Kufa Journal of Engineering
Vol. 16, No. 3, July 2025, P.P. 639-657
Article history: Received 25 June 2024, last revised 16 January 2025, accepted 16 January 2025

PREDICTION OF SOFTWARE ANOMALIES METHODS BASED ON ENSEMBLE LEARNING METHODS

Raghda Azad Hasan[1] and Ibrahim Ahmed Saleh[2]

[1] MSC student, Software Department, Faculty of Computer Science and Mathematics, University of Mosul, Nineveh, Iraq. Email: raghda.22csp16@student.uomosul.edu.iq

[2] Professor, Department of Software, Faculty of Computer Science and Mathematics, University of Mosul, Nineveh, Iraq. Email: i.hadedi@uomosul.edu.com

https://doi.org/10.30572/2018/KJE/160336

ABSTRACT

Software plays a vital role in all aspects of our daily lives, specifically in the fields of medicine and industry. In order to design high-quality and reliable software and avoid risks resulting from software errors, including physical and human errors, this is considered a major challenge due to the limited time and budget specified. Therefore, most software development companies tend to use machine learning for prediction. With the presence of software defects that contribute to improving the quality and safety of the software produced, this is done by relying on and using records, previous projects, and available data. this paper proposed machine learning and ensemble learning suite to predict software anomalies. The evaluated approach is for models in the PROMISE real-word dataset repository containing 5 projects (Turkish company SOFTLAB). The model applies the basic algorithms (Random Forest (RF), Decision Tree (DT), Extra Tree) and the learning model ensemble (Adaboost, xgboost ,Stack, Voting, bagging) and metrics (accuracy, recall, F1 score, accuracy) to measure the prediction performance of the models and a comparison was made between the proposed model algorithms. Both adaboost , stack achieved prediction accuracy about 99.2% when implemented on the ar5 dataset.

KEYWORDS

Software Engineering, Software Defect Prediction, Ensemble Learning, Random Forest, Decision Tree, Boosting, Stacking.

1. INTRODUCTION

The importance of identifying anomalies in software systems has increased over the past few years. With increasing software applications in our daily lives, ensuring its quality has now become extremely important. Because modern software systems are complex and implicitly interconnected, the quality assurance process, in turn, is insufficient for large systems that are constantly updated (Balogun,2020). Software failures have been led to catastrophic disasters so it's is necessary to link software failures, underscoring of effective anomaly detection. Furthermore, software bugs and glitches have a significant financial impact. The field of software engineering always gives priority to producing high-quality software. The process of detecting software anomalies is an important part of software development. If software defects are found early in software development process, this enables QA experts to focus on the problematic modules rather than the entire software. This method can reduce development costs without sacrificing the quality of the final product. Early detection of problematic modules can facilitate early changes, ensure timely delivery of a high-quality product that satisfies customers, and enhance confidence of development team. Anticipating software defects can reduce testing and maintenance costs while increasing the quality of the final output. A simple software glitch can lead to serious consequences and system crashes. Software defects are of different types, including incorrect program data, errors in design, installation, specifications, and others. Since the testing phase is the most expensive in the software development life cycle, discovering and fixing software defects before the testing phase. makes identifying the causes of failure easier and less expensive, as the process of predicting software defects is an essential part of software testing (Shi and Abbas ,2023) and (Abdou,2018) Over the past two decades, many algorithm such as statistics, neural network, machine learning techniques….etc used to predict and detect software anomaly. To achieve high prediction accuracy, ensemble learning is used, which combines many individual classifiers, which is more accurate in prediction than using a single classifier. In general, machine learning techniques are divided into three types**:**

- First supervised techniques, which need pre-trained training data.
- second, unsupervised techniques, which need to use specific algorithms to identify and structure the data,
- Finally semi-supervised hybrid techniques Combines the two types.

One critical element that makes it possible to reduce field failure rates and increase the reliability of complex software systems is the ability to anticipate or identify problems before they arise. Various methods have been proposed in many recent researches to identify anomalies in software systems (Monni ,2019) and (Hersh A. Mohammed et al,2020). Because

ensemble learning techniques can enhance the model's capacity for generalization, they are employed to address the issue of class imbalance. Different classifiers can produce different classification errors because they are trained on different sets of data. That being said, each classifier's matching errors aren't always the same. The ensemble learning approaches, which combine these classifiers through various mechanisms, can mitigate the biased learning resulting from class imbalance classification. Both bagging and boosting are popular ensemble learning techniques. Boosting-based approaches concentrate on hard-to-classify cases and continuously train each classifier using data from a weighted sample of the original data. Through the use of bootstrapped clones of training data, bag-based techniques construct distinct classifiers (AL-FRAIHAT ,2024)and(Balogun ,2018). Since ensemble learning techniques are widely used for the detection and prediction process and are working to improve them, they have recently been widely used (Huda ,2021). The goal of this research is to improve the prediction of software anomalies, where a framework is presented for predicting anomalies in a software system and using ensemble learning techniques (Matloob ,2019),(Sharm ,2023) and (Zhao ,2017). It presents a framework for predicting anomalies in a software system and using ensemble learning techniques. The paper proposes ensemble learning methods to predict software anomalies to obtain the best prediction accuracy, where ensemble learning models (bagging, stacking, XGboost, Adaboost) were used. Using (Random Forest (RF) , Decision Tree (DT)) algorithms as two basic parameters and trained on a dataset from a Turkish white goods company SOFTLAB.

