e-Informatica Software Engineering Journal, Volume 16, Issue 1, 2022, pages: 220105, DOI 10.37190/e-Inf220105

A Systematic Review of Ensemble Techniques for Software Defect and Change Prediction

Megha Khanna*

*Department of Computer Science, Sri Guru Gobind Singh College of Commerce, University of Delhi

meghakhanna86@gmail.com

Abstract

Background: The use of ensemble techniques have steadily gained popularity in several software quality assurance activities. These aggregated classifiers have proven to be superior than their constituent base models. Though ensemble techniques have been widely used in key areas such as Software Defect Prediction (SDP) and Software Change Prediction (SCP), the current state-of-the-art concerning the use of these techniques needs scrutinization.
Aim: The study aims to assess, evaluate and uncover possible research gaps with respect to the use of ensemble techniques in SDP and SCP.
Method: This study conducts an extensive literature review of 77 primary studies on the basis of the category, application, rules of formulation, performance, and possible threats of the proposed/utilized ensemble techniques.
Results: Ensemble techniques were primarily categorized on the basis of similarity, aggregation, relationship, diversity, and dependency of their base models. They were also found effective in several applications such as their use as a learning algorithm for developing SDP/SCP models and for addressing the class imbalance issue.
Conclusion: The results of the review ascertain the need of more studies to propose, assess, validate, and compare various categories of ensemble techniques for diverse applications in SDP/SCP such as transfer learning and online learning.

Keywords: Ensemble learning, Software change prediction, Software defect prediction, Software quality, Systematic review

1. Introduction

Technology has ensured that software is a fundamental part of every activity. This necessitates the development and maintenance of good quality software products. However, rigid deadlines, limited budgets, and scarce resources often impede the development of competent software products. Thus, it is essential to perform Software Quality Assurance (SQA) activities so that the quality of software products is not compromised. Software Defect Prediction (SDP) and Software Change Prediction (SCP) models, which predict defect prone and change prone parts of software in its early stages of development are popular means of prioritizing effort for SQA activities. Defect prone and change prone parts, though few, account for a majority of the defects and changes in a software [1, 2]. Thus, SQA efforts should be focused on these parts as they need to be meticulously designed and

Submitted: 23 Jul. 2021; Revised: 09 Jan. 2022; Accepted: 21 Feb. 2022; Available online: 04 Mar. 2022

carefully verified [3–5]. Software practitioners may design and verify these parts in such a manner so that future occurrences of defects can be minimized and the effect of changes may be localized [2, 6–9]. These activities would assure the timely delivery of cost-effective and maintainable software products.

Over the years, the research community has extensively explored various algorithms for developing SDP and SCP models. Amongst the various categories, the "ensemble" techniques are a key category that have been widely investigated by the researchers [10–13]. These techniques are an assembly of diverse base models, where each base model attempts to resolve the original problem at hand [12], which in our case is the determination of defect prone and change prone classes. Ensemble Techniques (ET) output the result of the aggregated base models as "aggregation" provides a more stable and reliable estimate with an improved predictive ability [14–17]. Combining several diverse base models is analogous to consulting a committee of experts thereby resulting in more accurate predictions [18].

Given the unique nature of ET and its improved performance over single models, it is vital to systematically summarize and analyze the empirical evidence for its use in SDP and SCP literature. Previous studies have comprehensively evaluated the use of ET for feature selection [19], effort estimation [12], and class imbalance problem [20]. Also, there have been several efforts by researchers that systematically summarize the SDP and SCP studies from various aspects. Radjenovic et al. [21] examined various software metrics in the context of SDP and found object-oriented metrics to be most prevalent. Catal [22] investigated SDP studies in the period 1990–2009 to summarize the metrics, performance measures, methods, datasets, and experimental results used in the studies. Hosseini et al. [23] synthesized the state of the art concerning the use of cross-project models in SDP studies. Malhotra [10] evaluated the use of several machine learning techniques for SDP. Amongst other findings, she reported that ET were used in 18% of 65 primary studies. Wahono [11] conducted a systematic literature review of 71 SDP studies to investigate the methods, datasets, frameworks, and research trends in SDP. An interesting result of the review pointed out that researchers have suggested the use of ET and the use of the boosting algorithm for improving the performance of existing machine learning classifiers. Two other previous reviews have also scrutinized SDP and SCP studies [24, 25], but with respect to the use of search-based algorithms and validity threats specific to its usage. A recent review by Malhotra and Khanna [13] assessed the various predictors, techniques and their predictive performance, experimental settings and validity threats in 38 SCP studies. Amongst other results, the review study encouraged the use of ET as they were found to be popular as well as effective (when evaluated in terms of accuracy and AUC measures) in the SCP domain. This study complements the previous work as we investigate the use of ET in both SCP as well as SDP domain (a related area of SCP). We analyze the several categories of ET, their rules, predictive capability and their possible application in aiding the SDP/SCP problems. Certain other researchers [26–28] have also reviewed SDP and SCP literature. However, to the best of the author's knowledge, there has been no study till date which has focused on a comprehensive evaluation of the use of ET for SDP and SCP, which is the primary aim of this study.

To facilitate an extensive analysis of ET used in SDP and SCP literature we examine the following Research Questions (RQ):

– RQ1: What is the categorization of ET? Which is the most popular category of ET in SDP/SCP literature?
– RQ2: What are the various applications of ET in SDP/SCP literature?

- RQ3: Which rules/mechanisms are used for combining base models to form ET in SDP/SCP literature?
- RQ4: What is the performance of ET for various tasks in the domain of SDP/SCP? How does the performance of ET compare to other non-ensemble techniques and amongst each other?
- RQ5: What are the various reported threats to validity specific to the use of ET in SDP/SCP literature?

The objective of the study is to systematically collect and rigorously evaluate literature studies that develop classification models using ET to predict defect prone and change prone parts of a software. This would help in summarizing the current trends for the use of ET in SDP/SCP literature and further determine gaps in existing research. The study is structured into five further sections, which includes research methodology followed to conduct the review (Section 2), review results (Section 3), discussion and proposed future work (Section 4), threats to validity of the review (Section 5) and conclusions (Section 6).

2. Review methodology

To accomplish our goals, we performed a systematic literature review in three stages according to the guidelines stated by Kitchenham et al. [29]. The first stage was planning which included identifying the review objectives and the protocol for conducting the review. As discussed in the previous section, we evaluated existing systematic reviews on the topic. However, these reviews did not focus on the application of ET in SDP and SCP. Thus, the primary objective of this review was to study the existing literature and provide a critical overview of the use of ET in the domain of SDP and SCP. Thereafter, the research questions were formulated and the review protocol was defined. The review protocol characterizes the search strategy for extracting relevant studies from literature, criteria for including and excluding the collected studies, a benchmark for quality assessment of candidate studies, processes for data extraction from primary studies, and the method for synthesis of extracted data. The second stage of the review involves conducting the review according to the procedures decided in the planning stage. This stage collects the relevant studies and scrutinizes them if they are fit to be primary studies of the review. Thereafter, data pertaining to the formulated RQ's is extracted and synthesized. The last stage of the review concerns itself with reporting of the findings of the review. Here, we report crisp answers to the investigated RQ's and document research gaps in the form of future work to interested researchers.

2.1. Search strategy

To search for relevant studies, we need to prepare a search string by combining appropriate search terms. These search terms were determined by selecting "key" terms from the RQ's of the review [21]. Furthermore, equivalent terms and other possible spellings were examined for the identified search terms. Thereafter, the search string was defined by combining all synonymous terms using Boolean "OR" and all distinct terms by Boolean "AND". The following search string was used:

("software") AND ("Defect" OR "Fault" OR "Error" OR "Bug" OR "Change" OR "Evolution") AND ("proneness" OR "prone" OR "predict*" OR "probability" OR "classification" OR "empirical") AND ("Ensemble" OR "Bagging" OR "Boosting"

OR “Machine learning” OR “Soft Computing” OR “Random Forest” OR “Bootstrap Aggregating” OR “Adaboost” OR “Combin*” OR “Stack*” OR “Meta*” OR “Rotation Forest” OR “Voting” OR “Logitboost”).

We conducted the search in five well-known literature sources namely ScienceDirect, ACM Digital Library, IEEEXplore, Wiley Online Library, and SpringerLink. These sources were chosen based on our previous knowledge of conducting reviews in the SDP and SCP domains [13, 24]. Moreover, most of the primary studies in previously conducted systematic reviews in SDP and SCP are indexed in these sources [10, 11]. The search string was modified suitably according to the requirements of each literature source. It examined the title, abstract, and keywords of the studies in the literature databases. The period of the search was limited from January 2000 to December 2020. We also removed the duplicate studies which were extracted from more than one source. To avoid missing a relevant study, we also scanned the reference lists of recent reviews on SDP and SCP [10, 13] and those of the already collected candidate studies. These efforts resulted in the collection of 182 relevant studies. These studies were further scrutinized using the inclusion and exclusion criteria stated in the next section.

2.2. Inclusion and exclusion criteria

Before stating the criteria for inclusion and exclusion, we first define “defect proneness” and “change proneness” attributes of a software entity. Both these attributes are dichotomous. A software entity is designated as defect-prone if a defect is likely to occur in a subsequent release of the software. Most of the studies in literature, label a class/module as defect-prone if one or more bugs have occurred in the class [3, 4]. On the other hand, a software entity is termed as change-prone if it is likely to change in a future released version of the software product. Majority of studies labeled software entities with a threshold value of one or more changes as change-prone [13]. Certain other studies in literature use “median-based” [30] or “boxplot-based partition method” [31] for labeling change-prone classes [13]. Keeping these definitions in mind we state the following criteria for inclusion and exclusion of the collected studies.

2.2.1. Inclusion criteria

- Empirical studies that use ET for SDP or SCP.
- Empirical studies that compare different ET with each other or with other non-ensemble techniques for SDP or SCP.
- Empirical studies that propose new ET for SDP or SCP.

2.2.2. Exclusion criteria

- Studies that use ET for dependent variables other than defect proneness and change proneness, such as the number of defects/changes, class stability, just in time defect prediction, bug assignment, code churn, etc.
- Studies including ET just to compare or demonstrate their proposed models/performance measures. These studies were excluded as they included ET without any discussion and did not perform or focus on their empirical evaluation.
- Similar studies that were conducted by the same authors. In case a conference paper is extended in a journal, the conference paper is excluded.

– Studies that used clustering or clustering ensembles for prediction.
– Review studies, poster papers, Ph.D. dissertations, and studies with little or no empirical analysis.
– Studies that were not written in English language.

Study inclusion and exclusion was done in two steps. We first applied the mentioned criteria on the title, abstract, and keywords. Thereafter, the remaining studies were adjudged based on their full text. After applying the above discussed criteria, we obtained 106 candidate studies.

2.3. Quality assessment

Each candidate study obtained after application of the inclusion and exclusion criteria was further subject to quality assessment. This step ensures the selection of only those studies, which are capable of effectively answering the investigated research questions. Quality assessment was done by two researchers by formulating a checklist shown in Table 1. According to the table, criteria (i) evaluates whether the study clearly states its aims, while criteria (ii) assesses whether ET and its uses have been clearly mentioned in the study. Criteria (iii) assesses whether the study mentions the base learners and aggregation mechanism used by the ET. Criteria (iv) allocates lower score to a study if it basis it's results on less than five datasets. While criteria (v) focus on selection of an appropriate validation method such as ten-fold, cross-project or others, criteria (vi) evaluates whether robust and appropriate performance measures such as Area Under the Receiver operating characteristic Curve (AUC), Balance, Mathews Correlation Coefficient (MCC) etc., have been used. Studies that base their results only on biased performance measures like accuracy are given less scores. Criteria (vii) evaluates whether models developed using ET are compared with other models, while criteria (viii) allocates a higher score to studies that have performed statistical validation of their results. Criteria (ix) checks if the study has mentioned its probable threats. Finally, Criteria (x) gives higher score to a study that add value to existing literature on ensembles at the time of its publication. As this was hard to evaluate, due to temporal aspect of relevance of the work, the authors allocated lesser score to similar studies that were published in the same year. Similar checklists were used in previous reviews [12, 13, 32]. Each of the two researchers conducting the quality assessment independently assessed each study on the ten questions stated in the checklist.

Table 1. Quality questions

(i)	Does the study state its objectives in a clear and precise manner?
(ii)	Is the use of ET and its application clearly defined?
(iii)	Are the base learners clearly stated? Are the rules/mechanisms for combining base learners to form ET clearly described?
(iv)	Is the experiment conducted on an appropriate number of datasets?
(v)	Are the models developed using ET validated appropriately using effective validation methods?
(vi)	Are the models developed using ET effectively assessed using suitable performance measures?
(vii)	Are the models developed using ET compared with models developed using other non-ensemble techniques or amongst each other?
(viii)	Do the results of the study map to its objectives? Are the results statistically validated?
(ix)	Does the study state possible threats to validity specific to the use of ET?
(x)	Does the study add value to the existing work on ensembles in SDP/SCP literature?

The questions could have three possible responses: Yes (score of +1), Partly (score of +0.5), and No (score of 0). In case the researchers disagreed on the allocated score, a discussion ensued to allocate a reasonable score. The cumulative score (CS) of the study was a sum of all the scores of the questions mentioned in the checklist. A study obtaining a score of 5 or more (50% or more) was considered to be of acceptable quality [12, 13, 32]. We rejected 29 studies on the basis of quality assessment. The remaining 77 studies were termed as primary studies.

2.4. Data extraction and synthesis

For each of the study selected after quality assessment, we extracted the relevant data to answer the RQ's. The extracted data consisted of basic details of a paper such as a title, authors, year of publication, etc. as well as details specific to the experiment that is required to answer the RQs (mentioned in Table 2). After extracting the data, we need to synthesize it to appropriately answer the investigated RQ's. Table 2 mentions the manner in which the data was analyzed and synthesized with respect to each RQ and the expected result after the synthesis and analysis.

Table 2. Data extraction and synthesis

RQ	Data extracted from primary study	Data analysis	Result
RQ1: What is the categorization of ET? Which is the most popular category of ET in SDP/SCP literature?	ET Used, Base learners of ET	Categorization of ET used on the basis of base models, i.e., their similarity, course of aggregation, cooperative or competitive relationship, means of diversity, dependency amongst themselves and the type of base learner used	Pie-charts depicting percentage of SDP and SCP studies using a specific categorization of ET, Bar chart of primary studies using different categories of base learners
RQ2: What are the various applications of ET in SDP/SCP literature?	Application or stage where ET was used	Listing and analyzing the percentage of primary studies that utilized ET for a specific stage/application while developing SDP/SCP models	Finding the most common and sporadic applications of ET while developing SDP/SCP models
RQ3: Which rules/mechanisms are used for combining base models to form ET in SDP/SCP literature?	Rules for formulating the ET used in the study	Categorizing the ET in accordance with aggregation mechanism (Weighing or Meta-learning)	A table listing the various combination rules, the various ET using the specific rules and the number of primary studies using the rule

Table 2 continued

RQ	Data Extracted from Primary Study	Data Analysis	Result
RQ4: What is the performance of ET for various tasks in the domain of SDP/SCP? How does the performance of ET compare to other non-ensemble techniques and amongst each other?	Other non-ensemble techniques used, datasets used, validation method used, performance measures used (such as accuracy, recall, AUC), dataset-wise value of performance measures for the developed SPD/SCP model in the study	Computing boxplots and various descriptive statistics of the extracted performance measure values, Dataset-wise comparison (vote count method) and statistical analysis using Wilcoxon signed-rank test of ET with other non-ensemble techniques based on their application in SDP/SCP domain, Pairwise comparisons using wilcoxon test amongst ET based on their application	Evaluation of performance of ET based on computed statistics and boxplots, Graphs representing the comparative performance of ET with other non-ensemble techniques, Tables representing ET comparison amongst themselves, Superior ET for specific applications
RQ5: What are the various reported threats to validity specific to the use of ET in SDP/SCP literature?	Threats specific to use of ET (only extracted from studies containing "Threats to Validity" or "Limitations" section)	Listing and categorizing (Construct/External/Internal) various threats specific to the scenario when ET were used in SDP/SCP	Recommendations to researchers for planning future studies that minimize the commonly found threats.

