Innovations in Systems and Software Engineering (2022) 18:301–308
https://doi.org/10.1007/s11334-021-00390-x

S.I. : ACITSEP

A sequential ensemble model for software fault prediction

Monika Mangla[1] · **Nonita Sharma**[2] · **Sachi Nandan Mohanty**[3]

Received: 10 September 2020 / Accepted: 11 November 2020 / Published online: 28 March 2021

Abstract
Unlike several other engineering disciplines, software engineering lacks well-defined research strategies. However, with the exponential rise in automation, the demand for software has observed an enormous elevation. Simultaneously, it necessitates having zero failures in the software modules to maximize the availability and optimize the maintenance cost. This has attracted many researchers to try their hand in formalizing the strategies for testing of software. Numerous researchers have suggested various models in this context. The authors in this paper present a sequential ensemble model to predict software faults. The employment of ensemble modeling in software fault prediction is motivated by its competence in various domains. The proposed model is also implemented on the 8 datasets taken from PROMISE and ECLIPSE repository. The proposed model's performance is evaluated using various error metrics, viz. average absolute error, average relative error, and prediction. The obtained results are encouraging and thus establish the competence of the proposed model.

Keywords Ensemble modeling · Software fault prediction · Normalization · Fault proneness · Software module

1 Introduction

Various researchers generally have well-defined research strategies that not only have detailed guidance but also possess simplified views from observers' perspectives. For instance, the public can understand large-scale medical studies well enough so as to discuss the risks associated with an experimental treatment. However, this is not true for software engineering researchers as they do not have any well-understandable guidance [1]. Various researchers have made several attempts to formalize software engineering research, but it fails to paint a comprehensive picture [2]. As a result, rigorous research is taking place in this dimension to fill the void. Authors in this paper attempt to formalize the method of software fault prediction through a sequential ensemble model [3, 4].

The motive for selecting this topic for research is that the prime reason for software failures is the faults present in software modules affecting the software's reliability. It may lead to dissatisfaction among users, eventually leading to the downfall of the company. However, in the current scenario, when software demand is exponentially increasing in the industries, there is nearly a zero-tolerance for software faults. Additionally, the software has a considerable number of modules that further intricate the identification of fault-prone modules. This has further opened avenues for research in the field of software engineering, particularly in the field of software fault prediction (SFP) [5, 6].

SFP aims to thoroughly inspect the software's quality before its release by inspecting the fault vulnerability of the software modules [7]. Identification of fault vulnerable modules emphasizes a specific focus on these modules to efficiently manage the resources by reducing the number of faults post-implementation [8]. SFP focuses on accurately predicting the fault vulnerability of the software modules so as to maximize software availability. Moreover, it also helps to minimize the maintenance cost and thus achieves high-quality software products [9].

Machine Learning (ML) has demonstrated successful and widespread deployment for solving classification problems of SFP [10]. Here, classification problem refers to classifying Fault-Prone (FP) and Non-Fault-Prone (NFP) modules. Now, this classification problem in SFP has several associated challenges: class imbalance

✉ Nonita Sharma
nonita@nitj.ac.in

[1] Lokmanya Tilak College of Engineering, Navi Mumbai, India

[2] Dr. B.R. Ambedkar National Institute of Technology Jalandhar, Jalandhar, India

[3] College of Engineering Pune, Pune, India

problems, irrelevant features, and noise [11]. Hence, a single ML technique fails to handle all these challenges and thus leads to performance degradation. Hence, it is widely accepted that ensemble modeling may overcome the limitations that remain unaddressed by individual ML classifier [6, 12]. This belief is strengthened by the proven competence of ensemble learning algorithms (ELA) in various research fields [10, 13]. In the literature, it is also claimed that ensemble classifiers overcome the limitations of individual classifiers. Moreover, no learning technique can handle SFP's significant challenges like imbalance problems, the presence of redundant features, and noise in the dataset [14, 15]. All these applications advocated that the application of ELA in SFP as it outperforms the individual classifier algorithm [6, 16].

