Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Software Defect Prediction with a Stacking Ensemble Model

Maher F. Ismael
University of al-hamdaniya, Nineveh ,Mosul, Iraq.Email[1]: maher.f@uohamdaniya.edu.iq

ARTICLE INFO

Article history:
Received:28 /01/2026
Rrevised form:01 /04/2026
Accepted : 05 /04/2026
Available online: 30/06/2026

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

ABSTRACT

Software plays a critical role in modern systems, making defect prediction an essential task in software engineering. in all aspects of life, including in the industrial and medical fields, as designing reliable and high-quality software is considered one of the most important aspects of daily life to avoid potential risks, for example, financial and human, resulting from software errors, as they are considered a major challenge in terms of time and budget. Therefore, many companies have adopted and paid attention to developing their own software in terms of employing computer science, including its own algorithms, including algorithms that use machine learning to predict and detect these errors by using previous project records that contain the required data. This approach makes decision-making more accurate and effective in improving software performance and security. In our study, the use of ensemble machine learning was proposed, the purpose of which is to build a model to predict errors, as this approach helps in increasing the accuracy of prediction due to the diversity of data used... as well as improving the performance of defect classification models by using ensemble learning techniques based on the principle of stacking. In our work, we used the process of a combination among base learning algorithms (Bagging, Gradient Boosting, and CatBoost) and then fed into a meta learning algorithm such as XGBoost, in each dataset loaded containing a set of features and a target column called "Defective" that specifies whether the sample is defective or not. All possible combinations of these models (duplexes, triples) are generated and combined into a higher-order ensemble model known as a stacking classifier. In this context, NASA's MDP data warehouse, which specializes in defect prediction, was used, for feature selection and dataset balancing. The defect prediction was then evaluated, and evaluation metrics (accuracy, precision, recall, and F1 score) were calculated. The proposed model was executed with the help of the stacking method on the dataset that was used.

MSC..

https://doi.org/ 10.29304/jqcsm.2026.18.22616

1. Introduction

Modern software systems have become increasingly complex and large-scale, making software testing and quality assurance more challenging. With rapid technological advancements in the information technology sector, software has become a core component of many engineering systems and critical applications. Consequently, ensuring the quality and reliability of software systems has become a major concern in software engineering. Software reliability is closely related to the number of defects present in the system. A higher number of defects generally leads to lower software reliability and requires additional effort and cost to maintain software quality[1]. Software testing aims to ensure that software systems operate correctly and meet user requirements. However, achieving completely defect-free software is both time-consuming and costly. Furthermore, software failures can lead to serious consequences, particularly in safety-critical systems such as traffic control or aviation systems[2]. To address these challenges, software defect prediction techniques have been widely explored to identify fault-prone components at early stages of software development. These techniques typically apply machine learning models to predict defective software modules using historical software metrics. Early identification of defect-prone modules helps reduce testing costs, improve development efficiency, and enhance overall software quality. Various software metrics have been used for building prediction models, including McCabe metrics, Halstead metrics, and static code metrics, which capture

structural and complexity characteristics of the source code. In addition, ensemble learning techniques have gained considerable attention due to their ability to combine multiple classifiers and improve prediction accuracy[3].

Research Gap and Motivation

Although several machine learning and ensemble learning approaches have been proposed for software defect prediction, many existing studies focus on individual classifiers or limited ensemble configurations. In particular, the potential benefits of combining heterogeneous ensemble models using stacking strategies have not been sufficiently explored across multiple datasets. Moreover, limited attention has been given to systematically analyzing how different ensemble combinations affect prediction performance on datasets with varying characteristics. Therefore, there is a need for further investigation into stacking-based ensemble frameworks that integrate multiple learning algorithms and evaluate their effectiveness in software defect prediction. Motivated by these limitations, this study investigates the effectiveness of a stacking ensemble learning approach that combines several base learners to improve prediction performance and model robustness.

