e-Informatica Software Engineering Journal, Volume 13, Issue 1, 2019, pages: 227–259, DOI 10.5277/e-Inf190107

Software Change Prediction: A Systematic Review and Future Guidelines

Ruchika Malhotra*, Megha Khanna**
*Department of Computer Science & Engineering, Delhi Technological University
**Sri Guru Gobind Singh College of Commerce, University of Delhi
ruchikamlahotra2004@yahoo.com, meghakhanna86@gmail.com

Abstract

Background: The importance of Software Change Prediction (SCP) has been emphasized by several studies. Numerous prediction models in literature claim to effectively predict change-prone classes in software products. These models help software managers in optimizing resource usage and in developing good quality, easily maintainable products.
Aim: There is an urgent need to compare and assess these numerous SCP models in order to evaluate their effectiveness. Moreover, one also needs to assess the advancements and pitfalls in the domain of SCP to guide researchers and practitioners.
Method: In order to fulfill the above stated aims, we conduct an extensive literature review of 38 primary SCP studies from January 2000 to June 2019.
Results: The review analyzes the different set of predictors, experimental settings, data analysis techniques, statistical tests and the threats involved in the studies, which develop SCP models.
Conclusion: Besides, the review also provides future guidelines to researchers in the SCP domain, some of which include exploring methods for dealing with imbalanced training data, evaluation of search-based algorithms and ensemble of algorithms for SCP amongst others.

Keywords: change-proneness, machine learning, software quality, systematic review

1. Introduction

The importance of planning and implementing change in a software is accepted universally. It is crucial for a software to reform in order to remove existing defects, to upgrade itself with the changing user requirements and technological progressions or to improve the current performance and structure [1–3]. In case a software product is unable to do so, it rapidly becomes obsolete and extinct. Thus, change management of a software product is a vital activity, which needs to be properly enforced.

Prediction of change-prone parts of a software product is an effective mechanism for software change management. Change-proneness is defined as the likelihood that a class would change across different versions of a software product [1, 4]. Since it indicates whether a specific class would require change in the forthcoming release of the software, it is generally represented by a binary variable indicating "yes" (change-prone class) or "no" (not change-prone class). Knowledge of change-prone classes aids software managers in effectively planning critical software resources such as cost, time and human effort. Sufficient allocation of these resources to change-prone classes ensures that they are carefully designed and rigorously verified [1–3]. Such activities would result in a good quality, easily maintainable and cost-effective software products.

Various studies in literature have successfully developed software quality models to predict change-prone classes of a software. These studies have explored a variety of predictor variables,

Submitted: 3 July 2019; Revised: 14 October 2019; Accepted: 14 October 2019; Available online: 12 November 2019

numerous classification algorithms and extensive software datasets for empirical validation. At such a stage, it is imperative for researchers to analyze the current state of literature and compare the capabilities of existing SCP models in literature. Such a step is important in order to summarize the existing trends in the domain and simultaneously analyze the shortcomings and future directions in the area. Thus, we conduct a systematic literature review of SCP studies from the period of January 2000 to June 2019. The review is conducted according to the guidelines specified by Kitchenham et al. [5].

Though, software change-proneness prediction studies have been earlier assessed by previous review studies [6,7], they have not been effectively scrutinized for trends specific to the software change domain. These studies have either been explored as an application of Search-Based Algorithms (SBA) to predictive modeling [6] or for only assessing the threats specific to SBA [7]. A previous study by Godara and Singh [8] assessed change-proneness prediction in Object-Oriented (OO) software. However, they only presented a survey of the studies without critically analyzing their various parameters. Also, though a previous attempt by the authors [9] evaluated SCP studies, the analysis was limited with no emphasis on the predictive capabilities of various data analysis algorithms. To the best of our knowledge, no study in the literature has comprehensively evaluated and summarized the experimental settings, predictor metrics, datasets, capabilities of data analysis algorithms, statistical tests and threats with respect to SCP studies.

On the other hand, there are various review studies which analyze defect-proneness prediction (a closely related area to change-proneness) literature. A study by Catal and Diri [10] reviewed 74 defect prediction studies to assess the metrics, data analysis algorithms and datasets used in defect prediction literature. They concluded that method-level metrics, machine learning algorithms and public datasets are the most dominant in the area. Hall et al. [11] reviewed 36 primary studies to study the context, predictors and data analysis algorithms used for defect prediction. They concluded that combination of predictor variables yields better defect prediction models and feature selection enhances the performance of the developed models. Radjenovic et al. [12] reviewed 106 primary studies and classified the metrics used in defect prediction literature as Object-Oriented (OO) metrics, traditional source code metrics and process metrics. A review study by Wahono [13] assessed 71 defect prediction studies between the period January 2000 to December 2013. The review assessed the trends and frameworks in defect prediction literature apart from datasets and data analysis algorithms. As pointed out in the review, the frameworks developed by certain defect prediction studies do not address the issue of class imbalance and noisy data. Hosseini et al. [14] reviewed 30 primary studies to assess the state of the art in cross project defect prediction. They concluded that cross-project defect prediction still requires extensive research before it yields reliable results. Certain other reviews on defect-proneness prediction includes the one conducted by Malhotra [15], Singh et al. [16] and Catal [17]. Though these reviews yield a significant contribution to defect prediction literature, there a huge gap in change prediction literature in terms of an effective and extensive review.

We investigate the following Research Questions (RQs) in the review:

- RQ1: Which predictors are useful for developing SCP models?
- RQ2: What have been the various experimental settings used in primary studies while developing SCP models?
  - RQ2.1: What are the various feature selection or dimensionality reduction techniques?
  - RQ2.2: What are the characteristics of datasets used?
  - RQ2.3: What are the various validation methods used?
  - RQ2.4: Which performance measures have been used?
- RQ3: What are the various categories of data analysis algorithms used for developing SCP models?
  - RQ3.1: Which is the most popular category of data analysis algorithm used?

  - RQ3.2: Which Machine Learning (ML) algorithms have been evaluated?
- RQ4: What is the predictive performance of ML algorithms for developing SCP models? How does the predictive capability of ML algorithms compare amongst themselves?
- RQ5: What are the various statistical tests used for validating the results of SCP models?
- RQ6: What threats to validity exist while developing SCP models?
  - RQ6.1: What are the various categories of threats which exist while developing SCP models?
  - RQ6.2: What are the steps required to mitigate the threats identified in RQ6.1?

The aim of the review is to summarize the empirical evidence reported in literature with respect to SCP. It would also help in identification of research gaps and will provide future possible guidelines to researchers and practitioners. The organization of the study includes the importance of SCP (Section 2), review procedure and the various review stages (Section 3), the review protocol (Section 4), the answers to the investigated RQ's (Section 5), the threats (Section 6) and finally the conclusions and future guidelines (Section 7).

2. Importance of SCP

There are several diverse reasons for change-proneness of a specific code segment. Some real-world examples of why a specific code segment could be prone to changes are provided below:

- A code segment may have bad structure or rigid design which is difficult to extend [18].
- A code segment might contain errors which have escaped the testing phase and now requires maintenance [19–21].
- Business requirements of an organization could change necessitating a change in source code segment [19, 20].

Therefore, SCP is critical in order to identify such change-prone code segments in the early stages of software lifecycle so that developers allocate proper manpower, cost and time to modify them [18, 21–23]. Such a step is crucial in order to keep the software operational, and ensure customer satisfaction. Even in the era of agile development, SCP is an approach to continuously monitor change-proneness so that effective product quality is maintained. Neglect of change-prone code fragments could result in poor software quality, extensive costs [20] as the cost to correct errors increases manifold as they propagate to later stages of the software product coupled with delayed delivery schedules.

Before we state the review procedure, it is important to ascertain as to how a specific segment (class/file/module) of source code is adjudged as change-prone or not change-prone in primary studies. Majority of primary studies consider a specific segment as change-prone if there is at least one insertion, removal or modification of Source Lines of Code (SLOC) in the specific code segment from current software release to its next [24–26]. We term this definition as "SLOC-based" method. In certain other primary studies, a specific code segment is considered change-prone if the number of changes it underwent from one release to the next is greater than the median value (median of number of changes in all the source code segments in the software) [19, 27, 28]. We term the definition as "median-based" method. However, few primary studies used boxplot-based partition method [18], class stability [29] or other methods to define the dichotomous change-proneness dependent variable.

As discussed above, primary studies have used various methods to define change-proneness. However, it is difficult to adjudge any one of them as best as each method has its own merits and demerits. The "SLOC-based method" may designate a segment as change-prone even if very few (even one) SLOC has been changed. This ensures that no change is missed as it is possible that critical changes to the code are performed by changing very few SLOC. It may be noted that such code segments may be candidates for corrective maintenance but not very much for preventive maintenance as only few SLOC has been modified and not much structure would have been altered. However, this method has a downside,

even trivial changes to the code would also make the code segment counted as change-prone. On the other hand, the "median-based" method ignores the code segments with very few changes. It focuses on identifying classes which requires preventive maintenance. The goal of "median-based" method is to find which classes will change more than others, not which classes will change in an absolute manner. But as discussed earlier the "median-based" method may ignore critical changes if they are performed using very few lines.

3. Review procedure

According to the guidelines advocated by Kitchenham et al. [5], a review is conducted in three fundamental stages. These stages are reportedly planning, conducting and reporting. The foremost step of the planning stage is to evaluate the necessity of the review. As already discussed, this review is important so as to evaluate, assess and summarize the empirical evidence with respect to prediction of change-prone classes in software products. It intends to provide an overview of existing literature in the domain and would scrutinize possible future directions. Once the need of the review is assessed, the planning stage involves formation of RQs. Thereafter, a review protocol is formulated. The protocol includes a detailed search strategy. The search strategy consists of the list of possible search databases one intends to scrutinize, the search string and the criteria for inclusion or omittance of the extracted studies. Apart from the search strategy, the protocol also includes the criteria for assessing the quality of the candidate studies, the procedure for collecting the relevant data from the primary studies and synthesis of the collected data. The second stage involves the actual execution of the review protocol. In this stage, all the primary studies are extracted, scrutinized and the relevant data is obtained. The final stage of the review reports the results of the investigated RQs. The RQs are answered on the basis of the data collected from the primary studies of the review.

4. Protocol for conducting the review

The following sections describe the review protocol followed which lists the search strategy used, the criteria used for selecting or omitting the extracted studies and the criteria for evaluating the quality of the collected candidate studies.

4.1. Search strategy

The search terms were designed by dividing the explored RQs into logical units. Moreover, terms were identified from paper titles, keywords and abstracts. Thereafter, all equivalent terms and synonyms were compiled using Boolean OR (||), while distinguishable search terms were aggregated using Boolean AND (&). As indicated earlier, the search period was January 2000–June 2019. The designed search-string is:

("software product" || "open source project" || "software application" || "software system" || "software quality" || "source code") & ("change" || "evolution" || "maintenance") & ("prediction" || "proneness" || "classification" || "classifier" || "empirical") & ("machine learning" || "statistical" || "search-based" || "evolutionary" || "data analysis")

The search was conducted in SCOPUS, ACM digital library, Wiley online library, IEEExplore and SpringerLink, as these are well-known search databases. We also searched the reference lists of the studies and found seven studies. In all, we identified and extracted 67 relevant studies, which were further subjected to the criteria indicated in Section 4.2.

4.2. Inclusion and omittance criteria

We use the following inclusion and omittance criteria for selecting or rejecting a study based on the RQs, after which we get 41 candidate studies.

– *Inclusion Criteria*: All studies which predict the dichotomous change-proneness attribute of a class/module or determine class stability with the aid of software metrics were included. We also include studies which reported and compared various data analysis algorithms amongst themselves for developing SCP models.

– *Omittance Criteria*: Extracted studies which predict other dependent variables such as maintenance effort, maintainability, change-count, fault-proneness, amount of changes, etc. were excluded. A related concept to change-proneness is code churn. It is defined as the volume of SLOC that is changed (inserted, modified or removed) between two versions of a software and represents the extent of change [30]. It is a continuous attribute and encapsulates the maintenance effort required by the class while it undergoes changes (bug correction, enhancements or refactoring). The current review limits itself to binary change-proneness attribute and does not include studies which assess code churn. Also, studies which predict ordinal dependent variables for change-proneness such as low, medium, high, etc. were not included as a part of the review.
We also omitted survey or review papers, PhD dissertations, short or poster papers and studies with limited or non-existent empirical analysis. A conference paper which has been published as a journal article was also omitted and only the corresponding journal article was included. Though change-proneness attribute has been explored in design pattern literature [31] and technical debt literature [32], we exclude such studies. Therefore, studies which used only design patterns or code smells for determining change-prone nature of a class /module were removed.

4.3. Quality criteria

We assess the importance of each candidate study in answering the investigated RQs. The 41 candidate studies were evaluated according to the quality criteria illustrated in Table 1. Each candidate study was given a Quality Score (QS) by combining the scores of a specific study on the basis of the 10 quality questions stated in Table 1. For each question, a candidate study can be either given a score of 0 (No), 0.5 (Partly) or 1 (Yes). Table 1 states the number of candidate studies which were allocated different scores (Yes, Partly or No). All the studies whose QS was less than 5 (50% of the total quality score) were rejected. We rejected three studies [33–35]. After quality analysis, we selected 38 studies, which we term as the primary studies of the review. The data needed to answer the RQs was extracted only from the primary studies.

