Received June 24, 2021, accepted July 5, 2021, date of publication July 8, 2021, date of current version July 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3095559

Software Defect Prediction Using Ensemble Learning: A Systematic Literature Review

FASEEHA MATLOOB[1], TAHER M. GHAZAL[2,3], (Member, IEEE), NASSER TALEB[4], SHABIB AFTAB[1,5], MUNIR AHMAD[5], (Member, IEEE), MUHAMMAD ADNAN KHAN[6], SAGHEER ABBAS[5], AND TARIQ RAHIM SOOMRO[7], (Senior Member, IEEE)

[1]Department of Computer Science, Virtual University of Pakistan, Lahore 44000, Pakistan
[2]Center for Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Selangor 43600, Malaysia
[3]School of Information Technology, Skyline University College, University City of Sharjah, Sharjah, United Arab Emirates
[4]Faculty of Management, Canadian University Dubai, Dubai, United Arab Emirates
[5]School of Computer Science, National College of Business Administration and Economics, Lahore 54660, Pakistan
[6]Pattern Recognition and Machine Learning Laboratory, Department of Software, Gachon University, Seongnam 13557, South Korea
[7]CCSIS, Institute of Business Management, Karachi, Sindh 75190, Pakistan

Corresponding authors: Muhammad Adnan Khan (adnan@gachon.ac.kr) and Munir Ahmad (munir@ncbae.edu.pk)

ABSTRACT Recent advances in the domain of software defect prediction (SDP) include the integration of multiple classification techniques to create an ensemble or hybrid approach. This technique was introduced to improve the prediction performance by overcoming the limitations of any single classification technique. This research provides a systematic literature review on the use of the ensemble learning approach for software defect prediction. The review is conducted after critically analyzing research papers published since 2012 in four well-known online libraries: ACM, IEEE, Springer Link, and Science Direct. In this study, five research questions covering the different aspects of research progress on the use of ensemble learning for software defect prediction are addressed. To extract the answers to identified questions, 46 most relevant papers are shortlisted after a thorough systematic research process. This study will provide compact information regarding the latest trends and advances in ensemble learning for software defect prediction and provide a baseline for future innovations and further reviews. Through our study, we discovered that frequently employed ensemble methods by researchers are the random forest, boosting, and bagging. Less frequently employed methods include stacking, voting and Extra Trees. Researchers proposed many promising frameworks, such as EMKCA, SMOTE-Ensemble, MKEL, SDAEsTSE, TLEL, and LRCR, using ensemble learning methods. The AUC, accuracy, F-measure, Recall, Precision, and MCC were mostly utilized to measure the prediction performance of models. WEKA was widely adopted as a platform for machine learning. Many researchers showed through empirical analysis that features selection, and data sampling was necessary pre-processing steps that improve the performance of ensemble classifiers.

INDEX TERMS Systematic literature review (SLR), ensemble classifier, hybrid classifier, software defect prediction.

I. INTRODUCTION

The ensemble learning model is built by combining the multiple machine learning classifiers to improve prediction performance [2]. According to the literature, many terms, such as hybrid, combined, integrated, and aggregated classification, are employed for ensemble learning [20]–[23]. In the traditional method of defect prediction, an individual classifier, such as the naïve Bayes classifier, decision trees, or a multilayer perceptron, is used to build the prediction model on a pre-labelled dataset. Individual classifiers might have some weaknesses to predict a certain defect under a specific circumstance [24], [25]. For this reason, ensemble learning was applied so that the strengths of multiple classifiers can be combined to provide better defect discovery in the dataset. Many researchers have provided empirical evidence in the last decade, which suggests that ensemble methods provide better classification accuracy than individual classifiers [26]–[30]. Broadly, ensemble methods are classified into two groups based on the types of base

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Liu.

learners: 1) homogeneous ensemble methods and 2) heterogeneous ensemble methods. In homogenous ensemble methods, the same base learners are applied to a different set of instances in a dataset. Examples include bagging, boosting, rotation forest, etc [18]. In the heterogeneous ensemble method, different base learners are generated using different machine learning techniques. These base learners are combined, and final prediction is performed by integrating the results of base learners either statistically or by voting [18]. Heterogeneous methods are more diverse than homogeneous methods due to the different natures of base learners. Ensemble methods can also be categorized as linear and nonlinear. In linear ensemble methods, the output of base learner models is combined using a linear function, such as a weighted average or simple average, while in nonlinear ensemble methods, a nonlinear technique, such as a decision tree or support vector machine (SVM), is applied to combine the decision of base learners [19]. Researchers also take into account diversity while adding multiple classifiers into an ensemble. Diversity of classifiers refers to the notion that the chosen classifiers in an ensemble method make mistakes about different instances of data. Different measures are employed to evaluate the diversity between two classifiers, such as the Correlation Diversity Measure, Q-Statistics, Precision, and Weighted Accuracy & Diversity (WAD). Apart from advancements in ensemble learning techniques, many more promising SDP approaches are being proposed. These approaches aim to predict defects much earlier in the software development lifecycle using the concepts of code smells and requirements smells. Hennning *et al.* [75] proposed a lightweight static requirements analysis approach named Smella that allowed for immediate rapid checks when requirements were written down.

This paper provides a review to reflect the recent research conducted on the use of ensemble learning techniques for software defect prediction. The latest papers published since 2012 are considered for this study. Four renowned and widely employed online search libraries are selected for the extraction of relevant literature, such as ACM, IEEE, Science Direct, and Springer Link. Initially, *3715* papers are extracted, and then 30 most relevant papers are selected as Primary Studies (PS) after following a thorough systematic research process. The remainder of the paper is organized as follows: Section 2 presents the research protocol. Section 3 presents the findings of this review. Finally, section 4 concludes the paper with suggestions for future work.

II. RESEARCH PROTOCOL

A systematic literature review (SLR) is a well-defined systematic process to analyze multiple studies and answer predefined research questions. SLR starts by defining a research protocol that involves the identification of well-defined research questions to be addressed. The process explicitly defines the search strategy and inclusion & exclusion criteria for the selection of relevant studies and guides the extraction of information from each primary study (PS) [33]. The SLR

TABLE 1. Data sources and query results.

Data source	Data searched	Total results
IEEE Xplore	8/01/2021	4,065
ACM	8/01/2021	334
Science Direct	8/01/2021	580
Springer Link	8/01/2021	1179

process is divided into three phases: planning the review, conducting the review, and reporting the review. Each phase consists of sub-phases, as explained in Fig 1. The systematic research process followed in this paper is based on the guidelines provided by [34]–[39].

A. PHASE 1: PLANNING THE REVIEW

The first phase of an SLR provides the initial guidelines to select the relevant studies by defining the following aspects: research questions, data sources, search string, inclusion criteria, exclusion criteria, and quality criteria.

1) RESEARCH QUESTIONS

The purpose of this study is to identify, analyze and summarize the empirical shreds of evidence regarding the use of ensemble learning techniques for software defect prediction by focusing on evaluation criteria, simulation tools, and datasets. These objectives are reflected in the form of research questions, and obtaining the answers to these questions via a critical review is the ultimate goal of an SLR. The research questions identified for this research are given as follows:

RQ1: Which ensemble learning techniques are applied for software defect prediction?

RQ2: Which evaluation criterion is utilized to measure the performance of ensemble learning techniques?

RQ3: What are the most commonly employed tools to implement ensemble learning techniques?

RQ4: Which datasets are used to analyze the performance of ensemble learning techniques?

RQ5: With which technique (s) is the proposed ensemble method compared?

2) DATA SOURCES

Data sources refer to the search space or libraries from where the research studies should be extracted. In this paper, 'ACM digital library', 'IEEE Xplore', 'Springer link' and 'Science Direct' are selected to extract the Primary Studies. The search is performed on a full text of papers. Each electronic database has different options to search for the relevant material. For this reason, the search string is modified to meet the specific requirements of data sources to obtain the most relevant literature. Table 1 shows selected data sources and the number of results generated by search queries.

3) SEARCH STRING

To formulate the search string, particular keywords and their synonyms are selected from the identified research questions,

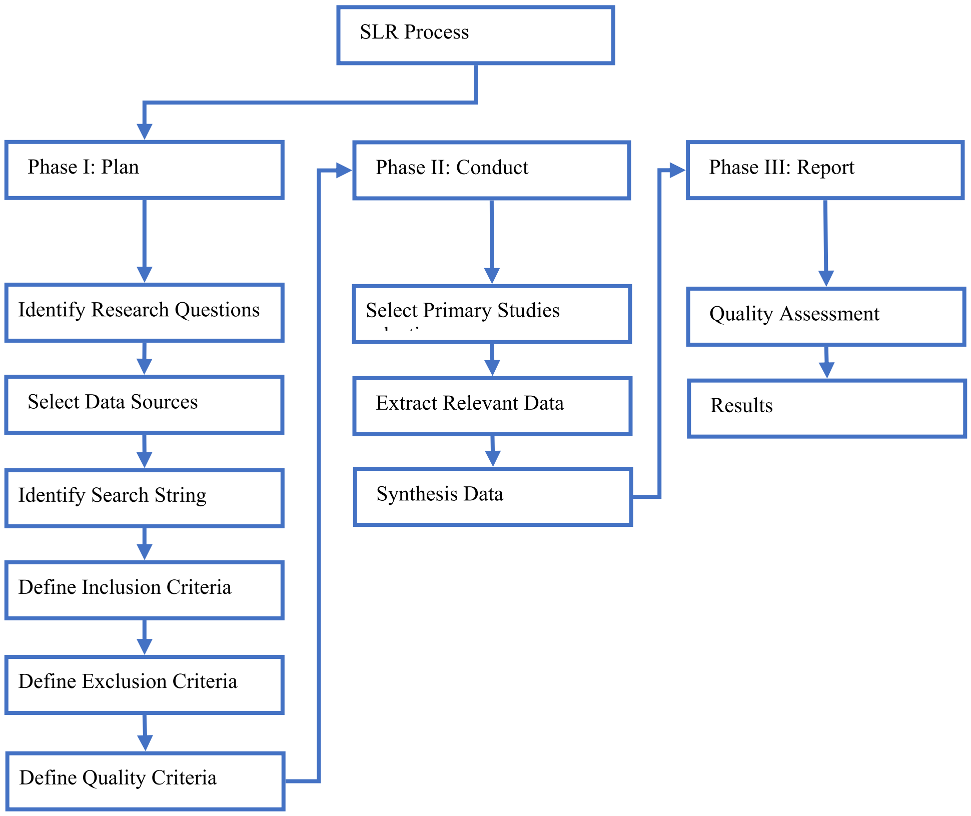

FIGURE 1. SLR process.

TABLE 2. Search string.

Keywords	Alternatives/ synonyms
Software	('program' OR 'system')
Defect	('error' OR 'fault' OR 'bug')
Predict	('estimate' OR 'classify')
Ensemble	('integrated' OR 'hybrid')
Learner	('machine learning algorithm' OR 'algorithm' OR 'classifier' OR 'technique' OR 'method' OR 'model')

as shown in Table 2. The keywords are then arranged with the conditions of 'AND' and 'OR' in a particular sequence to form the following query:

((“software” OR “program” OR “system”) AND (“defect” OR “error” OR “fault” OR “bug”) AND (prediction” OR “estimation” OR “classification”) AND (ensemble OR integrated OR hybrid) AND (learning OR machine learning algorithm OR algorithm OR classifier OR technique OR method OR model))

4) INCLUSION CRITERIA

Studies published in English from 2012 to 2021 are selected for review. The focus is to select studies that have discussed ensemble classifiers for the prediction of software defects. Empirical studies that include experiments on any particular dataset are selected to solve the classification problem of software defects with an ensemble learning approach and a performance evaluation. Studies that present a comparison of ensemble techniques with other techniques are also selected. Preference is given to the latest studies. The selected articles can only belong to journals, conferences, or books.

5) EXCLUSION CRITERIA

Studies that were published before 2012 are excluded. An article whose main focus is not ensemble classifiers or was not written about software defect prediction (SDP) is excluded. Papers that did not include the results of an empirical analysis or did not evaluate the performance of the applied ensemble technique are also excluded.