3. Results

This section discusses the review results. We first state the overview of the selected studies followed by an analysis to answer the various investigated RQ's. We also discuss and analyze the review results to determine research gaps. This aids in proposing directions for future work in the domain.

3.1. Overview of primary studies

The various steps followed to collect the primary studies have already been mentioned in Sections 2.1–2.3. Figure 1 states the number of studies collected in each step from the various sources (A–F). The year-wise distribution of primary studies is depicted in Figure 2. It may be seen from the figure that there has been a consistent increase in the number of studies using ET in recent years (2016–2020).

Table 3 states all the 77 primary studies along with their study identifier (SI) and cumulative quality score (CS). ES12, ES37, and ES40 are top-scoring primary studies with a CS of 9.5. Amongst the primary studies, the most popularly cited study in accordance

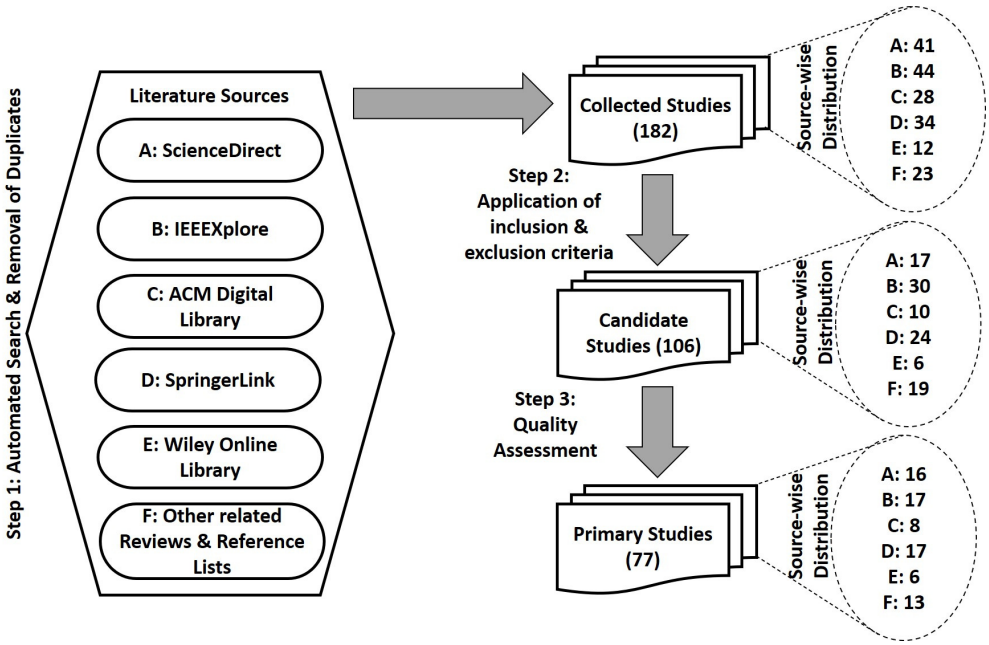

Figure 1. Primary studies collection and source-wise distribution

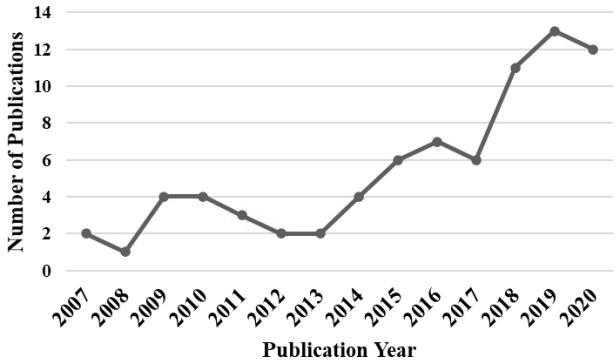

Figure 2. Year-wise distribution of primary studies

with citation count normalized with respect to year was ES3, followed by ES18, ES8, ES26, and ES35. We also classified the primary studies based on their publication venue. 33% of the primary studies were published in conferences, while 63% of the studies were published in various reputed journals. Three studies were published as chapters. The most popular publication venues were "Information and Software Technology" journal and "Software Quality" journal with six and five studies, respectively. Thereafter, "IEEE Transactions on Software Engineering" was the source of four primary studies. No conference was the source of more than one primary study. It was also noted that 67 primary studies used ensembles for SDP, while only 10 primary studies investigated the use of ensembles for SCP. A similar trend was also observed if we accounted for the rejected studies. Out of the 29 rejected studies, only four studies developed SCP models, all other rejected studies developed SDP models.

Table 3. Primary studies with quality score

SI	Study Name	CS	SI	Study Name	CS	SI	Study Name	CS
ES1	Jiang et al. 2007 [33]	5.5	ES27	Rubinic et al. 2015 [34]	5.5	ES53	Ali et al. 2019 [35]	7.5
ES2	Ma et al. 2007 [36]	7	ES28	Siers and Islam 2015 [37]	6.5	ES54	Campos et. al. 2019 [38]	6.5
ES3	Lessmann et al. 2008 [3]	8	ES29	Li and Wang 2016 [39]	6.5	ES55	Catolino and Ferrucci 2019 [30]	8.5
ES4	Jia et al. 2009 [40]	5.5	ES30	Malhotra 2016 [41]	8	ES56	Gong et al. 2019 [42]	8.5
ES5	Khoshgoftaar et al. 2009 [43]	6	ES31	Ryu et al. 2016 [44]	8.5	ES57	He et al. 2019 [45]	8.5
ES6	Mende and Koschke 2009 [46]	6.5	ES32	Petric et al. 2016 [47]	9	ES58	Kumar et al. 2019 [48]	6
ES7	Seiffert et al. 2009 [49]	8	ES33	Wang et al. 2016a [50]	8.5	ES59	Li et al. 2019a [51]	10
ES8	Arisholm et al. 2010 [52]	6.5	ES34	Wang et al. 2016b [53]	8.5	ES60	Li et al. 2019b [54]	6
ES9	Liu et al. 2010 [55]	8	ES35	Xia et al. 2016 [56]	8.5	ES61	Malhotra and Kamal 2019 [57]	6
ES10	Zheng 2010 [58]	6	ES36	Alsawalqah et al. 2017 [59]	6	ES62	Malhotra and Khanna 2019b [60]	9
ES11	Seliya et al. 2010 [4]	8.5	ES37	Di Nucci et al. 2017 [61]	9.5	ES63	Tong et. al. 2019 [62]	9
ES12	Misirh et al. 2011 [63]	9.5	ES38	Kumar et al. 2017 [64]	7	ES64	Tran et al. 2019 [65]	6.5
ES13	Peng et al. 2011 [66]	6	ES39	Malhotra and Khanna 2017b [67]	7	ES65	Zhou et. al. 2019[68]	9
ES14	Seliya and Khoshgoftaar 2011 [69]	8.5	ES40	Ryu et al. 2017[70]	9.5	ES66	Abbas et al. 2020 [71]	6
ES15	Gao et al. 2012 [72]	5	ES41	Yohannese et al. 2017 [73]	6	ES67	Aljamaan and Alazba 2020 [74]	8.5
ES16	Sun et al. 2012 [75]	8.5	ES42	Agarwal and Singh 2018 [76]	5.5	ES68	Ansari et al. 2020 [77]	7
ES17	Wang et al. 2013 [78]	7.5	ES43	Bowes et al. 2018 [79]	7.5	ES69	Banga and Bansal 2020 [80]	5.5
ES18	Wang and Yao 2013 [81]	8	ES44	Chen et al. 2018 [82]	8.5	ES70	Elahi et al. 2020 [83]	7.5
ES19	Kaur and Kaur 2014 [84]	8.5	ES45	El-Shorbagy et al. 2018 [85]	6	ES71	Goel et al. 2020 [86]	5.5
ES20	Panichella et al. 2014 [87]	9	ES46	Malhotra and Bansal 2018 [88]	8	ES72	Khuat and Le 2020 [89]	7.5
ES21	Rodriguez et al. 2014 [90]	8	ES47	Malhotra and Khanna 2018a [17]	8.5	ES73	Malhotra and Jain 2020 [91]	6
ES22	Suma et al. 2014 [92]	5.5	ES48	Mousavi et al. 2018 [93]	7.5	ES74	Pandey et al. 2020 [94]	9
ES23	Chen et al. 2015 [95]	9	ES49	Moustafa et al. 2018 [96]	6	ES75	Rathore and Kumar 2020 [97]	9
ES24	Elish et al. 2015 [98]	6.5	ES50	Tong et al. 2018 [99]	8.5	ES76	Saifan and Abuwaridh 2020 [100]	7.5
ES25	Hussain et al. 2015 [101]	7	ES51	Zhang et al. 2018 [102]	7.5	ES77	Yucular et al. 2020 [103]	9
ES26	Laradji et al. 2015 [104]	7.5	ES52	Zhu et al. 2018 [31]	8.5			

3.2. Categorization of ET

We first list the various ET used in primary studies along with the study identifier using the specific ET (Appendix, Table A1). An analysis of the Appendix indicates that the most popular ET were Random Forests (RF), Bagging (BAG), AdaBoost (AB) and Voting amongst Heterogenous Base Learners (VHetBL) used in 42%, 32%, 32% and 23% of primary studies, respectively. Thereafter, we categorize the various ET according to five different criteria [105–107] on the basis of base models as shown in Figure 3. We also evaluated the percentage of SDP and SCP primary studies according to the ET used by them corresponding to various categories (Figures 4 and 5). The various categorizations are explained as follows:

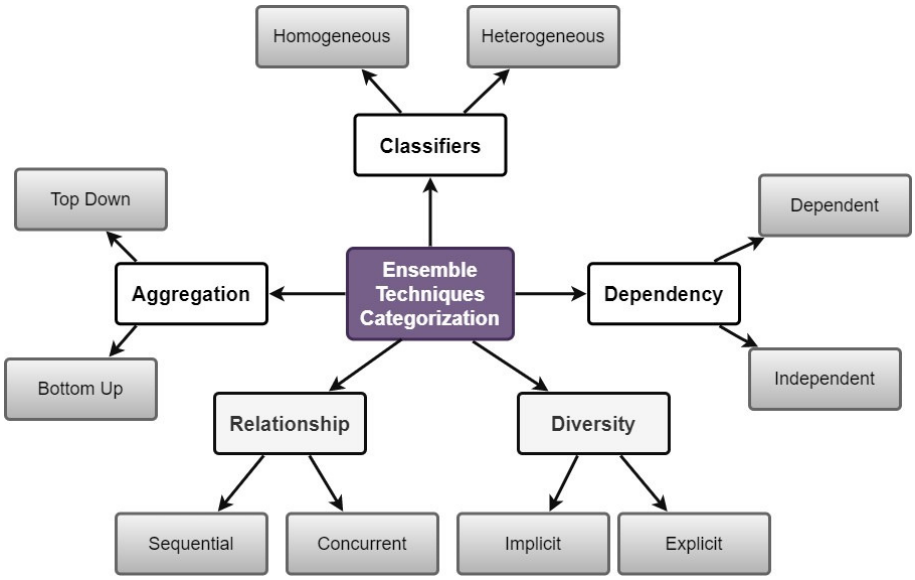

Figure 3. Categorization of ET

1. *Similarity of base models*: This categorization indicates the similarity of learners used to construct the base models in an ensemble (homogeneous or heterogeneous). A homogeneous ensemble combines base models developed using the same data analysis technique. Some examples of the homogeneous ensemble include RF, BAG, LB, AB, Rotation Forest (ROT), Dagging (Dag), Random Subspace (RS), MultiBoost (MBoost), DECORATE, Logit Model Trees (LMT), Double Transfer Boosting (DTB), Adaptive Selection of Optimum Fitness (ASOF), Multiple Kernel Ensemble Learning (MKEL), various other cost-sensitive ensembles like AdaCost, MetaCost (MC), SMOTEBoost (SMBoost), AdaBoost.NC (BNC), Voting amongst Homogeneous Base Learners (VHomBL), etc. On the other hand, a heterogeneous ensemble combines base models developed using diverse data analysis techniques. Examples of heterogeneous ET used in primary studies are voting based ensembles with varied learners for base models, Stacking, Two Stage Ensemble (TSE), Non-Linear Decision Tree Forest (NDTF), Best Training Ensemble (BTE), Combined Defect Predictor (CODEP), Adaptive Selection of Classifiers (ASCI), etc. It may be noted that the Validation and Voting (VV) ensemble is homogeneous in ES9 but heterogeneous in ES36. Also, Omni Ensemble Learning (OEL) used by ES48 is a special category of ET which combines the concept of both homogeneity and heterogeneity. It uses bagging approach along with random oversampling for

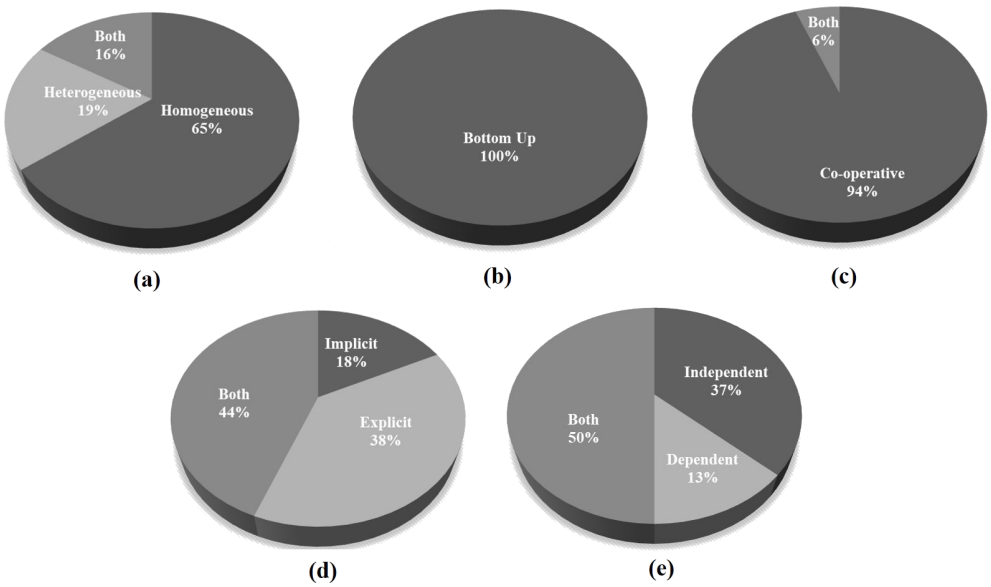

Figure 4. Percentage of SDP primary studies in each category according to: (a) learner similarity, (b) aggregation, (c) relationship, (d) diversity, and (e) dependency categorizations

