ISSN 2278-3091
Volume 8, No.2, March - April 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse33822019.pdf
https://doi.org/10.30534/ijatcse/2019/33822019

Progress on Machine Learning Techniques for Software Fault Prediction

Jyoti Goyal[1] **, Bal Kishan**[2]
[1,2]Department of Computer Science & Applications, Maharshi Dayanand University, Rohtak, Haryana (India)
[1] Jyoti.goyal24@gmail.com
[2]balkishan248@gmail.com

ABSTRACT

Software fault prediction is a significant part of software engineering. Fault prediction means to identify fault prone modules at the early stage of software development. It helps to reduce overall testing time, effort, and cost. It significantly improves the goodwill and profit of the organization by providing customer satisfaction. This area attracted many researchers over the years to improve overall software quality. Machine learning techniques are the most widely used techniques now-a-days in this area. This paper presents a comprehensive review on various machine learning techniques that will help the practitioners who are interested in building fault prediction model. This paper also discusses the substantial research performed in software fault prediction using machine learning techniques. A future dimension is also proposed to narrow the research gap by utilizing the research findings of existing models.

Key words: Classification, Machine learning, Software faults Prediction, Software metrics.

1. INTRODUCTION

Today, we are living in the world of computers where software's are used in almost every field of life. In 2018, the worldwide software development market is about $389 billion according to IT research and advisory firm Statista [1]. This data shows the importance of software. So, it is necessary for a software development company to deliver error free software. But practically it is not possible to make software 100% error free. We can reduce it by using well known techniques called fault prediction models which constitute the topic of this paper [2-6]. Software fault prediction (SFP) models are used to identify the fault prone modules at the early stage of software development because detecting fault at later stage will increase the cost exceptionally high. So, this will decrease the quality as well as leads to customer dissatisfaction. So, SFP models helps the testing team to focus more on fault prone modules and enables to optimize the utilization of resources [7][8].

Machine learning techniques play a significant role in software fault prediction. Various researchers have proved the importance of machine learning in SFP and it is empirically proved that the performance of the prediction model is highly influenced by the kind of technique used. So, it is essential to select the technique appropriate for the given dataset [9,10]. So, this paper presents various machine learning techniques that are utilized in the field of software fault prediction by various researchers over the years. Modifications in the existing ML techniques making them more efficient day by day that attracts many researchers in this field. A future dimension is also proposed to develop hybrid techniques for software defect prediction to improve the overall software quality.

Our next section discusses some selected machine learning techniques; Section 3 presents related work carried out over years by various researchers in chronological order and tabular form, Section 4 shows the comparison between different machine learning techniques used by different researchers, Section 5 presents the research contribution and at last section 6 presents the conclusion and future work.

2. MACHINE LEARNING TECHNIQUES USED FOR SOFTWARE FAULT PREDICTION

Machine learning methods are mainly categorized into two main categories:

- Supervised learning: Supervised learning is a method where both the predictors and response variables are given. We have various techniques like Decision tree, Random Forest, Naïve Bayes that comes under the category of supervised learning [11].
- Unsupervised learning: Unsupervised learning approach is basically used in those situations where no fault data is given. Here the algorithm finds the hidden structure or pattern in unlabeled data. In case of fault prediction, if we

need to predict faults at different levels then clustering will be the better approach [12].
Below is a brief introduction of some selected machine learning techniques.

2.1. Support vector machine

SVM comes under the category of supervised learning approach. It is used for both binary classification and finding the number of faults. SVM are most commonly used for classification problems. SVM are based on the concept of finding a hyperplane that best divides the dataset into two classes. When the distance between any training data of a class and a hyperplane is large then a good separation is obtained because a larger margin leads to a smaller error of classifier It works better on smaller cleaner dataset. The major drawback of SVM is that it is not possible to separate the dataset linearly in a finite dimensional space. The original finite space is mapped into higher dimensional space so that we can separate the dataset. Another problem is that it is not effective on noisier dataset with overlapping classes [13].