Table 2 states the Primary Studies (PS) with a specific study number (SNo.) and its QS. The most popularly cited studies were PS10 and PS13.

4.4. Data extraction

The primary studies were classified according to publication year, publication venue, predictors, datasets, data analysis algorithms, performance measures, validation methods, statistical tests and threats to validity.Table A1 (Appendix A)

Table 1. Quality assessment questions

Quality questions	Yes	Partly	No
Are the objectives of the research/research questions clear and concise?	41	0	0
Are the predictor variables clearly defined and described?	27	12	2
Are the number and magnitude of datasets analyzed suitable?	30	10	1
Are the predictors effectively chosen using feature selection/dimensionality reduction techniques?	20	4	17
Are the data analysis techniques clearly defined and described?	25	9	7
Is there any comparative analysis amongst various models/techniques?	34	1	6
Are the performance measures clearly specified?	33	7	1
Did the study perform statistical hypothesis testing?	25	1	15
Does the study use appropriate validation methods?	32	1	8
Is there a description of threats to validity of research?	19	3	19

Table 2. Primary studies with quality score

SNo.	Study	QS	SNo.	Study	QS
PS1	Liu and Khoshgoftaar 2001 [36]	6.5	PS20	Elish et al. 2017 [37]	8
PS2	Khoshgoftaar et al. 2003 [38]	6.5	PS21	Kumar et al. 2017a [39]	8
PS3	Tsantalis et al. 2005 [40]	8	PS22	Kumar et al. 2017b [41]	9
PS4	Sharafat and Tahvildari 2008 [42]	5.5	PS23	Kumar et al. 2017c [43]	7
PS5	Azar 2010 [44]	6.5	PS24	Malhotra and Jangra 2017 [45]	9
PS6	Han et al. 2010 [46]	6	PS25	Malhotra and Khanna 2017a [26]	9.5
PS7	Azar and Vybihal 2011 [29]	7.5	PS26	Malhotra and Khanna 2017b [47]	9.5
PS8	Eski and Buzluca 2011 [48]	5	PS27	Yan et al. 2017 [49]	9.5
PS9	Lu et al. 2011 [24]	7	PS28	Agrawal and Singh 2018 [50]	8
PS10	Romano and Pinzger 2011 [27]	8	PS29	Catolino et al. 2018 [19]	9.5
PS11	Giger et al. 2012 [28]	8	PS30	Ge et al. 2018 [23]	6.5
PS12	Elish et al. 2013 [25]	9.5	PS31	Liu et al. 2018 [51]	7
PS13	Malhotra and Khanna 2013 [52]	9	PS32	Kaur and Mishra 2018 [53]	8
PS14	Malhotra and Bansal 2014 [54]	6	PS33	Malhotra and Khanna 2018a [55]	9.5
PS15	Malhotra and Khanna 2014 [56]	9	PS34	Malhotra and Khanna 2018b [57]	9.5
PS16	Marinescu 2014 [58]	7	PS35	Zhu et al. 2018 [18]	9.5
PS17	Elish et al. 2015 [59]	6	PS36	Catolino and Ferrucci 2019 [19]	9.5
PS18	Malhotra and Khanna 2015 [60]	8.5	PS37	Kumar et al. 2019 [22]	8
PS19	Bansal 2017 [61]	9.5	PS38	Malhotra and Khanna 2019 [21]	9.5

states the key parameters of primary studies after the data extraction step.

5. Review results

The current section states the review results and the discussions corresponding to the obtained results. We categorized the primary studies according to their publication venue and found that 37% of the 38 primary studies were conference publications, 59% of the studies were journal publications and one study each was published as a technical report and a chapter. The most popular journals were "Information and Software Technology" (11% studies) and "Journal of Software: Evolution and Process" (8% studies). "International Conference on Advances in Computing, Communication and Informatics" and "Innovations in Software Engineering Conference" were found to be the most popular conference venues with 5% of publications each. Figure 1 depicts a distribution of all the primary studies according to "publication year". According to the figure, the highest number of SCP studies were published in 2017.

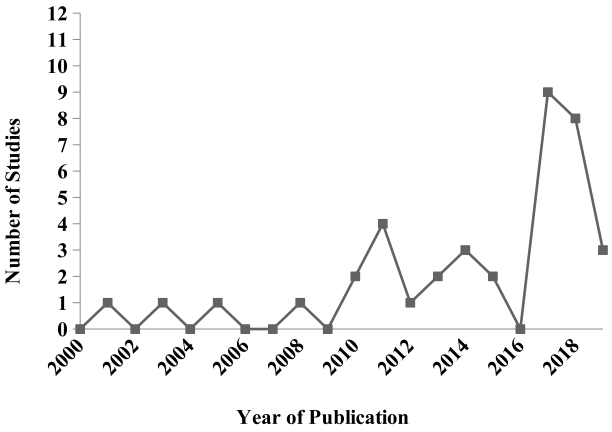

Figure 1. Publication year of primary studies

5.1. Predictors used for SCP (RQ1)

This RQ determines the various predictors which have been used for developing SCP models. An analysis of primary studies reveals that both product as well as process metrics have been used as predictors for SCP. Table 3 lists the various metrics used in primary studies for developing SCP models.

According to Table 3, product metrics especially structural metrics extracted from source code design have been widely used in SCP

Table 3. Predictors used by SCP studies

Metric category	Brief description	Study numbers	Category
Structural metrics	These metrics are generally source code design metrics, which depict the structural attributes of a class such as its inheritance, cohesiveness, size, etc. Many such metric suites have been proposed in literature such as Chidamber and Kemerer (CK) metrics suite [62], Quality Models for Object Oriented Design metrics suite [63], Lorenz and Kidd metrics suite [64], Li and Henry metrics suite [65] and many others.	PS1–PS38	Product
Network metrics	These metrics are extracted from the dependency graph of the software and identifies files which are "more central" and are more likely to change, e.g. Degree centrality, Closeness centrality, Reachability, etc.	PS11, PS35	Product
Evolution based metrics	These metrics characterize evolution history of a class, i.e. release by release history of how a class has evolved in previous versions, e.g. Birth of a Class, Frequency of changes, Change density, etc.	PS12, PS20, PS29, PS33, PS36	Process
Word vector metrics	These metrics quantify the terms used in the source code files and their names by using bag of words approach.	PS35	Product
Developer related metrics	These metrics quantify various developer related factors such as entropy of changes introduced by a developer in a given time period, number of developers employed on a specific software segment in a specific time, structural and semantic scattering of developers in a specific time period, etc., e.g. entropy of changes applied by developers in a given time period, structural scattering of developers that work on a particular class in a given time period, etc.	PS29, PS36	Process
Combination of structural and evolution based metrics	This metric suite combines structural and evolution-based metrics as they quantify two different attributes (software design and evolution history) of a class.	PS12, PS20, PS33	Product and Process
Others	Metrics such as instability and maintainability index used by PS32, probability of change based on inheritance, reference and dependency used by PS3.	PS32, PS3	Product

literature. It was found that all the studies evaluated this category of metrics. Even primary studies which proposed other possible category of metrics (evolution-based, network, developer-related, etc.) assessed and compared their proposed predictors with structural metrics as they are well established and successfully used by numerous studies. We found that the CK metrics suite was the most commonly used structural metrics suite in primary studies, which characterized various OO attributes. A similar observation was stated by Radjenovic et al. [12] while analyzing defect prediction studies. The CK metrics suite consists of Weighted Methods of a Class (WMC), Lack of Cohesion amongst Methods (LCOM), Coupling Between Objects (CBO), Response for a Class (RFC), Depth of Inheritance Tree (DIT) and Number of Children (NOC) metrics. Apart from the CK metric suite, the SLOC metric (a measure of class size) has also been frequently used in primary studies. Other product metrics used were network metrics, word vector metrics and the "others" category.

Only 16% of primary studies used process metrics. While PS12, PS20, PS29, PS33 and PS36 used evolution-based metrics which characterize the evolution history of a class, PS29

and PS36 used metrics which depict the development process complexity by quantifying developer related factors. It may be noted that PS12, PS20 and PS33 advocated the combination of both process as well as product metrics for determining change-prone nature of a class.

We also analyzed the granularity level over which these metrics were collected. Five studies (PS1, PS2, PS11, PS32 and PS35) collected file-level metrics, one study each collected structural metrics at interface level (PS10) and method level (PS4). However, all other studies analyzed class-level metrics. It may also be noted that certain studies (PS5, PS7, PS9, PS22, PS23, PS24), analyzed a large number of OO metrics with respect to different dimensions (cohesion, coupling, size and inheritance) in order to obtain generalized results.

5.2. Experimental settings for SCP (RQ2)

This RQ explores the various experimental settings, i.e. the feature selection or dimensionality reduction methods, the characteristics of datasets used for empirical validation, the validation methods and the performance measures used by SCP studies.

5.2.1. Feature selection and dimensionality reduction techniques (RQ2.1)

Primary studies use feature selection or dimensionality reduction techniques to aid the development of effective SCP models. We analyzed these studies to determine the most commonly used methods (Table 4). An analysis of 38 primary studies revealed that 58% of them used either a feature selection or a dimensionality reduction technique. According to Table 4 the most commonly used feature selection technique was Correlation-based Feature Selection (CFS). Apart from the techniques listed in Table 4, other primary studies used several other miscellaneous methods (Best-first search (PS12), Variable Importance (PS15), Rough set analysis (PS21, PS37), Information Gain (PS21, PS37), t-test (PS23), Chi-square test (PS37), Genetic Algorithm (PS23, PS37), Metric Violation Score (PS27), Wrapper Method (PS36), Consistency Feature selection (PS37), OneR feature evaluation (PS37)). Apart from feature selection, several studies performed correlation analysis to investigate whether the predictors used are correlated with change-proneness attribute (PS8, PS9, PS10, PS11, PS12, PS13, PS19, PS21, PS22, PS37).

Certain studies in literature reported specific OO metrics as effective predictors of change-prone nature of a class. These metrics were selected after application of feature selection or dimensionality reduction techniques. Since, in RQ1 we reported that the CK metrics suite and the SLOC metric are popular amongst primary studies, we state the studies which report these metrics as effective indicators of change-proneness (Table 5). According to the table, 14 studies reported metrics which characterize size attribute (SLOC and WMC) and the ones which characterize coupling attribute (CBO and RFC) as efficient indicators of change-proneness. Moreover, it may be noted that the inheritance attribute metric, DIT was only selected as an effective metric by three studies and there was no study which selected NOC (another inheritance metric) as an effective predictor of change-proneness.

5.2.2. Dataset characteristics (RQ2.2)

In order to perform empirical validation, primary studies have used a number of datasets. This question explores the characteristics of these datasets which includes their nature (public/private), size, percentage of change and other attributes.

Software datasets used by primary studies can be broadly categorized into public/open–source datasets or private/commercial datasets. We categorized the datasets in SCP studies and found that only 5% of these studies used commercial/private datasets. All other SCP studies used open–source datasets, which are publicly available. This trend was observed as commercial datasets are difficult to obtain and is similar to the one observed by Catal and Diri [10] while

Table 4. Feature selection/dimensionality reduction techniques

Feature selection/dimensionality reduction	Study numbers
Correlation-based Feature Selection (CFS)	PS13, PS18, PS19, PS24, PS25, PS26, PS30, PS34, PS35, PS37, PS38
Univariate Analysis	PS21, PS22, PS24, PS28, PS37
Principal Component Analysis (PCA)	PS12, PS20, PS21, PS37
Gain Ratio	PS21, PS29, PS37
Multivariate Regression with forward and backward selection	PS3, PS13, PS22

Table 5. OO metrics selected for SCP in primary studies

Metric Acronym	OO Attribute	Study Numbers
SLOC	Size	PS4, PS8, PS15, PS18, PS19, PS21, PS24, PS25, PS26, PS28, PS33, PS34, PS37, PS38
WMC	Size	PS8, PS15, PS18, PS20, PS22, PS24, PS25, PS28, PS33, PS34
CBO	Coupling	PS8, PS18, PS20, PS21, PS22, PS24, PS25, PS28, PS33, PS37, PS38
RFC	Coupling	PS8, PS13, PS15, PS19, PS20, PS21, PS28, PS37
LCOM	Cohesion	PS15, PS20, PS21, PS28, PS37
DIT	Inheritance	PS20, PS28, PS37

reviewing defect prediction literature. Therefore, researchers tend to validate their results on datasets that are open–source and easily available in software repositories.

We also investigated the language used to develop the datasets, which are used for empirical validation for SCP. Only four studies (PS1, PS2, PS15, PS18) used datasets developed using the C++ language. It may be noted that all other studies used datasets developed in Java language.

