Journal of Innovative Engineering and Research (JIER)
Vol.- 5, Issue -2, October 2022, pp. 21-24 (4 pages)

A Literature Review of Software Defect Prediction: Using Ensemble Learning

MAYANK YADAV
Department of Software Engineering, Delhi Technological University
Correspond Author Email: dineshgupta1111@gmail.com

Abstract— Ensemble Learning is a vital area in machine learning. In recent years, it has captivated the interest of academics and researchers, leading to its use in fields such as software defect identification, fault prediction, and bug classification. This research presents a thorough examination of ensemble learning algorithms employed in software defect prediction during the last few years.

Index Terms— software defect prediction methods, ensemble methods, ensemble classifiers, defect prediction models.

I. INTRODUCTION

A software fault produces results that are not expected. To obtain the prediction outcome, the ensemble learning model is created. A fault, according to IEEE, is "any activity conducted by a developer that results in an imperfection." Prediction models are useful for software venture managers since they aid in quantitative planning as well as venture executives. Furthermore, the availability of free software measurement information archives has opened up new areas for study, implementation, and evaluation of machine learning algorithms for software deformity prediction models based on software measurements.

This part also discusses the topic of deformity prediction and how it is fed nourishment by the group learning to use predictive displaying.

II. RELATED WORK

Table A: List of Primary Studies

Study No.	Year	Technique Discussed
[1]	2018	Weighted randomized majority voting
[2]	2018	SADEsTSE model
[3]	2018	SMOTE
[4]	2018	PBIL-Auto-Ens
[5]	2018	ROS
[6]	2018	Multi-objective optimization for ensemble classification
[7]	2018	Adaboost
[8]	2017	Stacking
[9]	2017	Random Forest
[10]	2017	SmoteNDBoost
[11]	2017	Feature Selection
[12]	2017	Data Balancing
[13]	2017	Boosting
[14]	2017	Two Layer Ensemble
[15]	2017	EMKCA

Study [1] investigated the ensembles of weighted randomized majority voting systems. Providing high software dependability and quality control ensures users that they will be receiving a high-quality software at their end. A key component of the software development process is the software reliability prediction.

Study [2] developed a two-phase model which was used for prediction of defects. The existence or absence of flaws like faults, errors and bugs influences software product quality very much.

Study [3] created an ensemble approach, they used two rules for this. A software fault is an erroneous condition that occurs due to faulty specifications or flawed programming logic in some cases.

Study [4] picked the optimum ensemble approach and its parameters were a PBIL algorithm. This stands for 'population based incremental learning' algorithm. It is an evolutionary algorithm. Defective software modules prevent the software from functioning as intended, thereby increasing the development and maintenance cost. Also, contributing to customer unhappiness, which should be a major concern.

Study [5] demonstrated several oversampling approaches which are to be utilized in building an ensemble classifier which will be mitigating the impact of data belonging to minority classes.

Study [6] proposed that a multi-objective classification method is used for ensemble classification.
Study [7] analyzed NASA MDP data sets PC2 along with their assessment.
Study [8] researched for defect prediction, and found that stacking, bagging, and boosting are some ensemble classifiers. This study compared them individually to single classification techniques.

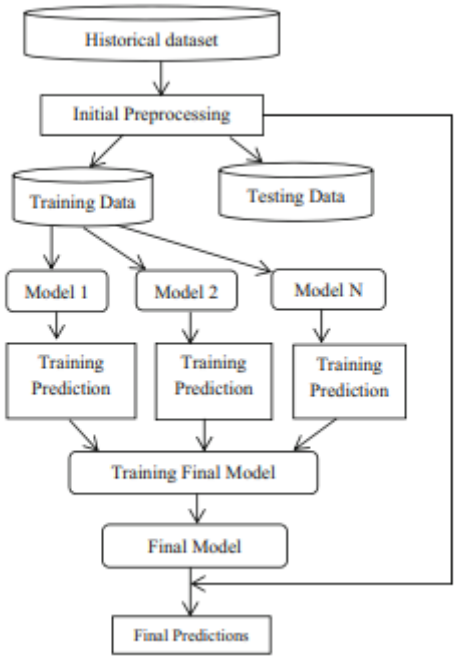

Figure A: Flowchart Diagram of Stacking Ensemble [15]

Study [9] found that Ensembles for bagging, boosting, and stacking were heavily employed.
Study [10] used resampling and ensemble approaches to enhance a model's prediction performance.
Study [11] proposed FS and Data Balancing (DB) which were integrated using ensemble approaches.
Study [12] utilized BTE, MVE, and NDTF are three heterogeneous ensemble models. These three models had two linear combinations each.
Study [13] mentioned heterogeneous and homogeneous ensemble approaches. The goal of heterogeneous ensemble approach is to identify software modules that contain bugs using the data taken from other projects. A large number of such techniques have been suggested so far. EMKCA which stands for Ensemble Multiple Kernel Correlation Alignment, is a unique approach based on homogeneous defect prediction concept. In this, the differences between different data distributions are reduced via kernel correlation alignment. Multiple kernel classifiers are combined together in order to combine the results of prediction on the basis of probabilistic outputs. The major roadblock is the difference occurring between the project taken as a target and a source both, while learning phase of prediction modelling.
Study [14] conducted a replication study that involved six datasets, obtained from Bugzilla, Columba, Eclipse JDT, Eclipse Platform, Mozilla, and PostgreSQL. These datasets are termed as large-scale software project datasets.
Study [15] suggested a heterogeneous defect prediction technique based on Ensemble Multiple Kernel Correlation Alignment. EMKCA is its abbreviated form.