2.2. Naive bayes

The NB classifier is a probabilistic classifier based on the Bayes theorem, assuming that there is a strong (naive) independency between the features. Naïve Bayes classifier calculates the probability for every given input feature and then selects the outcome with the highest probability. Naive Bayes model is easy to build and is useful for large datasets. It required small amount of training data for classification process [14].

2.3. Random forest

Random develops lots of decision tree based on random selection of data and random selection of variable. The result of the output class is known as the mode of output classes obtained from the individual trees. It is based on two major belief that most of the tree can provide correct prediction of class for most part of the data and the tree are making mistakes at different place. In the previous studies it is proved that it works efficiently and increases the classification accuracy [15].

2.4. Neural network

Neural Network is a machine learning technique which is based on human brains. It is a collection of artificial neurons. The beauty of this technique is that it can be customized according to needs and problems because each artificial neuron takes number of inputs and provides single output. Neural can solve many complex problems which cannot be solved by human brain. GDA technique of neural network provide superior result for fault prone modules and it can provide simulated result in less number of iterations than other prediction models [16].

3. LITERATURE SURVEY

This section presents some latest ongoing research performed in SFP using machine learning approaches.

Amritanshu et al (2018) proposed smotuned that is an automatic parameter setting tool which self-tunes the parameters for each dataset. The author uses various learners i.e. RF, LR, KNN, NB, DT, SVM and empirically he proves that no learner is best across all datasets. So instead of finding the best learner we should focus on creating the better training data because improvement in the fault prediction model is independent of good classifier [17].

Garvit et al. (2018) proposed the fault prediction model using three classifiers DT, KNN and Random Forest. The author proposed two new set of change metrics i.e. LOC-WORKED-ON, MAX-LOC-WORKED-ON that increase the accuracy of the fault prediction model [18]. F. Karimian et al. (2017) presented the paper for evaluation of classifiers. Authors analyses two issues for the selection of classifiers. First, selection of appropriate set of metrics and instance sampling to deal with the problem of class imbalancing. After analysis, we conclude that the software quality prediction model without balancing up of classes will not produce efficient fault predictors also feature selection has less effect on model performance [19].

David Bowes et al. (2017) able presented a very novel sight that each classifier can identify different kinds of faults. He empirically proves that each classifier has their own prediction capacity. Some classifiers are consistent with the set of detected defects, but some may vary. Here the researcher does not focus only on performance figure but on the different set of defects detected by classifiers [20]. Lov et al. (2017) proposed Least square support vector machine for building fault prediction model. The performance of SFP model depends on the input features of the model and to select the appropriate features feature selection and feature ranking methods are used. These methods help to find the set of metrics having good discriminatory power which in turn reduces the misclassification rate [21].

Santosh Singh et al (2017) proposed heterogeneous ensembling method means number of different base classifier are used to predict number of faults in a given data set. This approach is based on the assumption that each different base classifier has different ability to predict different types of fault [23]. Sanjay et al (2017) develops a framework that validate and select only those set of source code metrics which increases prediction performance. The author uses t-test analysis and Univariate logistic regression analysis to evaluate the potential of source code metrics in predicting fault proneness of a module. It is empirically proved that reduced set of metrics provides good accuracy with less misclassification errors [24].

Gitika et al. (2016) propose a framework for providing support in the development of Ideal BTS by creating a precedence list of various mining algorithms which are used in Software Bug classification.The result shows that chi square and correlation methods are the best indicator of severity of bugs than feature selection methods [25]. Divya et al. (2016) uses WLSTSVM technique for software fault prediction. Also, the author focuses on misclassification cost because most of the defect data generally suffers from the problem of class imbalancing. So, misclassification cost is assigned to the software modules of each class to compensate the negative effect of the imbalanced data on the performance of software defect prediction. The result shows that WLSTSWM is better than other techniques. But results varied with the different features selection and parameters selection techniques [26].

Tiejian et al. (2015) proposed the use of Multiple kernel with ensemble learning approach for predicting defective modules. MKEL is a supervised approach and based on the historical data that generally suffers from data imbalancing problem. So, to reduce the misclassification cost author proposed weighted vector updating procedure that overall improve the performance of the model [27].