6) QUALITY CRITERIA

The purpose of establishing the quality criterion is to ensure that selected Primary Studies provide enough details to answer the identified research question. We refer to each quality criterion as 'QA' followed by their number. The QA and data extraction processes are carried out concurrently. A QA checklist to evaluate the eminence of selected primary studies is shown in Table 3.

B. PHASE 2: CONDUCTING THE REVIEW

1) PRIMARY STUDIES SELECTION

Primary Studies are known as the most appropriate articles selected by following the tollgate approach [32] to answer the identified questions. The tollgate approach, which consists of

TABLE 3. QA criteria for the primary study.

S.no.	QA checklist
QA1.	Does the selected study provide enough details regarding the used ensemble classifier, to answer the RQ1?
QA2.	Does the selected study provide enough detail regarding the performance evaluation, to answer the RQ2?
QA3.	Does the selected study provide the detail regarding the simulation tool, to answer the RQ3?
QA4.	Does the study include details of the used dataset, to answer the RQ4?
QA5.	Does the study provide the detail regarding the comparison of ensemble technique with other techniques, to answer the RQ5?

TABLE 4. Tollgate approach.

Selected Sources	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
IEEE Xplore	4,065	31	26	24	18
ACM	334	10	9	8	7
Science direct	580	10	13	12	10
Springer Link	1179	17	14	12	11
Total	3715	68	62	56	46

five phases P-1 to P-5, facilitates the selection of 46 Primary Studies [1]–[12], [40]–[60], [62]–[74], as shown in Table 4. The mentioned quality criteria (Table 3) are followed during the selection of each primary study. The filters of the tollgate phases are given as follows:

Phase 1 (P-1): Initially extracted data by using various combinations of keywords from a search query.

Phase 2 (P-2): Removed duplicates and applied inclusion/exclusion criteria by reading the title.

Phase 3 (P-3): Applied inclusion/exclusion criteria by reading the abstract.

Phase 4 (P-4): Applied inclusion/exclusion criteria by reading the introduction and conclusion.

Phase 5 (P-5): Applied inclusion/exclusion criteria by reading the full text of selected studies. These articles are considered primary studies.

2) DATA EXTRACTION

The extracted data from each primary study includes the following details: proposed/used ensemble learning technique, criteria of performance evaluation, a tool that is used for the implementation of ensemble learning, datasets utilized for the experiments, and the techniques with which proposed/used ensemble learning methods are compared. The selected 46 primary studies are provided in Table 5. Nineteen journal articles, and 21 studies are part of the conference proceedings. Fig 2 shows the distribution of studies over the years.

3) DATA SYNTHESIS

This stage includes the fusion of relevant extracted data, how much data are needed to address each question and how to compile and present the data.

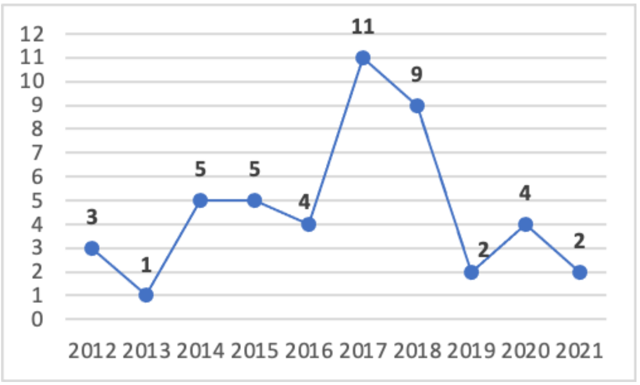

FIGURE 2. Distribution of primary studies over the years.

C. PHASE 3: REPORTING THE REVIEW

1) QUALITY ASSESSMENT

Each selected primary study is assessed against QA criteria (table 3) and assigned a score between 0 and 1, as shown in Table 6. This process of quality assessment is adopted by many researchers in SLRs [76], [77]. If the article explicitly answers the QA question, the study is given a score of 1, and if it partially answers the question, the study is given a score of 0.5. A score of 0 is assigned to studies that fail to answer QA questions. The final score is calculated by summing the scores for all QA questions.

After assessing the quality of selected primary studies, it is found that the score of each primary study $\geq 70\%$ against QA criteria. This finding means that selected primary studies provide adequate information about ensemble learners.

2) RESULTS

The last stage of the systematic research process evaluates the answers to identified research questions after a critical review. The detailed extracted answers from each primary study are discussed in the next section.

III. FINDINGS

This section answers the research questions mentioned in section 1.

RQ1: Which ensemble learning techniques are applied for software defect prediction?

Fig 3 shows the distribution of primary studies in each year according to the type of study. Some studies proposed new methods based on ensemble learning to predict the defects in software, whereas other studies compared the performance of existing techniques to identify better performing techniques. Table 7 summarizes the techniques discussed in selected primary studies. In [53], researchers proposed a multiclass learning method to address imbalanced datasets. First, they converted imbalanced binary class data into balanced multiclass data. Second, some coding-based methods were employed to convert multiclass data into diverse binary data. Last, classification was applied to predict defects. Three coding schemes, six data sampling methods, and four classification algorithms were compared through

TABLE 5. List of selected primary studies using SLR.

Ref.	Year	Library	Type	Journal/ conference Name
[53]	2012	IEEE	Journal	IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)
[58]	2012	Science Direct	Journal	Data mining applications and case study
[59]	2012	IEEE	Journal	IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)
[54]	2013	IEEE	Journal	IEEE Transactions on Reliability
[43]	2014	IEEE	Conference	International Conference on Information Communication and Embedded Systems (ICICES)
[44]	2014	IEEE	Conference	13th International Conference on Machine Learning and Applications.
[45]	2014	IEEE	Conference	Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI)
[46]	2014	IEEE	Conference	International Conference on Advances in Computing, Communications, and Informatics (ICACCI)
[49]	2014	ACM	Journal	The Journal of Machine Learning Research
[40]	2015	Science Direct	Journal	Information and Software Technology
[50]	2015	ACM	Conference	24th Australasian Software Engineering Conference (ASWEC)
[56]	2015	Springer Link	Journal	Automated Software Engineering
[60]	2015	Science Direct	Journal	Information and Software Technology
[62]	2015	Science Direct	Journal	Information Systems
[57]	2016	Springer Link	Journal	Software Quality Journal
[1]	2016	ACM	Conference	Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement
[2]	2016	IEEE	Conference	International Conference on Advances in Electrical, Electronics, Information, Communication, and Bio-Informatics (AEEICB16)
[3]	2016	IEEE	Conference	International Conference on Industrial and Information System (ICIIS)
[51]	2017	Springer Link	Conference	Computer Science On-line Conference
[48]	2017	IEEE	Conference	IEEE International Conference on Software Maintenance and Evolution (ICSME).
[10]	2017	Science Direct	Journal	Information and Software Technology
[11]	2017	Science Direct	Journal	Expert Systems with Applications
[4]	2017	Science Direct	Journal	Knowledge-Based Systems
[5]	2017	IEEE	Conference	Annual Computer Software and Applications Conference
[6]	2017	IEEE	Conference	International Conference on Intelligent Systems and Knowledge Engineering (ISKE)
[7]	2017	IEEE	Conference	International Symposium on Software Reliability Engineering
[63]	2017	ACM	Journal	International Journal of Data Analysis Techniques and Strategies
[64]	2017	IEEE	Conference	International Conference on New Trends in Computing Sciences (ICTCS)
[65]	2017	Springer Link	Journal	Software Quality Journal
[8]	2018	Science Direct	Journal	Alexandria Engineering Journal
[9]	2018	Science Direct	Journal	Information and Software Technology
[12]	2018	ACM	Conference	Proceedings of the 7th International Conference on Software and Information Engineering (ICSIE).
[41]	2018	IEEE	Conference	7th Brazilian Conference on Intelligent Systems (BRACIS)
[42]	2018	IEEE	Journal	IEEE Access
[47]	2018	IEEE	Conference	International Joint Conference on Neural Networks (IJCNN)
[52]	2018	Springer Link	Conference	International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI)
[55]	2018	Springer Link	Conference	Proceedings of the Computational Methods in Systems and Software
[66]	2018	IEEE	Conference	2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)
[67]	2019	ACM	Conference	2nd International Conference on Data Science and Information Technology - DSIT 2019
[68]	2019	Science Direct	Journal	Engineering Science and Technology, an International Journal
[69]	2020	ACM	Conference	Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering
[70]	2020	Springer Link	Journal	Micro-Electronics and Telecommunication Engineering
[71]	2020	Springer Link	Conference	Artificial Intelligence and Evolutionary Computations in Engineering Systems
[72]	2020	Springer Link	Conference	International Conference on Computational Science and Its Applications (ICCSA 2020)
[73]	2021	Springer Link	Journal	Applied Intelligence
[74]	2021	Springer Link	Journal	Neural Computing and Applications

experiments. The results showed that one coding scheme outperformed conventional methods. Moreover, the random forest (RF) performed better than decision trees, Ripper and naïve Bayes (NB). In [54], researchers investigated the impact of different class imbalance methods, including resampling methods, threshold moving, and ensemble learning. Through experiments, it was shown that AdaBoost.NC performed best compared to other methods. Researchers also presented a dynamic version of AdaBoost.NC that was able to automatically adjust its parameters during training. In [49], researchers built a software EnsembleSVM that provided ensembles of instance-weighted SVMs. Their framework

TABLE 6. Quality assessment for primary studies selection.

Criteria	Responding grade	Grade obtained
QA1	{1, 0.5,0} (Yes, nominally, NO)	46 studies (100%)
QA2	{1, 0.5,0} (Yes, nominally, NO)	46 studies (100%)
QA3	{1, 0.5,0} (Yes, nominally, NO)	33 studies (72%)
QA4	{1, 0.5,0} (Yes, nominally, NO)	46 studies (100%)
QA5	{1, 0.5,0} (Yes, nominally, NO)	36 studies (78%)

aimed to reduce the complexity of large-scale nonlinear learning and speed up prototyping of ensemble classifiers. EnsembleSVM was compared to LIBSVM 3.17. Although there was no significant difference in the performance accuracy of the two libraries, EnsembleSVM trained the model faster than LIBSVM. In [50], AdaboostM1, Vote, and StackingC were compared with themselves and base classifiers. NB, Logistic, J48, voted perceptron, and SMO were utilized as base classifiers. The results showed that ensemble models outperformed base classifiers. Among the ensemble models tested, StackingC performed better than others.

In [40], researchers analyzed the effect of feature selection (FS) combined with ensemble learning on the performance of defect prediction. They proposed an average probability ensemble (APE) approach for defect classification. Greedy forward selection (GFS) and correlation-based FS were used to select features in the pre-processing step. Seven base classifiers were employed to form the APE model: RF, gradient boosting, stochastic gradient descent, W-SVMs, logistic regression, multinomial NB, and Bernoulli naive Bayes. The model, evaluated on six datasets, showed that GFS outperformed correlation-based FS, and the APE model achieved the highest AUC compared with W-SVMs and RF. They also proposed an enhanced version of APE, combined with GFS, which further achieved a higher AUC measure. In [41], researchers proposed an evolutionary algorithm based on the general PBIL algorithm to automatically select the best ensemble method and its parameters. This method was referred to as PBIL-Auto-Ens. They compared the proposed method with Auto-Weka and found that the PBIL-Auto-Ens error rate was 11% smaller than Auto-Weka. In [42], different oversampling methods were used to build an ensemble classifier to overcome the effect of minority class data. Three oversampling methods—ROS, MWM, and FIDos—were selected to form an ensemble. Using RF as a base learner and 5-fold cross-validation in the experiments, the results showed that the proposed method achieved a significantly lower false-positive rate compared to individual oversampling techniques. In [58], researchers examined 17 ensembles built using 18 feature ranking techniques. The applied feature ranking techniques include 6 filter-based

TABLE 7. Summary of techniques discussed in primary studies.