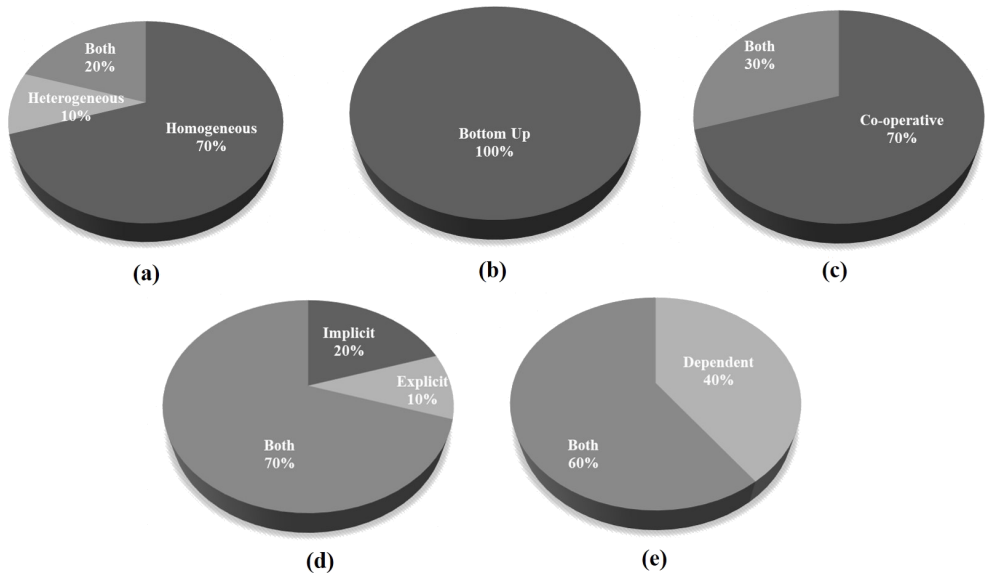

Figure 5. Percentage of SCP primary studies in each category according to: (a) learner similarity, (b) aggregation, (c) relationship, (d) diversity, and (e) dependency categorizations

handling the class imbalance issue. This can be categorized as homogeneous. However, in the next stage for SDP, genetic algorithm is used for ensemble selection amongst 34 different base classifiers. This stage of the OEL is heterogeneous. According to Figure 4a, 65% of the SDP primary studies used homogeneous ET, while 19% of the studies used heterogeneous ET. The other remaining 16% used both heterogeneous and homogeneous ET. On the other hand, as depicted in Figure 5a, 70% of the SCP primary studies used homogeneous ET, 10% used only heterogeneous ET, while 20% used both homogeneous and heterogeneous categories of ET.

2. *Aggregation of base models*: This categorization indicates whether the ensemble aggregation is top-down or bottom-up. A top-down ensemble combines its base models without taking into account the outputs generated by them. On the other hand, in a bottom-up ensemble, the outputs of base models are crucial for aggregating them [108]. Ensembles that use voting or stacking are a subset of bottom-up ensembles [105]. All the ET used in primary studies (both SDP and SCP) are bottom-up ensembles as depicted in figures (Figure 4b and 5b).
3. *Relationship amongst base models*: This categorization suggests the relationship amongst different base models for producing the ensemble output. The base models of an ET could be competitive or cooperative. An ensemble is termed as competitive if only one of the constituent base model is selected to produce the final output [105]. Examples of such ET used in primary studies are ASOF, ASCI, BTE, NDTF, Omni Ensemble Learning, and Multischeme. A co-operative ensemble is the one in which the output of all the constituent base models is combined to produce the final output [106]. All other ET mentioned in Table A2 (Appendix) such as RF, AB, LB, CODEP, BAG, etc. are co-operative ensembles. As shown in Figures 4c and 5c, 94% and 70% of the primary studies used co-operative ET in SDP and SCP, respectively, while 6% of the SDP primary studies and 30% of the SCP primary studies used both co-operative and competitive ET. There was no primary study that used only competitive ET.
4. *Diversity of base models*: This categorization dictates the means for the diversity of base models, i.e., implicit or explicit. Implicit ET employs mechanisms for assuring that the constituent models are diverse [106]. These mechanisms could be random subsamples of the training data or random selection of features etc. Implicit techniques do not measure if diversity is introduced or not. Examples of implicit ET are RF, BAG, SysFor, Multischeme, MC, ROT, RS, Dag, Cost-sensitive Forest, MBoost, Roughly Balanced Bagging (RBBag), Balanced RF, Sampling based Online Bagging (SOB), Oversampling based Online Bagging (OOB) and Undersampling based Online Bagging (UOB). Explicit ET employs a measurement to ensure that the constituent models are different from each other [106]. Examples include AB, LB, DECORATE, AdaCost, etc. Explicit ensembles may also use different base learners to ensure diversity among base learners such as in ASCI, BIT, CODEP, TSE or may use the same learner but with a significant difference in basic configurations (such as different kernels or different fitness variant) as in MKEL and ASOF. Figure 4d depicts 38% of the SDP primary studies used explicit ET, 18% used implicit ET, and the other studies (44%) evaluated both implicit and explicit ET. Amongst the primary studies that use ET in SCP (Figure 5d), majority of the studies (70%) investigated both implicit and explicit ET.
5. *Dependency amongst base models*: This categorization dictates how the various base models interact with each other (dependent or independent). In dependent ET the various base models or consequent iterations of an ensemble interact with each other. A base model constructed later may benefit from the guidance provided by a base model (iteration) constructed earlier [107]. Some of the dependent ET used in primary studies are AB, LB, MKEL, AdaCost, DECORATE, TransferBoost, SMBoost, and DTB. On the other hand, in an independent ET, several base models are constructed in parallel, which are independent of the other base models (iterations). BTE, NDTF, CODEP, Stacking, RF, BAG, MC, ROT, VHomBC, and VHetBC are examples of independent ET used in primary studies [107]. It was observed (Figure 4e) that 37% of the SDP primary studies used independent ET, 13% used dependent ET and 50% investigated both dependent and independent ET. However, there was no SCP primary

study (Figure 5e) that investigated the use of only independent ET, 40% of the studies used dependent ET, while the other 60% used both independent and dependent category of ET.

It may be noted that Coding based Multiclassifier (CEL) proposed by ES15 was a very different type of ET and could not be categorized into the discussed categories.

We also assessed the various categories of base learners used by the ET in each primary study. The base learners were categorized into various families as suggested by [10, 13, 43]. These categories were tree-based learners, support vector machine, Bayesian learners, rule-based learners, instance-based learners, search-based algorithms, artificial neural networks, ensemble learners, logistic regression, and other miscellaneous learners. The base learners which were included in each category are mentioned in Appendix (Table A2). While analyzing the data for base learners we found that 14 primary studies did not mention the base learners used by them. Figure 6 depicts the number of the remaining 63 primary studies which use base learners from a specific family. According to the figure, tree-based learners are the most popular category used by 70% of the studies (both SDP and SCP). Thereafter, Logistic Regression (LR) was used by 56% of the studies. Also, Bayesian learners were used as base learners in 42% of the primary studies. It was interesting to note that ET were themselves used as base learners for constructing other ET in 42% of the studies. We term such techniques as an ensemble of ensembles. Rule-based learners and instance-based learners were less popular as base learners of ET (used in 20% and 30% of studies, respectively). Search based algorithms and miscellaneous learners were the least popular categories as they were each used by only 14% and 19% of the studies, respectively.

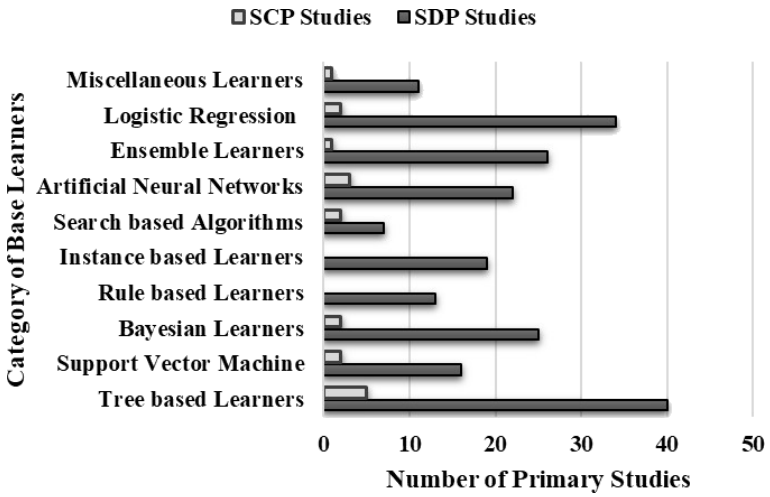

Figure 6. Number of primary studies with specific category of base learners

3.3. Application of ET

All the studies collected in the review develop either SDP or SCP models using ET. However, there are multiple factors that make the task of developing SPD/SCP model difficult and challenging. ET in the collected studies were not only used for model development but for also handling these other critical factors which include existence of large number of features, lack of defect-prone or change-prone instances making the training data imbalanced, evaluating prediction in realistic online scenarios or unavailability of appropriate training

data. We investigated the primary studies to ascertain the various applications of ET, i.e., what was the underlying use of ET in SDP/SCP. The various applications are listed as under along with the percentage of primary studies that utilized the ET for the particular application.

- As a learning algorithm for developing the SDP model (65%).
- As a learning algorithm for developing the SCP model (13%).
- Addressing the class imbalance issue (37%).
- Transfer learning (10%).
- Online Learning (3%).
- Feature selection (1%).

As indicated above, the majority of the studies (65%) used ET as learners for developing SDP models. On the other hand, only 13% of the primary techniques used ET as learning algorithms for developing SCP models. ET used for these two applications included RF, LMT, AB, BAG, LB, DECORATE, VHetBL, VHomBL, VV, ROT, CODEP, Stacking, NDTF, BTE, RS, Dag, XGBoost, ASCI, ASOF, etc. It may be noted that 85% of the primary studies developed within-project models for SDP/SCP using validation techniques such as hold-out validation, k-fold cross-validation, or inter-release validation. However, there may be a scenario where previous data related to the same project may be unavailable or difficult to collect [52]. For such a situation, researchers suggest the use of cross-project models [86, 87, 102]. A critical issue in developing these models is that varied projects may have different data distributions or different metric sets. To overcome such issues, a transfer learning mechanism, which derives common observations and expertise from the available projects and transfers it to the target project [56, 70] is proposed by the research community. This is a relatively recent application of ET as the first primary study which used ET for transfer learning was published in 2014 (ES32). As mentioned above, 10% of the primary studies used ET for transfer learning. These ET were TransferBoost, Improved Transfer Adaptive Boosting (ITrAdaBoost), Kernel Spectral Embedding Transfer Ensemble (KSETE), TSE, Value Cognitive Boosting with Support Vector Machine (VCB-SVM), DTB, VHetBL, and TransferCostSensitive Boosting (TCSBoost).

Another critical issue while developing SDP/SCP models is the presence of imbalanced training data [20, 57, 67, 109]. In general, a standard classifier assumes that each class is present in equal proportion, i.e., there is an equal number of defect-prone/change-prone and not defect prone/not change prone classes in a dataset. This assumption hinders the development of an effective SDP/SCP model as the class distributions in actual datasets are biased. This results in erroneous identification of the minority class instances. Therefore, a popular application of ET in the primary studies was using them for addressing the class imbalance issue (37% of primary studies). As proposed by Galar et al. [20], we categorized the ET used for class imbalance in primary studies into Cost-sensitive ET, Boosting-based ET, Bagging-based ET, and Hybrid ET. Boosting based and Bagging based ensembles combine Bagging and Boosting with data preprocessing techniques such as random undersampling, SMOTE, oversampling, etc. Hybrid ensembles combine both bagging and boosting techniques along with data preprocessing techniques. Furthermore, we also mention a category of ET namely "Novel" ET, which are proposed by primary studies but could not be categorized into the above categories.

- *Cost-Sensitive ET*: MetaCost, AdaCost, Csb2, Adc2, Dynamic Adaboost.NC (DNC), Adaboost.NC (BNC), Cost-Sensitive Forest, Cost-sensitive Boosting Neural Networks, MKEL, TCSBoost.

- *Boosting based ET*: SMBoost, RUSBoost, WeightedSmoteBoost, SelectRUSBoost, Non-negative Sparse based Semiboost, DataBoost, MSMOTEBoost.
- *Bagging based ET*: SOB, OOB, UOB, RBBag, OEL.
- *Hybrid ET*: MBoost, Ensemble Random Undersampling
- *Novel ET*: CEL, KSETE, TSE, Bug Prediction using Deep representation and Ensemble learning.

It may be noted that there were five studies (ES32, ES40, ES56, ES59, ES63) which proposed ET (VCB-SVM, TCSBoost, ITrAdaBoost, TSE, KSETE) dealing with both the important issues, i.e., handling class imbalance and transfer learning for cross-project defect prediction. However, only one study ES14 proposed the SelectRUSBoost technique, which incorporates feature selection and class imbalance learning. Feature selection involves choosing a subset of features (independent variables) that develops SDP/SCP models with good predictive capability. Two studies (ES17 and ES54) used ET for online learning. ES54 used ET for online failure prediction, where past data is correlated with the existing state of the system to predict the occurrence of faults in near future. The study by Wang et al. [78], i.e., ES17, deals with the scenario where data continuously arrives in streams, and the training data is constantly updated with new data. There are multiple runs of time sensitive prediction which leads to better prediction models that are less biased [110]. ES17 also addressed the issue of class imbalance and proposed the SOB technique. As there are very few studies that propose comprehensive ET which deal with multiple issues simultaneously, more such techniques should be proposed and validated in future studies.

3.4. Rules/mechanisms for combining base models

As ET are a combination of different base models, this RQ concerns itself with the mechanism of aggregating the outputs of constituent base models. We found that the base models were combined either by giving appropriate weights to the output of constituent base models or through the process of meta-learning. The construction of a meta-learning ensemble generally involves multiple learning stages. The outputs of constituent base models act as inputs to the meta-learner, which is responsible for producing the final ensemble output. The ET which use meta-learning as a combination mechanism were ASCI, ASOF, MKEL, MC, Ensemble Selection, Grading, OEL, Stacking, CODEP, GcForest, DeepForest and NDTF. All other ET used the weighing mechanism for combining the base models. However, there were two exceptions, which we could not categorize properly into weighing or meta-learning mechanisms. These were: a) CEL (ES16), which used a specific coding mechanism for aggregation, and b) LMT, though many LB iterations were performed in LMT, there was only one resultant tree at the end. Table 4 lists the various combination rules used by the ET, the number of ET, and the number of primary studies using the specific combination rule. According to the Table, the most popular combination rule was “majority voting” amongst the base models, which was used in 78% of the primary studies. The ET which used this rule were BAG, RF, UOB, OOB, SOB, RS, SysFor, Balanced RF, VHetBL, VHomBL, VV, Extra Trees and Dag. The next popular combination rule was “weighing based on misclassification error of the base model” used in 56% of the primary studies. This is a combination rule generally used by several ET using the Boosting mechanism such as AB, LB, VCB-SVM, SMBoost, WeightedSmoteBoost, SelectRUSBoost, MBoost, etc. and some others like MKEL. Another popular combination rule was “Average Probability”, which was used in 14 primary studies.

