IJRETS: International Journal of Research in Engineering, Technology and Science,
Volume XIII, Issue VIII, January. 2021

SOFTWARE DEFECTS PREDICTION USING SUPERVISED AND UNSUPERVISED MACHINE LEARNING APPROACHES: A COMPARATIVE PERFORMANCE ANALYSIS

Richa Vats , Dr. Arvind Kumar

SRM University, Delhi-NCR, Sonepat, Haryana, 131029, India
ritzi1606@gmail.com, k.arvind33@gmail.com

ABSTRACT:

Software defect prediction is a sub class of software engineering process which is used to determine the defects in the software modules. Its important task is to achieve reliable software and identify the defects before the delivery of software. This work highlights the applicability of machine learning methods to determine defects in software module. In this work, supervised and unsupervised machine learning techniques are adopted for defect prediction. These are bagging, K-means, AdaBoost, random forest, and K-harmonic means (KMH). The aim of this work is to identify which method is more suitable for defect prediction in software. The performance of these methods is evaluated using nine benchmark defect predication datasets. Simulation results showed that supervised machine learning techniques has state of art result for defect prediction as compared to unsupervised machine learning techniques.

Keywords: *Software, Defects, Supervised Learning, Unsupervised Learning, AdaBoost, Bagging, Random Forest, K-means, K-harmonic means.*

[1] INTRODUCTION

In present time, software's play significant role in human life. Day to day work is carried out using software enabled system. So, the quality of software become an important concern for software developers. If, any of the module of a software is faulty, then the working of the software is affected and leads to unpredictable behaviour. During the development cycle of software process, some bugs or faults can be induced and these faults tend to defects in software. In turn, the quality of software can be degraded and sometimes lead to failure of the software. The main reason behind these faults is human action. These faults and defects can be described as follows- a fault and defect in software are human errors that are mistakenly embedded during the development of software of product. It can be interpreted as a programmer can build a program, but forget to use initialization bracket, data type, use duplicate variable etc. Due to above mentioned mistake, the program does not run successfully and might give some error during compilation time. A defect or fault can be described as incorrect code; data definition etc. which can occur in software and hardware of a system. The defect can lead to the failure of software. To detect the defects in software module is one of the rigorous task and requires lot of time, effort and manpower. So, software defect prediction is the sub part of the software quality that can predict the defects in software modules and ensure quality of software. It can also help to develop the software in timely manner. The key advantages of defect prediction are highlighted below [1].

- To improve and enhance the system quality and reliability.
- Rearrangement and refactoring of software modules during the maintenance phase, if required.
- It is also helpful to choose best alternative design during design phase.

- To ensure the stability and higher assurance of developed software.
- Time and effort devoted during the code review process is effectively reduced using software fault prediction.

Due to above mentioned key points, software defect prediction becomes a good research activity. In last few spans, this problem attracted wide attention from research community. Large numbers of defect prediction methods or techniques are developed by various researchers. The experimental and theoretical methods can be developed to find optimum solution for software defect prediction problems. Some of the most popular predication methods are NB, DT, NN, SVM, LR, and Random Forest etc. Several soft computing approaches are also applied for defect prediction like- ANFIS, ELM, weighted ELM and so on. During the extensive literature review, it is observed that the applicability of supervised and unsupervised machine learning algorithm for software defect prediction is an active area of research or debt [2-4]. Researcher also adopted supervised and unsupervised machine learning methods for defect prediction [2-3]. Initially, researchers explored the supervised machine learning algorithm for software defect prediction [5-7]. But, these methods work well in the presence of historical data. These methods require lot of training data for accurate prediction of defects. In meanwhile, some researchers also focused on the applicability of the unsupervised machine learning methods for defect prediction [6, 8]. These methods are quite useful in absence of historical data. Other points that make unsupervised technique more beneficial are simple implementation, no training data required, ease to use with new project and less computational time as compared to supervised machine learning method. Hence, the aim of this research work is to compare the performances of different supervised and unsupervised machine learning methods and determine which method is more suitable for software defect predication. So, for this work, Adaboost, Bagging, Random tree, IBK, K-means, EM and K-harmonic means clustering methods are applied for accurate prediction of software defects. Adaboost, Bagging, Random tree and IBK methods are classified as supervised machine learning techniques. Rest of methods are classified as unsupervised machine learning techniques. The performances of these methods are evaluated using several benchmark datasets downloaded from PROMISE repository. Rest of paper is organized as follows- Section 2 presents the reported work in field of software defect predication. Sections 3 and 4 present the supervised and unsupervised techniques adopted for defect predication. The simulation results are demonstrated in section 5. The entire work is summarized in section 6.

