International Journal on
"Technical and Physical Problems of Engineering"
(IJTPE)
Published by International Organization of IOTPE

ISSN 2077-3528
IJTPE Journal
www.iotpe.com
ijtpe@iotpe.com

December 2021 | Issue 49 | Volume 13 | Number 4 | Pages 63-71

ENHANCED HETEROGENEOUS ENSEMBLE TECHNIQUE FOR IMPROVING SOFTWARE FAULT PREDICTION

J. Goyal Jain B. Kishan

Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, India
jyoti.goyal24@gmail.com, balkishan248@gmail.com

Abstract- Nowadays, the software is utilized in almost every field of life, including health, industry, automobiles etc. Thus, it is imperative that the software must perform as expected. It is only possible if the software is free of faults. Software fault prediction is used to predict faults in the early stage of the software development life cycle. As a result, errors can be corrected immediately with minimal resources. Due to the versatile nature of software, it becomes very difficult for any single technique to predict diverse faults. Numerous research studies exist in the literature that predicts diverse faults using ensembling, but none of the studies detect faults of lower magnitude, which constitute the principal cause of failure. So, in this study, we propose a heterogeneous ensemble model with stacking for predicting those diverse faults. Moreover, features play a dominant role in the prediction of diverse faults. Consequently, we use a novel SHAP feature selection technique that selects the features based on their local interpretability. The experimental work is implemented in Python on Google Colab. The results show that the proposed model detects diverse faults based on the SHAP value of features and gives accuracy of more than 95% in almost all datasets.

Keywords: Fault Prediction, SHAP Feature Selection, Heterogeneous Ensembling, SMOTE TOMEK.

1. INTRODUCTION

Technology advances at a rapid pace, resulting in better application products. High-quality software products always demand a great deal of software testing[1]. In the early days, manual testing was used to evaluate the quality of software, but this was a very complex process because it takes a lot of time and effort. As a result, software fault prediction is introduced that predicts fault-prone modules before their occurrence. It helps the testing team to focus only on faulty modules. This will save the scarce testing resources that are time, effort, and money. Software metrics are the measurable properties of software that helps to detect fault-prone modules. Different software metrics exist in the literature that depends on the language in which software is developed[2]. Predicting faulty modules using machine learning techniques is a hot topic in recent research studies.

Various techniques have been used by researchers for predicting faults i.e., NB, SVM, and DT. Researchers proved that predicting faults using ensembling is always has good prediction results than individual techniques[3].

Class imbalance is an important consideration in fault prediction studies. Class imbalance arises when the majority class dominates the minority class in a dataset, which is common in defect datasets. This imbalance could cause classification models to be biased towards the main class (non-defect class), lowering prediction accuracy. In this issue, several solutions have been presented in [4, 5].

Feature engineering is another crucial issue that has a significant impact on the model's performance. Some datasets suffer from the problem of multi-collinearity. To solve this problem, we need to remove highly correlated features. Repetitive characteristics, on the other hand, lengthen training time and may even overfit classification models. Ref. [6] in his study found that the same classifier with different features will give different results. So, feature selection plays a crucial role in fault prediction models. Many features selection and feature engineering techniques are available in the literature. But all these existing techniques select the features based on global interpretability and there is no transparency regarding how much each feature contributes either positively or negatively to predict outcome [7]. So, to alleviate this consideration, we implement a novel SHAP feature selection technique that resolves this problem "unpublished" [8].

The motive of this study is to propose and implement a heterogeneous ensemble fault prediction model based on stacking that deal with all these issues and exhibit prediction results that are better than existing fault prediction frameworks. The reason for using heterogeneous ensembling is the diversity feature because each different classifier is having a unique ability to detect different faults. Due to the small magnitude, those unique defects are not counted in traditional frameworks such as ensembling based on voting [9]. So, this stacking technique will consider those unique faults which are small in magnitude.

The main contributions of this research work are:

1. Implementation of heterogeneous ensemble model based on stacking.

2. Handling class imbalance using hybrid technique SMOTE + Tomek.
3. Implementation of novel SHAP features selection technique.
4. Statistical tests is performed to compare the proposed model with standalone base classifiers.

The remaining section of this research work is laid out in the following way: Section 2 presents the recent ensemble based fault prediction studies; section 3 discusses the materials and methods being used. Section 4 includes a detailed explanation of the experimental results. The statistical significance and threats to the validity of the proposed model are presented in sections 5 and 6; in section 7 we discuss the conclusion and future scope.

2. LITERATURE REVIEW

In this section, the recent research work based on ensembling for the prediction of faults will be discussed. A recent study was performed where SMPSO-HS-AdaBoost, an intelligent fusion technique is proposed that combines sampling, feature selection, and classification approaches together. To alleviate the feature redundancy problem, the researcher employs AdaBoost classification based on hybrid sampling and particle swarm optimization based on subgroup migration feature selection [10].

In another study, a solution for the filter rank selection problem where multiple diverse filter methods are applied independently to obtain the required result. DT and NB are used as a classifier to implement the proposed filter technique. The result proves that the proposed model performs better than the traditional rank-based models [11]. A study where Bootstrap aggregating ensemble learning technique for software defect prediction is proposed. This technique is implemented on object-oriented modules. The accuracy, recall, precision, F-measure, and AUC-ROC efficiency of the suggested technique are compared to those of several competent machine learning algorithms.

The proposed strategy out performed existing approaches based on simulation results and performance comparisons [12].

Another sequential ensemble model for the prediction of software faults was presented. Eight datasets of the promise data repository are used to empirically evaluate the results of the proposed model. The results are better than the established models [13]. A comparative study was performed, where twenty-one classifiers from five categories are applied to five open source applications to find the best classifier with Object-Oriented metrics. MATLAB's classification Learner App was used to test multiple classification models. Bagging trees and SVM are found to be the best predictors among twenty-one classifiers [14].

