A Comparative Study of Homogeneous and Heterogeneous Ensemble Techniques for Prediction of Software Faults

Ruchika Malhotra
Department of Software Engineering
Delhi Technological University
New Delhi, India
ruchikamalhotra@dtu.ac.in

Rohit Ramchandani
Department of Software Engineering
Delhi Technological University
New Delhi, India
ramchandani.rohit16@gmail.com

***Abstract* — Software defects have always been considered a major problem in the software industry and for software engineers, early detection improves software performance and reduces faults, time, and cost. The primary goal of this research is to assess the effect of Object-Oriented metrics on defect prediction performance using various classification and ensemble techniques. In this study, data preprocessing is performed using missing value treatment, mean imputation, and data is normalized using z score normalization, then feature selection is performed using correlation analysis followed by ANOVA. Imbalanced data treatment is performed using Random Over Sampling to consider removing the bias from prediction. Before applying classification to test data, KNN, DT, NB, LR, and RF parameters are tuned using 10-fold cross-validation on train data. In this study, ensemble models (boosting and voting) are built on NB, DT, and LR with saved parameters based on a comparison of the accuracy of KNN, NB, DT, and RF using 10-fold cross-validation on train data. The data was gathered from the PROMISE Repository. The accuracy-based analysis of prediction performance determined whether different classification and ensemble techniques have significantly different predictive performances. The accuracy of detecting software faults is significantly improved by the use of ensemble learning.**

***Keywords*—Faults, Predictions, Feature Selection, PROMISE, Imbalanced Data, Metrics, Classification, Ensemble Learning.**

I. INTRODUCTION

Software testing is a resource and time-consuming task in the software development lifecycle. The end goal of the testing process is to deliver error-free software that meets all stakeholder's requirements. Continuous modifications, severe deadlines, and the requirement to assure the correct functioning of functionality are some of the issues that developers encounter on a regular basis. However, limited time and workforce are threats to practical testing. Therefore, instead of testing the whole software for bugs, allocating all the resources to the bug-prone classes will be easier if we know them. Defects prediction and proneness in the software are always considered a major problem in the software industry and for software engineers. In classical methods, previous experience with the defective or non-defective software is required while detecting the software defect in a software application. [1] Defects in the software applications have been predicted by using different classification techniques by different researchers as of now but the results of the methods changed with the dataset, so they lack the property of robustness for defect prediction in an unknown software application, However, ensemble techniques for defect detection in software applications will be very efficient, as there will be an advantage of using various techniques on a specific dataset to predict software defects with much more accurate than using a single technique, for the reasons stated above.[2] An ensemble learning software defect prediction model will allow the software to significantly predict and rectify software defects using soft voting and hard voting. This ability makes the software run more effectively and reduces faults, time, and cost. [3] Defect Prediction Model, which predicts that the software application which needs to be tested more extensively and is more likely to have defects, offers an effective solution to the above problem. Software defect prediction is one of the most assisting activities in the Testing Phase of the software development life cycle. Software defect prediction models use multiple software metrics data collected either from previous versions of the same system (within-project approach) or from the metric data of other systems (cross-project approach).

The software defect prediction model predicts the software application that needs to be analyzed more for the defects as the defect probability is higher in that component. The defect prediction model is a supervised learning method in which the dependent variable (defects i.e. true or false) are predicted using a set of independent variables selected also called predictors using one or more ML Classifiers.

In this study, a balanced dataset for software defect prediction is built, and parameter tuning is also done for the base learners to build a robust ensemble algorithm for software defect detection. A heterogeneous ensemble learning model (boosting and voting) is introduced and the comparison of the ensemble model is made with all the classifiers i.e., DT, LR, KNN, NB which have been used to build the model, and the homogenous

ensemble learning model i.e., RF. In this paper, accuracy has been used so to determine whether the ensemble algorithm is accurate or not in determining whether the software is defective or not defective.

The following is how this paper is structured: Section 2 features related work. The methodology is presented in Section 3. Section 4 discusses the findings, and Section 5 provides the conclusion.