Main Contributions of This Study

The main contributions of this study can be summarized as follows:

1. Proposing a stacking-based ensemble learning framework that integrates Bagging, Gradient Boosting, and CatBoost as base learners with XGBoost as a meta-learner for software defect prediction.
2. Evaluating the proposed model on multiple NASA MDP benchmark datasets (PC1, PC2, PC3, and PC4), which enhances the reliability and reproducibility of the experimental results.
3. Investigating different combinations of ensemble models, including pairwise and triple configurations, to analyze their impact on prediction performance.
4. Conducting a comprehensive experimental evaluation using widely adopted classification metrics such as Accuracy, Precision, Recall, and F1-score.

2. Related Works

Several studies have been reviewed a number of studies related to software defect prediction and they were scrutinized to determine the most effective ways of predicting software defects, with the help of the approaches and techniques described in the thesis. Among these studies there are:

Researcher Iqbal and associates (2019) created a predictive framework of software defects based on ensemble classification and feature selection methods. The structure consists of three principal steps: the creation of a collection of various classifiers by the use of ensemble classification methods, the identification of the most significant features by the virtue of integrated feature selection, and the combination of outcomes of the analyzed classifiers. The framework uses two dimensions, including feature selection and no selection. Ensemble learning using a random forest classifier is done with two methods, Bagging and Boosting. The framework is tested on different datasets and was found to increase the prediction of software defects with a smaller number of features[4].

In 2020, Khuat and Le, authors of a study, compared the effectiveness of various sampling strategies and ensemble learning to the work on imbalanced software defect datasets. The findings suggested that balanced training data provided the best performance of the ensemble models as well as the base classifiers in comparison to using the original imbalanced datasets. The methodology is broken down into three parts namely; data balancing, training classifiers and classification [5].

Suresh Kumar et al. (2021), proposed a new Bagging model to predict the presence of software defects based on Bagging, which is an ensemble-based learning method. The model was evaluated in terms of its performance in comparison to the traditional machine learning algorithms, which include decision trees, K-nearest Neighbor, Support Vector machine, among others, on certain datasets. The proposed Bagging strategy had shown high effectiveness in prediction of software defects with an accuracy of more than 95% on various datasets, a higher score compared to the other models[6].

In 2021, researcher Ali and others examined the software defect prediction as follows: in some cases, and overall across all data sets, reducing the features set by 60 percent is used to improve measurement scalability and reduce processing overhead to the classifier, which assists in improving the quality of defect predictions. Second, applying logistic regression and decision tree-based prediction models on the selected features to improve defect prediction accuracy [7].

In 2022, researchers Yang and others utilized various machine learning techniques, specifically stacking, to develop an improved approach for building better models and enhancing evaluation methods. They also emphasized the need for appropriate preprocessing tasks that impact software defect data [8]

In 2023, researcher Ibrahim and others used ELFF datasets, based on 23 open-source Java projects, to employ various ensemble learning algorithms such as AdaBoost, Gradient Boost, Bagging, Random Forest, and their balanced versions to build a software defect prediction model on the ELFF dataset [9].
The primary aims of developing a model to predict software-related defects with the help of an ensemble machine learning are the following:
Maintain Predictive Strength of Defects: The ensemble machine learning integrates the forecasts of numerous models to generate a definitive forecasting, which is more precise than the forecasting of each model.
Increase the Accuracy of Predictions with Model combination: When several models are trained on varied subsets of data or with varying algorithms and then combined, the ensemble model will be in a position to capture a large variety of relationships and patterns within the data resulting in better predictions.
Make Defect prediction Models More Robust This is due to the fact that ensemble models are able to compute the averaged errors or biases of individual models which translates to the more stable and reliable predictions.

3. BACKGROUND OF THE RESEARCH

3.1. SOFTWARE DEFECT PREDICTION

In the realm of computer programming, programming imperfection expectation has gotten a great deal of interest. Furthermore, it can raise programming item quality [8], which can bring down improvement expenses and lift advancement viability. Using measurements that are connected with issues in the program, it makes a defect prediction model 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 Table (1):

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 defect prediction model, but the previously mentioned four are more delegate.

