the class size in the dataset moderation effect on the association between OO metrics and the faults; they recommended the researchers to contemplate the potential mediation and moderation of class size while constructing a prediction model. Liu, Khoshgoftaar, and Seliya (1994) conducted two case study over seven projects. They found the cumulative cost of misclassification over the search-based technique is lower than the non-search-based model. Turhan, Menzies, Bener, and Di Stefano (2009) demonstrated the minimum number of datasets required for an effective fault prediction method. They recommended the two-phase approach, which allows the employing prediction mechanism instantly.

Catal and Diri (2009a) concluded that method-level metrics are more suitable for miscellaneous based approach such as AIRS2Parallel. Karim et al. (2017) found that class level is more effective over compared to the method-level metric they employed SVM, RF, and ANN prediction mechanism. Zhou and Leung (2006) found that design matrices are more statistically associated with fault-proneness across different fault severity. They also concluded that the prediction ability of the examined metrics mostly depends on the severity of faults; moreover, the design metrics are capable of predicting the fault-prone module in low severity faults. Zhou, Xu, and Leung (2010) found that metrics such as average method complexity and standard deviation method complexity are more effective over McCabe a LOC while employing logistic regression. They also conclude that the odds ratio linked with one standard deviation rise, alternately one unit rise. The software metrics should be used to compare the relative effect over fault-proneness or misleading results can be achieved. Radjenović et al. (2013) found in his investigation that OO metrics and process metrics have been more successful in finding the faults as compared with the traditional size and complexity metrics. Moser, Pedrycz, and Succi (2008) conducted experiments using two different sets of software metrics, one set of product metrics, and the second set of process-related metrics of java based Eclipse project. Their results conclude that process metrics are more efficient than process code metrics for fault prediction. Aggarwal, Singh, Kaur, and Malhotra (2009) proposed a predictive model using java based project and found that the coupling and size metrics and more related to fault-proneness. However, their results also conclude that the inheritance metric, which counts the methods inherited in a class, is also useful in the prediction mechanism.

Weyuker, Ostrand, and Bell (2010) compared the effectiveness of NBR, recursive partitioning, RF, and Bayesian additive regression trees methods. They found RF and NBR outperforms over recursive partitioning and Bayesian additive regression trees, as assessed by either of the metrics. Kumar et al. (2017) conducted experiments over 30 open source java project and found SVM is more suitable for faulty classes less than a specific threshold, depending upon fault identification efficiency (low- 52.139%, median- 46.206%, and high- 32.080%). Malhotra and Khanna (2018) proposed strategies of ensemble learning-based SFP model, their results indicate EL based approach outperforms over each individual base classifiers. They also found voting technique is more effective compared with other combination rules. They concluded the accuracy and diversity of every classification technique were instrumental in the outperforming of voting based EL method. Huda et al. (2018) conducted experiments over PROMISE datasets and concluded that the oversampling based EL model reduces the false positive rate and more suitable for class imbalanced datasets. Yu (2012) suggested an evolutionary process asymmetric weighted SVM learning-based SFP framework. They found promising accuracy over software repository mining compared to other classical listed methods. Bishnu and Bhattacherjee (2012) applied a clustering-based approach over SFP; they proposed an approach Quad Tree-based initialization algorithm and compared to other initialization techniques. They found clusters obtained by the Quad Tree-based algorithm have the highest gain values. They also conclude the error rates of the prediction model is lower in other existing prediction models.

Li et al. (2019) suggested that the predictive performance of any SFP method is positively influenced by performance measure and testing procedure. The most regular performance metric for SFP is ROC curve. It is not a set of metrics; it's just a one performance measure (Catal, 2012). They also suggested to employed R2, and AAE/ARE along with ROC to ensure the number of faults prediction. Saito and Rehmsmeier (2015) suggested that the precision-recall curve is more explanatory than the ROC curve when binary classification is employed over the imbalanced dataset. Bezerra, Oliveira, and Meira (2007) proposed SFP framework using RBF and neural network. They conclude that error rate and probability detection is adequate for NN based model. They also found the overfitting problem by a plot of train-test accuracy. For predicting the number of faults, it is better to choose regression-based performance measures such as mean squared error, mean absolute error, etc. After analyzing all the studies, it is better to choose an adequate number of performance measures because class imbalance and overfitting problems are the main two challenges. The predictive model conquers these challenges should be detectable.

Xing, Guo, and Lyu (2005) predicted software quality using SVM; they employed Halsted, McCabe metrics, and 11 other method-level

metrics. They conclude that SVM does not perform well on such public projects. But SVM outperforms over classification tree method and quadratic discriminant analysis. Koru and Liu (2005) examined the different module size in fault prediction by employing Kstar and J48 techniques. They suggested method-level metrics is not preferable over small components instead of large components, whereas class-level metrics are suitable for the large component. Gyimothy et al. (2005) reported that in the multivariate model, object class metrics are more useful than LOC, and DIT, whereas NOC should be used in such models. Li and Reformat (2007) suggested fuzzy labels for classification model over JM1 dataset (method-level metrics). They concluded that SimBoost did not work over such software metrics, whereas fuzzy labels were optimally performed over this dataset.

Catal and Diri (2009a) reported that the RF-based approach is more suitable for a large dataset, whereas the NB technique is best for a small project in terms of ROC. However, the NB technique is not detrimental when PCA is applied in preprocessing. Weyuker et al. (2010) suggested a random forest and negative binomial regression model, and they found both of them performed better over the Bayesian additive tree method as assessed by any of the performance evaluators. Ma, Luo, Zeng, and Chen (2012) concluded that when there is too low training data to train the classifier, than helpful knowledge of existing (different distribution) on same feature level can be useful is such situations. Due to the lack of training data, and overfitting problem may occur. The transfer learning method may be significantly productive. Laradji, Alshayeb, and Ghouti (2015) concluded that forward selection indicates that only limited metrics contribute to high ROC. They found a greedy forward selection technique outperformed other feature selection methods such as Pearson's correlation over-tested data. Tong et al. (2018, 2020) concluded that deep representation is promising for SFP compared with classical software features. They also suggested the heterogeneous EL model is more effective compare with the traditional EL model.

5. Threats to validity

Our SLR is not considered other domains of SFP, such as within-project, cross-project, inter-project, heterogeneous prediction, just in time prediction and status work prediction mechanism; so our SLR is incomplete because we have not included these aspects.

5.1. Biasness over publication

It is a tendency to report more positive results over negative results in every publication. If any articles report negative results have less likely to publish (Kitchenham & Charters, 2007) and vice versa. We did not limit our analysis before the search and selection method. We searched for all kinds of articles based on the selection procedure. Besides, we have also selected most of the conference proceeding articles and uses the snowballing sampling method. Authors of the selected articles can be biased over selecting specific data or methods; this might be because of more number of studies that endorse specific metrics or learning models such as OO metric and ensemble learning. The selected articles for reviews were mainly from different time periods when several organizations embraced or employed such methods. The broad acceptance of such a scenario leads to influenced software practitioners to use such techniques to a greater extent.

5.2. Searching and selecting of primary studies

Our article search strategy was designed to discover as many articles as possible. We constructed a very wide search string that over eight databases, as mentioned in Section 3.2. Although the search results include many irrelevant articles, then we decided to keep our original search string, so that we cannot miss any relevant articles. Additionally, we have used a snowballing sampling method to get more related studies. We found a total of 154 relevant studies. We have also go through the articles of the related works, the studies selected by recent related work are more relevant for our primary studies.

We have used inclusion and exclusion criteria to select relevant articles. If any study belongs to any requirements of exclusion, it was discarded. Three researchers examined the principle of inclusion and exclusion; therefore, the possibility that the exclusion was biased and that a study may have been incorrectly excluded.

We did not include several conference and journal articles apart from suggested venues because experience reports are mostly published in conference proceedings. Therefore, we could consist of a source of information about the industry's experience. Catal and Diri (2009b) and Radjenović et al. (2013) did not use many conference proceedings in their review process because it substantially increases the workload. They mainly include articles of journals and few conferences of ostensible venues.

We found that searching for primary studies in the various databases is neither practical nor efficient; it can be because of inconsistent terminology; a similar conclusion is also made by (Malhotra, Kaur, & Singh, 2010; Staples & Niazi, 2007; Brereton, Kitchenham, Budgen, Turner, & Khalil, 2007).

5.3. Quality assessment

According to Kitchenham and Charters (2007), the quality assessment performed by one researcher and validated by the second researcher. But in our case, the second researcher conducted a data extraction procedure and quality assessment on twenty randomly selected articles. The most bias was expected in the quality assessment, since questions may be hard to answer to any objective. The score of the same article is a varied form of different researchers. According to our belief, the quality specification might not be most precise from the quality assessment, but it is adequate to differentiate between the weaker and better studies.

6. Conclusion and future work

In the SLR process, we go through a systematic literature review to investigate the performance of various ML techniques over several SFP models. We firstly develop a review protocol by previous articles (1993–2019). Secondly, we framed seven research queries after analyzed the different ML-based SFP methods. Table A.14 to Table A.16 addresses the research answer of all seven research queries (see Table 1). Table A.14, Tables A.15, and A.16 also consist the evaluated score which obtained from various standard assessment queries of Table 3. When the score is higher (more than 7), the article is more relevant to SFP over ML. Table A.9 express the various articles with their corresponding score and categories.

This survey will help the researchers or software practitioners for better analysis over prediction modeling. They will select more optimal learning techniques over different conditions; they can map the correlation between the type of projects, learning algorithm, and required evaluation metrics over such scenarios. It will also help them to address various potential threats and challenges of SFP. We also provided a few guidelines for predictive modeling that will also help cross-version fault prediction or cross-project fault prediction mechanism.

We have also summarized the datasets, reduction/selection techniques, ML techniques, software features, performance measures, and investigated them. We have compared various classifiers over different prediction models. In most of the cases, we observed that the performance of ML-based models outperforms over classical statistical methods such as LR. At last, we have summarized the strength and weaknesses of different SFP models. The conclusions drawn from our survey are given below.

- The top seven categories of ML that have broadly employed SFP are Bayesian learner, Decision Tree, Evolutionary Algorithm, Ensemble

Learners, Neural Networks, Support Vector Machine, Rule-Based Learning, Miscellaneous. However, the learning techniques from these categories that are predominately used for SFP are Random Forrest, SVM, Naive Bayes, C4.5, LR, and MLP.

- The most common feature selection methods used for metrics is Co-relation followed by Infogain. PCA is mostly applied as feature extraction. SMOTE and random sampling were broadly employed as sampling methods. The most popular metrics which were used for SFP in studies are Halsted, McCabe, and source code metrics. The NASA dataset used as the most common dataset for the experiment. Accuracy, F-Measure, AUC, Recall, Precision used as the most common performance measure. F-measure3, G-mean, and Precision-Recall curve are considered as unbiased performance measure.
- We observed that the average of AUC lies between 0.72 to 0.84, and the mean value of accuracy lies in the range between 75.7% to 85.01% of ML-based SFP models. In general, ML models outperformed in SFP over the LR model.
- NB and BN techniques were used as frequent ML model for SFP with an efficient result. BL is more robust, less computation, not useful when features are co-related.
- EL based SFP models broadly applied and outperformed over most of the state of the art SFP methods. It also avoids the over-fitting problem. More suitable for imbalanced data, high computational cost, adequate to handle large data size, and more stable.
- DT based models are more robust and simple, produce high accuracy at low cost. They are more diverse toward various datasets. More suitable for small projects, unstable for large projects.
- Deep learning models used as a feature selection. DL is preferable over large instance (at least more than 10 k) data, and when the feature set and/or class label are not presents. Due to lack of massive data, it is inefficient to use end to end DL.
- The latest few articles focus on the issue of class imbalance problem in the public dataset; it inconsistent and biased results when ML techniques applied. Sampling methods were used to avoid this problem.

The following are the future guidelines for software practitioners and researchers who are working in SFP discipline.

(i) There should be the more generalized result of various ML-based SDP techniques; there are only a few studies which have analyzed in a generalized manner. It will be easy to compare the performance and effectiveness of SFP when someone follows this prospective.
(ii) There are very few studies which investigated the predictive ability of LB, AB, RBL, Bagging, ADT, and RBF. Some of the ML techniques have never been implemented for fault prediction. There are decidedly fewer articles which investigate the efficiency of evolutionary algorithm example ACO and GP for SFP.
(iii) The dataset should be freely available by industries so that more research experiments can be carried out for SFP. The number of features and test cases need to be added so that DL can easily apply without over-fitting.
(vi) Selection of classifier is an essential phase of an experiment, the researcher should carefully analyze the problem, and according to that select the corresponding ML technique, because different ML algorithm gives the different result.
(v) Class imbalance and over-fitting becomes two critical issues in SFP arena. As most of the dataset were skewed distribution; this needs to be addressed by redistribution of data instances.
(vi) To compare the potential of ML techniques over SFP, its performance should be compared to other ML/Statistical methodologies.
(vii) The experimental parameters/hyperparameters of ML or DL based models should have precisely reveled so that a similar experiment can be conducted.
(viii) Deep learning-based architecture can also be applied over SFP and can give the prime prediction outcome when the dataset is huge. Practitioner should precisely be taking care of the optimization method and hyperparameters tuning.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This paper supported by the Indian Institute of Technology-BHU (IIT-BHU), Varanasi, India. The authors would like to thank the Department of CSE of IIT-BHU for providing the research platform for scholars. The author would also like to thank their research fellows, faculty members, and other staff members of the CSE dept. of IIT-BHU.

Appendix A

We have listed all supporting articles from AN-1 to AN-154 in Table A.10A.11,A.12,A.13, with descriptions of year, Journal, methodologies, datasets, software metrics, and performance measures. We also addressed the research queries of Table 1 and verified the assessment of each article based on Table 3, we have evaluated the score of each article accordingly as shown in Tables A.14,A.15,A.16. Table A.9 reports the

Table A.9
Categories and ranking of studies over ML techniques that are used for SFP.