The class imbalance problem occurs when there is an extreme imbalance between Fault-Prone (FP) and Non-Fault-Prone (NFP) modules. Hence, in this scenario, the dataset is highly skewed toward FP or NFP modules. Generally, FP modules are relatively small and rarely occur but have considerable significance. However, learners primarily focus on NFP modules while ignoring the FP modules. Here, data balancing is implemented to resolve the skewness in the dataset to improve the performance of ELA for SFP models. In the context of class imbalance, ML researchers have suggested two methods to handle this. As per a suggested method, a typical cost is to be assigned to training examples. The other method suggests resampling the original dataset by oversampling the minority class and under-sampling the majority class [14]. Hence, various studies suggest Synthetic Minority Oversampling Techniques (SMOTE) for balancing to enhance the performance of classification [17].

Also, it is evident that the performance of classification learner is affected by the quality of data used [13, 18]. Resultantly, corrupted data in real datasets may impede the decisions, and thus, the ensemble classifiers built from such data lack accuracy. A specifically designed ensemble-based framework may be helpful in this case, and hence, authors propose a framework that combines ELA with noise filtering, feature selection, and data balancing. Here, feature selection eliminates the redundant and less significant features to consider only principal features for training. Consideration of principal features aids in reducing the complexity of the algorithm, thus achieving speed and cost-effectiveness. Here, in order to eliminate the less useful features, it employs the Information Gain approach, which has been widely accepted in various studies related to SFP [19].

Hence, the authors in this paper aim to study the large scale experiment to understand the effect of ensemble modeling in SFP. The paper also presents a thorough comparison of individual techniques at each stage. Thereafter, a new sequential ensemble model is presented that aims to address the challenges of the individual model.

The related work by various researchers demonstrates that ELA achieves robustness. Summarizing this, when ELA is implemented on selected features of balanced data, it achieves remarkable performance improvement. Thus, it needs to examine the ensemble techniques to achieve robust performance. The objective of the study is to identify the FP modules efficiently.

The work is organized into various sections. The requirement for a formalized method for SFP is discussed in Sect. 1. Materials and methods are presented in Sect. 2. The proposed sequential ensemble model is elaborated in Sect. 3. The results obtained from the proposed model are discussed in Sect. 4. Finally, the manuscript is concluded in Sect. 5.

2 Materials and methods

The ensemble model contains a learning prototype where multiple models are trained on the same datasets, and the forecast of these models are combined to forecast the future values. The ensemble approach gains performance enhancement over individual models owing to reduced bias and variance. Ensembling involves training multiple (same or different) models individually, which further combines their results. These results are combined by feeding into a meta-model that uses individual models' values to predict the final value. The ensemble modeling may be broadly categorized into sequential model and parallel model, which are defined as follows:

2.1 Sequential

In this method, base learning models are dependent on each other. A classic example of a sequential model is AdaBoost.

2.2 Parallel

In this model, base learning models are independent of each other such as Random Forest.

This broad classification of the ensemble model is shown in Fig. 1a, b.

The following section discusses the material and method used for this research work.

2.3 Data collection

In this manuscript, authors have selected 8 fault datasets for experimental analysis. These datasets have been taken from various open-source projects present in PROMISE and Eclipse bug data repository [20]. Here, PROMISE is an open-source repository that contains fault datasets for various open-source software. Eclipse repository contains fault information of the Eclipse project that contains thousands of