Another fault prediction framework that employs the Multi-Filter feature selection technique and the MLP as the classifier was proposed. The results are calculated both by using and without using over sampling technique. The results proved that the framework using oversampling technique provides better results than without oversampling [15]. A comparative study was performed where the classification results using random forest are compared with SVM, backpropagation NN, and D Trees. Random forest is performing better than other classifiers [16].

3. MATERIAL AND METHODS

3.1. Datasets Description

Six available benchmark datasets from PROMISE Data Repository and GitHub [17]are used to aid replication and verification of our investigations. These datasets are gathered from real NASA software programs that are based on spacecraft instruments, storage management, and soon. These datasets are written in C or C++ language. Table 1 provides a full description of these datasets.

Table 1. Description of datasets

Name of dataset	Language	# of Attributes	# of Modules	# of Non Defects	# of Defects	% of Non Defects	% of Defects	Imbalance Ratio
ar1	C	31	121	112	9	93%	7%	8.04%
Camel 1.6	C++	24	965	776	189	80%	20%	24%
MW1	C	38	253	226	27	89%	11%	12%
PC1	C	22	1109	1032	77	93%	7%	7%
PC2	C	37	745	729	16	98%	2%	2%
Xerces 1.3	C++	24	453	241	212	53%	47%	88%

3.2. Methodology

In this section, we discuss the methodology we use to implement our proposed framework mentioned in the paper [18]. The success of any machine learning model depends upon the data quality we use to train our models [19]. So, our first task is to pre-process the data by handling missing values, discretizing categorical features, and handling outliers [20, 21]. After that, we check whether the dataset is balanced or not. If the data is unbalanced, we oversample minor labels to make the data more balanced.

To combat with class imbalance in our research, we apply the SMOTE + Tomek approach. Then we use chi-square, manual correlation, and a novel SHAP feature selection technique to select features. The Min-Max scaling approach is used to normalise the features. Then, to predict faulty software modules, we use our suggested stacking-based heterogeneous ensembling technique. All these experiments are conducted on Goggle Colab using python. The framework of our model is described in Figure 1.

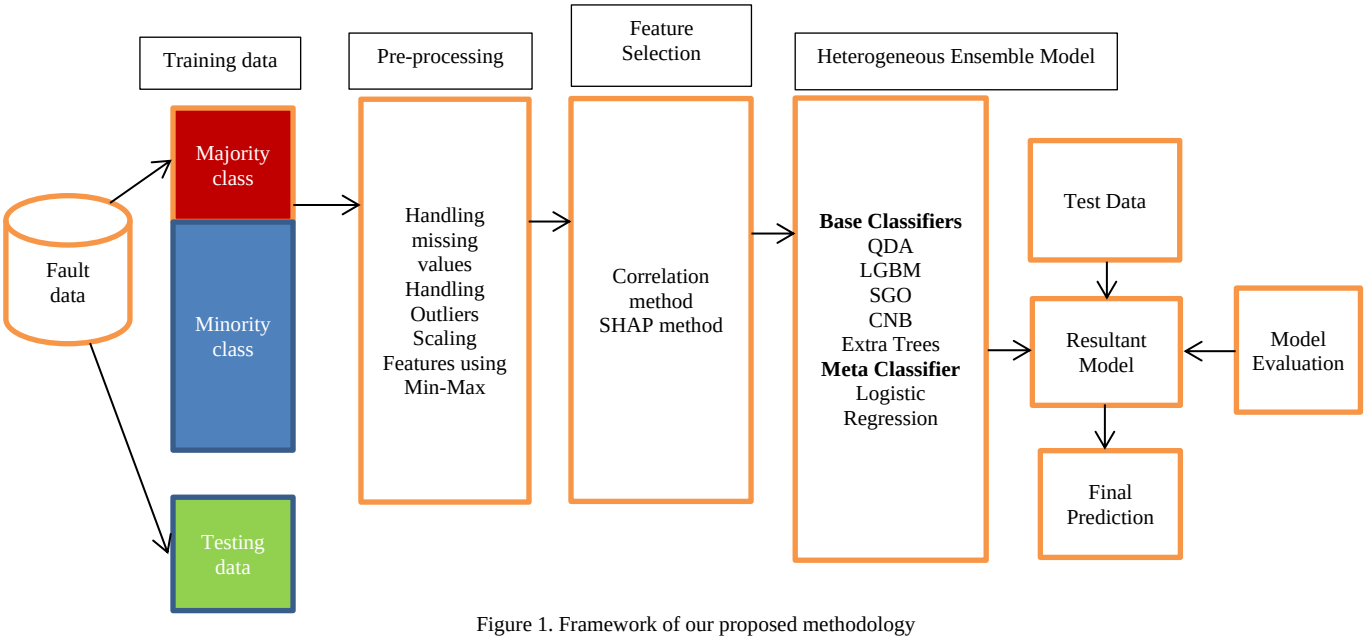

Figure 1. Framework of our proposed methodology

The following section briefly describes the tasks we perform for the implementation of our model:

3.2.1. Data Pre-Processing Step

Data pre-processing is one of the crucial steps of any fault prediction model. The work seems half done when the pre-processing step is finished. The key tasks we undertake as part of data pre-processing are mentioned below.

3.2.1.1. Handling Missing Values and Feature Scaling

Almost all real-world data have some missing value columns. After analysing our fault dataset, we also find some missing value columns; to fill these missing values we replace them with the mean of that column. The distribution of values of attribute total_loc of dataset ar1 is mentioned in Figure 2. These feature values are of different magnitude and the feature that has a high magnitude will dominate the prediction result, so we normalize those features using the min-max scaling technique to take all features on the same scale. The values of features after scaling are mentioned in Figure 3.