Category	Research articles	Score
Excellent	AN-7, AN-27, AN-32, AN-53, AN-61, AN-68	13
	AN-12, AN-23, AN-34, AN-52, AN-67, AN-133, AN-134, AN-136, AN-139, AN-147	12.5
	AN-9, AN-29, AN-40, AN-41, AN-48, AN-49, AN-63, AN-64, AN-69, AN-144, AN-145, AN-150	12
	AN-25, AN-38, AN-43, AN-113, AN-125	11.5
	AN-12, AN-14, AN-16, AN-17, AN-19, AN-42, AN-44, AN-46, AN-47, AN-50, AN-57, AN-58, AN-59, AN-70, AN-72, AN-89, AN-99, AN-100, AN-101, AN-102, AN-110, AN-132, AN-135	11
	AN-3, AN-28, AN-71, AN-128, AN-129, AN-131, AN-138, AN-154	10.5
Good	AN-1, AN-2, AN-4, AN-5, AN-6, AN-10, AN-13, AN-18, AN-22, AN-33, AN-35, AN-39, AN-51, AN-55, AN-66, AN-141, AN-151	10
	AN-21, AN-30, AN-31, AN-73, AN-74, AN-124, AN-140, AN-142, AN-149	9.5
	AN-8, AN-11, AN-81, AN-83, AN-84, AN-88, AN-90, AN-95, AN-96, AN-98, AN-103, AN-107, AN-108, AN-109, AN-111, AN-112, AN-116, AN-130	9
	AN-15, AN-24, AN-26, AN-37, AN-45, AN-54, AN-56, AN-60, AN-62, AN-80, AN-143, AN-152	8.5
Average	AN-76	8
	AN-82, AN-85, AN-87, AN-91, AN-97, AN-105, AN-114, AN-115, AN-119, AN-120, AN-123, AN-126, AN-127, AN-137, AN-153	7.5
	AN-92, AN-94, AN-75, AN-146	6.5
	AN-77, AN-78, AN-79, AN-86, AN-104, AN-106, AN-118, AN-121	5.5
Below average	AN-36, AN-93, AN-117, AN-122, AN-148	(0 to 5)

Table A.10
Analysis of study.

Article no.	Article and journal name	Methodology	Dataset	Software Metrics	Performance metrics
AN-1	Sherer (1995) (Sherer, 1995) Journal of system software, Elsevier Science	Neural network	National Aeronautics and Space Administration software development projects	Lines of code	Simulator and Attitude
AN-2	Guo(2003) (Guo et al., 2003), ASE-03, IEEE	Dempster-Shafer (D-S) Belief networks	NASA dataset	McCabe, Halstead, Line Counts and Branch Counts, etc	Accuracy, Specificity and Sensitivity
AN-3	Guo(2004) (Guo et al., 2004), ISSRE'04-2004	Random forest	NASA datasets (CM1,JM1, KC1, KC2, PC1)	McCabe, Derived Halstead, Line count, Basic Halstead and Branch	Specificity and Sensitivity.
AN-4	Gyimothy(2005) (Gyimothy et al., 2005), IEEE Transactions on software engineering	Decision Tree, Neural Network and Logistic Regression	Open source (Mozilla)	Object-oriented metrics by Chidamber/Kemerer	Correctness, Precision and Completeness
AN-5	Koru(2005) (Koru & Liu, 2005), IEEE software	Neural Network and J48	NASA datasets (CM1, JM1, KC1, KC2, and PC1)	Lines of Code	F-measure
AN-6	Zhou(2006) (Zhou & Leung, 2006), IEEE Transactions on Software Engineering	Logistic Regression, Naive Bayes, NNge (nearest neighbor with generalization), and Random Forest	NASA datasets	Object-Oriented design metrics and, Chidamber and Kemerer(CK) metrics	Coefficient (estimated regression), p-value (Statistical hypothesis), R^2 and Stand-error
AN-7	Arisholam(2007) (Arisholm et al., 2007), IEEE International Symposium on Software Reliability Engineering	C4.5, PART, SVM, Logistic Regression, Neural Network, PART + C4.5, Boost, C4.5, CFSC4.5 and DecorateC4.5	JAVA based projects	NSUP, MAXCC, RFC, MI, INST, LOC, LOCOM, LCOM, and FOUT	Precision, Recall and ROC
AN-8	Catal(2007) (Catal et al., 2007), DepCoS-RELCOMEX'07- IEEE	J48	PROMISE repository	Chidamber-Kemerer (CK) metrics	Accuracy, G-mean1, G-mean2, Precision, Recall and F-measure
AN-9	Jiang(2007) (Jiang et al., 2007), IEEE International Symposium on Software Reliability Engineering	OneR, Voted perceptrol, Logistic Regression, J48, IBK, VFI and Random forest	NASA Metrics Data Program (MDP) data repository	Requirements metrics	Precision, Recall and ROC
AN-10	Kanmani(2007) (Kanmani et al., 2007), Information and software technology, Elsevier	Probabilistic Neural Network, Back propagation Neural Network	Institute level dataset	Object oriented metrics	Correctness, Completeness, Effectiveness and Efficiency.
AN-11	Li(2007) (Li & Reformat, 2007), IEEE International Conference on Information Reuse and Integration	MLP, Naive Bayes, SVM, C4.5 and SIMBOOST	NASA's Metrics Data Program	McCabe, Halstead, and Line of Code	Accuracy (MDP)
AN-12	Menzies(2007) (Menzies et al., 2007), IEEE Transactions on Software Engineering	J48, ROCKY, LSR, and M5'	NASA dataset(CM1)	McCabe, Halstead, Basic Halstead, Line of Count and Branch Count	Precision, Recall, Accuracy and Selectivity
AN-13	Ma(2007) (Ma et al., 2006), IGI global	Decision tree using Random Forest	NASA defect datasets (KC1, KC2, JM1, PC1 and CM1)	McCabe, Derived Halstead, Line Count, Basic Halstead, and Branch	Accuracy, Recall, True Negative Rate (TNR), G-mean1, G-mean2 and F-measure
AN-14	Pai(2007) (Pai & Dugan, 2007), IEEE Transactions on Software Engineering	Bayesian Network and Variation of Bayesian classifier combined with linear, Poisson, and Binomial, Logistic Regression	NASA datasets	Object-Oriented software metrics	Specificity, Sensitivity, False Positive Rate and False Negative Rate.
AN-15	Turhan(2007) (Turhan & Bener, 2007a), In Quality Software, 2007. QSIC'07	Multi-Variant approach in Bayesian Network	NASA dataset (CM1, KC1, KC2, KC3, KC4, PC1, and MW1)	Static Code metrics	Probabilistic Prediction Rate
AN-16	Turhan(2007a) (Turhan & Bener, 2007b), ICSOFT (SE)	Weighted Naive Bayes and PCA based Naive Bayes	NASA dataset (CM1, PC1, PC2, PC3, PC4, KC3, KC4 and MW1)	McCabe, Halstead, Basic Halstead, Line of Count and Branch Count	Probability of faulty and non-faulty using confusion matrix
AN-17	Calvalho(2008) (de Carvalho et al., 2008), ICTAI'08-IEEE	Multi-Objective Particle Swarm Optimization	NASA(KC1)	Chitamber/Kemerer metric	AUC of ROC
AN-18	Elish(2008) (Elish & Elish, 2008), Journal of Systems and Software, Elsevier	Support Vector Machines	NASA dataset(KC1, KC3, PC1 and CM1)	McCabe, Halstead, Basic halstead, Line of Count and Branch	Accuracy, Precision, Recall and F-measure
AN-19	Gondra (2008) (Gondra, 2008) Journal of Systems and Software, Elsevier	Artificial Neural Network and SVM	NASA(IV,V)Facility Metrics Data Program (MDP) and JM1	Halstead, Basic halstead, Line of Count and Branch	Mean Squared Error

(*continued on next page*)

Table A.10 (*continued*)

Article no.	Article and journal name	Methodology	Dataset	Software Metrics	Performance metrics
AN-20	Jiang(2008) (Jiang et al., 2008), Empirical Software Engineering, Springer	Naive Bayes, Logistic Regression, IB1, J48 and Bagging	NASA dataset (KC1, KC2, KC4, CM1, PC1, PC5, JM1 and MC2)	Halstead, Basic Halstead, McCabe, Line of Count and Branch	Accuracy, Error rate, Precision, Recall, F-measure, G-mean1, G-mean2, and ROC
AN-21	Kaur(2008) (Kaur & Malhotra, 2008), ICACTE.2008- IEEE	Random Forest	PROMISE repository	CBO, LCOM, NOC, DIT and WMC	Precision, Recall, F-measure, and ROC
AN-22	Kim(2008) (Kim et al., 2008), IEEE Transactions on Software Engineering	SVM, and Evolutionary methods for change buggy prediction	Web based dataset(Bugzilla, Colamba etc)	Software quality metrics or a project's change history metrics, complexity	Precision, Recall, Accuracy and F-measure
AN-23	Lessmann(2008) (Lessmann et al., 2008), IEEE Transactions on Software Engineering	Statistical based, nearest neighbor methods, SVM based classifier, Tree based and Ensemble Learning	NASA datasets (CM1, KC1, KC4, MW1, JM1, PC1, PC2, PC3, and PC4)	LOC, Counts, Halstead, McCabe, and Miscellaneous	Rank measurement using based on AUC of ROC
AN-24	Menzies(2008) (Menzies et al., 2008), Proceedings of the 4th International workshop on Predictor models in software engineering-ACM	Variation of Naive Bayes and J48	NASA dataset (PC5, MC1, PC2, KC1, MC3, PC3, PC4, MC2, MW1, KC3, CM1, KC2, and PC1)	LOC counts, Halstead, McCabe, and Miscellaneous	Precision, Recall and Balance
AN-25	Moser(2008) (Moser et al., 2008), ICSE, ACM	Logistic regression, Naive ayes, and Decision Tree	Java based Eclipse projects	Code and change metrics	Percentage of correctly classified instances (PC), Recall, and False Positive Rate
AN-26	Turhan(2008) (Turhan et al., 2008), SEAA'08, Euromicro Conference-IEEE	Static Call Graph Based Ranking	Embedded projects (AR3, AR4, and AR5)	Lines of Code, McCabe and Halstead	True Positive Rate, False Negative Rate and Balance
AN-27	Vendecruys(2008) (Vandecruys et al., 2008), Journal of Systems and Software, Elsevier	Ant Colony Optimization and Oversampling on dataset based AntMiner+	NASA dataset (PC1, KC1, and PC4)	Halstead and Complexity measures	Specificity, Sensitivity and Accuracy
AN-28	Bener(2009) (Turhan & Bener, 2009), Knowledge Engineering, Elsevier	Naibe Bayes with subset selection and PCA	NASA (CM1, PC1, PC2, PC3, PC4, KC3, KC4 and MW1)	Basic Halstead, Object Oriented and McCabe	Balance, Probability of Detection and Probability of False Alarm
AN-29	Catal(2009) (Catal et al., 2009) Sixth International Conference on Information Technology: New Generations-IEEE	Clustering (fuzzy and hard) and Metrics thresholds based methods	Public dataset (AR3, AR4, and AR5)	LOC, CC, UOp, UOpand, TOp, and TOpand	False Positive Rate, False Negative Rate and error.
AN-30	Menzies(2009) (Turhan et al., 2009), Empirical Software Engineering, Springer	Analogy-based learning i.e nearest neighbor filtering	Public and cross company dataset (PC1, KC1, KC2, CM1, KC3, MW1, AR4, AR3, AR5, and MC2)	Static code features(40 features)	Probability of Detection and Probability of False Alarm
AN-31	Singh(2009) (Singh & Verma, 2009), International Conference on Advances in Computing, Control, and Telecommunication Technologies-IEEE	Naive Bayes and J48 Integrated with discretization algorithm	PROMISE repository	McCabe, Halstead and LOC measures	Mean Absolute Error and Root Mean Squared Error
AN-32	Singh(2010) (Singh et al., 2010), Software quality journal, Springer	Logistic Regression, ANN and Decision Tree	NASA dataset (KC1)	Object Oriented metrics (CBO, RFC, LCOM, NOC, DIT, WMC, and SLOG)	ROC, Sensitivity, Specificity, Cutoff and Completeness.
AN-33	Okutan(2014) (Okutan & Yildiz, 2014), Empirical Software Engineering, Springer	Bayesian Network	Poi, Tomcat, and Xalan	NOD, LOCQ, LOC, RFC, CBO, LCOM, WMC, NOC, NOD and DIT	Probability of Defectiveness/Non-Defectiveness
AN-34	Erturk(2016) (Erturk & Sezer, 2016), Applied SoftComputing, Elsevier	ANN, Fuzzy Inference System and Adaptive Neuro fuzzy inference system	PROMISE repository (Ant, jEdit, Camel, Xalan, Log4j and Lucene projects)	Object Oriented metrics	ROC
AN-35	Rathore(2017) (Rathore & Kumar, 2017), Expert systems with Applications, Elsevier	Predict number of faults using heterogeneous ensemble learning, Friedman's test	Public dataset (Xerces, Came, Ant, jEdit, Xalam, Pro, and Camel)	Eclipse, Complexity, and Change metrics	Average Absolute Error, and Average Relative Error
AN-36	Tantitham(2016) (Tantithamthavorn et al., 2016), IEEE Transactions on Software Engineering	Comments on Research Bias	NIL	NIL	NIL
AN-37	Singh(2009) (Singh et al., 2009), WCE 2009,	Support Vector Machine	NASA subset (KC1)		ROC, Sensitivity, Specificity, Completeness and Cutoff

(*continued on next page*)

Table A.10 (*continued*)