Ezgi et al. (2016) proposed an iterative software defect prediction model that uses fuzzy inference system The result shows that it is a successful technique and it becomes an automated tool to locate fault-prone modules. It is also implemented as a plug-in for the Eclipse environment [28]. Santosh et al. (2016) demonstrates the capability of DTR for finding the number of faults in two different releases of the software i.e. inter release where training and testing data are from different release and intra release where training and testing data are from same release. The results proved that DTR with intra release have better accuracy than inter release [29].

Diego et al. (2016) deals with the situation when it is difficult to classify the module into defective and non-defective ones. So, the author designed an alternative called reject option where modules that does not come under the category of defected and non-defected are rejected for expert opinion. So, this method reduces misclassification error up to great extent [30]. Rathore et al. (2016) fills the research gap where each practitioner is working on binary classification only. Here the author is estimating the number of defects in a given module so that it will help the testing team to optimize the scarce resources. [31]. Ezgi et al. (2016) proposed the framework for application of the "ANFIS". The proposed framework uses McCabe metrics and suggest using the expert knowledge with ANFIS. The performance achieved by ANFIS is 0.8573 [32].

Ming Tan et al (2015) proposed the novel approach called online fault prediction means predicting the faults at change level. Here the author removes the limitation of previous works i.e. imbalancing of training data, delay between training the model and testing the model and false high precision of the model by resampling and updatable classification. This model convinces the developers to believe in the benefits of fault prediction model [33].

Issam et al. (2015) consider the two main issues related to fault data i.e. data imbalancing and feature selection. To deal with these two issues they proposed a software fault classification method based on ensembling that is "average probability ensemble" learning module. The proposed APE system incorporates two main classifiers: random forest and weighted SVMs (WSVMs) and the results proved that the proposed ensembling model provide good performance with still having poor features [34]. Agasta et al. (2014) explains the use of supervised learning methods for software fault prediction in case where fault data is not available. They propose genetic algorithm for binary classification by using the data from the similar projects for training the model. The proposed technique is performing well on given data set [35]. Santosh et al. (2014) performs this study based on the fact that the result of the prediction model is influenced by the quality of fault data and to maintain the quality of data feature ranking and feature selection techniques play a significant role. The author empirically proved that feature ranking techniques improves more efficiency of the model than feature selection techniques [36].

Verbraken et al. (2013) proposes a model for fault prediction that uses Markov blanket principle for selection of features and different BN classifiers for making simple and comprehensible networks with minimum arcs and nodes.

AUC and H-measure is used as a performance metric and the results shows that augmented BN classifier is better among other different BN classifiers [37]. Menka et al. (2013) proposed a method that used Synthetic data Program (SD). This method uses two step process for developing a model. One step is for training the model using training data and another is for testing the model using testing data. The proposed model focused on software fault classification based on their recovery strategies [38].

Rathore et al. (2012) evaluates the capability of design level metrics to predict faults in individual and combined basis. The result demonstrated that CBO, RFC, import and export coupling metrics are equally important for predicting faults. But this paper analysed the result at class level but not system level [39]. Ayse et al. (2011) provided a generic fault prediction model based on ensembling which is implemented on embedded software projects. The proposed framework uses three algorithms i.e. the "NBM", "ANN" and for ensembling they use "VFI". The results show that false alarms have reduced up to 15% and precision has increased up to 43% while keeping balance rates up to 74% [40].

Chen et al. (2010) designed a novel approach "Fuzzy Support Vector Regression" for predicting fault counts in a given module. To handle unbalanced software metrics dataset, fuzzification input of regressor is used. The result states that FSVR provides better prediction for fault counts than conventional (SVR) [41].

