Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

PREDICTION OF SOFTWARE DEFECTS USING ENSEMBLE LEARNING: REVIEW

TARIQ N. AL-HADIDI
Software Engineering Department, Collage of Computer Science and Mathematics, University of Mosul, Mosul, Iraq. Email: tariq2022hadidi@gmail.com

SAFWAN O. HASOON
Software Engineering Department, Collage of Computer Science and Mathematics, University of Mosul, Mosul, Iraq. Email: Dr.safwan1971@uomosul.edu.iq

Abstract

Compilation calculations are now frequently used in various software quality assurance processes. These classifiers showed a better implementation of their component models. Despite the fact that ensemble approaches are regularly used in urgent areas such as Programming Distortion Predictions (SDP) and Programming Change Predictions (SCP), the most recent exploration into their application requires careful evaluation. The objective of the review is to assess, scan, and mark any neglected exploration needs associated with the use of Group Policy in SDP and SCP. This study includes a comprehensive evaluation of the research in light of the classification, application, and rules of definition, implementation, and potential threats to the progress that was used. Important criteria used to rank grouped strategies are similarity, grouping, connectivity, diversity, and reliance on key ideal models. It's also proven useful for a wide number of uses, including learning computation for making SDP/SCP models and taking care of class imbalance. The survey findings arguably support the need for further examination to propose, evaluate, approve and look at different classes of diverse innovations for a variety of SDP/SCP applications, including web-based education.

Keywords: Ensemble learning, Software defect prediction, Software quality, Machine Learning

1. INTRODUCTION

By optimizing different ranking or characterization models using various AI procedures, software defect prediction (SDP) is a way to improve programming quality and reduce programming testing costs. Many organizations that manufacture different types of programming try to anticipate problems to protect programming quality to keep the customer happy and reduce testing costs. As part of the product improvement life cycle, SDP uses machine learning (ML) to make predictions of software failures using verifiable information [1]. An organized strategy enables the work to provide a high degree of accuracy and reliability in the software while consuming a short programming period to meet the customer's requests [2].

To improve forecast execution, the ensemble learning model is built by consolidating many AI classifiers [3]. Various names, including cross breed, consolidated, coordinated, and collected classification, are utilized in the writing to depict group picking up, as per [4]. A solitary classifier, for example, naïve Bayes classifier, Decision trees, or a multilayer perceptron, is utilized to foster the expectation model on a pre-marked dataset in the customary procedure of deformity forecast [4].

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

Individual classifiers will be unable to precisely conjecture a specific blemish in a given circumstance [5]. Along these lines, ensemble learning was utilized to join the qualities of different classifiers to further develop shortcoming revelation in the dataset. In the beyond a decade, an enormous number of scholastics have offered exact information that proposes troupe approaches offer more prominent order precision than individual classifiers [6].

In view of the sorts of base students, outfit techniques might be comprehensively partitioned into two gatherings:

- Techniques for homogeneous gatherings.
- Procedures for heterogeneous gatherings.

Similar fundamental students are applied to a few arrangements of cases in a dataset utilizing homogeneous gathering techniques. Models incorporate supporting, turn of the bagging, boosting, rotation forest [7].

Different base learners are configured using different AI procedures in the heterogeneous clustering technique. These basic students join, and the extreme prediction is completed by statistically integrating the results of the basic learners [7]. Due to the many attributes of basic learners, heterogeneous techniques are optimized more than homogeneous strategies.

Straight and nonlinear troupe techniques fall inside these classes also. While the result of base student models is consolidated utilizing a direct capability, for example, a weighted normal or straightforward normal, in direct gathering strategies, the choice of base students is joined utilizing a nonlinear strategy, for example, decision tree or support vector machine (SVM) [8]. While consolidating various classifiers into an outfit, scientists likewise think about assortment.

Variety of classifiers is the possibility that unmistakable occurrences of information are misclassified by the troupe strategy's chosen classifiers. The variety between two classifiers is surveyed utilizing different measurements, including the Connection Variety Measure, Q-Insights, Accuracy, and Weighted Accuracy and Diversity (Roll). Alongside enhancements in troupe learning techniques, other fascinating SDP approaches are likewise being put out. These techniques utilize the thoughts of "code scents" and "prerequisites smells" to anticipate blunders far prior in the product improvement lifecycle [8].