Table 4. Combination rules

Combination rule	Number of ET	Number of primary studies
Average Probability	9	14
Selection of Best	2	4
Majority Vote	13	60
Maximum Confidence Score	1	1
Weighing based on misclassification error and cost adjustment	4	4
Weighing based on misclassification error of the base model	22	45
Weighted adjustment based on misclassification error and penalty for ambiguity	2	4
Weighing based on MCC obtained by the base model	1	1
Weighing based on data distribution of target data, misclassification error, and cost adjustment	2	2
Weighing based on error on the prediction of instances in the training target data	1	2
Voting based on cost-sensitive labeling of records	1	1
Weighing of base models that lead to objective function (inconsistency between labels and similarities) minimization	1	1
Weighted adjusted probabilities and probability adjustment	1	3
Weighing based on predictive performance on other data	1	2
Weighing based on the predictive capability to predict hard instances	1	2
Weighing based on predictive performance on other data and ability to predict hard instances	1	2

3.5. Performance of ET

It is crucial to evaluate the effectiveness of ET for SDP and SCP. To do so, we analyze the performance measures of the developed SDP and SCP models by various ET in the primary studies. An analysis of the performance measures used in the primary studies indicates AUC as the most widely used performance measure (65%) in these studies. Moreover, the use of AUC for assessing the performance of predictive models has also been propagated in literature studies as it is capable of handling skewed datasets with disparate class distributions and the dissimilar cost of various classification errors [67, 111]. AUC is computed by plotting recall and 1-specificity on the y-axis and x-axis, respectively and estimating the area under the plotted curve. Apart from AUC, Recall and F-measure performance measures were found to be popularly used in the primary studies. While recall states the percentage of correctly identified defect-prone/change-prone classes, it gives us no insight into the number of incorrectly identified defect-prone/change-prone classes. However, this measure has been used in many previous review studies [10, 13, 23, 24, 27] to assess the performance of the developed SDP/SCP models. Thus, we include it for assessing the predictive capability of ET. On the other hand, F-measure, which is computed as the harmonic mean of precision and recall has not been included in our analysis. Menzies et al. [112] have criticized the use of precision due to its unstable nature, thus, raising

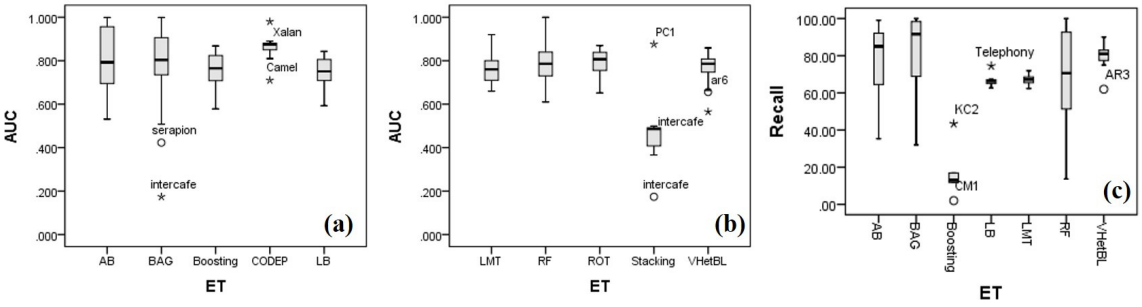

Figure 7. Dataset-wise boxplots when ET is used as a learning algorithm for developing SDP models (a), (b) AUC (c), recall

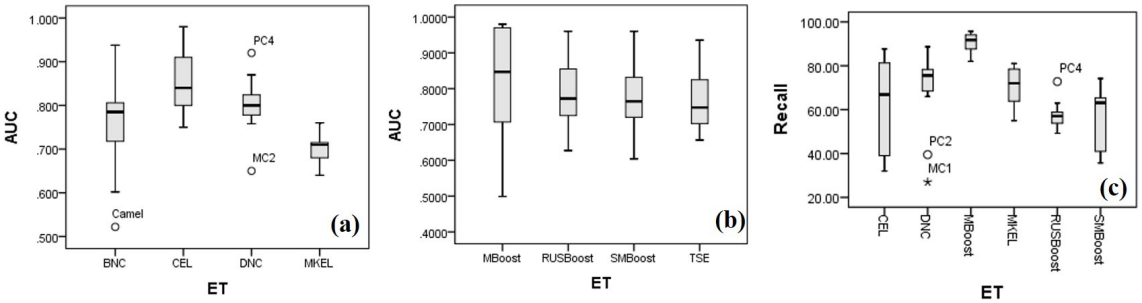

Figure 8. Dataset-wise boxplots for ET that handle class-imbalance issue (a), (b) AUC (c), recall

concerns over use of F-measure as a performance evaluator [10]. He and Garcia [109] have also doubted the capability of F-measure while "comparing the performance of different classifiers over a range of sample distributions". Therefore, we analyze two performance measures, AUC, and Recall for determining the predictive performance of ET.

The performance of ET was analyzed dataset wise. Similar to [10], we found that the most popular datasets used in SDP were NASA datasets (CM1, JM1, KC1, KC2, KC3, KC4, MC1, MC2, MW1, PC1, PC2, PC3, PC4, PC5) and Promise repository datasets such as AR5, AR6, Jedit (http://promise.site.uottawa.ca/SERepository/datasets-page.html). Some other open-source datasets (Gate, Intercafe, Lucene, Xalan, Tomcat, Synapse, Velocity, etc.) and application package datasets (Bluetooth, Contacts, Email, Calendar, Telephony, etc.) from the Android operating system were also used by primary studies for SDP. Similar to SDP, various open-source software datasets were used by SCP studies such as ArgoUML, FreeMind, Eclipse, Ant, Lucene, Gate, KolMafia, etc. and datasets from the Android operating system.

We analyze the performance of only those ET whose performance measures (AUC and Recall) could be extracted from at least two or more studies and have been validated on at least three or more datasets. This was done to yield generalized results and comparisons across studies. The performance of ET was assessed with respect to datasets and outlier values (Figs 7–9) were disposed off. Thereafter, various statistics were reported. By using such rules, an ET that might have shown exceptional performance in just one study or on specific datasets will not be designated as a good performer in the SDP/SCP domain. As ET have several parameter values such as the number of base models, different base learners, etc., their performance values may vary a lot. However, we are interested in finding the best values of ET. Thus, we use the following rules while extracting the values of performance measures:

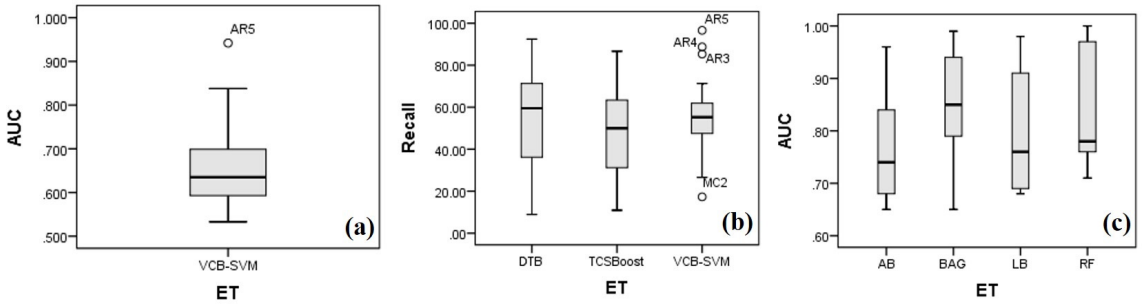

Figure 9. (a), (b) Dataset wise AUC and recall boxplots for ET used for transfer learning, (c) dataset wise AUC boxplot for ET used as a learning algorithm for developing SCP models

– If an ET has been evaluated on the same dataset, more than once in a study (maybe with different internal parameter settings), we report the highest values.
– If two studies have evaluated an ET on the same dataset, we again report the highest values amongst the two studies for the particular dataset.

The rules for extracting the performance measures are similar to the ones used by [13, 24].

3.5.1. Assessment of performance statistics of ET

We grouped the ET according to the various applications they were used for (Section 3.3). Figures 7–9 depict the boxplots of the ET for AUC and Recall performance measures. The outliers are labeled with their corresponding dataset names. As some ET were used for more than one application, we segregated the studies according to the application the ET was used for to construct the boxplots. For instance, AB, BAG, LB, and RF were used both as learning algorithms for developing SDP and SCP models. Figures 7a and 7b depicts the boxplots when they were used for developing SDP models and Figure 9c depicts the boxplot when they were used for developing SCP models. TCSBoost has two applications (addressing class imbalance and transfer learning). However, we include it in the Transfer learning application for simplicity. The performance measure values for VHetBL could be extracted for two applications: as a learning algorithm for developing SDP models and for transfer learning but since we could extract the values for transfer learning application in only one study for VHetBL, we only reported the values for its application as a learning algorithm for developing SDP models. Similarly, the performance measures values for TSE could be extracted for two applications, i.e., for transfer learning and for addressing the class imbalance issue. But since only one study reported values for the transfer learning application, we include TSE as an algorithm for addressing the class imbalance issue.

Table 5 reports the AUC and Recall statistics of ET for various applications. These statistics were computed after removing the outliers depicted in Figures 7–9. The table reports the minimum, maximum, mean, median, standard deviation, and the count of the number of datasets from which the statistics are computed for each ET. According to the table, apart from Stacking and MKEL all ET depicted a mean AUC score of 0.75 or above for three of the discussed applications namely as a learning algorithm for developing SDP and SCP models and for addressing the class imbalance issue. This indicates the favorability of ET for these applications. Though, a little lower, but ET for transfer learning showed AUC mean values of 0.65 indicating they could be effective for it. However, we could analyze the AUC values for just one ET in the transfer learning domain. CODEP,

Table 5. Performance measure statistics of ET for various applications

ET	Performance measure	Dataset count	Minimum	Maximum	Mean	Median	Standard deviation
Learning algorithm for developing SDP models							
AB	AUC	40	0.53	0.99	0.81	0.79	0.13
	Recall	20	35.34	99.00	78.63	85.00	16.43
BAG	AUC	42	0.51	0.99	0.81	0.81	0.11
	Recall	24	32.00	100.00	84.16	91.65	17.40
Boosting	AUC	18	0.58	0.87	0.76	0.77	0.07
	Recall	3	11.80	16.90	13.90	13.00	2.17
CODEP	AUC	8	0.81	0.89	0.87	0.88	0.02
LB	AUC	28	0.60	0.84	0.75	0.75	0.07
	Recall	6	62.70	67.22	65.54	66.09	1.61
LMT	AUC	17	0.66	0.92	0.76	0.76	0.07
	Recall	7	62.30	71.95	67.05	67.14	2.95
RF	AUC	55	0.61	1.00	0.80	0.79	0.10
	Recall	46	13.70	100.00	68.06	70.60	26.41
ROT	AUC	20	0.65	0.87	0.79	0.81	0.06
Stacking	AUC	13	0.37	0.50	0.46	0.49	0.05
VHetBL	AUC	14	0.66	0.86	0.78	0.79	0.05
	Recall	6	75.00	90.00	82.00	81.00	4.61
Handling class imbalance issue							
BNC	AUC	12	0.60	0.94	0.78	0.79	0.08
CEL	AUC	17	0.75	0.98	0.86	0.84	0.07
	Recall	14	32.00	87.67	61.21	66.86	20.69
DNC	AUC	9	0.76	0.87	0.81	0.80	0.03
	Recall	10	66.00	88.70	76.71	76.42	5.78
MKEL	AUC	8	0.64	0.77	0.71	0.71	0.04
	Recall	12	55.00	81.00	70.59	72.07	8.93
MBOOST	AUC	18	0.50	0.98	0.83	0.85	0.13
	Recall	8	82.05	95.74	90.59	91.73	4.56
RUSBoost	AUC	16	0.63	0.96	0.77	0.79	0.09
	Recall	8	49.20	63.00	55.98	56.35	4.11
SMBoost	AUC	16	0.60	0.96	0.78	0.77	0.09
	Recall	9	35.70	74.20	55.53	63.10	13.44
TSE	AUC	12	0.66	0.94	0.76	0.75	0.08
Transfer learning							
DTB	Recall	33	8.90	92.40	55.56	59.50	21.61
TCSBoost	Recall	33	10.90	86.60	48.05	50.00	21.73
VCB-SVM	AUC	33	0.53	0.84	0.65	0.63	0.08
	Recall	30	26.60	71.30	53.23	54.75	9.53
A learning algorithm for developing SCP models							
AB	AUC	13	0.65	0.96	0.77	0.74	0.09
BAG	AUC	15	0.65	0.99	0.85	0.85	0.09
LB	AUC	13	0.68	0.98	0.79	0.76	0.11
RF	AUC	13	0.71	1.00	0.85	0.78	0.11

BAG and ROT depicted the best median AUC values 0.88, 0.81 and 0.81, respectively for their use as a learning algorithm for developing SDP models. BAG also attained the best median AUC value (0.85) for its use as a learning algorithm for SCP models. MBoost depicted the best median AUC values (0.85) for handling imbalanced datasets.

An analysis of the mean Recall values stated in Table 4 indicates that the values range from 60%–90% in the majority of the cases except for Boosting and some of its variants such as DTB, RUSBoost, SMBoost, VCB-SVM, and TCSBoost. These ET depicted poor recall values which were in the range of 48%–55%. On the other hand, other ET based on boosting mechanism such as AB and MBoost depicted exceptionally good median recall values (AB: 85.00%, MBoost: 91.73%). Thus, future studies should continue to explore and validate various ET based on the boosting mechanism. The statistics of the AUC and Recall values reported in Table 5 confirm the capability of ET for the various discussed applications. Moreover, the use of ET should be encouraged for these applications.

3.5.2. Comparative performance of ET with other techniques

To ascertain the effectiveness of ET, it is important to compare them with other non-ensemble techniques. The rules for extracting data for comparison were similar to the ones stated in Section 3.5.1. As discussed previously, the comparison was done with respect to the application the ET is used for. In order to perform the comparison, we used only the AUC performance measure due to its robustness and stability. We do not compare the techniques based on recall values as analyzing recall may not give a comprehensive picture in the case of imbalanced datasets [67] and since our comparison is dataset wise we should select the performance criteria for comparison wisely. It may be noted that we could extract relevant data for comparing only two applications of the ET, i.e., as a learning algorithm for developing SDP models and as a learning algorithm for developing SCP models.

We compared 10 ET (AB, BAG, LB, LMT, RF, Boosting, CODEP, ROT, Stacking, VHetBL) with 10 non-ensemble techniques (Artificial Neural Network (ANN), Bayesian Network (BN), Decision Tree C4.5, Classification and Regression Tree (CART), Decision Table (Dec.T), K-Nearest Neighbor (KNN), LR, Naïve Bayes (NB), Support Vector Machine (SVM), Voting Feature Intervals (VFI)) to assess their capability as a learning algorithm for developing SDP models. The non-ensemble techniques that were chosen for comparison were based on two criteria: a) the data for comparison could be extracted so that ET could be compared on at least 3 or more datasets, which were used in at least 2 or more studies b) they should represent the various categories of learners as depicted in Figure 6. The chosen non-ensemble techniques were representative of support vector machine, i.e., SVM, artificial neural networks, i.e., ANN, tree-based learners (C4.5 and CART), Bayesian learners (NB and BN), rule-based learner, i.e., Dec.T, instance-based learner, i.e., KNN, statistical learner, i.e., LR and miscellaneous learner, i.e., VFI. The comparison was performed using vote count method (dataset wise), i.e., we computed the number of datasets (votes) on which a specific ET is better than a specific non-ensemble technique and the number of datasets (votes) on which a specific ET is worse than a specific non-ensemble technique in terms of AUC value. These results are depicted in Figure 10. For instance, in Figure 10a, the AUC value of AB was better than BN in 17 datasets, while the AUC value of AB was worse than BN in 9 datasets. This means 17 votes favor AB and 9 votes are against AB, when compared with BN. Similarly, in Figure 10b, the AUC value of Boosting was better than C4.5 by 12 votes and there were 6 votes against Boosting when compared with C4.5.