The datasets used in primary studies for SCP are of varying sizes and with different percentage of change-prone classes. For each study, we analyzed the number of datasets used and the minimum and maximum size of datasets in terms of number of data points, i.e. classes (Table 6). We also state the minimum and maximum percentage of change in the datasets used by these studies (Table 6). It was noted that certain datasets were used by more than one primary study. The name of such datasets and the studies which use them are listed in Table A2 (Appendix A).

It is also important to evaluate whether the datasets used for developing models are imbalanced in nature. A dataset is said to be imbalanced if it has a disproportionate number of change-prone and not change-prone classes. We state the number of datasets which were found to be imbalanced for a specific study in Table 6. As it is more important to determine the change-prone classes correctly, one should have sufficient number of change-prone classes in a dataset for effectively training the model. We term a dataset as imbalanced if it has less than 40% of change-prone classes. Studies from which relevant information could not be extracted are not shown in the table. According to the information shown in Table 6, the size of datasets used in primary studies for SCP varies from 18–3,150 data points. Therefore, these studies have analyzed small sized, moderately sized and large-sized datasets for developing SCP models. It may also be noted that the percentage of change found in these datasets varies from 1–97%. However, in a majority of the studies 25–100% of datasets analyzed were imbalanced in nature. Only few studies (PS26, PS30, PS35, PS36) addressed the issue of learning from imbalanced training data in SCP literature. It is mandatory for researchers to take active steps to develop effective prediction models from imbalanced datasets in order to develop reliable and unbiased models. On the other hand, it should be noted that though having an imbalanced dataset is an issue while training the model, it is good

Table 6. Study-wise details of datasets

SNo.	Number of data points		Percentage of change		Number of datasets Imbalanced datasets (%)
	Minimum	Maximum	Minimum	Maximum	
PS1	–	1,211	–	24%	1 (100%)
PS2	–	1,211	–	24%	1 (100%)
PS3	58	169	25%	50%	2 (50%)
PS4	–	58	–	25%	1 (100%)
PS5	18	2,737	–	–	15 (–)
PS6	44	62	–	–	1 (–)
PS7	18	958	–	–	8 (–)
PS8	38	693	–	–	3 (–)
PS9	38	2,845	–	–	102 (–)
PS10	25	165	–	–	10 (–)
PS11	98	788	–	–	2 (–)
PS12	36	170	4%	91%	20 (50%)
PS13	254	657	10%	52%	3 (67%)
PS14	607	2,786	1%	97%	12 (25%)
PS15	108	510	45%	66%	6 (0%)
PS16	–	–	–	–	18 (–)
PS17	36	60	–	–	2 (0%)
PS18	108	510	45%	66%	3 (0%)
PS19	685	756	24%	33%	2 (100%)
PS20	36	170	4%	78%	13 (38%)
PS21	–	–	–	–	1 (–)
PS22	1,507	1,524	7%	16%	5 (100%)
PS23	83	1,943	30%	68%	10 (30%)
PS24	348	434	4%	30%	2 (100%)
PS25	72	374	19%	63%	6 (50%)
PS26	72	350	6%	37%	6 (100%)
PS27	53	3,150	8%	94%	14 (57%)
PS28	608	1,496	9%	46%	5 (80%)
PS29	–	–	19%	35%	20 (100%)
PS30	341	1,505	3%	83%	20 (65%)
PS31	53	3,150	2%	93%	14 (64%)
PS32	86	130	14%	59%	4 (75%)
PS33	210	375	4%	52%	9 (67%)
PS34	78	1,404	29%	88%	10 (40%)
PS35	272	1,705	8%	17%	8 (100%)
PS36	121	2,2,46	22%	37%	33 (100%)
PS37	83	1,943	30%	68%	10 (30%)
PS38	222	1,101	16%	62%	15 (53%)

Note: "–" indicates the corresponding information was not found in the study.

that only few classes are change-prone. Therefore, only these few classes require constant monitoring and most of the maintenance resources can be focused on these classes. In case majority of the classes are change-prone, software practitioners might face a tough time managing constraint resources during maintenance and testing.

5.2.3. Validation methods (RQ2.3)

Studies in literature have used various validation methods for developing SCP models which can be broadly categorized into within-project methods and cross-project methods. Within-project validation models use training and testing data of the same software project. The training data used by

the model is obtained from the previous versions of the same project and is validated on the later versions. On the contrary, in cross- project validation, the prediction model is trained using data from one project (say Project A) and is validated on another project (Project B). Cross-project validation is useful in case historical data of the same software project is not available. Figure 2 depicts the most commonly used validation methods in SCP studies. An analysis of the figure reveals that majority of studies developed models with within-project approach. It can be performed using either hold-out validation, K-fold cross validation or Leave-one-out Cross Validation (LOOCV), which are described below:

- LOOCV: For a dataset having N instances, this method requires N iterations. In each iteration, all data points except one are used as training instances. The remaining data point is used for validation. It is ensured that all data points are used at least once for validating the developed model. Only one study (PS17) used LOOCV.
- K-fold Cross Validation: The whole dataset is randomly split into K parts, which are nearly equal in size. Thereafter, K iterations are performed. In each iteration, only one partition is excluded for validation, while all others are used for training the model. As in LOOCV, each partition is used for validation at least once. The most frequently used value for K is 10. Only one primary study each used $K = 20$ (PS22) and $K = 5$ (PS37). It is the most popular method for validating SCP models.
- Hold-out Validation: The available data points are randomly split into testing and training sets using a specific ratio. One of the most common ratio used for partitioning is 75:25. In such a case, 75% of data points are used while training and the remaining 25% of data points are used while validation. However, the method has high variability due to random division of training and test sets. The points which make the training and test sets may affect the performance of the developed model. Only four studies used hold-out validation.

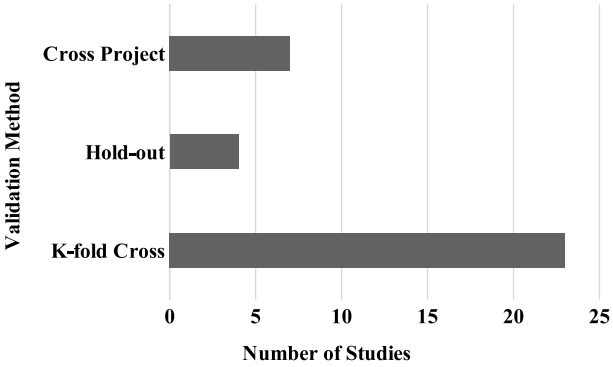

Figure 2. Validation methods in primary studies

Apart from within-project validation, cross-project validation was used by seven SCP studies (PS14, PS18, PS23, PS24, PS30, PS31, PS38). Also, inter-version validation, where different releases of the same dataset are used for training and validation was used by two studies (PS14, PS26). We found that k-fold cross validation is the most popular validation method as it provides the mean results obtained in various partitions, thereby reducing variability. As a result, the data is insensitive to the created partitions as in the case of hold-out validation. PS29 considered the time dimension for validating the developed model. They used a three month sliding window to train and test SCP models as the developers metrics used by the study encapsulate developer dynamics in a given time period.

5.2.4. Performance measures (RQ2.4)

The developed SCP models in primary studies are assessed using various performance measures. This RQ investigates the most commonly used performance measures, depicted in Figure 3. The definitions of these measures are stated as follows:

- Accuracy: It depicts the percentage of correctly predicted classes (change-prone and not change-prone category).
- Recall: It is an estimate of the percentage of correctly predicted change-prone classes amongst the total number of actual change-prone classes. It is also commonly referred to as Sensitivity. A complementary measure of Recall is specificity. Specificity represents the percentage of correctly predicted

not change-prone classes amongst the total number of actual not change-prone classes.
- Precision: It depicts the percentage of correctly predicted change-prone classes amongst the total number of predicted change-prone classes.
- F-measure: It is computed as the harmonic mean of recall and precision.
- Area Under Receiver Operating Characteristic Curve (AUC): It is a plot of recall and specificity. Recall is depicted on the y-axis, while a value of 1 specificity is depicted on the x-axis. The area under the depicted plot gives an estimate of the model's performance.

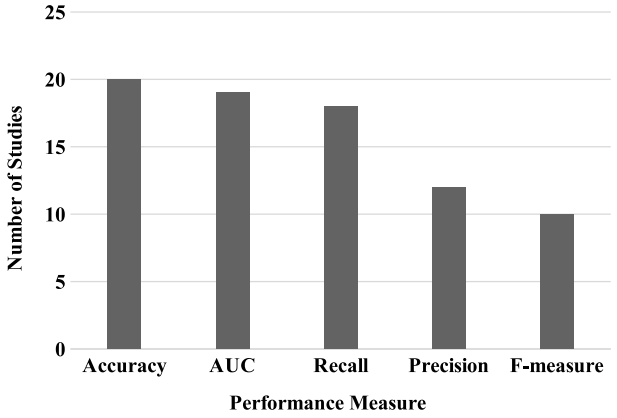

Figure 3. Commonly used performance measures in primary studies

It may be noted that a model which attains higher value for all the discussed performance measures is desirable. According to Figure 3, the most commonly used measure is accuracy. However, in case of imbalanced datasets, accuracy is not an appropriate measure [66–68]. Even if the percentage of correctly predicted change-prone classes are very few, accuracy values can be high as the performance measure is not sensitive to class distributions. On the contrary, the AUC measure is effective as it takes into account both recall and 1 specificity. Researchers should use an appropriate performance measure to yield unbiased results. Selection of an appropriate performance measure is vital to strengthen the conclusion validity of the study. Apart from the measures shown in Figure 3, there were several other performance measures (Type I error, Type II error, Overall misclassification error, False positive ratio, False negative ratio, Specificity, Probability of False Alarm (PF), Goodness of fit, J-index, G-measure, G-mean, Change cost, cost ratio, Balance, Mathews Correlation Coefficient), which were used by only few studies.

5.3. Data analysis algorithms used for SCP (RQ3)

Prediction models require the aid of data analysis algorithms, which can be broadly categorized into statistical or ML. Statistical algorithms include regression techniques such as binary Logistic Regression (LR), polynomial regression or Linear Discriminant Analysis (LDA). ML algorithms include various categories such as Decision Trees (DT), Bayesian algorithms, Artificial Neural Networks (ANN), ensembles, Search-Based Algorithms (SBA), etc. We first investigate the most popular category of algorithms for developing SCP models.

5.3.1. Popular category of data analysis algorithms (RQ3.1)

Certain primary studies used only a specific category of algorithm, i.e. only statistical or only ML, while certain others used more than one category. Figure 4 depicts the number of primary studies using the various categories of algorithms. A new category of algorithms, i.e. ensembles algorithms were used by certain studies (PS17, PS20, PS21, PS23, PS34, PS37, PS38), which were ensemble of several base learning algorithms. For instance, PS17 used an ensemble of Multilayer Perceptron (MLP), Support Vector Machine (SVM), Genetic Programming (GP), Logistic Regression (LR) and k-means techniques which were aggregated using majority voting. According to Figure 4, ML algorithms are the most popular category, followed by the statistical algorithms. The disadvantage of statistical algorithms over ML ones is that the models developed using statistical techniques are not easily interpretable [69]. Another disadvantage of statistical models is that they are highly reliant on data distribution and are based on assumptions which may not be fulfilled by the software product data whose change-prone-

ness is to be predicted [69]. Out of the 38 studies, three studies did not use any specific algorithm but predicted classes using a certain set of equations (PS4), by using a combined rank list (PS8) or by using random effect meta-analysis model (PS9).

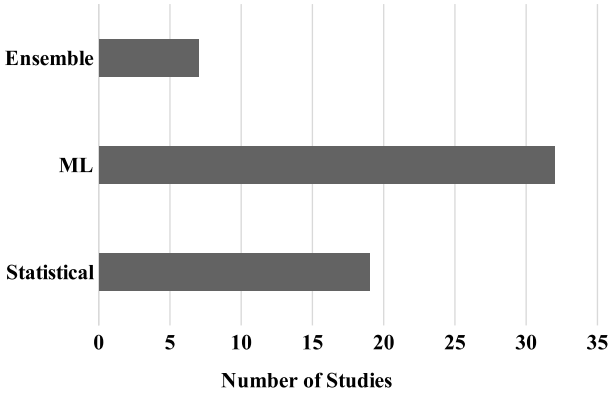

Figure 4. Categories of techniques

5.3.2. ML algorithms used for SCP (RQ3.2)

The ML algorithms can be further divided into several categories in accordance with Malhotra [15]. Table 7 states the various sub-categories of ML techniques which are used by SCP studies. These sub-categories are Decision Trees (DT), Bayesian algorithms, SVM, ML Ensemble, ANN and SBA. Other remaining algorithms were grouped into a miscellaneous category.

We further analyzed the percentage of primary studies which used a specific category of ML algorithms amongst the primary studies which used an ML algorithm for SCP (Figure 5). It was noted that ANN is the most popular category of ML algorithms which are used by 53% of studies. ANN are capable of modeling complex non-linear relationships and are adaptive in nature making them suitable for change prediction tasks. The next popular category of techniques were SBA, used by 41% of studies. It is a subclass of ML algorithms, which have recently gained popularity. SBA are self-optimizing techniques, which are capable of dealing with noisy and imprecise data. ML ensemble algorithms, which form several classification models using variants of training set and use voting scheme to combine these models are also a popular category of techniques used by 41% of studies.