The rest of the paper refers to related works that focuses on reviewing the prediction based on ensemble learning described in Section 2. While Section 3 includes the paper's methodology, dataset, preprocessing, and machine learning methods, the experimental results are compared and analyzed in detail at the end.

2. RELATED WORK

Prediction software anomalies are a process of predicting abnormal performance phenomena anomalies to frustrate future incidents. Many researchers worked about software anomalies some about detection and other about prediction, and artificial intelligence methods are widely applied. This section will briefly review the most important methods based anomaly detection methods and specifically highlight ensemble learning.

In 2018, Abdul Latif O. Balogun et al. It is argued that clustering methods can give better results in software defect prediction (SDP). They evaluated the performance of individual classifiers (Sequential Minimal Optimization (SMO), Multi-layer Perceptron (MLP), k-Nearest Neighbor

(kNN), decision tree) and ensemble classifiers (bagging, boosting, stacking, and voting) in SDP using 11 defective datasets. programs and also 11 performance measures, and the results of their proposed method showed that boosting is the best prediction method, and among the individual classifiers, the decision tree ranked first with a score of 0.0410, and they stressed the need to take into account performance measures to achieve the best predictive performance, so that they are implemented before selecting the model or Process classifier (Balogun ,2018).

In 2019, REN and LIU presented a predictive study using the self-data mining method to predict software defects in classification and ranking. They proved that software metrics and software defects are causally related. During the model training phase, when predicting the ranking, the errors of defect-free units are replaced with a negative value, and the errors of defective units remain unchanged. While the true values of defective units are replaced with a positive value ≥1.5, the false values of defect-free units are replaced with a negative value during classification prediction. Using the NASA, Promise, and Softlab datasets, the self-data mining approach is a very practical and effective method for predicting software defects (REN and LIU, 2019).

In 2020, Umair Ali and others presented a classification framework for prediction using ensemble learning and feature selection techniques to achieve high performance, where variable that causes low performance is eliminated. They proposed framework works using four data sets from MDP repository, the results of proposed framework showed that this framework outperforms ten basic classifiers that are subject to supervision, including the algorithms "Naïve Bayes (NB), Multi-Layer Perceptron (MLP), Radial Basis Function (RBF), Support Vector Machine (SVM), K Nearest Neighbor (KNN), k-Star (K*), One Rule (OneR), PART, Decision Tree (DT), and Random Forest (RF)." The performance was evaluated using three Performance metrics including F-measure, Accuracy, and MCC (Ali,2020). In the same year, Muhammad Amimul Ihsan Aqeel and Wan Hussein Wan Ishaq conducted a study on Ensemble, classification and clustering techniques to improve the quality and efficiency of software defect prediction. Their study was conducted using 13 data sets from the NASA MDP repository, and the prediction performance was evaluated using 3 metrics including (Accuracy, F-measure, MAE). Where ensemble learning models were used (RF, ET, XgBoost, LGBM, STC), As for supervised models (PAC, QDA, GNB) and regarding unsupervised learning models (KNN, GMM, K-mean) The results were decisive for the high-accuracy prediction of ensemble learning techniques, with STC outperforming all algorithms on all three metrics. QDA achieved high performance, while for the unsupervised learning algorithms, the KNN and GMM algorithms excelled for the three performance measures (Mohammad Amimul Ihsan,2020).

In 2021, Yaqub Kayode Sahid et al presented a model to eliminate what previous studies had struggled with in terms of prediction accuracy and performance metrics. Considering that most previous studies relied on accuracy to measure SDP which was insufficient, they presented their approach that uses six ensemble learning models (Cat Boost, Light Gradient Boosting Machine (LGBM), Extreme Gradient Boosting (XgBoost), Cat Boost, Logistic Regression, LGBM Boosting, XgBoost Boosting) was trained on three NASA data sets and its performance was evaluated using performance metrics (accuracy, area under the curve (AUC), accuracy, recall, F-measure and Matthew Correlation Coefficient (MCC). The results showed the superiority of the Cat Boost system in terms of reducing training time and excessive setup. The model was compared with the basic logistic regression model for six sets of data, and the result was high performance of the proposed ensemble t model (Saheed et al,2021).

In 2023, R. Mamatha and others emphasized importance of using machine learning for predicting software defects, they applied promise warehouse data sets for this task. The model integrating Naive Bayes and Boosting techniques to form a model whose predictions are strong and more reliable. The data was improved with accuracy and high performance, as these combined models allowed for the elimination and removal of excess structure, bias, and also variance. In general, the integration of algorithms showed the strengths and weaknesses of each classifier separately (R. Mamatha et al,2023). In the same year, Junyi Xin et al, By creating initial models, a processing process is carried out on the images, after which models of proposed approach based on ensemble learning are implemented to detect defects in images. The convolutional neural network models ((off-the-shelf CNN, bagged CNN, and boosted CNN) are also compared. The results of the proposed approach showed high performance of the bagged CNN network, with training and testing accuracy reaching (96.1% and 95.1%), respectively (Xin, J. et al,2023).

Also in the same year 2023, Sagheer Abbas and others also presented an intelligent system for predicting software anomalies (defective modules) present within software. The system is based on feature selection and clustering machine learning techniques. This approach was evaluated and trained on five datasets from NASA. This system consists of three stages for better forecasting. In the first stage, the applied decision tree is combined with support vector machines and naive Bayes. In the second stage, predictions of the techniques (mobilization, voting, and stacking) are combined. The final stage is applied to fuzzy logic to improve predictive accuracy. The results of the proposed system showed a predictive accuracy of up to (92.08%), and this result demonstrates the superiority of the system's performance over many other advanced techniques (Sagheer Abbas,2023).