2. BACKGROUND

2.1 Software defect prediction [8]

In the realm of computer programming, programming imperfection expectation has gotten a great deal of interest. Furthermore, it can raise programming item quality [9], which can bring down improvement expenses and lift advancement viability. Using measurements that are connected with issues in the program, it makes a deformity expectation model

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

and afterward gauges how the product will track down such imperfections. Three related areas of exploration are as per the following:

1. Examine information related to software defects. Unshakeable information quality issues can arise in light of the fact that there are many components that affect information for software delivery. Understanding the information and performing basic pre-processing tasks, such as normalization and standardization, are expected before building the model to further enhance the model at a later time.
2. Production of a model for anticipating programming surrenders. Models in view of supervised learning, semi- Supervised learning, and Unsupervised learning are remembered to fall into three classifications that can each be utilized to expect programming absconds. Whether the product deformity information is adequate is where they separate. To come by improved results, we should pick the right model in light of the amount of imperfection informational indexes currently accessible.
3. The model must be assessed utilizing a couple of pointers after it had been inherent request to know whether it was proceeding true to form. AUC, Accuracy, Precision, ROC bend, and F1-measure are all assessment lists. To begin with, the disarray framework definition is given, as in Figure (1) and Table (1):

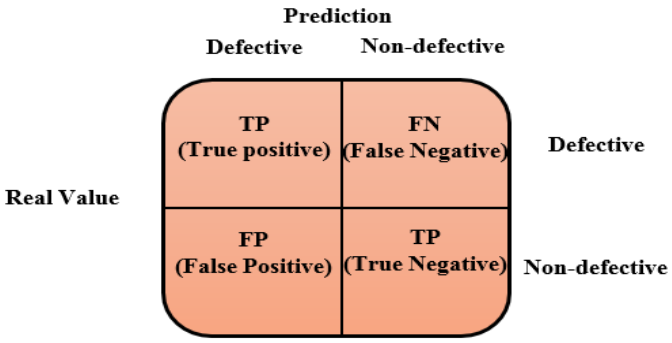

Figure 1: Confusion matrix

- **TP (True Positive)** - The data of interest in the disarray framework is the genuine positive point (TP) when a positive result is normal and exactly the same thing occurs.
- **FP (False positive)** - The data of interest in the disarray lattice is a bogus positive when a positive outcome was normal, and what happened is an adverse outcome. This situation is known as a sort 1 mistake. It resembles the gift of awful premonition.
- **FN (False Negative)** - The data of interest in the disarray lattice is bogus negative when an adverse result was normal, and what happened is a positive result. This situation is notable as a kind 2 blunder and is considered as perilous as a sort 1 mistake.
- **TN (True Negative)** - The data of interest in the disarray framework is Valid Negative (TN) when an adverse result is normal and the equivalent occurs.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

Table 1: The elements of the evaluation process (variables, definitions, and equations) [10]

Variable	Definition	Equation
Accuracy	The percentage of accurately anticipated data from tests is easily determined by dividing all accurate forecasts by all predictions.	$Accuracy = \frac{Tp+Tn}{TP+TN+FP+FN}$
Precision	He proportion of outstanding instances among all anticipated ones from a specific class	$Precision = \frac{TP}{TP+FP}$
Recall	The ratio of the total number of occurrences to the proportion of instances that were supposed to be members of a class	$Recall = \frac{TP}{TP+FN}$
F1-Score	The phrase is used to describe a test's accuracy. The maximum F1-score is 1, which denotes outstanding recall and precision, while the lowest F1-score is 0.	$F1 - Score = 2 \times \frac{percision \times recall}{Percison+recall}$

Markers like MCC, G-mean, and others are utilized to survey the product deformity expectation model, but the previously mentioned four are more delegate.

2.2 Ensemble Learning Algorithm and its Application

A machine learning approach known as ensemble learning is applicable to both supervised and unsupervised learning. Different students cooperate in a gathering to settle a particular problem [11]. The presentation of the general learning model is then improved by consolidating the results of gathering figuring out how to compensate for the error [12]. Supporting, packing, the Random Subspace Method (RSM), stacking, and group techniques in view of casting a ballot are a portion of the routinely utilized outfit strategies.

As well as delivering a consolidated model with further developed execution by blending numerous straightforward models, outfit advancing as a sort of learning approach for joined enhancement likewise empowers specialists to make consolidated models for specific AI moves to create more compelling arrangements. From a numerical stance, Dietterich distinguishes three outfit factors for the viability of troupe learning: measurements, calculation, and portrayal. Disintegration of deviation difference can likewise be utilized to assess the progress of gathering learning [13].

Calculations for ensemble learning are utilized for some aspects of genuine issues. Outfit learning procedures were used by Ruszczak in 2020 to track down tomato Alternaria diseases. Selim Buyrukolu will use troupe figuring out how to recognize Alzheimer's illness in 2021, and Leiyu Dai will research how to characterize landslip risk utilizing group learning. Also, group learning is regularly used in quality information examination, network interruption discovery, drug movement location, talk robot information securing, data recovery positioning learning, and programmed open air route for robots [9].