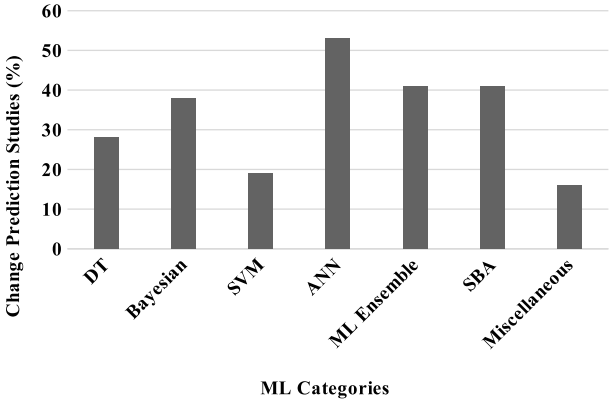

Figure 5. Sub-categories of ML algorithms

5.4. Predictive performance of ML algorithms for SCP (RQ4)

The various ML algorithms investigated in the primary studies for developing SCP models should be assessed so as to ascertain their effectiveness.

5.4.1. Predictive capability of ML algorithms

In order to assess the capability of ML algorithms, we state the values of popular performance measures of the developed SCP models. However, we need to generalize our results and avoid any bias. This was done by reporting the results of models developed by those algorithms which were validated by using at least three different datasets and by at least two of the primary studies. This would forbid an algorithm which exhibits exceptional performance only in a certain study or only by using certain datasets to be declared as a superior one. We analyze the statistics in accordance with the datasets. However, it may be the case that the performance of a technique varies due to its application on a specific dataset. Thus, we remove outlier values in accordance with the investigated datasets. We also report the median values to reduce biased results. The following rules were observed while extracting various statistics [6, 15]. The rules are chosen so that optimum values attained by a technique may be reported. This is

Table 7. Sub-categories of ML algorithms

Sub-category	ML algorithms
Decision Tree (DT)	C4.5, J48, Classification And Regression Tree (CART)
Bayesian	Naive Bayes (NB), Bayesian Network (BN)
SVM	SVM, Linear Kernel SVM, Sigmoid Kernel SVM, Polynomial Kernel SVM, Least-Square SVM (Linear, Polynomial and RBF kernels)
Artificial Neural Networks (ANN)	MLP, MLP with Conjugate Learning (MLP-CG), Radial Basis Function (RBF), Group Method of Data Handling (GMDH), Extreme ML (Linear, Polynomial and RBF kernels)
ML Ensemble	Random Forests (RF), Bagging (BG), Adaptive Boosting (AB), LogitBoost (LB)
Search Based Algorithms (SBA)	Ant Colony Optimization (ACO), Constricted Particle Swarm Optimization (CPSO), Decision Tree-GP, Decision Tree- GA (DT-GA), GP, Genetic Expression Programming (GEP), Hierarchical Decision Rules (HIDER), Memetic Pittsburgh Learning Classifier System (MPLCS), Supervised Classifier System (SUCS), X Classifier System (XCS), Genetic Algorithm with Adaptive Discretization Intervals (GA-ADI), Fuzzy Learning based on Genetic Programming Grammar Operators and Simulated Annealing (GFS-SP), Fuzzy Learning based on Genetic Programming (GFS-GP), Genetic Fuzzy System AdaBoost (GFS-AB), Genetic Fuzzy System- LogitBoost (GFS-LB), Genetic Fuzzy System MaxLogitBoost (GFS-MLB), Genetic Algorithm with Neural Networks (GANN), Neural Net Evolutionary Programming (NNEP), Particle Swarm Optimization- Linear Discriminant Analysis (PSO-LDA); Structural Learning Algorithm in a Vague Environment with Feature Selection (SLAVE)
Miscellaneous	K-Nearest Neighbor (K-NN), k-means, KStar, Non-Nested Generalized Exemplars (NNGE), PART, Decision Table

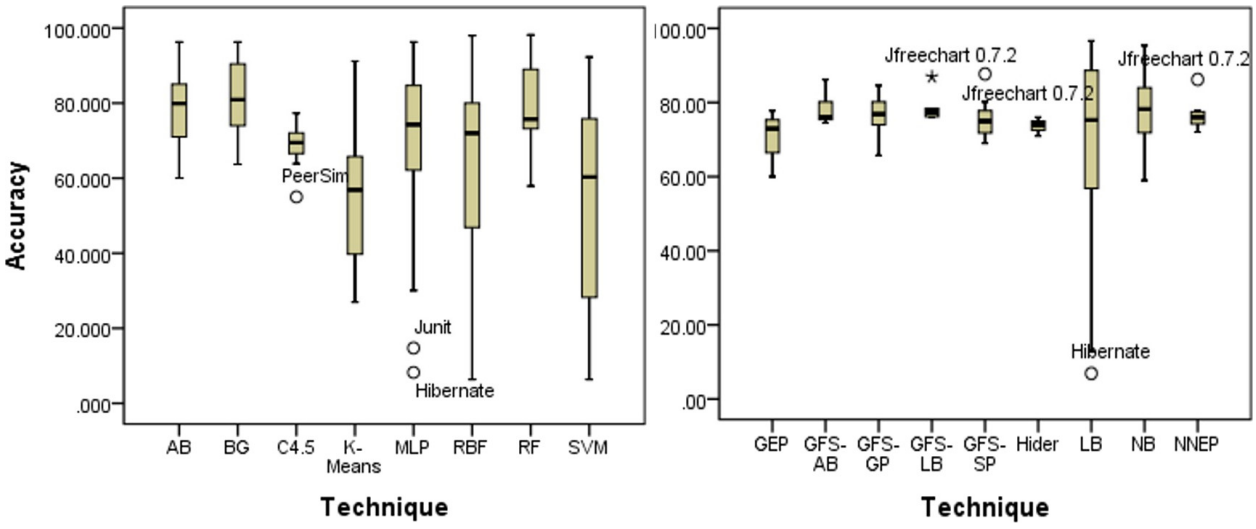

Figure 6. Dataset-wise accuracy outliers of ML algorithms

important as the performance of an ML algorithm is dependent on its internal parameter settings.

– If a specific study develops models on the same dataset more than once with different experimental settings, we choose the best performance measure values obtained by the technique.
– In case there is more than one study which develops models using the same dataset and the same technique, we use the best of per-

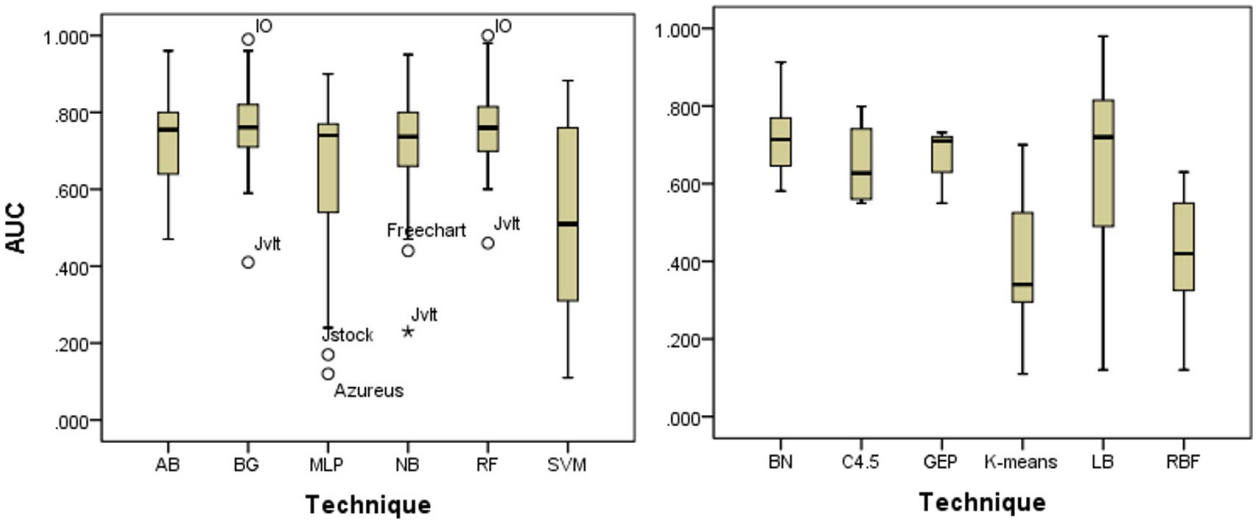

Figure 7. Dataset-wise AUC outliers of ML algorithms

formance measure value reported in all the studies.

According to Section 5.2.4, the most commonly used performance measures by SCP studies are accuracy and AUC. Figure 6 depicts the dataset-wise outliers of different ML algorithms with respect to accuracy measure. According to the figure, the Hibernate dataset was an outlier for both MLP and LB algorithms, showing lower accuracy values than all other investigated datasets. JFreechart 0.7.2 is exhibited as an outlier for GFS-LB, GFS-SP and NNEP algorithms. Figure 7 depicts the dataset-wise outliers of different ML algorithms with respect to AUC measure. According to the figure, NB, BG, RF and MLP algorithms were found to have outliers with lower AUC values except the IO dataset which had higher AUC values for BG and RF.

As discussed before, a good change prediction model exhibits higher values of accuracy and AUC measures. Table 8, 9 presents the comparative results of the change prediction models developed using ML algorithms for the accuracy and AUC measure respectively. The tables report the statistics values along with the count of datasets from which the statistics were extracted, after removing the outliers.

As depicted in Table 8, the majority of ML techniques (except k-means and SVM) depicted mean accuracy values in the range 60–80%. The BG technique depicted the best mean accuracy value of 81.72%. With respect to median accuracy values, the best median value was depicted by the BG technique. As depicted in Table 9, with respect to AUC, the majority of ML techniques (except k-means, RBF and SVM) depicted a mean AUC value in the range 0.65–0.78. Both the BG and RF techniques depicted the highest mean AUC value of 0.77. The best median AUC values were depicted by AB, RF and BG techniques of 0.76 each. These results indicate effectiveness of ML techniques in determining change-prone nature of classes/modules.

It may be noted that the BG, RF and AB techniques belong to the ensemble category of ML algorithms. Therefore, their effective predictive capability is a result of aggregation of results of several base models. This leads to stable and robust models. It may also be noted that the SBA (GFS-AB, GFS-GP, GFS-LB, GFS-SP, NNEP, HIDER) also exhibit good accuracy results. SBA are effective in optimizing the accuracy of the developed SCP models. This category of ML algorithms needs to be further explored as their results are promising. The statistics reported in Tables 8, 9 reveal that the use of ML algorithms for change-proneness prediction tasks should be encouraged as they yield effective results.

Furthermore, we also conducted a meta-analysis of the review studies, which reported the AUC performance measure. This was done in

Table 8. Accuracy results of ML algorithms for SCP models

ML Algorithm	Count	Minimum	Maximum	Mean	Median	S.D.
AB	17	60.00	96.30	78.92	79.90	10.78
BG	16	63.71	96.30	81.72	80.95	9.25
C4.5	11	63.86	77.33	69.95	69.99	3.55
GEP	3	60.00	77.78	70.24	72.94	7.50
GFS-AB	6	74.50	86.20	78.05	76.00	4.06
GFS-GP	6	65.70	84.60	76.38	76.90	5.91
GFS-LB	5	76.00	78.40	77.10	77.00	0.93
GFS-SP	6	69.00	80.20	73.87	73.50	3.56
HIDER	3	71.00	76.00	73.67	74.00	2.05
k-means	16	26.99	91.17	54.51	56.91	18.73
LB	28	12.88	96.61	71.13	78.10	22.75
MLP	38	30.10	96.30	73.00	75.72	16.56
NB	15	59.00	95.38	77.98	78.19	10.22
NNEP	6	72.00	77.90	75.25	75.25	1.98
RF	25	57.91	98.18	79.32	75.70	10.58
RBF	27	6.38	98.00	63.32	72.00	24.69
SVM	17	6.38	92.29	53.15	60.33	28.20

S.D. indicates Standard Deviation.

Table 9. AUC results of ML algorithms for SCP models

ML Algorithm	Count	Minimum	Maximum	Mean	Median	S.D.
AB	25	0.47	0.96	0.74	0.76	0.18
BG	32	0.59	0.96	0.77	0.76	0.11
BN	21	0.58	0.91	0.72	0.71	0.09
C4.5	4	0.55	0.80	0.65	0.63	0.09
GEP	3	0.55	0.73	0.66	0.71	0.08
k-means	16	0.11	0.70	0.40	0.34	0.17
LB	31	0.12	0.98	0.65	0.72	0.23
MLP	47	0.24	0.90	0.69	0.74	0.16
NB	32	0.47	0.95	0.74	0.75	0.09
RBF	16	0.12	0.63	0.41	0.42	0.16
RF	31	0.60	0.98	0.77	0.76	0.10
SVM	27	0.11	0.88	0.53	0.51	0.24

S.D. indicates Standard Deviation.

order to evaluate the performance of ML methods in the domain of SCP. Figure 8 reports the forest plot of primary studies with the summary performance measure statistic per study. The weights to the primary study were allocated on the basis of standard error, i.e. higher standard error indicated lower study weight [70]. The confidence interval is computed at 95%. We assessed the random effects model as the studies were heterogeneous in terms of datasets, ML methods and their performance. The overall effect in the figure indicates that ML methods are effective for SCP.