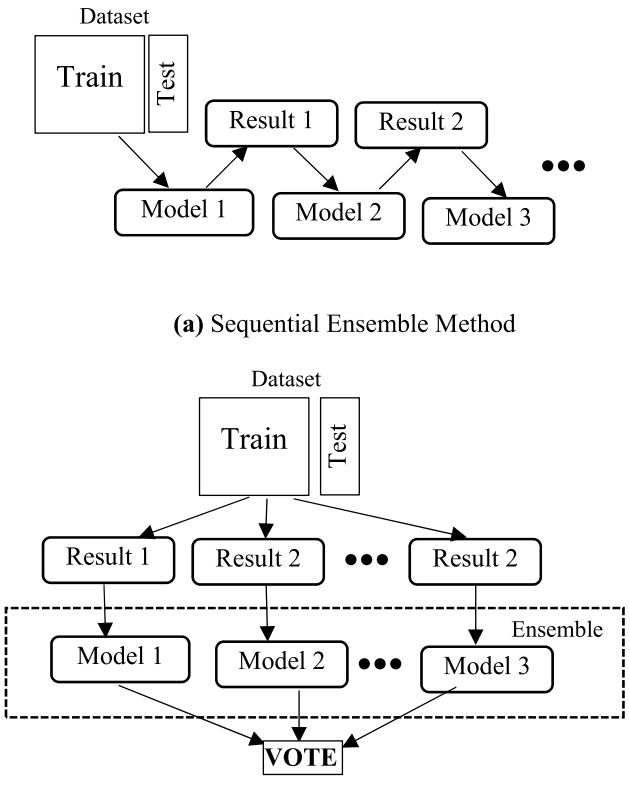

Fig. 1 **a** Sequential Ensemble Method. **b** Parallel ensemble methods

files. This project is similar to an industrial system in terms of size and complexity. The dataset contains various metrics like structure and complexity at the file-level. The dataset contains "filename," "count of errors reported six months before release," "count of errors reported six months after the release," and "complexity metrics." Usage of standard datasets enables the reproduction of the standard experiments that aids in performing comparative performance analysis. Here, we have taken the datasets with more than 300 modules to check the proposed model's efficiency. The datasets are represented in Table 1.

3 Proposed sequential ensemble model

The proposed model is presented abstractly in Fig. 2. Here, the output of each model is given to Neural Network Autoregression (NNAR). NNAR model is run multiple times so as to achieve the best-fitting model with the least error [21, 22]. Thereafter, the fitted values from the NNAR model are fed to support vector regression (SVR) model.

In the proposed model, authors suggest tuning the hyperparameters of SVR to obtain an accurate prediction. The

Table 1 Dataset for the proposed model of SFP

Dataset	Repository	Modules	KLOC	Modules with fault	% Distribution
PROP V121	PROMISE	2998	628	426	16.5
XERCES 1.4	PROMISE	589	141	438	74.3
CAMEL 1.6	PROMISE	966	113	189	19.5
ANT 1.7	PROMISE	746	208	167	22.3
XALAN 2.6	PROMISE	886	411	412	46.5
EMF 2.0	ECLIPSE	6730	796	976	14.5
EMF 2.1	ECLIPSE	7889	987	855	10.8
EMF 3.0	ECLIPSE	10,594	1305	1569	14.8

model mainly considers three hyperparameters, viz. soft margin constant cost, the linearity degree of the hyperplane (γ), and finally, the error tolerance (ϵ). These parameters refer to the misclassification of the training data. Here, the model picks a small margin hyperplane for larger values to enhance the said model's performance in terms of classification. On the contrary, it picks a larger margin hyperplane for smaller values. The model derived from a line function is mathematically defined by the following equation:

$$m.a + n = 0 \tag{1}$$

here,

$$m = \sum_{i=1}^{k} \Psi_i b_i a_i \tag{2}$$

$$n = \frac{1}{s} \sum_{i=1}^{s} \left(b_i - m.a\right) \tag{3}$$

s indicates the count of support vectors.

In the model C can be mathematically described as:

$$C = \underset{m,n,\xi}{min} \frac{1}{2}||w||^2 + C\sum_{i=1}^{p} \xi_i \tag{4}$$

where

$$b_i\left(m.a_i + n\right) \geq 1 - \xi_i \tag{5}$$

$$\xi_i \geq 0 \tag{6}$$

$$i = 1..p \tag{7}$$

These parameters indicate the impact of a single training example considering minimal gamma (distant) and maximum gamma (near). Here, gamma can be understood as the inverse of the radius of influence, and thus, a large

