Technical Journal, University of Engineering and Technology (UET) Taxila, Pakistan Vol. 23 No. 2-2018
ISSN:1813-1786 (Print) 2313-7770 (Online)

A Review on Machine Learning Techniques for Software Defect Prediction

F. Hassan[1], S. Farhan[2], M. A. Fahiem[3], H. Tauseef[4]

[1,2,3,4]*Computer Science Department, Lahore College for Women University, Lahore*
[2]saifar79@hotmail.com

Abstract-Software defect prediction has been an interest of research era because predicting defects in early stages improves software quality with reduced cost and effective software management. Researchers from different domains are contributing their efforts to propose an approach that effectively and efficiently help in this regard. Different machine learning techniques have been applied to remove unnecessary and fault data from defect prone modules and many approaches, frameworks, methods and models have been proposed using different datasets, metrics, and evaluation strategies. In this paper, 40 Clarivate Analytics indexed impact factor journal papers from 2009-2018 are reviewed for the upcoming practitioners of software defect prediction. Review in this paper reflects some of the work that has been done in software defect prediction so far. Detailed classification taxonomy of the machine learning techniques used for software defect prediction has been presented. Defective, non-defective datasets along with the classification of the metrics used are part of the review. Despite of all works and efforts done in this research domain, there still exist many ambiguities because no single technique and method dominates due to the imbalance nature of different datasets and methods. A lot of research work is needed to overcome the existing issues.

Keywords-Classification, Machine Learning, Software Metrics, Software Defect Prediction

I. INTRODUCTION

Software Defect Prediction (SDP) has been seen as the most important research area since the beginning of software era. It plays an important role for enhancing the software quality. Testing is considered as the most important phase of software development life cycle (SDLC) and it is closely related with software quality. Software quality is improved when we have an early prediction of errors that are expected to occur in future. It is very suitable to detect the defects in early stages of SDLC to reduce the cost and to increase the effectiveness of the testing process. When defects are detected before the software release, they can be removed before the deployment of the software. Defect prone modules are required to be identified at earlier stages so that practitioners can have an idea that which module requires detailed testing. When software is released with errors, huge amount of time and cost is invested to make it error free. Therefore, it is preferable to detect them at an early stage. Cost of developing software is very high and when high amounts are invested to build software, a lot of factors are considered to develop it perfectly. The goals of SDP are (i) to predict defects in early stages, (ii) to identify the important modules that need more resources and attention, (iii) to improve the quality of software, (iv) to reduce the cost and (v) to provide effective software management. An overview of SDP process is shown is Fig 1. Here, the datasets mostly used are gathered from repositories available for SDP like Promise, UCI and any software releases. Method or code level metrics are extracted, e.g. Lines of Code (LOC), Branch count etc. Generally, a scheme is developed that has its basis in some of the existing techniques and approaches such as Naïve Bayes (NB), Association Rules, and Neural Networks (NN). The performance of the proposed approach is evaluated by the use of evaluation measures e.g. Area under curve (AUC), F-Measure etc.

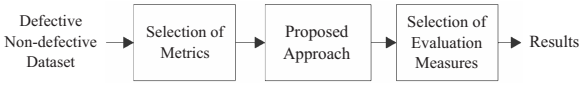


Fig. 1. Process of Software Defect Prediction

Usually software has faults/errors during development phases and efforts are needed to reduce these. Early prediction of defects is very necessary to reduce the cost. It helps the practitioners to devote extra resources and time to the non-defective software modules. In the era of Information Technology, software development is getting complex day by day and with increased complexity, the probability of faults are getting higher. Software is composed of many modules and during this process modules can be marked as likely to be defective or non-defective. When a module is predicted to be defected, more efforts and attention is paid to it. Many researchers from different domains are proposing variety of frameworks, models and techniques for the defect prediction. Researchers are also enhancing existing techniques and models for better prediction. Despite of all the efforts being performed, SDP research area still has many ambiguities. Although a number of schemes, models

and frameworks are proposed, no single technique or model dominates as each has its own limitations. Among all the domains, machine learning and statistics are considered to be the most important. Different machine learning algorithms and statistical techniques are applied to detect the defects and to remove unnecessary and faulty data in a specific module that is making the module more likely to be defective. Different datasets are available publicly to help the practitioners come up with improved and better results. Different attributes of these datasets are considered to be associated with defective or non-defective modules. Different performance evaluation measures help the researchers to check the performance of the applied technique, model or algorithm. All these are essence of this review.

SDP is a hot topic since many years and different techniques from various domains have been applied to predict the error prone modules [i]. Defects in software can be predicted by a number of machine learning algorithms. Different algorithms result in varying performance on different dataset. To decide, which algorithm and technique should be used for defect prediction is a difficult task. There exists no clear and straight forward answer to this question as same technique when applied on different dataset with different metric comes up in different results. Hence, it is stated by many researchers that current state of the art in SDP is still missing and no single technique is the king to dominate. Although, researchers are investigating different techniques to build a generic framework/model that can be used to detect the failures but the imbalance nature of datasets is the biggest hurdle in the way. Many researchers also came up with the solutions to deal with the imbalance nature but there are various factors that hinder the smooth performance of the specified solution. Best classifiers when run on different datasets can result in poor results. There exists many machine learning and statistical techniques in literature that have been used for SDP. Widely used techniques for SDP are presented in Fig 2.