5.4.2. Comparative performance of ML algorithms

We investigate the comparative performance of various ML algorithms with each other and with traditional statistical algorithms used for developing SCP models. The explored hypothesis is stated as follows:

Null Hypothesis (H0): There is no statistical difference amongst the performance of different ML algorithms when compared with each other and with the statistical technique (LR), while developing SCP models.

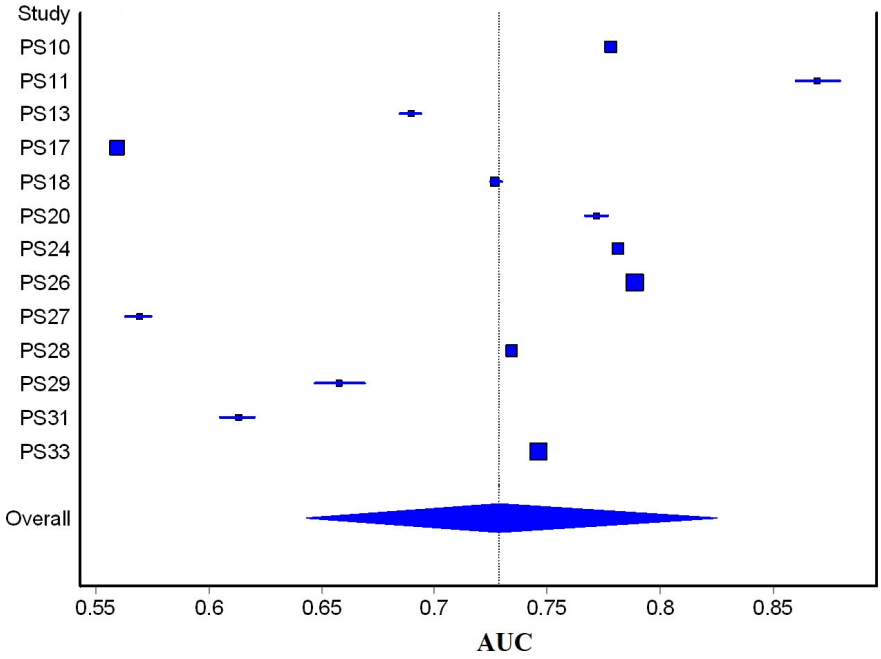

Figure 8. Forest plot

Alternate Hypothesis (H1): There is significant difference amongst the performance of different ML algorithms when compared with each other and with the statistical technique (LR), while developing SCP models.

The comparative performance was evaluated dataset-wise and the rules were similar to the ones followed in RQ4. Furthermore, Wilcoxon signed rank test was performed at a significance level of 0.05 for statistical evaluation of the comparative results. We compared the performance of 17 ML algorithms namely, MLP, BG, AB, RF, RBF, SVM, C4.5, k-means, LB, HIDER, GFS-AB, GFS-GP, GFS-LB, GFS-SP, NNEP, NB and BN amongst each other and with LR. LR is chosen as it is the most common statistical algorithm used in SCP literature. The other ML algorithms were chosen as we could extract sufficient data from primary studies for their comparison.

Tables 10, 11 report the results of the Wilcoxon signed rank test when different algorithms are compared with one another and with the LR algorithm according to accuracy and AUC performance measures respectively. The symbols used in the table represent whether the performance of the technique stated in the row is significantly superior (BT*), significantly inferior (WR*), superior but not significantly (BT), inferior but not significantly (WR) or equivalent (=), when compared with the technique stated in the column. We consider the two compared techniques as equivalent when the pairwise comparison amongst the techniques yield equal number of negative and positive ranks in Wilcoxon signed rank test. According to Table 10, the MLP technique shows significantly better performance than LR, C4.5 and NB techniques in terms of accuracy measure. The performance of MLP technique is worse but not significantly, when compared with the AB technique. MLP's accuracy performance is better than RF, RBF, SVM, k-means and LB techniques but not significantly. The Wilcoxon test results according to AUC measure depicted in Table 11 show that the RF, LB, BN and NB algorithms showed significantly better AUC performance than various other algorithms. The MLP algorithm also depicts better AUC values than five other compared algorithms, but not significantly.

It may be noted from the results of Table 10, 11 that three ML algorithms depicted better accuracy results than the statistical algorithm, LR. However, four algorithms (SVM, GFS-SP, NNEP

Table 10. Comparison of ML algorithms based on accuracy measure (Wilcoxon test results)

Algo.	MLP	BG	AB	RF	LR	RBF	SVM	C4.5	KM	LB	HIDER	GAB	GGP	GLB	GSP	NNEP	NB
MLP	–	WR*	WR	BT	BT*	BT	BT	BT*	BT	BT	ND	ND	ND	ND	ND	ND	BT*
BG	BT*	–	BT	WR	BT	ND	ND	ND	ND	WR	ND	ND	ND	ND	ND	ND	BT
AB	BT	WR	–	WR	WR	ND	ND	ND	ND	WR	WR	ND	ND	ND	WR	WR	BT*
RF	WR	BT	BT	–	BT*	WR	ND	BT*	ND	BT	ND	ND	ND	ND	ND	ND	BT*
RBF	WR	ND	ND	BT	ND	–	BT	BT*	WR	WR	ND	ND	ND	ND	ND	ND	ND
SVM	WR	ND	ND	ND	WR	ND	–	ND	WR	WR*	ND	ND	ND	ND	ND	ND	ND
C4.5	WR*	ND	ND	WR*	ND	WR*	ND	–	ND	ND	ND	ND	ND	ND	ND	ND	ND
k-means	WR	ND	ND	ND	ND	BT	BT	ND	–	WR	ND	ND	ND	ND	ND	ND	ND
LB	WR	BT	BT	WR	ND	BT	BT*	ND	BT	–	ND	ND	ND	ND	ND	ND	BT*
HIDER	ND	ND	BT	ND	ND	ND	ND	ND	ND	ND	–	ND	ND	ND	BT	WR	ND
GFS-AB	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	–	BT	WR	BT	BT	ND
GFS-GP	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	WR	–	WR	WR	WR	ND
GFS-LB	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	BT	BT	–	BT	BT	ND
GFS-SP	ND	ND	BT	ND	WR	ND	ND	ND	ND	ND	WR	WR	BT	WR	–	WR	ND
NNEP	ND	ND	BT	ND	WR	ND	ND	ND	ND	ND	BT	WR	BT	WR	BT	–	ND
NB	WR*	WR	WR*	WR*	ND	ND	ND	ND	ND	WR*	ND	ND	ND	ND	ND	ND	–

GAB: GFS-AB; GSP: GFS-SP; GLB:GFS-LB; GGP:GFS-GP; KM: k-means; "BT*": Significantly better results; "BT": Better but insignificant;
"WR*" indicates significantly worse results and "WR" means worse but not significant results;
"ND" indicates requisite comparison data could not be extracted; "–" indicates the techniques cannot be compared with itself.

Table 11. Comparison of ML algorithms based on AUC measure (Wilcoxon test results)

Technique	MLP	NB	SVM	BG	AB	RF	LR	LB	RBF	C4.5	k-means	BN
MLP	–	BT	BT	WR*	WR	WR*	WR*	WR*	BT	BT	BT	WR*
NB	WR	–	BT	BT*	BT*	WR*	BT*	BT*	ND	ND	ND	BT
SVM	WR	WR	–	WR	WR	WR	WR*	WR*	WR	ND	WR	WR*
BG	BT*	WR*	BT	–	BT*	WR	BT	BT	BT	ND	ND	BT
AB	BT	WR*	BT	WR*	–	WR	BT	WR	BT	ND	ND	WR
RF	BT*	BT*	BT	BT	BT	–	BT	BT	BT	ND	ND	WR
LB	BT*	WR*	BT*	WR	BT	WR	WR	–	BT	ND	BT	WR*
RBF	WR	ND	BT	WR	WR	WR	ND	WR	–	ND	WR	WR*
C4.5	WR	ND	ND	ND	ND	ND	ND	ND	ND	–	ND	BT
k-means	WR	ND	BT	ND	ND	ND	ND	WR	BT	ND	–	WR*
BN	BT*	WR	BT*	WR	BT	BT	BT	BT*	BT*	ND	BT*	–

Symbols same as Table 11

and AB) showed worse accuracy results than LR. With respect to AUC, five ML techniques were found better than LR and three were found worse than LR. This indicates effective performance of ML algorithms when compared to that of the LR algorithm. However, more studies need to be conducted for an extensive comparison of various ML algorithms with that of LR. Also, as a number of columns in Tables 10, 11 have the value "ND", where sufficient data was not found to compare the predictive ability of ML algorithms. Therefore, more studies are needed which perform extensive comparison of different ML algorithms with each other based on different performance measures. However, on the basis of current analysis we reject the null hypothesis H0.

5.5. Statistical tests used by SCP studies (RQ5)

Statistical verification of a study's results is important in order to yield reliable conclusions. 66% of primary studies used statistical tests for validating their results. These tests can be broadly categorized as parametric tests or non-parametric tests.

Twenty-five primary studies which predicted change-prone nature of a class/module statistically validated their results. Out of these 25 studies, 88% of studies used non-parametric tests, while the others used parametric tests. This trend was observed as parametric tests require stringent assumptions which should be fulfilled before their application. In order to verify the assumptions of normal tests, we require complete information about the population distribution. Though these characteristics make normal tests powerful but they are harder to apply when compared with non-parametric tests. Non-parametric tests are easy to understand and use. Thus, they are favored by the research community. Figure 9 states the number of studies using the most commonly used statistical tests. These tests were the Wilcoxon signed rank test, Friedman test, *t*-test, Scott–Knott test and Cliff's test used by 15, 9, 2, 2 and 2 studies respectively. The most popular test was Wilcoxon signed rank test. The popularity of Wilcoxon test is due to its non-parametric nature. Moreover, the test can be used individually for pairwise comparisons or as a post-hoc test after the application of Friedman test [6]. Certain other tests (ANOVA, Mann–Whitney, Nemenyi, Proportion) were used by one study each.

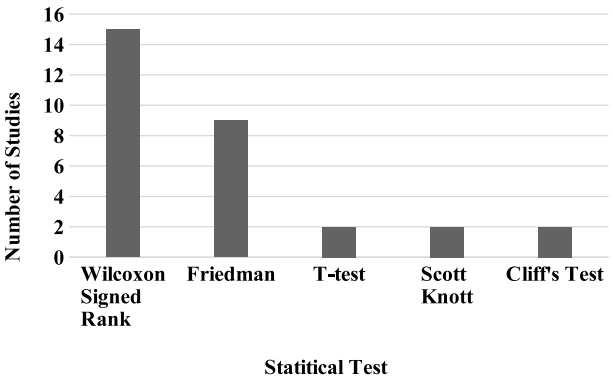

Figure 9. Statistical tests used in primary studies

5.6. Threats to validity in SCP studies (RQ6)

This RQ extracts and analyzes the threats to empirical studies which develop SCP models. It is essential for a researcher to scrutinize all probable threats in the early phases of an experiment so that the obtained results are valid and reliable. This would ensure proper experimental design so that majority of identified threats can be mitigated. Furthermore, one should mention the probable threats so that the readers are aware of the limitations. We extracted the threats from the primary studies of the review, which have a separate section for "Threats to validity" or "Limitations".

5.6.1. Categories of threats (RQ6.1)

The probable threats to SCP studies are categorized into conclusion, internal, construct and external threats.