Year	Ref.	Techniques
2012	[53]	proposed mutliclass learning method using Coding-Based Ensemble Learning
	[58]	compared 17 ensembles build using 18 feature ranking techniques
	[59]	reviewed state of art ensemble techniques for class imbalance problems
2013	[54]	compared class imbalance methods including resampling methods, threshold moving, and ensemble learning
2014	[43]	proposed bagged SVM model
	[44]	proposed clustering ensemble using Particle Swarm Optimization algorithm (PSO)
	[45]	compared the performance of two feature selection methods (individual and repetitively sampled feature selection) and ensemble classifier (RUSBoost).
	[46]	compared three ensemble methods bagging, boosting, and RF
	[49]	built a software EnsembleSVM that provided ensembles of instance-weighted SVMs.
2015	[40]	analyzed the effect of feature selection (FS) combined with ensemble learning
	[50]	AdaboostM1, Vote, and StackingC were compared with itself and with base classifiers
	[56]	proposed multiple kernel ensemble learning (MKEL) model
	[60]	proposed ELBlocker
	[62]	proposed "CSForest"
2016	[57]	proposed method based on class overlap reduction and ensemble imbalance learning
	[1]	compared the performance of stacking, bagging, and boosting ensemble classifiers with single classification techniques
	[2]	proposed hybrid phase-based ensemble model
	[3]	compared performance of random subspace and rotation forest with base learners
2017	[51]	proposed hybrid SMOTE-Ensemble
	[48]	proposed an Ensemble Multiple Kernel Correlation Alignment (EMKCA)
	[10]	proposed TLEL, a two-layer ensemble learning approach
	[11]	compared Linear Regression-based Combination Rule (LRCR) and Gradient Boosting Regression-based Combination Rule (GRCR)
	[4]	compared heterogeneous and homogeneous ensemble methods and linear and non-linear combination rules.
	[5]	performed analysis of FS and three ensemble learning methods

feature ranking techniques, the signal-to-noise filter technique, and 11 threshold-based feature ranking techniques.

TABLE 7. *(Continued.)* **Summary of techniques discussed in primary studies.**

2018	[6]	combined FS and Data Balancing (DB) with ensemble techniques
	[7]	proposed SmoteNDBoost and RusNDBoost
	[63]	compared the bagging, boosting, and stacking ensembles. 11 base classifiers were used
	[64]	RF was combined with feature selection and data sampling
	[65]	analyze whether different classifiers identify the same defects or not using RF, NB, RPart, and SVM
2018	[8]	analyzed ensembles of weighted randomized majority voting techniques
	[9]	proposed SDAEsTSE model
	[12]	analyzed performance model with and without applying SMOTE, AdaBoost, and Bagging
	[41]	Proposed PBIL-Auto-Ens technique
	[42]	proposed ensemble ROS, MWM, and FIDos methods with RF as base classifier
	[47]	proposed multi-objective optimization for ensemble classification
	[52]	proposed framework based on PCA in FS and RF, Adaboost, bagging, and classification via regression ensembles
	[55]	proposed enhancement in the SMOTE-Ensemble approach using cost-sensitive learning (CSL)
	[66]	proposed deep super learner (DSL)
2019	[67]	compared Adaboost, Bagging, RSM, RF, and Vote ensembles
	[68]	ten ensemble classifiers were compared to baseline classifiers
2020	[69]	compared seven Tree-based ensembles
	[70]	proposed a three-step framework using RF and XGBoost CPDP.
	[71]	proposed a model to predict defects across projects with an HDP
	[72]	proposed a method using SMOTE and homogeneous ensemble methods (bagging and boosting)
2021	[73]	compared seven ensemble techniques, namely Dagging, Decorate, Grading, MultiBoostAB, RealAdaBoost, Rotation Forest, and Ensemble Selection.
	[74]	proposed a model based on feature selection, feature extraction, class balancing and ensemble learning

The results showed that no single ensemble method outperformed others in all datasets. However, the researchers observed that the ensembles of a few ranking techniques performed better than the ensembles of many ranking techniques. In [59], a review of state-of-the-art ensemble techniques for class imbalance problems was conducted. The researchers provided a taxonomy of class balancing techniques based on base ensemble learning algorithms and how they address class imbalance problems. They concluded that ensemble-based methods showed significantly improved results. Moreover, RUSBoost or UnderBagging achieved higher performances. They also showed that sampling techniques worked better with the bagging ensemble learning algorithm. In [43], researchers performed bagging with an SVM on class-level and package-level metrics. The proposed model was compared with the baseline SVM classifier. The results suggested that the bagged SVM achieved better ROC values compared to the SVM. In [44], researchers proposed a clustering ensemble using PSO for defect classification. Three clustering algorithms—K-means, PSO, and Expectation maximization—were employed. K-means and PSO were utilized with two similarity measures viz. Euclidean and Manhattan similarity function. The PSO algorithm outperformed other algorithms when acting as a consensus function and clustering, in terms of average accuracy.

In [60], researchers proposed ELBlocker to identify blocking bugs. They divided a training dataset into equal-sized disjoint sets and then built classifiers on each disjoint set. Using a random forest, these classifiers were combined to compute the likelihood score for bugs to be blocking bugs. In [62], researchers proposed "CSForest": an ensemble of decision trees and CSVoting to minimize the classification cost. Further, they incorporated SMOTE and Safe-level-Smote in CSForest to solve the class imbalance problem. In [2], a hybrid phase-based ensemble model was proposed to predict metric relationships and defects on multiple associated products. The model involves three steps. In the first step, an integrated SDLC phase-based dataset from multiple projects was generated. In the second step, missing values were filtered using a phase-based preprocessing algorithm. The third step involved identifying relevant patterns using phase clustering and a classification algorithm. Patterns were extracted using hybrid decision tree construction. The proposed model outperformed base classifiers, such as ANN, SVM, NB, and RF. In [45], researchers compared the performance of two feature selection methods (individual and repetitively sampled feature selection) and ensemble classifier (RUSBoost). An MLP and SVM were employed in boosting and regular learning processes. The results showed that repetitively sampled feature selection achieved better performance with a plain learner in the learning process and that boosting resulted in better classification performance than without boosting. In [46], three ensemble methods—bagging, boosting, and RF—were compared. Fifteen base classifiers were utilized in ensembles. Experiments were performed on nine open-source datasets from the PROMISE repository. The results showed that AUC performance gain for bagging, boosting, and RF was achieved on 10, 6, and 12 base learners. RF outperformed bagging and boosting with no AUC performance loss. NB, logistic regression, and voted feature interval learners were not recommended as base learners for bagging, boosting, and RF. In [47], multi-objective optimization for ensemble classification was proposed. A multi-objective genetic algorithm was employed to choose base learners from a pool of 900 classifiers. Two objectives were diversity and prediction error. The researchers stated that while some classifiers, such as ANN and KNN, were more commonly selected than other classifiers, the results still depend on the type of dataset applied. Moreover, a very small number of classifiers were selected for the final ensemble from a large pool of 900 classifiers.

In [48], researchers proposed an Ensemble Multiple Kernel Correlation Alignment (EMKCA)-based approach for heterogeneous defect prediction. First, source and target project data were mapped into a high-dimensional kernel space using multiple kernel classifiers. Second, they used the kernel correlation alignment method to similarly distribute the source and target project's data. Last, kernel classifiers were integrated with ensemble learning to remove the impact of class imbalance. The EMKCA was compared with benchmark methods on 30 datasets; it outperformed in the majority of the cases. The researchers also analyzed the effect of ensemble learning by calling a version of EMKCA without ensemble KCA. The results showed that ensemble classifiers generally outperformed each sub-classifier. In [63], researchers compared the bagging, boosting, and stacking ensembles. Eleven base classifiers, including NB, Bayes net, SMO, PART, J48, RF, Random Tree, IB1, Decision Table, and an NB tree in each of the three ensembles. Their experimental analysis showed that boosting performed better than bagging. The researchers also concluded that an RF was an important algorithm in stacking and that it should be stacked with other classifiers to improve performance. In [64], an RF (ensemble of trees that vote for the class) was combined with feature selection and data sampling. They compared bat search, genetic search, and ant search algorithms in the feature selection step. For class balancing, instance resampling with replacement was utilized. In the classification step, a Fuzzy Unordered Rule Induction Algorithm (FURIA), MLP, NB, KStar, and RF algorithm were employed. All three FS methods showed improved performance with the RF compared with other classifiers, whereas bat search showed better total performance on the datasets. In [65], researchers analyzed the prediction uncertainty using four classifiers: RF, NB, RPart, and SVM. They aimed to analyze whether different classifiers identify the same defects. The results showed that the four algorithms were comparable in predictive performance. However, the subset of defects identified by each classifier was different. Some classifiers were consistent, while others vary in their prediction in multiple runs. Based on these results, researchers strongly suggested the use of ensemble classifiers. In [51], a hybrid SMOTE-Ensemble approach was presented. In this method, different ensembles were applied to a classification task. For this purpose, bagging, RF, and AdaBoost were selected. J48 was employed as a base classifier for bagging and Adaboost. In the second step, the best ensemble classifier was selected and applied again on an over-sampled dataset using SMOTE. The experimental results showed that the SMOTE-Ensemble method substantially improved the prediction performance. In [55], an enhancement in the SMOTE-Ensemble approach was presented using cost-sensitive learning (CSL). This method used SMOTE for data balancing and CSL and AdaBoost with J48 as a base learner in the classification step. The model was compared with DT, AdaBoost with DT, and SMOTE-AdaBoost with DT. The results showed that CSL on the hybrid SMOTE-AdaBoost method outperformed other methods in terms of G-mean. In [52], a framework in which PCA was used for feature selection and ensemble learning for classification was proposed. Four ensembles were employed: RF, Adaboost, bagging, and classification via regression. An improvement in prediction accuracy of 0.6% was observed with reduced features, whereas bagging achieved the highest average accuracy of 91.49% among all ensembles. In [56], researchers combined multiple kernel learning and ensemble learning and proposed a model referred to as multiple kernel ensemble learning (MKEL). By using metrics from open-source software, they obtained a multiple kernel classifier through ensemble learning. Thirty base kernels that mapped a dataset into a high-dimensional kernel space were utilized. Boosting was used in the ensemble process. The experimental results suggested that MKEL was affected by the misclassification cost, and therefore, tended to maintain a higher Pd value. Although MKEL did not achieve the best Pf value for most of the datasets, it achieved comparatively better results than other classification methods. In [57], researchers combined class overlap reduction and ensemble imbalance learning to predict software defects. The neighborhood cleaning learning (NCL) rule was applied to remove overlapped samples. Ensemble random undersampling was then performed on the whole dataset to generate a balanced dataset. Experiments were performed on nine highly unbalanced datasets. The results showed that the proposed method achieved the best total performance in terms of G-mean and AUC among tested models. Moreover, it achieved relatively higher Pd than other methods.

The authors in [1] compared the performance of stacking, bagging, and boosting ensemble classifiers with single classification techniques for defect prediction. In stacking, they used four base classifiers: NB, C4.5 Decision Tree, K-Nearest Neighbor (KNN), and Sequential Minimal Classification (SMO). They calculated diversity between pairs of classifiers using Precision, Weighted Accuracy & Diversity (WAD), Diversity (DIV), and the Matthews Correlation Coefficient (MCC). The results showed that the DIV technique, bagging and boosting outperform the Single classifier technique by 27.2%, 8.9%, and 18.5%. DIV and WAD techniques attained an average relative increase of 0.52 in true positives compared to the bagging technique. Researchers also concluded that adding more than three classifiers while building a stacking ensemble did not improve the performance of the ensemble. In [3], researchers predicted the defects in software projects using five different ensemble methods, including bagging, boosting, random subspace, rotation forest, and stacking. Three base learners—linear regression (LR), multilayer perceptron (MLP), and decision tree regression (DTR) were used to build ensembles. The results were compared using ARE and AAE values. Two datasets—Random Subspace with DTE and MLP—performed best, while the third dataset—Rotation Forest—had the lowest AAE value. Random subspace with MLP had the lowest ARE value. Researchers concluded that Random Subspace and Rotation Forest performed better than bagging, boosting, and stacking, and DTR as a base learner performed best in predicting results. In [4], researchers compared heterogeneous and homogeneous ensemble methods and linear and nonlinear combination rules. They evaluated 6 defect prediction techniques—DTR, LR, MLP, genetic programming (GP), negative binomial regression (NBR), and zero-inflated Poisson regression (ZIP)—on 15 datasets and selected the top 3 techniques—DTR, MLP, and LR—as base learners for ensemble models. The researchers compared the proposed heterogeneous ensemble model with homogeneous bagging and boosting ensembles with M5P as a base learner. For most of the datasets, heterogeneous ensemble methods based on nonlinear combination rules outpaced homogeneous ensemble methods. However, heterogeneous ensemble methods based on linear combination rules did not show significant differences compared to homogeneous ensemble methods. They also concluded that nonlinear-based ensemble methods were more stable and performed better in predicting the number of defects compared to linear combination rule-based ensembles.