In 2024, Misbah Ali et al presented a two-stage approach in which four supervised algorithms are used. In the second stage, individual algorithms and classifiers are combined to improve predictive capabilities. They relied on using NASA data from the MDP repository. Seven datasets were used to implement and train the two stages. Iterative parameter optimization was performed in the first stage to increase and improve the performance accuracy. In the second stage, the base classifiers are combined to make final high-accuracy predictions from these individual classifiers (Random Forest). , Support Vector Machine, Naïve Bayes, and Artificial Neural Network) The authors confirmed the superiority of the proposed approach voting ensemble-based software Defect prediction (VESDP) model based on twenty modern technologies (Misbah Ali et al,2024).

3. METHODOLOGY

This part present the methodology, where the method used ensemble learning to predict software anomalies is described. The motivation behind its use is that it improves the generalization ability of software defect prediction algorithms, and that the application of ensemble learning algorithms has achieved achievements. For published literature. The RF and DT algorithms were applied as individual basic classifiers in the proposed model.

3.1. Data Set

The PROMISE repository provided datasets for this study, including other software modules. The datasets were pre-processed by cleaning and normalizing the raw data to ensure compatibility with the models. For instance, Halstead complexity, code size, and McCabe cyclomatic complexity metrics were computed for each software module. The processed dataset was divided into 80% training data and 20% test data.

The input process involves mapping these computed metrics into the model's input structure. Specifically, features such as code complexity metrics were scaled to ensure balanced input ranges. If the software predicts defects, the program first extracts these features, organizes them in structured datasets, and then feeds them into the prediction model. This ensures the seamless integration of data collection and input into the prediction workflow.

Table 1. SOFTLAB Dataset.

Dataset	Instances	Attributes	Minority	Majority
ar1	122	30	9	113
ar3	64	30	8	56
ar4	108	30	20	88
ar5	37	30	8	29
ar6	101	30	15	87

Table 2. SOFTLAB Dataset Attribute

NO.	Attribute
1	total_loc numeric
2	blank_loc numeric
3	comment_loc numeric
4	code_and_comment_loc numeric
5	executable_loc numeric
6	unique_operands numeric
7	unique_operators numeric
8	total_operands numeric
9	total_operators numeric
10	halstead_vocabulary numeric
11	halstead_length numeric
12	halstead_volume numeric
13	halstead_level numeric
14	halstead_difficulty numeric
15	halstead_effort numeric
16	halstead_error numeric
17	halstead_time numeric
18	branch_count numeric
19	decision_count numeric
20	call_pairs numeric
21	condition_count numeric
22	multiple_condition_count numeric
23	cyclomatic_complexity numeric
24	cyclomatic_density numeric
25	decision_density numeric
26	design_complexity numeric
27	design_density numeric
28	normalized_cyclomatic_complexity numeric
29	formal_parameters numeric
30	defects {clean,buggy}

3.2. Data pre-processing

After selected training dataset, pre-processing of this data is performed, which includes four sub-activities that improves the performance of proposed model: 1) dataset partitioning, 2) cleaning, 3) normalization, and 4) oversampling.

The first sub-activity as mentioned previously of the pre-processing phase is process of partitioning data set. The training and testing datasets are divided into two groups in this step, 80% and 20%.

Cleaning, the second sub-activity is essential for the robustness of the model. Cleaning removes inconsistent, incorrect, or unnecessary data points to ensure and enhance the accuracy of anomaly detection and prediction models (Sivalingan H, 2024). By reducing noise, addressing missing values, checking consistency, and correcting errors within the dataset, this stage improves the quality and integrity of the data and produces better predictions by replacing

missing values in the dataset and using a statistical technique (mean) that allows gaps to be filled with appropriate values in the data being used.

Normalization is a sub-activity of the third pre-processing step. Normalization is a commonly used technique to balance feature scales between (0 and 1) to measure and standardize the attributes of the input dataset. Normalization helps the machine learning model become more stable and efficient while also facilitating convergence (Sivalingan H, 2024). By preventing large-scale attributes from unduly affecting the model, it helps in achieving a neutral and balanced learning process as a result. In addition, normalization helps ensure that the model works consistently by managing differences in the data distribution (Mesbah Ali et al., 2024). Logarithmic transformation, a technique that uses logarithmic transformation to transform data by applying a logarithm to each value, was used to reduce the effect of outliers and disparities between large and small numbers and improve the distribution by making the distribution close to the normal distribution of values, as (log) helps address the problem of dealing with zero values by adding one to each value before taking the logarithm (Dina Saeed et al., 2011) log(1+X)

$$X' = \log(1 + X) \qquad (1)$$

represents: X the original value in the dataset and X' represents the transformed value This type of transformation was used because the data used contains zero values.

Oversampling is a sub-activity of the four pre-processing steps to solve the problem of class imbalance when training models on imbalanced data. The oversampling technique solves the imbalance problem by increasing the defective or abnormal minority class. Thus, a balance is achieved between the two classes, the normal majority class and the abnormal or defective minority class (Thanh, 2020;Shu Feng et al., 2021; and Jiao Chen et al., 2021).