II. RELATED WORK

A lot of research has been done to build software fault prediction models using different fault prediction techniques such as Machine Learning techniques and ensemble learning techniques.

Aleem *et al*. [4] analyzed and compare the prediction outcomes of 11 machine learning techniques such as Multilayer Perceptron, NB, SVMs, Ada Boost, Bagging, RF, etc. using 15 NASA datasets that were downloaded from the PROMISE repository. The result of this study shows that Bagging and SVM give better prediction performance. A study done by Elish, Aljamaan, and Ahmad [5] shows that the ensemble method gives correct prediction results on the considered dataset. They analyzed the ensemble approaches for predicting change efforts and maintenance of software using two publicly available datasets. A study was done by Perreault *et al.* [6] on five NASA datasets compared Artificial Neural Networks, LR, NB, SVMs, and KNN however, there is no clear explanation as to which strategy is the most effective. Hussain *et al*. [7] in their study compared the three ensemble techniques that use five base classification techniques such as LR, NB, J48, Voted-Perceptron, and SVM in the Weka tool for software defect prediction. This study shows that Stacking gives better results than all the ensemble techniques used. Ghotra *et al.* [8] established that the accuracy of a prediction model can increase or decrease up to 30% depending upon the classifier used. Also, Panichella *et al.* [9] demonstrate that despite comparable prediction accuracy, the predictions of different classifiers are highly interconnected The model can be trained to utilize massive volumes of data from the project under observation (within-project strategy) or data from a similar but unobserved project (cross-project strategy).

III. RESEARCH METHODOLOGY

We have used the following steps:

- Obtain datasets with object-oriented metrics from publicly accessible repositories.
- Perform some preprocessing operations on datasets.
- Choose performance evaluation measures to evaluate the prediction performance.
- Choose some classification and ensemble techniques.
- Analyze the performance based on chosen evaluation measure.

A. Collection of Datasets

The dataset is collected from the publicly available PROMISE repository. We chose NASA project datasets JM1, PC1, and CM1 since they contain object-oriented metrics. Table I lists the datasets that were used in this study:

Table I. Datasets chosen

Project Name	Number of instances	True	False
JM1	10,885	8779	2106
CM1	498	49	449
PC1	1109	1032	77

Table II has a complete description of object-oriented metrics.

Table II. Metrics Description

Metrics Name	Type
(loc)	McCabe's Line of Code metric
(v(g))	McCabe's cyclomatic complexity metric
(ev(g))	McCabe's essential complexity metric
(iv(g))	McCabe's design complexity metric
(n)	Halstead's operators and operands measure
(v)	Halstead volume measure
(l)	Halstead program length measure
(d)	Halstead difficulty measure
(i)	Halstead intelligence measure
(e)	Halstead effort measure
(t)	Halstead time estimator measure
(lOCode)	Halstead line count measure
(lOComment)	Halstead comments and comment measure
(lOBlank)	Halstead blank line count measure
(locCodeAndComment)	Miscellaneous Metrics
(uniq_Op)	
(uniq_Opnd)	
(total_Op)	
(total_Opnd)	
(branchCount)	
(b)	Halstead numeric metrics
Defects	Defective or not defective module

B. Dataset Preprocessing

(i) Missing Value Treatment

Datasets obtained from publicly available repositories have some missing values, which impair the performance of the created model, hence preprocessing is done on datasets to avoid this type of problem, such as eliminating data points with missing values because they are extremely small in count.

(ii) Mean Imputation

In the independent variables, the fields of the attribute which cannot be 0 for e.g. loc, having 0 values have been replaced

with the mean of the attribute of the particular field i.e. mean of loc.

(iii) Normalization

Normalization technique selection is user specified in accordance with the data. In this comparative study of models for the dataset considered, 21 input variables are not on the same scale and not in the same range of magnitude. Hence, data normalization using mean centralizing and variance scaling of the training data is done which scales the data in the range [-1,1]. The equation for z score normalization is given below that results in avoiding the dominance of the features which have large values when algorithms based on distances are used such as KNN.

$$z = \frac{x - mean.x}{standard\ dev.x}$$

where, x denotes the data point.
mean.x denotes the mean of attribute of x.
standard dev.x denotes the standard deviation of attribute of x.