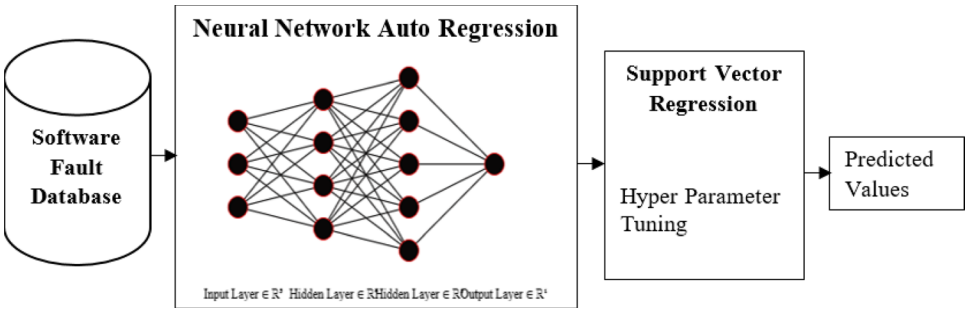

Fig. 2 Abstract view of proposed model

gamma indicates that the radius of the area of influence of support vectors includes only support-vector. In such a case, normalization fails to avoid overfitting. Conversely, smaller gamma indicates the overly contrived model that fails to retain the complexity of data. It can be mathematically expressed as follows:

$$R(a_i, a_j) = exp\left(-\gamma \left|\left|a_i - a_j\right|\right|^2\right) \tag{8}$$

An extensive detail of the proposed model is given in Fig. 3. Here, the historical data are segregated into training data and testing data. In the model, the actual number of faults is given to NNAR individually. This NNAR gives the fitted values which are input to the SVR model. Finally, the predicted values are obtained through the SVR model. In the end, the performance of the model is established in terms of several error metrics [23].

3.1 Implementation steps

Various steps in the proposed ensemble model for SFP are depicted in Fig. 4.

1. As mentioned earlier, the proposed model considers 8 well-known fault datasets from PROMISE and Eclipse bug data repository [20].
2. Further, if the dataset consists of zero to indicate missing values, it is suggested to take z-score transformation as log transformation may again lead to undefined values. However, log transformation may be done for the rest of the datasets. The equations for log and z-score transformation are given in Eq. 9 and 10, respectively. These transformations basically help in minimizing the variances among the dataset.

$$\hat{a}_t = log(a_t) \tag{9}$$

$$\hat{a}_t = \frac{a_t - mean(a_t)}{SD_t} \tag{10}$$

where

$$mean(a_t) = \frac{1}{p}\sum_{t=1}^{p} a_t \tag{11}$$

$$SD_t = \frac{1}{p}\sum_{t=1}^{p} \sqrt{a_t - \hat{a}_t} \tag{12}$$

3. Thereafter, the transformed dataset is classified into training dataset and test dataset.
4. The training dataset part after the classification is given to the model for learning purposes. During learning, the model extracts useful patterns and information in the data. The feeding of actual data follows it into the NNAR model. The equation for the same is expressed as follows:

$$\hat{b}_t = f(\hat{a}_t) + \varepsilon_t \tag{13}$$

Then, fitted values of NNAR model are given as input to the SVR model which can be mathematically expressed as follows:

$$g(c_t) = (w.\Psi_t(b_t)) + C \tag{14}$$

Various parameters that have been used in above equations are described as follows in Table 2:

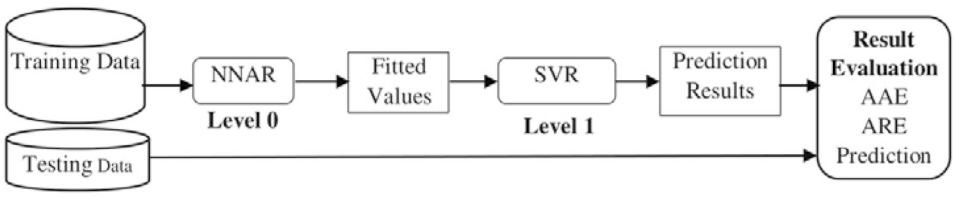

Fig. 3 Detailed view of sequential ensemble model

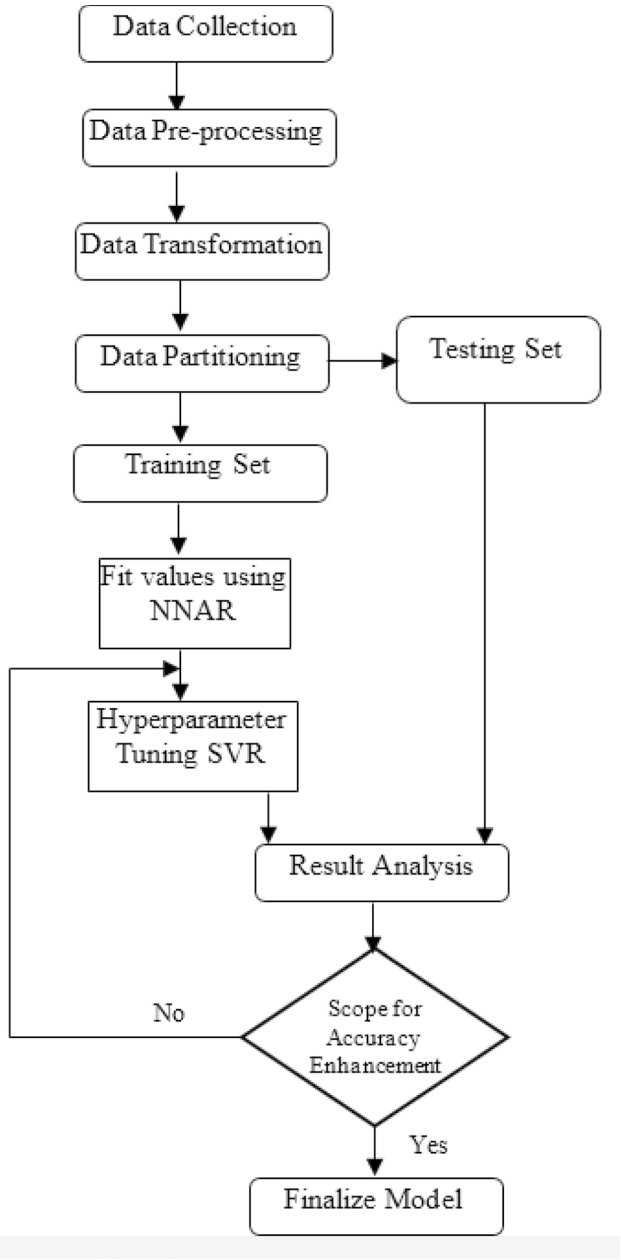

Fig. 4 Step-wise illustration of the proposed ensemble model

The parameters are manually tuned in the model by fixing a parameter to its default value, and the other parameter is adjusted accordingly. During each adjustment, the accuracy of the model is checked regularly. Now, when some parameter achieves a required value, other parameters are adjusted accordingly while checking the model accuracy. Thus, the values of all parameters are manually adjusted. The NNAR model is executed multiple times with different seed value each time to make autoregression, and seasonal autoregression equal to zero that helps avoid missing values in the fitted result of NNAR. SVR model is also set to default and modified so as optimize its value. This process of fixing some parameter's value and adjusting other parameters in order to find optimum value is called cross-validation.

5. The trained model's performance can be measured in terms of various error metrics for the test dataset. The accuracy of the prediction of the proposed model is also verified by various error metrics as described in the subsequent section.

Table 2 Descriptions of used parameters

Parameters	Descriptions
a_t	Actual value at time t
$\hat{a}_t$	Predicted value at time t
p	Total number of values
T	Time period
k	Number of lags
b_t	Regression value of NNAR at time t
$f(a_t)$	Vector with lagged values of series
ϵ_t	Error series (homoscedastic)
$g(c_t)$	SVR function
w	Direction of vector
Ψ_t	Kernel Function

4 Results and discussions

In order to find the efficacy of the proposed approach, experiments are conducted with reference to the collected datasets and other ensemble techniques. The section is divided into two subsections: In the first subsection, an empirical evaluation of the proposed ensemble learning approach is done using two scenarios, viz. Intrarelease prediction and Intrarelease prediction. Secondly, a comparative analysis is performed for proposed technique with other ensemble techniques. The datasets are classified into two subsets as training dataset (80%) and testing datasets (20%). The obtained results are analyzed with respect to several performance metrics, viz. Average Absolute Error (AAE), Average Relative Error (ARE), and prediction (level 1).