3.3. MODELS

The motivation behind using ensemble learning is achieve most accurate in machine learning where multiple models (often called "base learners" or "weak learners") are combined to create a more robust and accurate predictive model. The main idea is that by aggregating the predictions from multiple models, the ensemble can achieve better performance than any single model could on its own. There are several common methods for creating ensembles, including bagging, boosting, and stacking. Typically, there are two processes involved in creating an ensemble of classifiers: 1. combining the classifier predictions; 2. Training several base classifiers (Jun-hai Zhai et al,2012). From this angle, ensemble learning can be combined a wide range of machine learning models to perform a wide range of tasks, including model

clustering and classification tasks (Xibin DONG et al,2020). In this paper table 3 shows the selection of hyper-parameters for ensemble learning models

Table 3. hyper-parameters for the proposed models

Models	Parameters of SOFTLAB data
Bagging classifier	estimator=RF n_estimator=500 random_state=42 max_feature=5 tarin_size=80 test_size=20
Voting classifier	Estimators= Extrees,Catboost,GB tarin_size=80 test_size=20
Stacking classifier	estimators=RF,SVM final_estimator=GB
Xgboost	n_estimator=700 learn_rate=0.3 tarin_size=80 test_size=20
Adaboost	n_estimator=300 random_state=42 learn_rate=0.3 tarin_size=80 test_size=20

Parameters directly affect the prediction accuracy, as they determine how well the model learns from the data and how complex it is. For example, increasing the number of classifiers (n_estimators) usually improves performance, but only up to a point, after which it may lead to longer training times without significant improvement. Learning rate (learning_rate) balances learning speed and accuracy; smaller values give more accurate results .

3.3.1. Diction Tree

A popular data mining technique for creating multi-covariate classification systems or creating prediction algorithms for a target variable is decision tree approach. By using this strategy, a population is divided into segments that resemble branches, creating an inverted tree with leaf, internal, and root nodes. Being non-parametric, the technique may effectively handle sizable, complex datasets without requiring a convoluted parametric structure. (Yan-yan SONG,2015) The research data can be separated into training and validation datasets once the sample size is sufficiently big. constructing a decision tree model using the training dataset and determining the right tree size required to produce the best possible final model using the validation dataset. Fig 1. shows the DT structure.

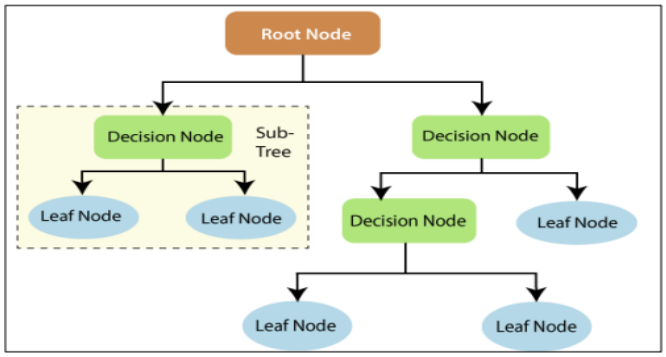

Fig. 1. Decision Tree algorithm

3.3.2. Random Forest algorithm

It is a popular supervised ensemble learning method that may be applied to regression or classification tasks. The technique computes the average results from all the decision trees in the group, which is made up of several independent groups of decision trees, and displays the average results as the final output. Because it is a grouping strategy and has unpredictability, which lowers the variance of the model, it performs better than individual models. The random forest approach is displayed in Fig 2 because of its simplicity and accuracy of results (Xin-She Yang,2019)and(Sebastian Raschka,2017).

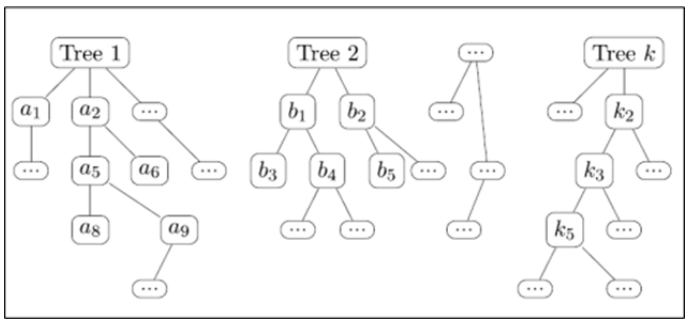

Fig. 2. Random Forest Algorithm

3.3.3. Bagging classifier

An ensemble technique is commonly referred to as Bootstrap Aggregation. Bootstrap Aggregating Ensemble Learning Bagging belongs to the category of group learning. It chooses a random subset of the whole set of data. By randomly selecting a portion of original training data and training a collection of decision trees, this method creates distinct models for each tree that are finished before training. The way it operates is to gather predictions from the trees and then determine a final forecast by voting for the majority of those who have similar expectations or by calculating the average of these choices (Burkov, Andriy,2019).

3.3.4. Voting classifier

Many classifiers continuously generate predictions and test result data in maximum voting method. By examining who received the most votes—more than half—the final prediction is

ascertained. This approach can be made more accurate by combining different classifiers (Q. Song et al,2010). The following is how the maximum voting classifier operates with Maximum:

1- Apply RF and ET classifiers to the training set.
2- Keep track of both classes' performances and create a comparison.

3.3.5. Stacking classifier

As one of the clustering techniques in machine learning, stacking has shown benefits in the competitive world (Mohammad Amimul Ihsan,2020). The basic idea behind this technique is to train a Meta classifier to help integrate the predictions of multiple learners using confidence scores as features in different model ensembles. Three classifiers were used in the study: Adaboot, decision tree, and Random Forest Classifiers. These classifiers use to test and train different types of slots.