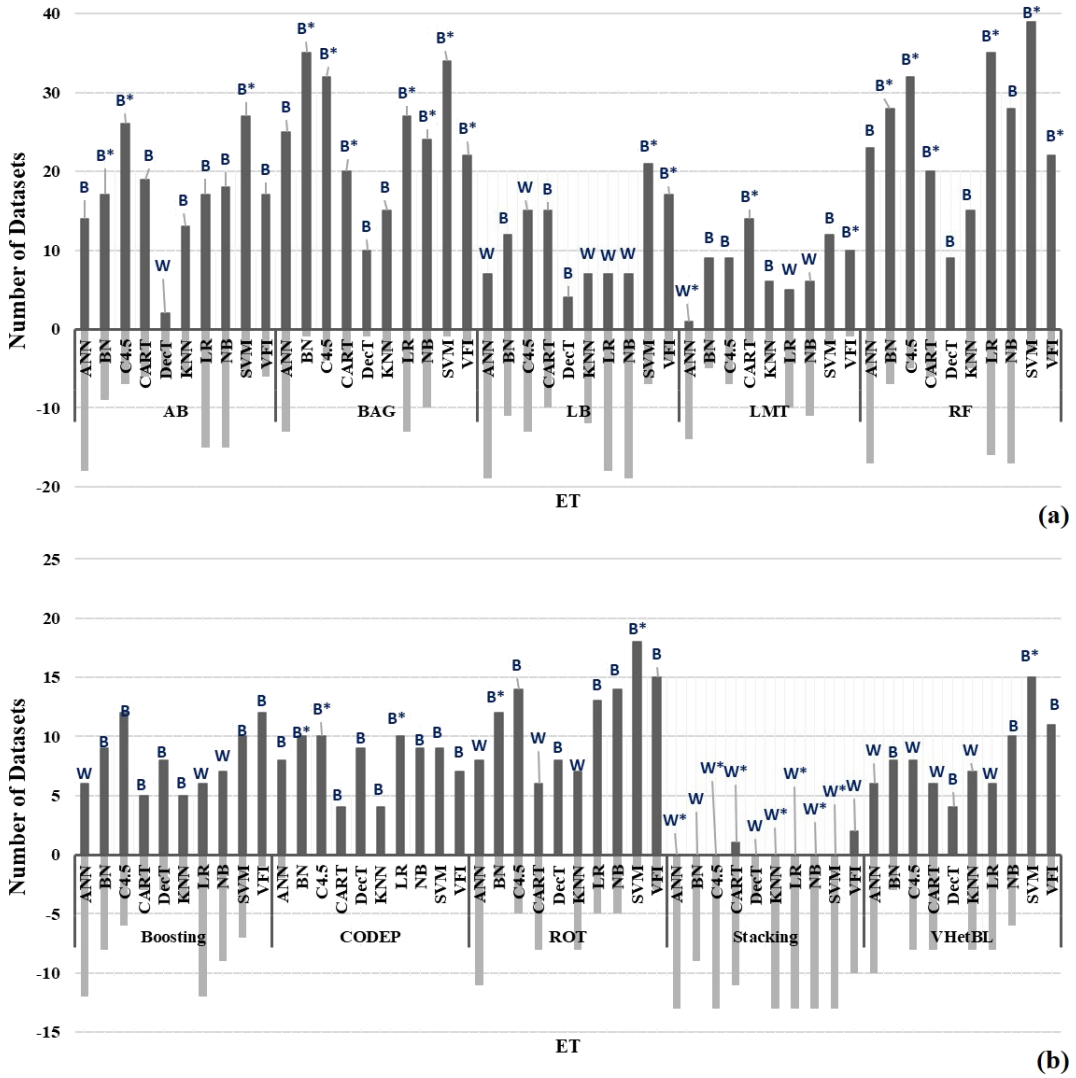

Figure 10. Comparison results of ET with non-ensemble techniques when they are used as a learning algorithm for developing SDP models

Apart from comparing the results dataset wise, we also performed a statistical analysis of the comparison between an ET and the various chosen non-ensemble techniques. Wilcoxon signed-rank test with Bonferroni correction was conducted to pairwise compare the AUC values of an ET and the various other compared techniques. Though, while conducting pairwise comparisons paired t-test is a common choice, Wilcoxon signed-rank test is considered safe as it does not require the underlying data to follow normal distribution. Moreover, in case of Wilcoxon signed rank test, its outcome is generally less influenced by exceptionally superior or inferior performance of a technique corresponding to a dataset (i.e., an outlier) [113]. These reasons favor the use of the test for the comparison. The test was conducted at an α value of 0.05. The results of these pairwise comparisons are also depicted in Figure 10 (at the top of data columns). If an ET fared significantly better than the compared non-ensemble technique it was depicted as B*, however, if the ET was better but the results were not significant it was depicted as B at the top of the data column (in Figure 10). Similarly, if the ET turned out to be significantly poorer than the compared non-ensemble technique, it was depicted as W* and if it was worse but not significantly,

it was depicted with the symbol W. For instance, according to Figure 10a, RF is better than ANN, Dec.T, KNN and NB. But these results were not significant. However, RF was significantly better than BN, C4.5, CART, LR, SVM, and VFI according to the Wilcoxon signed-rank test. As depicted in Figure 10b, the AUC values of ROT on various datasets were found to be worse than ANN, CART, and KNN, but not significantly.

According to the results shown in Figure 10, BAG, RF and CODEP exhibited the best AUC values as they were better than all the compared non-ensemble techniques, and more so significantly better than 7, 6 and 3 non-ensemble techniques, respectively. Thereafter, the results of AB, Boosting, and ROT were also good as they were better than the majority of the compared non-ensemble techniques and were only not significantly worse than a maximum of three compared techniques. The results of Stacking were quite poor as it was found worse than the majority of the compared techniques. It was interesting to note while comparing different ET that Stacking, a heterogeneous ET showed the worst results. However, VHetBL and CODEP other heterogeneous techniques showed encouraging results. Though, VHetBL, uses the weighing mechanism for aggregation, both CODEP and stacking use meta-learning as combination mechanisms.

We also compared the performance of ET as a learning algorithm for developing SCP models. However, we could only extract the AUC results of three non-ensemble techniques (ANN, LR, and NB) to be compared with four ET namely AB, BAG, LB, and RF. Similar to the comparison performed for the application as a learning algorithm for developing SDP models, we compared the AUC values dataset wise and performed Wilcoxon signed-rank test with Bonferroni correction. The results of the comparison and the statistical test are indicated in Figure 11. According to the figure apart from the case when AB was compared with ANN, all other ET were found better than the compared non-ensemble techniques. Wilcoxon test results indicated BAG to be the best as it was significantly superior than all the compared techniques.

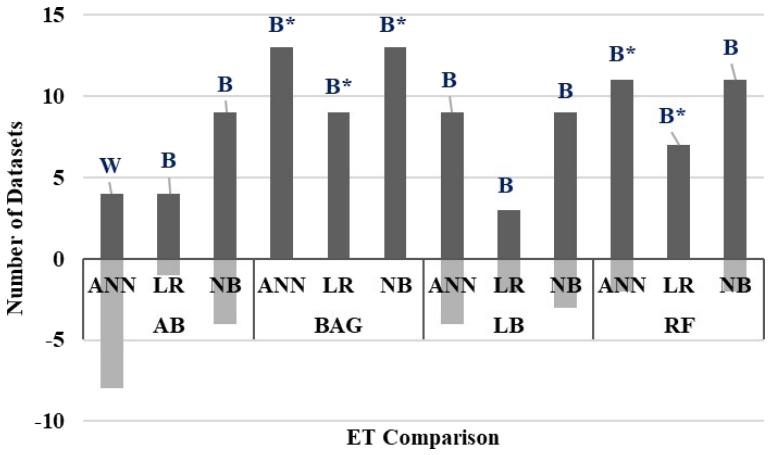

Figure 11. Comparison results of ET with non-ensemble techniques when they are used as a learning algorithm for developing SCP models

The results discussed in the section indicate that the AUC values of the majority of the analyzed ET were found better than the non-ensemble techniques. The primary reason for the good performance of ET is that they combine multiple learners and give stable results as compared to single learners. Also, the various base models of ET are of diverse nature, i.e., several base classifiers are combined that explore a “set of hypotheses” as an

alternative to a single model that searches the "best hypothesis" [114]. This mechanism thereby improves the performance of ET as compared to non-ensemble techniques that are single classifiers. The results discussed in the section favor the use of ET for the explored applications and in other related domains.

3.5.3. Comparative performance amongst ET

As indicated in the previous two sections, ET have been found effective in the SDP/SCP domain for doing various tasks. Furthermore, we intend to evaluate if a specific ET outperforms others in the various applications where they are used. We pairwise compare the AUC values (dataset wise) of all the ET amongst each other for each application and analyze whether a specific ET is significantly better than the majority of other ET for a particular application. The rules for comparing different ET and extracting the AUC values are similar to the ones mentioned in the previous sections. For comparison amongst ET we use Wilcoxon signed-rank test with Bonferroni correction at an α value of 0.05.

Tables 6–8 state the Wilcoxon test results when we compare ET amongst each other, which are used for applications, i.e., as a learning algorithm for developing SDP models, for addressing the class imbalance issue, and as a learning algorithm for developing SCP models respectively. The tables use the following symbols:

- B: When the results of ET depicted in the row is found better than the results of ET depicted in the column. However, not significantly.
- B*: When the results of ET depicted in the row are significantly better than the results of the ET depicted in the column.
- W: When the results of ET depicted in the row are found worse than the results of ET depicted in the column. However, not significantly.
- W*: When the results of ET depicted in the row are significantly worse than the results of the ET depicted in the column.
- EQ: When both the compared ETs get equivalent results, i.e., neither worse nor better.
- ND: When the data to compare the ETs could not be extracted from the primary studies.

It may be noted that we could not compare ET for the application of transfer learning as we could extract AUC statistics for only one technique (VCB-SVM) for this application.

Table 6 compares the ET that were used as a learning algorithm for developing SDP models. The Wilcoxon test results in Table 6 depict that the Stacking technique was found

Table 6. Comparison amongst ET for use as a learning algorithm for developing SDP models (Wilcoxon test results)

	AB	BAG	Boosting	CODEP	LB	LMT	RF	ROT	Stacking	VHetBL
AB	–	W	B	W	B	B	W	B	B*	B
BAG	B	–	W	W*	B	B	W	B	B*	B
Boosting	W	B	–	W	B	W	W	W	B	W
CODEP	B	B*	B*	–	B	ND	B	B	B	B
LB	W	W	W	W	–	W	W	W	B*	W
LMT	W	W	B	ND	B	–	W	W	B	W
RF	B	B	B	W	B	B	–	W	B*	B
ROT	W	W	B	W	B	B	B	–	B*	B
Stacking	W*	W*	W	W	W*	W	W*	W*	–	W*
VHetBL	W	W	B	W	B	B	W	W	B*	–

worse than all the compared ET. On the other hand, the results of CODEP, another heterogenous ensemble that uses the meta-learning combination mechanism was found better than all the other compared ET. Apart from Stacking and CODEP, all the other compared ET use the weighing mechanism for combining the base models. Amongst the ET that used weighing as a mechanism for combination, LB exhibited the worst results. It was only found better than Stacking. The best results were shown by the RF technique as it fared better than seven of the other compared ET. However, it may be noted that the results of RF was significantly better in only one case (when compared with stacking). After CODEP and RF, ROT, AB and BAG showed good results as they were found better than 5 or more compared ET.

Table 7 depicts the comparison results (Wilcoxon test) amongst ET which were used for specifically handling the class imbalance issue. According to the Table, the MBoost technique was the best as its AUC results were better than seven of the compared ET, moreover significantly better than three ET (MKEL, RUSBoost, SMBoost). The next best results were shown by the CEL technique, which was found better than all the other compared ET except MBoost. The worst ET according to the table was MKEL, as its AUC values were poorer than all the compared ET. The poor performance of MKEL could be due to the random selection strategy used to initialize the training set. It is a possibility that no defective samples were selected in the initial training set, thus leading to poor results [50]. RUSBoost was found better than only MKEL and TSE while SMBoost was found better than three ET (MKEL, TSE, and RUSBoost). It is interesting to note that MBoost, a combination of wagging and boosting gives exceptionally good results however, just boosting when combined with a sampling technique (such as SMOTE (SM) or Random Undersampling (RUS)), though handles the class imbalance issue, but fares poorer than most of the other explored ET in the domain.

Table 7. Comparison amongst ET for handling class imbalance issue (Wilcoxon test results)

	BNC	CEL	DNC	MKEL	MBoost	RUSBoost	SMBoost	TSE
BNC	–	W	B	B	W	EQ	B	B
CEL	B	–	B	B*	W	B*	B*	B
DNC	W	W	–	B	W	B	B	B
MKEL	W	W*	W	–	W*	W	W	W
MBoost	B	B	B	B*	–	B*	B*	B
RUSBoost	EQ	W*	W	B	W*	–	W	B
SMBoost	W	W*	W	B	W*	B	–	B
TSE	W	W	W	B	W	W	W	–

Table 8 states the Wilcoxon test results of the comparative performance of ET when used as a learning algorithm for developing SCP models. According to the table, the best ET was BAG as it was better than all the other compared ET, moreover, the results were significant in two out of three cases. The AB technique exhibited poor AUC values and was found significantly worse in all the comparisons. Similarly, the LB technique also showed poor results than most of the other compared ET. The RF technique also exhibited good results for this application.

According to the results, we find that RF and BAG turned out to be a superior technique as a learning algorithm for developing both SDP and SCP models. BAG creates multiple bootstrap training samples for developing diverse base models. RF combines the bootstrap samples used in bagging along with random features to create diverse base

Table 8. Comparison amongst ET for use as a learning algorithm for developing SCP models (Wilcoxon test results)

	AB	BAG	LB	RF
AB	–	W*	W	W*
BAG	B*	–	B*	B
LB	B	W*	–	W*
RF	B*	W	B*	–

models. Though, CODEP technique also showed good results, it was evaluated in only three primary studies. However, both BAG and RF have been widely used in literature studies for various applications in the SDP/SCP domain. A key reason for their popularity is their effective performance and ease of availability, i.e., open-source tools such as WEKA [115], etc. have efficient implementations of BAG and RF. On the other hand, ET like CODEP, ROT, MBoost, and CEL though exhibited good results in different applications are rarely used in SDP/SCP literature. This could be possibly because of the lack of tools that provide their implementation. These techniques should be widely explored in future studies. It may also be noted that though the comparison results indicate that Stacking, TSE and MKEL performed worse than the majority of the other compared ET, but there were very few studies that could provide data for comparing these ET. Thus, these results are not necessarily true in all scenarios. Researchers must perform more experiments that investigate different ET and compare different ET for a specific application.