Rest of the paper is organized as follows: Section II describes methodology of this research. Section III presents findings considering various aspects of SDP. Section IV presents a review of research articles used in this research. Section V is the comparative analysis of review articles already published in literature. Section VI is the summary of review and section VII provides conclusion of this study.

II. METHODOLOGY

In this paper, several strategies are applied to retrieve related publications. The search started with key terms 'software defect prediction' and 'machine learning'. The search was restricted for the past 10 years i.e. 2009-2018. Only Clarivate Analytics indexed journal articles are selected. The steps for search strategy that have been followed for the review can be seen in Fig 3.

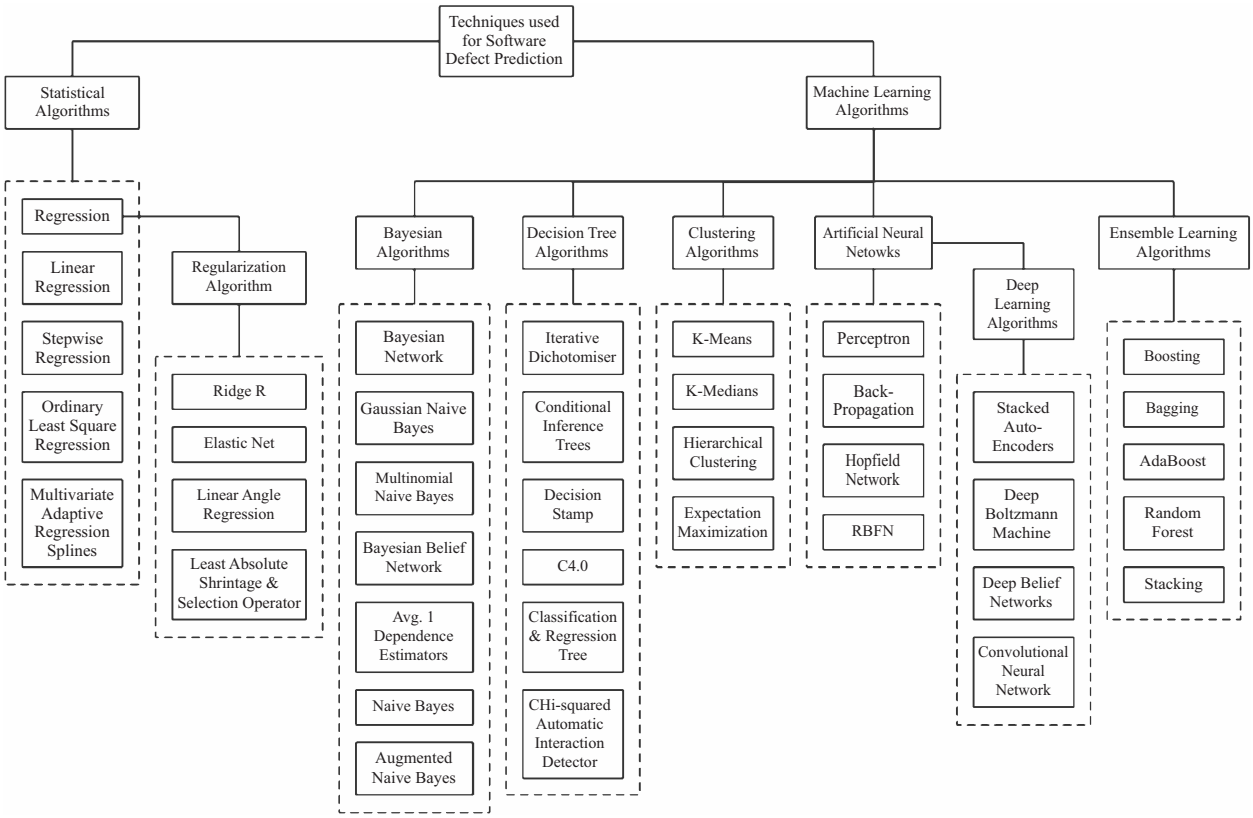


Fig. 2. Classification taxonomy of machine learning and statistical techniques used for Software Defect Prediction