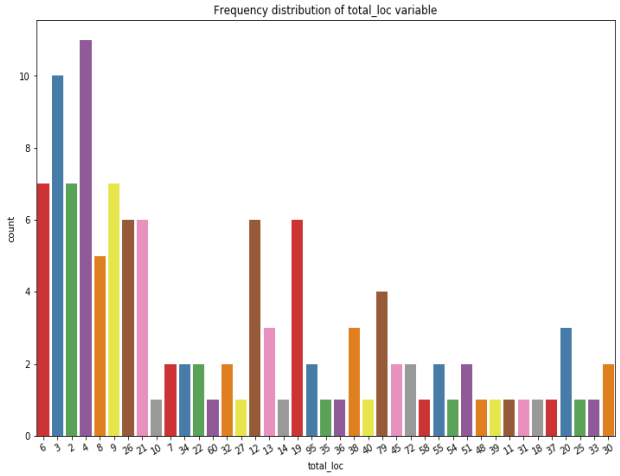

Figure 2. Frequency distribution of total_loc of dataset ar1

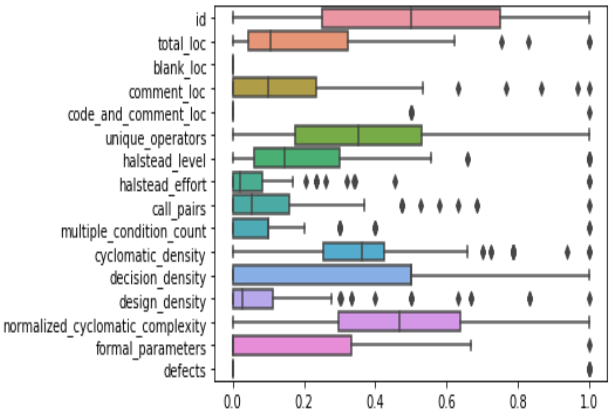

Figure 3. Magnitude of features after min-max scale

3.2.1.2. Handling Outliers

All values that are highly dispersed from the mean value are called outliers and it negatively affects the performance of the model. Figure 4 signifies that almost all features are having outliers. To remove those outliers, we replace the outlier values with the median of that attribute. The attribute hastead_effort after the removal of outliers is shown in Figure 5.

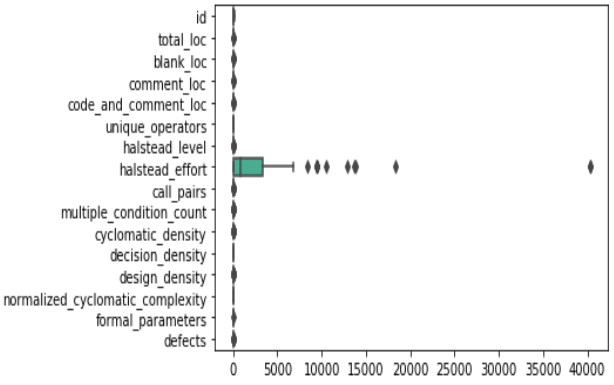

Figure 4. Boxplot of features of dataset ar1

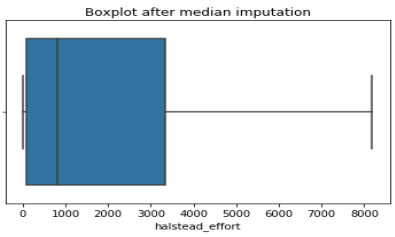

Figure 5. Boxplot of halstead_effort after outlier removal

3.2.1.3. Handling Class Imbalance

The majority of the datasets available for software fault prediction are highly imbalanced. The distribution of defects of dataset ar1 is mentioned in Figure 6. To balance data sets, we use a hybrid resampling methodology called SMOTE + Tomek "unpublished" [22]. The distribution of class labels after performing SMOTE + Tomek is shown in Figure 7.

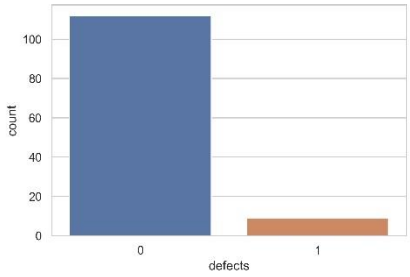

Figure 6. Distribution of defects of dataset ar1

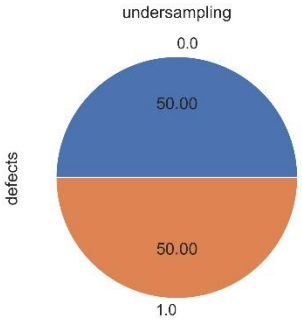

Figure 7. Defects distribution after SMOTE + Tomek

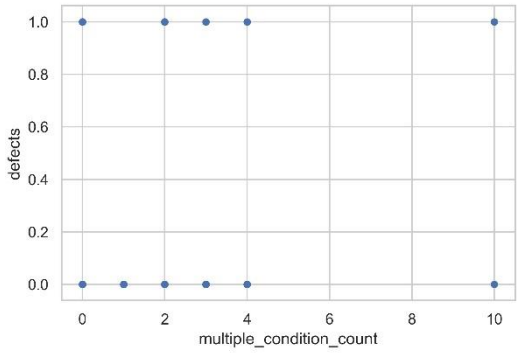

Figure 8. Relation between mcc and defects

3.2.2. Feature Selection

Feature selection is the process of identifying the most important characteristics from a dataset and uses those features for predicting faults. Based on Figure 8, there is no relation between multiple_condition_count and defects. A huge number of irrelevant features exist in the data that will exponentially increase training time and raises the risk of over fitting models [23]. In this research work, we have used three different feature selection techniques i.e., chi-square technique, correlation-based technique, and a novel SHAP feature selection technique to find optimal features to give better results. These are briefly discussed below.