Article no.	Article and journal name	Methodology	Dataset	Software Metrics	Performance metrics
				Chidamber and Kemerer (CBO, LCOM, NOC, DIT, WMC, RFC and SLOC)	
AN-38	Tosun(2009) (Tosun et al., 2009), International Conference on Predictor Models in Software Engineering, ACM	Naive Bayes, Linear Regression and Logistic Regression	Public dataset (AR3, AR4 and AR5)	McCabe, Halstead, LOC and Complexity metrics	Precision, Recall, Probability Defect and Balance
AN-39	Zimmermann(2009) (Zimmermann et al., 2009), SIGSOFT- ACM	Decision Tree	Open Source Software, Internet Information Services, Windows file system, Apache, Tomcat and SQL server based projects	Relative metrics were used of Basic Halstead metrics	Precision, Recall and Accuracy
AN-40	Afzal (2010) (Afzal, 2010), APSEC-ACM	ANN, SVM, LR, C4.5, RF, GP and NB	Institute level dataset	SF-CR, SF-MDR, SF-R, SF-HD, SF-MT, SF-U, SF-H, ST-I and ST-S	Precision, Recall, Probability of Defect and ROC
AN-41	Arisholm(2010) (Arisholm et al., 2010), Journal of Systems and Software, Elsevier	C4.5 (variation), PART, Logistic Regression, Neural Network	NASA(cos20, cos21, cos22, cos23)	Object Oriented, Halstead, and Derived Halstead	Recall, Precision, Accuracy, ROC and Cost effective
AN-42	Carvalho(2010) (De Carvalho et al., 2010), Journal of Systems and Software, Elsevier	Multi-objective Particle Swarm Optimization	NASA (PC1, KC1, JM1, CM1, and KC2)	Object Oriented (CBO, DIT, LCOM, NOC, RFC, and WMC)	Sensitivity, Specificity, Support, Confidence and ROC
AN-43	Malhotra(2018) (Malhotra & Khanna, 2018), Information and Software Technology, Elsevier	Particle Swarm Optimization and Ensemble Learning	Mobile application dataset (android calendar/contact/gallery/bluetooth)	Object Oriented metrics (Size, Coupling dependency, Cohesiveness)	Accuracy, Precision, G-mean, f-score, Balance
AN-44	Mahlotra(2010) (Malhotra et al., 2010), International Journal of System Assurance Engineering and Management, Springer	Variation of SVM	NASA dataset (KC1)	Object Oriented (CBO, LCOM, NOC, RFC, WMC, LOC, and DIT)	Specificity, Sensitivity, Completeness, ROC
AN-45	Ostrand(2010) (Weyuker et al., 2010), Empirical Software Engineering, Springer	Negative Binomial Regression, Random Forest, Bayesian Additive Hidden Tree and Recursive Partitioning	Industrial dataset	No. of faults using LOC	Average Fault Ratio, co-relation % between system
AN-46	Pendharkar (2010) (Pendharkar, 2010), Engineering Applications of Artificial Intelligence, Elsevier	Probabilistic Neural Network and Simulated Annealing	NASA(JM1)	Halsted, McCabe, no. of graphs/operator/operand	Probability % of Defective and Variable available frequency
AN-47	Seliya(2010) (Seliya et al., 2010), HASE-2010, IEEE	Balanced Bagging (RBBag) combined with Naive Bayes and C4.5	NASA (JM1, KC1, KC2, KC3, MW1M PC1, SP1, SP2, SP3, CCCS-2, CCCS-4, CCCS-8,and CCCS-12)	Numbers of metrics made available, Process and Object Oriented metrics	TPR, FPR, TNR, TPR, F-measure, G-mean, and AUC of ROC/PRC
AN-48	Singh(2010) (Singh et al., 2010), IJCA	Logistic Regression, ANN, DT, SVM, Cascade Correlation Network, Gene Expression Programming, Group method of Data handling Polynomial method	Public dataset (AR1)	Lines of Code and Miscellaneous	ROC
AN-49	Zhou(2010) (Zhou et al., 2010), Journal of Systems and Software, Elsevier	Logistic Regression	Eclipse2.0,2.1,3.0, Rhino1.4R1, 1.5R1, 1.5R2, 1.5R3	Miscellaneous (SDMC, WMC, AMC, CCmax, NIM, NCM, NTM, NLM, AvgLOC and LOC)	ROC, Somers' D and Goodman-Kruskal's γ
AN-50	Azar(2011) (Azar andVybihal, 2011), Information and Software Technology, Elsevier	Ant Colony Optimization	Web based projects (Bean browser, Javamapper, Jchempaint, Jigsaw 4 846-958 SourceForge Jlex, Jms, Voji)	Coupling, Cohesion, Inheritance, and Size metrics	Average, Minimum, Maximum and Standard Deviation of testing Accuracy

Table A.11
Analysis of study.

Article no.	Article and journal name	Methodology	Dataset	Software Metrics	Performance metrics
AN-51	Diri(2011) (Catal et al., 2011), Expert Systems with Applications, Elsevier	Naive Bayes based for Java program	Eclipse-based Software systems	Derived Halstead, Primetive Halstead, McCabe, and CK metrics	Probability of Detection, Gamma, Log, Normal and Poisson Distributions
AN-52	Malhotra(2011) (Malhotra & Singh, 2011), Software Engineering: An International Journal	Logistic Regression	Open source Software Systems	Object Oriented metrics(CBO, LCOH, NOC, DIT, WMC, RFC, NMP, Ca, LCOM1, DAM, MOA, MFA, CAM, IC, CBM, AMC, CC and LOC)	Maximum likelihood co-efficient, Significance, R^2 Statistic, ROC
AN-53	Martino(2011) (Martino et al., 2011), International Conference on Product Focused Software Process Improvement-Springer	Genetic algorithm with Support Vector Machine	PROMISE repository (jEdit version 4.0, 4.2, and 4.3)	Halstead, LOC, CK, Object Oriented	TPR, TPN, FPR, FPN and F-measure
AN-54	Mishra(2011) (Mishra & Shukla, 2011), ICCCT-IEEE	Naive Bayes based	Eclipse and KC1	CBO, DEAPTH, LCOM, NOC, DOC, FAN-IN, RFC, and WMC	Precision, recall, and F-measure
AN-55	Misirli(2011) (Misirli et al., 2011), Software Quality Journal, Springer	Ensemble Learning	Embedded Software projects and NASA (AR3, AR4, AR5, AE6, PC1, PC2, PC3, CM1)	Object Oriented, Halstead and CK	Probability of Detection, False Alarm, Code-Benefits Analysis, Precision, Balance Rates
AN-56	Recca(2011) (Kpodjedo et al., 2011), Empirical Software Engineering, Springer	Designed Evolution metrics	Rhino, ArgoUML, and Eclipse	Object Oriented system	% of error, % defect and Top rank
AN-57	Rodrigues(2012) (Rodriguez et al., 2012), Information Science, Elsevier	Evolutionary Algorithm	PROMISE dataset (KC1, KC2, KC3, MC2, MW1, CM1,PC1, and MC2)	LOC, McCabe, Halstead, Derived Halstead	Precision, Recall, Support, Complexity, Confidence, Significance, Weighted Related Accuracy, and Lift
AN-58	Song(2011) (Song et al., 2011), IEEE Transactions on Software Engineering	Evolution process based on Niave Bayes, J48 and OneR	NASA dataset (CM1, KC3, KC4, MW1, PC1, PC2, PCR, PC4, JM1, KC1, MC1, MC2, PC5, AR1, AR3, AR4, and AR6)	LOC, Halsted, McCabe, Derived Halsetead and Miscellaneous	ROC, TPR, and TNR
AN-59	Twala (2011) (Twala, 2011), ICCRD-2011, IEEE	Ensemble Learning (Boosting and Bagging)	NASA dataset (JM1 and PC1)	LOC, Halsted, McCabe, Derived Halstead	Accuracy and Error rate
AN-60	Chen(2012) (Ma et al., 2012), Information and Software Technology, Elsevier	Variation of Naive Bayes (Transfer Naive Bayes)	NASA (KC1, KC2, KC3, PC1, CM1, MC1, MC) SOFTLAB (AR3, AR4 and AR5)	McCabe, LOC, Halstead, Branch Count	ROC, F-measure, TPR and TNR
AN-61	Malhotra(2012) (Malhotra & Jain, 2012), Journal of Information Processing Systems	Multi Variant Logistic Regression, Bagging, Boosting, SVM, and Multilayer Perceptron	Open Source dataset (Apache POI)	Object Oriented, CK, QMOOD metrics	ROC, Precision, Recall, and Sensitivity
AN-62	Okutan(2014) (Okutan & Yildiz, 2014), Empirical Software Engineering, Springer	Bayesian Network based models	Xalan, Poi, Tomcat, Ant, Jedit, Velocity, Lucene, Xalan, and Ivy	Promised repository (RFC, LOC, CBO, WMC, NOC, DIT, NOD)	Average of score metrics
AN-63	Yang(2012) (He et al., 2012), Automated Software Engineering, Springer	J48, Naive Bayes, SVM, Decision Tree and Logistic Regression	Public dataset (Poi, Ant, Camel, Ive, Lucene, Synapse, Velocity, Xalan, and Xerces)	WMC, DIT, NOC, CBO, RFC, LCOM, LCOM3, NPM, DAM, MOA, MFA, CAM, IC, CBM, AMC, Ce, Ca, Max-CC, Avg-CC, LOC	Precision, Recall and F-measure
AN-64	Yu (2012) (Yu, 2012), Information Sciences, Elsevier	Ensemble Learning with SVP	NASA dataset (KC1, PC4)	Halstead, LOC, Condition Count, CC, Call pairs	Specificity, Sensitivity, and ROC
AN-65	Zhou(2012) (Li et al., 2012), Automated Software Engineering, Springer	Sample based Learning, Random sampling with traditional ML	PROMISE (Eclipse, JDT, SWT, Lucene, and Xalan)	Object Oriented, Program dependency, and Code complexity	Precision, Recall, Balance, and F-measure
AN-66	Cahill(2013) (Cahill et al., 2013), ASWEC.2013-IEEE	Rank sum classification, and Gradient Binning	NASA dataset (CM1, JM1, KC1, KC3, PC1, PC2, PC4, MW1)	Halstead, McCabe, and Code based	Precision, Recall, and F-measure
AN-67	Chen(2014) (Chen et al., 2014), Empirical Software Engineering, Springer	Procedure feature of Software processes, Formulation of problem	Personal projects	Distance metrics (Simple string distance, Nonlinear string distance)	Precision, Recall, and F-measure
AN-68	(Dejaeger et al., 2012), IEEE Transactions on Software Engineering	15 different Bayesian Network models, Markov blanket principle for feature selection	JM1, KC1, MC1, PC1, PC2, PC3, PC4, PC5, Eclipse2.0a, Eclipse2.1a, and Eclipse3.0a	McCabe, Halsted, and LOC	ROC, and H-measure

(*continued on next page*)

Table A.11 (*continued*)

Article no.	Article and journal name	Methodology	Dataset	Software Metrics	Performance metrics
AN-69	Laradji(2015) (Laradji et al., 2015), Information and Software technology, Elsevier	Ensemble Learning (Enhanced SVM and Random Forest)	PROMISE (PC2, PC4, Ant-1.7, Camel-1.6.and MC1)	Halstead, McCabe and LOC	Precision, Recall and ROC
AN-70	Mesquita(2016) (Mesquita et al., 2016), Applied Soft Computing	Weighted Extreme Learning Machine	Lucene, Equinox, Eclipse JDT, Eclipse PDE, and Mylyn	Process, and Change metrics	F-measure, and Accuracy
AN-71	Hamill(2017) (Hamill & Goseva-Popstojanova, 2017), Information and Software Technology, Elsevier	ZeroR, Naive Bayes, J48, PART and Oversampling, SMOTE, Under-sampling for (imbalance data)	NIL	NIL	Accuracy
AN-72	Kumar(2017) (Kumar et al., 2017), Journal of Systems and Software, Elsevier	SVM	Industry dataset (ant, Arc, Berek, Camel, Forest, Ivy, Jedit, Poi, Prop, Redacror, Skarbonka, Ynapse, Systemdata, Szybkafucha, Termoproject, Tomcat, Velocity, Workflow, Xalan, Xerces, and Zuzel)	Abreu MOOD suite, CK metrics	Accuracy, F-measure, Chi-Squared test, Gain ratio, and Info-gain
AN-73	Pandey(2018) (Pandey et al., 2018), Procedia Computer Science- Elsevier	Naive Bayes, and Bayesian Network	NASA dataset (PC1, PC2, PC3, PC4, PC5, KC1, JM1, PC2, PC3, ECL-2.0 and MC1)	Object Oriented, and McCabe	AUC of ROC, Defective rate, H-measure, Friedman test and Nemenyi test
AN-74	Song(2018) (Song et al., 2018), IEEE Transactions on Software Engineering	Class imbalance addressing (C4.5, SVM, LR, RIPPER, IBK, RF, and NB)	NASA, AEEEM, Softlab, Relink, and NetGene	CK, NET, PROC, CK+NET, CK+PROC, NET+PROC, CK+NET+PROC	MCC, F-measure, ROC
AN-75	Bennin(2018) (Bennin et al., 2018), IEEE Transactions on Software engineering	Proposed an approach for oversampling	PROMISE Data(Ant, Arc, Camel, Ivy, Jedit, Pbeam, Redaktor, Synapse, Systemdata, Tomcat, Xerces)	Static Code Metrics (WMC, DIT, NOC, CBO, RFC, LCOM, LCOM3, NPM, DAM, MOA, MFA, CAM, IC, CBM, AMC, Ca, Ce, CC, Max(CC), Avg(CC), and LOC)	Effect-size comparison, Recall ,and Probability of Faulty
AN-76	Nam(2018) (Nam et al., 2018), IEEE Transactions on Software Engineering	Metric selection and Metric matching	AEEEM, ReLink, MORPH, NASA, PROMISE, and SOFTLAB	Halstead, McCabe, Derived Halstead, Line of Count	Gain ration, Shi squared, Significance, and Relief-f
AN-77	Afzal(2008) (Afzal & Torkar, 2008), ICSEA 2008	Genetic Programming	Telecommunicationt company project dataset	Not mentioned	PLR, log(PLR), t-test, Braun static, AMSE, and K-S
AN-78	Afzal(2009) (Afzal et al., 2009), ISSBSE 2009, IEEE	Symbolic Regression using Genetic Programming	16 public application datasets	Not mentioned	Goodness of fit (Kolmogorov-Smirnov test), Model bias (U-plot), Model bias trend (Y-plot), Short-term predictability (Prequential likelihood), and Average relative error
AN-79	Aggarwal(2009) (Aggarwal et al., 2009), Software Process Improvement Practice	Principal-Component , Method and Logistic Regression	JAVA based application data	Object Oriented metrics- ACAIC, ACMIC, AMMIC, CBO, CBO1, DAC, DAC1,DCAEC, DCMEC, DMMEC, ICP, IHICP, MPC, NIHICP, OCAEC, OCAIC, OCMEC, OCMIC,OMMEC, OMMIC, RFC, ICP, IHICP, NIHICP, OCAIC, OMMIC, ICH, LCC, LCOM1, LCOM2, LCOM3, LCOM4, and TCC	Accuracy, Sensitivity, and Specificity
AN-80	Baisch(1997) (Baisch & Liedtke, 1997), IEEE International Conference	Mullilinear Discriminant Analysis, Fuzzy Expert Systems, Genetic Algorithms	Alcatel 1000 S12 Telecommunication system, COSMOS (ESPRIT) project	Complexity metrics, Number of (executable) statements, Statement complexity, Expression complexity, Data complexity, Depth of nesting (control flow), and Data base access (Number and complexity of data base accesses)	Multilinear Discriminant Analysis (MLDA) over test and training data, and Classification prediction
AN-81	Bangcharoensap(2012) (Bangcharoensap et al., 2012), IWESEP.2012	Mullilinear Discriminant Analysis, Fuzzy Expert Systems, Genetic Algorithms	Eclipse platform project data	SLOC, MLOC ,LOC, PAR, NOF, NOM, NSF, NSM, NBD, VG, DIT, LCOM, WMC, CodeChurn, LOCAddes, LOCDeleted, BugFixes, Refactorings, LastModDate	Accuracy, Sensitivity, and Specificity
AN-82	Bezerra(2007) (Bezerra et al., 2007),IJCNN 2007, IEEE	RBF and Neural Network	NASA dataset (CM1, JM1, KC1, KC2 and PC1)	Halstead, McCabe	Error rate in Probability of Detection
AN-83	Bibi(2008) (Bibi et al., 2008), Expert Systems with Applications, Elsevier	Regression via Classification (Jrip, J48, PART, SMO)	Pekka, ISBSG	FP, mts, Date, EI, EO, EE, ILF, EIF, Added, Changed, and Deleted	Mean Absolute Error, and Accuracy
AN-84			Promise data (AR3, AR4, AR5, iris, SYD1, SYD2)		