[2] RELATED WORKS

Rong et al., [9] applied SVM model to predict the software defects. In this work, authors optimize the parameters of SVM using bat algorithm, called CBA-SVM. The simulation results are taken over standard bench mark defect prediction datasets. It is stated that the CBA-SVM model gives more promising results in comparison to other algorithms.

Mausaa and Grbaca considered genetic programming method to detect the software defects [10]. The genetic programming method integrates with different selection strategies for handling population diversity. Moreover, colonization and migration operators are also integrated with genetic programming method. The performance of the proposed method is evaluated on standard defect prediction datasets, downloaded from UCI repository. Authors claimed that genetic programming method obtains promising results for defect prediction problems.

Ozturk et al., [11] investigated the performance of clustering algorithms for defect prediction. In this work, four variants of K-mean clustering algorithm are taken into consideration. The performance of these variants is tested on four real life datasets. Authors claimed that K-mean++ variant gives better results than other K-mean variants.

Ni et al., [12] explored the multi objective algorithm to determine defects in software. In this work, Pareto based concept was consider to handle defect prediction problem. The proposed algorithm consider two objective functions in terms of minimization and maximization. RELINK and PROMISE datasets are used to evaluate the performance of proposed algorithm and gives quality results.

Xu et al., [13] developed a subset selection model to address the defect prediction problem. In first stage, the proposed model considers sparse modelling selection method to select the initial model from historical datasets. In second stage, dissimilarity based sparse representation is used to refine the selected subset. Moreover, extreme machine learning classifier is adopted to classify the datasets. Simulation results showed that two stage model gives improved results as compared to eleven defect prediction models.

Malhotra and Kamal examined the performance of oversampling method to detect the accurate defects in imbalanced datasets [14]. In their work, five oversampling methods are used to detect the defects. Further, in this work a new oversampling method called SPIDER3 is also proposed for imbalanced defect prediction datasets. The performance of above mentioned methods is evaluated using twelve imbalanced NASA repository datasets. The simulation results stated that integration of oversampling method with machine learning classifiers improves the performance of these algorithms.

Singh et al., [15] developed an automatic framework to extract the fuzzy rules for software defects. The proposed model has capability to determine attributes of faults. Initially, the model assumed that every attribute is a useless feature. The performance of the proposed framework is investigated on publically available software defect datasets. It is seen that the proposed model is capable to find fuzzy rules for software faults.

Chen et al., [16] considered the data dimension to improve the accuracy rate and developed a multi-view transfer learning method. The proposed method can also work with heterogeneous data. In the proposed model, class labels are learned using neural network approach. Authors claimed that the proposed model provides state of art prediction results.

Balogun et al., [17] applied several clustering techniques on software defect prediction problem and provided a comparative performance analysis of these techniques. In this work, K-mean, X-mean, hierarchal clustering, density based clustering and Expectation minimization methods are considered for defect prediction problem. The performance of these techniques is evaluated using eight benchmark dataset. It is noticed that first clustering method provides optimum results than other clustering methods.

Bowes et al., [18] evaluated the performance of several classifiers to detect defects. These classifiers are RF, NB, RPart and SVM. The standard dataset from NASA, open source and commercial are considered for defect prediction. Authors claimed that although all classifiers have similar performance for defect prediction, but these classifiers indentify different set of defects.

It is observed that suitability of supervised and unsupervised methods for defect prediction is an active area of research. Chen et al.,[19] focussed on above mentioned research area. In their work, two unsupervised and eleven supervised methods are selected to evaluate rank of module. It is noticed that unsupervised method can be worked as baseline method for defect prediction.

To minimize the classification cost, Siers and Islam developed cost sensitivity classification technique called CSVoting for defect prediction [20]. The proposed technique is an ensemble method of decision tree approach. The proposed technique is tested over six defects datasets. Authors claimed that CSVoting method provides superior results than compared methods.