3.2.2.1. Correlation-Based Feature Selection Technique

CFS [24] is a well-known technique for determining the importance of features by calculating the correlation between features and target class, as well as between features and other characteristics. Figure 9 shows the correlation matrix of dataset ar1. The yellow section shows the highly correlated features.

3.2.2.2. SHAP Technique

SHapley Additive exPlanations technique is the novel technique for feature selection. It increases the transparency in the prediction process. The major benefits of SHAP are global interpretability means it helps us to know the contribution of each predictor in the whole population, local interpretability where it finds the predictor's contribution on each observation. The third benefit is the SHAP value can be identified for any tree-based model. Figure 10 describes the contribution of each feature based on the SHAP value. The ranking of features is mentioned in Figure 11. The red line shows the positive contribution of the feature where the blue line shows the negative contribution. The contribution of features on first observation of dataset ar1 is presented in Figure 12.

3.2.3. Fitting Base Models and Meta-Models

In our research work, we have selected the base models based on their diversity; it means they belong to different classification categories. Therefore, their prediction capability will be different. Each base classifier will detect unique and different kinds of faults. The base classifier we select is described below.

3.2.3.1. Quadratic Discriminant Analysis (QDA)

QDA is always attractive because it finds solutions for those classification problems that are not linearly separable. It works on multiclass problems and provides better predictions. It needs no hyper-parameter tuning [25].

3.2.3.2. Light Gradient Boosting Method (LGBM)

Light GBM is a tree-based gradient boosting method that can be used for classification and other machine learning problems. It gives higher accuracy than another boosting algorithm because it splits the tree leaf-wise. Therefore, it gives fast results and is named "Light", "unpublished" [26].

3.2.3.3. Stochastic Gradient Descent (SGD)

Stochastic gradient descent abbreviated as SGD is an iterative algorithm used to find the optimal parameters for the model. It works better on the massive dataset and gives fast results than gradient and batch gradient descent [27].