The research performed by various researches along with their limitations is briefly mentioned in table.1

Table 1: Summary of Selected Studies Along with the Proposed Technique

Pub. Year	**References**	**Authors**	**Proposed Technique**	**Limitations**
2018	[17]	Amritanshu et al	Random forest Logistic regression	SMOTUNED (SMOTE with automatic parameter setting tool) improves performance of model which is independent of classifiers.
2018	[18]	Garvit et al	Random forest Decision tree(J48) KNN	Improving performance using change metrics only for binary classification.
2017	[19]	F. Karimian et al	Bagging K* Random forest	The performance of models improves when training data is created using sampled data over original data.
2017	[20]	David et al	Naive Bayes SVM Random Forest	Does not specify which feature better suits to specific classifier.
2017	[21]	Lov et al	Neural network BTE method	No fault count and specific to object-oriented paradigm only.
2017	[23]	Santosh et al	Genetic programming Multilayer perceptron Linear regression	Need to be implemented on industry projects to generalize the findings of the study.
2017	[24]	Lov et al.	LSSVM	Restricted with OO approach only.

2016	[26]	Divya et al	WLSTSM	Biased approach.
2016	[29]	Santosh et al	Decision Tree Regression	The findings are not generalized i.e. proper care of underlying pattern of faults should be taken.
2016	[30]	Diego et al	rejoELM and IrejoELM	Success of the model based on expert knowledge.
2016	[31]	Rathore et al	Decision tree regression	More possibility of biasness.
2015	[32]	Erturk et al	ANN, SVM ANFIS	Not applicable in the absence of experts.
2015	[34]	Issam et al	RF, GB Regular and weighted SVM	Does not compare the performance of APE ensembling technique with majority voting.
2013	[37]	verbraken et al	Bayesian Network	Results are not generalized.
2012	[39]	Atul et al	ULR,MLR	Applicable for object-based system only.
2011	[40]	Ayse et al	NB, ANN,VF!	No comparative analysis is presented.
2010	[41]	chen et al	FSVR	Results are not evaluated thoroughly.

4. COMPARISON OF DIFFERENT MLT FOR FAULT PREDICTION

In the above section we have discussed various machine learning techniques used by researchers for prediction of faults. The performance of each technique varies according to dataset. So it is the responsibility of the practitioner to select the best technique depending upon the requirements of the dataset. The following figure 1 shows the comparative use of Machine learning techniques over last few years.

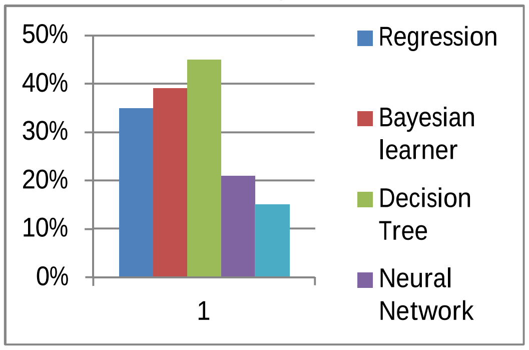

Figure 1: Comparative Use of Various ML Techniques over last few years

It is clear from the extensive literature survey that decision tree is the most widely used techniques for prediction of faults. Bayesian learner and regression is also used by the researchers depending upon the requirements of the dataset. In future it is better to implement hybrid approaches to improve the accuracy of the model [42][43].

5. RESEARCH CONTRIBUTION

This paper presents a comprehensive survey to show the current trend of various machine learning techniques to predict faults in software modules at different levels. Most of the experimental work done by researchers is based on promise data repository which does not reflect the real-life problems. After detailed literature survey, we find some limitations in the existing research work that is also presented in the tabular form. In the previous research works there are various issues that need to be reconsidered like class rebalancing, threshold dependent performance measure and unavailability of well documented modelling scripts from published settings so that we are not able to generalize the findings of the study. This survey will guide the practitioners to explore more problems and hence solve them by providing the relevant solution.

6. CONCLUSION AND FUTURE WORK

This paper presents detailed review on various machine learning techniques for SFP. SFP is necessary for minimizing the cost as well as time of software testing. Those modules which are more prone to errors requires more resources. SFP enables testing team to optimally utilize the resources which helps to improve the quality of the system. The aim of this study is to access research works done by various researchers related to machine learning techniques for software fault prediction so that it will helps the practitioners who are interested in building fault prediction model. After detailed review we found that random forest, neural network and naïve Bayes are good enough for SFP, but no single technique is appropriate for all kinds of dataset. So, it is better to choose the result from the set of prediction models. Hence, in future we are planning to implement heterogeneous ensembling to overall increase the efficiency of the system.