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

2.3 The most commonly employed tools to implement ensemble learning techniques

To do prescient investigation on information utilizing different AI classifiers, various information mining instruments have been created. These strategies can discover a critical example in information to uncover unseen information [14][15].

Each device has a specific arrangement of capacities, consequently scientists pick apparatuses as per the AI methods they have picked. The dissemination of essential exploration among AI libraries and advances is portrayed in Fig. 5. We found that the most famous AI device for investigating, picturing, and performing arrangement over information in our picked essential examinations is the Waikato Climate for Information Investigation (WEKA) [16] sklearn [17], a Python AI bundle, and LIBSVM [17] are further devices.

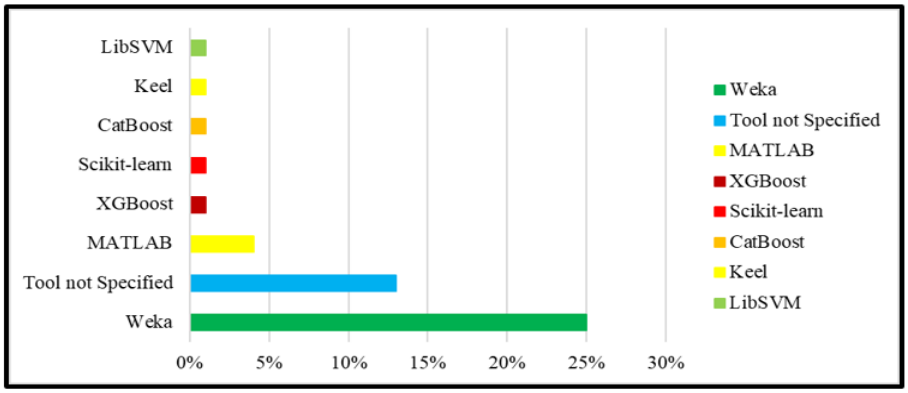

Figure 2: Dispersion of essential exploration across libraries and AI devices.

All techniques that were evaluated were executed in Python in [18]. In any case, the paper made no notice of the AI strategy portrays how arrangement on fall datasets was completed utilizing the fall program. To make tree-based troupes, analysts utilized the scikit-learn, CatBoost, and XGBoost bundles in [19].

3. METHODOLOGY OF SOFTWARE DEFECT PREDICTION [1]

Expectation models made for DP are generally used to conjecture programming blemishes. Albeit a wide range of techniques and calculations have been utilized to deliver better performing (i.e., more precise) SDP models, Figure 1 sums up the fundamental SDP steps: The means are as per the following: (1) accumulate spotless and defective code tests from programming stores; (2) extricate elements to make a dataset; (3) balance the dataset on the off chance that it is uneven; (4) train an expectation model on the dataset; (5) gauge the imperfect parts for a dataset separated from another product (different variant of prepared dataset or new programming task); and (6) assess the exhibition of the SDP model. As a result of its straightforwardness [20], Figure (3) overlooks the emphasis steps that make up this interaction.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

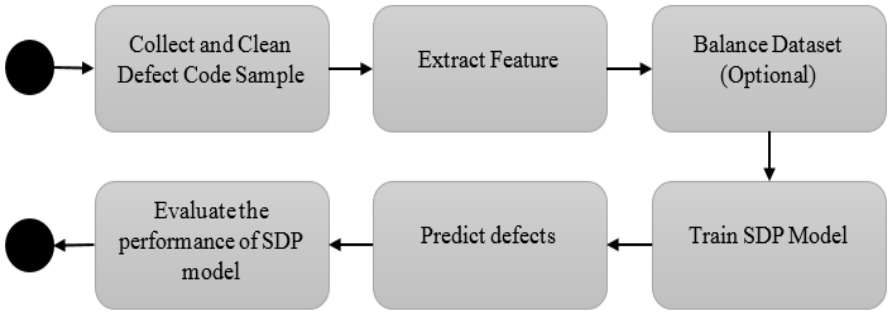

Figure 3. Software defect prediction process

The technique starts with social occasion tests of both spotless and defective code, as found in Figure 3. There are various organizations in which programming information is accessible, including source codes, commit messages, bug reports, and other programming ancient rarities. Ordinarily, this information is removed from files and stores [20].

The element extraction period of SDP is the following stage. Programming antiques, source codes, commit logs, and messages, among others, are changed into measurements at this stage and utilized as info information for preparing models. The element extraction stage relies intensely upon the kind of information, including McCabe measures [21], CK measurements [22], change narratives, get together code, and source code. Various DL calculations today offer programmed include extraction from additional confounded, high layered information notwithstanding metric-based information. Imperfection data from notable public deformity stores, similar to the NASA [23] and Commitment [24] data sets, has been utilized in different exploration in the writing.