In [5], an empirical analysis of feature selection and ensemble learning was performed. Researchers used three heterogeneous ensemble models—BTE, MVE, and NDTF—with two linear combinations (best in training and voting) and one nonlinear combination (DTR) rule. They also considered 5 classification techniques—LOGR, ANN, RBFN-RAN, RBFN-FCM, and RBFNKMC—to build a fault prediction model. An analysis was performed on 45 datasets and concluded that MVE performed best and ensemble methods outpaced single classifiers. It was also observed that feature selection improved the performance of the proposed ensemble methods. In [6], researchers combined FS and Data Balancing (DB) with ensemble techniques. Information Gain (IG) was utilized for FS, and the SMOTE method was employed to balance data. Two ensemble methods—bagging and AdaBoost.M1 with J48 DT as a base learner—were selected. AUC and accuracy performance measures of established ensemble methods with IG and SMOTE showed that bagging outpaced Adaboost.M1 in most of the datasets. Researchers in [7] investigated resampling and ensemble techniques to improve the prediction performance of a model. They used SMOTE and RUS to balance instances in the dataset and one ensemble

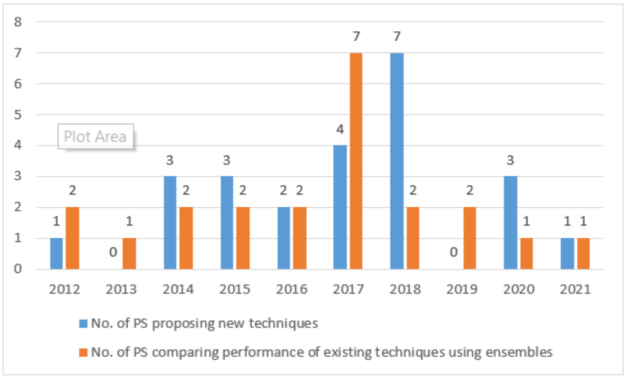

FIGURE 3. Number of techniques reported each year grouped by type of study.

method, AdaBoost.R2. The researchers proposed two hybrid algorithms, namely, SmoteNDBoost and RusNDBoost, and compared them with individual techniques. They utilized three regression models as base learners: DTR, Bayesian Ridge Regression (BRR), and LR. The results showed that RusNDBoost and SmoteNDBoost performed better than SmoteND, RusND, and AdaBoost.R2 in most cases. Through fault-percentile-average (FPA) values, the RusNDBoost performed better than SmoteNDBoost with DTR and LR as a base learner and subsequently showed better results with BRR. In [8], researchers proposed SDP models based on weighted randomized majority voting techniques. They compared models build on change and static code metrics and discovered that change metrics showed better performance than static code metrics and a combined set of these metrics. The proposed model consisted of two steps. The first step was a metric selection, and the second step created an ensemble classifier using 4 types of weighted voting majority techniques. These techniques were the Weighted Majority Voting (WM), Randomized Weighted Majority Voting (RWM), Cascading Weighted Majority Voting (CWM), and Cascading Randomized Weighted Majority Voting (CRWM) techniques. Five base classifiers—NB, C4.5, LR, SVM, and RF—were also employed in the building ensemble. The results showed that CRWM based on 5 classifiers performed best in almost all cases. In [9], a two-phase defect prediction model, SDAEsTSE, was proposed. This model was based on Stacked De-noising Auto Encoders (SDAEs) and Two-Stage Ensemble Learning (TSE). The first phase was the deep learning phase to extract deep representations, and the second phase was the ensemble learning phase, in which logistic regression was employed as base learners of Bagging and AdaBoost. TSE was compared with baseline methods, where RF, bagging, and AdaBoost outperformed in the majority of cases. SDAEsTSE was compared with state-of-the-art defect prediction models—RF, Bayesian networks (BN), (NB), and AdaBoost_NN (AB_NN)—which outpaced in terms of the AUC, MCC, and F-measure in the majority of cases.

In [10], researchers proposed TLEL, a two-layer ensemble learning approach to leverage ensemble methods by hybridizing bagging and stacking. In this model, the first layer used bagging with DT to build the RF model; in the second layer, many RF models were combined using stacking to predict just-in-time defects. The proposed model was compared with three baseline methods: Deeper, DNC, and MKEL. The results showed that TLEL achieved a higher percentage of buggy instances when 20% of the code was reviewed. Four models were also compared based on precision and recall: TLEL showed better performance than the other three methods across five datasets. In [66], researchers performed a replication study to verify the results obtained by [10]. They also proposed a deep super learner (DSL), in which they generalized the original approach by using any arbitrary set of classifiers in the ensemble, optimized the weights of the classifiers, and allowed additional layers. The experimental results showed that DSL achieved a significantly better F1 score than the original method. In [11], two combination rules for combining base learner output were analyzed by researchers. The authors presented the Linear Regression-based Combination Rule (LRCR), which is a linear combination rule, and the Gradient Boosting Regression-based Combination Rule (GRCR), which is a nonlinear combination rule approach. To select the base learner for the ensemble method, they compared five prediction techniques: LR, MLP, GP, ZIP, and NBR. Comparative analysis showed that LR, GP, and MLP had a lower AAE and ARE than the other two techniques, and therefore, were selected as base learners for ensemble methods. A relative comparison of the LRCR and GRCR combination rule-based ensemble methods showed that the GRCR performed better than the LRCR across multiple datasets. In [12], researchers reinforced two rules while building an ensemble method. The first rule stated that the output of base learners should be combined in a way that strengthens correct decisions and disregards incorrect decisions. The second rule stated that diverse classifiers should be employed as base learners. They analyzed the impact on the prediction performance of the defect prediction model with and without applying SMOTE, AdaBoost, and bagging as meta classifiers and NB, MLP, and J48 as base classifiers. The results of the comparison showed that the stacking ensemble with SMOTE provided better performance than that without SMOTE. The choice of a meta classifier did not show any significant differences in the performance of the ensemble model.

In [67], researchers used Adaboost, bagging, RSM, RF, and Vote ensembles to analyze defect prediction. J48 was utilized as a base learner for these ensembles. In the first step, an optimal ensemble (i.e., RF) was selected using experimentation. In the next step, SMOTE and Resample were applied for class balancing on the dataset, and classification was performed using an optimal ensemble from the previous step. The results obtained after classification confirmed the improvement in performance by combining sampling techniques with an ensemble classifier. In [68], ten ensemble classifiers were compared to baseline classifiers. The evaluated ensemble learning algorithms were adaBoostM1, LogicBoost, Multiboost AB, Bagging, RF, Dagging, Rotation forest (ROF), stacking, multi scheme, and voting. Base classifiers included NB, LR, MLP, RBF, SMO, Pegasos, Voted Perceptron, Instance-based Learner, KStar, Jrip, OneR, PART, J48, CART, Hyperpipes, and Voting Feature Intervals. The results showed that the ensemble classifier did increase the prediction performance compared to a base classifier. Among the ensembles, RF was mentioned as a highly developed ensemble classifier. Other successful ensembles include ROF, Logic Boost, Adaboost, and Voting. Moreover, for ROF, AB, and RF, it was shown that increasing the number of base classifiers enhanced performance.

In [69], researchers analysed and compared the prediction performance of seven Tree-based ensembles in defect prediction. Two bagging ensembles, i.e., random forest and Extra Trees, and five boosting ensembles, i.e., Ada boost, Gradient Boosting, Hist Gradient Boosting, XGBoost and CatBoost, were employed. The empirical results showed the better performance of Tree-based bagging ensembles over Tree-based boosting ensembles. However, in prediction performance, none of the Tree-based ensembles was significantly lower than individual decision trees. Moreover, Adaboost ensemble was the worst performing ensemble among all Tree-based ensembles. In [70], researchers used RF and XGBoost to propose a defect prediction model for Cross-project defect prediction (CPDP). They propose a three-stage framework. In the first stage, PCA for dimensionality reduction of the dataset into two components was applied. In the second phase, SMOTE was applied to solve the class imbalance problem. The ensemble classifiers RF and XGBoost were applied. Experimental analysis showed that the proposed framework performed better than some baseline methods.

In [71], reseachers proposed a model to predict defect across projects with a heterogeneous metric set (HDP). They selected datasets and removed the common metric to make it heterogeneous in nature. Subsequently, steps such as feature selection, metric matching, maximum weighted bipartite matching, feature transformation and ensemble learning techniques, were applied. They applied a voting ensemble classifier with 11 base classifiers. Their technique showed promising results with the highest AUC of 0.93 in one group of source and target datasets. In [72], researchers proposed a method using SMOTE and homogeneous ensemble methods (bagging and boosting) to improve the performance of defect prediction models. They employed DT and BN as baseline classifiers in their model. Their experimental results showed that the proposed method significantly outperformed base classifiers.

In [73], researchers empirically accessed the performance of seven ensemble techniques, namely, Dagging, Decorate, Grading, MultiBoostAB, RealAdaBoost, Rotation Forest, and Ensemble Selection. Naive Bayes, logistic regression, and J48 (decision tree) were used as base learners. An experimental analysis showed that for most cases, the Rotation Forest yielded better performance compared to other ensemble techniques. MultiBoostAB, Decorate, and Dagging produced better performance in some cases. This finding concluded that J48 as a base learner improved prediction performance, whereas NB as a base learner generally resulted in the inferior performance of the ensemble techniques. In [74], researchers proposed a model based on feature selection, feature extraction, class balancing and ensemble learning. First, they compared FS techniques, such as Recursive Feature Elimination (RFE), Correlation-based feature selection, Lasso, Ridge, ElasticNet and Boruta. Logistic regression, decision Trees, K-nearest neighbor, support vector machines and ensemble learning. RFE performed better for most of the datasets. Therefore, the proposed method combined PLS Regression with RFE. Five models were created using PLS, RFE, SMOTE and one of the five best performing algorithms, i.e., XGBoost, Stacking, Random Forest, Extra Trees and AdaBoost. The results showed that XGBoost and Stacking gives better results.

Summary: Many ensemble learning methods were proposed from 2012 to 2021. Researchers proposed hybrid frameworks using ensemble learning, feature selection, and class imbalance techniques. Frequently utilized ensemble methods are the random forest, boosting, and bagging. Less commonly employed methods in our selected primary studies include stacking, voting and Extra Trees. Frameworks such as EMKCA, SMOTE-Ensemble, MKEL, SDAEsTSE, TLEL, and LRCR with improved classification performance were proposed using ensemble learning methods.

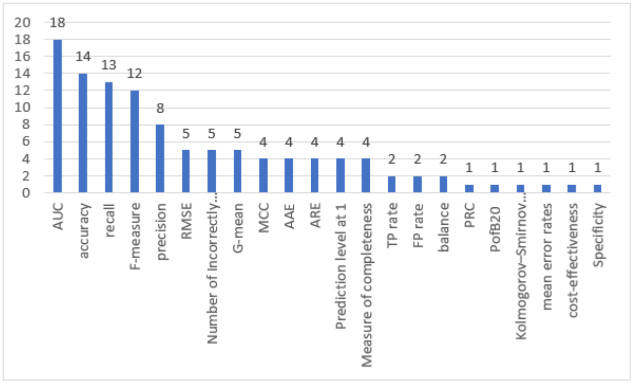

FIGURE 4. 4 Performance measure and corresponding number of primary studies.