3.3.6. Xgboost

One tool that is part of the distributed machine learning community (DMLC) is called xgboost, and it is well-known for its faster and more efficient gradient boosted decision tree performance. In order to deploy it in computer environments, eXtreme, also known as XGBoost, is utilized to help exploit all available hardware and memory resources, tweaking the model and improving the algorithm (YI PENG et al ,2011). XGBoost offers three different methods for gradient boosting: random, regular, and gradient boosting. It is also highly efficient in adding and changing regulation parameters, maximizing memory usage, and cutting down on the amount of time spent on computational tasks. Additionally, XGBoost can handle missing values and enable parallel structures when data is fed to the trained model.

3.3.7. Adaboosting

It is one of the ensemble learning techniques for supervised machine learning that is used to address classification and regression issues. Weight adjustments are used to train weak models into strong models. Specifically, models with bad performance are assigned larger weights than models with good performance. Basic models are trained on the data after the weights of the training samples are changed, which enhances the models' ability to produce accurate results. Models are trained in a sequential fashion, with the first model being trained first, and its expectations being utilized to teach the subsequent model. The procedure is repeated until the final model is reached and a final forecast is obtained (Xibin DONG et al,2020) The operation of Adaboosting is shown in Fig 3.

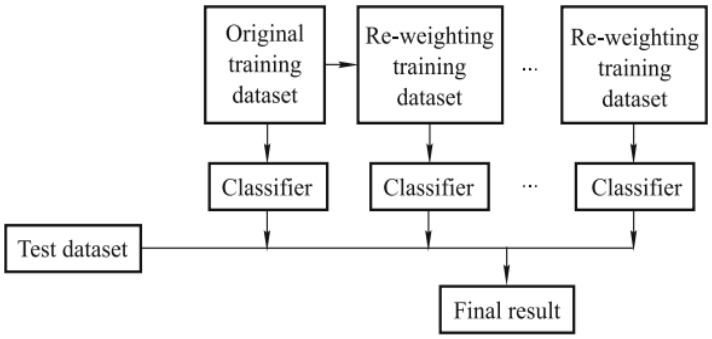

Fig. 3. Adaboosting the framework

3.4. Evaluation Criteria

They are measures used to measure performance of predictive models used in the following will mention the measures used to measure performance of proposed method:

The performance of ensemble machine learning models is evaluated using benchmarks. These metrics are the traditional quality standards used to evaluate the effectiveness of predictive models and used to measure the performance of our proposed method. Their definitions are given below:

1- 1- Accuracy is the percentage of total cases that the classifier correctly predicts, shown in equation (1) (Ernest Kwame Ampomah et al ,2020).

2- $$\boldsymbol{accuracy} = \frac{tp+tn}{tp+tn+fp+fn} \quad (1)$$

3- Precision is the percentage of positively predicted cases out of all positively predicted instances that the classifier correctly anticipated it is shown in equation (2)

$$\boldsymbol{precision} = \frac{tp}{tp+fp} \quad (2)$$

4- Recall: the percentage of positively correlated cases out of all positively correlated instances that the classifier correctly predicted it is shown in equation (3)

$$\boldsymbol{recall} = \frac{tp}{tp+fn} \quad (3)$$

5- The F1- score shows the harmonic mean of recall and precision it is shown in equation (4).

$$\boldsymbol{F1-score} = \frac{2*precision*recall}{precision+recall} \quad (4)$$

where fp stands for false positive, tn for true negative, tp for true positive, and fn for false negative.

4. RESULTS AND DISCUSSION

This section discusses the results that emerged when applying the proposed ensemble learning classifiers or models to predict bugs and software defects using 5 SOFTLAB datasets (ar1, ar3, ar4, ar5, ar6) after the processing operations mentioned in the previous section were completed by removing the columns that It contains the values, applies normalization to the data, and then

performs processing to balance the data through the oversampling process for the purpose of increasing performance accuracy. After that, the balanced data is divided into 20% for testing and 80% for training. . Then the ensemble model classifiers and base classifiers were applied, after which the predictive performance was evaluated through four performance metrics (accuracy, precision, recall, F1 score). The results showed high accuracy 99.2% in some data sets using the classifiers mentioned using different performance measures, including in Table4, when the five data set was used, and accuracy performance measure was applied to it using the Ensemble algorithms and the basic classifiers DT and RF, When implementing the (ar1) data set The results showed high accuracy ranging between 91.18 and 98.53, with Adaboost achieving 98.53, while(bagging and stacking) achieved an accuracy of 91, while (voting and xgboost) achieved an accuracy of 92. When implementing the ar3 data set, the accuracy ranged between (90 and 99), with voting achieving 99.3%, while (bagging and stacking) had lower accuracy, reaching 90.91%, while the accuracy of both (adaboost and xgboost) reached 95%, as shown in Table 5. When implementing the (ar4) data set, the results were not high compared to the rest of the other groups, as the accuracy ranged between (84 and 92). When implementing the various algorithms, the accuracy of Adaboost was the lowest, as it achieved 84%, while xgboost achieved 92%. As for (bagging, stacking, and voting), the accuracy was 90 % . The results showed the highest accuracy when implemented on the data set (ar5), as the accuracy reached 99.2% for the classifiers (adaboosting, stacking), while the results for the classifiers (Vote and xgboosting) were 95.56%, while the filling classifier was the lowest, achieving 91.11% as shown in Table 7**.**

Table 4 outputs of ar1 dataset

Evaluation Criteria	Algorithms				
	stacking	**Voting**	**adaboost**	**Xgboost**	**Bagging**
Accuracy	91.18%	92.65%	98.53%	92.65%	91.18%
Precision	91.84%	92.70%	98.58%	92.67%	91.84%
Recall	91.18%	92.65%	98.53%	92.68%	91.18%
F1-score	91.19%	92.66%	98.53%	92.64%	91.19%