Normally, the subsequent stage is discretionary. Since imperfection datasets frequently incorporate much less broken parts than non-defective, this stage involves adjusting the information. Sadly, this class lopsidedness issue influences most of SDP procedures, as it causes misleading discoveries for different measures used to survey SDP execution [25]. This issue can be settled and SDP execution worked on by various strategies, including oversampling.

Distinguishing the item's faulty parts is the fourth stage in the SDP cycle. The decision of DL computations and cycles, which can incorporate a wide assortment of plans (for instance, Convolutional Mind Associations) and man-made intelligence classes (for instance, coordinated or not), is the fundamental worry in this step. Furthermore, a vital worry at this stage is the granularity of the hazardous parts that should be distinguished; these can be at the module, report, class, capacity, or sentence level, for instance [25].

Distinguishing the item's faulty parts is the fourth stage in the SDP cycle. The decision of DL computations and cycles, which can incorporate a wide assortment of plans (for instance, Convolutional Mind Associations) and man-made intelligence classes (for instance, coordinated or not), is the fundamental worry in this step. Furthermore, a vital

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

worry at this stage is the granularity of the hazardous parts that should be distinguished; these can be at the module, report, class, capacity, or sentence level, for instance [20].

Notwithstanding the scope of choices in the previously mentioned process steps, SDP studies can likewise be gathered by their conditions. Within-Project Defect Prediction (WPDP) and Cross-Project Defect Prediction (CPDP) are the two essential SDP situations that are ordinarily used in the writing. WPDP centers around shortcoming expectations inside a similar programming project on which it is prepared, utilizing verifiable information from a task (i.e., various variants) to foresee the faulty segments [26]. In this way, the preparation set and the test set are both piece of a similar undertaking.

Oppositely, CPDP trains a SDP model utilizing information from past tasks (source projects) and afterward uses the model to gauge the risky region of an alternate undertaking (target project) [27]. This procedure, which gets from move learning, is particularly vital when the objective venture needs more named preparing information. The critical test with this procedure is limiting the appropriation of elements hole between the source projects and the objective undertaking.

The greatest test in CPDP is that all tasks utilized in the CPDP situation should utilize similar measurements. By joining information from source and target projects into a solitary measurement space, heterogeneous deformity forecast (HDP) permits imperfection expectation across projects with different measurements [28].

Notwithstanding these SDP situations, In the nick of time Programming Deformity Expectation (JIT-SDP) is a popular technique for anticipating programming issues at the product change level [29]. Programming quality is guaranteed by JIT-SDP, otherwise called change level imperfection expectation, which empowers engineers to find and address mistakes rapidly. It has specific worth in SDP since it gives on time heading to engineers at a better granularity, i.e., change level. Designers may rapidly analyze and test their alterations utilizing JIT-SDP as opposed to sitting tight for extensive tests and tedious code audits. A last extraordinary SDP approach is Cross-Variant Deformity Forecast (CVDP) which examines the shortcoming information in prior renditions of a similar venture to expect the current form of the product project.

Stacking is the establishment whereupon the half and half coordination worldview is fabricated. The Stacking calculation makes a new informational collection for preparing the optional student in the wake of preparing the essential student utilizing the first informational index [30]. The essential student's result can be seen as another info include for the new informational index. Following is an outline of stacking's construction:

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

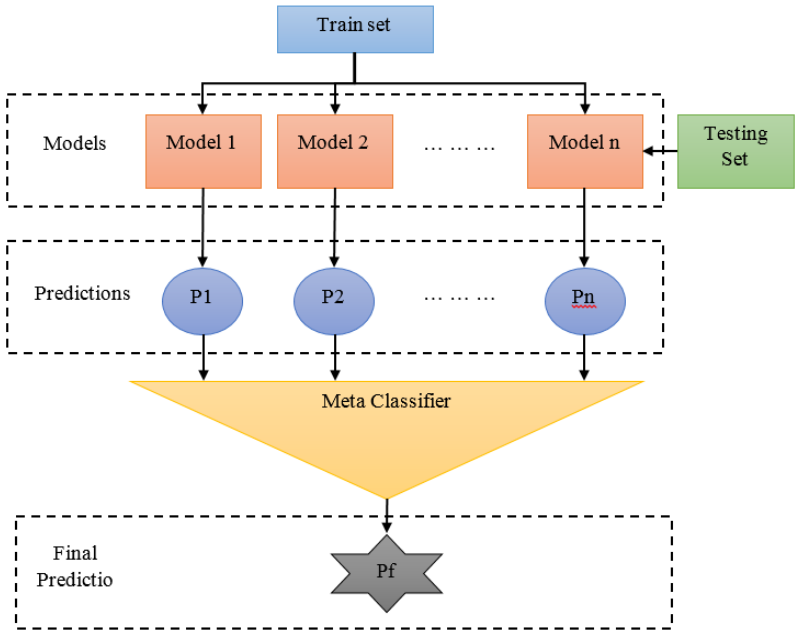

Figure 4: Stacking structure

The information layer, the secret layer, and the result layer make up the crossover reconciliation model. The classifier calculation in view of the help vector machine, Bayesian, choice tree, and so forth can be utilized as the singular student in the model layer [31]. Every hub's result is equivalent to the contribution of the hub in the layer beneath it. A reasonable classifier is regularly picked for the result layer, and to accomplish solid characterization execution, the democratic calculation component is applied.