RQ2: Which evaluation criterion is used to measure the performance of ensemble learners?

Scholars used various combinations of performance measures to evaluate the prediction ability of ensemble learners. Fig 4 shows the distribution of studies over performance measures. The APE model in [40] and performance results of the proposed model in [46]–[48], [53], [58], [59], [63] were evaluated using AUC. In [42], performance was compared using AUC, TP-rate, FP-rate, and F-measure. In [43], the root mean square error (RMSE) and AUC were used to compare performance. In [1], the MCC was employed to evaluate bagging, boosting, and stacking ensembles. Researchers in [9] applied the F-measure and AUC in addition to the MCC. In [12], the MCC, TP-rate, FP-rate, Precision, Recall, F-measure, AUC, and PRC were employed to evaluate the heterogeneous stacking model. Researchers in [3], [4], [7], [11] used the Average Absolute Error (AAE) and Average Relative Error (ARE) as a performance measure in addition to other measures, such as RMSE, prediction at level l, and measure of completeness. In another study [50], performance measure recall, precision, accuracy, and F-value were used to evaluate and compare results. Other performance measures that were frequently applied include accuracy in [5]–[6], [8], [44], [49], [52], and [64]; PofB20 by [10]; and the Kolmogorov–Smirnov (K-S) test in [11]. In [41], the results of Auto-Weka and Auto-PBIL-Ens models were compared using mean error rates. In [51], [55], accuracy, recall (Pd), Number of Incorrectly Predicted cases with No Defects (PF), and G-mean were utilized as performance measures. In [54], [57], AUC, recall, Pf, and G-mean and balance were used for evaluation. In [56], Pd, Pf, and F-measure were applied. In [65], the MCC and F-measure were used to compare the performances of classifiers against 600 defect prediction performances reported in published studies. In [60], the F1 score and cost-effectiveness were used to measure the efficiency of ELBlocker. In [62], weighted precision and weighted recall were utilized to compare the cost of classifiers. In [67], precision rate, f-measure and AUC were applied to evaluate the defect prediction model. [68] used F-measure and AUC only to evaluate performance. In [69], AUC and accuracy were employed as a measure to compare performance. In [70], [74], researchers used Accuracy, precision, Recall and f-measure to compare the performance of their proposed models. In [71], the AUC, recall, precision and F-measure were used to measure the performance of the proposed ensemble learning technique. In [72], the AUC, Accuracy and F-measure are employed to measure the performance of the proposed defect prediction model. In [73], five performance measures, namely precision, recall, AUC (area under ROC curve), specificity, and G-means were used.

A brief description of the performance measures used in Primary Studies is detailed as follows: MCC takes into account all true and false positives and negatives to determine the quality of binary classification [14]. AAE is the difference between the actual value and the predicted value of several defects. AAE shows the closeness of the predicted value to the actual value. ARE takes into account the size of the object that is being measured and determines the size of the absolute error. The K-S test is performed to determine the goodness-of-fit that evaluates how effective an ensemble method is at predicting defects. The F-measure is the harmonic mean of precision and recall, while the AUC is a trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR), which indicates the effectiveness of the classifier to correctly predict classes. PofB20 is the percentage of defects that can be obtained by inspecting 20% of the line of code [13]. Recall (TPR) measures the ratio of correctly classified positive instances to the actual number of positive instances, where precision is a measure of correctly classified positive instances out of all positive instances [31].

To further analyze and compare results, researchers also utilized different comparison and measurement techniques, such as box plot analysis [11] of the AAE and ARE measure to identify variations in the samples; W/D/L [7] (win/ draw/loss) to compare the number of datasets on which their proposed method performed the best, same or worse than other methods; the Wilcoxon signed-rank test [4], [5], [9], [11] to compare the classification technique and selected source of metrics to identify the best performer [4], [11]; the two-tailed Friedman's test to measure the difference between prediction models and Cliff's delta [9] to determine the effect of size.

Summary: Most common performance measures include AUC, accuracy, F-measure, Recall, Precision, and MCC. While less common measures include RMSE, Average Absolute Error (AAE), Average Relative Error (ARE), FRP, G-mean, Pf, Pd, and PofB20.

RQ3: What are the most commonly employed tools to implement ensemble learning techniques?

Various data mining tools have been developed to perform predictive analysis on data using different machine learning classifiers. These tools have the ability to identify some meaningful patterns in data to gain hidden knowledge [15]. Each tool comes has different capabilities; therefore, researchers choose tools according to their selected machine learning methods. Fig 5 shows the distribution of primary studies over machine learning tools and libraries. In our selected primary studies, we determined that Waikato Environment for Knowledge Analysis (WEKA) is the most frequently utilized machine learning tool to explore, visualize and perform classification over data [1], [3], [4], [6], [8], [9], [11], [12], [41], [43], [45], [50]–[55], [58], [60], [62]–[64], [67], [68], [73]. Other tools include MATLAB [9], [11], [47], [48] and sklearn [7], which is a python machine learning library and LIBSVM [56]. In [40], all evaluated algorithms were implemented in python language. However, the machine learning tool was not mentioned in the study. In [59], KEEL software was used to perform classification on KEEL datasets. In [69], researchers applied scikit-learn, CatBoost and XGBoost libraries to build tree-based ensembles.

Summary: WEKA was adopted by the majority of the studies. Other tools include MATLAB, LIBSVM, KEEL, and sklearn

RQ4: Which datasets are used to analyze the performance of ensemble learning techniques?

A dataset is a collection of historical software data that is applied to predict bugs in code before the release of software.

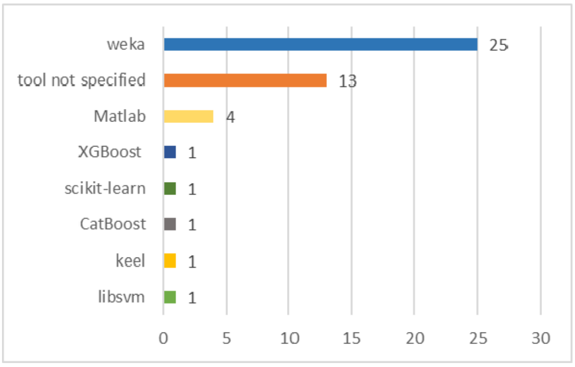

FIGURE 5. Distribution of primary studies over machine learning tool/ Library.

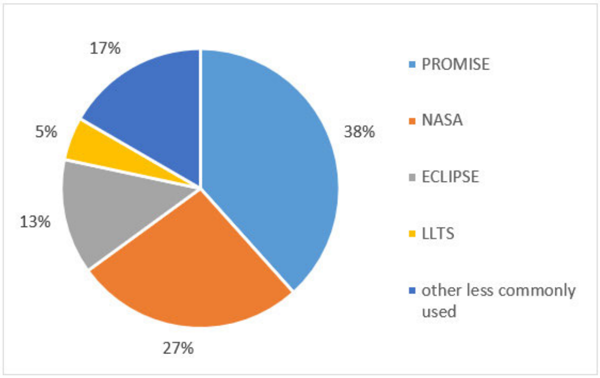

FIGURE 6. Distribution of studies over types of dataset repositories.

Scholars discovered that different classification algorithms perform better on different datasets. Therefore, most of the selected studies are using multiple sets of datasets from a different domain to manifest the potency of the proposed ensembles. Fig 6 shows the distribution of studies over dataset repositories used in selected primary studies. Most of the selected primary studies used a publicly available dataset from PROMISE repository [1], [4]–[7], [11], [12], [42], [50], [51], [54], [55], [57], [63], [64], [68], [73]. This repository contains datasets with three types of software metrics, namely, CK Object-Oriented metrics, Halstead/McCabe procedural code metrics, and some other static code metrics. In [4], researchers have presented details of five projects' datasets viz., Xerces, Camel, Xalan, Ant, and PROP. These projects were written in java and C++. They selected 15 releases of these five projects, which contained more than 500 software modules. These datasets contained several object-oriented (OO) metrics that were determined to perform better for fault prediction problems [16], [17]. Scholars in a study [11] also utilized the same set of datasets in addition to the Jedit software project. They selected 11 releases of 6 projects, which contained more than 300 software modules. Researchers in [5] compared models on 45 releases of 18 software projects, namely, Ant, arc, berek, camel, e-learning, ivy, jedit, kalkulator, log4j, lucene, pdftranslator, prop, redactor, serapion, synapse, termoproject, velocity, and xereces. In [1], researchers analyzed the performance of prediction models on 4 software projects: Ant, Jedit, Tomcat and Xalan. In [7], researchers implemented ensemble models on 22 releases of 6 software projects: Ant, Camel, Jedit, Synapse, Xalan, and Log4j. These datasets contained 20 independent software metrics and 1 dependent number of defect variables. PC3, JM1, CM1, and KC1 were employed in [51]. In [50], 12 datasets from the PROMISE repository were utilized: Ant, Camel, e-learning, Forest, Jedit, Tomcat, Xalan, Xerces, Zuzel, Berek, Pbean2, and velocity. In [40], the research used Ant 1.7 and Camel 1.6 in addition to the NASA datasets KC3, MC1, PC2, and PC4. In [43], the Eclipse package level dataset and NASA KC1 dataset were used to conduct classification experiments. In [44], CM1, KC1, and KC2 datasets were selected from the NASA dataset repository. In [12], researchers applied only one dataset, PC1, which was created by the NASA Metric Data Program (MDP), which contained 22 features. This dataset had a very small number of truly defective software, making it suitable for a class imbalance problem study. In [6], researchers selected datasets that had McCabe and Halstead Static Code Metrics. They selected 8 NASA software project datasets; AR1, AR4, JM1, KC2, MC1, MW1, PC3, and PC4. Researchers in [9] used 12 public NASA datasets from the PROMISE and tera-PROMISE repositories, including CM1, KC1, KC2, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, and JM1. In [52], 5 datasets from NASA MDP—PC2, PC3, CM1, KC3, and MW1— were employed. In [53], all 14 datasets from NASA were applied. Similarly, in [54], 12 NASA datasets were used. In [62], MC1, MC2, KC1, PC1, and PC2 from NASA MDP utilized. In [67], 5 datasets from NASA MDP—CM1, JM1, KC1, KC2, and PC1—were selected. Researchers in [3, 46] used software data from three releases of Eclipse projects, which was an IDE developed by IBM. These datasets comprised 32 software metrics related to various source code and structural measures. Researchers in [8] applied the dataset of Apache open-source java projects found in the Apache subversion website. In [10], [66], researchers build a defect prediction model for just time prediction on six large-scale change-based software projects' datasets, i.e., Bugzilla, Columba, Eclipse JDT, Eclipse Platform, Mozilla, and PostgreSQL. These datasets also contained imbalanced class data. In [58], 16 datasets were employed to evaluate ensembles of feature ranking techniques, including LLTS, Eclipse, and KC1. In [59], 44 datasets from the KEEL repository were utilized. In [60], datasets of 6 open-source software projects—Freedesktop, Chromium, Mozilla, NetBeans, OpenOffice, and Eclipse—were applied.

In [65], 12 NASA datasets, 3 open-source datasets—Ant, Ivy, and Tomcat—and 3 commercial telecommunication datasets were employed to perform an analysis of datasets with different metric granularity and software domains. Researchers in [41] and [47] utilized 15 datasets and 11 datasets, respectively, from the UCI machine learning repository for classification purposes. In [49], researchers selected covtype and ijcnn1 data sets to evaluate the EnsembleSVM library. In [48], 30 projects were collectively obtained from NASA, Softlab, Relink, AEEEM, and PROMISE repository. In [69], 11 datasets of NASA software projects were employed; in [72], 5 datasets of NASA software projects were utilized. In [70], researchers selected five projects with a homogeneous set of metrics from http://openscience.us and www.Promisedata.org. In [71], three datasets each were selected from NASA, AEEEM, ReLink and SOFTLAB repositories. They removed common metrics from the selected datasets so that they become heterogeneous in nature. In [73], 28 dataset were selected from the PROMISE repository. Datasets of several open-source software systems, such as Apache Camel, Apache Xerces, Apache Xalan, PROP, etc., were selected. In [74], CM1, PC1, KC1 and KC2 datasets were obtained from the PROMISE and NASA repository.