REFERENCES

[1] M. Jrgensen, K. Molkkenstvold, **how large are software cost overruns?** A review of the 1994 CHAOS report, Information and Software Technology 48 (4) (2006) 297{301}.

[2] H. Uwano, Y. Kamei, A. Monden, K.-i. Matsumoto, **an analysis of cost overrun projects using financial data and software metrics, in: Software Measurement**, 2011 Joint Conference of the 21st International Workshop on and 6th International Conference on Software Process and Product Measurement, 2011, pp. 227{232}.

[3] S. Grimstad, M. Jrgensen, K. Molkkenstvold, **Software effort estimation terminology**: The tower of babel, Information and Software Technology 48 (4) (2006) 302{310}.

[4] M. Bloch, S. Blumberg, J. Laartz, **delivering large-scale it projects on time, on budget, and on value**, McKinsey on Business Technology (27) (2012) 2{7}.

[5] Sandeep D and R. S, **Case Studies of Most Common and Severe Types of Software System Failure**, International Journal of Advanced Research in Computer Science and Software Engineering vol. 2, pp. 341-347 August 2012.

[6] Venkata U and R. A, **Empirical Assessment of Machine Learning based Software Defect Prediction Techniques**, Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems 2005.

[7] Robert N, **Why Software Fails**, 2005.

[8] Rajkumar G and K.Alagarsamy, **The Most Common Factors For The Failure Of Software Development Project**, vol. 11, pp. 74-77, January 2013.

[9] L. J, **Major Causes of Software Project Failures** CROSSTALK the Journal of Defense Software Engineering pp. 9-12, July 1998.

[10] Rathore, S. S., & Kumar, S, **A study on software fault prediction techniques** Artificial Intelligence Review, 1–73. https://doi.org/10.1007/s10462-017-9563-5

[11] Hammouri, A., Hammad, M., Alnabhan, M., & Alsarayrah, F. **Software Bug Prediction using Machine Learning Approach** International journal of advanced computer science and applications, 9(2), 78-83, 2018.

[12] Amruthnath, N., & Gupta, T. **A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance** in 2018 5th International Conference on Industrial Engineering and Applications (ICIEA) (pp. 355-361). IEEE.2018

[13] Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. **Numerical Recipes: The Art of Scientific Computing**, Section 16.5, Support Vector Machines, Cambridge University Press, The 3rd Edition, 2007

[14] John, G. H., & Langley, P. **Estimating continuous distributions in Bayesian classifiers**. In the 11th Conference on Uncertainty in artificial intelligence, pp. 338-345,1995

[15] Leo Breiman. **RANDOM FORESTS**. Machine Learning, pp. 5-32.2001

[16] Kumar, R., & Gupta, D. L. **Software bug prediction system using neural network**. Eur. J. Adv. Eng. Technol, 3(7), 78-82.2016

[17] Agrawal, A., & Menzies, T. **Is better data better than better data miners**? on the benefits of tuning SMOTE for defect prediction. In Proceedings of the 40th International Conference on Software Engineering (pp. 1050-1061). ACM may 2018.

[18] Choudhary, G. R., Kumar, S., Kumar, K., Mishra, A., & Catal, C. **Empirical analysis of change metrics for software fault prediction.** Computers & Electrical Engineering, 67, 15-24. 2018

[19] Karimian, F., & Babamir, S. M.. **Evaluation of Classifiers in Software Fault-Proneness Prediction**. Journal of AI and Data Mining, 5(2), 149-167, 2017.

[20] Hall, T. **Software defect prediction**: **do different classifiers find the same defects? 2017.**

[21] Kumar, L., Sripada, S. K., Sureka, A., & Rath, S. K. **Effective fault prediction model developed using least square support vector machine (lssvm)**. Journal of Systems and Software, 137, 686-712.2018

[22] Goyal, R., Chandra, P., & Singh, Y**. Fuzzy inferencing to identify degree of interaction in the development of fault prediction models.** Journal of King Saud University-Computer and Information Sciences, 29(1), 93-102.2017.