Here, AAE is represented in Eq. (15) that represents the absolute difference between the predicted faults (PF) and the actual faults (AF) for n number of modules:

$$AAE = \frac{1}{n}\sum_{i=1}^{n}\left|AF_i - PF_i\right| \tag{15}$$

ARE is given in Eq. (16) which represents the proportion of absolute error with respect to the average fault.

$$ARE = \frac{1}{n}\sum_{i=1}^{n}\frac{|AF_i - PF_i|}{AF_i + 1} \quad (16)$$

Further, another measure, i.e., prediction at level l, represents the percentage of faults predicted that lies within the range of 1% considering the actual faults. In order to consider the model as acceptable, the value of this metric must be kept as 1 less than or equal to 0.3 [24].

4.1 Empirical evaluation

In Intrarelease prediction, only a single version of the software is used to collect the datasets, Interrelease prediction where several versions of the same software are used to derive the datasets. The results obtained are illustrated below:

4.1.1 Intrarelease prediction

The results obtained are demonstrated in Table 3. The value of AAE measure remains between 0.13 and 1.78. CAMEL 1.6 and XALAN 2.6 demonstrate the highest values. Eclipse datasets have shown the lowest values. Most of the datasets demonstrate the values lesser than 0.50. However, the range of ARE values remains between 0.12 and 0.46, and most of the values were lying below 0.31 except XERES 1.4. The results regarding prediction (level 1) range between 45 and 90%. With the highest value for EMF 2.1. Xerces 1.4 and XALAN 2.6 show lower values contrasted with other datasets. The average values of the proposed ensemble approach metrics are 0.51, 0.21, and 72.14% for AAE, ARE, and pred(0.3) analysis, respectively. The visualization of the obtained results is depicted in Fig. 5.

Table 3 Error metrics of ensemble approach for Intrarelease prediction

Datasets	Avg. absolute error	Avg. relative error	Prediction (30%) (%)
PROP V121	0.24	0.13	82
XERCES 1.4	1.78	0.46	47.33
CAMEL 1.6	0.65	0.31	65.68
ANT 1.7	0.35	0.25	69.31
XALAN 2.6	0.50	0.28	59.54
EMF 2.0	0.23	0.12	84.27
EMF 2.1	0.13	0.065	88.2
EMF 3.0	0.23	0.13	80.85

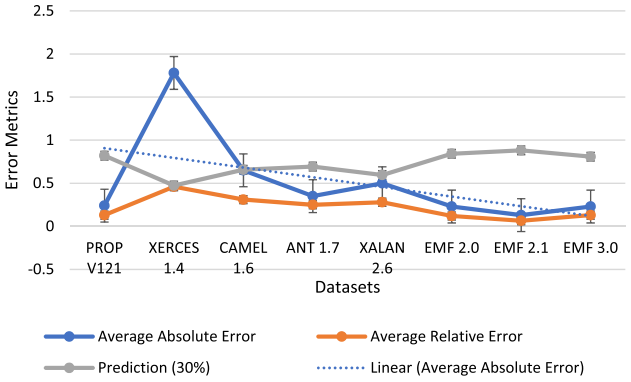

Fig. 5 Illustration of obtained results for intra release predictions

4.1.2 Interrelease prediction

In Interrelease prediction, the software's current version is used as testing datasets, whereas the previous versions of the software are used as a training set. The simulation results are provided in Table 4. The value of AAE metrics range from 0.2 to 2.0; ARE metrics range from 0.12 to 0.44. Eclipse datasets have shown promising results with the lowest AAE