3.6. Threats specific to the use of ET

While using ET for SDP/SCP, it is essential to understand the possible threats one needs to address for the effective application of these techniques. This would allow the computation of effective and realistic results. We extracted threats specific to the use of ET from the "Threats to Validity" or the "Limitations" section of the primary studies. However, we found that 42% of the primary studies did not report their threats (i.e., did not have any "Threats to Validity" or "Limitations" section). Another section of primary studies (25%) though stated their corresponding threats but did not specify any threats on the use of ET. Only 33% of studies stated threats specific to the use of ET.

The threats extracted from the primary studies were further categorized into 'Construct validity', 'Internal Validity', and 'External Validity' threats [116]. We state only those threats which could be extracted from two or more primary studies. This was done to eliminate threats that are specific to the experimental designs of a corresponding study. The various threats extracted from primary studies are listed in Table 9.

The extracted 'Construct Validity' threats in Table 9 state that the various internal parameter settings, base learners, and combination mechanisms are not experimented by the primary studies. However, it may be noted that though parameter tuning mechanisms [117, 118] may produce effective internal parameter settings, it is very difficult for a researcher to account for a change in base learners and combination mechanisms. In fact, researchers may perform experiments just to evaluate various base learners and combination mechanisms of specific ET (such as ES18).

A critical threat to internal validity is the proper re-implementation of ET (proposed/explored by other studies) for comparing its results with the ET proposed in the corresponding primary study. This needs to be done very carefully, and the results of the

Table 9. Threats to validity specific to the use of ET

Threat	Supporting studies
Construct validity	
Does not experiment with various internal parameter settings of the ET or ensemble size	ES8, ES9, ES25, ES31, ES32, ES40, ES44, ES54, ES56, ES59, ES65, ES67
Does not experiment with different base learners for a specific ET	ES11, ES12, ES20, ES52, ES54, ES59, ES75
Does not account for the variation of results with the change in combination mechanism	ES12, ES20
Internal validity	
Threat concerning proper re-implementation of ET proposed by other studies for comparison with other ET	ES22, ES37, ES59
Bias concerning the selection of ET used in the study	ES25, ES37, ES70
Use of random sample selection strategy for training the ET	ES33, ES34, ES57
External validity	
The number and nature of datasets used for validating the ET may not be appropriate to produce generalized results	ES25, ES77, ES65
Bias concerning the selection of baseline models for comparing ET to obtain generalized results	ES52, ES55, ES77

re-implemented ET should be matched with base studies to ensure they have been properly replicated. However, we would like to add, as previously mentioned in Section 3.2, 18% of primary studies did not mention the base learners used by the ET, moreover, 17% did not mention (or partially mention) the ensemble size used by them. Such practice makes replication of ET impossible. Researchers must mention all the parameter settings, base learners, and ensemble size of the ET used by them. Other internal validity threats involve bias in the selection of ET used by a study, and use of random selection strategy used for training certain ET (such as MKEL).

One threat to "External Validity" (Table 9) states the bias in the selection of datasets for performing the experiment. However, this threat can only be mitigated by using datasets of varied domains, sizes, and programming languages. The other external validity threat concerns itself with the selection of baseline models for comparing the ET. A researcher should choose a representative set of baseline models that are widely used by researchers for a specific application or represent various categories of algorithms (such as while analyzing ET for the class imbalance issue, a researcher may select baseline models that are representative each from Cost-sensitive ET, Boosting based ET, Bagging based ET, Hybrid ET and Novel ET as discussed in Section 3.3). Moreover, a researcher should clearly state the reason behind his choice of baseline models for comparison.

4. Discussion of results and future work

This section discusses the results presented in Section 3 and analyzes the gaps in the literature. Out of the 77 primary studies, only 10 studies used ET for SCP, all other studies were focused on SDP. Both SDP and SCP are important key activities that aid in software quality improvement. Thus, researchers should compulsorily conduct more studies that

analyze and compare the capabilities of ET for various tasks in SCP apart from SDP. Furthermore, on the basis of the analysis conducted in this paper, we propose future work to researchers, which is discussed in the following sections.

4.1. Discussions related to RQ1

RQ1 attempts to categorize various ET used in literature according to the base learners used and other criteria involving their fucntioning.

- As discussed in RQ1, ET were categorized on various parameters such as the similarity of learners used by base models, their aggregation, relationship, diversity, and dependency. Though the primary studies of the review investigated ET that corresponded to most of these categories, few categories were ignored by a majority of the studies. For instance, no study used an ET which used the top-down approach for aggregation of base models. Also, there were very few studies which investigated ET that had competitive relationship amongst base models. Future studies should evaluate these less commonly explored categorizations of ET.
- While analyzing the families of machine learning techniques used as base learners, we found that only 11% of primary studies used search-based algorithms as base learners. Researchers have ascertained the effectiveness of search-based algorithms in the domain of software quality predictive modeling [26]. Therefore, future studies should extensively explore ET that use search-based algorithms as base learners.
- Another interesting class of ET (explored by 35% of primary studies) were ensembles of ensembles that use ET such as RF, BAG, AB, and Gradient Boosting as base learners. Apart from these ET other techniques such as ROT, MC, Random Subspace or other ET should be investigated as base learners for forming new ensembles. Certain primary studies also proposed new ensemble of ensembles such as Deep Forest (ES65), Ensemble Random Undersampling (ES44) and others. However, it was also observed that only one primary SCP study evaluated ET as base learners. More studies which assess the use of ensemble of ensembles should be conducted in the domain of SCP.
- The essence of ET is aggregation of several base models to yield a more stabilized and reliable predictive outcome. However, all the base models of an ET use the original feature set of the training dataset. These original features primarily quantify the measurements in the process (such as evolution-based metrics [9]) or the code structure (represented by code metrics [7]). On the contrary, deep learning techniques in SDP generates new higher-level features from the original given feature set which symbolize the semantic attributes and have found to yield better predictive outcomes than models developed using original feature set [119, 120]. However, they do not generate multiple models to provide aggregated and stabilized results. A culmination of both these techniques, i.e., deep learning and ensemble learning is promising. Such a combination has been explored by two primary studies. ES65 uses Deep Forest for ensembles while ES74 uses deep representation of software metrics followed by two stage ensemble learning. The results of both the studies have ascertained that blend of deep learning and ensemble learning is successful and would yield upscaling of current SDP models. Researchers in future should further explore the combination of these two paradigms for conclusive and generalized results.

4.2. Discussions related to RQ2

RQ2 investigates the various applications for which ET have been utilized in SDP/SCP literature.

- We found only six ET namely SOB, OOB, UOB and three heterogeneous learners based on plurality voting, soft voting and stacking, which were explored for online learning in SDP. Moreover, there were only two primary studies that evaluated ET for online learning. There is an urgent need for more studies that assess and evaluate ET for online learning not just for SDP but for SCP too.
- After analyzing the results of RQ2, we found that 65% of the studies used ET as a learning algorithm for developing SDP models. However, there are various other less commonly explored applications of ET such as transfer learning (explored by 10% of primary studies) which need more investigation by the research community. It may also be noted that we could not find sufficient data to assess and compare ET for these less explored applications. These observations necessitate a mandatory step by the research community in exploring ET for transfer learning and other less explored applications.
- Researchers in the future should propose ET that collectively deal with diverse issues such as handling imbalanced data and online learning together or other unified ET such as TCSBoost that deal with multiple issues simultaneously. Studies should also be conducted to extensively validate such proposed techniques and obtain generalized conclusions concerning their effectiveness.

4.3. Discussions related to RQ3

RQ3 analyzes the various mechanisms/rules which have been used to aggregate base models in ET used in SDP/SCP literature.

- Researchers may conduct studies where they experiment with different base learners for a specific ensemble technique. The results of such studies can be used to effectively choose base learners as there are a wide variety of options available in literature as discussed in RQ1. Studies should also be conducted to evaluate different combination rules, i.e., if they improve or deteriorate the performance of a specific ensemble technique.
- As we evaluated the various combination mechanisms used in ET, we found that there was a need to extensively validate the ET which are based on the meta-learning mechanism. The performance of such techniques (evaluated in RQ4) could not be generalized as though CODEP yielded exceptionally good results, the results of Stacking was found to be poor when compared with non-ensemble techniques and amongst each other. However, as the comparison data for these techniques could be extracted from very few studies, these techniques should be explored by large number of studies in both SDP as well as SCP.

4.4. Discussions related to RQ4

RQ4 evaluates the performance of ET for the various applications in SDP/SCP domain. It also attempts to compare the performance of ET amongst each other and with other non-ensemble techniques for the various applications.

- While conducting comparisons for RQ4, we were not able to effectively compare all the applications of ET. Thus, more studies should be conducted which provide comparisons of ET amongst each other and with different non-ensemble techniques for varied

applications. Moreover, apart from AUC, other stable performance measures such as MCC [90] or Balance [63] should be widely used by researchers to report and compare the results of SDP/SCP models developed using various ET. Different ET that address the class imbalance issue may also be compared based on "cost-effectiveness" to provide a comprehensive picture to other researchers and software practitioners.
– Certain ET such as ROT, MBoost, and CEL, though exhibited promising results were not widely used by primary studies. Such ET should be thoroughly investigated for various applications. Also, various ET such as Multischeme, Non Negative Sparse-based Semiboost, RBBag, ASCI, DECORATE, ASOF, etc., were only investigated in one primary study each. More studies must be conducted which evaluate and compare such ET in the SDP/SCP domain.
– It was also observed that ET belonging to the same category may exhibit contrasting results. For instance, as Stacking, CODEP and VHetBL are heterogeneous ET, but exhibit very different results when they were compared as a learning algorithm for developing SDP models. Researchers in future should conduct comparisons amongst specific categories of ET such as comparison amongst heterogeneous learners for several applications to observe their capabilities and effectiveness.

4.5. Discussions related to RQ5

RQ5 scrutinizes the primary studies for the various threats specific to the use of ET in SDP/SCP literature.
– The threats specific to ET could be extracted from just a few studies. As a good practice, researchers should state all the possible threats to validity in their studies. Moreover, they should design their experiments so that possible threats can be minimized as far as possible.
– The review results indicated several primary studies that did not either state the base learners used (18% of primary studies) or did not mention the ensemble size (17% of primary studies). Such incomplete information hinders the replication of results by other researchers. Also, several researchers proposed new ET, however, they should be encouraged to provide tools for their proposed techniques. This would enable other researchers to validate and replicate their proposed techniques. If not tools, researchers should at least clearly state all the internal parameter settings, base learners used, and combination mechanisms so that others may replicate and repeat their results for comparison.

5. Threats to validity

This section discusses the various threats to the validity of this review. The search for relevant studies of the review included the formulation of a search string by choosing specific search terms from the research questions. The search string was thereafter used to retrieve studies from five electronic databases. However, it may be the case that certain relevant studies may not include the search terms in their titles, abstracts, or keywords. We might miss such studies. In order to address this threat, we manually scanned the reference lists of all the extracted studies so that we may not miss a relevant study. Furthermore, we also scanned the reference lists of two recent reviews [10, 13] conducted on SDP and SCP. We are positive that these steps reduce the risk of missing out on a relevant study.

Another possible threat to the review results occurs from our assumption that all the primary studies present their results in an unbiased manner. However, there could be publication bias, wherein there are higher chances that positive results of ET are reported rather than negative results [32]. There is a possibility that the authors of a study may incorrectly claim that their proposed ET is better than other ET prevalent in literature. In order to encounter this threat, we included "empirical studies which compare different ET with each other or with other non-ensemble techniques for SDP or SCP" as an inclusion criterion (mentioned in Section 2.2.1). Such studies only aim to compare various existing ET, and do not propose their own techniques. Therefore, such studies would report both favorable and unfavorable results of ET, mitigating the publication bias.

To evaluate the capability of ET for various applications related to SDP and SCP, we performed a comparison of the results of SDP/SCP models developed by various ET and non-ensemble techniques. For doing so, data was extracted from different primary studies. However, these studies use diverse experimental designs and settings (internal parameters of ET, size of ET, base learners of ET, independent variables, datasets, preprocessing techniques, etc.). This could be a possible threat to the review. This threat was mitigated by reporting the statistics dataset wise, after removing the outliers. This would ensure that exceptional values reported by a specific study due to its corresponding experimental design are removed. Moreover, we also state the median values to report the most common values rather than extreme results reported by a study. Another possible threat in comparing ET and non-ensemble techniques is that there could be certain bias in the dataset wise comparison performed in the review. As already pointed out, we collect only those studies that use ET or compare ET with each other and other non-ensemble techniques. Since, we do not collect and extract data from studies that have used only non-ensemble techniques on the compared datasets, the comparison may be biased and more favorable for ET. The only way to address this threat is to additionally collect and extract data from studies that have employed non-ensemble techniques on the said datasets. However, this is beyond the purview of the study.

The external validity of the review concerns itself with the appropriateness of the primary studies of the review, as per the review's objective, so that the review results are valid and generalizable. The review protocol is clearly defined so that we extract a valid set of primary studies, which are in line with the review objectives. Also, the study clearly states the review protocol, which supports the replicability and repeatability of the review.

6. Conclusion

This review systematically summarizes the use of ET in SDP and SCP studies. We analyzed the studies that used ET in SDP/SCP literature from five perspectives namely their category, application, combination mechanism, performance, and probable threats that could occur while using ET. We extensively explored 5 online libraries and extracted 77 primary studies in the period from January 2000 to December 2020. The primary findings of the review are summarized below:

– ET used in SDP/SCP literature can be categorized according to five criteria which includes: a) similarity of the base models (homogeneous and heterogeneous), b) aggregation mechanism of base models (top-down and bottom-up), c) relationship amongst base models (competitive or cooperative), d) diversity of base models (implicit and explicit), and e) dependency amongst base models (dependent and independent). Amongst the

mentioned criteria, we found that homogeneous, bottom-up, cooperative, explicit, and independent ET are popular with respect to learner similarity, aggregation, relationship, diversity, and dependency categorizations. Tree-based learners were the most popular machine learning family which were used as base learners for ET.

– After analyzing the primary studies, we found six different applications of ET in SDP/SCP literature. The most common application of ET was its use as a learning algorithm for developing an SDP model. The other applications were addressing the issue of imbalanced training data, their use as a learning algorithm for developing an SCP model, transfer learning, online learning, and feature selection.
– Primarily, there are two mechanisms for combining base models, one is where the output of constituent base models are given specific weights to get an aggregated ensemble output, the second is when an ensemble is constructed through meta-learning. Only twelve ET used the meta-learning mechanism while all others used the weighing mechanism. We found sixteen combination rules for ET that used the weighing mechanism for aggregation. Amongst them, some of the popular ones were majority voting, providing weights according to misclassification error, and combining the base models according to average probability.
– The performance of ET was analyzed dataset wise by evaluating the AUC and recall performance metrics. A mean AUC value of 0.75 or above was depicted by a majority of the explored ET when used as a learning algorithm for developing SDP or SCP models or for addressing the imbalanced data issue. Majority of ET that were used as a learning algorithm for developing SDP models depicted median recall values in the range 70%–90%. A comparison of ET with other non-ensemble techniques (conducted using vote count method and Wilcoxon signed ranked test) indicated that RF and BAG were superior and popular ET as they exhibited better results than most of the other compared non-ensemble techniques when being used as learners for developing SDP or SCP models. The CODEP technique, a heterogeneous ET also exhibited favourable results. We also compared ET amongst each other and found CODEP, RF and BAG to be the best performing ET when used for developing SDP/SCP models and MBoost as the best technique for handling skewed data.
– Amongst 77 primary studies, only 33% of them reported the threats specific to the use of ET. The construct validity threats included the inability of the study to account for the change in parameter settings, base learners, and the combination mechanism of the ET. The internal validity threats need to address the biased selection of ET in a study, suitable replication of ET proposed by other studies, and accounting for the random selection strategy for training certain ET. The reported external validity threats could be addressed by the selection of an appropriate number and nature of datasets for empirical validation and selection of appropriate baseline models for comparing the ET.