Table 5 outputs of ar3 dataset

Evaluation Criteria	Algorithms				
	Stacking	**Voting**	**Adaboost**	**Xgboost**	**Bagging**
Accuracy	90.91%	99.3%	95.45%	95.45%	90.91%
Precision	92.42%	99.3%	95.87%	95.87%	92.42%
Recall	90.91%	99.3%	95.45%	95.45%	90.91%
F1-score	90.91%	99.3%	95.46%	95.46%	90.91%

Table 6 outputs of ar4 dataset

Evaluation Criteria	Algorithms				
	stacking	**Voting**	**Adaboost**	**Xgboost**	**bagging**
Accuracy	90.57%	90.57%	84.91%	92.45%	90.57%
Precision	91.11%	91.11%	85.13%	93.46%	91.11%
Recall	90.57%	90.57%	84.91%	92.45%	90.57%
F1-score	90.55%	90.55%	84.89%	92.42%	90.55%

Table 7 outputs of ar5 dataset

Evaluation Criteria	Algorithms				
	Stacking	**Voting**	**Adaboost**	**Xgboost**	**bagging**
Accuracy	99.2%	95.56%	99.2%	95.56%	91.11%
Precision	99.2%	95.56%	99.2%	95.56%	91.42%
Recall	99.2%	95.56%	99.2%	95.56%	91.11%
F1-score	99.2%	95.56%	99.2%	95.56%	91.08%

Table 8 outputs of ar6 dataset

Evaluation Criteria	Algorithms				
	stacking	**Voting**	**adaboost**	**Xgboost**	**bagging**
Accuracy	97.78%	97.78%	97.78%	97.78%	88.89%
Precision	97.87%	97.87%	97.87%	97.87%	89.56%
Recall	97.78%	97.78%	97.78%	97.78%	88.89%
F1-score	97.78%	97.78%	97.78%	97.78%	88.89%

The proposed model using ensemble learning clearly outperforms previous studies in anomaly prediction, as it benefits from combining the strengths of a set of base models to achieve higher accuracy and better generalization ability. As Table 9 shows, the improvement in the performance of the proposed model is manifested in better accuracy and balance rates compared to traditional models, which enhances its reliability and effectiveness in detecting abnormal patterns.

The table above shows the superiority of the proposed model using ensemble learning in anomaly prediction over other traditional methods and techniques .

5. CONCLUSION

Recently, there has been an increasing need to develop complex, high-quality software systems, and to improve the quality of software before delivering it to users, prediction of software defects or anomalies must be used. In this research paper, ensemble learning models (bagging, stacking, voting, Adaboost, Xgboost) were used for the prediction process. The method was to use the Softlab data set, and after the data processing process was completed, including data balancing and the data oversampling process, then the data was divided into training data and test data of 80% and 20%, and to evaluate the prediction performance. Four performance measures were used (accuracy,precision,recall,F1-score) where the results of the method showed the superiority of the ensemble model (stacking ,adaboost, voting) as it obtained a

prediction accuracy of to (99.2%) compared to the previous studies shown in the table and it had the best predictive performance.

In future work, the work can be applied to more than one data set. Deep learning models can also be combined with the ensemble model.