Ji et al., [21] applied an improved Naive Bayes algorithm with kernel density estimation to improve accuracy rate for defect prediction. The performance of improved Naive Bayes algorithm is tested on ten NASA repository defect datasets. Simulation results are compared with NB, SVM, Random Forest and logistic regression techniques and NB with kernel estimation gives superior results.

Machine learning methods for defect prediction is presented [22]. In this work, ANFIS, ANN and SVM are considered to detect the software defects in efficient manner. The performances of these methods are evaluated using PROMISE repository defect datasets. It is observed that ANN obtains slightly better results than ANFIS, whereas, SVM exhibits worst performance among all three methods.

Lamba et al., [23] applied several machine learning methods for bug prediction. The methods are linear regression, RF, NN, SVM and DT. The performances of these algorithms are evaluated using standard defect prediction datasets. It is revealed that SVM method outperforms among all other methods for bug prediction.

Ji et al., [24] proposed a weighted NB classifier based on the concept of information diffusion. Further, six weight assignment methods are considered to determine optimum weight of features. The performance of weighted Naive bayes is examined over ten defect prediction datasets. These datasets are taken from PROMISE repository. Authors claimed that proposed improvements significantly improves detection rate of defects.

Laradji et al., [25] developed an ensemble learning method for accurate prediction of software defects. In this work, feature selection technique is integrated with ensemble classifier. The aim of feature selection technique is to handle imbalance data and redundancy feature. The benchmark software defect prediction datasets are considered to evaluate the performance of proposed classifiers. The simulation results showed that greedy forward selection method performs better than other feature selection methods.

To reduce the decision cost, Li et al., [26] developed a decision framework for software defects. In proposed framework, three way decision and ensemble learning is integrated to predict software defects. It is revealed that the proposed frame work provides better prediction accuracy.

Liu et al. Developed two phase transfer learning model to overcome the limitation associated with TCA+ [27]. In first phase, source project estimator is developed to select the source project with higher distribution similarity. In second phase, leverage TCA+ model is used to make prediction model. The performance of model is evaluated on forty two defect datasets. It is observed that proposed two phase learning model significantly improves the defect prediction accuracy. A review on machine learning techniques adopted for defect prediction is reported in [28].

Marjuni et al., [29] applied the unsupervised approach for software defect prediction due to absence of historical data. In their work, signed Laplace spectral classifier is used to predict defects. The simulation results stated that proposed signed classifier significantly improve the performance of unsupervised method.

Marjuni et al., [30] developed LM based classifier to improve the reliability of decision making. In their work, two variants of weighted ELM are proposed to handle the software defect prediction. Both the variants use the concept of reject option when classification is performed. The performance of classifiers is evaluated on standard datasets. It is concluded that rejoEM provides better result as compared to other ELM based classifiers.

Mori and Uchihira developed a superposed Naive bayes to determine the defects in software [31]. Simulations results are taken on thirteen datasets. It is noticed that superposed NB provides a balance between accuracy and interpretability.

Ryu and Biak developed a multi-objective NB classifier for measuring defects in software [32]. In their work, three objectives are considered for addressing the class imbalance issue. The multi-objective NB provides more promising results in comparison to single and multi-objective approaches.

To handle the software fault prediction task, Erturk and Sezerb developed an iterative defect prediction model based on hybrid approach for identification of defects in software[33]. The proposed model works in two modules. In first module, fuzzy inference system is used to make initial prediction. Whereas, in second module, data driven methods are employed to measure final outcome. Several benchmark datasets are downloaded from PROMISE. Simulation results indicated that iterative model significantly identifies the defect in software modules.

Wang et al., [34] employed multiple kernels leaning to predict the defects in software. Moreover, the multiple kernel learning is embedding with ensemble learning for accurate prediction. It is revealed that the combination of multiple kernel learning and ensemble classifier achieves higher accuracy rate.

Wei et al., [35] adopted support vector machine and local tangent space alignment, called LTSA-SVM to detect defects in software. In the proposed method, SVM works as baseline classifier to predict defects in software. While, the user defined parameters of SVM are optimized using grid search and ten cross fold validation technique. The LTSA method is applied to extract the features of dataset. The simulation results are compared with simple SVM, LLE-SVM and it is noticed that LTSA-SVM provides more promising results than other methods.