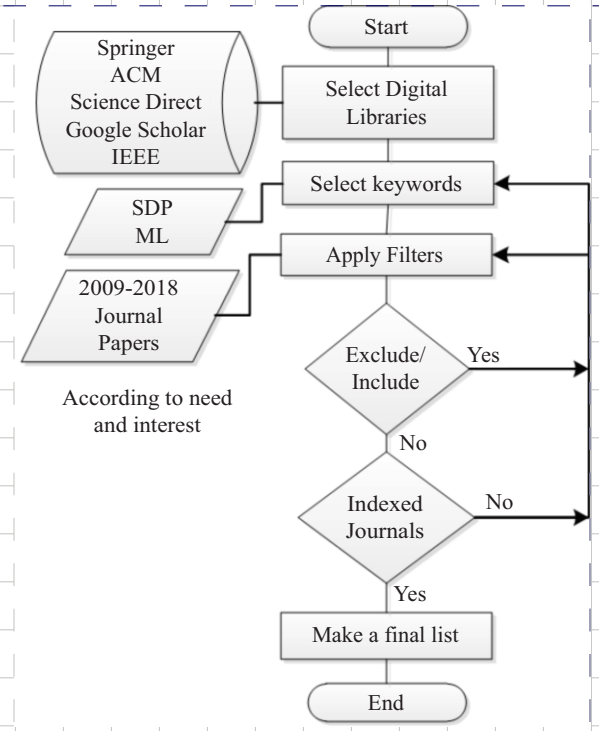


Fig. 3. Methodology used for Review

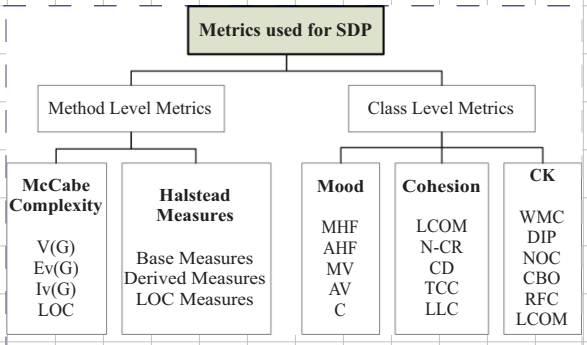
III. FINDINGS

This section presents our finding relating to SDP and machine learning. It comprises of three subsections i.e. software metrics, datasets and performance evaluation measures.

A. Software Metrics

A number of software metrics exist in literature for the prediction of software defects. Two classes of software metrics most widely used in SDP are method level and class level metrics as presented in Fig 4.

System level, package level, project level and design level metrics are also part of software metrics but they have not been used in SDP so far because of the nature of their attributes. To analyze a model, LOC and Function Point metric are also used. CK [ii] is the code level metrics. Percentages of metrics used in different studies are presented in Fig 5, as used in different journal papers.



V(G): Cyclometric complexity, Ev(G): Essential cyclometric complexity, Iv (G): Design complexity, LOC: Lines of Code, WMC: Weighted methods per Class, DIT: Depth of inheritance tree, NOC: Number of children, CBO: Coupling between objects, RFC: Response for a Class, LCOM: Lack of cohesion in methods

Fig. 4. Hierarchy of most widely used Metrics for Software Defect Prediction

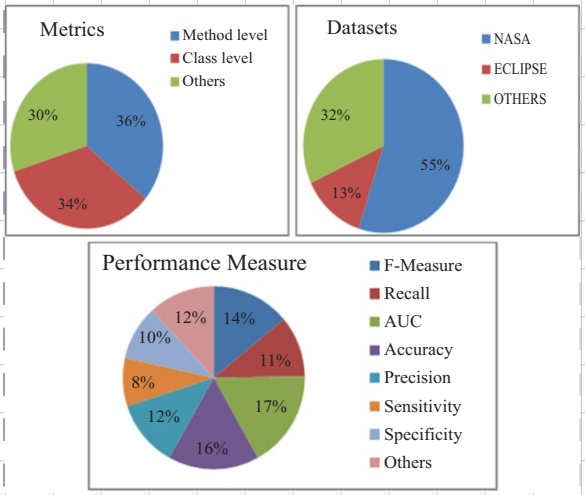


Fig. 5. The Percentage of various Metrics, Datasets and Performance Measures used in SDP studies

B. Datasets

Dataset is a collection of information that is used in the specific field for the problem under consideration. Many datasets are available publicly for the practitioners of SDP. The problem is the unavailability of standard datasets that can be used in SDP studies. Many practitioners proposed different frameworks by using different datasets and it is very hard to predict and access those frameworks because of the different nature of datasets used to develop it. Percentage of datasets most commonly used in different studies for SDP has been given in Fig. 5 and statistical analysis is based on the relevant SDP studies [iii-xxxii].

The problem of unavailability of standard datasets is present from ages and has been widely faced by the practitioners of machine learning. For this purpose, UCI repositories were developed to help the practitioners. The PROMISE repository is the inspired version of the UCI repository developed in 2005. NASA datasets with default ARFF file format are

widely used for SDP and are publicly available on NASA MDP repository. Some NASA datasets are also available on PROMISE repository. NASA datasets are the mostly widely used in SDP research [xxi]. Before 2005, most of the studies were carried out using limited sources of private datasets. From the foundation of PROMISE and other publicly available repositories, the rate of public datasets is getting higher. According to [xxv] only 35.21% of the studies used private datasets. Different defective and non-defective NASA datasets along with their properties are given in Table I which is obtained from Tuned IT platform.