3.2. ENSEMBLE LEARNING ALGORITHMS AND ITS APPLICATION

A machine learning approach known as ensemble learning is applicable in 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]. The resultant explosion of high dimensions datasets, which have been driven by state-of-the-art data-collected and computed data, has brought with it an age of enormous analytical complexity. In this topography, bagging Classifier, or bootstrap gathering, has been developed as a practical but expensive method, with the ability to put to costs the volatility that plagues most estimate and classification processes. Bagging helps in taming overfitting and mute variance, and can therefore be an especially beneficial technique when the size and complexity of a problem are such that it is not possible to arrive at a fully optimised model after a single analytical run. [13]. Gradient Boosting is one of such sophisticated algorithmic models of supervised learning which, with proper use, may turn a noisy dataset into a extremely precise prediction model. We may take it apart, study its mechanics, its great force, and a few practical thoughts, which, with your projects, will make it shine.

Gradient Boosting works like a careful apprentice: with every additional tree, the preceding errors are coded, and the complete presentation is progressively enhanced. Through delicate adjustment of the learning rate, the depth of the trees, regularisation, and other parameters, an uncomplicated collection of decision trees can be made to the state-of-the-art on tabular data and often outperform more sophisticated neural models. [14]. CatBoost (the abbreviation of Categorical Boosting) is the classy answer that Yandex found to the ubiquitous categorical-feature nightmare haunting many tree-based learning algorithms. It is a gradient-boosted decision-tree public library, which not only takes categorical columns directly, but also changes them to a representation that can be directly used by the trees, without the typically time-consuming pre-processing choreography of one-hot encoding, routing lost values, or manually ordinal engineering. CatBoost can be described in simple words as the best (or the only) gradient-boosting package when your data is very much category based, or it has a great number of missing values, or you want to have top-quality results but not to write a mile-long list of pre-processing commands. [15]. XGBoost (Extreme Gradient Boosting) is an extremely optimized ensemble tree algorithm based on the tree-based algorithm, which has become an essential component of machine-learning processes because of its synergistic speed, accuracy, and scalability. Based on the gradient-boosting model, the algorithm progressively adds shallow decision trees to the model, trained in such a way that each tree is trained to correct the residual errors of its predecessors. This kind of iterative modification allows the model to capture the non-linear, complex relationships and maintain computational efficacy through features like sparsity-sensitive divided discovery, data-handling with cache-efficiency in the expression of high parallelism using multiple CPU cores or a big number of graphic card units. XGBoost has regularization strategy, L1 and L2 penalties, which reduce overfitting but simultaneously, its integrated cross-validation, early-stopping, and hyper-parameter optimization utilities, make the approach easy to implement by both beginners and experienced researchers. This has seen XGBoost take a lead in the frontier of predictive modeling with the model forming the basis of the highest-ranking solutions in Kaggle competition, industrial-scale analytics, and other research projects. [16]. As well as delivering a consolidated model with further developed execution by blending numerous 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 ensemble learning: measurements, calculation, and portrayal. Disintegration of deviation difference can likewise be utilized to assess the progress of gathering learning [17].

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 [8].

3.3. 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 [18][19].

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.1. 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) [20] sklearn [21], a Python AI bundle, and LIBSVM [21] are further devices.