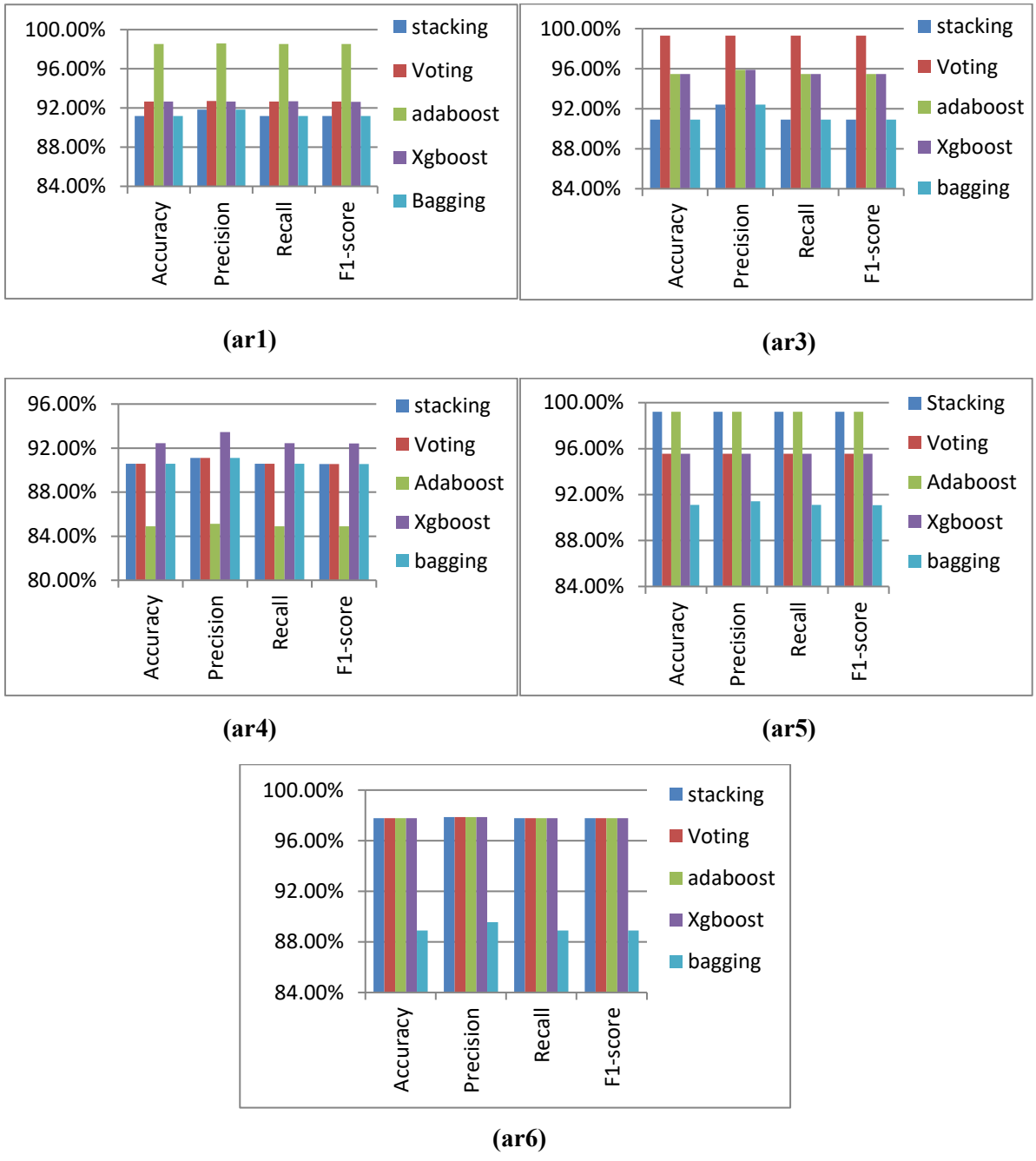

Fig. 4. shows the results of the dataset (ar1,ar3,ar4,ar5,ar6) when implementing the proposed approach using ensemble learning.

Table 9 Comparison of The Performance of The Proposed Model With Other Techniques of Previous Studies

dataset	Evaluation Criteria	Proposed Model Ensemble learning					(Ren And Liu,2019)	(Elife Ozturk et al,2021)	
		Adaboost	Xgboost	bagging	stacking	Voteing	SODM	KNN	MVKNN
ar1	Accuracy	98.53%	92.65%	91.18%	91.18%	92.65%	0.975	-	-
	Precision	98.58%	92.67%	91.84%	91.84%	92.70%	0.214	0.86	0.86
	Recall	98.53%	92.68%	91.18%	91.18%	92.65%	0.67	0.93	0.93
	F1-Score	98.53%	92.64%	91.19%	91.19%	92.66%	0.324	0.89	0.89
ar3	Accuracy	95.45%	95.45%	90.91%	90.91%	99.3%	-	-	-
	Precision	95.87%	95.87%	92.42%	92.42%	99.3%	-	0.89	0.89
	Recall	95.45%	95.45%	90.91%	90.91%	99.3%	-	0.90	0.90
	F1-Score	95.46%	95.46%	90.91%	90.91%	99.3%	-	0.89	0.89
ar4	Accuracy	84.91%	92.45%	90.57%	90.57%	90.57%	0.84	-	-
	Precision	85.13%	93.46%	91.11%	91.11%	91.11%	0.548	0.82	0.85
	Recall	84.91%	92.45%	90.57%	90.57%	90.57%	0.85	0.84	0.86
	F1-Score	84.89%	92.42%	90.55%	90.55%	90.55%	0.667	0.83	0.85
ar5	Accuracy	99.2%	95.56%	91.11%	99.2%	95.56%	0.914	-	-
	Precision	99.2%	95.56%	91.42%	99.2%	95.56%	0.778	0.76	0.85
	Recall	99.2%	95.56%	91.11%	99.2%	95.56%	0.875	0.80	0.83
	F1-Score	99.2%	95.56%	91.08%	99.2%	95.56%	0.824	0.78	0.84
ar6	Accuracy	97.78%	97.78%	88.89%	97.78%	97.78%	0.842	-	-
	Precision	97.87%	97.87%	89.56%	97.87%	97.87%	0.478	0.85	0.85
	Recall	97.78%	97.78%	88.89%	97.78%	97.78%	0.733	0.79	0.79
	F1-Score	97.78%	97.78%	88.89%	97.87%	97.87%	0.579	0.73	0.73

6. REFERENCES

Abbas, S. et al ,(2023), “Data and Ensemble Machine Learning Fusion Based Intelligent Software Defect Prediction System”, http://dx.doi.org/10.32604/cmc.2023.037933

Abdou, A. S, and Darwish , N. R, (2018),“Early Prediction of Software Defect using Ensemble Learning: A Comparative Study”, International Journal of Computer Applications (0975 – 8887) Volume 179 – No.46.

AL-FRAIHAT, D. et al, (2024), “Hyperparameter Optimization for Software Bug Prediction Using Ensemble Learning”, Digital Object Identifier 10.1109/ACCESS.2024.3380024

Ali , U. et al, “Software Defect Prediction Using Variant based Ensemble Learning and Feature Selection Techniques”, Published Online October (2020) in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijmecs.2020.05.03

Balogun, A. et al , (2018), “Software Defect Prediction Using Ensemble Learning: An ANP Based Evaluation Method”, http://dx.doi.org/10.46792/fuoyejet.v3i2.200

Balogun, A.. et al , (2020), “SMOTE-Based Homogeneous Ensemble Methods for Software Defect Prediction”, O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12254, pp. 615–631. https://doi.org/10.1007/978-3-030-58817-5_45

Burkov, Andriy. (2019). “The hundred-page machine learning” book. Vol. 1. Quebec City, QC, Canada: Andriy Burkov, https://doi.org/10.1080/15228053.2020.1766224

Elife Ozturk et al, (2021), "Multi-view learning for software defect prediction" , e-Informatica Software Engineering Journal , Volume 15, Issue 1.