Acknowledgement

The author would like to acknowledge the contribution of Ms. Sugandha Gupta for helping in data extraction and quality analysis of the candidate studies.

References

[1] N.E. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Transactions on Software Engineering*, Vol. 26, No. 8, Aug. 2000,

pp. 797–814.
[2] 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, Aug. 2005, pp. 625–642.
[3] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Transactions on Software Engineering*, Vol. 34, No. 4, May 2008, pp. 485–496.
[4] N. Seliya, T.M. Khoshgoftaar, and J. Van Hulse, "Predicting faults in high assurance software," in *12th International Symposium on High Assurance Systems Engineering*. IEEE, Nov. 2010, pp. 26–34.
[5] 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, Sep. 2017, pp. 673–717.
[6] 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, Jan. 2007, pp. 63–73.
[7] D. Romano and M. Pinzger, "Using source code metrics to predict change-prone java interfaces," in *27th International Conference on Software Maintenance (ICSM)*. IEEE, Sep. 2011, pp. 303–312.
[8] 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, Jun. 2012, pp. 217–226.
[9] 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, May 2013, pp. 407–437.
[10] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, Vol. 27, Feb. 2015, pp. 504–518.
[11] 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, Apr. 2015, pp. 1–16.
[12] A. Idri, M. Hosni, and A. Abran, "Systematic literature review of ensemble effort estimation," *Journal of Systems and Software*, Vol. 118, Aug. 2016, pp. 151–175.
[13] R. Malhotra and M. Khanna, "Software change prediction: A systematic review and future guidelines," *e-Informatica Software Engineering Journal*, Vol. 13, No. 1, 2019, pp. 227–259.
[14] L.I. Kuncheva and C.J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine Learning*, Vol. 51, No. 2, May 2003, pp. 181–207.
[15] L. Jonsson, M. Borg, D. Broman, K. Sandahl, S. Eldh et al., "Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts," *Empirical Software Engineering*, Vol. 21, No. 4, Aug. 2016, pp. 1533–1578.
[16] S.S. Rathore and S. Kumar, "Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems," *Knowledge-Based Systems*, Vol. 119, Mar. 2017, pp. 232–256.
[17] R. Malhotra and M. Khanna, "Particle swarm optimization-based ensemble learning for software change prediction," *Information and Software Technology*, Vol. 102, Oct. 2018, pp. 65–84.
[18] M. Re and G. Valentini, "Ensemble methods: A review," in *Advances in Machine Learning and Data Mining for Astronomy, Data Mining and Knowledge Discovery*. Chapman-Hall, 2012, pp. 563–594.
[19] V. Bolón-Canedo and A. Alonso-Betanzos, "Ensembles for feature selection: A review and future trends," *Information Fusion*, Vol. 52, Dec. 2019, pp. 1–12.
[20] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 42, No. 4, Aug. 2011, pp. 463–484.

[21] 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, Aug. 2013, pp. 1397–1418.
[22] C. Catal, "Software fault prediction: A literature review and current trends," *Expert Systems With Applications*, Vol. 38, No. 4, Apr. 2011, pp. 4626–4636.
[23] 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, Nov. 2017, pp. 111–147.
[24] 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, Feb. 2017, pp. 85–109.
[25] R. Malhotra and M. Khanna, "Threats to validity in search-based predictive modelling for software engineering," *IET Software*, Vol. 12, No. 4, Jun. 2018, pp. 293–305.
[26] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Systems With Applications*, Vol. 36, No. 4, May 2009, pp. 7346–7354.
[27] 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, Oct. 2011, pp. 1276–1304.
[28] R. Malhotra and A.J. Bansal, "Software change prediction: A literature review," *International Journal of Computer Applications in Technology*, Vol. 54, No. 4, Nov. 2016, pp. 240–256.
[29] B.A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-based software engineering and systematic reviews*. CRC press, Nov. 2015, Vol. 4.
[30] G. Catolino and F. Ferrucci, "An extensive evaluation of ensemble techniques for software change prediction," *Journal of Software: Evolution and Process*, Mar. 2019, p. e2156.
[31] 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, Oct. 2018, p. e2111.
[32] 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, Jan. 2012, pp. 41–59.
[33] Y. Jiang, B. Cukic, and T. Menzies, "Fault prediction using early lifecycle data," in *The 18th International Symposium on Software Reliability (ISSRE'07)*. IEEE, Nov. 2007, pp. 237–246.
[34] E. Rubinić, G. Mauša, and T.G. Grbac, "Software defect classification with a variant of NSGA-II and simple voting strategies," in *International Symposium on Search Based Software Engineering*. Springer, Sep. 2015, pp. 347–353.
[35] A. Ali, M. Abu-Tair, J. Noppen, S. McClean, Z. Lin et al., "Contributing features-based schemes for software defect prediction," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, Dec. 2019, pp. 350–361.
[36] Y. Ma, L. Guo, and B. Cukic, "A statistical framework for the prediction of fault-proneness," in *Advances in Machine Learning Applications in Software Engineering*. IGI Global, 2007, pp. 237–263.
[37] M.J. Siers and M.Z. Islam, "Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem," *Information Systems*, Vol. 51, Jul. 2015, pp. 62–71.
[38] J.R. Campos, E. Costa, and M. Vieira, "Improving failure prediction by ensembling the decisions of machine learning models: A case study," *IEEE Access*, Vol. 7, Dec. 2019, pp. 177 661–177 674.
[39] G. Li and S. Wang, "Oversampling boosting for classification of imbalanced software defect data," in *35th Chinese Control Conference (CCC)*. IEEE, Jul. 2016, pp. 4149–4154.
[40] H. Jia, F. Shu, Y. Yang, and Q. Wang, "Predicting fault-prone modules: A comparative study," in *International Conference on Software Engineering Approaches for Offshore and Outsourced Development*. Springer, Jul. 2009, pp. 45–59.
[41] R. Malhotra, "An empirical framework for defect prediction using machine learning techniques with Android software," *Applied Soft Computing*, Vol. 49, Dec. 2016, pp. 1034–1050.

[42] L. Gong, S. Jiang, and L. Jiang, "An improved transfer adaptive boosting approach for mixed-project defect prediction," *Journal of Software: Evolution and Process*, Vol. 31, No. 10, Oct. 2019, p. e2172.

[43] T.M. Khoshgoftaar, P. Rebours, and N. Seliya, "Software quality analysis by combining multiple projects and learners," *Software Quality Journal*, Vol. 17, No. 1, Mar. 2009, pp. 25–49.

[44] D. Ryu, O. Choi, and J. Baik, "Value-cognitive boosting with a support vector machine for cross-project defect prediction," *Empirical Software Engineering*, Vol. 21, No. 1, Feb. 2016, pp. 43–71.

[45] H. He, X. Zhang, Q. Wang, J. Ren, J. Liu et al., "Ensemble multiboost based on ripper classifier for prediction of imbalanced software defect data," *IEEE Access*, Vol. 7, Aug. 2019, pp. 110 333–110 343.

[46] T. Mende and R. Koschke, "Revisiting the evaluation of defect prediction models," in *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, May 2009, pp. 1–10.

[47] J. Petrić, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, "Building an ensemble for software defect prediction based on diversity selection," in *Proceedings of the 10th ACM/IEEE International symposium on empirical software engineering and measurement*, Sep. 2016, pp. 1–10.

[48] 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 (formerly known as India Software Engineering Conference)*. ACM, Feb. 2019, p. 8.

[49] C. Seiffert, T.M. Khoshgoftaar, and J. Van Hulse, "Improving software-quality predictions with data sampling and boosting," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 39, No. 6, Sep. 2009, pp. 1283–1294.

[50] T. Wang, Z. Zhang, X. Jing, and L. Zhang, "Multiple kernel ensemble learning for software defect prediction," *Automated Software Engineering*, Vol. 23, No. 4, Dec. 2016, pp. 569–590.

[51] Z. Li, X.Y. Jing, X. Zhu, H. Zhang, B. Xu et al., "Heterogeneous defect prediction with two-stage ensemble learning," *Automated Software Engineering*, Vol. 26, No. 3, 2019, pp. 599–651.

[52] E. Arisholm, L.C. Briand, and E.B. Johannessen, "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models," *Journal of Systems and Software*, Vol. 83, No. 1, Jan. 2010, pp. 2–17.

[53] T. Wang, Z. Zhang, X. Jing, and Y. Liu, "Non-negative sparse-based semiboost for software defect prediction," *Software Testing, Verification and Reliability*, Vol. 26, No. 7, Nov. 2016, pp. 498–515.

[54] R. Li, L. Zhou, S. Zhang, H. Liu, X. Huang et al., "Software defect prediction based on ensemble learning," in *Proceedings of the 2019 2nd International conference on data science and information technology*, Jul. 2019, pp. 1–6.

[55] Y. Liu, T.M. Khoshgoftaar, and N. Seliya, "Evolutionary optimization of software quality modeling with multiple repositories," *IEEE Transactions on Software Engineering*, Vol. 36, No. 6, May 2010, pp. 852–864.

[56] X. Xia, D. Lo, S.J. Pan, N. Nagappan, and X. Wang, "Hydra: Massively compositional model for cross-project defect prediction," *IEEE Transactions on software Engineering*, Vol. 42, No. 10, Nov. 2016, pp. 977–998.

[57] R. Malhotra and S. Kamal, "An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data," *Neurocomputing*, Vol. 343, May 2019, pp. 120–140.

[58] J. Zheng, "Cost-sensitive boosting neural networks for software defect prediction," *Expert Systems with Applications*, Vol. 37, No. 6, Jun. 2010, pp. 4537–4543.

[59] H. Alsawalqah, H. Faris, I. Aljarah, L. Alnemer, and N. Alhindawi, "Hybrid SMOTE-ensemble approach for software defect prediction," in *Computer Science on-Line Conference*. Springer, Apr. 2017, pp. 355–366.

[60] R. Malhotra and M. Khanna, "Dynamic selection of fitness function for software change

prediction using particle swarm optimization," *Information and Software Technology*, Vol. 112, Aug. 2019, pp. 51–67.

[61] D. Di Nucci, F. Palomba, R. Oliveto, and A. De Lucia, "Dynamic selection of classifiers in bug prediction: An adaptive method," *IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. 1, No. 3, May 2017, pp. 202–212.

[62] H. Tong, B. Liu, and S. Wang, "Kernel spectral embedding transfer ensemble for heterogeneous defect prediction," *IEEE Transactions on Software Engineering*, Vol. 14, No. 8, Sep. 2019, pp. 1–21.

[63] A.T. Mısırlı, A.B. Bener, and B. Turhan, "An industrial case study of classifier ensembles for locating software defects," *Software Quality Journal*, Vol. 19, No. 3, Sep. 2011, pp. 515–536.

[64] L. Kumar, S. Misra, and S.K. Rath, "An empirical analysis of the effectiveness of software metrics and fault prediction model for identifying faulty classes," *Computer Standards and Interfaces*, Vol. 53, Aug. 2017, pp. 1–32.

[65] H.D. Tran, L.T.M. Hanh, and N.T. Binh, "Combining feature selection, feature learning and ensemble learning for software fault prediction," in *11th International Conference on Knowledge and Systems Engineering (KSE)*. IEEE, Oct. 2019, pp. 1–8.

[66] Y. Peng, G. Kou, G. Wang, W. Wu, and Y. Shi, "Ensemble of software defect predictors: an AHP-based evaluation method," *International Journal of Information Technology and Decision Making*, Vol. 10, No. 01, Jan. 2011, pp. 187–206.

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

[68] T. Zhou, X. Sun, X. Xia, B. Li, and X. Chen, "Improving defect prediction with deep forest," *Information and Software Technology*, Vol. 114, Oct. 2019, pp. 204–216.

[69] N. Seliya and T.M. Khoshgoftaar, "The use of decision trees for cost-sensitive classification: an empirical study in software quality prediction," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 1, No. 5, Sep. 2011, pp. 448–459.

[70] D. Ryu, J.I. Jang, and J. Baik, "A transfer cost-sensitive boosting approach for cross-project defect prediction," *Software Quality Journal*, Vol. 25, No. 1, Mar. 2017, pp. 235–272.

[71] R. Abbas, F.A. Albalooshi, and M. Hammad, "Software change proneness prediction using machine learning," in *International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*. IEEE, Dec. 2020, pp. 1–7.

[72] K. Gao, T.M. Khoshgoftaar, and A. Napolitano, "A hybrid approach to coping with high dimensionality and class imbalance for software defect prediction," in *11th international conference on machine learning and applications*, Vol. 2. IEEE, Dec. 2012, pp. 281–288.

[73] C.W. Yohannese, T. Li, M. Simfukwe, and F. Khurshid, "Ensembles based combined learning for improved software fault prediction: A comparative study," in *12th International conference on intelligent systems and knowledge engineering (ISKE)*. IEEE, Nov. 2017, pp. 1–6.

[74] H. Aljamaan and A. Alazba, "Software defect prediction using tree-based ensembles," in *Proceedings of the 16th ACM international conference on predictive models and data analytics in software engineering*, Nov. 2020, pp. 1–10.

[75] Z. Sun, Q. Song, and X. Zhu, "Using coding-based ensemble learning to improve software defect prediction," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 42, No. 6, Dec. 2012, pp. 1806–1817.

[76] 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, Nov. 2018, pp. 69–84.

[77] A.A. Ansari, A. Iqbal, and B. Sahoo, "Heterogeneous defect prediction using ensemble learning technique," in *Artificial Intelligence and Evolutionary Computations in Engineering Systems*. Springer, 2020, pp. 283–293.

[78] S. Wang, L.L. Minku, and X. Yao, "Online class imbalance learning and its applications in fault detection," *International Journal of Computational Intelligence and Applications*, Vol. 12, No. 4, Dec. 2013, p. 1340001.

[79] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: Do different classifiers find the same defects?" *Software Quality Journal*, Vol. 26, No. 2, Jun. 2018, pp. 525–552.