C. Performance Evaluation Measures

When an approach is proposed, it needs to be evaluated to check its effectiveness and efficiency. Different researchers used different evaluation strategies to access the performance of their proposed approaches. Fig. 5 shows the percentage of evaluation measures in different studies. For reliable SDP, many performance evaluation metrics exists including accuracy, AUC, F-measure, Recall etc. that helps to check the efficiency of the proposed scheme.

Ranking based evaluation for the feature selection scenarios is another measure for evaluation. True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR) and Positive Predicted Value (PPV) are some common terms used in performance measures. Performance evaluation measures are given in Table II.

TABLE I
NASA DATASETS AND ITS CHARACTERISTICS

NASA Datasets	Description	Attributes	Defective	Non-Defective
KC1	Storage management	22	326 (15.4%)	1783 (84.5%)
KC2	Storage management	22	105 (20.1%)	415 (79.8%)
KC3	Storage management	40	36 (18.5%)	158 (81.4%)
JM1	Real time predictive ground system	22	2106 (19.3%)	8779 (80.6%)
CM1	NASA space craft instrument	22	49 (9.83%)	449 (90.1%)
AR1	Embedded software	30	9 (7.43%)	112 (92.5%)
MW1	A zero gravity experiment	38	27 (10.5%)	228 (89.4%)
PC1	Flight software	22	77 (6.94%)	1032 (93.05 %)

IV. PAPERS IN REVIEW

40 impact factor journal papers on SDP published from 2009 - 2018 have been selected in this study. This review paper indicates the relevant work done in the specified duration by the practitioners and researchers. It is different from other reviews in the following aspects.

- Detailed classification of machine learning techniques and metrics that are used in SDP have been given
- Only Clarivate Analytics indexed Journal papers in the duration 2014 - 2016 have been reviewed (before this 1990 - 2013 papers were reviewed)
- Articles belonging to deep learning are also the part of the review

The percentage of distribution of our selected journals according to the year published can be seen in Fig. 6.

The effects of dataset size and selected metrics on SDP are studied by [xxxiii]. Five public NASA datasets have been used. After examining 9 classifiers, Random Forest (RF) came up with the highest prediction performance for larger datasets and NB came out to be the best one for smaller datasets. To cater class imbalance problem, SMOTE and Resample with substitution techniques are used with Fisher linear Discriminant Analysis (FLDA) for attribute selection [xxxiv]. Artificial neural network (ANN), support vector machine (SVM), NB and RF turned out to be performing better with Resample-FLDA.

Action based defect prediction (ABDP) includes fault patterns and corresponding pattern of actions to find out the causes of faults before their occurrence[iv]. This approach is based on association rules and decision trees. A modeling techniques, in terms of cost and fault proneness, for java systems using ensemble learning methods is proposed by [vii] . The quality of the effective model is highly dependent on the criteria used to build it. When ensemble learner, AdaBoost, is combined with decision tree algorithm, C4.5, it produced better results. A framework for SDP is proposed by [xxxv] that consisted of two parts i.e. evaluation scheme and defect prediction. Twelve learning schemes have been compared for evaluation. NASA and AR datasets are used with Halstead attributes as well as McCabe complexity measures. ROC measures are used for evaluation. Cohesion, complexity and coupling (CCC) are considered to be the best metrics for SDP[ix]. Before this research, there was no framework to utilize CCC by considering security aspects to predict the failures automatically for software development. Fifty two releases of Mozilla fire fox are used as a dataset. Ensemble learning algorithms for the improved fault prediction by considering metrics as an important parameter in SDP are presented in [xi]. Seventeen ensemble methods have been used to combine all metrics instead of using single one. Filter based Ranking technique is used to evaluate and predict the model performance by selecting best attributes. A systematic review in the duration 2000 - 2010 on 208 studies on SDP is conducted by [xxxvi]. Thirty six of these studies are extracted on the common and important factors for evaluation. Independent variables that should be

considered in SDP models to detect the defects are also focused. NB and Logistic Regression (LR) produced best results and are concluded to be used to build a model. Based on another perspective, it is pointed that successful model should be built on larger datasets. In addition to supervised learning, clustering techniques e.g. K-means, are also used for defect prediction resulting in maximum gain values [xxxvii]. NASA datasets are used by [xxxviii] focusing on the issues present in it. The importance of these datasets is highlighted and it is added to preprocess the datasets before using it. The research is based on unpredictable experimental results achieved from NASA MDP. The use of NN for software defect prediction is encouraging [xxxix, xl] when non-linear and complicated relationship exists between software metrics. Fifteen Bayesian Network (BN) classifiers are used and a comparative analysis with other machine learning techniques is conducted to prove the efficiency of BN among all [xiv]. Instead of using a complex BN structure to enhance SDP other BN can be used effectively with simpler network.