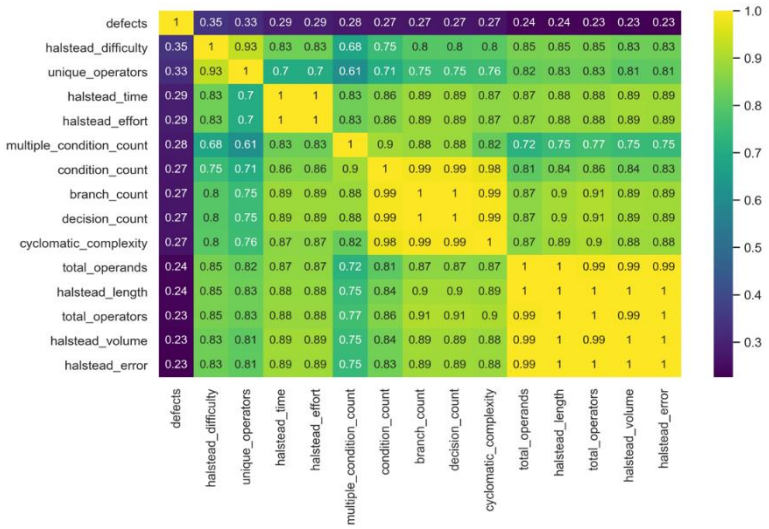

Figure 9. Correlation matrix of dataset ar1

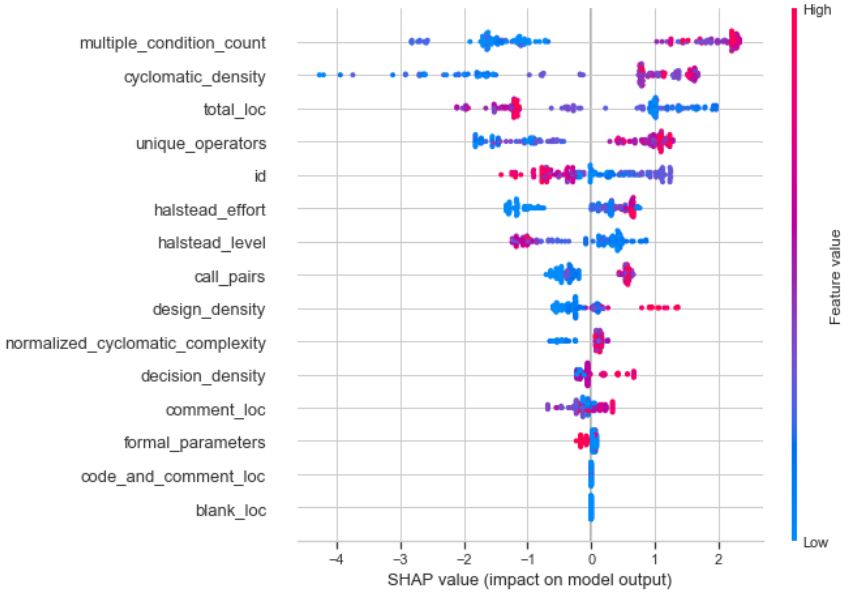

Figure 10. Impact of each feature on fault prediction

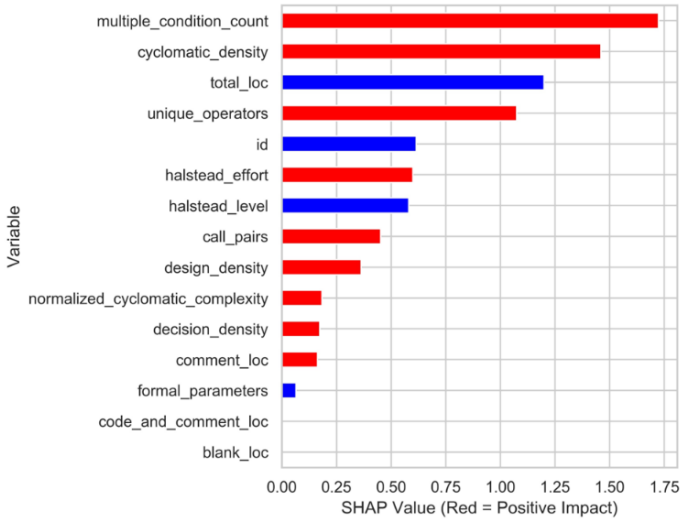

Figure 11. Ranking of features based on SHAP value

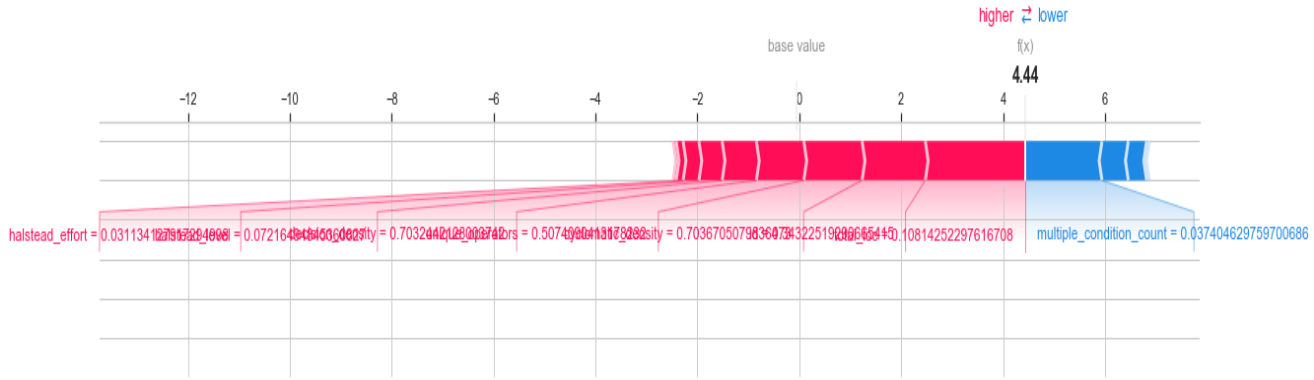

Figure 12. Features contribution on observation 1

3.2.3.4. Complement Naive Bayes (CNB)

Complement Naive Bayes works better with imbalanced datasets. As the name suggests, for each class it finds the probability of observation. So the smallest value indicates the highest probability of observation belonging to that class "unpublished" [28].

3.2.3.5. Extremely Randomized Trees Classifier (Extra Tree Classifier)

Extra Trees Classifier is a form of ensemble learning algorithm that outputs a classification result by aggregating the results of several de-correlated decision trees collected in the "forest" [29].

3.2.3.6. Meta-Model

Currently, the most prevalent learning strategy as a meta-classifier is logistic regression [30]. So, in our work, we also use logistic regression as a meta-classifier.

3.3. Performance Evaluation Step

Performance evaluation is the important step for accessing the performance of any defect prediction model. In this research work, the following performance metrics are used to evaluate the efficiency of the model. All these metrics are defined in [31].

3.3.1. Accuracy

Accuracy is the base metric for model evaluation. It works better only if the data are balanced.
Accuracy (all correct / all) = $TP+TN / P + N$

3.3.2. Precision

Precision identifies the actual faulty modules among predicted modules.
$Precision = TP/ TP+FP$

3.3.3. Recall

Recall identifies the actual faulty modules among the faulty modules.
$Recall = TP / P$

3.3.4. F1-Score

F-measure is calculated based on precision and recall.

3.3.5. MCC

It is a measure of the quality of binary classifications.

4. RESULTS AND DISCUSSION

The outcomes of the experiments are described in this section. The results obtained by implementing our model are depicted in tables and it shows that our proposed model gives better results in almost all datasets. Different performance metrics are used to evaluate the performance of the model like accuracy, precision, recall, MCC, f1-score, and ROC AUC. The following Table 2, 3, 4, 5, 6, and 7 depicts the scores of these metrics on dataset ar1, camel 1.6, mw1, pc1, pc2, and xerces. The ROC AUC scores of these datasets are mentioned in Figures 13, 14, 15, 16, 17, and 18.

As we can see the accuracy of the proposed model is more than 90% in all datasets except mw1 and xerces. The MCC value of our model is higher in dataset ar1. The precision is also more than 90%, except for pc2 and xerces. The following figures shows that the ROC AUC value is 1 in ar1 and camel 1.6. The value of 0.99 is achieved in pc1 and pc2. So, the implemented heterogeneous model based on stacking performs better than all existing fault prediction models and standalone base classifiers.

Table 2. Results on dataset ar1

Classifiers	Accuracy	Recall	Precision	MCC	F1
qda	0.98	1	1	1	1
lgbm	0.97	1	0.943	0.941	0.97
sgd	0.909	1	0.846	0.832	0.908
cnb	0.833	1	0.75	0.707	0.829
etc	0.985	1	0.971	0.97	0.985
stack	0.985	1	0.971	0.97	0.985

Table 3. Results on dataset camel1.6

Classifiers	Accuracy	Recall	Precision	MCC	F1
qda	0.618	0.364	0.741	0.275	0.592
lgbm	0.844	0.829	0.855	0.689	0.844
sgd	0.614	0.855	0.577	0.26	0.59
cnb	0.586	0.754	0.564	0.182	0.573
etc	0.864	0.899	0.84	0.73	0.864
stack	0.866	0.864	0.868	0.732	0.866

5. STATISTICAL SIGNIFICANCE TEST PROCEDURE

Statistical testing is used to determine the statistical significance of the model. A pairwise T-test is performed to know the statistical difference between the proposed model and base classifiers. Two hypothesis are framed to determine the difference; Null and Alternate hypothesis.