Xu et al., [36] developed a prediction model to determine defects in software datasets. The proposed defect prediction model is combination of kernel PCA and weighted extreme machine learning. In their work, kernel PCA is applied to determine the optimum features from data. The work of WEML is to predict the defects using reduced dataset. Forty four projects are considered in this work, out of forty four projects, thirty four projects are chosen form PROMISE repository, while ten are selected from NASA repository. The proposed model obtains better results compared to similar models.

Yadav et al. developed a fuzzy based approach to handle the software defects during the development cycle of software [37]. The proposed approach is tested using twenty real life datasets. It is revealed that accuracy of proposed approach is near to actual defect prediction rate.

Yousef applied the data mining algorithms for defects prediction [38]. In their work, three data mining algorithms i.e. NB, NN and DT are adopted for same. The performances of these algorithms are evaluated using defect datasets from NASA repository. It is observed that NB outperforms than NN and DT algorithms.

[3] SUPERVISED MACHINE LEARNING METHODS

This section presents the supervised machine learning techniques adopted for software defect predication.

3.1 AdaBoost

AdaBoost is an ensemble classifier worked with a set of classifiers [39]. This algorithm processes the classifiers in sequential manner, whereas bagging algorithm can process the classifier in parallel fashion. Moreover, the AdaBoost algorithm has capability to change the weights of training instances. The aim of this strategy is to minimize the expected error over different input. For given a training set X, initially specify the number of trails i.e. T. After that T weighted training sets are computed from X such as S_1,S_2,…..S_T and describe the T classifier for weighted training sets like C_1,C_2,…..C_T . The algorithmic steps of AdaBoost algorithm are mentioned below.

Algorithm 1: Steps of AdaBoost Algorithm	
Step 1	*Initialize the input training set (X), inducer (M) and integer trails (N)*
Step 2	X' *=X // instance weight to be 1*
Step 2	*for i=1 to N* *{*
Step 3	$C_i = M(X')$
Step 4	α_i *= arg max* $\sum_{x_j \in X' :\, C_i(x_j) \neq y}(x)$
Step 5	*If* $\alpha_i > 1/2$
Step 6	$\beta_i = \alpha_i/(1-\alpha_i)$
Step 7	*For each* $x_j \in X'$*, if* $C_i(x_j) = y_j$ *then weight* x_j *= weight* $x_j.\beta_i$
Step 8	*Normalize the weight of training instances.* *}*
Step 9	*C*(x)= arg max* $\sum_{i \in C_i(x)=y} \log \frac{1}{\beta_i}$
Step 10	*Compute final outcome*

3.2 Bagging

Breiman developed the Bagging algorithm in 1996 based on the different bootstrap samples [40]. It is extension of the bootstrap algorithm which is developed by Efron & Tibshirani in 1993. The bootstrap sample can be computed using uniform distribution m instances from the training set. Moreover, T bootstrap sample such as $A_1, A_2, \ldots . A_T$ is determined using training set replacement. Further, a classifier C_i is designed for each bootstrap sample. The complete

classifier C is formed using $C_1, C_2, \ldots . C_T$. The final outcome of classifier C is computed through sub classifiers $C_1, C_2, \ldots . C_T$. The algorithmic steps of bagging classifier is listed below.

Algorithm 2: Steps of Bagging Algorithm	
Step 1	*Initialize the input training set (X), inducer (M) and number of bootstrap samples (N)*
Step 2	*for i=1 to N* *{*
Step 3	X' *= bootstrap sample X*
Step 4	$C_1 = M(X')$ *}*
Step 5	*C(x)= arg max* $\sum_{i \in C_i(x)=y} 1$ *// measuring class label*
Step 6	*Compute final outcome*

3.3 Random Forest

Random Forest is an ensemble classifier developed by Breiman [41]. This classifier consists of k number of decision tree. The decision tree is designed using the bootstrap method. In each iteration, the candidate solution is selected from a set of variables in random fashion. Further, a tree is constructed using bagging method. The algorithm steps of random forest technique are highlighted below.

Algorithm 3: Steps of Random Forest	
Step 1	*Divide the training set (X) into K subsets such as* $X_{train} = X_1, X_2, \ldots . X_K$
Step 2	*Construct the decision tree using C 4.5 method for each training subset* X_i *and K number of decision tree is formed for K number of training subsets.*
Step 3	*Combined K decision tree in Random forest model using following equation.* $T(X, j) = \sum_{i-1}^{K} t_i(x,j)\,, (j = 1,2\,,3 \ldots \ldots m)$
Step 4	*Compute final outcome*

[4] UNSUPERVISED MACHINE LEARNING METHODS

This section presents the unsupervised machine learning techniques considered for software defect predication.