4. RELATED WORKS [2]

In view of four (4) datasets from the NASA dataset, Researchers of [32] assessment of the different ML algorithms for SDP was proposed. As per the discoveries, no classification algorithm delivered the best outcomes across all datasets. However, it likewise shown that, as far as exactness, the model in view of occurrence learning and one rule fared better compared to the others.

Wahono, et al. [33] introduced a molecule swarm streamlining approach as a component determination strategy and the stowing approach for order. Nine NASA datasets were utilized in the review, alongside eleven order calculations, and the aftereffects of the correlation of the outcomes showed that everything except the SVM had gotten to the next level.

An original mixture technique between the SVM and CBA was set up by Rong et al. [34] that joins the qualities of the two classifiers: the advancement force of the Bat calculation joined with the centroid approach and the non-direct force of the last option. The examination was directed on a couple of notable datasets, and the outcomes were

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

contrasted with regards to exactness and those from recently distributed approaches. Other execution measurements are not considered in the review.

PSO algorithms and transformative metaheuristics calculations were used as the element choice methodology by Wahono, et al. [35]. Ten elective characterization calculations were used related to the packing strategy to address the lopsidedness issue. They viewed that as in spite of the fact that there was no way to see a contrast between the two calculations used for the component choice methodology, the packing technique performed better when molecule swarm enhancement and hereditary calculations were added.

In an examination of component determination procedures utilizing relationship based highlight choice (CFS), data endlessly gain proportion, Magal.R. what's more, Jacob [36] found that the CFS gave higher precision and utilized five different order calculations, with the outcomes showing that RF was the tried classifier with the most elevated exactness. Utilizing RF and the CFS include choice strategy, the exactness was 98.3%. Other execution pointers got no thought in the review.

Ibrahim, et al. [37] include choice technique and order SDP both utilized the Bat calculation and the RF. The viability of various component choice strategies in SDP was likewise examined in this review. In any case, exactness was the main basis utilized to assess the model's presentation.

Seven ensemble models for SDP were proposed by Aljamaan and Alazba [38], who classified the outfit models as endlessly helping groups. Eleven NASA imperfection data sets were utilized in the review. The outcomes for every outfit model were positive. Notwithstanding, precision and AUC were utilized as execution markers to evaluate the work. The SDP can't be legitimate utilizing basically these two standards. From these examination, it very well may be seen that a great deal of them involved exactness as their only execution metric. SDP execution can't be guaranteed by exactness alone.

Rathore & Santosh [39] a few troupes students were looked at against NB, LR, C4.5, SM, and RF; it was found that in pretty much every occasion, CRWM beat them as far as f-measure, exactness, and AUC values.

Tong et. al., [40] specialists utilized the W/D/L technique to direct an examination of the gauge draws near and the proposed methodology SDAEsTSE on every one of the picked datasets regarding F1, AUC, and MCC. The discoveries showed that the outfit approach SDAEsTSE beat the standard methodologies in most of the examinations.

Five outfit classifiers — Adaboost, Sacking, RSM, RF, and Vote — were contrasted with one classifier, J48, as well as to each other in [41]. In the troupes, J48 filled in as the base student too. It was exhibited that the characterization execution of all gatherings was better than that of the single classifier utilizing accuracy rate, F-measure, and AUC. RF performed better compared to different gatherings, as well.

Ten ensemble classifiers and 16 standard classifiers were analyzed in [42]. AdaBoostM1, LogicBoost, Multiboost Stomach muscle, Packing, RF, Dagging, Pivot woods (ROF),

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

stacking, multi plan, and casting a ballot were among the troupe learning techniques that were tried. NB, LR, MLP, RBF, SMO, Pegasos, Casted a ballot Perceptron, Occurrence based Student, KStar, Jrip, OneR, PART, J48, Truck, Hyperpipes, and Casting a ballot Element Spans were a portion of the major classifiers utilized. The F-measure and AUC were utilized to think about the results. J48's typical most noteworthy F-measure was 0.802 while MLP's typical most noteworthy AUC was 0.743 among base classifiers. The ROF group classifier scored the normal most extreme F-measure and AUC of 0.808 and 0.776, individually.