Table 4. Results on dataset mw1

Classifiers	Accuracy	Recall	Precision	MCC	F1
qda	0.618	0.364	0.741	0.275	0.592
lgbm	0.844	0.829	0.855	0.689	0.844
sgd	0.596	0.268	0.782	0.256	0.548
cnb	0.586	0.754	0.564	0.182	0.573
etc	0.873	0.908	0.848	0.747	0.873
stack	0.86	0.851	0.866	0.719	0.86

Table 5. Results on dataset pc1

Classifiers	Accuracy	Recall	Precision	MCC	F1
qda	0.67	0.395	0.878	0.407	0.643
lgbm	0.956	0.964	0.949	0.913	0.956
sgd	0.794	0.994	0.711	0.642	0.786
cnb	0.65	0.466	0.738	0.324	0.638
etc	0.969	0.981	0.959	0.939	0.969
stack	0.968	0.974	0.962	0.935	0.968

Table 6. Results on dataset pc2

Classifiers	Accuracy	Recall	Precision	MCC	F1
qda	0.67	0.395	0.878	0.407	0.643
lgbm	0.956	0.964	0.949	0.913	0.956
sgd	0.794	0.994	0.711	0.642	0.786
cnb	0.65	0.466	0.738	0.324	0.638
etc	0.969	0.981	0.959	0.939	0.969
stack	0.968	0.974	0.962	0.935	0.968

Table 7. Results on dataset xerces

Classifiers	Accuracy	Recall	Precision	MCC	F1
qda	0.375	0.522	0.398	-0.257	0.361
lgbm	0.449	0.403	0.435	-0.105	0.447
sgd	0.485	0.925	0.488	-0.033	0.367
cnb	0.471	0.448	0.462	-0.06	0.47
etc	0.493	0.493	0.485	-0.015	0.493
stack	0.449	0.478	0.444	-0.102	0.448