4.1 K-Means

K-means is one of popular unsupervised machine learning method [42]. This algorithm is initially applied to analysis the multi variety data. This algorithm is widely adopted to solve

large number of engineering problems such as clustering, feature selection, dimension reduction, image segmentation etc. This algorithm works with the predefined number of clusters. Moreover, the distance function is used to compute the closeness between data objects. The algorithmic steps of the k-means algorithm are given as.

Algorithm 4: Steps of K-means algorithm	
Step 1	*Initially upload the dataset and defined the number of clusters (K)*
Step 2	*Determine the K number of clusters from dataset in random order.*
Step 3	*Compute the objective function values i.e. sum of squared error using K number of clusters with each data object.*
Step 4	*Allocate the data objects to clusters .*
Step 5	*Compute the mean value of each cluster*
Step 6	*Update the initial cluster centres using new mean values*
Step 7	*Repeat the steps 3-6, until optimized results is not obtained*

4.2 K-harmonic Mean

K-harmonic mean is a variant of K-means algorithm [43]. In K-mean algorithm, arithmetic mean is used to compute the new cluster centres. Whereas, in K-harmonic mean, harmonic mean is used to compute the new cluster centres instead of arithmetic mean. The K-harmonic mean gives higher convergence rate as compared to K-means. The steps of K-harmonic mean algorithm is listed below.

Algorithm 5: Steps of K-Harmonic Mean algorithm	
Step 1	*Initially upload the dataset and defined the number of clusters (K)*
Step 2	*Determine the K number of clusters from dataset in random order.*
Step 3	*Compute the objective function values i.e. sum of squared error using K number of clusters with each data object.*
Step 4	*Allocate the data objects into clusters*
Step 5	*Compute the membership function and weight values for each data objects.*
Step 6	*Update the initial cluster centres using harmonic mean and weight of data objects.*
Step 7	*Repeat the steps 3-6, until there is no change in cluster centres.*
Step 8	*Obtain the optimized cluster centres.*

[5] RESULTS AND DISCUSSION

This section presents the experimental results of supervised and unsupervised machine learning approaches. The performances of these approaches are tested on a set of benchmark software defect datasets. The defect datasets are taken from the NASA and PROMISE repositories. These datasets are CM1, JM1, KC3, MC1, MC2, PC1, PC2, PC3 and PC4. These datasets are widely adopted to evaluate the performance of newly designed classifiers. Moreover, accuracy, precision and recall are considered as performance metrics to evaluate the performance of supervised and unsupervised machine learning techniques. The simulation results of supervised and unsupervised machine learning techniques are reported in tables 1-3 using accuracy, precision and recall parameters. Table 1 illustrates the results of bagging, AdaBoost, random forest, k-means and KHM techniques using all datasets. It is noticed that random forest technique providers better results than bagging, AdaBoost, k-means and KHM techniques. It is also observed that K-means technique exhibits the worst results among all techniques.

Table 1: Comparison of supervised and unsupervised machine learning techniques using accuracy parameter

Datasets	Bagging	AdaBoost	Random Forest	K-Mean	KHM
CM1	79	76	82	77	81
JM1	88	85	86	82	84
KC3	95	93	96	88	90
MC1	69	72	75	67	71
MC2	89	85	88	83	86
PC1	94	92	95	92	93
PC2	83	81	87	77	79
PC3	86	87	89	78	80
PC4	91	92	94	89	90
Avg. Accuracy	86	85	88	81	84

The simulation results of the precision parameter are reported in Table 2. It is seen that random forest technique have higher precision rate as compared to other techniques. Again, K-means technique obtains lesser precision rate in comparison to other machine learning techniques. Table 3 demonstrates the simulation results of the recall parameter. Recall parameter is inversely proportional to the precision parameters. This parameter also signifies the strength of the technique. If the value of recall parameter is higher than 0.5, then it is said that the particular technique is good for prediction. It is observed that the AdaBoost technique obtains higher recall values for defect prediction.