Table B: Datasets and Techniques discovered

S. No.	Dataset Utilized	Technique Used
1	Ant 1.3	Weighted Randomized Majority Voting
2	Ant 1.5	SDAEsTSE Model
3	Ant 1.7	SMOTE
4	Camel 1.3	AdaBoost
5	Camel 1.5	PBIL-AUTO-ENS
6	Camel 1.7	ROS
7	Xalan 1.3	MWM
8	Xalan 1.5	FIDoS
9	Xalan 1.7	RF
10	Synapse 1.3	Bagging
11	Synapse 1.5	Boosting
12	Synapse 1.7	Stacking
13	Ivy 1.3	Multi-Object Optimization for Ensemble Classification
14	Ivy 1.5	SmoteNDBoost
15	Jedit 1.3	RusNDBoost

Some abbreviated forms have been used in the above table. Their full forms are as follows:
SMOTE stands for Synthetic Minority Oversampling Technique. RF stands for Random Forest. SDAEsTSE stands for Stacked De-noising Auto Encoders Two Stage Ensemble. AdaBoost stands for Adaptive Boosting. RUS stands for Random Undersampling.
Rest are basically different machine learning techniques. Some are combinations of basic existing techniques. Some have been newly introduced as a result of research work done by the researchers themselves.
Furthermore, these techniques are described briefly.
In Bagging we combine predictions of classification algorithms to improve results of our own targeted prediction. The objective is there in producing random training sets and then training these with a possible classification algorithm.

The class label is predicted using voting technique. For unstable learning algorithms that are namely decision trees and neural networks, it was found that these bagging results were highly suitable. As here smaller change in the training data results in a much bigger and substantial change in overall prediction result.

In Random Forest which is a subset of Bagging. Researchers found that random forest picks random characteristics to build models. These models came up with Decision Tree capabilities. The Random Forest technique works by randomly picking data and variables. This helps in generating decision trees of numerous amounts. Random Forest then combines the outputs of these multiple decision trees. After, tweaking and pruning of these decision trees is done. Aggregating the results from these many trees helps us in getting an accurate forecast.

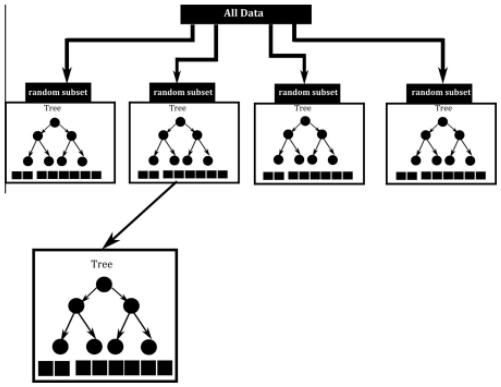

Figure B: Typical Representation of Random Forest [14]

In AdaBoost, we first do training of the model by consecutively learning from the mistakes seen in the results of the preceding models. We begin by identification of the cases which are difficult to predict. This is done with the help of basic classifiers. After that we use next classifiers in order to get better prediction results. Few weak classifier predictions are taken by the adaptive boosting algorithm.

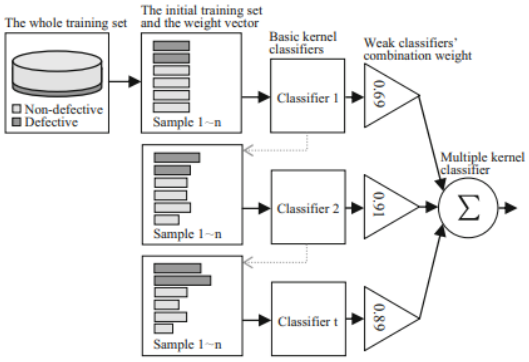

Figure C: Schematic Representation of AdaBoost [2]

In SMOTE oversampling is performed. All this oversampling strategy does is that it alters the class distribution in the dataset. By this oversampling is performed on the minority class. When work is done in feature space samples are produced. These samples are called synthetic samples. In order to perform Oversampling of the minority class, we need to remove each sample and generate corresponding synthetic sample. By doing this, all the k minority class nearest neighbors are connected with the line segment created in this way [16].

In Boosting, which is a common ensemble learning strategy in which we create a weak classifiers series. After doing this, we combine their outputs on the basis of weights relating them with the error rate of training. Though, it was revealed through one of the studies that Boosting was not primarily designed to address the imbalance of class. It can produce better classification results. Only when combined with other balanced algorithms. Balanced algorithms revealed were like sampling, cost-sensitive learning and a few more.