[80] M. Banga and A. Bansal, "Proposed software faults detection using hybrid approach," *Security and Privacy*, Jan. 2020, p. e103.
[81] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Transactions on Reliability*, Vol. 62, No. 2, Apr. 2013, pp. 434–443.
[82] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Tackling class overlap and imbalance problems in software defect prediction," *Software Quality Journal*, Vol. 26, No. 1, Jun. 2018, pp. 97–125.
[83] E. Elahi, S. Kanwal, and A.N. Asif, "A new ensemble approach for software fault prediction," in *17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. IEEE, Jan. 2020, pp. 407–412.
[84] A. Kaur and K. Kaur, "Performance analysis of ensemble learning for predicting defects in open source software," in *international Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, Sep. 2014, pp. 219–225.
[85] S.A. El-Shorbagy, W.M. El-Gammal, and W.M. Abdelmoez, "Using SMOTE and heterogeneous stacking in ensemble learning for software defect prediction," in *Proceedings of the 7th International Conference on Software and Information Engineering*, May 2018, pp. 44–47.
[86] L. Goel, M. Sharma, S.K. Khatri, and D. Damodaran, "Defect prediction of cross projects using PCA and ensemble learning approach," in *Micro-Electronics and Telecommunication Engineering*. Springer, 2020, pp. 307–315.
[87] A. Panichella, R. Oliveto, and A. De Lucia, "Cross-project defect prediction models: L'union fait la force," in *Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. IEEE, Feb. 2014, pp. 164–173.
[88] R. Malhotra and A. Bansal, "Investigation of various data analysis techniques to identify change prone parts of an open source software," *International Journal of System Assurance Engineering and Management*, Vol. 9, No. 2, Apr. 2018, pp. 401–426.
[89] T.T. Khuat and M.H. Le, "Evaluation of sampling-based ensembles of classifiers on imbalanced data for software defect prediction problems," *SN Computer Science*, Vol. 1, No. 2, Mar. 2020, pp. 1–16.
[90] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J.C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, May 2014, pp. 1–10.
[91] R. Malhotra and J. Jain, "Handling imbalanced data using ensemble learning in software defect prediction," in *10th International Conference on Cloud Computing, Data Science and Engineering (Confluence)*. IEEE, Jan. 2020, pp. 300–304.
[92] V. Suma, T. Pushphavathi, and V. Ramaswamy, "An approach to predict software project success based on random forest classifier," in *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol II*. Springer, 2014, pp. 329–336.
[93] R. Mousavi, M. Eftekhari, and F. Rahdari, "Omni-ensemble learning (OEL): utilizing over-bagging, static and dynamic ensemble selection approaches for software defect prediction," *International Journal on Artificial Intelligence Tools*, Vol. 27, No. 6, Sep. 2018, p. 1850024.
[94] S.K. Pandey, R.B. Mishra, and A.K. Tripathi, "BPDET: An effective software bug prediction model using deep representation and ensemble learning techniques," *Expert Systems with Applications*, Vol. 144, Apr. 2020, p. 113085.
[95] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Negative samples reduction in cross-company software defects prediction," *Information and Software Technology*, Vol. 62, Jun. 2015, pp. 67–77.
[96] S. Moustafa, M.Y. ElNainay, N. El Makky, and M.S. Abougabal, "Software bug prediction using weighted majority voting techniques," *Alexandria Engineering Journal*, Vol. 57, No. 4, Dec. 2018, pp. 2763–2774.
[97] S.S. Rathore and S. Kumar, "An empirical study of ensemble techniques for software fault prediction," *Applied Intelligence*, Vol. 51, No. 6, Jun. 2021, pp. 3615–3644.
[98] M.O. Elish, H. Aljamaan, and I. Ahmad, "Three empirical studies on predicting software maintainability using ensemble methods," *Soft Computing*, Vol. 19, No. 9, Sep. 2015, pp. 2511–2524.
[99] H. Tong, B. Liu, and S. Wang, "Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning," *Information and Software Technology*, Vol. 96, Apr. 2018,

pp. 94–111.
[100] A.A. Saifan and L. Abu-wardih, "Software defect prediction based on feature subset selection and ensemble classification," *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, Vol. 14, No. 2, Oct. 2020, pp. 213–228.
[101] S. Hussain, J. Keung, A.A. Khan, and K.E. Bennin, "Performance evaluation of ensemble methods for software fault prediction: An experiment," in *Proceedings of the ASWEC 24th Australasian software engineering conference*, Sep. 2015, pp. 91–95.
[102] Y. Zhang, D. Lo, X. Xia, and J. Sun, "Combined classifier for cross-project defect prediction: an extended empirical study," *Frontiers of Computer Science*, Vol. 12, No. 2, 2018, pp. 280–296.
[103] F. Yucalar, A. Ozcift, E. Borandag, and D. Kilinc, "Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability," *Engineering Science and Technology, an International Journal*, Vol. 23, No. 4, Aug. 2020, pp. 938–950.
[104] I.H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology*, Vol. 58, Feb. 2015, pp. 388–402.
[105] L. Rokach, "Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography," *Computational statistics & data analysis*, Vol. 53, No. 12, Oct. 2009, pp. 4046–4072.
[106] C. Sammut and G.I. Webb, Eds., *Encyclopedia of Machine Learning*. Springer Science & Business Media, 2011.
[107] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 8, No. 4, 2018, p. e1249.
[108] A.J. Sharkey, "Types of multinet system," in *International Workshop on Multiple Classifier Systems*. Springer, Jun. 2002, pp. 108–117.
[109] H. He and E.A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, No. 9, Jun. 2009, pp. 1263–1284.
[110] M. Tan, L. Tan, S. Dara, and C. Mayeux, "Online defect prediction for imbalanced data," in *37th International Conference on Software Engineering*, Vol. 2. IEEE, May 2015, pp. 99–108.
[111] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, Vol. 27, No. 8, Jun. 2006, pp. 861–874.
[112] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, "Problems with precision: A response to "Comments on 'Data mining static code attributes to learn defect predictors"'," *IEEE Transactions on Software Engineering*, Vol. 33, No. 9, Aug. 2007, pp. 637–640.
[113] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, Vol. 1, No. 1, Mar. 2011, pp. 3–18.
[114] T.G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*. Springer, Jun. 2000, pp. 1–15.
[115] F. Eibe, M.A. Hall, and I.H. Witten, "The WEKA workbench. online appendix for data mining: practical machine learning tools and techniques," in *Morgan Kaufmann*. Elsevier Amsterdam, The Netherlands, 2016.
[116] T.D. Cook, D.T. Campbell, and A. Day, *Quasi-experimentation: Design and analysis issues for field settings*. Houghton Mifflin Boston, 1979, Vol. 351.
[117] W. Fu, V. Nair, and T. Menzies, "Why is differential evolution better than grid search for tuning defect predictors?" *arXiv preprint arXiv:1609.02613*, 2016.
[118] C. Tantithamthavorn, S. McIntosh, A.E. Hassan, and K. Matsumoto, "The impact of automated parameter optimization on defect prediction models," *IEEE Transactions on Software Engineering*, Vol. 45, No. 7, Jan. 2018, pp. 683–711.
[119] S. Omri and C. Sinz, "Deep learning for software defect prediction: A survey," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, Jun. 2020, pp. 209–214.
[120] E.N. Akimova, A.Y. Bersenev, A.A. Deikov, K.S. Kobylkin, A.V. Konygin et al., "A survey on software defect prediction using deep learning," *Mathematics*, Vol. 9, No. 11, Jan. 2021, p. 1180.

Glossary

AB: AdaBoost
ANN: Artificial Neural Network
AUC: Area Under the Receiver operating characteristic Curve
ASOF: Adaptive Selection of Optimum Fitness
ASCI: Adaptive Selection of Classifiers
BAG: Bagging
BN: Bayesian Network
BNC: AdaBoost.NC
BTE: Best in Training Ensemble
CART: Classification and Regression Tree
CEL: Coding based Multiclassifier
CS: Cumulative Quality Score
CODEP: Combined Defect Predictor
Dag: Dagging
Dec.T: Decision Table
Decorate: Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples
DNC: Dynamic Adaboost.NC
DTB: Double Transfer Boosting
ET: Ensemble Techniques
ITrAdaBoost: Improved Transfer Adaptive Boosting
KNN: K-Nearest Neighbor
KSETE: Kernel Spectral Embedding Transfer Ensemble
LB: LogitBoost
LMT: Logit Model Tree
LR: Logistic Regression
MBoost: MultiBoost
MC: MetaCost
MCC: Mathews Correlation Coefficient
MKEL: Multiple Kernel Ensemble Learning
NB: Naïve Bayes
NDTF: Non-Linear Decision Tree Forest
OEL: Omni Ensemble Learning
OOB: Oversampling based Online Bagging
RBBag: Roughly Balanced Bagging
RF: Random Forests
ROT: Rotation Forest
RQ: Research Question
RUSBoost: Random UnderSampling Boosting
RS: Random Subspace
SCP: Software Change Prediction
SDP: Software Defect Prediction
SI: Study Identifier
SMBoost: SMOTEBoost
SOB: Sampling based Online Bagging
SQA: Software Quality Assurance
SVM: Support Vector Machine
TCSBoost: TransferCostSensitive Boosting
TSE: Two Stage Ensemble
UOB: Undersampling based Online Bagging
VCB-SVM : Value Cognitive Boosting with Support Vector Machine
VFI: Voting Feature Intervals
VHetBL: Voting amongst Heterogenous Base Learners
VHomBL: Voting amongst Homogeneous Base Learners
VV: Validation and Voting
WEKA: Waikato Environment for Knowledge Analysis

Appendix

Table A1 lists all the primary studies that use a specific ET. Table A2 states the machine learning family of the techniques, which have been used as base learners for ET in the primary studies.

Table A1. List of ET used by primary studies

ET	Primary Studies
AdaBoost	ES7, ES8, ES13, ES21, ES24, ES25, ES29, ES30, ES36, ES39, ES41, ES44, ES46, ES47, ES50, ES55, ES56, ES60, ES61, ES62, ES67, ES73, ES74, ES76, ES77
AdaBoost.NC	ES18, ES73
AdaCost	ES14, ES40
Adc2	ES14
Adaptive Selection of Classifiers	ES37
Adaptive Selection of Optimum Fitness	ES62
Average Probability Ensemble	ES26, ES54, ES70
Average Voting	ES51
Bagging	ES6, ES13, ES16, ES19, ES21, ES24, ES30, ES31, ES36, ES39, ES41, ES42, ES46, ES47, ES50, ES51, ES52, ES55, ES60, ES61, ES62, ES69, ES74, ES76, ES77
Balanced Random Forests	ES2
Best in Training Ensemble	ES24, ES38, ES58
Boosting	ES2, ES16, ES19, ES31, ES32, ES51
Bug Prediction using Deep representation and Ensemble learning	ES74
Cascaded Weighted Majority Voting	ES49
Cascaded Randomized Weighted Majority Voting	ES49
Categorical Boosting	ES67
Combined Defect Predictor	ES20, ES35, ES51
Coding based Multi classifier	ES16, ES33, ES57
Cost-sensitive Forest	ES28
Cost-sensitive Boosting Neural Networks	ES10, ES33
Csb2	ES14
Dagging	ES75, ES77
Data Boost	ES73
DeepForest	ES65
Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples (DECORATE)	ES8, ES75
Double Transfer Boosting	ES23, ES56
Dynamic Adaboost.NC	ES18, ES33, ES44, ES57
Ensemble learning phase in HYDRA	ES35, ES56
Ensemble Random Undersampling	ES44
Ensemble Selection	ES75
Extra Trees	ES67
GcForest	ES65
Gradient Boosting	ES67
Grading	ES75

Table A1 continued

ET	Primary Studies
Histogram based Gradient Boosting	ES67
Improved Transfer Adaptive Boosting	ES56
Kernel Spectral Embedding Transfer Ensemble	ES63
Logistic Model Tree	ES3, ES30, ES46
LogitBoost	ES4, ES30, ES39, ES46, ES47, ES62, ES74, ES77
Maximum Voting	ES51
Metacost	ES14, ES21, ES39, ES61
Multiboost	ES57, ES66, ES75, ES77
Multiple Kernel Ensemble Learning	ES33, ES57
Multischeme	ES77
MSMOTEBoost	ES73
Non Negative Sparse based Semiboost	ES34
Non-linear Decision Tree Forest	ES24, ES38, ES58
Oversampling-based Online Bagging	ES17
Omni Ensemble Learning	ES48
Random Subspace method	ES60
Random Forest	ES1, ES2, ES3, ES4, ES6, ES14, ES16, ES18, ES22, ES23, ES26, ES30, ES32, ES36, ES39, ES42, ES43, ES44, ES46, ES47, ES48, ES50, ES51, ES53, ES55, ES60, ES61, ES62, ES65, ES66, ES67, ES71, ES74, ES77
Rotation Forest	ES19, ES21, ES48, ES75, ES77
Roughly balanced Bagging	ES11
Random Undersampling Boosting (RUSBoost)	ES15, ES21, ES44, ES73
RealAdaBoost	ES75
Randomized Weighted Majority Voting	ES49
Sampling based Online Bagging	ES17
SelectRUSBoost	ES15
SMOTEBoost	ES18, ES21, ES44, ES73
Stacking	ES13, ES25, ES31, ES45, ES54, ES66, ES70, ES77
SysFor	ES28
Transfer Adaptive Boosting	ES56
Transfer Cost-Sensitive Boosting	ES40, ES56
TransferBoost	ES35, ES40
Two Stage Ensemble	ES50, ES59, ES64
Undersampling based Online Bagging	ES17
Validation and Voting Classifier	ES9, ES37
Value Cognitive Boosting with Support Vector Machine	ES32, ES56, ES59
Voting amongst Homogeneous Base Learners (VHomBL)	ES27, ES47, ES62
Voting amongst Heterogeneous Base Learners (VHetBL)	ES5, ES9, ES12, ES13, ES24, ES25, ES38, ES48, ES54, ES55, ES58, ES60, ES66, ES68, ES70, ES72, ES77
Weighted Majority Voting	ES49
WeightedSmoteBoost	ES29
XGBoost	ES67, ES71

Table A2. List of techniques (used as base learners) belonging to each machine learning family

Tree-based Learners	C4.5, Random Tree, Decision Tree, Decision Stump, J48, CART, Alternating Decision Tree, Partial Decision Tree, Tree Disc Classification, Naïve Bayes Tree, REP Tree.
Support Vector Machine	Support Vector Machine, Sequential Minimal Optimization, Voted Perceptron, Pegasos.
Bayesian Learners	Naïve Bayes, Bayesian Network, Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Parzen classifier with the Gaussian kernel, Uncorrelated normal densities based quadratic Bayes.
Rule-based Learners	One Rule, Lines-of-Code, Decision Table, Ripper Down Rules, Repeated Incremental Pruning to Produce Error Reduction.
Instance-based Learners	Locally weighted learning with decision stump, 1-Instance based Learning, K-Instance based Learning, K-Nearest Neighbor, Nearest Mean Classifier, Scaled Nearest Mean Classifier.
Search based Algorithms	Genetic Algorithm, Genetic Programming, Particle Swarm Optimization, Non-Dominated Sorting Genetic Algorithm-II (NSGA-II).
Artificial Neural Networks	Multilayer Perceptron, Radial Basis Function, Linear Perceptron classifier with Batch Processing, Levenberg–Marquardt feed-forward neural network, Automatic Levenberg–Marquardt feed-forward neural network.
Ensemble Learners	RF, BAG, AB, Boosting, XGBoost, Boosting, Gradient Boosting.
Miscellaneous Learners	Voting Feature Interval, KStar, KMeans, Random Subspace, Stochastic Gradient Descent, Minimum Least Square Linear Classifier, Subspace Classifier, Linear classifier based on Principal Component Analysis, Linear Discriminant Classifier, Quadratic Discriminant Classifier, Minimum Linear Least Square Classifier, Linear classifier based on Karhunen Loeve (KL) expansion of common covariance matrix.