Table 2: Comparison of supervised and unsupervised machine learning techniques using precision parameter

Datasets	Bagging	AdaBoost	Random Forest	K-Mean	KHM
CM1	0.75	0.78	0.87	0.79	0.82
JM1	0.78	0.73	0.74	0.68	0.71

KC3	0.90	0.85	0.94	0.81	0.79
MC1	0.74	0.69	0.82	0.73	0.80
MC2	0.71	0.67	0.83	0.72	0.74
PC1	0.88	0.90	0.91	0.86	0.87
PC2	0.89	0.86	0.84	0.78	0.81
PC3	0.84	0.85	0.89	0.82	0.85
PC4	0.83	0.81	0.87	0.79	0.84
Avg. Precision	0.81	0.79	0.86	0.78	0.80

Table 3: Comparison of supervised and unsupervised machine learning techniques using recall parameter

Datasets	Bagging	AdaBoost	Random Forest	K-Mean	KHM
CM1	0.62	0.65	0.59	0.68	0.73
JM1	0.68	0.71	0.61	0.69	0.67
KC3	0.73	0.69	0.67	0.72	0.69
MC1	0.74	0.76	0.74	0.78	0.71
MC2	0.67	0.68	0.71	0.72	0.69
PC1	0.64	0.72	0.68	0.64	0.66
PC2	0.69	0.73	0.63	0.65	0.62
PC3	0.72	0.76	0.66	0.77	0.74
PC4	0.66	0.74	0.71	0.76	0.68
Avg. Recall	0.68	0.72	0.67	0.71	0.69

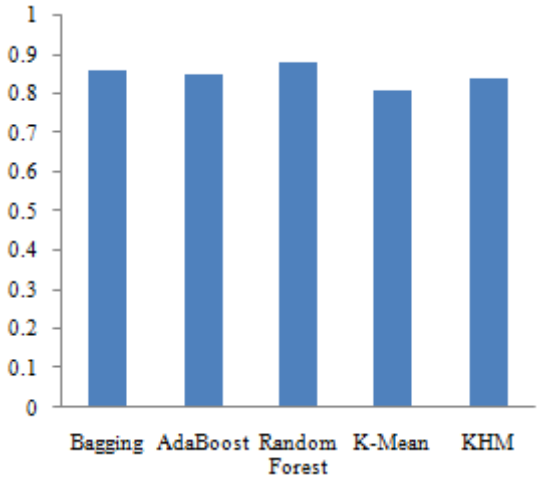

Fig. 1: depicts comparison of the average accuracy of supervised and unsupervised techniques

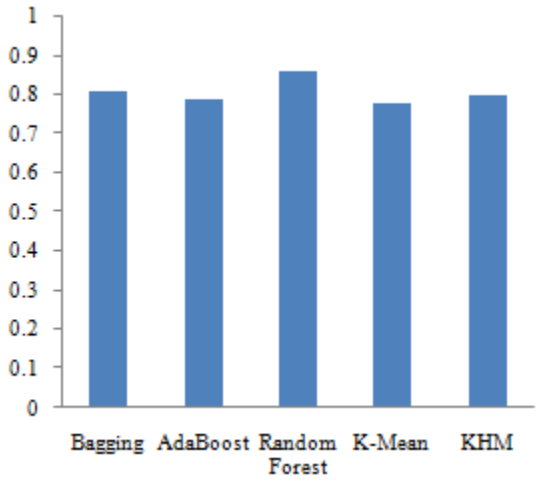

Fig. 2: depicts comparison of the average precision of supervised and unsupervised techniques

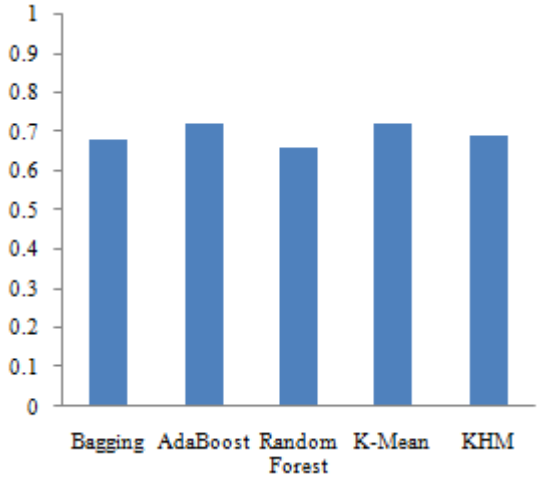

Fig. 3: shows the average recall comparison of supervised and unsupervised techniques