TABLE II
PERFORMANCE EVALUATION MEASURES FOR SDP

Measure	Mathematical Model
AUC	$A = \int_{-\infty}^{\infty} TPR(T)FPR(T)dT$
Fall Out or FPR	$\frac{FP}{TN + FP}$
Recall or Sensitivity or TPR	$\frac{TP}{TP + FN}$
Precision or PPV	$\frac{TP}{TP + FP}$
Specificity or TNR	$\frac{TN}{TN + FP}$
F-score or F-Measure	$\frac{2(Recall * Precision)}{Recall + Precision}$
Accuracy	$\frac{TP + TN}{TP + FP + TN + FN}$

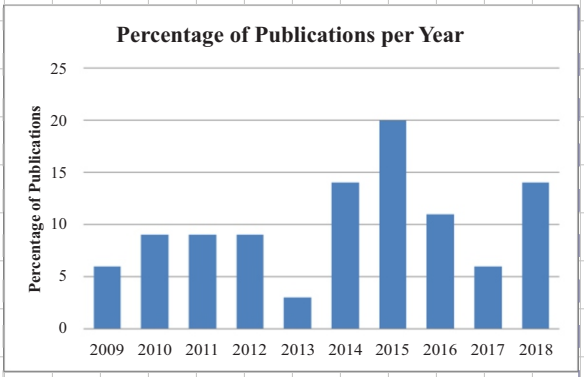


Fig. 6. Percentage Distribution of Selected Studies over the Years

The most important factors that influence and enhance SDP are classifiers, datasets, metrics and performance evaluation parameters [xvi]. Researcher group is another important factor to be considered as it has the biggest impact on SDP. The researcher's biasness is checked by performing a co-author analysis. A novel classification model, defect prediction using relational association rules (DPRAR), is proposed that predicted the existing correlations between different attributes of NASA dataset [xviii]. An algorithm, average probability ensemble (APE), combines feature selection with ensemble learning to overcome the redundant features and the problem of imbalance data [xx]. When classifying defective modules, the selection of features is done very carefully for effective defect classification. A systematic review based on 64 primary studies published during 1991- 2013 is conducted by [xxi]. The focused feature in this review is the performance capability of the machine learning techniques in different contexts. Three soft computing methods i.e. artificial neuro fuzzy interface system (ANFIS), SVM and ANN are compared in [xxvi]. Expert knowledge has been combined with the learning ability in ANFIS which differentiate it from other soft computing methods. The reduced set of parameters has a huge impact on the complexity and results. An approach for SDP with most commonly used classifiers i.e. Lazy K-Star, NB, RF, J48 and NASA datasets using TP, FP, precision, recall and accuracy as performance evaluation parameters is stated in [xxii]. NB is stated best for smaller datasets and RF is stated best for larger datasets. An empirical study is conducted and a simplified metric set is proposed by[xxiii]. The models, frameworks and best predicted classifiers, all failed when seen in the perspective of cross project defect prediction. Experiments have been performed with static code metrics on 10 releases of open source projects. The four classification methods i.e. NB, DT, RF and LR have been widely used in the studies until now [xxv]. A three way decision (3WD) based approach for SDP considering cost on the basis of two-stage classification and ranking approach is proposed in [xxviii]. Ensemble learning methods and two stage classification methods are combined to come up with a three way decision based approach on NASA datasets to lower the decision cost. An empirical framework for SDP using ensemble learning methods, statistical and machine learning techniques with the use of android software as a dataset is proposed by [xxix]. Statistical and post-hoc analysis have been used for the significant performance evaluation with object oriented metrics.

The latest trend in software defect prediction is the use of ANN [xli, xlii] and especially deep learning techniques[xliii, xliv]. Deep neural network (DNN) not only deepens the layers but also extracts suitable features for prediction. DNN is combined with genetic algorithm (GA) for feature optimization over

PROMISE dataset [xlv]. Another type of DNN i.e. recurrent NN (RNN) is used by [xlvi] and the results are compared with four NN models and five parameter models. Traditional metrics focus on designing features and are unable to capture the semantics of source code. Convolutional neural networks (CNN) with control flow graphs extracted from assembly instructions are designed to learn software's semantic features [xlvii]. Similarly, Long Short Term Memory (LSTM) network is used for SDP as it matches with abstract syntax tree of the code [xlviii]. In all these studies using DNN, the results produced are better than traditional learning models. Table III presents the techniques that practitioners should consider in SDP.

TABLE III
TECHNIQUES TO CONSIDER IN SDP

Review show that this research area can be improved by using	
• Machine Learning Algorithms	• Association rules
• Frameworks, Models	• ANN structures
• Statistical techniques	• Preprocessing of Dataset
• Classification	• Comparative analysis
• Clustering	• Ensemble approaches

V. COMPARATIVE ANALYSIS OF REVIEW ARTICLES

Reviews conducted earlier in the domain of SDP and machine learning are presented in this section to elaborate the difference with our review.