(*continued on next page*)

Table A.11 (*continued*)

Article no.	Article and journal name	Methodology	Dataset	Software Metrics	Performance metrics
	Bishnu(2012) (Bishnu & Bhattacherjee, 2012), IEEE Transactions on Knowledge and Data Engineering	Quad Tree-Based K-means Clustering Algorithm		Lines of Code, Cyclomatic Complexity, Unique Operator, Unique Operand, Total operator, Total Operand, Threshold vector [LoC, CC, UOp, UOpnd, TOp, TOpnd	Gain value, Error rate, and Sum of squares error
AN-85	Briand(2000) (Briand et al., 2000), Journal of Systems and Software, Elsevier	Logistic Regression	Public NASA dataset (PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, PC9, PC10, PC11, PC12, PC13)	Coupling, Cohesion, and Inheritance	p-value, kappa Ψ, Correctness, and Statistical analysis (R_2)
AN-86	Canfora(2005)(2018) (Canfora & Cerulo, 2005), METRICS 2005, IEEE	Naive Bayes Classier Bayesian Network based model	Open-source projects (klac, kpdf, kspread, firefox)	not mentioned	Precision, Recall, and Precision-Recall curve
AN-87	Challagulla(2006) (Challagulla et al., 2006), ICTAI'06, IEEE	Memory Based Reasoning	NASA (CM1, JM1a, JM1b, JM1c, JM1d, PC1)	McCabe, Halstead, Line count, Operator/Operand, and Branch	Accuracy, and Positive Prediction
AN-88	Challagulla(2006) (Challagulla et al., 2005), WORDS'05, IEEE	Decision Trees, Na've-Bayes, Logistic Regression, Nearest Neighbor, and 1-Rule	NASA (CM1, KC1, KC1, PC1)	McCabe, Halstead, Line Count, Operator/Operand, and Branch Count	Mean Absolute Error, and Error Difference Percentage
AN-89	Chiu (2011) (Chiu, 2011), Expert Systems with Applications, Elsevier	Swarm Optimization Algorithm, ANN	NASA(KC2)	Lines of Code Metrics, McCabe metrics, Basic Halstead, Derived Halstead, and Branch-Count	PMcNemar test (Chi-square, Asymptote Significance), TIE, TIIE, Absolute Gap,Precision, Recall and f-score
AN-90	Marco(2006) (D'Ambros et al., 2010), MSR'06, IEEE	Compared various classical defect prediction models	Open source dataset (Eclipse JDT Core, Eclipse PDE, Equinox framework, Mylyn, Apache Lucene)	McCabe, Halstead, CK, Line Count, Operator/ Operand, and Branch	Explanative power (adjusted R^2), and Predictive power (Spearman's correlation).
AN-91	Denaro(2002) (Denaro & Pezzè, 2002), ICSE, 2002	Logistic Regression	Apache 1.3, Apache 2.0.	LOC, eLOC, lLOC, Comments, dComm, eComm, Lines, CDENS, Functions, FP, FR, IC, Exits, V(g), eV (g), Knots, eKnots, CO, Blocks, AveBlockL, NES	Correctness, Completeness, and R^2
AN-92	Evett(1998) (Evett et al., 1998), ACGP, 1998	Genetic Programming	Industrial data	LOC, eLOC, lLOC, Comments, dComm, eComm, Lines, CDENS, Functions, FP, FR, IC, Exits, V(g), eV (g), Knots, eKnots, CO, Blocks, AveBlockL, NES, η_1. N_1, η_2. N_2, V (G), V2(G), V (G), LOC, ELOC	Rank percentile
AN-93	Haribar(2010) (Hribar & Duka, 2010), MIPRO' 2010	KNN and Fuzzy logic	Ericsson project	Not mentioned	Weibull distribution, Shape parameter and Scale parameter, TR (% of sensitive information)
AN-94	Illes(2008) (Illes-Seifert et al., 2008)	Defect Count using number of dependent/independent variable	Open source Java project (Ant, ApacheFOB, CDK, Freenet, Jetspeed2, Jmol, Oschache, Pentahu, TV-browser)	Product metrics and History metrics	Mean Defect Count, and min/max % of Significant Correct
AN-95	Jiang(2009) (Jiang & Cukic, 2009), ICPM 2009, ACM	Cost-Sensitive modeling	NASA dataset (JM1, PC1, MC2, PC5, PC2, PC3, PC4, CM1, MW1, KC1, KC3, KC4, MC2)	McCabe's and Source code	Precision, Recall, and ROC

Table A.12
Analysis of study.

Article no.	Article and Journal name	Methodology	Dataset	Software metrics	Performance metrics
AN-96	Kaminsky(2004) (Kaminsky & Boetticher, 2004),NAFIPS 2004, IEEE	Genetically Engineerable Evolvable Program	NASA dataset (KC2)	Cyclomatic Complexity, Essential Complexity, Module Design Complexity, Halstead length, Halstead volume, Halstead level,Halstead difficulty, Halstead intelligent content, Halstead programming effort, Halstead error estimate, Halstead programming time, Number of Unique Operators, Number of Unique Operands, Total operators, and Total operands	T-test, Error rate, and Average fitness result
AN-97	Kamiya(1999) (Kamiya et al., 1999), ISORC, 1999	Multivariate Logistic Regression	Created four checkpoints into the development process based on Object Oriented modeling technique	CK metrics (WMC, DIT, NOC, CBO, RFC, LCOM, NIV)	Correctness, Completeness and Error based Completeness
AN-98	Kaur(2009) (Kaur et al., 2009), ICMV, 2009, IEEE	K-means	NASA dataset (JM1, PC1 and CM1)	Requirement metrics, Code metrics and Join the requirement and Code metrics	Confusion metrics, ROC, Probability of Detection
AN-99	Khoshgoftaar(2009) (Khoshgoftaar & Gao, 2009), ICMLA 2009, IEEE	Wrapper-based attribute ranking technique, Random Undersampling Technique and SVM	Legacy Telecommunications (TC1, TC2, TC3, TC4), NASA (CM1, JM1, MW1, PC1)	Product metrics, Process metrics, Execution metrics, Halstead, McCabe, Line of Counts	Overall Accuracy, Arithmetic Mean, ROC, and AUC of PRC curve
AN-100	Khoshgoftaar(2002) (Khoshgoftaar & Seliya, 2002), METRICS 2002, IEEE	CART-LS (Least Square error),S-PLUS, and CART-LAD (Least Absolute Deviation)	Principal components checkpoints on legacy Telecommunications data	Call graph metrics and Control flow graph metrics	Average Absolute Error and Average Relative Error
AN-101	Khoshgoftaar(2003) (Khoshgoftaar & Seliya, 2003), Empirical Software Engineering, Springer	CART-LS (Least Square), S-PLUS, and CART-LAD (Least Absolute Deviation), Multiple Leaner Regression, ANN and CBR	Principal Components checkpoints on legacy Telecommunications data	Product metrics, Process metrics, Execution metrics	Average Absolute and Average Relative Error
AN-102	Khoshgoftaar(2006) (Khoshgoftaar et al., 2006), Software Quality Journal, Springer	Case-Based Reasoning	Legacy Telecommunications System	Product and Execution metrics and Domain metrics	Average Absolute Error and Average Relative Error
AN-103	Knab(2006) (Knab et al., 2006), MSR 2006, ACM	Decision Tree Learners	Layout modules of the Mozilla open source project (DOM, NewLayoutEngine, XPToolkit, NewHTMLStyleSystem, MathML, XML, and XSLT)	Source Code, Modification, and Defect report metrics	Correctly classified instances, Incorrectly classified instances
AN-104	Mahaweerawat(2002) (Mahaweerawat et al., 2002), ICIT 2002	Fuzzy clustering, MLP and RBF network	Not reviled	McCabe, Halstead, Density comment	Misclassification rate (type 1 error, type 2 error), Quality achieved and Verification cost
AN-105	Marcus(2008) (Marcus et al., 2008), IEEE transactions on Software Engineering	Multivariate Logistic Regression	Open source systems	OO metrics (LCOM1, LCOM2, LCOM3, LCOM4, C3, LCOM5, TCC, LCC)	Precision, Precision Rank, Correctness, Completeness, and R^2, C_0, C_1, C_2
AN-106	Masud(2011) (Masud et al., 2011), IEEE Traction on Knowledge and Data Engineering	Propose a data stream classification technique that integrates a novel class detection mechanism into traditional classifiers (Decision Tree and KNN)	Nil	Nil	Overall Error Rates, % of Novel class instances Misclassified as existing class, % Of Existing class instances Falsely identified as novel class
AN-107	Mertik(2006) (Mertik et al., 2006), ICSEA 2006, IEEE	DT, C4.5 (un-pruned and pruned), SVM (Linear kernel and RBF kernel) and Multi-method technique (Combined mentioned single method techniques within dynamical genetic environment)	NASA dataset(PC4, KC3, KC4)	McCabe, Error count, Halstead, Line of Code	Accuracy
AN-108	Monden(2013) (Monden et al., 2013),IEEE, Transaction on Software Engineering	RF, Linear Regression, CART	Applied over Telecommunication industry	Complexity metrics, Modules size, Design metrics	Fault Density, Accuracy, and Prediction Probability
AN-109	Nagappan(2010) (Nagappan et al., 2010), ISSRE 2010	Stepwise regression, Random data split	Windows Vista binaries exceeding 50 million LOC, Open source Eclipse	Determine change metrics, Temporal metrics, people metrics, and Churn metrics	Precision, and Recall
AN-110	Pizzi(2002) (Pizzi et al., 2002), IJCNN	NN	EVIDENT(JAVA/C++ based Open source software)	Object oriented (Meth, LOC, mLOC, TOK, mTOK, DEC, mDEC, WD, mWD, IC, Kid, Sib, Face, DI, rCC, IOC, RFO, LCOM)	Accuracy

(*continued on next page*)

Table A.12 (*continued*)

Article no.	Article and Journal name	Methodology	Dataset	Software metrics	Performance metrics
AN-111	Quah(2004) (Quah & Thwin, 2004), Information and Software Technology, Elsevier	General Regression Neural Network and Ward Neural Network	SQL defect dataset	NSC, NIUO, NDO, NT, NSCC, NJQ	R^2,r^2, Mean Square Error, Mean Absolute Error, Min. Absolute Error, Max. Absolute Error, t-Value, P-value
AN-112	Sandhu(2010) (Sandhu et al., 2010), ICCAE 2010, IEEE	DT, K-mean	NASA data(CM1)	Requirement metrics, Code metrics,	Precision, Probability of False Alarm, and Probability of Detection
AN-113	Seliya(2007) (Seliya & Khoshgoftaar, 2007), Software Quality Journal, Springer	Expectation Maximization Algorithm	NASA (JM1, KC1, KC2, KC3)	Line count, Halstead, McCabe, Branch Count	Type1 error, type2 error, and Overall error
AN-114	Shafi(2008) (Shafi et al., 2008), ICET 2008, IEEE	Instance Based Learning, Nearest Neighbor Based Learning, Locally Weighted Learning, Attribute Selected Classifier, DT, LR, RIPPER, ZeroR, OneR	PROMISE data (AR3 and JEditData)	Object Oriented	Root Mean Square Error, Aggregated Performance Index, Precision, Recall, Specificity, and Accuracy
AN-115	Singh(2009) (Singh et al., 2009), Product Focused Software Process Improvement, Springer	Gene Expression Programming	JAVA based project	OO metrics (CBO, LCOM, NOC, DIT, WMCRFC, NMP, LOC)	Probability of Misclassification, Accuracy, Precision, Recall, ROC, and Specificity
AN-116	Tomaszewski(2007) (Tomaszewski et al., 2007), Journal of Systems and Software, Elsevier	Expert Estimation	Telecommunication system's data	Class Level Metrics (Coup, NOC, WMC, RFC, DIT, LCOM, ClassStam, MaxCyc, ClassChang), Component Level (CompStmt, CompMeth, CompClass, CompChg)	Fault Density, % of fault
AN-117	Upadhya(1995) (Upadhya et al., 1995), Applied Mathematical Modelling, Elsevier	Mathematical Modeling of Casting Process	Casting a disc	Nil	Radiation loss, Time Consuing Analysis, and Time Distribution
AN-118	Wang(2004) (Wang et al., 2004), Tools with Artificial Intelligence 2004, IEEE	Neural Network, Genetic Programming	Telecommunications Software System data	Product metrics	Accuracy, Precision, and Recall
AN-119	Xing(2005) (Xing et al., 2005), ISSRE 2005, IEEE	SVM with different kernel	Software for a Medical Imaging System	Complexity metrics, Halsted, McCabe, Belady's bandwidth metric	Accuracy, Calculate correct classification, Type 2 error
AN-120	Yang(2007) (Yang et al., 2007), ICNC 2007, IEEE	ANN and Fuzzy logic	Expert Knowledge based project	DIT, Complexity and Reliability	Efficiency
AN-121	Yuan(2000) (Yuan et al., 2000), An Application of Fuzzy Clustering to Software are Quality Prediction, IEEE	Fuzzy Subtractive Clustering with module-order modeling	Telecommunications systems data	Product and Process metrics	Type1, Type2, Efficiency, and Effectiveness
AN-122	Zhang (2007) (Zhang, 2007), Applied Optics	Absolute linearity measurements, Optical society of America	Nil	Nil	Recall and Precision
AN-123	Zhao(1998) (Zhao et al., 1998), Information and Software Technology, Elsevier	Stastical method	Telecommunication domain data	Design metrics and Code metrics	F-measure, and Mean Square Error
AN-124	Zimmermann(2007) (Zimmermann et al., 2007), IEEE Computer Society	Logistic Regression, Linear and Regression	Eclipse bug data	Object Oriented (method, class, file and package)	Precision, Recall, and Accuracy
AN-125	Bansal (2017) (Bansal, 2017), Computer Languages, Systems and Structures, Elsevier	Hybrid Search based Algorithm	Open source math dataset	Object oriented (WMC, DIT, NOC, CBO, RFC, LCOMSLOC)	Accuracy, and G-mean
AN-126	Zhu(2018) (Zhu & Pham, 2018), Computer Languages, Systems and Structures, Elsevier	Non-Homogeneous Poisson Process	Three open source dataset	Change metrics	Failure rate, Least square estimate, Mean Squared Error, Variation, and Biased
AN-127	Shippey(2019) (Shippey et al., 2019), Information and Software Technology, Elsevier	Abstract syntax tree n-gram	Open source Commercial dataset	Static Code metrics, LOC, Variable Declarations and Halstead metrics	Median Effect Size, MCC, Precision, Recall, and f-score
AN-128	Bennin(2019) (Bennin et al., 2019), Empirical Software Engineering, Springer	Sampling techniques (SMOTE, Random sampling) Comind with classical ML methods	PROMISE dataset	Static and Process metrics	ROC, G-mean, Balance, and Probability of Detection
AN-129		Kernel Principal Component Analysis and Weighted Extreme Learning machine	PROMISE, and NASA dataset	Product, LOC, Halstead, McCabe	MCC, ROC, f-score, and G-mean