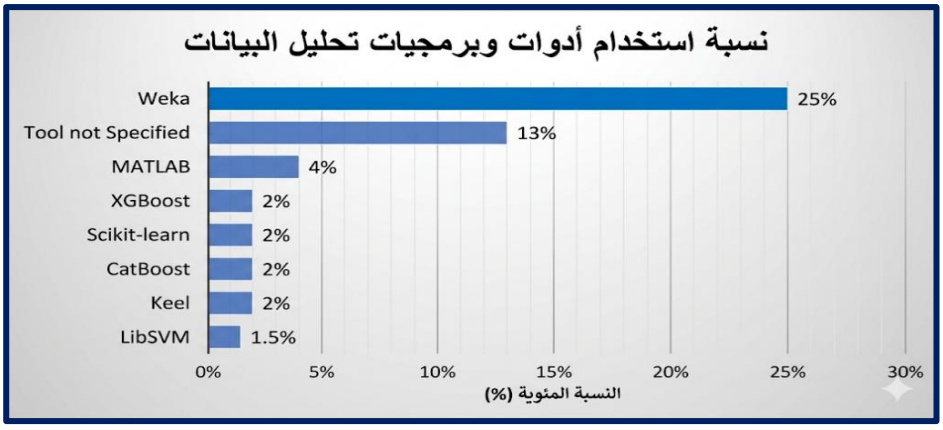

FIGURE 1. - Dispersion of essential exploration across libraries and AI devices

3.4. STACKING CLASSIFIER

It is a model used to achieve a final prediction using a diverse set of base classifiers. This is done by integrating the outputs of the base classifiers using a meta-classifier. The base model in this case is considered to be heterogeneous due to the fact that it mixes different classifiers which may vary in terms of structure and patterns. The meta-classifier takes into consideration the prediction of the base classifiers and in the process produces a more precise final prediction of the target desired. In this step, original prediction results are created by the initial base classifier and they are of new features. These characteristics are then presented as an input to the second meta-classifier which examines these characteristics and utilizes them to give a more precise final prediction to the targeted object. Thus, the model enhances the performance of prediction because it draws knowledge of the meta-classifier and incorporates this with the predictions of the base classifiers. In this model individual knowledge of the base classifiers is merged with the overall analysis of the meta-classifier, which leads to an accurate and reliable end prediction. This enables the maximum use of the strengths and capabilities of the individual classifiers and will improve by the process of aggregation[22].

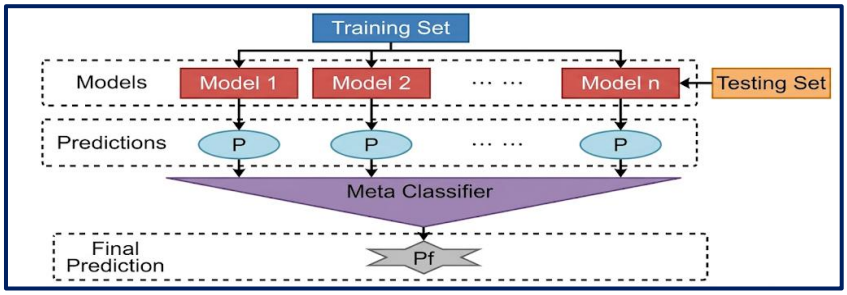

FIGURE 2. - Stacking structure

4. *METHODOLOGY*:

In this paper an ensemble learning stacking model is presented which is used to predict software defects using Metrics Data Program (MDP) datasets of NASA (PC1, PC2, PC3 and PC4).. Every dataset includes module-based

metrics and binary labels of defects Data preprocessing was performed on the dataset used, as data preprocessing is a critical step in machine learning. It involves transforming raw and inconsistent data into an organized and usable format. This process is necessary because real-world data often contains missing, unnecessary, or inconsistent information, which can hinder the discovery of valuable patterns and insights. and we make use of stratified sampling to divide the data in training and test set, hence maintaining an equal representation of classes- which is a common strategy in defect-prediction studies. Its methodology proceeds in five steps, namely: (1) preparation of a dataset, (2) training of various base learners, (3) creating meta-features based on the output of the base-models, (4) training a meta-model on the meta-features, and (5) a serious performance assessment. There are three different machine learning algorithms that are used as base learners to represent complementary decision boundaries. Bagging Classifier eliminates variance by learning various models on bootstrapped samples and combining their predictions. Gradient Boosting Classifier is a series of models that are trained in a cascading fashion, each attempting to fix the mistakes of the models developed before it. CatBoost Classifier is effective in categorical features and it avoids prediction shift with ordered boosting. The individual base models would be trained on the same training data and would give the probability estimates of whether a module is defective or not. After the stacked generalization approach suggested, the results of the base models are gathered and combined to create another feature. The prediction probabilities of each software module of the sampled base learners are stacked into a single-vector. This involves building a meta-feature dataset which models the behavior of the underlying models, and which is input to the meta-learner[23,24]. Building such meta-feature vectors is intuited by previous research on stacking ensembles. The meta-feature dataset is subsequently trained on a second-level model XGBoost which is characterized by its scalability, regularization and high predictive ability. The meta-model is taught to combine the base model outputs in the best way possible to enhance the overall accuracy.