Table 12 states the various threats corresponding to each category along with the studies which state them. It may be noted that we state only those threats which are mentioned in at least two or more primary studies. Threats specific to a study's experimental design are omitted to

Table 12. Threats to validity in SCP studies

Threat No.	Category	Threat Description (Study Numbers)
T1	Conclusion	Absence of appropriate statistical tests for validating study's results. (PS10, PS16, PS25, PS26, PS29, PS33, PS34, PS36, PS38).
T2	Conclusion	Absence of multiple and stable performance measures. (PS25, PS26, PS29, PS33, PS36, PS38).
T3	Conclusion	Not accounting for validation bias by using inappropriate validation method, (PS25, PS26, PS29, PS33, PS34, PS38).
T4	Internal	Omittance of significant variables that act as predictors or may affect the predictors, (PS3, PS11, PS31).
T5	Internal	Inability to address the confounding effect of other variables such as class size or other factors (such as developer experience, application domain, etc.) on the relationship between dependent and independent variables, (PS9, PS12, PS27, PS33).
T6	Internal	Does not account for the "causal effect" of the predictors on the target variable, (PS11, PS12, PS13, PS19, PS24, PS25, PS26, PS32).
T7	Internal	Does not account for different rules or thresholds for computing the dependent and the independent variables, (PS9, PS10, PS24, PS27, PS29).
T8	Construct	The type of change, i.e. whether it is corrective, adaptive, perfective or preventive is not taken into account, (PS9, PS12, PS13, PS19, PS24, PS29, PS36, PS38).
T9	Construct	OO metrics may not be accurate representatives of the OO concepts they propose to measure, (PS9, PS12, PS19, PS24, PS25, PS26, PS33, PS34, PS38).
T10	Construct	Independent variables (OO metrics) and dependent variable may not be correctly collected, (PS9, PS11, PS12, PS13, PS16, PS19, PS21, PS24, PS25, PS26, PS32, PS34).
T11	Construct	There may be possible imprecisions in computation of change-proneness attribute, (PS29, PS32, PS36, PS38).
T12	Construct	The severity of change and the effort spent by software practitioners in changing code fragment is not taken into account while computing change-proneness, (PS29, PS21, PS36).
T13	Construct	Absence of data pre-processing to eliminate noisy data or feature selection for choosing effective feature sets, (PS11, PS26, PS29, PS33, PS36).
T14	External	Obtained results may be specific to a certain domain, i.e. all validated datasets belonging to the same domain, (PS3, PS10, PS11, PS12, PS19, PS21, PS25, PS26, PS29, PS34, PS35, PS36, PS38).
T15	External	Obtained results may not be validated on datasets of appropriate size or appropriate number of datasets, (PS10, PS12 PS13, PS16, PS19, PS22, PS24, PS26, PS27, PS29, PS31 PS32, PS34, PS35, PS36, PS38).
T16	External	Obtained results may not be easily replicated, (PS10, PS16, PS25, PS26, PS33, PS34, PS38).
T17	External	Obtained results may not be validated on industrial datasets, (PS16, PS22, PS31).
T18	External	Obtained results may not be validated on datasets developed using different programming languages or programming paradigms, (PS13, PS16, PS21, PS24, PS25, PS26, PS27, PS29, PS31, PS32, PS33, PS34, PS36, PS38).

yield unbiased results. As stated in Table 12, we found 3 conclusion validity threats, 4 threats to internal validity, 6 construct validity threats and 5 external validity threats. It may be noted that T16 was also referred to as "Reliability threat" in two studies.

5.6.2. Mitigation of threats (RQ6.2)

This RQ explores how the various threats identified in RQ7.1 are addressed by the primary studies. We state the steps suggested by primary studies to mitigate the corresponding threats

Table 13. Mitigation of threats to validity in SCP studies

Threat No.	Threat Mitigation
T1	The results of a study should be validated using proper statistical tests. In case the underlying data does not fulfill the assumptions of a parametric statistical test, non-parametric statistical tests may be used.
T2	Multiple and stable performance measures should be used which give a realistic estimate of the model's performance.
T3	One should use an appropriate validation method so that the results are not biased due to selection of training and testing datasets.
T5	The confounding effect of variables may be evaluated by first building a univariate regression model of the confounding variable C on each predictor P. Thereafter, find the difference between predicted values by the regression model from P to obtain a new variable P'. The obtained P' is free from confounding effect.
T6	Controlled experiments should be carried out where only one specific predictor variable should be varied while keeping all other variables constant to determine the "causal" effect of predictor variables.
T7	Additional thresholds or rules may be used to determine the impact of these on dependent and the independent variables.
T9	OO metrics which are commonly used in literature and have been validated by previous studies may be used.
T10	The tools used for collecting independent and dependent variables should be manually verified to ascertain their correctness. The use of public datasets, which have been verified by previous studies also mitigate the threat.
T11	Strategies which have been well recognized in previous literature studies for computation of change-proneness attribute should be adopted, i.e. designation of a class as change-prone or not change-prone should be done in accordance with the definitions that have been well established in the past such as those followed by [9, 12].
T13	Effective data pre-processing strategies should be adopted for eliminating noisy data. Moreover, selection or extraction methods should be used for selecting relevant features.
T14	The results should be validated on datasets belonging to different domains.
T15	The results should be validated on datasets of appropriate size and on an appropriate number of datasets.
T16	The use of open–source datasets enhances the replicability of the study. Furthermore, the tools used to implement the approach should be available. The steps conducted in the experiment should be clearly presented to ease replicated experiments.
T17	The results should be validated on industrial datasets or datasets whose characteristics are similar to industrial datasets.
T18	The results should be validated on datasets developed using different programming languages.

in Table 13. The table states the mitigation of only those threats, whose mitigation could be extracted from primary studies.

The threats which were only mentioned in the "Threats to Validity" section of primary studies (T4, T8, T12), but could not be mitigated by the study or whose mitigation was not suggested are not stated in the table. Researchers should incorporate these steps (Table 13), while designing the experimental set-ups of their study in order to ensure reliable results. Also, several studies should be performed with different size, category, domain and other dataset characteristics to obtain generalized results in the domain of SCP.

6. Threats to validity

While searching for relevant candidate studies, we applied the search string to only the titles of the studies. Thus, we may fail to include studies which do not use the key terms in their titles. However, as we have extensively searched for candidate studies in the mentioned search databases, have included journal as well as conference stud-

ies, have searched for key authors and have also searched the reference lists of the included papers, we are positive that we have not missed a relevant study. It may be noted that the review is based on the presumption that all the primary studies are unprejudiced. In case, this is not true, there is a possible threat to the review results [6,71] The review also rules out all unpublished results [6].

In order to extract primary studies from candidate studies, both the authors independently applied the quality assessment criteria on each study. This practice ensures conclusion validity of the obtained results. Publication bias is a possible threat to the results of this review. In lieu of publication bias, it is highly likely that a primary study would publish positive results on application of a ML technique for developing SCP model as compared to negative results [72]. It could also be a scenario where researchers might claim that their proposed technique outperforms other established techniques in literature. This could lead to an exaggeration of the capability of ML techniques for developing SCP models. This threat was addressed in two ways. Firstly, we included primary studies which "reported and compared various data analysis algorithms amongst themselves for developing SCP models". These studies are unlikely to be biased towards specific ML techniques as they do not propose a data analysis algorithm of their own. Secondly, while comparing the predictive capability of ML techniques, we compared only those techniques which were "validated on at least three different datasets and were used by at least two primary studies" to avoid bias. Also, the statistics reported in the review were dataset wise after removal of outliers. Furthermore, we state the median values to get a realistic estimate of the capability of ML techniques for developing SCP models.

While evaluating the predictive capability of ML techniques, we also state the AUC results apart from accuracy results. Thus, we have accounted for possible bias which could occur by using imbalanced data as AUC is a stable performance measure.

In order to statistically compare the performance of ML techniques for developing SCP models (Table 9, 10), we have conducted several tests. However, certain erroneous inferences may occur due to conduct of several tests on the same data. This threat exists in the study.

7. Conclusions and future guidelines

An extensive systematic review was performed to analyze the current state of existing literature in the domain of SCP and to further identify research gaps in this domain. 38 primary studies were chosen to answer the various RQs. In lieu of the result discussions with respect to the explored RQs, we suggest certain guidelines to researchers in the SCP domain which are mentioned below.

- The product metrics especially the CK metrics suite have been widely used in primary studies for developing SCP models. However, the validation of process metrics and their combination with product metrics is limited in this domain. Researchers should conduct studies to assess the capability of only process metrics as well as a combination of both process and product metrics as predictors of software change.
- Feature selection/dimensionality reduction techniques have been used by a majority of studies. However, more studies should examine effective predictors using feature selection techniques in order to develop efficient SCP models.
- Most of the datasets used by the primary studies were open–source in nature. However, more studies should be conducted to validate commercial datasets to yield practical and generalized results. Also, datasets developed using other languages such as C#, Python, etc. needs to be evaluated by literature studies.
- It was observed that 25–100% of datasets in a majority of the SCP studies were imbalanced in nature (had less than 40% of changed classes). Researchers in future should evaluate methods to develop effective models from imbalanced datasets as correct identification of change-prone classes is crucial. This would aid developers in prioritizing their resources

effectively during maintenance and testing phases of a software development lifecycle.
- Within project validation has become a common standard while validating SCP models. Though, cross-project validation has also been investigated, however, studies in the future should explore cross-organization and cross-company validation. Effective transfer learning in cross-organization and cross-company scenario is the need of the hour, which should be actively investigated by researchers in future studies. Furthermore, temporal validation, which takes into account the time dimension should also be explored in the SCP domain.
- Apart from accuracy, the use of AUC measure is prominent in literature for evaluating SCP models. Stable performance measures such as AUC should be used by researchers in future as they give a realistic estimate of the performance of models which are developed from imbalanced datasets.
- It was observed that a majority of studies used ML algorithms and these algorithms are effective in the domain of SCP. However, more studies should be conducted which assess and compare the effectiveness of statistical and ML algorithms for SCP as we could find limited data in literature which compares the performance of different algorithms for developing effective SCP models. Also, more researchers should explore the use of ensemble of algorithms as an alternative to other data analysis algorithms for developing SCP models.
- It was found that SBA (HIDER, GFS-AB, GFS-GP, GFS-LB, GFS-SP and NNEP) exhibited effective accuracy results. However, data to assess and compare the ability of SBA's was limited. More studies which investigate the effectiveness of SBA in the domain of SCP are required to yield conclusive results about their capability. Studies should be conducted to evaluate the effectiveness of SBA and compare their performance with other established ML and statistical techniques.
- The results indicate that a majority (66%) of primary studies use statistical tests for verifying the obtained results. This is a good practice which should be continued in future studies.
- It is mandatory to account for possible "Threats to Validity", while designing experiments to yield effective and reliable results.

Though there are a number of research papers that illustrate quantitative results from SCP in lab environments, there is a need for longitudinal studies with developers in industry that focuses on qualitative research so that the effectiveness of SCP models in industry may be understood in depth.

References

[1] A.G. Koru and H. Liu, "Identifying and characterizing change-prone classes in two large-scale open-source products," *Journal of Systems and Software*, Vol. 80, No. 1, 2007, pp. 63–73.

[2] Y. Zhou, H. Leung, and B. Xu, "Examining the potentially confounding effect of class size on the associations between object-oriented metrics and change-proneness," *IEEE Transactions on Software Engineering*, Vol. 35, No. 5, 2009, pp. 607–623.

[3] A.G. Koru and J. Tian, "Comparing high-change modules and modules with the highest measurement values in two large-scale open-source products," *IEEE Transactions on Software Engineering*, Vol. 31, No. 8, 2005, pp. 625–642.

[4] E. Arisholm, L.C. Briand, and A. Foyen, "Dynamic coupling measurement for object-oriented software," *IEEE Transactions on software engineering*, Vol. 30, No. 8, 2004, pp. 491–506.

[5] B.A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-based software engineering and systematic reviews*. CRC Press, 2015, Vol. 4.

[6] R. Malhotra, M. Khanna, and R.R. Raje, "On the application of search-based techniques for software engineering predictive modeling: A systematic review and future directions," *Swarm and Evolutionary Computation*, Vol. 32, 2017, pp. 85–109.

[7] R. Malhotra and M. Khanna, "Threats to validity in search-based predictive modelling for software engineering," *IET Software*, Vol. 12, No. 4, 2018, pp. 293–305.

[8] D. Godara and R. Singh, "A review of studies on change proneness prediction in object oriented software," *International Journal of Computer Applications*, Vol. 105, No. 3, 2014, pp. 35–41.

[9] R. Malhotra and A.J. Bansal, "Software change prediction: A literature review," *International Journal of Computer Applications in Technology*, Vol. 54, No. 4, 2016, pp. 240–256.

[10] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert systems with applications*, Vol. 36, No. 4, 2009, pp. 7346–7354.

[11] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, Vol. 38, No. 6, 2011, pp. 1276–1304.

[12] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: A systematic literature review," *Information and Software Technology*, Vol. 55, No. 8, 2013, pp. 1397–1418.

[13] R.S. Wahono, "A systematic literature review of software defect prediction: research trends, datasets, methods and frameworks," *Journal of Software Engineering*, Vol. 1, No. 1, 2015, pp. 1–16.

[14] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," *IEEE Transactions on Software Engineering*, Vol. 45, No. 2, 2017, pp. 111–147.

[15] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, Vol. 27, 2015, pp. 504–518.

[16] P.K. Singh, D. Agarwal, and A. Gupta, "A systematic review on software defect prediction," in *2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2015, pp. 1793–1797.

[17] C. Catal, "Software fault prediction: A literature review and current trends," *Expert systems with applications*, Vol. 38, No. 4, 2011, pp. 4626–4636.

[18] X. Zhu, Y. He, L. Cheng, X. Jia, and L. Zhu, "Software change-proneness prediction through combination of bagging and resampling methods," *Journal of Software: Evolution and Process*, Vol. 30, No. 12, 2018, p. e2111.