Ernest Kwame Ampomah et al , (2020),"Evaluation of Tree-Based Ensemble Machine Learning Models in Predicting Stock Price Direction of Movement", http://dx.doi.org/10.3390/info11060332

Hersh A. Mohammed et al, (2020), "A COMPARATIVE EVALUATION OF DEEP LEARNING METHODS IN DIGITAL IMAGE CLASSIFICATION" , Kufa Journal of Engineering Vol. 13, No. 4, October 2022, P.P. 53-69

Huda, S. et al , (2021),"An ensemble oversampling model for class imbalance problem in software defect prediction", Citation information: DOI 10.1109/ACCESS.2018.2817572

Jiayao Chen et al, (2021),"Machine learning-based classification of rock discontinuity trace: SMOTE oversampling integrated with GBT ensemble learning", https://doi.org/10.1016/j.ijmst.2021.08.004

Jun-hai Zhai et al, (2012) ,"Dynamic ensemble extreme learning machine based on sample entropy", Soft Comput 16:1493–1502,

Khuat, T. T. and Le , M. , (2019), "Ensemble learning for software fault prediction problem with imbalanced data", Vol. 9, No. 4, pp. 3241~3246 ISSN: 2088-8708, DOI: 10.11591/ijece.v9i4.pp3241-3246

Matloob, F. et al , (2019),"A Framework for Software Defect Prediction Using Feature Selection and Ensemble Learning Techniques", Published Online December 2019 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijmecs 12.02.

MISBAH ALI et al , (2024),"Software Defect Prediction Using an Intelligent Ensemble-Based Model", VOLUME 12, https://creativecommons.org/licenses/by-nc-nd/4.0/

Mohammad Amimul Ihsan Aquil, Wan Hussain Wan Ishak, (2020), "Predicting Software Defects using Machine Learning Techniques", https://doi.org/10.30534/ijatcse/2020/352942020

Monni, C. , and Pezze`, M. , (2019), "Energy-Based Anomaly Detection A New Perspective for Predicting Software Failures", http://dx.doi.org/10.13140/RG.2.2.29124.88967

Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu. (2010),A general software defect-proneness prediction framework. IEEE transactions on software engineering, 37(3), 356-370 ,

R. Mamatha et al, (2023),“Enhanced Software Defect Prediction Through Homogeneous Ensemble Models .”

REN AND LIU, (2019), “Predicting Software Defects Using Self-Organizing Data Mining” , Digital Object Identifier 10.1109/ACCESS.2019.2927489.

Sagheer Abbas et al, (2023), “Bata and Ensemble Machine Learning Fusion Based Intelligent Software Defect Prediction System”, http://dx.doi.org/10.32604/cmc.2023.037933

Saheed , Y. K. et al, (2021),“An Ensemble Learning Approach for Software Defect Prediction in Developing Quality Software Product”, https://www.researchgate.net/publication/355490443_An_Ensemble_Learning_Approach_for_Software_Defect_Prediction_in_Developing_Quality_Software_Product?enrichId=rgreq-732adb78eca05ef05a16767ff9755563-XXX&enrichSource=Y292ZXJQYWdlOzM1NTQ5MDQ0MztBUzoxMTA0MDg4MjMwMDQzNjQ4QDE2NDAyNDY1NTUwMDc%3D&el=1_x_2&_esc=publicationCoverPdf

Sebastian Raschka, (2017) ,“Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow”,[book], Copyright © Packt Publishing

Sharm, T. et al, (2023),“Ensemble Machine Learning Paradigms in Software Defect Prediction”, Available online at www.sciencedirect.com Procedia Computer Science 218 199–209

Shi ,T. , Zou , Z. and Ai , J., (2023),“Software Operation Anomalies Diagnosis Method Based on a Multiple Time Windows Mixed Model”, Appl. Sci. 13, 11349. https://doi.org/10.3390/app132011349

Shuo Feng, et al. (2021),“Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction”. https://doi.org/10.1016/j.infsof.2021.106662

Sivalingan H , (2024),“Cloud-Smart Surveillance: Enhancing Anomaly Detection In Video Streams With DfConvlstm-Based Vae-Ga" , Kufa Journal of Engineering Vol. 15, No. 4, October 2024, P.P. 125-140.

Xibin DONG et al , (2020),“A survey on ensemble learning”, Front. Comput. Sci., 14(2): 241–258 https://doi.org/10.1007/s11704-019-8208-z

Xin, J. et al, (2023),“Ensemble learning based defect detection of laser sintering”, https://doi.org/10.1049/ote2.12108

Xin-She Yang, (2019),"Introduction to Algorithms for Data Mining and Machine Learning" [book], https://www.elsevier.com/books-and-journals

Yan-yan SONGand Ying LU, (2015),"Decision tree methods: applications for classification and prediction", Shanghai Archives of Psychiatry, Vol. 27, No. 2 http://dx.doi.org/10.11919/j.issn.1002-0829.215044

YI PENG et al , (2011) ,"ENSEMBLE OF SOFTWARE DEFECT PREDICTORS: AN AHP-BASED EVALUATION METHOD", International Journal of Information Technology & Decision Making Vol. 10, No. 1 187–206 .

Zhao, Z. , (2017), "Ensemble Methods for Anomaly Detection", Dissertations - ALL. 817. https://surface.syr.edu/etd/817