Figs. 1-3 show the graphical representation of the average accuracy, average precision and average recall parameter using bagging, AdaBoost, random forest, K-means and KHM techniques. Moreover on the analysis of Tables 1-3, it is concluded that the supervised machine techniques provides better performance than unsupervised machine learning techniques. Overall, random forest technique predicts more defects than all other technique. Whereas, among unsupervised machine learning techniques, KMH provides better result than K-means algorithm

[6] CONCLUSION

This paper presents the comparative analysis between supervised and unsupervised machine learning techniques and tries to identify which machine learning techniques are better for software defects prediction. The performance of these techniques is evaluated on nine benchmark software defect prediction dataset. These datasets are downloaded from NASA repository. In this work, accuracy, recall and precision metrics are considered to evaluate the performance of machine learning techniques. From the experimental results, it is noticed that random forest technique provides better results for most of defect prediction datasets as compared to other

techniques. Further, it is observed that among unsupervised techniques KMH provides good results for defect prediction. The simulation results also stated that the performance of supervised machine learning techniques is better than unsupervised machine learning techniques. Finally, it is concluded that supervised machine learning techniques are more suitable to indentify defects in software as compared to unsupervised machine learning techniques.

REFERENCES

[1] Catal, C. (2011). Software fault prediction: A literature review and current trends. Expert systems with applications, 38(4), 4626-4636.

[2] Yang, Y., Zhou, Y., Liu, J., Zhao, Y., Lu, H., Xu, L., & Leung, H. (2016, November). Effort-aware just-in-time defect prediction: simple unsupervised models could be better than supervised models. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 157-168). ACM.

[3] Fu, W., & Menzies, T. (2017, August). Revisiting unsupervised learning for defect prediction. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (pp. 72-83). ACM.

[4] Huang, Q., Xia, X., & Lo, D. (2017, September). Supervised vs unsupervised models: A holistic look at effort-aware just-in-time defect prediction. In 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 159-170). IEEE.

[5] Yu, X., Liu, J., Yang, Z., Jia, X., Ling, Q., & Ye, S. (2017, October). Learning from imbalanced data for predicting the number of software defects. In 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE) (pp. 78-89). IEEE.

[6] Chen, M., & Ma, Y. (2015). An empirical study on predicting defect numbers. In SEKE (pp. 397-402).

[7] Rathore, S. S., & Kumar, S. (2017). An empirical study of some software fault prediction techniques for the number of faults prediction. Soft Computing, 21(24), 7417-7434.

[8] Yan, M., Fang, Y., Lo, D., Xia, X., & Zhang, X. (2017, November). File-level defect prediction: Unsupervised vs. supervised models. In 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 344-353). IEEE.

[9] Rong, X., Li, F., & Cui, Z. (2016). A model for software defect prediction using support vector machine based on CBA. International Journal of Intelligent Systems Technologies and Applications, 15(1), 19-34.

[10] Mauša, G., & Grbac, T. G. (2017). Co-evolutionary multi-population genetic programming for classification in software defect prediction: An empirical case study. Applied soft computing, 55, 331-351.

[11] Öztürk, M. M., Cavusoglu, U., & Zengin, A. (2015). A novel defect prediction method for web pages using k-means++. Expert Systems with Applications, 42(19), 6496-6506.

[12] Ni, C., Chen, X., Wu, F., Shen, Y., & Gu, Q. (2019). An Empirical Study on Pareto based Multi-objective Feature Selection for Software Defect Prediction. Journal of Systems and Software.

[13] Xu, Z., Li, S., Luo, X., Liu, J., Zhang, T., Tang, Y., ... & Keung, J. (2019). TSTSS: A Two-Stage Training Subset Selection Framework for Cross Version Defect Prediction. Journal of Systems and Software.

[14] Malhotra, R., & Kamal, S. (2019). An Empirical Study to Investigate Oversampling Methods for Improving Software Defect Prediction using Imbalanced Data. Neurocomputing.

[15] Singh, P., Pal, N. R., Verma, S., & Vyas, O. P. (2016). Fuzzy rule-based approach for software fault prediction. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(5), 826-837.

[16] Chen, J., Yang, Y., Hu, K., Xuan, Q., Liu, Y., & Yang, C. (2019). Multiview Transfer Learning for Software Defect Prediction. IEEE Access, 7, 8901-8916.