After training, the meta-model uses the output of the base models together to provide predictions of other software modules. This last prediction is based on the combined opinion of the ensemble about the probable defectiveness of any given module, capitalizing on the strengths of each base learner to accomplish that. To measure the effectiveness of the stacking framework, the standard classification measures are employed, such as Accuracy, Precision, Recall, and F1 Score. These measures evaluate the performance of generalization and the sensitivity to class imbalance, This methodology is explained in Figure(3).

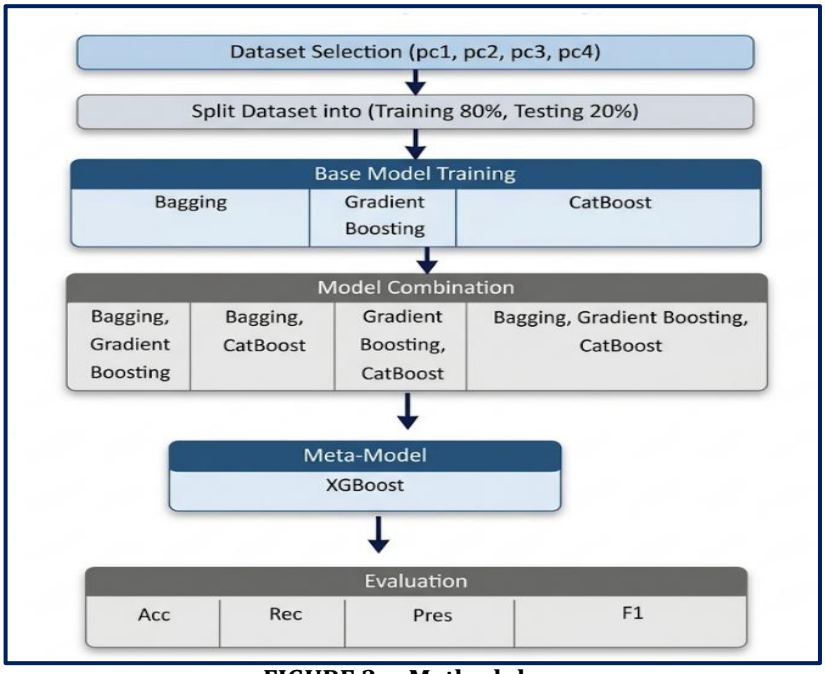

FIGURE 3. – Methodology

5. Results and Discussion

This section presents and discusses the results of each dataset which conducted using Python 3.4 and pandas for loading and manipulating CSV datasets, Detailed results and insights derived from the classifier performance are presented in tables from(2-5)

Table 2: The result on pc1 dataset

id	Model Combination	accuracy	Precision:	Recall:	F1 Score:
1	Bagging,Gradient Boosting	0.93798	0.91852	0.96124	0.93939
2	Bagging, CatBoost	0.93798	0.92481	0.95349	0.93893
3	Gradient Boosting, CatBoost	0.94574	0.93893	0.95349	0.94615
4	Bagging,GradientBoosting, CatBoost	0.93798	0.92481	0.95349	0.93893

Table 3: The result on pc2 dataset

id	Model Combination	accuracy	Precision:	Recall:	F1 Score:
1	Bagging,Gradient Boosting	0.9726	0.98592	0.9589	0.97222
2	Bagging, CatBoost	0.9863	0.9754	0.9644	0.97238
3	Gradient Boosting, CatBoost	0.9726	0.97917	0.96575	0.97241
4	Bagging,GradientBoosting, CatBoost	0.97603	0.98601	0.96575	0.97578