Rathore & Kumar[43], have 28 datasets were utilized to think about seven gathering draws near. The aftereffects of the trial examination showed that Enrich had the most elevated AUC worth of 0.986 and that the turning timberland involving J48 as the base student had the most extreme accuracy, review, and G-mean upsides of 0.995, 0.994, and 0.994, individually.

MLP, LR, DT, KNN, SVM, RF, ET, Sacking, AdaBoost, Slope Helping, XGBoost, and Stacking were a couple of the AI classifiers that were looked at by Mehta &Patnaik [44]. In each of the four datasets, stacking and XGBoost outperformed each and every classifier with regards to accuracy, review, exactness, and F-measure. In the PC1 dataset, XGB and Stacking had the most noteworthy exactness (0.968).

In summary, several studies have been conducted on software defect prediction (SDP) using different machine learning algorithms and techniques. Some of the findings from these studies include:

- A combination of feature selection techniques, such as particle swarm optimization (PSO) and genetic algorithms, with classification algorithms showed promising results in addressing the issue of class imbalance in SDP datasets.
- Some studies only considered accuracy as the performance metric, while ignoring other important metrics such as Matthews correlation coefficient (MCC) and receiver operating characteristic (ROC).
- Ensemble methods, such as stacking and bagging, were found to generally outperform individual base classifiers in terms of F-measure, accuracy, and area under the curve (AUC).
- Random forest (RF) was found to be a commonly used and effective classifier in SDP, often achieving high accuracy and F-measure.
- Some studies compared different ensemble methods against each other and found that RF performed better compared to other ensemble methods in terms of classification performance [67].
- Feature selection techniques, such as correlation-based feature selection (CFS), were found to improve the accuracy of SDP models.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

- Some studies proposed hybrid approaches that combine different classifiers or techniques, such as combining SVM and CBA, to leverage the strengths of multiple algorithms.
- Some studies evaluated the performance of SDP models using multiple performance metrics, such as F-measure, AUC, and MCC, to provide a more comprehensive assessment of model performance.

Overall, it is important to consider multiple performance metrics and experiment with different algorithms and techniques to achieve the best results in SDP. Ensemble methods, such as stacking and bagging, and feature selection techniques, such as CFS and PSO, have shown promising results in improving the accuracy and effectiveness of SDP models.

5. CONCLUSIONS

SDP has different strategies for recognizing defects and thus reducing the work expected to remediate them. Nowadays, due to the high volume of software and the restricted assets of quality assurance, this is particularly worthwhile. The DL is used to lead an effective written assessment of accessible SDP approaches. Help us collect the data, to find the distributions in the different Boolean datasets. We chose the papers to consider for examination because of a different quality assessment stage for the rater with clear criteria. Totally excellent 102 Core Exams were recalled for our overview. In light of this, we conducted a quantitative and subjective examination on the review group as for various parts of the SDP, including SDP situations, ML categories, data sets, source code depiction, level of forecast detail, class imbalance problem management, assessment measures and approach Agree, reproduce, finally challenges with various recommended arrangements.

References

1. A. Iqbal et al., “Performance analysis of machine learning techniques on software defect prediction using NASA datasets,” Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 5, pp. 300–308, 2019, doi: 10.14569/ijacsa.2019.0100538.
2. I. Arora, V. Tetarwal, and A. Saha, “Open issues in software defect prediction,” Procedia Comput. Sci., vol. 46, no. Icict 2014, pp. 906–912, 2015, doi: 10.1016/j.procs.2015.02.161.
3. A. N. R. Moparthi and B. N. Geethanjali, “Design and implementation of hybrid phase based ensemble technique for defect discovery using SDLC software metrics,” Proceeding IEEE - 2nd Int. Conf. Adv. Electr. Electron. Information, Commun. Bio-Informatics, IEEE - AEEICB 2016, pp. 268–274, 2016, doi: 10.1109/AEEICB.2016.7538287.
4. S. Czmil, J. Kluska, and A. Czmil, “CACP: Classification Algorithms Comparison Pipeline,” SoftwareX, vol. 19, p. 101134, 2022, doi: 10.1016/j.softx.2022.101134.
5. H. Hosseini, B. Xiao, M. Jaiswal, and R. Poovendran, “On the limitation of convolutional neural networks in recognizing negative images,” Proc. - 16th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2017, vol. 2017-December, pp. 352–358, 2017, doi: 10.1109/ICMLA.2017.0-136.
6. C. De Stefano, F. Fontanella, G. Folino, and A. S. Di Freca, “A Bayesian approach for combining ensembles of GP classifiers,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect.

ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

Notes Bioinformatics), vol. 6713 LNCS, no. May 2014, pp. 26–35, 2011, doi: 10.1007/978-3-642-21557-5_5.