(*continued on next page*)

Table A.12 (*continued*)

Article no.	Article and Journal name	Methodology	Dataset	Software metrics	Performance metrics
	Xu(2019) (Xu et al., 2019), Information and Software Technology, Elsevier				
AN-130	Mori(2019) (Mori & Uchihira, 2019), Empirical Software Engineering, Springer	NB	NASA dataset(PC5, MC1)	Halsted, LOC, McCabe	ROC
AN-131	Ji(2019) (Ji et al., 2019), Software Quality Journal, Springer	Weighted- Naive Bayes	PROMISE dataset (Ivy, Lucene, Poi, Velocity, Xalan, CM1, MW1, PC1)	Object Oriented metrics, Halsted, and McCabe	Precision, Recall, and f-score
AN-132	Ni(2019) (Ni et al., 2019), Journal of Systems and Software, Elsevier	Multi- Objective Feature Selection Method applied K-NN, J48, LR, and NB	RELINK (Apache, Safe, Zxing) and PROMISE (Ant, Camel, Ivy, Jedit, Lucene, Poi, Synapse, Velocity, Xalan, Xerces)	CK metrics	ROC, and f-score
AN-133	Wei(2019) (Wei et al., 2019), Information Sciences, Elsevier	Local tangent space alignment Support Vector machine	NASA dataset (CM1, KC3, PC1, PC4, MC2 MW1)	Halstead, McCabe	Accuracy, Precision, Recall, and f-score
AN-134	Malhotra(2019) (Malhotra & Kamal, 2019), Neurocomputing, Elsevier	Sampling Methods (SMOTE, SPIDER, ADASYN)	NASA dataset (CM1, JM1, KC2 , KC3, MC1, MC2, MW1 , PC2 PC3)	Halstead, McCabe, Size, Number of Code	Sensitivity, Recall, Specificity, and ROC
AN-135	Chen(2019) (Chen et al., 2019), Information and Software Technology, Elsevier	Differential Evolutionary, SMOTE	Complexity, Coupling, Cohesion, Abstraction and Encapsulation metrics	PROMISE dataset (Ant, Camel, Ivy, Jedit, Synapse, Xalan, Xerces)	Cliff delta, Kendall, P-value
AN-136	Mauvsa(2017) (Mauša & Grbac, 2017), Applied Soft Computing, Elsevier	Genetic Programming over imbalance data, Ensemble Selection strategies for the Multi-Objective Evolutionary Algorithm	UCI dataset (Ion, Spt, Yst, Bal)	55 source code metrics	Accuracy, ROC, TPR, TNR, and FPR

Table A.13
Analysis of study.

Article no.	Article and Journal name	Methodology	Dataset	Software metrics	Performance metrics
AN-137	Kamei(2008) (Kamei et al., 2008), ACM-IEEE international symposium on Empirical software engineering and measurement	Association rule mining with Logistic Regression	Eclipse project	LOC, LOCM, DIT, WMC, SIX, CC, no. of (Methods, Children, Parameters, Statistic method, Statistic attribute, Overridden method)	F1-value, Support, Confidence and Lift
AN-138	Zheng (2010)(Zheng, 2010), Expert System with Application, Elsevier	Three cost-sensitive Boosting algorithms, Boost Neural Networks	KC1, KC2, CM1 and PC1	LOC, McCabe, Halstead, and Branch Count metrics	Normalized, and Expected Cost of Misclassification
AN-139	(Pandey et al., 2020), Expert System with Applications	EL, and Stacked Denoising Autoencoders	CM1, KC1, KC2, KC3, JM1, PC1, PC2, PC3, PC4, MC1, MC2, and MW1	Mccabe, Basic Halsted and Derived Halsted	ROC, f-score, and MCC,
AN-140	Czibula(2014) (Czibula et al., 2014), Information Sciences, Elsevier	Relational Association Rules Mining (extension of rule mining)	CM1, KC1, KC3, PC1, JM1, MC2, MW1, PC2,PC3, and PC4	DIT, NOC, FIN, and FOUT	ROC, Specificity, Precision, and Probability of Detection
AN-141	Ertutk(2015)(Erturk et al., 2015), Expert System with Applications, Elsevier	Adaptive Neuro Fuzzy Inference System, SVM, and ANN	CM1, JM1, KC1, KC2 and PC1	McCabe metric (LOC, v(g), ev(g), and iv (g))	ROC
AN-142	Abdi(2015) (Abdi et al., 2015), Innovations in Systems and Software Engineering, Springer	K-means Clustering Algorithm and Distance Based Multi-Objective Particle Swarm Optimization	JM1, KC1, PC1 and CM1	McCabe, Halstead, and Branch Count metrics	ROC, Accuracy and Precision
AN-143	Erturk(2015) (Erturk & Sezer, 2015), Journal of Software	Mamdani Style Fuzzy Inference System	KC1 project of PROMISE repository	Coupling Between Object, LOC and, Cyclomatic Complexity	ROC
AN-144	Jin(2015) (Jin & Jin, 2015), Applied Soft Computing, Elsevier	Hybrid ANN and Quantum Particle Swarm Optimization	PC1, JM1, KC1, and KC3	McCabe, McCabeand butler, Halstead	Sensitivity, Specificity, and ROC
AN-145	Liu(2015) (Liu et al., 2015), IEEE Transactions of Reliability	Threshold-based Clustering method, and Random Oversampling	Eclipse2.0, Eclipse2.1, Eclipse3.0, CM1, KC1, KC3, KC4, MC2, MW1, PC1, PC3, PC4, and PC4	LOC, McCabe Complexity metrics, and Halstead complexity metrics	ROC
AN-146	Wan(2017) (Wan et al., 2017), Book chapter at SEKE 2017	Cost Sensitive Method, Dictionary Learning	CM1, JM1, KC1, KC3, MC2, MC2, MW1, PC1, PC3, PC4, and PC5	Halsted, LOC, and McCabe	Precision, Recall, f-score, and Error rate (Type I, Type II)
AN-147	Tong(2018) (Tong et al., 2018), Information and Software Technology, Elsevier	Ensemble Learning, and Stacked Denoising Autoencoders	CM1, KC1, KC2, KC3, JM1, PC1, PC2, PC3, PC4, MC1, MC2, and MW1	Mccabe, Basic Halsted and Derived Halsted	ROC, f-score, and MCC
AN-148	Arshad(2019)(Arshad et al., 2019), IEEE Access	Decomposition of Fuzzy C-mean Clustering, Random Oversampling	Open source 18 industrial data (Auto mobile, Glass factory, Zoo etc.,)	Unknown	Precision, Recall, f-score, ROC and their mean values
AN-149	Shao(2018) (Shao et al., 2018), Expert System with Applications, Elsevier	Atomic Class-Association Rule Mining	CM1, JM1, KC1, KC3, KC4, MC1, MC2, MW1,PC1, PC2, PC3, PC4, AR1, AR4, and AR6	LOC, Halstead, McCabe and Miscellaneous	ROC, G-mean, and Balance
AN-150	Turabieh(2019) (Turabieh et al., 2019), Expert System with Applications, Elsevier	Layered Recurrent Neural Network for Feature Selection, Binary Genetic Algorithm, Binary Particle Swarm Optimization, Ant Colony Optimization	19 PROMISE dataset	Object Oriented metrics	ROC, Sensitivity, and Specificity
AN-151	Borandag(2019) (Borandag et al., 2019), Computer Science and Information Systems	Majority Vote based Feature Selections, J48, NB, and K-NN	CM1, JM1, KC1, PC1, Eclipse, Equinox, and JDT	McCabe, Halstead, Derived Hasltead, CK, Object Oriented, Entropy and Change	G-mean, Precision, and Recall
AN-152	Sunil (2018) (Sunil et al., 2018), SEKE	Bayesian Learner and its variation	PROMISE dataset	WMC, DIT, CBO, RFC, LCOM, NPM, DAM, MOA, MFA, CAM, IC, CBM, AMC, Ca, Ce, and CC	Type1 error, Type 2 error, and Normalized Penalty
AN-153	Devi(2012)(Devi et al., 2012), IJCSA	Hybrid Feature Selection and Hybrid Classifier	Open source	LOC, Halsted	Linear Discriminant Analysis
AN-154	Borandag(2019) (Borandag et al., 2019), Computer Science and Information Systems	J48, NB, K-NN (IBK), Majority Voting, Info gain, Symmetrical uncertainty, ReliefF	CM1, JM1, KC1, PC1, Eclipse Quinox, and JDT	McCabe, Halsted, Derived Halsted, Entropy, and Change	G-mean

Table A.14
Mapping of research queries with article number- Part 1.

Article no.	RQ-1	RQ-2	RQ-3	RQ-4	RQ-5	RQ-6	RQ-7	Score
AN-1	✓	✓	✓	✓	✓	✗	✗	10
AN-2	✓	✗	✓	✓	✓	✗	✓	10
AN-3	✓	✗	✓	✓	✓	✗	✓	10
AN-4	✓	✓	✓	✓	✓	✗	✗	10
AN-5	✓	✓	✓	✓	✓	✗	✗	10
AN-6	✓	✓	✓	✓	✓	✗	✗	10
AN-7	✓	✓	✓	✓	✓	✓	✓	13
AN-8	✓	✗	✓	✓	✓	✗	✗	9
AN-9	✓	✓	✓	✓	✓	✓	✗	12
AN-10	✓	✓	✓	✓	✓	✗	✗	10
AN-11	✓	✗	✓	✓	✓	✗	✗	9
AN-12	✓	✓	✓	✓	✓	✗	✓	11
AN-13	✓	✗	✓	✓	✓	✗	✓	10
AN-14	✓	✓	✓	✓	✓	✗	✓	11
AN-15	✓	✗	✓	✓	✓	✗	✗	8.5
AN-16	✓	✓	✓	✓	✓	✗	✓	11
AN-17	✓	✓	✓	✓	✓	✗	✓	11
AN-18	✓	✓	✓	✓	✓	✗	✗	10
AN-19	✓	✓	✓	✓	✓	✗	✓	11
AN-20	✓	✓	✓	✓	✓	✓	✓	12.5
AN-21	✓	✗	✓	✓	✓	✗	✓	9.5
AN-22	✓	✓	✓	✓	✓	✗	✗	10
AN-23	✓	✓	✓	✓	✓	✓	✓	12.5
AN-24	✓	✗	✓	✓	✓	✗	✗	8.5
AN-25	✓	✓	✓	✓	✓	✓	✗	11.5
AN-26	✓	✗	✓	✓	✓	✗	✗	8.5
AN-27	✓	✓	✓	✓	✓	✓	✓	13
AN-28	✓	✓	✓	✓	✓	✗	✓	10.5
AN-29	✓	✓	✓	✓	✓	✗	✓	12
AN-30	✓	✓	✓	✓	✓	✗	✗	9.5
AN-31	✓	✗	✓	✓	✓	✗	✓	9.5
AN-32	✓	✓	✓	✓	✓	✓	✓	13
AN-33	✓	✓	✓	✓	✓	✗	✗	10
AN-34	✓	✓	✓	✓	✓	✓	✗	12.5
AN-35	✓	✓	✓	✓	✓	✗	✗	10
AN-36	✗	✗	✗	✗	✗	✗	✗	1
AN-37	✓	✗	✓	✓	✓	✗	✗	8.5
AN-38	✓	✓	✓	✓	✓	✓	✗	11.5
AN-39	✓	✓	✓	✓	✓	✗	✗	10
AN-40	✓	✓	✓	✓	✓	✓	✓	12
AN-41	✓	✓	✓	✓	✓	✓	✗	12
AN-42	✓	✓	✓	✓	✓	✗	✓	11
AN-43		✓	✓	✓	✓	✗	✓	11.5
AN-44	✓	✓	✓	✓	✓	✗	✓	11
AN-45	✓	✓	✓	✗	✓	✗	✗	8.5
AN-46	✓	✓	✓	✓	✓	✗	✓	11
AN-47	✓	✓	✓	✓	✓	✗	✓	11
AN-48	✓	✓	✓	✓	✓	✓	✗	12
AN-49	✓	✓	✓	✓	✓	✓	✗	12
AN-50	✓	✓	✓	✓	✓	✗	✓	11
AN-51	✓	✓	✓	✓	✗	✗	✓	10
AN-52	✓	✓	✓	✓	✓	✓	✓	12.5
AN-53	✓	✓	✓	✓	✓	✓	✓	13
AN-54	✓	✗	✓	✓	✗	✗	✓	8.5
AN-55	✓	✓	✓	✓	✗	✗	✓	10
AN-56	✓	✓	✓	✓	✗	✗	✗	8.5
AN-57	✓	✓	✓	✓	✗	✓	✗	11
AN-58	✓	✓	✓	✓	✗	✓	✗	11
AN-59	✓	✓	✓	✓	✗	✓	✗	11
AN-60	✓	✓	✓	✓	✗	✗	✗	8.5
AN-61	✓	✓	✓	✓	✓	✓	✓	13
AN-62	✓	✓	✓	✓	✗	✗	✗	8.5

(*continued on next page*)

Table A.14 (*continued*)

Article no.	RQ-1	RQ-2	RQ-3	RQ-4	RQ-5	RQ-6	RQ-7	Score
AN-63	✓	✓	✓	✓	✓	✓	⋉	12
AN-64	✓	✓	✓	✓	✓	✓	⋉	12
AN-65	✓	✓	✓	✓	✓	✓	⋉	12
AN-66	✓	⋉	✓	✓	✓	✓	⋉	10
AN-67	✓	✓	✓	✓	✓	✓	✓	12.5
AN-68	✓	✓	✓	✓	✓	✓	✓	13
AN-69	✓	✓	✓	✓	✓	✓	⋉	12
AN-70	✓	✓	✓	✓	✓	⋉	⋉	11
AN-71	✓	✓	✓	⋉	⋉	✓	✓	10.5
AN-72	✓	✓	✓	✓	✓	⋉	⋉	11
AN-73	⋉	⋉	✓	✓	✓	✓	⋉	9.5
AN-74	⋉	⋉	✓	✓	✓	⋉	⋉	9.5
AN-75	⋉	⋉	✓	✓	✓	⋉	⋉	6.5
AN-76	⋉	⋉	✓	✓	✓	✓	⋉	8

Table A.15
Mapping of research queries with article number- Part 2.