[23] Rathore, S. S., & Kumar, S. **Towards an ensemble-based system for predicting the number of software faults**. Expert Systems with Applications, 82, 357-382.2017.

[24] Kumar, L., Misra, S., & Rath, S. K.. **An empirical analysis of the effectiveness of software metrics and fault prediction model for identifying faulty classes**. Computer Standards & Interfaces, 53, 1-32.2017.

[25] Sharma, G., & Sharma, S. **Mining Algorithms Precedence List for Software Bug Classification**, 2016, 9(46), 1–16.
[26] Tomar, D., & Agarwal, S. **Prediction of defective software modules using class imbalance learning**. Applied Computational Intelligence and Soft Computing, 2016, 6.
[27] Wang, T., Zhang, Z., Jing, X., & Zhang, L. **Multiple kernel ensemble learning for software defect prediction**. Automated Software Engineering, 2016, 23 (4), 569-590.
[28] Erturk, E., &Sezer, E. A. **Iterative software fault prediction with a hybrid approach. Applied Soft Computing**, 2016, 49, 1020-1033.
[29] Rathore, S. S. **A Decision Tree Regression based Approach for the Number of Software Faults Prediction**, 2016.41(1), 1–6. https://doi.org/10.1145/2853073.2853083
[30] Mesquita, D. P., Rocha, L. S., Gomes, J. P. P., & Neto, A. R. R. **Classification with reject option for software defect prediction.** Applied Soft Computing, 2016, 49, 1085-1093.
[31] Rathore, S. S., & Kumar, S. **Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems.** Knowledge-Based Systems, 2017, 119, 232-256.
[32] Erturk, E., &Sezer, E. A. **A comparison of some soft computing methods for software fault prediction**. Expert systems with applications, 2015, 42 (4), 1872-1879.
[33] Tian, Y., Lo, D., Xia, X., & Sun, C. **Automated prediction of bug report priority using multi-factor analysis**. Empirical Software Engineering, 2015, 20(5), 1354-1383.
[34] Laradji, I. H., Alshayeb, M., & Ghouti, L. **Software defect prediction using ensemble learning on selected features**. Information and Software Technology, 2015, 58, 388-402.
[35] Adline, A., & Ramachandran, M. **Predicting the software fault using the method of genetic algorithm**. Int. J. Adv. Res. Electr. Electron. Instrum. Eng, 2014, 3(2), 390-398.
[36] Rathore, S. S., & Gupta, A. **A comparative study of feature-ranking and feature-subset selection techniques for improved fault prediction.** In Proceedings of the 7th India Software Engineering Conference (p. 7). (2014, February) ACM.
[37] Dejaeger, K., Verbraken, T., &Baesens, B. **Toward comprehensible software fault prediction models using bayesian network classifiers.** IEEE Transactions on Software Engineering, 39(2), 237-257, 2013.
[38] Gupta, M., & Gautam, P. **A Novel Approach for Identifying Software Fault Prediction in mining**.
[39] Rathore, S. S., & Gupta, A. **Investigating object-oriented design metrics to predict fault-proneness of software modules**. In Software Engineering (CONSEG), 2012 CSI Sixth International Conference on (pp. 1-10) . (2012, September) IEEE.
[40] Mısırlı, A. T., Bener, A. B., & Turhan, B. **An industrial case study of classifier ensembles for locating software defects**. Software Quality Journal, 2011 19(3), 515-536.
[41] Yan, Z., Chen, X., & Guo, P. **Software defect prediction using fuzzy support vector regression**. In International symposium on neural networks (pp. 17-24). (2010, June) Springer, Berlin, Heidelberg.
[42] Maria zemzami, **A modified particle swarm optimization algorithm linking dynamic neighbourhood topology to parallel computation**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol 8, no. 2, 2019,112-118. http://doi.org/10.30534/ijatcse/2019/03822019.
[43] Maria Zemzami, **An evolutionary hybrid algorithm for complex optimization problems**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol8,no.2,2019.https://doi.org/10.30534/ijatcse/2019/05822019.