[19] G. Catolino and F. Ferrucci, "An extensive evaluation of ensemble techniques for software change prediction," *Journal of Software: Evolution and Process*, 2019, p. e2156.

[20] G. Catolino, F. Palomba, A. De Lucia, F. Ferrucci, and A. Zaidman, "Enhancing change prediction models using developer-related factors," *Journal of Systems and Software*, Vol. 143, 2018, pp. 14–28.

[21] R. Malhotra and M. Khanna, "Dynamic selection of fitness function for software change prediction using particle swarm optimization," *Information and Software Technology*, Vol. 112, 2019, pp. 51–67.

[22] L. Kumar, S. Lal, A. Goyal, and N. Murthy, "Change-proneness of object-oriented software using combination of feature selection techniques and ensemble learning techniques," in *Proceedings of the 12th Innovations on Software Engineering Conference*. ACM, 2019, p. 8.

[23] Y. Ge, M. Chen, C. Liu, F. Chen, S. Huang, and H. Wang, "Deep metric learning for software change-proneness prediction," in *International Conference on Intelligent Science and Big Data Engineering*. Springer, 2018, pp. 287–300.

[24] H. Lu, Y. Zhou, B. Xu, H. Leung, and L. Chen, "The ability of object-oriented metrics to predict change-proneness: a meta-analysis," *Empirical software engineering*, Vol. 17, No. 3, 2012, pp. 200–242.

[25] M.O. Elish and M. Al-Rahman Al-Khiaty, "A suite of metrics for quantifying historical changes to predict future change-prone classes in object-oriented software," *Journal of Software: Evolution and Process*, Vol. 25, No. 5, 2013, pp. 407–437.

[26] R. Malhotra and M. Khanna, "An exploratory study for software change prediction in object-oriented systems using hybridized techniques," *Automated Software Engineering*, Vol. 24, No. 3, 2017, pp. 673–717.

[27] D. Romano and M. Pinzger, "Using source code metrics to predict change-prone java interfaces," in *27th International Conference on Software Maintenance (ICSM)*. IEEE, 2011, pp. 303–312.

[28] E. Giger, M. Pinzger, and H.C. Gall, "Can we predict types of code changes? An empirical analysis," in *9th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2012, pp. 217–226.

[29] D. Azar and J. Vybihal, "An ant colony optimization algorithm to improve software quality prediction models: Case of class stability," *Information and Software Technology*, Vol. 53, No. 4, 2011, pp. 388–393.

[30] S. Karus and M. Dumas, "Code churn estimation using organisational and code metrics: An experimental comparison," *Information and Software Technology*, Vol. 54, No. 2, 2012, pp. 203–211.

[31] J.M. Bieman, G. Straw, H. Wang, P.W. Munger, and R.T. Alexander, "Design patterns and change proneness: An examination of five evolving systems," in *Proceedings. 5th International*

Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. No. 03EX717). IEEE, 2004, pp. 40–49.

[32] N. Zazworka, C. Izurieta, S. Wong, Y. Cai, C. Seaman, F. Shull *et al.*, "Comparing four approaches for technical debt identification," *Software Quality Journal*, Vol. 22, No. 3, 2014, pp. 403–426.

[33] X. Zhu, Q. Song, and Z. Sun, "Automated identification of change-prone classes in open source software projects." *Journal of Software*, Vol. 8, No. 2, 2013, pp. 361–366.

[34] M. Lindvall, "Are large C++ classes change-prone? An empirical investigation," *Software: Practice and Experience*, Vol. 28, No. 15, 1998, pp. 1551–1558.

[35] M. Lindvall, "Measurement of change: stable and change-prone constructs in a commercial C++ system," in *Proceedings Sixth International Software Metrics Symposium*. IEEE, 1999, pp. 40–49.

[36] Y. Liu and T.M. Khoshgoftaar, "Genetic programming model for software quality classification," in *Proceedings Sixth International Symposium on High Assurance Systems Engineering. Special Topic: Impact of Networking*. IEEE, 2001, pp. 127–136.

[37] M. Al-Khiaty, R. Abdel-Aal, and M.O. Elish, "Abductive network ensembles for improved prediction of future change-prone classes in object-oriented software." *International Arab Journal of Information Technology*, Vol. 14, No. 6, 2017, pp. 803–811.

[38] T.M. Khoshgoftaar, N. Seliya, and Y. Liu, "Genetic programming-based decision trees for software quality classification," in *15th International Conference on Tools with Artificial Intelligence*. IEEE, 2003, pp. 374–383.

[39] L. Kumar, S.K. Rath, and A. Sureka, "Empirical analysis on effectiveness of source code metrics for predicting change-proneness," in *10th Innovations in Software Engineering Conference*. ACM, 2017, pp. 4–14.

[40] N. Tsantalis, A. Chatzigeorgiou, and G. Stephanides, "Predicting the probability of change in object-oriented systems," *IEEE Transactions on Software Engineering*, Vol. 31, No. 7, 2005, pp. 601–614.

[41] L. Kumar, S.K. Rath, and A. Sureka, "Using source code metrics to predict change-prone web services: A case-study on ebay services," in *Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE)*. IEEE, 2017, pp. 1–7.

[42] A.R. Sharafat and L. Tahvildari, "Change prediction in object-oriented software systems: A probabilistic approach," *Journal of Software*, Vol. 3, No. 5, 2008, pp. 26–39.

[43] L. Kumar, R.K. Behera, S. Rath, and A. Sureka, "Transfer learning for cross-project change-proneness prediction in object-oriented software systems: A feasibility analysis," *ACM SIGSOFT Software Engineering Notes*, Vol. 42, No. 3, 2017, pp. 1–11.

[44] D. Azar, "A genetic algorithm for improving accuracy of software quality predictive models: a search-based software engineering approach," *International Journal of Computational Intelligence and Applications*, Vol. 9, No. 02, 2010, pp. 125–136.

[45] R. Malhotra and R. Jangra, "Prediction and assessment of change prone classes using statistical and machine learning techniques," *Journal of Information Processing Systems*, Vol. 13, No. 4, 2017, pp. 778–804.

[46] A.R. Han, S.U. Jeon, D.H. Bae, and J.E. Hong, "Measuring behavioral dependency for improving change-proneness prediction in uml-based design models," *Journal of Systems and Software*, Vol. 83, No. 2, 2010, pp. 222–234.

[47] R. Malhotra and M. Khanna, "An empirical study for software change prediction using imbalanced data," *Empirical Software Engineering*, Vol. 22, No. 6, 2017, pp. 2806–2851.

[48] S. Eski and F. Buzluca, "An empirical study on object-oriented metrics and software evolution in order to reduce testing costs by predicting change-prone classes," in *Fourth International Conference on Software Testing, Verification and Validation Workshops*. IEEE, 2011, pp. 566–571.

[49] M. Yan, X. Zhang, C. Liu, L. Xu, M. Yang, and D. Yang, "Automated change-prone class prediction on unlabeled dataset using unsupervised method," *Information and Software Technology*, Vol. 92, 2017, pp. 1–16.

[50] A. Agrawal and R.K. Singh, "Empirical validation of OO metrics and machine learning algorithms for software change proneness prediction," in *Towards Extensible and Adaptable Methods in Computing*. Springer, 2018, pp. 69–84.

[51] C. Liu, Y. Dan, X. Xin, Y. Meng, and Z. Xiaohong, "Cross-project change-proneness prediction," in *42nd Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2018, pp. 64–73.

[52] R. Malhotra and M. Khanna, "Investigation of relationship between object-oriented metrics and change proneness," *International Journal of Ma-*

chine Learning and Cybernetics, Vol. 4, No. 4, 2013, pp. 273–286.

[53] L. Kaur and M. Ashutosh, "A comparative analysis of evolutionary algorithms for the prediction of software change," in *International Conference on Innovations in Information Technology (IIT)*. IEEE, 2018, pp. 188–192.

[54] R. Malhotra and A.J. Bansal, "Cross project change prediction using open source projects," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2014, pp. 201–207.

[55] R. Malhotra and M. Khanna, "Prediction of change prone classes using evolution-based and object-oriented metrics," *Journal of Intelligent and Robotic Systems Fuzzy Systems*, Vol. 34, No. 3, 2018, pp. 1755–1766.

[56] R. Malhotra and M. Khanna, "A new metric for predicting software change using gene expression programming," in *5th International Workshop on Emerging Trends in Software Metrics*. ACM, 2014, pp. 8–14.

[57] R. Malhotra and M. Khanna, "Particle swarm optimization-based ensemble learning for software change prediction," *Information and Software Technology*, Vol. 102, 2018, pp. 65–84.

[58] C. Marinescu, "How good is genetic programming at predicting changes and defects?" in *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE, 2014, pp. 544–548.

[59] M.O. Elish, H. Aljamaan, and I. Ahmad, "Three empirical studies on predicting software maintainability using ensemble methods," *Soft Computing*, Vol. 19, No. 9, 2015, pp. 2511–2524.

[60] R. Malhotra and M. Khanna, "Mining the impact of object oriented metrics for change prediction using machine learning and search-based techniques," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2015, pp. 228–234.

[61] A. Bansal, "Empirical analysis of search based algorithms to identify change prone classes of open source software," *Computer Languages, Systems and Structures*, Vol. 47, 2017, pp. 211–231.

[62] S.R. Chidamber and C.F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, 1994, pp. 476–493.

[63] J. Bansiya and C.G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, Vol. 28, No. 1, 2002, pp. 4–17.

[64] M. Lorenz and J. Kidd, *Object-oriented software metrics: A practical guide*. Prentice-Hall, Inc., 1994.

[65] W. Li and S. Henry, "Object-oriented metrics that predict maintainability," *Journal of Systems and Software*, Vol. 23, No. 2, 1993, pp. 111–122.

[66] K. Gao, T.M. Khoshgoftaar, and A. Napolitano, "Combining feature subset selection and data sampling for coping with highly imbalanced software data," in *Software Engineering Knowledge Engineering Conference*, 2015, pp. 439–444.

[67] H. He and E.A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, No. 9, 2009, pp. 1263–1284.

[68] C.G. Weng and J. Poon, "A new evaluation measure for imbalanced datasets," in *7th Australian Data Mining Conference*. Australian Computer Society, Inc., 2008, pp. 27–32.

[69] M.A. De Almeida and S. Matwin, "Machine learning method for software quality model building," in *International symposium on methodologies for intelligent systems*. Springer, 1999, pp. 565–573.

[70] R. Malhotra, *Empirical research in software engineering: Concepts, analysis and applications*. CRC Press, 2016.

[71] W. Afzal and R. Torkar, "On the application of genetic programming for software engineering predictive modeling: A systematic review," *Expert Systems with Applications*, Vol. 38, No. 9, 2011, pp. 11 984–11 997.

[72] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Software Technology*, Vol. 54, No. 1, 2012, pp. 41–59.

Appendix A.

Table A1 states the results of data extraction, i.e. the key parameters extracted from each primary study. Table A2 states the datasets that have been used by at least two primary studies.

Table A1: Key parameters of primary studies

S No.	Year	Venue	Predictors	Datasets	Data Analysis algorithms	Performance measures	Validation method	Statistical test
PS1	2001	Conf.	NOCI, 4 Lines of code measures	Windows based software application (VLWA dataset)	LR, GP	Type I error, Type II error, Overall misclassification rate	Hold-out	–
PS2	2003	Conf.	NOCI, 4 Lines of Code measures	Windows based Software Application (VLWA Dataset)	GP, Decision tree based GP	Type I error, Type II error, Overall misclassification rate	Hold-out	–
PS3	2005	Journal	CK, NOM, Probability values related to history, Probability of Change	JFlex, JMol	LR	Accuracy, Recall, False Positive Ratio, False Negative Ratio, Goodness of Fit	–	–
PS4	2005	Journal	AID, ALD, CC, SLOC, NOP, MNOB, MPC, NOLV, NIC	JFlex	Non-linear system of equations, Linear system of equations, Depth first search graphs, Binary dependencies	Accuracy, False Positive Ratio, False Negative Ratio	–	–
PS5	2010	Journal	22 OO metrics: 4 cohesion metrics, 4 coupling metrics, 7 inheritance metrics, 7 size metrics	Bean Browser, Ejbvoyager, Free, Javamapper, Jchempaint, Jedit, Jetty, Jigsaw, Jlex, Lmjs, Voji, 4 versions of JDK	C4.5, GA	Accuracy, J-Index	Ten-fold	–
PS6	2010	Journal	CK, Lorenz and Kidd [64], MOOD metrics, BDM	Jflex (13 versions)	Stepwise Multiple Regression	Goodness of Fit	–	ANOVA
PS7	2011	Journal	22 OO metrics: 4 cohesion metrics, 4 coupling metrics, 7 inheritance metrics, 7 size metrics	Bean Browser, Ejbvoyager, Free, Javamapper, Jchempaint, Jedit, Jetty, Jigsaw, Jlex, Voji	C4.5, ACO	Accuracy	Ten-fold	Wilcoxon Signed Rank