III. CONCLUSION

Previous research on the use of ensemble learning to predict software problems was thoroughly reviewed. This review has focused on the application of several techniques. Finally, ensemble learning techniques, approaches, and a variety of other algorithms have been proved to be efficient. All of this is based on their software defect prediction research. I have emphasized the effectiveness of ensemble learning methodologies through this review paper.

REFERENCES

1. S. Moustafa, M. Y. ElNainay, N. E. Makky, and M. S. Abougabal, ‘‘Software bug prediction using weighted majority voting techniques,’’ Alexandria Eng. J., vol. 57, no. 4, pp. 2763–2774, Dec. 2018
2. H. Tong, B. Liu, and S. Wang, ‘‘Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning,’’ Inf. Softw. Technol., vol. 96, pp. 94–111, Apr. 2018
3. S. A. El-Shorbagy, W. M. El-Gammal, and W. M. Abdelmoez, ‘‘Using SMOTE and heterogeneous stacking in ensemble learning for software defect prediction,’’ in Proc. 7th Int. Conf. Softw. Inf. Eng., pp. 44–47, 2018.
4. J. C. Xavier-Junior, A. A. Freitas, A. Feitosa-Neto, and T. B. Ludermir, ‘‘A novel evolutionary algorithm for automated machine learning focusing on classifier ensembles,’’ in Proc. 7th Brazilian Conf. Intell. Syst. (BRACIS), pp. 462–467.
5. S. Huda, K. Liu, M. Abdelrazek, A. Ibrahim, S. Alyahya, H. Al-Dossari, and S. Ahmad, ‘‘An ensemble oversampling model for class imbalance problem in software defect prediction,’’ IEEE Access, vol. 6, pp. 24184–24195.
6. S. Fletcher, B. Verma, Z. M. Jan, and M. Zhang, ‘‘The optimized selection of base-classifiers for ensemble classification using a multi-objective genetic algorithm,’’ in Proc. Int. Joint Conf. Neural Netw. (IJCNN), pp. 1–8.
7. N. Dhamayanthi and B. Lavanya, ‘‘Improvement in software defect prediction outcome using principal component analysis and ensemble machine learning algorithms,’’ in Proc. Int. Conf. Intell. Data Commun. Technol. Internet Things (ICICI) (Lecture Notes on Data

Engineering and Communications Technologies). Cham, Switzerland: Springer, pp. 397–406.
8. D. Bowes, T. Hall, and J. Petrić, ‘‘Software defect prediction: Do different classifiers find the same defects?’’ Softw. Qual. J., vol. 26, no. 2, pp. 525–552.
9. I. Alazzam, I. Alsmadi, and M. Akour, ‘‘Software fault proneness prediction: A comparative study between bagging, boosting, and stacking ensemble and base learner methods,’’ Int. J. Data Anal. Techn. Strategies, vol. 9, no. 1, p. 1.
10. X. Yu, J. Liu, Z. Yang, X. Jia, Q. Ling, and S. Ye, ‘‘Learning from imbalanced data for predicting the number of software defects,’’ in Proc. Int. Symp. Softw. Rel. Eng. (ISSRE), pp. 78–89, 2017.
11. C. W. Yohannese, T. Li, M. Simfukwe, and F. Khurshid, ‘‘Ensembles based combined learning for improved software fault prediction: A comparative study,’’ in Proc. 12th Int. Conf. Intell. Syst. Knowl. Eng., Nov., pp. 1–6, 2017.
12. X. Yang, D. Lo, X. Xia, and J. Sun, ‘‘TLEL: A two-layer ensemble learning approach for just-in-time defect prediction,’’ Inf. Softw. Technol., vol. 87, pp. 206–220, Jul. 2017L. Kumar, S. Rath, and A. Sureka, ‘‘An empirical analysis on effective fault prediction models developed using ensemble methods,’’ in Proc. Int. Annu. Comput. Softw. Appl. Conf., vol. 1, Jul. 2017, pp. 244–249Sureka, ‘‘An empirical analysis on effective fault prediction model developed using ensemble methods,’’ 2017.
13. Z. Li, X.-Y. Jing, X. Zhu, and H. Zhang, ‘‘Heterogeneous defect prediction through multiple kernel learning and ensemble learning,’’ in Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME), Sep., pp. 91–102,
14. Issam H. Laradji, Mohammad Al Shayeb, Lahouari Ghouti, “Software Defect Prediction Using Ensemble Learning on Selected Features.” Elsevier.
15. Sweta Mehta, K. Sridhar Patnaik, “Improved Prediction of Software Defects Using Ensemble Learning Techniques,” Neural Computing and Applications.
16. Ravi Patharia, Dr. Sanjay Singh Bhadoriya, “An Analysis of Multi-Cloud Environment with Security Challenges,” Journal of Innovative Engineering and Research, vol. 3, no. 2, pp. 16-19, 2020.