A review on SDP, considering studies published during 1990-2009, mainly used classification trees with only method level metrics [xlix]. Modeling techniques in terms of cost and fault proneness for java systems are reviewed in [vii]. Our review covers machine learning algorithms and is not limited to java systems. A systematic review for articles published during 2000-2010 is conducted by [xxxvi]. Out of 208 studies, 36 are selected based on the common and important factors for evaluation of software defects.

The review conducted by [xxi] considered 19 studies till 2013 focusing on comparison of LR models and machine learning algorithms. The review is limited to 7 selected machine learning techniques; DT, BL, EL, NN, SVM, RBL and EA. The most commonly used metric that he found from his studies was Correlation based feature selection.

Papers published from 2010 to 2013 on SDP are reviewed focusing on clustering and estimation methods using few private datasets along with Public datasets [xxv]. An analysis on SDP, reviewing assumptions that are based on NB algorithm are presented in [iii]. Meta Analysis technique for SDP is used for review by [xvi].

Keeping all earlier reviews in mind, our review is different from others according to the following aspects

- A detailed classification of machine learning techniques for SDP is presented. A new practitioner can utilize the classification taxonomy and can have a clear picture of the machine learning hierarchy.
- Only Clarivate Analytics indexed impact factor journal papers published in duration 2009-2018 are included. The earlier reviews cover publications till 2013.
- A review of metrics, datasets, and performance evaluation parameters used in SDP are presented.
- The role of deep learning algorithms in SDP is focused. There is no such prior review to the best of our knowledge that discusses the impact of deep learning algorithms in SDP.

VI. SUMMARY OF THE REVIEW

SDP studies based on different techniques, approaches, models, frameworks and methods itself are a proof that the current state of art in this research area is still missing. No matter how deep practitioner's research efforts are in SDP, the imbalance and diverse nature of different factors will always be the reason of the missing hole that exists in this field. Some researchers considered dataset to be the most important parameter when predicting faults in software and they contributed to make the dataset redundant, efficient, and concise for efficient SDP. There exists multiple defective and non-defective public and private datasets for practitioners. The unbalanced proportion of defective and non-defective datasets is the problem to consider. If number of defective modules is very higher than the non-defective modules, then no matter how best machine learning classifiers are used, overall poor results will be achieved. So, researchers considered dataset as the very important feature of any SDP approach.

Some researchers stated that no matter which dataset is used, the classifier used for prediction is the most important. Different researchers used different classifiers belonging to statistics and machine learning domain with different datasets. Different classifiers on different evaluation and experimental strategies against various large and small datasets showed different results. Other researchers from the research community of SDP predicted that no matter which classifier is used against which dataset; the performance evaluation strategy must be strong enough to predict accurate results. The perspective of few researchers is that input metric is the important feature as evaluation and experimental strategies will be performed using it. A number of machine learning rules are also applied in the SDP research and the results with various classifiers and rules are compared to check the validity of the proposed scheme and approach. Cross-project defect prediction is the challenge for researchers and some researchers are making their efforts in it.

Based on our primary study of 40 most relevant

journals, it can be seen that NASA datasets are the most widely used by the practitioners of SDP. So, journal papers which include NASA datasets have been considered to summarize the results presented in Table IV. Though we tried to find out the most relevant SDP impact factor journal papers to help the upcoming practitioners, there still exists chance that a more suitable and relevant study is missed out.

VII. CONCLUSION

Based on our primary study of 40 most relevant Clarivate Analytics indexed Journal papers, NASA datasets have been found as the most widely used datasets. In the perspective of software metric, McCabe, a procedural metric, is found as the most widely used metric. To evaluate the performance of the proposed approach, AUC, F-measure and accuracy are the most commonly used evaluators. Machine learning algorithms, ensemble approaches, feature selection are the trending techniques to enhance the SDP process. It is clear from the review that a lot of effort is being made by researchers in this emerging research topic. The importance of SDP has motivated many researchers to come up with different and better approaches. Despite of all the work done for predicting software defects, no generic approach is available due to certain issues; two of the most important are cross defect projection and imbalance nature of datasets. Different techniques and methods used in SDP can be seen in literature but despite of their use against different metrics and dataset, it is hard to decide which technique is better. This research topic still needs a lot of attention to find out the missing state of the art. Software development and usage are amazingly increasing and complexity with the increased number of software usage is getting even higher. That complexity is seen in terms of defects in software, which need to be predicted earlier. Despite of all the work and efforts done in this research topic, there still exists many ambiguities and a lot of research work is needed to overcome the existing issues. More number of studies needs to be carried out in future to help the upcoming practitioners.