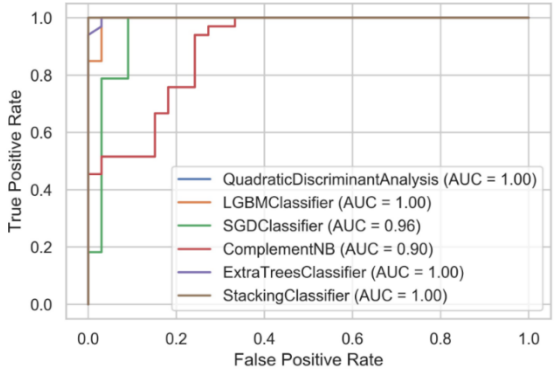

Figure 13. AUC values on dataset ar1

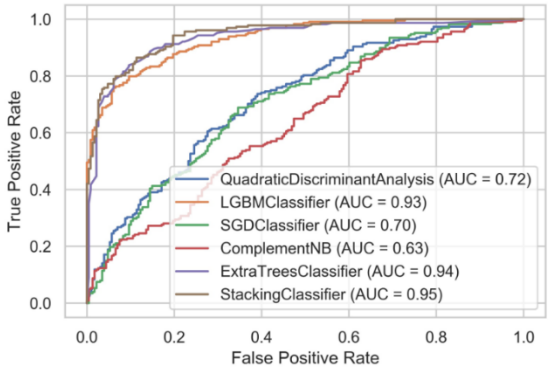

Figure 14. AUC values on dataset camel1.6

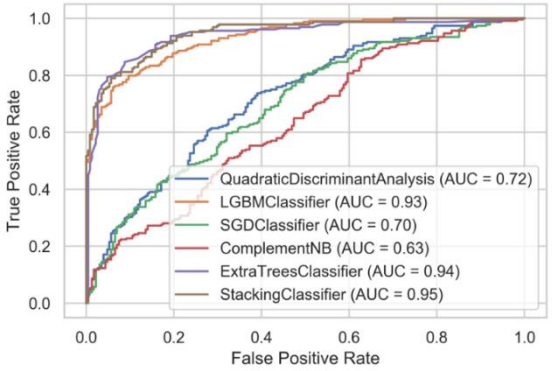

Figure 15. AUC values on dataset mw1

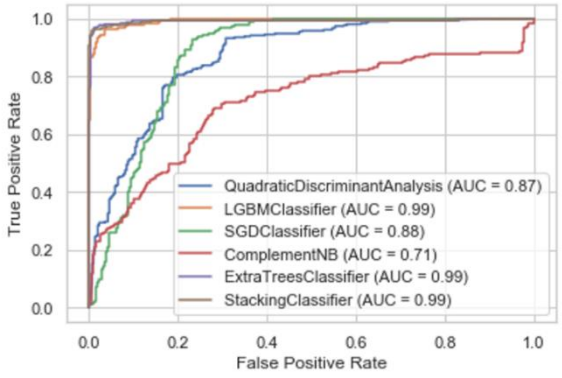

Figure 16. AUC values on dataset pc1

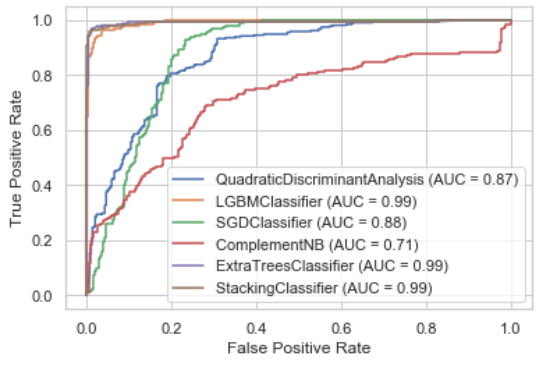

Figure 17. AUC values on dataset pc2

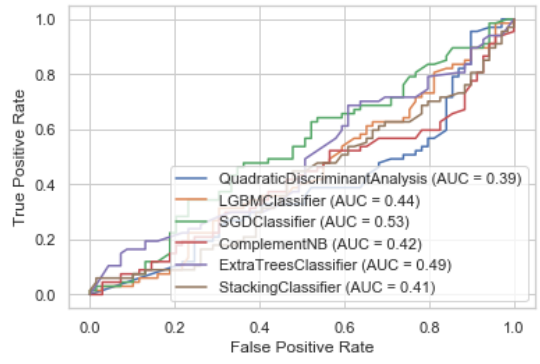

Figure 18. AUC values on dataset xerces

A) Null hypothesis H0: H0 signifies there is no performance difference between the proposed model and standalone base classifiers.
B) Alternate Hypothesis H1: H1 signifies there is a significant performance difference between the proposed model and standalone base classifiers. To find the statistical significance of the model, we use the "T-test and F-test procedure". Figure 19 shows the boxplot of the mean difference between the proposed model and individual base classifiers. The mean value of the proposed model is better than base classifiers. Table 8 shows the pairwise t-test results on all datasets. The bold values show the significant statistical difference between the proposed model and base classifiers. Table 9 shows the f-value based on the accuracy of all datasets. The f-value of 11.149 is higher so the difference is significant.

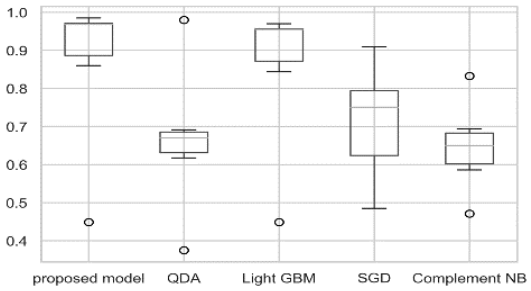

Figure 19. Mean scores of PM in comparison with B

Table 8. Pairwise T-test between PM and BC

Pairwise T-test based on Accuracy values		
Pairs	P-Value	Statistic
pair 1	0.012218	3.832349325
Pair 2	0.005551	4.656203156
pair 3	0.023349	3.224474649
pair 4	0.009919	4.040390789
pair 5	0.411282	-0.896057871

Table 9. F-test between proposed model and Base Classifier

F-test based on all performance metrics		
Metric	F-Value	Result
Accuracy	11.149	Significant

6. THREATS TO VALIDITY

In this research work, we have used real world dataset. Due to versatile nature of software it can be related to any domain and the performance of model is highly dependent on data. As a result, outcomes in these datasets may not be generalizable. Moreover, to determine the model's statistical significance we have used T-test and F-test. Others may use Friedman or Kruskal Wallis tests. It depends on the researcher's experimental purpose.

Another concern is the choice of classifiers in building heterogeneous fault prediction models. Other classifiers can be used to build the model and definitely the results will be different.

7. CONCLUSION AND FUTURE SCOPE

The main goal of this study is to implement the heterogeneous ensembling technique with a novel SHAP feature selection technique. The reason behind performing this study is predict minority faults that always remains undetected and becomes the primary cause of failure. To the best of my knowledge, all fault prediction studies implemented to date have used feature selection based on global interpretability. None of the studies focused on the concept of local interpretability. SHAP is the only feature selection technique that works on local interpretability. The intuition behind using heterogeneous classifiers is the unique property of each classifier in detecting different faults. So, we implemented a combined model that focuses on data quality as well as the application of heterogeneous classifiers. The results proved that, except for dataset Xerces, our proposed methodology outperforms in all datasets than existing frameworks.

This work can be extended by building same heterogeneous model with different sets of classifiers and results of this model can be used for comparison. Moreover, different feature selection and feature engineering techniques can be used and can be implemented on different datasets.

REFERENCES

[1] "IEEE Standard Glossary of Software Engineering Terminology", IEEE Std 610.12-1990, pp. 1-84, 31 Dec. 1990.
[2] M. Zhao, C. Wohlin, N. Ohlsson, M. Xie, "A comparison between software design and code metrics for the prediction of software fault content", Information and Software technology, Issue 14, Vol. 40, pp. 801-809, 1998.
[3] I.H. Laradji, M. Alshayeb, L. Ghouti, "Software defect prediction using ensemble learning on selected features", Information and Software Technology, Vol. 58, pp. 388-402, 2015.
[4] C. Pak, T.T. Wang, X.H. Su, "An empirical study on software defect prediction using over-sampling by SMOTE", International Journal of Software Engineering and Knowledge Engineering, Issue 6, Vol. 28, pp. 811-830, 2018.
[5] R. Shatnawi, "Improving software fault-prediction for imbalanced data", International conference on innovations in information technology (IIT), IEEE, Abu Dhabi, UAE, 20 March 2012.
[6] C. Catal, B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem", Information Sciences, Issue 8, Vol. 179, pp. 1040-1058, 29 March 2009.
[7] K. Gao, T. M. Khoshgoftaar, H. Wang, N. Seliya, "Choosing software metrics for defect prediction: an investigation on feature selection techniques", Software: Practice and Experience, Iss. 5, Vol. 41, pp. 579-606, 2011.
[8] D. Dataman, "Explain Your Model with the SHAP Values", Towards Data Science, Sept. 2019, https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d.
[9] D. Bowes, T. Hall, J. Petric, "Software defect prediction: do different classifiers find the same defects?", Software Quality Journal, Iss. 2, Vol. 26, pp. 525-552, 2018.
[10] T. Li, L. Yang, K. Li, J. Zhai, "An Intelligent Fusion Algorithm and Its Application Based on Subgroup Migration and Adaptive Boosting", Symmetry, Issue 4, Vol. 13, pp. 569, 2021.