Table 4: The result on pc3 dataset

id	Model Combination	accuracy	Precision:	Recall:	F1 Score:
1	Bagging,Gradient Boosting	0.88095	0.85294	0.92063	0.8855
2	Bagging, CatBoost	0.89947	0.88325	0.92063	0.90155
3	Gradient Boosting, CatBoost	0.91534	0.89055	0.94709	0.91795
4	Bagging,GradientBoosting, CatBoost	0.91799	0.90722	0.93122	0.91906

Table 5: The result on pc4 dataset

id	Model Combination	accuracy	Precision:	Recall:	F1 Score:
1	Bagging,Gradient Boosting	0.95495	0.97196	0.93694	0.95413
2	Bagging, CatBoost	0.94996	0.97664	0.94144	0.95872
3	Gradient Boosting, CatBoost	0.95946	0.96789	0.95045	0.95909
4	Bagging,GradientBoosting, CatBoost	0.95495	0.96759	0.94144	0.95434

A comparison was conducted to assess the performance of the proposed model for defect prediction in contrast to a set of previous works. The comparison included the utilization of the dataset used for defect prediction (PC1, PC2, PC4,) by employing accuracy metrics to determine the best-performing model, as illustrated in Table (6):

Table 6: Comparison of the results of the proposed model with previous works

			The proposed model algorithm.		
		PC1	PC2	PC3	**PC4**
	Our Stacking	0.94574	0.9863	0.91799	**0.95946**
[25]	Ensemble oversampling	0.930	0.134	0.843	**0.906**
[26]	Omni – ensemble learning (OEL)	0.9096	0.9748	-	**-**
[4]	Boost RF	0.955882	0.976959	0.860759	**0.913386**
	Boost RF&Fs	0.960784	0.972350	0.873418	**0.916010**
	Bagging RF	0.960784	0.976959	0.867089	**0.902887**
	Bagging RF&Fs	0.960784	0.976959	0.873418	**0.908136**
[27]	Hellinger net(A Hybrid imbalance learning model)	0.854	0.66	0.835	**0.842**
[7]	Best features-based cost-sensitive Logistic regression performance	0.75	-	0.80	**0.75**
	Best features-based cost-sensitive decision tree ensemble performance	0.88	-	0.80	**0.86**

From the table (5) above, we observe the superiority of the results of the proposed model over all the results of previous studies in terms of using the same dataset employed in each study.

5. Conclusions

This research involved designing a model for predicting software defects using ensemble machine learning models (Stacking). The comparison with previous works, using the same dataset for software defect prediction (PC1, PC2, PC4), led to the following conclusions:

The proposed model can successfully contribute to the discovery of software defects during software development stages. It also helps the developers and code testers to evaluate software defects efficiently and better, so that they can predict defects more accurately and within less time. Preprocessing data was found to largely contribute to efficiency of the model in assessing the algorithms used in feeding the training models and proper data analysis and processing. Enhancement of the performance of ensemble machine learning algorithms was done through grid search with cross-validation to optimize the parameters. The data were broken down into 5 subsets and the model was trained several times to identify the best parameter configuration. The effectiveness and accuracy in predicting the defects was done by employing a wide range of assessment techniques that included; quantitative performance analysis of the Confusion Matrix and the qualitative evaluation of the individual defects. It was determined that ensemble machine learning algorithms were more accurate compared to single machine learning algorithms. The reason is that ensemble algorithms have the ability of using overall context and data interactions which minimize noise and are therefore better at predicting defects.

References

[1] T. Zhang, Q. Du, J. Xu, J. Li, and X. Li, "Software defect prediction and localization with attention-based models and ensemble learning," *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, vol. 2020-Decem, pp. 81–90, 2020, doi: 10.1109/APSEC51365.2020.00016.

[2] T. Menzies, R. Krishna, and D. Pryor, "e Promise Repository of Empirical So ware Engineering Data. hp," *openscience. us/repo. North Carolina State Univ. Dep. Comput. Sci.*, 2015.