Table 4 Error metrics of ensemble approach for Interrelease prediction

Datasets	Avg. absolute error	Avg. relative error	Prediction (30%) (%)
PROP V121	0.42	0.26	74.21
XERCES 1.4	2.0	0.44	32.66
CAMEL 1.6	0.69	0.39	63.48
ANT 1.7	0.55	0.38	63.58
XALAN 2.6	0.53	0.32	51.58
EMF 2.0	0.26	0.15	83.97
EMF 2.1	0.24	0.12	81.25
EMF 3.0	0.2	0.13	81.85

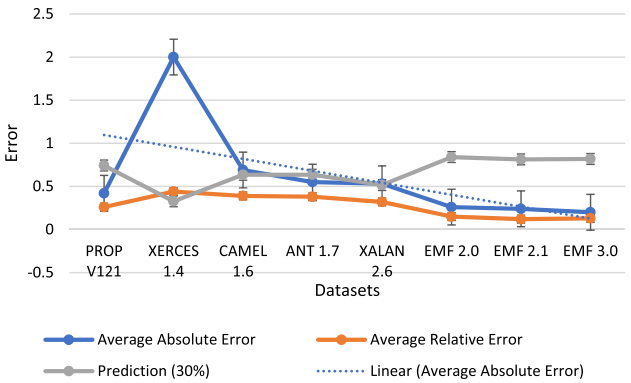

Fig. 6 Illustration of obtained results for inter release predictions

Table 5 Comparative analysis for Intrarelease prediction

Techniques	AAE			ARE		
	Min	Max	SD	Min	Max	SD
Random Forest	0.141	2.27	0.46 ± 0.09	0.07	0.44	0.20 ± 0.08
Bagging	0.136	1.69	0.43 ± 0.08	0.08	0.66	0.27 ± 0.04
Stacking	0.158	2.28	0.57 ± 0.11	0.09	0.6	0.28 ± 0.04
XGBoost	0.147	2.25	0.45 ± 0.08	0.09	0.75	0.29 ± 0.05
Ensemble	0.13	0.44	0.31 ± 0.083	0.16	0.91	0.49 ± 0.87

Table 6 Comparative analysis for Interrelease prediction

Techniques	AAE			ARE		
	Min	Max	SD	Min	Max	SD
Random Forest	0.2	2.02	0.56 ± 0.14	0.07	0.44	0.20 ± 0.08
Bagging	0.21	2.48	0.83 ± 0.16	0.06	0.46	0.27 ± 0.02
Stacking	0.19	2.46	0.60 ± 0.17	0.05	0.56	0.28 ± 0.02
XGBoost	0.201	2.25	0.45 ± 0.08	0.06	0.63	0.23 ± 0.03
Ensemble	0.2	1.78	0.41 ± 0.083	0.12	0.44	0.24 ± 0.11

and ARE values; on the contrary, Xerces 1.4 dataset has the highest values. The average value of the AAE metric is 0.61, and all datasets have lesser than the average value except XERES 1.4. Correspondingly, the average value for the ARE metric is 0.27. Regarding the pred(0.3) measure, the values range between 35 and 85%, with the average being 66.56%. Figure 6 shows the visualization of the results.

4.2 Comparative analysis

The proposed ensemble approach's performance has been compared with the state of art ensemble techniques, viz., random forest, bagging, stacking, and XGBoost. The simulation results obtained along with the comparative analysis are demonstrated in Tables 5 and 6.

The obtained results, as shown in Tables 5 and 6, witness that the ensemble modeling outperforms the individual model by a significant margin. Thus, the competence of ensemble modeling for SFP is established in addition to various other domains.

5 Conclusion

The field of software engineering has lacked well-defined strategies, unlike other related fields. However, during the past few decades, there has been a significant rise in software demand. This growth of the software industry also necessitates the presence of well-defined strategies. As a result, rigorous research is taking place in this direction. The authors in this manuscript aim to formalize the method of SFP through a sequential ensemble model. The proposed model is applied on the 8 datasets taken from well-known repositories. The proposed sequential ensemble model's performance is analyzed in terms of various error metrics, viz. average absolute error, average relative error, and prediction. Root mean squared error (RMSE), another error metric, is not employed for analyzing the performance as RMSE assigns larger weights to larger error, as it squares the errors before averaging out, and hence, it is more suitable for applications focusing on large errors. This means the RMSE should be more useful when large errors are particularly undesirable. The results obtained through the proposed model are encouraging and thus support the employment of ensemble modeling for SFP.