(iv) Feature Selection

Feature selection methods have been used to get only the relevant features i.e. the features which contribute significantly to the classification.

(a) Karl Pearson Correlation Coefficient(r)

It is used to find the correlation between the independent variables.

$$r = \frac{n\,(\sum xy) - n(\sum x)(\sum y)}{\sqrt{([n(\sum x^2 - (\sum x)^2)]\,[n(\sum y^2 - (\sum y)^2)]\,)}}, r \in [-1, 1]$$

Variables that have a correlation greater than 0.9 have been eliminated.

(b) ANOVA

Analysis of Variance technique is used after the removal of features using correlation since the input variables are numerical and the output variable is categorical. Based on the F-score of the remaining features and the p-value, total 5 best features are selected with a high F-score from the remaining features since the total ⌈log2(n)⌉ number of features should be used, where n is the total number of features for the classification of data points [10].

(v) Imbalanced Data Treatment

There is an imbalance class problem in the datasets chosen. The imbalanced data identification is done by taking the ratio of a number of Non-Defective and Defective Classes in the train set since the ratio should be nearly equal to 1: 1 for balanced data, but it is 4: 1, 9: 1, and 13: 1 for JM1, CM1, and PC1 respectively for the train data, so random oversampling of the data is done to balance it. In random oversampling, the weightage of the class which is low in the count is increased by repetition of that class's data points randomly. The defective class is very low in count in all the datasets used, so it should be given more weightage by repetition to balance the dataset, if the ratio of non-defective to defective class is X: 1 defective class's data point will be repeated X times to balance the data and to avoid biasness of the majority class. Random Over Sampling of the minority class is done (only during the training of the model). Figure 1 shows imbalanced data problems in JM1, CM1, and PC1 datasets and the balancing of the data in JM1, CM1, and PC1 respectively.

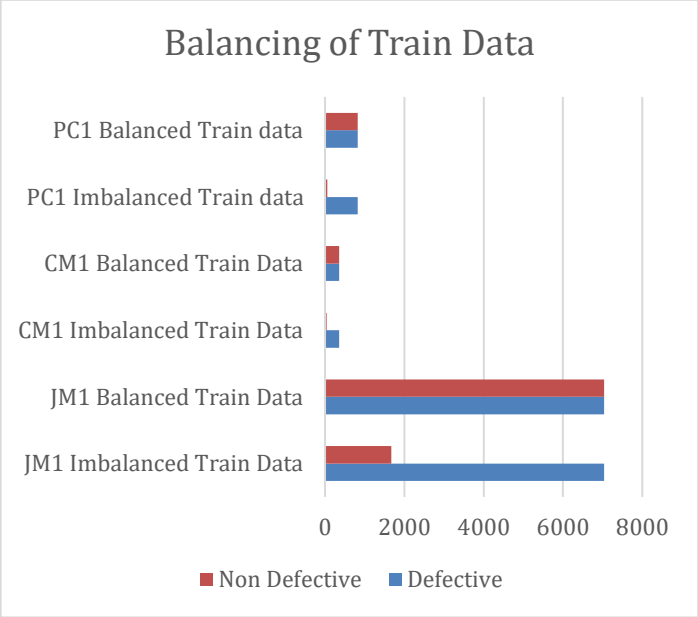

Fig1. Balancing of Dataset using Random Over Sampling

C. Techniques Used

Classification Techniques

The process of predicting class or categorical data from a set of independent data is known as classification. Mathematically we can say that classification is the mapping of a function from an input variable (independent variable) to an output variable (dependent variable). In this study, we use NB [12], LR [13], DT [14], SVM [15], KNN [16], RF [17], and MLP [18] technique for developing the prediction model.

Ensemble Techniques

The objective is to combine the prediction outputs of various learning approaches so that the overall performance of the decision is enhanced over the output of the individual techniques [19]. The ensemble model improves the performance of the individual model for example it improves the performance of the decision tree by reducing variance in the model. They are classified as either homogeneous or heterogeneous ensembles. Learning techniques of the same type, such as bagging, boosting, and so on, are used in a homogeneous ensemble [20]. Different types of learning

techniques are used in heterogeneous ensembles. We built a defect prediction model using voting, homogeneous boosting, and heterogeneous boosting in this study.