Summary: Most of the researchers applied datasets from the PROMISE and NASA MDP repositories. Some researchers used the Eclipse dataset developed by IBM. Others used datasets from the UCI machine learning repository, Apache, Bugzilla, Columba, Mozilla and PostgreSQL, Softlab, Relink, and AEEEM software projects.

RQ5: With which technique (s) is the proposed ensemble method is compared?

To measure the effect and potency of proposed ensemble approaches, researchers compared the performance with individual classifiers. After analyzing selected primary studies, we can infer that ensemble methods showed better performance than individual classifiers in a majority of the cases. The proposed model in [53] was compared to RUS, ROS, SMOTE, cost-sensitive learning, bagging, and boosting. The proposed model outperformed all methods except bagging, with which comparable results were achieved. In classification, RF outperformed C4.5, NB, and Ripper. In [59], researchers compared ensemble methods for class imbalance problems with classic ensembles and the nonensemble classifier (C4.5 and SMT). The results showed that ensemble methods significantly improved the performances compared to the results obtained by the mere use of preprocessing techniques. In [60], ELBlocker employed an ensemble of multiple classifiers to predict the likelihood of a blocking bug. The model was compared with Garcia and Shihab's method [61], SMOTE, one-sided selection (OSS), and Bagging. ELBlocker achieved F1 and ER@20% scores of 0.345 and 0.668, respectively. The improvement in ELBlocker over Garcia and Shihab's method, SMOTE, OSS, and bagging was statistically significant. In [62], proposed CSForest and CSVoting were compared with two cost-sensitive classifiers, named Weighting and CSTree, and two cost-insensitive classifiers named C4.5 and SysFor. The proposed technique achieved the lowest classification cost in all 6 datasets. In [54], five data balancing techniques were compared with NB and RF as learners. These techniques were

RUS, Balanced RUS, threshold moving, SMOTEBoost, and AdaBoost.NC (DNC). DNC achieved the best balance and AUC in 8 of 10 cases.

In [63], the performance of bagging, boosting, and stacking ensembles were compared to 11 base classifiers, which were also employed as base classifiers in these ensembles. Using the Wilcoxon signed ranked test, they showed that the performances of NB, Bayes net, PART, RF, IB1, VFI, decision table, and NB tree base learners, were the same as those of ensemble learner classifiers. However, for boosted SMO, bagged J48 and boosted and a bagged random tree performance increase was observed compared to base classifiers. In [64], the performance of RF was compared to 4 classifiers: Fuzzy Unordered Rule Induction Algorithm (FURIA), MLP, NB, and KStar on the PC1, PC2, PC3, and PC4 datasets. Bat search, genetic search, and ant search were applied in the feature selection step. All three feature selection methods worked best with RF compared to other classifiers, whereas the combination of bat search and RF stood out among all cases. In [50], stacking, vote, and AdaBoost ensembles were compared with each other and base classifiers. Although there was variation in the performance of classifiers on different datasets, on average StackingC outperformed other ensembles. When compared with base classifiers, it was observed that either performance of ensemble remained the same or was improved, except for j48 for the tomcat dataset. The proposed model APE in [40] was compared with W-SVMs and RF. APE achieved the highest performance in all six datasets applied in the study. APE achieved the highest AUC 0.91 in the PC2 dataset, and enhanced APE achieved the highest accuracy of 0.98 in the MC1 dataset. No conventional method outperformed the proposed model in any dataset. In [42], the proposed ensemble oversampling technique was compared with the individual sampling methods ROS, MWM, and FIDos. The results showed that the proposed ensemble method achieved the lowest FP-Rate at 39% and achieved the highest AUC values in 68% of cases among all datasets and all approaches. In [43], the Bagged SVM was compared with the baseline SVM model using package level and class level metrics. The bagged SVM achieved AUC values of 0.78 and 0.832 for package-level metrics and class level metrics, respectively, whereas the SVM achieved AUC values of 0.721 and 0.798 for package-level metrics and change level metrics, respectively. Researchers in [47] compared the error score of the proposed model with the error score reported by three recent papers. Through comparison, it was shown that the proposed approach achieved lower classification errors in most cases.

In [48], proposed EMKCA was compared to WPDP, NN-filter, VCB, SSTCA+ISDA, CPDP-IFS, CCA+, HDP-KS, and CCT-SVM. The results compared using the AUC in 30 projects indicated that EMKCA outperformed in most cases. EMKCA attained the highest mean AUC among all classifiers. In [49], researchers compared EnsembleSVM and LIBSVM library on two datasets, covtype and ijcnn1, using accuracy measures. For covtype, the EnsembleSVM accuracy was 3% lower than LIBSVM; for ijcnn1, EnsembleSVM was 0.2% better than LIBSVM. The results showed no significant difference in performance. However, the difference in training time between the two libraries is significant. The EnsembleSVM trained the model in 35 sec and 0.3 sec for the covtype dataset and ijcnn1 dataset, respectively, whereas, LIBSVM trained the models in 728 sec and 9.5 sec for both datasets. In [51], an experiment was performed in three steps; 1) NB, MLP, and J48 were applied on datasets; 2) ensemble classifiers RF, bagging, and AdaBoost were applied, and 3) SMOTE combined with the best ensemble from the previous stage was applied. The results showed that SMOTE with ensemble obtained better results compared to other methods. Moreover, AdaBoost achieved the best performance among ensembles on four datasets. These results were further compared with the method proposed in [55], where a CSL was employed with AdaBoost in the classification step. The results showed that CSL on hybrid SMOTE-AdaBoost achieved the highest G-mean in all four datasets among DT, AdaBoost, and SMOTE-AdaBoost. In [56], MKEL was compared with coding-based ensemble learning [53], AdaBoost.NC [54], weighted Naïve Bayes, Compressed C4.5 decision tree (CC4.5), Cost-sensitive Boosting Neural Network (CBNN), and Asymmetric Kernel Principal Component Classification (AKPCC). The average Pd and Pf values of MKEL on the 12 NASA datasets were 0.68 and 0.26, which was higher than the corresponding values of all other methods. MKEL also achieved a higher F-measure than other methods with an average of 0.48 on 12 datasets. In [56], the proposed method was compared with ensemble random under-sampling without NCL, NB, RFNB+log filter, Dynamic AdaBoost.NC, SMOTE+NB, RUS+NB, SMOTEBoost, and RUSBoost. The results showed that more complicated imbalance methods, such as ERUS, DNC, SMOTEBoost, and RUSBoost, generated better results than NB. With a relatively higher Pd and the best total performance in terms of AUC and G-mean, the proposed method confirmed the importance of implementing NCL as a preprocessing step.

Researchers in [1] compared NB, KNN, SMO, j48 with WAD and DIV stacking ensembles. The results showed that WAD and DIV showed better performance by 18.2% for NB, 31.4% for KNN, 22.3% for SMO, and 20.1% for J48. These results were further fortified by the Wilcoxon significance test. In a study [3], the researchers showed the improved performance of the ensemble method: random subspace and rotation forest over base learners for all three datasets. However, they also showed that in some cases, base learners provided better performance; for example, stacking with MLP and boosting with MLP showed the worst performance than the individual base learner. The comparison results were presented in the study [5], which also strengthened the conclusion drawn in other primary studies that ensemble methods outperformed in prediction performance compared to single classifiers.

In [8], NB, LR, C4.5, SM, and RF were compared with multiple ensembles learners; it is found that in almost all cases, CRWM performed better in terms of f-measure, accuracy, and AUC values. In [9], researchers performed a comparative analysis on the proposed approach SDAEsTSE and the baseline approaches on each of the selected datasets in terms of F1, AUC, and MCC using the W/D/L method. The results showed that in the majority of the experiments, the ensemble approach SDAEsTSE outperformed the baseline methods. In [10], TLEL was compared with three baseline methods: Deeper, DNC, and MKEL. The results showed that the proposed approach could discover 70% of just-in-time defects by analyzing only 20% of the line of codes, while the baseline method DNC achieved a maximum value of 55% defective changes. Comparative results in the study [11] also reinforce the conclusion drawn from RQ5. The results showed that GBCR performed better than individual baseline methods in most cases. In [12], researchers compared the base learners MLP, NB, and J48 with and without the SMOTE technique and found no significant difference in the classifier's performance in both cases. However, when stacking was applied with SMOTE, an eminent performance improvement was observed in defect prediction models.

In [67], five ensemble classifiers—Adaboost, Bagging, RSM, RF, and Vote—were compared among themselves and to a single classifier J48. J48 was also used as a base learner in the ensembles. Using precision rate, F-measure and AUC, it was shown that the classification performance of all ensembles was better than that of the single classifier. Moreover, RF outperformed other ensembles. In [68], ten ensemble classifiers were compared to 16 baseline classifiers. The ensemble learning algorithms that were evaluated included adaBoostM1, LogicBoost, Multiboost AB, Bagging, RF, Dagging, Rotation forest (ROF), stacking, multi scheme, and voting. Base classifiers included NB, LR, MLP, RBF, SMO, Pegasos, Voted Perceptron, Instance-based Learner, KStar, Jrip, OneR, PART, J48, CART, Hyperpipes, and Voting Feature Intervals. The results were compared using the F-measure and AUC. The average highest F-measure among base classifiers was 0.802 by J48, whereas the average highest AUC was 0.743 by MLP. The average highest F-measure and AUC of 0.808 and 0.776, respectively, among the ensemble classifiers was achieved by ROF.

In [69], researchers compared the prediction performance of individual decision tree classifiers to tree-based ensembles, including Extra Trees (ET), XGBoost, Cat-Boost, Gradient Boosting, Hist Gradient Boosting, AdaBoost and RF. They discovered that the decision tree classifier performed worst for all defect datasets, whereas Tree-based ensembles showed a consistent higher prediction performance, except the Ada ensemble. Among all ensembles, RF and ET bagging ensembles showed a higher prediction performance. In [70], researchers proposed a three-step framework for CPDP. They compared its performance with traditional within-project defect prediction (WPDP) and found that the highest accuracy achieved using this framework in CPDP is 0.91, which is comparable to the highest accuracy of 0.93 obtained for WPDP. They concluded that PCA and SMOTE improved the performance of the ensemble classifier in CPDP.

In [72], the performance of DT and BN was compared to a SMOTE-based homogeneous ensemble method. In this method, bagging and boosting ensembles were employed with DT and BN as base classifiers. They concluded that SMOTE improved the performances of classifiers. Moreover, BoostedDT + SMOTE achieved the highest average accuracy of 86.8%. The total results suggested that BoostedBN + SMOTE and BoostedDT + SMOTE were superior to BaggedBN + SMOTE and BaggedDT + SMOTE. In [73], seven ensemble techniques were compared on 28 datasets. The experimental analysis revealed that the rotation forest with J48 as the base learner achieved the highest precision, recall, and G-mean values of 0.995, 0.994, and 0.994, respectively and Decorate achieved the highest AUC value of 0.986. In [74], machine learning classifiers, such as MLP, LR, DT, KNN, SVM, RF, ET, Bagging, AdaBoost, Gradient Boosting, XGBoost and Stacking, were compared. Stacking and XGBoost outperformed all other classifiers in all four datasets in terms of precision, recall, accuracy and F-measure. XGB and Stacking achieved the highest accuracy of 0.968 in the PC1 dataset.

Summary: In a majority of the research papers, researchers compared the performance of ensemble learning/hybrid frameworks with base classifiers, such as DT, MLP, SVM, NB, and KNN. The performance of the models were also compared with and without cases of feature selection and class imbalance techniques. SMOTE and RUS were mostly employed for data sampling. Our review showed that in most cases, ensemble learning techniques performed better than base classifiers. Moreover, our study revealed that feature selection and data sampling were important pre-processing steps that improve the performance of classifiers.