[17] Balogun, A., Oladele, R., Mojeed, H., Amin-Balogun, B., Adeyemo, V. E., & Aro, T. O. (2019). Performance Analysis of Selected Clustering Techniques for Software Defects Prediction.

[18] Bowes, D., Hall, T., & Petrić, J. (2018). Software defect prediction: do different classifiers find the same defects?. Software Quality Journal, 26(2), 525-552.

[19] Chen, X., Zhang, D., Zhao, Y., Cui, Z., & Ni, C. (2019). Software defect number prediction: Unsupervised vs supervised methods. Information and Software Technology, 106, 161-181.

[20] Siers, M. J., & Islam, M. Z. (2015). Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. Information Systems, 51, 62-71.

[21] Ji, H., Huang, S., Lv, X., Wu, Y., & Feng, Y. (2019). Empirical Studies of a Kernel Density Estimation Based Naive Bayes Method for Software Defect Prediction. IEICE TRANSACTIONS on Information and Systems, 102(1), 75-84.

[22] Erturk, E., & Sezer, E. A. (2015). A comparison of some soft computing methods for software fault prediction. Expert systems with applications, 42(4), 1872-1879.

[23] Lamba, T., & Mishra, A. K. (2019). Optimal Machine learning Model for Software Defect Prediction. International Journal of Intelligent Systems and Applications, 11(2), 36.

[24] Ji, H., Huang, S., Wu, Y., Hui, Z., & Zheng, C. (2019). A new weighted naive Bayes method based on information diffusion for software defect prediction. Software Quality Journal, 1-46.

[25] Laradji, I. H., Alshayeb, M., & Ghouti, L. (2015). Software defect prediction using ensemble learning on selected features. Information and Software Technology, 58, 388-402.

[26] Li, W., Huang, Z., & Li, Q. (2016). Three-way decisions based software defect prediction. Knowledge-Based Systems, 91, 263-274.

[27] Liu, C., Yang, D., Xia, X., Yan, M., & Zhang, X. (2019). A two-phase transfer learning model for cross-project defect prediction. Information and Software Technology, 107, 125-136.

[28] Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. Applied Soft Computing, 27, 504-518.

[29] Marjuni, A., Adji, T. B., & Ferdiana, R. (2019). Unsupervised software defect prediction using signed Laplacian-based spectral classifier. Soft Computing, 1-12.

[30] Mesquita, D. P., Rocha, L. S., Gomes, J. P. P., & Neto, A. R. R. (2016). Classification with reject option for software defect prediction. Applied Soft Computing, 49, 1085-1093.

[31] Mori, T., & Uchihira, N. (2018). Balancing the trade-off between accuracy and interpretability in software defect prediction. Empirical Software Engineering, 1-47.

[32] Ryu, D., & Baik, J. (2016). Effective multi-objective naïve Bayes learning for cross-project defect prediction. Applied Soft Computing, 49, 1062-1077.

[33] Erturk, E., & Sezer, E. A. (2016). Iterative software fault prediction with a hybrid approach. Applied Soft Computing, 49, 1020-1033.

[34] Wang, T., Zhang, Z., Jing, X., & Zhang, L. (2016). Multiple kernel ensemble learning for software defect prediction. Automated Software Engineering, 23(4), 569-590.

[35] Wei, H., Hu, C., Chen, S., Xue, Y., & Zhang, Q. (2019). Establishing a software defect prediction model via effective dimension reduction. Information Sciences, 477, 399-409.

[36] Xu, Z., Liu, J., Luo, X., Yang, Z., Zhang, Y., Yuan, P., ... & Zhang, T. (2019). Software defect prediction based on kernel PCA and weighted extreme learning machine. Information and Software Technology, 106, 182-200.

[37] Yadav, H. B., & Yadav, D. K. (2015). A fuzzy logic based approach for phase-wise software defects prediction using software metrics. Information and Software Technology, 63, 44-57.

[38] Yousef, A. H. (2015). Extracting software static defect models using data mining. Ain Shams Engineering Journal, 6(1), 133-144.

[39] Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine learning, 36(1-2), 105-139.

[40] Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

[41] Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R news, 2(3), 18-22.

[42] MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

[43] Zhang, B., Hsu, M., & Dayal, U. (1999). K-harmonic means-a data clustering algorithm. Hewlett-Packard Labs Technical Report HPL-1999-124, 55.