[11] M. Mabayoje, A. Balogun, A. Bajeh, B. Musa, "Software Defect Prediction: Effect of Feature Selection and Ensemble Methods", UNILORIN Institutional Repository, September, 2018.
[12] P.S. Kumar, H.S. Behera, J. Nayak, B. Naik, "Bootstrap aggregation ensemble learning-based reliable approach for software defect prediction by using characterized code feature", Innovations in Systems and Software Engineering, pp. 1-25, 2021.
[13] M. Mangla, N. Sharma, S.N. Mohanty, "A sequential ensemble model for software fault prediction", Innovations in Systems and Software Engineering, pp. 1-8, 2021.
[14] I. Kaur, A. Kaur, "Comparative analysis of software fault prediction using various categories of classifiers", International Journal of System Assurance Engineering and Management, Springer, The Society for Reliability, Engineering Quality and Operations Management (SREQOM), India, and Division of Operation and Maintenance, Lulea University of Technology, Sweden, Iss. 3, Vol. 12, pp. 520-535, June 2021.
[15] A. Iqbal, S. Aftab, "A Classification Framework for Software Defect Prediction Using Multi-Filter Feature Selection Technique and MLP", Int. Journal of Modern Education & Computer Science, Iss. 1, Vol. 12, 2020.
[16] P. Kumari, A. Chatterjee, D.P. Mohapatra, "Smart Innovation, Systems and Technologies", Intelligent and Cloud Computing: Proceedings of ICICC, Vol. 194, pp. 95-103, 2019.
[17] S.J. Sayyad, "Promise software engineering repository", http://promise.site.uottawa.ca/SERepository/
[18] J. Goyal, B. Kishan, "TLHEL: Two Layer Heterogeneous Ensemble Learning for Prediction of Software Faults", International Journal of Engineering Trends and Technology, Issue 4, Vol. 69, pp. 16-20, 2021.
[19] J. Goyal, B. Kishan, "Empirical evaluation to identify the effectiveness of Ensemble technique for Prediction of Software Fault", International Journal of Advanced Research in Engineering and Technology (IJARET), Issue 12, Vol. 11, pp. 885-94, December 2020.
[20] O. Alan, C. Catal, "An outlier detection algorithm based on object-oriented metrics thresholds", The 24th IEEE International Symposium on Computer and Information Sciences, 2009.
[21] O. Alan, C. Catal, "Thresholds based outlier detection approach for mining class outliers: An empirical case study on software measurement datasets", Expert Systems with Applications, Vol. 38, No. 4, pp. 3440-3445, 2011.
[22] R. Aurelius A. Viadinugroho, "Imbalanced Classification in Python: SMOTE-Tomek Links Method", Towards Data Science, 18 April 2021, https://towardsdatascience.com/imbalanced-classification-in-python-smote-tomek-links-method-6e48dfe69bbc.
[23] H. Wei, C. Hu, S. Chen, Y. Xue, Q. Zhang, "Establishing a software defect prediction model via effective dimension reduction", Information Sciences, Vol. 477, pp. 399-409, March 2019.
[24] M.A. Hall, "Correlation-based feature selection of discrete and numeric class machine learning", Hamilton, New Zealand: University of Waikato, Department of Computer Science, 2000.
[25] S. Srivastava, M.R. Gupta, B.A. Frigyik, "Bayesian quadratic discriminant analysis", Journal of Machine Learning Research, Issue 6, Vol. 8, 2007.
[26] M.H. Abdurrahman, B. Irawan, C. Setianingsih, "A Review of Light Gradient Boosting Machine Method for Hate Speech Classification on Twitter", The 2nd IEEE International Conference on Electrical, Control and Instrumentation Engineering (ICECIE), Kuala Lumpur, Malaysia, 28 Nov. 2020.
[27] L. Bottou, "Large-scale machine learning with stochastic gradient descent", Proceedings of COMPSTAT', pp. 177-186, Physica-Verlag HD, 2010.
[28] P. Horbonos, "Comparing a variety of Naive Bayes classification algorithms", Towards Data Science, 15 February 2020.
[29] P. Geurts, D. Ernst, L. Wehenkel, "Extremely randomized trees", Machine learning, Issue 1, Vol. 63, pp. 3-42, 2006.
[30] S. Le Cessie, J.C. Van Houwelingen, "Ridge estimators in logistic regression", Journal of the Royal Statistical Society: Series C (Applied Statistics), Issue 1, Vol. 41, pp. 191-201, 1992.
[31] Z. Li, X. Y. Jing, X. Zhu, "Progress on approaches to software defect prediction", IET Software, The Institution of Engineering and Technology, Issue 3, Vol. 12, pp. 161-175, 2018.

BIOGRAPHIES

Jyoti Goyal Jain received her Bachelor degree in Computer Applications from Vaish Mahilla Mahavidyalya Womens College, Rohtak, India in 2005 and Master degree in Business Administration from Maharshi Dayanand University, Rohtak, India in 2007. She has passed her Masters also in Computer Applications from the same university in 2010 and Masters in Philosophy from Singhania University, Jhunjhunu, India in 2012. Currently, she is pursuing Ph.D. from Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, India. She has more than 9 publications in national and international journals and conference proceedings. Her research interests include machine learning, natural language processing, data mining, artificial intelligence and software engineering. Presently. She is working as an Assistant Professor in Hindu Institute of Management and Technology, Rohtak, India. She has 10 years of teaching experience.

Bal Kishan received his Ph.D. degree in Computer Science from Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, India. He is currently working as an Assistant Professor in Department of Computer Science and Applications of the same university. His research interests include soft computing, artificial intelligence, data mining, software engineering.