Article no.	RQ-1	RQ-2	RQ-3	RQ-4	RQ-5	RQ-6	RQ-7	Score
AN-77	✓	✓	⋉	⋉	⋉	✓	⋉	5.5
AN-78	✓	⋉	✓	⋉	✓	⋉	⋉	5.5
AN-79	✓	✓	⋉	✓	⋉	⋉	⋉	5.5
AN-80	✓	⋉	✓	✓	✓	⋉	✓	8.5
AN-81	✓	✓	⋉	✓	⋉	✓	✓	9
AN-82	✓	✓	✓	⋉	✓	⋉	⋉	7.5
AN-83	✓	✓	✓	⋉	✓	⋉	✓	9
AN-84	✓	✓	✓	⋉	✓	✓	⋉	9
AN-85	✓	✓	⋉	⋉	✓	✓	⋉	7.5
AN-86	✓	✓	⋉	⋉	⋉	✓	⋉	5.5
AN-87	✓	⋉	⋉	✓	✓	✓	⋉	7.5
AN-88	✓	✓	⋉	✓	✓	✓	⋉	9
AN-89	✓	✓	✓	✓	✓	✓	⋉	11
AN-90	✓	✓	⋉	✓	✓	✓	⋉	9
AN-91	✓	⋉	⋉	✓	✓	✓	⋉	7.5
AN-92	✓	⋉	⋉	✓	✓	⋉	⋉	6.5
AN-93	✓	⋉	⋉	⋉	⋉	✓	⋉	4
AN-94	✓	✓	⋉	⋉	✓	✓	⋉	6.5
AN-95	✓	✓	⋉	✓	✓	✓	⋉	9
AN-96	✓	⋉	✓	✓	✓	✓	⋉	9
AN-97	✓	⋉	⋉	✓	⋉	✓	✓	7.5
AN-98	✓	⋉	✓	✓	✓	✓	⋉	9
AN-99	✓	✓	✓	✓	✓	✓	⋉	11
AN-100	✓	✓	✓	✓	✓	✓	⋉	11
AN-101	✓	✓	✓	✓	✓	✓	⋉	11
AN-102	✓	✓	✓	✓	✓	✓	⋉	11
AN-103	✓	⋉	✓	✓	✓	✓	⋉	9
AN-104	✓	✓	⋉	⋉	⋉	✓	⋉	5.5
AN-105	✓	✓	⋉	✓	⋉	✓	⋉	7.5
AN-106	✓	✓	⋉	⋉	⋉	✓	⋉	5.5
AN-107	✓	✓	✓	⋉	✓	✓	⋉	9
AN-108	✓	✓	✓	✓	⋉	✓	⋉	9
AN-109	✓	✓	⋉	✓	✓	✓	⋉	9
AN-110	✓	✓	✓	✓	✓	✓	⋉	11
AN-111	✓	✓	⋉	✓	✓	✓	⋉	9
AN-112	✓	⋉	✓	✓	✓	✓	⋉	9
AN-113	✓	✓	✓	✓	✓	✓	⋉	11.5
AN-114	✓	✓	⋉	⋉	✓	✓	⋉	7.5
AN-115	✓	⋉	✓	⋉	✓	✓	⋉	7.5
AN-116	✓	✓	✓	✓	⋉	✓	⋉	9
AN-117	✓	⋉	⋉	⋉	⋉	✓	⋉	4

(*continued on next page*)

Table A.15 (*continued*)

Article no.	RQ-1	RQ-2	RQ-3	RQ-4	RQ-5	RQ-6	RQ-7	Score
AN-118	✓	⋉	✓	⋉	⋉	✓	⋉	5.5
AN-119	✓	⋉	✓	✓	⋉	✓	⋉	7.5
AN-120	✓	⋉	✓	⋉	✓	✓	⋉	7.5
AN-121	✓	⋉	✓	⋉	⋉	✓	⋉	5.5
AN-122	⋉	⋉	⋉	⋉	⋉	✓	⋉	1.5
AN-123	✓	⋉	✓	✓	⋉	✓	⋉	7.5
AN-124	✓	⋉	✓	✓	✓	✓	⋉	9.5
AN-125	✓	✓	✓	✓	⋉	✓	✓	11.5
AN-126	✓	✓	✓	⋉	⋉	✓	⋉	7.5
AN-127	✓	✓	✓	⋉	⋉	✓	⋉	7.5
AN-128	✓	✓	✓	✓	✓	✓	⋉	10.5
AN-129	✓	✓	✓	✓	✓	✓	⋉	10.5
AN-130	✓	⋉	✓	✓	✓	✓	⋉	9
AN-131	✓	✓	✓	✓	✓	✓	⋉	10.5
AN-132	✓	✓	✓	✓	✓	✓	⋉	11
AN-133	✓	✓	✓	✓	✓	✓	✓	12.5
AN-134	✓	✓	✓	✓	✓	✓	✓	12.5
AN-135	✓	✓	✓	✓	✓	✓	⋉	11
AN-136	✓	✓	✓	✓	✓	✓	✓	12.5

Table A.16
Mapping of research queries with article number- Part 3.

Article no.	RQ-1	RQ-2	RQ-3	RQ-4	RQ-5	RQ-6	RQ-7	Score
AN-137	✓	⋉	⋉	✓	✓	⋉	⋉	7.5
AN-138	✓	✓	✓	✓	✓	⋉	⋉	10.5
AN-139	✓	✓	✓	✓	✓	✓	⋉	12.5
AN-140	✓	✓	✓	⋉	✓	✓	⋉	9.5
AN-141	✓	✓	✓	⋉	✓	✓	⋉	10
AN-142	✓	✓	✓	⋉	✓	✓	⋉	9.5
AN-143	✓	✓	✓	⋉	✓	⋉	⋉	8.5
AN-144	✓	✓	✓	✓	✓	✓	⋉	12
AN-145	✓	✓	✓	✓	✓	✓	⋉	12
AN-146	✓	⋉	⋉	✓	✓	⋉	⋉	6.5
AN-147	✓	✓	✓	✓	✓	✓	⋉	12.5
AN-148	⋉	✓	✓	⋉	⋉	⋉	⋉	2.5
AN-149	✓	✓	⋉	✓	✓	✓	⋉	9.5
AN-150	✓	✓	✓	✓	✓	✓	⋉	12
AN-151	✓	✓	✓	✓	✓	⋉	⋉	10
AN-152	✓	⋉	✓	✓	✓	⋉	⋉	8.5
AN-153	✓	⋉	✓	✓	⋉	✓	⋉	7.5
AN-154	✓	✓	✓	✓	✓	⋉	⋉	10.5

categories of various articles with their corresponding score as we have discussed in Section 3.3. Out of 154, 64 articles comes under excellent categories, 56 articles belongs to good categories, 29 under average and 5 under below average categorizes.

References

Abaei, G., & Selamat, A. (2014). A survey on software fault detection based on different prediction approaches. *Vietnam Journal of Computer Science, 1*(2), 79–95.

Abdi, Y., Parsa, S., & Seyfari, Y. (2015). A hybrid one-class rule learning approach based on swarm intelligence for software fault prediction. *Innovations in Systems and Software Engineering, 11*(4), 289–301.

Afzal, W. (2010). Using faults-slip-through metric as a predictor of fault-proneness. In *Software Engineering Conference (APSEC)* (pp. 414–422). Asia Pacific: IEEE.

Afzal, W., & Torkar, R. (2008). A comparative evaluation of using genetic programming for predicting fault count data. In *Software Engineering Advances, 2008. ICSEA '08. The Third International Conference on* (pp. 407–414). IEEE.

Afzal, W., Torkar, R., & Feldt, R. (2009). Search-based prediction of fault count data. In *Search Based Software Engineering, 2009 1st International Symposium on* (pp. 35–38). IEEE.

Aggarwal, K., Singh, Y., Kaur, A., & Malhotra, R. (2006). Empirical study of object-oriented metrics. *Journal of Object Technology, 5*(8), 149–173.

Aggarwal, K., Singh, Y., Kaur, A., & Malhotra, R. (2009). Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: A replicated case study. *Software Process: Improvement and Practice, 14*(1), 39–62.

Akiyama, F., (1971). An example of software system debugging. In: IFIP Congress (1), pp. 353-359.

Allen, E. (2002). *Bug patterns in Java*. APress.

Arisholm, E., Briand, L. C., & Fuglerud, M. (2007). Data mining techniques for building fault-proneness models in telecom java software. In *Software Reliability, 2007. ISSRE'07. The 18th IEEE International Symposium on* (pp. 215–224). IEEE.

Arisholm, E., Briand, L. C., & Johannessen, E. B. (2010). A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. *Journal of Systems and Software, 83*(1), 2–17.

Arshad, A., Riaz, S., & Jiao, L. (2019). Semi-supervised deep fuzzy c-mean clustering for imbalanced multi-class classification. *IEEE Access, 7*, 28100–28112.

Atkinson, R., & Flint, J. (2001). Accessing hidden and hard-to-reach populations: Snowball research strategies. *Social Research Update, 33*(1), 1–4.

Azar, D., & Vybihal, J. (2011). An ant colony optimization algorithm to improve software quality prediction models: Case of class stability. *Information and Software Technology, 53*(4), 388–393.

Baisch, E., & Liedtke, T. (1997). Comparison of conventional approaches and soft-computing approaches for software quality prediction. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on* (vol. 2, pp. 1045–1049). IEEE.

Bangcharoensap, P., Ihara, A., Kamei, Y., & Matsumoto, K.-I. (2012). Locating source code to be fixed based on initial bug reports-a case study on the eclipse project. In