7. J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. De Sousa, “Ensemble approaches for regression: A survey,” ACM Comput. Surv., vol. 45, no. 1, 2012, doi: 10.1145/2379776.2379786.
8. S. E. Lacy, M. A. Lones, and S. L. Smith, “A comparison of evolved linear and non-linear ensemble vote aggregators,” 2015 IEEE Congr. Evol. Comput. CEC 2015 - Proc., pp. 758–763, 2015, doi: 10.1109/CEC.2015.7256967.
9. Z. Yang, C. Jin, Y. Zhang, J. Wang, B. Yuan, and H. Li, “Software Defect Prediction: An Ensemble Learning Approach,” J. Phys. Conf. Ser., vol. 2171, no. 1, 2022, doi: 10.1088/1742-6596/2171/1/012008.
10. H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” Int. J. Data Min. Knowl. Manag. Process, vol. 5, no. 2, pp. 01–11, 2015, doi: 10.5121/ijdkp.2015.5201.
11. N. Ahmad Alawad and N. Ghani Rahman, “Design of (FPID) controller for Automatic Voltage Regulator using Differential Evolution Algorithm,” Int. J. Mod. Educ. Comput. Sci., vol. 11, no. 12, pp. 21–28, 2019, doi: 10.5815/ijmecs.2019.12.02.
12. N. Monga and P. Sehgal, “An Extensive Study of Various Software Defect Prediction Techniques,” NeuroQuantology, vol. 20, no. 14, pp. 291–302, 2022, doi: 10.4704/nq.2022.20.14.
13. N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, “Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data,” IEEE Geosci. Remote Sens. Lett., vol. 14, no. 5, pp. 778–782, 2017, doi: 10.1109/LGRS.2017.2681128.
14. S. A. Case, H. Zhao, Z. Chen, H. Jiang, W. Jing, and L. Sun, “Evaluation of Three Deep Learning Models for Early Crop Classification Using Sentinel-1A Imagery Time,” Remote Sens, vol. 11, no. 2673, pp. 1–23, 2019.
15. J. August, I. No, and A. Sharma, “Available Online at www.ijarcs.info International Journal of Advanced Research in Computer Science 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. C. W. Yohannese, T. Li, M. Simfukwe, and F. Khurshid, “Ensembles Based Combined Learning for Improved Software Fault Prediction : A Comparative Study,” 2017 12th Int. Conf. Intell. Syst. Knowl. Eng. Ensembles, 2017.
17. F. Matloob et al., “Software Defect Prediction Using Ensemble Learning : A Systematic Literature Review,” IEEE Access, vol. 9, no. July, pp. 98754–98771, 2021, doi: 10.1109/ACCESS.2021.3095559.
18. I. H. Laradji, M. Alshayeb, and L. Ghouti, “Software defect prediction using ensemble learning on selected features,” Inf. Softw. Technol., 2014, doi: 10.1016/j.infsof.2014.07.005.
19. H. Aljamaan, “Software Defect Prediction using Tree-Based Ensembles,” Assoc. Comput. Mach., vol. 8, no. 9, pp. 1–10, 2020.
20. K. E. Bennin, B. Tekinerdogan, I. Researcher, I. T. Group, and C. Science, “On the Use of Deep Learning in Software Defect Prediction,” J. Syst. Softw., no. October, 2022.
21. T. J. Mccabe, “A Complexity Measure,” IEEE Trans. Softw. Eng., vol. 2, no. 4, pp. 308–320, 1976.
22. S. R. Chidamber and C. F. Kemerer, “A Metrics Suite for Object Oriented Design,” IEEE Trans. Softw. Eng., vol. 20, no. 6, pp. 476–493, 1994.
23. N. Li, M. Shepperd, and Y. Guo, “A Systematic Review of Unsupervised Learning Techniques for Software Defect Prediction,” Inf. Softw. Technol., p. 106287, 2020, doi: 10.1016/j.infsof.2020.106287.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

24. N. Li, M. Shepperd, and Y. Guo, “A Systematic Review of Unsupervised Learning Techniques for Software Defect Prediction,” arXiv Prepr. arXiv, 2020.

25. X. Yu, M. Wu, Y. Jian, K. Ebo, B. Mandi, and F. Chuanxiang, “Cross-company defect prediction via semi-supervised clustering-based data filtering and MSTrA-based transfer learning,” Soft Comput., 2018, doi: 10.1007/s00500-018-3093-1.

26. S. Omri, “Deep Learning for Software Defect Prediction : A Survey,” IEEE/ACM 42nd Int. Conf. Softw. Eng. Work. Deep, pp. 209–214, 2020.