References

1. Glass RL, Vessey I, Ramesh V (2002) Research in software engineering: an analysis of the literature. Inf Softw Technol 44(8):491–506
2. Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14(2):131
3. Malhotra R, Singh Y (2011) On the applicability of machine learning techniques for object oriented software fault prediction. Softw Eng Int J 1(1):24–37
4. Sultana N, Sharma N, Sharma KP, Verma S (2020) A Sequential ensemble model for communicable disease forecasting. Curr Bioinform 15(4):309–317
5. Sherer SA (1995) Software fault prediction. J Syst Softw 29(2):97–105
6. Rathore SS, Kumar S (2017) Towards an ensemble based system for predicting the number of software faults. Expert Syst Appl 82:357–382

7. Rathore SS, Kumar S (2018) An approach for the prediction of number of software faults based on the dynamic selection of learning techniques. IEEE Trans Reliab 68(1):216–236
8. Rathore SS, Kumar S (2017) An empirical study of some software fault prediction techniques for the number of faults prediction. Soft Comput 21(24):7417–7434
9. Saliu O, Ruhe G (2005) Software release planning for evolving systems. Innov Syst Softw Eng 1(2):189–204
10. Hall T, Bowes D (2012, December) The state of machine learning methodology in software fault prediction. In: 2012 11th international conference on machine learning and applications, Vol 2, pp 308–313. IEEE.
11. Shatnawi R (2017) The application of ROC analysis in threshold identification, data imbalance and metrics selection for software fault prediction. Innov Syst Softw Eng 13(2–3):201–217
12. Dejaeger K, Verbraken T, Baesens B (2012) Toward comprehensible software fault prediction models using bayesian network classifiers. IEEE Trans Softw Eng 39(2):237–257
13. Sharma D, Chandra P (2018) Software fault prediction using machine-learning techniques. In: Smart computing and informatics, pp 541–549. Springer, Singapore.
14. Kalsoom A, Maqsood M, Ghazanfar MA, Aadil F, Rho S (2018) A dimensionality reduction-based efficient software fault prediction using Fisher linear discriminant analysis (FLDA). J Supercomput 74(9):4568–4602
15. Malhotra R, Jain A (2012) Fault prediction using statistical and machine learning methods for improving software quality. J Inf Process Syst 8(2):241–262
16. Gondra I (2008) Applying machine learning to software fault-proneness prediction. J Syst Softw 81(2):186–195
17. Alsawalqah H, Faris H, Aljarah I, Alnemer L, Alhindawi N (2017, April). Hybrid SMOTE-ensemble approach for software defect prediction. In: Computer science on-line conference, pp 355–366. Springer, Cham.
18. Karim S, Warnars HLHS, Gaol FL, Abdurachman E, Soewito B (2017, November) Software metrics for fault prediction using machine learning approaches: a literature review with PROMISE repository dataset. In: 2017 IEEE international conference on cybernetics and computational intelligence (CyberneticsCom), pp 19–23. IEEE.
19. Yohannese CW, Li T (2017) A combined-learning based framework for improved software fault prediction. Int J Comput Intell Syst 10(1):647–662
20. Bal PR, Kumar S (2018) Cross project software defect prediction using extreme learning machine: an ensemble based study. In: ICSOFT, pp 354–361.
21. Malhotra R (2015) A systematic review of machine learning techniques for software fault prediction. Appl Soft Comput 27:504–518
22. Rathore SS, Kumar S (2016) A decision tree regression based approach for the number of software faults prediction. ACM SIGSOFT Softw Eng Notes 41(1):1–6
23. Catal C (2012) Performance evaluation metrics for software fault prediction studies. Acta Polytechnica Hungarica 9(4):193–206
24. MacDonell SG (1997) Establishing relationships between specification size and software process effort in case environments. Inf Softw Technol 39(1):35–45