IV. CONCLUSION

In this paper, an SLR is conducted to track the most recent research advances in ensemble learning techniques for software defect prediction. This review is performed after critically analyzing the most relevant research papers published in three well-known online libraries ACM, IEEE, Springer Link, and Science Direct. Five research questions regarding the different aspects of research progress on the use of ensemble learning techniques for software defect prediction are defined and addressed in this study. It is concluded that ensemble learning techniques performed significantly better than individual classifiers. In the future, a review of the effects of feature selection techniques on ensemble learning should be performed. Moreover, the diversity of classifiers, while building the ensemble model, should also be investigated to improve the effectiveness of the ensembles.

REFERENCES

[1] J. Petrić, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, "Building an ensemble for software defect prediction based on diversity selection," in *Proc. 10th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, Sep. 2016, pp. 1–10.

[2] A. N. R. Moparthi and B. D. N. Geethanjali, "Design and implementation of hybrid phase based ensemble technique for defect discovery using SDLC software metrics," in *Proc. 2nd Int. Conf. Adv. Electr., Electron., Inf., Commun. Bio-Inform. (AEEICB)*, Feb. 2016, pp. 268–274.

[3] S. S. Rathore and S. Kumar, "Ensemble methods for the prediction of number of faults: A study on eclipse project," in *Proc. 11th Int. Conf. Ind. Inf. Syst.*, Dec. 2016, pp. 540–545.

[4] S. S. Rathore and S. Kumar, "Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems," *Knowl.-Based Syst.*, vol. 119, pp. 232–256, Mar. 2017.

[5] L. Kumar, S. Rath, and A. Sureka, "An empirical analysis on effective fault prediction model developed using ensemble methods," in *Proc. Int. Annu. Comput. Softw. Appl. Conf.*, vol. 1, Jul. 2017, pp. 244–249.

[6] C. W. Yohannese, T. Li, M. Simfukwe, and F. Khurshid, "Ensembles based combined learning for improved software fault prediction: A comparative study," in *Proc. 12th Int. Conf. Intell. Syst. Knowl. Eng.*, Nov. 2017, pp. 1–6.

[7] X. Yu, J. Liu, Z. Yang, X. Jia, Q. Ling, and S. Ye, "Learning from imbalanced data for predicting the number of software defects," in *Proc. Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2017, pp. 78–89.

[8] S. Moustafa, M. Y. ElNainay, N. E. Makky, and M. S. Abougabal, "Software bug prediction using weighted majority voting techniques," *Alexandria Eng. J.*, vol. 57, no. 4, pp. 2763–2774, Dec. 2018.

[9] H. Tong, B. Liu, and S. Wang, "Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning," *Inf. Softw. Technol.*, vol. 96, pp. 94–111, Apr. 2018.

[10] X. Yang, D. Lo, X. Xia, and J. Sun, "TLEL: A two-layer ensemble learning approach for just-in-time defect prediction," *Inf. Softw. Technol.*, vol. 87, pp. 206–220, Jul. 2017.

[11] S. S. Rathore and S. Kumar, "Towards an ensemble based system for predicting the number of software faults," *Expert Syst. Appl.*, vol. 82, pp. 357–382, Oct. 2017.

[12] 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 *Proc. 7th Int. Conf. Softw. Inf. Eng.*, 2018, pp. 44–47.

[13] S. Wang, T. Liu, J. Nam, and L. Tan, "Deep semantic feature learning for software defect prediction," *IEEE Trans. Softw. Eng.*, vol. 46, no. 12, pp. 1267–1293, Dec. 2020.

[14] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using Matthews correlation coefficient metric," *PLoS ONE*, vol. 12, no. 6, pp. 1–17, 2017.

[15] A. Sharma and B. Kaur, "A research review on comparative analysis of data mining tools, techniques and parameters," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 7, pp. 523–529, 2017.

[16] J. Petrić, D. Bowes, T. Hall, and N. Baddoo, "The jinx on the NASA software defect data sets," in *Proc. 20th Int. Conf. Eval. Assessment Softw. Eng.*, 2016, pp. 1–5.

[17] M. O. Elish, A. H. Al-Yafei, and M. Al-Mulhem, "Empirical comparison of three metrics suites for fault prediction in packages of object-oriented systems: A case study of Eclipse," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 852–859, 2011.

[18] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa, "Ensemble approaches for regression: A survey," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 1–40, 2012.

[19] S. E. Lacy, M. A. Lones, and S. L. Smith, "A comparison of evolved linear and non-linear ensemble vote aggregators," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 758–763.

[20] L. Lam and C. Y. Suen, "Optimal combinations of pattern classifiers," *Pattern Recognit. Lett.*, vol. 16, no. 9, pp. 945–954, 1995.

[21] J. Kittler, M. Hater, and R. P. W. Duin, "Combining classifiers," in *Proc. Int. Conf. Pattern Recognit.*, 1996, vol. 2, no. 3, pp. 897–901.

[22] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, Jan. 2000.

[23] C. Cortes, Y. LeCun, V. Vapnik, H. Drucker, and L. D. Jackel, "Boosting and other ensemble methods," *Neural Comput.*, vol. 6, no. 6, pp. 1289–1301, 2008.

[24] K. Stąpor, "Evaluating and comparing classifiers: Review, some recommendations and limitations," in *Proc. Int. Conf. Comput. Recognit. Syst.*, 2017, pp. 12–21.

[25] H. Hosseini, B. Xiao, M. Jaiswal, and R. Poovendran, "On the limitation of convolutional neural networks in recognizing negative images," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 352–358.

[26] C. D. Stefano, F. Fontanella, G. Folino, and A. S. D. Freca, "A Bayesian approach for combining ensembles of GP classifiers," in *Proc. Int. Workshop Multiple Classifier Syst.*, 2011, pp. 26–35.

[27] L. Rokach, "Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography," *Comput. Statist. Data Anal.*, vol. 53, no. 12, pp. 4046–4072, 2009.

[28] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1619–1630, Oct. 2006.

[29] J. Canul-Reich, L. Shoemaker, and L. O. Hall, "Ensembles of fuzzy classifiers," in *Proc. IEEE Int. Fuzzy Syst. Conf.*, Jul. 2007, pp. 1–6.

[30] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A comparison of decision tree ensemble creation techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 173–180, Jan. 2007.

[31] K. Punitha and S. Chitra, "Software defect prediction using software metrics—A survey," in *Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES)*, 2013, pp. 555–558.

[32] W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Inf. Softw. Technol.*, vol. 51, no. 6, pp. 957–976, 2009.

[33] S. Keele, "Guidelines for performing systematic literature reviews in software engineering, version 2.3," Dept. Comput. Sci., School Comput. Sci. Math., Softw. Eng. Group, Keele Univ., Keele, U.K., Tech. Rep. EBSE-2007-01, 2007.

[34] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, pp. 7–15, Jan. 2008.

[35] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering version 2.3," *Engineering*, vol. 45, no. 4, p. 1051, 2007.

[36] F. Anwer and S. Aftab, "Latest customizations of XP: A systematic literature review," *Int. J. Mod. Educ. Comput. Sci.*, vol. 9, no. 12, pp. 26–37, 2017.

[37] S. Ashraf and S. Aftab, "Scrum with the spices of agile family: A systematic mapping," *Int. J. Mod. Educ. Comput. Sci.*, vol. 9, no. 11, pp. 58–72, 2017.

[38] S. Ashraf and S. Aftab, "Latest transformations in scrum: A state of the art review," *Int. J. Mod. Educ. Comput. Sci.*, vol. 9, no. 7, pp. 12–22, 2017.

[39] M. Ahmad, S. Aftab, M. S. Bashir, and N. Hameed, "Sentiment analysis using SVM: A systematic literature review," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 2, pp. 182–188, 2018.

[40] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Inf. Technol.*, vol. 58, pp. 388–402, Feb. 2015, doi: 10.1016/j.infsof.2014.07.005.

[41] J. C. Xavier-Junior, A. A. Freitas, A. Feitosa-Neto, and T. B. Ludermir, "A novel evolutionary algorithm for automated machine learning focusing on classifier ensembles," in *Proc. 7th Brazilian Conf. Intell. Syst. (BRACIS)*, 2018, pp. 462–467, doi: 10.1109/bracis.2018.00086.

[42] S. Huda, K. Liu, M. Abdelrazek, A. Ibrahim, S. Alyahya, H. Al-Dossari, and S. Ahmad, "An ensemble oversampling model for class imbalance problem in software defect prediction," *IEEE Access*, vol. 6, pp. 24184–24195, 2018, doi: 10.1109/access.2018.2817572.

[43] A. Shanthini and R. M. Chandrasekaran, "Analyzing the effect of bagged ensemble approach for software fault prediction in class level and package level metrics," in *Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES)*, 2014, pp. 1–5, doi: 10.1109/icices.2014.7033809.

[44] R. A. Coelho, F. D. R. Guimaraes, and A. A. Esmin, "Applying swarm ensemble clustering technique for fault prediction using software metrics," in *Proc. 13th Int. Conf. Mach. Learn. Appl.*, 2014, pp. 356–361, doi: 10.1109/icmla.2014.63.

[45] T. M. Khoshgoftaar, K. Gao, and A. Napolitano, "Improving software quality estimation by combining feature selection strategies with sampled ensemble learning," in *Proc. IEEE 15th Int. Conf. Inf. Reuse Integr. (IEEE IRI)*, Aug. 2014, pp. 428–433, doi: 10.1109/iri.2014.7051921.

[46] A. Kaur and K. Kaur, "Performance analysis of ensemble learning for predicting defects in open source software," in *Proc. Int. Conf. Adv. Comput., Commun., Informat. (ICACCI)*, 2014, pp. 219–225, doi: 10.1109/icacci.2014.6968438.

[47] S. Fletcher, B. Verma, Z. M. Jan, and M. Zhang, "The optimized selection of base-classifiers for ensemble classification using a multi-objective genetic algorithm," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2018, pp. 1–8, doi: 10.1109/ijcnn.2018.8489467.

[48] Z. Li, X.-Y. Jing, X. Zhu, and H. Zhang, "Heterogeneous defect prediction through multiple kernel learning and ensemble learning," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2017, pp. 91–102, doi: 10.1109/icsme.2017.19.

[49] M. Clasen, F. D. Smet, J. A. K. Suykens, and B. D. Moor, "EnsembleSVM: A library for ensemble learning using support vector machines," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 141–145, 2014.

[50] S. Hussain, J. Keung, A. A. Khan, and K. E. Bennin, "Performance evaluation of ensemble methods for software fault prediction," in *Proc. ASWEC 24th Australas. Softw. Eng. Conf. (ASWEC)*, vol. 2, Sep. 2015, pp. 91–95, doi: 10.1145/2811681.2811699.

[51] H. Alsawalqah, H. Faris, I. Aljarah, L. Alnemer, and N. Alhindawi, "Hybrid SMOTE-ensemble approach for software defect prediction," in *Software Engineering Trends and Techniques in Intelligent Systems* (Advances in Intelligent Systems and Computing). Cham, Switzerland: Springer, 2017, pp. 355–366, doi: 10.1007/978-3-319-57141-6_39.

[52] N. Dhamayanthi and B. Lavanya, "Improvement in software defect prediction outcome using principal component analysis and ensemble machine learning algorithms," in *Proc. Int. Conf. Intell. Data Commun. Technol. Internet Things (ICICI)* (Lecture Notes on Data Engineering and Communications Technologies). Cham, Switzerland: Springer, 2018, pp. 397–406, doi: 10.1007/978-3-030-03146-6_44.

[53] Z. Sun, Q. Song, and X. Zhu, "Using coding-based ensemble learning to improve software defect prediction," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1806–1817, Nov. 2012, doi: 10.1109/tsmcc.2012.2226152.

[54] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Trans. Rel.*, vol. 62, no. 2, pp. 434–443, Jun. 2013, doi: 10.1109/tr.2013.2259203.

[55] I. Abuqaddom and A. Hudaib, "Cost-sensitive learner on hybrid smote-ensemble approach to predict software defects," in *Computational and Statistical Methods in Intelligent Systems* (Advances in Intelligent Systems and Computing). Cham, Switzerland: Springer, 2018, pp. 12–21, doi: 10.1007/978-3-030-00211-4_2.