27. T. Mikolov, G. Corrado, K. Chen, and J. Dean, “Efficient Estimation ofWord Representations in Vector Space,” arXiv:1301.3781v3 [cs.SE], pp. 1–12.

28. P. Paramshetti and D. A. Phalke, “Survey on Software Defect Prediction Using Machine Learning Techniques,” Int. J. Sci. Res., vol. 3, no. 12, pp. 2012–2015, 2014.

29. G. G. Cabral, L. L. Minku, E. Shihab, and S. Mujahid, “Class Imbalance Evolution and Verification Latency in Just-in-Time Software Defect Prediction,” 2019 IEEE/ACM 41st Int. Conf. Softw. Eng., pp. 666–676, 2019, doi: 10.1109/ICSE.2019.00076.

30. T. Liang, X. Sheng, L. Zhou, Y. Li, H. Gao, and Y. Yin, “Mobile app recommendation via heterogeneous graph neural network in edge computing,” Appl. Soft Comput. J., vol. 103, p. 107162, 2021, doi: 10.1016/j.asoc.2021.107162.

31. T. Mohanraj, J. Yerchuru, H. Krishnan, R. S. N. Aravind, and R. Yameni, “Development Of Tool Condition Monitoring System In End Milling Process Using Wavelet Features And Hoelder’S Exponent With Machine Learning Algorithms,” Measurement, p. 108671, 2020, doi: 10.1016/j.measurement.2020.108671.

32. F. B. B. and I.-L. Y. VENKATA UDAYA B. CHALLAGULLA, “Empirical Assessment Of Machine Learning Based Software Defect Prediction Techniques Venkata,” Int. J. Artif. Intell. Tools, vol. 17, no. 2, pp. 389–400, 2008.

33. R. S. Wahono, U. Dian, N. Semarang, N. Suryana, and S. Ahmad, “Metaheuristic Optimization based Feature Selection for Software Defect Prediction,” J. Softw., vol. 9, no. 5, pp. 1324–1333, 2014, doi: 10.4304/jsw.9.5.1324-1333.

34. F. L. and Z. C. Xiaotao Rong, “A model for software defect prediction using support vector machine based on CBA,” Int. J. Intell. Syst. Technol. Appl., vol. 15, no. 1, 2016.

35. R. S. Wahono, U. Dian, N. Semarang, N. Suryana, and S. Ahmad, “Metaheuristic Optimization based Feature Selection for Software Defect Prediction,” no. May, 2014, doi: 10.4304/jsw.9.5.1324-1333.

36. K. M. R, T. Nadu, and S. G. Jacob, “Improved Random Forest Algorithm for Software Defect Prediction through Data Mining Techniques,” Int. J. Comput. Appl. (0975 – 8887), vol. 117, no. 23, pp. 18–22, 2015.

37. D. R. Ibrahim, “Software Defect Prediction using Feature Selection and Random Forest Algorithm Software Defect Prediction using Feature Selection and Random Forest Algorithm,” 2017 Int. Conf. New Trends Comput. Sci. Softw., no. October, 2017, doi: 10.1109/ICTCS.2017.39.

38. H. Aljamaan and A. Alazba, “Software Defect Prediction Using Tree-Based Ensembles,” in Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering, 2020, pp. 1–10. doi: 10.1145/3416508.3417114.

39. S. S. Rathore and S. Kumar, “A study on software fault prediction techniques,” Artif. Intell. Rev., 2017, doi: 10.1007/s10462-017-9563-5.

40. H. Tong, B. Liu, and S. Wang, “Software Defect Prediction Using Stacked Denoising Autoencoders and Two-stage Ensemble Learning,” Inf. Softw. Technol., vol. 96, Dec. 2017,

ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 05: 2023
DOI10.17605/OSF.IO/Q59B7

doi: 10.1016/j.infsof.2017.11.008.

41. R. Li, L. Zhou, S. Zhang, H. Liu, X. Huang, and Z. Sun, “Software Defect Prediction Based on Ensemble Learning,” in Proceedings of the 2019 2nd International Conference on Data Science and Information Technology, 2019, pp. 1–6. doi: 10.1145/3352411.3352412.
42. 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. an Int. J., vol. 23, no. 4, pp. 938–950, 2020, doi: https://doi.org/10.1016/j.jestch.2019.10.005.
43. S. S. Rathore and S. Kumar, “An empirical study of ensemble techniques for software fault prediction,” Appl. Intell., vol. 51, no. 6, pp. 3615–3644, 2021, doi: 10.1007/s10489-020-01935-6.
44. S. Mehta and K. S. Patnaik, “Improved prediction of software defects using ensemble machine learning techniques,” Neural Comput. Appl., vol. 33, no. 16, pp. 10551–10562, 2021, doi: 10.1007/s00521-021-05811-3.