TABLE IV
SUMMARIZED RESULTS OF STUDIES THAT USED NASA DATASETS

Reference	Proposed Approach	Classifier	Metrics	Datasets	Results (%)	
[vi]	Multi-objective particle swarm optimization **(MOPSO)**	NB, NN, SVM, C4.5, BN, RIPPER	Class Level Method level	Pc1, CM1, JM1, KC2, KC1, KC1-CL	AUC Accuracy	78.72 82.02
[viii]	Ensemble learning with Analytic Hierarchy Process	**Trees SimpleCart Adaboost**	-	CM1, JM1, KC3, KC4, MC1, MW1, PC1-4	AUC F-Measure Precision Accuracy Recall	89.98 90.88 91.95 92.63 90.32
[xii]	Evolutionary Decision Rules for Subgroup Discovery **(EDER**-SD)	SD, Apriori CN2-SD	Method Level (McCabe, Halstead)	CM1, KC1, KC2, KC3, MC2, MW1, Pc1	Accuracy Specificity Precision	87.94 97.68 56.08
[xiii]	Ensemble learning for SDP (1-all, **1-1**, RCC)	**RF,** C4.5, Ripper, NB	Method Level (McCabe)	CM1, JM1, KC1, KC2, KC3, KC4, MC1, MC2, MW1, PC1, PC2, PC3, PC4, Pc5	AUC	84.35
[xix]	Analysis of linear regression and machine learning	**DT**, SVM, GEP, CCN, ANN, MLR	Object Oriented	AR1, AR6	AUC Specificity Sensitivity	90.65 79.55 91.3
[xviii]	**DPRAR**	CBA2, Bagging, 1R, EDER-SD	Object Oriented (CK)	CM1, KC1, KC3, PC1, JM1, MC2, MW1, PC2, PC3, Pc4	AUC Accuracy Specificity Precision Pd	89.75 91.89 93.21 72.45 86.27
[xxii]	Analytical Approach for SDP	NB, J48, **RF,** K-Star	Object Oriented	CM1, KC2, PC1, KC1	Accuracy F-Measure Recall Precision	88 86.17 85.27 85.45
[xx]	APE and **Enhanced APE**	W-SVM, RF	Method Level (McCabe)	KC3, MC1, PC2, PC4	AUC G-mean	93.75 90.75
[xxvi]	**ANFIS** for SDP	SVM, ANN	Method Level (McCabe)	CM1, JM1, KC1, KC2, PC1	AUC	85.73
[xxviii]	3W decision based SDP **(3WD)**	2WD	Fault Percentage Average	CM1, JM1, KC2, KC3, MC2, MW1, PC1, PC2, PC3, PC4, PC5	Accuracy F-Measure	80.20 87.55

REFERENCES

[i] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert systems with applications,* vol. 36, pp. 7346-7354, 2009.

[ii] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on software engineering,* vol. 20, pp. 476-493, 1994.

[iii] B. Turhan and A. Bener, "Analysis of Naive Bayes' assumptions on software fault data: An empirical study," *Data & Knowledge Engineering,* vol. 68, pp. 278-290, 2009.

[iv] C.-P. Chang, C.-P. Chu, and Y.-F. Yeh, "Integrating in-process software defect prediction with association mining to discover defect pattern," *Information and Software Technology,* vol. 51, pp. 375-384, 2009.

[v] Y. Zhou, B. Xu, and H. Leung, "On the ability of complexity metrics to predict fault-prone classes in object-oriented systems," *Journal of Systems and Software,* vol. 83, pp. 660-674, 2010.

[vi] A. B. De Carvalho, A. Pozo, and S. R. Vergilio, "A symbolic fault-prediction model based on multiobjective particle swarm optimization," *Journal of Systems and Software,* vol. 83, pp. 868-882, 2010.

[vii] E. Arisholm, L. C. Briand, and E. B. Johannessen, "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models," *Journal of Systems and Software,* vol. 83, pp. 2-17, 2010.

[viii] Y. Peng, G. Kou, G. Wang, W. Wu, and Y. Shi, "Ensemble of software defect predictors: an AHP-based evaluation method," *International Journal of Information Technology & Decision Making,* vol. 10, pp. 187-206, 2011.

[ix] I. Chowdhury and M. Zulkernine, "Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities," *Journal of Systems Architecture,* vol. 57, pp. 294-313, 2011.

[x] I. Alsmadi and H. Najadat, "Evaluating the change of software fault behavior with dataset attributes based on categorical correlation," *Advances in Engineering Software,* vol. 42, pp. 535-546, 2011.

[xi] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, "Software measurement data reduction using ensemble techniques," *Neurocomputing,* vol. 92, pp. 124-132, 2012.

[xii] D. Rodríguez, R. Ruiz, J. C. Riquelme, and J. S. Aguilar–Ruiz, "Searching for rules to detect defective modules: A subgroup discovery approach," *Information Sciences,* vol. 191, pp. 14-30, 2012.

[xiii] Z. Sun, Q. Song, and X. Zhu, "Using coding-based ensemble learning to improve software defect prediction," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 42, pp. 1806-1817, 2012.

[xiv] K. Dejaeger, T. Verbraken, and B. Baesens, "Toward comprehensible software fault prediction models using bayesian network classifiers," *IEEE Transactions on Software Engineering,* vol. 39, pp. 237-257, 2013.

[xv] S. Agarwal and D. Tomar, "A feature selection based model for software defect prediction," *assessment,* vol. 65, 2014.