D. Performance Evaluation Measure

We employ accuracy, to evaluate the prediction performance of machine learning techniques and ensemble techniques. It is the proportion of the total correct outcomes to the total number of outcomes [11]. Total correct outcomes are the sum of true positive and true negative values. A total number of outcomes is the sum of true positive, true negative, false positive, and false negative values. We can find values of true positive, true negative, false positive, and false negative with the help of a confusion matrix.

E. Experimental Results

This section demonstrates the experimental result found after applying machine learning and ensemble learning techniques to 3 considered datasets.

1) Table III represents the accuracy values for JM1, CM1, and PC1 datasets for all machine learning (NB, KNN, DT, SVM, MLP, LR, RF) and ensemble learning techniques (homogeneous (boosting) and heterogeneous (boosting and voting)). Results from Table III show that a homogeneous ensemble outperforms the predictive performance of the individual machine learning technique. Heterogeneous ensemble (voting and boosting) outperforms the predictive performance of homogeneous ensemble techniques and individual machine learning techniques.

Table III. Accuracy values for the JM1, CM1, and PC1 datasets for all machine learning and ensemble learning techniques.

Accuracy	JM1	PC1	CM1
KNN	73	86	85
NB	79	82	85
LR	73	79	73
DT	74	91	80
SVM	72	82	70
MLP	70	91	73
RF	77	92	88
Boosting DT	77	94	89
Boosting NB	78	92	89
Boosting LR	75	80	73
Boosting SVM	76	88	85
Heterogeneous Ada Boost	78	94	84
Voting	79	89	89

IV. RESULTS AND DISCUSSION

In this section, we analyze the predictive performance of machine learning techniques and ensemble learning techniques on the basis of 3D bar graphs. Fig 4. shows 3D bar graph of accuracy values for all the datasets using machine learning techniques and ensemble learning techniques. This graph shows that heterogeneous ensemble outperforms homogeneous ensemble and homogeneous ensemble outperforms individual machine learning techniques.

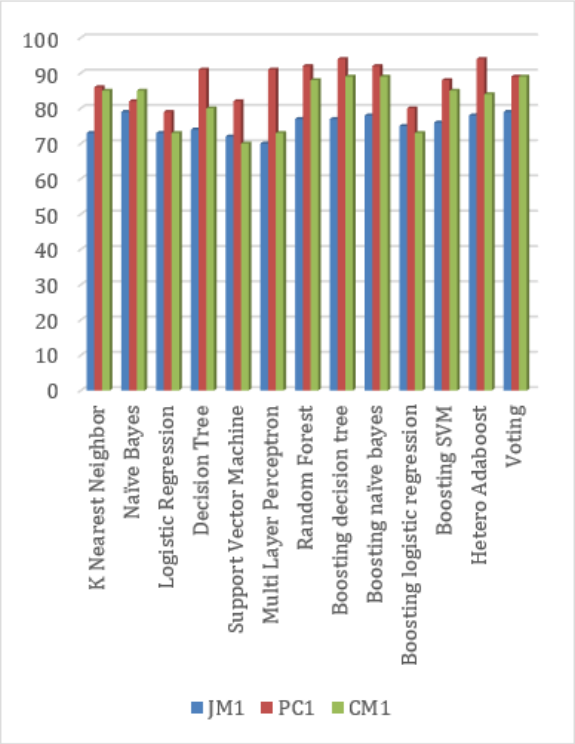

Fig. 2 3D bar graph of all datasets using machine learning and ensemble learning techniques.