Empirical Software Engineering in Practice (IWESEP), 2012 Fourth International Workshop on (pp. 10–15). IEEE.
Bansal, A. (2017). Empirical analysis of search based algorithms to identify change prone classes of open source software. *Computer Languages, Systems & Structures, 47*, 211–231.
Bengio, Y., & Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *Journal of Mchine Learning Research, 5*(Sep), 1089–1105.
Bennin, K. E., Keung, J., Phannachitta, P., Monden, A., & Mensah, S. (2018). Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. *IEEE Transactions on Software Engineering, 44*(6), 534–550.
Bennin, K. E., Keung, J. W., & Monden, A. (2019). On the relative value of data resampling approaches for software defect prediction. *Empirical Software Engineering, 24*(2), 602–636.
Bezerra, M. E., Oliveira, A. L., & Meira, S. R. (2007). A constructive rbf neural network for estimating the probability of defects in software modules. In *Neural Networks, 2007. IJCNN 2007. International Joint Conference on* (pp. 2869–2874). IEEE.
Bibi, S., Tsoumakas, G., Stamelos, I., & Vlahavas, I. (2008). Regression via classification applied on software defect estimation. *Expert Systems with Applications, 34*(3), 2091–2101.
Bishnu, P. S., & Bhattacherjee, V. (2012). Software fault prediction using quad tree-based k-means clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering, 24*(6), 1146–1150.
Borandag, E., Ozcift, A., Kilinc, D., & Yucalar, F. (2019). Majority vote feature selection algorithm in software fault prediction. *Computer Science and Information Systems, 16*(2), 515–539.
Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software, 80*(4), 571–583.
Briand, L. C., Wüst, J., Daly, J. W., & Porter, D. V. (2000). Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of Systems and Software, 51*(3), 245–273.
Cahill, J., Hogan, J. M., & Thomas, R. (2013). Predicting fault-prone software modules with rank sum classification. In *Software Engineering Conference (ASWEC), 2013 22nd Australian* (pp. 211–219). IEEE.
Canfora, G., & Cerulo, L. (2005). Impact analysis by mining software and change request repositories. In *Software Metrics, 2005. 11th IEEE International Symposium* (p. 9). IEEE.
Catal, C. (2011). Software fault prediction: A literature review and current trends. *Expert Systems with Applications, 38*(4), 4626–4636.
Catal, C. (2012). Performance evaluation metrics for software fault prediction studies. *Acta Polytechnica Hungarica, 9*(4), 193–206.
Catal, C., & Diri, B. (2009a). Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Information Sciences, 179*, 1040–1058.
Catal, C., & Diri, B. (2009b). A systematic review of software fault prediction studies. *Expert Systems with Applications, 36*(4), 7346–7354.
Catal, C., Diri, B., & Ozumut, B. (2007). An artificial immune system approach for fault prediction in object-oriented software. In *Dependability of Computer Systems, 2007. DepCoS-RELCOMEX'07. 2nd International Conference on* (pp. 238–245). IEEE.
Catal, C., Sevim, U., & Diri, B. (2009). Clustering and metrics thresholds based software fault prediction of unlabeled program modules. In *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on* (pp. 199–204). IEEE.
Catal, C., Sevim, U., & Diri, B. (2011). Practical development of an eclipse-based software fault prediction tool using naive bayes algorithm. *Expert Systems with Applications, 38*(3), 2347–2353.
Challagulla, V. U., Bastani, F. B., & Yen, I.-L. (2006). A unified framework for defect data analysis using the mbr technique. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on* (pp. 39–46). IEEE.
Challagulla, V. U., Bastani, F. B., Yen, I.-L., & Paul, R. A. (2005). Empirical assessment of machine learning based software defect prediction techniques. In *Object-Oriented Real-Time Dependable Systems, 2005. WORDS 2005. 10th IEEE International Workshop on* (pp. 263–270). IEEE.
Chen, N., Hoi, S. C., & Xiao, X. (2014). Software process evaluation: a machine learning framework with application to defect management process. *Empirical Software Engineering, 19*(6), 1531–1564.
Chen, X., Zhang, D., Zhao, Y., Cui, Z., & Ni, C. (2019). Software defect number prediction: Unsupervised vs supervised methods. *Information and Software Technology, 106*, 161–181.
Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering, 20*(6), 476–493.
Chiu, N.-H. (2011). Combining techniques for software quality classification: An integrated decision network approach. *Expert Systems with Applications, 38*(4), 4618–4625.
Czibula, G., Marian, Z., & Czibula, I. G. (2014). Software defect prediction using relational association rule mining. *Information Sciences, 264*, 260–278.
D'Ambros, M., Lanza, M., & Robbes, R. (2010). An extensive comparison of bug prediction approaches. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on* (pp. 31–41). IEEE.
de Carvalho, A. B., Pozo, A., Vergilio, S., & Lenz, A. (2008). Predicting fault proneness of classes trough a multiobjective particle swarm optimization algorithm. In *Tools with Artificial Intelligence, 2008. ICTAI'08. 20th IEEE International Conference on* (Vol. 2, pp. 387–394). IEEE.
De Carvalho, A. B., Pozo, A., & Vergilio, S. R. (2010). A symbolic fault-prediction model based on multiobjective particle swarm optimization. *Journal of Systems and Software, 83*(5), 868–882.
Dejaeger, K., Verbraken, T., & Baesens, B. (2012). Toward comprehensible software fault prediction models using bayesian network classifiers. *IEEE Transactions on Software Engineering, 39*(2), 237–257.
Denaro, G., & Pezzè, M. (2002). An empirical evaluation of fault-proneness models. In *Proceedings of the 24th International Conference on Software Engineering* (pp. 241–251). ACM.
Devi, C. A., Kannammal, K., & Surendiran, B. (2012). A hybrid feature selection model for software fault prediction. *International Journal of Computer Applications, 2*(2), 25–35.
Di Martino, S., Ferrucci, F., Gravino, C., & Sarro, F. (2011). A genetic algorithm to configure support vector machines for predicting fault-prone components. In *International Conference on Product Focused Software Process Improvement* (pp. 247–261). Springer.
Elish, K. O., & Elish, M. O. (2008). Predicting defect-prone software modules using support vector machines. *Journal of Systems and Software, 81*(5), 649–660.
Erturk, E., & Sezer, E. A. (2015). A comparison of some soft computing methods for software fault prediction. *Expert Systems with Applications, 42*(4), 1872–1879.
Erturk, E., & Sezer, E. A. (2015). Software fault inference based on expert opinion. *JSW, 10*(6), 757–766.
Erturk, E., & Sezer, E. A. (2016). Iterative software fault prediction with a hybrid approach. *Applied Soft Computing, 49*, 1020–1033.
Evett, M., Khoshgoftar, T., Chien, P.-D., & Allen, E. (1998). Gp-based software quality prediction. In *Proceedings of the Third Annual Conference Genetic Programming* (pp. 60–65).
Gondra, I. (2008). Applying machine learning to software fault-proneness prediction. *Journal of Systems and Software, 81*(2), 186–195.
Goodman, L. A. (1961). Snowball sampling. *The Annals of Mathematical Statistics*, 148–170.
Guo, L., Cukic, B., & Singh, H. (2003). Predicting fault prone modules by the dempster-shafer belief networks. In *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on* (pp. 249–252). IEEE.
Guo, L., Ma, Y., Cukic, B., & Singh, H. (2004). Robust prediction of fault-proneness by random forests. In *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on* (pp. 417–428). IEEE.
Gyimothy, T., Ferenc, R., & Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software Engineering, 31*(10), 897–910.
Hall, M. A., (2000). Correlation-based feature selection of discrete and numeric class machine learning.
Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2011). A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering, 38*(6), 1276–1304.
Halstead, M. H. (1977). *Elements of software science* (Vol. 7). Elsevier New York.
Hamill, M., & Goseva-Popstojanova, K. (2017). Analyzing and predicting effort associated with finding and fixing software faults. *Information and Software Technology, 87*, 1–18.
Hawkins, D. M. (2004). The problem of overfitting. *Journal of Chemical Information and Computer Sciences, 44*(1), 1–12.
He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284.
He, Z., Shu, F., Yang, Y., Li, M., & Wang, Q. (2012). An investigation on the feasibility of cross-project defect prediction. *Automated Software Engineering, 19*(2), 167–199.
Herbold, S., Trautsch, A., & Grabowski, J. (2017). A comparative study to benchmark cross-project defect prediction approaches. *IEEE Transactions on Software Engineering, 44*(9), 811–833.
Hosseini, S., Turhan, B., & Mäntylä, M. (2018). A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction. *Information and Software Technology, 95*, 296–312.
Hribar, L., & Duka, D. (2010). Software component quality prediction using knn and fuzzy logic. In *MIPRO, 2010 Proceedings of the 33rd International Convention* (pp. 402–408). IEEE.
Huda, S., Liu, K., Abdelrazek, M., Ibrahim, A., Alyahya, S., Al-Dossari, H., & Ahmad, S. (2018). An ensemble oversampling model for class imbalance problem in software defect prediction. *IEEE Access, 6*, 24184–24195.
Illes-Seifert, T., & Paech, B. (2008). Exploring the relationship of a file's history and its fault-proneness: An empirical study. In *Practice and Research Techniques, 2008. TAIC PART'08. Testing: Academic & Industrial Conference* (pp. 13–22). IEEE.
Jalali, S., & Wohlin, C. (2012). Systematic literature studies: database searches vs. backward snowballing. In *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 29–38). IEEE.
Ji, H., Huang, S., Wu, Y., Hui, Z., & Zheng, C. (2019). A new weighted naive bayes method based on information diffusion for software defect prediction. *Software Quality Journal*, 1–46.
Jiang, Y., & Cukic, B. (2009). Misclassification cost-sensitive fault prediction models. In *Proceedings of the 5th international conference on predictor models in software engineering* (p. 20). ACM.
Jiang, Y., Cukic, B., & Ma, Y. (2008). Techniques for evaluating fault prediction models. *Empirical Software Engineering, 13*(5), 561–595.
Jiang, Y., C, B., & Menzies, T. (2007). Fault prediction using early lifecycle data. In *Software Reliability, 2007. ISSRE'07. The 18th IEEE International Symposium on* (pp. 237–246). IEEE.
Jin, C., & Jin, S.-W. (2015). Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization. *Applied Soft Computing, 35*, 717–725.
Kamei, Y., Monden, A., Morisaki, S., & Matsumoto, K.-I. (2008). A hybrid faulty module prediction using association rule mining and logistic regression analysis. In

Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement (pp. 279–281).
Kaminsky, K., & Boetticher, G., (2004). Building a genetically engineerable evolvable program (geep) using breadth-based explicit knowledge for predicting software defects. In: Fuzzy Information, 2004. Processing NAFIPS'04. IEEE Annual Meeting of the. vol. 1. IEEE, pp. 10–15.
Kamiya, T., Kusumoto, S., & Inoue, K., (1999). Prediction of fault-proneness at early phase in object-oriented development. In: Object-Oriented Real-Time Distributed Computing, 1999. (ISORC'99) Proceedings. 2nd IEEE International Symposium on. IEEE, pp. 253–258.
Kanmani, S., Uthariaraj, V. R., Sankaranarayanan, V., & Thambidurai, P. (2007). Object-oriented software fault prediction using neural networks. *Information and Software Technology, 49*(5), 483–492.
Karim, S., Warnars, H. L. H. S., Gaol, F. L., Abdurachman, E., Soewito, B., et al. (2017). Software metrics for fault prediction using machine learning approaches: A literature review with promise repository dataset. In *2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)* (pp. 19–23). IEEE.
Kaur, A., & Malhotra, R., (2008). Application of random forest in predicting fault-prone classes. In: Advanced Computer Theory and Engineering, 2008. ICACTE'08. International Conference on. IEEE, pp. 37–43.
Kaur, A., Sandhu, P. S., & Bra, A. S., (2009). Early software fault prediction using real time defect dataknab2006predicting. In: Machine Vision, 2009. ICMV'09. Second International Conference on. IEEE, pp. 242–245.
Keele, S., et al., 2007. Guidelines for performing systematic literature reviews in software engineering. In: Technical report, Ver. 2.3 EBSE Technical Report. EBSE. sn.
Khoshgoftaar, T.M. & Gao, K. (2009). Feature selection with imbalanced data for software defect prediction. In: Machine Learning and Applications, 2009. ICMLA'09. International Conference on. IEEE, pp. 235–240.
Khoshgoftaar, T. M., Gao, K., & Seliya, N. (2010). Attribute selection and imbalanced data: Problems in software defect prediction. In: 2010 22nd IEEE International conference on tools with artificial intelligence. Vol. 1. IEEE, pp. 137–144.
Khoshgoftaar, T. M. & Seliya, N. (2002). Tree-based software quality estimation models for fault prediction. In: Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on. IEEE, pp. 203–214.
Khoshgoftaar, T. M., & Seliya, N. (2003). Fault prediction modeling for software quality estimation: Comparing commonly used techniques. *Empirical Software Engineering, 8*(3), 255–283.
Khoshgoftaar, T. M., Seliya, N., & Sundaresh, N. (2006). An empirical study of predicting software faults with case-based reasoning. *Software Quality Journal, 14*(2), 85–111.
Kim, S., Whitehead, E. J., Jr, & Zhang, Y. (2008). Classifying software changes: Clean or buggy? *IEEE Transactions on Software Engineering, 34*(2), 181–196.
Kim, S., Zhang, H., Wu, R., & Gong, L. (2011). Dealing with noise in defect prediction. In *2011 33rd International Conference on Software Engineering (ICSE)* (pp. 481–490). IEEE.
Kitchenham, B. & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.
Knab, P., Pinzger, M., & Bernstein, A. (2006). Predicting defect densities in source code files with decision tree learners. In *Proceedings of the 2006 international workshop on Mining software repositories* (pp. 119–125). ACM.
Koru, A. G., & Liu, H. (2005). Building effective defect-prediction models in practice. *IEEE Software, 22*(6), 23–29.
Kpodjedo, S., Ricca, F., Galinier, P., Guéhéneuc, Y.-G., & Antoniol, G. (2011). Design evolution metrics for defect prediction in object oriented systems. *Empirical Software Engineering, 16*(1), 141–175.
Kumar, L., Sripada, S. K., Sureka, A., & Rath, S. K. (2017). Effective fault prediction model developed using least square support vector machine (lssvm). *Journal of Systems and Software*.
Laradji, I. H., Alshayeb, M., & Ghouti, L. (2015). Software defect prediction using ensemble learning on selected features. *Information and Software Technology, 58*, 388–402.
Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering, 34*(4), 485–496.
Li, L., Lessmann, S., & Baesens, B. (2019). Evaluating software defect prediction performance: an updated benchmarking study. arXiv preprint arXiv:1901.01726.
Li, M., Zhang, H., Wu, R., & Zhou, Z.-H. (2012). Sample-based software defect prediction with active and semi-supervised learning. *Automated Software Engineering, 19*(2), 201–230.
Li, Z., Jing, X.-Y., & Zhu, X. (2018). Progress on approaches to software defect prediction. *IET Software, 12*(3), 161–175.
Li, Z., & Reformat, M. (2007). A practical method for the software fault-prediction. In: Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on. IEEE, pp. 659–666.
Liu, W., Liu, S., Gu, Q., Chen, J., Chen, X., & Chen, D. (2015). Empirical studies of a two-stage data preprocessing approach for software fault prediction. *IEEE Transactions on Reliability, 65*(1), 38–53.
Liu, Y., Khoshgoftaar, T. M., & Seliya, N. (2010). Evolutionary optimization of software quality modeling with multiple repositories. IEEE Transactions on Software Engineering 36, 852-864.
Lorenz, M., & Kidd, J. (1994). *Object-oriented software metrics: a practical guide*. Prentice-Hall Inc.
Ma, Y., Guo, L., & Cukic, B. (2006). A statistical framework for the prediction of fault-proneness. *Advances in Machine Learning Application in Software Engineering*, 237–265.
Ma, Y., Luo, G., Zeng, X., & Chen, A. (2012). Transfer learning for cross-company software defect prediction. *Information and Software Technology, 54*(3), 248–256.
Mahaweerawat, A., Sophasathit, P., & Lursinsap, C., (2002). Software fault prediction using fuzzy clustering and radial basis function network. In: International conference on intelligent technologies. Vietnam, 304. Vol. 313.
Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing, 27*, 504–518.
Malhotra, R., & Jain, A. (2011). Software fault prediction for object oriented systems: a literature review. *ACM SIGSOFT Software Engineering Notes, 36*(5), 1–6.
Malhotra, R., & Jain, A. (2012). Fault prediction using statistical and machine learning methods for improving software quality. *Journal of Information Processing Systems, 8*(2), 241–262.
Malhotra, R., & Kamal, S. (2019). An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data. *Neurocomputing*.
Malhotra, R., Kaur, A., & Singh, Y. (2010). Empirical validation of object-oriented metrics for predicting fault proneness at different severity levels using support vector machines. *International Journal of System Assurance Engineering and Management, 1*(3), 269–281.
Malhotra, R., & Khanna, M. (2018). Particle swarm optimization-based ensemble learning for software change prediction. *Information and Software Technology, 102*, 65–84.
Malhotra, R., & Singh, Y. (2011). On the applicability of machine learning techniques for object oriented software fault prediction. *Software Engineering: An International Journal, 1*(1), 24–37.
Marcus, A., Poshyvanyk, D., & Ferenc, R. (2008). Using the conceptual cohesion of classes for fault prediction in object-oriented systems. *IEEE Transactions on Software Engineering, 34*(2), 287–300.
Masud, M., Gao, J., Khan, L., Han, J., & Thuraisingham, B. M. (2011). Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering, 23*(6), 859–874.
Mauša, G., & Grbac, T. G. (2017). Co-evolutionary multi-population genetic programming for classification in software defect prediction: An empirical case study. *Applied Soft Computing, 55*, 331–351.
McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on Software Engineering, 4*, 308–320.
Menzies, T., Greenwald, J., & Frank, A. (2007). Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering, 33*(1), 2–13.
Menzies, T., Turhan, B., Bener, A., Gay, G., Cukic, B., & Jiang, Y. (2008). Implications of ceiling effects in defect predictors. In *Proceedings of the 4th international workshop on Predictor models in software engineering* (pp. 47–54). ACM.
Mertik, M., Lenic, M., Stiglic, G., & Kokol, P. (2006). Estimating software quality with advanced data mining techniques. In *Software Engineering Advances, International Conference on* (p. 19). IEEE.
Mesquita, D. P., Rocha, L. S., Gomes, J. P. P., & Neto, A. R. R. (2016). Classification with reject option for software defect prediction. *Applied Soft Computing, 49*, 1085–1093.
Mishra, B., & Shukla, K. (2011). Impact of attribute selection on defect proneness prediction in oo software. In *Computer and Communication Technology (ICCCT), 2011 2nd International Conference on* (pp. 367–372). IEEE.
Misirli, A. T., Bener, A. B., & Turhan, B. (2011). An industrial case study of classifier ensembles for locating software defects. *Software Quality Journal, 19*(3), 515–536.
Mitchell, T. M., (1997). Machine learning (mcgraw-hill international editions computer science series).
Monden, A., Hayashi, T., Shinoda, S., Shirai, K., Yoshida, J., Barker, M., & Matsumoto, K. (2013). Assessing the cost effectiveness of fault prediction in acceptance testing. *IEEE Transactions on Software Engineering, 39*(10), 1345–1357.
Mori, T., & Uchihira, N. (2019). Balancing the trade-off between accuracy and interpretability in software defect prediction. *Empirical Software Engineering, 24*(2), 779–825.
Moser, R., Pedrycz, W., & Succi, G. (2008). A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In *Proceedings of the 30th international conference on Software engineering* (pp. 181–190). ACM.
Nagappan, N., Zeller, A., Zimmermann, T., Herzig, K., & Murphy, B. (2010). Change bursts as defect predictors. In *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on* (pp. 309–318). IEEE.
Nam, J. (2014). *Survey on software defect prediction. Department of Compter Science and Engineerning*. Tech. Rep. The Hong Kong University of Science and Technology.
Nam, J., Fu, W., Kim, S., Menzies, T., & Tan, L. (2018). Heterogeneous defect prediction. *IEEE Transactions on Software Engineering, 44*(9), 874–896.
Ni, C., Chen, X., Wu, F., Shen, Y., & Gu, Q. (2019). An empirical study on pareto based multi-objective feature selection for software defect prediction. *Journal of Systems and Software*.
Okutan, A., & Yildiz, O. T. (2014). Software defect prediction using bayesian networks. *Empirical Software Engineering, 19*(1), 154–181.
Pai, G. J., & Dugan, J. B. (2007). Empirical analysis of software fault content and fault proneness using bayesian methods. *IEEE Transactions on software Engineering, 33*(10).
Pandey, S. K., Mishra, R. B., & Tripathi, A. K. (2020). Bpdet: An effective software bug prediction model using deep representation and ensemble learning techniques. *Expert Systems with Applications, 144*, Article 113085.
Pandey, S. K., Mishra, R. B., & Triphathi, A. K. (2018). Software bug prediction prototype using bayesian network classifier: A comprehensive model. *Procedia Computer Science, 132*, 1412–1421.
Pandey, S. K., & Tripathi, A. K. (2020). Bcv-predictor: A bug count vector predictor of a successive version of the software system. *Knowledge-Based Systems, 105924*.
Pendharkar, P. C. (2010). Exhaustive and heuristic search approaches for learning a software defect prediction model. *Engineering Applications of Artificial Intelligence, 23*(1), 34–40.