[xvi] M. Shepperd, D. Bowes, and T. Hall, "Researcher bias: The use of machine learning in software defect prediction," *IEEE Transactions on Software Engineering,* vol. 40, pp. 603-616, 2014.

[xvii] M. Liu, L. Miao, and D. Zhang, "Two-stage cost-sensitive learning for software defect prediction," *IEEE Transactions on Reliability,* vol. 63, pp. 676-686, 2014.

[xviii] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Information Sciences,* vol. 264, pp. 260-278, 2014.

[xix] R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules," *Applied Soft Computing,* vol. 21, pp. 286-297, 2014.

[xx] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology,* vol. 58, pp. 388-402, 2015.

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

[xxii] R. Sathyaraj and S. Prabu, "An approach for software fault prediction to measure the quality of different prediction methodologies using software metrics," *Indian Journal of Science and Technology,* vol. 8, 2015.

[xxiii] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," *Information and Software Technology,* vol. 59, pp. 170-190, 2015.

[xxiv] S. Aleem, L. F. Capretz, and F. Ahmed, "Benchmarking Machine Learning Technologies for Software Defect Detection," *arXiv preprint arXiv:1506.07563,* 2015.

[xxv] R. S. Wahono, "A systematic literature review of software defect prediction: Research trends, datasets, methods and frameworks," *Journal of Software Engineering,* vol. 1, pp. 1-16, 2015.

[xxvi] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," *Expert Systems with Applications,* vol. 42, pp. 1872-1879, 2015.

[xxvii] S. S. Rathore and S. Kumar, "Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems," *Knowledge-Based Systems,* 2016.

[xxviii] W. Li, Z. Huang, and Q. Li, "Three-way decisions based software defect prediction," *Knowledge-Based Systems,* vol. 91, pp. 263-274, 2016.

[xxix] R. Malhotra, "An empirical framework for defect prediction using machine learning techniques with Android software," *Applied Soft Computing,* vol. 49, pp. 1034-1050, 2016.

[xxx] D. Ryu and J. Baik, "Effective multi-objective naïve bayes learning for cross-project defect prediction," *Applied Soft Computing,* vol. 49, pp. 1062-1077, 2016.

[xxxi] M. Hamill and K. Goseva-Popstojanova, "Analyzing and predicting effort associated with finding & fixing software faults," *Information and Software Technology,* 2017.

[xxxii] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: do different classifiers find the same defects?," *Software Quality Journal,* pp. 1-28, 2017.

[xxxiii] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Information Sciences,* vol. 179, pp. 1040-1058, 2009.

[xxxiv] A. Kalsoom, M. Maqsood, M. A. Ghazanfar, F. Aadil, and S. Rho, "A dimensionality reduction-based efficient software fault prediction using Fisher linear discriminant analysis (FLDA)," *The Journal of Supercomputing,* pp. 1-35, 2018.

[xxxv] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," *IEEE Transactions on Software Engineering,* vol. 37, pp. 356-370, 2011.

[xxxvi] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering,* vol. 38, pp. 1276-1304, 2012.

[xxxvii] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," *IEEE Transactions on knowledge and data engineering,* vol. 24, pp. 1146-1150, 2012.

[xxxviii] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Reflections on the NASA MDP data sets," *IET software,* vol. 6, pp. 549-558, 2012.

[xxxix] J. Zheng, "Cost-sensitive boosting neural networks for software defect prediction," *Expert Systems with Applications,* vol. 37, pp. 4537-4543, 2010.

[xl] M. Benaddy and M. Wakrim, "Simulated annealing neural network for software failure prediction," *International Journal of Software Engineering and Its Applications,* vol. 6, pp. 35-46, 2012.

[xli] D.-L. Miholca, G. Czibula, and I. G. Czibula, "A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks," *Information Sciences,* vol. 441, pp. 152-170, 2018.

[xlii] R. Jayanthi and L. Florence, "Software defect prediction techniques using metrics based on neural network classifier," *Cluster Computing,* pp. 1-12, 2018.

[xliii] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks,* vol. 61, pp. 85-117, 2015.

[xliv] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature,* vol. 521, p. 436, 2015.

[xlv] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Computing,* pp. 1-17, 2018.

[xlvi] J. Wang and C. Zhang, "Software reliability prediction using a deep learning model based on the RNN encoder–decoder," *Reliability Engineering & System Safety,* vol. 170, pp. 73-82, 2018.

[xlvii] A. V. Phan, M. L. Nguyen, and L. T. Bui, "Convolutional Neural Networks over Control Flow Graphs for Software Defect Prediction," *arXiv preprint arXiv:1802.04986,* 2018.

[xlviii] H. K. Dam, T. Pham, S. W. Ng, T. Tran, J. Grundy, A. Ghose*, et al.*, "A deep tree-based model for software defect prediction," *arXiv preprint arXiv:1802.00921,* 2018.

[xlix] C. Catal, "Software fault prediction: A literature review and current trends," *Expert systems with applications,* vol. 38, pp. 4626-4636, 2011.