[56] T. Wang, Z. Zhang, X. Jing, and L. Zhang, "Multiple kernel ensemble learning for software defect prediction," *Automated Softw. Eng.*, vol. 23, no. 4, pp. 569–590, 2015, doi: 10.1007/s10515-015-0179-1.

[57] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Tackling class overlap and imbalance problems in software defect prediction," *Softw. Qual. J.*, vol. 26, no. 1, pp. 97–125, 2016, doi: 10.1007/s11219-016-9342-6.

[58] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, "Software measurement data reduction using ensemble techniques," *Neurocomputing*, vol. 92, pp. 124–132, Sep. 2012, doi: 10.1016/j.neucom.2011.08.040.

[59] 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 Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012, doi: 10.1109/tsmcc.2011.2161285.

[60] X. Xia, D. Lo, E. Shihab, X. Wang, and X. Yang, "ELBlocker: Predicting blocking bugs with ensemble imbalance learning," *Inf. Softw. Technol.*, vol. 61, pp. 93–106, May 2015, doi: 10.1016/j.infsof.2014.12.006.

[61] H. V. Garcia and E. Shihab, "Characterizing and predicting blocking bugs in open source projects," in *Proc. 11th Work. Conf. Mining Softw. Repositories*. Washington, DC, USA: IEEE Computer Society, 2014, pp. 72–81.

[62] 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," *Inf. Syst.*, vol. 51, pp. 62–71, Jul. 2015, doi: 10.1016/j.is.2015.02.006.

[63] I. Alazzam, I. Alsmadi, and M. Akour, "Software fault proneness prediction: A comparative study between bagging, boosting, and stacking ensemble and base learner methods," *Int. J. Data Anal. Techn. Strategies*, vol. 9, no. 1, p. 1, 2017, doi: 10.1504/ijdats.2017.10003991.

[64] D. R. Ibrahim, R. Ghnemat, and A. Hudaib, "Software defect prediction using feature selection and random forest algorithm," in *Proc. Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2017, pp. 252–257, doi: 10.1109/ictcs.2017.39.

[65] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: Do different classifiers find the same defects?" *Softw. Qual. J.*, vol. 26, no. 2, pp. 525–552, 2017, doi: 10.1007/s11219-016-9353-3.

[66] S. Young, T. Abdou, and A. Bener, "A replication study: Just-in-time defect prediction with ensemble learning," in *Proc. IEEE/ACM 6th Int. Workshop Realizing Artif. Intell. Synergies Softw. Eng. (RAISE)*, May/Jun. 2018, pp. 42–47.

[67] R. Li, L. Zhou, S. Zhang, H. Liu, X. Huang, and Z. Sun, "Software defect prediction based on ensemble learning," in *Proc. 2nd Int. Conf. Data Sci. Inf. Technol. (DSIT)*, Jul. 2019, pp. 1–6, doi: 10.1145/3352411.3352412.

[68] F. Yucalar, A. Ozcift, E. Borandag, and D. Kilinc, "Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability," *Eng. Sci. Technol., Int. J.*, vol. 23, no. 4, pp. 938–950, Aug. 2020, doi: 10.1016/j.jestch.2019.10.005.

[69] H. Aljamaan and A. Alazba, "Software defect prediction using tree-based ensembles," in *Proc. 16th ACM Int. Conf. Predictive Models Data Anal. Softw. Eng. (PROMISE)*. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 1–10, doi: 10.1145/3416508.3417114.

[70] 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* Lecture Notes in Networks and Systems, vol. 106, D. Sharma, V. Balas, L. Son, R. Sharma, and K. Cengiz, Eds. Singapore: Springer, 2020, doi: 10.1007/978-981-15-2329-8_31.

[71] A. A. Ansari, A. Iqbal, and B. Sahoo, "Heterogeneous defect prediction using ensemble learning technique," in *Artificial Intelligence and Evolutionary Computations in Engineering Systems* (Advances in Intelligent Systems and Computing), vol. 1056, S. Dash, C. Lakshmi, S. Das, and B. Panigrahi, Eds. Singapore: Springer, 2020, doi: 10.1007/978-981-15-0199-9_25.

[72] A. O. Balogun, F. B. Lafenwa-Balogun, H. A. Mojeed, V. E. Adeyemo, O. N. Akande, A. G. Akintola, A. O. Bajeh, and F. E. Usman-Hamza, "SMOTE-based homogeneous ensemble methods for software defect prediction," in *Computational Science and Its Applications—ICCSA 2020* (Lecture Notes in Computer Science), vol. 12254, O. Gervasi *et al.*, Eds. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-58817-5_45.

[73] S. S. Rathore and S. Kumar, "An empirical study of ensemble techniques for software fault prediction," *Int. J. Speech Technol.*, vol. 51, no. 6, pp. 3615–3644, Jun. 2021, doi: 10.1007/s10489-020-01935-6.

[74] S. Mehta and K. S. Patnaik, "Improved prediction of software defects using ensemble machine learning techniques," *Neural Comput. Appl.*, pp. 1–12, Mar. 2021, doi: 10.1007/s00521-021-05811-3.

[75] H. Femmer, D. M. Fernández, S. Wagner, and S. Eder, "Rapid quality assurance with requirements smells," *J. Syst. Softw.*, vol. 123, pp. 190–213, Jan. 2017, doi: 10.1016/j.jss.2016.02.047.

[76] A. Qazi, R. G. Raj, G. Hardaker, and C. Standing, "A systematic literature review on opinion types and sentiment analysis techniques: Tasks and challenges," *Internet Res.*, vol. 27, no. 3, pp. 608–630, 2017, doi: 10.1108/IntR-04-2016-0086.

[77] A. Qazi, F. Hussain, N. A. Rahim, G. Hardaker, D. Alghazzawi, K. Shaban, and K. Haruna, "Towards sustainable energy: A systematic review of renewable energy sources, technologies, and public opinions," *IEEE Access*, vol. 7, pp. 63837–63851, 2019, doi: 10.1109/ACCESS.2019.2906402.

FASEEHA MATLOOB received the M.S. degree in computer science from the Virtual University of Pakistan with a focus on software engineering. Her research interests include software engineering and data mining.

TAHER M. GHAZAL (Member, IEEE) received the B.Sc. degree in software engineering from Al Ain University, in 2011, the M.Sc. degree in information technology management from The British University in Dubai, in 2013, associated with The University of Manchester and The University of Edinburgh, and the Ph.D. degree in IT/software engineering from Damascus University, in 2019. He is currently pursuing the Ph.D. degree in information science and technology from Universiti Kebangsaan Malaysia. He served in engineering, computer science, ICT, and head of stem and innovation departments, and involved in quality assurance, accreditation, and data analysis, in several governmental and private educational institutions under KHDA, Ministry of Education, and the Ministry of Higher Education and Scientific Research, United Arab Emirates. He has more than ten years of extensive and diverse experience as an instructor, a tutor, a researcher, a teacher, an IT support/specialist engineer, and a business/systems analyst. His research interests include the IoT, IT, artificial intelligence, information systems, software engineering, Web developing, building information, modeling, quality of education, management, big data, quality of software, and project management. He is also actively involved in community services in the projects and research field.

NASSER TALEB received the M.Sc. degree in computer science from Minnesota State University, USA, in 1991, and the Ph.D. degree in IT from Salford University, U.K., in 2005. He has 25 years of teaching experience in the field of information technology management with the Faculty of Management, Canadian University Dubai, and research experience with Ajman University and Al Ain University. His most recent publication is the *Technology Education Management Informatics* Journal. His current research interests include blockchain and bitcoin cryptocurrency technology, cloud computing, and big data. He served as the Deputy Dean of the College of Business, Al Ain University.

SHABIB AFTAB received the M.S. degree in computer science from the COMSATS Institute of Information Technology, Lahore, Pakistan, and the M.Sc. degree in information technology from the Punjab University College of Information Technology (PUCIT) Lahore. He is currently pursuing the Ph.D. degree in computer science with the National College of Business Administration & Economics. He is also serving as a Lecturer in computer science with the Virtual University of Pakistan. His research interests include data mining and software process improvement.

MUNIR AHMAD (Member, IEEE) received the Master of Computer Science degree from the Virtual University of Pakistan, in 2018. He is currently pursuing the Ph.D. degree with the School of Computer Science, National College of Business Administration & Economics. He has spent several years in industry. He is also working as an Executive Director or the Head of IT Department, United International Group, Lahore, Pakistan. He has vast experience in data management and efficient utilization of resources at multinational organizations. He has conducted many research studies on sentiment analysis and utilization of AI for prediction on various healthcare issues. His research interests include data mining, big data, and artificial intelligence.

MUHAMMAD ADNAN KHAN received the B.S. and M.Phil. degrees from International Islamic University, Islamabad, Pakistan, and the Ph.D. degree from ISRA University, Islamabad, in 2016. He is currently working as an Assistant Professor with the Pattern Recognition and Machine Learning Laboratory, Department of Software, Gachon University, Republic of Korea. Before joining Gachon University, he worked in various academic and industrial roles in Pakistan. He has been teaching graduate and undergraduate students in computer science and engineering for the past 12 years. He is also guiding five Ph.D. and seven M.Phil. students. He has published more than 190 research articles with Cumulative JCR-IF and more than 240 in international journals and reputed international conferences. His research interests include machine learning, MUD, image processing and medical diagnosis, and channel estimation in multi-carrier communication systems using soft computing. He received scholarship awards from the Punjab Information Technology Board, Government of Punjab, Pakistan, for his B.S. and M.Phil. degrees, and the Higher Education Commission, Islamabad, for his Ph.D. degree, in 2016.

SAGHEER ABBAS received the M.Phil. degree in computer science from the School of Computer Science, NCBA&E, Lahore, Pakistan, and the Ph.D. degree from the School of Computer Science, NCBA&E, in 2016. He is currently working as an Associate Professor with the School of Computer Science, NCBA&E. He has been teaching graduate and undergraduate students in computer science and engineering for the past eight years. He has published about 48 research articles in international journals and reputed international conferences. His research interests include cloud computing, the IoT, intelligent agents, image processing, and cognitive machines, with various publications in international journals and conferences.

TARIQ RAHIM SOOMRO (Senior Member, IEEE) received the B.Sc. (Hons.) and M.Sc. degrees in computer science from the University of Sindh, Jamshoro, Pakistan, and the Ph.D. degree in computer applications from Zhejiang University, Hangzhou, China. He is currently a Professor of computer science, the Dean of the College of Computer Science and Information Systems, and an Acting Rector with the Institute of Business Management (IoBM). He has more than 27 years of extensive and diverse experience as an administrator, a computer programmer, a researcher, and a teacher. As an administrator, he served as a coordinator, the head of the department, the head of the faculty, the dean of the faculty, the head of the academic affairs, an acting rector, and has wide experience in accreditation related matters, including ABET, HEC Pakistan, and the Ministry of Higher Education and Scientific Research, United Arab Emirates (UAE). He is also an IEEE Computer Society Distinguished Visitor (2021–2023). He is a member of the Task Force on Arabic Script IDNs by the Middle East Strategy Working Group (MESWG) of ICANN. He has been a member of the Project Management Institute (PMI), since 2007, and ACM, since 2019. He has been a Senior Member of the IEEE Computer Society and the IEEE Geosciences and RS Society, since 2005, and the International Association of the Computer Science and Information Technology (IACSIT), since 2012. He has been a Life Member of the Computer Society of Pakistan (CSP), since 1999, and a Global Member of the Internet Society (ISOC), USA, since 2006. He has been an Active Member of the IEEE Karachi Section (Region 10). He is also serving as the member of the Executive Committee (ExCom) (2017–2021), the IEEE R10 Southern Area Coordinator Computer Society (2020–2021), and the IEEE R10 Education Activity Committee (EAC) (2021–2022). He received the ISOC Fellowship to the IETF for the 68th Internet Engineering Task Force (IETF) Meeting. He served as the Secretary for the Karachi Section (2018–2019), the Chair for the GOLD Affinity Group, a member for the Executive Committee, in 2014, and a branch councilor (2002 & 2016). He is also serving as the Vice-Chair Karachi Section (2020–2021).