Pizzi, N. J., Summers, A. R., & Pedrycz, W., (2002). Software quality prediction using median-adjusted class labels. In: Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on. Vol. 3. IEEE, pp. 2405–2409.

Quah, T.-S., & Thwin, M. M. T. (2004). Prediction of software development faults in pl/sql files using neural network models. *Information and Software Technology, 46*(8), 519–523.

Radjenović, D., Heričko, M., Torkar, R., & Živkovič, A. (2013). Software fault prediction metrics: A systematic literature review. *Information and Software Technology, 55*(8), 1397–1418.

Rathore, S. S., & Kumar, S. (2017). Towards an ensemble based system for predicting the number of software faults. *Expert Systems with Applications, 82*, 357–382.

Rathore, S. S., & Kumar, S. (2019). A study on software fault prediction techniques. *Artificial Intelligence Review, 51*(2), 255–327.

Rätsch, G., Onoda, T., & Müller, K. R. (1998). An improvement of adaboost to avoid overfitting. In *Proc. of the Int. Conf. on Neural Information Processing. Citeseer*.

Rodríguez, D., Ruiz, R., Riquelme, J. C., & Aguilar-Ruiz, J. S. (2012). Searching for rules to detect defective modules: A subgroup discovery approach. *Information Sciences, 191*, 14–30.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one, 10*(3).

Sandhu, P. S., Goel, R., Brar, A. S., Kaur, J., & Anand, S., (2010). A model for early prediction of faults in software systems. In: Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on. Vol. 4. IEEE, pp. 281–285.

Sayyad Shirabad, J., & Menzies, T., (2005). The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada. http://promise.site.uottawa.ca/SERepository.

Seiffert, C., Khoshgoftaar, T. M., & Van Hulse, J. (2009). Improving software-quality predictions with data sampling and boosting. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 39*(6), 1283–1294.

Seliya, N., & Khoshgoftaar, T. M. (2007). Software quality estimation with limited fault data: a semi-supervised learning perspective. *Software Quality Journal, 15*(3), 327–344.

Seliya, N., Khoshgoftaar, T. M., & Van Hulse, J. (2010). Predicting faults in high assurance software. In *High-Assurance Systems Engineering (HASE), 2010 IEEE 12th International Symposium on* (pp. 26–34). IEEE.

Shafi, S., Hassan, S.M., Arshaq, A., Khan, M.J., & Shamail, S., 2008. Software quality prediction techniques: A comparative analysis. In: Emerging Technologies, 2008. ICET 2008. 4th International Conference on. IEEE, pp. 242–246.

Shao, Y., Liu, B., Wang, S., & Li, G. (2018). A novel software defect prediction based on atomic class-association rule mining. *Expert Systems with Applications, 114*, 237–254.

Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data quality: Some comments on the nasa software defect datasets. *IEEE Transactions on Software Engineering, 39*(9), 1208–1215.

Sherer, S. A. (1995). Software fault prediction. *Journal of Systems and Software, 29*(2), 97–105.

Shippey, T., Bowes, D., & Hall, T. (2019). Automatically identifying code features for software defect prediction: Using ast n-grams. *Information and Software Technology, 106*, 142–160.

Singh, P., & Verma, S., (2009). An investigation of the effect of discretization on defect prediction using static measures. In: Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT'09. International Conference on. IEEE, pp. 837–839.

Singh, P. K., Panda, R., & Sangwan, O. P. (2015). A critical analysis on software fault prediction techniques. *World Applied Sciences Journal, 33*(3), 371–379.

Singh, Y., Kaur, A., & Malhotra, R. (2009). Prediction of software quality model using gene expression programming. *Product-Focused Software Process Improvement*, 43–58.

Singh, Y., Kaur, A., & Malhotra, R. (2009). Software fault proneness prediction using support vector machines. *Proceedings of the World Congress on Engineering, 1*, 1–3.

Singh, Y., Kaur, A., & Malhotra, R. (2010). Empirical validation of object-oriented metrics for predicting fault proneness models. *Software Quality Journal, 18*(1), 3.

Singh, Y., Kaur, A., & Malhotra, R. (2010). Prediction of fault-prone software modules using statistical and machine learning methods. *International Journal of Computer Applications, 1*(22), 8–15.

Song, Q., Guo, Y., & Shepperd, M. (2018). A comprehensive investigation of the role of imbalanced learning for software defect prediction. *IEEE Transactions on Software Engineering*.

Song, Q., Jia, Z., Shepperd, M., Ying, S., & Liu, J. (2011). A general software defect-proneness prediction framework. *IEEE Transactions on Software Engineering, 37*(3), 356–370.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.

Staples, M., & Niazi, M. (2007). Experiences using systematic review guidelines. *Journal of Systems and Software, 80*(9), 1425–1437.

Sunil, J.M., Kumar, L., & Neti, L.B.M., (2018). Bayesian logistic regression for software defect prediction (s). In: SEKE. pp. 421–420.

Tahir, A., Bennin, K.E., MacDonell, S.G., & Marsland, S., (2018). Revisiting the size effect in software fault prediction models. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 1-10.

Tantithamthavorn, C., McIntosh, S., Hassan, A. E., & Matsumoto, K. (2016). Comments on 'researcher bias: The use of machine learning in software defect prediction'. *IEEE Transactions on Software Engineering, 42*(11), 1092–1094.

Tomaszewski, P., Håkansson, J., Grahn, H., & Lundberg, L. (2007). Statistical models vs. expert estimation for fault prediction in modified code–an industrial case study. *Journal of Systems and Software, 80*(8), 1227–1238.

Tong, H., Liu, B., & Wang, S. (2018). Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning. *Information and Software Technology, 96*, 94–111.

Tosun, A., Turhan, B., & Bener, A. (2009). Validation of network measures as indicators of defective modules in software systems. In *Proceedings of the 5th international conference on predictor models in software engineering* (p. 5). ACM.

Turabieh, H., Mafarja, M., & Li, X. (2019). Iterated feature selection algorithms with layered recurrent neural network for software fault prediction. *Expert Systems with Applications, 122*, 27–42.

Turhan, B., & Bener, A., (2007a). A multivariate analysis of static code attributes for defect prediction. In: Quality Software, 2007. QSIC'07. Seventh International Conference on. IEEE, pp. 231–237.

Turhan, B., & Bener, A. (2009). Analysis of naive bayes- assumptions on software fault data: An empirical study. *Data & Knowledge Engineering, 68*(2), 278–290.

Turhan, B., & Bener, A.B., (2007b). Software defect prediction: Heuristics for weighted naïve bayes. In: ICSOFT (SE). pp. 244–249.

Turhan, B., Kocak, G., & Bener, A., (2008). Software defect prediction using call graph based ranking (cgbr) framework. In: Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference. IEEE, pp. 191–198.

Turhan, B., Menzies, T., Bener, A. B., & Di Stefano, J. (2009). On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering, 14*(5), 540–578.

Twala, B. (2011). Software faults prediction using multiple classifiers. In: Computer Research and Development (ICCRD), 2011 3rd International Conference on. Vol. 4. IEEE, pp. 504–510.

Upadhya, G., Das, S., Chandra, U., & Paul, A. (1995). Modelling the investment casting process: a novel approach for view factor calculations and defect prediction. *Applied Mathematical Modelling, 19*(6), 354–362.

Vandecruys, O., Martens, D., Baesens, B., Mues, C., De Backer, M., & Haesen, R. (2008). Mining software repositories for comprehensible software fault prediction models. *Journal of Systems and Software, 81*(5), 823–839.

Wagner, S. (2006). A literature survey of the quality economics of defect-detection techniques. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering* (pp. 194–203).

Wahono, R. S. (2015). A systematic literature review of software defect prediction. *Journal of Software Engineering, 1*(1), 1–16.

Wan, H., Wu, G., Cheng, M., Huang, Q., Wang, R., & Yuan, M., (2017). Software defect prediction using dictionary learning. In: SEKE. pp. 335–340.

Wang, Q., Yu, B., & Zhu, J., (2004). Extract rules from software quality prediction model based on neural network. In: Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on. IEEE, pp. 191–195.

Wang, S., & Yao, X. (2013). Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability, 62*(2), 434–443.

Wei, H., Hu, C., Chen, S., Xue, Y., & Zhang, Q. (2019). Establishing a software defect prediction model via effective dimension reduction. *Information Sciences, 477*, 399–409.

Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology, 54*(1), 41–59.

Weyuker, E. J., Ostrand, T. J., & Bell, R. M. (2010). Comparing the effectiveness of several modeling methods for fault prediction. *Empirical Software Engineering, 15*(3), 277–295.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wohlin, C., Mendes, E., Felizardo, K. R., & Kalinowski, M. (2020). Guidelines for the search strategy to update systematic literature reviews in software engineering. *Information and Software Technology, 127*, Article 106366.

Xia, X., Lo, D., Pan, S. J., Nagappan, N., & Wang, X. (2016). Hydra: Massively compositional model for cross-project defect prediction. *IEEE Transactions on Software Engineering, 42*(10), 977–998.

Xing, F., Guo, P., & Lyu, M. R., (2005). A novel method for early software quality prediction based on support vector machine. In: Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on. IEEE, pp. 10–pp.

Xu, Z., Liu, J., Luo, X., Yang, Z., Zhang, Y., Yuan, P., Tang, Y., & Zhang, T. (2019). Software defect prediction based on kernel pca and weighted extreme learning machine. *Information and Software Technology, 106*, 182–200.

Yang, B., Yao, L., & Huang, H. -Z., (2007). Early software quality prediction based on a fuzzy neural network model. In: Natural Computation, 2007. ICNC 2007. Third International Conference on. Vol. 1. IEEE, pp. 760–764.

Yang, X., & Wen, W. (2018). Ridge and lasso regression models for cross-version defect prediction. *IEEE Transactions on Reliability, 67*(3), 885–896.

Yu, L. (2012). An evolutionary programming based asymmetric weighted least squares support vector machine ensemble learning methodology for software repository mining. *Information Sciences, 191*, 31–46.

Yuan, X., Khoshgoftaar, T. M., Allen, E. B., & Ganesan, K., (2000). An application of fuzzy clustering to software quality prediction. In: Application-Specific Systems and Software Engineering Technology, 2000. Proceedings. 3rd IEEE Symposium on. IEEE, pp. 85–90.

Zhang, Z. M. (2007). Comments on. *Applied Optics, 46*(25), 6483–6484.

Zhao, M., Wohlin, C., Ohlsson, N., & Xie, M. (1998). A comparison between software design and code metrics for the prediction of software fault content. *Information and Software Technology, 40*(14), 801–809.

Zheng, J. (2010). Cost-sensitive boosting neural networks for software defect prediction. *Expert Systems with Applications, 37*(6), 4537–4543.

Zhou, Y., & Leung, H. (2006). Empirical analysis of object-oriented design metrics for predicting high and low severity faults. *IEEE Transactions on Software Engineering, 32*(10), 771–789.

Zhou, Y., Xu, B., & Leung, H. (2010). On the ability of complexity metrics to predict fault-prone classes in object-oriented systems. *Journal of Systems and Software, 83*(4), 660–674.

Zhu, M., & Pham, H. (2018). A two-phase software reliability modeling involving with software fault dependency and imperfect fault removal. *Computer Languages, Systems & Structures, 53*, 27–42.

Zimmermann, T., Nagappan, N., Gall, H., Giger, E., & Murphy, B. (2009). Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering* (pp. 91–100). ACM.

Zimmermann, T., Premraj, R., & Zeller, A., (2007). Predicting defects for eclipse. In: Proceedings of the third international workshop on predictor models in software engineering. IEEE Computer Society, p. 9.