V. CONCLUSION

In this study, we have analyzed the predictive performance of classification, homogeneous and heterogeneous ensemble learning techniques for predicting the software to be defective or not defective. The dataset utilized is from the PROMISE Repository and includes various Object-Oriented Metrics as independent and dependent variables with dichotomous class labels of defective and non-defective. In this study for data preprocessing missing value treatment, mean imputation and data normalization using z score normalization is used which keeps the value in the range [-1,1], further the feature selection technique used is correlation analysis and ANOVA, followed by imbalanced data treatment using Random Over Sampling on train data. The performance of the techniques is evaluated using Accuracy. The findings reveal that different approaches perform differently on various datasets. From Figure 4 and Table III, it can be concluded that heterogeneous ensemble outperforms both homogeneous ensemble and individual classification machine learning techniques since it gives better

accuracy for software fault prediction. More research will be conducted in the future to evaluate other feature selection approaches. Other datasets can also be used in the future.

VI. REFERENCES

[1] Md. R. Ahmed, Md. A. Ali, N. Ahmed, Md. F. bin Zamal, and F. M. J. M. Shamrat, "The Impact of Software Fault Prediction in Real-World Application: An Automated Approach for Software Engineering," in *Proceedings of 2020 the 6th International Conference on Computing and Data Engineering*, 2020, pp. 247–251. doi: 10.1145/3379247.3379278.

[2] S. S. Rathore and S. Kumar, "Towards an ensemble based system for predicting the number of software faults," *Expert Systems with Applications*, vol. 82, pp. 357–382, 2017.

[3] D. di Nucci, F. Palomba, R. Oliveto, and A. de Lucia, "Dynamic selection of classifiers in bug prediction: An adaptive method," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 3, pp. 202–212, 2017.

[4] S. Aleem, L. F. Capretz, and F. Ahmed, "Benchmarking machine learning technologies for software defect detection," *arXiv preprint arXiv:1506.07563*, 2015.

[5] M. O. Elish, H. Aljamaan, and I. Ahmad, "Three empirical studies on predicting software maintainability using ensemble methods," *Soft Computing*, vol. 19, no. 9, pp. 2511–2524, 2015.

[6] L. Perreault, S. Berardinelli, C. Izurieta, and J. Sheppard, "Using classifiers for software defect detection," in *26th International Conference on Software Engineering and Data Engineering*, 2017, pp. 2–4.

[7] S. Hussain, J. Keung, A. A. Khan, and K. E. Bennin, "Performance evaluation of ensemble methods for software fault prediction: An experiment," in *Proceedings of the ASWEC 2015 24th Australasian software engineering conference*, 2015, pp. 91–95.

[8] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, vol. 1, pp. 789–800.

[9] A. Panichella, R. Oliveto, and A. de Lucia, "Cross-project defect prediction models: L'union fait la force," in *2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, 2014, pp. 164–173.

[10] H. Wang, T. M. Khoshgoftaar, J. van Hulse, and K. Gao, "Metric selection for software defect prediction," *International Journal of Software Engineering and Knowledge Engineering*, vol. 21, no. 02, pp. 237–257, 2011.

[11] E. A. Felix and S. P. Lee, "Predicting the number of defects in a new software version," *PLoS One*, vol. 15, no. 3, p. e0229131, 2020.

[12] I. Rish and others, "An empirical study of the naive Bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 2001, vol. 3, no. 22, pp. 41–46.

[13] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on software engineering*, vol. 22, no. 10, pp. 751–761, 1996.

[14] Y. Zhao and Y. Zhang, "Comparison of Decision Tree methods for finding active objects," *Advances in Space Research*, vol. 41, no. 12, pp. 1955–1959, 2008.

[15] R. Malhotra and A. Bansal, "Use of Support Vector Machine to Check Whether Process Metrics are as Good as Static Code Metrics," in *Topical Drifts in Intelligent Computing: Proceedings of International Conference on Computational Techniques and Applications (ICCTA 2021)*, 2022, vol. 426, p. 35.

[16] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans Inf Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[17] Z. Masetic and A. Subasi, "Congestive heart failure detection using random forest classifier," *Comput Methods Programs Biomed*, vol. 130, pp. 54–64, 2016.

[18] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, and others, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[19] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, 2000, pp. 1–15.

[20] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. de Sousa, "Ensemble approaches for regression: A survey," *Acm computing surveys (csur)*, vol. 45, no. 1, pp. 1–40, 2012.