[3] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Appl. Soft Comput.*, vol. 27, pp. 504–518, 2015.

[4] A. Iqbal, S. Aftab, I. Ullah, M. S. Bashir, and M. A. Saeed, "A feature selection based ensemble classification framework for software defect prediction," *Int. J. Mod. Educ. Comput. Sci.*, vol. 11, no. 9, p. 54, 2019.

[5] T. T. Khuat and M. H. Le, "Evaluation of sampling-based ensembles of classifiers on imbalanced data for software defect prediction problems," *SN Comput. Sci.*, vol. 1, no. 2, p. 108, 2020.

[6] P. Suresh Kumar, H. S. Behera, J. Nayak, and B. Naik, "Bootstrap aggregation ensemble learning-based reliable approach for software defect prediction by using characterized code feature," *Innov. Syst. Softw. Eng.*, vol. 17, no. 4, pp. 355–379, 2021.

[7] A. Ali, N. Khan, M. Abu-Tair, J. Noppen, S. McClean, and I. McChesney, "Discriminating features-based cost-sensitive approach for software defect prediction," *Autom. Softw. Eng.*, vol. 28, pp. 1–18, 2021.

[8] Z. Yang, C. Jin, Y. Zhang, J. Wang, B. Yuan, and H. Li, "Software Defect prediction: An Ensemble Learning Approach," in *Journal of Physics: Conference Series*, 2022, vol. 2171, no. 1, p. 12008.

[9] A. M. Ibrahim, H. Abdelsalam, and I. A. T. F. Taj-Eddin, "Software Defects Prediction At Method Level Using Ensemble Learning Techniques," *Int. J. Intell. Comput. Inf. Sci.*, vol. 23, no. 2, pp. 28–49, 2023.

[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] P. Bühlmann and B. Yu, "Analyzing bagging," The Annals of Statistics, vol. 30, no. 4, pp. 927–961, 2002.

[14] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of Gradient Boosting algorithms," Artificial Intelligence Review, vol. 54, no. 3, pp. 1937–1967, 2021.

[15] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," in Advances in Neural Information Processing Systems, vol. 31, 2018.

[16] M. Nalluri, M. Pentela, and N. R. Eluri, "A scalable tree boosting system: XGBoost," Int. J. Res. Stud. Sci. Eng. Technol., vol. 7, no. 12, pp. 36–51, 2020.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] F. Matloob *et al.*, "Software defect prediction using ensemble learning: A systematic literature review," *IEEE Access*, vol. 9, pp. 98754–98771, 2021, doi: 10.1109/ACCESS.2021.3095559.

[22] Y. Xia, K. Chen, and Y. Yang, "Multi-label classification with weighted classifier selection and stacked ensemble," *Inf. Sci. (Ny).*, vol. 557, pp. 421–442, 2021.

[23] S. Antonik and B. Bąba, "Stacked Generalization—Investigating the impact on predictive performance of basic machine learning models," vol. 3885, p. 49, 2024.

[24] C. Giraud-Carrier, "Combining base-learners into ensembles," in METALEARNING: APPLICATIONS TO AUTOMATED MACHINE LEARNING AND DATA MINING, Cham: Springer International Publishing, pp. 169–188, 2022.

[25] S. Huda et al., "An ensemble oversampling model for class imbalance problem in software defect prediction," IEEE access, vol. 6, pp. 24184–24195, 2018.

[26] R. Mousavi, M. Eftekhari, and F. Rahdari, "Omni-ensemble learning (OEL): utilizing over-bagging, static and dynamic ensemble selection approaches for software defect prediction," Int. J. Artif. Intell. Tools, vol. 27, no. 06, p. 1850024, 2018.

[27] T. Chakraborty and A. K. Chakraborty, "Hellinger net: a hybrid imbalance learning model to improve software defect prediction," IEEE Trans. Reliab., vol. 70, no. 2, pp. 481–494, 2020.