S No.	Year	Venue	Predictors	Datasets	Data analysis algorithms	Performance measures	Validation method	Statistical test
PS8	2011	Conf.	CK, QMOOD	Yari (3 versions), UCdetector (4 versions), JFreeChart (4 versions)	Combined Rank List Mechanism	Hit-Ratio (Recall), Change Cost, Cost ratio	–	–
PS9	2011	Journal	62 OO metrics: 18 cohesion metrics, 20 coupling metrics, 17 inheritance metrics, 7 size metrics	102 Java Systems	Random effect meta-analysis	AUC	–	–
PS10	2011	Technical Report	CK, 3 usage metrics, 3 complexity metrics, Interface Usage Cohesion metric	8 Eclipse plug-in projects and 2 Hibernate Systems	NB, SVM, MLP	Recall, Precision, AUC	Ten-Fold	Wilcoxon Signed Rank
PS11	2012	Conf.	CK, 7 Network centrality measures from social network analysis	19 Eclipse plug-in projects, Azureus	MLP, BN	Recall, Precision, AUC	Ten-Fold	Friedman, Wilcoxon Signed rank
PS12	2013	Journal	CK, 16 Evolution-based metrics	PeerSim (9 versions), VSSPlugin (13 versions)	LR	Accuracy	Ten-Fold	Wilcoxon Signed Rank
PS13	2013	Journal	CK, 16 other class-level metrics	Frinika, FreeMind, OrDrumBox	LR, RF, MLP, BG	Recall, Specificity, AUC	Ten-Fold	–
PS14	2014	Conf.	CK, SLOC	Apache Abdera (4 versions), Apache POI (4 versions), Apache Rave (4 versions)	LB	Precision, AUC	Ten-Fold, Cross-project	–
PS15	2014	Conf.	CK, SLOC	Simutrans, Glest	GEP	Accuracy, AUC	Ten-Fold	–
PS16	2014	Conf.	NOM, DIT, RFC, NOC, CBO, TCC, SLOC	ArgoUmL, Findbugs, FOP, FreeCol	GP	Recall, Precision	–	Proportion test
PS17	2015	Journal	CK	PeerSim, VSSPlugin	LR, MLP, RBF, SVM, DT, GEP, k-means, Ensemble of Models (Best in training, Bagging, Boosting, Majority Voting, Non-linear Decision tree Forest)	Accuracy, AUC	Hold-out, leave-one out	–

S No.	Year	Venue	Predictors	Datasets	Data analysis algorithms	Performance measures	Validation method	Statistical test
PS18	2015	Conf.	CK, SLOC, NOM, NIV, NPM, NIM, NOA	Simutrans, Glest, Celestia	LR, RF, BG, MLP, AB, CPSO, HIDER, MPLCS, SUCS, GFS-SP, NNEP	Accuracy, AUC, Precision, Specificity, Recall, F-measure, G-measure	Ten-fold, Cross-project	Friedman
PS19	2017	Journal	CK, SLOC	Apache Rave, Apache Math	NB, BN, LB, AB, GFS-AB, GFS-LB, GFS-MLB, HIDER, NNEP, PSO-LDA, GFS-GP, GFS-SP, SLAVE	Accuracy, G-mean	Ten-fold	Wilcoxon Signed Rank
PS20	2017	Journal	CK, 16 Evolution-based metrics	VSSPlugin (13 versions)	GMDH	Accuracy, AUC, Precision, Recall, F-measure	Hold-out	–
PS21	2017	Conf.	62 OO metrics: 19 cohesion metrics, 19 coupling metrics, 17 inheritance metrics, 7 size metrics	Eclipse	LR, NB, Extreme Machine Learning (Linear, Polynomial and RBF kernels), SVM (Linear, Polynomial and Sigmoid kernels), Ensembles of Techniques (Best in Training, Majority Voting)	Accuracy, AUC	Ten-fold	–
PS22	2017	Conf.	21 OO metrics including CK metrics	Ebay Services (5 versions)	Least Square SVM (Linear, Polynomial and RBF kernels)	Accuracy, F-measure	Twenty-fold	*t*-test
PS23	2017	Journal	61 OO metrics	10 Eclipse plug-ins	LR, MLP, RBF, DT, RF, Ensembles of Techniques (Best in Training, Majority Voting, Non-Linear Decision Tree Forest)	Accuracy, Precision, Recall, F-measure	Ten-fold, Cross-project	Wilcoxon Signed Rank
PS24	2017	Journal	13 OO metrics including CK suite	AOI, SweetHome 3D	LR, RF, AB, BG, MLP, NB, BN, J48, NNGE	Recall, Specificity, AUC	Ten-fold, Cross-project	*t*-test

S No.	Year	Venue	Predictors	Datasets	Data analysis algorithms	Performance measures	Validation method	Statistical test
PS25	2017	Journal	18 OO metrics including CK suite, SLOC, QMOOD suite, AC, EC, LCOM3, AMC, IC, CBM	Six Android application packages	PSO-LDA, NNEP, GFS-LB, CART, SUCS, CPSO, C4.5, GA-ADI, HIDER, MLP-CG, MPLCS, LDA, DT-GA, XCS, SVM	Recall, PF, Balance, G-mean	Ten-fold	Friedman, Wilcoxon Signed Rank
PS26	2017	Journal	18 OO metrics including CK suite, SLOC, QMOOD suite, AC, EC, LCOM3, AMC, IC, CBM	Three Android application packages, Net, IO, Log4j	MLP, RF, NB, AB, LB, BG	Recall, Precision, Accuracy, AUC, Balance, G-mean	Ten-fold, Inter-version	Friedman, Wilcoxon Signed Rank
PS27	2017	Journal	10 OO metrics including CK suite, Li and Henry [65], SIZE1	Ant, Antlr, Argouml, Azureus, Freecol, Freemind, Hibernate, Jgraph, Jmeter, Jstock, Jung, Junit, Lucene, Weka	LB, MLP, RBF, SVM, k-means, CLAMI, CLAMI+	Accuracy, AUC, F-measure	Within project, Cross-project	Friedman, Nemenyi
PS28	2018	Chapter	15 OO metrics including CK suite	GATE, Tuxguitar, FreeCol, KolMafia, Legatus	MLP, LR, RF, KStar, PART, BG, BN	Recall, Specificity, AUC, F-measure	Ten-fold	Friedman
PS29	2018	Journal	Entropy of changes, number of developers, structural and semantic scattering of developers, evolutio-based metrics, OO metrics	Ant, Cassandra, Lucene, POI, Synapse, Velocity, Xalan, Xerces, ArgoUML, aTunes, FreeMind, JEdit, JFreeChart, JHotDraw, JVLT, pBeans, pdfTranslator, Redaktor, Serapion, Zuzel	LR	Recall, Precision, AUC, MCC, Brier Score	3 month sliding window to train and test models	Mann–Whitney, Cliff
PS30	2018	Conf.	CK, SLOC	ArgoUML, FreeCol, JMeter, Jung, Weka (4 versions each)	LR, NB, DT, SVM, Decision Table, Deep Metric Learning	Recall, Precision, F-Measure	Cross-project	–
PS31	2018	Conf.	CK	Ant, Antlr, ArgoUML, Azureus, FreeCol, FreeMind, Hibernate, JGraph, JMeter, JStock, Jung, JUnit, Lucene, Weka	BN	AUC	Cross-project	Wilcoxon Signed Rank

S No.	Year	Venue	Predictors	Datasets	Data analysis algorithms	Performance measures	Validation method	Statistical test
PS32	2018	Conf.	TCC, CHL, CHV, CHE, CHB, MI, AC, EC, Instability	JFreeChart (4 versions)	LR, LDA, GFS-GP, GFS-LB, GFS-SP, GFS-AB, NNEP, GANN	Accuracy	Ten-fold	Friedman, Wilcoxon Signed Rank
PS33	2018	Journal	CK, 16 Evolution-based metrics	Android Contacts (5 versions), Android Gallery2 (4 versions)	LR, MLP, NB, RF, AB, BG, LB	Accuracy, AUC	Ten-fold	Friedman, Wilcoxon Signed Rank
PS34	2018	Journal	CK, SLOC	Six Android application packages, IO, Net, Math, Log4j	RF, BG, AB, LB, 4 CPSO voting based fitness ensembles	Balance, G-Mean	Ten-fold	Friedman, Wilcoxon Signed Rank
PS35	2018	Journal	Complexity metrics, Word metrics, Network Metrics	Ant, Eclipse, JEdit, Itextpdf, Liferay, Lucene, Struts, Tomcat	C4.5, NB, SVM	Recall, Precision, F-measure, AUC, MCC	Ten-fold	Scott–Knott
PS36	2019	Journal	OO metrics, Process metrics, Developer related factors	Ant, Log4j, Lucene, Pbeans, POI, Synapse, Velocity, Xalan, Xerces, JEdit	LR, Simple Logistic, NB, MLP, AB, BG, RF, Voting	F-measure, AUC, MCC	Ten-fold	Scott–Knott, Cliff
PS37	2019	Conf.	20 OO metrics including CK metrics suite	compare, webdav, debug, update, core, swt, team, pde, ui, jdt	LR, Linear Regression, Polynomial Regression, DT, SVM (Linear, Polynomial, RBF), Extreme ML (Linear, Polynomial, RBF), Least-Square SVM (Linear, Polynomial, RBF) Simple Logistic, Neural network with 5 training algorithms, Ensembles of Techniques (Best in Training, Majority Voting, Non-Linear Decision Tree Forest)	Accuracy, F-measure	Five-fold	Wilcoxon Signed Rank

S No.	Year	Venue	Predictors	Datasets	Data analysis algorithms	Performance measures	Validation method	Statistical test
PS38	2019	Journal	CK, SLOC	AOI, CLick, DrJava, Giraph, Gora, Hama, HyperSQL DB, JabRef, JMeter, JEdit, LogicalDoc, Maven, Phoenix, SubSonic, ZooKeeper	LR, AB, BG, RF, LB, 4 CPSO voting based fitness ensembles, ASOF Classifier	F-measure, AUC, MCC	Ten-fold	Scott–Knott, Cliff

Note: "–" indicates the corresponding information was not found in the study.

AC: Afferent Coupling; AID: Access of Imported Data; ALD: Access of Local Data; AMC: Average Method Complexity; ASOF: Adaptive Selection of Optimum Fitness; BDM: Behavioral Dependency Measurement; CBM: Coupling Between Methods of a Class; Conf.: Conference; CC: Cyclomatic Complexity; CLAMI: Clustering Lableing Metric selection and Instance selection; EC: Efferent Coupling; IC: Inheritance Coupling; MCC: Mathews Correlation Coefficient; MNOB: Maximum Number Of Branches; MOOD: Metrics for Object-Oriented Design; MPC: Message Passing Coupling; NIC: Number of Imported Classes, NIM: Number of Instance Methods; NIV: Number of Instance Variables; NOCI: Number of Times Source File was Inspected; NOLV: Number Of Local Variables; NOA: Number of Attributes; NOM: Number of Methods per Class; NOP: Number Of Parameters; NPM: Number of Public Methods; RBF: Radial Basis Function; TCC: Total Cyclomatic Complexity; CHL: Cumulative Halstead Length; CHV: Cumulative Halstead Volume; CHE: Cumulative Halstead Effort; CHB: Cumulative Halstead Bugs; MI: Maintainability Index; QMOOD: Quality Model for Object-Oriented Design.

Table A2. Commonly used datasets

Dataset Name	Study Numbers
Android Bluetooth	PS27, PS26, PS34
Android Calendar	PS27, PS26, PS34
Android Contacts	PS26, PS33, PS34
Android Gallery	PS26, PS33, PS34
Android MMS	PS27, PS26, PS34
Android Telephony	PS26, PS34
Ant	PS27, PS29, PS31, PS35, PS36
Antlr	PS27, PS31
AOI	PS24, PS38
ArgoUML	PS16, PS27, PS29, PS30, PS31
Azureus	PS11, PS27, PS31
Bean Browser	PS5, PS7
Eclipse	PS10, PS11, PS21, PS23, PS35
Free	PS5, PS7
FreeCol	PS16, PS27, PS28, PS30, PS31
FreeMind	PS13, PS27, PS29, PS31
Glest	PS15, PS18
Hibernate	PS10, PS27, PS31
IO	PS27, PS34
JChempaint	PS5, PS7
JEdit	PS5, PS29, PS35, PS36, PS38
Jetty	PS5, PS7
JFlex	PS3, PS4
JFreeChart	PS8, PS29, PS32
JGraph	PS27, PS31
Jigsaw	PS5, PS7
Jlex	PS5, PS7
JavaMapper	PS5, PS7
JMeter	PS30, PS31, PS38
Jung	PS30, PS31
Log4j	PS27, PS34, PS36
Lucene	PS29, PS31, PS35, PS36
Math	PS19, PS34
Net	PS27, PS34
PeerSim	PS12, PS17
POI	PS14, PS29, PS36
Synapse	PS29, PS36
Velocity	PS29, PS36
Voji	PS5, PS7
VSSPlugin	PS12, PS17, PS20
Weka	PS30, PS31
Windows based software application (VLWA)	PS1, PS2
Xalan	PS29, PS36